МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧОРНОМОРСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІМЕНІ ПЕТРА МОГИЛИ

На правах рукопису

КРАЙНИК ЯРОСЛАВ МИХАЙЛОВИЧ

УДК 004.312.26:519.725

ДОСЛІДЖЕННЯ ТА РОЗРОБЛЕННЯ ВИСОКОЕФЕКТИВНИХ ЧАСТКОВО ПАРАЛЕЛЬНИХ LDPC-ДЕКОДЕРІВ НА БАЗІ FPGA

05.13.05 – Комп’ютерні системи та компоненти

Дисертація
на здобуття наукового ступеня
кандидата технічних наук

Науковий керівник:
Мусієнко Максим Павлович,
д-р техн.наук, проф.

Миколаїв – 2015

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

МПП – матриця перевірки парності

ПЛІС – програмовна логічна інтегральна схема

BER – Bit Error Rate

DSP – Digital Signal Processing

FPGA – Field Programmable Gate Array

LDPC – Low Density Parity Check

SoC – System-on-Chip

VHDL – VHSIC Hardware Description Language

VHSIC – Very High-Speed Integrated Circuit

ВСТУП

Актуальність теми. Можливість передачі великих обсягів інформації багато в чому визначає сучасні технічні засоби, що використовуються суспільством: Інтернет, високошвидкісне цифрове телебачення, мобільні мережі та ін. Побудова систем з високою пропускною здатністю стала можливою як завдяки розвитку рівня техніки, так і теоретичним дослідженням у даній галузі. Обчислювальні потужності окремих типів мікросхем та комп'ютерних систем загалом постійно збільшуються. Теорії передачі інформації присвячені основні праці К. Шеннона, Р. Галлахера та ін. Результати їх досліджень отримали широке застосування в сучасних системах передачі інформації. Для підвищення надійності та швидкості передачі інформації широке застосування отримали завадостійкі коди. Завадостійкі коди передбачають додавання певної надлишкової інформації, на основі якої можливо виконати виправлення помилок, що виникли при передачі по каналу з шумами. Найкращими завадостійкими кодами є коди з низькою щільністю перевірки на парність (англ. Low Density Parity Check – LDPC). При цьому, для реалізації декодеру таких кодів найчастіше обирається апаратна платформа програмовних логічних інтегральних схем (ПЛІС; у роботі розглядаються ПЛІС, що відносять до типу Field Programmable Gate Array – FPGA), яка дозволяє забезпечити високу пропускну здатність та повторне використання ресурсів за рахунок реконфігурації. Частково паралельні LDPC-декодери знайшли найбільш широке застосування серед усіх типів декодерів. Питанням побудови частково паралельних LDPC-декодерів займались Дж. МакКей, Р. Ніл, Г. Фалькао, Т. Мосенін, К. Гілу та ін.

LDPC-коди на основі матриць перевірки парності (МПП) з довільним розташуванням значущих елементів забезпечують найкращі показники захищеності сигналу. Розглянуті дослідження вчених дозволяють вирішити широкий набір проблем, проте представлені в їх роботах моделі та методи не можуть бути застосовані для побудови декодерів LDPC-кодів на основі МПП з

довільним розташуванням значущих елементів через те, що це призведе до значного підвищення складності такого декодеру.

У зв'язку з цим актуальною є науково технічна задача вдосконалення архітектур частково-паралельних LDPC-декодерів на базі ПЛІС для кодів на основі матриці перевірки парності з довільним розташуванням значущих елементів.

Зв'язок з науковими програмами, планами та темами. Дисертаційна робота проводилась відповідно до задач таких програм: «Розроблення теоретичних основ забезпечення функціонування комп'ютерних систем на базі MCU/FPGA в умовах обмеженого енергозабезпечення» (номер державної реєстрації 0114U003952), в якій автор брав участь в якості виконавця, «Розробка LDPC-декодерів на базі ПЛІС» (номер державної реєстрації 0115U000017), в якій автор брав участь в якості відповідального виконавця, та «Розроблення поліметричних датчиків інформаційно вимірювальних систем з живленням елементів від енергії вимірювального сигналу» (номер державної реєстрації 0115U000316), в якій автор приймав участь в якості виконавця.

Мета дисертаційної роботи. Метою дисертаційного дослідження є підвищення ефективності побудови архітектур LDPC-декодерів для нерегулярних LDPC-кодів з довільним характером розташування значущих елементів та підвищення швидкодії таких декодерів за рахунок розробки моделей та методів побудови архітектур частково паралельних LDPC-декодерів на базі ПЛІС.

Для досягнення мети необхідно вирішити наступні завдання:

1. Провести аналіз існуючих методів організації архітектур частково паралельних LDPC-декодерів з метою виявлення недоліків їх використання при побудові LDPC-декодерів, що працюють з МПП з довільним розташуванням значущих елементів.

2. Удосконалити метод побудови частково паралельного LDPC-декодеру для матриць перевірки парності з довільним розташування значущих

елементів, що дозволить зменшити використання ресурсів пам’яті при реалізації.

3. Удосконалити модель організації паралельних обчислень для алгоритму мінімальної суми, що дозволить підвищити пропускну здатність декодеру.

4. Удосконалити модель запису/зчитування, що дозволить підвищити швидкодію LDPC-декодеру для матриць перевірки парності з довільним розташуванням значущих елементів.

5. Удосконалити моделі організації подвійного буферу для запису вхідного повідомлення, що дозволить зменшити час на очікування запису повідомлення та підвищити пропускну здатність.

6. Удосконалити метод побудови декодеру на основі алгоритму перестановки рядків матриці для попередньої обробки матриці перевірки парності, що дозволить виконати підключення двох блоків декодування, які працюють паралельно, та підвищити пропускну здатність.

7. Удосконалити метод побудови реконфігуровного частково паралельного LDPC-декодеру, що дозволить проводити обробку даних на основі матриці, на яку накладаються мінімальні обмеження.

8. Удосконалити метод побудови LDPC-декодеру з конвеєрною архітектурою для нерегулярних матриць перевірки парності з довільним розташуванням значущих елементів.

9. Виконати програмну та апаратну реалізацію розроблених положень та провести їх практичне тестування.

Об’єктом дослідження є декодування інформації з використанням LDPC-кодів на основі матриць перевірки парності з довільним розташуванням значущих елементів за допомогою частково паралельних декодерів.

Предметом дослідження методи та засоби організації архітектур частково паралельних LDPC-декодерів на базі мікросхем FPGA, що виконують декодування на основі матриці перевірки парності з довільним розташуванням значущих елементів.

Методи дослідження. Теоретичні результати дисертації базуються на: теорії цифрових автоматів для моделей та методів побудови частково паралельного LDPC-декодеру; теорії паралельних обчислень для організації обчислень у запропонованих моделях та методах; теорії передачі інформації для розроблених моделей та методів побудови систем; теорії прийняття рішень для визначення функціонального розподілу між пристроями у комплексній гетерогенній інформаційній системі.

Наукова новизна одержаних результатів:

–удосконалено метод побудови частково паралельного LDPC-декодеру з алгоритмом декодування мінімальної суми за рахунок введення додаткового проміжного етапу обчислення, що дозволило зменшити використання ресурсів блокової пам’яті при реалізації декодеру на базі мікросхем ПЛІС; для тестової реалізації декодеру для коду зі швидкістю 1/2 виграш у кількості збережених блоків пам’яті становив 254 блоки; максимальний виграш від використання даного методу становить зменшення використання блокової пам’яті у 2 рази;

–отримали подальший розвиток моделі підвищення пропускної здатності декодеру, зокрема, модель організації обчислень для частково паралельного LDPC-декодеру за рахунок виконання операції обчислення синдрому та обчислення значень вузлів перевірки паралельно з іншими операціями відповідно до отримання даних при декодування, що дозволило підвищити швидкодію декодеру, яка для тестового прикладу підвищилась в 1,4 рази, а в залежності від параметрів МПП може становити 1,1…1,8 разів; моделі організації LDPC-декодеру шляхом застосування подвійної буферизації вхідних повідомлень, що дозволило зменшити час, необхідний на запис вхідного повідомлення, а отже, підвищити швидкодію декодеру: за рахунок використання даних моделей вдається отримати виграш у пропускній здатності до 2 разів; модель частково паралельного LDPC-декодеру на основі організації паралельних черг запису/зчитування, що дозволило підвищити швидкодію декодеру в 1,1…1,5 рази;

–удосконалено метод організації реконфігуровного частково паралельного LDPC декодеру за рахунок організації чотирьохпортової пам’яті та принципів уникнення колізій, що дозволяє працювати з матрицями перевірки парності з довільним характером розташування значущих елементів; розроблений метод дозволяє підвищити швидкодію декодеру при реалізації на ПЛІС в 1,4…1,6 разів;

–удосконалено метод організації паралельних обчислень для частково паралельного LDPC-декодеру на основі попередньої обробки матриці перевірки парності, що дозволило використання двох блоків декодування, які працюють паралельно, за рахунок чого пропускну здатність частково паралельного LDPC декодеру вдалося збільшити у 2 рази; вдосконалений у роботі метод побудови конвеєрної архітектури LDPC-декодеру для МПП з довільним розташуванням значущих елементів дозволяє у 20…50 разів підвищити пропускну здатність LDPC декодеру.

Практичне значення одержаних результатів.

1. Розроблений схемотехнічний опис частково паралельного LDPC-декодеру на базі ПЛІС для матриць перевірки парності з довільним розташуванням значущих елементів, що дозволяє отримати швидкодію вищу, ніж існуючі продукти.

2. Розроблений опис мовою VHDL частково паралельного LDPC-декодеру з можливістю реконфігурації, який дозволяє проводити декодування на основі матриці перевірки парності, на структуру якої накладаються мінімальні обмеження.

3. Розроблений опис мовою VHDL LDPC-декодеру, що дозволяє зменшити використання ресурсів блокової пам’яті, що дозволяє виконати реалізацію декодеру на мікросхемах середнього цінового діапазону, в той час, як існуючі рішення передбачають використання лише мікросхем вищого цінового діапазону.

4. Розроблене програмне забезпечення дозволяє проводити тестування роботи декодеру та відповідність основних параметрів апаратній реалізації.

5. Результати дисертаційного дослідження впроваджені в навчальний процес у Чорноморському державному університеті імені Петра Могили (м. Миколаїв), Черкаському національному університеті ім. Б. Хмельницького, Черкаському державному технологічному університеті та на підприємствах ПАТ «Електротехнічний завод» РЕЛСіС (м. Київ, Україна), Straight Way Electronix, s.r.o. (м. Прага, Чехія), uPC Labs, (м. Стемфорд, США).

Особистий внесок здобувача. Основні результати дисертаційного дослідження здобувач отримав самостійно. В роботах, опублікованих у співавторстві [1, 2, 6-11, 13, 21, 22], здобувачеві належать: розроблена модель паралельної реалізації алгоритму мінімальної суми на базі ПЛІС [7]; розроблена модель паралельних черг запису/зчитування при декодуванні LDPC-кодів [8]; розроблено метод побудови реконфігурованого LDPC-декодеру [11]; розроблено метод підвищення швидкодії LDPC-декодеру [6]; розроблено метод виконання розподілу функцій між модулями у гетерогенних обчислювальних системах [1]; розроблено спосіб виконання функціонального розподілу між ПЛІС та МК при виконанні обчислень [10]; представлено побудову систем на базі USB [13]; розроблено метод виконання обчислень для модифікованих алгоритмів Брезенхема [2, 9]; розроблено метод побудови конвеєрної архітектури [21]; розроблений принцип побудови частково паралельного LDPC-декодеру зі зменшеним використанням блокової пам’яті [22].

Апробація результатів дисертації. Основні результати дисертаційного дослідження доповідались на 11 наукових конференціях: "Автоматизація та комп’ютерно-інтегровані технології у виробництві та освіті: стан, досягнення, перспективи розвитку" (Черкаси, 2014); IV міжнародній науково практичній конференції студентів, аспірантів та молодих вчених "Комп'ютерні науки для інформаційного суспільства" (Харків, 2013); І міжнародної науково практичної конференції "Проблеми інформатизації" (Київ, 2013); XV міжнародній науково-практичній конференції "Сучасні інформаційні та електронні технології" (Одеса, 2014); Міжнародній конференції студентів і молодих науковців "Сучасні інформаційні технології-2014" (Одеса, 2014); ІІІ міжнародній науково-

практичній конференції "Напівпровідникові матеріали, інформаційні технології та фотовольтаїка" (Кременчук, 2014); VII міжнародній науково-практичній конференції "Сучасні проблеми і досягнення науки в галузі радіотехніки, телекомунікацій та інформаційних технологій" (Запоріжжя, 2014); міжнародній науково-практичній конференції ІОН-2014 (Вінниця, 2014); Всеукраїнській науково-методичній конференції "Могилянські читання-2014" (Миколаїв, 2014); МНПК Ольвійський форум – 2015 (Миколаїв, 2015); міжнародній конференції «ELNANO» (Київ, 2015).

Публікації. За тематикою дослідження опубліковано 22 наукових праці, у тому числі 10 статей у фахових наукових виданнях (три з яких опубліковані в іноземних виданнях, що входять до РІНЦ; одна у вітчизняному виданні, що входить до індексу цитування Index Copernicus) та тези 11 доповідей на наукових конференціях (тези доповіді на міжнародній конференції «ELNANO-2015» включені до бази цитування Scopus). Також отримано 1 патент України на корисну модель.

Структура дисертації. Дисертація складається зі вступу, 4 розділів, висновків, додатків, списку використаних джерел, який містить 162 найменування вітчизняних та зарубіжних джерел. Загальний обсяг роботи – 173 сторінки, кількість рисунків – 73, кількість таблиць – 13.

РОЗДІЛ І
СТАН ПРЕДМЕТУ ДОСЛІДЖЕННЯ ТА ФОРМУЛЮВАННЯ ЗАДАЧ

1.1 Визначення предметної області дослідження

Завадостійкі коди з прямим виправленням помилок (англ. Forward Error Correction, FEC) – це коди, що здатні виявляти та виправляти помилки на стороні отримувача без необхідності запиту повторної відправки. Дана властивість забезпечується за рахунок передачі додаткової інформації [1-10].

Відповідно до того, який тип інформації є входом для коду, розрізняють:

− блокові коди – входом виступає блок інформації фіксованої довжини [10, с. 23-25];

− потокові або згорткові коди – входом виступає потік даних довільного розміру [8, с. 452-456].

Основними типами завадостійких кодів є:

− циклічні коди (коди Боуза-Чодурі-Хоквінгема [5], коди Хеммінга [7, с. 58-60], коди Ріда-Соломона [2, 8, 10]);

− потокові та блочні турбо-коди (англ. Turbo-Product Codes, TPC) [1, с. 271-293];

− коди з низькою щільністю перевірки на парність LDPC [11, 12].

Часто для підвищення ефективності виправлення помилок окремих кодів використовуються комбіновані коди, які можуть складатися з кількох різних кодів, та модифікації кодів [10, с. 169-200].

Порівняння ефективності виправлення помилок [13] при різних значеннях співвідношення сигнал/шум для основних типів завадостійких кодів наведено на рис. 1.1.

Таким чином, саме LDPC-коди – коди, що наближаються найбільше до границі Шеннона [14]. Даний тип кодів винайдений у 60-х рр. XX ст. [11], проте через відсутність технічних засобів для реалізації, не використовувались довгий час. У 90-х рр. LDPC-коди були повторно відкриті [12] та отримали

широке використаннч. Завдяки високій ефективності виправлення помилок [15-19] LDPC-коди знайшли широке застосування в багатьох стандартах високошвидкісної передачі даних:

- DVB-S2 [20];
- DVB-T2 та DVB-T2-Lite;
- DVB-C2;
- IEEE 802.11n (Wi-Fi) [21];
- IEEE 802.3 10-G BASE-T [22];
- IEEE 802.16e (WiMAX).

Також дані коди заплановані для використання у стандартах передачі, що знаходяться у стадії розробки, наприклад, DVB-NGH.

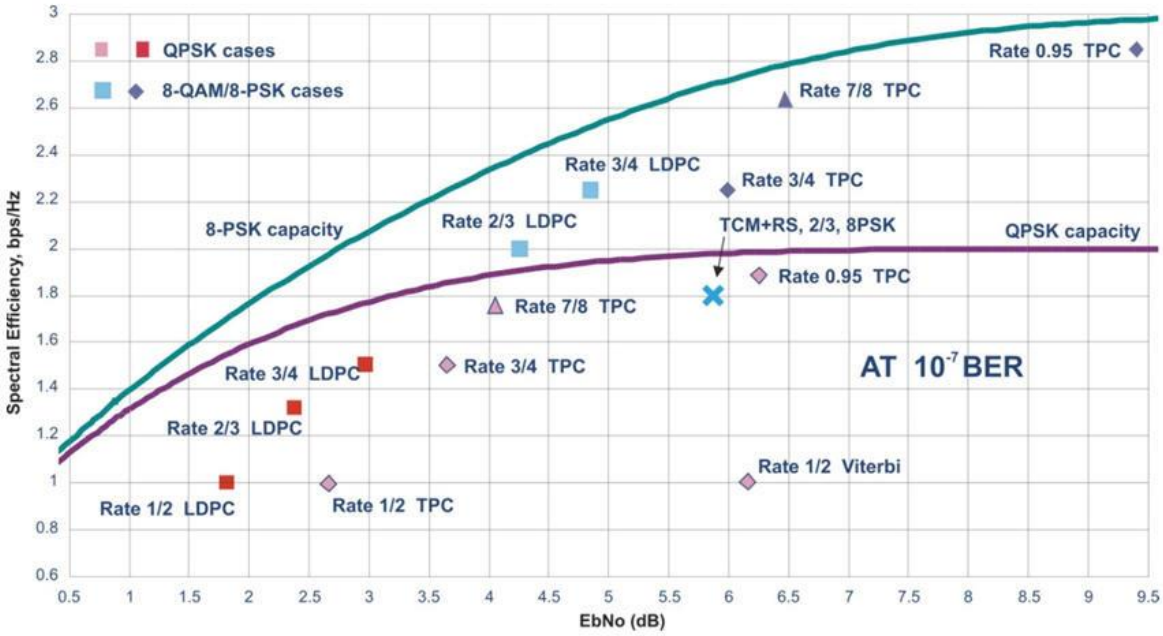


Рис. 1.1. Ефективність виправлення помилок завадостійкими кодами

Основними елементами, що характеризують LDPC-код є матриця перевірки парності (МПП) H, що містить M рядків та N стовпців, та породжуюча матриця G [7, с. 635]. При цьому, кількість значущих (відмінних від нуля) елементів у рядку матриці (вага рядка) є значно меншою за N:

$$w_r << N. \tag{1.1}$$

Таким чином матриця H є розрідженою. У даній роботі розглядаються лише двійкові LDPC-коди, тобто, такі коди, для яких значущим елементом у МПП є одиниця. Існують також недвійкові LDPC-коди [23].

У свою чергу, LDPC-коди також прийнято класифікувати відповідно до значення параметра ваги рядка w_r та ваги колонки w_c. Прийнято розрізняти:

− регулярні коди, для яких $w_r = const$ та $w_c = const$;

− нерегулярні коди, для яких $w_r \neq const$ і/або $w_c \neq const$ [8, с. 660, 2, с. 922-929].

Велика кількість проведених досліджень [15, 24-26] показала, що нерегулярні коди забезпечують кращі показники виправлення помилок. Порівняння ефективності регулярних та нерегулярних (англ. irregular LDPC, iLDPC) LDPC-кодів представлено на рис. 1.2 [15].

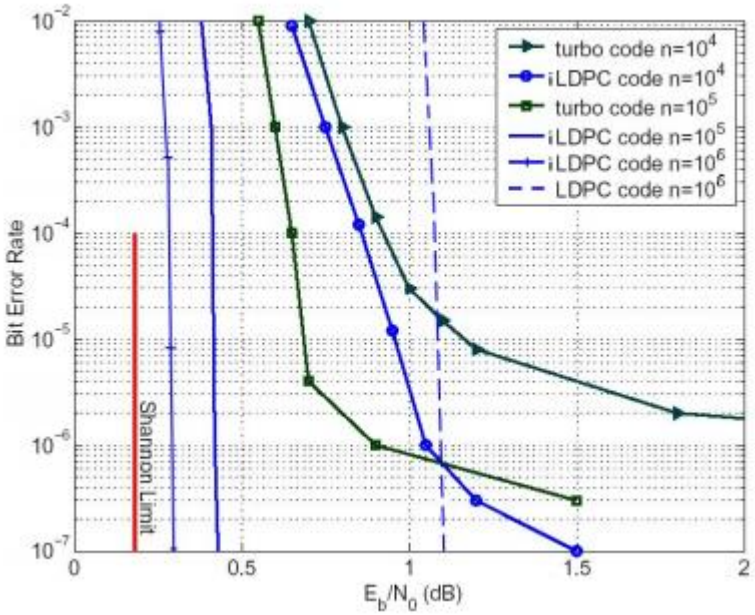


Рис. 1.2. Порівняння ефективності виправлення помилок регулярних та нерегулярних LDPC-кодів

Вища ефективність нерегулярних LDPC-кодів пояснюється тим, що біти в повідомленні захищені по-різному.

Окрім розділенням за параметрами кількості елементів у складових МПП, LDPC-коди розрізняють також за принципом отримання МПП:

1) випадкові [1, с. 312-315, 2, с. 920-922,];
2) комбіновані або систематичні [2, 8, 15].

Випадкові МПП отримують завдяки псевдовипадковим генераторам. При генерації на результуючу матрицю накладаються додаткові обмеження (відсутність циклів малої довжини, обмеження на кількість елементів у складовому елементі МПП і т.д.), внаслідок чого генерація таких матриць, особливо для довгих кодів, потребує значних обчислювальних ресурсів. Обмеження накладаються для того, щоб результуючий код мав високі показники виправлення помилок. Такі коди використовуються в тому випадку, коли передача інформації відбувається при низькому значенні співвідношення сигнал/шум та при реалізації протоколів передачі інформації, що потребують вищої якості виправлення помилок (для військових цілей, радіо- та супутникових систем). Візуалізація МПП з випадковим розташуванням значущих елементів наведена на рис. 1.3.

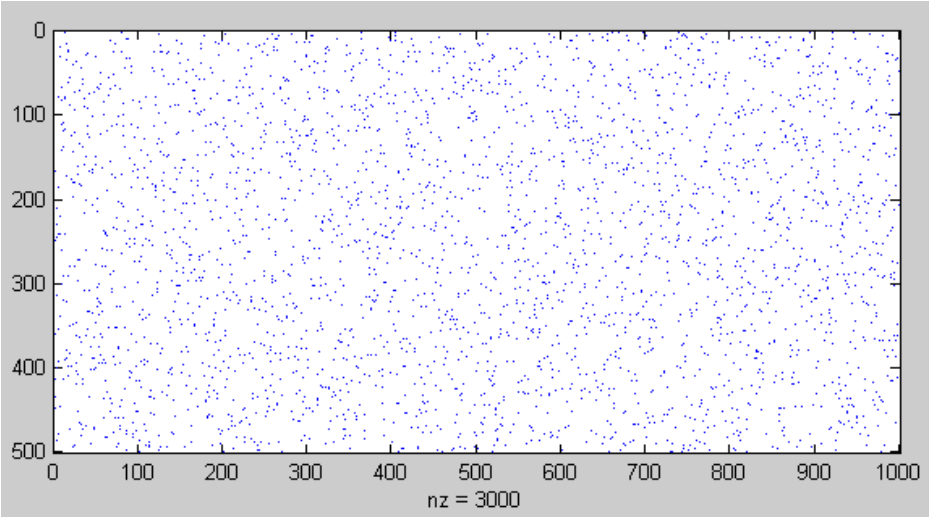


Рис. 1.3. МПП з випадковим розташуванням значущих елементів

Р. Галлахер у своїй основній роботі [11], в якій описані LDPC-коди, вказував на те, що найкращими властивостями для виправлення помилок наділені саме випадкові LDPC-коди великої довжини. Це підтвердили сучасні дослідження: випадкові коди зі швидкістю 1/2 дозволяють наблизитись до границі Шеннона на відстань 0.0045 дБ [16].

У свою чергу, послідовність отримання систематичних МПП чітко визначена. Часто комбіновані LDPC-коди отримують на основі формування МПП з одного базового елементу (підматриці) шляхом зсуву значущих елементів. До систематичних LDPC-кодів відносять такі коди, що використовують квазіциклічні МПП (використовуються у Wi-Fi та WiMAX) [2, с. 912-917], нерегулярні МПП з накопиченням (англ. irregular repeat accumulative, IRA) [8, с. 673] та ін. Саме такі коди застосовуються при реалізації згаданих високошвидкісних інтерфейсів. Спрощена структура МПП дозволяє спростити архітектуру компонентів системи передачі інформації та підвищити пропускну здатність. Візуалізація МПП, побудованої з використанням комбінованого підходу наведена на рис. 1.4.

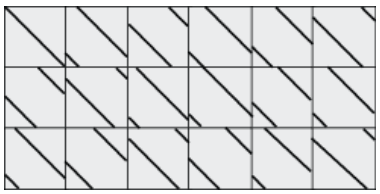


Рис. 1.4. Квазіциклічна МПП

Матриця на рис. 1.4 є прикладом квазіциклічної МПП.

Розглянемо використання LDPC-кодів у системах передачі інформації. Основними компонентами системи передачі інформації на стороні передавача та приймача є такі пари пристроїв:

- кодер/декодер;
- модулятор/демодулятор.

Також можуть використовуватись такі пристрої як скремблер, інтерлівер та ін. та їх відповідники на іншій стороні.

Загальна система передачі інформації [2, с. 1-3] з прямим виправленням помилок на прикладі LDPC-кодів представлена на рис. 1.5.

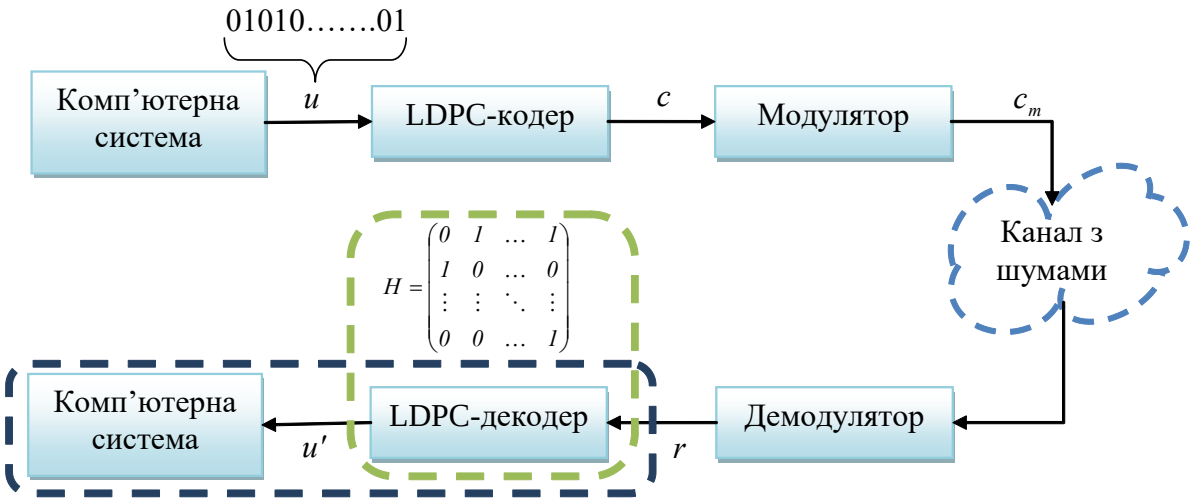


Рис. 1.5. Система передачі інформації на основі завадостійкого кодування

Після виходу з модулятора інформація передається за допомогою інформаційного каналу. У каналі передачі інформації наявні шуми, що спричиняють зміни сигналів [9, с. 128-129]. Задачею декодеру є виявлення та виправлення помилок, що виникли в ході передачі повідомлення.

Для кодування інформації на стороні передавача використовується породжуюча матриця G. За допомогою G отримують відповідну послідовність бітів перевірки парності для вхідного повідомлення u як

$$c = uG. \tag{1.2}$$

В залежності від того, як саме розташовуються біти повідомлення та біти парності код може бути [9, с. 134]:

– систематичним (біти повідомлення відділені від бітів парності);

– несистематичні (біти парності та біти повідомлення не розділені).

Декодер має обробити отриману інформацію на основі матриці перевірки парності G. Висновок про наявність або відсутність помилок у отриманій послідовності декодер робить, обчисливши значення синдрому [9, с. 135-141, 10, с. 77]

$$s = r \cdot H^T. \tag{1.3}$$

Для коректної кодової послідовності синдром є нульовим вектором [2, 8-10]:

$$s = (0_1, ..., 0_m). \tag{1.4}$$

Для пари матриці перевірки та породжуючої матриці вірною є рівність [2, 8-10]

$$H \cdot G^T = 0. \tag{1.5}$$

На основі алгоритмів декодування декодер має передати для подальшої обробки результуюче повідомлення u'. Причому, коректною вважається передача в тому випадку, якщо

$$u' = u. \tag{1.6}$$

Після того, як результуюче повідомлення виділено, можливе виконання обробки інформації комп'ютерною системою. Варто зазначити, що призначення інформації що передається, може бути різним: відео або аудіо, повідомлення, текст, специфічні команди та ін. Усе визначається протоколами взаємодії між системами.

Прикладом такої системи є гетерогенна навігаційно-комунікаційна комп'ютерна система [27-30], що здатна проводити обробку медіаконтенту. Подібна система відзначається наявністю обчислювальних модулів різних типів (CPU [31, 32], FPGA [33-36], MCU [37], GSM/GPS-модулів [38]), призначених для обробки різного типу інформації.

Іншим прикладом слугує система, призначена для прийому та оброки команд, що передаються за специфічним протоколом. Подібна система може розміщуватись у промисловому приміщені, для якого характерний високий рівень електромагнітного зашумлення. Даний фактор впливає на передачу інформації та може призвести до помилок. Використання LDPC-кодування при

передачі інформації дозволяє зменшити вплив фактору зашумлення та підвищити коректність роботи системи.

У подібних системах, враховуючи апаратну базу реалізації декодеру, окремі компоненти комп'ютерної системи, що виконують обробку інформації, реалізуються також з використанням апаратної платформи декодеру.

Серед усіх представлених компонентів системи передачі інформації найбільш важливим, з точки зору пропускної здатності, є декодер. Це пояснюється тим, що для декодування інформації, використовуються алгоритми, що мають велику обчислювальну складність та потребуються виконання великої кількості операцій. Внаслідок цього, пропускна здатність декодеру та системи може різко знижуватись.

Окрім того, визначено, що LDPC-кодами, що забезпечують найкращі показники виправлення помилок є нерегулярні LDPC-коди на основі МПП, яка згенерована випадковим чином. Реалізація декодеру для таких кодів ускладнюється випадковим характером розташування значущих елементів МПП, що ускладнює архітектуру декодеру та зменшує його пропускну здатність. Підтвердженням цього є відомі мікросхеми LDPC-декодерів [39, 40], які забезпечують пропускну здатність далеку від пропускної здатності високошвидкісних інтерфейсів.

Саме тому існує проблема побудови декодеру для випадкових LDPC-кодів, що здатен забезпечувати достатню пропускну здатність для передачі необхідної інформації.

1.2 Типи LDPC-декодерів та їх особливості

LDPC-декодери можливо класифікувати за різними ознаками. Найбільш поширеними загальними характеристиками класифікації виступають апаратна платформа реалізації.

Апаратною базою для LDPC-декодерів можуть виступати:

− процесори загального призначення [31, 32];

– графічні процесори (NVidia, ATI Radeon та ін.) [41, 42];

– спеціалізовані процесори [43];

– ПЛІС [33-36, 44-47] (розглядаються мікросхеми з архітектурою FPGA);

– спеціалізовані мікросхеми (наприклад, фірми AHA) [39, 40, 48, 49].

На рис. 1.6 представлені найбільш важливі характеристики вказаних апаратних платформ при реалізації LDPC-декодеру.

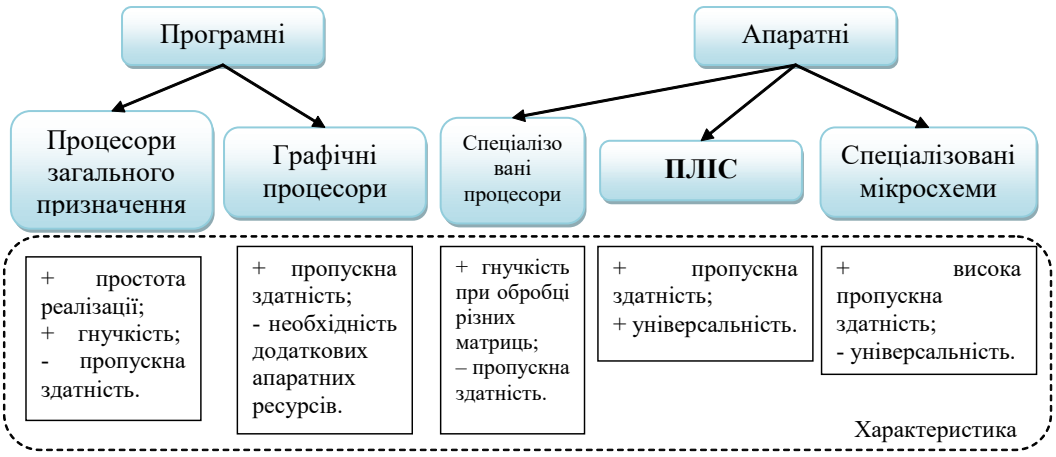


Рис. 1.6. Порівняння апаратних платформ для реалізації декодерів

Декодери, що реалізовані апаратно, зазвичай, переважають програмні декодери. Декодер для процесору загального призначення реалізується за допомогою мов програмування C, C++, Java [50] та ін. Реалізація декодеру для графічного процесору потребуватиме використання фреймворків OpenCL [51] або CUDA [52]. Апаратні декодери реалізуються з використанням мов схемотехнічного опису VHDL [53], Verilog та ін. Окрім цього, спеціалізований процесор потребує написання додаткової мікропрограми.

З точки зору повторного використання апаратних ресурсів та забезпечення пропускної здатності, декодери на базі ПЛІС можуть бути перепрограмовні та використовуватись повторно, демонструючи при цьому високу швидкодію [44-47]. Саме тому апаратна платформа ПЛІС активно використовується для вирішення задачі побудови декодеру, про що свідчить велика кількість досліджень, присвячених LDPC-декодерам на базі ПЛІС. Окрім того, варто зазначити, що більшість методів побудови декодерів для спеціалізованих

мікросхем можуть також застосовуватись і для ПЛІС. Тому вони також будуть розглянуті далі.

З точки зору реалізації декодерів у ПЛІС, основними загальними проблемами можна назвати:

1) забезпечення високої швидкодії декодеру;
2) використання апаратних ресурсів.

Методи побудови архітектур декодерів на базі ПЛІС часто передбачають активне використання апаратних ресурсів та потребують для реалізації мікросхем верхнього цінового діапазону. Це може значно збільшити вартість системи, оскільки вартість мікросхем середнього та верхнього цінового діапазону може відрізнятися більше ніж на порядок.

Особливо актуальною проблема використання ресурсів ПЛІС для реалізації LDPC-декодеру є для випадку МПП з випадковим розташуванням значущих елементів. Це пов’язано з тим, представлення матриці не може бути спрощеним та потребує ускладнення структури декодеру. Ускладнення структури декодеру також впливає на можливість реалізації паралельних обчислень при декодуванні, а, відповідно, і на пропускну здатність.

Іншою формою представлення матриці перевірки парності є граф Таннера [54, 55]. Він складається з двох типів вузлів:

1) вузлів перевірки;
2) вузлів значень.

Даний граф використовується для опису процесу ітеративного декодування для LDPC-кодів при реалізації різних алгоритмів. Наявність зв’язків між вузлами графа визначається на основі розташування значущих елементів. Кожен рядок у матриці відповідає вузлу перевірки, а кожен стовпець – вузлу значень. Обмін інформацією за ребрами графа відбувається в обох напрямках – від вузлів перевірки до вузлів значень та навпаки. У випадку нерегулярної МПП [56] кількість зв’язків між вузлами не є однаковою. Загальне представлення графу Таннера представлено на рис. 1.7.

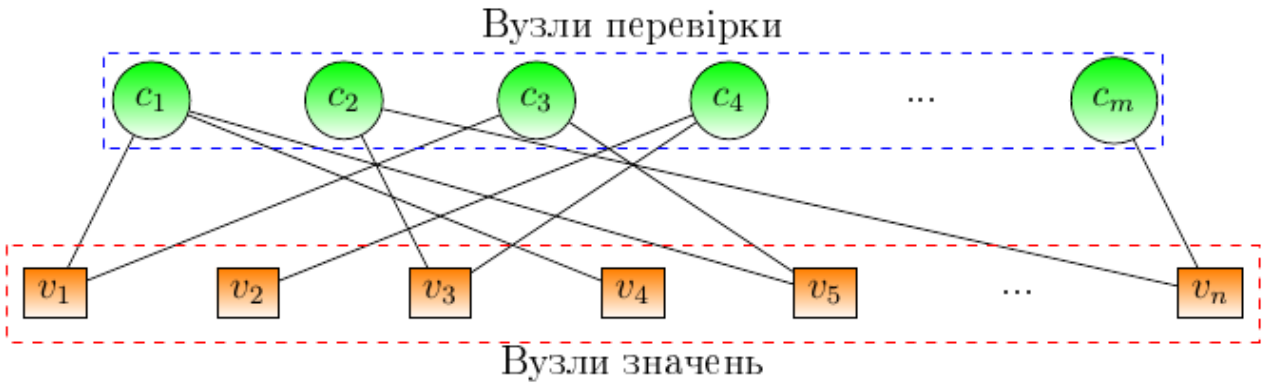


Рис. 1.7. Загальне представлення графу Таннера

Відповідно до повноти відображення графу Таннера на реалізацію, розрізняють такі декодери:

1) послідовні [57] – відображається одна пара вузлів;
2) повністю паралельні [58] – часткове відображення вузлів;
3) частково паралельні [44-49, 59-61] – відображення всіх вузлів графу.

Порівняння даних типів декодерів за пропускною здатністю та складністю реалізації наведено на рис. 1.8.

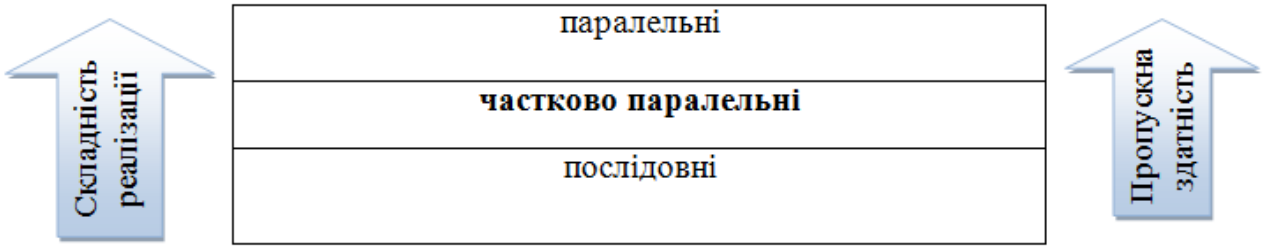


Рис. 1.8. Порівняння типів декодерів за повнотою відображення графу Таннера

Саме на основі параметрів пропускної здатності та складності реалізації визначається тип декодеру. Послідовний декодер забезпечує найнижчу пропускну здатність, але є найпростішим для реалізації. Повністю паралельний декодер теоретично повинен показувати найкращу швидкодію. Проте, в залежності від структури МПП, організація повністю паралельного декодеру для сучасного рівня розвитку техніки не є складною, особливо для кодів великої довжини. Особливо складними для реалізації є декодери для

випадкових МПП, оскільки необхідно забезпечити нерегулярні зв’язки між обчислювальними елементами декодеру.

Найбільш збалансованими за параметрами пропускної здатності та складності є частково паралельні декодери. Саме тому вони отримали найбільше поширення.

Таким чином, у роботі проводяться дослідження для частково-паралельних LDPC-декодерів на базі ПЛІС. Визначено, що для декодерів на базі ПЛІС є актуальною розробка моделей та методів побудови декодерів, що дозволяють мінімізувати використання ресурсів та забезпечити високу пропускну здатність.

1.3 Алгоритми декодування LDPC-кодів

Для алгоритмів декодування LDPC-кодів використовуються наступні види представлення вхідної інформації:

− жорсткі вхідні дані (у випадку двійкового LDPC-коду – 0 або 1) про кожен символ повідомлення [1, с. 309];

− м’які вхідні дані (значення, що належать до визначеного діапазону; можуть мати представлення як у вигляді цілих чисел, так і чисел з десятковою комою) [62-64].

З точки зору реалізації, жорстке декодування є простішим, однак демонструє гірші результати щодо виправлення помилок у вхідному повідомленні. У той же час, використання м’яких вхідних даних дозволяє отримати більше інформації про вхідний сигнал, за рахунок чого показує вищу ефективність декодування (рис. 1.9). Зважаючи на значний виграш у ефективності декодування, в системах, що потребують надійної роботи при низьких значеннях співвідношення сигнал/шум, використовуються м’які вхідні дані.

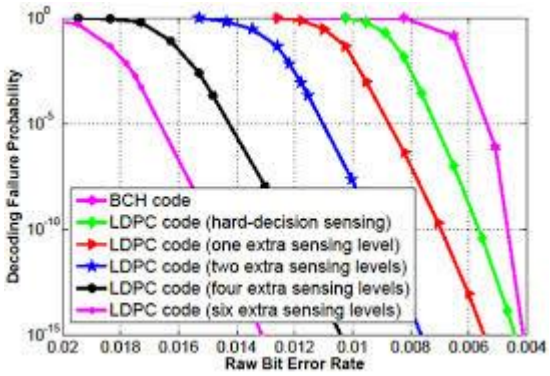


Рис. 1.9. Порівняння ефективності декодування при використанні м’яких та жорстких вхідних даних

Найбільш поширеними алгоритмами, що використовують м’які вхідні дані, є:

1. Алгоритм поширення довіри (англ. Belief Propagation) [1, с. 309-312].
2. Алгоритм суми добутків (англ. Sum Product Algorithm) [65-67].
3. Алгоритм мінімальної суми (англ. Min Sum Algorithm) [68].

Усі вказані алгоритми є ітеративними алгоритмами на основі обміну повідомленнями між вузлами графу Таннера (англ. Message Passing Algorithm, MPA).

Крім того, для декодування LDPC-кодів використовуються також алгоритми, які є загальними для декодування всіх типів завадостійких кодів [69, 70].

Алгоритм поширення довіри для декодування LDPC-кодів запропонований винахідником LDPC-кодів Р. Галлахером. У свою чергу, інші два алгоритми є спрощеними похідними версіями алгоритму поширення довіри. Вони дають можливість спростити реалізацію декодування, але, при цьому, втрачають у ефективності декодування. Проте такий програш у багатьох випадках є допустимим з урахуванням того, що бюджет каналу [6] має формуватися з запасом відносно ефективної границі виправлення помилок.

Саме тому алгоритм мінімальної суми є алгоритмом, що найчастіше обирається для реалізації LDPC-декодерів: даний алгоритм застосовується в декодерах, представлених у роботах [10-11]. Факт широкого використання

даного алгоритму пояснюється простотою основних операцій, які використовуються: суми та порівняння двох значень з визначенням мінімального за модулем. Завдяки великій поширеності даного алгоритму, крім його оригінальної версії, з'явилися модифікації [71-75], що враховують особливості апаратної реалізації та підвищують можливості паралельної обробки даних: алгоритм λ-min [71], пошаровий алгоритм мінімальної суми [75], алгоритм з розділенням рядків [74].

У даній роботі проводиться розробка методів побудови архітектур LDPC-декодерів кодів на основі МПП з випадковим розташуванням значущих елементів, які використовують алгоритм мінімальної суми.

1.4 Аналіз методів побудови архітектур LDPC-декодерів на базі ПЛІС

Як вже зазначалось, проблемі побудові архітектур LDPC-декодерів на базі ПЛІС присвячена велика кількість наукових робіт [44-47, 57, 60, 61, 76-84]. Результати, які отримані в цих роботах, демонструють, що LDPC-декодер здатен забезпечувати швидкість обробки в десятки Гбіт/с.

У вказаних роботах запропоновані методи, що дозволяють реалізувати LDPC-декодер. Незважаючи на відмінності, що характерні для всіх методів, можна виділити загальні риси архітектури декодеру.

1.4.1 Загальна архітектура частково-паралельного LDPC-декодеру

Даний тип декодерів забезпечує обробку певної підмножини вузлів з графу Таннера. Причому обробці можуть підлягати як вузли перевірки, так і вузли значень, так і обидва типи вузлів одночасно. Загальна архітектура декодеру забезпечує відображення частини вузлів та їх паралельну обробку. У такому випадку основними компонентами, які забезпечують обробку є обробники (процесори) значень перевірки та значень бітових вузлів. Взаємодія двох типів обробників відбувається за допомогою пам'яті. Зв'язок між обробниками забезпечується за рахунок мережі з'єднань вузлів. Дана мережа може

змінювати підключення між обробниками в залежності від зміни керуючих сигналів. Для управлінням послідовністю виконання операцій у ході декодування наявний окремий контролер. Основними функціями контролера є:

1) зміна сигналів шини адреси;
2) контроль за станом мережі з’єднань;
3) подача керуючих сигналів для обробників (активність/неактивність конкретного обробника, необхідність виконання запису та ін.).

Графічне представлення загальної архітектури декодеру наведено на рис. 1.10 [44, с. 69].

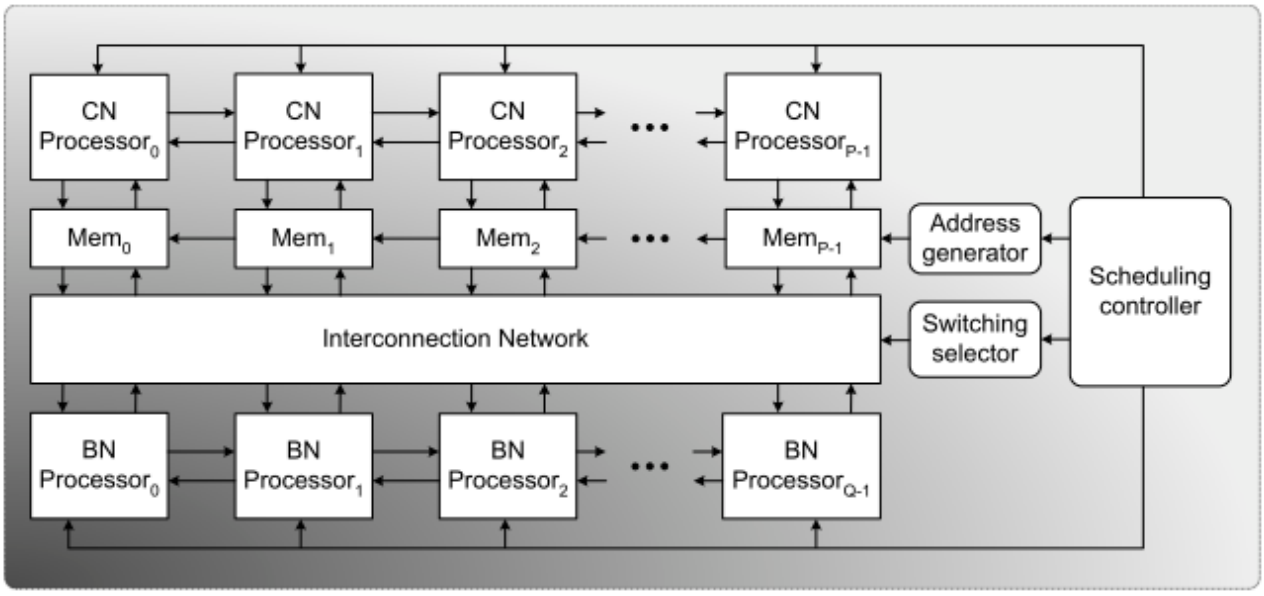


Рис. 1.10. Представлення загальної архітектури LDPC-декодеру

Особливості підключення компонентів системи та кількість компонентів визначаються на основі МПП та вимог до декодеру відносно пропускної здатності. Зі збільшенням необхідної швидкодії збільшується кількість обробників та ускладнюється структура мережі з’єднань між компонентами. Це, у свою чергу, ускладнює реалізацію на ПЛІС, оскільки збільшується необхідна кількість ресурсів. За такої ситуації необхідно враховувати особливості МПП та організацію операцій у ході декодування для того, щоб забезпечити необхідну пропускну здатність [81-82].

Прикладом архітектури, що використовує особливості МПП є архітектура [44], яка наведена на рис. 1.11. Декодер на основі даної архітектури призначений для обробки сигналу за стандартом DVB-S2. Зазначений стандарт вимагає швидкості обробки у 90 Мбіт/с від декодеру та використовує кодові слова довжиною 16200 (короткі слова) та 64800 (довгі слова) біт [20]. МПП для даного стандарту відноситься до типу IRA.

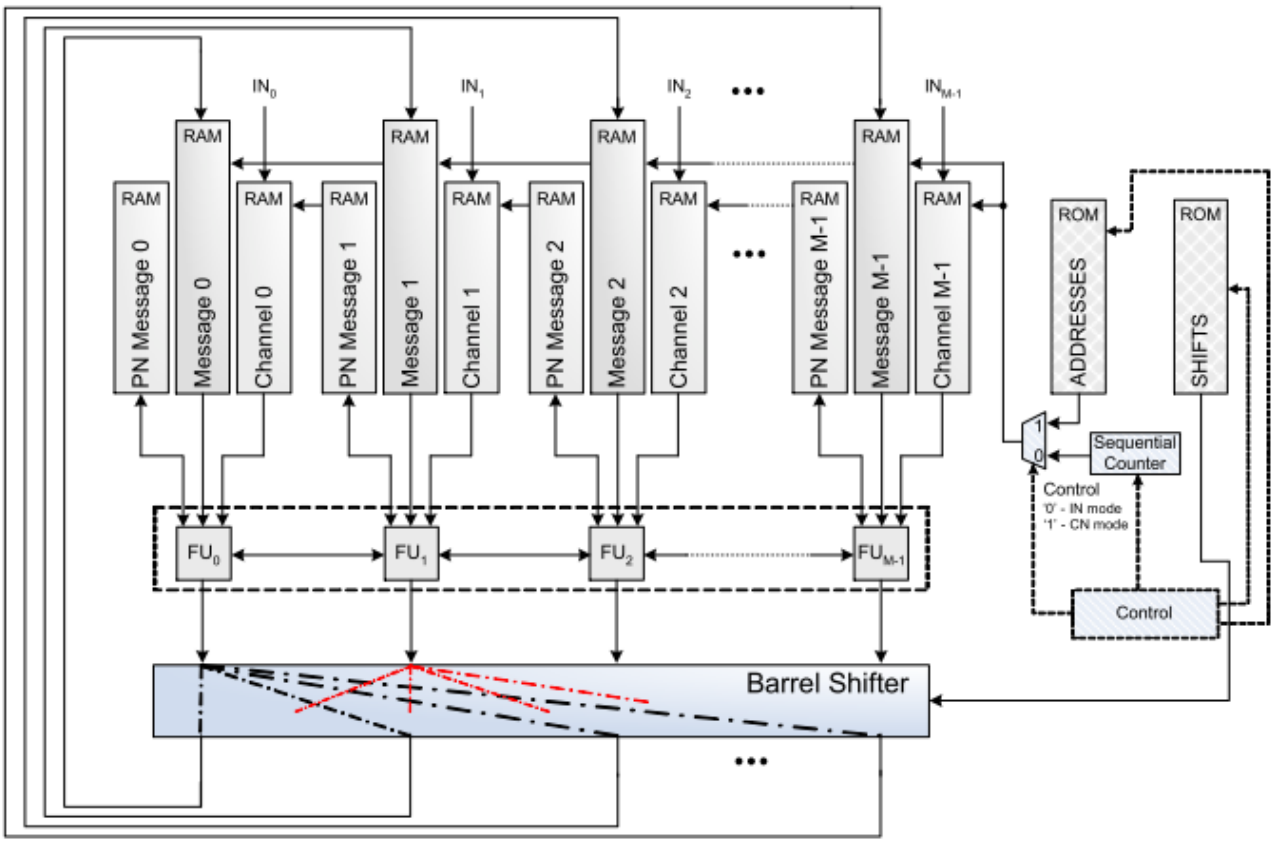


Рис. 1.11. Архітектура декодеру для прийому DVB-S2

Велика довжина кодового слова та високе значення необхідної пропускної здатності зумовлюють необхідність забезпечення паралельних операцій у процесі декодування. Дана проблема вирішується за рахунок наявності великої кількості обробників (360, 180, 90 та 45) та реалізації мережі з'єднань у вигляді універсального регістру зсуву (barrel shifter). Така реалізація є можливою завдяки особливостям МПП, для якої наступне значення індексу значущого елементу в межах певного діапазону може визначатись на основі попереднього додаванням фіксованого значення.

Основними обчислювальними елементами декодеру є обробники (процесори). Розрізняють три типи обробників у складі декодеру:

1) обробники вузлів перевірки [48, с. 31];

2) обробники вузлів значень [48, с. 32];

3) комбіновані обробники – об’єднують операції обробки вузлів перевірки та значень [1, с. 319-323, 44, с. 78-80, 88].

Розглянемо особливості обробників на прикладі комбінованого обробника, представленого на рис. 1.12 [44].

Даний процесор реалізує операції, що використовуються в алгоритмі мінімальної суми. Для досягнення максимальної пропускної здатності операції обчислень виконуються паралельно для всіх значень у рядку (для вузлів перевірки) або стовпці (для вузлів значень). Спільною рисою обробки вузлів обох типів є необхідність організації каскаду у вигляді операцій суми (для вузлів значень) та знаходження мінімуму (для вузлів перевірки). Таке рішення характерне для більшості розглянутих робіт [10-12].

Основна складність реалізації декодеру LDPC-кодів на основі МПП з довільним розташуванням значущих елементів з використанням розглянутої загальної архітектури полягає у тому, що елементи матриці розташовані випадковим чином. Це призводить до того, що:

1) збільшується обсяг пам’яті, необхідний для збереження індексів та результатів обробки, а також використання логічних ресурсів ПЛІС;

2) організація паралельних обчислень є ускладненою через можливість колізій доступу та обробки, що призводить до значного програшу у пропускній здатності; крім того, ускладнюється внутрішня структура мережі з’єднань між процесорними блоками декодеру.

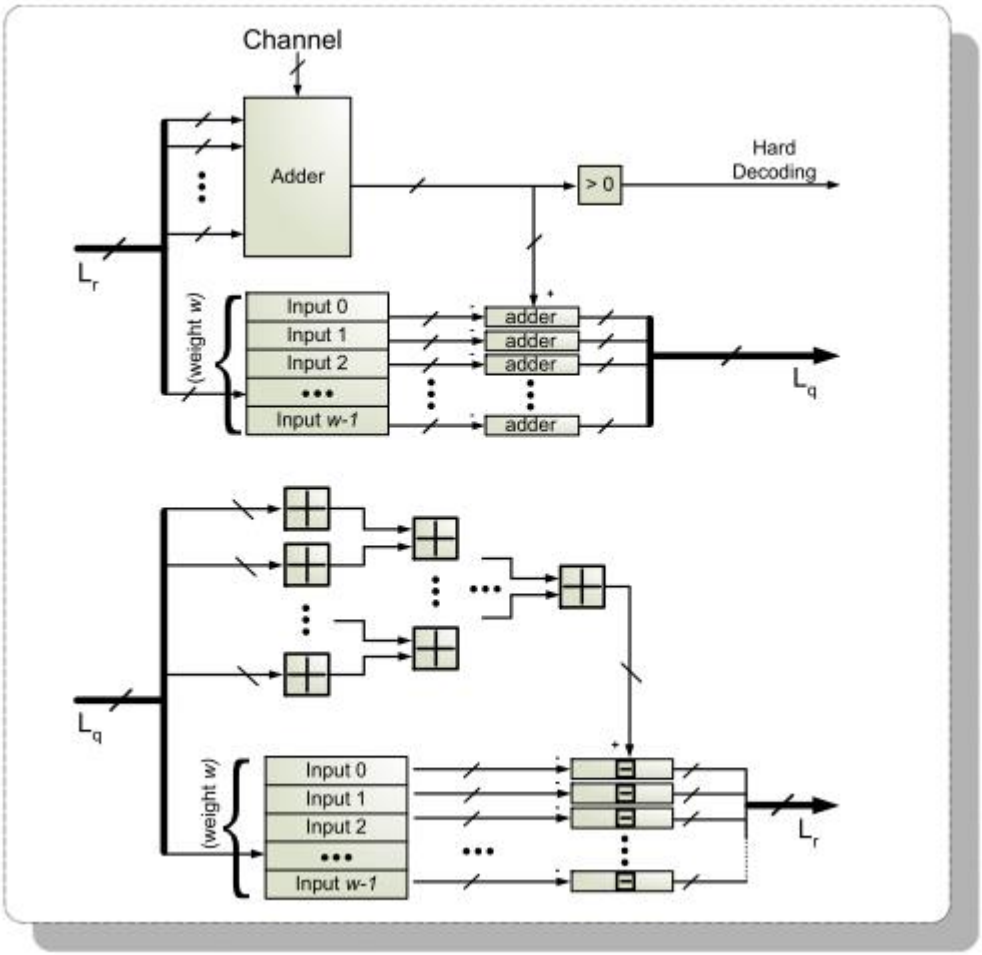


Рис. 1.12. Структура комбінованого обробника

Таким чином, розглянута загальна архітектура LDPC-декодеру не забезпечує характеристик, які дозволяють реалізувати декодер для LDPC-кодів на основі довільних МПП. Тому розробка моделей та методів, які дозволяють реалізувати такий декодер є актуальною та важливою.

1.4.2 Забезпечення пропускної здатності декодеру

Проблема підвищення пропускної здатності вирішується в роботах [44-47, 57, 60, 61, 76-97]. Основним засобом підвищення пропускної здатності LDPC-декодеру є організація паралельної обробки даних у процесі декодування. При цьому може використовуватись як дрібнозернистий паралелізм (паралельна обробка кількох повідомлень від вузлів), так і великозернийстий (паралельна обробка кількох кодових слів). Крім того, для забезпечення високої швидкодії обчислення при декодуванні конвеєризують.

Конвеєризовані обчислення характерні для більшості розглянутих декодерів [44-47, 57, 60, 61, 76-97]. У роботі [84] запропонована архітектура (рис. 1.13), яка дозволяє організувати конвеєризовані обчислення для окремих типів матриць.

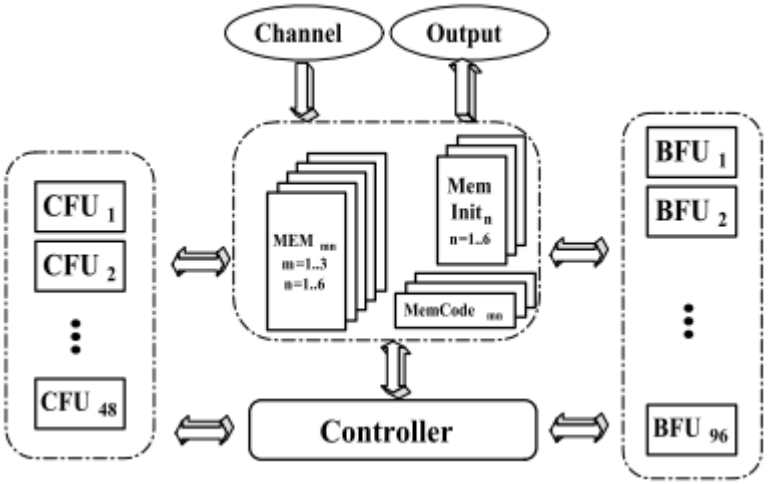


Рис. 1.13. Архітектура для реалізації конвеєризованих обчислень

Дана архітектура передбачає одночасне виконання зчитування значень, обробку та запис для кожного рядка та стовпця з урахуванням зсуву, за рахунок чого вдається уникнути колізій доступу та отримання некоректного результату при обробці. Проте дана обставина спричинена тим, що використовуються лише спеціальні матриці, на структуру яких накладені обмеження.

Алгоритм мінімальної суми та його модифікації добре піддаються розпаралелюванню. У роботах [48, 74] представлена модифікація алгоритму мінімальної суми – Split-Row. Заміна операції обміну повідомленням між вузлами перевірки на множення на значення вагового коефіцієнту дозволяє зменшити кількість зв’язків між вузлами та підвищити паралельність обробки. На основі даної модифікації розроблена архітектура декодеру (рис. 1.14) [48].

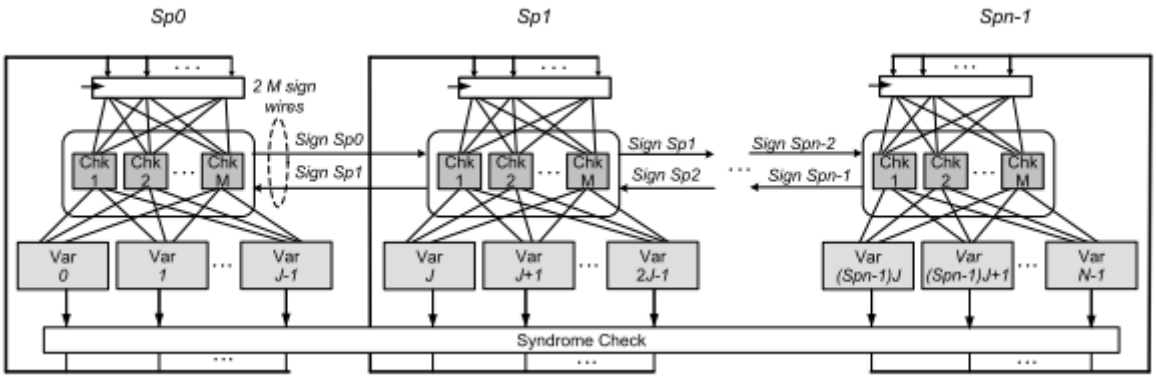


Рис. 1.14. Архітектура декодеру для алгоритму Split-Row

Проте дана архітектура передбачає розташування значущих елементів МПП у межах певного діапазону, тому не може бути використання для довільних МПП.

Враховуючи складність або неможливість застосування існуючих моделей та методів підвищення пропускної здатності декодерів для обробки довільних LDPC-кодів за рахунок паралельних обчислень, для таких кодів паралельні операції замінюються на послідовні. Це значно впливає швидкодію декодеру. Тому проблема підвищення швидкодії декодеру довільних LDPC-кодів ж актуальною та важливою.

1.4.3 Забезпечення підвищення ефективності використання ресурсів пам’яті декодеру

Роботи [83, 85, 98-102] присвячені зменшенню складності реалізації архітектур LDPC-декодерів. У роботі [85] представлена архітектура LDPC-декодеру для квазіциклічних МПП, яка дозволяє зменшити використання пам’яті, що відведена на збереження проміжних результатів обчислень (значень вузлів перевірки). Графічне представлення архітектури наведено на рис. 1.15. У роботі [93] представлена архітектура зі низьким використанням ресурсів, але вона орієнтована на спеціальні типи МПП.

Дана архітектура характеризується тим, що для збереження значень вузлів перевірки використовується мінімальна кількість біт. Тим не менш, декодер на основі такої архітектури зможе працювати лише з квазіциклічними матрицями,

для яких достатньо збереження лише одного значення адреси на окремий діапазон – інші значення можуть бути обчислені на основі інкременту поточного.

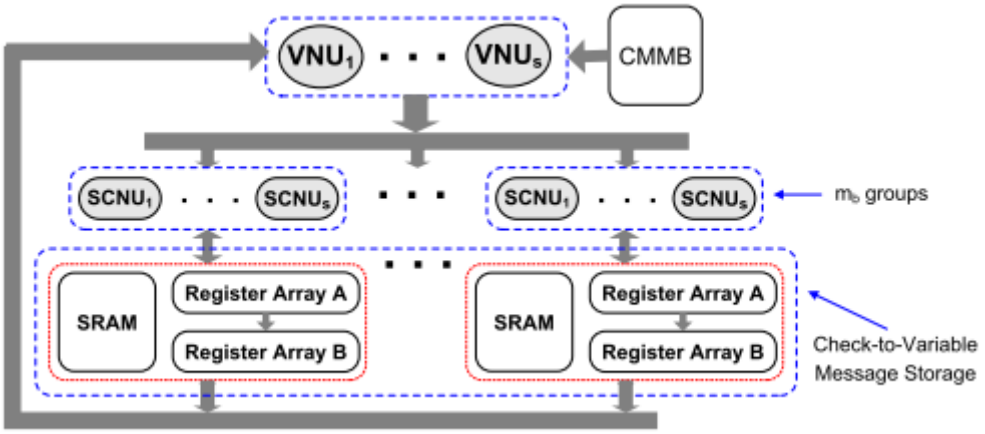


Рис. 1.15. Архітектура декодеру для квазіциклічних МПП

Проте для довільних МПП таке спрощення не є можливим, а тому необхідне збереження усіх значень адрес у пам’яті, що значно підвищує її використання. Крім того, для даної архітектури індекси є необхідними як для вузлів перевірки, так і для вузлів значень, що фактично означатиме підвищення використання ресурсу пам’яті на пам’ять адрес у два рази.

Саме тому проблема підвищення використання пам’яті для декодеру LDPC-кодів на основі довільних МПП є актуальною.

1.4.4 Реконфігурація LDPC-декодеру на базі ПЛІС для обробки різних МПП

Процес реконфігурації у зміні налаштувань або функцій, які виконує комп’ютерна система. Розрізняють наступні види реконфігурації, які характерні для ПЛІС:

− статична – передбачає зупинку роботи ПЛІС та перепрограмування відповідними апаратними засобами;

− динамічна – заміна одного функціонального блоку іншим, який знаходиться на кристалі ПЛІС.

У випадку динамічної реконфігурації зупинка роботи ПЛІС не потрібна. Функціональні блоки, які беруть участь в процесі динамічної реконфігурації повинні мати однакову кількість входів та виходів.

З точки зору LDPC-декодеру реконфігурація означає можливість зміни МПП для обробки повідомлення без перепрограмування ПЛІС [103, 104]. Реконфігурація забезпечується за рахунок завантаження нових індексів, що описують МПП, у пам’ять декодеру та зміни налаштувань роботи відповідно до нової МПП.

Основною перевагою реконфігуровного LDPC-декодеру є можливість його використання для декодування на основі різних МПП. При цьому, обмеження відносно структури МПП зменшуються. Як правило, такі декодери демонструють нижчу пропускну здатність, ніж оптимізовані під конкретну матрицю (тип матриць) декодери. Тим не менш, реконфігуровні декодери є універсальними, тому здатні забезпечити економію коштів при необхідності зміни МПП та її параметрів (довжина кодового слова, кількість значущих елементів у окремому рядку та ін.) в заданих межах.

Існуючі архітектури реконфігуровних LDPC-декодерів [103, 104] дозволяють працювати з кількома типами матриць без перепрограмування, проте вони накладають обмеження на нову МПП. При виході за границі даних обмежень необхідно виконувати заміну (розробляти новий схемотехнічний опис декодеру, купувати новий IP-блок та ін.). Тому розробка моделей та методів побудови архітектури реконфігуровного LDPC-декодеру, який накладає мінімальні обмеження на структуру МПП є важливою та актуальною.

1.5 Організація систем обробки інформації на базі ПЛІС

Розвиток систем передачі та обробки інформації призвів до того, що обчислювальних ресурсів одного модуля недостатньо для виконання всіх функцій, які покладаються на систему. Тому для розширення функціональності систем до їх складу залучають додаткові модулі різних типів. Системи, які

використовуються різні типи обчислювальних модулів є гетерогенними комп'ютерними системами. Прикладом такої системи є навігаційно-комунікаціна гетерогенна мультипроцесорна система (НКГМС), яка забезпечує обробку даних про позиціонування, передачу та обробку аудіо- та відеоінформації. Внутрішня структура НКГМС представлена на рис. 1.16 [27-30].

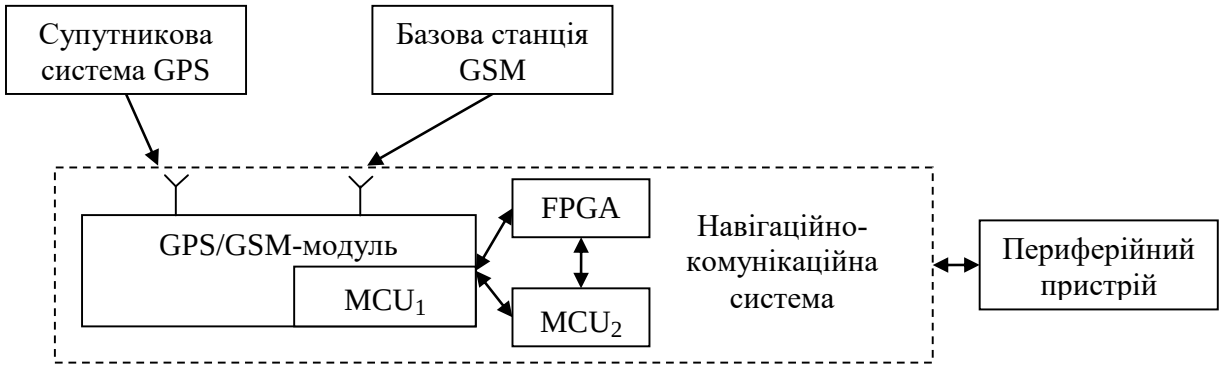


Рис. 1.16. Приклад внутрішньої організації НКГМС

До складу НКГМС в якості керуючих пристроїв найчастіше входять MCU, FPGA, GPS/GSM-модулі. Збільшення функціональних можливостей у таких системах часто проводиться простим додаванням модулів до системи з метою реалізації певної функції. При цьому не враховуються можливості наявних модулів для розподіленої реалізації даних функцій.

У таких системах ПЛІС може виконувати одразу декілька функцій за рахунок розміщення відповідних модулів, наприклад, прийом сигналу та його LDPC-декодування, виконання арифметичних розрахунків та ін. При цьому, інші модулі у складі системи можуть так само виконувати обчислювальні операції, тому важливим є виконання оптимального розподілу обчислювальних функцій між обчислювальними модулями.

Прикладом обчислювальної системи, яка потребує реалізації обчислень на базі ПЛІС може бути віддалений 3D-принтер, який розміщується в приміщення з високим рівнем завад. У такому випадку можливе використання LDPC-кодів для забезпечення надійної передачі даних. У такій системі для побудови

геометричних примітивів можуть використовуватись алгоритми Брезенхема [105, 106]. Реалізація обчислень на базі ПЛІС дозволить отримати виграш у швидкості обчислень.

1.6 Формулювання задач дослідження

Визначивши предметну область дослідження та проаналізувавши існуючи моделі, методи та засоби побудови LDPC-декодерів, визначено, що створення архітектур декодерів для МПП з довільним розташуванням значущих елементів дозволяє отримати значні результати завадостійкості при передачі сигналу, проте проблемі створення таких архітектур приділяється менша увага ніж для спеціальних типів МПП. Виявлено, що декодери кодів на основі довільних МПП при використанні існуючих архітектур, є складними в реалізації, оскільки потребують значних апаратних ресурсів при значній втраті у пропускній здатності. Таким чином, реалізація декодерів на основі розглянутих архітектур є недоцільною. Зважаючи на це, поставлені наступні задачі дослідження, які спрямовані на вирішення вказаних проблем:

1. Провести аналіз існуючих методів побудови архітектур частково паралельних LDPC-декодерів.
2. Розробити метод побудови частково паралельного LDPC-декодеру для матриць перевірки парності з довільним розташування значущих елементів, що дозволить зменшити використання ресурсів пам’яті при реалізації.
3. Розробити модель організації паралельних обчислень для алгоритму мінімальної суми, що дозволить підвищити пропускну здатність декодеру.
4. Розробити модель запису/зчитування, що дозволить підвищити швидкодію LDPC-декодеру для матриць перевірки парності з довільним розташуванням значущих елементів.
5. Розробити моделі організації подвійного буферу для запису вхідного повідомлення, що дозволить зменшити час на очікування запису повідомлення та підвищити пропускну здатність.

6. Розробити алгоритм перестановки рядків матриці для попередньої обробки матриці перевірки парності, що дозволить виконати підключення двох блоків декодування, що працюють паралельно та підвищити пропускну здатність.

7. Розробити метод побудови реконфігуровного частково паралельного LDPC-декодеру, що дозволить проводити обробку даних на основі матриці, на яку накладаються мінімальні обмеження.

8. Розробити метод побудови LDPC-декодеру з конвеєрною архітектурою для нерегулярних матриць перевірки парності з довільним розташуванням значущих елементів.

9. Виконати програмну та апаратну реалізацію розроблених положень та провести їх практичне тестування.

На основі вказаних задач побудована структурно-логічна схема дисертаційного дослідження (рис. 1.17).

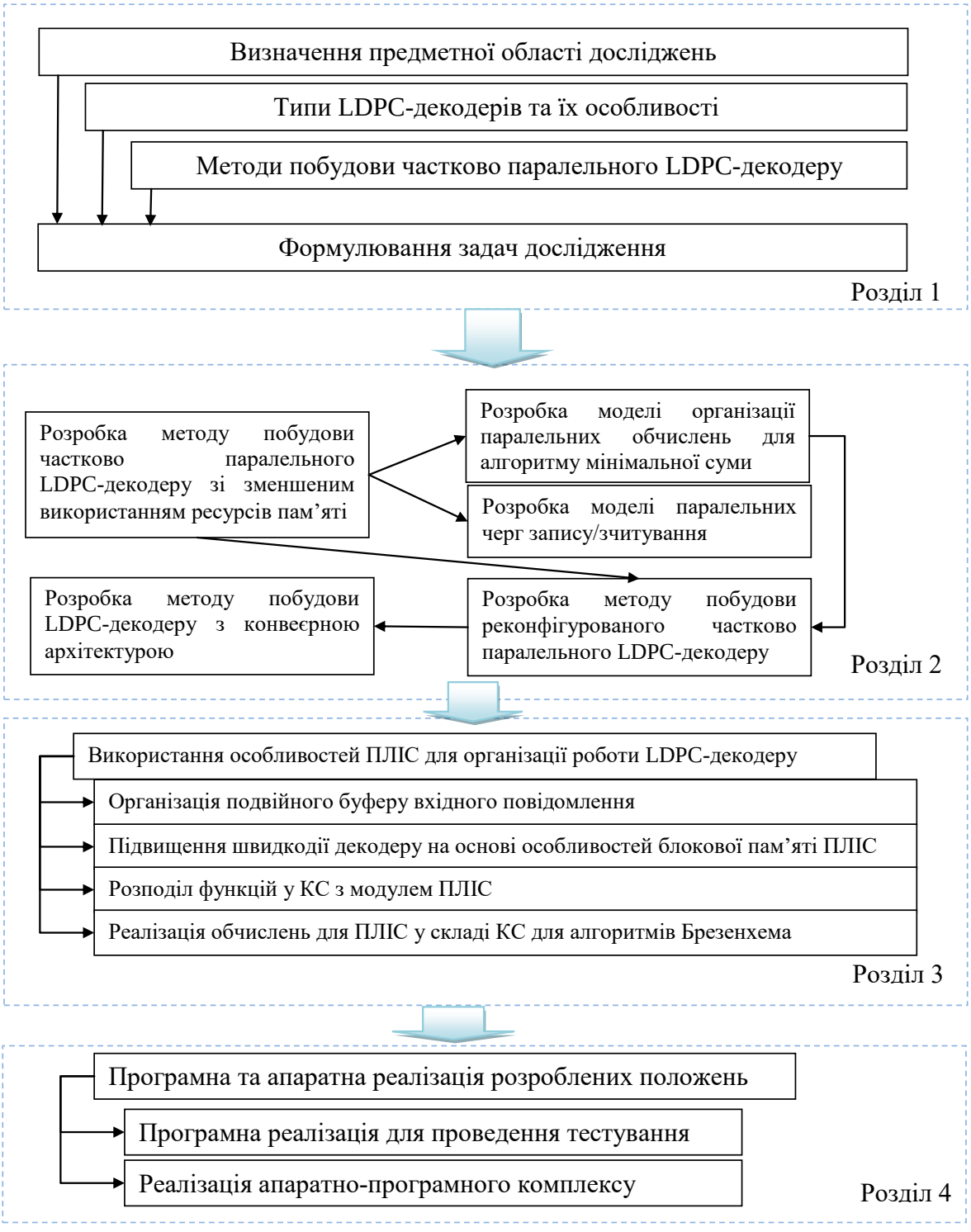


Рис. 1.17. Структурно-логічна схема дисертаційного дослідження

Висновки до розділу І

1. У результаті проведеного аналізу визначено, що існує науково-прикладна задача підвищення ефективності організації архітектур

частково паралельних LDPC-декодерів на базі FPGA, що працюють на основі МПП з довільним розташуванням значущих елементів, зокрема, потребують вирішення задачі зменшення використання ресурсів пам’яті та підвищення пропускної здатності.

2. Проведений аналіз предметної області досліджень, який показав, що LDPC-коди є завадостійкими кодами, що забезпечують найкращі характеристики виправлення помилок. Визначено, що найкращими LDPC-кодами є нерегулярні LDPC-коди на основі МПП з довільним (випадковим) розташуванням значущих елементів.

3. Проведено аналіз типів LDPC-декодерів за апаратною базою, паралельністю виконуваних операцій; визначено, які алгоритми використовуються для декодування LDPC-кодів. На основі аналізу для дослідження обрана апаратна платформа ПЛІС та алгоритм мінімальної суми.

4. Визначені основні проблеми, що характерні для реалізації архітектур LDPC-декодерів на основі довільних МПП: значне використання апаратних ресурсів та нижча пропускна здатність через складність організації паралельної обробки даних.

5. Розглянуто відомі архітектури LDPC-декодерів та визначено, що вони є складними для застосування при реалізації декодеру довільних LDPC-кодів, тому існують проблеми, пов’язані з розробкою архітектур декодерів для таких кодів.

6. Визначено основні задачі дослідження та побудована структурно-логічна схема дисертаційного дослідження.

РОЗДІЛ ІІ
МОДЕЛІ ТА МЕТОДИ ПОБУДОВИ LDPC-ДЕКОДЕРІВ

2.1 Розробка методу побудови частково паралельного LDPC-декодеру зі зменшеним використанням ресурсів пам’яті

При розробці теоретичних положень даного підрозділу використовувались відомі роботи [44, 48, 59-61, 63, 76, 78-84]. Отримані результати представлені у власних розробках [107, 108].

Декодування повідомлення за допомогою алгоритму мінімальної суми відноситься до алгоритмів з обміном повідомленнями між вузлами. Декодеру відома матриця H, розмірністю $M \times N$ (M – кількість рядків, N – кількість стовпців). Кожну ітерацію алгоритму можна розділити на декілька частин:

1) обмін інформацією від вузлів значень до вузлів перевірки (обробка рядка матриці – горизонтальний крок);

2) обмін інформацією від вузлів перевірки до вузлів значень (обробка стовпця матриці – вертикальний крок);

3) формування результуючого значення ітерації;

4) перевірка синдрому.

На етапі виконання горизонтального кроку для ітерації k виконується обробка значень, отриманих від v-вузлів [68]:

$$\alpha_{ij}^{k} = \prod_{j' \in J \setminus j} sign(\beta_{ij'}^{k-1}) \times \min_{j' \in J \setminus j} (|\beta_{ij'}^{k-1}|), \qquad (2.1)$$

де α_{ij}^{k} – значення повідомлення вузла перевірки для поточної ітерації,

$\beta_{ij'}^{k-1}$ – значення повідомлення v-вузла на попередній ітерації.

Для кожного вузла значення визначається мінімальне значення та його знак, проте при розрахунках не використовуються дані від поточного вузла.

На другому етапі проводиться сумування повідомлень вузлів перевірки, з’єднаних з вузлом значення [68]:

$$\beta_{ij}^{k} = \sum_{i' \in I \setminus i} \alpha_{i'j}^{k} + \lambda_{j}, \tag{2.2}$$

де λ_j – початкове значення повідомлення v-вузла, яка відповідає початковим даним для декодування.

Формування результуючого значення ітерації відбувається за рахунок сумування всіх повідомлень вузлів перевірки, під'єднаних до вузла значення [68]:

$$Z_{j}^{k} = \sum_{i \in I} \alpha_{ij}^{k} + \lambda_{j} \tag{2.3}$$

де Z_j^k – результуюче значення для ітерації k.

Відповідно до (2.2) та (2.3) другий та третій етапи ітерації можна проводити паралельно, оскільки вони відрізняються лише одним доданком при проведенні сумування.

Отримавши результуюче значення ітерації, виконується пошук жорстокого рішення за наступним співвідношенням [10-12]:

$$V_{j}^{k} = \begin{cases} 0, \ Z_{j}^{k} > 0 \\ 1, \ Z_{j}^{k} \le 0 \end{cases}. \tag{2.4}$$

На основі отриманого жорсткого рішення обчислюється синдром [10-12]:

$$s = H \cdot V^{T} \quad . \tag{2.5}$$

У випадку, якщо [10-12]

$$s = 0 \ \ mod \ \ 2 \quad , \tag{2.6}$$

то кодове слово проходить перевірку на парність і декодування можна завершувати, видавши в якості результату значення вектору V. Декодування проходить поки не виконані всі ітерації або не виконана умова (2.6).

Перевагою даного методу при апаратній реалізації є те, що використовуються прості операції знаходження мінімуму, суми, виділення знака, що мають просте відображення в реалізації у вигляді блоків компараторів, суматорів та логічних елементів виключного АБО (XOR) відповідно.

Тим не менш, реалізація алгоритму мінімальної суми потребує також значних ресурсів блокової пам'яті для ПЛІС, які є обмеженими. Тому зменшення використання пам'яті за рахунок проведення додаткових простих обчислень дозволить спростити архітектуру декодеру. Можливим рішенням даної проблеми є обчислення значень повідомлень від вузлів значень безпосередньо перед їх обробкою на основі поточного результуючого значення, а також значення повідомлення вузла перевірки. Виразимо на основі (2.2) та (2.3) значення повідомлення v-вузла:

$$\beta_{ij}^{k} = Z_{j}^{k} - \alpha_{ij}^{k}. \tag{2.7}$$

Вказана операція легко може бути реалізована апаратно за допомогою блоку віднімання. При цьому, оскільки таких блоків може бути декілька, то всі необхідні операції для (2.7) можуть виконуватися паралельно.

Проте, оскільки відповідно до (2.7) пам'ять значень Z_j^k повинна залишатися незмінною до завершення поточної ітерації, то необхідно виділити додаткову пам'ять для збереження результатів проміжних обчислень ітерації. Розмір даної пам'яті відповідає розміру пам'яті значень Z_j^k. Позначимо значення, що знаходяться в даній пам'яті як $Ztemp_j^k$.

Після проведення обчислення повідомлення вузла перевірки для одного рядка є можливість виконати часткове обчислення результуючого значення ітерації

$$Ztemp_j^k = Ztemp_j^k + \alpha_{ij}^k \quad . \tag{2.8}$$

Після виконання обробки всіх рядків у пам'яті значень $Ztemp_j^k$ міститиметься значення результуючого значення ітерації. Оскільки в такому випадку відпадає необхідність проведення операцій вертикального кроку, то кількість операцій для однієї ітерації декодування зменшується до кількості рядків у матриці H – M .

Розглянемо метод реалізації декодеру на основі вказаного підходу. Визначається максимальна кількість одиниць у рядку (вага рядка) $w_{r\,max}$ для заданої матриці H :

$$w_{r\,max} = max(w_{ri}),\ r \in R \quad , \tag{2.10}$$

де R – множина рядків матриці H ; w_{ri} – вага i -го рядка.

Значення $w_{r\,max}$ характеризує те, на скільки блоків необхідно розділити пам'ять для збереження індексів та пам'ять значень α_{ij}^k . Розмірність кожного блоку визначається кількістю рядків M для заданої матриці H . Розрядність значень залежить від формату представлення даних для декодера.

Збереження вихідних даних для декодування, а також проміжних результатів обчислень та кінцевих результатів для кожної ітерації відбувається в двох блоках пам'яті, що мають розмірність $1 \times N$. Розрядність пам'яті так само визначається форматом представлення даних.

Таким чином, архітектура декодера передбачає наявність 4 видів пам'яті:

1) пам'ять індексів одиниць у рядках матриці H , $Memory_{address}$;

2) пам’ять для значень α_{ij}^{k}, $Memory_{\alpha}$ (перед початком декодування ініціалізується значенням 0);

3) пам’ять збереження проміжних результатів ітерації, $Memory_{Ztemp}$ (ініціалізується значеннями вхідного набору даних для декодування);

4) пам’ять початкових даних та кінцевих результатів ітерації, $Memory_{Z}$.

Пам’ять $Memory_{address}$ ініціалізується індексами одиниць у рядках матриці H. У випадку нерегулярної матриці, такої що

$$w_{ri} \neq w_{r\,max}, \tag{2.11}$$

значення в рядках заповнюються адресою, яка відсутня у початковій матриці, наприклад, максимальним значенням для даної розрядності. У ході виконання кожної ітерації дані з такими адресами обробляються спеціальним чином, щоб вони не мали вплив на коректні дані.

На кожній ітерації декодування відбувається порядкове зчитування значень з пам’яті $Memory_{address}$ та пам’яті $Memory_{\alpha}$. За отриманими індексами відбувається зчитування значень з пам’яті $Memory_{Z}$. Відповідно до (2.7), поточні значення α_{ij}^{k} і Z_{j}^{k} подаються на блоки віднімання, у результаті чого отримуємо значення β_{ij}^{k}. Ці дані подаються на вхід блоку компараторів та побітового виключного АБО для реалізації операцій, описаних в (2.1). Отримані α_{ij}^{k} перезаписуються до пам’яті $Memory_{\alpha}$ у поточний рядок. Також для цих даних виконується сумування з відповідними значеннями $Ztemp_{j}$ та збереження результату в пам’яті $Memory_{Ztemp}$.

Така послідовність організації ітерації з частковим виконанням обчислень вертикального кроку на етапі горизонтального кроку дозволяє отримати такий самий результат ітерації, як і при послідовному виконанні горизонтального та вертикальних кроків. В той же час, це дає можливість відмовитись від

виділення окремих блоків пам’яті для збереження значень β_{ij}^{k} та відповідних індексів, а також додаткових логічних ресурсів ПЛІС для організації обробки вертикального кроку. Додатково необхідно лише виділити пам’ять для $Memory_{Ztemp}$. Для матриці з максимальною вагою стовпця

$$w_{c\,max} = max(w_{cj}),\ c \in C \quad , \tag{2.12}$$

де C – множина стовпців матриці H; w_{cj} – вага j-го стовпця, виграш відносно необхідності збереження кількості значень в пам’яті становить

$$Count = 2 \cdot w_{c\,max} \cdot N - N \quad . \tag{2.13}$$

На завершальному етапі ітерації виконується копіювання значень з пам’яті $Memory_{Ztemp}$ в пам’ять $Memory_{Z}$. Отримані значення використовуються для проведення перевірки синдрому, на основі чого визначається коректність результатів декодування.

2.2 Розробка моделі організації паралельних обчислень для алгоритму мінімальної суми

Представлені в даному підрозділі розроблені положення [109, 110] отримані на основі відомих розробок [44, 68, 73-75, 83, 86-90].

Відповідно до алгоритму мінімальної суми послідовність дій, що виконується, є наступною (рис. 2.1).

Рис. 2.1. Загальна блок-схема для алгоритму мінімальної суми

Представлені стадії обчислень алгоритму мінімальної суми можуть бути реалізовані паралельно. Наприклад, обчислення синдрому та жорстке декодування, оскільки потребують однакових даних для виконання, тому їх реалізацію можна об'єднати.

Розглянемо детальніше процес виконання декодування для абстрактного декодера, що має паралельний доступ до пам'яті індексів значущих елементів та послідовний доступ до пам'яті повідомлення. Такий декодер може виконувати зчитування значень адрес елементів для декодування на поточному циклі ітерації як одну операцію. Проте, оскільки пам'ять повідомлення організована послідовно, то зчитування/запис елементів для даної пам'яті відбувається також послідовно. Це значно зменшує швидкодію декодеру. Крім того, зчитування/запис елементів на одному циклі для обчислення, перевірки

синдрому та жорсткого декодування відбувається з використанням одних і тих самих адрес, що означає додаткове зменшення швидкодії за умови послідовної реалізації даних процесів.

Для підвищення швидкодії LDPC-декодеру розроблена модель організації роботи декодеру, яка передбачає паралельне виконання етапів обчислення повідомлень, обчислення синдрому та жорсткого декодування, а також обчислення значення повідомлень вузлів перевірки, реалізуючи таким чином принцип Single Instruction-Multiple Data (SIMD). На рис. 2.2 зображене представлення розробленої моделі.

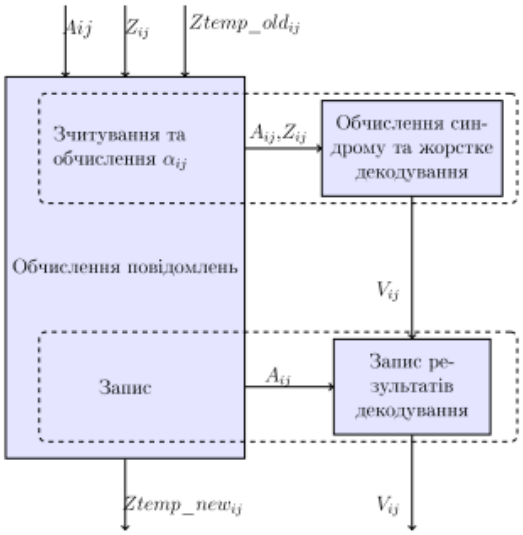


Рис. 2.2. Представлення моделі паралельного виконання операцій для LDPC-декодеру

Відповідно до рис. 2.2, до блоку виконання обчислень надходять дані про індекси значущих елементів A_{ij} (i – номер рядка, j – номер стовпця), за якими послідовно зчитуються значення результуючих Z_{ij} та проміжних $Ztemp\_old_{ij}$ обчислень. Паралельно зі зчитуванням значень Z_{ij} відбувається збір даних для перевірки синдрому для поточного рядка матриці перевірки парності, а також виконується жорстке декодування на основі знаку. Також на основі отриманих значень α_{ij} для попередньої ітерації та Z_{ij} проводиться обчислення значень вузлів перевірки для поточної ітерації. Після цього проводиться обчислення

нових проміжних значень ітерації $Ztemp\_new_{ij}$. Далі слідує етап запису нових значень, паралельно з яким виконується також запис результатів декодування. При цьому, слід зазначити, що операції виконання перевірки синдрому та жорсткого декодування проводяться для даних попередньої ітерації обчислень, а не для поточної, оскільки остаточних даних про результати поточної ітерації немає. Таким чином, наприклад, на початку роботи декодеру проводиться перевірка синдрому та жорстке декодування для початкового вхідного повідомлення, паралельно з чим виконується обчислення нового повідомлення як результату першої ітерації. При виконанні другої ітерації обчислень проводиться перевірка синдрому та декодування результатів першої ітерації і т.д. Результати обчислень повідомлень для останньої ітерації не використовуються. При цьому для отримання результуючого повідомлення слід проводити додаткову ітерацію в порівнянні з послідовним підходом. Результатом додаткової ітерації є лише обчислення синдрому та жорстке декодування. Наприклад, якщо кількість ітерацій при послідовному підході становила 7, то для запропонованої моделі вона збільшується до 8.

Розглянемо детальніше, як відбувається обчислення значень перевірок для поточної ітерації. Відповідно до алгоритму мінімальної суми, значення перевірок для поточної ітерації визначаються наступним чином [68]:

$$\alpha_{ij} = \prod_{j' \in V_i \setminus j} sign(\beta_{ij'}) \cdot min(|\beta_{ij'}|), \qquad (2.14)$$

де V_i – множина вузлів значень для поточного вузла перевірки. У ході виконання зчитування до блоку обробки надходить інформація про значення α_{ij} і Z_{ij}, на основі яких обчислюються значення змінних β_{ij}. Подальшим етапом у обчисленні повідомлень є обчислення значень перевірок відповідно до (2.14). Оскільки дані надходять послідовно і β_{ij} обчислюється послідовно, то обчислення α_{ij} можна також організувати послідовно. Для цього необхідно провести ініціалізацію початкового мінімального значення найбільшим

можливим значенням, а початкового значення знаку як '+'. При надходженні наступного значення β_{ij} проводити порівняння з поточним мінімальним значенням та виконувати множення значень знаків. Таким чином, після подачі всіх значень отримаємо кінцеве значення α_{ij}.

Виконання обчислень відповідно до розробленої моделі потребує зчитування значень $A_{ij_{n\,max}}$:

$$\begin{aligned} &i = 1...rows,\ j = 1...cols; \\ &n_{max} = \max_{i}(\operatorname{count}_{j}(1)),\ j_{n\,max} = 1...n_{max}; \\ &A_{ij_{n\,max}} = read(i). \end{aligned} \tag{2.15}$$

Далі виконується зчитування необхідних значень з інших блоків пам'яті за отриманими адресами та обробка даних:

$$\begin{aligned} &Z_j = read(A_{ij_{n\,max}}),\quad Z_{temp\_oldj} = read(A_{ij_{n\,max}}); \\ &\alpha_{ij}^{k-1} = read(A_{ij_{n\,max}}); \\ &V_j = sign(Z_j),\ error = error \oplus V_j; \\ &\beta_{ij} = Z_j - \alpha_{ij}^{k-1}; \\ &\alpha_{ij}^{k} = \prod_{j' \in V_i \setminus j} sign(\beta_{ij'}) \cdot min(|\beta_{ij'}|). \end{aligned} \tag{2.16}$$

Наступним етапом є перезапис тимчасових значень за ітераціями:

$$Z_{temp\_newj} = Z_{temp\_oldj} + \alpha_{ij}^{k}. \tag{2.17}$$

Останнім етапом, що може виконуватися паралельно є запис результаті до пам’яті:

$$\begin{cases} write(Z_{temp\_newj,}\ A_{ij_{n\,max}}); \\ write(V_{j,}\ A_{ij_{n\,max}}). \end{cases} \tag{2.18}$$

У (2.15) паралельне виконання операцій за наявності необхідних даних показано за допомогою фігурних дужок.

Проведемо порівняння пропускної здатності LDPC-декодеру для послідовної та паралельної моделей. У випадку послідовної моделі за обчисленням повідомлення слідує перевірка синдрому та жорстке декодування. У представленій паралельній моделі ці операції здійснюються одночасно. Для оцінки пропускної здатності декодера скористаємось наступним виразом [98]:

$$T = \frac{F \cdot L}{I \cdot N}, \tag{2.19}$$

де T – пропускна здатність декодеру, Мбіт/с; F – тактова частота роботи декодеру, МГц; L – довжина повідомлення, біт; I – кількість циклів декодування; N – кількість ітерацій декодування.

Введемо позначення для кількості операцій на проведення обчислення повідомлень для одного циклу ітерації m, кількості операцій на перевірку синдрому для одного циклу ітерації – s. Тоді з урахуванням того, що паралельна модель потребує проведення однієї додаткової ітерації, обчислення швидкодії для послідовної та паралельної моделей можна проводити наступним чином:

$$T_{sequential} = \frac{F \cdot L}{(m+s) \cdot I \cdot N}, \tag{2.20}$$

$$T_{parallel} = \frac{F \cdot L}{m \cdot I \cdot (N+1)}. \tag{2.21}$$

На основі (2.20) та (2.21) визначимо відносне збільшення пропускної здатності для паралельної моделі:

$$E = \frac{T_{parallel}}{T_{sequential}} = \frac{m+s}{m} \cdot \frac{N}{N+1}. \tag{2.22}$$

Визначимо значення границі для (2.22) за умови, що $N \to \infty$:

$$\lim_{N \to \infty} E = \frac{m+s}{m} . \quad (2.23)$$

Однак кількість ітерацій для практичного застосування часто обмежують певним значенням, тому множник, що містить значення кількості ітерацій, у такому випадку матиме значний вплив на показник E.

2.3 Розробка моделі паралельних черг запису/зчитування

Отримані у даному розділі теоретичні положення [111, 112] розроблені на основі відомих робіт [74, 77, 81, 82, 90, 94, 97-104].

Основними характеристиками матриці перевірки парності для нерегулярних LDPC-кодів є: M – кількість рядків матриці, N – кількість стовпців матриці, $w_{r\,max}$ – максимальна кількість значущих елементів у рядку.

Причому, хоча б для одного рядка з індексом i виконується умова

$$w_{ri} \neq w_{r\,max} . \quad (2.24)$$

Для паралельного обчислення значень повідомлень перевірки стовпці матриці розділяють на інтервали, які характеризуються мінімальним та максимальним індексами, що входять до інтервалу:

$$I_k = [\, j_{k\,min};\, j_{k\,max} \,];\, 0 \le j_{k\,min} < N;\, j_{k\,min} < j_{k\,max} . \quad (2.25)$$

Кількість інтервалів k обирається відповідно до характеристик матриці та на основі подальших досліджень відносно розташування значущих елементів. При цьому значення мінімального індексу наступного інтервалу має бути на одиницю більше за максимальний індекс попереднього:

$$j_{k+1\,min} = j_{k\,max} + 1. \tag{2.26}$$

Кожному значущому елементу в рядку можна співставити його порядковий номер:

$$n_{iv} = v; v = 1...w_{ri}. \tag{2.27}$$

Добре піддаються розпаралелюванню LDPC-коди, для матриці яких виконується умова, що одному індексу значущого елементу j завжди відповідає одне значення порядкового номеру n_{ij} для всіх рядків:

$$n_{ij} = const; i = 1...M. \tag{2.28}$$

У такому випадку розбиття на інтервали I_k значно спрощується.

В той же час, коли індексу значущого елементу j може відповідати декілька різних значень порядкових номерів, організація паралельних обчислень стає проблематичною, оскільки можуть виникати колізії доступу до елементів, які можна вирішити лише послідовною організацією операцій. Виникнення колізій пов'язано з тим, що у такому випадку неможливо знайти таке розбиття на інтервали, при якому вдається уникнути перетину індексів інтервалів, що розташовуються поряд та не вдається зберегти виконання рівності (2.26). Це означає, що не має можливості відобразити розподіл на інтервали в апаратну реалізацію без уникнення колізій доступу елементів, наприклад, у пам'яті.

Матриці перевірки парності, що мають такі характеристики, найчастіше генеруються випадковим чином без накладання обмежень на характер розташування значущих елементів. Це означає, що для частини рядків можлива паралельна реалізація, а для іншої частини це призводить до колізій. Таким чином, подібні матриці мають неоднорідну структуру з точки зору паралельного виконання обчислень.

З точки зору паралельної організації обчислень для алгоритму декодування мінімальної суми, така неоднорідність означає необхідність очікування даних перед обробкою. Наприклад, для стадії обчислення повідомлень перевірки алгоритму необхідна наявність всіх даних про елементи в рядку для обчислення мінімуму та знаку [68]:

$$\alpha_{ij} = \prod_{j' \in V \setminus j} sign(\beta_{ij'}) \cdot min(|\beta_{ij'}|). \tag{2.29}$$

Послідовна організація запису/зчитування означає, що значно збільшується час отримання елементів у порівнянні з можливою паралельною реалізацією, відповідно до їх кількості в рядку w_{ri}.

Для реалізації обробки нерегулярних LDPC-кодів для алгоритму мінімальної суми, матриця перевірки парності характеризується подібною неоднорідністю пропонується модель організації паралельних черг запису/зчитування. Вона передбачає розбиття на $w_{r\,max}$ інтервалів I_k, яке в найпростішому випадку виконується наступним чином:

$$\begin{cases} step = N \,/\, w_{r\,max}; \\ j_{k\,min} = round(\,step * (k-1) + 1); \\ j_{k\,max} = round(\,j_{k\,min} + step\,); \\ k = 1 \ldots r_{r\,max}, \end{cases} \tag{2.30}$$

де $step$ – крок збільшення індексів інтервалу;

$round$ – функція округлення до найближчого цілого.

Вхідними даними виступають адреси, за якими слід виконати необхідні операції, дані для запису, за необхідності та вхід вибору операції. Виходом моделі є вихід, що сигналізує про завершення виконання операції, а також зчитані дані при виконанні операції зчитування.

За обслуговування елементів інтервалу відповідає співставлений йому модуль виконання запису/зчитування. Кожен такий модуль має доступ до окремого елементу пам'яті, в якому міститься інформація про результати

обчислень. Приналежність елементу до певного блоку обслуговування визначається на основі входження індексу елементу до інтервалу I_k. Для представлення приналежності вхідних даних для блоку використовується шаблон приналежності – бітовий рядок, в якому приналежність даних до інтервалу визначається як «1», а відсутність приналежності – як «0»:

$$pattern_{ij}(u) = \begin{cases} 1,\ j \in I_k; \\ 0,\ j \notin I_{k.} \end{cases} \tag{2.31}$$

Довжина кожного бітового рядка становить $w_{r\,max}$. Такий бітовий рядок формується для кожного значущого елемента рядку, а у випадку, якщо кількість значущих елементів у рядку менша за $w_{r\,max}$, то бракуюча кількість елементів заповнюється лише нулями. Шаблони приналежності зберігаються в пам'яті; кількість блоків пам'яті для збереження шаблонів – $w_{r\,max}$.

Одночасно з подачею вхідних даних на блок подається шаблон відповідності, на основі якого він визначає дані призначені для його обробки. На основі отриманого шаблону формується черга обробки. Всі елементи черги мають отримати обслуговування, а відмови відсутні. Час роботи моделі – дискретний. Таким чином, модель паралельних черг представляє собою систему масового обслуговування (СМО) без відмов з дискретним часом.

Результатом операції запису є подача службового сигналу про завершення виконання операції. В той же час, у результаті виконання зчитування необхідно подати отримані дані для обробки в основному циклі. Для цього слугує мережа комутації вихідних сигналів блоків обробки. Мережа має $w_{r\,max}^{\ 2}$ входів та $w_{r\,max}$ виходів даних. Також на вхід мережі подаються значення сигналів шаблонів, які для кожної ітерації i групуються наступним чином:

$$pattern_{netwj}(d) = pattern_{id}(j);\ d = 1...w_{r\,max.} \tag{2.32}$$

Таким чином, у згрупованому сигналі $pattern_{netwj}$ зустрічається максимум одна одиниця, порядковий номер якої в бітовому рядку визначає, як саме виконати з'єднання між входами та виходами. Номер одиниці в рядку позначає, який сигнал з блоків обробки необхідно зчитати для подачі на конкретний вихід. У випадку роботи лише з однією матрицею, мережа комутації може бути спрощена шляхом дослідження інтервалів та відкиданням зв'язків, що не використовуються.

Обробка черги для кожного блоку відбувається послідовно. За один елементарний дискретний інтервал часу можливе обслуговування лише одного блоку даних. Обробка даних всією системою вважається завершеною лише тоді, коли завершена обробка для всіх модулів. Загальний час обробки окремого блоку залежить від кількості елементів, що необхідно обробити. Тому степінь паралельності може змінюватися від повністю паралельної до послідовної, в залежності від розташування значущих елементів. Тому вибір на користь використання моделі паралельних черг слід робити з урахуванням того, чи дозволить це підвищити пропускну здатність декодеру. Час обробки для моделі паралельних черг для одного циклу ітерації визначиться як

$$T_{queuei} = T_{queue\_aux} + T_{queue\_servicei}, \tag{2.33}$$

де T_{queuei} – загальний час обробки;

T_{queue_aux} – час на виконання допоміжних дій (подача сигналів тощо), постійне значення для кожного циклу;

$T_{queue\_servicei}$ – час безпосереднього обслуговування, який визначається як максимальна кількість елементів, призначена для обслуговування одним блоком:

$$T_{queue\_servicei} = max_j(count\_ones(pattern_{ij})), \tag{2.34}$$

де $count\_ones$ – функція, що повертає кількість одиниць у бітовому рядку. Середній час для повної ітерації декодування на основі моделі паралельних черг визначить як

$$T_{queue\_servicei\_mean} = T_{queue\_aux} + M(T_{queue\_servicei}), \tag{2.35}$$

де $M(T_{queue\_servicei})$ – математичне сподівання кількості елементів на одному циклі. У тому випадку, якщо час обробки для моделі паралельних черг виявиться меншим за час послідовної обробки

$$T_{queue\_servicei\_mean} < T_{sequential}, \tag{2.36}$$

то доцільним є використання запропонованої моделі.

Перевагою даної моделі з точки зору проведення реконфігурації декодеру є те, що зміна матриці перевірки парності може проводитися без внесення додаткових модифікацій до структури блоку, завдяки тому, що принцип обробки елементів перенесений на обробку шаблонів приналежності. Це означає, що можливості паралельного виконання операцій використовуються відповідно до розташування елементів. Основною умовою до матриць є те, щоб кількість значущих елементів у рядках матриці не перевищувала $w_{r\max}$.

2.4 Розробка методу побудови реконфігуровного частково паралельного LDPC-декодеру

Представлені у даному розділі розроблені теоретичні положення [113] отримані на основі відомих робіт [44, 48, 68, 81-82, 94].

Введемо позначення для характеристик матриці перевірки парності. Позначимо як M кількість рядків матриці, N – кількість стовпців матриці, а також n_{max} – максимальну кількість значущих елементів у i-му рядку матриці, $i = 1...M$. Алгоритм MinSum є ітеративним алгоритмом на основі обміну

повідомленнями між вузлами перевірки та вузлами значень. У ході обчислень використовуються м'які дані, що представляють логарифмічне відношення подібності (LLR) відносно значення j-го символу, $j = 1...N$. Позначимо як Z_j значення LLR, на основі якого приймається рішення про значення символу. Значення Z_j ініціалізується даними з каналу (a priori LLR), $Zstart_j$, та змінюється в ході ітерації декодування (posterior LLR). У ході ітерації відбувається обчислення повідомлення вузлів перевірки як [68]

$$\alpha_{ij} = \prod_{J'} sign(\beta_{ij \in J'}) \cdot min(\left|\beta_{ij \in J'}\right|) \tag{2.37}$$

де β_{ij} – повідомлення вузла значень (початково ініціалізуються як $Zstart_j$), J' – множина вузлів значень, підключених до i-го вузла перевірки за виключенням j.

Повідомлення вузлів значень обчислюються як [68]

$$\beta_{ij} = \sum_{I'} \alpha_{i \in I' j} + Zstart_j, \tag{2.38}$$

де I' – множина всіх вузлів перевірки підключених до j-го вузла значень за виключенням i.

Значення Z_j за результатами ітерації обчислюється як [68]

$$Z_j = \sum_{I} \alpha_{i \in Ij} + Zstart_j, \tag{2.39}$$

де I – множина всіх вузлів перевірки, підключених до вузла значень j.

Відповідно до (2.38) та (2.39) значення Z_j та β_{ij} відрізняються лише одним доданком. Тому β_{ij} може бути представлено як

$$\beta_{ij} = Z_j - \alpha_{ij} \quad . \tag{2.40}$$

З точки зору реалізації це означає, що значення β_{ij} може бути швидко обчислене, а, значить, є можливість для зменшення використання пам'яті на їх збереження. Такі зміни також означають, що Z_j у ході ітерації декодування має залишатися незмінним. Оскільки β_{ij} є лише проміжним результатом, а остаточне рішення приймається на основі Z_j, то саме це значення є важливим. Введення додаткового проміжного значення $Ztemp_j$ дозволить виконати обчислення результуючого значення ітерації при частковому виконанні вертикального кроку (2.39) за результатами (2.38) як

$$Ztemp_j = Ztemp_j + \alpha_{ij} \quad . \tag{2.41}$$

У такому випадку, побудова блоку декодування, який виконує обробку одного рядка та частково виконує обробку стовпця на основі отриманих даних, полягає у наступному. У пам'яті необхідно зберігати індекси значущих елементів по горизонталі. Дану частину пам'яті можна розділити на n_{max} частин для організації паралельного доступу. Пам'ять індексів ініціалізується як

$$index_l = order(1_{ij \in J}); l = 1...n_{max} \quad , \tag{2.42}$$

де $order$ – функція, що повертає порядковий номер елементу, J – множина всіх вузлів значень, що підключені до i-го вузла перевірки. У тому випадку, якщо кількість елементів у окремому рядку матриці n_i є меншою за максимальну кількість елементів

$$n_i < n_{max} \quad , \tag{2.43}$$

значення індексів, яких не вистачає у даному рядку доповнюється спеціальним значенням, яке визначається заздалегідь. Наприклад, якщо кількість бітів, які необхідні для збереження значення індексів є $index\_num\_bits$, то доцільно це значення представити як

$$2^{index\_num\_bits} - 1 \quad . \tag{2.44}$$

У такому випадку при обробці необхідно проводити перевірку індексів на рівність даному значенню та не проводити обчислення для відповідних елементів.

Пам’ять значень α_{ij} слід розділити на n_{max} частин. Доступ до пам’яті при обробці елементів одного рядка також виконується паралельно. Перед початком декодування значення в пам’яті мають бути встановлені в 0.

Оскільки структура матриці парності заздалегідь невідома, то невідомі і значення індексів, за яким буде відбуватися звернення до пам’яті, що містить значення Z_j та $Ztemp_j$. Відповідно, отримання необхідних даних не може бути організовано паралельно, а, значить, не може бути розділена на декілька частин. Доступ до даних компонентів виконується послідовно, що гарантує уникнення колізій доступу.

Відповідно до представлених положень, спрощена організація роботи блоку декодування на ітерації k без урахування стадії перевірки синдрому та виконання жорсткого декодування представлена на рис. 2.3.

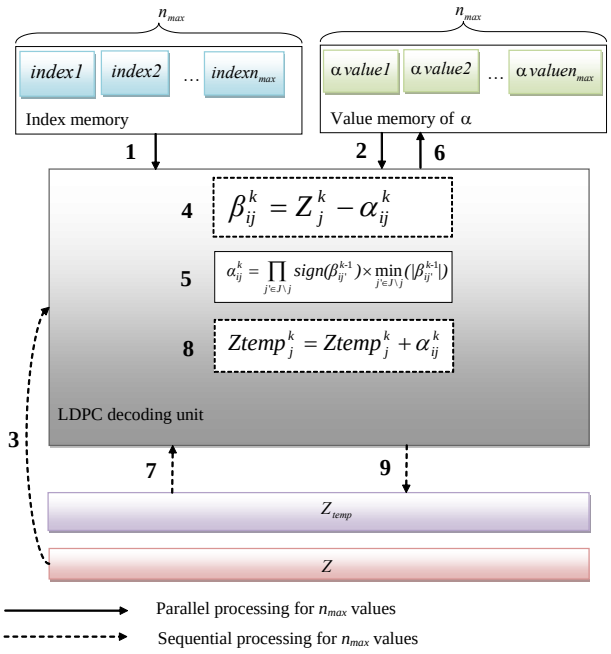


Рис. 2.3. Організація модуля декодування

На рис. 2.3 числа означають послідовність виконання операцій. Також вказано розділення операцій, що виконуються паралельно або послідовно для n_{max} значень.

Основною перевагою LDPC-декодеру з можливістю реконфігурації є можливість його повторного використання для різних матриць перевірки парності. При цьому на структуру матриці накладаються мінімальні обмеження:

– кількість рядків матриці не має перевищувати максимального встановленого для декодера значення;

– максимальна кількість значущих елементів у рядку матриці не має перевищувати обране значення найбільшої кількості елементів у одному рядку, з якою може працювати декодер n_{max_dec}.

Для того, щоб забезпечити обробку матриць з різною структурою та не втратити у пропускній здатності, необхідно забезпечити, щоб запис/обробка/зчитування непотрібних елементів не проводились.

Роботу блоку декодування при виконанні послідовних операцій можна представити як скінчений автомат (FSM). У найпростішому випадку блок декодування проходить усю послідовність станів, наприклад, для зчитування $S_{read} = \{ s_{1read}, s_{2read}, ..., s_{n\,max\_read}, s_{finish} \}$. При обробці матриці перевірки парності, що має значення $n_{max} < n_{max\_dec}$ обробка всіх значень не є обов'язковою та може спричинити програш у пропускній здатності. Для забезпечення максимальної гнучкості налаштувань декодеру використовуємо вхідний сигнал $var\_processing$, який є бітовим рядком, а елементи формуються відповідно до

$$var\_processing(u) = \begin{cases} 1, u \le n_{max}; \\ 0, n_{max} < u \le n_{max\_dec;} \end{cases}$$
$$u = 1...n_{max\_dec.} \tag{2.45}$$

У такому випадку, для етапу зчитування даних правила переходу декодеру з поточного стану s_{uread} у наступний стан можуть бути описані наступним чином:

$$s_{next} = \begin{cases} s_{u+1read}, & var\_\ proces\ sin\ g(u) = 1; \\ s_{finish}, & var\_\ proces\ sin\ g(u) = 0. \end{cases} \tag{2.46}$$

Таким чином забезпечується обробка лише необхідної кількості значень.

За відповідним принципом організована робота декодеру при виконанні послідовних операцій. Дану особливість слід враховувати при реалізації інших стадій обробки. За рахунок цього можливе спрощення реалізації декодеру шляхом об'єднання окремих стадій, а також підвищення швидкодії, незважаючи на те, що операції, що були паралельними стають послідовними. Етап обчислення α_{ij} на рис. 2.3 позначений як паралельний може проводитись послідовно відповідно до надходження необхідних даних β_{ij}. Для кожного

елементу при обробці використовується окремий блок обчислення. Загальна структура організації обчислення α_{ij} представлена на рис. 2.4.

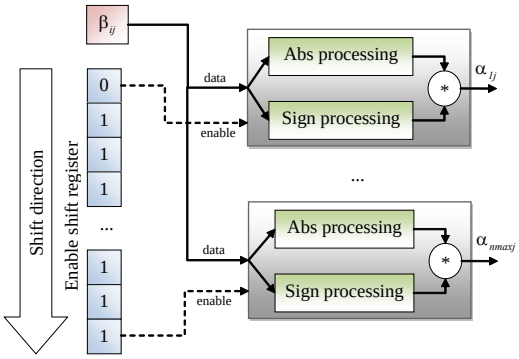


Рис. 2.4. Спрощена структура послідовного блоку обчислень повідомлень вузлів перевірки

Для позначення необхідності обробки поточного значення використовується регістр зсуву. Представлений блок забезпечує сталу затримку на обчислення незалежно від значення n_{max}.

Така реалізація дозволяє, подавши необхідні сигнали на входи декодеру, забезпечити обробку матриці, на структуру якої накладаються мінімальні обмеження. Таким чином, може відбуватися зміна матриці перевірки парності шляхом завантаження нових значень індексів та зміни додаткових сигналів (кількість рядків, довжина вхідного повідомлення та ін.) без необхідності перепрограмування ПЛІС.

Описана вище архітектура блоку декодування передбачає використання лише одного порту пам’яті. За рахунок підключення кількох таких блоків до пам’яті є можливість підвищити швидкість обробки і, відповідно, пропускну здатність LDPC-декодеру. При цьому слід враховувати рівень технологічного розвитку сучасних мікросхем ПЛІС, який дозволяє використовувати максимум 2 порти запису/зчитування. За рахунок роботи пам’яті на тактовій частоті вдвічі більшій, за основну частоту, є можливою робота чотирьох портів запису/зчитування при наявності реальних двох [114].

Просте підключення додаткових блоків декодування до нових портів запису/зчитування з незначними змінами відносно сигналів налаштувань не гарантує коректної обробки через:

− обмеження блокової пам’яті – не можна одночасно проводити запис за однією адресою для різних портів;

− обмеження колізій, що можливі при декодуванні, – при паралельній обробці кількох рядків необхідно, щоб усі значення індексів були різними.

За наявності чотирьох блоків декодування необхідно, щоб у будь-якому наборі з чотирьох послідовних рядків не зустрічались однакові індекси:

$$\begin{aligned} &\forall index_{i1j}, \forall index_{i2j}, \neg\exists\, index_{i1j} = index_{i2j}, \\ &j = 1...n_{max}, i1, i2 = 1...4, i1 \neq i2. \end{aligned} \tag{2.47}$$

Виконання (2.47) можливо забезпечити з використанням операції перестановки рядків матриці. Для реалізації доцільно створити результуючу матрицю, яка є початково порожньою, та додати в неї всі рядки вихідної матриці з дотриманням умови (2.47), яку проводити лише для рядка, що додається.

Паралельна робота кількох блоків декодування потребує синхронізації для коректного представлення результатів. Синхронізація необхідна для визначення продовження роботи усіх блоків декодування та відповідності усіх стадій обробки, що проводяться блоками.

Умовою завершення роботи декодеру є

$$\begin{aligned} &finish = (iter = iter\_max) \vee \\ &(e_1 = 0 \wedge e_2 = 0 \wedge e_3 = 0 \wedge e_4 = 0), \end{aligned} \tag{2.48}$$

де $iter$ – номер поточної ітерації, $iter\_max$ – максимальна кількість ітерацій, $e_1...e_4$ – кількість помилок, виявлених при перевірці синдрому відповідними блоками декодування.

Для підвищення пропускної здатності також доцільно обмежити виконання стадії перевірки синдрому до знаходження першої помилки. З точки зору синхронізації, це потребуватиме переходу до наступної ітерації всіх блоків декодування. Для цього проводиться перевірка

$$exit\_synd = cur\_error \vee error_1 \vee error_2 \vee error_3 \tag{2.49}$$

де cur_error – значення, що вказує на виявлення помилки у поточному блоці, $error_1 ... error_3$ – вхідні значення від інших блоків про наявність помилки. Перехід до наступної ітерації декодування відбувається на наступному циклі перевірки синдрому після виявлення помилки.

Таким чином, підключення чотирьох блоків декодування, що працюють паралельно здатне в 4 рази збільшити пропускну здатність декодеру. При цьому, важливою для коректної роботи декодеру є синхронізація всіх блоків.

2.5 Конвеєрна архітектура LDPC-декодування на базі ПЛІС з використанням модифікованого алгоритму мінімальної суми

Представлені у даному розділі розроблені теоретичні положення [115, 116] отримані на основі відомих робіт [44, 68, 74, 76, 81-84, 89, 94-98].

Реалізація декодеру за методом, що пропонується, передбачає виконання трьох наступних етапів:

1) попередня обробка МПП;
2) організація конвеєрної архітектури декодеру;
3) реалізація модифікованого алгоритму мінімальної суми.

Введемо наступні позначення для важливих характеристик МПП, відповідному їй графі Таннера та для опису обчислень алгоритму мінімальної суми у процесі декодування. МПП містить M рядків та N стовпців. При цьому, максимальна кількість значущих елементів у окремому рядку – w_{rmax}. Граф Таннера містить вузли перевірки $c_i, i = 1...M$ та вузли значень $v_j, j = 1...N$.

Алгоритм декодування мінімальної суми є ітеративним алгоритмом на основі обміну повідомленнями між вузлами графа Таннера. Вузли перевірок обчислюють значення повідомлення α_{ij} для кожного вузла значення на основі отриманого від з'єднаних вузлів значень повідомлень β_{ij} та навпаки. Остаточні значення за результатами ітерації декодування зберігається в пам'яті Z, та визначаються як [68]

$$z_j = z_{jstart} + \sum_{i \in I} \alpha_{ij}, \tag{2.50}$$

де z_{jstart} – початкове значення для j-го вузла; I – множина всіх вузлів перевірки, що підключені до j-го вузла значень. При цьому з урахуванням того, що у ході ітерації остаточне значення z_j визначається не одразу, то використовується також пам'ять проміжних результатів ітерації Z_{temp} з елементами z_{tempj}.

Попередня обробка МПП проходить у декілька кроків. На першому кроці використовується операція перестановки стовпців матриці з метою розділення на інтервали, які містять не більше одного значущого елементу. На даній стадії, визначається ширина кожного інтервалу $width_i$ та їх кількість n. У процесі заповнення інтервалів необхідно відслідковувати, щоб стовпець, що додається add_col, не був доданий до інтервалу, що вже містить такий стовпець $check\_col$, який має значущий елемент з однаковим індексом рядка:

$$\begin{aligned} &\forall row\_ind(add\_col, j_1) \ \neg\exists\, row\_ind(add\_col, j_1) = row\_ind(check\_col, j_2), \\ &j_1 = 1...N_1, j_2 = 1...N_2, \end{aligned} \tag{2.51}$$

де row_ind – функція, що повертає індекс рядка; N_1, N_2 – кількість значущих елементів у відповідних стовпцях. Якщо дана умова виконується, то стовпець може бути доданий до інтервалу, інакше, необхідно спробувати додати його до

наступного інтервалу. Блок-схема описаного алгоритму перестановки стовпців матриці наведена на рис. 2.5.

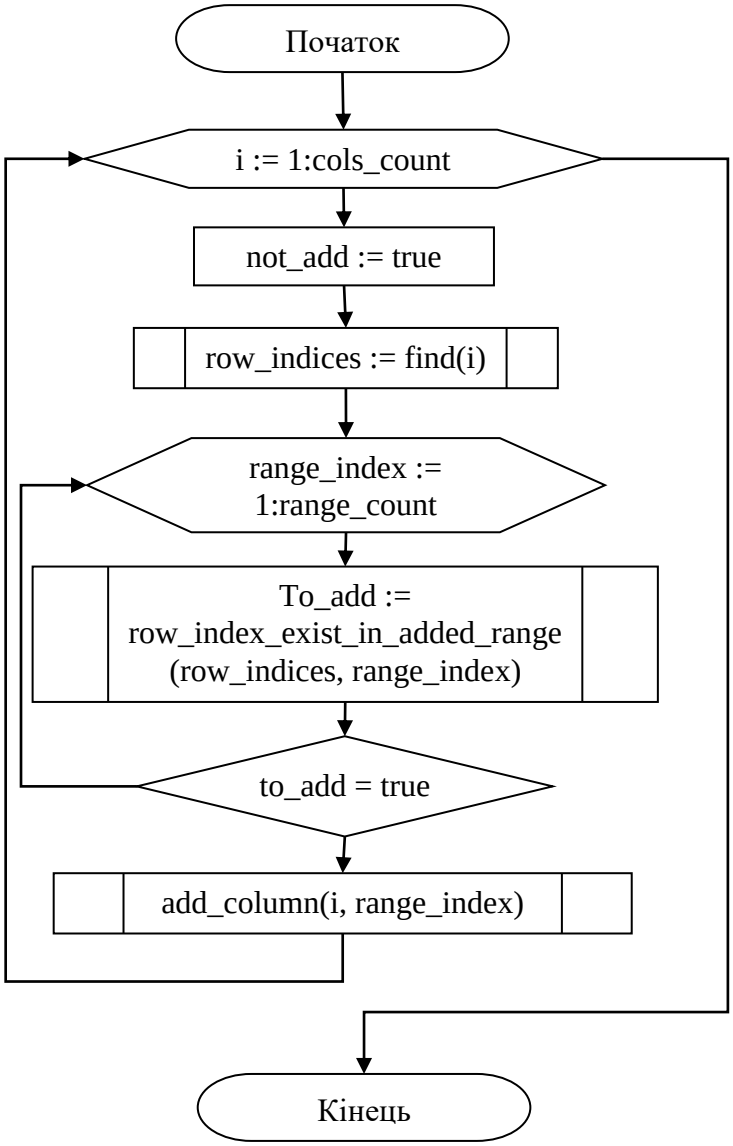


Рис. 2.5. Блок-схема алгоритму перестановки стовпців для попередньої обробки матриці

У тому випадку, коли наявні такі стовпці, що не можна додати до будь-якого інтервалу, слід збільшити кількість інтервалів. У результаті даного кроку отримується масив відповідності для початкової та результуючої матриці.

На другому кроці при обробці матриці використовується інша операція елементарного перетворення матриці – перестановка рядків. На даному етапі

забезпечується виконання умови відсутності однакових індексів у m попередніх та послідовних рядках відносно поточного. Значення m визначається характеристиками конвеєру обробки.

Конвеєрна архітектура декодеру залежить від обраного алгоритму декодування. Організація конвеєру передбачає, що вихід одного обчислювального блоку є входом іншого. Відповідно до цього та з урахуванням обчислювальних операцій, що використовуються в алгоритмі мінімальної суми, пропонується наступна організація конвеєру (рис. 2.6).

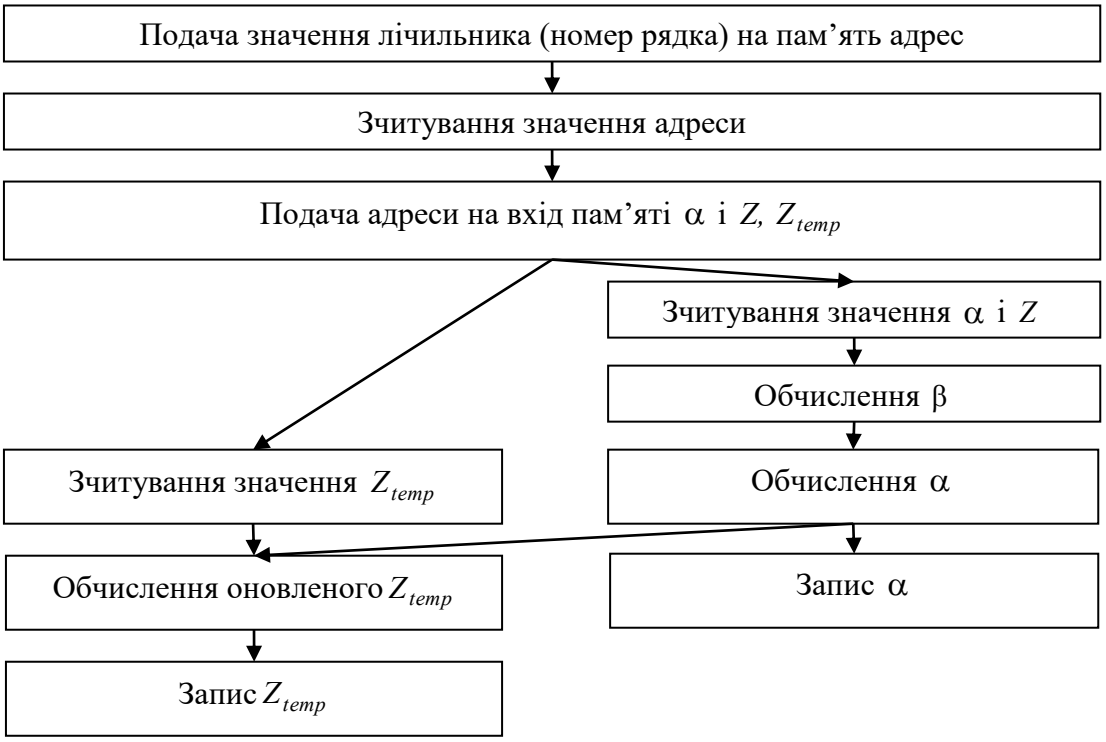


Рис. 2.6. Структурна організація конвеєру обчислень для декодеру

Дана організація конвеєру при повному завантаженні всіх складових на одному циклі забезпечує обробку одного рядка з частковою обробкою стовпців значущих елементів у рядку. Таким чином, загальний час обробки даних на основі МПП з M рядками становитиме

$$time\_total = M + lag_{pipeline}, \qquad (2.52)$$

де $lag_{pipeline}$ – загальна затримка конвеєру, що визначається кількістю стадій обробки, а також залежить від характеристик МПП.

Загальна діаграма зміни частини сигналів при конвеєрній обробці показані на рис. 2.7.

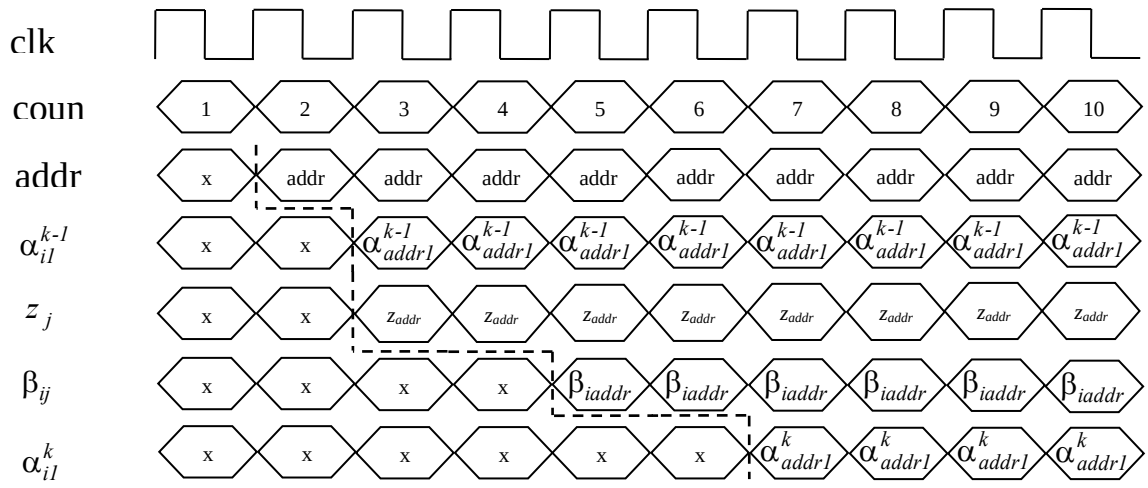


Рис. 2.7. Загальна діаграма роботи конвеєру декодування

Частина значень сигналів на діаграмі позначена неактивною через те, що конвеєр на початку роботи завантажений необхідними даними не на всіх стадіях, а лише на частині. Через це наявна затримка для різних сигналів, яка виділена на рисунку.

Алгоритм мінімальної суми передбачає те, що на початок кожної ітерації у Z та Z_{temp} знаходяться однакові значення. Тим не менш, у тому випадку, коли у вхідному повідомленні наявна відносно невелика кількість помилок, то бажаною характеристикою алгоритму декодування є здатність виправляти якомога більше некоректних значень за одну ітерацію, не вносячи при цьому помилки в коректні позиції. Це дозволяє зменшити загальну кількість ітерацій, а, відповідно, підвищити швидкодію. Такого результату вдається досягти в тому випадку, якщо не проводити запис з пам’яті проміжних значень Z_{temp} у пам’ять Z, а використовувати Z_{temp} в якості Z на наступній ітерації. При

цьому, значення в іншій пам’яті не змінюються. Фактично, це відповідає обміну значеннями між Z та Z_{temp}.

$$\begin{aligned} buf &= Z; \\ Z &= Z_{temp}; \\ Z_{temp} &= buf. \end{aligned} \tag{2.53}$$

Таким чином, значення у Z та Z_{temp} на початку ітерації для модифікованого алгоритму не є однаковими.

У випадку реалізації для ПЛІС це може бути реалізовано за допомогою простого селектору пам’яті, що змінює своє значення при зміні номеру ітерації:

$$sel\_val = num\_iter \bmod 2, \tag{2.54}$$

де num_iter – номер поточної ітерації. Представлення алгоритму з точки зору графу Таннера наведена на рис. 2.8.

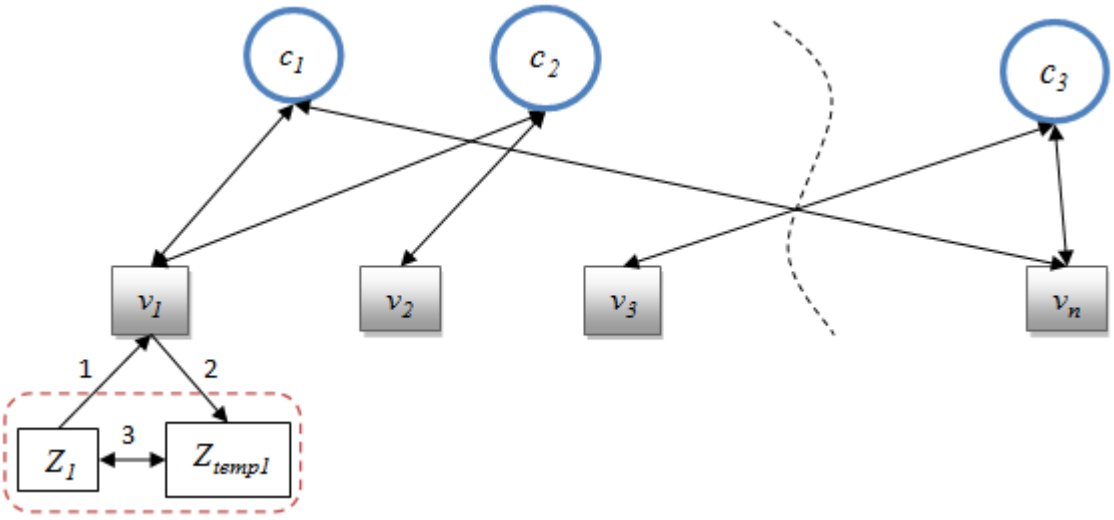


Рис. 2.8. Послідовність виконання операцій з пам’яттю за модифікованим алгоритмом

Висновки до розділу II

1. Запропоновано метод побудови LDPC-декодеру, який дозволяє зменшити використання ресурсів пам’яті ПЛІС. За рахунок зміненого

представлення повідомлень вузлів значень вони можуть обчислюватись безпосередньо у необхідний момент на основі інших даних, а не зберігатися в пам’яті мікросхеми ПЛІС. Додана проміжна стадія обчислень результатів ітерації дозволяє дотримуватись коректності виконання операцій відповідно до алгоритму.

2. Розроблена модель організації паралельних обчислень для алгоритму мінімальної суми, що дозволило підвищити пропускну здатність декодеру. Запропонована модель передбачає паралельне виконання обчислення синдрому та декодування, у результаті чого, стадія обчислення синдрому фактично випереджає стадію декодування на одну ітерацію. Такий підхід дозволяє збільшити кількість проведених ітерацій декодування за менший час.

3. Розроблена модель паралельних черг запису/зчитування, яка дозволяє підвищити швидкодію декодеру на основі підвищення паралельності обробки. Модель передбачає організацію потоку даних у декодері у вигляді СМО, робота яких керується патернами відповідності.

4. Розроблено метод побудови реконфігуровного декодеру, який здатен проводити обробку на основі МПП, на структуру якої накладаються мінімальні обмеження. Це дозволяє змінювати налаштування роботи такого декодеру без необхідності його зупинки та перепрограмування ПЛІС.

5. Розроблена архітектура конвеєрного LDPC-декодеру, застосування якої дозволяє значно підвищити пропускну здатність декодеру. За рахунок того, що усі стадії виконуються паралельно одразу для всього набору вхідних даних, вдається забезпечити високу пропускну зданість.

РОЗДІЛ ІІІ
ВИКОРИСТАННЯ ОСОБЛИВОСТЕЙ ПЛІС ДЛЯ ОРГАНІЗАЦІЇ РОБОТИ LDPC-ДЕКОДЕРУ

3.1 Організація подвійного буферу вхідного повідомлення

Представлені у даному розділі розроблені теоретичні положення [117, 118] отримані на основі відомих робіт [44-48, 68, 76, 84].

Перед початком декодування виконується запис вхідного повідомлення в пам'ять. Після завершення декодування слідує стадія видачі декодованого повідомлення. При цьому при підготовці до наступної ітерації необхідно очікувати запису нового повідомлення. Стадія очікування перед декодуванням кожного повідомлення, за умови, що дані до декодеру можуть надходити постійно, значно зменшує пропускну здатність LDPC-декодеру. Таким чином, саме стадія очікування є вузьким місцем у роботі декодеру.

На основі виразу для обчислення пропускної здатності декодеру [98] визначимо пропускну здатність з урахуванням необхідності очікування запису повідомлення:

$$T = \frac{F \cdot L}{I \cdot N + \frac{L}{W}}, \tag{3.1}$$

де T – пропускна здатність, Мбіт/с; F – тактова частота роботи декодеру, МГц; L – довжина повідомлення, біт; I – кількість циклів за ітерацію; N – кількість ітерацій; W – розмір записуваного слова, біт. Другий доданок у знаменнику характеризує затрати на запис повідомлення.

Для максимального зменшення часових затрат пропонується використання моделі організації пам'яті декодеру з подвійним буфером для збереження та обробки повідомлення. Вона полягає у тому, що виконується дублювання пам'яті запису повідомлення та результатів ітерації ($Memory_Z$)та пам'яті для

збереження проміжних обчислень ($Memory_{Ztemp}$). Таке рішення дозволяє після запису першого повідомлення одразу записати дані для наступного циклу декодування. Таким чином, очікування запису має місце лише для першого повідомлення, а для наступних повідомлень декодування може розпочатися одразу після завершення декодування попереднього повідомлення. В цей же час розпочинається запис у інший буфер для проведення наступного циклу.

При цьому можливі різні варіанти реалізації подвійного буферу. Це може виконуватися за допомогою дублювання існуючих блоків (додаються нові блоки пам'яті) або за рахунок подвоєння розмірів існуючої пам'яті. Крім цього, для першого варіанту також можливі різні реалізації вибірки даних.

Розроблена модель організації подвійного буферу для випадку дублювання пам'яті з вибіркою значень на основі окремих мультиплексорів та демультиплексорів для кожного запиту представлена на рис. 3.1. Керуючий блок (control unit) містить внутрішній сигнал, selector , відповідно до якого відбуваються запити до пам'яті. Даний сигнал слугує сигналом вибору для мулитиплексорів вхідних та демультиплексорів вихідних сигналів. Відповідно до його значення сигнали керуючого блоку декодеру для доступу до пам'яті подаються лише на одну з пар виходів Z та $Ztemp$ та зчитуються для обробки лише з обраної пари входів. Варто зауважити, що такі комбінаційні логічні елементи необхідні при кожній операції зчитування або запису. Тому основним недоліком даної моделі є те, що сигнал вибору необхідно розвести для великої кількості елементів. Це впливає на результуючу пропускну здатність, оскільки зменшується максимальна частота роботи декодеру. Даний недолік проявляється при реалізації декодеру.

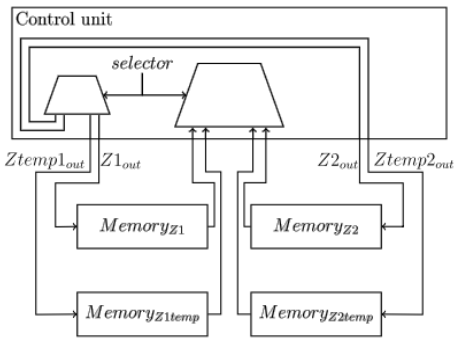


Рис. 3.1. Організація подвійного буферу на основі дублювання пам'яті з окремими елементами вибірки

Іншим варіантом організації подвійного буферу є модель з дублюванням пам'яті (рис. 3.2). У ній сигнали управління вибіркою перенесені з керуючого блоку в блок керування пам'яті, що об'єднує у своєму складі блоки пам'яті, а також містить комбінаційну логіку для управління сигналами пам'яті. На відміну від попереднього випадку, контролюючий блок не містить сигналу вибору пам'яті, який перенесений у блок декодера верхнього рівня. Також використовується лише одна пара мультиплексор/демультиплексор для прийому/видачі повідомлень, що дозволяє зменшити використання логічних ресурсів. Використання пам'яті залишається таким самим, як і в попередньому випадку.

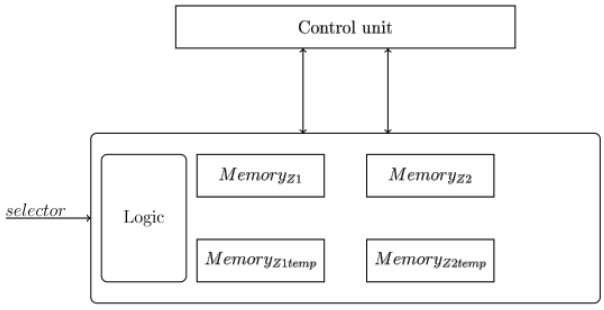


Рис. 3.2. Представлення моделі організації подвійного буферу декодеру на основі окремого блоку

Відповідно до рисунку, блок комбінаційної логіки виконує ті самі функції, що мультиплексор та демультиплексор на рис. 3.1. До нього надходять

управляючі сигнали від контролюючого блоку, а також сигнал вибору, відповідно до яких виконується запис/зчитування з необхідних блоків пам'яті.

Третій варіант організації подвійного буферу передбачає, що обсяг пам'яті існуючих блоків збільшується в 2 рази. Розроблена модель організації подвійного буферу на основі подвоєння об'єму пам'яті представлена на рис. 3. Сигнал вибору в даному випадку відіграє роль старшого розряду адреси. Для реалізації даної моделі необхідно, щоб виконувались умови відносно розмірів вхідного повідомлення, W, та пам’яті $Length_{memory}$. Обсяг пам'яті в даному випадку визначиться як

$$Length_{memory} = 2^{\lceil \log_2 W \rceil + 1}. \quad (3.2)$$

Обсяг пам'яті обраний саме таким чином через те, що, якщо розглянути двійкове представлення адреси при такому варіанті, то сигнал вибору пам'яті, що належить керуючому блоку верхнього рівня декодеру, визначатиме половину пам'яті, в якій необхідно проводити операцію, тобто, стає частиною сигналу адреси. Він конкатенується до поданого сигналу адреси як старший біт. Оскільки сигнал може приймати значення 0 або 1, то таким чином відбуватиметься переключення з однієї половини пам'яті на іншу та навпаки. Витрата додаткових логічних ресурсів при цьому зведена до мінімуму, а використання пам'яті є таким самим, як і для попередніх випадків. На рис. 3.3 представлена модель подвійної буферизації на основі подвоєння пам'яті.

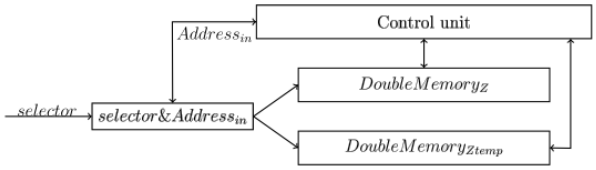


Рис. 3.3. Представлення моделі організації подвійного буферу на основі подвоєння обсягів пам'яті

На рис. 3.3, на відміну від попередніх, окремо виведений сигнал адреси для пам'яті, який конкатенується з сигналом вибору перед подачею порт адреси

пам'яті. Варто також зазначити, що операція конкатенації біту вибору пам'яті в даному випадку є асинхронною з метою зменшення витрат на додаткові службові сигнали.

Таким чином, математичне представлення розробленої моделі можна записати наступним чином:

$$\begin{aligned} &Length_{memory} = 2^{\lceil \log_2 W \rceil + 1}; \\ &selector = \begin{cases} 1, Iter \bmod 2 = 1, \\ 0, Iter \bmod 2 = 0; \end{cases} \\ &Address_{out} = selector \,\&\, Address_{in}, \end{aligned} \tag{3.3}$$

де $Iter$ – номер ітерації, $Address_{in}$ – вхідне значення адреси; $Address_{out}$ – результуюче значення адреси.

3.2 Підвищення швидкодії декодеру на основі особливостей блокової пам’яті ПЛІС

Представлені у даному розділі розроблені теоретичні положення [119] отримані на основі відомих робіт [44-48, 68, 74, 76-79, 83, 94, 96, 100].

Матриця перевірки парності H LDPC-декодеру має M рядків та N стовпців. Максимальну кількість значущих елементів у окремому рядку матриці позначимо як n_{max}. Для представлення H при реалізації декодеру з метою збереження пам'яті доцільно зберігати лише значення індексів значущих елементів, а не всю матрицю. Визначення для такого представлення елементів матриці є наступним:

$$ind_{ij} = ord_i(1_j); i = 1...M; j = 1...n_{max}, \tag{3.4}$$

де ord – функція, що повертає порядковий номер елементу в рядку.

Частково паралельний LDPC-декодер, робота якого організована на основі послідовної поелементної обробки повідомлень відповідно до індексів, що

наявні в поточному рядку, забезпечує уникнення колізій доступу до пам'яті. Така реалізація передбачає наявність лише одного функціонального блоку, який відповідає за операції зчитування-обчислення-запису. Відповідно, використовується лише один порт пам'яті, до якого підключений обчислювальний блок. Це означає, що можливість реалізації другого порту не задіяна, а, значить, не використовуються ресурси для забезпечення максимальної пропускної здатності.

Наявність двох повноцінних портів запису/зчитування в блоковій пам'яті ПЛІС означає, що можлива робота двох обчислювальних блоків, що можуть працювати паралельно. Проте, просте підключення другого блоку до пам'яті без попереднього аналізу розташування елементів та можливих перетворень може призвести до виконання некоректних обчислень через наявність колізій, у ході яких виконується обчислення для одного елементу.

Уникнення такого роду колізій можна забезпечити на основі попередньої обробки H. У випадку, якщо робота обчислювальних блоків організована таким чином, що кожен з них одночасно виконує обробку одного рядка, то для уникнення колізій потрібно забезпечити, щоб виконувалась наступна умова:

$$\forall ind_{i1j}, \forall ind_{i2j}, \neg\exists ind_{i1j} = ind_{i2j}, j = 1...n_{max}, \quad (3.5)$$

де $i1$, $i2$ – номери рядків, що обробляються першим та другим обчислювальними блоками. Відсутність повторів індексів у двох рядках означатиме відсутність колізій доступу до пам'яті та при виконанні обчислень, що дозволяє проводити обробку одразу двох рядків, а, значить, підвищити швидкодію в два рази.

Для забезпечення виконання умови (3.5) при попередній обробці H пропонується виконувати наступні дії. Вважатимемо, що H не містить двох рядків, що мають два однакові значення індексів ind_{ij}. У протилежному випадку, це означає наявність у матриці циклів з довжиною 4, що говорить про недостатню придатність такої матриці для декодування через утворення

поглинаючих множин. Також позначимо як m_{max} максимальну кількість значущих елементів, що наявні в одному стовпці матриці. Для попередньої обробки матриці використаємо лише один тип елементарних перетворень – перестановку рядків.

Реалізацію перестановки рядків для уникнення колізій будемо проводити за допомогою заповнення результуючої матриці, яка початково є порожньою, рядками вихідної матриці, що еквівалентно проведенню серії операцій перестановок рядків.

Для використання пропонується використовувати наступний алгоритм. Перший рядок вихідної матриці додається у першу позицію результуючої матриці. Перевіряється можливість додавання наступного рядка у наступну вільну позицію. Якщо немає можливості додати поточний рядок, то переходять до наступного. Подібна перевірка проводиться до моменту, поки не буде додано новий рядок. Індекс доданого рядка запам'ятовується та більше не використовується у ході додавання. Умовою завершення даного процесу є заповнення результуючою матриці рядками вихідною. При цьому, в найгіршому випадку можлива ситуація, коли $m_{max} - 1$ рядків не вдається додати таким чином, через те, що вони всі містять індекс, що знаходиться у рядку, що був доданий перед цим. У такому випадку поточний рядок додається до таблиці, але для нього одразу шукається перестановка, яка дозволить зберегти умову неповторюваності індексів для сусідніх рядків, з якими виконується перестановка. Як тільки така перестановка знайдена, так само виконується перевірка можливості додавання наступних рядків, поки не будуть додані всі рядки. Псевдокод для отримання матриці представлено на рис. 3.4.

```
function PERMUTEROWS(inputMatrix)
    newMatrix(1) ← inputMatrix(1)
    permut_ind ← 2
    start_ind ← 2
    while permut_ind < length(inputMatrix) do
        prev_row ← newMatrix(permut_ind − 1)
        cur_ind ← start_ind
        check ← true
        while check do
            temp_row ← inputMatrix(cur_ind)
            check ← hasSameElements(prev_row, temp_row)
            if check then
                newMatrix(permut_ind) ← temp_row
                permut_ind ← permut_ind + 1
                addToUsed(temp_row)
                start_ind ←= findNextStartInd()
                break
            else
                check ← true
                cur_ind ← cur_ind + 1
                if cur_ind == length(inputMatrix) then
                    newMatrix(permut_ind) ← temp_row
                    for i ← 1 to permut_ind − 1 do
                        permuteRows(inputMatrix, i, permut_ind)
                        check1 ← hasSameElements(i, i + 1)
                        if i ≠ 1 then
                            check2 ← hasSameElements(i, i − 1)
                        else
                            check2 ← true
                        end if
                        check3 ← hasSameElements(prev_row, prev_row − 1)
                        if check1 & check2 & check3 then
                            break
                        else
                            rollbackPermutation()
                        end if
                    end for
                end if
            end if
        end while
    end while
    return newMatrix
end function
```

Рис. 3.4. Псевдокод алгоритму перестановки рядків

На початковому етапі заповнення матриці рядками контролюється умова неповторюваності лише відносно індексів попереднього рядка для рядка, який є поточним. Проте, як тільки вільні рядки завершуються, та необхідно використовувати перестановки в самій результуючій матриці, перевірка умов неповторюваності проводиться для обох суміжних рядків, з яким проводиться

перестановка. У випадку, якщо за вказаним алгоритмом не вдається розмістити усі рядки, то їх обробку можна організувати лише для одного блоку декодування.

Варто також зазначити, що зміна послідовності появи індексів у рядку для уникнення колізій не може використовуватися, оскільки слід враховувати увесь цикл обробки – зчитування-обчислення-запис. Якщо при зчитуванні колізій не виникає, то виконання обчислень з подальшим записом даних означатиме, втрату результату обчислення, що завершиться та буде записано першим. Таким чином, не буде врахована інформація, отримана від усіх вузлів.

Отримана матриця індексів при реалізації декодеру має бути описана як двопортова блокова пам’ять. Кожному стовпцю матриці у такому випадку відповідатиме окремий модуль пам’яті для збереження індексів. Два блоки декодування використовують окремі порти пам’ять для зчитування. При цьому робота блоків декодування має бути організована таким чином, щоб один блок відповідав за обробку парних рядків, а інший – непарних. Обидва блоки працюють паралельно, тому швидкість обробки усієї матриці збільшується в 2 рази. Доступ до елементів у пам’яті результуючого повідомлення також виконується з використання різних портів блокової пам’яті. Оскільки одночасно відбувається звернення за різними адресами, то вдається уникнути колізій.

Важливим моментом є також синхронізація паралельної роботи двох блоків декодування. Для коректної роботи пропонується наступний механізм синхронізації. За наявності двох модулів можлива ситуація при перевірці синдрому, що кількість помилок для одного блоку виявилась рівною нулю, в той час як при перевірці синдрому іншим блоком помилки були виявлені. У такому випадку слід забезпечити продовження роботи першого блоку, оскільки при подальшому декодуванні та корекції, може виявитися, що помилкові біти перемістились до рядків, що обробляються першим блоком. Тому синхронна робота та її закінчення для обох модулів є обов'язковими. Таким чином, умова завершення роботи декодеру є наступною:

$$finish = (iter_{current} = iter_{max}) \vee (error_1 = 0 \wedge error_2 = 0), \quad (3.6)$$

де $iter_{current}$ – номер поточної ітерації;

$iter_{max}$ – максимальна встановлена кількість ітерацій;

$error_1$, $error_2$ – кількість помилок перевірки парності, виявлених двома блоками декодування на поточній ітерації. Умова (3.6) є найбільш загальною, проте можуть використовуватися також інші показники з урахуванням додаткової специфіки реалізації.

Як зазначалось, при реалізації декодеру, стовпці H будуть представлені як двопортова блокова пам'ять. Такий підхід дозволяє отримувати доступ до індексів у рядках паралельно за 1 такт. Проте це означатиме необхідність великої кількості портів блокової пам'яті. Даний фактор негативно впливає на максимально можливу робочу тактову частоту, оскільки порт блокової пам'яті є складним у реалізації.

Специфіка частково паралельного декодеру, що досліджується, полягає у тому, що обробка даних на основі отриманих індексів відбувається поелементно. Дана обставина дозволяє зменшити кількість окремих блоків пам'яті та об'єднати їх в один, організувавши послідовне зчитування індексів. При цьому швидкість роботи декодеру не зменшиться, оскільки обробка виконується поелементно. При використанні такої організації пам'яті індексів є можливість збільшити робочу тактову частоту пристрою.

Заміна окремих блоків пам'яті на один передбачає, що необхідно створити блокову пам'ять для наступної кількості елементів:

$$mem_{length} = 2^{\lfloor \log_2 n_{max} * M \rfloor}. \quad (3.7)$$

Заповнення пам'яті індексів відбуватиметься наступним чином:

$$ind_k = ind_{i=k/n_{max},\, j=k \bmod n_{max}},\ k = 1...mem_{length}. \quad (3.8)$$

У такому випадку два блоки декодування підключаються до двох портів пам’яті та у ході обробки зчитують послідовно n_{max} значень. Крім того, оскільки обробка даних також потребуватиме час на виконання, то випереджаюче зчитування під час обробки дозволить отримати усі індекси паралельно при наступному запиті.

Для функціонування декодеру важливою є розрядність при представленні даних м’яких рішень. Даний параметр також впливає на використання пам’яті при побудові декодеру. Вибір розрядності для представленні даних у ході обробки має базуватися на тому, при якій кількості помилок у вхідному повідомленні декодер зможе забезпечувати їх декодування, а також на обмеження відносно пам’яті.

3.3 Розподіл функцій у комп’ютерних системах з модулем ПЛІС

Представлені у даному розділі розроблені теоретичні положення [120-123] отримані на основі відомих робіт [27-30, 124, 125].

До складу НКГМС входять наступні керуючі пристрої: MCU, FPGA, GPS/GSM-модуль. Використання мікроконтролерів у даній області пояснюється тим, що за допомогою їх апаратного забезпечення є можливість вирішення різних задач. Особливістю мікроконтролерів, що орієнтовані на роботу з зображеннями, є наявність акселератора для обробки зображень, наприклад, Chrom-Art Accelerator DMA2D. Це спеціальний механізм DMA, призначений для роботи з зображеннями. Він забезпечує рисування та копіювання всього або окремих частин зображення. У зв’язку з наявністю модуля роботи з зображеннями не можна не згадати про наявність інтерфейсу керування LCD-TFT-дисплеями з максимальною роздільною здатністю монітора 600x800 пікселів [124]. Тим не менш, оскільки такі апаратні можливості присутні не у всіх контролерів, виділимо згадані контролери в окрему групу.

Для FPGA, незважаючи на гнучкість, яку вони надають, критичним фактором є кількість логічних елементів, наприклад, реалізація всього двох алгоритмів для знаходження координат малювання примітивів (без виконання малювання) займає 7% логічної ємності пристрою. Таким чином, реалізація більш вагомої функціональності займе набагато більше ємності, що потрібно враховувати при проектуванні системи.

GPS/GSM-модуль представляє собою особливий пристрій, орієнтований саме на забезпечення комунікаційних можливостей та можливостей позиціонування. З його допомогою зручніше за все реалізовувати подібні системи, завдяки наявності вбудованих AT-команд, які надають велику кількість готового функціоналу. Окрім того, вказаний у якості прикладу модуль здатен виконувати роботу з аудіо (голос). У нових пристроях подібного типу є ще одна особливість: основою для його побудови є мікропроцесор з потужним ядром, наприклад, ARM. Завдяки цьому розширюються можливості створення програмних користувацьких застосувань саме за допомогою даного модуля.

На основі проаналізованого матеріалу визначено, що розглянуті пристрої не можна безпосередньо порівнювати за визначеними ознаками. Тому для візуалізації можливостей пристроїв відносно один одного створено діаграму, на якій виділено задачі та продуктивність, яку пристрої показують при їх вирішенні (рис. 3.5).

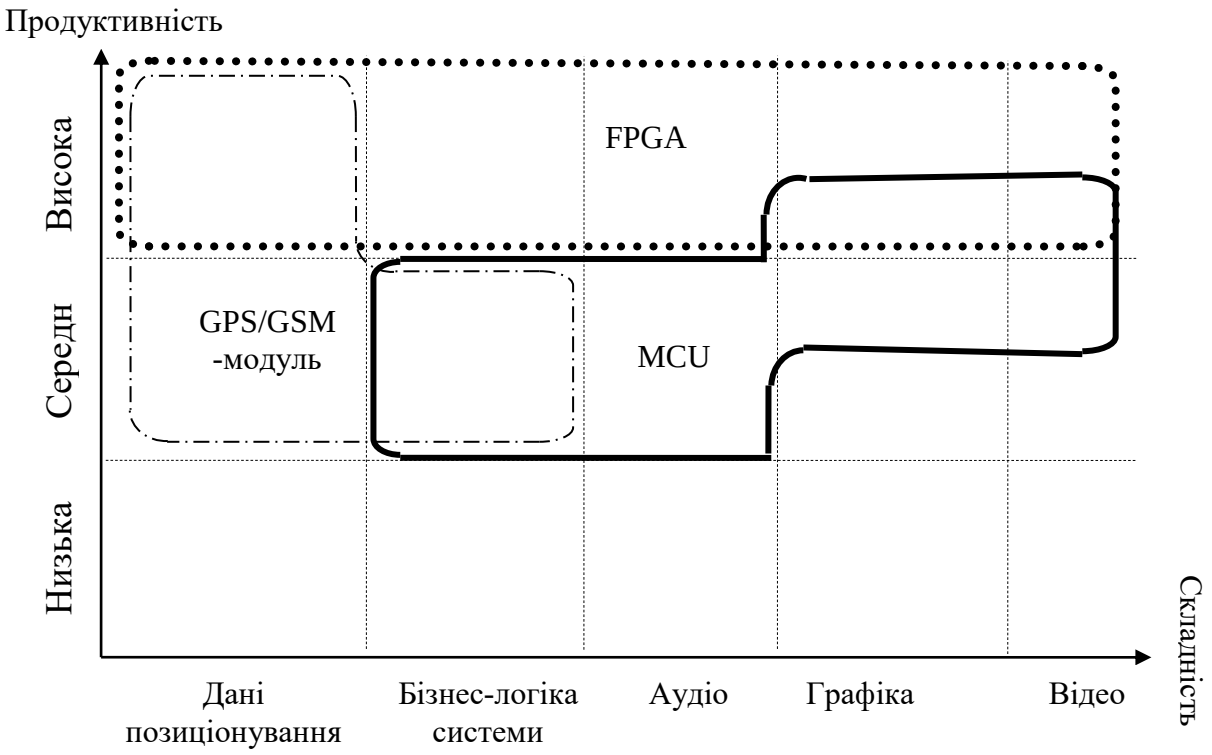


Рис. 3.5. Діаграма відповідності функціональності пристроїв задачам в системі

З діаграми видно, що найкраще для виконання усіх перелічених задач підходить модуль FPGA, проте він має суттєво обмежені ресурси, тому перенесення більшої частини функціональності на зазначений модуль є проблематичним. Те ж саме можна зазначити відносно інших модулів: незважаючи на здатність покрити виконання кількох задач, слід враховувати ресурси модулів.

У зв’язку з тим, що спостерігається перетин функціональних можливостей пристроїв у відповідності з тим, які конкретні вимоги висуваються відносно продуктивності системи в цілому, а також неможливістю абсолютного порівняння характеристик пристроїв, використаємо апарат методу аналізу ієрархій (MAI) [125] з оцінкою альтернатив за визначеними критеріями (рис. 3.6). У результаті використання методу отримаємо рекомендований розподіл функціональності між пристроями, на основі якого можна визначити остаточний склад мультипроцесорної системи.

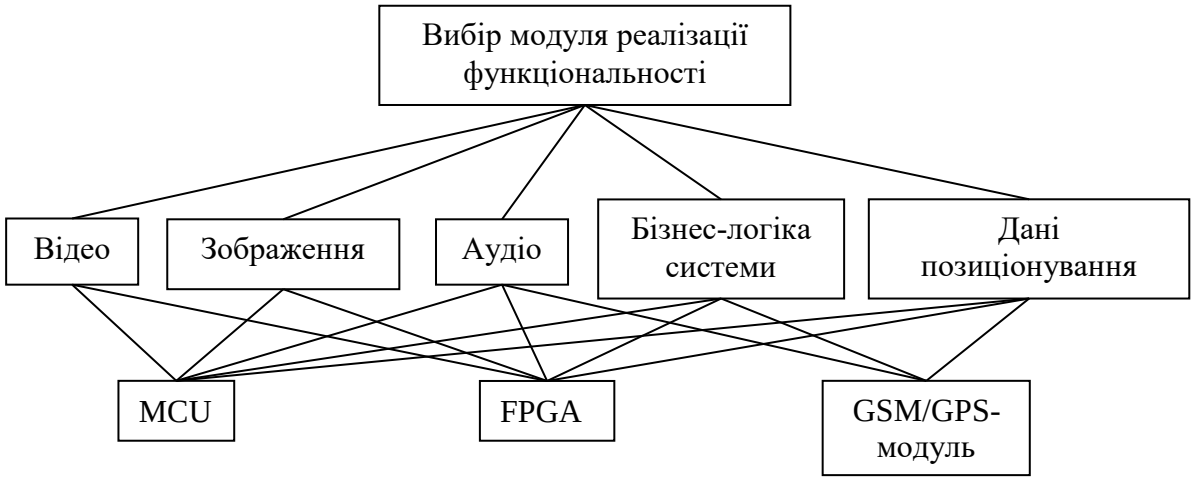


Рис. 3.6. Структура системи для розподілу функціональності

У даному випадку маємо множину з 3 можливих альтернатив

$$A = \{ A_i \}; i = \overline{1, n}; n = 3, \tag{3.9}$$

які порівнюємо за елементами множини критеріїв

$$K = \{ K_j \}; j = \overline{1, m}; m = 5. \tag{3.10}$$

За кожним з критеріїв з множини K складемо оберненосиметричну матрицю попарних порівнянь альтернатив для цього критерію [125] (оцінки a обираємо в діапазоні від 1 до 9, а для обернених елементів від 1 до 1/9; розмірність матриці 3x3 або 2x2, в залежності від кількості альтернатив, що оцінюються за критерієм)

$$M_j = \begin{pmatrix} 1 & a^j_{1n-1} & a^j_{1n} \\ 1/a^j_{1n-1} & 1 & a^j_{2n} \\ 1/a^j_{1n} & 1/a^j_{2n} & 1 \end{pmatrix}. \tag{3.11}$$

Також складемо матрицю попарних порівнянь критеріїв [125] (розмірність матриці – 5x5)

$$M_k = \begin{pmatrix} 1 & \dots & a_{1m} \\ \vdots & \ddots & \vdots \\ 1/a_{1m} & \dots & 1 \end{pmatrix}. \qquad (3.12)$$

Далі обчислюємо вектори пріоритетів альтернатив за кожним критерієм [8]. Елементи вектору знаходимо як відношення суми елементів рядку матриці до суми усіх елементів матриці. Позначимо e_{ij} елемент вектора пріоритетів альтернатив, знайденого для критерію K_j. За аналогією позначимо e_{kj} елементи вектора пріоритетів критеріїв, а сам вектор як X. Елементи векторів є нормованими.

Сформуємо матрицю спеціального вигляду, елементами якої є значення елементів векторів пріоритетів для кожної альтернативи. Якщо відповідні елементи відсутні, вони замінюються нулями [125]. Елементи матриці позначимо як a'_{ij}:

$$A = \left(a'_{ij}\right). \qquad (3.13)$$

Введемо структурну матрицю L, яка є діагональною, елементами якої є відношення кількості елементів, відмінних від 0, у матриці A для критерію K_j, до загальної кількості елементів відмінних від 0.

Тоді пріоритети альтернатив визначаться як [125]

$$W = ALX^T, \qquad (3.14)$$

де W – ненормований вектор пріоритетів альтернатив;

L – структурна матриця;

X^T – вектор пріоритетів критеріїв.

Оскільки отримані значення пріоритетів не є нормованими, то введемо матрицю B [125], яка є діагональною, значення елементів на діагоналі дорівнює

$$\left(\sum_{i=1}^{n} w_i\right)^{-1}. \qquad (3.15)$$

Добуток матриць

$$W' = WB, \qquad (3.16)$$

де W' – нормований вектор пріоритетів альтернатив;

B – матриця нормування, дозволяє отримати нормований вектор пріоритетів альтернатив [125]. Оскільки значення вектора пріоритетів представляють собою значення залучення конкретного модуля до реалізації конкретної функції, ці значення разом з векторами альтернатив по кожному критерію (функціональність) можна використовувати при розподілу навантаження між модулями навігаційно-комунікаційної мультипроцесорної системи.

Як інший варіант функціонального розподілу, розглянемо комп'ютерну систему керування позиціонування головки 3D-системи (принтеру, станка з ЧПУ та ін.), побудованих з використанням двох обчислювальних ресурсів – одного MCU та одного модулю FPGA. Частіше за все такі системи будуються на основі використання одного MCU, який, окрім виконання розрахункових операцій, виконує також інші функції: передачу керуючих імпульсів, прийом інформації з датчиків з її обробкою, контроль стану системи з урахуванням показників точності та надійності. Незважаючи на можливість використання механізму прямого доступу до пам'яті (DMA – Direct Memory Access), що дозволяє зняти певне навантаження з основного модулю, програмна модель MCU, як правило, не передбачає можливість паралельного виконання команд.

На відміну від MCU, модулі FPGA надають можливість досягти паралельного виконання інструкцій, за рахунок чого швидкість обробки даних FPGA в багато разів перевищує швидкість MCU: наприклад, для задачі побудови графічних примітивів у 2D-просторі FPGA використовує всього 4

тактових імпульси на один цикл інтерполяції, в той час як показники MCU для лінійної і, особливо, для кругової інтерполяції значно більші – десятки тактових імпульсів. При розрахунках для третьої осі координат (3D-моделі) перевага буде ще більшою.

Одночасне ж використання двох модулів дозволяє отримати значний виграш у швидкодії у порівнянні з використанням одного MCU для тієї самої задачі. Основним питанням при цьому залишається, яким чином оптимально провести розпаралелювання обчислювальні задачі між модулями.

При організації паралельної взаємодії MCU та FPGA також виникає задача синхронізації, яку можна вирішити методом «рандеву», при якому керуючий пристрій блокує подальші дії до отримання сигналу від розрахункового модулю.

Схематично розподіл задач між модулями представлено на рис. 3.7.

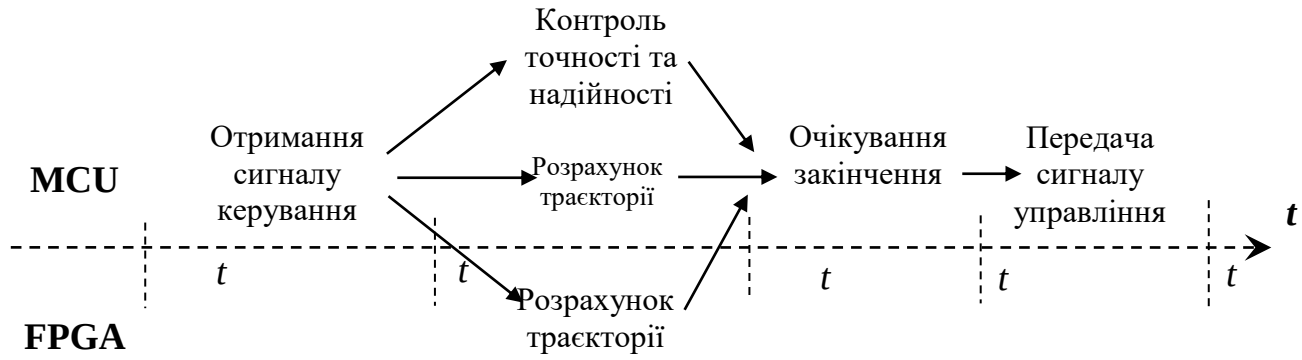


Рис. 3.7. Розподіл задач між модулями

Найбільша ефективність всієї системи буде досягнута при мінімальних простоях обчислювальних модулів. У такому випадку це відповідає одночасному завершенню розрахунків обох модулів у момент часу t_2.

Введемо показник коефіцієнту корисної роботи модулю:

$$x = \frac{U_p}{U_a} \cdot 100\%, \tag{3.17}$$

де U_p – кількість тактів, що використовуються на корисну роботу, U_a – загальна кількість тактів на вимірюваному діапазоні.

Для визначення ефективності будемо використовувати показник квадратичного відхилення σ, який визначає, наскільки рівномірно розподілене навантаження між модулями.

У такому випадку оптимальним рішенням буде виконання умови:

$$\begin{cases} \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2} \to min; \\ x_i \to max; i = (1,2); n = 2. \end{cases} \quad (3.18)$$

Для експериментальних досліджень в якості MCU був використаний мікроконтролер архітектури ARM Cortex-M4 на базі відлагоджувального модулю STM32F4 Discovery. В якості модулю FPGA використана плата з Xilinx Spartan 6. Для передачі даних між модулями використовувався інтерфейс SPI.

Отримані експериментальним шляхом результати дослідження наведені на рис. 3.8.

Як видно з рис. 3.8, оптимальне вирішення задачі, тобто, виконання умови, виконується при розподілі обчислювальних ресурсів між модулями FPGA та MCU рівному 65 та 35% відповідно.

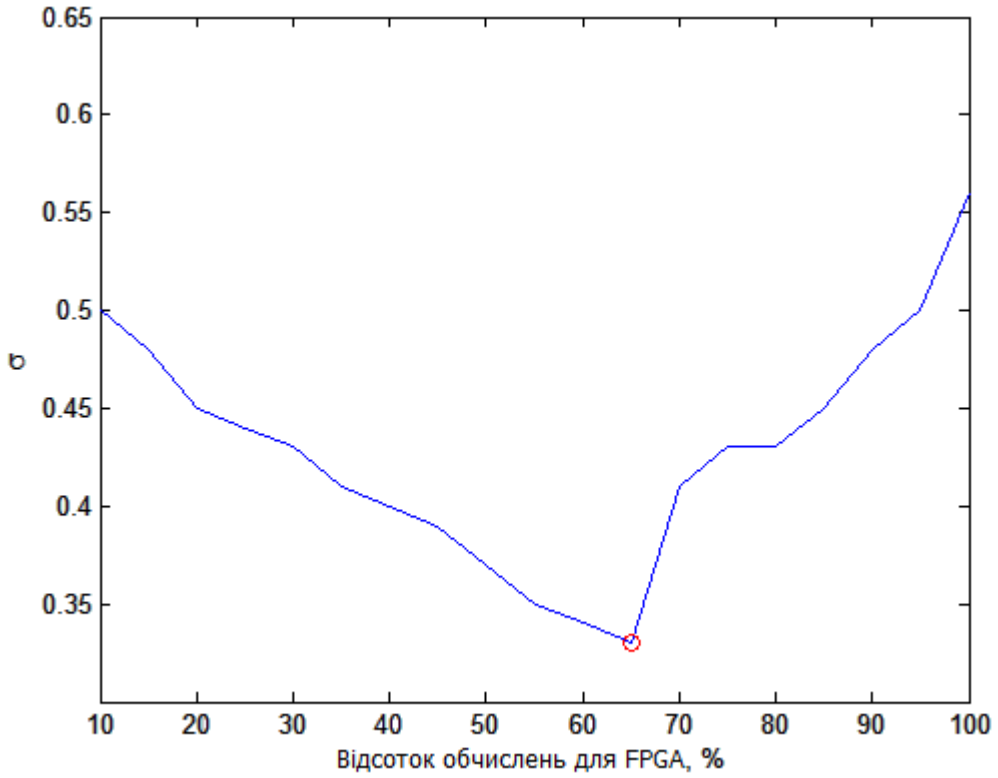


Рис. 3.8. Графік залежності σ від долі обчислювального навантаження для FPGA

3.4 Створення систем управління GPS/GSM-пристрою на базі інтерфейсу USB

Представлені у даному розділі розроблені теоретичні положення [126, 127] отримані на основі відомих робіт [128-134].

Розвиток технологій у сфері створення систем управління зумовлює постійне зростання вимог щодо функціональності та продуктивності таких систем. При цьому подібні системи можуть використовуватися і для роботи з вже існуючими системами, що накладає суттєві обмеження на їх роботу: максимальна швидкість отримання інформації, доступні інтерфейси інтеграції, забезпечення доступу до складових системи, можливість повного контролю над системою та ін. Прикладом такої системи може бути існуюча MCU-система, для якої необхідно створити систему управління. При цьому,

якщо відповідно до вимог ставиться завдання забезпечення інтеграції системи управління з такими апаратними засобами як комп’ютер, мобільні пристрої, це ще більше ускладнює розвиток розробки, оскільки апаратні можливості для передачі інформації цих пристроїв також слід враховувати при вирішенні даної проблеми. Найбільш поширеним інтерфейсом передачі даних для вказаних пристроїв є інтерфейс послідовної передачі даних USB, тому забезпечення роботи саме з цим інтерфейсом надасть системі максимальної гнучкості при виборі пристроїв управління.

Таким чином, створення систем управління GSM/GPS/MCU-системами є задачею, що потребує комплексного підходу до її вирішення та базується на виборі апаратних засобів, а також розробці програмного забезпечення для всіх складових системи управління, яке надасть можливість реалізувати закладений функціонал для контролю системи та застосувати можливості інтерфейсу USB для отримання максимальної ефективності роботи.

Отже, задача є актуальною та важливою.

Найчастіше GSM/GPS/MCU-системи є комплексними, спрямованими на вирішення широкого кола задач, тому включають до свого складу велику кількість компонентів. Також вони потребують розробки програмного забезпечення для організації взаємодії складових системи, що підвищує її загальну складність. Підключення до такої системи є складним та не може відбуватися безпосередньо у будь-який момент через відсутність відповідних інтерфейсів зв’язку.

Дослідження проводились на системі бездротової передачі даних для рухомих об’єктів, що включає до свого складу GSM/GPS-модуль, мікроконтролер та використовує в якості керуючої системи промисловий ПК з операційною системою Windows.

Для проведення дослідження вказана система модифікована таким чином, як показано на рис. 3.9. Засобом для інтеграції USB є апаратний USB-міст, що має два канали прийому/передачі інформації. Безпосередньо від USB-мосту інформація надходить до управляючого пристрою. Від GSM/GPS-модуля

надходить інформація про передачу даних всередині бездротової мережі, а також інформація про розташування від GPS-частини модуля. Цим пояснюється необхідність використання саме двоканальнального мосту. Крім того, до системи включено також додатковий модуль обробки інформації, в ролі якого виступає MCU. Доцільність його використання продемонстрована нижче.

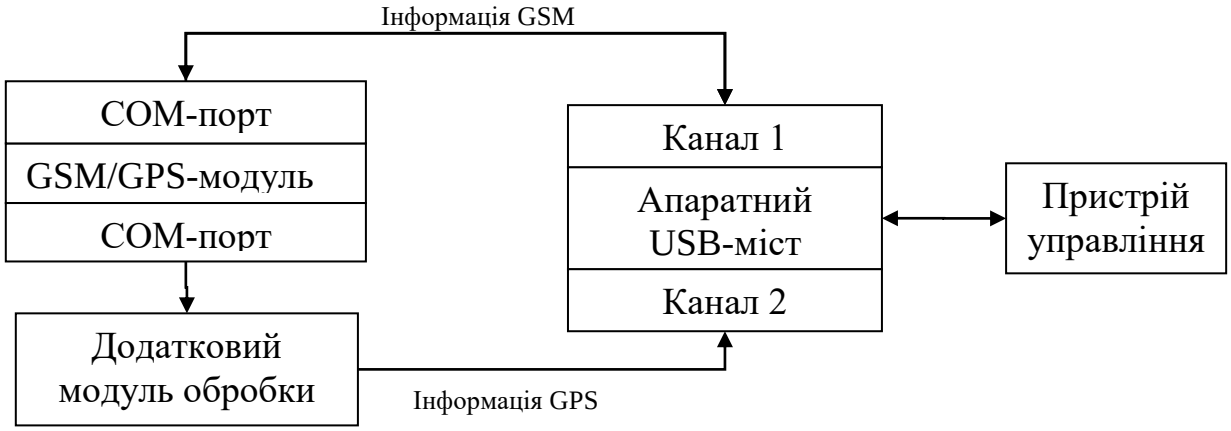


Рис. 3.9. Структура системи для проведення досліджень

Основними виробниками апаратних USB-мостів є фірми Cypress та FTDI. Перевагою продукції Cypress є більша функціональність, проте вона дорожча. Продукція FTDI надає однакові з Cypress базові можливості, однак виграє у неї в ціні [129, 130]. Тому для використання в системі обрано саме модуль компанії FTDI FT2232H.

У ході досліджень використано розроблене програмне забезпечення для управління бездротової та навігаційною складовою системи. Розроблено програмне забезпечення з використанням мови програмування Java. Цільова операційна система пристрою – Android 4.0 та вище [131]. Для розробки використано середовище програмування Intellij IDEA Community Edition 12.1.4.

Складова частина, що відповідає за роботу з GSM-частиною реалізує основні функції пристрою для роботи з повідомленнями, а також надає можливість роботи зі специфічними повідомленнями, призначеними для

використання всередині системи. За допомогою навігаційної функціональності, що надає розроблене застосування, користувач може:

– налаштувати точки відправлення повідомлень усередині мережі;

– вказати місця прийому інформації від інформаційних агентів системи та налаштувати точність прийому для конкретної точки з міркувань безпеки та надійності роботи системи;

– отримати статистику за службовими повідомленнями в мережі для кожного вузла, від якого отримана інформація;

– вказати маршрут транспортного засобу, за яким здійснюватиметься збір інформації.

Для завантаження та відображення мап обрано сервіс Google Maps Android API v2 [132]. Доступ до сервісу потребує підключення до мережі Internet, тому пристрій управління повинен бути підключений до мережі, щоб отримати географічні дані. Інтерфейс користувача розробленої системи при роботі з навігаційною складовою наведений на рис. 3.10.

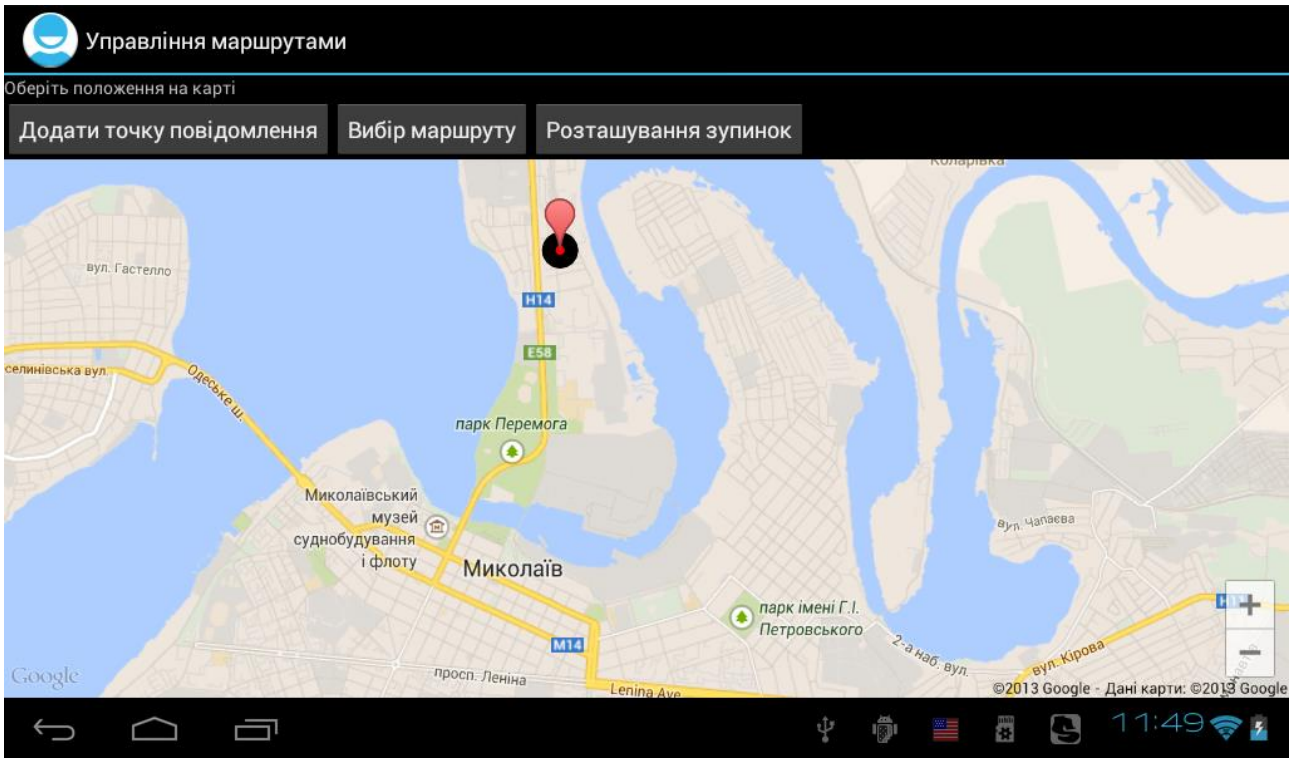


Рис. 3.10. Інтерфейс користувача розробленого програмного забезпечення для платформи Android

У результаті дослідження виявлено, що при необхідності відображення великої частини графічної інформації, якими є карти, та одночасного виконання розрахунків для маршрутів та прийому даних від GPS-частини, пристрій потребує використання суттєвих обсягів оперативної пам’яті. Так для вказаної системи експериментальним шляхом отримане значення у 44% використання загальної оперативної пам’яті. Потенційною можливістю для зменшення показника використання оперативної пам’яті є аналіз даних, що надходять від модулю, за протоколом NMEA 0183 [133, 134] та відслідковуванням змін в отриманих даних та передачі лише керуючому пристрою лише інформації про зміни. Для зменшення цього показника пропонується використати додатковий мікроконтролер, який виконуватиме функцію фільтрації повідомлень від GPS-блоку та надсилатиме лише визначену важливу інформацію до пристрою управління. Це дозволить уникнути дублювання інформації без втрати інформативності.

Для проведення дослідження використано наступні апаратні засоби:

– GSM/GPS-модуль – відлагоджувальний модуль на базі SIMCOM SIM 508 [38];

– USB-міст FTDI FT2232H;

– мікроконтролер PIC18F252;

– планшет GoClever A103 з операційною системою Android 4.0.4.

Результати показали, що використання оперативної пам’яті вдалося зменшити до 35% (рис. 3.11).

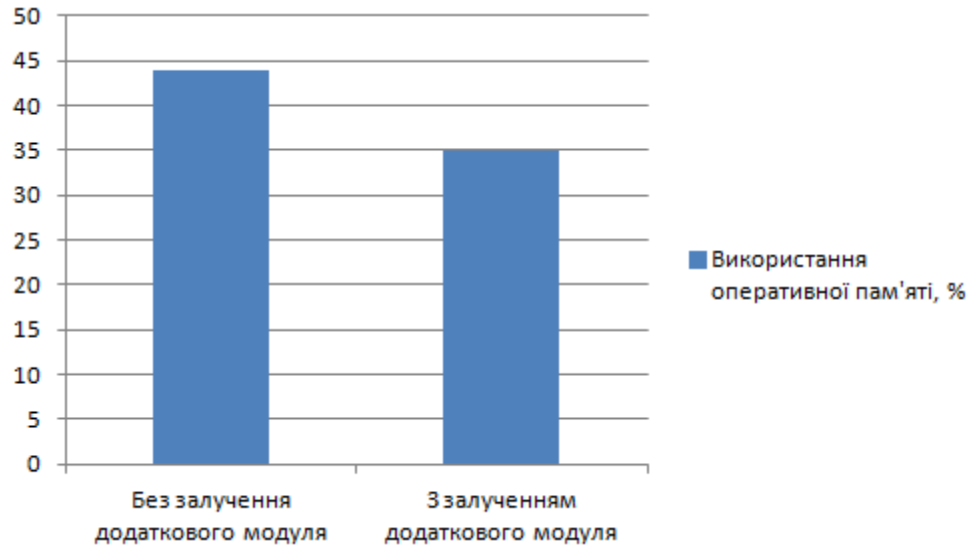


Рис. 3.11. Гістограма використання оперативної пам’яті в залежності від залучення додаткового модуля

Таким чином підтверджена доцільність використання мікроконтролера в якості додаткового модуля обробки та фільтрації даних для зменшення використання ресурсів управляючим пристроєм.

3.5 Реалізація обчислень алгоритмів Брезенхема для ПЛІС у складі комп’ютерної системи

Представлені у даному розділі розроблені теоретичні положення [135-139] отримані на основі відомих робіт [140-144].

У 3D-принтері траєкторію руху екструдеру необхідно вказувати відносно трьох осей. При цьому слід враховувати, що точність переміщення можна забезпечити за рахунок збільшення кількості дискретних кроків (положень) керованого пристрою на всій траєкторії. Таким чином, розрахунок траєкторії руху для трьох осей з урахуванням вимог високої точності позиціонування є задачею, що потребує значних обчислювальних ресурсів від керуючого пристрою. Частково, цим пояснюється низька швидкодія систем 3D-друку.

Часто керуючим пристроєм в системах позиціонування екструдеру виступає MCU, на який покладено виконання всіх функцій. Розглянемо

можливі недоліки такого рішення, а також можливість підвищення керуючої системи за рахунок використання FPGA у якості модуля для проведення обчислень.

Проблема позиціонування екструдеру включає в себе велику кількість аспектів, серед яких необхідно виділити швидкодію самої системи керування позиціонування, оскільки це впливає на кінцеву швидкість друку в цілому. Тому забезпечення максимального швидкодії при проведенні обчислювальних операцій є важливим для системи управління на базі MCU/FPGA.

Часто в системах позиціонування використовується пропорційно-інтегрально-диференціальний регулятор, однак процес розрахунку для такого регулятору є ресурсоємким та потребує великого числа обчислювальних операцій, що негативно впливає на швидкодію системи. Тому метою роботи є дослідження алгоритмів, які здатні забезпечити виконання операцій позиціонування за рахунок меншої кількості обчислювальних операцій, а також можливість заміни складних операцій на ділення та множення на більш прості, наприклад, виконання арифметичного зсуву цілого числа та ін. [57-59].

Проведені дослідження продуктивності керуючих модулів системи в ході виконання розрахунків за алгоритмами Брезенхема, які дають можливість отримати цілочислове рішення. Для отримання результатів для мікроконтролеру використовувалась максимальна оптимізація коду за швидкодією. Отримані результати представлені в таблицях 3.1 та 3.2.

Таблиця 3.1

Кількість тактів, необхідних на розрахунки для лінії

Вихідні дані	$p_1(0,0);\ p_2(3,-3)$	$p_1(0,0);\ p_2(3,-3)$	$p_1(0,0);\ p_2(3,-3)$	$p_1(0,0);\ p_2(3,-3)$
Перша ітерація	13	13	13	13
Повний розрахунок	66	77	88	99

При розрахунках для побудови лінії на першу ітерацію затрачується однакова кількість тактів – 13, в той час, як при розрахунку для кола тривалість першої ітерації змінюється в залежності від початкових координат тат радіусу.

Кількість тактів, що використовуються на весь цикл розрахунку для випадку лінії поступово збільшується в залежності від її довжини, а також від її нахилу: чим менше нахил лінії, тим менше операцій присвоєння необхідно виконати.

Таблиця 3.2

Кількість тактів, необхідних на розрахунки для кола

Вихідні дані	$C(0,0), R = 3$	$C(1,1), R = 5$
Перша ітерація	46	67
Повний розрахунок	132	191

Крім того, з таблиць видно, що виконання однієї ітерації для обчислення наступної точки для кола потребує набагато більше часу, ніж для розрахунків лінії.

Для отримання результатів розрахунків FPGA для побудови лінії використаний стандартний алгоритм, а для кола – модифікований алгоритм Брезенхема, який заснований на операції відображення точок з розділенням площини побудови на вісім октантів, що дозволяє уникнути великої кількості розрахункових операцій.

Алгоритмічна реалізація використаного методу для побудови кола може бути представлена автоматом, граф станів якого представлено на рис. 3.12.

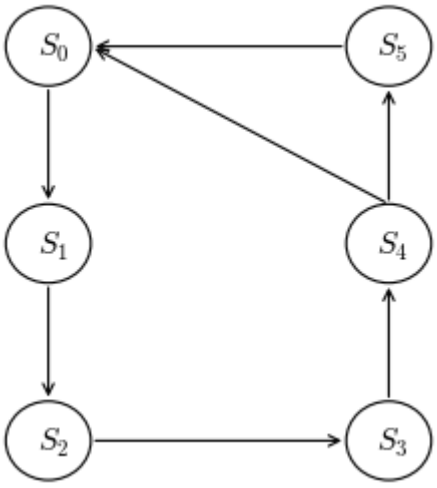


Рис. 3.12. Граф станів для реалізації модифікованого алгоритму Брезенхема для побудови кола

Стани, представлені на рисунку 1, мають наступне функціональне призначення:

– S_0 – стан ініціалізації;

– S_1 – отримання першого значення для відображення;

– S_2 – корекція параметрів з урахуванням значень першого розрахунку;

– S_3 – виконання відображення та отримання значень координат для всіх точок кола;

– S_4 – виведення отриманих значень на виходи пристрою;

– S_5 – відправка сигналу про завершення розрахунку.

Реалізація алгориму Брезенхема для FPGA виконана з використанням мови VHDL. Для перевірки результатів роботи модифікованого алгоритму для кола використане середовище розробки Xilinx ISE 14.2 та програмний симулятор ISim. Результати моделювання представлені на рис. 3.13.

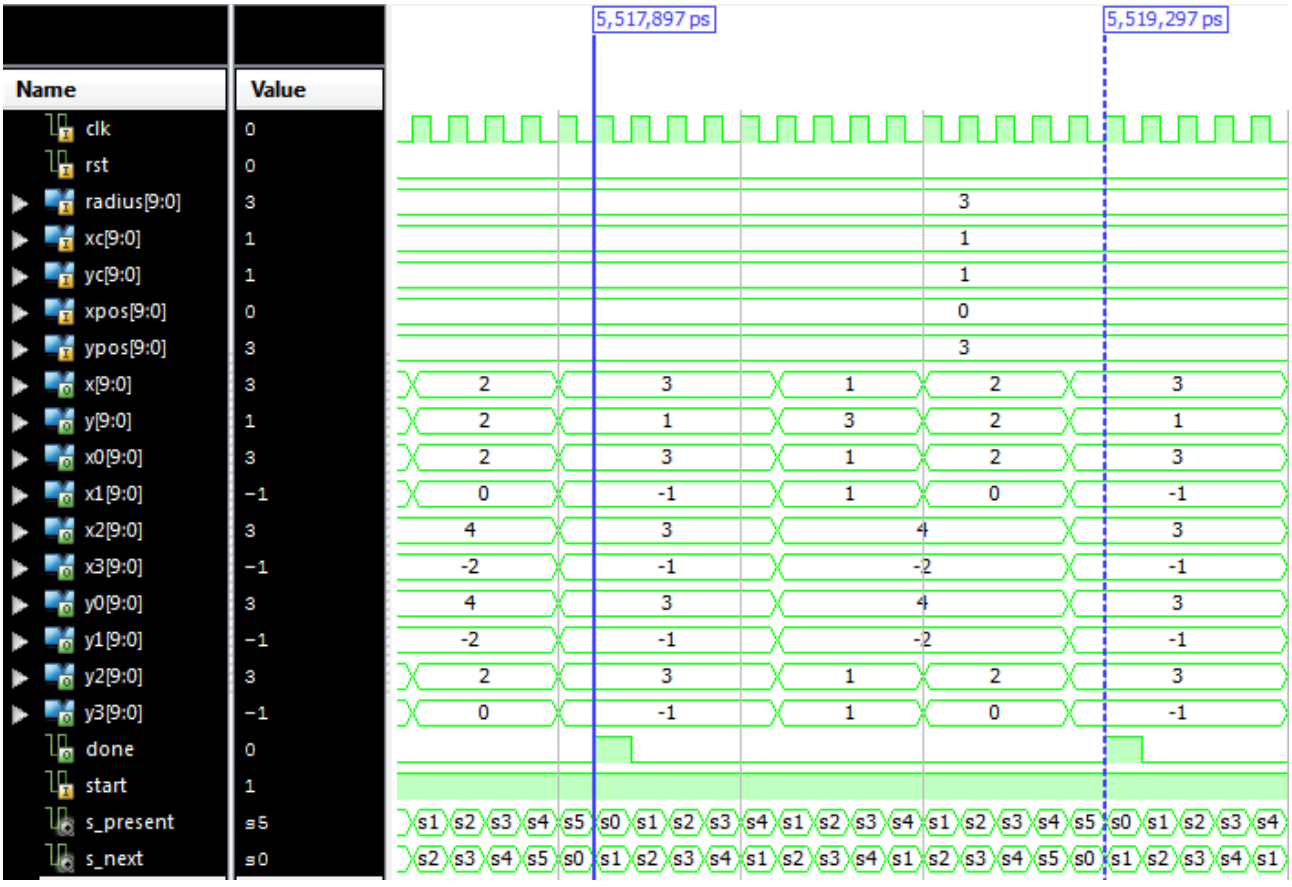


Рис. 3.13. Результати моделювання реалізованого алгоритму

На рис. 3.13 добре видно, що на обчислення наступної точки витрачається лише чотири тактових імпульси після подачі сигналу про початок обчислення (сигнал на лінії clk). Також вказані зміни станів у процесі обчислення, відповідно до розглянутого вище графа.

Таким чином, виявлено, що при використанні FPGA в якості основного обчислювального модулю можна отримати значний виграш у швидкодії системи, який залежить від кількості точок, які використовуються в ході траєкторії руху екструдеру. Цього вдалося досягти за рахунок зменшення кількості виконання операцій, а також за рахунок складних операцій більш простими.

Інтерполятор у системах числового програмного управління (ЧПУ) виконує розрахунок координат точок, за якими виконується подальший розрахунок траєкторії руху інструменту. За результатами роботи інтерполятору можна скласти лише наближену схему переміщення, тим не менш, від розрахунків на даному рівні залежить якість вихідного виробу, оскільки вони є базовими.

Вхідними даними для розрахунків у інтерполяторі є дані рядка G-кодів. Найчастіше, з точки зору друку виробу, використовуються команди побудови геометричних примітивів: лінії (G00, G01) та дуги (G02, G03). Саме в дані команди перетворюються траєкторії, які описані за допомогою ліній, сплайнів, кривих у CAD-засобах 3D-візуалізації. З вхідного рядка отримується інформація про наступну точку переміщення, після чого проводиться інтерполювання для вказаної точки.

У проекті RepRap для розрахунків при побудові дуги використовуються обчислення з плаваючою десятковою точкою. Але даний підхід не підходить для реалізації в FPGA, оскільки операції для чисел з плаваючою десятковою точкою потребують значних апаратних ресурсів, а також є менш продуктивними у порівнянні з цілочисловими. Тому в роботі [140] запропоновано використання алгоритмів Брезенхема для кола та прямої, що

дозволило використовувати саме цілочислові обчислення. У той же час нерозглянутою є проблема побудови дуги на основі даного підходу.

Відповідно до формату стандартних команд кругової інтерполяції (G02 та G03) для побудови дуги повинні бути відомі кінцева точка, в яку необхідно переміститися з поточної точки, а також координати центру кола, на якому знаходиться дуга. При відомих значеннях початкових координат початкової та кінцевої точок є можливість розрахувати центр кола [139].

При цьому слід враховувати, що обчислення центру кола для підвищення продуктивності роботи алгоритму доцільно реалізовувати на основі цілочислових обчислень. Це пояснюється також складністю реалізації в FPGA обчислень для чисел з плаваючою точкою (використання нестандартних бібліотек, зниження швидкодії і т.д.). відповідно, при використанні цілочислових розрахунків втрачається точність, але вдається досягти виграшу в продуктивності апаратної реалізації.

Для застосування даних виразів для цілочислових розрахунків на кожному етапі обчислень виконується відкидання дробової частини виразу. Порівняння точності проведення обчислень для цілих чисел та чисел з плаваючою десятковою точкою на столі для друку з фізичними розмірами, що забезпечує роздільну здатність 30000 точок показало, що даний підхід дозволяє отримати достатньо малу неточність при обчисленнях (рис. 3.14), а значить, здатен забезпечити необхідну точність обчислень для переміщення інструменту.

Перевірка характеристик проводилась для точок координат за віссю абсцис та ординат у інтервалі [0; 30000] і для радіусу 1000 з кроком 100 за різними осями координат. При цьому, значення неточності для усіх розглянутих випадків не перевищує 1.

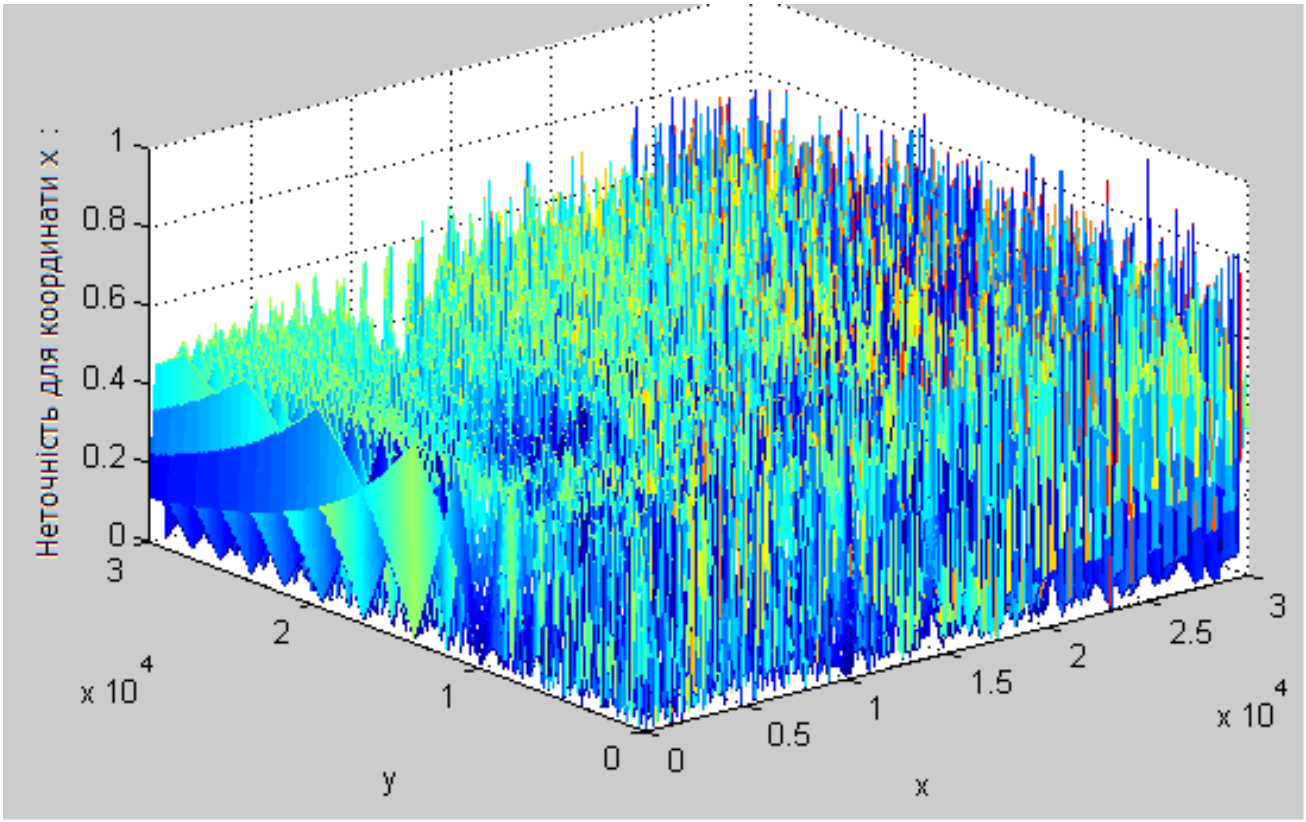


Рис. 3.14. Неточність при розрахунках цілочислових обчислень

Після знаходження центру кола з’являється можливість проведення розрахунків побудови кола за алгоритмом Брезенхема. Однак, це є лише проміжним етапом на шляху до побудови цільового примітиву – дуги (команди G02, G03). Виходячи з того, що модуль побудови кола дозволяє отримати на виході усі точки, які належать колу, то задача полягає в тому, щоб блокувати точки, які не належать дузі. Для визначення приналежності точки дузі відомі координати початкової та кінцевої точок дуги (точки A та B). Позначимо поточну точку кола як P. У такому випадку можна визначити приналежність точки P дузі при обході проти часової стрілки.

При обході в протилежному напрямку достатньо змінити знак нерівності у виразі на протилежний.

Приклад визначення точки за вказаним співвідношенням для випадку переміщення проти часової стрілки приведений на рис. 3.15.

Оскільки окрім передачі даних у мікроконтролер, існує можливість проводити керування кроковим двигуном за допомогою FPGA, то з’являється

необхідність збереження даних інтерполяції в пам’яті пристрою. Розмір блоку пам’яті слід обирати на основі геометричних параметрів простору для друку. У такому випадку для проведення інтерполяції дуги на основі модулю кола кількість точок для збереження складає

$$n_{circle} = 8 \cdot r . \tag{3.22}$$

Таким чином, для паралельного запису значень з модулю побудови кола необхідно розділити на 8 рівних частин, у які записуються координати для 8 октантів. При цьому, слід враховувати, що у випадку, якщо точка знаходиться на діагональній лінії симетрії, то вона при відображенні буде наявна і в списку точок для іншого октанту, що необхідно враховувати при обробці. Для збереження та передачі отриманих значень для керування кроковим двигуном опис переміщення по дузі потребує додаткових вказівників на початок та кінець дуги.

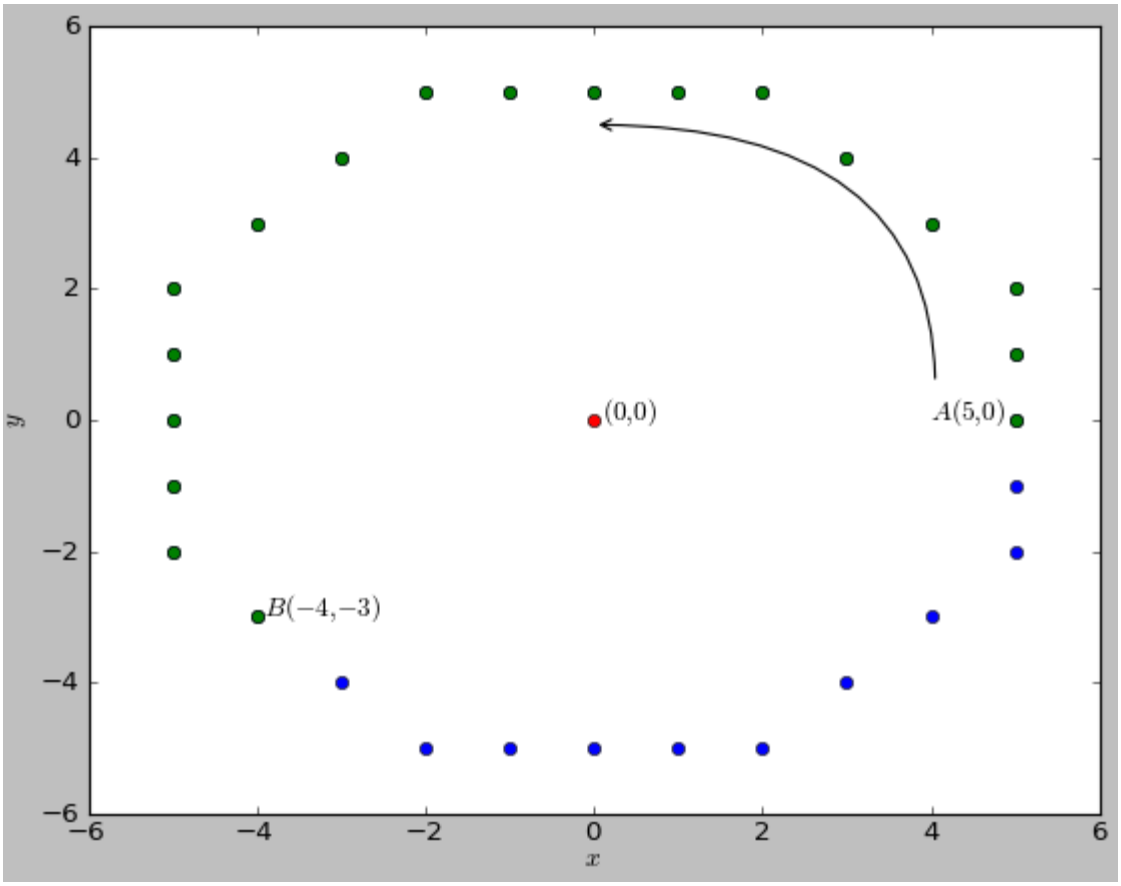


Рис. 3.15 Побудова дуги на основі початкової та кінцевої точок

Для випадку лінії кількість точок, які необхідно зберігати в буфері, буде складати

$$n_{line} = max(x_0 - x_1, y_0 - y_1). \quad (3.23)$$

Для найпростішого випадку заповнення буферу буде відбуватися послідовно. Однак, у зв'язку з тим, що при роботі алгоритму Брезенхема для побудови лінії з'являється можливість використовувати симетричне відображення точок відносно середини відрізку (рис. 3.16), то кількість циклів інтерполяції можна скоротити вдвічі. Внесення змін потребує додавання двох додаткових виводів у модулі для розрахунку відрізку. Тому доцільно організувати додаткові виходи координат симетричного відображення. У такому випадку збереження координат слід проводити від початку буферу в прямому порядку та елемента n_{line} буферу в зворотному.

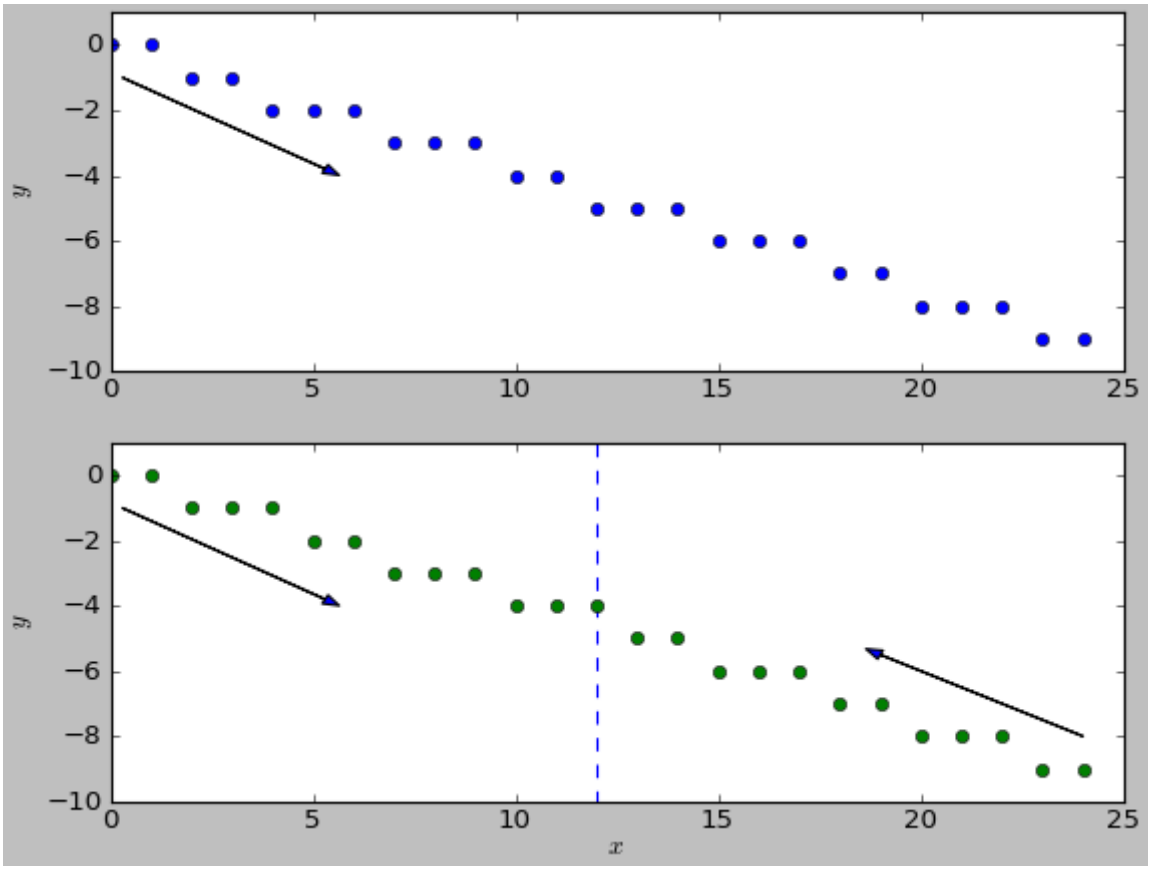


Рис. 3.16. Порівняння результатів побудови лінії за алгоримом Брезенхема з використанням симетричного відображення

Повна реалізація інтерполятору в FPGA передбачає також наявність каналу прийому даних та їх попередньої обробки перед передачею до модуля інтерполяції. Таким чином, необхідно створення прийомного та мультиплексуючого модулів у FPGA, які забезпечать передачу даних у відповідні модулі розрахунку. Від MCU до FPGA за послідовним інтерфейсом надходить рядок, який представляє команду G-код. Модуль приймає дані та виконує їх розбір та передає необхідну інформацію в модулі розрахунку для лінії та кола. Приклад рядка, який подається на вхід модулю: G01 10 20 200 1000.

Для реалізації розбору вхідного рядка необхідно реалізувати скінчений автомат у модулі прийому. Стани автомату, які використовуються при розборі вхідного рядка для отриманні значень параметрів команд представлені у вигляді графу на рис. 3.17.

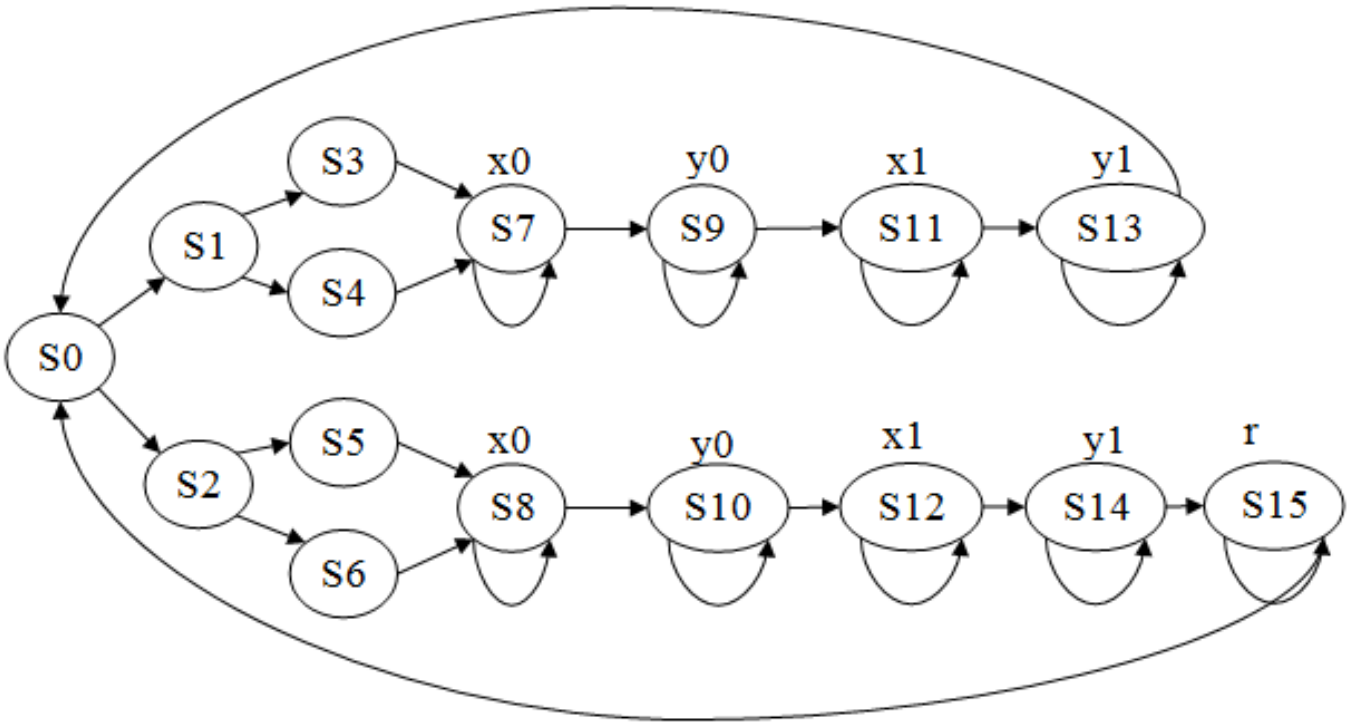


Рис. 3.17. Граф станів автомату розбору рядка G-кодів

На приведеному рисунку наявні стани, для яких можливий перехід у той самий стан. Вони відповідають перетворенню символьного представлення числа в числове значення. Наведені стани можна розділити на дві підгрупи:

1) стани визначення типу команд (S0-S6);
2) стани визначення параметрів команд (S7-S15).

Після виконання розбору дані надходять до мультиплексуючого модулю, схема підключення до інших модулів якого показана на рис. 3.18.

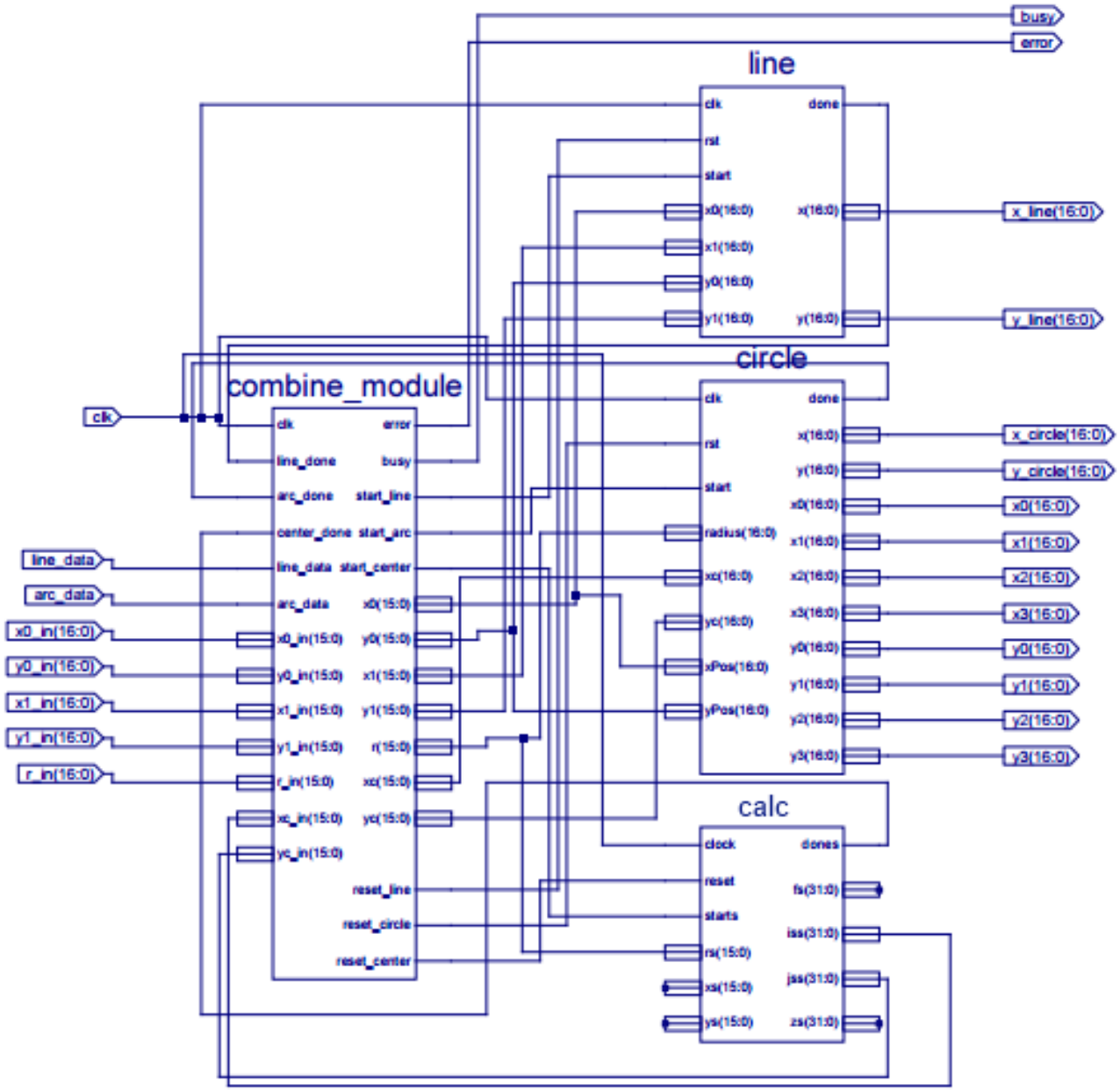


Рис. 3.18. Схема підключення модулів системи

Модулі на рис. 3.18 виконують наступні функції:

– combine_module – прийом даних для розрахунків та мультиплексування між модулями лінії та кола;

– line – модуль розрахунків для побудови лінії;

– circle – модуль розрахунків для побудови кола;

– calc – модуль вибору наступної точки.

Оскільки для побудови кола необхідно попередньо виконати розрахунок координат центру, то модулі, що забезпечують дану функцію доцільно об’єднати в один (рис. 3.19).

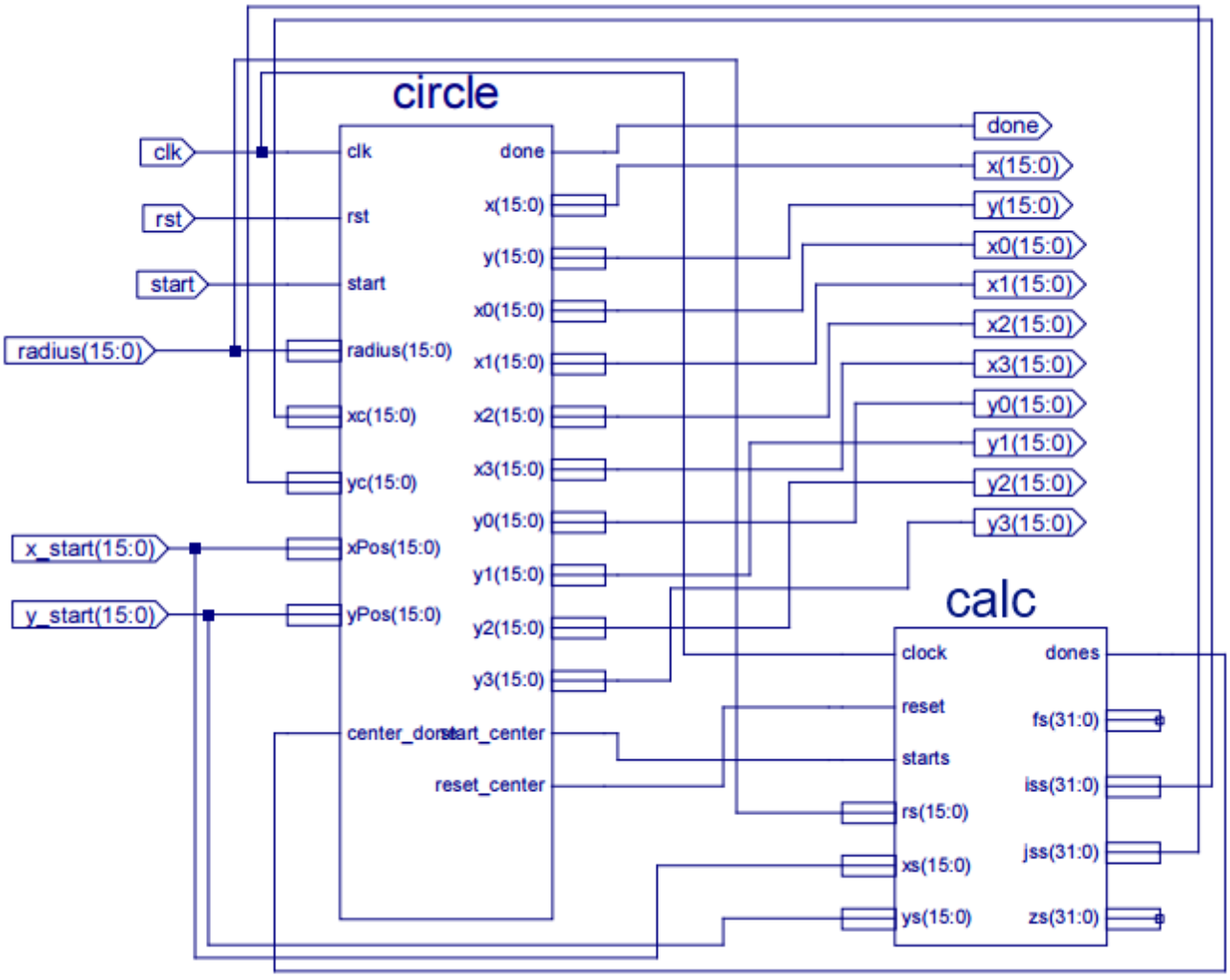


Рис. 3.19. Об’єдання модулів розрахунку центру кола та проведення інтреполяції для кола

Створення структури модулю верхнього рівня та об’єдання розроблених модулів виконано з використанням програмного продукту Xilinx ISE 14.2 WebPack.

Таким чином, запропоновані рішення дозволяють виконати інтерполяцію для дуги на основі швидкого обчислення координат за колом, досягти меншої кількості циклів інтерполяції при виконанні розрахунків для відрізку, об’єднати дані рішення на основі запропонованої структурної організації модулів обчислень та мультиплексування даних, організувати збереження обчислених результатів для подальшого використання.

Висновки до розділу III

1. Розроблені моделі організації подвійного буферу вхідного повідомлення, що дозволяє зменшити час очікування на завантаження повідомлення для декодування. Серед представлених у роботі моделей обрана така, яка забезпечить найкращу пропускну здатність та використання ресурсів та рекомендується для використання.
2. Розроблено метод побудови декодеру на основі особливостей двопортової блокової пам’яті ПЛІС, що дозволяє забезпечувати обробку вхідного повідомлення кількома блоками декодування. Це дозволяє забезпечити підключення одразу двох блоків декодування, а, отже, підвищити пропускну здатність у 2 рази.
3. Проведено дослідження розподілу функцій для гетергенної комп’ютерної системи на прикладі НКГМС, та розглянуто випадок розподілу для системи ПЛІС/МК. Визначено те, як проводити розпаралелювання обчислень для такої системи, а також як розподіляти функції серед модулів НКГМС.
4. Розроблено модулі організації обчислень на базі ПЛІС для алгоритмів Брезенхема. Вони працюють на основі цілочислових вхідних даних

та забезпечують більшу швидкість обробки даних для побудови геометричних примітивів у порівнянні з мікроконтролером.

РОЗДІЛ IV
АПАРАТНА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ РОЗРОБЛЕНИХ ПОЛОЖЕНЬ

Розроблені в роботі моделі та методи побудови LDPC-декодерів мають бути реалізовані практично та пройти перевірку на основі результуючих даних декодування. Доцільно проводити перевірку програмної та апаратної реалізацій. Окрім того, необхідно визначити, як саме буде працювати декодер при різних значеннях співвідношення сигнал/шум.

4.1 Реалізація апаратно-програмного комплексу для проведення тестування розроблених положень

Для експериментальної перевірки розроблених у роботі положень необхідно:

1) генерація вхідних даних для проведення тестування;
2) виконати програмну реалізацію розроблених моделей та методів;
3) розробити схему взаємодії програмного забезпечення та апаратної реалізації LDPC-декодеру;
4) розробити схемотехнічний опис LDPC-декодеру для реалізації в ПЛІС;
5) провести моделювання роботи декодеру;
6) розробити програмне забезпечення для обміну даними з апаратним декодером;
7) провести тестування роботи декодеру та перевірити результати декодування для програмної та апаратної складових комплексу для проведення тестування.

Очікуваним результатом проведення тестування за допомогою апаратно-програмного комплексу є те, що кінцеві дані декодування мають співпадати для програмної та апаратної складової. Крім того, апаратна реалізація декодеру дозволить отримати дані про використання основних

ресурсів мікросхеми ПЛІС та визначити, мікросхеми якого цінового діапазону необхідні для реалізації такого декодеру.

4.1.1 Генерація вхідних даних для тестування

Вхідними даними, які необхідні для проведення тестування, є:

1) МПП з випадковим розташуванням значущих елементів;
2) вхідне слово (набір слів).

Для отримання МПП було використане відкрите програмне забезпечення [145], яке розроблено відомими дослідниками LDPC-кодів. Програмне забезпечення може використовуватись у операційній системі сімейства Linux або у Windows за умови встановлення пакетів MinGW [85] або Cygwin [86]. Програмні засоби для роботи з LDPC-кодами реалізовані у вигляді набору утиліт командного рядка. Для генерації матриці використана наступна команда:

```
./make-pchk 500x1000.pchk 500 1000.
```

Згенерована МПП представлена на рис. 4.1.

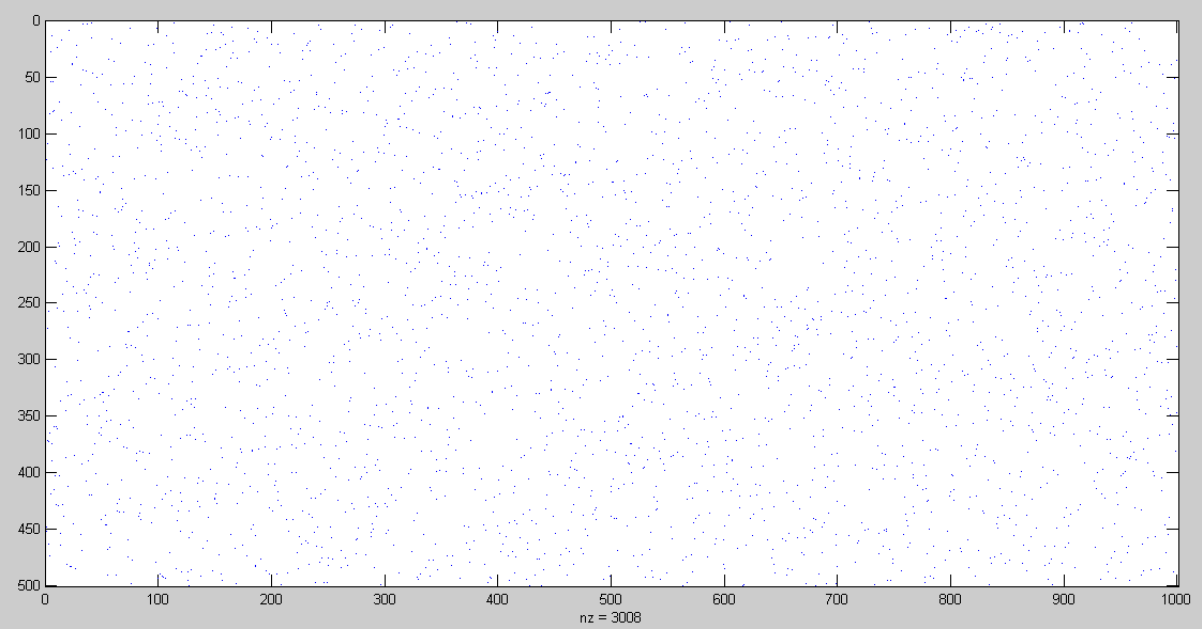


Рис. 4.1. МПП для реалізації апаратно-програмного комплексу

Отримана матриця має розмірність M =500 та N =1000. Максимальна вага рядку в матриці становить $w_{r\,max}$=8, а мінімальна – 5. Критеріями для вибору матриці були випадкове розташування значущих елементів та розмірність

МПП, яка б не перевищувала б ресурси доступної ПЛІС по використанню пам’яті.

4.1.2 Розробка програмного забезпечення для програмної перевірки LDPC-декодування

Реалізація програмного декодеру виконувалась на мові програмування Java [146, 147]та з використанням програмного середовища NetBeans 7.3.1. Діаграма класів [148] розробленого програмного забезпечення з розділенням на пакети представлена на рис. 4.2 (з метою зменшення перевантаженості діаграми візуальними елементами деякі зв’язки між класами опущені).

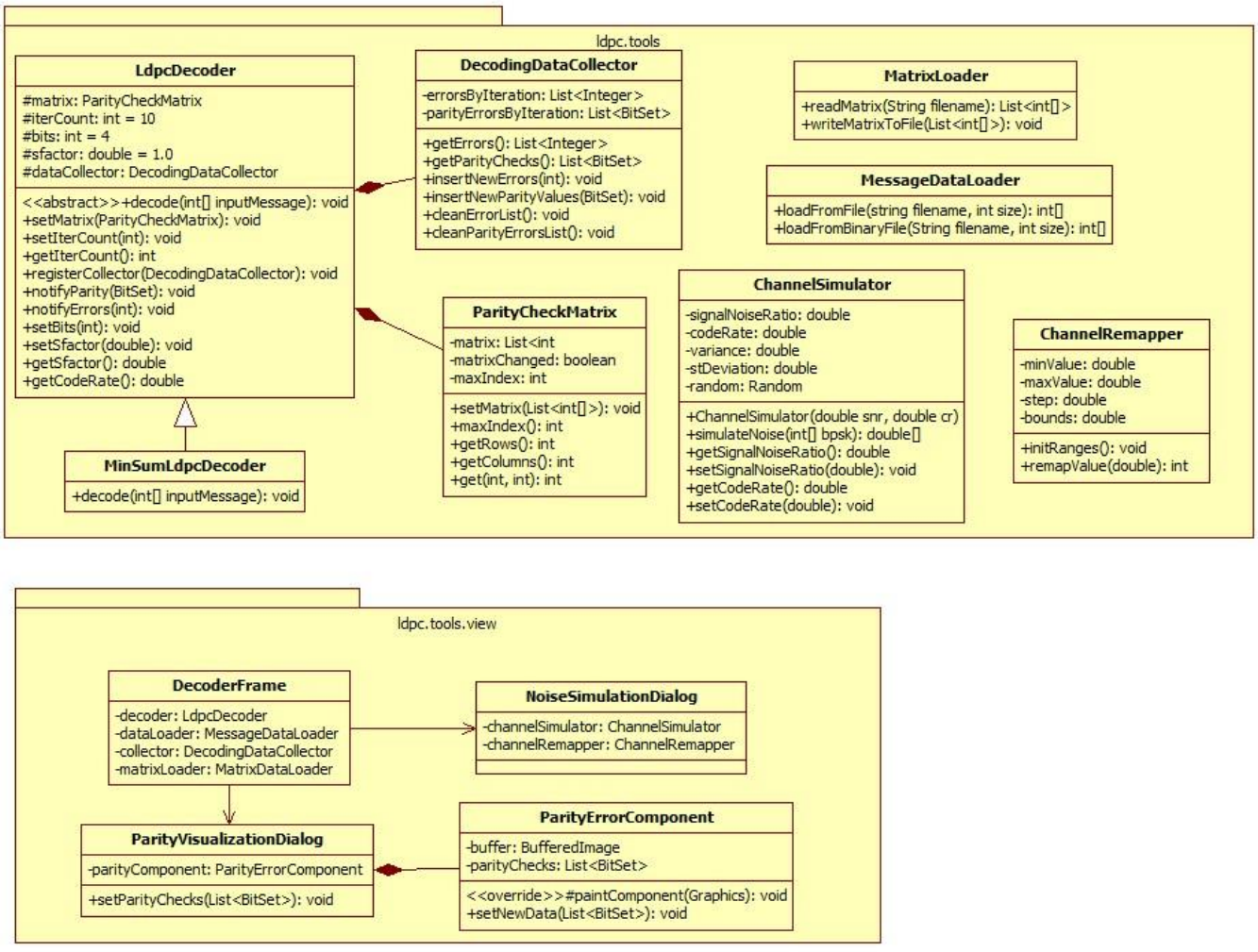


Рис. 4.2. Діаграма класів розробленого програмного забезпечення

Для створення діаграми класів використано програмний продукт для побудови UML-діаграм StarUML 5.0.2. Дане програмне забезпечення може використовуватись безкоштовно.

Для створення інтерфейсу користувача використана бібліотека Swing та бібліотека для побудови графіків JFreeChart [42]. Головне вікно розробленого програмного забезпечення наведено на рис. 4.3.

Рис. 4.3. Головне вікно програмного декодеру

При роботі з програмою користувач отримає можливість дослідити процес декодування за кількістю помилок парності після кожної ітерації. Головна

форма призначена для дослідження декодування одного пакету вхідних даних. Для цього користувач задає наступні вхідні дані:

1) текстовий файл з індексами МПП – на його основі формується матриця;

2) текстовий файл з вхідним повідомленням;

3) кількість бітів доступних для квантизації вхідних даних;

4) максимальну кількість ітерацій декодування.

Ввівши усі необхідні дані, користувач натискає на кнопку початку декодування та отримує дані про кількість помилок для кожної ітерації. Крім того, для візуального представлення розташування помилок парності розроблено додатковий компонент, який розміщується в окремому діалоговому вікні (рис. 4.4).

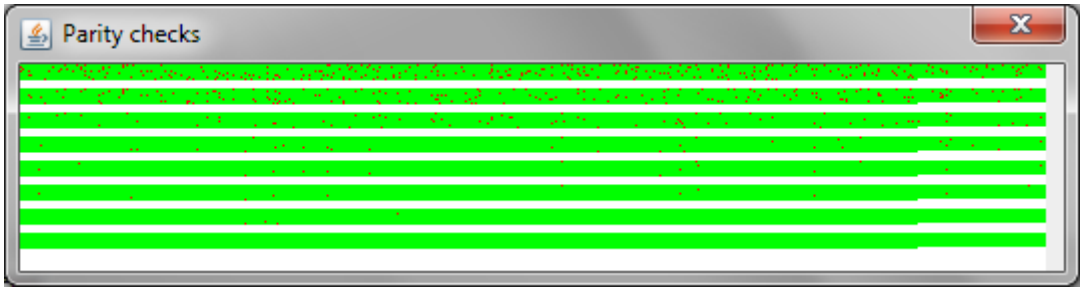


Рис. 4.4. Діалогове вікно візуалізації помилок перевірки парності

Даний компонент надає можливість візуально оцінити те, наскільки швидко виправляються помилки парності.

4.1.3 Розробка структурної схеми взаємодії програмного забезпечення та апаратної реалізації LDPC-декодеру

Для створення тестового комплексу використано наступні компоненти:

– ноутбук Toshiba Satellite L500;

– плата USB-мосту FTDI FT2232H Mini Module;

– відлагоджувальна плата ZedBoard, до складу якої входить SoC Zynq-7020 [149-152].

Мікросхема SoC має у своєму складі ядро мікропроцесору ARM Cortex A9 та програмовну логіку Xilinx Artix 7. Для створення комплексу

використовується лише програмовна логіка. Загальна структурна схема взаємодії компонентів наведена на рис. 4.5.

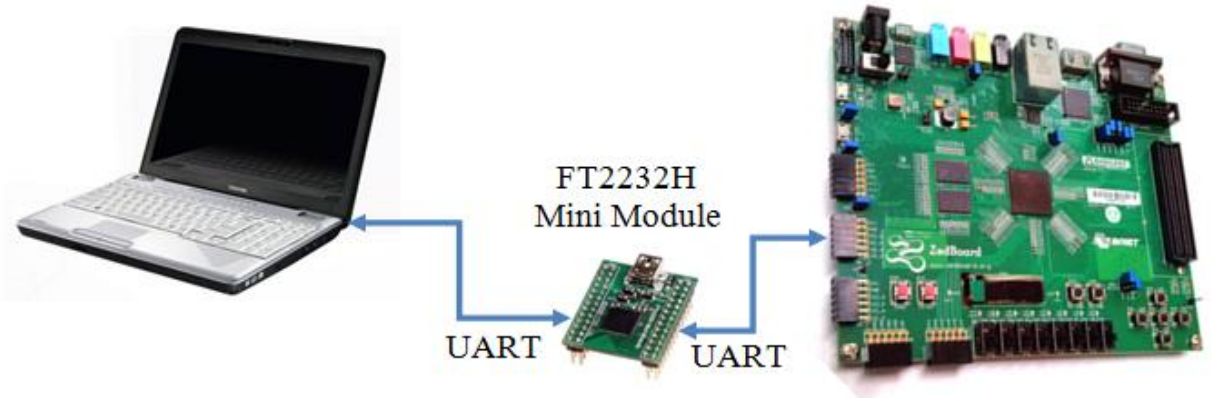


Рис. 4.5. Загальна структурна схема взаємодії компонентів

Інтерфейсом для передачі даних між ноутбуком та відлагоджувальною платою обрано UART. Проміжним компонентом для обміну інформацією між ПК та платою є USB-міст. Таке рішення означає те, що на платі також має бути реалізований модуль прийому/передачі даних за допомогою UART. Зі сторони ноутбуку він підключається за допомогою інтерфейсу USB, а зі сторони плати – за допомогою слотів розширення PMOD. З програмної точки зору, це означає, що для передачі даних використовуватиметься віртуальний COM-порт.

Для запуску стенду необхідно забезпечити наявність модулів для ноутбуку та відлагоджувальної плати таким чином, як це показано на діаграмі розгортання (рис. 4.6). Для створення діаграми використано програмне забезпечення Visual Paradigm Community Edition 12.1.

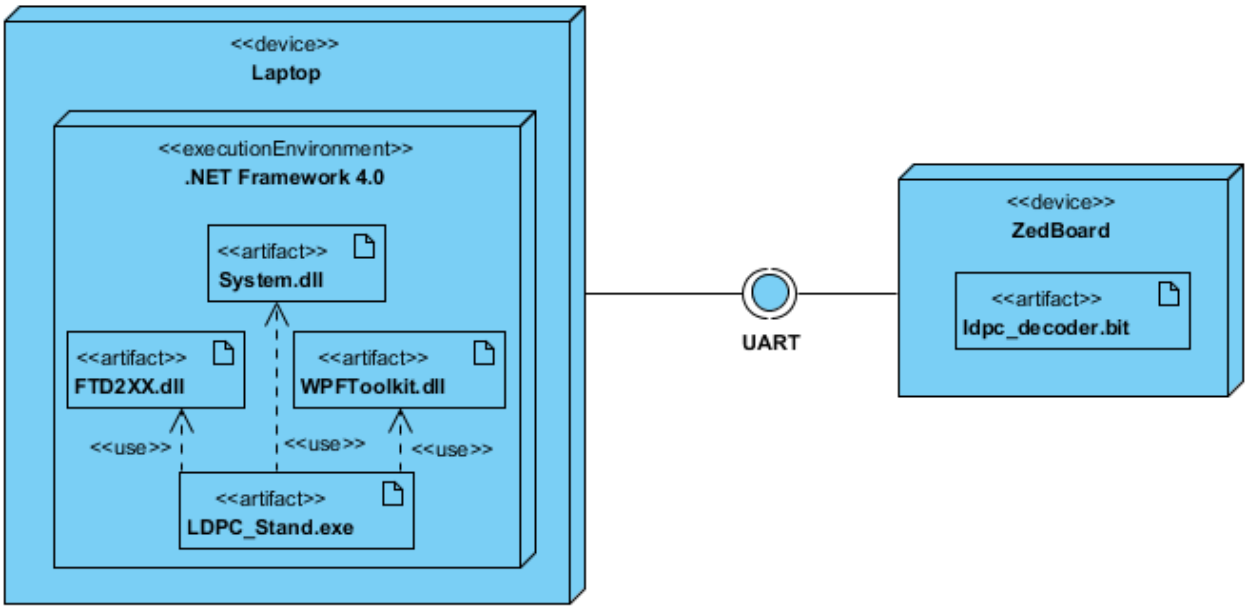


Рис. 4.6. Діаграма розгортання компонентів комплексу

Окрім стандартних бібліотек, що входять до складу .NET Framework 4.0, необхідна також наявність додаткових бібліотек.

4.1.4 Розробка схемотехнічного опису LDPC-декодеру

Для створення схемотехнічного опису LDPC-декодеру з використанням мови VHDL [153-155] використано середовище Xilinx PlanAhead 14.7 WebPack. Воно надає можливість централізувати розробку систем, що використовують як процесорні системи, так і програмовну логіку.

При розробці схемотехнічного опису LDPC-декодеру виконано опис на основі організації скінченних автоматів [156-158]. Використано трьохпроцесний спосіб виконання опису [53]. На рис. 4.7 представлено граф станів, на основі якого розроблено головний модуль обробки.

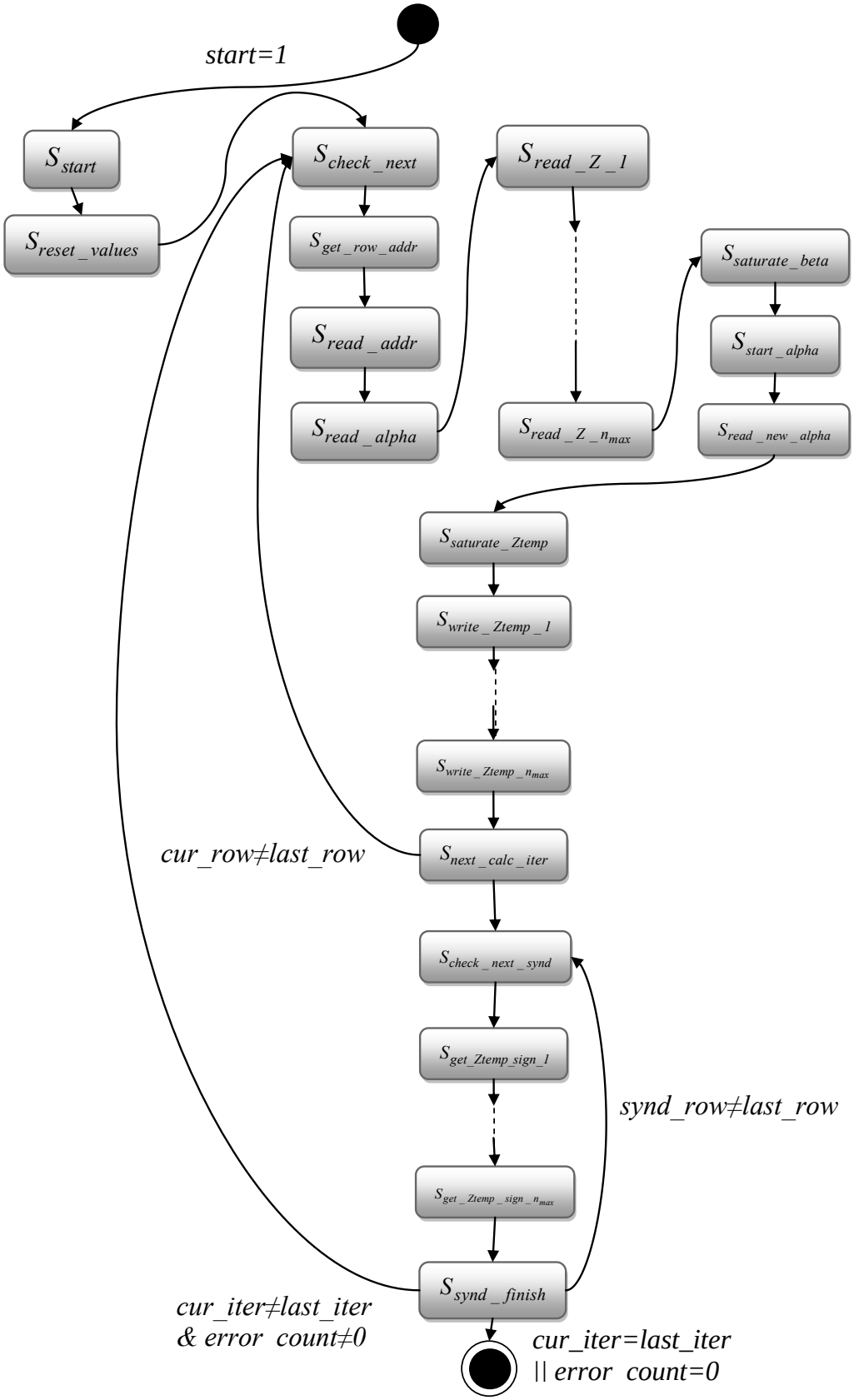


Рис. 4.7. Граф станів основного модулю декодеру

Дії модуля, що виконуються при знаходженні у певному стані, наведені в табл. 4.1.

Таблиця 4.1

Опис дій для графу станів основного модуля декодеру

Назва стану	Опис дій
S_{start}	Початок ітерації декодування
S_{reset_values}	Встановлення початкових значень (обнулення) основних змінних результатів ітерації
S_{check_next}	Перевірка, чи виконана обробка усіх рядків
$S_{get\_row\_addr}$	Встановлення значення номеру поточного рядка в якості адреси для зчитування з пам'яті адрес
S_{read_addr}	Зчитування значень адрес
S_{read_alpha}	Зчитування значень повідомлень вузлів перевірки
$S_{read\_Z\_1},\ldots,S_{read\_Z\_n_{max}}$	Встановлення адреси зчитування з пам'яті Z; зчитування з пам'яті Z; обчислення β з розширенням розрядності
$S_{saturate\_beta}$	Виконання зменшення розрядності значення β з урахуванням його розширеного значення
S_{start_alpha}	Початок обчислень значень α; передача β у модуль обчислень
$S_{read\_new\_alpha}$	Зчитування значень α з модулю обробки
S_{update_alpha}	Оновлення (запис у пам'ять) α для поточного рядка; обчислення розширеного значення Z_{temp}
$S_{saturate\_Ztemp}$	Виконання зменшення розрядності значення Z_{temp} з урахуванням його розширеного значення
$S_{write\_Ztemp\_1},\ldots,S_{write\_Ztemp\_n_{max}}$	Запис нового значення Z_{temp}
$S_{next\_calc\_iter}$	Перевірка переходу до наступної ітерації обчислення для нового рядка або до перевірки синдрому
$S_{check\_next\_synd}$	Перевірка наявності рядків для обчислення синдрому
$S_{get\_Ztemp\_sign\_1},\ldots,S_{get\_Ztemp\_sign\_n_{max}}$	Зчитування значень Z_{temp} та отримання на їх основі значень для перевірки парності; перезапис даних з Z_{temp} до Z та в пам'ять результатів
S_{synd_finish}	Перевірка завершення процесу декодування для поточного повідомлення

Логіка управління для основного керуючого модуля також реалізована за допомогою скінчених автоматів. Вони використовуються для:

1) прийому даних та подачі сигналу запуску процесу декодування для основного модуля;

2) видачі результатів декодування.

Графи станів для цих скінчених автоматів наведені на рис. 4.8 та 4.9.

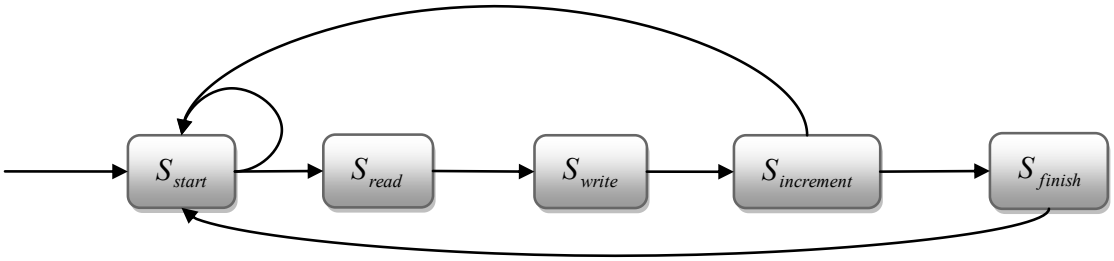


Рис. 4.8. Граф станів для процесу прийому даних

Граф станів автомату видачі результатів містить більшу кількість станів через необхідність очікування сигналу готовності.

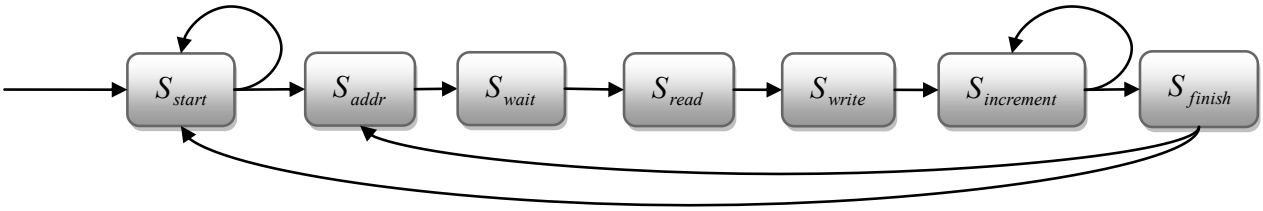


Рис. 4.9. Граф станів для процесу видачі результатів

Проведені стадії синтезу та реалізації (трасування) та генерації бітового файлу для програмування за допомогою середовища розробки. Результати, щодо використання основних ресурсів програмовної логіки наведено на рис. 4.10.

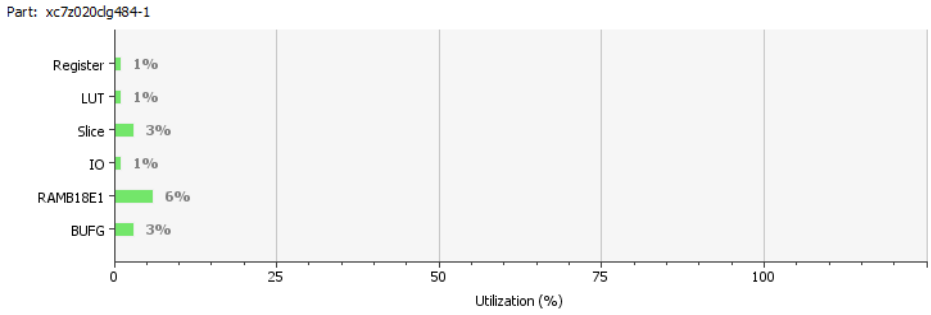


Рис. 4.10. Використання основних ресурсів ПЛІС для реалізації

За допомогою інструменту Schematic у середовищі розробки отримана наступна схема підключення основних модулів декодеру (рис. 4.11).

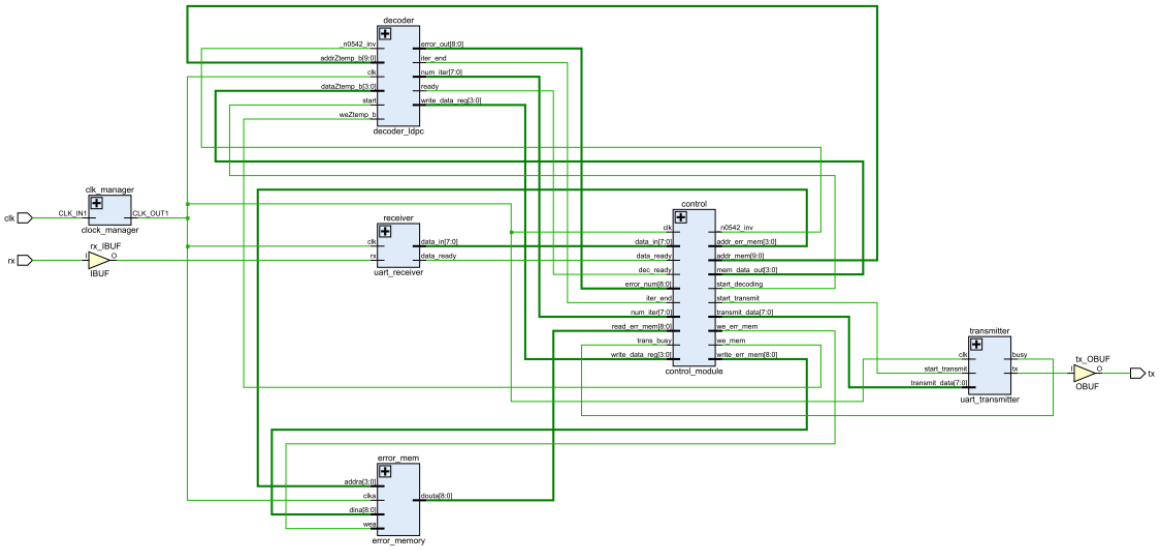


Рис. 4.11. Підключення компонентів декодеру

Опис підключення основних складових декодеру виконано з використанням структурної методології опису.

Інструменти, які відповідальні за перевірку тактових частот, на яких можлива робота декодеру показують, що робоча тактова частота перевищує 200 МГц (рис. 4.12).

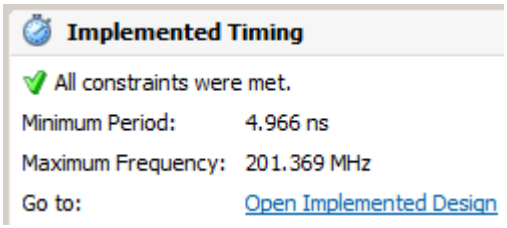


Рис. 4.12. Результат щодо можливого використання тактової частоти

Для генерації такого значення частоти (до програмовної логіки підключений кварц, який генерує частоту 100 МГц) використовується додатковий модуль управління тактовою частотою MMCM. Він дозволяє підвищити тактову частоту до 200 МГц.

Отримане значення частоти може бути підвищено за допомогою оптимізації опису.

4.1.5 Моделювання роботи декодеру

Перед перевіркою роботи декодеру в складі пристрою проведена моделювання на рівні опису. Для моделювання використовувався пакет Modelsim 10.1d Altera Starter Edition,

Розроблене тестове оточення (testbench) [159], яке передбачає:

1) запис вхідних даних у пам’ять декодеру;

2) початок декодування по завершенню запису даних;

3) перегляд процесу декодування;

4) видачу результатів декодування, у т.ч. кількість помилок перевірок парності.

Проведене моделювання для різних пакетів вхідних даних. Значення основних сигналів у ході процесу декодування наведені на рис. 4.13.

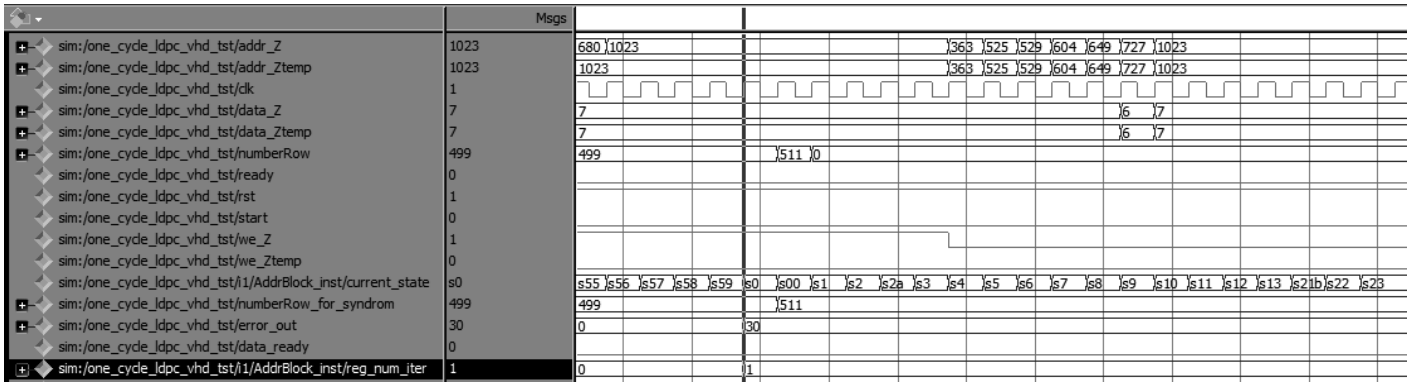


Рис. 4.13. Знімок екрану моделювання роботи декодеру

На рис. 4.13 виведені сигнали, що позначають адреси даних для обробки, значення даних, поточний стан декодеру, кількість помилок парності за результом ітерації, номер поточної ітерації.

Окрім візуальної перевірки отриманих результатів та перевірки на основі кількості помилок перевірки парності, у ході декодування виконувався запис у файл додаткової інформації про стан усіх регістрів. Дані файли проходять

перевірку на ідентичність їх вмісту з файлами, які генерує програмна реалізація декодеру.

Інші приклади моделювання роботи декодеру наведені у Додатку Б.

4.1.6 Розробка програмного забезпечення для обміну даними з LDPC-декодером

Для створення програмного забезпечення, яке забезпечуватиме обмін інформацією між компонентами комплексу, використано середовище розробки Visual Studio 2012 Express for Desktop та мова програмування C# [160]. Програмна платформа для створення ПЗ – .NET Framework 4.0. У якості системи для побудови користувацького інтерфейсу обрано Windows Presentation Foundation, а для відображення графіків – WPF Toolkit 3.5 [161, 162]. Інтерфейс користувача програми для обміну даними наведено на рис. 4.14.

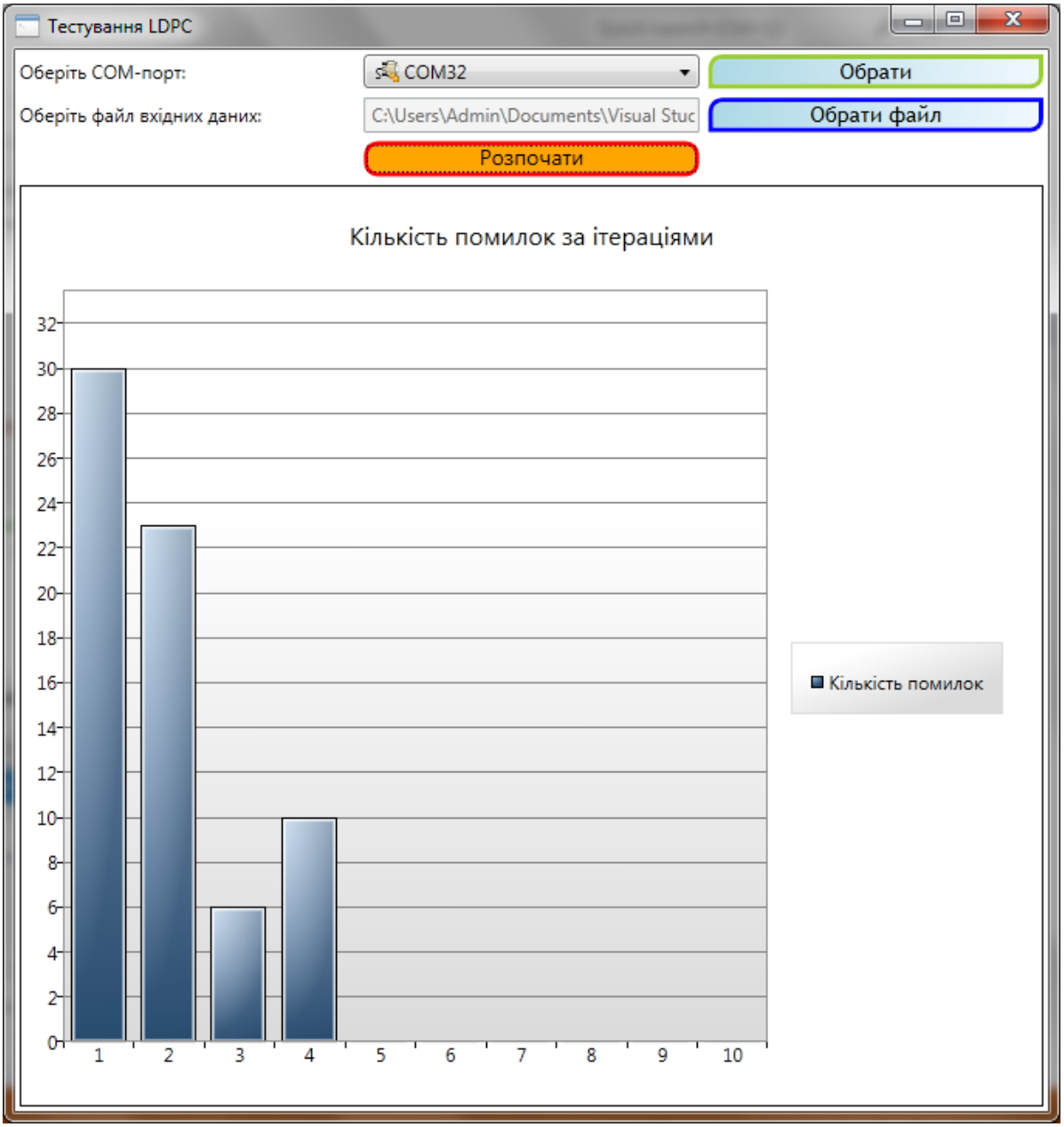


Рис. 4.14. Вікно програми обміну даними з декодером

Користувач має можливість обрати COM-порт, за допомогою якого відбуватиметься обмін даними, та файл з вхідними даними.

4.1.7 Тестування LDPC-декодеру у складі програмно-апаратного комплексу

Для представлення процесу роботи програмно-апаратного комплексу розроблена діаграма активностей, на якій наведені основні етапи роботи з його складовими та послідовність виконання цих етапів (рис. 4.15). Передбачається,

що усі необхідні компоненти розміщені на відповідних пристроях відповідно до діаграми розгортання на рис. 4.6.

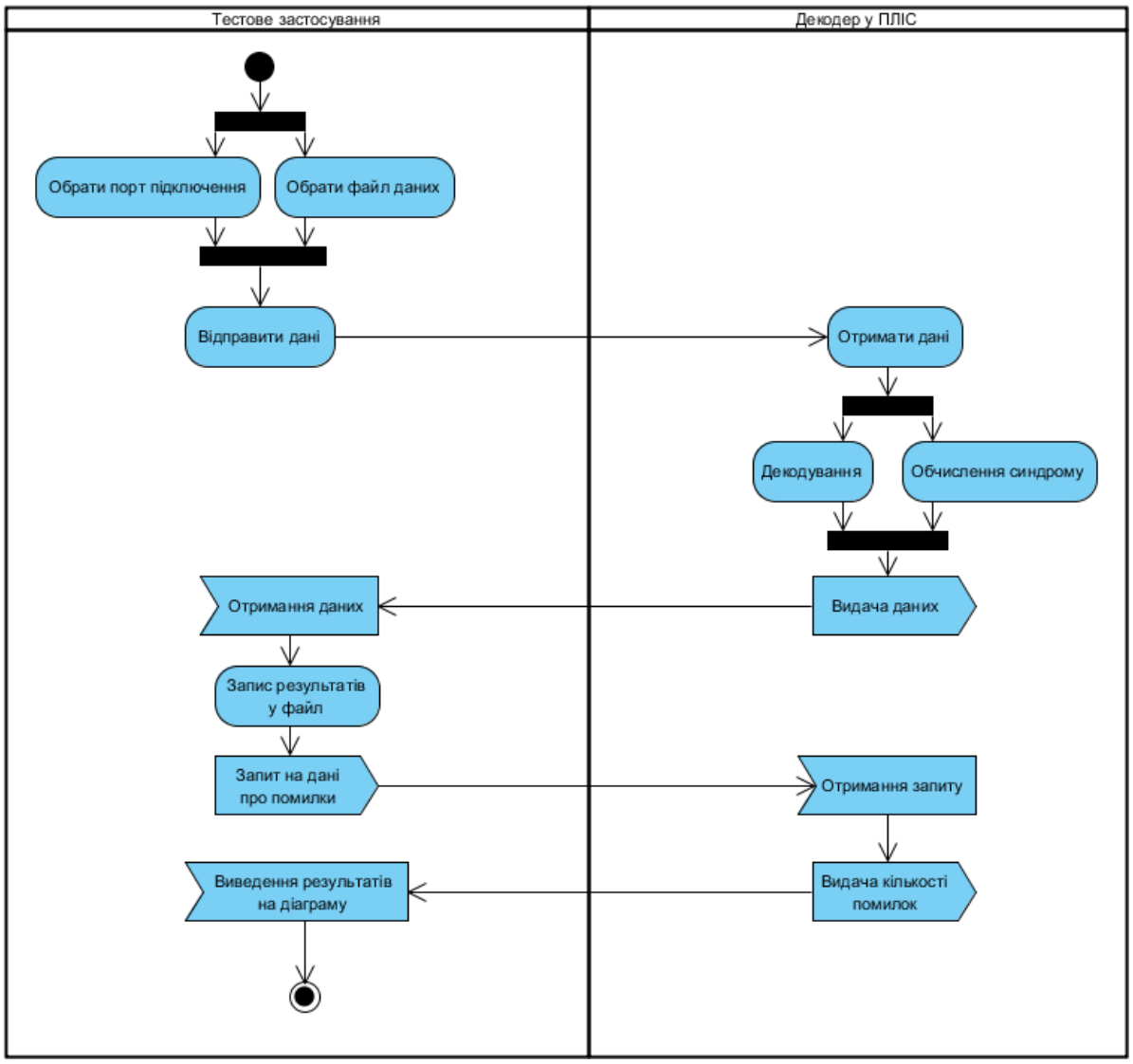


Рис. 4.15. Діаграма активності роботи з комплексом

Окрім виведення результатів на діаграму, виконується також запис отриманих результатів у файл. На основі цих даних та результатів роботи програмного декодеру визначається коректність роботи декодеру на базі ПЛІС. Порівняння проводиться за допомогою програми, що може порівнювати двійкові файли (.bin), наприклад, HxD (рис. 4.16).

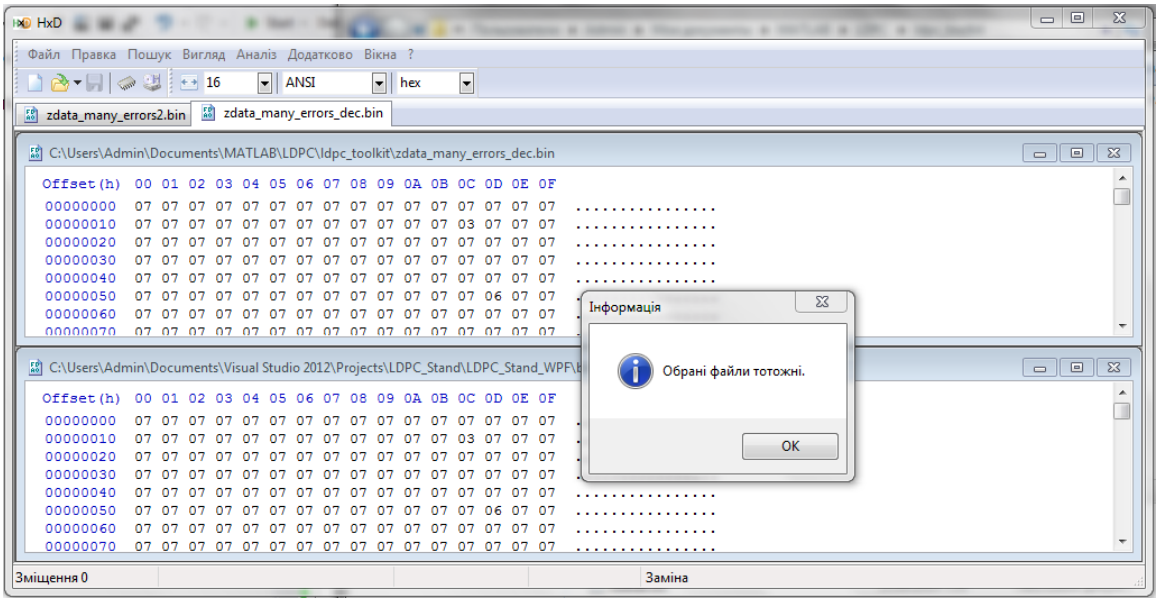


Рис. 4.16. Порівняння файлів у програмі HxD

Тестування проводилось на основі вектора вхідних даних, заповненого нулями, для деяких бітів якого проведена інверсія, що спричиняє наявність помилки парності. Результуючий вектор так само має містити лише нульові значення.

Підключення пристроїв (відлагоджувальної плати та USB-мосту) під час тестування комплексу наведено на рис. 4.17.

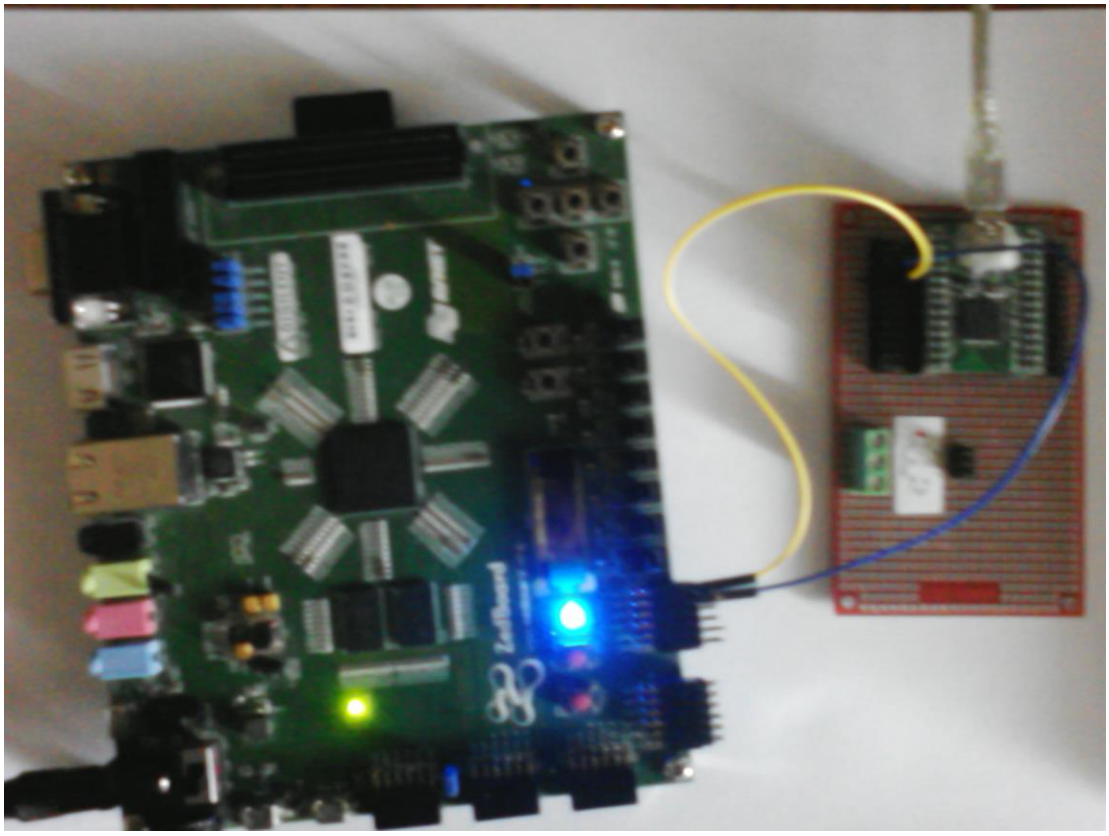


Рис. 4.17. Перевірка роботи декодеру

Для коректної роботи комплексу підключення компонентів необхідно виконати до початку програмування ПЛІС. Інакше, підключення коли декодер очікує на вхідні дані внаслідок джиттеру контактів може бути сприйнято декодером як початок передачі даних та некоректних результатів декодування.

4.2 Реалізація LDPC-декодерів на основі розроблених положень

У даному розділі наведені практичні результати реалізації LDPC-декодерів на основі розроблених теоретичних положень.

Окрім окремих МПП, наведених у подальших підпунктах, пропускна здатність декодерів перевірялась на наборі з 25 матриць (табл. 4.2). Візуалізація деяких МПП наведена на рисунках у доадатку В.

Таблиця 4.2

Характеристики тестових МПП та пропускна здатність розроблених декодерів

№ з/п	Кількість стовпців	Кількість рядків	Час обчислення 1000000 ітерацій, с (тактова частота – 240МГц)	
			декодер з одним блоком декодування	конвеєрний декодер
1	16416	8208	693	22
2	16416	5472	693	22
3	16512	8256	647,4	16,6
4	16512	5504	647,4	16,6
5	16512	4128	647,4	16,6
6	16416	8208	801,6	33,4
7	16416	16416	801,6	33,4
8	8368	2092	210,375	4,25
9	8368	2092	237,15	4,25
10	8368	2092	255	5,1
11	8368	2092	293,25	6,8
12	8368	2092	346,5	8,5
13	6276	2092	198,9	11,55
14	6276	2092	217,35	6,9
15	6276	2092	247,95	8,7
16	6276	2092	266,475	10,45
17	4184	2092	144	3
18	4184	2092	183,3	4,7
19	4184	2092	180	6
20	4184	2092	184,8	7,7
21	2092	2092	102	4,25
22	1046	523	45,675	1,05
23	1046	523	48,3	1,4
24	1046	523	53,55	2,1
25	1046	1046	53,55	2,1

Як видно з табл. 4.2, конвеєрна версія декодеру забезпечує виграш у порівнянні з версіює з одним блоком декодування в діапазоні 20-50 разів.

4.2.1 Розробка LDPC-декодеру зі зменшеним використанням блокової пам’яті

У ході дослідження для перевірки реалізації LDPC декодеру використана нерегулярна матриця з низькою щільністю значимих елементів розмірністю

(5000, 10000). Для даної матриці $w_{r\max} = 9$, а $w_{r\min} = 5$. Візуальне представлення значущих елементів матриці представлено на рис. 4.18.

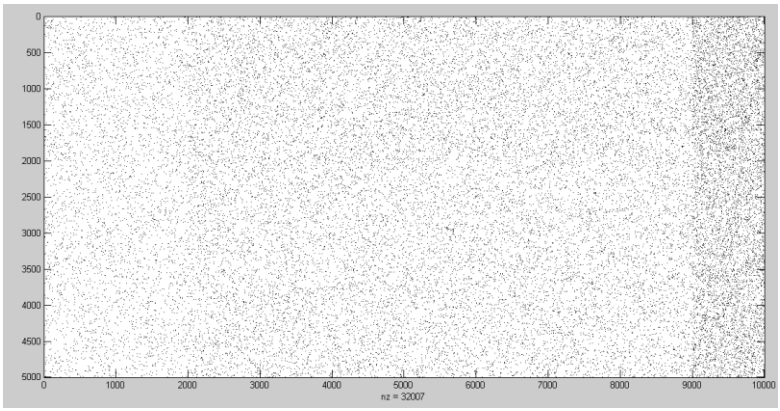


Рис. 4.18. Використана МПП

Кількість одиниць у стовпцях матриці також не є однаковою. Характер розподілу кількості значущих елементів відповідно до стовпців наведено на рис. 4.19.

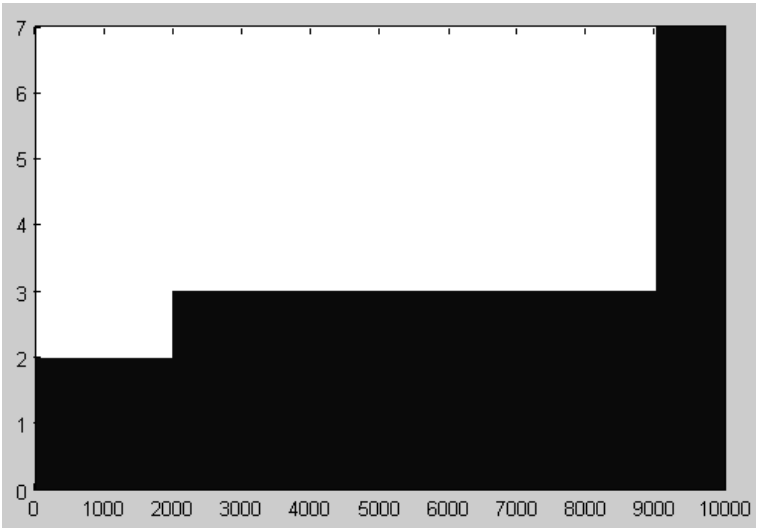


Рис. 4.19. Кількість значущих елементів за стовпцями у МПП

З точки зору розпаралелювання виконання обчислень важливим є також розподіл індексів значущих елементів по позиціях у рядках. У тому випадку, коли діапазони значень індексів елементів не перетинаються, пам’ять можна організувати таким чином, щоб запис та зчитування даних виконувалися паралельно. Результати розподілу для обраної матриці наведені в табл. 4.3.

Таблиця 4.3

Розподіл індексів для обраної матриці

№ вузла перевірки	Мінімальне значення індексу	Максимальне значення індексу
1	0	8234
2	49	9433
3	375	9592
4	812	9974
5	1207	9972
6	2669	9999
7	4748	9999
8	5355	9999
9	9651	9651

Для обраної матриці на основі запропонованого методу побудови реалізовано LDPC-декодер. Для реалізації використовувались мова схемотехнічного опису VHDL та середовище розробки Altera Quartus II 13.1. Відповідно до запропонованого методу, використовується лише один тип блоку обробки повідомлень вузлів. Максимальна тактова частота, на якій може працювати модуль становить вище 200 МГц для мікросхеми Altera Stratix IV. З урахуванням того, що організації пам’яті використовувалась блочна пам’ять M9K, розмірністю 512x16 біт, зменшити використання даного ресурсу вдалося на 254 таких блоків пам’яті. Кількість обробок для кожної ітерації становить 5000.

4.2.2 Реалізація LDPC-декодеру на основі моделі організації паралельних обчислень

Для проведення практичного порівняння моделей реалізовано два LDPC-декодери. Реалізація виконана на мові схемотехнічного опису VHDL у середовищі Xilinx ISE Design Suite WebPack 14.7 для мікросхеми System-on-Chip (SoC) сімейства Zynq-7020. Робоча тактова частота обох декодерів

становила 250 МГц. Довжина вхідного повідомлення для декодування L становила 1000, кількість циклів для однієї ітерації I – 500, а параметри m і s дорівнювали 43 та 17 відповідно. Максимальна кількість ітерацій обмежена для послідовної моделі 10, а для паралельної – 11 ітераціями. Отримані результати швидкодії наведені на рис. 4.20.

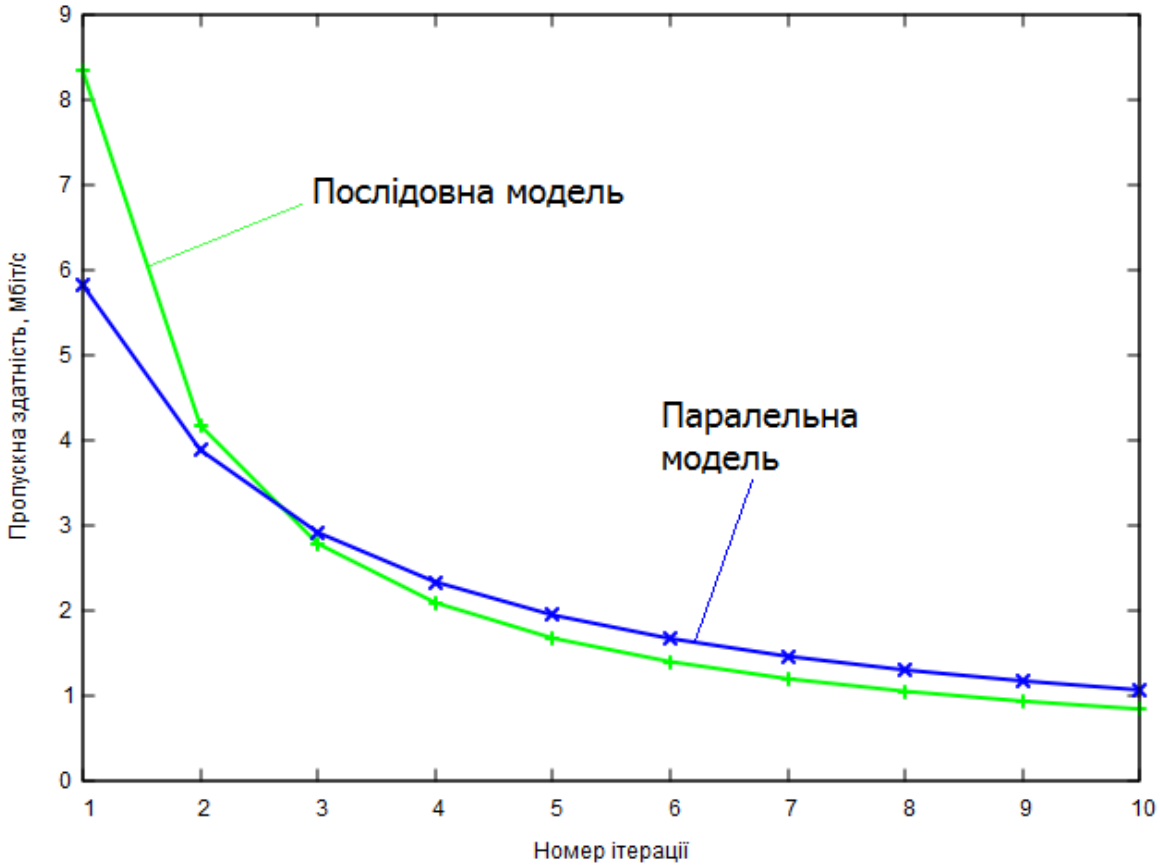


Рис. 4.20. Порівняння послідовної та паралельної моделей

Слід зазначити, що для порівняння перша ітерація для паралельного декодеру не використовувалась, оскільки для неї немає відповідного номеру для послідовного декодеру. Видача результату на першій операції для паралельного декодеру означатиме, що перевірка синдрому не виявила помилок парності та декодоване повідомлення доступне для зчитування. Значення пропускної здатності в такому випадку становить 11,63 Мбіт/с. Графік параметру E для розроблених декодерів наведено на рис. 4.21.

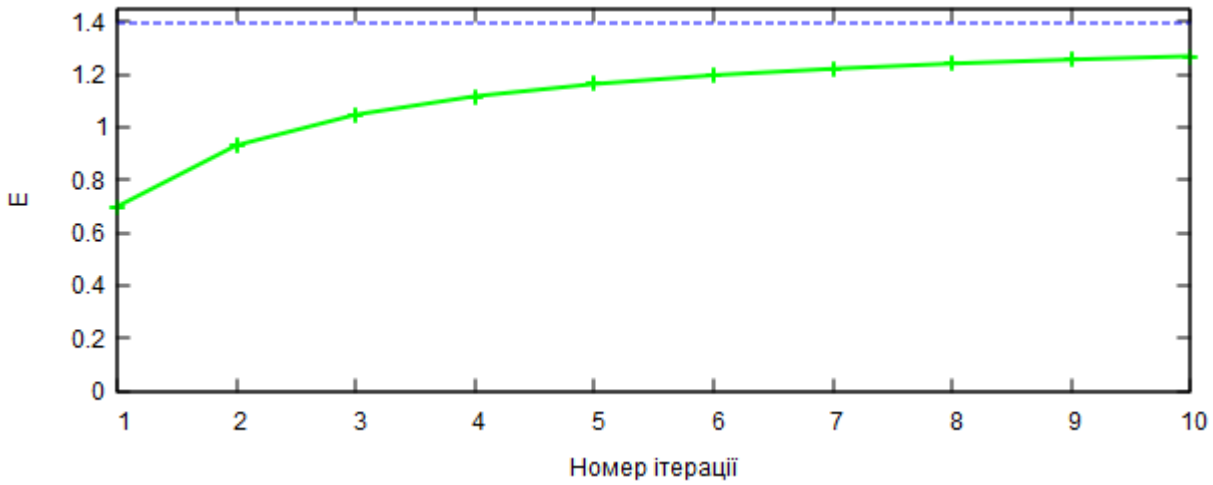


Рис. 4.21. Виграш у швидкодії для паралельної моделі відносно послідовної

Граничне значення E для даного випадку становить 1,39. Таким чином, зі збільшенням кількості ітерацій, що може бути пов'язано, наприклад, з підвищенням рівня шуму в каналі, відносна пропускна здатність для запропонованої паралельної моделі буде збільшуватись.

Крім того, реалізована модель LDPC-декодеру з послідовним обчисленням значення α_{ij} при зчитуванні даних. Реалізація дозволила зменшити значення параметру m до 38, у порівнянні з 43 для паралельної організації обчислень. Циклограма розробленого модулю з прикладом роботи представлена на рис. 4.22.

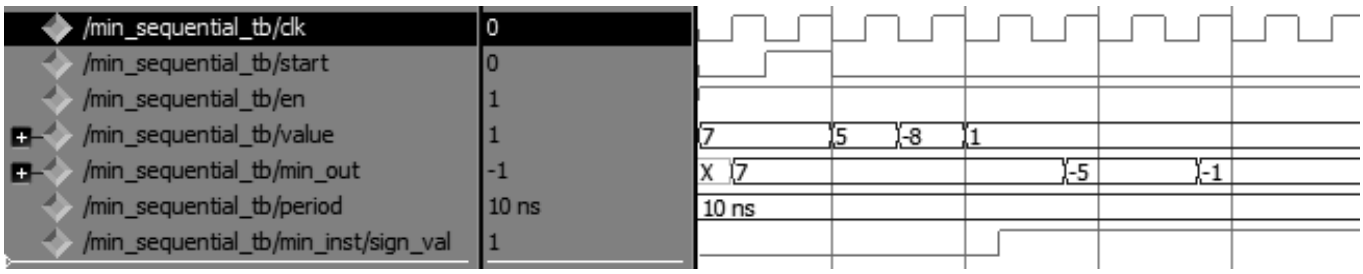


Рис. 4.22. Моделювання роботи модулю послідовного обчислення

Для прикладу, представленого на рис. 4.22, тестування проводилося з 4-бітними даними.

Варто також зауважити, що реалізація представленої моделі для ПЛІС, потребуватиме виділення додаткового блоку пам'яті для збереження результатів ітерації $Memory_Z$. Це пов'язано з тим, що в ході запису оновлених проміжних значень необхідне виконання перезапису з пам'яті $Memory_{Ztemp}$ до пам'яті $Memory_Z$. Проте, до кінця обчислення для поточної ітерації вміст даної пам'яті не має змінюватися, оскільки це змінить результати перевірки синдрому. Саме тому, зчитування відбувається з одного блоку пам'яті, а запис вже в інший. Для кожної наступної ітерації функції даних блоків почергово змінюються. Для реалізованих декодерів різниця у використанні блоків пам'яті виявилася незначною: знадобився один додатковий блок пам'яті RAM18 на 18 кбіт. Порівняння використання основних ресурсів для обох реалізацій наведено в табл. 4.4.

Таблиця 4.4

Порівняння паралельної та послідовної реалізації

Назва ресурсу	Паралельна реалізація		Послідовна реалізація	
	Використано, од.	Використано, %	Використано, од.	Використано, %
Регістрів	1415	1	1529	1
Таблиць LUT	843	1	934	1
Блоків пам’яті RAM18	19	6	18	6

Таким чином, з порівняльної таблиці 4.3 видно, що паралельна реалізація потребує незначного збільшення використання апаратних ресурсів програмовної логіки, але, в той же час, забезпечує вищу пропускну здатність.

4.2.3 Реалізація LDPC-декодеру на основі моделі паралельних черг запису/зчитування

Для перевірки розробленої моделі в роботі виконана реалізація декодеру з використанням моделі паралельних черг запису/зчитування на основі мови схемотехнічного опису VHDL у середовищі розробки Altera Quartus II 13.1 Web Edition. Основою для декодеру обрана мікросхема ПЛІС Altera Cyclone IV EP4CGX150DF31I7AD. Реалізація декодеру

проводилась для матриці перевірки парності з параметрами M =500, N =1000, $w_{r\max}$ =8.

Використання основних ресурсів ПЛІС для реалізації наведено у табл. 4.5.

Таблиця 4.5

Використання основних ресурсів ПЛІС

Ресурс	Використано, од.	Використано, %
Логічних елементів	4582	3
Комбінаційних елементів	2761	2
Регістрів	3937	3
Бітів пам’яті	86016	1

У табл. 4.6 наведені значення тактових частот, на яких може працювати мікросхема за граничних температурних умов.

Таблиця 4.6

Тактова частота роботи декодеру

Температура, °С	Тактова частота, МГц
100	198,37
-40	220,22

Для обраної матриці реалізовано декодер з послідовним записом/зчитуванням. Реалізація показала, що на зчитування всіх необхідних даних для обробки рядка необхідно 11 тактів, в той час як за (12) середня кількість тактів на обробку становила 11,178. Такий показник пояснюється необхідністю подачі додаткових сигналів. В той же час показник $M(T_{queue\_servicei})$ становив лише 2,178, що значно менше 8, які відповідають максимальній кількості елементів у рядку. Тому подальшим ресурсом для підвищення продуктивності декодеру є максимальне зменшення службових сигналів для розробленого блоку.

4.2.4 Реалізація LDPC-декодерів з використанням моделей подвійної буферизації вхідного повідомлення

Реалізація запропонованих моделей подвійної буферизації для LDPC-декодеру виконана для мікросхеми ПЛІС Altera сімейства Stratix IV. Результати відносно тактової частоти представлені у табл. 4.7.

Таблиця 4.7

Тактова частота роботи декодеру

Варіант реалізації	I	II	III
Тактова частота, МГц	220	250	270

Усі запропоновані варіанти вирішують проблему очікування запису перед декодуванням, тому вирішальним фактором при виборі рішення, що здатне забезпечити найбільшу пропускну спроможність є тактова частота, на якій може працювати декодер при обраному підході. Практична реалізація показала, що модель буферизації на основі подвоєння існуючого об'єму пам'яті для організації буферу переважає інші варіанти за параметром тактової частоти. Варто також зазначити, що в даному випадку використовується асинхронна комбінаційна логіка. Це пов'язано з тим, що кількість додаткових логічних ресурсів для організації подвійного буферу зводиться до мінімуму.

4.2.5 Реалізація декодеру з використанням особливостей двопортової блокової пам’яті ПЛІС

На основі розробленого підходу реалізовано LDPC-декодер для матриць, що можуть містить до 16000 стовпців та до 8000 рядків з максимальною кількістю значущих елементів у рядку – 20. Для реалізації використано середовище Quartus II Web Edition. Розрахункова пропускна здатність декодеру збільшилась з 23,6 Мбіт/с до 47,2 Мбіт/с. Для збереження індексів реалізовано блок у складі декодеру з використанням 249 блоків пам’яті типу M9K. Це дозволило у 20 разів зменшити кількість портів блокової пам’яті.

4.2.6 Реалізація реконфігуровного LDPC-декодеру

LDPC-декодер з можливістю реконфігурації реалізований за допомогою емуляції чотирьохпортової на основі підвищення її робочої тактової частоти в два рази. Реалізація виконана з використанням мови VHDL. Середовище розробки Altera Quartus 13.1, а обраний для реалізації пристрій – Altera Stratix IV EP4SGX180KF40C2. Результати щодо використання основних ресурсів ПЛІС наведені в табл. 4.8.

Таблиця 4.8

Використання основних ресурсів ПЛІС для реалізації декодеру

Назва ресурсу	**Використання**	
	Кількість елементів	***%***
Combinational ALUTs	10869	8
Dedicated logic registers	13017	9
Logic utilization	17188	12
M9K blocks	388	41
M144K blocks	0	0
PLL	2	25
I/O pins	16	2
Clock pins	16	57
Global clocks	2	12

Проведено моделювання роботи розробленого декодеру та порівняно результати з декодером, що використовує особливості блокової пам’яті (попередній підрозділ). Порівняння показало, що на основі запропонованих положень можливо кількість інформації, що обробляється одночасно, у два рази (рис. 4.23).

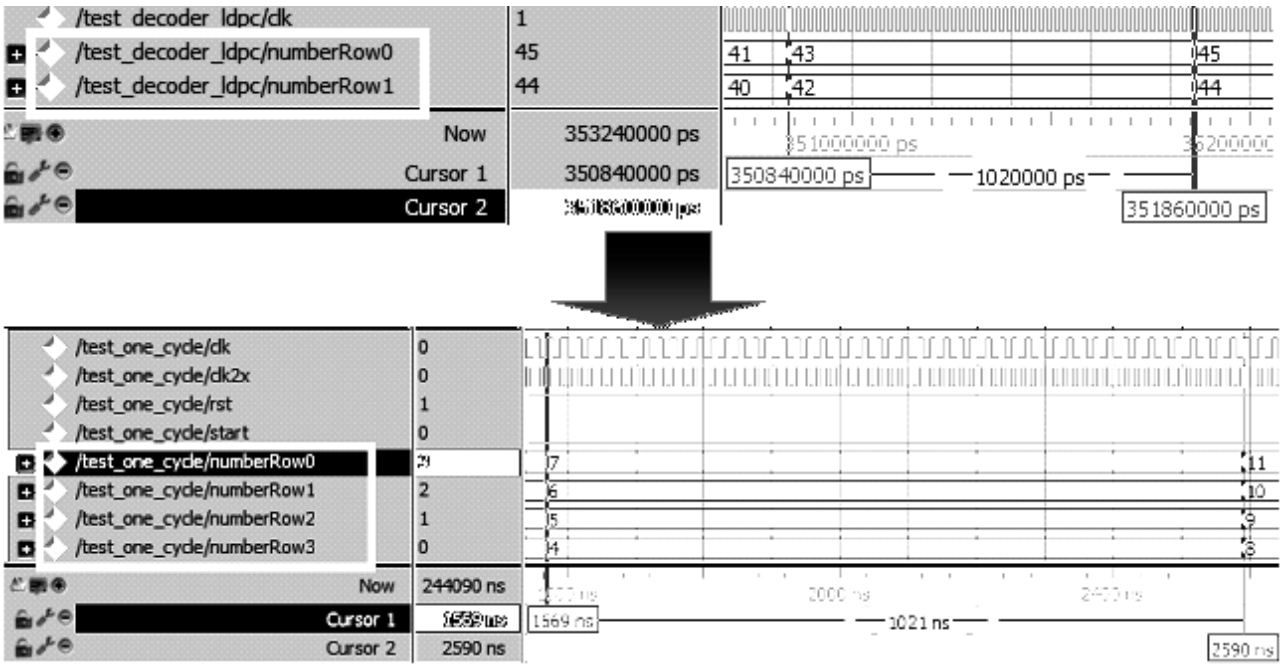


Рис. 4.23. Порівняння швидкодії для двох- та чотирьох процесорного декодеру

Крім того, проведено дослідження ефективності декодування декодеру за параметрами кількості бітів квантизації, співвідношення сигнал/шум та кількості помилок перевірки парності на виході декодеру. Тестування проводилось для декодеру з розрядністю від 4 до 8 бітів, в діапазоні відношення сигнал/шум 10-1,5 дБ. Досліджуваним параметром обрано відношення кількості рівнянь парності, що показали помилку після останньої ітерації декодування до загальної кількості рівнянь парності. При моделюванні використовувався стандартний пакет коректних даних, на який накладали шум, а результат подавали для декодування декодеру. Для отримання результатів дана операція повторювалась протягом 100 ітерацій. Результати проведених досліджень представлені на рис. 4.24.

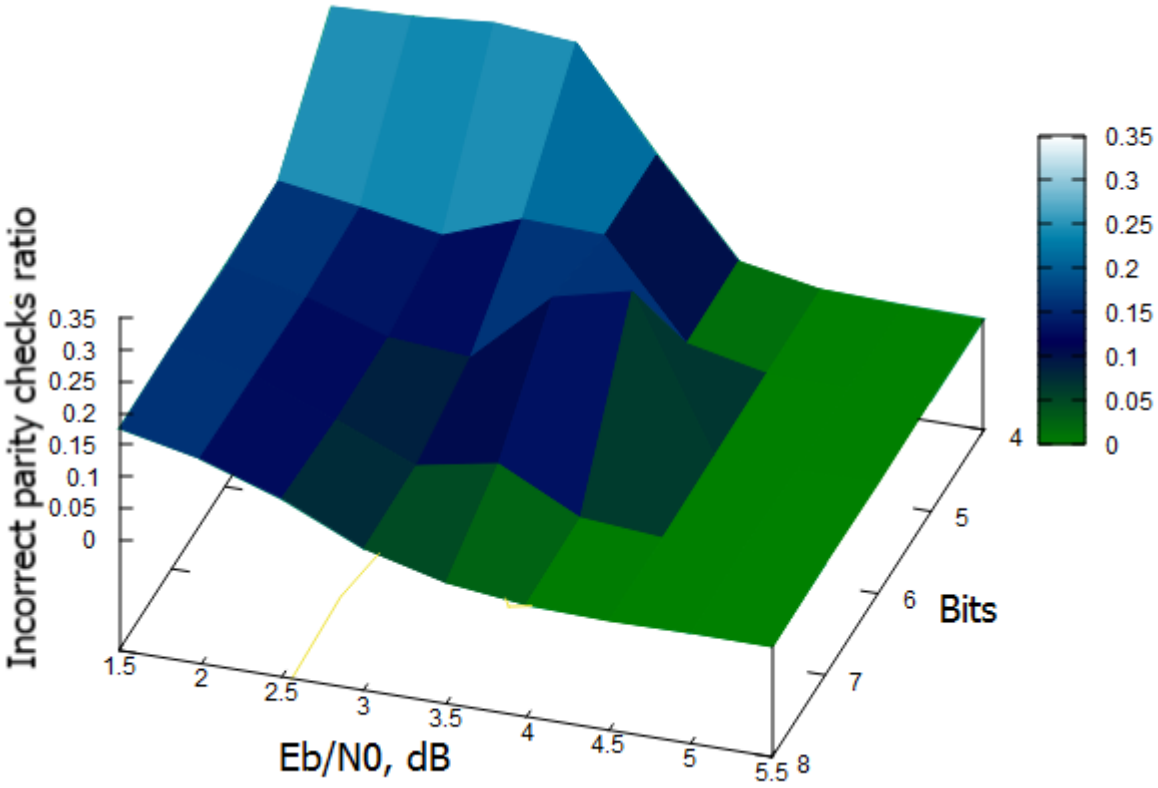


Рис. 4.24. Графік ефективності декодування в залежності від кількості бітів квантування та значення сигнал/шум

При моделюванні використано розроблене програмне забезпечення на мові програмування Java (рис. 4.25).

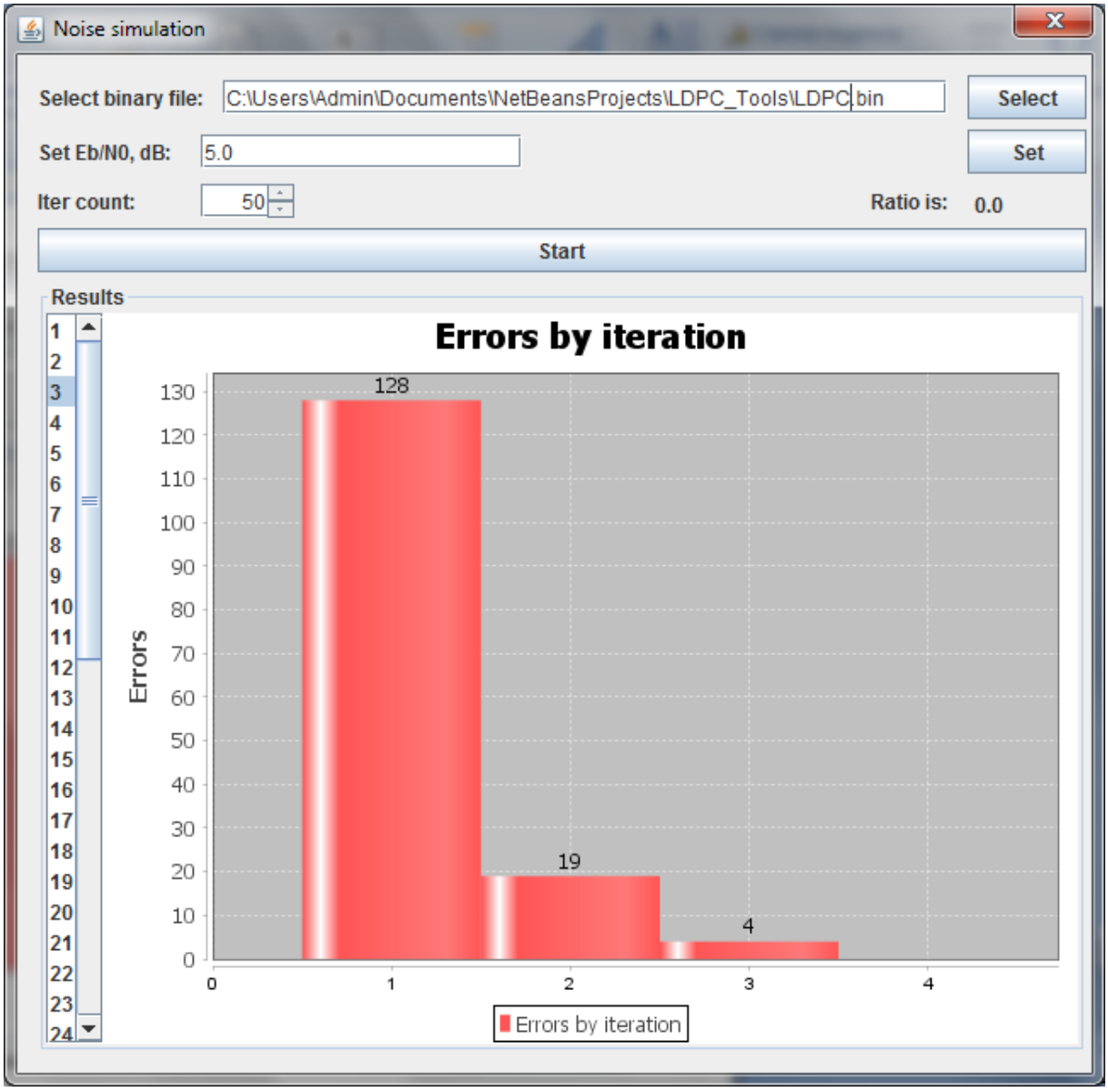


Рис. 4.25. Вікно програми моделювання обробки потоку пакетів

4.2.7 Реалізація конвеєрного LDPC-декодеру

Розроблені положення реалізовані для декодеру МПП, що містить 16000 стовпців та 4000 рядків. Реалізація виконана за допомогою мови схемо технічного опису VHDL. Для моделювання розробленого рішення використовувалось програмне середовище MentorGraphics ModelSim Starter Edition 10.1. На рис. 4.26 представлений скріншот початку процесу декодування.

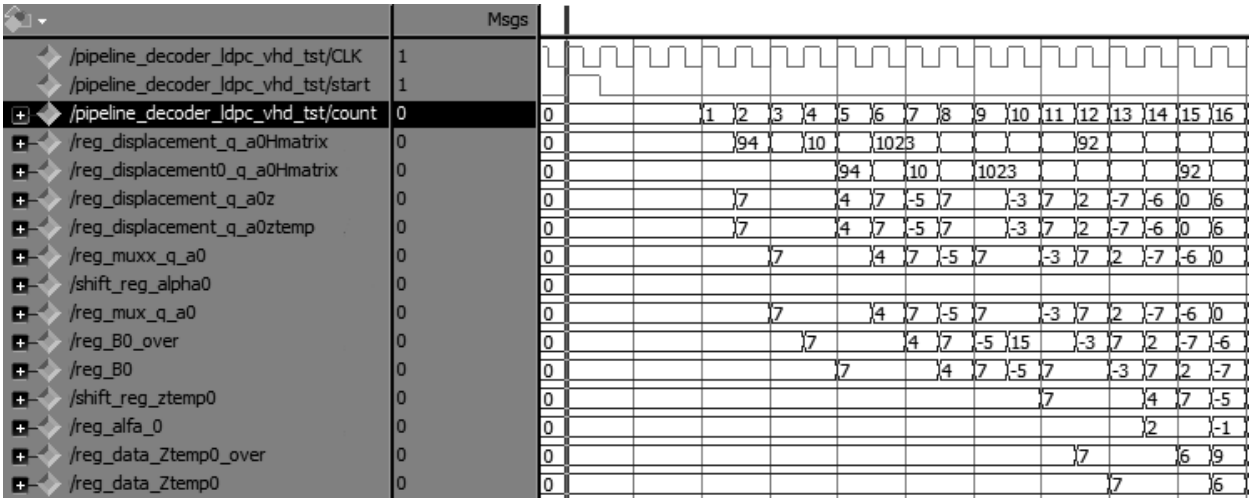


Рис. 4.26. Моделювання розробленої конвеєрної архітектури

З рисунку видно початок роботи конвеєру декодеру та поступове підключення до роботи всіх складових конвеєру. Конвеєрна організація обчислень дозволила підвищити пропускну здатність декодеру у порівнянні з не конвеєризованою версією в 40 разів.

Проведена процедура розміщення розробленого декодеру для ПЛІС Altera Stratix IV, яка показала, що декодер здатен працювати на тактовій частоті 240 МГц, а розрахункова пропускна здатність при 10 ітераціях становить понад 20 Мбіт/с.

4.3 Програмне тестування модифікованого алгоритму мінімальної суми

Для проведення перевірки переваги модифікованого алгоритму мінімальної суми над стандартним розроблено програмне забезпечення з використанням мови програмування Java та інтегрованого середовища розробки NetBeans 7.3.1 [92]. Дане програмне забезпечення (рис. 4.27) симулює спрощену передачу зображення через канал з шумами з використанням LDPC-декодування. Користувач має можливість обрати зображення для передачі та значення співвідношення сигнал/шум у каналі в дБ.

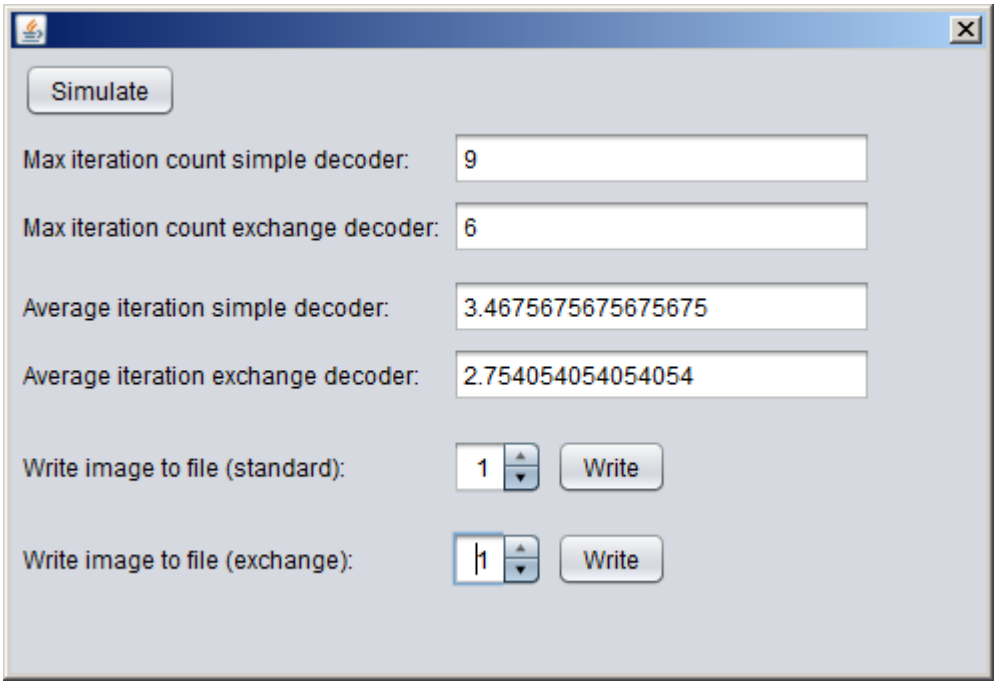


Рис. 4.27. Результати моделювання передачі зображення при значенні сигнал/шум 5 дБ

Користувач отримує дані про:

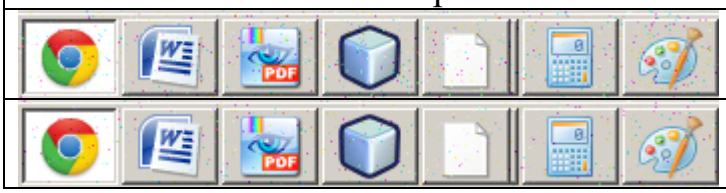
– середню кількість ітерацій, які необхідні для декодування усіх пакетів зображення;

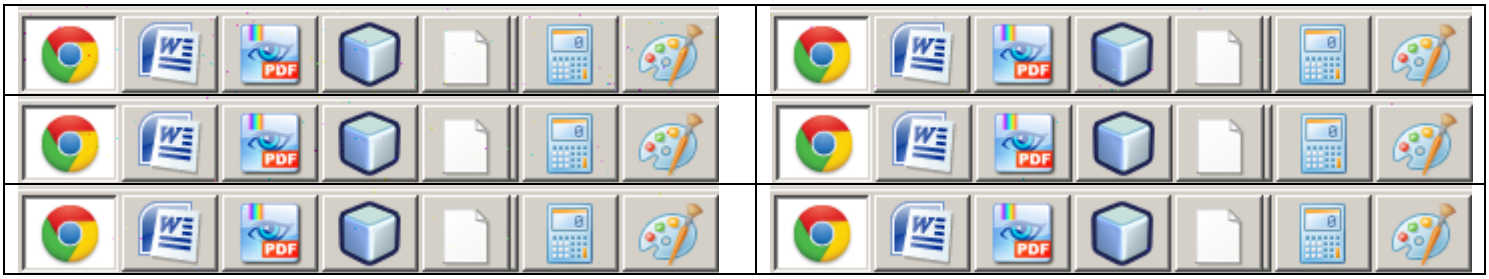
– максимальну кількість ітерацій при декодуванні для обох декодерів.

Крім того, є можливість записати зображення отримані за результатами кожної ітерації. В залежності від вказаного рівня шуму формується початкове зображення з шумами. У табл. 4.9 наведено результуючі зображення для перших п’яти ітерацій декодування для випадку, коли значення сигнал/шум становить 5 дБ.

Таблиця 4.9

Результуюче зображення за ітераціями

Базовий алгоритм	Модифікований алгоритм
	



На рис. 4.28 наведені показники для такого ж зображення, але при значенні сигнал/шум в 4.75 дБ.

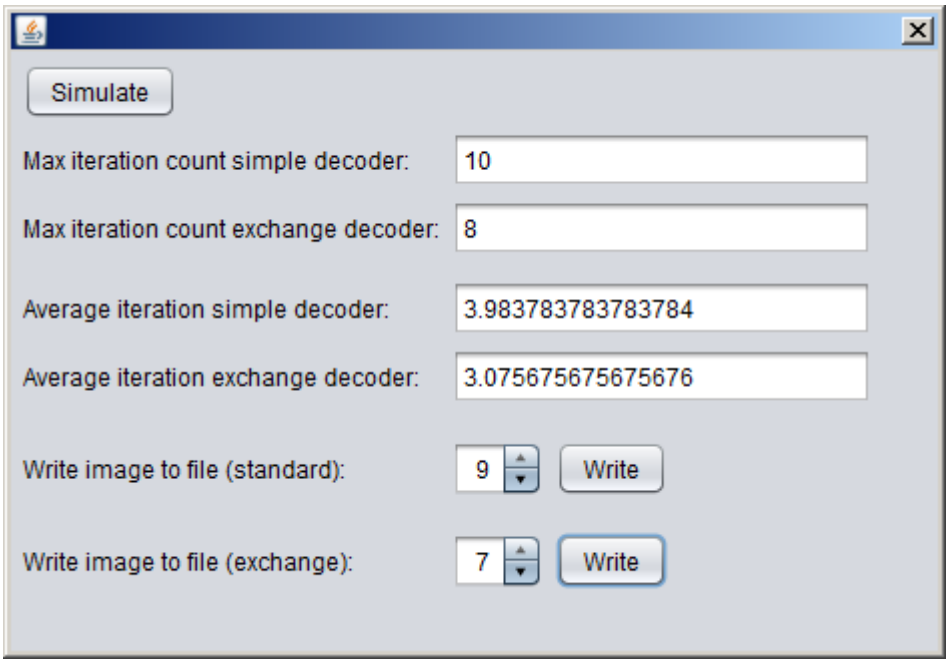
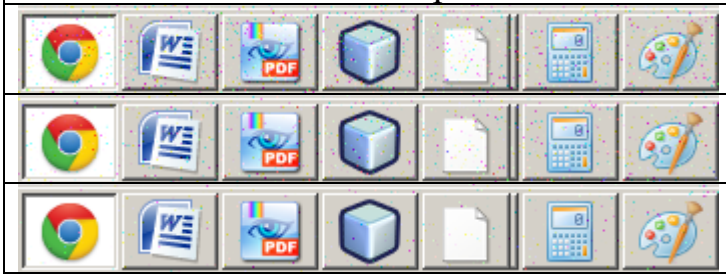
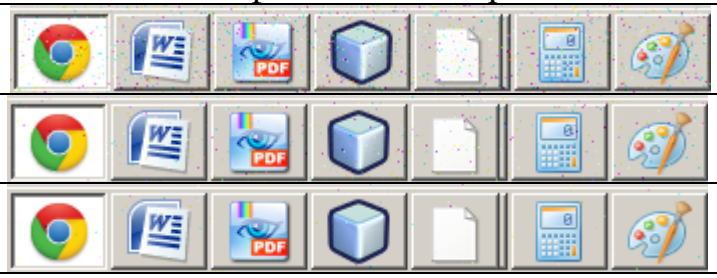


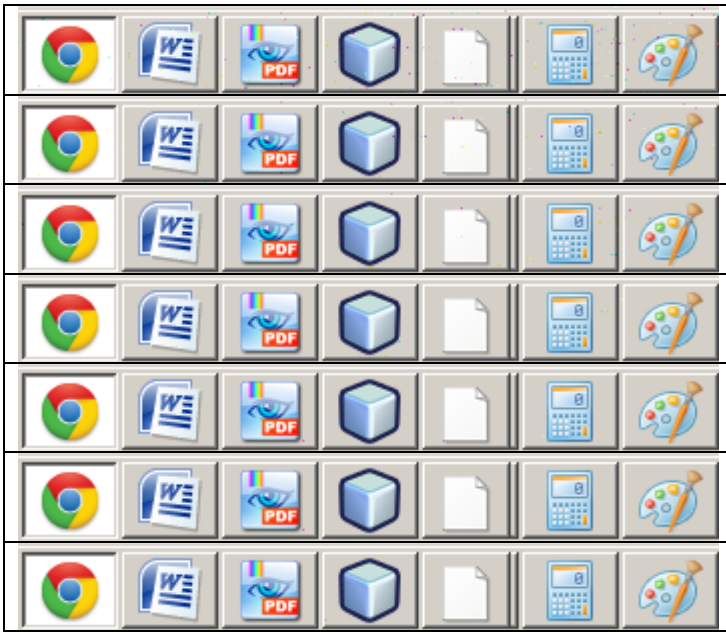
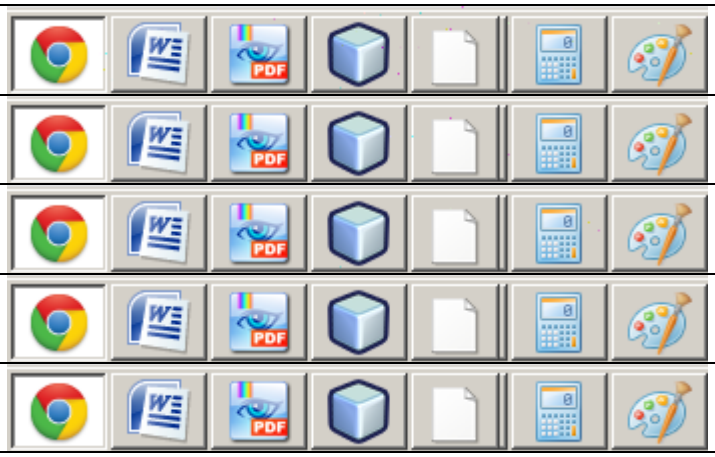
Рис. 4.28. Результати моделювання передачі зображення при значенні сигнал/шум 4.75 дБ

У табл. 4.10 представлені початкове та результуючі за ітераціями зображення.

Таблиця 4.10

Результуюче зображення за ітераціями

Базовий алгоритм	Модифікований алгоритм
	

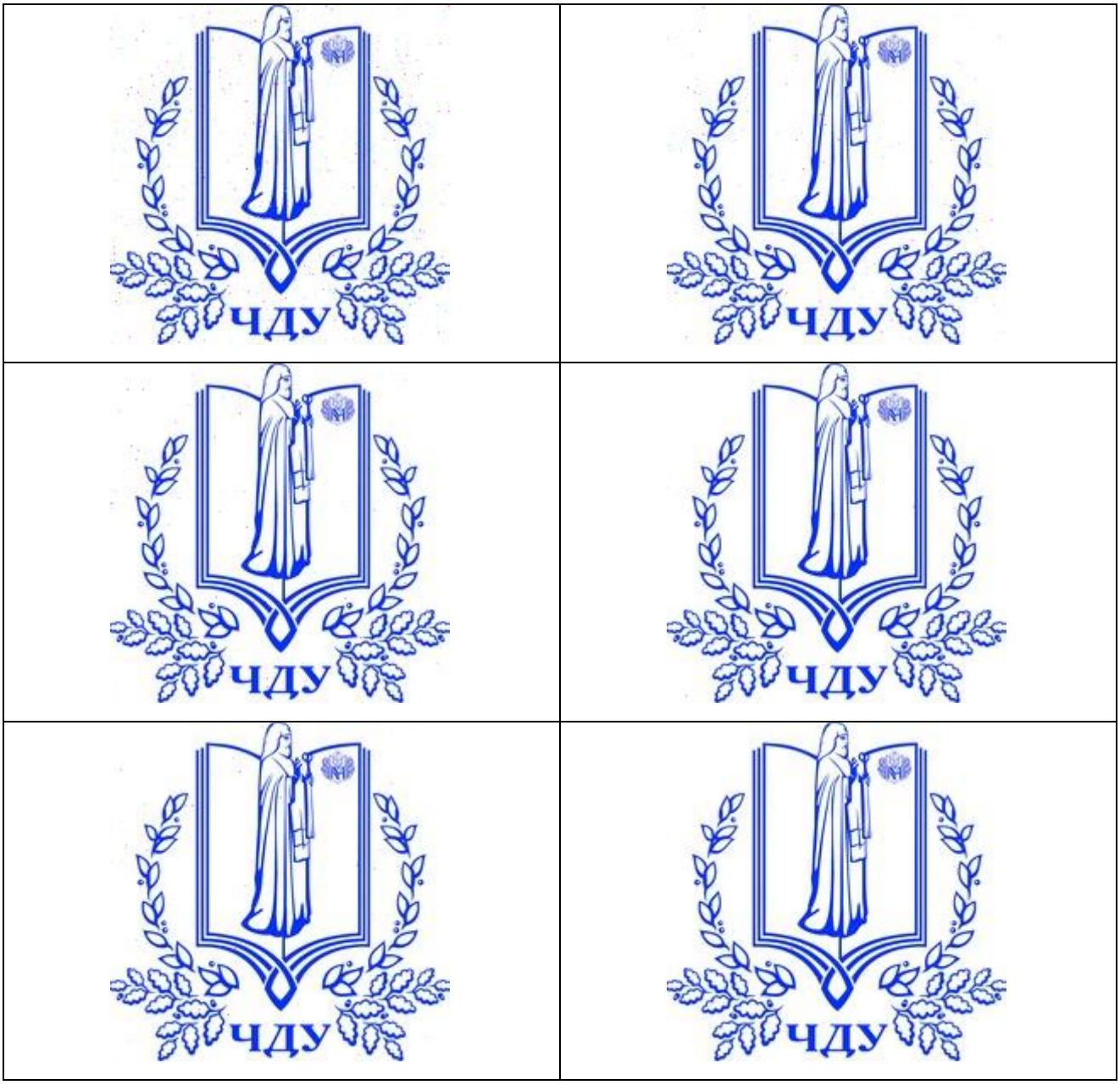
	
	–
	–

Результати декодування для іншого вхідного зображення за ітераціями наведені в табл. 4.11.

Таблиця 4.11

Результуюче зображення за ітераціями



Показники ефективності для випадку наведені на рис. 4.29.

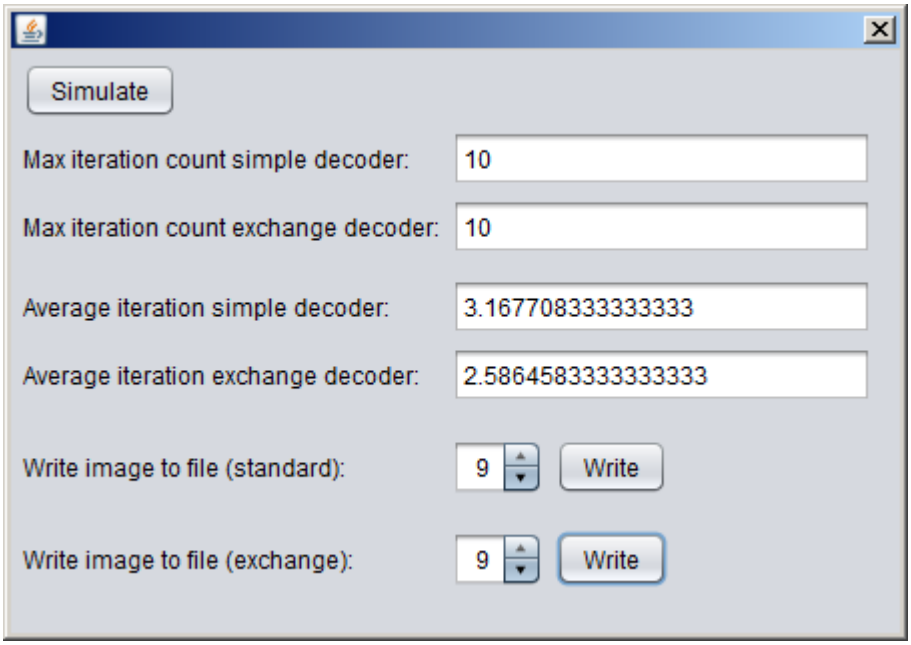


Рис. 4.29. Результати обробки для іншого зображення

Проведений експеримент зі зміною вхідного значення сигнал/шум у діапазоні 4-5,5 дБ. Результати експерименту представлені на рис. 4.30.

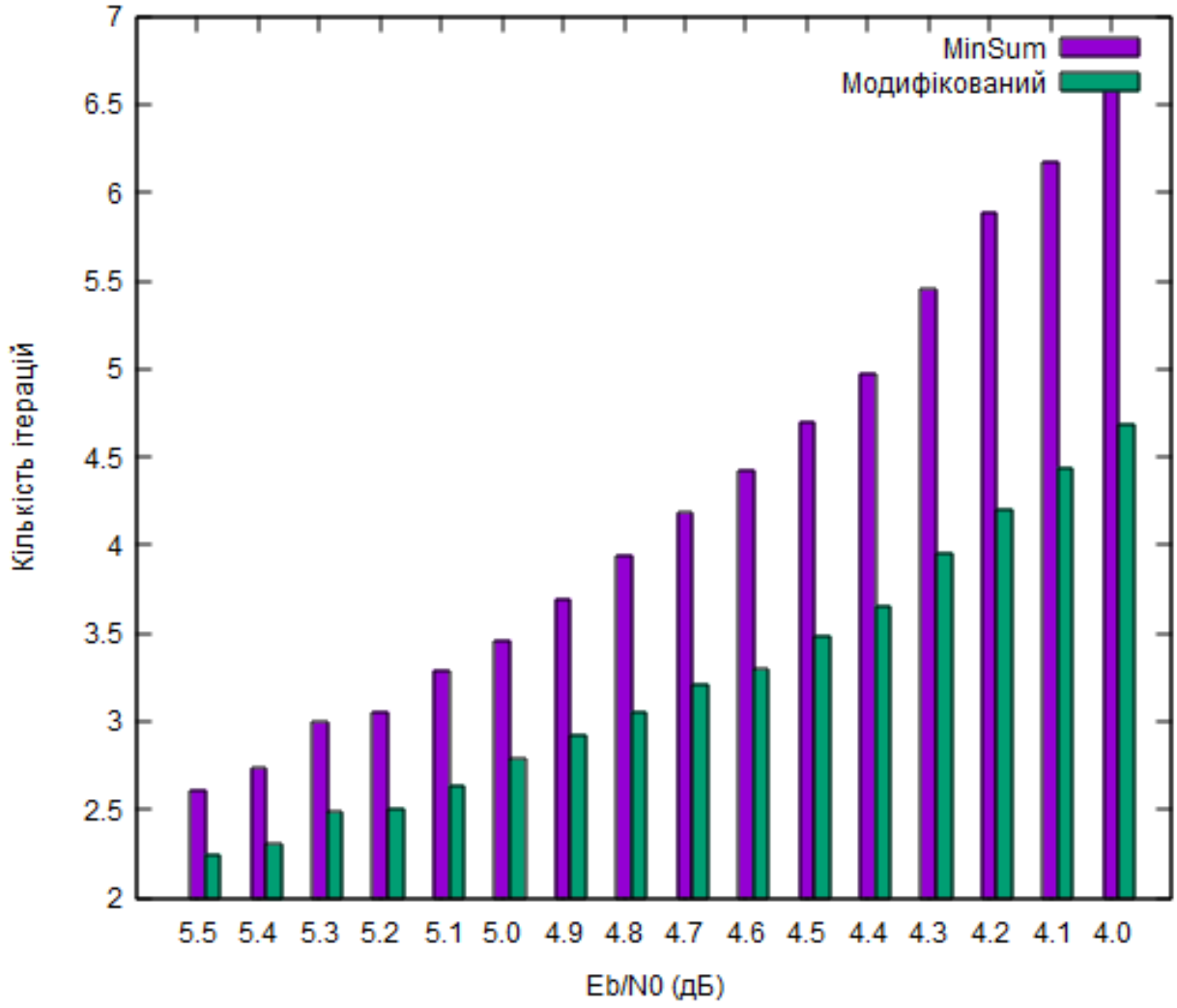


Рис. 4.30. Порівняння середньої кількості ітерацій декодування для алгоритму MinSum та модифікованого алгоритму

Даний експеримент підтверджує, що модифікована версія алгоритму потребує меншої кількості ітерацій декодування, ніж початкова версія алгоритму.

Висновки до розділу IV

1. Розроблено апаратно-програмний комплекс LDPC-декодування на основі представлених теоретичних положень. До складу комплексу входять програмне забезпечення на мовах програмування Java та C#, яке дозволяє проводити перевірку процесу декодування та відповідальне за передачу даних

до декодеру, схемотехнічний опис декодеру на мові VHDL. Для роботи комплексу використані відлагоджувальна плата ZedBoard та USB-міст FTDI FT2232H Mini-Module.

2. На основі представлених теоретичних положень виконана реалізація LDPC-декодерів. Для всіх положень створена відповідна реалізація. Отримані результати підтверджують, що, навіть, для найскладніших випадків обробки МПП.

3. Розроблено програмне забезпечення для перевірки роботи модифікованого алгоритму мінімальної суми. Результати перевірки показали, що модифікований алгоритм дозволяє більш агресивно виправляти помилки за умови невисокого значення сигнал/шум у каналі, що дозволяє підвищити пропускну здатність декодеру за рахунок зменшення кількості ітерацій декодування.

ВИСНОВКИ

У дисертаційній роботі на основі виконаних автором теоретичних досліджень та їх реалізації вирішено науково-технічну задачу підвищення ефективності організації частково паралельних LDPC-декодерів на базі ПЛІС для кодів на основі МПП з довільним розташуванням значущих елементів.

У роботі отримані такі основні наукові та практичні результати:

1. Проведений аналіз предметної області досліджень, який показав, що LDPC коди є завадостійкими кодами, що забезпечують найкращі характеристики виправлення помилок. Визначено, що найкращими LDPC-кодами є нерегулярні LDPC-коди на основі МПП з довільним (випадковим) розташуванням значущих елементів.

2. За рахунок зміни введення додаткового проміжного етапу обчислення удосконалено метод побудови частково паралельного LDPC-декодеру з алгоритмом декодування мінімальної суми, що дозволило зменшити використання ресурсів блокової пам’яті при реалізації декодеру на базі мікросхем ПЛІС; для тестової реалізації декодеру для коду зі швидкістю 1/2 виграш у кількості збережених блоків пам’яті становив 254 блоки; максимальний виграш від використання даного методу становить зменшення використання блокової пам’яті у 2 рази.

3. Отримала подальший розвиток модель організації обчислень для частково паралельного LDPC-декодеру за рахунок виконання операції обчислення синдрому та обчислення значень вузлів перевірки паралельно з іншими операціями відповідно до отримання даних при декодування, що дозволило підвищити швидкодію декодеру, яка для тестового прикладу підвищилась в 1.4 рази, а в залежності від параметрів МПП може становити 1,1…1,8 разів.

4. Удосконалено моделі побудови LDPC-декодеру шляхом застосування подвійної буферизації вхідних повідомлень, що дозволило зменшити час, необхідний на запис вхідного повідомлення, а отже, підвищити швидкодію

декодеру; за рахунок використання даних моделей вдається отримати виграш у пропускній здатності до 2 разів.

5. Отримала подальший розвиток модель частково паралельного LDPC-декодеру на основі організації паралельних черг запису/зчитування, що дозволило підвищити швидкодію декодеру в 1,1…1,5 рази.

6. Удосконалено метод побудови реконфігуровного частково паралельного LDPC-декодеру за рахунок організації чотирьохпортової пам'яті та принципів уникнення колізій, що дозволяє працювати з матрицями перевірки парності з довільним характером розташування значущих елементів; розроблений метод дозволяє підвищити швидкодію декодеру при реалізації на ПЛІС в 1.4…1.6 разів; результати підтверджуються проведеними випробуваннями на підприємстві Straight Way Electronix, s.r.o. (м. Прага, Чехія).

7. Удосконалено метод організації паралельних обчислень для частково паралельного LDPC-декодеру на основі попередньої обробки матриці перевірки парності, що дозволило використання двох блоків декодування, які працюють паралельно, за рахунок чого пропускну здатність частково паралельного LDPC-декодеру вдалося збільшити у 2 рази; результати підтверджуються проведеними випробуваннями на ПАТ Електротехнічний завод РЕЛСІС (м. Київ, Україна).

8. Вдосконалений у роботі метод побудови конвеєрної архітектури LDPC-декодеру для МПП з довільним розташуванням значущих елементів дозволяє у 20…50 разів підвищити пропускну здатність LDPC-декодеру; результати підтверджуються проведеними випробуваннями на підприємстві uPC Labs, Inc. (м. Стемфорд, США).

9. Виконано програмну та апаратну реалізації розроблених положень: розроблено програмне забезпечення та опис мовою VHDL декодерів на основі розроблених положень.

Результати дисертаційного дослідження впровадженні в навчальний процес у Чорноморському державному університеті імені Петра Могили (м. Миколаїв), Черкаському національному університеті ім. Б. Хмельницького,

Черкаському державному технологічному університеті, а також на трьох підприємствах: ПАТ Електротехнічний завод РЕЛСІС (м. Київ, Україна), Straight Way Electronix, s.r.o. (м. Прага, Чехія) та uPC Labs, Inc. (м. Стемфорд, США).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Berrou, C. (2010), Codes and Turbo Codes, Springer-Verlag, Paris.

2. Lin, S., and Costello, D. J. (2004), Error Control Coding (2nd edition), Prentice Hall, p. 1272.

3. Блейхут Р. Теория и практика кодов, контролирующих ошибки: Пер. с англ. - М.: Мир, 1986. - 576 с., ил.

4. Електронні системи: навчальний посібник / Й. Й. Білинський, К. В. Огороднік, М. Й. Юкиш. – Вінниця : ВНТУ, 2011. – 208 с.

5. Квєтний Р. Н. Методи та засоби передавання інформації у проблемно- орієнтованих розподілених комп'ютерних системах : монографія / Р. Н. Квєтний, А. Я. Кулик. – Вінниця : ВНТУ, 2010. – 362 с. – ISBN 978-966-641-372-0.

6. Скляр Б. Цифровая связь. Теоретические основы и Практическое применение. Изд. 2-е, испр. : Пер. с англ. – М. : Издательский дом "Вильямс", 2003. - 1104 с. : ил. - Парал. тит. англ. ISBN 5-8459-0497-8 (рус.).

7. Кларк Дж. мл. Кодирование с исправлением ошибок в системах цифровой связи / Дж. Кларк мл., Дж. Кейн. – М. : Радио и связь, 1987. – 391 с.

8. Moon, T. K. (2005), Error correction coding Mathematical methods and algorithms, Wiley-Interscience, Hoboken, New Jersey, 794 p.

9. Вернер М. Основы кодирования : учебник для ВУЗов. Москва: Техносфера, 2004. – 288 с. – ISBN 5-94836-019-9.

10. Морелос-Сарагоса Р. Искусство помехоустойчивого кодирования. Методы, алгоритмы, применение. – М. : Техносфера, 2005. – 320 с. ISBN 5-94836-035-0.

11. Gallager, R. (1963), Low-Density Parity-Check Codes, M.I.T. Press.

12. MacKay, D.J.C., and Neal, R.M. (1996), Near Shannon Limit Performance of Low Density Parity Check Codes, Electronics Letters.

13. Forward Error Correction [Електронний ресурс]. Режим доступу: URL: http://www.comtechefdata.com/technologies/fec. – Підзагол. з екрану.

14. Шеннон, К. Математическая теория связи / К. Шеннон // Работы по теории информации и кибернетике. – М. : Иностранная литература, 1963. – С. 243-332.

15. Золотарев В.В., Овечкин Г.В. Обзор исследований и разработок методов помехоустойчивого кодирования [Электронный ресурс]. Режим доступа: URL : http://www.mtdbest.iki.rssi.ru/pdf/obzor_po_kodir2.pdf. – Загол. с экрана.

16. Chung, S., et al. (2001), “On the design of low density parity-check codes within 0.0045 dB of the Shannon limit”, IEEEComm. Lett., Vol. 5, pp. 58-60.

17. Prasad, R., and Mihovska, A. (2009), “Coding and Modulation”, in New horizons in mobile and wireless communations, Vol. I, p. 105.

18. Uryvskyi L. O. Analysis of corrective properties of ultra-long LDPC codes [Електронний ресурс] / L. O. Uryvskyi, S. O. Osypchuk // Telecommunication sciences. - 2013. - Vol. 4, no. 1. - С. 21-26. - Режим доступу: http://nbuv.gov.ua/j-pdf/Telnau_2013_4_1_6.pdf.

19. Uryvsky L. O. Comparative analysis of LDPC and BCH codes error-correcting capabilities [Електронний ресурс] / L. O. Uryvsky, S. O. Osypchuk // Information and telecommunication sciences. - 2014. - Vol. 1, no. 1. - С. 5-9. - Режим доступу: http://nbuv.gov.ua/j-pdf/Telnau_2014_1_1_3.pdf.

20. Digital Video Broadcasting (DVB) (2009), Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2).

21. IEEE Standard for Information Technology – Telecommunications and information exchange between systems – Local and Metropolitan Area Network – Special Requirements. Part 11: Wireless LAN MAC and PHY Specifications. Amendment 5: Enhancements for Higher Throughput. pp. 289-293.

22. Laugner, P., Woodruff, B., and Kohl, B. (2006) “Moving 10 Gigabit Ethernet into a Volume Platform”, paper presented at DesignCon 2006 10 Gigabit Ethernet on Unshielded Twisted-Pair Cabling, Version 1, pp. 13-14.

23. Declercq, D., and Fossorier, M. (2007), “Decoding Algorithms for Nonbinary LDPC Codes Over GF(q)”, IEEE Transactions on Communications, Vol. 55, №4, pp. 633-643.

24. Luby, M.G., Mitzenmacher, M., Shokrollahi, M.A., and Spielman D.A. (1998), “Improved Low-Density Parity-Check Codes Using Irregular Graphs and Belief Propagation”, Proceedings of 1998 IEEE Intl. Symp. Inform. Theory, Cambridge, August 16-21, 1998, p. 171.

25. MacKay, D.J.C. (1998), “Gallager Codes That Are Better Than Turbo Codes”, Proceedings of 36th Allerton Conf. Commun., Control, and Computing, Montiecello, III., September 1998.

26. H. Jin, A. Khandekar, and R. J. McEliece (2000), “Irregular repeat-accumulate codes”, Proceedings 2nd Int. Symp. Turbo Codes and Related Topics , Brest, France, Sept. 4, 2000, pp. 1–8.

27. Мусиенко М. П. Разработка навигационных програмно-аппаратных GPS/GPRS комплексов на движущихся объектах [Текст] / М. П. Мусиенко, В. И. Томенко, О. Л. Савчук, М. П. Рудь // Вісник Черкаського державного технологічного університету. – 2007. – № 1. – С. 119-122.

28. ZIT Track (2012), System On-Line Monitoring "ZIT Track", available at: http://zit.lviv.ua/index.php/gps-monitoring-en.html (accessed 23 November, 2013).

29. Al-Khedler, M. A. (2011), “Hybrid GPS-GSM Localization of Automobile Tracking System”, International Journal of Computer Science & Information Technology (IJCSIT), Vol. 3, No 6., Dec 2011, pp. 75-85.

30. Vamsi, K. P., and Yugandhar, D. (2013), “An Enhanced Railway Transport System using FPGA through GPS & GSM”, International Journal of Soft Computing and Engineering (IJSCE), Vol 2, No 6, January 2013.

31. Мельник А.О. Архітектура комп'ютера - Наукове видання. - Луцьк: Волинська обласна друкарня, 2008. - 470 с. ISBN 978-966-361-264-5.

32. Харрис Д.М., Харрис С.Л. Цифровая схемотехника и архитектура компьютера. - Morgan Kauffman, 2013. - 1622 с.

33. Тарасов И. Описание архитектуры FPGA семейств UltraScale компании Xilinx / И. Тарасов // Компоненты и технологии. – 2014. – №2. – С. 38-46.

34. Зотов В.Ю. Проективание цифровых устройств на основе ПЛИС фирмы Xilinx в САПР WebPack ISE. - М.: Горячая линия-Телеком, 2003. – 624 с. : ил.

35. Максфилд К. Проектирование на ПЛИС. Курс молодого бойца. – М.: Издательский дом «Додэка-XXI», 2007. – 408 с.: илл. (Серия «Программируемые системы»). – ISBN 978-5-94120-147-1.

36. В. В. Казимир Проектування комп'ютерних систем на основі мікросхем програмованої логіки : монографія / С. А. Іванець, Ю. О. Зубань, В. В. Казимир, В. В. Литвинов. – Суми : Сумський державний університет, 2013. – 313 с.

37. Джозеф Ю. Ядро Cortex M3 компании ARM. Полное руководство / Джозеф Ю; пер. с англ. А. В. Евстифеева. - М. : Додэка XXI, 2012. - 552 с. : ил. (Мировая электроника). - Доп. тит. л. англ. - ISBN 978-5-94120-243-0.

38. SIMCOM (2013), SIM908 GSM/GPRS+GPS Module available at: http://www.simcom.us/product_detail.php?cid=1&pid=38 (accessed 22 November, 2013).

39. AHA4701 30 Mbps LDPC Low Density Parity Check Code Encoder/Decoder Core Product Brief.

40. AHA4704 15 Mbps LDPC Low Density Parity Check Code Encoder/Decoder Core, Product Brief.

41. Falcão, G., Sousa, L., and Silva, V. (2008), “Massive Parallel LDPC Decoding on GPU”, In Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming (PPoPP’08), Salt Lake City, Utah, USA. ACM, p. 83–90.

42. Falcão, G., Yamagiwa, S., Silva, V., and Sousa, L. (2007), “Stream-Based LDPC Decoding on GPUs”, In Proceedings of the First Workshop on General Purpose Processing on Graphics Processing Units, GPGPU’07, p. 1–7.

43. Velayuthan, P.M. (2012). Towards Optimized Flexible Multi-ASIP Architectures for LDPC/Turbo Decoding, Electronics, T´el´ecom Bretagne, Universit´e de Bretagne-Sud, 2012.

44. Falcao G. (2010), Parallel Algorithms and Architectures for LDPC Decoding, Gabriel Falcao Ph.D. Thesis, FCTUC, University of Coimbra, 204 p.

45. Porcello, J.C. (2014), “Designing and implementing Low Density Parity Check (LDPC) Decoders using FPGAs”, 2014 IEEE Aerospace Conference, pp. 1-7.

46. Darabiha, A., Carusone, A.C., Kschischang, F.R. (2006), “A Bit-Serial Approximate Min-Sum LDPC Decoder and FPGA Implementation”, 2006 IEEE International Symposium on Circuits and Systems, 2006.

47. Balatsoukas-Stimming, A., Dollas, A. (2012), “FPGA-Based Design and Implementation of a Multi-GBPS LDPC Decoder”, 2012 22nd International Conference on Field Programmable Logic and Applications (FPL), pp. 262-269.

48. Mohsenin ,T. (2010) Algorithms and Architectures for Efficient Low Density Parity Check (LDPC) Decoder Hardware, Tinoosh Mohsenin Dissertation Submitted in partial satisfaction of the requirements for the degree of Doctor of Philosophy in Electrical Engineering and Computer Science, University of California, Davis, 102 p.

49. Syafei, W.A., Yohena, R., Shimajiri, H., Yoshida, T. et al. (2009), “Performance Evaluation and ASIC Design of LDPC Decoder for IEEE802.11n”, Proceedings of 6th IEEE Consumer Communications and Networking Conference, 2009, pp. 1-5.

50. Эккель Б. Философия Java. Библиотека программиста. 4-е изд. - СПб.: Питер, 2009. - 640 с.: ил.

51. Munshi, A., Gaster, B. R., Mattson, T. G., Fung, J., and Ginsburg D. (2011), OpenCL Programming Guide, Addison-Wesley Professional, 648 p. — ISBN 978-0-321-74964-2.

52. Сандерс Дж., Кэндрот Э. Технология CUDA в примерах: введение в программирование графических процессоров: Пер. с англ. Слинкина А.А., научный редактор Боресков А.В. - М.: ДМК Пресс, 2011. - 232 с.: ил.

53. Chu, P.P. (2005). RTL Hardware Design Using VHDL Coding for Efficiency, Portability, and Scalability, A Wiley Interscience Publications, 694 p.

54. Tanner, R.M. (1981), “A recursive approach to low complexity codes”, IEEE Transactions on Information Theory, Vol. 27, No. 5, Sept. 1981, pp. 533-547.

55. Bo, L., Wang, G., and Yang, H.-J. (2009), “A new method of detecting cycles in Tanner graph of LDPC codes”, Proceedings of International Conference on Wireless and Signal Processing, 2009, pp. 1-3.

56. Soleymani, M.R., Gao., Y., and Vilaipornsawai, U. (2002), Turbo Coding For Satellite and Wireless Communications, Kluwer Academic Publishing, Dodrecht. eBook ISBN: 0-306-47677-0, pp. 189-194.

57. Prabhakar, A., and Narayanan, K. A (2005), “Memory efficient serial LDPC decoder architecture”, Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, v/41 - v/44 Vol. 5, 2005.

58. Tehrani, S.S., Mannor, S., and Gross, W.J. (2008), “Fully Parallel Stohastic LDPC Decoders”, IEEE Transactions on Signal Processing, Vol. 56, No. 11, November 2008, pp. 5692-5703.

59. Mehr, H.S., Mohsenin, T., and Baas, B. (2011), “A Reduced Routing Network Architecture for Partial Parallel LDPC decoders”, Proceedings of 2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR), pp. 2192-2196.

60. Wang, Z., and Cui, Z. (2005), “A memory efficient partially parallel decoder architecture for QC-LDPC codes,” in Proc. of the 39th Asilomar Conference on Signals, Systems & Computers, pp. 729-733.

61. Wang, W., Li, L., and Zhang, H. (2014), “Simplified partially parallel DVB-S2 LDPC decoder architectural design based on FPGA”, 2014 IEEE/CIC International Conference on Communications in China, 2014, pp. 314-318.

62. Гладких А. А., Основы теории мягкого декодирования избыточных кодов в стирающем канале связи / А. А. Гладких. – Ульяновск : УлГТУ, 2010. – 379 с.

63. Fossorier, M.P.C. (2004), “Reliability-Based Soft-Decision Decoding Algorithms for Linear Block Codes” in Lin, S., and Costello D.J (Ed.), Error Control Coding (2nd edition), Prentice Hall, pp. 395-400.

64. Панкратов А. В. Исследование LDPC декодеров [Текст] / А. В. Панкратов // Молодой ученый. — 2011. — №7. Т.1. — С. 39-42.

65. Hu, X.-Y., Eleftheriou, E., Arnold, D.-M., and Dholakia, A. (2001), “Efficient implementations of the sum-product algorithm for decoding LDPC codes”, Global Telecommunications Conference, 2001, Vol. 2, pp. 1036-1036.

66. Chung, S.-Y., Richardson, T.J., and Urbanke, R.L. (2001), “Analysis of Sum-Product Decoding of Low-DensityParity-Check Codes Using a Gaussian Approximation”, IEEE TRANSACTIONS ON INFORMATION THEORY, Vol. 47, No. 2, February 2001.

67. Traore, N., Kant, S., Jensen, T.L. (2007), Message Passing Algorithm and Linear Programming Decoding for LDPC and Linear Block Codes, Aalborg University, pp. 32-53.

68. Fossorier, M.P.C., Mihaljević, M., and Imai, H. (1999), “Reduced complexity iterative decoding of low-density parity check codes based on belief propagation”, IEEE Transactions on Communications, Vol. 47, Issue 5, 673-680 pp.

69. L.Bahl, J.Cocke, F.Jelinek, and J.Raviv (1974), "Optimal Decoding of Linear Codes for minimizing symbol error rate", IEEE Transactions on Information Theory, March 1974, vol. IT-20(2), pp.284-287.

70. J. Hagenauer , E. Offer and L. Papke (1996), "Iterative decoding of binary block and convolutional codes", IEEE Trans. Inf. Theory, vol. 42, no. 2, pp.429-445.

71. F. Guilloud, E. Boutillon, and J.-L. Danger, (2003), “Lambda-min decoding algorithm of regular and irregular LDPC codes”, Proceedings of the 3rd International Symposium on Turbo Codes and Related Topics, Sept. 2003.

72. Znong, Z., Li, Y., Chen, X., Hu, H., amd Wang, J. “Modified Min-sum Decoding Algorithm for LDPC Codes Based on Classified Correction”.

73. Ullah, W., Jiangtao, and Fengfan, Y. “Two-way normalization of min-sum decoding algorithm for medium and short length low density parity check codes”.

74. Mohsenin, T., Truong, D., ans Baas, B. (2009), “Multi-Split-Row Threshold Decoding Implementations for LDPC Codes”, IEEE International Symposium on Circuits and Systems, 2009, pp. 2449-2452.

75. Cui, Z., Wang, Z., and Zhang, X. (2011), “Reduced-Complexity Column-Layered Decoding and Implementation for LDPC Codes”, Communications, IET, Vol. 5, Issue 15, pp. 2177-2186.

76. Guilloud, F. (2004), Generic Architecture for LDPC Codes Decoding, PhD Thesis By Frederic Guilloud, ENST Paris, July, 2004, 200 p.

77. Chen, X., Kang, J., Lin, S., and Akella, V. (2010), “Memory System Optimization for FPGA-Based Implementation of Quasi-Cyclic LDPC Codes Decoders”, IEEE Transactions on Circuits and Systems I: Regular Papers, Vol.58, Issue 1, pp. 98-111.

78. Cao, Z., Kang, J., amd Fan, P. (2005), “An FPGA implementation of a structured irregular LDPC decoder”, IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications, 2005, Vol. 2, pp. 1050-1053.

79. Zhang, L., amd Jiang, Y. (2013), “A low-hardware consumption FPGA based configurable LDPC decoder”, 2013 International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS), pp. 221-224.

80. Loi, K.C.C. (2010), Field-Programmable Gate-Array (FPGA) Implementation of Low-Density Parity-Check (LDPC) Decoder in Digital Video Broadcasting – Second Generation Satellite (DVB-S2), Thesis for the Degree of Master of Science, 2010.

81. Radosavljevic, P., de Baynast, A., Karkooti, M., ans Cavallaro, J.R. (2006), High-throughput multi-rate LDPC-decoder based on architecture-oriented parity check matrices, 2006 14th European Signal Processing Conference, pp. 1-5.

82. Karkooti, M., Radosavljevic, P., and Cavallaro, J.R. (2006), “Configurable, High Throughput, Irregular LDPC Decoder Architecture: Tradeoff Analysis and Implementation”, International Conference on Application-specific Systems, Architectures and Processors, 2006, pp. 360-367.

83. Radosavljevic, P., de Baynast, A., Karkooti, M., and Cavallaro, J.R. (2006), “Multi-Rate High-Throughput LDPC Decoder: Tradeoff Analysis Between Decoding Throughput and Area”, The 17th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC’06).

84. Karkooti, M. (2004), Semi-Parallel Architectures For Real-Time LDPC Coding, A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree Master of Science By Marjan Karkooti, Rice University, Houston, Texas, 87 p.

85. Zhong, H., Xu, W., Xie, N., and Zhang, T. (2007), “Area-Efficient Min-Sum Decoder Design for High-Rate QC-LDPC Codes in Magnetic Recording”, IEEE Transactions on Magnetics, Vol. 43, Issue 12, pp. 4117-4122. ISSN – 0018-9464.

86. Zhang, Z. (2010), “An Efficient 10GBASE-T Ethernet LDPC Decoder Design With Low Error Floors”, IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 45, NO. 4, APRIL 2010.

87. Zhang, Z. (2009), Design of LDPC Decoders for Improved Low Error Rate Performance, A dissertation submitted in partial satisfaction of the requirements for the degree of Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences, University of California, Berkeley.

88. Zhang, Z., Dolecek, L., Nikolic, B., Anantharam, V. et al. (2009), “Design of LDPC decoders for improved low error rate performance: quantization and algorithm choices”, IEEE Transactions on Communications, Vol. 57 , Issue 11, pp. 3258-3268.

89. Falcao, G., Silva, V., Marinho, J., and Sousa, L. (2009) “LDPC Decoders for the WiMAX (IEEE 802.16e) Based on Multicore Architectures”, WIMAX New Developments, Upena D Dalal and Y P Kosta (Ed.), ISBN: 978-953-7619-53-4, InTech, DOI: 10.5772/8265. Available at:

http://www.intechopen.com/books/wimax-new-developments/ldpc-decoders-for-the-wimax-ieee-802-16e-based-on-multicore-architectures.

90. Zhang, T., and Parhi, K.K. (2002), “A 54 MBPS (3,6)-Regular LDPC Decoder”, IEEE Workshop on Signal Processing Systems, 2002, pp. 127-132.

91. Chi, H. (2009), Hardware Design of Decoder for Low-Density Parity Check Codes, Helsinki University of Technology, Master of Science Thesis.

92. Pei, Y., Yin, L., and Lu, J. (2004), Design of Irregular LDPC Codec on a Single FPGA Chip.

93. Gunnam, K.K., Choi, G.S., Yeary, M.B., and Atiquzaaman, M. (2007), “VLSI Architectures for Layered Decoding for Irregular LDPC Codes for WiMAX”, IEEE International Conference on Communications, 2007, pp. 4542-4547.

94. Marchand, C., Conde-Canencia, L., and Boutillon, E. (2013), “High-speed conflict-free layered LDPC-decoder for the DVB-S2, -T2 and -C2 standards, 2013 IEEE Workshop on SIgnal Processing Systems (SISP'2013), France (2013).

95. Leduc-Primeau, F., Gaudet, V.C., and Gross, W.J. (2015), “Stohastic Decoder for LDPC Codes”, in Chavet, C., Coussy P. (Ed.) Advanced Hardware Design for Error Correcting Codes, Springer International Publishing Switzerland, p. 192.

96. Chavet, S., Sani, A.H., and Coussy, P. (2015), “Hardware Design of Parallel Interleaver Architecture: A Survey”, in Chavet, C., Coussy, P. (Ed.), Advanced Hardware Design for Error Correcting Codes, Springer International Publishing Switzerland, pp. 180-183.

97. Marchand, C., Conde-Canencia, L., and Boutillon, E. (2010), “Architecture and Finite Precision Optimization for Layered LDPC Decoders”, Signal Processing Systems (SIPS), 2010, pp.350-355.

98. Zhang, K., and Huang, X. (2009), “A low-complexity rate-compatible LDPC-decoder”, 2009 Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers, pp. 749-753.

99. Aravind, N., and Kumar, P. (2013), “Low Hardware Layered Decoding Architecture for LDPC Codes”, International Journal of Science and Research (IJSR), Volume 2 Issue 3, March 2013.

100. Gunnam, K.K., Choi, G.S., Wang, W., and Kim, E. (2006), “Decoding of Quasi-cyclic LDPC Codes Using an On-the-Fly Computation”, Fortieth Asilomar Conference on Signals, Systems and Computers, 2006, pp. 1192-1199.

101. Sun, Y., and Cavallaro, J.R. (2013), “VLSI Architecture for Layered Decoding of QC-LDPC Codes with High Circulant Weight”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 21, Issue 10, 2013, pp. 1960-1964.

102. Darabiha, A., Carusone, A.C., and Kschischang, F.R. (2008), “Block-Interlaced LDPC Decoders With Reduced Interconnect Complexity”, IEEE Transactions on Circuits and Systems—II, Express briefs, Vol. 55, No. 1, January 2008.

103. Beuschel, C. (2010), Fully Programmable LDPC Decoder Hardware Architectures, Dissertation, Ulm University.

104. Mansour, M.M., and Shanbhag, N.R. (2006), “A 640-Mb/s 2048-Bit Programmable LDPC Decoder Chip”, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 41, No. 3, March 2006, pp. 684-698.

105. Bresenham, J. E. (1 January 1965), "Algorithm for computer control of a digital plotter", IBM Systems Journal 4 (1), pp. 25–30.

106. Abrash, Michael (1997), Michael Abrash's graphics programming black book. Albany, NY: Coriolis, pp. 654–678. ISBN 978-1-57610-174-2.

107. Крайник Я.М. Підвищення ефективності використання пам'яті частково паралельного LDPC-декодеру / Я.М. Крайник // Вісник Черкаського державного технологічного університету. – 2014. С. 10-14. ISSN 2306-4455.

108. Пристрій LDPC декодування / Крайник Я.М., Денисов О.О., Мусієнко М.П., Заявка №u201500416 / Патент опубліковано 27.04.2015, бюл. № 8/2015, Номер патенту: 98611.

109. Мусієнко М.П. Паралельна реалізація алгоритму мінімальної суми для LDPC-декодеру / Мусієнко М.П., Крайник Я.М. // Збірник наукових праць ВІКНУ. – 2014. – Вип. 47. – С. 139-147.

110. Крайник Я.М. Організація паралельних обчислень алгоритму мінімальної суми для LDPC декодеру / Я.М. Крайник / ІНТЕРНЕТ-ОСВІТА-НАУКА-2014, дев'ята міжнародна науково практична конференція ІОН-2014, 14-17 жовтня : Збірник праць. – Вінниця : ВНТУ, 2014. – С. 134-135. – ISBN 978-966-641-491-8.

111. Мусієнко М.П. Підвищення швидкодії декодеру нерегулярних LDPC-кодів на основі організації паралельних черг запису/зчитування / Мусієнко М.П., Крайник Я.М. // Вимірювальна та обчислювальна техніка в технологічних процесах. – №3(48). – 2014. – С. 111-115. – ISSN 2219-9365.

112. Крайник Я.М. Реалізація частково паралельного LDPC декодеру на базі паралельних черг / Я.М. Крайник / Могилянські читання-2014, всеукраїнська науково-методична конференція, Миколаїв. – Миколаїв, Вид-во ЧДУ ім. П. Могили. – С. 66-67.

113. Musiyenko M. Reconfigurable decoder for irregular random low density parity check matrix based on FPGA / Musiyenko M., Krainyk. Y., Denysov. O // Conference Proceedings of IEEE ELNANO-2015 – 2015. – pp. 498-503.

114. Sawyer, N., and Deffosez, M. (2002), "Quad-Port Memories in Virtex Devices", Xilinx, Application Note.

115. Крайник Я.М. Конвеєрна архітектура LDPC-декодування на базі пліс з використанням модифікованого алгоритму мінімальної суми / Крайник Я.М., Денисов О.О. // Вісник Черкаського державного технологічного університету. – №4. – 2015. – С. 23-30.

116. Крайник Я.М. Забезпечення побудови високошвидкісного LDPC декодеру для нерегулярних матриць перевірки парності з довільним розташуванням значущих елементів/ Крайкник Я.М., Денисов О.О. / МНПК Ольвійський форум – 2015: стратегії розвитку країн Причорноморського

регіону в геополітичному просторі, Миколаїв, 3-6 червня 2015, Тези, Том 2. – Миколаїв: Вид-во ЧДУ ім. П. Могили, 2015. – С. 103-104.

117. Мусієнко М.П. Підвищення швидкодії LDPC-декодеру на основі організації подвійного буферу вхідного повідомлення / Мусієнко М.П., Крайник Я.М., Куценко С.В. // Системи обробки інформації. – 2014. – Вип. 9. – С. 54-57.

118. Крайник Я.М. Підвищення швидкодії LDPC декодеру за допомогою подвійної буферизації повідомлень / Я.М. Крайник / Сучасні проблеми і досягнення науки в галузі радіотехніки, телекомунікацій та інформаційних технологій: Тези доповідей VII Міжнародної науково практичної конференції, Запоріжжя, 17 19 вересня 2014 р. – Запоріжжя : ЗНТУ, 2014. – С. 94-95. – ISBN 978-617-529-098-9.

119. Крайник Я.М. побудова частково паралельного LDPC-декодеру з використанням особливостей двопортової блокової пам’яті ПЛІС / Я.М. Крайник // Наукові праці [Чорноморського державного університету імені Петра Могили]. Сер. : Комп'ютерні технології. – 2014. – С. 63-68.

120. Мусієнко М.П. Розподіл функцій між керуючими пристроями навігаційно-телекомунікаційних гетерогенних мультипроцесорних систем високопотокової передачі даних / Мусієнко М.П., Крайник Я.М. // Наукові праці [Чорноморського державного університету імені Петра Могили]. Сер. : Комп'ютерні технології. – 2013. – Т.229, Вип. 217. – С. 59-63. – ISSN: 1609-7742.

121. Мусиенко М.П. Оптимальное распараллеливание задач диспетчеризации в распределенных энергоограниченных компьютерных системах на базе MCU / FPGA модулей / Мусиенко М.П., Савинов В.Ю., Крайнык Я.М. // Вектор науки Тольяттинского государственного университета. – №1(27). – 2014. – С. 18-21. – ISSN: 2073-5073.

122. Крайник Я.М. Інтеграція інтелектуальних модулів на базі ПЛІС/мікропроцесор / Я.М. Крайник / Сучасні інформаційні та електронні

технології : труди XV міжнародної науково практичної конференції, Одеса, 26 30 травня 2014 р. – О., Политехпериодика, 2014. – С. 64 65. – ISSN: 2308 8060.

123. Крайнык Я.М. Поиск оптимального распределения организации вычислений для системы MCU/FPGA / Я.М. Крайнык / Проблемы информатизации : тезисы докладов I международной научно практической конференции, Черкассы-Киев-Тольятти-Полтава, 19-20 декабря 2013 г. – 2013. – С. 19.

124. STMicroelectronics (2013), RM0090 Reference manual, Available at: http://www.st.com/web/en/resource/technical/document/reference_manual/ DM00031020.pdf (accessed January 2013).

125. Saaty T. (1990), “How to make a decision: The Analytic Hierarchy Process”, European Journal of Operational Research, No 48, 1990, pp. 9-26.

126. Крайник Я.М. Створення систем управління GSM/GPS/MCU пристроями на базі USB / Крайник Я.М. // Технологический аудит и резервы производства. – 2013. – №6/4(14). – С. 13-15. – ISSN: 2226-3780.

127. Крайник Я.М., Мусієнко М.П. Інтеграція інтерфейсу USB у MCU-системи / Крайник Я.М., Мусієнко М.П. / Комп'ютерні науки для інформаційного суспільства: Матеріали IV Міжнародної науково практичної конференції студентів, аспірантів та молодих вчених (веб-конференція), Луганськ, 11-12 грудня 2013 р. – Луганськ: Вид-во Ноулідж, 2013. – С. 109-111. – ISSN: 2306-7233.

128. Savchhuk, O. L., Musiyenko, M. P., (2007). *Ukrainian Patent No. 27895.*

129. Farnell (2013), Accessories, available at: http://uk.farnell.com/ftdi/ft2232hq-mini-module/mod-usb-to-serial-fifo-ft2232h/dp/1697465 (accessed November 16, 2013).

130. Cypress (2013), Pricing and inventorying availabilit, available at: http://www.cypress.com/?mpn=CY7C68001-56LTXC (accessed November 19, 2013).

131. Android 4.0 APIs (2013), available at: http://developer.android.com/about/versions/android-4.0.html (accessed November 23, 2013).

132. Google Maps Android API v2 (2013), available at: https://developers.google.com/maps/documentation/android/ (accessed November 23, 2013).

133. NMEA 0183 (2013), available at: http://en.wikipedia.org/wiki/NMEA_0183 (accessed December 1, 2013).

134. NMEA data (2013), available at: http://www.gpsinformation.org/dale/nmea.htm (accessed December 1, 2013).

135. Мусиенко М.П. Повышение быстродействия позиционирования экструдера 3D-принтера с использованием MCU/FPGA / Мусиенко М.П., Бугаев В.И., Крайнык Я.М., Денисов А.О. // Технические науки – от теории к практике, Новосибирск. – № 12 (25). – 2013. – С. 33-36.

136. Крайник Я.М. Підвищення точності позиціонування екструдеру 3D-принтеру при забезпеченні профілю швидкості крокового двигуна / Я.М. Крайник / Автоматизація та комп'ютерно-інтегровані технології у виробництві та освіті: стан, досягнення, перспективи розвитку: матеріали Всеукраїнської науково-практичної Internet-конференції, Черкаси, 17 21 березня 2014 р. – Черкаси, 2014. – С. 44-46.

137. Крайник Я.М. Модифікація алгоритму управління швидкістю крокових двигунів з реалізацією на базі ПЛІС / Я.М. Крайник / Сучасні інформаційні технології 2014 (МІТ 2014)/ Матеріали четвертої Міжнародної конференції студентів і молодих науковців, 22 26 квітня 2014 р. – Одеса, ТЕС, 2014. – С. 117 118. – ISBN-978-617-7054-28-2.

138. Крайник Я.М. Зменшення використання ресурсів ПЛІС при реалізації алгоритму управління кроковими двигунами / Я.М. Крайник / ІІІ Міжнародна науково-практична конференція «Напівпровідникові матеріали, інформаційні технології та фотовольтаїка»: Тези доповідей, 20 23 травня 2014

р, Кременчук. − Кременчук: Кременчуцький національний університет імені Михайла Остроградського, 2014. – ISSN 2222-4386.

139. Мусиенко М.П. К вопросу построения геометрических примитивов в интерполяторе 3D-принтера с управлением на основе FPGA / Мусиенко М.П., Крайнык Я.М., Денисов А.О., Бугаев В.И. // Universum: Технические науки : электрон. научн. журн. – 2014. – №4(5). – ISSN: 2311-5122.

140. Денисов А. Применение FPGA и алгоритмов Брезенхема для повышения быстродействия в системах позиционирования. // Компоненты и технологии. – №10. – 2013. – с. 96-100.

141. RepRap [Електронний ресурс]. Режим доступу. – URL: http://reprap.org. – Загол. з екрану.

142. RepRap Options [электронный ресурс] — Режим доступа. — URL: http://reprap.org/wiki/RepRap_Options#The_controller (дата обращения 25.11.2013).

143. Pirated CupCake [электронный ресурс] — Режим доступа. — URL: http://reprap.org/wiki/Pirated_CupCake (дата обращения 25.11.2013).

144. Reprap host software [электронный ресурс] — Режим доступа. — URL: http://reprap.org/wiki/Reprap_host_software (дата обращения 25.11.2013).

145. Software for Low Density Parity Check Codes [Електронний ресурс]. Режим доступу : URL : http://www.cs.utoronto.ca/pub/radford/LDPC-2000-03-19/index.html. – Загол. з екрану.

146. Freisen, J. (2011), Beginning Java 7, Apress.

147. Horstmann, C. (2010), Big Java Compatible with Java 5, 6 & 7, 4th Edition, John Wiley & Sons Inc.

148. Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя. 2-е изд.: Пер. с англ. Мухин Н. – М.: ДМК Пресс, 2006.

149. Zynq-7000 All Programmable SoC: Concepts, Tools, and Techniques (CTT) A Hands-On Guide to Effective Embedded System Design, UG873 (v14.3), Xilinx, October 16, 2012.

150. Тарасов И. Расширяемая процессорная платформа семейства Zynq-7000. // Компоненты и технологии, №4, 2011. – 88-92 С.

151. Crockett, L.H., Elliot, R.A., Enderwitz, M.A., Stewart, R.W. (2014), The Zynq Book Embedded Processing with the ARM® Cortex®-A9 on the Xilinx Zynq-7000 All Programmable SoC, First Edition, Strathclyde Academic Media.

152. Crockett, L.H., Elliot, R.A., Enderwitz, M.A., and Stewart, R.W. (2014), The Zynq Book Tutorials, Strathclyde Academic Media.

153. Hwang, E.O. (2004), Microprocessor Design Principles and Practices With VHDL, Brooks/Cole.

154. Mealy, B., and Tappero, F. (2013), Free Range VHDL, available at: freerangefactory.org, p. 190.

155. Pedroni, V.A. (2010), Circuit Design And Simulation with VHDL, Massachusetts Institute of Technology, 2nd ed.

156. Самофалов К.Г., Романкевич А.М., Валуйский В.Н., Каневский Ю.С. Пиневич М.М. Прикладная теория цифровых автоматов. - К.: Вища шк. Головное изд-во, 1987. - 375 с.

157. Савельев А.Я. Прикладная теория цифровых автоматов: Учеб. для вузов по спец. ЭВМ. – М.: Высш. шк., 1987. – 272 с.: ил.

158. Савельев А.Я. Арифметические и логические основы цифровых автоматов: Учебник. – М.: Высш. школа, 1980. – 255 с., ил.

159. Bergeron, J. Writing Testbenches Functional Verification of HDL Models, 2nd ed., Kluwer Academic Publishers, 2003.

160. Рихтер Д. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. – Питер, 2015. – 896 С.

161. Петцольд Ч. Microsoft Windows Presentation Foundation: - М.: Издательство "Русская Редакция"; СПб.: Питер, 2008. - 944 с.

162. Garofalo, R. (2011), Applied WPF 4 in Context, Apress.

Додаток А. Акти впровадження результатів дисертаційної роботи

РЕЛСіС®
ПАТ Електротехнічний завод
Релейні Схеми і Системи

П/р 26006001000839 в ПАТ "БТА Банк" м. Київ
МФО 321723, код ЄДРПОУ 22965117

СИСТЕМА МЕНЕДЖМЕНТУ ЯКОСТІ ISO 9001:2008

03680, Україна, м. Київ
вул. Сім'ї Сосніних, 9
тел.: (044) 406-61-00
факс: (044) 407-36-77
e-mail: Office@relsis.ua
web: www.relsis.ua

Вих №06/1900 від «25» січня 2015 р

АКТ

про впровадження результатів дисертаційної роботи

Крайника Ярослава Михайловича

ПАТ «Електротехнічний завод» займається розробкою засобів релейного захисту. На сьогоднішній день проблема побудови надійних та швидкодіючих мікропроцесорних систем релейного захисту, які розробляються ПАТ «Электротехнический завод», є актуальною. Тому розроблені у дисертації Крайника Я.М. моделі та методи реалізації LDPC-декодерів у частині побудови частково паралельних LDPC-декодерів можуть бути використані в наступних розробках ПАТ «Електротехнічний завод»:

– при моделюванні та тестуванні мікропроцесорних систем релейного захисту;

– при розробці перспективних зразків мікропроцесорних систем релейного захисту з високими показниками надійності та швидкості обробки інформації.

Результати тестування показали, що розроблений за результатами досліджень Крайника Я.М. LDPC-декодер може забезпечувати надійну роботу системи при співвідношенні сигнал/шум 6 дБ, що підвищує надійність при прийомі інформації на 40 %, а також збільшити кількість оброблюваної інформації у 2 рази.

Питання про впровадження LDPC-декодерів у серійний випуск та обсяги виробництва буде вирішено після завершення маркетингових досліджень та формування пакету замовлень.

Генеральний директор *В.В.Бучнев*

м. Київ * Україна * Публічне товариство "ЕЛЕКТРОТЕХНІЧНИЙ ЗАВОД" Ідентифікаційний код 22965117

IMPLEMENTATION ACT
for dissertation work of
Yaroslav M. Krainyk
in Straight Way Electronix s.r.o.

Straight Way Electronix s.r.o. is a FPGA SoC design office. Nowadays High Speed System for Radiorelay links that is developed by Straight Way Electronix s.r.o. is topical and valuable. Development of this system requires implementation of reliable and high-throughput protocols of information exchange. Thus, models and methods for partially parallel LDPC-decoder implementation in FPGA that are developed in dissertation of Y. Krainyk could be used in such products of Straight Way Electronix s.r.o.:

- High Speed System for Radiorelay links;
- LDPC Core for Satellite communication.

Implementation and testing results confirm that LDPC-decoder developed by investigation of Y. Krainyk provides reliable work of High Speed System for Radiorelay links with signal-to-noise ratio at 5 dB and guarantees 50 Mbps throughput value for implementation of specific information exchange protocol.

CEO of Straight Way Electronix s.r.o Motspan Viktor

STRAIGHT WAY ELECTRONIX s.r.o.
Jeseniova 1151/55
130 00 Praha 3 - Žižkov
IČ: 01894340, DIČ: CZ01894340

µPC Labs, Inc.

1 Strawberry Hill Ave Suite 6B, Stamford, CT 06902, USA

September 24, 2015

Subject: Yaroslav M. Krainyk's research

To Whom It May Concern:

uPC Labs, Inc. is a company producing micro-miniature PCs. Micro-miniature computers require reliable and high-throughput protocols of information exchange. Thus, models and methods for partially parallel pipeline LDPC-decoder implementation in FPGA that were developed by Y. Krainyk could be used in such products of uPC Labs, Inc.:

- µPC™ to ensure the security of personal data;
- µTV™ for secure remote connections.

Implementation and testing results confirm that pipeline LDPC-decoder developed by Y. Krainyk provides reliable work of micro-miniature computers with signal-to-noise ratio value at 4.5 dB and guarantees 70 Mbps throughput with at least 10 decoding iterations for implementation of specific reliable information exchange protocol.

CEO and President A. Podelko

ЗАТВЕРДЖУЮ
Ректор Черкаського національного
університету ім. Б.Хмельницького,
д.е.н., професор

О.В. Черевко

«___»_____2015 р.

ДОВІДКА
про впровадження в навчальний процес результатів дисертаційної роботи
Крайника Ярослава Михайловича

Основні результати роботи Крайника Я.М. застосовуються при викладанні дисципліни «Комп'ютерна техніка і організація обчислювальних робіт» студентам напряму підготовки 6.050202 – Автоматизація та комп'ютерно-інтегровані технології.

До програми лекційного курсу введені такі теми, які містять матеріал роботи здобувача (нумерація лекцій згідно з робочою програмою):

- лекція № 12: «Організація паралельних обчислень в КС»;
- лекція № 14: «Навігаційно-комунікаційні системи».

До лекційного курсу включено такі результати, отримані автором, а саме: модель організації паралельних черг запису/зчитування та функціональний розподіл між модулями в навігаційно-комунікаційних системах.

Матеріал, отриманий Крайником Я.М., дозволяє студентам ознайомитися з новими методами організації комунікації та паралельних обчислень.

Зав. кафедри автоматизації
та комп'ютерно-інтегрованих технологій
к.т.н., доцент Г. В. Гриценко

Директор
ННІ фізики, математики та КІС
д. ф.-м. н, доцент Ю. О.Ляшенко

ЗАТВЕРДЖУЮ
Ректор Чорноморського державного
університету ім. П.Могили,
д.т.н., професор

Л.П.Клименко
«13» жовтня 2015 р.

ДОВІДКА
про впровадження в учбовий процес результатів дисертаційної роботи Крайника Ярослава Михайловича

Основні результати дисертаційної роботи Крайника Я.М. застосовуються при викладанні дисципліни «Телекомунікаційні системи» студентам спеціальностей 7.05010202, 8.05010202 – системне програмування.

До програми лекційного курсу введені такі теми, які містять матеріал роботи здобувача (нумерація лекцій згідно з робочою програмою):

- лекція № 7: «Завадостійке кодування при передачі інформації»;
- лекція № 8: «Завадостійке кодування LDPC».

До лекційного курсу включено такі результати, отримані автором, а саме: методи побудови та моделі організації роботи частково паралельних LDPC-декодерів.

Матеріал, отриманий Крайником Я.М., дозволяє магістрантам та студентам ознайомитися з новими методами організації комунікації за наявності шумів у каналі передачі даних.

На основі отриманих матеріалів розроблено завдання для практичних занять на тему: «Завадостійке кодування. Жорстке декодування LDPC-кодів. Декодування з використанням алгоритму мінімальної суми». Виконання практичного заняття забезпечує ознайомлення студентів з новими методами декодування завадостійких кодів LDPC.

Заступник декана
факультету комп'ютерних наук,
к.т.н., ст.викл. Є. В. Сіденко

ЗАТВЕРДЖУЮ

В.о. ректора Черкаського державного технологічного університету

д.е.н., доцент С.А. Назаренко

«____»_____2015 р.

ДОВІДКА
про впровадження в навчальний процес результатів дисертаційної роботи
Крайника Ярослава Михайловича

Основні результати дисертаційної роботи Крайника Я.М. застосовуються при викладанні дисципліни «Основи теорії інформації та кодування» студентам напряму підготовки 6.050101 – Комп'ютерна інженерія.

До програми лекційного курсу введені такі теми, які містять матеріал роботи здобувача (нумерація лекцій згідно з робочою програмою):

- лекція № 8: «Завадостійке кодування в системах передачі інформації»;
- лекція № 9: «Організація декодування даних з використанням завадостійкого кодування».

До лекційного курсу включено такі результати, отримані автором, а саме: модель організації паралельних для алгоритму мінімальної суми модифікований алгоритм декодування мінімальної суми.

Матеріал, отриманий Крайником Я.М., дозволяє студентам ознайомитися з новими методами організації комунікації та паралельних обчислень.

Зав. кафедри інформаційної безпеки
та комп'ютерної інженерії
д.т.н., професор К. В. Рудницький

Додаток Б. Моделювання роботи декодеру у середовищі ModelSim

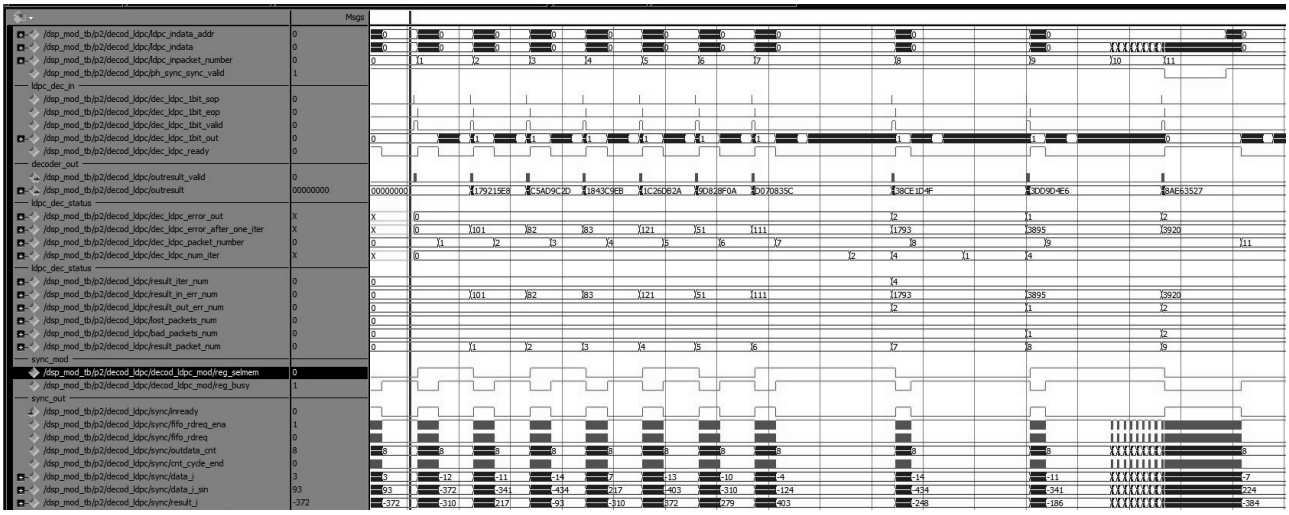


Рис. Б1. Моделювання роботи декодеру з двома блоками обробки

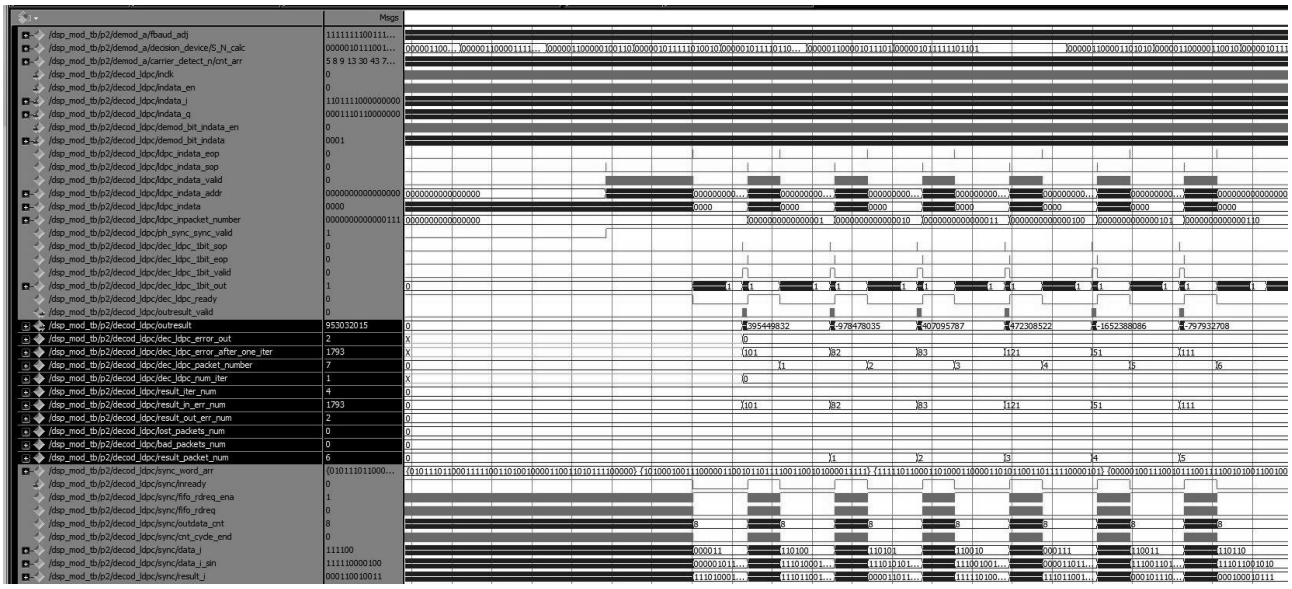


Рис. Б2. Моделювання роботи декодеру з подвійним буфером

Рис. Б3. Моделювання роботи декодеру з конвеєрною архітектурою

Додаток В. Матриці перевірки парності, що використовувались для перевірки роботи розроблених декодерів

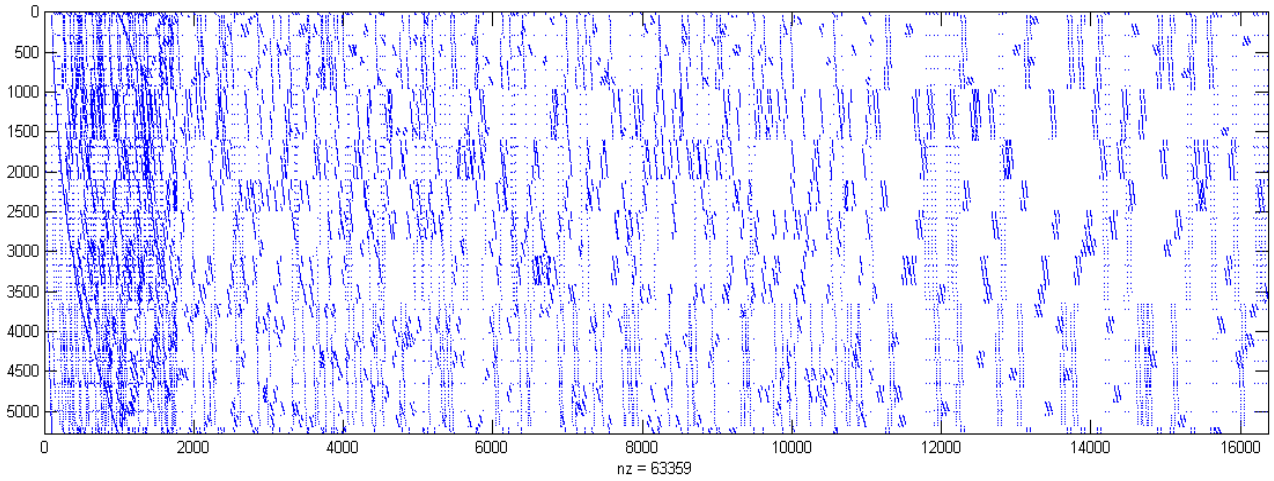


Рис. В1. Матриця перевірки парності для перевірки роботи декодеру

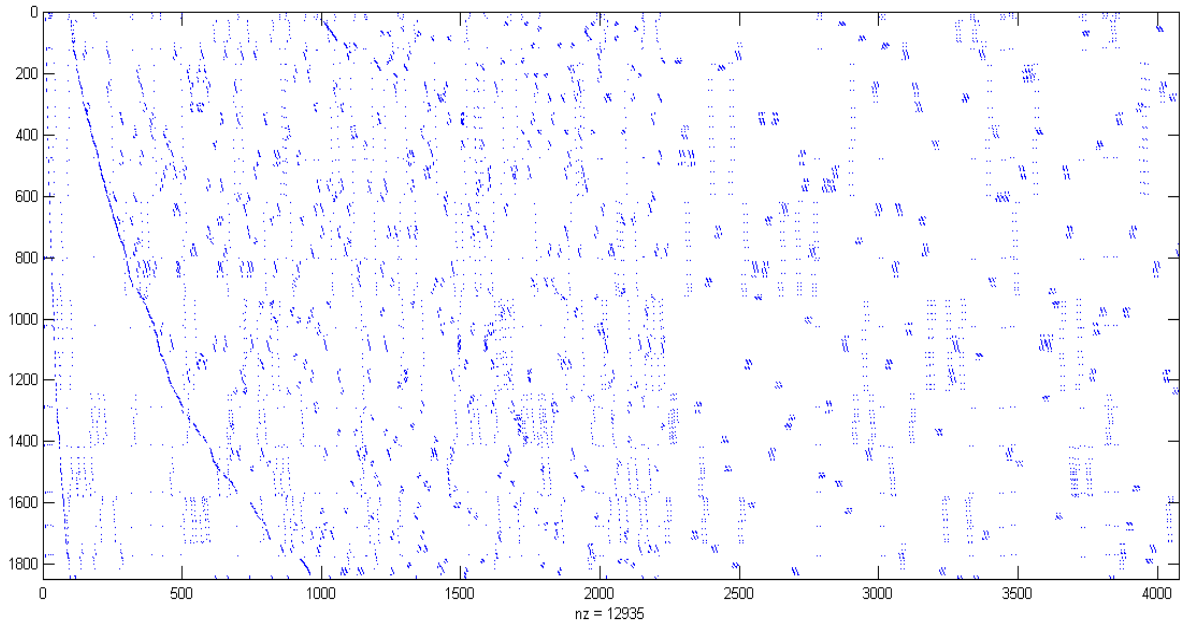


Рис. В2. Матриця перевірки парності для перевірки роботи декодеру