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

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

«До захисту допущено»

Завідувач кафедри

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

“___” червня 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки **6.050102 «Комп'ютерна інженерія»**

на тему: Формувач послідовностей псевдовипадкових наборів заданої ваги

Виконав: студент IV курсу, групи КВ-52

Воронцов Павло Станіславович _____

Керівник: професор кафедри СПіСКС,
д.т.н., доцент Романкевич В. О. _____

Консультант з нормоконтролю, доц.каф.СПСКС,
к.т.н. Клятченко Я.М. _____

Рецензент _____

Засвідчую, що у цьому дипломному проекті
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____

Київ – 2019 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 – перший (бакалаврський)

Напрям підготовки 6.050102 «Комп’ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)
«___» червня 2019 р.

ЗАВДАННЯ
на дипломний проект студента
Воронцова Павла Станіславовича
(прізвище, ім’я, по батькові)

1. Тема проекту: Формувач послідовностей псевдовипадкових наборів заданої ваги

керівник проекту професор кафедри СПіСКС, д.т.н., доцент Романкевич В. О.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «22» травня 2019 р. №1330-С

2. Термін подання студентом проекту «13» червня 2019 р.

3. Вихідні дані до проекту: науково-технічна література щодо формувачів псевдовипадкових двійкових наборів з заданою вагою та їх імітації.

4. Зміст пояснювальної записки: аналіз існуючих рішень та обґрунтування актуальності розробки формувачів псевдовипадкових послідовностей; апаратна основа та принцип роботи; програмна реалізація алгоритму формування послідовностей; аналіз роботи; висновки.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

- Структура модулів програми
- Функціональна схема формувача
- Діаграма класів програми
- Алгоритм роботи формувача
- Алгоритм роботи генератора

6. Консультанти розділів проекту[1*]

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М.		

7. Дата видачі завдання

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Видача завдання на дипломне проектування	16.12.2018	
2	Розробка технічного завдання	07.02.2019	
3	Аналіз існуючих рішень	12.03.2019	
4	Розробка програмного продукту	23.03.2019	
5	Відлагодження програмного продукту	30.03.2019	
6	Підготовка пояснювальної записки	14.05.2019	
7	Оформлення матеріалів проекту	19.05.2019	
8	Попередній огляд матеріалів диплому на кафедрі	30.05.2019	

Студент _____ Воронцов П. С.
(підпис) (ініціали, прізвище)

Керівник проекту _____ Романкевич В. О.
(підпис) (ініціали, прізвище)

[1*] Консультантом не може бути зазначено керівника дипломного проекту.

Поз.	Формат	№ прим.	ПОЗНАЧЕННЯ	НАЙМЕНУУВАННЯ	Кільк.	Прим.
				Документація загальна		
				Новозроблена		
	А4		ІАЛЦ. 468900.002 ТЗ	Формувач послідовностей	4	
				псевдовипадкових наборів		
				заданої ваги		
				Технічне завдання		
	А4		ІАЛЦ. 468900.003 ТП	Формувач послідовностей	2	
				псевдовипадкових наборів		
				заданої ваги		
				Відомість технічного		
				проекту		
	А4		ІАЛЦ. 468900.004 ПЗ	Формувач послідовностей	53	
				псевдовипадкових наборів		
				заданої ваги		
				Пояснювальна записка		
	А1		ІАЛЦ. 468900.005 Д1	Структура модулів	1	
				програми		
				Схема структурна		
	А1		ІАЛЦ. 468900.006 Д2	Функціональна схема	1	
				формувача		
				Схема структурна		
	А1		ІАЛЦ. 468900.007 Д3	Діаграма класів	1	
				програми		

Зм.	Лист	№ докум.	Підп.	Дата	*ІАЛЦ. 468900.001 ОА*		
Розробив		Воронцов П.С.			*Формувач послідовностей псевдовипадкових наборів заданої ваги* *Опис альбому*	Лит.	Аркуш 1 / Аркушів 2
Перевірив		Романкевич В.О.					
Н. контр.		Клятченко Я.М.				*НТУУ «КПІ ім. Ігоря Сікорського», ФПМ, КВ-52*	
Затвердив		Тарасенко В.П.					

Поз.	Формат	№ прим.	ПОЗНАЧЕННЯ	НАЙМЕНУУВАННЯ	Кільк.	Прим.
				Схема структурна		
	А1		ІАЛЦ. 468900.008 Д4	Алгоритм роботи	1	
				формувача		
				Блок-схема		
	А1		ІАЛЦ. 468900.009 Д5	Алгоритм роботи	1	
				генератора		
				Блок-схема		
			Диск CD-ROM	Текст ПЗ. Текст	1	
				програми. Графічний		
				матеріал. Презентація.		

ЗМІСТ

					ІАЛЦ. 468900.002 ТЗ		
Зм.	*Лист*	*№ докум.*	*Підп.*	*Дата*			
Розробив		*Воронцов П.С.*			*Формувач послідовностей псевдовипадкових наборів заданої ваги* *Технічне завдання*	*Лит.*	*Аркуш*	*Аркушів*
Перевірив		*Романкевич В.О.*					*1*	*4*
						НТУУ «КПІ ім. Ігоря Сікорського», ФПМ, КВ-52		
Н. контр.		*Клятченко Я.М.*						
Затвердив		*Тарасенко В.П.*						

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Формувач псевдовипадкових двійкових послідовностей з заданою вагою».

Галузь застосування: тестування багатопроцесорних систем, тестування електронних схем на коректність роботи з довільними векторами на вході, генерація псевдовипадкових чисел.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання дипломного проекту першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення програми на основі принципової схеми формувача для генерації псевдовипадкових послідовностей з постійною вагою з можливістю конфігурації генератора.

4 ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5 ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до програмного продукту, що розробляється

- Сумісність з операційними системами Windows, Linux, MacOS
- Можливість конфігурації первинного генератора
- Можливість ініціалізації первинного генератора
- Можливість ініціалізації та конфігурації формувача послідовностей з за-

даною вагою

- Можливість виводу згенерованих послідовностей
- Можливість збереження у файл згенерованих послідовностей
- Можливість статистичного аналізу згенерованих послідовностей за весь період генератора

5.2 Вимоги до апаратного забезпечення

- Процесор: 1,2,4-ядерний, Intel, AMD, ARM;
- Оперативна пам’ять: 4 Гб;

5.3 Вимоги до програмного та апаратного забезпечення користувача

- Інтерпретатор мови Python
- Бібліотеки для мови Python: Pandas, NumPy

6 ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	15.04.2019
2.	Розроблення та узгодження технічного завдання	30.04.2019
3.	Аналіз існуючих рішень	12.03.2019
4.	Підготовка матеріалів першого розділу дипломного проекту	10.05.2019
5.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2019
6.	Підготовка графічної частини дипломного проекту	20.05.2019
7.	Оформлення документації дипломного проекту	19.05.2019
8.	Попередній огляд матеріалів диплому на кафедрі	30.05.2019

Поз.	Формат	№ прим.	ПОЗНАЧЕННЯ	НАЙМЕНУУВАННЯ	Кільк.	Прим.
				Документація загальна		
				Новозроблена		
	А4		ІАЛЦ. 468900.004 ПЗ	Формувач послідовностей	53	
				псевдовипадкових наборів		
				заданої ваги		
				Пояснювальна записка		
	А1		ІАЛЦ. 468900.005 Д1	Структура модулів	1	
				програми		
				Схема структурна		
	А1		ІАЛЦ. 468900.006 Д2	Функціональна схема	1	
				формувача		
				Схема структурна		
	А1		ІАЛЦ. 468900.007 Д3	Діаграма класів	1	
				програми		
				Схема структурна		
	А1		ІАЛЦ. 468900.008 Д4	Алгоритм роботи	1	
				формувача		
				Блок-схема		
	А1		ІАЛЦ. 468900.009 Д5	Алгоритм роботи	1	
				генератора		
				Блок-схема		
			Диск CD-ROM	Текст ПЗ. Текст	1	

Зм.	*Лист*	*№ докум.*	*Підп.*	*Дата*	*ІАЛЦ. 468900.003 ТП*
Розробив		*Воронцов П.С.*			*Формувач послідовностей псевдовипадкових наборів заданої ваги* *Технічний проект*
Перевірив		*Романкевич В.О.*			
Н. контр.		*Клятченко Я.М.*			
Затвердив		*Тарасенко В.П.*			

Лит.	*Аркуш*	*Аркушів*
	1	*2*

НТУУ «КПІ ім. Ігоря Сікорського», ФПМ, КВ-52

Поз.	Формат	№ прим.	ПОЗНАЧЕННЯ	НАЙМЕНУУВАННЯ	Кільк.	Прим.
				програми. Графічний		
				матеріал. Презентація.		

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (53 с., 30 рис. 17 табл.).

Об’єкт розробки – створення комп’ютерної програми, відтворюючої роботу формувача псевдовипадкових двійкових послідовностей з заданою вагою на базі регістру зсуву з лінійним зворотним зв’язком.

Комп’ютерна програма дозволяє: перевіряти ініціалізуючі параметри, задавати поліном та початновий стан для первинного генератора псевдовипадкових двійкових послідовностей, ініціалізувати генератор псевдовипадкових двійкових послідовностей з постійною вагою, конфігурувати обидва генератори, генерувати псевдовипадкові послідовності з постійною вагою та без, зберігати згенеровані послідовності, зберігати, аналізувати та виводити статистичні дані після генерації усіх можливих послідовностей.

В ході розробки:

- створено програму на базі алгоритму роботи принципової схеми генератора;
- проведено аналіз методів побудови генераторів псевдовипадкових двійкових послідовностей;
- проведено аналіз можливих модифікацій базового генератора для генерації послідовностей з постійною вагою;
- розроблена принципова схема генератора псевдовипадкових послідовностей з постійною вагою;
- створено алгоритм моделювання роботи принципової схеми;
- розроблено застосунок для роботи з генератором;

Ключові слова: ГЕНЕРАТОР ПСЕВДОВИПАДКОВИХ ДВІЙКОВИХ ПОСЛІДОВНОСТЕЙ, ДВІЙКОВІ ПОСЛІДОВНОСТІ З ПОСТІЙНОЮ ВАГОЮ, РЕГІСТР ЗСУВУ З ЛІНІЙНИМ ЗВОРОТНИМ ЗВ’ЯЗКОМ.

ABSTRACT

The qualifying work includes an explanatory note (53 p., 30 pic. 17 tables.).

The object of development - the creation of a computer program reproducing the work of the scheme of a pseudorandom binary sequence generator with a given weight based on a shift register with linear feedback.

The computer program allows you to: validate initializing parameters, initialize the primary pseudorandom binary generator, initialize the pseudorandom binary sequence generator with constant weight, configure both generators, generate pseudorandom sequences with constant weight and without, store generated sequences, output statistics after generation of all possible sequences.

During development:

- a custom application developed on the basis of the algorithm of the principal circuit for working with the generator;
- An analysis of methods for constructing generators of pseudorandom binary sequences is carried out;
- the analysis of the progress of the basic generator for fulfilling the requirements for the generation of sequences with constant weight;
- the principal scheme of the generator of pseudorandom sequences of constant weight is developed;
- the algorithm is developed on the basis of the principle scheme;
- a custom application developed on the basis of the algorithm of the principal circuit for working with the generator;

Keywords: GENERATOR PSEUDORANDOM BINARY SEQUENCE, BINARY SEQUENCES WITH CONSTANT WEIGHT, LINEAR FEEDBACK SHIFT REGISTER.

ЗМІСТ

					ІАЛЦ. 468900.004 ПЗ	
Зм.	*Лист*	*№ докум.*	*Підп.*	*Дата*		
Розробив		*Воронцов П.С.*			*Формувач послідовностей псевдовипадкових наборів заданої ваги Пояснювальна записка*	*Лит.* | *Аркуш* | *Аркушів*
Перевірив		*Романкевич В.О.*				*1* | *53*
Н. контр.		*Клятченко Я.М.*				*НТУУ «КПІ ім. Ігоря Сікорського», ФПМ, КВ-52*
Затвердив		*Тарасенко В.П.*				

Додатки

Додаток 1. Копії графічних матеріалів;

ІАЛЦ. 468900.005 Д1 Структура модулів програми

ІАЛЦ. 468900.006 Д2 Функціональна схема формувача

ІАЛЦ. 468900.007 Д3 Принципова схема формувача

ІАЛЦ. 468900.008 Д4 Алгоритм роботи формувача

ІАЛЦ. 468900.009 Д5 Алгоритм роботи генератора

Додаток 2. Фрагменти програмного коду

Додаток 3. Презентація

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ГПЧП — генераторів псевдовипадкових числових послідовностей

РЗЛЗЗ — регістр зсуву з лінійним зворотним зв’язком

Функціональна схема — схема, що роз’яснює процеси, що відбуваються у функціональних частинах пристрою чи у виробі.

Імітаційна модель — опис об’єкта за допомогою математичних та логічних методів для налагодження, аналізу та проектування об’єкта.

Псевдовипадкова послідовність або вектор — число, отримане за детермінованим алгоритмом, яке має властивості випадкового числа

Регістр — послідовний або паралельний пристрій для зберігання, приймання та передавання інформації у вигляді числа, поданого набором біт на виходах регістру.

Діаграма розсіювання — графік у декартовій системі координат для відображення значень по двом точкам.

ВСТУП

В наш час обчислювальні системи стають все складнішими, вони складаються з багатопроцесорних модулів та підсистем. Окрім засобів для розробки, моніторингу та підтримки цих систем, галузь потребує засобів для налагодження та тестування роботи складних систем.

Одним з методів тестування систем є перевірка на працездатність та поведінку системи за умов виведення з ладу декількох елементів цієї системи. Для того щоб цей процес був контрольованим та відтворюваним потрібні програмні або апаратні засоби які б дозволяли з прогнозованою ймовірністю визначати вузли для відключення з можливістю відтворення цього стану для аналізу поведінки системи на кожному етапі тестування.

Загальний підхід до такого тестування незмінний як в апаратній розробці, так і у програмній. Одним з головних кроків цього тестування є визначення послідовності вузлів, які підлягають виключенню. У випадку великих систем, коли потрібно визначити мінімальну робочу кількість вузлів, генерацію послідовностей можна оптимізувати, якщо використовувати формувач послідовностей з постійною вагою, це дозволить встановлювати кількість вузлів для відключення та визначати мінімально робочу конфігурацію. Також завдяки розподілу цих послідовностей можна визначити вузли відключення яких більш суттєве для системи.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ АКТУАЛЬНОСТІ РОЗРОБКИ ФОРМУВАЧІВ ПСЕВДОВИПАДКОВИХ ПОСЛІДОВНОСТЕЙ

1.1 Сфери застосування

З розвитком апаратних можливостей та ступенів інтеграції елементної бази сучасних високопродуктивних комп'ютерних систем, зростають вимоги до правдивості дискретних систем та пристроїв та оптимізації процесів тестування шляхом збільшення продуктивності та зниження вартості процедур оцінки технічного стану систем що проектуються та розроблюються.

Використання методів для контролю якості на етапі розробки дозволяють завчасно виявити несправності та впевнитись у правильності функціонування цифрового пристрою. Серед методів для оцінки технічного стану схем високу розповсюдженість мають ймовірнісні методи в контрольно-діагностичній апаратурі. В роботах [5, 12] описані методи для тестування відмовостійких систем на етапі проектування за допомогою GL-моделей.

Одним з варіантів ймовірнісних методів є метод псевдовипадкового тестування який верифікується сигнатурним аналізом отриманих результуючих векторів з перевіряємих пристроїв. Цей підхід використовується для побудови самотестуємих систем, ВІС, НВІС та системах контролю побудованих на ймовірнісних принципах. В загальному випадку цей метод полягає в тому що на тестуємий пристрій подається псевдовипадкові набори вхідних векторів, після чого вони аналізуються за допомогою сигнатурного аналізу на наявність помилок та необроблених варіантів вхідних станів. [13, 15]

Окрім використання у тестуванні, псевдовипадкові послідовності активно використовуються у криптографії, кодуванні інформації, моделюванні систем. Генерація потокових шифрів у криптографії основана на операції XOR, гамою для якої береться згенерована псевдовипадкова послідовність. В кодуванні інформації псевдовипадкові послідовності використовуються для збільшення завадостій-

кості закодованих повідомлень. При моделюванні систем та їх поведінки методом Монте-Карло потребуються послідовності випадкових або псевдовипадкових чисел, які беруться з псевдовипадкових послідовностей.

Розвиток апаратних та програмних можливостей обчислювальних пристроїв дає можливість для удосконалення наявний та побудови нових рішень для вирішення задачі генерації псевдовипадкових послідовностей. Для генерації псевдовипадкових послідовностей виокремлюють декілька основних методів: лінійний конгруентний метод, запізнілі генератори Фібоначчі, методи на основі регістру зсуву з лінійним зворотним зв'язком [1–4]. Загальні теоретичні положення та принципи про генерацію псевдовипадкових послідовностей за допомогою зсувних регістрових структур були описані в 60-70 роках минулого сторіччя в роботах таких авторів, як А. Гілл, Э. Селмер, а вдосконалені та досліджені в майбутньому Ярмоликом В.Н., Яковлевим В.В., Добрисом Г.В.

1.2 Генератор на базі регістру зсуву зі зворотним лінійним зв'язком

В загальному випадку генерації псевдовипадкових послідовностей можна виділити такі критерії, як: довжина періоду генератора, ефективність, відтворюваність та однакове функціонування на різних системах.

- Довжина періоду визначає кількість сформованих послідовностей після яких генератор буде повторюватись. Чим більший період генератора тим краще. У випадку генератора на основі регістру зсуву з лінійним зворотним зв'язком максимальним періодом буде 2^n, де n - довжина послідовності;
- Ефективність генератора визначається його швидкодією та затратністю по пам'яті;
- Відтворюваність визначається можливістю генерації однакових послідовностей на основі однакових ініціалізуючих параметрів;

- Однакове функціонування на різних системах дозволяє легко переносити реалізацію генератора на різні платформи та відтворення результатів незалежно від платформи;

Загальна схема генерації послідовностей генератора з лінійним зворотним зв'язком можна визначити як сумування за модулем 2 визначених бітів з послідовності, тому за допомогою формули многочлена (1) можна описати генерацію наступного біту послідовності.

$$X = a_1 \oplus a_2 \oplus ... \oplus a_n \tag{1}$$

Принципова схема генератора на рисунку 1 складається з регістру зсуву та суматора (або суматорів) за модулем 2, операндами якого є виходи визначених комірок пам'яті регістру.

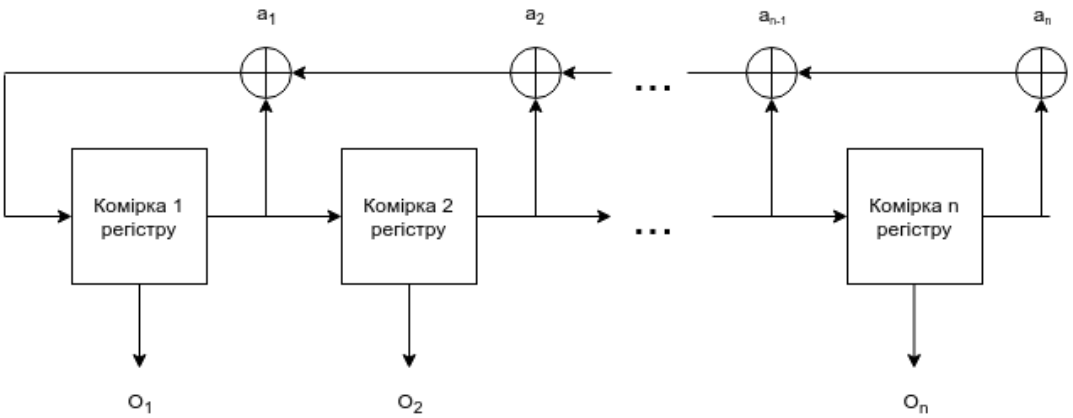

Рисунок 1 – Регістр зсуву з лінійним зворотним зв'язком

Ця схема конфігурується суматорами за модулем 2 та їх з'єднанням, що визначені поліномом. Алгоритм роботи цієї схеми складається з трьох частин: операція зсуву, додаванням операндів усіх з'єднаних суматорів, запис отриманого результату у старший біт регістру. Прикладом роботи генератора довжиною 4 біти та поліномом $x^4 + x^3 + 1$ за цим алгоритмом може бути таблиця 1. На кожному кроці роботи генератора виконується додавання за модулем 2 значень з бітів під номерами 3 та 4 (нумерація з 1) та значенням 1, після чого виконується зсув регістру

вправо на 1 біт, та запис у початок регістру значення отриманого при додаванні.

Таблиця 1 – Результат роботи регістру зсуву з лінійним зворотнім зв’язком

i	1	2	3	4
0	1	0	0	0
1	1	1	0	0
2	1	1	1	0
3	0	1	1	1
4	1	0	1	1
5	1	1	0	1
6	0	1	1	0
7	0	0	1	1
8	1	0	0	1
9	0	1	0	0
10	1	0	1	0
11	0	1	0	1
12	0	0	1	0
13	0	0	0	1
14	0	0	0	0
15	1	0	0	0

Для більшості розмірностей РЗЛЗЗ визначені таблиці цих многочленів, які дозволяють отримати максимальний період для генерації, прикладом може слугувати таблиця 2 в якій наведені деякі розмірності послідовностей та їхні поліноми для отримання максимального періоду генератора.

Також для однакової довжини послідовності може бути декілька таких многочленів. Для більшості генераторів на основі РЗЛЗЗ характерні такі переваги як:

- висока швидкодія;
- використання простих логічних операцій;
- хороші статистичні властивості;
- легкість аналізу алгебраїчними методами;

Але незважаючи на це при програмній реалізації цих генераторів швидкодія падає внаслідок використання біль щільних многочленів для запобігання злому та

Таблиця 2 – Поліноми для регістру зсуву з лінійним зворотнім зв'язком

Біти, n	Поліном	Період, $2^n - 1$	Кількість примітивних поліномів
2	$x^2 + x + 1$	3	1
3	$x^3 + x^2 + 1$	7	2
4	$x^4 + x^3 + 1$	15	2
8	$x^8 + x^6 + x^5 + x^4 + 1$	255	16
10	$x^{10} + x^7 + 1$	1023	60
12	$x^{12} + x^{11} + x^{10} + x^4 + 1$	4095	144
14	$x^{14} + x^{13} + x^{12} + x^2 + 1$	16383	756
16	$x^{16} + x^{14} + x^{13} + x^{11} + 1$	65535	2048
18	$x^{18} + x^{11} + 1$	262143	7776

аналізу. Хоча у випадку коли генератор не використовується у криптографічних цілях, можна використовувати більш розріджені многочлени.

У випадку генераторів з постійною вагою виникають додаткові обмеження, через те, що кількість можливих послідовностей зменшується, треба зберегти максимально можливий рівномірний розподіл за умови зменшення кількості можливих послідовностей.

Коли використовується звичайний генератор, на базі РЗЛЗЗ, ми маємо однакову ймовірність появи кожної послідовності, а при обмеженні на кількість одиниць в послідовності ймовірність змінюється в меншу сторону через те що набори починають повторюватись у випадку перестановки місцями однакових значень в бітах, наприклад $1 \to 1$ або $0 \to 0$.

Також виникає проблема зі збільшенням затримки при послідовностях з великою відстанню між одиницями, яка може бути вирішена у представленій реалізації формувача за допомогою додавання схеми перенесення у схему формувача. Перенесення у цій схемі дозволяє не пускати в дію елементи регістру у випадку наявності 0 на виході первинного генератора.

Загальна функціональна схема для генераторів, що забезпечують фіксовану вагу, зображена на рисунку 2, та описана в роботах [4,6]. Вона складається з РЗЛЗЗ, який може бути звичайним генератором псевдовипадкових двійкових наборів, ве-

ктори з якого передаються на комутаційну логічну схему, яка у свою чергу займається встановленням бітів у вихідному регістрі базуючись на поточних значеннях бітів вихідного регістру та вектору з РЗЛЗЗ.

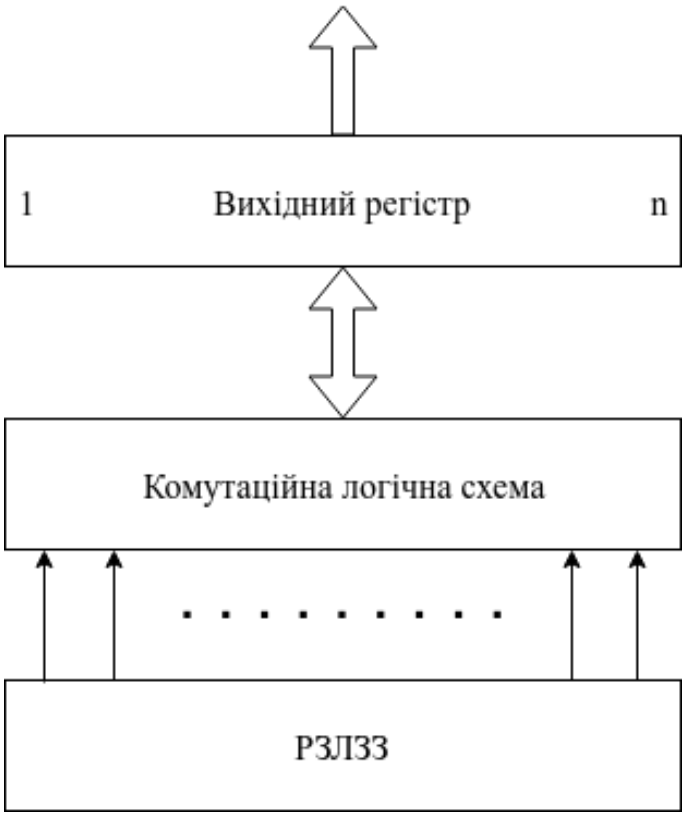

Рисунок 2 – Функціональна схема генератора

Модифікація комутаційної схеми, реалізована у даному дипломному проекті має на меті підвищення швидкодії генерації послідовностей завдяки вставки у вихідну послідовність з регістру первинного генератора додаткової одиниці, базуючись на довгій послідовності нулів. Саме ці послідовності нулів, викликають довготривалі періоди ітерацій формувача, коли він обмінює місцями однакові біти, чим провокує генерацію однакових послідовностей. Якщо позбавитись цього недоліку то формуємі послідовності будуть генеруватись швидше та самі послідовності будуть більш розрізненими.

Незважаючи на підвищення швидкодії, дана конфігурація потребує детального аналізу та підбору для збереження статистичних властивостей послідовностей та їх можливої кількості.

2 АПАРАТНА ОСНОВА ТА ПРИНЦИП РОБОТИ

2.1 Аналіз базової схеми формувача наборів заданої ваги

Оскільки звичайний генератор псевдовипадкових послідовностей не дає змоги отримати виключно набори заданої ваги, тому потрібен формувач цих послідовностей, який буде використовувати набори зі звичайного генератора, як основу для формування послідовностей фіксованої ваги. Для того щоб досягти однакової ваги у послідовності використовується пересування одиничних бітів по розрядах вихідного регістру формувача в залежності від вихідного набору на первинному генераторі, який задає ”активні”комірки формувача, серед яких потрібно провести зсув. Сам регістр формувача являє собою кільцевий регістр на N біт, які визначають довжину послідовності, з модифікованою схемою зсуву, в якій додана керуюча схема перенесення. Принципова схема однієї комірки вихідного регістру формувача відображена на рисунку 3.

Вона складається з 5 логічних елементів, серед яких:

- 3 елементи І;
- 1 елемент АБО;
- 1 D-тригер;

2.2 Алгоритм роботи базової схеми

Для початку роботи формувача він має бути ініціалізован послідовністю, яка буде змінюватись під час роботи. У апаратній реалізації це забезпечується встановленням бітів у 1 або 0 у регістрі формувача за допомогою подання ініціалізуючої послідовності на інформаційні входи регістру та двійковому набору, на управляючі входи, що переводить регістр у режим запису. Після ініціалізації встановлюються значення переносів між елементами регістру.

Під час роботи формувача поточна комірка є активною, якщо на і-му біті первинного генератора встановлена логічна ”1 інакше вона не змінює і-й біт фор-

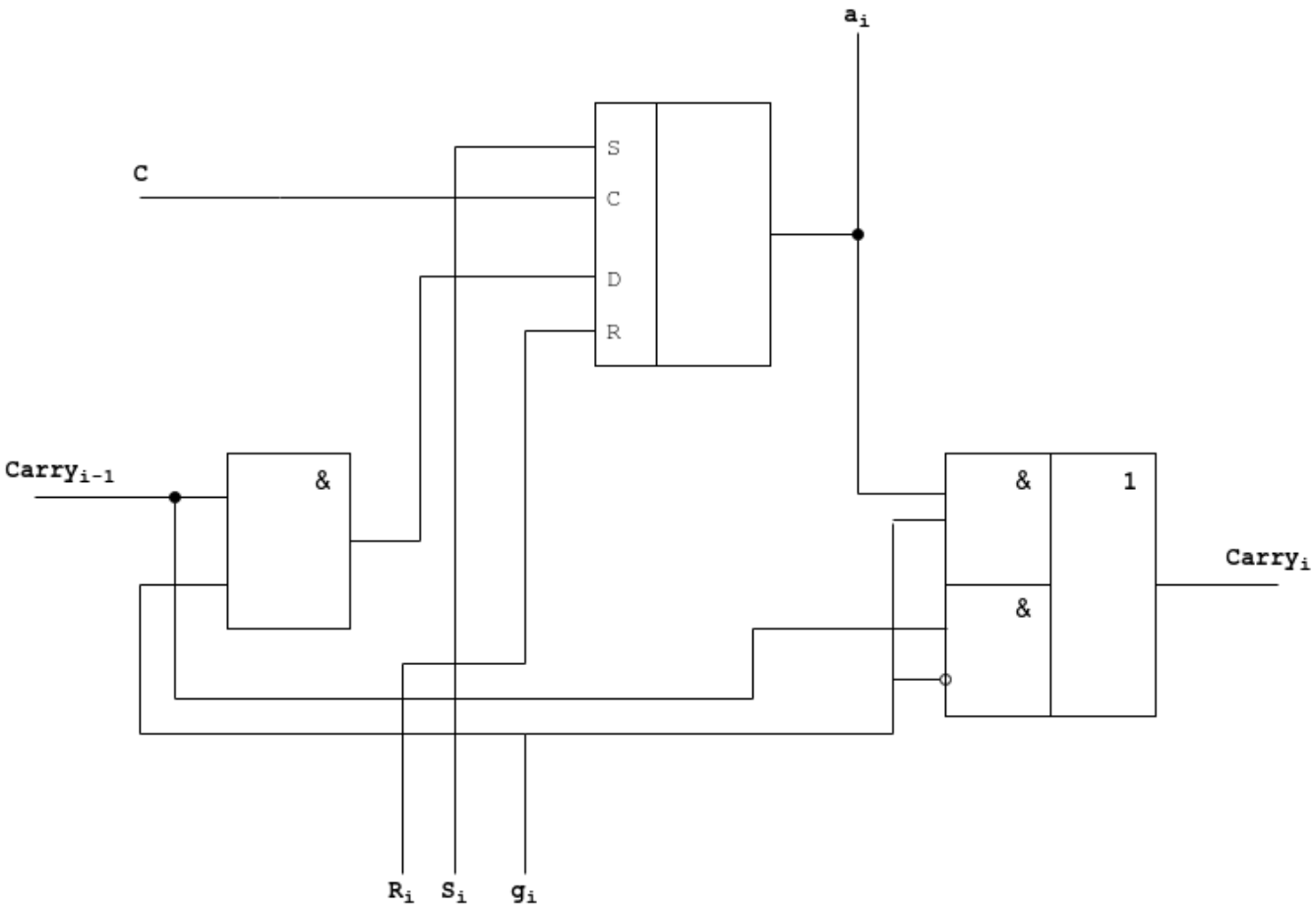

Рисунок 3 – Комірка пам’яті зі схемою переносу

мувача та тільки передає сигнал переносу з попереднього елемента на наступний. При надходженні синхросигналу значення переносу на наступний елемент пам’яті встановлюється базуючись на поточному значенні на виході D-тригеру, значенню переносу з попереднього елементу та стану і-го біту на первинному генераторі.

Якщо поточна комірка формувача активна, то передається значення з виходу поточної комірки, в протилежному випадку - значення зсуву з попередньої комірки. Також відбувається встановлення нового значення на виході комірки на основі значення переносу $Carry_{i-1}$ з попереднього елементу та значення і-го біту первинного генератору з g_i. Якщо наявний перенос з попереднього елементу та на первинному генератору поточний біт дорівнює ”1 то на виході D-тригера встановлюється рівень логічної ”1”. Таблиція переходів для однієї комірки представлена у табл. 3

На рівні однієї комірки все здається оптимізованим, але на рівні всього формувача можна побачити, що при вхідних наборах з первинного генератора в яких

Таблиця 3 – Таблиця переходів

C	g_i	a_i	$Carry_i$
0	g_i	a_i	$Carry_i$
1	0	a_i	$Carry_{i-1}$
1	1	$Carry_{i-1i}$	a_i

відстань між сусідніми одиницями близька до $\frac{N}{2}$, де N - розмірність регістру, значно зростає час на перенесення бітів у вихідному формувачі внаслідок перебору багатьох комірок лише для переносу.

2.3 Оптимізація вхідних наборів

Виявивши недолік в наборах з первинного генератора ми можемо його виправити за допомогою ущільнення тих вихідних векторів, в яких спостерігається довга послідовність з нулів. Оскільки впливати на роботу первинного генератора ми не можемо, тоді можна додати проміжний регістр зсуву з керуючою схемою, між первинним генератором та формувачем, який при наявності послідовності з M нулів в кінці або початку вихідного вектора первинного генератора буде записувати собі на вхід 1 при зсуві. Схема включення проміжного регістру зсуву з керуючою схемою наведено на рисунку 4, в якості керуючої схеми використовується зв'язка з двох логічних елементів АБО-НІ(D1) на M входів та АБО(D2). При наявності M нулів у кінці вектору з вихідного регістра первинного генератора на виході D1 ми отримуємо логічну 1, при наявності хоча б одної 1 в останніх M бітах первинного вектора ми отримуємо 0. Логічний елемент D2 дозволяє додавати до проміжного генератора кожну одиницю, котру ми можемо отримати або з останнього біту первинного вектора, або D1. Тим самим не змінюючи роботу первинного генератора ми змінили послідовність яка буде подана на вход формувача.

Дана схема може бути виражена у вигляді формули (2), де обчислюєме значення Y відповідатиме значенню біта який буде записано у проміжний регістр схеми. У формулі використано такі змнінні як out, яка є значеннями вихідного регістра

генератора, N - розмірність генератора та M - кількість біт, які перевіряються на нульову послідовність з кінця.

$$Y = \overline{\bigvee_{i=N-M}^{N} out_i} \bigvee out_N \quad (2)$$

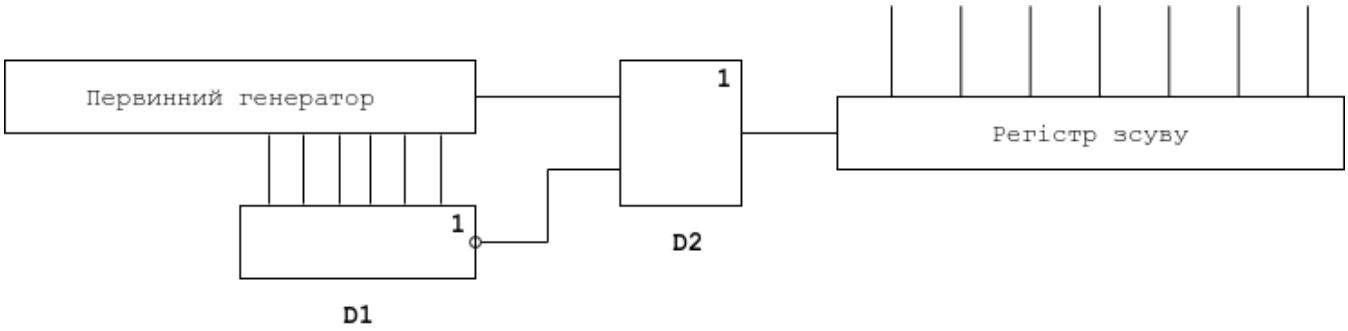

Рисунок 4 – Проміжний регістр з керуючою схемою

Для формувача послідовностей з постійною вагою первинна послідовність втратила у статистичних характеристиках, деякі стани почали повторюватись, але головний недолік, який непомітний на наборах малої довжини, був виправлений. Нас цікавлять саме набори великої довжини, бо на них більше послідовностей з розрідженими одиницями та ця розрідженість суттєвіша, а враховуючи те, що ці набори використовуються у системах для тестуванні багатопроцесорних систем, саме набори великої розрядності мають суттєве значення в роботі схеми.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ ФОРМУВАННЯ ПОСЛІДОВНОСТЕЙ

3.1 Інструменти для розробки

3.1.1 Мова програмування Python та середа Jupyter

Для розробки програмного модуля, який буде імітувати роботу схеми було обрано мову Python, яка протягом останніх років використовується для побудови програмних рішень для симуляції та моделювання процесів, обробки статистичних даних.

Апаратна реалізація може бути обмежена розмірністю псевдовипадкових векторів розмірністю регістрів зсуву та елементної бази, а зміни в розміностях та зв’язків між елементами вимагають переробки схеми, яка тягне за собою більші фінансові та часові затрати за рахунок додаткової складності та необхідності в нових компонентах. В цей же час програмна реалізація позбавлена цього недоліку завдяки використанню динамічних масивів великої довжини та спеціальних структур даних. Також програмна реалізація дозволяє швидко змінювати конфігурації первинного генератора псевдовипадкових послідовностей, формувача послідовностей заданої ваги та схеми управління проміжного регістру, чим забезпечує можливість генерувати довільні набори з різними характеристиками, на відміну від апаратної.

Для інтерактивної розробки прототипу та перевірки ефективності роботи формувача було використано веб додаток Jupyter Notebook, який створює середовище для виконання коду частинами з можливістю зберігати стани кожного фрагменту та виконувати їх базуючись на попередніх результатах. Веб орієнтованість цієї середи дозволяє будувати графіки на основі даних та додавати текстові вставки з можливістю оформити сторінку з поясненням кожного фрагменту обробки та аналізу даних та результатами роботи цього фрагменту, як на рисунку 5.

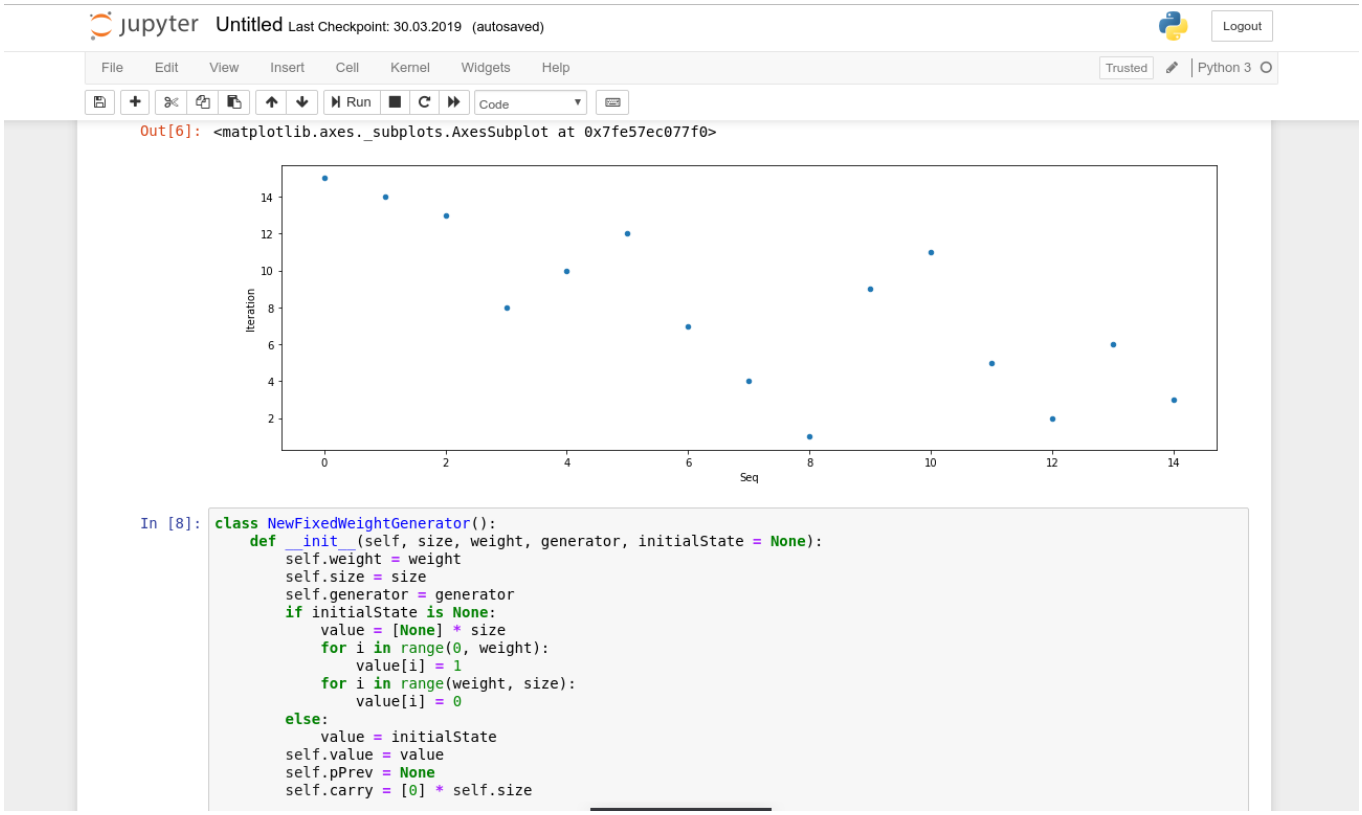

Рисунок 5 – Наочність роботи

3.1.2 Бібліотека Pandas

Для збереження усього набору сформованого генератором використовується бібліотека Pandas. Ця бібліотека розроблена для роботи з даними, побудови графіків, аналізу даних та моделювання. Вона надає спеціальні структури даних та операції над ними для роботи з числовими таблицями та часовими рядами.

При використанні цієї бібліотеки ми оперуємо DataFrame, надає змогу маніпулювати індексованими двомірними масивами даних. Також Pandas надає таку структуру, як Series, яка визначає одномірний масив по одній колонці зі збереженням індексації. [18]

В дипломному проекті вона використана для зберігання великої кількості наборів, побудови графіків розподілу значень у наборі, аналізу статистичних властивостей наборів. Завдяки вбудованим функціям ця бібліотека значно зменшує час на первинний аналіз даних, їхнє очищення та підготовку. Вибір цієї бібліотеки пов’язаний з тим що, при генерації послідовностей великих розмірів, для перевір-

ки їх статистичних влативостей потребується значні об’єми пам’яті та процессорні обчислення, які можуть бути розпараллелені завдяки цій бібліотеці, що збільшує ефективність аналізу отриманого рішення.

3.1.3 Бібліотека NumPy

Бібліотека для роботи з багатомірними масивами, яка оптимізована внаслідок використання компілюємих мов програмування, на відміну від інтерпритуємого Python. Будь-який алгоритм, яки може бути записаний у вигляді послідовності операцій над матрицями та векторами та реалізований за допомогою NumPy за швидкодією буде відповідати аналогічному алгоритму в середовищі MATLAB. [19]

Необхідність цієї бібліотеки обумовлена використанням бібліотеки Pandas та роботи з чисельними послідовностями, які можуть бути розбиті на окремі біти кожна, щоб мати можливість комфортно працювати не лише з представленням послідовності як числа, але й побітово. Зважаючи на те що ця бібліотека орієнтована саме на роботу з числовими структурами, вона значно підвищує швидкодію обробки послідовностей.

3.1.4 Бібліотека Matplotlib

Бібліотека на мові Python для візуалізації даних за допомогою 2D та 3D графіки. Згенеровані зображення готові для використання у наукових публікаціях, користувацьких інтерфейсах, веб додатках, де потрібна візуалізація діаграм. Здебільшого згенеровані зображення відповідають зображенням в MATLAB.

Matplotlib легко налаштовується та конфігурується, тому разом з NumPy та IPython надає можливості для обробки даних, подібні до MATLAB.

Бібліотека підтримує багато видів графіків та діаграм [16, 17]:

- Графіки (line plot);
- Діаграми розкиду (scatter plot);

- Столбчаті діаграми (bar chart)
- Гістограми (histogram);
- Кругові діаграми (pie chart);
- Ствол-лист діаграми (stem plot);
- Контурні графіки (contour plot);
- Поля градієнтів (quiver);
- Спектральні діаграми (spectrogram);

Користувач може вказати осі координат, решітку, добавити надписи та пояснення, використовувати логарифмічну шкалу або полярні координати.

Ця бібліотека була вибрана для того щоб мати можливість надавати результати аналізу не лише у вигляді таблиць та значень, а й у вигляді графіків, що дозволяє одразу оцінювати ефективність обраних конфігурацій, без необхідності аналізувати результати в таблицях.

Приклади графіків, які дозволяє побудувати Matplotlib наведені на рисунках 6 - 8.

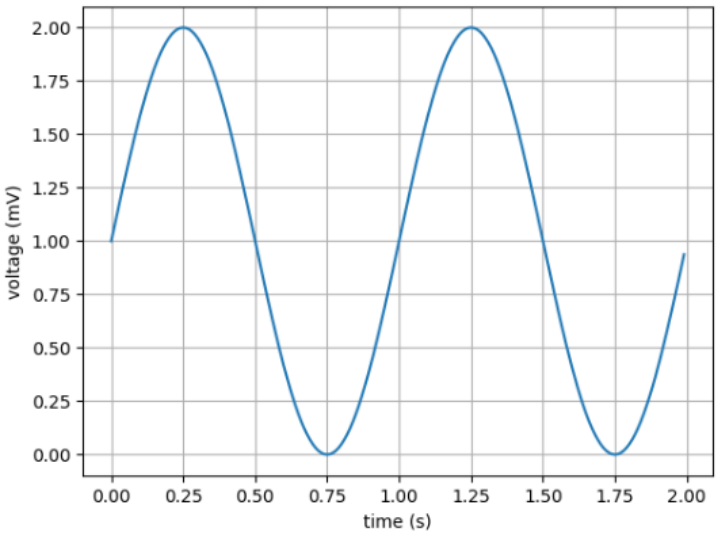

Рисунок 6 – Простий графік

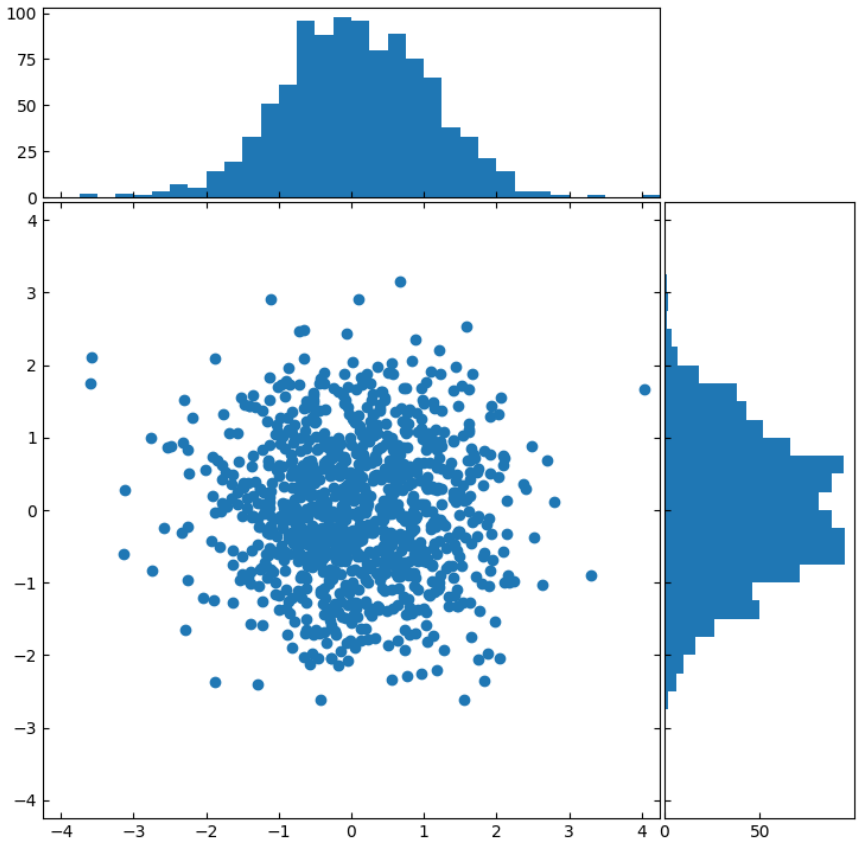

Рисунок 7 – Складний графік

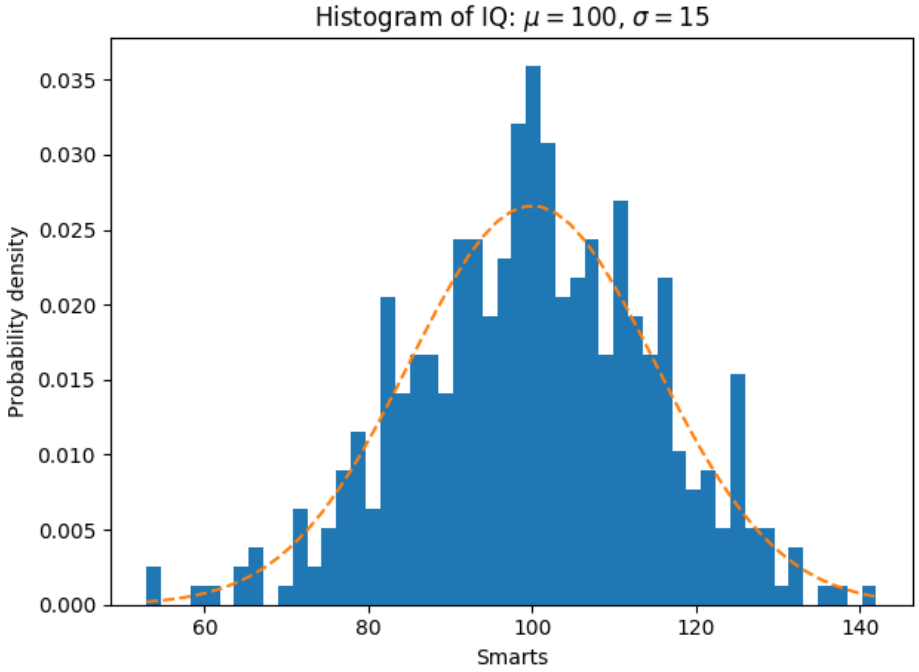

Рисунок 8 – Гістограма

3.2 Модулі програми

Зважаючи на те що програма має багато функціональних можливостей та складність через необхідність імітації схеми та роботи з вхідними параметрами та результатами, вона розбита на модулі, більщість яких виконані у вигляді класів для забезпечення інкапсуляції логіки та поліморфізму компонентів пов’язаних зі схемою та компонентів забезпечуючих підготовку складових схеми.

Програма складається з 7 основних модулів та модуля допоміжних методів. Головним модулем є модуль управління програмою, який ініціалізує користувацький інтерфейс та структури даних для роботи генератора. Точкою входу у програму для користувача є модуль користувацького інтерфейсу, який виконаний у вигляді інтерфейсу командного рядка. Після заданя режиму роботи, конфігурації первинного генератора, проміжної керуючої схеми та формувача, модуль керування створює моделі цих частин та починає виконання алгоритму. Під час виконання кожна отримана послідовність зберігається для подальшого аналізу. Після генерації усіх можливих послідовностей при заданій конфігурації, вони зберігаються у файл за допомогою модуля зберігання наборів. Поверхнева структура модулів зазначена на рисунку 9

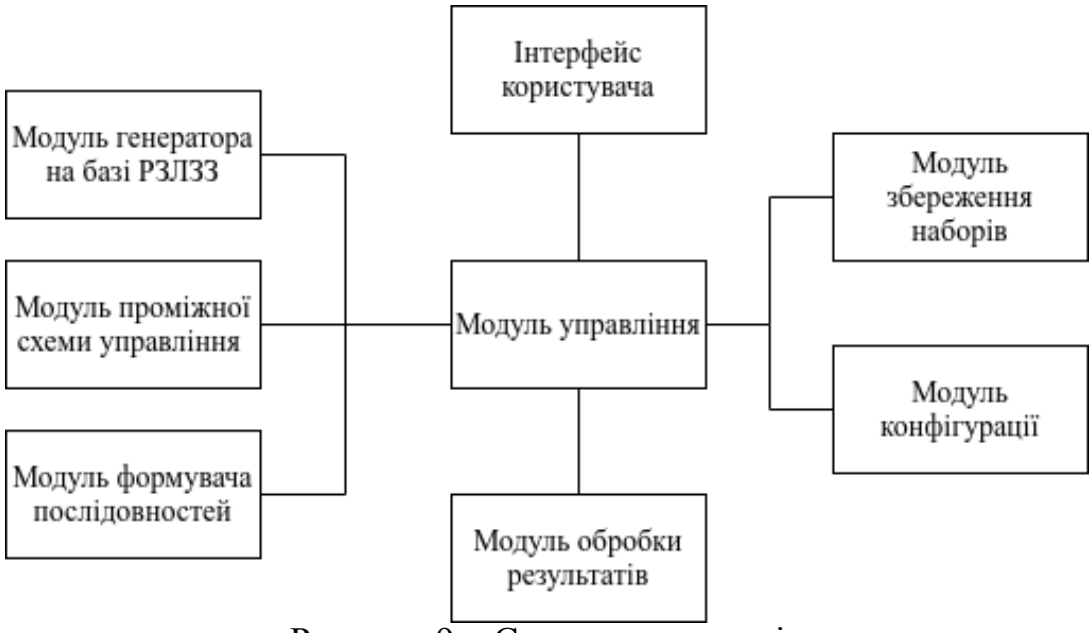

Рисунок 9 – Структура модулів

Зважаючи на те що була використана мова Python, яка дозволяє писати в ООП стилі, для кращого розуміння програми були розроблені класи, які являють собою реалізації складових системи. Діаграма класів зображена на рисунку 10.

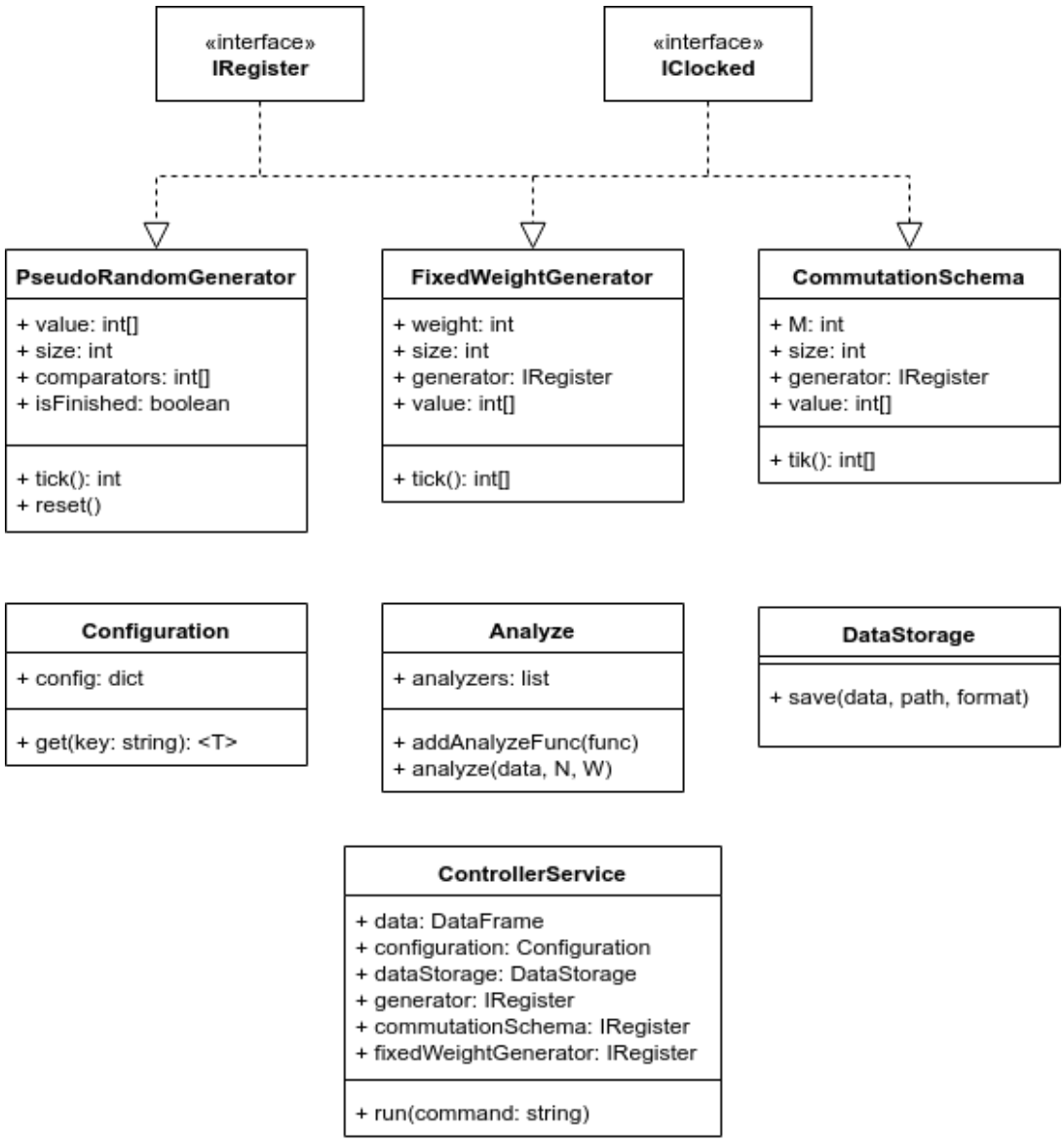

Рисунок 10 – Діаграма класів

Класи що реалізують логіку генератора, формувача та комутаційної схеми реалізують інтерфейси IRegister та IClocked. Перший з яких вимагає наявності реалізації регістру зсуву в елементі, а другий визначає метод tick в елементі.

Клас ControllerService є управляючим, який зберігає у собі посилання на складові схеми, класи для роботи з конфігурацією, збереження даних та аналізу. Також цей клас реалізує функцію тактування складових схеми та збереження про-

міжного результату генерації послідовностей.

Клас Configuration відповідає за взаємодію з конфігурацією, він надає метод для отримання параметра конфігурації за ключем та завантаженню конфігурації з файлу. У випадку виникнення виключення, воно буде передано до викликаючої функції.

Клас DataStorage надає один метод для збереження згенерованого набору з заданим форматом. Під час збереження кожний вектор приводиться до заданого формату та записується у файл sequences.txt у вказану директорію для результатів.

3.2.1 Користувацький інтерфейс

Модуль користувацького інтерфейсу відповідає за взаємодію користувача з програмою. Цей інтерфейс виконаний у вигляді інтерфейсу командного рядка для того щоб надалі цю програму можна було використовувати в автоматизованих системах, які потребують таких генераторів, але не мають змоги використовувати його як пакет в мові Python. При реалізації цього модуля було ураховано що більщість операцій для вводу параметрів потребують перевірки цих параметрів з наданням мжливості користувачу виправити їх, тому у методах отримання параметрів від користувача виокристовуються первірка на введені дані доки користувач не введе допустимий варіант. Також релізовано додатковий метод для форматованого виводу саме на командний рядок.

3.2.2 Конфігурація генераторів

Цей модуль відповідає за завантаження конфігурації системи з файлу, зберігання конфігурації під час роботи програми та надання відповідних параметрів конфігурації за запитом до нього. Конфігурація складається з параметрів, зазначених у таблиці 4:

Ключ	Тип	Опис
n	integer	Розмірність генератора
polinom	list[integer]	Поліном представлений у вигляді номерів виходів регістру
initialStateGenerator	list[integer]	Початковий стан генератора у вигляді списку біт
m	integer	Значення M у проміжній схемі управління
initialStateFixedGenerator	list[integer]	Початковий стан формувача у вигляді списку біт
weight	integer	Вага послідовності з фіксованою вагою

Таблиця 4 – Параметри конфігурації

3.2.3 Первинний генератор

Модуль первинного генератора відповідає за симуляцію роботи генератора на основі регістру зсуву зі зворотнім лінійним зв’язком. Симуляція відбувається на основі алгоритму створеному за роботою схеми. Під час роботи генератор відстежує кількість згенерованих послідовностей для інформування о стані періоду.

Для його роботи потрібні ініціалізуючі параметри, а саме:

- N - розмірність генератора;
- polinom - поліном записаний у вигляді списку номерів виходів регістру, які будуть додаватись за модулем 2;
- initialState - початковий стан генератора, у вигляді списку біт;

Під час ініціалізації генератора він створює імітацію регістру зсуву заданої довжини, відтворює запис в регістр зсуву початкової послідовності, створює послідовність зв’язків між виходами регістру. У випадку відсутності початкового стану у параметрах він ініціалізує його послідовністю біт, яка заповнюється 1 в старшому розряді, а всі інші розряди заповнюються нулями.

Після ініціалізації генератор готовий для роботи. Оскільки цей модуль виконаний в ООП стилі, він успадковує батьківський абстрактний клас PrimaryGenerator, який у свою чергу надає один метод для взаємодії - tick. Цей ме-

тод дозволяє здійснити один такт роботи генератора, симулює подачу синхросигналу на генератор, в результаті виконання цього методу генератор виконає алгоритм, змінить стан свого регістру, зробить перевірку на закінчення періоду генерації, та поверне нову згенеровану послідовність. Якщо це був останній можливий вектор у періоді, для даної конфігурації, то генератор встановить флаг isFinished у значення істини.

Алгоритм роботи генератора відображено на рисунку 11. За цим алгоритмом спершу виконується ініціалізація результату одиницею, бо в формулі обчислення наступного біту генератора в кінці виконується додаваня одиниці. Після цього у циклі виконується перевірка усіх бітів вихідної послідовності на входження поточного індексу біта у перелік індексів що визначають наявність суматора за модулем 2 для цього біта. Якщо поточний індекс є у переліку суматорів, то значення результата перезаписується на сумму за модулем два значення поточного біта та минулого результату, після чого відбувається перехід до наступного індексу. Після перевірки усіх бітів генератора виконується логічний зсув вихідного регістру вправо та запис у перший біт результату отриманоо на попередніх кроках.

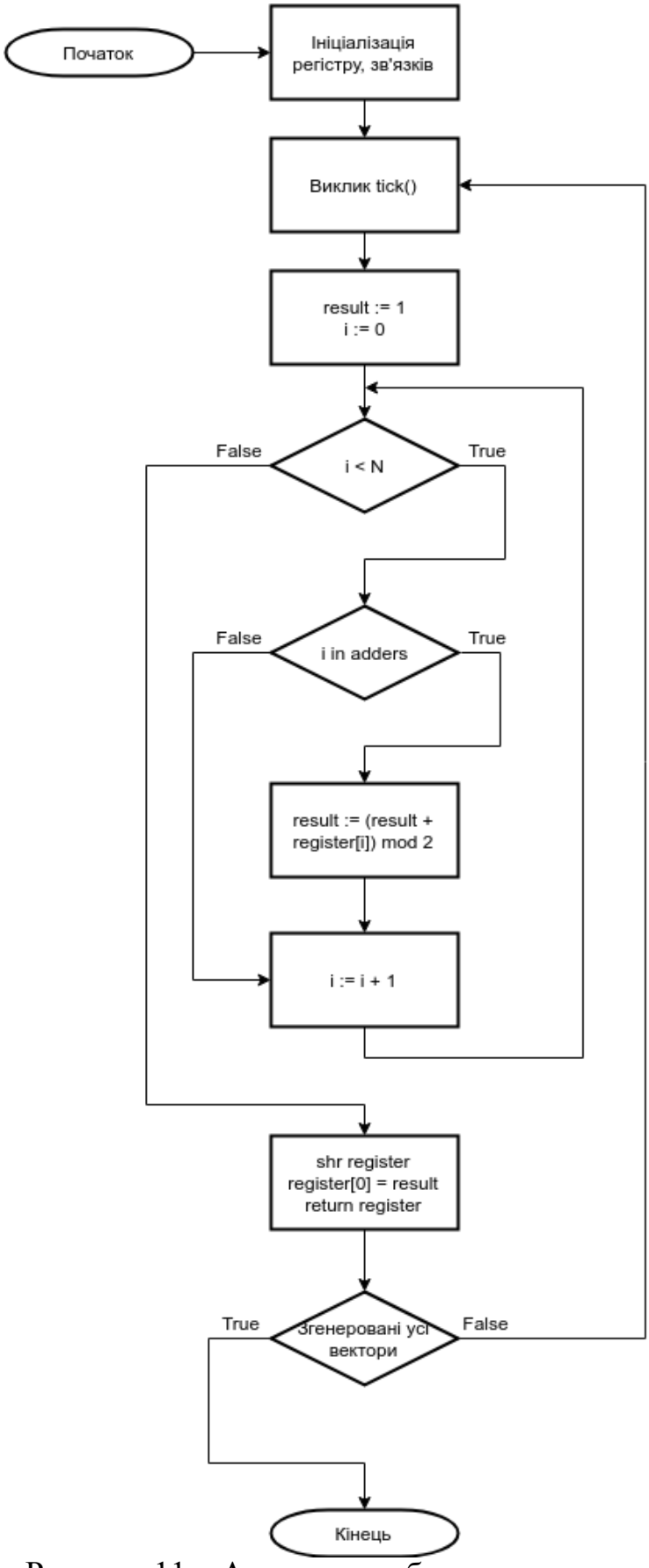

Рисунок 11 – Алгоритм роботи генератора

3.2.4 Проміжна керуюча схема

Модуль проміжної керуючої схеми відповідає за формування нового набору послідовностей, базуючись на наборах, отриманих з вихідного регістру первинного генератора псевдовипадкових послідовностей. Цей модуль конфігурується за допомогою одного параметра: M - кількість біт з кінця регістру первинного генератора з яких буде зчитуватись значення. Також для роботи схеми потрібне посилання на первинний генератор, з якого буде зчитуватись значення.

Загальний алгоритм роботи проміжної керуючої схеми зображено на рисунку 12. За цим алгоритмом під час ініціалізації модуля встановлюється значення M та формується внутрішній вихідний регістр, який може бути ініціалізований поточним значенням вихідного регістру первинного генератора. Після ініціалізації при кожному виклику метода tick, виконується ініціалізація нулями змінної результату перевірки M бітів регістру та лічильника проглянутих бітів. У циклі, доки лічильник не дійшов до біту з номером N - M, тобто до кінця підпослідовності біт, яка перевіряється на нуль, в результат перевірки записується результат операції АБО між поточним значенням результату та значенням і-го біту на вихідному регістрі первинного генератора, після чого виконується декремент лічильника для переходу на наступний біт.

Коли уся підпослідовність була перевірена, визначається кінцеве значення для нового біта на базі інвертованого результату перевірки бітів генератора та останнього біту генератора. Для встановлення нового значення у вихідному регістрі проміжної схеми робиться логічний зсув вправо та записується значення нового біта в перший біт вихідного регістру. Для оптимізації роботи з модулем, після кожного виклику метода tick у викликаючу функцію повертається нове значення вихідного регістру.

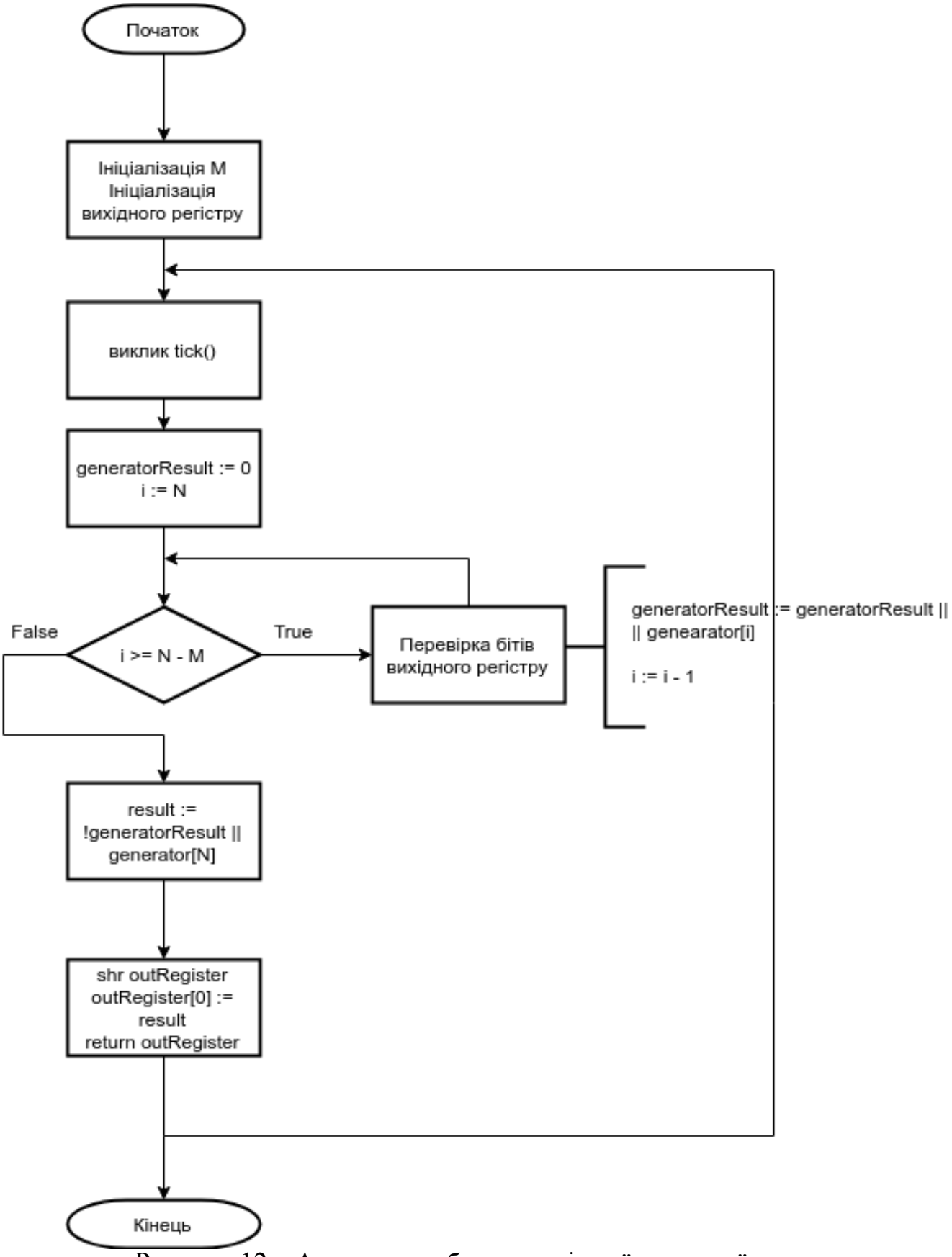

Рисунок 12 – Алгоритм роботи проміжної керуючої схеми

3.2.5 Формувач послідовностей з постійною вагою

Модуль формувача послідовностей з постійною вагою відповідає за формування послідовностей на базі проміжної схеми або первинного генератора.

Для ініціалізації модуля потрібні такі параметри:

- weight - вага формуємих послідовностей;
- initialState - початковий стан генератора, у вигляді списку біт;

Також модулю потрібне посилання на генератор чи регістр, з якого він буде отримувати базові послідовності для формування послідовності з заданою вагою.

Алгоритм роботи формувача послідовностей з заданою вагою відображено на рисунку 13. За цим алгоритмом при визові методу tick формувача виконується наступна послідовність операцій: ініціалізація, перебір біт первинної послідовності, обмін першого з останнім бітів для забезпечення кільцевої структури переносу.

Під час ініціалізації встановлюється лічильник біт, змінна що вказує на останній одиничний біт первинної послідовності, значення переносу, змінна що вказує на перший одиничний біт. Після ініціалізації виконується перебір усіх бітів первинної послідовності з 1 по N. Під час перебору перевіряється поточний біт на його значення. Якщо значення біту дорівнює нулю, тоді цей біт пропускається. Якщо поточний біт первинної послідовності дорівнює одиниці, тоді перевіряється значення посилання на перший одиничний біт первинної послідовності. Якщо він ще не знайдений, то в цю змінну записується поточне значення лічильника i. Після цього перевіряється наявність переносу у змінній val. Якщо перенос наявний, то за допомогою тимчасової змінної ми міняємо значення переносу та значення в поточному біті вихідного регістру формувача. Якщо значення переносу не встановлено, то відбувається запис в змінну переносу значення поточного біту вихідного регістру. Так відбувається доки не будуть перебрані усі біти.

Зважаючи на те що імітуємий регістр є модифікований, до тригерів зі значення додані схеми переносу, та має кільцеву структуру переносу, тобто остання схе-

ма переносу з’єднана з першою, нам потрібно врахувати це в алгоритмі. Тому після завершення циклу ми перевіряємо перший одиничний біт, знайдений у первинній послідовності. Якщо він наявний, то ми замінюємо його зі значенням переносу. Тут можливий випадок, коли цей одиничний біт є єдиним у первинній послідовності, тоді у вихідному регістрі відповідний біт буде поміняний місцями з самим собою. У всіх інших випадках перестановка відбудеться між останнім одиничним бітом та першим.

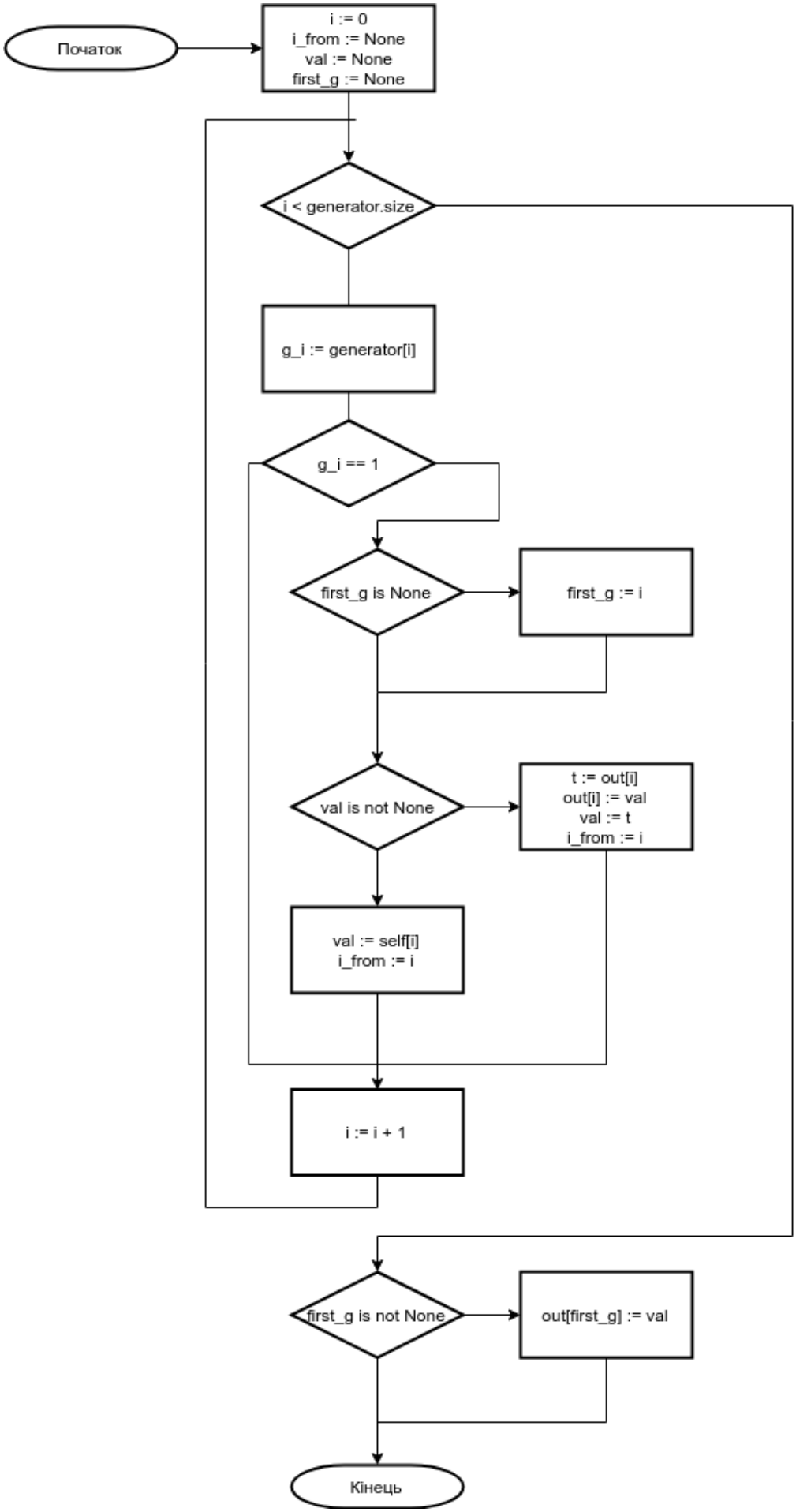

Рисунок 13 – Алгоритм роботи формувача послідовностей з постійною вагою

3.2.6 Управління

Модуль призначений для управління функціоналом та ініціалізації інших модулів. Робота програми починається з нього, коли він створює інтерфейс користувача з якого отримує команди для пошуку конфігурації та створення композиції з генератора псевдовипадкових послідовностей, проміжної схеми та формувача послідовностей. Головна задача цього модуля реагувати на отримані команди з інтерфейсу користувача та оперуючи міжмодульною взаємодією організовувати отримання необхідного результату.

Алгоритм роботи модуля залежить від отриманого режиму роботи з користувацького інтерфейсу. Загальна частина алгоритму однакова для обох режимів. Спочатку виконується ініціалізація користувацького інтерфейсу, та модуля конфігурації. Після отримання режиму роботи та шляху до файлу з конфігурацією цей модуль вказую модулю конфігурації завантажити дані з файлу.

На основі отриманої конфігурації модуль створює імітації складових схеми з вказаними параметрами. Після того як складові схеми було ініціалізовано, цей модуль починає генерувати вектори. Після генерації усіх можливих векторів, за період первинного генератора, він зберігає їх у файл за допомогою вказівки модулю зберігання наборів.

Після завершення загальної частини модуль базуючись на отриманому режимі роботи або завершує свою роботу, або виконує аналіз з виводом результатів у користувацький інтерфейс за допомогою команди модулю обробки результатів.

3.2.7 Зберігання наборів

Модуль зберігання наборів використовується для збереження згенерованих наборів у файл для подальшого використання або аналізу згенерованих наборів. Це необхідно для того щоб була можливість відокремити роботу програми від аналізу, тобто мати можливість аналізувати вже отримані результати надалі. Модуль не має ініціалізуючих параметрів, та виділяє один метод для взаємодії - save, якому при

виклику передається три аргументи:

- vectors - список згенерованих послідовностей;
- format - формат у якому зберігти послідовності;
- path - шлях до файлу у який зберігти послідовності;

Формат для збереження послідовностей може бути або number, або bits. У випадку формату number послідовність буде збережена як цілі числа, кожне число з нового рядка. У випадку формату bits послідовність буде збережена у вигляді двійкового представлення числа, де кожне число з нового рядка.

3.2.8 Обробки результатів

Завдяки цьому модулю здійснюється статистичний аналіз отриманого набору. Модуль є розширюваним, тобто є можливість додавання функцій для аналізу у його перелік функцій аналізу, тим самим роблячи цей модуль гнучкішим. Функції аналіза мають однакову сигнатуру func(dataset, N, M).

де dataset - Pandas DataFrame з згенерованими послідовностями

N - довжина послідовності

M - вага послідовності

Прикладом роботи функції підрахунку ймовірності появи одиниці в окремому біті послідовності може слугувати графік (рисунок 14) та таблиця (табл. 5):

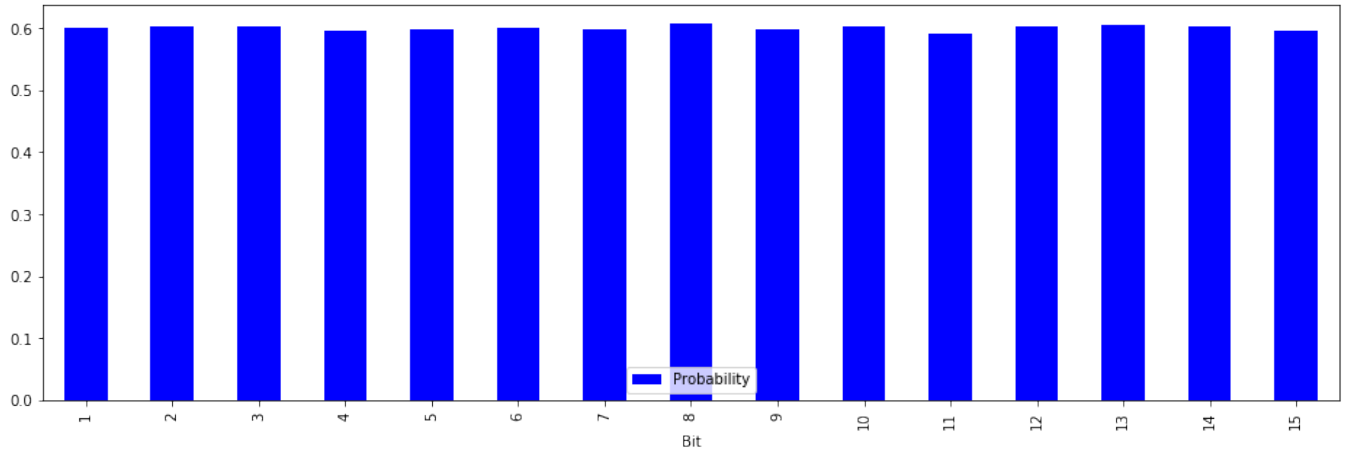

Рисунок 14 – Графік ймовірності появи одиниці в послідовності довжиною 15 та вагою 9

Біт	Ймовірність
0	0.599749748222
1	0.602557451094
2	0.60280159917
3	0.59465315714
4	0.59813226722
5	0.600054933317
6	0.599078341014
7	0.606921597949
8	0.598040711692
9	0.602038636433
10	0.590990936003
11	0.603228858303
12	0.604327524644
13	0.60164189581
14	0.59578234199

Таблиця 5 – Ймовірності появи одиниці в послідовності довжиною 15 та вагою 9

3.3 Режими роботи програми

Зважаючи на те що ця програма імітує роботу формувача псевдовипадкових послідовностей з заданою вагою, ми можемо використовувати її як для генерації послідовностей для використання у тестуванні надійності системи так і для

знаходження конфігурацій для генерації необхідних наборів. Саме тому програма реалізує роботу в двох режимах: аналізу отриманого набору та робочому.

3.3.1 Аналіз отриманого набору

Цей режим роботи дозволяє виконувати аналіз статистичних властивостей згенерованих наборів. Він потрібен у випадках коли користувачу потрібно знайти вдалу конфігурацію, або дослідити існуючу. У цьому режимі програма виконує повну генерацію усіх векторів набору, збереження їх у файл та аналіз отриманого набору. Серед статистичних властивостей набору аналізується ймовірність появи одиниці у кожному біті окремо

3.3.2 Робочий режим

Робочий режим не здійснює аналізу отриманого набору, тому він призначений лише для формування послідовностей. Цей режим також як і режим аналізу отриманого набору генерує усі вектори набору за встановленою конфігурацією та зберігає отримані набори у файл. Цей режим вигідно використовувати з точки зору продуктивності, бо не витрачається час на аналіз, побудові графіків та інше.

3.4 Інструкція роботи для користувача

Завдяки тому що взаємодія користувача з програмою реалізована за допомогою інтерфейсу командного рядка програма є незалежною від графічного оточення та працює у більшості емуляторів терміналів операційних систем.

Для запуску програми необхідно вказати режим роботи та шлях до файлу конфігурації. Ці параметри можуть бути вказані при виклику програми або введені в інтерактивному режимі, якщо не були вказані при запуску програми. Запуск програми має такий формат: python3 main.py <config_file_path> <mode> <output_dir> <out_format>. Чотири параметри для запуску є опціональними, але мають суворий порядок. Перший параметр відповідає за шлях до файлу конфігурації з якого буде

завантажено параметри для усіх складових програми. Другий параметр відповідає за режим роботи програми і допускає два значення:

- work - робочий режим в якому будуть згенеровані усі вектори та збережені у файл;
- analyze - режим аніалізу конфігурації;

Третій параметр відповідає за директорію, у яку буде збережено результат роботи програми. Четвертий параметр відповідає за формат в якому буде збережено згенерований набір.

При запуску програми без параметрів користувачу буде запропоновано ввести спочатку шлях до файлу з конфігурацією, як зображено на рисунку 15

```
pavel@deb-hp17:/tmp/program$ python3 main.py
Provide configuration file path: config.yaml
config.yaml
Configuration will be loaded from config.yaml
Configuration loaded
{'size': 8, 'adds': [8, 6, 5, 4], 'weight': 5, 'm': 3}
```

Рисунок 15 – Запуск програми та вказання файлу конфігурації

Після вказання шляху до файлу конфігурації, його буде завантажено, розібрано та збережено у модулі конфігурації. Враховуючи те, що файл має формат запису yaml, то можливий варіант, що вміст файлу не відповідає специфікації цього стандарту. В такому випадку буде виброшене виключення що повідомить про помилку розбору файлу, як показано на рисунку 16.

```
pavel@deb-hp17:/tmp/program$ python3 main.py
Provide configuration file path: config.yaml
config.yaml
Configuration will be loaded from config.yaml
YAML parsing error:  mapping values are not allowed here
  in "config.yaml", line 2, column 5
```

Рисунок 16 – Помилка при розборі конфігураційного файлу

У випадку коли файл не було знайдено - буде виброшене виключення, що повідомить про неможливість знайти вказаний файл, як показано на рисунку 17.

Після цього буде запропоновано наново ввести шлях до файлу в інтерактивному режимі, а в режимі запуску з вказаними параметрами при виклику - програма завершиться з кодом 1.

```
pavel@deb-hp17:/tmp/program$ python3 main.py
Provide configuration file path: conf.tt
conf.tt
Configuration will be loaded from conf.tt
Configuration file not found
```

Рисунок 17 – Помилка при спробі відкрити конфігураційний файл

Після успішного вказання файлу конфігурації потрібно вказати режим роботи програми. Користувачу виводиться підказка щодо доступних режимів роботи, як зображено на рисунку 18. У випадку некоректно введеного режиму роботи користувачу буде запропоновано ввести режим роботи ще раз.

```
Select execution mode.
1 - work mode. Just save sequences to file
2 - analyze mode. Save sequences to file and provide staistics info by sequences
Mode: 1
```

Рисунок 18 – Вибір режиму роботи програми в інтерактивному режимі

Якщо програму було запущено з параметрами, серед яких режим роботи було вказано некоректно, то програма завершиться зі статусом 2, а в терміналі буде виведено повідомлення про невідповідність вказаного режиму дозволеним варіантам, як на рисунку 19.

```
pavel@deb-hp17:/tmp/program$ python3 main.py config.yaml test
Configuration will be loaded from config.yaml
Configuration loaded
{'size': 8, 'adds': [8, 6, 5, 4], 'weight': 5, 'm': 3}
Invalid program mode. Should be 'work' or 'analyze'
```

Рисунок 19 – Помилка при вказанні некоректного режиму роботи

Наступним кроком роботи програми є визначення директорії, в яку буде збережено згенерований файл з послідовностями у випадку робочого режиму роботи та файлу послідовностей, звіту по статистичним характеристикам згенерованого

набору, графіків у випадку режиму роботи - аналіз. На цьому кроці також визначається формат, у якому буде збережено згенеровані послідовності. Результат роботи цього кроку зображено на рисунку 20.

```
pavel@deb-hp17:/tmp/program$ python3 main.py config.yaml work
Configuration will be loaded from config.yaml
Configuration loaded
{'size': 8, 'adds': [8, 6, 5, 4], 'weight': 5, 'm': 3}
Provide output dir path: ./out
./out
Select output format.
1 - binary format
2 - integer format
Output format: 1
```

Рисунок 20 – Вибір директорії для запису результатів та формату запису

У випадку визначення неприпустимого формату для збереження, буде запропоновано вказати формат ще раз. Якщо програму було запущено з параметрами, то буде виконано завершення програми зі статусом 3 та виводом помилки на термінал, як показано на рисунку 21.

```
pavel@deb-hp17:/tmp/program$ python3 main.py config.yaml work ./out test
Configuration will be loaded from config.yaml
Configuration loaded
{'size': 8, 'adds': [8, 6, 5, 4], 'weight': 5, 'm': 3}
Invalid out format. Should be 'bin' or 'int'
```

Рисунок 21 – Помилка при вказанні некоректного формату запису

4 АНАЛІЗ РОБОТИ

4.1 Статистичні властивості отриманих послідовностей

Конфігурація №1

N	Поліном	W	M
17	$x^{17} + x^{14} + 1$	12	12

Таблиця 6 – Конфігурація 1

Конфігурація №2

N	Поліном	W	M
17	$x^{17} + x^{14} + 1$	8	8

Таблиця 7 – Конфігурація 2

Результати:

Біт	Конфігурація 1	Конфігурація 2
0	0.705068245455	0.472362307452
1	0.706182145555	0.470477832625
2	0.707006126451	0.470645680585
3	0.706479694212	0.469173196207
4	0.704602848838	0.471172112824
5	0.705548900977	0.470981376506
6	0.707425746351	0.470309984665
7	0.704381594708	0.470866934715
8	0.706494953117	0.470454944267
9	0.706205033913	0.471462032028
10	0.70739522854	0.468677281779
11	0.705571789336	0.468974830435
12	0.706662801077	0.472224977302
13	0.704930915305	0.470935599789
14	0.707105309336	0.472873480785
15	0.705434459186	0.469836958595
16	0.703504207643	0.46857046944

Таблиця 8 – Результати 3

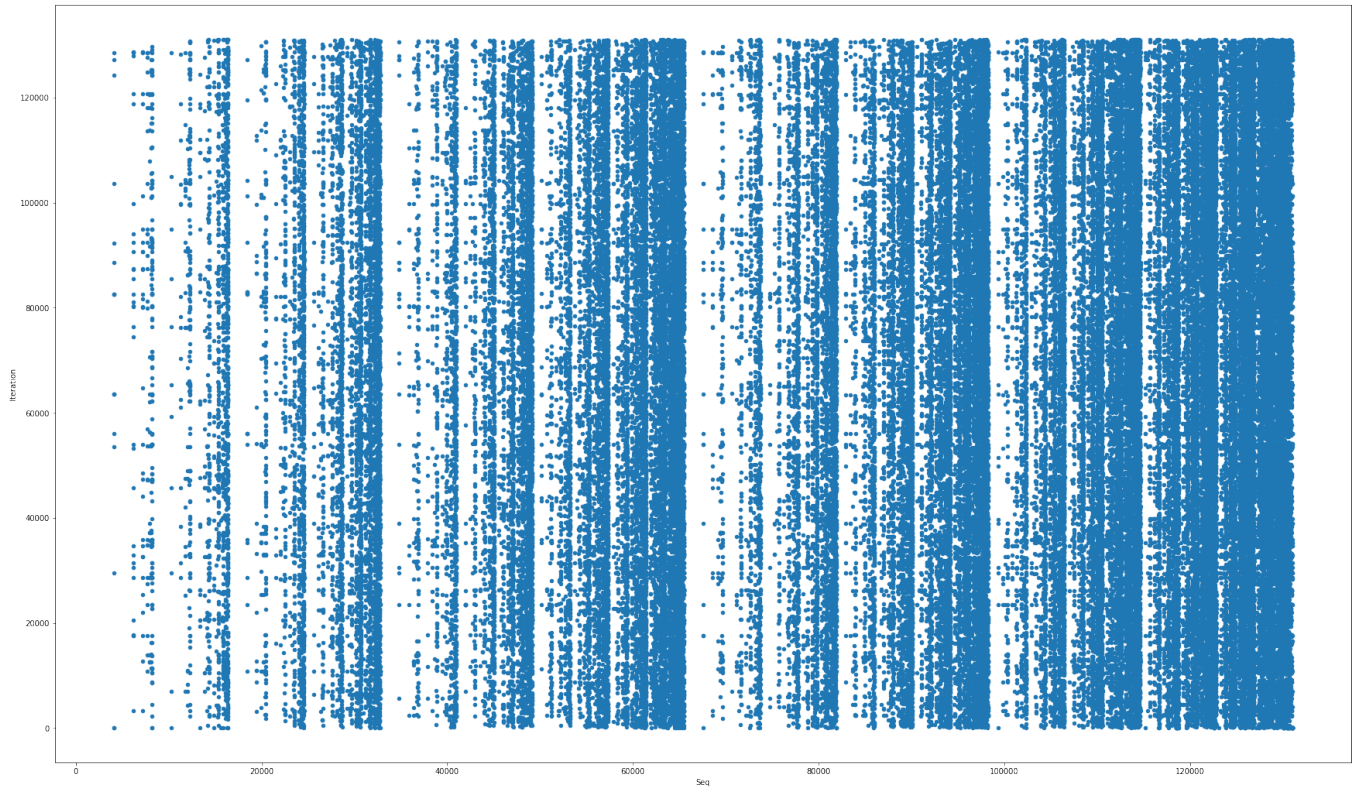

Рисунок 22 – Графік розкидання значень для конфігурації 1

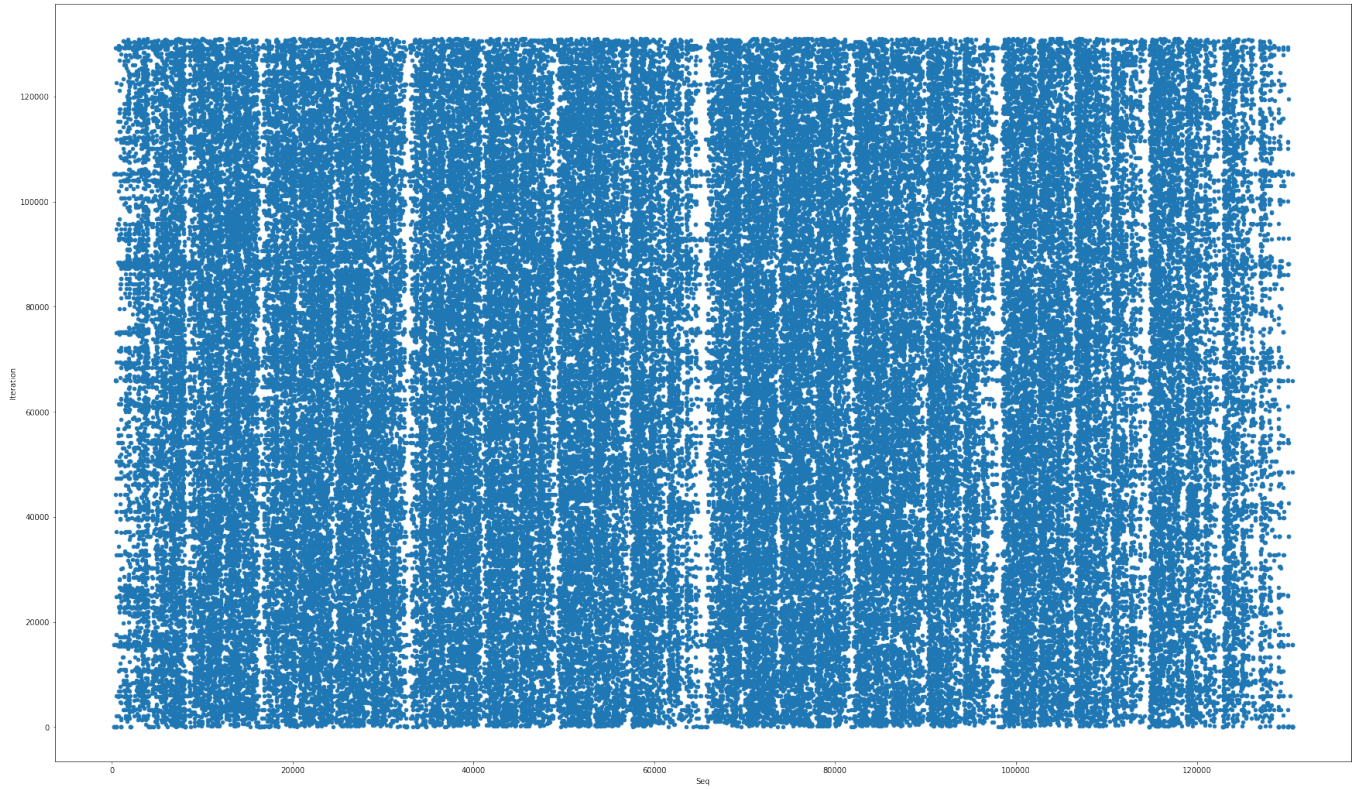

Рисунок 23 – Графік розкидання значень для конфігурації 2

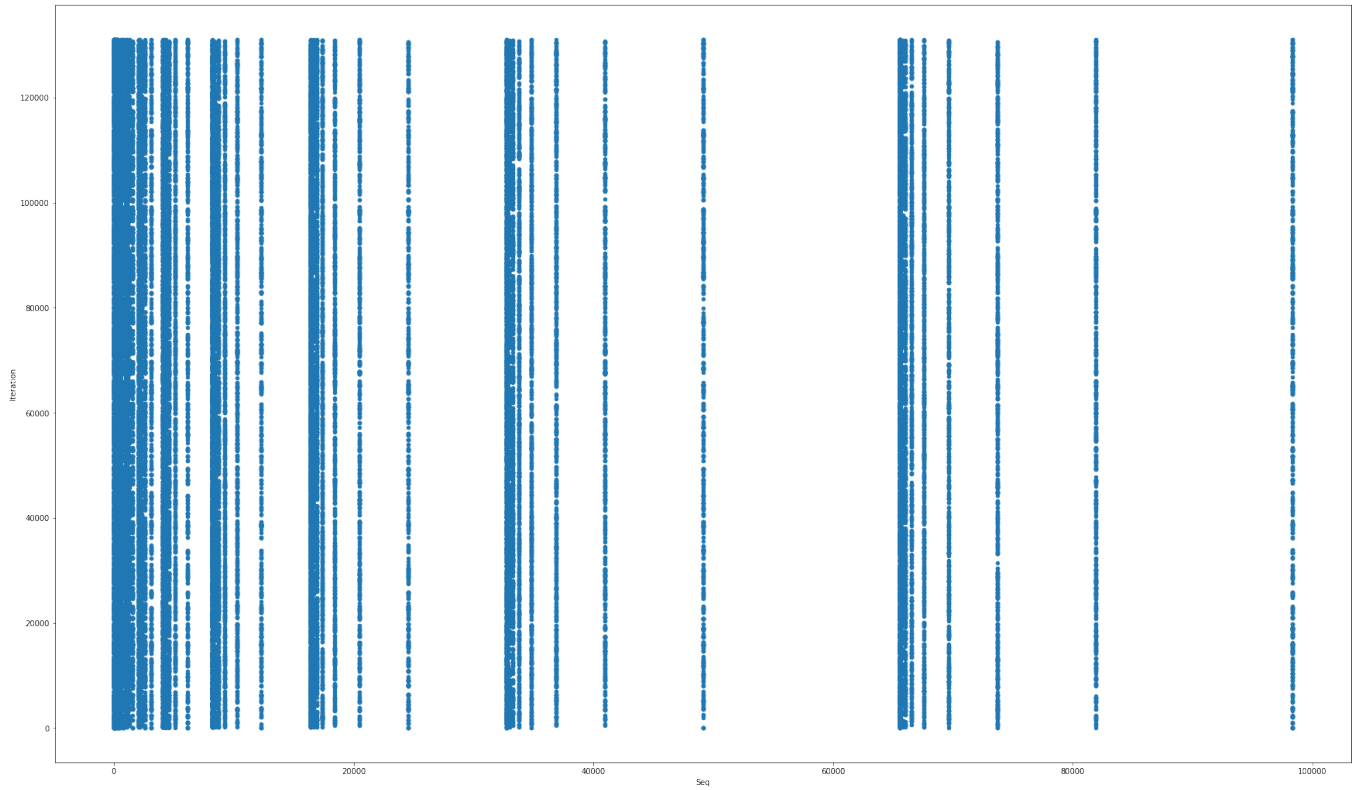

Рисунок 24 – Графік розкидання значень для послідовності з вагою 2

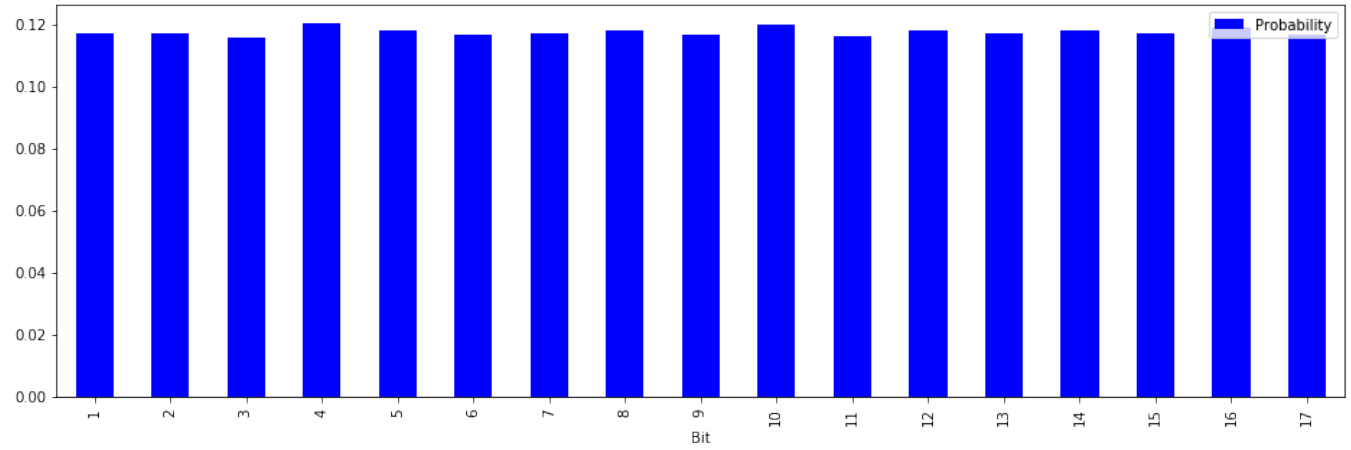

Рисунок 25 – Графік ймовірності одиниці в біті для послідовності з вагою 2

4.2 Часові характеристики генерації

Для дослідження часових характеристик генерації використовується генерація за допомогою базової схеми формувача та модифікації з проміжною схемою. Далі будуть наведені конфігурації та порівняння їх результатів. У приведених таблицях результатів значення кращі у рядку «Унікальних наборів» там, де воно ви-

ще, а у рядку «Згенеровано за» там де воно нижче. Тобто чим більше можливих наборів можна згенерувати - тим краще, чим за меншу кількість ітерацій було згенеровано - тим краще.

Конфігурація №1

N	Поліном	W	M
15	$x^{15} + x^{14} + 1$	9	10

Таблиця 9 – Конфігурація 1

Результати:

	Без проміжної схеми	З проміжною схемою	Різниця
Унікальних наборів	4999	4998	-1
Загальна к-ть ітерацій	32767	32767	0
Згенеровано за	32469	32419	-50
%	99.09	98.93	-0.16%

Таблиця 10 – Результати 1

Конфігурація №2

N	Поліном	W	M
17	$x^{17} + x^{14} + 1$	10	12

Таблиця 11 – Конфігурація 2

Результати:

	Без проміжної схеми	З проміжною схемою	Різниця
Унікальних наборів	19424	19427	+3
Загальна к-ть ітерацій	131071	131071	0
Згенеровано за	129837	128895	-942
%	99.05	98.33	-0.72%

Таблиця 12 – Результати 2

Конфігурація №3

N	Поліном	W	M
17	$x^{17} + x^{14} + 1$	12	12

Таблиця 13 – Конфігурація 3

Результати:

	Без проміжної схеми	З проміжною схемою	Різниця
Унікальних наборів	6188	6188	0
Загальна к-ть ітерацій	131071	131071	0
Згенеровано за	53914	53613	-301
%	41.13	40.9	-0.23%

Таблиця 14 – Результати 3

Конфігурація №4

N	Поліном	W	M
12	$x^{12} + x^{11} + x^{10} + x^4 + 1$	4	5

Таблиця 15 – Конфігурація 4

Результати:

	Без проміжної схеми	З проміжною схемою	Різниця
Унікальних наборів	495	491	-4
Загальна к-ть ітерацій	4095	4095	0
Згенеровано за	4046	4063	+17
%	98.8	99.21	+0.41%

Таблиця 16 – Результати 4

Більш наочно залежність значення M у комутаційній схемі, вагою генерує-мої послідовності та швидкості генерації усіх можливих послідовностей з задано-ною вагою можна побачити на графіках залежності для схеми, сконфігурованої для розміру послідовності 17 з суматорами за модулем 2 на виходах 17 та 14. Графіки залежності зображено на рисунках 26 - 30. На цих графіках червоною лінією, яка

підписана minItWithoutCommutation, зображено кількість тактів роботи схеми, необхідних для генерації усіх можливих послідовностей заданої ваги без модифікації комутаційною схемою. Синьою лінією зображено кількість тактів знадобившихся для генерації усіх послідовностей з модифікацією комутаційною схемою.

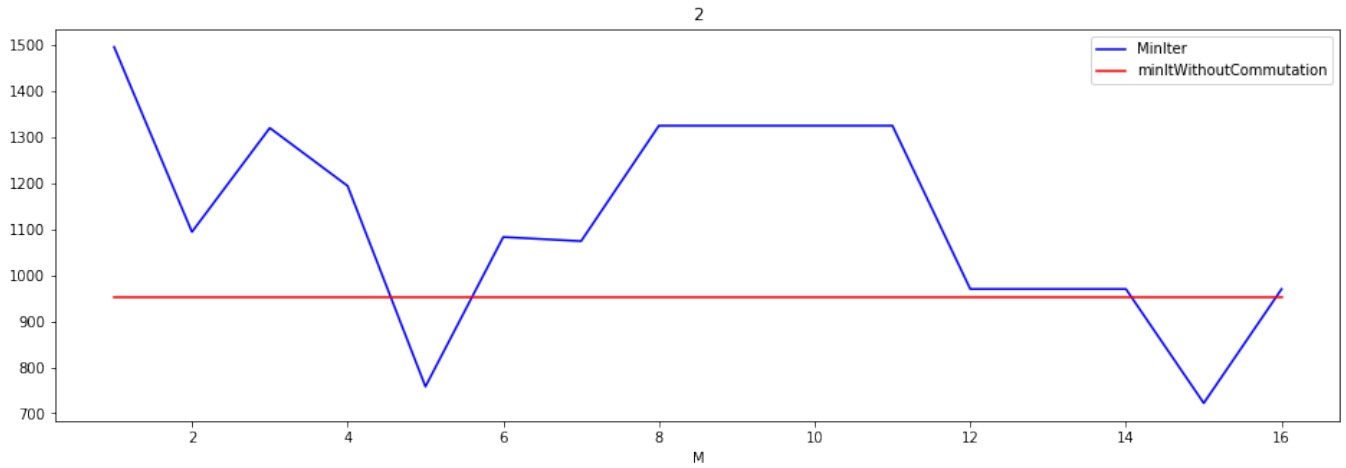

Рисунок 26 – Графік залежності тактів для генерації всіх послідовностей з вагою 2

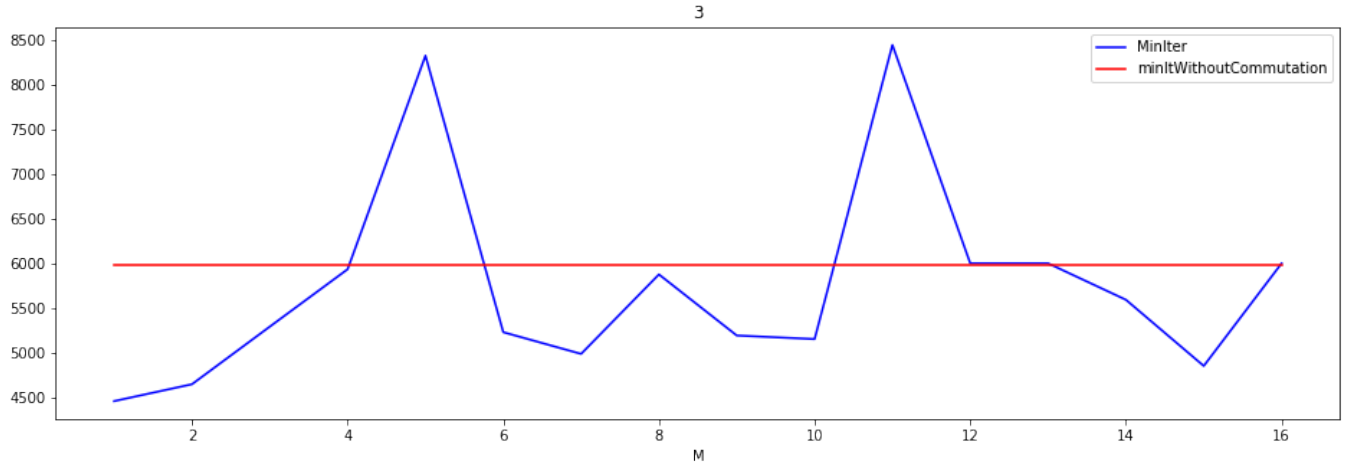

Рисунок 27 – Графік залежності тактів для генерації всіх послідовностей з вагою 3

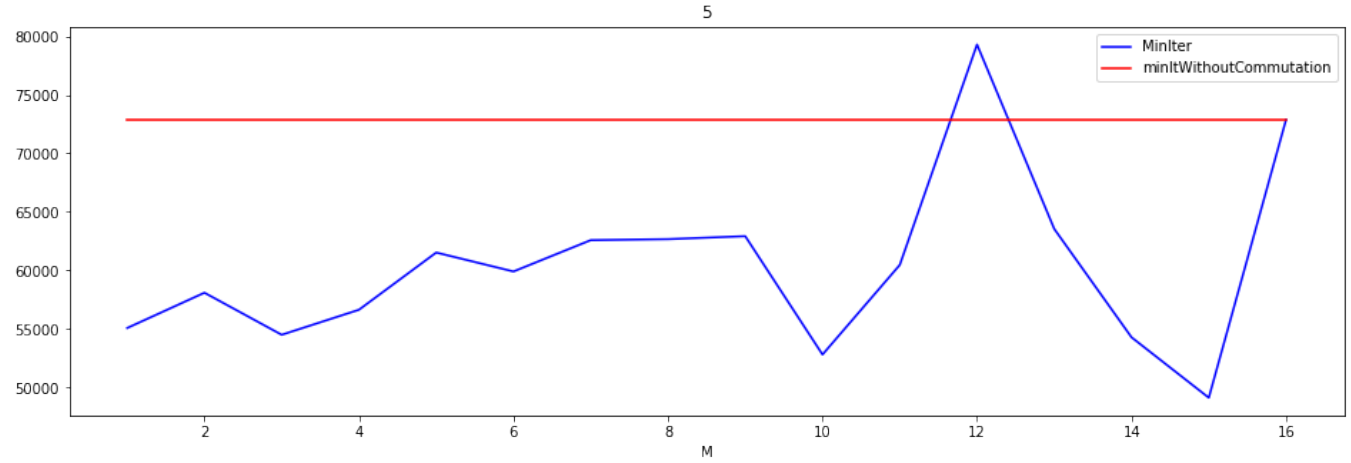

Рисунок 28 – Графік залежності тактів для генерації всіх послідовностей з вагою 5

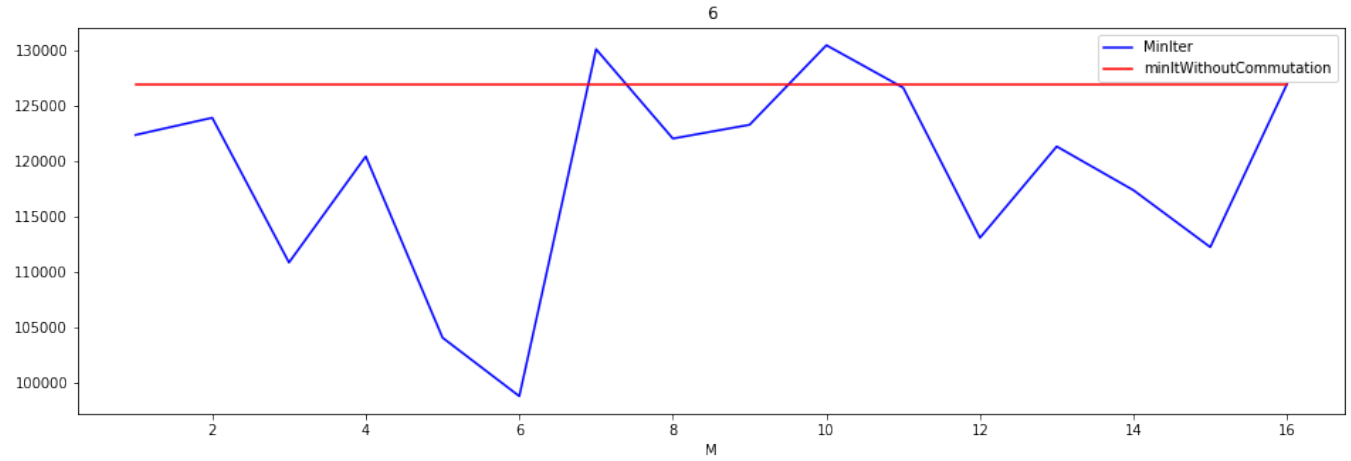

Рисунок 29 – Графік залежності тактів для генерації всіх послідовностей з вагою 6

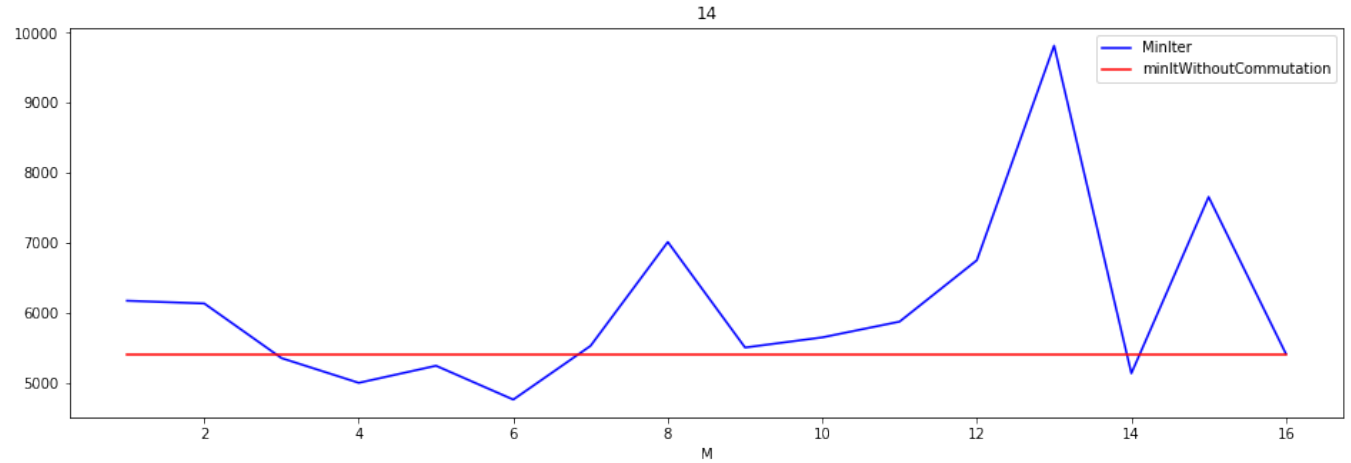

Рисунок 30 – Графік залежності тактів для генерації всіх послідовностей з вагою 14

Як можна побачити з результатів у таблицях та графіках, конфігурація кожного елементу схеми грає важливу роль. Так зміна значення M - змінює кількість біт, які перевіряються на первинному генераторі, тим самим зменшуючи можливі послідовності в більшості випадків, але інколи навпаки збільшую кількість можливих перестановок завдяки дублюванню деяких векторів у певний момент. Тому використовувати дану модифікацію потрібно після підбору конфігурації та перевірки її оптимальності.

Зважаючи на наявність залежності між вагою послідовності та кількістю біт, перевіряємих у комутаційній схемі, та на те що при вдалій конфігурації виникає значне підвищення ефективності роботи генератора, можна побудувати таблицю з ефективними конфігураціями, яка отримана шляхом повного перебору можливих ваг послідовностей та значень M. У результуючій таблиці 17 наведені деякі оптимальні варіанти конфігурацій схеми. Варіанти з вагою 1 та 16 опущено через малу кількість генеруємих послідовностей яка суттєво впливає на кількість ітерацій, необхідних для їх генерації.

Вага	M	к-сть послідовностей	ітер. для генерації усіх	% від всіх ітер.
2	5	136	758	0.58
	15	136	722	0.55
3	1	680	4459	3.40
	2	680	4648	3.55
	3	680	5293	4.04
	4	680	5936	4.53
	6	680	5231	3.99
	7	680	4988	3.81
	8	680	5878	4.48
	9	680	5194	3.96
	10	680	5154	3.93
	14	680	5594	4.27

Вага	M	к-сть послідовностей	ітер. для генерації усіх	% від всіх ітер.
3	15	680	4853	3.70
4	3	2380	19841	15.14
	5	2380	19082	14.56
	6	2380	21186	16.16
	7	2380	19028	14.52
	9	2380	20458	15.61
	13	2380	21559	16.45
5	1	6188	55085	42.03
	2	6188	58098	44.33
	3	6188	54506	41.59
	4	6188	56643	43.22
	5	6188	61530	46.94
	6	6188	59912	45.71
	7	6188	62587	47.75
	8	6188	62674	47.82
	9	6188	62927	48.01
	10	6188	52817	40.30
	11	6188	60470	46.14
	13	6188	63544	48.48
	14	6188	54290	41.42
	15	6188	49131	37.48
6	1	12373	122387	93.37
	2	12376	123939	94.56
	3	12376	110882	84.60
	4	12376	120451	91.90
	5	12376	104099	79.42
	6	12375	98823	75.40

Вага	M	к-сть послідовностей	ітер. для генерації усіх	% від всіх ітер.
6	8	12376	122064	93.13
	9	12376	123306	94.08
	11	12375	126649	96.63
	12	12376	113108	86.30
	13	12376	121350	92.58
	14	12375	117401	89.57
	15	12376	112267	85.65
9	9	24178	130708	99.72
	15	24197	130660	99.69
10	3	19419	128242	97.84
	5	19425	129338	98.68
	12	19427	128894	98.34
11	1	12370	120111	91.64
	2	12376	121744	92.88
	4	12376	114508	87.36
	8	12376	107002	81.64
	9	12376	109226	83.33
	10	12374	110586	84.37
	11	12376	114905	87.67
	12	12376	109240	83.34
	13	12374	107374	81.92
	14	12376	103937	79.30
12	4	6188	52741	40.24
	6	6188	52696	40.20
	10	6188	50713	38.69
	12	6188	53612	40.90
	14	6188	51681	39.43

Вага	M	к-сть послідовностей	ітер. для генерації усіх	% від всіх ітер.
12	15	6188	53719	40.98
13	7	2380	17879	13.64

Таблиця 17 – Оптимальні конфігурації

З наведених у цій таблиці конфігурацій, для послідовності довжиною 17 біт, можна побачити кількість послідовностей які можна згенерувати за заданою вагою та параметрами схеми комутації, номер ітерації на якій будуть згенеровані усі можливі послідовності заданої ваги та відсоток необхідних ітерацій від загальної кількості ітерацій первинного генератора.

Наведені конфігурації є оптимальними внаслідок того що кількість тактів для генерації усіх можливих векторів заданої ваги є меншою ніж в аналогічної схеми без проміжної логічної схеми для перевірки та зміни послідовності з первинного генератора.

ВИСНОВКИ

Зважаючи на отримані результати, можна зробити висновок, що розроблена схема має сенс, бо в деяких випадках дозволяє отримати кращі результати, ніж існуючі аналоги. Розроблена програма дозволяє комфортно підбирати конфігурації та аналізувати отримані результати. Програма може працювати з великими розмірностями генераторів та їх конфігураціями. Уніфікований інтерфейс роботи з імітаційними модулями дозволяє поєднувати генератор, керуючу схему та формувач між собою. При необхідності отримати кращі результати в генерації є можливість підбирати довільні варіанти налаштувань складових системи. Оскільки програма реалізована на мові Python, складові програми можуть біти використані як пакети в інших застосунках.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Rabah AlShboul, Vitaliy A. Romankevich. Method of Numbers' Dichotomic Decomposition for Generation of Equal Probability Binary Sets // IJCSNS International Journal of Computer Science and Network Security.– 2019.– Vol. 19, No.2.– P. 120-125.
2. Rabah AlShboul, Vitaliy A. Romankevich. Structural Means Generating Pseudorandom Sequences Of Fixed Weight Binary Patterns // IJCSNS International Journal of Computer Science and Network Security. – 2017. – Vol. 17, No.10. – P. 62-66.
3. Романкевич А.М., Майданюк И.В., Романкевич В.А О формировании функций управления для генератора последовательностей двоичных векторов // Радіоелектронні і комп'ютерні системи.-№6, 2014.- С.157-163
4. Романкевич В.А., Майданюк И.В. Структурный метод формирования двоичных псевдослучайных векторов заданного веса // УСиМ. - 2011. - № 5. - С. 28-33, 58.
5. Романкевич В.О., Майданюк І.В., Фесенюк А.П., Шкира Д.С. Генератор рівноважних векторів для проведення статистичних експериментів з GL-моделями // Науковий вісник Чернівецького університету. Сер.: Комп'ютерні системи та компоненти.– 2010.– Т.1, вип. 2.– С.28-30
6. Гроль В.В., Романкевич В.А., Потапова Е.Р., Мораведж Сейед Милад. Структурный метод генерации псевдослучайных последовательностей специального вида // Радіоелектронні і комп'ютерні системи.–№5, 2010.– С.230-236
7. Моделирование. Тестирование, надёжность, контроль и диагностика компьютерных систем [Електронний ресурс]: учебное пособие для изучения дисциплин «Моделирование» и «Тестирование, надёжность, контроль и диагностика компьютерных систем» для иностранных студентов специальности «Компью-

терные системы и сети», «Системное программирование» и «Специализированные компьютерные системы» / НТУУ «КПІ» ; сост. В. В. Гроль, В. А. Романкевич, Е. Р. Потапова.– Електронні текстові дані (1 файл: 782.5 Кбайт). – Киев: НТУУ «КПИ», 2011

8. Гроль В.В., Романкевич В.О. Базові поняття і конструкції мови програмування Сі. Методичні вказівки до вивчення дисципліни “Моделювання” // Київ.– “Політехніка”, 2003.– 24с.
9. Самофалов К.Г., Романкевич А.М., Валуйский В.Н., Каневский Ю. С. , Пиневич М.М. Прикладная теория цифровых автоматов. – К.: Вища Школа. – 1987г. – 375с.
10. Гроль В.В., Романкевич В.А., Потапова Е.Р. Организация процедур логического моделирования цифровых блоков. Методические указания к изучению дисциплин «Моделирование», «Тестирование, надёжность, контроль и диагностика компьютерных систем» // Київ.– “Принт-центр”, 2007.– 44с.
11. Романкевич В.О., Фесеннюк А.П. Про розрахунок надійності відмовостійких багатопроцесорних систем, підсистеми яких мають спільні процесори // Радіоелектронні і комп‘ютерні системи.– №3, 2010.– №3.– С. 62-67
12. Романкевич А.М., Гроль В.В., Романкевич В.А., Фесенюк А.П. Оценка погрешности статистического расчёта надёжности ОМС, которым соответствуют иерархические GL-модели // Радіоелектронні і комп‘ютерні системи.–№7, 2010.– С.142-146
13. В.В. Гроль, В.А. Романкевич, А.П. Фесенюк. Об оценке погрешности расчета надежности отказоустойчивых многопроцессорных систем // Радіоелектронні і комп‘ютерні системи.–№5, 2009.– С.56-59
14. Романкевич О.М., Карачун Л.Ф., Романкевич В.О. До питання побудови моделі поведінки багатомодульних систем // Наукові вісті НТУУ “КПІ”.– 1998.– №1.– С.38-40.

15. Романкевич А.М, Гроль В.В., Карачун Л.Ф., Орлова М.Н., Романкевич В.А. Об одном подходе к расчёту надёжности отказоустойчивых многопроцессорных систем // ВМНТС ”АСУ и приборы автоматики”.– ХНУРЭ, Харьков.– 2002.– №119.–С.54-58.
16. Simple Plot — Matplotlib 3.1.0 documentation. [Електронний ресурс] - Режим доступу: https://matplotlib.org/gallery/lines_bars_and_markers/simple_plot.html#sphx-glr-gallery-lines-bars-and-markers-simple-plot-py
17. Visualization — pandas 0.24.2 documentation. [Електронний ресурс] - Режим доступу: https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html
18. The pandas project — pandas: Python Data Analysis Library. [Електронний ресурс] - Режим доступу: https://pandas.pydata.org/about.html
19. Андреас Мюллер, Сара Гвидо. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными = Introduction to Machine Learning with Python: A Guide for Data Scientists. — Вильямс, 2017. — 480 с. — ISBN 978-5-9908910-8-1, 978-1-449-36941-5.