Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)

Кафедра Автоматизації проектування обчислювальної техніки
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)
(рівень вищої освіти)

Моделі та методи проектування завадостійких систем бездротової передачі даних з використанням ПЛІС.
(тема)

Виконав: студент 2 курсу, групи СКСм-19-1
Сергієнко В.І.
(прізвище, ініціали)

Спеціальність
123 – Комп'ютерна інженерія
(код і повна назва спеціальності)

Тип програми
освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Спеціалізовані комп'ютерні системи
(повна назва освітньої програми)

Керівник доц. Філіппенко І.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Чумаченко С.В.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
Кафедра Автоматизації проектування обчислювальної техніки
Рівень вищої освіти другий (магістерський)
Спеціальність 123 – Комп'ютерна інженерія
Тип програми Освітньо-професійна
Освітня програма Спеціалізовані комп'ютерні системи

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____» _____ 2020 р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ (ПРОЕКТ)

Студентові *Сергієнко Владиславу Ігоровичу*
(прізвище, ім'я, по батькові)

1. Тема роботи (проекту) Моделі та методи проектування завадостійких систем бездротової передачі даних з використанням ПЛІС
затверджена наказом по університету від " 30 " 10 2020 р. № 1489 Ст.
2. Термін подання студентом роботи (проекту) 13.12.2020
3. Вихідні дані до роботи (проекту)
Мова опису апаратури VHDL,
САПР Active-HDL,
САПР XILINX Vivado,
ПЛІС XILINX XC7Z035-2LFBG676i.

4. Зміст пояснювальної записки (перелік питань, що потрібно розробити)
Аналіз предметної галузі та постановка задачі проектування.
Розробка кодера завадостійкого кодування.
Розробка декодера завадостійкого кодування.
Розробка VHDL моделей кодеку.
Моделювання розробленої моделі з використанням інструментальних засобів Active-HDL.
Синтез апаратної реалізації пристрою з використанням САПР Xilinx Vivado.
Проведення діагностичного експерименту.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів) 14 слайдів

6. Консультанти розділів роботи (проекту)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

7. Дата видачі завдання 30.10.2020

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи (проекту)	Термін виконання етапів проекту (роботи)	Примітка
1	Видача теми проекту, узгодження і затвердження теми	30.10.2020 – 01.11.2020	
2	Аналіз проблемної галузі, постановка задачі, вибір інструментальних засобів	02.11.2020 – 04.11.2020	
3	Розробка моделі кодера	05.11.2020 – 07.11.2020	
4	Розробка моделі декодера	08.11.2020 – 11.11.2020	
5	Розробка VHDL опису кодека	12.11.2020 – 17.11.2020	
6	Проведення діагностичних експериментів	18.11.2020 – 22.11.2020	
7	Оформлення пояснювальної записки	23.11.2020 – 06.12.2020	
8	Перевірка виконаного проекту керівником, допуск до захисту	07.12.2020 – 13.12.2020	
9	Захист проекту	14.12.2020 – 27.12.2020	

Студент _____ _____
(підпис)

Керівник роботи (проекту) _____ доц. Філіппенко І.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 65 сторінок, 35 рисунків, 2 таблиці, 17 джерел за переліком посилань.

VHDL-МОДЕЛЬ, СИНТЕЗ, КОДИ З НИЗЬКОЮ ЩІЛЬНІСТЮ ПЕРЕВІРОК НА ПАРНІСТЬ, ЗГОРТКОВЕ КОДУВАННЯ, ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ

Проведено аналіз існуючих на даний момент видів завадостійкого кодування, їх характеристик і принципів побудови та функціонування. Проведено математичне моделювання двох видів завадостійкого кодування інформації в радіолінії зв’язку та виконано їх порівняння. Спроектовано тестопридатний пристрій реконфігурованого LDPC-кодера та декодера. При проектуванні пристрою сформовано VHDL-модель. Виконано синтез та імплементацію пристрою в середовищі Xilinx. Проведено діагностичний експеримент розроблених модулів.

Розроблені модулі пропонується використовувати для реалізації завадостійкого кодування інформації за допомогою LDPC в системах зв’язку.

ABSTRACT

Master’s thesis contains 65 pages, 35 figures, 2 annexes, 17 sources according to the list of links.

VHDL MODEL, SYNTHESIS, LOW DENSITY PARITY-CHECK CODES, CONVOLUTIONAL CODING, IMPLEMENTATION

The analysis of existing types of noise protection coding, their characteristics and principles of construction and functioning is made. The mathematical modelling of two kinds of noise-protecting encoding in the radio link of communication is carried out and their comparison is performed. A prototype device for reconfigurable LDPC encoder and decoder has been designed. Formed VHDL model of device. Synthesis of the device in the Xilinx environment is conducted. Diagnostic experiment of the developed modules was done.

Created modules are proposed to be used to implement the noise-protecting encoding of information using LDPC in communication systems.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ....7
ВСТУП....8
1 АНАЛІЗ ПРОБЛЕМИ. ПОСТАНОВКА ЗАДАЧІ ПРОЕКТУВАННЯ....10
1.1 Загальна характеристика методів кодування....10
1.2 Первинне кодування....11
1.3 Економне кодування....11
1.4 Завадостійке кодування....12
1.4.1 Згорткове кодування....14
1.4.2 Турбокоди....18
1.4.3 LDPC-кодування даних....20
1.5 Обґрунтування вибору завадостійкого кодування....31
2 МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ РАДІОЛІНІЙ З РІЗНИМИ МЕТОДАМИ ЗАВАДОСТІЙКОГО КОДУВАННЯ....32
2.1 Математичне моделювання згорткового кодування в радіолінії з методом модуляції QPSK....32
2.2 Математичне моделювання LDPC-кодування в радіолінії з методом модуляції QPSK....34
2.3 Порівняльний аналіз ефективності згорткового та LDPC-кодування....36
3 РОЗРОБКА VHDL-МОДЕЛІ LDPC-КОДЕКА....39
3.2 Розробка VHDL-моделі декодера....42
3.3 Перевірка працездатності системи....46
3.4 Синтез і імплементація пристрою....48
4 ДІАГНОСТИЧНИЙ ЕКСПЕРИМЕНТ....55
ВИСНОВКИ....61
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ....63
ДОДАТОК А....65
ДОДАТОК Б....70
ДОДАТОК В....81
ДОДАТОК Г....82

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

BER – bit error rate (бітова швидкість помилок)

CCSDS – Consultative Committee for Space Data Systems (Міжнародний Консультативний комітет із космічних систем передачі даних)

CPLD – complex programmable logic device (складний програмований логічний пристрій)

Eb/N0 – energy per bit to noise power spectral density ratio (Відношення енергії сигналу, що припадає на 1 біт прийнятого повідомлення (Eb), до енергетичної спектральної щільності шуму (N0))

FPGA – field-programmable gate array (різновид ПЛІС)

IP Core – intellectual property core – (ядро інтелектуальної власності)

LDPC – low-density parity-check code (код з низькою щільністю перевірок на парність)

QPSK – quadrature phase shift keying (квадратурна фазова маніпуляція)

VHDL – very high speed integrated circuits HDL (мова опису апаратури для швидкісних інтегральних схем)

ЕОМ – електронно обчислювальна машина

ПЛІС – програмовані логічні інтегральні схеми

САПР – система автоматизованого проектування

ВСТУП

В час сучасних технологій постійно збільшується об’єм інформації, який кудись постійно передається чи звідкись отримується. Зазвичай ця інформація передається через канали зв’язку, що не є ідеальними або близькими до ідеальних. У цих каналах постійно виникають перешкоди, викликані різними обставинами – наприклад, погодні явища, випромінювання від різних технічних засобів, відображення тощо. Ці перешкоди псують якість зв’язку і призводять до втрати корисної інформації.

Основним методом боротьби з цією проблемою є використання завадостійкого кодування. За допомогою додавання надлишкової інформації до корисної на стороні прийому даних можна дізнатися, чи були пошкоджені дані, а в деяких випадках можна навіть відновити первісну інформацію.

Одним із видів завадостійкого кодування є використання кодів з низькою щільністю перевірок на парність. Ці коди дозволяють знаходити і виправляти помилки у прийнятій інформації. Даний вид кодування відноситься до блокових кодів. Корисна інформація ділиться на інформаційні слова, до них додається блок перевірочної інформації, за допомогою якої і відбувається відтворення первинної інформації. В результаті кодування отримуються кодові слова, що й передаються через канали зв’язку.

Найчастіше розмір інформаційного слова у кодах з низькою щільністю перевірок на парність починається з декількох тисяч біт. Через це виникає проблема швидкості кодування інформації. Оскільки для обчислення блока перевірочної інформації необхідно провести операцію множення інформаційного слова на породжуючу матрицю, а розміри слів і матриць починаються з декількох тисяч, складність такого кодування може бути занадто велика, що призведе до неможливості передачі даних без затримки. Це може бути критичним для систем управління літальними засобами та іншою технікою, де швидкість реагування повинна бути дуже великою.

Якщо проводити кодування за допомогою процесорних ресурсів, то може виникнути ситуація, коли швидкість даних буде перевищувати швидкість кодування – наприклад, в системах передачі відеоданих з надвисокою якістю. Через це необхідні методи кодування, які будуть обходити цю проблему. Одним з таких методів є використання ПЛІС, оскільки засобами ПЛІС можна дуже сильно розпаралелювати обчислення. Для кодування інформації в кодах з низькою щільністю перевірок на парність використовують спеціально згенеровані породжуючі матриці, що мають певні закономірності та структуру. Це дозволяє оптимізувати процес множення інформаційного слова на цю матрицю.

Об'єкт дослідження: завадостійкі системи бездротової передачі даних.

Предмет дослідження: методи побудови завадостійких систем бездротової передачі даних, шляхом застосування блокового кодування з використанням ПЛІС.

Мета: розробити IP Core пристрою кодеку завадостійкої системи передачі даних на ПЛІС.

1 АНАЛІЗ ПРОБЛЕМИ. ПОСТАНОВКА ЗАДАЧІ ПРОЕКТУВАННЯ

1.1 Загальна характеристика методів кодування

Узагальнену схему системи передачі інформації можна представити у наступному вигляді (рис. 1.1). Джерело інформації видає послідовність знаків повідомлення, яка має бути отримана одержувачем каналом зв'язку. Повідомлення надходить на вхід кодера, який здійснює кодування. У кодері відбувається кодування інформації, що являє собою процес перетворення знаків дискретного повідомлення в кодові комбінації за наперед визначеним правилом. Іншими словами, інформаційний потік перетвориться на кодовий потік. Далі цей потік передається через канал зв'язку на декодер. Потрібно зауважити, що в процесі передачі в каналі зв'язку можливе виникнення перешкод, що може призвести до виникнення помилок в прийнятій інформації. У декодері відбувається процес декодування прийнятого кодового потоку в інформаційний наперед визначеним правилом. Після цього одержувач отримує вже інформаційний потік.

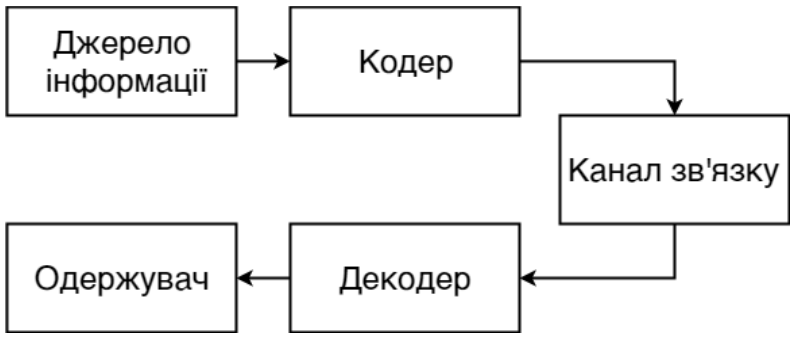


Рисунок 1.1 – Структурна схема системи передачі інформації

У сучасних системах передачі інформації широко використовуються три методи кодування: первинне, економне і завадостійке (рис. 1.2) [1, 2, 3].

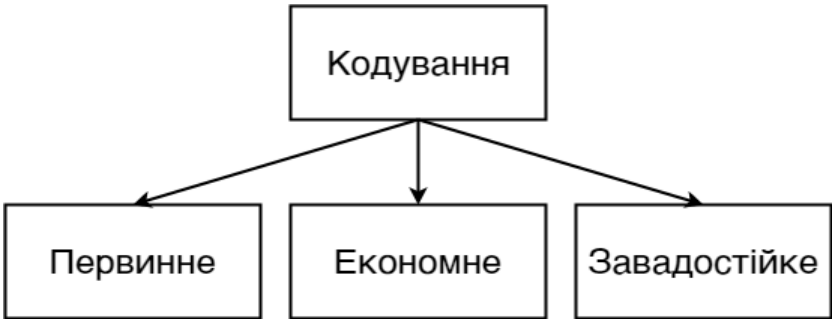


Рисунок 1.2 – Класифікація методів кодування

1.2 Первинне кодування

Даний вид кодування інформації дозволяє перетворювати дискретні величини у кодові комбінації. Наприклад, щоб надіслати текст українською мовою каналом зв'язку, 33 літери алфавіту можна закодувати двійковими числами. В даному випадку кожній букві можна поставити у відповідність шестизначне двійкове число. Якщо всі можливі кодові комбінації використовуються для передачі повідомлень, то код називається безнадмірним або первинним.

Прикладами таких кодів є міжнародний телеграфний код МТК-2, коди обміну інформацією ЯКІ-7, ЯКІ-8 та ін. Первинні коди зазвичай представляють у вигляді таблиць.

1.3 Економне кодування

Даний вид кодування дозволяє скоротити надлишковість коду – іншими словами, стиснути дані. Такий метод кодування враховує статистичні характеристики джерела повідомлень і дає можливість представити знаки повідомлення в середньому меншим числом кодових символів. Наприклад, значно скоротити надлишковість переданих повідомлень можна, якщо символам повідомлень, що найчастіше зустрічаються, поставити у відповідність коротші кодові комбінації, а менш імовірним символам –

довші. Операцію скорочення надлишковості джерела повідомлень називають економним кодуванням. Економне кодування широко використовується в цифрових системах передачі інформації для стиснення мовних, факсимільних і телевізійних повідомлень, які мають велику надлишковість. Первинне і економне кодування називають також кодуванням джерела.

1.4 Завадостійке кодування

В останні роки зусиллями вітчизняних та зарубіжних вчених освоєні методи коригуючого кодування, які забезпечують енергетичний виграш (6… 7 дБ) в каналах з постійними параметрами та Гаусовим шумом, що дозволило підвищити завадостійкість систем зв’язку з космічними апаратами.

Із теорії інформації відомо, що ефективні коди повинні бути достатньо довгими зі структурою, подібною структурі випадкового шуму. При цьому з ростом довжини коду складність алгоритму декодування (апаратна або програмна складність декодера) катастрофічно зростає. Тому практично важливим завданням спеціалістів з кодування залишається пошук та синтез кодів з високою коригуючою здатністю і, в цей же час, з прийнятною складністю реалізації декодера. Перспективними та рекомендованими до застосування CCSDS на сьогодні є згорткові коди (ЗК) та коди з малою густиною перевірок на парність (англ. – Low Density Parity Check Codes – LDPC codes) [4, 5].

Завадостійке кодування дозволяє виявляти і виправляти помилки, що виникають при передачі повідомлень каналом зв'язку. Завадостійке кодування здійснюється за рахунок введення до складу переданих кодових символів великого обсягу надлишкової інформації (наприклад, перевірочних символів). Операцію введення надмірності для підвищення завадостійкості каналу зв'язку називають канальним кодуванням. Завадостійке кодування знаходить широке застосування в системах зв'язку, в ЕОМ, в цифровій аудіо- і відеотехніці.

Теорія завадостійкого кодування базується на результатах досліджень, проведених Клодом Шенноном [1]. Він сформулював теорему для дискретного каналу з шумом: при будь-якій швидкості передачі двійкових символів, меншій, ніж пропускна здатність каналу, існує такий код, при якому ймовірність помилкового декодування буде як завгодно мала.

Побудова такого коду досягається ціною введення надлишковості. Тобто, застосовуючи для передачі інформації код, у якому використовуються не всі можливі комбінації, а тільки деякі з них, можна підвищити стійкість перед перешкодами прийому. Такі коди називають надлишковими або коригуючими. Коригувальні властивості надлишкових кодів залежать від правил побудови цих кодів і параметрів коду (тривалості символів, числа розрядів, надлишковості та інших).

В даний час найбільша увага приділяється двійковим рівномірним коригуючим кодам [6]. Вони мають добрі коригуючі властивості і їх реалізація відносно проста.

Найбільш часто застосовуються блокові коди [7, 8]. При використанні блокових кодів цифрова інформація передається у вигляді окремих кодових блоків рівної довжини. Кодування і декодування кожного блоку здійснюється незалежно один від одного, тобто кожній букві повідомлення відповідає блок з t символів.

Блоковий код називається рівномірним, якщо n (значність) залишається однаковою для всіх букв повідомлення.

Розрізняють роздільні і нероздільні блокові коди.

При кодуванні роздільними кодами кодові операції складаються з двох окремих частин: інформаційної та перевірочної. Інформаційні та перевірочні розряди у всіх кодових комбінаціях роздільного коду займають одні й ті ж позиції.

При кодуванні нероздільними кодами розділити символи вихідної послідовності на інформаційні та перевірочні неможливо.

Безперервними називаються такі коди, в яких введення надлишкових символів в кодованих послідовностях інформаційних символів здійснюється безперервно, без поділу її на незалежні блоки.

1.4.1 Згорткове кодування

Згорткові коди засновані на перетворенні нескінченної вхідної послідовності бінарних символів, в якій на кожен символ вхідної послідовності припадає понад одного символа. Збільшення кількості бінарних символів, які передаються, при використанні згорткових кодів характеризується відносною швидкістю коду, іноді званою просто швидкістю коду:

де $Q_{ВХ}$ – швидкість передачі двійкових символів на вході кодера, біт/сек;

$Q_{ВИХ}$ – швидкості передачі двійкових символів на виході кодера, біт/сек;

k – число біт вхідної послідовності;

n – біти вихідної послідовності.

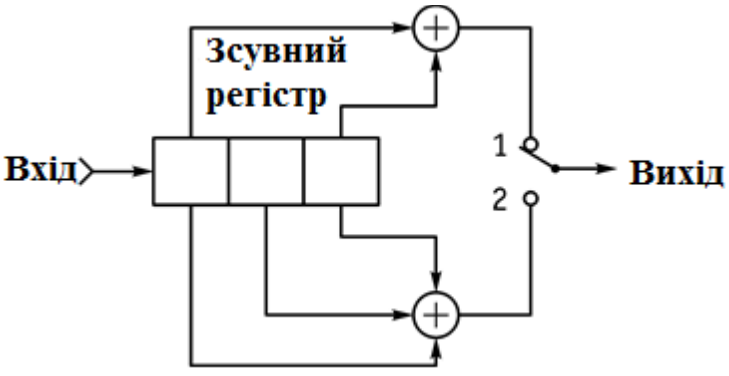


Рисунок 1.3 – Згортковий кодер

Приклад згорткового кодера показаний на рисунку 1.3 [9]. Кодер містить трьохрозрядний зсувний регістр, на вхід якого надходить вхідна послідовність двійкових символів. На кожний такт біти в комірках регістру зсуваються на крок праворуч, при цьому черговий біт вхідної послідовності записується в першу комірку, а біт із крайньої правої комірки викидається. Виходи розрядів регістру підключені до входів двох суматорів по модулю 2. Вихідна послідовність двійкових символів формується за допомогою комутатора, який на кожен такт вхідної послідовності спочатку передає на вихід біт з верхнього суматора (точка 1), а потім – біт з нижнього суматора (точка 2). Таким чином, на кожен біт вхідної послідовності формується два біти вихідної послідовності, тобто швидкість цього коду R=1/2.

Важливий параметр згорткових кодів – кодове обмеження, яке позначається K. Цей параметр показує, скільки груп по k біт міститься в зсувному регістрі, а отже, одночасно приймає участь у формуванні біт вихідної послідовності. В прикладі, який розглядається, k=1, K=3.

Робота згорткового кодера пояснюється ґратчастою діаграмою (рис. 1.4) [10]. Кожен двійковий символ вхідної послідовності перетворюється в пару двійкових символів вихідної послідовності, яка визначається бінарним символом вхідної послідовності і поточним станом кодуючого пристрою. Таких станів може бути чотири: 00, 01, 10 і 11.

Кожному стану відповідає горизонтальний ряд вузлів на діаграмі. З кожного вузла, відповідного поточному стану, виходять дві гілки. Верхня (рис. 1.4) гілка відповідає двійковому символу «0» вхідної послідовності, а нижня гілка – двійковому символу «1». Пара цифр у кожної гілки показує пару двійкових символів вихідної послідовності, які формуються при даному переході кодуючого пристрою з одного стану в інший. Отримана структура переходів кодуючого пристрою утворює ґрати, тому такі коди часто називаються ґратчастими.

Якщо вхідна послідовність складається лише з нулів, то і вихідна послідовність також містить виключно нулі. Нехай вхідна послідовність містить один одиничний біт, а інші – дорівнюють нулю: «…001000…».

За допомогою структурної схеми кодера та ґратчастої діаграми побудуємо вихідну послідовність: «…00 00 11 01 11 00 …»

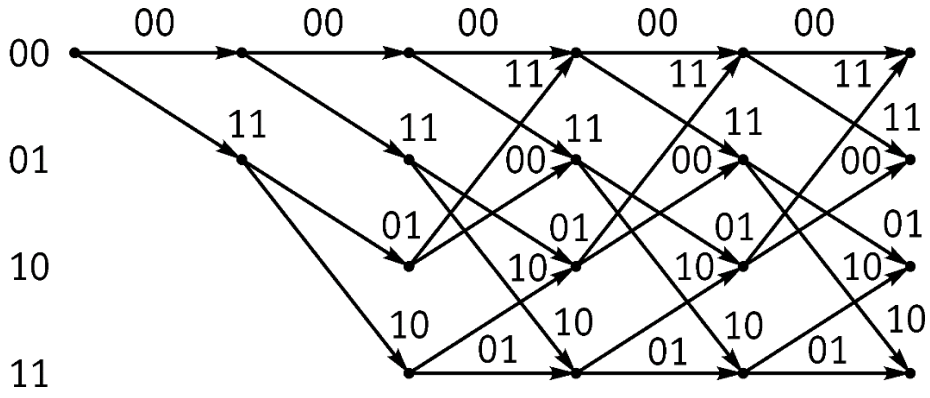


Рисунок 1.4 – Ґратчаста діаграма згорткового коду

Ця послідовність містить 5 одиниць, тому відстань Хеммінга між нею та послідовністю, що складається лише з нулів, дорівнює 5. Вивчення властивостей згорткового коду, який розглядається, показує, що відстань Хеммінга між вихідними послідовностями, що виходять із різноманітних вхідних послідовностей, і тими, що не містять помилок, виявляється не менше 5. Взагалі відстань між вихідними послідовностями зростає зі зменшенням R та зі збільшенням K.

Для декодування згорткових кодів найчастіше застосовується алгоритм Вітербі, який дозволяє із безлічі можливих шляхів, які призводять до останнього символу прийнятої послідовності, яка декодується, обрати відносно невелику кількість шляхів, які є найбільш правдоподібними, та визначити правильне значення символу вихідної послідовності.

Алгоритм Вітербі – це алгоритм максимальної правдоподібності для декодування згорткових кодів, заснований на використанні імовірнісних характеристик сигналів, які приймаються. Декодування може мати як жорстке, так і м’яке рішення. У випадку жорсткого декодування рішенням

про прийнятий сигнал вибирається кодове слово, яке відрізняється від прийнятого у найменшому числі символів. При м’якому рішенні використовується інформація про апостеріорну імовірність символів, які приймаються. Одним з переваг алгоритму є те, що складність реалізації декодера з м’яким рішенням мало відрізняється від складності реалізації декодера з жорстким рішенням. Недолік – експоненціальне зростання складності декодера в залежності від довжини кодового обмеження згорткового коду.

Процес декодування полягає в стеженні за кодовими ґратами станів шляхів з максимальною апостеріорною імовірністю з використанням для оцінки відстаней, наприклад, метрики Хеммінга. На окремому кроці декодування в кожному із станів ґратчастої діаграми здійснюється обчислення метрик гілок за канальним символом, який приймається, складання метрик попередніх станів з метриками відповідних гілок, порівняння метрик шляхів, які входять в даний вузол ґрат, і вибір шляхів з найменшими метриками, величини яких використовують в якості станів на наступному кроці декодування. При рівності метрик порівнюваних шляхів, наприклад, одного з двох шляхів, вибір роблять випадково. На кожному кроці в результаті порівняння половина можливих шляхів відкидається і надалі не використовується.

Інша половина утворює продовження шляхів для наступного кроку декодування. З кожного кроку на наступному кроці знову з’являються варіанти продовження, досліджувані з використанням тієї ж самої послідовності обчислень.

Таким чином, в процесі декодування за кодовими ґратами простежується шлях, який має мінімальну відстань від шляху, який генерується в кодері. Алгоритм Вітербі забезпечує високу завадостійкість і є достатньо простим при технічній реалізації. Приклад такої реалізації показаний на рисунку 1.5.

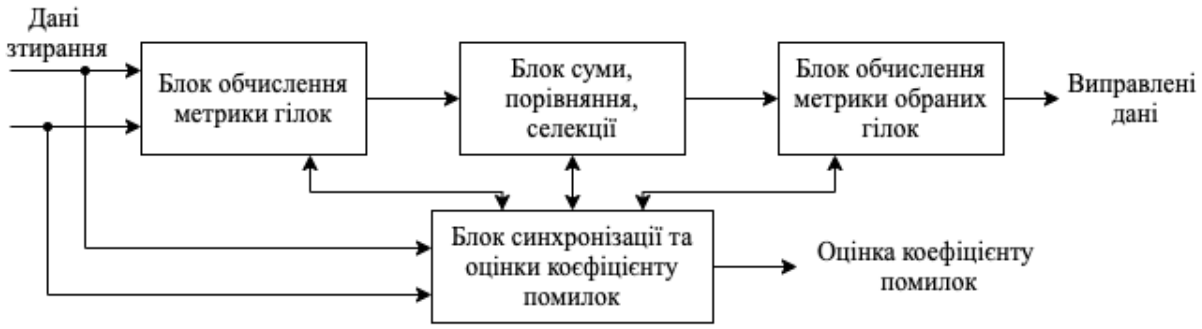


Рисунок 1.5 – Структурна схема декодера Вітербі

1.4.2 Турбокоди

Основна ідея турбокодування полягає в використанні ітераційної процедури декодування за допомогою декодерів з «м'якими» рішеннями на виході. Укрупнена блок-схема кодера турбокодів приведена на рисунку 1.6 [11]. Послідовність інформаційних символів на вході кодера розбивається на блоки фіксованої довжини. Інформаційні символи надходять у кодер 1, який формує послідовність перевірочних символів П1. Одночасно цей самий блок інформаційних символів через переміжник надходить на другий кодер, який формує послідовність перевірочних символів П2. Перевірочні символи П1, П2 і відповідний інформаційний блок попадають на мультиплексор і передаються каналом зв'язку.

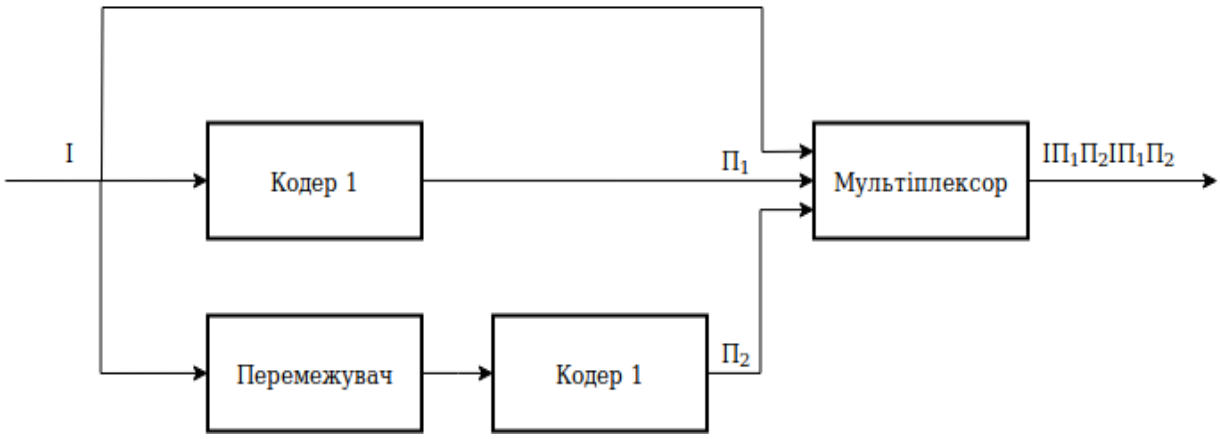


Рисунок 1.6 – Кодер турбокоду

Сигнал на вході декодера за допомогою схеми синхронізації розділяється на інформаційну і перевірочну частини і у вигляді чисел («м'які» рішення на вході) надходить на два декодери. Особливістю декодерів є «м'яке» рішення на виході. Це означає, що декодер не виносить остаточного рішення про те, яку конкретно послідовність інформаційних символів було передано, а формує послідовність чисел, кожне з яких пропорційно ймовірності того, що відповідний інформаційний символ декодовано правильно. Процедура декодування носить ітераційний характер, причому при кожній ітерації декодер 1 використовує «м'які» рішення декодера 2 для корекції своїх власних рішень і навпаки. Після виконання заданого числа ітерацій виноситься остаточне рішення про отримання прийнятого інформаційного блоку.

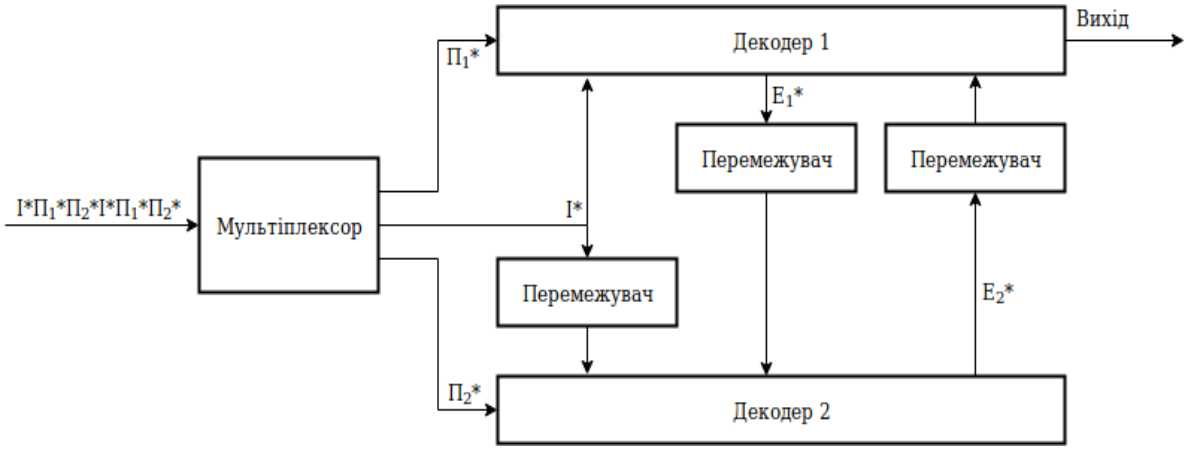


Рисунок 1.7 – Декодер турбокоду

Оскільки алгебраїчні декодери не допускають «м'яких» рішень на виході, основними кандидатами для використання в описаній схемі кодування-декодування виявилися згорткові коди. Незважаючи на умовну перспективу турбокодування згортальних кодів, названих паралельно-каскадними згортковими кодами, їх використання не привело до очікуваних результатів. По-перше, складність реалізації алгоритму Вітербо з «м'якими» рішеннями на виході робить його широке практичне використання вельми

сумнівним.

По-друге, якщо при досить великих можливостях помилкового прийому двійкового символу (близько 10 ~ 5) коди показували свою високу ефективність – програш потенційній межі Шеннона склав величину всього близько 1 дБ, – то при більш реальних менших можливостях помилки експоненціальні залежності від ставлення сигнал/шум вигиналися вгору. Через це паралельно-каскадні згорткові коди залишилися більш поширеними у вжитку.

М'які рішення щодо інформаційних символів, отримані в результаті попередньої ітерації, використовуються в якості вхідних величин для подальшої, що нагадує розгадування кросворду. При збільшенні числа ітерацій зростає енергетичний виграш коду, але і збільшується затримка завдяки роботі декодера.

Матричні турбокоди виявилися ефективнішими за традиційні каскадні коди.

1.4.3 LDPC-кодування даних

LDPC-коди відносяться до блокових кодів. Основна відмінність блокових кодів від згортальних полягає в тому, що для кодування інформації інформаційний потік ділиться на блоки певної довжини, які називаються інформаційними словами, і до кожного з цих блоків додається блок перевірочних даних (надлишкова частина). В результаті цієї операції отримується кодове слово. Для уявлення блокового кодування зазвичай використовується наступне відображення c = i * G, де i – це інформаційне слово заздалегідь відомої довжини, а c – отримане кодове слово. G являє собою матрицю, при множенні на яку інформаційного слова виходить кодове слово. Одним із головних завдань класичної теорії завадостійкого кодування є наступне: знайти таку породжуючу матрицю G, яка забезпечувала б максимально можливе значення мінімальної відстані Хеммінга коду. Під мінімальною відстанню Хеммінга коду мається на увазі мінімальна кількість

біт, в яких є відмінності в будь-яких двох кодових словах серед усього кодового простору. Це викликано тим, що цей показник напряму впливає на коригувальну здатність коду, тобто здатність коду виправляти помилки, викликані неідеальністю каналу передачі даних. Відповідно, чим більше мінімальна відстань Хеммінга для коду, тим краще стійкість коду.

Також, крім породжуючої матриці коду G, для кожного блокового коду існує перевірочна матриця H. Вона відповідає наступній рівності: $c * H^T = 0$, де H^T – транспонована перевірочна матриця.

Однією з основних відмінностей LDPC-кодів від звичайних лінійних блокових кодів є те, що при генерації зв'язки породжуючої і перевірочної матриці немає завдання максимізації мінімальної відстані Хеммінга. Забезпечення оптимальних параметрів завадостійкості цих кодів досягається за рахунок того, що загальна кількість слів з мінімальним значенням відстані Хеммінга щодо всього простору слів дуже мала, а з іншого боку, всі інші слова знаходяться на досить великій відстані. З цієї причини більшість LDPC-кодів мають невелике значення мінімальної відстані Хеммінга. З цього випливає той факт, що LDPC-коди найбільш ефективні на досить великих розмірах кодових слів.

Також відмінністю LDPC-кодів від звичайних блокових кодів є те, що їх перевірочні матриці є розрідженими, тобто містять малу кількість одиниць. Універсального правила, за яким можна визначити, чи є матриця розрідженою, немає. Все залежить від методу генерації перевірочної і породжуючої матриць. У більшості випадків їх кількість не буде перевищувати одиниць або частки одиниць відсотків від загальної кількості елементів. Як приклад можна розглянути стандарт супутникового зв'язку DVB-S2. У цьому стандарті використовується LDPC-код, в якому кількість ненульових елементів від загального числа елементів дорівнює 0,02%.

Незважаючи на складність реалізації кодування і декодування використовуючи LDPC-коди, в порівняння з турбокодами, в телекомунікаційній галузі LDPC-коди є досить популярними. Це пов'язано з

тим, що для використання LDPC-кодів відсутні юридичні перешкоди, вони не захищені патентами і можуть вільно використовуватися. У 2003 році LDPC-код замінив турбокод і став частиною стандарту супутникової передачі даних для цифрового телебачення DVB-S2. Аналогічна заміна сталася і в стандарті DVB-T2, який використовується для наземного телевізійного мовлення.

1.4.3.1 Способи опису LDPC-кодів. Як було зазначено раніше, систематичні LDPC-коди повністю описуються перевірочною матрицею коду, яка є розрідженою. На рисунку 1.8 наведено приклад виду перевірочної матриці (500,250) LDPC-коду [12].

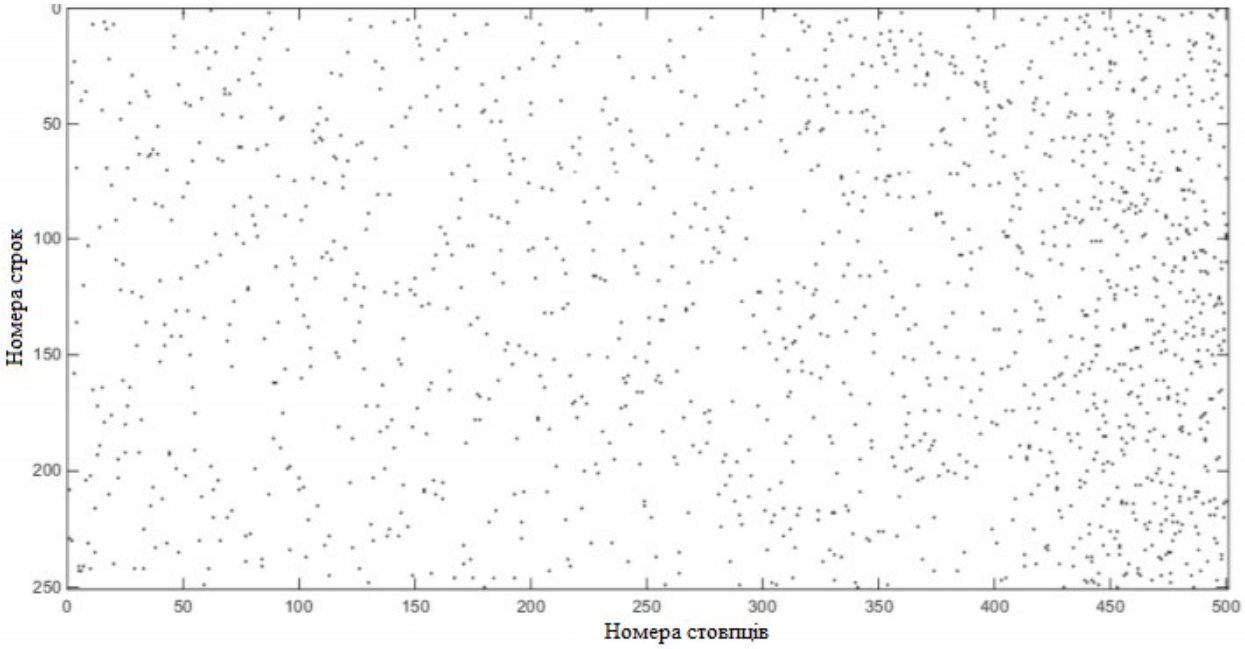


Рисунок 1.8 – Перевірочна матриця нерегулярного (500, 250) LDPC-коду (чорними точками відображені ненульові елементи матриці)

Загальна кількість елементів в матриці з рисунка 1.8 становить 125000, при цьому кількість ненульових елементів всього 1493, що становить 1,1% від загального числа. У загальному випадку перевірочна матриця використовується для кодування і декодування LDPC-коду. На її підставі може бути побудовано кодер і декодер коду. Також перевірочна матриця визначає характеристики завадостійкості коду. При цьому визначальним

фактором є не конкретний вид матриці (в загальному випадку вона є псевдовипадковою), а розподіл ненульових елементів по її рядках і стовпцях.

1.4.3.2 Подання кодів у вигляді графів Таннера. Крім матричного і поліноміального опису LDPC-кодів широко поширений опис за допомогою графа Таннера [12, 13].

Граф Таннера – це двочастковий граф, який показує перевірочні співвідношення, що описують завадостійкий код. Двочастковий граф являє собою безліч вершин, які можна розбити на дві частини таким чином, що кожне ребро графа буде з'єднувати вершину з однієї частини з однією або декількома вершинами з іншої частини. Для графа Таннера, вершини, які знаходяться з одного боку, називаються вузлами змінних і зображуються кружечком. Вершини, які знаходяться з іншого боку, називаються вузлами перевірки парності і зображуються квадратиками. Ребра графа зображуються лініями і називаються шляхами. На рисунку 1.9 показаний приклад того, як зображується граф Таннера.

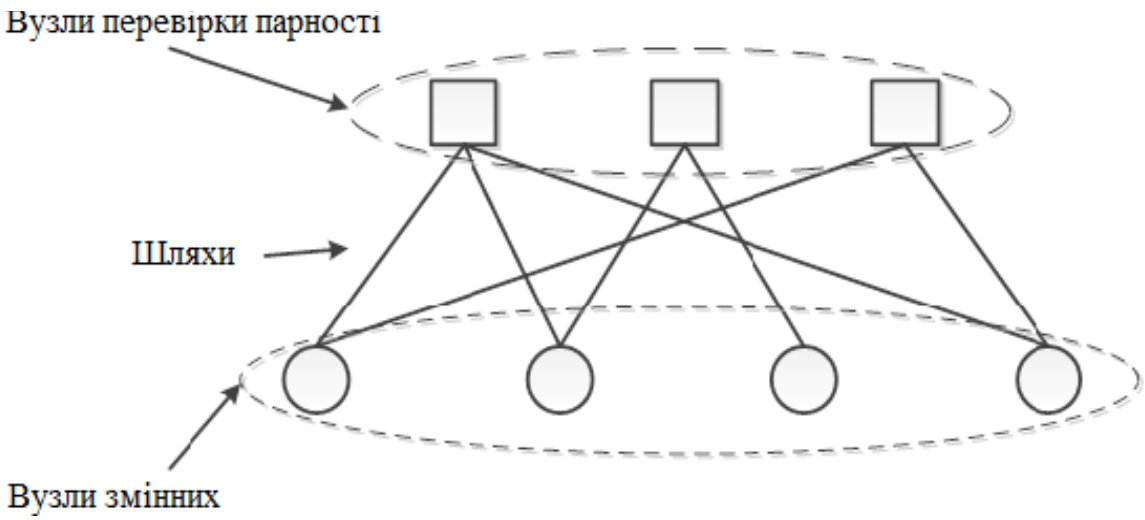
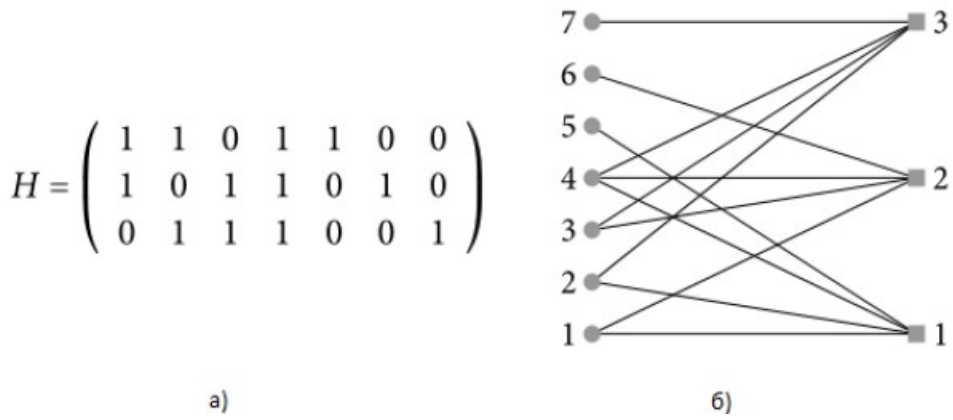


Рисунок 1.9 – Приклад графа Таннера

У такому випадку може бути наочно зображений вид розподілу шляхів, які сполучають вузли змінних і вузли перевірки парності, а також їх кількість.

Граф Таннера може бути побудовано на підставі перевірочної матриці. Для LDPC-коду, який описується перевірочної матрицею H розміру m * n, може бути побудований двочастковий граф. Цей граф буде містити n вузлів змінних і m вузлів перевірки парності. Вузли змінних і вузли перевірки парності з'єднуються шляхами наступним чином: вузол змінної j з'єднується з вузлом перевірки парності, і в тому випадку, якщо $H_{ji} \neq 0$, тобто якщо в j-му стовпці на і-му рядку знаходиться ненульовий елемент. Один і той самий код може бути описано кількома перевірочними матрицями, також він може бути описаний і декількома графами Таннера, але з точки зору алгоритму декодування ці графи Таннера не будуть еквівалентними.

За допомогою графа Таннера може бути описано будь-які лінійні блокові коди. На рисунку 1.10 представлено перевірочну матрицю для (7, 4) коду Хеммінга і відповідний їй граф Таннера [14].



а) перевірочна матриця (7, 4) коду Хеммінга;

б) граф Таннера для (7, 4) коду Хеммінга.

Рисунок 1.10 – Лінійні блокові коди.

Граф Таннера найбільш часто використовується при описі алгоритму декодування LDPC-кодів.

1.4.3.3 А л г о р и т м к о д у в а н н я. Кодування посилання виконується простим перемноженням інформаційної послідовності u на генераторну матрицю LDPC-коду G. Це можна вважати процесом, коли вихідне посилання в кодовому слові залишається незмінним, а до нього додається вектор з перевірочних біт [15].

1.4.3.4 А л г о р и т м и д е к о д у в а н н я. Алгоритми декодування LDPC-кодів можна розділити на дві групи: алгоритми декодування жорстких і м'яких рішень демодулятора. Серед алгоритмів декодування жорстких рішень найбільш ефективним є послідовний алгоритм інверсії біт (BitFlipping). Блок-схема алгоритму представлена на рисунку 1.11.

На першому кроці алгоритму здійснюється ініціалізація максимальної кількості ітерацій. У разі, якщо поточне число ітерацій алгоритму досягне максимального значення, і множення декодованого слова на перевірочну матрицю не дорівнюватиме нулю, то це буде означати відмову декодування.

На другому кроці здійснюється розрахунок синдрому $S = r * H^T$, де r – це кодове слово в двійковому вигляді з бінарними помилками.

На третьому кроці алгоритму формується тимчасова матриця $H_{тм}$ шляхом поелементного множення кожного стовпця перевірочної матриці H на вектор синдромів S. Потім рядки тимчасової матриці підсумовуються, і знаходиться позиція максимального значення сумарного рядка. У разі, якщо позицій, відповідних максимальному значенню, кілька, то вибирається одна з них випадковим чином. Ця позиція відповідає біту кодового слова, для якого найбільшу кількість перевірок на парність не було виконано і такий біт є найбільш недостовірним. Відповідний цій позиції кодовий біт інвертується, і алгоритм переходить до наступного кроку. У разі, якщо максимальне значення сумарного рядка дорівнює нулю, то це говорить про відсутність помилок в кодовому слові, що декодується. Тоді алгоритм завершується.

На останньому кроці алгоритму здійснюється перевірка поточного числа ітерацій. Якщо поточне значення дорівнює максимальному, то алгоритм декодування завершується відмовою. В іншому випадку алгоритм

переходить до другого кроку. Крім послідовного алгоритму інверсії біт існує паралельний алгоритм.

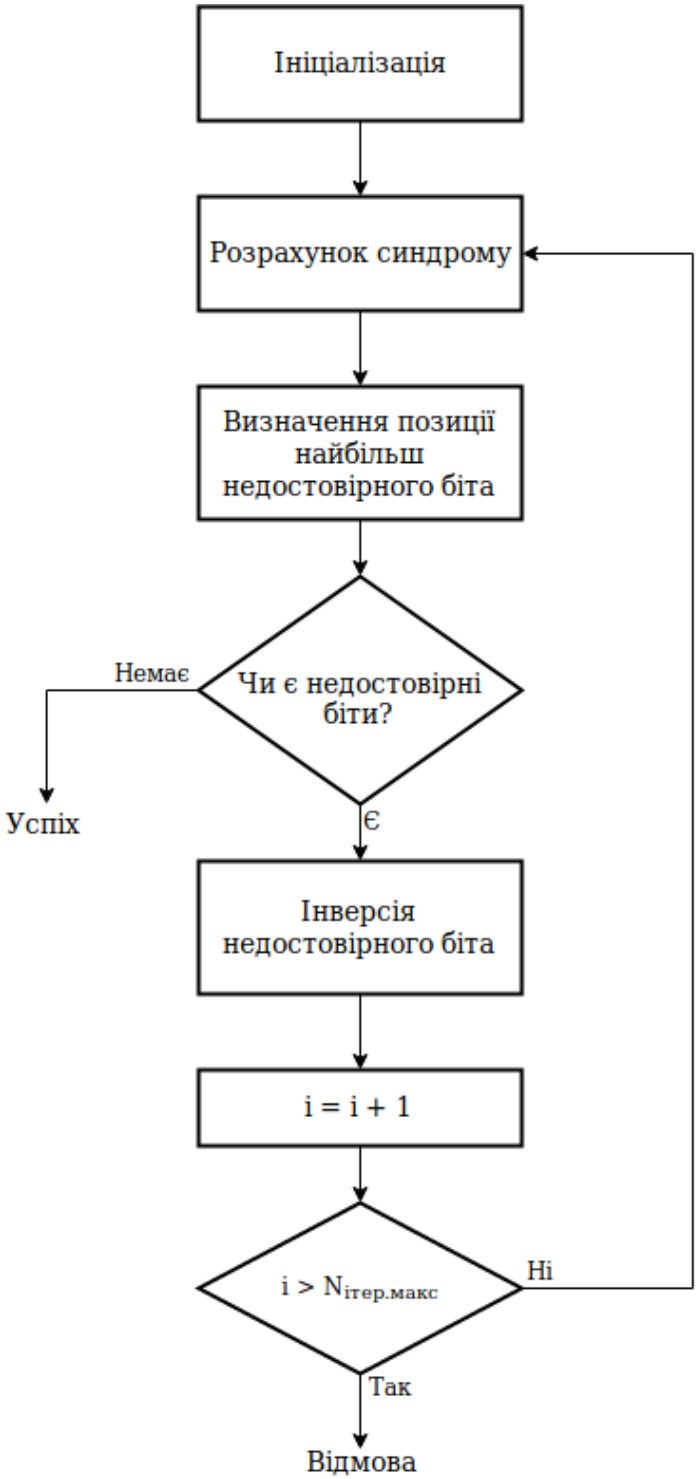


Рисунок 1.11 – Блок-схема послідовного алгоритму інверсії біт

Однією з переваг LDPC-кодів в порівнянні з іншими видами кодування є ефективний алгоритм декодування з лінійною складністю від довжини кодового слова – алгоритм пересилання повідомлень (алгоритм поширення довіри). Цей алгоритм дозволяє здійснювати ефективне завадостійке декодування м'яких рішень демодулятора. Опис алгоритму декодування здійснюється з використанням графа Таннера. На рис. 1.12 наведена блок-схема алгоритму.

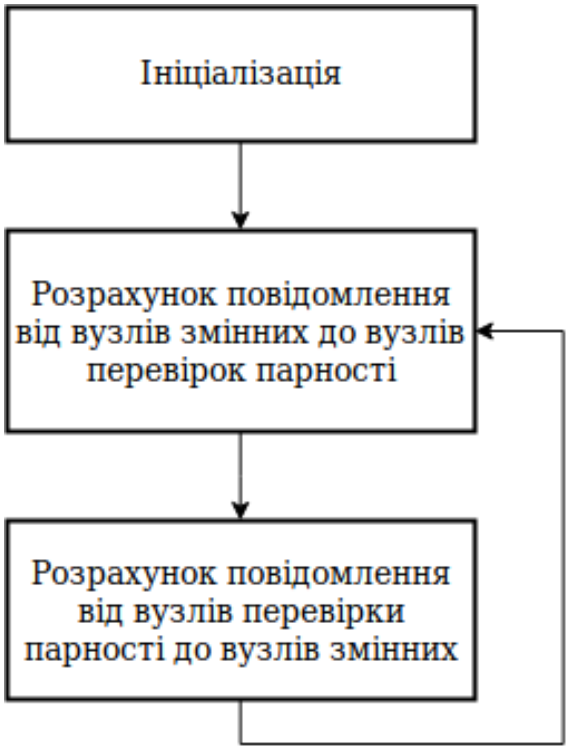


Рисунок 1.12 – Блок-схема алгоритму декодування м’яких рішень демодулятора

Першим кроком алгоритму є ініціалізація. В процесі ініціалізації:

1) визначається максимальне число ітерацій алгоритму;

2) здійснюється розрахунок логарифму відношення правдоподібності біт декодуємого сигналу.

Другий крок алгоритму полягає в розрахунку повідомлень вздовж ребер графа Таннера від вузлів змінних до вузлів перевірки парності.

Повідомленням $\mu_{i,j}$ від і-го вузла змінних до j-ого вузлу перевірки парності називається числове значення, що розраховується за формулою $\mu_{i,j} = L_i + \sum_{i,j \neq i} Y_{i,j}$, де L_i – логарифм відношення правдоподібності для і-го біта сигналу, а $Y_{i,j}$ –повідомлення від j-го вузла перевірки парності до і-ого вузлу змінних (на першій ітерації дорівнюють нулю).

Третій крок алгоритму полягає в розрахунку повідомлень від вузлів перевірки парності до вузлів змінних. Повідомленням $Y_{i,j}$ від j-го вузла перевірки парності до і-ого вузлу змінних називається числове значення, що розраховується за формулою:

Після третього кроку алгоритм переходить до другого і так доти, доки не буде досягнуто максимальну кількість ітерацій. Після виконання всіх ітерацій алгоритму необхідно прийняти жорсткі рішення по числових значеннях, що містяться у вузлах змінних. В результаті буде отримано двійкове кодове слово. Якщо в результаті множення цього слова на транспоновану перевірочну матрицю коду буде отримано нульовий вектор, то це означає успіх декодування (відсутність помилок в декодованому кодовому слові). В іншому випадку відбувається відмова декодування.

Вищеописаний алгоритм використовується для декодування LDPC-кодів [16]. Його обчислювальна складність залежить від кількості шляхів в графі Таннера або, що еквівалентно, від числа ненульових елементів в перевірочній матриці коду.

Також слід зазначити, що для декодування LDPC-кодів можуть застосовуватися і інші алгоритми декодування лінійних блокових кодів, але ефективність їх буде свідомо гірше.

1.4.3.5 Основні види LDPC-кодів. Існує велика безліч видів LDPC-кодів. На рисунку 1.13 наведено дерево, яке показує основні види LDPC-кодів.

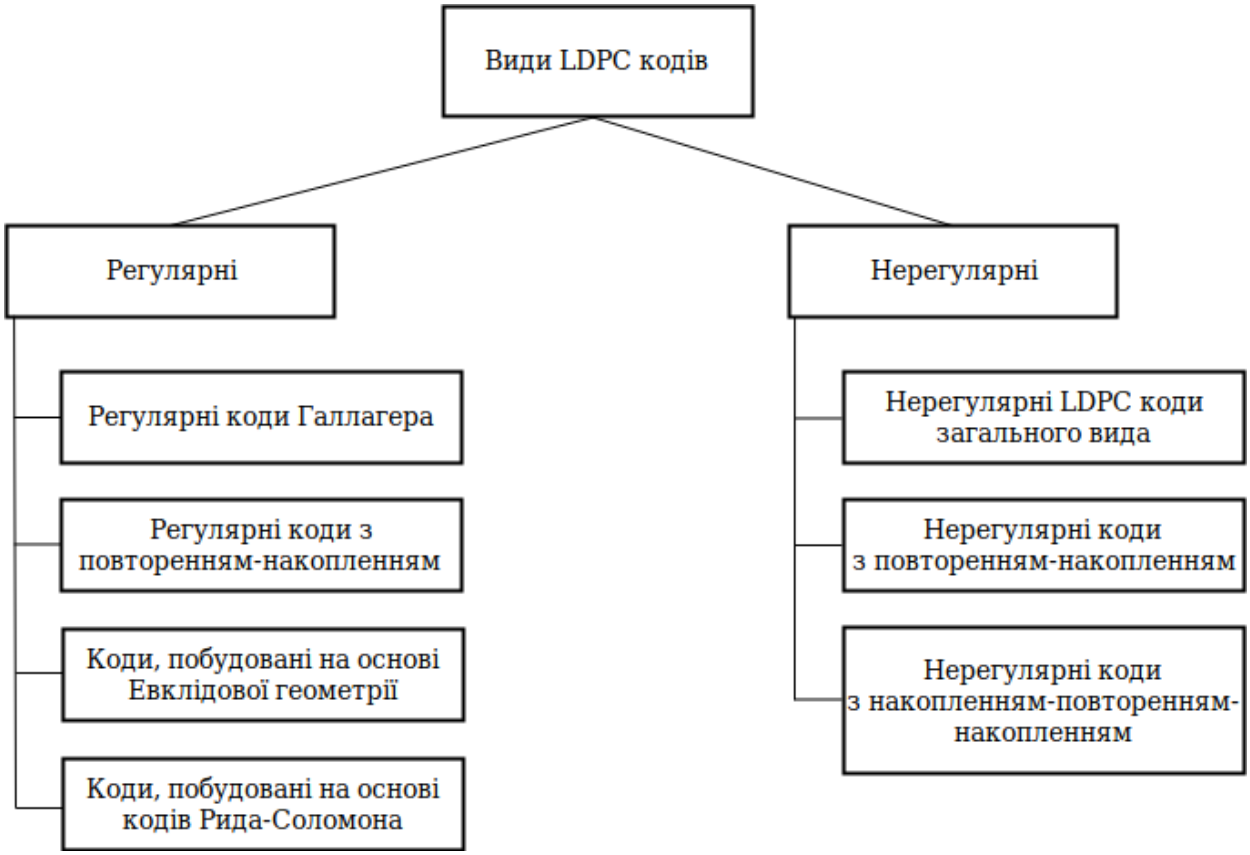


Рисунок 1.13 – Основні види LDPC-кодів

Як показано на рисунку 1.13, LDPC-коди діляться на два основні види – регулярні і нерегулярні. Регулярні LDPC-коди описуються перевірочною матрицею, у якій кількість ненульових елементів в рядках (стовпчиках) є постійною величиною. Перевагою таких кодів є простота синтезу і можливість отримати хороші характеристики завадостійкості при відносно невеликих довжинах кодового слова. Недоліком таких кодів є програш у завадостійкості в порівнянні з нерегулярними кодами при великих довжинах кодового слова і мала гнучкість вибору швидкості коду.

Нерегулярні коди описуються перевірочною матрицею, в якій ненульові елементи в рядках і стовпцях не є постійною величиною. Такі коди

можуть бути синтезовані для будь-якої швидкості коду і володіють найкращими характеристиками завадостійкості.

Найбільш поширеними регулярними кодами є регулярні коди Галлагера, коди з повторенням накопиченням (RepeatAccumulate, RA), коди, побудовані на підставі Евклідової геометрії (ЕГ-коди) і коди, побудовані на підставі кодів Ріда-Соломона (RSLDPC). Серед нерегулярних кодів найбільшого поширення набули нерегулярні коди з повторенням накопиченням (Irregular Repeat-Accumulate, IRA) і нерегулярні коди з накопиченням повторенням накопиченням (Accumulate-Repeat-Accumulate, ARA).

1.4.3.6 П о б у д о в а к о д і в. В даний час використовуються два принципи побудови перевірочної матриці коду. Перший заснований на генерації початкової перевірочної матриці за допомогою псевдовипадкового генератора. Коди, отримані таким способом, називають випадковими (англ. Random-like codes). Другий – використання спеціальних методів, заснованих, наприклад, на групах і кінцевих полях. Коди, отримані цими способами, називають структурованими. Кращі результати щодо виправлення помилок показують саме випадкові коди, однак структуровані коди дозволяють використовувати методи оптимізації процедур зберігання, кодування і декодування, а також отримувати коди з більш передбачуваними характеристиками.

У своїй роботі Галлагер вважав за краще за допомогою генератора псевдовипадкових чисел створити початкову перевірочну матрицю невеликого розміру з заданими характеристиками, а далі збільшити її розмір, дублюючи матрицю і використовуючи метод перемішування рядків і стовпців для позбавлення від циклів певної довжини.

У 2003 році Джеймсом МакГованом і Робертом Вільямсоном був запропонований спосіб видалення циклів з матриць LDPC-кодів, в зв'язку з чим стало можливим на початку згенерувати матрицю із заданими

характеристиками (n, j, k), а потім видалити з неї цикли. Так відбувається в схемі Озарова-Вайнера.

У 2007 році в журналі «IEEE Transactions on Information Threory» було опубліковано статтю про використання кінцевих полів для побудови квазі-циклічних LDPC-кодів для каналів з адитивним білим Гауссовим шумом і двійковими каналами зі стиранням.

1.5 Обґрунтування вибору завадостійкого кодування

В результаті порівняльного аналізу різноманітних кодів, які використовуються в бездротових системах зв’язку, було обрано два види завадостійкого кодування для подальшого моделювання, а саме: згорткове кодування з (171,133) породжувальним поліномом і LDPC-кодування з довжиною інформаційного слова 128, 1024, 4096 біт, різними швидкостями коду та bit-flipping алгоритмом для декодування.

Ці коди дозволять реалізувати завадостійку систему передачі даних з найбільшою якістю передачі даних і завадостійкістю. Також ці коди можуть бути реалізовані на ПЛІС, через що система передачі даних не буде вносити значну затримку через витрачений час на кодування та декодування інформації.

2 МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ РАДІОЛІНІЙ З РІЗНИМИ МЕТОДАМИ ЗАВАДОСТІЙКОГО КОДУВАННЯ

2.1 Математичне моделювання згорткового кодування в радіолінії з методом модуляції QPSK

Для моделювання згорткового кодування в радіолінії було побудовано функціональну модель в програмному пакеті MATLAB Simulink (рис. 2.1).

Щоб отримати показники ймовірності невиправленої похибки, в блоці binary symmetric channel змінювалося значення імовірності помилки в каналі [17]. Отримані результати показано на рисунку 2.2.

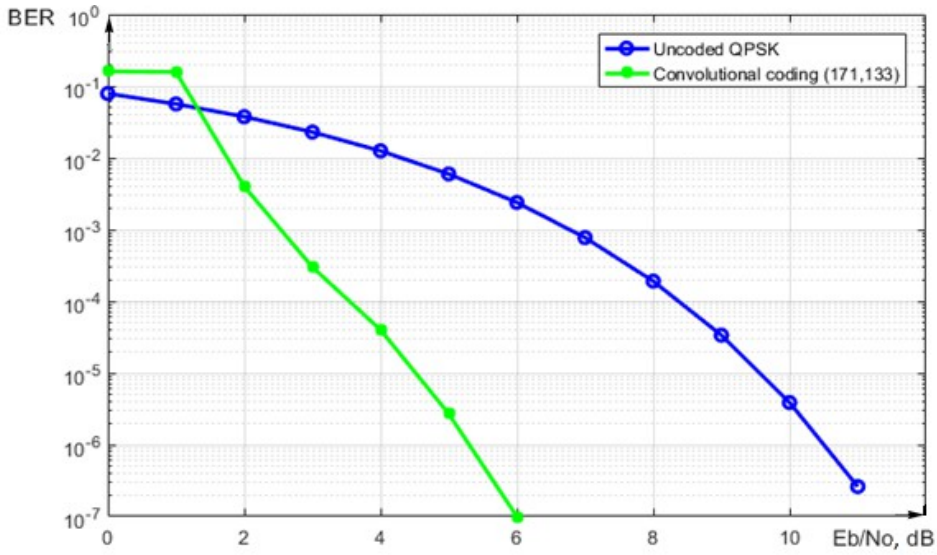


Рисунок 2.2 – Графік залежності ймовірності бітової помилки при згортковому кодуванні

Із залежності на рисунку 2.2 видно, який енергетичний виграш можна отримати, використавши згорткове кодування для передавання даних з метою підвищення завадостійкості. Тобто, додавши згорткове кодування, можна доволі значно зменшити енергетику сигналу, але при цьому імовірність невиправленої помилки залишиться тією ж.

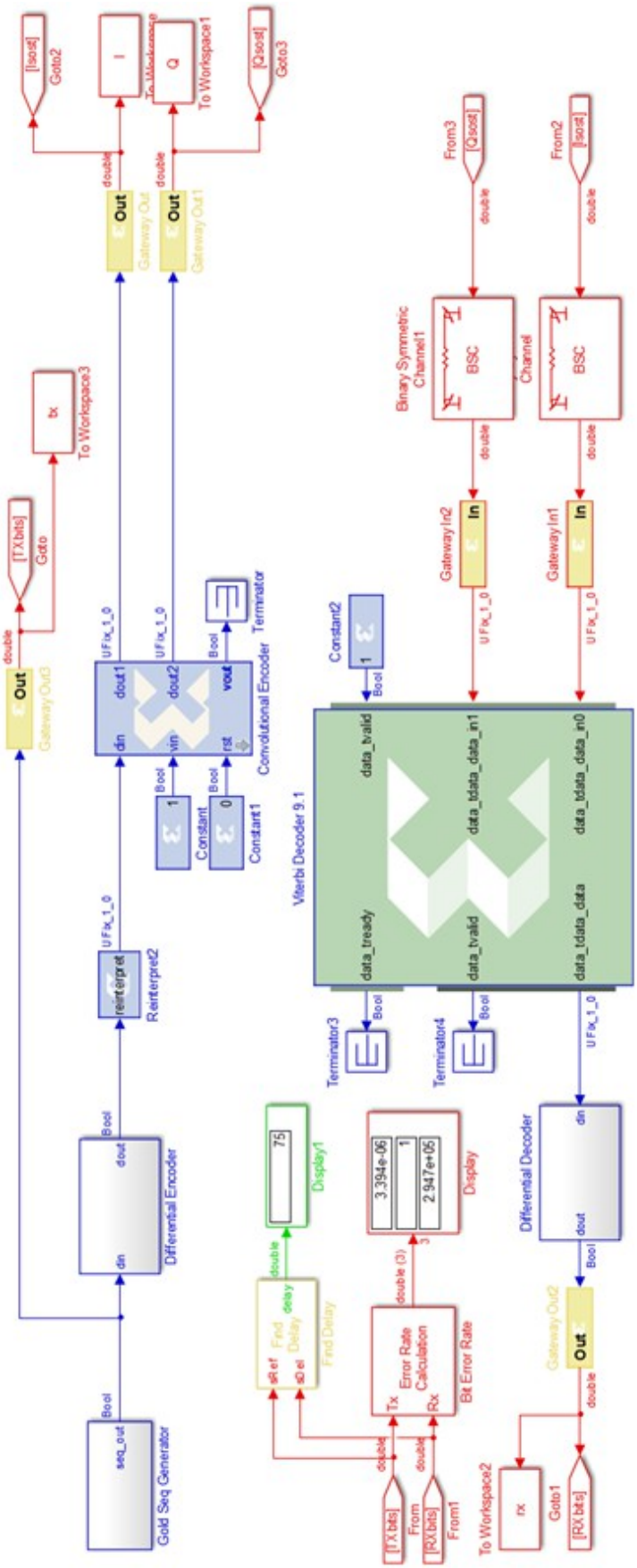


Рисунок 2.1 – Схема згорткового кодування та декодування в радіолінії з шумами

2.2 Математичне моделювання LDPC-кодування в радіолінії з методом модуляції QPSK

Моделювання завадостійкого LDPC-кодування даних в радіолінії виконувалося за допомогою програмного пакету MATLAB.

Щоб закодувати інформаційну послідовність, потрібно перемножити цю послідовність з генераторною матрицею G обраного варіанту пари LDPC-матриць (перевірочна та генераторна).

Перевірочну та генераторну матриці для довжин інформаційного слова 128, 1024 та 4096 біт було рекомендовано CCSDS. Генераторну та перевірочну матриці для довжини інформаційного слова 1024 біти показано на рисунках 2.3 та 2.4 (чорні точки – ненульові елементи матриці).

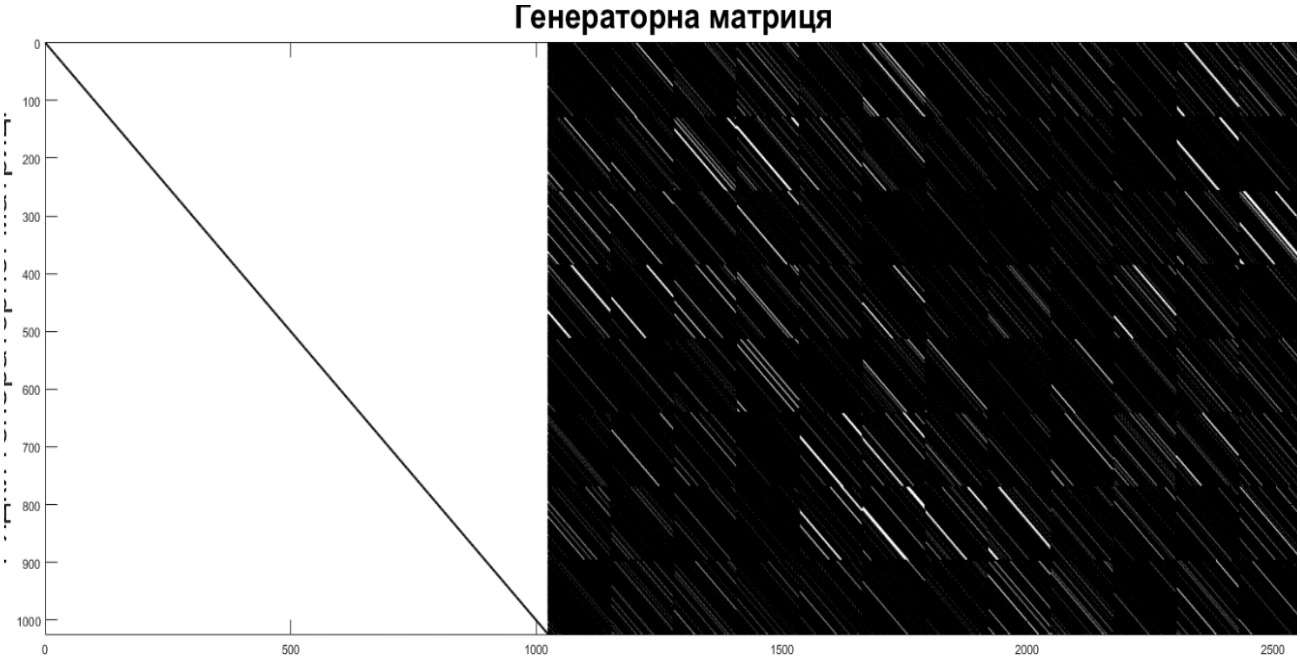


Рисунок 2.3 – Генераторна матриця LDPC-коду для довжини інформаційного слова 1024 біт

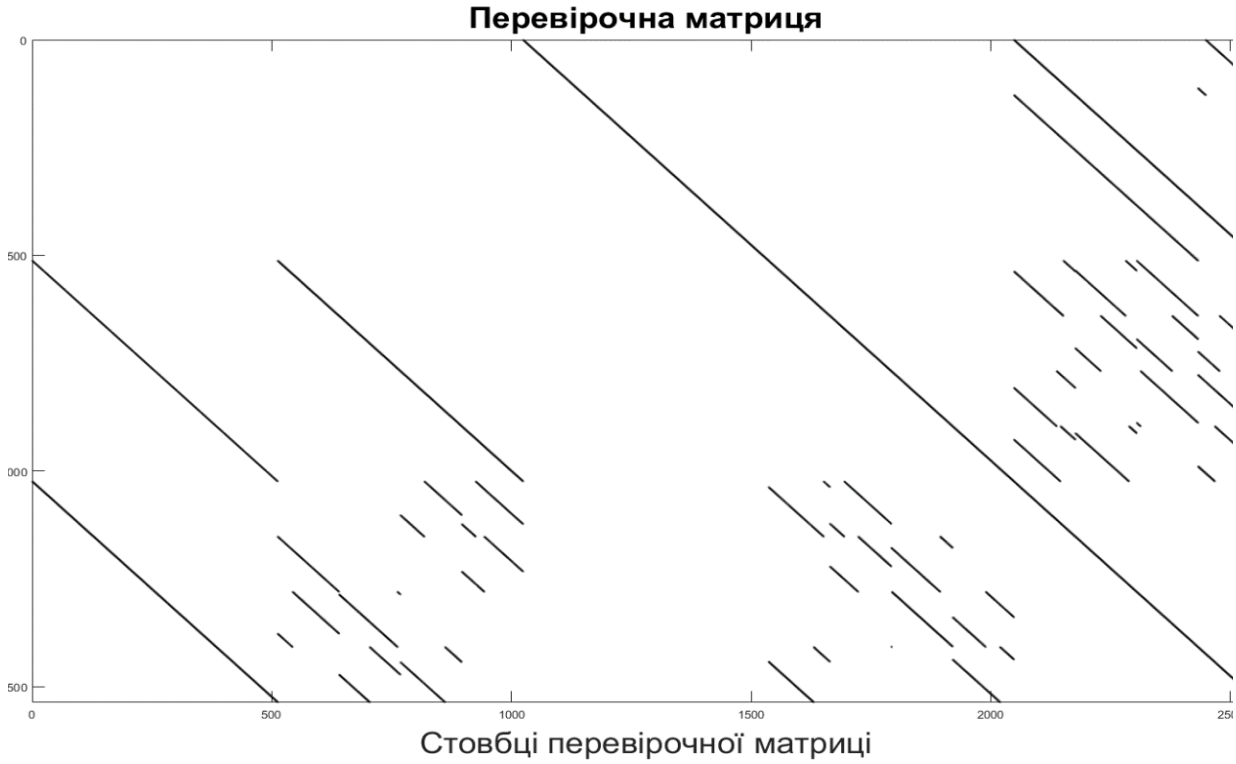


Рисунок 2.4 – Перевірочна матриця LDPC-коду для довжини інформаційного слова 1024 біт

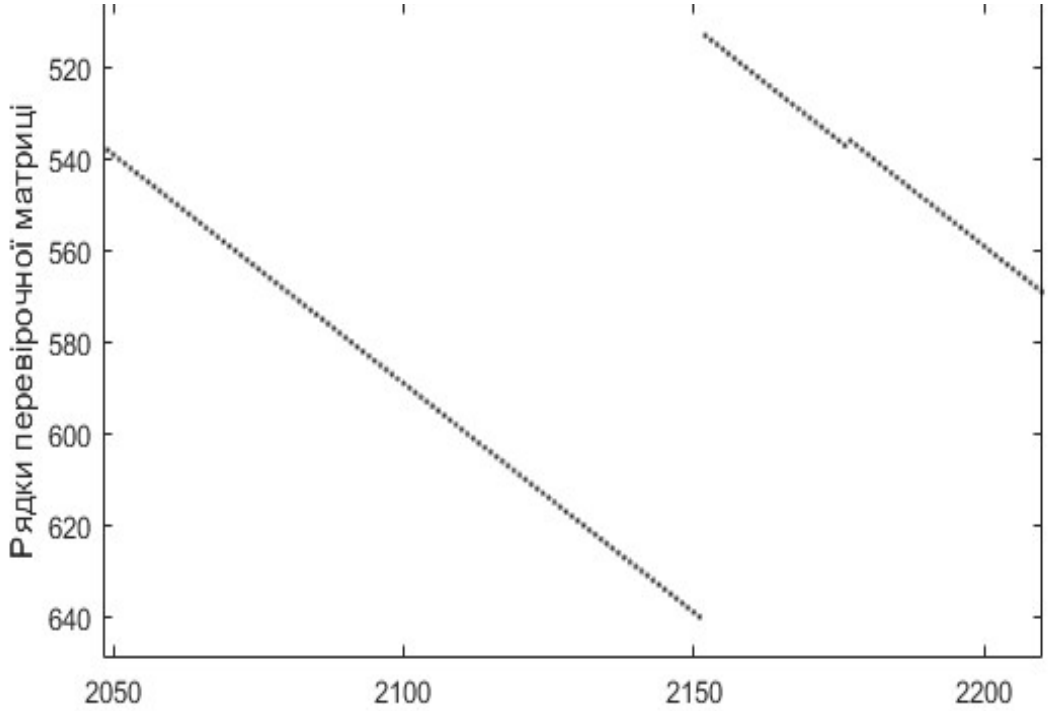


Рисунок 2.5 – Наближений фрагмент перевірочної матриці

Як можна побачити з рисунку 2.4, щільність ненульових елементів в перевірочній матриці дуже мала, тому ці коди і називаються кодами з низькою щільністю перевірок на парність (перевірки на парність – ненульові елементи в перевірочній матриці).

Отримані результати показано на рисунку 2.6.

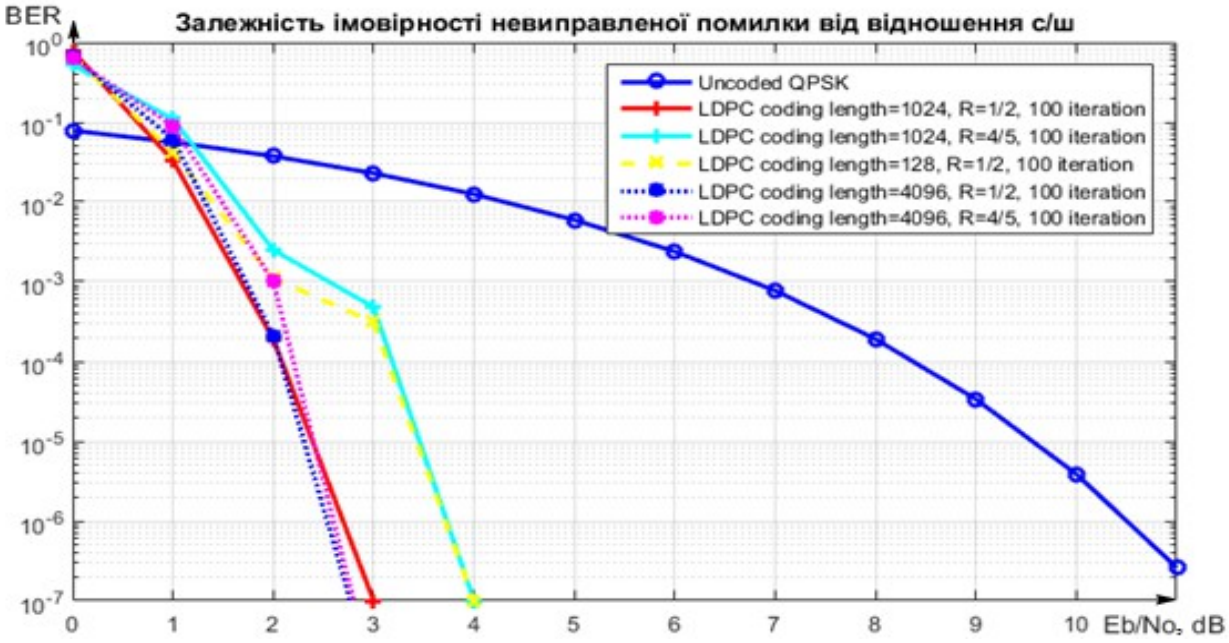


Рисунок 2.6 – Графік залежності ймовірності бітової помилки при LDPC-кодуванні

Із залежності на рисунку 2.6 можна побачити, що, застосувавши LDPC-кодування в радіолінії, вдасться зменшити енергетику сигналу приблизно у 6,5 децибел при ймовірності невиправленої похибки 10^{-6}, порівнюючи незакодований QPSK-сигнал та закодований LDPC-кодом.

2.3 Порівняльний аналіз ефективності згорткового та LDPC-кодування

В умовах, коли передавач знаходиться далеко від приймача та обмежений в потужності, з якою може передавати повідомлення, дуже важливу роль грає кодування та енергетичний виграш, який воно може

забезпечити при застосуванні. Тож порівняємо два види змодельованого завадостійкого кодування.

Графік залежності ймовірності бітової помилки з різними видами кодування показано на рисунку 2.7.

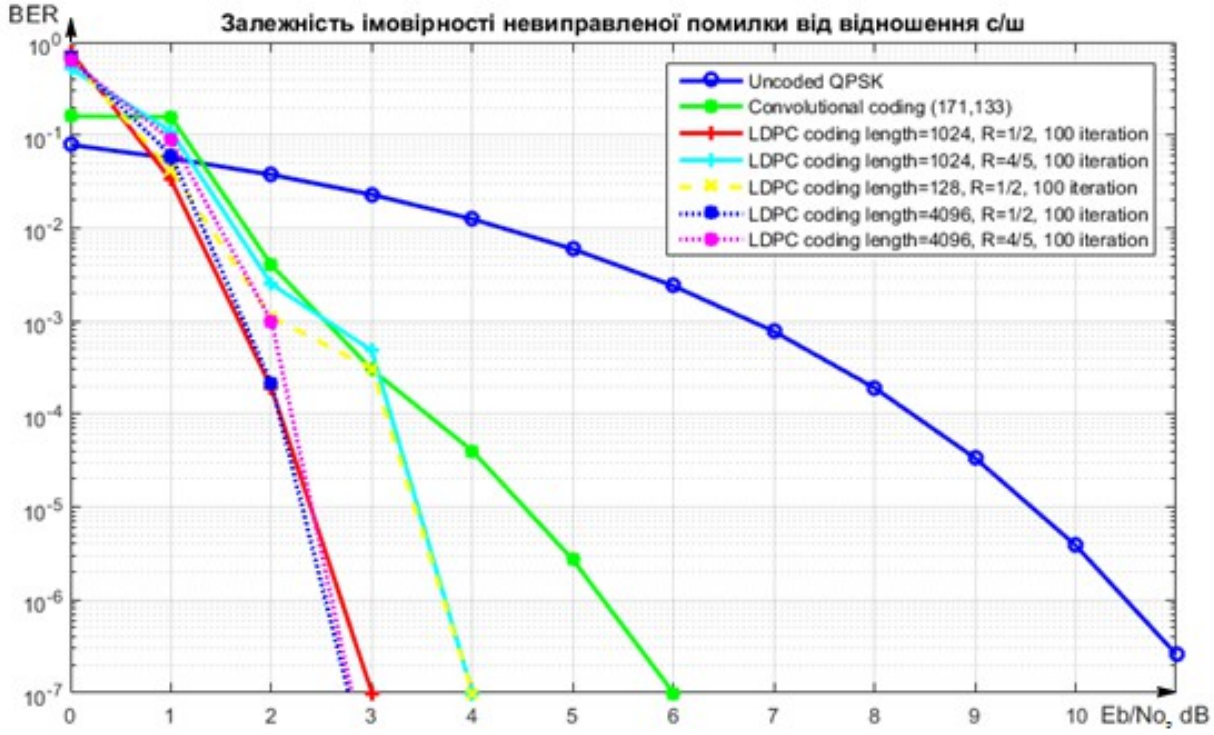


Рисунок 2.7 – Графік для порівняльного аналізу різних видів кодування

З рисунку 2.7 видно, що LDPC-кодування при застосуванні в радіолінії дає кращі показники бітової помилки, навіть при жорсткому декодуванні, при тих самих значеннях відношення сигнал/шум згорткового кодування.

Моделювання LDPC-кодування проводилося для інформаційного слова довжиною 128, 1024 та 4096 біта, а з ростом довжини інформаційного слова залежність бітової помилки від відношення сигнал/шум при застосуванні LDPC-кодів наближається до межі Шеннона.

Отже, відповідно до теореми Шеннона, система, що проектується, без коригуючого кодування буде складнішою, дорожчою і енергоємнішою. Звідси було зроблено висновок: система, яка не використовує коригуючого кодування і працює без помилок, не може бути ефективною системою.

Навпаки, ефективна система повинна мати можливість працювати в режимі з досить високою частотою помилок в потоці на вході декодера, а сам декодований потік повинен мати вкрай малу ймовірність помилки на біт. Тож застосування LDPC-кодування в радіолініях зв’язку к космічними апаратами на сьогодні є стандартом, який має найкращі показники. Через це для програмно-апаратної реалізації було обрано саме LDPC-кодування з довжиною коду 128.

3 РОЗРОБКА VHDL-МОДЕЛІ LDPC-КОДЕКА

3.1 Розробка VHDL-моделі кодера

У якості алгоритму кодування було використано алгоритм, запропонований в [5], що наведено на рис. 3.1. Цей алгоритм якнайкраще підходить для використання на ПЛІС, бо він використовує зсувні регістри, що дозволяє значно збільшити швидкість кодування порівняно з кодуванням, використовуючи ресурси процесора.

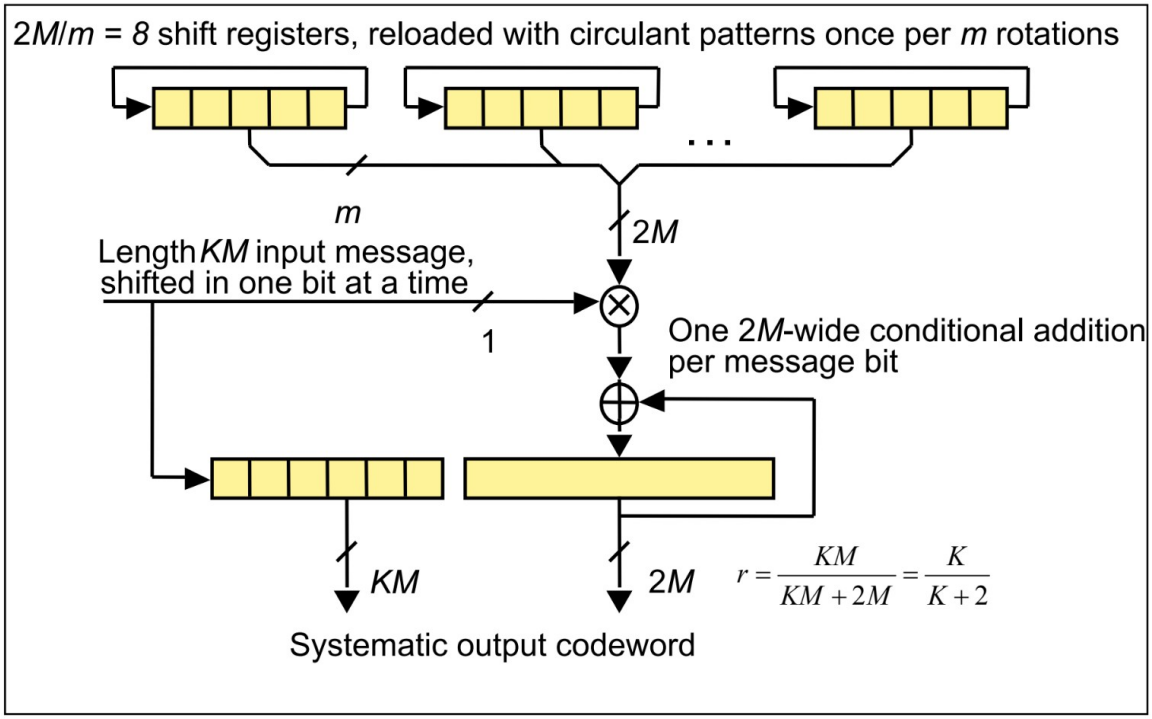


Рисунок 3.1 – Алгоритм кодування даних

Породжуючу матрицю для цього коду згенеровано таким чином, що для відтворення матриці на кодері не обов’язково зберігати всю матрицю. Достатньо зберігати кожен шістнадцятий рядок матриці. Ці рядки називаються циркулянтами. Інші рядки, а саме 60 (15*4), можна відтворити

через використання циклічного зсуву над рядком циркулянта, як наведено на рис. 3.1. Для цього використовується наступний модуль.

Модуль будується на базі чотирьох простих зсувних регістрів. Ці регістри виконують операцію циклічного зсуву вправо. Вони об’єднані в один регістр, шо називається shift_register (див. лістінг 3.1). Цей регістр має три вхідні сигнали: clk, load та load_vector. Перший з них робить зсув даних, як показано на рисунку 3.1. Load та load_vector дозволяють завантажити нові дані (циркулянт) у цей зсувний регістр. Для цього необхідно передати дані по шині load_vector, а потім подати строб даних.

Лістинг 3.1 – VHDL-модель зсувного регістра

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity shift_register is
	generic(
		WIDTH : integer := 64
		);
	port(
		clk : in STD_LOGIC;
		load : in STD_LOGIC;
		load_vector : in STD_LOGIC_VECTOR (0 to WIDTH - 1);
		-------------------------------------------------
		output : out STD_LOGIC_VECTOR (0 to WIDTH - 1)
		);
end shift_register;


architecture shift_register of shift_register is
	constant m : integer := WIDTH/4; -- 16
	signal memory : std_logic_vector (0 to WIDTH - 1) := (others => '0');
	signal next_m : std_logic_vector (0 to WIDTH - 1) := (others => '0');
begin

next_m ((1-1)*m to 1*m - 1) <= memory(1*m - 1) & memory((1-1)*m to 1*m - 1 - 1);
next_m ((2-1)*m to 2*m - 1) <= memory(2*m - 1) & memory((2-1)*m to 2*m - 1 - 1);
next_m ((3-1)*m to 3*m - 1) <= memory(3*m - 1) & memory((3-1)*m to 3*m - 1 - 1);
next_m ((4-1)*m to 4*m - 1) <= memory(4*m - 1) & memory((4-1)*m to 4*m - 1 - 1);
process(clk)
```

```
begin
        if (rising_edge(clk)) then
            if (load = '1') then
               memory <= load_vector;
            else
               memory <= next_m;
            end if;
          end if;

end process;
output <= memory;
end shift_register;
```

Таким чином, використовуючи цей зсувний регістр, можна не зберігати усю породжуючу матрицю, а зберігати лише перший, сімнадцятий, тридцять третій та сорок дев'ятий рядок цієї матриці. Для зберігання цих рядків використовується пакет f_file_encoder (лістинг 3.2). Для відтворення усіх інших рядків використовується зсувний регістр shift_register.

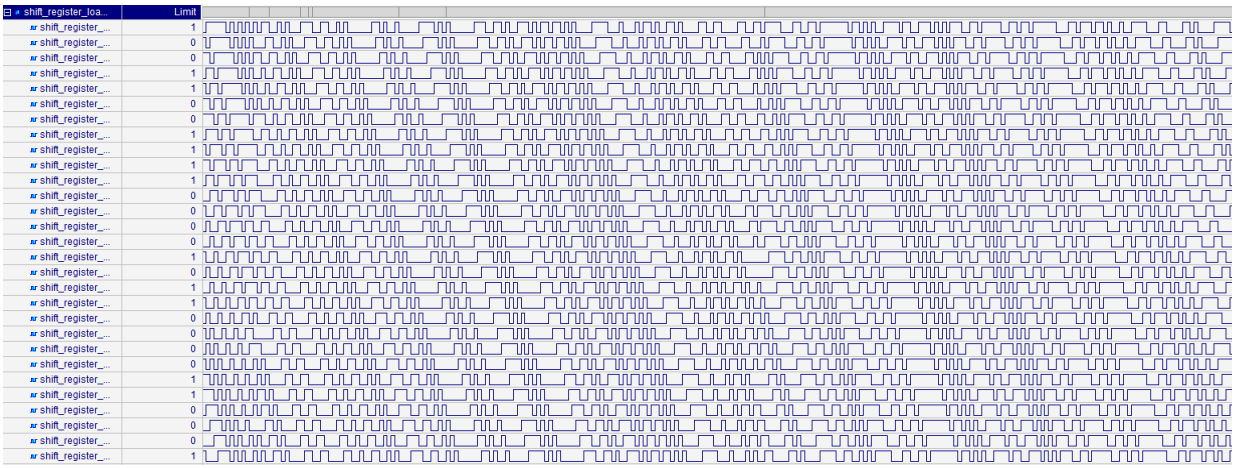


Рисунок 3.2 – Часова діаграма зсувного регістра

Лістинг 3.2 – VHDL-опис породжуючої матриці

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

package f_files_encoder is
        constant row1 : STD_LOGIC_VECTOR  (0 to 63) :=  -- matrix;
        constant row17 : STD_LOGIC_VECTOR (0 to 63) :=  -- matrix;
```

```
    constant row33 : STD_LOGIC_VECTOR (0 to 63) :=  -- matrix;
    constant row49 : STD_LOGIC_VECTOR (0 to 63) :=  -- matrix;
end package;
```

Маючи спосіб відтворити усю породжуючу матрицю, є можливість проводити кодування. Для цього інформація, що потрапляє на кодер, відразу потрапляє вперед кодового слова як інформаційна частина (див. рис. 3.1). Далі береться перший біт інформаційного слова і множиться на перший циркулянт. Результат записується в кодову частину кодового слова, а циркулянт записується в зсувний регістр. Потім проводиться операція зсуву зсувного регістра і множення вже другого інформаційного біта на значення зсувного регістра. Результат складається по модулю два з кодовою частиною кодового слова, яку вже збережено в кодері. Такі операції проводяться ще 14 разів. Потім проходить зміна циркулянта, і операції повторюються. Після проходження всіх циркулянтів буде отримано кодову частину, яка додається до інформаційної, і на виході буде отримано кодове слово. Код, що реалізує цей алгоритм, наведено в додатку А.

3.2 Розробка VHDL-моделі декодера

В роботі було розроблено схему декодера, яка наведена на рисунку 3.3. Функціонує вона наступним чином.

На приймальній стороні встановлюють LDPC-декодер, в якому записано відповідну перевірочну матрицю для декодування прийнятого кодового слова. Вхідна послідовність з демодулятора, після якого значення кодової послідовності є двійковими, разом зі стробом, який сигналізує про надходження послідовності на вхід декодера, поступає на механізм станів. Цей механізм станів записує вхідну послідовність до внутрішньої пам’яті decoded та починає очікувати закінчення першої ітерації декодування. У цей час дані з внутрішньої пам’яті надходять на вхід блоку syndrome. У блоці

відбувається перевірка кодового слова на парність з використанням перевірочної матриці, яка описана у лістингу 3.3.

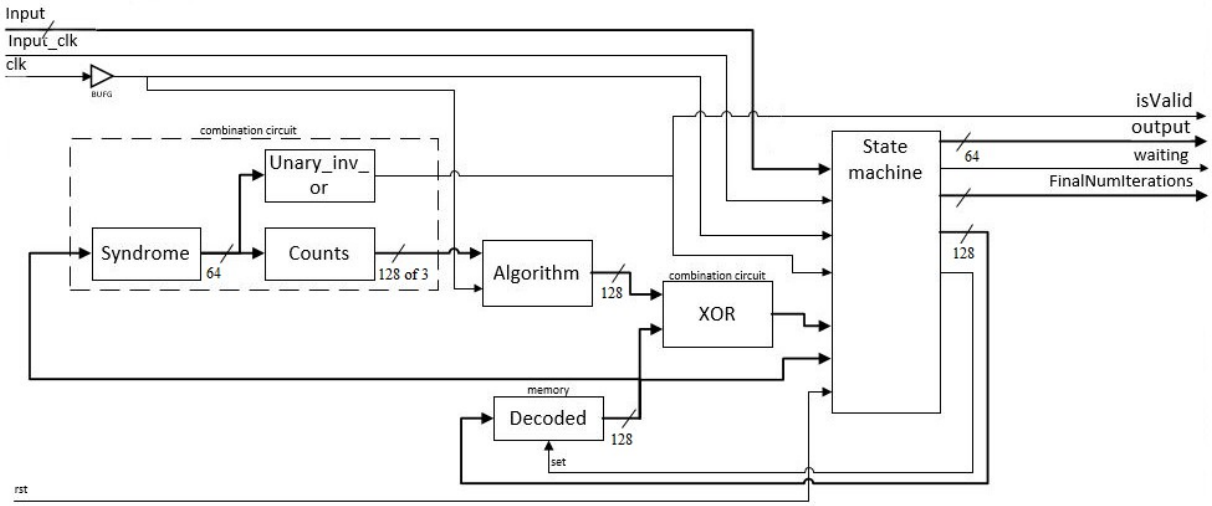


Рисунок 3.3 – Схема декодера

Лістинг 3.3 – VHDL-опис перевірочної матриці

```
library IEEE;
   use IEEE.STD_LOGIC_1164.all;

package decoder_package is
          type STD_LOGIC_TH_ARRAY is array (integer range <>) of
std_logic_vector(2 downto 0);
        type T_DATA is array (0 to 1727) of integer;
        constant f_file_refactored : T_DATA := -- matrix
        constant parent_nodes     : integer := 0;
        constant child_nodes      : integer := 1152;
        constant node       : integer := 9;
        constant node_size  : integer := 0;
        constant index0     : integer := 1;
        constant index1     : integer := 2;
        constant index2     : integer := 3;
        constant index3     : integer := 4;
        constant index4     : integer := 5;
        constant index5     : integer := 6;
        constant index6     : integer := 7;
        constant index7     : integer := 8;
end package;
```

Блок перевірки проходження на парність наведено в додатку Б. Вихідними даними цього блоку є однобітні значення, які символізують, чи проходить кодове слово перевірку на парність. У вибраній перевірочній матриці цих перевірок 64. Умовою проходження перевірки кодового слова декодером є проходження усіх перевірок на парність. Отже, після блоку syndrome підключено блок unary_inv_or, лістинг якого наведено нижче (лістинг 3.4).

Лістинг 3.4 – VHDL-модель модулю unary_inv_or

```
library IEEE;
   use IEEE.STD_LOGIC_1164.all;

-- Module of execution operation nor under WIDTH bits

entity unary_inv_or is
   generic(
         WIDTH: integer := 64
         );
   port(
         inp: in std_logic_vector(WIDTH - 1 downto 0);
         outp: out std_logic
         );
end entity;


architecture unary_inv_or of unary_inv_or is
   signal temp: std_logic_vector(WIDTH - 1 downto 0);
begin

   outp <= not temp(WIDTH - 1);

   temp(0) <= inp(0);
   gen: for i in 1 to WIDTH - 1 generate
         temp(i) <= temp(i-1) or inp(i);
   end generate;

end architecture;
```

Цей модуль дозволяє виявити, чи є у вхідного кодового слова хоча б

одна непройдена перевірка на парність. Якщо непройдені перевірки є, то вихідний сигнал модуля unary_inv_or буде дорівнювати нулю; якщо ж усі перевірки на парність пройдено, то вихідний сигнал буде мати значення одиниці.

У цей же час значення проходження перевірок на парність з модуля syndrome потрапляють на лічильники непройдених перевірок. Для кожного біта кодового слова створюється свій лічильник, який у двійковому вигляді відображає кількість непройдених перевірок на парність для даного біта. В результаті цієї операції можна буде знайти біт, який не проходить максимальну кількість перевірок. Цей модуль наведено в лістингу 3.5.

Лістинг 3.5 – VHDL-модель модулю counter_of_signals

```
library IEEE;
   use IEEE.STD_LOGIC_1164.all;

entity counter_of_signals is
   port(
        a : in STD_LOGIC_VECTOR(5 downto 0);  -- max num of one's in column
         b : out STD_LOGIC_VECTOR(2 downto 0)  -- var for num of one's in column
        );
end counter_of_signals;


architecture counter_of_signals of counter_of_signals is
begin
       b <= "000" when (a = "000000") else
                                "001" when (a = "000001" or a = "000010" or a = "000100" or a = "001000" or a = "010000" or a = "100000") else
                               "010" when(a = "110000" or a = "101000" or a = "100100" or a = "100010" or a = "100001" or a = "011000" or a = "010100" or a = "010010" or a = "010001" or a = "001100" or a = "001010" or a = "001001" or a = "000110" or a = "000101" or a = "000011") else
                                "100" when (a = "001111" or a = "010111" or a = "011011" or a = "011101" or a = "011110" or a = "100111" or a = "101011" or a = "101101" or a = "101110" or a = "110011" or a = "110101" or a = "110110" or a = "111001" or a = "111010" or a = "111100") else
                                "101" when (a = "111110" or a = "111101" or a = "111011" or a = "110111" or a = "101111" or a = "011111") else
```

```
"110" when (a = "111111") else
"011"; end counter_of_signals;
```

Після цього значення виходів модулів counts потрапляють в модуль algorithm (Лістинг Б.1, додаток Б). Цей модуль відповідає за вибір біта для інверсії в процесі декодування. В даній реалізації було використано алгоритм Bif Flipping, але при бажанні цей алгоритм можна замінити саме тут. Головною ідеєю цього алгоритму є вибір біта, який не пройшов максимальну кількість перевірок на парність із всіх бітів кодового слова, і повинен бути інвертований. Для вибору цього біта використовується послідовне порівняння кількості непройдених перевірок на парність для кожного з бітів. Для оптимізації цього процесу і підвищення швидкості однієї ітерації декодування можна використати паралельний пошук максимального числа непройдених перевірок, реалізувавши це комбінаційною схемою, але це понесе за собою велику кількість витрачених ресурсів ПЛІС. В результаті блок algorithm генерує вектор виправлення помилок, який на позиціях, які треба інвертувати, виставляє одиницю, а на позиціях, які потрібно залишити без змін, ставить нуль. Цей вектор потрапляє на наступний блок (рис. 3.3), в якому побітно виконується операція xor. Результат цієї операції передається в механізм станів, і в кінці кожної ітерації перезаписується у внутрішню пам’ять. Ітерації повторюються до моменту повного виправлення всіх помилок, або до того моменту, коли кількість ітерацій перевищить поріг, заданий generic-константою.

3.3 Перевірка працездатності системи

Для тестування роботоспроможності кодера було використано середовище Active-HDL, в якому було проведено симуляцію роботи кодера. На вхід тактування було подано міандр з частотою 10МГц, на вхід даних – випадкове слово з 64-ох біт, а на вхід input_clk було подано строб. Результат

кодування було перевірено за допомогою MATLAB, та він виявився правильним.

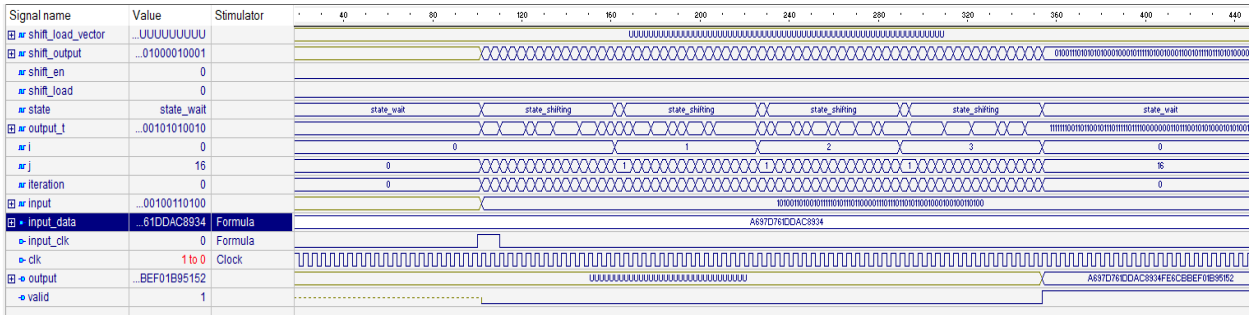


Рисунок 3.4 – Часова діаграма роботи кодера

Для перевірки роботи декодера також було використано середовище Active-HDL, в якому було проведено симуляцію роботи декодера. На вхід тактування було подано міандр з частотою 10МГц, на вхід даних спочатку було подано нульове кодове слово та було подано строб. Як видно з рисунку 3.5, декодер відразу визначив, що подане на нього кодове слово є правильним, та вийшов з циклу декодування з нульовою кількістю ітерацій.

Після цього на вхід даних декодера було подано слово з однією помилкою та строб даних. Як видно з рисунку 3.5, декодер визначив, що кодове слово не є вірним (сигнал IsValid дорівнює нулю), та почав цикл декодування. Після першої ітерації декодер сформував вектор виправлення помилки, в якому, правильно виставивши біт виправлення, зміг відновити первісну інформацію і вивести слово без помилки.

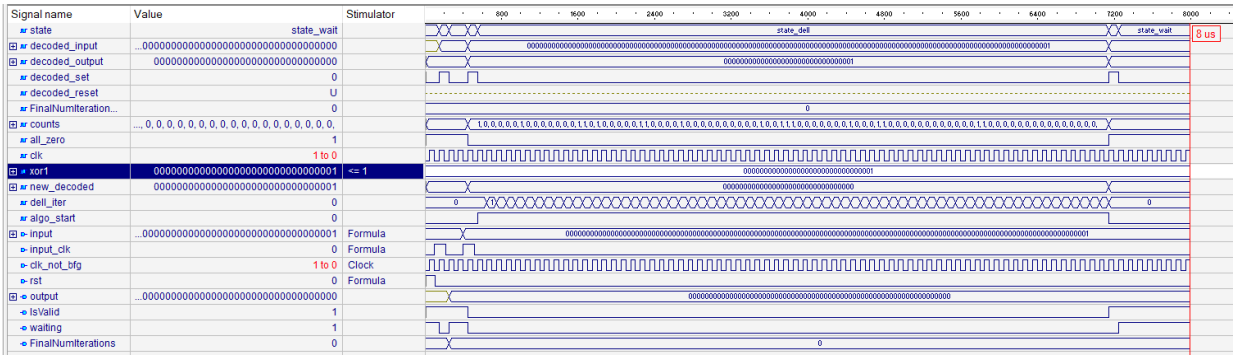


Рисунок 3.5 – Часова діаграма роботи декодера

3.4 Синтез і імплементація пристрою

Оскільки пристрій, який проектується, в основному складають елементи пам'яті (тригери), виконаємо синтез його на FPGA. Хоча CPLD мають більш високу швидкодію, їх використовують переважно для реалізації логічних функцій великого числа змінних. Оскільки для синтезу та імплементації використовувалося інструментальне середовище Xilinx Progect Navigator V5.1, то вибір мікросхеми FPGA проводився з елементної бази фірми Xilinx.

Виконано синтез і імплементацію, створено файли опису пристрою після імплементації.

Результати синтезу кодера наведені на рисунку 3.6.

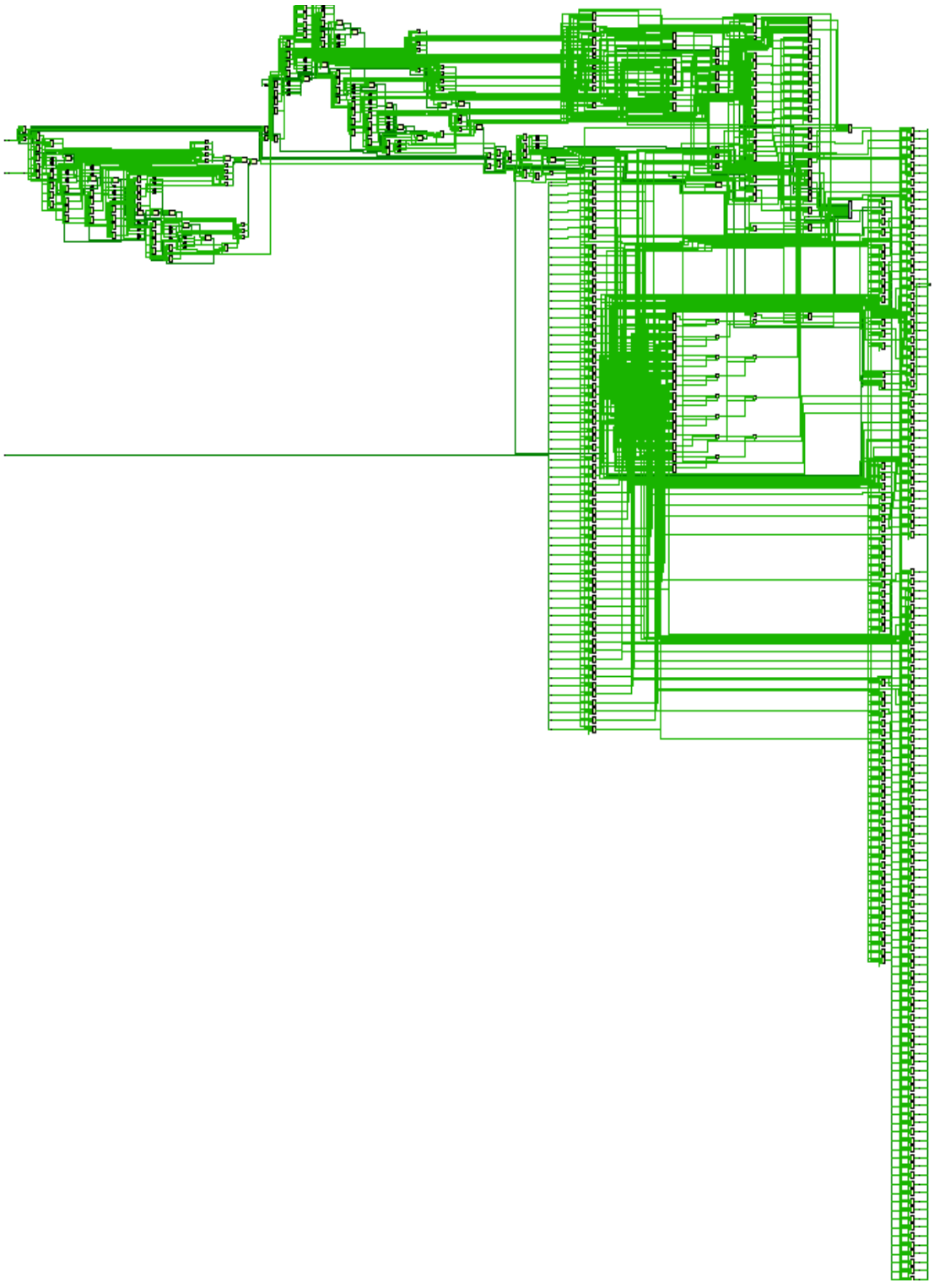


Рисунок 3.6 – Узагальнена схема кодера після синтезу

В узагальненій схемі кодера присутній модуль shift_memory. Схема модулю наведено на рисунку 3.7.

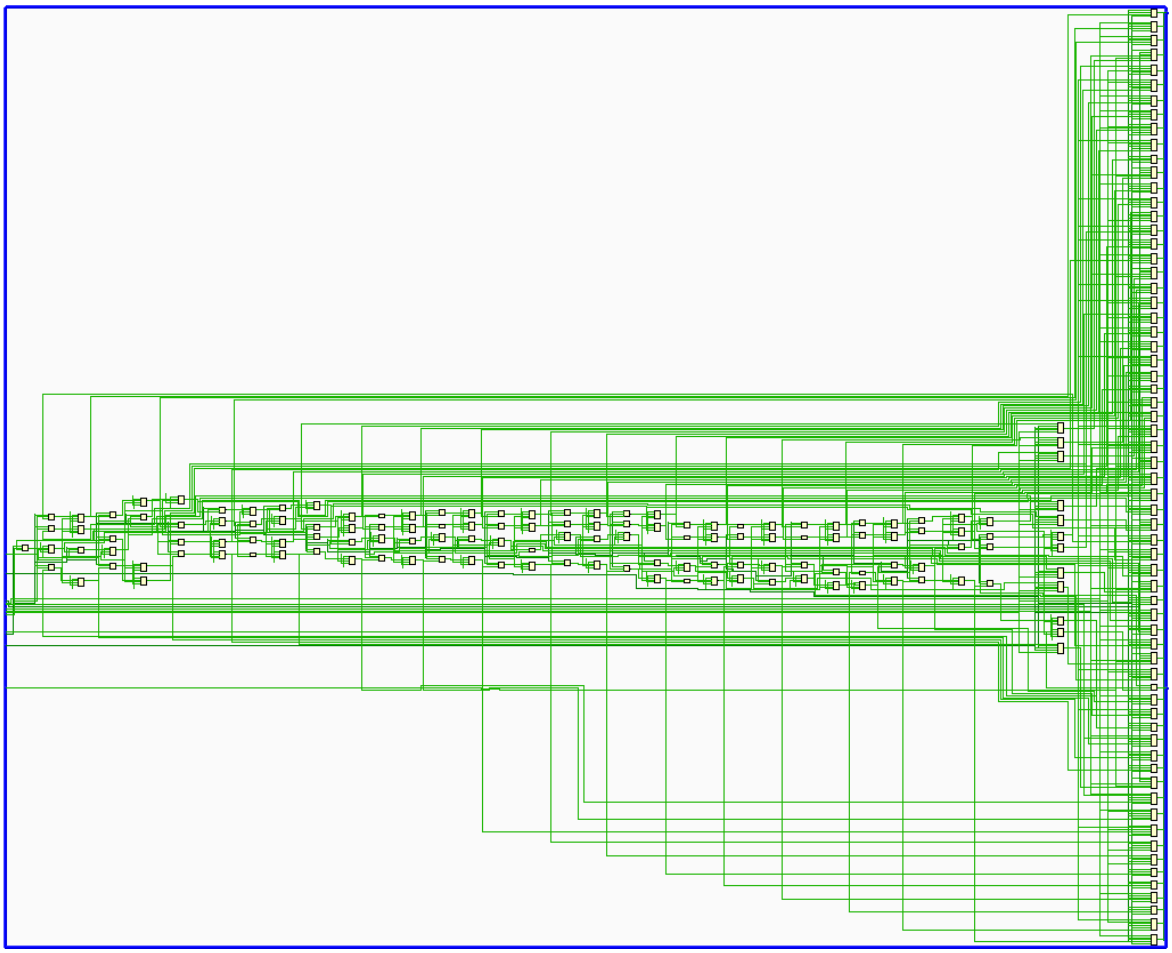


Рисунок 3.7 – Схема модуля shift_memory

Загальна кількість використаних ресурсів при синтезі кодера наведено на рисунку 3.8.

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	259	0	53200	0.49
LUT as Logic	259	0	53200	0.49
LUT as Memory	0	0	17400	0.00
Slice Registers	410	0	106400	0.39
Register as Flip Flop	345	0	106400	0.32
Register as Latch	65	0	106400	0.06
F7 Muxes	8	0	26600	0.03
F8 Muxes	4	0	13300	0.03

Рисунок 3.8 – Використані ресурси при синтезі кодера

Результати синтезу декодера наведено на рисунку 3.9.

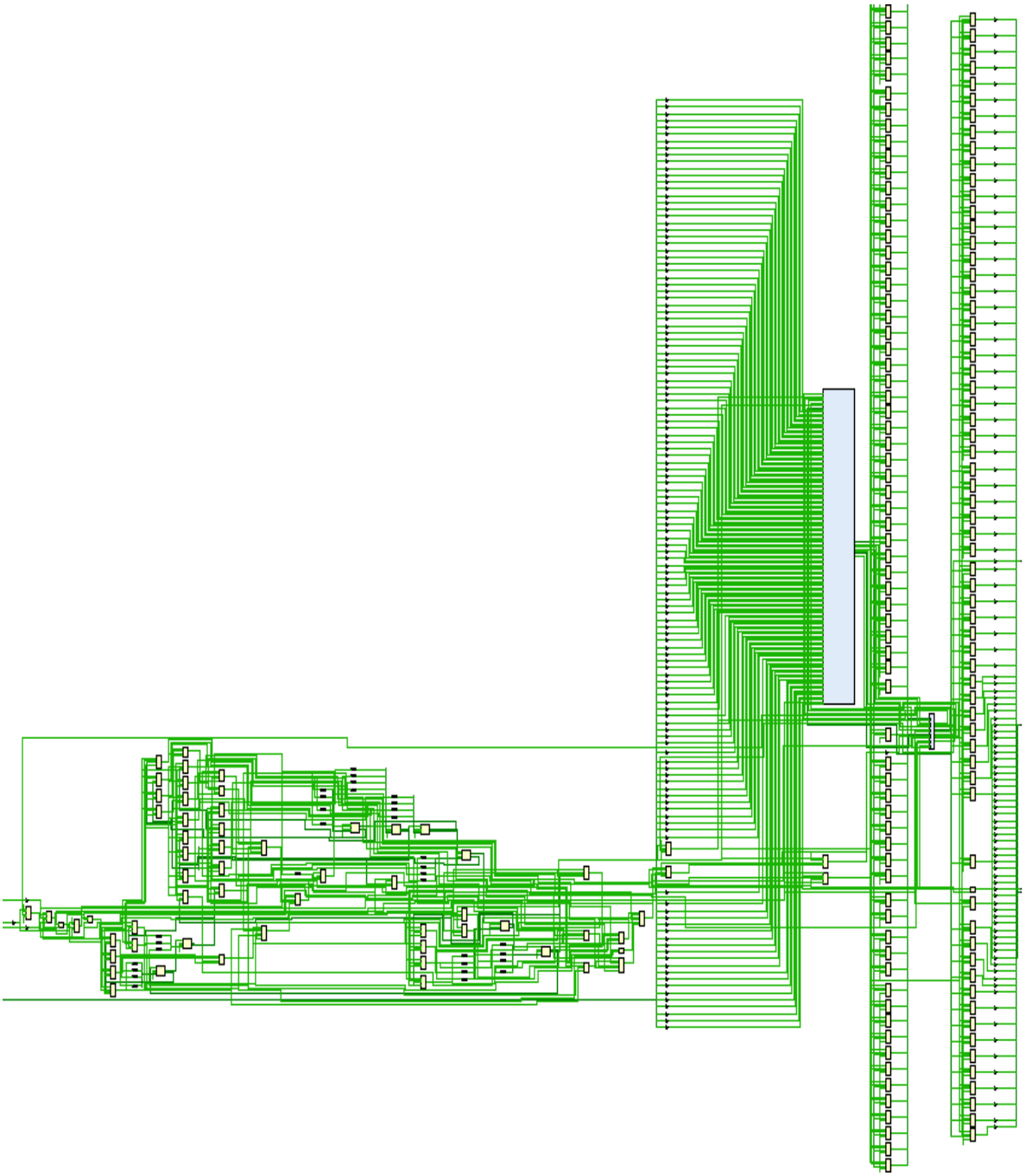


Рисунок 3.9 – Фрагмент схеми декодера

В схемі декодера присутній блок memory, який відповідає за збереження кодового слова в процесі декодування. Фрагмент схеми цього блоку наведено на рисунку 3.10.

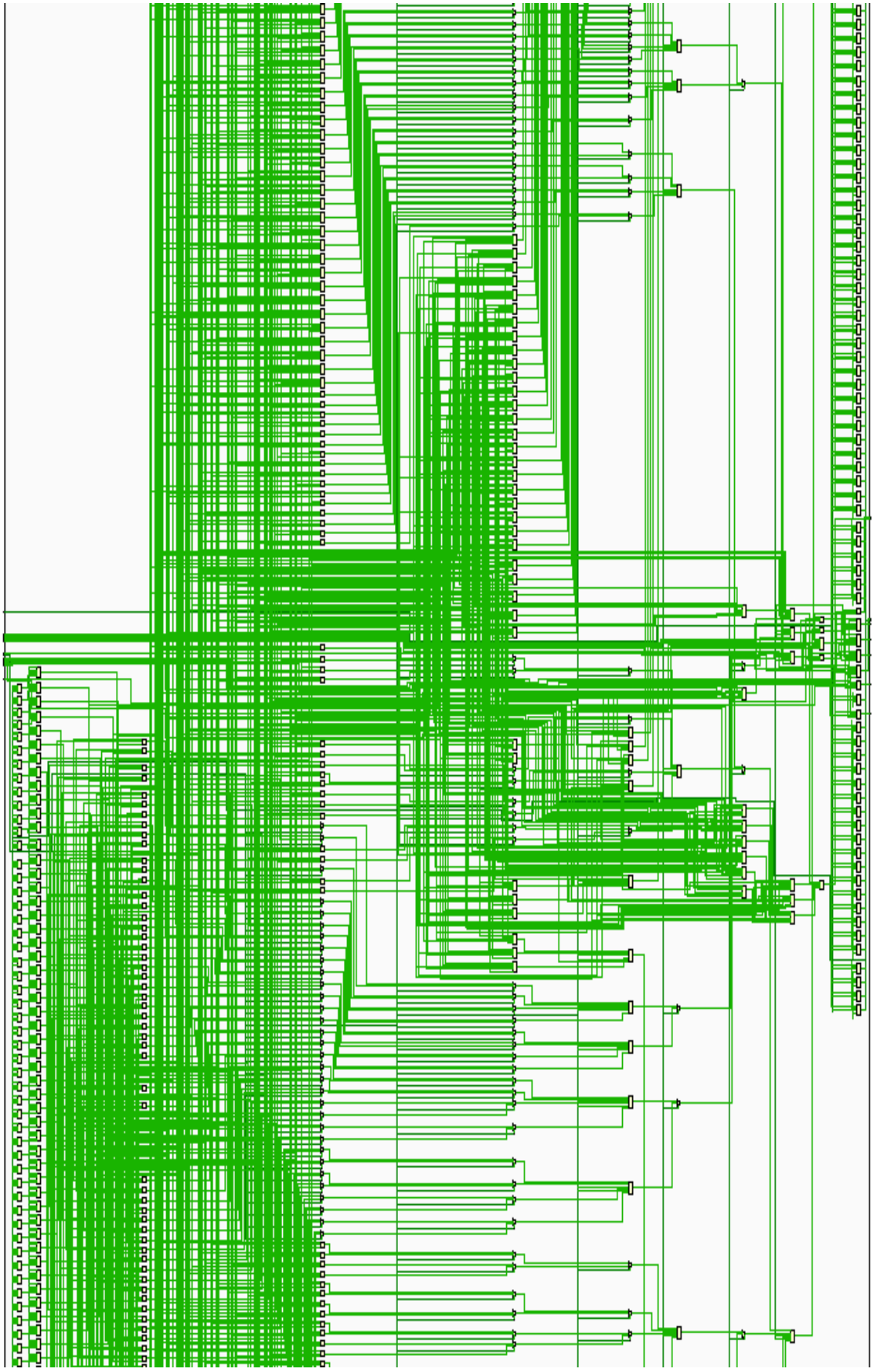


Рисунок 3.10 – Фрагмент схеми блоку memory

Також в узагальненій схемі декодера присутній модуль algorithm, який відповідає за вибір біта для інверсії у кодовому слові. Схему цього блоку наведено на рисунку 3.11.

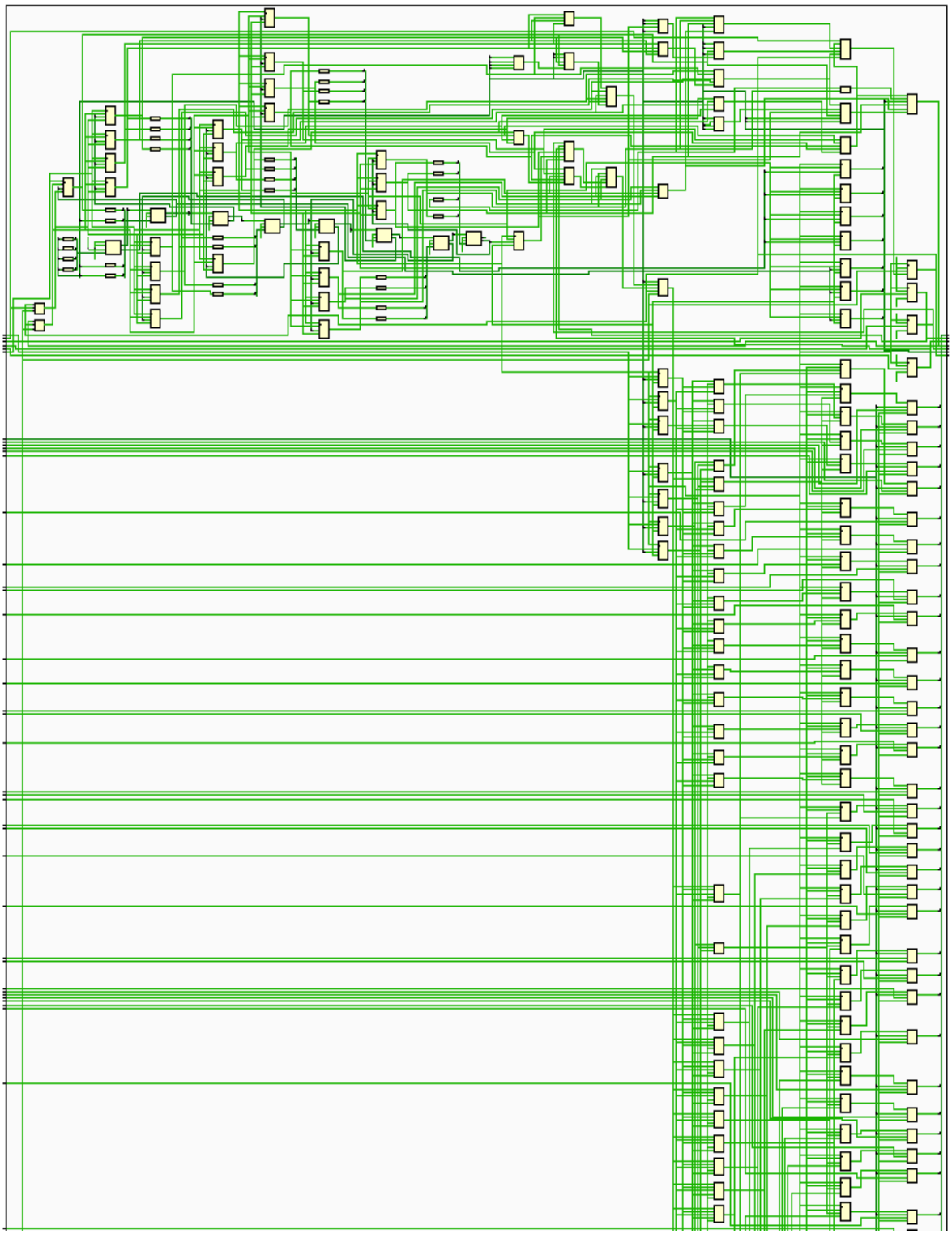


Рисунок 3.11 – Фрагмент схеми блоку algorithm

Загальна кількість використаних ресурсів для синтезу декодера наведено на рисунку 3.12.

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	750	0	53200	1.41
LUT as Logic	750	0	53200	1.41
LUT as Memory	0	0	17400	0.00
Slice Registers	528	0	106400	0.50
Register as Flip Flop	528	0	106400	0.50
Register as Latch	0	0	106400	0.00
F7 Muxes	163	0	26600	0.61
F8 Muxes	78	0	13300	0.59

Рисунок 3.12 – Використані ресурси при синтезі декодера

4 ДІАГНОСТИЧНИЙ ЕКСПЕРИМЕНТ

Було проведено тестування розробленого пристрою, при цьому переслідувалися дві мети. Першим етапом необхідно було перевірити працездатність розроблених модулів кодера і декодера в реальній радіолінії. Метою другого етапу було отримати якісний аналіз роботи даних модулів.

Для проведення першого етапу діагностичного експерименту використовувалися дві плати PicoZed SDR AD9361 Development Kit з вбудованими приймачами Analog Devices AD9361 з пласкими антенами і системами на кристалах XILINX XC7Z035-2LFBG676i, які містять двоядерні ARM Cortex-A9 процесори і ПЛІС (FPGA). В даному експерименті використовувалися приймачі та ПЛІС, на яких були прошиті розроблені модулі LDPC-кодера і декодера. Експеримент проводився в лабораторних умовах без використання підсилювачів потужності і з використанням атенюаторів для додаткового зниження рівня потужності сигналу, що передається.

У процесі підготовки до проведення діагностичного експерименту було побудовано схему, яку наведено на рис. 4.1.

У схемі діагностичного експерименту є два пристрої діагностування. Пристрій 1 складається з генератора інформації 1, LDPC-кодера і модулятора.

Генератор інформації 1 відповідає за формування потоку заздалегідь відомих пакетів даних, які надходять на LDPC-кодер. У модулі LDPC-кодера відбувається безпосередньо кодування інформації, яка надалі передається на модулятор, після чого поступає в радіоканал.

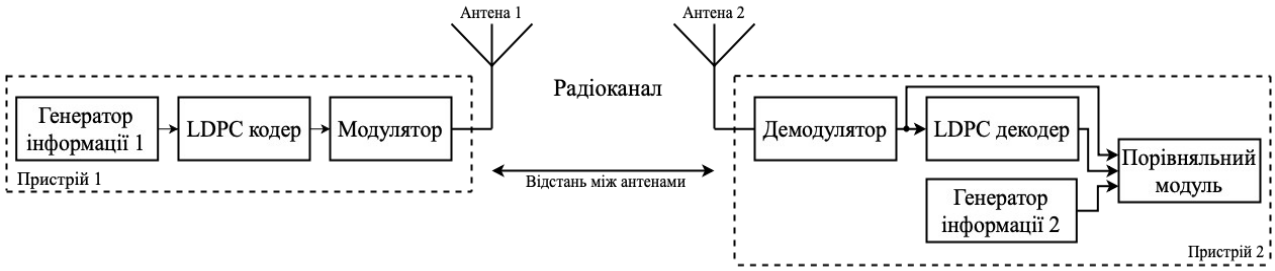


Рисунок 4.1 – Схема діагностичного експерименту (етап 1)

Пристрій 2 складається з демодулятора, LDPC-декодера, генератора інформації 2 і модуля порівняння. Демодулятор демодулює дані з радіоканалу і передає їх на LDPC-декодер і модуль порівняння. Розроблений модуль LDPC-декодера виконує декодування і передає пакети даних на модуль порівняння. В цей же час генератор інформації 2 генерує пакети інформації в тій же послідовності, що і модуль генерації інформації 1, та передає їх на модуль порівняння. Все це дозволяє зробити порівняння трьох потоків даних: ідеального (з генератора інформації 2), з помилками після радіоканалу (з демодулятора) і з виправленими помилками (з LDPC-декодера). Порівняння потоків в даному експерименті дозволяє визначити, чи виникають помилки в радіоканалі і чи справляється LDPC-декодер з ними.

Для проведення аналізу отриманих результатів було проведено низку експериментів з різними співвідношеннями сигнал/шум. Для імітації різних радіоканалів використовувалася зміна відстані між антенами пристроїв 1 і 2. Було проведено два експерименти, в яких антени знаходилися на відстані 10 см і 10 м відповідно. Перший експеримент показав, що дані, які відправлялися з пристрою 1, потрапляли через радіоканал в пристрій 2 без спотворень. Інформація на модулі порівняння збігалася для всіх трьох потоків даних.

Другий експеримент (рис. 4.2) показав, що інформація, яку було передано через радіоканал, спотворювалася, але LDPC-декодер виконував виправлення помилок. В середньому в радіоканалі виникала одна помилка на 1000 біт.

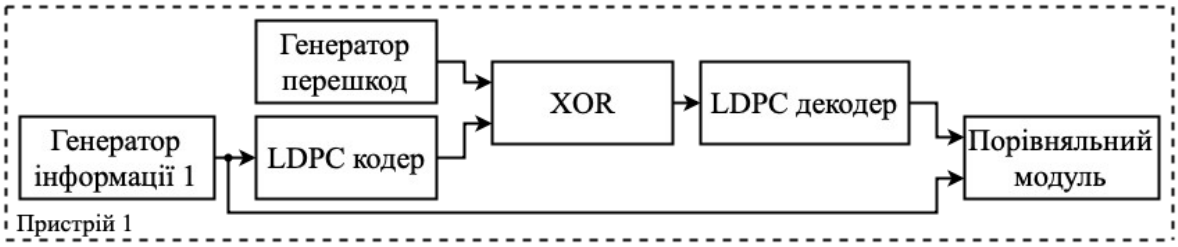


Рисунок 4.2 – Схема діагностичного експерименту (етап 2)

Для якісного аналізу роботи розроблених пристроїв був проведений другий етап діагностичного експерименту. Метою було визначити, скільки помилок в прийнятих пакетах здатний виправити LDPC-декодер. Також метою було визначити, чи є різниця в результатах роботи в залежності від відносного розташування помилок в пакеті. Для проведення даного експерименту була побудована наступна схема.

Відмінністю даної схеми від попередньої є те, що експеримент проводився всередині одного пристрою. Для імітації передачі даних радіоканалом використовувався наступний підхід. Дані з генератора інформації подавалися на LDPC-кодер, на якому відбувалося кодування пакетів. Після цього до сформованого пакету за допомогою операції суми по модулю два додавалися помилки, згенеровані модулем генерації помилок. Тільки після цього дані з помилками подавалися на LDPC-декодер, в якому відбувалися декодування і передача отриманих даних на модуль порівняння. В результаті використання даного підходу під час проведення експерименту контролювалася кількість і відносне розташування помилок в переданих пакетах. Експеримент проводився для різної кількості помилок в пакеті.

При аналізі результатів експериментів було виділено дві основні групи помилок в прийнятих пакетах: помилки, розташовані в пакетах випадковим чином, та помилки, розташовані поспіль (пакетні помилки). Аналіз показав, що дані групи помилок необхідно розглядати окремо, оскільки дані групи показують різні статистичні дані. Результати експериментів щодо роботи

декодера при випадковому розташуванні помилок в прийнятому пакеті наведені в табл. 4.1. Результати експериментів щодо роботи декодера при пакетному розташуванні помилок в прийнятому пакеті наведені відповідно в табл. 4.2.

Таблиця 4.1 – Результати експериментів щодо роботи декодера при випадковому розташуванні помилок

Кількість помилок, доданих в переданий пакет інформації	Загальна кількість проведених тестів (пакетів), для даної кількості доданих помилок	Кількість виправлених пакетів	Кількість невиправлених пакетів	Відсоток виправлених пакетів
10	52	38	14	73%
11	40	26	14	65%
12	39	21	18	54%
13	36	17	19	47%
14	33	7	26	21%
15	32	5	27	16%
16	30	3	27	10%

Таблиця 4.2 – Результати експериментів щодо роботи декодера при пакетному розташуванні помилок

Кількість помилок, доданих в переданий пакет інформації	Загальна кількість проведених тестів (пакетів), для даної кількості доданих помилок	Кількість виправлених пакетів	Кількість невиправлених пакетів	Відсоток виправлених пакетів
8	18	10	8	56%
13	19	8	11	42%
16	18	8	10	44%
19	18	7	11	38%
23	18	6	12	33%
28	18	2	16	11%
29	18	2	16	11%

Якщо порівняти значення відсотка виправлених пакетів для двох виділених груп появи помилок, то можна помітити наступне: з якогось моменту (а саме від 13 помилок в пакеті) LDPC-декодер краще справляється з пакетними помилками, ніж із випадково освіченими (рис. 4.3). Це можна пояснити тим, що для декодування використовуються перевірочні матриці з малою кількістю перевірок на парність, і ці перевірки розташовані псевдовипадковим чином.

Це дозволяє отримувати інформацію для декодування певного біта пакета не тільки від сусідніх біт, але і від далеко розташованих біт. Дана особливість дозволяє не використовувати переміжник і деперемiжник схемі радіолінії, що є економією апаратних і часових ресурсів всього пристрою.

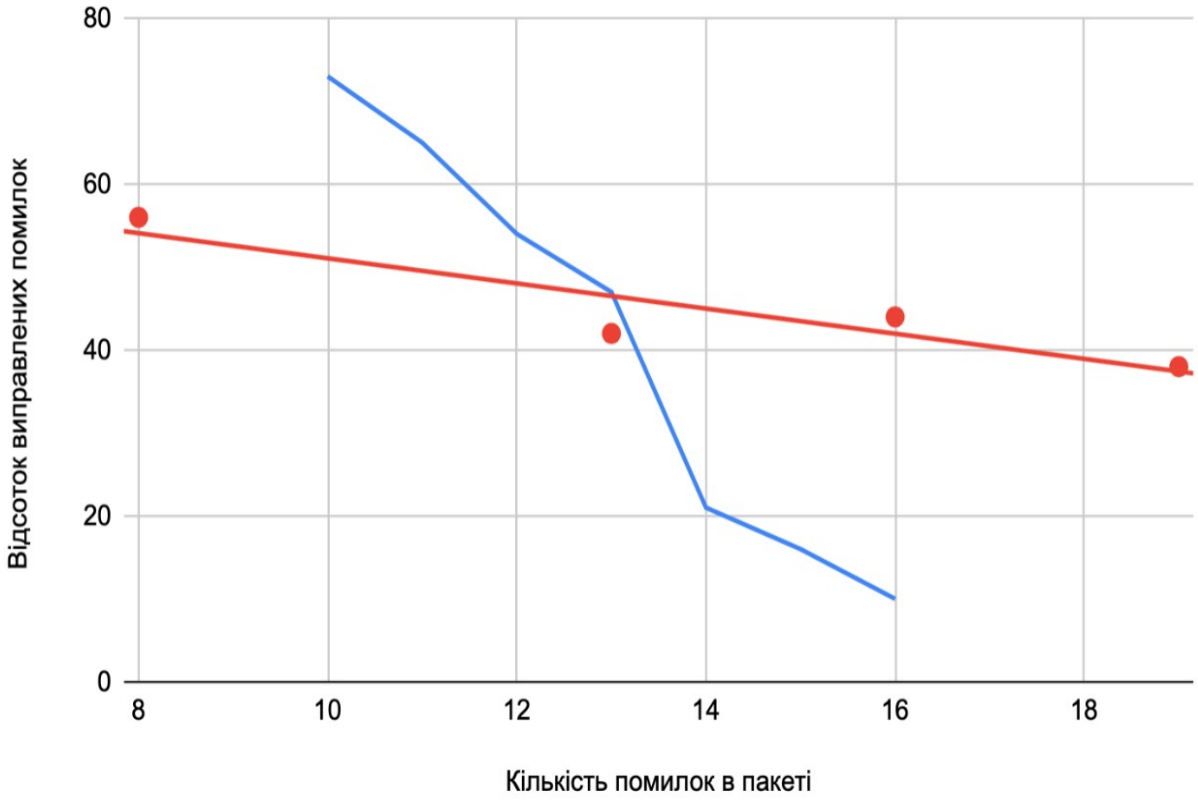


Рисунок 4.3 – Порівняння результатів для пакетних і випадкових помилок

Проведений діагностичний експеримент показав наступне: 1) спроектовані модулі LDPC-кодера і декодера виконують своє функціональне

завдання, а саме забезпечують стійкість під час передачі інформації; 2) LDPC-коди краще справляються з пакетними помилками, ніж з випадково розташованими.

ВИСНОВКИ

В результаті виконання роботи було вирішено актуальну технічну задачу розробки IP Core кодека завадостійкої системи передачі на ПЛІС.

В процесі виконання роботи було побудовано математичну модель Simulink згорткового кодування та декодування в радіолінії та програму, яка реалізує симуляцію кодування інформації лінійним блоковим кодом з низькою щільністю перевірок на парність, проходження закодованої послідовності через канал з шумами та її декодування за допомогою перевірочної матриці.

В результаті були отримані графіки залежності імовірності бітової помилки від відношення сигнал/шум для згорткового та LDPC-кодування, які було порівняно, і виявлено, що при кодуванні інформаційної послідовності згортковим кодом відношення сигнал/шум може бути знижено на 6 децибел порівняно з незакодованою передачею. Але використовуючи LDPC-кодування, імовірність бітової помилки наближається до межі Шеннона, що є дуже добрим показником, і через що ці коди стали стандартом завадостійкого кодування у зв’язку з космічними апаратами та будуванні мереж п’ятого покоління.

Для інтегрування алгоритму LDPC-кодування в прилади зв’язку було виконано програмно-апаратну реалізацію та імплементацію складного функціонального блоку на ПЛІС. Виконано розробку алгоритму декодування та структури декодера для реалізації його на ПЛІС.

Також виконано верифікацію вихідної моделі кодека, моделі кодека після імплементації. Було виконано проведення діагностичного експерименту.

Наукова новизна одержаних результатів:

1) отримано подальший розвиток алгоритму LDPC-кодування для реалізації його на ПЛІС за рахунок використання зсувних регістрів і суматорів та керуючого автомата;

2) розроблено алгоритм та схеми LDPC-декодера на ПЛІС з використанням зсувних регістрів і суматорів, виконуючого автомата та керуючого автомата, який синхронізує роботу блоків пристрою.

Розроблені модулі може бути використано для реалізації завадостійкого LDPC-кодування інформації в високошвидкісних системах передачі даних, а саме відеоданих із супутників, дронів, безпілотників. Результати роботи можна використовувати як готові модулі при розробці різноманітних пристроїв, в яких використовується завадостійке кодування – наприклад, у радіолінії зв’язку з космічними апаратами. Це дає можливість зменшити часові витрати, підвищити якість проекту в цілому і автоматизувати процес проектування нових систем.

Результати було представлено на студентському форумі у 2019 р., опубліковано у науковому журналі переліку ВАК у 2019 р., а також результати роботи впроваджено в ДП «ЗАО НДІРВ» у 2019 р. у формі IP Core для програмно-апаратної системи передачі даних, та реалізовано в цій системі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Прокис, Д. Цифровая связь [Текст] / Д. Прокис. – М. : Радио и связь, 2000. – 800 с.

2. Волков, Л. Н. Системы цифровой радиосвязи: базовые методы и характеристики [Текст] / Л. Н. Волков, М. С. Немировский, Ю. С. Шинаков. – М. : Эко-Трендз, 2005. – 392с.

3. Вернер, М. Основы кодирования [Текст] : пер. с нем. – К. : Зигангирова; М. : ТЕХНОСФЕРА, 2004. – 288с.

4. Short block length LDPC codes for TC synchronization and channel coding [Текст] – М. : Consultative Committee for Space Data Systems, 2015. – 39 p.

5. TM synchronization and channel coding: Recommended standard 131.0-B-2. [Текст] – М. : CCSDS, 2011. – 93 p.

6. Морелос-Сарагоса, Р. Искусство помехоустойчивого кодирования. Методы, алгоритмы, применение [Текст] / Р. Морелос-Сарагоса. – М. : Техносфера, 2005. – 320 с.

7. Шульгин, В. И. Основы теории передачи информации. Часть 1 Экономное кодирование [Текст] / В. И. Шульгин. – М. : ХАИ, 2003. - 103с.

8. Шульгин, В. И. Основы теории передачи информации. Часть 2 Помехоустойчивое кодирование [Текст] / В. И. Шульгин. – М. : ХАИ, 2003. - 88с.

9. Золотарев, В. В. Помехоустойчивое кодирование. Методы и алгоритмы. Справочник. [Текст] / В. В. Золотарев, Г. В. Овечкин. – М. : Горячая линия - Телеком, 2004. - 126с.

10. Березюк, А. Г. Кодирование информации (двоичные коды) [Текст] / А. Г. Березюк, Н. Т. Андрущенко, С. С. Моощицкий. – М. : Издательское объединение «Вища школа», 1978. – 252 с.

11. Файнстейн, А. Основы теории информации [Текст] / А. Файнстейн. – М. : Изд-во иностранной литературы, 1960. - 144с.

12. Галагер, Р. Д. Коди з низькою щільністю перевірок на парність [Текст]: Пер. з англ. – М. : Мир, 1966. – 145 с.

13. Горюнов, А. Г. Телеконтроль и управление [Текст] / А. Г. Горюнов, С. Н. Ливенцов, А. А. Лисенок. – М. : ТПУ, 2008. – 160 с.

14. Хэмминг, Р. В. Теория кодирования и теория информации [Текст] : пер. с англ. – М. : Радио и связь, 1983. - 176с.

15. Галлагер, Р. Теория информации и надежная связь [Текст] / Р. Галлагер. – М. : Советское радио, 1974. - 720с.

16. Radio frequency and modulation systems – part 1 earth stations and spacecraft: Recommended standard 401.0-B. [Текст] – М. : CCSDS, 2013. – 120с.

17. Silage, D. Digital Communication Systems Using MATLAB and Simulink [Текст] / D. Silage. – M. : Bookstand Publishing, 2009. - 199p.