НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І
СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

«На правах рукопису»
УДК _____

«До захисту допущено»

Завідувач кафедри СПСКС

_____ В.П.Тарасенко_
(підпис) (ініціали, прізвище)

"___"_____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 123 Комп'ютерна інженерія
(Спеціалізовані комп'ютерні системи)

на тему: МЕТОД ФОРМУВАННЯ ПОТОКУ ВХІДНИХ ВЕКТОРІВ ДЛЯ СТАТИСТИЧНИХ ЕКСПЕРИМЕНТІВ З GL-МОДЕЛЯМИ

Виконав: студент ІІ курсу, групи КВ-63м

Ткаченко Михайло Геннадійович _ _____
(підпис)

Науковий керівник к. т. н., доц. каф. СП і СКС Потапова К.Р. _ _____
(підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – другий (магістерський)
Спеціальність 123 Комп'ютерна інженерія
(Спеціалізовані комп'ютерні системи)

ЗАТВЕРДЖУЮ
Завідувач кафедри СПСКС
_____ _В.П.Тарасенко__
(підпис) (ініціали, прізвище)
«___»_____2018р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Ткаченко Михайлу Геннадійовичу

1. Тема дисертації МЕТОД ФОРМУВАННЯ ПОТОКУ ВХІДНИХ ВЕКТОРІВ ДЛЯ СТАТИСТИЧНИХ ЕКСПЕРИМЕНТІВ З GL-МОДЕЛЯМИ, науковий керівник дисертації к. т. н., доц. каф. СП і СКС Потапова К.Р, затверджені наказом по університету від «22» березня 2018 р. №986-с.

2. Термін подання студентом дисертації 14 травня 2018 р.

3. Об'єкт дослідження - процес статистичного тестування відмовостійких багатопроцесорних систем у вигляді GL-моделей.

4. Предмет дослідження - методи, алгоритми формування псевдовипадкових наборів для статистичного тестування.

5. Перелік завдань, які потрібно розробити: алгоритм створення всіх можливих еталонних наборів заданої ваги, комбінаційна схема реалізації певного зсуву вихідного регістра, алгоритм заповнення елемента пам'яті для коректного розподілу ймовірності появи набору заданої ваги, алгоритми тестування вихідної послідовності, програмна реалізація методу.

6. Перелік ілюстративного матеріалу: креслення до патенту на корисну модель, загальна структурна схема апаратної реалізації формувача, діаграма ймовірнісного розподілу появи набору за його вагою, діаграма ймовірності

появи заданого значення на одному з розрядів вихідного регістра, графік кількості повторів кожного з існуючих наборів заданої ваги, діаграма залежності похибки ймовірності появи заданого значення на одному розряді набору від кількості еталонних наборів.

7. Перелік публікацій: наукова конференція магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2018 (Київ, 21-23 березня 2018 р.) - тези, патент ДП "Український інститут промислової власності" №119842 (Київ, Бюл. №19, 10 жовтня 2017 р.)

8. Дата видачі завдання 5 вересня 2017 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
	Видача завдання на магістерську дисертацію	05.09.2017	
	Розробка технічного завдання	16.10.2017	
	Аналіз існуючих рішень	17.12.2017	
	Підготовка матеріалів першого розділу магістерської дисертації	30.02.2017	
	Підготовка матеріалів другого розділу магістерської дисертації	15.03.2018	
	Підготовка графічної частини магістерської дисертації	10.04.2018	
	Оформлення документації магістерської дисертації	23.04.2018	
	Попередній розгляд магістерської дисертації на кафедрі	26.04.2018	

Студент _____ Ткаченко М.Г.
(підпис)

Науковий керівник дисертації _____ Потапова К.Р.
(підпис)

РЕФЕРАТ

Актуальність теми. Незважаючи на наявність численних робіт з технічних засобів формування випадкових та псевдовипадкових процесів, значущих результатів, орієнтованих на цифрову реалізацію, отримано не було. Елементи теорії цифрових генераторів і методи їх інженерних додатків в задачах побудови механізмів формування випадкових і псевдовипадкових процесів на цифровий елементній базі є внеском у вирішення науково-технічної проблеми, актуальної для областей статистичного моделювання, захисту інформації та інших галузей. Зокрема, визначення характеристик відмовостійких багатопроцесорних систем, представлених у вигляді GL-меделей, за допомогою статистичного тестування потребує якісних джерел псевдовипадкових вхідних даних.

Об’єктом дослідження є процес статистичного тестування відмовостійких багатопроцесорних систем у вигляді GL-моделей.

Предметом дослідження є методи, алгоритми формування псевдовипадкових наборів для статистичного тестування.

Мета роботи: розробка методів та засобів, які дають змогу розробнику відмовостійкої багатопроцесорної системи за допомогою її GL-моделі оцінити статистичні характеристики шляхом тестування вхідними наборами.

Наукова новизна полягає в наступному:

1. Створено метод формування послідовностей багато-розрядних псевдовипадкових двійкових наборів заданої ваги. .
2. Розроблено спосіб отримання будь-якого розподілу ймовірності появи псевдовипадкових двійкових наборів заданих ваг при їх генеруванні.

Практична цінність отриманих в роботі результатів полягає в тому, що запропоновані методи дають змогу визначити характеристики функціонування відмовостійких багатопроцесорних систем за допомогою статистичного тестування вхідними наборами GL-моделі. Теоретичні та практичні результати роботи можуть бути використані в організаціях, що займаються проектуванням, експлуатацією та оцінкою показників надійності відмовостійких багатопроцесорних обчислювальних систем.

Апробація роботи. Основні положення і результати роботи були представлені та обговорювались на науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2018 (Київ, 21-23 березня 2018 р.). Опис методу був представлений у патенті ДП "Український інститут промислової власності" №119842 (Київ, Бюл. №19, 10 жовтня 2017 р.).

Структура та обсяг роботи. Магістерська дисертація складається з вступу, трьох розділів та висновків.

У вступі подано загальну характеристику роботи, зроблено оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень, сформульовано мету і задачі досліджень, показано наукову новизну отриманих результатів і практичну цінність роботи, наведено відомості про апробацію результатів.

У першому розділі розглянуто існуючі генератори псевдовипадкових наборів, а також проведений аналіз, який дає змогу визначити основні переваги та недоліки цих генераторів. Розглянуті характеристики, за якими відбувається тестування псевдовипадкових послідовностей. Описані основні визначення для GL-моделі.

У другому розділі наведено опис розробленого методу генерації послідовностей псевдовипадкових двійкових наборів та його програмної реалізації.

У третьому розділі проводиться аналіз характеристик отриманих послідовностей псевдовипадкових двійкових наборів.

У висновках представлені результати проведеної роботи, дані рекомендації до вхідних параметрів формувача.

Робота представлена на 83 аркушах, містить посилання на список використаних літературних джерел.

Ключові слова: відмовостійкість, генератор, псевдовипадковий двійковий набір.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

АТД – абстрактний тип даних.

БС – бістабільна схема.

ГВП – генератор випадкових послідовностей.

ГПСЧ – генератор псевдовипадкових чисел.

КН – кінцевий набір.

НСД – найбільший спільний дільник.

ПВП – первинний вимірювальний перетворювач.

ПК – персональний комп’ютер.

ПЗ – програмне забезпечення.

ПСП – псевдовипадкова послідовність.

СУ – схема управління.

ІБР – ймовірність безвідмовної роботи.

C++ – мова програмування високого рівня.

LFSR – лінійний зсувний регістр зі зворотним зв'язком.

M-S – master-slave, тип тригеру.

MS Visual Studio 2013 Ultimate – середовище розробки додатку.

NIST – Національний інститут стандартів і технологій.

ВСТУП

Випадкові числа грають важливу роль в криптографії її різних додатках, а також при рішенні проблеми моделювання багатопроцесорних систем. Ці числа падвються у вигляді вихідних двійкових наборів.

Основні критеріі послідовності наборів:

- непередбачуваність;
- випадковість.

При генеруванні послідовності двійкових наборів, з точки зору статистики, вони повинні бути випадковим. Для доказу того, що дана послідовність є випадковою, використовуються 2 критерії[1]:

- однорідність розподілу;
- незалежність наборів.

При однорідному розподілі частота появи кожного набору повинна бути приблизно однаковою.

Незалежність або непередбачуваність послідовності виражається у відсутності статистичних зв’язків між різними двійковими. Тобто неможливо, знаючи попередні елементи послідовності і алгоритм, передбачити наступний елемент.

Існують тести, які показують, що послідовність чисел відповідає деякому розподілу, такому як однорідний розподіл, але тесту для "доказу" незалежності немає. Проте, можна підібрати набір тестів для доказу того, що послідовність є залежною. Загальна стратегія передбачає застосування набору таких тестів до тих пір, поки не буде впевненості, що незалежність існує.

Послідовність називається випадковою, якщо її не можна відтворити. Це означає, що якщо запустити генератор випадкових послідовностей двічі

при одному і тому ж вході, то на його виході вийдуть різні випадкові послідовності.

Реалізація джерела таких випадкових подій затратні по ресурсам. Прикладами фізичних генераторів шумів є:

- газові розрядні трубки;
- детектори радіації;
- конденсатори.

Однак застосування цих пристроїв обмежено. Одна з альтернатив – це створення множини наборів з певного числа вже згенерованих випадкових послідовностей та публікація її в літературі. Але невеликий розмір множини таких наборів являє собою обмежене джерело. Хоча набори з такої літератури дійсно забезпечує статистичну випадковість, вони передбачувані, так як можна отримати їх копію.

Отже, для генератору обрані псевдовипадкові набори через можливість повторів тестів та зменшення кількості ресурсів для реалізації формувача.

Для створення псевдовипадкових чисел використовуються спеціалізовані алгоритми. Незважаючи на те, що алгоритми є детермінованими (послідовність наборів, яка формується, не є статистично випадковою), можна створити оптимальний алгоритм. Ця оптимальність визначається тим, що отримана послідовність наборів у більшості тестів на випадковість мати позитивний результат. Набори, згенеровані такими алгоритмами називають псевдовипадковими наборами.

Створенню якісних генераторів псевдовипадкових послідовностей приділяється чимала увага в математиці. Проблема в тому, що всі генератори псевдовипадкових послідовностей за певних умов дають передбачувані результати і кореляційні залежності.

Створення псевдовипадкової послідовності з допомогою суто детермінованого процесу здійснюється силами псевдовипадкового

генератора. У більшості таких генераторів послідовностей встановлюється початковий стан, а в слід за цим з допомогою певних алгоритмів генеруються з нього випадкові послідовності двійкових наборів.

Комп'ютери - це детерміновані машини, що виконинують лише ті дії, які описані програмно. Цей критерій робить неможлим використовувати комп'ютери як джерело істинної випадковості. У найліпшому випадку комп'ютер здатний згенерувати псевдовипадкову послідовність наборів, яка на погляд користувача виглядає випадковою, але такою не являється.

Для генерування дійсно випадкової послідовності за допомогою комп'ютера необхідно використовувати його апаратні засоби. Ці засоби можуть фіксувати наступні явища;

- шум від напівпровідникових приладів;
- біти оцифрованого звуку з мікрофону;
- інтервали між перериванням зовнішніх або внутрішніх пристроїв;
- інтервали між натисканням клавіш;
- температура повітря на апаратних складових.

Також існують генератори випадкових чисел у вигляді плат або зовнішніх пристроїв. Такі генератори виористовуються в сучасних криптосистемах для військових. Вони підключаються до комп’ютерів за допомогою портів вводу-виводу. Основні джерела для них слугують:

- білий Гаусівський шум;
- запис радіоефіру;
- виміри теплових флуктуацій.

Незважаючи на те, що для проектування генераторів псевдовипадкових наборів необхідне комплексе тестування на випадковість, вони в багатьох випадках знаходять використання в комп'ютерних програмах прикладного характеру. Також можуть бути реалізовані для будь-яких типів комп'ютерних систем.

Метою цієї кваліфікаційної роботи є створення методу, а також його програмна реалізація, що здійснює генерування і тестування псевдовипадкових послідовностей.

Далі будуть розглянуті та наведені:

- теоретичні основи псевдовипадкових послідовностей;
- описи генераторів псевдовипадкових послідовностей;
- тестування псевдовипадкових послідовностей;
- опис роботи розробленої програми;
- експериментальні результати виконаної розробки.

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ АЛГОРИТМІВ ГЕНЕРУВАННЯ ПОСЛІДОВНОСТЕЙ ТА ЇХ ТЕСТУВАННЯ

Значне застосування генераторів псевдовипадкових наборів набуло в багатьох галузях. Серед них особливе значення становлять в областях, які використовують електронну та електронно-обчислювальну техніку. На Рисунку 1.1 відображені основні сфери використання ГПВЧ.

Рисунок 1.1 - Основні сфери застосування генераторів

Одна з сфер застосування генераторів псевдовипадкових наборів - це захист приватності інформації. За допомогою загальних та приватних мереж передачі даних з часом все більш зростає кількість інформації, яка зберігається та обмінюється між комп'ютерами. Отже, питання приватності інформації становиться все більш актуальним. Блок генератору випадкових чисел (ГВЧ) – стандартний блок безпеки.

Для забезпечення конфіденційності передачі даних використовуються випадкові числа. Вони є основою для наступних елементів криптографії:

- цифровий підпис;
- протоколи безпеки;
- інше забезпечення надійності при передачі даних через комп'ютер.

Інша галузь застосування генераторів випадкових наборів - імітаційне моделювання. В більшості випадків необхідна кількість різних послідовночтей випадкових наборів перевищує одну послідовність. Наприклад, для отримання статистично незалежніх розультатів на кожному з процесорів ВБС необхідно запустити однакову програмну розробку, але використовуючи різні псевдовипадкові послідовності. Після цього результати повинні бути усереднені.

Використовуючи ж детермінований алгоритм генерування випадкових послідовностей, також є свої переваги. Наприклад, при виникненні необхідності щоб послідовність псевдовипадкових наборів існувала можливість повтору досліджень заданої задачі з різними вхідними параметрема. Така необхідність існує для наступних процесів;

- автоматизація телефонних ліній ;
- тестування засобів управління дорожнім рухом.

При роботі з ГВЧ така можливість відсутня.

Галузь застосувань псевдовипадкові генератори для імітаційного моделювання являє собою не тільки фізичні галузі. Наприклад, в біологічних дослідженнях [2] він знаходить застосування в імітації мутацій.

Одна з важливих галузей застосування генераторів випадкових послідновностей – виробництво апаратних складових. Функції використання:

- контроль якості друкованих плат;
- тестування окремих модулів;
- тестування функцій пристроїв в цілому.

В сучасному світі це питання розглядається на такому ж рівні, як і при вирішенні проблем, пов'язаних із імітаційним моделюванням та захистом інформації.

Наприклад, нові методики дослідження аналогивих схем та модулів з радіокомпонентів представлені в [3].

Використання білого цифрового шуму при обмеженій смузі пропускання як генератора вхідного шуму для тестування описано в [4]. Характеристика визначається за допомогою взаємної кореляції між вхідною послідовністю псевдовипадкових даних та реакцією виходу схеми.

Схеми з вбудованою самоперевіркою для реалізації операції сканування на основі лінійних зсувних регістрів для генерування псевдовипадкових наборів розглянуті в [5]. При присутності в об’єктах досліджень дефектів, такі схеми сканування мають високе значення їх покриття.

Під надійністю використання псевдовипадкових генраторів мають на увазі незалежність кожного наступного набору від згенерованих наборів до нього. Наприклад, у випадку електронного автомату, якщо для гравця існує можливість визначити значення для наступних обертів на основі аналізу моделі попередніх значень, то прибуток від нього буде від’ємний. Подібна ситуація для кодування повідомлень, тобто необхідно, щоб при розкритті частини розсекреченого повідомлення неможливо було розшифрувати всю іншу інформацію.

Псевдовипадкові набори мають широке застосування й в сучасній інформатиці, наприклад, для різних математичних методів, таких як методу Монте-Карло. При цьому від якості використовуваних генераторів псевдовипадкових чисел безпосередньо залежить якість отримуваних результатів. Будь-який ГПСЧ з обмеженими ресурсами рано чи пізно зациклюється, тобто певна впорядкована множина наборів починає повторюватися. Довжина такої множини називається періодом ГПСЧ. Період залежить від власне схеми генератора і в середньому дорівнює $2^{n/2}$, де n - розмір внутрішніх статків у бітах, хоча лінійні конгруентні генератори і генератори на основі зсувних регістрів мають максимальні періоди близько 2^n. Якщо ГПСЧ для певних задач має замалий період, то такий генератор стає передбачуваним і його використання не є доцільним.

Більша частина арифметичних генераторів має перевагу в швидкодії, але й страждає від настпнух недоліків:

- не достатньо тривалий період генератора;

- взаємнозалежність значень наборів;

- оборотність

- відсутність рівномірного розподілу ймовірності появи значення на будь-якому біті набора;

- відсутність рівномірного розподілу власне вихідних наборів.

Генератори псевдовипадкових послідовностей бувають апаратними (на основі зсувних регістрів) і програмними (детерміновані генератори, генератори з джерелом ентропії).

Класифікація генераторів псевдовипадкових послідовностей наочно представлена на наступному Рисунку 1.2.

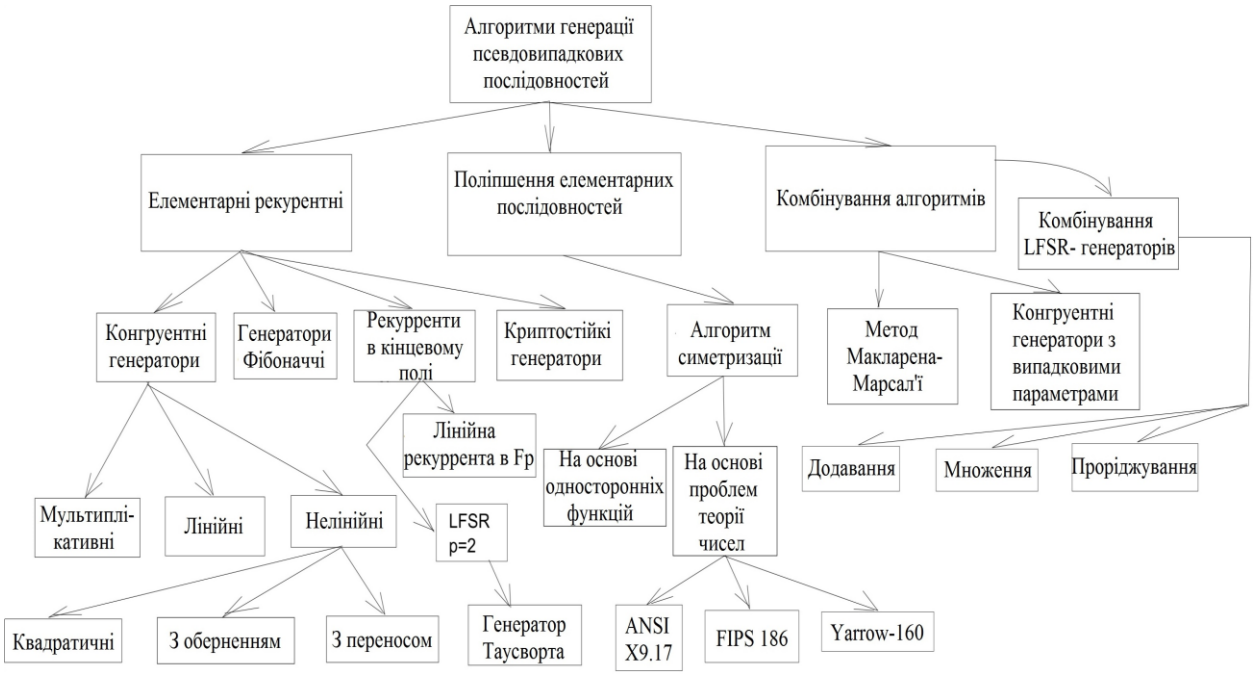


Рисунок 1.2 - Класифікація генераторів ПСП

1.1 Класичний генератор послідовностей

Класичним представником генератора випадкових послідовностей (ГВП) є генератор випадкових послідовностей на основі фізичних явищ(фізичний). Фізичний ГВП представляє собою пристрій, який генерує випадкові послідовності на основі наступних фізичних процесів:

- фотоелектричний ефект;
- тепловий шум;
- квантові явища і т.д.

Виміряні значення таких процесів є величинами абсолютно непередбачуваними.

Апаратна частина, що фіксує фізичні явища називається фізичним джерелом шуму. В більшості випадків значення отримуються за допомогою оцифровки(дискретизації) аналового сигналу з джерела шуму. Дискретизовані значення від фізичного джерела при обробці перетворюються на випадкову послідовність наборів. Ця обробка

відбувається для зменшення похибки розподілу ймовірності появи відповідних значень дискретизованої величини.

Також існує можливість подання дискретизованиого значення фізичного джерела шуму безпосередньо у вихідний блок не виконуючи додаткові операції. Тобто, в такому випадку, результативна випадкова послідовність наборів повністю відповідає дискретизованій послідовності значень від джерела шуму. Для отримання кінцевої випадкової послідовності наборів після кожної операції дискретизації(або декількох операцій при збереженні значень в пам’яті) використовюється сигнал синхронізації(безперервний або аперіодичний). Ентропія випадкової послідовності, що видається джерелом шуму, підвищується з генеруванням кожного символу випадкової послідовності. Генератори випадкових послідовностей можуть будуватись не тільки на основі фізичних явищ, а й комплексно з додатковим програмним продуктом.

На Рисунку 1.3 показана функціональна блок-схема ГВП, яка визначена міжнародним стандартом ISO/IEC 18031[6]. В залежності від ряду факторів деякі компоненти можуть бути присутніми або відсутніми у складі ГВП. Такі компоненти на блок-схемі позначені пунктирними лініями. Також існує можливість реалізації всіх вказаних компонентів не тільки у фізичній формі. Головна умова для цього – всі компоненти повинні бути реалізовані функціонально. До кожного з компонентів у стандарті висуваються вимоги.

Прикладами фізичних явищ можуть бути:

- зняття даних температури з діода пристрою;
- фіксація характеристик радіоактивного розпаду;
- частотна нестійкість осцилятора в режимі вільних генерацій;
- конденсатор, який за заданий період часу повністю заряджається.

Для реалізації генараторів на основі перших двох явищ необхідні зовнішні пристрої. Але такі пристрої швидко піддаються спостереженням,

крім цього можливі й шкідливі маніпуляції зловмисників. Саме тому доцільніше використовувати осцилятори і конденсатори як джерело фізичного шуму, через те, що такці пристою можуть бути реалізовані за допомогою надвисоких інтегральних схем. Такі схеми захищені від несанкціонованого втручання.

Якщо ж розглядати ГВП з використанням програмних продуктів, то в якості джерела шуму використовують:

- час, що минув між натисканнями клавіші або переміщенням миші;
- системний годинник комп'ютера;
- зміст буферів вводу/виводу;
- системне навантаження;
- мережеві статистичні дані;
- інші параметри операційної системи.

Рисунок 1.3 - Блок-схема генератора випадкових послідовностей

Генератори випадкових послідовностей потребують більше ресурсів для реалізації, тому для роботи обрано генератор псевдовипадкових послідовностей.

1.2 Методи отримання псевдовипадкових послідовностей

Розглянемо наступні генератори псевдовипадкових послідовностей: лінійний конгруентний генератор, генератор Фібоначчі, вихор Мерсенна, RSA генератор ПСП, зсувний регістр як генератор ПСП і самоврядний 2-лінійний регістр зсуву.

1.2.1 Метод Фібоначчі

Клас генераторів псевдовипадкових послідовностей заснований на використанні послідовностей Фібоначчі[7]. Приклад такої послідовності - {1, 1, 2, 3, 5, 8, 13, 21, 34, 55б 89, ...}. Перші два члени дорівнюють 0 та 1 чи 1 та 1 відповідно. Кожен з наступних членів дорівнює сумі двох попередніх значень послідовності.

Широке застосування генератори послідовностей на основі чисел Фібоначчі отримали в зв'язку з високою швидкодією виконання арифметичних операцій з числами, що беруть участь для формування послідовності. Значення цього параметру порівняне з швидкістю цілочисельної арифметики, а генератори Фібоначчі природно реалізуються в речовій арифметиці.

Ітеративна формула (1.1) представляє собою один з генераторів сімейства Фібоначчі.

$$X_k = \begin{cases} X_{k-a} - X_{k-b}, & \text{if } X_{k-a} \geq X_{k-b}; \\ X_{k-a} - X_{k-b} + 1, & \text{if } X_{k-a} < X_{k-b}; \end{cases} \tag{1.1}$$

Де Xk - числа з діапазону [0,1); a та b - цілі позитивні числа(їх також називають лагами). Для роботи алгоритму необхідні обрахунки max {a, b} для випадкових чисел, які були згенеровані на попередніх кроках.

Один з варіантів програмної реалізації передбачає зберігання згенерованих випадкових чисел у вигляді кінцевої циклічної черги за допомогою вказівників, або з використанням стандартного масиву.

Вхідні дані, які забезбечують одні з кращих результатів роботи генератора для пари (a, b):

- (17,5);
- (55,24);
- (97,33).

Значення константи впливає на якість одержуваних випадкових чисел. З його зростанням збільшується розмірність простору, в якому зберігається рівномірний розподіл ймовірності появи псевдовипадкових векторів, які утворені з отриманих псевдовипадкових наборів.

Але потрібно враховувати й обмеження по пам’яті пристрою. Зі зростанням величини константи збільшується й обсяг пам'яті, якої потрібно виділити на роботу алгоритму.

Значення вхідних параметрів (17,5) доцільно задавати для простих прикладних додатків, які не вимагають генерування псевдовипадкових векторів з високою розрядністю.

Значення вхідних параметрів (55,24) задовільняють умовам якості псевдовипадкових послідовностей для більшості алгоритмів.

Значення вхідних параметрів (97,33) дозволяють отримати псевдовипадкові набори, які можуть бути використані в алгоритмах, що працюють з псевдовипадковими векторами високої розрядності.

Описаний генератор Фібоначчі випадкових чисел (з a та b, які дорівнюють 20 і 5 відповідно) використовується в широко відомій системі Matlab.

Результативні псевдовипадкові послідовності задовільняють умовам рівномірності розподілу ймовірності появи наборів та задовільняють іншим статистичним тестам.

Період таких генераторів розраховується за формулою (1.2), де l - число розрядів мантиси дійсного числа.

$$T = (2^{\max\{a,b\}} - 1) \cdot 2^{\ell} \tag{1.2}$$

1.2.2 Вихор Мерсенна

Вихор Мерсенна - це генератор псевдовипадкових чисел, оснований на властивостях простих чисел Мерсенна і забезпечує швидке генерування високоякісних псевдовипадкових чисел[8]. Вихор Мерсенна позбавлений наступних недоліків:

- малий період;
- передбачуваність;
- виявлення статистична залежностей як затратна операція.

Проте цей генератор не відповідає вимогам криптостійкості, що обмежує його використання в криптографії.

Вихор Мерсенна представляє собою витковий регістр зсуву з узагальненою віддачею. «Вихор» - це перетворення, яке забезпечує рівномірний розподіл. Генерування псевдовипадкових наборів відбувається в 623 вимірах. Саме тому у послідовності наборів, згенерованої за допомогою Вихора Мерсенна, кореляція між послідовними значеннями нехтовно мала.

Одна з переваг вихору Мерсенна - великий період псевдовипадкової послідовності. Значення її періоду на одиницю менше, ніж число Мерсенна - 219937. Такого значення достатньо в більшості випадків для практичного застосування.

Деякі реалізації вихора Мерсенна за швидкодією перевищуються в 2-3 рази лінійні конгруентні методи. За цим критерієм генератор перевершує більшість стандартних ГПСЧ.

Програмна реалізація вихора Мерсенна створена в окремій бібліотеці gLib та стандартних бібліотеках для мов:

- PHP;
- Python;
- Ruby.

Алгоритм заснований на рекуррентному вираженні (1.3), де n - ціле число, яке позначає ступінь рекурентності, m – ціле число в діапазоні 1<m<n, A – матриця розміру WxW.

$$x_{k+n} = x_{k+m} \oplus \left(x_k^u | x_{k+1}^l\right)A, (k = 0,1, \dots) \quad (1.3)$$

Значення x_k^u позначає старші біти, x_k^u і x_{k+1}^l - молодші біти.

1.2.3 Лінійний зсувний регістр зі зворотним зв'язком

Регістри зсуву або зсувні регістри представляють собою послідовно з'єднаний ланцюг тригерів[9].

Приклад такого генератора відображений на Рисунку 1.4.

Основний режим роботи таких генераторів - це логічний зсув значень, які записані в кожен з тригерів регістра. Тобто після синхронізуючого сигналу значення кожного попереднього тригера регістра записується в наступний по порядку в ланцюгу тригерів. Набір значень, що зберігається в регістрі, на кожному такті зсувається на один розряд в бік старших розрядів(shift left) або в бік молодших розрядів(shift right).

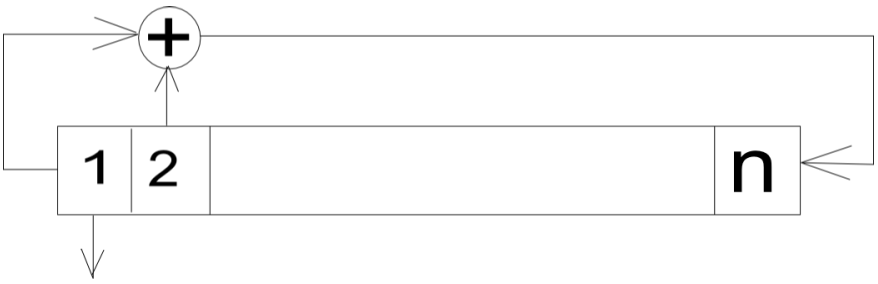


Рисунок 1.4 - Лінійний зсувний регістр зі зворотним зв'язком (LFSR)

Складові частини зсувного регістра зі зворотним зв'язком:

- власне зсувний регістр;
- функція зворотного зв'язку.

Довжина зсувного регістра - кількість бітів(тригерів). Коли потрібно «витягти» біт, все біти зсувного регістра зсуваються вправо або вліво на одну позицію. Нове значення крайнього біта визначається функцією, аргументами якої являються інші біти регістра. В більшості випадків значущим бітом вихідного регістра є молодший біт.

Період зсувного регістра - довжина одержуваної послідовності до початку її повторення.

Для LFSR функція зворотного зв'язку представляє собою операцію суми по модулю 2 (тобто xor) деяких бітів регістра (ці біти називаються відвідною послідовністю). LFSR може перебувати у внутрішніх станах, де n - довжина зсувного регістра. Якщо значення всіх бітів зсувного регістра - нулі, то в результаті такого стану результативними бітами на виході будуть тільки нулі (так як в якості опції зворотного зв'язку використовується xor), тому такий стан не є доцільним. Теоретично LFSR може генерувати послідовність з довжиною біт $2^n - 1$, тому що довжина псевдовипадкової послідовності збігається внутрішніми станами вихідного регістра. LFSR згенерує всі можливі комбінації вихідних наборів(матиме максимальний період) тільки при певних відвідних послідовностях. Наприклад,

многочлен, що утворений з константи 1 та відвідної послідовності - примітив по модулю два.

Ступінь многочлена - довжина зсувного регістра. Примітивний многочлен ступеня n - це многочлен, що не приводиться, який є дільником $x^{2^n-1}+1$, але не є дільником x^d+1 для всіх d, що діляють 2^n-1.

Переваги генератора ПСП, заснованого на зсувному регістрі:

- ефективна апаратна реалізація;
- швидкодія.

Недоліком генераторів на основі регістрів утворюють тільки циклічні послідовності чисел. Для отримання нециклічних послідовностей необхідно під'єднати на виході генератора комбінаційний перетворювач кодів. Але при цьому погіршуються основні параметри генератора:

- швидкодія;
- потужність;
- розмір площі кристалу.

1.2.4 Самоврядний 2-лінійний регістр зсуву

Приклад такого генератора приведений на Рисунку 1.5[10].

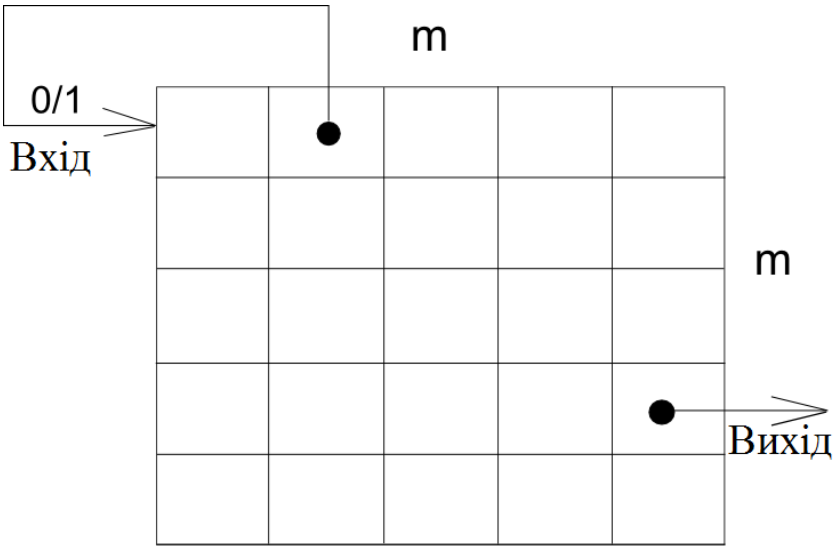


Рисунок 1.5 - Самоврядний 2-лінійний регістр зсуву

Візьмемо два неприведених многочлена $F_1(x)$ і $F_2(x)$, причому діє рівність (1.4), і складемо відповідну матрицю розміром m * m.

$$degF_1(x) = degF_2(x) = m \tag{1.4}$$

На вхід генератора подається або "0", або "1".

Стовбець та рядок матриці генератора відображені на Рисунку 1.6 та Рисунку 1.7 відповідно.

Якщо на вхід генератора подається "0", то всі рядки генератора перетворюються відповідно до заданого законом рекурсії $F_1(x)$.

Якщо на вхід генератора подається "1", то всі стовпці матриці перетворюються в відповідно до закону рекурсії $F_2(x)$.

Початкове заповнення генератора - будь-яке, окрім нульового.

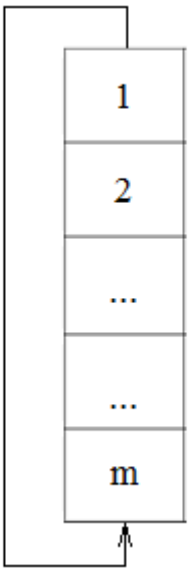


Рисунок 1.6 - Стовбець матриці генератора

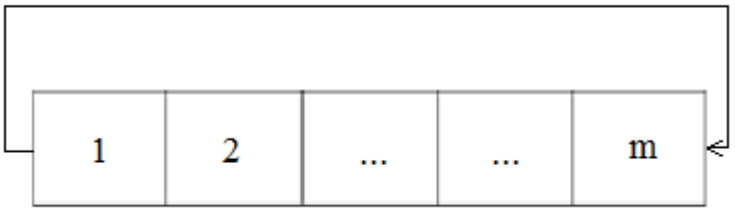


Рисунок 1.7 - Рядок матриці генератора

1.2.5 RSA-алгоритм генерування псевдовипадкових послідовностей

RSA-алгоритм генерування двійкової псевдовипадкової послідовності $x_1, x_2, ..., x_n \in A$ складається з шести кроків[11].

1) Згенерувати чималі прості різні числа p, q та вирахувати числа відповідно (1.5) та (1.6),. Вибрати випадкове ціле число k, при чому (1.7), взаємно просте з φ, так, що виконується рівність (1.8).

$$N = pq \tag{1.5}$$

$$\varphi = (p-1)(q-1) \tag{1.6}$$

$$1 < k < \varphi \tag{1.7}$$

НСД(k, φ) = 1 (1.8)

2) Обрати випадкове ціле – стартове значення - $u_0 \in \{1,2,...,N-1\}$.

3) Для $i = \overline{1,n}$ виконати кроки 4,5.

4) Обрахувати (1.9).

$$u_i = u_{i-1}^k \bmod N \in \{0,1,...,N-1\}. \tag{1.9}$$

5) Обрахувати $x_i \in A = \{0,1\}$ - наймолодший біт числа u_i у двійковому представленні.

6) Сформувати вихідну послідовність $x_1, x_2, ..., x_n$.

Недоліком RSA - алгоритму являється його невисока швидкодія при реалізації на універсальних комп'ютерах, викликане великими витратами машинного часу при виконанні модулярного множення на кроці 4.

1.2.6 Лінійний конгруентний генератор

Лінійний конгруентний метод є однією з найпростіших і найбільш уживаних в даний час процедур, що імітують випадкові числа[12]. У цьому методі використовується операція mod (x, y), результатом якої є залишок від ділення x на y. Кожний наступний член псевдовипадкової послідовності залежить від попереднього. Ітеративна формула для розрахунку (i+1) члена представлена в (1.10), де D - дільник (D > 0); k - множник (0 ≤ k < D); s - приріст (0 ≤ s < D);

$$r_{i+1} = mod(k * r_i + s, D) \tag{1.10}$$

Послідовність випадкових чисел, отриманих за допомогою цієї формули, називається лінійної конгруентної послідовністю. Більшість авторів називають лінійну конгруентністю послідовність при b = 0 мультиплікативний конгруентним методом. У випадку s ≠ 0 послідовність називають змішаним конгруентним методом.

Для того, щоб статистичні характеристики генератора на основі вищевказаного методу відповідали заданим, необхідні критерії для визначення коефіцієнтів. Один з критеріїв – значення змінної D. Необхідно, щоб її значення було досить великим, оскільки період послідовності не перевищує D елементів. Але для отримання результату

операції визначення залишку використовується ділення, що є самою повільною арифметичною операцією. Саме тому для ЕОМ з двійковою системою числення є доцільним вибір D = 2N. В цьому випадку для ЕОМ команда знаходження залишку від ділення зводиться до логічної операції «AND».

Одним з варіантів визначення значення D – використання простих чисел. Таке просте число повинно бути меншим, ніж 2N. В спеціалізованій літературі є докази того, що в такому випадку молодші розряди результативного випадкового числа r (i+1)-го мають однаковий випадковий характер, як і старші його розряди. Для всієї послідовності псевдовипадкових чисел це позитивно позначається.

Як приклад можна навести одне з чисел Мерсенна, рівне 511(512 - 1), і таким чином, D = 511.

Збільшення довжини періоду псевдовипадкової послідовності – одна з головних вимог до лінійних конгруентних послідовностей. Довжина періоду таких послідовностей залежить від значень D, k і s.

Теорема, яка наведена нижче, дозволяє визначити, чи можливе досягнення періоду максимальної довжини для конкретних значень D, k і s.

Теорема. Лінійна конгруентністю послідовність, яка задана значенням чисел D, k, s і r0(початкове значення), має період довжиною D тоді і тільки тоді, коли:

- числа s і D взаємно прості;
- (k – 1) ⋮ p для для будь-якого простого значення p, якщо D ⋮ p;
- (k – 1) ⋮ 4, якщо D ⋮ 4.

Швидкодія – це головна перевага лінійних конгруентних генераторів. Це відбувається через малу кількість операцій на біт псевдовипадкової послідовності.

Недоліком таких генераторів є передбачуваність послідовностей. Варто відмітити, що лінійні конгруєнтні генератори в сучасний час

використовуються для задач не пов’язаних з криптографією. Прикладом такої задачі є симулювання випадкової поведінки. Такі псевдовипадкові послідовності після генерування демонструють на тестуванні значення статистичних властивостей із однією з найменших похибок для генераторів. Ці результати спостерігаються при виконанні більшості емпіричних тестів.

1.3 Генерування послідовності за законом розподілу

Для імітації псевдовипадкових дій будь-якої складності за допомогою моделювання на ЕОМ необхідно виконати два кроки[13]:

- генерація стандартного базового процесу;
- функціональне перетворення.

Для вибору базового процесу не існує критеріїв, тому обирається той, який зручніший для моделювання. Для ЕОМ базовий процес представляє собою послідовність чисел $\{x_j\} = x_0, x_1, \ldots, x_n$. Ця послідовність відповідає наступним критеріям:

- незалежність;
- рівномірной розподіл в інтервалі (0,1).

Розподіл моделюється з функцією густини (1.11), інтервальною функцією розподілу (1.12), математичним сподіванням (1.13) та дисперсією (1.14).

$$f(x) = \begin{cases} 1, 0 \leq x \leq 1 \\ 0, x < 0 \cup x > 1 \end{cases} \tag{1.11}$$

$$F(x) = \begin{cases} 0, x < 0 \\ x, 0 \leq x \leq 1 \\ 1, x > 1 \end{cases} \tag{1.12}$$

$$M[\xi] = \int_0^1 x\, f(x)dx = \frac{1}{2} \tag{1.13}$$

$$D[\xi] = \int_0^1 (x - M[\xi])^2 f(x)dx = \frac{1}{12} \tag{1.14}$$

На цифрових ЕОМ одержати як результат такий розподіл не представляється можливим, тому що система оперує з n–розрядними наборами з певним інтервалом дискретності.

Через це для цифрових ЕОМ замість неперервної сукупності n–розрядних наборів з рівномірним розподілом значень в інтервалі (0,1) випадкових чисел набуло використання дискретної послідовності з 2^n псевдовипадкових чисел з цього ж інтервалу. Таким чином з'являється можливість змоделювати квазірівномірний розподіл. Випадкова величина ξ з квазірівномірним розподілом значень в інтервалі (0,1), приймає значення (1.15) з ймовірністю (1.16). Математичне сподівання та дисперсія величин ξ (1.17) та (1.18) відповідно.

$$x_i = i/(2^n - 1) \tag{1.15}$$

$$P_i = (1/2)^n,\ i = 0,2^n - 1. \tag{1.16}$$

$$M[\xi] = \sum_{i=0}^{2^n-1} \frac{i}{2^n - 1}\frac{1}{2^n} = \frac{1}{2} \tag{1.17}$$

$$D[\xi] = \sum_{i=0}^{2^n-1} \frac{1}{2^n}\left[\frac{i}{2^n - 1} - \frac{1}{2}\right]^2 = \frac{1}{12}\frac{2^n + 1}{2^n - 1} \tag{1.18}$$

Наслідком дискретного подання неперервних чисел та періодичності генерованої за допомогою алгоритмів послідовності є відсутність можливості генерування послідовності дійсно випадкових наборів на ЕОМ. Тому генератори, які реалізовані за допомогою програмних засобів генерують лише псевдовипадкові набори.

Для генераторів псевдовипадкових послідовностей наборів встановлюються наступні необхідні умови:

- квазірівномірність розподілу ймовірності появи наборів;
- існування можливісті відтворення послідовності псевдовипадкових наборів;
- статистична незалежність наборів (тобто відсутність їх кореляції).

Додаткові умови реалізації генераторів програмним способом:

- мінімальні витратами часу;
- мінімальний об’єм пам’яті для змінних.

Для генерування псевдовипадкових послідовностей наборів у випадках моделювання систем на практиці використовуються рекурентні співвідношення першого та другого порядку, які представлені в (1.19).

$$x_{i+1} = \varphi(x_i); \quad x_{i+1} = \psi(x_i, x_{i-1}) \tag{1.19}$$

Перелічені умови необхідні, але недостатні. Тобто при розгляді та проектуванні генераторів існує можливість відразу відкинути неперспективні розробки. В інших випадках необхідно провести емпіричні випробування для визначення інших статистичних характеристик генератора.

За допомогою загальних та спеціалізованих методів результуюча квазірівномірна вихідна послідовність перетворюється у послідовність псевдовипадкових наборів із заданим законом. Прикладами загальних матодів є:

- вихідна послідовність піддається аналітичному перетворенню елементів;
- метод остач;
- східчаста функцієя замінює заданий закон розподілу.

Дані методи доцільно застосовувати для отримання псевдовипадкових послідовностей, які мають різний тип законів розподілу.

Спеціальні алгоритми ж в свою чергу надають можливість перетворення вихідної послідовності у послідовність з конкретним законом розподілу, але він придатний лише до одного розподілу. Тобто кожен алгоритм здійснює перетворення ли на один заданий тип розподілу.

Тому, розробка генератора, який здатен генерувати вихідні набори за будь-яким заданим розподілом є актуальною та новою.

1.4 Тестування псевдовипадкових послідовностей

За допомогою сукупності методів для визначення кореляції між заданою псевдовипадковою послідовністю наборів та випадковою послідовністю відбувається тестування генератора. В більшості випадків випадковість послідовності наборів встановлюється за допомогою наступних критеріїв:

- рівномірного розподіл наборів;
- великого періоду генератора;
- частота появи однакових частин наборів.

Для тестування псевдовипадкових послідовностей існують наступні різновидності тестів:

- графічний;
- статистичний.

1.4.1 Графічні тести

Відображення результатів дослідження псевдовипадкової послідовності наборів за певними статистичними експериментами у виглядя графіків носить назву статистичних тестів[14].

Приклади тестів:

- розподіл на площині (визначення взаємозалежності між елементами псевдовипадкової послідовності);

- гістограма розподілу наборів зображена на Рисунку 1.8 (оцінка рівномірності розподілу наборів у послідовності, визначення частоти повторів кожного з наборів);

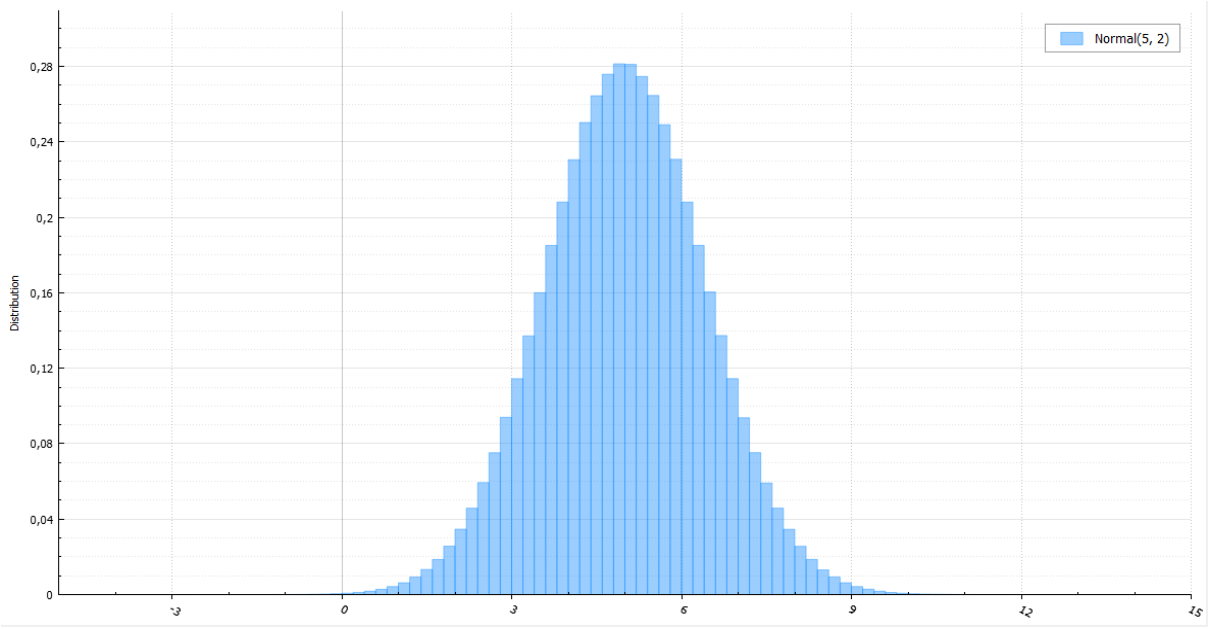


Рисунок 1.8 - Приклад гістограми розподілу(нормальний розподіл)

- перевірка на монотонність (визначення рівномірності розподілу за допомогою аналізу впорядкування підпослідовності, наприклад, спадна чи зростаюча);

- перевірка серій (визначення рівномірності розподілу окремих розрядів наборів в послідовності, а також рівномірність розподілу серій з повного числа біт);

- автокореляційна функція зображена на Рисунку 1.9(оцінка кореляції між зсунутими співпадаючою множиною наборів послідовності і окремих підпослідовностей наборів);

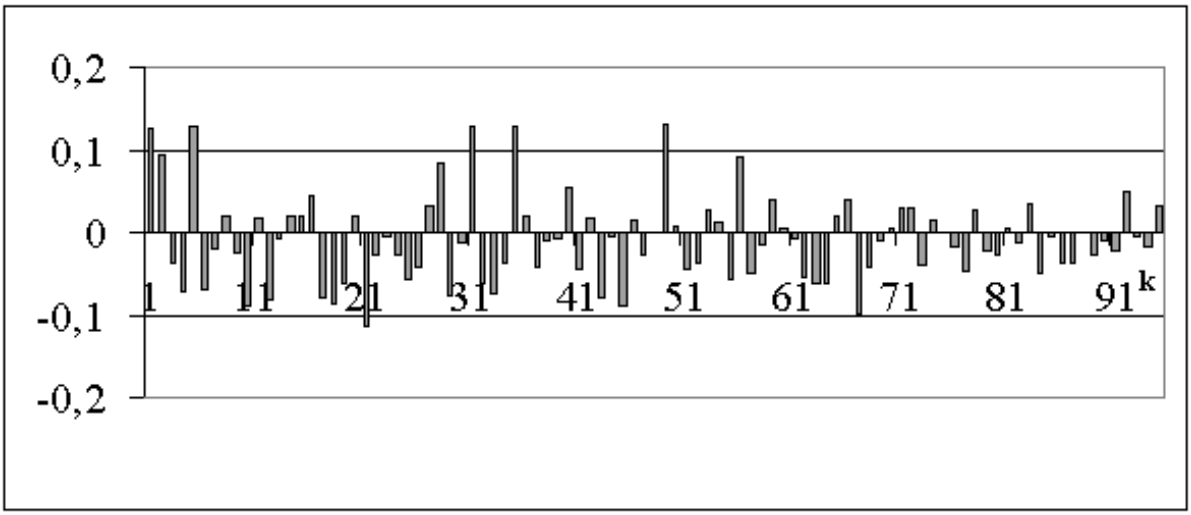


Рисунок 1.9 – Приклад частини автокореляційної функції ряду

- графічний спектральний тест (оцінка рівномірність розподілу біт псевдовипадкової послідовності за допомогою аналізу значень викидів перетворення Фур'є);

- профіль лінійної складності (оцінка залежності лінійної складності псевдовипадкової послідовності від її довжини).

Результати графічних тестів можуть бути неоднозначними, оскільки висновки на їх основі формує людина.

1.4.2 Статистичні тести

Відмінність статистичних тестів від від графічних полягає в тому, що їх результатом є чисельні характеристики псевдовипадкових послідовностей. Саме тому вони мають змогу однозначно отримати відповідь на питання проходження тесту.

Найбільш відомі тести:

- статистичні тести Д. Кнута;
- DIEHARD;
- CRYPT-X;
- NIST STS;

На основі ймовірнісних властивостей випадкових послідовностей існує можливість створення певних критеріїв. Такі критерії повинні відповідати лише випадковим послідовностям.

Після збору статистичних даних по виділеним критеріям для випадковою послідовності та згенерованої формувачем, необхідно здійснити наступну перевірку: чи перевищує різниця цих значень деяке критичне число. Критичне число встановляюється заздалегідь користувачем. Якщо результат перевірки незадовільний, то послідовність вважається невипадковою.

Позначимо ймовірність події того, що послідовність хибно визнається невипадковою, як α. А ймовірність події того, що послідовність хибно визнається випадковою, - β. Існує залежність між значенням довжини набору n, α і β. Або експериментально задається таке α та n, щоб β мало мінімальне значення.

Позничимо ймовірність події того, що ідеальний генератор сфорував послідовність наборів менш випадкову, ніж об’єкт досліджень, як P-value. Тоді, якщо P-value більше α, то досліджувана послідовність вважається випадковою і навпаки в іншому випадку.

Кроки статистичного тестування зображено у Таблиці 1.1.

№ кроку	Процес	Коментарі
1	Постановка гіпотези	Припускаємо, що послідовність є випадковою

2	Обчислення статистики досліджуваної послідовності	Тестування на бітовому рівні
3	Обчислення P-value	$P-value \in [0;1]$
4	Порівняння P-value з α	Задаємо P-value у межах [0,001;0,001]; якщо P-value > α - тест пройдений

Таблиця 1.1 - Статистичне тестування

Тести Кнута засновані на статистичному критерії $\chi 2$. Переваги цих тестів:

- невелика кількість;
- існування готових алгоритмів виконання.

Недоліком є невизначеність трактування результатів.

Список тестів[15][16]:

- перевірка перестановок;
- перевірка на монотонність;
- перевірка незчеплених серій;
- перевірка інтервалів;
- перевірка комбінацій;
- тест збирача купонів;
- перевірка кореляції.

Пакет статистичних тестів NIST[18][19] - програмний пакет розроблений Лабораторією інформаційних технологій, що є головною організацією Національного інституту стандартів і технологій. До його складу входять 15 статистичних тестів. Мета цих тестів - визначення характеру випадковості двійкових послідовностей довільної довжини, які згенеровані апаратним або програмним формувачем випадкових або

псевдовипадкових наборів. Тести засновані на особливостях, які властиві лише випадковим послідовностям.

Список тестів:

- частотний побітовий;
- частотний блоковий;
- визначення послідовності однакових бітів;
- визначення найдовшої послідовності одиниць в блоці;
- ранг двійкової матриці;
- спектральний тест;
- збіг шаблонів, які не перекриваються;
- збіг шаблонів, які перекриваються;
- універсальний статистичний тест Маурера;
- лінійна складність;
- періодичність;
- з використанням приблизної ентропії;
- тест кумулятивних сум;
- тест на довільні відхилення.

Тести Diehard[17], що розроблені Джорджем Марсалу за декілька років, вперше були опубліковані на диску, присвяченому випадковим наборам. Це набір статистичних тестів для визначення якості наборів випадкових чисел. Сукупність тестів розглядається як однна з найбільш відомих та вимогливих.

Список тестів:

- ранги матриць;
- мавпячі тести;
- дні народження;
- пересічні перестановки;
- підрахунок одиниць;
- тест на парковку;

- мінімальна відстань;
- випадкові сфери;
- стиснення;
- тест пересічних сум.

1.4 Відмовостійкі багатопроцесорні системи. GL-модель

Відмовостійкі багатопроцесорні системи (ВБС) знаходять своє застосування у системах управління складними об’єктами (авіакосмічні системи, системи управління АЕС, системи управління великим виробництвом та ін.). В зв’язку із специфікою області до ВБС встановлюють високі вимоги по надійності[20].

K-out-of-n - це система, що складається з *n* елементів, яка може виконувати свої функції, тобто, зберігає працездатність, не менше ніж k елементів системи працездатні. Для таких систем обчислення ймовірності безвідмовної роботи (ІБР) відбується за допомогою простих формул і співвідношень, що не включають в себе складних обчислень. Але для ВБС, які управляють складними об’єктами, тобто кількість процесорів вимірюється щонайменше десятками або сотнями, за допомогою цих формул результат дає оцінку показника надійності з суттєвими похибками. В деяких випадках їх не можна застосувати взагалі.

Для складних ВБС, при їх проектуванні, показники надійності рекомендується розраховувати за допомогою методу статистичних експериментів. Ці експерименти проводяться над моделями систем, при цьому вона повинна відображати реакцію системи у реальному часі на виникнення відмов, тобто результати роботи деяких процесорів некоректні.

Тестові набори будуть подаватися на модель ВБС. Серед існуючих моделей реакції системи на появу відмов обрана модель, що розроблена на

кафедрі спеціалізованих комп'ютерних систем НТУУ «КПІ» графо-логічна модель (GL-модель).

GL-модель – це неорієнтований граф. В цьму графі ребра позначаються булевими функціями, аргументами яких є стан множини процесорів(індикаторних змінних). Тобто, кожна індикаторна змінна відображає стан відповідного процесорного елемента, приймаючи значення «0» при відмові та «1» при працездатності. Такі функції називаються реберними функціями. Сукупність значень індикаторних змінних - це вектор стану системи. Якщо на заданому векторі стану роботи система працездатна, то граф GL-моделі є зв'язним. Таким чином існує можливість встановити відповідність працездатність системи у певному стані до вектора, а вектор стану системи до графа GL-моделі.

ВБС, яка зберігає працездатність на всіх векторах стану системи, кількість відмов в яких не вище фіксованого значення величини k, - це базова ВБС.

ВБС, яка зберігає працездатність на всіх векторах стану системи, кількість відмов в яких не вище значення величини k та на деякій не повній множині значень більше ніж k, - це небазова ВБС.

Одне з застосувань статистичних експериментів - це визначення небазових ВБС.

1.5 Висновки до розділу 1

Значне застосування генераторів псевдовипадкових наборів набуло в багатьох галузях. Серед них особливе значення становлять в областях, які використовують електронну та електронно-обчислювальну техніку.

Більша частина арифметичних генераторів має перевагу в швидкодії, але й страждає від настпнух недоліків:

- не достатньо тривалий період генератора;
- взаємнозалежність значень наборів;

- оборотність

- відсутність рівномірного розподілу ймовірності появи значення на будь-якому біті набора;

- відсутність рівномірного розподілу власне вихідних наборів.

Кожен з розглянутих алгоритмів генерування має свої переваги та недоліки, але кожен з них здатен генерувати вихідну послідовність лише за одним законом розподілу. Тому розробка генератора, який здатен генерувати вихідні набори за будь-яким заданим розподілом є актуальною та новою.

Для складних ВБС, при їх проектуванні, показники надійності рекомендується розраховувати за допомогою методу статистичних експериментів. Ці експерименти проводяться над моделями систем, при цьому вона повинна відображати реакцію системи у реальному часі на виникнення відмов, тобто результати роботи деяких процесорів некоректні.

Тестові набори будуть подаватися на модель ВБС. Серед існуючих моделей реакції системи на появу відмов обрана модель, що розроблена на кафедрі спеціалізованих комп’ютерних систем НТУУ «КПІ» графо-логічна модель (GL-модель).

РОЗДІЛ 2. МЕТОД ФОРМУВАННЯ ВХІДНИХ ДАНИХ В МОДЕЛЯХ ВІДМОСТІЙКИХ БАГАТОПРОЦЕСОРНИХ СИСТЕМ

2.1 Опис структури формувача

Метод реалізується за допомогою керованого генератора псевдовипадкових двійкових наборів з заданим розподілом ймовірності появи набору заданої ваги.

Керований генератор псевдовипадкових двійкових наборів можна розділити на дві частини за своїми функціями:

- генерування наборів заданої ваги;
- забезпечення значення ймовірності появи набору заданої ваги.

Реалізацію функції генерування наборів заданої ваги узагальнено представляють собою елементи:

- регістр, у якому відбувається зсув певним чином;
- комбінаційна схема, яка забезпечує зсув за один такт;
- генератор псевдовипадкових чисел;
- дешифратор.

Зв’язок між частинами генератора відображена на Рисунку 2.1.

У регістрі використовується задана кількість бітів n.

Також можливе не повне використання виходів дешифратора. Це пояснюється тим, що дешифратор має 2^k виходів (де k – кількість входів дешифратора). А використовуються лише n виходів дешифратора (адресних бітів Lb(n) з округленням до більшого цілого значення), кожен з яких зв’язаний з одним розрядом вихідного набору.

Генератор псевдовипадкових чисел являє собою окрему схему, яка не буде описана, оскільки це окрема область для дослідження. Детальнішу інформацію, а також реалізацію цього генератора, можна знайти у

документації з мови C++ відносно команди rand(). Головна його особливість полягає в тому, що він генерує число, яке знаходиться в діапазоні від 1 до n, де n – необхідна розрядність набору. При цьому всі ймовірності появи числа у заданому діапазоні рівні (або квазірівномірний розподіл цих чисел).

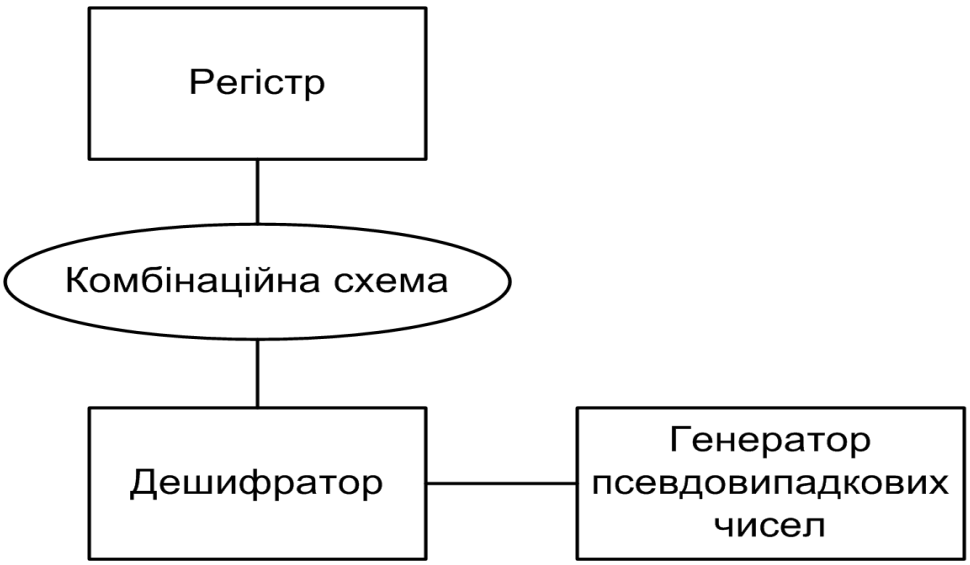


Рисунок 2.1 - Структура функціональної частини генератора

Ну цю частину генератора виданий патент на корисну модель[21], формула якого наведена нижче та кресленням Рисунок 2.2.

Генератор псевдовипадкових двійкових наборів з рівною вагою, що містить багатоканальний вузол генерації випадкових бітів, кожен канал в якому включає в себе послідовно з'єднані генератор шуму, підсилювач-обмежувач і лічильний тригер, виходи усіх вузлів об'єднані схемою "ВИКЛЮЧНЕ АБО", і канал спряження з ПЕОМ, що включає в себе регістр зсуву, виходи якого увімкнуті до входів вихідного регістра, з'єднаного виходами з шиною даних ПЕОМ, тактовий генератор, вихід якого з'єднаний з синхровходом регістра зсуву та входом лічильника імпульсів, вихід якого під'єднаний до синхровходу вихідного регістра та входу тригера "прапора", а його вихід з'єднаний з виходом запиту переривання та через буферний елемент І з шиною даних ПЕОМ, і дешифратор адреси, включений входами до шини адрес ПЕОМ, а першим

виходом до входу дозволу вихідного регістра та входу скидання тригера "прапора", а другим виходом до буферного елемента I, який відрізняється тим, що присутні n-1 одноадресних мультиплексорів 1, n D-тригерів 2, дешифратор 3, [Lbn]-адресний мультиплексор 4, причому на адресні входи n-1 одноадресних мультиплексорів 1 підключено 1…(n-1) розряди виходу дешифратора 3, відповідно, а на інформаційні входи подається значення 1…(n-1)-го D-тригера відповідно та n-ний D-тригер, на вхід 2…n D-тригерів 2 підключені виходи n-1 одноадресних мультиплексорів 1, а на вхід 1-го D-тригера 2 підключений вихід [Lbn]-адресного мультиплексора 4, на n інформаційних входів [Lbn]-адресного мультиплексора 4 підключено n D-тригерів 2, а на адресні входи подається канал випадкових бітів, також він подається й на вхід дешифратора 3.

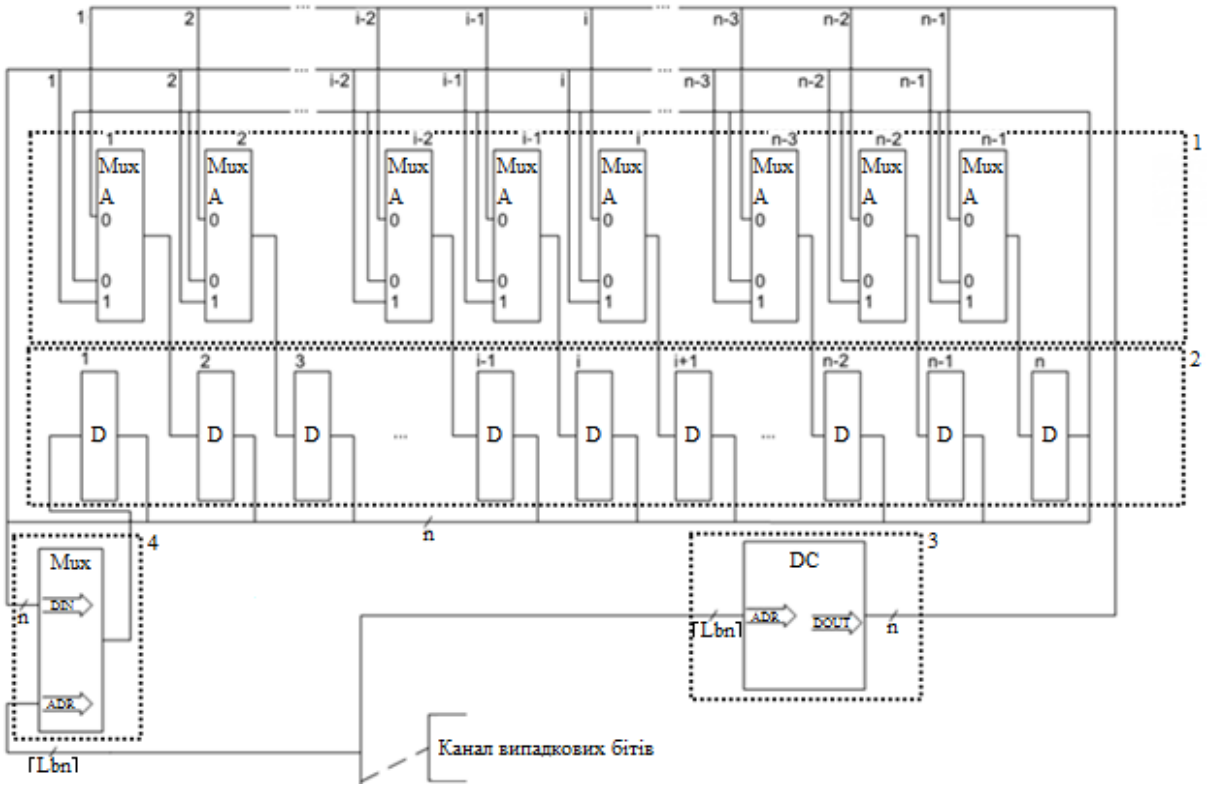


Рисунок 2.2 – Креслення корисної моделі

Комбінаційна схема складається з 2 частин:

- схема управління зсувом;
- схема забезпечення зворотного зв’язку.

До обох частин входять лише мультиплексори.

Мультиплексор - комутатор цифрових сигналів. Мультиплексор є комбінаційним пристроєм, який складається з наступних частин:

- m інформаційних входів;
- n управляючих входів;
- один вихід.

На функціональних схемах мультиплексор представляється як m логічних елементів «І», виходи яких подаються як входи логічного елемента «АБО». На перший вхід всіх логічних елементів «І» подається відповідний інформаційний сигнал, а інші входи цих елементів пов’язані з виходами дешифратора відповідно. Дешифратор має n входів.

На схемі, ща представлена на Рисунку 2.3[22] слід, показано мультиплексор на чотири входи. Він містить дешифратор на відповідно чотири виходи виходи, логічні елементі «І» на два або на три входи та логічний елемент «АБО» на чотири входи. При збільшенні кількості інформаційних входів D до числа k кількість логічних елементів «І» та входів логічного елемента «АБО» має дорівнювати k, а кількість виходів дешифратора має бути не менша ніж k.

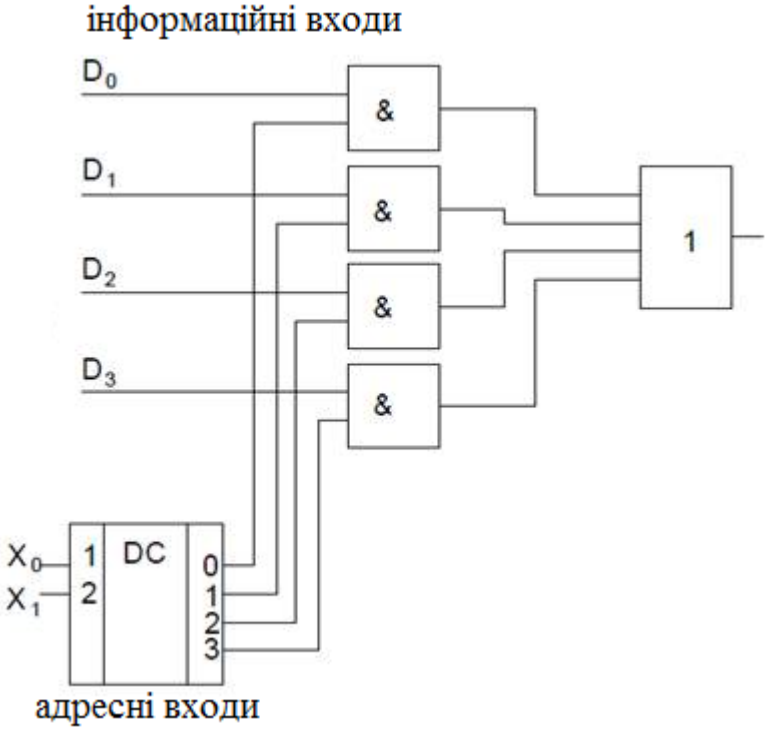


Рисунок 2.3 – Схема мультиплексора

Логічний елемент «І» повинен мати лише два входи, але в більшості ситуацій з’являється необхідність присутності стробуючого вихідного сигналу мультиплексора. Цей сигнал представляє собою імпульс незалежного джерела. Саме тому в структурі мультиплексора використовуються логічні елементи «І» з трьома входами. В останньому випадку один з входів всіх елементів кон'юнкції об'єднується в шину. Пой цій шині також передається сигнал стробу, тобто дозвіл роботи мультиплексора. Додатковий керуючий вхід дозволяє розширити функціональні можливості мультиплексора.

На функціональних електричних та принципових схемах мультиплексор позначається як на Рисунку 2.4.

Рисунок 2.4 – Мультиплексор на схемах

Схема управління зсувом представляє собою блок мультиплексорів, які під’єднані до входу кожного з елементів пам’яті регістра, окрім першого. Управління елементами здійснюється сигналом від відповідного виходу дешифратора. Сигнал, що подається на вихід, обирається серед попереднього елемента пам’яті чи останнього елемента регістра. Таким чином визначаються нові значення усіх бітів регістра, окрім першого.

Схема забезпечення зворотного зв’язку й визначає нове значення першого біту регістра. Управляючими сигналами для мультиплексора є сигнали, які у сукупності містять номер біту, який уявно ділить регістр на дві частини, у яких відбуваються кільцеві зсуви. Сигнал, що подається на вихід обирається серед усіх значень бітів регістра.

Одна з частин керованого генератора - регістр, у якому відбуваються два кільцеві зсуви. Яким чином повинні відбуватися зсуви демонструє Рисунок 2.5.

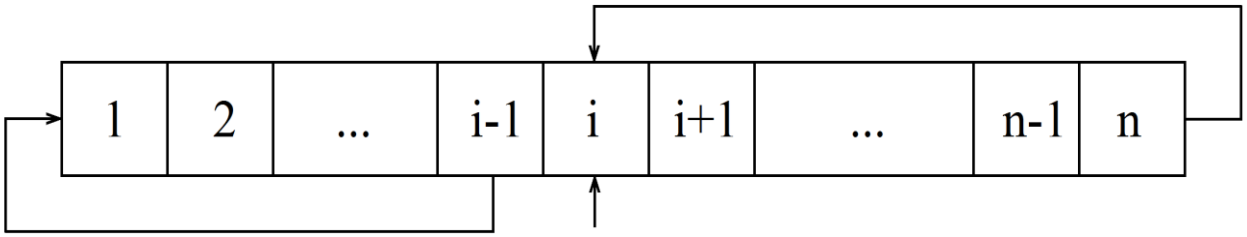


Рисунок 2.5 - Кільцеві зсуви в регістрі

Для забезпечення значення ймовірності появи набору заданої ваги використовуються наступні частини генератора:

- схема зміни ваги;
- елемент пам’яті, сформований відповідно розподілу ймовірності появи набору заданої ваги;
- лічильник.

Узагальнену схему генератора представлена на Рисунку 2.6.

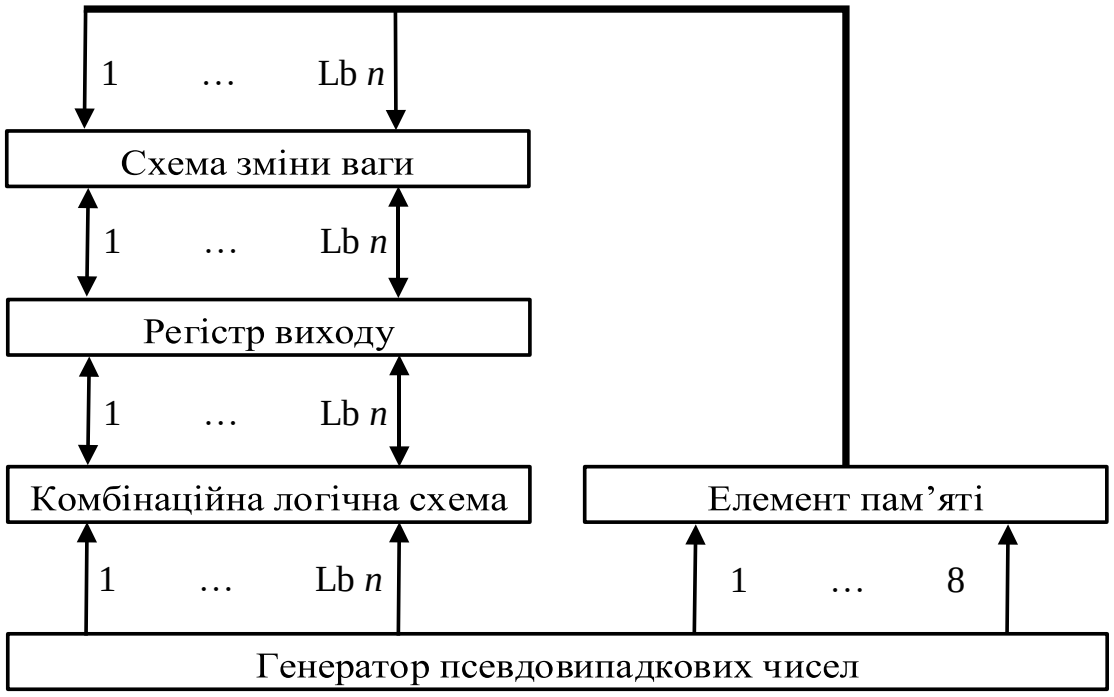


Рисунок 2.6 - Загальна схема генератору

Кожні T тактів відбувається зміна ваги набору вихідного регістра. Саме для цього використовується лічильник.

Після сигналу лічильника, що сповіщає про видачу T вихідних наборів, ініціюється зміна ваги набору для наступних T тактів роботи генератора. Це відбувається за допомогою схеми зміни ваги, що поміщає у вихідний регістр псевдовипадковий набір з новою вагою.

Значення нової ваги визначається за допомогою декількох розрядів ГПВЧ (залежить від кількості адресних входів елемента пам’яті) та елемента пам’яті. Кількість ствобців у пам’яті повинно бути не менше Lb(n), оскільки вага набору не перевищує n, то її двійкове подання дорівнює двійковому логарифму від n. Кількість рядків обирається відповідно до прогнозованої пихибки ймовірності появи набору заданої ваги. Тобто, якщо кількість рядків – 256 (8 адресних входів), то похибка буде дорівнювати відношенню одиниці до 256 (0,4%).

Запам’ятовуючий пристрій повинен бути заповнений до початку роботи генератора.

Алгоритм заповнення елемента пам’яті:

1. Введення значень кількості різних ваг kW. Лічильник k = 0. Кількість рядків елемента пам’яті kR = 256. Номер поточного рядка numb = 0. Сума ймовірностей sum = 0.
2. Перевірка (k < kW). Твердження вірне – п.3, інакше – п.11.
3. Введення значення ваги w, а також її ймовірності появи p.
4. Перевірка (numb/kR <= (sum + p)). Твердження вірне – п.5, інакше – п.8.
5. Заповнення рядка numb значенням w.
6. numb = numb + 1.
7. Перехід на п.4.
8. sum = sum + p.
9. k = k + 1.
10. Перехід на п.2.
11. Кінець.

2.2 Опис програмної реалізації

Програмна реалізація керованого генератора псевдовипадкових двійкових наборів з розподілом ймовірності появи набору заданої ваги дозволяє власне побачити згенеровані набори, а також отримати результати тестування генератора.

При запуску програми ми побачимо меню користувача, яке зображено на Рисунку 2.7.

Меню включає в себе:

- задання параметрів двійкового набору:
  - розрядність набору;
  - вага набору;
- запуск генератора;

- задання декількох ваг:
    - вага набору;
    - ймовірність появи набору заданої ваги;
- задання кількості тактів генератора між змінами ваги(T);
- тестування генератора;
- вихід з програми.

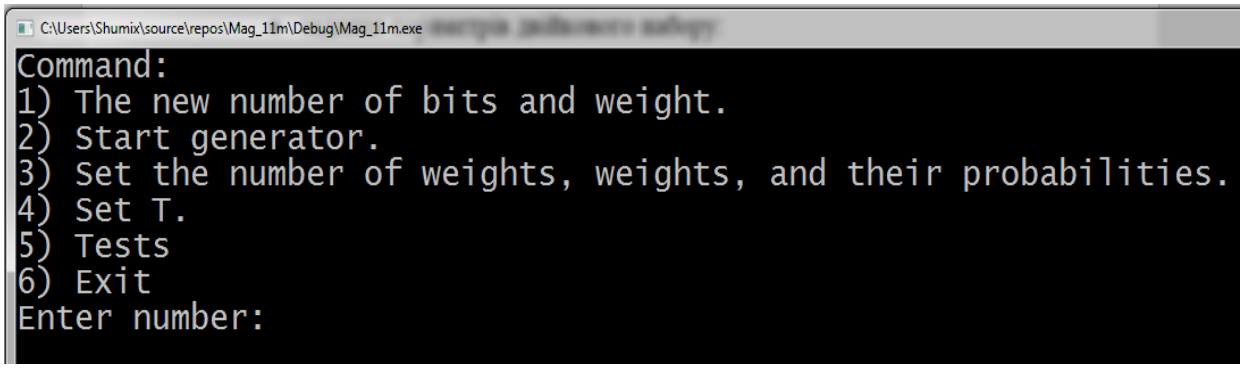


Рисунок 2.7 - Меню користувача

На Рисунку 2.8 зображений процес зміни ваги.

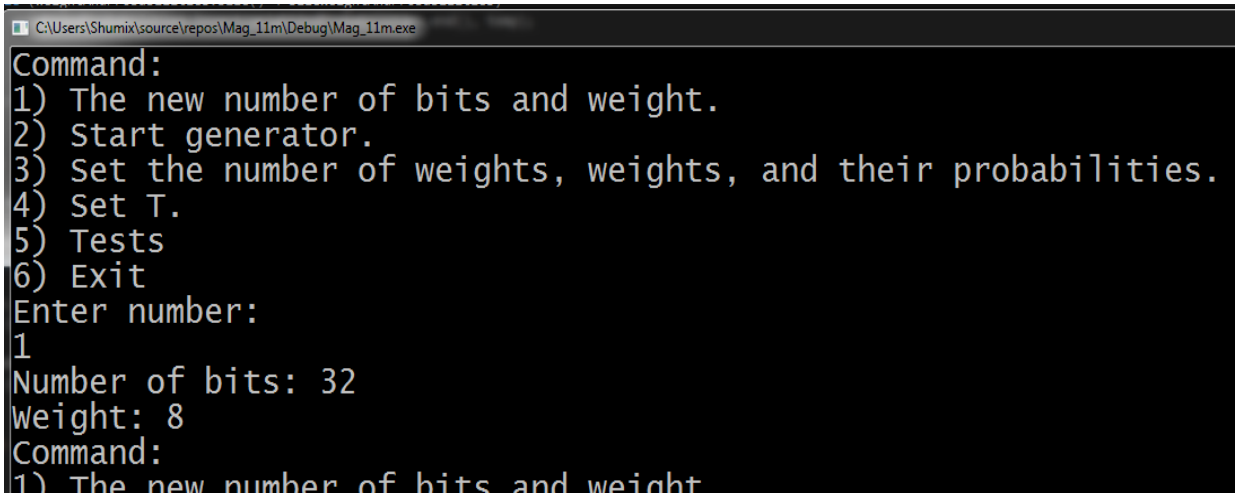


Рисунок 2.8 - Зміна ваги

На Рисунку 2.9 зображений процес зміни кількості тактів генератора між змінами ваги(T).

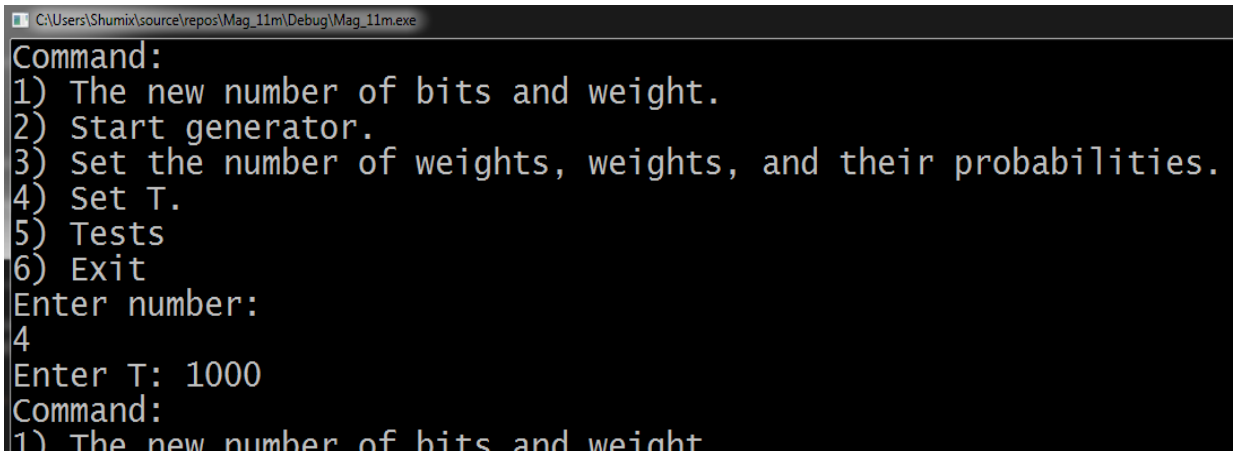


Рисунок 2.9 - Зміна T

Перед початком роботи задаються початкові значення, які можна надалі змінити за допомогою меню. Реалізація:

```
Generator::Generator()
{
    sizeSet = 16;
    wghtSet = 8;
    reg = "0000111111110000";
    kWeights = 1;
    maxT = 10000;
    WP temp;
    temp.weight = wghtSet;
    temp.probability = 1;
    weightAndProbabilities.insert(weightAndProbabilities.end(), temp);
}
```

Тобто, для початку роботи необхідно:

- задати вагу вихідних наборів(на початку роботи генерування відбувається лише для однієї ваги набору);
- задати розрядність вихідних наборів;
- вказати початкове значення вихідного регістра;
- заповнити всі рядки елемента памя’ті значенням поточної ваги.

З головного меню можна потрапити в підрозділ тестування, який зображений на Рисунку 2.10.

Він включає в себе:

- розрахунок ймовірності появи значення в заданому розряді;
- розрахунок кількості згенерованих наборів щоб отримати всі еталонні набори;
- повернення до головного меню.

Під еталонними наборами маються на увазі всі можливі перестановки розрядів двійкового набору, вага та розрядність якого введені, або були задані при ініціалізації програми.

У програмі роль ГПСЧ (однієї з головних складових ГПВН) відіграє команда rand().

ЇЇ використання показане наступною командою:

int sh = 2 + rand() % (::size - 2);

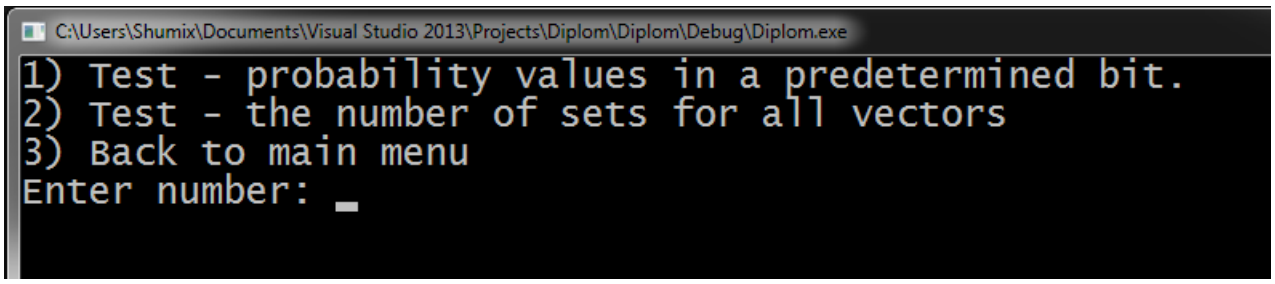


Рисунок 2.10 - Підрозділ тестування

Змінні:

- sh - номер біту, який уявно ділить двійковий набір на дві частини, у яких відбуваються кільцеві зсуви;
- size – необхідна розрядність набору.

Для отримання еталонних наборів розроблно алгоритм, який представляє собою побудову дерева.

Реалізація цього алгоритму на мові C++:

```
void rec(string s, int k, int k1)
{
	if (k == ::size)
		::arr.insert(pair <string, int>(s, 0));
	else
	{
		if ((::size - k) > (::ves - k1)) rec(s + "0", k + 1, k1);
		if (k1 < ::ves) rec(s + "1", k + 1, k1 + 1);
	}
}
```

Змінні:

- s – рядок, що являє собою двійковий набір;
- arr – контейнер для збереження всіх рядків;
- k1 – поточна кількість «одиничних» символів у рядку;
- k – поточна довжина рядка;
- size – необхідна розрядність набору;
- ves – необхідна вага набору.

Цей рекурсивний алгоритм, який формує дерево, що зображено на Рисунку 2.11.

Завдяки умовам, які застосовуються в алгоритмі, ми отримуємо не всі можливі перестановки набору заданої розрядності, а лише з потрібною нам вагою.

Умови наступного виклику функції:

- поточна розрядність набору менше заданої;
- додаємо «1» лише у випадку, якщо поточна кількість «1» у рядку менше заданої ваги набору;
- додаємо «0» лише у випадку, якщо залишилось дописати символів у рядок до потрібної довжини більше ніж потрібно додати «1» в набір.

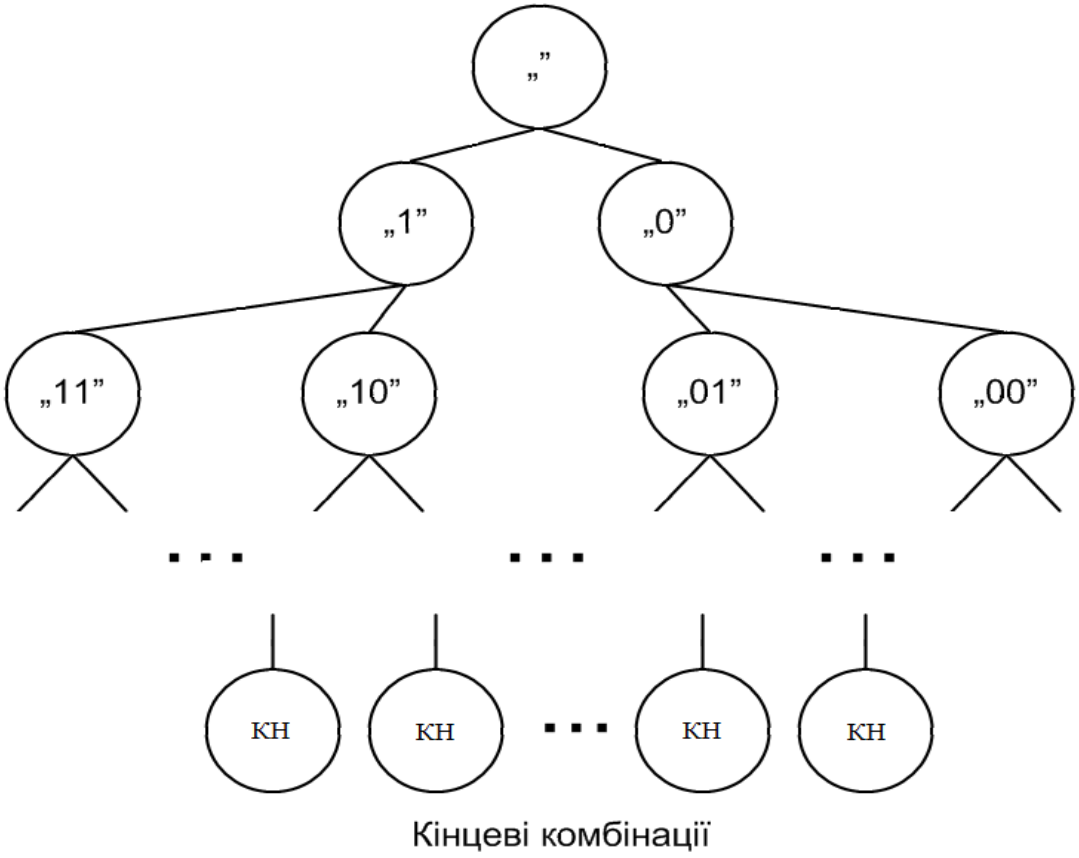


Рисунок 2.11 - Дерево значень рядків s

Щоб зберігати всі еталонні набори використовується контейнер MAP.

MAP — контейнер у вигляді набору даних, полями якого є ключ та відповідне йому значення (цю структуру також називають асоціативним масивом[23]). Змінні будь-якого типу можуть слугувати ключами за умови

їх унікальності. Також існує контейнер multimap, який за функціями схожий на map. Відмінність полягає в тому, шо серед ключів можуть бути повтори.

MAP – це асоціативний контейнер. Метод асоціативності має переваги в доступі за ключем. Послідовні контейнери ж в свою чергу ефективніші для доступу по порядковому індексу.

Основні дії для контейнерів:

- вставка елемента;
- видалення елемента;
- пошук елемента.

Усі перелічені дії асоціативний контейнер виконує за логарифмічний час - O(log n). Логаріфмічна складність виконання досягається за допомогою бінарного збалансованого дерева, в якій реалізована двостороння ітерація.

Абстрактний тип даних[24] - це тип даних, який надає для роботи з елементами певного набіру цього типу функцій, а також можливість створювати елементи цього типу за допомогою спеціальних функцій. Вся внутрішня структура такого типу захована від розробника програмного забезпечення - в цьому і полягає суть абстракції.

В програмуванні, абстрагування — спосіб та метод відокремлення деталей з метою отримання можливості зосередитись на найважливіших особливостях об'єкта.

MAP - це контейнер, в якому ітератори вказують на конкретні елементи. Змінна, яка являє собою ітератор, має два поля. Ці поля власне й представляють собою ключ і елемент, який відповідає ключу.

За допомогою ітератора реалізується послідовний доступ до будь-якого елемента контейнера. При цьому внутрішнє улаштування елемента не розкривається.

Також одна з функцій програми, яка виконується автоматично, - це вивід у зовнішній файл кількості повторів еталонних наборів, які були згенеровані у момент роботи керованого генератора псевдовипадкових двійкових наборів. Це зроблено для того, щоб визначити чи мають набори рівну ймовірність появи на виході приладу для однієї ваги та заданих розподіл для декількох ваг.

Для здійснення звичайного кільцевого зсуву використовується наступна функція:

```
string Shift(string s)
{
        if (!s.empty()) s = s[s.size() - 1] + s.substr(0, s.size() - 1);
        return s;
}
```

У функції використовується лише одна змінна-рядок, який представляє собою двійковий набір.

Алгоритм роботи:

- отримуємо останній розряд набору;
- отримуємо всі розряди набору, крім останнього;
- розташовуємо у новий рядок отримані частини в потрібному порядку;
- повертаємо результат функції.

Але це звичайний зсув, а потрібен зсув у двох частинах двійкового набору. Це реалізується, використовуючи вище описану функцію, наступним чином:

```
void Form()
```

```
{
        while (true)
        {
                if (_kbhit() && _getch() == 27) break;
                int sh = 2 + rand() % (::size - 2);
            ::reg = Shift(::reg.substr(0, sh - 1)) +
                                        Shift(::reg.substr(sh - 1,
                                ::reg.size()));
                cout << ::reg << endl;
            }
}
```

Змінні:

- sh - номер біту, який уявно ділить двійковий набір на дві частини, у яких відбуваються кільцеві зсуви;
- reg – рядок, який являє собою двійковий набір.

Числова константа 27 відповідає коду клавіші Esc на клавіатурі.

Алгоритм роботи:

- перевірка на завершення роботи генератора
- отримання номеру біту, який умовно розділить двійкових набір на 2 частини, в яких відбудеться зсув;
- зсув у лівій частині двійкового набору;
- зсув у правій частині двійкового набору;
- розташування у новий рядок нової лівої та правої частини двійкового набору;
- вивід результату функції.

Двійковий набір у всій програмі представлений як рядок – string.

Для мови програмування C++ string – це стандартний клас, який представляє собою рядок з символів, тобто, текст[25]. В цьому класі вирішується проблеми, які виникають при роботі з рядками. Ці рядки в C++ прийшли ще з мови C. Логіка управління пам'яттю покладена на клас string. Тому програмісту не потрібно самому виділяти пам’ять під рядок та точно вказувати кількість символів у рядку, також додано можливість використовувати NUL байт при роботі з рядком. Клас string – це клас, який реалізує операції, що є типовими для рядків:

- порівняння;
- конкатенація;
- пошук і заміна;
- функція отримання підрядків.

Існує можливість отримати рядок мови C з string та навпаки.

Підрядок — непорожня зв'язна частина рядка.

У програмі широко використовується отримання підрядка за допомогою наступної команди:

string substr (string stringForSeek, int startValue [, int lengthSubstr]).

Substr повертає частину рядка string, специфіковану параметрами start і length.

Якщо start позитивний, то повертається рядок починається зі стартової позиції в string, відлічуваної від нуля. Наприклад, в рядку 'abcdef' символ в позиції 0 це 'a', символ в позиції 2 це 'c', і так далі.

На Рисунку 2.12 зображений процес зміни кількості ваг наборів, а також задання ймовірності появи набору з заданою вагою.

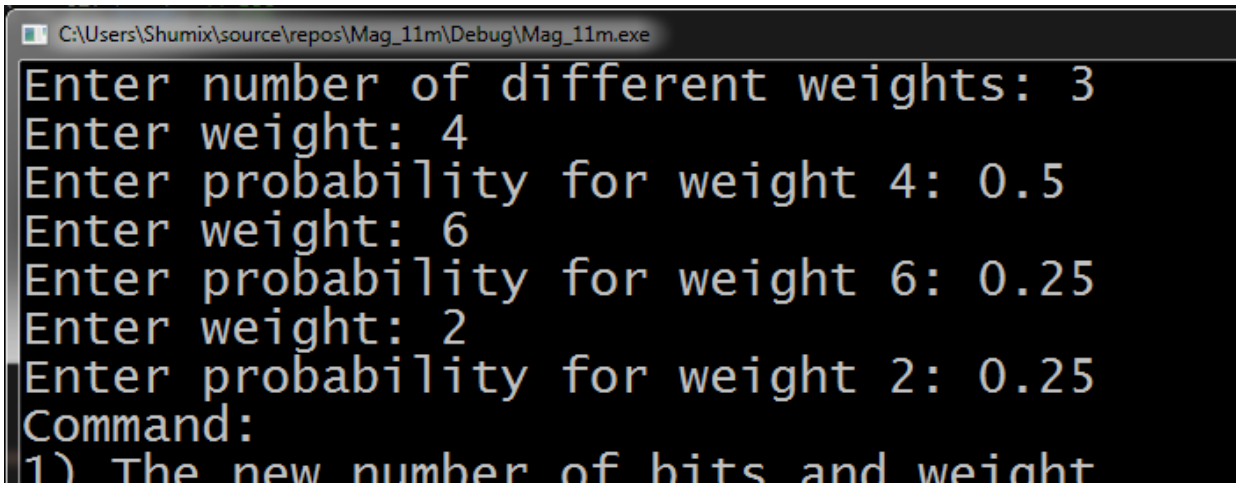


Рисунок 2.12 - Зміна кількості ваг

Частина програмної реалізації алгоритму заповнення елемента пам’яті:

```
cin >> kWeights;
weightAndProbabilities.clear();
for (int i = 0; i < kWeights; i++)
{
    cin >> temp.weight;
    cin >> temp.probability;
    float setProbability = 0;
    while ((setProbability < temp.probability) &&
(weightAndProbabilities.size() < sizeWeightAndProbabilities))

{weightAndProbabilities.insert(weightAndProbabilities.end(), temp);
        setProbability += 1.0 /
float(sizeWeightAndProbabilities);}
    if (i == (kWeights - 1))
    while (weightAndProbabilities.size() <
sizeWeightAndProbabilities)
weightAndProbabilities.insert(weightAndProbabilities.end(), temp);}}
```

Змінні:

- kWeights – загальна кількість ваг наборів різних за значенням, які будуть чергуватися між собою кожні T тактів;
- temp – екземпляр структури WP;
- setProbability – поточна ймовірність появи набору з вагою temp.weight;
- sizeWeightAndProbabilities – кількість рядків елемента пам’яті;
- weightAndProbabilities – масив значень елемента пам’яті.

В кінці роботи алгоритму відбувається перевірка на те, чи повністю елемент пам’яті заповнений, оскільки через похибки є ймовірність некоректного заповнення останнього рядка.

Структура WP описана наступним чином:

```
struct WP
{
    int weight;
    double probability;
};
```

Змінні:

- weight – вага двійкового набору;
- probability – ймовірність появи набору з вагою weight.

Реалізацію тесту на ймовірність появи «1» на будь-якому розряді вихідного ругістру наведено нижче.

```
while (i < numbSetsTest)
    {
        reg = GetRegNewWeight();
        for (int T = 0; T < maxT; T++)
        {
```

```
                i++;
                if (i >= numbSetsTest) break;

                GenOneSet();
                for (int j = 0; j < weightAndRealProbabilities.size(); j++)
                    if (weightAndRealProbabilities[j].weight == wghtSet)
                    {
                        weightAndRealProbabilities[j].probability +=
1.0 / numbSetsTest;
                        break;
                    }
                if (reg[bitTest] == '1') k1++;
            }
        }
            cout << "k1 = " << k1 << " p1 = " << fixed << setprecision(6) <<
float(k1) / float(numbSetsTest) << endl;
```

Змінні:

- i, T – лічильник;
- numbSetsTest – максимальна кількість наборів, який сформує генератор перед видачею результату;
- maxT – кількість тактів генератора, після якого буде зменена вага набору вихідного регістра;
- reg – вихідний регістр;
- bitTest – номер розряду, на якому обраховуємо статистику появи «1»;
- weightAndRealProbabilities – масив значень ймовірності появи набору для всіх різних ваг;
- wghtSet – поточна вага нобору.

Функції:

GetRegNewWeight() – функція для визначення нової ваги вихідного набору кожні T тактів та для поміщення у вихідний регістр новий псевдовипадковий набір з заданою вагою;

GenOneSet() – функція генерування одного псевдовипадкового вихідного набору.

Генерування нової ваги:

```
string Generator::GetRegNewWeight(){
    WP temp;
    if (kWeights > 1)
        temp = weightAndProbabilities[rand() %
weightAndProbabilities.size()];
    else temp.weight = wghtSet;
    temp = wghtSet;
```

Змінні:

- kWeights – кількість різних можливих ваг вихідного набору;
- wghtSet – поточна вага набору;
- temp – проміжна змінна;
- weightAndProbabilities – масив значень ваг.

2.3 Висновки до розділу 2

Метод реалізується за допомогою керованого генератора псевдовипадкових двійкових наборів з заданим розподілом ймовірності появи набору заданої ваги.

Керований генератор псевдовипадкових двійкових наборів можна розділити на дві частини за своїми функціями:

- генерування наборів заданої ваги;
- забезпечення значення ймовірності появи набору заданої ваги.

На частину генерування наборів заданої ваги видан патент на корисну модель[21].

Створено рекурсивний алгоритм, завдяки умовам якого ми отримуємо не всі можливі перестановки набору заданої розрядності, а лише з потрібною нам вагою.

Створено алгоритм заповнення елемента пам’яті для отримання розподілу ймовірності появи нобору заданої ваги, при якому похибка кожної з ймовірностей не перевищує відношення одиниці до кількості рядків елемента пам’яті.

РОЗДІЛ 3
АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

3.1 Аналіз отриманих даних

Головна задача програмного продукту – власне отримання згенерованих наборів. Приклад таких наборів розрядністю двійкового набору 8 та ваги 3 показаний на Рисунку 3.1.

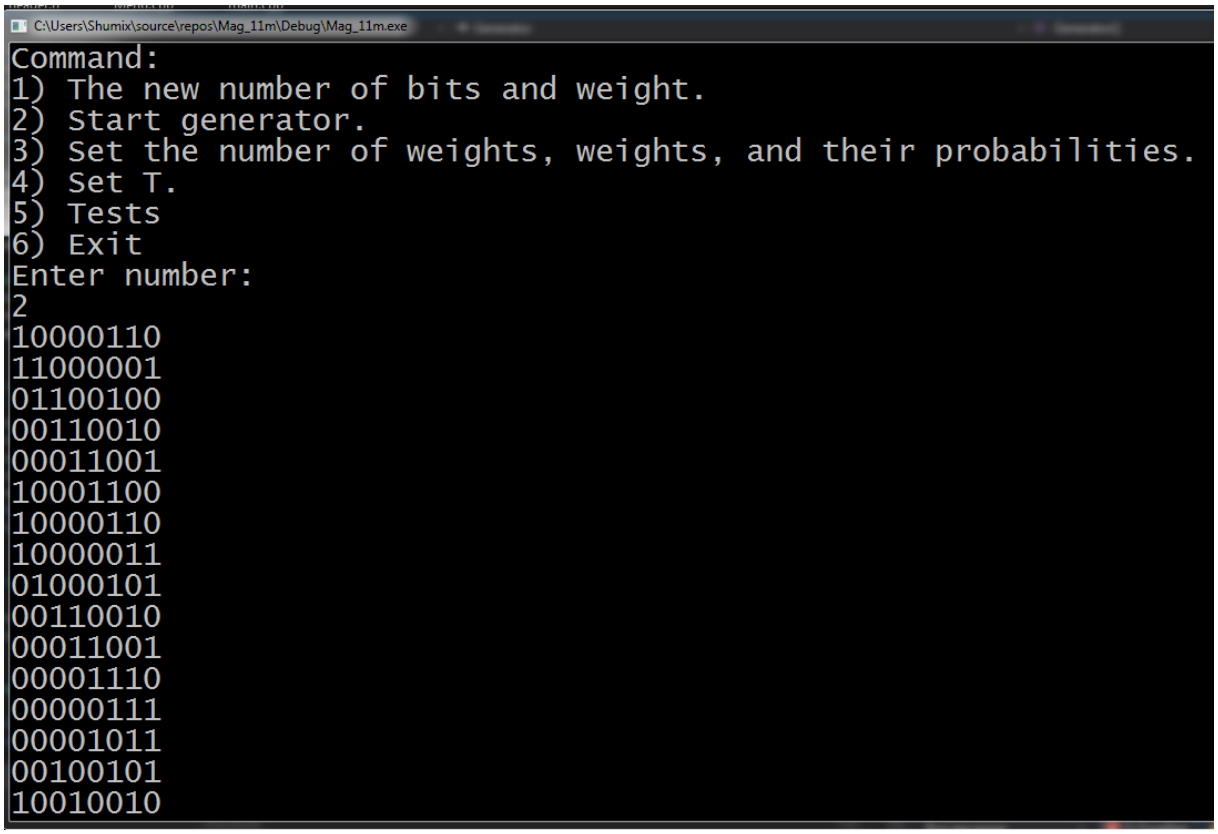


Рисунок 3.1 - Згенеровані набори

На Рисунку 3.1, усі комбінації задовольняють вимогам розрядності та ваги наборів.

Щоб перевірити правильність роботи керованого генератору псевдовипадкових двійкових наборів, перевіримо два параметри:

- ймовірність появи значення в заданому розряді двійкового набору;
- загальна кількість двійкових наборів заданих розрядності та ваги.

Також перевірки потребує власне заданий користувачем розподіл ймовірності появи набору заданої ваги та рівномірний розподіл наборів в межах однієї ваги.

Загальна кількість двійкових наборів заданих розрядності та ваги обраховується як сполука з n по k, де n – розрядність набору, k – вага набору.

Комбінація або сполука - це спосіб вибору, при якому з групи речей обирається декілька, при чому порядок не має значення.

Сполука k елементів з множини S - це підмножина, яка утворена k різними елементами з множини S. Кількість k-сполук множини, яка містить n елементів, являє собою біноміальний коефіцієнт, який представлено в (3.1). Цей коефіцієнт записаний за допомогою факторіалів показано в (3.2).

$$\binom{n}{k} = C_n^k = \frac{n(n-1)\dots(n-k+1)}{k(k-1)\dots 1} \tag{3.1}$$

$$\frac{n!}{k!(n-k)!} \tag{3.2}$$

Тобто при тих самих розрядності та вазі набору справедливе наступне твердження: кількість наборів дорівнює (3.3).

$$C_8^3 = \frac{8!}{(8-3)!3!} = \frac{8!}{5!3!} = 56 \tag{3.3}$$

Запустимо тестування генератору в програмі та отримаємо результат, який відображено на Рисунку 3.2.

На Рисунку 3.2, кількість еталонних наборів збігається – 56.

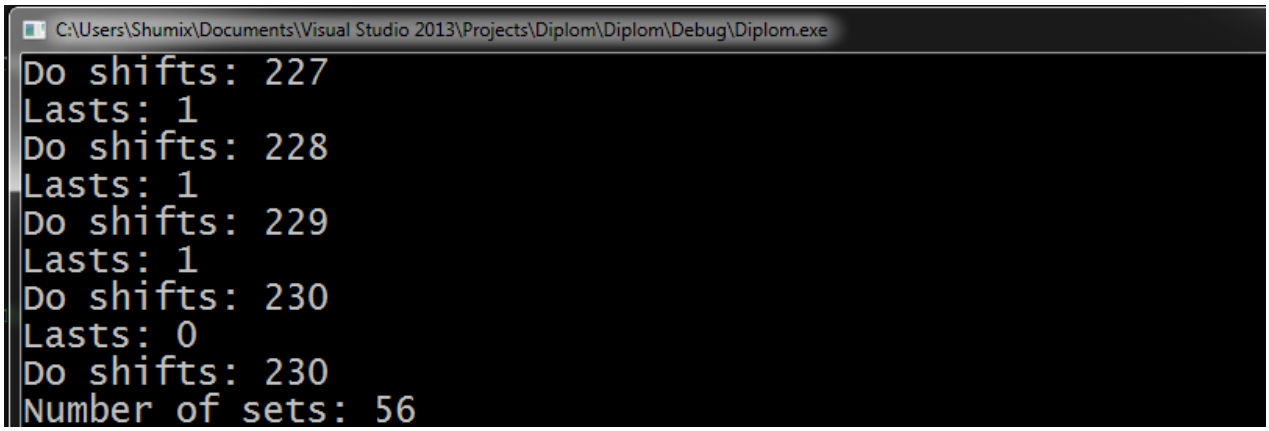


Рисунок 3.2 - Результат тестування генератору на кількість згенерованих наборів, щоб отримати усі еталонні набори.

Ймовірність появи значення в заданому розряді двійкового набору розраховується для будь-якого розряду наступним чином (3.4), де p – ймовірність, k – вага набору, n – розрядність набору.

$$p = \frac{k}{n} \tag{3.4}$$

Для отримання більш достовірних результатів обрано розрядність двійкового набору 16, а вагу 8. Тобто, ймовірність буде дорівнювати 0,5.

Після запуску програми на даний вид тестування отримаємо результат, наведений на Рисунку 3.3

На Рисунку 3.3, бачимо, що ймовірність точна до сотих відсотка. Також й кількість еталонних наборів вірна (3.5).

$$C_{16}^{8} = \frac{16!}{(16-8)!\,8!} = \frac{16!}{8!\,8!} = 12870 \tag{3.5}$$

.

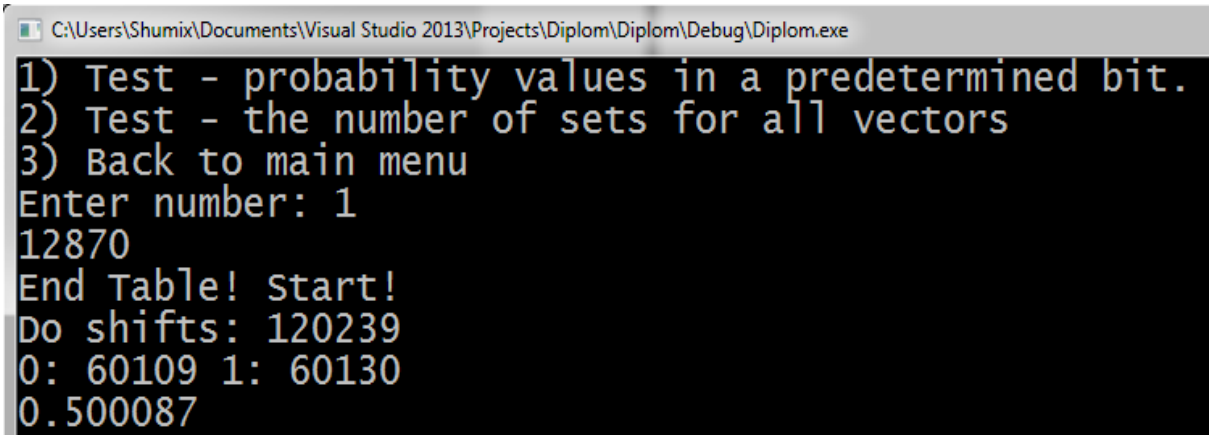


Рисунок 3.3 - Результат тестування генератору на ймовірність появи значення в заданому розряді двійкового набору

Перевіримо ще декілька значень розрядності та ваги двійкових наборів.

Отримані результати відносно кількості еталонних наборів занесені до Таблиці 3.1.

Розрядність n	Вага k	Кількість еталонних наборів	Кількість наборів у програмі
8	3	56	56
8	6	28	28
16	8	12 870	12 870
16	11	4 368	4 368
32	14	471 435 600	471 435 600
32	7	3 365 856	3 365 856
10	2	45	45
12	5	792	792
14	4	1 001	1 001
20	11	167 960	167 960
22	13	497 420	497 420

Таблиця 3.1 - Результати тестування генератору на кількість еталонних наборів

Розподіл кількості еталонних наборів в залежності від довжини набору n та його ваги наведено на Рисунку 3.4 та Рисунку 3.5.

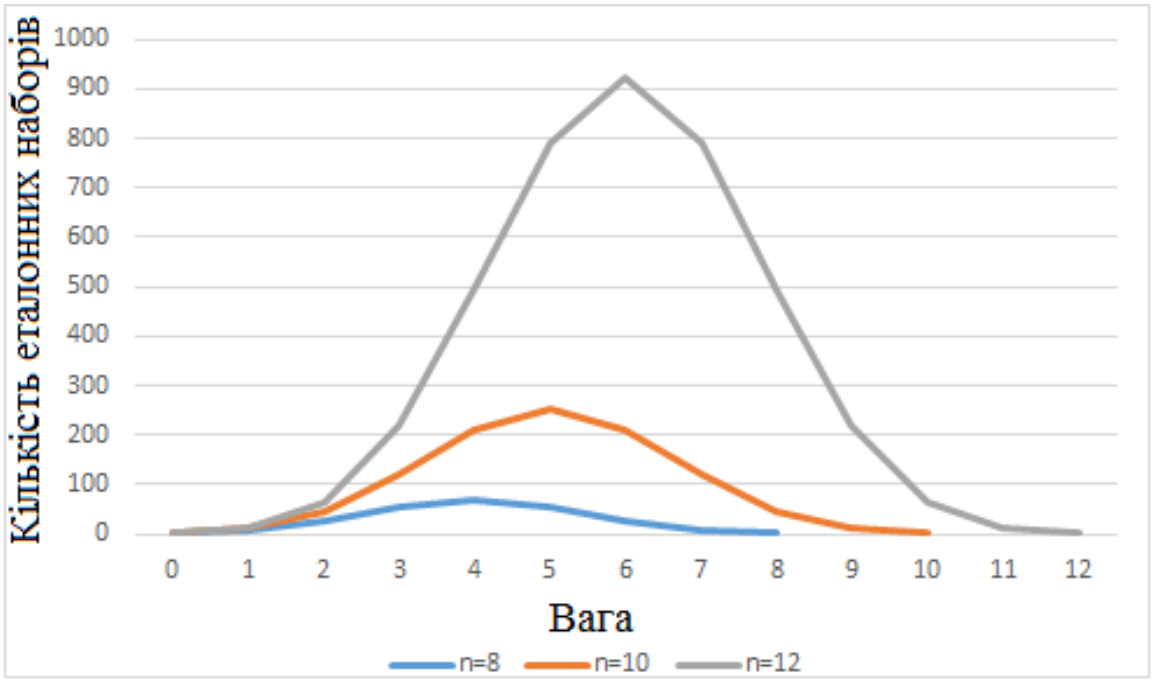


Рисунок 3.4 – Розподіл кількості еталонних наборів для 8,10,12-розрядного набору

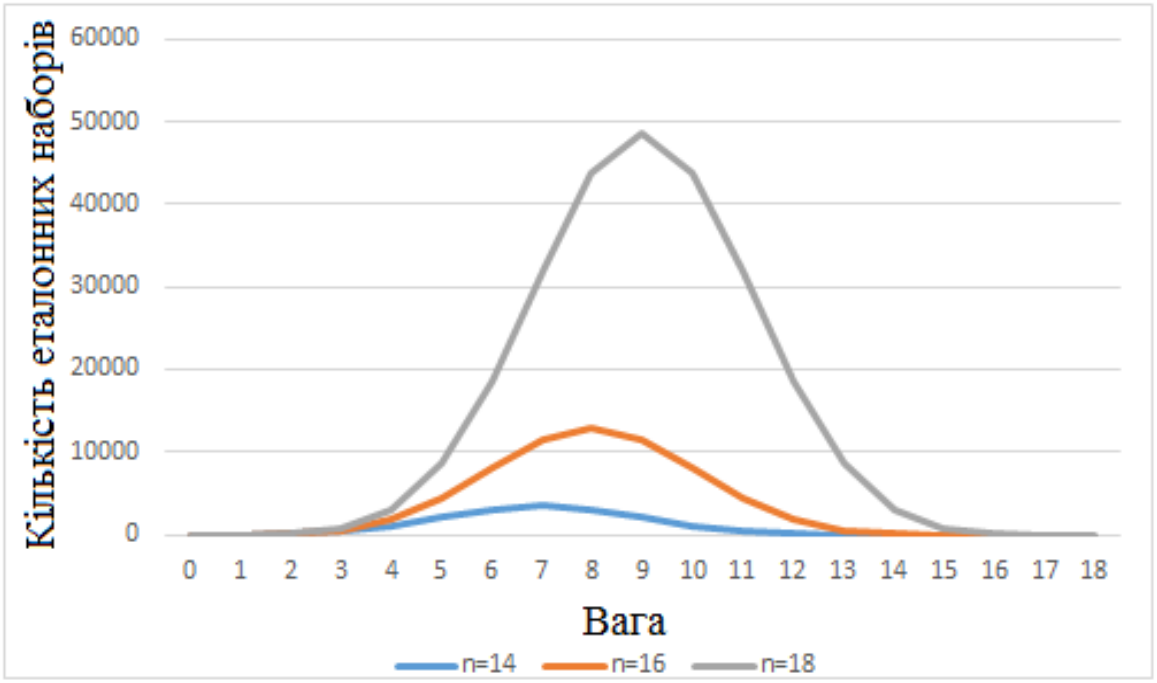


Рисунок 3.5 – Розподіл кількості еталонних наборів для 14,16,18-розрядного набору

Отже, розподіл кількості еталонних наборіб – це нормальний розподіл. Навіть для для набору з довжиною тридцять два розряди кількість еталонних наборів сягає значень дев’ятої степені десяти. Саме тому рекурсивний алгоритм для формування всіх еталонних наборів має особливе значення, тому що серед всіх значень n-розрядного двійкового набору (2 в степені n різних наборів) він отримує потрібні набори за кілікість тактів, яка не перевищує кількість еталонних наборів помножена на їх розрядність(один розряд в такт).

Отримані результати відносно ймовірність появи значення в заданому розряді двійкового набору занесені до Таблиці 3.2 та показані на Рисунку 3.6.

Розряд-ність n	Вага k	Розрахункова ймовірність появи значення в заданому розряді двійкового набору	Ймовірність появи значення в заданому розряді двійкового набору у програмі
8	3	0.37500	0.37822
8	6	0.75000	0.78846
16	8	0.50000	0.50009
16	11	0.68750	0.68568
32	14	0.43750	0.43750
32	7	0.21875	0.21874
10	2	0.20000	0.18196
12	5	0.41667	0.41369
14	4	0.28571	0.28360
20	11	0.55000	0.55005
22	13	0.59091	0.59093

Таблиця 3.2 - Результати тестування генератору на ймовірність появи значення в заданому розряді двійкового набору

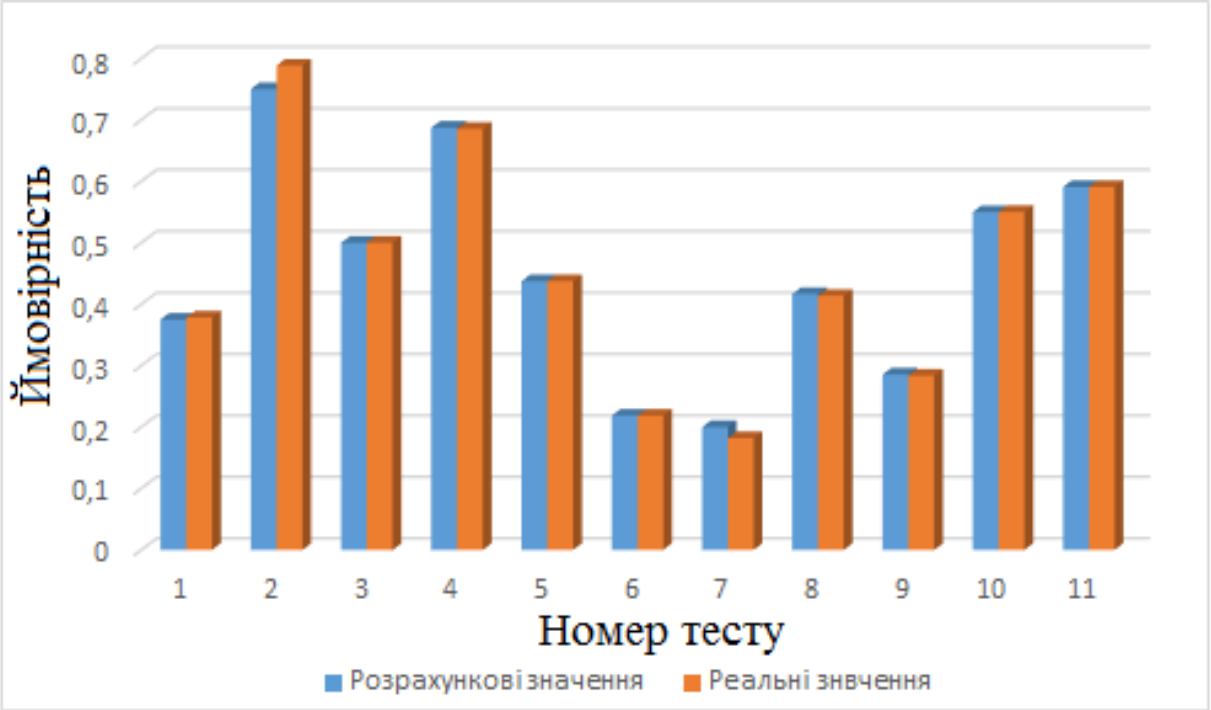


Рисунок 3.6 – Тестування ймовірності появи значення в заданому розряді двійкового набору для однієї ваги

Порівняємо кількість еталонних наборів у Таблиці 3.1 та результатом формули (3.6), застосованої до результатів Таблиці 3.2. Результати порівняння занесемо до Таблиці 3.3, при цьому розташуємо кількість еталонних наборів за зростанням.

Результат = |розрахункова ймовірність – ймовірність з програми| (3.6)

Таким чином, згідно з Таблицею 3.3, Рисунком 3.7 та Рисунком 3.8, чим більше наборів, тим менша похибка ймовірності появи значення в заданому розряді двійкового набору в програмі.

Кількість еталонних наборів	Результат за формулою (3.6)
28	0.03856
45	0.01804

56	0.00322
792	0.00298
1 001	0.00211
4 368	0.00182
12 870	0.00009
167 960	0.00005
497 420	0.00002
3 365 856	0.00001
471 435 600	0.00000

Таблиця 3.3 - Залежність похибки ймовірності від кількості еталонних наборів

Отже, для того, щоб похибка не перевищувала 0,4% рекомендується, щоб кількість еталонних наборв була більша ніж 56.

Рисунок 3.7 – Залежність похибки від кількості еталонних наборів

Рисунок 3.8 - Залежність похибки від кількості еталонних наборів

Для кількості різних ваг m, при чому ймовірність появи набору ваг рівні між собою, формула ймовірності появи значення в заданому розряді двійкового набору (3.4) зміниться наступним чином (3.6), де p – ймовірність, k_i – вага i-го набору, n – розрядність набору.

$$p = \frac{k_1+k_2+\ldots+k_i+k_{i+1}\ldots+k_m}{m*n} \tag{3.6}$$

При кількості ваг більше однієї розглядаються випадки розподілу:

- рівномірний (ймовірність появи двійкового набору кожної різної ваги однакова);
- розподіл, заданий користувачем.

По аналогії з Таблицею 3.2 отримаємо результати відносно ймовірності появи значення в заданому розряді двійкового набору для двох ваг з рівною ймовірністю появи нібору заданої ваги(0.5), що занесено до Таблиці 3.4 та показано на Рисунку 3.10. На Рисинку 3.9 відображені реальні значення ймовірності появи набору кожної з двох ваг.

Розряд-ність n	Ваги k_i	Розрахункова ймовірність появи значення в заданому розряді двійкового набору	Ймовірність появи значення в заданому розряді двійкового набору у програмі
8	3,5	0.50000	0.50316
8	6,4	0.62500	0.62188
16	8,10	0.56250	0.56360
16	11,12	0.71875	0.71767
32	14,17	0.48438	0.48437
32	7,5	0.18750	0.18752
10	2,5	0.35000	0.35212
12	5,8	0.54167	0.54446
14	4,9	0.46429	0.46618
20	11,14	0.62500	0.62402
22	13,15	0.63636	0.63548

Таблиця 3.4 - Результати тестування генератору на ймовірність появи значення в заданому розряді двійкового набору для двох ваг з рівною ймовірністю появи

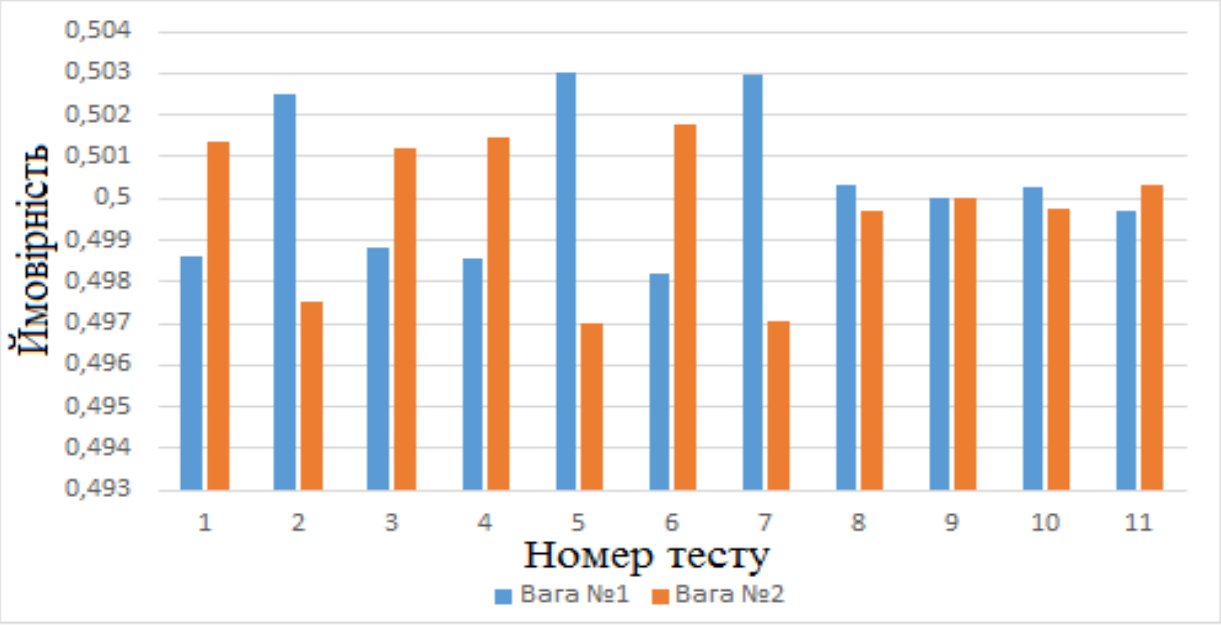


Рисунок 3.9 – Ймовірність появи набору заданої ваги

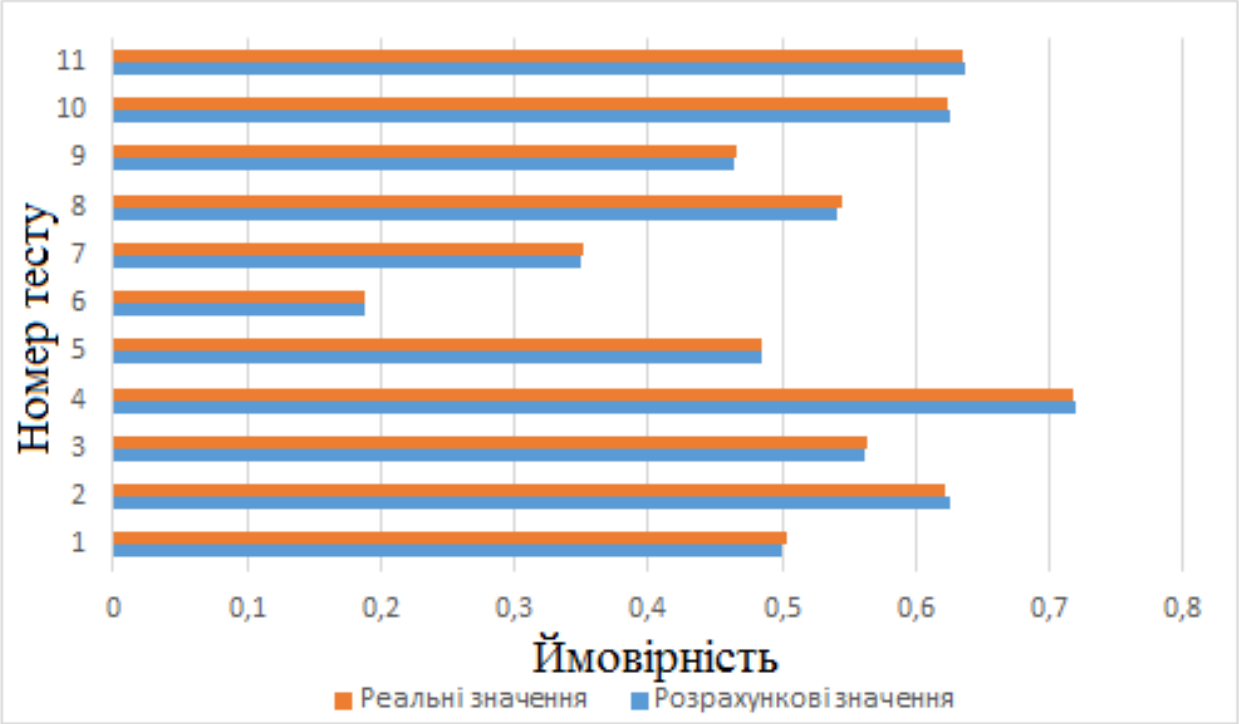


Рисунок 3.10 - Тестування генератору на ймовірність появи значення в заданому розряді набору для двох ваг з рівною ймовірністю появи

Для кількості різних ваг m, при чому ймовірність появи набору ваг різні, формула ймовірності появи значення в заданому розряді двійкового набору (3.4) зміниться наступним чином (3.7), де p – ймовірність появи «1» у будь-якому розряді набору, k_i – вага і-го набору, p_i – ймовірність появи набору з вагою k_i, n – розрядність набору.

$$p = \frac{p_{1*k_1} + p_{2*k_2} + \cdots + p_{i*k_i} + p_{i+1*k_{i+1}} + \cdots + p_{m*k_m}}{n} \quad (3.7)$$

По аналогії з Таблицею 3.2 отримаємо результати відносно ймовірності появи значення в заданому розряді двійкового набору для двох ваг з різною ймовірністю появи нібору заданої ваги(0.25 та 0.75 для першої та другої ваги відповідно), що занесено до Таблиці 3.5 та показано на Рисунку 3.12. На Рисинку 3.11 відображені реальні значення ймовірності появи набору кожної з двох ваг.

Розряд-ність n	Ваги k_i	Розрахункова ймовірність появи значення в заданому розряді двійкового набору	Ймовірність появи значення в заданому розряді двійкового набору у програмі
8	3,5	0.56250	0.56580
8	5,4	0.53125	0.53501
16	8,10	0.59375	0.59497
16	11,12	0.73438	0.73327
32	14,17	0.50781	0.50782
32	7,5	0.17188	0.17190
10	2,5	0.42500	0.42219
12	5,8	0.60417	0.60686
14	4,9	0.55357	0.55545
20	11,14	0.66250	0.66360
22	13,15	0.65909	0.66001

Таблиця 3.5 - Результати тестування генератору на ймовірність появи значення в заданому розряді двійкового набору для двох ваг з різною ймовірністю появи

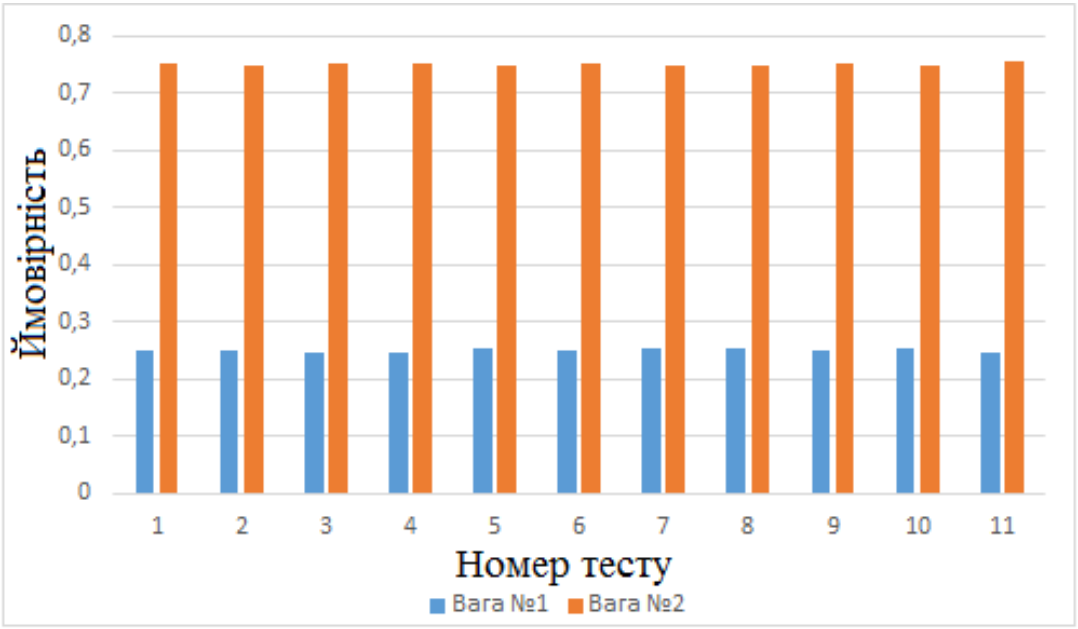


Рисунок 3.11 - Ймовірність появи набору заданої ваги

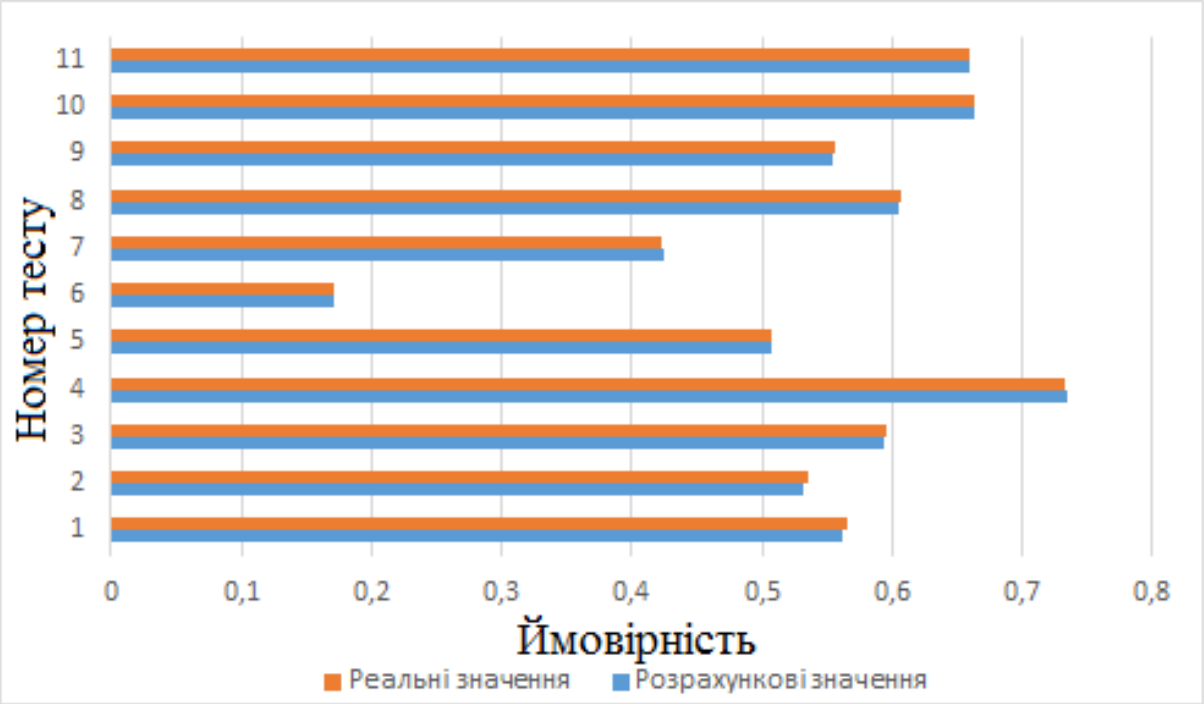


Рисунок 3.12 - Тестування генератору на ймовірність появи значення в заданому розряді набору для двох ваг з різною ймовірністю появи

Обрахуємо результати за формулою (3.6) для Таблиці 3.4, Таблиці 3.5 та розмістимо у Таблиці 3.6.

Для Таблиці 3.4	Для Таблиці 3.5
0,00316	0,00330
0,00312	0,00376
0,00110	0,00122
0,00108	0,00111
0,00001	0,00001
0,00002	0,00002
0,00212	0,00281
0,00279	0,00269
0,00189	0,00188
0,00098	0,00110
0,00088	0,00092

Таблиця 3.6 - Результати розрахунків за формулою (3.6)

Згідно Таблиці 3.6 похибка не перевищує 0,4% для заданих характеристик генератора.

Перевірка розподілу ймовірності появи набору заданої ваги зображено на Рисунку 3.13.

Для більшої кількості ваг ця перевірка приведено у вигляді діаграм на Рисунку 3.14, Рисунку 3.15 та Рисунку 3.16.

В результаті аналізу цих діаграм виявлено, що похибка розподілу ймовірності появи набору заданої ваги не пов’язана з кількістю різних ваг.

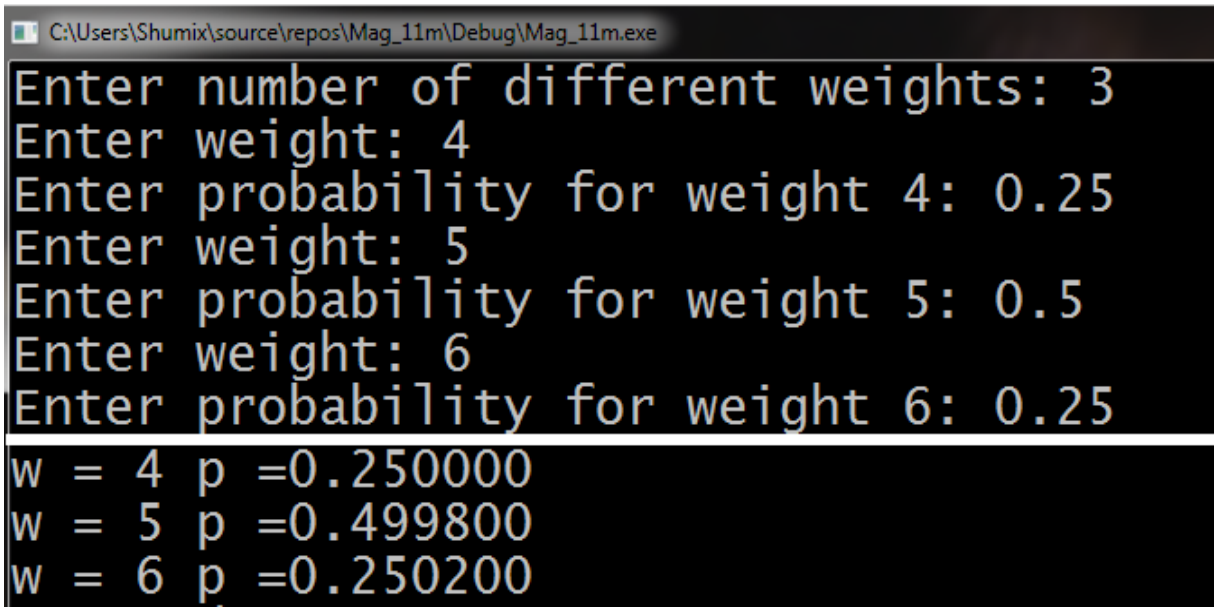


Рисунок 3.13 Задання ймовірності появи заданої ваги та результат тестування

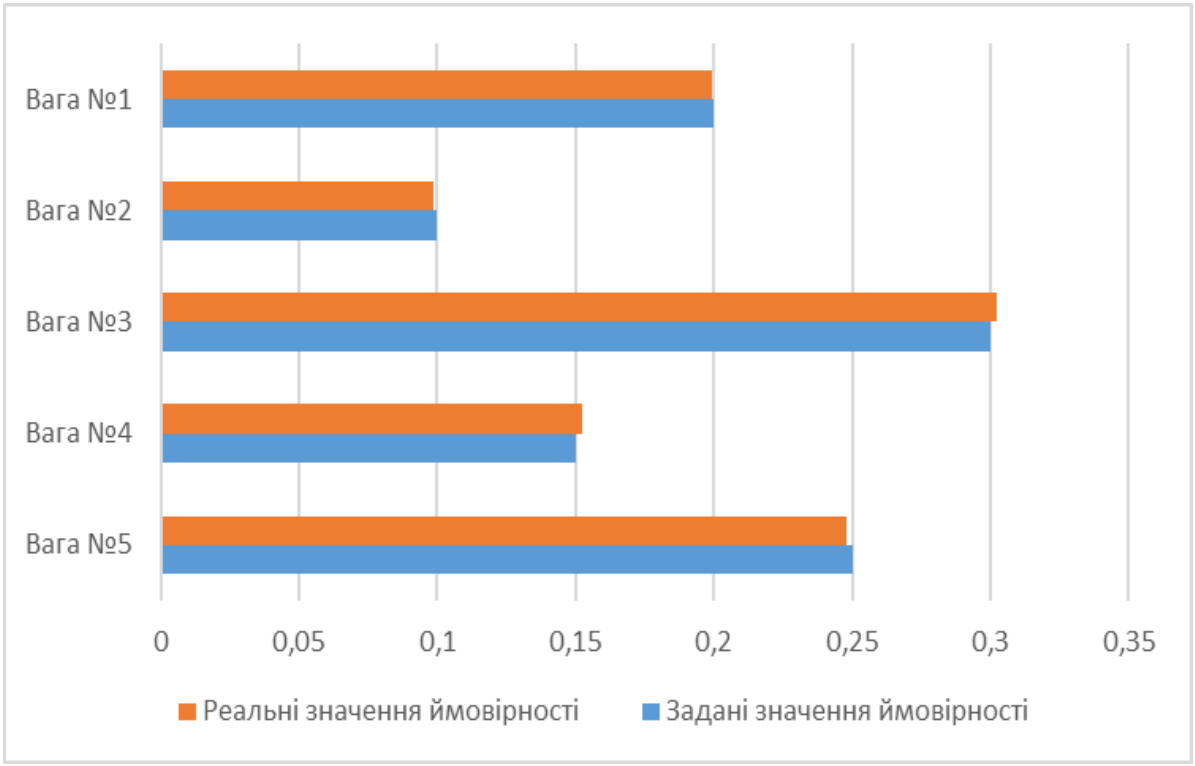


Рисунок 3.14 – Порівняння розподілу ймовірності появи набору заданної ваги для п’яти різних ваг

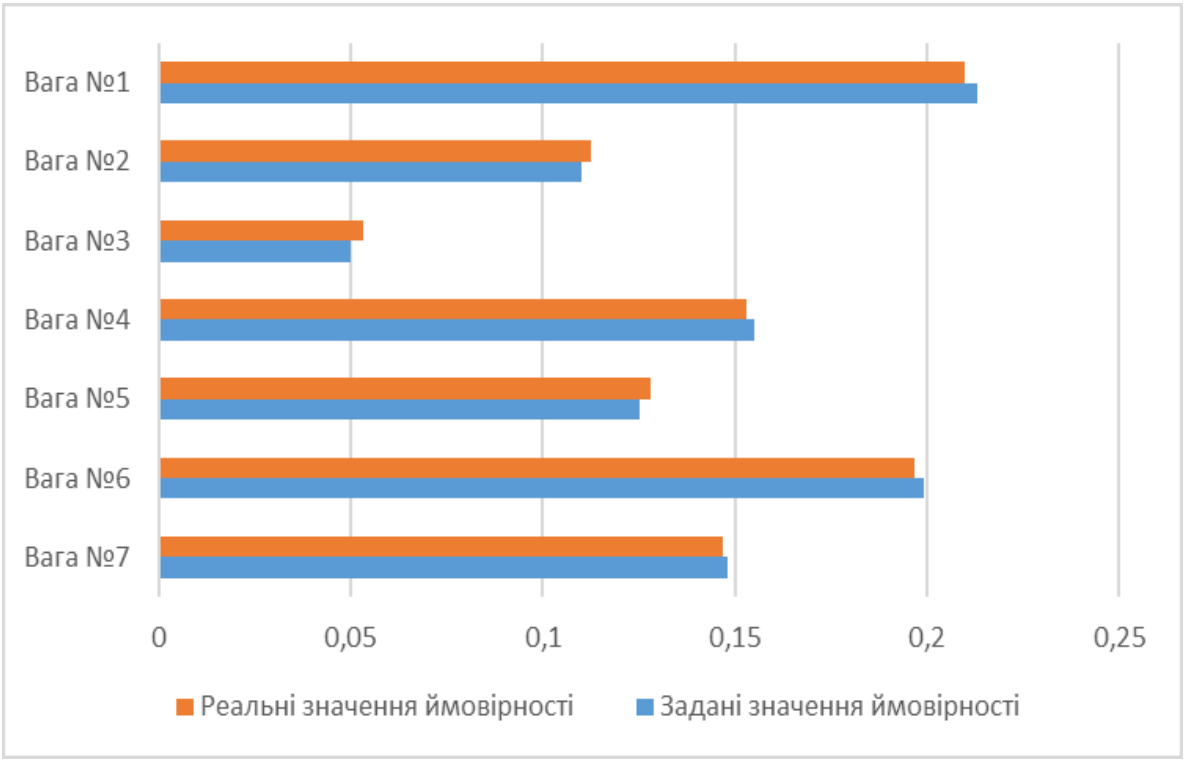


Рисунок 3.15 – Порівняння розподілу ймовірності появи набору заданної ваги для семи різних ваг

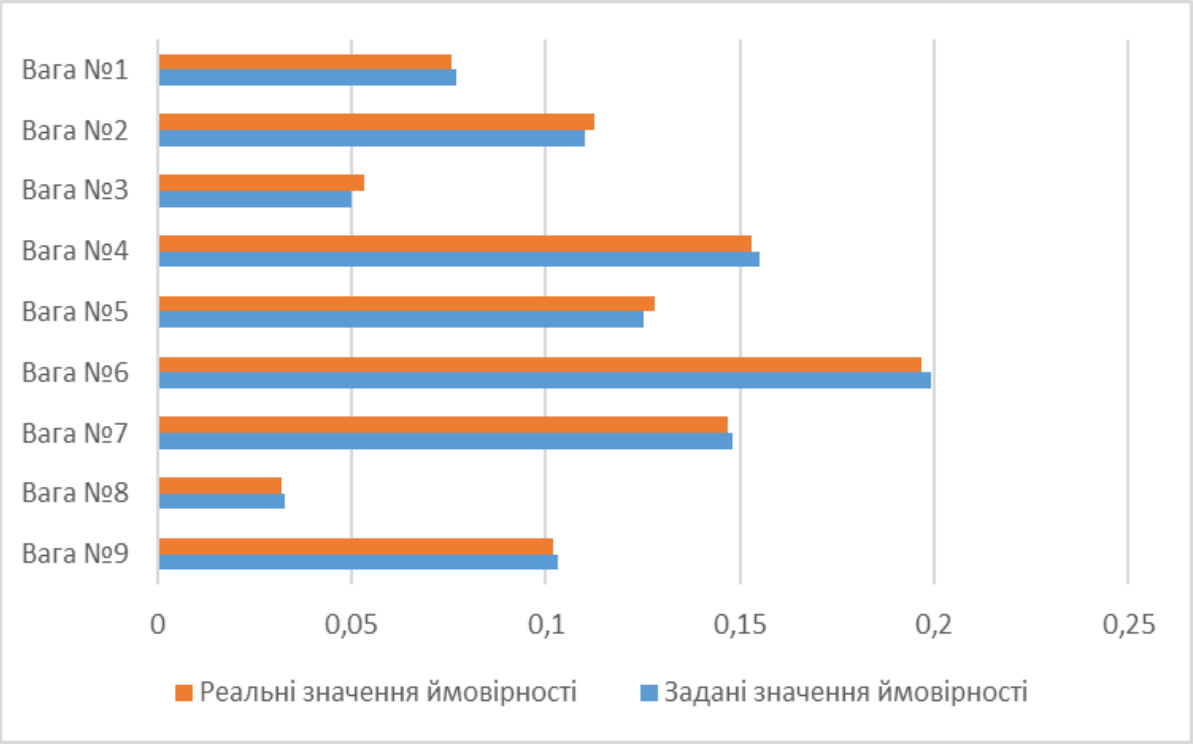


Рисунок 3.16 – Порівняння розподілу ймовірності появи набору заданної ваги дев’яти семи різних ваг

Вище була описана можливість отримання кількості повторів еталонних наборів, які були згенеровані у момент роботи керованого генератору псевдовипадкових двійкових наборів. Малюнок діаграми цих результатів представлений на Рисунку 3.17.

Кількість повторів знаходиться у певному діапазоні, що підтверджує те, що розроблений генератор псевдовипадкових двійкових наборів генерує набори з квазірвномірного розподілу для наборів однієї ваги(гістограма розподілу елементів послідовності).

Рисунок 3.17 - Кількість повторів еталонних наборів через деякий час роботи генератору

Перевірка квазірвномірного розподілу для наборів однієї ваги при генеруванні наборів декількох різних ваг зображено на Рисунку 3.18, Рисунку 3.19.

Оскільки при різній вазі наборів кількість еталонних наборів змінюється, то тривалість ліній на Рисунку 3.9, Рисунку 3.10 відрізняється. Для наглядності кількості повторів еталонних наборів графіки зображені один під одним, але самі еталонні набори для різних ваг відрізняються. Ймовірності появи набору заданої ваги відсортовані таким чином, що зі зростанням номеру ваги, зростає ймовірність появи набру з цією вагою.

Аналіз цих графіків показав:

- квазірівномірність розподілу в межах однієї ваги;
- залежність кількості повторів еталонних наборів від ймовірності появи набору заданої ваги.

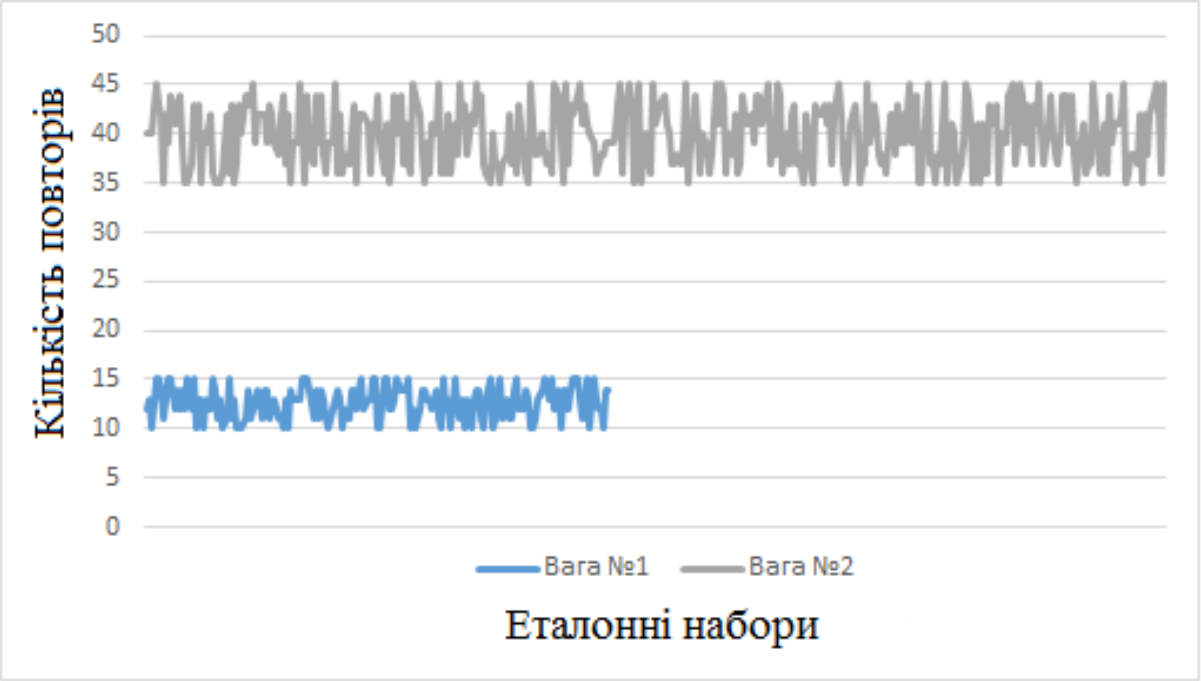


Рисунок 3.18 – Кількість повторів наборів заданої ваги для двох різних ваг

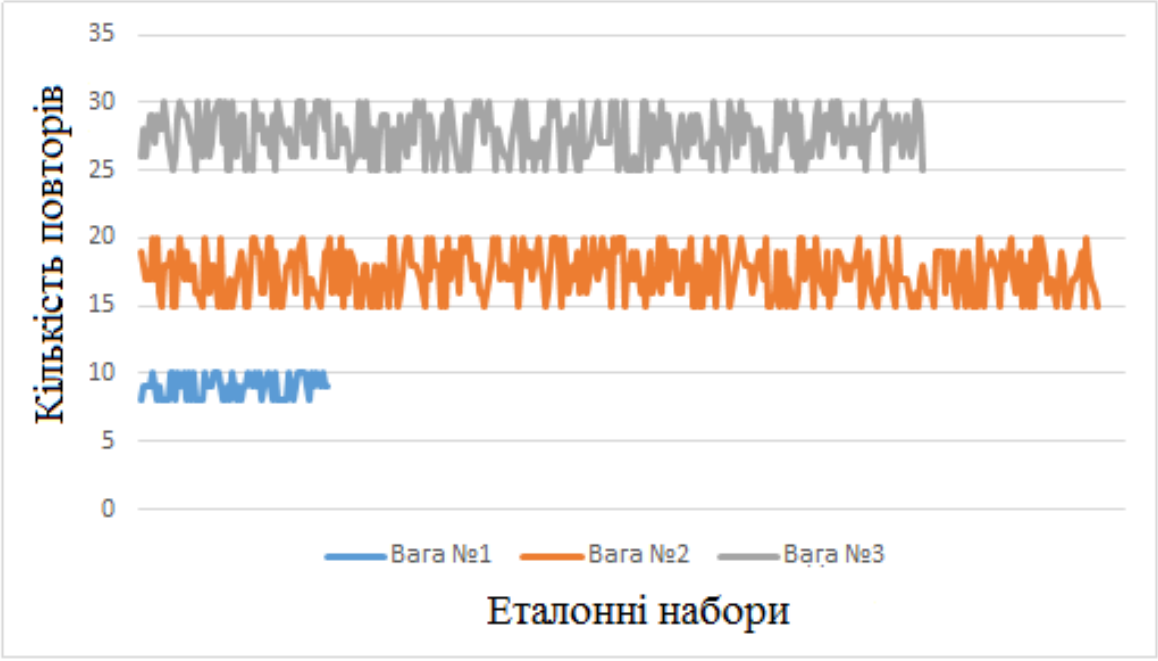


Рисунок 3.19 – Кількість повторів наборів заданої ваги для трьох різних ваг

3.2 Висновки до розділу 3

В результаті аналізу генератора виявлено, що набори, які були отримані на виході керованого генератора мають наступні властивості:

- випадковість;
- квазірівномірний розподіл наборів в межах однієї ваги(за гістограмою розподілу елементів послідовності);
- заданий користувачем розподіл ймовірності появи набору заданої ваги.

Проведена апробація керованого генератору псевдовипадкових двійкових наборів з рівною вагою за такими критеріями:

- ймовірність появи значення в заданому розряді двійкового набору;
- кількість еталонних наборів;
- кількість повторів еталонних наборів через заданий час роботи генератору.

Похибка розподілу ймовірності появи набору заданої ваги не пов’язана з кількістю різних ваг.

Похибка розподілу ймовірності появи набору заданої ваги складається з двох частин:

- похибка через кількість еталонних наборів;
- похибка розподілу ваг в елементі пам’яті.

Для того, щоб похибка через кількість еталонних наборів не перевищувала 0,4% рекомендується, щоб їх кількість перевищувала 56.

Для того, щоб похибка розподілу ваг в елементі пам’яті не перевищувала 0,4% рекомендується, щоб кількість рядків елемента пам’яті перевищувала 256.

Похибка ймовірністі появи значення в заданому розряді двійкового набору не перевищена при генеруванні наборів різних ваг.

Результати апробації генератору підтвердили збіг розрахункових результатів та результатів, отриманих на виході генератору програмної моделі.

ВИСНОВКИ

Реалізація методу являє собою керований генератор псевдовипадкових двійкових наборів з заданим розподілом ймовірності появи наборів заданої ваги.

Аналіз публікацій, присвячених теоретичним і експериментальним дослідженням генераторів псевдовипадкових послідовностей показав, що розробка генераторів являє собою перспективний напрямок у розвитку імітаційного моделювання, шифрування данних та криптографії, а також тестуванні багатопроцесорних систем.

Проектування на основі регістра зсуву дозволяє збільшувати розрядність двійкового набору, що збільшить період послідовності, без суттєвої зміни схеми.

Створено алгортм заповнення елемента пам’яті для даного генератора.

Створено рекурсивний алгоритм для отримання всіх можливих еталонних двійкових наборів заданої ваги.

Частина формувача вхідних даних, що забезпечує генерування наборів заданої ваги запатентовано як корусна модель[21].

Вперше формувач може генерувати дані за будь-яким розподілом. Існуючі алгоритми генерують лише один розподіл ймовірності появи наборів.

В результаті аналізу тестування програмної моделі генератора набори, які були отримані на виході мають наступні властивості:

- випадковість;
- квазірівномірний розподіл наборів в межах однієї ваги;

- заданий користувачем розподіл ймовірності появи набору заданої ваги.

Апробацію керованого генератору псевдовипадкових двійкових наборів проведено за критеріями:

- ймовірність появи значення в заданому розряді двійкового набору;
- кількість еталонних наборів;
- кількість повторів еталонних наборів через заданий час роботи генератору.

Похибка розподілу ймовірності появи набору заданої ваги не пов’язана з кількістю різних ваг. На цю похибку впливають фактори:

- похибка через кількість еталонних наборів;
- похибка розподілу ваг в елементі пам’яті.

Для того, щоб похибка через кількість еталонних наборів не перевищувала 0,4% рекомендується, щоб їх кількість перевищувала 56.

Для того, щоб похибка розподілу ваг в елементі пам’яті не перевищувала 0,4% рекомендується, щоб кількість рядків елемента пам’яті перевищувала 256.

Результати апробації генератору підтвердили збіг розрахункових результатів та результатів, отриманих на виході генератору програмної моделі.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Создание случайных чисел : [Електронний ресурс]. — Режим доступу: http://www.intuit.ru/studies/courses/28/28/lecture/20414?page=6 — (22.03.2018).

2. Модели в теории биологической эволюции: [Електронний ресурс]. — Режим доступу: http://earthpapers.net/modeli-v-teorii-biologicheskoy-evolyutsii — (01.10.2017).

3. Analog and Mixed-Signal Test Methods Using On-Chip Embedded Test Cores: [Електронний ресурс]. — Режим доступу: http://digitool.library.mcgill.ca/webclient/StreamGate?folder_id=0&dvs=1524785121425~634 — (17.11.2017).

4. Characterization of a Pseudo-Random Testing Technique for Analog and Mixed-Signal Built-in-Self-Test: [Електронний ресурс]. — Режим доступу: https://www.researchgate.net/publication/221203272_Characterization_of_a_Pseudo-Random_Testing_Technique_for_Analog_and_Mixed-Signal_Built-in-Self-Test — (12.12.2017).

5. Приложение теории детерминированного хаоса в криптографии: [Електронний ресурс]. — Режим доступу: http://padaread.com/?book=2672&pg=4 — (10.01.2018).

6. Random bit generation ISO/IEC 18031: [Електронний ресурс]. — Режим доступу: https://www.iso.org/standard/30816.html — (10.09.2017).

7. Метод Фибоначчи с запаздыванием: [Електронний ресурс]. — Режим доступу: http://www.intuit.ru/studies/courses/691/547/lecture/12383?page=2 (11.04. 2018).

8. Вихрь Мерсенна: [Електронний ресурс]. — Режим доступу: http://gamesmaker.ru/programming/c/generaciya-sluchaynyh-chisel/ — (05.04. 2018).

9. Зсувні регістри з лінійної зворотним зв'язком: [Електронний ресурс]. — Режим доступу: http://um.co.ua/14/14-7/14-73899.html — (06.04. 2018).

10. Самоуправляемый 2-линейный регистр сдвига: [Електронний ресурс]. — Режим доступу: http://studbooks.net/2186312/matematika_himiya_fizika/kvadratichnyy_kongruentnyy_generator — (06.04. 2018).

11. RSA АЛГОРИТМ: [Електронний ресурс]. — Режим доступу: https://kovalchukmm14.wordpress.com/2014/12/16/rsa-алгоритм — (07.04. 2018).

12. Линейный конгруэнтный генератор псевдослучайных чисел и метод раскрытия его параметров [Електронний ресурс]. — Режим доступу: https://nauchforum.ru/studconf/tech/xli/17030 — (08.04. 2018).

13. Алгоритмічні методи генерації псевдовипадкових чисел за рівномірним законом розподілу [Електронний ресурс]. — Режим доступу: https://knhelp.wordpress.com/2012/05/03/лаб-4-алгоритмічні-методи-генерації-пс/ — (09.04. 2018).

14. Графические тесты: [Електронний ресурс]. — Режим доступу: https://studfiles.net/preview/272674/page:21/ — (14.02.2018).

15. Тестирование генератора [текст]. / Теория, применение и оценка качества генераторов псевдослучайных последовательностей — М.: КУДИЦ-Образ, 2003. Кн. 2. — 240 с.

16. Тестирование псевдослучайных последовательностей - Статистические тесты: [Електронний ресурс].— Режим доступу: http://chinapads.ru/c/s/testirovanie_psevdosluchaynyih_posledovatelnostey_-_statisticheskie_testyi — (15.02.2018).

17. The Marsaglia random number CDROM including the DIEHARD battery of tests of randomness web site: [Електронний ресурс]. — Режим доступу: http://stat.fsu.edu/pub/diehard — (05.03.2018).

18. Статистическая проверка случайности двоичных последовательностей методами NIST: [Електронний ресурс]. — Режим доступу: https://habrahabr.ru/company/securitycode/blog/237 695 — (07.03.2018).

19. Статистические тесты NIST : [Електронний ресурс]. — Режим доступу: http://sibac.info/studconf/tech/xxxi/41826. — (15.03.2018).

20. А.М. Романкевич Граничные оценки числа рёбер GL-моделей поведения отказоустойчивых многопроцессорных систем в потоке отказов / А.М. Романкевич, В.А. Романкевич, И.В. Майданюк — Киев: Электронное моделирование, 2008. Т. 30, № 1. – С. 59–70.

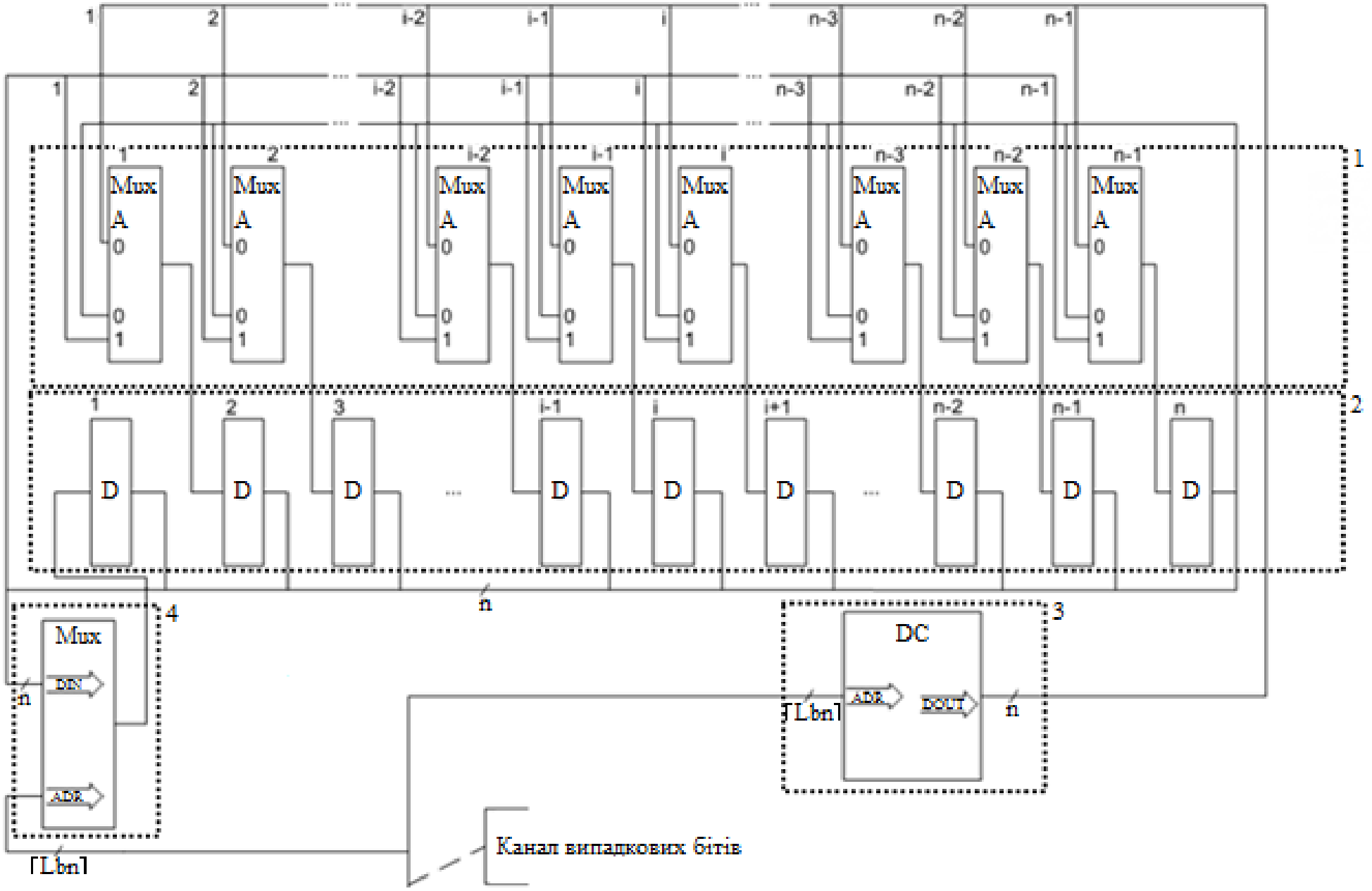
21. Пат. 119842 Україна, МПК G06F 7/58, G06F 11/263. Генератор псевдовипадкових двійкових наборів з рівною вагою / Ткаченко М.Г.; заявник і власник патенту Ткаченко М.Г. – № u 2017 03948; заявл. 21.04.17 ; опубл. 10.10.17, Бюл. № 29.

22. Розподільники і мультиплексори: [Електронний ресурс]. — Режим доступу: http://stud.com.ua/28286/tovaroznavstvo/rozpodilniki_multi pleksori. — (16.03.2018).

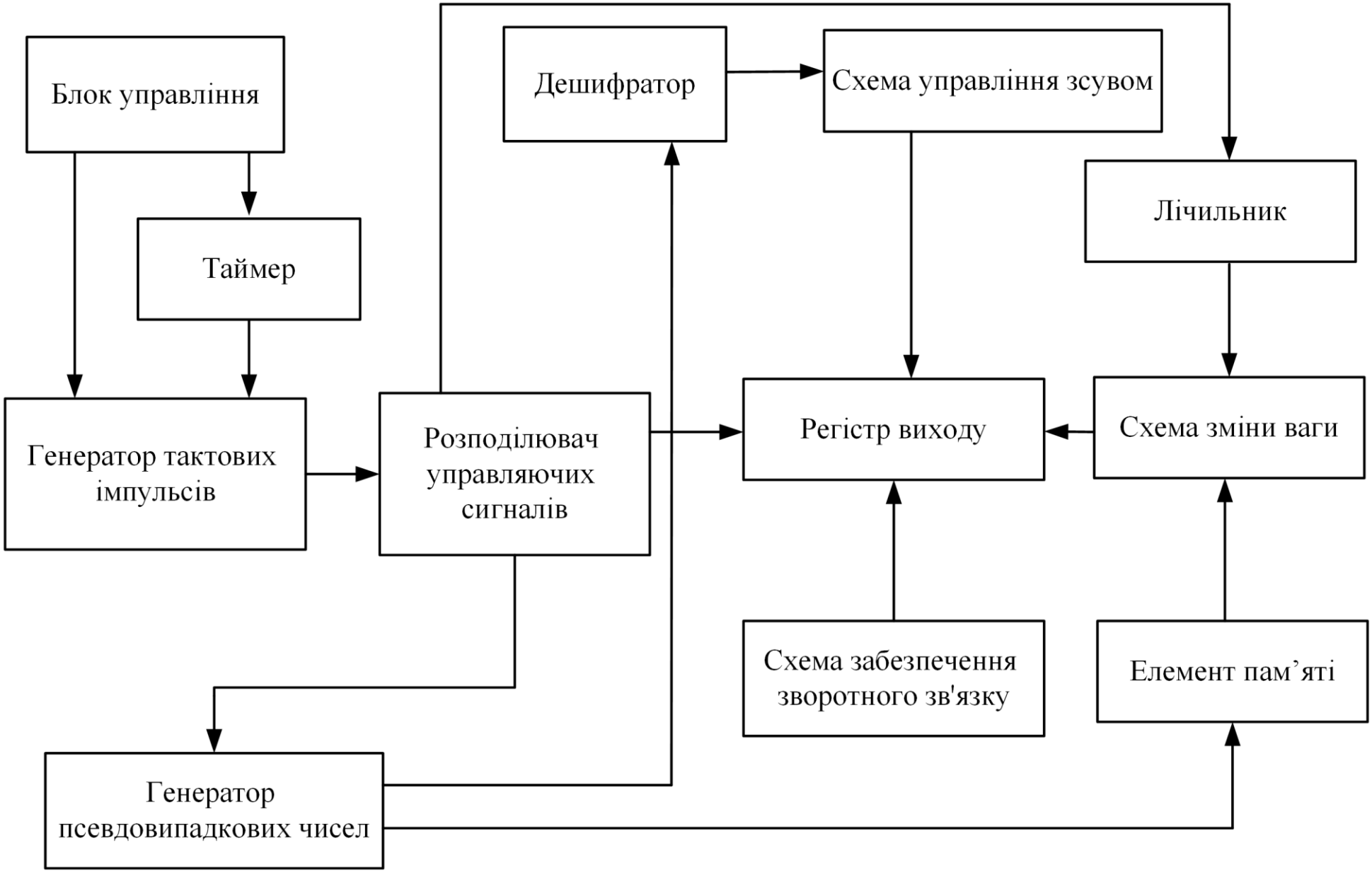
23. Ассоциативный масив: [Електронний ресурс]. — Режим доступу: http://gamesmaker.ru/programming/c/izuchaem-map-associativnyy-massiv-c/ — (28.03.2018).

24. Абстрактный тип данных: [Електронний ресурс]. — Режим доступу: http://prog-cpp.ru/cpp-atd/— (01.04. 2018).

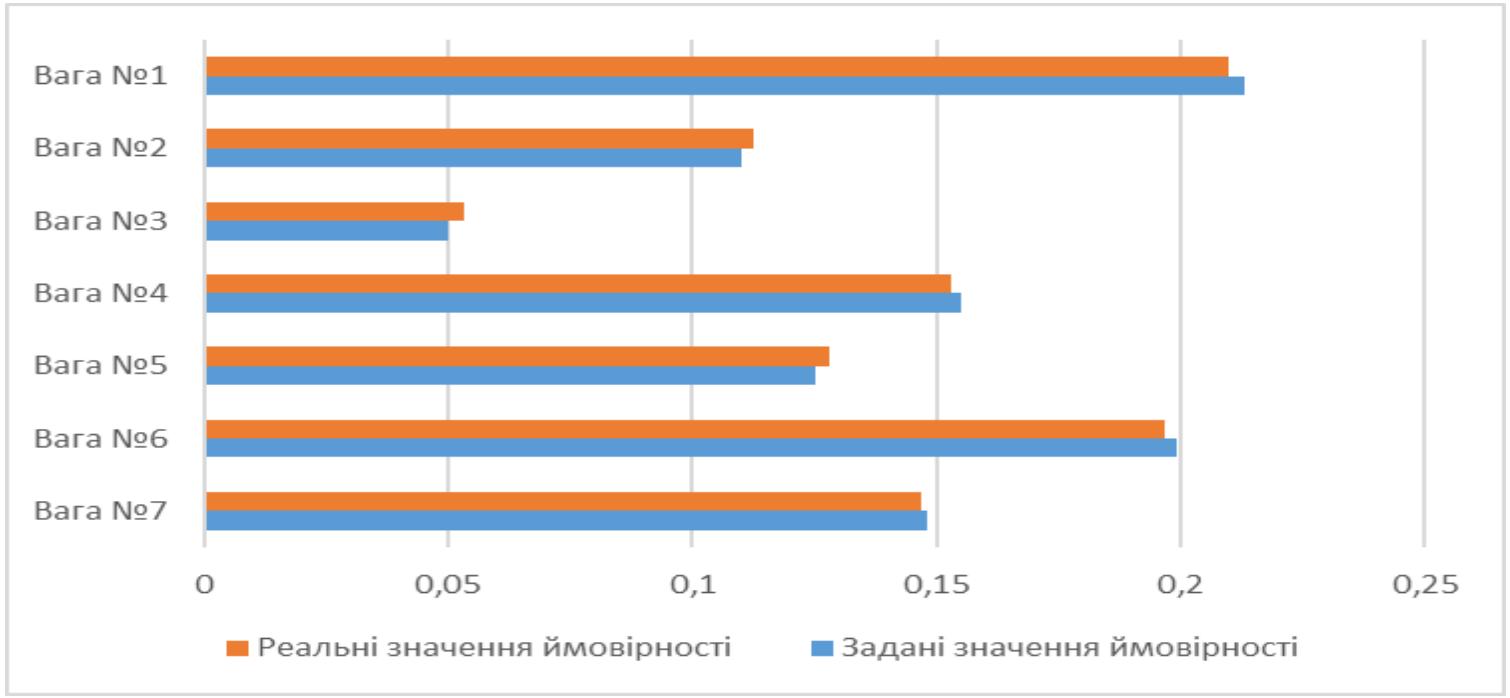
25. string (C++): [Електронний ресурс]. — Режим доступу: http://www.cplusplus.com/reference/string/string/ — (03.04. 2018).

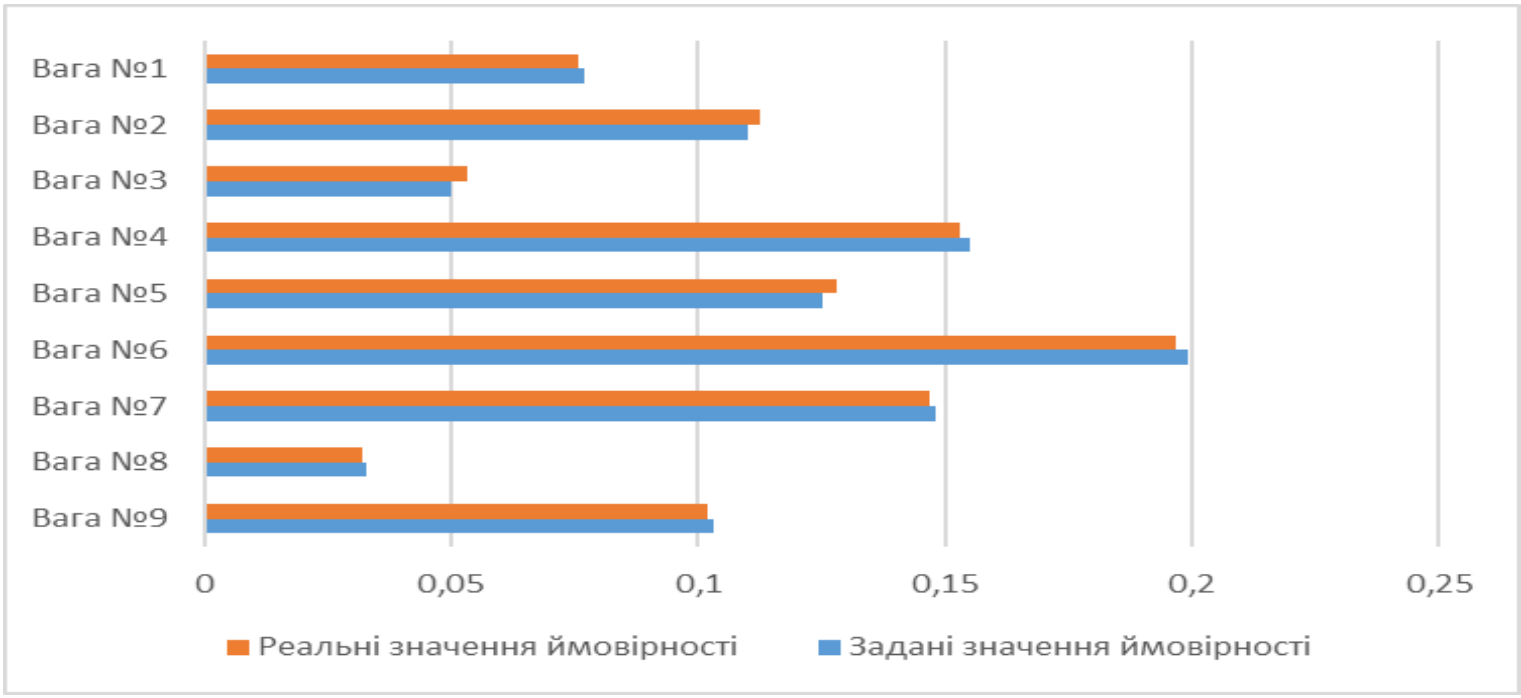


Креслення до патенту на корисну модель

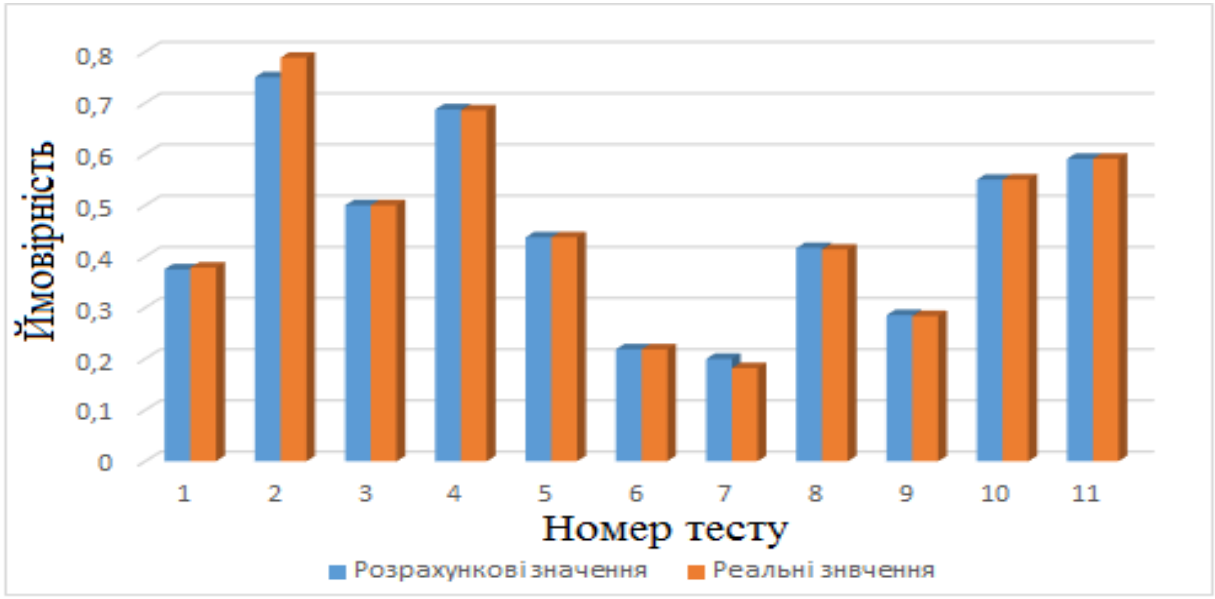


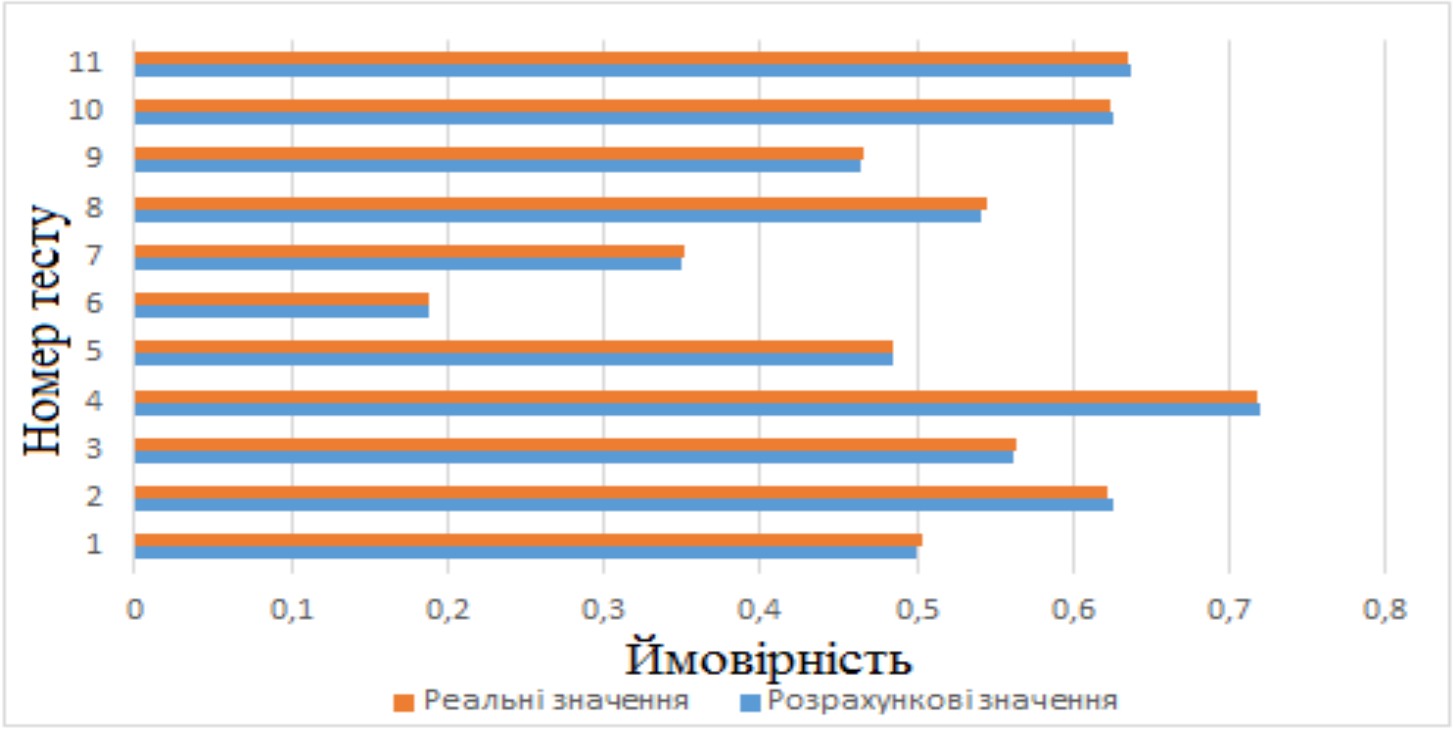
Загальна структура апаратної реалізації формувача



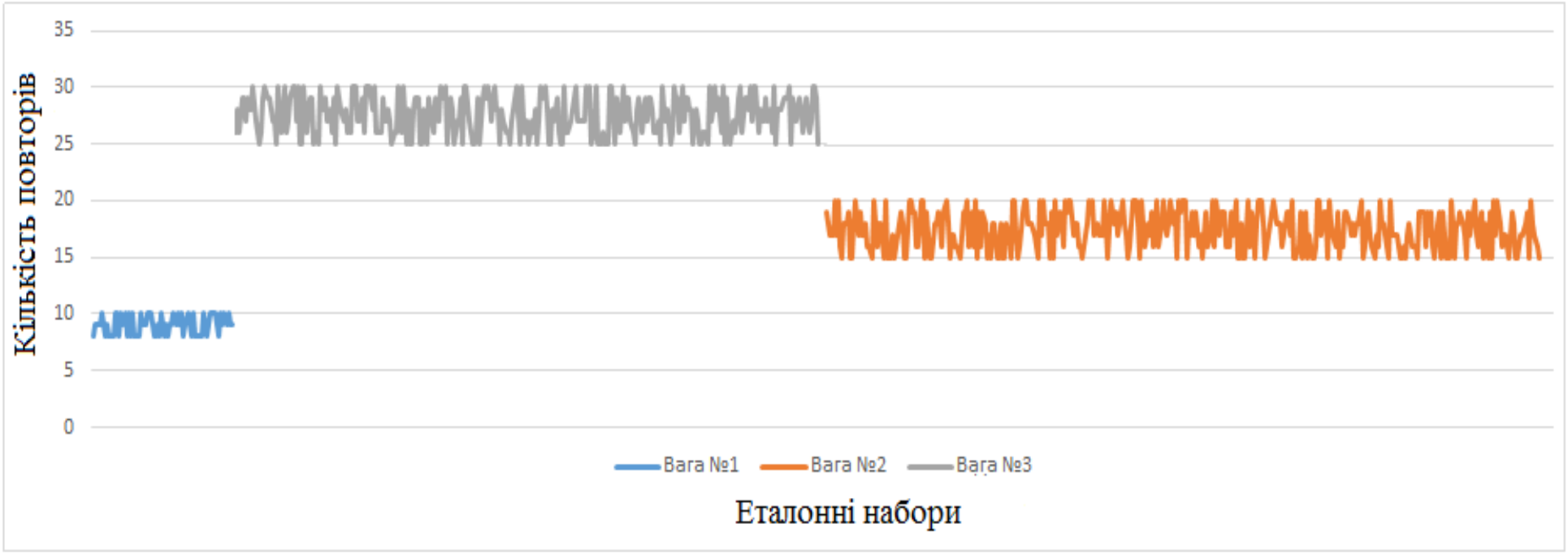


Ймовірнісні розподіли появи набору за його вагою, задані користувачем

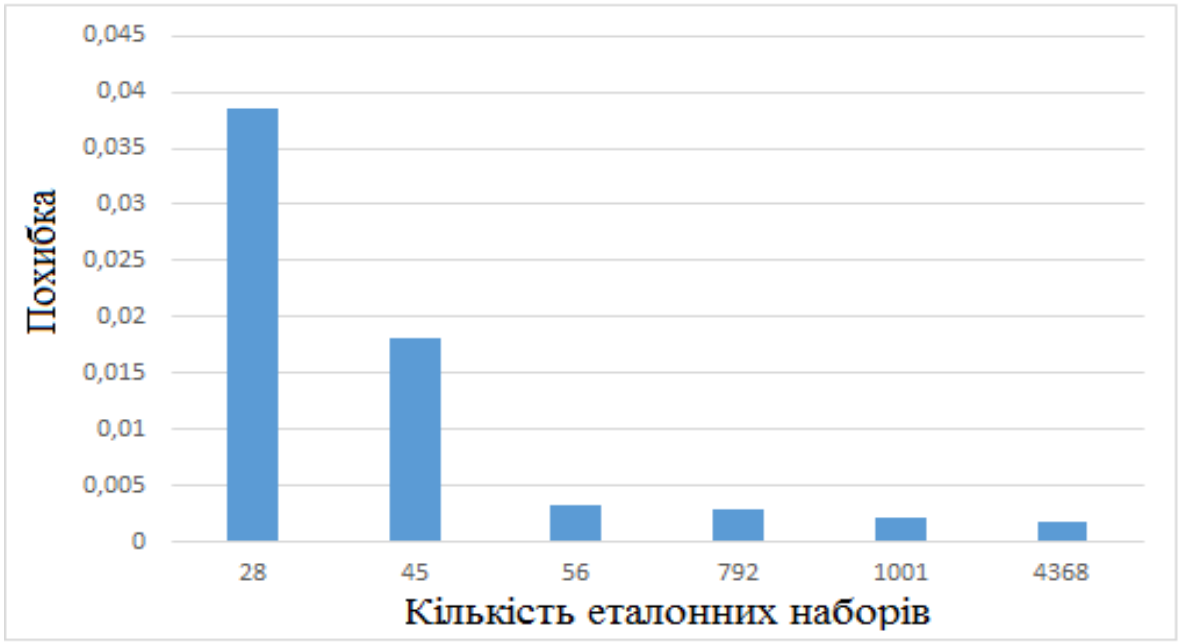


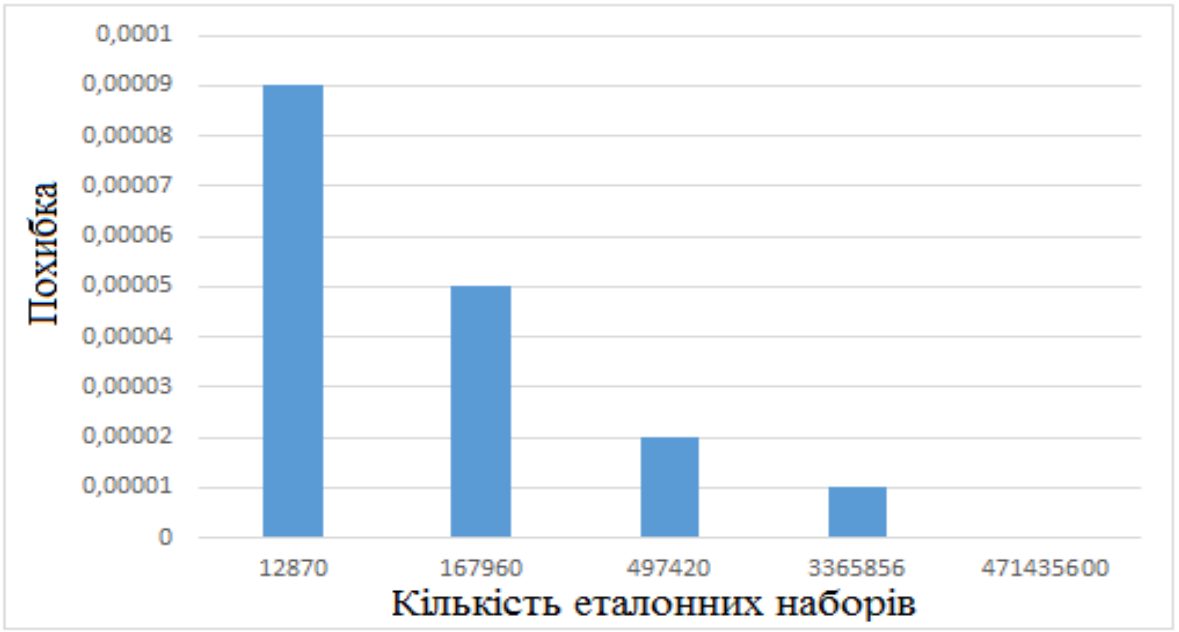


Ймовірність появи заданого значення на одному з розрядів вихідного регістра



Кількість повторів кожного з існуючих наборів заданої ваги





Залежність похибки ймовірності появи заданого значення на одному розряді набору від кількості еталонних наборів

УКРАЇНА **(19) UA** **(11) 119842** **(13) U**

(51) МПК
G06F 7/58 (2006.01)
G06F 11/263 (2006.01)

МІНІСТЕРСТВО ЕКОНОМІЧНОГО РОЗВИТКУ І ТОРГІВЛІ УКРАЇНИ

(12) ОПИС ДО ПАТЕНТУ НА КОРИСНУ МОДЕЛЬ

(21) Номер заявки: **u 2017 03948**
(22) Дата подання заявки: **21.04.2017**
(24) Дата, з якої є чинними права на корисну модель: **10.10.2017**
(46) Публікація відомостей про видачу патенту: **10.10.2017, Бюл.№ 19**

(72) Винахідник(и):
Ткаченко Михайло Геннадійович (UA)
(73) Власник(и):
Ткаченко Михайло Геннадійович,
пров. Ковальський, 5, м. Київ, 03057 (UA)

(54) ГЕНЕРАТОР ПСЕВДОВИПАДКОВИХ ДВІЙКОВИХ НАБОРІВ З РІВНОЮ ВАГОЮ

(57) Реферат:

Генератор псевдовипадкових двійкових наборів з рівною вагою містить багатоканальний вузол генерації випадкових бітів, кожен канал в якому включає в себе послідовно з'єднані генератор шуму, підсилювач-обмежувач і лічильний тригер, виходи усіх вузлів об'єднані схемою "ВИКЛЮЧНЕ АБО", і канал спряження з ПЕОМ, що включає в себе регістр зсуву, виходи якого увімкнуті до входів вихідного регістра, з'єднаного виходами з шиною даних ПЕОМ, тактовий генератор, вихід якого з'єднаний з синхровходом регістра зсуву та входом лічильника імпульсів, вихід якого під'єднаний до синхровходу вихідного регістра та входу тригера "прапора", а його вихід з'єднаний з виходом запиту переривання та через буферний елемент І з шиною даних ПЕОМ, і дешифратор адреси, включений входами до шини адрес ПЕОМ, а першим виходом до входу дозволу вихідного регістра та входу скидання тригера "прапора", а другим виходом до буферного елемента І, причому присутні n-1 одноадресних мультиплексорів 1, n D-тригерів 2, дешифратор 3, [Lbn]-адресний мультиплексор 4, причому на адресні входи n-1 одноадресних мультиплексорів 1 підключено 1…(n-1) розряди виходу дешифратора 3, відповідно, а на інформаційні входи подається значення 1…(n-1)-го D-тригера відповідно та n-ний D-тригер, на вхід 2…n D-тригерів 2 підключені виходи n-1 одноадресних мультиплексорів 1, а на вхід 1-го D-тригера 2 підключений вихід [Lbn]-адресного мультиплексора 4, на n інформаційних входів [Lbn]-адресного мультиплексору 4 підключено n D-тригерів 2, а на адресні входи подається канал випадкових бітів, також він подається й на вхід дешифратора 3.

UA 119842 U

Корисна модель належить до обчислювальної техніки, а саме до генераторів вхідних послідовностей для тестування цифрових пристроїв, і може бути використана в машинобудівних технологіях.

Відомий лічильник Джонсона на основі замкнутого регістра зсуву з одним перехресним (інверсним) зв'язком, вихід останнього розряду регістра з'єднаний з входом D-тригера, а на 5 інформаційний вхід регістра подано сигнал з інверсного виходу тригера [1]. Недоліки аналога: при розрядності n($n \geq 2$) лічильника Джонсона на вихід пристрою видається 2n двійкових наборів, тобто генерується обмежена множина різних двійкових наборів.

Найбільш близьким по сукупності ознак є генератор рівномірно розподілених ймовірних чисел [2], що містить багатоканальний вузол генерації випадкових бітів, кожен канал в якому включає в себе послідовно з'єднані генератор шуму, підсилювач-обмежувач і лічильний тригер, виходи усіх вузлів об'єднані схемою "ВИКЛЮЧНЕ АБО", і канал спряження з ПЕОМ, що включає в себе регістр зсуву, виходи якого увімкнуті до входів вихідного регістра, з'єднаного виходами з шиною даних ПЕОМ, тактовий генератор, вихід якого з'єднаний з синхровходом регістра зсуву та входом лічильника імпульсів, вихід якого під'єднаний до синхровходу вихідного регістра та входу тригера "прапора", а його вихід з'єднаний з виходом запиту переривання та через буферний елемент І з шиною даних ПЕОМ, і дешифратор адреси, включений входами до шини адрес ПЕОМ, а першим виходом до входу дозволу вихідного регістра та входу скидання тригера "прапора", а другим виходом до буферного елемента І.

Недоліки найближчого аналога: генератор видає на вихід числа з різною вагою у двійковому представленні.

В основу корисної моделі поставлено задачу створення генератора послідовності n-розрядних двійкових наборів з рівною вагою шляхом введення дешифратора та n мультиплексорів забезпечено генерування послідовності псевдовипадкових двійкових наборів з повним перебором всіх двійкових наборів заданої ваги на виходах пристрою, в результаті чого розширюються функціональні можливості щодо тестування цифрових схем.

Поставлена задача вирішується тим, що у генераторі псевдовипадкових двійкових наборів з рівною вагою, що містить багатоканальний вузол генерації випадкових бітів, кожен канал в якому включає в себе послідовно з'єднані генератор шуму, підсилювач-обмежувач і лічильний тригер, виходи усіх вузлів об'єднані схемою "ВИКЛЮЧНЕ АБО", і канал спряження з ПЕОМ, що включає в себе регістр зсуву, виходи якого увімкнуті до входів вихідного регістра, з'єднаного виходами з шиною даних ПЕОМ, тактовий генератор, вихід якого з'єднаний з синхровходом регістра зсуву та входом лічильника імпульсів, вихід якого під'єднаний до синхровходу вихідного регістра та входу тригера "прапора", а його вихід з'єднаний з виходом запиту переривання та через буферний елемент І з шиною даних ПЕОМ, і дешифратор адреси, включений входами до шини адрес ПЕОМ, а першим виходом до входу дозволу вихідного регістра та входу скидання тригера "прапора", а другим виходом до буферного елемента І, згідно з корисною моделлю, присутні n-1 одноадресних мультиплексорів 1, n D-тригерів 2, дешифратор 3, [Lbn]-адресний мультиплексор 4, причому на адресні входи n-1 одноадресних мультиплексорів 1 підключено 1-…(n-1) розряди виходу дешифратора 3 відповідно, а на інформаційні входи подається значення 1…(n-1)-го D-тригера відповідно та n-ний D-тригер, на вхід 2…n D-тригерів 2 підключені виходи n-1 одноадресних мультиплексорів 1, а на вхід 1-го D-тригера 2 підключений вихід [Lbn]-адресного мультиплексора 4, на n інформаційних входів [Lbn]-адресного мультиплексора 4 підключено n D-тригерів 2, а на адресні входи подається канал випадкових бітів, також він подається й на вхід дешифратора 3.

Суть корисної моделі пояснюється кресленням, де зображена функціональна схема генератора.

На кресленні присутні n-1 одноадресних мультиплексорів 1, n D-тригерів 2, дешифратор 3, [Lbn]-адресний мультиплексор 4, причому на адресні входи n-1 одноадресних мультиплексорів 1 підключено 1…(n-1) розряди виходу дешифратора 3 відповідно, а на інформаційні входи подається значення 1…(n-1)-го D-тригера відповідно та n-ний D-тригер, на вхід 2…n D-тригерів 2 підключені виходи n-1 одноадресних мультиплексорів 1, а на вхід 1-го D-тригера 2 підключений вихід [Lbn]-адресного мультиплексора 4, на n інформаційних входів [Lbn]-адресного мультиплексора 4 підключено n D-тригерів 2, а на адресні входи подається канал випадкових бітів, також він подається й на вхід дешифратора 3.

Пристрій працює у такий спосіб.

Після встановлення початкових значень n D-тригерів 2 у початковий момент часу на вхід дешифратора 3 та [Lbn]-адресного мультиплексора 4 подається число k через канал випадкових бітів у проміжку 1…n у двійковому вигляді (номер елемента, який умовно ділить n D-тригерів 2 на дві частини, у яких відбуваються циклічний зсув). На вхід 1-го D-тригера 2

подається k-ий D-тригер 2 через вихід [Lbn]-адресного мультиплексора 4, на адресні входи n-1 одноадресних мультиплексорів 1 подається сигнал з дешифратора 3, на k-ий D-тригер 2 через (k-i)-ий одноадресний мультиплексор 1 подається n-ний D-тригер, на всі інші входи n D-тригерів 2 (окрім 1-го та k-го) через n-1 одноадресних мультиплексорів 1 подається сигнал з попереднього D-тригера, тобто на і-й подається (і-1)-й. Таким чином, отримуємо дві частини n D-тригерів 2, у яких відбувається циклічний зсув.

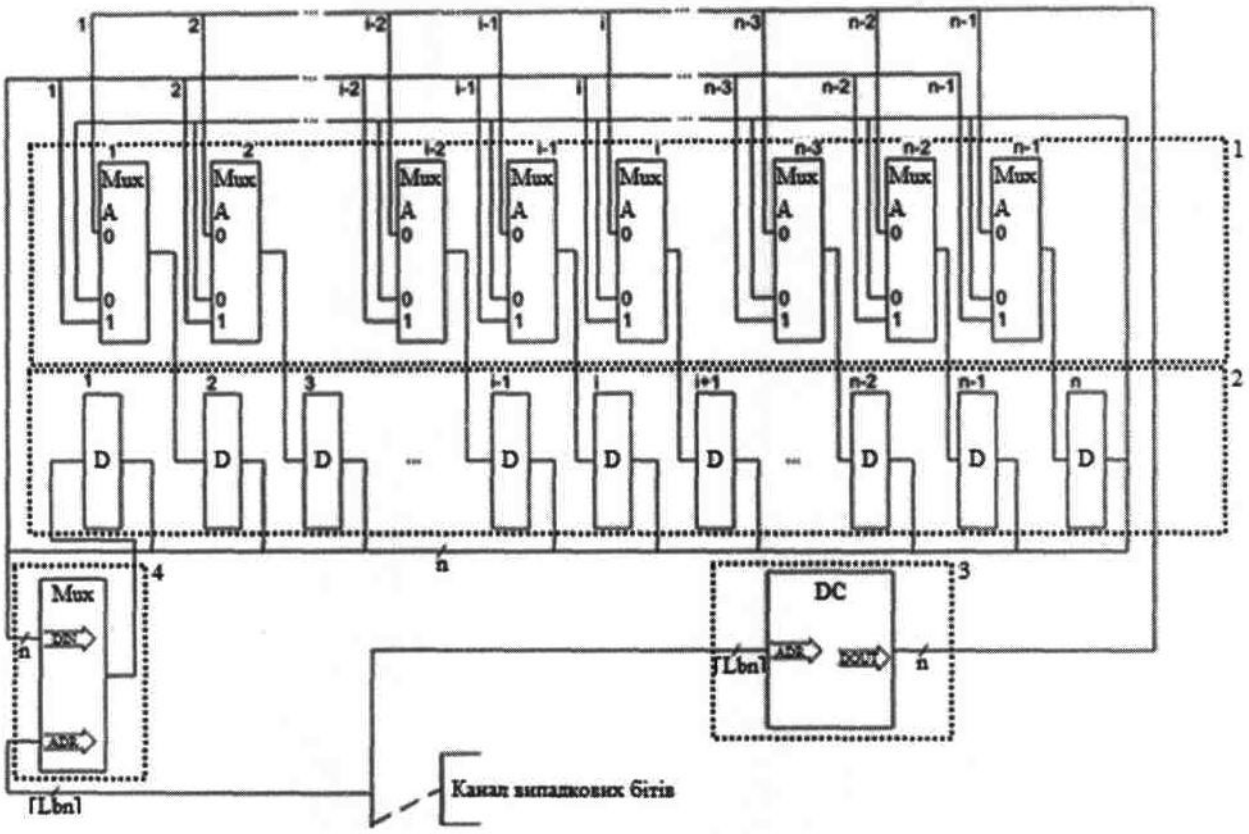
Джерела інформації:

1. Угрюмов Е.П. Цифровая схемотехника: Учеб. пособие для вузов. - СПб.: БХВ-Петербург, 2004. - 800 с. - С. 218, рис. 3.59;

2. UA №33361 МПК G06F 7/58, G07C 15/00. Генератор рівномірно розподілений ймовірних чисел / Горбенко І.Д., Торба О.О., Єлаков С.Г., Степченко О.З., Риженко О.І. - Опубл. Бюл. №1, 2001.

ФОРМУЛА КОРИСНОЇ МОДЕЛІ

Генератор псевдовипадкових двійкових наборів з рівною вагою, що містить багатоканальний вузол генерації випадкових бітів, кожен канал в якому включає в себе послідовно з'єднані генератор шуму, підсилювач-обмежувач і лічильний тригер, виходи усіх вузлів об'єднані схемою "ВИКЛЮЧНЕ АБО", і канал спряження з ПЕОМ, що включає в себе регістр зсуву, виходи якого увімкнуті до входів вихідного регістра, з'єднаного виходами з шиною даних ПЕОМ, тактовий генератор, вихід якого з'єднаний з синхровходом регістра зсуву та входом лічильника імпульсів, вихід якого під'єднаний до синхровходу вихідного регістра та входу тригера "прапора", а його вихід з'єднаний з виходом запиту переривання та через буферний елемент І з шиною даних ПЕОМ, і дешифратор адреси, включений входами до шини адрес ПЕОМ, а першим виходом до входу дозволу вихідного регістра та входу скидання тригера "прапора", а другим виходом до буферного елемента І, який **відрізняється** тим, що присутні n-1 одноадресних мультиплексорів 1, n D-тригерів 2, дешифратор 3, [Lbn]-адресний мультиплексор 4, причому на адресні входи n-1 одноадресних мультиплексорів 1 підключено 1…(n-1) розряди виходу дешифратора 3, відповідно, а на інформаційні входи подається значення 1…(n-1)-го D-тригера відповідно та n-ний D-тригер, на вхід 2…n D-тригерів 2 підключені виходи n-1 одноадресних мультиплексорів 1, а на вхід 1-го D-тригера 2 підключений вихід [Lbn]-адресного мультиплексора 4, на n інформаційних входів [Lbn]-адресного мультиплексора 4 підключено n D-тригерів 2, а на адресні входи подається канал випадкових бітів, також він подається й на вхід дешифратора 3.



Комп'ютерна верстка Г. Паяльніков

Міністерство економічного розвитку і торгівлі України, вул. М. Грушевського, 12/2, м. Київ, 01008, Україна

ДП "Український інститут промислової власності", вул. Глазунова, 1, м. Київ – 42, 01601