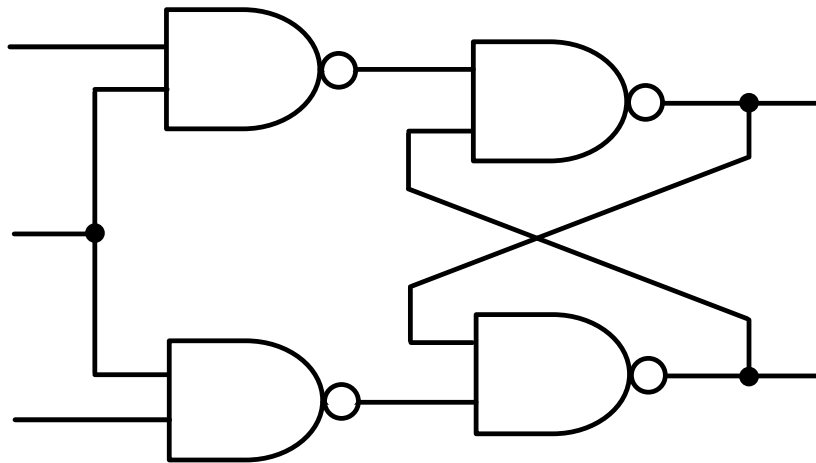


І. А. Дичка, В. П. Тарасенко, М. В. Онай

ОСНОВИ ПРИКЛАДНОЇ ТЕОРІЇ ЦИФРОВИХ АВТОМАТІВ



Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

І. А. Дичка, В. П. Тарасенко, М. В. Онай

Основи прикладної теорії цифрових автоматів

Підручник

*Рекомендовано Вченою радою КПІ ім. Ігоря Сікорського
як підручник для студентів, які навчаються за спеціальностями
121 «Інженерія програмного забезпечення», 123 «Комп'ютерна інженерія»*

Київ
КПІ ім. Ігоря Сікорського
2019

УДК 004.31 (075.8)

О49

*Рекомендовано Вченою радою КПІ ім. Ігоря Сікорського
(Протокол № 6 від 27.05.2019 р.)*

Рецензенти:

С. Д. Погорілий, д-р техн. наук, проф.,
Київський національний університет імені Тараса Шевченка,

А. О. Мельник, д-р техн. наук, проф.,
Національний університет “Львівська політехніка”.

Відповідальний редактор

Є. С. Сулема, канд. техн. наук, доц.,
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”.

Дичка І.А.

О49 Основи прикладної теорії цифрових автоматів : підручник / І. А. Дичка,
В. П. Тарасенко, М. В. Онай. – Київ : КПІ ім. Ігоря Сікорського, Вид-во
“Політехніка”, 2019. – 508 с.

ISBN 978-966-622-949-9

Подано перший модуль (семестр) двосеместрового курсу “Комп’ютерна логіка”, передбачений освітніми програмами та навчальними планами для студентів першого (бакалаврського) рівня освіти за спеціальностями “Інженерія програмного забезпечення”, “Комп’ютерна інженерія”.

Висвітлено прикладні аспекти теорії цифрових автоматів: аналіз та синтез комбінаційних схем, схем з пам’яттю, методи синтезу керуючих цифрових автоматів; запропоновано методики синтезу типових вузлів цифрової обчислювальної техніки (на прикладах розв’язування задач).

Для студентів, аспірантів, викладачів, а також для фахівців, які працюють у галузі проектування цифрових систем.

УДК 004.31

ISBN 978-966-622-949-9

© І. А. Дичка, В. П. Тарасенко, М. В. Онай, 2019

© КПІ ім. Ігоря Сікорського (ФПМ), 2019

ЗМІСТ

ПЕРЕДМОВА.....	5
ВСТУП.....	6
1. АПАРАТНА РЕАЛІЗАЦІЯ ПЕРЕМІКАЛЬНИХ ФУНКЦІЙ.....	9
1.1. Подання інформації в цифровій обчислювальній техніці....	9
1.2. Перемікальні функції та логічні схеми.....	18
1.3. Фізична реалізація логічних елементів.....	33
1.4. Принцип суперпозиції в теорії перемікальних функцій.....	48
1.5. Оцінювання логічних схем.....	51
2. АЛГЕБРИ ПЕРЕМІКАЛЬНИХ ФУНКЦІЙ.....	55
2.1. Алгебра Буля.....	55
2.2. Алгебра Шеффера.....	73
2.3. Алгебра Пірса.....	80
2.4. Алгебра Жегалкіна.....	87
2.5. Декомпозиція перемікальних функцій.....	94
2.6. Розширена алгебра Буля-Шеффера-Пірса.....	110
2.7. Принцип двоїстості в теорії перемікальних функцій.....	120
2.8. Функціональна повнота систем перемікальних функцій...	128
3. МІНІМІЗАЦІЯ ПЕРЕМІКАЛЬНИХ ФУНКЦІЙ.....	138
3.1. Проблема мінімізації перемікальних функцій.....	138
3.2. Метод мінімізації Квайна.....	142
3.3. Метод мінімізації Квайна–Мак–Класкі.....	152
3.4. Спосіб Петріка знаходження тупикових ДНФ перемікальних функцій.....	161
3.5. Графічні методи мінімізації перемікальних функцій.....	169
3.6. Мінімізація неповністю визначених перемікальних функцій.....	186
3.7. Мінімізація систем перемікальних функцій.....	195
4. АНАЛІЗ ТА СИНТЕЗ КОМБІНАЦІЙНИХ СХЕМ.....	208
4.1. Аналіз комбінаційних схем.....	208
4.2. Синтез комбінаційних схем у заданому елементному базисі.....	211
5. ФУНКЦІОНАЛЬНІ ВУЗЛИ КОМБІНАЦІЙНОГО ТИПУ В ЦИФРОВИХ ОБЧИСЛЮВАЛЬНИХ ЗАСОБАХ.....	223
5.1. Дешифратори.....	223
5.2. Шифратори.....	239
5.3. Перетворювачі кодів.....	255

5.4.	Мультиплексори.....	262
5.5.	Демультиплексори.....	279
5.6.	Схеми порівняння кодів.....	284
5.7.	Комбінаційні суматори.....	289
5.8.	Схема формування ознаки парності.....	295
5.9.	Програмовні логічні матриці.....	297
6.	БІСТАБІЛЬНІ СХЕМИ.....	302
6.1.	Аналіз послідовнісних схем.....	302
6.2.	Бістабільна схема на елементах І-НЕ.....	307
6.3.	Бістабільна схема на елементах АБО-НЕ	313
7.	СИНТЕЗ ТРИГЕРІВ.....	320
7.1.	Класифікація тригерів.....	320
7.2.	Асинхронні тригери.....	327
7.3.	Синхронні тригери, керовані рівнем синхросигналу.....	332
7.4.	Тригери з внутрішньою затримкою.....	337
7.5.	Методика синтезу синхронних тригерів.....	345
7.6.	Умовні графічні позначення тригерів.....	356
7.7.	Взаємозамінюваність тригерів.....	360
8.	ФУНКЦІОНАЛЬНІ ВУЗЛИ НА ОСНОВІ ТРИГЕРІВ.....	368
8.1.	Виконання мікрооперацій на регістрах.....	368
8.2.	Синтез синхронних регістрів.....	382
8.3.	Лічильники.....	395
9.	ФУНКЦІОНАЛЬНІ ВУЗЛИ НА ОСНОВІ РЕГІСТРІВ.....	415
9.1.	Накопичувальні суматори.....	415
9.2.	Регістрові запам'ятовувальні пристрої.....	416
9.3.	Стеки.....	418
9.4.	Взаємозв'язок апаратних засобів цифрової обчислювальної техніки.....	425
10.	СИНТЕЗ КЕРУЮЧИХ ЦИФРОВИХ АВТОМАТІВ.....	429
10.1.	Принцип мікропрограмного керування.....	429
10.2.	Керуючі автомати з жорсткою логікою.....	432
10.3.	Канонічний метод структурного синтезу керуючих автоматів з жорсткою логікою.....	435
10.4.	Керуючі автомати з програмовною логікою.....	451
	ВИСНОВКИ.....	464
	СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ.....	468
	ІМЕННИЙ ПОКАЖЧИК.....	470
	ПРЕДМЕТНИЙ ПОКАЖЧИК	472
	ГЛОСАРІЙ.....	480

ПЕРЕДМОВА

Сучасний етап розвитку людства характеризується всеосяжним проникненням інформаційних технологій у всі сфери діяльності людини – науку, виробництво, медицину, освіту, військову галузь, і навіть у побут. Технічно розвинені країни вирішують надзвичайно важливу і складну проблему інформатизації суспільного життя. Глобальна інформатизація суспільства потребує не лише створення передової індустрії електронно-обчислювальних засобів, але й відповідного кадрового забезпечення галузі – підготовки висококваліфікованих фахівців, здатних розробляти нові зразки комп'ютерної техніки та створювати сучасне програмне забезпечення.

У світовому освітньому просторі підготовку фахівців, що мають забезпечувати вирішення проблеми інформатизації, здійснюють за п'ятьма освітніми напрямками: *комп'ютерні науки* (computer science), *комп'ютерна інженерія* (computer engineering), *інженерія програмного забезпечення* (software engineering), *інформаційні системи* (information systems), *інформаційні технології* (information technology). Разом ці напрями утворюють галузь знань, яку прийнято називати *комп'ютингом* (computing). Саме фахівцям з комп'ютингу належить забезпечувати сталий розвиток інформаційної компоненти цивілізаційного процесу.

Одним з основних інструментальних засобів при вирішенні проблеми інформатизації є комп'ютерні системи. Комп'ютер (електронна обчислювальна машина – ЕОМ) – це складна технічна система, призначена для *автоматизованого* оброблення цифрової інформації. Організація комп'ютера ґрунтується на певних принципах, головним з яких є *принцип програмного керування*. При проведенні дослідних, проектних, конструкторсько-технологічних робіт, пов'язаних з удосконаленням обчислювальних засобів, а також при вивченні комп'ютерних систем доцільно застосовувати ієрархічний підхід до питання організації та функціонування комп'ютера. Найпоширенішим є підхід, коли при синтезі та аналізі комп'ютера використовують чотири рівні ієрархії: віртуальний, функціональний, логічний та фізичний. Фізичний рівень ієрархії комп'ютера розглядають у навчальній дисципліні «Комп'ютерна електроніка». Функціональний та логічний рівень ієрархії є предметом розгляду в навчальній дисципліні «Комп'ютерна логіка».

Вивчення «Комп'ютерної логіки» ґрунтується на навчальних дисциплінах «Комп'ютерна дискретна математика» та «Теорія інформації». «Комп'ютерна логіка» як навчальна дисципліна традиційно складається з двох частин (модулів) – прикладної теорії цифрових автоматів та комп'ютерної арифметики. У підручнику подаються основи прикладної теорії цифрових автоматів. Підручник призначений для студентів, що навчаються за спеціальностями «Інженерія програмного забезпечення» та «Комп'ютерна інженерія».

ВСТУП

Науково-технічний прогрес, розвиток галузей народного господарства тісно пов'язані з використанням комп'ютерів та комп'ютерних мереж. Успішне вирішення наукових, технічних, технологічних проблем та виробничих завдань значною мірою залежить від рівня розвитку електронно-обчислювальних засобів. Тому комп'ютерна техніка є важливим фактором розвитку продуктивних сил суспільства.

Становлення науково-інженерної думки у галузі створення комп'ютера як обчислювального засобу тривало декілька століть. У 1642 році французький вчений Блез Паскаль (1623 – 1662) сконструював механічний пристрій, який забезпечував виконання операцій додавання та віднімання. Пристрій застосовувався для виконання розрахунків під час стягування податків у Франції. Німецький математик Готфрід-Вільгельм Лейбніц (1646 – 1716) у 1694 р. удосконалив пристрій Паскаля. Пристрій Лейбніца забезпечував виконання чотирьох основних арифметичних операцій, а також піднесення до степеня та добування квадратного кореня. У 1820 р. французький інженер Тома де Кольмар (1785 – 1870) розробив арифмометр, який масово випускався та застосовувався в комерційних операціях.

У 1833 р. професор Кембріджського університету Чарльз Беббідж (1791 – 1871) дійшов висновку про можливість побудови *машини*, яка могла б виконувати *будь-які обчислення*, що задаються оператором, а не лише обчислення одного спеціального виду. Він розробив проект механічної *універсальної цифрової машини* з програмним керуванням, яку назвав *аналітичною машиною*. На думку Беббіджа машина мала складатися з п'яти пристроїв: арифметичного пристрою (Беббідж називав його «млином»), пристрою пам'яті («складу», за проектом «склад» мав складатися з 50000 коліщат, кожне коліща мало бути з 10-ма зубцями), пристрою керування, пристрою введення та пристрою виведення. Для введення чисел в машину та керування ходом обчислень пропонувалось використовувати перфоровані картки (перфокарти), з'єднані в суцільну стрічку (перфострічку). На жаль, аналітична машина не була реалізована, вона так і лишилась проектом на папері. Проте ідеї, висловлені Беббіджем, стали надзвичайно плідними для подальшого розвитку теорії побудови комп'ютера. Заслуга Ч. Беббіджа полягає в тому, що він вперше запропонував структуру обчислювальної машини (вона має складатися з 5-ти пристроїв), а також вперше висловив *ідею програмного керування* комп'ютером (принцип програмного керування).

Після проекту аналітичної машини Ч. Беббіджа майже понад століття теорія побудови комп'ютера не розвивалась.

Створення перших універсальних комп'ютерів припадає на період 40-50-х років минулого століття.

1941 р. (Німеччина) – перша в світі *електромеханічна* обчислювальна машина (на електромагнітних реле) з використанням двійкової системи числення (розробник – Конрад Цузе).

1944 р. (США) – обчислювальна машина «Марк1» на електромагнітних реле з програмним керуванням; особливості – електромеханічна, 10-кова система числення.

1946 р. (США) – перша *електронна* обчислювальна машина «ЕНІАК» (розробники – Джон Мочлі, Преспер Еккерт) на електронних лампах, з програмним керуванням; особливості – 10-кова система числення, команди програми набирались за допомогою механічних перемикачів.

1948 р. (Велика Британія) – ЕОМ «Бєбі», м. Манчестер (розробники – Алан Тьюрінг, Фредерік Вільямс, Том Кілбурн); для запису даних і програми обчислень використовувалась електронно-променева трубка, вперше доведено можливість зберігання чисел та програми у загальній пам'яті машини.

1949 р. (Велика Британія) – ЕОМ «ЕДСАК», м. Кембрідж (розробник – Моріс Вілкс), передбачала зберігання програми обчислень в пам'яті машини.

1952 р. (США) – ЕОМ «ЕДВАК» (розробники – Джон Мочлі, Преспер Еккерт, Джон фон Нейман); особливість – зберігання програми в оперативній пам'яті машини.

1952 р. (Україна) – ЕОМ «МЭСМ» – малая электронная счетная машина, м. Київ, Інститут електромеханіки АН УРСР (розробник – Сергій Олексійович Лебедев); особливість – на електронних лампах.

1953 р. (США) – ЕОМ «ІАК» (розробник – Джон фон Нейман).

Отже, творцями перших універсальних ЕОМ (комп'ютерів) були: Конрад Цузе (Німеччина); Джон Мочлі, Преспер Еккерт, Джон фон Нейман (США); Алан Тьюрінг, Фредерік Вільямс, Том Кільбурн, Моріс Вілкс (Велика Британія); С.О. Лебедев (Україна).

Саме в 40-50-х роках ХХ століття були остаточно сформовані основні принципи побудови комп'ютера:

- 1) *структурний принцип* (принцип 5-ти пристроїв) (сформульований Ч. Беббіджем) – до складу комп'ютера мають входити 5 пристроїв: арифметико-логічний пристрій, оперативний запам'ятовувальний пристрій, пристрій введення даних, пристрій виведення даних (результатів), пристрій керування (в сучасній інтерпретації арифметико-логічний пристрій разом з пристроєм керування називають процесором);
- 2) *принцип програмного керування* (сформульований Ч. Беббіджем) – обчислення здійснюються *автоматично* на основі

програми; комп'ютер має виконувати: арифметичні та логічні операції, операцію порівняння, умовні та безумовні переходи в програмі;

- 3) *принцип зберігання програми в оперативній пам'яті комп'ютера* (сформульований Дж. фон Нейманом та незалежно від нього – С.О. Лебедевим) – програма обчислень кодується і зберігається в оперативній пам'яті так само, як числа;
- 4) *принцип числового розв'язування задач* – для обчислень використовуються *числові (чисельні) методи* розв'язування задач;
- 5) *принцип двійковості* – в комп'ютері на фізичному рівні має використовуватись двійкова система числення.

Український вчений С.О. Лебедев (1902 – 1974) незалежно від зарубіжних учених розробив принципи побудови ЕОМ (комп'ютера) з програмним керуванням і зберіганням програми в пам'яті машини, які були реалізовані в ЕОМ «МЭСМ».

Саме зберігання програми в оперативній пам'яті машини стало завершальним кроком на шляху створення теорії побудови ЕОМ. Авторство формулювання принципу зберігання програми обчислень в оперативній пам'яті комп'ютера приписують Дж. фон Нейману і С.О. Лебедеву.

С.О. Лебедев також розробив методику виконання операцій у двійковій системі числення.

Особливостями принципу зберігання програми в оперативній пам'яті комп'ютера є:

- 1) для зберігання програми обчислень і даних (початкових, проміжних, кінцевих) використовується одне й те саме фізичне середовище – ОЗП (оперативний запам'ятовувальний пристрій);
- 2) над окремими компонентами коду програми можна виконувати операції як над числами (в тому числі й у процесі виконання програми).

Перелічені вище *п'ять основних принципів* побудови комп'ютера науково обґрунтував і оприлюднив Джон фон Нейман – американський вчений угорського походження. Як видатний математик він зумів науково узагальнити досвід, набутий в ході розроблення перших комп'ютерів, і виклав його у вигляді основ теорії побудови комп'ютера в науковій статті.

Якщо комп'ютер побудовано на основі перелічених вище п'яти принципів, то вважається що він має *фон-нейманівську архітектуру*.

Опанування особливостей фон-нейманівської архітектури – основної архітектури сучасних комп'ютерів загального застосування, слід розпочинати з вивчення логічних основ побудови комп'ютера – прикладної теорії цифрових автоматів.

1. АПАРАТНА РЕАЛІЗАЦІЯ ПЕРЕМИКАЛЬНИХ ФУНКЦІЙ

Теорія цифрових автоматів ґрунтується на властивостях перемикальних функцій та особливостях їх апаратної реалізації на основі логічних елементів, які фізично є транзисторними структурами.

У даному розділі розглядається взаємозв'язок між аналітично-табличними формами задання перемикальних функцій та їх фізичною реалізацією у вигляді логічних схем, які є сукупністю відповідним чином з'єднаних між собою логічних елементів.

1.1. Подання інформації в цифровій обчислювальній техніці

Інформація в комп'ютері (в ЕОМ – електронній обчислювальній машині) подається у вигляді значень фізичних величин. Такими фізичними величинами можуть бути: електрична напруга, електричний струм, електричний заряд, напруженість магнітного поля, інтенсивність світлового потоку.

Фізичні величини мають форму сигналів. *Сигнал* – це процес, що характеризує зміну фізичної величини в часі. Якщо сигнал на заданому часовому інтервалі набуває нескінченну кількість значень, то його називають *неперервним (аналоговим)*. Якщо сигнал на заданому часовому інтервалі набуває скінченну (обмежену) кількість значень, то його називають *дискретним*.

Відповідно, за видом оброблюваної інформації ЕОМ (комп'ютери) бувають *аналоговими, цифровими, гібридними*. Аналогові ЕОМ оперують з інформацією, поданою у вигляді фізичних величин, що неперервно змінюються (аналогових величин). Аналогові ЕОМ бувають пневматичними, гідравлічними, електромеханічними, електронними.

Цифрові ЕОМ оперують з інформацією, поданою у вигляді дискретних (цифрових) значень фізичних величин.

У гібридних ЕОМ частина блоків оперує з аналоговою інформацією, решта – з дискретною інформацією.

Термін *цифрова величина* означає: всі стійкі значення (рівні) фізичної величини можна пронумерувати (оцифрувати). Прикладами дискретних величин є: *двійкова величина* – має два стійкі значення (рівні), *трійкова величина* – має три стійкі значення (рівні), *десятькова величина* – десять стійких значень.

Сучасні цифрові комп'ютери оперують з двійковими дискретними величинами. Двома рівнями (значеннями) дискретної величини в комп'ютері можуть бути: наявність електричного заряду – відсутність

заряду, висока (вища) напруга – низька (нижча) напруга, висока напруженість – низька напруженість магнітного поля, велика (більша) сила струму – мала (менша) сила струму, висока інтенсивність – низька інтенсивність світлового потоку.

Використання дискретних величин з двома стійкими значеннями («0», «1») дає наступні переваги: високу надійність, високу швидкодію процесів передавання, зберігання та перетворення інформації, простоту апаратної реалізації (в комп'ютері достатньо мати фізичні елементи з двома стійкими станами).

У цифровому комп'ютері фізичні сигнали за своєю природою є неперервними. Фізичний сигнал характеризується амплітудою (наприклад, напругою U), що неперервно змінюється за часом (рис. 1.1а). Такий сигнал (Q) математично можна описати неперервною функцією, $Q = f(U, t)$.

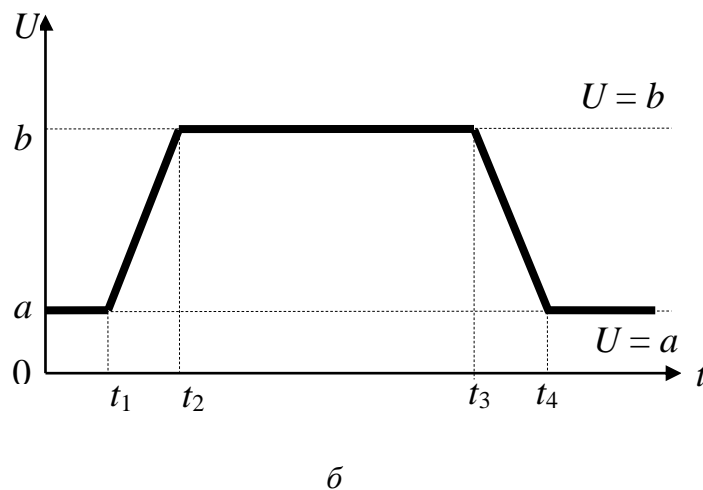
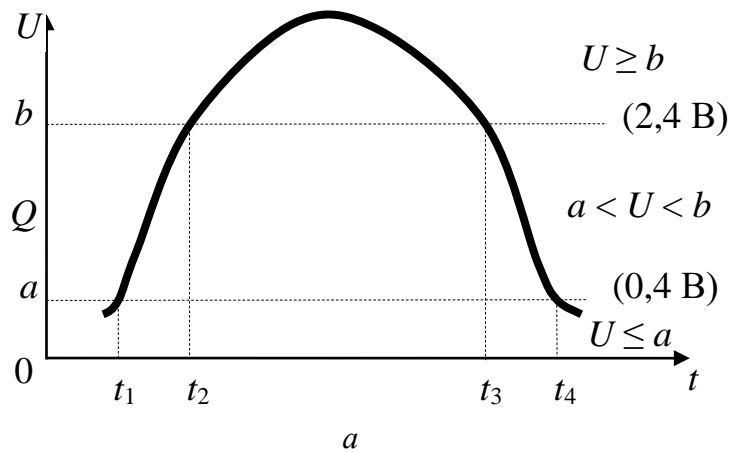


Рис. 1.1. Квантування фізичного сигналу за рівнем:
 a – фізичний сигнал; b – квантований за рівнем сигнал

Для встановлення відповідності між фізичним (неперервним за природою) сигналом і логічним (двійковим) сигналом (штучно створеним)

застосовують процес дискретизації фізичного сигналу, який включає три процедури: квантування сигналу за рівнем (амплітудою), кодування квантованого за рівнем сигналу, квантування сигналу за часом.

Квантування фізичного сигналу за рівнем виконують в такий спосіб. З практичних міркувань призначають два рівні (значення) a і b сигналу, які добре електрично розрізняються, $a < b$. Виділяють три області: $U \geq b$, $U \leq a$, $a < U < b$. Области $U \geq b$ та $U \leq a$ відповідають двом стійким значенням сигналу, в області $a < U < b$ значення сигналу не визначено. Результатом цієї процедури є перетворення (перехід) неперервного фізичного сигналу в фізичний сигнал, що має два стійкі рівні (значення).

Далі встановлюють відповідність між фізичним сигналом, що має два стійкі рівні, і двійковим сигналом. Встановлення такої відповідності називають *кодуванням квантованого за рівнем сигналу*. Можливі два способи отримання (кодування) двійкового сигналу D : у системі високих потенціалів і в системі низьких потенціалів (рис. 1.2).

Функція кодування в системі високих потенціалів має вигляд

$$D = \begin{cases} 1, & \text{якщо } U \geq b, \\ 0, & \text{якщо } U \leq a. \end{cases}$$

У системі низьких потенціалів функцію кодування визначають як

$$D = \begin{cases} 0, & \text{якщо } U \geq b, \\ 1, & \text{якщо } U \leq a. \end{cases}$$

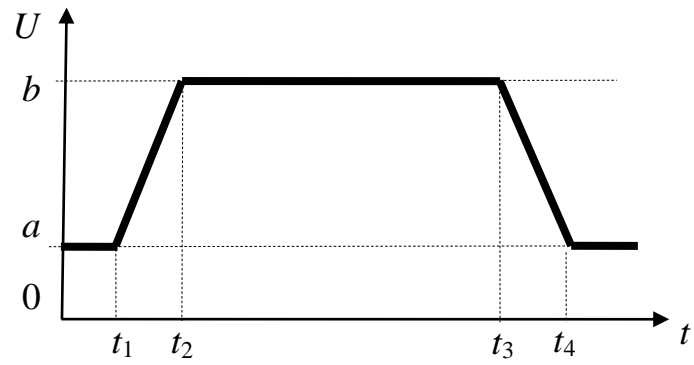
Результатом процедури кодування є отримання двійкового (логічного) сигналу, що має два *логічні значення*: логічний «0» (рівень логічного нуля) і логічна «1» (рівень логічної одиниці). Таким чином, двійковий сигнал – це дискретний сигнал, який може набувати лише двох значень, одне з яких ототожнюється з нулем, а інше – з одиницею.

Реальний двійковий сигнал характеризується певною сукупністю часових параметрів (рис. 1.3).

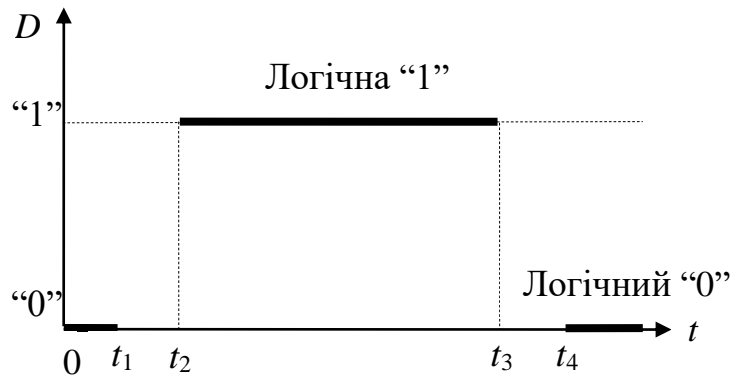
Тривалість W сигналу визначають як $W = t_4 - t_1$.

Процес переходу сигналу з рівня «0» на рівень «1» називають *переднім фронтом сигналу*. Відповідно, t_{Π}^{01} – тривалість (час) перемикавання з «0» в «1» (тривалість переднього фронту сигналу), визначають як $t_{\Pi}^{01} = t_2 - t_1$.

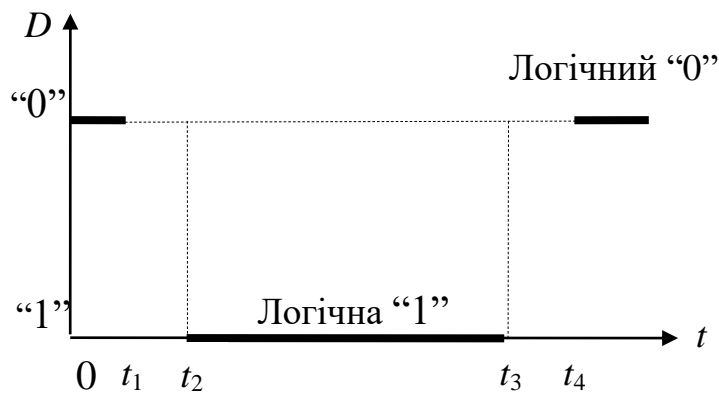
Процес переходу сигналу з рівня «1» на рівень «0» називають *заднім фронтом сигналу*. Відповідно, t_{Π}^{10} – тривалість (час) перемикавання сигналу з «1» в «0» (тривалість заднього фронту сигналу) визначають як $t_{\Pi}^{10} = t_4 - t_3$.



a



б



в

Рис. 1.2. Кодування квантованого за рівнем сигналу:
a – квантований за рівнем сигнал; *б* – логічний (двійковий) сигнал в системі високих потенціалів; *в* – логічний (двійковий) сигнал в системі низьких потенціалів

Узагальнений час перемикання ($t_{\text{п}}$) сигналу з одного рівня на інший визначають як $t_{\text{п}} = t_{\text{п}}^{\text{max}} = \max(t_{\text{п}}^{01}, t_{\text{п}}^{10})$ або як $t_{\text{п}} = (t_{\text{п}}^{01} + t_{\text{п}}^{10}) / 2$.

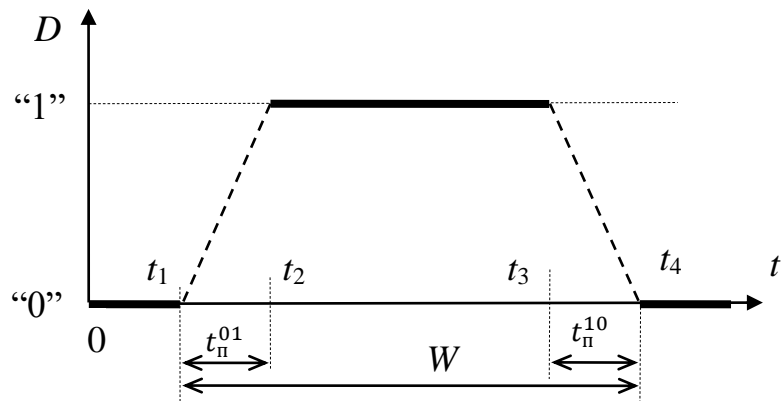


Рис. 1.3. Часові параметри логічного (двійкового) сигналу в системі високих потенціалів

Параметри W , t_n^{01} , t_n^{10} , t_n вимірюють в одиницях часу – секундах (с), мілісекундах (мс), мікросекундах (мкс), наносекундах (нс):

$$\begin{aligned} 1 \text{ мс} &= 10^{-3} \text{ с}, & 1 \text{ с} &= 1000 \text{ мс}, \\ 1 \text{ мкс} &= 10^{-6} \text{ с}, & 1 \text{ с} &= 1000000 \text{ мкс}, \\ 1 \text{ нс} &= 10^{-9} \text{ с}, & 1 \text{ с} &= 1000000000 \text{ нс}. \end{aligned}$$

Часові параметри сигналів у сучасних комп'ютерах подають в наносекундах, наприклад, $W = 20$ нс, $t_n^{01} = 2$ нс, $t_n^{10} = 3$ нс.

Після квантування сигналу за рівнем та кодування виконують процедуру *квантування сигналу за часом*, яка полягає в наступному.

Оскільки сигнал є процесом, перебіг якого відбувається в часі, то для того, щоб чітко й надійно фіксувати два можливі значення (рівні) сигналу – «0» або «1», необхідно позбавитись невизначеності сигналу на проміжках $t_1 - t_2$, $t_3 - t_4$ перемикання з одного рівня на інший (рис. 1.3). Для цього слід відмовитись від інтервалів t_1, t_2, \dots реального часу й запровадити натомість точки відліку (позначки) τ_1, τ_2, \dots *автоматного часу*, які виробляє генератор, що входить до складу процесора. Тобто пропонується розглядати сигнал як процес не в реальному, а в дискретному (автоматному, абстрактному) часі (рис. 1.4).

Оскільки генератор виробляє часові позначки τ_1, τ_2, \dots через однакові проміжки часу, що задаються його частотою, то одиницею вимірювання автоматного (абстрактного) часу є *такт* – період між двома сусідніми часовими позначками (відліками).

Тривалість T такту визначають як $T = \tau_{i+1} - \tau_i$.

Процедуру фіксування рівня («0» або «1») сигналу в точках відліку (позначках) τ_1, τ_2, \dots автоматного часу називають *квантуванням сигналу за часом*.

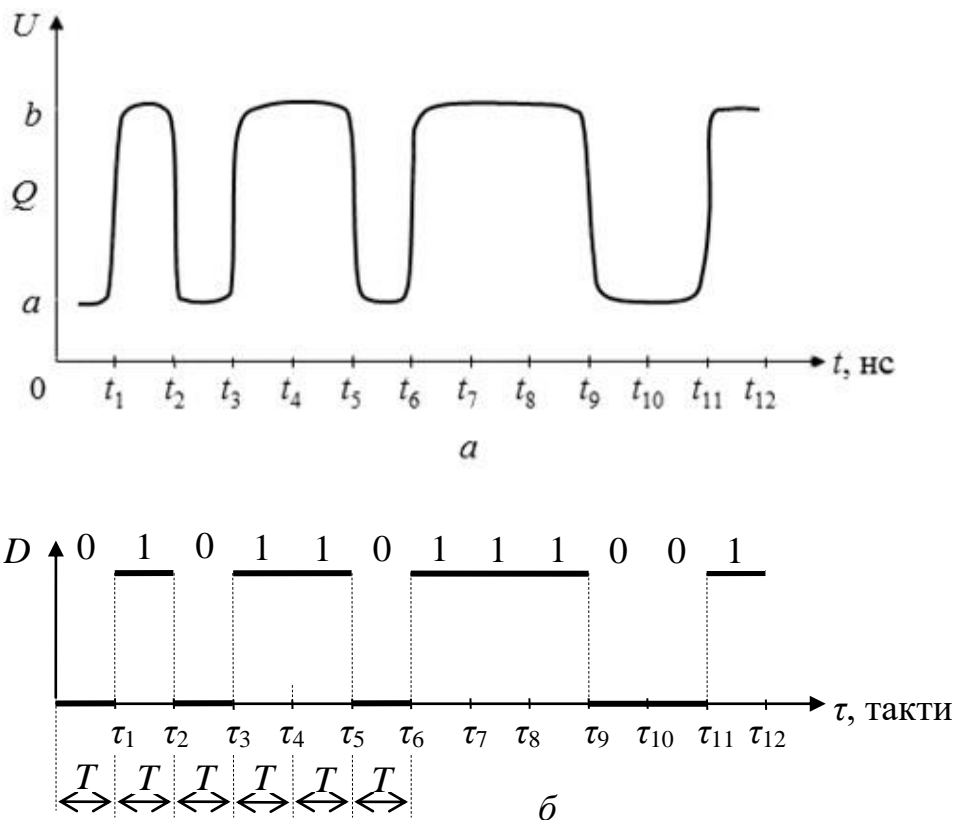


Рис. 1.4. Дискретизація фізичного сигналу:

a – фізичний сигнал;

$б$ – квантований за рівнем, кодований та квантований за часом сигнал

При такому підході значення сигналу в інтервалах між будь-якими сусідніми часовими позначками вважаються невизначеними; вони є визначеними лише строго в точках τ_i , тобто наприкінці кожного такту. Вважається, що всередині такту має місце перехідний процес.

Отже, умовою ідентифікації сигналів у часі є: тривалість T такту має перевищувати тривалість більшого з двох фронтів сигналу, тобто

$$T > t_n, \text{ де } t_n = \max(t_n^{01}, t_n^{10}).$$

Тривалість такту вибирають якнайменшою, але такою, щоб $T > t_n$.

Відповідно, генератор має працювати з такою частотою F , щоб виконувалась умова $T > t_n$.

Наприклад, якщо частота F генератора $F = 1$ МГц, то тривалість такту складає $T = 1$ мкс ($1/10^6 = 10^{-6} = 1$ мкс). Якщо $F = 100$ МГц, то тривалість такту дорівнює $T = 10$ нс ($1/(100 \cdot 10^6) = 1/10^8 = 10 \cdot 10^{-9} = 10$ нс).

Тривалості всіх сигналів мають бути кратними тривалості T такту: $W_j = nT$, де $n = 1, 2, 3, \dots$. Тобто, сигнал може діяти протягом одного або кількох тактів.

Таким чином, дискретизація фізичних сигналів дозволяє абстрагуватись від фізичної природи сигналів (неперервних у реальному

часі) і оперувати лише абстрактними (дискретними, логічними) сигналами в автоматному часі, що значно спрощує процеси аналізу та синтезу окремих вузлів і блоків комп'ютера, а також комп'ютерної системи в цілому на функціональному рівні.

З вищесказаного випливають два висновки.

1. На фізичному рівні в електронних колах вузлів та блоків комп'ютера поширюються неперервні (аналогові) сигнали. Дискретність є абстракцією, яка створюється штучно – для спрощення процесів оброблення сигналів (виконання над ними мікрооперацій – додавання, інвертування, зсуву тощо).
2. Тривалість обчислювальних процесів у комп'ютері вимірюють не в одиницях реального часу (секундах, мілісекундах тощо), а в одиницях автоматного часу – тактах.

Якщо певний об'єкт (елемент, сигнал) може перебувати у двох рівномірних станах (рівнях), то вважається, що він передає (подає, несе) один *біт* інформації. Біт є мінімальною структурною одиницею інформації. Назва біт (bit) є скороченням слів binary digit (двійкова цифра). Кількість інформації вимірюють у бітах.

Оскільки в моменти τ_i автоматного часу сигнал може набувати лише одне з двох рівномірних значень – «рівень логічного нуля» або «рівень логічної одиниці» (рис. 1.4), то в кожному такті сигнал подає 1 біт інформації.

Отже, числовим значенням біта може бути або 0, або 1. Тобто, одна двійкова цифра (один двійковий розряд) подає 1 біт інформації, а послідовність, наприклад, з чотирьох двійкових цифр (чотири двійкові розряди) подає 4 біти інформації. Біту, отже, відповідає двійкова (логічна) змінна x , що може набувати значення 0 або 1, $x \in \{0, 1\}$.

Наприклад, фізичний сигнал Q_1 (якому після дискретизації відповідає сигнал D_1) на проміжку тривалістю 12 тактів подає 12-розрядну двійкову послідовність 011011100101, а фізичний сигнал Q_2 (якому після дискретизації відповідає логічний сигнал D_2) – послідовність 010011010001 (рис. 1.5).

Якщо деякий пристрій (рис. 1.6), формує на виходах y_2, y_1 сигнали Q_2, Q_1 відповідно, то в дискретні (автоматні) моменти часу τ_1, τ_2, \dots , тобто в тактах 1, 2, ... на виходах y_2, y_1 отримують двійкові вектори, показані на рис. 1.6.

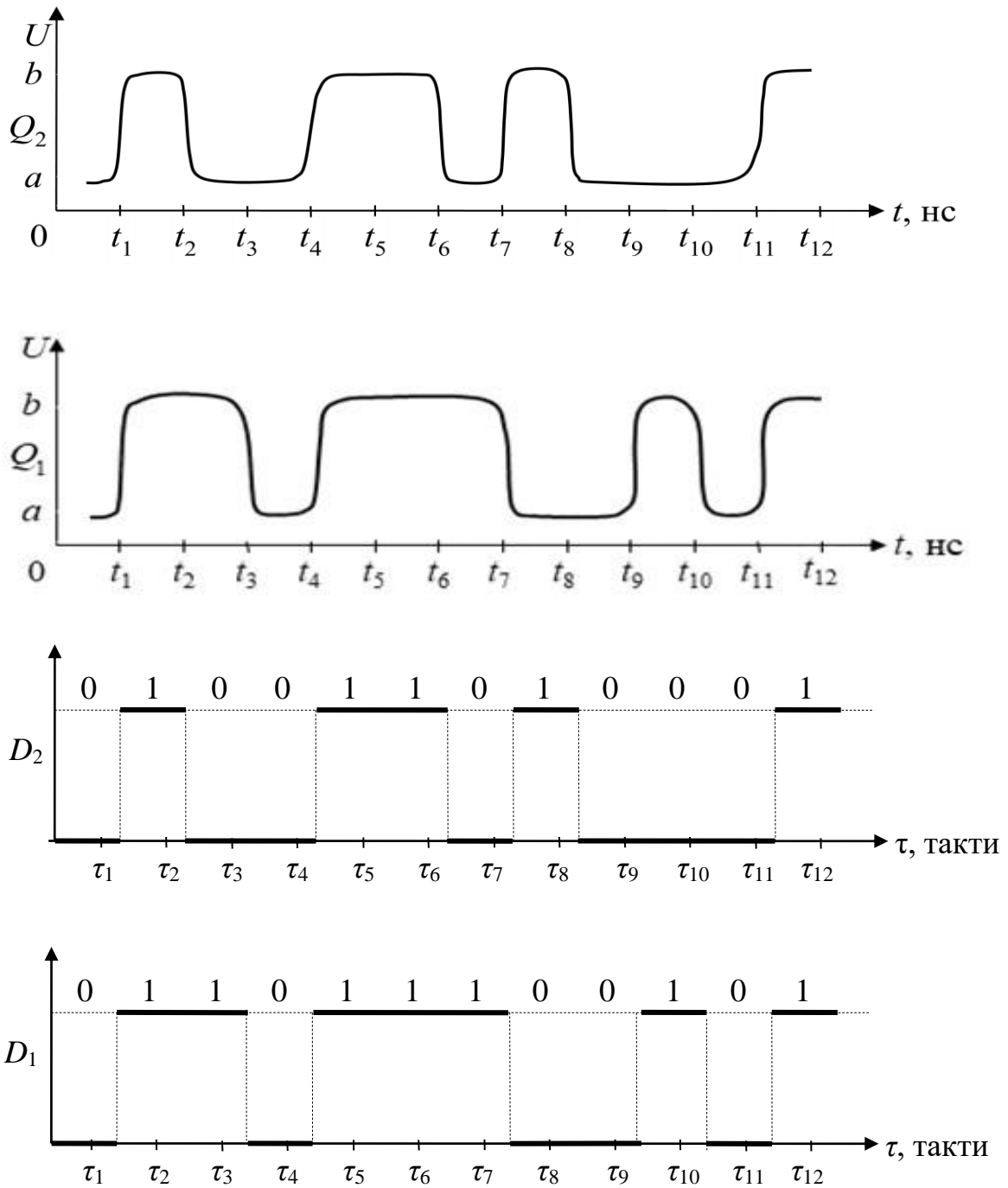


Рис. 1.5. Дискретизація фізичних сигналів Q_2, Q_1

Крім біта, для комп'ютера характерними є також такі *структурні одиниці інформації*: поле, байт, слово, масив.

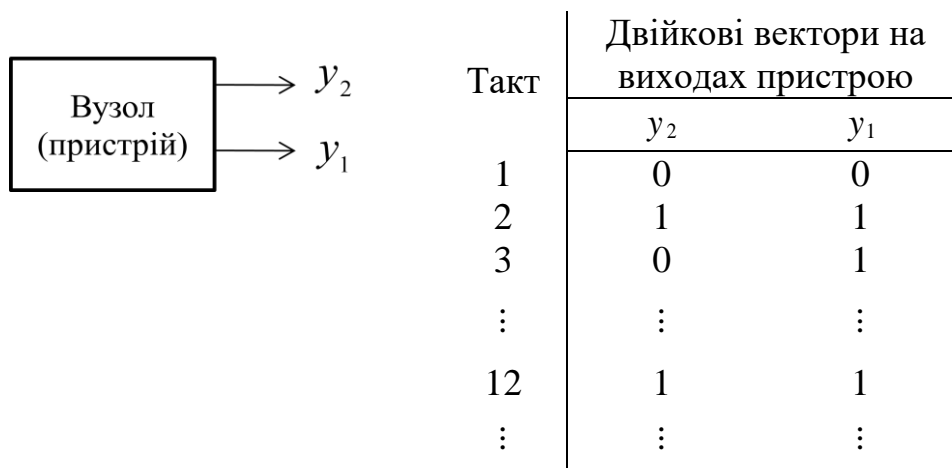


Рис. 1.6. Формування вузлом (пристроєм) вихідних двійкових послідовностей y_2 , y_1

Послідовність з двох і більше бітів, що має визначений зміст, називають *полем*. Поле, що складається з 8-ми бітів, називають *байтом*. Байт подає один символ комп'ютерного алфавіту.

Послідовність, що складається зі строго визначеної кількості байтів (бітів) і має певний зміст, називають *словом*.

Послідовність полів, байтів або слів, що мають певний зміст, утворює *масив*.

Запитання та завдання

1. Перелічіть принципи фон-нейманівської архітектури комп'ютера.
2. За рахунок чого досягається автоматизованість процесу оброблення інформації в комп'ютері?
3. Назвіть рівні опису комп'ютера.
4. За допомогою яких фізичних величин подається інформація в комп'ютері?
5. Який сигнал називають аналоговим, а який – дискретним?
6. Яку фізичну величину називають двійковою?
7. Сформулюйте мету дискретизації фізичних сигналів.
8. Які процедури слід застосовувати, щоб перейти від фізичного сигналу до двійкового?
9. Охарактеризуйте процедуру квантування сигналу за рівнем.
10. Якою є мета кодування квантованого за рівнем сигналу?
11. Запишіть функцію кодування сигналу в системі: а) високих потенціалів; б) низьких потенціалів.
12. Перелічіть часові параметри квантованого за рівнем сигналу в системі високих потенціалів.
13. Як визначають час перемикання сигналу з одного стану в інший?
14. Назвіть одиниці вимірювання тривалості сигналів у комп'ютері.

15. Порівняйте неперервний (реальний) час та дискретний (автоматний) час у комп'ютері.
16. Сформулюйте необхідну умову ідентифікації сигналу при його квантуванні за часом.
17. Якою є тривалість такту, якщо частота процесора становить $F = 50$ МГц?
18. В яких одиницях вимірюють тривалість обчислювальних процесів у комп'ютері?
19. Які переваги дає використання в комп'ютері дискретних величин з двома стійкими значеннями (рівнями)?
20. В яких одиницях вимірюють кількість інформації?
21. Перелічіть структурні одиниці інформації, що використовуються в комп'ютері.

1.2. Перемикальні функції та логічні схеми

На логічному та функціональному рівнях опису комп'ютер є системою, до складу якої входять відповідні функціональні вузли, блоки та пристрої. Будь-який функціональний вузол (схему, пристрій) можна вважати *перетворювачем інформації*, що має n входів та m виходів (рис. 1.7), на входи якого в моменти дискретного (автоматного) часу надходять вхідні (логічні) послідовності $x_n x_{n-1} \dots x_1$, а значення на кожному виході y_j ($j = 1, 2, \dots, m$) є функцією від вхідних двійкових послідовностей: $y_j = f_j(x_n, x_{n-1}, \dots, x_1)$. Функції y_1, y_2, \dots, y_m називають перемикальними.

Функція $y = f(x_n, x_{n-1}, \dots, x_1)$ називається *перемикальною* (логічною, булевою), якщо сама функція y і кожен з її аргументів x_i ($i = 1, 2, \dots, n$) можуть набувати лише двох значень: 0 або 1, тобто якщо $x_i, y \in \{0, 1\}$.

Отже, поведінку схеми (вузла, пристрою) можна описати *системою перемикальних функцій*:

$$\begin{cases} y_1 = f_1(x_n, x_{n-1}, \dots, x_1), \\ y_2 = f_2(x_n, x_{n-1}, \dots, x_1), \\ \vdots \\ y_m = f_m(x_n, x_{n-1}, \dots, x_1). \end{cases} \quad (1.1)$$

Перемикальну функцію можна задати: таблицею істинності, словесним описом (вербально), геометричним зображенням, аналітично (формулою).

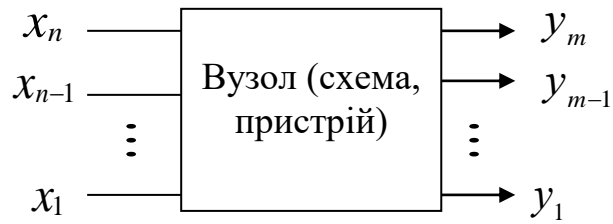


Рис. 1.7. Узагальнене позначення перетворювача інформації

Найпоширенішим є спосіб задання перемикальної функції *таблицею істинності* (табл. 1.1), у якій перелічують всі можливі набори значень аргументів функції та вказують, яке саме значення функція набуває на відповідному наборі.

Набором (кортежем) називають упорядковану послідовність значень аргументів. Кожному набору відповідає десятковий номер, який є кількісним еквівалентом двійкового числа, утвореного значеннями аргументів.

Наприклад, набору (кортежу) $\langle 1 \ 1 \ 0 \rangle$ відповідає десяткове число 6.

Таблиця 1.1

Таблиця істинності перемикальної функції $y = f(x_3, x_2, x_1)$

Номер набору	Набір значень аргументів			Значення функції y
	x_3	x_2	x_1	
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

Якщо набори впорядковують у вигляді $\langle x_n \dots x_2 \ x_1 \rangle$, то десятковий номер N набору визначають як $N = \sum_{i=1}^n x_i 2^{i-1}$.

Тому набори в таблиці істинності перемикальної функції можуть позначатися відповідними десятковими номерами.

Перевагами табличного задання перемикальної функції є наочність та універсальність – у вигляді таблиці істинності можна задати функцію від довільної кількості змінних (аргументів).

Перемикальну функцію, що залежить від n змінних, називають n -місною.

Перемикальна функція є повністю визначеною, якщо відомі (задані) її значення на всіх без винятку наборах (кортежах) змінних (аргументів). Якщо перемикальна функція задана не на всіх наборах змінних, то її називають неповністю визначеною функцією.

Перемикальну функцію, задану табл. 1.1, можна описати словесно (вербально), наприклад, так: «функція y від трьох змінних x_3, x_2, x_1 є повністю визначеною, дорівнює одиниці на наборах 3, 4, 5, 6 і нулю – на решті наборів».

Вербальний спосіб задання перемикальної функції використовують рідко.

Перемикальну функцію можна задати геометрично. Геометричним заданням перемикальної функції $f(x_n, x_{n-1}, \dots, x_1)$ від n змінних є зображення n -вимірного одиничного куба в декартовій системі координат зі спеціальним позначенням (зазвичай «крапкою») тих вершин, координати яких відповідають наборам (кортежам), на яких функція дорівнює одиниці. Кількість вершин куба дорівнює кількості наборів (кортежів). При $n = 1$ куб вироджується у пряму лінію, а при $n = 2$ – в одиничний квадрат.

Наприклад, на рис. 1.8а перемикальна функція від однієї змінної дорівнює нулю на наборі $\langle 0 \rangle$ і одиниці – на наборі $\langle 1 \rangle$; функція від двох змінних (рис. 1.8б) дорівнює одиниці на наборах (кортежах) $\langle 0 0 \rangle, \langle 1 0 \rangle, \langle 1 1 \rangle$ і нулю – на наборі $\langle 0 1 \rangle$; тривимірний куб (рис. 1.8в) подає перемикальну функцію від трьох змінних, що задана таблицею істинності (табл. 1.1).

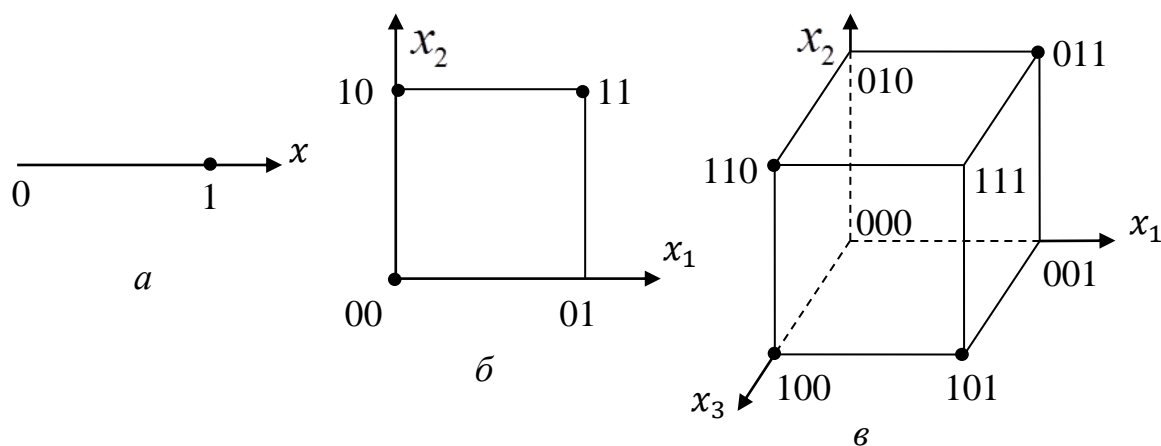


Рис. 1.8. Геометричне задання перемикальних функцій:
 а – від однієї змінної; б – від двох змінних; в – від трьох змінних

Геометричне задання перемикальних функцій є наочним і компактним за невеликої кількості змінних (до трьох), із зростанням кількості аргументів функції воно ускладнюється.

Найпростішим і найкомпактнішим є *аналітичне* (формульне) задання перемикальної функції, яке ґрунтується на використанні логічних операцій.

Наприклад, перемикальну функцію $y = f(x_3, x_2, x_1)$ від трьох змінних, якій відповідає таблиця істинності табл. 1.1, можна задати формулою

$$f(x_3, x_2, x_1) = x_3 \bar{x}_2 \vee x_3 \bar{x}_1 \vee \bar{x}_3 x_2 x_1 \quad (1.2)$$

або формулою

$$f(x_3, x_2, x_1) = (x_3 \vee x_2)(x_3 \vee x_1)(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_1). \quad (1.3)$$

Цю саму функцію $f(x_3, x_2, x_1)$ можна задати й іншими формулами. Аналітичне (формульне) задання перемикальної функції залежить від використовуваної алгебри (див. розділ 2).

Як бачимо, аналітичний спосіб задання перемикальної функції допускає множинність – одній і тій самій функції можуть відповідати декілька формул.

Від аналітичного задання перемикальної функції легко перейти до таблиці істинності функції. Для цього в задану формулу (аналітичний запис перемикальної функції) необхідно підставити по черзі всі набори значень аргументів (кортежі) – від $\langle 00\dots 0 \rangle$ до $\langle 11\dots 1 \rangle$, та для кожного набору обчислити значення функції, які й утворять стовпець у таблиці істинності.

Наприклад, підставляючи в формулу (1.2) по черзі всі кортежі від $\langle 0 0 0 \rangle$ до $\langle 1 1 1 \rangle$, отримаємо стовпець у табл. 1.1. Той самий результат отримаємо, якщо підставити вказані кортежі в формулу (1.3).

Найпростішими є перемикальні функції від однієї та двох змінних.

Існують чотири функції від однієї змінної (одномісні функції) (табл. 1.2).

Таблиця 1.2

Перемикальні функції від однієї змінної

x	$y_0 = f_0(x)$	$y_1 = f_1(x)$	$y_2 = f_2(x)$	$y_3 = f_3(x)$
0	0	0	1	1
1	0	1	0	1

Функція Константа нуль, $y_0 = f_0(x) = 0$, та *функція Константа одиниця*, $y_3 = f_3(x) = 1$, не залежать від значень аргументу, і тому називаються *виродженими* перемикальними функціями.

Невиродженими перемикальними функціями є *функція Повторення* (функція ТАК), $y_1 = f_1(x) = x$, та *функція Заперечення* (функція Інверсія, функція НЕ), $y_2 = f_2(x) = \bar{x}$ (табл. 1.3), причому функція $f_1(x) = x$ є тривіальною. Функція $y_1 = x$ повторює значення аргументу: якщо $x = 0$, то й $y_1 = 0$; якщо $x = 1$, то й $y_1 = 1$. Єдиною нетривіальною серед

одномісних перемикальних функцій є функція $y_2 = f_2(x) = \bar{x}$ – функція НЕ: якщо $x = 0$, то $y_2 = 1$; якщо $x = 1$, то $y_2 = 0$.

В електронній обчислювальній техніці перемикальні функції реалізують за допомогою логічних елементів.

Логічний елемент – це електронна схема (на основі транзистора), що реалізує певну перемикальну функцію. На кресленнях (рисунок) логічний елемент зображають у вигляді умовного графічного позначення (УГП).

Функцію $y = x$ реалізує логічний елемент, що має назву *Повторювач*, а функцію $y = \bar{x}$ – логічний елемент, що має назву *Інвертор* (рис. 1.9).

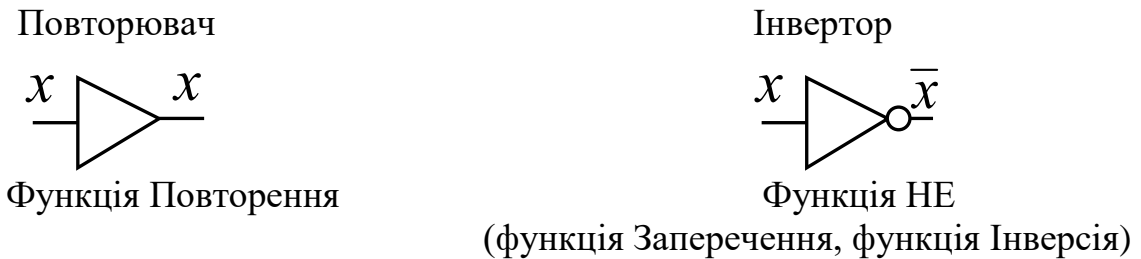
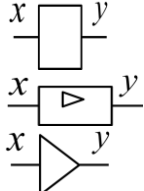
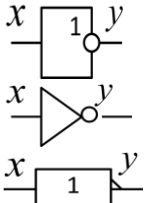


Рис. 1.9. Апаратна реалізація невироджених перемикальних функцій від однієї змінної

Таблиця 1.3

Невироджені перемикальні функції від однієї змінної

Функція	Таблиця істинності функції		Назва функції (варіанти)	Аналітичний запис функції (варіанти)	УГП логічного елемента (варіанти)	Назва логічного елемента (варіанти)
f_1	x	y	Функція	$y = x$		Повторювач
	0	0	• Повторення			
	1	1	• ТАК			
f_2	x	y	Функція	$y = x'$ $y = \bar{x}$ $y = \neg x$		<ul style="list-style-type: none"> • Інвертор • Елемент НЕ
	0	1	• Інверсія			
	1	0	• Заперечення • НЕ • NOT			

Розглянемо можливі перемикальні функції від двох змінних (табл. 1.4), загальна кількість яких – 16.

Таблиця 1.4

Перемикальні функції від двох змінних

$x_2 x_1$	F ₀	F ₁	F ₂	F ₃	F ₄	F ₅	F ₆	F ₇	F ₈	F ₉	F ₁₀	F ₁₁	F ₁₂	F ₁₃	F ₁₄	F ₁₅
0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

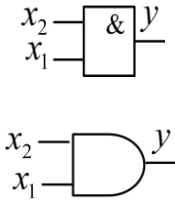
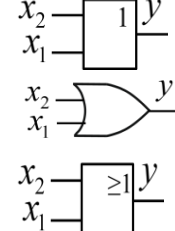
Виродженими (що не залежать від значень аргументів) є дві перемикальні функції: $F_0(x_2, x_1) = 0$ – функція Константа нуль, $F_{15}(x_2, x_1) = 1$ – функція Константа одиниця. Решта 14 функцій є *невиродженими*, серед них 4 функції є тривіальними, а 10 – нетривіальними.

Тривіальними є функції, що залежать лише від одного з аргументів:
 $F_3(x_2, x_1) = x_2$ – функція Повторення x_2 (функція Змінна x_2),
 $F_5(x_2, x_1) = x_1$ – функція Повторення x_1 (функція Змінна x_1),
 $F_{10}(x_2, x_1) = \bar{x}_1$ – функція Інверсія x_1 (функція Заперечення x_1 , функція НЕ x_1),
 $F_{12}(x_2, x_1) = \bar{x}_2$ – функція Інверсія x_2 (функція Заперечення x_2 , функція НЕ x_2).

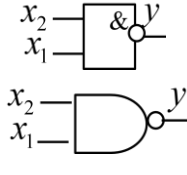
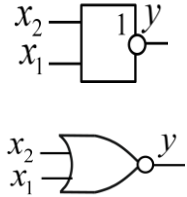
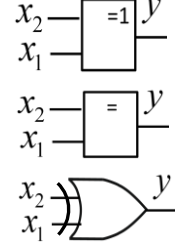
Нетривіальні двомісні функції залежать від обох аргументів (табл. 1.5).

Таблиця 1.5

Нетривіальні перемикальні функції від двох змінних
(залежать від обох аргументів)

Функція	Таблиця істинності функції		Назва функції (варіанти)	Аналітичний запис функції (варіанти)	УГП логічного елемента (варіанти)	Назва логічного елемента (варіанти)
1	2		3	4	5	6
F ₁	$x_2 x_1$	y	<ul style="list-style-type: none"> • Кон'юнкція • І • Логічне множення • AND 	$y = x_2 x_1$ $y = x_2 \cdot x_1$ $y = x_2 \& x_1$ $y = x_2 \wedge x_1$		<ul style="list-style-type: none"> • Елемент І • Кон'юнктор
	0 0	0				
	0 1	0				
	1 0	0				
	1 1	1				
F ₇	$x_2 x_1$	y	<ul style="list-style-type: none"> • Диз'юнкція • АБО • Логічне додавання • OR 	$y = x_2 \vee x_1$ $y = x_2 + x_1$		<ul style="list-style-type: none"> • Елемент АБО • Диз'юнктор
	0 0	0				
	0 1	1				
	1 0	1				
	1 1	1				

Продовження табл. 1.5

1	2		3	4	5	6
F ₁₄	x ₂ x ₁	y	<ul style="list-style-type: none"> • Заперечення кон'юнкції • І-НЕ • Функція Шеффера • Штрих Шеффера • NAND 	$y = \overline{x_2 x_1}$ $y = \overline{x_2 \cdot x_1}$ $y = \overline{x_2 \& x_1}$ $y = \overline{x_2 \wedge x_1}$ $y = \overline{x_2 / x_1}$		<ul style="list-style-type: none"> • Елемент І-НЕ • Елемент Шеффера
	0 0	1				
	0 1	1				
	1 0	1				
	1 1	0				
F ₈	x ₂ x ₁	y	<ul style="list-style-type: none"> • Заперечення диз'юнкції • АБО-НЕ • Функція Пірса • Стрілка Пірса • Функція Пірса-Вебба • NOR 	$y = \overline{x_2 \vee x_1}$ $y = \overline{x_2 + x_1}$ $y = x_2 \downarrow x_1$		<ul style="list-style-type: none"> • Елемент АБО-НЕ • Елемент Пірса
	0 0	1				
	0 1	0				
	1 0	0				
	1 1	0				
F ₆	x ₂ x ₁	y	<ul style="list-style-type: none"> • Сума за модулем два • Функція нерівнозначності • Виключне АБО • XOR 	$y = x_2 \oplus x_1$ $y = (x_2 + x_1) \bmod 2$ $y = x_2 \neq x_1$		<ul style="list-style-type: none"> • Суматор за модулем два • Елемент Виключне АБО
	0 0	0				
	0 1	1				
	1 0	1				
	1 1	0				
F ₉	x ₂ x ₁	y	<ul style="list-style-type: none"> • Заперечення суми за модулем два • Функція рівнозначності • Функція еквівалентності • NXOR 	$y = \overline{x_2 \oplus x_1}$ $y = x_2 \sim x_1$ $y = x_2 \equiv x_1$	Не має	Не має
	0 0	1				
	0 1	0				
	1 0	0				
	1 1	1				
F ₁₃	x ₂ x ₁	y	<ul style="list-style-type: none"> • Імплікація • Функція Від x₂ до x₁ 	$y = x_2 \rightarrow x_1$ $y = \overline{x_2} \vee x_1$	Не має	Не має
	0 0	1				
	0 1	1				
	1 0	0				
	1 1	1				
F ₂	x ₂ x ₁	y	<ul style="list-style-type: none"> • Інверсія імплікації • Функція Заборона за x₁ 	$y = \overline{x_2} \rightarrow \overline{x_1}$ $y = x_2 \overline{x_1}$	Не має	Не має
	0 0	0				
	0 1	0				
	1 0	1				
	1 1	0				

Продовження табл. 1.5

1	2		3	4	5	6	
F ₁₁	x_2	x_1	<ul style="list-style-type: none"> • Зворотна імплікація • Функція Від x_1 до x_2 	$y = x_2 \leftarrow x_1$ $y = x_2 \vee \bar{x}_1$	Не має	Не має	
	0	0					1
	0	1					0
	1	0					1
1	1	1					
F ₄	x_2	x_1	<ul style="list-style-type: none"> • Інверсія зворотної імплікації • Функція Заборона за x_2 	$y = \overline{x_2 \leftarrow x_1}$ $y = \bar{x}_2 x_1$	Не має	Не має	
	0	0					0
	0	1					1
	1	0					0
1	1	0					

На апаратному рівні використовують лише перші 5 функцій з табл. 1.5: F₁, F₇, F₁₄, F₈, F₆.

Функція F₁(x_2, x_1) – кон'юнкція, дорівнює одиниці лише на одному наборі аргументів – $\langle 1 \ 1 \rangle$, на решті наборів – нулю. Кон'юнкцію називають також функцією І або логічним множенням. У мовах програмування її позначають AND (наприклад, X2 AND X1). Можливі 4 варіанти аналітичного запису цієї функції:

$$F_1(x_2, x_1) = x_2 x_1 = x_2 \cdot x_1 = x_2 \& x_1 = x_2 \wedge x_1.$$

Функція F₇(x_2, x_1) – диз'юнкція, дорівнює нулю лише на одному наборі аргументів – $\langle 0 \ 0 \rangle$, на решті наборів – одиниці. Диз'юнкцію називають також функцією АБО або логічним додаванням. У мовах програмування її позначають OR (наприклад, A OR B). Використовують два варіанти аналітичного запису цієї функції: $F_7 = x_2 \vee x_1 = x_2 + x_1$.

Функція F₁₄(x_2, x_1) – функція І-НЕ, дорівнює нулю лише на наборі аргументів $\langle 1 \ 1 \rangle$, на решті наборів – одиниці. Цю функцію називають також: Заперечення кон'юнкції, функція Шеффера, штрих Шеффера – на честь американського математика Д. Шеффера, який на основі цієї функції розробив алгебру, яку називають алгеброю Шеффера. В англійській літературі цю функцію позначають NAND (Not AND). Використовують 5 варіантів аналітичного запису функції І-НЕ:

$$F_{14}(x_2, x_1) = \overline{x_2 x_1} = \overline{x_2 \cdot x_1} = \overline{x_2 \& x_1} = \overline{x_2 \wedge x_1} = x_2 / x_1.$$

Функція F₈(x_2, x_1) – функція АБО-НЕ, дорівнює одиниці лише на одному наборі аргументів – $\langle 0 \ 0 \rangle$, на решті наборів – нулю. Цю функцію називають також: Заперечення диз'юнкції, функція Пірса, стрілка Пірса, функція Пірса-Вебба. Математики Ч. Пірс та Д. Вебб, які незалежно один від одного вивчали властивості цієї функції, розробили алгебру, що дістала назву алгебри Пірса. В англійській літературі функцію АБО-НЕ

позначають NOR (Not OR). Аналітично цю функцію записують так (3 варіанти):

$$F_8(x_2, x_1) = \overline{x_2 \vee x_1} = \overline{x_2 + x_1} = x_2 \downarrow x_1.$$

Функція $F_6(x_2, x_1)$ – сума за модулем два, дорівнює одиниці лише тоді, коли значення її аргументів протилежні, тобто на наборах $\langle 0 \ 1 \rangle$ і $\langle 1 \ 0 \rangle$; в інших випадках функція дорівнює нулю. Тому її називають також *функцією нерівнозначності*.

Крім того, F_6 часто називають *функцією Виключне АБО*. В англійській літературі і в мовах програмування її позначають XOR (eXclusive OR), в російськомовній – Исключающее ИЛИ.

Використовують три варіанти аналітичного запису цієї функції:

$$F_6(x_2, x_1) = x_2 \oplus x_1 = (x_2 + x_1) \bmod 2 = x_2 \neq x_1.$$

Для апаратної реалізації перелічених вище п'яти функцій AND, OR, NAND, NOR, XOR промисловістю випускаються відповідні логічні елементи (у вигляді мікросхем) (рис. 1.10, 1.12).

Для інших п'яти функцій від двох змінних, що залежать від обох аргументів, – F_9 , F_{13} , F_2 , F_{13} , F_{11} , F_4 , логічні елементи не розробляють, оскільки прямого застосування в комп'ютерній схемотехніці ці функції не мають. Але дві з них – F_9 та F_{13} можуть використовуватись для опису процесів оброблення даних на логічному рівні.

Так, у деяких мовах програмування використовують оператор NXOR (Not eXclusive OR), який відповідає функції F_9 – *заперечення суми за модулем два*. Цю функцію називають також функцією рівнозначності або функцією еквівалентності. Аналітично її записують у вигляді (3 варіанти):

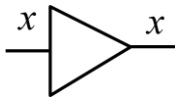
$$F_9(x_2, x_1) = \overline{x_2 \oplus x_1} = x_2 \sim x_1 = x_2 \equiv x_1.$$

Функціонування ЛЕ часто описують за допомогою часової діаграми (рис. 1.11).

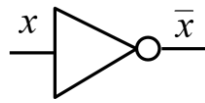
Функція $F_{13}(x_2, x_1)$ – *імплікація*, дорівнює нулю лише на наборі аргументів $\langle 1 \ 0 \rangle$, на решті наборів – одиниці. Ця функція має й іншу назву – *функція Від x_2 до x_1* . Аналітичний запис функції:

$$F_{13}(x_2, x_1) = x_2 \rightarrow x_1 \text{ (кажуть «}x_2 \text{ імплікує }x_1\text{») або } F_{13}(x_2, x_1) = \bar{x}_2 \vee x_1.$$

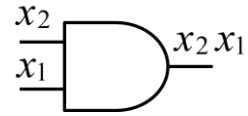
Повторювач
(buffer gate)



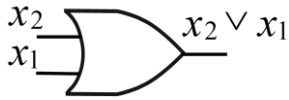
Інвертор
(NOT gate)



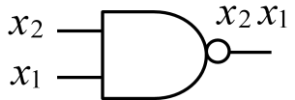
Елемент І
(AND gate)



Елемент АБО
(OR gate)



Елемент І-НЕ
(NAND gate)



Елемент АБО-НЕ
(NOR gate)



Суматор за модулем два
(XOR gate)

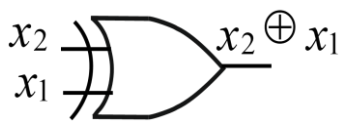


Рис. 1.10. Умовні графічні позначення деяких типів логічних елементів

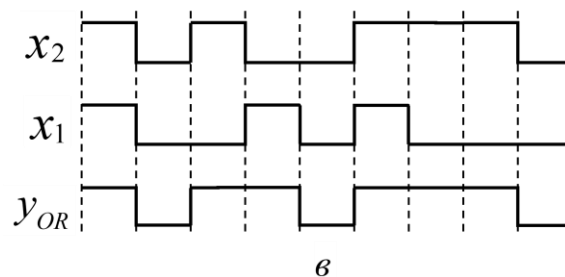
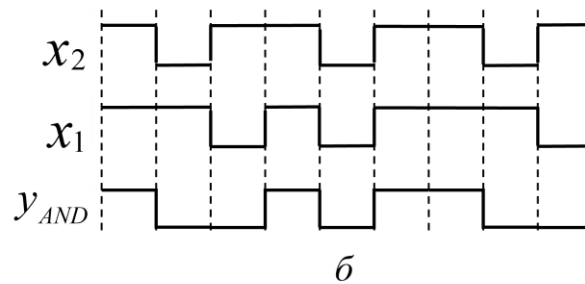
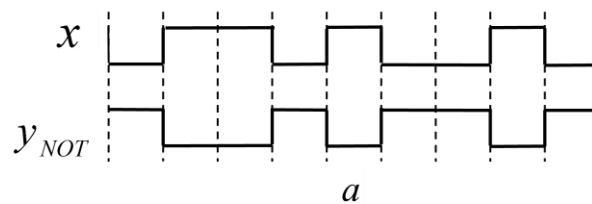


Рис. 1.11. Часові діаграми роботи логічних елементів (без урахування затримки сигналів): *a* – НЕ; *б* – І; *в* – АБО

Функція F_{13} має й алгебраїчне трактування: вона дорівнює одиниці, якщо $x_2 \leq x_1$, тобто

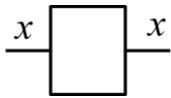
$$F_{13}(x_2, x_1) = \begin{cases} 1, & \text{якщо } x_2 \leq x_1, \\ 0, & \text{якщо } x_2 > x_1. \end{cases}$$

x_2	x_1
0	0
0	1
1	1

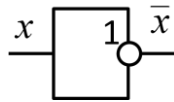
Якщо розглянути лише ті набори, на яких F_{13} дорівнює одиниці, то бачимо, що значення у стовпці x_2 або співпадає зі значенням у стовпці x_1 , або є меншим за нього, тобто стовпець x_2 є ніби складовою частиною стовпця x_1 . Звідси зрозумілим стає слово «імплікує» – воно означає «є складовою частиною» (x_2 є складовою частиною x_1). Властивість *імпліковності* є важливим при мінімізації перемикальних функцій (див. розділ 3).

Перемикальні функції від однієї та двох змінних називають *елементарними*.

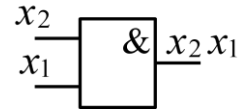
Повторювач



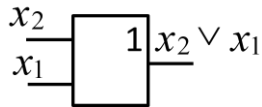
Інвертор



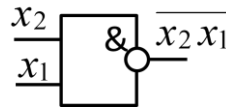
Елемент І



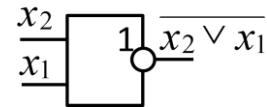
Елемент АБО



Елемент І-НЕ
(Елемент Шеффера)



Елемент АБО-НЕ
(Елемент Пірса)



Суматор за модулем два

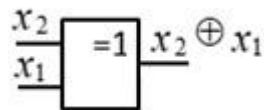


Рис. 1.12. Альтернативні умовні графічні позначення деяких типів логічних елементів у вітчизняній літературі

Кількість перемикальних функцій від однієї змінної ($n = 1$) дорівнює 4 ($4 = 2^{2^1}$), від двох змінних ($n = 2$) становить $16 = 2^{2^2}$, від n змінних – 2^{2^n} .

Дійсно, з n змінних можна утворити $A = 2^n$ різних наборів (кортежів) значень змінних. На кожному наборі перемикальна функція може набувати

двох значень (0 або 1). Тому загальна кількість перемикальних функцій від n змінних становить $2^A = 2^{2^n}$ (рис. 1.13).

Наприклад, при $n = 3$ можна утворити 256 перемикальних функцій, а при $n = 4$ – 65536 функцій.

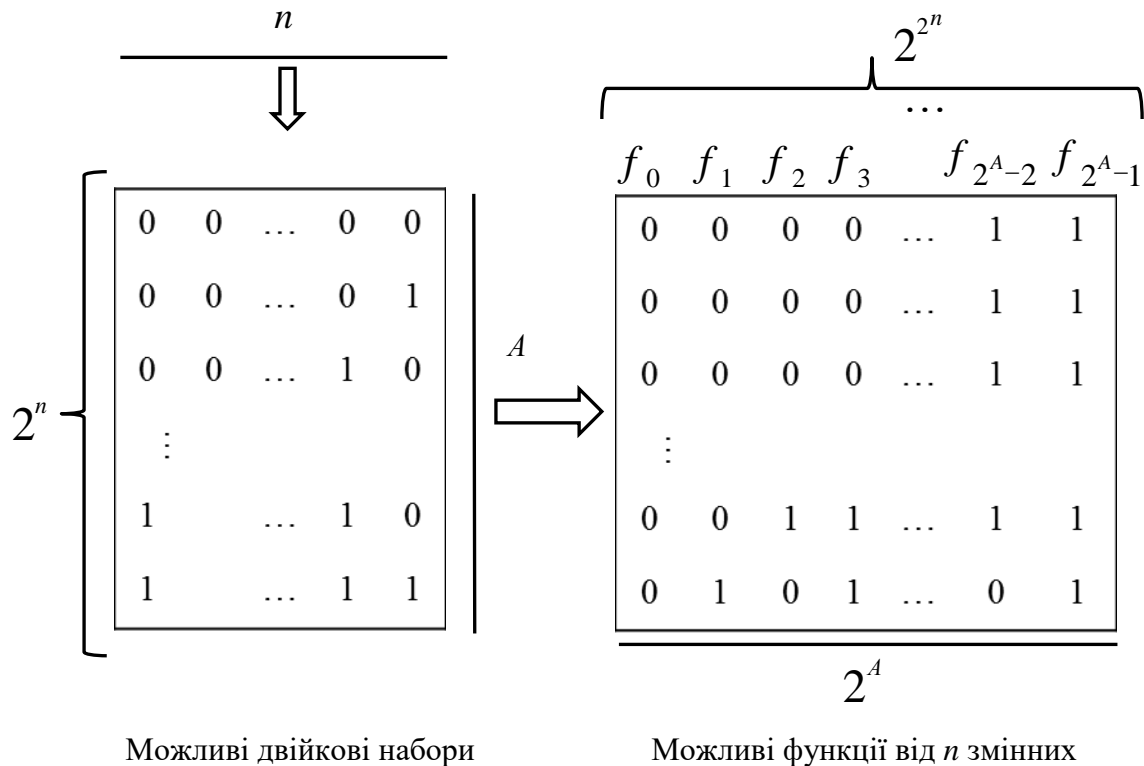


Рис. 1.13. Побудова перемикальних функцій від n змінних

Областю визначення перемикальної функції від n змінних є сукупність 2^n двійкових наборів (кортежів) значень змінних.

Областю значень перемикальної функції від n змінних є множина $\{0, 1\}$.

Такі перемикальні функції від двох змінних як кон'юнкція (І), диз'юнкція (АБО), функція Шеффера (І-НЕ), функція Пірса (АБО-НЕ) можна поширити на довільну кількість аргументів. При цьому зберігаються форми запису функцій та УГП відповідних логічних елементів:

$$AND \rightarrow x_k \cdot x_{k-1} \cdot \dots \cdot x_1,$$

$$OR \rightarrow x_k \vee x_{k-1} \vee \dots \vee x_1,$$

$$NAND \rightarrow \overline{x_k \cdot x_{k-1} \cdot \dots \cdot x_1},$$

$$OR \rightarrow \overline{x_k \vee x_{k-1} \vee \dots \vee x_1}.$$

Наприклад, кон'юнкція трьох змінних має вигляд $F_K(x_3, x_2, x_1) = x_3 x_2 x_1$, диз'юнкція чотирьох змінних – $F_D(x_4, x_3, x_2, x_1) = x_4 x_3 x_2 x_1$, функція І-НЕ чотирьох аргументів – $F_{III}(x_4, x_3, x_2, x_1) = \overline{x_4 x_3 x_2 x_1}$, функція АБО-НЕ трьох аргументів $F_{II}(x_3, x_2, x_1) = \overline{x_3 \vee x_2 \vee x_1}$. Цим функціям відповідають логічні елементи з відповідною кількістю входів (рис. 1.14).

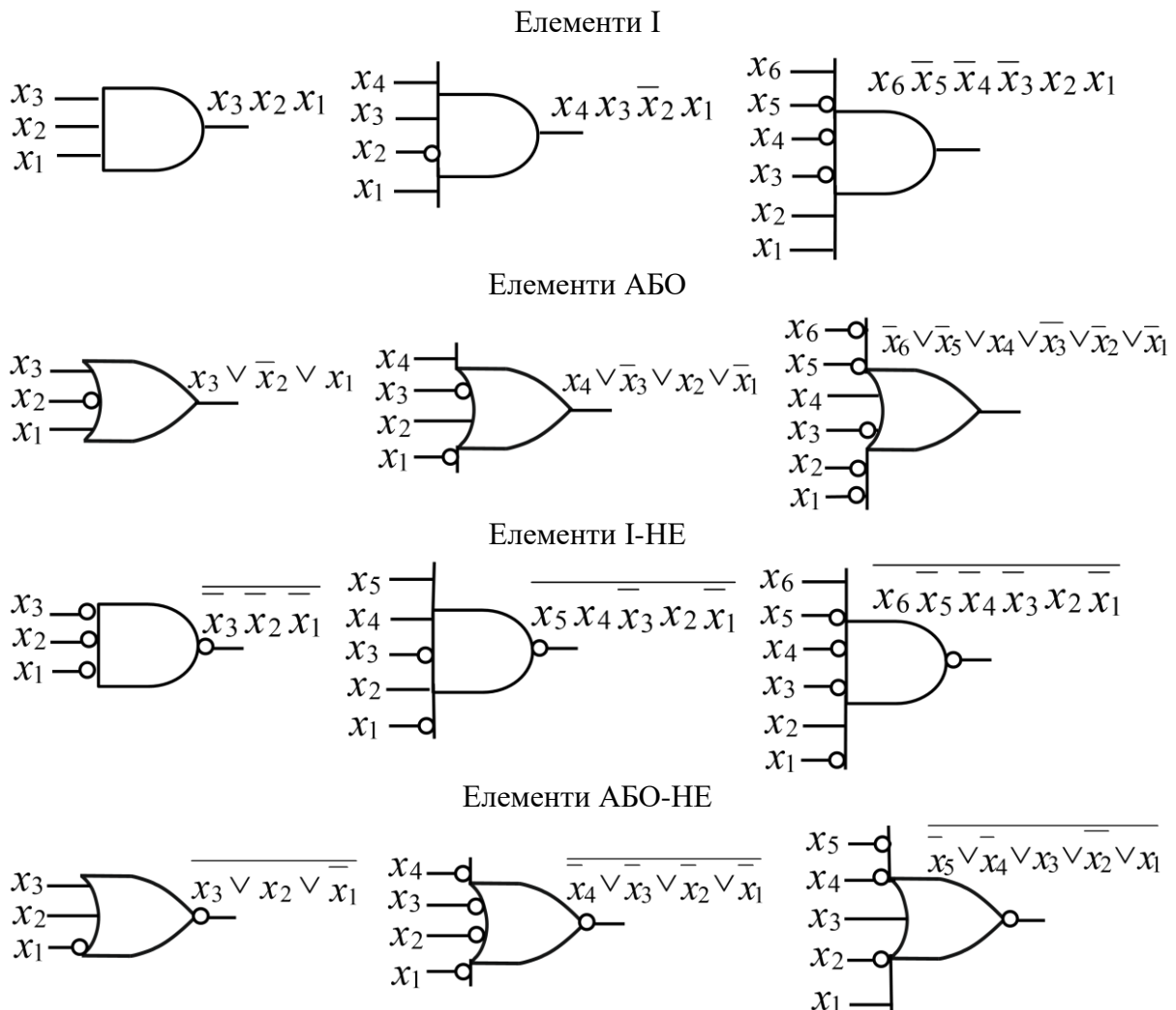


Рис. 1.14. УГП багатовходових логічних елементів

Таким чином, перемикальні функції AND, OR, NAND, NOR є n -місними ($n \geq 2$), функція NOT – одномісна, функція XOR у загальному випадку є n -місною, але логічні елементи, які реалізують цю функцію (суматори за модулем два) зазвичай випускають 2-входовими.

Перемикальній функції відповідає логічна схема, що є сукупністю відповідним чином з'єднаних логічних елементів. Логічна схема є апаратною реалізацією перемикальної функції.

Наприклад, перемикальній функції

$$y = f(x_3, x_2, x_1) = \overline{x_3 \vee x_2 \vee x_2 x_1}$$

відповідає логічна схема на рис. 1.15.

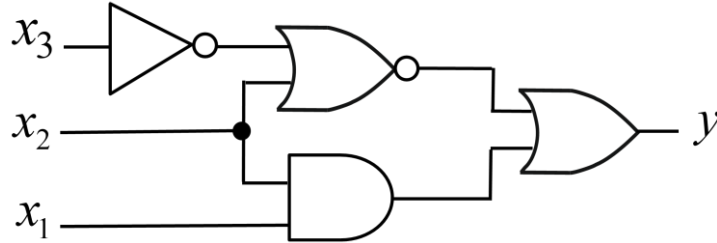


Рис. 1.15. Логічна схема, що реалізує функцію $y = f(x_3, x_2, x_1)$

І навпаки, якщо відома логічна система, то її поведінку можна описати системою перемикальних функцій (кожному виходу логічної схеми відповідає одна перемикальна функція).

Наприклад, логічній схемі на рис. 1.16 відповідає система перемикальних функцій

$$\begin{cases} y_2 = (x_4 \vee x_3) \overline{x_3 x_2}, \\ y_1 = \overline{x_3 x_2} \vee x_1. \end{cases} \quad (1.4)$$

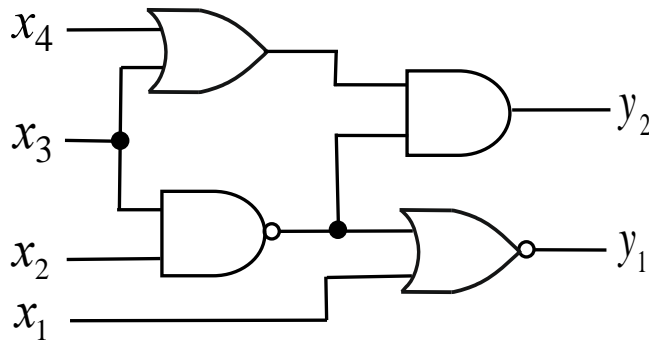


Рис. 1.16. Логічна схема, що описується системою перемикальних функцій (1.4)

Розрізняють два види логічних схем – комбінаційні схеми та послідовні схеми.

Якщо сукупність вихідних сигналів логічної схеми з n входами і m виходами в даний момент часу повністю визначається сукупністю вхідних сигналів у цей самий момент часу і не залежить від вхідних сигналів, що діяли на входах в попередні моменти часу, то таку логічну схему називають *комбінаційною схемою*. Вважають, що комбінаційна схема має один стан. Поведінку комбінаційної схеми можна описати системою перемикальних функцій (1.1).

Якщо сукупність вихідних сигналів логічної схеми в даний момент часу залежить не лише від вхідних сигналів, що діють у цей самий момент часу, але й від стану, в якому перебуває схема (тобто й від сигналів, що діяли в попередній момент часу), то таку логічну схему називають *послідовнісною схемою* або *схемою з пам'яттю* (рис. 1.17).

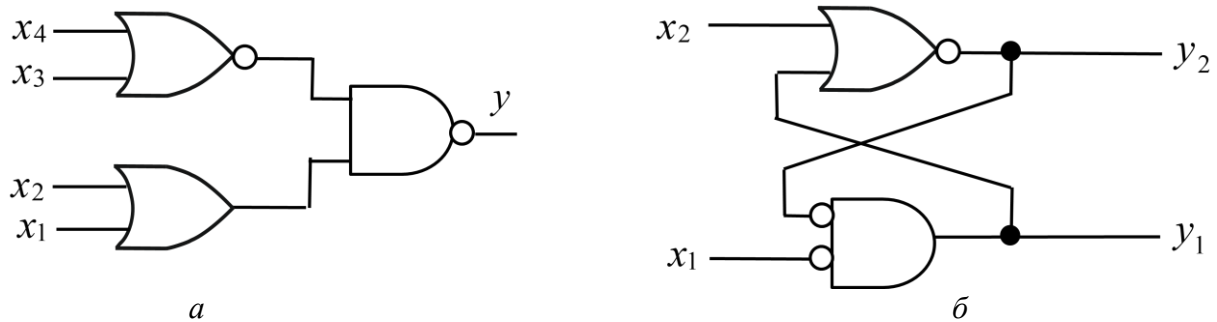


Рис. 1.17. Види логічних схем:
a – комбінаційна схема; *б* – послідовнісна схема (схема з пам'яттю)

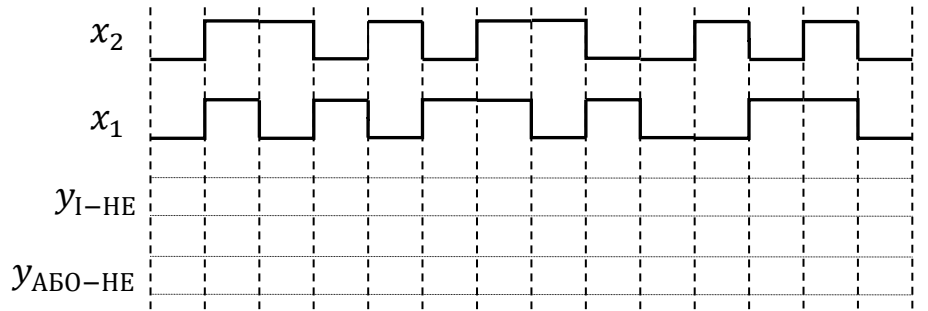
Основною ознакою послідовнісної схеми (схеми з пам'яттю) є наявність у ній петель. *Петля* – це електричне з'єднання від виходу деякого логічного елемента до одного з входів цього самого елемента (можливо й через інші логічні елементи). У комбінаційних схемах петлі відсутні.

Як комбінаційні, так і послідовнісні схеми називають *цифровими автоматами*. При цьому комбінаційні схеми є *тривіальними автоматами* (тобто автоматами, що не мають пам'яті), а послідовнісні схеми є *автоматами з пам'яттю*.

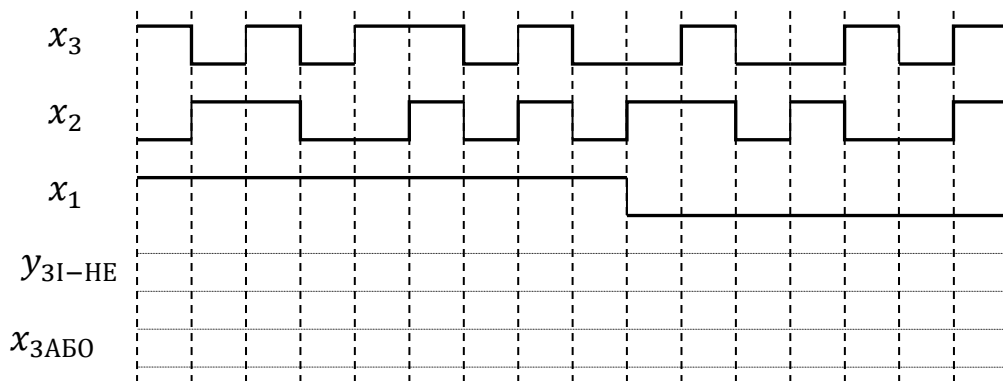
Запитання та завдання

1. Яку функцію називають перемикальною?
2. Перелічіть способи задання перемикальних функцій.
3. Яку перемикальну функцію називають *n*-місною?
4. Яку перемикальну функцію називають повністю визначеною (неповністю визначеною)?
5. Як перейти від аналітичного (формульного) задання перемикальної функції до таблиці істинності функції?
6. Які перемикальні функції називаються виродженими, невиродженими, тривіальними, нетривіальними?
7. Наведіть умовні графічні позначення логічних елементів, які реалізують елементарні перемикальні функції: NOT, AND, OR, NAND, NOR, XOR.
8. Яку кількість перемикальних функцій можна утворити від *n* змінних?

9. Що є областю визначення та областю значень перемикальної функції від n змінних?
10. Назвіть види логічних схем. Дайте визначення кожної з них.
11. Побудуйте часову діаграму вихідного сигналу у логічного елемента І-НЕ, АБО-НЕ (без урахування затримки сигналів у елементі), якщо вхідні сигнали x_2, x_1 мають вигляд:



12. Побудуйте часову діаграму вихідного сигналу у логічного елемента ЗІ-НЕ, ЗАБО (без врахування затримки сигналів елементом), якщо вхідні сигнали x_3, x_2, x_1 мають вигляд:



1.3. Фізична реалізація логічних елементів

Апаратні засоби сучасних комп'ютерних систем будують на основі напівпровідникових *інтегральних мікросхем* (ІМС). Напівпровідникова ІМС – це мікроелектронний виріб, всі елементи та міжелементні з'єднання якого виконані в товщі (об'ємі) і/або на поверхні напівпровідникового кристала – кремнію, германію, галію, індію. Сформовані на кристалі методами напівпровідникової технології області еквівалентні елементам електричної схеми – резисторам, діодам, конденсаторам, транзисторам тощо. ІМС може бути взята в корпус, що має зовнішні виводи (рис. 1.18). Корпус – це частина конструкції ІМС, призначена для захисту ІМС від зовнішнього середовища та для з'єднання із зовнішніми електричними колами за допомогою виводів. У корпусі можуть розміщуватись декілька напівпровідникових кристалів, розташованих на одній підкладці.

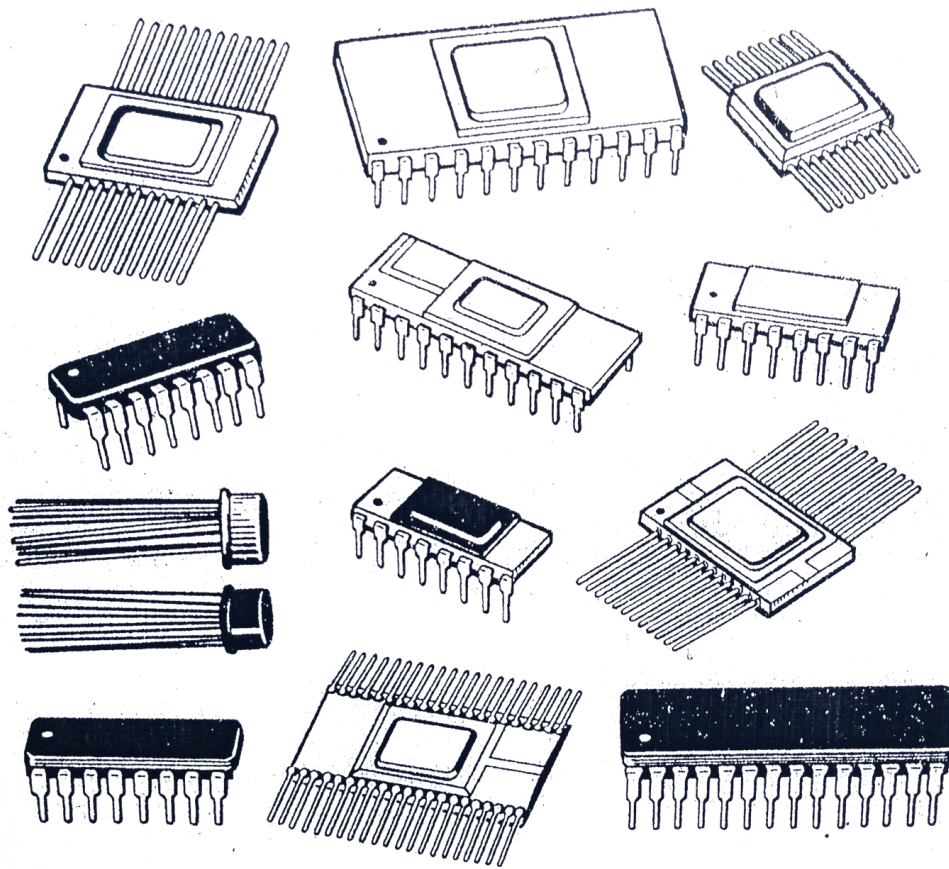


Рис. 1.18. Загальний вигляд деяких типів інтегральних мікросхем

Головною тенденцією розвитку інтегральних мікросхем є збільшення кількості елементів на одному кристалі, тобто підвищення їх функціональної складності.

Функціональну складність мікросхем прийнято характеризувати ступенем інтеграції, тобто кількістю елементів на кристалі або в корпусі.

Ступінь інтеграції ІМС визначають за формулою $k = \lceil \lg N \rceil$, де k – коефіцієнт, що характеризує складність мікросхеми, значення якого округлюють до найближчого більшого цілого числа, а N – кількість елементів та компонентів, що входять до складу ІМС.

ІМС 1-го ступеня інтеграції (МІС – малі інтегральні схеми, SSI – small scale integration circuit) містять на кристалі до 10-ти елементів та компонентів, 2-го ступеня інтеграції – від 11 до 100 елементів, 3-го ступеня інтеграції – від 101 до 1000 елементів і т.д. (табл. 1.6).

Таблиця 1.6

Ступені інтеграції ІМС

k	Ступінь інтеграції	Назва		Кількість елементів на кристалі, N	Розмір елементів, мкм
		англ.	укр.		
1	малий	SSI	МІС	до 10	10
2	середній	MSI	СІС	до 100	5
3	великий	LSI	ВІС	до 1000	3 – 1
4	надвеликий	VLSI	НВІС	до 10000	1
5	ультравеликий	ULSI	УВІС	понад 10000	0,1 – 0,001

Примітка. S – *small*, M – *medium*, L – *large*, VL – *over large*, UL – *ultra large*.

Сучасний етап розвитку електроніки характеризується розробленням ІМС зі ступенем інтеграції понад 100000 елементів на кристалі.

Крім ступеня інтеграції, для характеристики ІМС використовують також показник *щільності розміщення елементів* – кількість елементів (найчастіше транзисторів) на одиниці площі кристала. Цей показник характеризує головним чином рівень технології, зараз він складає понад 1000 елементів/мм².

В основі напівпровідникової технології лежить поняття *p-n* переходу. Електронно-дірковим переходом або *p-n* переходом називають перехід між двома областями напівпровідника, одна з яких має провідність *n*-типу, а інша – *p*-типу.

У процесі дифузії електрони і дірки переміщуються через поверхню розмежування напівпровідників різних типів електропровідності в протилежних напрямках – електрони переміщуються з *n*-області в *p*-область, а дірки – з *p*-області в *n*-область. Але оскільки ці носії переносять електричні заряди протилежних знаків, то в підсумку виникають електронна та діркова складові єдиного дифузійного струму, напрям якого співпадає з напрямом переміщення дірок.

Вольт-амперна характеристика (ВАХ) *p-n* переходу має експоненціальний характер (рис. 1.19), а *p-n* перехід еквівалентний діоду.

Основними елементами ІМС є транзистори.

Транзистор у 1948 р. винайшли американські вчені-фізики Шоклі, Бардін та Браттейн.

Розрізняють два види транзисторів: біполярні (характерна особливість – наявність двох *p-n* переходів, керування здійснюється струмом), уніполярні (польові) (один перехід, керування здійснюється напругою).

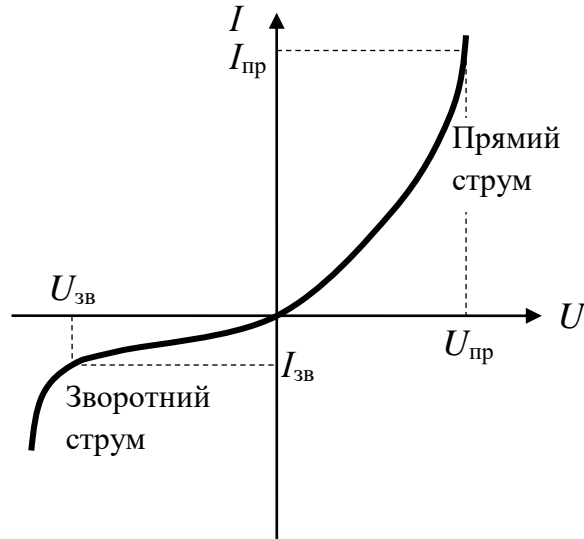


Рис. 1.19. Вольт-амперна характеристика $p-n$ переходу

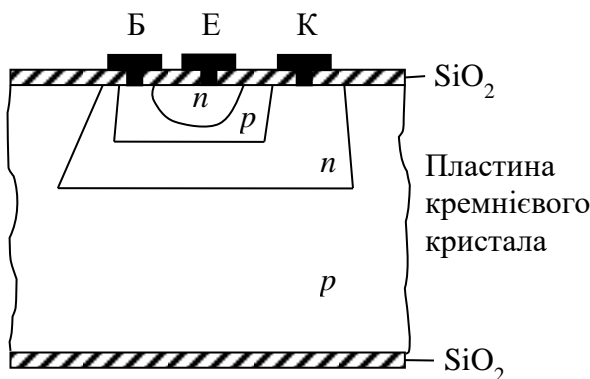
Біполярні транзистори бувають двох типів: $n-p-n$ транзистори та $p-n-p$ транзистори.

Розглянемо будову $n-p-n$ транзистора.

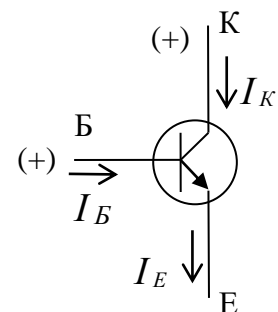
На пластині кремнієвого кристала, що має p -провідність, за допомогою низки технологічних прийомів отримують чотиришарову транзисторну структуру (рис. 1.20), де Б (база), Е (емітер), К (колектор) – з'єднувальні виводи, виготовлені з алюмінію. У транзисторній структурі наявні два $p-n$ переходи – один між базою та емітером (перехід Б-Е), інший – між базою та колектором (перехід Б-К).

У біполярних транзисторах переходи Б-К та Б-Е працюють як діоди. На основі транзисторних структур будують логічні елементи.

Розглянемо реалізацію на основі біполярного $n-p-n$ транзистора найпростішого логічного елемента – інвертора (елемента НЕ) (рис. 1.21).



Структура транзистора
(у розрізі)



Умовне графічне
позначення

Рис. 1.20. Біполярний $n-p-n$ транзистор

Транзистор VT може перебувати в трьох різних робочих станах (режимах): закритому (режим відсічення), активному, насиченому. Закритий стан транзистора відіграє роль розімкненого ключа, а насичений – замкненого. В активний (підсилювальний) режим транзистор переходить в процесі зміни свого стану.

У режимі відсічення (транзистор закритий) опір між колектором та емітером наближається до нескінченності.

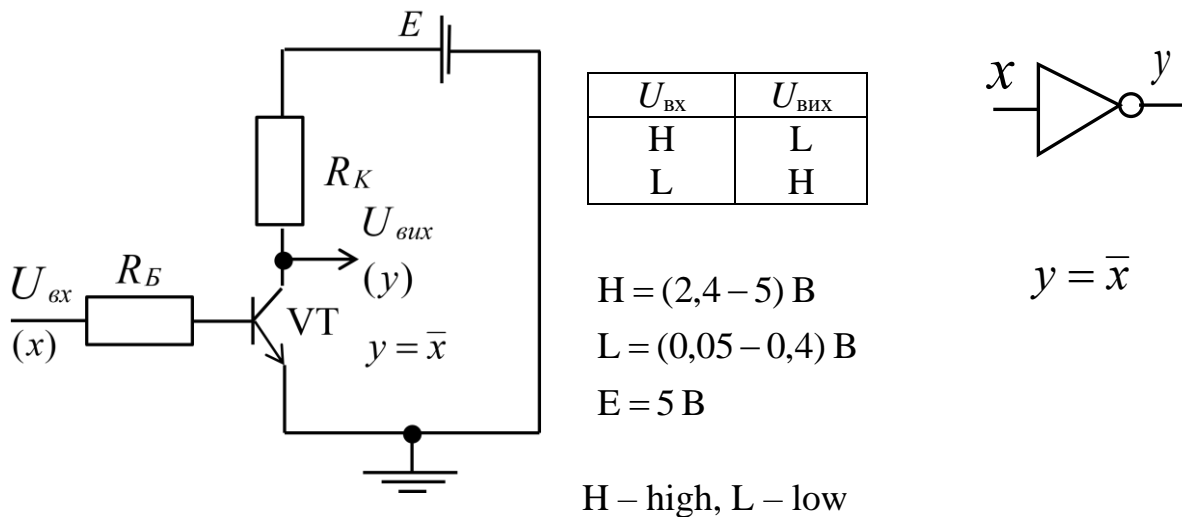


Рис. 1.21. Транзисторна структура, що реалізує інвертор (логічний елемент НЕ)

На колектор та базу подають напругу позитивної полярності, причому напруга на колекторі має бути вищою, ніж на базі. Коли на вході напруга ($U_{вх}$) має низький рівень (L) – діапазон (0,05 – 0,4) В, то $p-n$ перехід база-емітер відкритий (має пряме зміщення), а $p-n$ перехід база-колектор закритий (зміщений у зворотному напрямі). Такий стан транзистора відповідає розімкненому ключу, і його називають режимом відсічення. При цьому через резистор R_K протікає малий залишковий струм $I_K = I_{зал}$, а напруга на виході складає величину $U_{вих} = E - I_{зал} R_K \approx E$, тобто має високий рівень (H) – зазвичай (2,4 – 5) В.

При поступовому збільшенні входньої напруги (від рівня L до рівня H) в базі транзистора виникає струм I_B , і транзистор переходить в активний режим – починає зростати струм у колі колектора та емітера: $I_K = \beta I_B$, $I_E = I_B + I_K = (1 + \beta) I_B$, де β – коефіцієнт підсилення, а $p-n$ перехід база-колектор зміщується у прямому напрямі, транзистор починає відкриватись.

Коли $U_{вх}$ досягне високого рівня (H) транзистор переходить в режим насичення (повністю відкривається), тобто в такий стан, коли збільшення струму в базі транзистора не приводить до значної зміни колекторного та емітерного струму. Із збільшенням колекторного струму до величини $I_K = \beta_H I_B$, де β_H – коефіцієнт підсилення струму в режимі

насичення, напруга на виході знижується до величини $U_{\text{вих}} = E - I_{\text{к}}R_{\text{к}} = U_{\text{ке}}$, яка відповідає низькому рівню (L), тобто $U_{\text{вих}}$ змінюється від високого рівня до низького.

Далі, якщо на вхід подати низький рівень напруги (L), то струм в базу не надходить, і транзистор закривається, тобто переходить в режим відсічення, на виході встановлюється високий рівень (H).

Таким чином, схема транзисторного ключа на рис. 1.21 забезпечує інвертування вхідної напруги, яка подається на базу транзистора (рис. 1.22).

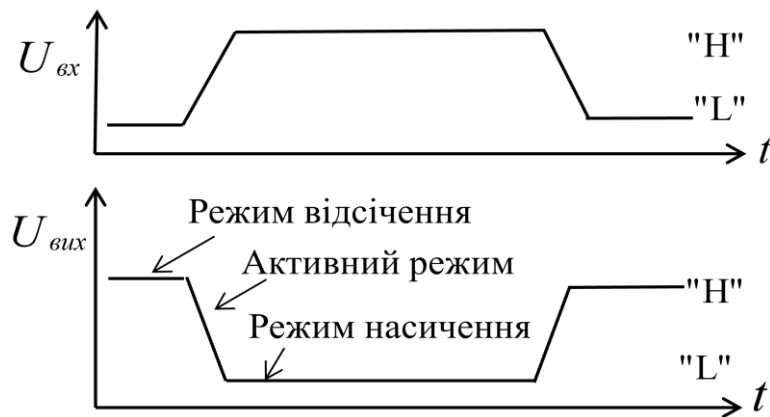


Рис. 1.22. Часова діаграма роботи транзисторного ключа – інвертора

Якщо $U_{\text{вх}}$ позначити як x (змінна), а $U_{\text{вих}}$ як y (функція), то схема на рис. 1.21 виконує перетворення $y = \bar{x}$, тобто реалізує логічний елемент НЕ (інвертор).

Інвертор можна реалізувати також на основі біполярного транзистора $p-n-p$ типу, який структурно відрізняється від $n-p-n$ транзистора тим, що його вирощують на пластині кремнієвого кристала,

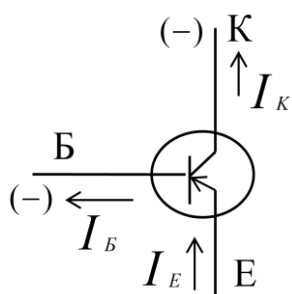


Рис. 1.23. Умовне графічне позначення біполярного $p-n-p$ транзистора

що має n -провідність (див. рис. 1.20). При цьому необхідно враховувати, що полярність напруги має бути змінена на протилежну (рис. 1.23).

Застосовуючи різні варіанти з'єднання кількох транзисторів між собою можна отримувати найпростіші логічні елементи.

Реалізація двовходового логічного елемента АБО (елемента 2АБО)

Розглянемо паралельне з'єднання двох транзисторів зі спільним емітерним навантаженням $R_{\text{Е}}$ (рис. 1.24).

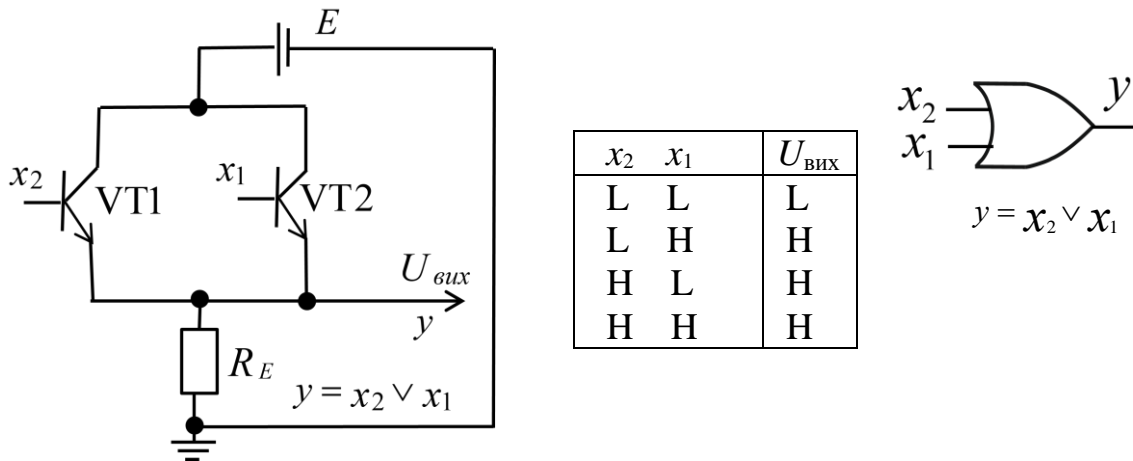


Рис. 1.24. Транзисторна структура, що реалізує логічний елемент 2АБО

Якщо на входи x_2 , x_1 подати низький рівень напруги (L), то обидва транзистори VT1, VT2 будуть закриті, і вихідна напруга $U_{\text{вих}}$ буде низькою (L). Якщо хоча б на один з входів x_2 , x_1 подати високий рівень напруги (H), то за рахунок відкриття відповідного транзистора (VT1, VT2 або обох одразу) на виході $U_{\text{вих}}$ отримаємо високий рівень напруги (H).

Отже, структура на рис. 1.24 відповідає функціонуванню двовходового логічного елемента АБО.

Реалізація двовходового логічного елемента І (елемента 2І)

Якщо виконати послідовне з'єднання транзисторів зі спільним емітерним навантаженням R_E (рис. 1.25), то отримаємо схему, поведінка якої відповідає функціонуванню двовходового логічного елемента І.

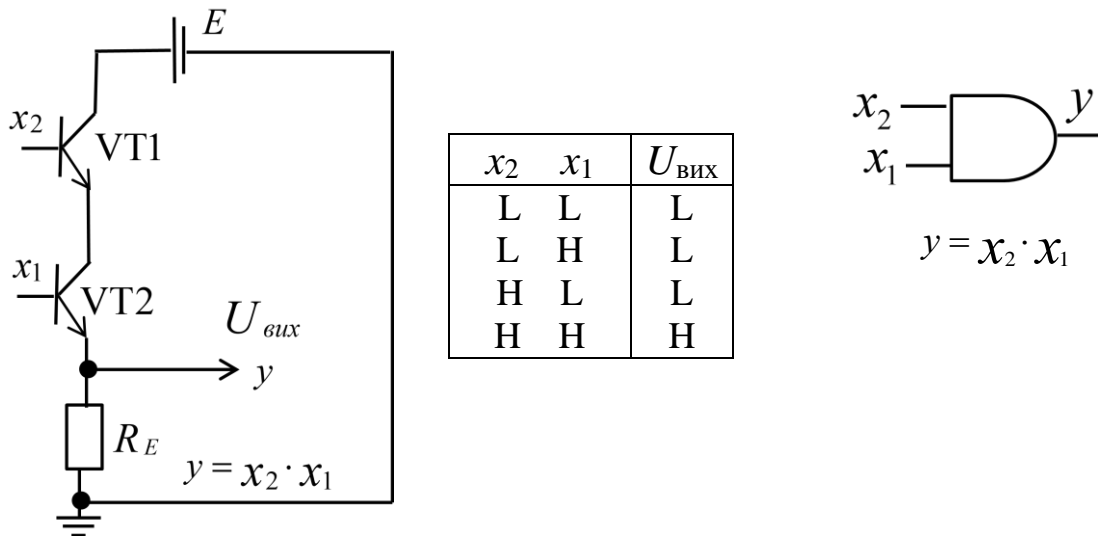


Рис. 1.25. Транзисторна структура, що реалізує логічний елемент 2І

Дійсно, високий рівень (Н) напруги на виході буде мати місце лише в тому випадку, коли обидва транзистори будуть відкриті. Цього можна досягти, якщо одночасно на обидва входи x_2 , x_1 подати високий рівень (Н) напруги. Якщо хоча б на один з входів x подати низький рівень (L), то один з транзисторів (або обидва) буде закритий, і, відповідно, значення напруги на виході $U_{\text{вих}}$ матиме низький рівень (L).

Реалізація логічного елемента 2АБО-НЕ

Виконаємо паралельне з'єднання двох транзисторів зі спільним колекторним навантаженням R_K (рис. 1.26).

На виході схеми буде високий рівень (Н) напруги, якщо обидва транзистори закриті. Це досягається лише в тому випадку, якщо на входи x_2 , x_1 обох транзисторів подати низький рівень (L) напруги. Якщо хоча б один із транзисторів відкритий (для цього на вхід x відповідного транзистора слід подати високий рівень (Н) напруги), то на виході $U_{\text{вих}}$ встановиться низький рівень (L) напруги.

Отже, структура на рис. 1.26 відтворює функціонування двовходового логічного елемента АБО-НЕ.

Реалізація двовходового логічного елемента 2І-НЕ

Якщо при послідовному з'єднанні двох транзисторів застосувати спільне колекторне навантаження R_K , то отримаємо схему, функціонування якої відповідає таблиці істинності двовходового елемента І-НЕ (рис. 1.27).

Для того, щоб отримати на виході $U_{\text{вих}}$ низький рівень напруги, обидва транзистори VT1, VT2 мають бути відкриті. Це можливо лише в тому випадку, якщо на входи x_2 , x_1 обох транзисторів подати високий рівень (Н) напруги. Якщо на вході (на базі) хоча б одного з транзисторів присутній низький рівень (L) напруги, то відповідний транзистор буде закритий. Відтак коло між точкою $U_{\text{вих}}$ та точкою заземлення буде розірване, у ньому протікатиме дуже малий залишковий струм. Падіння напруги на резисторі R_K буде незначним, і на виході $U_{\text{вих}}$ встановиться напруга $U_{\text{вих}} \approx E$, що відповідає високому рівню (Н).

Отже, транзисторна схема на рис. 1.27 функціонує за законом, який відповідає таблиці істинності логічного елемента 2І-НЕ.

У наведених на рис. 1.24 – 1.27 структурах кількість транзисторів дорівнює кількості входів відповідних логічних елементів. Для отримання логічних елементів з більшою кількістю входів у зазначених структурах необхідно або збільшувати кількість транзисторів, або використовувати багатоємні інтегральні транзистори.

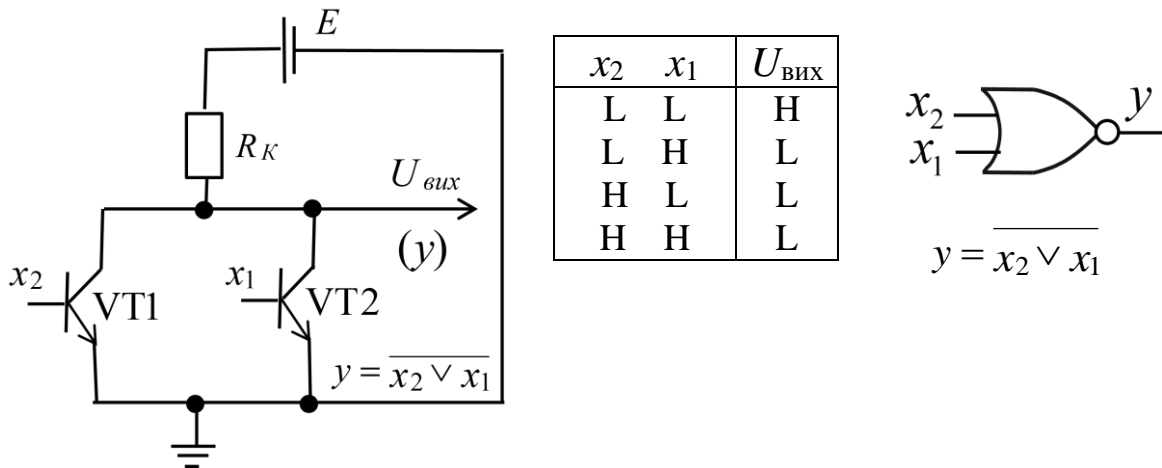


Рис. 1.26. Транзисторна структура, що реалізує логічний елемент 2АБО-НЕ

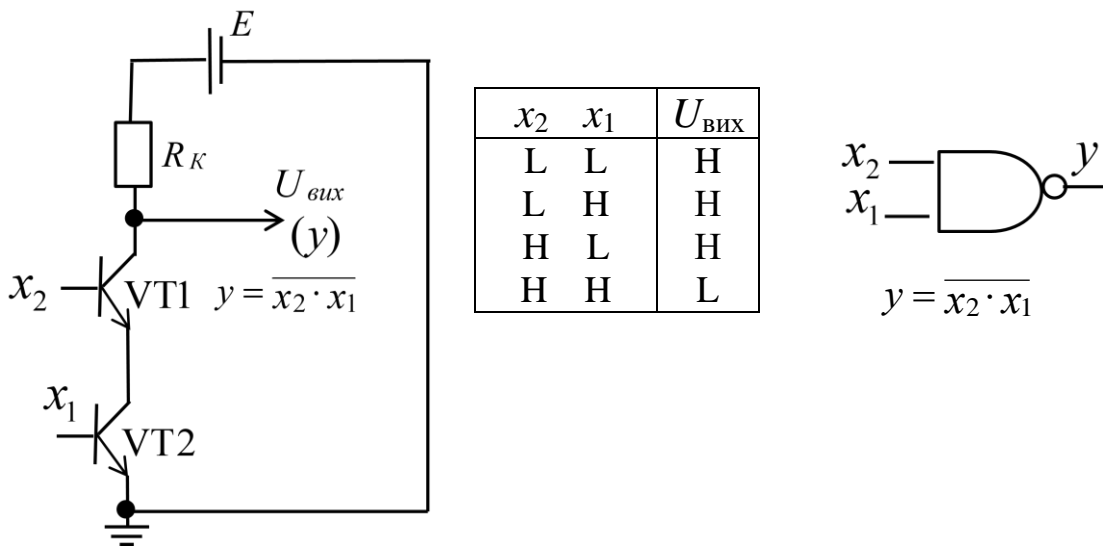


Рис. 1.27. Транзисторна структура, що реалізує логічний елемент 2І-НЕ

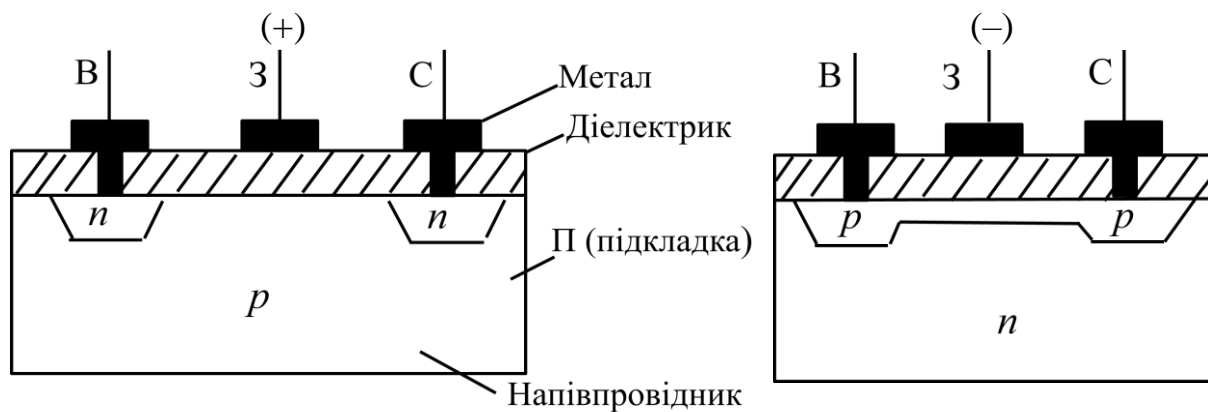
Послідовне з'єднання біполярних n - p - n транзисторів відтворює операцію кон'юнкції, паралельне – операцію диз'юнкції; при цьому спільне емітерне навантаження забезпечує пряме значення операції, а спільне колекторне навантаження – інверсне значення операції.

Розглянемо реалізацію логічних елементів на основі уніполярних (польових) транзисторів.

Уніполярні (польові) транзистори з ізольованим заслоном поділяють на дві групи – з індукційованим каналом та з вбудованим каналом (рис. 1.28).

Узагальнена назва уніполярних (польових) транзисторів – МДН-транзистори (метал-діелектрик-напівпровідник). Якщо діелектриком є оксид кремнію, то польові транзистори мають назву МОН-транзистори.

В обчислювальній техніці використовують МДН-транзистори як n -типу, так і p -типу (рис. 1.29).



n-типу з індукційованим каналом

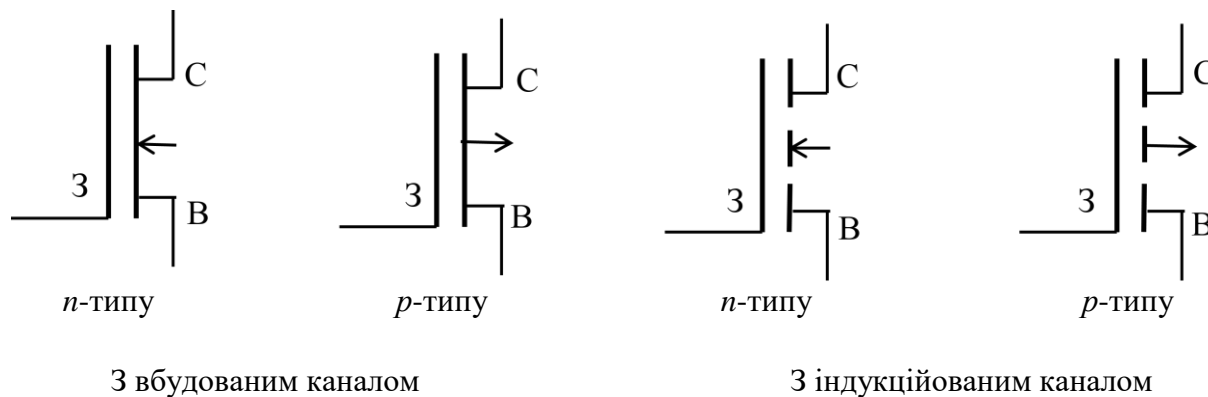
p-типу з вбудованим каналом

Рис. 1.28. Уніполярний (польовий) транзистор (у розрізі)

МДН-транзистори мають три зовнішні електроди: витік (В) – електрод, від якого починають рух носії заряду; стік (С) – електрод, до якого рухаються заряди; заслін (З) – електрод, на який подають напругу керування.

Металевий заслін З ізолюваний від каналу тонким шаром діелектрика завтовшки $1000-2000 \text{ \AA}$. Відстань між стоком та витокком становить $5 - 50 \text{ мкм}$. МДН-транзистори мають високий вхідний опір, на відміну від біполярних. Це зумовлено тим, що при подачі на заслін напруги керування струм через заслін не протікає. Саме через це вважають, що керування в МДН-транзисторах здійснюється напругою, а не струмом – як у біполярних транзисторах.

МДН-транзистори *p*-типу виготовляють на підкладці з кремнію типу *n*, у якій дифузією акцепторних домішок створені сильно-леговані *p*-області стоку С та витокку В; МДН-транзистори *n*-типу виготовляють на підкладці з кремнію типу *p*, а під стоком та витокком формують *n*-області (рис. 1.28).



З вбудованим каналом

З індукційованим каналом

Рис. 1.29. Умовні графічні позначення МДН-транзисторів

Керування в транзисторах n -типу здійснюють позитивною напругою, а в транзисторах p -типу – негативною (див. рис. 1.28). Відлік напруги між електродами транзистора здійснюють від витoku В.

Надалі розглядатимемо МДН-транзистори n -типу з індукційованим каналом.

Транзистор з індукційованим каналом у початковому стані, коли $U_{зв} = 0$ ($U_{зв}$ – напруга між витком В та заслоном З), практично не має провідності, тобто канал у ньому відсутній, і струм між витком В та стоком С не протікає (транзистор закритий). Але оскільки в напівпровіднику p -типу завжди присутня невелика кількість електронів, то при прикладанні до заслону З напруги позитивної полярності і досягненні нею деякого порогового значення електрони будуть притягуватись до діелектрика, і біля напівпровідника p -типу утвориться тонкий шар, який і утворює провідний канал між витком В і стоком С (рис. 1.30). При збільшенні напруги $U_{зв}$ (позитивної полярності) струм між витком В і стоком С збільшується (транзистор починає відкриватися).

Мінімальну напругу на заслоні, при якій між витком В і стоком С з'являється струм, називають порогом. Для МДН-транзисторів n -типу з індукційованим каналом поріг становить приблизно 3 В (рис. 1.31).

МДН-транзистори мають лінійну область на ВАХ. У цій області (область P) зміна напруги стік-витік ($U_{св}$) приводить до лінійної зміни струму між витком та стоком ($I_{св}$). Це відбувається коли $U_{св} \ll U_{зв}$. Між стоком С та витком В індукціюється канал, а в районі стоку та витку формуються збіднені області (показані пунктиром на рис. 1.30).

Подальше збільшення $U_{св}$ приводить до зміни форм збіднених областей. Транзистор переходить в режим насичення (область M на рис. 1.31) і повністю відкривається. У режимі насичення збільшення $U_{св}$ не приводить до істотного збільшення струму стік – витік.

Подальше збільшення $U_{св}$ призводить до того, що збіднені області перекривають канал, і тоді незначне зростання $U_{св}$ спричиняє лавиноподібний струм між стоком С і витком В та до руйнування (пробою) транзистора (область N на ВАХ). Пробій транзистора може виникнути також при подачі на заслін високої напруги, що призведе до руйнування діелектрика.

Оскільки ВАХ має лінійну область, то транзистор у цьому режимі може використовуватись в схемах як резистор, що забезпечує однорідність схеми.

На основі МДН-транзисторів також можна будувати логічні елементи.

Розглянемо реалізацію інвертора (елемента НЕ) на основі МДН-транзистора n -типу з індукційованим каналом (рис. 1.32).

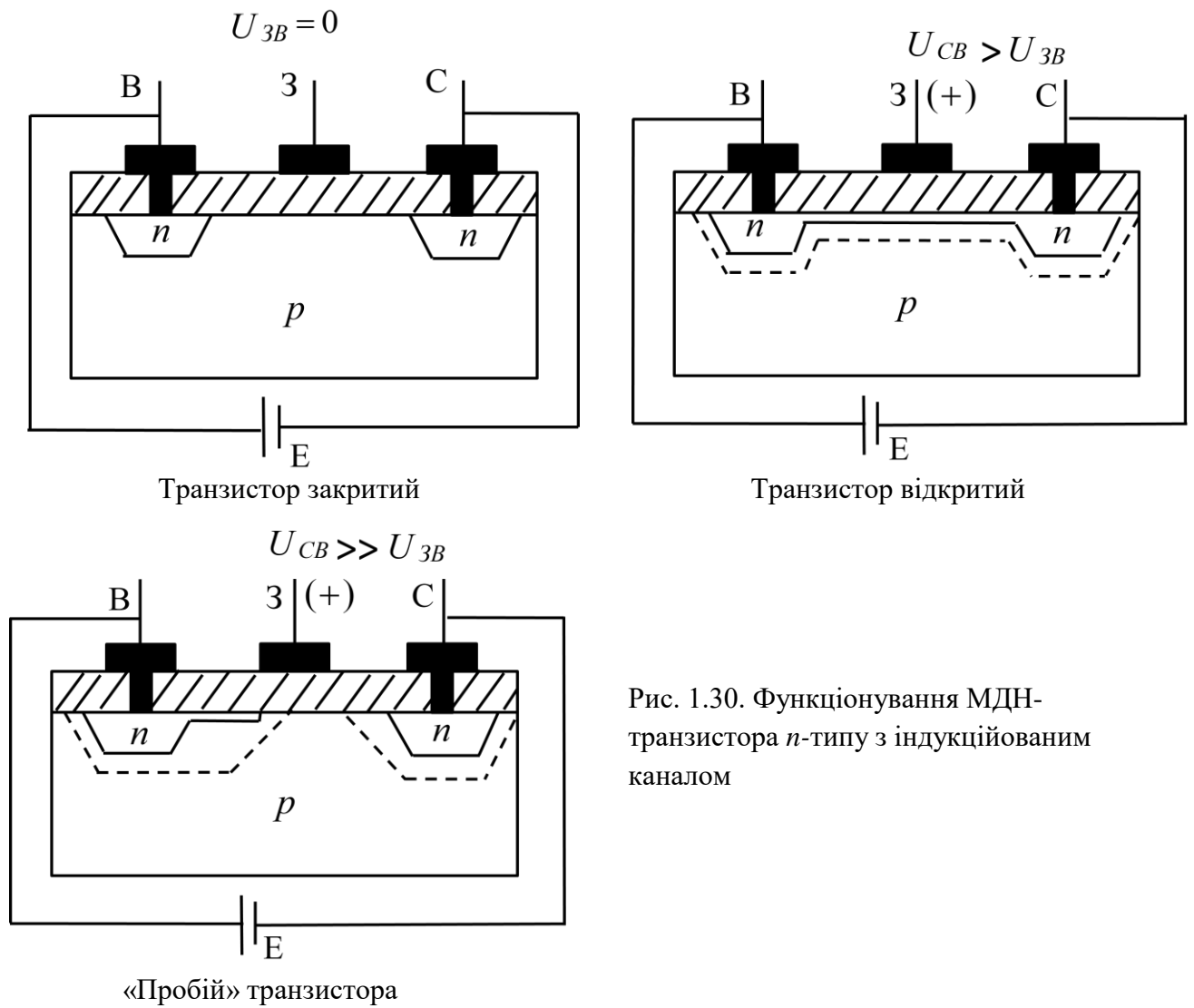


Рис. 1.30. Функціонування МДН-транзистора n -типу з індукційованим каналом

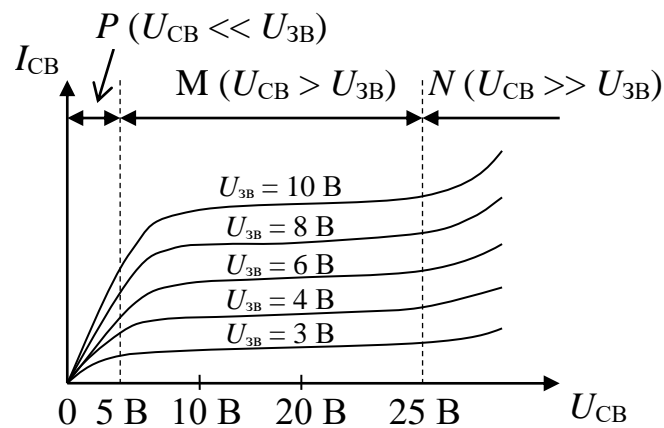


Рис. 1.31. Вольт-амперна характеристика МДН-транзистора n -типу з індукційованим каналом

Інвертор складається з активного (VT2) та навантажувального (VT1) транзисторів. Транзистор VT1 використовується як резистор.

Логічну одиницю кодують позитивною напругою U_1 , величина якої перевищує порогову напругу транзистора, а логічний нуль – позитивною напругою U_0 , меншою за значення порогової напруги.

При $U_{\text{вх}} = U_1$ ($x = 1$) транзистор VT2 відкривається, а транзистор VT1 також проводить струм, оскільки між його заслоном 3 та витком В буде діяти напруга $E - U_0$, що має перевищувати значення порогової напруги.

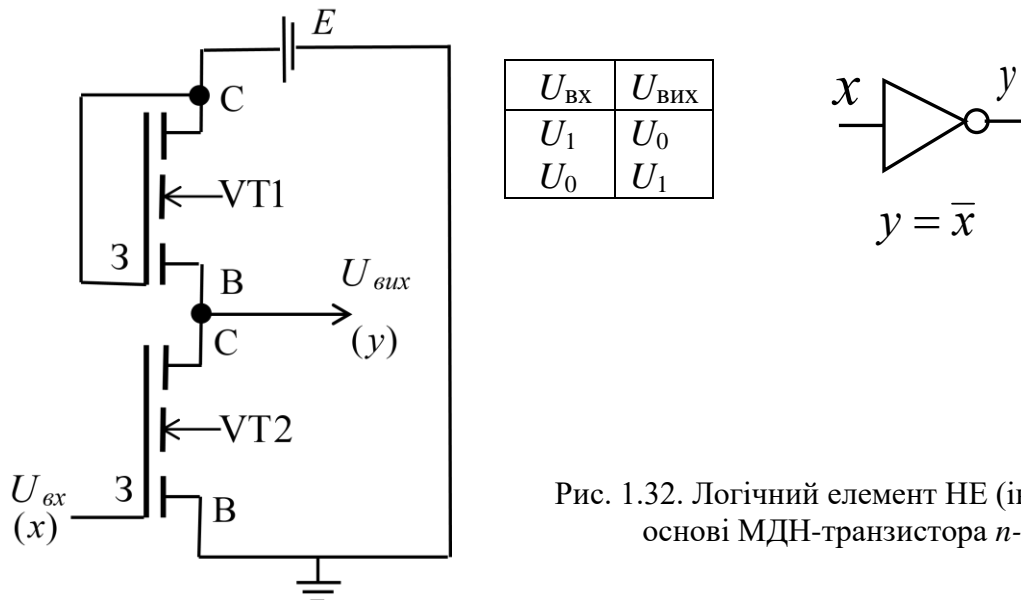


Рис. 1.32. Логічний елемент НЕ (інвертор) на основі МДН-транзистора n -типу

Вихідна напруга $U_{\text{вих}}$ знімається з діляника, утвореного опорамі каналів провідних транзисторів VT2 та VT1, і дорівнює

$$U_{\text{вих}} = U_0 = E \frac{R_{\text{VT2}}}{R_{\text{VT2}} + R_{\text{VT1}}},$$

де R_{VT2} та R_{VT1} – опори каналів провідних транзисторів VT2 і VT1.

При $U_{\text{вх}} = U_0$ ($x = 0$) транзистор VT2 закритий, а на виході має місце високий рівень (U_1) напруги, що кодує логічну одиницю.

Таким чином, схема на рис. 1.32 реалізує логічний елемент НЕ (інвертор).

Застосувавши паралельне з'єднання двох активних транзисторів (VT2, VT3) отримаємо схему, що реалізує двовходовий логічний елемент АБО-НЕ (рис. 1.33).

Послідовне з'єднання двох активних транзисторів (VT2, VT3), як показано на рис. 1.34, дозволяє отримати двовходовий логічний елемент І-НЕ.

В обох схемах VT1 – навантажувальний транзистор, що виконує функцію резистора.

За аналогією можна побудувати транзисторні схеми, що реалізують логічні елементи І та АБО.

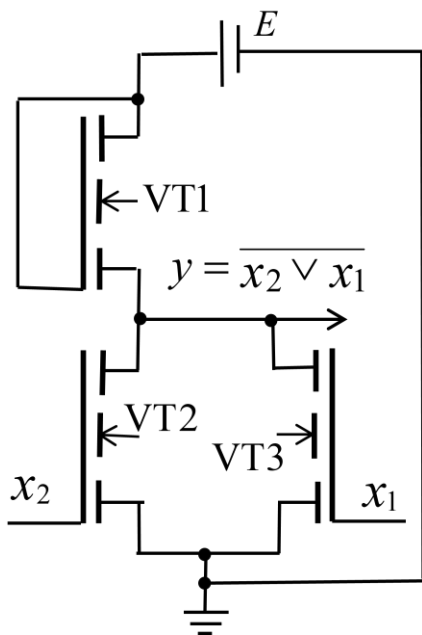


Рис. 1.33. Схема логічного елемента 2АБО-НЕ на основі МДН-транзисторів

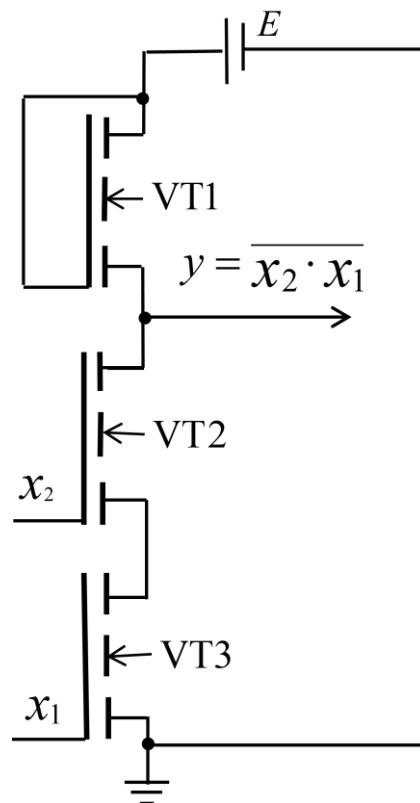


Рис. 1.34. Схема логічного елемента 2І-НЕ на основі МДН-транзисторів

Збільшуючи кількість транзисторів у відповідних структурах паралельного та послідовного з'єднання транзисторів можна отримувати відповідні логічні елементи з потрібною кількістю входів.

Фізично логічні елементи виготовляють лише у складі мікросхеми.

Наприклад, якщо у деякій логічній схемі необхідно використати інвертор, то слід розуміти, що окремо взятий інвертор є складовою частиною мікросхеми, яка містить, наприклад, 6 інверторів (рис. 1.35).

Або, наприклад, якщо необхідно використати 2-входовий логічний елемент АБО-НЕ, то слід розуміти, що елемент 2АБО-НЕ є складовою частиною мікросхеми, яка містить, наприклад, 4 елементи 2АБО-НЕ (рис. 1.36).

Для того, щоб фізично (на друкованій платі) реалізувати логічну схему, яка складається з деякої сукупності логічних елементів, необхідно розбити логічну схему на мікросхеми (корпуси ІМС) – наприклад, 6 інверторів утворюють одну мікросхему (корпус), 4 двовходові логічні елементи АБО-НЕ – іншу мікросхему (корпус), і т.д. Конструктор електронно-обчислювальної апаратури оперує лише з корпусами ІМС, які є мінімальними конструктивними одиницями.

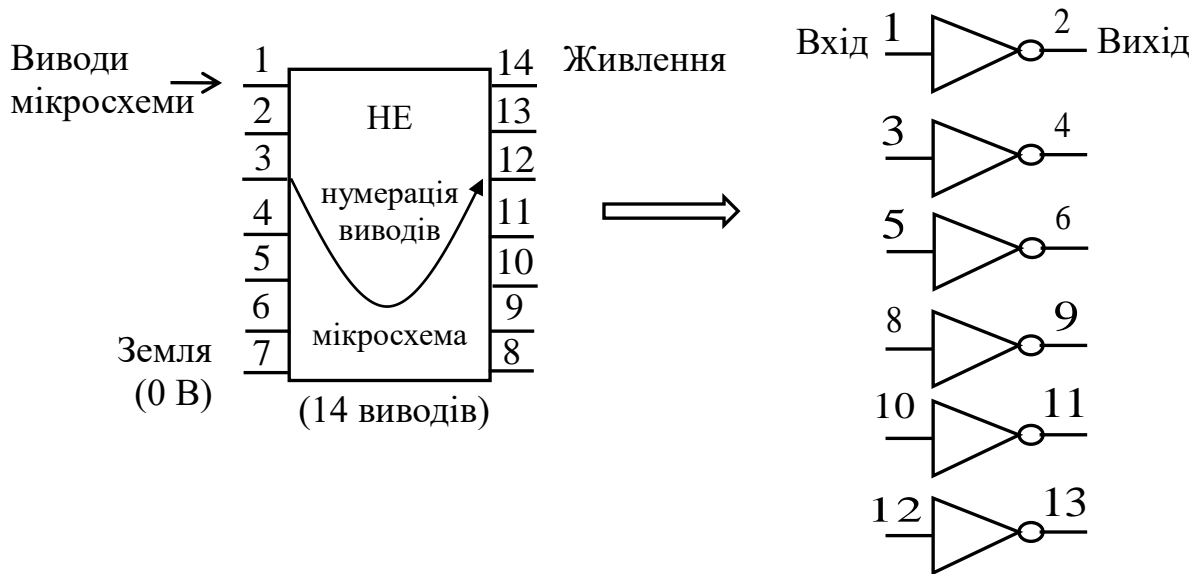


Рис. 1.35. Мікросхема, що містить шість інверторів

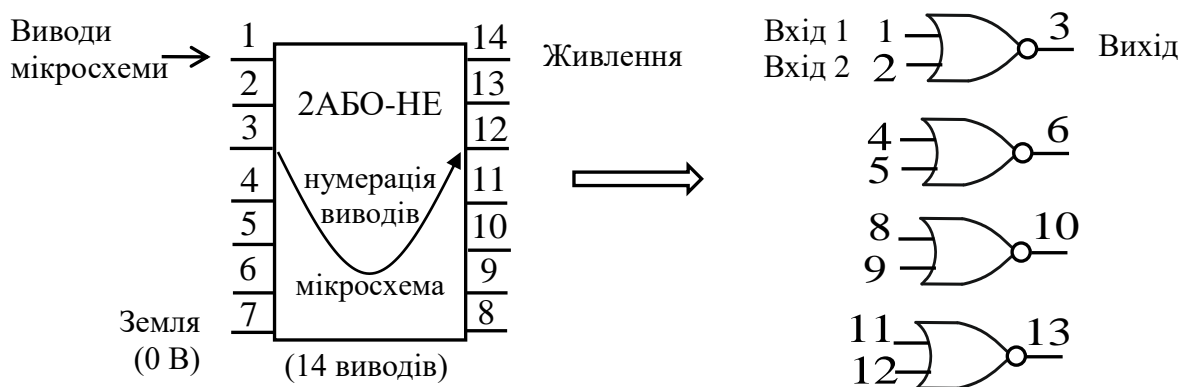


Рис. 1.36. Мікросхема, що містить двохходові логічні елементи АБО-НЕ

Запитання та завдання

1. Дайте визначення напівпровідникової інтегральної мікросхеми.
2. Яким показником характеризують функціональну складність мікросхем?
3. Який вигляд має вольт-амперна характеристика $p-n$ переходу?
4. Чим відрізняються біполярні транзистори від уніполярних (польових)?
5. Нарисуйте структуру біполярного $n-p-n$ транзистора та його умовне графічне позначення.
6. Наведіть електричну схему транзисторної структури на основі біполярного $n-p-n$ транзистора, що реалізує логічний елемент НЕ.

7. Наведіть електричну схему транзисторної структури на основі біполярного *n-p-n* транзистора, що реалізує логічний елемент 2АБО, та поясніть її функціонування.
8. Поясніть принцип роботи уніполярного (польового) транзистора.
9. Поясніть вольт-амперну характеристику МДН-транзистора *n*-типу з індукційованим каналом.
10. Наведіть електричну схему логічного елемента НЕ на основі МДН-транзистора *n*-типу з індукційованим каналом.

1.4. Принцип суперпозиції в теорії перемикальних функцій

Елементарні перемикальні функції є одно- та двомісними, тобто залежать лише від однієї або двох змінних (аргументів). А вузли, блоки пристрої комп'ютера можуть мати значно більше входів, тобто для їх опису (синтезу, аналізу) потрібно використовувати перемикальні функції від довільної кількості змінних (аргументів). Тому постає питання – як можна утворювати перемикальні функції від 3-х і більше аргументів маючи лише функції від одного та двох аргументів; або як функції від 3-х і більше аргументів можна пов'язати з одно- та двомісними перемикальними функціями.

Це можна реалізувати використовуючи принцип суперпозиції перемикальних функцій, який полягає в наступному.

Особливістю перемикальних функцій є те, що *область значень функції співпадає з множиною допустимих значень аргументів*, тобто для функції

$$y = f(x_n, \dots, x_1) \quad y, x_i \in \{0, 1\}, \quad i = 1, \dots, n.$$

Це означає, що деяку перемикальну функцію можна підставити замість аргументу в іншу перемикальну функцію.

Підстановку одних перемикальних функцій замість аргументів в іншу перемикальну функцію називають *суперпозицією перемикальних функцій*.

Принцип суперпозиції (підстановки одних перемикальних функцій замість аргументів в іншу перемикальну функцію) означає, що перемикальна функція від більшої кількості аргументів може бути одержана на основі перемикальних функцій від меншої кількості аргументів. Звідси випливає висновок – з одно- та двомісних перемикальних функцій можна утворити довільну перемикальну функцію від будь-якої кількості змінних.

Розглянемо два приклади застосування принципу суперпозиції перемикальних функцій.

Приклад 1. Якщо дано деякі дві перемикальні функції від 2-х аргументів $f_1(a, b)$ та $f_2(a, b)$, то деяку перемикальну функцію $f_3(a, b, c)$ від 3-х аргументів можна побудувати так: з двох заданих функцій $f_1(a, b)$, $f_2(a, b)$ беремо, наприклад, $f_2(a, b)$; у функцію f_2 замість аргументу a підставляємо функцію f_1 , а замість аргументу b підставляємо третій аргумент c , тобто

$$f_2(a, b) \Rightarrow f_2(f_1(a, b), c) = f_3(a, b, c).$$

Або можна побудувати іншу перемикальну функцію $f_4(a, b, c)$ від 3-х аргументів, наприклад так: беремо $f_1(a, b)$; у функцію f_1 замість аргументу a підставляємо третій аргумент c , а замість аргументу b підставляємо функцію f_2 , тобто

$$f_1(a, b) \Rightarrow f_1(c, f_2(a, b)) = f_4(a, b, c).$$

Приклад 2. Нехай є система трьох перемикальних функцій від одного та двох аргументів: заперечення – $Z(\omega) = \bar{\omega}$, кон'юнкція – $K(\mu, \eta) = \mu \cdot \eta$, диз'юнкція – $D(\gamma, \delta) = \gamma \vee \delta$.

Необхідно на основі принципу суперпозиції утворити перемикальну функцію f від трьох змінних такого виду:

$$f(x_3, x_2, x_1) = x_3 \bar{x}_2 x_1 \vee \bar{x}_1.$$

Перемикальна функція f має таку саму структуру, що й перемикальна функція D .

Виконаємо наступну послідовність дій.

1. Беремо $D(\gamma, \delta) = \gamma \vee \delta$.
2. Виконуємо заміну: $\delta \leftarrow Z(x_1) = \bar{x}_1$.
3. Виконуємо заміну: $\gamma \leftarrow K(K(x_3, Z(x_2)), x_1) = (x_3 \cdot \bar{x}_2) \cdot x_1$.
4. Тоді $D(\gamma, \delta) = \gamma \vee \delta \Rightarrow D(K(K(x_3, Z(x_2)), x_1), Z(x_1)) = (x_3 \cdot \bar{x}_2) \cdot x_1 \vee \bar{x}_1 = x_3 \bar{x}_2 x_1 \vee \bar{x}_1 = f(x_3, x_2, x_1)$.

Таким чином, на основі перемикальної функції $D(\gamma, \delta)$ від 2-х аргументів з використанням перемикальної функції $K(\mu, \eta)$ від 2-х аргументів та перемикальної функції $Z(\omega)$ від одного аргументу утворено нову перемикальну функцію $f(x_3, x_2, x_1)$ від 3-х аргументів.

Суперпозиції перемикальних функцій відповідає *суперпозиція операторів логічних елементів*.

Оператор логічного елемента – це функція, яку реалізує логічний елемент з урахуванням кількості його входів.

Наприклад, оператором логічного елемента НЕ є \bar{x} , оператором 2-входового логічного елемента І (елемента 2І) є $x_2 x_1$, оператором 3-входового логічного елемента І-НЕ (елемента 3І-НЕ) є $\overline{x_3 x_2 x_1}$, логічного елемента 2АБО – $x_2 \vee x_1$.

Процес встановлення відповідності між логічними операціями та операторами 1- та 2-входових логічних елементів для перемикальної

функції від 3-х змінних $f(x_3, x_2, x_1) = x_3 \bar{x}_2 x_1 \vee \bar{x}_1$ показано на рис. 1.37, а на рис. 1.38 – комбінаційну схему, що апаратно реалізує цю функцію.

Таким чином, можемо зробити наступні висновки.

1. Суперпозиція функцій є основним принципом в теорії перемикальних функцій.
2. Можливість отримання перемикальних функцій з більшою кількістю аргументів на основі перемикальних функцій з меншою кількістю аргументів означає на практиці, що використовуючи невеликий набір логічних елементів з малою кількістю входів, можна будувати комбінаційні схеми перемикальних функцій від довільної кількості змінних.

Замість аргументу підставлено перемикальну функцію
кон'юнкції аргументів (оператор логічного елемента 2І)

Замість аргументу підставлено
перемикальну функцію
кон'юнкції аргументів (оператор
логічного елемента 2І)

$$(x_3 \cdot \bar{x}_2) \cdot x_1 \vee \bar{x}_1$$

Замість аргументу підставлено
перемикальну функцію
заперечення аргументу
(оператор логічного
елемента НЕ)

Замість аргументу підставлено
перемикальну функцію
заперечення аргументу
(оператор логічного
елемента НЕ)

Рис. 1.37. Встановлення відповідності між логічними операціями та операторами логічних елементів

Зокрема:

- з 2- та 1-місних перемикальних функцій можна утворювати довільні перемикальні функції від будь-якої кількості аргументів,
- на основі 2- та 1-входових логічних елементів можна апаратно реалізувати довільну перемикальну функцію від будь-якої кількості змінних,
- логічний елемент з більшою кількістю входів можна замінити на сукупність (певним чином з'єднаних) логічних елементів такого самого типу з меншою кількістю входів.

Саме завдяки принципу суперпозиції функцій перемикальні функції І, АБО, І-НЕ, АБО-НЕ, можна поширити на довільну кількість змінних.

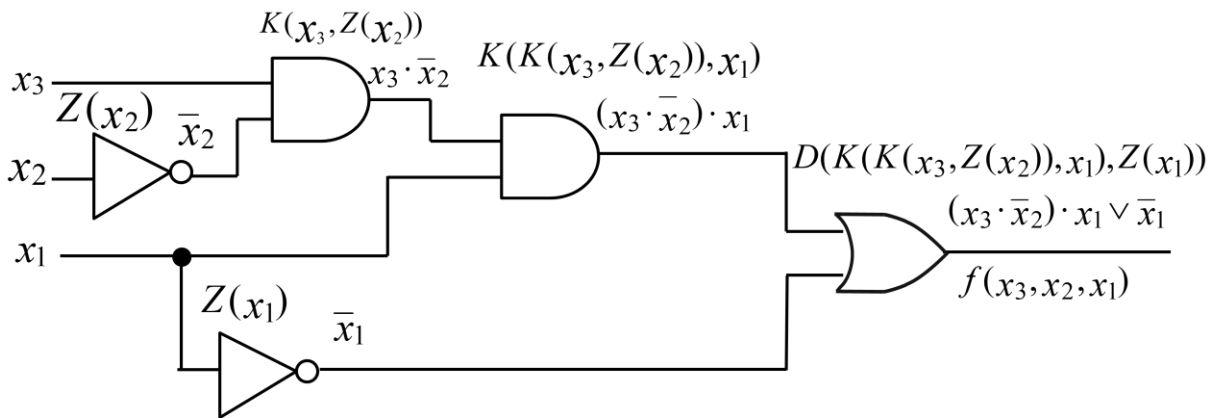


Рис. 1.38. Апаратна реалізація (комбінаційна схема) функції від 3-х змінних $f(x_3, x_2, x_1) = x_3 \bar{x}_2 x_1 \vee \bar{x}_1$ на основі 1- та 2-входових логічних елементів

Запитання та завдання

1. Що називають суперпозицією перемикальних функцій? Чому вона можлива?
2. Яку можливість дає принцип суперпозиції при побудові перемикальних функцій від багатьох змінних?
3. Як з одно- та двомісних перемикальних функцій утворити перемикальну функцію від довільної кількості змінних?
4. Що таке оператор логічного елемента?
5. Поясніть, що означає суперпозиція операторів логічних елементів.
6. Встановіть відповідність між логічними операціями та операторами логічних елементів при апаратній реалізації функції $(x_2 \bar{x}_1 \vee \bar{x}_2) x_3$.
7. Чому логічний елемент заданого типу з більшою кількістю входів можна замінити на сукупність відповідним чином з'єднаних логічних елементів такого самого типу з меншою кількістю входів?

1.5. Оцінювання логічних схем

Побудову логічних схем здійснюють на основі аналітичної форми запису перемикальної функції – формули. Але одній і тій самій перемикальній функції можуть відповідати декілька аналітичних виразів (формул) – залежно від використовуваної алгебри. Тому, для того щоб з кількох варіантів логічних схем, які реалізують одну й ту саму

перемикальну функцію, вибрати оптимальну схему, необхідно мати можливість кількісно порівнювати їх між собою.

Для кількісного оцінювання логічних схем найчастіше використовують два показники – складність схеми (структурний показник) та швидкодію схеми (часовий показник).

Складність логічної схеми оцінюють двома способами – способом Квайна (параметр K) та за кількістю умовних логічних елементів (параметр M).

За Квайном складність логічної схеми дорівнює сумарній кількості входів усіх її логічних елементів

$$K = \sum_{i=1}^q p_i,$$

де q – кількість логічних елементів, що входять до складу логічної схеми, p_i – кількість входів i -го логічного елемента.

Наприклад, складність за Квайном комбінаційної схеми на рис. 1.39а становить $K = 11$, а комбінаційної схеми на рис. 1.39б – $K = 7$.

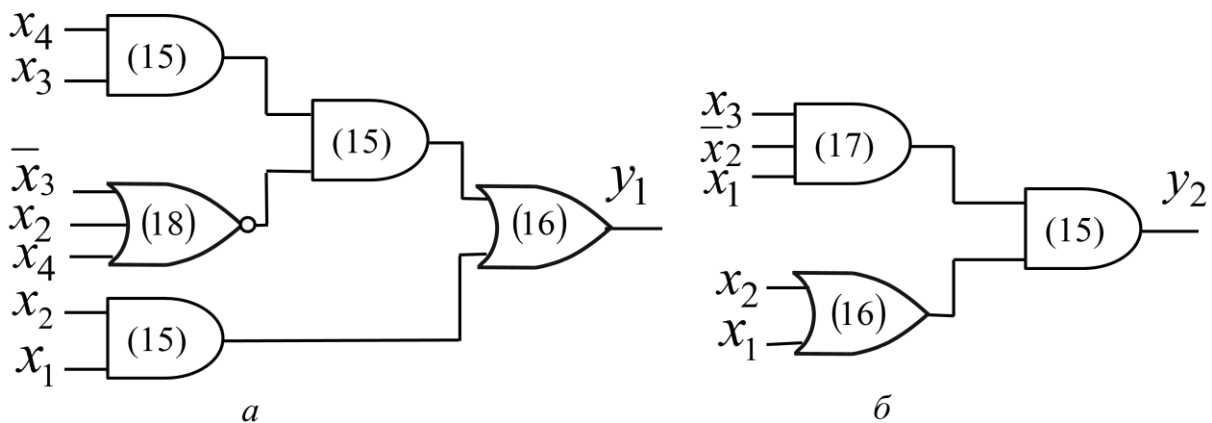


Рис. 1.39. Приклади комбінаційних схем

Складність логічної схеми за кількістю умовних логічних елементів (ЛЕ) дорівнює

$$M = K/v \text{ [} v\text{-входових умовних ЛЕ]},$$

де v – кількість входів умовного логічного елемента. За умовний зазвичай беруть 2-входовий логічний елемент ($v = 2$) або 3-входовий логічний елемент ($v = 3$).

Наприклад, складність комбінаційної схеми на рис. 1.39а дорівнює

$$M = 11/2 = 5,5 \text{ [2-входових умовних ЛЕ]} \\ \text{або } M = 11/3 = 3,7 \text{ [3-входових умовних ЛЕ]}.$$

Складність комбінаційної схеми на рис. 1.39б становить
 $M = 7/2 = 3,5$ [2-входових умовних ЛЕ]
або $M = 7/3 = 2,3$ [3-входових умовних ЛЕ].

Параметри складності K , M використовують під час проектування інтегральних мікросхем для оцінювання складності різних варіантів побудови з метою вибору оптимальної логічної схеми.

Швидкодія логічної схеми залежить від затримки сигналів логічними елементами, що входять до її складу. Логічний елемент внаслідок своєї фізичної природи (транзисторна структура) вносить затримку в поширення сигналів від його входів до виходу.

Під затримкою сигналів логічним елементом розуміють час $\tau_{ЛЕ}$ переходу сигналу на виході логічного елемента з одного логічного рівня до іншого від моменту зміни значення сигналів на входах елемента, які викликають цей перехід.

Кожний тип логічних елементів характеризується конкретним значенням часу затримки сигналів, який вимірюють в наносекундах. Затримка також може залежати й від кількості входів логічного елемента.

Якщо до складу логічної схеми входять логічні елементи з різною затримкою, то в схемі визначають шлях, що потребує максимального часу поширення сигналів від входів до виходу логічної схеми.

Нехай затримки сигналів логічними елементами, що входять до складу комбінаційних схем на рис. 1.39, становлять: $\tau_{21} = 15$ нс, $\tau_{ЗАБО-НЕ} = 18$ нс, $\tau_{АБО} = 16$ нс, $\tau_{31} = 17$ нс. Тоді швидкодія комбінаційних схем дорівнює: на рис. 1.39а (КС1) – $t_{КС1} = \tau_{ЗАБО-НЕ} + \tau_{21} + \tau_{АБО} = 18 + 15 + 16 = 49$ (нс); на рис. 1.39б (КС2) – $t_{КС2} = \tau_{31} + \tau_{21} = 17 + 15 = 32$ (нс).

Швидкодію логічної схеми, побудованої на основі однотипових логічних елементів, визначають як $t_{ЛС} = L \cdot \tau_{ЛЕ}$, де L – кількість логічних елементів, що входять у максимальний за довжиною ланцюжок елементів логічної схеми, $\tau_{ЛЕ}$ – затримка сигналів логічним елементом.

Мінімальний період T зміни сигналів на входах логічної схеми визначають з урахуванням максимальної затримки поширення сигналів у схемі, він має задовольняти умову $T > t_{ЛС}$, де $t_{ЛС}$ – швидкодія логічної схеми (максимальна затримка сигналів). Виходячи з цього визначають максимально допустиму частоту F зміни наборів сигналів на входах логічної схеми як $F = 1 / T$.

Задача 1.1. Оцінити складність за Квайном логічної схеми на рис. 1.40 та можливу частоту зміни вхідних сигналів на її входах, якщо $\tau_{21-НЕ} = 20$ нс, $\tau_{21} = 24$ нс, $\tau_{2АБО} = 22$ нс.

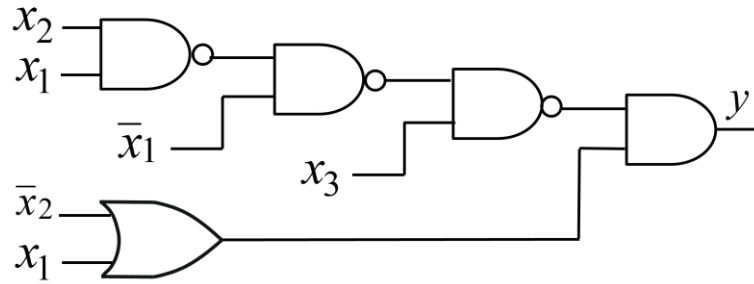


Рис. 1.40. Приклад комбінаційної схеми до задачі 1.1

Розв'язування.

1. Складність за Квайном: $K = 10$ (загальна кількість входів усіх логічних елементів).
2. Швидкодія: $t_{\text{ЛС}} = 3 \tau_{2\text{-НЕ}} + \tau_{2\text{I}} = 3 \cdot 20 \text{ нс} + 24 \text{ нс} = 84 \text{ нс}$.
3. Період T зміни сигналів на входах схеми: $T > 84 \text{ нс}$.
4. Нехай $T = 100 \text{ нс}$. Тоді допустима частота зміни наборів сигналів на входах логічної схеми становитиме:

$$F = 1 / T = 1 / (100 \text{ нс}) = 1 / (100 \cdot 10^{-9} \text{ с}) = 10^7 \text{ Гц} = 10 \text{ МГц.} \blacksquare$$

Надалі символом \blacksquare щоразу позначатимемо завершення тексту розв'язування задачі.

Запитання та завдання

1. З якою метою необхідно оцінювати логічні схеми?
2. Якими показниками оцінюють логічні схеми?
3. Якими способами оцінюють складність логічних схем?
4. Сформулюйте спосіб Квайна оцінювання складності логічної схеми. Яку розмірність має параметр K складності?
5. У чому полягає спосіб оцінювання складності логічних схем за кількістю умовних логічних елементів? Які логічні елементи обирають за умовні? Яку розмірність має параметр M складності?
6. Від чого залежить швидкодія логічної схеми?
7. В яких одиницях вимірюють швидкодію логічної схеми?
8. Як визначають швидкодію логічної схеми, якщо до її складу входять: а) логічні елементи з різною затримкою; б) однотипові логічні елементи (з однаковою затримкою)?
9. Яким має бути мінімальний період T зміни сигналів на входах логічної схеми, якщо відома швидкодія $t_{\text{ЛС}}$ логічної схеми?
10. Максимальна затримка сигналів у логічній схемі становить 75 нс . Яким має бути значення періоду T зміни сигналів на входах логічної схеми (вибрати правильну відповідь): а) $T = 75 \text{ нс}$; б) $T = 70 \text{ нс}$; в) $T > 75 \text{ нс}$?

2.

АЛГЕБРИ ПЕРЕМИКАЛЬНИХ ФУНКЦІЙ

Для опису процесів у комп'ютерних системах на логічному та функціональному рівнях застосовують апарат дискретних алгебр – алгебр перемикальних функцій. Дискретну алгебру задають двома множинами – множиною елементів $B = \{0, 1\}$ та множиною логічних операцій Ω , визначених на множині елементів B .

У теорії перемикальних функцій найбільшого поширення набули 4 алгебри: Буля, Шеффера, Пірса, Жегалкіна. Усі перелічені алгебри ґрунтуються на одній і тій самій множині дискретних елементів $B = \{0, 1\}$, але використовують різні множини Ω логічних операцій на множині елементів. Для кожної з алгебр розроблено логічні елементи, що апаратно реалізують відповідні логічні операції.

2.1. Алгебра Буля

У 1854 р. англійський математик Дж. Буль (George Boole (1815-1864)) сформулював основні закони двозначної логіки або алгебри логіки, які ґрунтуються на застосуванні логічних операцій кон'юнкції, диз'юнкції та заперечення. Алгебру Буля використовують в теорії множин та теорії ймовірностей. Особливо широкого застосування алгебра Буля набула в теорії перемикальних функцій.

Для перетворення аргументів (змінних) та виразів в алгебрі Буля використовують 3 логічні операції (функції) – І, АБО, НЕ (AND, OR, NOT). Операція НЕ – одномісна, операції І, АБО – n -місні.

У загальному випадку систему (множину) операцій (функцій) в алгебрі Буля можна записати у вигляді:

$$\begin{cases} f_1 = x_n \cdot x_{n-1} \cdot \dots \cdot x_1 & \text{(AND)} & \text{І} \\ f_2 = x_n \vee x_{n-1} \vee \dots \vee x_1 & \text{(OR)} & \text{АБО} \\ f_3 = \bar{x}_i, \quad i = 1, 2, \dots, n & \text{(NOT)} & \text{НЕ} \end{cases}$$

Алгебра Буля ґрунтується на наступних аксіомах:

$$\begin{array}{lll} x \cdot 0 = 0, & x \vee 0 = x, & \bar{\bar{x}} = x, \\ x \cdot 1 = x, & x \vee 1 = 1, & \overline{0} = 1, \\ x \cdot x = x, & x \vee x = x, & \overline{1} = 0. \\ x \cdot \bar{x} = 0, & x \vee \bar{x} = 1, & \end{array}$$

Аксіоми $x \cdot x = x$, $x \vee x = x$, які називають аксіомами повторення (тавтології, ідемпотентності), можна узагальнити на будь-яку кількість появи змінної x

$$x \cdot x \cdot \dots \cdot x = x, \quad x \vee x \vee \dots \vee x = x.$$

Аксіома *подвійного заперечення* ($\overline{\overline{x}} = x$) означає, що в логічному виразі члени, які мають подвійне заперечення, можна замінити їх початковим значенням (без заперечення) (рис. 2.1).

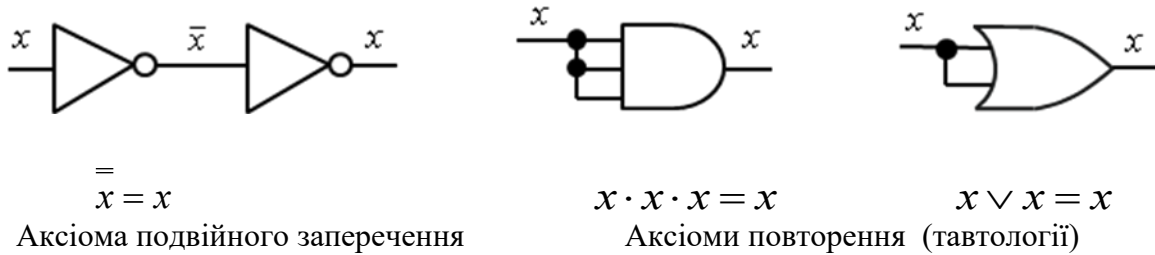
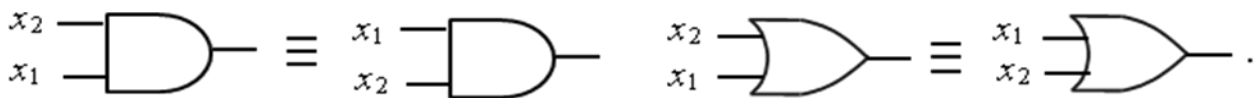


Рис. 2.1. Апаратна реалізація деяких аксіом алгебри Буля

Розглянемо основні закони (властивості) алгебри Буля.

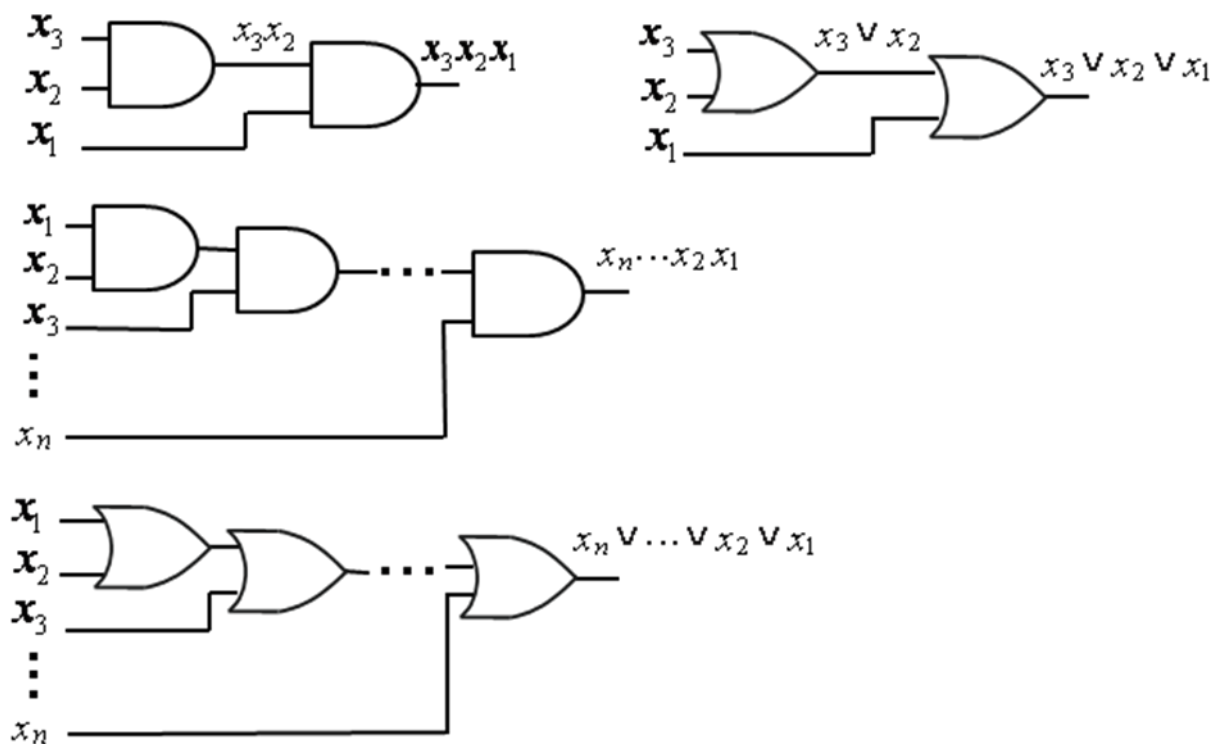
1. *Закон комутативності:* $x_2 \cdot x_1 = x_1 \cdot x_2$,
 (переставний закон) $x_2 \vee x_1 = x_1 \vee x_2$.

На практиці закон комутативності означає, що всі входи логічних елементів І та елементів АБО рівноцінні, і вхідні змінні можна подавати на будь-які входи цих елементів:



2. *Закон асоціативності:* $x_3 x_2 x_1 = (x_3 x_2) x_1$,
 (сполучний закон) $x_3 \vee x_2 \vee x_1 = (x_3 \vee x_2) \vee x_1$.

На практиці властивість асоціативності (розставляння дужок) означає, що за рахунок каскадування (каскадного з'єднання) логічних елементів можна реалізувати функції І та АБО від довільної кількості аргументів, використовуючи 2-входові логічні елементи І, АБО (рис. 2.2).



Суперпозиція операторів логічних елементів

Рис. 2.2. Апаратна реалізація кон'юнкції та диз'юнкції від довільної кількості змінних на основі 2-входових логічних елементів (властивість асоціативності)

Крім того, багатовходовий логічний елемент (ЛЕ) можна замінити на групу (сукупність) відповідним чином з'єднаних однотипових елементів з меншою кількістю входів (рис. 2.3, 2.4).

При заміні 5-входового логічного елемента АБО на сукупність 4-входових логічних елементів АБО (рис. 2.4), крім властивості асоціативності, застосована аксіома повторення (тавтології), згідно з якою $x \vee x \vee \dots \vee x = x$ або $x = x \vee x \vee \dots \vee x$.

3. Закон дистрибутивності (розподільний закон):

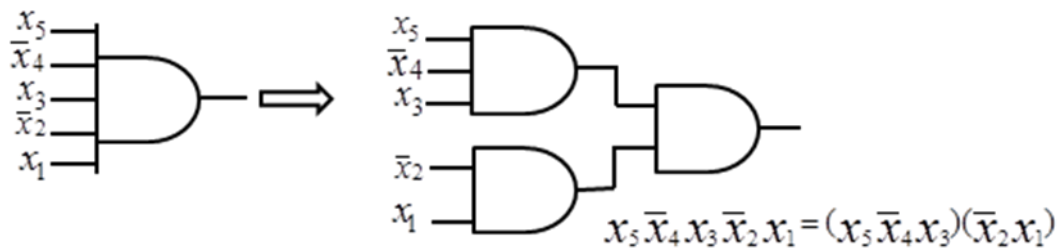
для кон'юнкції відносно диз'юнкції: $x_3(x_2 \vee x_1) = x_3x_2 \vee x_3x_1$,

для диз'юнкції відносно кон'юнкції: $x_3 \vee x_2x_1 = (x_3 \vee x_2)(x_3 \vee x_1)$.

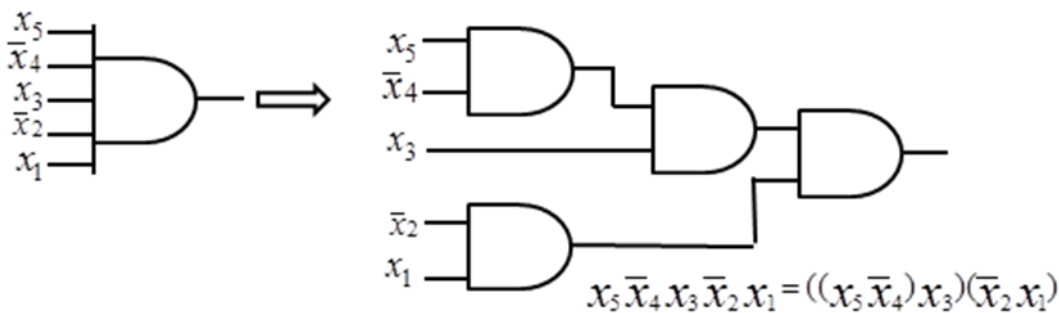
Доведемо властивість дистрибутивності для диз'юнкції відносно кон'юнкції.

Довести, що $x_3 \vee x_2x_1 = (x_3 \vee x_2)(x_3 \vee x_1)$

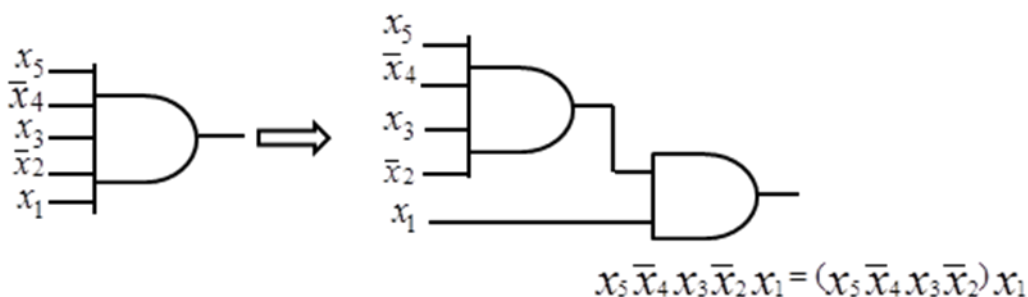
або $(x_3 \vee x_2)(x_3 \vee x_1) = x_3 \vee x_2x_1$.



На основі 3- та 2- входових ЛЕ



На основі 2- входових ЛЕ



На основі 4- та 2- входового ЛЕ

Рис. 2.3. Заміна 5-входового логічного елемента І на сукупність елементів І з меншою кількістю входів

Доведення:

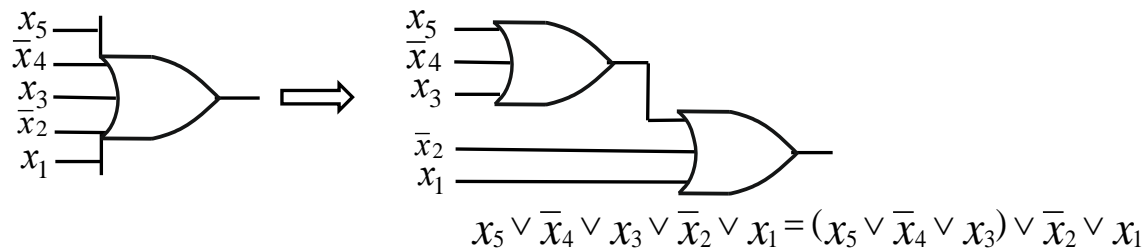
$$\begin{aligned} (x_3 \vee x_2)(x_3 \vee x_1) &= x_3 \cdot x_3 \vee x_3 \cdot x_1 \vee x_3 \cdot x_2 \vee x_2 \cdot x_1 = x_3 \vee x_3 x_1 \vee x_3 x_2 \vee x_2 x_1 = \\ &= x_3(1 \vee x_1 \vee x_2) \vee x_2 x_1 = x_3 \vee x_2 x_1. \end{aligned}$$

Закон дистрибутивності визначає правило винесення за дужки змінної:

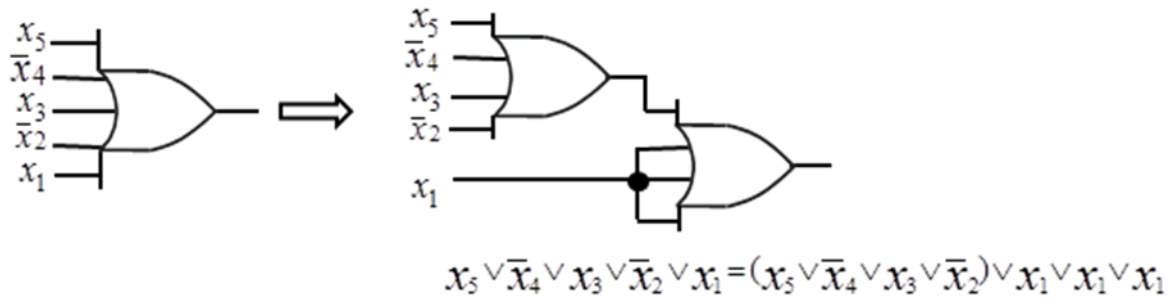
$$x_3 x_2 \vee x_3 x_1 = x_3 (x_2 \vee x_1), \quad (2.1)$$

$$(x_3 \vee x_2)(x_3 \vee x_1) = x_3 \vee x_2 x_1. \quad (2.2)$$

На практиці винесення за дужки змінної зменшує складність комбінаційних схем (рис. 2.5).



На основі 3- входових ЛЕ



На основі 4- входових ЛЕ

Рис. 2.4. Заміна 5-входового логічного елемента АБО на сукупність елементів АБО з меншою кількістю входів

Так, для апаратної реалізації лівої частини виразу (2.1) необхідно 3 логічні елементи, складність схеми за Квайном становить $K = 6$, тоді як апаратна реалізація правої частини виразу (2.1) потребує лише 2-х логічних елементів і характеризується складністю $K = 4$. Аналогічне має місце й під час апаратної реалізації виразу (2.2) (рис. 2.5).

4. Закон абсорбції:

$$x_2(x_2 \vee x_1) = x_2,$$

(поглинання)

$$x_2 \vee x_2 x_1 = x_2.$$

Властивість (співвідношення) поглинання легко довести на основі аксіом булевої алгебри:

$$x_2(x_2 \vee x_1) = x_2 x_2 \vee x_2 x_1 = x_2 \vee x_2 x_1 = x_2(1 \vee x_1) = x_2,$$

$$x_2 \vee x_2 x_1 = x_2(1 \vee x_1) = x_2.$$

5. Закон склеювання:

$$x_2 x_1 \vee x_2 \bar{x}_1 = x_2,$$

$$(x_2 \vee x_1)(x_2 \vee \bar{x}_1) = x_2.$$

Правильність цих двох тотожностей можна довести, користуючись аксіомами булевої алгебри:

$$x_2 x_1 \vee x_2 \bar{x}_1 = x_2(x_1 \vee \bar{x}_1) = x_2,$$

$$(x_2 \vee x_1)(x_2 \vee \bar{x}_1) = x_2 x_2 \vee x_2 \bar{x}_1 \vee x_2 x_1 \vee x_1 \bar{x}_1 = x_2 \vee x_2 \bar{x}_1 \vee x_2 x_1 = x_2(1 \vee \bar{x}_1 \vee x_1) = x_2.$$

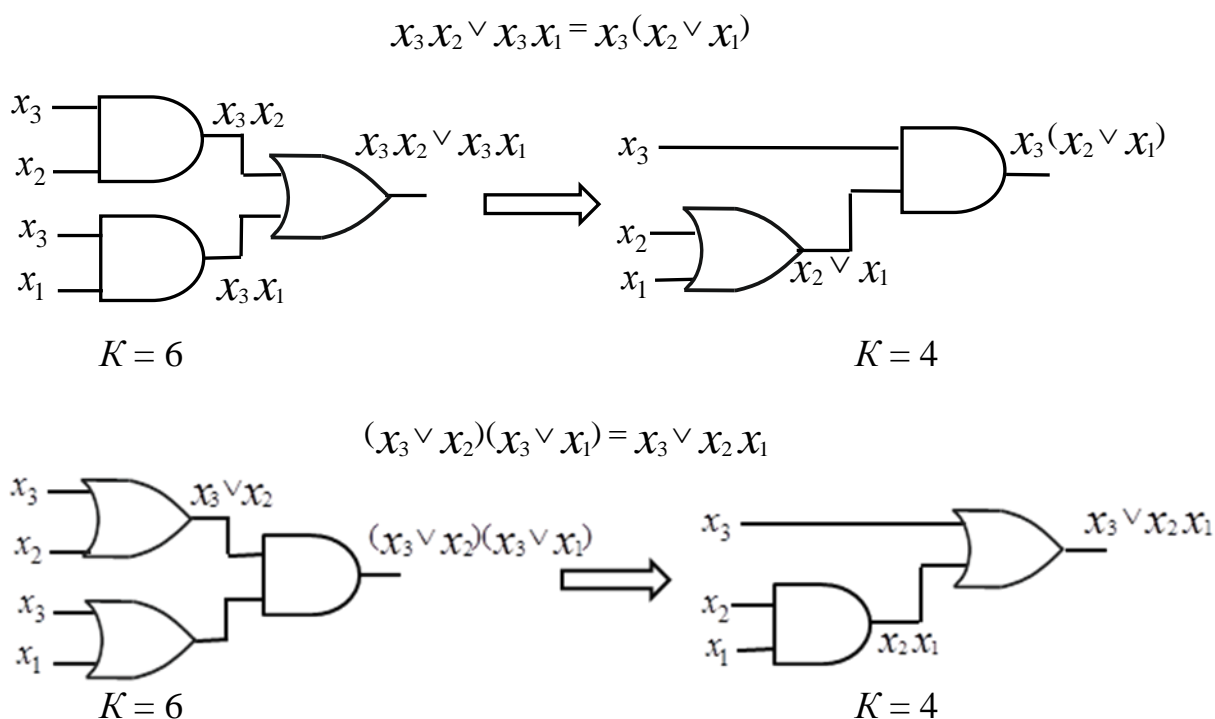


Рис. 2.5. Зменшення складності комбінаційних схем за рахунок винесення за дужки змінної (властивість дистрибутивності)

6. *Закони (правила) де Моргана (De Morgan's laws)* встановлюють зв'язок між функціями (операціями) І, АБО, І-НЕ, АБО-НЕ:

- 1) $\overline{x_2 \vee x_1} = \bar{x}_2 \cdot \bar{x}_1$ – заперечення диз'юнкції еквівалентне кон'юнкції заперечень,
(**NOR** → **AND**)
- 2) $\overline{x_2 \cdot x_1} = \bar{x}_2 \vee \bar{x}_1$ – заперечення кон'юнкції еквівалентне диз'юнкції заперечень,
(**NAND** → **OR**)
- 3) $x_2 \vee x_1 = \overline{\bar{x}_2 \cdot \bar{x}_1}$ – диз'юнкція еквівалентна кон'юнкції заперечень із загальним запереченням,
(**OR** → **NAND**)
- 4) $x_2 x_1 = \overline{\bar{x}_2 \vee \bar{x}_1}$ – кон'юнкція еквівалентна диз'юнкції заперечень із загальним запереченням.
(**AND** → **NOR**)

Правила де Моргана допускають узагальнення на випадок довільної кількості змінних:

$$\overline{x_n \vee \dots \vee x_1} = \bar{x}_n \cdot \dots \cdot \bar{x}_1,$$

$$\overline{x_n \cdot \dots \cdot x_1} = \bar{x}_n \vee \dots \vee \bar{x}_1,$$

$$x_n \cdot \dots \cdot x_1 = \overline{\bar{x}_n \vee \dots \vee \bar{x}_1},$$

$$x_n \vee \dots \vee x_1 = \overline{\bar{x}_n \cdot \dots \cdot \bar{x}_1}.$$

Наслідком правил де Моргана на практиці є те, що в логічних схемах елемент АБО можна замінити на елемент І-НЕ з інверсними входами (і навпаки), а елемент І – на елемент АБО-НЕ з інверсними входами (і навпаки) (рис. 2.6).

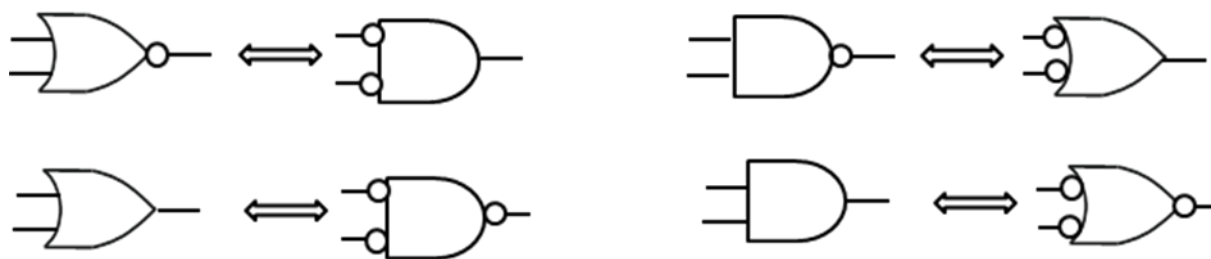


Рис. 2.6. Взаємозамінюваність логічних елементів в логічних схемах (на основі правил де Моргана)

При виконанні перетворень в аналітичних виразах булевої алгебри необхідно дотримуватись пріоритету операцій: спочатку виконують операції в дужках, далі найвищий пріоритет має операція заперечення, потім кон'юнкція, а диз'юнкція має найнижчий пріоритет. Пріоритет операцій слід враховувати при застосуванні правил де Моргана, наприклад,

$$\overline{x_3 \vee x_2 x_1} = \overline{x_3 \vee (x_2 x_1)} = \bar{x}_3 \cdot \overline{x_2 x_1} = \bar{x}_3 (\bar{x}_2 \vee \bar{x}_1).$$

Для доведення тотожностей (співвідношень) в булевій алгебрі використовують два способи:

- 1) застосовують аксіоми та закони (властивості) (рис. 2.7) для зведення лівої та правої частин співвідношення до однакового вигляду;
- 2) за допомогою таблиці відповідності.

Відповідно до другого способу будують таблицю лівої частини досліджуваного виразу та правої частини виразу. Якщо в таблиці підсумковий стовпець лівої частини виразу співпадає з підсумковим стовпцем правої частини виразу, то вважають, що вираз справджується (є справедливим).

Задача 2.1. Довести, що $\overline{\bar{x}_2 \cdot x_1} = \bar{x}_2 \vee \bar{x}_1$.

Розв'язування.

Будуємо таблицю відповідності заданого співвідношення (табл. 2.1).

Як бачимо, ліва частина виразу набуває тих самих значень, що й права, для всіх можливих значень аргументів. ■

Аксиоми

$$\begin{array}{llll}
 a \vee 0 = a, & a \cdot 1 = a, & a \vee \bar{a} = 1, & a \cdot \bar{a} = 0, \\
 a \vee 1 = 1, & a \cdot 0 = 0, & \bar{\bar{0}} = 1, & \bar{\bar{1}} = 0. \\
 a \vee a = a, & a \cdot a = a \text{ (ідемпотентність)}, & & \\
 \bar{\bar{a}} = a \text{ (подвійне заперечення)}. & & &
 \end{array}$$

Закони (властивості)

$$\begin{array}{lll}
 a \vee b = b \vee a, & a \cdot b = b \cdot a, & \text{(комутативність)} \\
 (a \vee b) \vee c = a \vee (b \vee c), & (a \cdot b) \cdot c = a \cdot (b \cdot c), & \text{(асоціативність)} \\
 a(b \vee c) = a \cdot b \vee a \cdot c, & a \vee b \cdot c = (a \vee b)(a \vee c), & \text{(дистрибутивність)} \\
 a \vee a \cdot b = a, & a(a \vee b) = a, & \text{(поглинання)} \\
 a \cdot b \vee a \cdot \bar{b} = a, & (a \vee b)(a \vee \bar{b}) = a, & \text{(склеювання)} \\
 \overline{a \vee b} = \bar{a} \cdot \bar{b}, & \overline{a \cdot b} = \bar{a} \vee \bar{b}, & \text{(правила де Моргана)} \\
 \overline{\overline{a \vee b}} = a \vee b, & \overline{\overline{a \cdot b}} = a \cdot b. &
 \end{array}$$

Рис. 2.7. Аксиоми та закони (властивості) алгебри Буля

Таблиця 2.1

Таблиця відповідності для доведення співвідношення $\overline{x_2 \cdot x_1} = \bar{x}_2 \vee \bar{x}_1$

Значення аргументів		Ліва частина виразу		Права частина виразу		
x_2	x_1	$x_2 \cdot x_1$	$\overline{x_2 \cdot x_1}$	\bar{x}_2	\bar{x}_1	$\bar{x}_2 \vee \bar{x}_1$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0



Канонічні форми в алгебрі Буля

В алгебрі Буля будь-яку перемикальну функцію можна подати (записати) у двох канонічних формах:

- у ДДНФ – досконалій диз'юнктивній нормальній формі,
- у ДКНФ – досконалій кон'юнктивній нормальній формі.

Для визначення ДДНФ та ДКНФ введемо деякі означення.

Будь-яка послідовність аргументів (змінних), об'єднаних однією операцією, називається *термом*.

Приклади термів: $x_3 x_2 x_1$, $x_3 \vee \bar{x}_1$, x_2 , \bar{x}_1 .

Змінну (пряму або із запереченням) в термі називають *буквою*.

Кількість букв у термі називають *рангом терма*. Залежно від операції, якою об'єднані змінні у термі, розрізняють диз'юнктивні та кон'юнктивні терми.

У *диз'юнктивному термі* змінні (букви) об'єднані операцією диз'юнкції, наприклад, $x_4 \vee \bar{x}_3 \vee x_2 \vee x_1$, $\bar{x}_3 \vee x_2 \vee \bar{x}_1$, $x_2 \vee \bar{x}_1$, x_3 .

У *кон'юнктивному термі* змінні (букви) об'єднані операцією кон'юнкції, наприклад, $x_3 \bar{x}_2 x_1$, $x_4 \bar{x}_3 \bar{x}_2 x_1$, $\bar{x}_4 x_1$, \bar{x}_2 .

Макстермом називають диз'юнктивний терм, до складу якого входять всі без винятку змінні функції (змінна може бути прямою або з інверсією). Для функції від 4-х змінних $f(x_4, x_3, x_2, x_1)$ макстермами є, наприклад, $x_4 \vee \bar{x}_3 \vee \bar{x}_2 \vee x_1$, $x_4 \vee x_3 \vee x_2 \vee x_1$, а вирази (терми) $x_4 \vee \bar{x}_2 \vee x_1$, $\bar{x}_4 \vee \bar{x}_1$ не є макстермами.

Мінтермом називають кон'юнктивний терм, до складу якого входять всі без винятку змінні функції (змінна може бути прямою або інверсною). Так, для функції від 3-х змінних $f(x_3, x_2, x_1)$ мінтермами є, наприклад, $x_3 x_2 x_1$, $\bar{x}_3 \bar{x}_2 x_1$, а вирази (терми) $x_3 x_2$, $\bar{x}_3 x_1$, \bar{x}_2 мінтермами не є.

Диз'юнктивною формою називають диз'юнкцію кон'юнктивних термів, у якій хоча б один з термів не є мінтермом.

Наприклад, вираз $f(x_3, x_2, x_1) = \bar{x}_3 x_2 x_1 \vee \bar{x}_3 \bar{x}_2 \vee x_3 x_2 x_1$ є диз'юнктивною формою для функції f , оскільки другий терм не містить змінну x_1 (тобто не є мінтермом).

Диз'юнктивною нормальною формою називають диз'юнкцію мінтермів.

Наприклад, вираз $f(x_4, x_3, x_2, x_1) = x_4 x_3 x_2 x_1 \vee \bar{x}_4 x_3 \bar{x}_2 x_1 \vee x_4 x_3 x_2 \bar{x}_1$ є нормальною диз'юнктивною формою, оскільки до складу кожного з термів входять всі без винятку змінні функції.

Кон'юнктивною формою називають кон'юнкцію диз'юнктивних термів, у якій хоча б один з термів не є макстермом.

Наприклад, вираз

$$f(x_4, x_3, x_2, x_1) = (x_4 \vee x_2 \vee x_1)(x_4 \vee \bar{x}_3 \vee x_2 \vee \bar{x}_1)(\bar{x}_4 \vee x_3 \vee x_2 \vee x_1)$$

є кон'юнктивною формою для функції f , оскільки перший терм не містить змінну x_3 (тобто не є макстермом).

Кон'юнктивною нормальною формою називають кон'юнкцію макстермів.

Наприклад, вираз $f(x_3, x_2, x_1) = (x_3 \vee x_2 \vee x_1)(\bar{x}_3 \vee \bar{x}_2 \vee x_1)$ є нормальною кон'юнктивною формою, оскільки до складу кожного з термів входять всі без винятку змінні функції.

Оскільки макстерм (мінтерм) містить у своєму складі всі без винятку змінні функції (прямі або з інверсією), то його можна розглядати як функцію. Макстерм називають також конституентою нуля, а мінтерм – конституентою одиниці.

Функцію від n аргументів, що набуває значення 1 лише на одному наборі значень аргументів і 0 – на решті наборів, називають *конституентою одиниці* (мінтермом).

Якщо задано, наприклад, 3 аргументи: x_3, x_2, x_1 , то можна утворити 8 функцій – конституент одиниці (мінтермів) (табл. 2.2).

У загальному випадку, якщо n – кількість аргументів, то кількість можливих конституент одиниці дорівнює кількості можливих наборів значень аргументів, тобто 2^n .

Таблиця 2.2

Можливі конституенти одиниці (мінтерми) у випадку трьох аргументів

Значення аргументів			Конституенти одиниці (мінтерми)							
x_3	x_2	x_1	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Конституенти одиниці у випадку трьох змінних записують у вигляді:

$$C_0 = \bar{x}_3\bar{x}_2\bar{x}_1, \quad C_1 = \bar{x}_3\bar{x}_2x_1, \quad C_2 = \bar{x}_3x_2\bar{x}_1, \quad C_3 = \bar{x}_3x_2x_1, \\ C_4 = x_3\bar{x}_2\bar{x}_1, \quad C_5 = x_3\bar{x}_2x_1, \quad C_6 = x_3x_2\bar{x}_1, \quad C_7 = x_3x_2x_1.$$

Між набором значень аргументів і конституентою одиниці (функцією) є взаємно-однозначна відповідність: заданому набору значень аргументів відповідає єдина конституента одиниці (функція), а заданій конституенті одиниці (функції) відповідає єдиний набір значень аргументів, на якому конституента (функція) дорівнює 1.

Наприклад, набору $\langle 1010 \rangle$ відповідає конституента одиниці (функція) $x_4\bar{x}_3x_2\bar{x}_1$, а, наприклад, конституенті одиниці (функції) $x_4x_3\bar{x}_2\bar{x}_1$ відповідає набір значень аргументів $\langle 1100 \rangle$.

При аналітичному запису конституенти одиниці у вигляді кон'юнктивного терма (мінтерма) одиничному значенню аргументу в двійковому наборі відповідає в термі буква (змінна) без заперечення, а нульовому – із запереченням, наприклад, $\langle 10010 \rangle \rightarrow x_5\bar{x}_4\bar{x}_3x_2\bar{x}_1$.

Між заданою конституентою одиниці як функцією і відповідним набором значень аргументів можна встановити взаємно-однозначну відповідність тому, що конституента одиниці є функцією, яка набуває значення 1 лише на одному наборі значень аргументів.

У загальному випадку конституента одиниці є кон'юнктивним термом n -го рангу (мінтермом):

$$C_i = x_n^{\sigma_n} \cdot x_{n-1}^{\sigma_{n-1}} \cdot \dots \cdot x_1^{\sigma_1},$$

$$\text{де } x_j = \begin{cases} x_j, & \text{якщо } \sigma_j = 1, \\ \bar{x}_j, & \text{якщо } \sigma_j = 0, \end{cases} \quad i = 0, 1, 2, \dots, 2^n - 1, \quad j = 1, 2, \dots, n,$$

$\langle \sigma_n \sigma_{n-1} \dots \sigma_1 \rangle$ – двійковий набір, на якому функція C_i набуває значення 1.

Будь-яку перемикальну функцію можна подати в досконалій диз'юнктивній нормальній формі.

Досконала диз'юнктивна нормальна форма функції (ДДНФ) – це диз'юнкція конституент одиниці (мінтермів), що відповідають наборам, на яких перемикальна функція набуває значення 1, тобто

$$F_{\text{ДДНФ}} = \bigvee_{i=0}^{2^n-1} C_i \alpha_i,$$

де \bigvee – узагальнений знак диз'юнкції, $\alpha_i \in \{0,1\}$ – значення перемикальної функції на i -му наборі аргументів.

ДДНФ – називають також формою І/АБО (AND/OR), підкреслюючи що в ДДНФ внутрішньою функцією є І, а зовнішньою – АБО.

Задача 2.2. Подати в ДДНФ перемикальні функції y_1, y_2 , задані таблицею істинності (табл. 2.3).

Таблиця 2.3
Таблиця істинності перемикальних
функцій y_1, y_2

Номер набору	Значення аргументів			Функції	
	x_3	x_2	x_1	y_1	y_2
0	0	0	0	1	0
1	0	0	1	1	1
2	0	1	0	0	1
3	0	1	1	1	1
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	0	1

Розв'язування.

Аналізуючи стовпець y_1 з табл. 2.3, бачимо, що функція y_1 набуває значення 1 на наборах з номерами 0, 1, 3, 5, 6.

Тому

$$y_{1\text{ДДНФ}} = 0 \vee 1 \vee 3 \vee 5 \vee 6 = \bar{x}_3 \bar{x}_2 \bar{x}_1 \vee \bar{x}_3 \bar{x}_2 x_1 \vee \bar{x}_3 x_2 x_1 \vee x_3 \bar{x}_2 x_1 \vee x_3 x_2 \bar{x}_1.$$

Функція y_2 набуває значення 1 на наборах з номерами 1, 2, 3, 4, 7.

Тому

$$y_{2\text{ДДНФ}} = 1 \vee 2 \vee 3 \vee 4 \vee 7 = \bar{x}_3 \bar{x}_2 x_1 \vee \bar{x}_3 x_2 \bar{x}_1 \vee \bar{x}_3 x_2 x_1 \vee x_3 \bar{x}_2 \bar{x}_1 \vee x_3 x_2 x_1. \blacksquare$$

Функцію від n аргументів, що набуває значення 0 лише на одному наборі значень аргументів і 1 – на решті наборів, називають *конституентою нуля* (макстермом).

У випадку трьох аргументів x_3, x_2, x_1 , можна утворити 8 функцій – конституент нуля (макстермів) (табл. 2.4).

Таблиця 2.4

Можливі конституенти нуля (макстерми) у випадку трьох аргументів

Значення аргументів			Конституенти нуля (макстерми)							
x_3	x_2	x_1	Z_0	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0

Якщо n – кількість аргументів, то кількість можливих конституент нуля дорівнює кількості можливих наборів значень аргументів, тобто 2^n .

При аналітичному запису конституенти нуля у вигляді диз'юнктивного терма (макстерма) нульовому значенню аргументу у двійковому наборі відповідає в термі буква (змінна) без заперечення, а одиничному – із запереченням, наприклад, $\langle 01011 \rangle \rightarrow x_5 \vee \bar{x}_4 \vee x_3 \vee \bar{x}_2 \vee \bar{x}_1$.

Конституенти нуля у випадку трьох змінних записують у вигляді:

$$Z_0 = x_3 \vee x_2 \vee x_1, \quad Z_1 = x_3 \vee x_2 \vee \bar{x}_1, \quad Z_2 = x_3 \vee \bar{x}_2 \vee x_1, \quad Z_3 = x_3 \vee \bar{x}_2 \vee \bar{x}_1,$$

$$Z_4 = \bar{x}_3 \vee x_2 \vee x_1, \quad Z_5 = \bar{x}_3 \vee x_2 \vee \bar{x}_1, \quad Z_6 = \bar{x}_3 \vee \bar{x}_2 \vee x_1, \quad Z_7 = x_3 \vee \bar{x}_2 \vee x_1.$$

Між набором значень аргументів і конституентою нуля (функцією) є взаємно-однозначна відповідність: заданому набору значень аргументів відповідає єдина конституента нуля (функція), а заданій конституенті

нуля (функції) відповідає єдиний набір значень аргументів, на якому конститuenta (функція) дорівнює нулю.

Наприклад, набору <1001> відповідає конститuenta нуля (функція) $\bar{x}_4 \vee x_3 \vee x_2 \vee \bar{x}_1$, а конститuentі нуля (функції) $x_4 \vee \bar{x}_3 \vee \bar{x}_2 \vee x_1$ – набір <0110>.

Між заданою конституентою нуля як функцією і відповідним набором значень аргументів можна встановити взаємно-однозначну відповідність тому, що конститuenta нуля є функцією, яка набуває значення 0 лише на одному наборі значень аргументів.

У загальному випадку конститuenta нуля є диз'юнктивним термом n -го рангу (макстермом):

$$Z_i = x_n^{\bar{\sigma}_n} \vee x_{n-1}^{\bar{\sigma}_{n-1}} \vee \dots \vee x_1^{\bar{\sigma}_1},$$

$$\text{де } x_j = \begin{cases} \bar{x}_j, & \text{якщо } \sigma_j = 1, \\ x_j, & \text{якщо } \sigma_j = 0, \end{cases} \quad i = 0, 1, 2, \dots, 2^n - 1, \quad j = 1, 2, \dots, n,$$

< $\sigma_n \sigma_{n-1} \dots \sigma_1$ > – двійковий набір, на якому функція Z_i набуває значення 0.

Будь-яку перемикальну функцію можна подати в досконалій кон'юнктивній нормальній формі.

Досконала кон'юнктивна нормальна форма функції (ДКНФ) – це кон'юнкція конститuent нуля (макстермів), що відповідають наборам, на яких перемикальна функція набуває значення 0, тобто

$$F_{\text{ДКНФ}} = \bigg\&_{i=0}^{2^n-1} (Z_i \vee \alpha_i),$$

де $\&$ – узагальнений знак кон'юнкції, $\alpha_i \in \{0, 1\}$ – значення функції на i -му наборі аргументів.

ДКНФ називають також формою АБО/І (OR/AND), де АБО – внутрішня функція, а І – зовнішня.

Задача 2.3. Подати в ДКНФ перемикальні функції y_1 , y_2 , задані таблицею істинності (табл. 2.3).

Розв'язування.

Аналізуючи стовпець y_1 з табл. 2.3, бачимо, що функція y_1 набуває значення 0 на наборах з номерами 2, 4, 7.

Тому $y_{1\text{ДКНФ}} = (x_3 \vee \bar{x}_2 \vee x_1)(\bar{x}_3 \vee x_2 \vee x_3)(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_1)$ або в числовому вигляді: $y_{1\text{ДКНФ}} = \bar{2} \cdot \bar{4} \cdot \bar{7}$.

Функція y_2 набуває значення 0 на наборах з номерами 0, 5, 6.

Тому $y_{2\text{ДКНФ}} = \bar{0} \cdot \bar{5} \cdot \bar{6} = (x_3 \vee x_3 \vee x_1)(\bar{x}_3 \vee x_2 \vee \bar{x}_1)(\bar{x}_3 \vee \bar{x}_2 \vee x_1)$. ▣

Між конституентою одиниці (мінтермом) та конституентою нуля (макстермом), які відповідають одному набору аргументів, є наступний зв'язок:

$$\bar{C}_i = Z_i \text{ або } C_i = \bar{Z}_i; \bar{Z}_i = C_i \text{ або } Z_i = \bar{C}_i.$$

Наприклад, нехай $C_5 = x_3 \bar{x}_2 x_1$, тоді $Z_5 = \bar{C}_5 = \overline{x_3 \bar{x}_2 x_1} = \bar{x}_3 \vee x_2 \vee \bar{x}_1$.

Якщо $Z_4 = \bar{x}_3 \vee x_2 \vee x_1$, то $C_4 = \bar{Z}_4 = \overline{\bar{x}_3 \vee x_2 \vee x_1} = x_3 \bar{x}_2 \bar{x}_1$.

Якщо є деякий набір значень аргументів, то йому відповідають дві функції – конституента одиниці (мінтерм), та конституента нуля (макстерм).

Наприклад, якщо задано набір $\langle 1100 \rangle$, то йому відповідає конституента одиниці $x_4 x_3 \bar{x}_2 \bar{x}_1$, а також конституента нуля $\bar{x}_4 \vee \bar{x}_3 \vee x_2 \vee x_1$.

У назвах канонічних форм алгебри Буля слово «досконалий» означає, що в цих формах використовуються терми n -го рангу, тобто до складу кожного терма, що бере участь в утворенні канонічної форми, входять всі без винятку аргументи функції – x_n, x_{n-1}, \dots, x_1 ; слово «нормальна» означає, що канонічна форма є дворівневою, тобто формула канонічної форми утворена двома функціями – внутрішньою і зовнішньою.

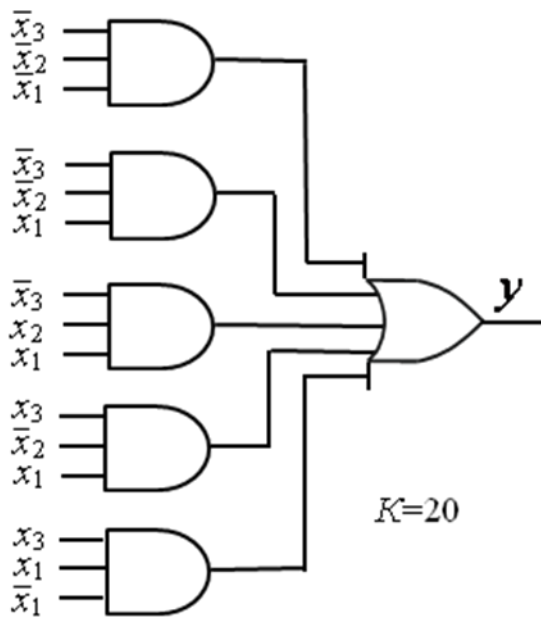
Наприклад, для ДДНФ внутрішньою функцією є кон'юнкція, а зовнішньою – диз'юнкція, тобто ДДНФ є диз'юнкцією кон'юнкцій; для ДКНФ внутрішньою функцією є диз'юнкція, а зовнішньою – кон'юнкція, тобто ДКНФ є кон'юнкцією диз'юнкцій.

На практиці властивість нормальності канонічної форми означає, що якщо немає обмежень на кількість входів логічних елементів, то комбінаційна схема, що реалізує нормальну форму, має два каскади (стовпці) логічних елементів.

Наприклад, комбінаційна схема, що реалізує функцію y_1 (табл. 2.3), подану в ДДНФ, містить такі два каскади логічних елементів: у першому – 5 логічних елементів ЗІ, у другому – один логічний елемент 5АБО (рис. 2.8а); комбінаційна схема, що реалізує цю саму функцію (y_1), але подану в ДКНФ, також містить два каскади логічних елементів: у першому – 3 логічні елементи 3АБО, у другому – один логічний елемент ЗІ (рис. 2.8б).

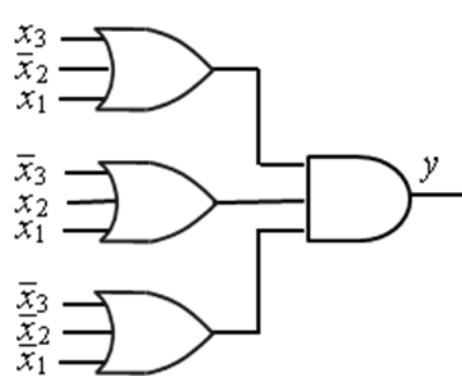
ДДНФ і ДКНФ заданої перемикальної функції f є еквівалентними, тобто $f_{\text{дднф}} \equiv f_{\text{дкнф}}$. Еквівалентність $f_{\text{дднф}}$ і $f_{\text{дкнф}}$ перемикальної функції доводять на основі аксіом та законів булевої алгебри.

$$y_{1\text{ДДНФ}} = 0 \vee 1 \vee 3 \vee 5 \vee 6$$



a

$$y_{1\text{ДКНФ}} = \bar{2} \cdot \bar{4} \cdot \bar{7}$$



b

$K=12$

Рис. 2.8. Комбінаційні схеми, що реалізують функцію y_1 з табл. 2.3, подану в: *a* – ДДНФ; *b* – ДКНФ

Нехай ϵ перемикальна функція f від двох змінних, задана таблицею істинності:

x_2	x_1	f
0	0	0
0	1	1
1	0	1
1	1	0

Доведемо, що ДДНФ та ДКНФ цієї функції є еквівалентними:

$$f_{\text{ДДНФ}} = \bar{x}_2 x_1 \vee x_2 \bar{x}_1,$$

$$f_{\text{ДКНФ}} = (x_2 \vee x_1)(\bar{x}_2 \vee \bar{x}_1) = x_2 \bar{x}_2 \vee x_2 \bar{x}_1 \vee \bar{x}_2 x_1 \vee x_1 \bar{x}_1 =$$

$$= \bar{x}_2 x_1 \vee x_2 \bar{x}_1 = f_{\text{ДДНФ}}.$$

Розглянемо отримання ДКНФ з ДДНФ.

Задача 2.4. Повністю визначену перемикальну функцію $\delta(x_2, x_1)$ від двох змінних подано в ДДНФ: $\delta_{\text{ДДНФ}} = \bar{x}_2 \bar{x}_1 \vee x_2 x_1$. Знайти її ДКНФ.

Розв'язування. Спосіб 1 (через таблицю істинності).

x_2	x_1	δ
0	0	1
0	1	0
1	0	0
1	1	1

Підставляючи в формулу $\bar{x}_2 \bar{x}_1 \vee x_2 x_1$ всі можливі набори значень аргументів x_2, x_1 отримаємо таблицю істинності функції δ . З таблиці знаходимо $\delta_{\text{ДКНФ}} = (x_2 \vee \bar{x}_1)(\bar{x}_2 \vee x_1)$.

Спосіб 2 (через подання функції у числовому вигляді):

$\delta_{\text{ДДНФ}} = \bar{x}_2 \bar{x}_1 \vee x_2 x_1 = 0 \vee 3$. Оскільки функція – повністю визначена, і дорівнює одиниці на наборах 0, 3, то на решті наборів – 1, 2, вона дорівнює нулю. Тоді $\delta_{\text{ДКНФ}} = \bar{1} \cdot \bar{2} = (x_2 \vee \bar{x}_1)(\bar{x}_2 \vee x_1)$. ■

Розглянемо отримання ДДНФ з ДКНФ.

Задача 2.5. Повністю визначену перемикальну функцію від трьох змінних $\gamma(x_3, x_2, x_1)$ подано в ДКНФ:

$$\gamma_{\text{ДКНФ}} = (x_3 \vee \bar{x}_2 \vee x_1)(\bar{x}_3 \vee x_2 \vee \bar{x}_1)(\bar{x}_3 \vee \bar{x}_2 \vee x_1).$$

Знайти її ДДНФ.

Розв'язування.

Оскільки функція повністю визначена, і $\gamma_{\text{ДКНФ}} = \bar{2} \cdot \bar{5} \cdot \bar{6}$, тобто дорівнює нулю на наборах 2, 5, 6, то на решті наборів – 0, 1, 3, 4, 7, вона дорівнює одиниці. Тоді

$$\gamma_{\text{ДДНФ}} = 0 \vee 1 \vee 3 \vee 4 \vee 7 = \bar{x}_3 \bar{x}_2 \bar{x}_1 \vee \bar{x}_3 \bar{x}_2 x_1 \vee \bar{x}_3 x_2 x_1 \vee x_3 \bar{x}_2 \bar{x}_1 \vee x_3 x_2 x_1. \blacksquare$$

Якщо перемикальна функція подана в ДНФ, тобто містить кон'юнктивні терми, які не є мінтермами, то для отримання ДДНФ функції необхідно виконати процедуру розгортання – домножити на $(x_i \vee \bar{x}_i)$, тобто на одиницю, ті терми, у яких відсутні відповідні змінні функції.

Задача 2.6. Дано перемикальну функцію $y(x_3, x_2, x_1)$ від трьох змінних: $y = x_3 x_1 \vee \bar{x}_3 x_2$.

Отримати ДДНФ функції.

Розв'язування.

У першому термі відсутня змінна x_2 , тому домножимо його на $x_2 \vee \bar{x}_2$, у другому термі відсутня змінна x_1 – домножимо його на $x_1 \vee \bar{x}_1$:

$$\begin{aligned} y &= x_3 x_1 \vee \bar{x}_3 x_2 = x_3 (x_2 \vee \bar{x}_2) x_1 \vee \bar{x}_3 x_2 (x_1 \vee \bar{x}_1) = \\ &= x_3 x_2 x_1 \vee x_3 \bar{x}_2 x_1 \vee \bar{x}_3 x_2 x_1 \vee \bar{x}_3 x_2 \bar{x}_1. \end{aligned}$$

Отриманий вираз є ДДНФ функції:

$$y_{\text{ДДНФ}} = \bar{x}_3 x_2 \bar{x}_1 \vee \bar{x}_3 x_2 x_1 \vee x_3 \bar{x}_2 x_1 \vee x_3 x_2 x_1 = 2 \vee 3 \vee 5 \vee 7. \blacksquare$$

Якщо перемикальна функція подана в КНФ, тобто містить диз'юнктивні терми, які не є макстермами, то для отримання ДКНФ функції необхідно виконати процедуру розгортання – додати (логічне додавання) вираз $x_j \bar{x}_j$, тобто нуль, до тих термів, у яких відсутні відповідні змінні.

Задача 2.7. Дано перемикальну функцію $y(x_3, x_2, x_1)$ від трьох змінних: $y = (\bar{x}_3 \vee x_1)(x_3 \vee x_2)$.

Отримати ДКНФ функції.

Розв'язування.

У першому термі відсутня змінна x_2 , тому додамо до нього $x_2 \bar{x}_2$ (вираз $x_2 \bar{x}_2$ дорівнює нулю), у другому термі відсутня змінна x_1 – додамо до нього $x_1 \bar{x}_1$:

$$y = (\bar{x}_3 \vee x_1)(x_3 \vee x_2) = (\bar{x}_3 \vee x_2 \bar{x}_2 \vee x_1)(x_3 \vee x_2 \vee x_1 \bar{x}_1).$$

Далі, застосувавши властивість дистрибутивності $a \vee bc = (a \vee b)(a \vee c)$ (див. рис. 2.7), отримуємо:

$$\begin{aligned} y &= ((\bar{x}_3 \vee x_1) \vee x_2 \bar{x}_2)((x_3 \vee x_2) \vee x_1 \bar{x}_1) = \\ &= (\bar{x}_3 \vee x_1 \vee x_2)(\bar{x}_3 \vee x_1 \vee \bar{x}_2)(x_3 \vee x_2 \vee x_1)(x_3 \vee x_2 \vee \bar{x}_1). \end{aligned}$$

Отриманий вираз є ДКНФ функції:

$$y_{\text{ДКНФ}} = (x_3 \vee x_2 \vee x_1)(x_3 \vee x_2 \vee \bar{x}_1)(\bar{x}_3 \vee x_2 \vee x_1)(\bar{x}_3 \vee \bar{x}_2 \vee x_1) = \bar{0} \cdot \bar{1} \cdot \bar{4} \cdot \bar{6}. \blacksquare$$

На основі вищерозглянутого можна зробити наступні висновки.

1. Конституенти одиниці (мінтерми) та конституенти нуля (макстерми) використовуються для переходу від табличного подання перемикальної функції до аналітичного: диз'юнкція конституент одиниці тих наборів, на яких функція дорівнює одиниці, дає ДДНФ перемикальної функції (формулу); кон'юнкція конституент нуля тих наборів, на яких функція дорівнює нулю, дає ДКНФ перемикальної функції (формулу).
2. Будь-яка перемикальна функція має єдину ДДНФ і єдину ДКНФ.
3. ДДНФ і ДКНФ перемикальної функції є еквівалентними.
4. У ДДНФ і ДКНФ використовуються три логічні операції – заперечення, кон'юнкція і диз'юнкція.
5. Перехід від аналітичного до табличного подання перемикальної функції виконують послідовною підстановкою в алгебраїчний вираз усіх можливих наборів значень змінних, визначенням значення функції на кожному наборі та заповненням відповідних рядків таблиці істинності. Інший спосіб – заповнити таблицю істинності функції на основі її ДДНФ або ДКНФ.

Слід зазначити, що ДДНФ та ДКНФ перемикальних функцій у загальному випадку не є найпростішими аналітичними виразами. Використовуючи аксіоми та закони булевої алгебри, а також інші прийоми зазвичай з ДДНФ або ДКНФ отримують простіші форми (формули), які називають мінімізованими.

Запитання та завдання

1. Як задати дискретну алгебру?
2. На яких логічних операціях ґрунтується алгебра Буля?
3. Сформулюйте та запишіть в аналітичному вигляді аксіоми булевої алгебри.
4. Сформулюйте та запишіть аналітично основні закони (властивості) булевої алгебри.
5. Як доводять тотожності (співвідношення) в булевій алгебрі?
6. Назвіть канонічні форми подання перемикальних функцій в алгебрі Буля.
7. Дайте визначення таких термінів: терм, макстерм, диз'юнктивний терм, кон'юнктивний терм. Наведіть приклади зазначених термів.
8. Дайте визначення таким формам перемикальних функцій: диз'юнктивна форма, кон'юнктивна форма, нормальна диз'юнктивна форма, нормальна кон'юнктивна форма, досконала диз'юнктивна нормальна форма, досконала кон'юнктивна нормальна форма. Наведіть приклади вказаних форм.
9. Запишіть аналітично всі можливі конституенти одиниці та конституенти нуля у випадку 2-х змінних.
10. Як пов'язані між собою конституента нуля та конституента одиниці для заданого набору значень аргументів?
11. Як знайти ДКНФ повністю визначеної функції, якщо відома її ДДНФ?
12. Як знайти ДДНФ повністю визначеної функції, якщо відома її ДКНФ?

Задачі для самостійного розв'язування

1. Використовуючи аксіоми та закони алгебри Буля спростити вирази:
 - а) $y = \overline{x_3} \overline{x_2} \overline{x_1} \vee \overline{x_3} x_2 \overline{x_1} \vee \overline{x_3} x_2 x_1 \vee x_3 x_2 \overline{x_1} \vee x_3 x_2 x_1$,
 - б) $y = x_4 \overline{x_3} x_1 \vee \overline{x_4} x_3 x_2 \vee \overline{x_3} x_1 \vee \overline{x_3}$,
 - в) $y = (\overline{x_3} x_1 \vee \overline{x_2} \vee x_4 x_3)(\overline{x_3} x_1 \vee x_2)$.
2. Дано два набори (кортежі): $\langle 101101 \rangle$ та $\langle 10110 \rangle$. Для кожного з них записати відповідну конституенту нуля та конституенту одиниці.

3. Визначити набори (кортежі), які відповідають наступним конституентам: $\bar{x}_5 \vee x_4 \vee x_3 \vee \bar{x}_2 \vee x_1$, $x_4 \vee \bar{x}_3 \vee \bar{x}_2 \vee x_1$, $x_6 \bar{x}_5 x_4 \bar{x}_3 \bar{x}_2 x_1$, $\bar{x}_4 \bar{x}_3 x_2 x_1$.
4. Записати всі можливі конституенти нуля (макстерми) та конституенти одиниці (мінтерми) у випадку 4-х змінних.
5. Повністю визначена перемикальна функція від трьох змінних дорівнює одиниці на наборах 1, 2, 5, 6 і нулю – на решті наборів. Подати функцію в ДДНФ та ДКНФ.
6. ДДНФ перемикальної функції від трьох змінних має вигляд $f(x_3, x_2, x_1) = 1 \vee 2 \vee 5 \vee 6$. Побудувати комбінаційну схему в елементному базисі алгебри Буля, що реалізує функцію f , та визначити її складність за Квайном.
7. ДКНФ перемикальної функції від чотирьох змінних має вигляд $\gamma(x_4, x_3, x_2, x_1) = \bar{1} \cdot \bar{3} \cdot \bar{9} \cdot \bar{13}$. Побудувати комбінаційну схему в елементному базисі алгебри Буля, що реалізує функцію γ , на двовходових логічних елементах та визначити її складність за Квайном.
8. Повністю визначена перемикальна функція від трьох змінних задана в диз'юнктивній формі: $\delta(x_3, x_2, x_1) = x_3 x_2 \vee \bar{x}_2$. Подати функцію δ в ДДНФ та ДКНФ.
9. Повністю визначена перемикальна функція від трьох змінних задана в кон'юнктивній формі: $\omega = (\bar{x}_3 \vee x_1)(x_3 \vee \bar{x}_2)(x_2 \vee \bar{x}_1)$. Подати функцію ω в ДДНФ та ДКНФ.

2.2. Алгебра Шеффера

Для перетворення виразів в алгебрі Шеффера використовують лише одну логічну операцію – операцію І-НЕ (штрих Шеффера, функція Шеффера).

Операція І-НЕ (NAND) є n -місною, $n \geq 2$:

$$f = \overline{x_n \cdot x_{n-1} \cdot \dots \cdot x_1} = x_n / x_{n-1} / \dots / x_1.$$

Аксіоми алгебри Шеффера:

$$\begin{array}{ll} \overline{x \cdot 0} = x / 0 = 1, & \overline{\bar{x} \cdot \bar{x}} = x / x = \bar{x}, \\ \overline{x \cdot 1} = x / 1 = \bar{x}, & \overline{x \cdot \bar{x}} = x / \bar{x} = 1. \end{array}$$

В алгебрі Шеффера виконується лише закон (властивість) комутативності (переставний закон):

$$\overline{x_2 x_1} = \overline{x_1 x_2}, \quad x_2 / x_1 = x_1 / x_2.$$

Закон асоціативності (сполучний закон) не виконується, тобто

$$\overline{x_3 x_2 x_1} \neq \overline{(x_3 x_2) x_1}, \quad x_3 / x_2 / x_1 \neq (x_3 / x_2) / x_1.$$

Не виконується також закон дистрибутивності (розподільний закон).

В алгебрі Шеффера будь-яку перемикальну функцію можна подати в канонічній формі, що має назву І-НЕ/І-НЕ (NAND/ NAND).

Форма І-НЕ/І-НЕ є нормальною (дворівневою) – як внутрішньою, так і зовнішньою є функція І-НЕ.

Канонічна форма І-НЕ/І-НЕ перемикальної функції від n змінних, складається з термів Шеффера n -го рангу, об'єднаних операцією І-НЕ.

Терм Шеффера n -го рангу має такий узагальнений вигляд:

$$\overline{x_n^{\sigma_n} \cdot x_{n-1}^{\sigma_{n-1}} \cdot \dots \cdot x_1^{\sigma_1}} \quad \text{або} \quad x_n^{\sigma_n} / x_{n-1}^{\sigma_{n-1}} / \dots / x_1^{\sigma_1},$$

$$\text{де } x_i^{\sigma_i} = \begin{cases} x_i, & \text{якщо } \sigma_i = 1 \\ \overline{x_i}, & \text{якщо } \sigma_i = 0, \quad i = 2, 3, \dots, n, \end{cases}$$

$\langle \sigma_n \sigma_{n-1} \dots \sigma_1 \rangle$ – двійковий набір.

Будь-якому двійковому набору можна поставити у відповідність терм Шеффера, і навпаки – терму Шеффера відповідає двійковий набір (кортеж).

Наприклад, набору $\langle 0110 \rangle$ відповідає терм Шеффера $\overline{x_4 x_3 x_2 x_1}$, а терму Шеффера $\overline{x_3 x_2 x_1}$ відповідає набір $\langle 101 \rangle$.

Якщо перемикальну функцію задано таблицею істинності, то для утворення канонічної форми перемикальної функції в алгебрі Шеффера (форми І-НЕ/І-НЕ) необхідно виписати терми Шеффера для тих наборів, на яких функція набуває значення 1, та об'єднати отримані терми операцією І-НЕ.

Нехай перемикальна функція y від трьох змінних задана таблицею істинності (табл. 2.5).

Таблиця 2.5
Таблиця істинності перемикальної функції y

Номер набору	x_3	x_2	x_1	y
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Канонічна форма функції y в алгебрі Шеффера (кфШ) має вигляд:

$$y_{\text{кфШ}} = \overline{\overline{\overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1} \cdot x_3 x_2 x_1}} \quad (\text{I-HE/I-HE}),$$

їй відповідає комбінаційна схема на рис. 2.9.

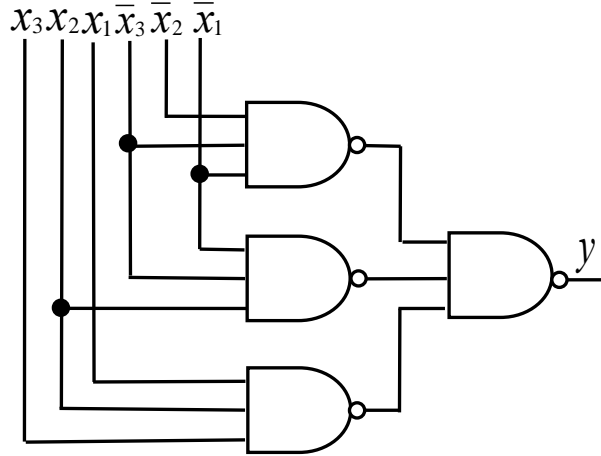


Рис. 2.9. Апаратна реалізація функції $y = 0 \vee 2 \vee 7$ (задана табл. 2.5) в елементному базисі алгебри Шеффера

Інший приклад. Нехай функція $\gamma(x_2, x_1)$ від двох змінних набуває значення 1 лише на одному наборі аргументів – $\langle 1, 0 \rangle$, а на решті наборів – 0. Тоді канонічна форма функції γ в алгебрі Шеффера має вигляд $\gamma_{\text{кфШ}} = \overline{x_2 \bar{x}_1}$, і їй відповідає комбінаційна схема на рис. 2.10.

x_2	x_1	γ
0	0	0
0	1	0
1	0	1
1	1	0

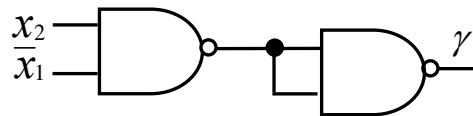


Рис. 2.10. Апаратна реалізація функції $\gamma(x_2, x_1)$ в елементному базисі алгебри Шеффера

Канонічну форму перемикальної функції в алгебрі Шеффера (форму I-HE/I-HE) можна одержати також з ДДНФ цієї функції, застосувавши до неї аксіому подвійного заперечення та правило де Моргана.

Задача 2.8. Дано ДДНФ функції $\delta(x_3, x_2, x_1)$:

$$\delta_{\text{ДДНФ}} = \bar{x}_3 \bar{x}_2 x_1 \vee \bar{x}_3 x_2 \bar{x}_1 \vee x_3 \bar{x}_2 x_1 \vee x_3 x_2 \bar{x}_1.$$

Отримати канонічну форму функції δ в алгебрі Шеффера.

Розв'язування.

Поставимо подвійне заперечення над правою частиною виразу та розкриємо нижнє заперечення за правилом де Моргана:

$$\begin{aligned} \delta &= \overline{\overline{x_3 x_2 x_1} \vee \overline{x_3 x_2 x_1} \vee \overline{x_3 x_2 x_1} \vee \overline{x_3 x_2 x_1}} = \\ &= \overline{\overline{\overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1}}} = \delta_{\text{кфш}}. \blacksquare \end{aligned}$$

Канонічна форма перемикальної функції в алгебрі Шеффера є нормальною (дворівневою), відповідно, вона забезпечує побудову двокаскадної комбінаційної схеми, якщо немає обмеження на кількість входів логічних елементів І-НЕ.

Якщо ж ранг термів Шеффера у канонічній формі перевищує кількість входів логічних елементів І-НЕ, то терми слід перетворити так, щоб враховувалась кількість входів логічних елементів, тобто перетворити в операторну форму, якій відповідає суперпозиція логічних елементів І-НЕ.

Нехай терм $\overline{x_4 \bar{x}_3 \bar{x}_2 x_1}$ необхідно реалізувати на логічних елементах 2І-НЕ. Тоді його слід перетворити так: $\overline{x_4 \bar{x}_3 \bar{x}_2 x_1} = \overline{\overline{\overline{\overline{x_4 \bar{x}_3 \bar{x}_2 x_1}}}}$.

Відповідна комбінаційна схема (рис. 2.11) є трирівневою (трикаскадною), вона заміщує 4-входовий елемент І-НЕ.

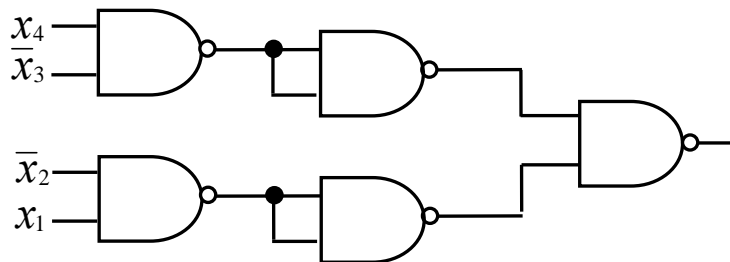


Рис. 2.11. Схема, що реалізує терм Шеффера $\overline{x_4 \bar{x}_3 \bar{x}_2 x_1}$ на елементах 2І-НЕ

Деякі приклади перетворення термів Шеффера при $n = 3, 4, 5$ наведено на рис. 2.12.

Задача 2.9. Нехай повністю визначена перемикальна функція $\beta(x_3, x_2, x_1)$ від трьох змінних задана в числовому вигляді: $\beta_{\text{дкнф}} = \bar{0} \cdot \bar{2} \cdot \bar{3} \cdot \bar{6} \cdot \bar{7}$. Реалізувати її на 2-входових елементах І-НЕ.

Розв'язування.

Оскільки β – повністю визначена функція, то її ДДНФ має вигляд: $\beta_{\text{дднф}} = 1 \vee 4 \vee 5$.

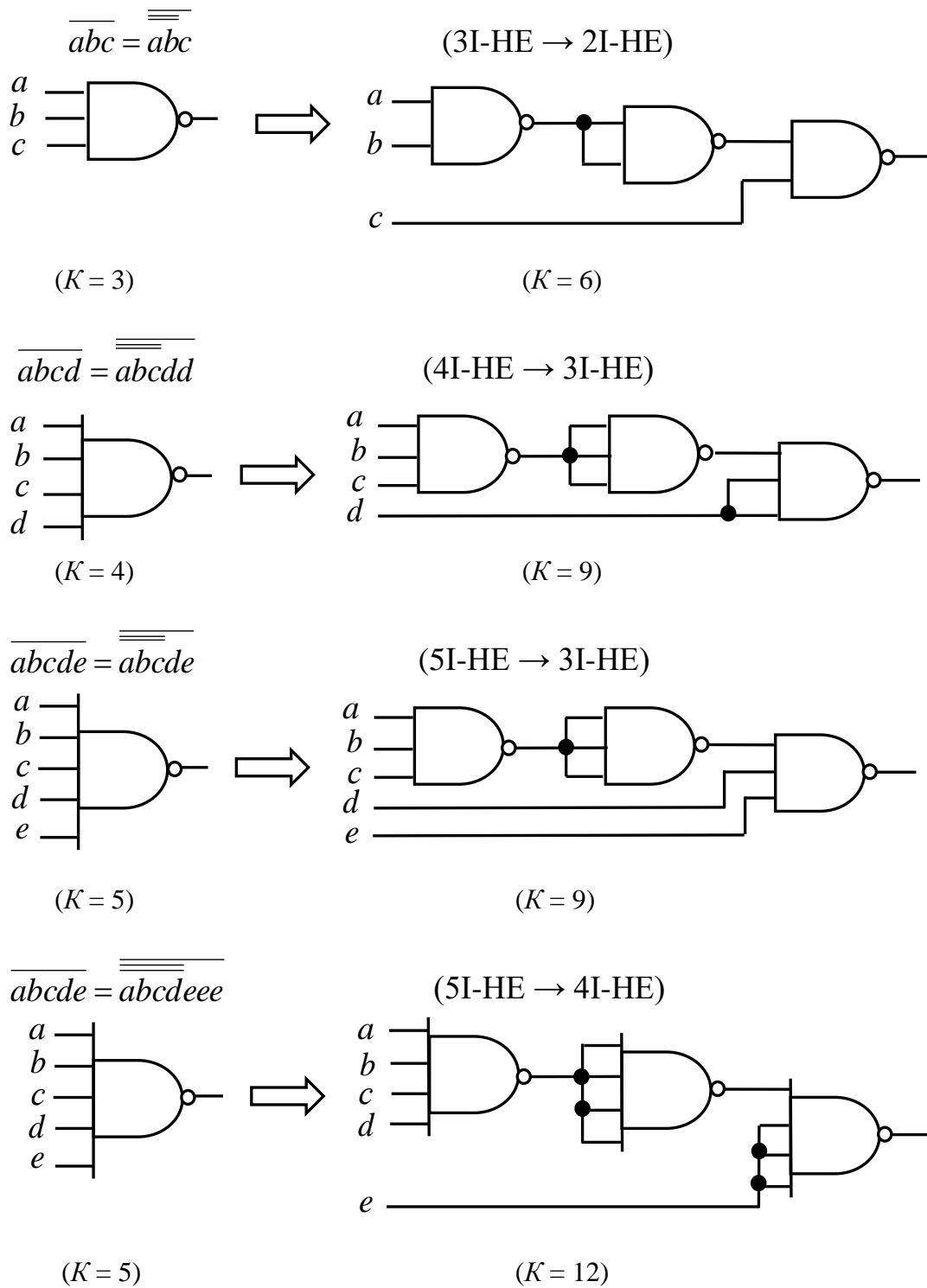


Рис. 2.12. Приклади перетворення термів Шеффера та їх апаратна реалізація

Подамо функцію β в канонічній нормальній формі алгебри Шеффера:

$$\beta_{\text{кфш}} = \overline{\overline{\overline{\overline{\overline{x_3 \overline{x_2} x_1} \cdot x_3 \overline{x_2} \overline{x_1}} \cdot x_3 \overline{x_2} x_1}}}} \quad (\text{I-НЕ/I-НЕ})$$

Запишемо функцію β в операторній формі:

$$\beta_{\text{оп}} = \overline{\overline{\overline{\overline{\overline{\overline{\overline{\overline{\overline{x_3 \overline{x_2} x_1} \cdot x_3 \overline{x_2} \overline{x_1}} \cdot x_3 \overline{x_2} x_1}}}}}}}}}}$$

На основі операторної форми функції будуємо відповідну комбінаційну схему (рис. 2.13). ■

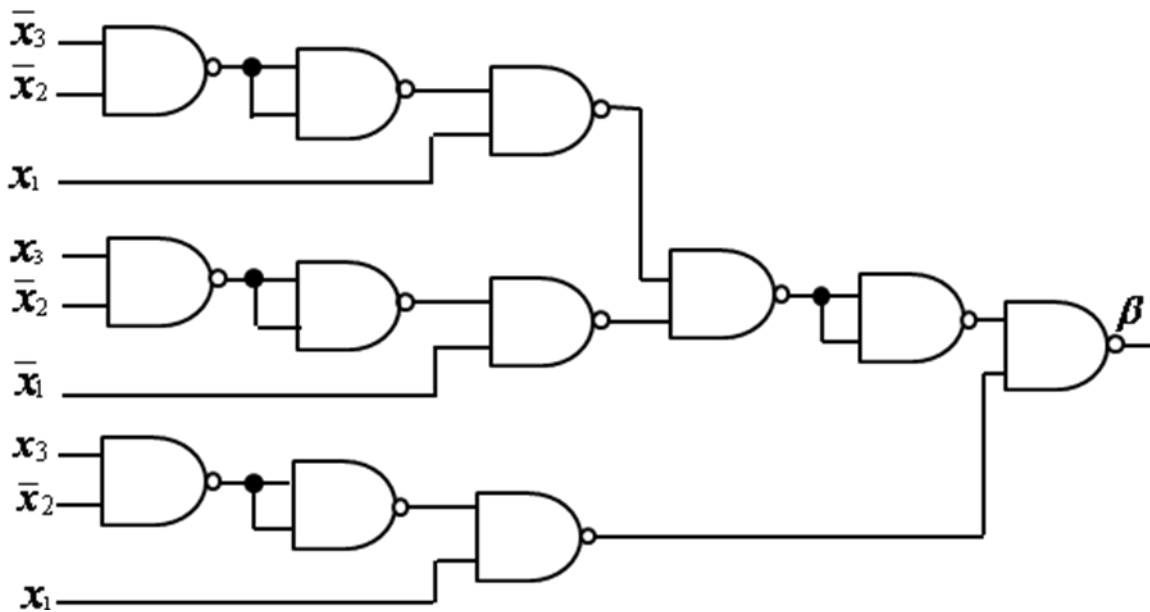
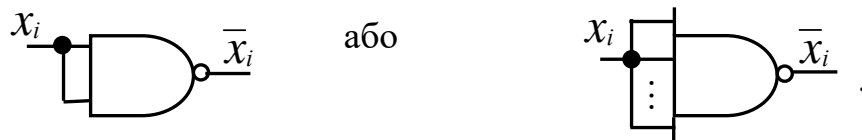


Рис. 2.13. Апаратна реалізація функції β на логічних елементах 2I-НЕ

Інверсія змінної в алгебрі Шеффера ґрунтується на аксіомі $\overline{x_i \cdot x_i} = \overline{x_i}$ або її узагальненні: $\overline{x_i \cdot x_i \cdot \dots \cdot x_i} = \overline{x_i}$. Відповідно, апаратна реалізація інверсії змінної в елементному базисі алгебри Шеффера має вигляд:



У деяких випадках вимагається, щоб на входи комбінаційної схеми подавалися лише прямі значення змінних. Тоді при побудові схеми в елементному базисі алгебри Шеффера отриманий вираз перетворюють так, щоб уникнути інверсій змінних.

Нехай перемикальна функція y подана в канонічній формі алгебри Шеффера: $y = \overline{\overline{\overline{x_3 x_2 x_1}} \cdot \overline{\overline{x_3 x_2 x_1}} \cdot x_3 x_2 x_1}$, а на входи комбінаційної схеми дозволяється подавати лише прямі значення змінних, тобто x_3, x_2, x_1 .

Для уникнення інверсій змінних на входах схеми змінні з інверсіями слід подати у вигляді: $\overline{x_i} = \overline{x_i \cdot x_i}$.

Тоді y набуває вигляду:

$$y = \overline{\overline{\overline{x_3 x_3 x_2 x_2 x_1 x_1}} \cdot \overline{\overline{\overline{x_3 x_3 x_2 x_1 x_1}} \cdot \overline{\overline{\overline{x_3 x_2 x_1}}}}$$

Відповідна комбінаційна схема буде вже не 2-рівневою (2-каскадною), а 3-рівневою. І це при тому, що обмеження на кількість входів логічних елементів не накладається.

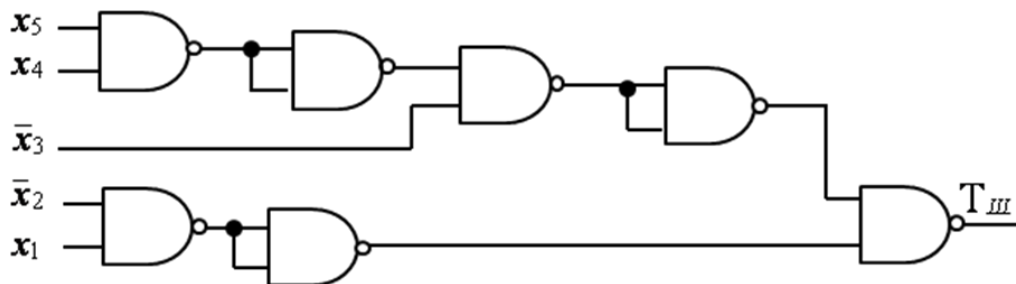
Заяитання та завдання

1. На застосуванні якої логічної операції ґрунтується алгебра Шеффера?
2. Сформулюйте аксіоми алгебри Шеффера.
3. Якою є канонічна форма перемикальної функції в алгебрі Шеффера?
4. Дайте визначення терму Шеффера n -го рангу.
5. Дано набори $\langle 1010 \rangle$ та $\langle 110 \rangle$. Утворіть відповідні терми Шеффера.
6. Дано терми Шеффера: $\overline{\overline{\overline{x_5 x_4 x_3 x_2 x_1}}}$, $\overline{\overline{\overline{x_4 x_3 x_2 x_1}}}$. Яким наборам (кортежам) вони відповідають?
7. Як утворити канонічну форму в алгебрі Шеффера перемикальної функції, заданої таблицею істинності?
8. Як отримати канонічну форму перемикальної функції в алгебрі Шеффера, якщо відома її ДДНФ?
9. Дано терми Шеффера: $\overline{\overline{\overline{x_5 x_4 x_3 x_2 x_1}}}$, $\overline{\overline{\overline{x_4 x_3 x_2 x_1}}}$. Реалізуйте їх на логічних елементах ЗІ-НЕ.
10. Як реалізувати інвертор в елементному базисі алгебри Шеффера?

Задачі для самостійного розв'язування

1. Перемикальна функція від чотирьох змінних дорівнює одиниці на наборах 1, 3, 6, 14, а на решті наборів – нулю. Отримати канонічну форму функції в алгебрі Шеффера.

2. Повністю визначена перемикальна функція від чотирьох змінних дорівнює нулю на наборах 0, 1, 3, 4, 5, 10, 13, 15, а на решті наборів – одиниці. Отримати канонічну форму функції в алгебрі Шеффера.
3. Дано функцію від трьох змінних $y_{\text{ДНФ}} = 0 \vee 1 \vee 4 \vee 7$. Побудувати комбінаційну схему, що реалізує функцію y , на 2-входових логічних елементах І-НЕ.
4. Дано повністю визначену функцію від трьох змінних $f_{\text{ДКНФ}} = \bar{1} \cdot \bar{2} \cdot \bar{3} \cdot \bar{5} \cdot \bar{6}$. Побудувати комбінаційну схему на елементах 2І-НЕ.
5. Канонічна форма перемикальної функції $y(x_3, x_2, x_1)$ в алгебрі Шеффера має вигляд $y_{\text{КФШ}} = \overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1}$. Реалізувати її на елементах 2І-НЕ за умови, що на входи комбінаційної схеми дозволяється подавати лише прямі значення змінних.
6. Дано комбінаційну схему, яка реалізує терм Шеффера $T_{\text{Ш}}$:



Записати терм $T_{\text{Ш}}$ аналітично.

7. Подати повністю визначену перемикальну функцію y від трьох змінних в канонічній формі алгебри Шеффера:
 - a) $y = (a \vee b)b(a \vee c)$,
 - b) $y = \bar{a}c \vee b\bar{c} \vee a\bar{b}\bar{c}$.

2.3. Алгебра Пірса

Алгебра Пірса ґрунтується на застосуванні лише однієї логічної операції – АБО-НЕ (стрілка Пірса, функція Пірса). Операція АБО-НЕ (NOR) є n -місною, $n \geq 2$:

$$f = \overline{x_n \vee x_{n-1} \vee \dots \vee x_1} = x_n \downarrow x_{n-1} \downarrow \dots \downarrow x_1.$$

Аксиоми алгебри Пірса:

$$\begin{aligned} \overline{x \vee 0} &= x \downarrow 0 = \bar{x}, & \overline{x \vee x} &= x \downarrow x = \bar{x}, \\ \overline{x \vee 1} &= x \downarrow 1 = 0, & \overline{x \vee \bar{x}} &= x \downarrow \bar{x} = 0. \end{aligned}$$

В алгебрі Пірса виконується лише закон (властивість) комутативності (переставний закон):

$$\overline{x_2 \vee x_1} = \overline{x_1 \vee x_2}, \quad x_2 \downarrow x_1 = x_1 \downarrow x_2,$$

Закон асоціативності (сполучний закон) не виконується, тобто

$$\overline{x_3 \vee x_2 \vee x_1} \neq \overline{(x_3 \vee x_2) \vee x_1}, \quad x_3 \downarrow x_2 \downarrow x_1 \neq (x_3 \downarrow x_2) \downarrow x_1.$$

Не виконується також й закон дистрибутивності (розподільний закон).

Будь-яку перемикальну функцію в алгебрі Пірса можна подати в канонічній формі, що має назву АБО-НЕ/АБО-НЕ (NOR/NOR). Форма АБО-НЕ/АБО-НЕ є дворівневою (нормальною) – як внутрішньою, так і зовнішньою є функція АБО-НЕ.

Канонічна форма АБО-НЕ/АБО-НЕ перемикальної функції від n змінних складається з термів Пірса n -го рангу, об'єднаних операцією АБО-НЕ.

Терм Пірса n -го рангу має такий узагальнений вигляд:

$$\overline{\overline{x_n^{\sigma_n} \vee x_{n-1}^{\sigma_{n-1}} \vee \dots \vee x_1^{\sigma_1}}} \text{ або } x_n^{\sigma_n} \downarrow x_{n-1}^{\sigma_{n-1}} \downarrow \dots \downarrow x_1^{\sigma_1},$$

$$\text{де } x^{\sigma_i} = \begin{cases} \bar{x}_i, & \text{якщо } \sigma_i = 1, \\ x_i, & \text{якщо } \sigma_i = 0, i = 2, 3, \dots, n, \end{cases}$$

$\langle \sigma_n \sigma_{n-1} \dots \sigma_1 \rangle$ – двійковий набір.

Будь-якому двійковому набору можна поставити у відповідність терм Пірса, і навпаки – терму Пірса відповідає двійковий набір (кортеж).

Наприклад, набору $\langle 0101 \rangle$ відповідає терм Пірса $\overline{x_4 \vee \bar{x}_3 \vee x_2 \vee \bar{x}_1}$, а терму Пірса $\overline{x_3 \vee \bar{x}_2 \vee \bar{x}_1}$ відповідає набір $\langle 011 \rangle$.

Якщо перемикальну функцію задано таблицею істинності, то для утворення канонічної форми функції в алгебрі Пірса (форма АБО-НЕ/АБО-НЕ) необхідно виписати терми Пірса для тих наборів, на яких функція набуває значення 0, та об'єднати отримані терми операцією АБО-НЕ.

Нехай перемикальна функція f від трьох змінних задана таблицею істинності (табл. 2.6).

Канонічна форма функції f в алгебрі Пірса (кфП) має вигляд:

$$f_{\text{кфП}} = \overline{\overline{x_3 \vee x_2 \vee \bar{x}_1} \vee \overline{\bar{x}_3 \vee x_2 \vee x_1} \vee \overline{\bar{x}_3 \vee x_2 \vee \bar{x}_1} \vee \overline{\bar{x}_3 \vee \bar{x}_2 \vee x_1}},$$

їй відповідає комбінаційна схема на рис. 2.14.

Таблиця 2.6
Таблиця істинності
перемикальної функції f

Номер набору	x_3	x_2	x_1	f
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

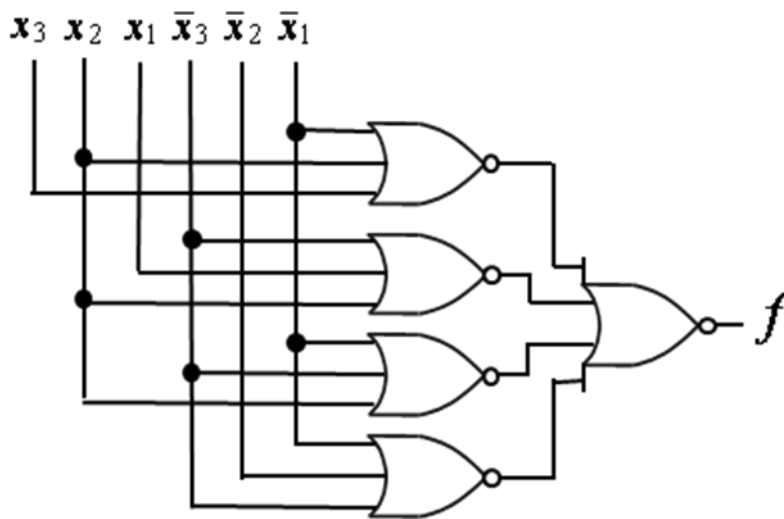


Рис. 2.14. Апаратна реалізація функції f (задана табл. 2.6) в елементному базисі алгебри Пірса

Інший приклад. Нехай функція $\delta(x_2, x_1)$ від двох змінних набуває значення 0 лише на одному наборі аргументів – $\langle 0 \ 1 \rangle$, а на решті наборів – 1. Тоді канонічна форма функції δ в алгебрі Пірса має вигляд $\delta_{кфП} = \overline{x_2 \vee x_1}$, і їй відповідає комбінаційна схема на рис. 2.15.

x_2	x_1	δ
0	0	1
0	1	0
1	0	1
1	1	1

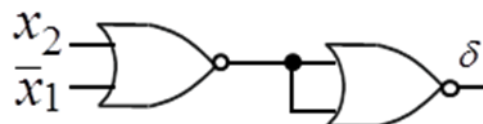


Рис. 2.15. Апаратна реалізація функції $\delta(x_2, x_1)$ в елементному базисі алгебри Пірса

Канонічну форму перемикальної функції в алгебрі Пірса (форму АБО-НЕ/АБО-НЕ) можна одержати також з ДКНФ цієї функції застосувавши до неї аксіому подвійного заперечення та правило де Моргана.

Задача 2.10. Дано ДКНФ функції $\beta(x_3, x_2, x_1)$:

$$\beta_{\text{ДКНФ}} = (x_3 \vee x_2 \vee x_1)(x_3 \vee \bar{x}_2 \vee \bar{x}_1)(\bar{x}_3 \vee \bar{x}_2 \vee x_1).$$

Отримати канонічну форму функції β в алгебрі Пірса.

Розв'язування.

Поставимо подвійне заперечення над правою частиною виразу та розкриємо нижнє заперечення за правилом де Моргана:

$$\begin{aligned} \beta &= \overline{\overline{(x_3 \vee x_2 \vee x_1)(x_3 \vee \bar{x}_2 \vee \bar{x}_1)(\bar{x}_3 \vee \bar{x}_2 \vee x_1)}} = \\ &= \overline{\overline{x_3 \vee x_2 \vee x_1} \vee \overline{\overline{x_3 \vee \bar{x}_2 \vee \bar{x}_1}} \vee \overline{\overline{\bar{x}_3 \vee \bar{x}_2 \vee x_1}}} = \beta_{\text{кфП}}. \blacksquare \end{aligned}$$

Канонічна форма перемикальної функції в алгебрі Пірса є нормальною (дворівневою), тому вона забезпечує побудову двокаскадної комбінаційної схеми, якщо немає обмеження на кількість входів логічних елементів АБО-НЕ.

Якщо ж ранг термів Пірса у канонічній формі перевищує кількість входів логічних елементів АБО-НЕ, то терми слід перетворити в операторну форму так, щоб враховувалась кількість входів логічних елементів.

Нехай терм $\overline{\overline{x_4 \vee x_3 \vee \bar{x}_2 \vee x_1}}$ необхідно реалізувати на логічних елементах 2АБО-НЕ. Тоді його слід перетворити так:

$$\overline{\overline{x_4 \vee x_3 \vee \bar{x}_2 \vee x_1}} = \overline{\overline{\overline{\overline{x_4 \vee x_3 \vee \bar{x}_2 \vee x_1}}}}$$

Відповідна комбінаційна схема (рис. 2.16) є трирівневою (трикаскадною), вона заміщує логічний елемент 4АБО-НЕ.

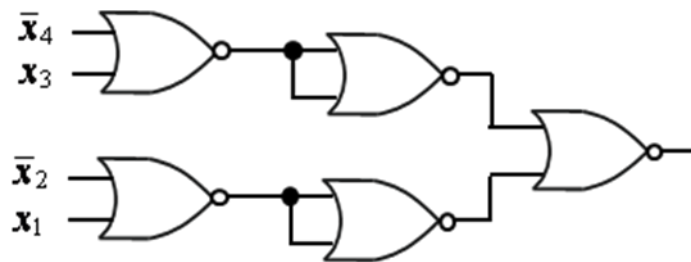


Рис. 2.16. Схема, що реалізує терм Пірса $\overline{\overline{x_4 \vee x_3 \vee \bar{x}_2 \vee x_1}}$ на елементах 2АБО-НЕ

Деякі приклади перетворення термів Пірса при $n = 3, 4, 5$ наведено на рис. 2.17.

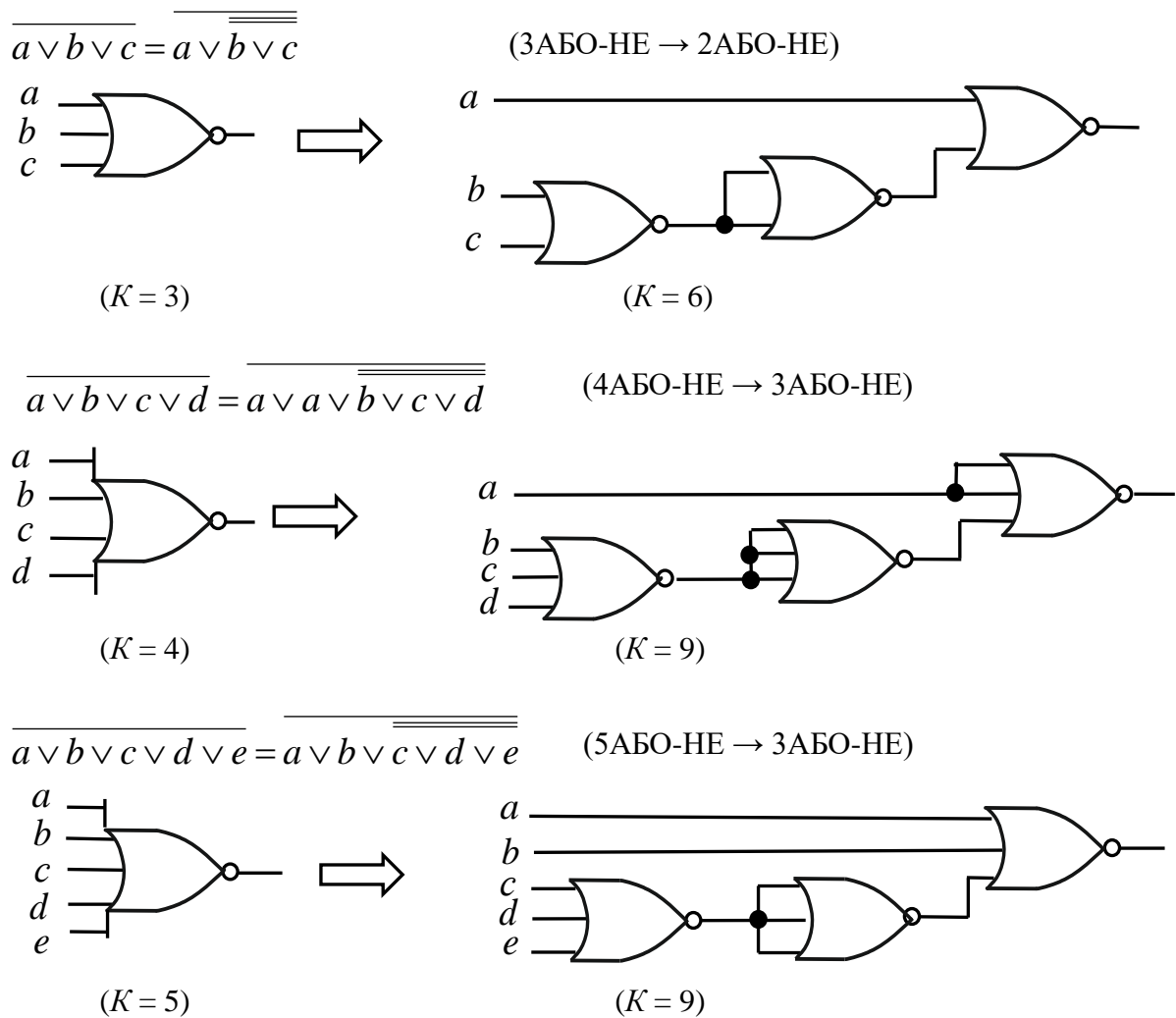


Рис. 2.17. Приклади перетворення термів Пірса та їх апаратна реалізація

Задача 2.11. Повністю визначена перемикальна функція $y(x_3, x_2, x_1)$ від трьох змінних задана в ДДНФ: $y_{\text{ДДНФ}} = 1 \vee 3 \vee 4 \vee 7$. Реалізувати її на елементах 2АБО-НЕ.

Розв'язування.

Оскільки y – повністю визначена функція, то її ДКНФ має вигляд:
 $y_{\text{ДКНФ}} = \overline{0} \cdot \overline{2} \cdot \overline{5} \cdot \overline{6}$.

Перетворимо її в канонічну форму в алгебрі Пірса:

$$\begin{aligned}
 y_{\text{ДКНФ}} &= \overline{0} \cdot \overline{2} \cdot \overline{5} \cdot \overline{6} = \overline{\overline{0} \cdot \overline{2} \cdot \overline{5} \cdot \overline{6}} = \\
 &= \overline{(x_3 \vee x_2 \vee x_1)(x_3 \vee \overline{x_2} \vee x_1)(\overline{x_3} \vee x_2 \vee \overline{x_1})(\overline{x_3} \vee \overline{x_2} \vee x_1)} = \\
 &= \overline{x_3 \vee x_2 \vee x_1 \vee x_3 \vee \overline{x_2} \vee x_1 \vee \overline{x_3} \vee x_2 \vee \overline{x_1} \vee \overline{x_3} \vee \overline{x_2} \vee x_1} = y_{\text{кфП}}.
 \end{aligned}$$

Перетворимо отриману канонічну форму АБО-НЕ/АБО-НЕ в операторну форму, придатну для реалізації на логічних елементах 2АБО-НЕ:

$$y = x_3 \vee x_2 \vee x_1 \vee x_3 \vee x_2 \vee x_1 \vee x_3 \vee x_2 \vee x_1 \vee x_3 \vee x_2 \vee x_1$$

Така форма і відповідна їй комбінаційна схема (рис. 2.18) є шестирівневою. ■

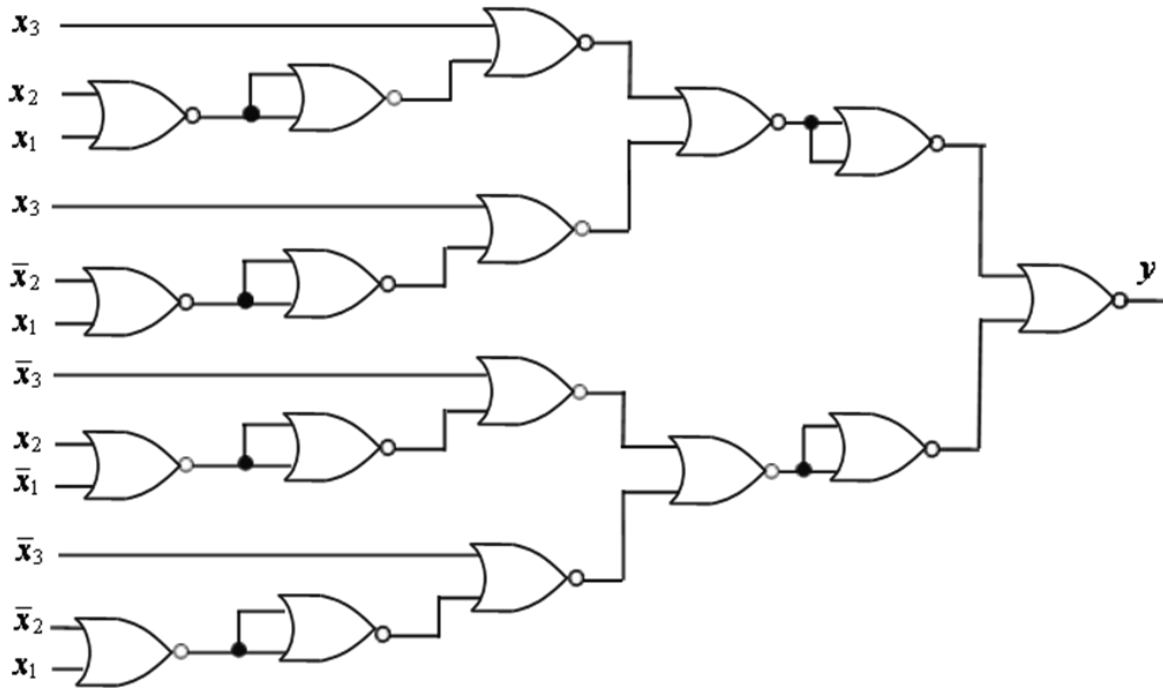


Рис. 2.18. Апаратна реалізація функції y на логічних елементах 2АБО-НЕ

Інверсія змінної (аргументу) в алгебрі Пірса ґрунтується на аксіомі $x_i \vee \bar{x}_i = \bar{x}_i$ або її узагальненні: $x_i \vee x_i \vee \dots \vee x_i = \bar{x}_i$. Відповідно, апаратна реалізація інверсії змінної має вигляд:



Якщо вимагається, щоб на входи комбінаційної схеми подавались лише прямі значення змінних, то вираз (функцію), що підлягає реалізації в елементному базисі алгебри Пірса, перетворюють так, щоб уникнути інверсій змінних.

Запитання та завдання

1. На застосуванні якої логічної операції ґрунтується алгебра Пірса?
2. Сформулюйте аксіоми алгебри Пірса?
3. Дайте визначення терма Пірса n -го рангу.
4. Якою є канонічна форма перемикальної функції в алгебрі Пірса?
5. Дано набори $\langle 010 \rangle$ та $\langle 0110 \rangle$. Утворіть відповідні терми Пірса.
6. Дано терми Пірса: $\overline{x_4 \vee x_3 \vee x_2 \vee x_1}$, $x_5 \vee \overline{x_4 \vee x_3 \vee x_2 \vee x_1}$. Яким наборам (кортежам) вони відповідають?
7. Перемикальна функція задана таблицею істинності. Як утворити канонічну форму функції в алгебрі Пірса?
8. Перемикальна функція задана в ДКНФ. Як подати її в канонічній формі алгебри Пірса?
9. Дано терм Пірса $\overline{x_5 \vee x_4 \vee \overline{x_3 \vee \overline{x_2 \vee x_1}}$. Реалізуйте його на логічних елементах 2АБО-НЕ.
10. Реалізуйте функції $f_1(x) = x$ та $f_2(x) = \bar{x}$ в елементному базисі алгебри Пірса.
11. Як реалізувати інвертор на основі логічного елемента 3АБО-НЕ?

Задачі для самостійного розв'язування

1. Перемикальна функція від чотирьох змінних дорівнює нулю на наборах 1, 2, 11, 15, а на решті наборів – одиниці. Отримати канонічну форму функції в алгебрі Пірса.
2. Відома ДДНФ повністю визначеної перемикальної функції від чотирьох змінних:

$$y_{\text{ДДНФ}} = 0 \vee 1 \vee 3 \vee 5 \vee 9 \vee 10 \vee 11 \vee 12 \vee 13 \vee 15.$$

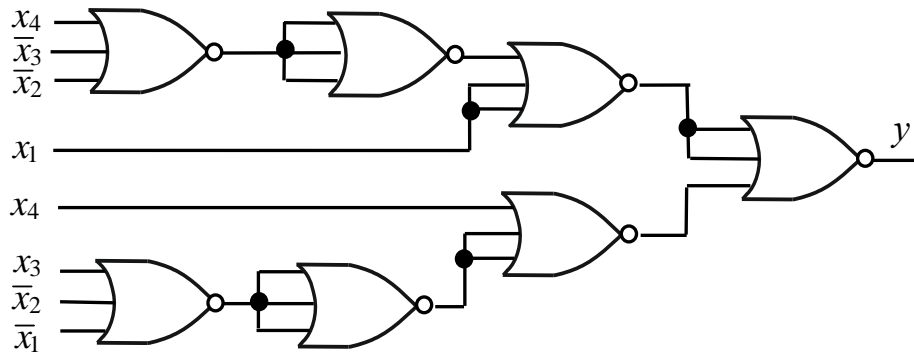
Подати функцію в канонічній формі алгебри Пірса.

3. Дано ДКНФ перемикальної функції від чотирьох змінних $f_{\text{ДКНФ}} = \bar{0} \cdot \bar{9} \cdot \bar{11} \cdot \bar{14}$. Побудувати комбінаційну схему на елементах 2АБО-НЕ та визначити її складність за Квайном.

4. Канонічна форма перемикальної функції $f(x_4, x_3, x_2, x_1)$ в алгебрі Пірса має вигляд: $f_{\text{кфП}} = \overline{x_4 \vee x_3 \vee x_2 \vee x_1 \vee \overline{x_4 \vee x_3 \vee x_2 \vee x_1}}$

Реалізувати її на елементах 3АБО-НЕ за умови, що на входи схеми мають надходити лише прямі значення змінних.

5. Дано комбінаційну схему, що реалізує перемикальну функцію y в елементному базисі алгебри Пірса:



Подати функцію y в канонічній формі алгебри Пірса.

6. Подати повністю визначену перемикальну функцію від трьох змінних в канонічній формі алгебри Пірса: а) $f = a\bar{b} \vee \bar{a}b\bar{c} \vee b\bar{c}$; б) $f = (\bar{b} \vee c)(a \vee \bar{c})(\bar{a} \vee \bar{b} \vee c)$; в) $f = a(b \vee c)(a \vee c)$.

2.4. Алгебра Жегалкіна

Алгебра Жегалкіна визначена на множині елементів $B = \{0, 1\}$ та ґрунтується на множині логічних операцій $\Omega = \{\&, \oplus, 1\}$ – кон'юнкція, додавання за модулем два та константа 1:

$$\begin{cases} f_1 = x_n \cdot x_{n-1} \cdots x_1, & \text{I (AND),} \\ f_2 = x_n \oplus x_{n-1} \oplus \dots \oplus x_1, & \text{mod2 (XOR),} \\ f_3 = 1. \end{cases}$$

Операція кон'юнкції має вищий пріоритет, ніж операція додавання за модулем два.

Аксіоми алгебри Жегалкіна:

$$\begin{array}{ll} x \cdot 0 = 0, & x \oplus 0 = x, \\ x \cdot 1 = x, & x \oplus 1 = \bar{x}, \\ x \cdot x = x, & x \oplus x = 0, \\ x \cdot \bar{x} = 0, & x \oplus \bar{x} = 1. \end{array}$$

Розглянемо основні закони (властивості) алгебри Жегалкіна.

- Властивість комутативності: $x_2 \cdot x_1 = x_1 \cdot x_2$,
(переставний закон) $x_2 \oplus x_1 = x_1 \oplus x_2$,
- Властивість асоціативності: $x_3 \cdot x_2 \cdot x_1 = (x_3 \cdot x_2) \cdot x_1$,
(сполучний закон) $x_3 \oplus x_2 \oplus x_1 = (x_3 \oplus x_2) \oplus x_1$,
- Властивість дистрибутивності: $x_3(x_2 \oplus x_1) = x_3 \cdot x_2 \oplus x_3 \cdot x_1$,
(розподільний закон).

Кон'юнкція дистрибутивна відносно додавання за модулем два. Дистрибутивність додавання за модулем два відносно кон'юнкції не виконується: $x_3 \oplus x_2 x_1 \neq (x_3 \oplus x_2) (x_3 \oplus x_1)$.

Інверсія змінної в алгебрі Жегалкіна ґрунтується на аксіомі $x_i \oplus 1 = \bar{x}_i$. Відповідно, апаратна реалізація інверсії має вигляд:



В алгебрі Жегалкіна будь-яку перемикальну функцію можна подати в канонічній формі (кфЖ), що має назву поліном Жегалкіна.

Поліном Жегалкіна має вигляд:

у випадку 1-ї змінної: $f(x) = a_0 \oplus a_1 x$,

у випадку 2-х змінних: $f(x_2, x_1) = a_0 \oplus a_1 x \oplus a_2 x_2 \oplus a_{12} x_1 x_2$,

у випадку 3-х змінних: $f(x_3, x_2, x_1) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_3 x_3 \oplus a_{12} x_1 x_2 \oplus a_{13} x_1 x_3 \oplus a_{23} x_2 x_3 \oplus a_{123} x_1 x_2 x_3$,

у випадку n змінних: $f(x_n, x_{n-1}, \dots, x_1) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \dots \oplus a_n x_n \oplus a_{12} x_1 x_2 \oplus \dots \oplus a_{1n} x_1 x_n \oplus a_{23} x_2 x_3 \oplus \dots \oplus a_{2n} x_2 x_n \oplus \dots \oplus a_{12\dots n} x_1 x_2 \dots x_n$, де $a \in \{0, 1\}$.

Поліном Жегалкіна є нормальною (дворівневою) формою, яку називають формою І/Виключне АБО (AND/XOR) – І є внутрішньою операцією, а XOR – зовнішньою. Відповідно, комбінаційна схема, що реалізує поліном Жегалкіна, є двокаскадною: перший каскад утворюють елементи І, другий – елемент XOR.

Поліном Жегалкіна ще називають поліномом за модулем два.

Степенем полінома Жегалкіна є найбільша кількість змінних, які входять у кон'юнктивні терми.

У поліномі Жегалкіна нульового степеня будь-які змінні відсутні; такий поліном дорівнює сталій – 0 чи 1: $f(x_n, \dots, x_1) = 0, f(x_n, \dots, x_1) = 1$.

Поліном $f(x_4, x_3, x_2, x_1) = x_1 \oplus x_3 \oplus x_1 x_2 \oplus x_1 x_3 \oplus x_1 x_2 x_4$ має степінь три.

У поліномі Жегалкіна перемикальної функції від n змінних максимально можлива кількість доданків становить 2^n , тому від n змінних можна утворити загалом 2^{2^n} різних поліномів Жегалкіна.

Оскільки довільній перемикальній функції можна поставити в однозначну відповідність поліном Жегалкіна, то це означає, що *будь-яку перемикальну функцію можна реалізувати на основі елементів І та суматорів за модулем два.*

Як отримати поліном Жегалкіна заданої перемикальної функції?

Залежність коефіцієнтів a від значень перемикальної функції $f(x_n, \dots, x_1)$ на різних наборах значень аргументів у загальному випадку визначати складно. Тому на практиці поліном Жегалкіна отримують з інших канонічних форм:

- з канонічної форми І-НЕ/І-НЕ (алгебра Шеффера),
- з канонічної форми АБО-НЕ/АБО-НЕ (алгебра Пірса),
- з канонічної форми І/АБО (ДДНФ) (алгебра Буля).

Розглянемо спосіб отримання полінома Жегалкіна з ДДНФ перемикальної функції.

1. Записати задану перемикальну функцію в ДДНФ.
2. Замінити знак операції АБО між кон'юнкціями (термами) на знак операції Виключне АБО ($\vee \rightarrow \oplus$).

Пояснення. Така заміна є правомірною, оскільки ДДНФ складається з конститuent одиниці. Це означає, що на будь-якому з наборів аргументів лише одна конститuent набуває значення одиниці, а решта конститuent – нульових значень. Тобто виконується рівність виду $1 \vee 0 \vee \dots \vee 0 = 1 \oplus 0 \oplus \dots \oplus 0$, де положення одиниці в запису може бути довільним.

3. Для кожного аргументу із запереченням виконати заміну: $\overline{x_i} \rightarrow x_i \oplus 1$.
4. Розкрити дужки та спростити отриманий вираз, використовуючи аксіому $x \oplus x = 0$ та її узагальнення:

$$\underbrace{x \oplus x \oplus \dots \oplus x}_{\text{парна кількість}} = 0, \quad \underbrace{x \oplus x \oplus \dots \oplus x}_{\text{непарна кількість}} = x.$$

парна кількість

непарна кількість

Задача 2.12. Дано таблицю істинності перемикальної функції γ . Реалізувати її в елементному базисі алгебри Жегалкіна.

x_3	x_2	x_1	γ
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Розв'язування.

Спочатку отримаємо канонічну форму функції γ в алгебрі Жегалкіна.

1. Запишемо ДДНФ перемикальної функції:

$$\gamma_{\text{ДДНФ}} = \overline{x_3} \overline{x_2} \overline{x_1} \vee \overline{x_3} x_2 \overline{x_1} \vee x_3 x_2 \overline{x_1}.$$

2. Замінімо знак \vee на \oplus :

$$\gamma = \overline{x_3} \overline{x_2} \overline{x_1} \oplus \overline{x_3} x_2 \overline{x_1} \oplus x_3 x_2 \overline{x_1}.$$

3. Виконаємо заміну $\overline{x_i} \rightarrow x_i \oplus 1$:

$$\gamma = (x_3 \oplus 1)(x_2 \oplus 1)(x_1 \oplus 1) \oplus (x_3 \oplus 1)x_2(x_1 \oplus 1) \oplus x_3 x_2(x_1 \oplus 1).$$

4. Розкриємо дужки та спростимо вираз:

$$\begin{aligned} \gamma &= (x_3 x_2 \oplus x_3 \oplus x_2 \oplus 1)(x_1 \oplus 1) \oplus (x_3 x_2 \oplus x_2)(x_1 \oplus 1) \oplus \\ &\oplus x_3 x_2 x_1 \oplus x_3 x_2 = \underline{x_3 x_2 x_1} + \underline{x_3 x_2} \oplus x_3 x_1 \oplus x_3 \oplus x_2 x_1 \oplus \\ &\oplus x_2 \oplus x_1 \oplus 1 \oplus \underline{x_3 x_2 x_1} \oplus \underline{x_3 x_2} \oplus x_2 x_1 \oplus x_2 \oplus \underline{x_3 x_2 x_1} \oplus \underline{x_3 x_2} = \\ &= x_3 x_2 x_1 \oplus x_3 x_2 \oplus x_3 x_1 \oplus x_3 \oplus x_1 \oplus 1 = \gamma_{\text{кфж}} \end{aligned}$$

Отриману канонічну форму реалізує комбінаційна схема на рис. 2.19. ■

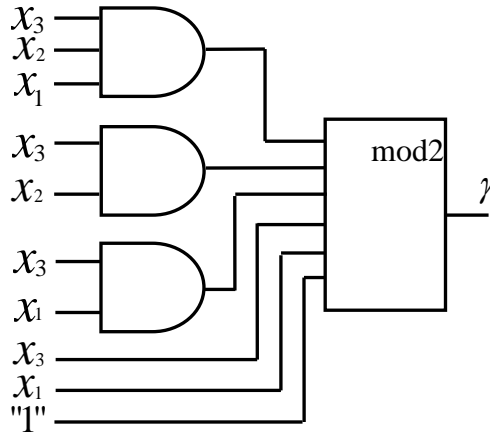


Рис. 2.19. Реалізація функції $\gamma_{\text{кфж}}$ в елементному базисі алгебри Жегалкіна

У деяких випадках при побудові комбінаційних схем використовують двовходові суматори за модулем два. Тоді отриману канонічну форму слід перетворити так, щоб враховувалась кількість входів суматора за модулем два.

Нехай отриману канонічну форму $\gamma_{\text{кфж}}$ необхідно реалізувати на елементах 1 та 2-входових суматорах за модулем два. Перетворимо $\gamma_{\text{кфж}}$ двома способами.

Спосіб 1: $\gamma = (((x_3 x_2 x_1 \oplus x_3 x_2) \oplus x_3 x_1) \oplus x_3) \oplus x_1 \oplus 1.$

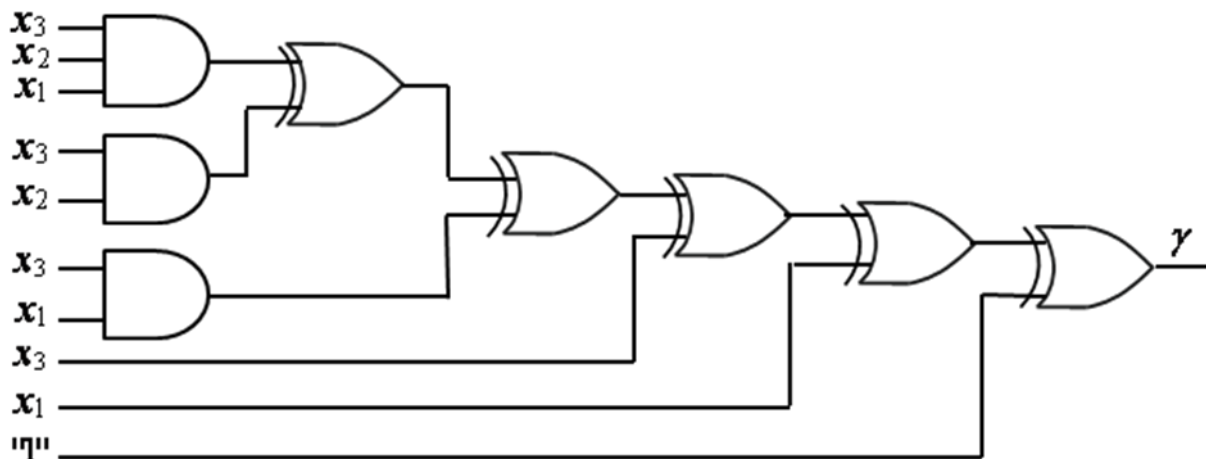
Такий спосіб перетворення реалізує лінійну згортку суми за модулем два (рис. 2.20а) й потребує 6 каскадів логічних елементів.

Спосіб 2: $\gamma = ((x_3 x_2 x_1 \oplus x_3 x_2) \oplus x_3 x_1) \oplus (x_3 \oplus (x_1 \oplus 1)).$

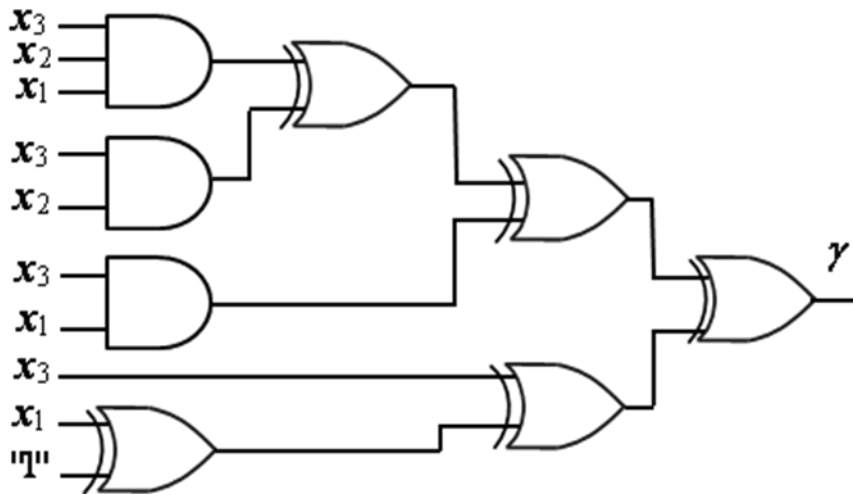
Таке перетворення реалізує пірамідальну згортку суми за модулем два (рис. 2.20б) й потребує 4 каскади логічних елементів.

Обидва варіанти реалізації на рис. 2.20 мають однакову складність за Квайном, але різну швидкодію.

Знайдемо зв'язок між алгеброю Жегалкіна та алгеброю Буля.



a



б

Рис. 2.20. Реалізація функції γ на логічних елементах І та двохходових суматорах за модулем два:
 а – лінійна згортка суми за модулем два;
 б – пірамідальна згортка суми за модулем два

Задача 2.13. Перемикальна функція y задана таблицею істинності. Отримати канонічну форму функції y в алгебрі Жегалкіна.

Розв'язування.

x_2	x_1	y
0	0	0
0	1	1
1	0	1
1	1	1

1. $y_{\text{ДДНФ}} = \bar{x}_2 x_1 \vee x_2 \bar{x}_1 \vee x_2 x_1.$

2. $y = \bar{x}_2 x_1 \oplus x_2 \bar{x}_1 \oplus x_2 x_1.$

3. $y = (x_2 \oplus 1) x_1 \oplus x_2 (x_1 \oplus 1) \oplus x_2 x_1.$

4. $y_{\text{кфЖ}} = x_2 x_1 \oplus x_1 \oplus x_2 x_1 \oplus x_2 \oplus x_2 x_1 = x_2 x_1 \oplus x_2 \oplus x_1. \blacksquare$

Але y – це перемикальна функція АБО (див. таблицю істинності). Отже, $x_2 \vee x_1 = x_2 x_1 \oplus x_2 \oplus x_1$. Таким чином, між алгеброю Буля (операції I, АБО, НЕ) та алгеброю Жегалкіна (операції I, Виключне АБО, 1) є наступний зв'язок:

$$\begin{cases} x_2 x_1 = x_2 x_1, \\ x_2 \vee x_1 = x_2 \oplus x_1 \oplus x_2 x_1, \\ \bar{x} = x \oplus 1. \end{cases} \quad (2.3)$$

Це означає, що для переходу від формул алгебри Буля до формул алгебри Жегалкіна необхідно в формулах алгебри Буля залишити без змін операції кон'юнкції, а операції диз'юнкції та заперечення замінити відповідно до (2.3).

Якщо перемикальну функцію задано в формі відмінній від ДДНФ, то для отримання канонічної форми цієї функції в алгебрі Жегалкіна необхідно спочатку перейти до ДДНФ заданої функції, а потім за правилом отримання полінома Жегалкіна з ДДНФ функції отримати потрібну канонічну форму.

Вигляд полінома Жегалкіна використовують для з'ясування (визначення) властивості лінійності перемикальної функції.

Перемикальна функція називається *лінійною*, якщо її поліном Жегалкіна містить кон'юнктивні терми лише 1-го рангу, інакше – функція є нелінійною.

Наприклад, функція y із задачі 2.13 є нелінійною, оскільки її поліном Жегалкіна $Y_{\text{кфж}} = x_1 \oplus x_2 \oplus x_1 x_2$ містить терм 2-го рангу $x_1 x_2$, а функція f , поліном Жегалкіна якої має вигляд $f(x_2, x_1) = 1 \oplus x_1 \oplus x_2$, є лінійною.

Запитання та завдання

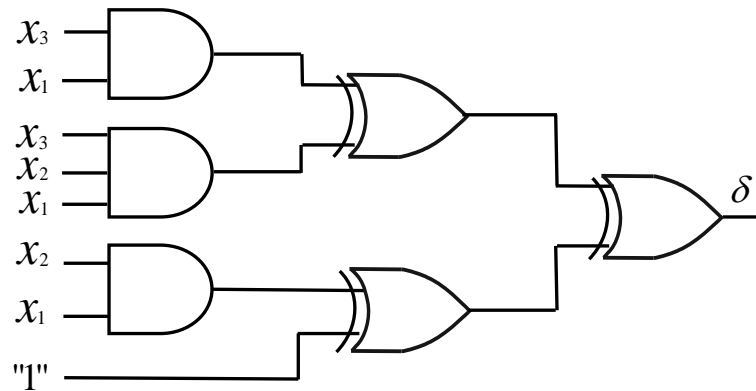
1. Дайте визначення алгебри Жегалкіна.
2. Сформулюйте аксіоми алгебри Жегалкіна.
3. Якою є канонічна форма подання перемикальних функцій в алгебрі Жегалкіна?
4. Яким є елементний базис алгебри Жегалкіна?
5. Сформулюйте правило отримання полінома Жегалкіна з ДДНФ перемикальної функції?
6. Як отримати поліном Жегалкіна, якщо перемикальна функція задана в формі відмінній від ДДНФ?
7. Як здійснити перехід від формул алгебри Буля до формул алгебри Жегалкіна?

8. Яка перемикальна функція називається лінійною?
9. Запишіть поліном Жегалкіна для кожної із заданих функцій:
 $f_0(x) = 0$, $f_1(x) = x$, $f_2(x) = \bar{x}$, $f_3(x) = 1$, $f_4(x_2, x_1) = x_2 x_1$,
 $F_8(x_2, x_1) = \overline{x_2 \vee x_1}$, $F_{14}(x_2, x_1) = \overline{x_2 x_1}$.
10. Реалізуйте функції $f_1(x) = x$ та $f_2(x) = \bar{x}$ в елементному базисі алгебри Жегалкіна.

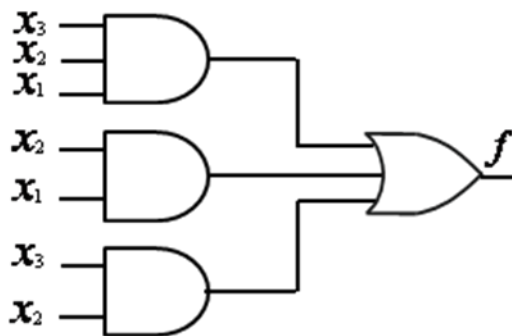
Задачі для самостійного розв'язування

- Повністю визначена перемикальна функція від трьох змінних $f(x_3, x_2, x_1)$ дорівнює нулю на наборах 1, 2, 5, 6, а на решті наборів – одиниці. Подати f в канонічній формі алгебри Жегалкіна.
- Повністю визначена перемикальна функція від трьох змінних $y(a, b, c)$ дорівнює нулю на наборах 0, 3, 4, 6, 7. Реалізувати функцію y на логічних елементах 2І, 2XOR. Цільова функція проектування – максимальна швидкодія.
- Перемикальну функцію y від трьох змінних подано в канонічній формі алгебри Шеффера: $y_{кфШ} = \overline{\bar{x}_3 x_2 x_1 \cdot x_3 \bar{x}_2 x_1 \cdot x_3 \bar{x}_2 \bar{x}_1}$.
 Реалізувати функцію y в елементному базисі алгебри Жегалкіна.
- Перемикальну функцію ω від трьох змінних подано в канонічній формі алгебри Пірса:
 $\omega_{кфП} = \overline{a \vee b \vee c \vee a \vee \bar{b} \vee c \vee a \vee b \vee c \vee a \vee b \vee \bar{c}}$.
 Побудувати комбінаційну схему, що реалізує функцію ω на елементах 3І, 2XOR. Оцінити складність схеми за Квайном.
- Відома ДКНФ повністю визначеної перемикальної функції від трьох змінних: $z_{ДКНФ} = (x_3 \vee x_2 \vee \bar{x}_1)(x_3 \vee \bar{x}_2 \vee \bar{x}_1)(\bar{x}_3 \vee \bar{x}_2 \vee x_1)$.
 Отримати поліном Жегалкіна перемикальної функції z .
- Подати перемикальну функцію y від трьох змінних в канонічній формі алгебри Жегалкіна: а) $y = \bar{a}b \vee \bar{a}\bar{b}c \vee b\bar{c}$, б) $y = abc \vee \bar{b}c \vee \bar{a}c$.
- Формулу $x_2 \vee x_1 = x_2 \oplus x_1 \oplus x_2 x_1$, яка встановлює зв'язок між операцією OR в алгебрі Буля та операціями AND і XOR в алгебрі Жегалкіна, поширити на випадок трьох змінних x_3, x_2, x_1 .

8. Дано комбінаційну схему, що реалізує повністю визначену перемикальну функцію δ від трьох змінних в елементному базисі алгебри Жегалкіна. Подати функцію в канонічній формі алгебри Пірса.



9. Дано комбінаційну схему, що реалізує функцію f , в елементному базисі алгебри Буля.



Реалізувати функцію f в елементному базисі алгебри Жегалкіна.

2.5. Декомпозиція перемикальних функцій

Зі збільшенням кількості аргументів перемикальної функції процес її апаратної реалізації значно ускладнюється. Необхідно знайти такий прийом, який дозволяв би представити функцію від заданої кількості аргументів як об'єднання кількох функцій, що залежать від меншої кількості аргументів. Тоді можна виконати апаратну реалізацію окремо кожної з функцій з меншою кількістю аргументів (що значно простіше) та об'єднати отримані результати.

Подання функції з більшою кількістю аргументів як об'єднання кількох функцій з меншою кількістю аргументів називають *декомпозицією (розкладанням) перемикальної функції*.

Візьмемо деяку функцію $\gamma(x)$, що залежить лише від одного аргументу.

Нехай $\gamma(x) = \bar{x}$ (рис. 2.21).

x	$\gamma(x)$	\Rightarrow	x	$\gamma(x)$	Примітки
0	1		0	1	$\gamma(0) = \gamma_0$
1	0		1	0	$\gamma(1) = \gamma_1$

Рис. 2.21. Розкладання функції $\gamma(x)$ від одного аргументу

Доведемо, що

$$\gamma(x) = \bar{x}\gamma(0) \vee x\gamma(1). \quad (2.4)$$

Поділимо таблицю істинності функції $\gamma(x)$ пунктиром на верхню та нижню частини. Верхній частині відповідає $x = 0$, а нижній – $x = 1$. Значенням функції у верхній частині таблиці істинності є $\gamma(0)$, а в нижній – $\gamma(1)$. Але $\gamma(0) = 1$, а $\gamma(1) = 0$ (див. рис. 2.21). Тоді підставивши ці значення в (2.4), отримаємо $\gamma(x) = \bar{x}\gamma(0) \vee x\gamma(1) = \bar{x} \cdot 1 \vee x \cdot 0 = \bar{x}$, що й доводить справедливість формули (2.4).

Позначивши $\gamma(0)$ як γ_0 , а $\gamma(1)$ як γ_1 запишемо (2.4) у вигляді

$$\gamma(x) = \bar{x}\gamma_0 \vee x\gamma_1.$$

Формула (2.4) з одного боку є математичним записом процедури об'єднання верхньої та нижньої частин таблиці істинності перемикальної функції від одного аргументу, а з іншого – розкладанням функції від однієї змінної $\gamma(x)$ на дві функції γ_0 та γ_1 , що не залежать від аргументів.

Візьмемо деяку перемикальну функцію $\varphi(x_2, x_1)$, задану таблицею істинності (рис. 2.22), що залежить від двох аргументів.

x_2	x_1	$\varphi(x_2, x_1)$	\Rightarrow	x_2	x_1	$\varphi(x_2, x_1)$	Примітки
0	0	1		0	0	1	} $\varphi(0, x_1) = \varphi_0(x_1)$
0	1	0		0	1	0	
1	0	1		1	0	1	} $\varphi(1, x_1) = \varphi_1(x_1)$
1	1	1	1	1	1		

Рис. 2.22. Розкладання функції $\varphi(x_2, x_1)$ від двох аргументів за змінною x_2

Доведемо, що

$$\varphi(x_2, x_1) = \bar{x}_2\varphi(0, x_1) \vee x_2\varphi(1, x_1). \quad (2.5)$$

Поділимо таблицю істинності функції на дві частини – верхню, якій відповідає $x_2 = 0$, та нижню, якій відповідає $x_2 = 1$. Але для верхньої

частини таблиці істинності значеннями функції є $\varphi(0, x_1)$, а для нижньої – $\varphi(1, x_1)$.

Об'єднаємо дві частини (верхню та нижню) таблиці істинності:

$$\begin{aligned}\varphi(x_2, x_1) &= (x_2 = 0) \& \varphi(0, x_1) \cup (x_2 = 1) \& \varphi(1, x_1) = \\ &= \bar{x}_2 \varphi(0, x_1) \vee x_2 \varphi(1, x_1),\end{aligned}$$

що й доводить справедливість (2.5).

Позначивши $\varphi(0, x_1)$ як $\varphi_0(x_1)$, а $\varphi(1, x_1)$ як $\varphi_1(x_1)$, запишемо (2.5) у вигляді

$$\varphi(x_2, x_1) = \bar{x}_2 \varphi_0(x_1) \vee x_2 \varphi_1(x_1),$$

де $\varphi_0(x_1)$, $\varphi_1(x_1)$ – функції від однієї змінної.

Таким чином, запис (2.5) є розкладанням (декомпозицією) функції від двох змінних за змінною x_2 .

Але декомпозицію (розкладання) функції $\varphi(x_2, x_1)$ можна виконати й за змінною x_1 (рис. 2.23):

$$\varphi(x_2, x_1) = \bar{x}_1 \varphi(x_2, 0) \vee x_1 \varphi(x_2, 1). \quad (2.6)$$

$x_2 \ x_1$	$\varphi(x_2, x_1)$	\Rightarrow	$x_2 \ x_1$	$\varphi(x_2, x_1)$	Примітки
0 0	1		0 0	1	} $\varphi(x_2, 0) = \Psi_0(x_2)$
0 1	0		1 0	1	
1 0	1		0 1	0	} $\varphi(x_2, 1) = \Psi_1(x_2)$
1 1	1		1 1	1	

Рис. 2.23. Розкладання функції $\varphi(x_2, x_1)$ від двох аргументів за змінною x_1

Для отримання функцій $\varphi(x_2, 0)$ та $\varphi(x_2, 1)$ необхідно в таблиці істинності функції $\varphi(x_2, x_1)$ переставити рядки так, щоб у верхній частині таблиці опинились ті рядки, в яких $x_1 = 0$, а в нижній – у яких $x_1 = 1$ (див. рис. 2.23).

Позначивши $\varphi(x_2, 0)$ як $\Psi_0(x_2)$, а $\varphi(x_2, 1)$ як $\Psi_1(x_2)$ запишемо (2.6) у вигляді

$$\varphi(x_2, x_1) = \bar{x}_1 \Psi_0(x_2) \vee x_1 \Psi_1(x_2),$$

де $\Psi_0(x_2)$, $\Psi_1(x_2)$ – функції від однієї змінної.

Формули (2.4) – (2.6) узагальнює наступна теорема.

Теорема Шеннона. Будь-яку перемикальну функцію від n змінних можна розкласти на перемикальні функції, що залежать від меншої за n кількості змінних, причому таке розкладання можна виконати за будь-

якою окремо взятою змінною, за будь-якою кількістю змінних, і навіть за всіма змінними.

Нехай перемикальна функція $f(x_n, x_{n-1}, \dots, x_1)$, що залежить від n змінних, задана таблицею істинності (рис. 2.24). Поділимо таблицю істинності на дві частини – верхню, для якої $x_n = 0$, та нижню, для якої $x_n = 1$. Верхній частині таблиці істинності відповідає функція $f(0, x_{n-1}, \dots, x_1)$, що залежить лише від $n - 1$ змінних (позначимо її $f_0(x_{n-1}, \dots, x_1)$), а нижній – функція $f(1, x_{n-1}, \dots, x_1)$, яка також залежить лише від $n - 1$ змінних (позначимо її $f_1(x_{n-1}, \dots, x_1)$).

Таблиця істинності функції від n змінних є об'єднанням двох менших таблиць – верхньої та нижньої. Так само й функцію від n змінних можна подати як об'єднання двох простіших функцій – таких, що залежать лише від $n - 1$ змінних, а саме (див. рис. 2.24):

$$\begin{aligned} f(x_n, x_{n-1}, \dots, x_1) &= (x_n = 0) \& f(0, x_{n-1}, \dots, x_1) \cup (x_n = 1) \& f(1, x_{n-1}, \dots, x_1) = \\ &= \bar{x}_n f_0(x_{n-1}, \dots, x_1) \vee x_n f_1(x_{n-1}, \dots, x_1) = \bar{x}_n f_0(x_{n-1}, \dots, x_1) \vee x_n f_1(x_{n-1}, \dots, x_1) = \\ &= \bar{x}_n f_0 \vee x_n f_1. \end{aligned}$$

x_n	x_{n-1}	\dots	x_1	$f(x_n, x_{n-1}, \dots, x_1)$	\equiv	x_n	x_{n-1}	\dots	x_1	$f(x_n, x_{n-1}, \dots, x_1)$		
0	0	\dots	0	} $f(0, x_{n-1}, \dots, x_1)$	\equiv	0	0	\dots	0	} $\bar{x}_n f_0(x_{n-1}, \dots, x_1)$		
0	0	\dots	1			0	0	\dots	1		0	
\vdots	\vdots	\vdots	\vdots			\vdots	\vdots	\vdots	\vdots		\vdots	\vdots
0	1	\dots	1			0	0	\dots	1		1	0
1	0	\dots	0	} $f(1, x_{n-1}, \dots, x_1)$	\equiv	1	0	\dots	0	} $x_n f_1(x_{n-1}, \dots, x_1)$		
1	0	\dots	1			1	0	\dots	1		1	
\vdots	\vdots	\vdots	\vdots			\vdots	\vdots	\vdots	\vdots		\vdots	\vdots
1	1	\dots	1			1	1	\dots	1		1	1

Рис. 2.24. Розкладання перемикальної функції $f(x_n, x_{n-1}, \dots, x_1)$ за змінною x_n

Отриманий запис є розкладанням функції від n змінних за змінною x_n на дві простіші функції, кожна з яких залежить лише від $n - 1$ змінних.

Розкласти функцію $f(x_n, x_{n-1}, \dots, x_1)$ можна й за будь-якою іншою змінною, наприклад, за x_{n-1} :

$$\begin{aligned} f(x_n, x_{n-1}, \dots, x_1) &= \bar{x}_{n-1} f(x_n, 0, x_{n-2}, \dots, x_1) \vee x_{n-1} f(x_n, 1, x_{n-1}, \dots, x_1) = \\ &= \bar{x}_{n-1} \theta_0(x_n, x_{n-2}, \dots, x_1) \vee x_{n-1} \theta_1(x_n, x_{n-2}, \dots, x_1) = \bar{x}_{n-1} \theta_0 \vee x_{n-1} \theta_1. \end{aligned}$$

Для цього необхідно переставити рядки в таблиці істинності функції $f(x_n, x_{n-1}, \dots, x_1)$ так, щоб у верхній частині таблиці опинились рядки, в яких $x_{n-1} = 0$, а в нижній – рядки, в яких $x_{n-1} = 1$.

Отже, та сама функція $f(x_n, x_{n-1}, \dots, x_1)$ від n змінних розкладена також на інші дві простіші функції θ_0 та θ_1 , кожна з яких залежить лише від $n - 1$ змінних, – але за змінною x_{n-1} .

Розкладання функції

$$f(x_n, x_{n-1}, \dots, x_1)$$

за змінною x_i ($i = n, n - 1, \dots, 1$) виглядає так:

$$\begin{aligned} f(x_n, x_{n-1}, \dots, x_1) &= \bar{x}_i f(x_n, \dots, x_{i+1}, 0, x_{i-1}, \dots, x_1) \vee x_i f(x_n, \dots, x_{i+1}, 1, x_{i-1}, \dots, x_1) = \\ &= \bar{x}_i \delta_0(x_n, \dots, x_{i+1}, x_{i-1}, \dots, x_1) \vee x_i \delta_1(x_n, \dots, x_{i+1}, x_{i-1}, \dots, x_1) = \bar{x}_i \delta_0 \vee x_i \delta_1. \end{aligned}$$

Розкладання перемикальної функції за однією змінною означає на практиці, що замість побудови комбінаційної схеми $КС(f)$, що реалізує функцію $f(x_n, x_{n-1}, \dots, x_1)$ від n змінних, можна побудувати дві простіші комбінаційні схеми – $КС(\delta_0)$ та $КС(\delta_1)$, кожна з яких реалізує простішу функцію, що залежить лише від $n - 1$ змінних, та об'єднати їх.

В алгебрі Буля таке об'єднання виглядає як показано на рис. 2.25.

Якщо перемикальну функцію $f(x_n, \dots, x_1)$ необхідно реалізувати в елементному базисі алгебри Шеффера, то розкладання функції f за змінною x_i слід подати у вигляді

$$f(x_n, \dots, x_1) = \bar{x}_i \delta_0 \vee x_i \delta_1 = \overline{\overline{\bar{x}_i \delta_0} \cdot \overline{x_i \delta_1}},$$

а структура комбінаційної схеми матиме вигляд як показано на рис. 2.26.

Розглянемо розкладання перемикальної функції $f(x_n, \dots, x_1)$ за однією змінною в алгебрі Пірса. Відомо, що в алгебрі Буля

$$\begin{aligned} f(x_n, x_{n-1}, \dots, x_1) &= \bar{x}_i f(x_n, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_1) \vee \\ &\vee x_i f(x_n, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_1). \end{aligned}$$

Візьмемо не пряму функцію $f(x_n, \dots, x_1)$, а її заперечення (інверсію), тобто $\bar{f}(x_n, \dots, x_1)$.

Тоді

$$\begin{aligned} \bar{f}(x_n, \dots, x_1) &= \bar{x}_i \bar{f}(x_n, \dots, x_{i+1}, 0, x_{i-1}, \dots, x_1) \vee x_i \bar{f}(x_n, \dots, x_{i+1}, 1, x_{i-1}, \dots, x_1) = \\ &= \bar{x}_i \bar{\delta}_0(x_n, \dots, x_{i+1}, x_{i-1}, \dots, x_1) \vee x_i \bar{\delta}_1(x_n, \dots, x_{i+1}, x_{i-1}, \dots, x_1) = \bar{x}_i \bar{\delta}_0 \vee x_i \bar{\delta}_1. \end{aligned}$$

Поставивши заперечення над лівою та правою частинами отриманого виразу дістанемо:

$$\begin{aligned} \overline{\bar{f}(x_n, \dots, x_1)} &= \overline{\bar{x}_i \bar{\delta}_0 \vee x_i \bar{\delta}_1}, \\ f(x_n, \dots, x_1) &= \overline{\bar{x}_i \bar{\delta}_0 \vee x_i \bar{\delta}_1} = \overline{\overline{\bar{x}_i} \vee \overline{\bar{\delta}_0}} \vee \overline{\overline{x_i} \vee \overline{\bar{\delta}_1}}. \end{aligned}$$

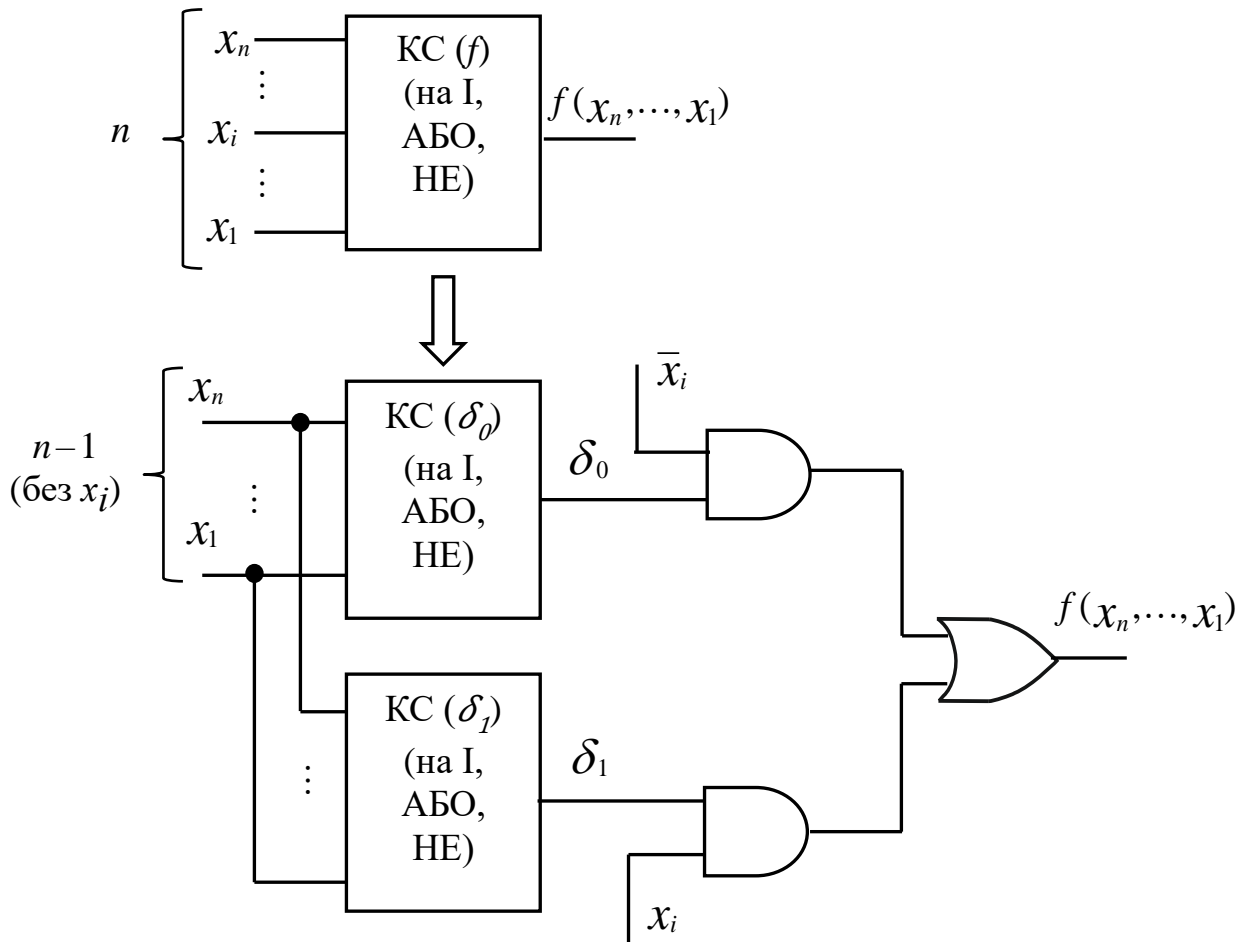


Рис. 2.25. Структура комбінаційної схеми, що реалізує перемикальну функцію $f(x_n, \dots, x_1)$ в елементному базисі алгебри Буля, при розкладанні функції за змінною x_i

Отриманий вираз є розкладанням перемикальної функції f від n змінних за змінною x_i в алгебрі Пірса (рис. 2.27).

Але перемикальну функцію $f(x_n, \dots, x_1)$ від n змінних можна розкласти й за двома змінними.

Наприклад, розкладання функції $f(x_n, \dots, x_1)$ за двома старшими змінними — x_n та x_{n-1} , має вигляд (рис. 2.28):

$$\begin{aligned}
 f(x_n, \dots, x_1) &= \bar{x}_n \bar{x}_{n-1} f(0, 0, x_{n-2}, \dots, x_1) \vee \bar{x}_n x_{n-1} f(0, 1, x_{n-2}, \dots, x_1) \vee \\
 &\vee x_n \bar{x}_{n-1} f(1, 0, x_{n-2}, \dots, x_1) \vee x_n x_{n-1} f(1, 1, x_{n-2}, \dots, x_1) = \bar{x}_n \bar{x}_{n-1} \mathcal{Y}_0(x_{n-2}, \dots, x_1) \vee \\
 &\vee \bar{x}_n x_{n-1} \mathcal{Y}_1(x_{n-2}, \dots, x_1) \vee x_n \bar{x}_{n-1} \mathcal{Y}_2(x_{n-2}, \dots, x_1) \vee x_n x_{n-1} \mathcal{Y}_3(x_{n-2}, \dots, x_1) = \\
 &= \bar{x}_n \bar{x}_{n-1} \mathcal{Y}_0 \vee \bar{x}_n x_{n-1} \mathcal{Y}_1 \vee x_n \bar{x}_{n-1} \mathcal{Y}_2 \vee x_n x_{n-1} \mathcal{Y}_3.
 \end{aligned}$$

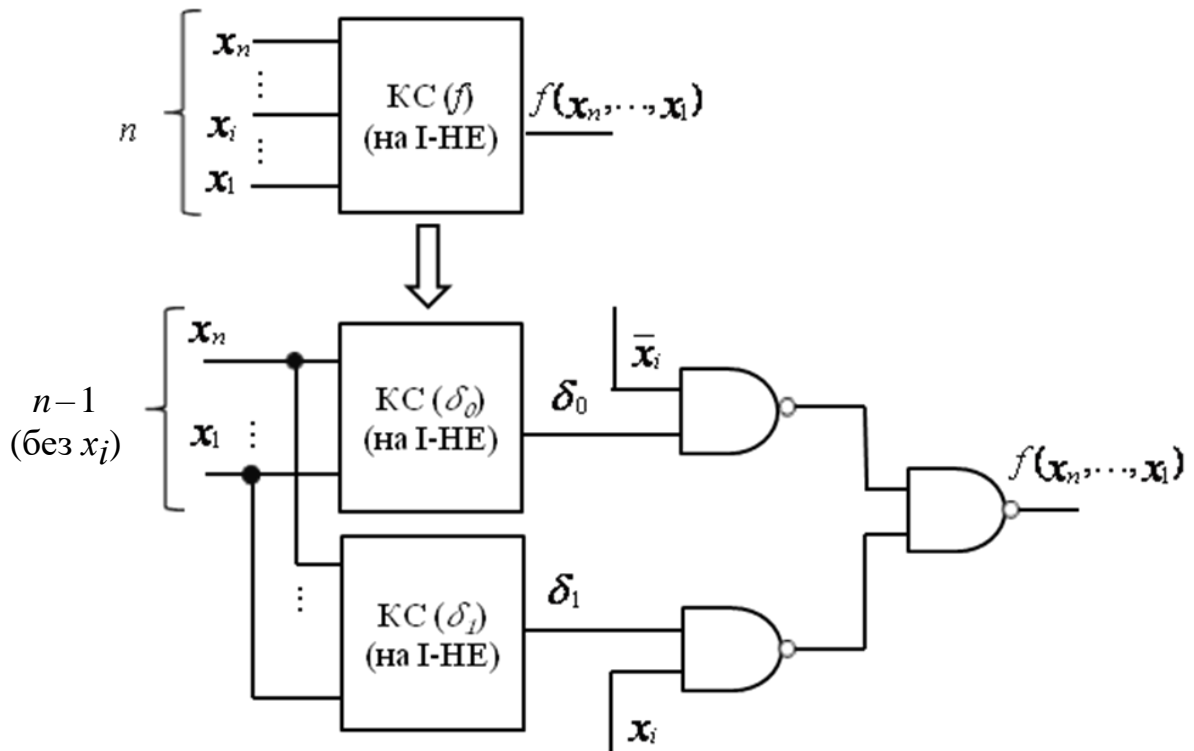


Рис. 2.26. Структура комбінаційної схеми, що реалізує перемикальну функцію $f(x_n, \dots, x_1)$ в елементному базисі алгебри Шеффера, при розкладанні функції за змінною x_i

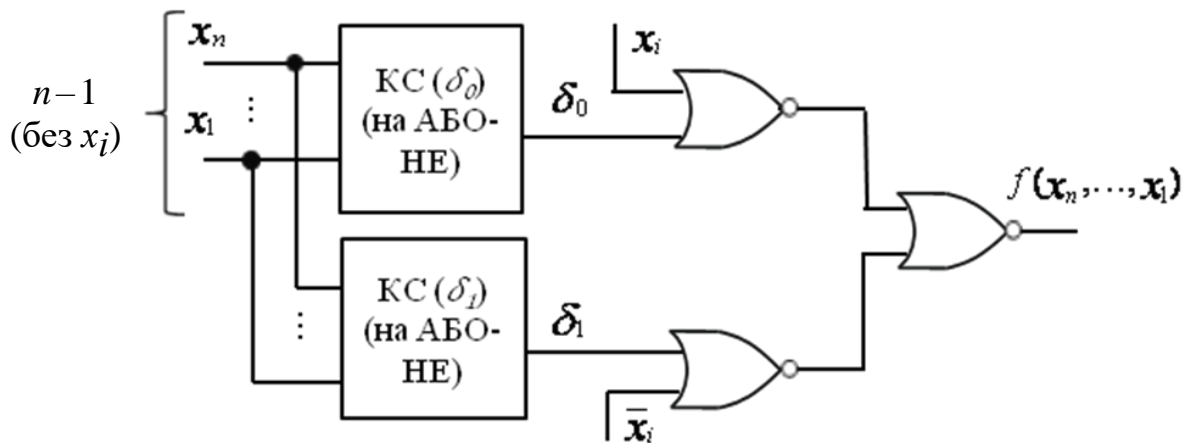


Рис. 2.27. Структура комбінаційної схеми, що реалізує перемикальну функцію $f(x_n, \dots, x_1)$ в елементному базисі алгебри Пірса, при розкладанні функції за змінною x_i

$x_n \ x_{n-1} \ x_{n-2} \ \dots \ x_1$	$f(x_n, \dots, x_1)$	$x_n \ x_{n-1} \ x_{n-2} \ \dots \ x_1$	$f(x_n, \dots, x_1)$
$\left. \begin{array}{c} 0 \ 0 \ 0 \ \dots \ 0 \\ \vdots \\ 0 \ 0 \ 1 \ \dots \ 1 \end{array} \right\} f(0, 0, x_{n-2}, \dots, x_1)$	$\left. \begin{array}{c} 0 \ 0 \\ \vdots \\ 0 \ 0 \end{array} \right\} \bar{x}_n \bar{x}_{n-1} \gamma_0(x_{n-2}, \dots, x_1)$	$\left. \begin{array}{c} 0 \ 0 \\ \vdots \\ 0 \ 0 \end{array} \right\} \begin{array}{ c } \hline 0 \ \dots \ 0 \\ \hline \gamma_0 \\ \hline 1 \ \dots \ 1 \\ \hline \end{array}$	$\left. \begin{array}{c} 0 \ 0 \\ \vdots \\ 0 \ 0 \end{array} \right\} \bar{x}_n \bar{x}_{n-1} \gamma_0(x_{n-2}, \dots, x_1)$
$\left. \begin{array}{c} 0 \ 1 \ 0 \ \dots \ 0 \\ \vdots \\ 0 \ 1 \ 1 \ \dots \ 1 \end{array} \right\} f(0, 1, x_{n-2}, \dots, x_1)$	$\left. \begin{array}{c} 0 \ 1 \\ \vdots \\ 0 \ 1 \end{array} \right\} \bar{x}_n x_{n-1} \gamma_1(x_{n-2}, \dots, x_1)$	$\left. \begin{array}{c} 0 \ 1 \\ \vdots \\ 0 \ 1 \end{array} \right\} \begin{array}{ c } \hline 0 \ \dots \ 0 \\ \hline \gamma_1 \\ \hline 1 \ \dots \ 1 \\ \hline \end{array}$	$\left. \begin{array}{c} 0 \ 1 \\ \vdots \\ 0 \ 1 \end{array} \right\} \bar{x}_n x_{n-1} \gamma_1(x_{n-2}, \dots, x_1)$
\equiv			
$\left. \begin{array}{c} 1 \ 0 \ 0 \ \dots \ 0 \\ \vdots \\ 1 \ 0 \ 1 \ \dots \ 1 \end{array} \right\} f(1, 0, x_{n-2}, \dots, x_1)$	$\left. \begin{array}{c} 1 \ 0 \\ \vdots \\ 1 \ 0 \end{array} \right\} x_n \bar{x}_{n-1} \gamma_2(x_{n-2}, \dots, x_1)$	$\left. \begin{array}{c} 1 \ 0 \\ \vdots \\ 1 \ 0 \end{array} \right\} \begin{array}{ c } \hline 0 \ \dots \ 0 \\ \hline \gamma_2 \\ \hline 1 \ \dots \ 1 \\ \hline \end{array}$	$\left. \begin{array}{c} 1 \ 0 \\ \vdots \\ 1 \ 0 \end{array} \right\} x_n \bar{x}_{n-1} \gamma_2(x_{n-2}, \dots, x_1)$
$\left. \begin{array}{c} 1 \ 1 \ 0 \ \dots \ 0 \\ \vdots \\ 1 \ 1 \ 1 \ \dots \ 1 \end{array} \right\} f(1, 1, x_{n-2}, \dots, x_1)$	$\left. \begin{array}{c} 1 \ 1 \\ \vdots \\ 1 \ 1 \end{array} \right\} x_n x_{n-1} \gamma_3(x_{n-2}, \dots, x_1)$	$\left. \begin{array}{c} 1 \ 1 \\ \vdots \\ 1 \ 1 \end{array} \right\} \begin{array}{ c } \hline 0 \ \dots \ 0 \\ \hline \gamma_3 \\ \hline 1 \ \dots \ 1 \\ \hline \end{array}$	$\left. \begin{array}{c} 1 \ 1 \\ \vdots \\ 1 \ 1 \end{array} \right\} x_n x_{n-1} \gamma_3(x_{n-2}, \dots, x_1)$

Рис. 2.28. Розкладання перемикальної функції $f(x_n, \dots, x_1)$ за змінними x_n та x_{n-1}

Розкладання перемикальної функції за двома змінними означає на практиці, що замість побудови комбінаційної схеми $КС(f)$, що реалізує функцію $f(x_n, \dots, x_1)$ від n змінних, можна побудувати чотири простіші комбінаційні схеми – $КС(\gamma_0)$, $КС(\gamma_1)$, $КС(\gamma_2)$, $КС(\gamma_3)$, кожна з яких реалізує функцію, що залежить лише від $n-2$ змінних, та об'єднати їх.

В алгебрі Буля таке об'єднання показано на рис. 2.29.

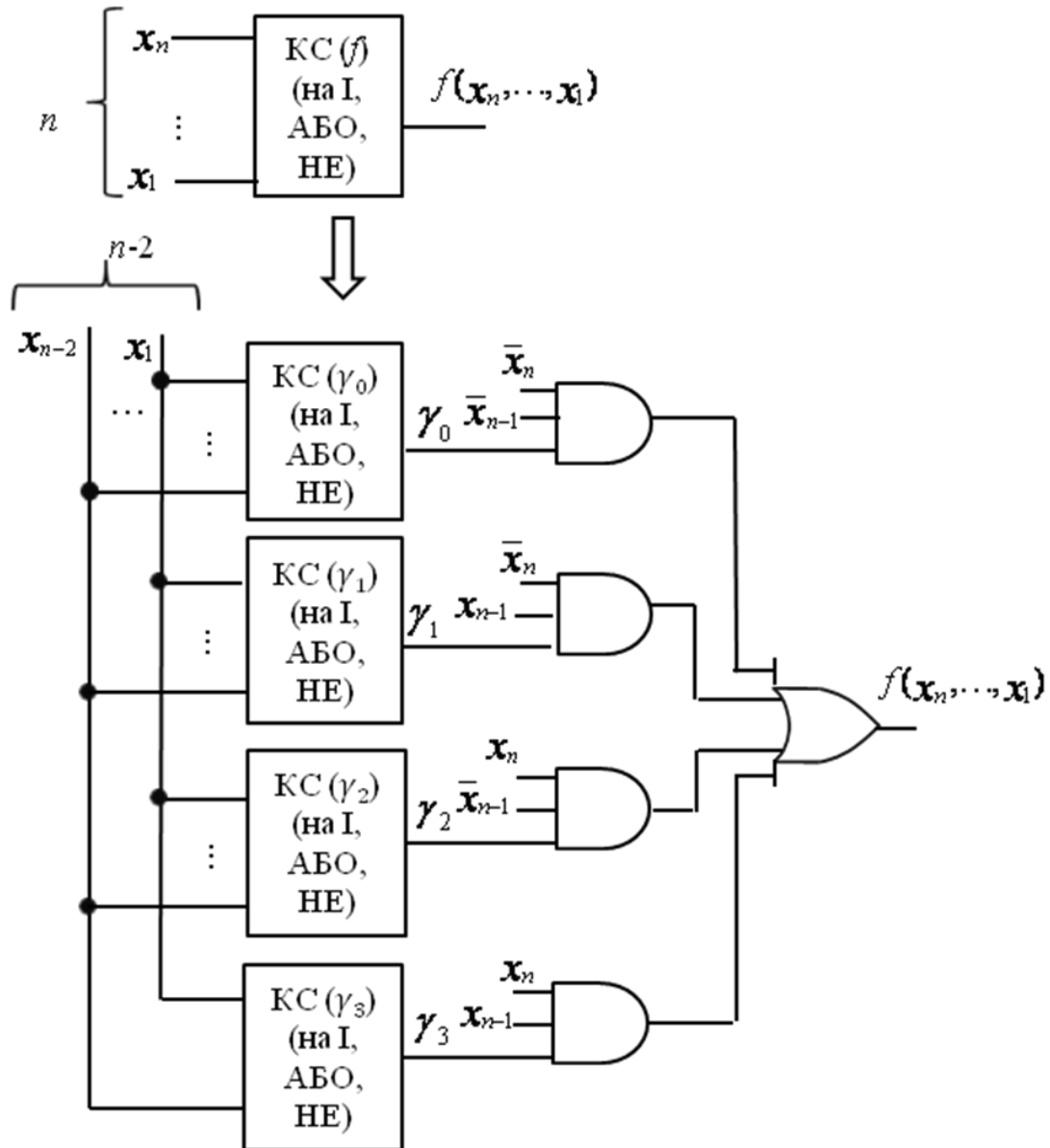


Рис. 2.29. Структура комбінаційної схеми, що реалізує перемикальну функцію $f(x_n, \dots, x_1)$ в елементному базисі алгебри Буля, при розкладанні функції за змінними x_n, x_{n-1}

Якщо перемикальну функцію $f(x_n, \dots, x_1)$ необхідно реалізувати на елементах І-НЕ (алгебра Шеффера), то розкладання функції f за змінними x_n та x_{n-1} слід подати у вигляді

$$f(x_n, \dots, x_1) = \overline{\overline{\overline{x_n} \overline{x_{n-1}} \gamma_0} \vee \overline{\overline{x_n} x_{n-1} \gamma_1} \vee \overline{x_n \overline{x_{n-1}} \gamma_2} \vee \overline{x_n x_{n-1} \gamma_3}} =$$

$$= \overline{\overline{\overline{x_n} \overline{x_{n-1}} \gamma_0} \cdot \overline{\overline{x_n} x_{n-1} \gamma_1} \cdot \overline{x_n \overline{x_{n-1}} \gamma_2} \cdot \overline{x_n x_{n-1} \gamma_3}},$$

а структура комбінаційної схеми матиме вигляд як показано на рис. 2.30.

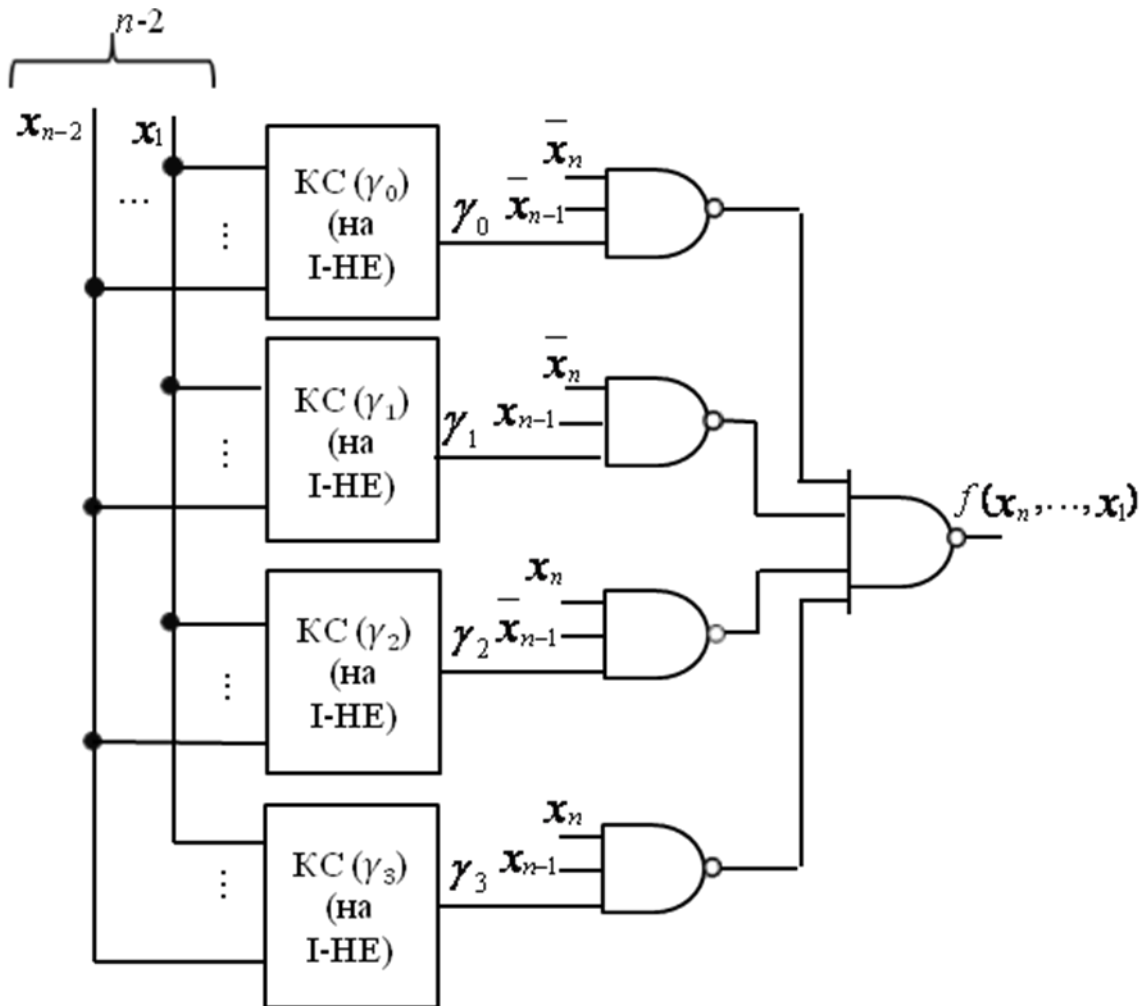


Рис. 2.30. Структура комбінаційної схеми, що реалізує перемикальну функцію $f(x_n, \dots, x_1)$ в елементному базисі алгебри Шеффера, при розкладанні функції за змінними x_n, x_{n-1}

Отримаємо аналітичний вираз для розкладання функції $f(x_n, \dots, x_1)$ за змінними x_n, x_{n-1} в алгебрі Пірса.

Спочатку візьмемо не пряму функцію $f(x_n, \dots, x_1)$, а її заперечення, тобто $\bar{f}(x_n, \dots, x_1)$. Тоді

$$\begin{aligned} \bar{f}(x_n, \dots, x_1) &= \bar{x}_n \bar{x}_{n-1} \bar{f}(0, 0, x_{n-2}, \dots, x_1) \vee \bar{x}_n x_{n-1} \bar{f}(0, 1, x_{n-2}, \dots, x_1) \vee \\ &\vee x_n \bar{x}_{n-1} \bar{f}(1, 0, x_{n-2}, \dots, x_1) \vee x_n x_{n-1} \bar{f}(1, 1, x_{n-2}, \dots, x_1) = \\ &= \bar{x}_n \bar{x}_{n-1} \bar{\gamma}_0(x_{n-2}, \dots, x_1) \vee \bar{x}_n x_{n-1} \bar{\gamma}_1(x_{n-2}, \dots, x_1) \vee x_n \bar{x}_{n-1} \bar{\gamma}_2(x_{n-2}, \dots, x_1) \vee \\ &\vee x_n x_{n-1} \bar{\gamma}_3(x_{n-2}, \dots, x_1) = \bar{x}_n \bar{x}_{n-1} \bar{\gamma}_0 \vee \bar{x}_n x_{n-1} \bar{\gamma}_1 \vee x_n \bar{x}_{n-1} \bar{\gamma}_2 \vee x_n x_{n-1} \bar{\gamma}_3. \end{aligned}$$

Поставивши заперечення над лівою та правою частиною отриманого виразу дістанемо

$$\begin{aligned} f(x_n, \dots, x_1) &= \overline{\bar{x}_n \bar{x}_{n-1} \bar{\gamma}_0 \vee \bar{x}_n x_{n-1} \bar{\gamma}_1 \vee x_n \bar{x}_{n-1} \bar{\gamma}_2 \vee x_n x_{n-1} \bar{\gamma}_3} = \\ &= \overline{\bar{x}_n \vee x_{n-1} \vee \bar{\gamma}_0 \vee \bar{x}_n \vee x_{n-1} \vee \bar{\gamma}_1 \vee x_n \vee x_{n-1} \vee \bar{\gamma}_2 \vee x_n \vee x_{n-1} \vee \bar{\gamma}_3} = \\ &= x_n \vee x_{n-1} \vee \gamma_0 \vee x_n \vee x_{n-1} \vee \gamma_1 \vee x_n \vee x_{n-1} \vee \gamma_2 \vee x_n \vee x_{n-1} \vee \gamma_3. \end{aligned}$$

Отриманий вираз є розкладанням перемикальної функції f від n змінних за змінними x_n, x_{n-1} в алгебрі Пірса (рис. 2.31).

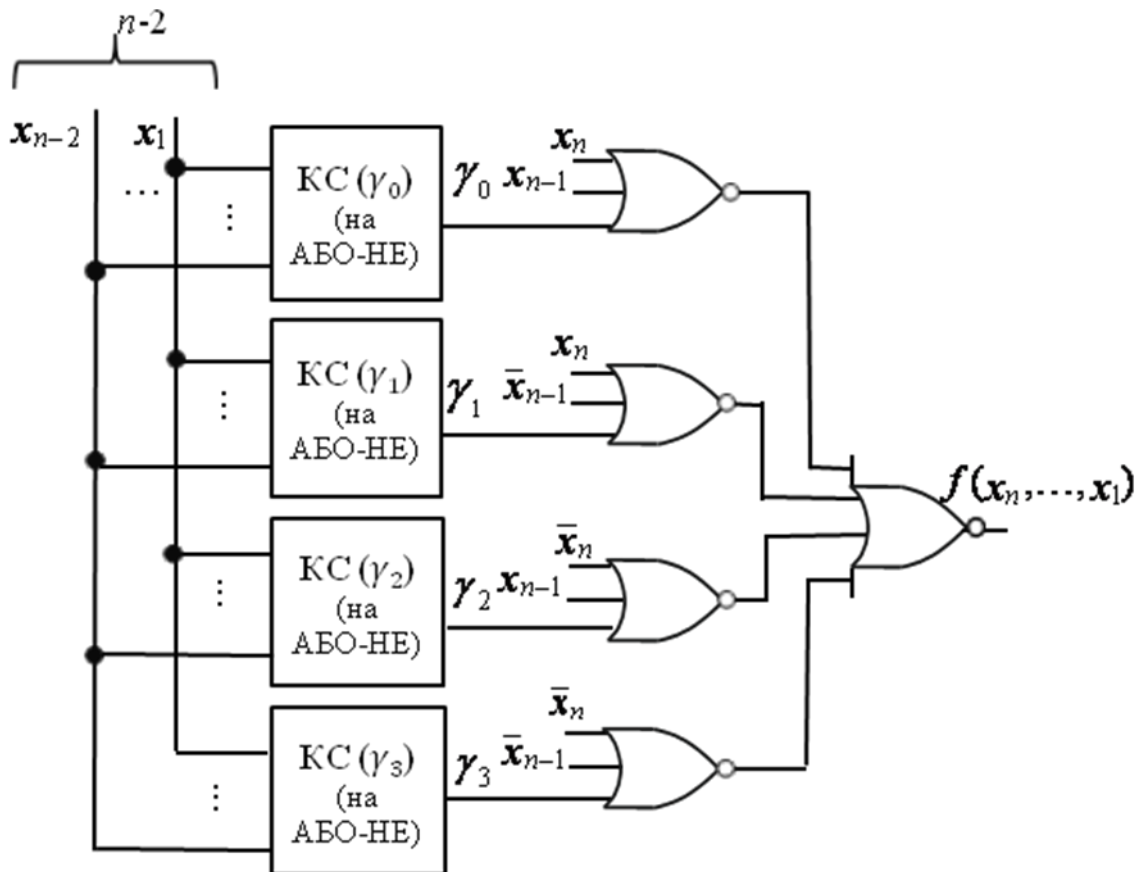


Рис. 2.31. Структура комбінаційної схеми, що реалізує перемикальну функцію $f(x_n, \dots, x_1)$ в елементному базисі алгебри Пірса, при розкладанні функції за змінними x_n, x_{n-1}

Перемикальну функцію можна розкласти за будь-якими двома змінними.

Функцію $f(x_n, \dots, x_1)$ можна розкласти також за будь-якою кількістю змінних.

Наприклад, розкладання функції $f(x_n, \dots, x_1)$ за m ($m < n$) старшими змінними має вигляд:

$$f(x_n, \dots, x_1) = \bigvee_{\vec{\sigma}_m} x_n^{\sigma_n} x_{n-1}^{\sigma_{n-1}} \dots x_{n-m+1}^{\sigma_{n-m+1}} f(\sigma_n, \sigma_{n-1}, \dots, \sigma_{n-m+1}, x_{n-m}, \dots, x_1),$$

$$\text{де } x_i^{\sigma_i} = \begin{cases} \bar{x}_i, & \text{при } \sigma_i = 0, \\ x_i, & \text{при } \sigma_i = 1, \end{cases} \sigma_i \in \{0, 1\}, i = n, \dots, n-m+1,$$

а $\bigvee_{\vec{\sigma}_m}$ – диз'юнкція на наборах $\vec{\sigma}_m = \langle \sigma_n, \sigma_{n-1}, \dots, \sigma_{n-m+1} \rangle$.

Це означає, що функцію $f(x_n, \dots, x_1)$ від n змінних можна розкласти на функції, кожна з яких залежатиме лише від $n-m$ змінних.

Нарешті, функцію (x_n, \dots, x_1) від n змінних можна розкласти за всіма змінними: $f(x_n, \dots, x_1) = \bigvee_{\vec{\sigma}_n} x_n^{\sigma_n} \dots x_1^{\sigma_1} f(\sigma_n, \dots, \sigma_1)$,

де $\bigvee_{\vec{\sigma}_n}$ – диз'юнкція на всіх наборах $\vec{\sigma}_n = \langle \sigma_n, \sigma_{n-1}, \dots, \sigma_1 \rangle$ значень змінних,

$$f(\sigma_n, \dots, \sigma_1) \in \{0, 1\}.$$

Відкинувши в отриманому виразі ті диз'юнктивні члени, для яких $f(\sigma_n, \dots, \sigma_1) = 0$, отримаємо не що інше як ДДНФ функції f :

$$f(x_n, \dots, x_1) = \bigvee_{\vec{\sigma}_n}^* x_n^{\sigma_n} \dots x_1^{\sigma_1},$$

де $\bigvee_{\vec{\sigma}_n}^*$ означає, що диз'юнкцію слід брати лише на тих наборах $\vec{\sigma}_n$, на

яких функція f дорівнює 1, тобто $\bigvee_{\vec{\sigma}_n}^* x_n^{\sigma_n} \dots x_1^{\sigma_1}$ – це ДДНФ (форма І/АБО)

перемикальної функції.

Отже, з теореми Шеннона випливають висновки:

- 1) будь-яку перемикальну функцію можна подати в ДДНФ,
- 2) ДДНФ – це розкладання (декомпозиція) перемикальної функції за всіма змінними.

Розв'яжемо декілька задач.

Задача 2.14.

Дано перемикальну функцію f , задану таблицею істинності. Реалізувати функцію в елементному базисі алгебри Буля, використовуючи декомпозицію функції за змінною x_3 .

x_3	x_2	x_1	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Розв'язування.

Розкладемо функцію f за змінною x_3 :

$$f = \bar{x}_3 f(0, x_2, x_1) \vee x_3 f(1, x_2, x_1) = \bar{x}_3 f_0(x_2, x_1) \vee x_3 f_1(x_2, x_1).$$

Функції f_0 відповідає верхня частина таблиці істинності (без стовпця x_3), а f_1 – нижня частина:

x_2	x_1	f_0
0	0	0
0	1	0
1	0	1
1	1	1

x_2	x_1	f_1
0	0	1
0	1	1
1	0	0
1	1	0

$$f_0 = x_2 \bar{x}_1 \vee x_2 x_1,$$

$$f_1 = \bar{x}_2 \bar{x}_1 \vee \bar{x}_2 x_1.$$

Комбінаційна схема, що реалізує функцію $f(x_3, x_2, x_1)$ від трьох змінних (рис. 2.32), структурно складається з двох простіших комбінаційних схем – $KC(f_0)$, $KC(f_1)$, кожна з яких реалізує перемикальну функцію від двох змінних. ■

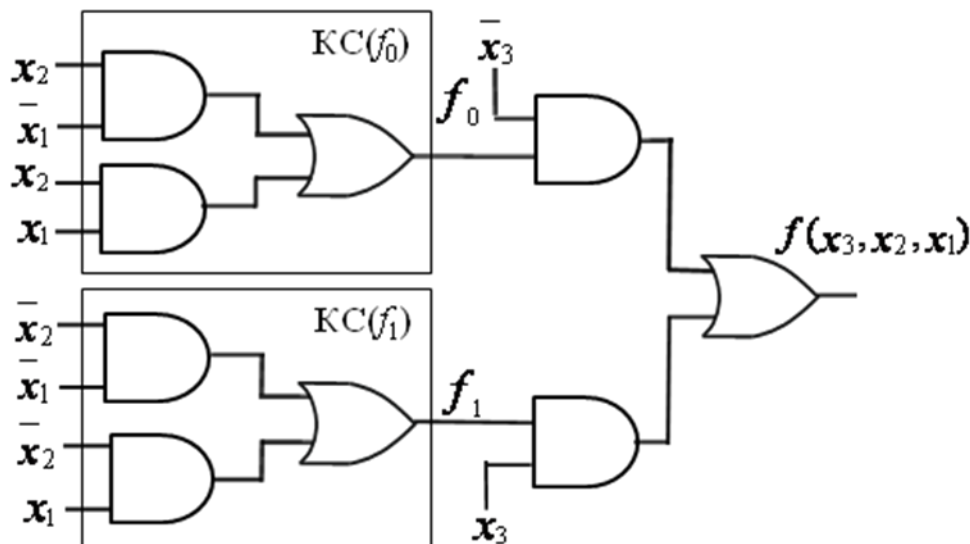


Рис. 2.32. Комбінаційна схема, що реалізує перемикальну функцію $f(x_3, x_2, x_1)$ в елементарному базисі алгебри Буля, при розкладанні функції за змінною x_3

Задача 2.15.

x_3	x_2	x_1	z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Дано перемикальну функцію z від трьох змінних, задану таблицею істинності. Реалізувати функцію в елементному базисі алгебри Шеффера, використовуючи декомпозицію функції за змінною x_2 .

Розв'язування.

Переставимо рядки в таблиці істинності так, щоб у верхній частині таблиці опинились рядки, у яких $x_2 = 0$.

x_3	x_2	x_1	z
0	0	0	0
0	0	1	1
1	0	0	0
1	0	1	1
0	1	0	0
0	1	1	1
1	1	0	1
1	1	1	0

Розкладемо функцію z за змінною x_2 :
 $z = \bar{x}_2 z(x_3, 0, x_1) \vee x_2 z(x_3, 1, x_1) =$
 $= \bar{x}_2 z_0(x_3, x_1) \vee x_2 z_1(x_3, x_1) = \bar{x}_2 z_0 \vee x_2 z_1.$

Функції z_0 відповідає верхня частина модифікованої таблиці істинності, а z_1 – нижня:

x_3	x_1	z_0
0	0	0
0	1	1
1	0	0
1	1	1

x_3	x_1	z_1
0	0	0
0	1	1
1	0	1
1	1	0

Новоотримані функції z_0, z_1 від двох змінних подамо в канонічній формі алгебри Шеффера:

$$z_0 = \bar{x}_3 x_1 \cdot x_3 x_1,$$

$$z_1 = x_3 x_1 \cdot x_3 x_1.$$

Структурно комбінаційна схема, що реалізує функцію z від трьох змінних (рис. 2.33), складається з двох простіших комбінаційних схем – $КС(z_0), КС(z_1)$, кожна з яких реалізує перемикальну функцію від двох змінних. ■

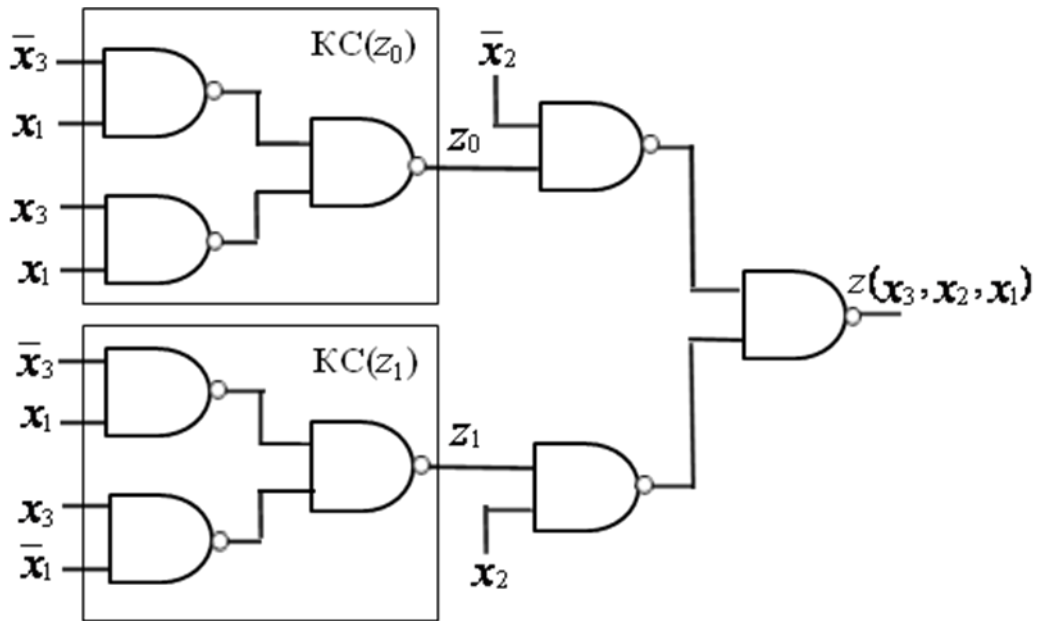


Рис. 2.33. Комбінаційна схема, що реалізує перемикальну функцію $z(x_3, x_2, x_1)$ в елементному базисі алгебри Шеффера, при розкладанні функції за змінною x_2

Задача 2.16.

Дано перемикальну функцію ω від трьох змінних, задану таблицею істинності. Реалізувати функцію в елементному базисі алгебри Пірса, використовуючи декомпозицію функції за змінною x_1 .

x_3	x_2	x_1	ω
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Розв'язування. Переставимо рядки в таблиці істинності функції ω так, щоб у верхній частині таблиці опинились рядки, у яких $x_1 = 0$.

x_3	x_2	x_1	ω
0	0	0	1
0	1	0	0
1	0	0	0
1	1	0	1
0	0	1	0
0	1	1	0
1	0	1	1
1	1	1	1

Розкладемо функцію ω , за змінною x_1 в алгебрі Буля:

$$\begin{aligned} \omega &= \bar{x}_1 \omega(x_3, x_2, 0) \vee x_1 \omega(x_3, x_2, 1) = \\ &= \bar{x}_1 \omega_0(x_3, x_2) \vee x_1 \omega_1(x_3, x_2) = \\ &= \bar{x}_1 \omega_0 \vee x_1 \omega_1. \end{aligned}$$

Отримаємо розкладання функції $\bar{\omega}$ за змінною x_1 в алгебрі Пірса:

$$\bar{\omega} = \bar{x}_1 \bar{\omega}_0 \vee x_1 \bar{\omega}_1,$$

$$\omega = \overline{\bar{x}_1 \bar{\omega}_0 \vee x_1 \bar{\omega}_1} = \overline{\bar{x}_1} \vee \overline{\bar{\omega}_0} \vee \overline{x_1} \vee \overline{\bar{\omega}_1}.$$

Функції ω_0 відповідає верхня частина модифікованої таблиці істинності, а ω_1 – нижня:

x_3	x_2	ω_0
0	0	1
0	1	0
1	0	0
1	1	1

x_3	x_2	ω_1
0	0	0
0	1	0
1	0	1
1	1	1

Новоотримані функції ω_0 , ω_1 від двох змінних подамо в канонічній формі алгебри Пірса:

$$\omega_0 = x_3 \vee x_2 \vee \bar{x}_3 \vee \bar{x}_2,$$

$$\omega_1 = x_3 \vee x_2 \vee x_3 \vee x_2.$$

Комбінаційна схема, що реалізує перемикальну функцію $\omega(x_3, x_2, x_1)$ від трьох змінних (рис. 2.34) складається з двох простих комбінаційних схем – $КС(\omega_0)$, $КС(\omega_1)$, кожна з яких реалізує функцію лише від двох змінних. ■

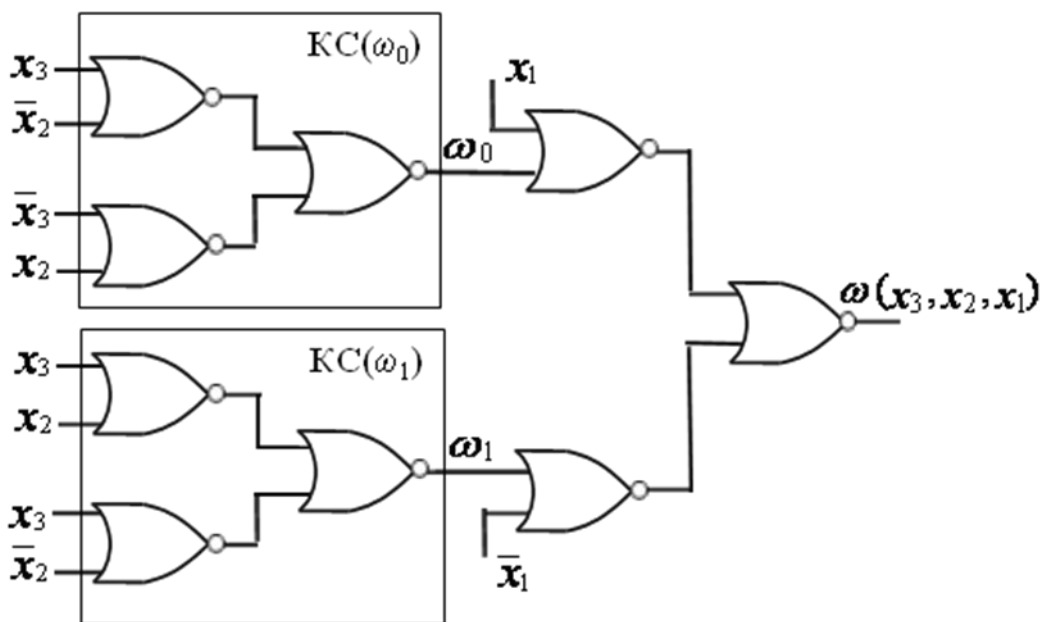


Рис. 2.34. Комбінаційна схема, що реалізує перемикальну функцію $\omega(x_3, x_2, x_1)$ в елементному базисі алгебри Пірса, при розкладанні функції за змінною x_1

Запитання та завдання

1. Дайте визначення декомпозиції перемикальної функції.
2. Чим викликана потреба в декомпозиції (розкладанні) перемикальних функцій?
3. Запишіть аналітичний вираз для розкладання функції $f(a, b)$ від двох змінних за змінною b в алгебрі: а) Буля; б) Шеффера; в) Пірса.
4. Сформулюйте теорему Шеннона.
5. Які два висновки випливають з теореми Шеннона?
6. Запишіть формулу розкладання перемикальної функції $f(x_n, \dots, x_1)$ за змінною: а) x_n ; б) x_i ; в) x_1 .
7. Запишіть формулу розкладання перемикальної функції $y(x_k, \dots, x_1)$ за двома старшими змінними x_k, x_{k-1} .
8. Наведіть структуру комбінаційної схеми, що реалізує перемикальну функцію $z(x_m, \dots, x_1)$ в елементному базисі алгебри: а) Буля; б) Шеффера; в) Пірса, при розкладанні функції за змінною x_m .
9. Наведіть структуру комбінаційної схеми, що реалізує перемикальну функцію $y(x_p, \dots, x_1)$ в елементному базисі алгебри: а) Буля; б) Шеффера; в) Пірса, при розкладанні функції за двома молодшими змінними x_2, x_1 .

Задачі для самостійного розв'язування

1. Записати формулу розкладання функції $f(a, b, c, d)$ за змінною c в алгебрі Шеффера.
2. Записати формулу розкладання функції $y(\alpha, \beta, \gamma)$ за змінною β в алгебрі Пірса.
3. Перемикальна функція $\varphi(x_4, x_3, x_2, x_1)$ дорівнює нулю на наборах 1, 3, 4, 5, 9, 10, 14, 15, на решті наборів – одиниці. Реалізувати функцію на логічних елементах ЗІ-НЕ, використовуючи декомпозицію функції за змінною x_3 .
4. Перемикальна функція $\delta(x_4, x_3, x_2, x_1)$ дорівнює одиниці на наборах 1, 2, 4, 7, 8, 9, 14, 15, на решті наборів – нулю. Реалізувати функцію на логічних елементах ЗАБО-НЕ, використовуючи декомпозицію функції за змінною x_2 .

2.6. Розширена алгебра Буля-Шеффера-Пірса

На практиці при проектуванні логічних схем найчастіше використовують логічні елементи І, АБО, НЕ, І-НЕ, АБО-НЕ. Така

множина логічних елементів є об'єднаним елементним базисом трьох алгебр – Буля, Шеффера, Пірса. Відповідно, з практичних міркувань доцільно розглядати три перелічені алгебри як єдину, сукупну алгебру з розширеним елементним базисом. Це дає можливість при проектуванні логічних схем розширити пошук таких аналітичних форм функцій в об'єднаній алгебрі, які б дозволяли отримувати схеми з поліпшеними показниками апаратної складності та швидкодії.

Розширена алгебра Буля-Шеффера-Пірса дозволяє подати будь-яку перемикальну функцію у 8-ми нормальних формах, які називають канонічними. Нормальні форми подання перемикальних функцій дозволяють будувати дворівневі (двокаскадні) логічні схеми, якщо не накладаються обмеження на кількість входів логічних елементів.

Оскільки в будь-якій нормальній формі подання перемикальної функції використовуються дві операції – внутрішня і зовнішня, то відповідній нормальній формі дають назву у вигляді: внутрішня_операція/зовнішня_операція.

Наприклад, ДДНФ перемикальної функції має й іншу назву – форма І/АБО, а ДКНФ – форма АБО/І.

Нехай перемикальна функція у задана таблицею істинності (табл. 2.7).

Таблиця 2.7

Таблиця істинності перемикальної функції у

x_2	x_1	у	\bar{y}
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

Отримаємо можливі нормальні форми подання функції у. Спочатку запишемо ДДНФ функції у:

$$y_{\text{ДДНФ}} = \bar{x}_2 x_1 \vee x_2 \bar{x}_1.$$

Виходячи з ДДНФ, беручи до уваги аксіому $\bar{\bar{x}} = x$ та використовуючи правила де Моргана, послідовно отримаємо:

$$\begin{aligned} y &= \bar{x}_2 x_1 \vee x_2 \bar{x}_1 = && \text{(І/АБО)} \\ &= \overline{\overline{x_2 x_1} \wedge \overline{x_2 x_1}} = \\ &= \overline{x_2 x_1 \wedge x_2 x_1} = && \text{(І-НЕ/І-НЕ)} \\ &= \overline{(x_2 \vee x_1)(x_2 \vee x_1)} = && \text{(АБО/І-НЕ)} \\ &= \overline{x_2 \vee x_1} \vee \overline{x_2 \vee x_1} = && \text{(АБО-НЕ/АБО)} \end{aligned}$$

Якщо до отриманого виразу ще раз застосувати правило де Моргана, то дістанемо вихідну форму – І/АБО, тобто ДДНФ функції.

Таким чином, виходячи з ДДНФ, можна отримати 4 дворівневі (нормальні) форми подання перемикальної функції.

Тепер запишемо ДКНФ функції y :

$$y_{\text{ДКНФ}} = (x_2 \vee x_1)(\bar{x}_2 \vee \bar{x}_1).$$

Виходячи з цієї форми функції, беручи до уваги аксіому $\bar{\bar{x}} = x$ та застосовуючи правила де Моргана, послідовно отримаємо наступні форми:

$$\begin{aligned} y &= (x_2 \vee x_1)(\bar{x}_2 \vee \bar{x}_1) = && \text{(АБО/І)} \\ &= \overline{\overline{(x_2 \vee x_1)(\bar{x}_2 \vee \bar{x}_1)}} = \\ &= \overline{x_2 \vee x_1 \vee \bar{x}_2 \vee \bar{x}_1} = && \text{(АБО-НЕ/АБО-НЕ)} \\ &= \overline{x_2 x_1 \vee \bar{x}_2 \bar{x}_1} = && \text{(І/АБО-НЕ)} \\ &= \overline{x_2 x_1} \cdot \overline{\bar{x}_2 \bar{x}_1} = && \text{(І-НЕ/І)} \end{aligned}$$

Застосувавши до отриманого виразу правило де Моргана, дістанемо вихідну форму – І/АБО, тобто ДКНФ функції.

Отже, виходячи з ДКНФ, також можна отримати 4 нормальні (дворівневі) форми подання функції.

Таким чином, в розширеній алгебрі Буля-Шеффера-Пірса можна утворити 8 нормальних форм, які й називають канонічними (рис. 2.35). З них: форма І-НЕ/І-НЕ є канонічною формою подання перемикальних функцій в алгебрі Шеффера, форма АБО-НЕ/АБО-НЕ – в алгебрі Пірса, а форми І/АБО (ДДНФ) та АБО/І (ДКНФ) – канонічними формами в алгебрі Буля.

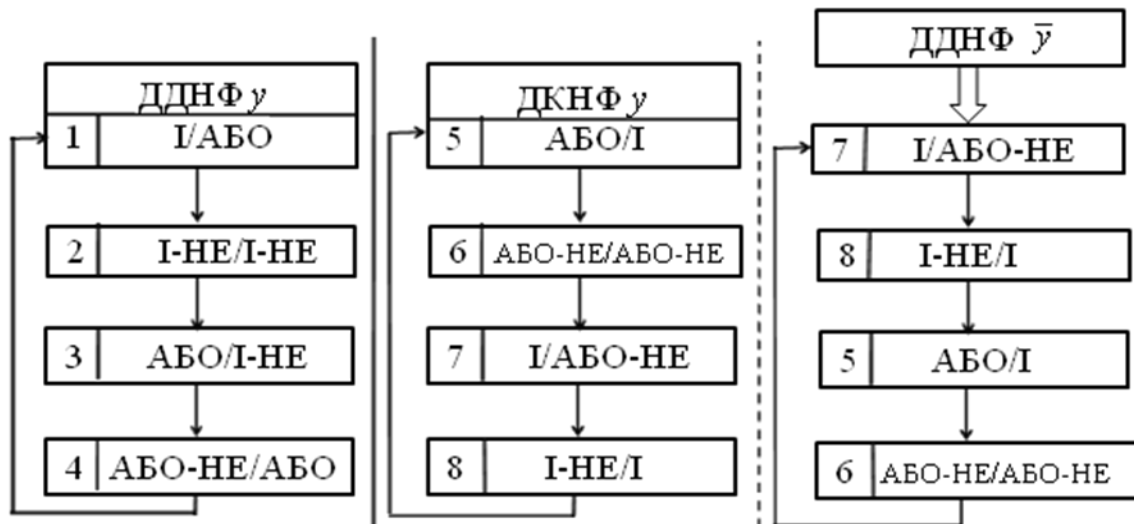


Рис. 2.35. Канонічні форми подання перемикальних функцій в розширеній алгебрі Буля-Шеффера-Пірса

Форми 5 – 8 можна отримати також виходячи з ДДНФ заперечення функції.

Запишемо ДДНФ функції \bar{y} (див. табл. 2.7):

$$\bar{y}_{\text{ДДНФ}} = \bar{x}_2 \bar{x}_1 \vee x_2 x_1.$$

Якщо поставити заперечення над лівою та правою частинами виразу, то послідовно отримаємо:

$$\begin{aligned} y &= \overline{\bar{x}_2 \bar{x}_1 \vee x_2 x_1} = && \text{(I/АБО-НЕ)} \\ &= \overline{\bar{x}_2 \bar{x}_1} \cdot \overline{x_2 x_1} = && \text{(I-НЕ/I)} \\ &= (x_2 \vee x_1)(\bar{x}_2 \vee \bar{x}_1) = && \text{(АБО/I)} \\ &= \overline{(x_2 \vee x_1)(\bar{x}_2 \vee \bar{x}_1)} = && \\ &= \overline{x_2 \vee x_1} \vee \overline{\bar{x}_2 \vee \bar{x}_1} = && \text{(АБО-НЕ/АБО-НЕ)} \end{aligned}$$

Застосовуючи ще раз правило де Моргана отримаємо початкову форму – I/АБО-НЕ. Отже, форми 5 – 8 подання перемикальної функції можна отримати двома способами – виходячи або з ДКНФ функції, або з ДДНФ заперечення (інверсії) функції.

Якщо задано підмножину логічних елементів з елементного базису розширеної алгебри Буля-Шеффера-Пірса, на основі яких необхідно спроектувати комбінаційну схему, що реалізує задану перемикальну функцію, то спочатку функцію подають у 8-ми канонічних формах, з яких вибирають ті форми-кандидати, які відповідають заданій підмножині логічних елементів, а потім внаслідок побудови та дослідження комбінаційних схем, що реалізують відповідні форми-кандидати, остаточно вибирають ту форму перемикальної функції, яка якнайповніше задовольняє цільову функцію проектування – мінімальну складність або/і максимальну швидкодію комбінаційної схеми.

Наприклад, якщо при побудові комбінаційних схем використовують логічні елементи I, АБО, I-НЕ, то при апаратній реалізації заданої перемикальної функції з восьми можливих канонічних форм функції слід вибрати 5 канонічних форм-кандидатів: I/АБО, I-НЕ/I-НЕ, АБО/I-НЕ, АБО/I, I-НЕ/I (див. рис. 2.35). Для кожної з цих форм необхідно побудувати комбінаційну схему та оцінити її за показником апаратної складності та швидкодії. Потім слід порівняти між собою 5 отриманих комбінаційних схем за заданою цільовою функцією проектування та вибрати комбінаційну схему з кращими показниками.

Задача 2.17.

Дано елементний базис – логічні елементи I, I-НЕ. Побудувати комбінаційну схему мінімальної складності, яка реалізує перемикальну функцію z , задану таблицею істинності (табл. 2.8).

Таблиця 2.8
Таблиця істинності
перемикальної функції z

x_3	x_2	x_1	z
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Розв'язування.

Запишемо ДДНФ функції z :

$$z_{\text{ДДНФ}} = \bar{x}_2 \bar{x}_2 \bar{x}_1 \vee \bar{x}_2 \bar{x}_2 x_1 \vee x_2 \bar{x}_2 \bar{x}_1 \vee x_2 x_2 \bar{x}_1 \vee x_2 x_2 x_1 \quad (\text{форма I/АБО}).$$

Послідовно отримуємо можливі канонічні форми подання функції z , виходячи з ДДНФ:

$$\begin{aligned} z &= \overline{\overline{x_3 x_2 x_1} \vee \overline{x_3 x_2 x_1} \vee \overline{x_3 x_2 x_1} \vee \overline{x_3 x_2 x_1} \vee \overline{x_3 x_2 x_1}} = \\ &= \overline{\overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1}} = \quad (\text{форма I-НЕ/I-НЕ}) \end{aligned}$$

$$\begin{aligned} &= (\overline{x_3 \vee x_2 \vee x_1})(\overline{x_3 \vee x_2 \vee x_1})(\overline{x_3 \vee x_2 \vee x_1})(\overline{x_3 \vee x_2 \vee x_1})(\overline{x_3 \vee x_2 \vee x_1}) = \\ & \quad (\text{форма АБО/I-НЕ}) \end{aligned}$$

$$\begin{aligned} &= \overline{\overline{x_3 \vee x_2 \vee x_1} \vee \overline{x_3 \vee x_2 \vee x_1} \vee \overline{x_3 \vee x_2 \vee x_1} \vee \overline{x_3 \vee x_2 \vee x_1} \vee \overline{x_3 \vee x_2 \vee x_1}} = \\ & \quad (\text{форма АБО-НЕ/АБО}) \end{aligned}$$

Запишемо ДКНФ функції z :

$$z_{\text{ДКНФ}} = (x_3 \vee \bar{x}_2 \vee x_1)(x_3 \vee \bar{x}_2 \vee \bar{x}_1)(\bar{x}_3 \vee x_2 \vee \bar{x}_1) \quad (\text{форма I/АБО-НЕ}).$$

Послідовно отримуємо решту канонічних форм подання функції z , виходячи з ДКНФ:

$$\begin{aligned} z &= \overline{\overline{(x_3 \vee x_2 \vee x_1)(x_3 \vee x_2 \vee \bar{x}_1)(\bar{x}_3 \vee x_2 \vee \bar{x}_1)}} = \\ &= \overline{\overline{x_3 \vee x_2 \vee x_1} \vee \overline{x_3 \vee x_2 \vee \bar{x}_1} \vee \overline{x_3 \vee x_2 \vee \bar{x}_1} \vee \overline{x_3 \vee x_2 \vee \bar{x}_1}} = \quad (\text{форма АБО-НЕ/АБО-НЕ}) \end{aligned}$$

$$= \overline{\overline{x_3 x_2 x_1} \vee \overline{x_3 x_2 x_1} \vee \overline{x_3 x_2 x_1}} = \quad (\text{форма АБО/I})$$

$$= \overline{\overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1}} \quad (\text{форма I-НЕ/I})$$

Формами-кандидатами для елементного базису I, I-НЕ є форми: I-НЕ/I та I-НЕ/I-НЕ.

Побудуємо комбінаційну схему (рис. 2.36а), що реалізує форму I-НЕ/I-НЕ функції z , тобто

$$z = \overline{\overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1}}$$

Складність комбінаційної схеми за Квайном: $K = 20$.

Побудуємо комбінаційну схему (рис. 2.36б), що реалізує форму I-НЕ/I функції z , тобто

$$z = \overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1}$$

Складність цієї комбінаційної схеми за Квайном становить $K = 12$.

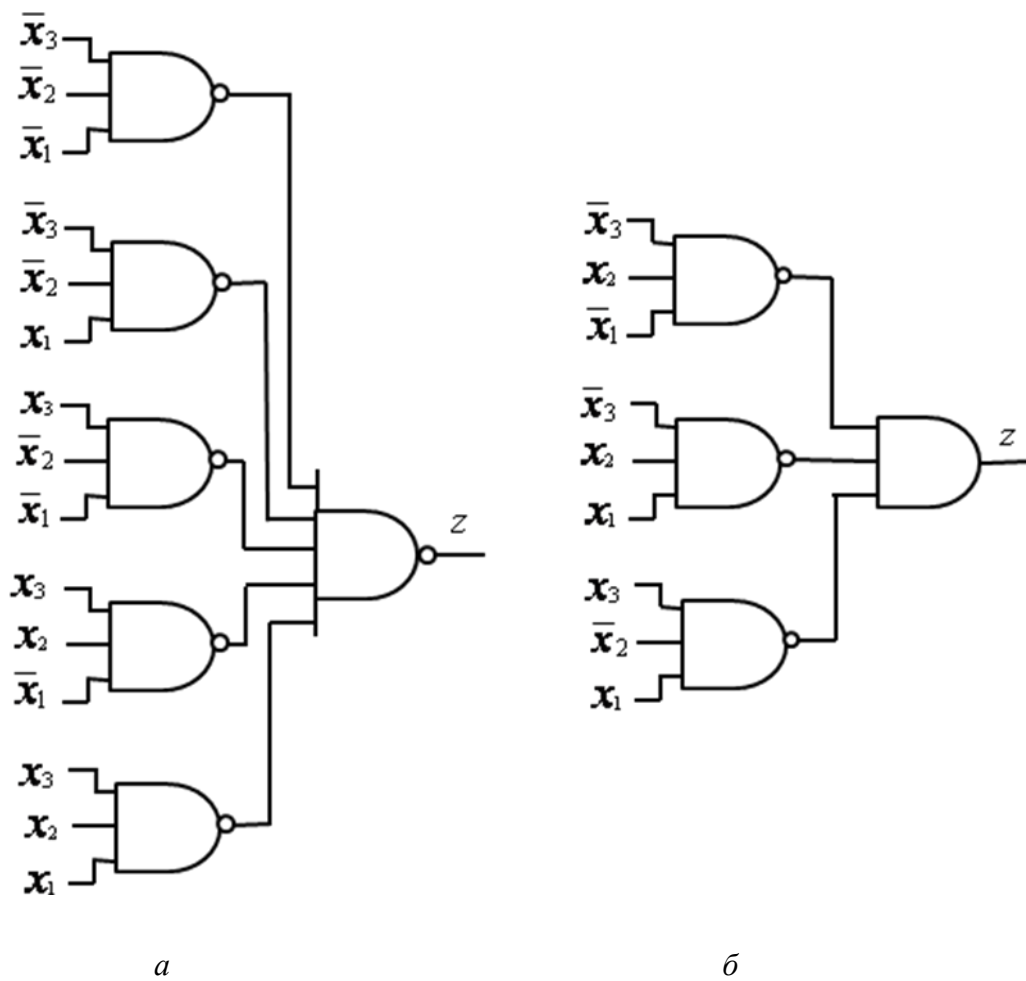


Рис. 2.36. Комбінаційні схеми, що реалізують функцію z , задану таблицею істинності (табл. 2.8), при поданні функції у формі: a – I-НЕ/I-НЕ; b – I-НЕ/I

Оскільки комбінаційна схема на рис. 2.36б має меншу апаратну складність, то перевагу слід віддати формі І-НЕ/І функції z :

$$z = \overline{\overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1}},$$

яка й забезпечує побудову схеми з мінімальною складністю. ■

Канонічні форми подання перемикальних функцій в розширеній алгебрі Буля-Шеффера-Пірса можна прямо застосовувати для побудови комбінаційних схем з оптимізованими параметрами (складність, швидкодія) лише в тому випадку, якщо не накладаються обмеження на кількість входів логічних елементів.

Якщо кількість входів логічних елементів, на основі яких проектують схему, менша за ранг термів, що входять до складу канонічних форм функції, то вибрані форми-кандидати слід перетворити в операторні форми так, щоб враховувалась кількість входів логічних елементів, і лише після цього будувати досліджувані комбінаційні схеми та оцінювати їх показники.

Операторні форми подання функції у загальному випадку не є дворівневими (нормальними).

Якщо p – кількість входів логічних елементів, що використовуються для побудови комбінаційної схеми, а n – місність функції, і $p < n$, то змінні в термах канонічної форми об'єднують у групи так, щоб до кожної групи входило не більше p змінних, застосовуючи при цьому такі співвідношення:

$$x_1 \cdot x_2 \cdot \dots \cdot x_n = (x_1 \cdot \dots \cdot x_g) \cdot \dots \cdot (x_s \cdot \dots \cdot x_n),$$

$$x_1 \vee x_2 \vee \dots \vee x_n = (x_1 \vee \dots \vee x_g) \vee \dots \vee (x_s \vee \dots \vee x_n),$$

$$\overline{\overline{x_1 \cdot x_2 \cdot \dots \cdot x_n}} = \overline{\overline{x_1 \cdot \dots \cdot x_g} \cdot \overline{\overline{x_s \cdot \dots \cdot x_n}}},$$

$$\overline{\overline{x_1 \vee x_2 \vee \dots \vee x_n}} = \overline{\overline{x_1 \vee \dots \vee x_g} \vee \overline{\overline{x_s \vee \dots \vee x_n}}},$$

де $g \leq p$ і $n - s + 1 \leq p$.

Кількість груп змінних також не має перевищувати p , інакше вказані перетворення виконують стосовно груп змінних.

Нехай функцію y , подану в канонічній формі І-НЕ/І-НЕ

$$y = \overline{\overline{\overline{x_3 x_2 x_1} \cdot \overline{\overline{x_3 x_2 x_1}} \cdot \overline{\overline{x_3 x_2 x_1}}}},$$

необхідно реалізувати на логічних елементах 2І-НЕ. Тоді її слід перетворити в таку операторну форму:

$$y_{\text{оп.ф}} = \overline{\overline{\overline{x_3 x_2 x_1} \cdot \overline{\overline{x_3 x_2 x_1}} \cdot \overline{\overline{x_3 x_2 x_1}}}}.$$

Якщо необхідно на логічних елементах 2АБО-НЕ, 2АБО побудувати комбінаційну схему, що реалізує функцію z , задану в канонічній формі АБО-НЕ/АБО

$$z = \overline{x_3 \vee x_2 \vee x_1} \vee \overline{x_3 \vee x_2 \vee x_1} \vee \overline{x_3 \vee x_2 \vee x_1} \vee \overline{x_3 \vee x_2 \vee x_1},$$

то придатною для реалізації є наступна операторна форма функції:

$$z_{\text{оп.ф.}} = ((\overline{x_3 \vee x_2}) \vee x_1 \vee (x_3 \vee \overline{x_2}) \vee x_1) \vee ((\overline{x_3 \vee x_2}) \vee x_1 \vee (x_3 \vee x_2) \vee \overline{x_1}).$$

Розширена алгебра Буля-Шеффера-Пірса дає проектувальнику під час побудови комбінаційної схеми можливість у межах об'єднаного елементного базису трьох алгебр – логічних елементів І, АБО, НЕ, І-НЕ, АБО-НЕ, замінювати одну сукупність логічних елементів на іншу (що відповідає заміні однієї канонічної форми функції на іншу), досягаючи бажаної мети проектування.

Наприклад, якщо комбінаційна схема побудована з використанням логічних елементів АБО-НЕ, І, а з певних причин є необхідність перейти на використання логічних елементів І-НЕ, АБО, то проектувальнику необхідно (див. рис. 2.35):

- 1) звести канонічну форму І/АБО-НЕ, яка відповідає сукупності логічних елементів АБО-НЕ, І, до ДКНФ функції;
- 2) за відомою ДКНФ подати функцію в ДДНФ;
- 3) перетворити ДДНФ у форму АБО/І-НЕ, яка відповідає сукупності елементів І-НЕ, АБО, та побудувати нову комбінаційну схему, яка буде еквівалентна заданій.

Запитання та завдання

1. Чому на практиці використовують розширену алгебру Буля-Шеффера-Пірса?
2. У скількох канонічних формах можна подати перемикальну функцію в розширеній алгебрі Буля-Шеффера-Пірса?
3. Яку структуру має комбінаційна схема, що реалізує канонічну форму перемикальної функції?
4. Нехай дано елементний базис – логічні елементи АБО, АБО-НЕ, та деяку перемикальну функцію, задану таблицею істинності. Якою має бути стратегія побудови комбінаційної схеми, що реалізує задану перемикальну функцію, якщо цільовою функцією проектування є максимально можлива швидкодія?
5. Дано елементний базис – логічні елементи І, І-НЕ, АБО-НЕ, а також деяку перемикальну функцію. Сформууйте список канонічних форм-кандидатів для побудови оптимізованої комбінаційної схеми, що

реалізує задану перемикальну функцію, відповідно до цільової функції проектування.

6. За якої умови канонічні форми розширеної алгебри Буля-Шеффера-Пірса можна прямо застосовувати для побудови комбінаційних схем з оптимізованими параметрами?
7. Як необхідно перетворити обрані для побудови оптимальної комбінаційної схеми канонічні форми-кандидати, якщо кількість входів логічних елементів, на основі яких здійснюється проектування комбінаційної схеми, менша за ранг термів, що входять до складу канонічних форм?
8. Яка відмінність між канонічною та операторною формою подання перемикальної функції?

Задачі для самостійного розв'язування

1. Повністю визначена перемикальна функція $f(x_3, x_2, x_1)$ дорівнює одиниці на наборах 0, 1, 5, 6, а на решті наборів – нулю. Подати функцію f у 8-ми канонічних формах розширеної алгебри Буля-Шеффера-Пірса .
2. Повністю визначена перемикальна функція $f(x_3, x_2, x_1)$ дорівнює нулю на наборах 1, 2, 4, 6, а на решті наборів – одиниці. Подати функцію f у 8-ми канонічних формах розширеної алгебри Буля-Шеффера-Пірса .
3. Перемикальну функцію $\alpha(x_3, x_2, x_1)$ подано в канонічній формі алгебри Шеффера:

$$\alpha = \overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1}$$

Подати функцію α у решті 7-ми канонічних формах об'єднаної алгебри Буля-Шеффера-Пірса.

4. Перемикальну функцію $\beta(a, b, c)$ подано в канонічній формі алгебри Пірса:

$$\beta = \overline{a \vee b \vee c} \vee \overline{a \vee b \vee c} \vee \overline{a \vee b \vee c} \vee \overline{a \vee b \vee c}$$

Подати функцію β у решті 7-ми канонічних формах об'єднаної алгебри Буля-Шеффера-Пірса .

5. Повністю визначену перемикальну функцію задано в канонічній формі І-НЕ/І-НЕ:

$$\delta = \overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1}$$

Подати функцію в операторній формі, придатній для реалізації в елементному базисі 2І, 2І-НЕ, за умови, що при побудові

комбінаційної схеми мають бути використані обидва типи заданих логічних елементів.

6. Повністю визначену перемикальну функцію γ задано в канонічній формі АБО-НЕ/АБО-НЕ:

$$\gamma = \overline{\overline{x_3 \vee \overline{x_2} \vee x_1} \vee \overline{\overline{x_3} \vee x_2 \vee \overline{x_1}} \vee \overline{\overline{x_3} \vee \overline{x_2} \vee x_1}}$$

Подати функцію γ в операторній формі, придатній для реалізації в елементному базисі 2АБО, 3І-НЕ, за умови, що при побудові комбінаційної схеми мають бути використані обидва типи заданих логічних елементів.

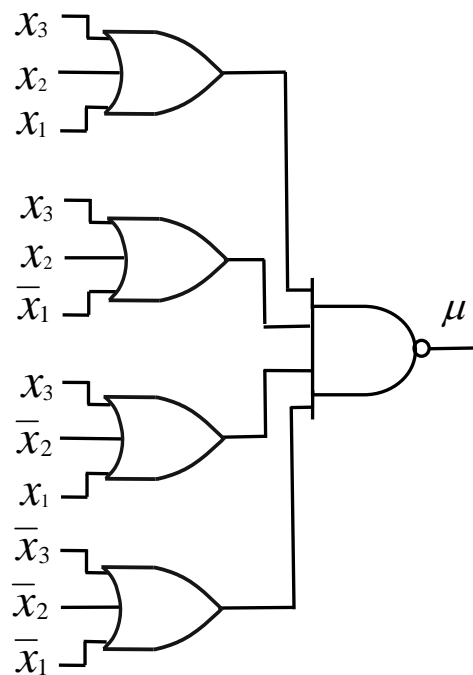
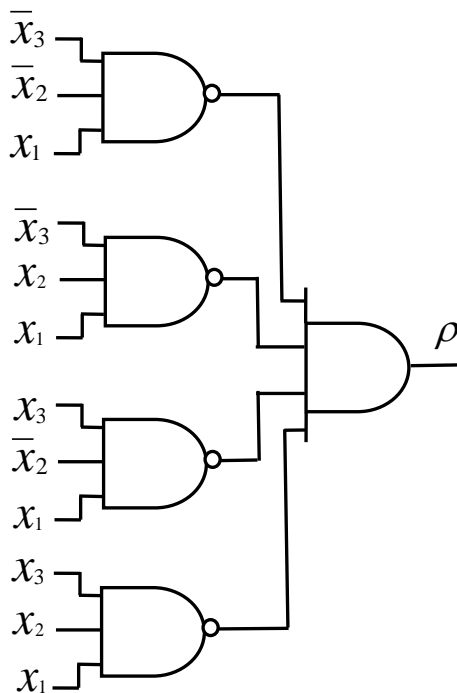
7. Дано повністю визначену перемикальну функцію $\omega(x_3, x_2, x_1)$ у формі: $\omega = \overline{x_3} \overline{x_2} \overline{x_1} \vee \overline{x_3} x_2 \overline{x_1} \vee x_3 \overline{x_2} \overline{x_1} \vee x_3 x_2 x_1$.

Подати функцію ω в канонічній формі АБО/І-НЕ.

8. Повністю визначена перемикальна функція u від трьох змінних дорівнює одиниці на наборах 1, 3, 4, 6, на решті наборів – нулю. Побудувати оптимальну комбінаційну схему за критерієм максимуму швидкодії в елементному базисі 2АБО, 3АБО-НЕ, якщо $\tau_{2АБО} = 20$ нс, а $\tau_{3АБО-НЕ} = 22$ нс.

9. Повністю визначена перемикальна функція f від трьох змінних дорівнює нулю на наборах 0, 2, 3, 6, на решті наборів – одиниці. Побудувати оптимальну комбінаційну схему за критерієм мінімуму апаратної складності (показник Квайна) на основі логічних елементів 3АБО, 2І-НЕ.

10. Дано комбінаційну схему, що реалізує перемикальну функцію ρ :



Побудувати еквівалентну комбінаційну схему на логічних елементах ЗАБО, 2АБО-НЕ.

11. Дано комбінаційну схему, що реалізує перемикальну функцію μ . Побудувати еквівалентну комбінаційну схему на логічних елементах ЗАБО-НЕ. Порівняти комбінаційні схеми за апаратною складністю.

2.7. Принципи двоїстості в теорії перемикальних функцій

У теорії перемикальних функцій важливими є поняття двоїстої функції, а також принцип двоїстості (дуальності).

Спочатку дамо визначення двоїстої перемикальної функції. Перемикальна функція $\Psi(x_n, \dots, x_1)$ є двоїстою (дуальною) до функції $f(x_n, \dots, x_1)$, якщо

$$\Psi(x_n, \dots, x_1) = \overline{f(\overline{x}_n, \dots, \overline{x}_1)}.$$

Якщо перемикальна функція f задана аналітично, то для знаходження Ψ – двоїстої до f функції, в формулі для f необхідно, не змінюючи знаків операцій, поставити заперечення над змінними, а потім над новоотриманим виразом поставити загальне заперечення.

Нехай $f(x_2, x_1) = x_2 \vee x_1$. Двоїстою до функції f є функція

$$\Psi = \overline{f(\overline{x}_2, \overline{x}_1)} = \overline{\overline{x}_2 \vee \overline{x}_1} = x_2 \cdot x_1.$$

Тобто, функція кон'юнкції є двоїстою до функції диз'юнкції.

Нехай $f(x_2, x_1) = x_2 \cdot x_1$. Двоїстою до функції f є функція

$$\Psi = \overline{f(\overline{x}_2, \overline{x}_1)} = \overline{\overline{x}_2 \cdot \overline{x}_1} = x_2 \vee x_1.$$

Тобто, функція диз'юнкції є двоїстою до функції кон'юнкції.

Як бачимо, кон'юнкція та диз'юнкція є взаємно двоїстими перемикальними функціями.

Властивість *взаємної двоїстості* справедлива для будь-якої пари двоїстих функцій: якщо перемикальна функція Ψ є двоїстою до функції f , то функція f є двоїстою до функції Ψ .

Якщо перемикальна функція f задана таблицею істинності, то для знаходження Ψ – двоїстої до f функції, необхідно стовпець $f(x_n, \dots, x_1)$ записати у зворотному порядку (знизу вгору) та проінвертувати значення в новоотриманому стовпці (замінити 0 на 1, а 1 на 0) (табл. 2.9).

Запис стовпця у зворотному порядку еквівалентний інвертуванню змінних, наприклад, для набору 111: $\overline{1} \overline{1} \overline{1} \rightarrow 000$; для набору 110: $\overline{1} \overline{1} \overline{0} \rightarrow 001$.

Таблиця 2.9

Приклад знаходження перемикальної функції
 Ψ – двоїстої до функції $f(x_3, x_2, x_1)$

Можливі набори значень аргументів			Задана функція $f(x_3, x_2, x_1)$	Стовпець f у зворотному порядку, $f^{3\bar{6}} = f(\bar{x}_3, \bar{x}_2, \bar{x}_1)$	Функція Ψ – двоїста до f , $\psi = \bar{f}^{3\bar{6}} = \bar{f}(\bar{x}_3, \bar{x}_2, \bar{x}_1)$
x_3	x_2	x_1			
0	0	0	0	1	0
0	0	1	1	1	0
0	1	0	1	1	0
0	1	1	0	0	1
1	0	0	0	0	1
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	0	1

Знайдемо двоїсті перемикальні функції серед 1-місних функцій:

<i>Константа нуль</i> $f_0(x) = 0$				<i>Повторення</i> $f_1(x) = x$				<i>Заперечення</i> $f_2(x) = \bar{x}$				<i>Константа одиниця</i> $f_3(x) = 1$			
x	f_0	$f_0^{3\bar{6}}$	Ψ_0	x	f_1	$f_1^{3\bar{6}}$	Ψ_1	x	f_2	$f_2^{3\bar{6}}$	Ψ_2	x	f_3	$f_3^{3\bar{6}}$	Ψ_3
0	0	0	1	0	0	1	0	0	1	0	1	0	1	0	1
1	0	0	1	1	1	0	1	1	0	1	0	1	1	0	1
$\Psi_0(x) = 1 = f_3(x)$ (Константа одиниця)				$\Psi_1(x) = x = f_1(x)$ (Повторення)				$\Psi_2(x) = \bar{x} = f_2(x)$ (Заперечення)				$\Psi_3(x) = 0 = f_0(x)$ (Константа нуль)			

Отже, серед одномісних є такі пари двоїстих функцій: Константа нуль – Константа одиниця, Повторення – Повторення, Заперечення – Заперечення, Константа одиниця – Константа нуль.

Таким чином:

- 1) перемикальна функція *Константа нуль* і перемикальна функція *Константа одиниця* є взаємно двоїстими функціями,
- 2) перемикальна функція *Повторення*, $f_1(x) = x$, є двоїстою до самої себе (самодвоїстою),
- 3) перемикальна функція *Заперечення*, $f_2(x) = \bar{x}$, є двоїстою до самої себе (самодвоїстою).

Знайдемо двоїсті перемикальні функції серед 2-місних функцій:

Константа нуль

$$F_0(x_2, x_1) = 0$$

x_2	x_1	F_0	F_0^{3B}	ψ_0
0	0	0	0	1
0	1	0	0	1
1	0	0	0	1
1	1	0	0	1

$$\psi_0 = 1 = F_{15}$$

(Константа одиниця)

Кон'юнкція

$$F_1(x_2, x_1) = x_2x_1$$

x_2	x_1	F_1	F_1^{3B}	ψ_1
0	0	0	1	0
0	1	0	0	1
1	0	0	0	1
1	1	1	0	1

$$\psi_1 = x_2 \vee x_1 = F_7$$

(Диз'юнкція)

Інверсія імплікації

$$F_2(x_2, x_1) = \overline{x_2 \rightarrow x_1}$$

x_2	x_1	F_2	F_2^{3B}	ψ_2
0	0	0	0	1
0	1	0	1	0
1	0	1	0	1
1	1	0	0	1

$$\psi_2 = x_2 \rightarrow x_1 = F_{11}$$

(Зворотна імплікація)

Інверсія зворотної імплікації

$$F_4(x_2, x_1) = \overline{x_2 \leftarrow x_1}$$

x_2	x_1	F_4	F_4^{3B}	ψ_4
0	0	0	0	1
0	1	1	0	1
1	0	0	1	0
1	1	0	0	1

$$\psi_4 = x_2 \rightarrow x_1 = F_{13}$$

(Імплікація)

Сума за mod 2

$$F_6(x_2, x_1) = x_2 \oplus x_1$$

x_2	x_1	F_6	F_6^{3B}	ψ_6
0	0	0	0	1
0	1	1	1	0
1	0	1	1	0
1	1	0	0	1

$$\psi_6 = \overline{x_2 \oplus x_1} = F_9$$

(Заперечення суми за mod 2)

АБО-НЕ

$$F_8(x_2, x_1) = \overline{x_2 \vee x_1}$$

x_2	x_1	F_8	F_8^{3B}	ψ_8
0	0	1	0	1
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0

$$\psi_8 = \overline{x_2 x_1} = F_{14}$$

(І-НЕ)

Серед двомісних перемикальних функцій взаємно двоїстими є 6 пар перемикальних функцій (табл. 2.10):

- функція $F_0 = 0$ (Константа нуль) і функція $F_{15} = 1$ (Константа одиниця),
- функція $F_1 = x_2x_1$ (кон'юнкція) і функція $F_7 = x_2 \vee x_1$ (диз'юнкція),

- функція $F_2 = \overline{x_2 \rightarrow x_1}$ (інверсія імплікації) і функція $F_{11} = x_2 \leftarrow x_1$ (зворотна імплікація),
- функція $F_4 = \overline{x_2 \leftarrow x_1}$ (інверсія зворотної імплікації) і функція $F_{13} = x_2 \rightarrow x_1$ (імплікація),
- функція $F_6 = x_2 \oplus x_1$ (сума за модулем два) і функція $F_9 = \overline{x_2 \oplus x_1}$ (заперечення суми за модулем два),
- функція $F_8 = \overline{x_2 \vee x_1}$ (функція Пірса) і функція $F_{14} = \overline{x_2 x_1}$ (функція Шеффера).

Таблиця 2.10

Взаємно двоїсті функції серед елементарних перемикальних функцій

Функція (Двоїста функція)	Аналітичний запис	Двоїста функція (Функція)
Константа нуль	$0 \leftrightarrow 1$	Константа одиниця
Кон'юнкція	$x_2 \cdot x_1 \leftrightarrow x_2 \vee x_1$	Диз'юнкція
Інверсія імплікації (Заборона за x_1)	$x_2 \cdot \bar{x}_1 \leftrightarrow x_2 \vee \bar{x}_1$	Зворотна імплікація (Від x_1 до x_2)
Інверсія зворотної імплікації (Заборона за x_2)	$\bar{x}_2 \cdot x_1 \leftrightarrow \bar{x}_2 \vee x_1$	Імплікація (Від x_2 до x_1)
Сума за модулем два	$x_2 \oplus x_1 \leftrightarrow \overline{x_2 \oplus x_1}$	Заперечення суми за модулем два
Стрілка Пірса (функція АБО-НЕ)	$\overline{x_2 \vee x_1} \leftrightarrow \overline{x_2 \cdot x_1}$	Штрих Шеффера (функція І-НЕ)

На основі поняття двоїстості функцій можна сформулювати *принцип двоїстості* в теорії перемикальних функцій: якщо формула $\Phi = F(\varphi_1, \varphi_2, \dots, \varphi_s)$ реалізує перемикальну функцію $f(x_n, \dots, x_1)$, то формула $\Phi^* = F(\varphi_1^*, \varphi_2^*, \dots, \varphi_s^*)$, отримана з Φ заміною функцій $\varphi_1, \varphi_2, \dots, \varphi_s$ на двоїсті до них функції $\varphi_1^*, \varphi_2^*, \dots, \varphi_s^*$, реалізує функцію $f^*(x_n, \dots, x_1)$ – двоїсту до функції $f(x_n, \dots, x_1)$.

Формулу Φ^* називають двоїстою до формули Φ .

В алгебрі Буля принцип двоїстості формулюють так: для отримання формули Φ^* – двоїстої до формули Φ , достатньо в формулі Φ скрізь замінити 0 на 1, 1 на 0, знак кон'юнкції (&) на знак диз'юнкції (\vee), а знак диз'юнкції (\vee) на знак кон'юнкції (&).

Нехай є формула $\Phi = x_3 \overline{x_2} x_1 \vee \overline{x_3} \overline{x_1}$.

Формула $\Phi^* = (x_3 \vee \bar{x}_2 \vee x_1)(\bar{x}_3 \vee \bar{x}_1)$ є двоїстою до формули Φ , вона отримана з Φ заміною знаків операцій: $\& \rightarrow \vee, \vee \rightarrow \&$.

Інші приклади співвідношень, отриманих за принципом двоїстості в алгебрі Буля:

$$\begin{array}{ccc} x_2 \vee x_1 = \overline{\overline{x_2 \cdot x_1}} & & \overline{\overline{x_3 x_2 x_1}} = \overline{\overline{x_3 \vee x_2 \vee x_1}} \\ \Downarrow & & \Downarrow \\ x_2 \cdot x_1 = \overline{\overline{x_2 \vee x_1}} & & \overline{\overline{x_3 \vee x_2 \vee x_1}} = \overline{\overline{x_3 \cdot x_2 \cdot x_1}} \end{array}$$

На принципі двоїстості ґрунтуються основні закони алгебри Буля:

правила де Морґана:

$$\begin{array}{ll} \overline{a \vee b} = \bar{a} \cdot \bar{b} & a \vee b = \overline{\overline{a \cdot b}} \\ \overline{a \cdot b} = \bar{a} \vee \bar{b} & a \cdot b = \overline{\overline{a \vee b}} \end{array}$$

закон дистрибутивності:

$$\begin{array}{l} a(b \vee c) = ab \vee ac \\ a \vee bc = (a \vee b)(a \vee c) \end{array}$$

закон поглинання (абсорбції):

$$\begin{array}{l} a(a \vee b) = a \\ a \vee ab = a \end{array}$$

закон склеювання:

$$\begin{array}{l} ab \vee a\bar{b} = a \\ (a \vee b)(a \vee \bar{b}) = a \end{array}$$

Принцип двоїстості (дуальності) в теорії перемикальних функцій дозволяє істотно скоротити трудомісткість отримання нових можливих співвідношень при розгляді (вивченні) властивостей функцій. Тобто, якщо є певна тотожність (формула), то для отримання нової тотожності (формули) необхідно у відомій тотожності (формулі) замінити знаки операцій на знаки двоїстих до них операцій (рис. 2.37).

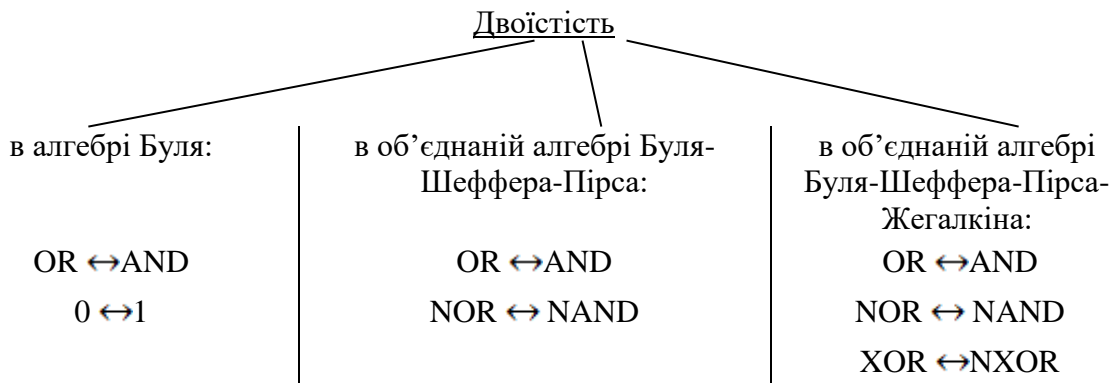


Рис. 2.37. Двоїстість в алгебрах перемикальних функцій

Для будь-якої перемикальної функції існує двоїста до неї функція.

Наприклад, в об'єднаній алгебрі Буля-Шеффера-Пірса-Жегалкіна взаємно двоїстими є операції: OR ↔ AND, NOR ↔ NAND, XOR ↔ NXOR.

Нехай задано функцію від трьох змінних в аналітичному вигляді:

$$f(x_3, x_2, x_1) = \overline{\overline{(x_3 \vee x_1)} x_2} \oplus x_3 x_2 x_1 x_2 \vee x_1$$

Відповідно до принципу двоїстості двоїстою до функції f є функція

$$\Psi = f^*(x_3, x_2, x_1) = \overline{x_3 x_1 \vee x_1} \oplus (x_3 \vee x_2 \vee x_1) \overline{x_2 x_1}$$

Серед перемикальних функцій є так звані самодвоїсті функції.

Самодвоїстою називається перемикальна функція, якщо вона є двоїстою до самої себе, тобто якщо виконується співвідношення

$$f(x_n, \dots, x_1) = \overline{f(\overline{x_n}, \dots, \overline{x_1})}$$

Якщо перемикальна функція f задана аналітично, то для того, щоб з'ясувати, чи є вона самодвоїстою, необхідно:

- 1) у формулі для f , не змінюючи знаків операцій, поставити заперечення над змінними, а над новоотриманою формулою поставити загальне заперечення;
- 2) якщо новоутворена формула співпадає з f , то функція f є самодвоїстою, інакше $-f$ не є самодвоїстою функцією.

Задача 2.18.

З'ясувати, чи є самодвоїстою функція $f(x_2, x_1) = \overline{x_2 \vee x_1}$.

Розв'язування.

Знаходимо функцію двоїсту до $f(x_2, x_1)$:

$$\overline{x_2 \vee x_1} \Rightarrow \overline{\overline{\overline{x_2 \vee x_1}}} = \overline{x_2 x_1}$$

Оскільки $\overline{x_2 x_1} \neq \overline{x_2 \vee x_1}$, то функція $f(x_2, x_1) = \overline{x_2 \vee x_1}$ не є самодвоїстою. ▣

Висновок. Функція Стрілка Пірса (функція АБО-НЕ) не є самодвоїстою.

Задача 2.19.

З'ясувати, чи є самодвоїстою функція $f(x) = \overline{x}$.

Розв'язування.

Знаходимо функцію двоїсту до функції $f(x)$: $\overline{x} \Rightarrow \overline{\overline{\overline{x}}} = \overline{x}$

Оскільки $\overline{x} = \overline{x}$, то функція $f(x) = \overline{x}$ є самодвоїстою. ▣

Висновок: Функція *Заперечення* є самодвоїстою.

Якщо перемикальна функція задана таблицею істинності, то для з'ясування чи є вона самодвоїстою необхідно:

- 1) стовпець f записати у зворотному порядку (знизу вгору), назвемо новий стовпець f' ;
- 2) проінвертувати стовпець f' , отримуємо стовпець f'' ;
- 3) порівняти стовпець f'' зі стовпцем f : якщо вони співпадають, то f – самодвоїста функція, інакше $-f$ не є самодвоїстою.

Самодвоїстими серед 1-місних є дві перемикальні функції: $f_1(x) = x$ (Повторення) та $f_2(x) = \bar{x}$ (Заперечення):

функція *Повторення*

$$f_1(x) = x$$

x	f_1	f'_1	f''_1
0	0	1	0
1	1	0	1

$$f_1''(x) = x$$

(функція *Повторення*)

функція *Заперечення*

$$f_2(x) = \bar{x}$$

x	f_2	f'_2	f''_2
0	1	0	1
1	0	1	0

$$f_2''(x) = \bar{x}$$

(функція *Заперечення*).

Самодвоїстими серед 2-місних перемикальних функцій є 4 функції: $F_3(x_2, x_1) = x_2$, $F_5(x_2, x_1) = x_1$, $F_{10}(x_2, x_1) = \bar{x}_1$, $F_{12}(x_2, x_1) = \bar{x}_2$

Але всі вони залежать лише від однієї змінної.

Задача 2.20. З'ясувати, чи є самодвоїстою повністю визначена перемикальна функція від трьох змінних $z(x_3, x_2, x_1)$ якщо відомо, що вона дорівнює одиниці на наборах 0, 1, 3, 6 і нулю – на решті наборів.

Розв'язування. Побудуємо таблицю істинності функції z та знайдемо функції z' та z'' (табл. 2.11). Оскільки $z'' \neq z$, то функція z не є самодвоїстою. ■

Таблиця 2.11

Перевірка на самодвоїстість функції $z(x_3, x_2, x_1)$

Можливі набори значень аргументів			Задана функція $z(x_3, x_2, x_1)$	Стовпець z у зворотному порядку, z'	Інверсія стовпця z', z''
x_3	x_2	x_1			
0	0	0	1	0	1
0	0	1	1	1	0
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	1	0	0	1
1	1	0	1	1	0
1	1	1	0	1	0

Існує ще одне правило знаходження самодвоїстих функцій. Воно ґрунтується на понятті протилежного набору.

Якщо маємо двійковий набір $\vec{\sigma}_n = \langle \sigma_n, \sigma_{n-1}, \dots, \sigma_1 \rangle$, то протилежним до нього є набір $\vec{\bar{\sigma}}_n = \langle \bar{\sigma}_n, \bar{\sigma}_{n-1}, \dots, \bar{\sigma}_1 \rangle$, утворений порозрядним інвертуванням набору $\vec{\sigma}_n$.

Наприклад, для набору <01101> протилежним є набір <10010>.

Серед можливих 2^n наборів від n змінних є 2^{n-1} пар протилежних наборів.

Перемикальна функція є самодвоїстою, якщо на будь-яких двох протилежних наборах вона має протилежні значення.

Запитання та завдання

1. Дайте визначення двоїстої (дуальної) перемикальної функції.
2. Перемикальна функція задана аналітично. Як знайти двоїсту до неї функцію?
3. Знайдіть двоїсті функції до заданих перемикальних функцій:
а) $f_1 = ab \vee cd$, б) $f_2 = \overline{ac} \vee \overline{c} \vee d$,
в) $f_3 = \overline{ac(b \vee d)}$, г) $f_4 = \overline{a}(b \vee \overline{c}) \vee ad$.
4. Сформулюйте властивість взаємної двоїстості перемикальних функцій.
5. Перемикальна функція задана таблицею істинності. Як знайти двоїсту (дуальну) до неї функцію?
6. Повністю визначена перемикальна функція γ від трьох змінних дорівнює нулю на наборах 1, 3, 4, 7, а на решті наборів – одиниці. Знайдіть двоїсту функцію до функції γ .
7. Сформулюйте принцип двоїстості в теорії перемикальних функцій. Сформулюйте принцип двоїстості в алгебрі Буля, а також в об'єднаній алгебрі Буля-Шеффера-Пірса-Жегалкіна.
8. Наведіть приклади формул, двоїстих до заданих в алгебрі Буля, а також в об'єднаній алгебрі Буля-Шеффера-Пірса-Жегалкіна.
9. Які переваги дає принцип двоїстості (дуальності) в теорії перемикальних функцій?
10. Яка перемикальна функція називається самодвоїстою?
11. Перемикальна функція задана аналітично. Як з'ясувати, чи є вона самодвоїстою?
12. Доведіть аналітично, що функція $f(x) = x$ є самодвоїстою.
13. Перемикальна функція задана таблицею істинності. Як з'ясувати, чи є вона самодвоїстою?

Задачі для самостійного розв'язування

1. Повністю визначена перемикальна функція від трьох змінних задана в ДКНФ: $f = \bar{0} \cdot \bar{3} \cdot \bar{4} \cdot \bar{6}$. Знайти двоїсту до неї функцію та записати її в: а) ДДНФ, б) ДКНФ.
2. Повністю визначена перемикальна функція від трьох змінних задана в ДДНФ: $y = 0 \vee 2 \vee 3 \vee 7$. Знайти двоїсту до неї функцію та записати її в: а) ДДНФ, б) ДКНФ.
3. Перемикальна функція z задана в канонічній формі алгебри Пірса:

$$z(x_3, x_2, x_1) = \overline{x_3 \vee x_2 \vee x_1 \vee x_3 \vee x_2 \vee x_1 \vee x_3 \vee x_2 \vee x_1}$$
 Знайти двоїсту до z функцію та подати її в канонічній формі алгебри Шеффера.
4. На основі принципу двоїстості (дуальності) знайти двоїсті функції до заданих функцій:
 а) $y(x_4, x_3, x_2, x_1) = \overline{x_4 \vee \overline{x_3 x_2 \vee x_1} \oplus \overline{x_4 x_3} (\overline{x_2 \vee x_1})}$,
 б) $y(x_3, x_2, x_1) = \overline{x_3 x_2} \oplus \overline{x_3 \vee x_1 x_3 x_1 \vee x_2}$.
5. Перемикальна функція f задана в канонічній формі алгебри Шеффера: $f = \overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1} \cdot \overline{x_3 x_2 x_1}$.
 З'ясувати, чи є вона самодвоїстою.

2.8. Функціональна повнота систем перемикальних функцій

Оскільки, використовуючи принцип суперпозиції, на основі одно- та двомісних перемикальних функцій можна побудувати довільну перемикальну функцію від будь-якої кількості змінних, природно постає задача – яку мінімальну кількість перемикальних функцій

$$f_1, f_2, \dots, f_k \tag{2.7}$$

достатньо мати, щоб на їх основі, застосовуючи принцип суперпозиції, можна було б утворити будь-яку перемикальну функцію від будь-якої кількості аргументів.

Ця задача є однією з найважливіших в теорії перемикальних функцій, і її формулюють як *проблему функціональної повноти системи перемикальних функцій*.

Систему перемикальних функцій, за допомогою яких шляхом суперпозиції можна подати будь-яку перемикальну функцію від довільної кількості змінних, називають *функціонально повною*.

З'ясуємо, які властивості повинні мати функції f_i , щоб система перемикальних функцій (2.7) була функціонально повною.

Розв'язання цієї задачі ґрунтується на понятті замкнутого відносно суперпозиції класу функцій.

Розрізняють такі типи класів перемикальних функцій: клас функцій, замкнутий клас функцій, передповний клас функцій, повний клас функцій (2^{2^n} функцій).

Будь-яку сукупність функцій можна вважати *класом функцій*.

Клас функцій є замкнутим, якщо будь-яка суперпозиція функцій в ньому не виходить за межі даного класу.

Серед замкнутих класів функцій є передповні класи функцій. *Передповний* – це такий клас функцій, що не співпадає з класом усіх 2^{2^n} перемикальних функцій, однак, якщо до нього включити будь-яку перемикальну функцію, що не входить до даного класу, і за допомогою суперпозиції цієї перемикальної функції та функцій передповного класу утворити новий клас, то він співпаде з класом усіх 2^{2^n} функцій (з *повним класом функцій*).

Дослідження класів перемикальних функцій показало, що існують 5 передповних класів функцій, які називають *класами Поста* (Еміль Леон Пост (Emil Leon Post), 1897 – 1954, – американський математик): клас функцій, що зберігають нуль (K_0); клас функцій, що зберігають одиницю (K_1); клас самодвоїстих функцій (K_C); клас монотонних функцій (K_M); клас лінійних функцій (K_L).

Клас K_0 функцій, що зберігають нуль, утворюють перемикальні функції, які на наборі $\langle 0, \dots, 0 \rangle$ дорівнюють нулю, тобто такі перемикальні функції, для яких $f(0, \dots, 0) = 0$.

Клас K_1 функцій, що зберігають одиницю, утворюють функції, які на наборі $\langle 1, \dots, 1 \rangle$ дорівнюють одиниці, тобто такі перемикальні функції, для яких $f(1, \dots, 1) = 1$.

Клас K_C самодвоїстих функцій утворюють перемикальні функції, для яких виконується співвідношення $f(x_n, \dots, x_1) = \bar{f}(\bar{x}_n, \dots, \bar{x}_1)$.

Клас K_M утворюють монотонні функції. Функція $f(x_n, \dots, x_1)$ є монотонною, якщо для будь-яких двох наборів $\langle \alpha_n, \dots, \alpha_1 \rangle$ і $\langle \beta_n, \dots, \beta_1 \rangle$ таких, що $\alpha_i \leq \beta_i$ для всіх $i = 1, 2, \dots, n$, має місце нерівність

$$f(\alpha_n, \dots, \alpha_1) \leq f(\beta_n, \dots, \beta_1).$$

Клас K_L лінійних функцій утворюють такі функції, які можуть бути подані у вигляді полінома Жегалкіна виду

$$f(x_n, \dots, x_1) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \dots \oplus a_n x_n,$$

де $a_i \in \{0, 1\}$ – коефіцієнти.

Критерієм функціональної повноти системи функцій (2.7) є наступна теорема.

Теорема Поста-Яблонського. Система перемикальних функцій $\{f_1, f_2, \dots, f_k\}$ є функціонально повною тоді і тільки тоді, коли вона сукупно не належить до жодного з передповних класів K_0, K_1, K_C, K_M, K_L .

Тобто, для того, щоб система перемикальних функцій була функціонально повною, необхідно і достатньо, щоб до її складу входила хоча б одна функція, що не зберігає нуль, хоча б одна функція, що не зберігає одиницю, хоча б одна несамодвоїста функція, хоча б одна немонотонна функція і хоча б одна нелінійна функція.

Критерій функціональної повноти системи функцій можна також сформулювати так. Для того, щоб система перемикальних функцій була функціонально повною, необхідно і достатньо, щоб ця система містила хоча б по одній з таких функцій: функцію, що не зберігає одиницю, функцію, що не зберігає нуль, нелінійну функцію, несамодвоїсту функцію та немонотонну функцію.

Не слід ототожнювати функціонально повну систему перемикальних функцій з алгеброю, оскільки поняття алгебри ширше за поняття функціонально повної системи функцій.

Задача 2.21.

Визначити належність функції $f = \overline{x_2 x_1}$ (функція І-НЕ, функція Штрих Шеффера) до передповних класів.

Розв'язування.

Розглянемо таблицю істинності функції f :

x_2	x_1	$f = \overline{x_2 x_1}$
0	0	1
0	1	1
1	0	1
1	1	0

З таблиці істинності бачимо, що $f(0, 0) \neq 0$, отже, f не належить до класу K_0 функцій, що зберігають нуль ($f \notin K_0$), для якого необхідно, щоб $f(0, 0) = 0$.

Оскільки $f(1, 1) \neq 0$, то f не належить до класу K_1 функцій, що зберігають одиницю ($f \notin K_1$), для якого вимагається, щоб $f(1, 1) = 1$.

Перевіримо, чи є функція f самодвоїстою:

x_2	x_1	$f = \overline{x_2 x_1}$	f'	f''
0	0	1	0	1
0	1	1	1	0
1	0	1	1	0
1	1	0	1	0

f' – стовпець, записаний у зворотному порядку, f'' – інверсія стовпця f' .

Оскільки $f'' \neq f$ то функція f не є самодвоїстою, і, отже, f не належить до класу K_C самодвоїстих функцій ($f \notin K_C$).

З'ясуємо, чи є функція $f = \overline{x_2 x_1}$ монотонною. Зафіксуємо набір

x_2	x_1	f
0	0	1
0	1	1
1	0	1
1	1	0

$\langle 00 \rangle = \langle \alpha_1 \alpha_2 \rangle$. Для цього набору наборами $\langle \beta_1 \beta_2 \rangle$ такими, що $\alpha_i \leq \beta_i \in \langle 01 \rangle$, $\langle 10 \rangle$ та $\langle 11 \rangle$. Порівнюючи значення функції на наборі $\langle 00 \rangle$ зі значеннями функції на наборах $\langle 01 \rangle$, $\langle 10 \rangle$, $\langle 11 \rangle$ бачимо, що $f(0,0) > f(1,1)$.

Це означає, що функція f не є монотонною, і, отже, не належить до класу K_M монотонних функцій ($f \notin K_M$), для якого має виконуватись умова $f(\alpha_2, \alpha_1) \leq f(\beta_2, \beta_1)$.

Немонотонність функції f можна визначити й зафіксувавши набір $\langle 0 1 \rangle$, для якого наборами такими, що $\alpha_i \leq \beta_i \in$ набір $\langle 1 1 \rangle$.

Оскільки $f(0, 1) > f(1, 1)$, то функція f не є монотонною.

Перевіримо, чи є функція $f = \overline{x_2 x_1}$ лінійною, для цього її необхідно подати у вигляді полінома Жегалкіна. Спочатку подамо f в ДДНФ:

$$f_{\text{ДДНФ}} = \overline{x_2} \overline{x_1} \vee \overline{x_2} x_1 \vee x_2 \overline{x_1}.$$

Виконаємо заміну: $\vee \rightarrow \oplus$, $\overline{x} \rightarrow x \oplus 1$:

$$f = \overline{x_2} \overline{x_1} \oplus \overline{x_2} x_1 \oplus x_2 \overline{x_1},$$

$$f = (x_2 \oplus 1)(x_1 \oplus 1) \oplus (x_2 \oplus 1)x_1 \oplus x_2(x_1 \oplus 1).$$

Розкриємо дужки та зведемо подібні члени:

$$f = x_2 x_1 \oplus x_2 \oplus x_1 \oplus 1 \oplus x_2 x_1 \oplus x_1 \oplus x_2 x_1 \oplus x_2 = x_2 x_1 \oplus 1.$$

Отриманий поліном Жегалкіна не є лінійним, оскільки містить добуток змінних $x_2 x_1$.

Отже, функція $f = \overline{x_2 x_1}$ не належить до класу K_L лінійних функцій ($f \notin K_L$).

Позначивши знаком “+” належність функції до відповідного передповного класу, а знаком “-” те, що функція до відповідного класу не належить, отримаємо остаточну відповідь, з якої випливає, що функція $f = \overline{x_2 x_1}$ (Штрих Шеффера) не належить до жодного з передповних класів, і, таким чином, утворює функціонально повну систему. ■

Розглянемо належність перемикальних функцій від однієї змінної до передповних класів (табл. 2.12).

Таблиця 2.12
Належність одномісних функцій
до передповних класів

Відповідь до задачі 2.21																																																													
x_2	x_1	$f = \overline{x_2 x_1}$																																																											
0	0	1	<table border="1" style="border-collapse: collapse;"> <thead> <tr> <th colspan="2" style="text-align: center;">x</th> <th>f_0</th> <th>f_1</th> <th>f_2</th> <th>f_3</th> </tr> <tr> <td colspan="2" style="text-align: center;">x</td> <td>0</td> <td>x</td> <td>\bar{x}</td> <td>1</td> </tr> <tr> <td colspan="2" style="text-align: center;">0</td> <td>0</td> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td colspan="2" style="text-align: center;">1</td> <td>0</td> <td>1</td> <td>0</td> <td>1</td> </tr> </thead> <tbody> <tr> <td rowspan="5" style="text-align: center; vertical-align: middle;">Клас</td> <td style="text-align: center;">K_0</td> <td style="text-align: center;">–</td> <td style="text-align: center;">+</td> <td style="text-align: center;">+</td> <td style="text-align: center;">–</td> <td style="text-align: center;">–</td> </tr> <tr> <td style="text-align: center;">K_1</td> <td style="text-align: center;">–</td> <td style="text-align: center;">–</td> <td style="text-align: center;">+</td> <td style="text-align: center;">–</td> <td style="text-align: center;">+</td> </tr> <tr> <td style="text-align: center;">K_C</td> <td style="text-align: center;">–</td> <td style="text-align: center;">–</td> <td style="text-align: center;">+</td> <td style="text-align: center;">+</td> <td style="text-align: center;">–</td> </tr> <tr> <td style="text-align: center;">K_M</td> <td style="text-align: center;">–</td> <td style="text-align: center;">+</td> <td style="text-align: center;">+</td> <td style="text-align: center;">–</td> <td style="text-align: center;">+</td> </tr> <tr> <td style="text-align: center;">K_L</td> <td style="text-align: center;">–</td> <td style="text-align: center;">+</td> <td style="text-align: center;">+</td> <td style="text-align: center;">+</td> <td style="text-align: center;">+</td> </tr> </tbody> </table>				x		f_0	f_1	f_2	f_3	x		0	x	\bar{x}	1	0		0	0	1	1	1		0	1	0	1	Клас	K_0	–	+	+	–	–	K_1	–	–	+	–	+	K_C	–	–	+	+	–	K_M	–	+	+	–	+	K_L	–	+	+	+	+
x		f_0					f_1	f_2	f_3																																																				
x		0					x	\bar{x}	1																																																				
0		0					0	1	1																																																				
1		0	1	0	1																																																								
Клас	K_0	–	+	+	–	–																																																							
	K_1	–	–	+	–	+																																																							
	K_C	–	–	+	+	–																																																							
	K_M	–	+	+	–	+																																																							
	K_L	–	+	+	+	+																																																							
0	1	1																																																											
1	0	1																																																											
1	1	0																																																											

Наприклад, функція $f_0 = 0$ належить (+) до класу K_0 функцій, що зберігають нуль, оскільки на наборі $\langle 0 \rangle$, вона дорівнює нулю; не належить (–) до класу K_1 функцій, що зберігають одиницю, оскільки на наборі $\langle 1 \rangle$, вона не дорівнює одиниці; не належить до класу K_C самодвоїстих функцій, оскільки $f_0(x) \neq \bar{f}_0(\bar{x})$; належить (+) до класу K_M монотонних функцій, оскільки $f_0(0) \leq f_0(1)$; належить (+) до класу K_L лінійних функцій, оскільки її можна подати у вигляді полінома Жегалкіна виду

$$f_0(x) = a_0 \oplus a_1 x = 0 \oplus 0x = 0.$$

Аналогічно можна визначити належність до передповних класів решти $f_1 - f_3$ функцій від однієї змінної (див. табл. 2.12).

Як видно з табл. 2.12 жодна з 1-місних функцій не утворює функціонально повної системи, оскільки немає жодного стовпця $K_0 - K_L$, який би містив лише позначки "–"; будь-яка сукупність з двох, трьох, і навіть всіх чотирьох одномісних функцій також не є функціонально повною системою, оскільки серед 1-місних перемикальних функцій немає жодної нелінійної функції (див. останній рядок табл. 2.12).

Розглянемо належність перемикальних функцій від двох змінних до передповних класів (табл. 2.13).

Таблиця 2.13

Належність 2-місних перемикальних функцій до передповних класів

$x_2 \ x_1$	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
	0	$x_2 x_1$	$x_2 \bar{x}_1$	x_2	$\bar{x}_2 x_1$	x_1	$x_2 \oplus x_1$	$x_2 \vee x_1$	$\bar{x}_2 \vee \bar{x}_1$	$\bar{x}_2 \oplus \bar{x}_1$	\bar{x}_1	$x_2 \vee \bar{x}_1$	\bar{x}_2	$\bar{x}_2 \vee x_1$	$\bar{x}_2 x_1$	1
0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	1	1	0	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	0	0	1	0	1	0	1	0	1	0	1
Клас	K_0	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-
	K_1	-	+	-	+	-	-	+	-	+	-	+	-	+	-	+
	K_C	-	-	-	+	-	-	-	-	-	+	-	+	-	-	-
	K_M	+	+	-	+	-	-	+	-	-	-	-	-	-	-	+
	K_L	+	-	-	+	-	+	-	-	+	+	-	+	-	-	+

Примітка. Знак "+" означає належність функції до відповідного передповного класу, знак "-" означає, що функція до відповідного класу не належить.

Оскільки функція $F_{10}(x_2, x_1) = \bar{x}_1$ та функція $F_{12}(x_2, x_1) = \bar{x}_2$ мають однакову належність до класів $K_0 - K_L$ (див. табл. 2.13), то можна вважати, що разом F_{10}, F_{12} подають функцію \bar{x} . Однакову належність до класів $K_0 - K_L$ мають також функції $F_3(x_2, x_1) = x_2$ та $F_5(x_2, x_1) = x_1$, тому також можна вважати, що разом F_3, F_5 подають функцію x . Отже, табл. 2.13 показує належність як 2-, так і 1-місних функцій до передповних класів.

Аналізуючи табл. 2.13 бачимо, що функція $F_8(x_2, x_1) = x_2 \downarrow x_1 = x_2 \vee x_1$ (АБО-НЕ, Стрілка Пірса) утворює функціонально повну систему, оскільки не належить (-) до жодного з передповних класів. Функціонально повну систему утворює й функція $F_{14}(x_2, x_1) = x_2 / x_1 = \overline{x_2 x_1}$ (І-НЕ, Штрих Шеффера), яка також не належить (-) до жодного з класів $K_0 - K_L$.

Функціонально повні системи утворюють також наступні сукупності функцій: $F_0 = 0$ і $F_{13} = x_2 \rightarrow x_1$;

$$F_1 = x_2 x_1 \text{ і } F_{10}(F_{12}) = \bar{x}; F_2 = \overline{x_2 \rightarrow x_1} \text{ і } F_9 = \overline{x_2 \oplus x_1};$$

$$F_2 = \overline{x_2 \rightarrow x_1} \text{ і } F_{10}(F_{12}) = \bar{x}; F_2 = \overline{x_2 \rightarrow x_1} \text{ і } F_{11} = x_2 \leftarrow x_1;$$

$$F_7 = x_2 \vee x_1 \text{ і } F_{10}(F_{12}) = \bar{x}; F_6 = x_2 \oplus x_1 \text{ і } F_{13} = x_2 \rightarrow x_1,$$

а також багато інших сукупностей по дві, три і т.д. функцій (див. табл. 2.13).

З функціонально повних систем, які можна утворити з 1- та 2-місних функцій, практичне застосування мають функціонально повні системи 1 – 6 (рис. 2.38). Перші п'ять функціонально повних систем є ненадлишковими, а система, утворена з функцій І, АБО, НЕ є надлишковою.

Дійсно, алгебра Буля має надлишкову систему функцій, що видно з табл. 2.14. З цієї системи можна вилучити функцію І або функцію АБО, тобто для забезпечення функціональної повноти достатньо залишити пари функцій {І, НЕ} або {АБО, НЕ}.

Таблиця 2.14

Належність функцій алгебри Буля до передповних класів

Клас	Функція		
	І	АБО	НЕ
K_0	+	+	-
K_1	+	+	-
K_C	-	-	+
K_M	+	+	-
K_L	-	-	+

Ненадлишкові функціонально повні системи

- 1) $\overline{x_2 \vee x_1}$ {АБО-НЕ} – основа алгебри Пірса,
- 2) $\overline{x_2 \cdot x_1}$ {І-НЕ} – основа алгебри Шеффера,
- 3) $x_2 \vee x_1, \overline{x}$ {АБО, НЕ},
- 4) $x_2 \cdot x_1, \overline{x}$ {І, НЕ},
- 5) $x_2 \cdot x_1, x_2 \oplus x_1, 1$ {І, Виключне АБО, 1} – основа алгебри Жегалкіна

Надлишкові функціонально повні системи

- 6) $x_2 \cdot x_1, x_2 \vee x_1, \overline{x}$ {І, АБО, НЕ} – основа алгебри Буля,
- 7) $x_2 \cdot x_1, x_2 \oplus x_1, \overline{x}$ {І, Виключне АБО, НЕ},
- 8) $x_2 \vee x_1, x_2 \oplus x_1, \overline{x}$ {АБО, Виключне АБО, НЕ}

Рис. 2.38. Приклади деяких функціонально повних систем функцій серед 1-та 2-місних функцій

Функціонально повні системи перемикальних функцій 1 – 6 з рис. 2.38 можна узагальнити на випадок довільної кількості змінних (табл. 2.15).

Функціональна повнота систем перемикальних функцій 1 – 6 з табл. 2.15 означає, що взявши будь-яку з цих систем, можна побудувати будь-яку перемикальну функцію від заданої кількості змінних.

Наприклад, достатньо мати лише одну функцію – функцію Пірса (функцію АБО-НЕ), щоб можна було утворити будь-яку перемикальну функцію, і, отже, достатньо мати лише один тип логічних елементів – елементи АБО-НЕ, щоб можна було побудувати логічну схему будь-якої перемикальної функції від довільної кількості змінних.

Або достатньо мати лише дві функції – функцію І та функцію НЕ, щоб можна було утворити будь-яку іншу перемикальну функцію, і, отже, достатньо мати лише два типи логічних елементів – елементи І та елементи НЕ, щоб можна було побудувати логічну схему будь-якої перемикальної функції від довільної кількості змінних.

Для реалізації на практиці функціонально повної системи {І, Виключне АБО, 1}, яка складає основу алгебри Жегалкіна, достатньо мати два типи логічних елементів – елементи І та суматори за модулем два (див. табл. 2.15), оскільки для відтворення третьої функції системи – одиниці ($F = 1$), логічний елемент не потрібен – в реальній електронній схемі завжди схемотехнічно присутня електрична шина, рівень напруги на якій є рівнем логічної одиниці.

Таблиця 2.15

Використовувані на практиці функціонально повні системи перемикальних функцій та їх елементний базис

№	Функціонально повна система функцій	Елементний базис функціонально повної системи функцій
1	{АБО-НЕ} (Стрілка Пірса)	
2	{І-НЕ} (Штрих Шеффера)	
3	{АБО, НЕ} (диз'юнкція, заперечення)	
4	{І, НЕ} (кон'юнкція, заперечення)	
5	{І, Виключне АБО, 1} (кон'юнкція, сума за модулем два, одиниця)	
6	{І, АБО, НЕ} (кон'юнкція, диз'юнкція, заперечення)	

Стосовно ненадлишкових функціонально повних систем {АБО, НЕ} та {І, НЕ} слід додатково сказати наступне. На основі кожної з них можна побудувати окрему алгебру. Але історично склалось так, що на практиці більшого поширення набула саме надлишкова функціонально повна система перемикальних функцій {І, АБО, НЕ} – основа алгебри Буля, яка є об'єднанням ненадлишкових систем {АБО, НЕ} та {І, НЕ}.

Розв'язання проблеми функціональної повноти систем перемикальних функцій забезпечує лише *можливість подання* довільної функціональної залежності від заданої кількості аргументів за допомогою *мінімальної кількості базових функцій* (операцій), які сукупно мають властивість функціональної повноти, а, отже, й *можливість побудови* комбінаційної схеми, що відтворює функціональну залежність, за допомогою *мінімальної кількості типів* логічних елементів. Однак, при цьому не вирішується питання оптимальності комбінаційної схеми. Як показує практика проектування логічних схем *об'єднання елементних базисів*, які належать кільком функціонально повним системам (наприклад, системам {АБО-НЕ}, {І-НЕ}, {І, АБО, НЕ}), дозволяє будувати *оптимальніші комбінаційні схеми* (за показниками апаратної складності та швидкодії), ніж при застосуванні елементного базису лише однієї якоїсь окремо взятої функціонально повної системи перемикальних функцій.

Запитання та завдання

1. Яку систему перемикальних функцій називають функціонально повною?
2. Сформулюйте проблему функціональної повноти систем перемикальних функцій.
3. Дайте визначення таких понять: клас функцій, замкнутий клас функцій, передповний клас функцій, повний клас функцій.
4. Дайте визначення п'яти передповних класів функцій (класів Поста).
5. Сформулюйте теорему Поста-Яблонського.
6. Визначте належність до передповних класів функції:
а) $f(x) = x$; б) $f(x) = \bar{x}$; в) $f(x_2, x_1) = \overline{x_2 \vee x_1}$; г) $f(x_2, x_1) = x_2 \cdot x_1$;
д) $f(x_3, x_2, x_1)$, яка дорівнює одиниці на наборах 0, 1, 3, 6 і нулю – на решті наборів.
7. Перевірте, чи є функціонально повною з числа 1- та 2-місних функцій система перемикальних функцій: а) $\{F_1, F_{12}\}$; б) $\{F_4, F_9\}$; в) $\{F_6, F_{10}\}$; г) $\{F_6, F_{10}, F_{11}\}$.

3.

МІНІМІЗАЦІЯ ПЕРЕМИКАЛЬНИХ ФУНКЦІЙ

Будь-яку перемикальну функцію можна подати в канонічній нормальній формі тієї чи іншої алгебри, зокрема в досконалій диз'юнктивній нормальній формі (ДДНФ) або в досконалій кон'юнктивній нормальній формі (ДКНФ). Однак ДДНФ та ДКНФ, як і інші канонічні нормальні форми, у загальному випадку не є найпростішими формами функції, тобто такими, що при їх апаратній реалізації досягається мінімальна складність комбінаційної схеми. Отже, необхідні способи спрощення форми подання функції, які б приводили до зменшення апаратної та часової складності відповідної комбінаційної схеми.

3.1. Проблема мінімізації перемикальних функцій

У теорії перемикальних функцій розрізняють *аналітичну складність* – складність аналітичної форми (формули) функції, та *апаратну складність* – складність комбінаційної схеми, що реалізує перемикальну функцію.

Аналітичну складність перемикальної функції оцінюють показником ціни. *Ціна аналітичної форми* перемикальної функції – це загальна кількість букв, що входять до алгебраїчного виразу.

Наприклад, ціна (C) аналітичної форми перемикальної функції $Y = x_3 \bar{x}_2 x_1 \vee \bar{x}_3 x_2$ становить 5 ($C = 5$).

Апаратну складність комбінаційної схеми, яка реалізує деяку перемикальну функцію, зазвичай оцінюють як загальну кількість входів (K) усіх логічних елементів, що входять до складу схеми, тобто показником Квайна.

Нехай перемикальна функція y задана таблицею істинності (табл. 3.1).

Подамо функцію в ДДНФ:

$$Y = \bar{x}_3 \bar{x}_2 x_1 \vee x_3 \bar{x}_2 \bar{x}_1 \vee x_3 \bar{x}_2 x_1 \vee x_3 x_2 \bar{x}_1. \quad (\text{форма 1})$$

Аналітична складність форми 1 функції становить $C_1 = 12$, а апаратна складність комбінаційної схеми, що реалізує форму 1, – $K_1 = 16$ (рис. 3.1a).

Спростимо форму 1. Винесемо за дужки спільну частину першого та третього доданків, а також другого та четвертого доданків:

$$\begin{aligned} y &= \bar{x}_2 x_1 (\bar{x}_3 \vee x_3) \vee x_3 \bar{x}_1 (\bar{x}_2 \vee x_2) = \\ &= \bar{x}_2 x_1 \vee x_3 \bar{x}_1. \end{aligned} \quad (\text{форма 2})$$

Таблиця 3.1
Таблиця істинності
функції y

x_3	x_2	x_1	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

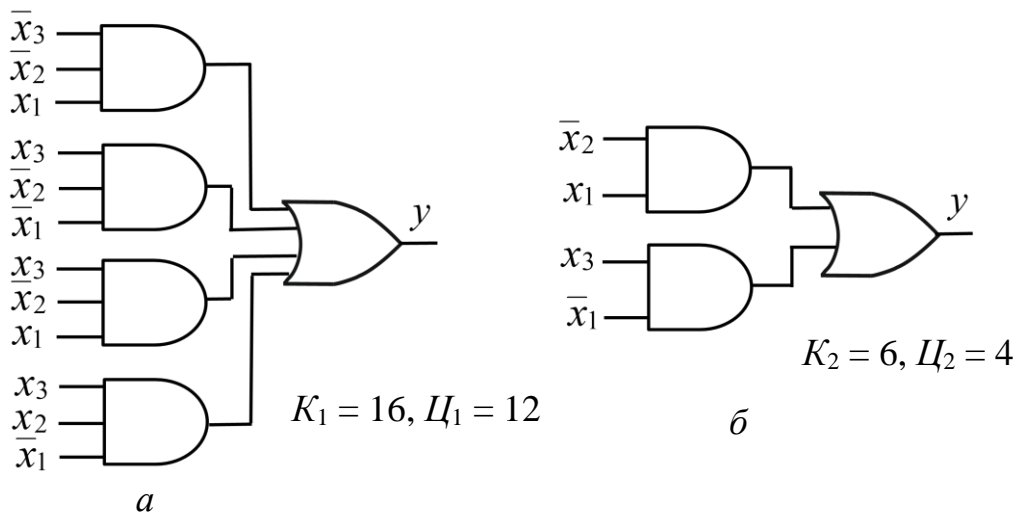


Рис. 3.1. Варіанти апаратної реалізація перемикальної функції y ,
заданої таблицею істинності (табл. 3.1):
 a – на основі ДДНФ; $б$ – після спрощення функції

Формі 2 функції y , яка має ціну $C_2 = 4$, відповідає комбінаційна схема на рис. 3.1б, складність якої за Квайном складає $K_2 = 6$.

Як бачимо, різні форми однієї й тієї самої функції можуть мати різну ціну, і їм можуть відповідати комбінаційні схеми різної складності. Очевидно, що необхідно задану функцію подати в такій формі, щоб відповідна їй комбінаційна схема мала якнайменшу складність.

Вважатимемо, що формі функції з мінімальною ціною відповідатиме комбінаційна схема мінімальної складності.

Отже, задача зводиться до пошуку форми функції з мінімальною ціною.

Процес знаходження форми функції з мінімальною ціною називають *мінімізацією функції*. Метою мінімізації перемикальних функцій є спрощення (зменшення складності) комбінаційних схем, що реалізують ці функції.

Надалі розглядатимемо методи мінімізації перемикальних функцій в диз'юнктивних формах булевої алгебри (хоча мінімізацію теоретично можна виконувати в різних алгебрах).

Введемо деякі означення. Перемикальну функцію φ називають *імплікантою* функції f , якщо на будь-якому наборі виконується умова $\varphi \leq f$. Імпліканта функції частково покриває функцію, тобто вона обов'язково дорівнює нулю на тих наборах, на яких нулю дорівнює функція, а значення одиниці може набувати не на всіх наборах, на яких функція має значення одиниці.

Нехай функція $f(x_2, x_1)$ задана таблицею істинності:

x_2	x_1	f	φ	ψ	γ	$\varphi \vee \psi$
0	0	1	1	0	0	1
0	1	1	1	1	1	1
1	0	0	0	0	1	0
1	1	1	0	1	1	1

Функції φ та ψ є імплікантами функції f , а функція γ імплікантою функції f не є.

Якщо φ і ψ є імплікантами функції f , то їх диз'юнкція $\varphi \vee \psi$ також є імплікантою функції f .

У наведеному прикладі диз'юнкція $\varphi \vee \psi$ імплікант співпадає з функцією f .

Надалі розглядатимемо такі імпліканти, які є кон'юнктивними термами r -го рангу, де $r \leq n$ (n – кількість аргументів функції):

$$A = \tilde{x}_{i_1} \cdot \tilde{x}_{i_2} \cdot \dots \cdot \tilde{x}_{i_r}.$$

Під власною частиною терма (добутку) A розуміють терм B , який можна отримати з A викреслюванням однієї або кількох змінних.

Простою імплікантою функції f називають такий терм (елементарний добуток), який є імплікантою функції f , а жодна його власна частина імплікантою функції f не є.

Будь-яка функція має скінченну множину простих імплікант, кількість яких не перевищує кількості конститuent одиниці в ДДНФ функції.

Диз'юнкцію всіх простих імплікант функції називають *скороченою ДНФ (СДНФ)* функції.

Термін «скорочена» означає, що кількість членів в диз'юнкції не більша за кількість конститuent одиниці в ДДНФ функції.

Будь-яка перемикальна функція має єдину скорочену ДНФ.

Сукупність усіх простих імплікант в СДНФ покриває всі одиничні значення перемикальної функції, але може містити надлишкові (зайві)

імпліканти, які повторно покривають функцію на деяких наборах, тобто в загальному випадку СДНФ не є мінімальною формою функції. З метою зменшення ціни форми такі надлишкові імпліканти можна вилучити зі складу СДНФ.

СДНФ без надлишкових імплікант називають *тупиковою ДНФ* (ТДНФ) перемикальної функції.

ТДНФ з мінімальною ціною називають *мінімальною ДНФ* (МДНФ) перемикальної функції. Функція може мати декілька ТДНФ, і, відповідно, декілька МДНФ. Мінімальну(-ні) ДНФ вибирають серед ТДНФ перемикальної функції.

Таким чином, мінімізація перемикальної функції зводиться до знаходження МДНФ функції.

Зазначимо, що задача переходу від тупикової (-вих) ДНФ до абсолютно мінімальної ДНД перемикальної функції дістала назву *задачі факторизації*. Ефективних алгоритмів (таких, що не ґрунтуються на повному наборі) вирішення цієї задачі на даний час не існує.

Отже, ланцюжком для отримання форми перемикальної функції (п.ф.) з мінімальною ціною є послідовність: ДДНФ → СДНФ → ТДНФ → МДНФ (рис. 3.2).

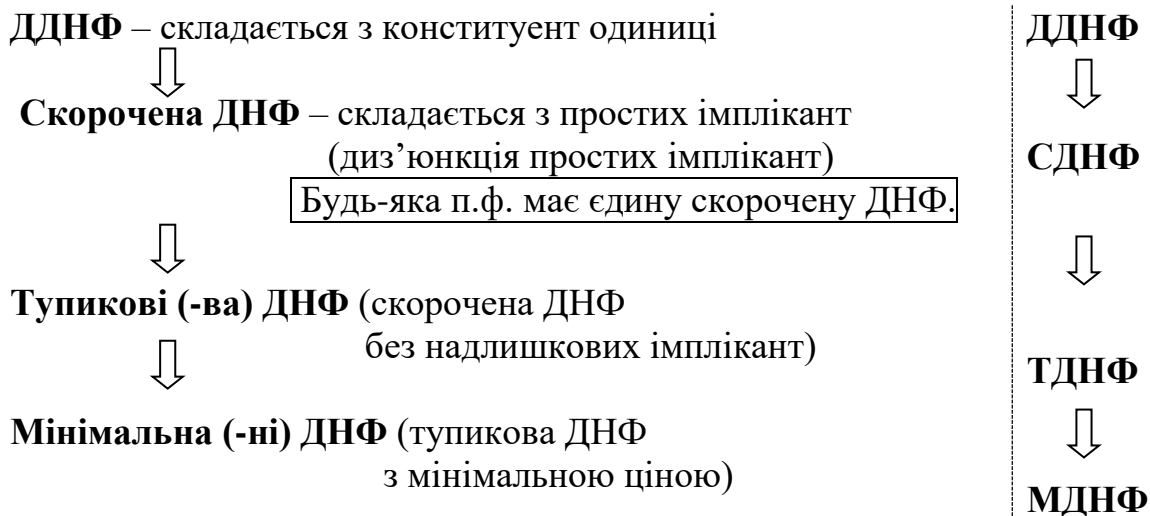


Рис. 3.2. Послідовність отримання форми перемикальної функції з мінімальною ціною

Розрізняють два види методів мінімізації перемикальних функцій – аналітичні та графічні. Найбільш поширеним серед аналітичних є метод мінімізації Квайна.

Запитання та завдання

1. Чим викликана потреба в мінімізації перемикальних функцій?
2. Дайте означення термінів: аналітична складність, апаратна складність.
3. Що розуміють під ціною аналітичної форми перемикальної функції?
4. Що розуміють під терміном *мінімізація функції*? Що є метою мінімізації перемикальної функції?
5. Дайте визначення *імпліканти* функції. Яку властивість мають імпліканти перемикальної функції?
6. Що називають *простою імплікантою* функції?
7. Дайте визначення таких термінів: скорочена ДНФ, тупикова ДНФ, мінімальна ДНФ.
8. Сформулюйте задачу факторизації.
9. До чого зводиться задача мінімізації перемикальної функції?
10. Назвіть види методів мінімізації перемикальних функцій.

3.2. Метод мінімізації Квайна

Метод мінімізації Квайна застосовують до перемикальної функції, яка подана в ДДНФ.

Якщо перемикальна функція подана в довільній формі, то за допомогою тотожних співвідношень булевої алгебри спочатку отримують ДНФ функції, а потім, застосувавши множення кон'юнктивних термів, що не є конституентами одиниці, на $x_i \vee \bar{x}_i$, отримують ДДНФ функції ($x_i \vee \bar{x}_i = 1$).

Множення на $x \vee \bar{x}$ називають *операцією розгортання*. Вона не змінює значення терма і є протилежною до операції склеювання.

Нехай дана ДНФ функції $y(x_3, x_2, x_1)$ від трьох змінних $y = x_3 x_1 \vee \bar{x}_3 \bar{x}_1 \vee \bar{x}_2$.

Отримаємо її ДДНФ застосувавши операцію розгортання:

$$\begin{aligned}
 y &= x_3 x_1 \vee \bar{x}_3 \bar{x}_1 \vee \bar{x}_2 = \\
 &= x_3 (x_2 \vee \bar{x}_2) x_1 \vee \bar{x}_3 (x_2 \vee \bar{x}_2) \bar{x}_1 \vee (x_3 \vee \bar{x}_3) \bar{x}_2 (x_1 \vee \bar{x}_1) = \\
 &= x_3 x_2 x_1 \vee \underline{x_3 \bar{x}_2 x_1} \vee \bar{x}_3 x_2 \bar{x}_1 \vee \underline{\bar{x}_3 \bar{x}_2 \bar{x}_1} \vee \underline{x_3 \bar{x}_2 x_1} \vee \\
 &\vee \bar{x}_3 \bar{x}_2 x_1 \vee \bar{x}_3 \bar{x}_2 x_1 \vee \underline{\bar{x}_3 \bar{x}_2 \bar{x}_1} = x_3 x_2 x_1 \vee x_3 \bar{x}_2 x_1 \vee \bar{x}_3 x_2 \bar{x}_1 \vee \\
 &\vee \bar{x}_3 \bar{x}_2 \bar{x}_1 \vee \bar{x}_3 \bar{x}_2 x_1 \vee \bar{x}_3 \bar{x}_2 \bar{x}_1 = \mathcal{Y}_{\text{ДДНФ}}.
 \end{aligned}$$

Метод мінімізації Квайна ґрунтується на використанні співвідношення неповного склеювання та співвідношення поглинання.

Операція звичайного (повного) склеювання виглядає так:

$$ab \vee a\bar{b} = a. \quad (3.1)$$

Дійсно $ab \vee a\bar{b} = a(b \vee \bar{b}) = a \cdot 1 = a$.

До правої і лівої частин (3.1) додамо вираз $ab \vee a\bar{b}$. Отримаємо

$$(ab \vee a\bar{b}) \vee (ab \vee a\bar{b}) = a \vee (ab \vee a\bar{b})$$

$$\text{або } ab \vee a\bar{b} = ab \vee a\bar{b} \vee a. \quad (3.2)$$

Вираз (3.2) називають *співвідношенням неповного склеювання*.

Це співвідношення часто записують у більш загальному вигляді:

$$Ax \vee A\bar{x} = Ax \vee A\bar{x} \vee A, \quad (3.3)$$

де A – кон'юнктивний терм, x – змінна.

Фізичний зміст співвідношення неповного склеювання: є два терми Ax і $A\bar{x}$, що відрізняються один від одного лише наявністю заперечення в одній зі змінних, їх спільною частиною є A .

Правило неповного склеювання застосовують при порівнянні двох термів з метою виділення їх спільної частини. Якщо два порівнювані терми n -го рангу відрізняються один від одного лише наявністю заперечення в однієї зі змінних, то їх спільною частиною є терм $(n - 1)$ -го рангу.

Нехай два терми 4-го рангу $\bar{x}_4 \bar{x}_3 x_2 \bar{x}_1$ і $\bar{x}_4 x_3 x_2 \bar{x}_1$ є членами деякої диз'юнктивної форми перемикальної функції. Виділимо їх спільну частину застосувавши правило неповного склеювання:

$$\bar{x}_4 \bar{x}_3 x_2 \bar{x}_1 \vee \bar{x}_4 x_3 x_2 \bar{x}_1 = \bar{x}_4 \bar{x}_3 x_2 \bar{x}_1 \vee \bar{x}_4 x_3 x_2 \bar{x}_1 \vee \bar{x}_4 x_2 \bar{x}_1$$

Спільною частиною є терм 3-го рангу $\bar{x}_4 x_2 \bar{x}_1$.

Співвідношення поглинання у загальному випадку записують як

$$BC \vee C = C, \quad (3.4)$$

де B, C – кон'юнктивні терми.

Теорема Квайна. Якщо в ДДНФ перемикальної функції спочатку виконати всі можливі операції неповного склеювання, а потім усі можливі операції поглинання, то дістанемо скорочену ДНФ функції, тобто диз'юнкцію всіх простих імплікант функції.

Процес знаходження МДНФ перемикальної функції (п.ф.) складається з двох етапів (рис. 3.3).

Перший етап полягає у виконанні, починаючи з ДДНФ, всіх можливих операцій неповного склеювання, а потім всіх можливих операцій поглинання. Результатом першого етапу є отримання скороченої ДНФ функції. Будь-яка перемикальна функція має єдину СДНФ.

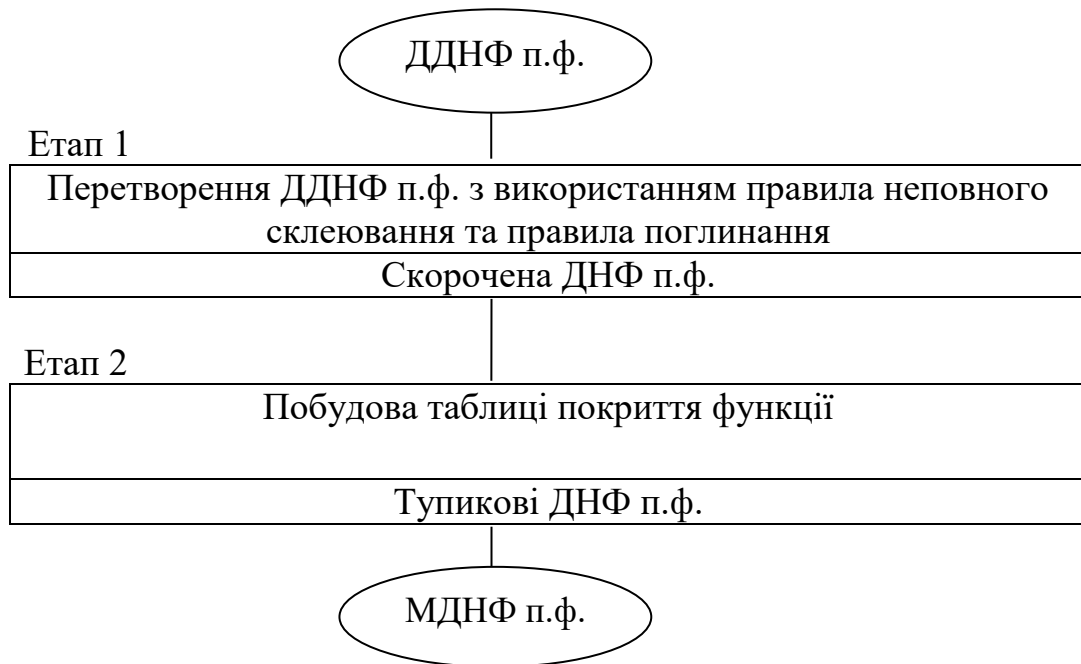


Рис. 3.3. Узагальнена схема процесу мінімізації перемикальної функції методом Квайна

Другий етап полягає в побудові спеціальної таблиці, яку називають *таблицею покриття функції або імплікантною матрицею*. Таблиця покриття функції дозволяє позбутися надлишкових простих імплікант, які можуть бути в складі скороченої ДНФ.

Результатом другого етапу є отримання множини тупикових ДНФ функції, серед яких вибирають ТДНФ з найменшою ціною. ТДНФ з найменшою ціною й буде мінімальною ДНФ (МДНФ) функції, тобто формою з мінімальною ціною.

Зауваження. Серед ТДНФ декілька форм можуть мати однакову найменшу ціну. Тоді мінімальною ДНФ функції може бути обрана будь-яка з них.

Розглянемо процес отримання МДНФ функції методом Квайна докладніше.

1. Записуємо перемикальну функцію в ДДНФ.
2. Застосовуємо правило неповного склеювання (співвідношення (3.3)) до всіх можливих пар конститuent одиниці функції, тобто до таких пар, які відрізняються лише наявністю заперечення в однієї зі змінних. Результатом цього є отримання імплікант – термів $(n - 1)$ -го рангу. Далі застосовуємо правило неповного склеювання до термів (імплікант) $(n - 1)$ -го рангу й отримуємо нові імпліканти –

терми $(n - 2)$ -го рангу і так далі, поки формування нових імплікант стане неможливим.

3. Записуємо задану функцію у вигляді диз'юнктивної форми, перша частина якої є диз'юнкцією конститuent одиниці (первісний запис функції в ДДНФ), друга частина – диз'юнкцією імплікант – термів $(n - 1)$ -го рангу, наступна частина – диз'юнкцією імплікант – термів $(n - 2)$ -го рангу і так далі, остання частина – диз'юнкцією імплікант, застосування операції неповного склеювання до яких вже не породжує нових імплікант.
4. В отриманому записі функції (п.3) виконаємо всі можливі поглинання. Результатом цього є отримання всіх простих імплікант, диз'юнкція яких є скороченою ДНФ функції.
5. Будуємо таблицю покриття (імплікантну матрицю) функції, стовпцями якої є конститuentи одиниці функції, а рядками – прості імпліканти.
6. На перетині рядків і стовпців таблиці покриття ставимо знак \vee у тому випадку, якщо дана проста імпліканта є власною частиною відповідної конститuentи одиниці.
7. Формуємо *ядро функції* як сукупність *істотних імплікант*. Пошук істотних імплікант полягає в наступному. Шукаємо в таблиці покриття стовпці, які містять лише один знак \vee . Розташування цього знака й показує рядок, у якому міститься істотна імпліканта.
Істотні імпліканти (ядро функції) входять до складу всіх тупикових ДНФ функції. Диз'юнкція істотних імплікант буде складовою частиною МДНФ функції.
8. Формуємо тупикові ДНФ. Тупикова ДНФ – це набір (множина) таких простих імплікант (таких рядків таблиці), знаки \vee в яких покривають всі стовпці таблиці покриття.
Для цього за основу беремо рядки, яким відповідають істотні імпліканти, тобто ядро, й долучаємо до них інші рядки (імпліканти) так, щоб покритими виявились всі стовпці таблиці.
Таких комбінацій рядків, а, отже, й ТДНФ, може бути декілька.
9. З отриманих ТДНФ вибираємо форму з найменшою ціною. Це й буде МДНФ функції.

Розв'яжемо задачу.

Задача 3.1. Функція $y(x_3, x_2, x_1)$ від 3-х змінних задана таблицею істинності.

x_3	x_2	x_1	y	Виконати мінімізацію функції у методом Квайна.
0	0	0	0	
0	0	1	1	Розв'язування.
0	1	0	1	1. Подамо функцію у в ДДНФ:
0	1	1	0	
1	0	0	1	1 2 3 4 5
1	0	1	1	$y = \bar{x}_3 \bar{x}_2 x_1 \vee \bar{x}_3 x_2 \bar{x}_1 \vee x_3 \bar{x}_2 \bar{x}_1 \vee x_3 \bar{x}_2 x_1 \vee x_3 x_2 \bar{x}_1$
1	1	0	1	
1	1	1	0	2. Застосуємо правило неповного склеювання до

всіх можливих пар констuent одиниці функції – таких, що відрізняються лише наявністю заперечення в одній з змінних:

$$1-4: \bar{x}_3 \bar{x}_2 x_1 \vee x_3 \bar{x}_2 x_1 = \bar{x}_3 \bar{x}_2 x_1 \vee x_3 \bar{x}_2 x_1 \vee \bar{x}_2 x_1,$$

$$2-5: \bar{x}_3 x_2 \bar{x}_1 \vee x_3 x_2 \bar{x}_1 = \bar{x}_3 x_2 \bar{x}_1 \vee x_3 x_2 \bar{x}_1 \vee x_2 \bar{x}_1,$$

$$3-4: x_3 \bar{x}_2 \bar{x}_1 \vee x_3 \bar{x}_2 x_1 = x_3 \bar{x}_2 \bar{x}_1 \vee x_3 \bar{x}_2 x_1 \vee x_3 \bar{x}_2,$$

$$3-5: x_3 \bar{x}_2 \bar{x}_1 \vee x_3 x_2 \bar{x}_1 = x_3 \bar{x}_2 \bar{x}_1 \vee x_3 x_2 \bar{x}_1 \vee x_3 \bar{x}_1.$$

Ми одержали множину імплікант – термів 2-го рангу: $\bar{x}_2 x_1, x_2 \bar{x}_1, x_3 \bar{x}_2, x_3 \bar{x}_1$.

Подальше склеювання цих імплікант неможливе.

3. Запишемо задану перемикальну функцію у вигляді

$$y = \bar{x}_3 \bar{x}_2 x_1 \vee \bar{x}_3 x_2 \bar{x}_1 \vee x_3 \bar{x}_2 \bar{x}_1 \vee x_3 \bar{x}_2 x_1 \vee x_3 x_2 \bar{x}_1 \vee \bar{x}_2 x_1 \vee x_2 \bar{x}_1 \vee x_3 \bar{x}_2 \vee x_3 \bar{x}_1.$$

4. В отриманому виразі виконаємо всі можливі поглинання:

$$y = \bar{x}_3 \bar{x}_2 x_1 \vee \bar{x}_3 x_2 \bar{x}_1 \vee \cancel{x_3 \bar{x}_2 \bar{x}_1} \vee \cancel{x_3 \bar{x}_2 x_1} \vee \cancel{x_3 x_2 \bar{x}_1} \vee \bar{x}_2 x_1 \vee x_2 \bar{x}_1 \vee \underline{x_3 \bar{x}_2} \vee \underline{x_3 \bar{x}_1}.$$

Отримуємо скорочену ДНФ функції $y_{\text{СДНФ}} = \bar{x}_2 x_1 \vee x_2 \bar{x}_1 \vee x_3 \bar{x}_2 \vee x_3 \bar{x}_1$, що складається з простих імплікант.

На цьому завершується перший етап мінімізації.

5. Будуємо таблицю покриття функції (табл. 3.2).

Таблиця покриття функції $y(x_3, x_2, x_1)$

	Прості імпліканти	Конституенти одиниці				
		$\bar{x}_3 \bar{x}_2 x_1$	$\bar{x}_3 x_2 \bar{x}_1$	$x_3 \bar{x}_2 \bar{x}_1$	$x_3 \bar{x}_2 x_1$	$x_3 x_2 \bar{x}_1$
1	$\bar{x}_2 x_1$	(v.)			(v)	
2	$x_2 \bar{x}_1$		(v.)			(v)
3	$x_3 \bar{x}_2$			v	v	
4	$x_3 \bar{x}_1$			(v)		v

6. На перетині рядків і стовпців таблиці покриття ставимо знак \vee у тому випадку, якщо дана проста імпліканта є власною частиною відповідної конституенти одиниці.

Наприклад, імпліканта $\bar{x}_2 x_1$ є власною частиною конституенти $\bar{x}_3 \bar{x}_2 x_1$ та конституенти $x_3 \bar{x}_2 x_1$.

7. Знаходимо ядро функції – сукупність *істотних імплікант*. Істотною називається така імпліканта, якій відповідає стовець, що містить лише одну позначку \vee ; поряд з позначкою \vee ставимо крапку.

Істотними імплікантами є $\bar{x}_2 x_1$ та $x_2 \bar{x}_1$. Отже, ядро функції: $\bar{x}_2 x_1, x_2 \bar{x}_1$.

8. Доповнюємо ядро мінімальною кількістю інших імплікант так, щоб отримати повне покриття всіх конституент одиниці перемикальної функції.

У загальному випадку можна отримати різні варіанти покриття, які будуть тупиковими ДНФ функції.

З таблиці покриття знаходимо дві ТДНФ: 1-а, 2-а та 4-та імпліканти

$$y_{ТДНФ1} = 1\text{-а} \vee 2\text{-а} \vee 4\text{-та} = \bar{x}_2 x_1 \vee x_2 \bar{x}_1 \vee x_3 \bar{x}_1;$$

1-а, 2-а та 3-тя імпліканти

$$y_{ТДНФ2} = 1\text{-а} \vee 2\text{-а} \vee 3\text{-тя} = \underbrace{\bar{x}_2 x_1 \vee x_2 \bar{x}_1}_{\text{ядро}} \vee x_3 \bar{x}_2.$$

9. Серед тупикових ДНФ знаходимо форму з мінімальною ціною. Обидві ТДНФ мають однакову ціну – 6. Відповідно й існують дві мінімальні ДНФ функції:

$$y_{МДНФ1} = \bar{x}_2 x_1 \vee x_2 \bar{x}_1 \vee x_3 \bar{x}_1, \quad (Ц = 6)$$

$$y_{МДНФ2} = \bar{x}_2 x_1 \vee x_2 \bar{x}_1 \vee x_3 \bar{x}_2. \quad (Ц = 6)$$

Будь-яку з них можна обрати як розв'язок задачі. Виберемо, наприклад, $y_{МДНФ1}$ (див. табл. 3.2).

Ціна початкової форми (ДДНФ) функції складала 15 (рис. 3.4). ▣

ДДНФ	$y_{ДДНФ} = \bar{x}_3 \bar{x}_2 \bar{x}_1 \vee \bar{x}_3 x_2 \bar{x}_1 \vee x_3 \bar{x}_2 \bar{x}_1 \vee x_3 \bar{x}_2 x_1 \vee x_3 x_2 x_1$	$C = 15$
СДНФ	$y_{СДНФ} = \bar{x}_2 x_1 \vee x_2 \bar{x}_1 \vee x_3 \bar{x}_2 \vee x_3 \bar{x}_1$	$C = 8$
ТДНФ	$y_{ТДНФ1} = \bar{x}_2 x_1 \vee x_2 \bar{x}_1 \vee x_3 \bar{x}_1$	$C = 6$
	$y_{ТДНФ2} = \bar{x}_2 x_1 \vee x_2 \bar{x}_1 \vee x_3 \bar{x}_2$	$C = 6$
МДНФ	$y_{МДНФ1} = \bar{x}_2 x_1 \vee x_2 \bar{x}_1 \vee x_3 \bar{x}_1$	$C = 6$
	$y_{МДНФ2} = \bar{x}_2 x_1 \vee x_2 \bar{x}_1 \vee x_3 \bar{x}_2$	$C = 6$

Рис. 3.4. Послідовність отримання форми перемикальної функції $y(x_3, x_2, x_1)$ з мінімальною ціною в задачі 3.1

Розв'яжемо ще одну задачу.

Задача 3.2. Дано перемикальну функцію y від 4-х змінних, подану в ДДНФ:

$$y = x_4 x_3 \bar{x}_2 \bar{x}_1 \vee x_4 x_3 x_2 \bar{x}_1 \vee \bar{x}_4 x_3 x_2 \bar{x}_1 \vee \bar{x}_4 x_3 \bar{x}_2 x_1 \vee$$

$$\vee x_4 \bar{x}_3 x_2 x_1 \vee \bar{x}_4 \bar{x}_3 x_2 x_1 \vee x_4 \bar{x}_3 x_2 \bar{x}_1 \vee \bar{x}_4 \bar{x}_3 x_2 \bar{x}_1.$$

Мінімізувати її методом Квайна.

Розв'язування.

1. Виконаємо попарне неповне склеювання тих конституент одиниці, які відрізняються одна від одної лише наявністю заперечення в одній змінних (рис. 3.5).

Конституенти одиниці	Імпліканти- терми 3-го рангу	Імпліканти-терми 2-го рангу
(1) $x_4 x_3 \bar{x}_2 \bar{x}_1$	(1-2) $x_4 x_3 \bar{x}_1$	(2-3, 7-8) $x_2 \bar{x}_1$
(2) $x_4 x_3 x_2 \bar{x}_1$	(2-3) $x_3 x_2 \bar{x}_1$	(2-7, 3-8)
(3) $\bar{x}_4 x_3 x_2 \bar{x}_1$	(2-7) $x_4 x_2 \bar{x}_1$	(5-6, 7-8) $\bar{x}_3 x_2$
(4) $\bar{x}_4 x_3 \bar{x}_2 x_1$	(3-8) $\bar{x}_4 x_2 \bar{x}_1$	(5-7, 6-8)
(5) $x_4 \bar{x}_3 x_2 x_1$	(5-6) $\bar{x}_3 x_2 x_1$	
(6) $\bar{x}_4 \bar{x}_3 x_2 x_1$	(5-7) $x_4 \bar{x}_3 x_2$	
(7) $x_4 \bar{x}_3 x_2 \bar{x}_1$	(6-8) $\bar{x}_4 \bar{x}_3 x_2$	
(8) $\bar{x}_4 \bar{x}_3 x_2 \bar{x}_1$	(7-8) $\bar{x}_3 x_2 \bar{x}_1$	

Рис. 3.5. Виконання всіх можливих операцій неповного склеювання в ДДНФ перемикальної функції $y(x_4, x_3, x_2, x_1)$

Наприклад, неповне склеювання конституент одиниці (1) $x_4 x_3 \bar{x}_2 \bar{x}_1$ та (2) $x_4 x_3 x_2 \bar{x}_1$ дає імпліканту – терм 3-го рангу (1-2) $x_4 x_3 \bar{x}_1$ (див. 2-й стовпець на рис. 3.5). Далі виконуємо можливі попарні неповні склеювання імплікант-термів 3-го рангу.

Наприклад, неповне склеювання імплікант (2-7) $x_4 x_2 \bar{x}_1$ та (3-8) $\bar{x}_4 x_2 \bar{x}_1$ дає імпліканту – терм 2-го рангу (2-7, 3-8) $x_2 \bar{x}_1$ (3-й стовпець на рис. 3.5). Такий самий результат дає склеювання імплікант (2-3) та (7-8).

Подальші склеювання імплікант неможливі.

2. Виконаємо всі можливі поглинання серед імплікант та конституент одиниці (рис. 3.6):

- імпліканта $\bar{x}_3 x_2$ поглинає такі імпліканти та конституенти одиниці: (7-8), (6-8), (5-7), (5-6), (8), (7), (6), (5);
- імпліканта $x_2 \bar{x}_1$ поглинає такі імпліканти та конституенти одиниці: (3-8), (2-7), (2-3), (3), (2);
- імпліканта (1-2) $x_4 x_3 \bar{x}_1$ поглинає конституенту одиниці (1).

3. Невикресленими на рис. 3.6. залишились терми, що є простими імплікантами. Їх диз'юнкція утворює скорочену ДНФ:

$$U_{СДНФ} = \bar{x}_4 x_3 \bar{x}_2 x_1 \vee x_4 x_3 \bar{x}_1 \vee x_2 \bar{x}_1 \vee \bar{x}_3 x_2.$$

На цьому завершується перший етап мінімізації.

4. Будуємо таблицю покриття (імплікантну матрицю) функції (табл. 3.3).
 5. На перетині рядків і стовпців ставимо позначки \vee тоді, коли відповідно проста імпліканта є власною частиною конституенти одиниці.

Конституенти одиниці	Імпліканти- терми 3-го рангу	Імпліканти-терми 2-го рангу
(1) $x_4 x_3 \bar{x}_2 \bar{x}_1$	(1-2) $x_4 x_3 \bar{x}_1$	(2-3, 7-8) $x_2 \bar{x}_1$
(2) $x_4 x_3 x_2 \bar{x}_1$	(2-3) $x_3 x_2 \bar{x}_1$	(2-7, 3-8)
(3) $\bar{x}_4 x_3 x_2 \bar{x}_1$	(2-7) $x_4 x_2 \bar{x}_1$	(5-6, 7-8) $\bar{x}_3 x_2$
(4) $\bar{x}_4 x_3 \bar{x}_2 x_1$	(3-8) $\bar{x}_4 x_2 \bar{x}_1$	(5-7, 6-8)
(5) $x_4 \bar{x}_3 x_2 x_1$	(5-6) $\bar{x}_3 x_2 x_1$	
(6) $\bar{x}_4 \bar{x}_3 x_2 x_1$	(5-7) $x_4 \bar{x}_3 x_2$	
(7) $x_4 \bar{x}_3 x_2 \bar{x}_1$	(6-8) $\bar{x}_4 \bar{x}_3 x_2$	
(8) $\bar{x}_4 \bar{x}_3 x_2 \bar{x}_1$	(7-8) $\bar{x}_3 x_2 \bar{x}_1$	

$$U_{СДНФ} = \bar{x}_4 x_3 \bar{x}_2 x_1 \vee x_4 x_3 \bar{x}_1 \vee x_2 \bar{x}_1 \vee \bar{x}_3 x_2$$

Рис. 3.6. Виконання всіх можливих операцій поглинання

Таблиця 3.3

Таблиця покриття функції $y(x_4, x_3, x_2, x_1)$

	Прості імпліканти	Конституенти одиниці											
		$x_4 x_3 \bar{x}_2 \bar{x}_1$	$x_4 x_3 x_2 \bar{x}_1$	$\bar{x}_4 x_3 x_2 \bar{x}_1$	$\bar{x}_4 x_3 \bar{x}_2 x_1$	$x_4 \bar{x}_3 x_2 x_1$	$\bar{x}_4 \bar{x}_3 x_2 x_1$	$x_4 \bar{x}_3 \bar{x}_2 \bar{x}_1$	$\bar{x}_4 \bar{x}_3 \bar{x}_2 \bar{x}_1$				
1	$\bar{x}_4 x_3 \bar{x}_2 x_1$				$\bar{x}_4 x_3 \bar{x}_2 x_1$								
2	$x_4 x_3 \bar{x}_1$	$\bar{x}_4 x_3 \bar{x}_2 \bar{x}_1$	$x_4 x_3 x_2 \bar{x}_1$										
3	$x_2 \bar{x}_1$		$x_4 x_3 x_2 \bar{x}_1$	$\bar{x}_4 x_3 x_2 \bar{x}_1$									
4	$\bar{x}_3 x_2$												

6. З таблиці покриття знаходимо ядро функції. До складу ядра входять всі прості імпліканти.
7. Знаходимо ТДНФ функції
- $$Y_{ТДНФ} = \bar{x}_4 x_3 \bar{x}_2 x_1 \vee x_4 x_3 \bar{x}_1 \vee x_2 \bar{x}_1 \vee \bar{x}_3 x_2.$$
8. Відповідно, мінімальною ДНФ є
- $$Y_{МДНФ} = \bar{x}_4 x_3 \bar{x}_2 x_1 \vee x_4 x_3 \bar{x}_1 \vee x_2 \bar{x}_1 \vee \bar{x}_3 x_2.$$
- Її ціна дорівнює 11 (ціна ДДНФ становила 32). ■

Запитання та завдання

1. У якій формі має бути подана перемикальна функція, щоб для її мінімізації можна було застосувати метод Квайна?
2. Як здійснюють перетворення ДНФ функції в ДДНФ?
3. На використанні яких двох співвідношень булевої алгебри ґрунтується метод мінімізації Квайна?
4. Сформулюйте теорему Квайна.
5. Назвіть етапи мінімізації перемикальної функції методом Квайна.
6. Яку структуру має таблиця покриття (імплікантна матриця) перемикальної функції?
7. Опишіть покроково отримання МДНФ перемикальної функції методом Квайна.

Задачі для самостійного розв'язування

1. Дано перемикальну функцію від трьох змінних:

$$y = x_3 x_2 x_1 \vee x_3 \bar{x}_1 \vee \bar{x}_3 x_2 x_1 \vee \bar{x}_3 \bar{x}_2 \bar{x}_1.$$
 Мінімізувати її методом Квайна.
2. Повністю визначена перемикальна функція від трьох змінних подана в КНФ:

$$f = (\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_1)(x_3 \vee \bar{x}_2 \vee \bar{x}_1)(x_3 \vee x_1)(\bar{x}_3 \vee x_2 \vee x_1).$$
 Знайти МДНФ функції f .
3. Функцію z подано в канонічній нормальній формі алгебри Пірса:

$$z = \overline{\overline{x_3 \vee x_2 \vee x_1 \vee x_3 \vee x_2 \vee x_1 \vee x_3 \vee x_2 \vee x_1 \vee x_3 \vee x_2 \vee x_1}}.$$
 Знайти МДНФ функції z .
4. Знайти МДНФ функції γ , якщо відома її канонічна нормальна форма в алгебрі Шеффера:

$$\gamma = \overline{\overline{\overline{x_3 x_2 x_1} \cdot \overline{\overline{x_3 x_2 x_1}} \cdot \overline{\overline{x_3 x_2 x_1}} \cdot \overline{\overline{x_3 x_2 x_1}}}}.$$

5. Перемикальна функція від трьох змінних дорівнює нулю на наборах 1, 5, 7 і одиниці – на решті наборів. Мінімізувати функцію методом Квайна.
6. Перемикальна функція від 4-х змінних дорівнює одиниці на наборах 0, 1, 2, 6, 7, 8, 10, 14, 15 і нулю – на решті наборів. Знайти МДНФ функції.
7. Відомо, що перемикальна функція y від 4-х змінних дорівнює одиниці на наборах 0, 1, 2, 3, 6, 8, 9, 15 і нулю – на решті наборів. Відома також її скорочена ДНФ:

$$y_{\text{СДНФ}} = \bar{x}_4 \bar{x}_2 \vee \bar{x}_3 \bar{x}_2 \vee \bar{x}_4 x_3 \bar{x}_1 \vee \bar{x}_3 \bar{x}_2 x_1 \vee x_4 x_3 x_2 x_1.$$
 Знайти МДНФ функції y .
8. Перемикальна функція від 4-х змінних дорівнює нулю на наборах 3, 10, 11, 12, 13, 14, 15 і одиниці – на решті наборів. Реалізувати функцію в елементному базисі 2АБО, 2І з мінімальними апаратними витратами.
9. Перемикальна функція від 4-х змінних дорівнює одиниці на наборах 0, 2, 3, 4, 6, 8, 10, 11, 12 і нулю – на решті наборів. Реалізувати функцію в елементному базисі 2І-НЕ з мінімальними апаратними витратами.

3.3. Метод мінімізації Квайна – Мак-Класкі

Особливістю методу Квайна мінімізації перемикальних функцій є те, що на етапі визначення простих імплікант необхідно попарно порівнювати між собою всі терми, що беруть участь у запису функції – спочатку конституенти одиниці (терми n -рангу), а потім терми (імпліканти) $(n-1)$ -го, $(n-2)$ -го рангу і т.д. Зі зростанням кількості конституент одиниці в ДДНФ функції зростає кількість таких порівнянь.

Крім того, із зростанням величини n – місності функції, незручною для роботи є алгебраїчна форма запису функції у вигляді диз'юнкції термів.

Мак-Класкі запропонував модернізацію першого етапу процесу мінімізації за методом Квайна (етапу, на якому знаходять скорочену ДНФ), спрямовану на істотне скорочення кількості порівнянь термів між собою та на спрощення роботи з термами за рахунок використання цифрової, а не алгебраїчної форми запису термів. При цьому другий етап в методі Квайна (етап визначення тупикових ДНФ функції та вибору серед них МДНФ функції) пропонувалось залишити без змін.

Модернізований в такий спосіб процес мінімізації перемикальних функцій дістав назву методу Квайна – Мак-Класкі.

Мак-Класкі запропонував записувати терми у цифровому вигляді.

Наприклад, функція $f(x_3, x_2, x_1) = \bar{x}_3 \bar{x}_2 \bar{x}_1 \vee x_3 \bar{x}_2 x_1 \vee x_3 x_2 x_1$ може бути записана так: $f(x_3, x_2, x_1) = 000 \vee 101 \vee 111$, де 0 на i -й позиції ($i = 3, 2, 1$) в термі позначає змінну \bar{x}_i , а 1 – змінну x_i .

Цифрова форма скорочує довжину запису та істотно полегшує виконання операцій над термами.

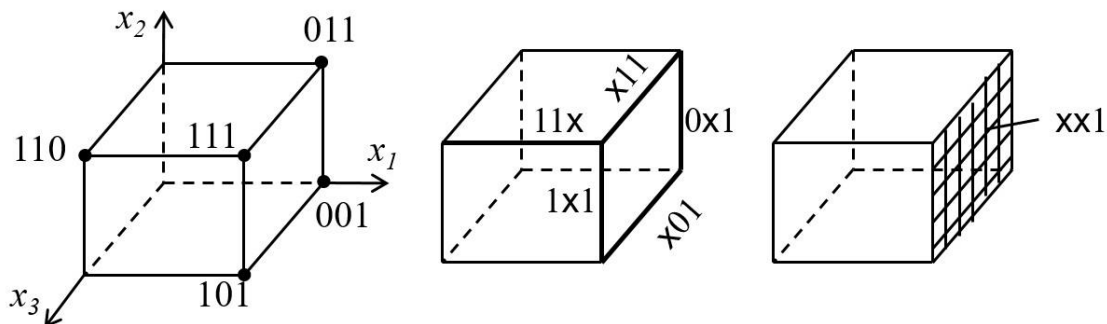
Метод Квайна – Мак-Класкі також (як і метод Квайна) ґрунтується на застосуванні операції неповного склеювання та операції поглинання.

Особливість знаходження простих імплікант в методі мінімізації Квайна – Мак-Класкі пояснимо за допомогою геометричної інтерпретації подання перемикальних функцій.

Нехай ДДНФ функції від 3-х змінних має вигляд:
 $Y = \bar{x}_3 \bar{x}_2 \bar{x}_1 \vee \bar{x}_3 x_2 x_1 \vee x_3 \bar{x}_2 x_1 \vee x_3 x_2 x_1$.

Такій функції відповідає тривимірний клуб на рис. 3.7.

$$y = \bar{x}_3 \bar{x}_2 \bar{x}_1 \vee \bar{x}_3 x_2 x_1 \vee x_3 \bar{x}_2 x_1 \vee x_3 x_2 \bar{x}_1 \vee x_3 x_2 x_1 = 001 \vee 011 \vee 101 \vee 110 \vee 111$$



<p><u>0-куби</u> (терми 3-го рангу), вершини куба: $1\ 1\ 0 \rightarrow x_3 x_2 \bar{x}_1$ $1\ 1\ 1 \rightarrow x_3 x_2 x_1$ $1\ 0\ 1 \rightarrow x_3 \bar{x}_2 x_1$ $0\ 1\ 1 \rightarrow \bar{x}_3 x_2 x_1$ $0\ 0\ 1 \rightarrow \bar{x}_3 \bar{x}_2 x_1$</p>	<p><u>1-куби</u> (терми 2-го рангу), ребра куба: $1\ 1\ \times \rightarrow x_3 x_2$ $\times\ 1\ 1 \rightarrow x_2 x_1$ $1\ \times\ 1 \rightarrow x_3 x_1$ $0\ \times\ 1 \rightarrow \bar{x}_3 x_1$ $\times\ 0\ 1 \rightarrow \bar{x}_2 x_1$</p>	<p><u>2-куби</u> (терми 1-го рангу), грані куба: $\times \times 1 \rightarrow x_1$</p>
<p>Конституенти одиниці – терми 3-го рангу</p>	<p>Імпліканти – терми 2-го рангу</p>	<p>Імпліканта – терм 1-го рангу</p>

Рис. 3.7. Геометрична інтерпретація знаходження імплікант функції y

Конституентами одиниці (термам 3-го рангу), які входять до складу ДДНФ, відповідають вершини куба з координатами 001, 011, 101, 110, 111.

Для функції від 3-х змінних визначимо правила склеювання та поглинання термів.

Правило 1 (для вершин). Дві вершини, що належать одному й тому самому ребру, склеюються за змінною (координатою), яка змінюється вздовж цього ребра.

Наприклад, для сусідніх вершин 110 і 111 операцію неповного склеювання записують у вигляді

$$110 \vee 111 = 110 \vee 111 \vee 11 \times,$$

де символ \times позначає змінну, за якою склеюють терми. Цей цифровий запис відповідає алгебраїчному виразу неповного склеювання

$$x_3 x_2 \bar{x}_1 \vee x_3 x_2 x_1 = x_3 x_2 \bar{x}_1 \vee x_3 x_2 x_1 \vee x_3 x_2.$$

Після операції поглинання $110 \vee 111 \vee 11 \times = 11 \times$ замість двох вершин отримуємо ребро $11 \times$, якому відповідає терм – імпліканта 2-го рангу $x_3 x_2$.

Правило 2 (для ребер). Два ребра, що належать одній і тій самій грані куба й відрізняються лише однією координатою, склеюють за змінною, яка відповідає цій координаті.

Наприклад, виконавши поетапно операції неповного склеювання та поглинання над ребрами $\times 11$ та $\times 01$, отримуємо грань $\times \times 1$, якій відповідає терм – імпліканта 1-го рангу x_1 :

$$\text{склеювання: } \times 11 \vee \times 01 = \times 11 \vee \times 01 \vee \times \times 1,$$

$$\text{поглинання: } \times 11 \vee \times 01 \vee \times \times 1 = \times \times 1.$$

Узагальнимо геометричну інтерпретацію знаходження імплікант на випадок перемикальної функції від n змінних.

Кожен набір значень аргументів $(x_n, \dots, x_2, x_1) \in n$ -вимірним вектором і визначає точку n -вимірного простору. Сукупність усіх наборів, на яких визначена перемикальна функція від n змінних, зображується n -вимірним кубом. Конституентами одиниці відповідають вершини куба, а імплікантам – ребра і грані.

Терми максимального рангу (n -го рангу), які відповідають вершинам n -вимірного куба, називають 0-кубами (їх сукупність позначають K^0 і називають комплексом 0-кубів); терми $(n-1)$ -го рангу називають 1-кубами (їх сукупність позначають K^1 і називають комплексом 1-кубів); терми $(n-2)$ -го рангу називають 2-кубами (їх сукупність позначають K^2 і називають комплексом 2-кубів) і т.д.

Об'єднання комплексів K^r r -кубів ($r = 0, 1, 2, \dots, m$), де $m \leq n-1$,

утворює комплекс $K = \bigcup_{r=0}^m K^r$, до складу якого входять конституенти

одиниці та всі імпліканти перемикальної функції.

Розглянемо алгоритми знаходження імплікант перемикальної функції у цифровій формі, що ґрунтується на неповному склеюванні r -кубів ($r = 0, 1, 2, \dots, n - 1$).

1. На основі ДДНФ або таблиці істинності записати перемикальну функцію у цифровій формі. Виписати 0-куби (конституенти одиниці) та впорядкувати їх за кількістю одиниць:

- 0-куб без одиниць (якщо ДДНФ містить терм $\bar{x}_n \bar{x}_{n-1} \dots \bar{x}_1$);
- групу 0-кубів, що містять одну одиницю;
- групу 0-кубів, що містять дві одиниці;
- групу 0-кубів, що містять три одиниці; і т.д.

Групи відділити одна від одної пунктирною лінією.

Результатом є комплекс K^0 0-кубів.

2. Виконати в комплексі K^0 всі можливі склеювання 0-кубів, що належать сусіднім групам. Оскільки 0-куби поділено на групи з однаковою кількістю одиниць, то склеювання можливі лише між кубами сусідніх груп, а всередині груп та між віддаленими (несусідніми) групами – неможливі. Саме в цьому полягає ідея Мак-Класкі, бо при такому впорядкуванні зменшується кількість порівнянь термів (кубів) між собою. Результатом є комплекс 1-кубів (K^1).

3. Упорядкувати комплекс K^1 поділяючи терми на групи з однаковою кількістю одиниць. Виконати в комплексі K^1 1-кубів всі можливі склеювання кубів, що належать сусіднім групам. Результатом є комплекс 2-кубів (K^2).

4. Виконати в комплексі K^2 всі можливі склеювання.

І так далі поки можливі склеювання.

Результатом роботи алгоритму є знаходження всіх імплікант перемикальної функції.

Роботу алгоритму демонструє рис. 3.8а, на якому показано процес формування комплексів K^r r -кубів заданої функції від 3-х змінних.

Метод мінімізації Квайна – Мак-Класкі перемикальних функцій полягає в наступному.

1. Для заданої перемикальної функції виконати всі можливі операції склеювання термів, використовуючи цифрову форму їх запису (за наведеним вище алгоритмом формування комплексів K^r r -кубів).
2. В отриманих комплексах K^r r -кубів виконати всі можливі поглинання (терми, що поглинаються, викреслюємо). Результатом є множина невикреслених кубів (термів), яку називають Z -покриттям функції.

Куби в Z -покритті відповідають усім простим імплікантам функції.

Диз'юнкція кубів Z -покриття дає скорочену ДНФ функції.

$$y = x_3 x_2 \bar{x}_1 \vee x_3 x_2 x_1 \vee x_3 \bar{x}_2 x_1 \vee \bar{x}_3 x_2 x_1 \vee \bar{x}_3 \bar{x}_2 x_1 = 110 \vee 111 \vee 101 \vee 011 \vee 001$$

$$K^0 = \left\{ \begin{array}{l} \underline{(1)} \underline{001} \\ (2) \ 011 \\ (3) \ 101 \\ \underline{(4)} \underline{110} \\ (5) \ 111 \end{array} \right\} \quad K^1 = \left\{ \begin{array}{l} (1-2) \ 0 \times 1 \\ \underline{(1-3)} \underline{\times 01} \\ (2-5) \times 11 \\ (3-5) \ 1 \times 1 \\ (4-5) \ 11 \times \end{array} \right\} \quad K^2 = \left\{ \begin{array}{l} (1-2, 3-5) \ \times \times 1 \\ (1-3, 2-5) \end{array} \right\}$$

K^0 – множина конститuent одиниці (комплекс 0-кубів), термів 3-го рангу,
 K^1 – множина імплікант 2-го рангу (комплекс 1-кубів),
 K^2 – множина імплікант 1-го рангу (комплекс 2-кубів),
 $K = K^0 \cup K^1 \cup K^2$ – множина конститuent одиниці та всіх імплікант
 перемикальної функції

a

$$K^0 = \left\{ \begin{array}{l} \underline{(1)} \underline{001} \\ (2) \ 011 \\ (3) \ 101 \\ \underline{(4)} \underline{110} \\ (5) \ 111 \end{array} \right\} \quad K^1 = \left\{ \begin{array}{l} (1-2) \ 0 \times 1 \\ \underline{(1-3)} \underline{\times 01} \\ (2-5) \times 11 \\ (3-5) \ 1 \times 1 \\ (4-5) \ 11 \times \end{array} \right\} \quad K^2 = \left\{ \begin{array}{l} \times \times 1 \\ 11 \times \end{array} \right\} \quad \Rightarrow \quad Z = \left\{ \begin{array}{l} \times \times 1 \\ 11 \times \end{array} \right\}$$

Прості імпліканти
 $\times \times 1 \rightarrow x_1$
 $11 \times \rightarrow x_3 x_2$

$$y_{\text{сднф}} = \times \times 1 \vee 11 \times = x_1 \vee x_3 x_2$$

б

Рис. 3.8. Знаходження імплікант (а) перемикальної функції у та скороченої ДНФ функції (б)

3. Побудувати таблицю покриття перемикальної функції.
 4. Визначити тупикові ДНФ.
 5. Серед ТДНФ знайти МДНФ функції.
- } як у методі Квайна

На рис. 3.8 показано перший етап процесу мінімізації функції у методом Квайна – Мак-Класкі. Після виконання всіх можливих операцій неповного склеювання термів (рис. 3.8а) отримуємо всі імпліканти функції.

Наприклад, склеювання 0-куба (1) 001 (перша група) з 0-кубами (2) та (3) другої (сусідньої) групи комплексу K^0 дає такі результати:

$$001 \vee 011 = 001 \vee 011 \vee 0 \times 1 \rightarrow (1-2) 0 \times 1,$$

$$001 \vee 101 = 001 \vee 101 \vee \times 01 \rightarrow (1-3) \times 01;$$

а склеювання 1-куба (1-2) 0×1 (перша група комплексу K^1) з 1-кубом (3-5) 1×1 (сусідня група) породжує 2-куб (1-2, 3-5) $\times \times 1$:

$$0 \times 1 \vee 1 \times 1 = 0 \times 1 \vee 1 \times 1 \vee \times \times 1.$$

Після виконання всіх можливих поглинань термів отримаємо скорочену ДНФ функції.

Так, 2-куб $\{\times \times 1\}$ з комплексу K^2 поглинає всі терми, що містять у молодшому розряді (справа) одиницю, а 1-куб (4-5) $11 \times$ поглинає терм (4) 110 .

Непоглинутими залишаються 2-куб $\{\times \times 1\}$ та 1-куб $\{1 \times \times\}$. Вони утворюють Z-покриття функції. Куби в Z-покритті є простими імплікантами функції, їх диз'юнкція є скороченою ДНФ функції у:

$$U_{\text{сднф}} = \times \times 1 \vee 11 \times = x_1 \vee x_3 x_2.$$

Розв'яжемо задачу.

Задача 3.3. Виконати мінімізацію заданої в ДДНФ перемикальної функції від 3-х змінних

$$y = \bar{x}_3 \bar{x}_2 \bar{x}_1 \vee \bar{x}_3 \bar{x}_2 x_1 \vee \bar{x}_3 x_2 x_1 \vee x_3 \bar{x}_2 \bar{x}_1 \vee x_3 \bar{x}_2 x_1 \vee x_3 x_2 \bar{x}_1$$

методом Квайна – Мак-Класкі.

Розв'язування.

1. Запишемо функцію у цифровій формі

$y = 000 \vee 001 \vee 011 \vee 100 \vee 101 \vee 110$ та сформуємо комплекс K^0 0-кубів (конституент одиниці функції) (рис. 3.9). Для цього двійкові набори, на яких функція набуває значення одиниці, розіб'ємо на групи за кількістю одиниць – набір 000, що не містить одиниць, набори 001 та 100, що містять по одній одиниці, та набори 011, 101, 110, до складу яких входять по дві одиниці. Сформовані групи наборів відділимо одна від одної пунктирною лінією. Упорядковані в такий спосіб набори занумеруємо – від (1) до (6).

2. Сформуємо комплекс K^1 1-кубів.

Для цього спочатку виконаємо можливі склеювання 0-кубів, що належать сусіднім групам. Так, 0-куб $\{000\}$ (перша група) склеюємо з 0-кубами $\{001\}$ та $\{100\}$, що належать другій групі:

$$000 \vee 001 = 000 \vee 001 \vee 00 \times,$$

$$000 \vee 100 = 000 \vee 100 \vee \times 00.$$

$$y = 000 \vee 001 \vee 011 \vee 100 \vee 101 \vee 110$$

$$K^0 = \left\{ \begin{array}{l} (1) \underline{000} \\ (2) 001 \\ (3) \underline{100} \\ (4) 011 \\ (5) 101 \\ (6) 110 \end{array} \right\} \quad K^1 = \left\{ \begin{array}{l} (1-2) 00\times \\ (1-3) \underline{\times 00} \\ (2-4) 0\times 1 \\ (2-5) \times 01 \\ (3-5) 10\times \\ (3-6) 1\times 0 \end{array} \right\} \quad K^2 = \left\{ \begin{array}{l} (1-2, 3-5) \times 0\times \\ (1-3, 2-5) \end{array} \right\}$$

Рис. 3.9. Формування комплексів (імплікант) K^r r -кубів ($r = 0, 1, 2$) перемикальної функції $y = 0 \vee 1 \vee 3 \vee 4 \vee 5 \vee 6$

Результатом склеювання є 1-куби $\{00\times\}$ та $\{\times 00\}$, які занумеруємо як (1-2) та (1-3) відповідно (див. рис. 3.9).

Можливими склеюваннями 0-куба $\{001\}$, що належить другій групі, з 0-кубами третьої групи є:

$$001 \vee 011 = 001 \vee 011 \vee 0\times 1,$$

$$001 \vee 101 = 001 \vee 101 \vee \times 01$$

(результат – 1-куби $\{0\times 1\}$ та $\{\times 01\}$ з номерами (2-4) та (2-5) відповідно), а 0-куба $\{100\}$, що входить до другої групи, з 0-кубами третьої групи –

$$100 \vee 101 = 100 \vee 101 \vee 10\times,$$

$$100 \vee 110 = 100 \vee 110 \vee 1\times 0$$

(результатом є 1-куби $\{10\times\}$ та $\{1\times 0\}$, занумеровані як (3-5) та (3-6)). Комплекс K^1 1-кубів (рис. 3.9) складається з двох груп.

3. Сформуємо комплекс K^2 2-кубів. Для цього в комплексі K^1 виконаємо можливі склеювання:

$$(1-2, 3-5) 00\times \vee 10\times = 00\times \vee 10\times \vee \times 0\times,$$

$$(1-3, 2-5) \times 00 \vee \times 01 = \times 00 \vee \times 01 \vee \times 0\times.$$

Оскільки обидва склеювання дали однаковий результат, то до складу комплексу K^2 входить лише один 2-куб – $\{\times 0\times\}$ (рис. 3.9).

Подальші склеювання неможливі.

4. В отриманих комплексах K^0 , K^1 , K^2 виконаємо всі можливі поглинання (рис. 3.10). Для цього:

- а) беремо 2-куб $\{\times 0\times\}$ з K^2 і почергово порівнюємо його з усіма 1-кубами комплексу K^1 та 0-кубами комплексу K^0 ; він поглинає 1-куби $\{10\times\}$, $\{\times 01\}$, $\{\times 00\}$ та $\{00\times\}$, які викреслюємо з K^1 , а також 0-куби $\{101\}$, $\{100\}$, $\{001\}$ та $\{000\}$, які викреслюємо з K^0 ;
- б) серед невикреслених 1-кубів з K^1 беремо 1-куб $\{1\times 0\}$, він поглинає 0-куб $\{110\}$, який викреслюємо з K^0 ; далі беремо 1-куб $\{0\times 1\}$, він поглинає 0-куб $\{011\}$, який також викреслюємо з K^0 .

5. Сформуємо Z-покриття функції.

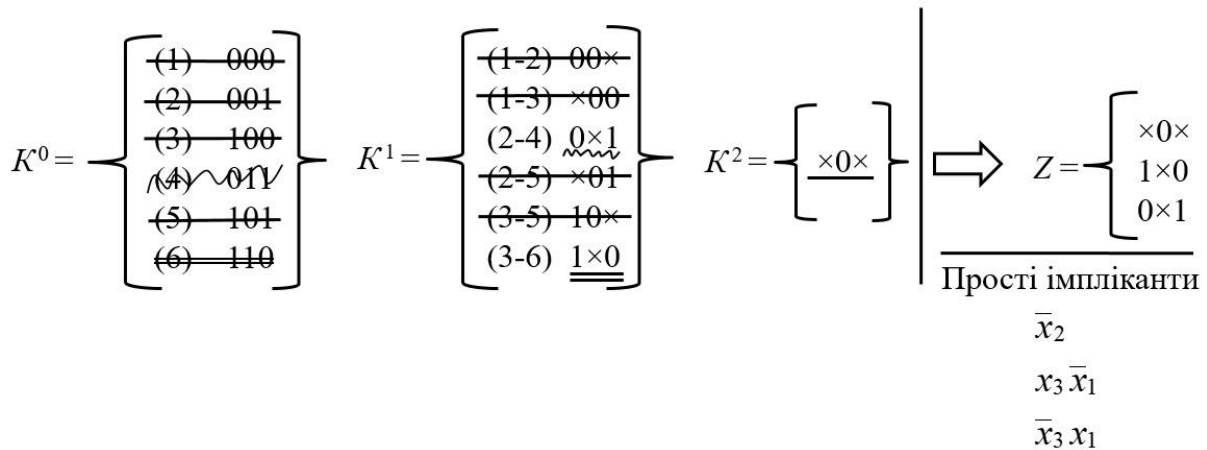
Непоглинутими залишились два 1-куби – $\{0 \times 1\}$, $\{1 \times 0\}$, та один 2-куб – $\{ \times 0 \times \}$. Вони й утворюють Z-покриття функції y (рис. 3.10).

Куби в Z-покритті відповідають простим імплікантам функції y .

Таким чином, скорочена ДНФ функції має вигляд:

$$y_{\text{СДНФ}} = \times 0 \times \vee 1 \times 0 \vee 0 \times 1 \text{ або } y_{\text{СДНФ}} = \bar{x}_2 \vee x_3 \bar{x}_1 \vee \bar{x}_3 x_1.$$

На цьому завершується перший етап мінімізації.



$$y_{\text{СДНФ}} = \times 0 \times \vee 1 \times 0 \vee 0 \times 1 = \bar{x}_2 \vee x_3 \bar{x}_1 \vee \bar{x}_3 x_1$$

Рис. 3.10. Формування Z-покриття функції $y = 0 \vee 1 \vee 3 \vee 4 \vee 5 \vee 6$

6. Для вилучення надлишкових імплікант зі скороченої ДНФ та отримання МДНФ будемо таблицю покриття функції y (табл. 3.4).

Таблиця 3.4

Таблиця покриття функції y

	Прості імпліканти	Конституенти одиниці					
		000	001	011	100	101	110
1	$\times 0 \times$	(v.)	(v)		(v)	(v.)	
2	1×0				v		(v.)
3	0×1		v	(v.)			

Проста імпліканта $\{ \times 0 \times \}$ є власною частиною стовпців 000, 001, 100 та 101 таблиці; друга імпліканта ($\{ 1 \times 0 \}$) – стовпців 100 та 110; третя ($\{ 0 \times 1 \}$) – стовпців 001 та 011.

З табл. 3.4 бачимо, що всі імпліканти є істотними, і, отже, всі три є ядром функції. Вони утворюють єдину тупикову ДНФ, і, отже, єдину МДНФ функції $y_{МДНФ} = x_0 \times \vee 1 \times 0 \vee 0 \times 1 = \bar{x}_2 \vee x_3 \bar{x}_1 \vee \bar{x}_3 x_1$.

Ціна МДНФ – 5, ДДНФ – 18. ■

Отже, метод мінімізації Квайна – Мак-Класкі має наступні переваги.

1. Спрощується робота з термами – за рахунок використання цифрової, а не алгебраїчної форми запису термів.
Цифрова форма скорочує довжину запису та полегшує виконання операцій над термами.
2. Прискорюється пошук скороченої ДНФ перемикальної функції, оскільки завдяки упорядкуванню термів за кількістю одиниць у порівняннях беруть участь не всі терми.

Запитання та завдання

1. Назвіть недоліки методу мінімізації Квайна.
2. Які новації запровадив Мак-Класкі в методі мінімізації Квайна – Мак-Класкі?
3. Які переваги має метод мінімізації Квайна – Мак-Класкі порівняно з методом Квайна?
4. За рахунок чого в методі Квайна – Мак-Класкі скорочується кількість порівнянь термів між собою?
5. Запишіть функцію $y(x_2, x_1) = x_2 x_1 \vee \bar{x}_2 \bar{x}_1$ у цифровій формі.
6. На застосуванні яких операцій булевої алгебри ґрунтується метод мінімізації Квайна – Мак-Класкі?
7. Застосуйте правило неповного склеювання у цифровій формі для таких пар термів: $\bar{x}_3 x_2 \bar{x}_1$ і $\bar{x}_3 x_2 x_1$, $\bar{x}_3 \bar{x}_2 x_1$ і $\bar{x}_3 x_2 x_1$, $\bar{x}_4 x_3 x_2 \bar{x}_1$ і $\bar{x}_4 \bar{x}_3 \bar{x}_2 \bar{x}_1$.
8. Виконайте операцію поглинання у цифровій формі для таких груп термів:
 - а) $x_3 x_2 \bar{x}_1$, $x_3 \bar{x}_1$, $x_3 \bar{x}_2 \bar{x}_1$
 - б) $x_4 \bar{x}_3 \bar{x}_2 x_1$, $x_4 x_3 \bar{x}_2 x_1$, $x_4 \bar{x}_2 x_1$.
9. Назвіть етапи мінімізації перемикальної функції методом Квайна – Мак-Класкі.
10. Опишіть покроково процес отримання МДНФ перемикальної функції методом Квайна – Мак-Класкі.

Задачі для самостійного розв'язування

1. Дано перемикальну функцію від трьох змінних:
 $\alpha_{\text{ДНФ}} = \bar{x}_3 \bar{x}_2 x_1 \vee \bar{x}_3 x_2 \bar{x}_1 \vee x_3 \bar{x}_2 \bar{x}_1 \vee x_3 x_2 \bar{x}_1$.
Мінімізувати її методом Квайна – Мак-Класкі.
2. Дано перемикальну функцію β від трьох змінних:
 $\beta = (\bar{x}_3 \vee x_2 \vee \bar{x}_1)(x_3 \vee x_1)(x_3 \vee \bar{x}_2 \vee \bar{x}_1)$.
Знайти МДНФ функції β методом Квайна – Мак-Класкі.
3. Виконати мінімізацію функції від чотирьох змінних методом Квайна – Мак-Класкі, якщо відомо, що вона дорівнює одиниці на наборах 3, 4, 5, 6, 7, 10, 12, 13 та нулю – на решті наборів.
4. Перемикальна функція γ від 4-х змінних дорівнює нулю на наборах 0, 1, 6, 8, 9, 12, 14 та одиниці – на решті наборів. Відома також її скорочена ДНФ у цифровій формі:
 $\gamma_{\text{СДНФ}} = \times 01 \times \vee \times \times 11 \vee \times 1 \times 1 \vee 010 \times$.
Знайти $\gamma_{\text{МДНФ}}$, застосувавши метод мінімізації Квайна – Мак-Класкі.
5. Перемикальна функція від 4-х змінних дорівнює одиниці на наборах 2, 4, 5, 11, 12, 13, 14, 15 та нулю – на решті наборів. Мінімізувати функцію методом Квайна – Мак-Класкі. Отриману мінімальну форму функції подати в операторних формах, придатних для реалізації на логічних елементах 2АБО, 2І-НЕ.
6. Перемикальна функція від 4-х змінних дорівнює нулю на наборах 1, 4, 5, 7, 8, 9, 12, 15 та одиниці – на решті наборів. Знайти МДНФ функції методом Квайна – Мак-Класкі. Отриману мінімальну форму функції подати в операторній формі, придатній для реалізації в елементному базисі 3АБО-НЕ, 2АБО, за умови, що для побудови комбінаційної схеми мають використовуватись обидва типи заданих логічних елементів.

3.4. Спосіб Петріка знаходження тупикових ДНФ перемикальної функції

Будь-який з відомих аналітичних методів мінімізації перемикальних функцій (рис. 3.11) процедурно складається з двох етапів – етапу визначення скороченої ДНФ та етапу знаходження тупикових ДНФ і вибору серед них мінімальної ДНФ.

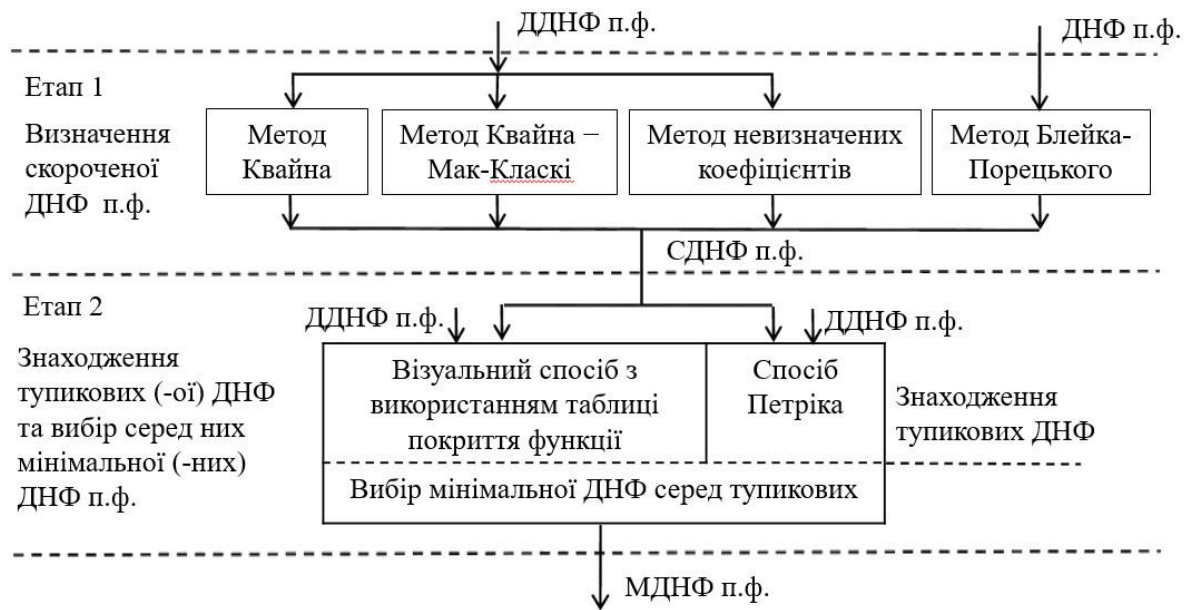


Рис. 3.11. Класифікація аналітичних методів мінімізації перемикальних функцій

Особливістю етапу 2 є те, що знаходження тупикових ДНФ можна здійснювати двома способами – *візуальним*, який ґрунтується на використанні таблиці покриття (імплікантної матриці) функції, або *аналітичним*, який називають способом Петріка (Petrick algorithm).

Узагальнимо візуальний спосіб, який використовувався при розгляді методів мінімізації Квайна та Квайна – Мак-Класкі. За цим способом для відшукування тупикових ДНФ будують таблицю покриття функції, стовпцями якої є всі конституенти одиниці, а рядками – всі прості імпліканти функції.

На перетині рядків і стовпців таблиці покриття ставлять знак \vee у тому випадку, якщо дана проста імпліканта є власною частиною відповідної конституенти одиниці. Після цього реалізують процедуру пошуку можливих тупикових ДНФ. Спочатку візуально знаходять істотні імпліканти – на місцезнаходження в рядках таблиці істотних імплікант вказують ті стовпці таблиці покриття, які містять лише один знак \vee . Диз'юнкція істотних імплікант утворює ядро функції. Знаходження тупикових ДНФ здійснюють візуально – долучають до ядра функції мінімальну кількість інших простих імплікант так, щоб разом з ядром вони покрили всі стовпці таблиці.

Зрозуміло, що такий неформалізований (суто візуальний) спосіб знаходження ТДНФ може призвести до неоптимального визначення тупикових ДНФ. Крім того, якщо таблиця покриття функції має велику розмірність, то визначити візуально всі ТДНФ складно.

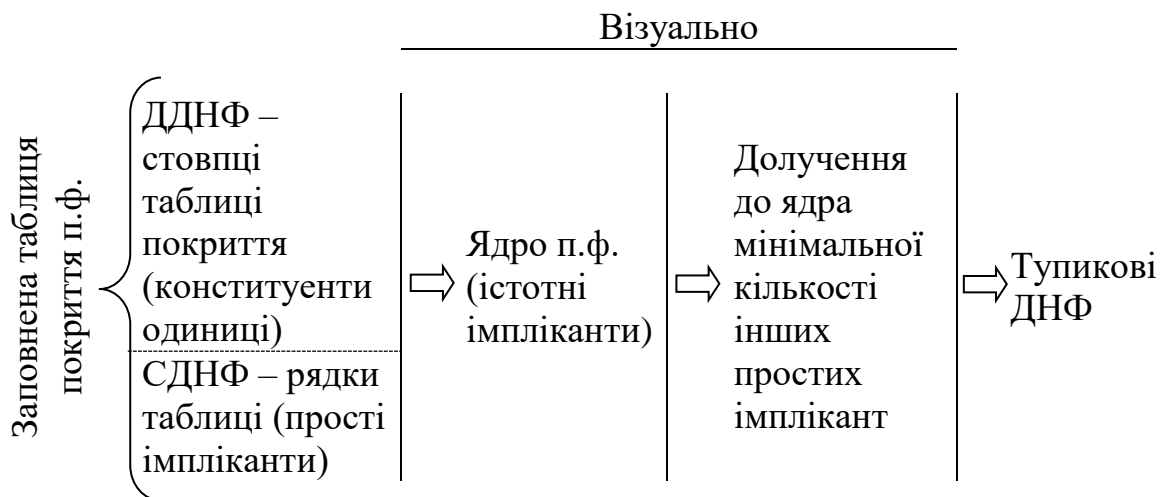
Другий спосіб (спосіб Петріка), є аналітичним, і дозволяє формалізувати процес знаходження всіх тупикових ДНФ перемикальної функції (рис. 3.12).

Його сутність полягає в наступному.

Вхідними для способу Петріка є ДДНФ та СДНФ перемикальної функції.

1. Формуємо множину KONST конститuent одиниці, які входять до складу ДДНФ, та множину IMPL простих імплікант, які входять до складу СДНФ функції. Кожній простій імпліканті з множини IMPL ставимо у відповідність довільний символ, наприклад, букву латинської абетки (рекомендується використовувати букви А, В, С і т.д.).

ВІЗУАЛЬНИЙ



АНАЛІТИЧНИЙ (спосіб Петріка)

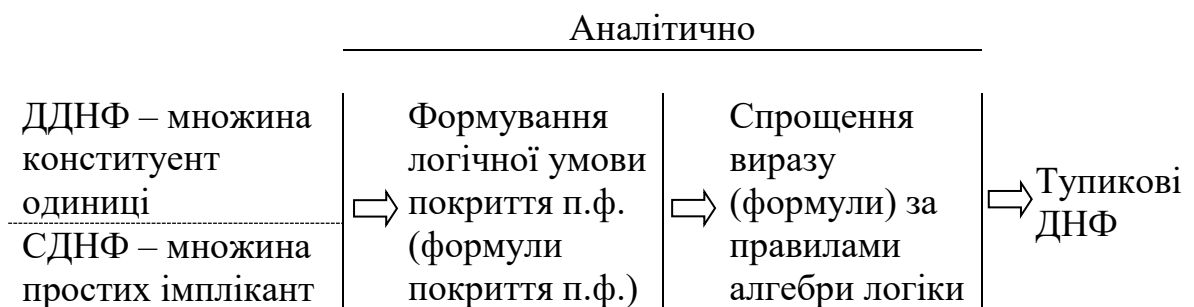


Рис. 3.12. Порівняння візуального та аналітичного способів визначення тупикових ДНФ

2. Формуємо логічні умови покриття конститuentи одиниці з множини KONST. Для цього зафіксуємо першу конститuentу одиниці з множини KONST та по чергово перевіряємо, чи є імпліканти з множини IMPL власною частиною цієї конститuentи одиниці. Записуємо логічний вираз як диз'юнкцію тих латинських літер, яким відповідають прості імпліканти, що є власною частиною зафіксованої конститuentи одиниці.

Наприклад, якщо перша та третя прості імпліканти з множини IMPL є власною частиною конститuentи одиниці (а інші прості імпліканти власною частиною конститuentи не є), то логічна умова покриття зафіксованої конститuentи одиниці має вигляд: $A \vee C$.

Таку процедуру слід виконати для кожної конститuentи одиниці з множини KONST.

3. Записуємо логічну умову покриття перемикальної функції. Для цього необхідно записати кон'юнкцію логічних умов покриття всіх конститuent одиниці функції, отриманий логічний вираз буде кон'юнкцією диз'юнкцій.
4. Перетворюємо логічний вираз так, щоб він набув вигляду диз'юнкції кон'юнкцій. Для цього необхідно розкрити дужки та скористатися аксіомою повторення ($a \cdot a = a$) та правилом поглинання ($(a \vee \beta)a = a$).
5. Формуємо тупикові ДНФ. У логічному виразі, що є диз'юнкцією кон'юнкцій, кожному кон'юнктивному терму відповідає тупикова ДНФ, вона дорівнює диз'юнкції латинських букв, що входять до складу терма, а, отже, диз'юнкції відповідних буквам простих імплікант.
6. Серед отриманих ТДНФ вибираємо форму з найменшою ціною, вона й буде МДНФ перемикальної функції.

Задача 3.4.

Дано ДДНФ перемикальної функції у від 4-х змінних:

$$y_{\text{ДДНФ}} = 0 \vee 1 \vee 2 \vee 6 \vee 9 \vee 10 \vee 11 \vee 14.$$

Виконати мінімізацію функції у методом Квайна, при цьому для визначення тупикових ДНФ функції застосувати спосіб Петріка.

Розв'язування.

1. Виконаємо неповне склеювання всіх можливих пар конститuent одиниці, які відрізняються одна від одної лише наявністю заперечення в одній зі змінних. Результатом є стовпець імплікант – термів 3-го рангу (рис. 3.13). Далі виконаємо неповне склеювання можливих пар імплікант – термів 3-го рангу. Результатом є

Конституенти одиниці	Імпліканти- терми 3-го рангу	Імпліканти-терми 2-го рангу
(1) $\bar{x}_4 \bar{x}_3 \bar{x}_2 \bar{x}_1$	(1-2) $\bar{x}_4 \bar{x}_3 \bar{x}_2$	(3-4, 6-8) $x_2 \bar{x}_1$
(2) $\bar{x}_4 \bar{x}_3 \bar{x}_2 \bar{x}_1$	(1-3) $\bar{x}_4 \bar{x}_3 \bar{x}_1$	(3-6, 4-8)
(3) $\bar{x}_4 \bar{x}_3 \bar{x}_2 \bar{x}_1$	(2-5) $\bar{x}_3 \bar{x}_2 \bar{x}_1$	
(4) $\bar{x}_4 \bar{x}_3 \bar{x}_2 \bar{x}_1$	(3-4) $\bar{x}_4 \bar{x}_2 \bar{x}_1$	
(5) $\bar{x}_4 \bar{x}_3 \bar{x}_2 \bar{x}_1$	(3-6) $\bar{x}_3 \bar{x}_2 \bar{x}_1$	
(6) $\bar{x}_4 \bar{x}_3 \bar{x}_2 \bar{x}_1$	(4-8) $\bar{x}_3 \bar{x}_2 \bar{x}_1$	
(7) $\bar{x}_4 \bar{x}_3 \bar{x}_2 \bar{x}_1$	(5-7) $x_4 \bar{x}_3 \bar{x}_1$	
(8) $\bar{x}_4 \bar{x}_3 \bar{x}_2 \bar{x}_1$	(6-7) $x_4 \bar{x}_3 \bar{x}_2$	
	(6-8) $\bar{x}_4 \bar{x}_2 \bar{x}_1$	

$$Y_{\text{СДНФ}} = x_2 \bar{x}_1 \vee x_4 \bar{x}_3 \bar{x}_2 \vee x_4 \bar{x}_3 \bar{x}_1 \vee \bar{x}_3 \bar{x}_2 \bar{x}_1 \vee \bar{x}_4 \bar{x}_3 \bar{x}_1 \vee \bar{x}_4 \bar{x}_3 \bar{x}_2$$

Рис. 3.13. Визначення скороченої ДНФ перемикальної функції у

імпліканта – терм 2-го рангу $x_2 \bar{x}_1$. Подальше склеювання імплікант неможливе.

- Виконаємо всі можливі поглинання серед імплікант та конституент одиниці (див. рис. 3.13). Наслідком цього є отримання скороченої ДНФ функції:

$$Y_{\text{СДНФ}} = x_2 \bar{x}_1 \vee x_4 \bar{x}_3 \bar{x}_2 \vee x_4 \bar{x}_3 \bar{x}_1 \vee \bar{x}_3 \bar{x}_2 \bar{x}_1 \vee \bar{x}_4 \bar{x}_3 \bar{x}_1 \vee \bar{x}_4 \bar{x}_3 \bar{x}_2.$$

- Сформуємо множину KONST конституент одиниці, які входять до складу ДДНФ

$$KONST = \left\{ \begin{array}{ccccc} \textcircled{1} & \textcircled{2} & \textcircled{3} & \textcircled{4} & \textcircled{5} \\ \bar{x}_4 \bar{x}_3 \bar{x}_2 \bar{x}_1, & \bar{x}_4 \bar{x}_3 \bar{x}_2 x_1, & \bar{x}_4 \bar{x}_3 x_2 \bar{x}_1, & \bar{x}_4 \bar{x}_3 x_2 x_1, & x_4 \bar{x}_3 \bar{x}_2 x_1, \\ \textcircled{6} & \textcircled{7} & \textcircled{8} & & \\ x_4 \bar{x}_3 x_2 \bar{x}_1, & x_4 \bar{x}_3 x_2 x_1, & x_4 x_3 x_2 \bar{x}_1 \end{array} \right\}$$

та множину IMPL простих імплікант, які входять до складу СДНФ функції:

$$IMPL = \left\{ \begin{array}{cccccc} A & B & C & D & E & F \\ x_2 \bar{x}_1, & x_4 \bar{x}_3 \bar{x}_2, & x_4 \bar{x}_3 \bar{x}_1, & \bar{x}_3 \bar{x}_2 \bar{x}_1, & \bar{x}_4 \bar{x}_3 \bar{x}_1, & \bar{x}_4 \bar{x}_3 \bar{x}_2 \end{array} \right\}$$

Кожній простій імпліканті з множини IMPL ставимо у відповідність символи латинської абетки:

$$\begin{array}{llll} x_2 \bar{x}_1 = A, & x_4 \bar{x}_3 \bar{x}_2 = B, & x_4 \bar{x}_3 \bar{x}_1 = C, & \bar{x}_3 \bar{x}_2 \bar{x}_1 = D, \\ \bar{x}_4 \bar{x}_3 \bar{x}_1 = E, & \bar{x}_4 \bar{x}_3 \bar{x}_2 = F. & & \end{array}$$

4. Сформуємо логічні умови покриття конституент одиниці з множини KONST. Власною частиною, наприклад, першої конституенти одиниці (KO1) є імпліканта $\bar{x}_4 \bar{x}_3 \bar{x}_1 = E$ та імпліканта $\bar{x}_4 \bar{x}_3 \bar{x}_2 = F$.

Тому логічною умовою покриття KO1 є вираз $E \vee F$. Отже,

$$\begin{aligned} \text{KO1} : \quad & \bar{x}_4 \bar{x}_3 \bar{x}_2 \bar{x}_1 \rightarrow E \vee F; & \text{KO2} : \quad & \bar{x}_4 \bar{x}_3 \bar{x}_2 x_1 \rightarrow D \vee F; \\ \text{KO3} : \quad & \bar{x}_4 \bar{x}_3 x_2 \bar{x}_1 \rightarrow A \vee E; & \text{KO4} : \quad & \bar{x}_4 x_3 x_2 \bar{x}_1 \rightarrow A \\ \text{KO5} : \quad & x_4 \bar{x}_3 \bar{x}_2 x_1 \rightarrow C \vee D; & \text{KO6} : \quad & x_4 \bar{x}_3 x_2 \bar{x}_1 \rightarrow A \vee B; \\ \text{KO7} : \quad & x_4 \bar{x}_3 x_2 x_1 \rightarrow B \vee C; & \text{KO8} : \quad & x_4 x_3 x_2 \bar{x}_1 \rightarrow A. \end{aligned}$$

5. Запишемо логічну умову покриття перемикальної функції у як кон'юнкцію логічних умов покриття всіх конституент одиниці функції:

$$(E \vee F) \cdot (D \vee F) \cdot (A \vee E) \cdot A \cdot (C \vee D) \cdot (A \vee B) \cdot (B \vee C) \cdot A.$$

Виконаємо операції поглинання, розкриємо дужки та застосуємо аксіому повторення, внаслідок цього отримаємо вираз

$$ABDE \vee ACDE \vee ABDF \vee ACF, \quad (3.5)$$

який є диз'юнкцією кон'юнкцій.

6. Визначимо тупикові ДНФ та їх ціну. В отриманому логічному виразі кожному кон'юнктивному терму відповідає тупикова ДНФ:

$$\begin{aligned} u_{\text{ТДНФ1}} &= A \vee B \vee D \vee E = x_2 \bar{x}_1 \vee x_4 \bar{x}_3 x_2 \vee \bar{x}_3 \bar{x}_2 x_1 \vee \bar{x}_4 \bar{x}_3 \bar{x}_1 & \text{Ц} &= 11; \\ u_{\text{ТДНФ2}} &= A \vee C \vee D \vee E = x_2 \bar{x}_1 \vee x_4 \bar{x}_3 x_1 \vee \bar{x}_3 \bar{x}_2 x_1 \vee \bar{x}_4 \bar{x}_3 \bar{x}_1 & \text{Ц} &= 11; \\ u_{\text{ТДНФ3}} &= A \vee B \vee D \vee F = x_2 \bar{x}_1 \vee x_4 \bar{x}_3 x_2 \vee \bar{x}_3 \bar{x}_2 x_1 \vee \bar{x}_4 \bar{x}_3 \bar{x}_2 & \text{Ц} &= 11; \\ u_{\text{ТДНФ4}} &= A \vee C \vee F = x_2 \bar{x}_1 \vee x_4 \bar{x}_3 x_1 \vee \bar{x}_4 \bar{x}_3 \bar{x}_2 & \text{Ц} &= 8. \end{aligned}$$

Принадно зазначимо наступне. До кожного кон'юнктивного терма з (3.5) входить літера А, якій відповідає проста імпліканта $x_2 \bar{x}_1$.

Якщо яка-небудь літера входить до складу всіх кон'юнктивних термів, то проста імпліканта, що відповідає даній літері, входить до складу ядра перемикальної функції.

Ядром перемикальної функції у є проста імпліканта $x_2 \bar{x}_1$ (А).

Вона входить до складу всіх ТДНФ.

7. Серед отриманих тупикових ДНФ виберемо форму з мінімальною ціною – $u_{\text{ТДНФ4}}$ ($\text{Ц} = 8$), вона й буде мінімальною ДНФ.

$$\text{Отже, } u_{\text{МДНФ}} = u_{\text{ТДНФ4}} = x_2 \bar{x}_1 \vee x_4 \bar{x}_3 x_1 \vee \bar{x}_4 \bar{x}_3 \bar{x}_2.$$

Початкова форма функції – $u_{\text{ДНФ}}$, мала ціну – 32. \blacksquare

Розв'яжемо ще одну задачу.

Задача 3.5. Перемикальна функція $f(x_4, x_3, x_2, x_1)$ дорівнює одиниці на наборах 0, 1, 2, 6, 7, 8, 9, 10, 15, а на решті наборів – нулю. Мінімізувати

функцію методом Квайна – Мак-Класкі. Для визначення тупикових ДНФ застосувати спосіб Петріка.

Розв'язування.

1. Запишемо функцію f у цифровій формі

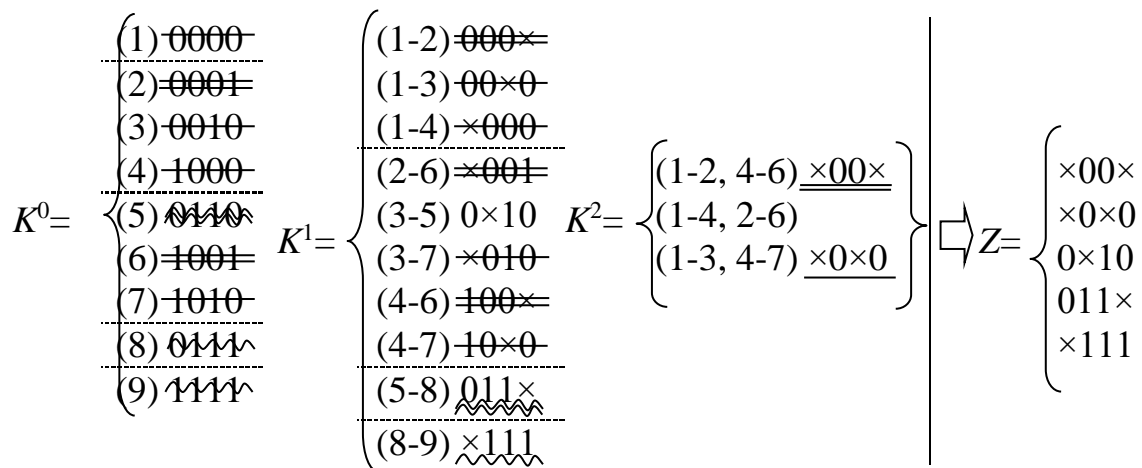
$$f = 0000 \vee 0001 \vee 0010 \vee 0110 \vee 0111 \vee 1000 \vee 1001 \vee 1010 \vee 1111$$

та сформуємо комплекс K^0 0-кубів (конституент одиниці функції) (рис. 3.14). Для цього конституенти одиниці розіб'ємо на групи за кількістю одиниць – набір 0000, що не містить одиниць, набори, що містять відповідно по одній, дві, три та чотири одиниці.

Сформуємо комплекс K^1 1-кубів, виконавши можливі склеювання 0-кубів, що належать сусіднім групам, а також комплекс K^2 2-кубів, виконавши можливі склеювання 1-кубів, що належать сусіднім групам. Результатом є 2-куби $\{ \times 00 \times \}$ та $\{ \times 0 \times 0 \}$.

Подальші склеювання неможливі.

$$f = 0000 \vee 0001 \vee 0010 \vee 0110 \vee 0111 \vee 1000 \vee 1001 \vee 1010 \vee 1111$$



$$f_{\text{сднф}} = \times 00 \times \vee \times 0 \times 0 \vee 0 \times 10 \vee 011 \times \vee \times 111$$

Рис. 3.14. Визначення скороченої ДНФ перемикальної функції f

2. В отриманих комплексах K^0 , K^1 , K^2 виконаємо всі можливі поглинання (рис. 3.14). Наслідком цього є утворення Z -покриття функції f . Куби в Z -покритті відповідають простим імплікантам функції f . Тому скорочена ДНФ функції має вигляд:

$$u_{\text{сднф}} = \times 00 \times \vee \times 0 \times 0 \vee 0 \times 10 \vee 011 \times \vee \times 111.$$

3. Сформуємо множину $KONST$ конституент одиниці функції f

$$KONST = \{ \textcircled{1}, \textcircled{2}, \textcircled{3}, \textcircled{4}, \textcircled{5}, \textcircled{6}, \textcircled{7}, \textcircled{8}, \textcircled{9} \}$$

$$KONST = \{ 0000, 0001, 0010, 0110, 0111, 1000, 1001, 1010, 1111 \}$$

та множину IMPL простих імплікант, які входять до складу СДНФ функції

$$\text{IMPL} = \{ \times 00 \times, \times 0 \times 0, 0 \times 10, 011 \times, \times 111 \}$$

4. Сформуємо логічні умови покриття конститuent одиниці з множини KONST:

$$\begin{array}{ll} \text{KO1} : 0000 \rightarrow A \vee B; & \text{KO2} : 0001 \rightarrow A; \\ \text{KO3} : 0010 \rightarrow B \vee C; & \text{KO4} : 0110 \rightarrow C \vee D; \\ \text{KO5} : 0111 \rightarrow D \vee E; & \text{KO6} : 1000 \rightarrow A \vee B; \\ \text{KO7} : 1001 \rightarrow A; & \text{KO8} : 1010 \rightarrow B; \\ \text{KO9} : 1111 \rightarrow E. & \end{array}$$

5. Запишемо логічну умову покриття перемикальної функції f як кон'юнкцію логічних умов покриття всіх її конститuent одиниці:

$$(A \vee B) \cdot A \cdot (B \vee C) \cdot (C \vee D) \cdot (D \vee E) \cdot (A \vee B) \cdot A \cdot B \cdot E.$$

Виконавши операції поглинання, застосувавши аксіому повторення, розкривши дужки отримаємо вираз у вигляді диз'юнкції кон'юнкцій:

$$ABCF \vee ABDF.$$

6. Визначимо тупикові ДНФ та їх ціну:

$$\begin{aligned} f_{\text{ТДНФ1}} &= A \vee B \vee C \vee F = \times 00 \times \vee \times 0 \times 0 \vee 0 \times 10 \vee \times 111 = \\ &= \bar{x}_3 \bar{x}_2 \vee \bar{x}_3 \bar{x}_1 \vee \bar{x}_4 x_2 \bar{x}_1 \vee x_3 x_2 x_1 \end{aligned} \quad (Ц_1 = 10)$$

$$\begin{aligned} f_{\text{ТДНФ2}} &= A \vee B \vee D \vee F = \times 00 \times \vee \times 0 \times 0 \vee 011 \times \vee \times 111 = \\ &= \bar{x}_3 \bar{x}_2 \vee \bar{x}_3 \bar{x}_1 \vee \bar{x}_4 x_3 x_2 \vee x_3 x_2 x_1 \end{aligned} \quad (Ц_2 = 10)$$

Як бачимо, ядром функції є прості імпліканти, яким відповідають букви А, В, F (ці букви входять до обох кон'юнктивних термів).

7. Оскільки обидві тупикові форми мають однакову ціну, то будь-яку з них можна обрати як мінімальну.

Нехай $f_{\text{МДНФ}} = f_{\text{ТДНФ2}}$, тобто

$$f_{\text{МДНФ}} = \bar{x}_3 \bar{x}_2 \vee \bar{x}_3 \bar{x}_1 \vee \bar{x}_4 x_3 x_2 \vee x_3 x_2 x_1 \quad (Ц = 10)$$

Початкова форма функції – $f_{\text{ДДНФ}}$, мала ціну – 36. ■

Запитання та завдання

1. Якими двома способами можна визначити тупикові ДНФ перемикальної функції?
2. Охарактеризуйте візуальний спосіб знаходження тупикових ДНФ функції.
3. Поясніть терміни: а) логічна умова покриття конститuentи одиниці; б) логічна умова покриття перемикальної функції.

4. Охарактеризуйте спосіб Петріка знаходження тупикових ДНФ функції.
5. Порівняйте візуальний спосіб та спосіб Петріка знаходження ТДНФ перемикальної функції.
6. На якому етапі мінімізації перемикальної функції може застосовуватись спосіб Петріка?

Задачі для самостійного розв'язування

1. Перемикальна функція від чотирьох змінних дорівнює нулю на наборах 0, 2, 4, 6, 11, 13, 14, 15 і одиниці – на решті наборів. Відома скорочена ДНФ функції в цифровій формі:

$$u_{\text{сднф}} = 0 \times \times 1 \vee 1 \times 00 \vee 100 \times \vee \times 100 \vee 010 \times \vee 1010.$$

Знайти МДНФ функції, застосувавши спосіб Петріка для пошуку тупикових ДНФ.

2. Перемикальна функція від трьох змінних дорівнює одиниці на наборах 0, 2, 4, 6, 7 та нулю – на решті наборів. Відома також скорочена ДНФ функції в аналітичній формі:

$$f_{\text{сднф}} = \bar{x}_3 \bar{x}_1 \vee \bar{x}_2 \bar{x}_1 \vee x_2 \bar{x}_1 \vee x_3 \bar{x}_1 \vee x_3 x_2.$$

Знайти МДНФ функції, застосувавши спосіб Петріка для пошуку тупикових ДНФ.

3. Мінімізувати перемикальну функцію від 4-х змінних методом Квайна – Мак-Класкі, якщо відомо, що функція дорівнює нулю на наборах 0, 1, 2, 6, 10, 13, 14 і одиниці – на решті наборів. Для знаходження тупикових ДНФ застосувати спосіб Петріка.
4. Мінімізувати перемикальну функцію від 4-х змінних методом Квайна, якщо відомо, що функція дорівнює одиниці на наборах 0, 8, 9, 11, 13, 14, 15 і нулю – на решті наборів. Тупикові ДНФ визначити способом Петріка.

3.5. Графічні методи мінімізації перемикальних функцій

Особливістю аналітичних методів мінімізації перемикальних функцій – Квайна, Квайна – Мак-Класкі, невизначених коефіцієнтів, Блейка-Порецького (див. рис. 3.11), є їх двоетапність – на першому етапі визначають скорочену ДНФ, а на другому – знаходять тупикові ДНФ, з яких вибирають мінімальну ДНФ. Ці методи є універсальними – їх можна

застосувати для мінімізації перемикальних функцій від довільної кількості змінних, поданих в диз'юнктивній формі (найчастіше – в ДДНФ).

Якщо перемикальна функція залежить від відносно невеликої кількості змінних (до 7-ми), то замість аналітичних застосовують спрощені методи, що дістали назву графічних методів мінімізації перемикальних функцій.

Розрізняють два графічні методи мінімізації перемикальних функцій:

- метод мінімізації на основі діаграм Вейча;
- метод мінімізації на основі карт Карно.

Перевагами обох методів графічної мінімізації є:

- 1) *наочність* – для отримання мінімальної форми функції використовується лише одна таблиця;
- 2) *простота та швидкість* отримання результату – мінімальну форму функції отримують з таблиці одразу, минаючи етап знаходження скороченої ДНФ та етап визначення тупикових ДНФ функції;
- 3) з таблиці можна отримати як мінімальну ДНФ, так і мінімальну КНФ функції.

Розглянемо метод мінімізації перемикальних функцій на основі діаграм Вейча.

Діаграма Вейча перемикальної функції від n змінних є прямокутною (якщо n – непарне) або квадратною (якщо n – парне) таблицею, що складається з 2^n клітинок (рис. 3.15 – 3.18). Така таблиця є аналогом таблиці істинності перемикальної функції. Кожній клітинці на діаграмі Вейча відповідає двійковий набір, для якого може бути записана конституента одиниці або конституента нуля. Якщо на деякому наборі аргументів функція дорівнює 1, то у клітинку, що відповідає даному набору, записують 1. Клітинки, що відповідають наборам, на яких функція дорівнює 0, або заповнюють нулями, або залишають незаповненими.

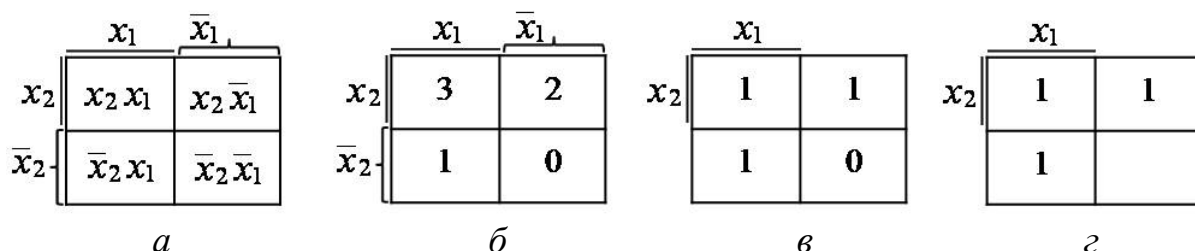


Рис. 3.15. Діаграма Вейча перемикальної функції від двох змінних:

a – конституенти одиниці; b – номери двійкових наборів;

v – приклад повного заповнення діаграми;

z – приклад скороченого заповнення діаграми

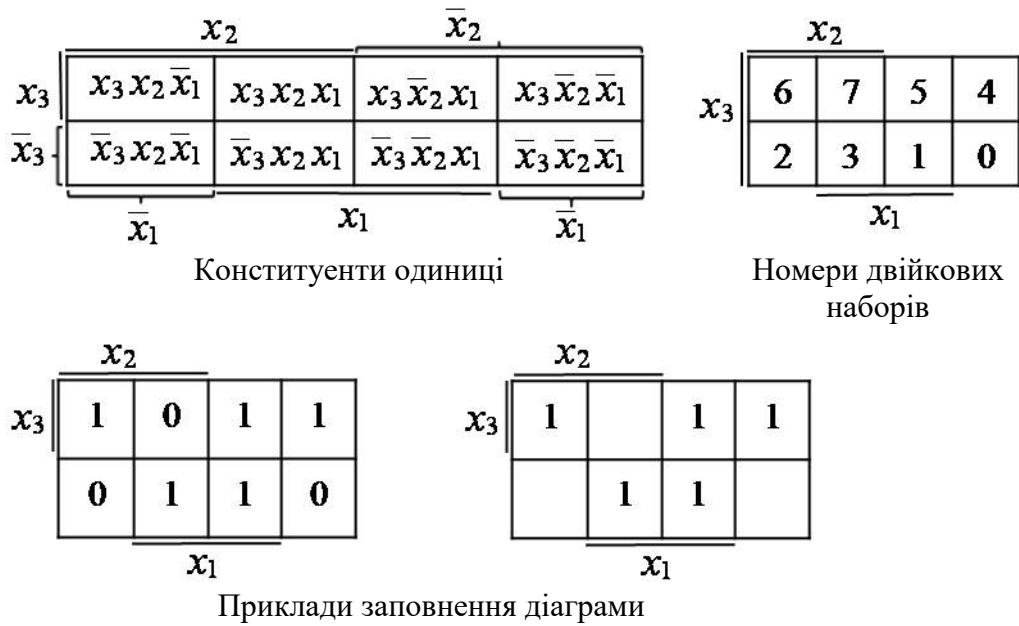


Рис. 3.16. Діаграма Вейча перемикальної функції від трьох змінних

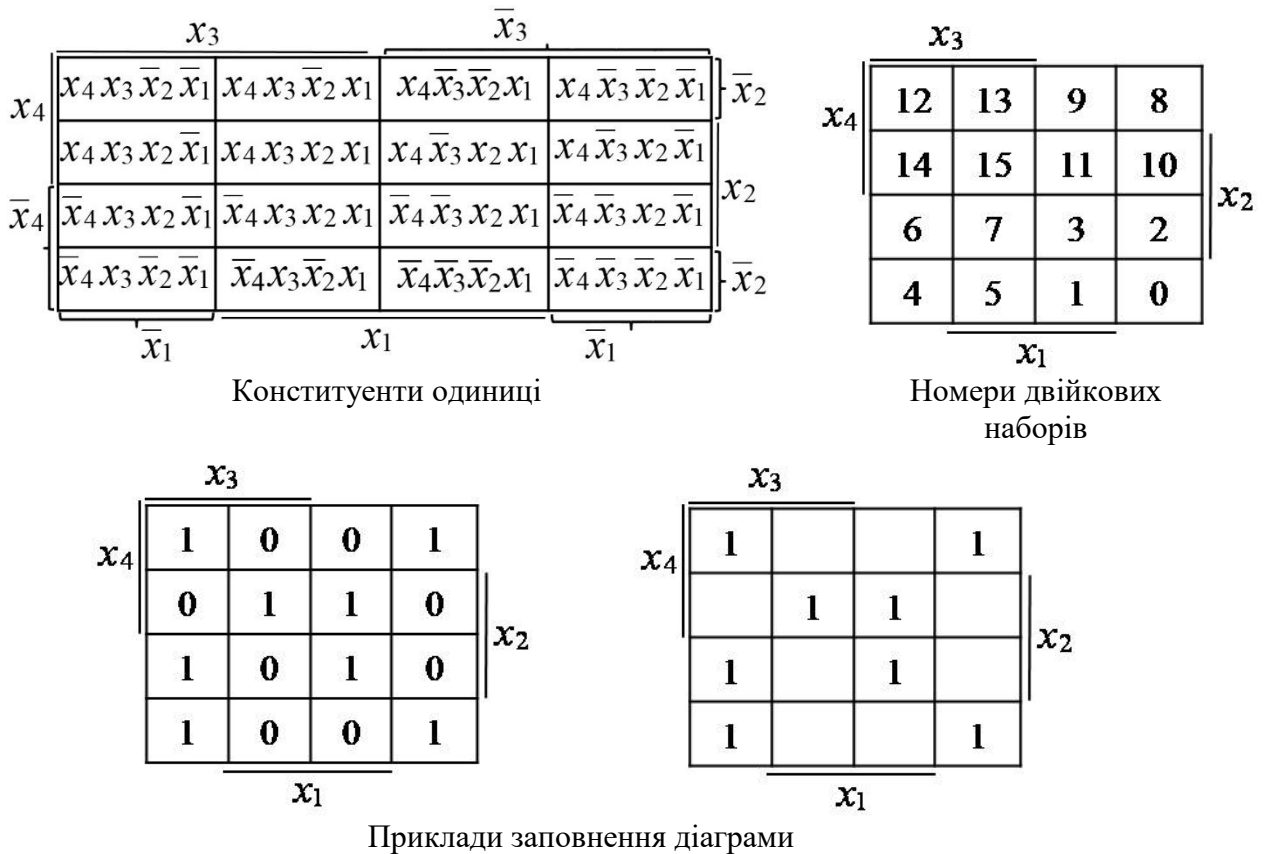


Рис. 3.17. Діаграма Вейча перемикальної функції від чотирьох змінних

Аналізуючи наведені на рис. 3.15 – 3.17 діаграми Вейча бачимо, що будь-яким двом сусіднім клітинкам діаграми, розташованим одна відносно одної горизонтально або вертикально (але не по діагоналі), відповідають сусідні конституенти одиниці, тобто такі, що відрізняються одна від одної лише запереченням в однієї зі змінних (інші змінні співпадають).

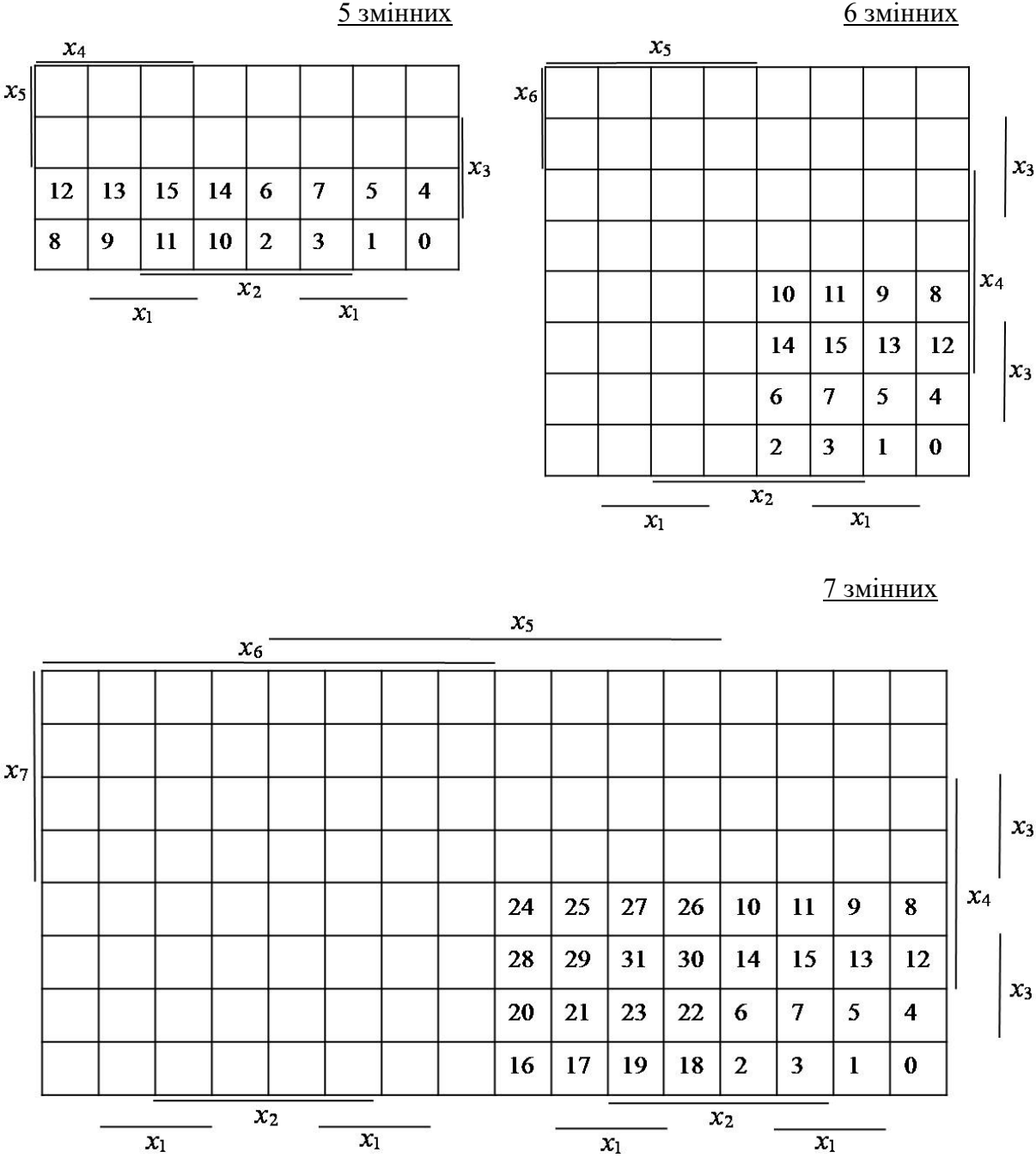


Рис. 3.18. Діаграма Вейча перемикальної функції від 5-ти, 6-ти та 7-ми змінних

Наприклад, на діаграмі Вейча від трьох змінних (рис. 3.16) сусіднім клітинкам з номерами 5 та 1 відповідають сусідні конституенти одиниці $x_3\bar{x}_2x_1$ та $\bar{x}_3\bar{x}_2x_1$ (ці конституенти відрізняються лише зміною x_3); на діаграмі Вейча від чотирьох змінних (рис. 3.17) сусіднім клітинкам 11 та 10 відповідають конституенти одиниці $x_4\bar{x}_3x_2x_1$ та $x_4\bar{x}_3x_2\bar{x}_1$ (вони відрізняються лише змінною x_1).

Діаграма Вейча перемикальної функції від n змінних є розгорткою n -вимірного куба на площину. Тому клітинки, розташовані на протилежних краях діаграми, є сусідніми (рис. 3.19).

Але до сусідніх конституент Ax та $A\bar{x}$ можна застосувати операцію повного склеювання $A\bar{x} \vee Ax = A$, де A – спільна частина обох конституент одиниці. Тому будь-яким двом сусіднім клітинкам на діаграмі Вейча відповідає кон'юнктивний терм (імпліканта), який є спільною частиною відповідних двох сусідніх конституент одиниці й має на одну букву менше за конституенти одиниці.

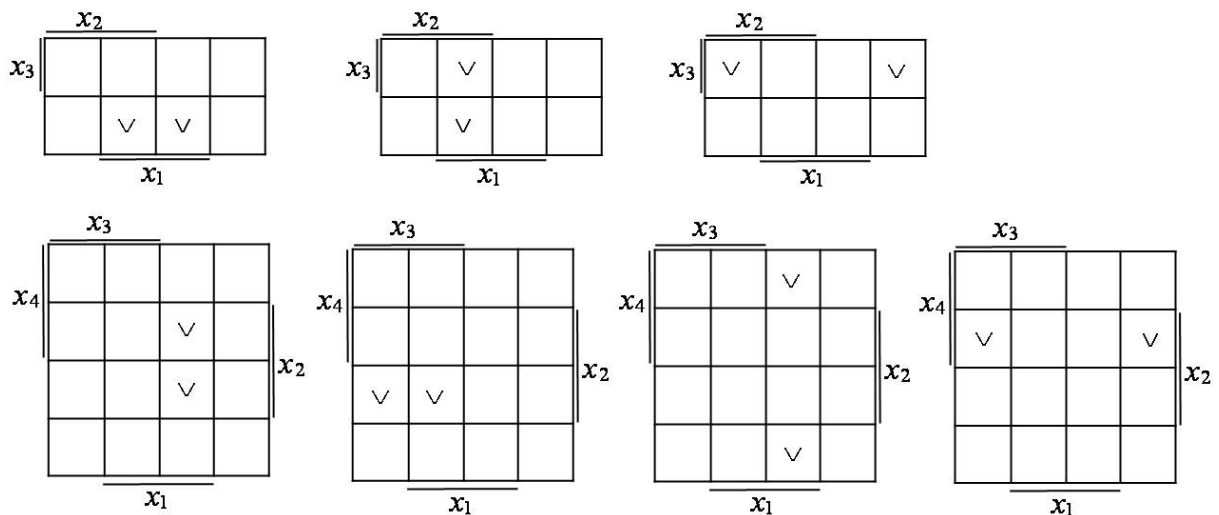
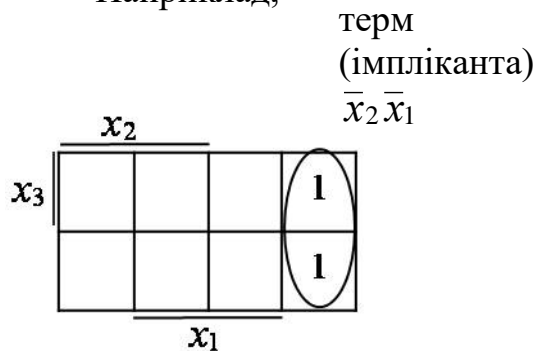
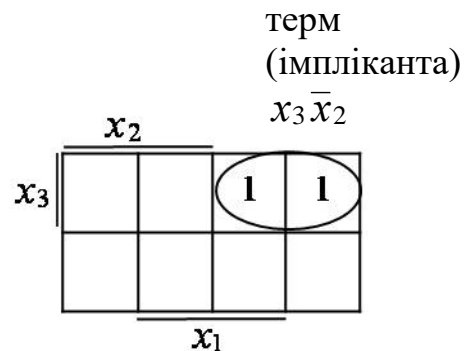


Рис. 3.19. Приклади сусідства двох клітинок на діаграмах Вейча

Наприклад,



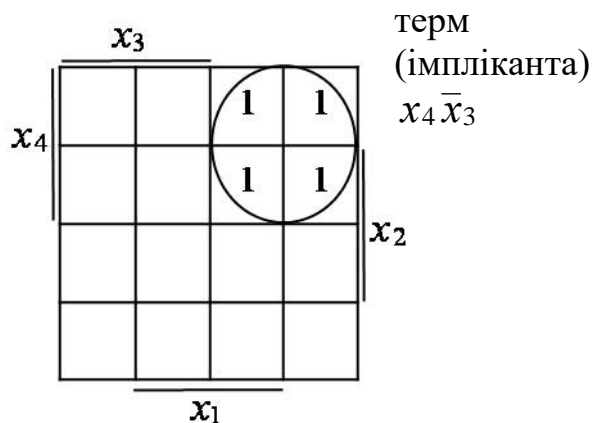
$$x_3 \bar{x}_2 \bar{x}_1 \vee \bar{x}_3 \bar{x}_2 \bar{x}_1 = \bar{x}_2 \bar{x}_1$$



$$x_3 \bar{x}_2 x_1 \vee x_3 \bar{x}_2 \bar{x}_1 = x_3 \bar{x}_2$$

Сусідніми можуть бути й 4 клітинки на діаграмі Вейча (рис. 3.20) (у загальному випадку – 2, 4, 8, 16 і т.д. клітинок). Тому будь-яким чотирьом сусіднім клітинкам на діаграмі Вейча відповідає терм (імпліканта), що має на дві букви менше порівняно з конститuentами одиниці.

Наприклад,



$$\begin{aligned} & x_4 \bar{x}_3 \bar{x}_2 x_1 \vee x_4 \bar{x}_3 \bar{x}_2 \bar{x}_1 \vee x_4 \bar{x}_3 x_2 x_1 \vee \\ & \vee x_4 \bar{x}_3 x_2 \bar{x}_1 = x_4 \bar{x}_3 \bar{x}_2 (x_1 \vee \bar{x}_1) \vee \\ & \vee x_4 \bar{x}_3 x_2 (x_1 \vee \bar{x}_1) = x_4 \bar{x}_3 \bar{x}_2 \vee \\ & \vee x_4 \bar{x}_3 x_2 = x_4 \bar{x}_3. \end{aligned}$$

Склеювати на діаграмі Вейча можна будь-які сусідні 2, 4, 8, 16, ... клітинок (конститuent одиниці). Прямокутнику (групі клітинок), що містить 2^k клітинок ($k = 1, 2, 3, \dots, n - 1$), відповідає імпліканта, що складається з $n - k$ букв.

Так, будь-яким 8-ми сусіднім клітинкам на діаграмі Вейча відповідає кон'юнктивний терм (імпліканта), що має на 3 букви менше порівняно з конститuentами одиниці (рис. 3.21).

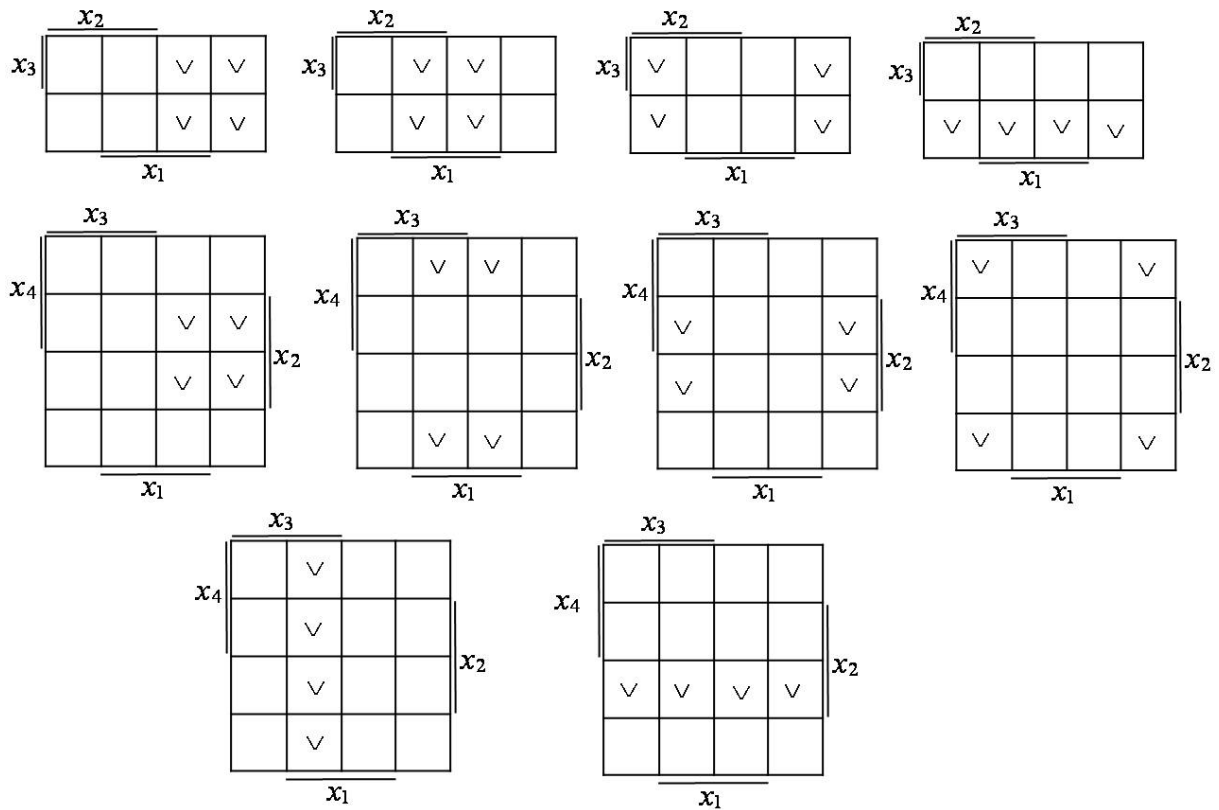


Рис. 3.20. Приклади сусідства чотирьох клітинок на діаграмах Вейча

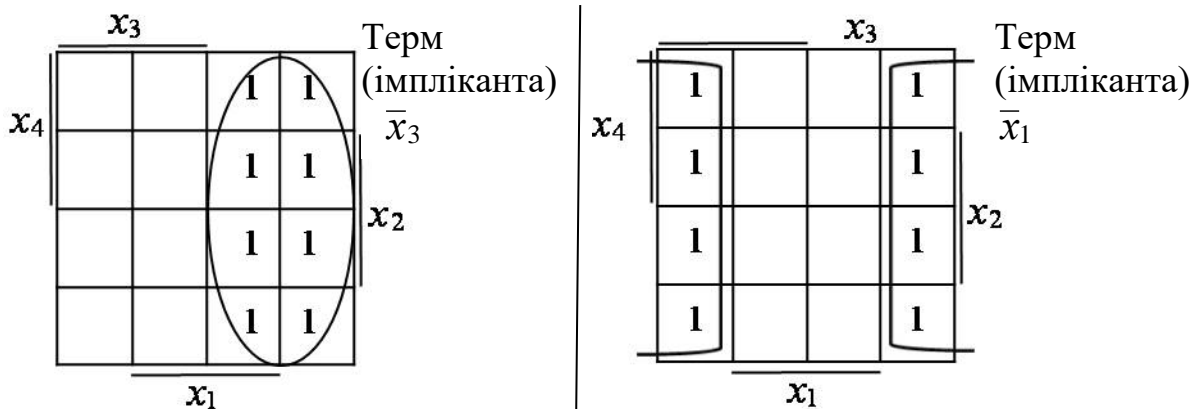


Рис. 3.21. Приклади склеювання восьми сусідніх конститuent (сусідніх клітинок) на діаграмах Вейча

Діаграми Вейча застосовують для мінімізації перемикальних функцій, поданих в ДДНФ, ДКНФ або у вигляді таблиці істинності. Діаграми Вейча забезпечують отримання МДНФ та МКНФ перемикальних функцій.

Розглянемо знаходження МДНФ перемикальних функцій за допомогою діаграм Вейча.

Знаходження МДНФ перемикальної функції зводиться до визначення мінімальної кількості найбільш коротких кон'юнктивних термів (простих імплікант), які покривають всі одиниці на діаграмі Вейча.

Процес мінімізації перемикальної функції полягає в наступному.

1. Заповнити діаграму Вейча (на підставі ДДНФ, ДКНФ або таблиці істинності функції).
2. Об'єднати одиниці в прямокутники (групи) максимально можливого розміру (... , 16, 8, 4, 2 одиниці) так, щоб покрити всі одиниці на діаграмі, прагнучи при цьому, щоб кількість утворених прямокутників (груп) була якнайменшою:
 - кожна одиниця має входити до складу принаймні одного прямокутника (групи),
 - одна й та сама одиниця може входити до складу кількох різних прямокутників (груп),
 - мінімальний за розміром прямокутник може включати лише одну одиницю.
3. Записати МДНФ – для кожного визначення в п.2 прямокутника сформулювати відповідний кон'юнктивний терм (просту імпліканту) та об'єднати їх знаком диз'юнкції.

Задача 3.6. Мінімізувати перемикальну функцію y від трьох змінних, задану таблицею істинності (табл. 3.5).

Таблиця 3.5

Таблиця істинності перемикальної функції y

Номер набору	Значення аргументів			Значення функції y
	x_3	x_2	x_1	
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Розв'язування.

Заповнимо діаграму Вейча функції – одиниці слід розташувати в клітинках з номерами 2, 3, 4, 6 та 7, нулі – в решті клітинках. Об'єднаємо одиниці в прямокутники (групи) максимально можливого розміру так, щоб покрити всі одиниці на діаграмі.

	x_2			
x_3	6	7	5	4
	2	3	1	0
	x_1			

	x_2			
x_3	1	1	0	1
	1	1	0	0
	x_1			

Таких груп – дві: першу утворюють чотири клітинки з номерами 6, 7, 2, 3; другу – дві клітинки з номерами 6, 4. Першій групі клітинок відповідає терм (імпліканта) x_2 (вона є спільною для всіх 4-х клітинок), другій – терм (імпліканта) $x_3\bar{x}_1$.

Тому мінімальною ДНФ функції є

$$y_{\text{МДНФ}} = x_2 \vee x_3\bar{x}_1. \blacksquare$$

Задача 3.7. Мінімізувати повністю визначену функцію z від трьох змінних, задану в ДКНФ:

$$z_{\text{ДКНФ}} = (x_3 \vee \bar{x}_2 \vee x_1)(x_3 \vee \bar{x}_2 \vee \bar{x}_1)(\bar{x}_3 \vee x_2 \vee x_1)(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_1).$$

Розв’язування. Задану функцію запишемо в числовому вигляді:

$$z_{\text{ДКНФ}} = \bar{2} \cdot \bar{3} \cdot \bar{4} \cdot \bar{7}.$$

Оскільки функція z – повністю визначена, то її ДДНФ має вигляд:

$$z_{\text{ДДНФ}} = 0 \vee 1 \vee 5 \vee 6.$$

Тепер заповнимо діаграму Вейча та мінімізуємо функцію:

	x_2			
x_3	1		1	
			1	1
	x_1			

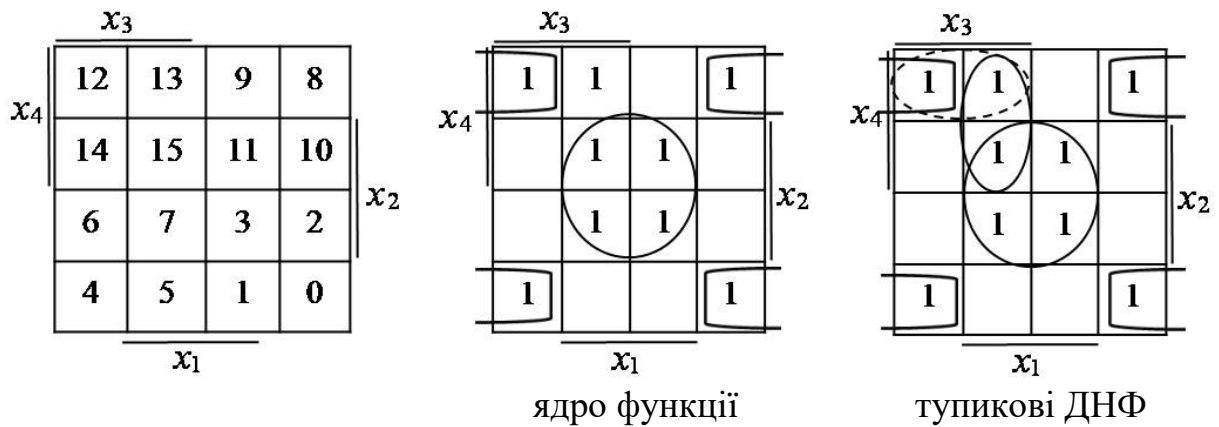
$$z_{\text{МДНФ}} = \bar{x}_3\bar{x}_2 \vee \bar{x}_2x_1 \vee x_3x_2\bar{x}_1. \blacksquare$$

Задача 3.8. Мінімізувати перемикальну функцію f від 4-х змінних, подану в ДДНФ:

$$z_{\text{ДДНФ}} = 0 \vee 3 \vee 4 \vee 7 \vee 8 \vee 11 \vee 12 \vee 13 \vee 15.$$

Розв'язування.

ДДНФ у числовому вигляді дозволяє безпосередньо заповнити діаграму Вейча:



Розташування одиниць на діаграмі дозволяє сформувати дві групи одиниць, які утворюють ядро функції: $x_2 x_1$, $\bar{x}_2 \bar{x}_1$. Одиницю в клітинці 13, що залишилась несклеєною, можна склеїти з одиницею в клітинці 12 або з одиницею в клітинці 15. Відповідно, отримуємо дві тупикові форми:

$$f_{\text{ТДНФ1}} = x_2 x_1 \vee \bar{x}_2 \bar{x}_1 \vee x_4 x_3 \bar{x}_2,$$

$$f_{\text{ТДНФ2}} = x_2 x_1 \vee \bar{x}_2 \bar{x}_1 \vee x_4 x_3 x_1.$$

Оскільки обидві ТДНФ мають однакову ціну ($C = 7$), то будь-яку з них можна обрати як мінімальну ДНФ. Як мінімальну оберемо другу тупикову форму:

$$f_{\text{МДНФ}} = f_{\text{ТДНФ2}} = x_2 x_1 \vee \bar{x}_2 \bar{x}_1 \vee x_4 x_3 x_1. \blacksquare$$

За допомогою діаграм Вейча знаходять не лише мінімальні диз'юнктивні (рис. 3.22), але й мінімальні кон'юнктивні форми перемикальних функцій.

Оскільки кожному набору змінних перемикальної функції можна поставити у відповідність як конститuentу одиниці, так і конститuentу нуля, то відповідно й кожен клітинку діаграми Вейча можна розглядати не лише як конститuentу одиниці, але й як конститuentу нуля (рис. 3.23).

Наприклад, клітинці з номером 3 діаграми Вейча перемикальної функції від трьох змінних відповідає як конститuentу одиниці $\bar{x}_3 x_2 x_1$, так і конститuentу нуля $x_3 \vee \bar{x}_2 \vee \bar{x}_1$.

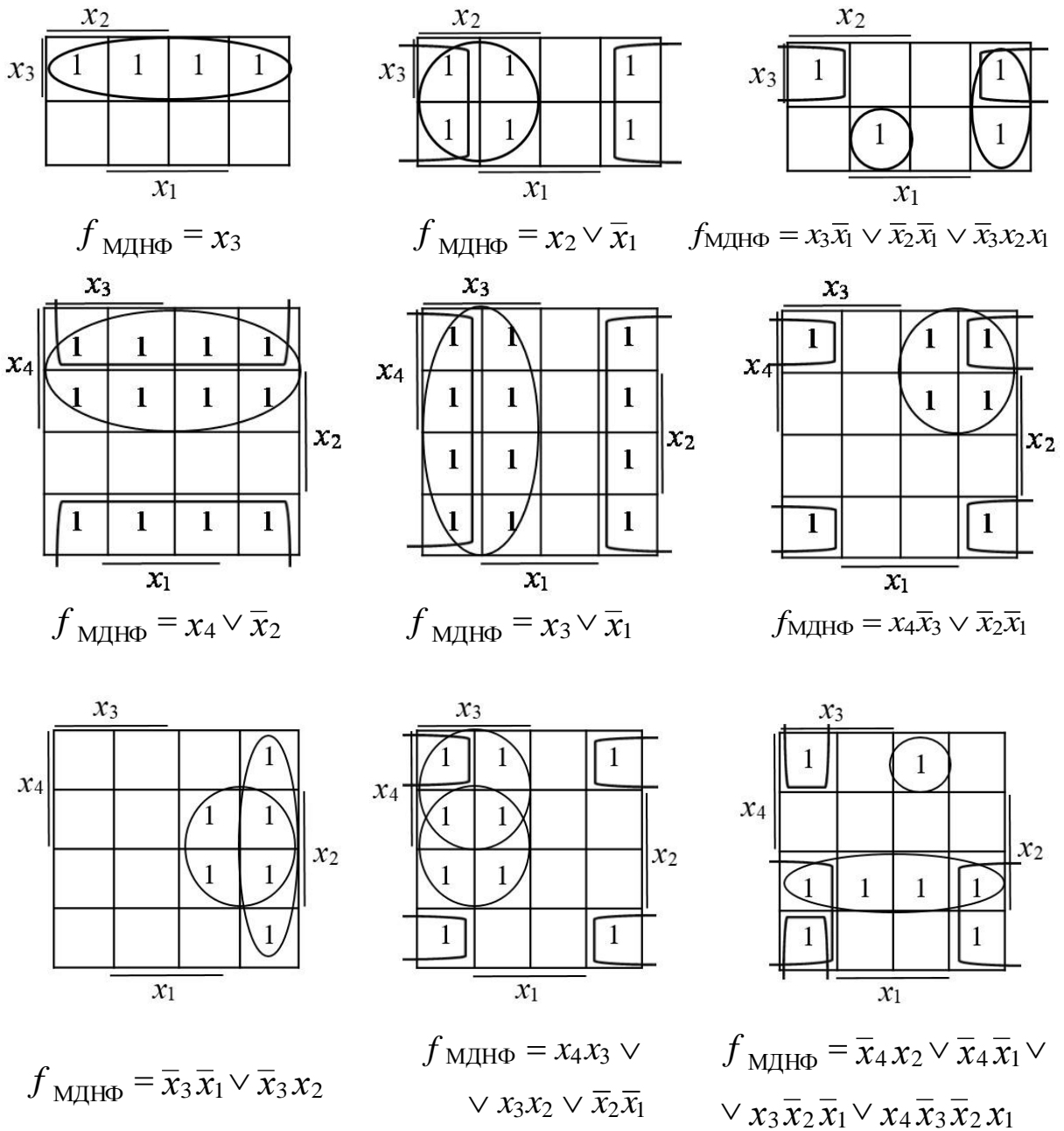


Рис. 3.22. Деякі приклади знаходження МДНФ перемикальних функцій від 3-х змінних

Для конституент нуля також справедливі правила сусідства.

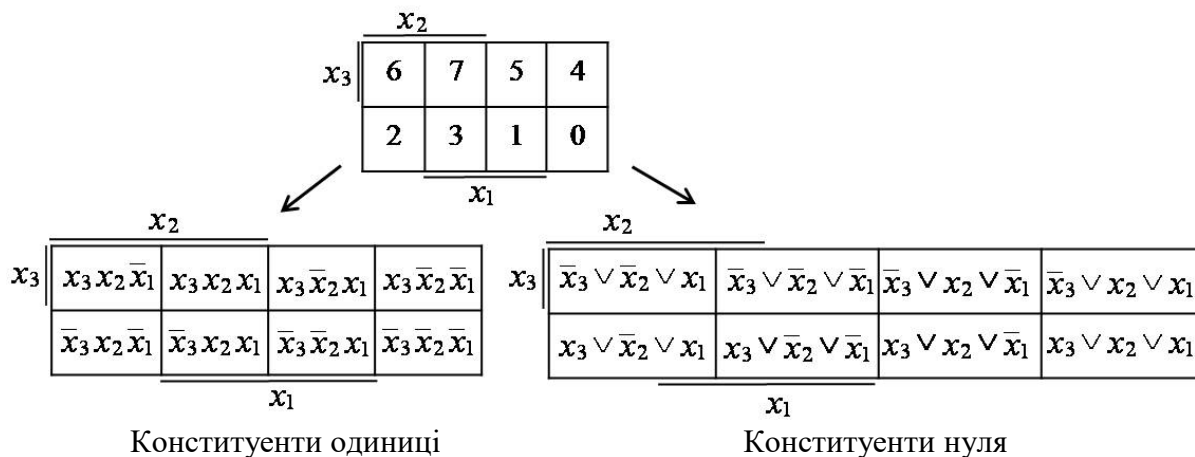
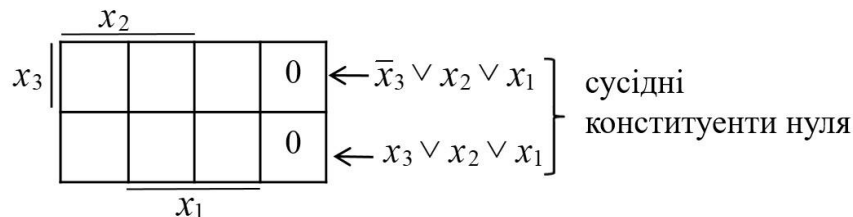


Рис. 3.23. Відповідність конститuent одиниці та конститuent нуля клітинкам діаграми Вейча перемикальної функції від 3-х змінних

Дві конститuentи нуля називають сусідніми, якщо вони відрізняються одна від одної лише запереченням в однієї зі змінних (решта змінних співпадають). Це означає, що будь-яким двом сусіднім клітинкам діаграми Вейча, розташованим горизонтально або вертикально (але не по діагоналі), відповідають сусідні конститuentи нуля.

Наприклад,



Якщо до деяких двох сусідніх конститuent одиниці Ax та $A\bar{x}$ можна застосувати правило повного склеювання

$$Ax \vee A\bar{x} = A, \tag{3.6}$$

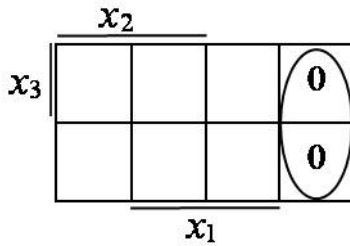
то до двох сусідніх конститuent нуля $A \vee x$ та $A \vee \bar{x}$ також можна застосувати правило повного склеювання виду

$$(A \vee x)(A \vee \bar{x}) = A. \tag{3.7}$$

Вираз (3.7) є двоїстим до виразу (3.6).

Отже, будь-яким двом сусіднім клітинкам на діаграмі Вейча відповідає диз'юнктивний терм (імпліканта), який є спільною частиною двох сусідніх конститuent нуля й має на одну букву менше за конститuentи нуля.

Наприклад,



терм (імпліканта) $x_2 \vee x_1$

$$(\bar{x}_3 \vee x_2 \vee x_1)(x_3 \vee x_2 \vee x_1) = (\bar{x}_3 \vee (x_2 \vee x_1))(x_3 \vee (x_2 \vee x_1)) = x_2 \vee x_1$$

Склеювання на діаграмі Вейча 2-х, 4-х, 8-ми, 16-ти, ... клітинок, у яких розміщуються нулі, забезпечує отримання диз'юнктивних термів-імплікант, добуток яких є кон'юнктивною нормальною формою перемикальної функції.

Знаходження МКНФ перемикальної функції зводиться до визначення мінімальної кількості найбільш коротких диз'юнктивних термів – простих імплікант, які покривають всі нулі на діаграмі Вейча.

Процес мінімізації перемикальної функції з метою отримання її МКНФ є таким самим, як і у випадку отримання МДНФ функції, але відрізняється тим, що виконувані процедури застосовують не до одиниць (конституент одиниці), а до нулів (конституент нуля), і отримані диз'юнктивні терми (прості імпліканти) об'єднують знаком кон'юнкції.

Задача 3.9. Знайти МДНФ та МКНФ перемикальної функції f від 3-х змінних, яка дорівнює одиниці на наборах 1, 4, 5, 6 та нулю – на решті наборів.

Розв'язування. Запишемо функцію в диз'юнктивній формі

$$f = 1 \vee 4 \vee 5 \vee 6 = 001 \vee 100 \vee 101 \vee 110$$

та розташуємо одиниці у клітинках з координатами 001, 100, 101, 110, а нулі – в решті клітинок (рис. 3.24).

Для отримання МДНФ функції об'єднаємо одиниці в групи як показано на рис. 3.24. Для кожної групи одиниць запишемо відповідні їм кон'юнктивні терми ($x_3 \bar{x}_1$, $\bar{x}_2 x_1$) та об'єднаємо їх знаком диз'юнкції:

$$f_{\text{МДНФ}} = x_3 \bar{x}_1 \vee \bar{x}_2 x_1.$$

Для отримання МКНФ функції об'єднаємо нулі на діаграмі Вейча в дві групи (рис. 3.24), для кожної з груп нулів запишемо відповідні їм диз'юнктивні терми – прості імпліканти ($x_3 \vee x_1$, $\bar{x}_2 \vee \bar{x}_1$) та об'єднаємо терми знаком кон'юнкції:

$$f_{\text{МКНФ}} = (x_3 \vee x_1)(\bar{x}_2 \vee \bar{x}_1). \blacksquare$$

Деякі приклади знаходження мінімальних кон'юнктивних форм перемикальних функцій за допомогою діаграм Вейча подано на рис. 3.25.

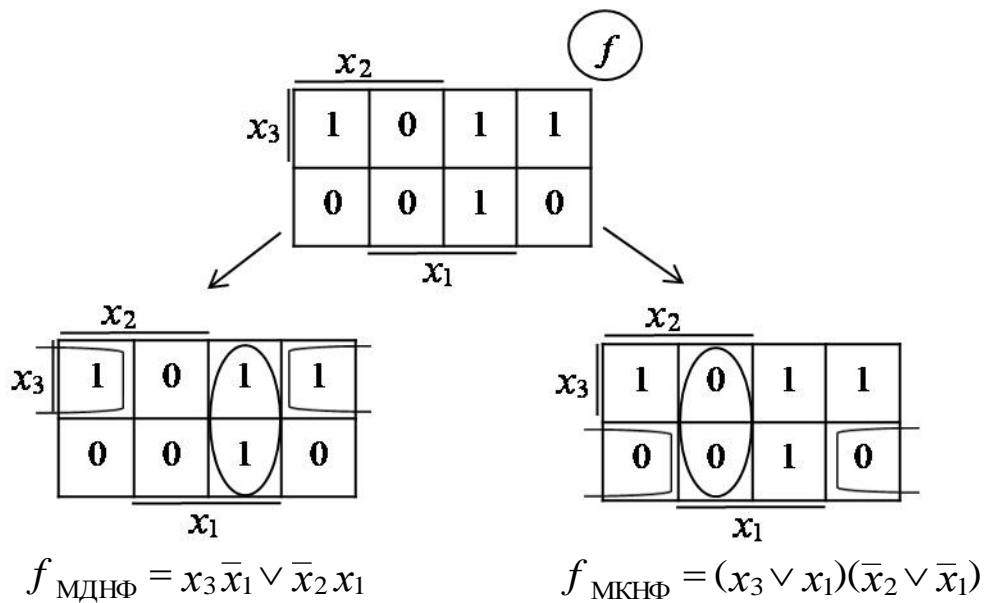


Рис. 3.24. Знаходження МДНФ та МКНФ перемикальної функції f

Іншим графічним методом мінімізації перемикальних функцій є карти Карно (Karnaugh maps). Як і діаграма Вейча, карта Карно є розгорткою n -вимірного куба на площину. Кожній клітинці діаграми (карти) відповідає вершина куба.

Карти Карно, як і діаграми Вейча, дозволяють отримувати як МДНФ, так і МКНФ перемикальних функцій. Рядки та стовпці карт Карно нумерують кодом Грея (рис. 3.26). Код Грея – це такий спосіб двійкової нумерації, коли наступне двійкове число відрізняється від попереднього лише в одному розряді. За такого способу нумерації сусіднім клітинкам відповідають сусідні конституенти одиниці (нуля).

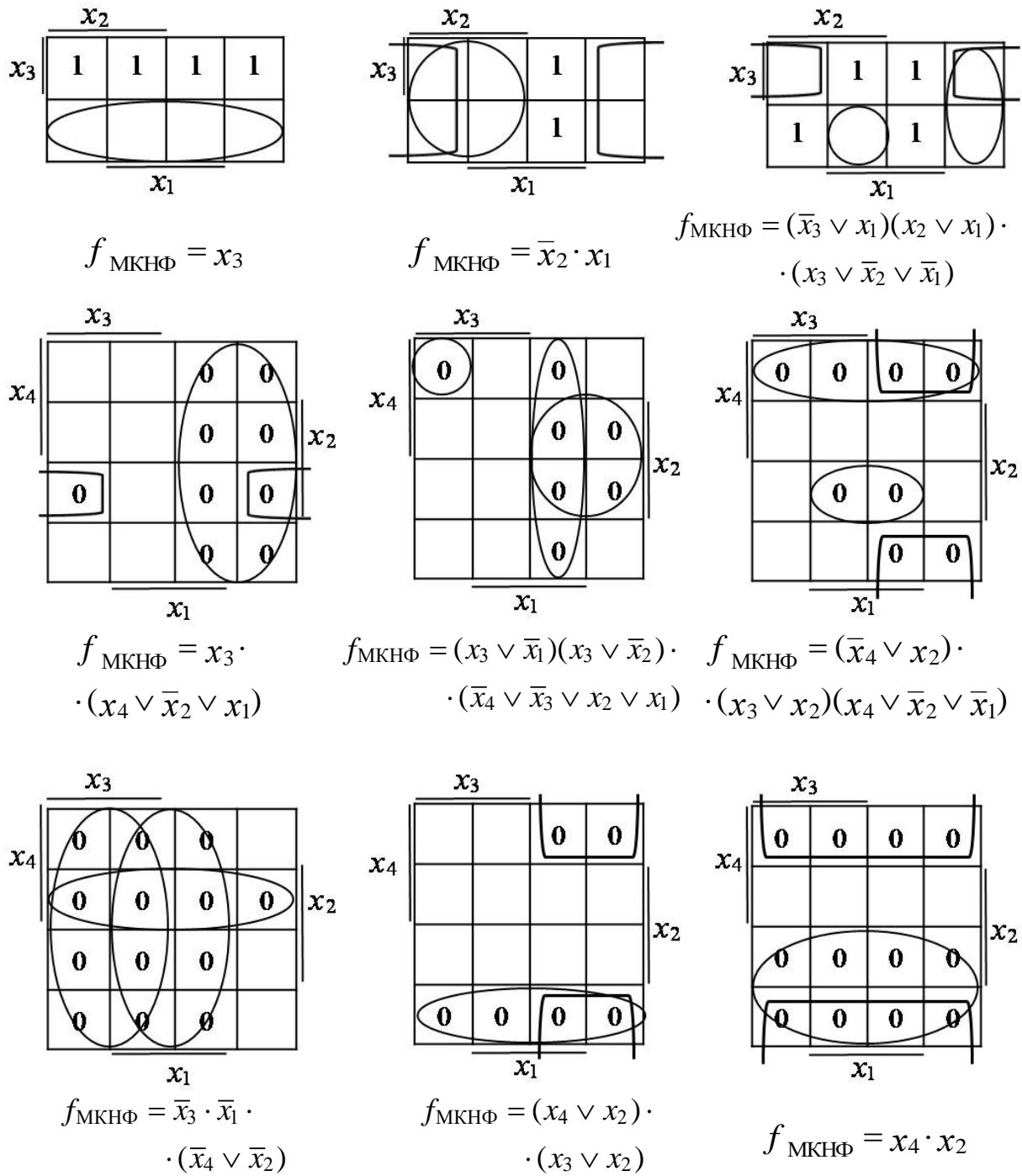


Рис. 3.25. Деякі приклади знаходження ДКНФ функцій від 3-х та 4-х змінних

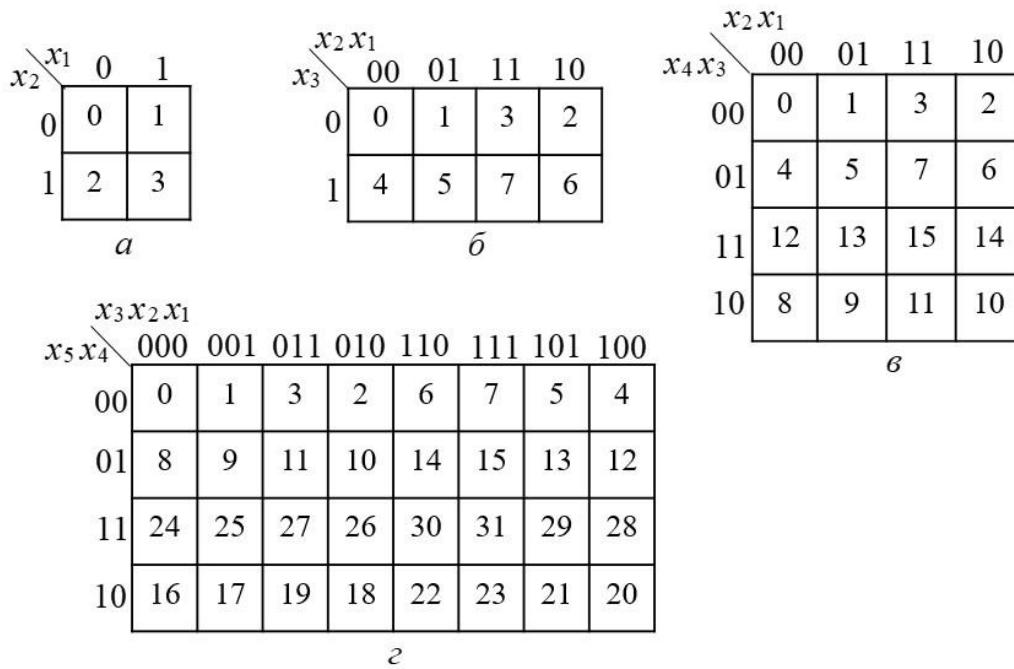


Рис. 3.26. Карти Карно для перемикальних функцій від:
a – двох; *b* – трьох;
v – чотирьох; *z* – п'яти змінних

Приклади мінімізації перемикальних функцій за допомогою карт Карно наведено на рис. 3.27.

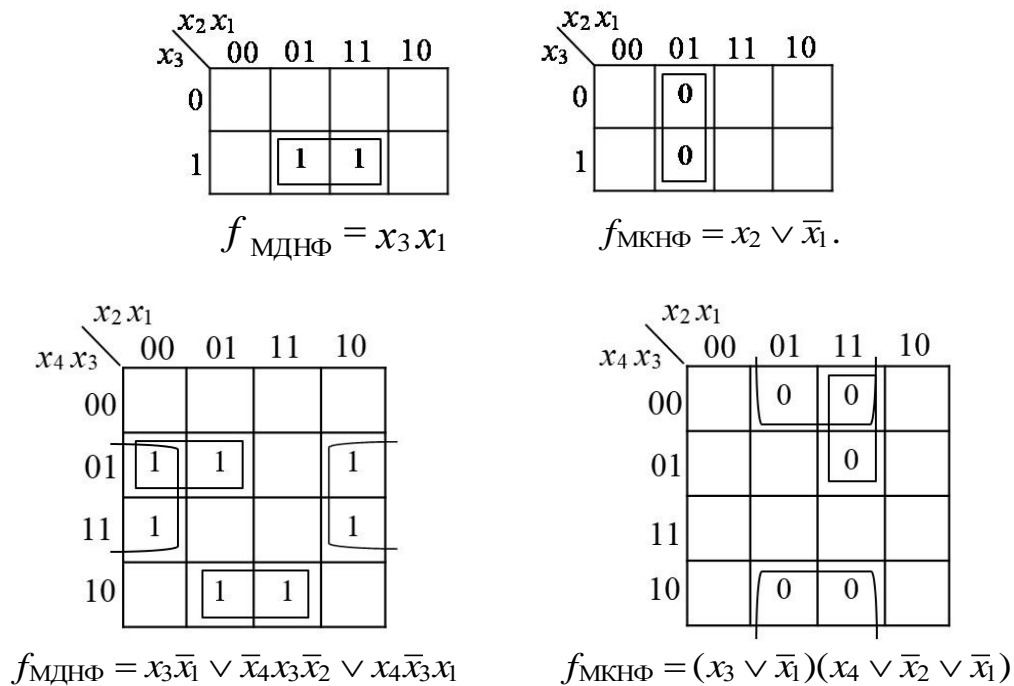


Рис. 3.27. Деякі приклади мінімізації перемикальних функцій від 3-х та 4-х змінних за допомогою карт Карно

Карта Карно відрізняється від діаграми Вейча лише порядком нумерації клітинок (рис. 3.28).

Обидва графічні методи під час мінімізації функцій дають однакові результати. Оскільки графічні методи мінімізації є ефективними лише за невеликої кількості змінних, то їх часто називають ручними.

Недоліком графічних методів є те, що процес відшукування простих імплікант є суто візуальним та неформалізованим, тому успішність результату залежить від кваліфікації виконавця.

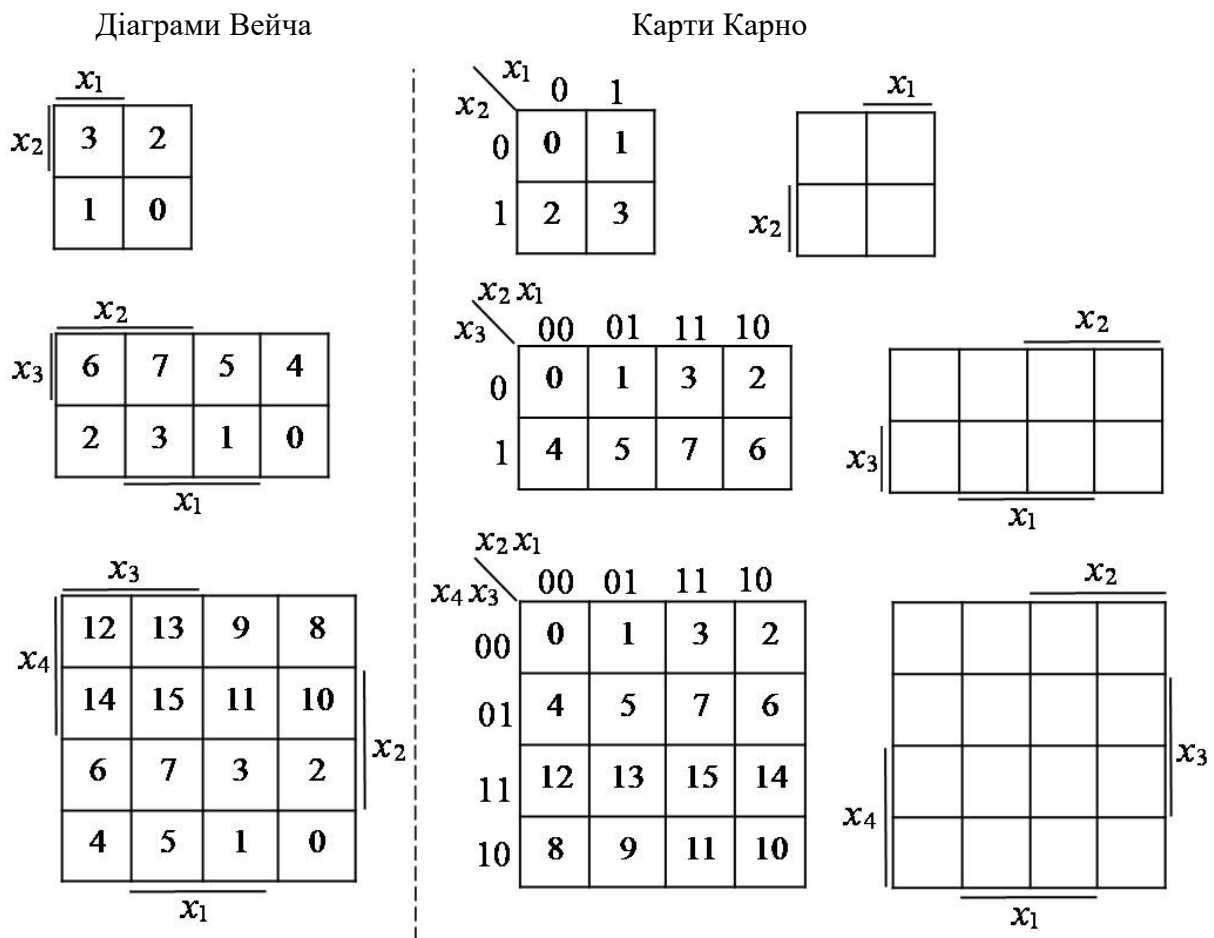


Рис. 3.28. Порівняння діаграм Вейча та карт Карно

Заятання та завдання

1. Які конституенти одиниці (нуля) називаються сусідніми? Наведіть приклади сусідніх конституент одиниці (нуля).

2. На якому правилі булевої алгебри ґрунтуються графічні методи мінімізації перемикальних функцій? Запишіть це правило в диз'юнктивній та кон'юнктивній формах.
3. Яку кількість сусідніх клітинок (конституент одиниці або конституент нуля) можна склеювати на діаграмах Вейча (картах Карно)?
4. Чим карти Карно відрізняються від діаграм Вейча?
5. Охарактеризуйте процес мінімізації перемикальної функції за допомогою діаграми Вейча (карти Карно)?
6. Назвіть переваги та недоліки графічних методів мінімізації функцій.
7. Порівняйте аналітичні та графічні методи мінімізації перемикальних функцій.

Задачі для самостійного розв'язування

1. За допомогою діаграми Вейча знайти МДНФ та МКНФ перемикальної функції від трьох змінних, яка дорівнює одиниці на наборах 0, 1, 2, 5, 7 та нулю – на решті наборів.
2. Перемикальна функція від чотирьох змінних дорівнює нулю на наборах 0, 1, 9, 10, 11, 12, 13, 15 та одиниці – на решті наборів. Знайти МДНФ та МКНФ функції за допомогою карти Карно.
3. Перемикальна функція від чотирьох змінних дорівнює одиниці на наборах 5, 6, 7, 8, 13, 14, 15 та нулю – на решті наборів. Мінімізувати функцію за допомогою діаграми Вейча та подати її в формі АБО-НЕ/АБО-НЕ.
4. Перемикальна функція від трьох змінних дорівнює нулю на наборах 0, 3, 4 та одиниці – на решті наборів. Мінімізувати функцію за допомогою карти Карно та подати її в формі І-НЕ/І-НЕ.
5. Побудувати комбінаційну схему з мінімальною апаратною складністю в елементному базисі 2І, 2АБО-НЕ, яка реалізує перемикальну функцію від 4-х змінних, що дорівнює одиниці на наборах 5, 6, 7, 8, 13, 14, 15 і нулю – на решті наборів. Мінімізацію функції виконати за допомогою діаграми Вейча.

3.6. Мінімізація неповністю визначених перемикальних функцій

У реальних цифрових схемах можливі випадки, коли:

- ✓ деякі комбінації (набори) значень змінних є забороненими для даної логічної схеми, або не всі комбінації значень змінних подають на входи схеми в процесі функціонування пристрою (системи); у цьому випадку значення функції на таких

наборах позначають "-" (прочерком), наприклад, на рис. 3.29б набір <11> є забороненим;

- ✓ на деяких наборах значень змінних функція може набувати довільне значення (0 або 1), і це не впливає на загальну поведінку пристрою (системи); у цьому випадку значення функції на таких наборах позначають "×" (хрестиком), наприклад, на рис. 3.29в для функції z дозволяється, щоб на наборі <00> функція набувала довільного значення (0 або 1).

У перелічених випадках вважають, що функція задана не на всіх 2^n наборах значень аргументів, і її вважають *неповністю визначеною* перемикальною функцією (або *частково визначеною*).

x_2	x_1	y	x_2	x_1	f	x_2	x_1	z	
0	0	0	0	0	1	0	0	×	
0	1	1	0	1	0	0	1	0	
1	0	1	1	0	1	1	0	1	
1	1	0	1	1	–	1	1	1	
a					b				v

Рис. 3.29. Види перемикальних функцій:

a – повністю визначена; b, v – неповністю визначені перемикальні функції

Як мінімізувати частково визначену перемикальну функцію?

Мінімізацію частково визначених перемикальних функцій можна виконувати будь-яким з раніше розглянутих методів мінімізації. Але у процесі мінімізації необхідно довизначити перемикальну функцію на наборах, на яких функція не визначена, таким чином, щоб досягти максимального спрощення функції.

Розглянемо мінімізацію неповністю визначених перемикальних функцій аналітичними методами.

Особливістю застосування аналітичних методів для мінімізації неповністю визначених функцій є те, що перемикальну функцію довизначають в такий спосіб – на наборах, на яких функція не визначена, значення функції прирівнюють до одиниці. Далі виконують перший етап мінімізації – знаходження скороченої ДНФ. Особливістю другого етапу мінімізації є те, що конституенти одиниці тих наборів, на яких функція не визначена, не включають у заголовки стовпців таблиці покриття функції, або, у випадку застосування способу Петріка знаходження тупикових ДНФ, – не формують логічні умови покриття тих конститuent одиниці, що відповідають наборам, на яких функція не визначена.

Задача 3.10. Мінімізувати неповністю визначену функцію y , задану таблицею істинності (табл. 3.6), методом Квайна.

Таблиця 3.6
Таблиця істинності
неповністю
визначеної функції у

x_3	x_2	x_1	y
0	0	0	–
0	0	1	–
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	–

Розв’язування. Без урахування наборів, на яких функція не визначена, ДДНФ функції має вигляд:

$$y = 2 \vee 4 \vee 5 = \bar{x}_3 x_2 \bar{x}_1 \vee x_3 \bar{x}_2 \bar{x}_1 \vee x_3 \bar{x}_2 x_1$$

($\mathcal{C} = 9$).

Доповнимо ДДНФ функції конститuentами одиниці, що відповідають наборам, на яких функція не визначена (це – набори 0, 1, 7):

$$y = 2 \vee 4 \vee 5 \vee 0^* \vee 1^* \vee 7^* = \bar{x}_3 x_2 \bar{x}_1 \vee x_3 \bar{x}_2 \bar{x}_1 \vee x_3 \bar{x}_2 x_1 \vee \bar{x}_3 \bar{x}_2 \bar{x}_1 \vee \bar{x}_3 \bar{x}_2 x_1 \vee x_3 x_2 x_1.$$

Виконаємо мінімізацію доповненої функції у методом Квайна. Для знаходження скороченої ДНФ виконаємо спочатку всі можливі операції неповного склеювання конститuent одиниці та імплікант функції, а потім всі можливі операції поглинання (рис. 3.30).

Конститuentи одиниці	Імпліканти – терми 2-го рангу	Імпліканти – терми 1-го рангу
(1) $\bar{x}_3 x_2 \bar{x}_1$	(1-4) $\bar{x}_3 \bar{x}_1$	(2-3, 4-5) \bar{x}_2
(2) $x_3 \bar{x}_2 \bar{x}_1$	(2-3) $x_3 \bar{x}_2$	(2-4, 3-5)
(3) $x_3 \bar{x}_2 x_1$	(2-4) $\bar{x}_2 \bar{x}_1$	
(4) $\bar{x}_3 \bar{x}_2 \bar{x}_1$	(3-5) $\bar{x}_2 x_1$	
(5) $\bar{x}_3 \bar{x}_2 x_1$	(3-6) $x_3 x_1$	
(6) $x_3 x_2 x_1$	(4-5) $\bar{x}_3 \bar{x}_2$	

$$y_{\text{СДНФ}} = \bar{x}_2 \vee x_3 x_1 \vee \bar{x}_3 \bar{x}_1$$

Рис. 3.30. Визначення скороченої ДНФ функції у

Результатом першого етапу мінімізації є отримання скороченої ДНФ функції:

$$y_{\text{СДНФ}} = \bar{x}_2 \vee x_3 x_1 \vee \bar{x}_3 \bar{x}_1.$$

Виконаємо другий етап мінімізації – знаходження тупикових ДНФ та визначення серед них мінімальної ДНФ. Для цього побудуємо таблицю покриття (імплікантну матрицю) функції y (табл. 3.7). Стівпцями таблиці є лише ті конституенти одиниці, що відповідають наборам, на яких функція визначена, тобто конституенти $\bar{x}_3 x_2 \bar{x}_1$ (набір 2), $x_3 \bar{x}_2 \bar{x}_1$ (набір 4) та $x_3 \bar{x}_2 x_1$ (набір 5).

Таблиця 3.7

Таблиця покриття неповністю визначеної функції y

	Прості імпліканти	Конституенти одиниці		
		$\bar{x}_3 x_2 \bar{x}_1$	$x_3 \bar{x}_2 \bar{x}_1$	$x_3 \bar{x}_2 x_1$
1	\bar{x}_2		v.	v
2	$x_3 x_1$			v
3	$\bar{x}_3 \bar{x}_1$	v.		

З табл. 3.7 визначаємо єдину тупикову ДНФ: $y_{ТДНФ} = \bar{x}_3 \bar{x}_1 \vee \bar{x}_2$.

Ця ТДНФ й буде мінімальною формою функції.

Отже, $y_{МДНФ} = \bar{x}_3 \bar{x}_1 \vee \bar{x}_2$ ($C = 3$). ■

Задача 3.11. Мінімізувати неповністю визначену функцію z , задану таблицею істинності (табл. 3.8), методом Квайна – Мак-Класкі.

Таблиця 3.8

Таблиця істинності неповністю визначеної функції z

x_3	x_2	x_1	z
0	0	0	×
0	0	1	1
0	1	0	×
0	1	1	0
1	0	0	–
1	0	1	0
1	1	0	1
1	1	1	1

Визначення тупикових ДНФ виконати способом Петріка.

Розв’язування. Оскільки функція z може набувати довільне значення (0 або 1) на наборах 0 та 2, а набір 4 є забороненим (див. табл. 3.8), то спочатку сформуємо ДНФ функції з урахуванням лише тих наборів, на яких функція дорівнює одиниці:

$$z = 1 \vee 6 \vee 7 = \bar{x}_3 \bar{x}_2 x_1 \vee x_3 x_2 \bar{x}_1 \vee x_3 x_2 x_1 \quad (C = 9)$$

Довизначимо функцію z на наборах 0, 2, 4 – вважатимемо, що на цих наборах вона дорівнює одиниці.

Тоді

$$z = 1 \vee 6 \vee 7 \vee 0^* \vee 2^* \vee 4^* =$$

$$= \bar{x}_3 \bar{x}_2 x_1 \vee x_3 x_2 \bar{x}_1 \vee x_3 x_2 x_1 \vee \bar{x}_3 \bar{x}_2 \bar{x}_1 \vee \bar{x}_3 x_2 \bar{x}_1 \vee x_3 \bar{x}_2 \bar{x}_1.$$

Виконаємо мінімізацію доповненої функції методом Квайна – Мак-Класкі.

Запишемо функцію у цифровій формі:

$$z = 001 \vee 110 \vee 111 \vee 000 \vee 010 \vee 100.$$

Знайдемо скорочену ДНФ функції, виконавши спочатку всі можливі операції неповного склеювання конститuent одиниці та імплікант функції, а потім всі можливі операції поглинання (рис. 3.31).

Результатом першого етапу мінімізації є Z-покриття, диз'юнкція компонентів якого дає скорочену ДНФ функції:

$$z_{\text{СДНФ}} = \times \times 0 \vee 11 \times \vee 00 \times.$$

$$K^0 = \left\{ \begin{array}{l} (1) \underline{000} \\ (2) \underline{001} \\ (3) \underline{010} \\ (4) \underline{100} \\ (5) \underline{110} \\ (6) \underline{111} \end{array} \right\} K^1 = \left\{ \begin{array}{l} (1-2) \underline{00} \times \\ (1-3) \underline{0} \times \underline{0} \\ (1-4) \underline{\times} 00 \\ (3-5) \underline{\times} 10 \\ (4-5) \underline{1} \times 0 \\ (5-6) \underline{11} \times \end{array} \right\} K^2 = \left\{ \begin{array}{l} (1-3, 4-5) \underline{\times \times 0} \\ (1-4, 3-5) \underline{\quad} \end{array} \right\} \Rightarrow Z = \left\{ \begin{array}{l} \times \times 0 \\ 11 \times \\ 00 \times \end{array} \right\}$$

$$z_{\text{СДНФ}} = \times \times 0 \vee 11 \times \vee 00 \times$$

Рис. 3.31. Визначення скороченої ДНФ функції z

Виконаємо другий етап мінімізації – знаходження тупикових ДНФ способом Петріка та визначення серед них мінімальної ДНФ.

Для цього сформуємо множину KONST конститuent одиниці функції z, включивши до неї лише ті конститuentи одиниці, що відповідають наборам, на яких функція визначена та дорівнює одиниці, тобто конститuentи 001 (набір 1), 110 (набір 6) та 111 (набір 7)

$$\textcircled{1} \quad \textcircled{2} \quad \textcircled{3}$$

$$\text{KONST} = \{001, 110, 111\},$$

та множину IMPL простих імплікант, які входять до складу СДНФ функції

$$\text{IMPL} = \{\times \times 0, 11 \times, 00 \times\}.$$

Запишемо логічні умови покриття конститuent одиниці з множини KONST:

$$KO1: 001 \rightarrow C, \quad KO2: 110 \rightarrow A \vee B, \quad KO3: 111 \rightarrow B.$$

Сформуємо логічну умову покриття перемикальної функції z як кон'юнкцію логічних умов покриття її конститuent одиниці: $C \cdot (A \vee B) \cdot B$.

Виконавши операцію поглинання отримаємо: CB .

Отже, функція z має одну тупикову ДНФ, а саме:

$$z_{\text{ДНФ}} = C \vee B = 00\times \vee 11\times = \bar{x}_3 \bar{x}_2 \vee x_3 x_2.$$

Відповідно, мінімальна ДНФ функції має вигляд

$$z_{\text{МДНФ}} = \bar{x}_3 \bar{x}_2 \vee x_3 x_2 \quad (C = 4). \blacksquare$$

Розглянемо мінімізацію неповністю визначених перемикальних функцій графічними методами. Ці методи є ефективними, якщо частково визначені перемикальні функції залежать від невеликої кількості змінних.

Головним питанням під час мінімізації є, як довизначити перемикальну функцію на тих наборах, на яких вона не визначена.

Стратегія довизначення при пошуку МДНФ – невизначені значення функції (прочерки, хрестики) розглядають як одиниці в тих випадках, коли це приводить до збільшення розміру склеюваного прямокутника, що відповідає імпліканті, інакше – прочерки, хрестики розглядають як нулі.

Стратегія довизначення при пошуку МКНФ – невизначені значення функції (прочерки, хрестики) розглядають як нулі в тих випадках, коли це приводить до збільшення розміру склеюваного прямокутника, інакше – прочерки, хрестики розглядають як одиниці.

Задача 3.12. Знайти МДНФ частково визначеної перемикальної функції f , заданої діаграмою Вейча

		x_3			
		–	×	1	0
x_4		1	×	1	1
		0	–	1	1
		0	–	1	0
		x_1			

f

Розв'язування. З метою знаходження форми функції з мінімальною ціною (початкова ціна становить $C = 28$) умовно замінимо прочерки, хрестики у другому стовпці на одиниці – щоб утворити прямокутник з 8-ми клітинок, якому відповідає імпліканта x_1 (рис. 3.32). Далі утворимо ядро функції $x_1, \bar{x}_3 x_2$ (рис. 3.32а). Непокритою залишається одиниця в клітинці з координатами 1110. Її

можна покрити двома способами – включити в прямокутник, утворений з клітинок другого рядка діаграми (рис. 3.32б), або включити в прямокутник, утворений чотирма клітинками у лівому верхньому куті діаграми (рис. 3.32в).

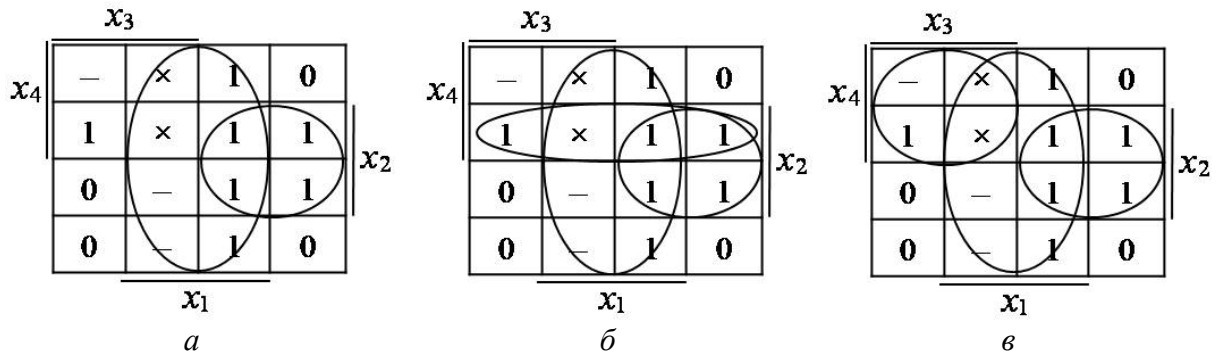


Рис. 3.32. Етапи мінімізації функції f

У першому випадку отримаємо тупикову ДНФ функції

$$f_{\text{ТДНФ}(\delta)} = x_1 \vee \bar{x}_3 x_2 \vee x_4 x_2. \quad (Ц = 5)$$

При цьому прочерк у клітинці з координатами 1100 буде до-
визначено як 0.

У другому випадку отримаємо іншу тупикову форму функції

$$f_{\text{ТДНФ}(\epsilon)} = x_1 \vee \bar{x}_3 x_2 \vee x_4 x_3, \quad (Ц = 5)$$

але при цьому прочерк у клітинці з координатами 1100 буде до-
визначено як 1.

Оскільки обидві тупикові ДНФ мають однакову ціну, то будь-яку з
них можна вважати мінімальною ДНФ функції:

$$f_{\text{МДНФ}} = \begin{cases} f_{\text{ТДНФ}(\delta)} \\ f_{\text{ТДНФ}(\epsilon)} \end{cases} \quad Ц = 5 \blacksquare$$

Задача 3.13. Знайти МКНФ частково визначеної функції y , якій
відповідає задана карта Карно

		$x_2 x_1$			
		00	01	11	10
$x_4 x_3$	00	×	—	1	0
	01	1	—	0	1
	11	0	—	0	1
	10	0	×	1	0

Розв'язування. Утворимо три
прямокутники (групи), які покривають всі нулі
на карті Карно (рис. 3.33), їм відповідають три
прості імпліканти, які є складовими
частинами МКНФ функції:

$$f_{\text{МКНФ}} = (\bar{x}_3 \vee \bar{x}_1)(\bar{x}_4 \vee x_2)(x_3 \vee x_1).$$

При цьому усі клітинки, крім однієї, у
яких розміщені прочерки, хрестики до-
визначені значенням 0, а прочерк у клітинці з
координатами 0001 до-визначено як 1. \blacksquare

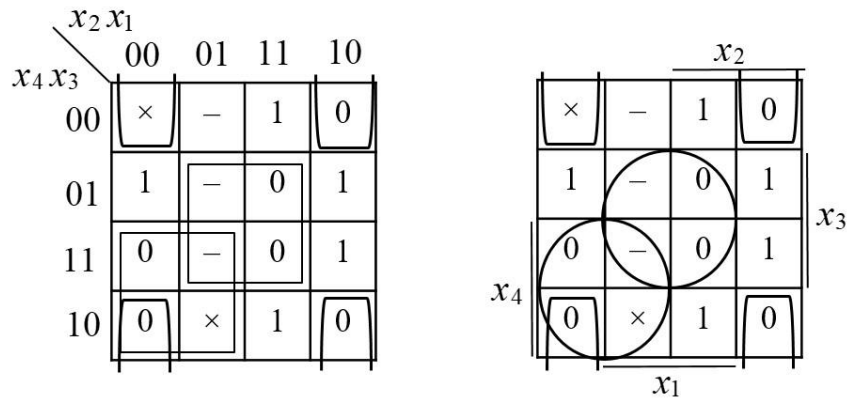


Рис. 3.33. Мінімізація неповністю визначеної функції у за допомогою карти Карно (2 способи позначення карти)

Деякі інші приклади мінімізації неповністю визначених функцій за допомогою діаграм Вейча подано на рис. 3.34.

Аналіз розглянутих методів мінімізації щодо визначення області їх ефективності показує наступне.

Аналітичні методи мінімізації перемикальних функцій (Квайна, Квайна – Мак-Класкі та інші) забезпечують знаходження форм функцій з мінімальною ціною лише для диз'юнктивних форм подання функції (МДНФ у, МДНФ \bar{y}).

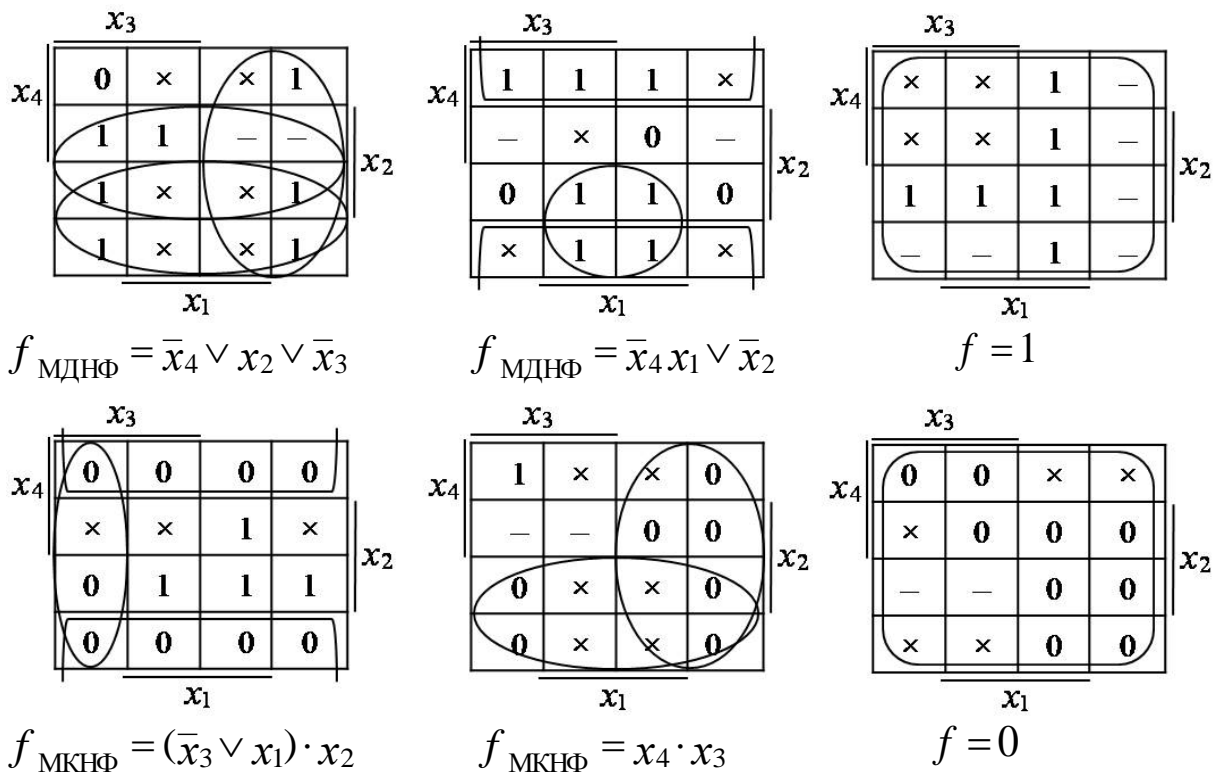


Рис. 3.34. Деякі приклади мінімізації неповністю визначених перемикальних функцій від 4-х змінних

Графічні методи мінімізації (діаграми Вейча, карти Карно) забезпечують знаходження форм функції з мінімальною ціною лише для диз'юнктивної і/або кон'юнктивної форми подання функції (МДНФ у, МКНФ у).

Розглянуті методи мінімізації перемикальних функцій у загальному випадку не забезпечують отримання абсолютно мінімальної ціни для довільної форми подання перемикальної функції.

Наприклад, нехай результатом мінімізації є МДНФ перемикальної функції у:

$$f_{\text{МДНФ}} = x_4 \bar{x}_3 x_2 \vee x_3 \bar{x}_2 \vee x_3 x_1. \quad (Ц = 7)$$

Однак, отриману форму можна перетворити в іншу (не диз'юнктивну) форму

$$y = x_4 \bar{x}_3 x_2 \vee x_3 (\bar{x}_2 \vee x_1), \quad (Ц = 6)$$

яка має меншу ціну, ніж МДНФ. Це означає, що після отримання функції з мінімальною ціною (МДНФ або МКНФ) за допомогою одного з розглянутих методів мінімізації, результат мінімізації для його подальшого спрощення – якщо це необхідно, слід додатково аналізувати та застосовувати евристичні прийоми перетворення – групування членів, винесення спільної змінної за дужки тощо, які є неформалізованими.

Запитання та завдання

1. Яку перемикальну функцію називають неповністю (частково) визначеною?
2. Наведіть приклад неповністю визначеної перемикальної функції, у якої деякі набори значень змінних є забороненими.
3. Наведіть приклад неповністю визначеної перемикальної функції, яка на деяких наборах значень змінних може набувати довільне значення (0 або 1).
4. Сформулюйте стратегію довизначення частково визначеної перемикальної функції в процесі її мінімізації: а) аналітичним методом; б) графічним методом.
5. Сформулюйте область ефективного застосування аналітичних (Квайна, Квайна – Мак-Класкі) та графічних методів (діаграми Вейча, карти Карно) мінімізації перемикальних функцій.

Задачі для самостійного розв'язування

1. Методом Квайна – Мак-Класкі мінімізувати перемикальну функцію від 3-х змінних, яка дорівнює одиниці на наборах 0, 2, 7, нулю – на наборах 3, 4, 6, на наборі 5 – може набувати довільне значення (0 або 1), а набір 1 є забороненим.
2. Перемикальна функція від чотирьох змінних дорівнює нулю на наборах 1, 2, 13, одиниці – на наборах 0, 3, 4, 8, 9, 12, на наборах 6, 11, 14 може набувати довільне значення (0 або 1), а набори 5, 7, 10, 15 є забороненими. Мінімізувати функцію методом Квайна, визначення тупикових ДНФ виконати способом Петріка.
3. Відомо, що перемикальна функція від чотирьох змінних дорівнює одиниці на наборах 1, 3, 6, 9, 12, 14, 15, нулю – на наборах 0, 2, 5, 8, 10, 13 та не визначена на решті наборів. Знайти МДНФ функції за допомогою карти Карно.
4. На основі логічних елементів 2АБО-НЕ побудувати комбінаційну схему, що реалізує перемикальну функцію від 4-х змінних, яка дорівнює одиниці на наборах 1, 2, 4, 6, 9, 11, нулю – на наборах 0, 3, 5, 7, 10, 12, 13 та не визначена на решті наборів. Цільова функція проектування – мінімальні апаратні витрати. Мінімізацію функції виконати за допомогою діаграми Вейча.
5. В елементному базисі 2І-НЕ побудувати оптимізовану за показником апаратної складності комбінаційну схему, що реалізує перемикальну функцію від 4-х змінних, яка дорівнює нулю на наборах 2, 6, 8, 9, 11, 14, на наборах 0, 1, 13 – не визначена, а на решті наборів дорівнює одиниці.

3.7. Мінімізація систем перемикальних функцій

На практиці часто необхідно створювати логічні схеми з кількома виходами.

Наприклад, нехай необхідно побудувати комбінаційну схему, що реалізує деякий вузол (перетворювач кодів), що має три входи та три виходи, а його функціонування описується таблицею істинності (табл. 3.9).

Таблиця 3.9
Таблиця істинності
перетворювача кодів

x_3	x_2	x_1	y_1	y_2	y_3
0	0	0	0	1	1
0	0	1	1	0	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	0	0	1
1	1	1	1	1	1

Розв'язати цю задачу можна наступним чином.

Можна вважати, що виходи перетворювача не залежать один від одного. Тоді поведінку перетворювача можна описати трьома незалежними перемикальними функціями y_1 , y_2 , y_3 , кожній з яких відповідає один стовпець в таблиці істинності.

За такого підходу задача зводиться до побудови трьох окремих (незалежних) комбінаційних схем, кожна з яких реалізує один вихід перетворювача. При цьому в незалежних схемах

можливе дублювання однакових логічних елементів з однаковими логічними сигналами на входах. Щоб уникнути повторного використання однакових елементів в окремих схемах необхідно функції, що описують поведінку перетворювача, розглядати не як незалежні, а як систему перемикальних функцій, і виконувати їх спільну мінімізацію.

Метою мінімізації системи перемикальної функції є знаходження мінімальної форми кожної з функцій, які б сукупно забезпечували мінімальну ціну системи перемикальних функцій.

Виконаємо спільну мінімізацію системи перемикальних функцій y_1 , y_2 , y_3 , що задані табл. 3.9.

1. Запишемо функції в ДДНФ:

$$\begin{cases} y_1 = \bar{x}_3 \bar{x}_2 x_1 \vee \bar{x}_3 x_2 x_1 \vee x_3 \bar{x}_2 \bar{x}_1 \vee x_3 \bar{x}_2 x_1 \vee x_3 x_2 x_1, \\ y_2 = \bar{x}_3 \bar{x}_2 \bar{x}_1 \vee \bar{x}_3 x_2 \bar{x}_1 \vee x_3 \bar{x}_2 \bar{x}_1 \vee x_3 \bar{x}_2 x_1 \vee x_3 x_2 x_1, \\ y_3 = \bar{x}_3 \bar{x}_2 \bar{x}_1 \vee \bar{x}_3 x_2 \bar{x}_1 \vee x_3 \bar{x}_2 x_1 \vee x_3 x_2 \bar{x}_1 \vee x_3 x_2 x_1. \end{cases}$$

2. Утворимо множину A , елементами якої є конституенти одиниці функцій y_1 , y_2 , y_3 . Кожній конституенті одиниці з множини A присвоїмо ознаку належності даної конституенти до тієї чи іншої функції системи.

Наприклад, запис $\bar{x}_3 \bar{x}_2 \bar{x}_1 \{2,3\}$ означає, що конституента одиниці $\bar{x}_3 \bar{x}_2 \bar{x}_1$ входить до складу функції y_2 та функції y_3 .

Ознака належності терма (конституенти одиниці) є множиною, елементами якої є номери функцій, до складу яких входить даний терм (конституента одиниці).

Множина A має вигляд:

$$A = \{ \bar{x}_3 \bar{x}_2 \bar{x}_1 \{2, 3\}; \bar{x}_3 \bar{x}_2 x_1 \{1\}; \bar{x}_3 x_2 \bar{x}_1 \{2, 3\}; \bar{x}_3 x_2 x_1 \{1\}; \\ x_3 \bar{x}_2 \bar{x}_1 \{1, 2\}; x_3 \bar{x}_2 x_1 \{1, 2, 3\}; x_3 x_2 \bar{x}_1 \{3\}; x_3 x_2 x_1 \{1, 2, 3\}.$$

3. Утворимо функцію φ як диз'юнкцію всіх елементів множини A :

$$\varphi = \{ \bar{x}_3 \bar{x}_2 \bar{x}_1 \{2, 3\} \vee \bar{x}_3 \bar{x}_2 x_1 \{1\} \vee \bar{x}_3 x_2 \bar{x}_1 \{2, 3\} \vee \bar{x}_3 x_2 x_1 \{1\} \vee \\ \vee x_3 \bar{x}_2 \bar{x}_1 \{1, 2\} \vee x_3 \bar{x}_2 x_1 \{1, 2, 3\} \vee x_3 x_2 \bar{x}_1 \{3\} \vee x_3 x_2 x_1 \{1, 2, 3\}.$$

4. Знайдемо скорочену ДНФ функції φ . Для цього можна скористатися будь-яким з аналітичних методів мінімізації перемикальних функцій.

Знайдемо скорочену СДНФ функції φ , використовуючи метод мінімізації Квайна.

Відповідно до теореми Квайна, якщо в ДДНФ функції φ виконати всі можливі операції неповного склеювання, а потім усі можливі операції поглинання, то дістанемо скорочену ДНФ функції φ , тобто диз'юнкцію всіх простих імплікант функції.

Скорочена ДНФ функції φ й буде скороченою ДНФ системи перемикальних функцій y_1, y_2, y_3 .

Проте, теорему Квайна слід застосовувати до функції φ , враховуючи ознаки належності конститuent одиниці (термів), а саме:

- 1) правило неповного склеювання можна застосовувати лише до таких двох конститuent одиниці (термів), ознаки належності яких мають хоча б один спільний елемент – номер функції, до складу якої вони входять;
- 2) при склеюванні двох конститuent одиниці (термів) новоотриманому терму слід присвоїти ознаку належності, що є перерізом ознак належності склеюваних конститuent одиниці (термів) (перерізом двох множин);
- 3) правило поглинання застосовують лише до таких двох термів, у яких ознаки належності (дві множини) повністю співпадають.

Виконаємо спочатку можливі склеювання конститuent одиниці (рис. 3.35).

Наприклад, склеювання конститuent (1) та (3) дає результат (1-3) $\bar{x}_3 \bar{x}_1 \{2, 3\}$, а склеювання конститuent (1) та (5) – (1-5) $\bar{x}_2 \bar{x}_1 \{2\}$.

Конституенти одиниці	Імпліканти- терми 3-го рангу	Імпліканти-терми 2-го рангу
(1) $\bar{x}_3 \bar{x}_2 \bar{x}_1 \{2, 3\}$	(1-3) $\bar{x}_3 \bar{x}_1 \{2, 3\}$	(2-4, 6-8) $x_1 \{1\}$
(2) $\bar{x}_3 \bar{x}_2 x_1 \{1\}$	(1-5) $\bar{x}_2 \bar{x}_1 \{2\}$	(2-6, 4-8)
(3) $\bar{x}_3 x_2 \bar{x}_1 \{2, 3\}$	(2-4) $\bar{x}_3 x_1 \{1\}$	
(4) $\bar{x}_3 x_2 x_1 \{1\}$	(2-6) $\bar{x}_2 x_1 \{1\}$	
(5) $x_3 \bar{x}_2 \bar{x}_1 \{1, 2\}$	(3-7) $x_2 \bar{x}_1 \{3\}$	
(6) $x_3 \bar{x}_2 x_1 \{1, 2, 3\}$	(4-8) $x_2 x_1 \{1\}$	
(7) $x_3 x_2 \bar{x}_1 \{3\}$	(5-6) $x_3 \bar{x}_2 \{1, 2\}$	
(8) $x_3 x_2 x_1 \{1, 2, 3\}$	(6-8) $x_3 x_1 \{1, 2, 3\}$	
	(7-8) $x_3 x_2 \{3\}$	

$$\varphi_{\text{СДНФ}} = x_1 \{1\} \vee x_3 x_2 \{3\} \vee x_3 x_1 \{1, 2, 3\} \vee x_3 \bar{x}_2 \{1, 2\} \vee x_2 \bar{x}_1 \{3\} \vee \bar{x}_2 \bar{x}_1 \{2\} \vee \bar{x}_2 x_1 \{2, 3\}$$

Рис. 3.35. Склеювання і поглинання конститuent одиниці та імпліканти функції φ

Серед імпліканти 2-го рангу можливі лише два випадки склеювання – (2-4) і (6-8), а також (2-6) і (4-8), вони дають однаковий результат – імпліканти першого рангу $x_1 \{1\}$. Подальші операції склеювання неможливі.

Далі виконаємо можливі операції поглинання.

Наприклад, терм $x_1 \{1\}$ поглинає терми (4-8) $x_2 x_1 \{1\}$, (2-6) $\bar{x}_2 x_1 \{1\}$, (2-4) $\bar{x}_3 x_1 \{1\}$, а також конститuentи одиниці (4) $\bar{x}_3 x_2 x_1 \{1\}$ та (2) $\bar{x}_3 \bar{x}_2 x_1 \{1\}$.

Таким чином, після виконання всіх можливих операцій неповного склеювання та поглинання отримуємо скорочену ДНФ функції φ :

$$\varphi_{\text{СДНФ}} = x_1 \{1\} \vee x_3 x_2 \{3\} \vee x_3 x_1 \{1, 2, 3\} \vee x_3 \bar{x}_2 \{1, 2\} \vee x_2 \bar{x}_1 \{3\} \vee \bar{x}_2 \bar{x}_1 \{2\} \vee \bar{x}_2 x_1 \{2, 3\},$$

а, отже, й множину простих імпліканти системи перемикальних функцій y_1, y_2, y_3 .

5. Побудуємо таблицю покриття системи функцій y_1, y_2, y_3 (табл. 3.10). Стівцями таблиці є конститuentи одиниці окремо кожної з функцій – y_1, y_2, y_3 , а рядками – прості імпліканти

Таблиця 3.10

Таблиця покриття системи перемикальних функцій y_1, y_2, y_3

Прості імпліканти	Конституенти одиниці														
	y_1			y_2			y_3								
	001	011	100	101	111	000	010	100	101	111	000	010	101	110	111
$x_1 \{1\}$	\checkmark	\checkmark		\checkmark	\checkmark										
$x_3 x_2 \{3\}$														\checkmark	\checkmark
$x_3 x_1 \{1, 2, 3\}$				\checkmark	\checkmark			\checkmark	\checkmark	\checkmark			\checkmark		\checkmark
$x_3 \bar{x}_2 \{1, 2\}$			\checkmark	\checkmark				\checkmark	\checkmark						
$x_2 \bar{x}_1 \{3\}$													\checkmark	\checkmark	
$\bar{x}_2 \bar{x}_1 \{2\}$						\checkmark					\checkmark				
$\bar{x}_3 \bar{x}_1 \{2, 3\}$						\checkmark	\checkmark				\checkmark	\checkmark	\checkmark	\checkmark	

системи функцій. З метою скорочення довжини записів конституенти одиниці подано в цифровій формі.

б. На перетині рядків і стовпців ставимо позначки \vee окремо для кожної з функцій, і лише тоді, коли:

- а) номер поточної функції, що розглядається, є складовою частиною ознаки належності даної простої імпліканти,
- б) дана проста імпліканта є власною частиною відповідної конституенти одиниці.

Наприклад, для простої імпліканти $\bar{x}_3 \bar{x}_1 \{2, 3\}$ позначки \vee ставимо лише для функцій y_2 та y_3 , а саме в стовпцях, що мають структуру 0×0 , де \times – довільне значення (0 або 1), тобто в стовпцях 000 та 010.

Для кожної з функцій y_1, y_2, y_3 визначаємо ядро, тупикові ДНФ, а серед них – мінімальну ДНФ.

Так, для функції y_1 ядро утворюють прості імпліканти x_1 та $x_3 \bar{x}_2$. Оскільки ядро покриває всі конституенти одиниці (стовпці) функції y_1 , то

$$y_{1, \text{мднф}} = x_1 \vee x_3 \bar{x}_2.$$

Для функції y_2 ядро утворюють прості імпліканти $\bar{x}_3 \bar{x}_1$ та $x_3 x_1$. Для цієї функції тупиковими ДНФ є:

$$y_{2, \text{тднф1}} = \bar{x}_3 \bar{x}_1 \vee x_3 x_1 \vee x_3 \bar{x}_2,$$

$$y_{2, \text{тднф2}} = \bar{x}_3 \bar{x}_1 \vee x_3 x_1 \vee \bar{x}_2 \bar{x}_1.$$

Будь-яку з них можна обрати як мінімальну форму функції y_2 , наприклад, першу. Тоді

$$y_{2, \text{мднф}} = \bar{x}_3 \bar{x}_1 \vee x_3 x_1 \vee x_3 \bar{x}_2.$$

Ядром функції y_3 є прості імпліканти $x_3 x_1$ та $\bar{x}_3 \bar{x}_1$, а тупиковими ДНФ є

$$y_{3, \text{тднф1}} = \bar{x}_3 \bar{x}_1 \vee x_3 x_1 \vee x_2 \bar{x}_1,$$

$$y_{3, \text{тднф2}} = \bar{x}_3 \bar{x}_1 \vee x_3 x_1 \vee x_3 x_2.$$

Будь-яку з них можна обрати як мінімальну форму функції y_3 , оскільки вони мають однакову ціну, наприклад, першу. Тоді

$$y_{3, \text{мднф}} = \bar{x}_3 \bar{x}_1 \vee x_3 x_1 \vee x_2 \bar{x}_1.$$

Таким чином, мінімальною формою системи перемикальних функцій y_1, y_2, y_3 є

$$\begin{cases} y_1 = x_1 \vee x_3 \bar{x}_2, \\ y_2 = \bar{x}_3 \bar{x}_1 \vee x_3 x_1 \vee x_3 \bar{x}_2, \\ y_3 = \bar{x}_3 \bar{x}_1 \vee x_3 x_1 \vee x_2 \bar{x}_1. \end{cases}$$

7. Для побудови оптимальної комбінаційної схеми, що реалізує систему функцій y_1, y_2, y_3 , необхідно виключити повторне використання однакових логічних елементів з однаковими логічними сигналами на входах. Візуальний аналіз отриманої системи функцій показує, що терм $x_3\bar{x}_2$ входить до складу функцій y_1 та y_2 , терм $\bar{x}_3\bar{x}_1$ – до складу y_2 та y_3 , а терм x_3x_1 – до y_2 та y_3 .
8. Логічні елементи, що реалізують ці три терми, мають зустрічатися в комбінаційній схемі лише один раз (рис. 3.36).

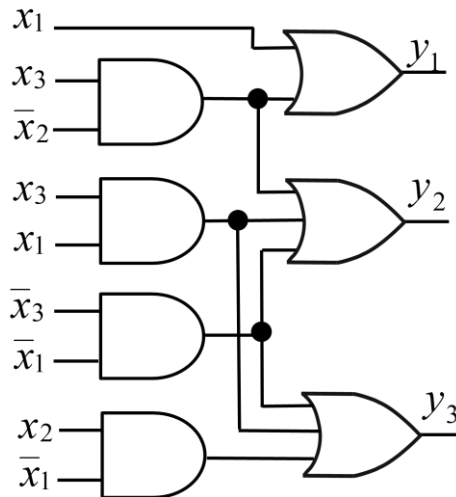


Рис. 3.36. Комбінаційна схема, що реалізує перетворювач кодів, функціонування якого описується таблицею істинності (табл. 3.9)

Таким чином, алгоритм мінімізації системи повністю визначених перемикальних функцій є таким.

1. Подати перемикальні функції в ДДНФ.
2. Утворити множину A всіх конституент одиниці всіх функцій системи та присвоїти кожній з них ознаку належності.
3. Утворити функцію φ як диз'юнкцію всіх елементів множини A .
4. Знайти скорочену ДНФ функції φ одним з відомих аналітичних методів – Квайна, Квайна – Мак-Класкі, Порецького. Операції склеювання та поглинання виконувати з врахуванням ознаки належності кожної з конституент одиниці (термів).
5. Побудувати таблицю покриття заданої системи функцій.
6. Отримати з таблиці покриття мінімальну форму для кожної з перемикальних функцій системи.
7. В отриманій мінімальній формі системи функцій визначити терми, що повторюються.

8. Побудувати комбінаційну схему, що реалізує систему перемикальних функцій та не містить дублювання логічних елементів.

Розглянемо мінімізацію систем неповністю визначених перемикальних функцій.

Під час мінімізації системи неповністю визначених перемикальних функцій кожна з функцій системи довизначають – на наборах, на яких функція не визначена, значення функції прирівнюють до одиниці. Далі застосовують таку саму методику, як і у випадку мінімізації системи повністю визначених функцій, але з одним винятком – конституенти одиниці, що відповідають наборам, на яких функції не визначені, до таблиці покриття не включають.

Задача 3.14. Побудувати оптимізовану комбінаційну схему, що реалізує систему неповністю визначених перемикальних функцій, заданих таблицею істинності (табл. 3.11). Мінімізацію виконати методом Квайна – Мак-Класкі.

Таблиця 3.11

Таблиця істинності системи неповністю визначених перемикальних функцій

x_3	x_2	x_1	y_1	y_2	y_3
0	0	0	0	1	1
0	0	1	–	0	0
0	1	0	1	0	0
0	1	1	×	1	×
1	0	0	0	–	–
1	0	1	1	0	1
1	1	0	1	0	0
1	1	1	1	1	1

Розв’язування.

1. Довизначимо кожен з функцій системи одиничними значеннями на наборах, на яких вони не визначені, та запишемо ДДНФ довизначених функцій у цифровій формі:

$$\begin{cases} y_1 = 001 \vee 010 \vee 011 \vee 101 \vee 110 \vee 111, \\ y_2 = 000 \vee 011 \vee 100 \vee 111, \\ y_3 = 000 \vee 011 \vee 100 \vee 101 \vee 111. \end{cases}$$

Утворимо множину A , елементами якої є конституенти одиниці функцій y_1, y_2, y_3 . Кожній конститuentі одиниці з множини A присвоїмо

ознаку належності даної конститuentі до тієї чи іншої функції системи.

Множина A має вигляд:

$$A = \{000\{2, 3\}; 001\{1\}; 010\{1\}; 011\{1, 2, 3\}; 100\{2, 3\}; 101\{1, 3\}; 110\{1\}; 111\{1, 2, 3\}\}.$$

3. Утворимо функцію φ як диз’юнкцію всіх елементів множини A :
 $\varphi = 000\{2, 3\} \vee 001\{1\} \vee 010\{1\} \vee 011\{1, 2, 3\} \vee 100\{2, 3\} \vee 101\{1, 3\} \vee 110\{1\} \vee 111\{1, 2, 3\}.$

4. Знайдемо скорочену ДНФ функції φ , використовуючи метод Квайна – Мак-Класкі.

На основі ДДНФ функції φ сформуємо комплекс K^0 0-кубів.

Конституенти в K^0 розбиваємо на групи, відділяючи їх пунктирними лініями: група 0-кубів без одиниць, група 0-кубів з однією одиницею, група 0-кубів з двома одиницями та трьома одиницями (рис. 3.37). Кожний 0-куб супроводжуємо ознакою його належності до тієї чи іншої функції.

Наприклад, запис $101\{1, 3\}$ означає, що терм (конституента одиниці) $x_3\bar{x}_2x_1$ входить до складу функцій y_1 та y_3 .

Шляхом склеювання в комплексі K^0 0-кубів з сусідніх груп формуємо комплекс K^1 1-кубів. Склеювання термів виконуємо з урахуванням ознак належності.

Наприклад, склеювання конституент (1) та (4) дає результат $(1-4) \times 00\{2, 3\}$, а конституенти (2) та (5) – $(2-5) 0 \times 1\{1\}$.

Комплекс K^2 2-кубів сформуємо шляхом склеювання в K^1 1-кубів з сусідніх груп.

В отриманих комплексах K^0 , K^1 , K^2 виконаємо всі можливі операції поглинання. При цьому слід взяти до уваги те, що правило поглинання застосовують лише до таких двох термів, у яких ознаки належності (дві множини) повністю співпадають.

Наприклад, імпліканта $\times 1 \times \{1\}$ поглинає імпліканти $(7-8) 11 \times \{1\}$, $(3-7) \times 10\{1\}$, $(3-5) 01 \times \{1\}$, а також конституенти одиниці $(7) 110\{1\}$ та $(3) 010\{1\}$.

Непоглинутими залишились два 2-куби $\times 1 \times \{1\}$, $\times \times 1\{1\}$, а також чотири 1-куби $1 \times 1\{1, 3\}$, $\times 11\{1, 2, 3\}$, $10 \times \{3\}$, $\times 00\{2, 3\}$. Вони й утворюють Z-покриття функції φ :

$$Z = \begin{cases} \times 1 \times \{1\} \\ \times \times 1\{1\} \\ 1 \times 1\{1, 3\} \\ \times 11\{1, 2, 3\} \\ 10 \times \{3\} \\ \times 00\{2, 3\} \end{cases} \quad \begin{array}{l} \text{Куби в Z-покритті відповідають простим} \\ \text{імплікантам функції } \varphi, \text{ а їх диз'юнкція дає скорочену} \\ \text{ДНФ функції } \varphi: \\ \varphi_{\text{сднф}} = \times 1 \times \{1\} \vee \times \times 1\{1\} \vee 1 \times 1\{1, 3\} \vee \times 11\{1, 2, 3\} \vee \\ \vee 10 \times \{3\} \vee \times 00\{2, 3\}, \\ \text{а, отже й множину простих імплікант системи} \\ \text{перемикальних функцій } y_1, y_2, y_3. \end{array}$$

5. Побудуємо таблицю покриття системи неповністю визначених перемикальних функцій y_1, y_2, y_3 , (табл. 3.12). Стівпцями таблиці є конституенти одиниці кожної з функцій y_1, y_2, y_3 , але лише ті, що відповідають наборам, на яких функція визначена (конституенти одиниці, що відповідають наборам, на яких функція не визначена, не включаємо до таблиці). Рядками таблиці є прості імпліканти функції φ , тобто всі прості імпліканти системи перемикальних функцій y_1, y_2, y_3 .

$$\begin{array}{c}
\text{Конституенти} \\
\text{одиниці} \\
\left\{ \begin{array}{l}
(1) \ 0 \cdot 0 \cdot 0 \cdot \{2, 3\} \\
(2) \ \cancel{0 \cdot 0 \cdot 1 \cdot \{1\}} \\
(3) \ 0 \cdot 1 \cdot 0 \cdot \{1\} \\
(4) \ 1 \cdot 0 \cdot 0 \cdot \{2, 3\} \\
(5) \ \cancel{0 \cdot 1 \cdot 1 \cdot \{2, 3\}} \\
(6) \ 1 \cdot 0 \cdot 1 \cdot \{1, 3\} \\
(7) \ 1 \cdot 1 \cdot 0 \cdot \{1\} \\
(8) \ \cancel{1 \cdot 1 \cdot 1 \cdot \{2, 3\}}
\end{array} \right. \\
K^0 =
\end{array}$$

$$\begin{array}{c}
\text{Імпліканти-} \\
\text{терми 2-го рангу} \\
\left\{ \begin{array}{l}
(1-4) \times 0 \cdot 0 \cdot \{2, 3\} \\
(2-5) \ \cancel{0 \cdot 1 \cdot \{1\}} \\
(2-6) \ \cancel{0 \cdot 1 \cdot \{1\}} \\
(3-5) \ 0 \cdot 1 \cdot \{1\} \\
(3-7) \ * \cdot 1 \cdot 0 \cdot \{1\} \\
(4-6) \ 1 \cdot 0 \cdot \{3\} \\
(5-8) \ \times \cdot 1 \cdot 1 \cdot \{1, 2, 3\} \\
(6-8) \ 1 \cdot 1 \cdot \{1, 3\} \\
(7-8) \ 1 \cdot 1 \cdot \{1\}
\end{array} \right. \\
K^1 =
\end{array}$$

$$\begin{array}{c}
\text{Імпліканти-терми} \\
\text{1-го рангу} \\
\left\{ \begin{array}{l}
(2-5, 6-8) \times \times 1 \cdot \{1\} \\
(2-6, 5-8) \\
(3-5, 7-8) \times 1 \cdot \{1\} \\
(3-7, 5-8)
\end{array} \right. \\
K^2 =
\end{array}$$

$$\varphi_{\text{сдно}} = \times 1 \times \{1\} \vee \times \times 1 \{1\} \vee 1 \times 1 \{1, 3\} \vee 1 \times 1 \{1, 2, 3\} \vee 1 \times 1 \{1, 2, 3\} \vee 1 \times 0 \{2, 3\}$$

Рис. 3.37. Склеювання і поглинання конституент одиниці та імплікант функції φ

6. На перетині рядків і стовпців ставимо позначки \checkmark окремо для кожної з функцій y_1, y_2, y_3 . Кожна проста імпліканта бере участь у покритті відповідних конститuent одиниці лише тих функцій, номери яких входять до складу ознаки належності даної імпліканти. При цьому знак \checkmark ставимо в тому стовпці, для якого поточна проста імпліканта є власною частиною конститuentи одиниці стовпця.

Таблиця 3.12

Таблиця покриття системи неповністю визначених перемикальних функцій y_1, y_2, y_3

Прості імпліканти	y_1				y_2			y_3		
	010	101	110	111	000	011	111	000	101	111
$\times 1 \times \{1\}$	\checkmark		\checkmark	\checkmark						
$\times \times 1 \{1\}$		\checkmark		\checkmark						
$1 \times 1 \{1, 3\}$		\checkmark		\checkmark					\checkmark	\checkmark
$\times 11 \{1, 2, 3\}$				\checkmark		\checkmark	\checkmark			\checkmark
$10 \times \{3\}$									\checkmark	
$\times 00 \{2, 3\}$					\checkmark			\checkmark		

Наприклад, для простої імпліканти $\times 00 \{2, 3\}$ позначки \checkmark ставимо в стовпці 000 функції y_2 (імпліканта $\times 00$ є власною частиною конститuentи 000) та в стовпці 000 функції y_3 .

Для кожної з функцій y_1, y_2, y_3 визначимо мінімальну диз'юнктивну форму.

Ядром функції y_1 є проста імпліканта $\times 1 \times$. Для цієї функції тупиковими ДНФ є:

$$y_{1, \text{тднф}1} = \times 1 \times \vee \times \times 1 = x_2 \vee x_1,$$

$$y_{1, \text{тднф}2} = \times 1 \times \vee 1 \times 1 = x_2 \vee x_3 x_1.$$

Оскільки $y_{1, \text{тднф}1}$ має меншу ціну, то

$$y_{1, \text{мднф}} = y_{1, \text{тднф}1} = \times 1 \times \vee \times \times 1 = x_2 \vee x_1.$$

Для функції y_2 :

$$y_{2, \text{мднф}} = \times 00 \vee \times 11 = \bar{x}_2 \bar{x}_1 \vee x_2 x_1.$$

Для функції y_3 :

$$y_{3, \text{мднф}} = \times 00 \vee 1 \times 1 = \bar{x}_2 \bar{x}_1 \vee x_3 x_1.$$

Таким чином, мінімальною формою системи неповністю визначених перемикальних функцій $y_1, y_2, y_3 \in$

$$\begin{cases} y_1 = x_2 \vee x_1, \\ y_2 = \bar{x}_2 \bar{x}_1 \vee x_2 x_1, \\ y_3 = \bar{x}_2 \bar{x}_1 \vee x_3 x_1. \end{cases}$$

7. Побудуємо оптимальну комбінаційну схему, що реалізує систему неповністю визначених перемикальних функцій y_1, y_2, y_3 .

Терм $\bar{x}_2 \bar{x}_1$ присутній в функціях y_2 та y_3 . У комбінаційній схемі логічний елемент, що відповідає цьому терму, зустрічається лише один раз (рис. 3.38). ▣

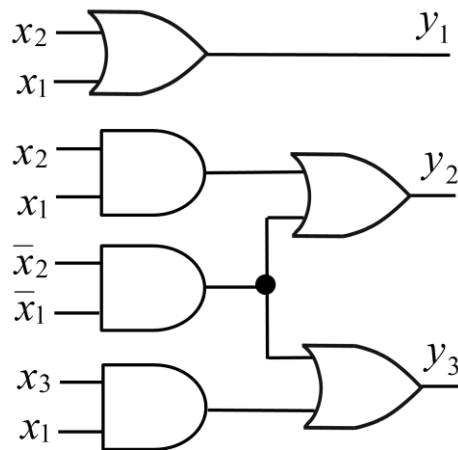


Рис. 3.38. Комбінаційна схема, що реалізує систему неповністю визначених перемикальних функцій

Запитання та завдання

1. Чим викликана необхідність мінімізації систем перемикальних функцій?
2. Якою є мета мінімізації системи перемикальних функцій?
3. Як формують ознаки належності конститuent одиниці до тієї чи іншої функції в системі перемикальних функцій?
4. Як знаходять скорочену ДНФ системи перемикальних функцій?
5. Як заповнюють таблицю покриття системи перемикальних функцій?
6. Як з таблиці покриття отримують мінімальну форму для кожної з перемикальних функцій системи?

7. Як довізначають частково визначені функції під час мінімізації системи неповністю визначених перемикальних функцій?
8. Як будують оптимальну комбінаційну схему, що реалізує задану систему перемикальних функцій?

Задачі для самостійного розв'язування

1. Побудувати оптимізовану за показником апаратної складності схему перетворювача кодів, що реалізує систему 1 перемикальних функцій z_3, z_2, z_1 , заданих таблицею істинності (табл. 3.13). Мінімізацію системи перемикальних функцій виконати методом Квайна – Мак-Класкі.

Таблиця 3.13

Приклади систем перемикальних функцій

x_3	x_2	x_1	Система 1			Система 2		
			z_3	z_2	z_1	γ_3	γ_2	γ_1
0	0	0	1	0	0	×	0	0
0	0	1	0	1	1	0	1	1
0	1	0	1	0	0	–	1	×
0	1	1	0	1	1	1	0	0
1	0	0	1	1	0	1	0	1
1	0	1	1	1	0	0	×	–
1	1	0	1	0	1	1	1	1
1	1	1	0	0	1	1	0	0

2. Побудувати оптимізовану комбінаційну схему з трьома входами та трьома виходами, що реалізує систему 2 неповністю визначених перемикальних функцій $\gamma_3, \gamma_2, \gamma_1$, заданих таблицею істинності (табл. 3.13). Мінімізацію системи перемикальних функцій виконати методом Квайна.

4.

АНАЛІЗ ТА СИНТЕЗ КОМБІНАЦІЙНИХ СХЕМ

У теорії перемикальних функцій розрізняють задачу аналізу та задачу синтезу комбінаційних схем.

Задача аналізу комбінаційної схеми за її відомою структурою (топологією) полягає у знаходженні системи перемикальних функцій, яка описує поведінку (логіку роботи) цієї схеми.

Задача синтезу комбінаційної схеми полягає в побудові із заданого набору типів логічних елементів оптимальної комбінаційної схеми, яка реалізує задану систему перемикальних функцій.

Задача синтезу комбінаційної схеми є оберненою до задачі аналізу комбінаційної схеми.

4.1. Аналіз комбінаційних схем

Будь-якій комбінаційній схемі, що має n входів та m виходів, відповідає система з m перемикальних функцій:

$$y_j = f_j(x_n, \dots, x_1), \quad j = 1, \dots, m.$$

Кількість функцій y_j дорівнює кількості виходів комбінаційної схеми.

Вважатимемо, що комбінаційна схема складається з певним чином з'єднаних між собою логічних елементів, а входи та виходи схеми позначені відповідними ідентифікаторами (змінними, назвами функцій). Процес аналізу заданої комбінаційної схеми полягає в наступному.

1. Позначити виходи всіх логічних елементів (крім тих, виходи яких є виходами схеми), наприклад, символами латинської абетки.
2. Для кожного позначеного логічного елемента записати його оператор.
3. Відповідно до топології комбінаційної схеми для кожного виходу схеми визначити всі можливі напрями (шляхи) поширення сигналів від входів схеми до цього виходу. Рухаючись визначеними напрямами від входів схеми до даного виходу записати формулу у вигляді поетапної суперпозиції операторів логічних елементів, що зустрічаються на цих напрямках.
4. Отриману аналітичну форму перетворити відповідно до правил використовуваної алгебри.

Задача 4.1. Виконати аналіз комбінаційної схеми на рис. 4.1. Отримані аналітичні вирази подати в ДДНФ.

Розв'язування. Комбінаційна схема реалізує дві перемикальні функції y_2, y_1 , що залежать від трьох змінних x_3, x_2, x_1 . Входи та виходи схеми позначені відповідними ідентифікаторами.

1. Позначимо виходи логічних елементів 1 – 3 латинськими літерами А, В, С (див. рис. 4.1).
2. Для позначених логічних елементів запишемо їх оператори: $A = x_3 x_2$, $B = \bar{x}_3 x_1$, $C = \bar{x}_3 \vee \bar{x}_2$.

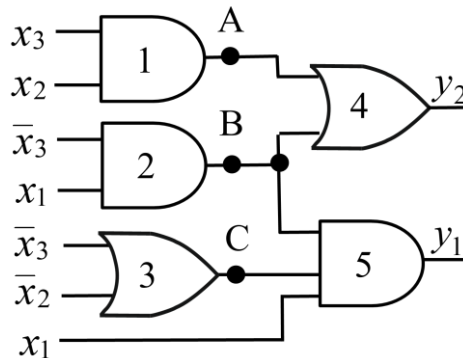


Рис. 4.1. Приклад комбінаційної схеми з двома виходами

3. Запишемо аналітичні вирази для виходів y_2, y_1 , використовуючи оператори логічних елементів:

$$y_2 = A \vee B = x_3 x_2 \vee \bar{x}_3 x_1,$$

$$y_1 = B \cdot C \cdot x_1 = \bar{x}_3 x_1 \cdot (\bar{x}_3 \vee \bar{x}_2) \cdot x_1 = \bar{x}_3 x_1 \cdot (\bar{x}_3 \vee \bar{x}_2) = \bar{x}_3 x_1 \vee \bar{x}_3 \bar{x}_2 x_1.$$

4. Подамо отримані аналітичні вирази в ДДНФ:

$$\begin{aligned} y_2 &= x_3 x_2 \vee \bar{x}_3 x_1 = x_3 x_2 (x_1 \vee \bar{x}_1) \vee \bar{x}_3 (x_2 \vee \bar{x}_2) x_1 = \\ &= x_3 x_2 x_1 \vee x_3 x_2 \bar{x}_1 \vee \bar{x}_3 x_2 x_1 \vee \bar{x}_3 \bar{x}_2 x_1 = y_{2\text{ДНФ}}, \end{aligned} \quad (4.1)$$

$$\begin{aligned} y_1 &= \bar{x}_3 x_1 \vee \bar{x}_3 \bar{x}_2 x_1 = \bar{x}_3 (x_2 \vee \bar{x}_2) x_1 \vee \bar{x}_3 \bar{x}_2 x_1 = \bar{x}_3 x_2 x_1 \vee \\ &\vee \bar{x}_3 \bar{x}_2 x_1 \vee \bar{x}_3 \bar{x}_2 x_1 = \bar{x}_3 x_2 x_1 \vee \bar{x}_3 \bar{x}_2 x_1 = y_{1\text{ДНФ}}. \quad \square \end{aligned} \quad (4.2)$$

Процедуру аналізу комбінаційної схеми застосовують коли для заданої схеми необхідно побудувати еквівалентну їй схему в іншому елементному базисі.

Задача 4.2. Для комбінаційної схеми на рис. 4.1 побудувати еквівалентну комбінаційну схему на основі 3-входових елементів І та 2-входових суматорів за модулем два.

Розв'язування. Оскільки задані логічні елементи (І, суматори за модулем два) є елементним базисом алгебри Жегаліна, а в схемі на рис. 4.1 використані логічні елементи І, АБО, які належать до елементного базису алгебри Буля, то задача зводиться до переходу від співвідношень в алгебрі Буля до відповідних співвідношень в алгебрі Жегаліна.

Канонічними формами в алгебрі Буля є ДДНФ та ДКНФ. Тому функції y_2, y_1 (див. рис. 4.1) необхідно подати в одній з цих форм.

Функції y_2, y_1 доцільно подати в ДДНФ, оскільки ДДНФ легко перетворити в канонічну форму алгебри Жегалкіна – поліном Жегалкіна. ДДНФ функцій y_2, y_1 знайдено в задачі 4.1 – це формули (4.1) та (4.2).

Перетворимо (4.1) та (4.2) в поліном Жегалкіна.

Перетворення $y_{2\text{ДДНФ}}$ в поліном Жегалкіна:

$$y_{2\text{ДДНФ}} = x_3 x_2 x_1 \vee x_3 x_2 \bar{x}_1 \vee \bar{x}_3 x_2 x_1 \vee \bar{x}_3 \bar{x}_2 x_1,$$

$$y_2 = x_3 x_2 x_1 \oplus x_3 x_2 \bar{x}_1 \oplus \bar{x}_3 x_2 x_1 \oplus \bar{x}_3 \bar{x}_2 x_1,$$

$$\begin{aligned} y_2 &= x_3 x_2 x_1 \oplus x_3 x_2 (x_1 \oplus 1) \oplus (\bar{x}_3 \oplus 1) x_2 x_1 \oplus (\bar{x}_3 \oplus 1)(x_2 \oplus 1) x_1 = \\ &= x_3 x_2 \oplus x_3 x_1 \oplus x_1. \end{aligned}$$

Перетворення $y_{1\text{ДДНФ}}$ в поліном Жегалкіна:

$$y_{1\text{ДДНФ}} = \bar{x}_3 x_2 x_1 \vee \bar{x}_3 \bar{x}_2 x_1,$$

$$y_1 = \bar{x}_3 x_2 x_1 \oplus \bar{x}_3 \bar{x}_2 x_1,$$

$$y_1 = (x_3 \oplus 1) x_2 x_1 \oplus (x_3 \oplus 1)(x_2 \oplus 1) x_1 = x_3 x_1 \oplus x_1.$$

Отримані вирази для y_2, y_1 подамо в операторній формі:

$$y_2 = (x_1 \oplus x_3 x_1) \oplus x_3 x_2,$$

$$y_1 = x_1 \oplus x_3 x_1.$$

Цим формам відповідає комбінаційна схема на рис. 4.2, яка є еквівалентною схемі на рис. 4.1. \square

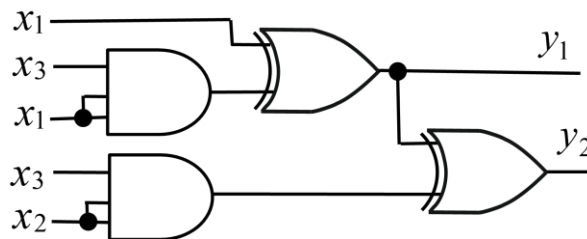


Рис. 4.2. Комбінаційна схема в елементному базисі алгебри Жегалкіна, еквівалентна схемі на рис. 4.1

Запитання та завдання

1. У чому полягає задача аналізу комбінаційної схеми?
2. Охарактеризуйте процес аналізу заданої комбінаційної схеми.
3. Для якої мети застосовують процедуру аналізу комбінаційних схем?
4. Запишіть оператор логічного елемента:
 - а) 3І-НЕ; б) 2 АБО-НЕ; в) 4І.

Задачі для самостійного розв'язування

1. Виконати аналіз комбінаційної схеми з одним виходом на рис. 4.3а. Отриманий аналітичний вираз подати в: а) ДДНФ; б) мінімізованій формі І-НЕ/І-НЕ.
2. Виконати аналіз комбінаційної схеми з двома виходами на рис. 4.3б. Отримані аналітичні вирази подати в: а) ДКНФ; б) мінімізованій формі АБО-НЕ/АБО-НЕ.
3. Дано комбінаційну схему з одним виходом на рис. 4.3в. Побудувати еквівалентну їй комбінаційну схему в елементному базисі 2І-НЕ, 2І з мінімальною апаратною складністю.
4. Дано комбінаційну схему з двома виходами на рис. 4.3г. Побудувати еквівалентну їй комбінаційну схему в елементному базисі 3І-НЕ, оптимізовану за показником апаратної складності.

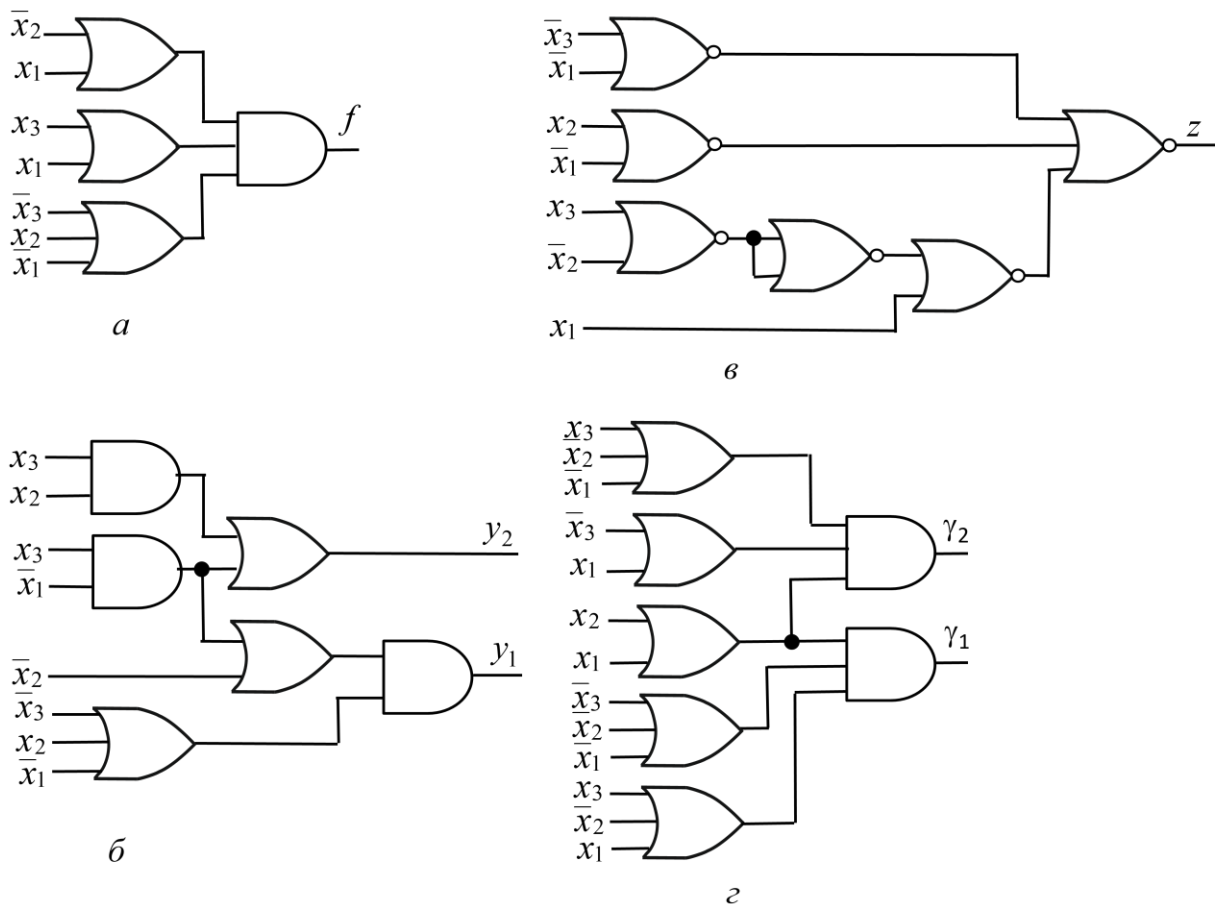


Рис. 4.3. Приклади комбінаційних схем для задач 1 – 4

4.2. Синтез комбінаційних схем у заданому елементному базисі

Цільовою функцією проектування під час синтезу комбінаційної схеми може бути зменшення апаратних витрат на реалізацію схеми та/або досягнення максимально можливої швидкодії схеми.

Розглянемо задачу синтезу комбінаційних схем на основі логічних елементів.

Синтез комбінаційних схем можна здійснювати на основі елементного базису будь-якої з чотирьох алгебр – Буля, Шеффера, Пірса, Жегалкіна, або їх комбінацій.

Розглянемо задачу синтезу комбінаційних схем з n входами і одним виходом в об'єднаному елементному базисі трьох алгебр – Буля-Шеффера-Пірса, тобто в елементному базисі $\{I, АБО, НЕ, I-НЕ, АБО-НЕ\}$.

Процес синтезу комбінаційних схем з n входами і одним виходом розбивають на 3 етапи.

На першому етапі виконують мінімізацію заданої перемикальної функції обраним методом та подання мінімізованої функції у формі, що відповідає заданому елементному базису.

Якщо мінімізацію функції виконують одним з аналітичних методів, то мінімізують пряму (y) функцію або інверсну функцію (\bar{y}) – залежно від заданих типів логічних елементів (табл. 4.1).

Таблиця 4.1

Вибір функції (y або \bar{y}) для мінімізації при використанні аналітичних методів

Використовувані типи логічних елементів	Функція для мінімізації		Використовувані типи логічних елементів
	y (пряма)	\bar{y} (інверсна)	
	Нормальні форми функції		
I, АБО	I/АБО	I/АБО-НЕ	I, АБО-НЕ
I-НЕ	I-НЕ/I-НЕ	I-НЕ/I	I, I-НЕ
АБО, I-НЕ	АБО/I-НЕ	АБО/I	I, АБО
АБО, АБО-НЕ	АБО-НЕ/АБО	АБО-НЕ/АБО-НЕ	АБО-НЕ

Якщо мінімізацію виконують одним з графічних методів, то для отримання форм функції, вказаних у стовпці y у табл. 4.1, необхідно знаходити МДНФ, а для отримання форм, розміщених у стовпці \bar{y} , – МКНФ функції.

На другому етапі функцію подають в операторній формі, у якій враховують кількість входів логічних елементів, заданих при проектуванні.

На третьому етапі будують найоптимальнішу комбінаційну схему відповідно до цільової функції проектування.

У загальному випадку заданий елементний базис може дозволяти отримувати на першому етапі синтезу не одну, а декілька мінімальних нормальних форм функції, і, відповідно, результатом другого етапу синтезу можуть бути дві або більше операторних форм. Завданням третього етапу синтезу є вибір лише однієї з них – форми, яка найповніше відповідає цільовій функції проектування.

Розглянемо приклад синтезу комбінаційної схеми, коли для мінімізації застосовують один з аналітичних методів.

Задача 4.3. Повністю визначена перемикальна функція y від 4-х змінних дорівнює одиниці на наборах 0, 1, 2, 3, 7, 8, 10, 13 та нулю – на решті наборів. Синтезувати оптимізовану за показниками апаратної складності та швидкодії комбінаційну схему, яка реалізує функцію в елементному базисі 2АБО, 3АБО-НЕ.

Часові параметри логічних елементів: $\tau_{2\text{АБО}} = 20$ нс, $\tau_{3\text{АБО-НЕ}} = 22$ нс.

Мінімізацію функції виконати аналітичним методом.

Розв’язування.

На першому етапі синтезу виберемо функцію (y або \bar{y}) для мінімізації. Беручи до уваги типи логічних елементів, заданих для проектування комбінаційної схеми – АБО та АБО-НЕ, і аналізуючи табл. 4.1, бачимо, що окремо взятому типу логічних елементів – АБО-НЕ, відповідає нормальна форма АБО-НЕ/АБО-НЕ функції. Відповідно, для отримання цієї форми необхідно виконувати мінімізацію інверсної функції, тобто \bar{y} (табл. 4.2).

Таблиця 4.2

Вибір функції (y або \bar{y}) для мінімізації при використанні елементного базису АБО-НЕ, АБО

Комбінації типів логічних елементів	Нормальна форма функції	Функція для мінімізації
АБО, АБО-НЕ	АБО-НЕ/АБО	y
АБО-НЕ	АБО-НЕ/АБО-НЕ	\bar{y}

Комбінації типів логічних елементів АБО, АБО-НЕ відповідає нормальна форма АБО-НЕ/АБО, і для її отримання необхідно виконати мінімізацію прямої функції, тобто y .

Обидві нормальні форми – АБО-НЕ/АБО та АБО-НЕ/АБО-НЕ є формами-кандидатами для побудови оптимізованої комбінаційної схеми.

Спочатку виконаємо мінімізацію функції y методом Квайна, а для визначення тупикових ДНФ застосуємо спосіб Петріка.

Знайдемо скорочену ДНФ функції y (рис. 4.4):

$$y_{\text{СДНФ}} = \bar{x}_3 \bar{x}_1 \vee \bar{x}_4 \bar{x}_3 \vee \bar{x}_4 x_2 x_1 \vee x_4 x_3 \bar{x}_2 x_1.$$

Конституенти одиниці	Імпліканти- терми 3-го рангу	Імпліканти-терми 2-го рангу
(1) $\bar{x}_4 \bar{x}_3 \bar{x}_2 \bar{x}_1$	(1-2) $\bar{x}_4 \bar{x}_3 \bar{x}_2$	(1-2, 3-4) $\bar{x}_4 \bar{x}_3$
(2) $\bar{x}_4 \bar{x}_3 \bar{x}_2 x_1$	(1-3) $\bar{x}_4 \bar{x}_3 \bar{x}_1$	(1-3, 2-4) $\bar{x}_4 \bar{x}_3$
(3) $\bar{x}_4 \bar{x}_3 x_2 \bar{x}_1$	(1-6) $\bar{x}_3 \bar{x}_2 \bar{x}_1$	(1-3, 6-7) $\bar{x}_3 \bar{x}_1$
(4) $\bar{x}_4 \bar{x}_3 x_2 x_1$	(2-4) $\bar{x}_4 \bar{x}_3 x_1$	(1-6, 3-7) $\bar{x}_3 \bar{x}_1$
(5) $\bar{x}_4 x_3 \bar{x}_2 \bar{x}_1$	(3-4) $\bar{x}_4 \bar{x}_3 \bar{x}_2$	
(6) $x_4 \bar{x}_3 \bar{x}_2 \bar{x}_1$	(3-7) $\bar{x}_3 x_2 \bar{x}_1$	
(7) $x_4 \bar{x}_3 x_2 \bar{x}_1$	(4-5) $\bar{x}_4 x_2 \bar{x}_1$	
(8) $x_4 x_3 \bar{x}_2 x_1$	(6-7) $x_4 \bar{x}_3 \bar{x}_1$	

$$y_{\text{СДНФ}} = \bar{x}_3 \bar{x}_1 \vee \bar{x}_4 \bar{x}_3 \vee \bar{x}_4 x_2 x_1 \vee x_4 x_3 \bar{x}_2 x_1$$

Рис. 4.4. Визначення скороченої ДНФ функції у методом Квайна

Далі визначимо тупикові ДНФ функції у способом Петріка.
Сформуємо множину KONST конститuent одиниці функції у

$$\text{KONST} = \left\{ \begin{array}{c} \textcircled{1} \quad \textcircled{2} \quad \textcircled{3} \quad \textcircled{4} \quad \textcircled{5} \\ \bar{x}_4 \bar{x}_3 \bar{x}_2 \bar{x}_1, \bar{x}_4 \bar{x}_3 \bar{x}_2 x_1, \bar{x}_4 \bar{x}_3 x_2 \bar{x}_1, \bar{x}_4 \bar{x}_3 x_2 x_1, \bar{x}_4 x_3 x_2 x_1, \\ \textcircled{6} \quad \textcircled{7} \quad \textcircled{8} \\ x_4 \bar{x}_3 \bar{x}_2 \bar{x}_1, x_4 \bar{x}_3 x_2 \bar{x}_1, x_4 x_3 \bar{x}_2 x_1 \end{array} \right\}$$

та множину IMPL простих імплікант, які входять до складу СДНФ функції,

$$\text{IMPL} = \left\{ \begin{array}{c} \textcircled{A} \quad \textcircled{B} \quad \textcircled{C} \quad \textcircled{D} \\ \bar{x}_3 \bar{x}_1, \bar{x}_4 \bar{x}_3, \bar{x}_4 x_2 x_1, x_4 x_3 \bar{x}_2 x_1 \end{array} \right\}.$$

Сформуємо логічні умови покриття конститuent одиниці:

$$\begin{array}{ll} \text{KO1: } \bar{x}_4 \bar{x}_3 \bar{x}_2 \bar{x}_1 \rightarrow A \vee B; & \text{KO2: } \bar{x}_4 \bar{x}_3 \bar{x}_2 x_1 \rightarrow B; \\ \text{KO3: } \bar{x}_4 \bar{x}_3 x_2 \bar{x}_1 \rightarrow A \vee B; & \text{KO4: } \bar{x}_4 \bar{x}_3 x_2 x_1 \rightarrow B \vee C; \\ \text{KO5: } \bar{x}_4 x_3 x_2 x_1 \rightarrow C; & \text{KO6: } x_4 \bar{x}_3 \bar{x}_2 \bar{x}_1 \rightarrow A; \\ \text{KO7: } x_4 \bar{x}_3 x_2 \bar{x}_1 \rightarrow A; & \text{KO8: } x_4 x_3 \bar{x}_2 x_1 \rightarrow D. \end{array}$$

Запишемо логічну умову покриття функції у:

$$(A \vee B) \cdot B \cdot (A \vee B) \cdot (B \vee C) \cdot C \cdot A \cdot A \cdot D.$$

Отриманий вираз зведемо до диз'юнкції кон'юнкцій: ABCD
(результуючий вираз складається лише з одного кон'юнктивного терма).

Кон'юнктивному терму відповідає єдина тупикова ДНФ:

$$y_{\text{ТДНФ}} = A \vee B \vee C \vee D = \bar{x}_3 \bar{x}_1 \vee \bar{x}_4 \bar{x}_3 \vee \bar{x}_4 x_2 x_1 \vee x_4 x_3 \bar{x}_2 x_1.$$

Відповідно, єдиною буде й мінімальна ДНФ:

$$y_{\text{МДНФ}} = \bar{x}_3 \bar{x}_1 \vee \bar{x}_4 \bar{x}_3 \vee \bar{x}_4 x_2 x_1 \vee x_4 x_3 \bar{x}_2 x_1.$$

На другому етапі синтезу подамо отриману МДНФ в нормальній формі АБО-НЕ/АБО:

$$y = \bar{x}_3 \bar{x}_1 \vee \bar{x}_4 \bar{x}_3 \vee \bar{x}_4 x_2 x_1 \vee x_4 x_3 \bar{x}_2 x_1 = \quad (\text{I/АБО})$$

$$= \overline{\overline{x_3 x_1} \cdot \overline{x_4 x_3} \cdot \overline{x_4 x_2 x_1} \cdot \overline{x_4 x_3 x_2 x_1}} =$$

$$= \overline{x_3 x_1 \cdot x_4 x_3 \cdot x_4 x_2 x_1 \cdot x_4 x_3 x_2 x_1} = \quad (\text{I-НЕ/I-НЕ})$$

$$= \overline{(x_3 \vee x_1)(x_4 \vee x_3)(x_4 \vee \bar{x}_2 \vee \bar{x}_1)(\bar{x}_4 \vee \bar{x}_3 \vee x_2 \vee \bar{x}_1)} = \quad (\text{АБО/I-НЕ})$$

$$= \overline{x_3 \vee x_1 \vee x_4 \vee x_3 \vee x_4 \vee \bar{x}_2 \vee \bar{x}_1 \vee \bar{x}_4 \vee \bar{x}_3 \vee x_2 \vee \bar{x}_1} \quad (\text{АБО-НЕ/АБО}).$$

Перетворимо отриману мінімальну нормальну форму АБО-НЕ/АБО функції в операторну форму, беручи до уваги кількість входів логічних елементів – 2АБО, 3АБО-НЕ:

$$y_{\text{ОПН}} = \overline{(x_3 \vee x_1 \vee x_4 \vee x_3) \vee (x_4 \vee \bar{x}_2 \vee \bar{x}_1 \vee (\bar{x}_4 \vee \bar{x}_3) \vee x_2 \vee \bar{x}_1)}.$$

Така операторна форма дозволяє побудувати комбінаційну схему (рис. 4.5), яка характеризується показниками складності $K_1 = 20$ та швидкодії $t_1 = 82$ нс. Для визначення показника t_1 на схемі показано найдовший шлях поширення сигналів від входів схеми до виходу y :

$$t_1 = 3\tau_{2\text{АБО}} + \tau_{3\text{АБО-НЕ}} = 3 \cdot 20 + 22 = 82 \text{ (нс)}.$$

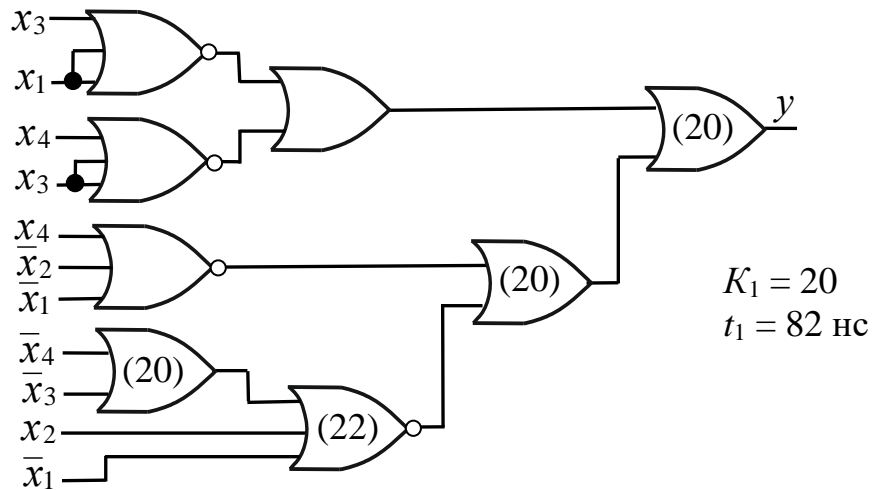


Рис. 4.5. Комбінаційна схема, що реалізує функцію y на основі логічних елементів 2АБО, 3АБО-НЕ (варіант 1)

Тепер необхідно дослідити іншу форму-кандидат –АБО-НЕ/АБО-НЕ.

Етап 1. Виконаємо мінімізацію функції \bar{y} (див. табл. 4.2) методом Квайна – Мак-Класкі. Функція \bar{y} дорівнює одиниці на наборах 4, 5, 6, 9, 11, 12, 14, 15.

Знайдемо скорочену ДНФ функції \bar{y} (рис. 4.6):

$$\bar{y}_{\text{СДНФ}} = \times 1 \times 0 \vee 111 \times \vee 1 \times 11 \vee 10 \times 1 \vee 010 \times.$$

Конституенти одиниці	Імпліканти- терми 2-го рангу	Імпліканти-терми 1-го рангу
$K^0 = \left\{ \begin{array}{l} (1) \text{ } \text{---} 0100 \\ (2) \text{ } \text{---} 0101 \\ (3) \text{ } 0110 \\ (4) \text{ } \text{---} 1001 \\ (5) \text{ } 1100 \\ (6) \text{ } \text{---} 1011 \\ (7) \text{ } \text{---} 1110 \\ (8) \text{ } \text{---} 1111 \end{array} \right\}$	$K^1 = \left\{ \begin{array}{l} (1-2) \text{ } 010 \times \\ (1-3) \text{ } \text{---} 01 \times 0 \\ (1-5) \text{ } \times 100 \text{---} \\ (3-7) \text{ } \times 110 \text{---} \\ (4-6) \text{ } 10 \times 1 \\ (5-7) \text{ } \text{---} 11 \times 0 \\ (6-8) \text{ } 1 \times 11 \\ (7-8) \text{ } \text{---} 111 \times \end{array} \right\}$	$K^2 = \left\{ \begin{array}{l} (1-3, 5-7) \text{ } \times 1 \times 0 \\ (1-5, 3-7) \end{array} \right\}$
$\bar{y}_{\text{СДНФ}} = \times 1 \times 0 \vee 111 \times \vee 1 \times 11 \vee 10 \times 1 \vee 010 \times$		

Рис. 4.6. Визначення скороченої ДНФ функції \bar{y} методом Квайна – Мак-Класкі

Тупикові ДНФ визначимо за допомогою таблиці покриття функції (табл. 4.3).

Таблиця 4.3

Таблиця покриття функції \bar{y}

	Прості імпліканти	Конституенти одиниці							
		0100	0101	0110	1001	1011	1100	1110	1111
1	$\times 1 \times 0$	✓		✓			✓	✓	
2	$111 \times$							✓	✓
3	1×11					✓			✓
4	10×1				✓	✓			
5	$010 \times$	✓	✓						

Ядром функції \bar{y} є прості імпліканти $010\times$ (5-й рядок), 10×1 (4-й рядок) та $\times 1\times 0$ (1-й рядок). Долучивши до ядра просту імплікantu $111\times$ (2-й рядок) отримаємо покриття всіх стовпців табл. 4.3.

Отже, $\bar{y}_{ТДНФ} = 010\times \vee 10\times 1 \vee \times 1\times 0 \vee 111\times$.

Інші можливі тупикові форми матимуть більшу ціну.

Знайдемо мінімальну ДНФ функції \bar{y}

$$\begin{aligned} \bar{y}_{МДНФ} &= \bar{y}_{ТДНФ} = 010\times \vee 10\times 1 \vee \times 1\times 0 \vee 111\times = \\ &= \bar{x}_4 x_3 \bar{x}_2 \vee x_4 \bar{x}_3 x_1 \vee x_3 \bar{x}_1 \vee x_4 x_3 x_2. \end{aligned}$$

Етап 2.

Подано отриману МДНФ в нормальній формі АБО-НЕ/АБО-НЕ:

$$\begin{aligned} y &= \overline{\bar{x}_4 x_3 \bar{x}_2 \vee x_4 \bar{x}_3 x_1 \vee x_3 \bar{x}_1 \vee x_4 x_3 x_2} = && (I/АБО-НЕ) \\ &= \overline{\bar{x}_4 x_3 \bar{x}_2} \cdot \overline{x_4 \bar{x}_3 x_1} \cdot \overline{x_3 \bar{x}_1} \cdot \overline{x_4 x_3 x_2} = && (I-НЕ/I) \\ &= (\overline{x_4 \vee \bar{x}_3 \vee x_2})(\overline{x_4 \vee x_3 \vee \bar{x}_1})(\overline{x_3 \vee x_1})(\overline{x_4 \vee x_3 \vee x_2}) = && (АБО/I) \\ &= (\overline{x_4 \vee \bar{x}_3 \vee x_2})(\overline{x_4 \vee x_3 \vee \bar{x}_1})(\overline{x_3 \vee x_1})(\overline{x_4 \vee x_3 \vee x_2}) = \\ &= \overline{x_4 \vee \bar{x}_3 \vee x_2 \vee x_4 \vee x_3 \vee \bar{x}_1 \vee x_3 \vee x_1 \vee x_4 \vee x_3 \vee x_2} \quad (АБО-НЕ/АБО-НЕ) \end{aligned}$$

Перетворимо отриману мінімальну нормальну форму АБО-НЕ/АБО-НЕ функції в операторну форму, придатну для реалізації на логічних елементах ЗАБО-НЕ, 2АБО (незважаючи на те, що виходимо з нормальної форми АБО-НЕ/АБО-НЕ, слід брати до уваги не лише елементи ЗАБО-НЕ, а й елементи 2АБО, задані в умові задачі):

$$y_{оп2} = \overline{(x_4 \vee \bar{x}_3 \vee x_2 \vee \bar{x}_4 \vee x_3 \vee \bar{x}_1) \vee \bar{x}_3 \vee x_1 \vee \bar{x}_4 \vee \bar{x}_3 \vee \bar{x}_2}.$$

На основі цієї операторної форми побудуємо комбінаційну схему (рис. 4.7).

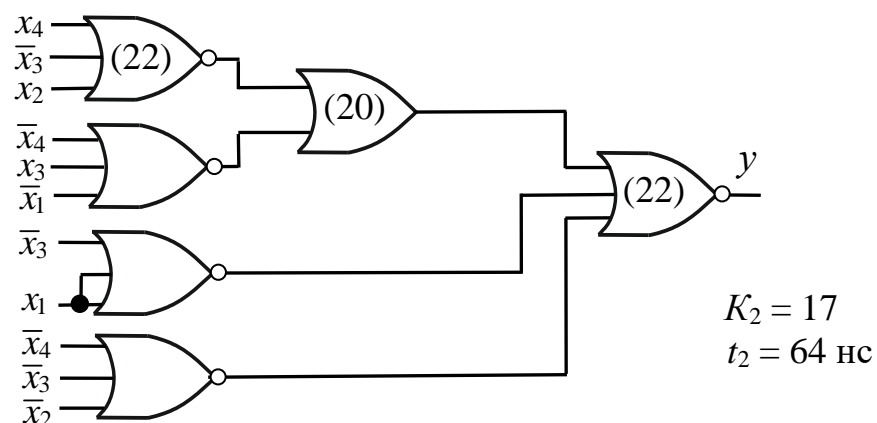


Рис. 4.7. Комбінаційна схема, що реалізує функцію y на основі логічних елементів ЗАБО-НЕ, 2АБО (варіант 2)

Вона характеризується показниками: складності $K_2 = 17$, швидкодії $t_2 = 64$ нс.

На третьому етапі синтезу порівнюємо між собою комбінаційні схеми на рис. 4.5 (показники $K_1 = 20$, $t_1 = 82$ нс) та на рис. 4.7 (показники $K_2 = 17$, $t_2 = 64$ нс).

Як бачимо, схема на рис. 4.7 має кращі показники – як апаратної, так і часової складності. Вона й є розв’язком задачі. ▣

Розглянемо приклад синтезу комбінаційної схеми, коли для мінімізації функції застосовують один з графічних методів.

Задача 4.4. Неповністю визначена перемикальна функція z від 4-х змінних дорівнює одиниці на наборах 1, 2, 4, 9, 12; нулю – на наборах 0, 3, 6, 7, 8, 10, 11; набори 13, 14 є забороненими, а на наборах 5, 15 функція може набувати довільних значень. Синтезувати оптимізовану за показниками апаратної складності та швидкодії комбінаційну схему, яка реалізує функцію z в елементному базисі 2І-НЕ, 3І.

Часові параметри логічних елементів: $\tau_{2І-НЕ} = 22$ нс, $\tau_{3І} = 24$ нс.

Для мінімізації функції застосовувати метод діаграм Вейча.

Розв’язування. На першому етапі синтезу беручи до уваги перелік форм перемикальної функції, які можна отримати з МДНФ та МКНФ (табл. 4.4), а також типи заданих логічних елементів, доходимо висновку, що для заданого елементного базису формами-кандидатами є І-НЕ/І-НЕ та І-НЕ/І. Форму І-НЕ/І-НЕ можна отримати з МДНФ функції, а форму І-НЕ/І – з МКНФ функції.

Отже, для синтезу оптимізованої комбінаційної схеми необхідно знайти як МДНФ, так і МКНФ функції z .

Побудуємо діаграму Вейча функції z (рис. 4.8) та мінімізуємо функцію:

$$z_{\text{МДНФ}} = x_3 \bar{x}_2 \vee \bar{x}_2 x_1 \vee \bar{x}_4 \bar{x}_3 \bar{x}_2 \bar{x}_1,$$

$$z_{\text{МКНФ}} = (\bar{x}_3 \vee \bar{x}_2)(\bar{x}_2 \vee \bar{x}_1)(\bar{x}_4 \vee \bar{x}_2)(x_3 \vee x_2 \vee x_1).$$

Таблиця 4.4
Нормальні форми перемикальної функції, які можна отримати з МДНФ та МКНФ

з МДНФ	з МКНФ
І/АБО	АБО/І
І-НЕ/І-НЕ	АБО-НЕ/ АБО-НЕ
АБО/І-НЕ	І/АБО-НЕ
АБО-НЕ/ АБО	І-НЕ/І

Подамо МДНФ функції z у формі І-НЕ/І-НЕ:

$$\begin{aligned} z_{\text{МДНФ}} &= x_3 \bar{x}_2 \vee \bar{x}_2 x_1 \vee \bar{x}_4 \bar{x}_3 \bar{x}_2 \bar{x}_1 = (І/АБО) \\ &= \overline{\overline{x_3 \bar{x}_2} \vee \overline{\bar{x}_2 x_1} \vee \overline{\bar{x}_4 \bar{x}_3 \bar{x}_2 \bar{x}_1}} = \\ &= \overline{x_3 \bar{x}_2 \cdot x_2 x_1 \cdot x_4 x_3 x_2 x_1} = z_1 \quad (І-НЕ/І-НЕ) \end{aligned}$$

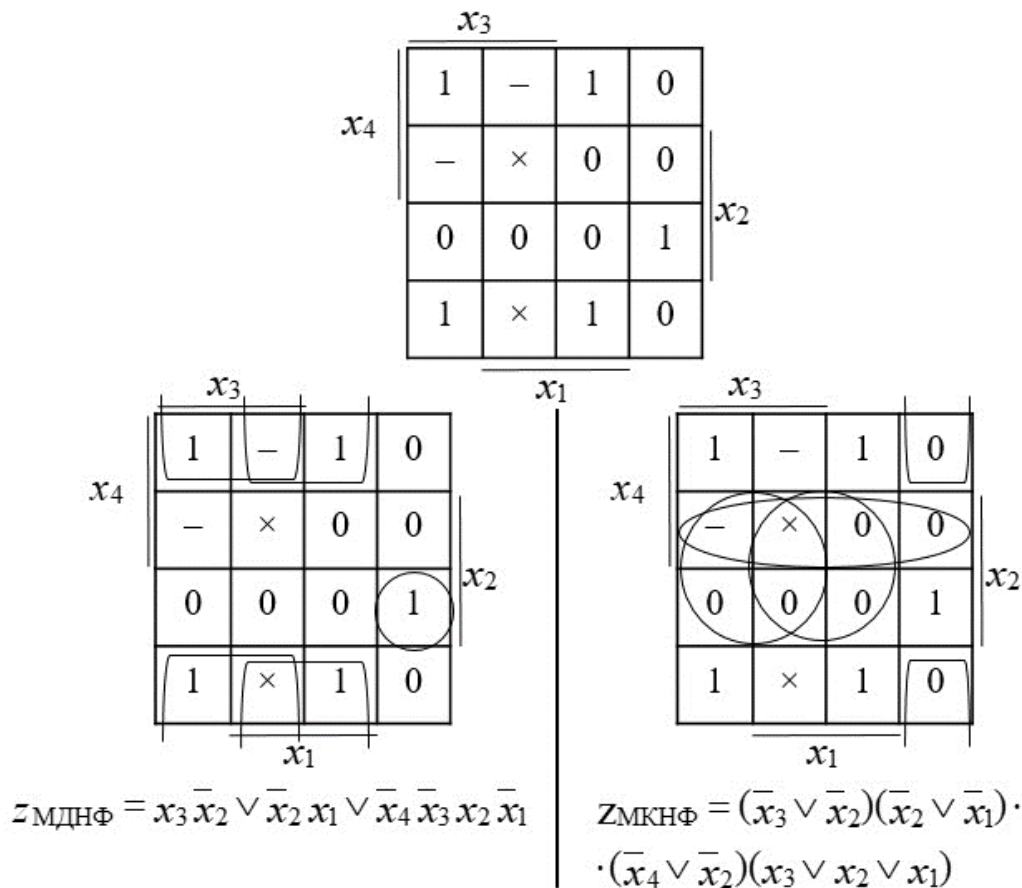


Рис. 4.8. Знаходження МДНФ та МКНФ функції z

Подамо МКНФ функції z у формі І-НЕ/І:

$$\begin{aligned}
 z_{\text{МКНФ}} &= (\bar{x}_3 \vee \bar{x}_2)(\bar{x}_2 \vee \bar{x}_1)(\bar{x}_4 \vee \bar{x}_2)(x_3 \vee x_2 \vee x_1) = && \text{(АБО/І)} \\
 &= \overline{(\overline{\bar{x}_3 \vee \bar{x}_2})(\overline{\bar{x}_2 \vee \bar{x}_1})(\overline{\bar{x}_4 \vee \bar{x}_2})(\overline{x_3 \vee x_2 \vee x_1})} = \\
 &= \overline{x_3 \vee x_2 \vee x_2 \vee x_1 \vee x_4 \vee x_2 \vee x_2 \vee x_1} = \text{(АБО-НЕ/АБО-НЕ)} \\
 &= \overline{x_3 x_2 \vee x_2 x_1 \vee x_4 x_2 \vee \bar{x}_3 \bar{x}_2 \bar{x}_1} = && \text{(І/АБО-НЕ)} \\
 &= \overline{x_3 x_2 \cdot x_2 x_1 \cdot x_4 x_2 \cdot \bar{x}_3 \bar{x}_2 \bar{x}_1} = z_2 && \text{(І-НЕ/І)}
 \end{aligned}$$

На другому етапі синтезу подамо функції z_1 та z_2 в операторних формах, які враховують кількість входів логічних елементів.

При формуванні $z_{1\text{оп}}$, незважаючи на те, що виходимо з нормальної форми І-НЕ/І-НЕ функції z , слід брати до уваги не лише елементи 2І-НЕ, а й також елементи 3І, задані в умові задачі.

Зокрема з метою зменшення апаратної складності терм $\overline{\overline{x_4 x_3 x_2 x_1}}$ доцільно подати у вигляді $\overline{(\overline{\overline{\overline{\overline{x_4 x_3 x_2}}}})x_1}$, а не як $\overline{\overline{\overline{\overline{\overline{\overline{\overline{\overline{x_4 x_3 x_2 x_1}}}}}}}}$. Та й до всього

виразу z_1 слід застосувати такий самий прийом: $z_1 = \overline{abc} = \overline{(\overline{abc})}$, а не $\overline{abc} = \overline{\overline{abc}}$.

При формуванні $z_{2оп}$ терм $\overline{\overline{\overline{x_3 x_2 x_1}}}$ також доцільно записати у вигляді $\overline{(\overline{\overline{x_3 x_2 x_1}})}$, а не як $\overline{\overline{\overline{x_3 x_2 x_1}}}$.

Остаточного отримаємо:

$$z_{1оп} = \overline{(x_3 \overline{x_2} \cdot \overline{x_2} x_1 \cdot (\overline{x_4} \overline{x_3} x_2) \overline{x_1})},$$

$$z_{2оп} = \overline{(x_3 x_2 \cdot x_2 x_1 \cdot x_4 x_2) \cdot (\overline{x_3} \overline{x_2} \overline{x_1})}.$$

На третьому етапі синтезу побудуємо дві комбінаційні схеми, які реалізують функції $z_{1оп}$ та $z_{2оп}$ відповідно (рис. 4.9), та порівняємо їх за показниками апаратної складності (K) та швидкодії (t).

Якщо цільовою функцією проектування є мінімальні апаратні витрати, то результатом синтезу є комбінаційна схема на рис. 4.9а (показник складності $K = 14$). Якщо цільовою функцією проектування є максимальна швидкодія, то перевагу слід віддати комбінаційній схемі на рис. 4.9б. ■

Ще раз зазначимо, що у загальному випадку заданій системі типів логічних елементів можуть відповідати декілька операторних форм функції, а, отже, можна побудувати декілька КС, що реалізують одну й ту саму булеву функцію.

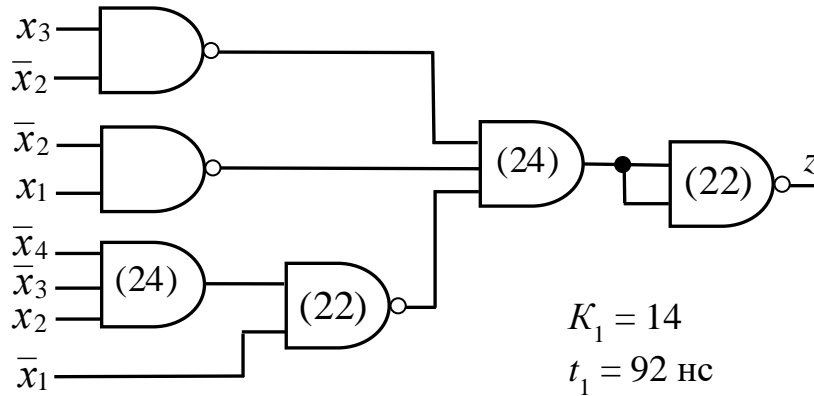
Наприклад, якщо до складу заданої системи елементів входять логічні елементи І, АБО та І-НЕ, то для одержання операторної форми функції можна використати як вихідну одну з п'яти нормальних форм: І/АБО, І-НЕ/І-НЕ, АБО/І-НЕ, І-НЕ/І, АБО/І. Отже, можна дістати п'ять різних операторних форм функції й побудувати п'ять різних КС, які реалізують одну й ту саму перемикальну функцію.

Для вибору оптимальної КС з кількох можливих необхідно порівняти всі КС за заданими параметрами, наприклад, за складністю та/або швидкістю.

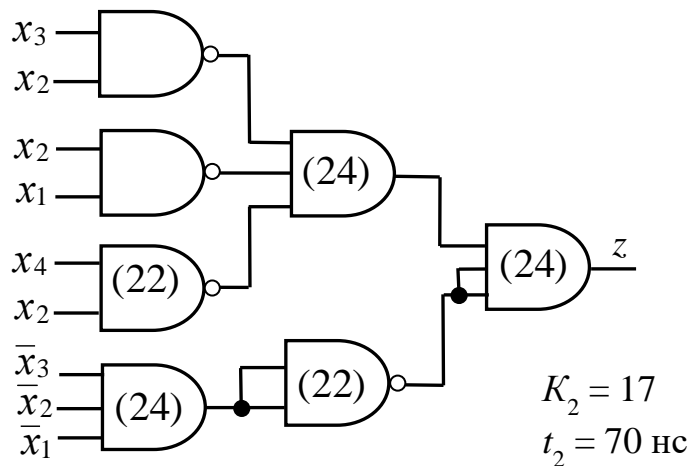
Запитання та завдання

1. У чому полягає задача синтезу комбінаційної схеми в заданому елементному базисі?
2. Якими можуть бути цільові функції проектування під час синтезу комбінаційних схем?
3. Охарактеризуйте етапи синтезу комбінаційної схеми в елементному базисі об'єднаної алгебри Буля-Шеффера-Пірса.

4. Як здійснюють вибір функції (прямої або інверсної) для мінімізації при використанні аналітичних методів мінімізації?
5. Які форми перемикальної функції можна отримати з: а) МДНФ; б) МКНФ?



а



б

Рис. 4.9. Комбінаційні схеми, що реалізують функцію z на основі: а) $z_{1оп}$; б) $z_{2оп}$

Задачі для самостійного розв'язування

1. В елементному базисі 2АБО-НЕ синтезувати комбінаційну схему, що реалізує повністю визначену перемикальну функцію від трьох змінних, яка дорівнює одиниці на наборах 1, 3, 5, 6 та нулю – на решті наборів. Цільова функція проектування – мінімальні апаратні витрати. Мінімізацію функції виконати методом Квайна.

2. Перемикальна функція f від 4-х змінних дорівнює одиниці на наборах 0, 2, 6, 7, 9, 14, 15 та нулю – на решті наборів. Синтезувати оптимізовану за показником апаратної складності комбінаційну схему, що реалізує функцію f в елементному базисі 2АБО, 2І-НЕ. Мінімізацію функції виконати методом Квайна – Мак-Класкі.
3. Синтезувати оптимізовану за показником апаратної складності комбінаційну схему, що реалізує перемикальну функцію від трьох змінних, яка дорівнює нулю на наборах 1, 3, 6 та одиниці – на решті наборів, в елементному базисі 2І, 2АБО. Мінімізацію функції виконати за допомогою діаграми Вейча.
4. Повністю визначена функція y від трьох змінних дорівнює одиниці на наборах 0, 1, 2, 4, 7 та нулю – на решті наборів. Синтезувати оптимізовану за показником апаратної складності та швидкодії комбінаційну схему, яка реалізує функцію в елементному базисі 2І-НЕ, 2АБО. Часові параметри логічних елементів: $\tau_{2І-НЕ} = 22$ нс, $\tau_{2АБО} = 20$ нс. Мінімізацію функції виконати методом Квайна – Мак-Класкі, для визначення тупикових ДНФ застосувати спосіб Петріка.
5. Повністю визначена функція z від трьох змінних дорівнює нулю на наборах 1, 2, 5 та одиниці – на решті наборів. Синтезувати оптимізовану за показником апаратної складності та швидкодії комбінаційну схему, яка реалізує функцію в елементному базисі 2АБО-НЕ, 2І. Часові параметри логічних елементів: $\tau_{2АБО-НЕ} = 20$ нс, $\tau_{2І} = 22$ нс. Мінімізацію функції виконати методом Квайна.
6. Неповністю визначена перемикальна функція γ від 4-х змінних задана діаграмою Вейча.

		x_3			
		0	1	0	0
x_4		1	0	1	×
		1	0	1	–
		0	0	–	×
		x_1			
		x_2			

Синтезувати оптимізовану за показником апаратної складності та швидкодії комбінаційну схему, яка реалізує функцію γ в елементному базисі 3АБО-НЕ, 2І-НЕ.

Часові параметри логічних елементів: $\tau_{3АБО-НЕ} = 26$ нс, $\tau_{2І-НЕ} = 22$ нс.

7. Синтезувати оптимізовану за показником апаратної складності комбінаційну схему, що реалізує перемикальну функцію від трьох змінних, яка дорівнює одиниці на наборах 0, 2, 3, 5, 7 та нулю – на решті наборів, в елементному базисі 3І, 2АБО-НЕ, 3І-НЕ. Для мінімізації функції застосувати карту Карно.

5.

ФУНКЦІОНАЛЬНІ ВУЗЛИ КОМБІНАЦІЙНОГО ТИПУ В ЦИФРОВИХ ОБЧИСЛЮВАЛЬНИХ ЗАСОБАХ

До складу блоків та пристроїв комп'ютерних систем входять різноманітні вузли (схеми), серед яких є такі, що найчастіше зустрічаються в обчислювальних засобах. До таких вузлів, які називають також типовими, належать дешифратори, шифратори, схеми порівняння кодів, мультиплексори, демультиплексори, суматори тощо. Перелічені вузли за своєю будовою є комбінаційними схемами.

5.1. Дешифратори

Дешифратор (decoder, скорочено – DC) – це комбінаційна схема з n входами та 2^n виходами, яка реалізує перетворення n -розрядного позиційного коду $X = \sum_{i=1}^n x_i 2^{i-1}$, $x_i \in \{0,1\}$, в 2^n -розрядний унарний код.

Унарним називають двійковий код (слово), лише один з розрядів якого дорівнює одиниці, а решта – нулю.

Прикладом унарних кодів є конституенти одиниці. Тому кажуть, що дешифратор (DC) – це комбінаційна схема, яка відтворює (формує) конституенти одиниці.

Дешифратор з n входами, що відтворює (реалізує) всі можливі (2^n) конституенти одиниці, і, отже, має 2^n виходів, називають повним. *Повний дешифратор* – це вузол, призначений для вибору в будь-який момент часу одного з 2^n об'єктів, якщо на вхід дешифратора в цей самий момент часу надходить n -розрядний номер X цього об'єкта (нумерація об'єктів від 0 до $2^n - 1$) (рис. 5.1).

Наприклад, повний дешифратор може використовуватись для доступу до слів в пам'яті комп'ютера (рис. 5.2). Якщо в пам'яті зберігають 2^n слів, то для доступу до будь-якого слова необхідна n -розрядна адреса. Так, якщо адреса дорівнює 00...0, то доступ здійснюватиметься до слова з номером 0, якщо адреса дорівнює 0...010, то – до слова з номером 2, адреса 11...1 забезпечує доступ до слова з номером $2^n - 1$.

Крім того, дешифратори входять до складу мікросхем пам'яті. Дешифратори використовують також при побудові перетворювачів кодів та реалізації систем перемикальних функцій.

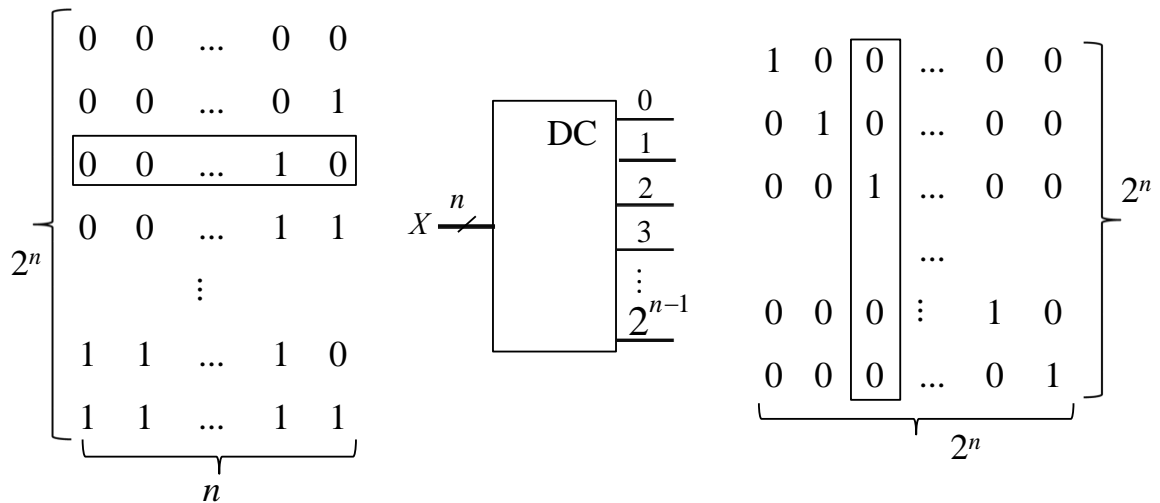


Рис. 5.1. Перетворення n -розрядного позиційного коду (числа) X в 2^n -розрядний унарний код

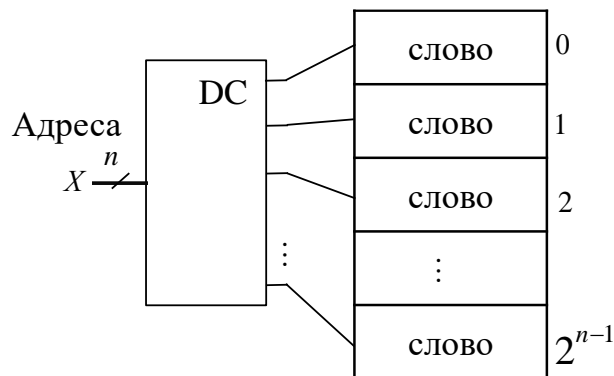


Рис. 5.2. Адресація слів у пам'яті комп'ютера

Повний дешифратор на 3 входи з прямими виходами (рис. 5.3) реалізує 8 конститuent одиниць: $C_0 = \bar{x}_3 \bar{x}_2 \bar{x}_1$, $C_1 = \bar{x}_3 \bar{x}_2 x_1$, ..., $C_7 = x_3 x_2 x_1$. На інформаційні входи дешифратора подають 3-розрядний код $x_3 x_2 x_1$, де x_3 – старший розряд (подається на вхід, що має вагу $4 = 2^2$), x_1 – молодший розряд (подається на вхід з вагою $1 = 2^0$).

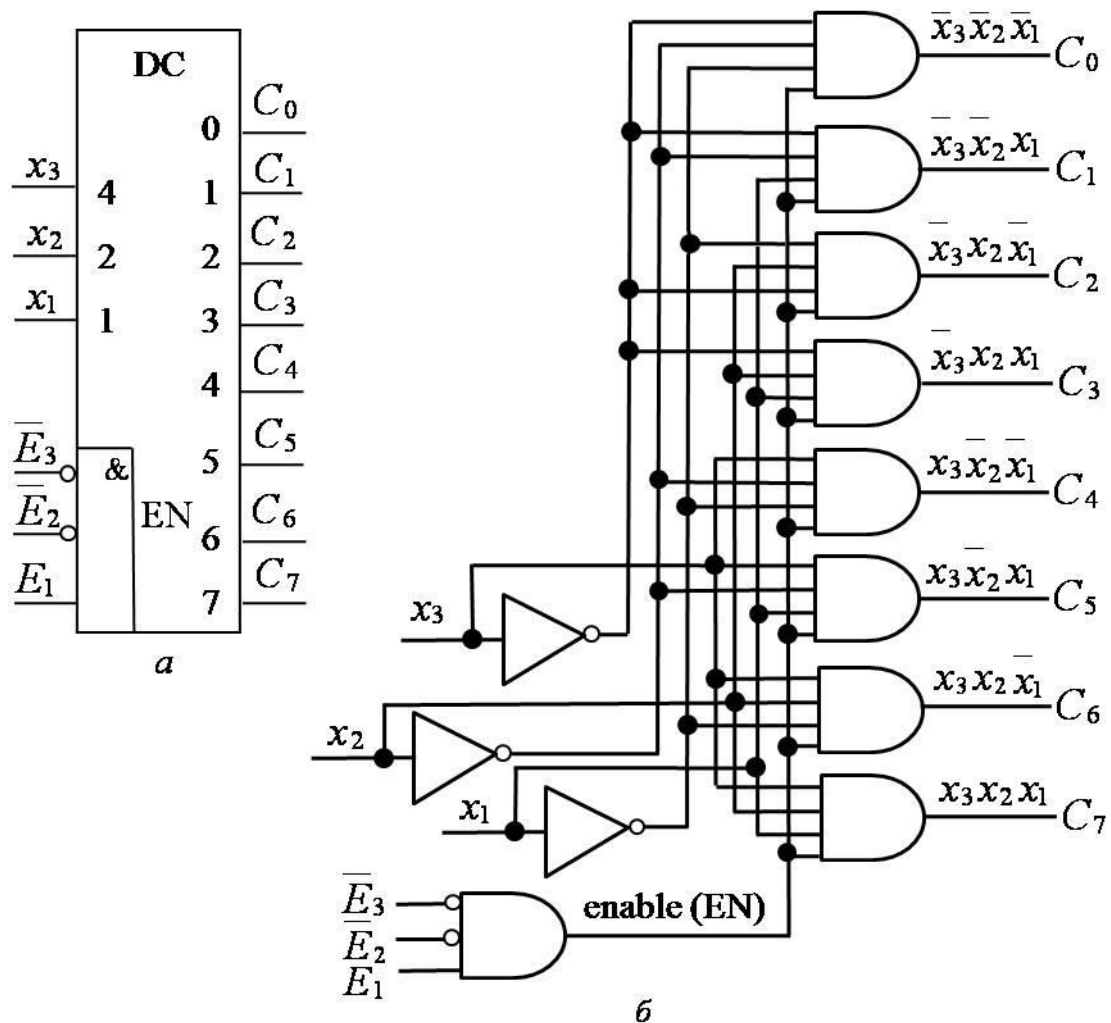
Такий дешифратор позначають DC(3×8), він функціонує за таблицею істинності дешифратора – табл. 5.1.

Дешифратор з прямими виходами функціонує за таблицею істинності, якщо на його входи керування \bar{E}_3 , \bar{E}_2 , E_1 подано код 001 ($\bar{E}_3 = 0$, $\bar{E}_2 = 0$, $E_1 = 1$), і коли $\bar{E}_3 \& \bar{E}_2 \& E_1 = 1$ – це забезпечує елемент 3І на рис. 5.3б; інакше – на всіх виходах $C_0 - C_7$ дешифратора будуть нульові значення. Код 001 є ключем для дозволу (enable – скорочено EN) роботи дешифратора. Сигнали \bar{E}_3 , \bar{E}_2 , E_1 можуть надходити від інших схем пристрою, до складу якого також входить дешифратор.

Таблиця 5.1

Таблиця функціонування повного 3-входового дешифратора з прямими виходами

Входи			Виходи							
x_3	x_2	x_1	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Рис. 5.3. Повний дешифратор 3×8 з прямими виходами: а – умовне графічне позначення; б – функціональна схема

На практиці часто використовують дешифратори з інверсними виходами. Такий дешифратор перетворює позиційний код X в 2^n -розрядний *інверсний унарний код*, тобто код, лише один з розрядів якого дорівнює нулю, а решта – одиниці. Такий дешифратор реалізує конституенти нуля.

У цьому випадку вибір того чи іншого об'єкта (з 2^n наявних об'єктів) здійснюється нульовим значенням відповідного вихідного сигналу дешифратора, а не одиничним значенням – як у випадку дешифратора з прямими виходами.

Наприклад, повний дешифратор на 3 входи з інверсними виходами (рис. 5.4) реалізує 8 конститuent нуля: $Z_0 = x_3 \vee x_2 \vee x_1$, $Z_1 = x_3 \vee x_2 \vee \bar{x}_1$, ..., $Z_7 = \bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_1$ відповідно до таблиці істинності (таблиці функціонування) дешифратора – табл. 5.2.

Легко помітити, що $Z_i = \bar{C}_i$, а $C_i = \bar{Z}_i$, наприклад

$$Z_0 = \bar{C}_0 = \overline{\overline{x_3 x_2 x_1}} = x_3 \vee x_2 \vee x_1.$$

Дешифратор з інверсними виходами функціонує за таблицею істинності, якщо на його входи керування $\bar{E}_3, \bar{E}_2, E_1$ також подано код 001 ($\bar{E}_3 = 0, \bar{E}_2 = 0, E_1 = 1$), – це забезпечує елемент ЗАБО на рис. 5.4б; інакше на всіх виходах дешифратора $Z_0 - Z_7$ будуть одиничні значення. На УГП дешифратора диз'юнкцію сигналів $\bar{E}_3, \bar{E}_2, E_1$ позначають символом ≥ 1 .

Повний дешифратор на 4 входи з прямими виходами, DC(4×16), реалізує 16 конститuent одиниці (рис. 5.5) й функціонує за таблицею істинності – табл. 5.3.

Якщо кількість об'єктів, які необхідно вибирати (ідентифікувати), відмінна від 2^n , то використовують *неповні дешифратори*. У неповному n -розрядному дешифраторі кількість виходів менша за 2^n .

Наприклад, якщо необхідно забезпечувати вибір одного з 10-ти об'єктів у довільні моменти часу, то доцільно використати неповний дешифратор з 10-ма виходами (дешифратор 4×10).

При побудові неповного дешифратора, наприклад, 4×10, використовують таблицю функціонування відповідного повного дешифратора (табл. 5.3) (дешифратора 4×16), у якій відкидають 6 нижніх рядків та 6 правих стовпців. Такий дешифратор реалізує 10 конститuent одиниці (рис. 5.6).

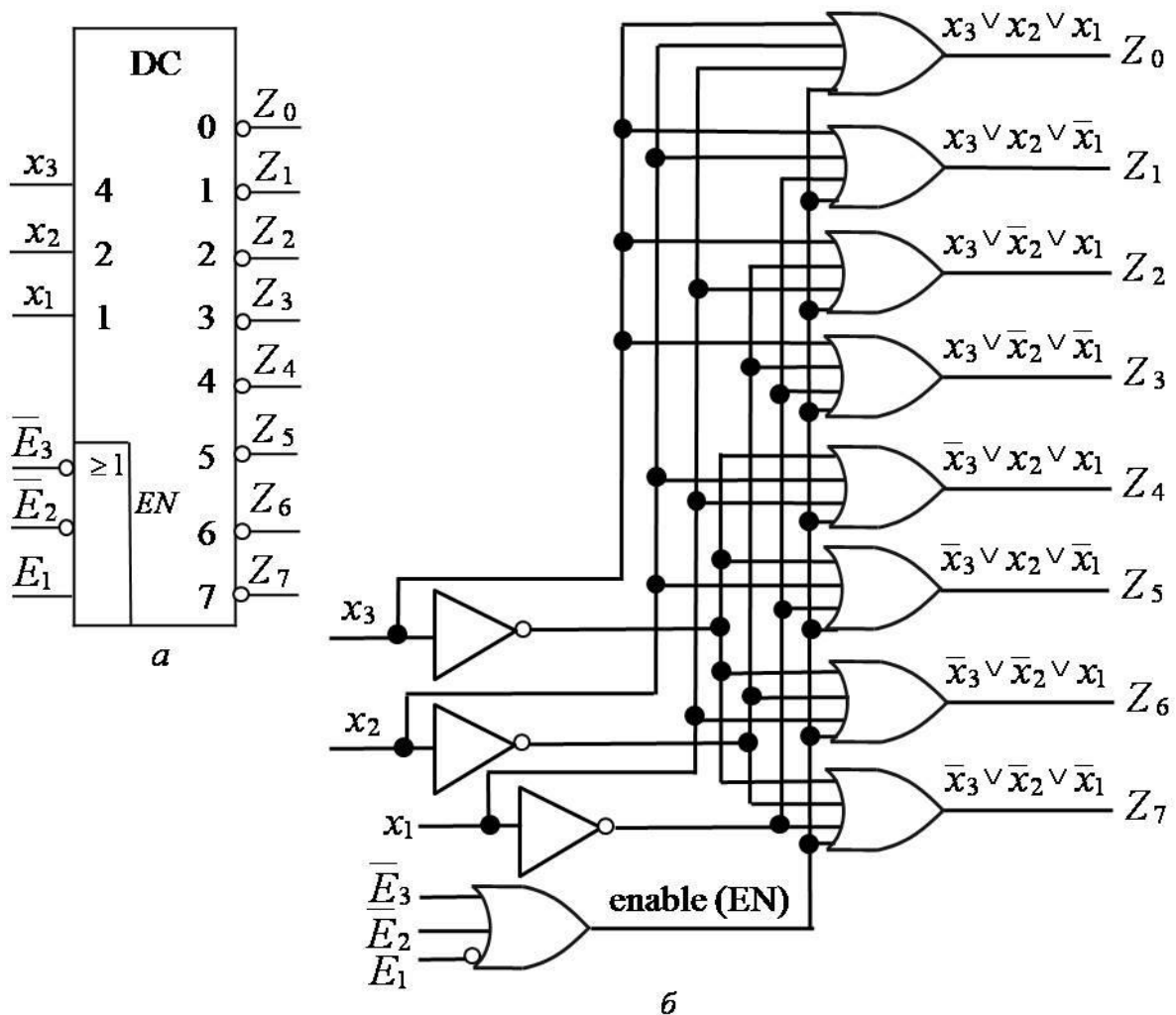
Під час побудови неповних дешифраторів доцільно застосовувати мінімізацію функцій виходів, оскільки на деяких наборах функції виходів невизначені.

Наприклад, функціонування дешифратора 4×10 не визначено на наборах 1010 – 1111.

Таблиця 5.2

Таблиця функціонування повного 3-входового
дешифратора з інверсними виходами

Входи			Виходи							
x_3	x_2	x_1	Z_0	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0

Рис. 5.4. Повний дешифратор 3×8 з інверсними виходами:
а – умовне графічне позначення; б – функціональна схема

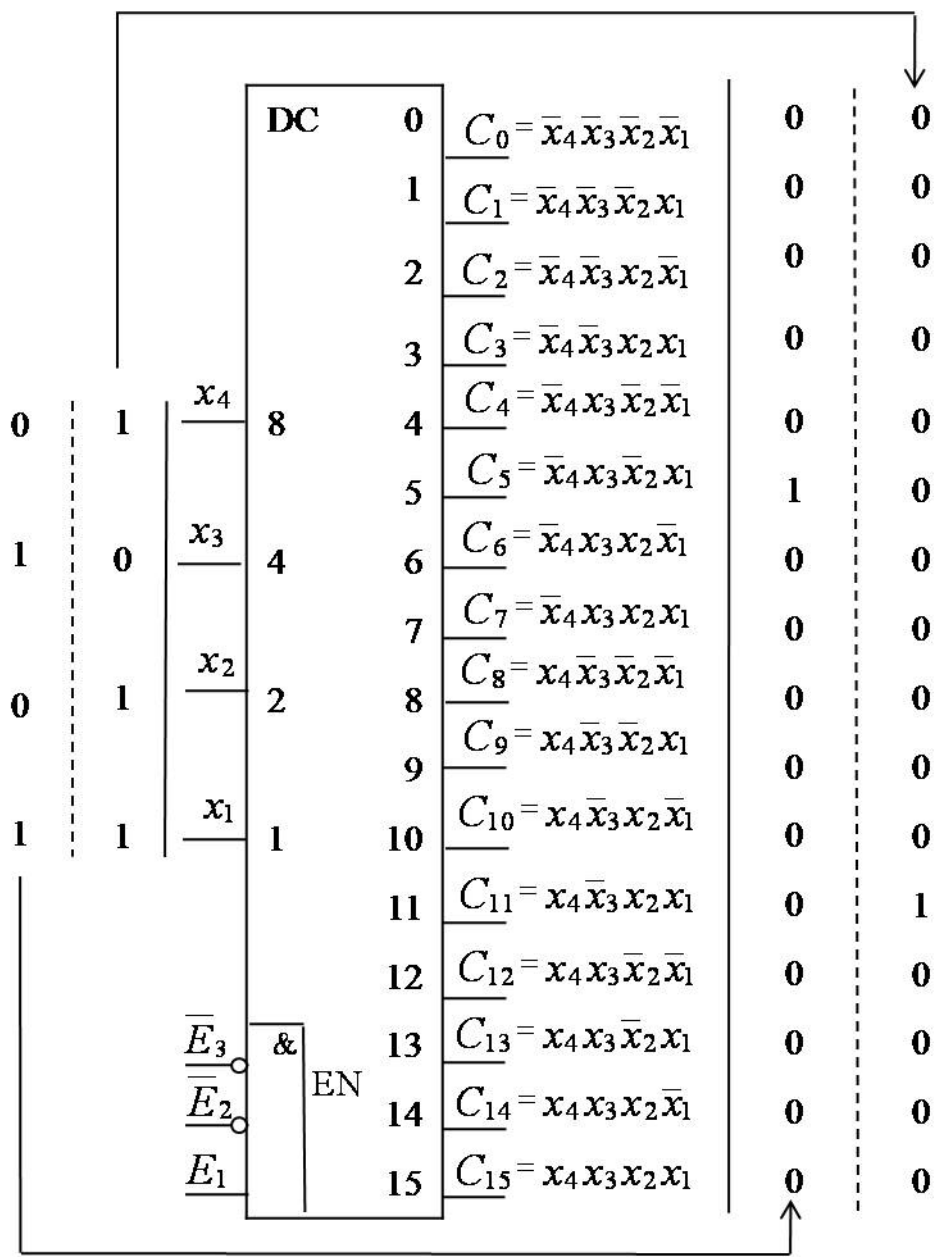


Рис. 5.5. Повний дешифратор 4×16 з прямими виходами

Таблиця 5.3

Таблиця функціонування повного дешифратора 4×16 з прямими виходами

Входи					Виходи																
x_4	x_3	x_2	x_1	x_0	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

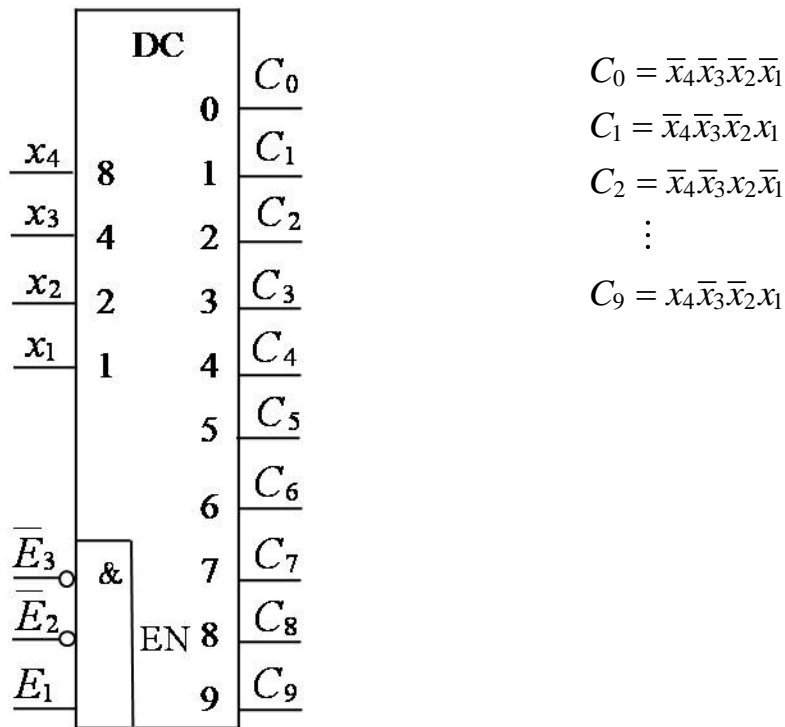


Рис. 5.6. Неповний дешифратор 4×10 з прямими виходами

Задача 5.1. Синтезувати неповний дешифратор 3×7 з інверсними виходами на логічних елементах 2І-НЕ.

Розв’язування.

- Побудуємо таблицю функціонування дешифратора (табл. 5.4) та визначимо дозволені набори (000, 001, ..., 110), а також заборонені набори (набір 111).

Таблиця 5.4

Таблиця функціонування дешифратора 3×7 з інверсними виходами

Входи			Виходи						
x_3	x_2	x_1	y_0	y_1	y_2	y_3	y_4	y_5	y_6
0	0	0	0	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1
0	1	1	1	1	1	0	1	1	1
1	0	0	1	1	1	1	0	1	1
1	0	1	1	1	1	1	1	0	1
1	1	0	1	1	1	1	1	1	0
1	1	1	–	–	–	–	–	–	–

Дозволені набори:

000 – 110,

заборонені набори:

111.

2. Використовуючи діаграми Вейча (рис. 5.7) знайдемо МКНФ функцій виходів y_0, y_1, \dots, y_6 дешифратора та подамо їх у формі І-НЕ, а потім в операторній формі:

$$\begin{aligned}
 y_0 &= \overline{x_3} \vee \overline{x_2} \vee \overline{x_1} = \overline{x_3 x_2 x_1} = \overline{x_3 x_2 x_1}, & y_4 &= \overline{x_3} \vee \overline{x_2} \vee \overline{x_1} = \overline{x_3 x_2 x_1} = \overline{x_3 x_2 x_1}, \\
 y_1 &= \overline{x_3} \vee \overline{x_2} \vee x_1 = \overline{x_3 x_2 x_1} = \overline{x_3 x_2 x_1}, & y_5 &= \overline{x_3} \vee \overline{x_1} = \overline{x_3 x_1}, \\
 y_2 &= \overline{x_3} \vee \overline{x_2} \vee x_1 = \overline{x_3 x_2 x_1} = \overline{x_3 x_2 x_1}, & y_6 &= \overline{x_3} \vee \overline{x_2} = \overline{x_3 x_2}, \\
 y_3 &= \overline{x_2} \vee \overline{x_1} = \overline{x_2 x_1}
 \end{aligned}$$

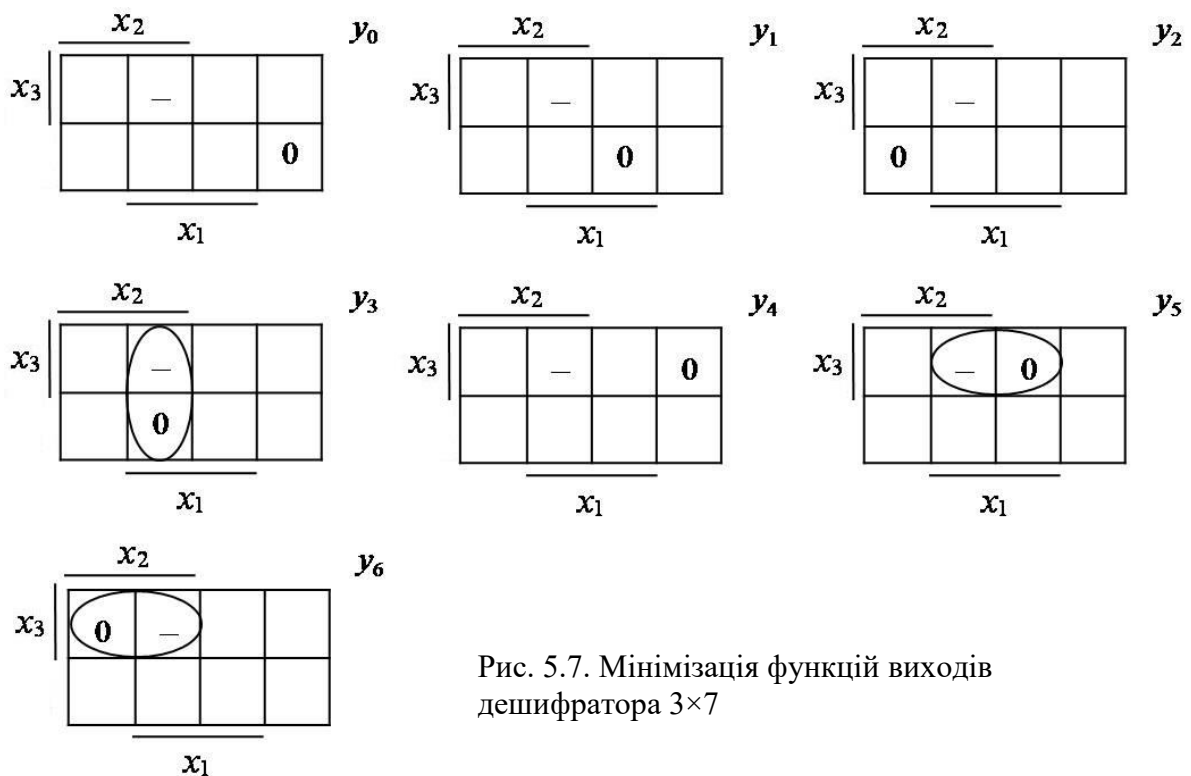


Рис. 5.7. Мінімізація функцій виходів дешифратора 3×7

3. На основі отриманих рівнянь побудуємо комбінаційну схему дешифратора (входи керування дешифратора не показано) та наведемо його умовне графічне позначення (рис. 5.8). ▣

Дешифратор як завершений функціональний вузол можна використовувати для реалізації перемикальних функцій.

Відомо, що будь-яку перемикальну функцію можна подати в ДДНФ, тобто об'єднати операцією АБО ті конституенти одиниці, що відповідають наборам, на яких перемикальна функція дорівнює одиниці. Оскільки повний дешифратор на n входів з прямими виходами дає всі можливі (2^n) конституенти одиниці, то будь-яку перемикальну функцію від n змінних можна реалізувати за допомогою дешифратора та логічного елемента АБО.

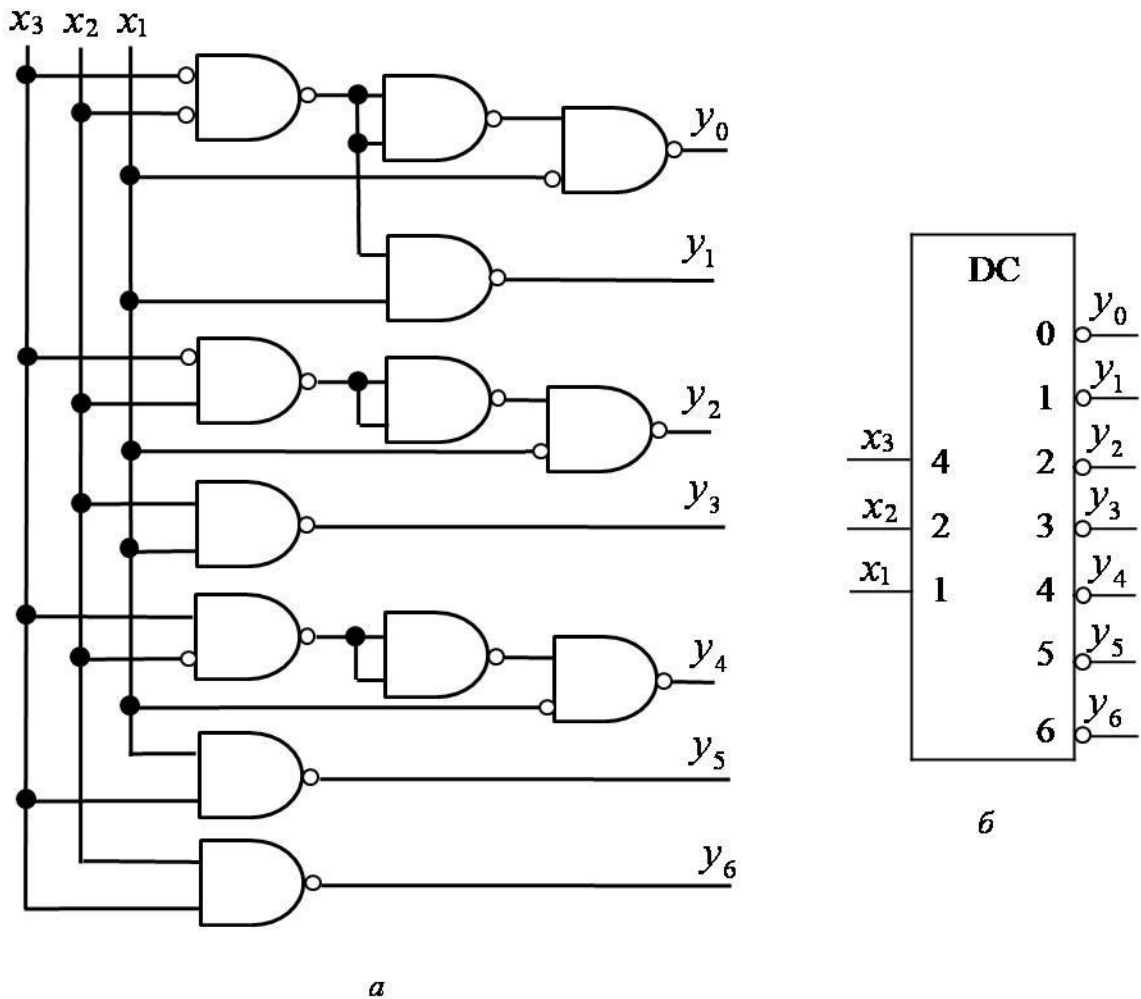


Рис. 5.8. Неповний дешифратор 3×7 з інверсними виходами:
a – функціональна схема; *б* – умовне графічне позначення

Для цього необхідно на входи дешифратора подати всі змінні функції, а входи логічного елемента АБО з'єднати з тими виходами дешифратора (конституентами одиниці), які відповідають наборам змінних, на яких функція дорівнює одиниці.

Нехай перемикальну функцію від 3-х змінних задано таблицею істинності (табл. 5.5). Запишемо функцію y в ДДНФ:

$$y_{\text{дднф}} = 2 \vee 3 \vee 7.$$

Такий запис означає, що для реалізації функції y необхідно виходи 2, 3, 7 дешифратора 3×8 з прямими виходами з'єднати з входами тривходового логічного елемента АБО (рис. 5.9а).

Таблиця 5.5
Таблиця істинності
перемикальної функції від
3-х змінних

Номер набору	x_3	x_2	x_1	y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Але в ДДНФ можна подати не лише пряму функцію (y), а й інверсію функції (\bar{y}), тобто:

$$\bar{y}_{\text{ДДНФ}} = 0 \vee 1 \vee 4 \vee 5 \vee 6.$$

Якщо поставити заперечення над лівою та правою частинами виразу, то отримаємо:

$$y = \overline{0 \vee 1 \vee 4 \vee 5 \vee 6}.$$

Це означає, що для реалізації функції на основі дешифратора з прямими виходами замість логічного елемента АБО можна використати логічний елемент АБО-НЕ, але до його входів слід приєднати ті виходи дешифратора, яким відповідають

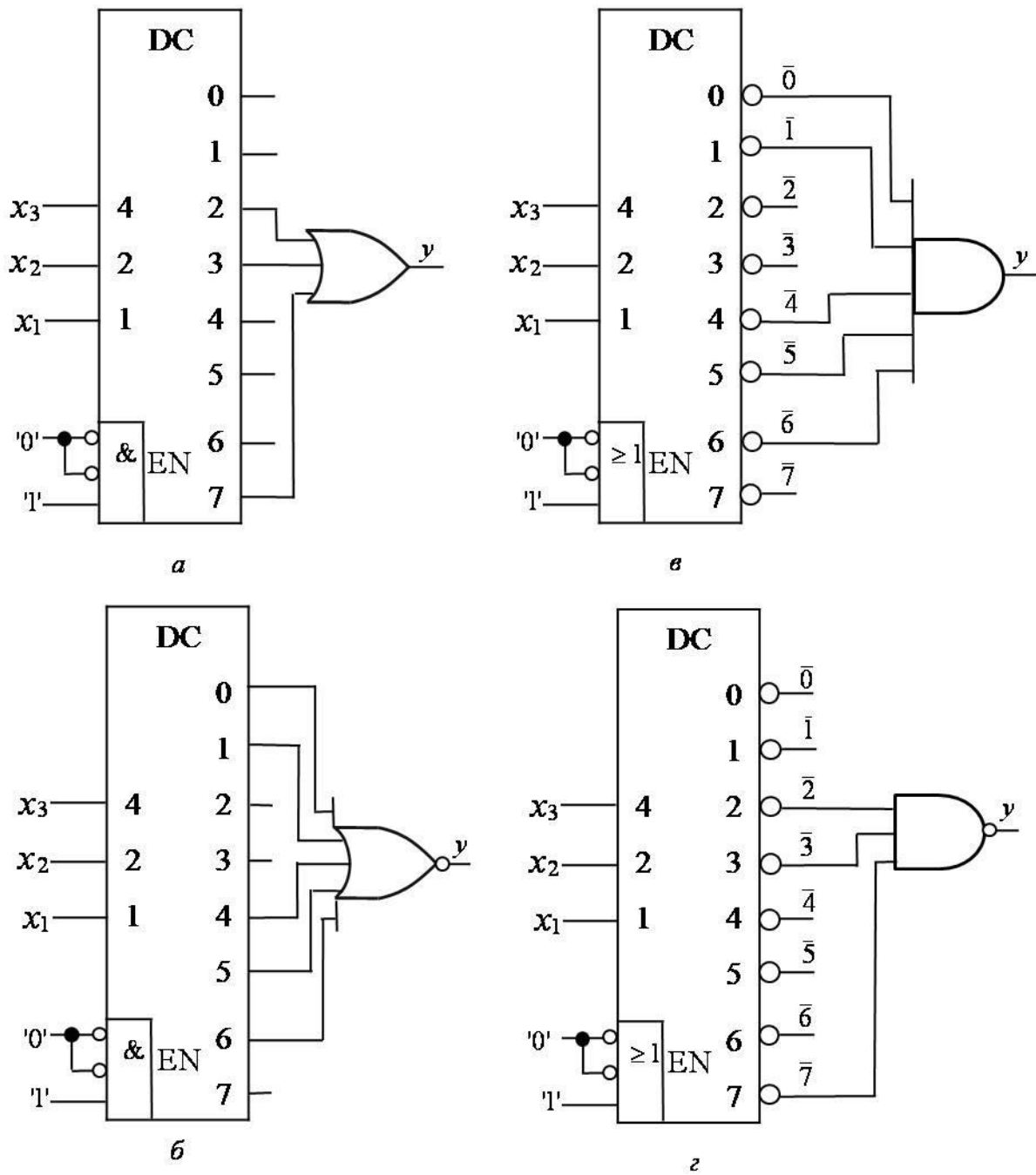
набори, на яких функція дорівнює нулю (рис. 5.9б).

Відомо також, що будь-яку перемикальну функцію можна подати в ДКНФ, тобто об'єднати операцією І ті конституенти нуля, що відповідають наборам, на яких функція дорівнює нулю. Оскільки повний дешифратор на n входів з інверсними виходами дає всі можливі (2^n) конституенти нуля, то будь-яку перемикальну функцію від n змінних можна реалізувати за допомогою дешифратора з інверсними виходами та логічного елемента І. Для цього входи логічного елемента І необхідно з'єднати з тими виходами дешифратора (конституентами нуля), які відповідають наборам змінних, на яких функція дорівнює нулю.

Запишемо функцію y , задану таблицею істинності – табл. 5.5, в ДКНФ: $y_{\text{ДКНФ}} = \bar{0} \cdot \bar{1} \cdot \bar{4} \cdot \bar{5} \cdot \bar{6}$.

Такий запис означає, що для реалізації функції y необхідно виходи 0, 1, 4, 5, 6 дешифратора 3×8 з інверсними виходами з'єднати з входами елемента І (рис. 5.9в).

Але в ДКНФ можна подати й інверсію функції (\bar{y}), тобто $\bar{y}_{\text{ДКНФ}} = \bar{2} \cdot \bar{3} \cdot \bar{7}$.



$$y = 2 \vee 3 \vee 7 \equiv \overline{0 \vee 1 \vee 4 \vee 5 \vee 6} \equiv \overline{0 \cdot 1 \cdot 4 \cdot 5 \cdot 6} \equiv \overline{2 \cdot 3 \cdot 7}$$

DC з прямими
виходами

DC з інверсними
виходами

Рис. 5.9. Реалізація перемикальної функції від 3-х змінних на основі дешифратора 3×8:
 $a, б$ – з прямими виходами; $в, г$ – з інверсними виходами

Якщо поставити заперечення над лівою та правою частинами виразу, то отримаємо: $y = \overline{2 \cdot 3 \cdot 7}$.

Це означає, що для реалізації перемикальної функції на основі дешифратора з інверсними виходами замість логічного елемента І можна використати логічний елемент І-НЕ, але до його входів слід приєднати ті виходи дешифратора з інверсними виходами, яким відповідають набори, на яких функція дорівнює одиниці (рис. 5.9с).

Отже, при реалізації перемикальної функції на основі повного дешифратора застосовують 4 форми подання функції:

- диз'юнкція конститuent одиниці тих наборів, на яких функція дорівнює одиниці,
 $y = 2 \vee 3 \vee 7;$ (АБО)
- заперечення диз'юнкції конститuent одиниці тих наборів, на яких функція дорівнює нулю,
 $y = 0 \vee \overline{1 \vee 4 \vee 5 \vee 6};$ (АБО-НЕ)
- кон'юнкція конститuent нуля тих наборів, на яких функція дорівнює нулю,
 $y = \overline{0 \cdot 1 \cdot 4 \cdot 5 \cdot 6};$ (І)
- заперечення кон'юнкції конститuent нуля тих наборів, на яких функція дорівнює одиниці,
 $y = \overline{2 \cdot 3 \cdot 7}.$ (І-НЕ)

Перші дві форми подання функції реалізують на основі дешифратора з прямими виходами, решту – на основі дешифратора з інверсними виходами.

Дешифратор як завершений функціональний вузол можна також застосовувати для реалізації систем перемикальних функцій.

Задача 5.2. Дано систему перемикальних функцій від трьох змінних, заданих таблицею істинності – табл. 5.6. Реалізувати її на основі дешифратора 3×8 та логічних елементів 2І-НЕ.

Розв'язування.

1. При реалізації перемикальних функцій на основі дешифратора тип логічного елемента – І-НЕ, може бути використаний лише в парі з дешифратором з інверсними виходами.
2. Оскільки має застосовуватись дешифратор з інверсними виходами, то перемикальні функції слід подавати в ДКНФ. Тип логічного елемента І-НЕ визначає, що в ДКНФ слід подавати не пряму функцію (y), а інверсію функції (\bar{y}).

Таблиця 5.6

Таблиця істинності системи перемикальних функцій y_3, y_2, y_1

Номер набору	x_3	x_2	x_1	y_3	y_2	y_1
0	0	0	0	0	1	1
1	0	0	1	1	1	0
2	0	1	0	0	0	1
3	0	1	1	1	0	1
4	1	0	0	1	1	0
5	1	0	1	0	0	1
6	1	1	0	0	0	1
7	1	1	1	1	0	0

Подамо в ДКНФ інверсії заданих функцій:

$$\begin{cases} \overline{y_3} = \overline{1 \cdot 3 \cdot 4 \cdot 7}, \\ \overline{y_2} = \overline{0 \cdot 1 \cdot 4}, \\ \overline{y_1} = \overline{0 \cdot 2 \cdot 3 \cdot 5 \cdot 6}. \end{cases}$$

3. Отриману систему перетворимо в операторну форму, придатну для реалізації в заданому елементному базисі (2І-НЕ):

$$\begin{cases} y_3 = \overline{\overline{1 \cdot 3 \cdot 4 \cdot 7}} = \overline{\overline{\overline{\overline{1 \cdot 3 \cdot 4 \cdot 7}}}}, \\ y_2 = \overline{\overline{0 \cdot 1 \cdot 4}} = \overline{\overline{\overline{\overline{0 \cdot 1 \cdot 4}}}}, \\ y_1 = \overline{\overline{0 \cdot 2 \cdot 3 \cdot 5 \cdot 6}} = \overline{\overline{\overline{\overline{\overline{\overline{0 \cdot 2 \cdot 3 \cdot 5 \cdot 6}}}}}}. \end{cases}$$

4. Побудуємо комбінаційну схему, що реалізує задану систему перемикальних функцій (рис. 5.10). ▣

Особливістю синтезу комбінаційних схем на основі дешифратора є те, що в цьому випадку мінімізацію перемикальних функцій не виконують, оскільки оперують з вже «готовими» конститuentами одиниці (нуля), які беруть з виходів дешифратора.

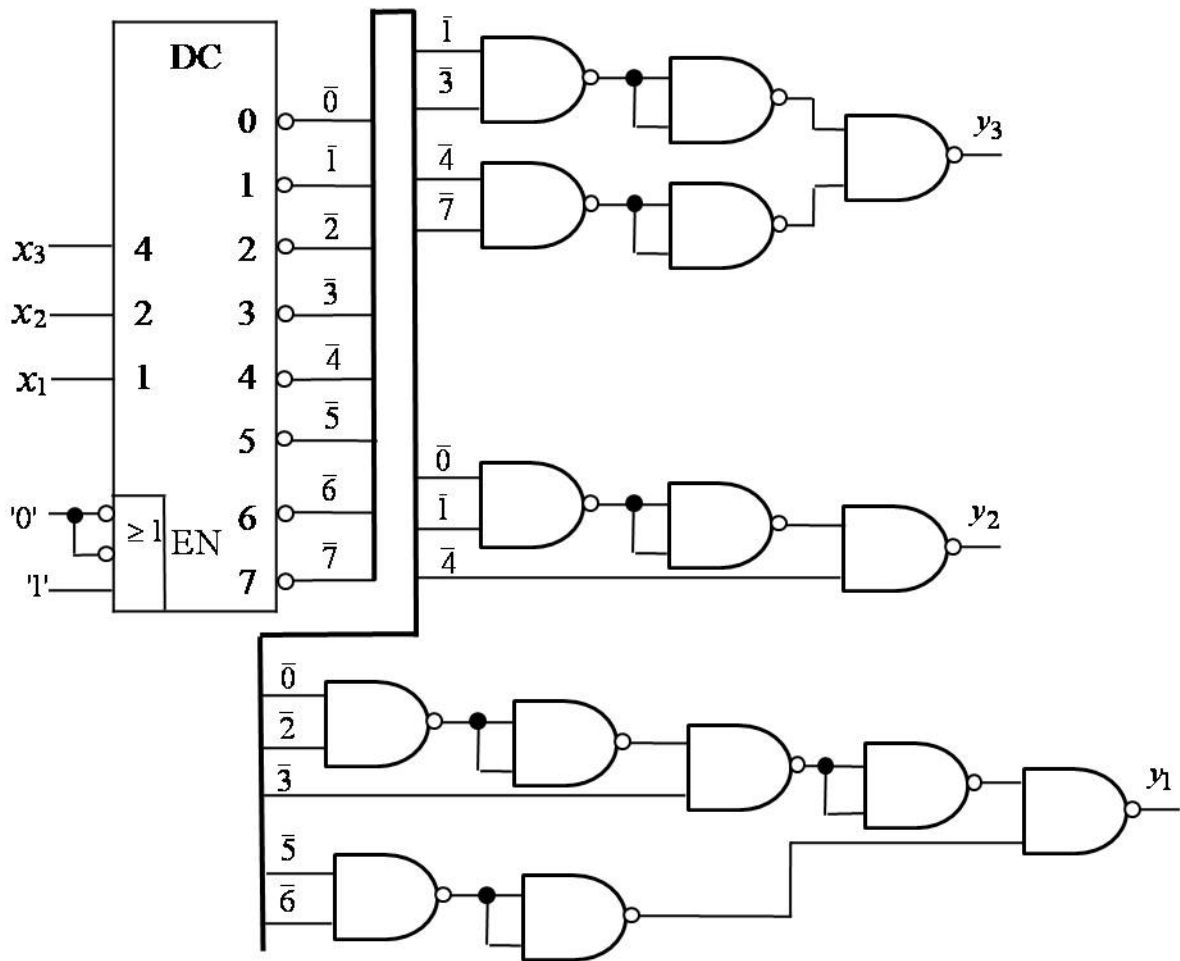


Рис. 5.10. Комбінаційна схема, що реалізує систему перемикальних функцій, заданих таблицею істинності (табл. 5.6)

Зпитання та завдання

1. Дайте визначення дешифратора. Для якої мети в комп'ютері використовують дешифратори?
2. Який код називають унарним? Наведіть приклад унарного коду.
3. Який код називають інверсним унарним кодом? Наведіть приклад інверсного унарного коду.
4. Який дешифратор називають повним (неповним)?
5. Як пов'язані між собою відповідні функції виходів дешифратора з прямими виходами та дешифратора з інверсними виходами?
6. Опишіть методику синтезу неповного дешифратора в заданому елементному базисі.
7. Чому дешифратори можна використовувати для реалізації окремо взятих перемикальних функцій та систем перемикальних функцій?

8. Перелічіть форми, в яких необхідно подавати перемикальні функції при їх реалізації на основі дешифратора.
9. Якою є особливість синтезу комбінаційних схем на основі дешифратора?

Задачі для самостійного розв'язування

1. Синтезувати неповний дешифратор 3×7 з прямими виходами на логічних елементах 2АБО-НЕ.
2. Синтезувати неповний дешифратор 3×6 з з прямими виходами на логічних елементах 2І.
3. Синтезувати неповний дешифратор 3×5 з інверсними виходами на логічних елементах 2І-НЕ.
4. Повністю визначена перемикальна функція від 3-х змінних дорівнює одиниці на наборах 0, 2, 4, 6. Реалізувати функцію на основі дешифратора 3×8 з прямими виходами та логічних елементів 2АБО-НЕ.
5. Повністю визначена перемикальна функція від 3-х змінних дорівнює нулю на наборах 1, 2, 5. Реалізувати функцію на основі дешифратора 3×8 з інверсними виходами та логічних елементів 3І-НЕ.
6. Дано систему повністю визначених перемикальних функцій від 3-х змінних $y_3 = 1, 2, 5$; $y_2 = 1, 3, 4, 6, 7$; $y_1 = 0, 2, 4, 5$, які дорівнюють одиниці на зазначених наборах. Реалізувати систему з використанням дешифратора 3×8 та логічних елементів 2АБО-НЕ.
7. Дано систему повністю визначених перемикальних функцій від 4-х змінних $y_3 = 0, 1, 3, 7, 9, 12, 14, 14, 15$; $y_2 = 1, 3, 7, 8, 10, 13, 15$; $y_1 = 2, 5, 9, 12, 14$, які дорівнюють одиниці на зазначених наборах. Реалізувати систему з використанням дешифратора 4×16 та логічних елементів 3І-НЕ.
8. Побудувати дешифратор 4×16 з прямими виходами на основі дешифратора 3×8 та логічних елементів.
9. Побудувати дешифратор 4×16 з інверсними виходами на основі дешифратора 3×8 та логічних елементів.
10. На основі двох дешифраторів 2×4 та логічних елементів побудувати дешифратор 3×8 з: а) прямими виходами; б) інверсними виходами.

5.2. Шифратори

Шифратор (encoder, coder, CD) – це комбінаційна схема з 2^n входами та n виходами, яка реалізує перетворення 2^n -розрядного унарного коду в n -розрядний позиційний код $X = \sum_{i=1}^n x_i 2^{i-1}$ (рис. 5.11).

Шифратор виконує обернену функцію до функції дешифратора (рис. 5.12).

На входи шифратора в деякий момент часу надходить код, що містить одиницю лише в одному розряді, а в решті розрядах – нулі. Функція шифратора полягає в тому, щоб розпізнати, в якому саме розряді вхідного унарного коду міститься одиниця, і видати на вихід номер цього розряду (нумерація розрядів починається з нуля).

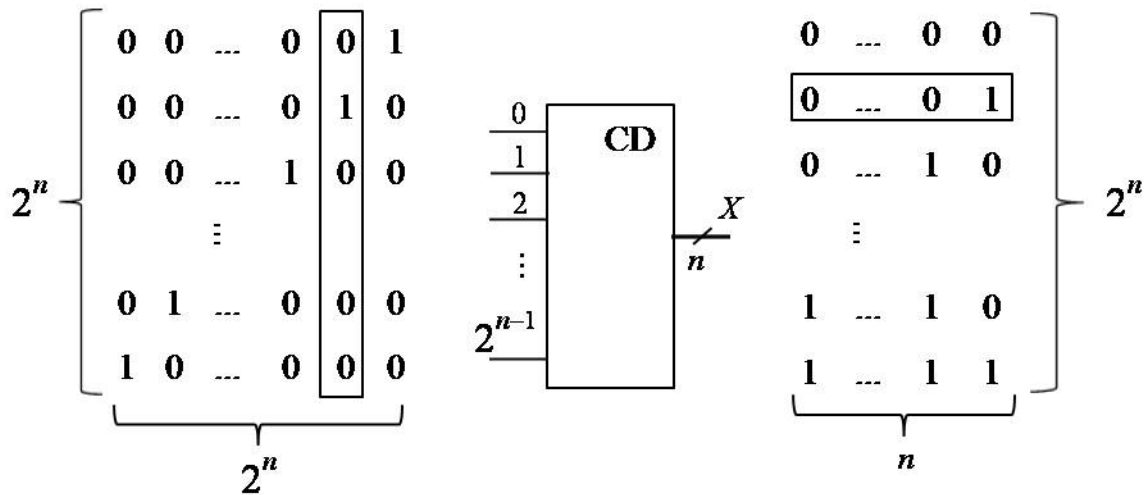


Рис. 5.11. Перетворення 2^n -розрядного унарного коду в n -розрядний позиційний код (число) X

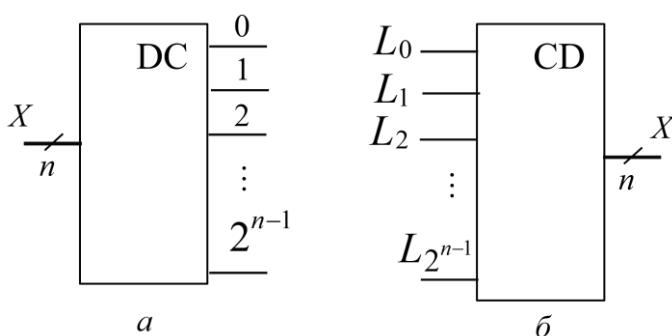


Рис. 5.12. Порівняння функцій дешифратора (а) та шифратора (б)

Наприклад, якщо є 8 об'єктів (пристроїв), пронумерованих від 0 до 7, а вихід кожного з них приєднаний до відповідного входу шифратора, і в будь-який момент часу сигнал може надходити лише від одного об'єкта (пристрою), то шифратор має розпізнати, з якого саме об'єкта (пристрою) надійшов сигнал та видати номер цього об'єкта (пристрою).

Функціонування шифратора, наприклад, на 8 входів (шифратора 8×3) описують таблицею істинності (табл. 5.7).

Таблиця 5.7

Таблиця функціонування шифратора 8×3 з прямими входами

Входи								Виходи		
L_7	L_6	L_5	L_4	L_3	L_2	L_1	L_0	y_3	y_2	y_1
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

З таблиці функціонування (істинності) визначаємо функції виходів шифратора:

$$\begin{cases} y_3 = L_4 \vee L_5 \vee L_6 \vee L_7, \\ y_2 = L_2 \vee L_3 \vee L_6 \vee L_7, \\ y_1 = L_1 \vee L_3 \vee L_5 \vee L_7. \end{cases} \quad (5.1)$$

Цю систему функцій реалізує комбінаційна схема на рис. 5.13а, яка є функціональною схемою шифратора.

Недоліком схеми на рис. 5.13а є те, що на виходах схеми код 000 має місце як у випадку надходження сигналу “1” на вхід L_0 (код 000000001 на входах схеми – перший рядок табл. 5.7), так і у випадку, коли сигнал “1” не надходить на жоден зі входів схеми (код 000000000 на входах). Тому для забезпечення коректності функціонування шифратора на кожен вихід схеми ставлять *керований повторювач* – логічний елемент, що реалізує одномісну *функцію Повторення*, але крім інформаційного входу у має додатково вхід керування (α або $\bar{\alpha}$), який використовується для примусового переведення елемента в так званий *третій стан*, коли рівень напруги на виході у повторювача встановлюється приблизно посередині між рівнем логічного нуля та рівнем логічної одиниці (рис. 5.14). Третій стан (рівень) позначають символом z .

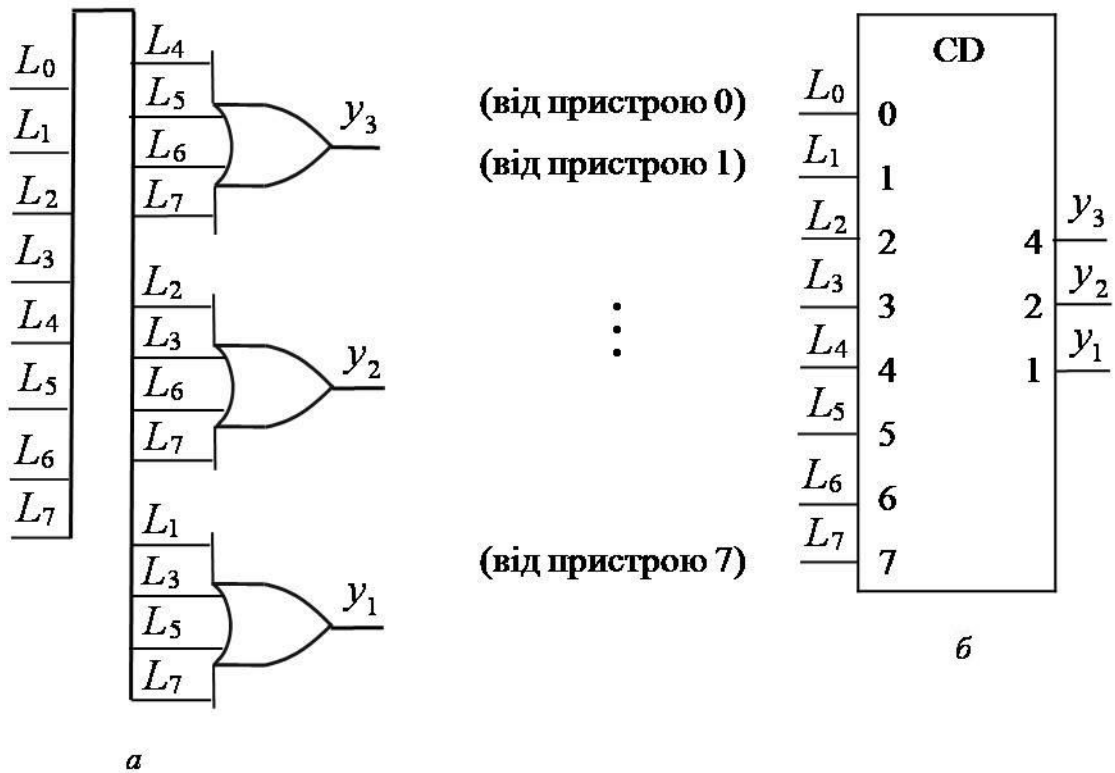


Рис. 5.13. Шифратор 8×3 з прямими входами:
а – функціональна схема; б – умовне графічне позначення

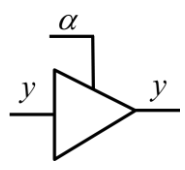
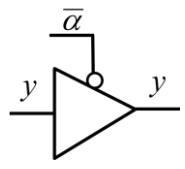
Оскільки логічний елемент Повторювач є транзисторною структурою, то за допомогою додаткового входу керування можна отримувати три рівні (стати) напруги на виході у:

- “0” – рівень (стан) “нуль”,
- “1” – рівень (стан) “одиниця”,
- “z” – третій рівень (стан) “вимкнено”, “від’єднано”.

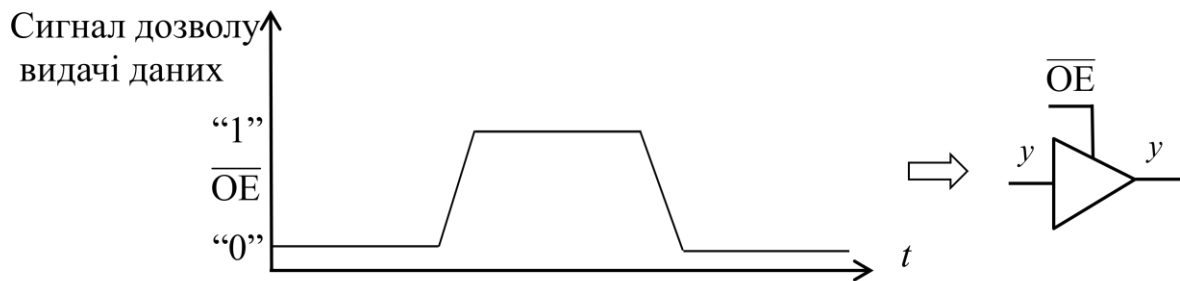
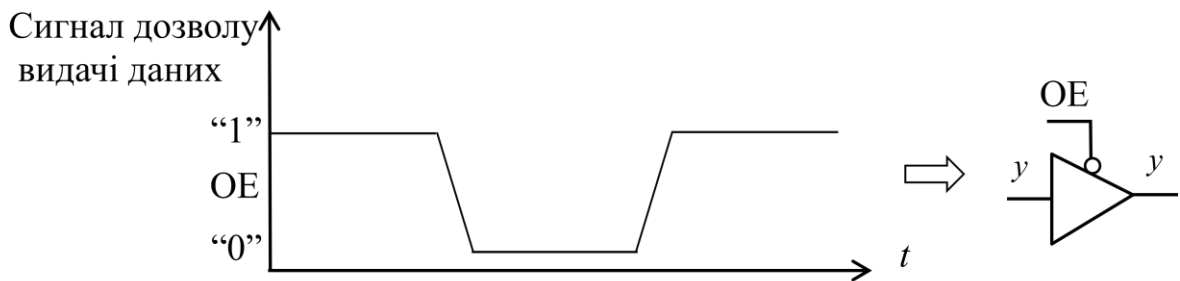
Слід зазначити, що стан “z” (третій рівень напруги) не є інформаційним, а лише схемотехнічним. Інформаційними є 2 рівні – рівень логічного нуля та рівень логічної одиниці.

Якщо напруга на виході керованого повторювача перебуває у третьому стані, то вважають, що даний вихід є від’єднаним від інших пристроїв, схем, з якими він має електричний зв’язок.

На практиці використовують два види керованих повторювачів: а) повторювачі, переведення яких у третій стан здійснюють логічною одиницею – повторювач з прямим входом керування (на вхід керування має надходити “1”); б) повторювачі, переведення яких у третій стан здійснюють логічним нулем – повторювач з інверсним входом керування (на вхід керування має надходити “0”) (див. рис. 5.14а).

Тип керованого повторювача	Входи		Вихід
	α	y	y
 З прямим входом керування	0	0	0
	0	1	1
	1	0	z
	1	1	z
 З інверсним входом керування	$\bar{\alpha}$	y	y
	0	0	z
	0	1	z
	1	0	0
	1	1	1

a



б

Рис. 5.14. Керований повторювач:
a – таблиця істинності керованого повторювача;
б – видача даних з використанням керованого повторювача

Для керування повторювачами у схемі шифратора має додатково формуватись внутрішній сигнал дозволу видачі даних (слова) з виходів шифратора. Такий сигнал називають ОЕ (output enable) – дозвіл виходу (видачі).

Видачу даних з виходів шифратора можуть забезпечувати нульовим значенням цього сигналу – тоді його позначають \overline{OE} , і видача даних буде відбуватись, якщо $\overline{OE} = 0$; або одиничним значенням сигналу – тоді його позначають ОЕ, і видача даних буде відбуватись, якщо ОЕ = 1.

Сигнали \overline{OE} , ОЕ пов'язують з входом керування повторювача наступним чином. Якщо переведення в третій стан керованого повторювача здійснюють логічним нулем, то на вхід керування повторювача подають сигнал ОЕ. Якщо переведення у третій стан керованого повторювача здійснюють логічною одиницею, то на вхід керування повторювача подають сигнал \overline{OE} (рис. 5.14б).

Для побудови повної функціональної схеми шифратора 8×3 (рис. 5.15) необхідно схему на рис. 5.13а доповнити 8-входовим елементом АБО. Якщо на входах шифратора має місце код 00000000 (жоден із зовнішніх об'єктів (пристроїв) не надсилає сигнал на шифратор), то виходи y_3, y_2, y_1 схеми перебуватимуть у третьому стані (від'єднано). Якщо ж на входи $L_0 - L_7$ шифратора надійде будь-який відмінний від нуля 8-розрядний код, то виходи y_3, y_2, y_1 схеми перебуватимуть у бістабільному стані (або логічний "0", або логічна "1") – залежно від конкретного коду на входах $L_0 - L_7$.

Функціональну схему шифратора 8×3 можна побудувати також на основі логічних елементів АБО-НЕ. Для цього замість системи рівнянь (5.1) необхідно записати систему для інверсій функцій y_3, y_2, y_1 , тобто у вигляді (див. табл. 5.7):

$$\begin{cases} \overline{y}_3 = L_0 \vee L_1 \vee L_2 \vee L_3 \\ \overline{y}_2 = L_0 \vee L_1 \vee L_4 \vee L_5 \\ \overline{y}_1 = L_0 \vee L_2 \vee L_4 \vee L_6 \end{cases} \quad (5.2)$$

Систему (5.2) перетворимо так:

$$\begin{cases} y_3 = \overline{L_0 \vee L_1 \vee L_2 \vee L_3}, \\ y_2 = \overline{L_0 \vee L_1 \vee L_4 \vee L_5}, \\ y_1 = \overline{L_0 \vee L_2 \vee L_4 \vee L_6}. \end{cases} \quad (5.3)$$

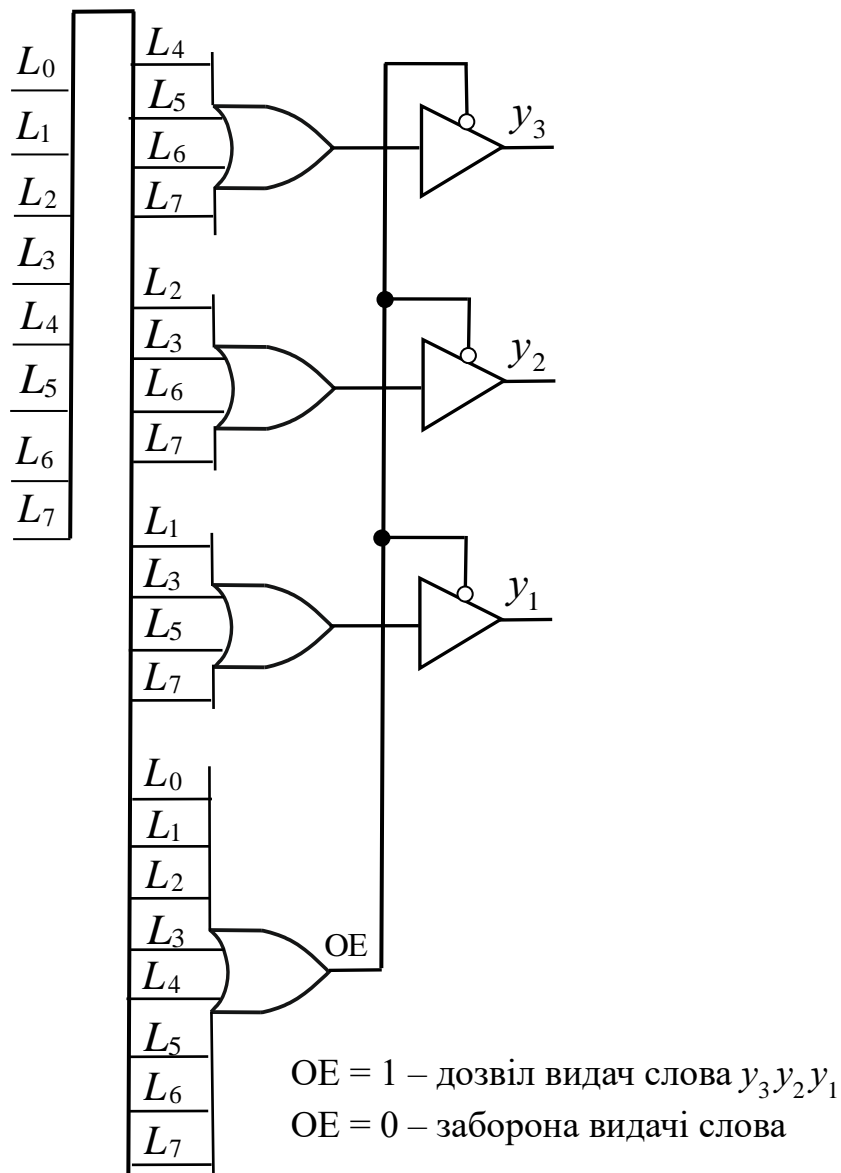


Рис. 5.15. Повна функціональна схема шифратора 8×3 з прямими входами

Тоді, для побудови шифратора на основі елементів АБО-НЕ необхідно (див. рис. 5.15):

- 1) замінити 4-входові елементи АБО на 4-входові елементи АБО-НЕ та з'єднати їх з входами схеми відповідно до системи рівнянь (5.3),
- 2) замінити 8-входовий елемент АБО на 8-входовий елемент АБО-НЕ,
- 3) на виходах схеми розмістити керовані повторювачі, у яких переведення у третій стан здійснюється логічною одиницею.

У загальному випадку кількість входів шифратора може бути відмінною від степеня двійки.

Нехай k – кількість входів шифратора. Тоді шифратор здійснює перетворення k -розрядного унарного коду в n -розрядний позиційний код, де $n = \lceil \log_2 k \rceil$.

Задача 5.3. Синтезувати 6-входовий шифратор з прямими входами на логічних елементах 2АБО-НЕ.

Розв’язування.

1. Визначимо кількість виходів шифратора: $n = \lceil \log_2 6 \rceil = 3$.
2. Побудуємо таблицю функціонування шифратора 6×3 . Вона утворюється з табл. 5.7, у якій необхідно викреслити стовпці L_7, L_6 та два останні рядки.
3. Оскільки тип заданого логічного елемента – АБО-НЕ, то систему перемикальних функцій слід записати для інверсій функцій y_3, y_2, y_1 та подати в формі АБО-НЕ:

$$\left\{ \begin{array}{l} \overline{y_3} = L_0 \vee L_1 \vee L_2 \vee L_3, \\ \overline{y_2} = L_0 \vee L_1 \vee L_4 \vee L_5, \\ \overline{y_1} = L_0 \vee L_2 \vee L_4, \end{array} \right. \quad \left\{ \begin{array}{l} y_3 = \overline{L_0 \vee L_1 \vee L_2 \vee L_3}, \\ y_2 = \overline{L_0 \vee L_1 \vee L_4 \vee L_5}, \\ y_1 = \overline{L_0 \vee L_2 \vee L_4}. \end{array} \right.$$

4. Запишемо отримані функції в операторній формі, придатній для реалізації на елементах 2АБО-НЕ:

$$\left\{ \begin{array}{l} y_3 = \overline{\overline{L_0 \vee L_1 \vee L_2 \vee L_3}}, \\ y_2 = \overline{\overline{L_0 \vee L_1 \vee L_4 \vee L_5}}, \\ y_1 = \overline{\overline{L_0 \vee L_2 \vee L_4}}. \end{array} \right.$$

5. Побудуємо функціональну (комбінаційну) схему шифратора та наведемо його умовне графічне позначення (рис. 5.16). ▣

Знаком \diamond позначають виходи на три стани.

Досить часто об’єкти (пристрої), приєднані до шифратора, надсилають як активний не одиничний, а нульовий сигнал. Тоді на входи шифратора в довільний момент часу надходить код, лише один з розрядів якого дорівнює нулю, а решта – одиниці, тобто інверсний унарний код. У такому випадку входи шифратора позначають \overline{L}_i , а таблиця функціонування шифратора має вигляд як показано в табл. 5.8.

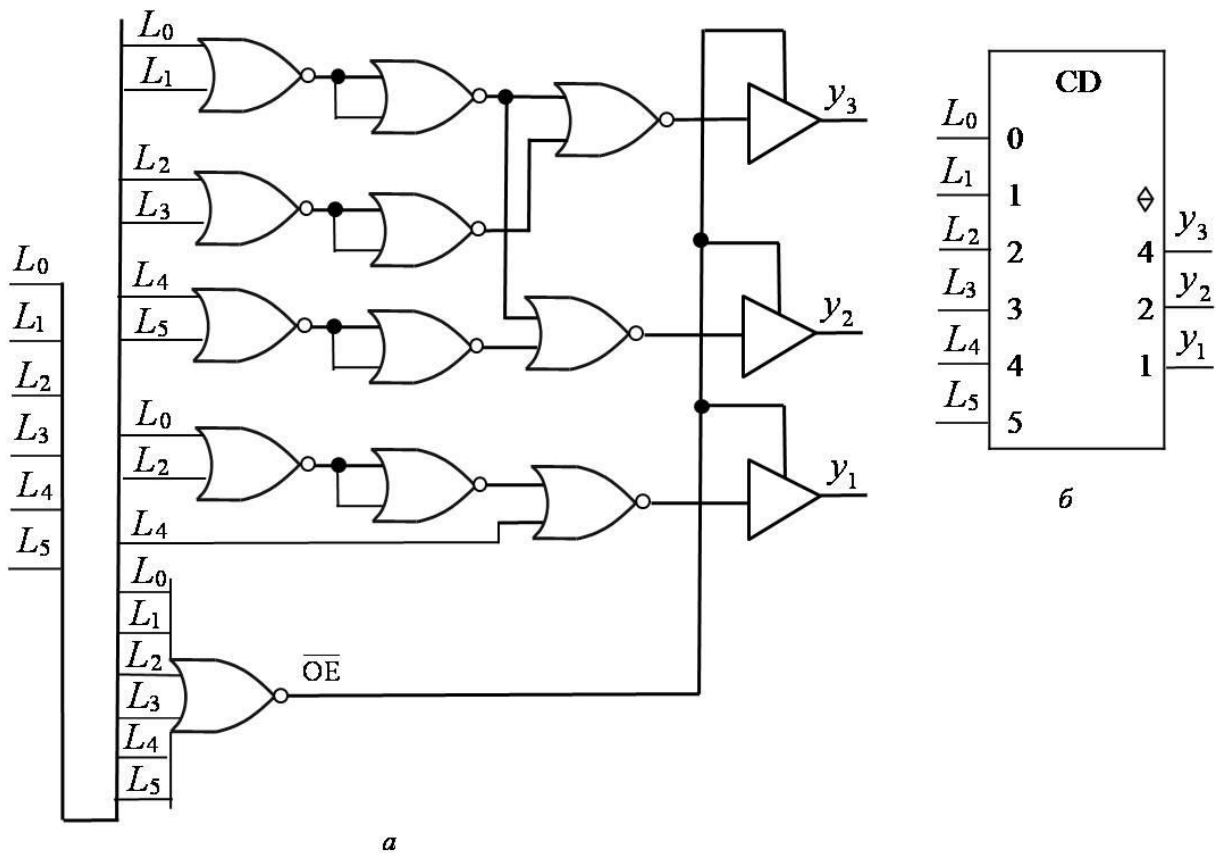


Рис. 5.16. Шифратор 6×3 з прямими входами:
a – функціональна схема; *б* – умовне графічне позначення

Таблиця 5.8

Таблиця функціонування шифратора 8×3 з інверсними входами

Входи								Виходи		
\bar{L}_7	\bar{L}_6	\bar{L}_5	\bar{L}_4	\bar{L}_3	\bar{L}_2	\bar{L}_1	\bar{L}_0	y_3	y_2	y_1
1	1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	0	1	0	0	1
1	1	1	1	1	0	1	1	0	1	0
1	1	1	1	0	1	1	1	0	1	1
1	1	1	0	1	1	1	1	1	0	0
1	1	0	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	1

З таблиці істинності (функціонування) визначаємо функції виходів шифратора:

$$\begin{cases} y_3 = \bar{L}_0 \cdot \bar{L}_1 \cdot \bar{L}_2 \cdot \bar{L}_3, \\ y_2 = \bar{L}_0 \cdot \bar{L}_1 \cdot \bar{L}_4 \cdot \bar{L}_5, \\ y_1 = \bar{L}_0 \cdot \bar{L}_2 \cdot \bar{L}_4 \cdot \bar{L}_6. \end{cases} \quad (5.4)$$

Цю систему функцій реалізує комбінаційна схема на рис. 5.17, яка є функціональною схемою шифратора з інверсними входами.

Логічний елемент 8І, на який надходить код $\bar{L}_7 \bar{L}_6 \dots \bar{L}_0$, формує сигнал \overline{OE} дозволу видачі слова y_3, y_2, y_1 з виходів шифратора. Активним рівнем сигналу \overline{OE} є логічний нуль. $\overline{OE} = 0$, якщо якийсь з 8-ми пристроїв надсилає сигнал “0” на відповідний вхід шифратора (на виходах решти 7-ми пристроїв – рівень “1”), у цьому випадку на виходах y_3, y_2, y_1 отримуємо номер цього пристрою.

Якщо жоден з пристроїв, з’єднаних з входами шифратора, не надсилає сигнал “0”, тобто на входах шифратора має місце код 11111111, то $\overline{OE} = 1$, і керовані повторювачі на виходах схеми переходять у третій стан – шифратор ніби від’єднується від решти блоків (хоча електрично він з ними з’єднаний).

Шифратор 8×3 з інверсними входами можна побудувати також на основі логічних елементів І-НЕ. Для цього замість системи (5.4) необхідно записати систему для інверсій функцій y_3, y_2, y_1 , тобто у вигляді (див. табл. 5.8):

$$\begin{cases} \bar{y}_3 = \bar{L}_4 \cdot \bar{L}_5 \cdot \bar{L}_6 \cdot \bar{L}_7, \\ \bar{y}_2 = \bar{L}_2 \cdot \bar{L}_3 \cdot \bar{L}_6 \cdot \bar{L}_7, \\ \bar{y}_1 = \bar{L}_1 \cdot \bar{L}_3 \cdot \bar{L}_5 \cdot \bar{L}_7. \end{cases} \quad (5.5)$$

Систему (5.5) перетворимо так:

$$\begin{cases} y_3 = \overline{\bar{L}_4 \cdot \bar{L}_5 \cdot \bar{L}_6 \cdot \bar{L}_7}, \\ y_2 = \overline{\bar{L}_2 \cdot \bar{L}_3 \cdot \bar{L}_6 \cdot \bar{L}_7}, \\ y_1 = \overline{\bar{L}_1 \cdot \bar{L}_3 \cdot \bar{L}_5 \cdot \bar{L}_7}. \end{cases} \quad (5.6)$$

Для побудови шифратора з інверсними входами на основі елементів І-НЕ необхідно (див. рис. 5.17):

- 1) замінити 4-входові елементи І на 4-входові елементи І-НЕ та з’єднати їх з входами схеми відповідно до системи (5.6),
- 2) замінити 8-входовий елемент І на 8-входовий елемент І-НЕ,
- 3) на виходах схеми використати керовані повторювачі, у яких переведення у третій стан здійснюють логічним нулем.

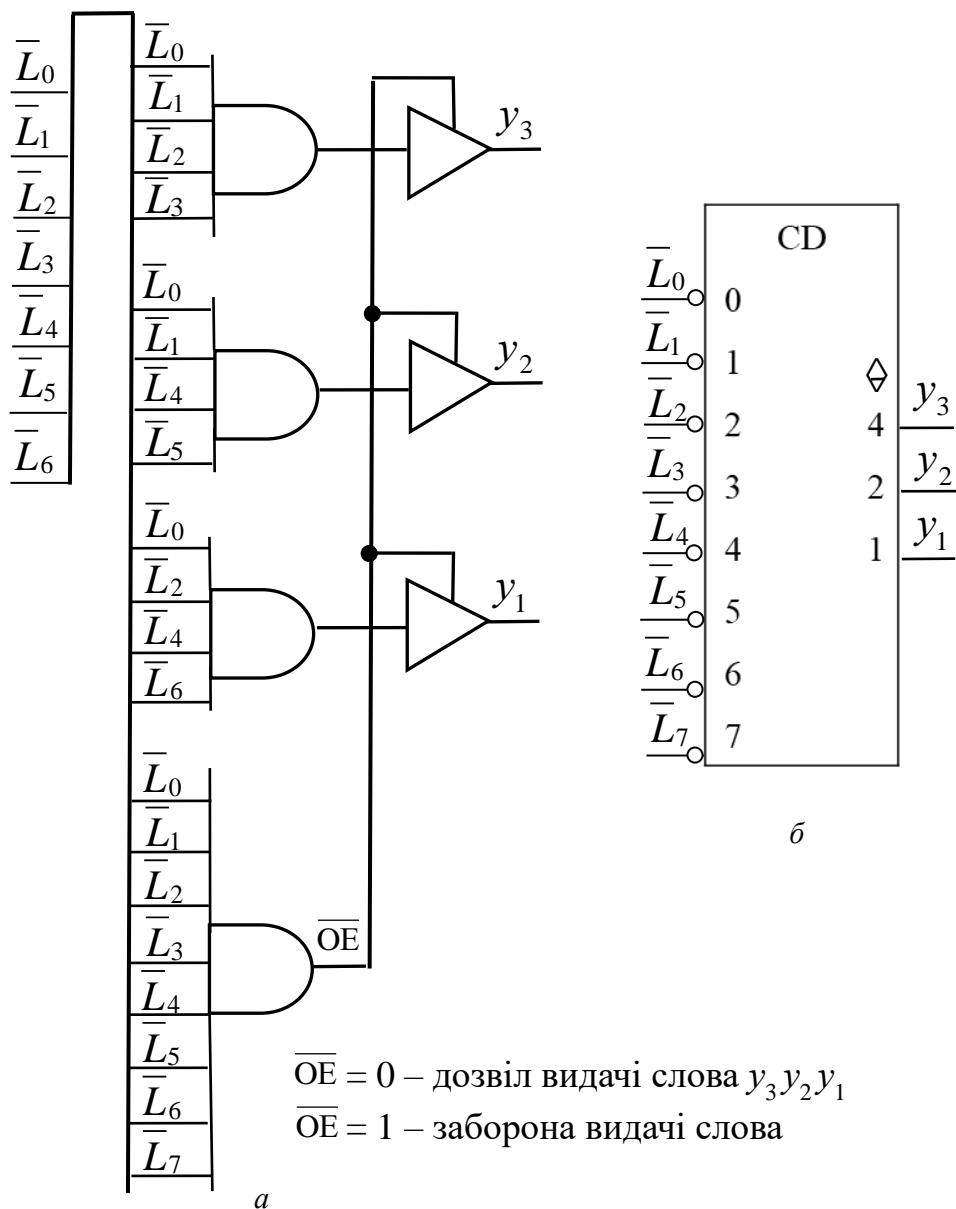


Рис. 5.17. Шифратор 8×3 з інверсними входами:
 а – функціональна схема; б – умовне графічне позначення

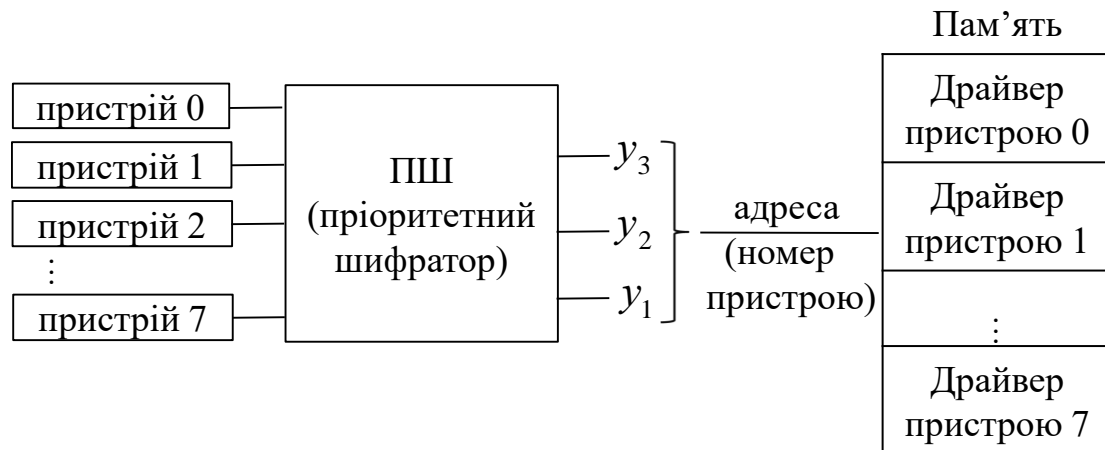
В блоках оброблення пріоритетних переривань від зовнішніх пристроїв комп'ютера використовують пріоритетні шифратори.

Кожен пристрій, що входить до складу комп'ютера (клавіатура, миша, принтер, дисковод тощо), в будь-який момент часу може надіслати запит на переривання (на його обслуговування). Запити на переривання можуть надійти одночасно від кількох пристроїв. Щоб усунути можливу конфліктність (адже реально в будь-який момент часу процесор може обслуговувати лише один пристрій) зовнішні пристрої наділяють пріоритетами. Наприклад, якщо у системі є вісім зовнішніх пристроїв, то

пристрій з номером 7 може мати найвищий пріоритет, пристрій з номером 0 – найнижчий (рис. 5.18) або навпаки.

Блок оброблення переривань працює за такою стратегією – у будь-який момент часу має обслуговуватись пристрій з найвищим пріоритетом серед тих, що надіслали запити на обслуговування.

Функціональний вузол, що забезпечує розпізнавання запитів від зовнішніх пристроїв, які наділені пріоритетами, та функціонує відповідно до сформульованої стратегії, називають *пріоритетним шифратором* (priority encoder, PCD).



u_3, u_2, u_1 – код номера пристрою (вектор переривання, адреса драйвера)

Рис. 5.18. Структурна організація блока пріоритетних переривань комп'ютера

На виході пріоритетного шифратора формується номер пристрою з найвищим пріоритетом серед тих, які в даний момент часу потребують обслуговування. Цей номер є адресою драйвера зовнішнього пристрою. *Драйвер* – це програма, яка зберігається в пам'яті комп'ютера та забезпечує взаємодію процесора з даним пристроєм. Кожний зовнішній пристрій має свій унікальний драйвер (клавіатура – драйвер клавіатури, принтер – драйвер принтера тощо), який вибирається в пам'яті за номером (адресою), що формується на виході пріоритетного шифратора.

Якщо в деякий момент часу одночасно надійдуть запити на переривання, наприклад, від пристроїв з номерами 3, 5, 6, то обслуговуватись має пристрій з номером 6, оскільки він наділений вищим пріоритетом порівняно з пристроями 3 та 5.

Шифратор, що забезпечує такий спосіб обслуговування (розпізнавання) восьми зовнішніх пристроїв, функціонує за таблицею істинності (табл. 5.9), де 7 – найвищий пріоритет, 0 – найнижчий, а \times – довільне значення вхідного сигналу (0 або 1).

Таблиця 5.9

Таблиця функціонування 8-входового пріоритетного шифратора з прямими входами

Входи								Виходи		
I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	y_3	y_2	y_1
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	×	0	0	1
0	0	0	0	0	1	×	×	0	1	0
0	0	0	0	1	×	×	×	0	1	1
0	0	0	1	×	×	×	×	1	0	0
0	0	1	×	×	×	×	×	1	0	1
0	1	×	×	×	×	×	×	1	1	0
1	×	×	×	×	×	×	×	1	1	1

Рядок 0001×××× таблиці означає, що якщо запит на обслуговування надсилає пристрій з номером 4, то незалежно від того, чи надсилатимуть одночасно з ним запити пристрої з номерами 0 – 3, обслуговуватиметься лише пристрій з номером 4 (код 100 на виходах шифратора).

Однак, запити на обслуговування від зовнішніх пристроїв можуть надходити у вигляді сигналів низького рівня (логічного нуля). Тоді функціонування пріоритетного шифратора у випадку 8-ми зовнішніх пристроїв задається табл. 5.10, де \bar{I}_i , $i = 0, 1, \dots, 7$, – входи пріоритетного шифратора (рис. 5.18), де 7 – найвищий, 0 – найнижчий пріоритет.

Таблиця 5.10

Таблиця функціонування 8-входового пріоритетного шифратора з інверсними входами

Входи								Виходи		
\bar{I}_7	\bar{I}_6	\bar{I}_5	\bar{I}_4	\bar{I}_3	\bar{I}_2	\bar{I}_1	\bar{I}_0	y_3	y_2	y_1
1	1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	0	×	0	0	1
1	1	1	1	1	0	×	×	0	1	0
1	1	1	1	0	×	×	×	0	1	1
1	1	1	0	×	×	×	×	1	0	0
1	1	0	×	×	×	×	×	1	0	1
1	0	×	×	×	×	×	×	1	1	0
0	×	×	×	×	×	×	×	1	1	1

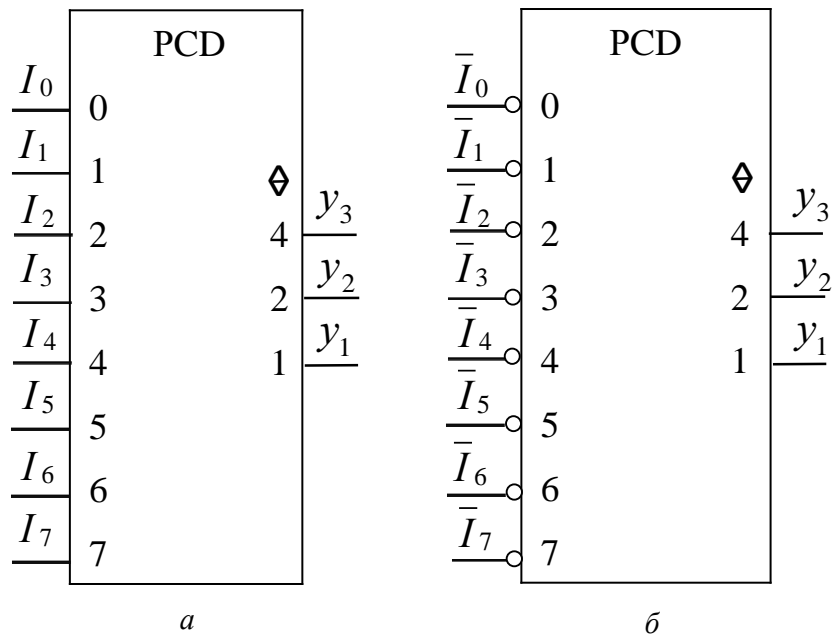


Рис. 5.19. Умовне графічне позначення 8-входового пріоритетного шифратора:
a – з прямими входами; *б* – з інверсними входами

Задача 5.4. Синтезувати 4-входовий пріоритетний шифратор з прямими входами.

Розв'язування.

1. Складемо таблицю функціонування пріоритетного шифратора (табл. 5.11), де 3 – найвищий пріоритет, 0 – найнижчий.

Таблиця 5.11
Таблиця функціонування 4-входового пріоритетного шифратора з прямими входами

Входи				Виходи	
I_3	I_2	I_1	I_0	y_2	y_1
0	0	0	1	0	0
0	0	1	×	0	1
0	1	×	×	1	0
1	×	×	×	1	1

2. Запишемо аналітичні вирази для функцій виходів y_2 , y_1 пріоритетного шифратора. Вирази можна отримати в диз'юнктивній або кон'юнктивній формах, тобто як диз'юнкцію конститuent одиниці або як кон'юнкцію конститuent нуля.

Деякі приклади утворення конституент одиниці та конституент нуля:

вхідний набір $I_3 I_2 I_1 I_0$	конституента одиниці	конституента нуля
0 0 0 1	$\overline{I_3} \overline{I_2} \overline{I_1} I_0$	$I_3 \vee I_2 \vee I_1 \vee \overline{I_1}$
0 0 1 ×	$\overline{I_3} \overline{I_2} I_1$	$I_3 \vee I_2 \vee \overline{I_1}$
1 × × ×	I_3	$\overline{I_3}$

Запишемо вирази для функцій виходів y_2, y_1 в диз'юнктивній формі:

$$\begin{cases} y_2 = \overline{I_3} I_2 \vee I_3 = (\overline{I_3} \vee I_3)(I_2 \vee I_3) = I_2 \vee I_3, \\ y_1 = \overline{I_3} \overline{I_2} I_1 \vee I_3 = \overline{I_3} (\overline{I_2} I_1) \vee I_3 = (\overline{I_3} \vee I_3)(\overline{I_2} I_1 \vee I_3) = \overline{I_2} I_1 \vee I_3. \end{cases}$$

При перетворенні цих виразів застосовано властивість дистрибутивності диз'юнкції щодо кон'юнкції, а саме $a \vee bc = (a \vee b)(a \vee c)$ або $bc \vee a = (b \vee a)(c \vee a)$.

Таким чином, функції виходів пріоритетного шифратора утворюють систему

$$\begin{cases} y_2 = I_2 \vee I_3, \\ y_1 = \overline{I_2} I_1 \vee I_3. \end{cases} \quad (5.7)$$

3. На основі системи (5.7) будемо схему пріоритетного шифратора (рис. 5.20). ▣

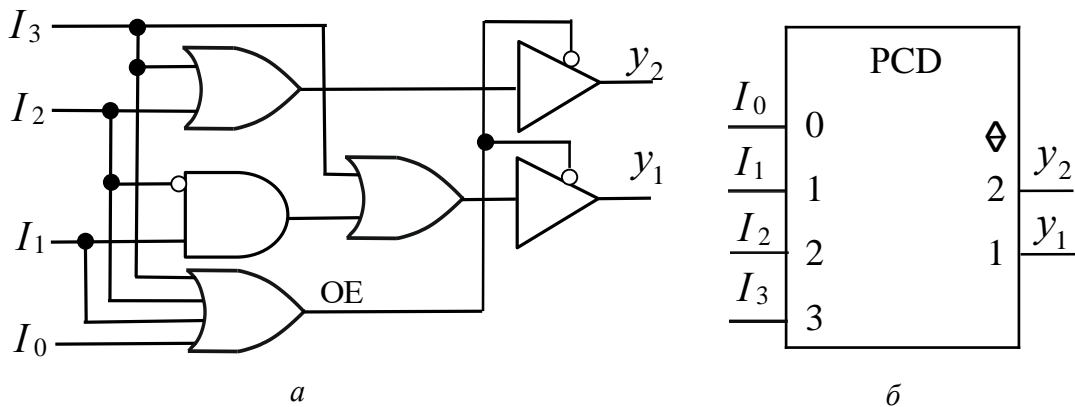


Рис. 5.20. Чотиривходовий пріоритетний шифратор з прямими входами:
а – функціональна схема; б – умовне графічне позначення

Для функцій виходів пріоритетного шифратора можна застосувати й кон'юнктивну форму запису – кон'юнкцію конституент нуля.

Тоді система, що описує поведінку пріоритетного шифратора матиме вигляд:

$$\begin{cases} \overline{y_2} = (I_3 \vee \overline{I_2}) \cdot \overline{I_3} = \overline{I_2} \overline{I_3}, \\ \overline{y_1} = (I_3 \vee I_2 \vee \overline{I_1}) \cdot \overline{I_3} = (I_2 \vee \overline{I_1}) \overline{I_3} \end{cases} \quad \text{або} \quad \begin{cases} y_2 = \overline{\overline{I_2} \overline{I_3}}, \\ y_1 = \overline{(I_2 \vee \overline{I_1}) \overline{I_3}}. \end{cases} \quad (5.8)$$

Системи (5.8) та (5.7) є еквівалентними, систему (5.8) можна отримати, якщо до рівнянь системи (5.7) застосувати правило де Моргана. Це означає, що комбінаційна схема, побудована на основі системи (5.8) матиме таку саму апаратну складність, що й комбінаційна схема, побудована на основі системи (5.7).

Задача 5.5. Синтезувати 4-входовий пріоритетний шифратор з інверсними входами.

Розв'язування.

1. Складемо таблицю функціонування пріоритетного шифратора (табл. 5.12), де \bar{I}_i ($i = 0, 1, 2, 3$) – входи шифратора, 3 – найвищий, а 0 – найнижчий пріоритет.
2. Запишемо вирази для функцій виходів шифратора в кон'юнктивній формі:

$$\begin{cases} \bar{y}_2 = (\bar{I}_3 \vee \bar{I}_2) \cdot \bar{I}_3 = \bar{I}_2 \bar{I}_3, \\ \bar{y}_1 = (\bar{I}_3 \vee \bar{I}_2 \vee \bar{I}_1) \cdot \bar{I}_3 = (\bar{I}_2 \vee \bar{I}_1) \bar{I}_3. \end{cases}$$

Таблиця 5.12

Таблиця функціонування 4-входового пріоритетного шифратора з інверсними входами

Входи				Виходи	
\bar{I}_3	\bar{I}_2	\bar{I}_1	\bar{I}_0	y_2	y_1
1	1	1	0	0	0
1	1	0	×	0	1
1	0	×	×	1	0
0	×	×	×	1	1

Після перетворення рівнянь отримаємо систему:

$$\begin{cases} y_2 = \bar{I}_2 \bar{I}_3, \\ y_1 = (\bar{I}_2 \vee \bar{I}_1) \bar{I}_3. \end{cases} \quad (5.9)$$

3. Побудуємо комбінаційну схему, що реалізує систему перемикальних функцій (5.9) (рис. 5.21). ▣

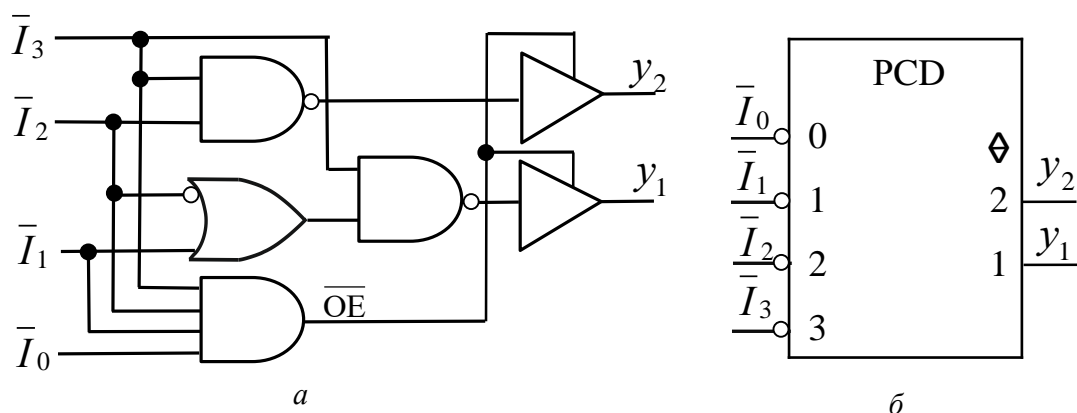


Рис. 5.21. Чотиривходовий пріоритетний шифратор з інверсними входами: а – функціональна схема; б – умовне графічне позначення

Подати функції пріоритетного шифратора з інверсними входами можна також в диз'юнктивній формі:

$$\begin{cases} y_2 = \bar{I}_3 \bar{I}_2 \bar{I}_3 = (\bar{I}_3 \vee \bar{I}_3)(\bar{I}_2 \vee \bar{I}_3) = \bar{I}_2 \vee \bar{I}_3, \\ y_1 = \bar{I}_3 \bar{I}_2 \bar{I}_1 \bar{I}_3 = \bar{I}_3 (\bar{I}_2 \bar{I}_1) \bar{I}_3 = (\bar{I}_3 \vee \bar{I}_3)(\bar{I}_2 \bar{I}_1 \vee \bar{I}_3) = \bar{I}_2 \bar{I}_1 \vee \bar{I}_3, \end{cases}$$

тобто

$$\begin{cases} y_2 = \bar{I}_2 \vee \bar{I}_3, \\ y_1 = \bar{I}_2 \bar{I}_1 \vee \bar{I}_3. \end{cases} \quad (5.10)$$

Системи перемикальних функцій (5.9) та (5.10) є еквівалентними, тому комбінаційні схеми, що їх реалізують, мають однакову складність.

Запитання та завдання

1. Дайте визначення шифратора.
2. Побудуйте таблицю функціонування шифратора 4×2 з прямими (інверсними) входами.
3. Запишіть систему функцій виходів шифратора 4×2 з прямими (інверсними) входами.
4. Поясніть, як функціонує керований повторювач.
5. Що означає поняття «вихід на три стани»?
6. Якою формулою пов'язані між собою у загальному випадку кількість входів та кількість виходів шифратора?
7. Дайте визначення пріоритетного шифратора.
8. Поясніть, як функціонує блок оброблення пріоритетних переривань комп'ютера.
9. Дано вхідні набори пріоритетного шифратора:

I_4	I_3	I_2	I_1	I_0
1	1	1	1	0
1	1	0	×	×
0	×	×	×	×

Утворіть конституенти одиниці та конституенти нуля для кожного із заданих наборів.

Задачі для самостійного розв'язування

1. Побудувати шифратор 7×3 з прямими входами на логічних елементах АБО.
2. Побудувати шифратор 7×3 з інверсними входами на логічних елементах І-НЕ.
3. Синтезувати шифратор 6×3 з інверсними входами на логічних елементах 2І-НЕ.

4. Синтезувати шифратор 5×3 з прямими входами на логічних елементах 2АБО-НЕ.
5. Побудувати шифратор 5×3 з інверсними входами на логічних елементах 2І-НЕ.
6. Синтезувати 6-входовий пріоритетний шифратор з прямими входами. Функції виходів шифратора подати в кон'юнктивній формі.
7. Синтезувати 6-входовий пріоритетний шифратор з інверсними входами. Функції виходів шифратора подати в диз'юнктивній формі.
8. Побудувати 5-входовий пріоритетний шифратор з прямими входами. Функції виходів шифратора подати в диз'юнктивній формі.

5.3. Перетворювачі кодів

У цифровій обчислювальній техніці використовують різноманітні перетворювачі кодів.

Перетворювач коду (code converter) – це комбінаційна схема, що має k входів та n виходів, яка реалізує перетворення k -розрядного слова, що надходить на входи схеми, в n -розрядне слово на виходах схеми.

Прикладами перетворювачів є схеми для:

- перетворення чисел з однієї системи числення в іншу,
- керування символічними індикаторами,
- перетворення одного двійково-десятькового коду (ДДК) в інший ДДК,
- перетворення позиційного коду в унарний код (дешифратор),
- перетворення унарного коду в позиційний (шифратор) тощо.

Універсальним способом реалізації перетворювачів кодів є використання схеми побудови «дешифратор-шифратор» (схема побудови «DC-CD») (рис. 5.22).

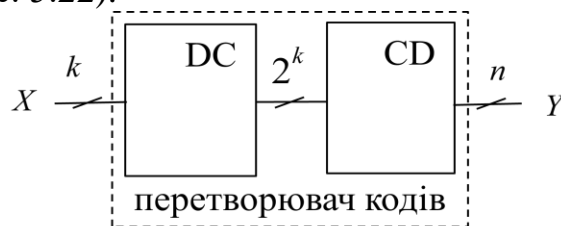


Рис. 5.22. Структура перетворювача кодів на основі схеми побудови «дешифратор-шифратор»

Дешифратор на k входів (на входах k -розрядний позиційний код X) забезпечує отримання 2^k конститuent одиниці (2^k -розрядний унарний код), а шифратор перетворює 2^k -розрядний унарний код в n -розрядний код Y .

У такій схемі побудови може використовуватись й неповний дешифратор.

Іншим способом реалізації перетворювачів є побудова (синтез) комбінаційної схеми на основі лише логічних елементів.

Застосуємо схему побудови «дешифратор-шифратор» для реалізації перетворювача двійково-десятькового коду «8-4-2-1» в код Грея (табл. 5.13), де 8, 4, 2, 1 – ваги двійкових розрядів.

Таблиця 5.13

Таблиця функціонування перетворювача ДДК «8-4-2-1» в код Грея

Десяткова цифра	Двійково-десятьковий код «8-4-2-1»				Код Грея			
	x_4	x_3	x_2	x_1	y_4	y_3	y_2	y_1
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	0	0	0

Код Грея, який також називають *рефлексним кодом*, використовують у пристроях, що перетворюють лінійні або кутові механічні переміщення в цифрову форму. У коді Грея двом будь-яким сусіднім десятковим цифрам відповідають двійкові тетради, що відрізняються лише в одному розряді.

Для побудови перетворювача використаємо неповний дешифратор 4×10 з прямими виходами. Тоді функції виходів y_4, y_3, y_2, y_1 перетворювача мають вигляд:

$$\begin{cases} y_4 = 8 \vee 9, \\ y_3 = 4 \vee 5 \vee 6 \vee 7 \vee 8, \\ y_2 = 2 \vee 3 \vee 4 \vee 5, \\ y_1 = 1 \vee 2 \vee 5 \vee 6, \end{cases} \quad (5.11)$$

де цифрами 1 – 9 позначено відповідні набори змінних $x_4 - x_1$, тобто номери виходів дешифратора.

Системі перемикальних функцій (5.11) відповідає комбінаційна схема на рис. 5.23а.

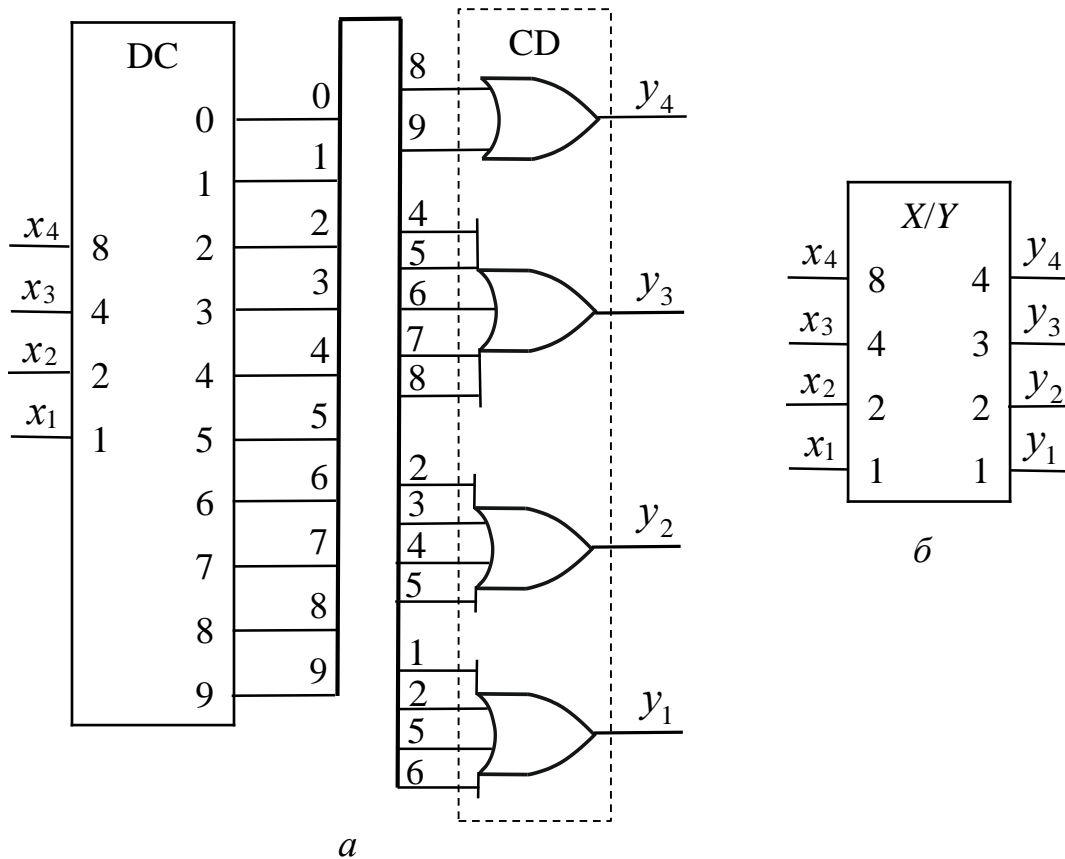


Рис. 5.23. Перетворювач ДДК «8-4-2-1» в код Грея:
a – функціональна схема; *б* – умовне графічне позначення

На умовному графічному позначенні перетворювача (рис. 5.23*б*) позначка X/Y означає, що вхідний код X перетворюється у вихідний код Y .

Синтезуємо перетворювач ДДК «8-4-2-1» в код Грея в елементному базисі алгебри Буля.

Функції $y_4 - y_1$ з табл. 5.13 розглядатимемо як систему неповністю визначених перемикальних функцій, які не визначені на 6-ти наборах: 1010 – 1111.

Виконаємо мінімізацію кожної з функцій за допомогою діаграми Вейча (рис. 5.24):

$$\begin{cases} y_4 = x_4, \\ y_3 = x_3 \vee x_4 \bar{x}_1, \\ y_2 = x_2 \bar{x}_3 \vee x_3 \bar{x}_2, \\ y_1 = x_2 \bar{x}_1 \vee \bar{x}_4 \bar{x}_2 x_1. \end{cases} \quad (5.12)$$

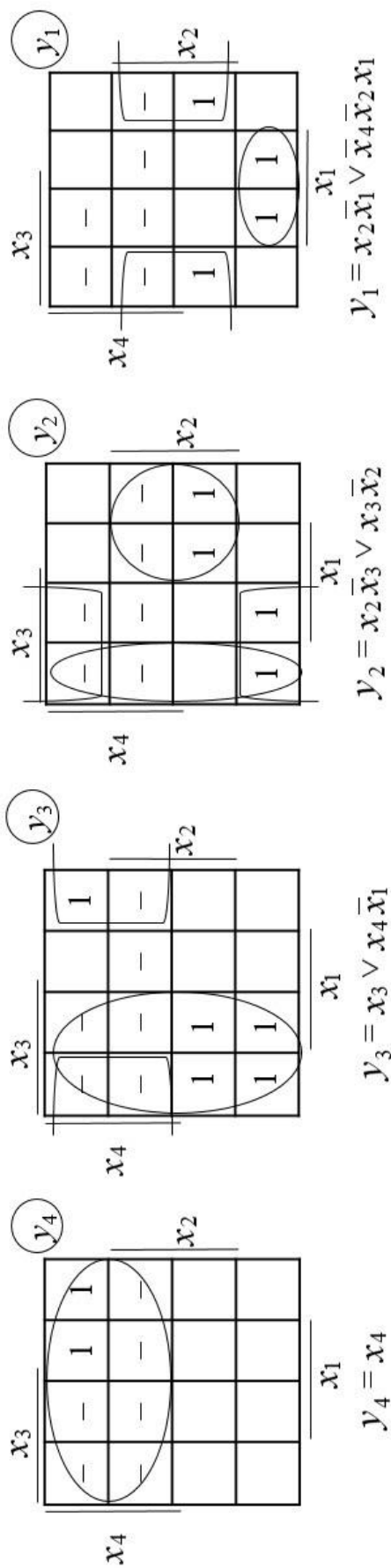


Рис. 5.24. Мінімізація функцій виходів перетворювача ДДК «8-4-2-1» в код Грея

Таблиця 5.14

Приклади деяких двійково-десяткових кодів

Десяткова цифра	Код «8-4-2-1»				Код «2 з 5»					Код «4-2-2-1»				Код з надлишком 3				Код «5-4-2-1»			
	x4	x3	x2	x1	y5	y4	y3	y2	y1	y4	y3	y2	y1	y4	y3	y2	y1	y4	y3	y2	y1
0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	1	0	0	1	0	0	0	0	1
2	0	0	1	0	0	0	1	0	1	0	0	1	0	0	1	0	1	0	0	1	0
3	0	0	1	1	0	0	1	1	0	0	0	1	1	0	0	1	0	0	0	1	1
4	0	1	0	0	0	1	0	0	1	0	1	1	0	0	1	1	1	0	1	0	0
5	0	1	0	1	0	1	0	1	0	1	0	0	1	0	0	0	0	1	0	0	0
6	0	1	1	0	0	1	1	0	0	1	1	0	0	1	0	0	1	1	0	0	1
7	0	1	1	1	1	0	0	0	1	1	1	0	1	0	1	0	1	1	0	1	0
8	1	0	0	0	1	0	0	1	0	1	1	1	0	1	1	0	1	1	0	1	1
9	1	0	0	1	1	0	1	0	0	1	1	1	1	1	1	0	0	1	1	0	0

Системі перемикальних функцій (5.12) відповідає комбінаційна схема на рис. 5.25, яка є функціональною схемою перетворювача ДДК «8-4-2-1» в код Грея.

Комбінаційна схема перетворювача на рис. 5.25 має меншу складність, ніж схема на рис. 5.23а (з урахуванням складності дешифратора 4×10).

Крім ДДК «8-4-2-1» та коду Грея в цифрових обчислювальних засобах для подання 10-кових цифр використовують й інші ДДК (табл. 5.14).

В апаратурі відображення цифрової інформації широке застосування знаходять перетворювачі двійково-десяткових кодів в семирозрядні коди цифрових індикаторів. У такому індикаторі кожній десятковій цифрі відповідає світіння певної комбінації сегментів $y_7 - y_1$ (рис. 5.26).

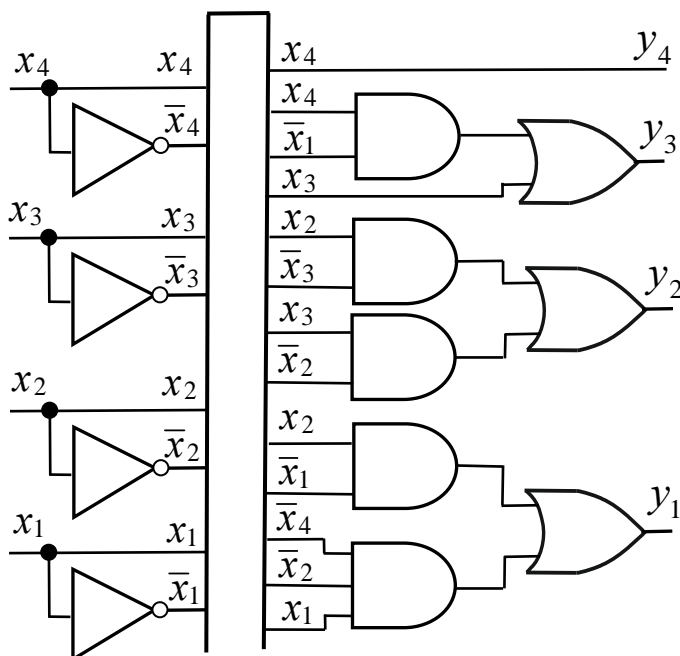


Рис. 5.25. Комбінаційна схема перетворювача ДДК «8-4-2-1» в код Грея на основі елементного базису алгебри Буля

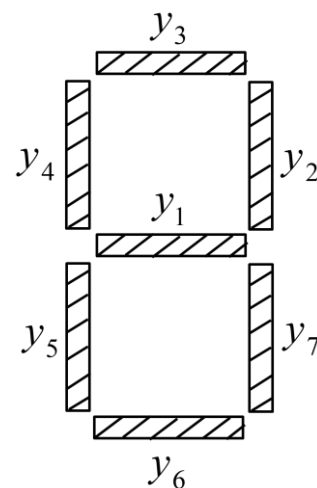


Рис. 5.26. Семисегментний цифровий індикатор

Такий індикатор функціонує за таблицею істинності (табл. 5.15).

Реалізуємо перетворювач за схемою побудови «DC-CD». Оскільки таблиця функціонування перетворювача (стовпці $y_7 - y_1$) містить мало нулів (21 з 70 цифр), то перетворювач доцільно побудувати на основі дешифратора з інверсними виходами. Це істотно знижує складність схеми.

Таблиця 5.15

Таблиця істинності перетворювача кодів для керування семисегментним цифровим індикатором

Десяткова цифра	Входи				Виходи						
	x_4	x_3	x_2	x_1	y_7	y_6	y_5	y_4	y_3	y_2	y_1
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	1	0	0	0	0	1	0
2	0	0	1	0	0	1	1	0	1	1	1
3	0	0	1	1	1	1	0	0	1	1	1
4	0	1	0	0	1	0	0	1	0	1	1
5	0	1	0	1	1	1	0	1	1	0	1
6	0	1	1	0	1	1	1	1	1	0	1
7	0	1	1	1	1	0	0	0	1	1	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	0	1	1	1	1

Роботу перетворювача описують системою перемикальних функцій:

$$\begin{cases} y_7 = \bar{2}, \\ y_6 = \bar{1} \cdot \bar{4} \cdot \bar{7}, \\ y_5 = \bar{1} \cdot \bar{3} \cdot \bar{4} \cdot \bar{5} \cdot \bar{7} \cdot \bar{9}, \\ y_4 = \bar{1} \cdot \bar{2} \cdot \bar{3} \cdot \bar{7}, \\ y_3 = \bar{1} \cdot \bar{4}, \\ y_2 = \bar{5} \cdot \bar{6}, \\ y_1 = \bar{0} \cdot \bar{1} \cdot \bar{7}, \end{cases} \quad (5.13)$$

де $\bar{1} - \bar{9}$ – номери інверсних виходів дешифратора.

Системі (5.13) відповідає комбінаційна схема на рис. 5.27а.

Застосування схеми побудови «дешифратор-шифратор» є універсальним способом реалізації перетворювачів кодів, але часто отримувані за цією схемою структури перетворювачів не є найбільш економічними та швидкодіючими. Тому для побудови конкретно заданого перетворювача застосовують окремі методики та прийоми.

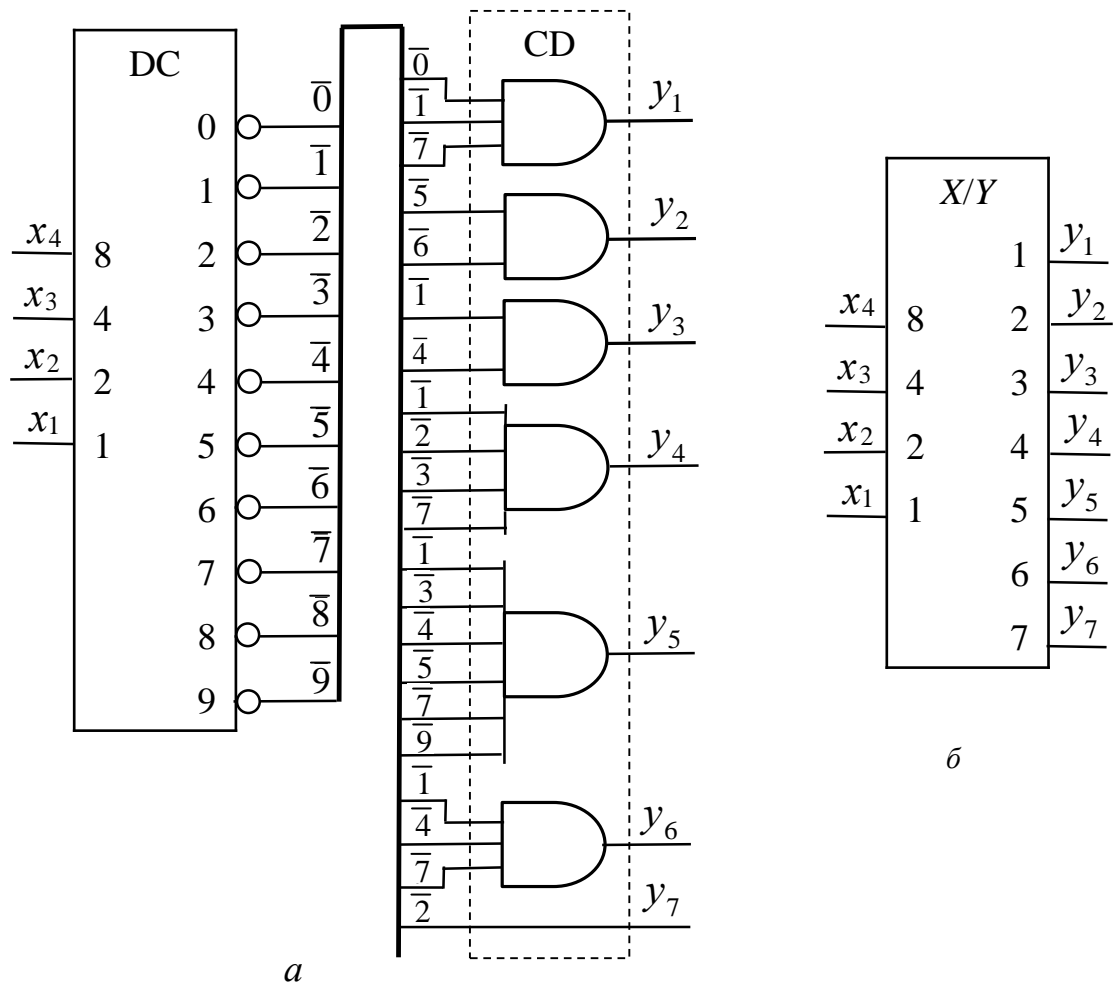


Рис. 5.27. Перетворювач кодів для керування семисегментним цифровим індикатором:
 а – функціональна схема; б – умовне графічне позначення

Затитання та завдання

1. Дайте визначення перетворювача кодів.
2. Наведіть приклади перетворювачів кодів.
3. Назвіть способи побудови перетворювачів кодів.
4. Охарактеризуйте спосіб реалізації перетворювачів кодів за схемою побудови «дешифратор-шифратор».
5. Як будують код Грея для подання десяткових цифр? Де застосовують перетворювач ДДК «8-4-2-1» в код Грея?
6. Як синтезують перетворювач кодів для керування цифровим індикатором?

Задачі для самостійного розв'язування

1. Записати систему функцій виходів перетворювача ДДК «8-4-2-1» в код Грея у випадку застосування схеми побудови «дешифратор-шифратор» з використанням ДС (4×10) з інверсними виходами.
2. Реалізувати перетворювач ДДК «8-4-2-1» в «код з надлишком 3» за схемою побудови «ДС-СД» на основі дешифратора з прямими виходами.
3. Побудувати перетворювач «коду з надлишком 3» в код «2 з 5» на основі дешифратора з інверсними виходами.
4. Синтезувати перетворювач ДДК «8-4-2-1» в ДДК «5-4-2-1» в елементному базисі алгебри Буля.
5. Синтезувати перетворювач ДДК «8-4-2-1» в ДДК «4-2-2-1» на логічних елементах АБО-НЕ.

5.4. Мультиплексори

Мультиплексор (multiplexor, MUX або MX) – це функціональний вузол (комбінаційна схема), призначений для комутації сигналів з кількох вхідних напрямів на один вихідний напрям.

Відповідно до схеми функціонування k -входового мультиплексора (рис. 5.28) $D_0, D_1, D_2, \dots, D_{k-1}$ – вхідні напрями, а y – вихідний напрям. Код на входах керування S визначає, саме з якого вхідного напрямку сигнал буде скомутовано на вихід y .

Мультиплексор часто називають *комутатором*.

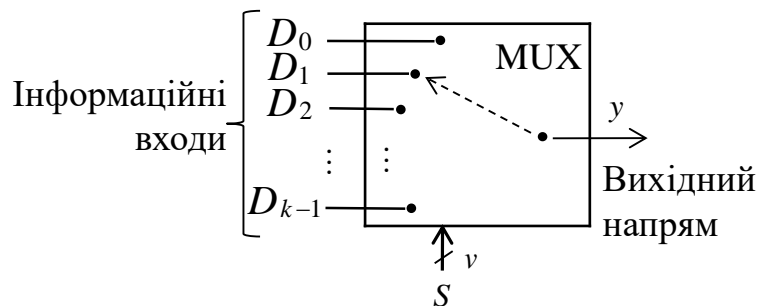


Рис. 5.28. Схема функціонування мультиплексора

Узагальнено мультиплексор позначають $MUX(k-1)$, де k – кількість вхідних напрямів, з яких надходять сигнали. Наприклад, $MUX(4-1)$ – мультиплексор, що комутує інформаційні сигнали з 4-х вхідних напрямів на один вихідний напрям; $MUX(2-1)$ – мультиплексор, що комутує інформаційні сигнали з 2-х вхідних напрямів на один вихідний напрям. Зазвичай $k = 2, 4, 8, 16, \dots$.

Якщо МХ (MUX) має k інформаційних входів, то кількість входів керування має становити $v = \lceil \log_2 k \rceil$, де v -розрядний код, встановлений на входах керування S у певний момент часу, вказує, який саме інформаційний вхід буде скомутовано на вихід схеми. Комбінаційна схема (2 варіанти) мультиплектора MUX(2 – 1), що комутує інформаційні сигнали з 2-х вхідних напрямів на один вихідний напрям, подана на рис. 5.29.

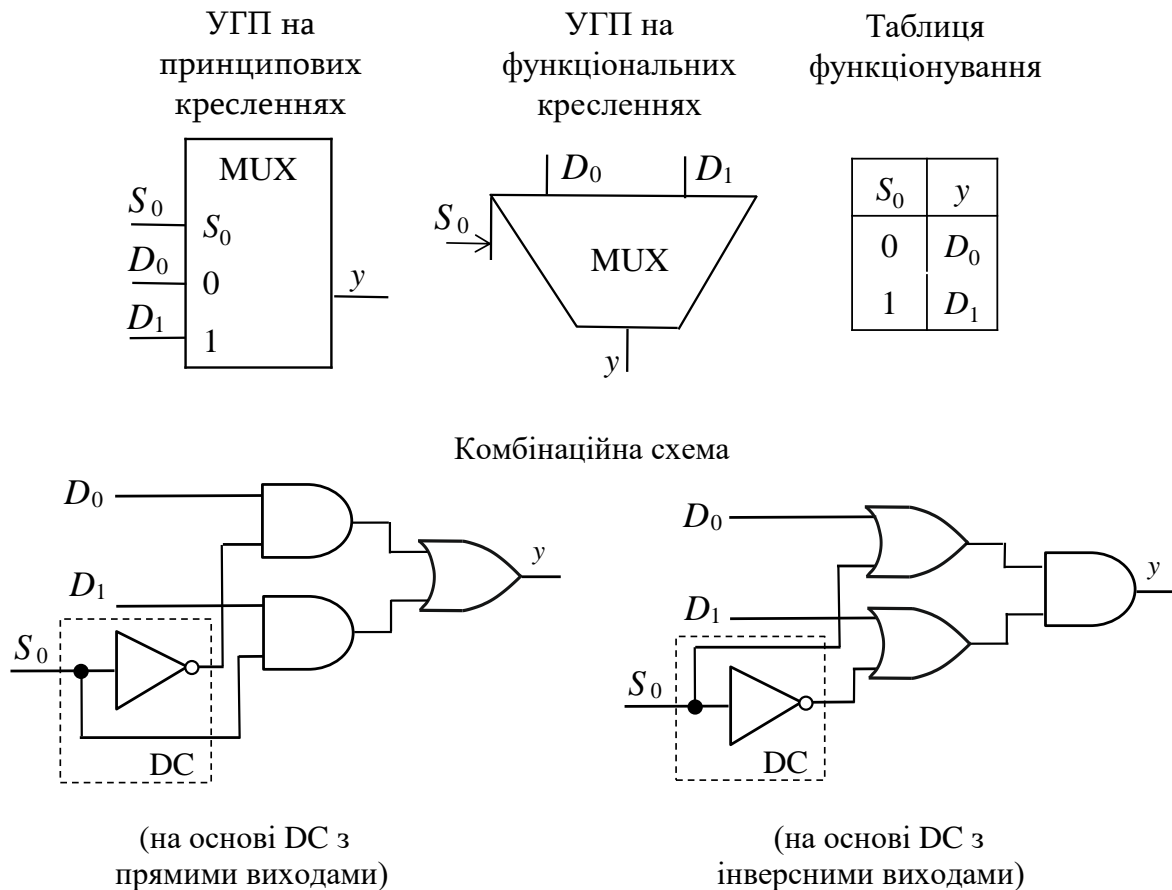


Рис. 5.29. Мультиплексор MUX(2 – 1)

Мультиплексор комутує сигнали з двох вхідних напрямів D_0 та D_1 на вихід y відповідно до таблиці функціонування. Якщо $S_0 = 0$, то на вихід y надходять сигнали з входу D_0 , а якщо $S_0 = 1$, то – з входу D_1 .

У комбінаційній схемі може використовуватись дешифратор з прямими або інверсними виходами. У схемі на рис. 5.29 використовується дешифратор на 2 виходи, до його складу входить лише один елемент – інвертор.

При $S_0 = 0$ у схемі мультиплексора на основі дешифратора з прямими виходами значення функції на виході y формують так:

$$y = D_0 \cdot 1 \vee D_1 \cdot 0 = D_0,$$

а в схемі на основі дешифратора з інверсними виходами –

$$y = (D_0 \vee 0)(D_1 \vee 1) = D_0 \cdot 1 = D_0.$$

При $S_0 = 1$ значення на виході y в зазначених комбінаційних схемах відповідно мають вигляд:

$$y = D_0 \cdot 0 \vee D_1 \cdot 1 = D_1,$$

$$y = (D_0 \vee 1)(D_1 \vee 0) = 1 \cdot D_1 = D_1.$$

Мультиплексор MUX(4–1) (рис. 5.30) забезпечує комутацію сигналів з 4-х вхідних напрямів D_0, D_1, D_2, D_3 на вихід y відповідно до таблиці функціонування мультиплексора:

$$y = \bar{S}_1 \bar{S}_0 D_0 \vee \bar{S}_1 S_0 D_1 \vee S_1 \bar{S}_0 D_2 \vee S_1 S_0 D_3$$

(на основі DC з прямими виходами)

$$\text{або } y = (S_1 \vee S_0 \vee D_0)(S_1 \vee \bar{S}_0 \vee D_1)(\bar{S}_1 \vee S_0 \vee D_2)(\bar{S}_1 \vee \bar{S}_0 \vee D_3)$$

(на основі DC з інверсними виходами).

Наприклад, якщо $S_1 = 1$, а $S_0 = 0$, то в обох схемах на вихід y комутуватимуться сигнали з вхідного напрямку D_2 .

Розрядність мультиплексора визначають кількістю ліній, що входять до складу кожного вхідного напрямку. Мультиплексори на рис. 5.29, 5.30 є однорозрядними.

В обчислювальних структурах крім однорозрядних можуть використовуватись багаторозрядні мультиплексори.

Розглянемо 4-розрядний MUX(2 – 1) (рис. 5.31).

До складу кожного з вхідних напрямів M та N входять по 4 лінії – $M_4 M_3 M_2 M_1$ та $N_4 N_3 N_2 N_1$ відповідно. До вихідного напрямку Y також входять 4 лінії – $Y_4 Y_3 Y_2 Y_1$.

Якщо $S_0 = 0$, то на вихід Y комутуються 4 біти з напрямку M , а якщо $S_0 = 1$, то – з напрямку N .

Комбінаційна схема 4-розрядного MUX(2 – 1) може бути побудована на основі дешифратора як з прямими (рис. 5.31), так і з інверсними виходами.

Під час синтезу комбінаційних схем у заданому елементному базисі найскладнішим є перший етап синтезу – мінімізація заданої функції. Процес мінімізації істотно ускладнюється зі збільшенням кількості аргументів перемикальної функції.

З метою спрощення процесу мінімізації функцій застосовують декомпозицію функції – перемикальну функцію від заданої кількості аргументів представляють як об'єднання кількох функцій, що залежать від меншої кількості аргументів. Тоді можна виконати мінімізацію окремо кожної з функцій з меншою кількістю аргументів (що значно простіше),

побудувати її оптимізовану комбінаційну схему та об'єднати результати (комбінаційні схеми) в єдину цілісну схему.

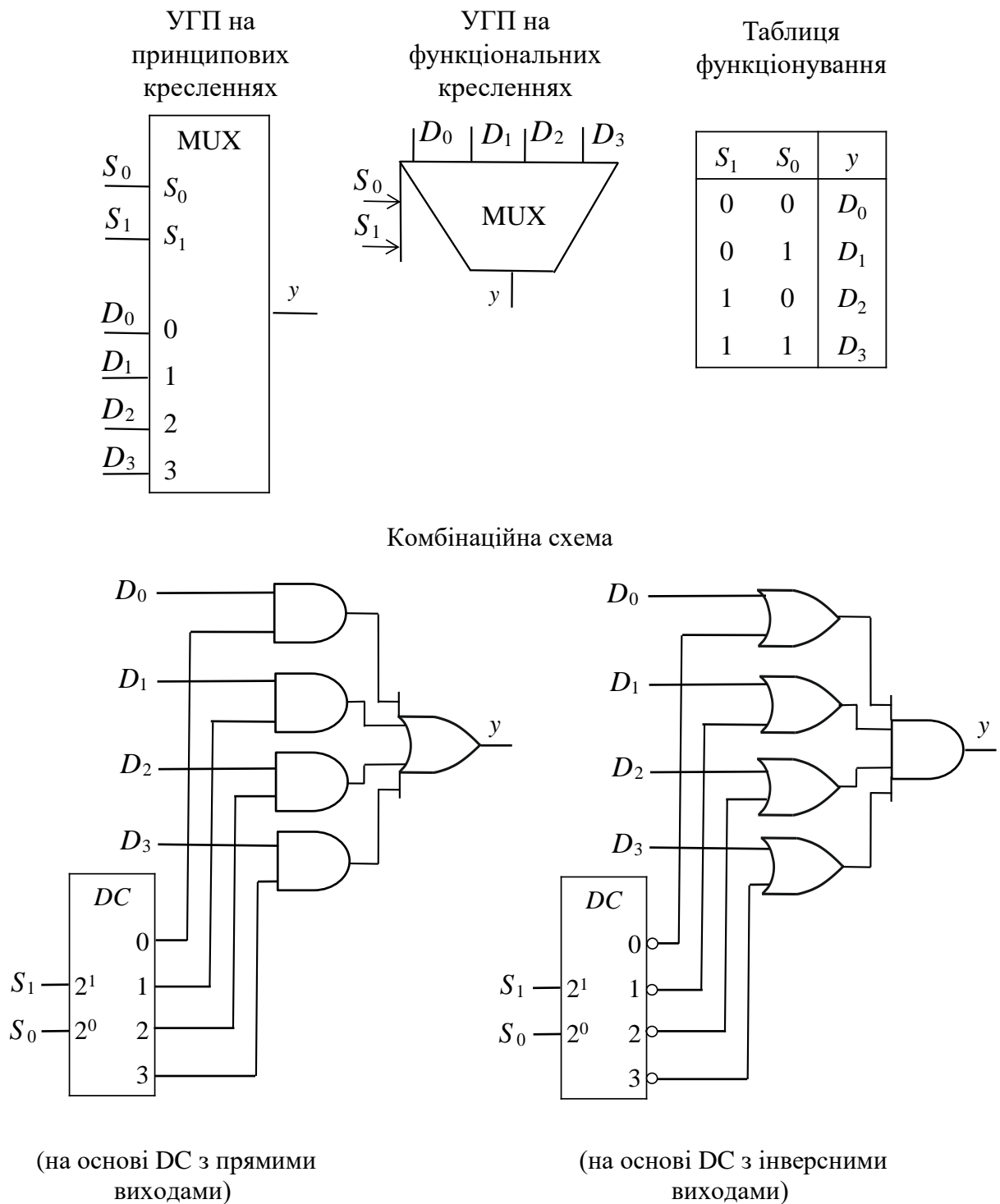
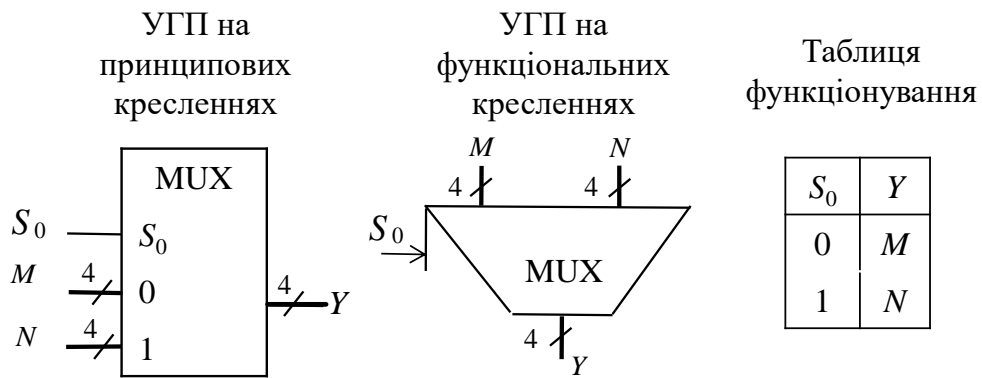
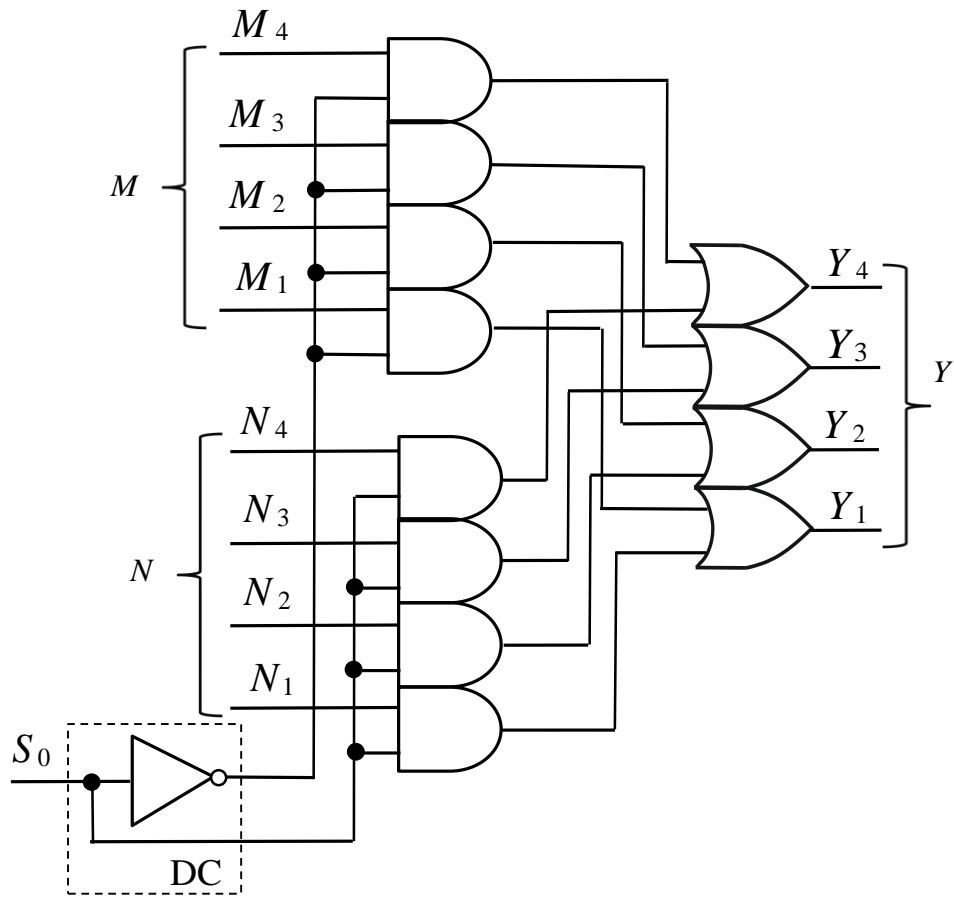


Рис. 5.30. Мультиплексор MUX(4 – 1)



Комбінаційна схема



(DC з прямими входами)

Рис. 5.31. Чотирирозрядний MUX(2 – 1) на основі дешифратора з прямими виходами

Мультиплексор є функціональним вузлом, що може бути використаний для об'єднання окремих комбінаційних схем при декомпозиції функції.

Наприклад, якщо виконують синтез комбінаційної схеми перемикальної функції від 5-ти змінних, то MUX(2 – 1) можна використати при декомпозиції (розкладанні) функції за однією змінною. Задача при цьому зводиться до синтезу двох комбінаційних схем, кожна з яких реалізує функцію від 4-х змінних, а MUX(2 – 1) забезпечує об'єднання двох схем в єдину комбінаційну схему, що реалізує в цілому функцію від 5-ти змінних.

Під час синтезу комбінаційної схеми перемикальної функції, наприклад, від 6-ти змінних зручно використовувати MUX(4 – 1). При цьому задача зводиться до розкладання функції за двома змінними, синтезу 4-х комбінаційних схем, кожна з яких реалізує окрему перемикальну функцію від чотирьох змінних, та їх об'єднання за допомогою MUX(4 – 1).

Розглянемо простіший випадок – синтез комбінаційної схеми перемикальної функції від трьох змінних з використанням MUX(2 – 1). Використання MUX(2 – 1) для реалізації перемикальної функції ґрунтується на розкладанні функції за однією змінною.

Розкладемо перемикальну функцію y від трьох змінних за змінною x_3 відповідно до теореми Шеннона:

$$\begin{aligned} y(x_3, x_2, x_1) &= \bar{x}_3 y(0, x_2, x_1) \vee x_3 y(1, x_2, x_1) = \\ &= \bar{x}_3 y_0(x_2, x_1) \vee x_3 y_1(x_2, x_1) = \bar{x}_3 y_0 \vee x_3 y_1, \end{aligned}$$

де y_0, y_1 – функції від двох змінних – x_2 та x_1 .

Такому розкладанню відповідає структура комбінаційної схеми на рис. 5.32а, де КС(y_0) – комбінаційна схема, що реалізує функцію $y_0(x_2, x_1)$, а КС(y_1) – комбінаційна схема, що реалізує функцію $y_1(x_2, x_1)$. Змінну x_3 використовують для керування мультиплексором: якщо $x_3 = 0$, то на виході мультиплексора маємо $y = y_0$, а якщо $x_3 = 1$, то $y = y_1$.

Отже, MUX(2–1) реалізує перетворення:

$$y = (x_3 = 0) \& y_0 \vee (x_3 = 1) \& y_1 = \bar{x}_3 y_0 \vee x_3 y_1.$$

При розкладанні перемикальної функції $y(x_3, x_2, x_1)$ за змінною x_3 діаграма Вейча функції y розпадається на дві самостійні діаграми Вейча від двох змінних, що відповідають функціям $y_0(x_2, x_1)$ та $y_1(x_2, x_1)$ (рис. 5.33а).

Розкладанню перемикальної функції $y(x_3, x_2, x_1)$ від трьох змінних за змінною x_2

$$\begin{aligned} y(x_3, x_2, x_1) &= \bar{x}_2 y(x_3, 0, x_1) \vee x_2 y(x_3, 1, x_1) = \\ &= \bar{x}_2 \delta_0(x_3, x_1) \vee x_2 \delta_1(x_3, x_1) = \bar{x}_2 \delta_0 \vee x_2 \delta_1 \end{aligned}$$

відповідає структура комбінаційної схеми на рис. 5.32б, де КС(δ_0) – комбінаційна схема, що реалізує функцію від двох змінних $\delta_0(x_3, x_1)$,

КС(δ_1) – комбінаційна схема, що реалізує $\delta_1(x_3, x_1)$, а керування мультиплексором здійснюється змінною x_2 . При цьому діаграма Вейча функції y розпадається на дві самостійні діаграми Вейча від двох змінних, показаних на рис. 5.33б.

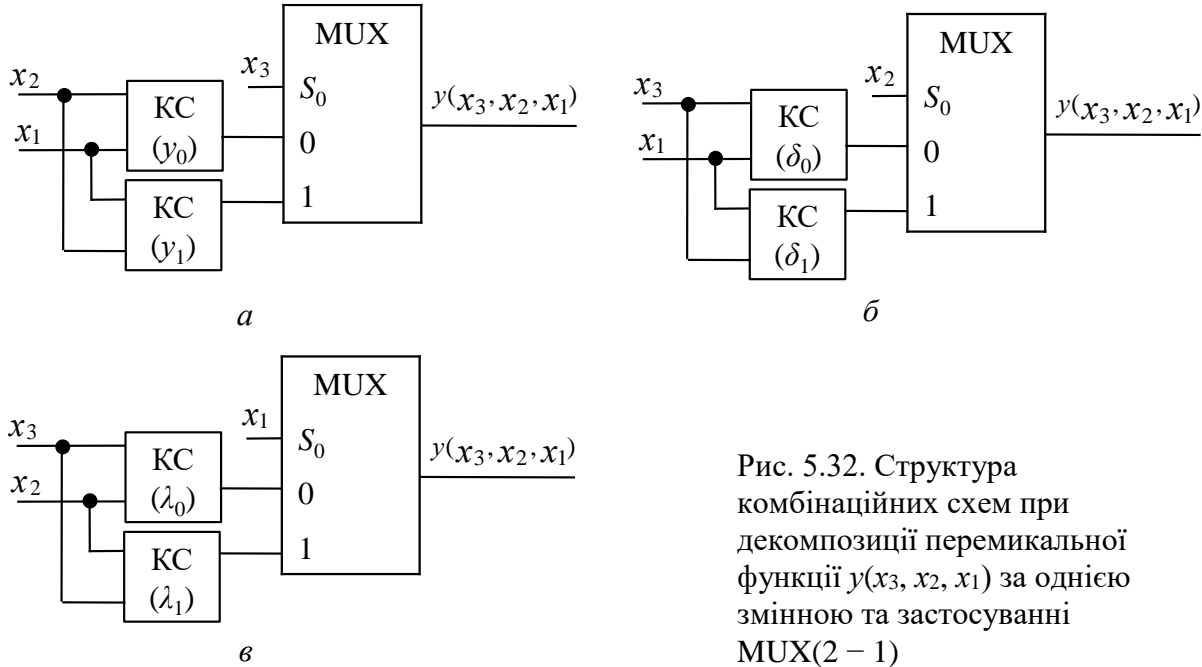


Рис. 5.32. Структура комбінаційних схем при декомпозиції перемикальної функції $y(x_3, x_2, x_1)$ за однією змінною та застосуванні MUX(2 – 1)

За аналогією розкладанню функції $y(x_3, x_2, x_1)$ за змінною x_1

$$\begin{aligned} y(x_3, x_2, x_1) &= \bar{x}_1 y(x_3, x_2, 0) \vee x_1 y(x_3, x_2, 1) = \\ &= \bar{x}_1 \lambda_0(x_3, x_2) \vee x_1 \lambda_1(x_3, x_2) = \bar{x}_1 \lambda_0 \vee x_1 \lambda_1 \end{aligned}$$

відповідає структура комбінаційної схеми на рис. 5.32в, а керування MUX(2 – 1) здійснюється змінною x_1 . При цьому діаграма Вейча функції y також розпадається на дві самостійні діаграми, що відповідають функціям від двох змінних λ_0 та λ_1 (рис. 5.33в).

Задача 5.6. Синтезувати комбінаційну схему перемикальної функції від трьох змінних $y(x_3, x_2, x_1)$, яка дорівнює одиниці на наборах 1, 2, 5, 7 та нулю – на решті наборів, з використанням мультиплексора MUX(2 – 1).

Розв’язування. Діаграма Вейча перемикальної функції y розпадається на дві самостійні діаграми залежно від того, за якою змінною виконують розкладання функції (рис. 5.34).

При розкладанні функції y за змінною x_3 отримуємо дві функції y_0 та y_1 від двох змінних – x_2 та x_1 :

$$\begin{cases} y_0 = x_2 \bar{x}_1 \vee \bar{x}_2 x_1, \\ y_1 = x_1. \end{cases}$$

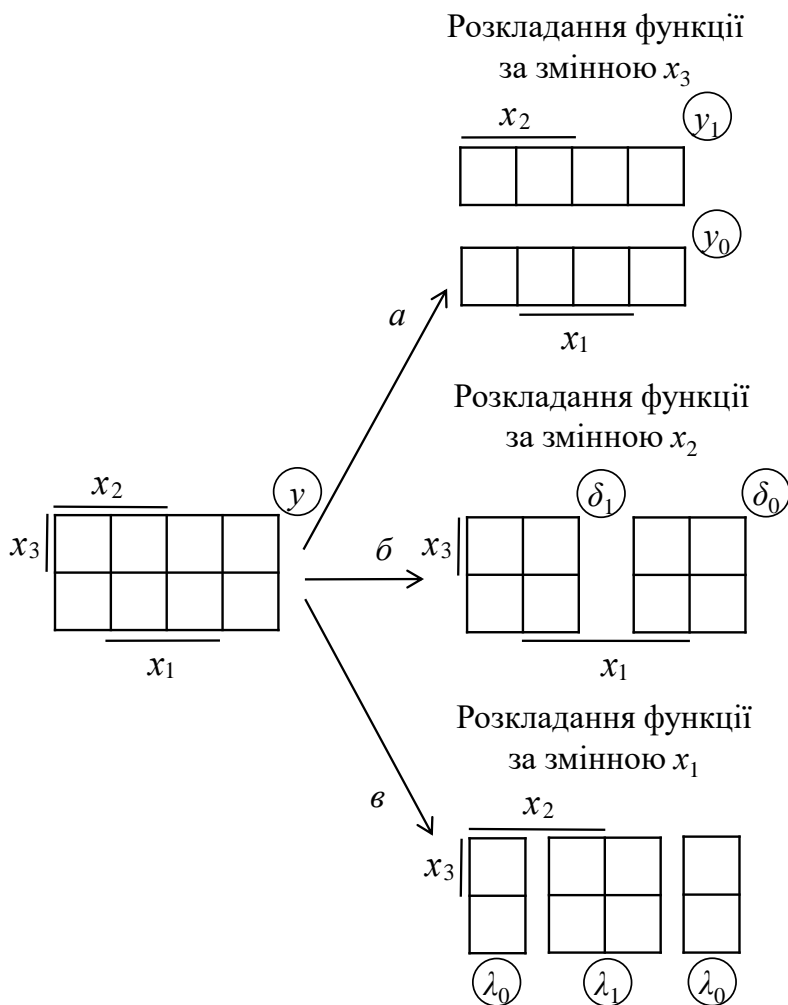


Рис. 5.33. Розпаданя діаграми Вейча перемикальної функції від трьох змінних на дві незалежні діаграми

Отриману систему реалізує комбінаційна схема на рис. 5.35а. Без урахування мультиплексора її складність за Квайном становить $K_1 = 6$, а швидкодія – $t = 2\tau$, де τ – затримка сигналу в одному логічному елементі.

При розкладанні функції y за змінною x_2 отримуємо дві функції δ_0 та δ_1 від двох змінних – x_3 та x_1 (рис. 5.34б):

$$\begin{cases} \delta_0 = x_1, \\ \delta_1 = x_3 x_1 \vee \bar{x}_3 \bar{x}_1. \end{cases}$$

Цій системі функцій відповідає комбінаційна схема на рис. 5.35б, складність якої становить $K_2 = 6$, а швидкодія – $t_2 = 2\tau$.

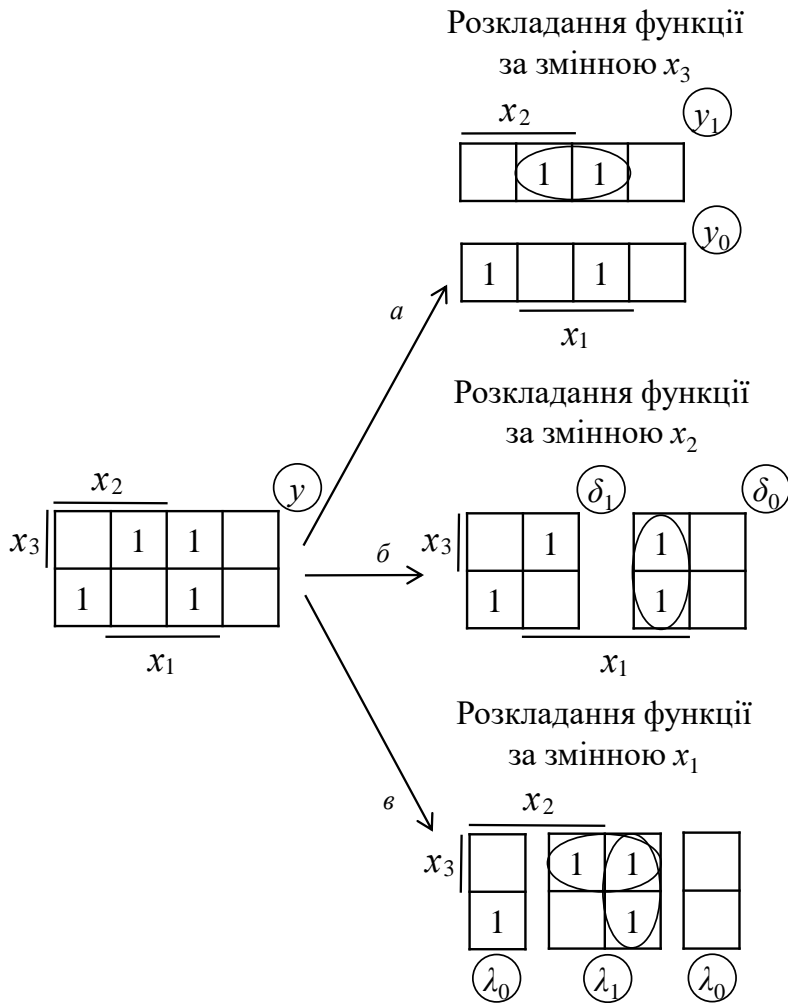


Рис. 5.34. Розпадання діаграми Вейча перемикальної функції y на дві самостійні діаграми Вейча перемикальних функцій від 2-х змінних

Якщо функцію y розкласти за змінною x_1 (рис. 5.34в), то отримаємо систему:

$$\begin{cases} \lambda_0 = \bar{x}_3 x_2, \\ \lambda_1 = x_3 \vee \bar{x}_2, \end{cases}$$

яку реалізує комбінаційна схема на рис. 5.35в. Її складність становить $K_3 = 4$, а швидкодія – $t_3 = \tau$.

За показниками K та t перевагу слід віддати комбінаційній схемі на рис. 5.35в. ■

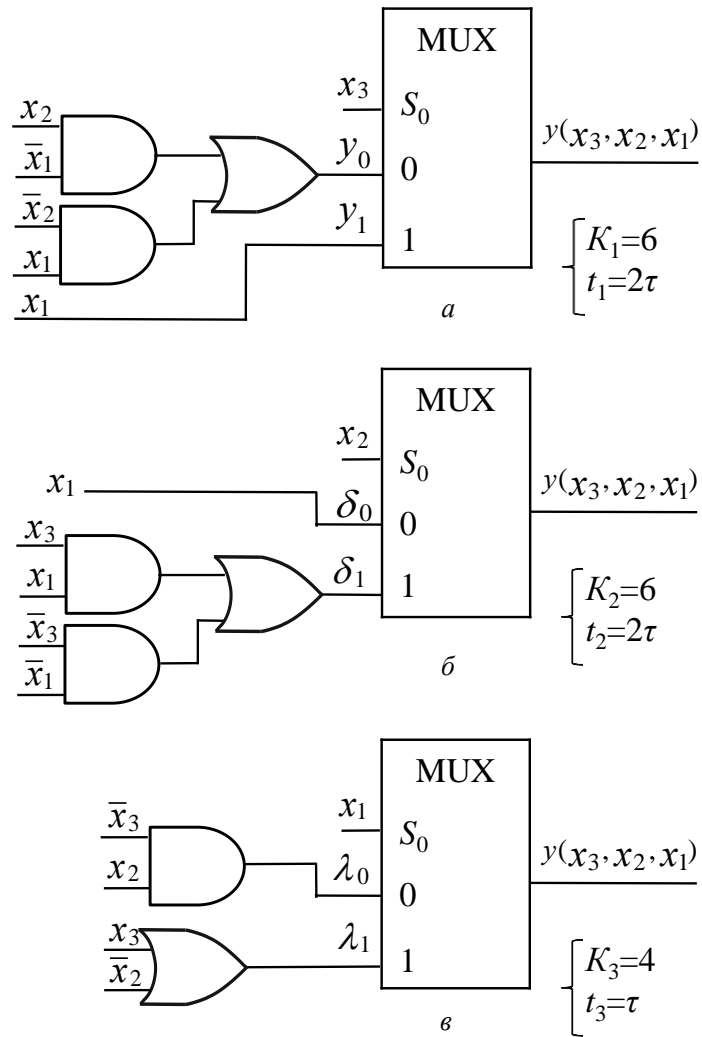


Рис. 5.35. Реалізація перемикальної функції від трьох змінних з використанням MUX(2 – 1)

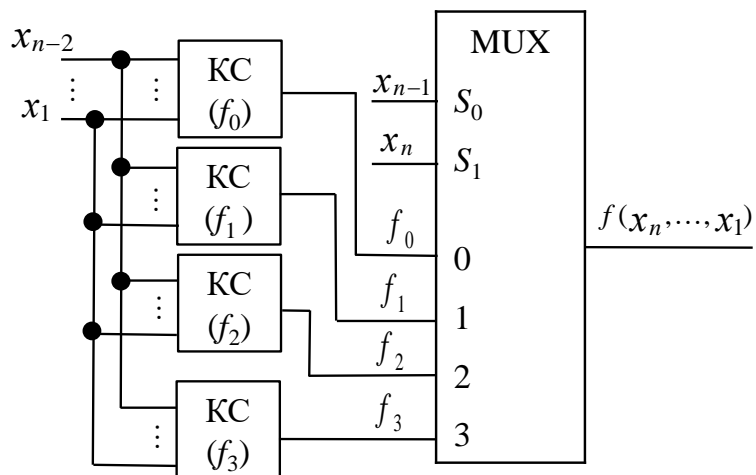


Рис. 5.36. Структура комбінаційної схеми, що реалізує функцію від n змінних, на основі MUX(4 – 1)

Як бачимо, для побудови оптимальної комбінаційної схеми необхідно розглянути всі можливі випадки розкладання функції y за однією змінною.

Під час синтезу комбінаційної схеми часто застосовують декомпозицію перемикальної функції за двома змінними.

Наприклад, розкладання n -місної перемикальної функції за двома старшими змінними відповідно до теореми Шеннона виглядає так:

$$\begin{aligned} f(x_n, \dots, x_1) &= \bar{x}_n \bar{x}_{n-1} f(0,0, x_{n-2}, \dots, x_1) \vee \bar{x}_n x_{n-1} f(0,1, x_{n-2}, \dots, x_1) \vee \\ &\vee x_n \bar{x}_{n-1} f(1,0, x_{n-2}, \dots, x_1) \vee x_n x_{n-1} f(1,1, x_{n-2}, \dots, x_1) = \bar{x}_n \bar{x}_{n-1} f_0(x_{n-2}, \dots, x_1) \vee \\ &\vee \bar{x}_n x_{n-1} f_1(x_{n-2}, \dots, x_1) \vee x_n \bar{x}_{n-1} f_2(x_{n-2}, \dots, x_1) \vee x_n x_{n-1} f_3(x_{n-2}, \dots, x_1) = \\ &= \bar{x}_n \bar{x}_{n-1} f_0 \vee \bar{x}_n x_{n-1} f_1 \vee x_n \bar{x}_{n-1} f_2 \vee x_n x_{n-1} f_3, \end{aligned}$$

де $f_0 - f_3$ – функції, що залежать лише від $n - 2$ змінних.

Тоді функцію $f(x_n, \dots, x_1)$ реалізують з використанням мультиплексо-ра MUX(4 – 1) (рис. 5.36). Кожній з функцій $f_0 - f_3$, що залежать лише від $n - 2$ змінних відповідає комбінаційна схема (КС(f_0) – (КС(f_3)).

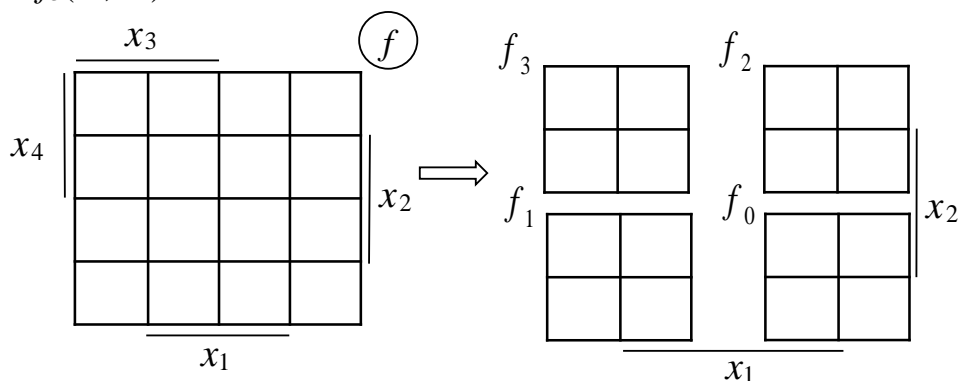
Мультиплексор, що має 4 інформаційні входи та два входи керування (S_1, S_0), забезпечує передачу (комутацію) на вихід f одного з сигналів, що надходять на інформаційні входи, залежно від комбінації керуючих сигналів:

S_1	S_0	Вихід f	Розглянемо синтез комбінаційної схеми перемикальної функції від чотирьох змінних ($n = 4$) з використанням MUX(4 – 1).
0	0	$f = f_0$	
0	1	$f = f_1$	
1	0	$f = f_2$	Розкладемо перемикальну функцію f від 4-х змінних за двома старшими змінними x_4 та x_3 відповідно до теореми Шеннона:
1	1	$f = f_3$	

$$f(x_4, x_3, x_2, x_1) = \bar{x}_4 \bar{x}_3 f_0 \vee \bar{x}_4 x_3 f_1 \vee x_4 \bar{x}_3 f_2 \vee x_4 x_3 f_3,$$

де $f_0 - f_3$ – функції від двох змінних – x_2 та x_1 .

При розкладанні перемикальної функції (п.ф.) $f(x_4, x_3, x_2, x_1)$ за змінними x_4, x_3 діаграма Вейча функції f розпадається на чотири самостійні діаграми Вейча від 2-х змінних, що відповідають функціям $f_0(x_2, x_1) - f_3(x_2, x_1)$:



Але функцію $f(x_4, x_3, x_2, x_1)$ можна розкласти також за змінними $x_4, x_2; x_4, x_1; x_3, x_2; x_3, x_1; x_2, x_1$, тобто загалом існує 6 варіантів розкладання функції від 4-х змінних за двома змінними, і, відповідно, – 6 варіантів розпаданя діаграми Вейча функції від 4-х змінних на 4 самостійні діаграми функцій від 2-х змінних (рис. 5.37).

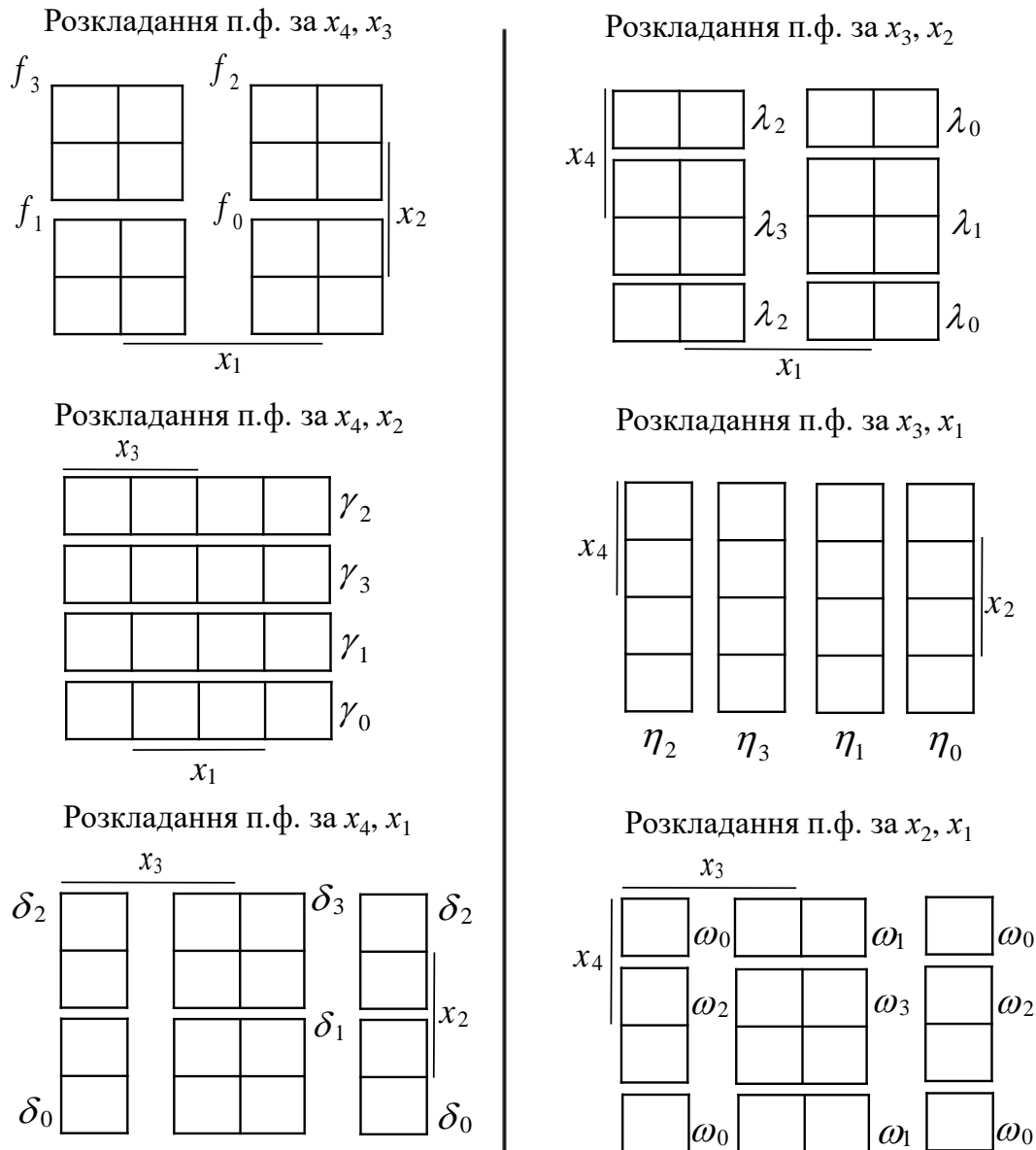


Рис. 5.37. Варіанти розпаданя діаграми Вейча перемикальної функції від 4-х змінних на 4 самостійні діаграми Вейча перемикальних функцій від 2-х змінних

Для кожного розкладання функції необхідно побудувати комбінаційну схему та визначити її складність і швидкодію.

Результатом синтезу комбінаційної схеми перемикальної функції від 4-х змінних з використанням мультиплексора з двома входами керування є комбінаційна схема, що має найкращий показник апаратної складності і/або показник швидкодії – залежно від цільової функції проектування.

Задача 5.7. Синтезувати комбінаційну схему перемикальної функції $f(x_4, x_3, x_2, x_1)$ від 4-х змінних, що дорівнює одиниці на наборах 2, 3, 5, 8, 9, 12, 14, 15 та нулю – на решті наборів, з використанням мультиплексора з двома входами керування.

Розв'язування. Необхідно виконати 6 варіантів розкладання перемикальної функції за змінними $x_4, x_3; x_4, x_2; x_4, x_1; x_3, x_2; x_3, x_1; x_2, x_1$, та побудувати 6 окремих комбінаційних схем на основі MUX(4 – 1), а серед них – вибрати кращу за показником апаратної складності та/або показником швидкодії.

Розкладемо функцію $f(x_4, x_3, x_2, x_1)$ за змінними x_4, x_3 (рис. 5.38). У цьому випадку діаграма Вейча функції розпадається на чотири самостійні діаграми функцій $f_0 - f_3$ від двох змінних. Для кожної з функцій $f_0 - f_3$ запишемо аналітичний вираз.

Тоді функції $f(x_4, x_3, x_2, x_1)$ від 4-х змінних відповідатиме система функцій від 2-х змінних:

$$\left\{ \begin{array}{l} f_3 = \bar{x}_1 \vee x_2, \\ f_2 = \bar{x}_2, \\ f_1 = \bar{x}_2 x_1, \\ f_0 = x_2. \end{array} \right. \begin{array}{l} \text{Дану систему реалізує комбінаційна схема на} \\ \text{основі мультиплексора з двома входами керування} \\ \text{(рис. 5.38), на які подають змінні, за якими виконано} \\ \text{розкладання функції } f, \text{ тобто } x_4 \text{ та } x_3. \\ \text{Складність за Квайном такої комбінаційної схеми} \\ \text{без урахування мультиплексора становить } K_1 = 4, \text{ а} \\ \text{швидкодія } - t_1 = \tau. \end{array}$$

Виконуючи розкладання функції $f(x_4, x_3, x_2, x_1)$ за іншими 5-ма варіантами комбінацій змінних (рис. 5.39) та оцінюючи складність і швидкодію відповідних комбінаційних схем доходимо висновку, що найкращий показник складності (K) та швидкодії (t) мають комбінаційні схеми за варіантом 1 та варіантом 2 розкладання. Для обох варіантів $K = 4$, а $t = \tau$ (без урахування мультиплексора). Будь-який з них можна обрати як розв'язок задачі.

Як розв'язок задачі оберемо комбінаційну схему за варіантом 2.

Отже, перемикальну функцію $f(x_4, x_3, x_2, x_1)$ реалізує комбінаційна схема на рис. 5.40. ■

Варіант 1. Розкладання п.ф. за змінними x_4, x_3

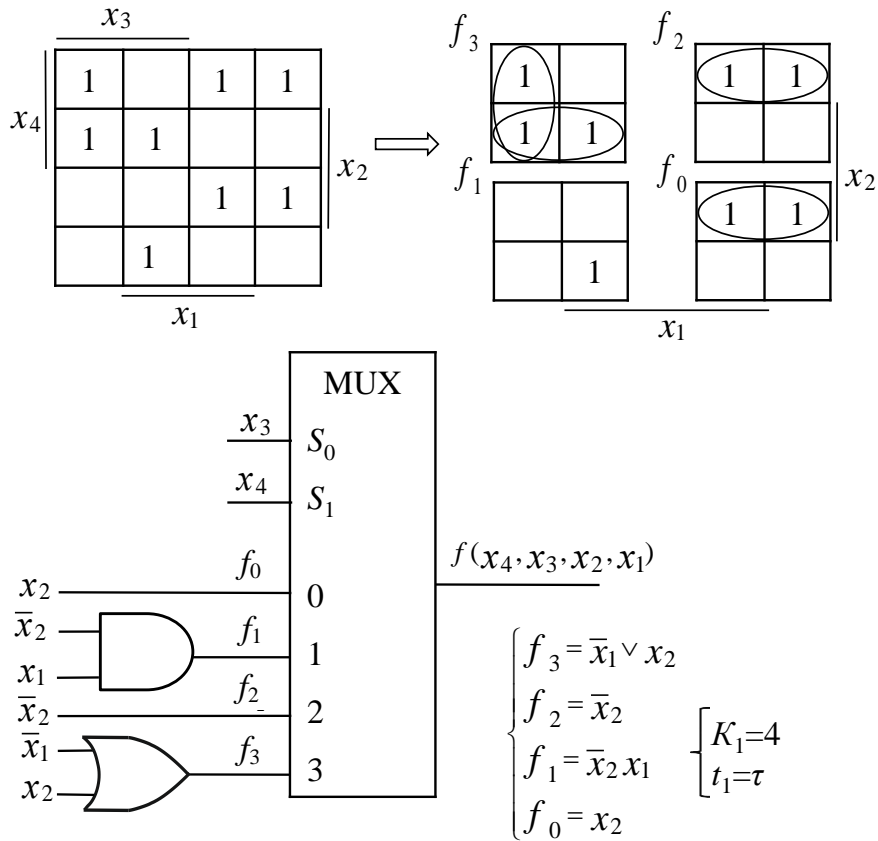


Рис. 5.38. Комбінаційна схема, що реалізує функцію $f(x_4, x_3, x_2, x_1)$ на основі MUX(4 – 1) при розкладанні функції за змінними x_4, x_3

Варіант 2. Розкладання п.ф. за змінними x_4, x_2

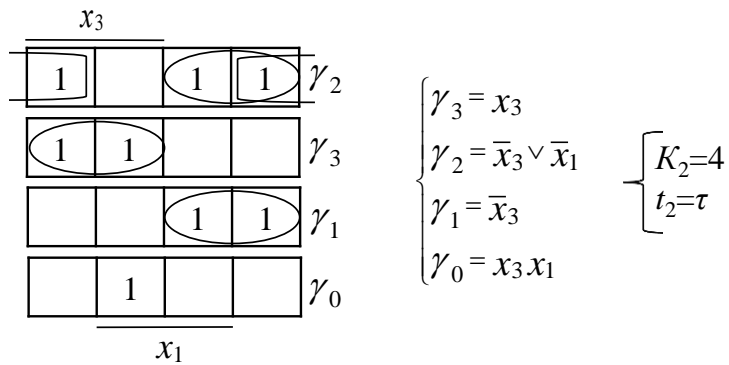
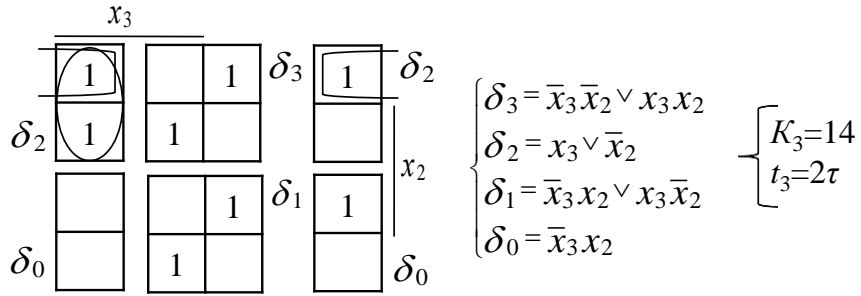
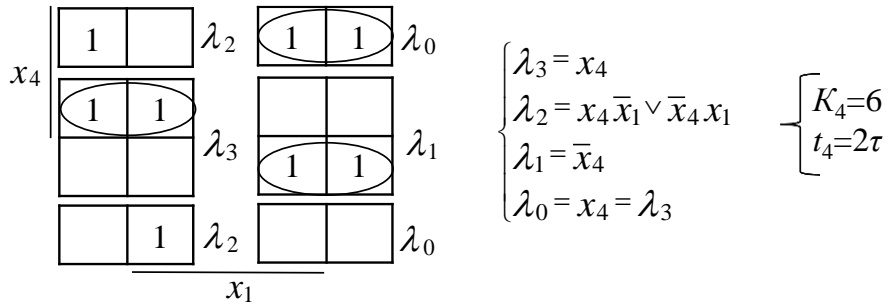


Рис. 5.39. Варіанти розкладання функції $f(x_4, x_3, x_2, x_1)$ за двома змінними

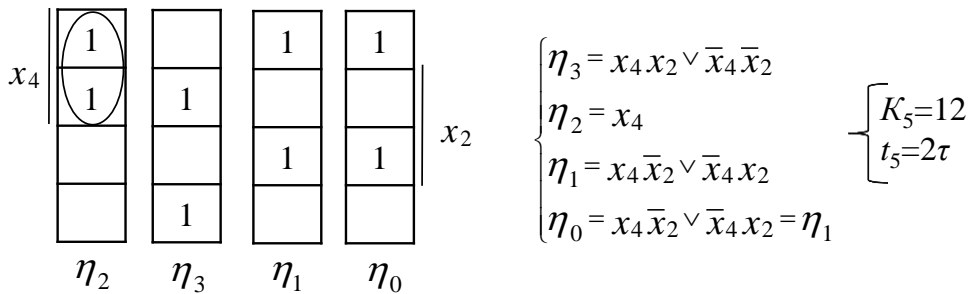
Варіант 3. Розкладання п.ф. за змінними x_4, x_1



Варіант 4. Розкладання п.ф. за змінними x_3, x_2



Варіант 5. Розкладання п.ф. за змінними x_3, x_1



Варіант 6. Розкладання п.ф. за змінними x_2, x_1

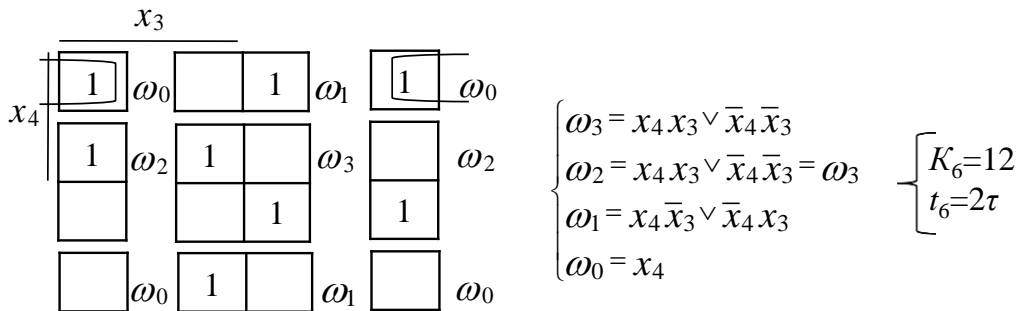


Рис. 5.39. Варіанти розкладання функції $f(x_4, x_3, x_2, x_1)$ за двома змінними (продовження)

КС, синтезована на основі мультиплексора із застосуванням декомпозиції перемикальної функції, у загальному випадку може поступатись за складністю еквівалентній КС, синтезованій за класичним підходом з використанням лише логічних елементів.

Існує взаємно однозначна відповідність між входами D_i мультиплексора та розташуванням конститuent одиниці на карті Карно (діаграмі Вейча).

Наприклад, на рис. 5.41 показано відповідність між картою Карно перемикальної функції z від двох змінних та MUX(4 – 1).

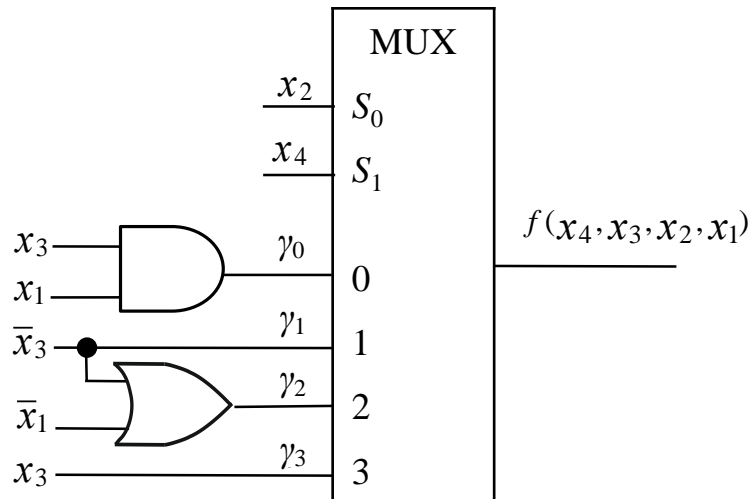


Рис. 5.40. Комбінаційна схема, що є розв'язком задачі 5.7

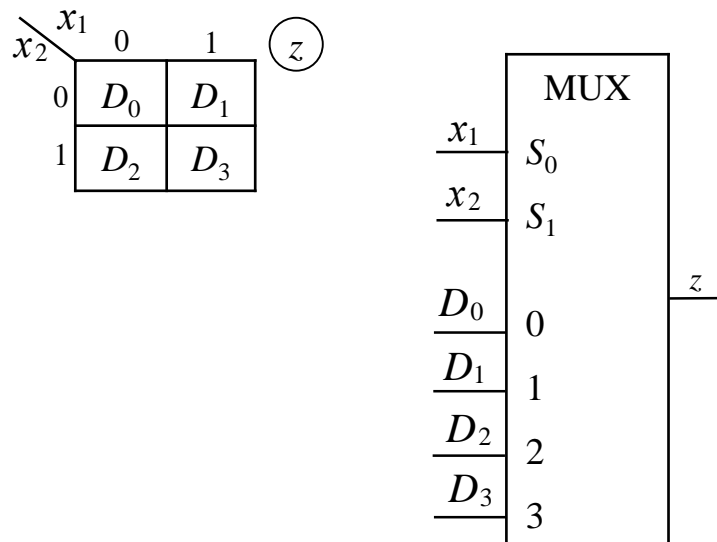


Рис. 5.41. Відповідність між картою Карно та MUX(4 – 1)

Розташування конститuentи одиниці $\bar{x}_2 \bar{x}_1$ відповідає входу D_0 мультиплексора, розташування конститuentи одиниці $\bar{x}_2 x_1$ – входу D_1 ,

конституенти одиниці $x_2\bar{x}_1$ – входу D_2 , а конституенти одиниці x_2x_1 – входу D_3 .

Карта Карно на рис. 5.41 відповідає перемикальній функції

$$z = \bar{x}_2\bar{x}_1D_0 \vee \bar{x}_2x_1D_1 \vee x_2\bar{x}_1D_2 \vee x_2x_1D_3.$$

Саме відповідність між інформаційними входами мультиплексора та розташуванням конституент одиниці на карті Карно (діаграмі Вейча) дає можливість використовувати мультиплексор для відтворення деяких співвідношень алгебри логіки.

Наприклад, MUX(4 – 1) на рис. 5.42 реалізує операцію XOR, а саме $y = x_2 \oplus x_1$.

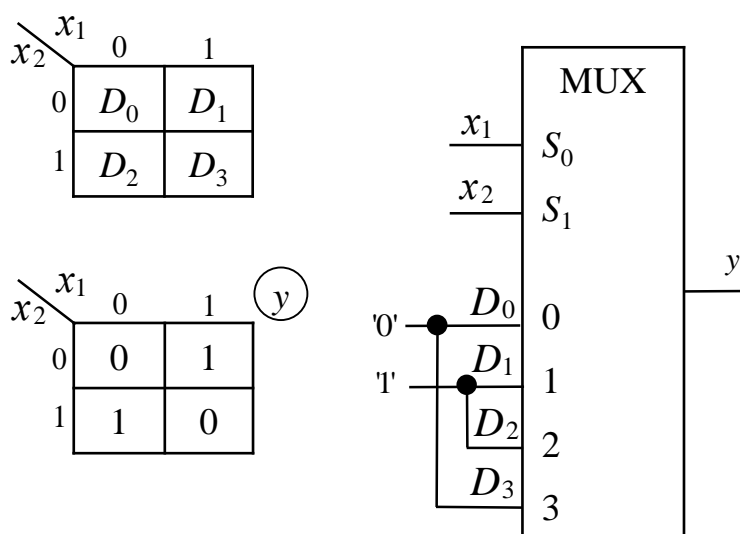


Рис. 5.42. Реалізація операції $y = x_2 \oplus x_1$ на основі MUX(4 – 1)

Запитання та завдання

1. Дайте визначення мультиплексора.
2. Як здійснюють керування мультиплексором?
3. Скільки входів керування має мультиплексор з 16-ма інформаційними входами?
4. Які мультиплексори називають однорозрядними, а які – багаторозрядними?
5. Наведіть умовне графічне позначення:
 - а) однорозрядного MUX(8 – 1);
 - б) 4-розрядного MUX(4 – 1).
6. Побудуйте таблицю функціонування:
 - а) однорозрядного MUX(8 – 1);
 - б) 4-розрядного MUX(4 – 1).
7. Поясніть, як використовують мультиплексори під час синтезу комбінаційної схеми із застосуванням декомпозиції перемикальної функції.

Задачі для самостійного розв'язування

1. Побудувати комбінаційну схему мультиплектора $MUX(8 - 1)$, використовуючи для вибору інформаційних напрямів дешифратор з прямими виходами.
2. Побудувати комбінаційну схему мультиплектора $MUX(8 - 1)$, використовуючи для вибору інформаційних напрямів дешифратор з інверсними виходами.
3. Побудувати 8-розрядний $MUX(2 - 1)$, використовуючи для вибору інформаційних напрямів дешифратор з інверсними виходами.
4. В елементному базисі: однорозрядний $MUX(2 - 1)$, логічні елементи 2І-НЕ – синтезувати оптимізовану за показником апаратної складності комбінаційну схему, яка реалізує перемикальну функцію від 3-х змінних, що дорівнює одиниці на наборах 1, 3, 4, 7 і нулю – на решті наборів.
5. В елементному базисі: однорозрядний $MUX(4 - 1)$, логічні елементи 2АБО-НЕ – синтезувати оптимізовану за показником апаратної складності комбінаційну схему, яка реалізує перемикальну функцію від 4-х змінних, що дорівнює одиниці на наборах 0, 2, 3, 4, 5, 9, 14, 15 і нулю – на решті наборів.
6. На основі $MUX(4 - 1)$ реалізувати логічну операцію: а) $f = x_2 \vee x_1$; б) $f = x_2 \cdot x_1$.

5.5. Демультимплектори

Демультимплексор (demultiplexor, DMX) – це функціональний вузол (комбінаційна схема), призначений для комутації сигналів з одного вхідного напрямку на декілька вихідних напрямів.

На схемі функціонування демультимплексора (рис. 5.43) d_0, d_1, \dots, d_{n-1} – вихідні напрями, а x – вхідний напрям. Код на входах керування S визначає, на який саме вихідний напрям буде скомутовано сигнал з вхідного напрямку x .

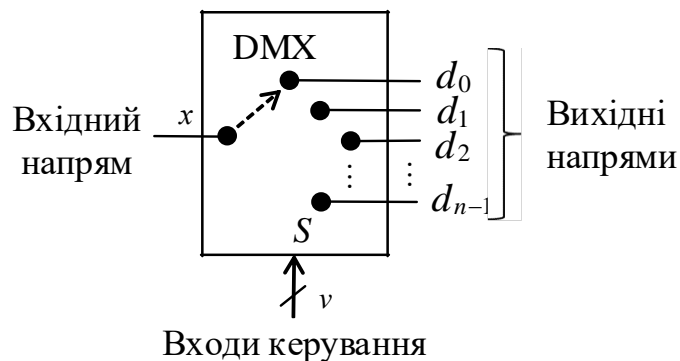


Рис. 5.43. Схема функціонування демультимплексора

Узагальнено демультиплексор позначають $DMX(1 - n)$, де n – кількість вихідних напрямів.

Наприклад, $DMX(1 - 2)$ – демультиплексор, що комутує інформаційні сигнали з одного вхідного напрямку на два вихідні напрями; $DMX(1 - 4)$ – з одного вхідного напрямку на 4 вихідні напрями. Зазвичай $n = 2, 4, 8, 16, \dots$

Якщо DMX має n вихідних напрямів, то кількість входів керування має становити $v = \lceil \log_2 n \rceil$; v -розрядний код, встановлений на входах керування S , вказує, на який саме вихід буде скомутовано сигнал з вхідного напрямку x .

Демультиплексор, зображений на рис. 5.44, забезпечує комутацію сигналів з вхідного напрямку x на два вихідні напрями – d_0 та d_1 , відповідно до таблиці функціонування. Якщо $S_0 = 0$, то сигнали з входу x надходитимуть на вихід d_0 , а якщо $S_0 = 1$, то – на вихід d_1 .

У комбінаційній схемі може використовуватись дешифратор з прямими або інверсними виходами.

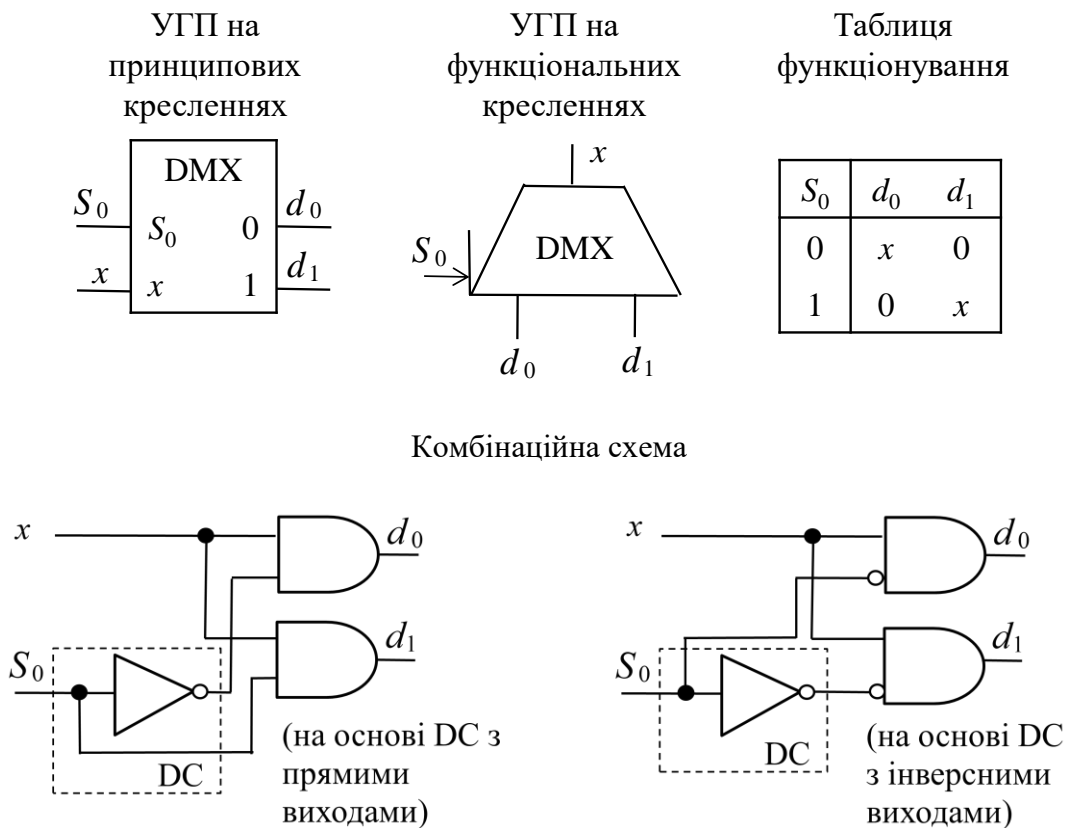


Рис. 5.44. Демультиплексор $DMX(1 - 2)$

Демультимплексор DMX(1 – 4) (рис. 5.45) забезпечує комутацію сигналів з вхідного напрямку x на 4 вихідні напрями d_0, d_1, d_2, d_3 відповідно до таблиці функціонування демультимплексора.

Наприклад, якщо $S_1 = 0$, а $S_0 = 1$, то сигнали з входу x надходять на вихід d_1 .

На рис. 5.45 подано два варіанти комбінаційної схеми – з використанням дешифратора з прямими та інверсними виходами.

Розрядність демультимплексора визначають кількістю ліній, що входять до складу вхідного напрямку x (і, відповідно, – до складу кожного з вихідних напрямів). Демультимплексори на рис. 5.44, 5.45 є однорозрядними.

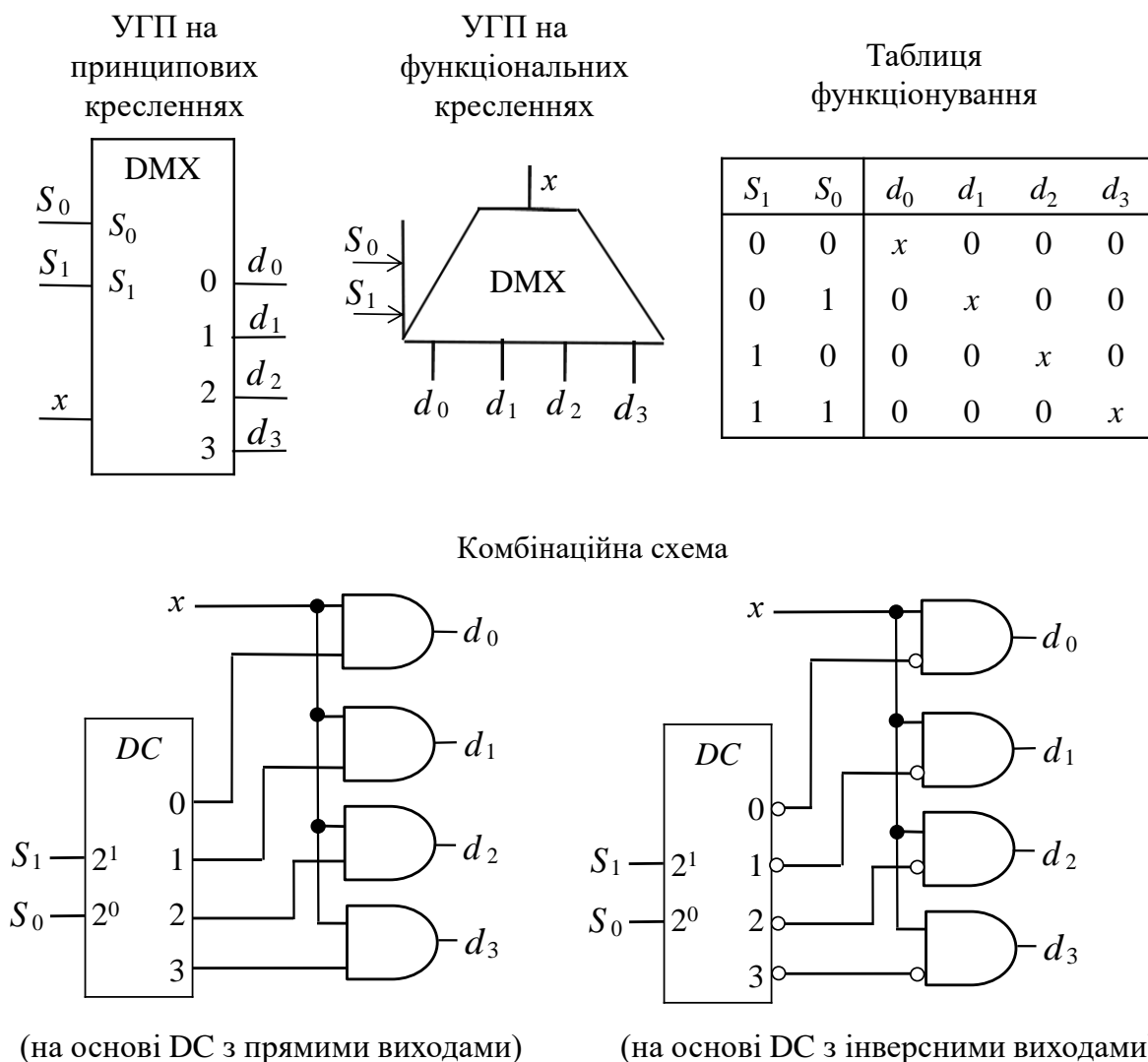


Рис. 5.45. Демультимплексор DMX(1 – 4)

Крім однорозрядних, в обчислювальних структурах можуть використовуватись багаторозрядні демультиплексори.

У 4-розрядному $DMX(1-2)$ (на рис. 5.46) до складу вхідного напрямку X входять 4 лінії – $X_4X_3X_2X_1$, а до складу вихідних напрямів V та W – $V_4V_3V_2V_1$ та $W_4W_3W_2W_1$ відповідно.

Якщо $S_0 = 0$, то сигнали з 4-х ліній вхідного напрямку X комутуються на відповідні 4 лінії напрямку V , а якщо $S_0 = 1$, то – напрямку W .

Комбінаційна схема 4-розрядного $DMX(1-2)$ може бути побудована з використанням дешифратора як з прямими виходами (рис. 5.46), так і з інверсними виходами.

Як й мультиплексор, демультиплексор також часто називають *комутатором*.

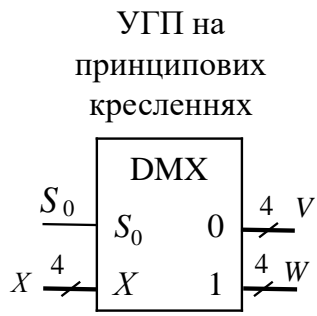
Демультиплексор виконує функцію обернену до функції мультиплексора.

Запитання та завдання

1. Дайте визначення демультиплексора.
2. Поясніть, як функціонує демультиплексор.
3. Скільки входів керування має демультиплексор з 16-ма виходами?
4. Чим відрізняються однорозрядні демультиплексори від багаторозрядних?
5. Наведіть умовне графічне позначення: а) однорозрядного $DMX(1-8)$; б) 4-розрядного $DMX(1-4)$.
6. Побудуйте таблицю функціонування:
а) однорозрядного $DMX(1-8)$; б) 4-розрядного $DMX(1-4)$.
7. Порівняйте між собою мультиплексори та демультиплексори. Що є відмінним та спільним для них?

Задачі для самостійного розв'язування

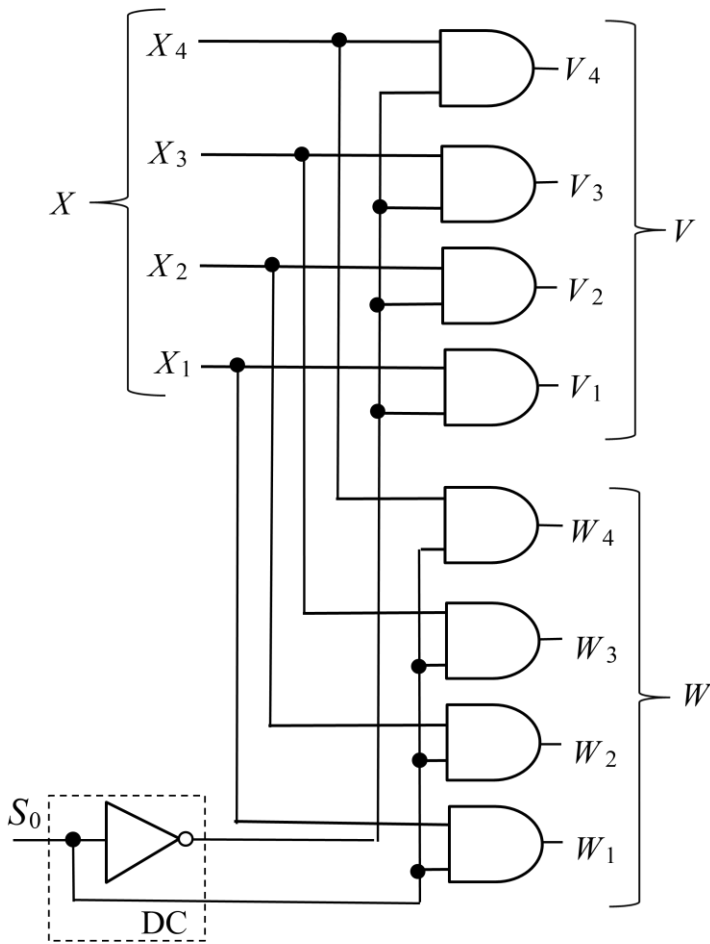
1. Побудувати комбінаційну схему однорозрядного демультиплексора $DMX(1-8)$, використовуючи для вибору вихідного напрямку дешифратор з прямими виходами.
2. Побудувати 8-розрядний $DMX(1-2)$, використовуючи для вибору вихідного напрямку дешифратор з інверсними виходами.
3. Побудувати 2-розрядний $DMX(1-4)$, використовуючи для вибору вихідного напрямку дешифратор з інверсними виходами.



Таблиця функціонування

S_0	V	W
0	X	\emptyset
1	\emptyset	X

Комбінаційна схема



(на основі DC з прямими виходами)

Рис. 5.46. Чотирирозрядний DMX(1 – 2)

5.6. Схеми порівняння кодів

У цифрових обчислювальних засобах можуть використовуватись такі функціональні вузли як схеми порівняння кодів (СПК) або компаратори (від compare – порівнювати), які забезпечують порівняння операндів у ході обчислювального процесу.

Нехай A, B – два n -розрядні слова (коди): $A = (a_n a_{n-1} \dots a_1)$, $B = (b_n b_{n-1} \dots b_1)$, які мають такі кількісні еквіваленти:

$$A_{ке} = \sum_{i=1}^n a_i 2^{i-1}, \quad B_{ке} = \sum_{i=1}^n b_i 2^{i-1}.$$

Між кодами (словами) можливі 6 відношень:

$$A = B, \quad A > B, \quad A < B, \quad (5.14)$$

$$A \neq B, \quad A \leq B, \quad A \geq B. \quad (5.15)$$

Для схемної реалізації 6-ти можливих відношень між кодами (словами) достатньо розглянути лише відношення ряду (5.14), тобто $A = B, A > B, A < B$, оскільки будь-яке відношення ряду (5.15) легко отримати як інверсію відповідного відношення ряду (5.14).

Наприклад, $A \neq B$ – це не що інше, як $\overline{A = B}$, а $A \geq B$ – це $\overline{A < B}$.

Схемою порівняння кодів (слів) A і B (або компаратором) називають комбінаційну схему, що має $2n$ входів та 3 виходи, яка реалізує такі 3 функції (рис. 5.47):

$$y_1(A, B) = \begin{cases} 1, \text{ якщо} & A = B \\ 0, \text{ якщо} & A \neq B \end{cases}$$

$$y_2(A, B) = \begin{cases} 1, \text{ якщо} & A > B \\ 0, \text{ якщо} & A \leq B \end{cases}$$

$$y_3(A, B) = \begin{cases} 1, \text{ якщо} & A < B \\ 0, \text{ якщо} & A \geq B \end{cases}.$$

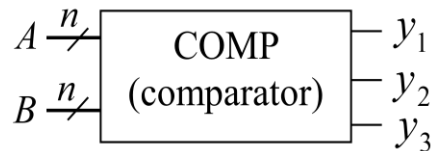


Рис. 5.47. Узагальнене позначення компаратора

Зрозуміло, що $y_1(A, B) = 1$ лише в тому випадку, коли $a_n = b_n, a_{n-1} = b_{n-1}, \dots, a_1 = b_1$, тобто коли всі розряди слова A дорівнюють відповідним розрядам слова B . Для формування відношення $A = B$ необхідно n разів виконати логічну операцію рівнозначності:

$$y_1(A, B) = (a_n = b_n) \& (a_{n-1} = b_{n-1}) \& \dots \& (a_1 = b_1) = \bigg\&_{i=1}^n (a_i = b_i) = \bigg\&_{i=1}^n F_i,$$

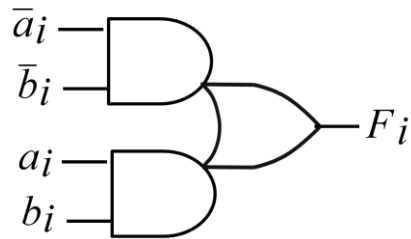
де F_i – функція рівнозначності (табл. 5.16).

Функцію F_i реалізує логічний елемент, що має назву 2(2I)-2АБО (рис. 5.48).

Таблиця 5.16

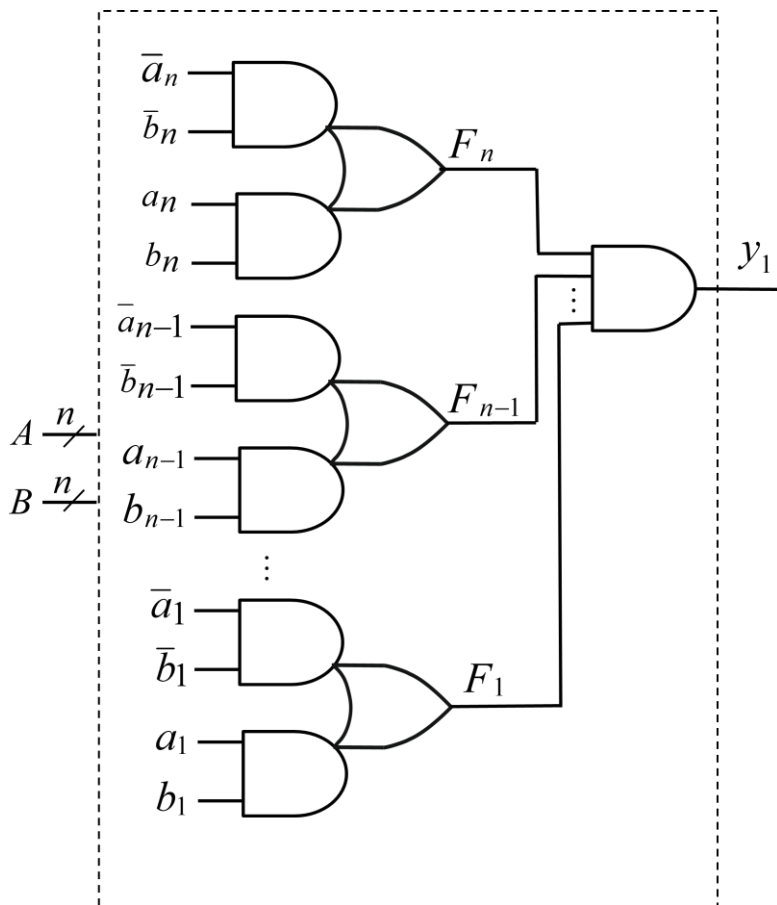
Функції, що відповідають відношенням ряду (5.14) у випадку однорозрядних операндів a_i та b_i

a_i	b_i	$F_i(a_i = b_i)$	$G_i(a_i > b_i)$	$H_i(a_i < b_i)$
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0



2(2I)-2АБО

Рис. 5.48. Апаратна реалізація функції рівнозначності

Рис. 5.49. Комбінаційна схема, що реалізує функцію $y_1(A, B)$

Відповідно, функцію $y_1(A, B)$ реалізує схема на рис. 5.49; $y_1 = 1$, коли $A = B$. Якщо ж $A \neq B$, то $y_1 = 0$.

Функція $y_2 = 1$, якщо $(a_n > b_n)$ або $(a_n = b_n) \& (a_{n-1} > b_{n-1})$ або $(a_n = b_n) \& (a_{n-1} = b_{n-1}) \& (a_{n-2} > b_{n-2})$ або ... або $(a_n = b_n) \& (a_{n-1} = b_{n-1}) \& \dots \& (a_2 = b_2) \& (a_1 > b_1)$.

Беручи до уваги, що функція $G_i(a_i > b_i)$ дорівнює одиниці лише на одному наборі аргументів (табл. 5.16), тобто $G_i = a_i \bar{b}_i$, логічний вираз для функції y_2 запишемо у вигляді:

$a_n \bar{b}_n$ або $F_n \& a_{n-1} \bar{b}_{n-1}$ або $F_n F_{n-1} a_{n-2} \bar{b}_{n-2}$ або ... або $F_n F_{n-1} \dots F_2 a_1 \bar{b}_1$, тобто $y_2 = a_n \bar{b}_n \vee F_n (a_{n-1} \bar{b}_{n-1} \vee F_{n-1} (a_{n-2} \bar{b}_{n-2} \vee F_{n-2} (\dots \vee F_3 (a_2 \bar{b}_2 \vee F_2 a_1 \bar{b}_1) \dots)))$.

За аналогією можна записати логічний вираз для функції y_3 :

$y_3 = \bar{a}_n b_n \vee F_n (\bar{a}_{n-1} b_{n-1} \vee F_{n-1} (\bar{a}_{n-2} b_{n-2} \vee F_{n-2} (\dots \vee F_3 (\bar{a}_2 b_2 \vee F_2 (\bar{a}_1 b_1) \dots)))$.

Але будувати комбінаційну схему, що реалізує функцію $y_3(A, B)$ немає потреби, оскільки $y_3 = \bar{y}_1 \& \bar{y}_2 = \overline{y_1 \vee y_2}$.

Дійсно, $(A < B) = (A = B) \& (A > B) = \overline{(A = B) \vee (A > B)}$.

Відтак, очевидна структура компаратора (рис. 5.50), до складу якої входить СПК, що ідентифікує випадок $A = B$, та СПК, що виявляє випадок $A > B$; логічний елемент 2АБО-НЕ, що формує функцію y_3 , фактично є СПК ($A < B$).

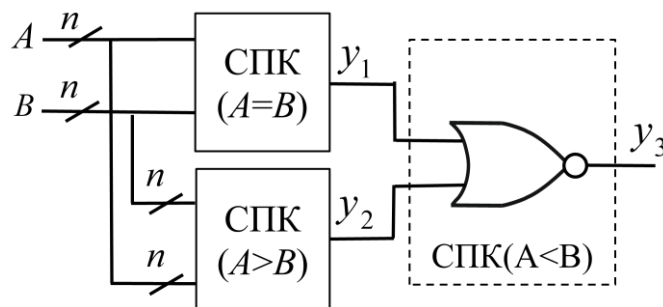


Рис. 5.50. Структура компаратора

Але, за аналогією $y_2 = \bar{y}_1 \& \bar{y}_3 = \overline{y_1 \vee y_3}$.

Дійсно, $(A > B) = (A = B) \& (A < B) = \overline{(A = B) \vee (A < B)}$. Це означає, що в розглянутій на рис. 5.50 структурі компаратора СПК($A > B$) можна замінити на СПК($A < B$), а на виході логічного елемента 2АБО-НЕ матимемо функцію y_2 ; і тоді елемент 2АБО-НЕ фактично реалізуватиме СПК($A > B$).

Отже, для реалізації компаратора, що забезпечує 3 відношення: $A = B$, $A > B$, $A < B$, достатньо мати лише дві схеми порівняння кодів – або СПК($A = B$) і СПК($A > B$), або СПК($A = B$) і СПК($A < B$).

Структурі на рис. 5.50 відповідає комбінаційна схема компаратора на рис. 5.51 та умовне графічне позначення на рис. 5.52а.

Для отримання всіх шести можливих відношень між кодами (словами) на виходах y_1, y_2, y_3 компаратора достатньо додатково розмістити три інвертори (рис. 5.52б).

Компаратор може використовуватись під час апаратної реалізації умовного оператора IF в мовах програмування.

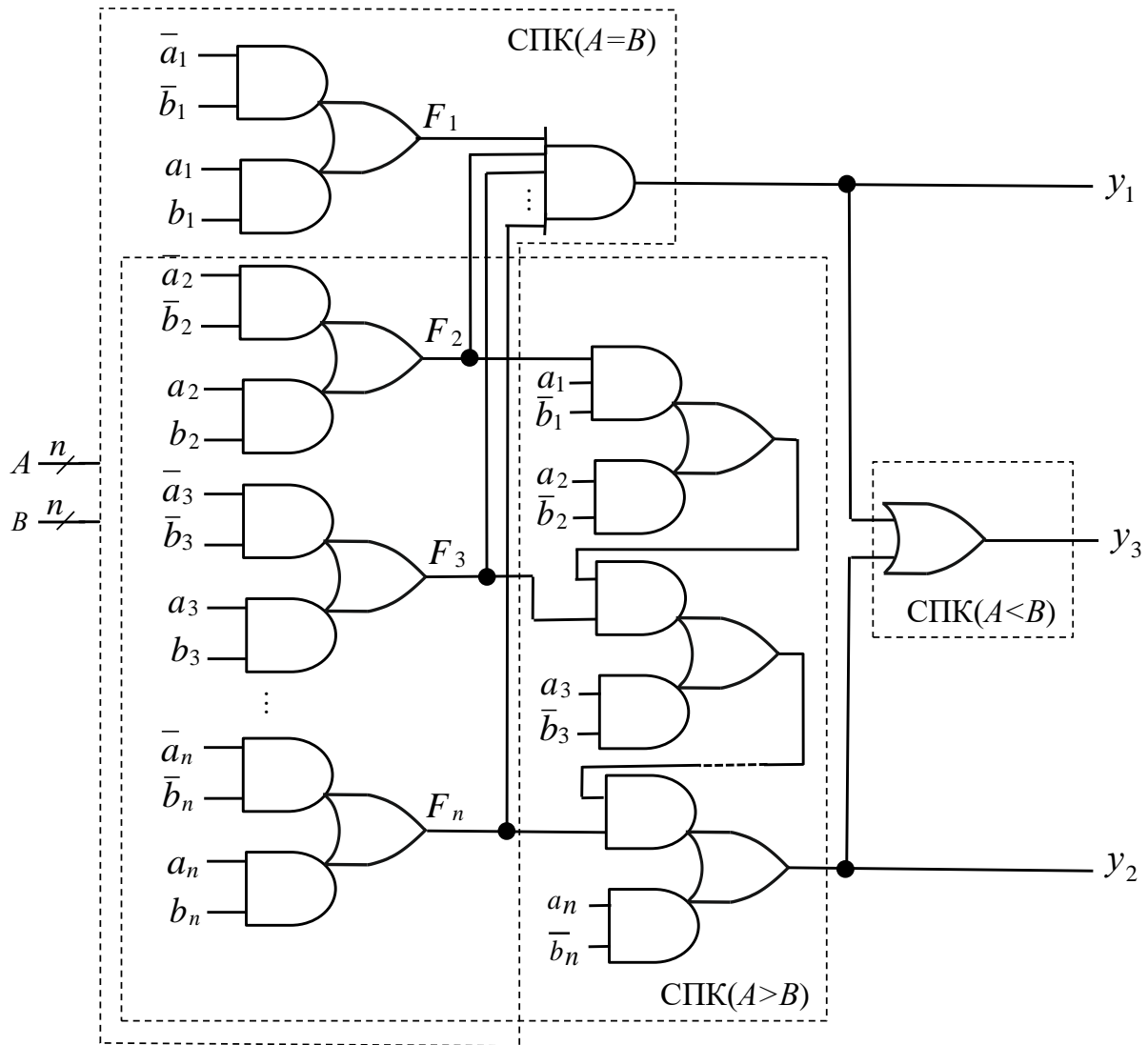


Рис. 5.51. Комбінаційна схема компаратора

Запитання та завдання

1. Яку комбінаційну схему називають схемою порівняння слів (кодів)?
2. Яка мета застосування схем порівняння кодів у цифрових обчислювальних засобах?
3. Назвіть можливі відношення між словами (кодами) A та B ?

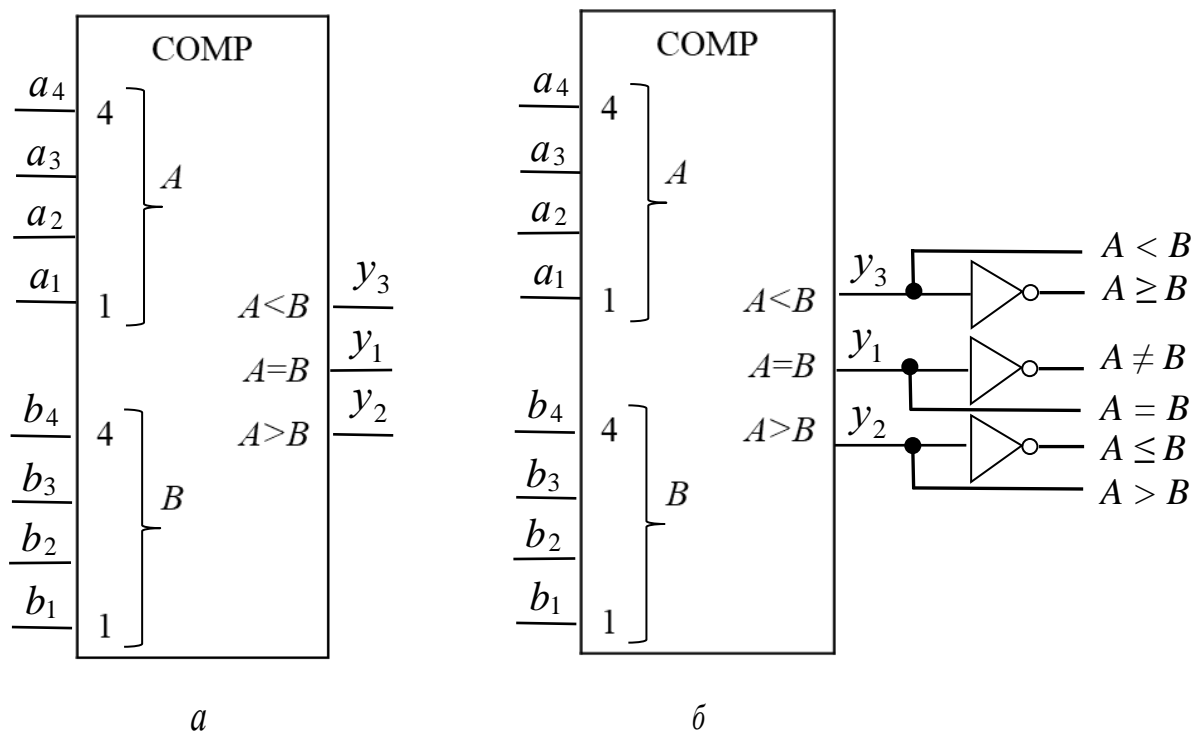


Рис. 5.52. Умовне графічне позначення 4-розрядного компаратора

4. Запишіть логічний вираз для формування функції $y_1(A, B)$, яка ідентифікує відношення $A = B$.
5. Який логічний елемент реалізує функцію рівнозначності?
6. Запишіть логічний вираз для формування функції $y_2(A, B)$, яка відповідає відношенню $A > B$.
7. Чому в складі компаратора достатньо мати лише дві СПК, і які саме?
8. Побудуйте комбінаційну схему компаратора, що забезпечує порівняння 3-розрядних слів A та B .

Задачі для самостійного розв'язування

1. На елементах 2І-НЕ побудувати оптимізовану за апаратною складністю комбінаційну схему, що генерує логічну одиницю на виході, коли вхідний 4-розрядний код $A = a_4a_3a_2a_1$ перевищує 6. Для мінімізації вихідної функції застосувати карту Карно.
2. На елементах ЗАБО-НЕ побудувати оптимізовану за апаратною складністю комбінаційну схему, яка генерує на виході логічну одиницю, коли вхідний 4-розрядний код $A = a_4a_3a_2a_1$ задовольняє умову $\begin{cases} 5 \leq A \leq 9, \\ 10 < A < 14. \end{cases}$

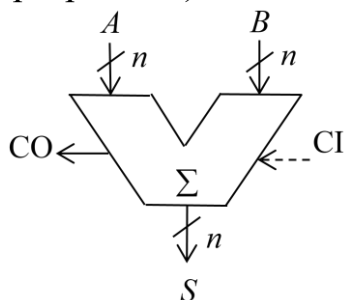
3. Використовуючи компаратори та додаткові логічні елементи побудувати схему, яка визначатиме належність 4-розрядного числа (слова) B до інтервалу $4 \leq B \leq 11$.
4. Побудувати комбінаційну схему 4-розрядного компаратора, що ідентифікує лише рівність слів (кодів).

5.7. Комбінаційні суматори

Комбінаційним суматором називають комбінаційну схему, що реалізує мікрооперацію додавання двох n -розрядних слів (кодів) $A = (a_n a_{n-1} \dots a_2 a_1)$ та $B = (b_n b_{n-1} \dots b_2 b_1)$, які мають такі кількісні еквіваленти: $A_{ке} = \sum_{i=1}^n a_i 2^{i-1}$, $B_{ке} = \sum_{i=1}^n b_i 2^{i-1}$.

Результатом мікрооперації є n -розрядна сума (слово) $S = (s_n s_{n-1} \dots s_2 s_1)$, кількісний еквівалент якої становить $S_{ке} = \sum_{i=1}^n s_i 2^{i-1}$, та значення вихідного переносу CO (carry output), – переносу за межі старшого розряду результату $CO \in \{0, 1\}$.

У мікрооперації додавання двох слів може брати участь (за рішенням програміста) вхідний перенос CI (carry input), $CI \in \{0, 1\}$ (рис. 5.53).

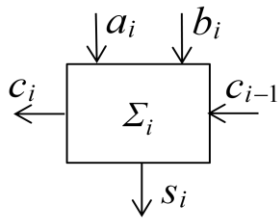


CI – carry input (вхідний перенос),
CO – carry output (вихідний перенос),
 $CI, CO \in \{0, 1\}$

Рис. 5.53. Умовне графічне позначення комбінаційного суматора

Схему n -розрядного комбінаційного суматора будують на основі однорозрядних суматорів.

Однорозрядним комбінаційним суматором називають комбінаційну схему з 3-ма входами та 2-ма виходами, яка забезпечує додавання однорозрядних операндів a_i , b_i з урахуванням переносу c_{i-1} з сусіднього справа (молодшого) розряду та формує один розряд s_i суми, а також значення переносу c_i в наступний розряд, $i = 1, 2, \dots, n$ (рис. 5.54).



c_{i-1} – вхідний перенос (при $i = 1$ $c_0 = CI$),
 c_i – вихідний перенос (при $i = n$ $c_n = CO$)

Рис. 5.54. Однорозрядний суматор

Роботу однорозрядного комбінаційного суматора описують таблицею функціонування (істинності) (табл. 5.17).

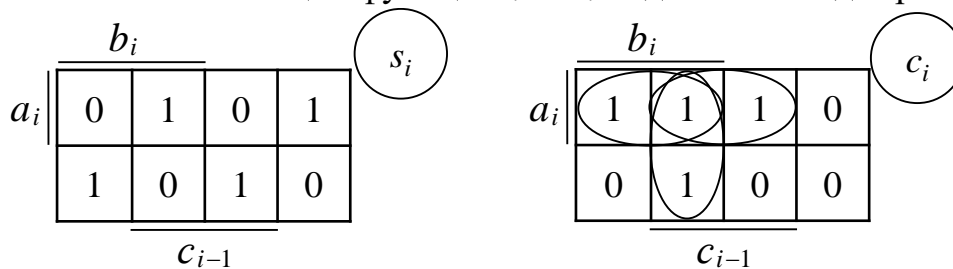
Таблиця 5.17

Таблиця функціонування однорозрядного комбінаційного суматора

Входи			Виходи		$a_i \oplus b_i \oplus c_{i-1}$
a_i	b_i	c_{i-1}	s_i	c_i	
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	0	1	0
1	0	0	1	0	1
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	1	1	1

Синтезуємо комбінаційну схему однорозрядного суматора в елементному базисі алгебри Буля.

Виконаємо мінімізацію функцій s_i та c_i за допомогою діаграм Вейча:



$$\begin{cases} s_i = \bar{a}_i \bar{b}_i c_{i-1} \vee \bar{a}_i b_i \bar{c}_{i-1} \vee a_i \bar{b}_i \bar{c}_{i-1} \vee a_i b_i c_{i-1} \\ c_i = a_i b_i \vee b_i c_{i-1} \vee a_i c_{i-1} \end{cases} \quad (5.16)$$

Системі функцій (5.16) відповідає комбінаційна схема на рис. 5.55, складність якої за Квайном становить $K = 26$, а швидкодія – $t = 3\tau$.

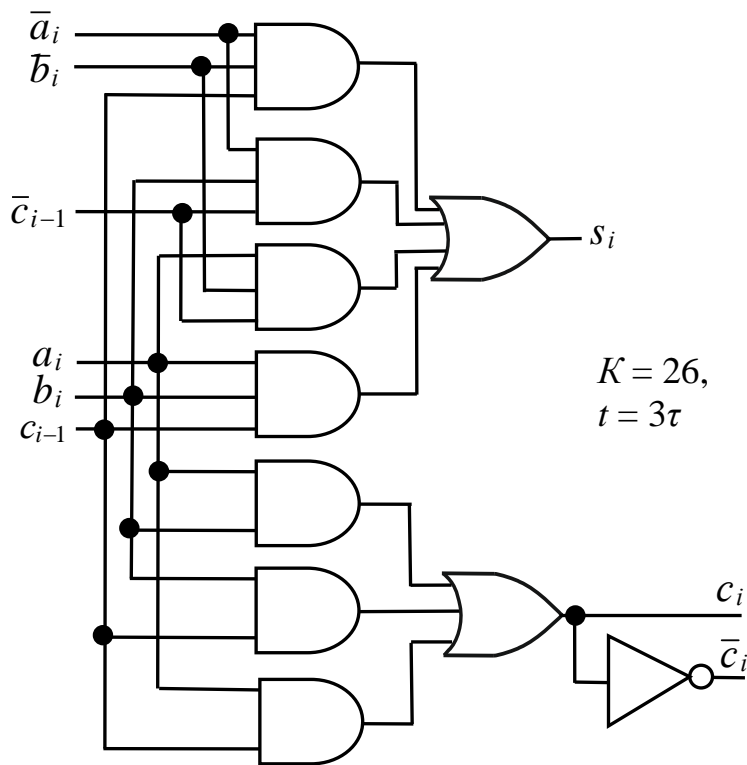
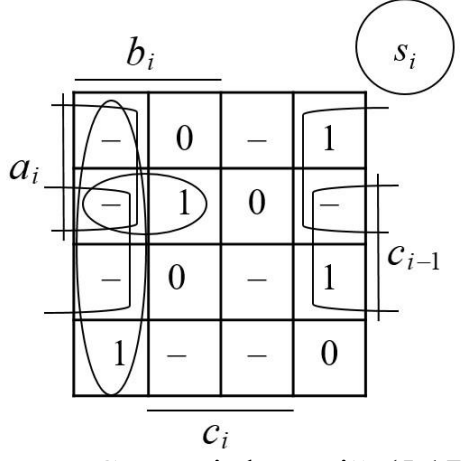


Рис. 5.55. Комбінаційна схема однорозрядного суматора

Функцію s_i можна спростити, якщо розглядати її як залежну від 4-х аргументів: $s_i = f(a_i, b_i, c_{i-1}, c_i)$. У цьому випадку функція визначена лише на 8-ми наборах з можливих 16-ти (див. табл. 5.17). Вона не визначена на наборах: 0001, 0011, 0101, 0110, 1001, 1010, 1100, 1110.

Мінімізуємо неповністю визначену функцію s_i від 4-х аргументів за допомогою діаграми Вейча:



$$s_i = b_i \bar{c}_i \vee a_i b_i c_{i-1} \vee a_i \bar{c}_i \vee c_{i-1} \bar{c}_i.$$

Для зменшення складності схеми функції s_i та c_i подамо у вигляді

$$\begin{cases} s_i = (a_i b_i) c_{i-1} \vee \bar{c}_i ((a_i \vee b_i) \vee c_{i-1}), \\ c_i = a_i b_i \vee c_{i-1} (a_i \vee b_i). \end{cases} \quad (5.17)$$

Системі функцій (5.17) відповідає комбінаційна схема на рис. 5.56, яка характеризується меншою складністю ($K = 17$) порівняно зі схемою на рис. 5.55, однак має нижчу швидкодію ($t = 6\tau$). Перевагою схеми

на рис. 5.56 є те, що в ній використовуються лише прямі змінні (інверсії змінних не потрібні).

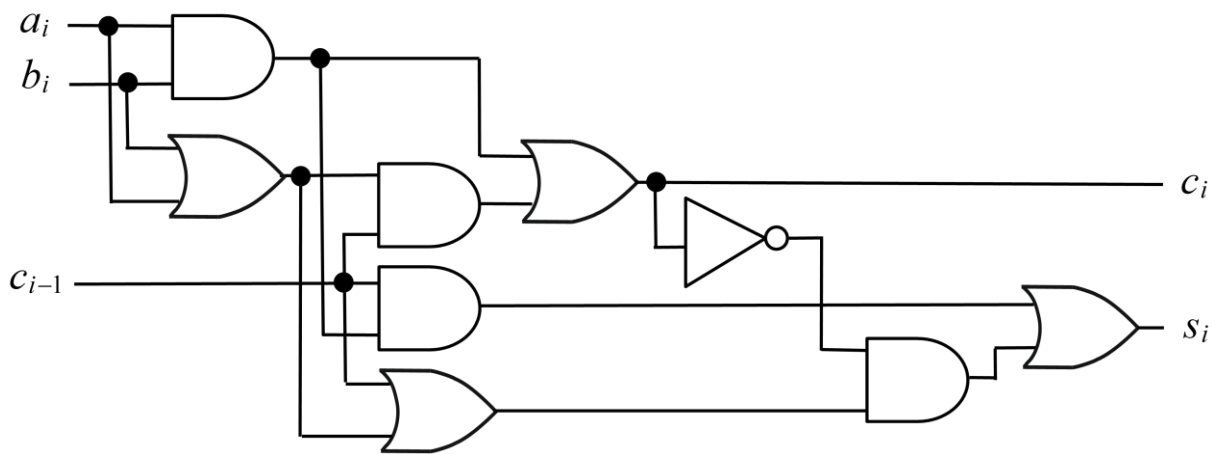


Рис. 5.56. Комбінаційна схема однорозрядного суматора, оптимізована за показником апаратної складності

У табл. 5.17 бачимо, що стовпець s_i співпадає зі стовпцем $a_i \oplus b_i \oplus c_{i-1}$. Це означає, що систему рівнянь (5.16) можна замінити наступною системою:

$$\begin{cases} s_i = a_i \oplus b_i \oplus c_{i-1}, \\ c_i = a_i b_i \vee b_i c_{i-1} \vee a_i c_{i-1}. \end{cases} \quad (5.18)$$

Перше рівняння системи (5.18) можна реалізувати на основі суматорів за модулем два (рис. 5.57).

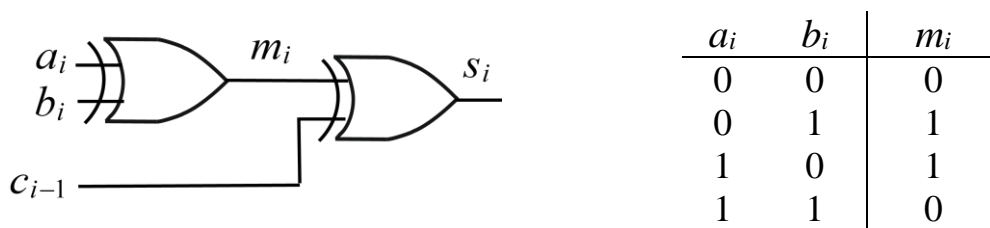


Рис. 5.57. Отримання значення s_i на основі суматорів за модулем два

Але $m_i = \bar{a}_i b_i \vee a_i \bar{b}_i$. Тому в елементному базисі алгебри Буля реалізація системи рівнянь (5.18) має вигляд як показано на рис. 5.58.

Однорозрядний суматор, побудований на основі системи (5.18), має параметри складності: $K = 21$, $t = 4\tau$.

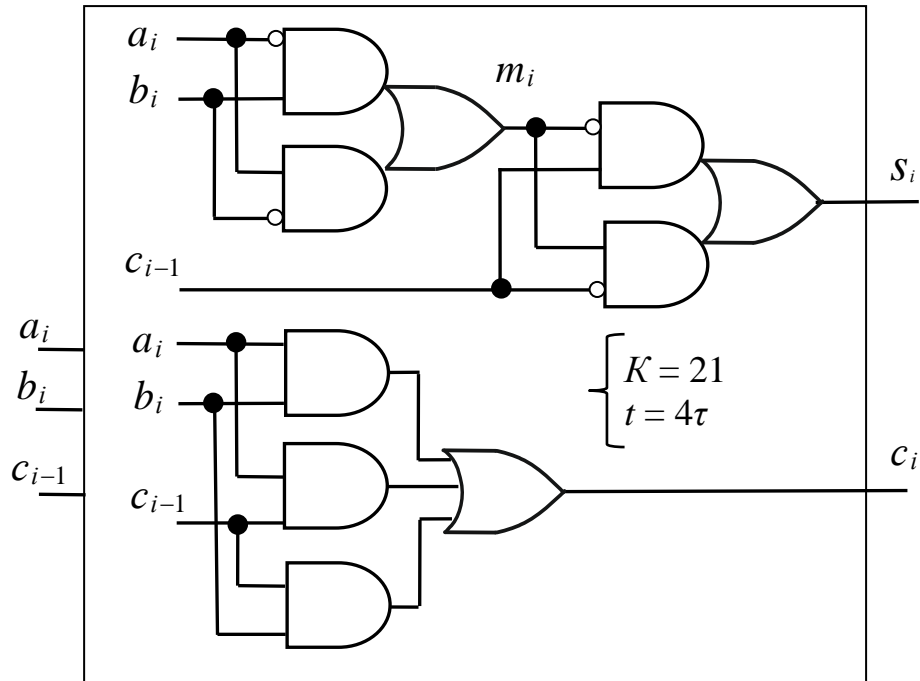


Рис. 5.58. Комбінаційна схема однорозрядного суматора на основі системи рівнянь (5.18)

Порівнюючи комбінаційні схеми однорозрядних суматорів на рис. 5.55, 5.56 та 5.58 за складністю (табл. 5.18) доходимо висновку, що максимальну швидкодію ($t = 3\tau$) має комбінаційна схема на рис. 5.55, яка реалізує систему (5.16), а мінімальну апаратну складність ($K = 17$) має схема на рис. 5.56, що реалізує систему (5.17).

Таблиця 5.18

Порівняння комбінаційних схем однорозрядних суматорів за складністю

Комбінаційна схема		
на рис. 5.55, система (5.16)	на рис. 5.56, система (5.17)	на рис. 5.58, система (5.18)
$K = 26, t = 3\tau$	$K = 17, t = 6\tau$	$K = 21, t = 4\tau$

На кресленнях однорозрядний суматор зображають у вигляді умовного графічного позначення (рис. 5.59), де c_i – вхідний перенос, c_o – вихідний перенос.

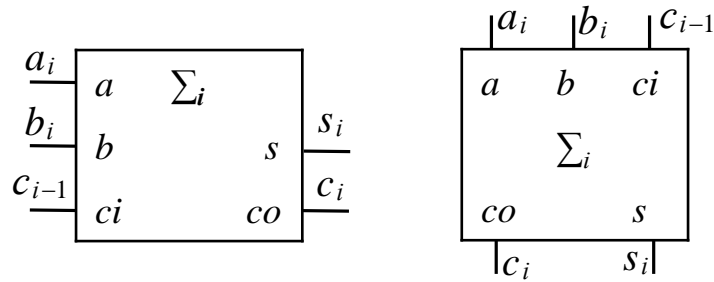


Рис. 5.59. Способи умовного графічного позначення однорозрядного комбінаційного суматора

На основі однорозрядних суматорів будують комбінаційні суматори для додавання багаторозрядних слів (кодів) (рис. 5.60).

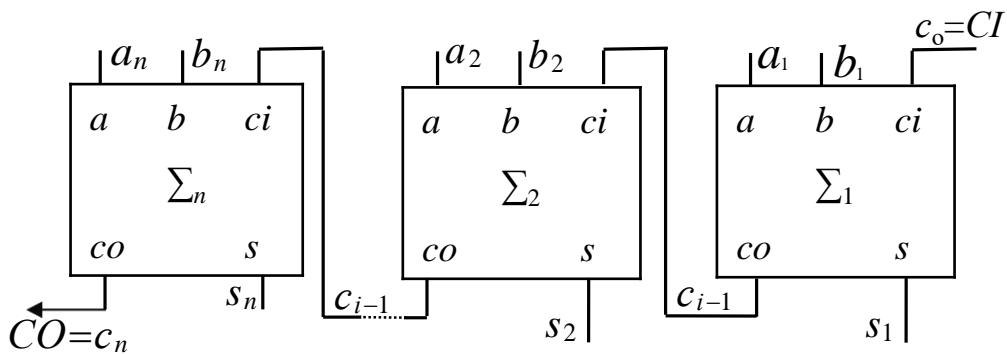


Рис. 5.60. Функціональна схема багаторозрядного комбінаційного суматора з послідовним переносом

Наприклад, 4-розрядний комбінаційний суматор позначають на кресленнях як показано на рис. 5.61.

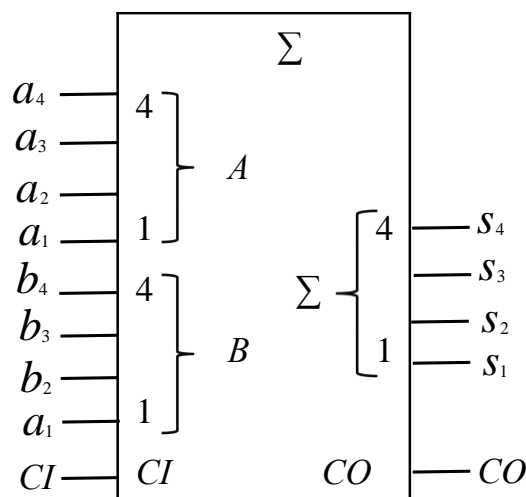


Рис. 5.61. Умовне графічне позначення 4-розрядного комбінаційного суматора

Крім послідовного переносу з одного розряду в інший (див. рис. 5.60), в багаторозрядному суматорі може застосовуватись паралельний перенос та комбіновані способи організації переносу.

Проектування суматора здійснюють виходячи із заданої системи логічних елементів (алгебри), вимог до тривалості виконання мікрооперації додавання та допустимих апаратних та часових витрат.

Суматор є основою арифметико-логічного пристрою (АЛП) процесора.

Запитання та завдання

1. Яку схему називають комбінаційним суматором?
2. Що є результатом мікрооперації додавання двох двійкових n -розрядних слів (кодів)?
3. Дайте визначення однорозрядного комбінаційного суматора.
4. Як описують функціонування однорозрядного комбінаційного суматора?
5. Побудуйте комбінаційну схему однорозрядного суматора в елементному базисі алгебри Жегалкіна.
6. Побудуйте функціональну схему 3-розрядного комбінаційного суматора з послідовним переносом на основі однорозрядних суматорів. Наведіть умовне графічне позначення 3-розрядного комбінаційного суматора.

5.8. Схема формування ознаки парності

Після виконання кожної операції процесор формує спеціальні біти – ознаки (флаги – flags) результату, які необхідні для організації умовних переходів у програмах або умовних викликів підпрограм. Однією з таких ознак є ознака парності коду (слова) результату операції.

Якщо двійковий код результату операції містить парну кількість одиниць, то встановлюється ознака парності $P = 1$, інакше – $P = 0$.

Для автоматичного формування ознаки парності P (її також називають бітом паритету – parity bit) у складі процесора має бути відповідна схема – схема формування ознаки парності (паритету) (СФОП) результату (рис. 5.62).

Схема формування ознаки парності результату операції – це комбінаційна схема, що має n входів ($s_n, s_{n-1}, \dots, s_2, s_1$) та один вихід (P), значення на якому визначають так:

$$P = \begin{cases} 1, \text{ якщо } \sum_{i=1}^n s_i \pmod{2} = 0, \\ 0, \text{ якщо } \sum_{i=1}^n s_i \pmod{2} = 1. \end{cases}$$

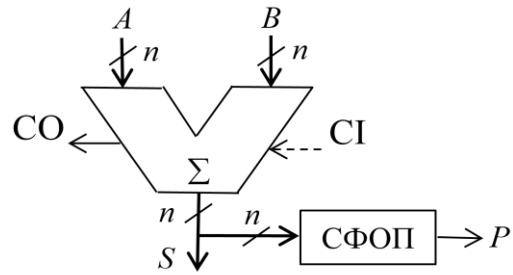
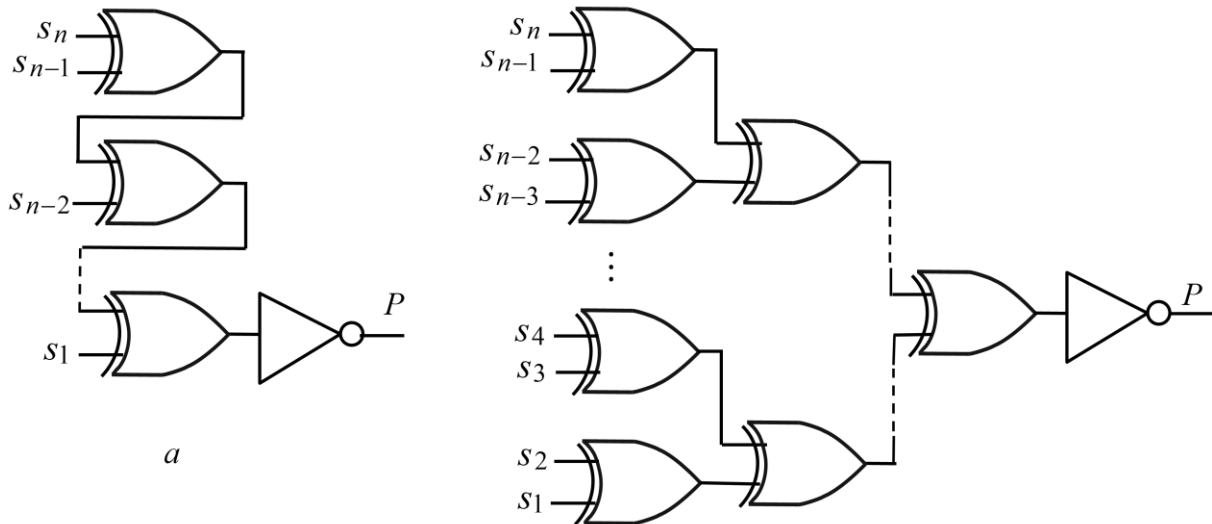


Рис. 5.62. Формування ознаки парності коду результату

Тобто, якщо сума за модулем два для всіх двійкових розрядів коду S результату операції дорівнює нулю (а це означає, що двійковий код результату містить парну кількість одиниць), то $P = 1$. Якщо ж сума за модулем два всіх розрядів коду результату дорівнює одиниці (а це означає, що в двійковому коді результату міститься непарна кількість одиниць), то $P = 0$.

Якщо $S = 0$, то $P = 1$.

Схема формування ознаки парності коду результату може мати послідовнісний (каскадний) або пірамідальний вигляд (рис. 5.63).



$$K_K = 2(n-1) + 1 = 2n - 1$$

$$t_K = (n-1)\tau_{\text{mod}2} + \tau_{\text{інв}}$$

\bar{b}

$$K_{\Pi} = 2(n-1) + 1 = 2n - 1$$

$$t_{\Pi} = (\log_2 n)\tau_{\text{mod}2} + \tau_{\text{інв}}$$

Рис. 5.63. Комбінаційна схема для формування ознаки парності P :
 a – послідовнісного (каскадного) типу; \bar{b} – пірамідального типу

Обидві схеми мають однакову складність за Квайном ($K_K = K_{II}$), але різну швидкодію. Оскільки $n - 1 > \log_2 n$, де $n > 2$, то схема пірамідального типу має вищу швидкодію (див. рис. 5.63).

Схеми формування ознаки парності використовують також в системах передачі даних та системах зберігання даних для виявлення спотворень окремих бітів в словах при їх передачі або зберіганні.

Запитання та завдання

1. Дайте визначення, що таке схема формування ознаки парності коду (слова) результату операції.
2. З якою метою в процесорі формують ознаку (флаг) парності коду результату?
3. Як часто в процесорі необхідно формувати ознаку парності (паритету) результату?
4. Яким є значення біта парності P , якщо отримано результат 01011001?
5. Якого значення набуває ознака парності P , якщо код результату:
а) містить непарну кількість одиниць; б) дорівнює нулю?
6. Побудуйте комбінаційну схему:
а) послідовнісного (каскадного) типу; б) пірамідального типу для формування ознаки парності коду результату в 8-розрядному процесорі.

5.9. Програмовні логічні матриці

Розвиток технології інтегральних схем дозволяє розмістити на одному кристалі велику кількість логічних елементів, які можна організувати у вигляді *програмовної логічної матриці* (ПЛМ), що має такі входи та виходи: x_n, \dots, x_1 – інформаційні входи; y_m, \dots, y_1 – інформаційні виходи; T_k, \dots, T_1 – проміжні внутрішні шини (рис. 5.64).

ПЛМ є комбінаційною схемою, що дозволяє реалізовувати системи перемикальних функцій y_m, \dots, y_1 , поданих у формі І/АБО або у формі І/АБО-НЕ. Структурно ПЛМ складається з двох масивів логічних елементів – $2n$ -входових елементів І та k -входових елементів АБО. Логічні елементи І, виходи яких утворюють внутрішні шини T_i ($i = 1, \dots, k$), дозволяють реалізовувати довільні кон'юнктивні терми, в тому числі й конституенти одиниці.

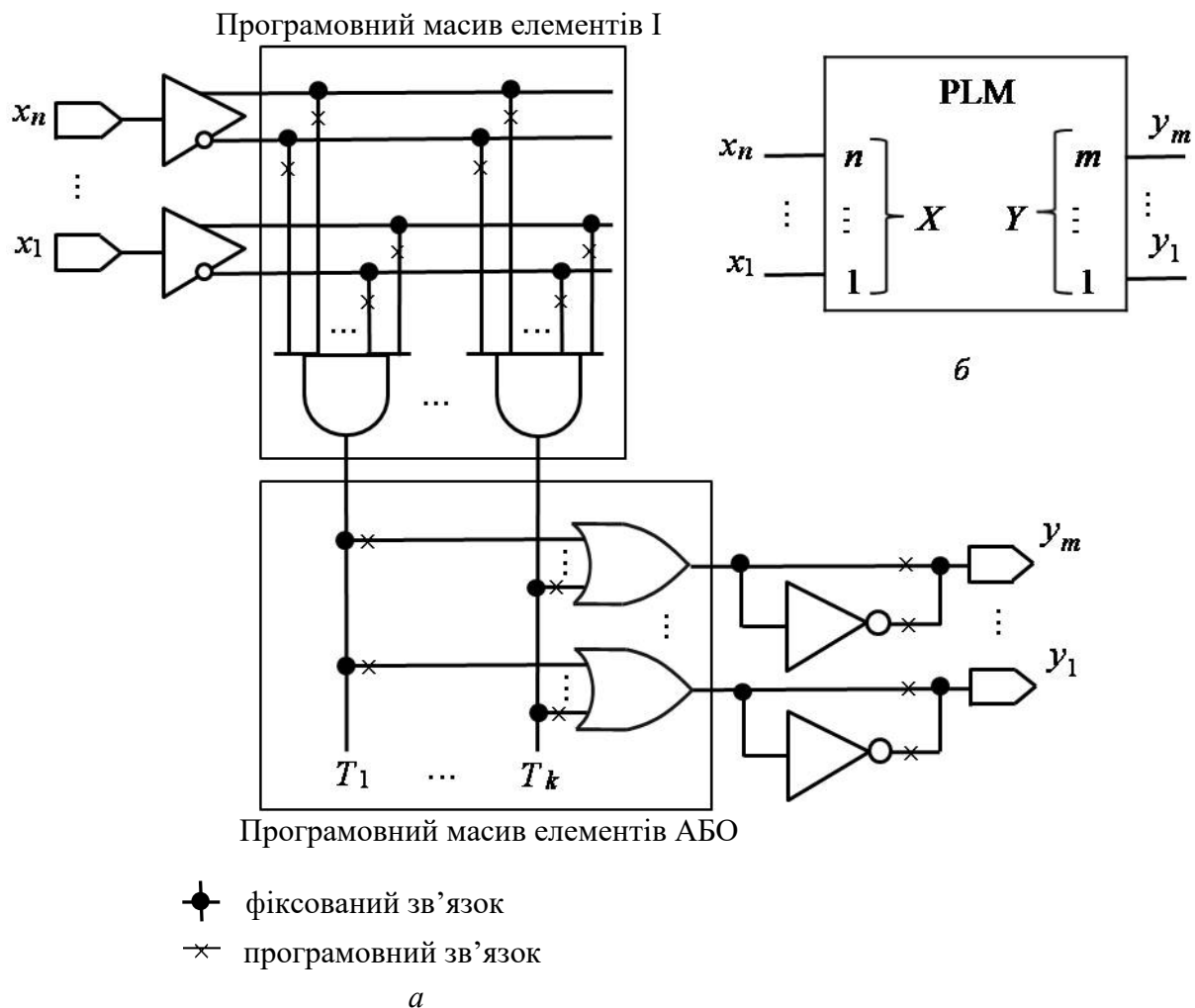


Рис. 5.64. Программовна логічна матриця:
a – узагальнена структура ПЛМ; *б* – умовне графічне позначення ПЛМ

Оскільки программовна логічна матриця характеризується параметрами n , k , m , де n – кількість інформаційних входів, k – кількість проміжних внутрішніх шин, m – кількість інформаційних виходів, то її позначають у вигляді: ПЛМ (n , k , m).

Програмування (настроювання) ПЛМ полягає у встановленні або вилученні (перепалюванні, руйнуванні) зв'язків у точках, позначених знаком \times . Програмування ПЛМ здійснюють за допомогою програматора – спеціальної плати, що приєднується до комп'ютера. На плату програматора встановлюють мікросхему ПЛМ, та за допомогою спеціальної комп'ютерної програми програмують (настроюють) її.

На систему перемикальних функцій, що підлягає реалізації на ПЛМ, накладаються такі обмеження:

- кількість аргументів системи функцій не має перевищувати n – кількість входів ПЛМ,

- кількість функцій не має перевищувати m – кількість виходів ПЛМ,
- сумарна кількість різних термів, що входять до складу системи перемикальних функцій, не повинна перевищувати k – кількості проміжних внутрішніх шин.

Задача 5.8. Реалізувати на ПЛМ систему перемикальних функцій

$$\begin{cases} y_4 = \overline{x_3 x_1} \vee \overline{x_4 x_3 x_2 x_1} \vee \overline{x_4 x_1}, \\ y_3 = \overline{x_4 x_2 x_1} \vee \overline{x_4 x_3 x_2} \vee \overline{x_4 x_3 x_1}, \\ y_2 = \overline{x_4 x_3 x_1} \vee \overline{x_4 x_3 x_2} \vee \overline{x_4 x_1}, \\ y_1 = \overline{x_4 x_2 x_1} \vee \overline{x_4 x_3 x_2 x_1} \vee \overline{x_4 x_1}. \end{cases}$$

Розв'язування.

1. Функції y_1, y_3 подані в формі І/АБО, а функції y_4, y_2 – у формі І/АБО-НЕ. Отже, задана система перемикальних функцій може бути реалізована на ПЛМ.

2. До складу системи перемикальних функцій входять 7 різних термів:

$$\begin{aligned} T_1 &= x_3 x_1, & T_2 &= \overline{x_4} \overline{x_3} \overline{x_2} x_1, & T_3 &= \overline{x_4} \overline{x_1}, & T_4 &= \overline{x_4} x_2 \overline{x_1}, \\ T_5 &= x_4 x_3 \overline{x_2}, & T_6 &= \overline{x_4} \overline{x_3} \overline{x_1}, & T_7 &= x_4 x_3 x_1. \end{aligned}$$

Тоді, задану систему функцій можна подати у вигляді:

$$\begin{cases} y_4 = \overline{T_1} \vee \overline{T_2} \vee \overline{T_3}, \\ y_3 = \overline{T_4} \vee \overline{T_5} \vee \overline{T_6}, \\ y_2 = \overline{T_7} \vee \overline{T_5} \vee \overline{T_3}, \\ y_1 = \overline{T_4} \vee \overline{T_2} \vee \overline{T_3}. \end{cases}$$

3. Складемо *мнемонічну схему* ПЛМ (рис. 5.65), яка спрощено показує зв'язок шин матриці елементів І (формування кон'юнктивних термів) та матриці елементів АБО, а також спосіб формування вихідних функцій (y_i) схеми (прямі чи інвертовані виходи).

З мнемонічної схеми визначаємо параметри ПЛМ: $n = 4, k = 7, m = 4$.

Отже, для реалізації заданої системи перемикальних функцій необхідно скористатися ПЛМ (4, 7, 4) (рис. 5.66).

4. На основі мнемонічної схеми складемо карту програмування ПЛМ (табл. 5.19). ▣

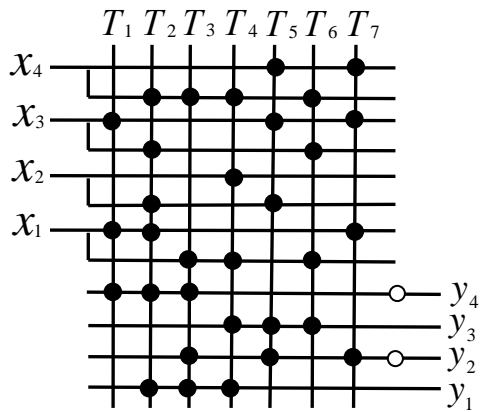


Рис. 5.65. Мнемонічна схема ПЛМ

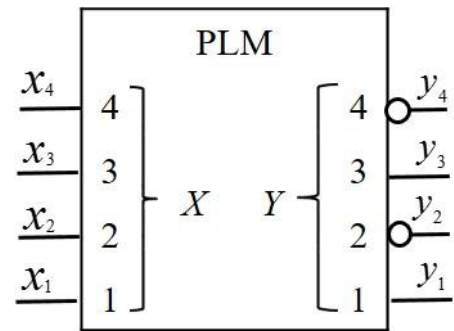


Рис. 5.66. Умовне графічне позначення ПЛМ (4, 7, 4)

Таблиця 5.19

Карта програмування ПЛМ (4, 7, 4)

x_4	x_3	x_2	x_1	T_i	\bar{y}_4	y_3	\bar{y}_2	y_1
—	1	—	1	T_1	1	0	0	0
0	0	0	1	T_2	1	0	0	1
0	—	—	0	T_3	1	0	1	1
0	—	1	0	T_4	0	1	0	1
1	1	0	—	T_5	0	1	1	0
0	0	—	0	T_6	0	1	0	0
1	1	—	1	T_7	0	0	1	0

У загальному випадку під час синтезу комбінаційних схем, що реалізують системи перемикальних функцій на основі ПЛМ, спочатку необхідно виконати мінімізацію кожної функції, що входять до заданої системи, та подати їх у формі І/АБО або І/АБО-НЕ. Мінімізована система перемикальних функцій повинна мати якомога меншу кількість термів, що відрізняються один від одного.

Запитання та завдання

1. Яку комбінаційну схему називають програмовною логічною матрицею.
2. З якою метою використовують ПЛМ під час синтезу комбінаційних схем?
3. У якій формі мають бути подані перемикальні функції, що підлягають реалізації на ПЛМ?

4. Як структурно організована ПЛМ?
5. Як здійснюють програмування ПЛМ?
6. Які обмеження накладаються на систему перемикальних функцій, яка має бути реалізована на ПЛМ?
7. В якій послідовності виконують реалізацію заданої системи перемикальних функцій на ПЛМ?
8. Реалізуйте на ПЛМ систему перемикальних функцій

$$\begin{cases} y_3 = x_4 x_2 \bar{x}_1 \vee x_4 \bar{x}_3 \bar{x}_2 \vee \bar{x}_2 x_1, \\ y_2 = \overline{x_4 x_3 x_2 x_1} \vee x_4 \bar{x}_2 x_1 \vee \bar{x}_3 x_2 x_1 \vee x_4 \bar{x}_3 \bar{x}_2, \\ y_1 = \bar{x}_4 x_3 x_2 \bar{x}_1 \vee x_4 x_2 x_1 \vee \bar{x}_3 x_2 x_1 \vee x_4 \bar{x}_3 \bar{x}_2. \end{cases}$$

6.

БІСТАБІЛЬНІ СХЕМИ

Для реалізації обчислень до складу операційних пристроїв комп'ютера мають входити логічні схеми, що здатні запам'ятовувати біти інформації. Для того, щоб логічна схема мала здатність запам'ятовувати 1 біт, вона повинна мати два стійкі стани, тобто бути бістабільною. Властивість бістабільності можуть мати лише логічні схеми, що мають петлі (зворотні зв'язки).

Як відомо, *петля* – це електричне коло, по якому сигнал з виходу i -го логічного елемента безпосередньо або через інші логічні елементи може надходити на один з входів цього самого i -го елемента.

Комбінаційні схеми не мають петель у своїй структурі. У комбінаційних схемах кожен з входів будь-якого логічного елемента з'єднаний з виходом одного з попередніх елементів.

Якщо логічні схеми мають петлі, їх називають *послідовнісними схемами* (ПС) або *схемами з пам'яттю*.

Функціонування комбінаційної схеми можна описати системою перемикальних функцій; поведінка послідовної схеми не може бути повністю описана системою перемикальних функцій.

Вивчення властивостей послідовних схем, що можуть перебувати у двох стійких станах, відкриває можливості для створення вузлів та пристроїв комп'ютерної техніки, які здатні запам'ятовувати як окремі біти, так і слова інформації. А без запам'ятовування проміжних результатів неможливо організувати обчислювальний процес у комп'ютері.

6.1. Аналіз послідовних схем

Як приклад проаналізуємо поведінку послідовної схеми на рис. 6.1. Вона має дві петлі: перша – з'єднує вихід логічного елемента (3) з першим входом цього самого елемента через логічний елемент (1); друга – з'єднує вихід логічного елемента (4) з його входом через логічні елементи (2) та (3).

Функціонування схеми на рис. 6.1 описують рівнянням

$$\overline{\overline{x_2 \vee \overline{Q} \vee x_1 \vee Q}} = Q \quad \text{або} \quad \overline{x_2}Q \vee \overline{x_1}\overline{Q} = Q,$$

де Q – прямий вихід схеми (вихід логічного елемента (4)), а \overline{Q} – інверсний вихід схеми (\overline{Q} є входом логічного елемента (4), який виконує функцію інвертора).

Для того, щоб розв'язати отримане рівняння, необхідно скласти таблицю функціонування схеми в часі (табл. 6.1).

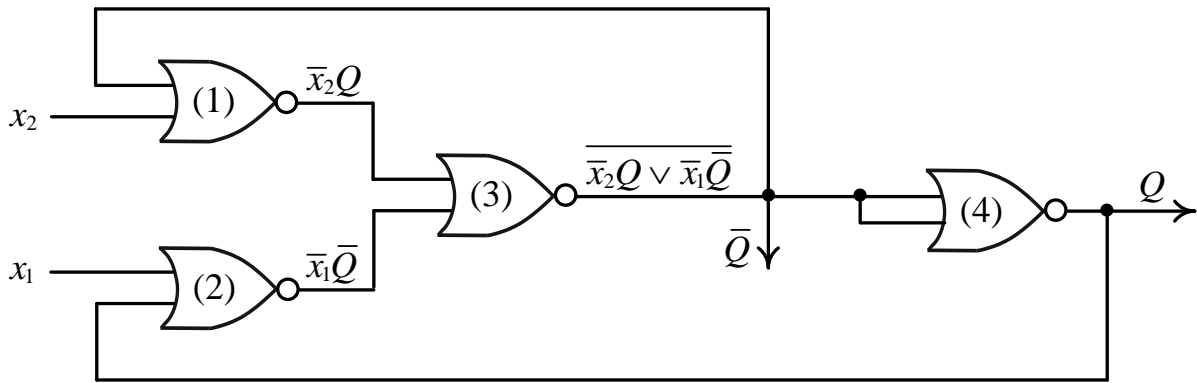


Рис. 6.1. Приклад послідовної схеми

Таблиця 6.1

Поведінка схеми на рис. 6.1 в часі

Входи схеми		Стан схеми	Новий стан схеми	Примітка
$x_2(t_i)$	$x_1(t_i)$	$Q(t_i)$	$Q(t_{i+1})$	
0	0	×	1	$Q(t_{i+1}) = 1$, стійкий стан
0	1	0	0	$Q(t_{i+1}) = Q(t_i)$, підтвердження (незмінність) попереднього стану
		1	1	
1	0	0	1	Коливання $0 \rightarrow 1, 1 \rightarrow 0$ з частотою $F = 1 / (6\tau)$
		1	0	
1	1	×	0	$Q(t_{i+1}) = 0$, стійкий стан

Нехай t_i та t_{i+1} – два сусідні відліки автоматного часу.

Якщо в момент часу t_i на входах схеми діють сигнали $x_2(t_i) = 0$, $x_1(t_i) = 0$, то незалежно від того, в якому стані в момент t_i перебуває схема – 0 чи 1 ($Q(t_i) = \times$), у наступний момент часу (t_{i+1}) на виході Q схеми матимемо стійке значення $Q(t_{i+1}) = 1$.

Якщо на входах схеми діють сигнали $x_2(t_i) = 1$, $x_1(t_i) = 1$, то незалежно від поточного стану схеми ($Q(t_i) = \times$), в момент часу t_{i+1} на виході Q матимемо стійке значення $Q(t_{i+1}) = 0$.

При $x_2(t_i) = 1$, $x_1(t_i) = 0$ у схемі відбуватиметься коливальний процес з частотою $F = 1 / (6\tau)$.

Дійсно, нехай у даний момент часу на виході схеми має місце $Q = 1$. Тоді через час τ , що дорівнює часу проходження сигналу через один логічний елемент, на виходах елементів (1) та (2) встановиться сигнал 0. Через час 2τ на виході елемента (3) встановиться сигнал 1, а через час 3τ на виході Q встановиться сигнал 0, і т.д. Таким чином, у схемі відбуватимуться коливання $1 \rightarrow 0, 0 \rightarrow 1$ з періодом 6τ .

При $x_2(t_i) = 0$, $x_1(t_i) = 1$ значення сигналу на виході Q схеми в момент часу t_{i+1} буде таким самим, як і в попередній момент (t_i), тобто $Q(t_{i+1}) = Q(t_i)$, що означає підтвердження попереднього стану.

Таким чином, бачимо, що послідовнісна схема на рис. 6.1 буде працювати стабільно, коли на входи x_2 , x_1 схеми надходять комбінації 00, 01 та 11. При $x_2 = 1$, $x_1 = 0$ стан на виході схеми не буде стабільним – то 0, то 1 (коливання).

Аналіз логічних схем, подібних до схеми на рис. 6.1, тобто таких, що мають петлі, дає підстави зробити наступні висновки.

1. У послідовнісних схемах значення на виході схеми в даний момент часу залежать не лише від значень вхідних змінних, що діють у цей самий момент часу, але й від стану, в якому в даний момент перебуває схема, тобто й від значень вхідних змінних, що діяли в попередній момент часу і викликали цей стан.

2. Послідовнісна схема може перебувати у кількох (двох або більше) стійких станах.

Стан схеми називають *стійким*, якщо значення сигналів на її виходах може зберігатися необмежено довго (поки на схему подано електроживлення).

3. Наявність у схемі двох або більше стійких станів означає, що схему можна використовувати для запам'ятовування інформації (значень сигналів, що надходять на її виходи).

Для того, щоб послідовнісну схему можна було використовувати для запам'ятовування одного біта інформації (значень сигналів, що надходять на її входи) необхідно, щоб забезпечувались три варіанти функціонування схеми:

- 1) встановлення схеми у стан, що відповідає цифрі 0,
- 2) встановлення схеми у стан, що відповідає цифрі 1,
- 3) режим зберігання – коли вхідні сигнали не змінюють стану схеми у наступний момент часу, тобто коли $Q(t_{i+1}) = Q(t_i)$.

Отже, умовою використання послідовнісної схеми як елемента пам'яті є:

$$\{ "0", "1", Q(t_{i+1}) = Q(t_i) \}.$$

Послідовнісну схему на рис. 6.1 можна використовувати як генератор (Γ) тактових сигналів з частотою $1/(6\tau)$ (коли на входи схеми подано комбінацію сигналів $x_2 = 1$, $x_1 = 0$), або як елемент пам'яті (ЕП), здатний запам'ятовувати 1 біт (рис. 6.2).

Розглянемо застосування послідовнісної схеми на рис. 6.1 як генератора синхросигналів. Такий генератор може бути організований на основі мікросхеми 4(2АБО-НЕ) на 14 виводів, до складу якої входять чотири незалежні двовходові логічні елементи АБО-НЕ (рис. 6.3). Живлення (+5 В) подають на вивод 14 мікросхеми, а 0 В – на вивод 7. Виводи 1 та 2

мікросхеми є входами логічного елемента (1), а вивод 3 є виходом цього елемента. Відповідність виводів мікросхеми та входів-виходів інших логічних елементів показана на рис. 6.3б.

Для отримання генератора необхідно виконати з'єднання виводів мікросхеми як показано на рис. 6.4. Виводи 2 та 4 мікросхеми є входами генератора, на них подають відповідно сигнал логічної одиниці (лог. "1") та логічного нуля (лог. "0") ($x_2 = 1, x_1 = 0$), а вивод 13 мікросхеми є прямим виходом (Q) генератора.

Порівнюючи між собою комбінаційні та послідовні схеми (табл. 6.2), бачимо, що послідовні схеми мають нову якість – здатність запам'ятовувати, оскільки можуть перебувати в кількох станах.

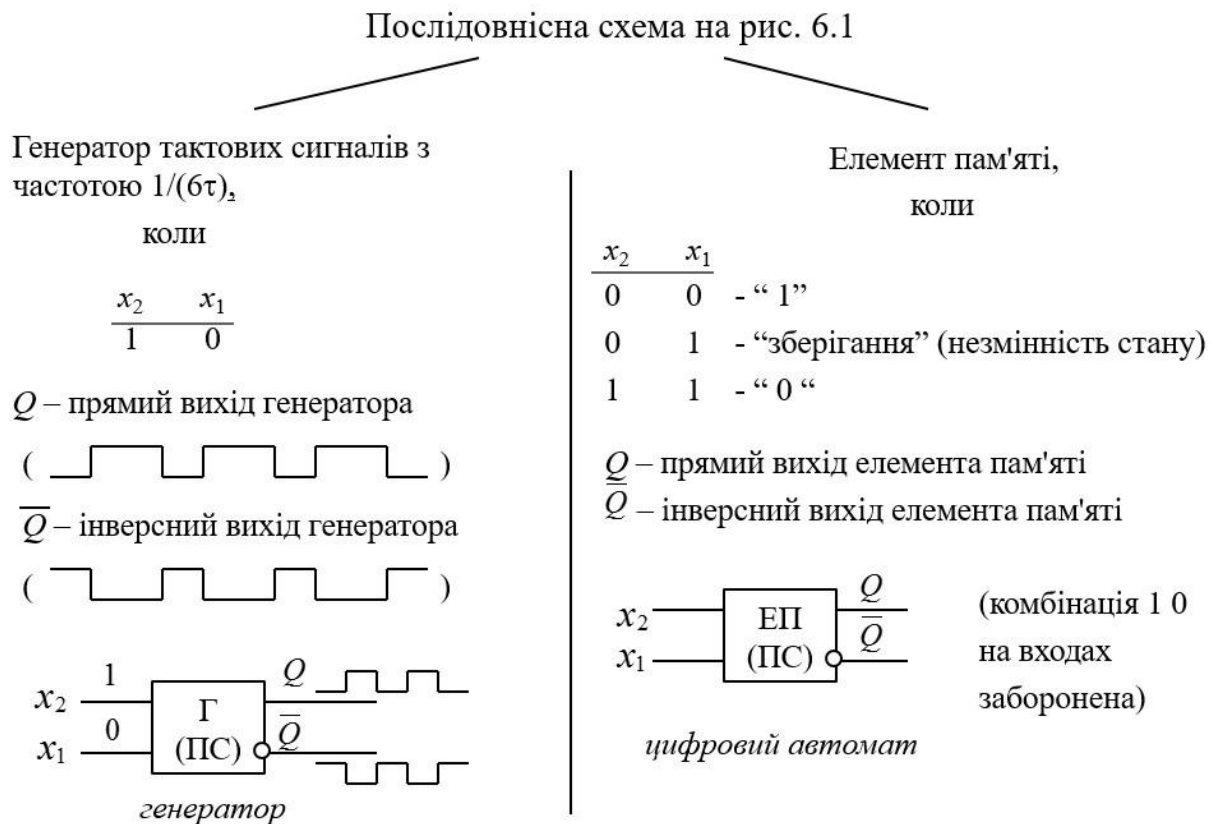


Рис. 6.2. Варіанти використання схеми на рис. 6.1 на практиці

Послідовнісну схему на рис. 6.1, що складається з чотирьох логічних елементів, головним чином використовують як генератор синхросигналів, а для запам'ятовування одного біта інформації розроблено дві простіші схеми, до складу кожної з яких входять лише по 2 логічні елементи. Одна з цих схем побудована на основі двох 2-входових елементів І-НЕ, і її називають *бістабільною схемою на елементах І-НЕ*, а інша – на основі двох 2-входових елементів АБО-НЕ, її називають *бістабільною схемою на елементах АБО-НЕ*. На основі цих елементарних схем будують складніші вузли, що здатні запам'ятовувати як окремі біти, так і слова.

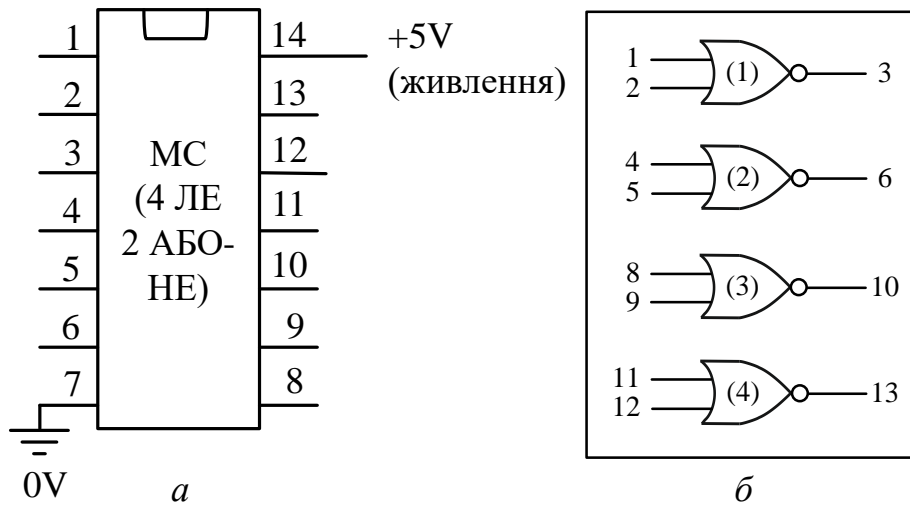


Рис. 6.3. Мікросхема 4(2АБО-НЕ):
a – розташування виводів; *б* – структура мікросхеми

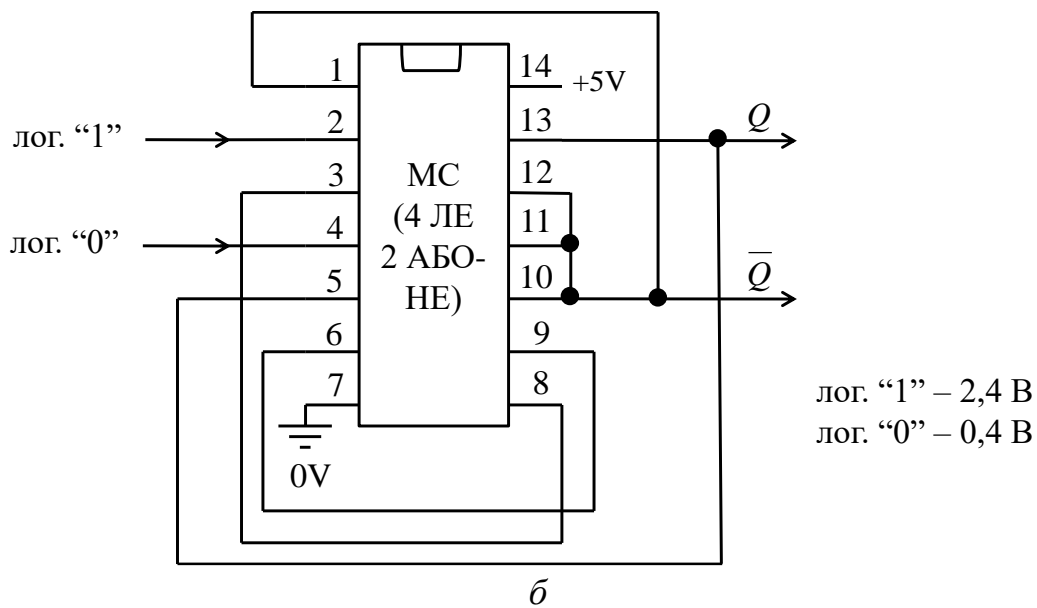
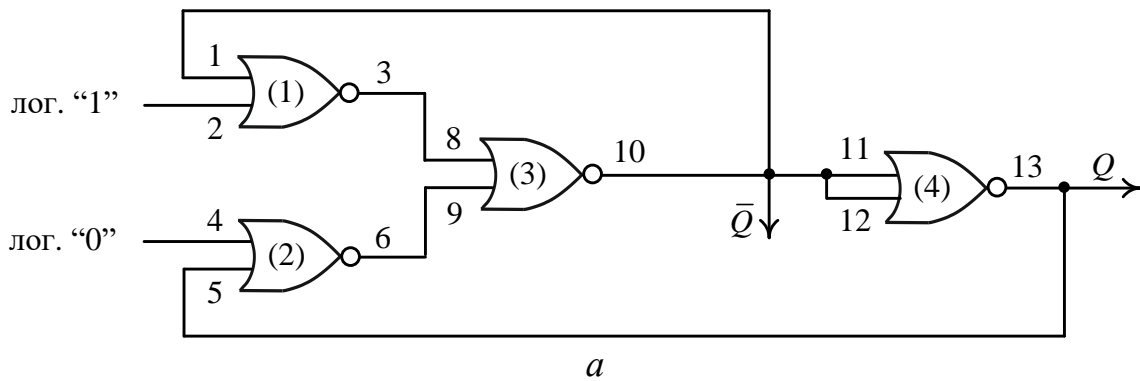


Рис. 6.4. Генератор синхросигналів:
a – схема генератора; *б* – з'єднання виводів мікросхеми

Таблиця 6.2

Порівняння комбінаційних та послідовнісних схем

Показник	Комбінаційна схема	Послідовнісна схема
Характер функціонування	Сукупність вихідних сигналів у даний момент часу повністю визначається сукупністю вхідних сигналів, що діють у цей самий момент часу, й не залежить від вхідних сигналів, що діяли в попередній момент часу	Сукупність вихідних сигналів у даний момент часу залежить не лише від значень вхідних сигналів, що діють у цей самий момент часу, а й від значень сигналів, що діяли в попередній момент часу
Опис поведінки	Поведінка схеми повністю описується системою перемикальних функцій	Поведінка схеми не може бути повністю описана системою перемикальних функцій
Кількість станів	1	2 або більше
Властивість пам'яті	не має	має

Запитання та завдання

1. Яка відмінність між комбінаційними та послідовнісними схемами?
2. Дайте визначення бістабільної схеми.
3. Для якої мети в комп'ютері застосовують бістабільні схеми?
4. Як візуально розрізняють комбінаційні схеми та послідовнісні схеми?
5. Запишіть та поясніть умову використання послідовнісної схеми як елемента пам'яті.
6. Побудуйте послідовнісну схему, замінивши на рис. 6.1 логічні елементи 2АБО-НЕ на елементи 2І-НЕ. Дослідіть роботу новоотриманої схеми та порівняйте її поведінку з поведінкою схеми на рис. 6.1.

6.2. Бістабільна схема на елементах І-НЕ

Схема із зворотними зв'язками перебуває у стійкому стані, якщо значення сигналів на її виходах можуть утримуватись необмежено довго.

Як організувати логічну схему мінімальної складності та максимальної швидкодії, яка б мала властивості елемента пам'яті?

Така схема повинна мати два стійкі стани, що й забезпечує зберігання одного біта інформації.

Розглянемо логічну схему, утворену з двох логічних елементів 2І-НЕ, з'єднаних перехресними зв'язками як показано на рис. 6.5. Схема має два входи – f_2 , f_1 , та два виходи: прямий (Q) та інверсний (\bar{Q}). Вона є послідовнісною, оскільки в ній присутні дві петлі: перша – з'єднує вихід Q верхнього логічного елемента з другим входом цього самого елемента через нижній логічний елемент; друга петля з'єднує вихід \bar{Q} нижнього логічного елемента з першим входом цього самого елемента через верхній логічний елемент.

Функціонування такої схеми можна описати системою рівнянь

$$\begin{cases} Q = f_2 \bar{Q}, \\ \bar{Q} = f_1 Q. \end{cases}$$

Схема на рис. 6.5 може перебувати у двох стійких станах, тому вона є бістабільною схемою (БС).

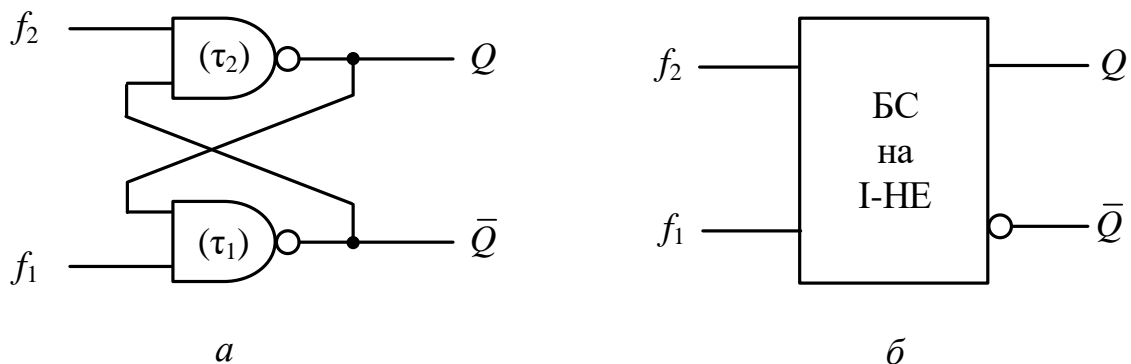


Рис. 6.5. Бістабільна схема на елементах 2І-НЕ:
a – структура схеми; *б* – умовне графічне позначення схеми

Нехай елементи у складі бістабільної схеми характеризуються часом затримки поширення сигналів τ_2 і τ_1 відповідно.

Для надійного функціонування схеми як елемента пам'яті необхідно, щоб у кожному стійкому стані значення на виходах схеми були протилежними, тобто $(Q) \neq (\bar{Q})$. Це означає, що якщо на виході Q встановився рівень логічного нуля, то на виході \bar{Q} має бути рівень логічної одиниці, а якщо на виході Q встановиться рівень логічної одиниці, то на виході \bar{Q} має бути рівень логічного нуля.

Необхідно знайти такі умови функціонування схеми, щоб вона могла перебувати у двох стійких станах.

Вважатимемо, що схема перебуває у стійкому стані "1", якщо на виходах схеми має місце $Q = 1$, $\bar{Q} = 0$, а у стійкому стані "0", якщо $Q = 0$, $\bar{Q} = 1$.

Твердження 6.1. Подавати комбінацію сигналів $f_2 = 0, f_1 = 0$ на входи схеми заборонено.

Доведення. Нехай на входах схеми встановлено сигнали $f_2 = 0, f_1 = 0$, а схема перебуває в одному зі стійких станів. Тоді на виходах схеми отримаємо

$$\begin{pmatrix} f_2 = 0 \\ f_1 = 0 \end{pmatrix} * \begin{pmatrix} Q \\ \bar{Q} \end{pmatrix} \Rightarrow \left\{ \begin{array}{l} Q = \overline{f_2 \cdot \bar{Q}} = \overline{0 \cdot \bar{Q}} = 1 \\ \bar{Q} = \overline{f_1 \cdot Q} = \overline{0 \cdot Q} = 1 \end{array} \right\} \Rightarrow \begin{pmatrix} Q = 1 \\ \bar{Q} = 1 \end{pmatrix},$$

тобто схема перейде у невизначений стан – ні “0”, ні “1”.

Крім того, поява на виходах схеми значень $Q = 1, \bar{Q} = 1$ може спричинити в схемі коливальний процес, наприклад, якщо в наступному такті на входи схеми надійде комбінація сигналів $f_2 = 1, f_1 = 1$. Дійсно:

$$\begin{pmatrix} f_2 = 1 \\ f_1 = 1 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \end{pmatrix} \Rightarrow \left\{ \begin{array}{l} Q = \overline{1 \cdot 1} = 0 \\ \bar{Q} = \overline{1 \cdot 1} = 0 \end{array} \right\} \Rightarrow \begin{pmatrix} f_2 = 1 \\ f_1 = 1 \end{pmatrix} * \begin{pmatrix} 0 \\ 0 \end{pmatrix} \Rightarrow \left\{ \begin{array}{l} Q = \overline{1 \cdot 0} = 1 \\ \bar{Q} = \overline{1 \cdot 0} = 1 \end{array} \right\} \Rightarrow \dots$$

Тобто на виходах схеми з частотою $F = 1/(2\tau)$, де $\tau = \max(\tau_2, \tau_1)$, матимуть місце коливання:

$$\begin{pmatrix} Q \\ \bar{Q} \end{pmatrix} \Rightarrow \begin{pmatrix} 1 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 0 \end{pmatrix} \rightarrow \dots$$

Таким чином, комбінація сигналів $f_2 = 0, f_1 = 0$ на входах схеми є неприйнятною.

Твердження 6.2. Комбінації сигналів 01, 10, 11 на входах f_2, f_1 забезпечують стабільну роботу схеми.

Доведення.

1. Нехай на входах схеми встановлено сигнали $f_2 = 0, f_1 = 1$. Вважатимемо, що на виходах мають місце сигнали $Q = 0, \bar{Q} = 1$, тобто схема перебуває в стані “0”.

Через час 2τ на виходах схеми остаточно встановляться нові значення сигналів: $Q = 1, \bar{Q} = 0$, тобто схема перейде в стан “1”. Дійсно:

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \Rightarrow \left\{ \begin{array}{l} Q = \overline{0 \cdot 1} = 1 \\ \bar{Q} = \overline{1 \cdot 0} = 1 \end{array} \right\} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \text{ – короткочасний нестійкий стан (некритичний)}$$

$$\downarrow$$

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \end{pmatrix} \Rightarrow \left\{ \begin{array}{l} Q = \overline{0 \cdot 1} = 1 \\ \bar{Q} = \overline{1 \cdot 1} = 0 \end{array} \right\} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ – стан “1”}.$$

У процесі переходу зі стану “0” в стан “1” через час τ матиме місце короткочасний нестійкий стан $Q=1, \bar{Q}=1$, який через час τ зміниться на стан $Q=1, \bar{Q}=0$ (стійкий стан).

Припустимо, що в момент встановлення на входах схеми комбінації вхідних сигналів $f_2=0, f_1=1$ на виходах мають місце $Q=1, \bar{Q}=0$, тобто схема перебуває в стані “1”.

Через час τ на виходах схеми встановляться значення сигналів: $Q=1, \bar{Q}=0$, тобто схема залишиться в стані “1”. Дійсно:

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \end{pmatrix} \Rightarrow \left\{ \begin{array}{l} Q = \overline{0 \cdot 0} = 1 \\ \bar{Q} = \overline{1 \cdot 1} = 0 \end{array} \right\} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} - \text{стан “1”}.$$

Таким чином, комбінація вхідних сигналів $f_2=0, f_1=1$ встановлює схему в стан “1” незалежно від того, в якому стані вона перебувала. Комбінація вхідних сигналів $f_2=0, f_1=1$ запам’ятовується як цифра 1.

2. Нехай на входи схеми подано комбінацію сигналів $f_2=1, f_1=0$.

Вважатимемо, що на виходах схеми у цей час мають місце сигнали $Q=0, \bar{Q}=1$, тобто схема перебуває у стані “0”. Через час τ на виходах схеми встановляться значення сигналів: $Q=0, \bar{Q}=1$, тобто схема залишиться в стані “0”. Дійсно,

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \Rightarrow \left\{ \begin{array}{l} Q = \overline{1 \cdot 1} = 0 \\ \bar{Q} = \overline{0 \cdot 0} = 1 \end{array} \right\} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} - \text{стан “0”}.$$

Припустимо, що в момент встановлення на входах схеми комбінації вхідних сигналів $f_2=1, f_1=0$ на виходах мають місце сигнали $Q=1, \bar{Q}=0$, тобто схема перебуває в стані “1”.

Через час 2τ на виходах схеми остаточно встановляться нові значення сигналів $Q=0, \bar{Q}=1$, тобто схема перейде в стійкий стан “0”, хоча в процесі такого переходу (через час τ) на виходах схеми встановиться короткочасний нестійкий стан $Q=1, \bar{Q}=1$, який, однак, не є критичним. Дійсно:

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \end{pmatrix} \Rightarrow \left\{ \begin{array}{l} Q = \overline{1 \cdot 0} = 1 \\ \bar{Q} = \overline{0 \cdot 1} = 1 \end{array} \right\} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} - \text{короткочасний нестійкий стан (некритичний)}$$

$$\downarrow$$

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \end{pmatrix} \Rightarrow \left\{ \begin{array}{l} Q = \overline{1 \cdot 1} = 0 \\ \bar{Q} = \overline{0 \cdot 1} = 1 \end{array} \right\} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} - \text{стан “0”}.$$

Таким чином, комбінація вхідних сигналів $f_2=1, f_1=0$ встановлює схему в стан “0” незалежно від того, в якому стані вона перебувала. Комбінація вхідних сигналів $f_2=1, f_1=0$ запам’ятовується як цифра 0.

3. Нехай на входи подано комбінацію $f_2 = 1, f_1 = 1$. Тоді на виходах матиме місце підтвердження попереднього стану схеми, тобто $Q(t_{i+1}) = Q(t_i), \bar{Q}(t_{i+1}) = \bar{Q}(t_i)$. Дійсно

$$\begin{pmatrix} f_2 = 1 \\ f_1 = 1 \end{pmatrix} * \begin{pmatrix} Q \\ \bar{Q} \end{pmatrix} \Rightarrow \left\{ \begin{array}{l} Q = 1 \cdot \bar{Q} = Q \\ \bar{Q} = 1 \cdot Q = \bar{Q} \end{array} \right\} \Rightarrow \begin{pmatrix} Q \\ \bar{Q} \end{pmatrix}.$$

Отже, комбінація сигналів на входах $f_2 = 1, f_1 = 1$ підтверджує попередній стан схеми – якщо схема перебувала в стані “0”, то в наступний момент часу вона залишиться в стані “0”; якщо схема перебувала в стані “1”, то вона залишиться в стані “1”.

Таким чином, аналіз функціонування послідовної схеми на рис. 6.5 дає підстави для наступних висновків.

1. Схема є елементом пам'яті: комбінація вхідних сигналів $f_2 = 0, f_1 = 1$ запам'ятовується як цифра 1; комбінація $f_2 = 1, f_1 = 0$ запам'ятовується як цифра 0; комбінація $f_2 = 1, f_1 = 1$ підтверджує попередній стан схеми: $Q(t_{i+1}) = Q(t_i)$. Отже, схема здатна запам'ятовувати 1 біт інформації.
2. Швидкодія схеми становить 2τ .
3. Тривалість дії сигналів на входах f_2, f_1 схеми має бути не меншою за 2τ .
4. Робота схеми є стабільною, якщо на входи подавати комбінації 01, 10, 11, тобто якщо $f_2 \vee f_1 = 1$. Рівняння $f_2 \vee f_1 = 1$ називають *характеристичним рівнянням бістабільної схеми на елементах І-НЕ*.
5. Необхідною і достатньою умовою стабільної роботи бістабільної схеми на елементах І-НЕ є:

$$\begin{cases} f_2 \vee f_1 = 1, \\ (Q) \neq (\bar{Q}). \end{cases}$$

Функціонування бістабільної схеми на елементах І-НЕ в часі описує табл. 6.3.

Подавати комбінацію сигналів $f_2 = 0, f_1 = 0$ на входи схеми заборонено. Комбінація 01 на входах завжди встановлює схему в стан “1”, а комбінація 10 – в стан “0”. Комбінація 11 на входах підтверджує попередній стан схеми: якщо в момент часу t_i схема перебувала в стані “0” (тобто $Q(t_i) = 0$), то в наступний момент часу t_{i+1} вона підтвердить попередній стан, тобто $Q(t_{i+1}) = Q(t_i) = 0$; якщо ж в момент часу t_i схема перебувала в стані “1” (тобто $Q(t_i) = 1$), то в наступний момент часу t_{i+1} вона залишиться в цьому самому стані, тобто $Q(t_{i+1}) = Q(t_i) = 1$.

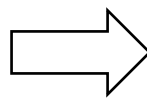
Таблицю 6.3 можна подати скорочено – як таблицю переходів бістабільної схеми (табл. 6.4).

На основі табл. 6.3, яка описує поведінку бістабільної схеми в часі, можна скласти таблицю функцій збудження бістабільної схеми (табл. 6.5), де \times означає довільне значення – 0 або 1.

Таблиця 6.3

Таблиця функціонування в часі бістабільної схеми на елементах I-HE

$f_2(t_i)$	$f_1(t_i)$	$Q(t_i)$	$Q(t_{i+1})$
0	0	0	–
0	0	1	–
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



Таблиця 6.4

Таблиця переходів бістабільної схеми на елементах I-HE

f_2	f_1	$Q(t_{i+1})$
0	0	–
0	1	1
1	0	0
1	1	$Q(t_i)$

Таблиця 6.5

Таблиця функцій збудження бістабільної схеми на елементах I-HE

$Q(t_i)$	$Q(t_{i+1})$	f_2	f_1
0	0	1	\times
0	1	0	1
1	0	1	0
1	1	\times	1

Табл. 6.5 показує, якими мають бути вхідні сигнали f_2 , f_1 щоб примусово переводити схему з поточного стану в будь-який інший.

Наприклад, для того, щоб примусово перевести схему зі стану “1” у стан “0” на входи схеми слід подати комбінацію $f_2 = 1, f_1 = 0$ та утримувати її протягом часу не меншим за 2τ . Або – для того, щоб схема, що перебуває у стані “0”, у наступний момент часу залишалась у цьому стані, необхідно на вході f_2 встановити $f_2 = 1$, а сигнал на вході f_1 може бути довільним (\times) – або нулем, або одиницею.

На основі таблиці функцій збудження схеми (табл. 6.5) записують рівняння збудження бістабільної схеми на елементах I-HE:

$$\begin{aligned}
 Q(t_{i+1}) &= \bar{Q}(t_i) \bar{f}_2 f_1 \vee Q(t_i) f_1 = f_1 (\bar{Q}(t_i) \bar{f}_2 \vee Q(t_i)) = \\
 &= f_1 (\bar{Q}(t_i) \vee Q(t_i)) (\bar{f}_2 \vee Q(t_i)) = f_1 (\bar{f}_2 \vee Q(t_i)).
 \end{aligned}$$

Запитання та завдання

1. Якою є швидкодія бістабільної схеми на елементах І-НЕ, якщо затримка поширення сигналів в елементах 2І-НЕ становить 20 нс?
2. Скільки петель присутні в структурі бістабільної схеми на елементах І-НЕ?
3. До яких наслідків може призвести поява на виходах бістабільної схеми на елементах І-НЕ значень $Q=1$, $\bar{Q}=1$?
4. Запишіть характеристичне рівняння бістабільної схеми на елементах І-НЕ.
5. Яку комбінацію сигналів заборонено подавати на входи бістабільної схеми на елементах І-НЕ?
6. Як довести, що бістабільна схема на елементах І-НЕ є елементом пам'яті, який здатний запам'ятовувати 1 біт інформації?
7. Якою має бути тривалість сигналів на входах f_2, f_1 бістабільної схеми на елементах І-НЕ?
8. Побудуйте таблицю переходів бістабільної схеми на елементах І-НЕ.
9. Побудуйте таблицю функцій збудження бістабільної схеми на елементах І-НЕ.
10. Яку комбінацію сигналів слід подати на входи f_2, f_1 бістабільної схеми на елементах І-НЕ, щоб примусово перевести її:
а) зі стану "0" у стан "1"; б) зі стану "1" у стан "0"?
11. Запишіть рівняння збудження бістабільної схеми на елементах І-НЕ.
12. Чи можливо побудувати бістабільну схему на основі двох елементів І?

6.3. Бістабільна схема на елементах АБО-НЕ

За принципом двоїстості (дуальності) (а функції І-НЕ та АБО-НЕ є взаємнодвоїстими) логічні елементи І-НЕ в послідовнісній схемі на рис. 6.5 можна замінити на логічні елементи АБО-НЕ. Тоді дістанемо послідовнісну схему на основі логічних елементів 2АБО-НЕ (рис. 6.6).

Як і в схемі на рис. 6.5 у даній схемі також присутні дві петлі.

Функціонування схеми на рис. 6.6 можна описати системою рівнянь

$$\begin{cases} Q = \overline{f_2 \vee \bar{Q}}, \\ \bar{Q} = \overline{f_1 \vee Q}, \end{cases}$$

де Q, \bar{Q} – виходи схеми, а f_2, f_1 – її входи.

Послідовнісна схема на рис. 6.6 є бістабільною, оскільки може перебувати в двох стійких станах.

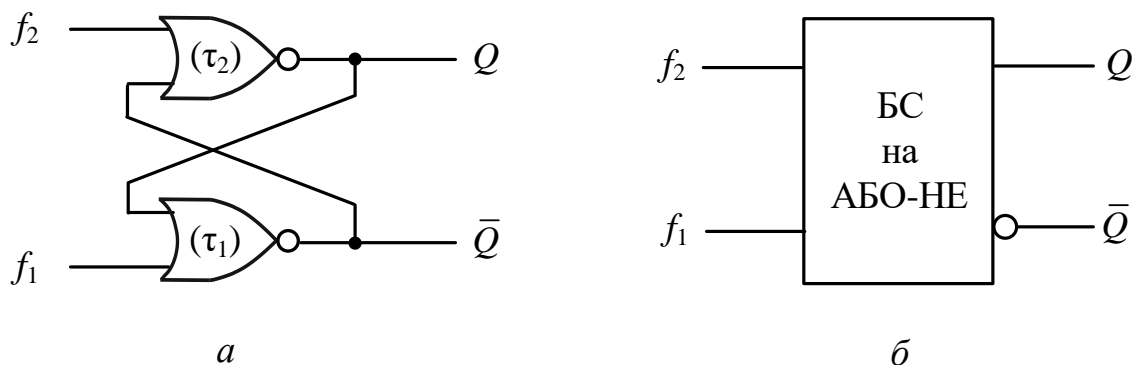


Рис. 6.6. Бістабільна схема на елементах 2АБО-НЕ:
a – структура схеми; *б* – умовне графічне позначення схеми

Для надійного функціонування схеми як елемента пам'яті необхідно, щоб значення на виходах схеми були протилежними, тобто $(Q) \neq (\bar{Q})$.

Вважатимемо, що схема на рис. 6.6 перебуває у стані "1", якщо на виходах схеми має місце $Q = 1, \bar{Q} = 0$, а у стані "0", – якщо $Q = 0, \bar{Q} = 1$.

Визначимо умови функціонування схеми, за яких вона може перебувати в двох стійких станах.

Твердження 6.3. Подавати комбінацію сигналів $f_2 = 1, f_1 = 1$ на входи схеми заборонено.

Доведення. Нехай на входах схеми встановлено сигнали $f_2 = 1, f_1 = 1$, а схема перебуває в одному зі стійких станів. Тоді на виходах схеми отримаємо

$$\begin{pmatrix} f_2 = 1 \\ f_1 = 1 \end{pmatrix} * \begin{pmatrix} Q \\ \bar{Q} \end{pmatrix} \Rightarrow \left\{ \begin{array}{l} Q = \overline{f_2 \vee \bar{Q}} = \overline{1 \vee \bar{Q}} = 0 \\ \bar{Q} = \overline{f_1 \vee Q} = \overline{1 \vee Q} = 0 \end{array} \right\} \Rightarrow \begin{pmatrix} Q = 0 \\ \bar{Q} = 0 \end{pmatrix},$$

тобто схема перейде у невизначений стан.

Крім того, поява на виходах значень $Q = 0, \bar{Q} = 0$ може спричинити в схемі коливальний процес, наприклад, якщо в наступному такті на входи схеми надійде комбінація сигналів $f_2 = 0, f_1 = 0$. Дійсно:

$$\begin{pmatrix} f_2 = 0 \\ f_1 = 0 \end{pmatrix} * \begin{pmatrix} 0 \\ 0 \end{pmatrix} \Rightarrow \left\{ \begin{array}{l} Q = \overline{0 \vee 0} = 1 \\ \bar{Q} = \overline{0 \vee 0} = 1 \end{array} \right\} \Rightarrow \begin{pmatrix} f_2 = 0 \\ f_1 = 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \end{pmatrix} \Rightarrow \left\{ \begin{array}{l} Q = \overline{0 \vee 1} = 0 \\ \bar{Q} = \overline{0 \vee 1} = 0 \end{array} \right\} \Rightarrow \dots$$

Отже, на виходах схеми з частотою $F = 1/(2\tau)$, де $\tau = \max(\tau_2, \tau_1)$, матимуть місце коливання:

$$\begin{pmatrix} Q \\ \bar{Q} \end{pmatrix} \Rightarrow \begin{pmatrix} 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 1 \end{pmatrix} \rightarrow \dots$$

Таким чином, комбінація сигналів $f_2 = 1, f_1 = 1$ на входах схеми є неприйнятною.

Твердження 6.4. Комбінації сигналів 00, 01, 10 на входах f_2, f_1 забезпечують стабільну роботу схеми.

Доведення.

1. Нехай на входах схеми встановлено сигнали $f_2 = 0, f_1 = 0$. Вважатимемо, що на виходах мають місце сигнали $Q = 0, \bar{Q} = 1$, тобто схема перебуває в стані “0”.

Через час τ на виходах схеми остаточно встановляться нові значення сигналів: $Q = 0, \bar{Q} = 1$, тобто схема залишиться в стані “0”. Дійсно,

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \Rightarrow \begin{cases} Q = \overline{0 \vee 1} = 0 \\ \bar{Q} = \overline{0 \vee 0} = 1 \end{cases} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \text{ – стан “0”}.$$

Припустимо, що в момент встановлення на входах схеми комбінації вхідних сигналів $f_2 = 0, f_1 = 0$ на виходах мають місце сигнали $Q = 1, \bar{Q} = 0$, тобто схема перебуває в стані “1”.

Через час τ на виходах схеми остаточно встановляться нові значення сигналів $Q = 1, \bar{Q} = 0$, тобто схема залишиться в стані “1”. Дійсно:

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \end{pmatrix} \Rightarrow \begin{cases} Q = \overline{0 \vee 0} = 1 \\ \bar{Q} = \overline{0 \vee 1} = 0 \end{cases} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ – стан “1”}.$$

Отже, комбінація сигналів $f_2 = 0, f_1 = 0$ на входах підтверджує попередній стан схеми: $Q(t_{i+1}) = Q(t_i), \bar{Q}(t_{i+1}) = \bar{Q}(t_i)$, тобто якщо схема перебувала в стані “0”, то в наступний момент часу вона залишиться в стані “0”; якщо ж схема перебувала в стані “1”, то вона залишиться в стані “1”.

2. Нехай на входи схеми подано комбінацію сигналів $f_2 = 0, f_1 = 1$.

Вважатимемо, що на виходах схеми у цей час мають місце сигнали $Q = 0, \bar{Q} = 1$, тобто схема перебуває у стані “0”.

Через час 2τ на виходах схеми встановляться нові значення сигналів: $Q = 1, \bar{Q} = 0$, тобто схема перейде в стійкий стан “1”, хоча в процесі такого переходу (через час τ) на виходах схеми встановиться короткочасний нестійкий стан $Q = 0, \bar{Q} = 0$, який, однак, не є критичним. Дійсно:

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \Rightarrow \begin{cases} Q = \overline{0 \vee 1} = 0 \\ \bar{Q} = \overline{1 \vee 0} = 0 \end{cases} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \text{ – короткочасний нестійкий стан (некритичний)}$$

$$\downarrow$$

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} * \begin{pmatrix} 0 \\ 0 \end{pmatrix} \Rightarrow \begin{cases} Q = \overline{0 \vee 0} = 1 \\ \bar{Q} = \overline{1 \vee 0} = 0 \end{cases} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ – стан “1”}.$$

Припустимо, що в момент встановлення на входах схеми комбінації вхідних сигналів $f_2 = 0, f_1 = 1$ на виходах мають місце сигнали $Q = 1, \bar{Q} = 0$, тобто схема перебуває в стані “1”.

Через час τ на виходах схеми остаточно встановляться нові значення сигналів $Q=1$, $\bar{Q}=0$, тобто схема залишиться в стані “1”. Дійсно,

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \end{pmatrix} \Rightarrow \left\{ \begin{array}{l} Q = \overline{0 \vee 0} = 1 \\ \bar{Q} = \overline{1 \vee 1} = 0 \end{array} \right\} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} - \text{стан “1”}.$$

Таким чином, комбінація вхідних сигналів $f_2=0$, $f_1=1$ встановлює схему в стан “1” незалежно від того, в якому стані вона перебувала. Комбінація вхідних сигналів $f_2=0$, $f_1=1$ запам'ятовується як цифра 1.

3. Нехай на входи схеми подано комбінацію сигналів $f_2=1$, $f_1=0$.

Вважатимемо, що на виходах схеми у цей час мають місце сигнали $Q=0$, $\bar{Q}=1$, тобто схема перебуває у стані “0”.

Через час τ на виходах схеми остаточно встановляться нові значення сигналів $Q=0$, $\bar{Q}=1$, тобто схема залишиться в стані “0”. Дійсно,

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \Rightarrow \left\{ \begin{array}{l} Q = \overline{1 \vee 1} = 0 \\ \bar{Q} = \overline{0 \vee 0} = 1 \end{array} \right\} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} - \text{стан “0”}.$$

Нехай в момент надходження на входи сигналів $f_2=1$, $f_1=0$ на виходах схеми мають місце сигнали $Q=1$, $\bar{Q}=0$, тобто схема перебуває в стані “1”.

Через час 2τ на виходах схеми встановляться нові значення сигналів $Q=0$, $\bar{Q}=1$, тобто схема перейде в стійкий стан “1”, хоча в процесі такого переходу (через час τ) на виходах схеми встановиться короткочасний нестійкий стан $Q=0$, $\bar{Q}=0$, який не є критичним. Дійсно:

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \end{pmatrix} \Rightarrow \left\{ \begin{array}{l} Q = \overline{1 \vee 0} = 0 \\ \bar{Q} = \overline{0 \vee 1} = 0 \end{array} \right\} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} - \text{короткочасний нестійкий стан (некритичний)}$$

$$\downarrow$$

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} * \begin{pmatrix} 0 \\ 0 \end{pmatrix} \Rightarrow \left\{ \begin{array}{l} Q = \overline{1 \vee 0} = 0 \\ \bar{Q} = \overline{0 \vee 0} = 1 \end{array} \right\} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} - \text{стан “0”}.$$

Отже, комбінація сигналів $f_2=1$, $f_1=0$ встановлює схему в стан “0” ($Q(t_{i+1})=0$, $\bar{Q}(t_{i+1})=1$) незалежно від того, в якому стані вона перебувала в попередній момент часу. Тобто, комбінація вхідних сигналів $f_2=1$, $f_1=0$ запам'ятовується як цифра 0.

Таким чином, аналіз послідовнісної схеми на елементах АБО-НЕ дає підстави зробити такі висновки.

1. Схема є елементом пам'яті: комбінація вхідних сигналів $f_2=0$, $f_1=1$ запам'ятовується як цифра 1; комбінація $f_2=1$, $f_1=0$ запам'ятовується як цифра 0, а комбінація $f_2=0$, $f_1=0$ підтверджує попередній стан схеми: $Q(t_{i+1})=Q(t_i)$. Отже, схема здатна запам'ятовувати 1 біт інформації.

2. Швидкодія схеми становить 2τ .

3. Тривалість дії сигналів на входах схеми має бути не меншою за 2τ .

4. Робота схеми є стабільною, якщо на входи подавати комбінації 00, 01, 10, тобто якщо $f_2 \cdot f_1 = 0$. Рівняння $f_2 \cdot f_1 = 0$ називають *характеристичним рівнянням бістабільної схеми на елементах АБО-НЕ*. Подавати комбінацію сигналів $f_2 = 1, f_1 = 1$ на входи схеми заборонено.

5. Необхідною і достатньою умовою стабільної роботи бістабільної схеми на елементах АБО-НЕ є:

$$\begin{cases} f_2 \cdot f_1 = 0 \\ (Q) \neq (\bar{Q}). \end{cases}$$

Функціонування бістабільної схеми на елементах АБО в часі описує табл. 6.6.

Подавати комбінацію сигналів $f_2 = 1, f_1 = 1$ на входи схеми заборонено.

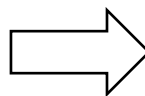
Комбінація 01 на входах завжди встановлює схему в стан "1", а комбінація 10 – в стан "0". Комбінація 00 на входах підтверджує попередній стан схеми: якщо в момент часу t_i схема перебувала в стані "0" (тобто $Q(t_i) = 0$), то в наступний момент часу t_{i+1} вона залишиться у цьому самому стані, тобто $Q(t_{i+1}) = Q(t_i) = 0$; якщо ж в момент часу t_i схема перебувала в стані "1" (тобто $Q(t_i) = 1$), то в наступний момент часу t_{i+1} вона залишиться в цьому самому стані, тобто $Q(t_{i+1}) = Q(t_i) = 1$.

Таблицю 6.6 можна подати скорочено – як таблицю переходів бістабільної схеми на елементах АБО-НЕ (табл. 6.7).

Таблиця 6.6

Таблиця функціонування в часі бістабільної схеми на елементах АБО-НЕ

$f_2(t_i)$	$f_1(t_i)$	$Q(t_i)$	$Q(t_{i+1})$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	–
1	1	1	–



Таблиця 6.7

Таблиця переходів бістабільної схеми на елементах АБО-НЕ

f_2	f_1	$Q(t_{i+1})$
0	0	$Q(t_i)$
0	1	1
1	0	0
1	1	–

На основі табл. 6.6, яка описує поведінку бістабільної схеми в часі, можна скласти таблицю функцій збудження бістабільної схеми (табл. 6.8),

яка показує, якими мають бути вхідні сигнали f_2, f_1 , щоб примусово переводити схему з поточного стану в будь-який інший.

Таблиця 6.8
Таблиця функцій збудження
бістабільної схеми на
елементах АБО-НЕ

$Q(t_i)$	$Q(t_{i+1})$	f_2	f_1
0	0	×	0
0	1	0	1
1	0	1	0
1	1	0	×

Наприклад, для того, щоб примусово перевести схему зі стану “0” у стан “1” на входи схеми слід подати комбінацію $f_2 = 0, f_1 = 1$ та утримувати її не менше ніж 2τ . Або – для того, щоб схема, що перебуває в стані “1”, у наступний момент часу залишилась у цьому самому стані, необхідно на вході f_2 встановити $f_2 = 0$, а сигнал на вході f_1 може бути довільним (×) – або нулем, або одиницею.

На основі таблиці функцій збудження схеми (табл. 6.8) отримують рівняння збудження бістабільної схеми на елементах АБО-НЕ:

$$\begin{aligned} Q(t_{i+1}) &= \bar{Q}(t_i) \bar{f}_2 f_1 \vee Q(t_i) \bar{f}_2 = \bar{f}_2 (\bar{Q}(t_i) f_1 \vee Q(t_i)) = \\ &= \bar{f}_2 (\bar{Q}(t_i) \vee Q(t_i)) (f_1 \vee Q(t_i)) = \bar{f}_2 (f_1 \vee Q(t_i)). \end{aligned}$$

Порівняємо між собою бістабільні схеми на елементах І-НЕ (рис. 6.5) та АБО-НЕ (рис. 6.6).

Обидві бістабільні схеми – як на елементах І-НЕ, так і на елементах АБО-НЕ, однаково запам’ятовують значення вхідних сигналів f_2, f_1 : в обох бістабільних схемах значення $f_2 = 0, f_1 = 1$ запам’ятовується як цифра 1 ($Q = 1, \bar{Q} = 0$), а значення $f_2 = 1, f_1 = 0$ – як цифра 0.

Підтвердження попереднього стану (зберігання) в бістабільних схемах забезпечується по різному: в бістабільній схемі на елементах І-НЕ – комбінацією $f_2 = 1, f_1 = 1$, а в бістабільній схемі на елементах АБО-НЕ – комбінацією $f_2 = 0, f_1 = 0$.

Забороненими комбінаціями вхідних сигналів є: для бістабільної схеми на елементах І-НЕ – комбінація $f_2 = 0, f_1 = 0$; для бістабільної схеми на елементах АБО-НЕ – комбінація $f_2 = 1, f_1 = 1$.

Швидкодія обох бістабільних схем однакова – 2τ .

Як елемент пам’яті обидві бістабільні схеми – рівноцінні.

Бістабільна схема обов’язково повинна мати два входи і два виходи – тому мінімальна кількість логічних елементів, що мають входити до складу схеми, – два.

Біт запам'ятовується як комбінація значень двох вхідних сигналів – f_2 та f_1 . Згенерувати 1 біт за допомогою значень одного сигналу – неможливо.

Бістабільні схеми є основою для побудови тригерних пристроїв (тригерів).

Серцевиною тригера є бістабільна схема.

Бістабільна схема є найпростішим тригером.

На основі тригерів будують оперативну пам'ять комп'ютера, а також регістри процесора та лічильники.

Запитання та завдання

1. Якою системою рівнянь описують функціонування бістабільної схеми на елементах АБО-НЕ?
2. Чому на входи бістабільної схеми на елементах АБО-НЕ заборонено подавати комбінацію вхідних сигналів $f_2 = 1, f_1 = 1$?
3. За яких умов у бістабільній схемі на елементах АБО-НЕ може розпочатись коливальний процес та з якою частотою?
4. Якою є швидкодія бістабільної схеми на елементах АБО-НЕ, якщо час затримки поширення сигналів в елементах 2АБО-НЕ становить 17 нс?
5. Яку комбінацію вхідних сигналів бістабільна схема на елементах АБО-НЕ запам'ятовує як: а) цифру 1; б) цифру 0?
6. Доведіть, що бістабільна схема на елементах АБО-НЕ є елементом пам'яті, який здатний запам'ятовувати 1 біт інформації.
7. Якою має бути тривалість сигналів на входах бістабільної схеми на елементах АБО-НЕ, щоб забезпечувалась стабільна робота схеми?
8. Запишіть характеристичне рівняння бістабільної схеми на елементах АБО-НЕ.
9. Якою є таблиця переходів бістабільної схеми на елементах АБО-НЕ?
10. Побудуйте таблицю функцій збудження бістабільної схеми на елементах АБО-НЕ.
11. Яку комбінацію сигналів слід подати на входи бістабільної схеми на елементах АБО-НЕ, щоб примусово перевести її:
а) зі стану "1" у стан "0"; б) зі стану "0" у стан "1"?
12. Порівняйте між собою бістабільну схему на елементах І-НЕ та бістабільну схему на елементах АБО-НЕ щодо: а) запам'ятовування значень вхідних сигналів f_2, f_1 ; б) підтвердження попереднього стану схеми; в) дозволених та забороненої комбінації вхідних сигналів.
13. Чи можливо побудувати бістабільну схему, що складається з двох елементів АБО?
14. В яких функціональних вузлах комп'ютера використовують бістабільні схеми?

7.

СИНТЕЗ ТРИГЕРІВ

В обчислювальних засобах як запам'ятовувальні елементи використовують тригерні схеми (тригери), які можуть перебувати у двох стійких станах. Тригер (тригерна схема) здатний запам'ятовувати і необмежено довго зберігати 1 біт інформації.

7.1. Класифікація тригерів

Тригером називають схему (вузол), побудовану на основі логічних елементів, яка може перебувати у двох стійких станах.

До складу тригера входить бістабільна схема (БС), яку називають також запам'ятовувальним елементом, та схема керування (СК) (рис. 7.1).

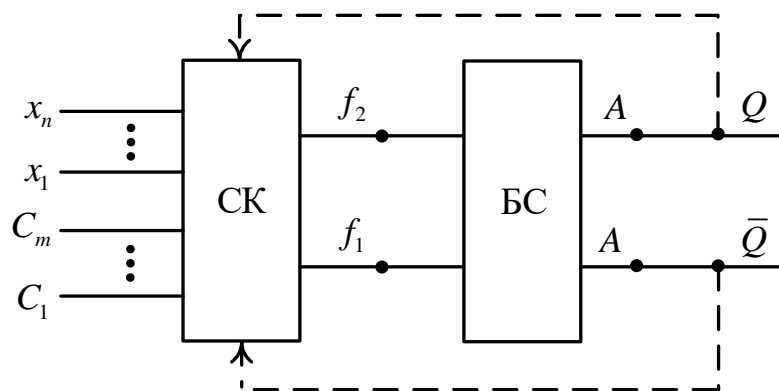


Рис. 7.1. Узагальнена структура тригера

Схема керування є комбінаційною схемою з двома виходами. На входи тригера (схеми керування) надходять інформаційні сигнали x_n, \dots, x_1 та сигнали синхронізації C_m, \dots, C_1 . Схема керування генерує функції збудження f_2, f_1 бістабільної схеми, які забезпечують перемикання (перехід) тригера з одного стану в інший.

Тригер має прямиий Q та інверсний \bar{Q} виходи. У деяких схемах тригерів зворотні зв'язки, показані пунктиром, можуть бути відсутні.

Для забезпечення надійного перемикання у схемах деяких тригерів у точках A необхідно розміщувати елементи затримки. Для затримки сигналів у точках A у разі потреби використовують додаткову бістабільну схему – такого самого типу як і основна БС тригера.

Можливі й простіші варіанти тригерних схем, коли у структурі тригера відсутня схема керування (при цьому функції схеми керування можуть бути реалізовані логічними елементами, що входять до складу бістабільної схеми), або коли тригер не має входів синхронізації.

На практиці переважно використовують тригери, які мають один або два інформаційні входи (рис. 7.2).

Тригери, що мають один вхід синхронізації (C), називають *однотактними*, тригери з двома входами синхронізації називають *двотактними*.

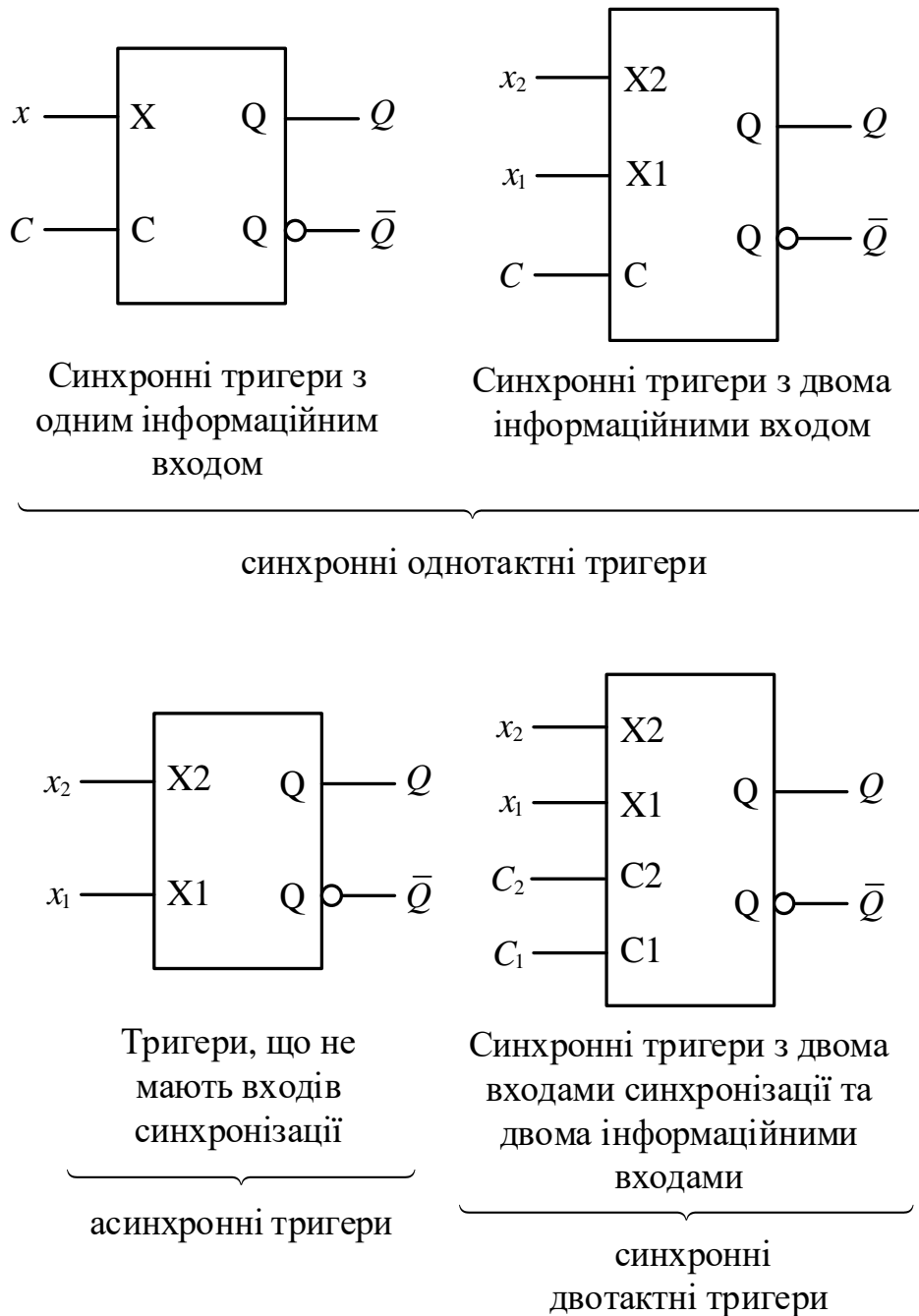


Рис. 7.2. Використовувані на практиці види тригерів

Тригери класифікують за видом функції переходів тригера та за способом запису (приймання) інформації в тригер.

Класифікація тригерів за функцією переходів

Функціонування тригера описують таблицею переходів тригера. *Таблиця переходів* тригера задає закон функціонування тригера. Вона відтворює функцію переходів тригера, яку узагальнено записують так:

$$Q(t_{i+1}) = \lambda(Q(t_i), X(t_i)),$$

де λ – функція переходів тригера, $Q(t_i)$ – значення вихідного сигналу (стан тригера) в момент часу t_i , $Q(t_{i+1})$ – значення вихідного сигналу (стан тригера) в наступний момент часу, $X(t_i)$ – значення інформаційних сигналів у момент часу t_i .

За видом функції переходів (таблиця переходів) розрізняють RS-, R-, S-, E-, D-, DV-, T-, JK-тригери (табл. 7.1 – 7.8) та інші.

Таблиця 7.1
Таблиця переходів
RS-тригера

$S(t_i)$	$R(t_i)$	$Q(t_{i+1})$
0	0	$Q(t_i)$
0	1	0
1	0	1
1	1	–

Таблиця 7.2
Таблиця переходів
R-тригера

$S(t_i)$	$R(t_i)$	$Q(t_{i+1})$
0	0	$Q(t_i)$
0	1	0
1	0	1
1	1	0

Таблиця 7.3
Таблиця переходів
S-тригера

$S(t_i)$	$R(t_i)$	$Q(t_{i+1})$
0	0	$Q(t_i)$
0	1	0
1	0	1
1	1	1

Таблиця 7.4
Таблиця переходів
E-тригера

$S(t_i)$	$R(t_i)$	$Q(t_{i+1})$
0	0	$Q(t_i)$
0	1	0
1	0	1
1	1	$Q(t_i)$

Таблиця 7.5
Таблиця переходів
D-тригера

$D(t_i)$	$Q(t_{i+1})$
0	0
1	1

Таблиця 7.6
Таблиця переходів
DV-тригера

$V(t_i)$	$D(t_i)$	$Q(t_{i+1})$
0	0	$Q(t_i)$
0	1	$Q(t_i)$
1	0	0
1	1	1

Таблиця 7.7
Таблиця переходів
Т-тригера

$T(t_i)$	$Q(t_{i+1})$
0	$Q(t_i)$
1	$\bar{Q}(t_i)$

Таблиця 7.8
Таблиця переходів
JK-тригера

$J(t_i)$	$K(t_i)$	$Q(t_{i+1})$
0	0	$Q(t_i)$
0	1	0
1	0	1
1	1	$\bar{Q}(t_i)$

RS-тригер має два інформаційні входи – S (set) та R (reset). Стан тригера є невизначеним ($Q(t_{i+1}) = -$), якщо на входи подати сигнали $R=1$, $S=1$. Ця комбінація сигналів для RS-тригера є забороненою.

При $R=0$, $S=0$ тригер не змінює свого стану. Для встановлення тригера в “0” на входи необхідно подати $R=1$, $S=0$; для встановлення тригера в “1” – $R=0$, $S=1$.

R-, S-, E-тригер відрізняються від RS-тригера способом встановлення стану тригера при дії на інформаційні входи тригера комбінації сигналів $R=1$, $S=1$.

При цій комбінації сигналів: R-тригер встановлюється в “0” (reset), S-тригер – в “1” (set), E-тригер підтверджує попередній стан: $Q(t_{i+1}) = Q(t_i)$.

При дії на входи тригера інших комбінацій сигналів (00, 01, 10) ці тригери функціонують так само, як і RS-тригер.

D-тригер має лише один інформаційний вхід – D (delay – затримати). D-тригер називають також *тригером затримки*. Тригер запам’ятовує (фіксує) сигнал, що надійшов у момент часу t_i на вхід D, тобто виконує функцію $Q(t_{i+1}) = D(t_i)$: якщо на вхід тригера надходить нульовий сигнал, то запам’ятовується 0, якщо на вхід надходить одиничний сигнал, то запам’ятовується 1.

DV-тригер має два інформаційні входи – D та V. При $V(t_i) = 1$ тригер працює як D-тригер, а при $V(t_i) = 0$ не змінює свого стану.

T-тригер, як і D-тригер, має лише один інформаційний вхід – T (toggle – бурюкатись, кувыркатись – рос.). Такий тригер при $T(t_i) = 0$ не змінює свого стану, а при $T(t_i) = 1$ щоразу (у наступний момент часу) змінює свій стан на протилежний – $Q(t_{i+1}) = \bar{Q}(t_i)$. T-тригер виконує додавання вхідних сигналів, що надходять на вхід T, за модулем два, тому його також називають *лічильним тригером*.

JK-тригер (J – jump (встановлювати), K – kill (скидати, гасити)) є тригером з двома інформаційними входами. Але, на відміну від RS-тригера, він не має заборонених комбінацій вхідних сигналів. При дії на входи JK-тригера комбінацій сигналів 00, 01, 10 він функціонує як RS-тригер, але при

надходженні на входи тригера комбінації $J(t_i) = 1, K(t_i) = 1$ він змінює свій стан на протилежний ($Q(t_{i+1}) = \bar{Q}(t_i)$), тобто спрацьовує як Т-тригер при дії на його вхід сигналу $T(t_i) = 1$.

Отже, при надходженні на входи JK-тригера комбінації $J = 1, K = 1$ він також виконує функцію лічби за модулем два.

Кількість типів тригерів надзвичайно велика. Наприклад, якщо тригер має один інформаційний вхід (можливі значення сигналу на вході тригера: $x=0$ та $x=1$; можливі значення на виході тригера в момент $Q(t_{i+1})$: $0, 1, Q(t_i), \bar{Q}(t_i)$ та невизначений стан), то можна одержати $5^2 = 25$ типів тригерів. Якщо тригер має два інформаційні входи, то можна отримати $5^4 = 5^{2^2} = 625$ типів тригерів. У загальному випадку при n інформаційних входах можна одержати 5^{2^n} типів тригерів.

На практиці використовують лише невелику кількість типів тригерів (табл. 7.1-7.8).

Класифікація тригерів за способом запису інформації

Класифікація тригерів за способом запису (приймання) інформації в тригер характеризує перебіг процесу перемикання тригера з одного стану в інший (рис. 7.3).

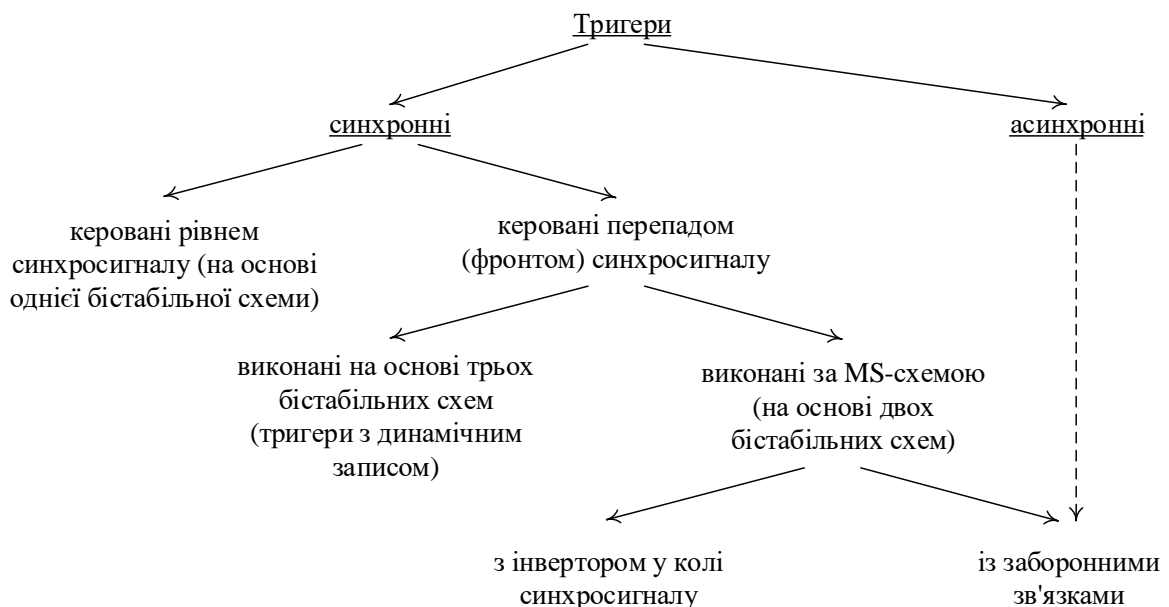


Рис. 7.3. Класифікація тригерів за способом запису інформації

Тригери поділяють на синхронні та асинхронні.

Асинхронні тригери не мають входів синхронізації, тому запис (приймання) інформації в такі тригери здійснюється безпосередньо

надходженням інформаційних сигналів на входи тригера. Такі тригери працюють в режимі очікування надходження вхідних сигналів.

Синхронні тригери мають входи синхронізації (тактові входи). На практиці використовують синхронні тригери з одним або двома входами синхронізації (синхровходами).

Тригери з двома синхровходами (C_2, C_1) називають *двотактними*, а з одним входом синхронізації (C – clock) – *однотактними*. Далі розглядатимемо лише однотактні тригери.

Запис інформації в синхронний тригер здійснюється, коли діє сигнал синхронізації.

Розрізняють синхронні тригери, *керовані рівнем синхросигналу*, та тригери, *керовані перепадом (фронтом) синхросигналу* (рис. 7.4).

Тригери, керовані перепадом (фронтом) тактового сигналу, містять у своєму складі дві або три бістабільні схеми, тому їх називають також *тригерами з внутрішньою затримкою*.

Тригери, керовані рівнем тактового сигналу, перемикаються відповідно до таблиці переходів тригера при появі тактового сигналу на вході C , і процес перемикання (переходу тригера з одного стану в інший) триває протягом усього проміжку часу поки на вході C діє синхросигнал з тривалістю τ_p (рис. 7.4), τ_p – тривалість рівня синхросигналу.

Тригери, керовані перепадом (фронтом) сигналу, переходять з одного стану в інший лише в момент зміни синхросигналу з 0 в 1 (передній фронт) або з 1 в 0 (задній фронт). Час перемикання цього виду тригерів набагато менший за час перемикання тригерів, керованих рівнем синхросигналу ($\tau_n \ll \tau_p$), що істотно підвищує рівень надійності перемикання з одного стану в інший, τ_n – тривалість перепаду (фронту) синхросигналу.

Але тригери з внутрішньою затримкою структурно складніші – до їх складу входять дві або три бістабільні схеми, тоді як тригери, керовані рівнем тактового сигналу, включають лише одну бістабільну схему.

Запитання та завдання

1. Яка роль відводиться тригерам в засобах обчислювальної техніки?
2. Дайте визначення тригера та назвіть структурні компоненти тригера.
3. Які тригери називають однотактними, а які – двотактними?
4. Які тригери називають асинхронними, а які – синхронними?
5. За якими ознаками класифікують тригери?
6. Запишіть аналітично узагальнену функцію переходів тригера.
7. Наведіть таблиці переходів RS-, D-, T- та JK-тригера.
8. Чим відрізняються R-, S- та E-тригер від RS-тригера?
9. Чим відрізняється DV-тригер від D-тригера?

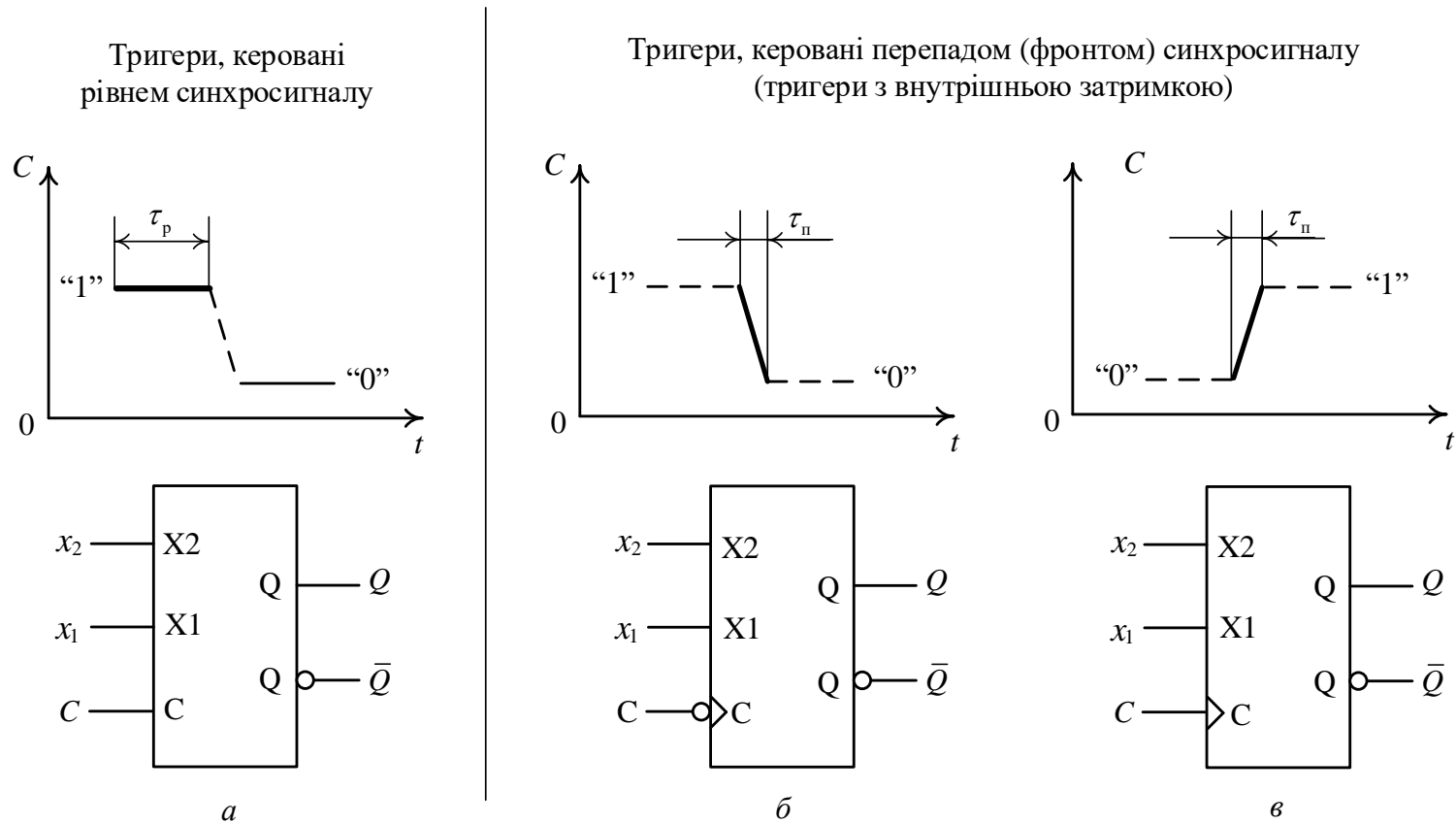


Рис. 7.4. Синхронізація (тактування) процесу запису (приймання) інформації в тригер
a – рівнем тактового сигналу; *б*, *в* – перепадом (фронтом) тактового сигналу
 (*б* – за заднім фронтом, *в* – за переднім фронтом)

Примітка. τ_p – тривалість рівня синхросигналу, τ_n – тривалість перепаду (фронту) синхросигналу ($\tau_n \ll \tau_p$).

10. Яка відмінність між RS- та JK-тригером?
11. Які тригери здатні реалізовувати мікрооперацію лічби?
12. Які два способи перемикання з одного стану в інший використовуються в тригерах?
13. Яку перевагу мають тригери, керовані фронтом синхросигналу?

7.2. Асинхронні тригери

Асинхронні тригери не мають синхровходів. Запис (приймання) інформації в асинхронні тригери здійснюється безпосередньо надходженням сигналів на інформаційні входи тригера.

Спочатку розглянемо асинхронні тригери з двома інформаційними входами (x_2, x_1) на основі однієї бістабільної схеми (рис. 7.5).

У структурі асинхронного тригера відсутня схема керування.

Бістабільна схема (БС) на елементах І-НЕ та бістабільна схема на елементах АБО-НЕ є елементами пам'яті, і, отже найпростішими тригерами, що мають два інформаційні входи.

Оскільки бістабільна схема не має синхровходів, то вона є асинхронним тригером.

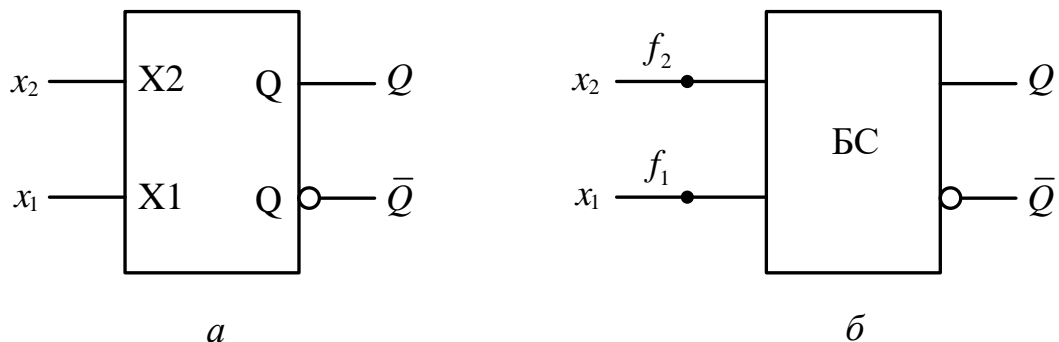


Рис. 7.5. Узагальнене позначення асинхронного тригера з двома інформаційними входами: *a* – умовне графічне позначення тригера; *б* – структура асинхронного тригера

Порівняймо між собою таблицю переходів RS-тригера та таблицю переходів бістабільної схеми на елементах АБО-НЕ (рис. 7.6). Якщо в таблиці переходів бістабільної схеми на елементах АБО-НЕ (таблиця 2) переставити місцями другий та третій рядки (таблиця 3), а потім переставити стовпці f_2 та f_1 , то отримаємо таблицю 4, яка співпадає з таблицею переходів RS-тригера (таблицею 1).

З таблиці 1 та таблиці 4 на рис. 7.6 бачимо, що бістабільна схема на елементах АБО-НЕ може виконувати функцію RS-тригера, якщо $f_1 \equiv S$, а $f_2 \equiv R$.

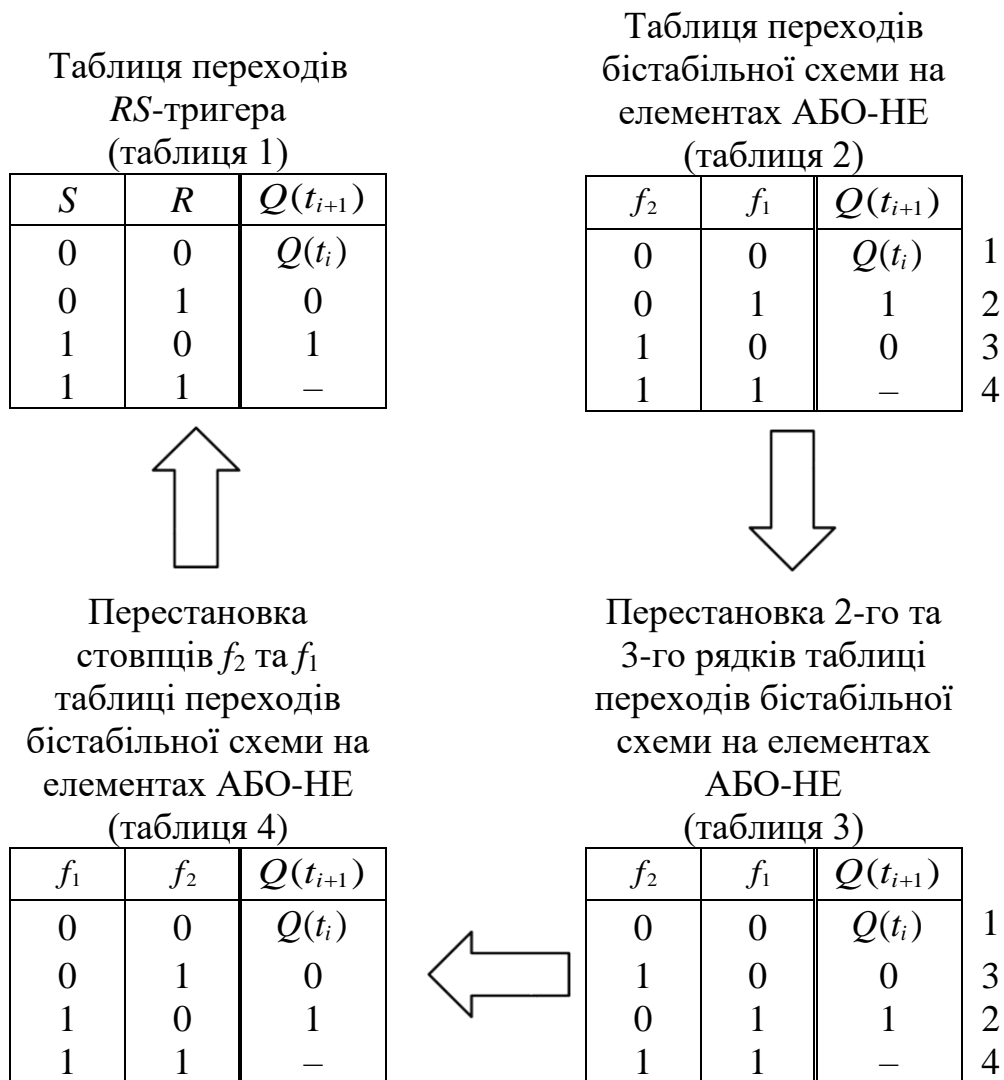


Рис. 7.6. Порівняння таблиці переходів RS-тригера та таблиці переходів бістабільної схеми на елементах АБО-НЕ

Таблиця 7.9
Таблиця переходів
асинхронного
RS-тригера на
елементах АБО-НЕ

S	R	$Q(t_{i+1})$
0	0	$Q(t_i)$
0	1	0
1	0	1
1	1	—

Отже, бістабільна схема на елементах АБО-НЕ, у якій вхід f_2 позначити як R (reset), а вхід f_1 позначити як S (set), є найпростішим тригером (рис. 7.7), функціонування якого задає табл. 7.9. Такий тригер не має входів синхронізації, і запис (приймання) інформації в тригер відбувається асинхронно – безпосередньо надходженням сигналів на інформаційні входи S і R .

Комбінація вхідних сигналів $S = 1, R = 1$ для цього тригера є забороненою, отже, на входах тригера має виконуватись умова $S \cdot R = 0$, яка є *характеристичним рівнянням RS-тригера*.

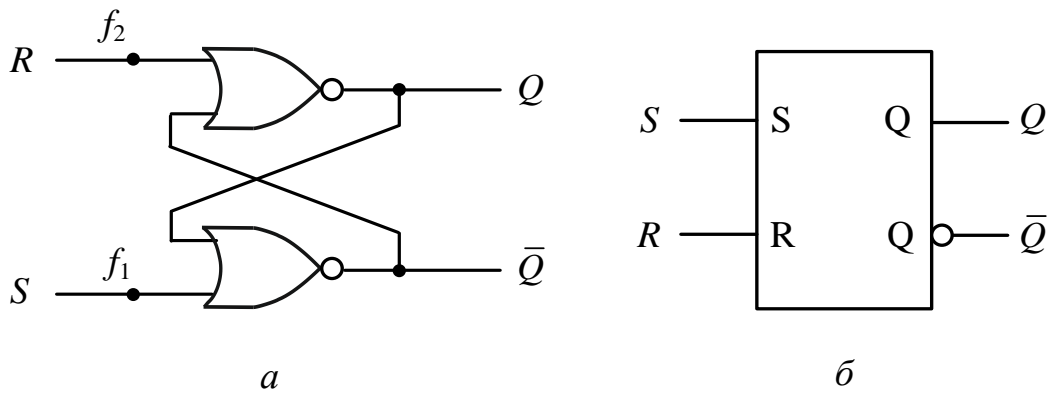
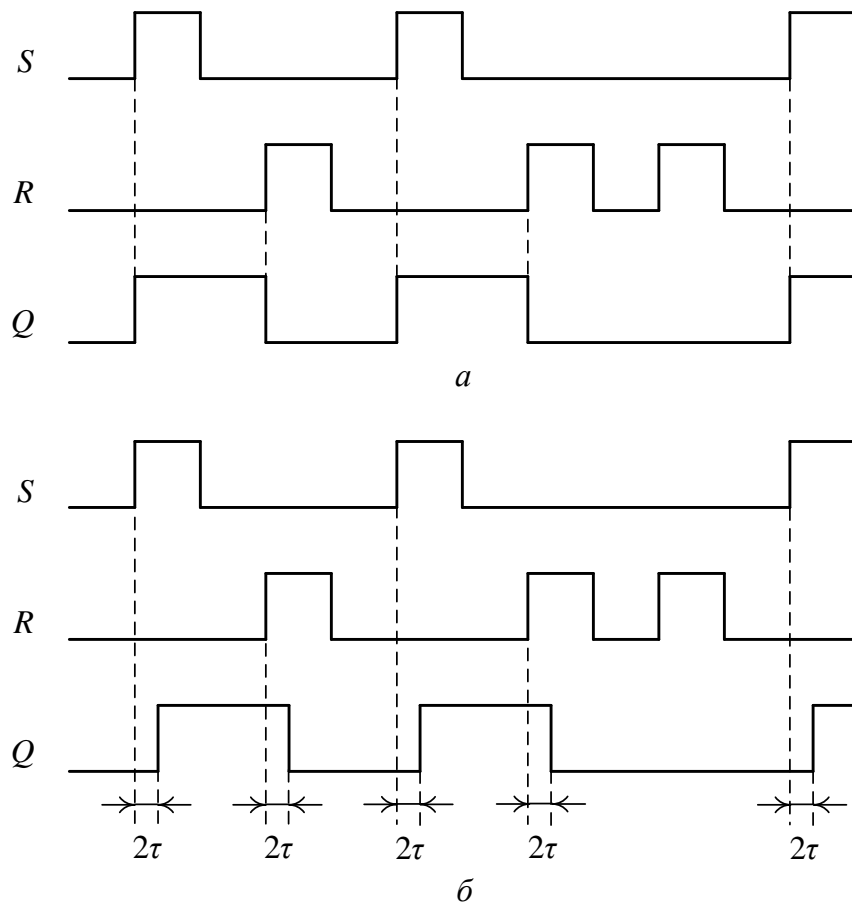


Рис. 7.7. Асинхронний RS-тригер на елементах АБО-НЕ:
a – функціональна схема; *б* – умовне графічне позначення тригера

Комбінація вхідних сигналів $S = 1, R = 0$ встановлює тригер у стан “1” ($Q = 1$) (рис. 7.8), а комбінація $S = 0, R = 1$ – у стан “0” ($Q = 0$). Комбінація вхідних сигналів $S = 0, R = 0$ не змінює стану тригера ($Q(t_{i+1}) = Q(t_i)$).

Швидкодія асинхронного RS-тригера становить 2τ (рис. 7.8б).



τ – затримка сигналу елементом 2АБО-НЕ,
 2τ – час перемикання тригера

Рис. 7.8. Часова діаграма роботи асинхронного RS-тригера на елементах АБО-НЕ:
a – ідеальна; *б* – реальна

Порівняймо між собою таблицю переходів RS-тригера та таблицю переходів бістабільної схеми на елементах І-НЕ (рис. 7.9).

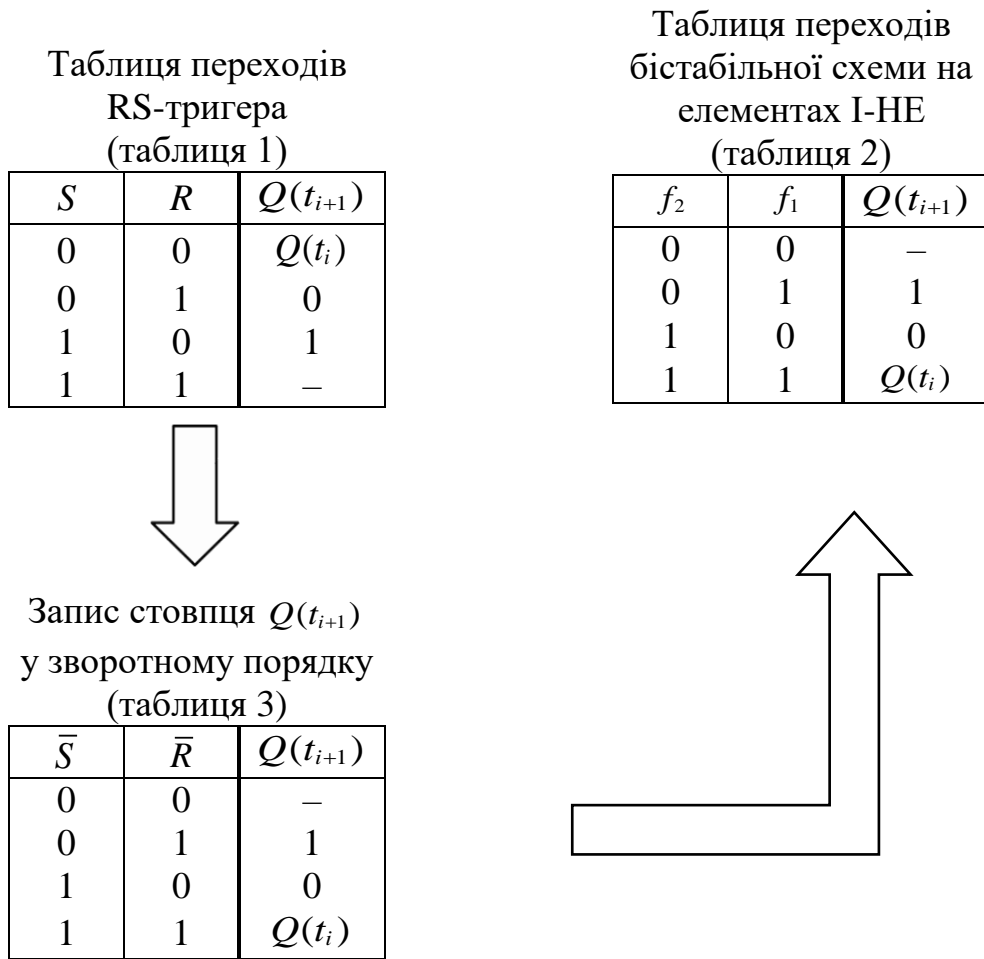


Рис. 7.9. Порівняння таблиці переходів RS-тригера та таблиці переходів бістабільної схеми на елементах І-НЕ

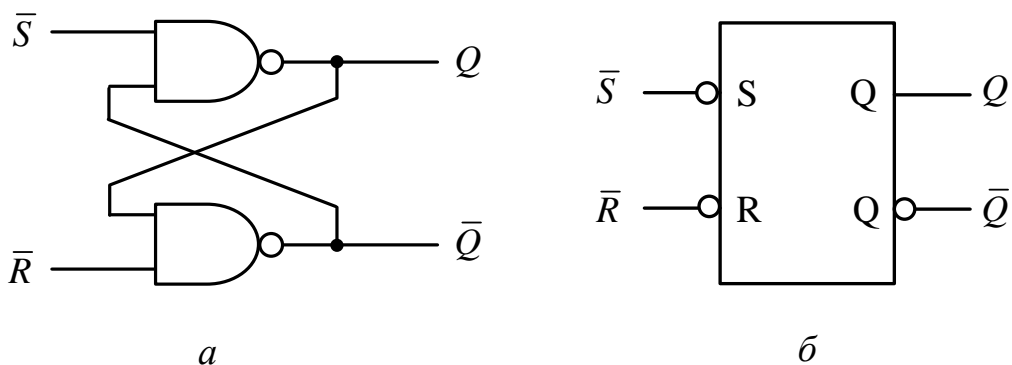


Рис. 7.10. Асинхронний RS-тригер на елементах І-НЕ:
a – функціональна схема; *б* – умовне графічне позначення

Запишемо стовпець $Q(t_{i+1})$ (функцію) у зворотному порядку (таблиця 3). Але запис стовпця $Q(t_{i+1})$ у зворотному порядку еквівалентний

інвертуванню змінних функції, тобто S і R . Тому відповідні стовпці в таблиці 3 слід назвати \bar{S} і \bar{R} .

Але таблиця 3 співпадає з таблицею 2. Отже, бістабільна схема на елементах І-НЕ може виконувати функцію RS-тригера, якщо $f_2 \equiv \bar{S}$, а $f_1 \equiv \bar{R}$.

Тому, бістабільна схема на елементах І-НЕ, у якій вхід f_2 позначений як \bar{S} , а вхід f_1 – як \bar{R} , є найпростішим тригером. Такий тригер не має входів синхронізації, і запис (приймання) інформації в тригер відбувається асинхронно – безпосереднім надходженням сигналів на інформаційні входи \bar{S} та \bar{R} .

У RS-тригері на елементах І-НЕ активним рівнем інформаційних сигналів є рівень логічного нуля: при $\bar{S} = 0$ ($\bar{R} = 1$) тригер встановлюється в стан “1” (set); при $\bar{R} = 0$ ($\bar{S} = 1$) тригер встановлюється в стан “0” (reset).

Таблиця переходів асинхронного RS-тригера на елементах І-НЕ

\bar{S}	\bar{R}	$Q(t_{i+1})$
0	0	–
0	1	1
1	0	0
1	1	$Q(t_i)$

Комбінація вхідних сигналів $\bar{S} = 0$, $\bar{R} = 0$ для цього тригера є забороненою, отже, на входах тригера має виконуватись умова $\bar{S} \vee \bar{R} = 1$, яка є характеристичним рівнянням тригера.

Швидкодія асинхронного RS-тригера на елементах І-НЕ становить 2τ (рис. 7.11б).

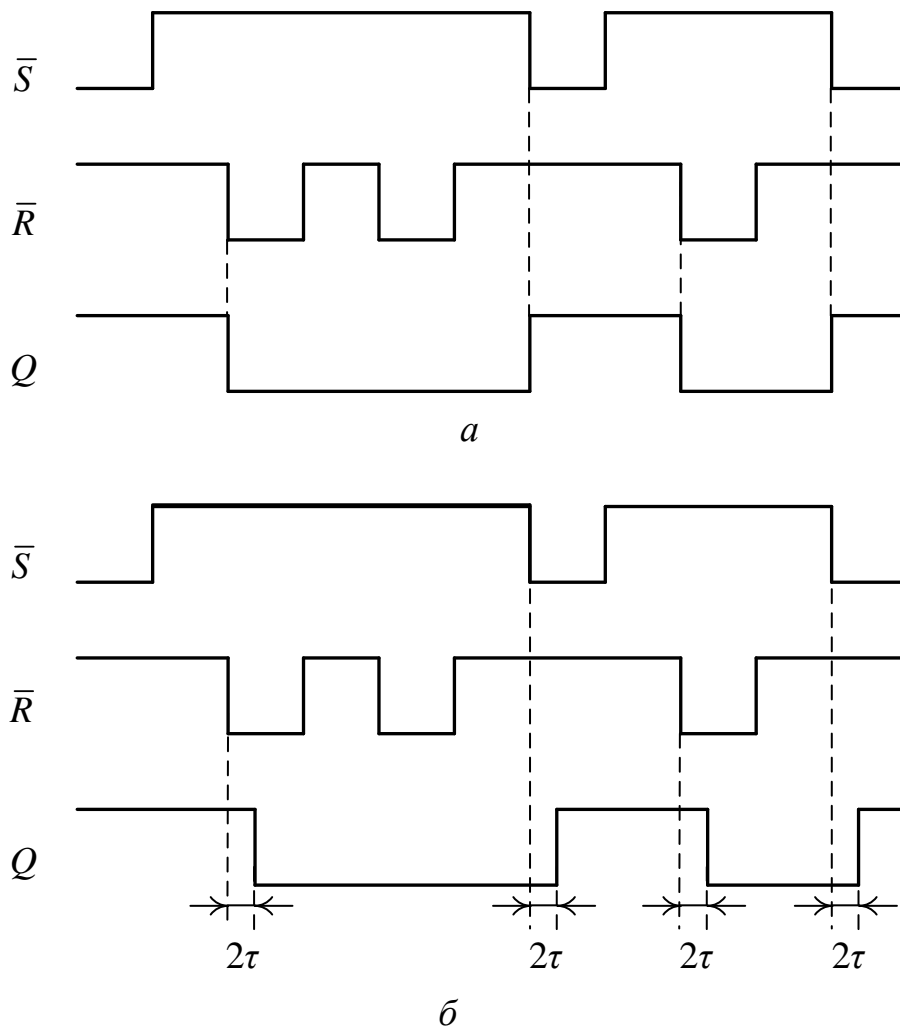
Асинхронний D-тригер на основі лише однієї бістабільної схеми не знайшов застосування як елемент пам'яті через відсутність режиму зберігання ($Q(t_{i+1}) = Q(t_i)$).

D-тригер на основі однієї бістабільної схеми може бути лише синхронним.

Далі розглянемо побудову синхронних тригерів на основі однієї бістабільної схеми.

Запитання та завдання

1. Дайте означення асинхронного тригера. Як здійснюється запис інформації в асинхронний тригер?
2. Наведіть приклади асинхронних тригерів на основі однієї бістабільної схеми.
3. Якою є залежність між входами асинхронного RS-тригера та функціями збудження f_2 , f_1 бістабільної схеми на елементах: а) АБО-НЕ; б) І-НЕ?
4. Наведіть таблицю переходів асинхронного RS-тригера на елементах: а) І-НЕ; б) АБО-НЕ.



τ – затримка сигналу елементом І-НЕ
 2τ – час перемикання тригера

Рис. 7.11. Часова діаграма роботи асинхронного RS-тригера на елементах І-НЕ: *a* – ідеальна; *б* – реальна

5. Якою є швидкодія асинхронного тригера, якщо τ – затримка сигналу в одному логічному елементі?
6. Побудуйте функціональну схему асинхронного RS-тригера на елементах: а) І-НЕ; б) АБО-НЕ.

7.3. Синхронні тригери, керовані рівнем синхросигналу

Запис (занесення) інформації (одного біта) в асинхронний тригер відбувається довільно в часі – в міру надходження сигналів на інформаційні входи тригера. Для того, щоб упорядкувати процес запису інформації в тригер в строго визначені моменти (точки відліку) автоматного часу необхідно, щоб тригер, поряд з інформаційними входами, мав також

спеціальний вхід синхронізації (синхровхід), на який подають керуючий сигнал – синхросигнал. Синхровхід зазвичай позначають літерою C (clock).

Тригери, що мають вхід синхронізації, називають *синхронними тригерами*.

Розглянемо будову синхронних тригерів, у яких керування процесом запису здійснюється рівнем синхросигналу. До складу таких тригерів входить лише одна бістабільна схема. Синхронні тригери, керовані рівнем синхросигналу, перемикаються (переходять з одного стану в інший), якщо на синхровході має місце активний рівень сигналу синхронізації.

Є два різновиди тригерів, керованих рівнем синхросигналу, – на елементах І-НЕ (рис. 7.12) та на елементах АБО-НЕ (рис. 7.13). Структурно тригер, керований рівнем синхросигналу, складається зі схеми керування та однієї бістабільної схеми.

У тригерах на основі елементів І-НЕ перемикання тригера з одного стану в інший відбувається поки на вході C діє високий рівень сигналу, тобто при $C = 1$ (рис. 7.14а), де τ_p – тривалість активного рівня синхросигналу. При $C = 0$ стан тригера не змінюється. Для таких тригерів активним рівнем синхросигналу є рівень логічної одиниці.

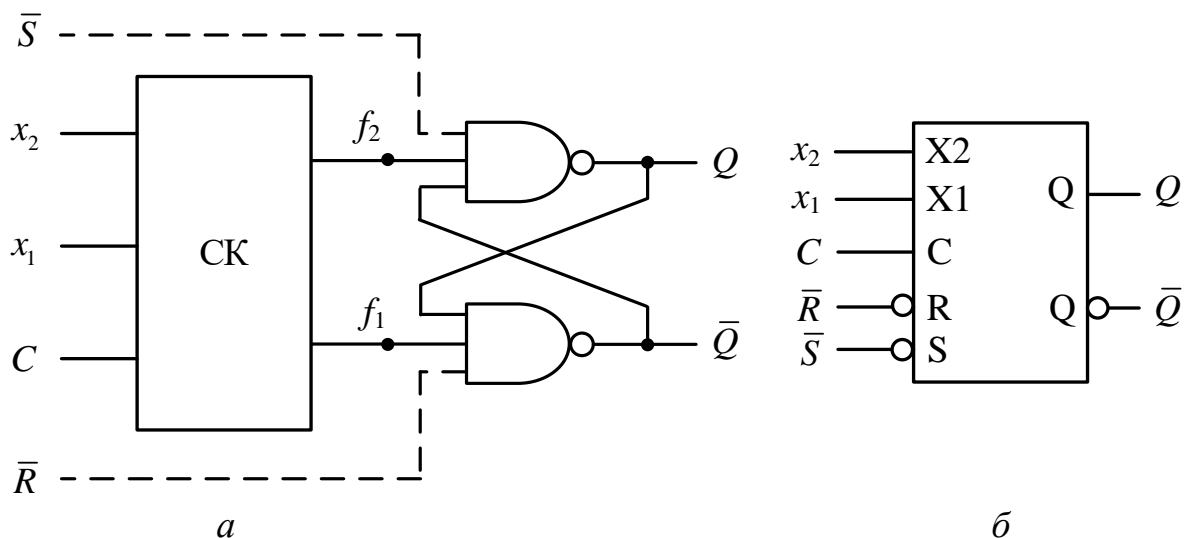


Рис. 7.12. Тригер, керований рівнем синхросигналу, на елементах І-НЕ:

a – узагальнена функціональна схема тригера;

б – умовне графічне позначення тригера

У тригерах на основі елементів АБО-НЕ перехід тригера з одного стану в інший можливий коли $C = 0$ (рис. 7.14б), при $C = 1$ стан тригера не змінюється. Для таких тригерів активним рівнем синхросигналу є рівень логічного нуля. Щоб це підкреслити на схемах таких тригерів сигнал, що подається на синхровхід, позначають \bar{C} та додатково на умовному графічному позначенні тригера вхід C зображають як інверсний.

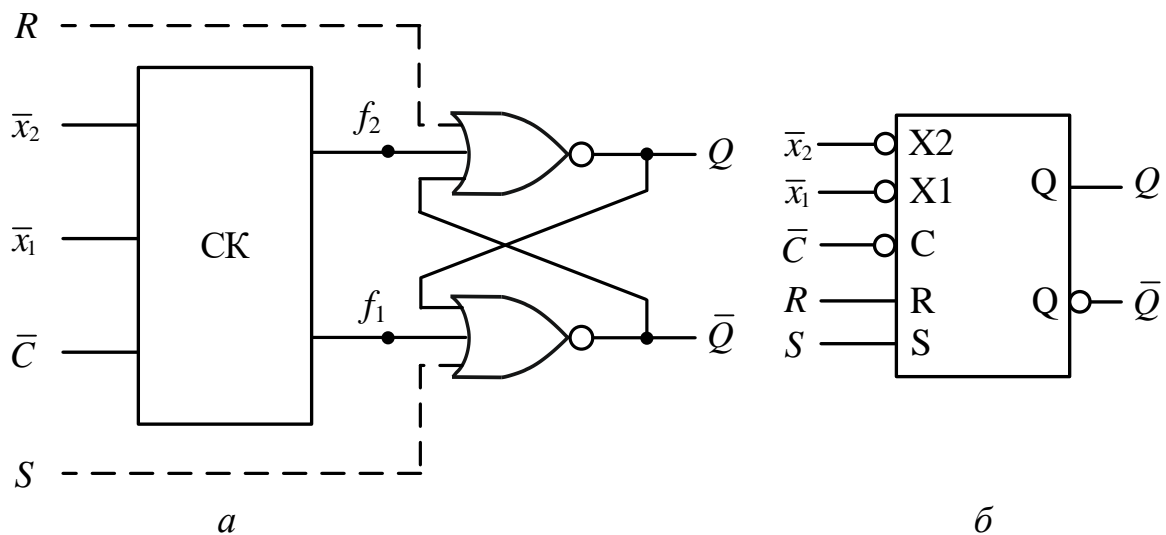


Рис. 7.13. Тригер, керований рівнем синхросигналу, на елементах АБО-НЕ:
a – узагальнена функціональна схема тригера;
б – умовне графічне позначення тригера

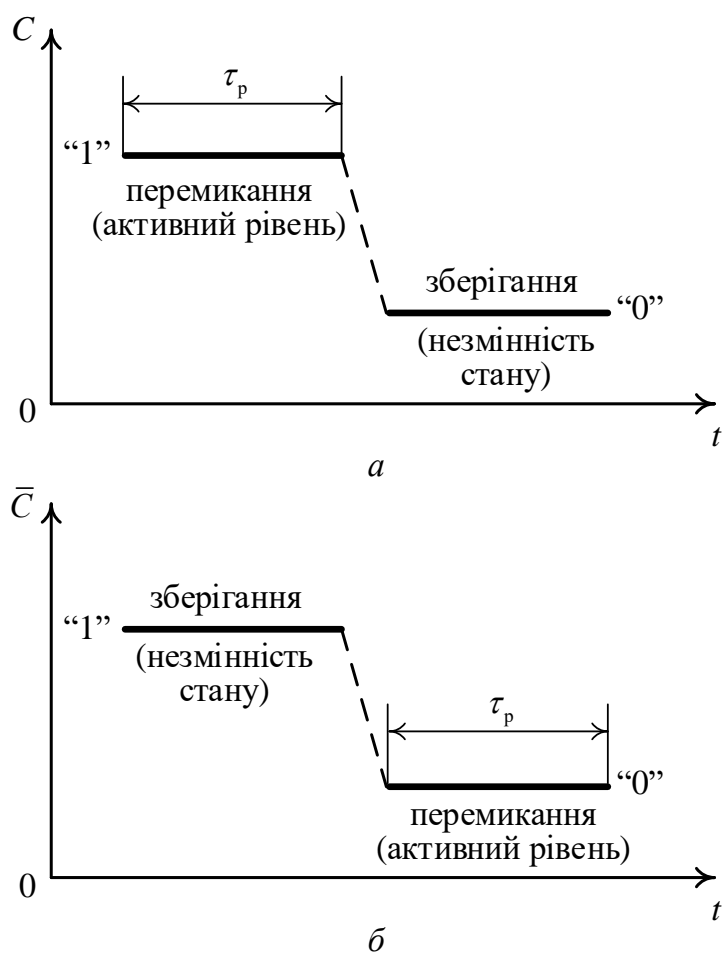


Рис. 7.14. Синхронізація процесу запису інформації в тригери, керовані рівнем синхросигналу, в часі: *a* – на основі елементів І-НЕ;
б – на основі елементів АБО-НЕ

Синхронні тригери можуть мати входи асинхронного (примусового) встановлення тригера в початковий стан “0” або в початковий стан “1”. Вони використовуються для встановлення тригера в потрібний стан (початковий стан) *перед початком роботи* для того, щоб саме з цього стану розпочати функціонування тригера за його таблицею переходів (табл. 7.10, 7.11).

У тригерах на елементах І-НЕ для асинхронного (примусового) встановлення тригера в початковий стан “0” або “1” використовують сигнали \bar{S} , \bar{R} (табл. 7.10). Перед початком роботи за допомогою комбінації $\bar{S} = 0$, $\bar{R} = 1$ тригер можна встановити в “1”, а за допомогою комбінації сигналів $\bar{S} = 1$, $\bar{R} = 0$ – в “0”. Зрозуміло, що при цьому активним рівнем сигналів \bar{S} , \bar{R} є рівень логічного нуля. Для забезпечення функціонування тригера на елементах І-НЕ за таблицею його переходів (режим “Робота”) необхідно встановити $\bar{S} = 1$, $\bar{R} = 1$.

Таблиця 7.10

Режими функціонування синхронного тригера на елементах І-НЕ

Входи асинхронного встановлення тригера в початковий стан “0”, “1”		Режим
\bar{S}	\bar{R}	
0	1	“Примусове встановлення в “1”” “Примусове встановлення в “0”” } Перед початком роботи
1	0	
1	1	“Робота” (функціонування за таблицею переходів тригера)
0	0	Заборонена комбінація

Примітка. Активним рівнем сигналів на входах \bar{S} , \bar{R} є рівень логічного нуля. Під час функціонування тригера (режим “Робота”) на входи \bar{S} , \bar{R} необхідно подавати комбінацію сигналів: $\bar{S} = 1$, $\bar{R} = 1$.

Для асинхронного (примусового) встановлення перед початком роботи в “0” або в “1” тригерів на елементах АБО-НЕ використовують сигнали S , R (рис. 7.13, табл. 7.11). Перед початком роботи тригер встановлюють в початковий стан “1” за допомогою комбінації сигналів $S = 1$, $R = 0$, а в “0” – за допомогою $S = 0$, $R = 1$. Тригер на елементах АБО-НЕ функціонує за його таблицею переходів (режим “Робота”), якщо $S = 0$, $R = 0$.

Синхронні тригери, керовані рівнем синхросигналу, мають недолік – протягом дії активного рівня синхросигналу тригер може перемикатися

стільки разів скільки разів при цьому змінюються сигнали на інформаційних входах тригера.

Для того, щоб усунути зазначений недолік, проєктують тригери, у яких керування процесом запису інформації в тригер здійснюється не рівнем синхросигналу, а перепадом (фронтом) синхросигналу (див. рис. 7.4 б, в).

Таблиця 7.11

Режими функціонування синхронного тригера на елементах АБО-НЕ

Входи асинхронного встановлення в "0", "1"		Режим
<i>S</i>	<i>R</i>	
1	0	"Примусове встановлення в "1" } Перед початком "Примусове встановлення в "0" } роботи
0	1	
0	0	"Робота" (функціонування за таблицею переходів тригера)
1	1	Заборонена комбінація

Примітка. Активним рівнем сигналів на входах *S*, *R* є рівень логічної одиниці. Під час функціонування тригера (режим "Робота") на входах *S*, *R* необхідно подавати комбінацію сигналів: $S = 0$, $R = 0$.

Синхронними тригерами на основі однієї бістабільної схеми, можуть бути лише ті тригери, які у стовпці $Q(t_{i+1})$ таблиці переходів тригера містять значення 0, 1, $Q(t_i)$. Якщо ж у цьому стовпці міститься значення $\bar{Q}(t_i)$, то такий синхронний тригер не може бути побудований на основі лише однієї бістабільної схеми.

Запитання та завдання

1. Які тригери називаються синхронними?
2. Як здійснюється запис інформації в синхронний тригер, керований рівнем синхросигналу, якщо він побудований на основі елементів: а) І-НЕ; б) АБО-НЕ?
3. Якою є структура тригера, керованого рівнем синхросигналу?
4. Яким є активний рівень синхросигналу для синхронних тригерів, побудованих на елементах: а) АБО-НЕ; б) І-НЕ?
5. Наведіть узагальнену функціональну схему тригера, керованого рівнем синхросигналу, якщо він побудований на елементах: а) АБО-НЕ; б) І-НЕ.

6. Якими є режими функціонування синхронного тригера, що має входи асинхронного встановлення тригера в початковий стан “0” та “1”?
7. Які значення сигналів слід утримувати на входах асинхронного встановлення тригера в початковий стан “0”, “1”, якщо тригер перебуває в режимі “Робота” і побудований на елементах: а) АБО-НЕ; б) І-НЕ?
8. Який недолік мають тригери, керовані рівнем синхросигналу?

7.4. Тригери з внутрішньою затримкою

Особливістю тригерів, керованих фронтом (перепадом) синхросигналу (тригерів з внутрішньою затримкою) є те, що вони перемикаються (переходять з одного стану в інший) протягом короткого проміжку часу – часу переходу (перепаду) τ_n синхросигналу з одного рівня до іншого: з нульового до одиничного (за переднім фронтом синхросигналу) або з одиничного до нульового (за заднім фронтом синхросигналу) (рис. 7.4).

Оскільки $\tau_n \ll \tau_p$, то це істотно підвищує надійність перемикання тригера з одного стану в інший. Однак, досягти цього можливо лише ускладненням структури тригера – в тригерах, які спрацьовують за фронтом сигналу, використовують дві або три бістабільні схеми.

Потреба в тригерах з внутрішньою затримкою викликана тим, що під час проектування тригерних схем таблиці переходів проєктованих тригерів можуть у стовпці $Q(t_{i+1})$ містити значення $\bar{Q}(t_i)$ (див., наприклад, табл. 7.7, 7.8). Це означає, що сигнали на виходах Q , \bar{Q} тригера є аргументами функцій збудження f_2, f_1 бістабільної схеми, і для забезпечення правильного перемикання тригера в точках A (рис. 7.1) необхідно розмістити елементи затримки (наприклад, додаткову бістабільну схему).

Потреба в тригерах з внутрішньою затримкою викликана також тим, що під час проектування, наприклад, лічильників або регістрів (вузлів, що складаються з тригерів), аргументами функцій f_2, f_1 у даному розряді лічильника або регістра є сигнали на виходах Q та \bar{Q} тригерів в інших розрядах, що мають перемикатися в процесі роботи одночасно з даним тригером.

У схемах тригерів, керованих перепадом (фронтом) синхросигналу, зміна вхідних (інформаційних) сигналів при встановленому (незмінному) рівні синхросигналу (стала одиниця або сталий нуль синхросигналу) не може спричинити перехід тригера у новий стан. При цьому, звичайно, можуть мати місце зміни рівнів сигналів на виходах деяких логічних елементів схеми, однак на стан тригера це не впливає.

Розрізняють два основні способи побудови тригерів, керованих фронтом синхросигналу: за MS-схемою (на основі двох бістабільних схем) (рис. 7.15, 7.16); на основі трьох бістабільних схем (рис. 7.17).

Можливі два варіанти побудови тригерів за MS-схемою: з інвертором у колі синхросигналу (рис. 7.15) та із заборонними зв'язками (рис. 7.16).

До складу тригера входять дві бістабільні схеми: основна – М-БС (М – master), та допоміжна – S-БС (S – slave). Виходами тригера в цілому є виходи S-БС.

В обох випадках запис інформації в М-БС тактується сигналом C , а передача інформації (біта) з М-БС у S-БС здійснюється через вентиля B .

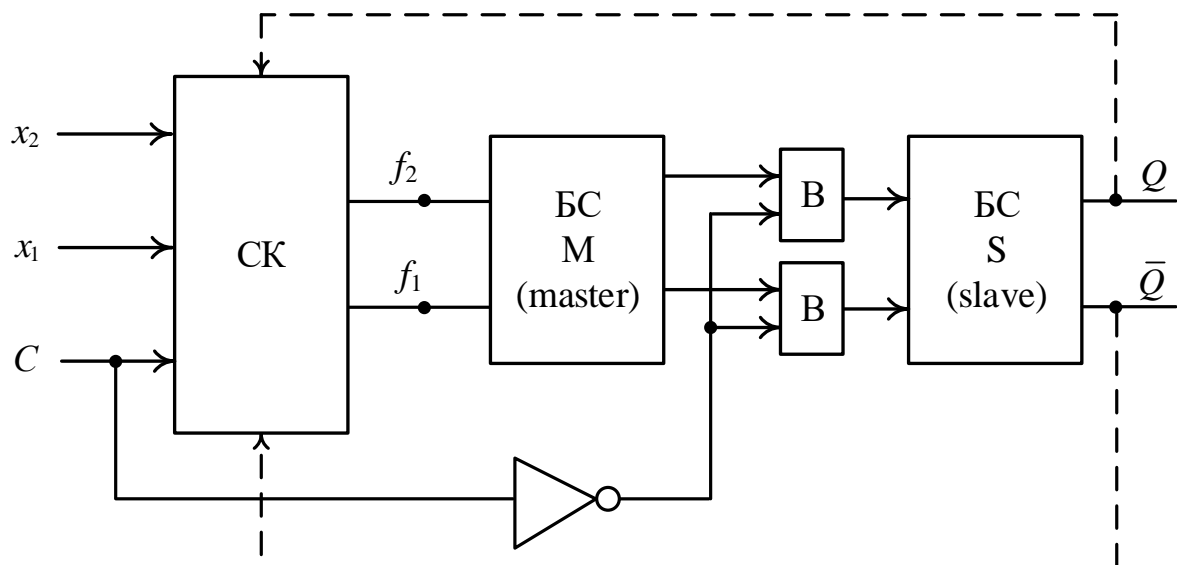


Рис. 7.15. Структура одноктапного синхронного тригера з внутрішньою затримкою за MS-схемою з інвертором у колі синхросигналу

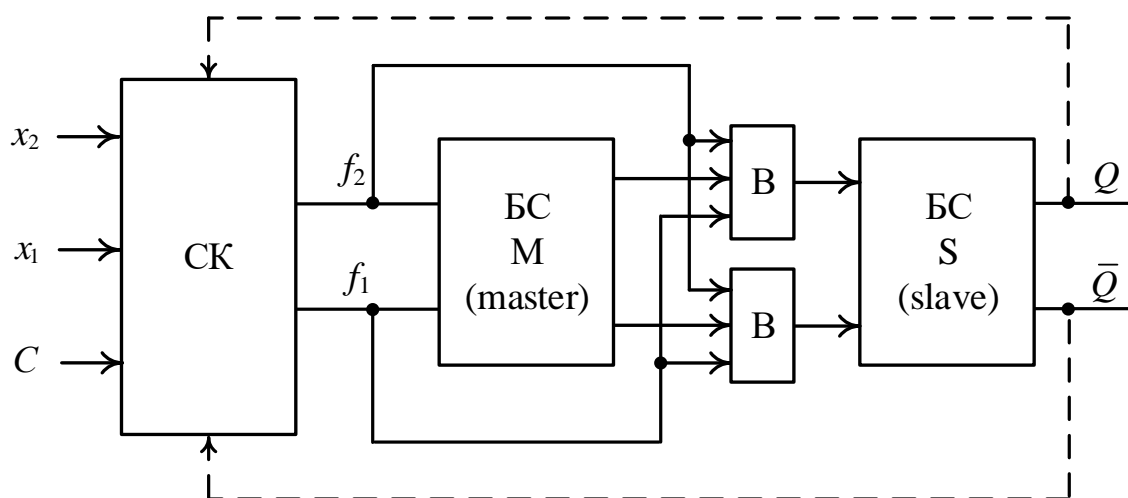


Рис. 7.16. Структура одноктапного синхронного тригера з внутрішньою затримкою за MS-схемою із заборонними зв'язками

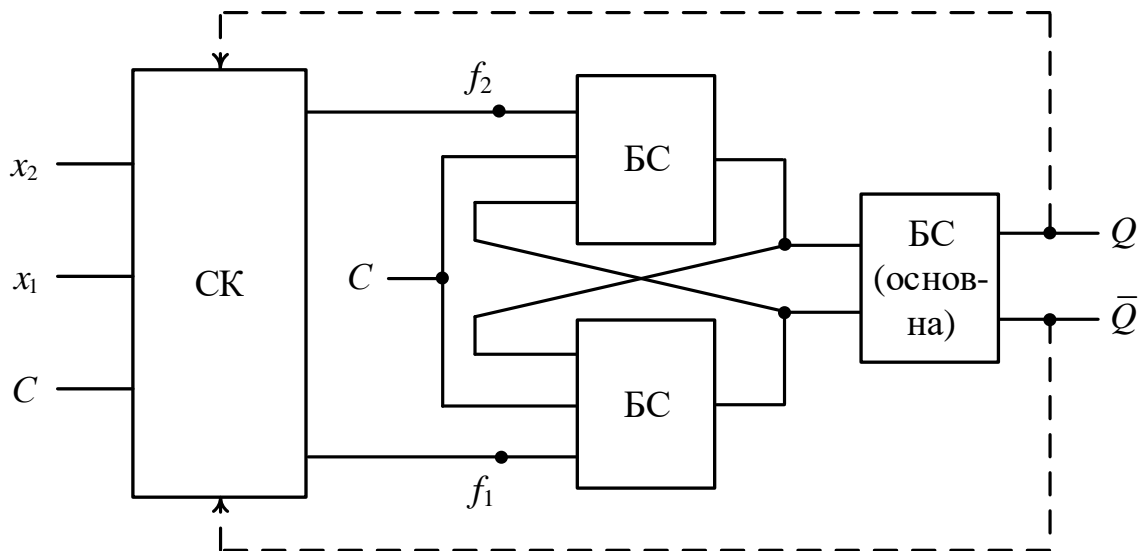


Рис. 7.17. Структура синхронного тригера на основі трьох бістабільних схем

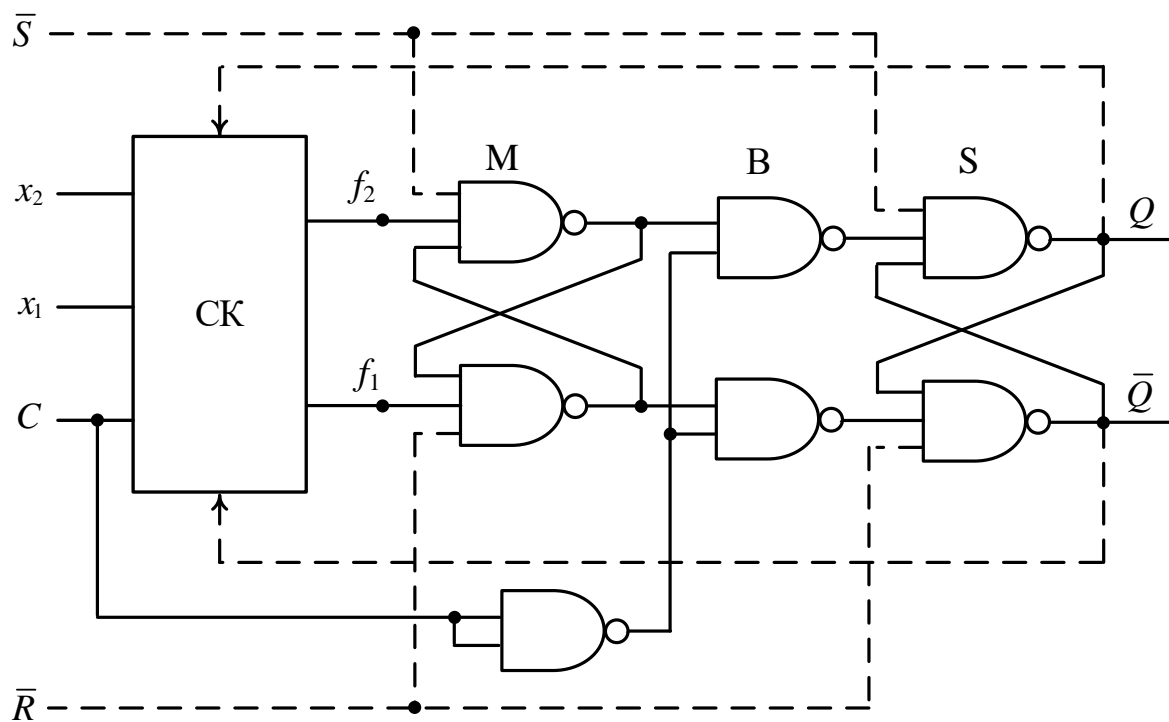


Рис. 7.18. Будова синхронного тригера за MS-схемою з інвертором у колі синхросигналу на елементах І-НЕ

Якщо тригер будують на елементах І-НЕ (рис. 7.18), то вентилям є елемент І-НЕ; якщо на елементах АБО-НЕ, то вентилям є елемент АБО-НЕ (рис. 7.19). У схемі тригера на елементах І-НЕ з інвертором у колі синхросигналу передача біта з М-БС у S-БС відбувається в момент переходу синхросигналу C з 1 в 0 (за заднім фронтом сигналу C) (табл. 7.12). Вентилі В при цьому відкриваються, і стан М-БС переписується в S-БС. При $C = 0$ (сталий нуль) зміна інформаційних сигналів на входах x_2 , x_1 не

може вплинути на стан S-БС. При $C = 1$ (стала одиниця) стан М-БС під впливом інформаційних сигналів може змінитися, але це не позначиться на S-БС, оскільки вентиля В будуть закриті. І лише при наступному переході C з 1 в 0 новий стан М-БС буде переписано в S-БС, і на виходах Q , \bar{Q} встановлюються нові значення.

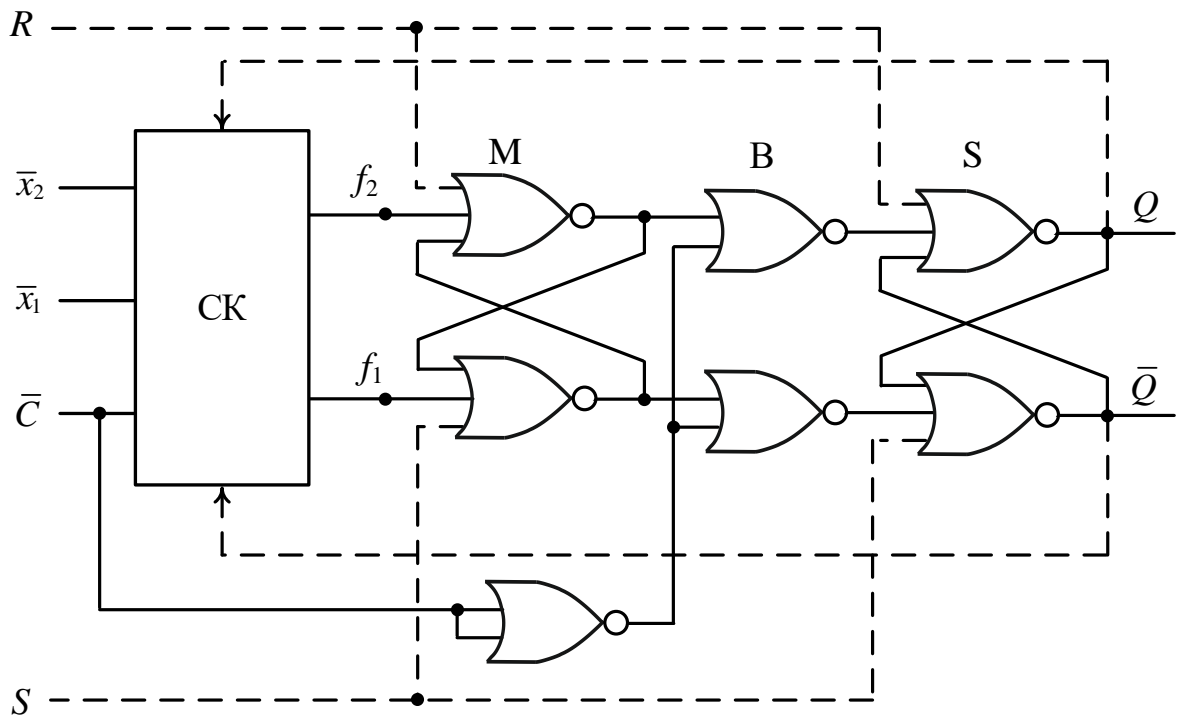


Рис. 7.19. Будова синхронного тригера за MS-схемою з інвертором у колі синхросигналу на елементах АБО-НЕ

У схемі із заборонними зв'язками (рис. 7.16) вентиля В відкриваються лише тоді, коли $f_2 = f_1$. Якщо $f_2 \neq f_1$, то обидва вентиля будуть закриті.

Рівність $f_2 = f_1$ виконується лише в момент переходу синхросигналу C з 0 в 1, або з 1 в 0. У тригерах на елементах АБО-НЕ (рис. 7.21) передача біта з М-БС у S-БС має місце, коли $f_2 = f_1 = 0$ (це досягається за переднім фронтом сигналу C) (табл. 7.12). У тригерах на елементах І-НЕ (рис. 7.20) передача біта з М-БС у S-БС має місце, коли $f_2 = f_1 = 1$ (це досягається за заднім фронтом сигналу C) (табл. 7.12).

При сталому значенні сигналу на вході C (сталий нуль або стала одиниця) S-БС ізольована від М-БС, що забезпечує стабільність роботи тригера.

За MS-схемою із заборонними зв'язками можуть проектуватися також асинхронні тригери, при цьому сигнал C відсутній (рис. 7.22). Такі тригери працюють в режимі очікування надходження вхідних інформаційних сигналів x_2 , x_1 .

Таблиця 7.12

Синхронізація процесу перемикання тригерів з внутрішньою затримкою з одного стану в інший

Структура тригера	Елементний базис	
	I-НЕ	АБО-НЕ
MS- з інвертором у колі синхросигналу		
MS- із заборонними зв'язками		
на основі трьох бістабільних схем		

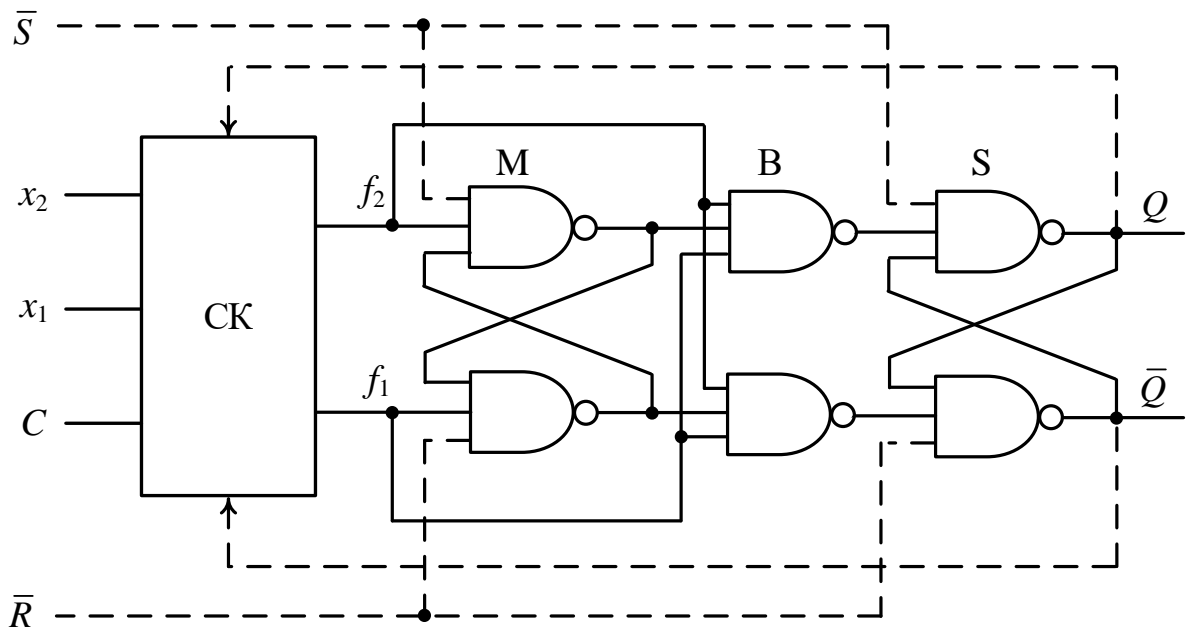


Рис. 7.20. Будова синхронного тригера за MS-схемою із заборонними зв'язками на елементах I-НЕ

У структурі тригера на рис. 7.17 присутні 3 бістабільні схеми – одна основна, виходи якої є виходами тригера, та дві комутувальні бістабільні схеми – верхня та нижня.

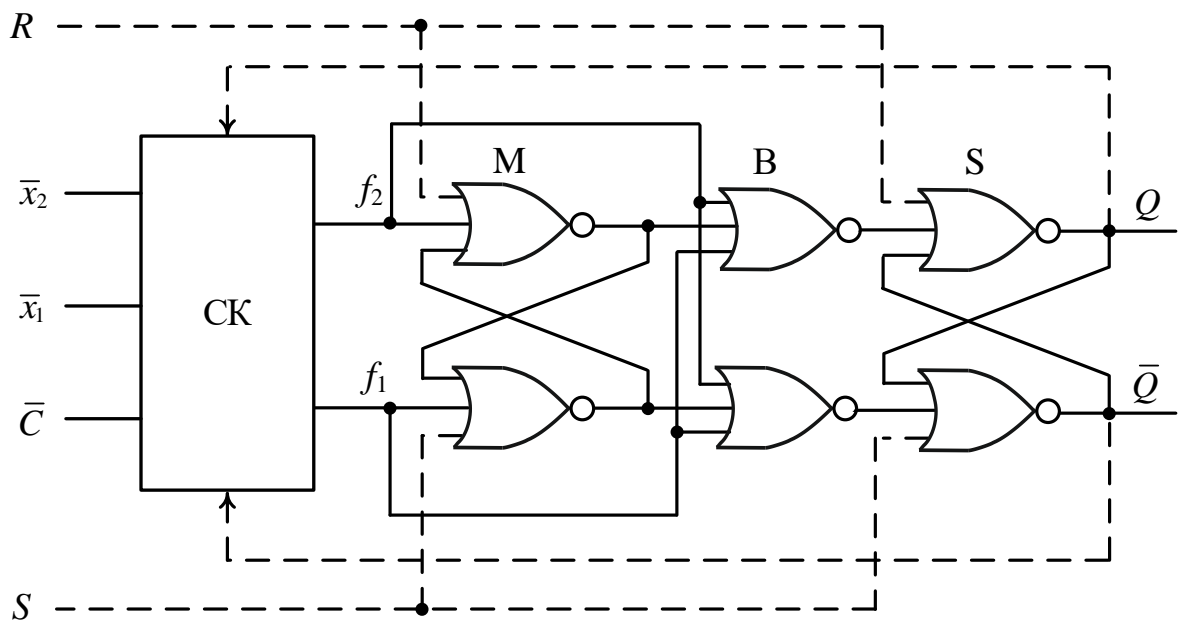


Рис. 7.21. Будова синхронного тригера за MS-схемою із заборонними зв'язками на елементах АБО-НЕ

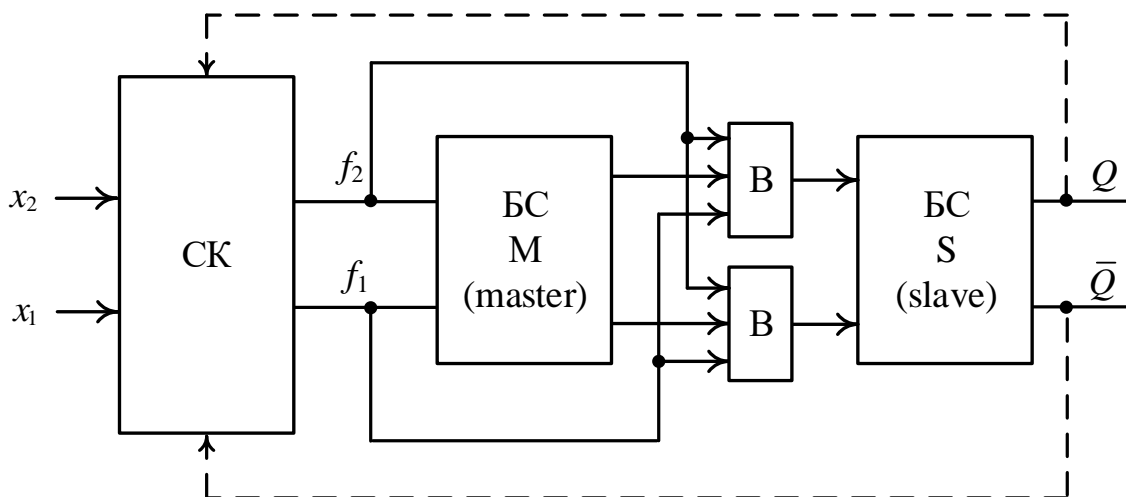


Рис. 7.22. Узагальнена структура асинхронного тригера з внутрішньою затримкою за MS-схемою із заборонними зв'язками

Спосіб запису інформації в такий тригер називають динамічним (тригер з динамічним записом).

У тригері на основі трьох бістабільних схем на елементах І-НЕ (рис. 7.23) новий стан тригера встановлюється при переході синхросигналу C з 0 в 1 (табл. 7.12).

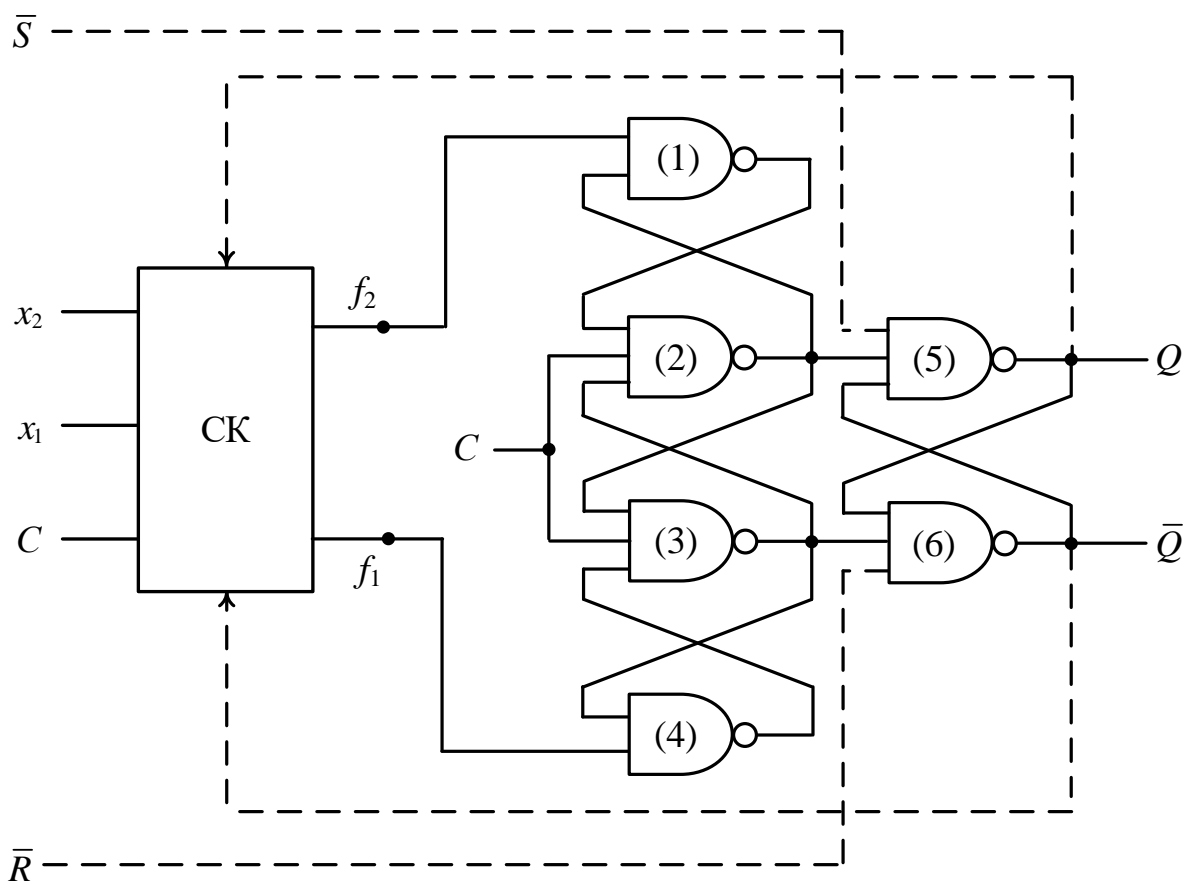


Рис. 7.23. Будова синхронного тригера на основі трьох бістабільних схем на елементах І-НЕ

При $C = 0$ на виходах елементів І-НЕ (2) та (3) присутній сигнал одиниця, тобто основна БС (на елементах (5), (6)) не змінює свого стану, а елементи (1), (4) виконують роль інверторів.

Перемикання тригера (основної БС) здійснюється відповідно до таблиці функцій збудження бістабільної схеми на елементах І-НЕ (табл. 6.5).

Наприклад, якщо $f_2 = 0$ і $f_1 = 1$, то при переході сигналу C з 0 в 1 на виході елемента (2) встановлюється нульовий сигнал, а на виході елемента (3) – одиничний сигнал. Комбінація сигналів 0, 1 на входах основної БС (0 – на вході елемента (5), 1 – на вході елемента (6)) встановлюють її, а, отже, й тригер у стан “1”. Після цього сигнали f_2, f_1 можуть змінюватися, але це не впливатиме на стан основної БС (елементи (5), (6)) доти, поки не відбудеться черговий перехід сигналу C з 0 в 1.

У тригері на основі трьох бістабільних схем на елементах АБО-НЕ (рис. 7.24) новий стан тригера встановлюється при переході синхросигналу C з 1 в 0 (табл. 7.12).

При $C = 1$ на виходах елементів АБО-НЕ (2) та (3) присутній сигнал нуль, тобто основна БС (на елементах (5), (6)) не змінює свого стану.

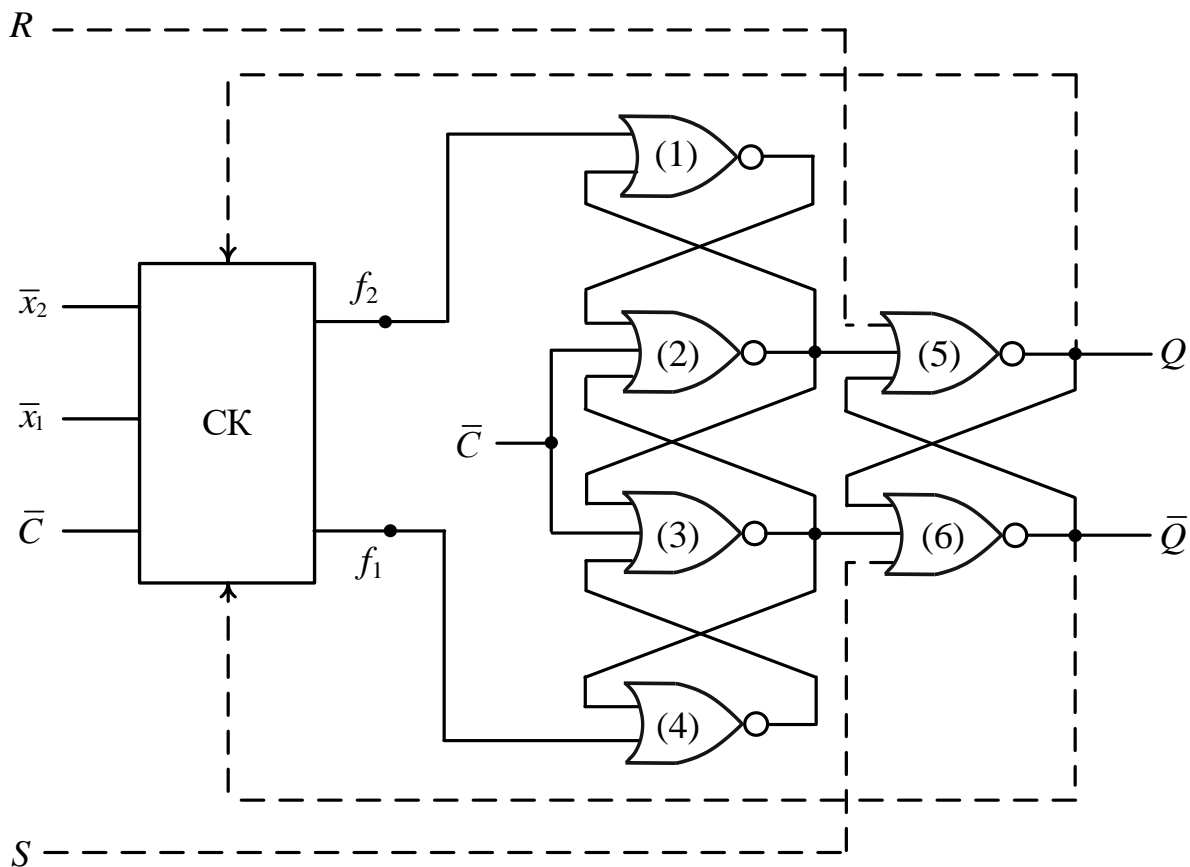


Рис. 7.24. Будова синхронного тригера на основі трьох бістабільних схем на елементах АБО-НЕ

Перемикання тригера (основної БС) здійснюється відповідно до таблиці функцій збудження бістабільної схеми на елементах АБО-НЕ (табл. 6.7).

Наприклад, якщо $f_2 = 1$ і $f_1 = 0$, то при переході сигналу C з 1 в 0 на виході елемента (2) встановлюється одиничний сигнал, а на виході елемента (3) – нульовий. Комбінація сигналів 1, 0 на входах основної БС (1 – на вході елемента (5), 0 – на вході елемента (6)) встановлюють її, а, отже, й тригер, у стан “0”. Після цього сигнали f_2, f_1 можуть змінюватися, але це не впливатиме на стан основної БС (елементи (5), (6)) доти, поки не відбудеться черговий перехід сигналу C з 1 в 0.

Порівнюючи схеми тригерів на елементах І-НЕ зі схемами цих самих тригерів на елементах АБО-НЕ доходимо висновку, що можна сформулювати таке *правило заміщення елементів І-НЕ елементами АБО-НЕ в структурах тригерів*.

У схемі тригера елементи І-НЕ можна замінити на елементи АБО-НЕ, і навпаки. При цьому в схемі тригера на елементах АБО-НЕ, порівняно зі схемою тригера на елементах І-НЕ, слід: сигнал на синхровході позначити \bar{C} ; сигнали \bar{S}, \bar{R} на входах асинхронного встановлення тригера в початковий стан “0”, “1” замінити на R, S відповідно (\bar{S} на R, \bar{R} на S);

замість прямих змінних на інформаційних входах тригера використовувати інверсії змінних (див., наприклад, рис. 7.20 і 7.21, або рис. 7.23 і 7.24).

Запитання та завдання

1. Чим відрізняються тригери з внутрішньою затримкою від тригерів, керованих рівнем синхросигналу?
2. У чому полягає особливість процесу перемикання тригерів з внутрішньою затримкою з одного стану в інший?
3. Чому до складу тригерів з внутрішньою затримкою входять дві або три бістабільні схеми?
4. Чим викликана потреба в тригерах з внутрішньою затримкою?
5. Перелічіть структури тригерів з внутрішньою затримкою.
6. За яким фронтом синхросигналу (переднім чи заднім) відбувається перемикання тригера з одного стану в інший, побудованого за MS-схемою на елементах: а) І-НЕ; б) АБО-НЕ?
7. Як здійснюється запам'ятовування інформації в тригерах, побудованих за MS-схемою із заборонними зв'язками? За яким фронтом синхросигналу це відбувається?
8. Як функціонує тригер, побудований на основі трьох бістабільних схем? За яким фронтом синхросигналу він перемикається з одного стану в інший?
9. Сформулюйте правило заміщення елементів І-НЕ елементами АБО-НЕ в структурах тригерів.

7.5. Методика синтезу синхронних тригерів

При проектуванні синхронних тригерів задають:

- тип тригера, який необхідно синтезувати (RS-, D-, JK-, T- тощо);
- тип логічних елементів, на основі яких має бути побудований тригер (на І-НЕ або на АБО-НЕ);
- спосіб запису інформації в тригер (рівнем чи перепадом синхросигналу).

Проектування зводиться до:

- 1) вибору необхідної структури тригера:
 - тригер, керований рівнем синхросигналу,
 - тригер, на основі MS-схеми (один з двох варіантів – з інвертором у колі синхросигналу або із заборонними зв'язками),
 - тригер на основі трьох бістабільних схем (тригер з динамічним записом інформації);
- 2) синтезу схеми керування тригера – комбінаційної схеми з двома виходами, яка реалізує функції збудження f_2, f_1 тригера.

Методику синтезу синхронних тригерів розглянемо на прикладах розв'язування задач.

Синтез тригерів, керованих рівнем синхросигналу

Задача 7.1. Синтезувати RS-тригер, керований рівнем синхросигналу, на елементах АБО-НЕ. На функціональній схемі тригера показати пунктиром входи асинхронного встановлення тригера в початковий стан “0” та “1”, а також навести умовне графічне позначення синтезованого тригера.

Розв'язування.

1. Вибираємо узагальнену структуру тригера, керованого рівнем синхросигналу, на елементах АБО-НЕ (рис. 7.13).

Далі синтезуємо схему керування тригера (комбінаційну схему з двома виходами).

2. Беремо таблицю переходів RS-тригера (табл. 7.1) та таблицю функцій збудження бістабільної схеми на елементах АБО-НЕ (табл. 6.8):

Таблиця переходів
RS-тригера

$S(t_i)$	$R(t_i)$	$Q(t_{i+1})$
0	0	$Q(t_i)$
0	1	0
1	0	1
1	1	–

Таблиця функцій збудження
бістабільної схеми на
елементах АБО-НЕ

$Q(t_i)$	$Q(t_{i+1})$	f_2	f_1
0	0	×	0
0	1	0	1
1	0	1	0
1	1	0	×

3. Будуємо повну таблицю переходів синхронного RS-тригера (табл. 7.13).

При $C = 0$ тригер не змінює свого стану: $Q(t_{i+1}) = Q(t_i)$. При $C = 1$ тригер має функціонувати за таблицею переходів RS-тригера. Аналізуючи значення $S(t_i)$, $R(t_i)$ у кожному рядку нижньої частини табл. 7.13 заповнюємо стовпець $Q(t_{i+1})$, керуючись таблицею переходів RS-тригера.

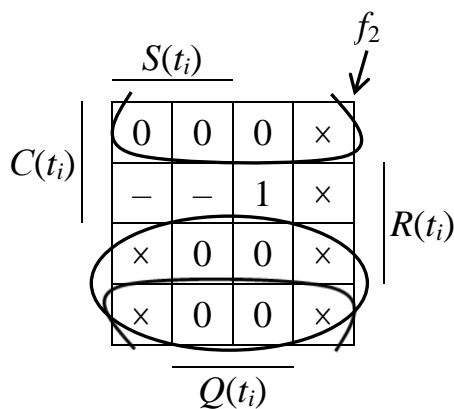
4. У повній таблиці переходів синтезованого тригера, аналізуючи порядково переходи $Q(t_i) \rightarrow Q(t_{i+1})$ та беручи до уваги таблицю функцій збудження бістабільної схеми на елементах АБО-НЕ, заповнюємо стовпці f_2, f_1 .

Таблиця 7.13

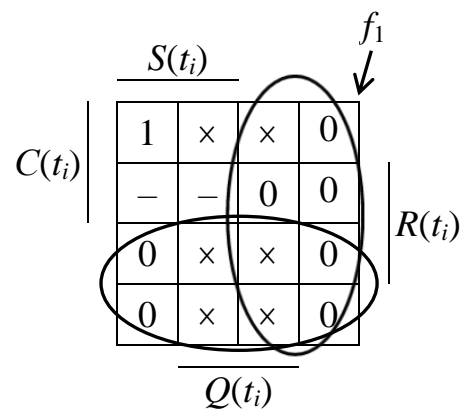
Повна таблиця переходів синхронного RS-тригера на елементах АБО-НЕ

$C(t_i)$	$S(t_i)$	$R(t_i)$	$Q(t_i)$	$Q(t_{i+1})$	f_2	f_1
0	0	0	0	0	×	0
0	0	0	1	1	0	×
0	0	1	0	0	×	0
0	0	1	1	1	0	×
0	1	0	0	0	×	0
0	1	0	1	1	0	×
0	1	1	0	0	×	0
0	1	1	1	1	0	×
1	0	0	0	0	×	0
1	0	0	1	1	0	×
1	0	1	0	0	×	0
1	0	1	1	0	1	0
1	1	0	0	1	0	1
1	1	0	1	1	0	×
1	1	1	0	—	—	—
1	1	1	1	—	—	—

5. За допомогою діаграм Вейча мінімізуємо функції f_2 та f_1 (склеюємо нульові значення):



$$f_2 = C \cdot R = \overline{\overline{C}} \vee \overline{\overline{R}}$$



$$f_1 = C \cdot S = \overline{\overline{C}} \vee \overline{\overline{S}}$$

6. Будуємо схему керування тригера – комбінаційну схему, яка реалізує функції f_2, f_1 (рис. 7.25).

7. Будуємо схему тригера. Для цього в узагальненій структурі тригера на рис. 7.13 замінюємо СК на щойно спроектовану комбінаційну схему з двома виходами. У підсумку отримуємо функціональну схему тригера на рис. 7.26а. ▣

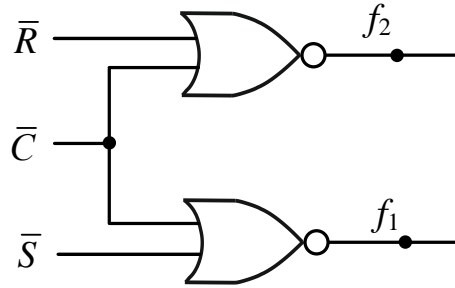


Рис. 7.25. Схема керування синхронного RS-тригера на елементах АБО-НЕ

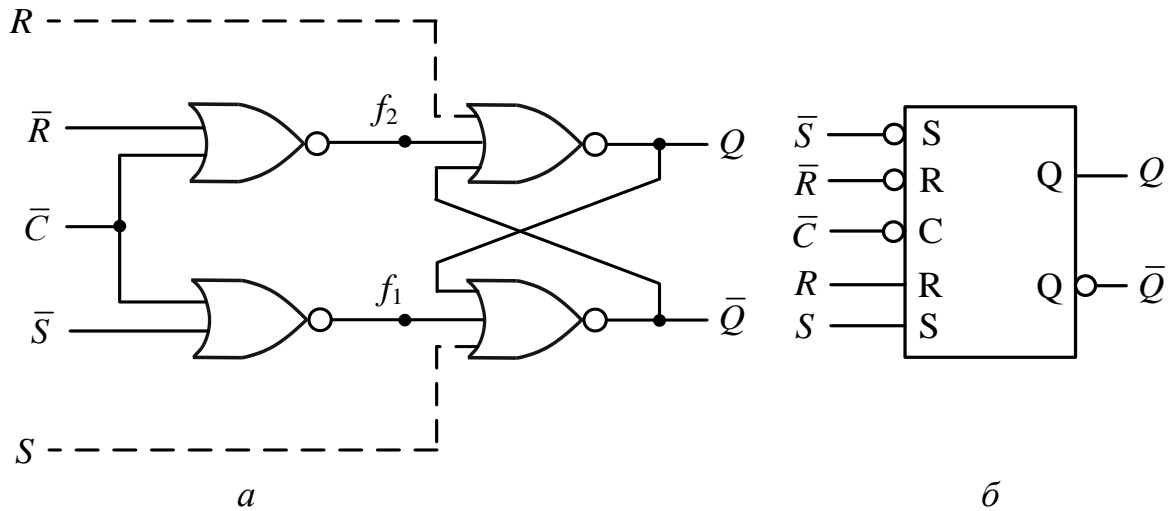


Рис. 7.26. Синхронний RS-тригер на елементах АБО-НЕ, керований рівнем синхросигналу: *a* – функціональна схема тригера; *б* – умовне графічне позначення тригера

Задача 7.2. Синтезувати RS-тригер, керований рівнем синхросигналу, на елементах І-НЕ. На функціональній схемі тригера показати пунктиром входи асинхронного встановлення тригера в початковий стан “0” та “1”, а також навести умовне графічне позначення синтезованого тригера.

Розв’язування.

1. Вибираємо узагальнену структуру тригера, керованого рівнем синхросигналу, на елементах І-НЕ (рис. 7.12).

Далі синтезуємо схему керування тригера – комбінаційну схему з двома виходами f_2, f_1 .

2. Беремо таблицю переходів RS-тригера (табл. 7.1) та таблицю функцій збудження бістабільної схеми на елементах І-НЕ (табл. 6.5):

Таблиця переходів
RS-тригера

$S(t_i)$	$R(t_i)$	$Q(t_{i+1})$
0	0	$Q(t_i)$
0	1	0
1	0	1
1	1	–

Таблиця функцій збудження
бістабільної схеми на
елементах І-НЕ

$Q(t_i)$	$Q(t_{i+1})$	f_2	f_1
0	0	1	×
0	1	0	1
1	0	1	0
1	1	×	1

3. Будуємо повну таблицю переходів синхронного RS-тригера (табл. 7.14).

При $C = 0$ (верхня частина таблиці) тригер не змінює свого стану: $Q(t_{i+1}) = Q(t_i)$. При $C = 1$ (нижня частина таблиці) аналізуючи значення $S(t_i)$, $R(t_i)$ у кожному рядку таблиці заповнюємо стовпець $Q(t_{i+1})$ відповідно до закону функціонування RS-тригера.

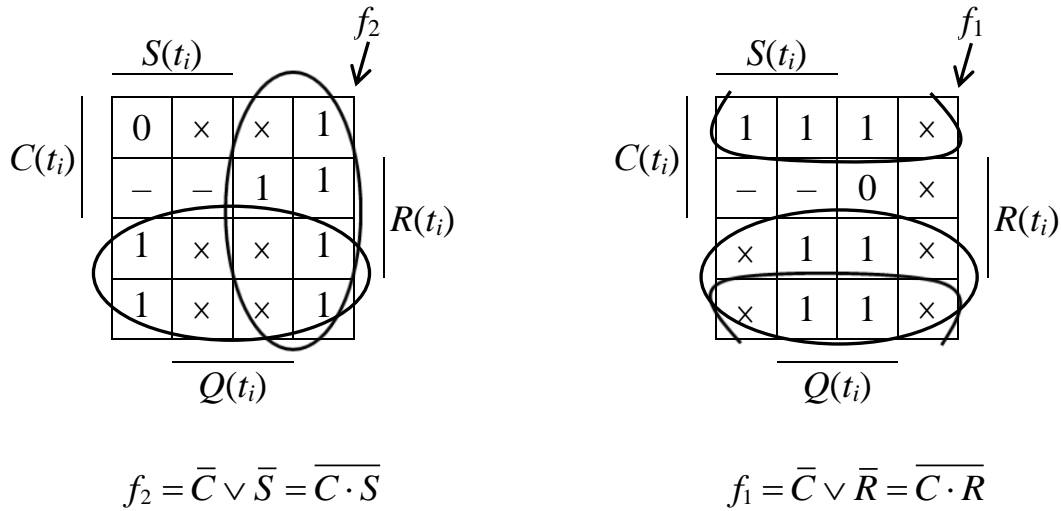
4. У повній таблиці переходів синтезованого тригера, аналізуючи порядково переходи $Q(t_i) \rightarrow Q(t_{i+1})$ та беручи до уваги таблицю функцій збудження бістабільної схеми на елементах І-НЕ, заповнюємо стовпці f_2, f_1 .

Таблиця 7.14

Повна таблиця переходів синхронного RS-тригера на елементах І-НЕ

$C(t_i)$	$S(t_i)$	$R(t_i)$	$Q(t_i)$	$Q(t_{i+1})$	f_2	f_1
0	0	0	0	0	1	×
0	0	0	1	1	×	1
0	0	1	0	0	1	×
0	0	1	1	1	×	1
0	1	0	0	0	1	×
0	1	0	1	1	×	1
0	1	1	0	0	1	×
0	1	1	1	1	×	1
1	0	0	0	0	1	×
1	0	0	1	1	×	1
1	0	1	0	0	1	×
1	0	1	1	0	1	0
1	1	0	0	1	0	1
1	1	0	1	1	×	1
1	1	1	0	–	–	–
1	1	1	1	–	–	–

5. За допомогою діаграм Вейча мінімізуємо функції f_2 та f_1 (склеюємо одиниці):



6. Будуємо схему керування тригера – комбінаційну схему з двома виходами – f_2 та f_1 (рис. 7.27).

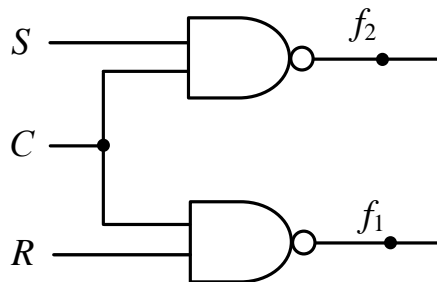


Рис. 7.27. Схема керування синхронного RS-тригера на елементах І-НЕ

7. Будуємо схему тригера. Для цього в узагальненій структурі тригера на рис. 7.12 замінюємо СК на щойно спроектовану комбінаційну схему з двома виходами. Як наслідок отримуємо функціональну схему тригера (рис. 7.28a). ▣

Синтез синхронних тригерів на основі MS-схеми

Задача 7.3. Синтезувати RS-тригер на елементах АБО-НЕ за MS-схемою із заборонними зв'язками. На функціональній схемі тригера показати пунктиром входи асинхронного встановлення тригера в початковий стан “0” та “1”, а також навести умовне графічне позначення синтезованого тригера.

Розв'язування.

1. Вибираємо за прототип узагальнену структуру тригера за MS-схемою із заборонними зв'язками на елементах АБО-НЕ на рис. 7.21.

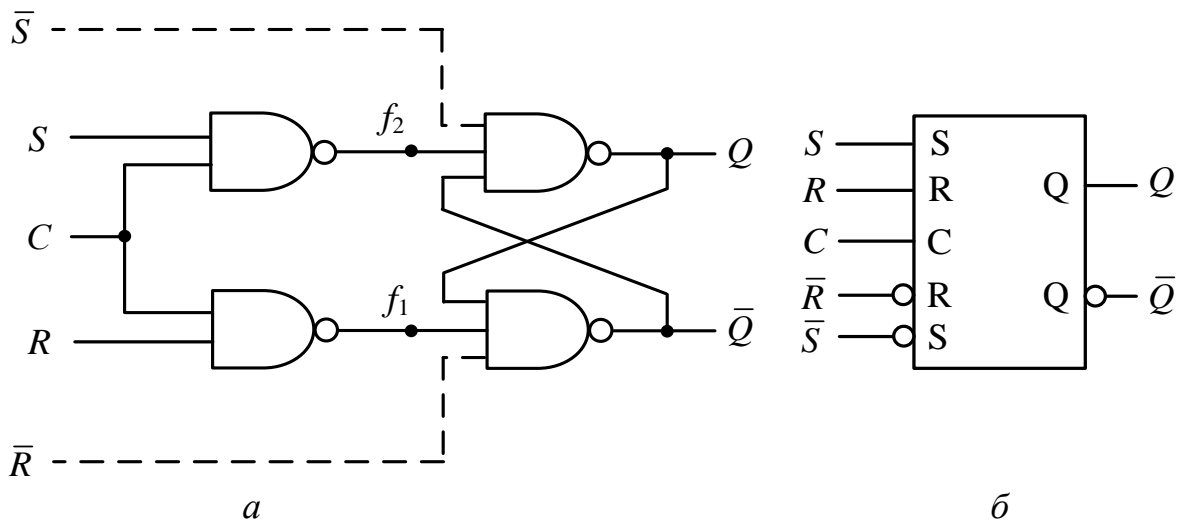


Рис. 7.28. Синхронний RS-тригер на елементах І-НЕ, керований рівнем синхросигналу:
а – функціональна схема тригера;
б – умовне графічне позначення тригера

Далі синтезуємо схему керування тригера, яка реалізує функції збудження f_2, f_1 .

Синтез схеми керування повністю співпадає з пунктами 2 – 6 задачі 7.1.

7. Будуємо схему тригера (рис. 7.29). Для цього в узагальненій структурі тригера на рис. 7.21 замінюємо СК на комбінаційну схему з рис. 7.25.

Перехід тригера з одного стану в інший відбувається за переднім фронтом синхросигналу (рис. 7.30). ■

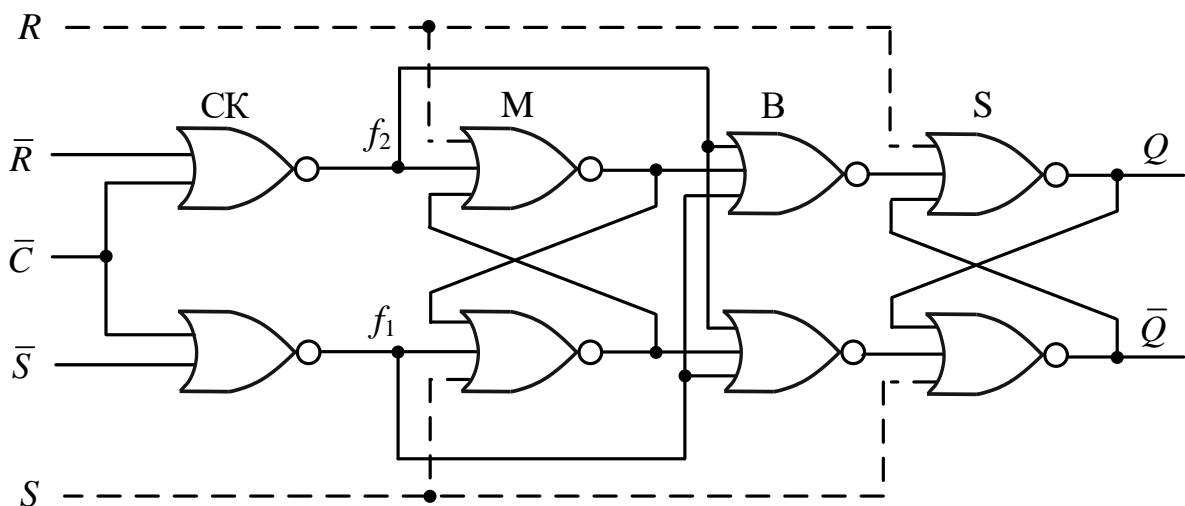


Рис. 7.29. Функціональна схема синхронного RS-тригера на елементах АБО-НЕ на основі MS-схеми із заборонними зв'язками

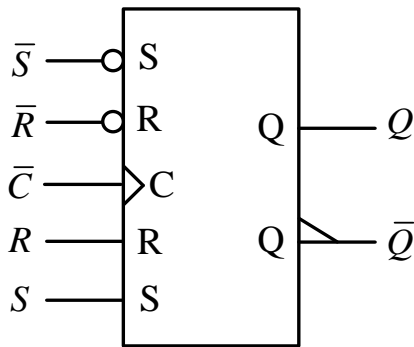


Рис. 7.30. Умовне графічне позначення синхронного RS-тригера на елементах АБО-НЕ на основі MS-схеми

Порівнюючи задачі 7.1 та 7.3 бачимо, що синтез схеми керування тригера не залежить від структурної організації тригера, а повністю визначається лише законом функціонування тригера (таблицею переходів тригера) та елементною базою (І-НЕ чи АБО-НЕ). Особливості структури синхронного тригера жодним чином не впливають на синтез схеми керування тригера. Дійсно, *синтез тригера зводиться до синтезу лише схеми керування тригера.*

Синтез синхронних тригерів на основі трьох бістабільних схем

Задача 7.4. Синтезувати синхронний Т-тригер на елементах І-НЕ на основі трьох бістабільних схем.

Розв'язування.

1. Вибираємо узагальнену структуру тригера на основі трьох бістабільних схем на елементах І-НЕ на рис. 7.23.

Далі синтезуємо схему керування тригера – комбінаційну схему з виходами f_2, f_1 .

2. Беремо таблицю переходів Т-тригера (табл. 7.7) та таблицю функцій збудження бістабільної схеми на елементах І-НЕ (табл. 6.5):

Таблиця переходів
Т-тригера

$T(t_i)$	$Q(t_{i+1})$
0	$Q(t_i)$
1	$\bar{Q}(t_i)$

Таблиця функцій збудження
бістабільної схеми на
елементах І-НЕ

$Q(t_i)$	$Q(t_{i+1})$	f_2	f_1
0	0	1	×
0	1	0	1
1	0	1	0
1	1	×	1

3. Будуємо повну таблицю переходів синхронного Т-тригера (табл. 7.15).

При $C = 0$ тригер не змінює свого стану: $Q(t_{i+1}) = Q(t_i)$. При $C = 1$ тригер має функціонувати за таблицею переходів Т-тригера. Аналізуючи значення $T(t_i)$ у кожному рядку нижньої частини табл. 7.15 заповнюємо стовпець $Q(t_{i+1})$, керуючись таблицею переходів Т-тригера.

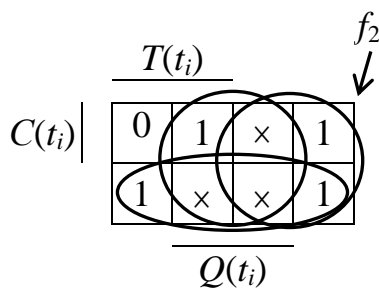
Таблиця 7.15

Повна таблиця переходів синхронного Т-тригера на елементах І-НЕ

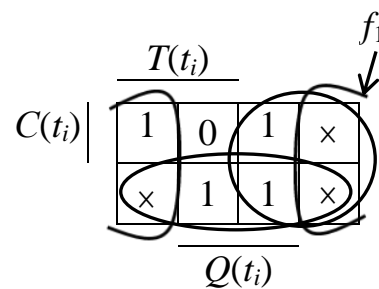
$C(t_i)$	$T(t_i)$	$Q(t_i)$	$Q(t_{i+1})$	f_2	f_1
0	0	0	0	1	×
0	0	1	1	×	1
0	1	0	0	1	×
0	1	1	1	×	1
1	0	0	0	1	×
1	0	1	1	×	1
1	1	0	1	0	1
1	1	1	0	1	0

4. У повній таблиці переходів тригера (табл. 7.15), аналізуючи порядково переходи $Q(t_i) \rightarrow Q(t_{i+1})$ та беручи до уваги таблицю функцій збудження бістабільної схеми на елементах І-НЕ, заповнюємо стовпці f_2 та f_1 .

5. За допомогою діаграм Вейча мінімізуємо функції f_2 та f_1 (склеюємо одиниці):



$$f_2 = \bar{C} \vee \bar{T} \vee Q = \overline{CT\bar{Q}}$$



$$f_1 = \bar{C} \vee \bar{T} \vee \bar{Q} = \overline{CTQ}$$

6. Будуємо схему керування тригера (рис. 7.31).

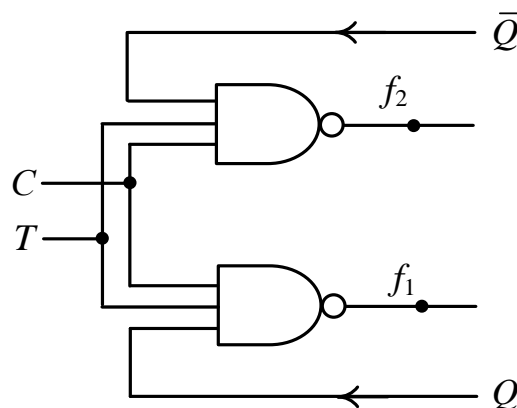


Рис. 7.31. Схема керування синхронного Т-тригера на елементах І-НЕ

7. Будуємо схему тригера. Для цього в узагальненій структурі тригера на рис. 7.23 замінюємо СК на комбінаційну схему з рис. 7.31. У підсумку отримуємо функціональну схему тригера на рис. 7.32.

Перехід тригера з одного стану в інший відбувається за переднім фронтом синхросигналу. ■

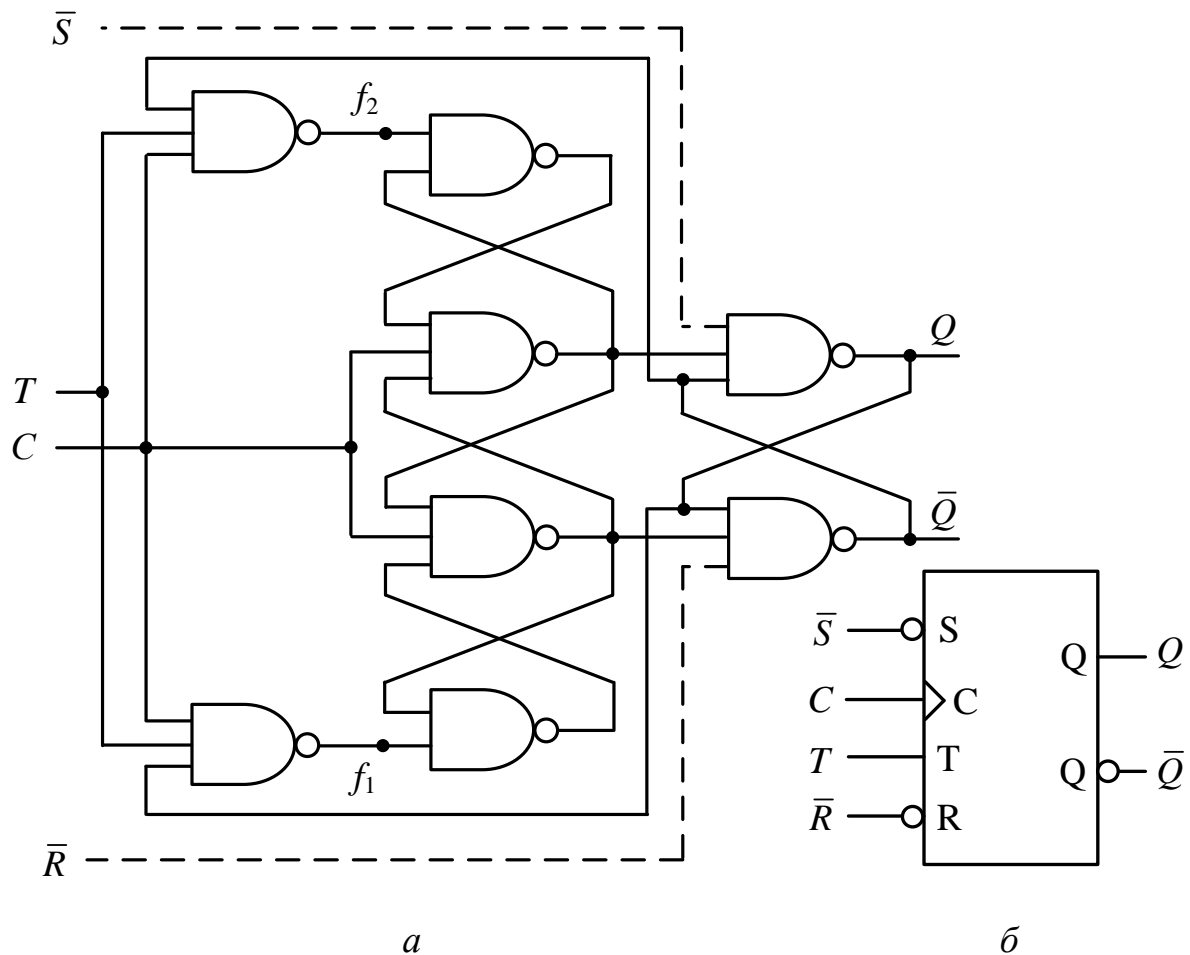


Рис. 7.32. Синхронний Т-тригер на елементах І-НЕ на основі трьох бістабільних схем (входи \bar{S} , \bar{R} асинхронного встановлення тригера в початковий стан “0”, “1” можуть бути відсутні): *a* – функціональна схема тригера; *б* – умовне графічне позначення тригера

За розглянутою методикою можна синтезувати будь-який синхронний тригер, зокрема тригери, закон функціонування яких задають таблиці переходів 7.1 – 7.8.

Якщо тригер синтезують на основі елементів І-НЕ, то під час мінімізації функцій збудження f_2 , f_1 тригера на діаграмі Вейча склеюють одиниці; якщо тригер синтезують на елементах АБО-НЕ, то склеюють нулі.

Якщо у стовпці $Q(t_{i+1})$ таблиці переходів тригера міститься $\bar{Q}(t_i)$ (див. табл. 7.7, 7.8), то такий тригер слід синтезувати лише на основі MS-схеми або на основі трьох бістабільних схем. *Наслідок:* Т- та JK-тригери не

можуть бути спроектовані на основі структури тригера, керованого рівнем синхросигналу, тобто Т- та JK-тригери не можуть у своїй структурі містити лише одну бістабільну схему. Структури Т- та JK-тригерів містять дві або три бістабільні схеми.

Запитання та завдання

1. Які характеристики синхронних тригерів задають під час їх синтезу?
2. До чого зводиться синтез синхронного тригера?
3. Яку послідовність дій виконують під час синтезу синхронного тригера заданого типу?
4. Наведіть таблицю функцій збудження бістабільної схеми на елементах: а) І-НЕ; б) АБО-НЕ.
5. Як на основі таблиці переходів тригера отримати повну таблицю переходів синхронного тригера?
6. Як синтез схеми керування тригера залежить від структурної організації тригера?
7. Чи можна синтезувати Т- та JK-тригери на основі структури тригера, керованого рівнем синхросигналу?

Задачі для самостійного розв'язування

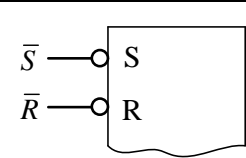
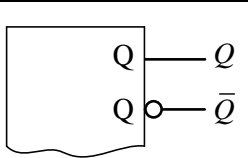
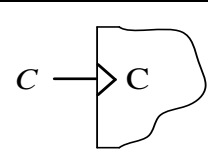
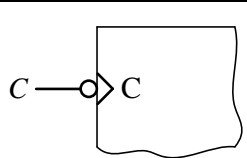
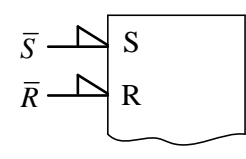
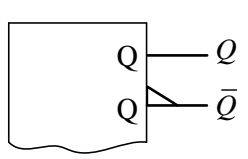

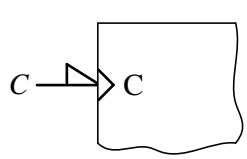
1. На елементах АБО-НЕ синтезувати R-тригер, керований рівнем синхросигналу. На функціональній схемі тригера показати пунктиром входи асинхронного встановлення тригера в початковий стан "0" та "1", а також навести умовне графічне позначення синтезованого тригера.
2. Синтезувати DV-тригер на елементах І-НЕ, керований рівнем синхросигналу.
3. На елементах АБО-НЕ синтезувати схему керування S-тригера, керованого рівнем синхросигналу.
4. На елементах АБО-НЕ синтезувати Т-тригер на основі MS-схеми із заборонними зв'язками. На функціональній схемі тригера показати пунктиром входи асинхронного встановлення тригера в початковий стан "0" та "1", а також навести умовне графічне позначення синтезованого тригера.
5. Синтезувати Е-тригер на елементах І-НЕ на основі MS-схеми з інвертором у колі синхросигналу.
6. Синтезувати схему керування JK-тригера на елементах І-НЕ.

7.6. Умовні графічні позначення тригерів

Під час формування умовних графічних позначень (УГП) тригерів використовують певні графічні елементи, щоб підкреслити особливості входів та виходів тригера (табл. 7.16).

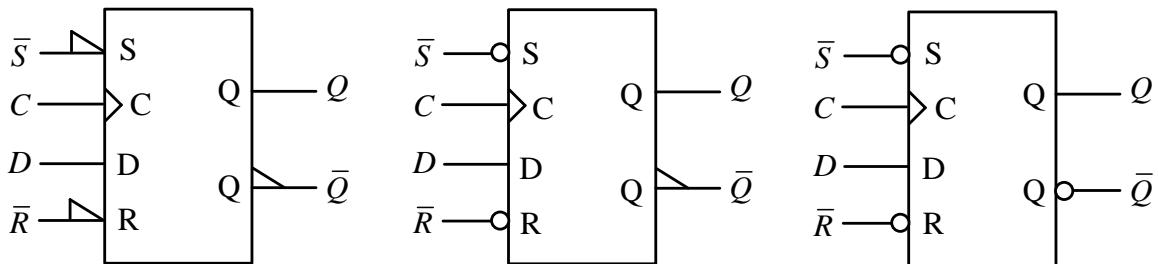
Таблиця 7.16

Графічні елементи, використовувані для позначення особливостей входів та виходів тригерів

Інверсні входи	Інверсний вихід	Передній фронт синхросигналу	Задній фронт синхросигналу
			
			

На рис. 7.33 наведено приклади УГП D- та JK-тригера з внутрішньою затримкою. При позначенні синхровходу C тригерів з внутрішньою затримкою слід керуватись табл. 7.12.

Синхронний D-тригер на основі трьох бістабільних схем на елементах І-НЕ, перемикання тригера відбувається за переднім фронтом синхросигналу



Синхронний JK-тригер на основі MS-схеми на елементах І-НЕ, перемикання тригера відбувається за заднім фронтом синхросигналу

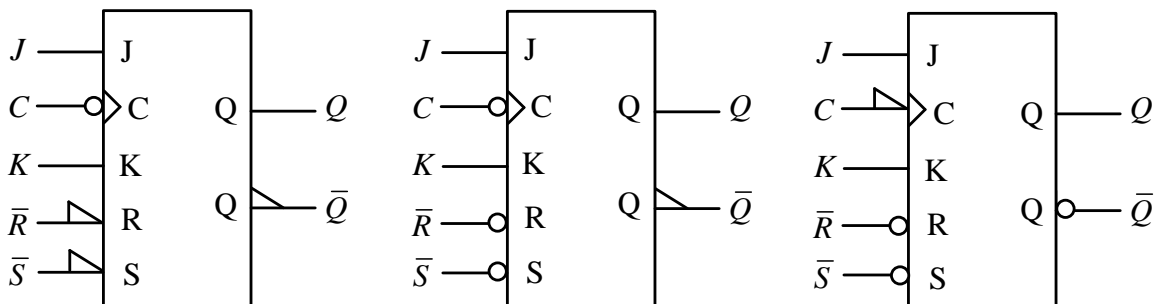


Рис. 7.33. Приклади умовних графічних позначень D- та JK-тригера

Якщо активним рівнем сигналу на деякому вході є рівень логічного нуля, то відповідний вхід позначають як інверсний (рис. 7.34). Якщо входи тригера зображені як прямі (прямий вхід), то активним рівнем сигналу на таких входах є рівень логічної одиниці (рис. 7.33, 7.34).

Інформаційні входи синхронного тригера є прямими, якщо тригер побудовано на елементах І-НЕ; інверсними – якщо тригер побудовано на основі елементів АБО-НЕ.

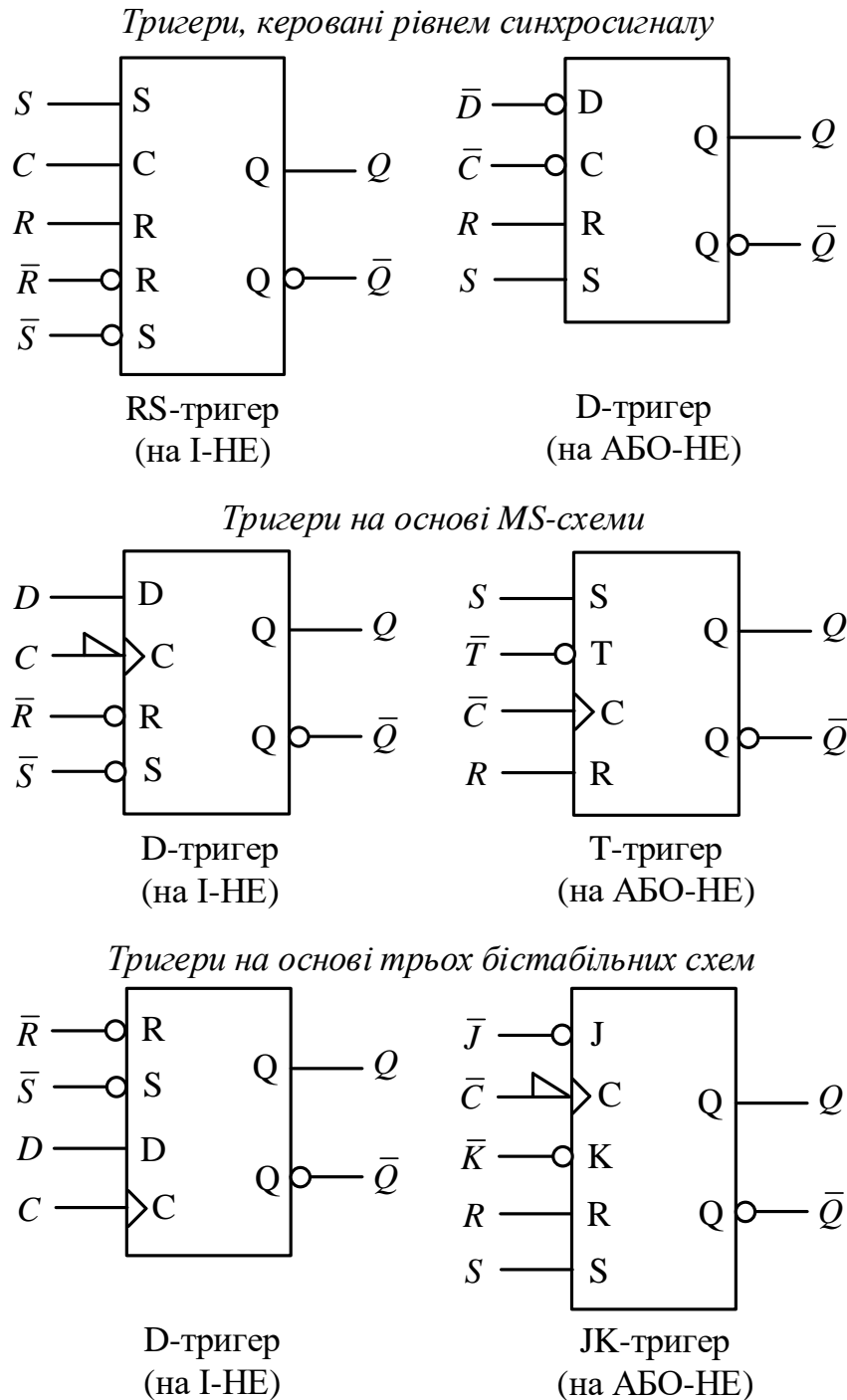


Рис. 7.34. Приклади умовних графічних позначень синхронних тригерів

У всіх структурах тригерів входи асинхронного встановлення в початковий стан ("0" або "1") позначають: S , R – в тригерах на основі елементів АБО-НЕ; \bar{S} , \bar{R} – в тригерах на основі елементів І-НЕ.

У вітчизняній літературі використовують також альтернативні умовні графічні позначення тригерів (рис. 7.35, 7.36).

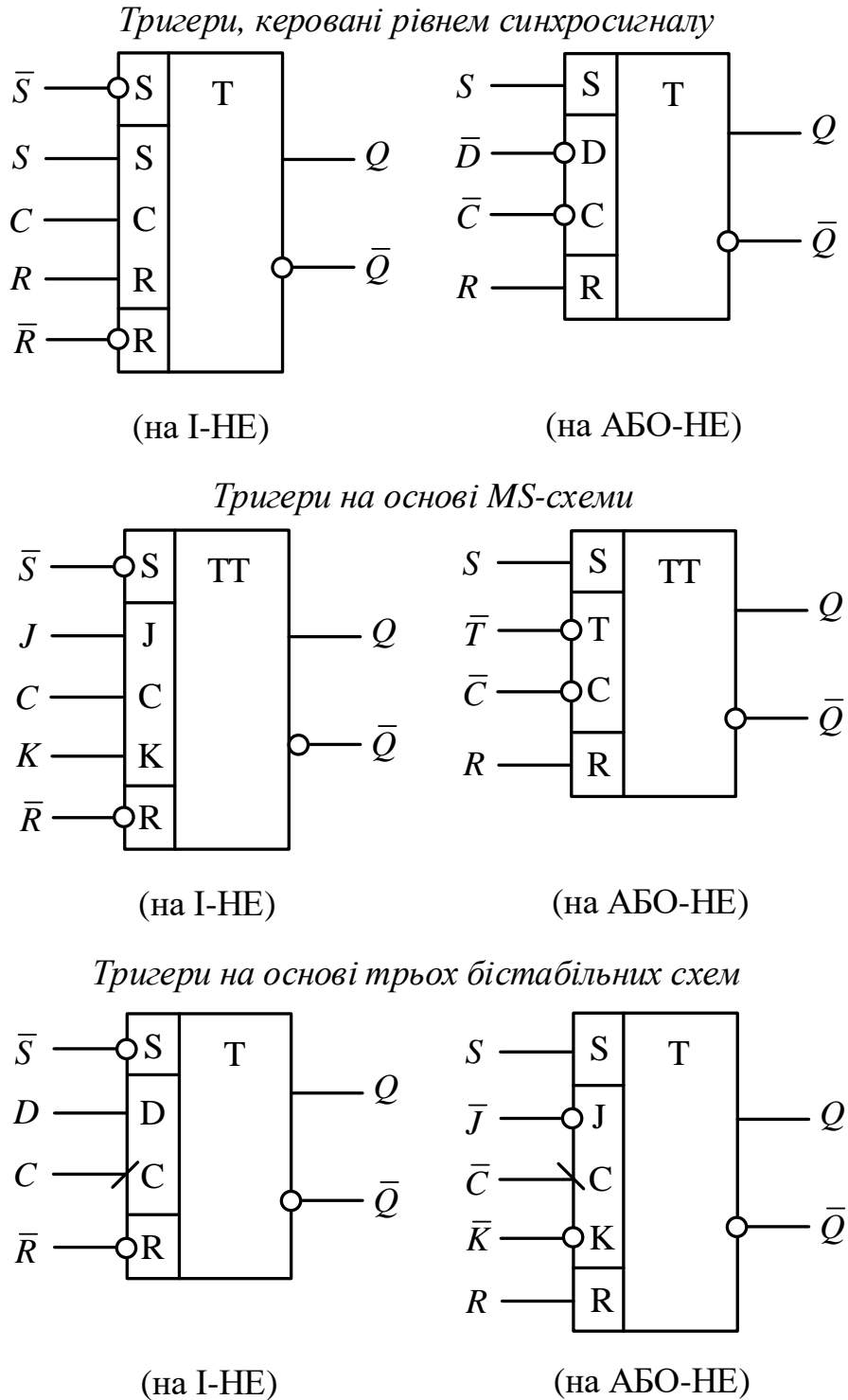
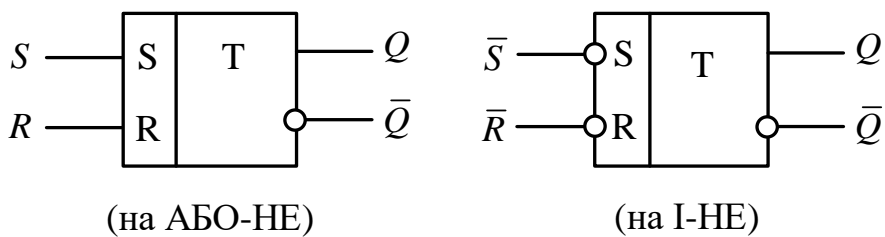


Рис. 7.35. Приклади альтернативних умовних графічних позначень синхронних тригерів

Синхровхід тригера, побудованого на основі трьох бістабільних схем, додатково позначають косою лінією з нахилом вліво, якщо перемикання тригера відбувається за заднім фронтом тактового сигналу (перехід C з 1 в 0); або з нахилом вправо, якщо перемикання тригера відбувається за переднім фронтом сигналу (перехід C з 0 в 1). Таке позначення не використовують для тактового входу тригерів, виконаних на основі MS-схеми.

Тригери, виконані на основі MS-схеми, позначають літерами ТТ.

Асинхронні тригери на основі однієї бістабільної схеми



Асинхронні тригери на основі MS-схеми із заборонними зв'язками

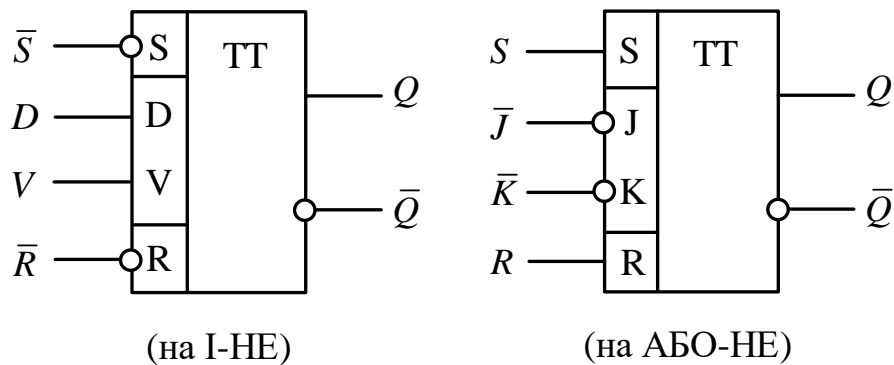


Рис. 7.36. Приклади альтернативних умовних графічних позначень асинхронних тригерів

Запитання та завдання

1. Як на умовному графічному позначенні тригера зображають деякий вхід тригера, якщо активним рівнем сигналу на цьому вході є рівень: а) логічного нуля; б) логічної одиниці?
2. Як на умовному графічному позначенні тригера на основі трьох бістабільних схем зображають вхід C синхросигналу, якщо перемикання тригера відбувається: а) за переднім фронтом сигналу C ; б) за заднім фронтом сигналу C ?

3. Сигнали на інформаційних входах JK-тригера на умовному графічному позначенні тригера позначені як \bar{J} , \bar{K} . На яких елементах побудовано тригер – І-НЕ чи АБО-НЕ?
4. Сигнали на входах асинхронного встановлення в початковий стан “0” та “1” синхронного DV-тригера на умовному графічному позначенні тригера позначено як S , R . На яких елементах побудовано тригер – І-НЕ чи АБО-НЕ?
5. Наведіть умовне графічне позначення синхронного DV-тригера, побудованою на елементах І-НЕ: а) на основі MS-схеми; б) на основі трьох бістабільних схем. Покажіть також входи асинхронного встановлення тригера в початковий стан “0”, “1”.
6. Наведіть умовне графічне позначення синхронного S-тригера на елементах АБО-НЕ, побудованого на основі MS-схеми. Покажіть також входи асинхронного встановлення тригера в початковий стан “0” та “1”.
7. Наведіть умовне графічне позначення синхронних тригерів: а) RS-тригера, керованого рівнем синхросигналу, на елементах АБО-НЕ; б) D-тригера, побудованого на основі MS-схеми, на елементах І-НЕ; в) JK-тригера, побудованого на основі MS-схеми, на елементах АБО-НЕ; г) T-тригера на елементах І-НЕ, побудованого на основі трьох бістабільних схем.
8. Наведіть УГП синхронного JK-тригера на елементах І-НЕ, побудованого: а) на основі MS-схеми; б) на основі трьох бістабільних схем. На УГП позначте також входи асинхронного встановлення тригера в початковий стан “0” та “1”.
9. Наведіть УГП синхронного D-тригера на елементах АБО-НЕ, побудованого: а) на основі MS-схеми; б) на основі трьох бістабільних схем; в) керованого рівнем синхросигналу. На УГП позначте також входи асинхронного встановлення тригера в початковий стан “0” та “1”.

7.7. Взаємозамінюваність тригерів

З’ясуємо, як, маючи тригер одного типу, на його основі отримати тригер іншого типу.

Розглянемо питання взаємозамінюваності на множині тригерів {RS-, D-, DV-, JK-, T-}.

Твердження 7.1. RS-тригер можна отримати з JK-тригера та DV-тригера.

Доведення.

1. Отримаємо RS-тригер з JK-тригера. Порівнюючи таблиці переходів RS-тригера та JK-тригера (рис. 7.37) бачимо, що JK-тригер виконує функцію RS-тригера, якщо

$$\begin{cases} J \equiv S, \\ K \equiv R, \\ J \cdot K = 0, \end{cases}$$

тобто, якщо його вхід J ототожнити з входом S, а вхід K – з входом R, але при цьому слід дотримуватися умови $J \cdot K = 0$, тобто на входи JK-тригера не подавати комбінацію сигналів $J = 1, K = 1$ (рис. 7.38).

S	R	$Q(t_{i+1})$
0	0	$Q(t_i)$
0	1	0
1	0	1
1	1	–

J	K	$Q(t_{i+1})$
0	0	$Q(t_i)$
0	1	0
1	0	1
1	1	$\bar{Q}(t_i)$

Рис. 7.37. Порівняння таблиць переходів RS- та JK-тригера

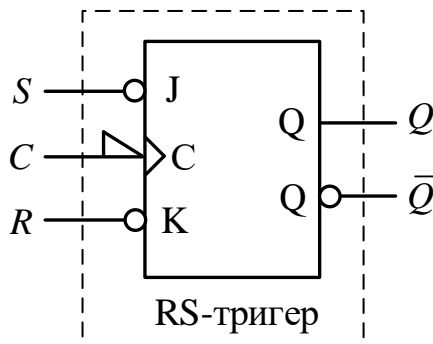


Рис. 7.38. RS-тригер на основі JK-тригера (JK-тригер побудований за MS-схемою на елементах I-HE)

Звідси випливає *наслідок* – отримати JK-тригер на основі RS-тригера неможливо, оскільки для RS-тригера дозволені лише три комбінації вхідних сигналів (комбінація $S = 1, R = 1$ заборонена), а JK-тригер має функціонувати при всіх чотирьох комбінаціях вхідних сигналів.

2. Отримаємо RS-тригер з DV-тригера. Порівняємо таблиці переходів DV-тригера та RS-тригера (рис. 7.39).

Якщо в таблиці переходів DV-тригера переставити місцями третій та четвертий рядки, то перші три значення у стовпці $Q(t_{i+1})$ співпадут з відповідними значеннями в таблиці переходів RS-тригера. При цьому співпадають стовпці S та D, а щодо стовпця V має виконуватись умова $V = S \vee R$.

Отже, DV-тригер виконує функцію RS-тригера, якщо

$$\begin{cases} D \equiv S, \\ V \equiv S \vee R. \end{cases}$$

D	V	$Q(t_{i+1})$
0	0	$Q(t_i)$
0	1	0
1	0	$Q(t_i)$
1	1	1

S	R	$Q(t_{i+1})$
0	0	$Q(t_i)$
0	1	0
1	0	1
1	1	—

D	V	$Q(t_{i+1})$
0	0	$Q(t_i)$
0	1	0
1	1	1
1	0	$Q(t_i)$

Рис. 7.39. Порівняння таблиць переходів DV- та RS-тригера

Отримані співвідношення показують, як можна пристосувати DV-тригер, щоб він виконував функцію RS-тригера (рис. 7.40).

Комбінація $S = 1, R = 1$ при цьому має бути заборонена, оскільки вона суперечить закону функціонування RS-тригера.

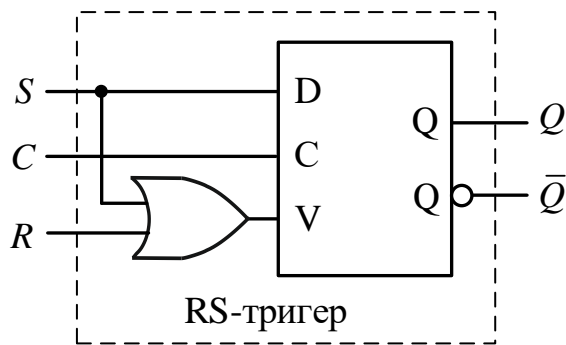


Рис. 7.40. RS-тригер на основі DV-тригера (DV-тригер, керований рівнем синхросигналу, на елементах І-НЕ)

Порівнюючи між собою таблиці переходів D- та RS-тригера (рис. 7.41) доходимо висновку, що RS-тригер виконує функцію D-тригера, якщо

$$\begin{cases} S \equiv D, \\ R \equiv \bar{D}. \end{cases} \quad (7.1)$$

Отримані співвідношення показують, як можна пристосувати RS-тригер, щоб він виконував функцію D-тригера (рис. 7.42).

2. Розглянемо, яким чином на основі JK-тригера можна отримати D-тригер.

Звідси випливає *наслідок* – отримати DV-тригер на основі RS-тригера неможливо, оскільки для RS-тригера дозволені лише 3 комбінації вхідних сигналів (комбінація $S = 1, R = 1$ заборонена), а DV-тригер має функціонувати при всіх чотирьох комбінаціях вхідних сигналів.

Твердження 7.2. D-тригер можна отримати з RS-, JK- та DV-тригера.

Доведення.

1. Розглянемо можливість отримання D-тригера на основі RS-тригера.

Таблиця переходів
D-тригера

D	$Q(t_{i+1})$
0	0
1	1

Таблиця переходів
RS-тригера

S	R	$Q(t_{i+1})$
0	0	$Q(t_i)$
0	1	0
1	0	1
1	1	—

Рис. 7.41. Порівняння таблиць переходів D- та RS-тригера

Беручи за основу співвідношення (7.1) та зважаючи на те, що вхід J JK-тригера можна ототожнити з входом S, а вхід K – з входом R, доходимо висновку, що JK-тригер виконує функцію D-тригера, якщо

$$\begin{cases} J \equiv D, \\ K \equiv \bar{D}. \end{cases} \quad (7.2)$$

Співвідношення (7.2) показує, в який спосіб можна пристосувати JK-тригер для виконання функції D-тригера (рис. 7.43).

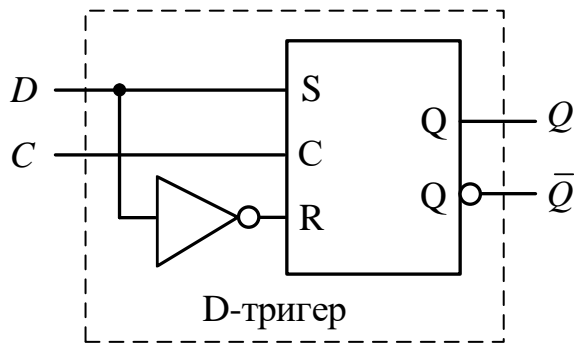


Рис. 7.42. D-тригер на основі RS-тригера (RS-тригер, керований рівнем синхросигналу, на елементах І-НЕ)

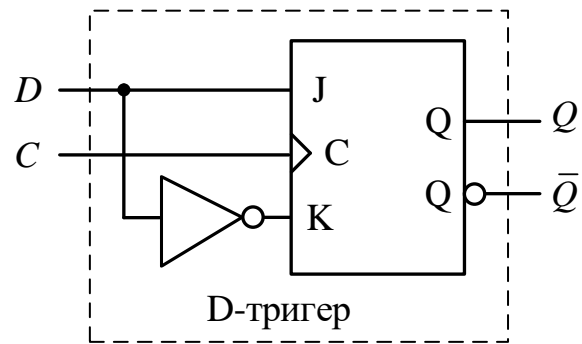


Рис. 7.43. D-тригер на основі JK-тригера (JK-тригер побудований на основі 3-х бістабільних схем на елементах І-НЕ)

3. Розглянемо, як на основі DV-тригера можна отримати D-тригер.

Аналізуючи таблицю переходів DV-тригера (табл. 7.6) бачимо, що DV-тригер виконує функцію D-тригера, якщо $V = 1$ (рис. 7.44).

Твердження 7.3. Т-тригер можна отримати з JK- та DV-тригера.

Доведення.

1. Розглянемо можливість отримання Т-тригера на основі JK-тригера. Для цього необхідно порівняти між собою таблиці переходів тригерів (рис. 7.45).

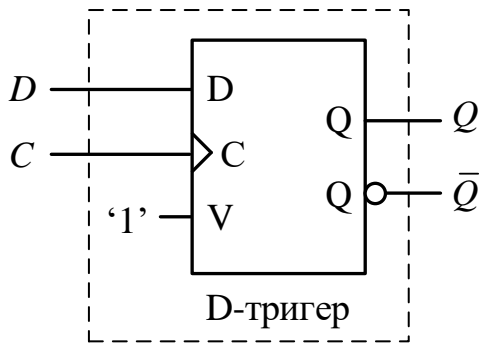


Рис. 7.44. D-тригер на основі DV-тригера (DV-тригер побудований на основі 3-х бістабільних схем на елементах І-НЕ)

Необхідною умовою того, щоб JK-тригер виконував функцію Т-тригера, є $J = K$.

Рівність $J = K$ означає, що входи J і K необхідно об'єднати.

Достатня умова:

а) JK-тригер виконує функцію Т-тригера, якщо

$$\begin{cases} (J \equiv K) \equiv T, \\ C \equiv C; \end{cases} \quad (7.3)$$

б) JK-тригер виконує функцію Т-тригера, якщо

$$\begin{cases} (J \equiv K) \equiv 1, \\ C \equiv T. \end{cases} \quad (7.4)$$

Таблиця переходів Т-тригера

T	$Q(t_{i+1})$
0	$Q(t_i)$
1	$\bar{Q}(t_i)$

Таблиця переходів JK-тригера

J	K	$Q(t_{i+1})$
0	0	$Q(t_i)$
0	1	0
1	0	1
1	1	$\bar{Q}(t_i)$

Рис. 7.45. Порівняння таблиць переходів D- та RS-тригера

Співвідношення (7.3) показує, як на основі синхронного JK-тригера отримати синхронний Т-тригер, а співвідношення (7.4) – як отримати асинхронний Т-тригер (рис. 7.46).

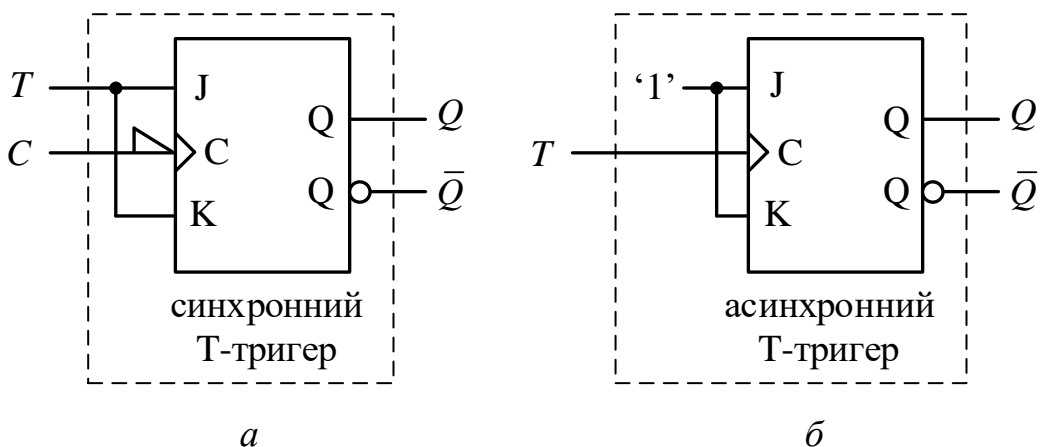


Рис. 7.46. Т-тригер на основі JK-тригера (JK-тригер побудований на елементах І-НЕ: а – за MS-схемою; б – на основі 3-х бістабільних схем)

2. Розглянемо, як на основі DV-тригера отримати Т-тригер.

Порівнюючи між собою таблиці переходів Т- та DV-тригера (рис. 7.47) доходимо висновку, що DV-тригер виконує функцію Т-тригера, якщо

$$\begin{cases} V \equiv T, \\ D \equiv \bar{Q}(t_i). \end{cases} \quad (7.5)$$

Дійсно, при $V=0$ стан DV-тригера в наступний момент часу не залежить від сигналу на вході D, а при $V=1$ стан DV-тригера є значенням сигналу, що діяв на вході D (див. рис. 7.47).

Таблиця переходів Т-тригера		Таблиця переходів DV-тригера		
T	$Q(t_{i+1})$	V	D	$Q(t_{i+1})$
0	$Q(t_i)$	0	0	$\left. \begin{matrix} Q(t_i) \\ Q(t_i) \end{matrix} \right\} Q(t_i)$
1	$\bar{Q}(t_i)$	0	1	
1	$\bar{Q}(t_i)$	1	0	$\left. \begin{matrix} 0 \\ 1 \end{matrix} \right\} D$
1	$\bar{Q}(t_i)$	1	1	

Рис. 7.47. Порівняння таблиць переходів Т- та DV-тригера

Співвідношення (7.5) показує, як пристосувати DV-тригер для виконання функції Т-тригера (рис. 7.48).

Таким чином, у підсумку отримуємо таблицю взаємозамінюваності тригерів на множині тригерів {RS-, D-, DV-, JK-, Т-} (табл. 7.17).

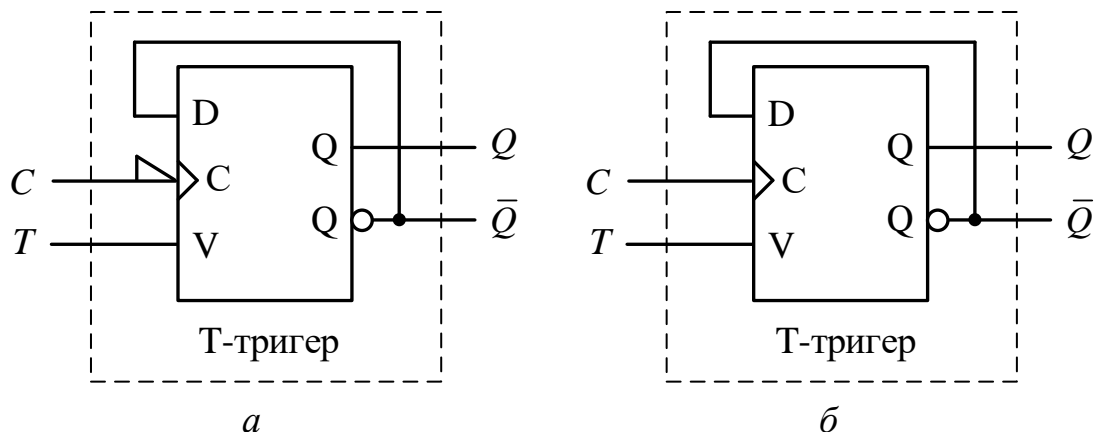


Рис. 7.48. Т-тригер на основі DV-тригера (DV-тригер побудований на елементах І-НЕ:
а – за MS-схемою; б – на основі 3-х бістабільних схем)

Аналізуючи табл. 7.17 бачимо наступне.

Рядки табл. 7.17 показують, що:

- RS-тригер можна отримати з JK- та DV-тригера,
- D-тригер можна отримати з RS-, JK- та DV-тригера,

- Т-тригер можна отримати з JK- та DV-тригера.
- Стовпці табл. 7.17 показують, що:
- з RS-тригера можна отримати D-тригер,
 - з JK-тригера можна отримати RS-, D- та Т-тригер,
 - з DV-тригера можна отримати RS-, D- та Т-тригер.

Таблиця 7.17

Таблиця взаємозамінюваності тригерів

Проекований тригер	Тригери-замінники		
	RS-	JK-	DV-
RS-		+	+
		(рис. 7.38)	(рис. 7.40)
D-	+	+	+
	(рис. 7.42)	(рис. 7.43)	(рис. 7.44)
T-	–	+	+
		(рис. 7.46)	(рис. 7.48)

Примітка. Знак “+” позначає взаємозамінюваність відповідних тригерів; “–” – відсутність взаємозамінюваності.

Додатково слід зробити також *висновок*: з Т-тригера та D-тригера не можна отримати жодного іншого типу тригера.

Таким чином, JK-тригер та DV-тригер є найбільш універсальними тригерами – з кожного з них можна отримати три інші типи тригерів, а саме: RS-, D- та Т-тригер.

Це означає, що якщо випускати у вигляді мікросхем JK-тригер, то можна отримати множину з 4-х тригерів: {JK-, RS-, D-, Т-}.

Якщо випускати у вигляді мікросхем DV-тригер, то можна отримати множину з таких 4-х тригерів: {DV-, RS-, D-, Т-}.

Якщо випускати у вигляді мікросхем два типи тригерів – JK-тригер та DV-тригер, то можна отримати множину з 5-ти тригерів: {JK-, DV-, RS-, D-, Т-}.

Запитання та завдання

1. Як отримати RS-тригер, маючи JK-тригер?
2. Дано DV-тригер. Отримати на його основі RS-тригер.
3. Як отримати D-тригер, маючи RS-тригер?
4. Дано JK-тригер. Отримати на його основі D-тригер.
5. На основі DV-тригера отримати D-тригер.
6. Як отримати асинхронний (синхронний) Т-тригер, маючи JK-тригер?
7. Дано DV-тригер. Отримати на його основі Т-тригер.
8. Навести таблицю взаємозамінюваності тригерів на множині {RS-, D-, DV-, JK-, Т-} тригерів.

9. Які тригери є найбільш універсальними?
10. Які типи тригерів можна отримати з: а) RS-тригера; б) JK-тригера; в) DV-тригера?
11. З яких тригерів можна отримати: а) RS-тригер; б) D-тригер; в) T-тригер?
12. Яку множину тригерів можна отримати на основі: а) JK-тригера; б) DV-тригера?
13. Яку множину тригерів можна отримати, маючи два типи тригерів – JK- та DV-?

8. ФУНКЦІОНАЛЬНІ ВУЗЛИ НА ОСНОВІ ТРИГЕРІВ

Винайдення тригера – пристрою (схеми), що надійно та необмежено довго може зберігати один біт, стало поворотним пунктом у розвитку комп'ютера, адже з'явилась можливість запам'ятовувати інформацію.

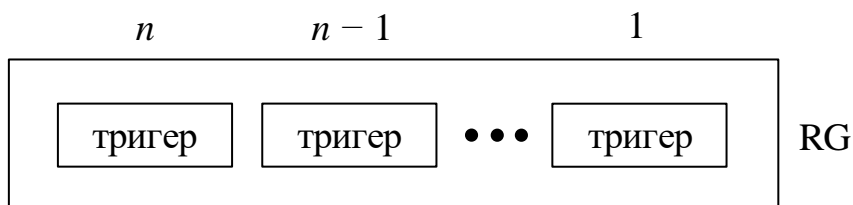
Зазвичай у комп'ютері інформацію обробляють у вигляді слів – послідовності бітів. Тому на основі тригерів будують складніші структури цифрових обчислювальних засобів – реєстри та лічильники, які здатні зберігати слово даних та забезпечувати виконання мікрооперацій над словами.

8.1. Виконання мікрооперацій на реєстрах

Реєстром називають функціональний (операційний) вузол, призначений для *тимчасового зберігання* двійкового слова та *виконання мікрооперацій* над ним.

Під *мікрооперацією* розуміють елементарну дію, що виконується за один такт.

Реєстр (RG) складається з тригерів (запам'ятовувальних елементів), кожен з яких може зберігати один біт (один розряд слова):



У реєстрі зберігають *n*-розрядне *двійкове слово*. Мікрооперації в реєстрі виконують над словом в цілому. В окремо взятому такті на реєстрі можна виконати лише одну мікрооперацію.

На вхід реєстра на рис. 8.1 надходить *n*-розрядне слово $X = (x_n x_{n-1} \dots x_1)$, виходом реєстра є слово $Z = (z_n z_{n-1} \dots z_1)$. На реєстрі виконують дві мікрооперації: WR (write) – приймання (запис) слова в реєстр, RD (read) – видача (читання) слова з реєстра.

Довжиною реєстра називають кількість розрядів (тригерів) у реєстрі. Усі розряди реєстра мають однакову будову та під час виконання мікрооперацій функціонують однаково.

У загальному випадку до складу реєстра, крім тригерів, може входити комбінаційна схема (КС) (рис. 8.2).

В узагальненій структурі реєстра використовують такі позначення: x_n, \dots, x_1 – інформаційні входи реєстра; z_n, \dots, z_1 – виходи реєстра;

y_m, \dots, y_1 – сигнали мікрооперацій; C – сигнал синхронізації; $F_{A_n}, F_{B_n}; \dots; F_{A_1}, F_{B_1}$ – входи функцій збудження тригерів.

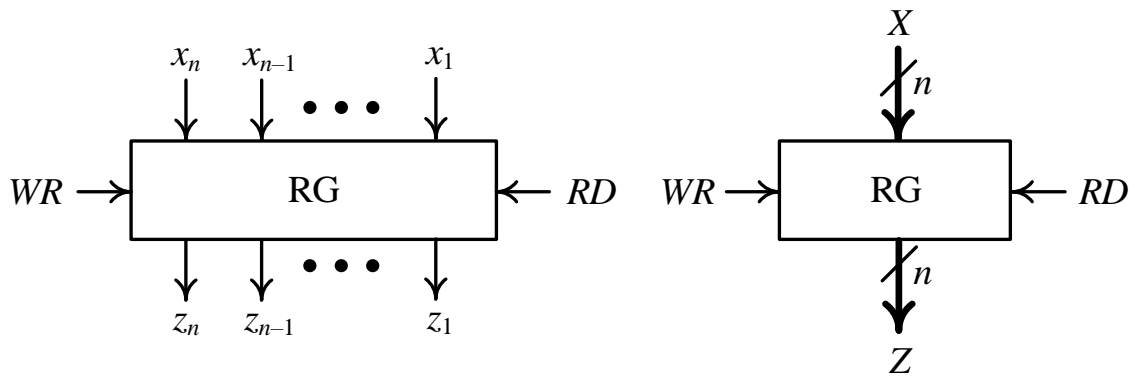


Рис. 8.1. Позначення регістра на функціональних кресленнях

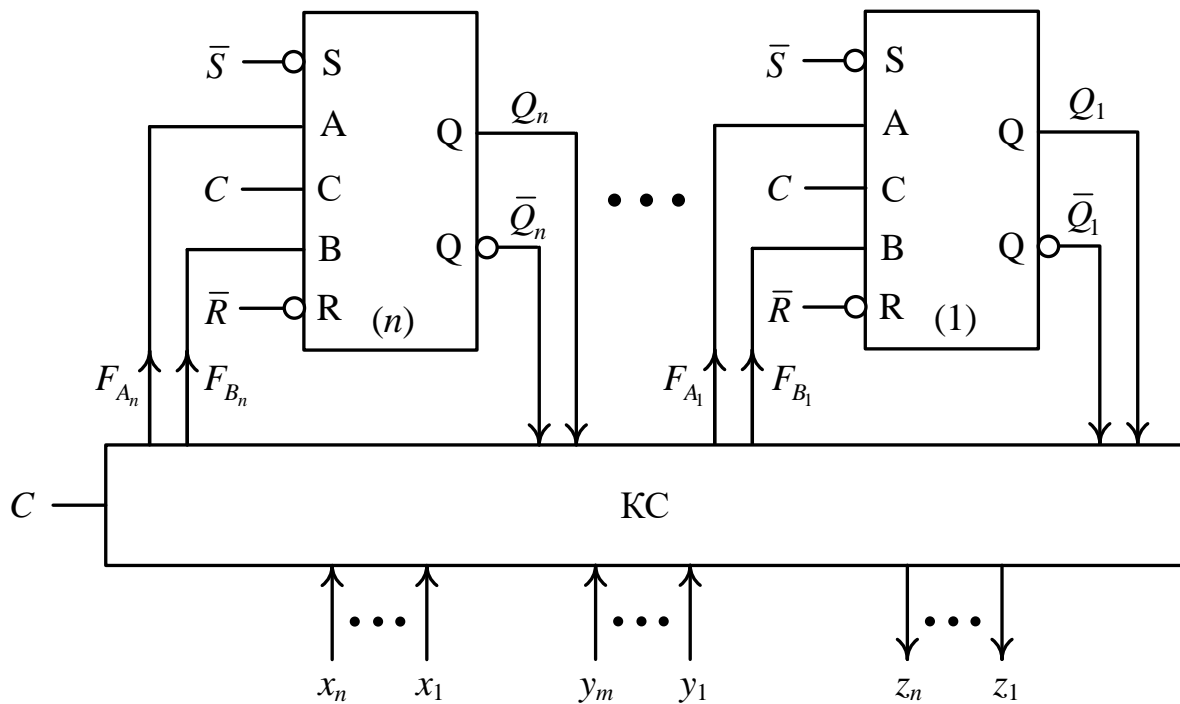


Рис. 8.2. Узагальнена структура регістра

Комбінаційна схема, що входить до складу регістра, забезпечує приймання (запис) вхідного інформаційного слова X в регістр, видачу слова Z з регістра та формує функції збудження тригерів.

Регістри будують на різноманітних тригерах: RS-, JK-, D-, T-тригерах тощо. Тригери можуть мати різну організацію – синхронні, асинхронні, з внутрішньою затримкою.

Складність комбінаційної схеми регістра залежить від виду й кількості виконуваних мікрооперацій на регістрі та від типу тригера.

Регістри, на яких реалізована мікрооперація зсуву слова, називають *регiстрами зсуву* (зсувними регiстрами). Зсув може здійснюватися вліво (убік старших розрядів) або вправо (убік молодших розрядів).

Регістри, що забезпечують зсув слова як вліво, так і вправо, називають *реверсивними регiстрами*.

Найчастіше на регiстрах виконують такі мікрооперації:

- RESET(y_1) – встановлення початкового стану регiстра “0...0”;
- SET(y_2) – встановлення початкового стану регiстра “1...1”;
- WRITE(y_3) – запис (приймання) слова в регiстр;
- AND(y_4) – порозрядна кон'юнкція двох слів (одне слово знаходиться в регiстрі, інше – надходить на інформаційні входи регiстра, а результат мікрооперації зберігається в регiстрі);
- OR(y_5) – порозрядна диз'юнкція двох слів;
- XOR(y_6) – порозрядна сума двох слів за модулем два (**eXclusive OR**);
- COM(y_7) – інвертування слова (**COMplement**);
- SLL(y_8) – логічний зсув слова на один розряд вліво (**Shift Left Logical**);
- SRL(y_9) – логічний зсув слова на один розряд вправо (**Shift Right Logical**);
- SLA(y_{10}) – арифметичний зсув слова на один розряд вліво (**Shift Left Arithmetic**);
- SRA(y_{11}) – арифметичний зсув слова на один розряд вправо (**Shift Right Arithmetic**);
- RL(y_{12}) – циклічний зсув слова на один розряд вліво (**Rotation Left**);
- RR(y_{13}) – циклічний зсув слова на один розряд вправо (**Rotation Right**);
- READ(y_{14}) – видача слова прямим кодом;
- RDCOM(y_{15}) – видача слова інверсним кодом (**ReaD COMplement**);
- RDTRC(y_{16}) – видача слова парафазним кодом (**ReaD Two Rail Code**).

Характеристика мікрооперацій

Мікрооперація RESET(y_1) – встановлення всіх розрядів регiстра в “0”

Під час виконання мікрооперації RESET на входи \bar{R} (вхід асинхронного встановлення тригера в початковий стан “0”) всіх тригерів регiстра одночасно подається логічний нуль ($\bar{R} = 0$) (рис. 8.3), а $\bar{S} = 1$.

Як наслідок, всі тригери регiстра встановлюються в стан “0”. Регiстр міститиме n -розрядне слово ‘00 ... 0’. Мікрооперація RESET реалізує оператор присвоювання $RG := 00 \dots 0$ ($RG := \emptyset$).

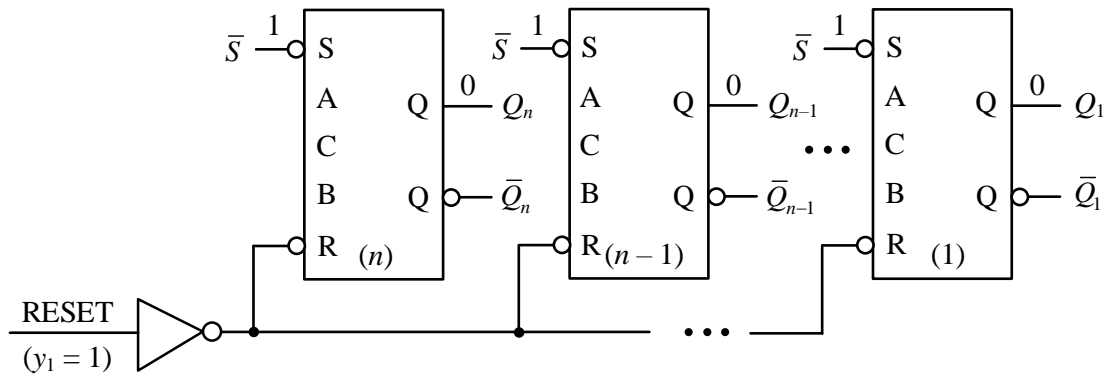


Рис. 8.3. Схемотехнічна реалізація мікрооперації RESET(y_1) на регістрі

Мікрооперація SET(y_2) – встановлення всіх розрядів регістра в “1”

Під час виконання мікрооперації SET на входи \bar{S} (вхід асинхронного встановлення тригера в початковий стан “1”) всіх тригерів регістра одночасно подається логічний нуль ($\bar{S} = 0$) (рис. 8.4), а $\bar{R} = 1$.

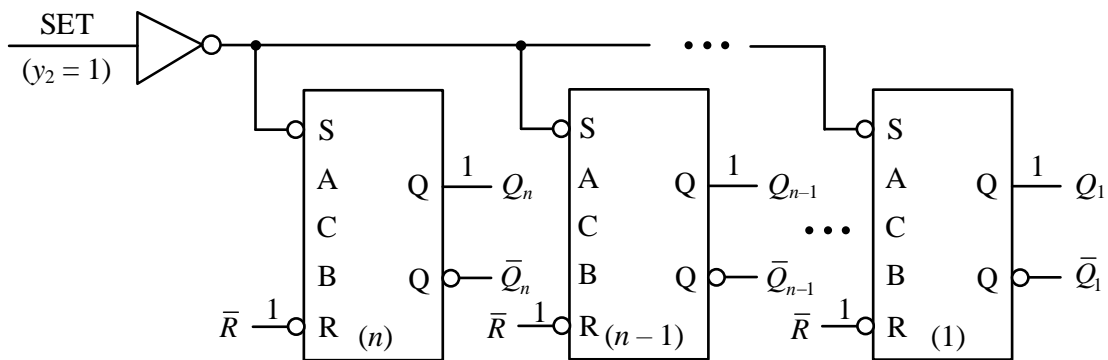


Рис. 8.4. Схемотехнічна реалізація мікрооперації SET на регістрі

Як наслідок, всі тригери регістра встановлюються в стан “1”. Регістр міститиме n -розрядне слово ‘11 ... 1’. Мікрооперація SET реалізує оператор присвоювання $RG := 11 \dots 1$.

Мікрооперація WRITE(y_3) – запис слова в регістр

Під час виконання мікрооперації WRITE вхідне слово $X = (x_n x_{n-1} \dots x_1)$ записується в регістр.

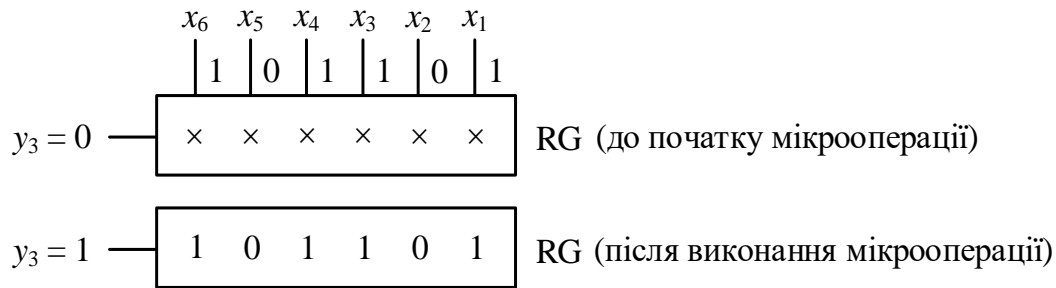
Мікрооперацію запису слова в регістр описують рівнянням

$$Q_j(t_{i+1}) = x_j(t_i),$$

де $x_j(t_i)$ – значення сигналу на j -му вході регістра в момент часу t_i , $Q_j(t_{i+1})$ – стан (вихід) j -го тригера (розряду) регістра після виконання мікрооперації, $j = 1, 2, \dots, n$.

Мікрооперація WRITE реалізує оператор присвоювання $RG := (X)$.

Наприклад, для 6-розрядного слова $X = 101101$ процес запису слова в регістр виглядає так:



Мікрооперація AND(y_4) – порозрядна кон'юнкція двох слів

Розглянемо приклади порозрядної кон'юнкції двох слів:

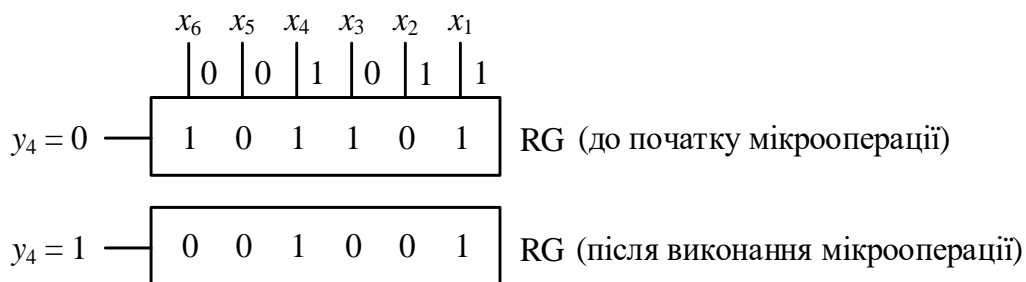
$$\begin{array}{r} \& \quad 1\ 0\ 1\ 1\ 0\ 1 \\ \quad 0\ 0\ 1\ 0\ 1\ 1 \\ \hline \quad 0\ 0\ 1\ 0\ 0\ 1 \end{array} \qquad \begin{array}{r} \& \quad 1\ 1\ 0\ 1\ 1\ 0 \\ \quad 1\ 1\ 1\ 0\ 0\ 1 \\ \hline \quad 1\ 1\ 0\ 0\ 0\ 0 \end{array}$$

Порозрядну кон'юнкцію двох слів, одне з яких (Q) знаходиться в регістрі, а інше (X) надходить на інформаційні входи регістра, описують рівнянням

$$Q_j(t_{i+1}) = Q_j(t_i) \& x_j(t_i),$$

де $Q_j(t_i)$ – розряд слова, що знаходиться в j -му тригері регістра, $x_j(t_i)$ – розряд слова, що надходить на j -й вхід регістра.

Наприклад, якщо слово 101101 зберігається в регістрі, а слово $X = 001011$ надходить на входи регістра, то процес виконання мікрооперації AND (y_4) виглядає так:



Мікрооперація AND реалізує оператор $RG := (RG) \& X$.

Мікрооперація OR(y_5) – порозрядна диз'юнкція двох слів

Порозрядну диз'юнкцію двох слів виконують наступним чином (приклади):

$$\begin{array}{r} \vee \quad 1\ 0\ 1\ 1\ 0\ 1 \\ \quad 0\ 0\ 1\ 0\ 1\ 1 \\ \hline \quad 1\ 0\ 1\ 1\ 1\ 1 \end{array} \qquad \begin{array}{r} \vee \quad 1\ 1\ 0\ 1\ 1\ 0 \\ \quad 1\ 1\ 1\ 0\ 0\ 1 \\ \hline \quad 1\ 1\ 1\ 1\ 1\ 1 \end{array}$$

Порозрядну диз'юнкцію двох слів, одне з яких (Q) знаходиться в регістрі, а інше (X) надходить на інформаційні входи регістра, описують рівнянням

$$Q_j(t_{i+1}) = Q_j(t_i) \vee x_j(t_i),$$

де $Q_j(t_i)$ – розряд слова, що знаходиться в j -му тригері регістра, $x_j(t_i)$ – розряд слова, що надходить на j -й вхід регістра.

Наприклад, якщо слово 101101 зберігається в регістрі, а слово $X = 001011$ надходить на входи регістра, то процес виконання мікрооперації OR(y_5) відбувається так:

		x_6	x_5	x_4	x_3	x_2	x_1	
		0	0	1	0	1	1	
$y_5 = 0$	—	1 0 1 1 0 1						RG (до початку мікрооперації)
$y_5 = 1$	—	1 0 1 1 1 1						RG (після виконання мікрооперації)

Мікрооперація OR реалізує оператор $RG := (RG) \text{ OR } (X)$.

Мікрооперація XOR(y_6) – порозрядна сума за модулем два двох слів

Розглянемо два приклади порозрядного додавання за модулем два двох слів:

\oplus	$\begin{array}{r} 101101 \\ 001011 \\ \hline 100110 \end{array}$	\oplus	$\begin{array}{r} 110110 \\ 111001 \\ \hline 001111 \end{array}$
----------	--	----------	--

Порозрядне додавання за модулем два двох слів, одне з яких (Q) знаходиться в регістрі, а інше (X) надходить на інформаційні входи регістра, описують рівнянням

$$Q_j(t_{i+1}) = Q_j(t_i) \oplus x_j(t_i).$$

Наприклад, якщо слово 101101 знаходиться в регістрі, а слово $X = 001011$ надходить на входи регістра, то процес виконання мікрооперації відбувається так:

		x_6	x_5	x_4	x_3	x_2	x_1	
		0	0	1	0	1	1	
$y_6 = 0$	—	1 0 1 1 0 1						RG (до початку мікрооперації)
$y_6 = 1$	—	1 0 0 1 1 0						RG (після виконання мікрооперації)

Мікрооперація XOR реалізує оператор $RG := (RG) \text{ XOR } (X)$.

Мікрооперація COM(y₇) – інвертування слова

Інвертування слова – це заміна значення біта в кожному розряді на протилежне значення: 0 на 1, 1 на 0.

Наприклад,

$$\text{інвертування } \frac{1\ 0\ 1\ 1\ 0\ 1}{0\ 1\ 0\ 0\ 1\ 0}$$

$$\text{інвертування } \frac{1\ 1\ 0\ 1\ 1\ 0}{0\ 0\ 1\ 0\ 0\ 1}$$

Інвертування слова в регістрі описують рівнянням

$$Q_j(t_{i+1}) = \bar{Q}_j(t_i).$$

Так, якщо в регістрі зберігається слово 101101, то процес його інвертування виглядає так:

$$y_7 = 0 \text{ — } \boxed{1\ 0\ 1\ 1\ 0\ 1} \text{ RG (до початку мікрооперації)}$$

$$y_7 = 1 \text{ — } \boxed{0\ 1\ 0\ 0\ 1\ 0} \text{ RG (після виконання мікрооперації)}$$

Особливістю мікрооперації COM є те, що в ній не використовується вхідне слово X.

Мікрооперація COM реалізує оператор RG:= NOT (RG).

Мікрооперація SLL(y₈) – логічний зсув слова на один розряд вліво

Розглянемо приклади логічного зсуву слова вліво:

$$\text{втрачається } \leftarrow \frac{1\ 0\ 1\ 1\ 0\ 1}{\text{SLL } \begin{array}{c} \swarrow \swarrow \swarrow \swarrow \swarrow \\ 0\ 1\ 1\ 0\ 1\ 0 \end{array} \leftarrow \boxed{0}}$$

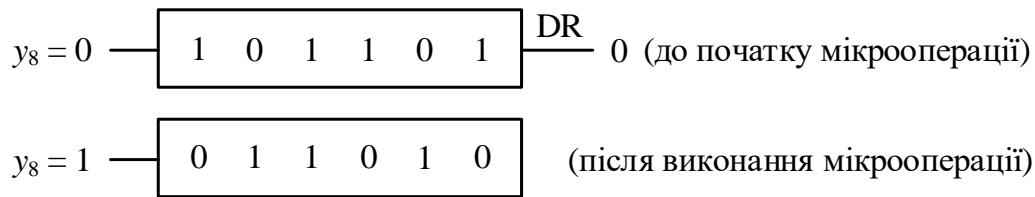
$$\text{втрачається } \leftarrow \frac{1\ 1\ 0\ 1\ 1\ 0}{\text{SLL } \begin{array}{c} \swarrow \swarrow \swarrow \swarrow \swarrow \\ 1\ 0\ 1\ 1\ 0\ 1 \end{array} \leftarrow \boxed{1}}$$

Під час зсуву слова на один розряд вліво старший (лівий) розряд втрачається, а молодший (правий) може бути заповнений або нулем, або одиницею. Для заповнення молодшого розряду регістр має спеціальний вхід заповнення молодшого розряду при зсуві вліво DR (data right). Перед зсувом слова на вході DR задають потрібне значення – 0 або 1.

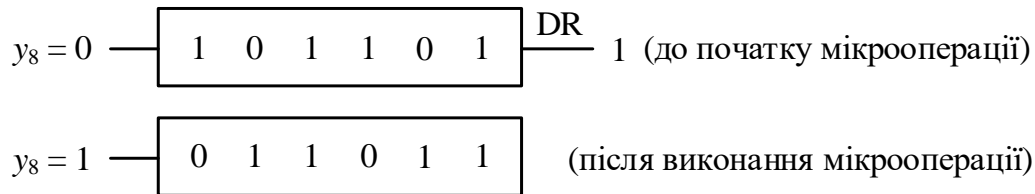
Мікрооперацію SLL описують системою рівнянь

$$\begin{cases} Q_j(t_{i+1}) = Q_{j-1}(t_i), j \neq 1, \\ Q_1(t_{i+1}) = DR. \end{cases}$$

Наприклад, якщо в регістрі зберігається слово 101101, а на вході DR регістра встановлено 0, то процес логічного зсуву слова вліво виглядає так:

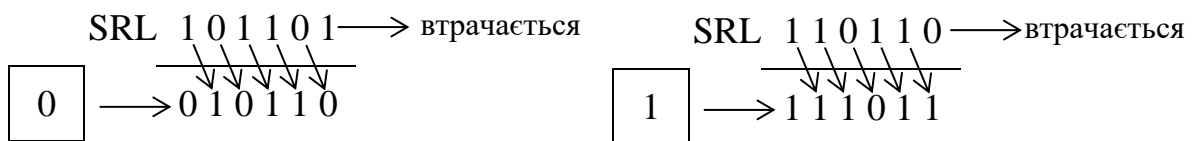


Якщо в регістрі зберігається те саме слово, а на вході DR встановлено 1, то процес зсуву вліво має вигляд:



Вхідне слово X в мікрооперації SLL не використовується. Мікрооперація SLL реалізує оператор $RG := SLL(RG)$.

Мікрооперація SRL(y_9) – логічний зсув слова на один розряд вправо
 Розглянемо приклади логічного зсуву слова вправо:

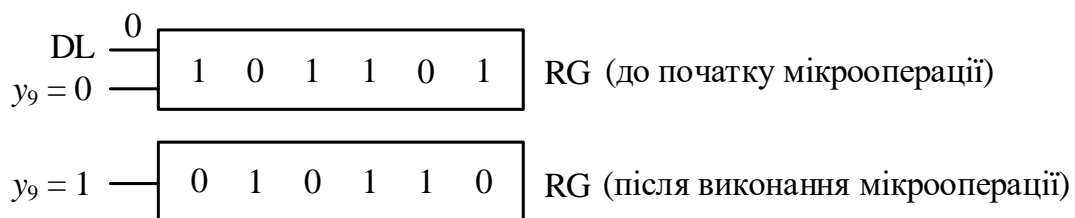


Під час зсуву слова вправо молодший (правий) розряд втрачається, а старший (лівий) може бути заповнений нулем або одиницею. Для заповнення старшого розряду регістр має спеціальний вхід заповнення старшого розряду при зсуві вправо DL (data left). Перед зсувом слова на вході DL задають потрібне значення – 0 або 1.

Мікрооперацію SRL описують системою рівнянь

$$\begin{cases} Q_j(t_{i+1}) = Q_{j+1}(t_i), j \neq n, \\ Q_n(t_{i+1}) = DL. \end{cases}$$

Наприклад, якщо в регістрі зберігається слово 101101, а на вході DL встановлено 0, то процес зсуву вправо відбувається так:



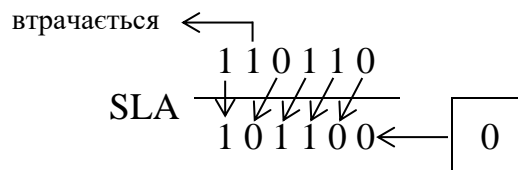
Вхідне слово X в мікрооперації SRL не використовується.

Мікрооперація SRL реалізує оператор $RG:= SRL (RG)$.

Мікрооперація $SLA(y_{10})$ – арифметичний зсув слова на один розряд вліво

При виконанні мікрооперації SLA слово розглядається як число зі знаком у доповняльному коді. Знаком числа вважається значення старшого (n -го) розряду. Тому у зсуві беруть участь всі розряди слова, крім старшого розряду, який має лишатися незмінним; при цьому $(n - 1)$ -й розряд втрачається, а молодший розряд регістра заповнюється нулем.

Схема арифметичного зсуву слова вліво на один розряд на прикладі 6-розрядного слова виглядає так:



Мікрооперацію SLA описують системою рівнянь

$$\begin{cases} Q_j(t_{i+1}) = Q_{j-1}(t_i), j \neq n, j \neq 1, \\ Q_n(t_{i+1}) = Q_n(t_i), \\ Q_1(t_{i+1}) = 0. \end{cases}$$

Наприклад, якщо в регістрі зберігається слово 101101, то процес виконання мікрооперації SLA відбувається так:

$y_{10} = 0$ — 1 0 1 1 0 1 RG (до початку мікрооперації)

$y_{10} = 1$ — 1 1 1 0 1 0 RG (після виконання мікрооперації)

Вхідне слово X в мікрооперації SLA не використується.

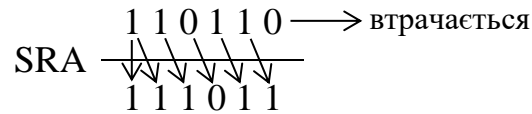
Мікрооперація SLA реалізує оператор $RG:= SLA (RG)$.

Мікрооперація $SRA(y_{11})$ – арифметичний зсув слова на один розряд вправо

Як і у випадку мікрооперації SLA при виконанні мікрооперації SRA слово розглядають як число зі знаком у доповняльному коді, а знаком числа вважається значення старшого (n -го) розряду. При арифметичному зсуві слова вправо молодший розряд слова втрачається, старший (n -й) розряд залишається незмінним, а $(n - 1)$ -й розряд заповнюється значенням n -го (знакового) розряду слова.

Заповнення $(n - 1)$ -го розряду значенням n -го (знакового) розряду слова називають *поширенням знакового розряду вправо*.

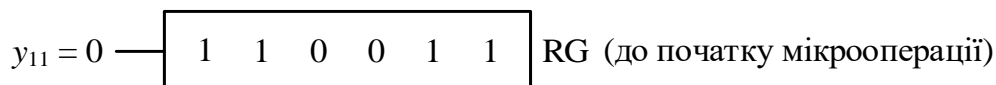
Схема арифметичного зсуву слова вправо на один розряд на прикладі 6-розрядного слова:



Мікрооперацію SRA описують системою рівнянь

$$\begin{cases} Q_j(t_{i+1}) = Q_{j+1}(t_i), j \neq n, \\ Q_n(t_{i+1}) = Q_n(t_i). \end{cases}$$

Наприклад, якщо в регістрі зберігається слово 110011, то процес виконання мікрооперації SRA виглядає так:



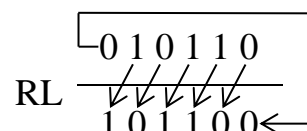
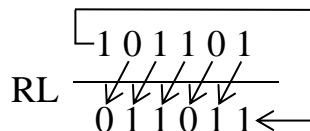
Вхідне слово X в мікрооперації SRA не використовується.

Мікрооперація SRA реалізує оператор $\text{RG} := \text{SRA}(\text{RG})$.

Мікрооперація RL(y_{12}) – циклічний зсув слова на один розряд вліво

Під час циклічного зсуву слова на один розряд вліво молодший розряд, що вивільняється, заповнюють значенням старшого (n -го) розряду.

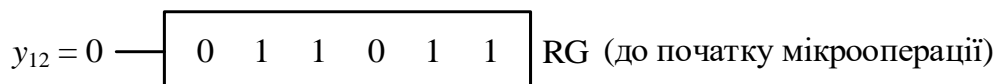
Приклади циклічного зсуву слова на один розряд вліво:



Мікрооперацію RL описують системою рівнянь

$$\begin{cases} Q_j(t_{i+1}) = Q_{j-1}(t_i), j \neq 1, \\ Q_1(t_{i+1}) = Q_n(t_i). \end{cases}$$

Наприклад, якщо в регістрі зберігається слово 011011, то процес циклічного зсуву слова вліво на один розряд відбувається так:



Мікрооперація RL реалізує оператор $\text{RG} := \text{RL}(\text{RG})$.

Мікрооперація RR(y_{13}) – циклічний зсув слова вправо на один розряд

Під час циклічного зсуву слова на один розряд вправо старший (n -й) розряд, що вивільняється, заповнюють значенням молодшого розряду.

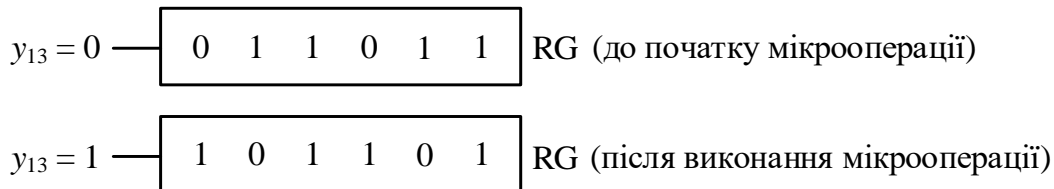
Наприклад,



Мікрооперацію RR описують системою рівнянь

$$\begin{cases} Q_j(t_{i+1}) = Q_{j+1}(t_i), j \neq n, \\ Q_n(t_{i+1}) = Q_1(t_i). \end{cases}$$

Так, якщо в регістрі зберігається слово 011011, то процес циклічного зсуву слова вправо на один розряд відбувається в такий спосіб:



Мікрооперація RR реалізує оператор $RG := RR(RG)$.

Часові діаграми виконання деяких мікрооперацій на регістрі наведено на рис. 8.5.

Мікрооперації видачі слова з регістра

Існують три мікрооперації видачі слова з регістра: READ – видача слова прямим кодом (read), RDCOM – видача слова інверсним кодом (read complement), RDTRC – видача слова парафазним кодом (read two rail code).

Слід розуміти, що стан регістра після видачі слова не змінюється.

Мікрооперація READ(y_{14}) – видача слова прямим кодом

Процес виконання мікрооперації READ характеризує рис. 8.6. Для видачі слова використовується лінійка 2-входових елементів І. На перші входи всіх елементів І надходить сигнал y_{14} мікрооперації, на другі входи – сигнали з прямих виходів Q_j ($j = 1, 2, \dots, n$) тригерів, що входять до складу регістра.

Якщо $y_{14} = 0$, то мікрооперація не виконується; при $y_{14} = 1$ на виходах елементів І отримуємо слово, що зберігається в регістрі. При цьому вміст регістра (стан тригерів) не змінюється. Відбувається ніби копіювання слова, що міститься в регістрі, на виходи регістра.

Мікрооперацію READ(y_{14}) описують рівнянням

$$z_j = y_{14} Q_j, \quad j = 1, 2, \dots, n.$$

Якщо $y_{14} = 1$, то $z_j \equiv Q_j$.

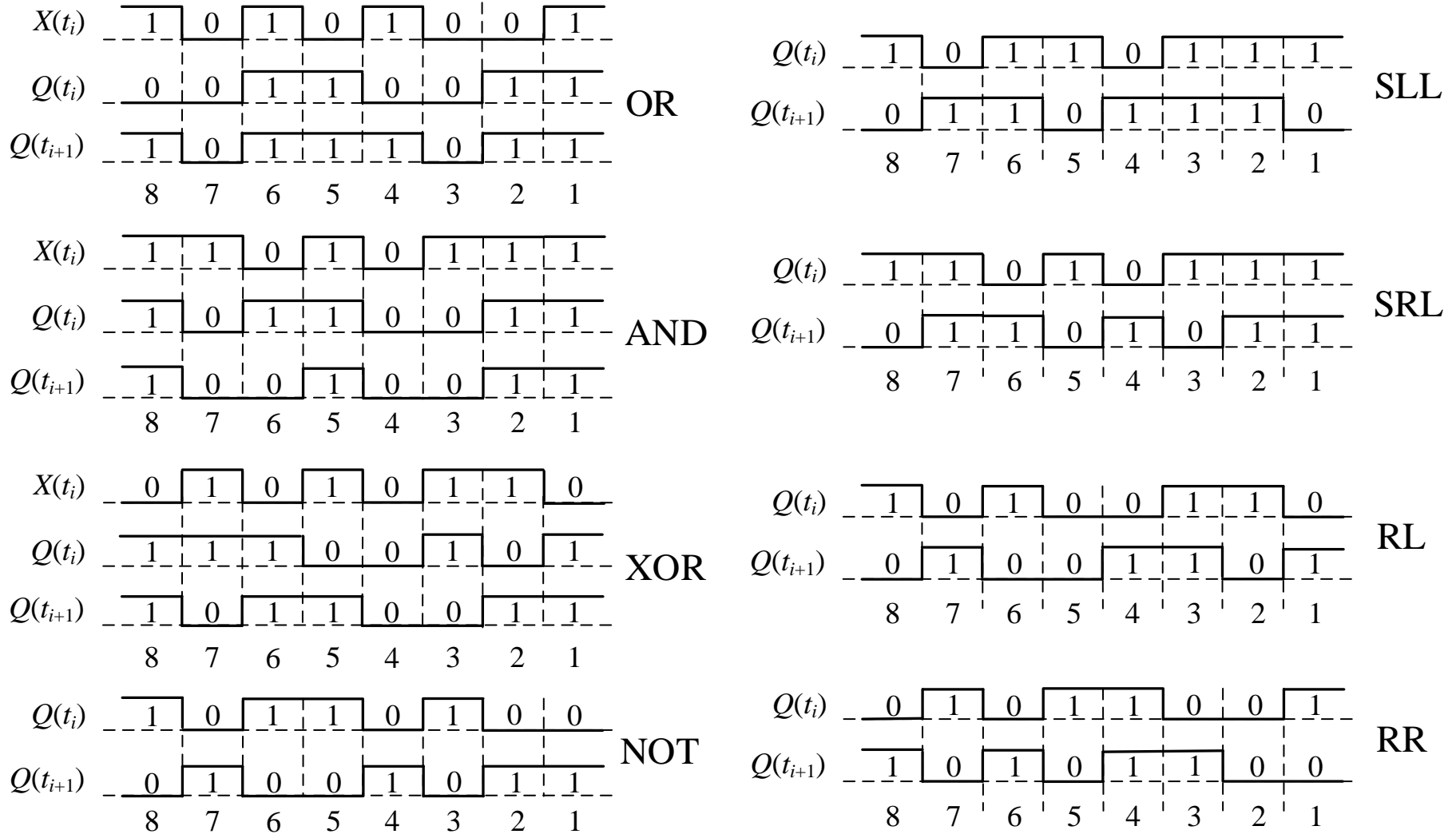


Рис. 8.5. Часові діаграми виконання деяких мікрооперацій на 8-розрядному регістрі

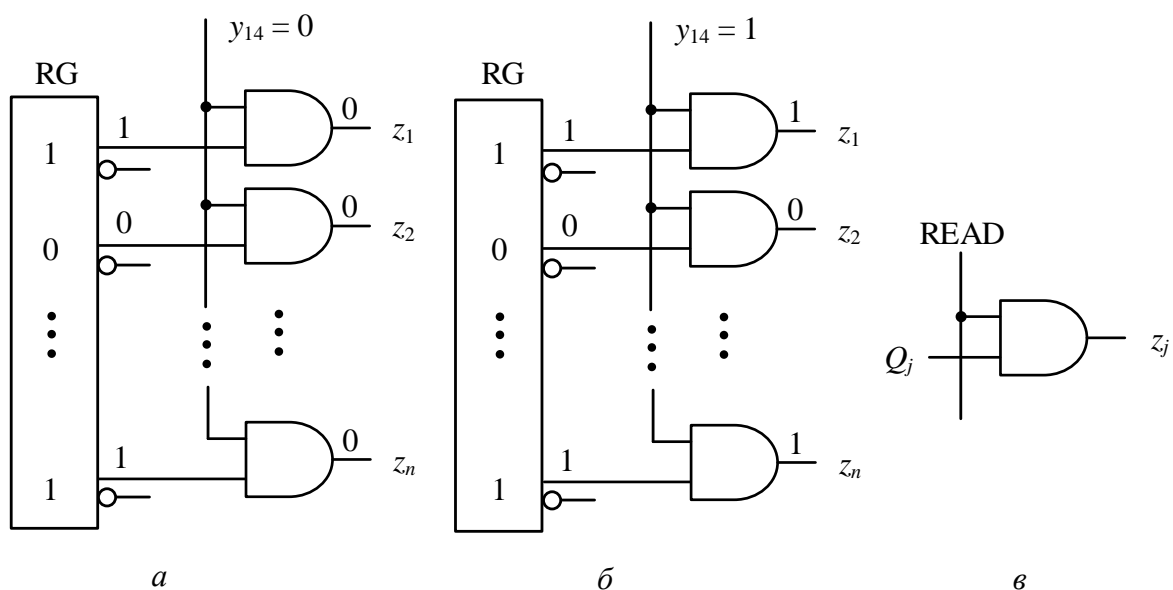


Рис. 8.6. Процес виконання мікрооперації READ(y_{14}) – видача слова з регістра прямим кодом: а – до початку мікрооперації; б – після виконання мікрооперації; в – схема видачі одного розряду слова

Мікрооперація RDCOM(y_{15}) – видача слова в інверсному коді

Рис. 8.7 характеризує процес виконання мікрооперації RDCOM. Для видачі слова інверсним кодом використовують двовходові елементи І. На перші входи всіх елементів І подають сигнал y_{15} мікрооперації, а на другі входи – сигнали з інверсних виходів \bar{Q}_j ($j = 1, 2, \dots, n$) тригерів, що входять до складу регістра.

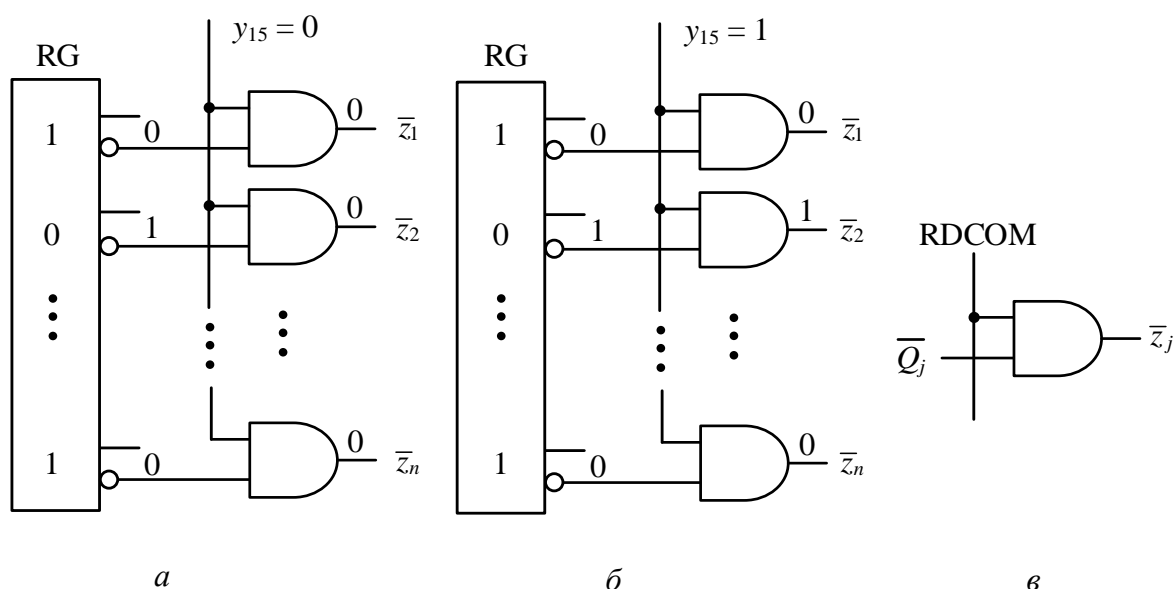


Рис. 8.7. Процес виконання мікрооперації RDCOM(y_{15}) – видача слова з регістра інверсним кодом: а – до початку мікрооперації; б – після виконання мікрооперації; в – схема видачі одного розряду слова

Якщо $y_{15} = 0$, то мікрооперація не виконується; при $y_{15} = 1$ на виходах елементів І отримуємо інверсію слова, що зберігається в регістрі. При цьому вміст регістра (стан тригерів) не змінюється.

Мікрооперацію RDCOM (y_{14}) описують рівнянням

$$z_j = y_{15} \bar{Q}_j, \quad j = 1, 2, \dots, n.$$

Якщо $y_{15} = 1$, то $z_j \equiv \bar{Q}_j$.

Мікрооперація RDTRC(y_{16}) – видача слова парафазним кодом

Процес виконання мікрооперації RDTRC характеризує рис. 8.8.

Для видачі слова парафазним кодом використовують $2n$ двохходових елементів І – по два елементи І на кожен розряд регістра.

Парафазний – означає, що в кожному розряді слова, яке зберігається в регістрі, видається як пряме значення біта, так і його інверсне значення.

Наприклад,

0	1	1	0	1	– прямий код слова 01101
01	10	10	01	10	– парафазний код слова

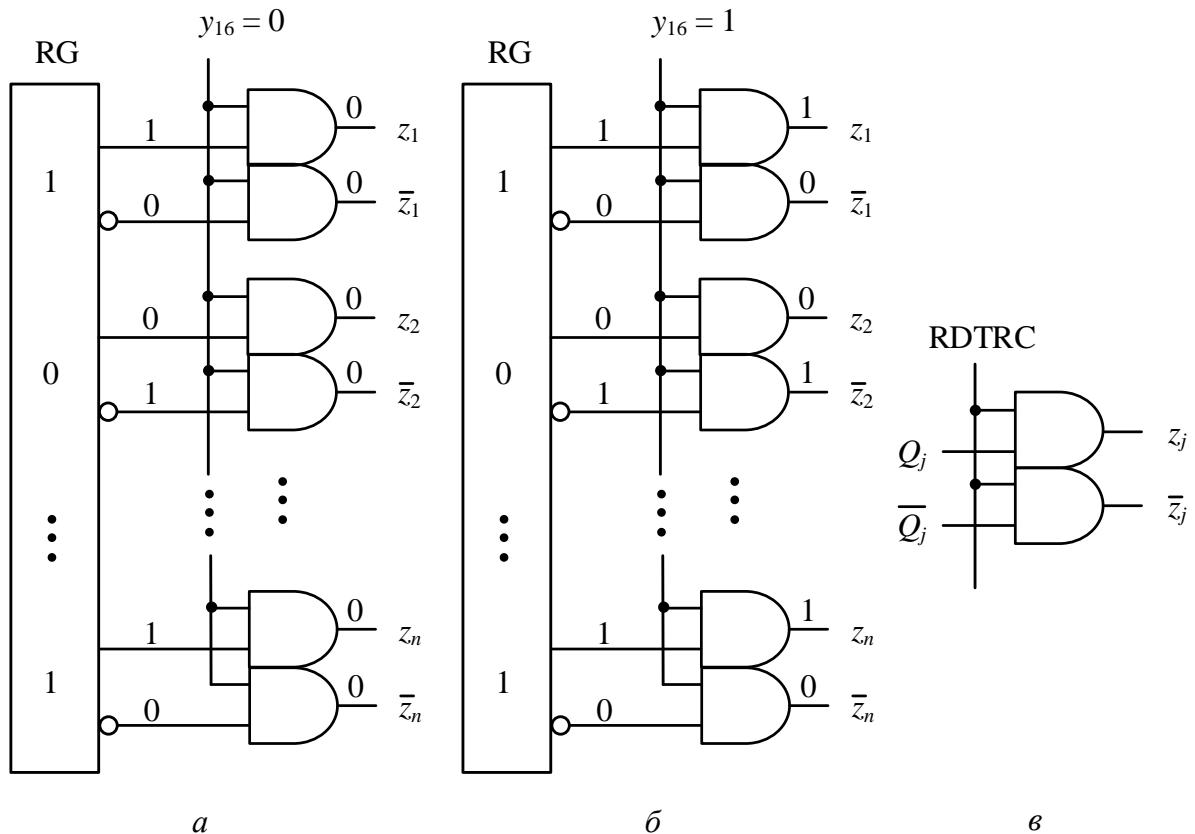


Рис. 8.8. Процес виконання мікрооперації RDTRC(y_{16}) – видача слова з регістра парафазним кодом: *a* – до початку мікрооперації; *б* – після виконання мікрооперації; *в* – схема видачі одного розряду слова

Якщо $y_{16} = 0$, то мікрооперація не виконується; при $y_{16} = 1$ на виходах елементів І отримуємо парафазний код слова, що міститься в регістрі. При цьому вміст регістра (стан тригерів) не змінюється.

Мікрооперацію $RDTRC(y_{16})$ описують системою

$$\begin{cases} z_j = y_{16} Q_j, \\ \bar{z}_j = y_{16} \bar{Q}_j. \end{cases}$$

$$\text{Якщо } y_{16} = 1, \text{ то } \begin{cases} z_j \equiv Q_j, \\ \bar{z}_j \equiv \bar{Q}_j. \end{cases}$$

Запитання та завдання

1. Який функціональний вузол називають регістром?
2. Дайте визначення мікрооперації.
3. Охарактеризуйте узагальнену структуру регістра.
4. Що називають довжиною регістра?
5. Для чого призначена комбінаційна схема, що входить до складу регістра?
6. Які регістри називають: а) регістрами зсуву; б) реверсивними регістрами?
7. Які: а) логічні мікрооперації; б) мікрооперації зсуву; в) мікрооперації видачі слова – виконують на регістрах?
8. У регістрі зберігається слово 10011, а на вхід регістра надходить слово 10110. Виконайте над цими словами мікрооперації AND, OR, XOR.
9. У регістрі зберігається слово 10101. Послідовно виконайте над ним мікрооперації: COM, SLL, SRL, SRA, SLA.
10. Чим відрізняється логічний зсув слова від арифметичного зсуву?
11. Дано слово 1011. Утворіть парафазний код заданого слова.

8.2. Синтез синхронних регістрів

Проектування регістра на тригерах заданого типу зводиться до синтезу комбінаційної схеми, яка формує функції збудження тригерів під час виконання заданих мікрооперацій, а також забезпечує введення (запис) слова в регістр та видачу (читання) слова з регістра.

Для мікрооперацій видачі слова з регістра синтез комбінаційної схеми як такий не виконують, при цьому використовується готова відповідна схема видачі (див. рис. 8.6 – 8.8).

Під час синтезу регістрів мінімальною структурною одиницею є тригер (а не бістабільна схема). Тому необхідно оперувати з таблицями функцій збудження тригерів (а не бістабільних схем).

Отже, слід розрізняти функції збудження f_2, f_1 бістабільної схеми (на І-НЕ або АБО-НЕ) та функції збудження тригера: F_R, F_S – RS-тригера; F_D – D-тригера; F_J, F_K – JK-тригера; F_T – Т-тригера тощо.

Визначимо функції F_R, F_S збудження RS-тригера.

Для цього необхідно скористатися таблицею переходів RS-тригера (рис. 8.9). Доповнивши таблицю переходів RS-тригера стовпцем $Q(t_i)$ отримаємо повну таблицю переходів RS-тригера (таблиця 2).

Таблиця переходів RS-тригера (таблиця 1)

$S(t_i)$	$R(t_i)$	$Q(t_{i+1})$
0	0	$Q(t_i)$
0	1	0
1	0	1
1	1	–

Повна таблиця переходів RS-тригера (таблиця 2)

Номер рядка	$S(t_i)$	$R(t_i)$	$Q(t_i)$	$Q(t_{i+1})$
1	0	0	0	→ 0
2	0	0	1	→ 1
3	0	1	0	→ 0
4	0	1	1	---→ 0
5	1	0	0	~→ 1
6	1	0	1	→ 1
7	1	1	0	–
8	1	1	1	–

Таблиця функцій збудження RS-тригера (таблиця 3)

$Q(t_i)$	$Q(t_{i+1})$	F_S	F_R
0	0	0	×
0	1	1	0
1	0	0	1
1	1	×	0

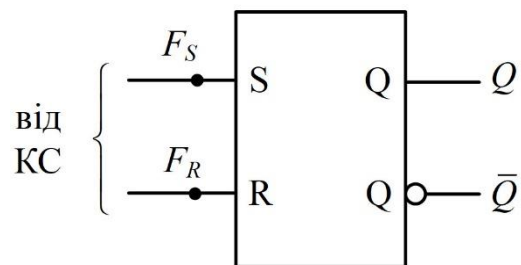


Рис. 8.9. Знаходження функцій збудження RS-тригера

У повній таблиці переходів аналізуємо порядково переходи $Q(t_i) \rightarrow Q(t_{i+1})$.

Перехід “0” → “0” зустрічається двічі – у рядку 1 та рядку 3. Для рядка 1 $S(t_i) = 0, R(t_i) = 0$, а для рядка 3 – $S(t_i) = 0, R(t_i) = 1$. Тому спільним для рядків 1 та 3 є $S(t_i) = 0, R(t_i) = \times$ (\times – довільне значення: 0 або 1).

Отже, для переходу “0” → “0” значення функцій збудження RS-тригера дорівнюють: $F_S = 0, F_R = \times$. Ці значення заносимо в таблицю 3.

Перехід “0” → “1” у стовпцях $Q(t_i), Q(t_{i+1})$ повної таблиці переходів зустрічається лише один раз – у рядку 5, для якого $S(t_i) = 1, R(t_i) = 0$. Тому

для переходу “0” → “1” значення функцій збудження RS-тригера дорівнюють $F_S = 1, F_R = 0$, які заносимо у таблицю 3.

Перехід “1” → “0” у стовпцях $Q(t_i), Q(t_{i+1})$ повної таблиці переходів зустрічається лише один раз – у рядку 4, для якого $S(t_i) = 0, R(t_i) = 1$. Тому для переходу “1” → “0” значення функцій збудження RS-тригера дорівнюють $F_S = 0, F_R = 1$, які заносимо у таблицю 3.

Перехід “1” → “1” у стовпцях $Q(t_i), Q(t_{i+1})$ повної таблиці переходів RS-тригера (таблиця 2) зустрічається двічі – у рядках 2 та 6. Для рядка 2 $S(t_i) = 0, R(t_i) = 0$, а для рядка 6 – $S(t_i) = 1, R(t_i) = 0$. Спільною для рядків 2 та 6 є комбінація $S(t_i) = \times, R(t_i) = 0$.

Тому для переходу “1” → “1” значення функцій збудження RS-тригера дорівнюють: $F_S = \times, F_R = 0$. Ці значення заносимо в таблицю 3.

Таким чином, таблиця 3 задає значення функцій F_S, F_R збудження RS-тригера при всіх можливих переходах тригера з одного стану в інший.

Функції F_S, F_R формуються комбінаційною схемою (КС) регістра.

За аналогією знайдемо функції збудження JK-тригера (рис. 8.10), D-тригера (рис. 8.11) та T-тригера (рис. 8.12).

Таблиця переходів
JK-тригера

$J(t_i)$	$K(t_i)$	$Q(t_{i+1})$
0	0	$Q(t_i)$
0	1	0
1	0	1
1	1	$\bar{Q}(t_i)$

Повна таблиця переходів
JK-тригера

$S(t_i)$	$R(t_i)$	$Q(t_i)$	$Q(t_{i+1})$
0	0	0	→ 0
0	0	1	→ 1
0	1	0	→ 0
0	1	1	---→ 0
1	0	0	~→ 1
1	0	1	→ 1
1	1	0	~→ 1
1	1	1	---→ 0

Таблиця функцій збудження
JK-тригера

$Q(t_i)$	$Q(t_{i+1})$	F_J	F_K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

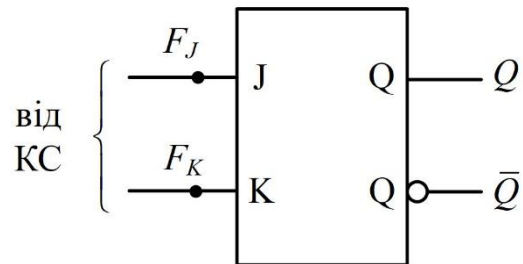


Рис. 8.10. Знаходження функцій збудження JK-тригера

Таблиця переходів
D-тригера

$D(t_i)$	$Q(t_{i+1})$
0	0
1	1

Повна таблиця
переходів D-тригера

$D(t_i)$	$Q(t_i)$	$Q(t_{i+1})$
0	0	→ 0
0	1	---→ 0
1	0	~→ 1
1	1	→ 1

Таблиця функції
збудження D-тригера

$Q(t_i)$	$Q(t_{i+1})$	F_D
0	0	0
0	1	1
1	0	0
1	1	1

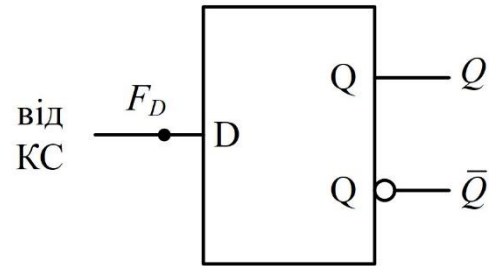


Рис. 8.11. Знаходження функцій збудження D-тригера

Таблиця переходів
T-тригера

$T(t_i)$	$Q(t_{i+1})$
0	$Q(t_i)$
1	$\bar{Q}(t_i)$

Повна таблиця
переходів D-тригера

$T(t_i)$	$Q(t_i)$	$Q(t_{i+1})$
0	0	→ 0
0	1	→ 1
1	0	~→ 1
1	1	---→ 0

Таблиця функції
збудження T-тригера

$Q(t_i)$	$Q(t_{i+1})$	F_T
0	0	0
0	1	1
1	0	1
1	1	0

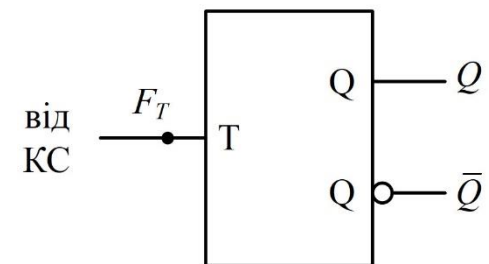


Рис. 8.12. Знаходження функцій збудження T-тригера

Для зручності використання під час синтезу регістрів функції збудження використовуваних тригерів доцільно звести в окремі таблиці (табл. 8.1 – 8.4).

Оскільки всі розряди регістра мають однакову будову, то синтез регістра зводиться до синтезу одного розряду регістра.

Розглянемо методику синтезу синхронного регістра при виконанні на ньому однієї мікрооперації.

1. Беруть таблицю функцій збудження заданого типу тригера.
2. Складають повну таблицю переходів j -го розряду регістра при виконанні заданої мікрооперації.
3. Мінімізують функції збудження тригера j -го розряду регістра (j -го тригера).
4. Будують схему j -го розряду регістра.
5. Будують повну схему регістра.
6. Наводять умовне графічне позначення синтезованого регістра.

Таблиця 8.1
Таблиця функцій збудження
RS-тригера

$Q(t_i)$	$Q(t_{i+1})$	F_S	F_R
0	0	0	×
0	1	1	0
1	0	0	1
1	1	×	0

Таблиця 8.2
Таблиця функцій збудження
JK-тригера

$Q(t_i)$	$Q(t_{i+1})$	F_J	F_K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

Таблиця 8.3
Таблиця функції
збудження D-тригера

$Q(t_i)$	$Q(t_{i+1})$	F_D
0	0	0
0	1	1
1	0	0
1	1	1

Таблиця 8.4
Таблиця функції
збудження T-тригера

$Q(t_i)$	$Q(t_{i+1})$	F_T
0	0	0
0	1	1
1	0	1
1	1	0

Запропоновану методику синтезу регістра розглянемо на прикладі розв'язування задачі.

Задача 8.1. Синтезувати 4-розрядний регістр на синхронних D-тригерах для виконання на ньому мікрооперації XOR(y_6), використовуючи для синтезу комбінаційної схеми логічні елементи 2І-НЕ.

Розв'язування.

1. Беремо таблицю функцій збудження D-тригера (див. табл. 8.3).
2. Складаємо повну таблицю переходів j -го розряду регістра ($j = 1, 2, 3, 4$) для виконання на ньому мікрооперації XOR(y_6) (табл. 8.5), яку описують рівнянням

$$Q_j(t_{i+1}) = Q_j(t_i) \oplus x_j(t_i).$$

Таблиця 8.5
Повна таблиця переходів j -го
розряду регістра при виконанні
мікрооперації XOR(y_6)

$x(t_i)$	$Q(t_i)$	$Q(t_{i+1})$	F_{D_j}
0	0	→0	0
0	1	→1	1
1	0	→1	1
1	1	→0	0

Для заповнення стовпця F_{D_j} користуємось таблицею функцій збудження D-тригера.

3. Мінімізуємо функцію збудження F_{D_j} та подаємо її в формі І-НЕ/І-НЕ:

$$\begin{array}{c}
 \begin{array}{c} \underline{Q(t_i)} \\ \swarrow F_{D_j} \end{array} \\
 x_j(t_i) \left| \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 1 & 0 \\ \hline \end{array} \right.
 \end{array}
 \quad
 F_{D_j} = x_j \bar{Q}_j \vee \bar{x}_j Q_j = \overline{x_j \bar{Q}_j \cdot \bar{x}_j Q_j}$$

4. Будуємо схему j -го розряду регістра (рис. 8.13). Сигнал мікрооперації XOR(y_6) подаємо на синхровхід С тригера.

5. Будуємо повну схему регістра (рис. 8.14). Розряди регістра ідентичні.

6. Умове графічне позначення 4-розрядного синхронного регістра, на якому виконують мікрооперацію XOR(y_6) показана на рис. 8.15. ■

Розглянемо методику синтезу синхронного регістра на якому виконують декілька мікрооперацій.

1. Беруть таблицю функцій збудження заданого типу тригера.
2. Складають повну таблицю переходів j -го розряду регістра для кожної мікрооперації (до складу таблиці включають сигнал y_k відповідної мікрооперації) та мінімізують функції збудження тригера j -го розряду регістра (j -го тригера) для кожної мікрооперації.
3. Формують узагальнені функції збудження тригера j -го розряду регістра (j -го тригера) як диз'юнкцію однойменних функцій збудження, що відповідають окремим мікроопераціям.
4. Будують схему j -го розряду регістра.
5. Будують повну схему регістра.
6. Наводять умове графічне позначення синтезованого регістра.

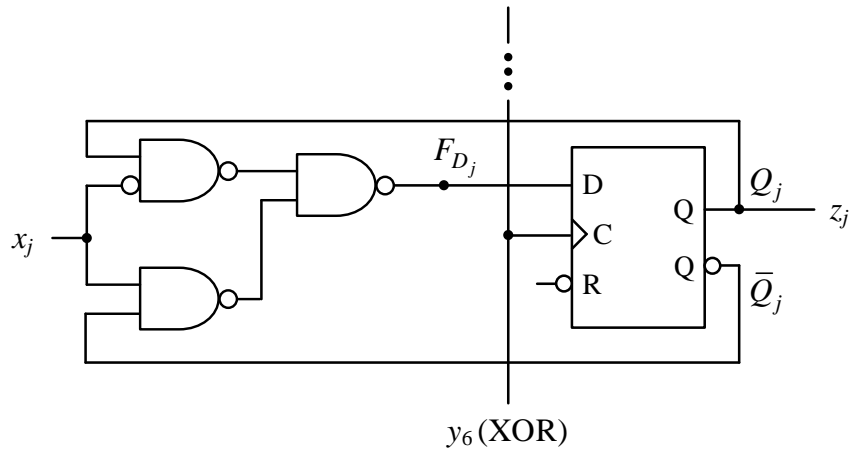


Рис. 8.13. Схема j -го розряду синхронного регістра, на якому виконують мікрооперацію XOR (D-тригер – на елементах І-НЕ на основі трьох бістабільних схем)

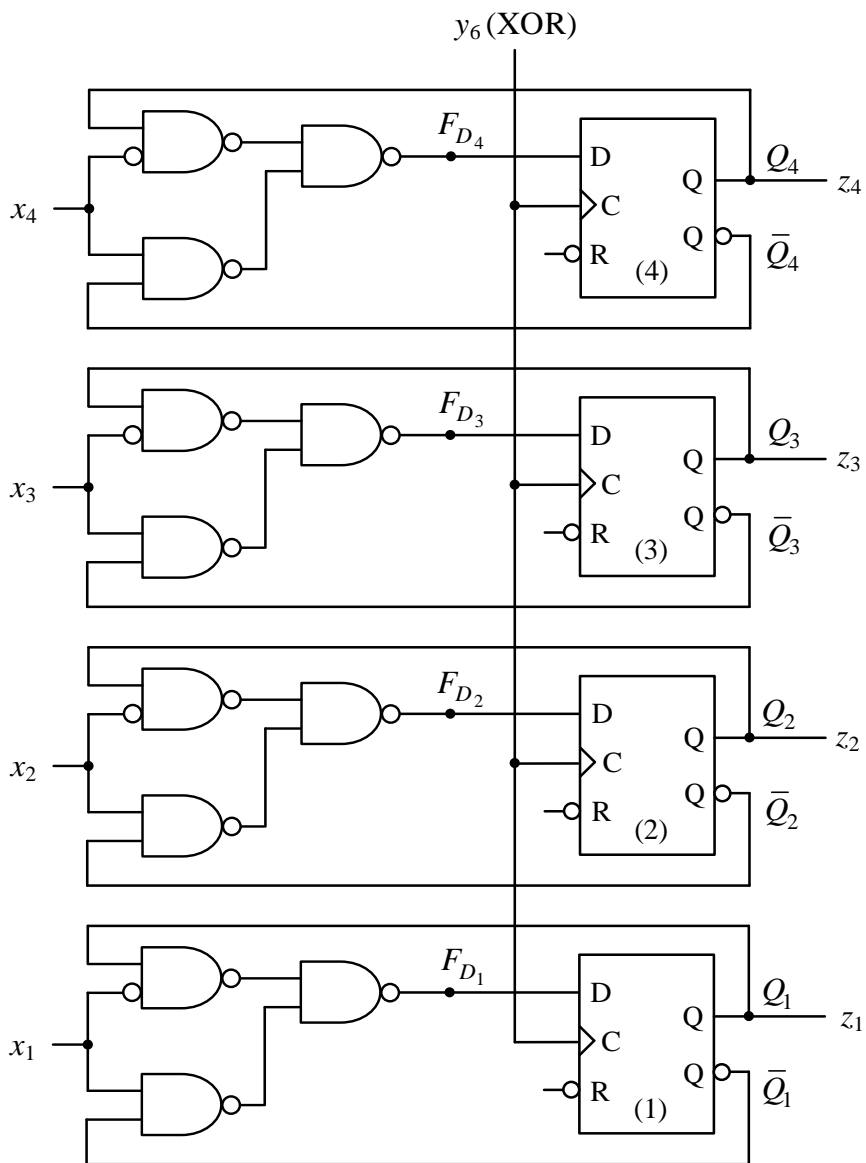


Рис. 8.14. Повна схема 4-розрядного синхронного регістра на D-тригерах, на якому виконують мікрооперацію XOR (y_6)

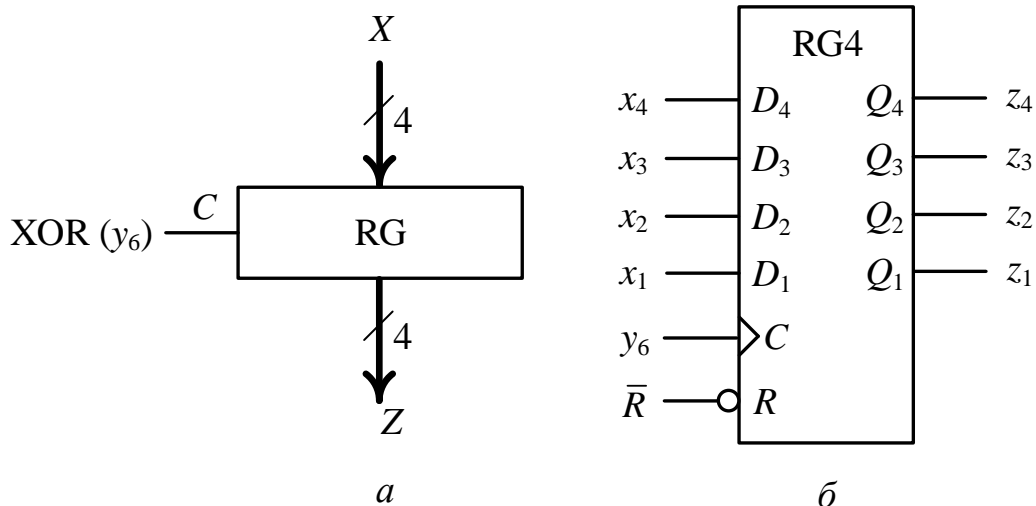


Рис. 8.15. Умовне графічне позначення 4-розрядного синхронного регістра, на якому виконують мікрооперацію XOR(y_6): *a* – на функціональних кресленнях; *б* – на принципових кресленнях

Запропоновану методику синтезу регістра розглянемо на прикладі розв’язування задачі.

Задача 8.2. На синхронних RS-тригерах синтезувати 4-розрядний регістр для виконання на ньому трьох мікрооперацій: WRITE(y_3) – запис (приймання) слова в регістр, OR(y_5) – порозрядна диз’юнкція двох слів, READ(y_{12}) – видача слова прямим кодом.

Розв’язування.

1. Беремо таблицю функцій збудження RS-тригера (табл. 8.1).
2. Складаємо повну таблицю переходів j -го розряду регістра окремо для кожної з мікрооперацій WRITE(y_3) та OR(y_5). Для мікрооперації READ(y_{12}) таблицю переходів тригера складати не потрібно, оскільки синтез комбінаційної схеми для мікрооперації видачі слова не виконують. Для цього скористаємось готовою схемою видачі (див. рис. 8.6).

2.1. Користуючись таблицею функцій збудження RS-тригера складаємо повну таблицю переходів j -го розряду регістра при виконанні мікрооперації WRITE(y_3) (табл. 8.6).

При $y_3 = 0$ мікрооперація не виконується, тому в стовпці $Q_j(t_{i+1})$ для кожного рядка повторюємо значення $Q_j(t_i)$.

Таблиця 8.6
Повна таблиця переходів j -го розряду регістра при виконанні мікрооперації WRITE

y_3	$x_j(t_i)$	$Q_j(t_i)$	$Q_j(t_{i+1})$	$F_{S_j}(y_3)$	$F_{R_j}(y_3)$
0	0	0	0	0	×
0	0	1	1	×	0
0	1	0	0	0	×
0	1	1	1	×	0
1	0	0	0	0	×
1	0	1	0	0	1
1	1	0	1	1	0
1	1	1	1	×	0

При $y_3 = 1$ стовпець $Q_j(t_{i+1})$ заповнюємо відповідно до алгебраїчного виразу мікрооперації $Q_j(t_{i+1}) = x_j(t_i)$.

Мінімізуємо функції збудження $F_{S_j}(y_3)$ та $F_{R_j}(y_3)$:

y_3	$x_j(t_i)$				
1	×	0	0	↙ $F_{S_j}(y_3)$	
0	×	×	0		
					$Q_j(t_i)$

$$F_{S_j}(y_3) = y_3 x_j$$

y_3	$x_j(t_i)$				
0	0	1	×	↙ $F_{R_j}(y_3)$	
×	0	0	×		
					$Q_j(t_i)$

$$F_{R_j}(y_3) = y_3 \bar{x}_j$$

2.2. Користуючись таблицею функцій збудження RS-тригера складаємо повну таблицю переходів j -го розряду регістра при виконанні мікрооперації OR(y_5) (табл. 8.7), яку описують рівнянням

$$Q_j(t_{i+1}) = Q_j(t_i) \vee x_j(t_i).$$

Мінімізуємо функції збудження $F_{S_j}(y_5)$ та $F_{R_j}(y_5)$:

y_5	$x_j(t_i)$				
1	×	×	0	↙ $F_{S_j}(y_5)$	
0	×	×	0		
					$Q_j(t_i)$

$$F_{S_j}(y_5) = y_5 x_j$$

y_5	$x_j(t_i)$				
0	0	0	×	↙ $F_{R_j}(y_5)$	
×	0	0	×		
					$Q_j(t_i)$

$$F_{R_j}(y_5) = 0$$

Таблиця 8.7

Повна таблиця переходів j -го розряду регістра при виконанні мікрооперації OR(y_5)

y_3	$x_j(t_i)$	$Q_j(t_i)$	$Q_j(t_{i+1})$	$F_{S_j}(y_5)$	$F_{R_j}(y_5)$
0	0	0	0	0	×
0	0	1	1	×	0
0	1	0	0	0	×
0	1	1	1	×	0
1	0	0	0	0	0
1	0	1	1	×	×
1	1	0	1	1	1
1	1	1	1	×	×

3. Сформуємо узагальнені функції збудження тригера j -го розряду регістра:

$$F_{S_j} = F_{S_j}(y_3) \vee F_{S_j}(y_5) = y_3 x_j \vee y_5 x_j = x_j (y_3 \vee y_5),$$

$$F_{R_j} = F_{R_j}(y_3) \vee F_{S_j}(y_5) = y_3 \bar{x}_j \vee 0 = y_3 \bar{x}_j.$$

4. Будуємо схему j -го розряду регістра (рис. 8.16).

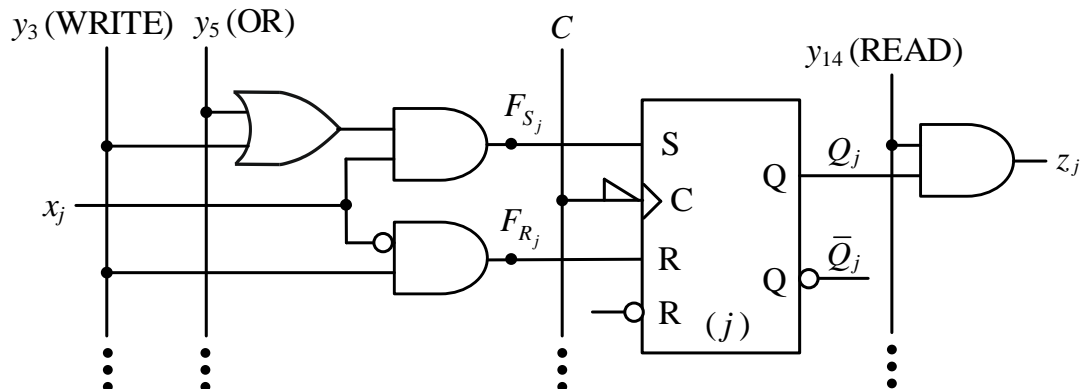


Рис. 8.16. Схема j -го розряду синхронного регістра, на якому виконують мікрооперації WRITE(y_3), OR(y_5), READ(y_{14}) (RS-тригер – на елементах I-HE за MS-схемою)

5. Будуємо повну схему 4-розрядного синхронного регістра (рис. 8.17).

6. Умовне графічне позначення 4-розрядного регістра, на якому виконують три мікрооперації – WRITE, OR, READ, показано на рис. 8.18. ■

Розглянута методика синтезу синхронних регістрів може застосовуватись при виконанні на регістрі довільних мікрооперацій, в тому числі й мікрооперацій зсуву слова вліво та вправо (рис. 8.19, 8.20).

У вітчизняній літературі часто використовують альтернативні умовні графічні позначення регістрів (рис. 8.21).

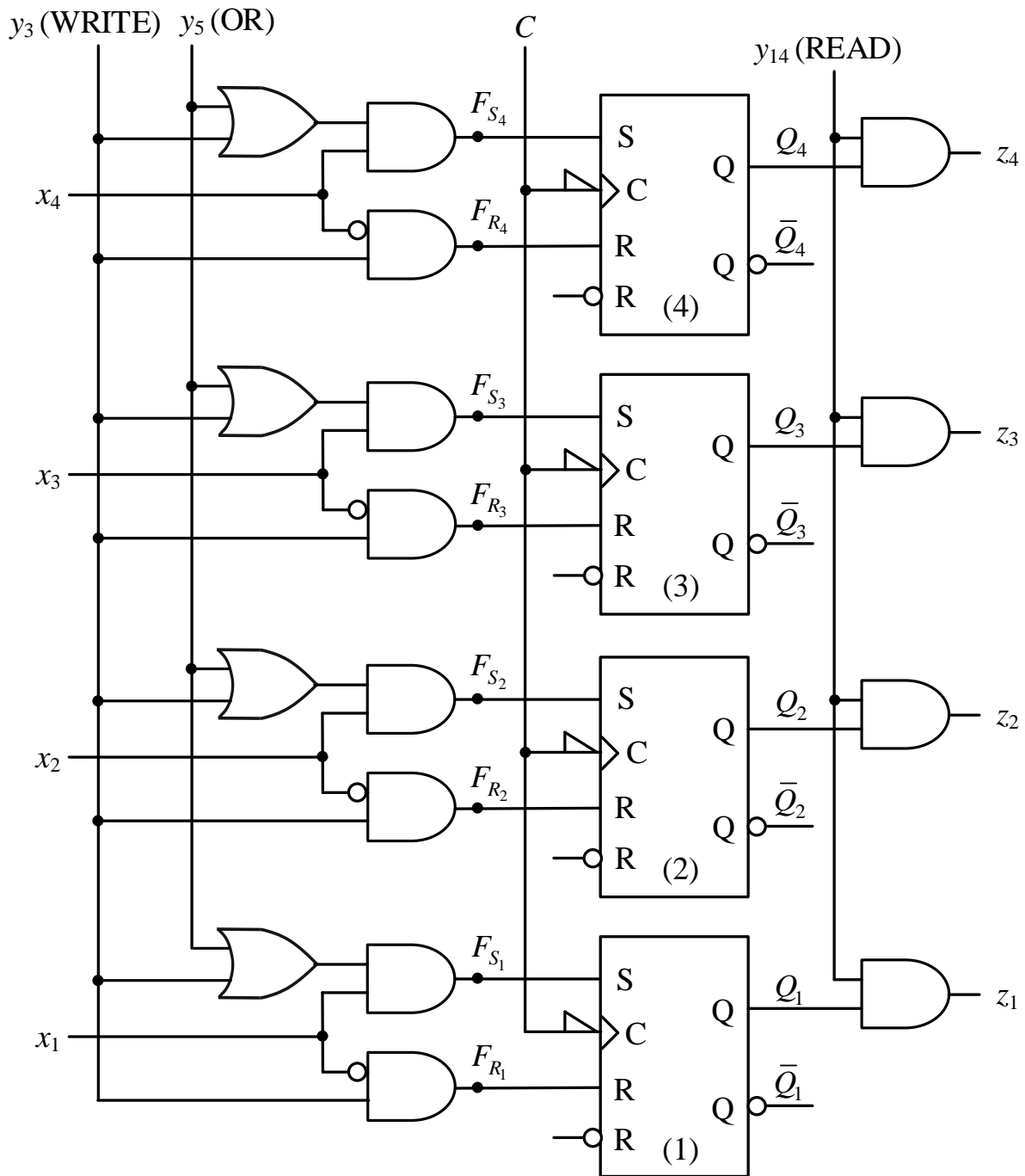


Рис. 8.17. Повна схема 4-розрядного синхронного регістра на RS-тригерах, на якому виконують мікрооперації WRITE(y_3), OR(y_5), READ(y_{14})

Запитання та завдання

1. До якої задачі зводиться синтез регістра на тригерах заданого типу?
2. Дайте визначення функцій збудження тригера заданого типу.
3. Знайдіть функції збудження: а) E-тригера; б) S-тригера.
4. Охарактеризуйте методику синтезу синхронного регістра при виконанні на ньому однієї мікрооперації.

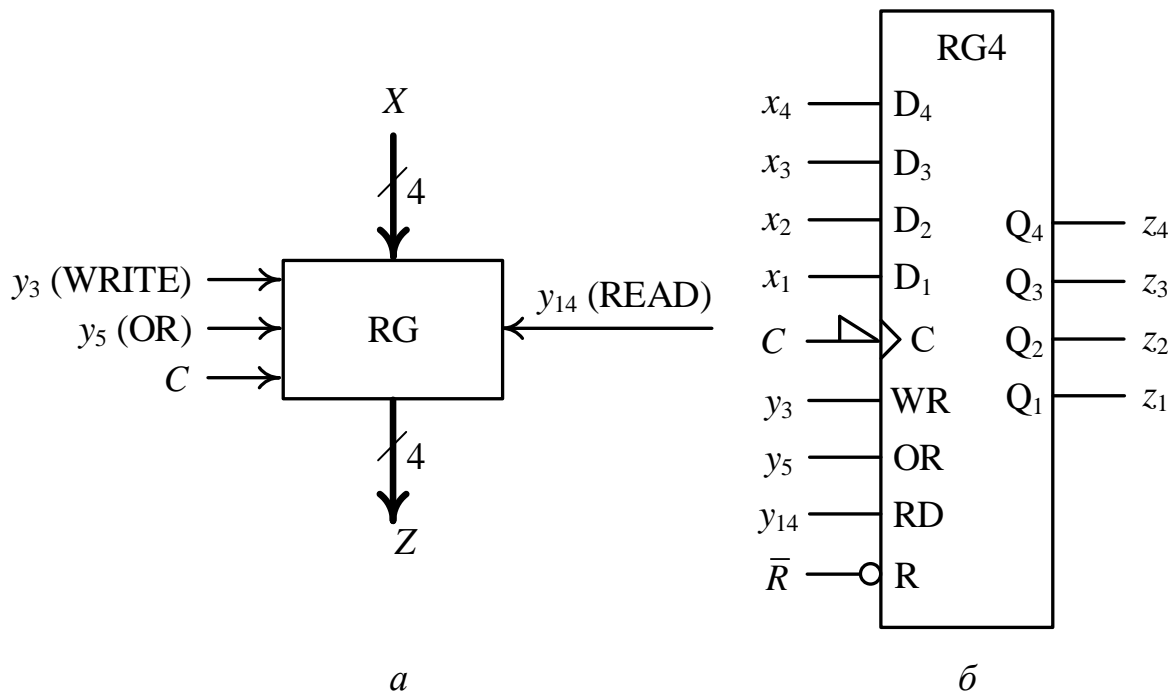


Рис. 8.18. Умовне графічне позначення 4-розрядного синхронного регістра, на якому виконують мікрооперації WRITE, OR, READ:
a – на функціональних кресленнях; *б* – на принципових кресленнях

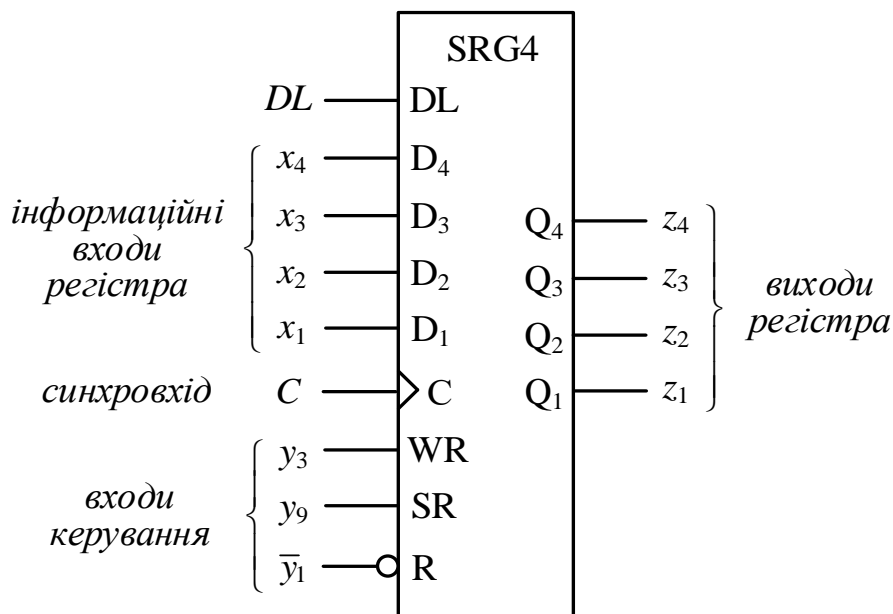


Рис. 8.19. Умовне графічне позначення 4-розрядного регістра зсуву вправо

5. Якою є методика синтезу синхронного регістра при виконанні на ньому кількох мікрооперацій?
6. Наведіть умовне графічне позначення 4-розрядного реверсивного регістра, на якому виконують мікрооперації WRITE, SLL, SRL.

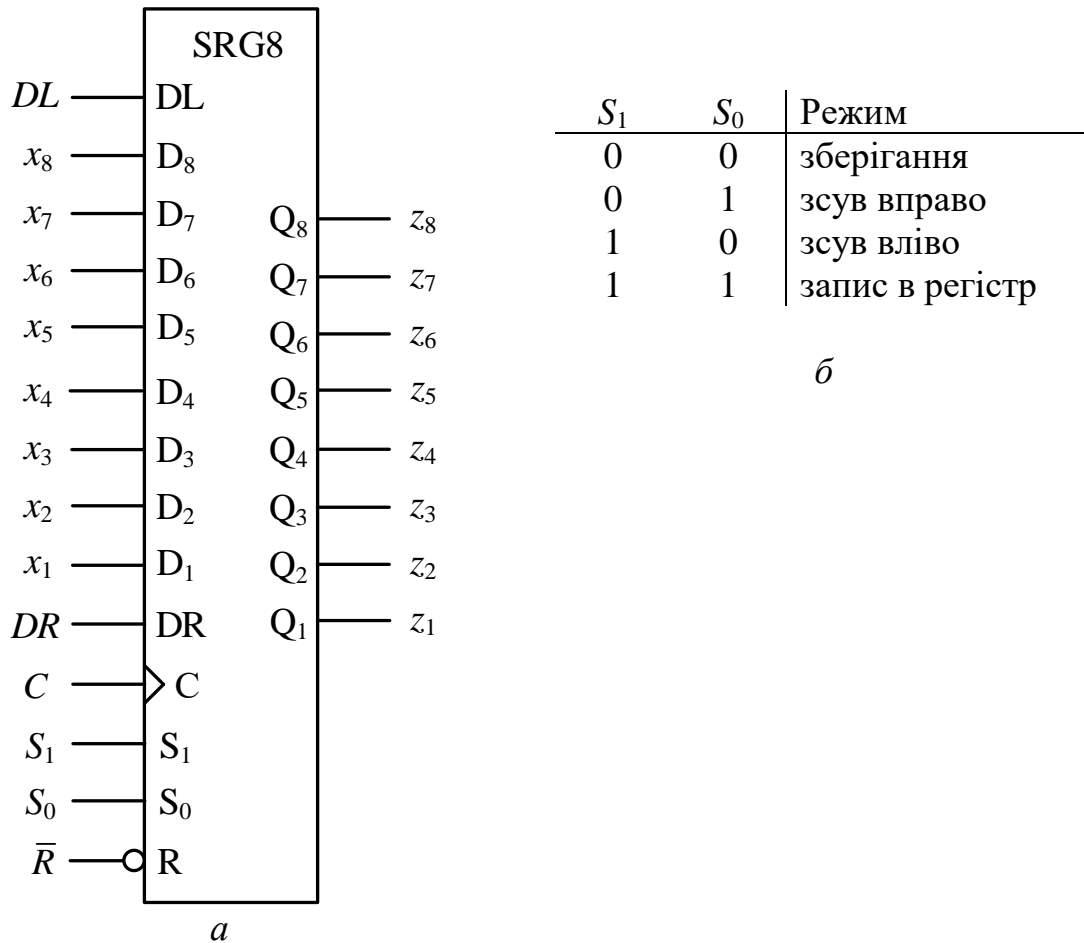


Рис. 8.20. Синхронний 8-розрядний реверсивний регістр: *а* – умовне графічне позначення регістра; *б* – режими роботи регістра

Задачі для самостійного розв'язування

1. Синтезувати 4-розрядний регістр на синхронних JK-тригерах для виконання на ньому мікрооперації $COM(y_7)$.
2. Синтезувати 4-розрядний регістр на синхронних T-тригерах для виконання на ньому мікрооперації $SLL(y_8)$.
3. Синтезувати 4-розрядний регістр на синхронних JK-тригерах для виконання на ньому мікрооперацій: $WRITE(y_3)$, $XOR(y_6)$, $RDCOM(y_{15})$.
4. На синхронних T-тригерах синтезувати 4-розрядний регістр для виконання на ньому мікрооперацій: $WRITE(y_3)$, $AND(y_4)$, $READ(y_{14})$.

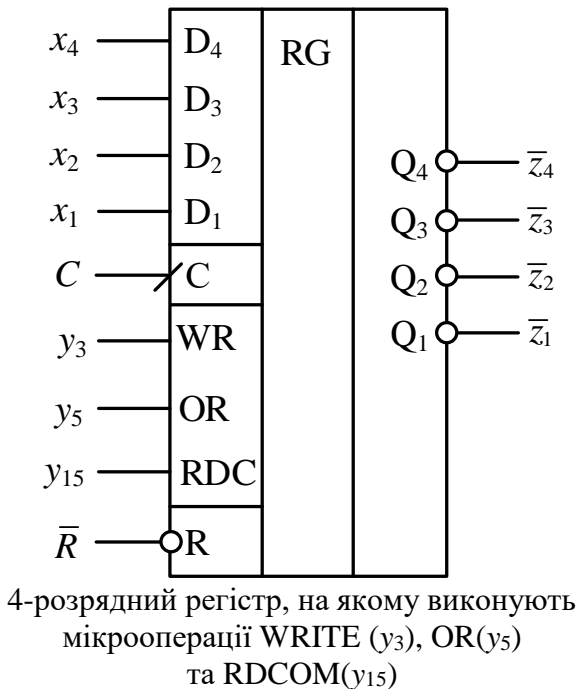
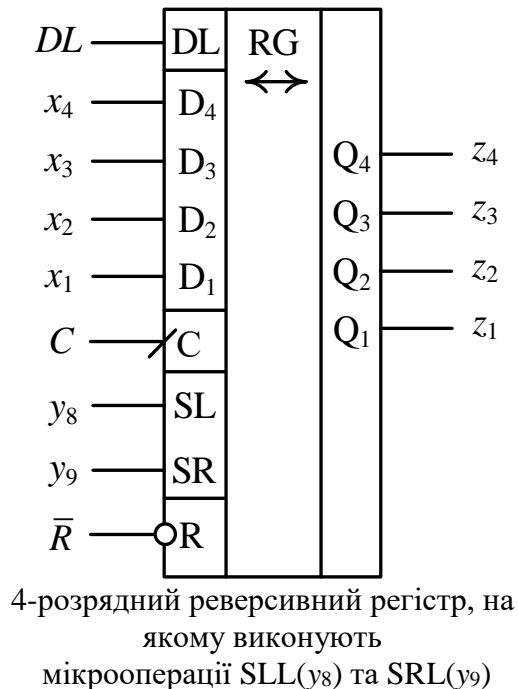
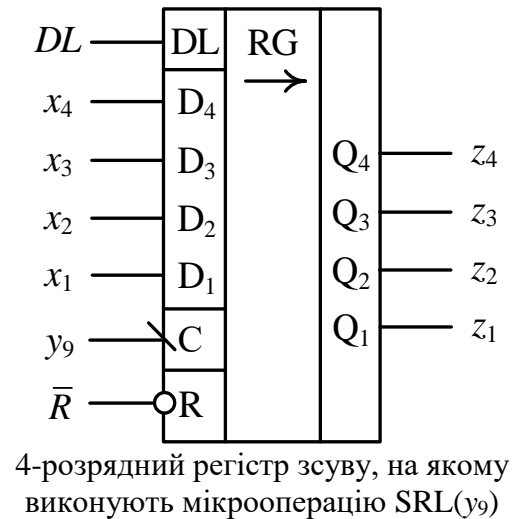


Рис. 8.21. Приклади альтернативних умовних графічних позначень регістрів

8.3. Лічильники

Лічильником називають послідовнісну схему (регістр), призначену для виконання мікрооперацій лічби.

Лічильники будують на основі тригерів.

Кількість дозволених станів лічильника називають його *періодом* або *модулем*.

Лічильник, до складу якого входять n тригерів, називають n -розрядним лічильником.

Залежно від основи системи числення, у якій виконують мікрооперацію лічби, розрізняють двійкові, двійково-десяткові, двійково-п'ятіркові лічильники тощо.

Лічильники бувають двох видів: з природним порядком лічби; з довільним (неприродним) порядком лічби.

Лічильники з природним порядком лічби

Лічильники з природним порядком лічби забезпечують виконання двох мікрооперацій лічби: інкременту (increment, INC) – збільшення стану лічильника на одиницю (+1), та декременту (decrement, DEC) – зменшення стану на одиницю (-1).

Якщо до складу лічильника з природним порядком лічби входять n тригерів, то період (модуль) лічильника дорівнює 2^n .

При проектуванні лічильників з природним порядком лічби використовують Т-тригери та JK-тригери, які називають *лічильними тригерами* (рис. 8.22). Якщо в JK-тригері вхід J електрично з'єднати з входом K (див. рис. 8.22в), то він функціонуватиме як Т-тригер.

Таблиця переходів
Т-тригера

$T(t_i)$	$Q(t_{i+1})$
0	$Q(t_i)$
1	$\bar{Q}(t_i)$

Таблиця переходів
JK-тригера

$J(t_i)$	$K(t_i)$	$Q(t_{i+1})$
0	0	$Q(t_i)$
0	1	0
1	0	1
1	1	$\bar{Q}(t_i)$

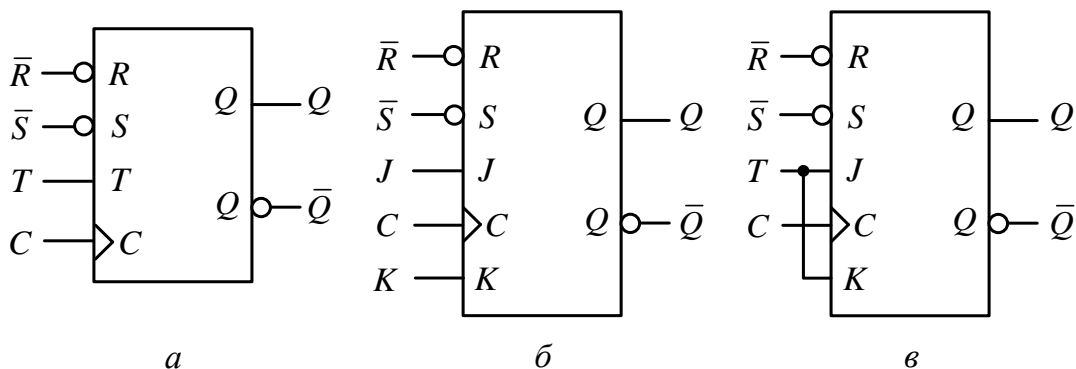


Рис. 8.22. Лічильні тригери: а – Т-тригер; б – JK-тригер; в – Т-тригер на основі JK-тригера

Тригери, УГП яких наведені на рис. 8.22, побудовані на елементах І-НЕ на основі трьох бістабільних схем.

Структурно лічильник складається з тригерів та комбінаційної схеми (КС) (рис. 8.23), де CI (*carry input*) – лічильний сигнал, $X = (x_n, \dots, x_1)$ – початковий стан лічильника, $Y = (y_m, \dots, y_1)$ – сигнали мікрооперацій, $Z = (z_n, z_{n-1}, \dots, z_1)$ – поточний стан лічильника, Φ_j – функція міжрозрядного переносу в j -й розряд лічильника.

Крім двох основних мікрооперацій – інкременту (INC) та декременту (DEC), зазвичай в лічильнику також виконуються мікрооперації: WRITE – запис початкового (довільного) стану (слова) X в лічильник, SET – встановлення лічильника в початковий стан (11...1), RESET – встановлення лічильника в початковий стан (00...0).

Для виконання мікрооперації SET на входи \bar{S} асинхронного встановлення тригерів у початковий стан “1” подають сигнал 0 ($\bar{S} = 0, \bar{R} = 1$), а для виконання мікрооперації RESET на входи \bar{R} асинхронного встановлення тригерів у початковий стан “0” подають сигнал 0 ($\bar{R} = 0, \bar{S} = 1$).

Функції міжрозрядних переносів $\Phi_1, \Phi_2, \dots, \Phi_n$ виконують роль функцій збудження тригерів.

На кресленнях лічильник позначають CTR (counter) (рис. 8.24).

У табл. 8.8 показано стани інкрементного та декрементного лічильників при підрахунку 20-ти лічильних сигналів.

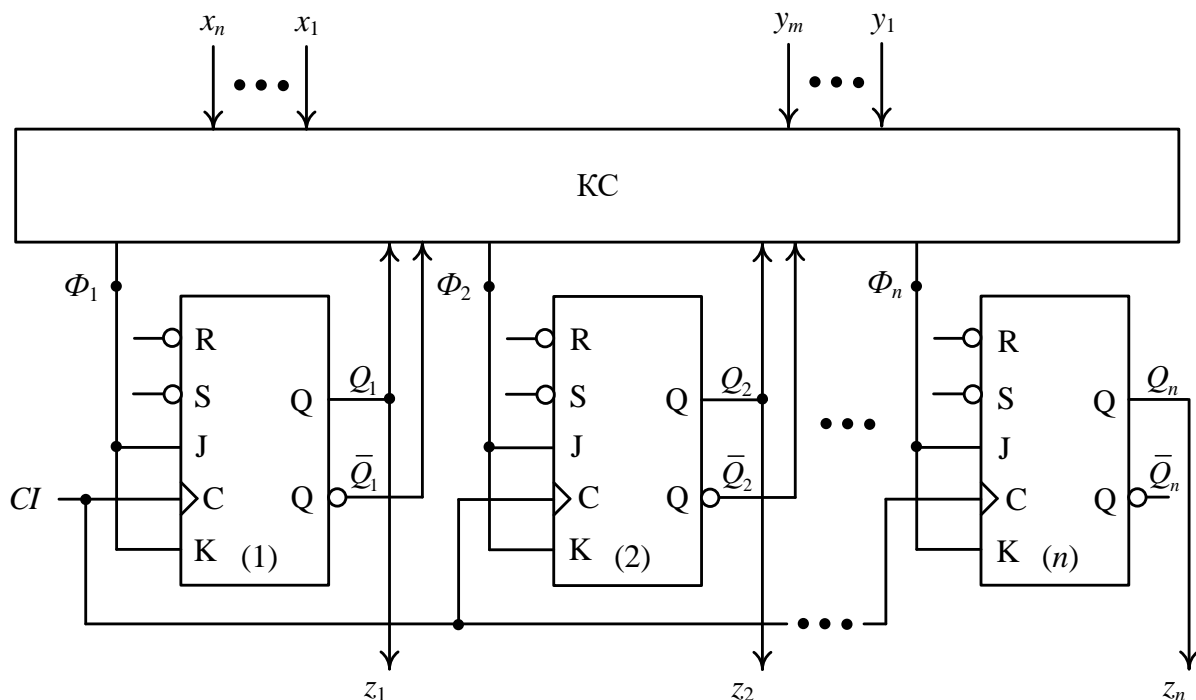


Рис. 8.23. Узагальнена структура лічильника

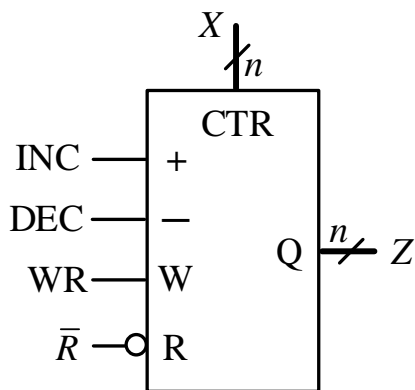


Рис. 8.24. Умовне графічне позначення лічильника

Період (модуль) лічильника дорівнює 16.

Лічильники з природним порядком лічби поділяють на інкрементні, декрементні та реверсивні.

У моменти надходження на вхід CI чергового лічильного сигналу стан інкрементного лічильника змінюється на $+1$, тобто $(CTR) := (CTR) + 1$, а декрементного – на -1 , тобто $(CTR) := (CTR) - 1$. Реверсивний лічильник може виконувати як мікрооперацію інкременту, так і мікрооперацію декременту (рис. 8.24).

Таблиця 8.8

Таблиця станів 4-розрядного лічильника з природним порядком лічби з модулем 16

Кількість лічильних сигналів	Стан лічильника	
	Інкрементного	Декрементного
0	0000	0000
1	0001	1111
2	0010	1110
3	0011	1101
4	0100	1100
5	0101	1011
6	0110	1010
7	0111	1001
8	1000	1000
9	1001	0111
10	1010	0110
11	1011	0101
12	1100	0100
13	1101	0011
14	1110	0010
15	1111	0001
16	0000	0000
17	0001	1111
18	0010	1110
19	0011	1101
20	0100	1100

Перемикання тригера молодшого розряду лічильника здійснюється з надходженням кожного лічильного сигналу CI , а решти тригерів – лише в тому випадку, коли всі тригери молодших розрядів відносно поточного

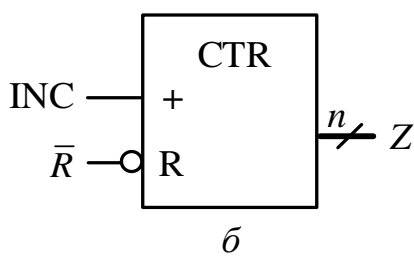
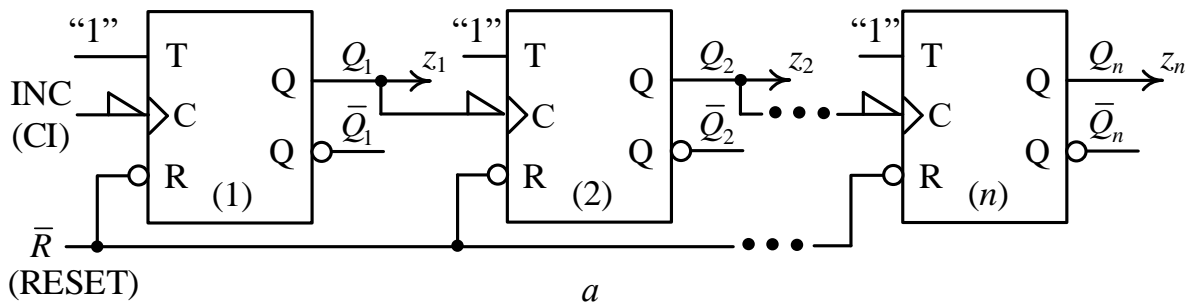
тригера встановлені в стан “1” (інкрементний лічильник) або в стан “0” (декрементний лічильник).

За способом організації переносу (позички) між розрядами лічильники з природним порядком лічби поділяють на: лічильники з послідовним переносом; лічильники з паралельним переносом.

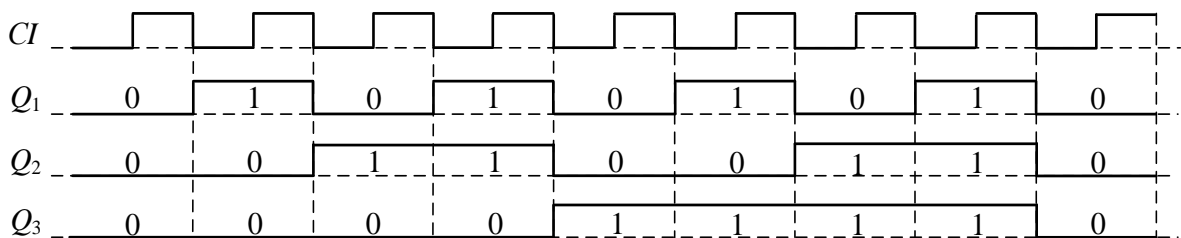
У лічильниках з послідовним переносом на тактові входи С синхронних тригерів надходять сигнали з виходів сусідніх розрядів. Перенос (позичка) у сусідній (старший) розряд формується лише після перемикаання тригера в попередньому (молодшому) розряді. Тригери в таких лічильниках спрацьовують неодноразово (асинхронно), оскільки перемикаання одних тригерів починається лише після зміни стану інших тригерів.

Отже, лічильники з послідовним переносом є асинхронними.

В інкрементному лічильнику з послідовним переносом (рис. 8.25) перемикаання 1-го розряду здійснюється з надходженням кожного лічильного сигналу, тобто коли $CI = 1$, а решти тригерів, коли всі тригери молодших розрядів встановлені в “1”. Перемикаання тригерів відбувається за заднім фронтом синхросигналу (тригери побудовані на елементах І-НЕ за MS-схемою).



$Z = (z_n z_{n-1} \dots z_1)$ – стан лічильника
 z_n – старший розряд лічильника
 z_1 – молодший розряд лічильника



в

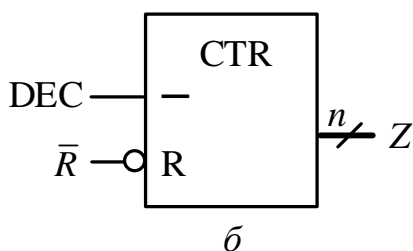
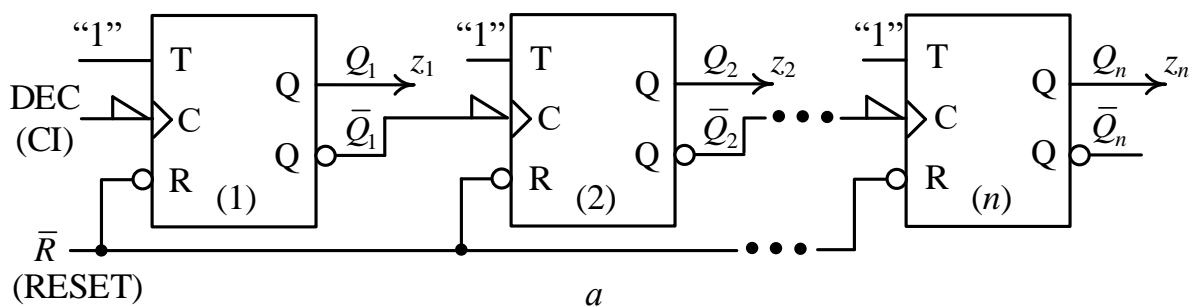
Рис. 8.25. Інкрементний лічильник з послідовним переносом:
 а – структура лічильника; б – умовне графічне позначення лічильника;
 в – часова діаграма станів лічильника при $n = 3$

В інкрементному лічильнику з послідовним переносом вихід Q_j тригера j -го розряду з'єднаний з входом С тригера $(j + 1)$ -го розряду, $j = 1, 2, \dots, n - 1$.

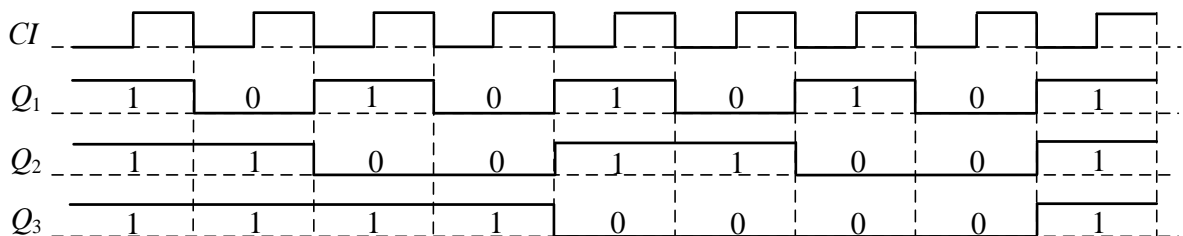
Перший розряд декрементного лічильника з послідовним переносом (рис. 8.26) щоразу перемикається, коли надходить черговий сигнал $CI = 1$, але перемикання решти тригерів відбувається, коли всі тригери молодших розрядів встановлені в "0".

У декрементному лічильнику (з послідовним переносом) вихід \bar{Q}_j тригера j -го розряду з'єднаний з входом С тригера $(j + 1)$ -го розряду, $j = 1, 2, \dots, n - 1$.

У лічильниках з природним порядком лічби з паралельним (одночасним) переносом усі тригери перемикаються одночасно під дією спільного синхросигналу, що надходить на входи С всіх тригерів. Аргументами функції міжрозрядного переносу в даний розряд є сигнали з виходів тригерів попередніх (молодших) розрядів. Переноси у всі розряди лічильника формуються одночасно.



$Z = (z_n z_{n-1} \dots z_1)$ – стан лічильника
 z_n – старший розряд лічильника
 z_1 – молодший розряд лічильника



в

Рис. 8.26. Декрементний лічильник з послідовним переносом:
 а – структура лічильника; б – умовне графічне позначення лічильника;
 в – часова діаграма станів лічильника при $n = 3$

Функції міжрозрядних переносів інкрементного лічильника з паралельним переносом мають вигляд:

$$\begin{cases} \Phi_j = Q_1 \cdot Q_2 \cdot \dots \cdot Q_{j-1}, j = 2, \dots, n, \\ \Phi_1 = 1. \end{cases}$$

Наприклад, для 4-розрядного інкрементного лічильника з паралельним переносом на рис. 8.27 функції міжрозрядних переносів $\Phi_1 - \Phi_4$ дорівнюють:

$$\begin{cases} \Phi_1 = 1, \\ \Phi_2 = Q_1, \\ \Phi_3 = Q_1 \cdot Q_2, \\ \Phi_4 = Q_1 \cdot Q_2 \cdot Q_3. \end{cases}$$

Сигнал мікрооперації INC подають на синхровходи C всіх тригерів лічильника, він виконує функцію лічильного сигналу CI.

У декрементному лічильнику з паралельним переносом на входи елементів I подають сигнали з інверсних виходів (\bar{Q}) тригерів.

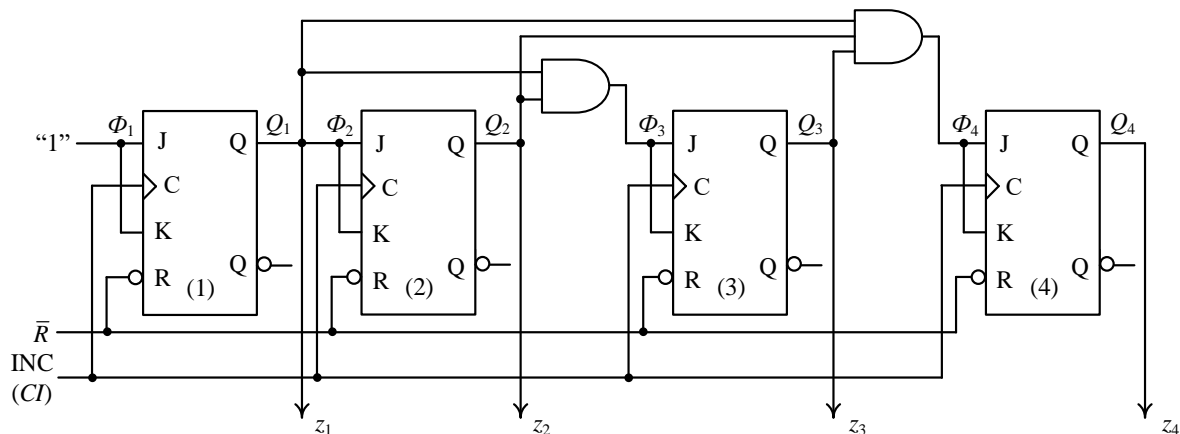


Рис. 8.27. Структура 4-розрядного інкрементного лічильника з паралельним переносом на JK-тригерах

Функції міжрозрядних переносів декрементного лічильника з паралельним переносом мають вигляд:

$$\begin{cases} \Phi_j = \bar{Q}_1 \cdot \bar{Q}_2 \cdot \dots \cdot \bar{Q}_{j-1}, j = 2, \dots, n, \\ \Phi_1 = 1. \end{cases}$$

Сигнал мікрооперації DEC подають на синхровходи всіх тригерів лічильника, він виконує функцію лічильного сигналу CI.

Наприклад, для 4-розрядного декрементного лічильника з паралельним переносом на рис. 8.28 функції міжрозрядних переносів $\Phi_1 - \Phi_4$ дорівнюють:

$$\begin{cases} \Phi_1 = 1, \\ \Phi_2 = \bar{Q}_1, \\ \Phi_3 = \bar{Q}_1 \cdot \bar{Q}_2, \\ \Phi_4 = \bar{Q}_1 \cdot \bar{Q}_2 \cdot \bar{Q}_3. \end{cases}$$

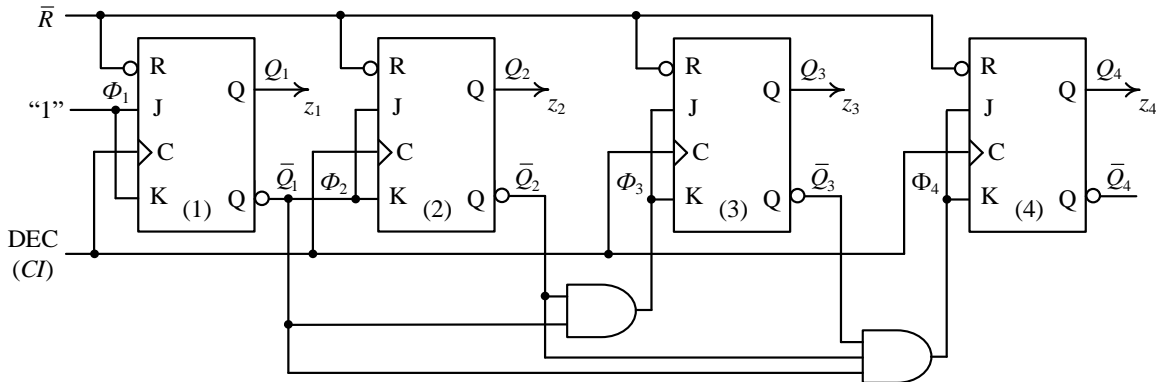


Рис. 8.28. Функціональна схема 4-розрядного декрементного лічильника з паралельним переносом на JK-тригерах

У реверсивному лічильнику з паралельним переносом функції міжрозрядних переносів мають вигляд:

$$\begin{cases} \Phi_j = Q_1 \cdot Q_2 \cdot \dots \cdot Q_{j-1} \cdot y_1 \vee \bar{Q}_1 \cdot \bar{Q}_2 \cdot \dots \cdot \bar{Q}_{j-1} \cdot y_2, j = 2, \dots, n, \\ \Phi_1 = 1. \end{cases}$$

При $y_1 = 1$ виконується мікрооперація інкременту (INC), при $y_2 = 1$ – декременту (DEC).

Наприклад, для 4-розрядного реверсивного лічильника з паралельним переносом функції міжрозрядних переносів дорівнюють:

$$\begin{cases} \Phi_1 = 1, \\ \Phi_2 = Q_1 \cdot y_1 \vee \bar{Q}_1 \cdot y_2, \\ \Phi_3 = Q_1 \cdot Q_2 \cdot y_1 \vee \bar{Q}_1 \cdot \bar{Q}_2 \cdot y_2, \\ \Phi_4 = Q_1 \cdot Q_2 \cdot Q_3 \cdot y_1 \vee \bar{Q}_1 \cdot \bar{Q}_2 \cdot \bar{Q}_3 \cdot y_2. \end{cases}$$

Структура 4-го розряду реверсивного лічильника з паралельним переносом подана на рис. 8.29.

Повна функціональна схема 3-розрядного реверсивного лічильника з паралельним переносом наведена на рис. 8.30а.

У реверсивному лічильнику на синхровходи всіх тригерів лічильника подають сигнал синхронізації С.

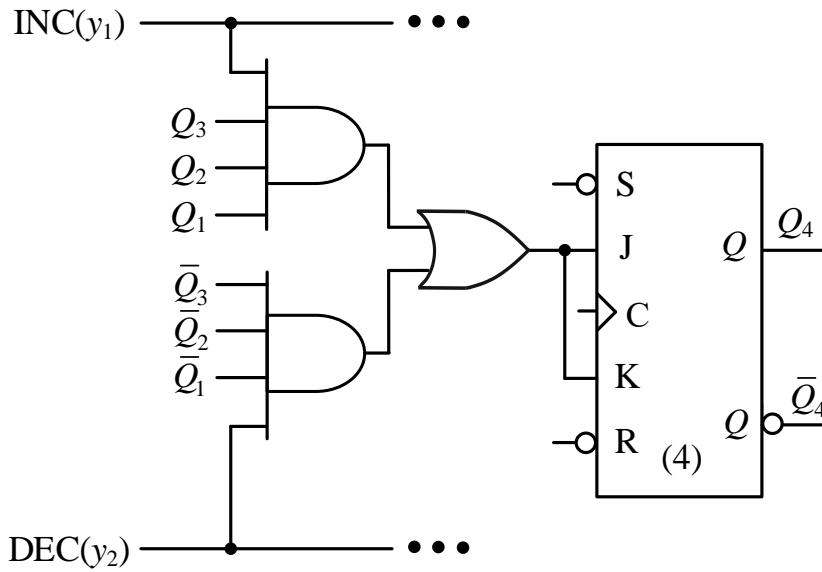
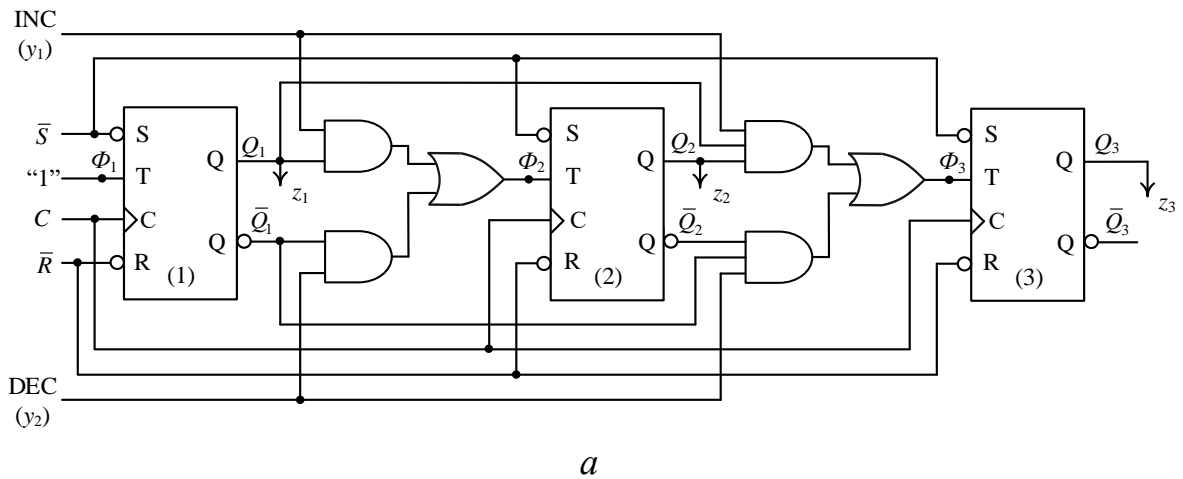
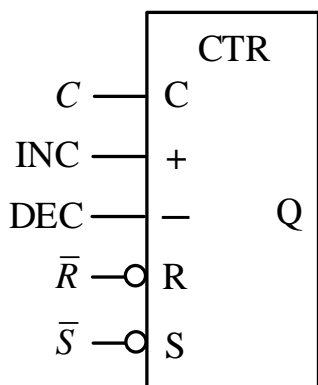


Рис. 8.29. Структура 4-го розряду реверсивного лічильника з паралельним переносом



a



б

$Z = (z_3 z_2 z_1)$
стан
лічильника

$$\begin{cases} \Phi_1 = 1 \\ \Phi_2 = Q_1 \cdot y_1 \vee \bar{Q}_1 \cdot y_2 \\ \Phi_3 = Q_1 \cdot Q_2 \cdot y_1 \vee \bar{Q}_1 \cdot \bar{Q}_2 \cdot y_2 \end{cases}$$

в

Рис. 8.30. Трирозрядний реверсивний лічильник з паралельним переносом:
a – функціональна схема лічильника; *б* – умовне графічне позначення;
в – функції міжрозрядних переносів лічильника

Таким чином, лічильним входом СІ в лічильнику з послідовним переносом є вхід синхросигналу С першого тригера лічильника (рис. 8.25а, 8.26а).

Лічильним входом СІ в лічильнику з паралельним переносом інкрементного (декрементного) типу є вхід сигналу мікрооперації INC (DEC) (рис. 8.27, 8.28).

Реверсивний лічильник має два лічильні входи – один з них є входом сигналу мікрооперації INC, а інший – входом сигналу мікрооперації DEC (рис. 8.30 а, б).

Регістри/лічильники

Узагальнена структура лічильника (рис. 8.23) має багато спільного з узагальненою структурою регістра (рис. 8.2).

Як і регістр, лічильник виконує мікрооперації:

- встановлення початкового стану “0...0” (RESET) або “1...1” (SET),
- запис (WRITE) слова X та його зберігання, $CTR := (X)$.

Лічильник, на відміну від регістра, додатково виконує дві спеціальні мікрооперації лічби:

- інкремент (INC), $CTR := (CTR) + 1$,
- декремент (DEC), $CTR := (CTR) - 1$.

Отже, лічильник можна означити як *регістр*, в якому додатково виконуються мікрооперації лічби (+1, -1); або як *регістр* зі спеціальними міжрозрядними зв'язками, що забезпечують мікрооперації лічби. Тому часто регістр (RG) і лічильник (CTR) виготовляють як суміщений функціональний вузол, який називають регістр/лічильник (RG/CTR).

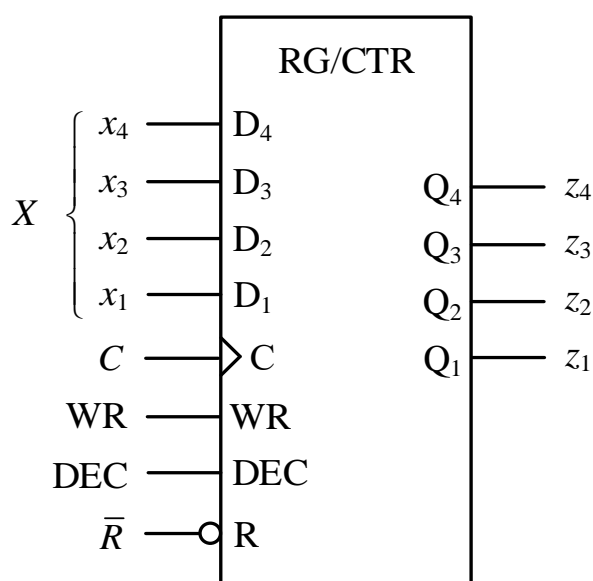


Рис. 8.31. Чотирирозрядний регістр/лічильник

В окремих тактах роботи цей функціональний вузол може виконувати мікрооперації притаманні регістру – встановлення початкового стану “0...0”, запис слова та його зберігання, а в інших тактах – працює як лічильник, тобто виконує мікрооперації INC або DEC.

Наприклад, 4-розрядний регістр/лічильник на рис. 8.31 здійснює запис 4-розрядного слова $X = (x_4 x_3 x_2 x_1)$ в регістр ($WR = 1 \Rightarrow RG := (x_4 x_3 x_2 x_1)$); встановлює регістр в нульовий

стан ($RG := (0 \dots 0)$), якщо подати сигнал $\bar{R} = 0$; виконує декремент стану регістра ($DEC = 1 \Rightarrow RG := (RG) - 1$).

Лічильники з природним порядком лічби використовують в комп'ютері під час апаратної реалізації операцій множення, ділення, добування квадратного кореня, піднесення до степеня, які виконуються як циклічні процеси (ітерації) з фіксованою кількістю повторень.

Перед початком операції, наприклад, множення двох n -розрядних слів, у лічильник записують початковий стан – кількість ітерацій, яку необхідно виконати. Після кожного проходження тіла циклу виконується декремент лічильника. Завершення операції визначають за нульовим вмістом лічильника.

Отже, лічильник є обов'язковим компонентом під час апаратної реалізації переважної більшості обчислювальних операцій.

Програмний лічильник

До складу процесора входить спеціальний регістр/лічильник, який називають *програмним лічильником* та позначають PC (program counter).

PC у кожному такті роботи процесора формує адресу наступної команди програми, яка зберігається в оперативному запам'ятовувальному пристрої (ОЗП) (рис. 8.32).

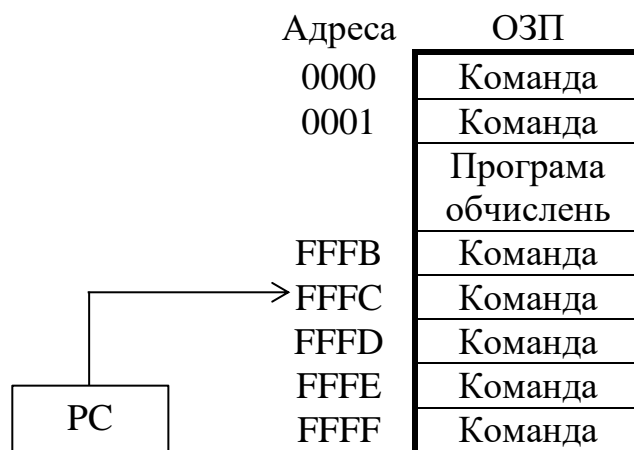


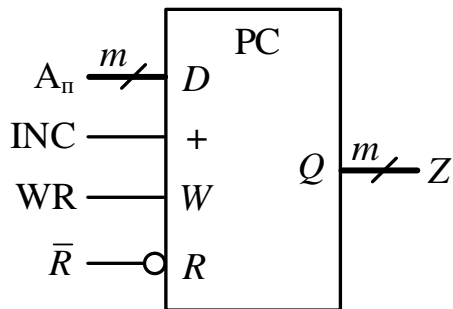
Рис. 8.32. Формування адреси наступної команди програми

Під час виконання поточної команди програми з адресою k програмний лічильник автоматично збільшує свій стан на $+1$, готуючись, таким чином, до вибірки з ОЗП наступної команди (з адресою на одиницю більшу за адресу поточної команди). Тобто, в програмному лічильнику заздалегідь встановлюється стан $k + 1$ ($PC := (PC) + 1$).

Якщо ж поточною командою (з адресою k) програми є команда переходу – умовного або безумовного, а це з'ясується в процесі

дешифрування коду операції команди, то новий стан $(k + 1)$ програмного лічильника скасовується, і в нього за допомогою мікрооперації WR (write) завантажується нове слово $A_{п}$ – адреса переходу ($PC := A_{п}$) (рис. 8.34).

Програмний лічильник не виконує мікрооперацію декременту, а лише інкременту та мікрооперацію WR запису слова в лічильник (рис. 8.33).



$A_{п}$ – початкова адреса програми (або адреса переходу),
 \bar{R} – встановлення PC в “0” ($PC := \emptyset$),
 INC – інкремент лічильника ($PC := (PC) + 1$),
 WR – запис слова в лічильник ($PC := (A_{п})$).

Рис. 8.33. Умовне графічне позначення програмного лічильника

Мікро-операція	Стан PC
\bar{R}	$PC := \emptyset$ \ якщо програма починається з адреси \emptyset , \ якщо програма починається не з нульової адреси, то \ першою виконується мікрооперація WR: $PC := (A_{п})$
INC	$PC := (PC) + 1$
	•
	•
	•
INC	$PC := (PC) + 1$
INC	$PC := (PC) + 1$ \ команда переходу 1
WR	$PC := (A_{п1})$ \ $A_{п1}$ – адреса переходу 1
INC	$PC := (PC) + 1$
	•
	•
	•
INC	$PC := (PC) + 1$
INC	$PC := (PC) + 1$ \ команда переходу 2
WR	$PC := (A_{п2})$ \ $A_{п2}$ – адреса переходу 2
INC	$PC := (PC) + 1$
	•
	•
	•

Рис. 8.34. Процес адресації команд програми

Лічильники з неприродним порядком лічби

Лічильники з довільним (неприродним) порядком лічби використовують в пристроях керування, коли кожному стану лічильника відповідають певні сигнали керування. Наприклад, стани лічильника $z_3z_2z_1 = 001, 100, 010$ забезпечують послідовну видачу керуючих сигналів z_1, z_3, z_2 .

Крім того, лічильники з довільним (неприродним) порядком лічби використовують для керування різноманітними індикаторами.

При проектуванні лічильників з неприродним порядком лічби можна використовувати тригери будь-якого типу (RS-, T-, JK-, D-, DV- тощо), але з внутрішньою затримкою.

Розглянемо методику синтезу лічильників з неприродним порядком лічби на прикладі розв'язування задачі.

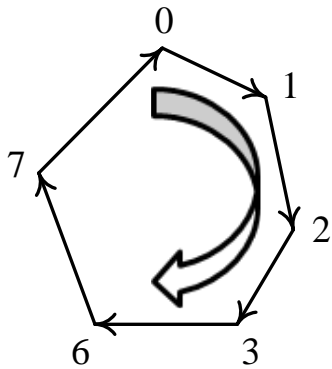


Рис. 8.35. Порядок зміни станів лічильника

Задача 8.3. Синтезувати лічильник на RS-тригерах, який може перебувати в таких станах: 000, 001, 010, 011, 110, 111.

Розв'язування.

1. Визначимо розрядність лічильника. Оскільки лічильник може перебувати в 6-ти станах, порядок зміни яких показано на рис. 8.35, то його модуль M дорівнює шість ($M = 6$).

Розрядність n лічильника визначимо так:

$$n = \lceil \log_2 M \rceil.$$

Отже, $n = \lceil \log_2 6 \rceil = 3$.

До складу лічильника мають входити 3 тригери.

2. Беремо таблицю функцій збудження заданого тригера (RS-тригера) (табл. 8.1):

$Q(t_i)$	$Q(t_{i+1})$	F_S	F_R
0	0	0	×
0	1	1	0
1	0	0	1
1	1	×	0

3. Складаємо таблицю переходів лічильника (табл. 8.9).

Користуючись таблицею функцій збудження T-тригера заповнюємо значення в стовпцях $\Phi_{3S}, \Phi_{3R}, \Phi_{2S}, \Phi_{2R}, \Phi_{1S}, \Phi_{1R}$ таблиці переходів лічильника.

Таблиця 8.9

Таблиця переходів лічильника

Стани лічильника						Функції міжрозрядних переносів					
Поточний стан			Наступний стан								
$Q_3(t_i)$	$Q_2(t_i)$	$Q_1(t_i)$	$Q_3(t_{i+1})$	$Q_2(t_{i+1})$	$Q_1(t_{i+1})$	Φ_{3S}	Φ_{3R}	Φ_{2S}	Φ_{2R}	Φ_{1S}	Φ_{1R}
0	0	0	0	0	1	0	×	0	×	1	0
0	0	1	0	1	0	0	×	1	0	0	1
0	1	0	0	1	1	0	×	×	0	1	0
0	1	1	1	1	0	1	0	×	0	0	1
1	1	0	1	1	1	×	0	×	0	1	0
1	1	1	0	0	0	0	1	0	1	0	1

4. Виконуємо мінімізацію функцій міжрозрядних переносів лічильника:

$$\begin{array}{c}
 \begin{array}{c} \overline{Q_2(t_i)} \\ \downarrow \Phi_{3S} \end{array} \\
 \begin{array}{c|cccc}
 Q_3(t_i) & \times & 0 & - & - \\
 \hline
 & 0 & 1 & 0 & 0
 \end{array} \\
 \hline
 \overline{Q_1(t_i)}
 \end{array}$$

$$\Phi_{3S} = \overline{Q_3}Q_2Q_1$$

$$\begin{array}{c}
 \begin{array}{c} \overline{Q_2(t_i)} \\ \downarrow \Phi_{3R} \end{array} \\
 \begin{array}{c|cccc}
 Q_3(t_i) & 0 & 1 & - & - \\
 \hline
 & \times & 0 & \times & \times
 \end{array} \\
 \hline
 \overline{Q_1(t_i)}
 \end{array}$$

$$\Phi_{3R} = Q_3Q_1$$

$$\begin{array}{c}
 \begin{array}{c} \overline{Q_2(t_i)} \\ \downarrow \Phi_{2S} \end{array} \\
 \begin{array}{c|cccc}
 Q_3(t_i) & \times & 0 & - & - \\
 \hline
 & \times & \times & 1 & 0
 \end{array} \\
 \hline
 \overline{Q_1(t_i)}
 \end{array}$$

$$\Phi_{2S} = \overline{Q_3}Q_1$$

$$\begin{array}{c}
 \begin{array}{c} \overline{Q_2(t_i)} \\ \downarrow \Phi_{2R} \end{array} \\
 \begin{array}{c|cccc}
 Q_3(t_i) & 0 & 1 & - & - \\
 \hline
 & 0 & 0 & 0 & \times
 \end{array} \\
 \hline
 \overline{Q_1(t_i)}
 \end{array}$$

$$\Phi_{2R} = Q_3Q_1$$

$$\begin{array}{c}
 \begin{array}{c} \overline{Q_2(t_i)} \\ \downarrow \Phi_{1S} \end{array} \\
 \begin{array}{c|cccc}
 Q_3(t_i) & 1 & 0 & - & - \\
 \hline
 & 1 & 0 & 0 & 1
 \end{array} \\
 \hline
 \overline{Q_1(t_i)}
 \end{array}$$

$$\Phi_{1S} = \overline{Q_1}$$

$$\begin{array}{c}
 \begin{array}{c} \overline{Q_2(t_i)} \\ \downarrow \Phi_{1R} \end{array} \\
 \begin{array}{c|cccc}
 Q_3(t_i) & 0 & 1 & - & - \\
 \hline
 & 0 & 1 & 1 & 0
 \end{array} \\
 \hline
 \overline{Q_1(t_i)}
 \end{array}$$

$$\Phi_{1R} = Q_1$$

5. Будемо функціональну схему лічильника (рис. 8.36). ▣

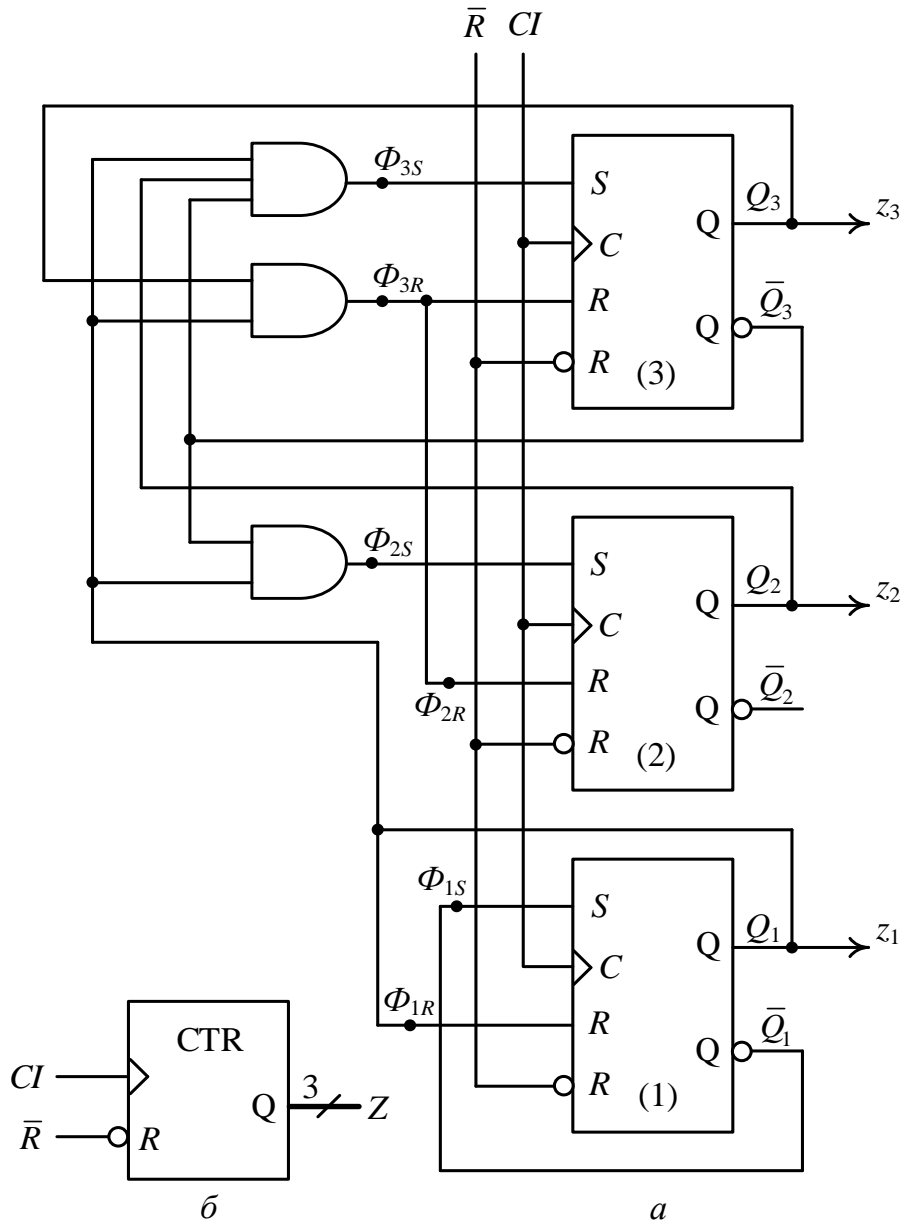


Рис. 8.36. Лічильник з неприродним порядком лічби, що може перебувати в 6-ти станах: *a* – функціональна схема; *б* – умовне графічне позначення

За розглянутою методикою синтезують двійково-десятковий лічильник (BCD-лічильник), а також лічильники Джонсона.

Двійково-десятковий лічильник може перебувати в таких послідовних станах: 0000, 0001, 0010, ... , 1001, 0000, ... (табл. 8.10), він має модуль 10 ($M = 10$).

Лічильник Джонсона (Johnson counter) – це лічильник, послідовність станів якого є кодом Грея. Кожен наступний стан лічильника відрізняється від попереднього стану лише в одному розряді (табл. 8.10).

Таблиця 8.10

Таблиця станів двійково-десятькового лічильника
та лічильників Джонсона

Двійково-десятьковий лічильник	4-розрядний лічильник Джонсона	5-розрядний лічильник Джонсона
0000	0000	00000
0001	1000	10000
0010	1100	11000
0011	1110	11100
0100	1111	11110
0101	0111	11111
0110	0011	01111
0111	0001	00111
1000	0000	00011
1001	...	00001
0000		00000
...		...

Наприклад, 4-розрядний лічильник Джонсона має модуль 8, а 5-розрядний – модуль 10.

У загальному випадку n -розрядний лічильник Джонсона генерує $2n$ станів, тобто має модуль $2n$, де n – кількість тригерів у лічильнику.

Задача 8.4. Синтезувати двійково-десятьковий лічильник на основі D-тригерів.

Розв’язування.

1. Модуль VCD-лічильника дорівнює десять ($M = 10$), оскільки він може перебувати в 10-ти станах: 0, 1, 2, 3, ..., 9 (див. табл. 8.10). Тому його розрядність дорівнює 4: $n = \lceil \log_2 10 \rceil = 4$. Послідовність станів двійково-десятькового лічильника, що описується табл. 8.10, утворює ДДК “8-4-2-1”.

2. Беремо таблицю функцій збудження заданого типу тригера (D-тригера) (табл. 8.3):

$Q(t_i)$	$Q(t_{i+1})$	F_D
0	0	0
0	1	1
1	0	0
1	1	1

3. Складаємо таблицю переходів лічильника (табл. 8.11).

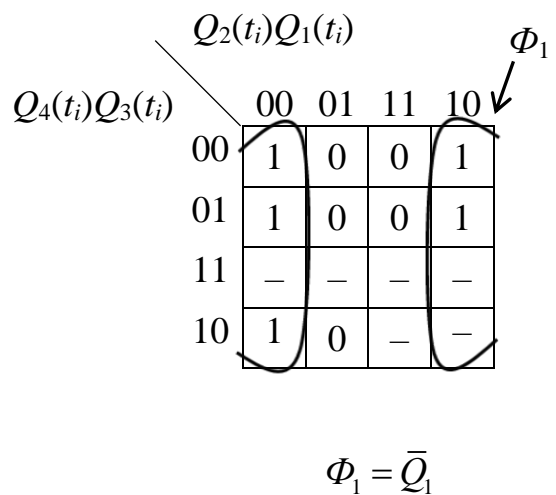
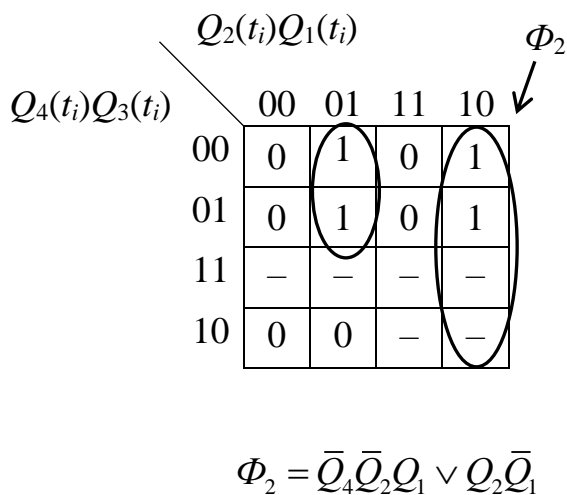
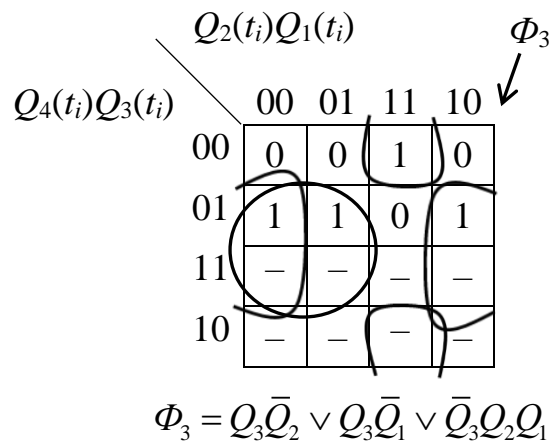
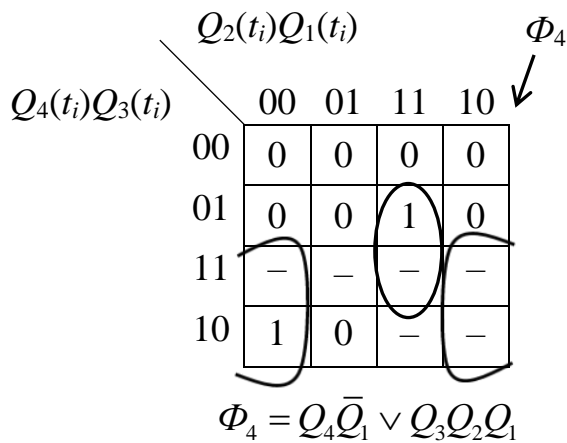
Користуючись таблицею функцій збудження D-тригера заповнюємо значення в стовпцях $\Phi_4, \Phi_3, \Phi_2, \Phi_1$ таблиці переходів лічильника.

Таблиця 8.11

Таблиця переходів двійково-десятькового лічильника

Стани лічильника								Функції міжрозрядних переносів			
Поточний стан				Наступний стан							
$Q_4(t_i)$	$Q_3(t_i)$	$Q_2(t_i)$	$Q_1(t_i)$	$Q_4(t_{i+1})$	$Q_3(t_{i+1})$	$Q_2(t_{i+1})$	$Q_1(t_{i+1})$	Φ_4	Φ_3	Φ_2	Φ_1
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	1	0	0	1	1
0	0	1	1	0	1	0	0	0	1	0	0
0	1	0	0	0	1	0	1	0	1	0	1
0	1	0	1	0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	1	0	1	1	1
0	1	1	1	1	0	0	0	1	0	0	0
1	0	0	0	1	0	0	1	1	0	0	1
1	0	0	1	0	0	0	0	0	0	0	0

4. Виконуємо мінімізацію функцій міжрозрядних переносів лічильника. Для цього застосуємо карти Карно:



5. Будуємо функціональну схему лічильника (рис. 8.37).■

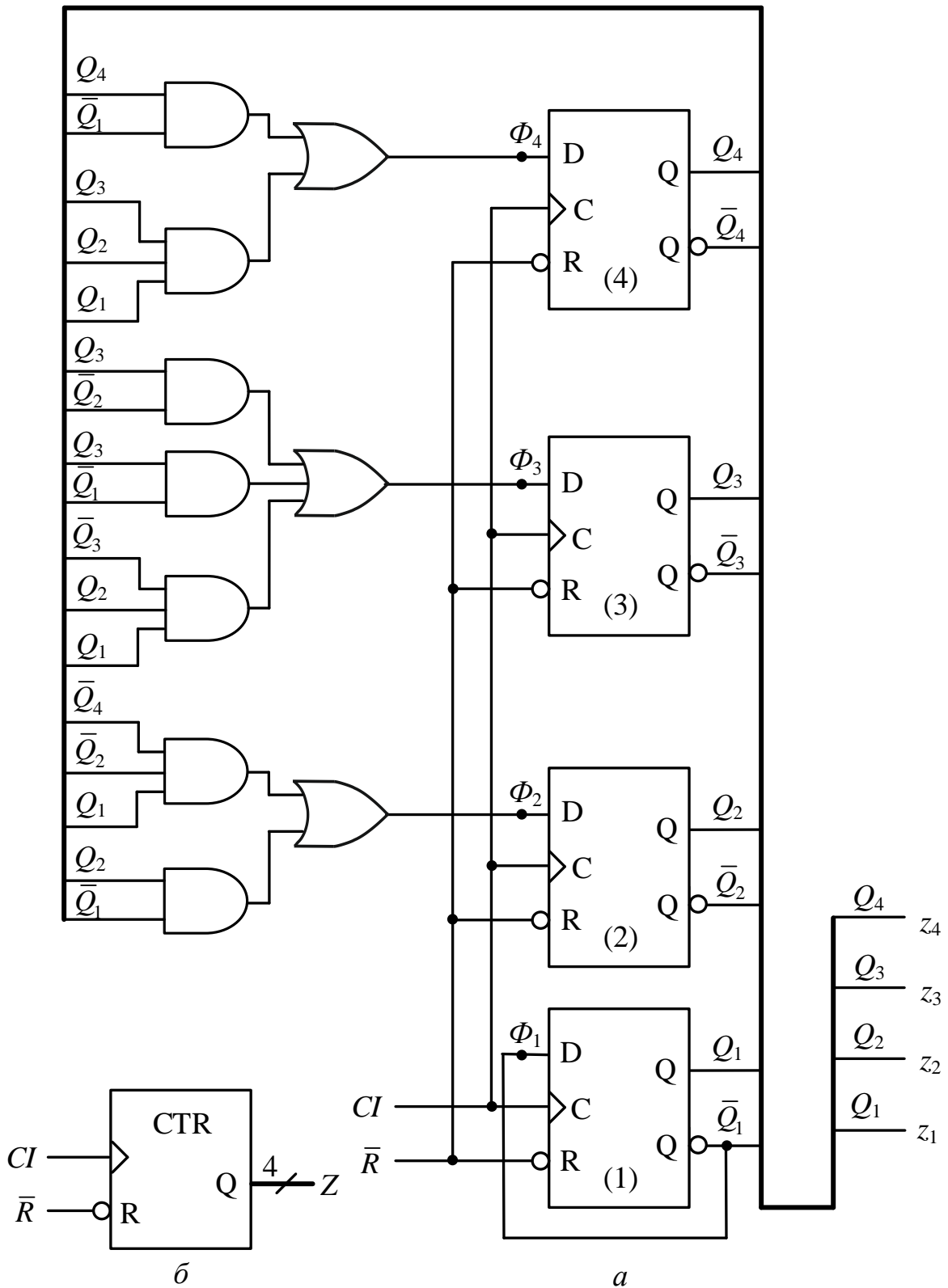


Рис. 8.37. Двійково-десятковий лічильник: *a* – функціональна схема; *б* – умовне графічне позначення

Методику синтезу лічильників з довільним порядком лічби можна застосовувати під час проектування будь-яких лічильників, у тому числі й лічильників з природним порядком лічби.

Функціональні вузли на основі тригерів забезпечують реалізацію надзвичайно важливої функції в комп'ютері – здатність запам'ятовувати слова даних. Зберігання слова та виконання над ним мікрооперацій можливе лише на регістрі. Однак, для розширення функціональних можливостей структур на основі тригерів необхідно проектувати вузли, які б забезпечували можливість роботи не лише з окремим словом, а з групою (масивом) слів. Такі вузли будують на основі регістрів.

Запитання та завдання

1. Який функціональний вузол називають лічильником?
2. Які функції в комп'ютері виконують лічильники?
3. Наведіть класифікацію лічильників.
4. Який лічильник називають n -розрядним? Що називають модулем (періодом) лічильника?
5. До складу лічильника з природним порядком лічби входять 6 тригерів. Яким є модуль лічильника?
6. Назвіть мікрооперації лічби.
7. Які тригери називають лічильними? В яких лічильниках їх можна використовувати?
8. Порівняйте між собою регістри та лічильники. Чим вони відрізняються?
9. Які мікрооперації можуть виконуватись в лічильнику?
10. Запишіть послідовність станів 3-розрядного інкрементного (декрементного) лічильника.
11. Який лічильник називають реверсивним?
12. Порівняйте між собою лічильник з послідовним та паралельним переносом.
13. Запишіть у загальному вигляді функції міжрозрядних переносів інкрементного, декрементного та реверсивного лічильників з паралельним переносом.
14. Нарисуйте структуру 3-го розряду п'ятирозрядного реверсивного лічильника з паралельним переносом на основі T-тригерів.
15. Наведіть умовне графічне позначення 4-розрядного реверсивного лічильника з можливістю встановлення його в довільний початковий стан.
16. Який функціональний вузол називають регістром/лічильником? Для якої мети його використовують у комп'ютері?
17. Дайте визначення програмного лічильника. Яку роль він відіграє в комп'ютері?

18. Охарактеризуйте функціонування програмного лічильника в процесі виконання програми обчислень.
19. Сформулюйте методику синтезу лічильника з довільним (неприродним) порядком лічби.
20. Який модуль має 8-розрядний лічильник Джонсона?

Задачі для самостійного розв'язування

1. Побудувати 4-розрядний асинхронний декрементний лічильник на синхронних JK-тригерах. Навести умовне графічне позначення лічильника.
2. Побудувати 3-розрядний інкрементний лічильник з паралельним переносом на T-тригерах. Навести умовне графічне позначення лічильника.
3. Синтезувати лічильник на JK-тригерах, який може перебувати в таких станах: 000, 010, 011, 101, 110.
4. Синтезувати двійково-десятковий лічильник на T-тригерах.
5. Синтезувати 4-розрядний лічильник Джонсона на RS-тригерах.
6. Синтезувати лічильник на D-тригерах, послідовністю станів якого є ДДК "4-2-2-1".
7. Визначити послідовність станів, починаючи зі стану 000, 3-розрядного лічильника на основі D-тригерів, якщо відомі його функції міжрозрядних переносів:

$$\Phi_3 = \bar{Q}_3 Q_2 \vee \bar{Q}_2 Q_1,$$

$$\Phi_2 = Q_2 \bar{Q}_1 \vee Q_3 Q_2 \vee \bar{Q}_3 \bar{Q}_2 Q_1,$$

$$\Phi_1 = \bar{Q}_3 \bar{Q}_2 \vee \bar{Q}_3 Q_1 \vee Q_3 Q_2 \bar{Q}_1.$$

8. На основі T-тригерів синтезувати лічильник, який забезпечує отримання такої послідовності: 000, 001, 100, 110, 101, 000,
9. На основі RS-тригерів синтезувати лічильник, що відтворює такий 3-розрядний код Грея: 000, 100, 110, 010, 011, 111, 101, 001, 000,

9.

ФУНКЦІОНАЛЬНІ ВУЗЛИ НА ОСНОВІ РЕГІСТРІВ

Регістр як функціональний вузол забезпечує зберігання та оброблення одного слова даних. Розглянемо, яким чином на основі регістрів можна створювати складніші функціональні вузли, які б істотно розширювали можливості роботи зі словами.

Такими функціональними вузлами є накопичувальні суматори, регістрові запам'ятовувальні пристрої та стеки.

9.1. Накопичувальні суматори

Накопичувальний суматор (НСМ) забезпечує арифметичне додавання послідовності слів D_1, D_2, \dots, D_m із запам'ятовуванням результату:

$$НСМ := \sum_{i=1}^m D_i \text{ (рис. 9.1).}$$

На k -розрядний вхід накопичувального суматора по чергово надходять слова D_i ($i = 1, 2, \dots, m$). В i -му такті виконується мікрооперація додавання $НСМ := (НСМ) + D_i$.

В m -му такті формується n -розрядна сума $S := \sum_{i=1}^m D_i$, де $n > k$, а також вихідний перенос CO .

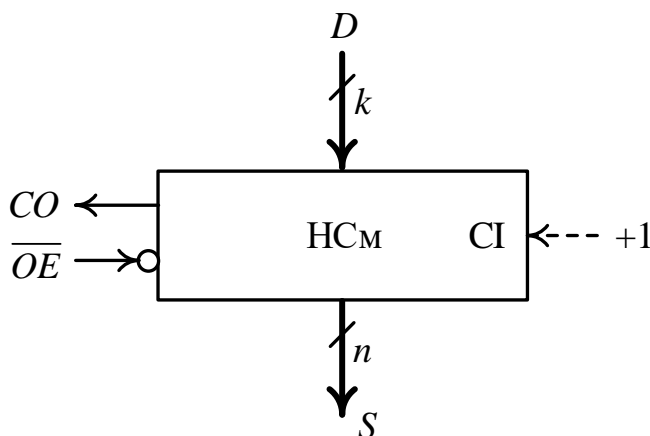


Рис. 9.1. Умовне графічне позначення накопичувального суматора

Накопичувальний суматор має вхід CI – вхідного переносу, на який, у разі потреби, може подаватись вхідний перенос $+1$.

Накопичена сума видається на вихід S при надходженні сигналу $\overline{OE} = 0$, OE (output enable) – дозвіл видачі.

Структурно накопичувальний суматор складається з комбінаційного суматора (Σ) та регістра (RG) (рис. 9.2).

Керування видачею накопиченої суми після m тактів роботи забезпечує сигнал \overline{OE} , який подають на перші входи двовходових

елементів І (рис. 9.3). На другі входи елементів І подають відповідні розряди з виходу регістра RG.

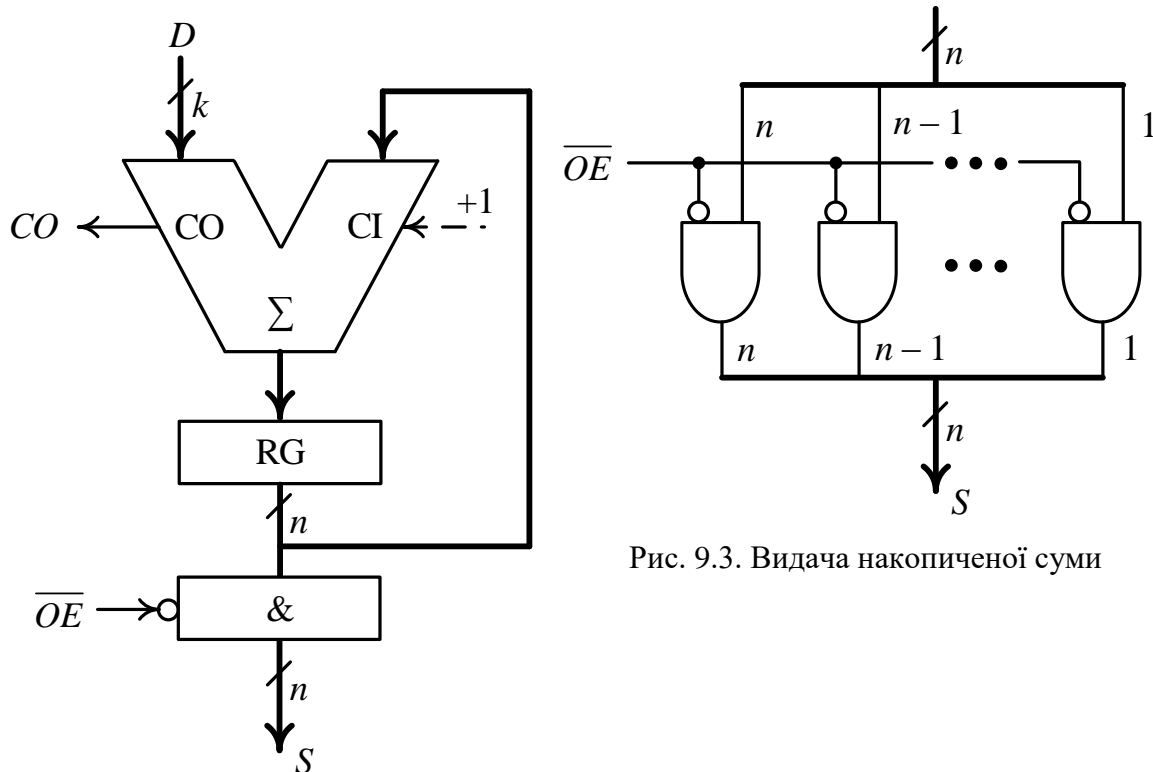


Рис. 9.2. Будова накопичувального суматора

Рис. 9.3. Видача накопиченої суми

Запитання та завдання

1. Які структурні одиниці інформації обробляють в регістрі?
2. Яку функцію виконує накопичувальний суматор?
3. Які структурні компоненти входять до складу накопичувального суматора?
4. Які переноси використовують в накопичувальному суматорі?
5. Як здійснюють видачу накопиченої суми з комбінаційного суматора?

9.2. Регістрові запам'ятовувальні пристрої

На основі регістрів будують регістрові запам'ятовувальні пристрої (РЗП). РЗП застосовують для побудови: надоперативної пам'яті; як буфери даних між швидкодіючим процесором та повільнодіючими

зовнішніми пристроями; для побудови багатоадресних оперативних запам'ятовувальних пристроїв (ОЗП).

За способом доступу до слів даних регістри ОЗП бувають з довільним та послідовним доступом.

Регістри ОЗП з довільним доступом у будь-який момент часу дозволяють адресувати будь-який регістр ОЗП з метою запису або зчитування інформації. В регістрових ОЗП з послідовним доступом для звертання до потрібного регістра потрібен перебір адрес – від адреси першого регістра до адреси потрібного регістра.

Як приклад розглянемо регістровий ОЗП, що складається з 16-ти 8-розрядних регістрів (рис. 9.4). Такий ОЗП має організацію 16 x 8, тобто така структура здатна зберігати 16 восьмирозрядних слів, які розташовуються в регістрах R0, R1, ..., R15.

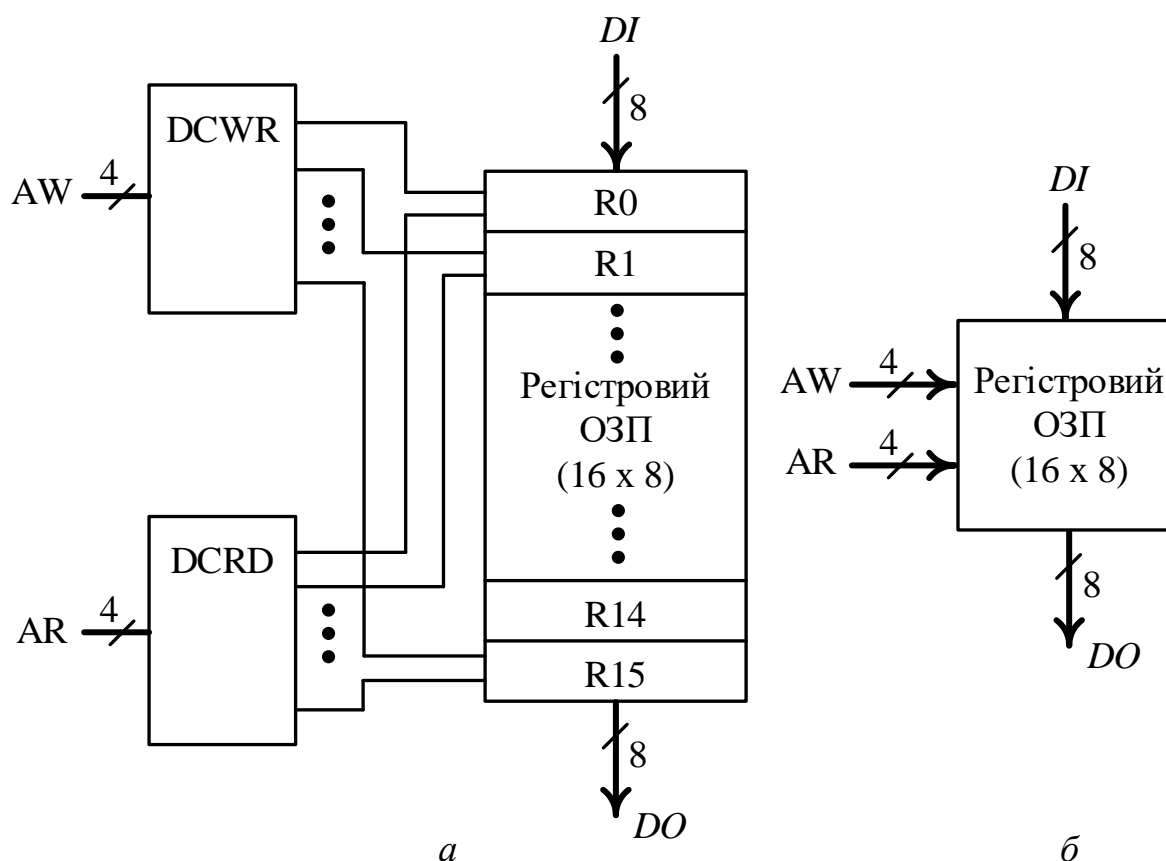


Рис. 9.4. Двоадресний (двопортовий) регістровий ОЗП:
 а – структурна організація; б – умовне графічне позначення

Для доступу до регістрів використовують два дешифратори: DCWR – дешифратор для запису слова в регістр, DCRD – дешифратор для читання слова з регістра. Відповідно, на регістровий ОЗП подають дві 4-розрядні адреси: AW – адресу для запису слова, AR – адресу для читання слова.

Наприклад, якщо в регістровий ОЗП необхідно записати слово, то його подають на вхід DI, а на DCWR подають адресу регістра, в який слово слід записати. Так, якщо $AW := 0101$, то слово з входу DI буде записано в R5.

Якщо з регістрового ОЗП необхідно зчитати слово, то на DCRD подають адресу цього слова, тобто номер регістра, в якому слово зберігається. Зчитане слово надходить на вихід DO.

Наприклад, якщо $AR := 1001$, то слово буде зчитано з регістра R9.

Запис слова в деякий регістр та зчитування слова з довільного регістра можна здійснювати одночасно, за умови, що адреси для запису і читання відрізняються.

Оскільки мікрооперацію запису слова можна виконувати одночасно з мікрооперацією зчитування слова, то вважається, що подібний РЗП є двопортовим (двоадресним) – один порт використовується лише для запису, а інший – лише для читання слів.

У зазначеній структурі доступ до регістрів – довільний, оскільки в будь-який момент часу можна на адресні входи AW та AR подати будь-які (довільні) адреси, а зв'язки між регістрами – відсутні.

Регістрові запам'ятовувальні пристрої можна організовувати і в інший спосіб – з міжрегістровими зв'язками.

Запитання та завдання

1. Для виконання яких функцій в обчислювальних засобах застосовують регістрові запам'ятовувальні пристрої?
2. Які способи доступу до регістрів використовуються в регістрових ОЗП?
3. Який регістровий ОЗП називають двопортовим?
4. Які структурні компоненти входять до складу двопортового регістрового ОЗП?
5. Сформулюйте умову коректного функціонування двопортового регістрового ОЗП.

9.3. Стеки

Стек (stack) – це регістровий запам'ятовувальний пристрій, який має спеціальну організацію на основі міжрегістрових зв'язків.

Розрізняють два види стеків:

- стек з організацією FIFO (first in – first out, першим прийшов – першим вийшов);
- стек з організацією LIFO (last in – first out, останнім прийшов – першим вийшов).

Стек з організацією FIFO часто називають буфером або чергою (queue).

Розглянемо функціонування стека з організацією FIFO (рис. 9.5), до складу якого входять 32 8-розрядні регістри (R_0, R_1, \dots, R_{31}).

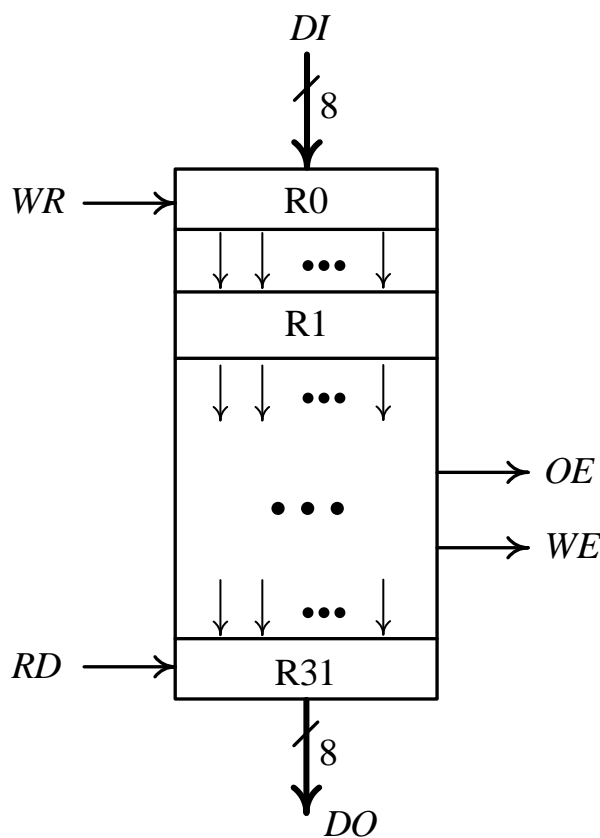


Рис. 9.5. Структура стека з організацією FIFO

Стек має інформаційний вхід DI (Data Input) та інформаційний вихід DO (Data Output). Явна адресація слів (регістрів) у стеку відсутня – запис слова завжди здійснюють в регістр R_0 , а читання слова завжди відбувається з регістра R_{31} .

Керування стеком забезпечують 4 сигнали: WR (write) – записати чергове слово в регістр R_0 ; RD (read) – зчитати слово з регістра R_{31} ; WE (write enable) – стек вільний, дозволяється запис; OE (output enable) – стек заповнено, дозволяється зчитування.

Стек заповнюють (вивільняють) блоками по 32 слова.

Коли стек вільний, $WE = 1$, $OE = 0$.

Стек заповнюють слово за словом; заповнення R_0 черговим словом відбувається коли $WR = 1$; під час запису чергового слова в R_0 слово, що в ньому зберігалось, передається в R_1 , з R_1 в R_2 і т.д. Заповнення стека триває 32 такти.

Заповнення стека можна узагальнено описати так:

$$\begin{cases} R_0 := (DI) \\ R(i) := R(i-1), \quad i = 1, 2, \dots, 31. \end{cases}$$

Після заповнення стека 32-ма словами формуються сигнали: $WE = 0$, $OE = 1$ – стек заповнено.

Читання даних зі стека відбувається слово за словом з регістра R_{31} . Кожне читання ініціюють сигналом мікрооперації читання – $RD = 1$. Після зчитування чергового слова з R_{31} , слова даних, що зберігаються в регістрах $R_0 - R_{30}$, просуваються на один регістр вниз: слово з R_{30} надходить в R_{31} , слово з R_{29} надходить в R_{30} і т.д.

Узагальнено зчитування слів зі стека (вивільнення стека), що триває 32 такти, можна описати так:

$$\begin{cases} DO := (R31) \\ R(j) := R(j-1), \quad j = 31, 30, \dots, 1. \end{cases}$$

Після зчитування зі стека останнього слова формуються сигнали: $WE = 1, OE = 0$ – стек вільний.

Далі в стек записують новий блок з 32-х слів.

Стек з організацією FIFO використовують як буфер даних між пристроями з різною швидкістю (наприклад, між процесором та повільнодіючим зовнішнім пристроєм).

Стек з організацією LIFO складається зі стовпця регістрів, що мають порозрядні двонапрямні вертикальні зв'язки, які дають можливість перемішувати слова даних вниз та вгору (рис. 9.6).

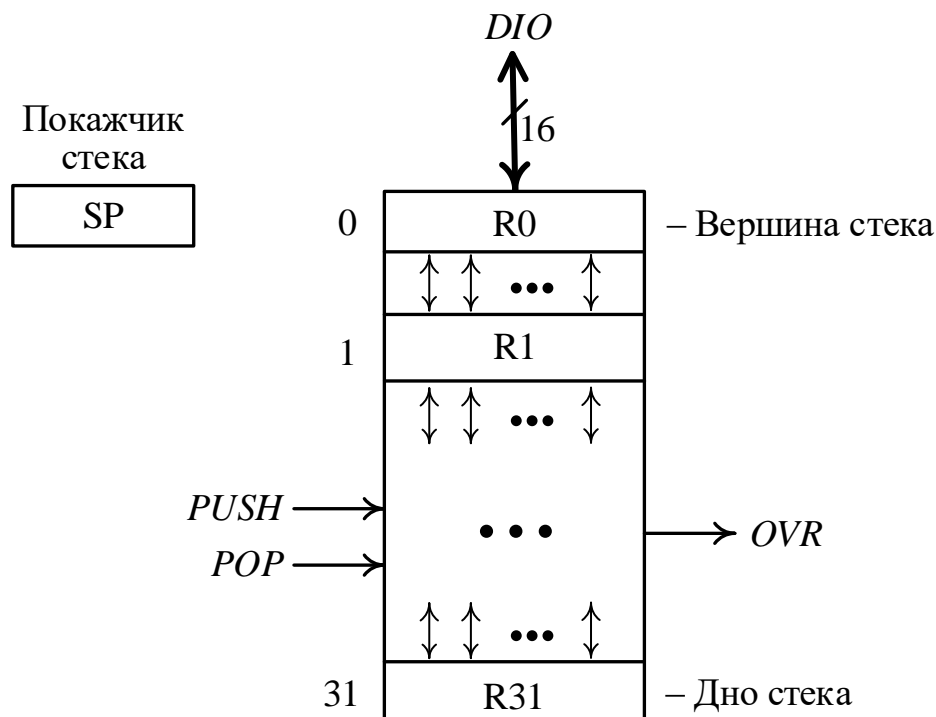


Рис. 9.6. Структура стека з організацією LIFO

Розглянемо функціонування 16-розрядного стека з організацією LIFO, до складу якого входять 32 регістри.

Стек має інформаційний двонапрямний вхід-вихід DIO (data input-output), через який слова записуються (заштовхуються) в стек та виводяться (виштовхуються) зі стека.

Стек з такою організацією часто називають стеком магазинного типу.

У стеку з організацією LIFO виконуються дві мікрооперації: PUSH – заштовхнути (записати) слово в стек, запис здійснюється завжди в регістр R0; POP – виштовхнути (зчитати) слово зі стека, зчитування також завжди відбувається з регістра R0.

Якщо стек повністю заповнений, і має місце спроба записати (заштовхнути) в стек 33-тє слово, то формується сигнал $OVR = 1$ – переповнення (overflow).

Для контролю за станом заповнення (вивільнення) стека застосовують спеціальний показчик, що має назву SP – stack pointer (покажчик стека). SP – містить адресу останнього заповненого регістра стека.

Покажчик стека (SP) є регістром/лічильником, тобто виконує функції як регістра, так і реверсивного лічильника. Після запису (заштовхування) в стек слова стан покажчика стека збільшується на одиницю: $SP := (SP) + 1$; після кожного зчитування (виштовхування) слова зі стека стан покажчика стека зменшується на одиницю: $SP := (SP) - 1$.

Коли стек незаповнений (вільний) всі розряди регістра SP перебувають у третьому стані (рис. 9.7).

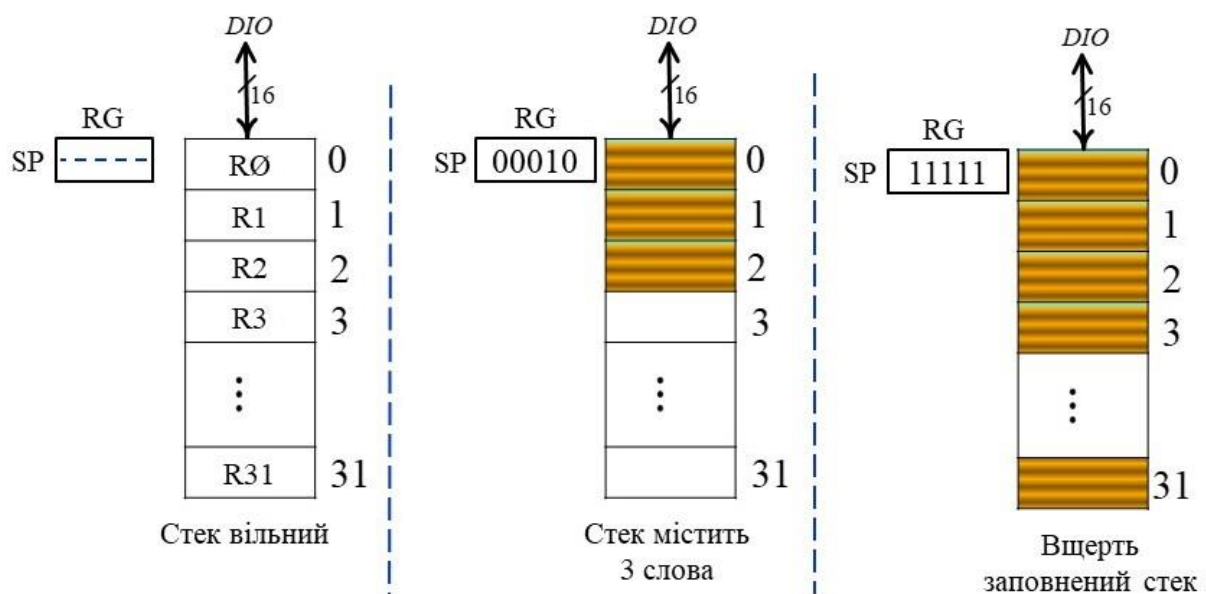


Рис. 9.7. Процес заповнення стека з організацією LIFO

Після запису в стек першого слова ($R0 := (DIO)$) в покажчику стека встановлюється стан 00000 – адреса (останнього) заповненого регістра – регістра R0.

Якщо в стек записано 3 слова, то покажчик стека міститиме код 00010 – адреса останнього заповненого регістра, тобто адреса регістра R2 (див. рис. 9.7).

Під час заштовхування (запису) в стек чергового слова вміст стека зсувається вниз на один регістр:

$$\text{PUSH} \rightarrow \begin{cases} \text{R0} := (\text{DIO}) \\ \text{R}(i) := \text{R}(i-1) \setminus \text{SP} := (\text{SP}) + 1, \quad i = 1, 2, \dots, 31. \end{cases}$$

Під час виштовхування (зчитування) слова зі стека вміст стека зсувається вгору на один регістр:

$$\text{POP} \rightarrow \begin{cases} \text{DIO} := (\text{R0}) \\ \text{R}(j) := \text{R}(j+1) \setminus \text{SP} := (\text{SP}) - 1, \quad j = 0, 1, 2, \dots, 30. \end{cases}$$

Слово, що перебувало у вершині стека (в R0), виштовхується та надходить на шину DIO.

Стек з організацією LIFO застосовують для реалізації механізму виклику підпрограм з основної програми, а також для реалізації механізму вкладених підпрограм.

Розглянемо апаратне забезпечення механізму реалізації вкладених підпрограм.

Нехай основна програма, що починається оператором BEGIN та завершується оператором END, під час свого виконання викликає підпрограму (п/п) SBR1, яка, в свою чергу, викликає підпрограму SBR2 (рис. 9.8).

Виклик підпрограми здійснюється командою CALL. Кожна підпрограма завершується командою RET (return – повернення).

Вважатимемо, що основна програма в пам'яті починається за адресою 0A00h, підпрограма SBR1 – за адресою 0B00h, а підпрограма SBR2 – за адресою 0C00h.

У реалізації механізму вкладених підпрограм беруть участь програмний лічильник PC та стек з організацією LIFO.

Нехай при виконанні основної програми за адресою 0A20h зустрічається команда CALL SBR1 – виклик підпрограми SBR1. В цей момент у програмному лічильнику вже сформована адреса 0A21h – адреса наступної команди, а стек є незаповненим.

Під час дешифрування команди CALL з'ясовується, що це команда переходу – безумовного виклику підпрограми. Тому в стек записується вміст регістра PC, а в регістр PC (програмний лічильник) записується початкова адреса підпрограми SRB1, тобто

$$\begin{cases} \text{стек (R0)} := (\text{PC}), \\ \text{PC} := 0B00. \end{cases}$$

Отже, в стек потрапляє 0A21h – адреса повернення в основну програму.

Далі виконується підпрограма SBR1.

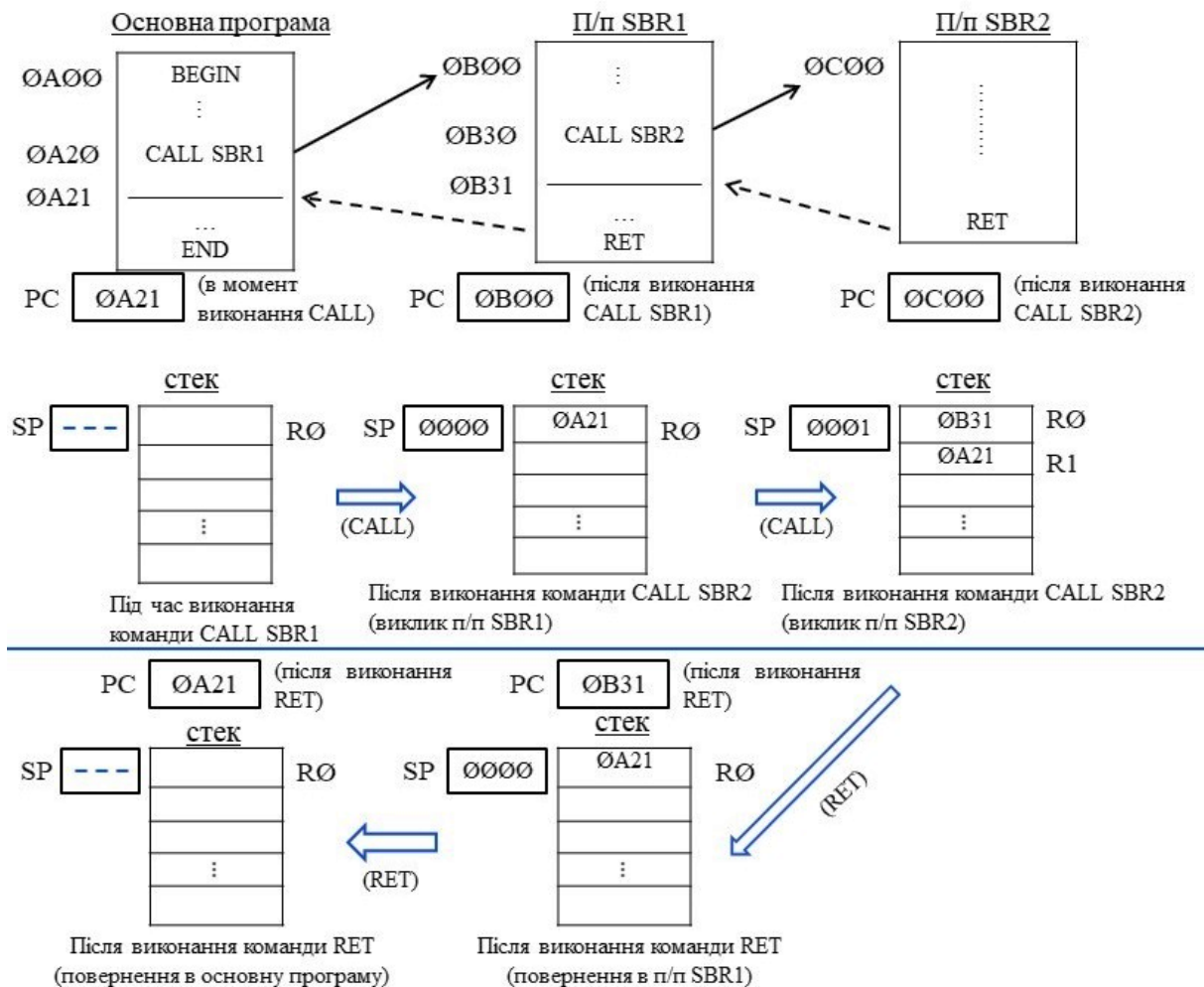


Рис. 9.8. Застосування стека з організацією LIFO для реалізації механізму вкладених підпрограм

Нехай за адресою 0B30h у цій підпрограмі зустрічається команда CALL SBR2 – виклик підпрограми SBR2, яка розміщується в пам'яті за адресою 0C00h. Тоді в стек записується 0B31h – адреса наступної за CALL команди, і ця адреса є адресою повернення в підпрограму SBR1 з підпрограми SBR2, а в PC записується початкова адреса підпрограми SBR2.

Отже,

$$\begin{cases} \text{стек (R0)} := 0B31, \\ R1 := R0 \setminus 0A21, \\ PC := 0C00. \end{cases}$$

Починає виконуватись підпрограма SBR2.

Після того, як у підпрограмі SBR2 зустрічається команда RET (повернення) відбувається виштовхування слова з вершини стека (а це –

адреса ØB31h – адреса повернення в підпрограму SBR1) та його запис у програмний лічильник, тобто

$$\begin{cases} \text{PC} := (\text{стек}(\text{R0})), \\ \text{R0} := \text{R1} \setminus \text{ØA21}. \end{cases}$$

Це приводить до поновлення виконання підпрограми SBR1, починаючи з адреси ØB31h .

Після того, як у підпрограмі SBR1 зустрічається команда RET, відбувається виштовхування слова з вершини стека – а це адреса повернення в основну програму ØA21h , та його запис в програмний лічильник, тобто

$$\text{PC} := (\text{стек}(\text{R0})) \setminus \text{ØA21},$$

а стек вивільняється.

Виконання основної програми продовжується з адреси ØA21h .

Таким чином, під час реалізації механізму вкладених підпрограм стек з організацією LIFO використовують для тимчасового зберігання адрес повернення: з поточної підпрограми в попередню, а потім з першої підпрограми – в основну програму.

Застосування стека з організацією LIFO забезпечує високу ефективність виконання програм в комп'ютері.

Порівнюючи програмний лічильник (PC) та покажчик стека (SP) слід зазначити наступне: PC є регістром/лічильником, на якому виконуються мікрооперації *reset*, *write*, *increment*, а SP є регістром/лічильником, на якому виконуються мікрооперації *reset*, *write*, *increment*, *decrement*.

Запитання та завдання

1. Перелічіть функціональні вузли, які забезпечують роботу з масивом слів.
2. Дайте визначення стека. Назвіть види стеків.
3. Як функціонує стек з організацією FIFO? Для якої мети його використовують?
4. Які мікрооперації над словами виконують в стеку з організацією FIFO?
5. Як функціонує стек з організацією LIFO?
6. Які мікрооперації виконують в стеку з організацією LIFO?
7. Яку функцію в стеку виконує покажчик стека SP? Охарактеризуйте його як функціональний вузол.
8. Для якої мети в комп'ютері застосовують стек з організацією LIFO?
9. Порівняйте між собою функціональні вузли – покажчик стека SP та програмний лічильник PC.
10. Як взаємодіють між собою стек та програмний лічильник під час виклику підпрограми?

9.4. Взаємозв'язок апаратних засобів цифрової обчислювальної техніки

Напівпровідникові транзистори є фізичною основою сучасних апаратних засобів обчислювальної техніки.

На основі транзисторів виготовляють логічні елементи (І, АБО-НЕ, І-НЕ, АБО-НЕ тощо), а також елементи пам'яті, в яких інформація зберігається у вигляді заряду (рис. 9.9). Наявність заряду позначає логічну одиницю, а відсутність – логічний нуль. На основі транзисторних структур, що забезпечують тривале зберігання заряду, проектують динамічні ОЗП (ДОЗП), репрограмовні постійні запам'ятовувальні пристрої (РПЗП), а також flash-пам'ять.

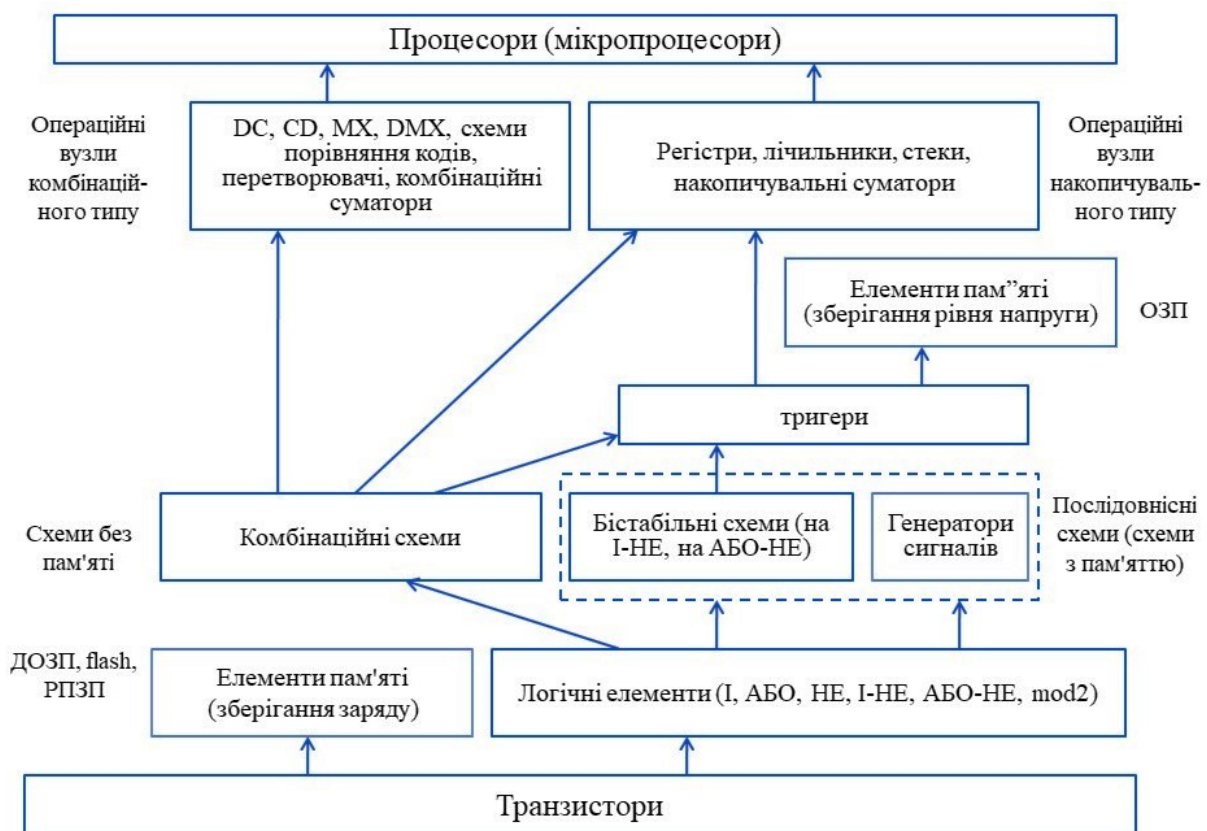


Рис. 9.9. Класифікація апаратних засобів обчислювальної техніки

На основі логічних елементів створюють два види схем – схеми без пам'яті (це комбінаційні схеми) та послідовнісні схеми (схеми з пам'яттю). Послідовнісні схеми проектують на основі логічних елементів І-НЕ або на основі елементів АБО-НЕ.

Найважливішими серед послідовнісних схем є бістабільні схеми, які становлять основу (серцевину) тригерів. Деякі послідовнісні схеми можуть виконувати функцію генератора сигналів.

Операційні вузли комбінаційного типу, такі як дешифратори, шифратори, мультиплексори, перетворювачі кодів тощо, синтезують як комбінаційні схеми.

Тригерні схеми (тригери) є надзвичайно важливими функціональними вузлами комп'ютерної техніки, оскільки на їх основі проектують оперативні запам'ятовувальні пристрої (пам'ять комп'ютера) та операційні вузли накопичувального типу. Тригер, який є об'єднанням бістабільної схеми (запам'ятовувального елемента) та комбінаційної схеми (схеми керування), забезпечує надійне та тривале зберігання одного біта.

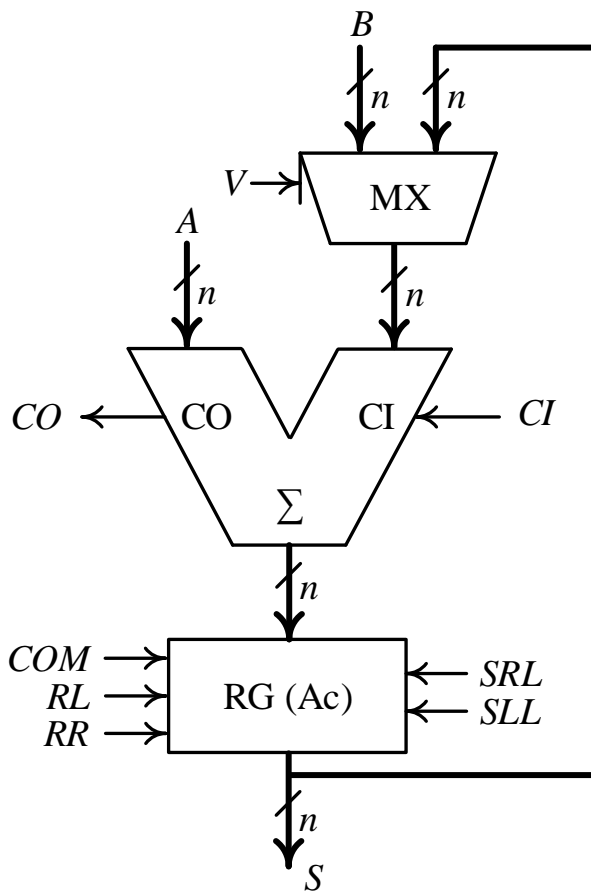


Рис. 9.10. Арифметико-логічний пристрій процесора

На основі тригерів та комбінаційної схеми (схеми керування) проектують регістри, лічильники, стеки, накопичувальні суматори – структури, що забезпечують виконання мікрооперацій над словами даних зі збереженням результату.

Процесор (мікропроцесор) комп'ютера містить у своєму складі операційні вузли комбінаційного та накопичувального типів. Серцевиною процесора є арифметико-логічний пристрій (АЛП) (рис. 9.10), головними вузлами якого є комбінаційний суматор (Σ), акумулятор (регістр зсуву) (A_c) та мультиплексор. Комбінаційний суматор виконує мікрооперацію додавання вхідних слів A і B , або додавання вхідного слова A та накопиченого результату (S) в акумуляторі.

Акумулятор є регістром зсуву, що забезпечує виконання мікрооперацій: COM – інвертування слова, SRL – логічний зсув вправо, SLL – логічний зсув вліво, RL – циклічний зсув вліво, RR – вправо.

В АЛП, отже, виконують такі види мікрооперацій:

- 1) $A_c := (A) + (B) + CI, CI \in \{0, 1\}$
- 2) $A_c := (A_c) + (A) + CI$
- 3) $A_c := (A_c) \ \backslash \ SRL$
- 4) $A_c := 2(A_c) \ \backslash \ SLL$

$$5) A_c := RR(A_c)$$

$$6) A_c := RL(A_c)$$

$$7) A_c := (\overline{A_c}) \setminus COM$$

Виконання перших двох типів мікрооперацій забезпечує комбінаційний суматор, мікрооперацій 3 – 6 – регістр зсуву, а мікрооперації 7 – регістр.

Крім того, слід взяти до уваги, що переважна більшість складних операцій в комп'ютері зводиться до циклічних процесів – багаторазового повторення певної послідовності мікрооперацій.

Таким чином, для реалізації будь-яких обчислень процесор повинен мати у своєму складі апаратні засоби, які б забезпечували виконання чотирьох типів мікрооперацій – додавання, зсувів, інвертування та лічби:

- додавання (комбінаційний суматор),
 - зсуви (регістр зсуву),
 - інвертування (регістр),
 - лічба (регістр/лічильник).
- } регістри

У мовах програмування є спеціальні команди, що дозволяють програмісту здійснювати безпосередній доступ до тих чи інших функціональних вузлів процесора з метою запису в них слова або окремих бітів, зчитування, модифікації окремих бітів та інших елементарних дій. Тому оперують таким поняттям як *програмістська модель процесора* (рис. 9.11).

Програміст сприймає процесор як набір функціональних вузлів, до яких він може здійснювати доступ за допомогою відповідних команд мови програмування. Програмістська модель включає такі вузли: регістри загального призначення, слово стану програми (PSW), програмний лічильник (PC), акумулятор (A), стек та покажчик стека.

Всі перелічені вузли будують на основі регістрів.

Отже, регістр є основою функціональних вузлів процесора.

Запитання та завдання

1. Що є фізичною основою сучасних апаратних засобів обчислювальної техніки?
2. Які компоненти комп'ютера виготовляють на основі транзистора?
3. Які види схем створюють на основі логічних елементів?
4. Яку роль відіграють бістабільні схеми в комп'ютері?
5. Які функціональні вузли комп'ютера створюють на основі тригерів?



Рис. 9.11. Програмістська модель процесора

6. Які структурні компоненти комп'ютера виконують мікрооперації над словами даних?
7. Перелічіть структурні компоненти арифметико-логічного пристрою процесора.
8. Які типи мікрооперацій мають виконувати апаратні засоби комп'ютера, щоб забезпечити виконання довільних обчислень? Які функціональні вузли забезпечують виконання цих типів мікрооперацій?
9. Перелічіть компоненти програмістської моделі процесора.

10. СИНТЕЗ КЕРУЮЧИХ ЦИФРОВИХ АВТОМАТІВ

Реалізація обчислювального процесу в комп'ютері ґрунтується на взаємодії процесора та оперативної пам'яті (рис. 10.1). Програма обчислень зберігається в оперативній пам'яті. Використовуючи шину адреси (A) і шину даних (D) процесор зчитує команди програми та виконує їх, записуючи, у разі потреби, проміжні результати в пам'ять або зберігаючи їх в регістрах загального призначення процесора.

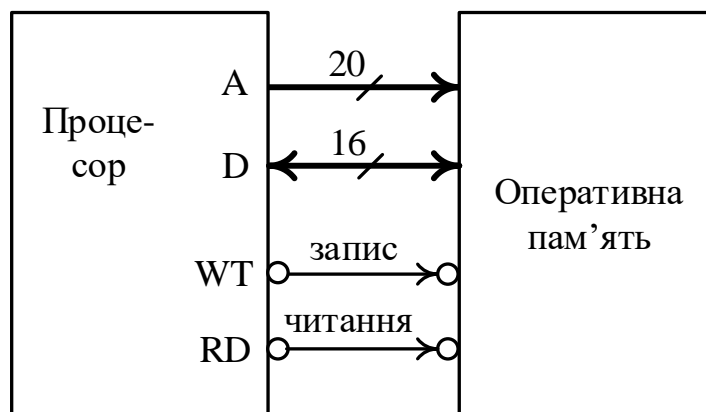


Рис. 10.1. Взаємодія процесора з оперативною пам'яттю

Розглянемо, якими засобами можна забезпечити реалізацію обчислювального процесу в процесорі комп'ютера.

10.1. Принцип мікропрограмного керування

Виконання команд програми обчислень в комп'ютері організовують на основі *принципу мікропрограмного керування*, який полягає в наступному.

Будь-яку операцію, що задається командою програми, розглядають як складну дію та розкладають на елементарні машинні дії над словами даних, які називають *мікроопераціями*. Такими елементарними діями є запис слова в регістр, видача слова з регістра (в прямому або інверсному коді), інвертування слова, зсув слова вліво або вправо (логічний, арифметичний), інкремент, декремент, додавання, виконання логічної операції (OR, AND, XOR) тощо.

Мікрооперація (елементарна дія) виконується за один такт.

Послідовність мікрооперацій, виконання яких приводить до отримання результату операції (команди), називають *мікроалгоритмом операції* (рис. 10.2).

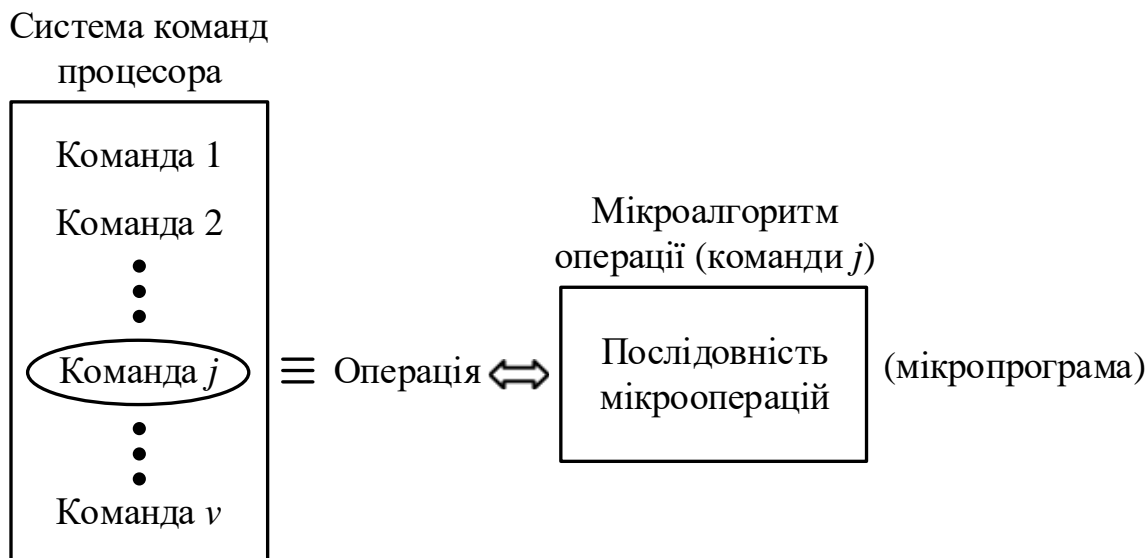


Рис. 10.2. Відповідність між командами та мікропрограмами

Після виконання кожної мікрооперації формуються *логічні умови* – ознаки результату мікрооперації. Логічні умови впливають на порядок виконання мікрооперацій.

Принцип мікропрограмного керування означає, що кожній команді з системи команд процесора ставиться у відповідність певна послідовність мікрооперацій, яка є мікропрограмою, що реалізує дану команду. Послідовність мікрооперацій, що відповідає даній команді, є мікроалгоритмом операції (команди).

Мікроалгоритми операцій (команд) реалізують за допомогою операційних пристроїв (ОПр).

Процесор комп'ютера можна розглядати як композицію операційних пристроїв, на кожний з яких покладається виконання певної сукупності операцій (команд). Разом операційні пристрої процесора реалізують систему команд процесора (заданий список команд).

Окремо взятий операційний пристрій виконує операції (команди) з набору операцій (команд) $U = \{u_1, u_2, \dots, u_v\}$ над операндами (словами) з множини D з метою отримання результатів (слів), які становлять множину слів W (рис. 10.3).

У кожний момент часу операційний пристрій здатний виконувати лише одну операцію u_j ($j = 1, 2, \dots, v$).

Акт роботи операційного пристрою полягає в обчисленні слова w_i :

$$\{w_i\} = u_j \{d_i\}.$$

Операційний пристрій складається з двох частин – операційного автомата (ОА) та керуючого автомата (КА).

Операційний автомат слугує для тимчасового зберігання інформаційних слів, виконання мікрооперацій над ними та вироблення логічних умов $X = \{x_1, x_2, \dots, x_n\}$ – ознак результату. До складу операційного

автомата можуть входити такі функціональні вузли: регістри, лічильники, мультиплексори, комбінаційні суматори, дешифратори, схеми порівняння кодів тощо.

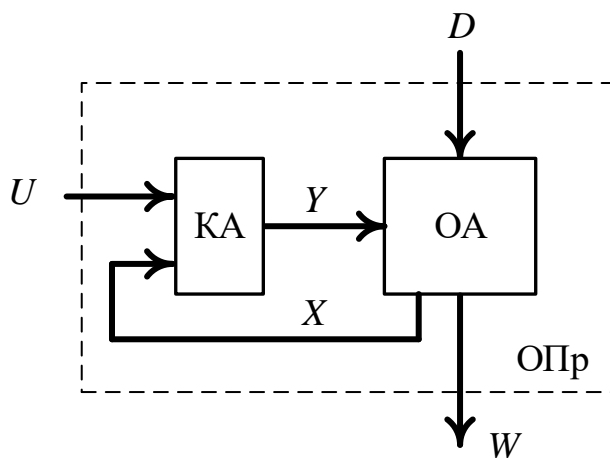


Рис. 10.3. Структура операційного пристрою

Мікрооперації збуджуються керуючими сигналами $Y = \{y_1, y_2, \dots, y_m\}$. Керуючий сигнал y_i відповідає одній мікрооперації.

Керуючий автомат, відповідно до значення u_j з набору U та сигналів логічних умов, генерує послідовність керуючих сигналів Y .

Призначення керуючого автомата – формувати послідовність мікрооперацій, яка відповідає поточній виконуваній команді програми.

Мікроалгоритми операцій (команд) можна реалізовувати двома способами – апаратним та мікропрограмним. Відповідно, розрізняють два способи побудови керуючих цифрових автоматів:

- керуючі автомати з жорсткою логікою (апаратний спосіб реалізації команд),
- керуючі автомати з програмовною логікою (мікропрограмний спосіб реалізації команд (мікроалгоритмів операцій)).

Запитання та завдання

1. Сформулюйте принцип мікропрограмного керування.
2. Перелічіть мікрооперації, які використовуються в комп'ютері для оброблення слів даних.
3. Що називають мікроалгоритмом операції?
4. Які структурні компоненти входять до складу операційного пристрою?
5. Опишіть функціонування операційного пристрою.
6. В чому полягає функціональне призначення операційного автомата операційного пристрою?
7. Яку функцію виконує керуючий автомат у складі операційного пристрою?
8. Назвіть два способи побудови керуючих цифрових автоматів.

10.2. Керуючі автомати з жорсткою логікою

Керуючим автоматом з жорсткою логікою називають дискретний пристрій, який задають трьома непустими множинами: $X = \{x_1, x_2, \dots, x_n\}$ – алфавіт вхідних сигналів (логічні умови), $Y = \{y_1, y_2, \dots, y_m\}$ – алфавіт вихідних сигналів (сигнали мікрооперацій), $Z = \{z_1, z_2, \dots, z_p\}$ – множина станів автомата, та двома функціями – функцією переходів δ та функцією виходів λ .

Тобто керуючий автомат з жорсткою логікою описують п'ятіркою $\langle X, Y, Z, \delta, \lambda \rangle$.

На множині Z фіксують один зі станів як початковий стан, і його позначають z_1 .

Функція переходів δ

$$z(t_{i+1}) = \delta(z(t_i), X(t_i))$$

визначає стан автомата $z(t_{i+1})$ в наступний момент дискретного часу t_{i+1} залежно від поточного стану автомата $z(t_i)$ та значень вхідних змінних $X(t_i)$ (логічних умов), що мають місце (діють) в даний момент дискретного часу t_i , де $i = 1, 2, \dots$.

Функція виходів λ

$$Y(t_i) = \lambda(z(t_i), X(t_i))$$

визначає значення вихідних сигналів $Y(t_i)$ (сигналів мікрооперацій) в момент дискретного часу t_i залежно від поточного стану автомата $z(t_i)$ та вхідних змінних $X(t_i)$ (логічних умов), що мають місце (діють) в даний момент часу t_i .

Сукупність правил, які описують послідовність перемикання вихідних сигналів Y та станів автомата Z залежно від послідовності вхідних сигналів X , називають *законом функціонування автомата*.

Залежно від способу визначення значень вихідних сигналів Y автомати поділяють на

- автомати Мілі (Mealy): $Y(t_i) = \lambda(z(t_i), X(t_i))$,
- автомати Мура (Moore): $Y(t_i) = \lambda(z(t_i))$.

Тобто функція виходів автомата Мілі (Mealy automaton (machine)) залежить як від поточного стану $z(t_i)$, в якому перебуває автомат, так і від значень вхідних змінних $X(t_i)$, що діють в момент часу t_i , а функція виходів автомата Мура (Moore automaton (machine)) повністю залежить лише від поточного стану автомата $z(t_i)$, і не залежить від значень вхідних змінних $X(t_i)$.

Функція переходів δ для автомата Мілі та автомата Мура визначається однаково:

$$z(t_{i+1}) = \delta(z(t_i), X(t_i)).$$

Таким чином, автомат Мура є окремим випадком автомата Мілі.

Розглянемо автомат Мілі.

Закон функціонування автомата Мілі визначають так:

$$\begin{cases} \text{функція переходів: } z(t_{i+1}) = \delta(z(t_i), X(t_i)) \\ \text{функція виходів: } Y(t_i) = \lambda(z(t_i), X(t_i)). \end{cases}$$

Функцію переходів $\delta(z(t_i), X(t_i))$ та функцію виходів $\lambda(z(t_i), X(t_i))$ автомата Мілі задають *таблицею переходів* та *таблицею виходів* відповідно, наприклад табл. 10.1, 10.2.

Таблиця 10.1
Таблиця переходів
автомата Мілі

λ	z_1	z_2	z_3	z_4
x_1	z_2	z_3	z_1	z_3
x_2	z_4	z_1	z_1	z_1
x_3	z_3	z_4	z_3	z_4

Таблиця 10.2
Таблиця виходів
автомата Мілі

δ	z_1	z_2	z_3	z_4
x_1	y_3	y_2	y_1	y_1
x_2	y_2	y_1	y_3	y_3
x_3	y_1	y_2	y_2	y_3

У таблиці переходів та таблиці виходів автомата Мілі заголовками рядків є логічні умови (вхідні сигнали), а заголовками стовпців – стани автомата. У таблиці переходів на перетині рядка та стовпця вказують новий стан, у який переходить автомат, а в таблиці виходів – вихідний сигнал (сигнал мікрооперації), який генерується при переході в цей стан.

Функція переходів та функція виходів автомата Мілі можуть бути задані суміщеною таблицею (табл. 10.3).

Таблиця 10.3
Суміщена таблиця автомата Мілі

	z_1	z_2	z_3	z_4
x_1	z_2 / y_3	z_3 / y_2	z_1 / y_1	z_3 / y_1
x_2	z_4 / y_2	z_1 / y_1	z_1 / y_3	z_1 / y_3
x_3	z_3 / y_1	z_4 / y_2	z_3 / y_2	z_4 / y_3

Перехід до іншого стану автомата Мілі визначається попереднім станом та діючими в даний момент вхідними сигналами (логічними умовами).

Вихідні сигнали автомата Мілі в даний момент залежать як від стану, в якому в цей момент перебуває автомат, так і від вхідних сигналів, що діють в цей момент.

Розглянемо автомат Мура.

Закон функціонування

автомата Мура визначають так:

$$\begin{cases} \text{функція переходів: } z(t_{i+1}) = \delta(z(t_i), X(t_i)) \\ \text{функція виходів: } Y(t_i) = \lambda(z(t_i)). \end{cases}$$

Перехід до іншого стану автомата Мура визначається попереднім станом та діючими в даний момент входними сигналами (як і в автоматі Мілі).

Вихідні сигнали автомата Мура в даний момент часу залежать лише від стану, в якому автомат перебуває в цей час.

Тому функцію переходів та функцію виходів автомата Мура задають однією позначеною таблицею переходів, наприклад (табл. 10.4).

Таблиця 10.4
Позначена таблиця переходів автомата Мура

λ, δ ↙	z_1	z_2	z_3	z_4
	y_3	y_2	y_2	y_1
x_1	z_3	z_3	z_3	z_1
x_2	z_2	z_4	z_4	z_1

У позначеній таблиці переходів у заголовку кожного стовпця, крім стану автомата z , записують вихідний сигнал y , що відповідає цьому стану.

Більш наочним є задання закону функціонування автомата за допомогою графа.

Наприклад, автомату Мілі, заданому суміщеною таблицею (табл. 10.3), відповідає граф на рис. 10.4. А автомату Мура, заданому позначеною таблицею переходів

(табл. 10.4), відповідає граф на рис. 10.5.

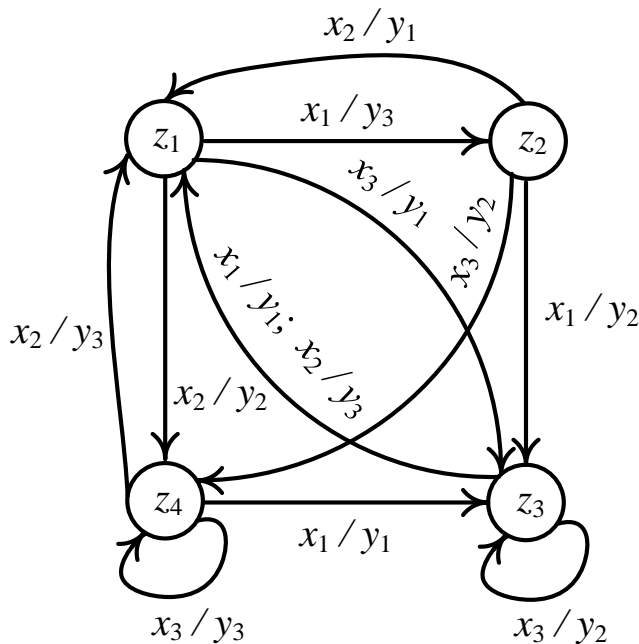


Рис. 10.4. Граф автомата Мілі, заданий суміщеною таблицею автомата (табл. 10.3)

Два автомати, у яких співпадають як входні, так і вихідні алфавіти, називають *еквівалентними*, якщо для будь-якого входного слова вихідне слово одного автомата співпадає з вихідним словом іншого автомата за умови, що автомати перебували в однаковому початковому стані. Під вихідним словом розуміють послідовність вихідних сигналів (сигналів мікрооперацій).

Для будь-якого автомата Мілі можна побудувати еквівалентний автомат Мура, і навпаки.

Отже, закон функціонування керуючого

автомата можна задати трьома способами:

- 1) словесним (вербальним) способом,

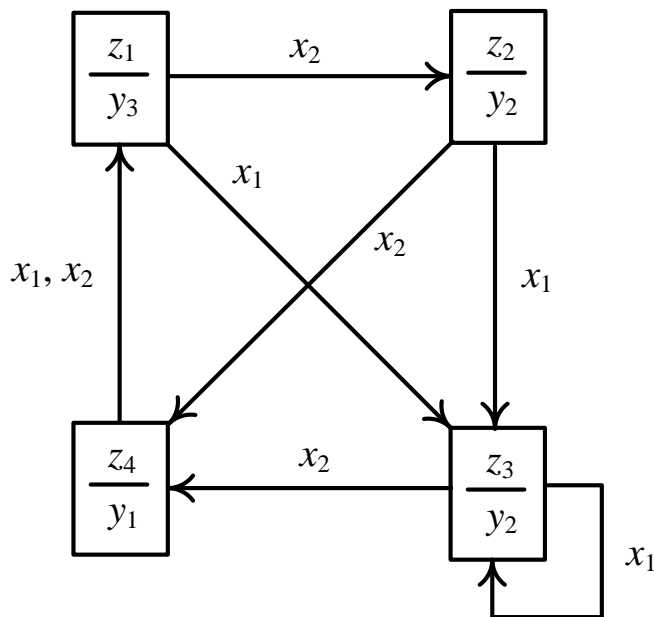


Рис. 10.5. Граф автомата Мура, заданий позначеною таблицею переходів автомата (табл. 10.4)

- 2) таблицею переходів та таблицею виходів автомата (суміщеною таблицею автомата – у випадку автомата Мілі; позначеною таблицею переходів – у випадку автомата Мура),
- 3) за допомогою графа.

Запитання та завдання

1. Як задають керуючий автомат з жорсткою логікою?
2. Дайте визначення функції переходів та функції виходів автомата з жорсткою логікою.
3. Що називають законом функціонування автомата?
4. Запишіть в аналітичному вигляді закон функціонування автомата: а) Мілі; б) Мура. Чим відрізняються між собою автомати Мілі та Мура?
5. Як таблично задають функцію переходів та функцію виходів автомата Мілі? Наведіть приклад.
6. Як таблично задають функцію переходів та функцію виходів автомата Мура?
7. Назвіть способи задання закону функціонування керуючого автомата.
8. Які керуючі автомати називають еквівалентними?

10.3. Канонічний метод структурного синтезу керуючих автоматів з жорсткою логікою

У прикладній теорії цифрових автоматів основними задачами є задача аналізу та задача синтезу автомата.

Під *аналізом автомата* розуміють встановлення закону його функціонування за заданою схемою автомата.

Під *синтезом автомата* розуміють побудову автомата за заданим законом функціонування автомата.

З інженерної точки зору більший інтерес становить задача синтезу керуючих автоматів.

Схему керуючого автомата можна реалізувати за допомогою елементарних автоматів з двома стійкими станами (тригерів) та функціонально повного набору (системи) логічних елементів.

Загальні методи синтезу керуючих автоматів з довільних структурно повних наборів елементарних автоматів та логічних елементів досі не створені.

В.М. Глушковым розроблено загальний конструктивний прийом, який називають *канонічним методом структурного синтезу керуючих автоматів*.

Канонічне подання структури керуючого автомата з жорсткою логікою наведено на рис. 10.6.

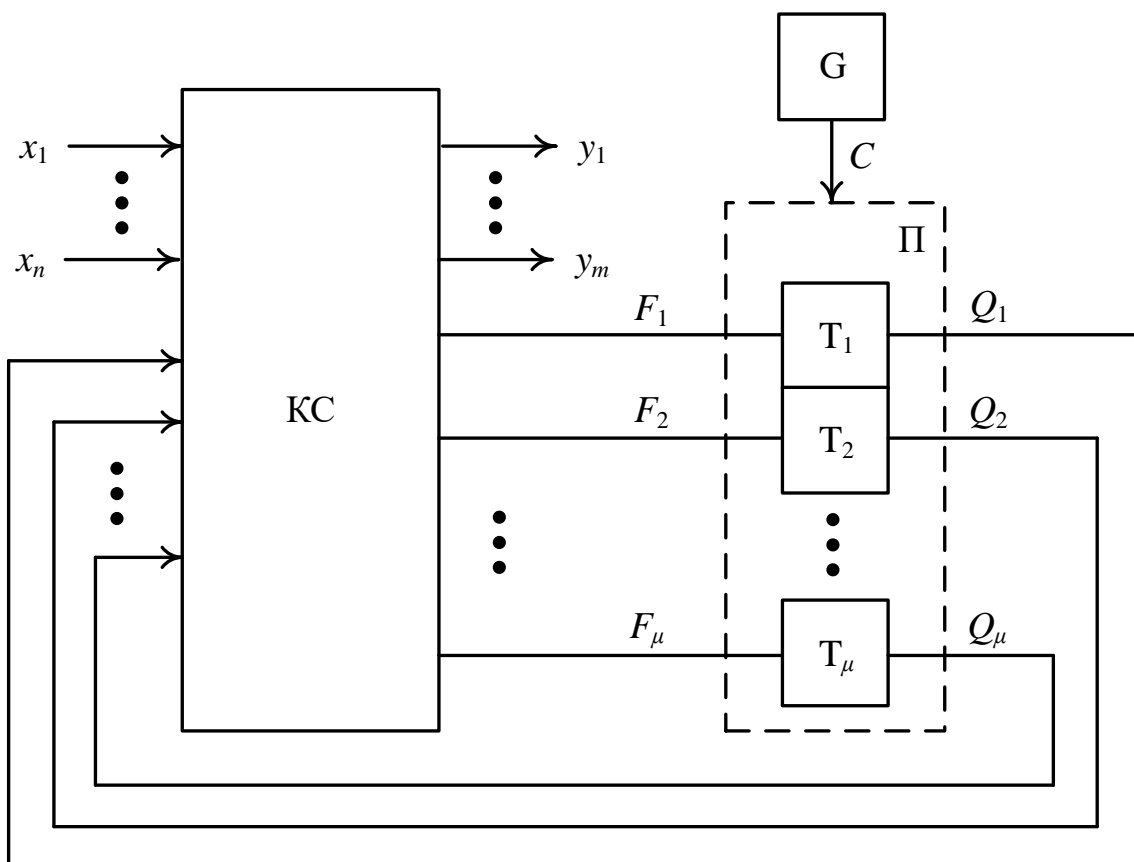


Рис. 10.6. Узагальнена структура керуючого автомата з жорсткою логікою

Автомат містить комбінаційну схему (КС) та пам'ять (П), яка складається з тригерів T_i . Входами КС є логічні умови $X = \{x_1, x_2, \dots, x_n\}$, які формуються операційним автоматом (ОА) операційного пристрою (див. рис. 10.3), та виходи $Q = \{Q_1, Q_2, \dots, Q_\mu\}$ тригерів.

Комбінаційна схема виробляє керуючі сигнали $Y = \{y_1, y_2, \dots, y_m\}$ для операційного автомата операційного пристрою та функції збудження тригерів $\mathcal{F} = \{F_1, F_2, \dots, F_\mu\}$, які визначають перехід автомата з одного стану в інший.

Кожному стану z_i з множини $Z = \{z_1, z_2, \dots, z_p\}$ відповідає певний набір значень $Q = \{Q_1, Q_2, \dots, Q_\mu\}$.

Перехід автомата з одного стану в інший відбувається під дією синхросигналу C , який виробляється генератором G .

Закон функціонування автомата задає порядок перетворення вхідної послідовності $X(t_1), X(t_2), \dots, X(t_i), \dots$ у вихідну послідовність $Y(t_1), Y(t_2), \dots, Y(t_i), \dots$.

Особливість канонічного методу структурного синтезу керуючих автоматів полягає в наступному.

1. Керуючий автомат з жорсткою логікою структурно має складатися з комбінаційної схеми та пам'яті на основі тригерів, яка призначена для запам'ятовування станів автомата (тригери є елементарними автоматами).

2. Задача синтезу автомата зводиться до задачі синтезу комбінаційної схеми шляхом отримання системи перемикальних функцій, яка виражає залежність вихідних сигналів Y та сигналів збудження тригерів \mathcal{F} від вхідних сигналів X та станів автомата Z .

Отримані перемикальні функції мінімізують та апаратно реалізують у вигляді комбінаційної схеми автомата.

Вихідними для задачі синтезу комбінаційної схеми керуючого автомата є:

- 1) структурна схема операційного автомата,
- 2) тип проєктованого автомата з жорсткою логікою (автомат Мілі або автомат Мура),
- 3) опис виконуваної операції (мікроалгоритм операції),
- 4) тип тригерів (елементарних автоматів) – RS-, T-, D-, JK- тощо, з яких має бути утворена пам'ять автомата,
- 5) типи логічних елементів, на основі яких будують комбінаційну схему.

Синтез керуючого автомата з жорсткою логікою включає наступні етапи.

1. Беручи до уваги структурну схему операційного автомата будують граф-схему змістового мікроалгоритму заданої операції.
2. Складають список керуючих сигналів Y , які забезпечують виконання кожної мікрооперації.
3. Визначають тривалість (в тактах) кожного керуючого сигналу.
4. Будують закодовану граф-схему мікроалгоритму операції.
5. На закодованій граф-схемі мікроалгоритму позначають стани проєктованого автомата (Мілі або Мура).
6. Складають граф керуючого автомата.
7. Кодують стани автомата.
8. Складають структурну таблицю автомата.

9. Отримують МДНФ (МКНФ) функцій збудження \mathcal{F} та керуючих сигналів Y .
10. Подають функції збудження тригерів та керуючих сигналів автомата в операторній формі.
11. Будуєть функціональну схему керуючого автомата.

Нехай відома структурна схема операційного автомата (рис. 10.7) та сигнали керування його структурними компонентами.

До складу операційного автомата входять: RG1, RG2 – реверсивні регістри; MUX1, MUX2 – мультиплектори; Σ – комбінаційний суматор; CTR – регістр/лічильник.

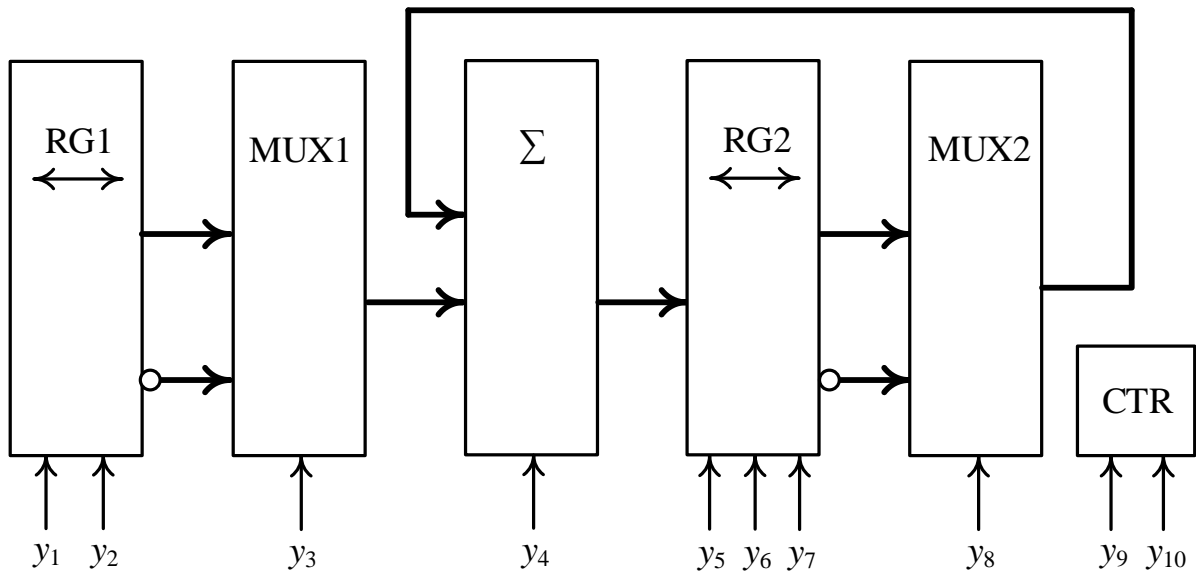


Рис. 10.7. Структурна схема операційного автомата

У структурній схемі операційного автомата використовуються такі сигнали керування його структурними компонентами:

- y_1 – зсув вліво на один розряд регістра RG1,
- y_2 – зсув вправо на один розряд регістра RG1,
- y_3 – керування мультиплексором MUX1 ($y_3 = 0$ – вибір першого входу, $y_3 = 1$ – другого входу),
- y_4 – вхідний перенос суматора ($y_4 = 0 \rightarrow CI = 0$, $y_4 = 1 \rightarrow CI = 1$),
- y_5 – запис слова в регістр RG2,
- y_6 – зсув вліво на один розряд регістра RG2,
- y_7 – зсув вправо на один розряд регістра RG2,
- y_8 – керування мультиплексором MUX2 ($y_8 = 0$ – вибір першого входу, $y_8 = 1$ – другого входу)
- y_9 – інкремент лічильника CTR,
- y_{10} – декремент лічильника CTR.

Розв'яжемо задачу.

Задача 10.1. Синтезувати автомат Мілі для керування виконанням операції $E = 0.5A^2 + 2B$ операційним автоматом, структурна схема якого

подана на рис. 10.7, використовуючи для побудови пам'яті автомата синхронні JK-тригери, а для комбінаційної схеми – логічні елементи 2І, 2АБО.

Розв'язування.

1. Побудуємо граф-схему змістового мікроалгоритму виконання заданої операції.

Розмістимо операнди A та B виконуваної операції в структурних компонентах операційного автомата наступним чином (див. рис. 10.7): $RG1 := A$, $RG2 := B$, $CTR := A$.

Спочатку можна виконати мікрооперацію (МО) зсуву регістра $RG1$ на 1 розряд вправо, внаслідок чого отримаємо $RG1 := 0.5A$. Цю мікрооперацію запишемо так:

$$MO1 : RG1 := R1(RG1),$$

де $R1$ позначає оператор зсуву вправо (Right) на 1 розряд.

Одночасно з $MO1$ можна виконати мікрооперацію зсуву регістра $RG2$ на 1 розряд вліво, внаслідок чого отримаємо $RG2 := 2B$. Цю мікрооперацію запишемо так:

$$MO2 : RG2 := L1(RG2),$$

де $L1$ позначає оператор зсуву вліво (Left) на 1 розряд.

Далі на суматорі Σ можна виконати додавання вмісту регістрів $RG1$ та $RG2$ із запам'ятовуванням результату в $RG2$, тобто таку мікрооперацію:

$$MO3 : RG2 := RG2 + RG1,$$

внаслідок чого в $RG2$ отримаємо величину $0.5A + 2B$. Але при цьому необхідно зменшити на 1 вміст лічильника CTR , тобто виконати декремент CTR :

$$MO4 : CTR := CTR - 1.$$

Додавання $RG2$ та $RG1$, а також декремент CTR необхідно повторити ще $A - 1$ разів.

Дійсно,

$$\underbrace{0.5A + 2B}_{1\text{-й такт}} + \underbrace{0.5A + 0.5A + \dots + 0.5A}_{A-1 \text{ разів}} = 0.5A + 2B = \underbrace{0.5A^2 + 2B}_A \text{ разів}.$$

Тобто додавання $RG2 := RG2 + RG1$ загалом необхідно виконати A разів, а $MO1$ та $MO2$ слід розглядати як встановлення початкового стану $RG1$ та $RG2$ відповідно.

Тому граф-схема змістового мікроалгоритму виконання заданої операції має вигляд як показано на рис. 10.8.

2. Складемо список керуючих сигналів, які забезпечують виконання мікрооперацій $MO1 - MO4$:

$$MO1 - y_2, \quad MO2 - y_6, \quad MO3 - y_3, y_8, y_5, \quad MO4 - y_{10}.$$

3. Визначимо тривалість кожного керуючого сигналу (в тактах). Тривалості керуючих сигналів визначаються з урахуванням затримок в структурних компонентах (вузлах) операційного автомата.

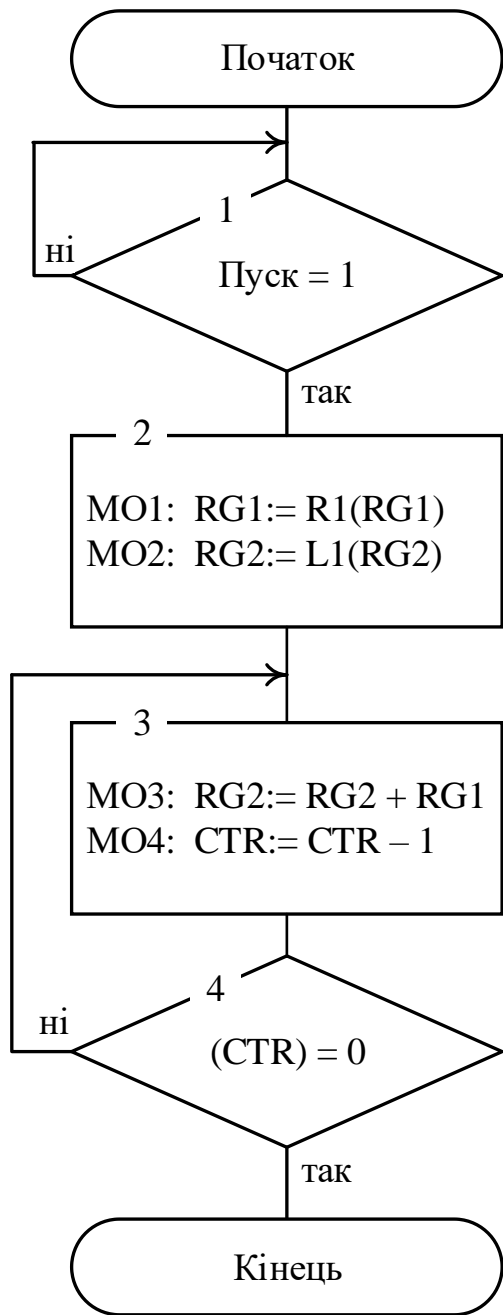


Рис. 10.8. Граф-схема змістового мікроалгоритму виконання операції $E = 0.5A + 2B$

Період T тактового сигналу (синхросигналу C) вибирають рівним мінімальній тривалості керуючих сигналів. При цьому величина T повинна бути не меншою, ніж час переходу автомата з одного стану в інший (час перемикання тригерів).

Вважатимемо, що тривалості керуючих сигналів такі: y_2 (зсув вправо $RG1$), y_6 (зсув вліво $RG2$), y_5 (запис слова в $RG2$), y_{10} (декремент CTR) – t ; y_3 (керування $MUX1$), y_8 (керування $MUX2$) – $2t$.

Складемо таблицю тривалостей керуючих сигналів (табл. 10.5).

4. Побудуємо закодовану граф-схему мікроалгоритму операції.

Складаємо таблицю позначень логічних умов (табл. 10.6).

Опис мікрооперацій замінюємо на відповідні керуючі сигнали.

Вводимо додаткову операторну вершину для керуючих сигналів, що мають тривалість $2t$ (y_3, y_8).

Як наслідок, отримаємо закодовану граф-схему мікроалгоритму операції (рис. 10.9).

5. На закодованій граф-схемі мікроалгоритму позначимо стани автомата Мілі.

Правило позначення станів автомата Мілі

А. Символом z_1 позначаємо вхід вершини (умовної або операторної) наступної за початковою вершиною, а також вхід кінцевої вершини.

Таблиця 10.5

Таблиця тривалостей керуючих сигналів

Мікрооперація	Керуючі сигнали	Тривалості керуючих сигналів
MO1 : $RG1 := R1(RG1)$	y_2	t
MO2 : $RG2 := L1(RG2)$	y_6	t
MO3 : $RG2 := RG2 + RG1$	y_3, y_8, y_5	$2t, 2t, t$
MO4 : $CTR := CTR - 1$	y_{10}	t

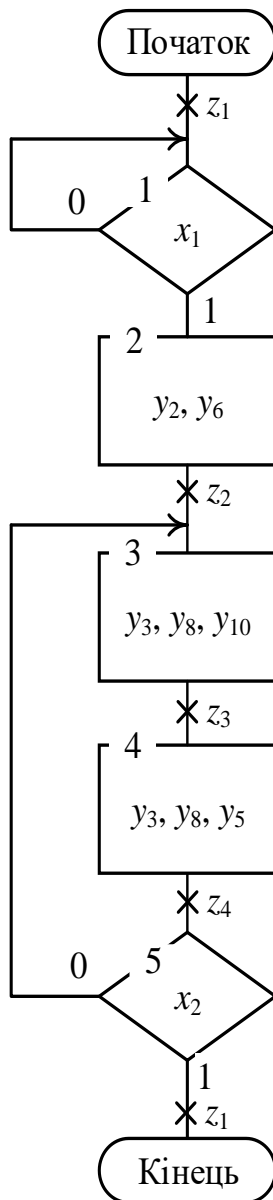


Рис. 10.9. Закодована граф-схема мікроалгоритму виконання операції

Б. Входи всіх вершини, що йдуть за операторними, послідовно позначаємо z_2, z_3, z_4, \dots

В. Входи двох різних вершин, за винятком кінцевої, не можуть бути позначені однаковим символом.

Г. Вхід вершини може позначатись лише одним символом.

Таблиця 10.6

Таблиця позначень логічних умов

Логічні умови	Позначення логічної умови
Пуск = 1	x_1
(CTR) = 0	x_2

б. Складемо граф автомат Мілі (рис. 10.10).

Кількість вершин графа дорівнює кількості станів автомата. Кожному переходу автомата з одного стану в інший відповідає дуга.

На закодованій граф-схемі мікроалгоритму переходу з позначки z_i в позначку z_j відповідає шлях

$$z_i \tilde{x}_\alpha \tilde{x}_\beta \dots \tilde{x}_\omega y_a y_b \dots y_\omega z_j,$$

якому на графі автомата відповідає дуга; $\tilde{x} = x$, якщо шлях проходить через вихід умовної вершини x , позначеної значенням 1; $\tilde{x} = \bar{x}$, якщо шлях проходить через вихід умовної вершини, позначеної значенням 0; $y_a y_b \dots y_\omega$ – перелік сигналів мікрооперацій, зазначений в операторній вершині, через яку проходить даний шлях.

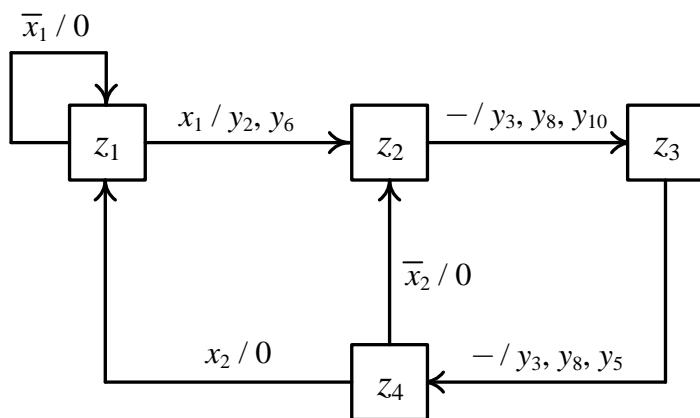


Рис. 10.10. Граф автомата Мілі

Можливі випадки, коли: шлях $z_i y_a y_b \dots y_\omega z_j$ не проходить через умовні вершини, він містить пусту множину логічних умов; шлях $z_i \tilde{x}_\alpha \tilde{x}_\beta \dots \tilde{x}_\omega z_j$ містить пусту множину сигналів мікрооперацій.

7. Закодуємо стани автомата.

Кількість тригерів, що необхідна для організації

пам'яті автомата, визначають зі співвідношення $\mu = \lceil \log_2 p \rceil$, де p – кількість станів автомата. Кожному стану z_i має відповідати визначена комбінація значень Q_μ, \dots, Q_1 .

Оскільки $p = 4$, то $\mu = 2$. Тобто, для пам'яті автомата необхідно використати два JK-тригери (табл. 10.7).

Таблиця 10.7
Таблиця кодування станів автомата Мілі

Стан	Код стану	
	Q_2	Q_1
z_1	0	0
z_2	0	1
z_3	1	1
z_4	1	0

8. Складемо структурну таблицю автомата Мілі (табл. 10.8).

Структурну таблицю автомата складають за графом автомата.

Кожен рядок структурної таблиці автомата відповідає переходу автомата з одного стану в інший – з поточного стану (ПС) у наступний стан (НС) з урахуванням значень логічних умов. Під час такого переходу формуються відповідні керуючі сигнали, а також функції збудження JK-тригерів (табл. 10.9).

Таблиця 10.8

Структурна таблиця автомата Мілі

ПС	Код ПС		НС	Код НС		Логічні умови		Керуючі сигнали					Функції збудження тригерів				
	$Q_2(t_i)$	$Q_1(t_i)$		$Q_2(t_{i+1})$	$Q_1(t_{i+1})$	x_2	x_1	y_2	y_6	y_3	y_8	y_{10}	y_5	F_{J_2}	F_{K_2}	F_{J_1}	F_{K_1}
z_1	0	0	z_1	0	0	x	0	0	0	x	x	0	0	0	x	0	x
z_1	0	0	z_2	0	1	x	1	1	1	x	x	0	0	0	x	1	x
z_2	0	1	z_3	1	1	x	x	0	0	0	0	1	x	1	x	x	0
z_3	1	1	z_4	1	0	x	x	0	0	0	0	0	1	x	0	x	1
z_4	1	0	z_2	0	1	0	x	0	0	x	x	0	0	x	1	1	x
z_4	1	0	z_1	0	0	1	x	x	0	x	x	x	0	x	1	0	x

Примітка. ПС – поточний стан, НС – наступний стан.

Таблиця 10.9
Таблиця функцій
збудження JK-тригера

$Q(t_i)$	$Q(t_{i+1})$	F_J	F_K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

У кожному рядку структурної таблиці автомата для j -го тригера ($j = 1, 2$) розглядають переходи $Q_j(t_i) \rightarrow Q_j(t_{i+1})$. Довільні значення керуючих сигналів (0 або 1) у структурній таблиці автомата позначають символом \times .

9. Отримаємо МДНФ керуючих сигналів $y_2, y_6, y_3, y_8, y_{10}, y_5$ та функцій збудження тригерів $F_{J_2}, F_{K_2}, F_{J_1}, F_{K_1}$ за допомогою діаграм Вейча (рис. 10.11).

Зі структурної таблиці автомата бачимо, що $y_3 = 0, y_8 = 0$.

10. Подамо МДНФ керуючих сигналів автомата та функцій збудження тригерів в операторній формі, беручи до уваги типи заданих логічних елементів (2І, 2АБО):

$$\begin{array}{l}
 y_2 = \bar{Q}_2 \bar{Q}_1 x_1 = (\bar{Q}_2 \bar{Q}_1) x_1 \\
 y_3 = 0 \\
 y_5 = Q_1 \\
 y_6 = \bar{Q}_2 \bar{Q}_1 x_1 = (\bar{Q}_2 \bar{Q}_1) x_1 = y_2 \\
 y_8 = 0 \\
 y_{10} = \bar{Q}_2 Q_1
 \end{array}
 \left|
 \begin{array}{l}
 F_{J_2} = Q_1 \\
 F_{K_2} = \bar{Q}_1 \\
 F_{J_1} = \bar{Q}_2 x_1 \vee Q_2 \bar{x}_2 \\
 F_{K_1} = Q_2
 \end{array}
 \right.$$

11. Побудуємо функціональну схему керуючого автомата (рис. 10.12). ■

Задача 10.2. Синтезувати автомат Мура для керування виконанням операції $E = 0.5A^2 + 2B$ операційним автоматом, структурна схема якого подана на рис. 10.7, використовуючи для побудови пам'яті автомата синхронні Т-тригери, а для комбінаційної схеми – логічні елементи АБО-НЕ.

Розв'язування

Кроки 1 – 4 розв'язання цієї задачі співпадають з відповідними кроками попередньої задачі.

Результатом 4-го кроку розв'язання є закодована граф-схема мікроалгоритму виконання операції (рис. 10.13).

5. На закодованій граф-схемі мікроалгоритму позначимо стани автомата Мура (див. рис. 10.13).

Правило позначення станів автомата Мура

А. Символом z_1 позначаємо початкову та кінцеву вершини.

Б. Всі операторні вершини послідовно позначаємо z_2, z_3, \dots .

В. Дві різні операторні вершини не можуть бути позначені однаковим символом z .

6. Складемо граф автомата Мура (рис. 10.14).

Кількість вершин графа дорівнює кількості станів автомата. Кожному переходу автомата з одного стану в інший відповідає дуга.

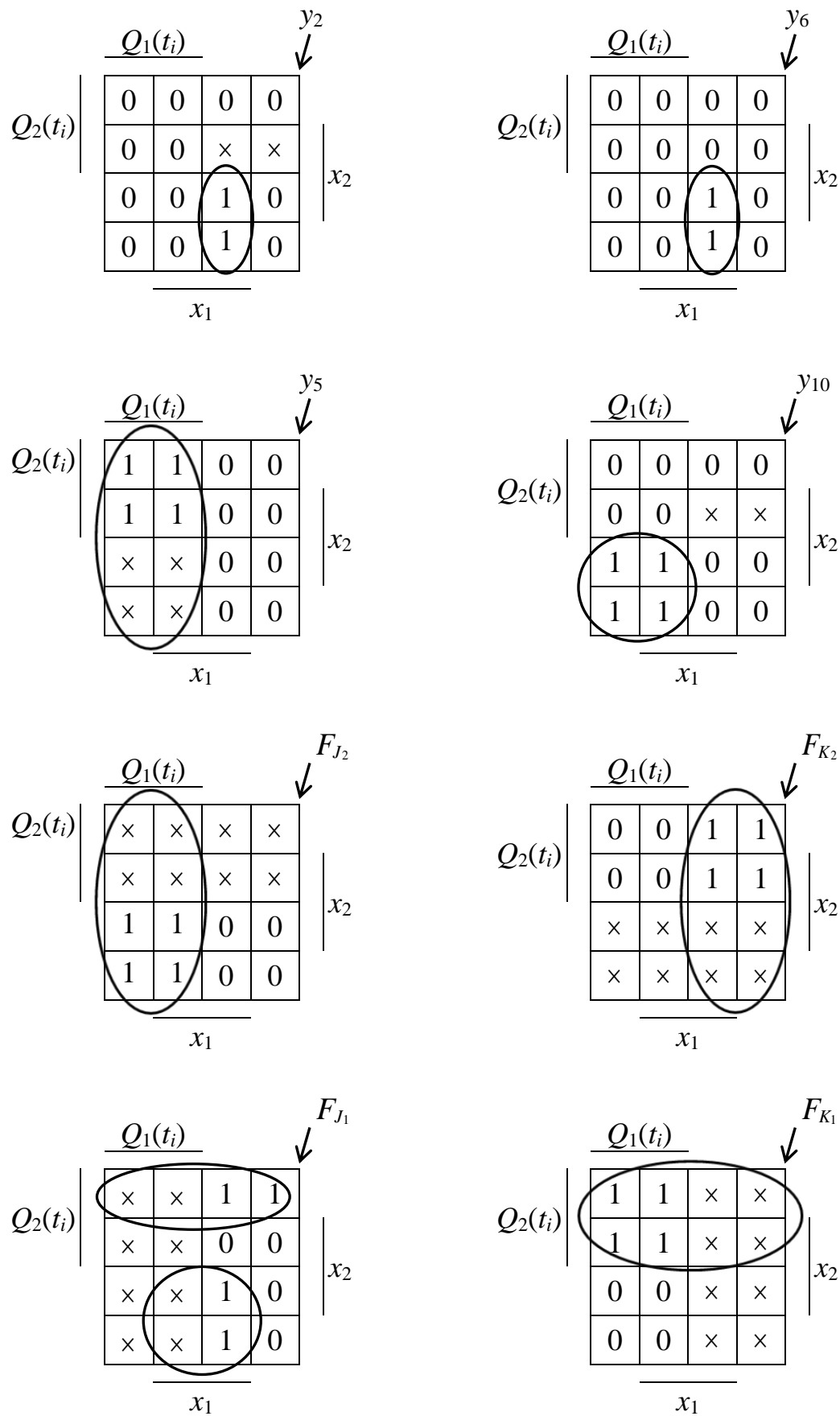


Рис. 10.11. Діаграми Вейча керуючих сигналів y_2 , y_6 , y_5 , y_{10} та функцій збудження JK-тригерів автомата Мілі

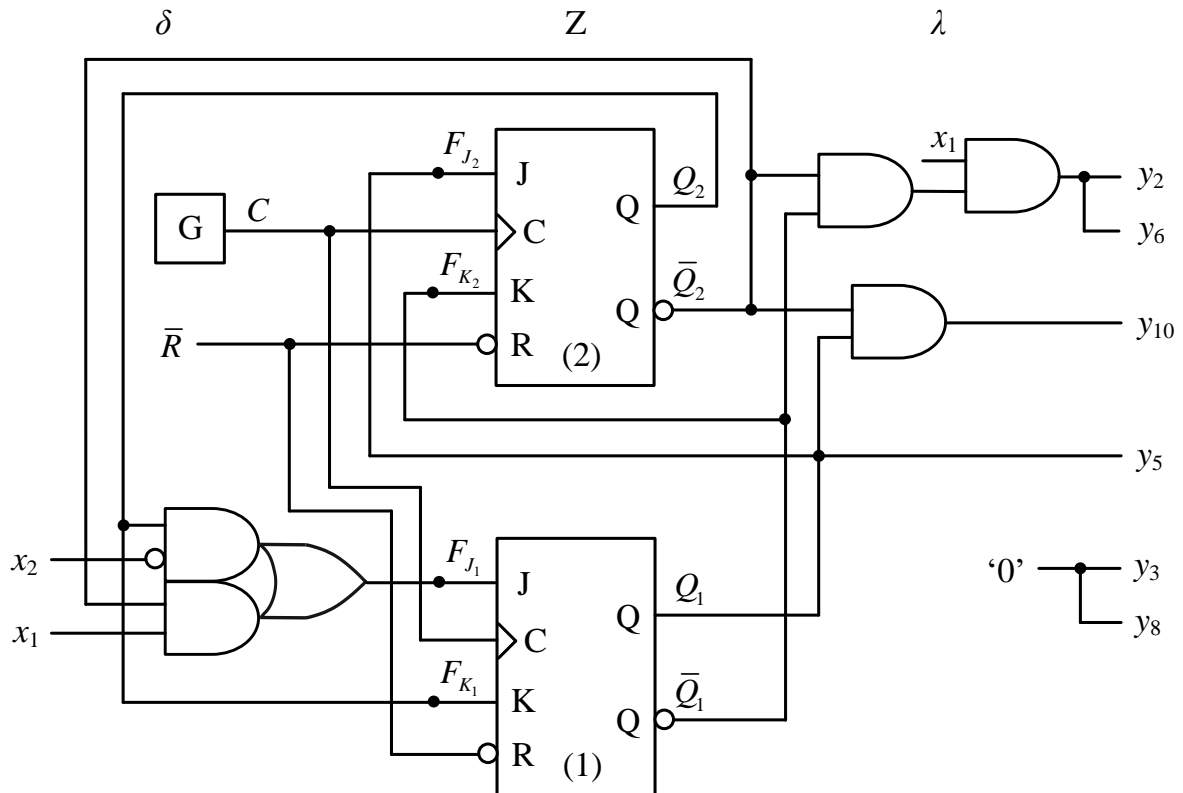


Рис. 10.12. Функціональна схема керуючого автомата Мілі

Примітка. \bar{R} – встановлення початкового стану автомата;
 G – генератор синхросигналів

На графі автомата Мура дугам приписують набори логічних умов, які забезпечують відповідний перехід.

Керуючі сигнали записують у вершинах графа, тому що вони не залежать від логічних умов.

На закодованій граф-схемі мікроалгоритму виконання операції переходу з позначки z_i в позначку z_j відповідає шлях

$$z_i \tilde{x}_\alpha \tilde{x}_\beta \dots \tilde{x}_\omega z_j,$$

якому на графі автомата відповідає дуга; $\tilde{x} = x$, якщо шлях проходить через вихід умовної вершини x , позначеної значенням 1; $\tilde{x} = \bar{x}$, якщо шлях проходить через вихід умовної вершини, позначеної значенням 0.

Можливий випадок, коли шлях $z_i z_j$ не проходить через умовні вершини, він містить пусту множину логічних умов.

Коли автомат не функціонує (мікроалгоритм не виконується), він перебуває в початковому стані z_1 .

7. Закодуємо стани автомата.

Оскільки автомат може перебувати в чотирьох станах ($z_1 - z_4$), тобто $p = 4$, то для пам'яті автомата слід використати два тригери (табл. 10.10).

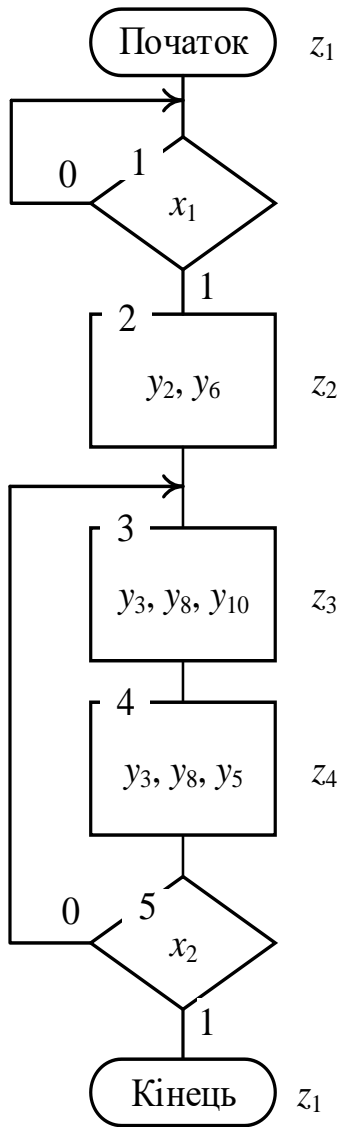


Рис. 10.13. Закодована граф-схема мікроалгоритму виконання операції

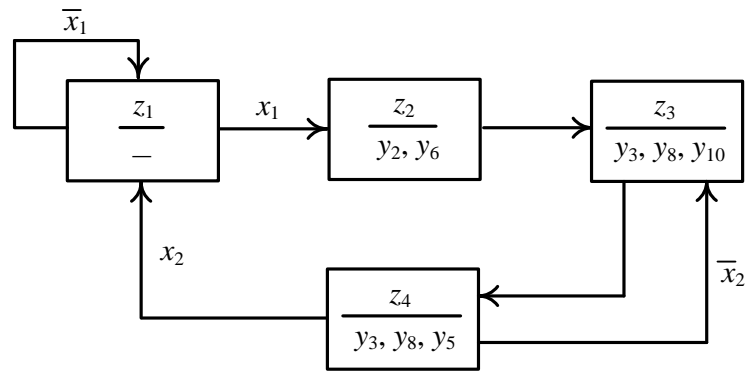


Рис. 10.14. Граф автомата Мура

Таблиця 10.10
Таблиця кодування станів автомата Мура

Стан	Код стану	
	Q_2	Q_1
z_1	0	0
z_2	0	1
z_3	1	1
z_4	1	0

8. Складемо структурну таблицю автомата Мура (табл. 10.11).

Керуючі сигнали автомата Мура залежать лише від стану, в якому перебуває автомат, тобто лише від $Q_2(t_i)$ та $Q_1(t_i)$.

Переходи автомата з одного стану в інший – з поточного стану (ПС) у наступний стан (НС), ініціюються функціями збудження F_{T_2} , F_{T_1} тригерів, які залежать як від ПС, так і від логічних умов.

Таблиця 10.11

Структурна таблиця автомата Мура

ПС	Код ПС		НС	Код НС		Логічні умови		Керуючі сигнали						Функції збудження тригерів	
	$Q_2(t_i)$	$Q_1(t_i)$		$Q_2(t_{i+1})$	$Q_1(t_{i+1})$	x_2	x_1	y_2	y_6	y_3	y_8	y_{10}	y_5	F_{T_2}	F_{T_1}
z_1	0	0	z_1	0	0	×	0	0	0	×	×	0	0	0	0
z_1	0	0	z_2	0	1	×	1	0	0	×	×	0	0	0	1
z_2	0	1	z_3	1	1	×	×	1	1	×	×	0	0	1	0
z_3	1	1	z_4	1	0	×	×	0	0	1	1	1	0	0	1
z_4	1	0	z_1	0	0	1	×	0	0	1	1	0	1	1	0
z_4	1	0	z_3	1	1	0	×	0	0	1	1	0	1	0	1

Примітка. ПС – поточний стан, НС – наступний стан.

Стовпці F_{T_2} , F_{T_1} структурної таблиці автомата заповнюємо порядково, аналізуючи переходи $Q_j(t_i) \rightarrow Q_j(t_{i+1})$ ($j = 2, 1$) з урахуванням таблиці функції збудження Т-тригера (табл. 10.12).

Таблиця 10.12
Таблиця функції
збудження Т-тригера

$Q(t_i)$	$Q(t_{i+1})$	F_T
0	0	0
0	1	1
1	0	1
1	1	0

9. Виконаємо мінімізацію функцій керуючих сигналів та функцій збудження F_{T_2} , F_{T_1} тригерів за допомогою діаграм Вейча (рис. 10.15). Оскільки за умовою задачі функції F_{T_2} , F_{T_1} слід подати в формі АБО-НЕ/АБО-НЕ, на діаграмах Вейча необхідно склеювати нулі, тобто визначати МКНФ функцій.

10. Мінімізовані функції подамо в формі АБО-НЕ/АБО-НЕ:

$$y_2 = \bar{Q}_2 \cdot Q_1 = \overline{Q_2 \vee \bar{Q}_1},$$

$$y_6 = \bar{Q}_2 \cdot Q_1 = \overline{Q_2 \vee \bar{Q}_1} = y_2,$$

$$y_3 = 1, y_8 = 1 = y_3,$$

$$y_{10} = Q_2 \cdot Q_1 = \overline{\bar{Q}_2 \vee \bar{Q}_1},$$

$$y_5 = Q_2 \cdot \bar{Q}_1 = \overline{\bar{Q}_2 \vee Q_1},$$

$$F_{T_2} = (\bar{Q}_2 \vee \bar{Q}_1)(\bar{Q}_2 \vee x_2)(Q_2 \vee Q_1) = \overline{\bar{Q}_2 \vee \bar{Q}_1 \vee \bar{Q}_2 \vee x_2 \vee Q_2 \vee Q_1},$$

$$F_{T_1} = (Q_2 \vee \bar{Q}_1)(Q_2 \vee x_1)(\bar{Q}_2 \vee Q_1 \vee \bar{x}_2) = \overline{Q_2 \vee \bar{Q}_1 \vee Q_2 \vee x_1 \vee \bar{Q}_2 \vee Q_1 \vee \bar{x}_2}.$$

Функції F_{T_2} , F_{T_1} можна спростити:

$$F_{T_2} = \overline{y_{10} \vee \bar{Q}_2 \vee x_2 \vee Q_2 \vee Q_1},$$

$$F_{T_1} = \overline{y_2 \vee \bar{Q}_2 \vee x_1 \vee \bar{Q}_2 \vee Q_1 \vee \bar{x}_2}.$$

11. Побудуємо функціональну схему автомата Мура (рис. 10.16). ▣

Беручи до уваги структурні особливості функціональних схем автоматів на рис. 10.12, 10.16 доходимо висновку, що узагальнену структуру керуючого автомата з жорсткою логікою (рис. 10.6) можна деталізувати на випадок автомата Мура та автомата Мілі (рис. 10.17).

Порівнюючи автомати Мілі та Мура можна констатувати наступне.

В автоматі Мілі керуючі сигнали Y виробляються в момент переходу автомата з одного стану в інший. В автоматі Мура керуючі сигнали Y виробляються поки автомат перебуває в даному стані.

У загальному випадку автомат Мура має більшу кількість станів, ніж автомат Мілі, який реалізує той самий мікроалгоритм виконання операції. Різниця в кількості станів впливає на кількість обладнання в автоматі.

Використання автомата Мілі може в окремих випадках приводити до економії обладнання порівняно з еквівалентним йому автоматом Мура.

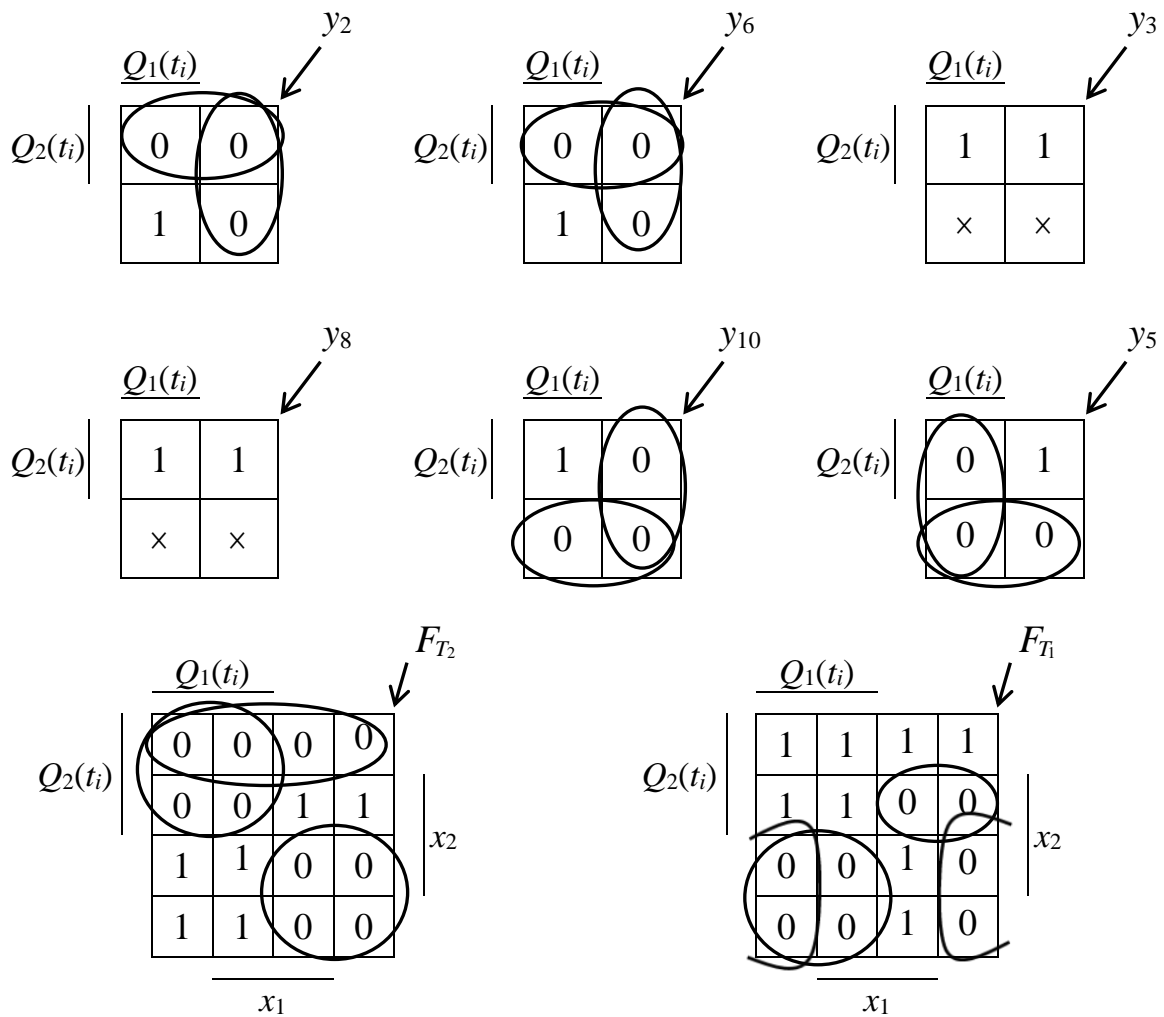


Рис. 10.15. Діаграми Вейча функцій керуючих сигналів та функцій збудження Т-тригерів автомата Мура

Якщо мікроалгоритми операцій (команд) реалізують на основі керуючих автоматів (КА) з жорсткою логікою (автоматів Мілі або Мура), то до складу процесора має входити множина операційних пристроїв $\{OPr_1, \dots, OPr_N\}$ (рис. 10.18), кожен з яких складається з операційного та керуючого автомата.

Якщо $U = \{u_1, \dots, u_h, \dots, u_q, \dots, u_v\}$ – система команд процесора, то на кожен з операційних пристроїв покладається реалізація певної кількості команд з системи команд U .

Під час виконання поточної команди програми відбувається дешифрування (DC) коду операції (КОП) команди та ініціювання відповідного операційного пристрою, до якого надходять операнди команди. Вибраний операційний пристрій забезпечує реалізацію мікроалгоритму, що відповідає даній команді (операції).

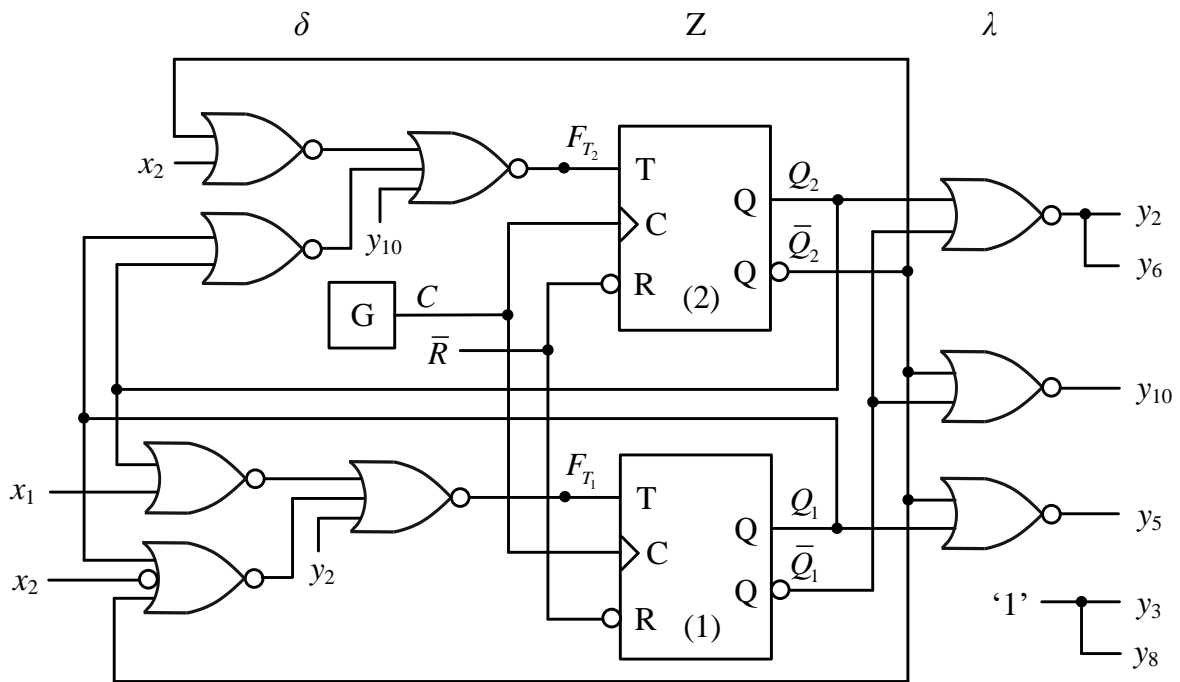


Рис. 10.16. Функціональна схема керуючого автомата Мура

Примітка. \bar{R} – встановлення початкового стану автомата;
 G – генератор синхросигналів

Запитання та завдання

1. Які дві основні задачі вирішують в прикладній теорії цифрових автоматів?
2. Які структурні компоненти входять до складу керуючого автомата з жорсткою логікою?
3. Охарактеризуйте особливість канонічного методу структурного синтезу керуючого автомата з жорсткою логікою, запропонованого В.М. Глушковым.
4. Перелічіть етапи синтезу керуючого автомата з жорсткою логікою.
5. Сформулюйте правило позначення станів автомата Мілі на закодованій граф-схемі мікроалгоритму виконання операції.
6. Сформулюйте правило позначення станів автомата Мура на закодованій граф-схемі мікроалгоритму виконання операції.
7. Виконайте порівняння автоматів Мілі та Мура за способом формування керуючих сигналів та складністю схеми.
8. Охарактеризуйте апаратний спосіб реалізації системи команд процесора.

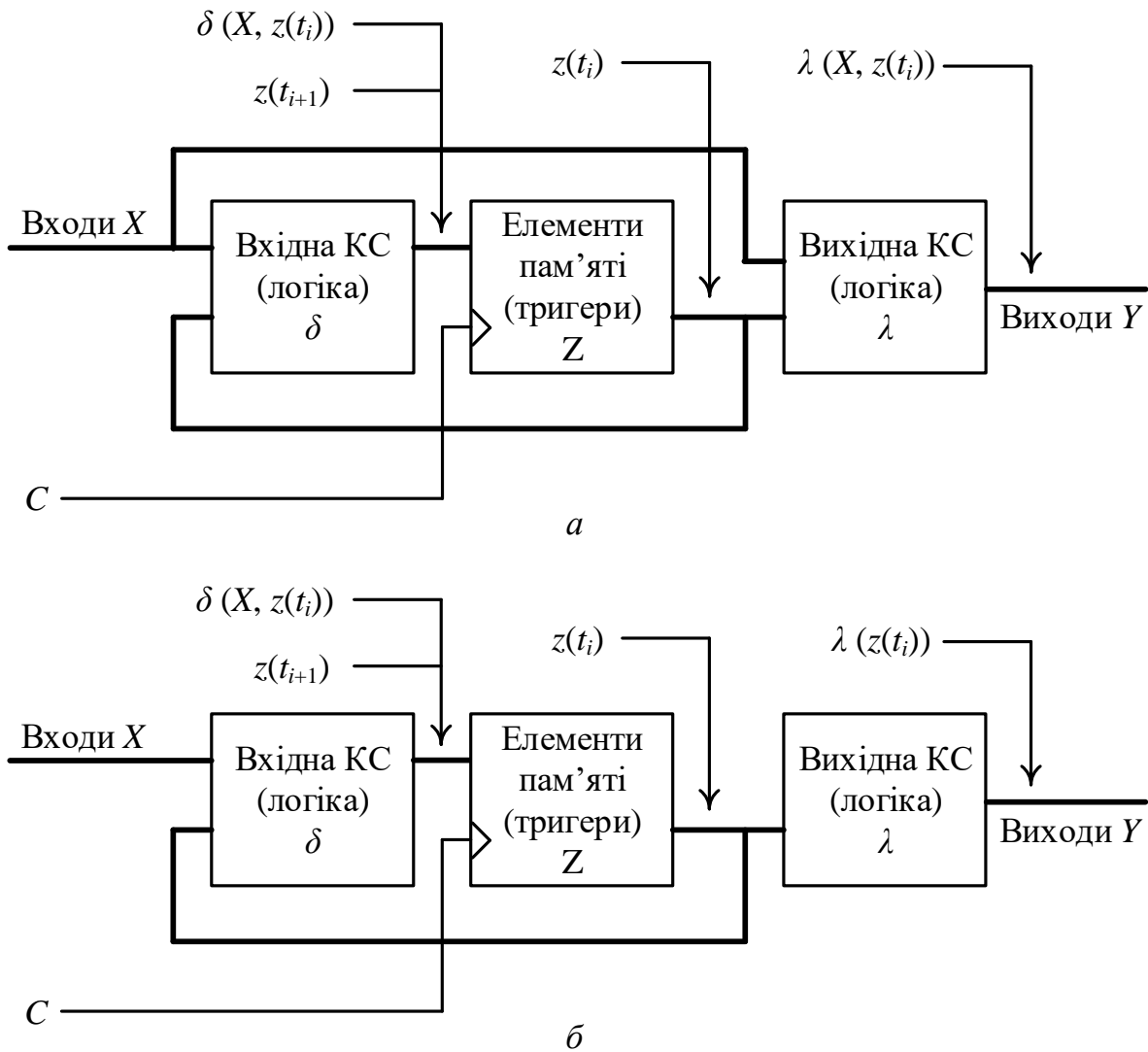


Рис. 10.17. Узагальнена структура автомата: *a* – Мілі; *б* – Мура

Задачі для самостійного розв'язування

1. Синтезувати автомат Мілі для керування виконанням операції $E = 0.5A(B + 1) + 2C$ операційним автоматом на рис. 10.7. Пам'ять автомата побудувати на основі D-тригерів, а комбінаційну схему автомата – на основі логічних елементів І-НЕ.
2. Синтезувати автомат Мура на основі RS-тригерів та логічних елементів АБО-НЕ для керування виконанням операції $E = (A + 1)C + 0.5(B + C)$ операційним автоматом на рис. 10.7.
3. Синтезувати автомат Мілі для керування виконанням операції $E = 4A(B - 2) + 0.5C$ операційним автоматом на рис. 10.7. Для побудови пам'яті автомата використати Т-тригери, а для комбінаційної схеми – елементи АБО-НЕ.

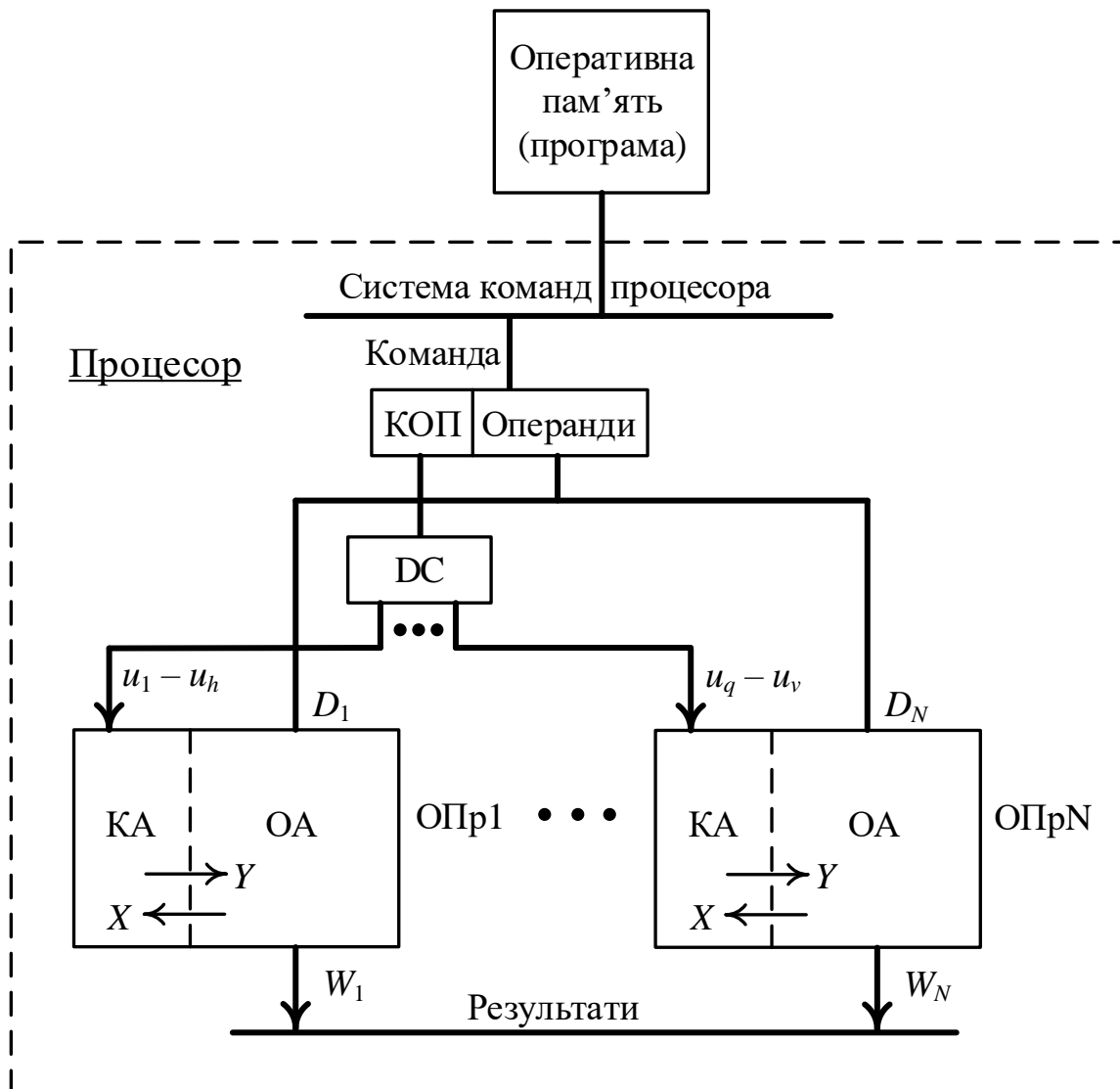


Рис. 10.18. Апаратний спосіб реалізації системи команд процесора

4. Синтезувати автомат Мура для керування виконанням операції $E = 2AC + 0.25(B + C + 1)$ операційним автоматом на рис. 10.7. Пам'ять автомата побудувати на основі JK-тригерів, а комбінаційну схему – на основі логічних елементів 2І-НЕ.

10.4. Керуючі автомати з програмовною логікою

Функціонування керуючого автомата (КА) в структурі операційного пристрою (див. рис. 10.3) визначається множиною вхідних сигналів (логічних умов) $X = \{x_1, x_2, \dots, x_n\}$, які формуються вузлами операційного автомата (ОА), та мікроалгоритмом виконання операції (команди), який задає порядок вироблення керуючих сигналів Y залежно від значень сигналів X .

Керуючий автомат можна синтезувати не лише як структуру з жорсткою логікою (на основі тригерів та логічних елементів), а й як

структуру з мікропрограмним способом реалізації мікроалгоритмів операцій (команд).

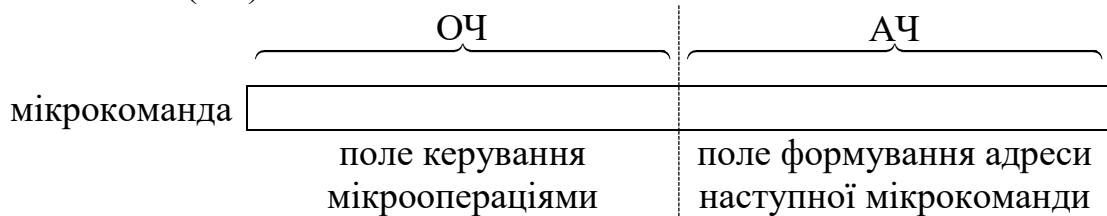
У цьому випадку мікроалгоритм операції (команди) подається у вигляді упорядкованої сукупності керуючих слів (мікрокоманд).

Керуюче слово визначає порядок функціонування операційного автомата протягом одного такту і називається *мікрокомандою* (МК). Сукупність мікрокоманд утворює масив мікрокоманд МК[0:Ω], окремі мікрокоманди в якому вибирають за допомогою адреси, яка є порядковим номером 0, 1, 2, ..., Ω елемента масиву мікрокоманд.

Мікрокоманда містить відомості про мікрооперації, які мають виконуватись у даному такті роботи операційного автомата, та відомості про адресу наступної мікрокоманди.

Масив мікрокоманд зберігається в пам'яті мікрокоманд (ПМК), яка є постійним запам'ятовувальним пристроєм (ПЗП).

Мікрокоманда структурно складається з операційної частини (ОЧ) та адресної частини (АЧ):



Нехай множина Y містить m сигналів мікрооперацій. До множини Y додамо ще один керуючий сигнал y_{m+1} – сигнал зупинення керуючого автомата.

Сигналам мікрооперацій $y_1, y_2, \dots, y_m, y_{m+1}$ присвоїмо номери 1, 2, ..., $m, m+1$, які кодуватимемо g -розрядними двійковими числами, де $g = \lceil \log_2(m+1) \rceil$. Нехай операційна частина мікрокоманди містить f полів Y_1, Y_2, \dots, Y_f , кожне з яких задає номер мікрооперації (рис. 10.19), тобто вважатимемо, що в кожному такті виконується не більше f мікрооперацій.

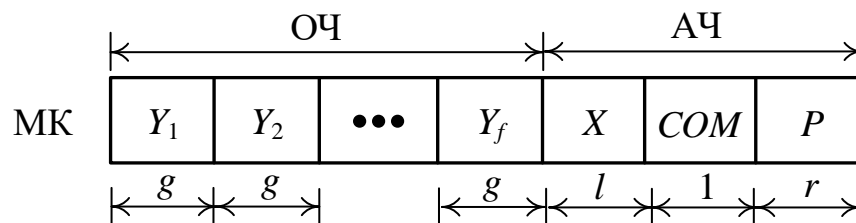


Рис. 10.19. Структура мікрокоманди

Якщо $Y_i = 0$, то поле Y_i не збуджує жодної мікрооперації. Якщо $Y_i = q \neq 0$, то збуджується мікрооперація y_q .

Таким чином, операційна частина мікрокоманди в кожному такті може збуджувати 1, 2, ..., f мікрооперацій або не збуджувати жодної.

Поле X мікрокоманди містить номер логічної умови, яка аналізується

мікрокомандою. У кожній мікрокоманді задається номер лише однієї умови. Якщо $X = 0$, то в даній мікрокоманді жодна логічна умова не аналізується.

Отже, поле X може містити один з кодів $0, 1, 2, \dots, n$, довжина поля X становить $l = \lceil \log_2(n+1) \rceil$ розрядів. Якщо $X = k \neq 0$, то в даній мікрокоманді аналізується логічна умова x_k .

Вважатимемо, що мікрокоманди розміщуються в ПМК починаючи з нульової адреси. Адресацію мікрокоманд в ПМК забезпечує лічильник мікрокоманд (ЛМК). Під час виконання мікрокоманди з адресою j ЛМК збільшує свій стан на одиницю, тобто виробляє адресу $j + 1$.

Адреса $Addr$ наступної мікрокоманди в ПМК формується так:

$$Addr := \begin{cases} (\text{ЛМК}), & \text{якщо } x_k \oplus \text{COM} = 0, \\ P, & \text{якщо } x_k \oplus \text{COM} = 1, \end{cases}$$

де P – адреса переходу. Додавання значення логічної умови x_j та поля COM (поля інвертування) здійснює суматор за модулем два (рис. 10.20).

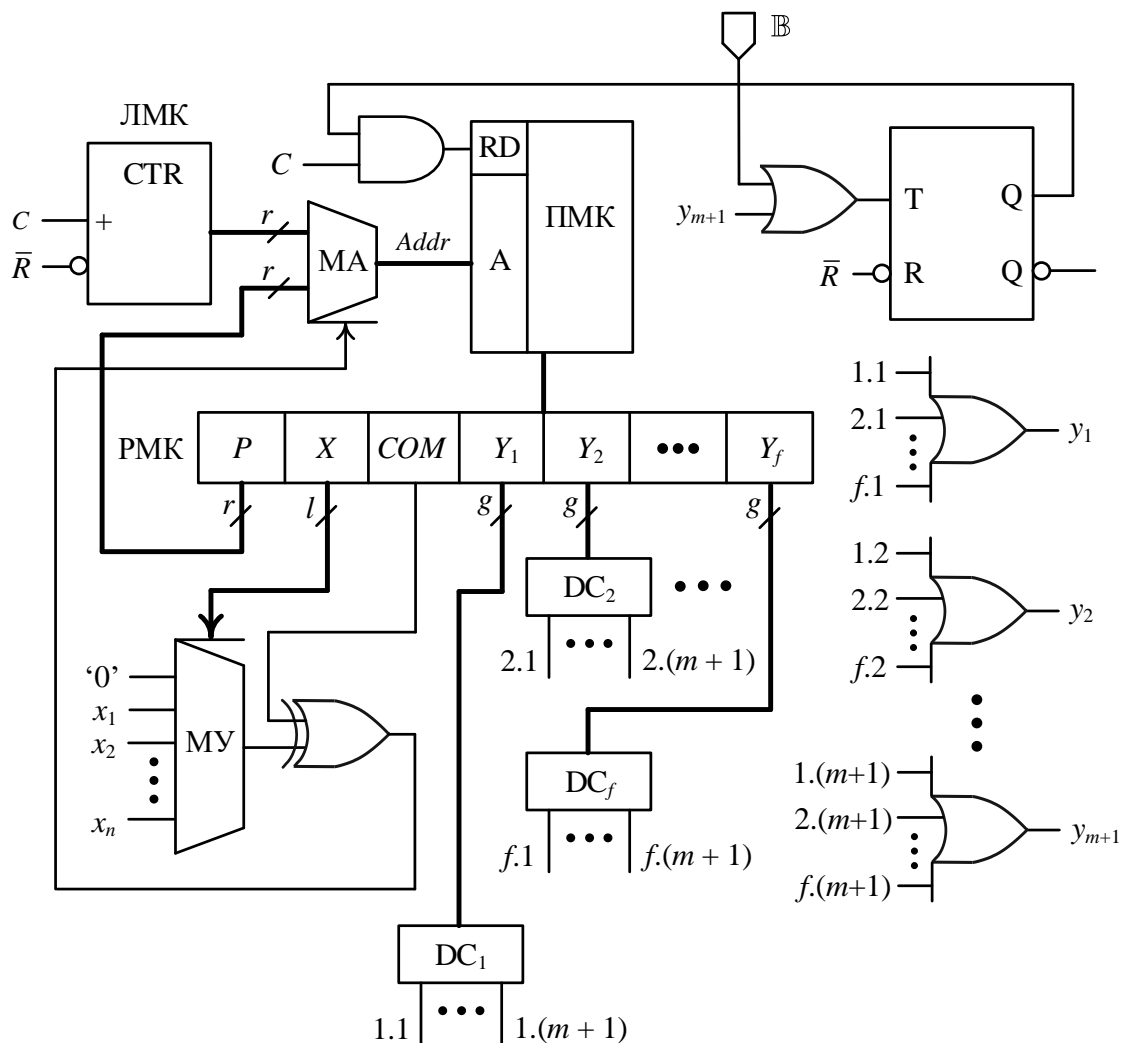


Рис. 10.20. Узагальнена структура керуючого автомата з програмовною логікою

Значення, що знаходиться в полі X мікрокоманди, є кодом керування мультиплексором умов (МУ), який пропускає на вихід значення x_k .

Значення $x_k \oplus \text{COM}$ використовується для керування мультиплексором адрес (МА). Якщо $x_k \oplus \text{COM} = 0$, то на вхід шини адреси A пам'яті мікрокоманд надходить значення (адреса) з виходу ЛМК.

Якщо $x_k \oplus \text{COM} = 1$, то на шину адреси A надходить значення (адреса) з поля P регістра мікрокоманди.

Для приведення ПМК в робочий режим використовують сигнал $B = 1$ (begin), який встановлює Т-тригер в "1". Це забезпечує зчитування мікрокоманд з ПМК. Першою зчитується мікрокоманда з адресою $0 \dots 0$.

Після кожного сигналу RD (read) в регістр мікрокоманди (РМК) приймається чергова мікрокоманда.

Мікрокоманда обробляється наступним чином. Кожне з полів Y_i дешифрується відповідним дешифратором (DC). Керуючі сигнали u_1, u_2, \dots, u_{m+1} формуються як диз'юнкції однойменних виходів дешифраторів DC_1, \dots, DC_f та надходять до операційного автомата, збуджуючи в ньому виконання відповідних мікрооперацій. Після виконання операційним автоматом заданих операцій та формування відповідної логічної умови x_k відбувається генерування адреси наступної мікрокоманди, яка зчитується з ПМК.

Якщо в поточній мікрокоманді буде сформовано сигнал $u_{m+1} = 1$ – зупинення керуючого автомата, то Т-тригер буде встановлено в "0", сигнал читання RD з ПМК буде заблоковано.

Керуючий автомат, який реалізує мікроалгоритм операції (команди) за допомогою мікропрограми, яка зберігається в ПЗП, називають *керуючим автоматом з програмовною логікою*, або *П-автоматом*.

Розглянемо методику синтезу керуючого автомата з програмовною логікою, тобто П-автомата.

Вихідними для задачі синтезу П-автомата є структурна схема операційного автомата та мікроалгоритм операції, яку необхідно виконати.

Синтез П-автомата включає наступні етапи.

1. Складають список керуючих сигналів Y операційного автомата, які забезпечують виконання кожної мікрооперації, та кодують їх.
2. Будують граф-схему змістового мікроалгоритму заданої операції. Складають список керуючих сигналів, які забезпечують виконання заданої операції.
3. Визначають тривалості (в тактах) кожного керуючого сигналу.
4. Кодують логічні умови та будують закодовану граф-схему мікроалгоритму.
5. Визначають структуру мікрокоманди.
6. Будують схему керуючого автомата.
7. Складають мікропрограму та розміщують її в ПМК.

Розв'яжемо задачу.

Задача 10.3. Синтезувати П-автомат для керування виконанням операції $E = 4A(B + 1) - 0.5C$ операційним автоматом, структурна схема якого подана на рис. 10.7.

Розв'язування.

1. Складемо список керуючих сигналів, необхідних для функціонування операційного автомата, та закодуємо їх (табл. 10.12).

2. Побудуємо граф-схему змістового мікроалгоритму виконання заданої операції.

Розмістимо операнди A , B , C в структурних компонентах операційного автомата так (див. рис. 10.7): $RG1 := A$, $RG2 := C$, $CTR := B$.

Таблиця 10.12

Список керуючих сигналів операційного автомата та їх кодування

Сигнал мікрооперації	Код мікрооперації	Зміст мікрооперації
y_1	0 0 0 1	Зсув вліво регістра RG1
y_2	0 0 1 0	Зсув вправо регістра RG1
y_3	0 0 1 1	Керування мультиплексором MUX1 ($y_3 = 0$ – вибір першого входу, $y_3 = 1$ – вибір другого входу)
y_4	0 1 0 0	Вхідний перенос суматора ($y_4 = 0 \rightarrow CI = 0$, $y_4 = 1 \rightarrow CI = 1$)
y_5	0 1 0 1	Запис слова в регістр RG2
y_6	0 1 1 0	Зсув вліво регістра RG2
y_7	0 1 1 1	Зсув вправо регістра RG2
y_8	1 0 0 0	Керування мультиплексором MUX2 ($y_8 = 0$ – вибір першого входу, $y_8 = 1$ – другого входу)
y_9	1 0 0 1	Інкремент лічильника CTR
y_{10}	1 0 1 0	Декремент лічильника CTR
y_{11}	1 0 1 1	Зупинення керуючого автомата

Спочатку виконаємо мікрооперацію зсуву регістра RG1 на 1 розряд вліво, внаслідок чого отримаємо $RG1 := 2A$

$$MO1 : RG1 := L1(RG1).$$

Далі знову виконаємо мікрооперацію зсуву регістра RG1 на 1 розряд вліво, внаслідок чого отримаємо $RG1 := 4A$, тобто

$$MO2 : RG1 := L1(RG1),$$

а також одночасно з цим виконаємо мікрооперацію зсуву регістра RG2 на 1 розряд вправо, внаслідок чого отримаємо $RG2 := 0.5C$, тобто

$$MO3 : RG2 := R1(RG2).$$

Після цього на суматорі Σ можна виконати віднімання $4A - C/2$, яке реалізуємо так:

$$\text{MO4 : } \text{RG2} := \text{RG1} + \overline{\text{RG2}} + 1,$$

де $\overline{\text{RG2}}$ – інверсія вмісту регістра RG2, а +1 є входним переносом суматора (CI = 1).

Далі виконаємо мікрооперації

$$\text{MO5 : } \text{RG2} := \text{RG2} + \text{RG1},$$

$$\text{MO6 : } \text{CTR} := \text{CTR} - 1,$$

які необхідно повторити B разів.

Дійсно,

$$\underbrace{4A + 4A + \dots + 4A}_{B+1} + (4A - C/2) = 4A(B+1) - C/2.$$

Граф-схема змістового мікроалгоритму виконання заданої операції показана на рис. 10.21.

Складемо список керуючих сигналів, які забезпечують виконання мікрооперацій MO1 – MO6:

$$\text{MO1} - y_1,$$

$$\text{MO2} - y_1,$$

$$\text{MO3} - y_7,$$

$$\text{MO4} - y_3, y_8, y_4, y_5,$$

$$\text{MO5} - y_3, y_8, y_5,$$

$$\text{MO6} - y_{10}.$$

3. Визначимо тривалість в тактах кожного керуючого сигналу з урахуванням затримок в структурних компонентах операційного автомата (табл. 10.13).

Таблиця 10.13

Таблиця тривалостей керуючих сигналів

Мікрооперація	Керуючі сигнали	Тривалості керуючих сигналів
MO1 : $\text{RG1} := \text{L1}(\text{RG1})$	y_1	t
MO2 : $\text{RG1} := \text{L1}(\text{RG1})$	y_1	t
MO3 : $\text{RG2} := \text{R1}(\text{RG2})$	y_7	t
MO4 : $\text{RG2} := \text{RG1} + \overline{\text{RG2}} + 1$	y_3, y_8, y_4, y_5	$2t, 2t, t, t$
MO5 : $\text{RG2} := \text{RG2} + \text{RG1}$	y_3, y_8, y_5	$2t, 2t, t$
MO6 : $\text{CTR} := \text{CTR} - 1$	y_{10}	t

Тривалість T циклу читання мікрокоманди з ПМК виберемо таким, щоб $T \geq 2t$.

4. Закодуємо логічні умови та побудуємо закодовану граф-схему мікроалгоритму.

Складемо таблицю позначень логічних умов (табл. 10.14).

Таблиця 10.14

Таблиця позначень логічних умов

Логічні умови	Позначення логічної умови
Пуск = 1	x_1
(CTR) = 0	x_2

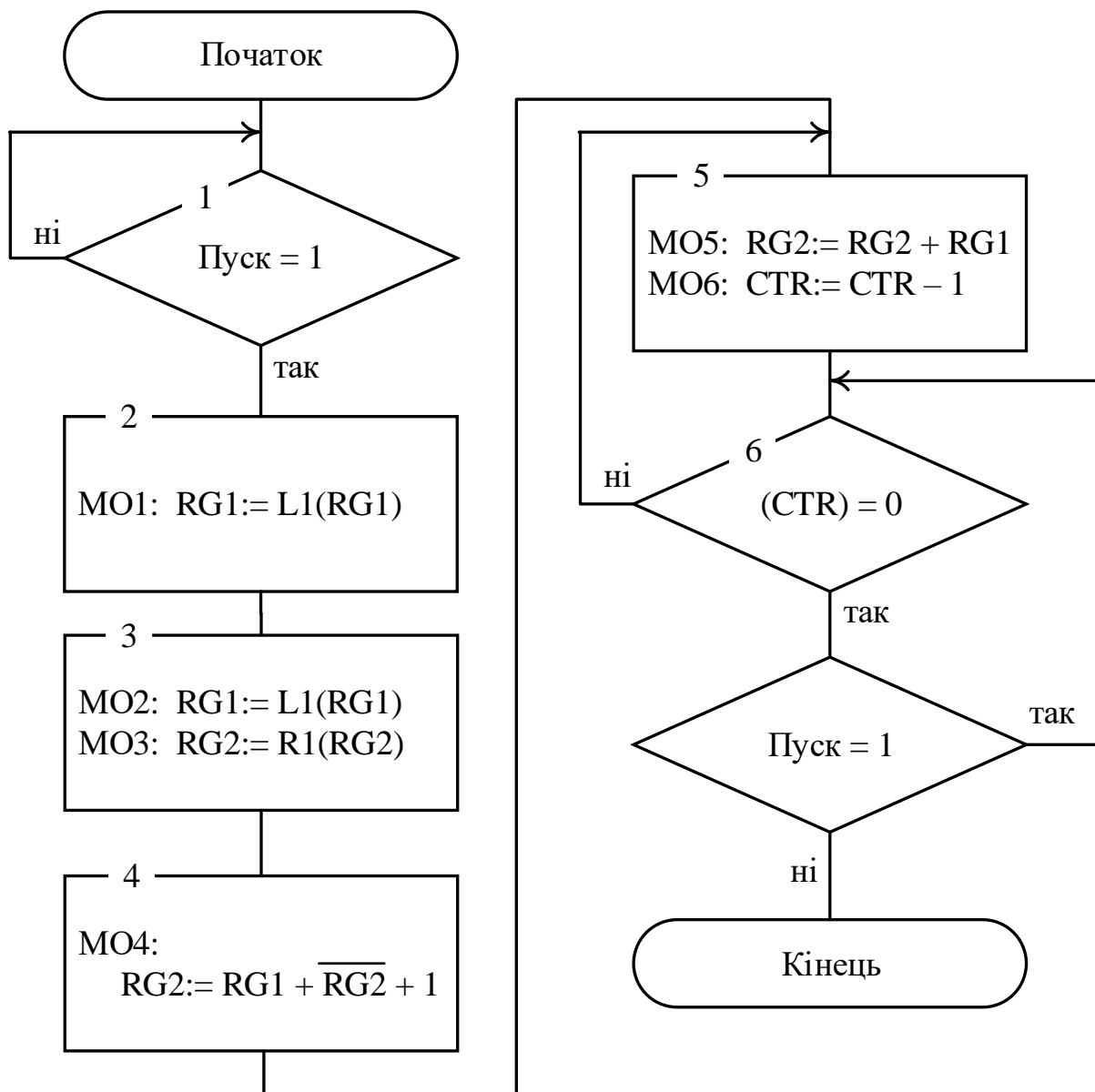


Рис. 10.21. Граф-схема змістового мікроалгоритму виконання операції $E = 4A(B + 1) - 0.5C$

Закодуємо логічні умови:

Логічна умова	Код
x_1	0 1
x_2	1 0
Відсутність логічної умови	0 0

Використовуючи позначення логічних умов та замінюючи опис мікрооперацій на відповідні керуючі сигнали отримаємо закодовану граф-схему мікроалгоритму операції (рис. 10.22), в якій присутня також операторна вершина, що містить сигнал y_{11} – зупинення керуючого автомата.

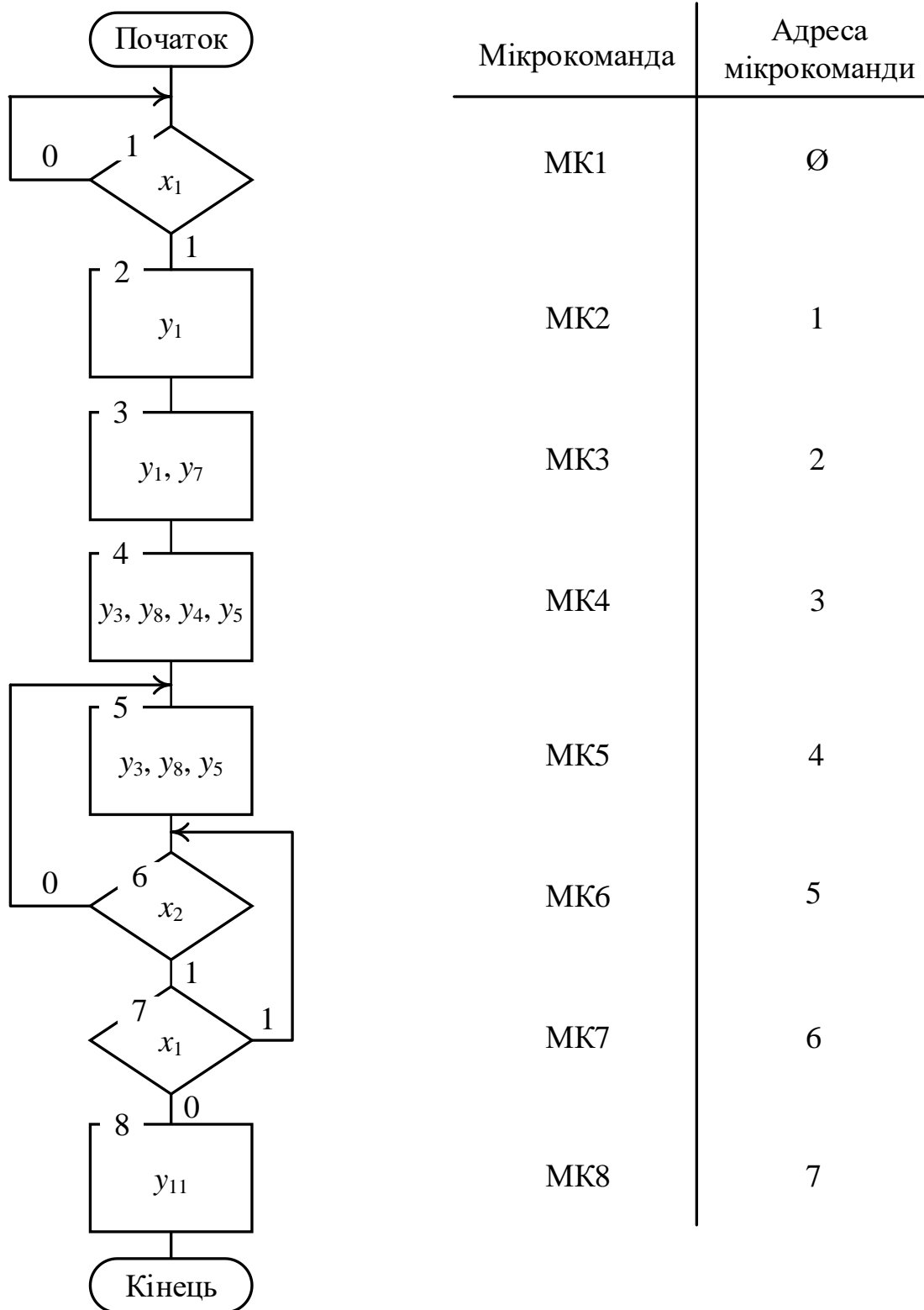


Рис. 10.22. Закодована граф-схема мікроалгоритму операції

5. Визначимо структуру мікрокоманди (рис. 10.23).

Вважатимемо, що одночасно (в одному такті) виконуватиметься не більше 4-х мікрокоманд. Загальна довжина мікрокоманди складає 22 розряди.

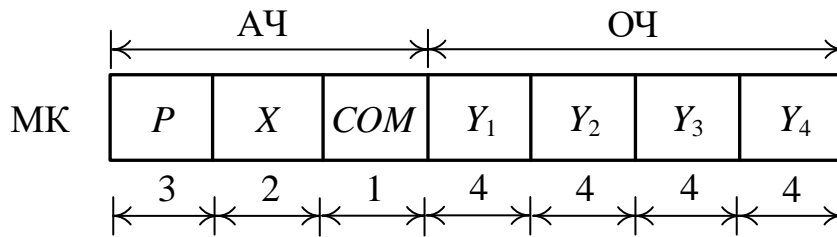


Рис. 10.23. Структура мікрокоманди

6. Побудуємо схему керуючого автомата.

Схема автомата є конкретизацією узагальненої структури керуючого автомата на рис. 10.20. Довжина регістра мікрокоманди РМК складає 22 розряди з поділом на поля як показано на рис. 10.23. Мультиплексор умов МУ має три входи '0', x_1 , x_2 . Для дешифрування полів $Y_1 - Y_4$ використовуються 4 неповні дешифратори (4×12). Для формування керуючих сигналів використовуються одинадцять 4-входових логічних елементів АБО.

7. Складемо мікропрограму, що реалізує задану операцію.

Структура мікропрограми має вигляд:

адреса	МК	виконувані мікрооперації / логічна умова
Ø	МК1	Ø Ø Ø Ø / x_1
1	МК2	y_1 Ø Ø Ø / 0
2	МК3	$y_1 y_7$ Ø Ø / 0
3	МК4	$y_3 y_8 y_4 y_5$ / 0
4	МК5	$y_3 y_8 y_5$ Ø / 0
5	МК6	Ø Ø Ø Ø / x_2
6	МК7	Ø Ø Ø Ø / x_1
7	МК8	Ø Ø Ø Ø / 0

Заповнюючи відповідні поля мікрокоманд двійковими кодами отримаємо мікропрограму, що підлягає розміщенню в ПМК (рис. 10.24).

Адреса ПМК	Мікрокоманда							
	P	X	COM	Y ₁	Y ₂	Y ₃	Y ₄	
0 0 0	0 0 0	0 1	1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	МК1
0 0 1	x x x	0 0	0	0 0 0 1	0 0 0 0	0 0 0 0	0 0 0 0	МК2
0 1 0	x x x	0 0	0	0 0 0 1	0 1 1 1	0 0 0 0	0 0 0 0	МК3
0 1 1	x x x	0 0	0	0 0 1 1	1 0 0 0	0 1 0 0	0 1 0 1	МК4
1 0 0	x x x	0 0	0	0 0 1 1	1 0 0 0	0 1 0 1	0 0 0 0	МК5
1 0 1	1 0 0	1 0	1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	МК6
1 1 0	1 0 1	0 1	0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	МК7
1 1 1	x x x	0 0	x	1 0 1 1	0 0 0 0	0 0 0 0	0 0 0 0	МК8

Рис. 10.24. Мікропрограма виконання операції $E = 4A(B + 1) - 0.5C$

В узагальненій структурі керуючого автомата на рис. 10.20 довжина операційної частини мікрокоманди становить

$$L_1 = g \cdot f = f \cdot \lceil \log_2(m+1) \rceil \text{ розрядів,}$$

а поля Y_1, \dots, Y_f підлягають дешифруванню.

Такий спосіб формування операційної частини мікрокоманди називають *вертикальним мікрокодуванням*.

При вертикальному мікрокодуванні операційну частину мікрокоманд поділяють на поля, в яких записують двійкові номери мікрооперацій, що виконуватимуться в даній мікрокоманді, а кожному полю має відповідати дешифратор. Сигнали мікрооперацій формують як диз'юнкції однойменних виходів дешифраторів.

Можливий інший спосіб формування операційної частини мікрокоманд, коли цю частину на поля не розбивають, а кожному розряду операційної частини мікрокоманд ставлять у відповідність безпосередньо сигнал мікрооперації. Такий спосіб формування операційної частини мікрокоманд називають *горизонтальним мікрокодуванням*.

За горизонтального мікрокодування довжина операційної частини мікрокоманди становить

$$L_2 = m + 1 \text{ розрядів.}$$

У цьому випадку спрощується структура керуючого автомата – не потрібні дешифратори та багатовходові логічні елементи АБО. Крім того, в одному такті можливе виконання будь-якої кількості мікрооперацій (до m мікрооперацій), тоді як при вертикальному мікрокодуванні в одному такті можливе виконання лише до f мікрооперацій.

Нехай цільовою функцією проектування під час синтезу керуючого автомата з програмовною логікою є мінімальна довжина мікрокоманд.

З'ясуємо, який спосіб мікрокодування – горизонтальне, чи вертикальне, слід застосовувати при формуванні операційної частини мікрокоманд.

Якщо в операторних вершинах закодованої граф-схеми мікроалгоритму максимальна кількість мікрооперацій в одній вершині становить f , то горизонтальне мікрокодування операційної частини мікрокоманд застосовувати доцільно, якщо

$$m + 1 \leq f \cdot \lceil \log_2(m+1) \rceil. \quad (10.1)$$

Інакше слід застосовувати вертикальне мікрокодування.

Нерівність (10.1) є умовою доцільності застосування горизонтального мікрокодування порівняно з вертикальним при реалізації заданого мікроалгоритму виконання операції.

З огляду на (10.1) у розв'язаній задачі 10.3 застосування горизонтального мікрокодування скоротило б операційну частину мікрокоманд з 16-ти розрядів до 11-ти розрядів. Крім того, відпала б потреба в дешифраторах та логічних елементах АБО.

Очевидно, що вертикальне мікрокодування може мати перевагу над горизонтальним лише в тому випадку, якщо в структурній схемі операційного автомата використовується велика кількість m сигналів мікрооперацій.

Структура керуючого автомата з програмовною логікою (див. рис. 10.19) не залежить від особливостей мікроалгоритму виконуваної операції (кількості умовних та операторних вершин у граф-схемі і їх зв'язків), тому один й той самий керуючий автомат може забезпечувати виконання довільної кількості операцій (команд), якщо в пам'яті мікрокоманд ПМК розмістити відповідні мікропрограми операцій (команд).

Кожну з мікропрограм розміщують в ПМК за певною адресою – початковою адресою мікропрограми. Під час виконання поточної команди програми відбувається перетворення коду операції (КОП) команди в початкову адресу мікропрограми, яка їй відповідає. Цю функцію виконує перетворювач початкової адреси (ППА) (рис. 10.25).

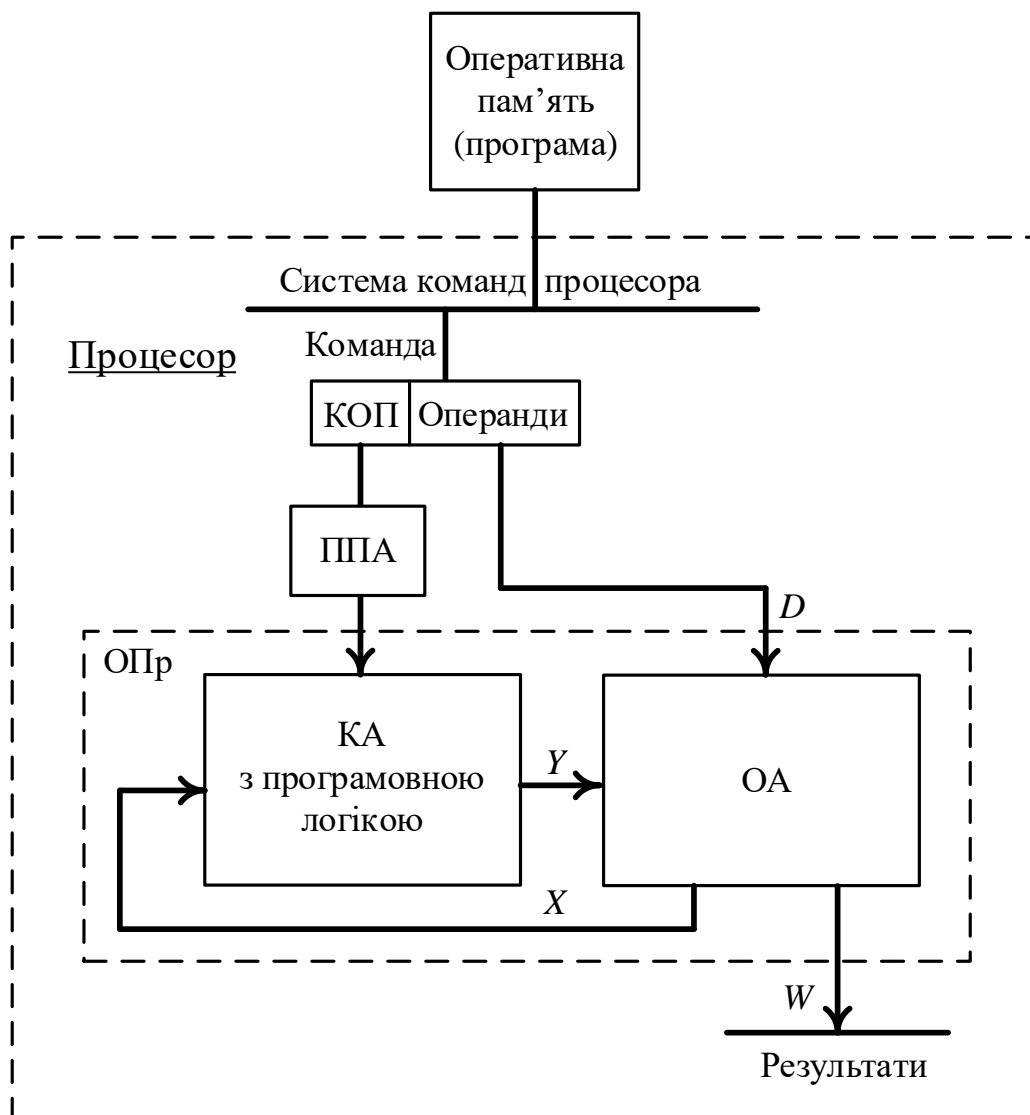


Рис. 10.25. Мікропрограмний спосіб реалізації системи команд процесора

Якщо в ПМК розмістити мікропрограми, що відповідають всім можливим командам процесора, то отримаємо можливість реалізувати систему команд процесора за допомогою єдиного керуючого автомата з програмовною логікою.

Запитання та завдання

1. Які чинники визначають функціонування керуючого автомата у складі операційного пристрою?
2. У якому вигляді подають мікроалгоритм операції (команди), якщо його реалізують мікропрограмним способом?
3. Що таке мікрокоманда? З яких структурних частин вона складається?
4. У якому функціональному вузлі зберігається масив мікрокоманд?
5. Поясніть, що означають терміни: “горизонтальне мікрокодування”, “вертикальне мікрокодування”.
6. Охарактеризуйте структуру мікрокоманди при застосуванні горизонтального (вертикального) мікрокодування її операційної частини.
7. Яку функцію виконує лічильник мікрокоманд в структурі керуючого автомата з програмовною логікою?
8. Як формується адреса наступної мікрокоманди в пам’яті мікрокоманд керуючого автомата з програмовною логікою?
9. Як формуються сигнали мікрооперацій в структурі керуючого автомата з програмовною логікою при застосуванні горизонтального (вертикального) мікропрограмування операційної частини мікрокоманд?
10. Опишіть процес виконання однієї мікрокоманди в керуючому автоматі з програмовною логікою при застосуванні вертикального (горизонтального) мікрокодування операційної частини мікрокоманд.
11. Дайте визначення П-автомата?
12. Перелічіть етапи синтезу П-автомата.
13. Нехай цільовою функцією проектування під час синтезу П-автомата є мінімальна довжина мікрокоманд. Який спосіб мікрокодування – горизонтальне чи вертикальне, слід застосувати при формуванні операційної частини мікрокоманд, якщо в структурній схемі операційного автомата використовуються 18 сигналів мікрооперацій, а в одному такті одночасно можуть виконуватись щонайбільше 3 мікрооперації?
14. Охарактеризуйте мікропрограмний спосіб реалізації системи команд процесора.

Задачі для самостійного розв'язування

1. Синтезувати П-автомат для керування виконанням операції $E = (A / 4) (B - 1) + 2(A + C)$ операційним автоматом, структурна схема якого подана на рис. 10.7, із застосуванням вертикального мікрокодування операційної частини мікрокоманд.
2. Синтезувати керуючий автомат з програмовною логікою для керування виконанням операції $E = 2A (B - 2) + 2(A + C + 1)$ операційним автоматом, структурна схема якого подана на рис. 10.7, із застосуванням горизонтального мікрокодування операційної частини мікрокоманд.
3. Побудувати закодовану граф-схему мікроалгоритму операції $E = AB + 4(A + C)$, яка має виконуватись операційним автоматом на рис. 10.7. Керування виконанням операції буде здійснюватись П-автоматом.
4. Розробити мікропрограму виконання операції $E = (A / 2) B + 0.25(A + C + 1)$ операційним автоматом на рис. 10.7, керування якою здійснюватиме П-автомат із застосуванням вертикального мікрокодування.

ВИСНОВКИ

При вивченні комп'ютера застосовують ієрархічний підхід до питання його організації та функціонування. Це означає, що на функціональному та логічному рівнях ієрархії комп'ютер розглядають як перетворювач інформації.

Перетворювач інформації є цифровим автоматом.

Цифровий автомат реалізують як логічну схему, яка є сукупністю певним чином з'єднаних між собою логічних елементів.

Логічний елемент є транзисторною структурою, реалізованою на поверхні або в товщі напівпровідникового матеріалу (пластини, підкладки), яка електрично виконує певну логічну функцію (AND, OR, NOT тощо), тобто здійснює перетворення сигналів.

Логічні елементи фізично реалізують у складі інтегральних мікросхем.

Розрізняють два види логічних схем – комбінаційні схеми та послідовнісні схеми (схеми з пам'яттю). Вважають, що комбінаційна схема має один стан, а послідовнісна схема може перебувати у кількох станах (найчастіше двох). Комбінаційна схема не має властивості пам'яті, а послідовнісна схема може запам'ятовувати один біт інформації. Логічна схема набуває властивість запам'ятовування значень електричних сигналів, якщо в її складі деякі логічні елементи пов'язані між собою перехресними (тригерними) зв'язками, які називають петлями.

Елементарну схему з пам'яттю можна отримати, якщо перехресними зв'язками з'єднати два логічні елементи І-НЕ або два логічні елементи АБО-НЕ. Такі елементарні схеми називають бістабільними схемами. На основі бістабільної схеми проектують тригер – схему, що забезпечує надійне зберігання одного біта. Тригер є елементарним цифровим автоматом, що реалізує функцію пам'яті.

Усі вузли, блоки, пристрої комп'ютера будують на основі комбінаційних схем та тригерів. Такі вузли комп'ютера як дешифратори, шифратори, мультиплексори, демультимплексори, схеми порівняння кодів, перетворювачі кодів, комбінаційні суматори – є комбінаційними схемами. А структури таких вузлів як регістри, лічильники, стеки проектують на основі комбінаційних схем та тригерів.

Оперативна пам'ять комп'ютера складається з тригерів, кожен з яких запам'ятовує один біт.

Оскільки логічний елемент (як транзисторна структура) електрично реалізує певну логічну операцію (елементарну функцію), то синтез логічних схем (комбінаційних та послідовнісних схем) ґрунтується на застосуванні

положень теорії перемикальних функцій – функцій, що можуть набувати лише двох значень – 0 та 1.

Теорія перемикальних функцій як математична конструкція базується на таких принципах: суперпозиції функцій, декомпозиції функцій, двоїстості функцій та функціональної повноти системи функцій.

Основною задачею, що вирішується в прикладній теорії цифрових автоматів, є синтез логічних схем на основі логічних елементів – транзисторних структур. Теорію перемикальних функцій використовують як математичний інструмент для синтезу логічної схеми – фізичного об'єкта.

Теорія перемикальних функцій сформувалась як сукупність чотирьох алгебр – Буля, Шеффера, Пірса, Жегалкіна. Кожній алгебрі (математичній конструкції) відповідає множина фізичних елементів – елементний (апаратний) базис. Алгебра Буля ґрунтується на трьох логічних операціях – AND, OR, NOT; їй відповідає елементний (фізичний) базис – логічні елементи I, АБО, НЕ. В алгебрі Шеффера використовується лише одна логічна операція – NAND, якій відповідає елементний базис – логічні елементи I-НЕ. Алгебра Пірса також побудована на застосуванні лише однієї логічної операції – NOR, а її матеріальною (фізичною) основою є елементи АБО-НЕ. Алгебра Жегалкіна використовує логічні операції AND, XOR, а також константу – одиницю; апаратним (елементним) базисом цієї алгебри є елементи I та суматори за модулем два (константу одиниця отримують схемотехнічно, спеціальний логічний елемент для цього не потрібен).

Синтез логічної схеми можна здійснювати в рамках лише однієї окремо взятої алгебри та відповідного їй елементного базису. Але практика проектування логічних схем показала, що доцільно виконувати синтез схем в об'єднаному елементному базисі кількох алгебр, наприклад, в елементному базисі трьох алгебр (на множині логічних елементів I, АБО, НЕ, I-НЕ, АБО-НЕ) – Буля-Шеффера-Пірса. Це може поліпшувати показники апаратної та часової складності проектованої логічної схеми.

Логічні схеми синтезують на основі функціонально повної системи логічних елементів (елементарних функцій) або об'єднання кількох функціонально повних систем.

Прикладами функціонально повних систем функцій є: функція NAND (їй відповідає логічний елемент I-НЕ) та функція NOR (їй відповідає логічний елемент АБО-НЕ). Це означає, що комп'ютер (включаючи й його оперативну пам'ять) може бути сконструйований з використанням логічних елементів лише одного типу – I-НЕ; комп'ютер включно з його оперативною пам'яттю можна також сконструювати виключно на основі лише логічних елементів АБО-НЕ.

При проектуванні вузлів, блоків, пристроїв комп'ютера використовують одночасно декілька функціонально повних систем логічних елементів.

Проектуванню логічної схеми має передувати процес мінімізації функції (системи функцій), яка підлягає апаратній реалізації. Для цієї мети розроблено методи мінімізації перемикальних функцій (Квайна, Квайна – Мак-Класкі, Блейка-Порецького та ін.) поданих в аналітичному вигляді, систем перемикальних функцій, а також функцій, поданих графічно – у вигляді таблиці (Вейча, Карно).

Задача мінімізації функцій, заданих аналітично, в теорії перемикальних функцій вирішена лише в класі диз'юнктивних форм. Результатом мінімізації при цьому є отримання мінімальної ДНФ функції. Якщо заданий елементний базис, який використовується для апаратної реалізації функції, вимагає подання функції в іншій (не диз'юнктивній) формі, то подальше спрощення аналітичної форми функції (після отримання МДНФ) можливе лише із застосуванням неформалізованих (евристичних) прийомів.

Задача мінімізації функцій, заданих графічно – у вигляді прямокутної таблиці (діаграми, карти) вирішена в класі диз'юнктивних і/або кон'юнктивних форм.

Особливим видом цифрових автоматів є керуючі цифрові автомати, які призначені для апаратного забезпечення процесу реалізації системи команд процесора. Кожна команда з системи команд процесора розглядається як складна дія, що розкладається на елементарні машинні дії (інвертування, додавання, зсув слів тощо) – мікрооперації, які виконуються вузлами процесора за один такт. Кожна команда з системи команд процесора розглядається як мікропрограма.

Тобто апаратна реалізація системи команд процесора ґрунтується на принципі мікропрограмного керування.

Реалізувати мікропрограму як процес виконання команди процесора можна двома способами – на основі керуючого автомата з жорсткою логікою або керуючого автомата з програмовною логікою.

Керуючий цифровий автомат з жорсткою логікою проектують як автомат Мура або як автомат Мілі (вони різняться способом видачі вихідних сигналів керування).

Задача синтезу керуючого цифрового автомата (Мілі або Мура) зводиться до синтезу комбінаційної схеми.

Слід зазначити, що задачі синтезу будь-яких структур на основі послідовнісних логічних схем (з пам'яттю) та синтезу керуючих автоматів в теорії цифрових автоматів зводяться до задачі синтезу комбінаційних схем.

Процесор комп'ютера розглядають як сукупність операційних пристроїв, на які покладається виконання системи команд процесора. До

складу операційного пристрою входять операційний автомат (регістри, лічильник, мультиплектори, суматор тощо) та керуючий автомат.

Для програміста процесор є набором функціональних вузлів (регістри загального призначення, акумулятор, стек, покажчик стека, програмний лічильник, регістр слова стану програми), до яких він може здійснювати доступ за допомогою відповідних команд з системи команд процесора (Асемблера).

Знання особливостей структурної організації комп'ютера, процесів апаратної реалізації операцій, а також можливість програмного доступу до окремих структурних компонентів процесора дають можливість програмісту створювати високоефективне програмне забезпечення. Включаючи в програмний код, створюваний мовою програмування високого рівня, окремі фрагменти мовою процесора (мовою Асемблера), програміст може забезпечувати високу продуктивність комп'ютерної системи при вирішенні конкретних прикладних задач.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Joseph Cavanagh Computer Arithmetic and Verilog HDL Fundamentals. – Santa Clara University, California, USA: CRC Press, 2010. – 952 p.
2. Азаров, О.Д. Комп'ютерна схемотехніка: підручник / О.Д. Азаров, В.А. Гарнага, Я.М. Клятченко, В.П.Тарасенко. – Вінниця : ВНТУ, 2018. – 230 с.
3. Бабич, М.П. Комп'ютерна схемотехніка : Навч. посіб. / М.П. Бабич, І.А. Жуков – К. : МК-Прес, 2004. – 412 с.
4. Бардачов, Ю.М. Дискретна математика: підручник / Ю.М. Бардачов, Н.А. Соколова, В.Є. Ходаков // За ред. В.Є. Ходакова. – К. : Вид-во Вища школа, 2002. – 287 с.
5. Бондаренко, М.Ф. Комп'ютерна дискретна математика: підручник / М.Ф. Бондаренко, Н.В. Білоус, А.Г. Руткас. – Харків : Компанія СМІТ, 2004. – 480 с.
6. Букреев, И.Н. Микроэлектронные схемы цифровых устройств / И.Н. Букреев, В.М. Мансуров, В.И. Горячев. – М. : Сов. Радио, 1975. – 368 с.
7. Галчѐнков, О.Н. Компьютерная схемотехника и архитектура компьютеров / О.Н. Галчѐнков, А.Н. Долголенко, В.И. Корнейчук. Учебное пособие. – К. : Корнійчук, 2013. – 604 с.
8. Ефимов, И.Е. Микроэлектроника. Физические и технологические основы, надежность / И.Е. Ефимов, И.Я. Козырь, Ю.И. Горбунов. – М. : Высш. шк., 1986. – 464 с.
9. Жабін, В.І. Прикладна теорія цифрових автоматів / В.І. Жабін, І.А. Жуков, І.А. Клименко, В.В. Ткаченко. – К. : Вид-во Нац. авіац. ун-ту “НАУ-друк”, 2009. – 360 с.
10. Жабін, В.І. Цифрові автомати. Практикум / В.І. Жабін, В.В. Ткаченко. – К. : ТОВ “ВЕК+”, 2004. – 160 с.
11. Кривуля, Г.Ф. Схемотехніка / Г.Ф. Кривуля, В.М. Рябенський, В.С. Буряк. – Харків: Компанія СМІТ, 2007. – 250 с.
12. Майоров, С.А. Принципы организации цифровых машин / С.А. Майоров, Г.И. Новиков. – Л. : Машиностроение, 1977. – 432 с.
13. Малиновський, Б.М. Відоме й невідоме в історії інформаційних технологій в Україні / Б.М. Малиновський. – К. : Видавничий дім “Академперіодика”, 2001. – 214 с.
14. Оранский, А.М. Аппаратные методы в ЦВТ / А.М. Оранский. – Минск : Изд-во БГУ, 1977. – 208 с.
15. Папернов, А.А. Логические основы цифровых машин и программирование / А.А. Папернов. – М. : Наука, 1968. – 591 с.

16. Поспелов, Д.А. Логические методы анализа и синтеза схем / Д.А. Поспелов. – М. : Энергия, 1974. – 367 с.
17. Проектирование цифровых вычислительных машин: Учеб. пособие для студентов вузов / Под ред. С.А. Майорова. – М. : Высшая шк., 1972. – 344 с.
18. Рабинович, З.Л. Типовые операции в вычислительных машинах / З.Л. Рабинович, В.А. Раманаускас. – К. : Техніка, 1980. – 264 с.
19. Савельев, А.Я. Арифметические и логические основы цифровых автоматов: Учебник / А.Я. Савельев. – М. : Высш. шк., 1980. – 255 с.
20. Савельев, А.Я. Прикладная теория цифровых автоматов: Учеб. для вузов по спец. ЭВМ / А.Я. Савельев. – М. : Высш. шк., 1987. – 344 с.
21. Самофалов, К.Г. Цифровые ЭВМ. Теория и проектирование / К.Г. Самофалов, В.И. Корнейчук, В.П. Тарасенко. – К. : Вища школа, 1989. – 424 с.
22. Самофалов, К.Г. Цифровые ЭВМ. Практикум/ К.Г. Самофалов, В.И. Корнейчук, В.П. Тарасенко, В.И. Жабин. – К. : Вища школа, 1990. – 216 с.
23. Самофалов, К.Г. Прикладная теория цифровых автоматов / К.Г. Самофалов, А.М. Романкевич, В.Н. Валуйский, Ю.С. Каневский, М.М. Пиневиц. – К. : Вища школа, 1987. – 375 с.

ІМЕННИЙ ПОКАЖЧИК

Бардін (Джон Бардін, англ. John Bardeen)	35
Бebbідж (Чарлз Беббідж, англ. Charles Babbage)	6, 7
Блейк	169, 466
Браттейн (Волтер Гаузер Браттейн, англ. Walter Houser Brattain)	35
Буль (Джордж Буль, англ. George Boole)	55-62, 68, 92, 98, 102
Вейч (Едвард Вейч, англ. Edward Westbrook Veitch)	170-178, 180-182, 185, 191
Вілкс (Моріс Вінсент Вілкс, англ. Maurice Vincent Wilkes)	7
Вільямс (Фредерік Калленд Вільямс, англ. Frederic Calland Williams)	7
Глушков В.М. (Глушков Віктор Михайлович)	436-438
Грей (Френк Грей, англ. Frank Gray)	182, 256-259
Джонсон (Роберт Ройс Джонсон, англ. Robert Royce Johnson)	409-411
Еккерт (Джон Преспер Еккерт-молодший, англ. John Presper Eckert, Jr.)	7
Жегалкін (Жегалкін Іван Іванович)	87-92, 129, 131, 132, 135, 210, 465
Карно (Моріс Карно, англ. Maurice Karnaugh)	182, 184-185, 192-194, 277, 278
Квайн (Віллард Ван Орман Квайн, англ. Willard Van Orman Quine)	52-54, 59, 138, 139, 141-148, 151-153, 155-157, 160-167, 169, 188-190, 193, 194, 216
Кілбурн (Том Кілбурн, англ. Tom Kilburn)	7
де Кольмар (Шарль Ксав'є Тома де Кольмар, фр. Charles Xavier Thomas de Colmar)	6
Лебедев С.О. (Лебедев Сергій Олексійович)	7, 8

Лейбніц (Готфрід Вільгельм Лейбніц, нім. Gottfried Wilhelm Leibniz)	6
Мак-Класкі (Едвард Джозеф Мак-Класкі, англ. Edward Joseph McCluskey)	152-160, 167, 168, 189, 190, 193, 202-206, 216
Мілі (Джордж Мілі, англ. George Mealy)	432-435, 439-442, 445, 447
де Морган (Аугустус де Морган, англ. Augustus De Morgan)	60-62, 76, 111-113, 124
Мочлі (Джон Вільям Мочлі, англ. John William Mauchly)	7
Мур (Едвард Форрест Мур, англ. Edward Forrest Moore)	432-435, 443, 445-451
фон Нейман (Джон фон Нейман, англ. John von Neumann)	7, 8, 17
Паскаль (Блез Паскаль, фр. Blaise Pascal)	6
Петрік (Стенлі Петрік, англ. Stanley Petrick)	161-164, 167, 169, 187, 189, 190, 195, 213, 214, 222
Пірс (Чарлз Сандерс Пірс, англ. Charles Sanders Peirce)	24, 25, 28, 29, 80-85, 98-100, 103, 104, 108, 109, 123-125, 134-136
Порецький (Порецький Платон Сергійович)	169, 201, 466
Пост (Еміль Леон Пост, пол. Emil Leon Post)	129, 130, 137
Тьюрінг (Алан Матісон Тьюрінг, англ. Alan Mathison Turing)	7
Цузе (Конрад Цузе, нім. Konrad Zuse)	7
Шеннон (Клод Елвуд Шеннон, англ. Claude Elwood Shannon)	96, 105, 110, 267, 272
Шеффер (Генрі Моріс Шеффер, англ. Henry Maurice Sheffer)	24, 25, 28, 29, 73-80, 98, 100, 103, 107, 108, 116, 123, 124, 130, 131, 134-136
Шоклі (Вільям Бредфорд Шоклі, англ. William Bradford Shockley)	35
Яблонський (Яблонський Сергій Всеволодович)	130, 137

ПРЕДМЕТНИЙ ПОКАЖЧИК

DV-тригер 322, 323, 359-366
D-тригер 322, 323, 356, 360,
362-366, 385-387, 410-412
E-тригер 322, 323
JK-тригер 323, 324, 354-356,
360-364, 366, 384, 386, 396,
401, 402
n-p-n транзистор 36, 38, 41
p-n перехід 35-37
p-n-p транзистор 36, 38
RS-тригер 322, 323, 327-332,
346-352, 360-366, 383, 384,
389, 390
R-тригер 322, 323
S-тригер 322, 323
T-тригер 323, 324, 352-355,
363-366, 385, 396
Z-покриття функції 155, 157, 159,
167, 203

А

автомат 432-438, 440-443, 445-447,
450, 459
автомат Мілі 432-435, 439-442,
444, 445, 447, 449, 450, 466
автомат Мура 432-435, 443,
445-451, 466
адреса 223, 249, 405, 406, 417, 418,
421-424, 429, 452-454, 459,
461, 462
адреса повернення 422-424
аксіома
повторення (тавтології) 56, 57
аксіома подвійного
заперечення 56, 75, 83

аксіоми алгебри Жегалкіна 87, 92
аксіоми алгебри Пірса 81, 86
аксіоми алгебри Шеффера 73, 79
аксіоми булевої алгебри 59, 72
акумулятор 426, 427, 467
алгебра Буля 55-71, 92, 97-99, 102,
105, 106
алгебра Жегалкіна 87-92, 135, 210
алгебра Пірса 80-85, 98-100, 103,
104, 108, 109
алгебра Шеффера 73-76, 78, 79,
98, 100, 103, 107, 108, 135
аналіз автомата 435
аналіз комбінаційної
схеми 208-211
аналітична машина 6
апаратна складність 52, 53
арифметико-логічний
пристрій 426
арифметичний зсув слова 376, 377
асинхронне встановлення тригера
в початковий стан 335, 344,
348, 354
асинхронний тригер 324, 327, 331,
332, 342, 359

Б

багатовходовий логічний
елемент 30, 57, 58
багаторозрядний
демультиплексор 282
багаторозрядний
мультиплексор 264
біполярний транзистор 36, 38

бістабільна схема на
елементах АБО-НЕ 313-318,
327, 328

бістабільна схема на
елементах І-НЕ 307-312,
330, 331

блок пріоритетних переривань 249

В

вертикальне
мікрокодування 460, 461
видача слова з регістра 378,
380-382

витік (транзистора) 42, 43, 45
відсічення (транзистора) 37, 38
вольт-амперна характеристика 35,
36, 44

вхідний перенос
суматора 289-294, 438, 455, 456

Г

генератор синхросигналів 304-306,
445, 449

горизонтальне
мікрокодування 460
граф автомата 434, 435, 441-443,
445, 446

графічні методи мінімізації
перемикальних функцій 169-185
граф-схема змістового
мікроалгоритму 437, 439, 440,
454-457

Д

двійкова величина 9
двійкова послідовність 15
двійкова цифра 15
двійковий вектор 15, 17
двійковий код 19, 223, 296, 459
двійковий набір 29, 64-67, 74, 81,
126, 157, 170, 171

двійково-десятковий
код (ДДК) 255, 256, 258, 259

двійково-десятковий
лічильник 409-412, 414
двоїста функція 120, 121, 123,
127, 128

двокаскадна комбінаційна
схема 76, 83

двотактний тригер 321

ДДК “4-2-2-1” 414

ДДК “8-4-2-1” 410

ДДНФ 65, 68-71

декомпозиція перемикальної
функції 94-109, 268, 278

декремент 396-398, 404

декрементний лічильник 397,
400-402

демультиплексор 279-282

дешифратор 223, 224

дешифратор з інверсними
виходами 226, 227, 233, 234, 261

дешифратор з прямими
виходами 224-226, 234, 262, 265

диз’юнктивний терм 63, 66, 67,
180, 181

диз’юнкція 23-25, 29, 41, 49, 55,
57, 60, 61, 63, 65, 68, 71, 92,
105, 120, 123, 136, 140, 143,
145, 149, 152, 157, 162, 164,
166, 168, 176, 181, 190, 197,
201-203, 214, 226, 235, 251,
252, 372, 373, 387, 389, 454,
460

дискретизація фізичного
сигналу 11, 14

дискретна величина 9-10, 18

дискретний сигнал 11

діаграма Вейча 170-178, 180-182,
185, 191, 193, 231, 267-270

ДКНФ 62, 67-71, 111-114

ДНФ 63, 142

Е

еквівалентна комбінаційна
схема 120, 209

електрод (транзистора) 36, 42, 43
електронна обчислювальна
машина 7, 9
елемент АБО 23, 27, 28, 30, 38-40,
56, 57, 59, 61
елемент АБО-НЕ 24, 27, 28, 30, 40,
45-47, 61, 83
елемент І 23, 24, 27, 28, 30, 34, 39,
56, 58, 59, 61
елемент І-НЕ 24, 27, 28, 30, 40, 45,
61, 76
елемент НЕ 22, 37, 38, 45, 47
елемент пам'яті 304, 307, 308, 311,
313, 314, 316, 318, 319
емітер 36, 37, 41

З

заборонений набір 230
задача аналізу комбінаційної
схеми 208-210
задача синтезу комбінаційної
схеми 212-220
задача факторизації 141
задній фронт сигналу 11, 339, 340
закодована граф-схема
мікроалгоритму 437, 440, 441, 443,
445, 446
закон абсорбції 59
закон асоціативності 56, 74, 81
закон дистрибутивності 57, 58, 74,
81, 124
закон комутативності 56
закон поглинання 124
закон склеювання 59, 124
закон функціонування
автомата 432-435, 437
закони булевої алгебри 68, 71
замкнений клас функцій 129, 137
запам'ятовувальний елемент 320
заперечення 22-25, 55, 56, 60-64
заперечення суми за модулем
два 26, 123
заслін (транзистора) 6, 41-43, 45

І

імпліканта функції 140, 153, 188
імплікантна матриця 144
інверсний код слова 380
інверсний унарний код 226,
237, 245
інвертор 22, 27, 28, 36-38, 43, 44,
46, 47
інвертування 38, 120, 126
інкремент 396-398, 402, 404, 406
інкрементний лічильник 398-401,
414
інтегральна мікросхема 33, 34,
47, 53

К

канал (транзистора) 41-45
канонічна форма функції 62, 74,
81, 88
карта Карно 182, 184, 185,
192, 193
квантування сигналу за рівнем 11,
13, 17
квантування сигналу за часом 11,
13
керований повторювач 240-244,
248
керуючий автомат з жорсткою
логікою 432, 435-437, 447,
449, 466
керуючий автомат з
програмовною логікою 451, 453,
454, 460-463, 466
керуючий сигнал 263, 265, 431,
436-449
кількість інформації 15
клас монотонних функцій 129, 131
клас самодвоїстих функцій 129
клас функцій 128, 129, 134, 137
клас функцій, що зберігають
нуль 129

клас функцій, що зберігають
одиницю 129
класи Поста 129
КНФ 70, 151, 170
код Грея 182, 256-259, 409
кодування сигналу 17, 455
колектор (транзистора) 36, 37,
40, 41
комбінаційна схема 31, 32, 52-54,
78, 85
комбінаційний суматор 289, 294,
426
компаратор 284, 286-289
кон'юнктивний терм 63-65,
70, 140
кон'юнкція 23-25, 29, 30
конституента нуля 66-68
конституента одиниці 64, 65,
68, 71
кортеж 21

Л

лінійна згортка суми за
модулем два 90, 91
лічильний сигнал 397-399, 401
лічильник 395-410
лічильник Джонсона 409, 410
лічильник з паралельним
переносом 400-404
лічильник з послідовним
переносом 399, 400
лічильник мікрокоманд 453
лічильні тригери 323, 396
логічна одиниця 11-15, 45,
241, 243
логічна умова покриття
функції 163, 164, 166, 168,
191, 214
логічна схема 30-32
логічний елемент 22, 23, 26-30, 33,
37-41, 45-47, 61
логічний зсув слова 370, 374, 375
логічний нуль 11-15, 45, 370, 371

логічний сигнал 10, 12, 13, 15
логічні операції 21-26, 49-51, 55,
80, 81, 87
логічні умови 430, 431, 433, 436,
440-442, 445, 446, 452-454,
456, 457

М

макстерм 63, 64
МДН-транзистор 41-46
метод мінімізації
Квайна – Мак-Класкі 152-160
метод мінімізації на основі
карт Карно 182, 184, 185,
192, 193
метод мінімізації Квайна 142-145
метод мінімізації на основі
діаграм Вейча 170, 174-176,
179-183
механізм вкладених
підпрограм 422-424
МДНФ 141, 143-145, 147, 159,
177-179, 184
мікроалгоритм 429-431, 452, 454
мікрокоманда 452-454, 458-460
мікрооперація 368-382, 386-392
мікропрограма 430, 452, 454, 459
мінімізація неповністю
визначених функцій 186-193
мінімізація перемикальних
функцій 138-141, 144
мінтерм 63, 64
місність функції 116
МКНФ 180-184
модуль (період) лічильника 395
МОН-транзистор 41
мультиплексор 262-266

Н

надлишкова функціонально повна
система функцій 135, 136
накопичувальний суматор 415,
416, 426

насичення (транзистора) 37, 38, 43
неповний дешифратор 226,
230-232, 256
неповністю (частково) визначена
перемикальна функція 20,
186, 189, 191, 193
нормальна форма функції 65, 67,
212, 213

О

область визначення
перемикальної функції 29
область значень перемикальної
функції 29
обчислювальна машина 5-7
однорозрядний комбінаційний
суматор 289, 290, 294
однотактний тригер 321, 325
ознаки результату 430
оперативна пам'ять 8, 429, 465
оператор логічного
елемента 49, 50
операторна форма функції 78, 116,
117, 219
операційний автомат 430, 436, 438
операційний пристрій 430, 431,
448, 451
операція NAND (І-НЕ) 73, 74
операція NOR (АБО-НЕ) 80, 81
оцінювання складності 52-54

П

П-автомат 454, 455
пам'ять автомата 436, 437,
442, 450
пам'ять мікрокоманд 452, 454,
459, 461
парафазний код слова 381, 382
передній фронт сигналу 11, 340
передповний клас
функцій 129, 134
перемикальна функція 18-23,
28-33, 48-52, 55, 62

перетворювач інформації 18,
19, 464
перетворювач кодів 201, 255,
260, 261
перетворювач початкової
адреси 461
петля (електрична) 32, 302, 303,
308, 314
підпрограма 295, 422-424
пірамідальна згортка суми за
модулем два 90, 91
повний дешифратор 223-229, 234
повний клас функцій 129, 137
повністю визначена
функція 20, 187
повторювач 22, 27, 28, 240-242
позначена таблиця переходів
автомата 434, 435
показчик стека 421, 424, 427, 467
поліном Жегалкіна 88, 89, 92, 129,
131, 132
польовий транзистор 41
послідовнісна схема 32, 302-305,
307, 313
початковий стан автомата 445, 449
правила де Моргана 60, 61,
111-113, 124
правило позначення станів
автомата 440, 443, 449
принцип двоїстості
функцій 120-125
принцип мікропрограмного
керування 429-431, 466
принцип програмного
керування 5, 7
принцип суперпозиції функцій 51
пристрій 6-8, 15, 17, 19
пристрій введення 6, 7
пристрій виведення 6, 7
пристрій керування 6, 7, 407
пріоритетний шифратор 248-255
проблема мінімізації
перемикальних функцій 138

проблема функціональної повноти системи перемикальних функцій 128, 136, 137
провідність n -типу 35
провідність p -типу 35
програмістська модель процесора 427, 428
програмний лічильник 405, 406, 422
програмовна логічна матриця 297, 298, 300
проста імпліканта функції 140, 141, 143, 155, 157
процесор 7, 13-15, 426-428, 451, 461
прямий код слова 381

Р

реверсивний лічильник 402-404, 421
реверсивний регістр 370, 394, 438
регістр 368-382, 404, 415, 426
регістр зсуву 370, 393, 395
регістр/лічильник 404, 405
регістровий запам'ятовувальний пристрій 416-418
рівень логічного нуля 11-16, 241
рівень логічної одиниці 11-16, 241
рівняння збудження бістабільної схеми 312, 318
розкладання перемикальної функції 94-105, 267, 272, 274
розрядність демультіплексора 281
розрядність лічильника 407
розрядність мультіплексора 264
розширена алгебра 110-113

С

самодвоїста функція 125-127
сигнал синхронізації 320, 325, 333, 369
синтез автомата 435, 437

синтез комбінаційної схеми 212-220, 267-276
синхронізація 320, 321, 324, 325, 333, 334, 341
синхронний тригер 325, 331, 333, 335-346, 350, 352, 354, 357, 358
система високих потенціалів 11-13, 17
система команд процесора 430, 448, 451, 461
система низьких потенціалів 11, 12
система числення 7, 8
системи перемикальних функцій 18, 31, 130, 135-137, 195-200, 245, 254, 256, 260
складність за Квайном 52-54
скорочена ДНФ функції 140, 141, 143-145, 155-157, 159, 165, 167, 188, 190, 197, 198, 201-203
слово стану програми 427, 467
співвідношення неповного склеювання 143
співвідношення поглинання 143
спосіб Петріка знаходження тупикових ДНФ функції 161-166, 168
стан автомата 432-434, 437, 440-443, 445, 446
стан лічильника 395-400, 403, 407-409, 411
стан схеми 303, 304, 311, 315-317, 319
стек 418-424, 427
стік (транзистора) 42, 43
стрілка Пірса 24, 25, 123, 125, 134
структурна таблиця автомата 442, 446
сума за модулем два 24, 26-28, 87-90, 123, 135, 136, 296, 373
суматор 289-295, 415

суматор за модулем два 24, 27, 28,
90, 91, 135, 292
суміщена таблиця автомата 433,
434
суперпозиція логічних
елементів 76
суперпозиція функцій 50, 129, 465
сусідство конститuent
одиниці 179
схема з пам'яттю 32, 302, 464
схема операційного автомата 437,
438, 454, 461, 462
схема порівняння кодів 284, 286,
287
схема формування ознаки
парності 295-297

Т

таблиця виходів автомата 433, 435
таблиця істинності функції 19, 23,
69, 95-98, 139, 140, 176
таблиця переходів автомата 433
таблиця переходів тригера 322,
353, 363
таблиця покриття функції 144,
147, 150, 159, 216
таблиця функції збудження
тригера 382-386
такт 13-17, 305, 309, 314, 368,
399, 460
теорема Квайна 143, 197
теорема Поста-Яблонського 130,
137
теорема Шеннона 96, 105, 267, 272
терм 62-64, 66, 68, 74, 76, 80, 81,
83, 92, 116, 140
терм Пірса 81, 83, 84
терм Шеффера 74, 76, 77
транзистор 33, 35-46, 48, 425,
427, 464
транзисторний ключ 38
тривалість такту 14, 16
тривіальна функція 21

тривіальний автомат 32
тригер 320-329, 333-360, 365-373
тригер з внутрішньою
затримкою 325, 337, 345,
356
тригер, керований
перепадом (фронтом)
синхросигналу 325, 326
тригер, керований рівнем
синхросигналу 325, 326,
332-336
тупикова ДНФ функції 141, 144,
145, 147, 164, 168, 192

У

умовне графічне
позначення (УГП) 22-24, 27,
28, 30, 47, 225, 227, 246, 248,
251-253, 356-359, 393, 394
унарний код 223, 224, 239,
245, 255

Ф

фон-нейманівська архітектура 8
форма АБО-НЕ/АБО-НЕ
функції 112, 113, 212, 213, 217
форма І-НЕ/І-НЕ функції 111, 112,
212, 219
функції збудження бістабільної
схеми 312, 318, 346, 349
функції міжрозрядних переносів
лічильника 401-403
функціональна повнота системи
перемикальних
функцій 128-136
функціонально повна система
функцій 130, 135, 136
функція АБО (диз'юнкції) 23, 25,
29, 30, 55-57, 59-62
функція АБО-НЕ 24, 25, 27-30,
80-85
функція Виключне АБО 24, 26-28,
87

функція виходів автомата 432-435
функція збудження
 тригера 382-386
функція І (кон'юнкції) 23, 25,
 27-30, 55-57, 58, 60, 61
функція І-НЕ 24, 25, 27-30, 73-78
функція Константа нуль 21, 23,
 121, 122
функція Константа одиниця 21,
 23, 121, 122
функція НЕ (заперечення) 21, 22,
 55, 56, 62
функція переходів
 тригера 322-324
функція Пірса 24, 25, 27-30, 80-85
функція повторення 21-23, 27-28,
 121, 126, 240
функція рівнозначності 24, 26,
 284, 285
функція Шеффера 24, 25, 27-30,
 73-78

Х

характеристичне рівняння
 бістабільної схеми 311, 317
характеристичне
 рівняння RS-тригера 331

Ц

циклічний зсув слова 370, 377, 378
цифрова величина 9
цифровий автомат 32, 435,
 464-466
цифровий індикатор 259-261

Ч

час затримки поширення
 сигналів 308, 319
часовий інтервал 9
частково визначена перемикальна
 функція 187, 191, 194

Ш

швидкодія схеми 53, 54
шифратор 239-241, 243-249
шифратор з інверсними
 входами 245-248, 250,
 253, 254
шифратор з прямими
 входами 239-241, 243, 251
штрих Шеффера 24, 25, 27-30,
 73-78

ГЛОСАРІЙ

А

АБО-НЕ – NOR (Not OR)

автомат – automaton, sequential machine

абстрактний ~ abstract automaton

~ Мілі – Mealy automaton (machine)

~ Мура – Moore automaton (machine)

дискретний ~ discrete automaton

цифровий ~ digital automaton

автоматичне оброблення – automation

автоматизувати – automate

автоматичний – automatic(al)

адитивний – additive

адреса (адресувати) – address

~ виклику – call address

~ даних – data address

~ операнда – operand address

~ пам'яті – memory address

~ переходу – jump address

~ повернення – return address

~ програми – program address

~ слова – word address

нульова ~ zero address

початкова ~ leading address

адресація – addressing

адресовний – addressable

аксіома – axiom (principle)

акцептор – acceptor

алгебра – algebra

~ логіки – algebra of logic

~ перемикальних функцій – switching algebra

~ Пірса – Peirce algebra

~ Шеффера – Sheffer algebra

булева ~ Boolean algebra

алгебраїчні перетворення – algebraic manipulation

алгебраїчний – algebraic(al)

алгоритм – algorithm, procedure

~ обчислень – computational procedure

амплітуда – amplitude

аналіз – analysis

~ схем – circuit analysis

аналізувати – analyze

аналоговий – analog

ангстрем (10^{-10} м) – angstrom

абсорбція (поглинання) – absorption

закон (правило) абсорбції (поглинання) – absorption law

апаратні засоби – hardware

апаратура – facility

апаратно – by hardware

архітектура – architecture

~ комп'ютера – computer architecture

асоціативність – associativity

асоціативний – associative

~ (сполучний) закон – associative law

Б

булева алгебра – Boolean algebra

булева властивість – Boolean property

булів вираз – Boolean

база (транзистора) – base

базис – basis
 елементний ~ elemental basis
байт – byte
бінарний – binary
біполярний – bipolar
бістабільний – bistable
біт – bit
 ~ парності – parity bit
блок (вузол) – block, unit
 арифметичний ~ arithmetic unit
 ~ оброблення – processing
 block
 ~ пам'яті – memory (storage)
 block
блок-схема (алгоритму) – detail
 chart, flowchart, flow diagram
 ~ програми – program flowchart
 логічна ~ logic chart, logic(al)
 diagram
 складання блок-схеми –
 flowcharting
 стрілка (в блок-схемі) –
 flow-line
буфер – buffer
 вихідний ~ output buffer
буква – letter, character
 буквене позначення – lettering
 буквено-цифровий –
 alpha-numeric,
 alphabetic-numeric

В

введення – input
 ~ даних – data input
 ~ з клавіатури – keyboard input
введення-виведення –
 input/output
вектор – vector
 вектор-рядок – row vector
 одиничний ~ unit vector
векторний – vectorial

величина – magnitude (матем.),
 quantity (кількість),
 value (див. *значення*),
 абсолютна ~ magnitude
 аналогова ~ analog quantity
 беззнакова ~ unsigned operand
 безрозмірна ~ dimensionless
 quantity
 векторна ~ vector quantity
 ~ числа – magnitude of the
 number
 випадкова ~ random quantity
 вихідна ~ output quantity
 вхідна ~ input quantity
 дискретна ~ discrete quantity
 (за)дана ~ datum
 за абсолютною величиною (за
 модулем) – in absolute
 magnitude
 змінна ~ alternating (variable)
 quantity
 невідома ~ unknown quantity
 нескінченно-мала ~
 infinitesimal
 порогова ~ threshold quantity
 скалярна ~ scalar quantity
 цифрова ~ digital quantity
вентиль – gate
 ~ суматора – adder gate
 інвертувальний ~ inverting gate
 логічний ~ logic(al) gate
вершина – vertex
 кінцева ~ terminal vertex
 початкова ~ initial vertex
вибір (варіант) – option
виведення – output
 ~ даних – data output
 ~ результату – readout
вивод (мікросхеми) – pin
видача результату – readout
виключне АБО – exclusive
 disjunction, XOR

вимірність – dimension, dimensionality
N-вимірний – *N*-dimensional
вираз – expression
перетворити ~ to convert the expression
перетворити ~ в диз'юнктивну форму – to convert the expression to a sum-of-products form
вистік (у транзисторі) – source
вихід – output
~ переносу (вихідний перенос) – carry output
інверсний ~ inverted output
прямий ~ straight output
відлік – count, counting, indication, reading
цифровий ~ digital indication
віднімання – subtraction
відношення – ratio, relation, relationship
~ сигнал-завада – signal-to-noise ratio
~ сигналу одиниці до сигналу нуля – one-to-zero ratio
бінарне ~ dyadic (binary) relation
~ замикання – closure relation
~ еквівалентності – equivalence relation
~ включення – inclusion relation
відповідність – correspondence
взаємно-однозначна ~ one-to-one correspondence
однозначна ~ univocal correspondence
відсікання – cutoff
ВІС (велика інтегральна схема) – large-scale integrated (integration) circuit
вісімковий – octal, octonary

вісь – axis
~ координат – datum line
часова ~ time base
вольт-амперна характеристика – current-voltage characteristic (curve)
встановлення – setting
~ в нуль – zero setting
вузол – assembly, node, portion, unit
~ керування – control assembly (portion, unit)
функціональний ~ (блок) – functional assembly
функціональний ~ комбінаційного типу – logic macro circuit
головний ~ master node
основний ~ reference node
підпорядкований ~ slave node
арифметичний ~ arithmetic portion (unit)
вхід – input
~ встановлення в стан “0” – reset input
~ встановлення в стан “1” – set input
~ для лічби у зворотному напрямку (лічильника) – countdown input
~ керування – select input
~ переносу (вхідний перенос) – carry input
~ синхронізації – clock input
інверсний ~ inverted input
інформаційний ~ data input
лічильний ~ (тригера) – complementing input
прямий ~ straight input

Г

галій – gallium

генератор – generator
 ~ імпульсів – pulser
 ~ синхроімпульсів – clock
генерування – generation
 ~ імпульсів – pulsing
генерувати – generate
германій – germanium
 ~ *n*-типу – *n*(-type) germanium
 ~ *p*-типу – *p*(-type) germanium
гістограма – bar chart, bar graph
граф – graph
 ~ без циклів – circuit-free graph
 ~ з петлями – graph with loops
 ~ переходів – transition graph
 ~ станів – state graph
 повний ~ complete graph

Д

дані – data, information
 аналогові ~ analog data
 буквено-цифрові ~
 alpha-numeric(al) data
 вихідні ~ output data, data-out
 вхідні ~ input data, data-in
 двійкові ~ binary data
 десяткові ~ decimal data
 дискретні ~ discrete data
 (за)кодовані ~ coded data
 накопичені ~ cumulative data
 рядок даних – bit string data
 цифрові ~ digital(ized),
 numeric(al) data
 числові ~ numerical data
двійковий – binary, dyadic
двійково-десятковий – binary
 coded decimal
двоїстий, дуальний – dual
двоїстість – duality
 принцип двоїстості – principle
 of duality
ДДФ – full disjunctive normal
 form

декомпозиція, розкладання –
 decomposition
декремент – decrement
декрементний – decremental
дешифратор, декодер – decoder,
 decoding circuit
 двійковий ~ binary decoder
 ~ адреси – address decoder
 ~ з прямими виходами –
 decoder with straight outputs
 ~ з інверсними виходами –
 decoder with inverted outputs
 ~ команд – operation decoder
дешифрування, декодування –
 decoding
диз'юнктивна форма –
 disjunctive form
диз'юнкція – disjunction
 ~ заперечень – dispersion,
 alternative denial
дискретний – discrete, digital
дискретизація – discretization
дистрибутивність – distribution
дистрибутивний – distributional,
 distributive
 ~ (розподільний) закон –
 distributive law
діаграма – chart, diagram, graph
 ~ Вейча – Veitch chart (diagram,
 map)
 ~ станів – state diagram
 кругова ~ circular graph
 часова ~ time chart, timing
діапазон – range
 ~ (період) лічби – counter range
 ~ чисел – number range
ділення – division
діод – diode
 ~ Шоткі – Shottky diode
 еквівалент діода – dummy
 diode
ДФНФ – full conjunctive normal
 form

доведення – proof
доповнення – complementation
доповнювати – to complement

Е

еквівалент – equivalent
еквівалентність – equivalence
логічна ~ logical equivalence
електрод – electrode, contact
~ заслону – gate electrode
емітерний ~ emitter contact
колекторний ~ collector contact
електрон – electron
вільний ~ free electron
надлишковий ~ excess electron
електроніка – electronics,
electronic engineering
інтегральна ~ integrated
electronics
транзисторна ~ transistor
electronics
електронний – electronic
елемент – element, device, cell,
unit, gate, primitive
бістабільний ~ bistable device
(unit)
вихідний ~ output element
вхідний ~ input element
~ АБО – OR element, OR gate
~ АБО-НЕ – NOR (NOT OR)
element, NOR gate
~ виключне АБО –
exclusive OR element
~ даних – data item
~ заперечення – negator
~ затримки – delay device
(element), time-delay element
~ I – AND element, AND gate
~ I-HE – NAND (NOT AND)
element, NAND gate
~ НЕ – negator, negator,
NOT element

~ пам'яті – storage (memory)
cell (device), memory element
запам'ятовувальний ~
storage element
інвертувальний ~ inverter,
inverter, inverting element
логічний ~ logic(al) element,
logic gate
напівпровідниковий ~
semiconductor device
елементний – elemental
емітер – emitter

Є

ємність – size
~ пам'яті – memory (storage)
size

Ж

живлення – power supply
напруга живлення +5В –
a 5 volt power supply
напруга живлення –5.2В –
a minus 5.2 volt power supply

З

завдання – definition, job
задача – problem, job, task
~ оброблення даних – data-
processing problem
~ про комівояжера – traveling
salesman job
~ розділена на частини –
divided job
логічна ~ logical problem
наукова ~ scientific job
обчислювальна ~ computational
problem
постановка, формулювання
задачі – problem definition
поточна ~ current task
технічна ~ problem of
engineering

тривимірна ~ three-dimensional
 problem
 формулювати задачу – to
 define a problem
заземлення – ground connection,
 ground, grounding, earth
 заземлений – earthed
закон – law, principle
 ~ асоціативності – associative
 law
 ~ дистрибутивності –
 distributive law
 ~ дистрибутивності диз'юнкції
 щодо кон'юнкції – distributive
 law disjunction of over
 conjunction
 ~ дистрибутивності кон'юнкції
 щодо диз'юнкції – distributive
 law of conjunction over
 disjunction
 ~ комутативності –
 commutative law
 ~ поглинання – law of
 absorption
 ~ подвійного заперечення –
 law of double negation
 ~ (правило) де-Моргана – law
 of De Morgan (DeMorgan's law)
 ~ тавтології (повторення) – law
 of tautology
закривання – cutoff
заміна – replacement
заміщення – substitution
запам'ятовувальний пристрій –
 memory, storage
запам'ятовування – storage,
 storing
запам'ятовувати (зберігати) –
 memorize, store
заперечення – negation, denial,
 complement
 виконувати операцію
 заперечення – negate

диз'юнкція заперечень –
 alternative denial
 ~ диз'юнкції – nondisjunction,
 NOR
 ~ еквівалентності –
 nonequivalence
 ~ заперечення x (x із
 запереченням) – the
 complement of x
 ~ змінної – the complement of
 a variable
 ~ імплікації – nonimplication
 ~ кон'юнкції – nonconjunction,
 NAND
 із запереченням – negated
 кон'юнкція заперечень –
 joint denial
 правило (закон) подвійного ~
 the involution law (the law of
 double complementation)
запит – demand, request
 ~ на обслуговування – service
 demand
 ~ на переривання – interrupt
 request
заслін – gate
 ізольований ~ insulated
 (isolated) gate
засоби – facility, mean
 апаратні ~ hardware
 ~ введення-виведення –
 input/output facilities
 ~ зв'язку – communication
 facilities
 ~ керування – controlling means
 обчислювальні ~ computing
 facilities
 програмні ~ software
затискач – clamp
затримка – delay, retardation
 ~ в часі – time lag
 затримувальний елемент –
 delayer

часова ~ time delay
затримувати – to delay, retard,
 suspend
зациклювання – cycling
зберігання (інформації) –
 holding, store, storing
згортка – convolution
змінна – variable
 булева (логічна) ~ Boolean
 (logical) variable
 вхідна ~ input variable
 двійкова ~ binary variable
 дискретна ~ digital variable
 інвертована ~ (~ з інверсією) –
 complemented variable
 пряма ~ true variable
значення – value, meaning,
 significance
 абсолютне ~ absolute value
 граничне ~ limiting value
 довільне ~ (0 або 1, тобто \times) –
 don't care
 допустиме ~ legitimate value
 заборонене ~ (прочерк, “–”) –
 dash
 задане ~ set value
 ~ числа – number magnitude
 істинне ~ truth value
 кінцеве ~ final value
 логічне ~ logical value
 максимальне ~ maximal value
 мінімальне ~ minimal value
 обчислене ~ computed value
 порогове ~ threshold value
 послідовні ~ successive value
 поточне ~ current value
 початкове ~ initial (reference)
 value
 середнє ~ average value
 фіксоване ~ fixed value
зсув – shift
 арифметичний ~ arithmetic(al)
 shift

~ вліво – left shift
 ~ вправо – right shift
 ~ на декілька розрядів –
 multiple-position shift
 ~ на одну позицію (на один
 розряд) – shift of one position,
 single-place shift
 логічний ~ logic(al) shift
 циклічний ~ cyclic shift,
 ring shift
зсувний пристрій, зсувач – shifter
зсувний регістр – shifter

I

ідемпотентність – idempotency
 правило (закон)
 ідемпотентності – idempotent
 law
ієрархія – hierarchy
імпліканта – implicant
 істотна ~ essential prime
 implicant
 проста ~ prime implicant
імплікантна матриця (таблиця
 покриття) – prime implicant chart
імплікація – implication
імплікувати – to imply
інверсія – inverse, inversion
 ~ вхідних сигналів – input
 inversion
 ~ змінних – inverse of variables
інверсний вхід – negative input
інвертувати – invert, negate,
 complement
інвертор – inverter, invertor,
 negator, negater
індикатор – indicator
 цифровий ~ digital (numerical)
 indicator
індій – indium
інжекція – injection
 ~ носіїв – carrier injection
 ~ дірок – hole injection

інкремент – increment
інкрементний – incremental
інформація – data, information
 вхідна ~ input (source) information
 двійкова ~ binary information
 двійково-кодована ~ binary coded information
 цифрова ~ digital (numerical) information
 кількість інформації – quantity of information
 структура інформації – information lattice
інтервал – interval, range, space
 часовий ~ time interval
існування – existence
 ~ і єдиність – unique existence
I-HE – NAND (Not AND)

К

канонічний – canonical
карта Карно – Karnaugh map
 ~ для 4-х змінних – Karnaugh map for four variables (a 4-variable Karnaugh map)
каскад – cascade
 об'єднувати в ~ to connect in cascade
квантування – quantization
 крок ~ quantum
квантувати – quantize
кварц – quartz
кілобіт – kilobit
кількість – quantity
 ~ інформації – quantity of information
код – code
 8-розрядний ~ 8-bit code
 k-розрядний ~ k-unit code
 двійковий ~ natural (straight) binary code
 двійково-кодований

десятковий ~ (код “8-4-2-1”) – binary-coded decimal code (BCD code)
 десятковий ~ decimal code
 десятково-п'ятірковий ~ biquinary code
 інвертований (інверсний) ~ inverted code
 ~ “два з п'яти” – two-out-of-five code
 ~ Грея – Gray code
 ~ керування – control code
 ~ команди – instruction code
 ~ операції – operation code
 ~ символа – symbol code
 ~ стану – state code
 ~ числа – number code
 п'ятірково-двійковий ~ quibinary code
 паралельний ~ parallel code
 позиційний ~ position code, weight code
 послідовний ~ serial code
 прямий ~ straight binary code
 рефлексний ~ reflected code
 трійковий ~ ternary code
 шістнадцятковий ~ hexadecimal code
кодування – coding, encoding
 двійкове ~ binary coding
 ~ інформації – information encoding
 ~ чисел – number coding
 правило ~ code rule
кодер – coder, encoding device, encoder
кодувати (шифрувати) – encode, code
коефіцієнт – factor, ratio
 безрозмірний ~ dimensionless factor
 змінний ~ float(ing) factor

~ зворотного зв'язку –
feedback factor

~ об'єднання по входу – fan-in

~ підсилення – gain
(amplification) factor

~ підсилення за струмом –
current amplification

~ розгалуження по виходу –
fan-out

~ складності – complexity
factor

масштабний ~ scale factor

КОЛО – circuit

замкнене ~ closed (made) circuit

~ зворотного зв'язку –
feedback loop

~ переносу – carry circuit

розімкнене ~ open circuit

КОМАНДА – instruction, command,
order

~ виклику (підпрограми) – call
instruction

~ передачі керування – control
transfer instruction

~ переходу – jump instruction

~ повернення – return
instruction

~ програми – program
instruction

~ умовного переходу –
conditional jump instruction

логічна ~ logic(al) instruction

поточна ~ current instruction

КОМБІНАЦІЯ – combination

заборонена ~ forbidden
combination

кодова ~ code combination

КОМБІНУВАТИ – combine

КОМПАРАТОР – comparator

КОМПЛЕКТ – kit

КОМП'ЮТЕР – computer

бортовий ~ airborne computer

КОМУТАТИВНИЙ – commutative
~ (переставний) закон –
commutative law

КОМУТАТИВНІСТЬ – commutation

КОМУТАТОР – commutator

КОМУТУВАТИ – commute

КОНСТИТУЕНТА – constituent
~ нуля – constituent of zero
~ одиниці – constituent of unity

КОНТУР ЗВОРОТНОГО ЗВ'ЯЗКУ –
feedback loop

КОН'ЮНКТИВНА ФОРМА –
conjunctive form

КОН'ЮНКЦІЯ – conjunction
~ заперечень – joint denial

КРАТНЕ (число) – multiple

КРЕМНІЙ – silicon
полікристалічний ~
polycrystalline silicon

КРЕСЛЕННЯ (дія) – plotting

КРИВА – curve, plot
гіпербола – hyperbolic curve
експоненційна ~ (експонента)
– exponential curve
логарифмічна ~ logarithmic
curve

синусоїда – harmonic curve

КРИСТАЛ – crystal, chip, dice
безкорпусний ~ unpackaged
chip

кремнієвий ~ silicon chip

Л

ЛАПКИ – quotes, quotation marks

ЛІЧБА – count, counting
двійкова ~ binary count
двійково-десятькова ~ binary-
coded decimal (BCD) count
~ у зворотному напрямі – to
count down, countdown,
counting-down, counting in
reverse

~ у прямому напрямі – to count forward(up), counting forward
 послідовність лічби (лічбова послідовність) ~ counting sequence
лічильник – counter
 двійковий ~ binary (radix two, scale-of-two) counter
 двійково-десятковий ~ decimal (binary decade, decade, scale-of-ten) counter, modulo-10 count-up counter
 декрементний ~ subtract (down) counter, count-down counter
 інкрементний ~ summary counter, count-up counter
 кільцевий ~ ring counter
 ~ адреси – address counter
 ~ Джонсона – Johnson counter
 ~ з довільним порядком лічби – nonsequential counter
 ~ за модулем N – modulo N (scale-of- N) counter
 ~ за модулем два – binary stage
 ~, що генерує код Грея – Gray code counter
 однонапрямний ~ unidirectional counter
 початковий стан лічильника – initial count
 програмний ~ program (instruction) counter
 реверсивний (двонапрямний) ~ bidirectional (forward-backward, reversible, up-down) counter
 трійковий ~ ternary counter
логіка – logic
 булева ~ (алгебра) – Boolean logic
 двійкова ~ binary logic

двозначна ~ two-valued logic
 комп'ютерна ~ computer logic
 ~ з емітерними зв'язками (ЕЗЛ) – emitter-coupled logic (ECL)
 програмовна ~ programmable logic
 транзисторно-транзисторна ~ (ТТЛ) – transistor-transistor logic (TTL)
логічна одиниця – logical 1 (true)
логічний – logical
логічний елемент – gate
логічний нуль – logical 0 (false)
логічні схеми ~ logical circuitry

М

максимальний – maximal
максимум, максимальне значення – maximum
макстерм – maxterm
масштаб – scale
 вибір масштабу (масштабування) – scaling
 збільшувати ~ to scale up
 зменшувати ~ to scale down
 лінійний ~ linear scale
 логарифмічний ~ logarithmic scale
 ~ часу – time scale
 масштабна (координатна) сітка – coordinate scale
МДНФ – minimized disjunctive normal form
мегабіт – megabit
мегагерц – megahertz
метод – approach, method, technique
 евристичний ~ heuristic approach
 емпіричний ~ cut-and-try method
 ~ вирішення задачі – approach

напруга – voltage
 вихідна ~ output voltage
 вхідна ~ input voltage
 ~ живлення – power-supply voltage
 ~ заслону – gate voltage
 ~ зміщення – bias voltage
 ~ колектора – collector voltage
 ~ пробою – breakdown voltage
напруженість – intensity, strength, density
 ~ магнітного поля – magnetic field intensity (strength)
 ~ поля – density of field
насичення – saturation
 ~ транзистора – transistor saturation

НЕ – NOT

невідоме – unknown
нееквівалентність – nonequivalence
нелінійний – nonlinear
ненульовий – nonzero
непарний – odd
непарність – oddness
неперервний – continuous
неперервність – continuity
нерівність – inequality, odds
нескінченний – infinite
нескінченність – infinity
нуль (нульовий) – zero
 встановлювати в ~ to set to zero
 двійковий ~ binary zero

O

обладнання – equipment, facility, outfit, technique, hardware
 аналогове ~ analog equipment
 допоміжне ~ support equipment
 ~ для аналогово-цифрового перетворення – analog-digital equipment

~ для оброблення даних – data-processing equipment
 ~ для перетворення даних – conversion equipment
 основне ~ primary equipment
 периферійне (зовнішнє) ~ peripheral equipment
 резервне ~ standby equipment
область – domain, range, zone
 активна ~ active region
 збіднена ~ depletion region
 ~ бази – base region (zone)
 ~ визначення – range of definition
 ~ визначення функції – domain of function
 ~ діркової провідності (*p*-область) – *p*-region
 ~ електронної провідності (*n*-область) – *n*-region
 ~ емітера – emitter region (zone)
 ~ значень змінної – variable range
 ~ колектора – collector region (zone)
 ~ переходу – barrier (transition, junction) region
 ~ пробою – breakdown region
обмежений – limited, restricted
обмеження – constraint, limitation, limiting, restraint, restriction
 ~ потужності – power limiting
обмежувач – limiter
 діодний ~ diode limiter
оброблення – handling, processing
 ~ переривань – interrupt handling
 ~ інформації – information processing
 ~ даних – computing, data handling, data processing
 дистанційне ~ даних – remote computing

обсяг – content, size
 ~ інформації (кількість інформації) – information content
 ~ блока (даних) – block-size
 ~ (довжина) програми – program size

обчислений – computed

обчислення – calculation, calculus, computing, computation
 ~ чисельними методами – numerical computation

обчислювальна машина – computer, computer

обчислювальна техніка – computer technique

обчислювальний – calculating, computational
 ~ пристрій – calculating apparatus

обчислювальні засоби – computing facilities

обчислюваний – computable

обчислювати – calculate, compute

обчислювач – calculator, computer

об'єм – size

об'єкт – object

об'єднання – join, merging, packing, union
 ~ масивів даних – merging
 ~ в блоки (слів, чисел) – blocking

одиниця (один) – one

одиниця – unity
 двійкова ~ binary one

одновимірний – univariate, one-dimensional

одноадресний – one-address

одновимірний – one-dimensional, single-dimensional

однозначний – one-valued, single-valued, univocal
 взаємно ~ one-one, one-for-one

однозначність – univocacy

однонапрямний – unidirectional, one-way

однорозрядний – single-digit

ознака – flag
 ~ парності – parity flag

операнд – operand

оперативний
 запам'ятовувальний пристрій (ОЗП) – random access memory (RAM)

оператор – operator, operation, clause, statement
 ~ логічного елемента – logic(al) element operation
 логічний ~ logical operator
 умовний ~ if-clause
 ~ присвоювання – replacement operator
 тіло оператора – statement body
 ~ І – AND-operator (operation)
 ~ АБО – OR-operator (operation)
 ~ ІНАКШЕ – else-clause
 операторний – operational

операція – operation, operator, action
 арифметична ~ arithmetic operation
 булева (логічна) ~ Boolean operation
 двійкова ~ binary operation
 бінарна ~ dyadic operation
 допоміжна ~ auxiliary operation
 ~ І – AND-operation
 ~ АБО – OR-operation
 ~ Виключне АБО – Exclusive OR operation, exjunction
 ~ еквівалентності – biconditional operation
 ~ імплікації – implication operation

~ заперечення – complementary operation
 ~ порівняння – comparison operation
 ~ розгалуження (умовний перехід) – branch operation
 ~ переходу – jump operation
 ~ лічби – counting operation
 порозрядна ~ digit-to-digit operation
 складна ~ complex operation
 логічна ~ logical operation
 ~ запису – write operation
 ~ зчитування – read operation
 ~ зсуву – shift operation
 елементарна ~ elementary operation
 унарна (одномісна) ~ unary operation
оптимізація – optimization
основа системи числення – radix number
оцінювання – evaluation
 чисельне ~ numerical evaluation
оцінювати – evaluate

П

пам'ять – memory, storage, store
 буферна ~ cache memory
 оперативна ~ random-access memory (RAM)
 ~ з можливістю стирання запису – erasable memory
 ~ із збереженням заряду – charge-storage memory
 ~ на МОН (метал-окисел-напівпровідник)-схемах (транзисторах) – metal-oxide-semiconductor memory (MOS-memory)
 ~ напівпровідникова – semiconductor memory

постійна ~ read-only memory (ROM)
 стекова (магазинна) ~ stack memory
параметр – parameter
 початковий ~ initial parameter
 формальний ~ formal parameter
парний – even
парність – evenness, parity
 контроль на ~ parity check
передавати – transmit
передавач – transmitter
передача – transmission, transmit, transfer
 однонапрямна ~ unidirectional transfer
 ~ даних (інформації) – information transfer
 ~ керування (команда) – transfer of control
 ~ між регістрами – register-to-register transfer
 ~ потоку даних – stream transmission
 побітова ~ bit (-by-bit) transfer
 послідовна ~ serial transfer, serial transmission
 синхронна ~ synchronous transmission
перемикання – commutation
перемикати – commute
перемикач – switch, commutator
перепад – drop, swing
 ~ напруги – voltage drop (swing)
переповнення – overflow
перепрограмування – reprogramming
переривання – interrupt, interruption
 автоматичне ~ automatic interrupt
 внутрішнє ~ internal interrupt

зовнішнє ~ external interrupt
 неперіоритетне ~ nonpriority interrupt
 ~ від процесора – processor interrupt
 періоритетне ~ priority interrupt
перетворення – conversion, transformation
 алгебраїчне ~ algebraic transformation
 аналогово-цифрове ~ analog-to-digital conversion
 зворотне (обернене) ~ reconversion, inverse transformation
 зворотне (обернене) ~ Фур'є – Fourier inversion
 лінійне ~ linear transformation
 логічне ~ logical conversion
 математичне ~ transformation
 ~ з двійкової у десяткову систему числення – binary-to-decimal conversion
 ~ коду – code conversion
 ~ координат – coordinate transformation
 ~ Лапласа – Laplace transformation
 ~ одного коду в інший – data translating
 ~ формули – formula translation
 ~ Фур'є – Fourier transformation
 тригонометричне ~ angular transformation
перетворювати – converse, convert, transcribe, transform
перетворювач – code converter, converter, coder, reducer, transducer
 автоматичний перетворювач даних ~ automatic data reducer
 аналогово-цифровий ~ analog-digital converter

~ даних – data transducer
 ~ коду – code converter, code translator
 ~ кутового положення (валу) в код (перетворювач кут-код) – angle-to-digit(al) converter
 цифровий ~ digitizer
перенос – carry
 двійковий ~ binary carry
 прискорений ~ look-ahead carry
перехід – jump, branch, transfer, transition
 безумовний ~ unconditional jump (branch)
 команда переходу (передачі керування) ~ transfer
 ~ база-емітер – base-emitter diode
 ~ до підпрограми – subroutine jump
 ~ при переповненні – jump (branch) on overflow
 умовний ~ conditional jump (branch on)
період – period, cycle, date
 ~ затримки – delay period
 ~ слідування імпульсів – pulse period
 ~ тактових сигналів (імпульсів) – clock cycle
періодичний – cyclic
петля – loop
 замкнута ~ closed loop
 ~ (контур) зворотного зв'язку – feedback loop
підкладка – substrate, wafer, base, sublayer
 багатошарова ~ multilayer substrate
 ізоляційна ~ insulating substrate
 керамічна ~ ceramic substrate
 кремнієва ~ silicon substrate
 ~ типу *n*- – *n*-substrate

~ типу *p*- – *p*-substrate
 полікристалічна ~
 polycrystalline substrate
 скляна ~ glass substrate
підмножина – subset
 ~ символів – character subset
підпрограма – subroutine,
 subprogram(me)
підраховувати – to figure out (up)
підсилення – amplification
підсилювати – amplify
підсилювач – amplifier
 ~ постійного струму – direct-
 current amplifier
 транзисторний ~
 transistor(ized) amplifier
плавлення, плавкий – melting
пластина – wafer
 безкорпусна ~ unpackaged
 wafer
 кремнієва ~ silicon backing
 ~ кристала – crystal wafer
плата – board
 багатошарова ~ multilayer
 board, multilayer printed circuit
 двостороння ~ double-sided
 board
 материнська (об'єднувальна) ~
 mother board
 монтажна ~ wire (wiring) board
 одностороння ~ single-sided
 board
 ~ з чіпами – chip board
повний – exhaustive, complete
повнота – exhaustiveness,
 completeness
 абсолютна ~ absolute
 completeness
 функціональна ~ functional
 completeness
повторювач – noninverting
 amplifier, follower, repeater
поглинання – absorption

правило (закон) ~ absorption
 law
подання (числа) – notation
позначення – denotation,
 designation, symbol, label, marking,
 lettering
 буквене ~ lettering
 позначення інверсного входу,
 виходу (кружечок) – wedge
 symbol (a small circle)
 символьне ~ symbolic
 designation
 умовне графічне ~ (УГП) –
 logic symbol
показчик – designator, indicator,
 pointer
 ~ стека – stack pointer
 ~ функції – function designator
покриття – covering
 ~ функції – covering the
 function
поліном – multinomial, polynomial
полярність – polarity
порівнювати – compare
порівняльність – comparability
порівняння – comparison
 порівнювальний пристрій –
 comparator
 порозрядне ~ logical
 comparison
потенціал – potential
 високий ~ high potential
 низький ~ low potential
 нульовий ~ ground potential
 різниця потенціалів – voltage
початок – origin
 ~ відліку часу – time origin
 ~ координат – origin of
 coordinates
 ~ програми – origin (ORG)
поширення (сигналу) –
 propagation

~ між імпульсами – pulse spacing
 ~ часу – time interval
 частотний інтервал – frequency spacing
простір – space
 векторний ~ vector space
 N -вимірний ~ N -space
процедура – procedure
 ~ доведення – validation procedure
процес – process, transient
 адитивний ~ additive process
 алгоритмічний ~ algorithmic process
 дискретний ~ discrete process
 неперервний ~ continuous process
 перехідний ~ turn-on transient
 ~ перемикання – switching process
 циклічний ~ cyclic process
процесор – processor
 ~ периферійного пристрою – peripheral processor
 центральний ~ central processor (unit)
прямий код (звичайна форма числа) – true form
п'ятірковий – quinary
п'ятнадцятковий – quindenary

Р

реалізація булевих функцій – the generation of Boolean functions
реалізувати булеву функцію – to generate of Boolean function
регістр – register, shifter
 ~ адреси – address register
 ~ зсуву (зсувний ~) – shift(ing) register, shifter
 ~ команд – instruction register

~ накопичувального типу – accumulator register
регістровий
запам'ятовувальний пристрій (РЗП) – register file
рекурсивний – recursive
рекурсія – recursion
рівень – level
 активний ~ active level
 високий ~ high level
 низький ~ low level
 нульовий ~ zero level
 ~ сигналу – signal level
рівний (дорівнювати) – equal
рівність – equality
рівнозначність – biconditional implication
рівняння – equation
 ~ збудження – excitation equation
 ~ збудження JK-тригера – excitation equation for JK flip-flop
різниця – difference
 ~ потенціалів – potential difference, voltage
розбиття – dissection, partition, partitioning, splitting
розгалуження – fork (алгоритму), branching (програми)
 точка ~ branchpoint
розділювання – separation
 ~ даних – data separation
розділювати – separate
розділювач – separator
розкладання – decomposition, expansion (ряд), factoring, factorization (на множники)
 ~ в ряд – expansion in series (series expansion)
 ~ в ряд Фур'є – harmonic expansion

розряд – location, position, digit
 двійковий ~ bit
 десятковий ~ decimal location,
 decade
 ~ парності – parity digit
розрядність (слова) – digits per
 word
розширювач – stretcher
ряд – series
 знакозмінний ~ alternate series
 ~ Фур'є – Fourier series
 степеневий ~ power(-low) series
 часовий ~ time series

С

селектор – selector
 ~ даних – data selector
серія імпульсів – pulse group
сітка координатна – coordinate
 scale
сигнал – signal
 аналоговий ~ analog signal
 вихідний ~ output signal
 вхідний ~ input signal
 дискретний (цифровий) ~
 digital signal, discrete signal
 квантовий ~ quantized signal
 кодований ~ coded signal
 потенціальний ~ steady-state
 signal
 ~ записування – write signal
 ~ запиту – request signal
 ~ зворотного зв'язку –
 feedback (return) signal
 ~ зсуву вліво – shift-left signal
 ~ зсуву вправо – shift-right
 signal
 ~ зчитування (читання) –
 reading signal
 ~ переносу – carry signal
 ~ переривання – interrupt signal
 ~ синхронізації (синхросигнал)
 – clock (synchronizing) signal

синтез – design, synthesis
 логічний ~ logic(al) design
 ~ комбінаційної схеми –
 synthesis of combinational logic
 ~ схеми – circuit design
 ~ функціональної схеми –
 functional design
синхронізація – timing, clock,
 clocking
система – set, system, scheme
 ~ аксіом – axiom scheme
 ~ кодування – coding
 ~ команд – instruction set, order
 set
 ~ правил – set of rules
 ~ рівнянь – equations set
система числення – notation,
 numerical notation, scale of notation
 двійкова ~ binary notation
 двійково-кодована десяткова ~
 binary-coded decimal notation
 позиційна ~ positional (radix)
 notation, radix scale of notation
 трійкова ~ ternary notation
складання блок-схеми
 (алгоритму) – flow-charting
складний – complex
складність – complexity
 апаратна ~ logic diagram
 complexity,
 hardware complexity,
спосіб – algorithm
 ~ Петріка – Petrick algorithm
стан – state
 інверсний ~ inverted state
 кінцевий ~ finite state
 наступний ~ next state
 нестійкий ~ unstable state
 поточний ~ present state
 початковий ~ initial (reset) state
 проміжний ~ intermediate state
 ~ “вимкнено” – off state
 ~ “нуль” – zero state, reset state

~ “одиниця” – one state, set
state

стійкий ~ steady (stable) state

стек – stack
“дно” стека – bottom of stack
показчик стека – stack pointer
~ (магазин) команд –
instruction stack
~ буферного типу (FIFO) –
first-in, first-out
~ магазинного типу (LIFO) –
last-in, first-out

стік (в транзисторі) – drain

структура – construction, lattice,
organization, structure
~ інформації – information
lattice
~ центрального процесора –
central processor organization
блокова ~ unit construction
кристалічна ~ (решітка) –
crystal lattice

структурна схема – block diagram
~ лічильника – counter block
diagram

струм – current
змінний ~ alternating
current (AC)
постійний ~ direct current (DC)
~ бази – base current
~ зміщення – bias current
~ пробою – breakdown current

сукупність – universe

суматор – adder, accumulator
двійковий ~ binary adder
накопичувальний ~ adder-
accumulator
напівсуматор – half-adder
однорозрядний ~ full (on-digit)
adder
паралельний ~ parallel adder
(accumulator)

суперпозиція – superposition

схема – circuit, diagram, chart,
scheme

аналогова ~ analog circuit

багатошарова схема –
multilayer

бістабільна ~ bistable circuit,
latch, storage device

блок-~ flow chart, block
diagram

буферна ~ buffer circuit

велика інтегральна ~ complex
integrated circuit, large-scale
integration circuit (LSI)

вентильна ~ gate circuit

комбінаційна ~ combinational
logic, logic diagram

логічна ~ logic diagram, logical
switch, logic chart, logical
structure

МОН (метал-окисел-
напівпровідник)-~
MOS (metal-oxide-
semiconductor) circuit

підсумовувальна ~ add(ing)
circuit

послідовнісна ~ sequential logic

послідовнісна ~ sequential logic

принципова ~ circuit diagram,
schematic diagram

структурна ~ block diagram,
structure diagram

(блок) ~ процесу – procedure
chart

~ десяткового лічильника –
logic diagram for the modulo-10
counter

~ доведення – proof scheme

~ затримки – delay circuit

~ зі спільним емітером –
common-emitter circuit

~ зі спільним колектором –
common-collector circuit

~ I – AND circuit, coincidence circuit
 ~ керування – control circuit
 ~ лічильника – counter circuit
 ~ лічильника – counter's logic diagram
 ~ на біполярних транзисторах – bipolar circuit
 ~ порівняння – comparator (comparison) circuit
 ~ синхронізації – clock, timer
 тригерна ~ flip-flop circuit, trigger, trigger circuit, latch circuit
 функціональна ~ functional (block) diagram, functional arrangement, function circuit (scheme)

Т

таблиця – table, chart
 складати таблицю – tabulate
 ~ виходів – output table
 ~ істинності – truth table
 ~ переходів (автомата) – transition table, jump table
 ~ переходів (тригера) – functional characteristic table, state transition table
 ~ переходів JK-тригера – JK functional characteristic table
 ~ покриття функції – prime implicant chart
 ~ станів лічильника – next-state table for a counter
 ~ функції – function table
 ~ функцій збудження – excitation table
 ~ функцій збудження JK-тригера – excitation table for a JK flip-flop
тавтологічний – tautological
тавтологія – tautology

правило (закон) тавтології – the law of tautology
тактова частота – clock speed
тактові (тактувальні) імпульси – clock
тактування – clocking
ТДНФ – irredundant disjunctive normal form
теорема – law, theorem
 доводити теорему – to prove the theorem
 обернена ~ reciprocal theorem
 ~ двоїстості (дуальності) – duality theorem
 ~ де Моргана – De Morgan's theorem
 ~ Квайна – Quine theorem
 ~ синусів – law of sines
 ~ Шеннона – Shannon theorem
теорія – theory
 абстрактна ~ автоматів – abstract theory of automata
 ~ алгоритмів – theory of algorithm
 ~ інформації – information theory
 ~ ймовірностей – probability theory
 ~ кодування – coding theory
 ~ множин – set theory
 ~ перемикальних функцій – switching theory
терм – term
 булів ~ Boolean term
 диз'юнктивний ~ sum term
 кон'юнктивний ~ product term
 склеювання термів – combining the terms
тетрада – tetrad
топологія (інтегральної схеми) – layout

точка – point
~ зупинення (програми) –
breakpoint

~ розгалуження – branchpoint

точність – accuracy, precision
задана ~ given accuracy
необхідна ~ desired precision
одинарна ~ single precision
подвійна ~ double precision
потрібна (достатня) ~ adequate
accuracy

транзистор – transistor
n-p-n ~ *n-p-n* transistor
p-n-p ~ *p-n-p* transistor
багатоемітерний ~ multiple
emitter transistor

безкорпусний ~ chip transistor

біполярний ~ bipolar transistor

МДН(метал-діелектрик-
напівпровідник)-~ MIS

(metal-insulator-
semiconductor) transistor

МОН (метал-окисел-
напівпровідник)-~

MOS (metal-oxide-
semiconductor) transistor

МОН-~ з каналом *n*-типу –
n-channel MOS-transistor

площинний ~ junction transistor
~ з високим коефіцієнтом

підсилення – high-gain
transistor

уніполярний (польовий) ~
field(-effect) transistor

уніполярний (польовий) ~
з ізольованим затвором –
insulated-gate field(-effect)
transistor (unipolar
transistor)

тривалість – duration
~ (ширина) імпульса – pulse
duration

тригер – flip-flop, trigger, toggle,
bistable storage device

D-~ D flip-flop

JK-~ JK flip-flop

динамічний ~ dynamic flip-flop

перехід тригера з стану
“1” у стан “0” – flip-flop
changing from “1” to “0”

~ з лічбовим входом –
complementing flip-flop

У

**умовне графічне
позначення (УГП) –**
logic symbol, symbolic
designation, symbol general block
diagram

~ елемента АБО – logic symbol
for OR gate

~ компаратора – general block
diagram of a comparator

~ мультиплектора –symbol for
multiplexer

уніполярний – unipolar

уніфікований – unitized

ущільнення – packing, squeeze,
compression

Ф

фаза – phase

файл – file

виділяти ~ to mark a file

відкрити ~ to open a file

відновлювати ~ to reset a file

допоміжний ~ optional file

закрити ~ to close a file

основний ~ master file

створити ~ to create a file

~ (масив) даних – data file

~ з високим пріоритетом –
privileged file

фактор – factor
визначальний ~ determining factor

фіксатор – clasper

фіксування – clamping, hold

форма – form

аналітична ~ analytic form

аналогова ~ analog form

диз'юнктивна ~ sum of products

диз'юнктивна нормальна ~
(ДНФ) – disjunctive normal form, sum-of-mintermsform, a canonical sum of products, a standard sum of products, minterm expansion

дискретна (цифрова) ~ digital form

досконала диз'юнктивна нормальна ~ (ДДНФ) – full disjunctive normal form

досконала кон'юнктивна нормальна ~ (ДКНФ) – full conjunctive normal form

закодована цифрова ~ coded digital form

канонічна ~ canonical form

кон'юнктивна ~ product of sums

кон'юнктивна нормальна ~ (КНФ) – conjunctive normal form, product-of-maxtermsform, a canonical product of sums, a standard product of sums, maxterm expansion

мінімальна диз'юнктивна нормальна ~ (МДНФ) – minimized disjunctive normal form

мінімальна кон'юнктивна нормальна (МКНФ) ~ minimized conjunctive normal form

нормальна ~ normal form

ТДНФ – irredundant disjunctive normal form

тупикова ~ irredundant form

~ двійкового сигналу – binary waveform

~ сигналу – waveform

формат – format

~ команди – instruction format

фронт – edge

задній ~ trailing edge

передній ~ leading edge

функція – function

булева ~ Boolean connective (function)

елементарна ~ elementary function

логічна ~ logical connective (function)

неперервна ~ continuous function

перемикальна ~ switching function

повністю визначена ~ completely defined function

реалізувати функцію ~ to generate the function

рекурсивна ~ recursive function

складна ~ composite function

~ (стрілка) Пірса – Peirce function, dagger function

~ (штрих) Шеффера – Sheffer (stroke) function

~ АБО – OR connective (function)

~ виключне АБО – XOR function

~ виходів – output function

~ задана таблицею істинності –
truth-table function
~ I – AND connective (function)
~ логічного додавання – logical
addition function
~ логічного множення – logical
multiplication function
~ НЕ – NOT function
~ переходів – next-state
function

Х

характеристика – characteristic,
curve, performance, response
амплітудно-фазова ~
gain-phase characteristic
амплітудно-частотна ~
amplitude versus frequency
response, gain frequency
characteristic
вольт-амперна ~ current-voltage
(voltage-current) characteristic
(curve)
динамічна ~ dynamic response
імпульсна ~ pulse response
~ зворотного зв'язку –
backward transfer characteristic
~ перемикання – switching
performance
~ системи – system performance
часова ~ time response
частотна ~ frequency
characteristic (response)

характеристичний – characteristic

Ц

цикл – cycle, iteration, loop, cyclic
path
багаторазово вкладені цикли –
multiply nested loops
вкладені цикли ~ nested
(nesting) loops
внутрішній ~ inner loop

змінний ~ variable cycle
зовнішній ~ outer loop
машинний ~ computer cycle
повторний ~ repetitive cycle
постійний ~ fixed cycle
~ виконання команди – execute
(execution) cycle, instruction
cycle
~ ітерації (ітераційний ~) –
iterative loop

циклічна робота – cycling

циклічний – cyclic

цифра – character, digit, figure,
numeral

арабська ~ Arabic numeral

вісімкова ~ octal digit

двійкова ~ binary digit
(numeral)

двійково-кодована ~
binary-coded digit

десяткова ~ decimal digit

значуща ~ significant figure

кодована ~ coded digit

контрольна ~ check digit

молодша ~ least significant
(rightmost) digit, low-order digit

римська ~ Roman numeral

старша ~ most significant
(leftmost) digit, high-order digit

цифровий – digital, numeric(al)

Ч

час – time

~ роботи схеми – circuit time

~ перетворення коду – code
conversion time

~ обчислення – computation
time

машинний ~ computer time

неперервний ~ continuous time

дискретний ~ discrete time

~ виконання команди –
instruction time

~ поширення – propagation time
 істинний (реальний) ~ real time
 ~ очікування – latency
 ~ наростання – rise time
 ~ перемикання – switching time
 сумарний ~ total time
 ~ передачі – transfer time
 ~ затримки – delay time
 інтервал часу – time lag
часова діаграма – timing
 waveform, waveform
частота – frequency, rate
 тактова ~ clock frequency (rate)
 робоча ~ operating frequency
 (rate)
 ~ імпульсів – pulse (recurrence)
 frequency (rate), repetition
 rate
 ~ сигналу – signal frequency
 ~ перемикання – switching
 frequency
 ~ синхронізації – synchronizing
 frequency
 ~ перемикання тригера – toggle
 frequency
частка – quotient
черга – queue
четвірковий – quaternary
чотирнадцятковий – quaterdenary
чіп – chip, dice
 ВІС-чіп – LSI chip (large scale
 integration chip)
 кремнієвий ~ silicon chip
 безкорпусний ~ unpackaged
 chip
 базовий ~ (кристал) – master
 dice
чисельник – numerator, term of
 fraction
число – number, figure
 двійкове ~ binary number
 двійкове *n*-розрядне ~ *n*-bit
 binary number

двійково-кодоване десяткове ~
 binary coded decimal
 number
 двійково-десяткове ~
 binary-decimal number
 парне ~ even number
 непарне ~ odd number
 шістнадцяткове ~ hexadecimal
 (sexadecimal) number
 ціле ~ integer
 багаторозрядне ~
 multi-digit number
 від'ємне ~ negative number
 додатне ~ positive number
 невід'ємне ~ nonnegative
 number
 вісімкове ~ octal number
 просте ~ prime number
 сімкове ~ septenary number
 трійкове ~ ternary number
 десяткове ~ decimal
 кратне ~ multiple
 найменше спільне кратне ~
 least common multiple
 ~ за основою 2 (10) – number in
 radix two (ten)
числовий – numeral, numeric(al)
член – member, term
 ~ рівняння – member
 почленно – term by term
 ~ ряду – term of series
 вільний ~ absolute term
 елементарний ~ elemental term
 подібні члени – similar terms
 залишковий ~ (ряду) –
 remainder term
числення – calculation, calculus
 булеве – Boolean calculations
 (calculus)
 система ~ number(ing) system,
 numerical system
 основа системи ~ radix, base

позиційна система ~ positional
number system

Ш

шар – layer

захисний ~ protective layer

ізолюваний ~ insulating layer

провідний ~ conducting layer

~ діелектрика ~ dielectric layer

швидкість – rate, speed

~ введення (даних) – input
speed

~ виведення (даних) – output
speed

~ обміну – rate of exchange

~ обчислень – computation
speed

~ передачі – bit rate, bit transfer
rate

~ передачі даних –
data (-transfer) rate,
transmission speed

~ передачі інформації –
information rate

~ перемикання – switching
rate (speed)

~ перетворення – conversion
rate

швидкодіючий – fast (-acting),
high speed, quick-operating, rapid

швидкодія – speed, speed of
operation, operating speed,
speed of response

середня ~ average speed

~ арифметичного пристрою –
arithmetic speed

~ обчислювальної машини –
computer speed

~ пам'яті – memory (storage)
speed

шина – bus, line, wire

інформаційна ~ data bus

розрядна ~ bit line

спільна ~ common bus

~ заземлення – ground wire

~ керування – control line

~ переносу – carry line

шифратор – coder, encoder
encoding device

пріоритетний ~ priority encoder

цифровий ~ digital encoder

~ 10 x 4 – BCD-To-Binary
encoder

~ 8 x 3 – Octal-To-Binary
encoder

шістнадцятковий ~ hexadecimal

штрих Шеффера – Sheffer stroke,
Sheffer (stroke) function, stroke
function

Щ

щит (панель) – board

розподілений ~ distributing
board, feeder switchboard

~ живлення – power distribution
unit

~ керування – control
board (desk, panel)

щільність – density

~ (концентрація) дірок – hole
density

~ (концентрація) електронів –
electron density

~ (розміщення) компонентів –
component density

~ зарядів – charge density

~ інформації – information
density

~ монтажу схеми – circuit
density

~ упаковки (розміщення)
компонентів – packaging
(packing) density