Хмельницький національний університет
Факультет інформаційних технолоігй
Кафедра комп’ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Програмно-технічний модуль завадозахищеного кодування даних на базі FPGA
Назва теми

КвРКІ. 190181.08.01.08 ПЗ
Шифр

Галузь знань 12 «Інформаційні технології»
Шифр, назва

Спеціальність 123 «Комп’ютерна інженерія»
Шифр, назва

Освітня програма «Комп’ютерна інженерія»
Назва

Виконав: студент ІІІ курсу, група КІ2с–19–1 _____ В.Є Голубович
Підпис Ініціали, прізвище

Керівник _____ В. В. Яцків
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ С.М. Лисенко
Підпис, дата Ініціали, прізвище

До захисту допускаю:
Зав. кафедри комп’ютерної інженерії та системного програмування _____ Т.О. Говорущенко
Підпис Ініціали, прізвище

«____» _____ 2022 р.

Хмельницький 2022

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень бакалавр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма освітня програма «комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Зав. кафедри __Т.О.Говорущенко___

_____

" __" _____202__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

_____Голубовичу Володимиру Євгеновичу_____
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) _Програмно-технічний модуль завадозахищеного кодування даних на базі FPGA

Керівник проекту (роботи) __Яцків В.В._____
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.03.2022 р. № 18

2. Строк подання студентом проекту (роботи) на кафедру 07.06.2022 р.

3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)_____

__Дослідження предметної області та постановка задачі_____

__Проєктування модулю завадозахищеного кодування даних_____

__Розробка модулю завадозахищеного кодування даних_____

-_____

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) _____

Використання пакету MRTG для аналізу трафіку_____

Розміщення мережі в приміщеннях_____

Структура локальної мережі_____

_____

_____

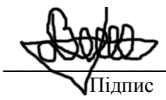
6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Лисенко С.М., професор кафедри КІСП		
Антиплагіат	Нічепорук А.О., доцент кафедри КІСП		

7. Дата видачі завдання «_11_»_____01_____2022 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	11.01.2022	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об’єкта та предмета дослідження	01.02.2022	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2022	виконано
4	Робота над розділом 2 – моделювання пристрою завадостійкого кодування на базі FPGA	01.04.2022	виконано
5	Робота над розділом 3 – апаратна реалізація коду Ріда-Соломона	30.04.2022	виконано
6	Оформлення пояснювальної записки згідно вимог	20.05.2022	виконано
7	Попередній захист ВКР	24.05.2022	виконано
8	Захист ВКР на засіданні ЕК	Червень 2022 року	

Студент  В.Є. Голубович
Підпис Ініціали, прізвище

Керівник проекту (роботи)  В. В. Яцків
Підпис Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 190178.05.03.10 ПЗ	Пояснювальна записка	67		
			Графічні матеріали			
2		КвРКІ 190181.05.03.10 Е8	Схема кодера	1		
3		КвРКІ 190181.05.03.10 Е8	Схема декодера	1		
4		КвРКІ 190181.05.03.10 Е8	Робота системи з корекцією помилок	1		

Зм	Арк	№ докум	Підпис	Дата	КвРКІ 190181.8.01.8 ВП			
Розробив		Голубович			Відомість проекту	Літера	Аркуш	Аркушів
Перевір.		Яцків				У	1	1
Н. контр.		Лисенко				ХНУ, КІ2с-19-1		
Затв.		Говорущенко						

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Програмно-технічний модуль завадозахищеного кодування даних на базі FPGA».

Автор роботи: Голубович Володимир Євгенович

Керівник роботи: Яцків Василь Васильович.

Пояснювальна записка: 67 с., 20 рис., 3 дод., 29 джерел.

Графічна частина: 7 презентаційних слайдів.

ЗАВАДОЗАХИЩЕНЕ КОДУВАННЯ. КОД РІДА СОЛОМОНА. FPGA

Метою роботи є розробка пристрою завадозахищеного кодування на базі FPGA.

Предметом дослідження є процес проектування та моделювання роботи коду Ріда-Соломона на базі FPGA.

Практичне значення отримав спроектований пристрій, що може бути реалізований для завадостійкої передачі даних.

Підпис студента

Дата

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

РС - код Ріда-Соломона

FPGA- field-programmable gate array

ЛПС - лінійні послідовні системи

ЕОМ - електро́нна обчи́слювальна маши́на

ВОК – Волоконно-оптичний кабель

ВСТУП

З тих пір, як з’явились обчислювальні системи, гостро стоїть питання надійності зберігання і передачі інформації. З плином часу, людство дедалі інтенсивніше для реалізації своїх потреб використовує обчислювальні системи різного призначення. Галузі застосування обчислювальних систем відрізняються в залежності від особливостей поставлених задач. Поняття «обчислювальні системи» доволі широке. До нього можна віднести складні мікропроцесорні кластери, різні засоби дистанційної передачі інформації.

Обчислювальні системи використовуються у великій кількості галузей, до яких відносяться побут, наукова діяльність, міжвідомчі повідомлення, промисловість, оборонний комплекс. Чим важливіша сфера використання, тим вище рівень вимог, які висуваються до обчислювальних систем..

Пам'ять комп'ютера іноді може робити помилки через сплески напруги на лінії електропередачі та з інших причин. Передача інформації також пов'язана з різними помилками.

Одним зі способів підвищення надійності інформації, що передається, або зберігається, є завадостійке кодування. Сутність такого кодування полягає у додаванні до вихідних послідовностей, що кодуються, контрольних символів. Дані символи призначено для використання при декодуванні даних – вони допомагають перевірити цілісність повідомлення або знайти та виправити помилки.

Фундамент завадостійкого кодування був закладений ще на початку другої половини двадцятого віку. Вагомий вклад у розвиток даної сфери внесли американські математики Клод Шеннон та Річард Геммінг. Дана тема, як і півстоліття назад, є актуальною. Завадостійке кодування застосовується при архівації даних, передаванні даних у пакетних мережах, у супутникових та стільникових системах зв’язку.

На сьогоднішній день відомо безліч кодів та методів їх декодування, які відрізняються складністю реалізації та рядом інших параметрів.

Різноманітність існуючих кодів поділяється на два класи: блокові коди та безперервні коди.

У блокових кодах передана інформаційна послідовність розбивається на окремі блоки з додаванням до кожного блоку певного числа перевірочних символів. Кодові комбінації кодуються та декодуються незалежно друг від друга.

У безперервних кодах інформаційна послідовність, що передається, не поділяється на блоки, а перевірочні символи розміщуються у певному порядку між інформаційними. Процеси кодування та декодування також здійснюються у безперервному режимі.

При фіксованій довжині коду (n) та кількості розрядів (k) не існує коду, у якого мінімальна відстань більше, ніж у коду Ріда-Соломона. Цей факт являється вагомою підставою для використання даного методу завадостійкого кодування.

Тематикою даної роботи є розробка кодера, декодера, основаного на алгоритмі роботи кода Ріда-Соломона, що знаходить двухпакетні та виправляє однопакетні помилки, що виникають при передачі даних.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Визначення завадостійкого кодування

Кодування сигналу - його подання у певному вигляді, зручному або придатному для обміну повідомленнями. У процесі кодування переданих елементів повідомлення перетворюються на відповідні кодові символи. На приклад, щоб передати 33 літери алфавіту потрібно передавати цифри від 0 до 32. Щоб перейти будь-яке число, записане у десятковій формі, потріьно десять цифр від 0 до 9 (у цьому випадку потрібно десять основних сигналів). Реалізація кодування на стороні передачі завжди передбачає використання зворотного процедури - декодування - для відновлення повідомлень за отриманими кодами символів.

Код - повний набір символів, що використовуються для кодування повідомлення. Відношення символів (або комбінації символів, якщо закодовані не окремі символи) вихідного алфавіту з їх кодовими комбінаціями становлять таблицю відповідності (або кодову таблицю). Множину можливих кодових символів (елементарні сигнали) називають кодовим алфавітом, а їх кількість m - основою кода.

Код з основою $m = 2$ називають бінарним, з іншими основаии - багатопозиційними.

При основі m, правила кодування K елементів повідомлення зводяться до правил запису K різних чисел у системі числення з основою m.

Найпростіші електричні сигнали, що утворюють символи під час кодування повідомлень, - це імпульси струму або напруги. Під час кодування кожен елемент повідомлення записується за допомогою певного кодового набору символів, який

називається кодовою комбінацією. Кількість бітів n, з яких складається кодова комбінація, називається розрядністю коду або довжиною кодової комбінації.

Для кодування вихідного повідомлення, потрібно встановити деякі правила. Сукупність правил кодування об'єктів називається системою кодування. Така система може бути виражена по-різному.

Часто дуже зручною системою кодування є кодова таблиця, яка містить алфавіт кодованих повідомлень і відповідні кодові комбінації.

Двійковий код, який має особливе значення в техніці, найчастіше використовується при кодуванні дискретних повідомлень.

Це пояснюється тим, що наявність і відсутність посилки найбільш впевнено помітні. Тим більше, достатньо просто створити пристрій, що може перебувати в двох можливих станах (відкрито - закрито).

Зокрема, пристрій з двома постійними станами типу електронного перемикача або тригера, здатного зберігати один біт інформації, N таких пристроїв - N біт. Через це двійковий код використовується не тільки в системах зв’язку, а й у комп’ютерах, автоматизації тощо.

Необхідна кількість бітів для кодування повідомлення при заданій максимальній кількості рівнів масштабу квантування U_{max} визначається співвідношенням $n = \log_2 N_{max}$.

При умові, що кодова група містить n символів 0 і 1, числа до $N_{max} = 2^n$ можуть бути закодовані за допомогою аналогічного n-бітового двійкового коду.

При умові, що всі кодові слова мають однакову кількість символів, код називається однорідним, інакше - нерівномірним.

Довжина кодової комбінації залежить від кількості окремих елементів, які вона містить. Рівномірний код з n = 3 називається трибітним кодом, з n = 5 — п’ятирозрядним або п’ятиелементним кодом тощо.

Код Бодо (рисунок 1.1) (m = 2, n = 5) — це код, елементами якого є сигнал і відсутність сигналу з однаковою тривалістю та абсолютним значенням, але з різною полярністю.

Код Бодо однорідний, кожна літера відповідає п'яти елементарним символам. Всього можна скласти N = 25 = 32 комбінації.

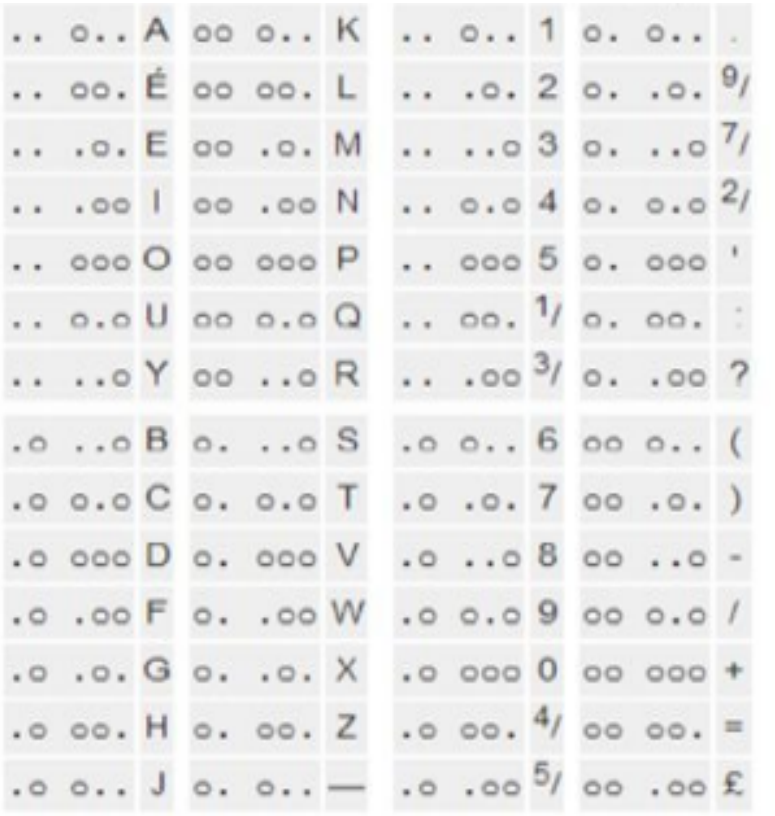


Рисунок 1.1 - Код Бодо

Другий регістр із 32 символів використовується для збільшення кількості символів у коді Бодо.

Основним і відомим прикладом нерівномірних кодів є азбука Морзе (рисунок 1.2), в якій кодові слова мають різну довжину: найпоширенішим символам присвоюються найкоротші символи коду і навпаки.

В азбуці Морзе символи 1 і 0 використовуються тільки в двох комбінаціях - одиночній (1 і 0) або потрійній (111 і 000).

Сигнал, що кодується однією одиницею (поточний короткий пакет), відповідає точці (.), трьом одиницям (поточний пакет утричі довший) - тире (відсутність струму або пауза).

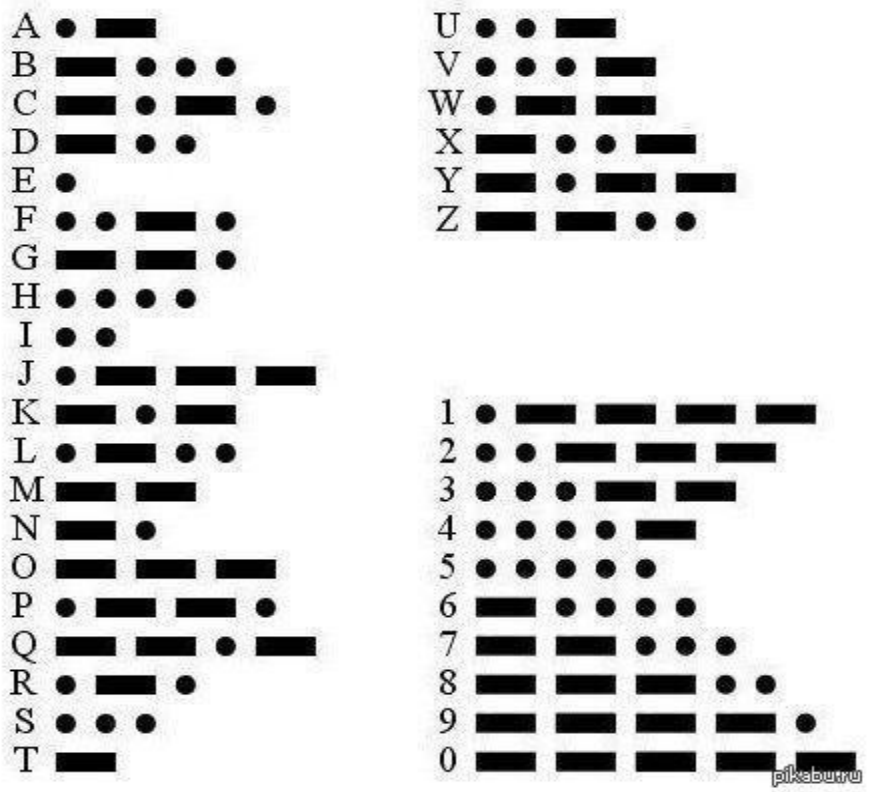


Рисунок 1.2 - Код Морзе

Символ 0 відокремлює крапку від тире, крапку від крапки та тире від тире.

Набір символів 000 використовується як розділовий знак між комбінаціями кодів.

Крім того, для розділення слів використовується набір символів 00000. Азбука Морзе заснована на англійській статистиці.

Слід зауважити, що азбука Морзе – це не код із двійковим введенням (крапка та тире), як здається на перший погляд, а потрійний алфавіт: крапка, тире та пробіл.

У радіозв’язку азбукою Морзе передавач надсилає радіоімпульси різної довжини, що відповідають точкам і тире кодової комбінації.

У приймачі спеціальний генератор і змішувач перетворюють радіоімпульси в ділянки коливань звукової частоти.

Оператор може обрати найбільш зручну для себе частоту звукових коливань за допомогою зміни частоти коливань генератора

При слуховому телеграфному зв’язку швидкість передачі інформації може коливатись в залежності від кваліфікації оператора та зазвичай становить приблизносто занків на хвилину.

Серед переваг коду Морзе необхідно виділити високу захищеність від завад прийому на слух, дякуючи чому він широко застосовується у радіозв'язку.

Перетворенням сигналу при прийомі на слух займається людина, що володіє певним досвідом, може відрізняти сигнали, що були значно уражені завадами.

У теорії інформації доведено, що як для неортогонального, так і для ортогонального сигналів значення ймовірності помилки p при оптимальному прийомі однаково визначається кількістю символів m і параметром відношення сигналу до шуму $Q_{вих0} = E_0 / W_0$, де E0 — енергія сигнального елемента; W_0 – спектральна щільність потужності білого шуму. З цього можна зробити висновок, що необмежене збільшення точності передачі повідомлень (тобто зниження ймовірності помилки p до нуля) можна отримати лише за рахунок необмеженого збільшення енергетичного параметра $Q_{вих0}$ (за рахунок збільшення сили сигналу).

Зокрема, якщо ви встановлено середню потужність сигналу P_c і спектральну щільність потужності білого шуму W_0, це може статися, лише прагенні до нуля швидкості передачі інформації v, оскільки $E_0 = P_c T_s$ і тривалість сигналу $T_s = l / v$.

К. Шенноном було доведено, що безпомилкова передача сигналів принципово можлива за умови, що швидкість передачі менше пропускної

здатності каналу зв'язку, котра тим більше, чим більше відношення сигналу до шуму на вході приймального пристрою

З цього слідує, що енергетикою ліній зв’язку визначається лише їх пропускна здатність, а за допомогою застосування спеціально побудованих кодів можна забезпечити як завгодно високу завадостійкість.

Революційні для тих часів ідеї Шеннона здійснили переворот у свідомості зв'язківців, тому що вважалося, що єдиними можливостями підвищення завадостійкості є збільшення потужності передавача або багаторазова передачі по одного і того самого повідомлення. Але ці методи ведуть до дуже неефективного використання пропускної здатності каналу зв'язку.

Фактично Шеннон стверджував, що потужністю сигналу, шумом каналі і смугою частот обмежується лише швидкість передачі, а не точність.

Шенноном було знайдено необхідні і достатні умови зменшення ймовірності помилки до нуля. Крім того, це було зроблено для різних моделей каналів зв'язку, зокрема з обмеженою смугою частот, для сталого числа форм сигналів, для надлишкових джерел, для джерел безперервних повідомлень, для перевірки вірності отримання окремих повідомлень, відмінних від імовірності помилки символу p, і т. д.

В цьому і полягає сутність теорії інформації і теорії кодування.

Однак К. Шенноном не було вказано, як можуть бути знайдені потрібні коди, а лише доведено, що вони існують .

1.2 Класифікація методів кодування

Методи кодування, які використовують в техніці зв’язку та коди можуть бути класифіковані за набором специфічних ознак.

У наведеній класифікації відображено коди, що використовуються (Рисунок 3.1).

Кодування за його призначенням можна поділити на примітивне, економне та завадостійке.

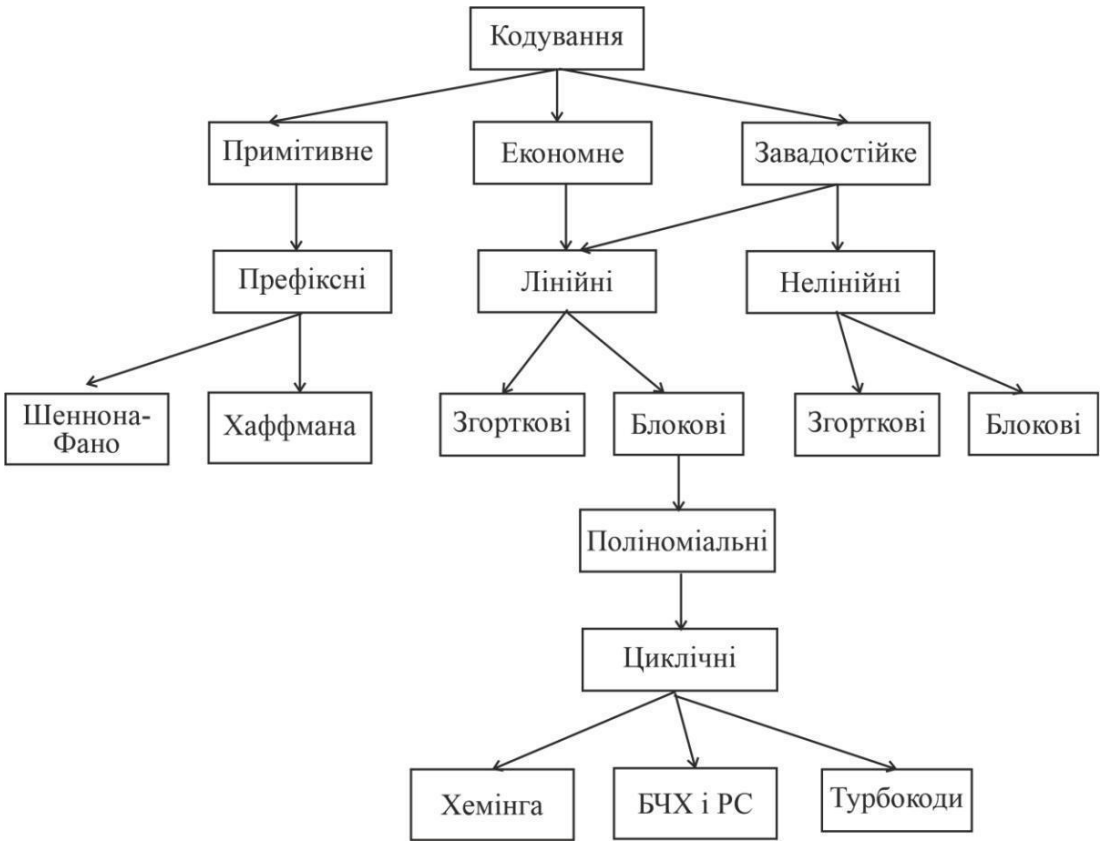


Рисунок 1.3 - Класифікація методів кодування

За допомогою кодування вирішують наступні завдання:

- узгодження алфавіта джерела і алфавіта канала - примітивне кодування;

- зменшення об’єму інформації при передачі - економне (ефективне) кодування;

- знаходження або виправлення помилок, шо виникають в каналі зв'язку - завадостійке кодування.

Завадостійке кодування часто називають канальним кодуванням.

Примітивне або безнадлишкове кодування узгоджує джерело алфавіту з алфавітом дискретного каналу, а також ряд випадків, що використовуються для

шифрування переданої інформації, для її захисту від несанкціонованого доступу, а також для підвищення стійкості роботи системи синхронізації зв'язку. Термін «примітвне» — теоретико-числове поняття, потребуюче алгебраїчного розгляду.

Відмінна особливість примітивного кодування полягає в тому, що надлишковість дискретного джерела, утвореного виходом примітивного кодера, рівна надлишковості джерела на вході кодера. Спрощенно можна сказати, що при примітивному кодуванні символ одного алфавіту замінюють на символ іншого алфавіту.

Первинні коди, отримані при примітивному кодуванні, характеризуються тим, що окремі його кодові комбінації можуть відрізнятися від іншого лише одним елементом. Тому навіть один помилково прийнятий елемент у кодовій комбінації призводить до заміни однієї кодової комбінації іншої і, дослідно, до неправильного прийому кодової комбінації в цілому.

Шифрування (при примітивному кодуванні), яке використовується для забезпечення секретності зв'язку, повертає повідомлення несанкціонованим користувачам і введення в систему хибних повідомлень. У цьому випадку правило примітивного кодування вибирається так, щоб імовірно, що поява на вихідному кодері довгої послідовності, що складається тільки з одиниць або тільки з нулів, була мінімальною. Подібний кодер називають скремблером (від англ, scramble — перемішувати).

Скремблювання має ціль придати потоку символів властивості, схожі на випадкову послідовність рівноймовірних і незалежних двоїчних символів. Скремблювання здійснюють, пропускаючи потік символів через реєстр зсуву зі зворотними зв'язками, в якому ці символи «перемішуються» і перетворюються. Звичайно, характеристика скремблера повинна бути зворотньою. Тоді прийняті після демодулятора символи піддаються зворотній операції — дескремблированию — для відновлення вихідної послідовності. Скремблювання

часто застосовують для підвищення надійності системи синхронизації. Але це також покращує роботу декодера, особливо в системах із зворотною зв'язком.

1.3 Завадостійкі коди

Будь-який канал зв'язку з обмеженими кількістю смуг, частот, часом передачі і динамічним діапазоном має кінцеву пропускну здатність. В теорії пропускна здатність - це найбільша кількість біт за одиницю часу при якмога малій ймовірності помилок.

Фактична кількість бітів, що передаються за одиницю часу, називається швидкістю передачі.

З необмеженою ймовірністю помилки швидкість передачі завжди менша за пропускну здатність.

У каналі з помилками максимальне значення швидкості можна отримати за допомогою використання завадозахищеного кодування. Воно вимагає введення надмірності в переданий сигнал: часу, частоти або амплітуди.

При умові, що код сумісний з каналом, тобто код дозволяє виправити найбільш вірогідні помилки, введена надмірність є виправданою.

Якщо код не відповідає каналу, помилки можна не тільки не виправити, але і розмножити кодом. У цьому випадку використання завадозахищеного кодування не принесе користі, а зашкодить. Щоб узгодити код з каналом зв'язку, необхідно мати максимальну кількість інформації про можливі перешкоди в каналах.

На даний момент було розроблено багато різних завадостійких кодів, які різняться між собою основою, відстанню, надлишковостю, структурою, функціональним призначенням, енергетичною ефективністю, кореляційними властивостями, алгоритмами кодування і декодування, формою частотного спектру.

На рисунку 1.4 наведено типи кодів, що розрізняються за особливостями структури, функціональним призначенням, фізичними властивостями коду як сигналу.

Найважливішим підкласом безперервних кодів є згорткові коди, які відрізняються від інших безперервних кодів своєю побудовою та ширшою сферою застосування.

Загалом, чим довший код з постійною надлишковістю, тим більше відстань і тим вище завадостійкість коду. Однак довгі коди важкі в реалізації. Складові є компромісним рішенням проблеми, основними в чому мають каскадні та похідні коди.

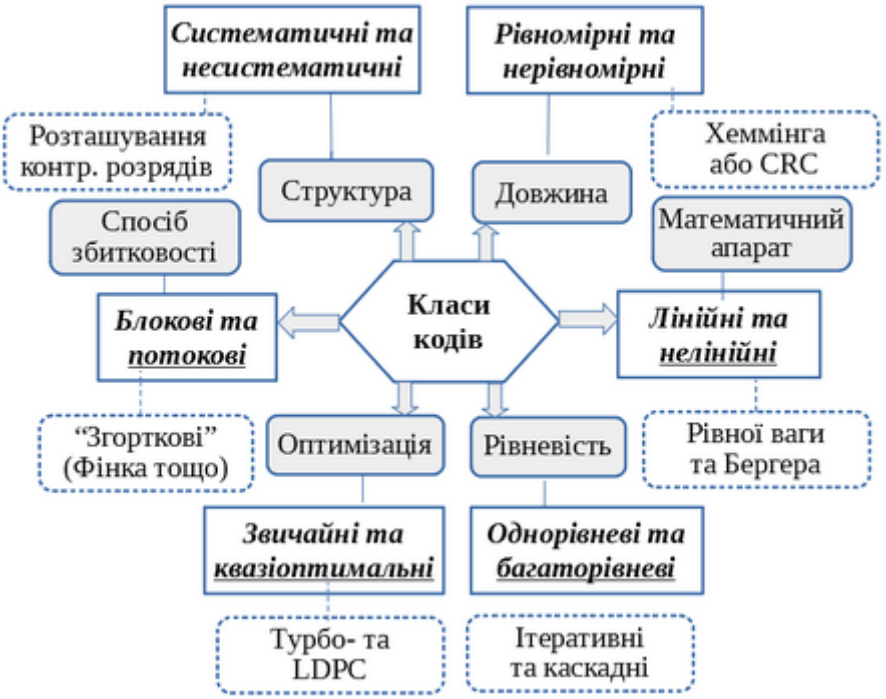


Рисунок 1.4 - Класифікація завадостійких кодів

Як правило, каскадний код складається з двох ступенів (каскадів): внутрішньої і зовнішньої. По лінії зв'язку сигнали передають внутрішнім кодом $n_{вн}$, символьні слова якого є символами зовнішнього коду довжини $n_{зн.}$ Основа зовнішнього коду дорівнює $q_{вн}^{k.}$

Похідні коди будують у вигляді матриці, в якій рядки представляють слова першого коду, а стовпці - того самого, або іншого коду.

При генеруванні каскадного коду вхідна послідовність символів розбивається на блоки символами $k_{вн}$ в кожному, кожен блок порівнюється з інформаційним символом зовнішнього коду з алфавіту, що містить значення символів $q_{вн}^{k}$. Потім інформаційні символи зовнішнього коду $k_{вн}$ перетворюються на блоки символів зовнішнього коду $n_{зн}$, і, нарешті, блоки символів символів внутрішнього коду $k_{вн}$ перетворюються на блоки символів внутрішнього коду $n_{вн}$.

Є декілька різни варіантів: зовнішній і внутрішній коди - блокові, зовнішній блочний, а внутрішній згортковий, зовнішній згортковий, а внутрішній блоковий, обидва згорткові.

Один з найпоширеніших методів генерування похідного коду полягає в тому, щоб послідовно записувати k_1 символів вхідної послідовності в k_2 рядків матриці (наприклад, в комірку RAM), додавати непотрібні n_{1-k1} символи в кожному рядку рядку і n_{2-k2} у кожному стовпці, за яким слідує рядок коду, він читається рядками або стовпцями матриці. Фізичним еквівалентом похідного коду, зокрема, є частотно-часовий код, у якому рядки розташовані на осі часу, а стовпці — на осі частот. Параметри складового коду: каскадного - $n = n_{зн} * n_{вн}$, $k = k_{зн} * k_{вн}$, $d = d_{зн} d_{вн}$; похідного - $n = n_1 * n_2$, $k = k_1 * k_2$, $d = d_1 * d_2$. Похідні коди будують на основі деякого вихідного коду, до якого або додають символи, збільшуючи відстань, або скорочують частина інформаційних символів без зміни відстані, або викидають деякі символи .

Код Хеммінга дає приклад процедури розширення, що дозволяє збільшити відстань коду з 3 до 4.

Необхідність в виколюванні виникає в результаті побудови іншого, слабшого, коротшого коду тієї ж відстані на основі вихідного коду. У ширшому тлумаченні терміна «похідний код» цей клас включає всі коди, отримані від

початкового додавання або виключення як символів, так і слів. Формально поділ кодів на двійкові та небінарні є штучним; за аналогією слід розрізняти потрійні, четвертинні та інші коди з більшою основою. Такий розподіл виправдовується складністю алгоритмів побудови, кодування та декодування небінарних кодів.

За інших таких умов, бажано, щоб інформаційні та зайві символи розміщувалися окремо. У систематичних кодах ця умова виконується. У циклічних кодах в кожному слові містяться усі його циклічні перестановки. Усі n циклічних перестановок (слів довжини n) утворюють цикл. У квазіциклічних кодах цикл складається з n-1 або рідше n-2 символів.

Циклічні коди важливі як з точки зору математичного опису, так і з точки зору побудови та реалізації коду. Помилки в каналах зв'язку мають дуже різний розподіл, однак для вибору завадостійкого коду доцільно всі можливі конфігурації помилок розділити на незалежні (некорельовані) і пакетні (корельовані помилки). На практиці необхідно враховувати якість інтервалу між пакетами: вони можуть бути бездоганними або містити випадкові незалежні помилки.

Кореляційні коди включають пари протилежних сигналів з хорошою автокореляційною функцією (метод внутрішньоімпульсної модуляції), коди інтервалів імпульсів, які мають постійну кількість імпульсів без перекривання (для будь-якого зворотного зсуву в часі) постійною для всіх кодових слів.

Ця загальна постановка проблеми в різних системах зв’язку розуміється по-різному: у дротових лініях і лінійних трактах, що містять фільтри, що обмежують смуги з крутими фронтами, необхідно «зміщувати» енергію основного сигналу з кінцевих частот до середини смуги пропускання, щоб зменшити міжсимвольне спотворення; в мережах радіозв'язку з жорсткими обмеженнями на електромагнітну сумісність радіопристроїв з коду необхідно істотно (на кілька десятків децибел) знизити рівень позасмугового випромінювання.

Побудова частотних компактних кодів кодування та декодування значною мірою залежить від методу модуляції. Арифметичні коди використовуються для боротьби з помилками під час виконання арифметичних операцій на процесорі комп'ютера. Існують також два інших типи кодів: блоковий і деревоподібний. Визначальною відмінністю кодерів для кодів цих двох типів є наявність або відсутність пам'яті. Кодер блочного коду - це пристрій без пам'яті, який відображає послідовність з k вхідних символів у послідовності з n вихідних символів. Термін «без пам'яті» вказує на те, що будь-який блок із n символів залежить лише від відповідного блоку з k символів і не залежить від інших блоків.

Це не означає, що кодер не містить елементів пам'яті. Важливими параметрами блокового коду є *n, k, R = k / n і* $d_{min.}$ На практиці значення *k* лежать між 3 і декількома сотнями, а *R == 1/4 ... 7/8*.

Значення, що перевищують ці межі, можливі, але часто призводять до деяких практичних труднощів.

Вхідні та вихідні послідовності зазвичай складаються з двійкових символів, але іноді можуть складатися з елементів більшого алфавіту.

Кодер для деревовидного коду - це пристрій пам'яті, який отримує m наборів символів двійкових входів, а набори з n символів двійкових виходів відображаються як вихідні дані. Кожен набір з n вихідних символів залежить від поточного вхідного набору та v попередніх вхідних символів. Отже, пам'ять кодера повинна містити вхідні символи v + m. Параметр v + m часто називають обмеженням довжини коду цього коду і позначається k = v + m (не плутати з параметром k для блочного коду). Довжина обмеження коду буде називатися значенням v. Типовими значеннями параметрів коду дерева є: m, n = 1 ... 8, R = 1/4 ... 7/8, v = 2 ... 60.

У випадку звичайних двійкових кодів ця операція є додаванням символ за символом двох кодових слів за модулем 2 (тобто 1 + 1 = 0, 1 + 0 = 1, 0 + 0 = 0). Ця

особливість призводить до двох важливих наслідків. Перший полягає в тому, що лінійність значно спрощує процес кодування та декодування, дозволяючи кожне кодове слово представляти як «лінійну» комбінацію невеликої кількості вибраних кодових слів, так званих базових векторів.

Друга властивість полягає в тому, що лінійність значно спрощує завдання обчислення параметрів коду, оскільки відстань між двома кодовими словами дорівнює відстані між кодовим словом, що повністю складається з нулів, і деяким іншим кодовим словом.

Таким чином, при розрахунку параметрів лінійного коду достатньо враховувати, що відбувається, коли передається кодове слово, що повністю складається з нулів. Розрахунок параметрів також спрощується, оскільки відстань Хеммінга між заданим кодовим словом і нульовим кодовим словом дорівнює кількості ненульових елементів кодового слова.

Це число часто називають вагою Хеммінга слова, а список, що містить кількість кодових слів для кожної ваги, можна використовувати для характеристики кодів за допомогою адитивних меж. Цей список називається спектром кодів.

Різниця між лінійними кодами і нелінійними кодами полягає в замкнутості кодового набору відносно лінійних операторів, таких як додавання або множення кодових слів, наприклад вектор простору, що складається з кодових слів. Лінійність коду спрощує його побудову та реалізацію.

Для великих довжин можна використовувати лише лінійні коди. Однак нелінійні коди, як правило, мають кращі параметри, ніж лінійні коди.

Для відносно коротких кодів складність побудови та реалізації лінійного та нелінійного коду приблизно однакова. І лінійні, і нелінійні коди утворюють широкий клас, що містить багато різних специфічних типів завадостійких кодів.

Серед лінійних блоків найбільше значення мають коди з первинною парністю, М-коди (симплекс), ортогональний, біортогональний, Хеммінга, Бозе-Чоудхурі-Хокінгема (BCH), Голі, Квадратний (КБ), Ріда-Соломона. Нелінійні коди включають контрольну суму, обернену, Нордстрома-Робінсона (НР), постійну вагу, переставне знакове і беззнакове повторення (повні коди для ортогональних таблиць, проективних груп, груп Матьє та інших груп перестановок). Майже всі схеми кодування, які використовуються на практиці, засновані на лінійних кодах.

Білінійні блокові коди часто називають груповими, оскільки кодові слова утворюють математичні структури, які називаються групами. Лінійні коди дерева часто називають згортковими кодами, оскільки операцію кодування можна розглядати як дискретну згортку вхідної послідовності з імпульсною характеристикою кодера

1.4 Основні характеристики завадостійких кодів

Основними характеристиками завадостійких кодів є: довжина коду n, його основа m, загальне число кодових комбінацій N, число дазволених кодових комбінацій Np, надлишковість коду Кі і мінімальне кодове розставання dmin.

Довжина коду n – це число символів у кодовій комбінації. Наприклад, комбінація 11010 складається з п’яти символів, отже, n=5. Якщо всі кодові комбінації містять однакове число символів, то код називається рівномірним. У нерівномірних кодах довжина кодових комбінацій може бути різною.

Основа коду m – це число різних символів у коді. Для двоїчних кодів символами є 1 і 0, тому m=2.

Число кодових комбінацій для рівномірного коду рівно N=mn. Наприклад, для рівномірного подвійного коду, що має довжину n=6, число різних кодових комбінацій рівно N=26=64.

Число дозволених кодових комбінацій Np – це кількість кодових комбінованих кодів, використовуваних для передачі повідомлень. Для завадостійких кодів $N_p<N$. Залишилися кодові комбінації $N-N_p$ називають забороненими. Якщо Np=N, то код є безвихідним. Для розділених кодів $N_p=2^k$.

Надлишковість коду K_i в загальному випадку визначається виразом:

$$K_n = 1 - \frac{log_2 N_p}{log_2 N}, \quad (1.1)$$

і показує, яка доля довжини кодової комбінації не використовується для передачі інформації, а використовується для підвищення завадостійкості коду. Для розділених кодів:

$$K_n = 1 - \frac{k}{n} = \frac{r}{n}, \quad (1.2)$$

де величина k/n називається відносною швидкістю коду.

Кодова відстань d(A,B) – це число позицій, у яких дві кодові комбінації A і B відрізняються друг від друга.

Наприклад, якщо A=01101, B=10111,то d(A,B)=3. Кодова відстань між комбінаціями A і B може бути знайдена в результаті додавання за модулем 2 одноіменних комбінацій, а саме

$$d(A,B) = \sum_{i=1}^{n} a_i \oplus b_i, \quad (1.3)$$

де ai и bi – i-е розряди кодових комбінацій A і B; символ $\oplus$ позначає складення за модулем 2. Наприклад, щоб отримати кодове розташування між комбінаціями 1101011 і 0111101, досить підрахувати число одиниць у сумі цих комбінацій за модулем 2:

$$\begin{array}{r} 1101011 \\ \oplus\ 0111101 \\ \hline 1010110 \end{array} \Rightarrow d(A,B)=4. \tag{1.4}$$

Кодова відстань між різними комбінаціями конкретного коду може бути різною. Так, для первинних кодів (приклади 1, 2) це положення для різних пар кодових комбінацій може приймати значення від одиниць до величини довжини коду.

Мінімальна кодова відстань d_{min} – це мінімальна відстань між дозволеними кодовими комбінаціями даного коду. Мінімальне кодове розташування є основною характеристикою коригуючої здатності коду.

У первинних (безнадлишкових) кодах всі комбінації є дозволеними, тому мінімальна кодова відстань для них дорівнює одиниці ($d_{min} = 1$).

Такі коди не здатні виявити і виправляти помилки.

Для того, щоб код володів коригуючими здібностями, його мінімальна кодова відстань повинна бути не менше двох ($d_{min} \geq 2$).

Для виявлення всіх помилок кратністю s і менше, мінімальна кодова відстань повинна задовольняти умову

$$d_{min} \geq s + 1. \tag{1.5}$$

Якщо код використовується для виправлення помилок кратності не більше t, то мінімальна кодова відстань повинна мати значення

$$d_{min} \geq 2t + 1\ . \tag{1.6}$$

Наприклад, з цих рівностей слідує, що для виявлення однократних помилок

(s = 1) потрібен код з $d_{min} = 2$, а для того, щоб виправити такі помилки, потрібен код з кодовим розташуванням $d_{min} = 3$.

Для виявлення помилок і виправлення помилок повинна виконуватись умова

$$d_{min} \geq s + t + 1 \qquad (1.7)$$

Таким чином, завдання побудови коду із заданою корректирующей здатністю зводиться до забезпечення необхідного кодового розставання.

Збільшення d_{min} призводить до зростання надлишковості коду.

При цьому бажано, щоб число перевірочних символів було мінімальним.

У нинішній час відомий ряд верхніх і нижніх кордонів, які встановлюють зв'язок між кодовими відстанями і кількостю перевірочних символів.

Дані про границі кодової відстані будуть наведені в процесі оцінок якості прийнятих кодованих повідомлень.

1.4 Код Ріда-Соломона

Код РС, ви може виправити одну помилку в одному блоці даних. При його використанні до кожного блоку інформації додаються два додаткових елементи X і Y, значення яких знаходять з умов:

- для трьох одиниць інформації (байт):

  byte1 + byte2 + byte3 + X + Y = 0

  byte1 + 2 * byte2 + 3 * byte3 + 4 * X + 5 * Y = 0

- для розрахунку конкретних значень X і Y для кодування трьох байт:

  Y = 3 * byte1 + 2 * byte2 + byte3

  X = -4 * byte1 - 3 * byte2 - 2 * byte

Тепер для знахоження помилки і її випавлення використаємо наступні розрахунки:

Error_Value = byte1 + byte2 + byte3 + X + Y

Оскільки раніше (до виникнення помилки) ця сума дорівнювала 0, то тепер вона безпосередньо дорівнює значенню помилки, яке досить легко відняти від субнормального байта. Якщо блок прийнято без помилок, Error_Value = 0. Тепер можна знайти байт, який потрібно виправити:

N = byte1 + 2 * byte2 + 3 * byte3 + 4 * X + 5 * Y

Error_Byte_Number = N / Error_Value

Впроваджуючи це в реальний алгоритм, слід перевірити, чи є помилка в блоці чи ні, тобто error_value = 0 чи ні, інакше ми отримаємо ділення на нуль.

Якщо є потреба захистити блок даних коду РС довжиною більш ніж три байти, то формули для розрахунку значення корекції лише незначно змінюються (для 16 байт):

Y = 16 * byte1 + 15 * byte2 + 14 * byte3 + ... + byte16

X = -17 * byte1 - 16 * byte2 - 15 * byte3 - ... - 2 * byte16

Error_Value = byte1 + byte2 + byte3 + ... + X + Y

N =byte1 + 2 *byte2 + 3 *byte3 + ... + 16 *byte16 + 17 * X + 18 * Y

Даним кодом незручно захищати блоки інформації менше 4 байт, так як довжина контрольних параметрів X і Y повинна бути як мінімум 4 байти.

2 байти (DW) для X та 2 байти на Y, тобто виходить. що до блоку даних з 4 байт буде доданий коригувальний блок з 4 байт.

Але що робити, якщо виникло дві або більше помилки в блоці?

Як одна з ознак виникнення двох помилок можна вважати отримання в якості Error_Byte_Number дробового числа, наприклад якщо в блоці з нулів зустрінеться 2 одиниці (дві помилки), в третьому і четвертому байтах, то Error_Byte_Number = 3.5 але якщо 4 одиниці, відповідно в 3, 4 і 5 байтах то Error_Byte_Number = 4.

Канал зв’язку або передачі даних, захищений Кодом РС, показаний на рисунку 1.5.

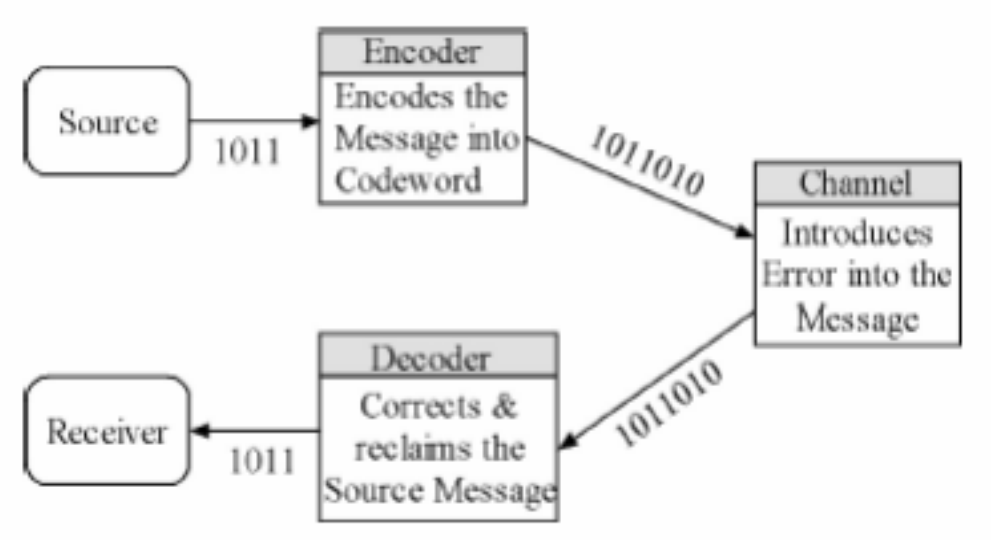


Рисунок 1.5 - Канал зв’язку, захищений кодом РС

1.5 Постановка задачі

Розробити пристрій завадостійкого обміну даними(даних) на основі коду Ріда-Соломона, що забезпечує передачу інформації з максимальною швидкістю при забезпеченні достовірності передачі даних не гірше заданої, при мінімальних апаратурних витратах.

З цією метою необхідно забезпечити вирішення наступних завдань:

1) Розробити алгоритм роботи пристрою завадостійкого обміну даними на основі завадостійкого коду Ріда-Соломона.

2) Розробка функціональної схеми кодуючого та декодуючого пристрою обраного коду.

3) Моделювання роботи розроблених пристроїв.

2. МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ПРИСТРОЮ ЗАВАДОСТІЙКОГО КОДУВАННЯ НА БАЗІ FPGA

2.1 Методи представлення коду Ріда-Соломона

Коди Ріда-Соломона представляють собою лінійні блокові коди під безліччю кодів БЧХ. Код РС заснований на кінцевих полях, що називають полями Галуа. Кількість помилок, які можуть бути виправлені, залежать від характеристик коду РС. Параметри *RS(n, k)* кодів описуються таким чином: *m* -і кількість біт на символ, *k* - довжина незашифрованого повідомлення в символах, n - довжина кодового слова в символах *(n-k)* - кількість символів перевірки парності, *t* - кількість помилок, що виправляються, *2t = n-k.*

На рисунку 2.1 показано типове систематичне кодове слово РС. Це систематичний код, оскільки дані зліва незмінні, а символи парності додаються разом з даними для формування кодового слова.

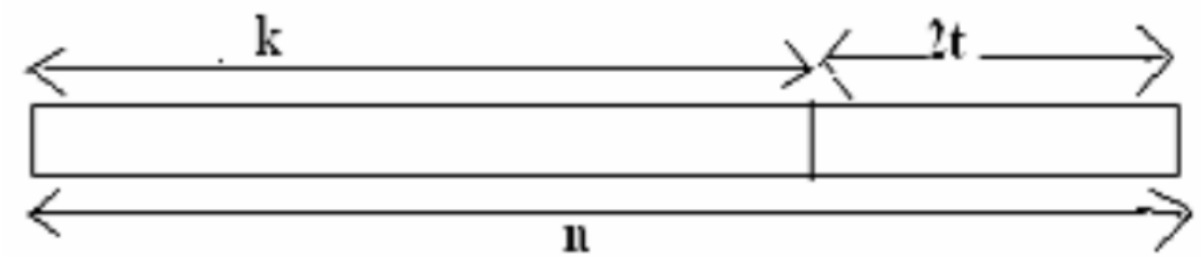


Рисунок 2.1 - Кодове слово кода Ріда соломона *RS(n, k)*

Традиційно код РС, що виправляє t_{min} помилок, описується над полем Галуа GF(q) породжувальним поліномом

$$x(x) = (x - \alpha)\left(x - a^{2}\right)\ldots\left(x - a^{2tmin}\right), GF(q). \qquad (2.1)$$

В випадку використання автоматно-аналітичного способу представлення кодів РС на основі лінійних послідовних схем, ЛПС над полем *GF(q)* (символьна ЛПС) задається функцією станів (переходів)

$$S(t+1) = A \times S(t) + B \times U(t), GF(q), \qquad (2.2)$$

і функцією виходів

$$Y(t) = C \times S(t) + D \times U(t), GF(q), \qquad (2.3)$$

де t – дискретний час; $A = [a_{ij}]_{r*r}$, $B = [b_{ij}]_{r*l}$, $C = [c_{ij}]_{m*r}$, $D = [d_{ij}]_{m*l}$ – характеристичні матриці; $S(t) = [s_i]_r$ – слово стану, $U(t) = [u_i]_l$ – вхідне слово, $Y(t) = [y_i]_m$ – вихідне слово.

Слід розрізняти автоматно-аналітичну і автоматно-графову моделі коду РС. Оскільки символьна ЛПС є кінцевим автоматом, тому автоматно-графову модель коду РС можна вибрати граф переходів-виходів G_{FA} цього автомата. До речі, формула для обчислення циклів на регістрах зсуву з лінійним оберненим зв'язком(РЗЛОЗ). безпосередньо випливає із формули 2 ЛПС, оскільки РЗЛОЗ є найпростішим випадоком ЛПС.

Проведемо детальний аналіз автоматно-аналітичної моделі коду РС, яка базується на характеристичних матрицях ЛПС.

Якщо породжувальний поліном перетворити до вигляду

$$g(x) = a_0^i + a_1^i x + a_2^i x^2 + \ldots + a_{t_{min}-1}^i x^{2t_{min}}, GF(q) \qquad (2.4)$$

тоді можна отримати чотири типи символьних ЛПС: рекурсивні ЛПС типу Галуа, рекурсивні ЛПС типу Фібоначчі, нерекурсивні ЛПС типу Галуа та нерекурсивні ЛПС типу Фібоначчі.

Розглянемо лише перші два типи ЛПС, які використовуються для систематичного кодування кодів РС. Рекурсивні ЛПС типу Галуа – це ЛПС, у яких сигнали на входи поступають з їх виходів, а характеристичні матриці мають вигляд:

$$A = \begin{vmatrix} 0 & 0 & \cdots & 0 & \alpha_0^i \\ \alpha^0 & 0 & \cdots & 0 & \alpha_1^i \\ 0 & \alpha^0 & \cdots & 0 & \alpha_2^i \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \alpha^0 & \alpha_{d_{\min}-1}^i \end{vmatrix}$$

$$B = \begin{vmatrix} \alpha^0 \\ 0 \\ 0 \\ \cdots \\ 0 \end{vmatrix}$$

$$C = |0 \ldots 0\alpha|$$
$$D = |0|$$

Рекурсивні ЛПС типу Фібоначчі – це ЛПС, у яких сигнали на входи поступають з їх виходів, а характеристичні матриці мають вигляд:

$$A = \begin{vmatrix} 0 & \alpha^0 & 0 & \ldots & 0 \\ 0 & 0 & \alpha^0 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \alpha_0^i & \alpha_1^i & \alpha_2^i & \ldots & \alpha_{2\tau_{\min}-1}^i \end{vmatrix}$$

$$B = \begin{vmatrix} 0 \\ 0 \\ 0 \\ \ldots \\ \alpha^0 \end{vmatrix}$$

$$C = |0 \ldots 0\alpha|$$
$$D = |0|$$

На основі автоматно-аналітичної моделі можна дати означення коду РС: множина всіх двійкових послідовностей *M* довжини *n* , які переводять ЛПС із будь-якого початкового стану $S(t)_{beg}$ знову в стан $S(t)_{beg}$, утворює *(n, k)*-код РС Ω над полем Галуа *GF(q)*. Кожна така послідовність *M* є кодовим словом *Z(n, k)*-коду РС.

Виходячи із наведеного означення коду РС, задача кодування зводиться до знаходження такого кодового слова *Z* довжиною n, яке при подачі на входи ЛПС переводить її з деякого початкового стану $S_{beg}(t)$, знову в цей же стан. Як початковий стан $S_{beg}(t)$ будемо надалі розглядати нульовий стан S(0) .

Розглянемо систематичне кодування циклічних кодів за допомогою рекурсивних ЛПС. Суть процедури кодування в цьому випадку буде такою.

Під дією на вхід інформаційного слова I ЛПС перейде з початкового стану S(0) в деякий стан.

$$S(k) = \begin{bmatrix} s_0^k \\ s_1^k \\ \dots \\ s_{r-2}^k \\ s_{r-1}^k \end{bmatrix} \qquad (2.5)$$

Далі необхідно визначити контрольне слово *Ψ*, яке переведе ЛПС із стану *S(k)* знову в стан *S(0)* . В підсумку стане відомим кодове слово Z . Спочатку покажемо існування такого слова Ψ.

Для будь-якої пари станів *S(i)* і *S(j)* існує вхідна послідовність із r символів, яка переводить r-вимірну ЛПС із *S(i)* в *S(j)* , якщо ЛПС є r-керованою. А ЛПС буде r - керованою, якщо ранг r * r -матриці

$$L_r = \left[A^{r-1} * A^{r-2} * B, \dots, A * B, B\right] \qquad (2.6)$$

буде дорівнювати r .

Підставляючи матриці A і B рекурсивної ЛПС типу Галуа в формулу 2.6, отримаємо такий вигляд матриці L_r :

$$L_r^G = \begin{bmatrix} 0 & 0 & \dots & 0 & \alpha^0 \\ 0 & 0 & \dots & \alpha^0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \alpha^0 & \dots & 0 & 0 \\ \alpha^0 & 0 & \dots & 0 & 0 \end{bmatrix}. \quad (2.7)$$

Підставляючи матриці A і B рекурсивної ЛПС типу Фібоначчі в формулу 2.6, отримаємо такий вигляд матриці L_r :

$$L_r^F = \begin{bmatrix} \alpha^0 & 0 & \dots & 0 & 0 \\ l_{2,1} & \alpha^0 & \dots & 0 & 0 \\ l_{3,1} & l_{3,2} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ l_{r,1} & l_{r,2} & \dots & l_{r,r-1} & \alpha^0 \end{bmatrix}, \quad l_{i,j} = \{\alpha^0, \alpha^1, \dots, \alpha^{n-1}\}, i \neq j . \quad (2.8)$$

Ненульові значення діагональних елементів отриманих матриць свідчать про те, що їх ранг дорівнює r . Отже, існує слово Ψ довжиною r , яке переводить ЛПС із стану S(k) в довільний заданий стан, в тому числі і в стан S(0) .

Розглянемо кодування за допомогою рекурсивної ЛПС типу Галуа.

Систематичне кодування *(n, k)*-коду РС за допомогою рекурсивної ЛПС типу Галуа може бути виконано або за *(k + 1)* тактів після подачі на її вхід інформаційного слова *I* , або за *n* тактів після подачі на її вхід інформаційного *I* і нульового *O* слів. В першому випадку компоненти Ψ_i слова $\Psi=\{\psi_0, \psi_1, \dots, \psi_{r-2}, \psi_{r-1}\}$ можуть бути знайдені в результаті обчислення згортки:

$$\psi_i = \sum_{j=0}^{r-1} a_{i,j} * s_j^k, GF(q) \quad (2.9)$$

де s_j^k - j-та компонента слова стану S(k); $a_{i,\ j}$ - компоненти r -го степеня матриці

$$A\left(s_j \epsilon S(k), a_{i,j} \epsilon A^r, i = 0r - 1, j = 0r - 1\right), \quad (2.10)$$

а у другому випадку дорівнюють:

$$\psi = s_{r-1-j}^m \quad . \quad (2.11)$$

Де s_{r-1-j}^m – (r – 1 – j)–та компонента слова стану S(m) ; ($s_{r-1-j}^m \in$S m , j = 0 . . . r – 1).

Із теорії ЛПС відомо, що при подачі на вхід ЛПС, яка знаходиться в деякому початковому стані S(0), інформаційного слова I довжиною k ЛПС перейде в стан S(k) , що визначається з рівності

$$S(k) = A^k * S(0) + L_k * I, GF(q) \quad (2.12)$$

Якщо далі на вхід ЛПС подати контрольне слово Ψ довжиною r , тоді ЛПС перейде в стан S(n) , який визначається співвідношенням

$$S(n) = A^r * S(k) + L_r * \Psi GF(q) \quad (2.13)$$

Оскільки S(n) =S(0) , тому рівність (2.12) можна записати як

$$L_r * \Psi = A^r * S(k) \quad \text{Формула 13}$$

Підставляючи значення матриці L_r із (2.8) в (2.14), отримаємо згортку (2.10). Оскільки стан S(k) буде отримано на k -му такті функціонування ЛПС, отже, контрольне слово Ψ може бути обчислено на (k +1)-му такті, і весь процес кодування виконується за (k + 1) тактів. Продовжимо аналіз рівності (2.13). Якщо в цю рівність підставити значення S(k) із (2.14), тоді отримаємо:

$$S(n) = A^r * \left(k * S(0) + L_k * I\right) + L_r * \Psi =$$
$$A^{k+r} * S(0) + A^r * L_k * I + L_r * \Psi =$$
$$= S(0) + A^r * L_k * I + L_r * \Psi, \ \mathrm{GF(q)} \qquad (2.15)$$

Добуток L_k * I визначає стан, в який перейде ЛПС після подачі на її вхід слова I , а добуток $A^r * L_k * I$ визначає деякий стан *S(m)* , в який перейде ЛПС після наступної подачі на її вхід нульового слова O довжиною r . Таким чином, значення слова стану S(n) буде таким:

$$S(n) = S(0) + S(m) + L_r * \Psi, GF(q). \qquad (2.16)$$

Оскільки метою кодування є перехід із початкового стану S(0) знову в цей же стан, тому S(n) =S(0) , і тоді із виразу (2.13) випливає, що повинна виконуватися рівність L_r * Ψ = S(m) . Підставивши значення матриці L_r із (Формули 9), отримаємо:

$$\Psi = - S(m) \qquad (2.17)$$

Знак «–» в (2.17) означає операцію векторної інверсії, тобто взаємної перестановки між молодшими і старшими компонентами слова.

В підсумку отримаємо співвідношення між компонентами слів *S(m)* і Ψ, яке міститься в (2.11). Оскільки стан S(m) буде отримано на n-му такті

функціонування ЛПС, контрольне слово Ψ може бути знайдено на n-му такті, і кодування виконується за n тактів.

Таким чином, процес кодування розбивається на два етапи. На першому етапі поступає інформаційне слово I, як правило, послідовно. Тому спочатку можна використати рекурентний спосіб кодування за формулою (2.6). Під дією слова I ЛПС протягом k тактів перейде з нульового стану S(0) в стан S(k) .

На другому етапі кодування формується контрольне слово Ψ. Якщо це слово буде передаватись в канал також послідовно, тоді обчислити компоненти Ψ_i можна рекурентним способом за r тактів. Якщо ж необхідно швидко, за один такт, обчислити слово Ψ і передати його в канал паралельно (наприклад, в комп'ютерній мережі), саме тоді знадобиться згортковий спосіб кодування.

На приклад, над полем Галуа GF(8) задано інформаційне слово

$$I = \left[a^1 a^5 a^3 a^4 a^0 a^2 a^5 a^6 a^0 a^2 a^{12}\right] \qquad (2.18)$$

Для систематичного кодування використаємо (15,11)-код РС, якому відповідає породжувальний поліном

$$g(x) = (x - a)\left(x - a^2\right)\left(x - a^3\right)\left(x - a^4\right) = a^{10} + a^3 + a^6 x^2 + a^{13} * x^3 + x^4 \qquad (2.19)$$

та характеристичні матриці рекурсивної ЛПС типу Галуа:

$$A = \begin{vmatrix} 0 & 0 & 0 & \alpha^{10} \\ \alpha^0 & 0 & 0 & \alpha^3 \\ 0 & \alpha^0 & 0 & \alpha^6 \\ 0 & 0 & \alpha^0 & \alpha^{13} \end{vmatrix}; \quad B = \begin{vmatrix} \alpha^0 \\ 0 \\ 0 \\ 0 \end{vmatrix} \qquad (2.20)$$

Перший етап кодування виконаємо рекурентним способом за (2.5). В результаті ЛПС протягом 11 тактів перейде зі стану S(0) в стан S(11) (з метою

економії місця будемо записувати стовпці станів ЛПС у вигляді рядків):

$$S(11) = \left[a^{9} a^{3} a^{4} a^{13}\right] \tag{2.21}$$

Другий етап кодування виконаємо згортковим способом за (Формула 9). Для цього знадобиться четверта степінь матриці А (її можна наперед підготувати і використовувати для будь-якого інформаційного слова цього коду):

$$A^4 = \begin{vmatrix} \alpha^{10} & \alpha^{8} & \alpha^{11} & \alpha^{11} \\ \alpha^{3} & \alpha^{8} & \alpha^{5} & \alpha^{13} \\ \alpha^{6} & \alpha^{7} & \alpha^{11} & \alpha^{13} \\ \alpha^{13} & \alpha^{1} & \alpha^{1} & \alpha^{10} \end{vmatrix} \tag{2.22}$$

Тепер можна обчислити компоненти слова $\Psi = [\Psi_0\ \Psi_1\ \Psi_2\ \Psi_3]$:

$$\begin{aligned} \Psi_0 &= a^{10} a^{9} + a^{8} a^{3} + a^{11} a^{4} + a^{11} a^{11} = a^{10}, GF(8) \\ \Psi_1 &= a^{3} a^{9} + a^{8} a^{3} + a^{5} a^{4} + a^{13} a^{11} = a^{0}, GF(8) \\ \Psi_2 &= a^{6} a^{9} + a^{7} a^{3} + a^{11} a^{4} + a^{13} a^{11} = a^{13}, GF(8) \\ \Psi_3 &= a^{13} a^{9} + a^{1} a^{3} + a^{1} a^{4} + a^{10} a^{11} = a^{1}, GF(8) \end{aligned} \tag{2.23}$$

2. 2 Кодер

Кодер РС приймає блок символів і додає додаткові символи перевірки парності. Кодове слово РС може бути обчислено з символів вхідних повідомлень з використанням генераторного багато члена. Багаточленгенератора залежить від порядку поля Галуа, над яким був визначений код RS, і числа помилок, які потрібно виправити.

Примітивний багаточлено визначається порядком поля Галуа. Одним з примітивних поліномів у полі GF(2^8) являється $x^8 + x^4 + x^3 + x^2 + 1$.

Нехай *a* - примітивний елемент в полі Галуав GF(2^8). Тоді генераторний багаточлендля t-коду виправлення мошибок RS виглядає таким чином.

$$\prod_{i=1}^{2t}\left(x + a^{i}\right). \tag{2.24}$$

Для двоїчного байтового коду з виправленням помилок (t = 2) відповідний генераторний многочлен:

$$g(x) = (x + a)\left(x + a^{2}\right)\left(x + a^{3}\right)\left(x + a^{4}\right). \tag{2.25}$$

Нехай вхідний многочлен M(x) Кодера

$$M(x) = m_{k-1} + m_{k-2}\, x^{k-2} + \ldots + c_1 x + c_0 \ . \tag{2.26}$$

Відповідний многочлен кодового слова задається формулою

$$C(x) = c_{n-1} + c_{n-2} x^{n-2} + \ldots + c_1 x + c_0. \tag{2.27}$$

Це кодове слово генерується вихідного інформаційного полінома, використовуючи наступну формулу.

$$C(x) = x^{2t} M(x) + x^{2t} M(x) mod g(x). \tag{2.28}$$

Якщо Q(x) і P(x) - відповідні поліноми приватного залишку, коли $x^{2t}M(x)$ ділиться на g(x), томного членового коду також може бути виражений іншим способом.

$$C(x) = x^{2t}M(x) + P(x) = Q(x)g(x). \qquad (2.29)$$

Тут P(x) - поліном перевірки парності. З (Формула 28) Можна отримати

$$\frac{x^{n-k}M(x)}{g(x)} = Q(x) - \frac{P(x)}{g(x)} \qquad (2.30)$$

Додаючи поліном P(x) перевірки на парність до поліному x^2 M(x), обчислюється поліном C(x) кодового слова, який націло ділиться на g(x).

Виходячи з розрахунків, можемо побудувати схему кодера Ріда-Соломона.

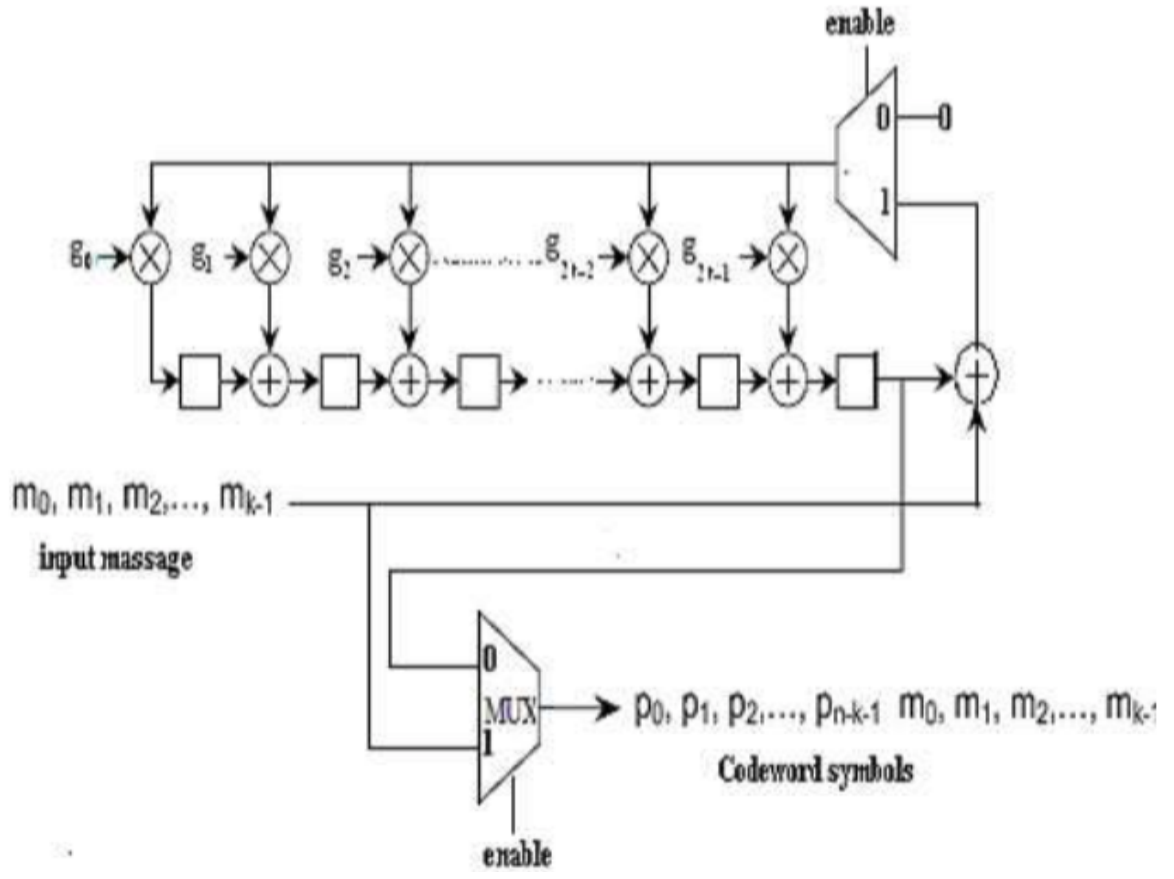


Рисунок 2.2 - Схема кодера Ріда-Соломона

2.3 Декодер

В якості полінома p (z) будемо використовувати примітивний поліном. Тоді, після розкриття скобо заміни α на z, що породжує поліном коду Ріда-Соломона, що виправляє подвійні помилки, можна представити в наступному вигляді:

$$RS(x) = (X - a)\left(X - a^2\right)\left(X - a^3\right)\left(x - a^4\right) = X^4\left(z^4 + z^3 + z^2 + z\right) + \\ + X^2\left(z^7 + z^6 + z^4 + z^3\right) + \left(z^9 + z^8 + z^7 + z^6\right) + z^{10} \quad (3.31)$$

Даний породжуючий поліном є справедливим для будь-якого коду Ріда-Соломона, що виправляє подвійні помилки, і будь-якого поля Галуа.

Для обчислення породжуючого полінома для конкретного поля Галуа необхідно обчислити відповідні коефіцієнти при псевдозмінних X. p(z). Залишок від поділу і буде шуканим коефіцієнтом.

Використовуючи таблицю непривідних поліномів, як p(z) можна вибрати перший поліном (у цій таблиці перший поліном завжди примітивний, причому з найменшою кількістю ненульових коефіцієнтів) восьмого ступеня

$$p(z) = 435_8 = 100011101_2 = z^8 + z^4 + z^3 + z^2 + 1 \quad (3.32)$$

Після приведення по модулю p(z) ступінь всіх коефіцієнтів полінома, що породжує, буде не більше семи:

$$\left(4 + z^3 + z^2 + z\right) modp(z) = z^4 + z^3 + z^2 + z$$
$$\left(z^7 + z^6 + z^4 + z^3\right) modp(z) = z^7 + z^6 + z^4 + z^3$$
$$\left(z^9 + z^8 + z^7 + z^6\right) modp(z) = z^7 + z^6 + z^5 + z^2 + z + 1$$
$$\left(z^{10}\right) modpz(z) = z^6 + z^5 + z^4 + z^2 \quad (3.33)$$

Таким чином, полінос РС для поля GF(2^8):

$$RS(x) = x^4 + x^3\left(z^4 + z^3 + z^2 + z\right) + x^2\left(z^7 + z^6 + z^4 + z^3\right) + \\ + x\left(z^7 + z^6 + z^5 + z^2 + z + 1\right) + \left(z^6 + z^5 + z^4 + z^2\right) \quad (3.34)$$

На основі цих розрахунків можемо побудувати схему декодеру.

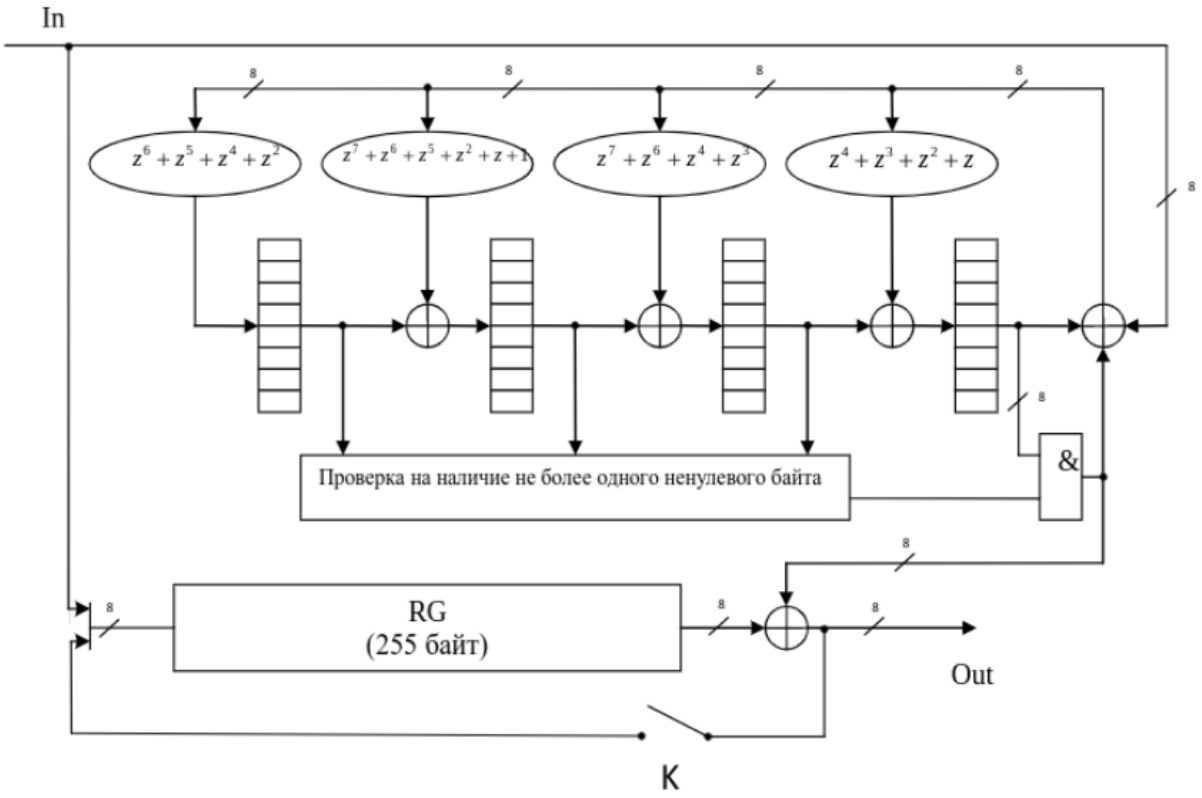


Рисунок 2.3 - Схема декодеру RS(255, 251)

Такий декодер виправляє два можливі помилкові байти за 3n тактів:

1-і n тактів формується синдром, кодове слово заноситься в буферний регістр;

2-і n тактів виконується виправлення однієї з двох можливих помилкових байтів. Синдром модифікується, кодове слово наново переписується в буферний регістр;

3-і n тактів виконується виправлення другого помилкового байта. побудови декодера на елементах двійкової логіки необхідно представити помножувачі на константу у вигляді суматорів за модулем два. Для цього необхідно елемент поля у загальному вигляді

Недолік такого декодера - неузгодженість роботи кодера і декодера (кодер повинен чекати 2n тактів, поки декодер виправить помилки) - усувається застосуванням конвеєрної реалізації.

Для побудови декодера на елементах двійкової логіки необхідно уявити помножувачі на константу як суматорів по модулю два.

Для цього необхідно елемент поля у загальному вигляді:

$$a_7 - z^7 + a_6 z^6 + a_5 z^5 + a_4 z^4 + a_3 z^3 + a_2 z^2 + a_1 z + a_0 \qquad (3.35)$$

помножити відповідний коефіцієнт і розділити на p(z). Коефіцієнти залишку від розподілу дозволяють уявити помножувач у вигяді суматорів.

$$R_7 = \left(a_7 z^6 + a_5 z^5 a_4 z^4 + a_3 z^3 + a_2 z^2 + a_1 z\right) z^7 mod p(z) =$$
$$\left(a_6 + a_5 + a_4 + a_0\right) z^7 + \left(a_5 + a_4 + a_3\right) z^6 +$$
$$+ \left(a_7 + a_4 + a_3 + a_2\right) z^5 + \left(a_7 + a_3 + a_2 + a_1\right) z^4 +$$
$$+ \left(a_5 + a_4 + a_2 + a_1\right) z^3 + \left(a_6 + a_5 + a_3 + a_1\right) z^2 +$$
$$+ \left(a_7 + a_6 + a_2\right) z + \left(a_7 + a_6 + a_5\right)$$
$$R_6 =$$
$$\left(a_7 + a_6 + a_5 + a_2\right) z^7 + \left(a_6 + a_5 + a_4\right) z^6 +$$
$$+ \left(a_5 + a_4 + a_3\right) z^5 + \left(a_7 + a_4 + a_3 + a_2\right) z^4 +$$
$$+ \left(a_6 + a_5 + a_3 + a_2\right) z^3 + \left(a_7 + a_6 + a_4 + a_2\right) z^2 +$$
$$+ \left(a_7 + a_3\right) z + \left(a_7 + a_6 + a_2\right)$$
$$R_5 =$$
$$\left(a_7 + a_6 + a_2\right) z^7 + \left(a_7 + a_6 + a_5 + a_1\right) z^6 +$$
$$+ \left(a_6 + a_5 + a_4 + a_0\right) z^5 + \left(a_5 + a_4 + a_3\right) z^4 +$$
$$+ \left(a_6 + a_4 + a_3\right) z^3 + \left(a_7 + a_5 + a_3\right) z^2 + a_5 z + a_5$$
$$R_4 =$$

$$\left(a_7 + a_3\right)z^7 + \left(a_7 + a_6 + a_2\right)z^6 +$$
$$+ \left(a_7 + a_6 + a_5 + a_1\right)z^5 + \left(a_6 + a_5 + a_4 + a_0\right)z^4 +$$
$$+ \left(a_7 + a_5 + a_4\right)z^3 + \left(a_7 + a_6 + a_4\right)z^2 + a_5 z \ + a_4$$

$$R_3 =$$
$$a_4 z^7 + \left(a_7 + a_3\right)z^6 + \left(a_7 + a_6 + a_2\right)z^5 + \left(a_7 + a_6 + a_5 + a_1\right)z^4 +$$
$$+ \left(a_6 + a_5 + a_0\right)z^3 + \left(a_7 + a_5\right)z^2 + \left(a_7 + a_6\right)z \ + a_5$$

$$R_2 =$$
$$a_5 z^7 + a_4 z^6 + \left(a_7 + a_3\right)z^5 + \left(a_7 + a_6 + a_2\right)z^4 +$$
$$+ \left(a_7 + a_6 + a_1\right)z^3 + \left(a_6 + a_0\right)z^2 + a_7 z \ + a_6$$

$$R_1 =$$
$$a_6 z^7 + a_5 z^6 + a_4 z^5 + \left(a_7 + a_3\right)z^4 + \left(a_7 + a_2\right)z^3 + \left(a_7 + a_1\right)z^2 + a_0 z \ + a_7$$
$$R_0 =$$
$$a_7 z^7 + a_6 z^6 + a_5 z^5 + a_4 z^4 + a_3 z^3 + a_2 z^2 + a_1 z \ + a_0 \qquad (3.36)$$

Використовуючи отримані залишки, можна перейти від помножувачів на константи до реалізації на елементах двійкової логіки. Наприклад, для псевдозмінної X_3:

$$\left(a_7 + a_6 + a_5 + a_4 + a_3\right)z^7 + \left(a_6 + a_5 + a_4 + a_3 + a_2\right)z^6 +$$
$$+ \left(a_7 + a_6 + a_5 + a_4 + a_3 + a_2 + a_1\right)z^5 + \left(a_7 + a_6 + a_5 + a_4 + a_3 + a_2 + a_0\right)z^4 +$$
$$+ \left(a_7 + a_4 + a_2 + a_1 + a_0\right)z^3 + \left(a_7 + a_5 + a_1 + a_0\right)z^2 + \left(a_7 + a_5 + a_4 + a_1 + a_0\right)z^2$$
$$+ \left(a_6 + a_2 + a_5 + a_0\right)z \ + \left(a_7 + a_6 + a_5 + a_4\right). \qquad (3.37)$$

По отриманому залишку від розподілу будується схема помножувача на константу. Інші елементи кодера і декодера для апаратної реалізації (255, 251) коду

Ріда-Соломона на елементах двійкової логіки не вимагають будь-яких спеціальних розрахунків або методів побудови.

Розглянуті способи декодування та перетворення множників можуть бути застосовні не тільки для даного аналізованого коду, але також для будь-яких кодів Ріда-Соломона, що виправляють поодинокі або подвійні помилки, посимвольно перемежеані та/або укорочені.

І, нарешті, слід зазначити, що для генератора синдрому будь-якого апаратно-реалізованого коду Ріда-Соломона необхідна заміна помножувачів на константу на основі примітивних елементів суматорами за модулем два.

3. АПАРАТНА РЕАЛІЗАЦІЯ КОДУ РІДА-СОЛОМОНА

3.1 Плата FPGA

Програмована вентильна матриця (FPGA) це інтегральна схема, призначена для налаштування замовником або конструктором після виготовлення, звідси термін, що програмується полем.

Конфігурація FPGA, як правило, визначається за допомогою мови опису апаратного забезпечення (HDL), подібної до тієї, що використовується для інтегральної схеми (ASIC) для конкретного застосування.

Раніше для визначення конфігурації використовувалися принципові схеми, але це трапляється все рідше через появу електронних засобів автоматизації проектування.

FPGA містять масив програмованих логічних блоків та ієрархію реконфігурованих взаємоз'єднань, що дозволяють з'єднувати блоки разом. Логічні блоки можуть бути налаштовані на виконання складних комбінаційних функцій або діяти як прості логічні вентилі, такі як І та XOR.

У більшості FPGA логічні блоки також містять елементи пам'яті, які можуть бути простими тригерами або більш повними блоками пам'яті.

Багато FPGA можна перепрограмувати, щоб реалізувати різні логічні функції, дозволяючи гнучко переконфігурувати обчислення, як це виконується в комп’ютерному програмному забезпеченні.

FPGA відіграють важливу роль у розробці вбудованих систем завдяки їх здатності починати розробку системного програмного забезпечення (SW) одночасно з апаратним забезпеченням (HW), забезпечувати моделювання продуктивності системи на дуже ранній стадії розробки та допускати різні системні розділи (SW і HW).

Область застосування FPGA величезна. Сьогодні вони використовуються в центрах обробки даних, аерокосмічній техніці, обороні, штучному інтелекті (AI),

промисловому IoT (інтернет речей), дротових та бездротових мережах, автомобільній промисловості та багатьох інших галузях.

Такі пристрої часто знаходяться в середовищах, де користувачам потрібна інформація в режимі реального часу.

Наприклад, домашня камера безпеки повинна миттєво передавати зображення на розумні пристрої власника будинку — з високою роздільною здатністю та мінімальною затримкою. Ці очікування будуть тільки зростати, оскільки споживачі стають все більше залежати від миттєвої інформації, яка є у них під рукою.

3.2 Мова VHDL

Very High-Speed Integrated Circuit Hardware Description Language (VHSIC HDL, VHDL) — це мова опису, яка використовується для опису обладнання. Вона використовується в автоматизації електронного проектування для передачі змішаних сигналів і цифрових систем, таких як інтегральні схеми і FPGA .

VHDL був розроблений на початку 1980-х років як додатковий продукт дослідницького проекту високошвидкісних інтегральних схем, який фінансується Міністерством оборони США.

Під час програми VHSIC дослідники зіткнулися з непростим завданням опису схем величезного масштабу (для свого часу) і управління дуже великими проблемами проектування схем, які залучали кілька команд інженерів.

Оскільки були доступні лише інструменти проектування на рівні вентилів, незабаром стало зрозуміло, що потрібно буде розробити кращі, більш структуровані методи та інструменти проектування.

Щоб вирішити цю проблему, Міністерство оборони уклало контракт з командою інженерів із трьох компаній – IBM, Texas Instruments та Intermetrics, щоб завершити специфікацію та впровадити новий, заснований на мові метод

опису дизайну. Перша загальнодоступна версія VHDL, версія 7.2, була доступна в 1985 році.

VHDL має багато функцій, що підходять для опису (до великого рівня деталізації) поведінки електронних компонентів, починаючи від простих логічних вентилів і закінчуючи повноцінними мікропроцесорами та користувацькими мікросхемами.

Особливості VHDL дозволяють точно описати електричні аспекти поведінки ланцюга (такі як час наростання і спаду сигналів, затримки через вентилі та функціональна робота).

Отримані імітаційні моделі VHDL потім можна використовувати як будівельні блоки в більших схемах (з використанням схем, блок-схем або описів VHDL на системному рівні) з метою моделювання.

Подібно до того, як мови програмування високого рівня дозволяють виражати складні концепції проектування у вигляді комп'ютерних програм, VHDL дозволяє зафіксувати поведінку складних електронних схем у системі проектування для автоматичного синтезу схем або для моделювання системи.

Як і Pascal, C і C++, VHDL включає функції, корисні для методів структурованого проектування, і пропонує багатий набір функцій керування та представлення даних.

На відміну від цих інших мов програмування, VHDL надає функції, що дозволяють описувати одночасні події.

Це важливо, оскільки обладнання, яке описується за допомогою VHDL, за своєю суттю одночасне. Користувачі мов програмування PLD, таких як PALASM, ABEL, CUPL та інших, знайдуть одночасні функції VHDL досить знайомими.

Тим не менш, ті, хто програмував лише програмними мовами програмування, повинні зрозуміти деякі нові поняття.

Одним з найважливіших (і недостатньо використовуваних) аспектів VHDL є його здатність отримувати специфікацію продуктивності схеми у формі, яку зазвичай називають тестовим стендом.

Тестові стенди – це VHDL-описи стимулів ланцюга та відповідних очікуваних результатів, які перевіряють поведінку схеми з часом.

Тестові стенди повинні бути невід'ємною частиною будь-якого проекту VHDL і повинні створюватися паралельно з іншими описами схеми.

VHDL — це потужна мова, за допомогою якої можна вводити нові проекти на високому рівні, але вона також корисна як низькорівнева форма зв'язку між різними інструментами в комп'ютерному середовищі проектування.

Структурні особливості мови VHDL дозволяють ефективно використовувати його як мову списків мереж, замінюючи (або доповнюючи) інші мови списків мереж, такі як EDIF.

Переваги VHDL:

- підтримує різні методології проектування, такі як підхід зверху вниз і підхід знизу вгору.
- забезпечує гнучку мову дизайну;
- дозволяє краще керувати дизайном;
- підтримує багаторівневу абстракцію;
- забезпечує щільне з’єднання з нижчими рівнями конструкції;
- підтримує всі інструменти CAD;
- підтримує можливість повторного використання коду та спільне його використання.

Недоліки VHDL:

- вимагає специфічних знань про структуру та синтаксис мови;

- складніше візуалізувати та усунути несправність дизайну;

- деякі програми VHDL не можна синтезувати;

- VHDL важко вивчати.

У VHDL є 4 типи стилів моделювання:

1. Моделювання потоку даних.

Моделювання потоку даних можна описати на основі булевого виразу. Він показує, як дані переходять від входу до виходу. Він працює на паралельному виконанні.

2. Поведінкове моделювання.

Поведінкове моделювання використовується для послідовного виконання операторів. Він показує, як система працює відповідно до поточного оператора.

Поведінкове моделювання може містити оператори процесу, послідовні оператори, оператори призначення сигналу та оператори очікування.

3. Структурне моделювання (Підключення підмодулів).

Структурне моделювання використовується для визначення функціональності та структури схеми.

Структурне моделювання містить оголошення сигналів, екземпляри компонентів і карти портів у екземплярі компонента.

VHDL використовує такі три типи об’єктів:

1. Константи

Константа — це об’єкт, який може містити лише одне значення, яке не можна змінити протягом усього коду.

Приклад: константа number_of_bytes integer:=8;

2. Змінні

Змінна також містить одне значення даного типу. Значення змінної можна змінити під час моделювання за допомогою оператора присвоєння змінної.

У процесах і підпрограмах використовуються змінні.

Змінні призначаються оператором присвоєння ":=".

Приклад:

variable index: integer :=0;

3. Сигнали

Сигнали можуть бути оголошені в архітектурі та використані в будь-якому місці архітектури. Сигнали призначаються оператором присвоєння "<=".

Приклад:

Signal sig1: std_logic;

Sig1 <= '1'

У VHDL є такі типи даних:

1. скалярні типи:
   - integer (цілочисельні типи даних — це набір додатних і від'ємних цілих чисел);
   - floating point (типи даних з плаваючою комою — це набір додатних і від'ємних чисел, які містять десяткову кому);
   - enumeration (тип даних перерахування використовується для підвищення читабельності коду);
   - physical (фізичний тип даних описує об'єкти в термінах базової одиниці, кратної базової одиниці та заданого діапазону);
2. композитні типи:
   - array (масиви використовуються для зберігання кількох значень одного і того ж типу під одним ідентифікатором);
   - record (записи використовуються для визначення одного або кількох елементів, і кожен елемент має іншу назву та інший тип).

У VHDL є такі типи операторів:

1. логічні оператори:
   - and;
   - or;
   - nand;
   - nor;
   - xor;
   - xnor;
   - not;
2. оператори порівняння
   - = (дорівнює);
   - /= (не дорівнює);
   - < (менше);
   - > (більше);
   - <= (менше або дорівнює);
   - >= (більше або дорівнює);
3. арифметичні оператори
   - + (додавання);

- - (віднімання);
- * (множення);
- / (ділення);
- & (конкатенація);
- mod (цілочисельне ділення);
- rem (залишок від цілочисельного ділення);
- abs (модуль);
- ** (зведення в степінь);

4. Оператори байтового зсуву:
   - sll (логічний зсув вліво);
   - srl (логічний зсув вправо);
   - sla (арифметичний зсув вліво);
   - sra (арифметичний зсув вправо);
   - rol (поворот вліво);
   - ror (поворот вправо).

3.3 Реалізація Кодера та декодера

На основі розрахунуів, проведених у попередньому розділі було реалізовано кодер та декодер Ріда-Соломона мовою VHDL.

Код кодера:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity RSEncoder is
        Port ( Clock : in  std_logic;
   Count255 : in  std_logic;
   Qs0 : in std_logic_vector(2 downto 0);
   Qp : out std_logic_vector(2 downto 0));
end RSEncoder;
architecture Behavioral of RSEncoder is
component flipflop is
    Port (    D: in std_logic_vector(2 downto 0) ;
          Clock : in std_logic;
          Reset : in std_logic;
          Q : out std_logic_vector(2 downto 0)) ;
end component;
component AdderXor is
    Port (    a: in     std_logic_vector(2 downto 0) ;
          b: in     std_logic_vector(2 downto 0);
          c: out     std_logic_vector(2 downto 0)) ;
end component;
component Mult is
   port( uncoded_a, uncoded_b:
   in std_logic_vector(2 downto 0);
   uncoded_multab: out std_logic_vector(2 downto 0)
   );
end component;
component mux6 IS
   Port (y1: in     std_logic_vector(2 downto 0 ) ;
          y0: in     std_logic_vector(2 downto 0 ) ;
          s: in     std_logic ;
          f: out     std_logic_vector(2 downto 0 ));
end component;
```

```
signal alpha3 : std_logic_vector(2 downto 0);
signal alpha4 : std_logic_vector(2 downto 0);
signal D1 : std_logic_vector(2 downto 0);
signal D2 : std_logic_vector(2 downto 0);
signal Q1 : std_logic_vector(2 downto 0);
signal Q2 : std_logic_vector(2 downto 0);
signal C0 : std_logic_vector(2 downto 0);

signal multa1 : std_logic_vector(2 downto 0);
signal multa2 : std_logic_vector(2 downto 0);

begin

alpha3(0) <= '0'; alpha3(1) <= '1'; alpha3(2) <= '1';
alpha4(0) <= '1'; alpha4(1) <= '1'; alpha4(2) <= '0';

ff1 : flipflop port map (D1,Clock,Count255,Q1);
ff2 : flipflop port map (D2,Clock,Count255,Q2);

add1 : AdderXor port map (Q2, Qs0, C0);
mult1 : Mult port map (C0, alpha4, multa1);
mult2 : Mult port map (C0, alpha3, multa2);

add2 : AdderXor port map(Q1, multa1, D2);

D1 <= multa2;
Qp <= Q2;

end Behavioral;
```

Код декодера:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity RSDecoder is
        Port ( Clock : in std_logic;
   Count255 : in std_logic;
```

```
    Qs0: in std_logic_vector(2 downto 0);
    Dsyn1: out std_logic_vector(2 downto 0);
    Dsyn2: out std_logic_vector(2 downto 0));
end RSDecoder;

architecture Behavioral of RSDecoder is
component flipflop is
     Port (    D: in std_logic_vector(2 downto 0);
          Clock : in std_logic;
          Reset : in std_logic;
          Q : out std_logic_vector(2 downto 0));
end component;

component AdderXor is
     Port (    a: in std_logic_vector(2 downto 0);
          b: in std_logic_vector(2 downto 0);
          c: out std_logic_vector(2 downto 0)) ;
end component;
component Mult is
    port(uncoded_a, uncoded_b:
    in std_logic_vector(2 downto 0);
    uncoded_multab: out std_logic_vector(2 downto 0));
end component;
signal alpha3 : std_logic_vector(2 downto 0);
signal alpha4 : std_logic_vector(2 downto 0);

signal D1 : std_logic_vector(2 downto 0);
signal D2 : std_logic_vector(2 downto 0);

signal Q1 : std_logic_vector(2 downto 0);
signal Q2 : std_logic_vector(2 downto 0);
signal C0 : std_logic_vector(2 downto 0);

signal multa1 : std_logic_vector(2 downto 0);
signal multa2 : std_logic_vector(2 downto 0);

begin
```

```
alpha3(0) <= '0'; alpha3(1) <= '1'; alpha3(2) <= '1';
alpha4(0) <= '1'; alpha4(1) <= '1'; alpha4(2) <= '0';

add1 : AdderXor port map (multa1, Qs0, C0);

D1 <= C0;
ff1 : flipflop port map (D1,Clock,Count7,Q1);

add2 : AdderXor port map(Q1, multa2, D2);

ff2 : flipflop port map (D2,Clock,Count7,Q2);

mult1 : Mult port map (Q2, alpha3, multa1);
mult2 : Mult port map (Q2, alpha4, multa2);

Dsyn1 <= D1;
Dsyn2 <= D2;

end Behavioral;
```

3.5 Моделювання спільної роботи кодера та декодера

У даному розділі проводиться перевірка спільної роботи кодера та декодера Ріда –Соломона. На рисунку 3.2 представлена схема цієї системи. Вона включає джерело даних, кодер, канал зв'язку, декодер і осцилограф Scope.

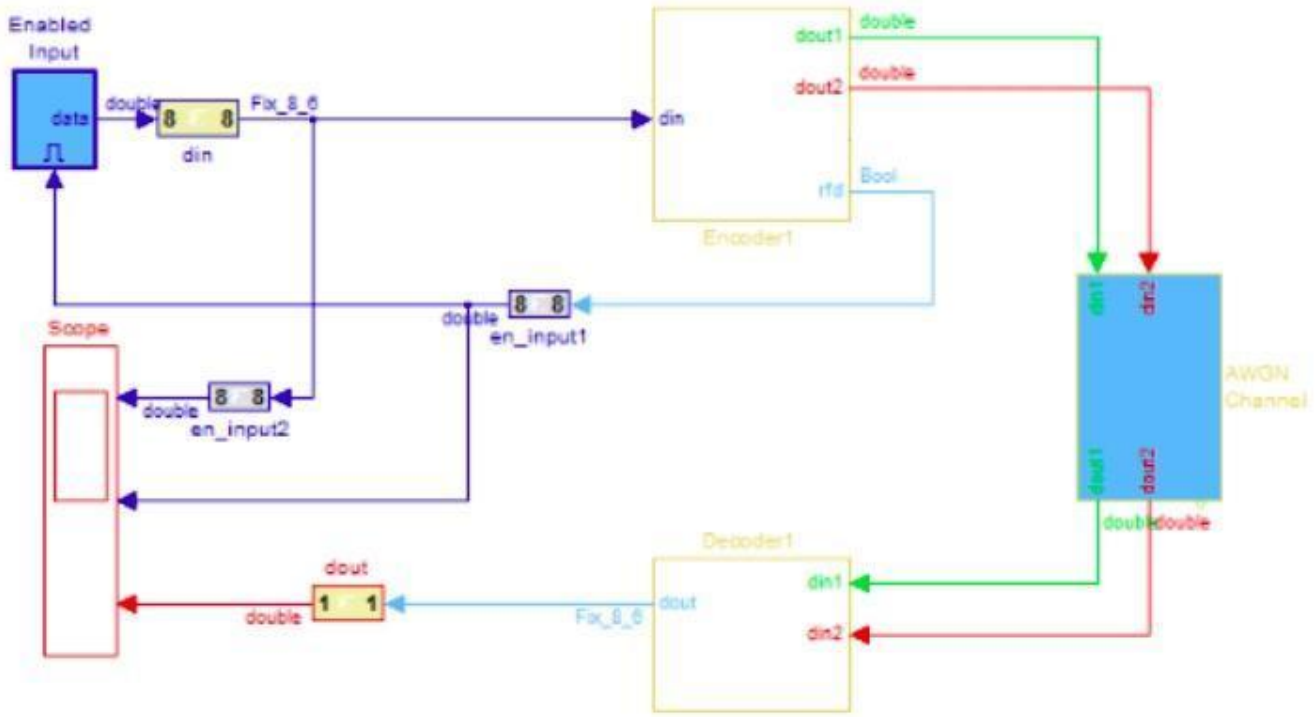


Рисунок 3.2 - Схема роботи системи з корекцією помилок

Джерело формує дискретизований і квантований за рівнями синусоїдальний сигнал. Блок кодера Encoder1 складається з кількох функціональних блоків (рис.3.4.2). з кодера Ріда-Соломона (OuterEncoder), перемежувача (interleaver), згорткового кодера (InnerEncoder) і викаливателя (puncturing). Блок затримки (EnabledDelay) необхідний для обліку часу виконання процесу кодування блоком OuterEncoder формується в кодері повного коду Ріда-Соломона (255, 251)

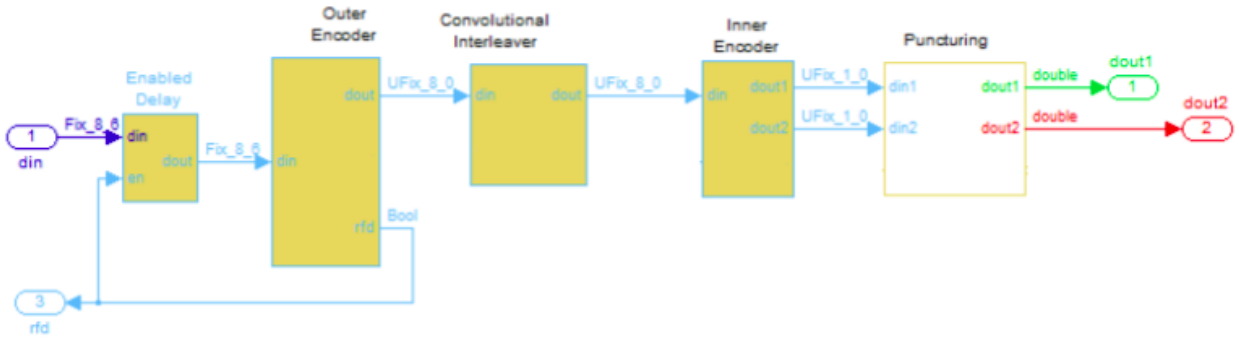


Рисунок 3.3 - Блок кодера РС

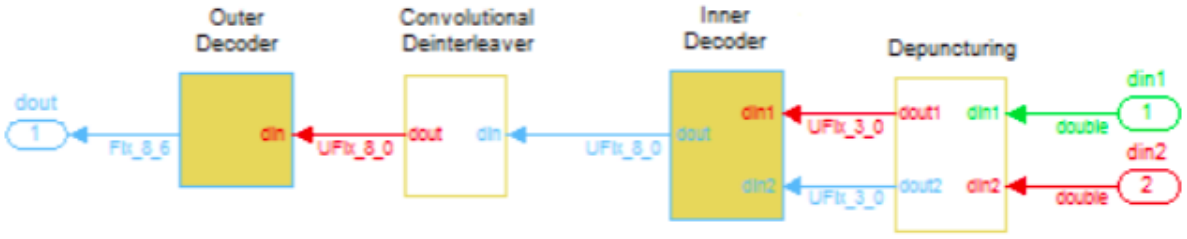


Рисунок 3.4 - Блок декодера РС

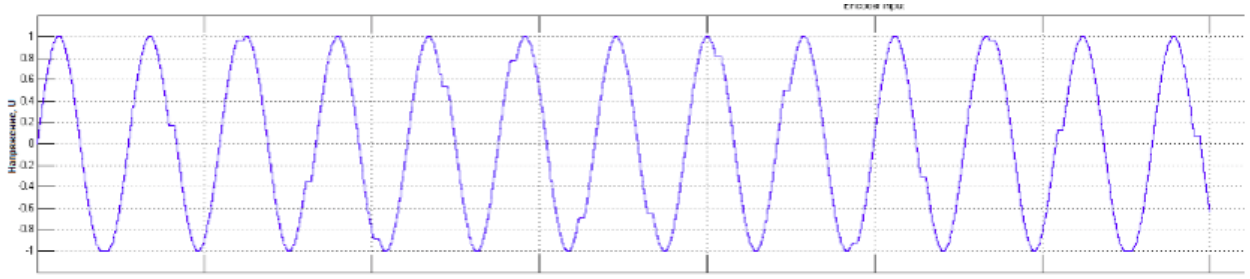


Рисунок 3.5 - Сигнал перед кодуванням

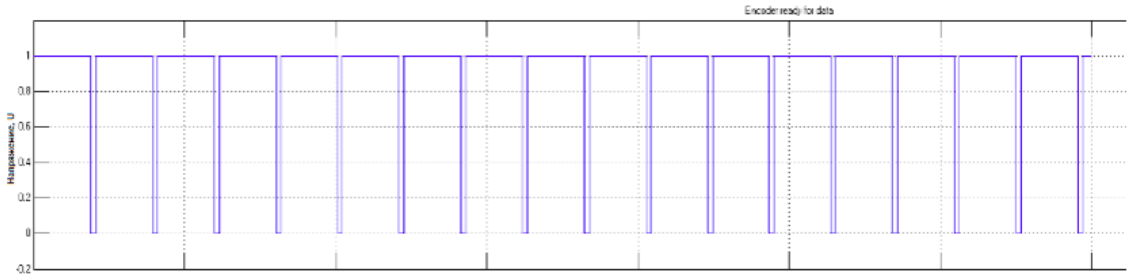


Рисунок 3.6- Сигнал про стан вхідного буферу

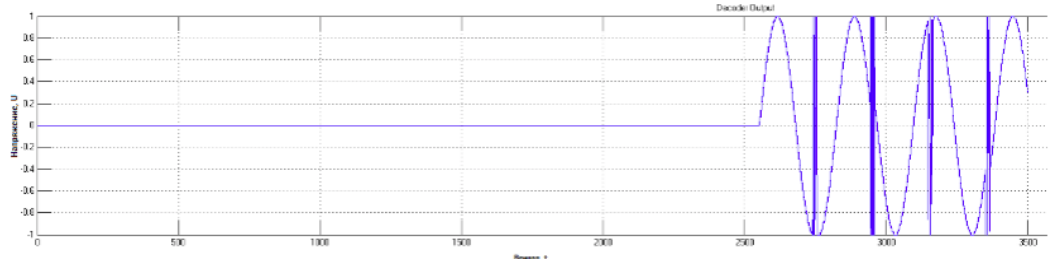


Рисунок 3.7 - Сигнал після декодування

3.6 Симуляція роботи пристрою в ModelSim

Була створена бібліотека dec_rs в середовищі ModelSim (рисунок 3.8).

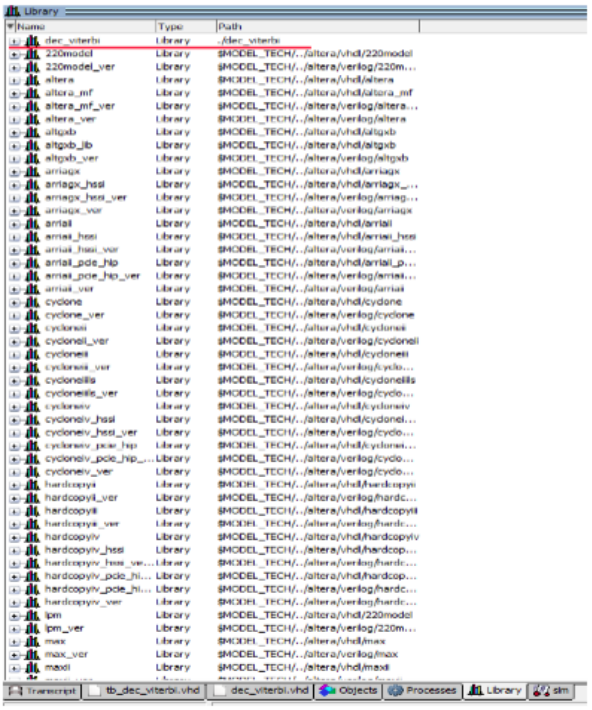


Рисунок 3.8 Створення бібліотеки dec_rs

За допомогою наступних команд були зкомпільовані вихідні файли:

vcom -work dec_rs ../packages/pkg_helper.vhd

vcom -work dec_rs ../packages/pkg_param.vhd

vcom -work dec_rs ../packages/pkg_param_derived.vhd

vcom -work dec_rs../packages/pkg_types.vhd

vcom -work dec_rs ../packages/pkg_components.vhd

vcom -work dec_rs ../packages/pkg_trellis.vhd

vcom -work dec_rs ../src/generic_sp_ram.vhd

vcom -work dec_rs ../src/axi4s_buffer.vhd

vcom -work dec_rs ../src/branch_distance.vhd

vcom -work dec_rs ../src/traceback.vhd

vcom -work dec_rs ../src/acs.vhd

vcom -work dec_rs ../src/ram_ctrl.vhd

vcom -work dec_rs ../src/reorder.vhd

vcom -work dec_rs ../src/recursion.vhd

vcom -work dec_rs ../src/dec_viterbi.vhd

vcom -work dec_rs ../testbench/txt_util.vhd

vcom -work dec_rs ../testbench/pkg_tb_fileio.vhd

vcom -work dec_rs ../testbench/tb_dec_viterbi.vhd

Запуск симуляції: vsim dec_rs.tb_dec_rs -c -do "run -all".

Результат виконання даної команди представлений на рисунку 3.9.

Параметри симуляції прочитуються з конфігураційного файлу. CLK_PERIOD: time:=10 ns; -період тактових імпульсів.
BLOCK_LENGTH_START: natural:= 200; -довжина першого блока даних.
BLOCK_LENGTH_END: natural: = 500; -Довжина останнього блоку даних.
BLOCK_LENGTH_INCR: integer:= 20; -Довжина блоку, що інкрементується до довжини наступного блоку. Це налаштування необхідне для симуляції блоків даних змінної довжини.

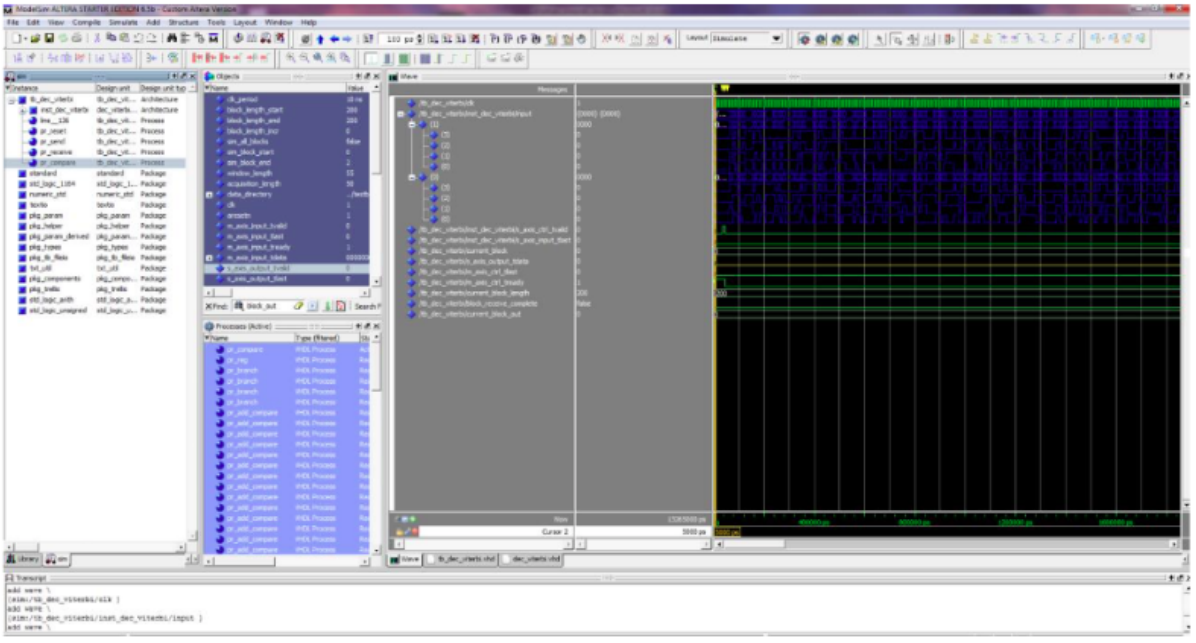


Рисунок 3.9 - Запуск симуляції декодера РС

SIM_BLOCK_START: natural:=0; -номер першого блоку даних.SIM_BLOCK_END: natural:= 10;-номер останнього блоку даних.

43-глибину декодування.

Далі були побудовані часові діаграми, представлені на рисунках 3.18 та 3.19.

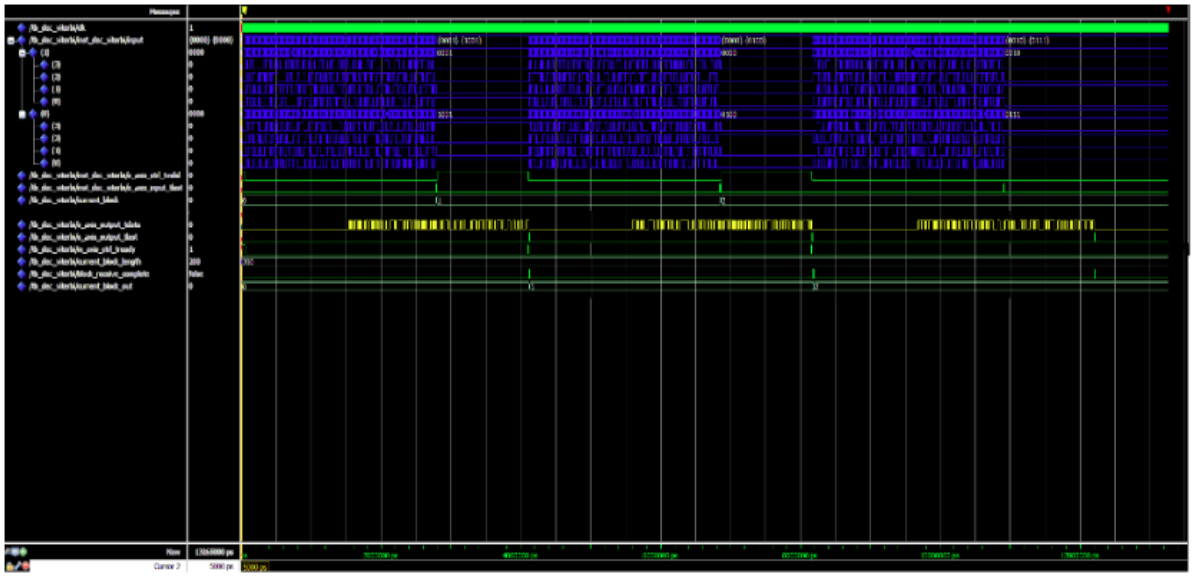


Рисунок 3.18 - Часова схема декодера

На діаграмі представлені наступні сигнали:

1) clk-тактові імпульси input-вхідний сигнал, що складається з 4-бітних м'яких рішень

2) ctrl_tvalid-імпульс, що сигналізує про неготовність декодера до прийому даних результат декодування)

3) output_tlast – покажчик початку блоку вихідних даних

4) ctrl_tready – готовність декодерак прийому даних

5) block_receive_complete – покажчик, що прийом блоку закінчено.

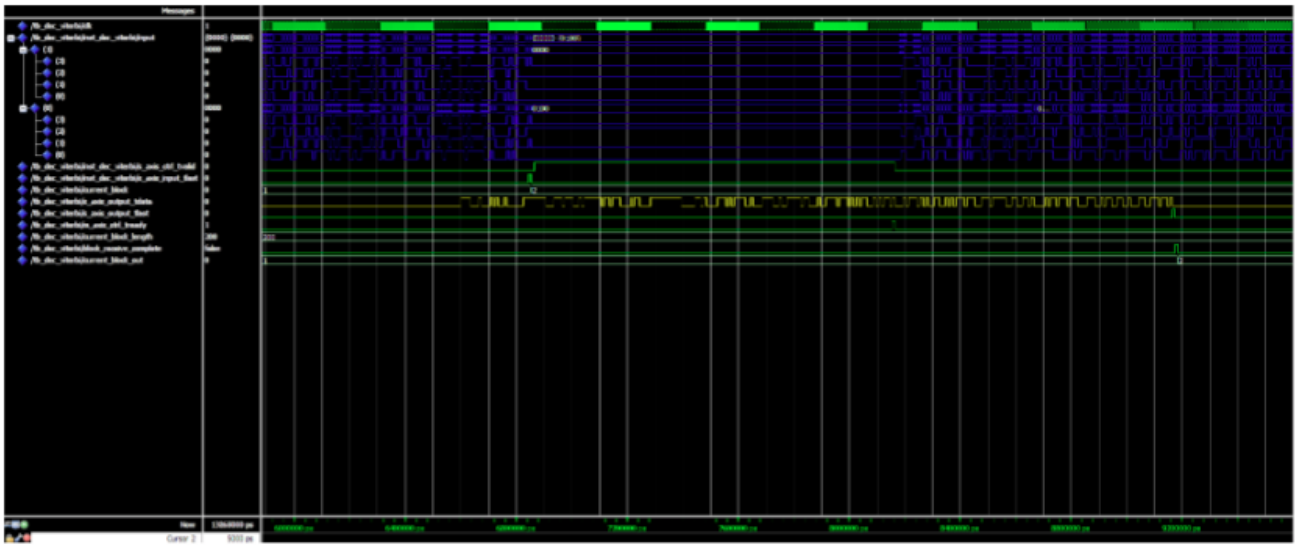


Рисунок 3.19 Часова діаграма роботи у з збільшеному масштабі

ВИСНОВКИ

В процесі виконання дипломного проєкту розроблено пристрій корекції помилок за допомогою коду Ріда-Соломона на базі FPGA та виконано моделювання його роботи.

Були досліджені корегуючі коди, їх властивості та види. Були наведені основні тезиси щодо використання та потреби корегуючи кодів.

Даний пристрій, основа якого базується на коді для корегування незалежних помилок, а саме коді Ріда-Соломона, забезпечує кодування і декодування інформаційних слів довжиною до 255 розрядів і дозволяє виправляти будь-які дві незалежні помилки в будь-якому слові.

Під час роботи над проєктом розроблена функціональна схема пристрою, вибрана оптимальна матриця декодування, та висунутий алгоритм функціонування пристрою.

Для підтвердження працездатності даного пристрою була спроєктована його модель, та проведені тести на контрольних наборах даних. При побудові моделі використовувались мови C++ та VHDL.

В результаті виконаного бакалаврського дипломного проєкту доведено, що використання корегуючих кодів для збільшення надійності ЗП та захисту даних від пошкодження є ефективним засобом.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Матвієнко М.П., Розен В.П., Закладний О.М. Архітектура комп’ютерів: Навчальний посібник . К.: Ліра-К, 2019. 264 с.
2. С.Є. Бантюков, О.В. Чаленко, В.С. Меркулов. Архітектура комп'ютерів та периферійні пристрої: Навч. посібник. Харків: УкрДУЗТ, 2018.Ч. 1. 116 с., рис. 35, табл. 2.
3. А.Я. Чураков, С.В. Шаров, О.В.Строкань. Архітектура ЕОМ. Мелітополь: РВЦ МДПУ, 2012. 195 с.
4. Blanchet G, Dupouy B. Computer Architecture . London: ISTE, 2013. 364 p.
5. Comer D. Essentials of Computer Architecture. Boca Raton, UNITED STATES: Chapman and Hall - CRC, 2017. 536 p.
6. Joseph D. Computer architecture: fundamentals and principles of computer design, University of Tennessee at Chattanooga, Chattanooga, TN, USA - CRC Press, 2017.462 p.
7. J.L. Hennessy, D.A. Patterson. Computer architecture: a quantitative approach. Amsterdam: Morgan Kaufmann, 2018. 856 p.
8. L. Chen, D. Penney, D. Jiménez. AI for computer architecture : principles, practice, and prospects. Morgan & Claypool Publishers, 2021. 142 p.
9. D.M. Harris, S.L. Harris Digital design and computer architecture. Morgan Kaufmann, 2021. - 720 p.
10. R.F. Hodson, A. Kande. Real-time expert systems computer architecture / l - CRC Press, 2018. - 300 p.
11. Bindal A. Fundamentals of computer architecture and design. Cham, Switzerland: Springer, 2019. - 606 p.
12. Ledin J. Modern Computer Architecture and Organization: Learn X86, ARM, and RISC-V Architectures and the Design of Smartphones, PCs, and Cloud Servers. 560 p.

13. G. Arroz, J. Monteiro, A. Oliveira, Instituto Superior Técnico, Portugal - Hackensack. Computer architecture : digital circuits to microprocessors. NJ: World Scientific, 2019. 750 p.

14. Микитишин А. Г., Митник М. М., Стухляк П. Д. Комп'ютерні мережі: Львів: «Магнолія 2006», 2021. 256 с. ISBN 978-617-574-087-3.

15. Тарнавський Ю.А., Кузьменко І.М.. Організація комп’ютерних мереж: підручник. КПІ ім. Ігоря Сікорського. Київ: КПІ ім. Ігоря Сікорського, 2018. – 259с.

16. Stallings W. Data and Computer Communications 10th - Pearson, 2013. 912 p.

17. Кулаков Ю.О., Луцький Г.М. Комп'ютерні мережі. Видавництво: Київ, "Юніор" 2005. С.397.

18. T. Qian, L. Chen, X. Li, H. Sun and J. Ni, A 1.25Gbps Programmable FPGA I/O Buffer with Multi-Standard Support, *2018 IEEE 3rd International Conference on Integrated Circuits and Microsystems (ICICM)*, 2018, pp. 362-365, doi: 10.1109/ICAM.2018.8596436.

19. Z. Zhang, Q. Yu, L. Njilla and C. Kamhoua, FPGA-oriented moving target defense against security threats from malicious FPGA tools, *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2018, pp. 163-166, doi: 10.1109/HST.2018.8383907.

20. N. Rezzak, J. -J. Wang, A. Cai, F. Hawley and E. Hamdy, Total Ionizing Dose Characterization of 28 nm PolarFire SONOS-based FPGA, *2019 19th European Conference on Radiation and Its Effects on Components and Systems (RADECS*), 2019, pp. 1-6, doi: 10.1109/RADECS47380.2019.9745684.

21. U. Farooq, I. Baig and B. A. Alzahrani, An Efficient Inter-FPGA Routing Exploration Environment for Multi-FPGA Systems, *IEEE Access*, vol. 6, pp. 56301-56310, 2018, doi: 10.1109/ACCESS.2018.2873041.

22. J. Yang, H. Guo, L. Chen, Y. Zhang and X. Li, SEU sensitivity evaluation of JTAG circuit used for SRAM-based FPGA, *2017 IEEE 24th International*

Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA), 2017, pp. 1-5, doi: 10.1109/IPFA.2017.8060186.

23. Z. Zhang, L. Njilla, C. A. Kamhoua and Q. Yu, Thwarting Security Threats From Malicious FPGA Tools With Novel FPGA-Oriented Moving Target Defense, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 27, no. 3, pp. 665-678, March 2019, doi: 10.1109/TVLSI.2018.2879878.

24. P. Liu, N. Xu and F. Hui, *An Automatic Method for Testing of FPGA Routing Resource,* 2018 2nd IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference (IMCEC), 2018, pp. 918-923, doi: 10.1109/IMCEC.2018.8469443.

25. N. Rezzak, J. -J. Wang, F. Hawley and E. Hamdy, Proton Characterization of RTG4 Flash-Based FPGA for LEO Environment, *2019 IEEE Radiation Effects Data Workshop*, 2019, pp. 1-5, doi: 10.1109/REDW.2019.8906605.

26. J. Lu, N. Xu, J. Yu and T. Weng, Research of timing graph traversal algorithm in static timing analysis based on FPGA, *2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC),* 2017, pp. 334-338, doi: 10.1109/ITOEC.2017.8122310.

27. G. Sung, L. Tung, H. Wang and J. Lin, USB Transceiver With a Serial Interface Engine and FIFO Queue for Efficient FPGA-to-FPGA Communication, *IEEE Access*, vol. 8, pp. 69788-69799, 2020, doi: 10.1109/ACCESS.2020.2986510.

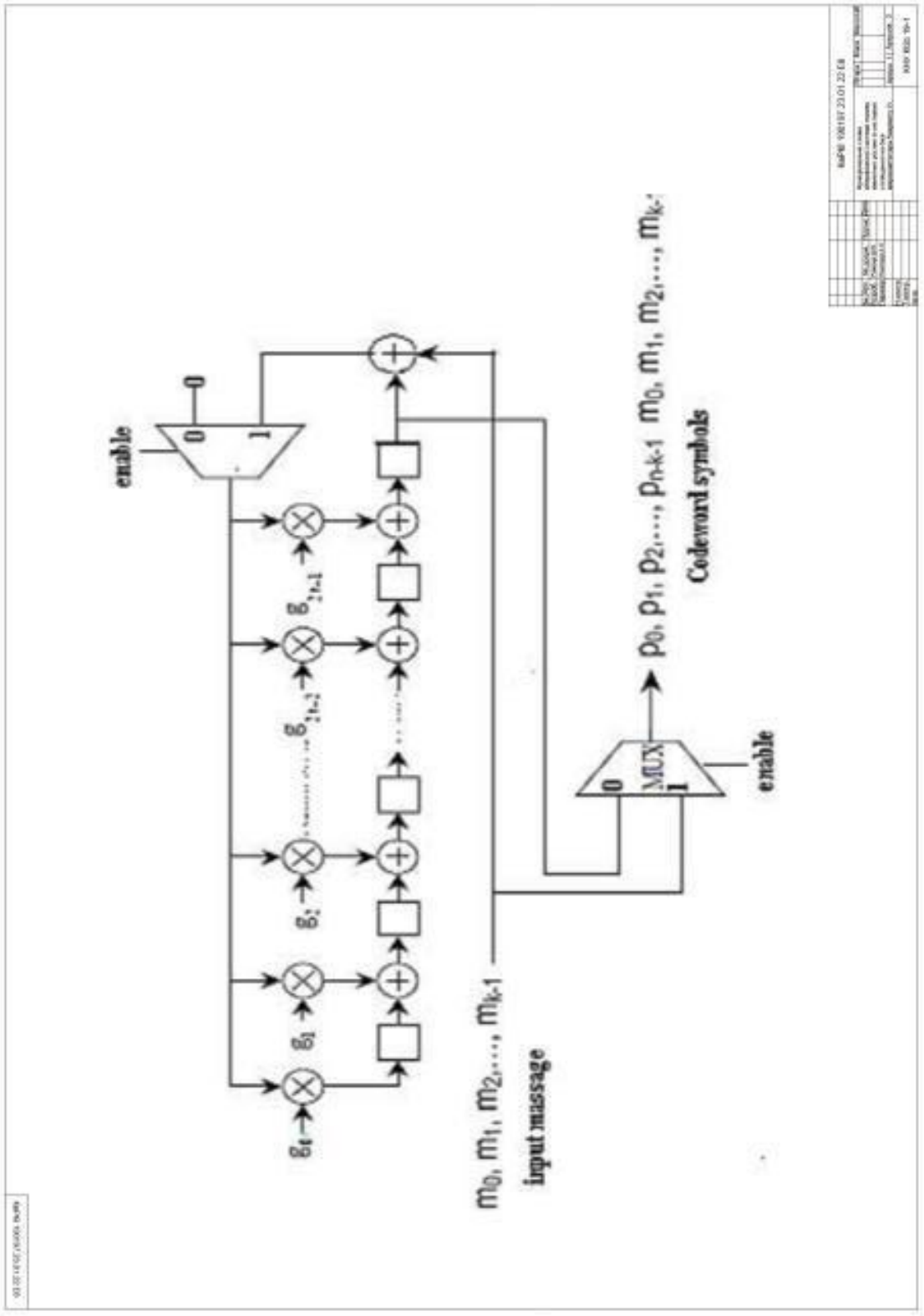
28. T. Indhumathi, R. Harshini, S. Janani and S. Navaneetha, An efficient scheme for identifying spam bots and terminate mailing, *2016 Second International Conference on Science Technology Engineering and Management (ICONSTEM)*, 2016, pp. 34-37, doi: 10.1109/ICONSTEM.2016.7560918.

29. J. Mitra and T. K. Nayak, An FPGA-Based Phase Measurement System, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 1, pp. 133-142, Jan. 2018, doi: 10.1109/TVLSI.2017.2758807.

Додаток А

(обов’язковий)

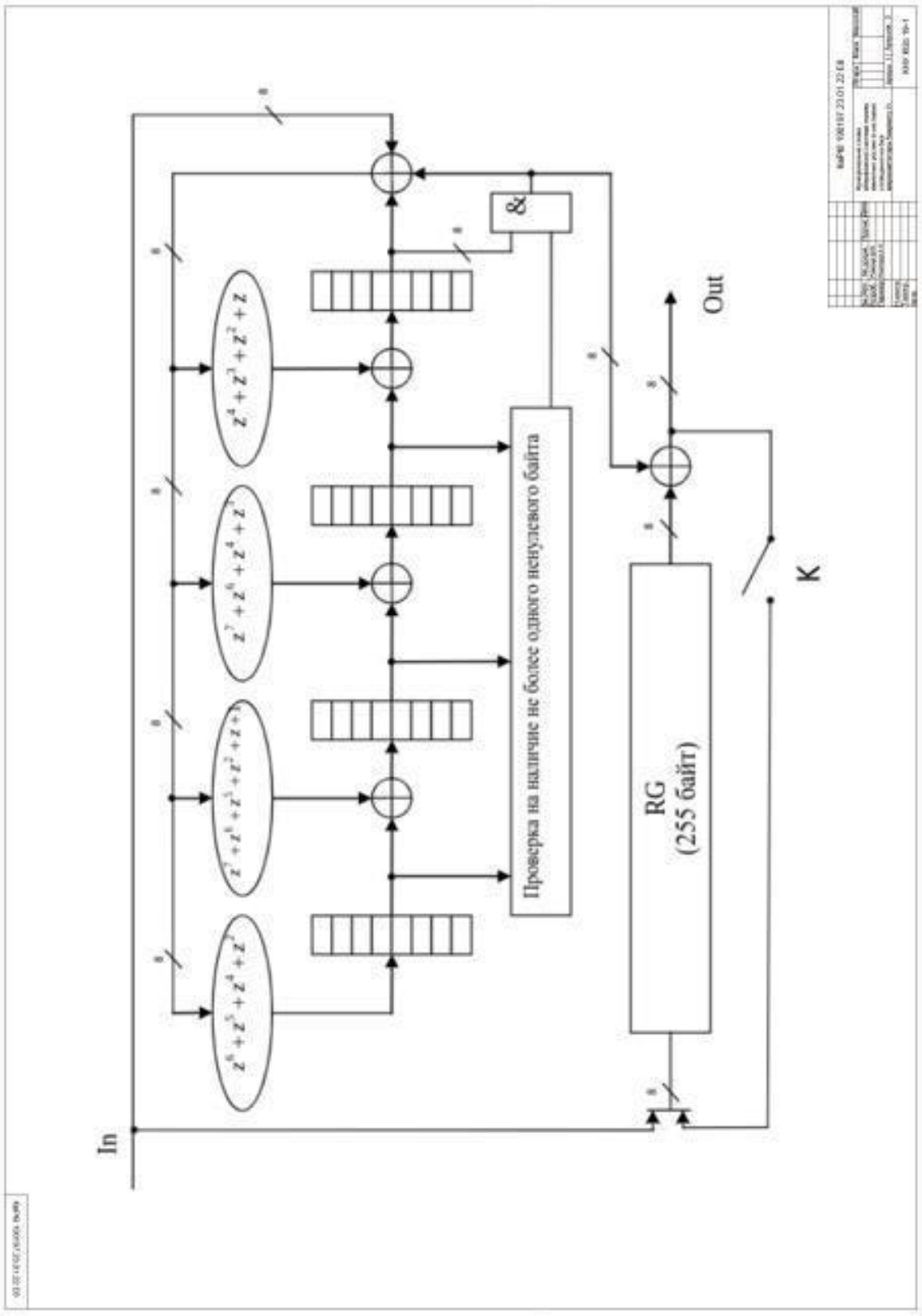
Копія креслення «Схема кодера»



Додаток Б

(обов’язковий)

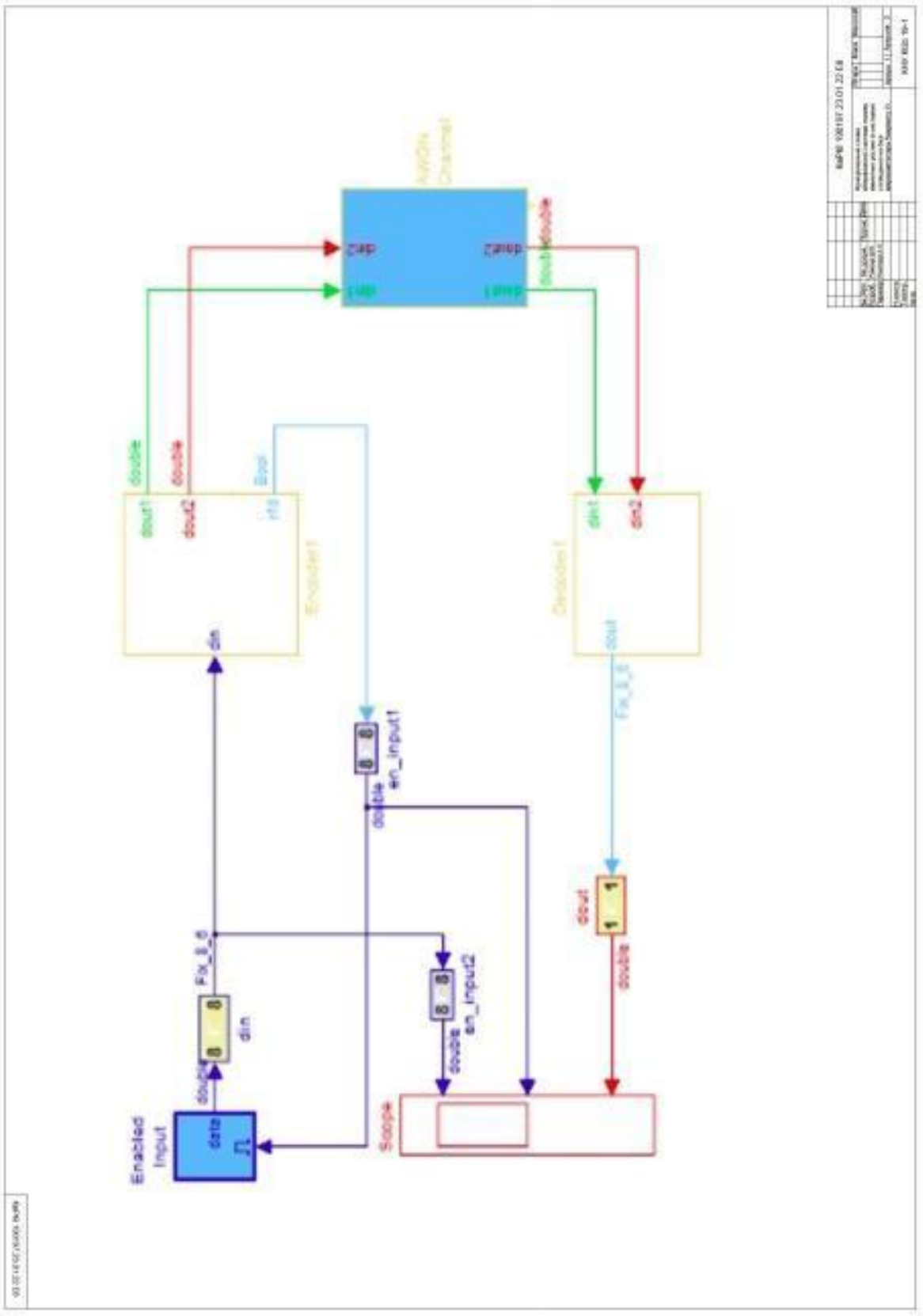
Копія креслення «Схема декодера»



Додаток В

(обов’язковий)

Копія креслення «Робота системи з корекцією помилок»



UNICHECK

Ім'я користувача:
Кафедра КН

ID перевірки:
1011625398

Дата перевірки:
21.06.2022 10:12:35 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
21.06.2022 10:13:27 EEST

ID користувача:
100005671

Назва документа: **Голубович В. КІ2с-19-1 Диплом_v2(2)_short**

Кількість сторінок: **60** Кількість слів: **11537** Кількість символів: **74103** Розмір файлу: **1.04 MB** ID файлу: **1011493128**

16.8%
Схожість

Найбільша схожість: **5.16%** з Інтернет-джерелом (https://github.com/rodrigoazs/VHDL-7-5-Reed-Solomon/blob/master/c...

12.6% Джерела з Інтернету | 168 ..... Сторінка 62

4.82% Джерела з Бібліотеки | 133 ..... Сторінка 64

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

33%
Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за **8 слів** та **0%**)

Немає вилучених Інтернет-джерел

33% Вилученого тексту з Бібліотеки | 1 ..... Сторінка 64

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи | 46

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Голубович Володимир Євгенович

Тема: Програмно-технічний модуль завадозахищеного кодування даних на базі FPGA

Спеціальність**:** 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3____ Кількість сторінок записки ___67___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є розробка пристрою завадозахищеного кодування на базі FPGA

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі кваліфікаційної роботи проведено дослідження предметної області (проаналізовано методи завадостійкого кодування, основні характеристики завадостійких кодів та розглянуто код Ріда Соломона). В другому розділі кваліфікаційної роботи проведено моделювання та проектування пристрою завадостійкого кодування на базі FPGA.

В третьому розділі кваліфікаційної роботи виконано апаратну реалізацію кодера та декодера, а саме: реалізовано кодера та декодер та проведено симуляцію роботи пристрою в ModelSim.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: недостатньо уваги приділено вибору FPGA.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на задовільному науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: задовільно

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) __Гурман_____

"___"_____2021 р. _____(підпис)

Завідувачу кафедри КІСП
д-ру техн.наук, проф. Говорущенко Т. О.

Голубовича В.Є.
ПІБ здобувача вищої освіти

ФПКТС, 4 курсу, групи КІ-17-3

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність плагіату ознайомлений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

_____ _____

дата підпис

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА СИСТЕМНОГО ПРОГРАМУВАННЯ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Програмно-технічний модуль завадозахищеного кодування даних на базі FPGA

Автор: Голубович Володимир Євгенович

Спеціальність: 123 – Компютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Яцків В.В.

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з 10-40 джерелами на один фрагмент речення;
4) в якості запозичень в окремих місцях системою зафіксовано послідовності чотирьохрозрядних двійкових кодів, які є вхідними даними до великої кількості задач і не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;
5) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ідентичності/схожості, складає 16,8% і адресується до 301 першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ В.В. Яцків

Гарант ОП _____ С. М. Лисенко

Завідувач кафедри КІСП _____ Т. О. Говорущенко