НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
Навчально-науковий фізико-технічний інститут
Кафедра математичних методів захисту інформації

«На правах рукопису»
УДК 519.6

«До захисту допущено»
Завідувач кафедри
_____ Сергій ЯКОВЛЄВ
«__» _____ 2025 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Математичні методи криптографічного захисту інформації»

зі спеціальності: 113 Прикладна математика
на тему: **«Оцінки ефективності реалізації S-блоків на малопотужних пристроях»**

Виконав:
студент IV курсу, групи ФІ-14
Геращенко Володимир Сергійович _____

Керівник:
зав. каф. ММЗІ, к.т.н
Яковлєв Сергій Володимирович _____

Рецензент:
ст.викл. каф. ІБ, к.ф.-м.н.
Рибак Олександр Владиславович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ — 2025

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
Навчально-науковий фізико-технічний інститут
Кафедра математичних методів захисту інформації

Рівень вищої освіти — перший (бакалаврський)
Спеціальність — 113 Прикладна математика,
ОПП «Математичні методи криптографічного захисту інформації»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій ЯКОВЛЄВ
«__» _____ 2025 р.

ЗАВДАННЯ
на дипломну роботу

Студент: Геращенко Володимир Сергійович

1. Тема роботи: «*Оцінки ефективності реалізації S-блоків на малопотужних пристроях*», науковий керівник дипломної роботи: зав. каф. ММЗІ, к.т.н Яковлєв Сергій Володимирович,

затверджені наказом по університету №__ від «__» _____ 2025 р.

2. Термін подання студентом роботи: «__» _____ 2025 р.

3. Об'єкт дослідження: *алгоритми генерації ефективних реалізацій S-блоків для малопотужних пристроїв*

4. Предмет дослідження: *S-блоки та їхні властивості для малопотужних пристроїв*

5. Перелік завдань:

1) Ознайомитися з основними метриками та алгоритмами оптимізації S-блоків для малопотужних пристроїв

2) Проаналізувати складність алгоритмів оптимізації S-блоків

3) Отримати результати згенерованих реалізацій для 4-бітових S-блоків, які є кандидатами для використання у шифрі ГОСТ та 8-бітових S-блоків шифру «Калина» за допомогою знайдених алгоритмів та порівняти їх реалізації з S-блоками для малопотужних шифрів

4) Проаналізувати вплив лінійного перетворення входів 4-бітових S-блоків, які є кадидатами для використання у шифрі ГОСТ

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: *Презентація доповіді*

7. Орієнтовний перелік публікацій: *публікація на XXIII Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики» у секції «Актуальні проблеми криптографічного захисту інформації»*

8. Дата видачі завдання: 10 вересня 2024 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Узгодження теми роботи із науковим керівником	01-15 вересня 2024 р.	Виконано
2	Огляд опублікованих джерел за тематикою дослідження	Вересень-жовтень 2024 р.	Виконано
3	Аналіз складності алгоритмів побудови ефективної реалізації S-блоків	Жовтень-грудень 2024р.	Виконано
4	Застосування алгоритмів побудови ефективної реалізації 4-бітових та 8-бітових S-блоків	Січень-березень 2025р.	Виконано
5	Проведення експериметну з метою знаходження більш ефективних реалізацій за допомогою лінійного перетворення входу S-блоку	Березень-квітень 2025р.	Виконано
6	Підготовка та оформлення публікації по темі роботи на конференцію	Квітень-травень 2025р.	Виконано
7	Оформлення дипломної роботи	Травень-червень 2025р.	Виконано

Студент _____ Володимир ГЕРАЩЕНКО

Керівник _____ Сергій ЯКОВЛЄВ

РЕФЕРАТ

Кваліфікаційна робота містить: 55 стор., 13 таблиць, 20 джерел.

Метою дослідження є дослідження основних метрик порівняння реалізацій S-блоків для малопотужних пристроїв та основні алгоритми оптимізації під основні критерії.

Об'єктом дослідження є алгоритми генерації ефективних реалізацій S-блоків для малопотужних пристроїв.

Предметом дослідження є S-блоки та їхні властивості для малопотужних пристроїв.

У роботі було досліджено та проаналізовано основні представлення S-блоків, основні метрики, за якими порівнюють реалізації для малопотужних пристроїв. Було розглянуто та проаналізовано складність алгоритмів, які генерують ефективні реалізації S-блоків за заданими критеріями. Також було застовано розглянуті алгоритми для 4-бітових блоків, які є кандидатами у шифр ГОСТ, а також 8-бітові S-блоки блокового шифру «Калина». Було знайдено більш ефективні реалізації S-блоків, які будуються шляхом лінійного перетворення входів 4-бітових S-блоків шифру ГОСТ.

S-БЛОК, АЛГОРИТМИ ОПТИМІЗАЦІЇ, МАЛОПОТУЖНА КРИПТОГРАФІЯ, ГЕЙТОВИЙ РОЗМІР, ГЛИБИНА

ABSTRACT

The qualification work contains: 55 p., 13 tables, 20 sources.

The aim of the study is to investigate the main metrics for comparing S-box implementations for resource-constraint devices and main optimization algorithms for these criteria.

The object of research is the algorithms for generating efficient S-box implementations on resource-constraint devices.

The object of the study is S-boxes and their metrics for implementation on resource-constraint devices.

In this work studied and analyzed the main representations of S-boxes, the main metrics by which implementations are compared for low-power devices. The complexity of algorithms that generate efficient implementations of S-boxes for given criteria was considered and analyzed. Also, the algorithms was applied for 4-bit S-boxes, which are candidates for the GOST cipher, as well as 8-bit S-boxes were presented of the «Kalyna» block cipher. More efficient implementations of S-boxes were found, which are constructed by linearly transforming the inputs of 4-bit S-boxes of the GOST cipher.

S-BOX, OPTIMIZATION ALGORITHMS, RESOURCE-CONSTRAINT CRYPTOGRAPHY, GATE SIZE, DEPTH

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

V_n — множина усіх n-бітових векторів

$GF(2)$ — поле Галуа характеристики 2

АНФ — алгебраїчна нормальна форма

КНФ — кон'юктивна нормальна форма

LUT — таблиця підстановки (англ. Look-up Table)

AES — Advanced Encryption Standart

ВСТУП

Актуальність дослідження. З поширенням інтернету речей (IoT) і RFID-чипів у побуті та промисловості, все більш важливими стають дослідження ефективних реалізацій для малопотужних пристроїв. Оскільки захист інформації на таких пристроях є також важливим, то в останні роки малопотужна криптографія стає все більш популярною темою для досліджень. Одним з перших алгоритмів оптимізації загальних логічних схем є алгоритм ESSPRESSO [6], який було запропоновано у 1982 році. Після того як симетричний шифр AES став стандартом у 2002 році, з'явилось багато досліджень про оптимізацію логічної схеми шифру та його лінійних та нелінійних компонент для використання у малопотужних пристроях [7, 14, 4], та деякі сучасні роботи досі досліджують його оптимізацію [12]. У 2016 році у дослідженні можливих атак на малопотужний шифр Midori64 [10] було запропоновано алгоритм, який генерує 4-бітові булеві функції з метою мінімізації глибини та використання їх у S-блоку шифру Midori64. У 2016 році було запропоновано перший працюючий метод, який використовує SAT-вирішувач [17]. У 2017 році було реалізовано алгоритм LIGHTER [11], який генерує оптимізовані логічні схеми для лінійних та нелінійних складових симетричних шифрів, який згодом став частиною набору інструментів PEIGEN [1].

Метою дослідження є розробка ефективних реалізацій S-блоків за допомогою автоматизованих методів для малопотужних пристроїв. Для досягнення мети необхідно вирішити такі завдання:

1) ознайомитися з основними метриками та алгоритмами оптимізації S-блоків для малопотужних пристроїв;

2) провести порівняльний аналіз існуючих алгоритмів оптимізації S-блоків;

3) побудувати ефективні реалізації для 4-бітових S-блоків, які є

кандидатами для використання у шифрі ГОСТ; та 8-бітових S-блоків шифру «Калина», порівняти їх складність з S-блоками для малопотужних шифрів;

4) проаналізувати вплив лінійного перетворення входів S-блоку на реалізації 4-бітових S-блоків, які є кадидатами у для викорастання у шифрі ГОСТ.

Об'єктом дослідження є алгоритми генерації ефективних реалізацій S-блоків для малопотужних пристроїв.

Предметом дослідження є S-блоки та їхні властивості для малопотужних пристроїв.

При розв'язанні поставлених завдань використовувались такі *методи дослідження*: теорія складності алгоритмів, теорія булевих функцій, теорія графів.

Наукова новизна отриманих результатів полягає в отримані ефективних реалізацій для практичного використання за допомогою лінійного перетворення входу 4-бітових S-блоків.

Практичне значення результатів полягає у використанні знайдених реалізацій на практиці, а також подальшого аналізу лінійних еквівалентів S-блоку для знаходження ефективних реалізацій.

Апробація результатів та публікації. Частина результатів даного дослідження були представлені на Всеукраїнській науково–практичній конференції студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики» у секції «Актуальні проблеми криптографічного захисту інформації».

1 ОПТИМІЗАЦІЯ РЕАЛІЗАЦІЇ S-БЛОКІВ

В цьому розділі буде розглянутий нелінійний елемент для криптографічних систем – S-блок та критерії його оптимізації для малопотужних пристроїв, такі як гейтовий розмір та глибина. Також буде розглянуто основні алгоритми автоматичної генерації оптимізованих реалізацій S-блоків, а саме S-box Depth Evaluation Tool, алгоритм Ко Штоффелена, LIGHTER та інструмент для синтезу логічних схем Yosys, який застосовують на практиці.

1.1 S-блок

Однією з важливих властивостей блокових шифрів є нелінійсть. Якраз цю властивість забезпечує, зазвичай єдиний нелінійний елемент який використовується у шифрі, **S-блок**.

Означення 1.1. S-блок (англ. substitution box, блок підстановки) – нелінійне відображення $V_n \to V_m$, причому n і m не обов'язково однакові.

S-блок є багатовимірною булевою функцією, тому можна представити її декількома способами, але для цієї роботи будуть розглядатися наступні:

1) Look-up Table (надалі LUT) – таблиця підстановки розміром 2^n, де для кожного можливого входу вказано результат;

2) Формульне представлення – кортеж координатних функцій S-блоку:

$$S(x_1,x_2,\ldots,x_n) = (f_1(x_1,x_2,\ldots,x_n), f_2(x_1,x_2,\ldots,x_n),\ldots, f_m(x_1,x_2,\ldots,x_n)),$$

де $f_1,\ldots,f_m$ – булеві функції, задані формулами;

3) Логічна схема (англ. logic circuit) — орієнтований ациклічний граф, у якого вершинами є вхідні змінні, логічні операції та вихідні змінні.

Представлення у вигляді LUT використовують при генерації або аналізі S-блоків, а також у програмних реалізаціях для звичайних пристроїв. Для малопотужної криптографії використовують два останніх представлення, оскільки LUT займає багато пам'яті та обчислювальних ресурсів. На практиці надають перевагу реалізації у вигляді логічної схеми, тому що кортеж булевих функцій ізолює кожен біт входу та виходу, що може бути наслідком у збільшені розміру реалізацій.

1.2 Метрики для малопотужної криптографії

Для визначення ефективності реалізації S-блоку для малопотужних пристроїв використовують декілька основних метрик. Для початку буде введено основне означення, яке буде використовуватись протягом усієї роботи.

Означення 1.2. Гейт (англ. gate, logic gate) — базова логічна операція або булева функція, яка повертає 1 або 0. У вітчизняній літературі зазвичай позначається синонімічним терміном «вентиль».

Зазвичай під гейтами маються на увазі фізичні пристрої, які якраз і реалізують певну операцію. Базовими операціями вважаються:

1) NOT – логічне НІ, формульно позначається $\neg x$;

2) AND – логічне ТА, формульно позначається $x \wedge y$;

3) OR – логічне АБО, формульно позначається $x \vee y$;

4) XOR – виключне АБО, формульно позначається $x \oplus y$;

Також визначаються NAND, NOR, та XNOR – заперечення ТА, АБО, та виключного АБО відповідно.

В залежності від технології, яка розглядається для практичної реалізації на мікроконтролерах, можуть бути реалізовані і додаткові гейти. Наприклад, також існують NAND3, NOR3, XOR3, XNOR3 – операції з трьома змінними та операції з чотирма змінними:

– MAOI1(a,b,c,d) = $\neg((a \wedge b) \vee (\neg(c \vee d)))$;

– MOAI1(a,b,c,d) = $\neg((a \vee b) \wedge (\neg(c \wedge d)))$;

Ці операції імплементовані як окремі гейти для технології UMC 180nm [13], яка буде розглядатись як основна технологія для цієї роботи. ів Приведемо основні метрики, за якими можна порівнювати булеві функції для реалізації [17].

Означення 1.3. Мультиплікативна складність (англ. multiplicative complexity) — кількість нелінійних гейтів у реалізації.

При обмежуємо реалізації функції тільки операціями {AND, OR, XOR, NOT} треба буде враховувати лише AND та OR. Оптимізація цієї властивості підвищує стійкість від атак за сторонніми каналами (англ. side-channel attack) з використанням випадкових масок. Так як ця властивість не показує складність реалізації і має суто криптографічну властивість, акценту на цю метрику буде не багато.

Означення 1.4. Бітова складність гейтів (англ. Bit-sliced complexity) – кількість операцій {AND, OR, XOR, NOT} у реалізації.

Зазвичай ця метрика використовується для архітектур, де немає окремих гейтів для NAND, NOR, XNOR або для програмних реалізаціях [17].

Так як залежно від архітектури, імплементації і фізичних відмінностей у різних гейтах, не всі вони мають однаковий розмір. Тому за стандартну одиницю виміру вважається GEs (англ. gate equivalents) – розмір NAND гейт у технології, яка розглядається [13]. Усі інші гейти вимірюються відносно GEs, тому що їх розмір, з точки зору фізичної реалізації, більший. Наприклад, для технології UMC 180nm [13], розмір гейтів приведено у таблиці 1.1:

Означення 1.5. Гейтовий розмір (англ. gate size, інколи gate area, area cost) – кількість GEs у реалізації.

У подальшому ця метрика буде основною властивістю для оптимізації.

Таблиця 1.1 – Розмір гейтів у бібліотеці UMC 180nm, приведені у GEs

Гейти	AND OR	NOT	NAND NOR	XOR XNOR	NAND3 NOR3	XOR3 XNOR3	MAOI1	MOAI1
Розмір гейту (GEs)	1.33	0.67	1.00	3.00	1.33	4.67	2.67	2.00

Деякі роботи розглядають не тільки розміри гейтів, але і їх час виконання відносно одне одного, що також може суттєво впливати на реалізацію.

Означення 1.6. Шлях затримки гейту — час виконання базового гейта, наскільки гейт «затримує» вивід результату.

Ця метрика, так само як і розмір, визначається відносно NAND гейта, тому значення з таблиці 1.1 також показують і шлях затримки гейту.

Означення 1.7. Шлях затримки функції — максимальна сума послідовних шляхів затримки базових операцій у представленні формулою, або максимальна сума шляху у представленні логічної схеми від вхідної вершини до вихідної.

Означення 1.8. Глибина (англ. depth) — максимальна сума послідовних шляхів затримки базових операцій у представленні формулою, або максимальна сума шляху у представленні логічної схеми від вхідної вершини до вихідної [10].

Глибина є також важливим показником при практичній реалізації, так як фактично є швидкістю виконання функції.

Приклад 1.1. Глибина для виразу

$$((c \text{ NAND } d) \text{ NAND } (b \text{ NAND } (\text{ NOT } a))) \text{ NOR } (b \text{ OR } (a \text{ XOR } d))$$

приблизно дорівнює 4.5. Дійсно, якщо взяти лівий доданок останньої операції NOR, глибина вийде 2.5, а правої частини 3.5. Так як операції NOR треба чекати на змінні з двох сторін, загальна затримка буде

дорівнювати 3.5 – додаючи до затримки самого NOR, отримуємо глибину 4.5. Розмір гейтів – 8 GEs;

Зауваження. Менша глибина не завжди значить менший розмір гейтів. Наприклад, якщо взяти майже будь-яку булеву функцію, яка представлена у КНФ, розмір збільшиться в рази, але глибина буде всього дорівнювати 2.

Означення 1.9. Ефективна реалізація S-блоку — представлення S-блоку у вигляді логічної схеми з мінімально можливими значеннями гейтового розміру та глибини.

Також є і інші метрики, які залежать від технології, такі як споживання енергії, але вони не будуть розглядатись, оскільки напряму залежать від гейтового розміру. Подальшою задачею буде знаходження ефективних реалізацій S-блоків.

1.3 Методи пошуку ефективних реалізацій S-блоків

Для криптографії на малопотужних пристроях зазвичай створюються окремі шифри, в яких елементи спеціально побудовані так, щоб вони витрачали якомога менше ресурсів.

Але для шифрів, які не були створені для такої задачі є методи як знайти та створювати ефективні реалізації для малопотужних пристроїв. Оскільки зазвичай S-блок є єдиною нелінійною частиною в шифрі та він займає найбільшу частину, тому його оптимізація і буде розглядатись.

1.3.1 S-box Depth Evaluation Tool

У роботі [10], яка описує «Invariant subspace attack» на шифр для малопотужних пристроїв Midori64, дослідники розробили алгоритм для знаходження глибини, розміру гейтів та формульного представлення булевої функції за допомогою вектора значень цієї булевої функції.

Описаний алгоритм працює при $n = 4$, але в теорії може бути розширений для будь-якого n.

Алгоритм ґрунтується на таблицях передобчислень, які містять в собі елементи, задані наступним чином:

– вектор значень булевої функції;

– формульний вираз, який описує задану функцію;

– глибина;

– розмір гейтів.

Рекурсивно запускається алгоритм для кожної глибини d:

– $d = 0$: самі вхідні біти (змінні) є функціями з такою глибиною;

– $d = 0.5$: застосувати операцію NOT до функцій з глибиною 0;

– $d = 1$: застосувати операцію NAND/NOR до функцій з глибиною 0;

– $d = 1.5$: три різних способи щоб згенерувати таку функцію:

1) використати операцію AND/OR до двох різних функцій з глибиною 0;

2) використати операцію NAND/NOR до двох різних функцій з глибиною 0.5;

3) використати операцію NOT до функцію з глибиною 1;

– $d \geq 2$: чотири різних способи щоб згенерувати таку функцію: використати операції XOR/XNOR, AND/OR, NAND/NOR та NOT до функцій з глибиною $d - 2$, $d - 1.5$, $d - 1$, $d - 0.5$ відповідно.

Якщо функція наприкінці не є збалансованою, то вона не додається в таблицю. Також якщо є декілька формульних виразів для однієї функції, які мають однакову глибину, то обирається та, у якої розмір менший.

Побудувавши таблицю передобчислень, можемо побудувати надійний S-блок наступним чином:

1) Знайти такі пари булевих функцій з бажаною глибиною так, щоб забезпечувати 2 бітовий баланс – щоб кількість 00, 01, 10, 11 була однаковою у векторах значень булевих функцій;

2) Переконатись, щоб комбінація таких пар була перестановкою.

За допомогою цього інструменту дослідники згенерували $\approx 2^{25.2}$ S-блоків з глибиною не більше 3.5, серед яких $\approx 2^{11.17}$ глибини 3. Кількість булевих функцій на кожній глибині представлено у таблиці 1.2.

Таблиця 1.2 – Кількість збалансованих функцій відносно глибини

Глибина	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5
Кількість	2^2	2^2	0	0	2^4	$2^{6.86}$	$2^{8.39}$	$2^{10.72}$	$2^{13.43}$	$2^{13.64}$

За допомогою цього алгоритму дослідники знайшли S-блок мінімальної глибини, який було запропоновано замість оригінального S-блоку Midori64. Цей алгоритм можна застосовувати як для знаходження S-блоків мінімальної глибини, так і для знаходження реалізації для заданого S-блоку. У цьому досліджені також було сказано, що використовувати на практиці побудовані S-блоки не ефективно, так як представлення у вигляді окремих булевих функцій не є оптимальним, хоча цей алгоритм є прийнятним для первинного аналізу реалізацій S-блоків.

1.3.2 Методи, основані на SAT–вирішувачі

Є декілька робіт, які використовують SAT-вирішувачі для знаходження реалізації S-блоків з мінімальним розміром або глибиною. Для початку розглянемо що таке SAT задача:

SAT (від англ. satisfiability) – це фундаментальна задача в математиці, яка полягає у визначенні, чи існує такий набір значень змінних, при якому булева формула стає істинною. Це була перша задача, для якої було доведено, що вона NP-повна.

SAT-вирішувачі працюють наступним чином:

1) Вхід: формульний вигляд булевої функції у кон'юктивній нормальній формі;

2) Вихід: повертає SAT і набір змінних, для яких формула є істинною, або UNSAT, якщо такого набору значень не існує;

Однією з перших робіт по оптимізації реалізації S-блоків була робота Ко Штоффелена [17] 2016 року, хоча були і інші роботи, на які він посилається [8, 15]. На відміну від інших, в цій роботі було приведено практичне застосування ідеї використання SAT-вирішувача для цієї задачі.

В його роботі було розглянуто оптимізацію для усіх чотирьох критеріїв, які були приведені вище, але тут буде розглядатись тільки два: бітова складність та кількість гейтів.

Для оптимізації S-блоку треба спочатку побудувати модель або задачу, для якої можна буде використати SAT-вирішувач. Ко Штоффелен запропонував бінарну модель, яка вирішує задану проблему: *Чи існує такий S-блок, який використовує щонайбільше k гейтів?* у випадку для кількості гейтів або *Чи існує такий S-блок, який має щонайбільше глибину k та використовуючи щонайбільше w гейтів на кожній глибині?* для знаходження глибини.

Розглянемо алгоритм детальніше. Побудуємо систему з декількома змінними над полем $GF(2)$, де:

– x_i змінні, що представляють входи S-блоку;

– y_i змінні, що представляють виходи S-блоку;

– q_i змінні, що представляють входи гейтів;

– t_i змінні, що представляють виходи гейтів;

– a_i змінні, що представляють з'єднання між гейтами;

– b_i змінні, що представляють який гейт використовується.

Спочатку розглянемо правила побудови для оцінки бітової складності.

Нехай k – бажана бітова складність гейтів, $S : V_n \to V_m$ – S-блок, поданий у вигляді таблиці. Тоді система C будується таким чином:

– $\forall i \in \{0,\ldots,k-1\}$:

$$t_i = b_{3i} \cdot q_{2i} \cdot q_{2i+1} + b_{3i+1} \cdot q_{2i+1} + b_{3i+2} \cdot q_{2i} + b_{3i+3} + b_{3i+1} \cdot b_{3i+2} \cdot q_{2i};$$

для кодування k гейтів AND, OR, XOR або NOT, причому b_i показує, який гейт це буде, як показано в Таблиці 1.3.

– $\forall i \in \{0,\ldots,2k-1\}$:

$$q_i = \left(\sum_{j=0}^{n-1} a_{l+j} \cdot x_j\right) + \left(\sum_{j=0}^{\lfloor \frac{i}{2} \rfloor - 1} a_{l+n+j} \cdot t_j\right)$$

де $l = in + \left\lceil \frac{i^2-2i+1}{2} \right\rceil$, для кодування того, що входи гейтів можуть бути будь-яким вхідним бітом S-блоку або будь-яким раніше обчисленим бітом.

– $\forall i \in \{0,\ldots,2k-1\}$,
$\forall j \in \{l,\ldots,l+n+\lfloor \frac{i}{2} \rfloor - 2\}$,
$\forall u \in \{j+1,\ldots,l+n+\lfloor \frac{i}{2} \rfloor - 1\}$:

$$a_j \cdot a_u = 0;$$

для кодування обмеження «не більше одного» на входах гейтів.

– $\forall i \in \{0,\ldots,m-1\}$:

$$y_i = \left(\sum_{j=0}^{n-1} a_{s+j} \cdot x_j\right) + \left(\sum_{j=0}^{k-1} a_{s+n+j} \cdot t_j\right);$$

де $s = 2kn + k(k-1) + i(n+k)$, для кодування того, що вихідний біт S-блоку може бути будь-яким вхідним бітом S-блоку або будь-яким виходом гейту.

– $\forall i \in \{0,\ldots,m-1\}$,
$\forall j \in \{s,\ldots,s+n+k-2\}$,
$\forall u \in \{j+1,\ldots,s+n+k-1\}$:

$$0 = a_j \cdot a_u;$$

для кодування обмеження «не більше одного» на виходах S-блоку.

Таблиця 1.3 – Кодування операцій через b_i для бітової складності

$b_{3i}\|\|b_{3i+1}\|\|b_{3i+2}$	Функція для гейта t_i
000	0
001	NOT q_{2i}
010	q_{2i} XOR q_{2i+1}
011	обмеження b_{3i+2} запобігає появі
100	q_{2i} AND q_{2i+1}
101	обмеження b_{3i+2} запобігає появі
110	q_{2i} OR q_{2i+1}
111	обмеження b_{3i+2} запобігає появі

Оскільки у цей набір рівнянь ще не закодовано S-блок, то треба створити 2^n копій рівнянь, у яких усі x_i, y_i, q_i, t_i пронумеровані по іншому, але всі a_i, b_i залишаються тими самими. Сам S-блок включається у систему $2^n \cdot (n+m)$ константними рівняннями для кожного біту входу та виходу. Фактично, C' – система рівнянь з багатьма змінними над полем $GF(2)$ для якої треба знайти розв'язок. Метод який дозволяє перевести таку систему рівнянь у КНФ для можливості застосувати SAT-вирішувач наведений у роботі [2], на який Ко Штоффелен і посилається.

Побудова рівнянь для гейтового розміру будується аналогічним чином, але t_i кодуються по іншому. Нехай k – бажана кількість гейтів, тоді:

$\forall i \in \{0, \ldots, k-1\}$:

$$t_i = b_{3i} \cdot q_{2i} \cdot q_{2i+1} + b_{3i+1} \cdot q_{2i} + b_{3i+1} \cdot q_{2i+1} + b_{3i+2};$$

для кодування k гейтів, причому b_i визначають, який тип гейта це буде, як показано в Таблиці 1.4.

Все інше таке саме, як і в системі рівнянь для бітової складності, включаючи останній крок з переведенням системи до КНФ.

Як було згадано раніше, цей алгоритм не розглядає розмір різних гейтів, тому використовувати знайдені реалізації на практиці не рекомендується. Але використання SAT-вирішувача дозволяє аналізувати

Таблиця 1.4 – Кодування операцій через b_i

$b_{3i}\|\|b_{3i+1}\|\|b_{3i+2}$	Функція для гейта t_i
000	0
001	1
010	q_{2i} XOR q_{2i+1}
011	q_{2i} XNOR q_{2i+1}
100	q_{2i} AND q_{2i+1}
101	q_{2i} NAND q_{2i+1}
110	q_{2i} OR q_{2i+1}
111	q_{2i} NOR q_{2i+1}

S-блоки з точки зору того, яка мінімальна кількість гейтів потрібна для реалізації.

1.3.3 LIGHTER

У 2017 році у роботі [11] було представлено евристичний алгоритм LIGHTER, який шукає реалізації лінійних та нелінійних компонент блокових шифрів. Згодом покращення цього алгоритму було представлено у роботі [1] про набір інструментів PEIGEN для S-блоків.

Для алгоритму LIGHTER на вхід подається перестановка $S : V^n \to V^n$, множина B – гейти, які будуть використовуватись та функція розміру гейтів $||o||$, де $o \in B$. Алгоритм базується на підході «зустрічі посередині» (Meet-in-the-Middle), будуючи орієнтований граф з двох сторін: від тотожньої функції Id і до перестановки S, шляхом ітеративного застосування гейтів з B. Роботу цього алгоритму можна побачити у псевдокоді 1.1.

Фактично алгоритм інтерпретує задачу як пошук найкоротшого шляху в орієнтованому зваженому графі $G = (V, E)$, де вершини V — це певні перестановки, а ребро $(v_1 \to v_2) \in E$ існує тоді й лише тоді, коли v_2 отримується з v_1 застосуванням гейту $o \in B$. Вартість ребра відповідає вартості інструкції.

Algorithm 1.1 Алгоритм пошуку реалізації для S-блоку

```
1: function MitM(S_0, S_1, Λ)
2:     v_0 ← Conv(S_0),   λ_0 ← 0,   V_0^0 ← {v_0},   G_0 ≝ (V_0, E_0) ← ({V_0^0}, ∅)
3:     v_1 ← Conv(S_1),   λ_1 ← 0,   V_1^0 ← {v_1},   G_1 ≝ (V_1, E_1) ← ({V_1^0}, ∅)
4:     σ ← 0
5:     while λ_0 + λ_1 < Λ do
6:         (G_σ, λ_σ) ← Expand(G_σ, λ_σ)
7:         I ← V_0 ∩ V_1
8:         σ ← σ ⊕ 1
9:     end while
10:    if I ≠ ∅ then
11:        return GetImplementation(v_0, I, v_1)
12:    end if
13:    return ∅
14: end function
15: function Expand(G = ({V^0, ..., V^{λ-1}}, E), λ)
16:    V^λ ← ∅,   E^λ ← ∅
17:    for all o ∈ B do
18:        c ← ||o||
19:        for all v ∈ V^{λ-c} do
20:            S ← Succ(v, o)
21:            V^λ ← V^λ ∪ S
22:            E^λ ← E^λ ∪ {v → w, w ∈ S}
23:        end for
24:    end for
25:    G ← ({V^0, ..., V^{λ-1}, V^λ}, E ∪ E^λ)
26:    if V^λ = ∅ then
27:        return Expand(G, λ + 1)
28:    else
29:        return (G, λ)
30:    end if
31: end function
```

На кожному кроці підтримується лічильник відстані λ та множина перестановок, досяжних на відстані не більше λ. Вартість операцій нормується таким чином, щоб 1 було найменше значення гейтового розміру. Розширення, за яке відповідає функція Expand, здійснюється шляхом генерації найближчих невключених перестановок додаванням їх до графа. Всі функції згруповано за відстанями у лексикографічно впорядковані списки.

Функція Conv відповідає за конвертацію перестановки у зручне для алгоритму представлення, функція GetImplementation відповідає за пошук у графі для знаходження імплементації, а Succ повертає функції, які досяжні з вершини v за допомогою операції o. Після перевищення порогу Λ якраз і шукається реалізація цього S-блоку за допомогою двохстороннього алгоритму Дейкстри для пошуку мінімального шляху

або повертається порожня множина, якщо колізії у графі не знайшлося.

Перевагою LIGHTER є те, що алгоритм дозволяє задавати розміри гейтів і використовувати усі операції з таблиці 1.1. PEIGEN є покращенням LIGHTER, який додає можливість генерації S-блоку з оптимізованою глибиною, а також пришвидшує роботу алгоритму використовуючи передобчислення графу.

1.3.4 Yosys: Yosys Open SYnthesis Suite

Усі розглянуті методи, на жаль, не є застосовними для S-блоків розміром $n > 5$. Також вони не знаходять ефективні реалізації для практичного використання, оскільки орієнтовані на теоретичні метрики без врахування особливостей конкретних технологій виготовлення мікросхем.

Yosys (Yosys Open SYnthesis Suite) [18] є інструментом синтезу цифрових схем, який використовується на практиці. На відміну від спеціалізованих алгоритмів для S-блоків, Yosys розроблявся як універсальний інструмент для синтезу довільних цифрових схем. Один з його елементів, інструмент ABC [5] використовувався у досліджені про LIGHTER, де порівнювались реалізації між цими двома методами, а також покращення схеми, згенерованої LIGHTER.

Для оптимізації S-блоків Yosys використовує декілька етапів обробки: застосовуються логічні оптимізації, такі як спрощення булевих функцій, видалення надлишкових операцій та реструктуризація схеми. На останньому етапі відбувається технологічне відображення (technology mapping) за допомогою ABC, де абстрактні логічні операції замінюються конкретними гейтами з обраної технологічної бібліотеки. Стандартним розширенням для роботи з Yosys є Verilog – стандарт для опису та дизайну логічних схем, але також він підтримує формат BLIF(Berkeley Logic Interchange Format), який використовується в більш простих описах логічних схем. Також Yosys може генерувати оптимізовані реалізації

навіть для більших S-блоків, наприклад 8-бітових S-блоків «Калини».

Висновки до розділу 1

В цьому розділі було розглянуто нелінійний елемент блокового шифру S-блок, його основні представлення та метрики, за якими можна порівнювати реалізації S-блоків для малопотужних пристроїв. Було розглянуто алгоритми оптимізації S-блоків за різними критеріями, а саме: S-box Depth Evaluation Tool – ітеративний алгоритм пошуку реалізацій мінімальної глибини; алгоритм Ко Штоффелена, який використовує SAT-вирішувач для пошуку оптимальних реалізацій; LIGHTER – алгоритм, який проектує проблему на граф та використовує модифікований алгоритм Дейкстри для пошуку; та Yosys – алгоритм синтезу будь яких логічних схем, який використовується на практиці.

2 АНАЛІЗ ТА ЗАСТОСУВАННЯ АЛГОРИТМІВ ГЕНЕРАЦІЇ ЕФЕКТИВНИХ S-БЛОКІВ

В цьому розділі буде приведено аналіз складності алгоритмів, наведених в першому розділі. Також буде наведено результати застосування цих алгоритмів для генерації реалізацій S-блоків, які є кандидатами у застосуванні для блокового шифру ГОСТ, S-блоків шифру «Калина» та експеремент по знаходженню ефективних S-блоків за допомогою лінійних перетворень.

2.1 Аналіз складності алгоритмів

Задача знаходження мінімальної можливої реалізації булевої функції є Σ_p^2-складною задачею, тому це дає загальне розуміння про складність знаходження такої оптимізації. Далі наведено аналіз приблизної складності алгоритмів для оптимізації S-блоку.

2.1.1 S-box Depth Evaluation Tool

Для того, щоб оцінити складність цього рекурсивного алгоритму, розглянемо його рекурентне співвідношення. Базова операція буде розглядатись як функція знаходження та перевірки булевої функції з формульної реалізації. Тоді, кількість викликів таких функцій відносно глибини буде рахуватись наступним чином:

$$\mathrm{T(d) = T(d - 0.5) + T(d - 1)(T(d - 1) - 1) + T(d - 1.5)(T(d - 1.5) - 1) + T(d - 2)(T(d - 2) - 1);} \quad (2.1)$$

де $T(d)$ – кількість викликів для глибини d.

Базові випадки для такої функції будуть наступними:

$$\begin{aligned} T(0) &= n; \\ T(0.5) &= n; \\ T(1) &= n(n-1); \\ T(1.5) &= 3n(n-1). \end{aligned}$$

У дослідженні [10] розглядались булеві функції розміру $n = 4$. Було розглянуто функції до глибини $d = 5$ включно, причому майже усі реалізації збалансованих булевих функцій було знайдено при $d = 4.5$. Цей алгоритм генерує булеві функції по декілька разів, фактично повністю перебираючи множину усіх булевих функцій при $n = 4$.

Таблиця 2.1 – Приблизна кількість перебраних функцій відносно глибини d

	$d = 3$	$d = 3.5$	$d = 4$	$d = 4.5$	$d = 5$	$d = 5.5$	$d = 6$
$n = 4$	2^{15}	2^{21}	2^{30}	2^{42}	2^{61}	2^{85}	2^{122}
$n = 8$	2^{23}	2^{30}	2^{46}	2^{60}	2^{93}	2^{120}	2^{187}

З'ясувати, скільки треба згенерувати булевих функцій є важкою задачею, оскільки невідомо, при якій глибині усі булеві функції буде розглянуто. Як можна побачити з таблиці 2.1, було згенеровано $\approx 2^{42}$ реалізацій булевих функцій, причому усього булевих функцій розміру 4 дорівнює 2^{16}. Для $n = 5$ цей алгоритм вже стає занадто складним на практиці.

Пошук у таблиці передобчислень не залежить від n, і на практиці зручно використовувати цей алгоритм для перебору багатьох S-блоків для знаходження оптимальних реалізацій після етапу генерування таблиці.

2.1.2 Методи, які використовують SAT-вирішувач

Алгоритми Ко Штоффелена складаються з декількох етапів, але самою складною частиною є саме знаходження результату згенерованої SAT-задачі. Через те, що немає точної оцінки для розв'язку SAT-задачі і вона залежить від складності КНФ, яка подається на вхід, то оцінити роботу алгоритму важко.

При оцінці складності використання SAT-вирішувача можуть використовувати кількість доданків у КНФ, як це приведено у роботі [2]. Хоча кількість доданків зростає з кількістю гейтів k, але Ко Штоффелен [17] зауважив, що робота SAT-вирішувача найдовша при меншому k. Також при практичному використанні цього алгоритму залежність між кількістю гейтів k та фактичним часом роботи SAT-вирішувача не є очевидною.

Відомо, що результати у цих роботах були приведені тільки для $n = 4$ для гейтової складності та для $n = 5$ для мультиплікативної складності, навіть з використанням потужних обчислювальних ресурсів.

2.1.3 LIGHTER

Нехай n – розмір S-блоку, $|B|$ – кількість доступних типів гейтів Λ – обмеження вартості реалізації.

Розглянемо випадок, коли усі гейти мають однаковий розмір, оскільки це не сильно впливає на складність алгоритму.

Тоді, часова складність функції Expand становить $O(|B| * |V^{\lambda-1}|)$ операцій Succ для λ, оскільки застосовується операція для кожної вершини. Оскільки ми маємо однакові розміри гейтів, то можна розширити цю асимптотичну оцінку до $O(|B|^{\lambda})$. Дійсно, $|V^0| = 1$, $|V^1| = |B|$, $|V^2| = |B|^2$, ..., тому складність в найгіршому складає $O(|B|^{\frac{\Lambda}{2}})$.

Просторова складність становить $O(\sum_{\lambda=0}^{\frac{\Lambda}{2}}(|B|^{\lambda}))$.

Знаходження оцінки складності алгоритму з різними розмірами гейту є вже більш складною задачею, але загальна асимптотична оцінка буде такою ж самою.

Це є справедливим і для PEIGEN, який покращує роботу LIGHTER, але не змінює основну структуру алгоритму. Як можна побачити, цей алгоритм також є експоненційним, хоча його застосовували навіть для 5-бітових S-блоків.

2.2 Результати та особливості застосування алгоритмів оптимізації S-блоків

У цій роботі було згенеровано реалізації восьми S-блоків, наведених у роботі [20], а саме:

$$\begin{aligned}
S_1 &= [7, 9, 4, D, 0, 2, C, B, A, 8, 1, 6, E, 5, F, 3],\\
S_2 &= [1, 9, 6, 5, B, E, 2, 8, 4, A, F, 3, 0, 7, C, D],\\
S_3 &= [A, C, 3, 8, B, 7, D, 0, 4, 5, 1, F, E, 9, 6, 2],\\
S_4 &= [E, A, F, 1, 0, D, 7, 4, 5, 2, 8, 6, 3, B, 9, C],\\
S_5 &= [B, 0, D, E, 6, 4, 7, 9, 5, 1, C, 2, 8, F, A, 3],\\
S_6 &= [1, C, 3, 8, 0, 6, E, D, F, B, 4, 5, 9, 2, A, 7],\\
S_7 &= [E, 7, B, D, 8, 2, 5, 6, 3, 0, 1, C, F, 9, 4, A],\\
S_8 &= [4, 3, A, B, D, E, 8, 5, 9, 1, 7, C, F, 2, 6, 0].
\end{aligned} \tag{2.2}$$

Ці S-блоки мають усі потрібні криптографічні властивості для використанні у шифрі ГОСТ. Буде розглянуто результати запуску алгоритмів, порівнюючи їх з реалізаціями S-блоків вже існуючих малопотужних блокових шифрів, таких як S-блоки PICCOLO [16] та SKINNY [3], а також для шифру Rectangle [19],які були згенеровані у роботах самих дослідників, якщо не зазначено інакше. Також буде

приведено опис певних властивостей запуску та час роботи цих алгоритмів.

2.2.1 S-box Depth Evaluation Tool

Дослідники представили цей алгоритм для генерації та знаходження S-блоків з потрібними властивостями, щоб замінити S-блок шифру Midori64 [10]. Також у дослідженні приведені тільки результати знайденого S-блока, який є мінімальним серед усіх знайдених S-блоків відносно глибини та гейтової складності, і має необхідні криптографічні властивості. У дослідженні не було вказано результати окрім як для шифру Midori64, тому було застосовано цей алгоритм вручну для усіх інших S-блоків. Результати можна побачити у таблиці 2.2.

Реалізація цього алгоритму є у відкритому доступі [10], разом із згенерованою таблицею передобчислень. Таблиця передобчислень була згенерована за 10 годин, але пошук у таблиці миттєвий, так як сама таблиця є невеликою для сучасних пристроїв.

Як можна побачити, S-блоки шифрів для малопотужних пристроїв показують кращі результати, ніж S-блоки ГОСТу, так як вони розроблялись спеціально для малопотужних пристроїв. Також тут використовувались свої значення гейтових розмірів, які не є точними на практиці, але показують приблизне відношення розмірів гейтів між собою, що достатньо для теориточного дослідження:

– NOT= 0.5 GEs;
– NAND/NOR= 1 GEs;
– AND/OR= 1.5 GEs;
– XOR/XNOR= 2 GEs.

Таблиця 2.2 – Результати застосування S-box Depth Evaluation Tool на S-блоках

S-блок	Гейтовий розмір	Глибина
S_1	39 GEs	4.5 GEs
S_2	39 GEs	4.5 GEs
S_3	38.5 GEs	4.5 GEs
S_4	38.5 GEs	4.5 GEs
S_5	38.5 GEs	4.5 GEs
S_6	39.5 GEs	4.5 GEs
S_7	38 GEs	4.5 GEs
S_8	38 GEs	4.5 GEs
PICCOLO	20 GEs	4 GEs
SKINNY	20 GEs	4 GEs
Rectangle	30.5 GEs	4.5 GEs
Midori64	30 GEs	4 GEs

2.2.2 Метод з використанням SAT-вирішувача

Алгоритм Ко Штоффелена також є у відкритому доступі [17]. Цей алгоритм тільки генерує систему рівнянь над полем, та переводить результат застосування SAT-вирішувача у логічну схему. Сам Ко Штоффелен розробляв цей алгоритм під застарілий SAT-вирішувач MiniSat [9], що додало труднощів у практичному використанні.

Цей алгоритм для деяких розмірів гейту є доволі затратним з точки зору часу, тому результати для шифру ГОСТ були отримані тільки для розмірів гейту $k \geq 18$. Для всіх розглянутих S-блоків значення k яке приводило до UNSAT дорівнює 7, що може свідчити про мінімально необхідну кількість гейтів для реалізації. Робота MiniSat для

згенерованих КНФ становила від 30 хвилин до 2 днів для знайдених результатів, та до 4 днів для $k \leq 17$ без результату. Результати застосування алгоритму можна побачити у таблиці 2.3.

Таблиця 2.3 – Результати застосування алгоритму Ко Штоффелена на S-блоках

S-блок	Кількість гейтів	Глибина
S_1	19	4
S_2	18	4
S_3	20	5
S_4	20	4
S_5	20	4
S_6	19	4
S_7	20	4
S_8	20	4
PICCOLO	8	3
Rectangle	8	3

Так як результати у роботі [17] були отримані з використанням потужних обчислювальних ресурсів, реалізовані S-блоки ГОСТу є більшими у реалізації. Також цей алгоритм не враховує розмір гейтів, що призводить до великої кількості XOR-гейтів у реалізаціях. Глибина у цій таблиці приведена як кількість вкладених функцій, і є суто показовою метрикою у цьому алгоритмі.

2.2.3 LIGHTER

На вхід LIGHTER подається LUT S-блоку у вигляді рядку, обмеження Λ, бажані гейти та розміри гейтів.

У даній роботі використовувалася покращена версія LIGHTER у

PEIGEN, який також є відкритому доступі [1]. Застосовувався алгоритм для режиму генерації з оптимізованої гейтовою складністю та обмеженням $\Lambda = 12$. Також були задані значення гейтових розмірів з таблиці 1.1.

Таблиця 2.4 – Результати застосування алгоритму LIGHTER на S-блоках

S-блок	Кількість гейтів	Глибина
S_1	22.99 GEs	10
S_2	20.66 GEs	11
S_3	22.67 GEs	10
S_4	25.00 GEs	9
S_5	24.67 GEs	9
S_6	22.00 GEs	10
S_7	21.00 GEs	10
S_8	24.00 GEs	9
PICCOLO	13.00 GEs	3
SKINNY	13.33 GEs	3
Rectangle	18.33 GEs	5

Передобчислення графу займає годину, а сама оптимізація S-блоку займає до хвилини. Результати згенерованих реалізацій можна побачити у таблиці 2.4.

Оскільки алгоритм оптимізує гейтову складність та обирає найменший гейтовий розмір, який знаходиться самим останнім у графі, то згенеровані реалізації для S-блоків ГОСТу мають занадто велику глибину для використання на практиці.

2.2.4 Yosys: Yosys Open SYnthesis Suite

Інструмент Yosys застосовується в багатьох комерційних проектах, та його відкритий код також є у вільному доступі [18]. Для оптимізації S-блоку треба закодувати його представлення у вигляді таблиці підстановки у PLA файл, обробити інструментом ABC, перетворивши його у Verilog формат, та застосувати синтез та відображення на задану технологію. Результати застосування можна побачити у таблиці 2.5.

Таблиця 2.5 – Результати застосування інструменту Yosys на S-блоках

S-блок	Кількість гейтів	Глибина
S_1	27.99 GEs	4.67 GEs
S_2	28.00 GEs	5.00 GEs
S_3	29.34 GEs	4.67 GEs
S_4	26.65 GEs	4.67 GEs
S_5	26.65 GEs	4.33 GEs
S_6	27.99 GEs	4.67 GEs
S_7	27.99 GEs	5.00 GEs
S_8	27.65 GEs	4.33 GEs
PICCOLO	24.99 GEs	5.33 GEs
SKINNY	24.01 GEs	5.00 GEs
Rectangle	30.67 GEs	4.67 GEs

Реалізації для S-блоків малопотужних шифрів були згенеровані самостійно. Отримані результати не є оптимальними, і навіть для S-блоків PICCOLO та SKINNY реалізації є гіршими, оскільки Yosys використовує загальні підходи для синтезу логічних схем. Перевагою цього інструменту є швидкість – на оптимізацію одного S-блоку витрачається менше секунди.

Також Yosys був застосований до реалізацій попередніх алгоритмів, як було зроблено у роботі про LIGHTER [11], але це призводило до таких самих реалізацій як у таблиці 2.5 або навіть більших за глибиною та розміром S-блоків.

Також цей інструмент може реалізувати S-блоки більшого розміру, що дозволяє знайти не оптимальну, але приблизну оцінку того, яка буде гейтова складність такої реалізації. Результати застосування цього алгоритму на S-блоках «Калини» приведені у таблиці 2.6.

Таблиця 2.6 – Результати застосування інструменту Yosys на S-блоках

S-блок	Гейтовий розмір	Глибина
π_0	886.22 GEs	10.99 GEs
π_1	886.0 GEs	13.00 GEs
π_2	857.38 GEs	12.66 GEs
π_3	877.99 GEs	11.99 GEs

Можна побачити, що реалізація 8-бітових S-блоків є набагато більшою за 4-бітові, що є суттєвим недоліком для використання на практиці.

2.2.5 Аналіз результатів застосування алгоритмів

Розглянемо результати та основні властивості застосованих алгоритмів:

1) S-box Depth Evaluation Tool – має найгірші показники у мінімізаії гейтового розміру, але має швидку генерацію реалізації, що можна використовувати у аналізі багатьох S-блоків,

2) Алгоритм Ко Штоффелена – в теорії є потужним алгоритмом, який може генерувати оптимальні S-блоки з мінімальної кількістю гейтів,

але на практиці для генерації таких реалізацій треба потужні обчислювальні ресурси. Також алгоритм не розглядає розміри гейтів, що впливає на розміри реалізації.

3) LIGHTER – має найкращі показники у мінімізації гейтового розміру S-блоків, але з найбільшою глибиною, оскільки найменші реалізації знаходяться пізніше у графі. Але для S-блоків створених для малопотужних пристроїв алгоритм генерує оптимальні реалізації з малою глибиною та гейтовим розміром, які дійсно можна використовувати на практиці.

4) Yosys – алгоритм генерує прийнятні реалізації, зі збалансованим гейтовим розміром та глибиною. Він може використовуватись як і для генерації S-блоку з таблиці підстановки, так і для оптимізації логічної схеми. Також є застосовним до 8-бітових S-блоків, що дозволяє оцінити приблизну складність реалізації таких S-блоків на практиці.

Як можна побачити, усі алгоритми мають своє застосування, як практичне, так і теоретичне, та можуть бути корисними в аналізі та реалізації власних S-блоків.

2.3 Лінійне перетворення входів S-блоку

За результатами застосування алгоритмів оптимізації S-блоків можна побачити, що звичайні S-блоки є більшими за перестановки, створені для малопотужних шифрів. Оскільки подальша оптимізація реалізацій заданих S-блоків вимагає великих обчислювальних ресурсів, то можна розглянути перестановки, які мають схожі властивості, але є меншими у реалізації.

Нехай $S : V_4 \to V_4$ — заданий S-блок, у вигляді таблиці підстановки. Застосуємо лінійне перетворення до входу S-блоку:

$$S_{A,b} = S(Ax \oplus b)$$

де A — невироджена матриця над полем $GF(2)$ та $b \in V_4$ — довільний 4-

бітовий вектор. $S_{A,b}$ є лінійно еквівалентним до оригінального S-блоку, але така модель не описує усі лінійно еквівалентні S-блоки.

Такі S-блоки будуть мати більшість криптографічних властивостей і деякі з них будуть менше за розміром ніж оригінальний S-блок. Усього можливих лінійних перетворень для 4-бітового вектора $(2^4 - 1)(2^4 - 2)(2^4 - 4)(2^4 - 8) \cdot 16 = 322560$, що дає великий простір пошуку оптимізованих S-блоків.

2.3.1 Перевірка лінійно еквівалентних S-блоків

Наведемо кроки для проведення такого експерименту:

1) На вхід подається S-блок S як таблиця перестановки;

2) До входу S застосовуються усі можливі лінійні перетворення, отримуємо множину перестановок L;

3) До кожного елементу множини перестановок L застосовується алгоритм S-box Depth Evaluation Tool, отримані результати сортуються за гейтовим розміром реалізації;

4) До N перших перестановок у відсортованих результатах застосовується інструмент Yosys, помічаючи перестановки з нерухомими точками.

Таким чином отримуємо N найефективніших перестановок за гейтовим розміром та глибиною. Оскільки оригінальні S-блоки були без нерухомих точок, то заради збереження цієї властивості, такі перестановки не будуть враховуватись.

Вибір алгоритмів для цього експерименту було визначено практичним чином: S-box Depth Evaluation Tool дозволяє швидко обробляти тисячі S-блоків, а Yosys генерує ефективні реалізації, враховуючи баланс гейтового розміру та глибини, і є швидшим ніж LIGHTER та алгоритм Ко Штоффелена.

Таблиця 2.7 – Загальні результати експерименту

S-блок	Кількість S-блоків без нерухомих точок	Мінімальне GEs	Мінімальна глибина
S_1	1853	16.00 GEs	3.33 GEs
S_2	1882	15.00 GEs	3.33 GEs
S_3	1761	15.99 GEs	3.33 GEs
S_4	1914	16.00 GEs	3.33 GEs
S_5	1936	16.00 GEs	3.33 GEs
S_6	1871	15.00 GEs	3.33 GEs
S_7	1809	15.33 GEs	3.33 GEs
S_8	1932	16.66 GEs	3.33 GEs

2.3.2 Результати застосування

В цьому експерименті було розглянуто усі можливі перестановки $S_{A,b}$ для кожного заданого S-блоку, які наведені раніше у виразі 2.2. Проведено запуск для $N = 5000$, так як мінімальні реалізації згенеровані за допомогою Yosys не відповідає згенерованим реалізаціям за допомогою S-box Depth Evaluation Tool.

Результати отриманих результатів приведено у таблиці 2.7

S-блоки мінімальної глибини та мінімального гейтового розміру – різні S-блоки, тому вони будуть розглядатись окремо. S-блоки мінімального гейтового розміру знаходяться у таблиці 2.8, а мінімальної глибини у таблиці 2.9.

Самою довгою частиною є запуск Yosys, що займає приблизно 0.5 секунд для одного S-блоку, що для $N = 5000$ перетворюється на 40 хвилин. Застосування лінійного перетворення та пошук реалізацій у таблиці, згенерованій S-box Depth Evaluation Tool, для кожної можливого лінійного перетворення займає 2 хвилини для одного S-блоку.

Хоча знайдені реалізації не є настільки меншими, ніж реалізації S-блоків PICCOLO або SKINNY, але вони є набагато ефективнішими за

Таблиця 2.8 – Знайдені S-блоки мінімального гейтового розміру

Оригінальний S-блок	Таблиця підстановки	GEs	Глибина
S_1	[A,2,3,4,B,1,7,5,C,6,9,E,8,0,F,D]	16.00 GEs	4.00 GEs
S_2	[7,6,0,5,1,D,9,C,2,A,8,4,3,B,F,E]	15.00 GEs	5.00 GEs
S_3	[D,9,F,A,C,1,E,0,5,3,B,2,6,7,8,4]	15.99 GEs	3.66 GEs
S_4	[7,2,5,4,B,F,1,3,6,0,D,8,E,C,9,A]	16.00 GEs	4.00 GEs
S_5	[D,F,8,E,3,B,0,A,1,7,9,5,6,2,C,4]	16.00 GEs	4.00 GEs
S_6	[4,C,6,A,2,E,3,B,0,7,5,1,8,F,9,D]	15.00 GEs	4.00 GEs
S_7	[1,6,8,0,2,3,C,5,9,E,D,4,B,A,F,7]	15.33 GEs	4.00 GEs
S_8	[2,D,1,4,A,C,8,0,B,7,5,6,F,E,9,3]	16.66 GEs	4.33 GEs

Таблиця 2.9 – Знайдені S-блоки мінімальної глибини

Оригінальний S-блок	Таблиця підстановки	GEs	Глибина
S_1	[7,3,5,4,F,9,D,E,B,A,1,2,8,C,0,6]	17.33 GEs	3.33 GEs
S_2	[2,A,9,C,7,6,F,E,4,8,D,1,5,0,B,3]	16.99 GEs	3.33 GEs
S_3	[F,D,A,9,E,C,0,1,B,5,2,3,8,6,4,7]	18.33 GEs	3.33 GEs
S_4	[1,3,B,F,9,A,E,C,5,4,7,2,D,8,6,0]	16.32 GEs	3.33 GEs
S_5	[9,5,C,4,8,E,0,A,1,7,6,2,D,F,3,B]	16.66 GEs	3.33 GEs
S_6	[2,4,1,D,E,C,5,9,3,6,7,F,B,A,0,8]	18.33 GEs	3.33 GEs
S_7	[C,F,5,7,8,D,0,4,2,B,3,A,1,9,6,E]	17.32 GEs	3.33 GEs
S_8	[2,8,B,9,A,1,F,5,7,3,D,0,E,6,C,4]	16.66 GEs	3.33 GEs

властивостями ніж оригінальні S-блоки та можуть використовуватись на практиці. Також можна дослідити усі можливі лінійні еквіваленти заданих S-блоків, що може ще більше оптимізувати реалізації S-блоків. Програму для цього дослідження можна знайти за посиланням: https://github.com/gerau/diploma_project.

Висновки до розділу 2

В цьому розділі було проаналізовано складність наведених алгоритмів для оптимізації S-блоків для малопотужних пристроїв та їх

застосування для S-блоків, які є кандидатами для використання у шифрі ГОСТ. Також було реалізовано S-блоки шифру «Калина» за допомогою Yosys, оскільки інші методи не можуть генерувати реалізації великих S-блоків. Було порівняно знайдені реалізації з реалізаціями відомих S-блоків для малопотужних шифрів та запропоновано схему знаходження мінімальної реалізації, побудованих за допомогою лінійного перетворення входу.

ВИСНОВКИ

У цьому розділі було проведено всебічне дослідження нелінійного компонента блокового шифру – S-блоку. Розглянуто основні способи його подання, а також метрики, що дозволяють оцінювати та порівнювати реалізації S-блоків у контексті їх придатності для використання в малопотужних пристроях. Особливу увагу приділено методам оптимізації реалізацій S-блоків за різними критеріями. Проаналізовано чотири підходи: S-box Depth Evaluation Tool — ітеративний алгоритм пошуку реалізацій з мінімальною логічною глибиною; метод Ко Штоффелена, який використовує SAT-вирішувач для знаходження оптимальних реалізацій; алгоритм LIGHTER, що формулює задачу як пошук на графі з використанням модифікованого алгоритму Дейкстри; а також Yosys — інструмент логічного синтезу, що застосовується на практиці для генерації реалізацій загальних схем.

Було здійснено аналіз обчислювальної складності вказаних методів, а також проведено їхнє експериментальне застосування до S-блоків, які є кандидатами для використання у шифрі ГОСТ. У випадку великих S-блоків, таких як у шифрі «Калина», реалізацію було виконано за допомогою Yosys, оскільки інші підходи не є застосовними для великих S-блоків. Отримані реалізації порівняно з реалізаціями відомих малопотужних S-блоків. Також було запропоновано загальну схему знаходження ефективних S-блоків, побудованих шляхом застосування лінійного перетворення до вхідного простору. Було застосовано наведену схему для знаходження ефективних реалізацій з 4-бітових S-блоків, які є кандидатами для використання у шифрі ГОСТ, які є в середньому на 43.2% за гейтовим розміром та на 11.5% за глибиною меншими у випадку оптимізації за розміром та на 37.9% за гейтовим розміром та на 28.5% за глибиною меншими у випадку оптимізації за глибиною.

ПЕРЕЛІК ПОСИЛАНЬ

[1] Zhenzhen Bao та ін. «PEIGEN – a Platform for Evaluation, Implementation, and Generation of S-boxes». В: *IACR Transactions on Symmetric Cryptology* (бер. 2019), 330–394. ISSN: 2519-173X. DOI: 10.46586/tosc.v2019.i1.330-394. URL: https://github.com/peigen-sboxes/PEIGEN.

[2] Gregory V. Bard, Nicolas T. Courtois та Chris Jefferson. «Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomials over GF(2) via SAT-Solvers». В: *IACR Cryptol. ePrint Arch.* 2007 (2007), с. 24. URL: https://api.semanticscholar.org/CorpusID:8263291.

[3] Christof Beierle та ін. «The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS». В: *Advances in Cryptology – CRYPTO 2016*. Springer Berlin Heidelberg, 2016, 123–153. ISBN: 9783662530085. DOI: 10.1007/978-3-662-53008-5_5. URL: http://dx.doi.org/10.1007/978-3-662-53008-5_5.

[4] Joan Boyar та René Peralta. «A new combinational logic minimization technique with applications to cryptology». В: *Experimental Algorithms: 9th International Symposium, SEA 2010, Ischia Island, Naples, Italy, May 20-22, 2010. Proceedings 9*. Springer. 2010, с. 178—189.

[5] Robert Brayton та Alan Mishchenko. «ABC: An Academic Industrial-Strength Verification Tool». В: *Computer Aided Verification*. За ред. Tayssir Touili, Byron Cook та Paul Jackson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, с. 24—40. ISBN: 978-3-642-14295-6.

[6] Robert K Brayton та ін. *Logic minimization algorithms for VLSI synthesis*. Т. 2. Springer Science & Business Media, 1984.

[7] David Canright. «A very compact S-box for AES». В: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2005, с. 441—455.

[8] Nicolas T. Courtois, Theodosis Mourouzis та Daniel Hulme. «Exact Logic Minimization and Multiplicative Complexity of Concrete Algebraic and Cryptographic Circuits». В: *International journal on advances in intelligent systems* 6 (2013), с. 165—176. URL: https://api.semanticscholar.org/CorpusID:58900184.

[9] Niklas Eén та Niklas Sörensson. «An Extensible SAT-solver». В: *Theory and Applications of Satisfiability Testing*. За ред. Enrico Giunchiglia та Armando Tacchella. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, с. 502—518. ISBN: 978-3-540-24605-3.

[10] Jian Guo та ін. «Invariant Subspace Attack Against Midori64 and The Resistance Criteria for S-box Designs». В: *IACR Transactions on Symmetric Cryptology* (груд. 2016), 33–56. ISSN: 2519-173X. DOI: 10 . 46586 / tosc . v2016 . i1 . 33 - 56. URL: https://github.com/qiaokexin/Sbox-depth-evaluation.

[11] Jérémy Jean та ін. «Optimizing Implementations of Lightweight Building Blocks». В: *IACR Transactions on Symmetric Cryptology* (груд. 2017), 130–168. ISSN: 2519-173X. DOI: 10.46586/tosc.v2017.i4.130-168. URL: http://dx.doi.org/10.46586/tosc.v2017.i4.130-168.

[12] Yongjin Jeon та ін. «A Framework for Generating S-Box Circuits with Boyer–Peralta Algorithm-Based Heuristics, and Its Applications to AES, SNOW3G, and Saturnin». В: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2025.1 (груд. 2024), 586–631. ISSN: 2569-2925. DOI: 10 . 46586 / tches . v2025 . i1 . 586 - 631. URL: http://dx.doi.org/10.46586/tches.v2025.i1.586-631.

[13] Zhenyu Lu та ін. «Pushing the Limits: Searching for Implementations with the Smallest Area for Lightweight S-Boxes». В: *IACR Cryptol. ePrint Arch.* 2021 (2021), с. 1644. URL: https://api.semanticscholar.org/CorpusID:245104018.

[14] Sumio Morioka та Akashi Satoh. «An optimized S-Box circuit architecture for low power AES design». В: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2002, с. 172—186.

[15] Theodosis Mourouzis. «Optimizations in algebraic and differential cryptanalysis». В: 2015. URL: https://api.semanticscholar.org/CorpusID:35660382.

[16] Kyoji Shibutani та ін. «Piccolo: An Ultra-Lightweight Blockcipher». В: *Cryptographic Hardware and Embedded Systems – CHES 2011*. Springer Berlin Heidelberg, 2011, 342–357. ISBN: 9783642239519. DOI: 10.1007/978-3-642-23951-9_23. URL: http://dx.doi.org/10.1007/978-3-642-23951-9_23.

[17] Ko Stoffelen. «Optimizing S-Box Implementations for Several Criteria Using SAT Solvers». В: *Fast Software Encryption*. Springer Berlin Heidelberg, 2016, 140–160. ISBN: 9783662529935. DOI: 10.1007/978-3-662-52993-5_8. URL: https://github.com/Ko-/sboxoptimization.

[18] Claire Wolf. *Yosys Open SYnthesis Suite*. URL: https://yosyshq.net/yosys/.

[19] Wentao Zhang та ін. *RECTANGLE: A Bit-slice Lightweight Block Cipher Suitable for Multiple Platforms*. Cryptology ePrint Archive, Paper 2014/084. 2014. DOI: 10.1007/s11432-015-5459-7. URL: https://eprint.iacr.org/2014/084.

[20] Сергій Володимирович Яковлєв. «Збалансовані критерії якості довгострокових ключових елементів алгоритму ГОСТ 28147-89». В: *Інформаційні технології та комп'ютерна інженерія* 1 (2009), с. 48—55.

ДОДАТОК А РЕАЛІЗАЦІЇ ЗНАЙДЕНИХ S-БЛОКІВ

Знайдені реалізації з використанням лінійного перетворення та саме лінійне перетворення у вигляді матриці A та вектору b буде приведено тут. Реалізації використовують BLIF формат, який використовують для теоретичних представлень логічних схем.

А.1 S-блок S_1

Лінійне перетворення для S_1 мінімального гейтового розміру:

$$A = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$$

Реалізація S-блоку у BLIF форматі:

```
.model gost1_ges
.inputs a_i b_i c_i d_i
.outputs a_o b_o c_o d_o
.gate NOT1 A=d_i O=_27_
.gate NOT1 A=c_i O=_19_
.gate NOR2 A=_27_ B=c_i O=_20_
.gate NOR2 A=a_i B=_19_ O=_21_
.gate NOR2 A=_20_ B=_21_ O=a_o
.gate NAND2 A=a_i B=_19_ O=_22_
.gate OR2 A=d_i B=b_i O=_23_
.gate MOAI1 A=b_i B=_22_ C=_23_ D=c_i O=b_o
.gate NAND2 A=c_i B=b_i O=_24_
.gate NAND2 A=a_i B=_24_ O=_25_
.gate OR2 A=_27_ B=b_i O=_26_
.gate MOAI1 A=_21_ B=_26_ C=_25_ D=_27_ O=c_o
.gate MOAI1 A=d_i B=_19_ C=b_i D=_22_ O=d_o
.end
```

Лінійне перетворення для S_1 мінімальної глибини:

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$$

Реалізація S-блоку у BLIF форматі:

```
.model gost1_depth
.inputs a_i b_i c_i d_i
.outputs a_o b_o c_o d_o
.gate NOT1 A=a_i O=_33_
.gate NOT1 A=c_i O=_20_
.gate NOT1 A=b_i O=_21_
.gate NOT1 A=d_i O=_22_
.gate NAND2 A=a_i B=_21_ O=_23_
.gate NAND2 A=_33_ B=c_i O=_24_
.gate AND2 A=_23_ B=_24_ O=a_o
.gate NOR2 A=c_i B=d_i O=_25_
.gate NOR2 A=a_i B=d_i O=_26_
.gate MOAI1 A=_33_ B=_25_ C=_26_ D=b_i O=b_o
.gate NOR2 A=_20_ B=d_i O=_27_
.gate NAND2 A=_23_ B=_27_ O=_28_
.gate NAND2 A=b_i B=_22_ O=_29_
.gate NOR2 A=a_i B=_21_ O=_30_
.gate NAND2 A=_33_ B=b_i O=_31_
.gate NAND3 A=_20_ B=_29_ C=_31_ O=_32_
.gate NAND2 A=_28_ B=_32_ O=c_o
.gate NOR3 A=_26_ B=_27_ C=_30_ O=d_o
.end
```

A.2 S-блок S_2

Лінійне перетворення для S_2 мінімального гейтового розміру:

$$A = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 1 & 1 & 0 & 1 \end{bmatrix}$$

Реалізація S-блоку у BLIF форматі:

```
.model gost2_ges
.inputs a_i b_i c_i d_i
.outputs a_o b_o c_o d_o
.gate NOT1 A=c_i O=_39_
.gate NOR2 A=b_i B=a_i O=_32_
.gate NAND2 A=d_i B=c_i O=_33_
.gate NOR2 A=b_i B=_33_ O=_34_
.gate NOR2 A=d_i B=c_i O=_35_
.gate NOR2 A=_34_ B=_35_ O=_36_
.gate NOR3 A=_32_ B=_34_ C=_35_ O=a_o
.gate NAND2 A=b_i B=a_i O=_37_
.gate OR2 A=b_i B=c_i O=_38_
.gate NAND2 A=_37_ B=_38_ O=c_o
.gate MAOI1 A=a_i B=_39_ C=c_o D=d_i O=b_o
.gate MOAI1 A=a_i B=_36_ C=_33_ D=b_i O=d_o
.end
```

Лінійне перетворення для S_2 мінімальної глибини:

$$A = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 1 & 1 & 0 & 1 \end{bmatrix}$$

Реалізація S-блоку у BLIF форматі:

```
.model gost2_depth
.inputs a_i b_i c_i d_i
.outputs a_o b_o c_o d_o
.gate NOT1 A=c_i O=_34_
.gate NOT1 A=d_i O=_25_
.gate NOT1 A=a_i O=_26_
.gate NAND2 A=c_i B=d_i O=_27_
.gate NAND2 A=d_i B=b_i O=_28_
.gate MAOI1 A=_34_ B=_28_ C=_27_ D=_26_ O=a_o
.gate NAND2 A=b_i B=_27_ O=_29_
.gate NOR2 A=_34_ B=b_i O=_30_
.gate NAND2 A=_25_ B=a_i O=_31_
.gate MOAI1 A=_30_ B=_31_ C=_26_ D=_29_ O=b_o
.gate NOR2 A=c_i B=a_i O=_32_
.gate OR2 A=_30_ B=_32_ O=c_o
.gate NAND2 A=_34_ B=b_i O=_33_
.gate MOAI1 A=_26_ B=_27_ C=_33_ D=_25_ O=d_o
.end
```

A.3 S-блок S_3

Лінійне перетворення для S_3 мінімального гейтового розміру:

$$A = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}$$

Реалізація S-блоку у BLIF форматі:

```
.model gost3_ges
.inputs a_i b_i c_i d_i
.outputs a_o b_o c_o d_o
.gate NOT1 A=c_i O=_28_
.gate NOT1 A=a_i O=_19_
.gate NAND2 A=b_i B=d_i O=_20_
.gate MOAI1 A=d_i B=_28_ C=_19_ D=_20_ O=a_o
.gate NAND2 A=b_i B=a_i O=_21_
.gate AND2 A=d_i B=_21_ O=_22_
.gate NOR3 A=d_i B=_28_ C=_19_ O=_23_
.gate NOR2 A=_22_ B=_23_ O=b_o
.gate OR2 A=b_i B=d_i O=_24_
.gate NAND3 A=c_i B=_20_ C=_21_ O=_25_
.gate NAND3 A=_28_ B=a_i C=_24_ O=_26_
.gate NAND2 A=_25_ B=_26_ O=c_o
.gate NAND2 A=d_i B=_28_ O=_27_
.gate NAND2 A=_24_ B=_27_ O=d_o
.end
```

Лінійне перетворення для S_3 мінімальної глибини:

$$A = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}$$

Реалізація S-блоку у BLIF форматі:

```
.model gost3_depth
.inputs a_i b_i c_i d_i
.outputs a_o b_o c_o d_o
.gate NOT1 A=a_i O=_33_
.gate NOT1 A=d_i O=_22_
```

```
.gate OR2 A=c_i B=_22_ O=_23_
.gate NAND2 A=b_i B=c_i O=_24_
.gate NAND2 A=_33_ B=_24_ O=_25_
.gate NAND2 A=_23_ B=_25_ O=a_o
.gate NAND2 A=a_i B=b_i O=_26_
.gate AND2 A=c_i B=_26_ O=_27_
.gate NOR3 A=_33_ B=c_i C=_22_ O=_28_
.gate NOR2 A=_27_ B=_28_ O=b_o
.gate OR2 A=b_i B=c_i O=_29_
.gate NAND3 A=d_i B=_24_ C=_26_ O=_30_
.gate NAND3 A=a_i B=_22_ C=_29_ O=_31_
.gate NAND2 A=_30_ B=_31_ O=c_o
.gate NAND2 A=c_i B=_22_ O=_32_
.gate NAND2 A=_29_ B=_32_ O=d_o
.end
```

A.4 S-блок S_4

Лінійне перетворення для S_4 мінімального гейтового розміру:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}$$

Реалізація S-блоку у BLIF форматі:

```
.model gost4_ges
.inputs a_i b_i c_i d_i
.outputs a_o b_o c_o d_o
.gate NOT1 A=b_i O=_28_
.gate NOT1 A=a_i O=_19_
.gate NOT1 A=c_i O=_20_
.gate NOR2 A=a_i B=_20_ O=_21_
.gate NOR2 A=b_i B=c_i O=_22_
.gate NOR2 A=_21_ B=_22_ O=a_o
.gate NOR3 A=_28_ B=a_i C=d_i O=_23_
.gate NAND2 A=_28_ B=d_i O=_24_
.gate NOR2 A=_21_ B=_24_ O=_25_
.gate NOR2 A=_28_ B=_20_ O=_26_
.gate NOR3 A=_23_ B=_25_ C=_26_ O=b_o
.gate AND2 A=a_i B=d_i O=_27_
```

```
.gate MOAI1 A=c_i B=_27_ C=_26_ D=d_i O=c_o
.gate MOAI1 A=_20_ B=d_i C=_24_ D=_19_ O=d_o
.end
```

Лінійне перетворення для S_4 мінімальної глибини:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}$$

Реалізація S-блоку у BLIF форматі:

```
.model gost4_depth
.inputs a_i b_i c_i d_i
.outputs a_o b_o c_o d_o
.gate NOT1 A=a_i O=_28_
.gate NOT1 A=b_i O=_19_
.gate NOT1 A=c_i O=_20_
.gate NOR2 A=b_i B=_20_ O=_21_
.gate NOR2 A=a_i B=c_i O=_22_
.gate NOR2 A=_21_ B=_22_ O=a_o
.gate NOR3 A=_28_ B=b_i C=d_i O=_23_
.gate NAND2 A=_28_ B=d_i O=_24_
.gate NOR2 A=_21_ B=_24_ O=_25_
.gate NOR2 A=_28_ B=_20_ O=_26_
.gate NOR3 A=_23_ B=_25_ C=_26_ O=b_o
.gate AND2 A=b_i B=d_i O=_27_
.gate MOAI1 A=c_i B=_27_ C=_26_ D=d_i O=c_o
.gate MOAI1 A=_20_ B=d_i C=_24_ D=_19_ O=d_o
.end
```

A.5 S-блок S_5

Лінійне перетворення для S_5 мінімального гейтового розміру:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$$

Реалізація S-блоку у BLIF форматі:

```
.model gost5_ges
.inputs a_i b_i c_i d_i
.outputs a_o b_o c_o d_o
.gate NOT1 A=a_i O=_35_
.gate NOT1 A=b_i O=_26_
.gate NOR2 A=d_i B=_26_ O=_27_
.gate NOR2 A=_35_ B=d_i O=_28_
.gate MOAI1 A=a_i B=_27_ C=_28_ D=c_i O=a_o
.gate NAND2 A=a_i B=c_i O=_29_
.gate NOR2 A=a_i B=c_i O=_30_
.gate NAND2 A=b_i B=_29_ O=_31_
.gate NOR2 A=_28_ B=_31_ O=_32_
.gate NOR3 A=d_i B=b_i C=_30_ O=_33_
.gate NOR2 A=_32_ B=_33_ O=b_o
.gate MOAI1 A=_26_ B=c_i C=_29_ D=d_i O=c_o
.gate NOR2 A=_35_ B=b_i O=_34_
.gate OR2 A=_30_ B=_34_ O=d_o
.end
```

Лінійне перетворення для S_5 мінімальної глибини:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$$

Реалізація S-блоку у BLIF форматі:

```
.model gost5_depth
.inputs a_i b_i c_i d_i
.outputs a_o b_o c_o d_o
.gate NOT1 A=c_i O=_34_
.gate NOT1 A=b_i O=_25_
.gate NOR2 A=d_i B=_34_ O=_26_
.gate NOR2 A=d_i B=_25_ O=_27_
.gate MOAI1 A=b_i B=_26_ C=_27_ D=a_i O=a_o
.gate NAND2 A=b_i B=a_i O=_28_
.gate NOR2 A=b_i B=a_i O=_29_
.gate NAND2 A=c_i B=_28_ O=_30_
.gate NOR2 A=_27_ B=_30_ O=_31_
.gate NOR3 A=d_i B=c_i C=_29_ O=_32_
.gate NOR2 A=_31_ B=_32_ O=b_o
.gate MOAI1 A=_34_ B=a_i C=_28_ D=d_i O=c_o
.gate NOR2 A=c_i B=_25_ O=_33_
.gate OR2 A=_29_ B=_33_ O=d_o
```

```
.end
```

A.6 S-блок S_6

Лінійне перетворення для S_6 мінімального гейтового розміру:

$$A = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$$

Реалізація S-блоку у BLIF форматі:

```
.model gost6_ges
.inputs a_i b_i c_i d_i
.outputs a_o b_o c_o d_o
.gate NOT1 A=a_i O=_27_
.gate NAND2 A=d_i B=_27_ O=_20_
.gate NAND2 A=b_i B=a_i O=_21_
.gate NAND2 A=_20_ B=_21_ O=a_o
.gate NAND2 A=c_i B=_21_ O=_22_
.gate NOR2 A=_27_ B=c_i O=_23_
.gate OR2 A=d_i B=b_i O=_24_
.gate MOAI1 A=_23_ B=_24_ C=d_i D=_22_ O=b_o
.gate NOR2 A=b_i B=c_i O=_25_
.gate MOAI1 A=a_i B=_25_ C=_23_ D=d_i O=c_o
.gate NOR2 A=d_i B=c_i O=_26_
.gate MOAI1 A=_27_ B=_26_ C=c_i D=b_i O=d_o
.end
```

Лінійне перетворення для S_6 мінімальної глибини:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$$

Реалізація S-блоку у BLIF форматі:

```
.model gost6_depth
.inputs a_i b_i c_i d_i
.outputs a_o b_o c_o d_o
.gate NOT1 A=d_i O=_31_
```

```
.gate NOT1 A=c_i O=_21_
.gate NAND2 A=b_i B=_21_ O=_22_
.gate NAND2 A=_31_ B=c_i O=_23_
.gate NAND2 A=_22_ B=_23_ O=a_o
.gate AND2 A=c_i B=a_i O=_24_
.gate NOR3 A=_31_ B=b_i C=_24_ O=_25_
.gate NAND2 A=b_i B=c_i O=_26_
.gate NOR2 A=d_i B=_26_ O=_27_
.gate AND2 A=b_i B=a_i O=_28_
.gate NOR3 A=_25_ B=_27_ C=_28_ O=b_o
.gate NOR2 A=_31_ B=c_i O=_29_
.gate AND2 A=a_i B=_26_ O=_30_
.gate OR2 A=_29_ B=_30_ O=c_o
.gate MOAI1 A=_21_ B=_28_ C=_29_ D=a_i O=d_o
.end
```

A.7 S-блок S_7

Лінійне перетворення для S_7 мінімального гейтового розміру:

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$$

Реалізація S-блоку у BLIF форматі:

```
.model gost7_ges
.inputs a_i b_i c_i d_i
.outputs a_o b_o c_o d_o
.gate NOT1 A=c_i O=_31_
.gate NAND2 A=a_i B=_31_ O=_24_
.gate OR2 A=d_i B=_31_ O=_25_
.gate NAND2 A=_24_ B=_25_ O=a_o
.gate NOR2 A=a_i B=b_i O=_26_
.gate NOR2 A=c_i B=b_i O=_27_
.gate MOAI1 A=_31_ B=_26_ C=_27_ D=d_i O=b_o
.gate NOR2 A=d_i B=b_i O=_28_
.gate MOAI1 A=c_i B=_28_ C=b_i D=a_i O=c_o
.gate NOR2 A=a_i B=_27_ O=_29_
.gate AND2 A=d_i B=b_i O=_30_
.gate MOAI1 A=d_i B=_29_ C=_30_ D=_24_ O=d_o
```

```
.end
```

Лінійне перетворення для S_7 мінімальної глибини:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$$

Реалізація S-блоку у BLIF форматі:

```
.model gost7_depth
.inputs a_i b_i c_i d_i
.outputs a_o b_o c_o d_o
.gate NOT1 A=a_i O=_30_
.gate NAND2 A=d_i B=_30_ O=_23_
.gate OR2 A=c_i B=_30_ O=_24_
.gate NAND2 A=_23_ B=_24_ O=a_o
.gate NOR2 A=d_i B=b_i O=_25_
.gate NOR2 A=a_i B=b_i O=_26_
.gate MOAI1 A=_30_ B=_25_ C=_26_ D=c_i O=b_o
.gate NOR2 A=c_i B=b_i O=_27_
.gate MOAI1 A=a_i B=_27_ C=b_i D=d_i O=c_o
.gate NOR2 A=d_i B=_26_ O=_28_
.gate AND2 A=c_i B=b_i O=_29_
.gate MOAI1 A=c_i B=_28_ C=_29_ D=_23_ O=d_o
.end
```

A.8 S-блок S_8

Лінійне перетворення для S_8 мінімального гейтового розміру:

$$A = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 1 & 1 & 0 & 1 \end{bmatrix}$$

Реалізація S-блоку у BLIF форматі:

```
.model gost8_ges
.inputs a_i b_i c_i d_i
.outputs a_o b_o c_o d_o
.gate NAND2 A=c_i B=d_i O=_32_
```

```
.gate OR2 A=d_i B=a_i O=_23_
.gate NAND2 A=d_i B=a_i O=_24_
.gate NAND2 A=_23_ B=_24_ O=_25_
.gate MOAI1 A=c_i B=_25_ C=_32_ D=b_i O=a_o
.gate OR2 A=c_i B=d_i O=_26_
.gate NOR2 A=b_i B=_26_ O=_27_
.gate NAND2 A=c_i B=b_i O=_28_
.gate NAND2 A=_23_ B=_28_ O=_29_
.gate NOR2 A=_27_ B=_29_ O=b_o
.gate NAND2 A=_24_ B=_26_ O=c_o
.gate NAND2 A=_26_ B=_28_ O=_30_
.gate NAND2 A=_32_ B=_26_ O=_31_
.gate MOAI1 A=b_i B=_31_ C=_30_ D=a_i O=d_o
.end
```

Лінійне перетворення для S_8 мінімальної глибини:

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 1 & 1 & 0 & 1 \end{bmatrix}$$

Реалізація S-блоку у BLIF форматі:

```
.model gost8_depth
.inputs a_i b_i c_i d_i
.outputs a_o b_o c_o d_o
.gate NOT1 A=b_i O=_36_
.gate NOT1 A=a_i O=_25_
.gate NOR2 A=b_i B=c_i O=_26_
.gate OR2 A=d_i B=_26_ O=_27_
.gate NAND2 A=d_i B=_25_ O=_28_
.gate NAND3 A=d_i B=_36_ C=_25_ O=_29_
.gate AND2 A=d_i B=b_i O=_30_
.gate NAND2 A=_27_ B=_29_ O=a_o
.gate OR2 A=c_i B=a_i O=_31_
.gate MOAI1 A=d_i B=_25_ C=_31_ D=b_i O=b_o
.gate NAND2 A=c_i B=a_i O=_32_
.gate AND2 A=_28_ B=_32_ O=c_o
.gate NOR2 A=_30_ B=_31_ O=_33_
.gate NOR2 A=d_i B=b_i O=_34_
.gate NOR3 A=_25_ B=_26_ C=_34_ O=_35_
.gate NOR2 A=_33_ B=_35_ O=d_o
.end
```