Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

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

КвРКІ.2001137.20.01.05 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Виконав: студент III курсу, група КІ2с–20–1 _____ І. С. Охріменко
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 інформаційних систем _____ Т.О. Говорущенко
Підпис — Ініціали, прізвище

« 26 » червня 2023 р.

Хмельницький 2023

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 «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.Ю.Говорущенко

" 11 " 01 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Охріменку Іллі Сергійовичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Програмно-технічний модуль завадозахищенного кодування даних на базі FPGA

Керівник проекту (роботи) Яцків В.В., д.т.н.,професор

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.03.2023 р. № 15

2. Строк подання студентом проекту (роботи) на кафедру 26.06.2023 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

Моделювання та проєктування програмно-технічного модулю завадозахищенного кодування на базі FPGA

Програмна-апаратна реалізація програмно-технічного модулю завадозахищенного кодування на базі FPGA

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

Опис герконового енкодеру Соломона

Опис герконового енкодеру Соломона

Схема роботи системи з корекцією помилок

6.

6. Консультанти розділів дипломного проекту (роботи)

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

7. Дата видачі завдання «_1_»_____03_____2023 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітк
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	20.02.2023	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.03.2023	виконан
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	10.03.2023	виконан
4	Робота над розділом 2 – елементна база кіберфізичної системи програмно-технічного модуля завадозахищенного кодування даних на базі FPGA	20.04.2023	виконан
5	Робота над розділом 3 –апаратна реалізація коду Ріда-Соломона	30.04.2023	виконано
6	Оформлення пояснювальної записки згідно вимог	11.05.2023	виконано
7	Попередній захист ВКР	26.05.2023	виконано
8	Захист ВКР на засіданні ЕК	Червень 2023 року	

Студент Підпис І. С. Охріменко Ініціали, прізвище

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

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ.2001137.20.01.05 ПЗ	Пояснювальна записка	63		
			Графічні матеріали			
2		КвРКІ.2001137.20.01.05Е8	Апаратна реалізація кодера Ріда-Соломона	1		
3		КвРКІ.2001137.20.01.05Е8	Апаратна реалізація декодера Ріда-Соломона	1		
4		КвРКІ.2001137.20.01.05Е8	Схема роботи системи з корекцією помилок	1		

Зм	Арк	№ докум	Підпис	Дата	КвРКІ.2001137.20.01.05ВП		
Розробив		Охріменко		26.06	Відомість проекту	Літера	Аркуш	Аркушів
Перевір.		Яцків				У	1	1
Н. контр.		Лисенко		2606.		ХНУ, КІ2с-20-1		
Затв.		Говорущенко		26.06				

АНОТАЦІЯ

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

Автор роботи: Охріменко Ілля Сергійович.

Керівник роботи: Яцків Василь Васильович

Пояснювальна записка: 63 с., 38 рис., 3 дод., 40 джерел.

Графічна частина: 3 креслення.

ЗАВАДОЗАХИЩЕНЕ КОДУВАННЯ. КОД РІДА СОЛОМОНА. FPGA.

Метою роботи є розробка програмно-технічного модулю завадозахищеного кодування даних на базі FPGA.

Об'єктом дослідження є програмно-технічний модуль завадозахищеного кодування даних на базі FPGA.

Предметом дослідження є процес проектування та моделювання програмно-технічного модулю завадозахищеного кодування даних на базі FPGA.

Практичне значення отримав спроєктований програмно-технічний модуль завадозахищеного кодування даних на базі FPGA, що має можливість бути реалізованим для завадостійкої передачі даних.

Підпис студента

23.06.23

Дата

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ 4
ВСТУП 5
1 АНАЛІЗ ВІДОМИХ ЗАСОБІВ ТА РІШЕНЬ 7
1.1 Визначення технології, що забезпечує надійне кодування, яке стійке до перешкод 7
1.2 Класифікація методів кодування 12
1.3 Завадостійкі коди 14
1.4 Код Ріда-Соломона 24
1.5 Постановка задачі 26
1.6 Висновки 27
2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ПРИСТРОЮ ЗАВАДОСТІЙКОГО КОДУВАННЯ НА БАЗІ FPGA 28
2.1 Методи представлення коду Ріда-Соломона 28
2.2 Кодер 36
2.3 Декодер 38
2.4 Висновки 42
3 АПАРАТНА РЕАЛІЗАЦІЯ КОДУ РІДА-СОЛОМОНА 44
3.1 Програмована користувачем вентильна матриця (FPGA) 44
3.2 Мова проектування VHDL 45
3.3 Опис розробки кодера та декодера 50
3.4 Створення моделі, яка відображає взаємодію кодера і декодера 51
3.5 Синтез і симуляція програмно-технічного модулю 52
3.6 Висновки 65
ВИСНОВКИ 67
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ 68
Додаток А Копія креслення « Апаратна реалізація кодера Ріда-Соломона» 72

Додаток Б Копія креслення « Апаратна реалізація декодера Ріда-Соломона» .. 73
Додаток В Копія креслення « Схема роботи системи з корекцією помилок».... 74

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

РС - код Ріда-Соломона

FPGA- field-programmable gate array

ЛПС - лінійні послідовні систе

ВСТУП

Протягом останніх десятиліть ми спостерігаємо швидкі зміни в різних сферах суспільного життя, які характеризуються величезним приростом різноманітної інформації. Ця інформація охоплює соціально-політичну, виробничу, наукову, культурну та інші галузі. Міжнародний досвід свідчить, що інформаційний потенціал суспільства стає все більш важливим, визначаючи його економічний розвиток нарівні з матеріальним виробництвом. Результати інтелектуальної діяльності стають ключовими для економічного прогресу такого суспільства. Це пояснюється зростаючою роллю інформації, яка стає визначальним фактором у розвитку різних сфер суспільства.

З появою обчислювальних систем виникло питання про надійність зберігання та передачі інформації. З часом людство все більше використовує різноманітні обчислювальні системи для вирішення своїх потреб. Ці системи мають різне призначення та застосовуються в різних галузях. Термін "обчислювальні системи" має широке значення і включає в себе складні мікропроцесорні кластери та засоби дистанційної передачі інформації.

У різних галузях використовуються обчислювальні системи, наприклад, у побуті, науковій діяльності, міжвідомчих повідомленнях, промисловості та оборонному комплексі. Якщо система застосовується у важливішій галузі, її вимоги до функціональності і надійності стають більш жорсткими. Часом комп'ютер може неправильно зберігати інформацію через різні причини, такі як сплески напруги на електромережі та інші фактори. Також передача даних може супроводжуватися різними помилками.

Один зі способів забезпечити надійність передачі та збереження інформації - це застосування завадостійкого кодування. Це означає, що до вихідних даних додаються додаткові символи, які допомагають перевірити правильність передачі та виявити та виправити помилки. Цей метод розвивали ще з початку другої половини двадцятого століття, а роботу в цьому напрямку внесли Клод Шеннон та Річард Геммінг. Сьогодні цей метод застосовується в пакетних мережах,

телекомунікаційних технологіях та супутникових системах зв'язку. Існує багато різних видів завадостійкого кодування, які можуть відрізнятися за складністю та іншими параметрами.

Доступні коди поділяються на два типи: блокові та безперервні. У блокових кодах інформаційна послідовність поділяється на окремі блоки, кожен з яких містить додаткові стабілізуючі символи. Окремі блоки кодуються та декодуються незалежно. У безперервних кодах інформаційні символи не розділяються на блоки, а стабілізуючі символи розміщуються між ними в певному порядку. Кодування та декодування відбувається безперервно.

За фіксованої довжини коду та кількості розрядів, не існує коду з більшою мінімальною відстанню, ніж у коду Ріда-Соломона. Це є вагомим аргументом для використання цього методу надійного кодування.

Метою цієї роботи є розробка кодера та декодера, які базуються на алгоритмі коду Ріда-Соломона. Цей алгоритм призначений для знаходження та виправлення помилок, які виникають під час передачі даних. Він спроможний виявити та виправити однопакетні помилки, а також знаходити двухпакетні помилки, що допомагає покращити надійність передачі даних.

1 АНАЛІЗ ВІДОМИХ ЗАСОБІВ ТА РІШЕНЬ

1.1 Визначення технології, що забезпечує надійне кодування, яке стійке до перешкод

Кодування сигналу включає його представлення в певній формі, яка є зручною або придатною для обміну повідомленнями. Під час кодування елементи повідомлення перетворюються на відповідні кодові символи. Наприклад, для передачі 33 літер алфавіту потрібно використовувати цифри від 0 до 32. Для представлення будь-якого числа в десятковій формі потрібно 10 цифр від 0 до 9 (у цьому випадку використовуються 10 основних сигналів). Кодування на стороні передачі завжди включає використання процедури декодування, щоб відновити повідомлення за отриманими кодовими символами.

Код - це повний набір символів, які використовуються для кодування повідомлення. Відношення між символами в початковому алфавіті та їх кодовими комбінаціями утворюють таблицю відповідності або кодову таблицю. Множину можливих кодових символів, які є елементарними сигналами, називають кодовим алфавітом, а їх кількість m є основою коду [2].

Код з основою m = 2 називається бінарним, а коди з іншими основами вважаються багатопозиційними. При використанні основи m, правила кодування K елементів повідомлення зводяться до правил запису K різних чисел у системі числення з основою m.

Прості електричні сигнали, які використовуються для кодування повідомлень, є імпульсами струму або напруги. Під час кодування кожен елемент повідомлення представляється в певному кодовому наборі символів, відомому як кодова комбінація. Кількість бітів n, які утворюють кодову комбінацію, визначає розрядність або довжину коду.

Для ефективного кодування вихідного повідомлення необхідно встановити певні правила. Сукупність цих правил називається системою кодування. Існує багато різних способів вираження такої системи кодування.

Один з зручних способів кодування - використання кодової таблиці, яка містить алфавіт кодованих повідомлень і відповідні кодові комбінації. Для кодування дискретних повідомлень часто використовується двійковий код, який має особливе значення в техніці. Це пояснюється тим, що наявність або відсутність посилки є дуже очевидними і легко помітними. Особливо просто це спостерігати, коли пристрій може перебувати у двох можливих станах, наприклад, відкритому або закритому.

Зокрема, існують пристрої з двома постійними станами, як, наприклад, електронні перемикачі або тригери, які можуть зберігати один біт інформації. Кількість таких пристроїв, позначених як N, визначає кількість бітів. Тому двійковий код використовується не тільки у системах зв'язку, але й у комп'ютерах, автоматизації та інших областях.

Кількість бітів, необхідних для кодування повідомлення, залежить від максимальної кількості рівнів масштабування (квантування) U_{max} і визначається формулою $n = \log_2 N_{max}$. Якщо кодова група містить n символів 0 і 1, то числа до $N_{max} = 2^n$ можуть бути закодовані за допомогою n-бітового двійкового коду.

Якщо всі кодові слова мають однакову кількість символів, то такий код називається однорідним, в іншому випадку - нерівномірним. Довжина кодової комбінації залежить від кількості окремих елементів, які вона містить. Наприклад, код з трьома символами називається трибітним кодом, або код з п'ятьма символами - п'ятирозрядним або п'ятиелементним кодом [4].

Код Бодо (рисунок 1.1) є кодом з основою m = 2 і довжиною кодової комбінації n = 5. В цьому коді елементами є сигнал і відсутність сигналу, які мають однакову тривалість і абсолютне значення, але відрізняються полярністю.

Код Бодо є однорідним кодом, де кожна літера відповідає п'яти елементарним символам. Всього можна скласти N = 2^5 = 32 комбінації. Для збільшення кількості символів у коді Бодо використовується другий регістр з 32 символами.

Керуючі символи	
o. ...	ПРОБІЛ, ПЕРЕЙТИ ДО ТАБЛИЦІ БУКВ
.o ...	ПРОБІЛ, ПЕРЕЙТИ ДО ТАБЛИЦІ ЦИФР
oo ...	ВИДАЛИТИ ОСТАННІЙ ЗНАК

Таблиця букв										Таблиця цифр									
..	o.	A	oo	o.	K					..	o.	1	o.	o.	.				
..	oo	É	oo	oo	L					..	.o	2	o.	.o	9/				
..	.o	E	oo	.o	M					..	..o	3	o.	..o	7/				
..	.oo	I	oo	.oo	N					..	o.o	4	o.	o.o	2/				
..	ooo	O	oo	ooo	P					..	ooo	5	o.	ooo	'				
..	o.o	U	oo	o.o	Q					..	oo.	1/	o.	oo.	:				
..	..o	Y	oo	..o	R					..	.oo	3/	o.	.oo	?				
.o	..o	B	o.	..o	S					.o	o..	6	oo	o..	(				
.o	o.o	C	o.	o.o	T					.o	.o.	7	oo	.o.	)				
.o	ooo	D	o.	ooo	V					.o	..o	8	oo	..o	-				
.o	.oo	F	o.	.oo	W					.o	o.o	9	oo	o.o	/				
.o	.o.	G	o.	.o.	X					.o	ooo	0	oo	ooo	+				
.o	oo.	H	o.	oo.	Z					.o	oo.	4/	oo	oo.	=				
.o	o..	J	o.	o..	—					.o	.oo	5/	oo	.oo	£				

Рисунок 1.1 – Код Бодо

Один з відомих прикладів нерівномірних кодів - азбука Морзе (рисунок 1.2). У цьому коді символи мають різну довжину кодових слів: найпоширенішим символам надаються найкоротші кодові послідовності, а менш частим символам - довші кодові послідовності.

У азбуці Морзе використовуються тільки дві комбінації символів 1 і 0: одиночний (1 і 0) або потрійний (111 і 000). Коли сигнал кодується як одна одиниця (короткий пакет), його представлення в азбуці Морзе відповідає

символу точки (.). У випадку, коли сигнал кодується як три одиниці (пакет, що триває утричі довше), він представляється як тире (довгий сигнал або пауза, що відповідає відсутності струму).

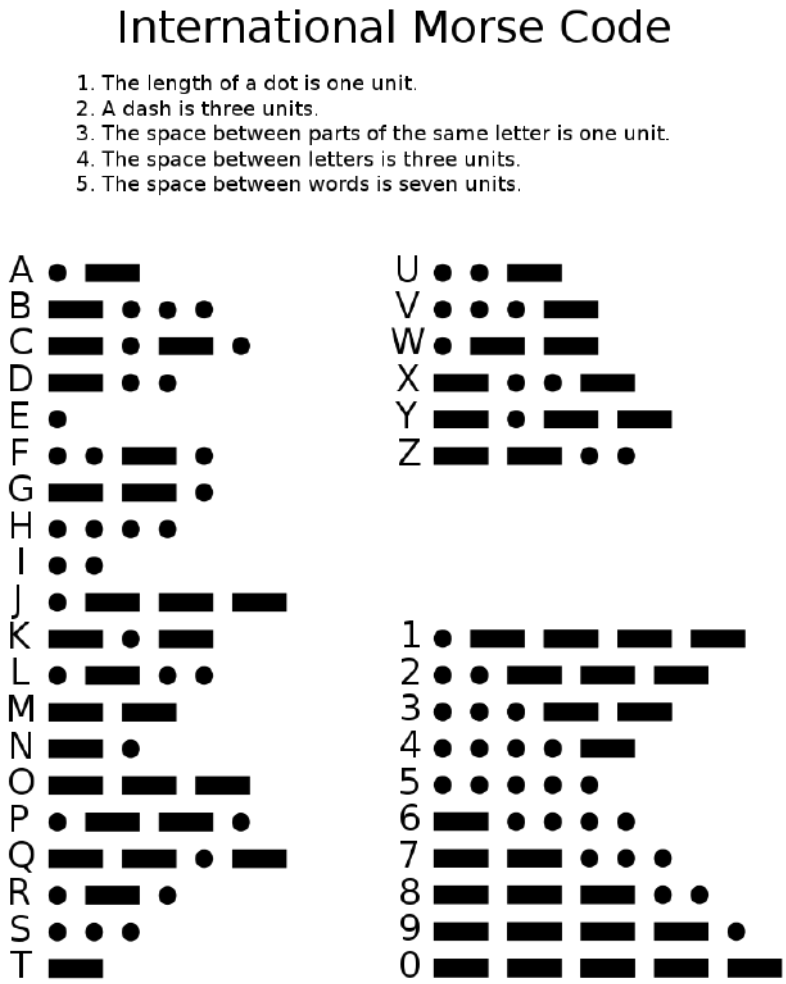


Рисунок 1.2 – Код Морзе

Символ 0 в азбуці Морзе використовується для розділення крапок від тире, окремих крапок між собою і тире від тире. Комбінація символів 000 використовується як розділовий знак між різними кодами. Крім того, для розділення слів використовується комбінація символів 00000. Азбука Морзе була розроблена на основі статистики використання англійської мови.

Варто зазначити, що азбука Морзе не є двійковим кодом з використанням крапок та тире, як може здатися на перший погляд. Насправді, вона представляє

собою потрійний алфавіт, який включає крапку, тире і пробіл. Кожен символ кодується через ці три елементи, що надає ширший набір можливих комбінацій для представлення символів.

У радіозв'язку, за допомогою азбуки Морзе, передавач передає радіоімпульси різної тривалості, що відповідають крапкам і тире кодової комбінації. Приймач використовує спеціальний генератор і змішувач для перетворення радіоімпульсів на звукові коливання. Оператор може вибрати зручну для себе частоту звукових коливань, регулюючи частоту генератора. В слуховому телеграфному зв'язку швидкість передачі інформації може залежати від навичок оператора і, зазвичай, становить приблизно кілька знаків на хвилину.

Однією з головних переваг коду Морзе є його висока стійкість до завад прийому на слух, завдяки чому він широко використовується в радіозв'язку. При отриманні сигналу на слух цим кодом займається досвідчена людина, яка може відрізнити сигнали, навіть якщо вони були значно спотворені або заваджені.

У теорії інформації було доведено, що як для неортогональних, так і для ортогональних сигналів ймовірність помилки при оптимальному прийомі залежить від кількості символів та відношення сигнал-шум. Це визначається параметром $Q_{вих0}$, що представляє відношення енергії сигнального елемента до спектральної щільності потужності білого шуму. З цього випливає, що для досягнення безпомилкової передачі повідомлень (тобто ймовірність помилки p рівна нулю) необхідно необмежено збільшити значення параметра $Q_{вих0}$ (збільшити силу сигналу).

Зокрема, якщо ми знаємо середню потужність сигналу P_c і спектральну щільність потужності білого шуму W_0, можливо встановити, що це станеться лише при наближенні швидкості передачі інформації v до нуля. Це пояснюється тим, що енергія сигнального елемента E_0 залежить від середньої потужності сигналу P_c та тривалості сигналу T_s, де $T_s = l / v$, де l - довжина сигналу.

Клодом Шенноном було встановлено, що передача сигналів без помилок можлива, якщо швидкість передачі менше пропускної здатності каналу зв'язку.

Пропускна здатність залежить від відношення сигналу до шуму на вході приймача - чим вище це відношення, тим більша пропускна здатність. З цього випливає, що енергетика ліній зв'язку визначає лише їх пропускну здатність. А за допомогою спеціально розроблених кодів можна забезпечити високу стійкість до завад. Це означає, що застосування таких кодів дозволяє досягти дуже високого рівня захисту від помилок передачі.

Ідеї Клода Шеннона, які були революційними для свого часу, змінили уявлення зв'язківців про передачу сигналів. Раніше вважалося, що єдиними способами підвищення стійкості до завад є збільшення потужності передавача або повторна передача повідомлення. Однак ці методи призводили до неефективного використання пропускної здатності каналу зв'язку. Фактично, Шеннон твердив, що швидкість передачі обмежується потужністю сигналу, шумом в каналі і шириною смуги частот, а не точністю передачі. Це означає, що точність передачі можна підвищити без збільшення потужності сигналу, просто використовуючи канал більш ефективно.

Шенноном були знайдені необхідні і достатні умови для зниження ймовірності помилки до нуля. Він досяг цього для різних моделей каналів зв'язку, таких як канали з обмеженою смугою частот, з використанням різних форм сигналів, надлишкових джерел, джерел безперервних повідомлень, перевірки вірності окремих повідомлень тощо. Його результати не обмежувалися лише ймовірністю помилки символу, а включали інші аспекти передачі інформації.

В цьому і полягає сутність теорії інформації і теорії кодування. К. Шеннон показав, що такі потрібні коди існують, але не надав конкретних методів їх знаходження.

1.2 Класифікація методів кодування

Можна класифікувати методи кодування та коди, які використовуються в техніці зв'язку, за допомогою специфічних ознак. У представленій класифікації

показано різні типи кодів (рисунок 1.3). За їхнім призначенням кодування може бути простим, економним або стійким до завад.

Рисунок 1.3 – Класифікація основних методів кодування і кодів

- стиснення інформації (зменшення або повне усунення надмірності, що міститься в повідомленні) - економне (ефективне) кодування;

- виявлення або виправлення помилок, що виникають в каналі зв'язку через перешкоди і спотворень сигналу, - перешкодостійке (надлишкове) кодування. Завадостійке кодування часто називають канальним кодуванням.

Примітивне кодування використовується для збігу алфавіту джерела з алфавітом дискретного каналу і часто застосовується для шифрування переданої інформації, захисту її від несанкціонованого доступу та підвищення стійкості систем зв'язку. Термін "примітивне" використовується в теорії кодування і вимагає математичного аналізу. Важливою властивістю примітивного кодування є те, що надмірність дискретного джерела, яке формується в результаті примітивного кодування, дорівнює надмірності джерела на вході кодера. Простіше кажучи, при примітивному кодуванні символ одного алфавіту замінюється символом іншого алфавіту [1].

Первинні коди, що отримуються при примітивному кодуванні, мають властивість, що окремі кодові комбінації відрізняються один від одного лише

одним елементом. Тому навіть один помилково прийнятий елемент у кодовій комбінації призводить до заміни цієї кодової комбінації іншою, що призводить до неправильного прийому всієї кодової комбінації в цілому.

Шифрування, що використовується при примітивному кодуванні, має на меті забезпечити конфіденційність зв'язку, запобігти розумінню повідомлення несанкціонованим користувачем і уникнути введення помилкових повідомлень до системи. В такому випадку правила примітивного кодування обираються таким чином, щоб ймовірність виникнення на виході кодера довгої послідовності, складеної лише з одиниць або лише з нулів, була мінімальною. Такий кодер називається скремблером (від англ. "scramble" - перемішувати).

Скремблювання використовується для надання потоку символів властивостей, які нагадують випадкову послідовність незалежних двійкових символів. Цей процес здійснюється шляхом пропускання символьного потоку через регістр зсуву з внутрішнім зворотним зв'язком, де символи "перемішуються" і перетворюються. Важливо, щоб характеристика скремблера була оборотною. Таким чином, прийняті символи після демодуляції піддаються дескремблюванню - зворотній операції - для відновлення початкової послідовності. Скремблювання часто використовується для покращення надійності системи синхронізації, а також поліпшення роботи декодера, зокрема в системах зі зворотним зв'язком.

1.3 Завадостійкі коди

У будь-якому каналі зв'язку, який має обмежену кількість смуг, частот, часу передачі і динамічний діапазон, існує максимальна пропускна здатність. Пропускна здатність визначається як найбільша кількість бітів, які можуть бути передані за одиницю часу при мінімальній ймовірності помилок. Фактична кількість бітів, які передаються за одиницю часу, називається швидкістю передачі. Однак, при необмеженій ймовірності помилок, швидкість передачі

завжди менша за пропускну здатність. Це означає, що неможливо досягти максимальної пропускної здатності без будь-яких помилок передачі.

У каналі з помилками, максимальну швидкість передачі можна досягти за допомогою завадозахищеного кодування. Цей метод включає введення додаткової інформації в переданий сигнал, таку як додатковий час, частота або амплітуда. Якщо код є сумісним з каналом, що означає, що він може виправити найбільш ймовірні помилки, то введена додаткова інформація є цілком обґрунтованою. Проте, якщо код не відповідає каналу, то помилки можуть не тільки не бути виправлені, але й розмножитись через кодування. У такому випадку використання завадозахищеного кодування не принесе користі, а може навіть завдати шкоди. Для того, щоб забезпечити відповідність коду і каналу зв'язку, необхідно мати максимальну кількість інформації про можливі перешкоди в каналі.

На сьогоднішній день було розроблено багато різних завадостійких кодів, які відрізняються один від одного за різними характеристиками. Ці характеристики включають основу коду, відстань між кодовими словами, надлишковість, структуру, функціональне призначення, енергетичну ефективність, кореляційні властивості, алгоритми кодування і декодування, форму частотного спектру і т.д. На рисунку 1.4 представлені типи кодів, які відрізняються за особливостями їх структури, функціональним призначенням і фізичними властивостями як сигналу.

Найважливішим підкласом безперервних кодів є згорткові коди, які відрізняються своєю конструкцією та ширшим спектром застосування порівняно з іншими безперервними кодами. Загалом, чим довший код з постійною надлишковістю, тим більша його відстань і вища стійкість до завад. Однак, довгі коди можуть бути складні у реалізації. Компромісним рішенням для цієї проблеми є складові коди, зокрема каскадні та похідні коди.

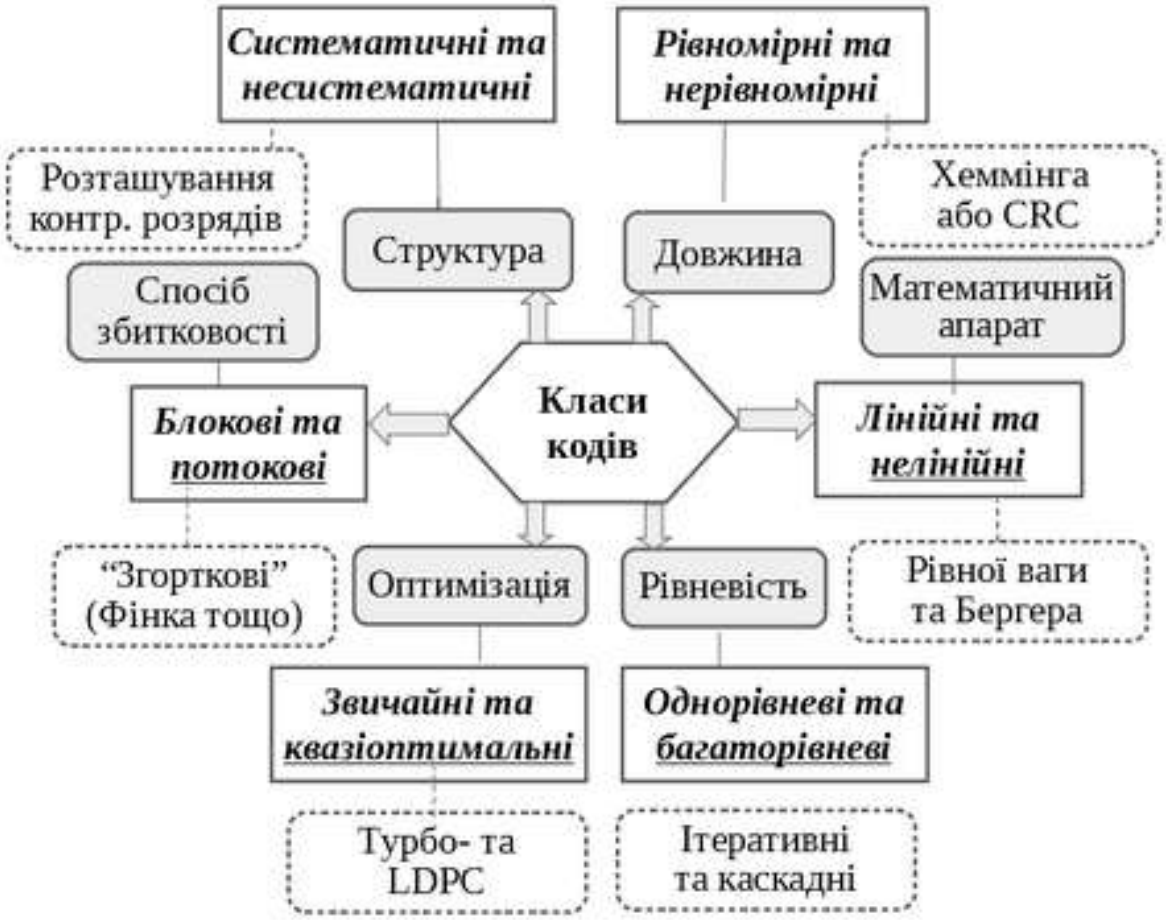


Рисунок 1.4 – Класифікація завадостійких кодів

Зазвичай каскадний код складається з двох рівнів або каскадів: внутрішнього та зовнішнього. Сигнали передаються по лінії зв'язку за допомогою внутрішнього коду $n_{вн}$, де символьні слова є символами зовнішнього коду, який має довжину $n_{зн}$. Основою зовнішнього коду є $q_{вн}^{k}$. Похідні коди можна побудувати у формі матриці, де рядки відповідають словам першого коду, а стовпці - тому ж або іншому коду.

При генерації каскадного коду вхідна послідовність символів розбивається на блоки, кожен з них містить $k_{вн}$ символів. Кожен блок порівнюється з інформаційним символом зовнішнього коду, який належить до алфавіту з $q_{вн}^{k}$ можливих значень. Потім інформаційні символи зовнішнього коду $k_{вн}$ перетворюються на блоки символів зовнішнього коду $n_{зн}$, а потім ці блоки

символів зовнішнього коду nзн перетворюються на блоки символів внутрішнього коду $k_{вн}$.

Існує кілька варіантів каскадних кодів: блокові зовнішні і внутрішні коди, зовнішні блочні коди, внутрішні згорткові коди, зовнішні згорткові коди, внутрішні блокові коди, а також комбінація згорткових кодів для обох рівнів.

Один з найпоширеніших способів створення похідних кодів полягає у записуванні послідовності вхідних символів у різні рядки матриці. Цей метод включає запис k_1 символів в кожен k_2 рядок матриці та додавання непотрібних n_{1-k1} символів до кожного рядка і n_{2-k2} стовпця, що наступною читається рядками або стовпцями. Фізичним еквівалентом такого похідного коду є частотно-часовий код, де рядки відображаються на вісі часу, а стовпці - на вісі частоти. Параметри такого похідного коду залежать від параметрів складового коду, такі як кількість рядків (n), кількість символів у кожному рядку (k) та відстань кодування (d). Похідні коди будуються на основі вихідних кодів, до яких можуть бути додані символи для збільшення відстані, скорочення деяких інформаційних символів без зміни відстані або викидання деяких символів.

Код Хеммінга є прикладом процедури розширення коду, яка дозволяє збільшити відстань коду з 3 до 4 символів. Необхідність у використанні похідних кодів виникає тоді, коли потрібно побудувати інший код з такою самою відстанню, але він має бути менш ефективним, коротшим або менш стійким, ніж вихідний код. Під терміном "похідний код" в ширшому розумінні розуміють всі коди, які були отримані з початкового коду шляхом додавання або видалення символів або слів. Формально поділ кодів на двійкові та небінарні є штучним, і за аналогією до цього можна розрізняти трійкові, четвертинні та інші коди з більшою основою. Такий поділ обумовлений складністю алгоритмів побудови, кодування та декодування небінарних кодів. При певних умовах бажано, щоб інформаційні символи та зайві символи розташовувалися окремо. У систематичних кодах ця умова виконується, оскільки вони мають властивість розділення цих символів. У циклічних кодах кожне слово містить усі його циклічні

перестановки. Усі циклічні перестановки довжиною n утворюють цикл. У квазіциклічних кодах цикл складається з n-1 або навіть рідше n-2 символів.

Циклічні коди мають велике значення як з математичної точки зору, так і з точки зору їх побудови та реалізації. У каналах зв'язку помилки мають різний розподіл. Для вибору надійного коду для боротьби з цими помилками, доцільно розглядати різні конфігурації помилок, які можуть бути незалежними (некорельованими) або групуватися у пакети (корельовані помилки). При практичній реалізації кодів необхідно також враховувати якість інтервалів між пакетами помилок, оскільки вони можуть бути абсолютно бездоганними або містити випадкові та незалежні помилки.

Кореляційні коди використовуються для імпульсної модуляції і включають пари протилежних сигналів з високою автокореляційною функцією. Іншим типом кодів є коди інтервалів імпульсів, які складаються з постійної кількості імпульсів без перекривання. Це означає, що для будь-якого зсуву в часі кількість імпульсів залишається постійною для всіх кодових слів. В різних системах зв'язку ці поняття можуть мати різний контекст. Наприклад, у дротових лініях і лінійних трактах з фільтрами, що обмежують смуги з крутими фронтами, потрібно перерозподілити енергію основного сигналу від крайніх до центральних частот смуги пропускання для зменшення спотворень міжсимвольного інтерференції. У радіозв'язкових мережах, де існують жорсткі обмеження на електромагнітну сумісність, кореляційні коди використовуються для значного (десятки децибел) зниження рівня позасмугового випромінювання.

Побудова компактних кодів для кодування та декодування частот в значній мірі залежить від використовуваного методу модуляції. Арифметичні коди використовуються для забезпечення точності обчислень під час арифметичних операцій на комп'ютерних процесорах. Також існують блокові і деревоподібні коди. Головна відмінність між цими двома типами кодів полягає у наявності або відсутності пам'яті. Блоковий кодер є безпам'ятним пристроєм, який перетворює вхідну послідовність з k символів на вихідну послідовність з n символів.

"Безпам'ятний" означає, що кожен блок з n символів залежить лише від відповідного блоку з k символів і не залежить від інших блоків. Проте це не означає, що кодер не має жодних елементів пам'яті. Важливими параметрами блокового коду є значення *n, k, R = k / n і* d_{min}. Зазвичай значення *k* знаходяться в діапазоні від 3 до кількох сотень, а співвідношення *R* знаходиться в діапазоні від 1/4 до 7/8. Значення, які перевищують ці межі, можуть бути можливими, але часто стикаються з практичними труднощами. Зазвичай вхідні та вихідні послідовності складаються з двійкових символів, але іноді вони можуть містити елементи з більш широкого алфавіту [10].

Для деревовидного коду кодер є пристроєм з пам'яттю, який отримує m наборів двійкових символів на вході і відображає їх як набори з n двійкових символів на виході. Кожен набір з n вихідних символів залежить від поточного вхідного набору та v попередніх вхідних символів. Тому пам'ять кодера повинна містити v + m вхідних символів. Параметр v + m часто називають обмеженням довжини коду цього кодера і позначають як k = v + m (не плутати з параметром k для блочного коду). Довжина обмеження коду буде відома як значення v. Типові значення параметрів для деревовидного коду включають: *m, n = 1 ... 8, R = 1/4 ... 7/8, v = 2 ... 60.*

У випадку звичайних двійкових кодів, операція додавання кодових слів полягає в додаванні символів по модулю 2. Це означає, що при додаванні 1 + 1 отримуємо 0, 1 + 0 дорівнює 1, а 0 + 0 дорівнює 0. Ця властивість має два важливих наслідки. По-перше, лінійність спрощує процес кодування та декодування, оскільки кожне кодове слово може бути представлено як лінійна комбінація невеликої кількості базових векторів або вибраних кодових слів.

Друга властивість полягає в тому, що лінійність значно спрощує обчислення параметрів коду. Відстань між двома кодовими словами можна виміряти як відстань між кодовим словом, що складається повністю з нулів, і цим самим кодовим словом. Це означає, що вимірювання відстані між двома будь-якими

кодовими словами є еквівалентним вимірюванню відстані до кодового слова, що складається повністю з нулів.

Таким чином, при розрахунку параметрів лінійного коду, достатньо враховувати поведінку, коли передається кодове слово, що складається повністю з нулів. Розрахунок параметрів спрощується, оскільки відстань Хеммінга між заданим кодовим словом і нульовим кодовим словом дорівнює кількості ненульових елементів у кодовому слові. Це число часто називають вагою Хеммінга слова. Список, який 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 тому, що лінійні коди утворюють замкнений набір кодових слів, який можна комбінувати за допомогою лінійних операцій, таких як додавання або множення. Лінійність спрощує процес побудови та реалізації таких кодів. На великих довжинах часто використовуються лінійні коди. Однак, нелінійні коди зазвичай мають кращі параметри, ніж лінійні коди. Для відносно коротких кодів складність побудови та реалізації лінійних і нелінійних кодів приблизно однакова. Як лінійні, так і нелінійні коди належать до широкого класу завадостійких кодів, який включає багато різних типів кодів з різними властивостями.

Серед лінійних блоків найбільше значення мають такі коди, як коди з первинною парністю, М-коди (симплекс), ортогональний код, біортогональний код, код Хеммінга, Бозе-Чоудхурі-Хокінгема (BCH), Голі код, Квадратний код (КБ), Ріда-Соломона код. Нелінійні коди включають такі типи як контрольна сума, обернена кодова послідовність, Нордстрома-Робінсона код (НР), постійна вага, переставне знакове і беззнакове повторення (повні коди для ортогональних таблиць, проективних груп, груп Матьє та інших груп перестановок). Більшість схем кодування, які використовуються на практиці, базуються на лінійних кодах.

Білінійні блокові коди часто отримують назву "групові" через те, що їх кодові слова утворюють математичні структури, які відомі як групи. Лінійні коди дерева

часто називаються "згортковими" кодами, оскільки операцію кодування можна розглядати як дискретну згортку вхідної послідовності з характеристикою кодера, яка подібна до імпульсної відповіді.

Основними характеристиками завадостійких кодів є їх довжина, основа, загальне число кодових комбінацій, число допустимих комбінацій, надлишковість коду і мінімальне кодове розсташування.

Довжина коду визначається кількістю символів у кодовій комбінації. Наприклад, якщо кодова комбінація складається з п'яти символів, то довжина коду буде рівна 5. Рівномірні коди мають однакову довжину для всіх кодових комбінацій, тоді як у нерівномірних кодах довжина може варіюватися.

Основа коду визначається кількістю різних символів, які можуть бути використані в коді. Для двійкових кодів основа буде рівна 2, оскільки доступні символи 1 і 0.

Загальне число кодових комбінацій в рівномірному коді визначається як добуток довжини коду (n) на основу коду (m). Наприклад, для рівномірного коду з довжиною n=6 і основою m=2 загальне число кодових комбінацій буде рівним N=26=64. Число допустимих кодових комбінацій (Np) вказує на кількість комбінацій, які можуть бути використані без виникнення помилок.

Надлишковість коду (Кі) визначається як різниця між загальним числом кодових комбінацій (N) і числом допустимих комбінацій (N_p).

Мінімальне кодове розсташування (d_{min}) вказує на мінімальну відстань між будь-якими двома кодовими комбінаціями в коді.

Число дозволених кодових комбінацій (N_p) представляє собою кількість комбінацій, які використовуються для передачі повідомлень в завадостійких кодах. Важливо зауважити, що N_p завжди менше за загальне число кодових комбінацій (N). Ті кодові комбінації, які залишилися після віднятття Np від N, називаються забороненими комбінаціями. Якщо N_p дорівнює N, то такий код є безвихідним, оскільки всі комбінації використовуються і жодна не залишається

незадіяною. У розділених кодах число дозволених комбінацій N_p рівне 2k, де k є цілим числом.

Надлишковість коду Кі визначається в загальному випадку за допомогою певного виразу:

$$K\_n = 1 - \log\_2 \quad ⟦N\_p⟧/\log\_2 \ N \ . \qquad (1.1)$$

Надлишковість коду Кі визначає, яка частина довжини кодової комбінації не використовується для передачі інформації, а використовується для забезпечення додаткової надійності та завадостійкості коду. У випадку розділених кодів:

$$K\_n = 1 - k/n = r/n \ , \qquad (1.2)$$

де величина k/n називається відносною швидкістю коду.

Кодова відстань d(A,B) визначається як кількість позицій, в яких дві кодові комбінації А і В відрізняються одна від одної. Наприклад, якщо А = 01101 і В = 10111, то кодова відстань d(A,B) дорівнює 3. Кодова відстань між комбінаціями А і В може бути обчислена шляхом додавання за модулем 2 відповідних символів на відповідних позиціях:

$$d(A,B) = \sum\_(i = 1)\^n ▒ ⟦a\_i \oplus b\_i⟧, \qquad (1.3)$$

де a_i і b_i є і-ми розрядами кодових комбінацій А і В, символ "+" позначає додавання за модулем 2. Наприклад, для обчислення кодової відстані між комбінаціями 1101011 і 0111101, ми просто підраховуємо кількість одиниць, які

вони мають на відповідних позиціях, і виконуємо додавання за модулем 2 для цих значень:

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

Кодова відстань між різними комбінаціями в конкретному коді може бути різною. Наприклад, для первинних кодів, це значення може коливатись від одиниць до довжини самого коду. Мінімальна кодова відстань, позначена як d_{min}, є найменшою відстанню між двома дозволеними кодовими комбінаціями в даному коді. Мінімальне кодове розташування є важливою характеристикою, яка вказує на коригуючу здатність коду.

У первинних кодах, де всі комбінації є дозволеними, мінімальна кодова відстань становить одиницю ($d_{min} = 1$). Однак такі коди не можуть виявляти або виправляти помилки передачі даних. Для того, щоб код мав коригуючі здібності, мінімальна кодова відстань повинна бути не менше двох ($d_{min} \geq 2$). Це означає, що код здатний виявити та виправити помилки, коли відстань між двома кодовими комбінаціями становить щонайменше два розряди.

Для того, щоб виявити всі помилки, які мають кратність s або менше, мінімальна кодова відстань повинна відповідати такій умові:

$$d\_min \geq s + 1, \qquad (1.5)$$

Якщо код використовується для виправлення помилок, кратність яких не перевищує t, тоді мінімальна кодова відстань повинна мати значення, яке задовольняє умову:

$$d\_min \geq 2t + 1, \qquad (1.6)$$

Наприклад, з цих рівнянь випливає, що для виявлення однократних помилок (s = 1) потрібен код з мінімальною кодовою відстанню $d_{min} = 2$, а для

виправлення таких помилок потрібен код з мінімальною кодовою відстанню dmin = 3.

Для виявлення помилок і виправлення помилок повинна виконуватись умова:

$$d\_min \geq s + t + 1, \tag{1.7}$$

Отже, завдання побудови коду з визначеною корекційною здатністю полягає у досягненні потрібного кодового розташування. Збільшення мінімальної кодової відстані призводить до зростання надлишковості коду. Проте, бажано, щоб кількість перевірочних символів була мінімальною. В даний час існують верхні та нижні межі, які встановлюють зв'язок між кодовими відстанями та кількістю перевірочних символів. Інформація про межі кодової відстані надаватиметься під час оцінки якості прийнятих кодованих повідомлень.

1.4 Код Ріда-Соломона

Код РС є типом коду, який може виправляти одну помилку в одному блоку даних. При застосуванні цього коду до кожного блоку інформації додаються два додаткових елементи, позначені як X і Y. Значення цих елементів визначаються згідно таких умов [7]:

- для трьох байтів інформації (byte1, byte2, byte3):
  byte1 + byte2 + byte3 + X + Y = 0
  byte1 + 2 * byte2 + 3 * byte3 + 4 * X + 5 * Y = 0
- щоб обчислити конкретні значення X і Y для кодування трьох байтів:
  Y = 3 * byte1 + 2 * byte2 + byte3
  X = -4 * byte1 - 3 * byte2 - 2 * byte3

Тепер для виявлення та виправлення помилок ми будемо використовувати наступні обчислення: Значення помилки (Error_Value) розраховується за формулою:

$$Error\_Value = byte1 + byte2 + byte3 + X + Y$$

Оскільки раніше, до виникнення помилки, сума цих значень дорівнювала нулю, тепер ця сума відображає безпосереднє значення помилки, яке можна легко відняти від пошкодженого байта. Якщо блок був прийнятий без помилок, то значення помилки (Error_Value) дорівнює нулю. Тепер ми можемо знайти номер байта, який потребує виправлення, шляхом обчислення:

$$N = byte1 + 2 * byte2 + 3 * byte3 + 4 * X + 5 * Y$$
$$Error\_Byte\_Number = N / Error\_Value$$

Якщо ми маємо потребу захистити блок даних, що складається з більш ніж трьох байтів, то формули для розрахунку корекційних значень трохи змінюються. Наприклад, для блоку довжиною 16 байт:

$$Y = 16 * byte1 + 15 * byte2 + 14 * byte3 + ... + byte16$$
$$X = -17 * byte1 - 16 * byte2 - 15 * byte3 - ... - 2 * byte16$$
$$Error\_Value = byte1 + byte2 + byte3 + ... + X + Y$$
$$N = byte1 + 2 * byte2 + 3 * byte3 + ... + 16 * byte16 + 17 * X + 18 * Y$$

Проте, важко застосовувати цей код для блоків даних коротше 4 байт, оскільки довжина контрольних параметрів X і Y повинна бути щонайменше 4 байти [8].

Для блоку даних розміром 4 байти, до нього буде доданий коригуючий блок, який складається з 2 байтів (DW) для X та 2 байтів на Y. Таким чином, загальна довжина блоку становитиме 8 байтів. Але що робити, якщо виникло дві або більше

помилки в цьому блоку? Один зі способів виявлення двох помилок - це отримання десяткового числа в якості Error_Byte_Number. Наприклад, якщо у блоку з нулями зустрічаються 2 одиниці (дві помилки) у третьому і четвертому байтах, то Error_Byte_Number буде дорівнювати 3.5. Але якщо у блоку зустрічаються 4 одиниці, відповідно у 3, 4 і 5 байтах, то Error_Byte_Number буде дорівнювати 4.

Канал зв'язку або передачі даних, захищений кодом РС, зображений на рисунку 1.5.

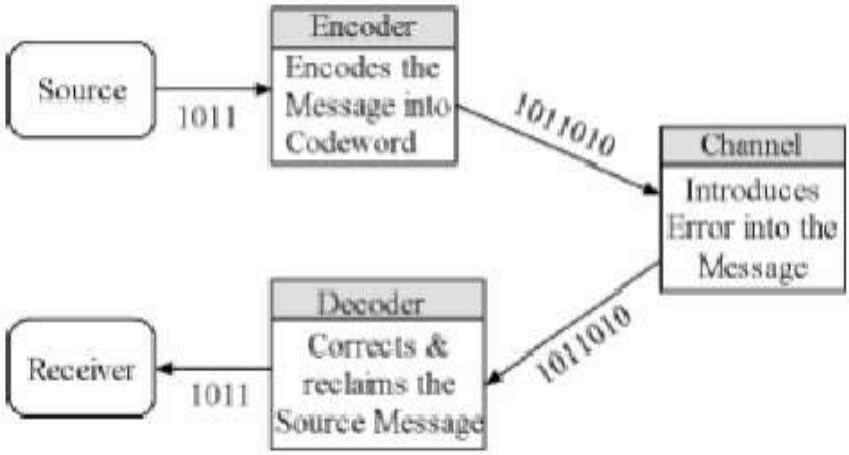


Рисунок 1.5 – Канал зв'язку, захищений кодом РС

1.5 Постановка задачі

Метою цього проєкту є розробка пристрою для обміну даними, який буде стійким до завад та забезпечувати надійну передачу даних з максимальною швидкістю при мінімальних апаратурних витратах. Для досягнення цієї мети необхідно вирішити наступні завдання:

- розробити алгоритм роботи пристрою для завадостійкого обміну даними на основі коду Ріда-Соломона. Цей алгоритм повинен забезпечувати надійну передачу інформації та враховувати можливі завади під час обміну даними;

- розробити функціональну схему кодуючого та декодуючого пристрою для обраного коду. Ці пристрої повинні здійснювати процес кодування та декодування даних з використанням завадостійкого коду Ріда-Соломона;

- провести моделювання роботи розроблених пристроїв. Це дозволить перевірити ефективність та надійність пристроїв в умовах симульованого обміну даними і виявити можливі проблеми чи несправності [38].

У результаті виконання цих завдань буде розроблений пристрій для завадостійкого обміну даними на основі коду Ріда-Соломона, який забезпечуватиме надійну передачу інформації з максимальною швидкістю при мінімальних витратах апаратури.

1.6 Висновки

Під час аналізу інструментів та рішень для завадостійкого кодування було виявлено різні методи, які можуть бути використані. Зокрема, коди Бодо та Морзе є прикладами схем кодування, які забезпечують стійкість передачі повідомлень. Крім того, схеми кодування, засновані на принципі Клода-Шеннона, можуть забезпечити високий захист від помилок передачі.

Також було виявлено, що схеми кодування можна класифікувати за різними критеріями, такими як помилкове виявлення всієї кодової комбінації і використовуване шифрування. Використання завадостійких кодів може максимізувати пропускну здатність каналу зв'язку при мінімізації ймовірності помилок передачі. Тому вибір схеми кодування залежить від надійності передачі повідомлення і вимог до каналу зв'язку, що використовується.

2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ПРИСТРОЮ ЗАВАДОСТІЙКОГО КОДУВАННЯ НА БАЗІ FPGA

2.1 Методи представлення коду Ріда-Соломона

Коди Ріда-Соломона (РС) є різновидом лінійних блокових кодів і підкласом кодів Боуза-Чодрі-Хокінгхема (БЧХ). Вони широко використовуються в завадостійкому кодуванні, особливо для передачі даних по зашумлених каналах, де можуть виникати помилки передачі.

В основі РС кодів лежить скінченне тіло, відоме як тіло Галуа. Однією з ключових особливостей РС-кодів є їх здатність виявляти та виправляти помилки: РС-код може виправити до t помилок, де *t* - це половина кількості символів перевірки на парність *(n-k)* у кодовому слові. Це означає, що якщо в переданому кодовому слові є до *t* помилок, дані можуть бути успішно відновлені [32].

Ефективність виправлення помилок для РС кодів залежить від їх властивостей. Параметри коду *RS(n,k)*: *m* - кількість бітів на символ, *k* - довжина незашифрованого повідомлення (в символах), *n* - довжина кодового слова (в символах), *(n-k)* - кількість символів перевірки на парність, *t* - кількість помилок, що виправляються, *2t = n-k.*

На рисунку 2.1 зображено типове систематичне кодове слово РС. Систематичні коди означають, що дані в лівій частині залишаються незмінними, а символи перевірки на парність додаються разом з даними для формування кодового слова.

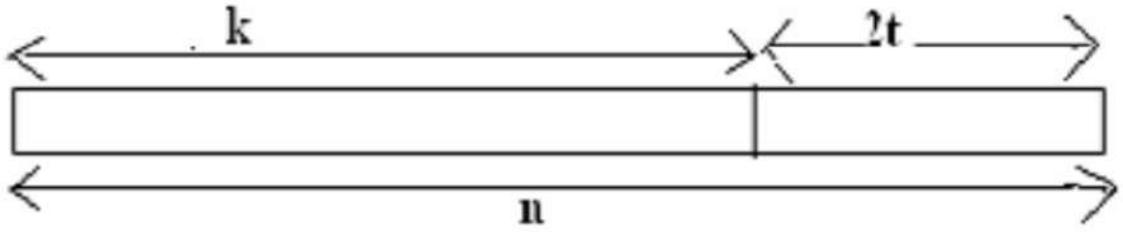


Рисунок 2.1 - Кодове слово кода Ріда соломона *RS(n, k)*

Коди РС є потужним інструментом для надійного і завадостійкого кодування і можуть забезпечити надійну передачу і зберігання даних при успішному використанні в FPGA.

Традиційно коди, що виправляють t_{min} помилки, описується над полем Галуа GF(q) породжувальним поліномом 2.1

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

В автоаналітичному представленні кодів РС на основі лінійних послідовних схем, ЛПС над полем GF(q) визначається як функція станів (переходів) 2.2.

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

і функцією виходів 2.3

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

де t - дискретний час, $A = [a_{ij}]_{r * r}$, $B = [b_{ij}]_{r * l}$, $C = [c_{ij}]_{m * r}$, $D = [d_{ij}]_{m * l}$ – характеристична матриця, $S(t) = [s_i]_r$ – слово стану, $U(t) = [u_i]_l$ вхідне слово і $Y(t) = [y_i]_m$ – 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 GFA 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 [25].

На основі характеристичної матриці ЛПС виконаємо детальний аналіз моделі автоматичного аналізу коду РС.

Породжуючий поліном виражається наступним чином:

$$g(x)= a^{i} + a^{i}x + a^{i}x^{2} +...+ a^{i}x^{2t_{min}}, GF(q) \qquad (2.4)$$

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:

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

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

$$\mathrm{C} = |0... \ 0\alpha|, \ D = |0| \qquad (2.5)$$

Рекурсивний ЛСП Фібоначчі - це ЛСП, в якому сигнал на вхід надходить з виходу, а характеристична матриця має наступний вигляд:

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

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

$$C = |0 \dots 0\alpha| \; D = |0| \qquad (2.6)$$

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) [24].

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.

$$S(k) = \begin{vmatrix} s_0^k \\ s_1^k \\ \dots \\ s_{r-2}^k \\ s_{r-1}^k \end{vmatrix} \qquad (2.7)$$

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

$$L = A^{r-1} * A^{r-2} * B,...,A * B,B \quad (2.8)$$

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

$$L_r^G = \begin{vmatrix} 0 & 0 & \cdots & 0 & \alpha^0 \\ 0 & 0 & \cdots & \alpha^0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \alpha^0 & \cdots & 0 & 0 \\ \alpha^0 & 0 & \cdots & 0 & 0 \end{vmatrix} \quad (2.9)$$

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

$$L_r^F = \begin{vmatrix} \alpha^0 & 0 & \cdots & 0 & 0 \\ l_{2,1} & \alpha^0 & \cdots & 0 & 0 \\ l_{3,1} & l_{3,2} & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ l_{r,1} & l_{r,1} & \cdots & l_{r,r-1} & \alpha^0 \end{vmatrix}, l_{i,j} = \{\alpha^0, \alpha^1, \cdots, \alpha^{n-1}\}, i \neq j. \quad (2.10)$$

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 О слів. В першому випадку компоненти Ψi слова $\Psi=\{\psi_0, \psi_1, \ldots, \psi_{r-2}, \psi_{r-1}\}$ можуть бути знайдені в результаті обчислення згортки:

$$\psi i = \sum ai,j * sj,GF(q) \quad (2.11)$$

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

$$A(s_j \in S(k),\ a_{\ i,j} \in A^{r},\ i = 0r - 1,\ j = 0r - 1), \tag{2.12}$$

а у другому випадку дорівнюють:

$$\psi = s^{m}_{r-1-j} \tag{2.13}$$

Де sm - (r − 1 − j)-та компонента слова стану S(m) ; ($sm\ r{-}1{-}j \in$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

$$S(k) = Ak * S(0) + Lk * I, GF(q) \tag{2.14}$$

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

$$S(n) = Ar * S(k) + Lr * \Psi\, GF(q) \tag{2.15}$$

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

$$Lr * \Psi = Ar * S(k) \qquad \text{Формула13}$$

Підставляючи значення матриці Lr 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:

$$r\ S(n)=A*k*S(0)+Lk*I+L\ r*\Psi=$$
$$=Ak+r*S(0)+Ar*Lk*I+Lr*\Psi=$$
$$=S(0)+Ar*Lk*I+Lr*\Psi,\ GF(q) \qquad (2.16)$$

Добуток Lk * I визначає стан, в який перейде ЛПС після подачі на її вхід слова I , а добуток $Ar*Lk*I$ 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:

$$S(n)=S(0)+S(m)+Lr*\Psi,GF(q) \qquad (2.17)$$

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 Lr * Ψ = S(m) . Підставивши значення матриці Lr із (Формули 9), отримаємо:

$$\Psi=-S(m)\ . \qquad (2.18)$$

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 Ψі 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 [15].

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

$$I = [a^1 a^5 a^3 a^4 a^0 a^2 a^5 a^6 a^0 a^2 a^{12}] . \quad (2.19)$$

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

$$g(x) = (x - a)(x - a^2)(x - a^3)(x - a^4) = a^{10} + a^3 + a^6 x^2 + a^{13} * x^3 + x^4 \quad (2.20)$$

та характеристичні матриці рекурсивної ЛПС типу Галуа:

$$A = \begin{vmatrix} 0 & 0 & 0 & \alpha^{10} \\ \alpha^0 & 0 & 0 & \alpha^3 \\ 0 & \alpha^0 & 0 & \alpha^6 \\ 0 & 0 & \alpha^0 & \alpha^{13} \end{vmatrix}; \; B = \begin{vmatrix} \alpha^0 \\ 0 \\ 0 \\ 0 \end{vmatrix}; \quad (2.21)$$

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)

$$S(11) = [a^9 a^3 a^4 a^{13}]. \quad (2.22)$$

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):

$$A^4 = \begin{vmatrix} \alpha^{10} & \alpha^8 & \alpha^{11} & \alpha^{11} \\ \alpha^3 & \alpha^8 & \alpha^5 & \alpha^{13} \\ \alpha^6 & \alpha^7 & \alpha^{11} & \alpha^{13} \\ \alpha^{13} & \alpha^1 & \alpha^1 & \alpha^{10} \end{vmatrix} \quad (2.23)$$

Тепер можна обчислити компоненти слова Ψ = [Ψ0 Ψ1 Ψ2 Ψ3]:

$$\Psi 0 = a10a9 + a8a3 + a11a4 + a11a11 = a10, GF(8)$$
$$\Psi 1 = a3a9 + a8a3 + a5a4 + a13a11 = a0, GF(8)$$
$$\Psi 2 = a6a9 + a7a3 + a11a4 + a13a11 = a13, GF(8)$$
$$\Psi 3 = a13a9 + a1a3 + a1a4 + a10a11 = a1, GF(8) \qquad (2.24)$$

2.2 Кодер

Кодери Ріда-Соломона (RS) виконують процес кодування, приймаючи блок символів і додаючи символ перевірки на парність. Кодове слово RS може бути обчислено з символів вхідного повідомлення за допомогою породжуючого полінома. Твірний поліном визначається порядком поля Галуа, в якому використовується кодове слово RS, та кількістю помилок, які потрібно виправити.

Твірний поліном RS використовується для обчислення символів перевірки на парність, що додаються до символів вхідного повідомлення. Ці перевірки на парність дозволяють виявити та виправити помилки під час декодування. Кількість символів перевірки на парність залежить від кількості помилок, які може виправити код РС.

Основне призначення кодера Ріда-Соломона - забезпечити надійне кодування даних, а також виявлення та виправлення помилок. Ці кодери широко використовуються в різних системах передачі та зберігання даних, де важлива надійність і цілісність даних.

Примітивний багаточлено визначається в залежності від порядку поля Галуа. В одному з прикладів, примітивний багаточлен у полі GF(2^8) може бути представлений як $x^8 + x^4 + x^3 + x^2 + 1$. Нехай a є примітивним елементом у полі GF(2^8). Тоді генераторний багаточлен для RS-коду з виправленням ^{t} помилок матиме такий вигляд.

$$\prod_{i=1}^{2t}(x + a^i) \qquad (2.25)$$

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

$$g(x)=(x+a)(x+a^2)(x+a^3)(x+a^4) \quad (2.26)$$

Нехай вхідний многочлен M(x) Кодера

$$M(x) = mk-1 + mk-2\ xk-2 +... + c1x + c0\ . \quad (2.27)$$

Відповідний многочлен кодового слова задається формулою

$$C(x) = cn-1 + cn-2xn-2 +... + c1x + c0. \quad (2.28)$$

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

$$C(x) = x^{2t}M\ (x) + x^{2t}M(x)\ mod g\ (x). \quad (2.29)$$

Якщо Q(x) і P(x) - відповідні поліноми приватного залишку, коли x2tM(x) ділиться на g(x), томного членового коду також може бути виражений іншим способом.

$$C(x) = x2tM(x) + P(x) = Q(x)g(x). \quad (2.30)$$

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

$$\frac{x^{n-k}M(x)}{g(x)} = Q(x) - \frac{P(x)}{g(x)} \quad (2.31)$$

Додаючи поліном P(x) перевірки на парність до поліному x2 M(x), обчислюється поліном C(x) кодового слова, який націло ділиться на g(x).

Виходячи з розрахунків, можемо побудувати схему кодера Ріда-Соломона зображену на рисунку 2.2.

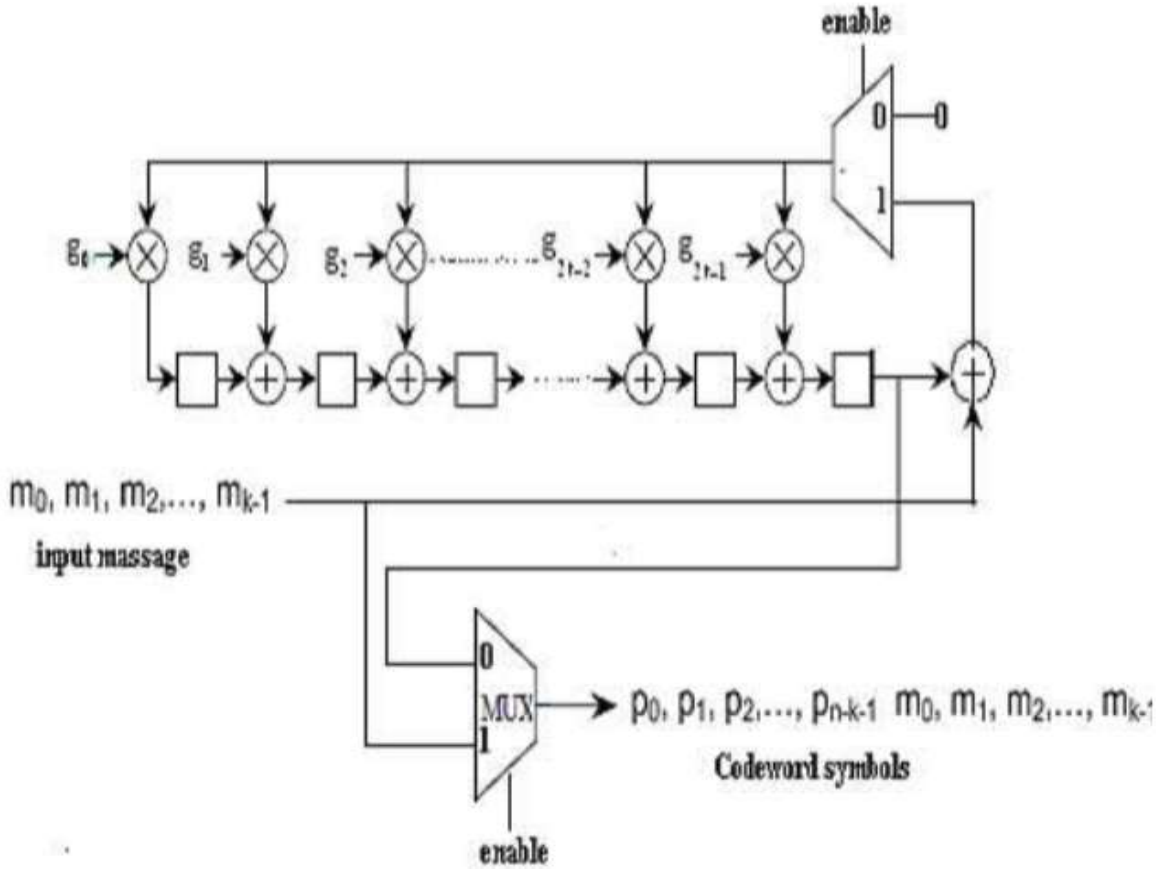


Рисунок 2.2 - Схема кодера Ріда-Соломона

2.3 Декодер

Декодери Ріда-Соломона використовуються для відновлення вихідного повідомлення з кодових слів RS, які можуть містити помилки. Декодер обчислює виправлення помилки за допомогою примітивного полінома p(z). Розкриваючи дужки і замінюючи α на z, корінь примітивного полінома, код Ріда-Соломона, який виправляє подвійні помилки.

Описаний вище породжуючий поліном є загальним для будь-якого коду Ріда-Соломона і будь-якого поля Галуа, яке може виправляти подвійні помилки.

Щоб обчислити конкретний породжуючий поліном для конкретного поля Галуа, необхідно визначити коефіцієнти псевдозмінної X полінома p(z).

Ці коефіцієнти можна обчислити за допомогою алгоритму ділення. Залишок від ділення і є шуканим набором коефіцієнтів. Одним із способів вибору примітивного полінома p(z) для конкретного тіла Галуа є використання таблиці незвідних поліномів.

Оскільки тіло Галуа GF(28) є найпоширенішим і найефективнішим у багатьох застосуваннях, можна вибрати перший поліном 8 в таблиці (зазвичай це поліном з найменшою кількістю ненульових коефіцієнтів).

$$p(z)=435_8 = 100011101_2 = z^8 + z^4 + z^3 + z^2 + 1 \quad (2.32)$$

Застосовуючи операцію піднесення до степеня p(z), степінь всіх коефіцієнтів у твірному поліномі обмежується 7. Це означає, що найвищий можливий степінь полінома дорівнює 7, тобто жоден коефіцієнт не має степеня, більшого за 7 [29].

Ця властивість важлива при обчисленні та використанні твірних поліномів, оскільки вона гарантує, що кількість і степінь коефіцієнтів залишаються обмеженими. Це спрощує обробку та зберігання поліномів і полегшує їх використання в алгоритмах декодування та виправлення помилок.

$$(4 + z^3 + z^2 + z) mod p(z) = z^4 + z^3 + z^2 + z$$
$$(z^7 + z^6 + z^4 + z^3) mod p(z) = z^7 + z^6 + z^4 + z^3$$
$$(z^9 + z^8 + z^7 + z^6) mod p(z) = z^7 + z^6 + z^5 + z^2 + z + 1$$
$$(z^{10}) mod p(z) = z^6 + z^5 + z^4 + z^2 \quad (2.33)$$

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

$$RS(x) = x^4 + x^3(z^4 + z^3 + z^2 + z)$$

$$+x^2(z^7 + z^6+z^4+z^3)$$
$$+x(z^7 + z^6+z^5+z^2 + z + 1)$$
$$+(z^6 + z^5+z^4+z^2) \quad (2.34)$$

На основі цих розрахунків можемо побудувати схему декодеру зображену на рисунку 2.3.

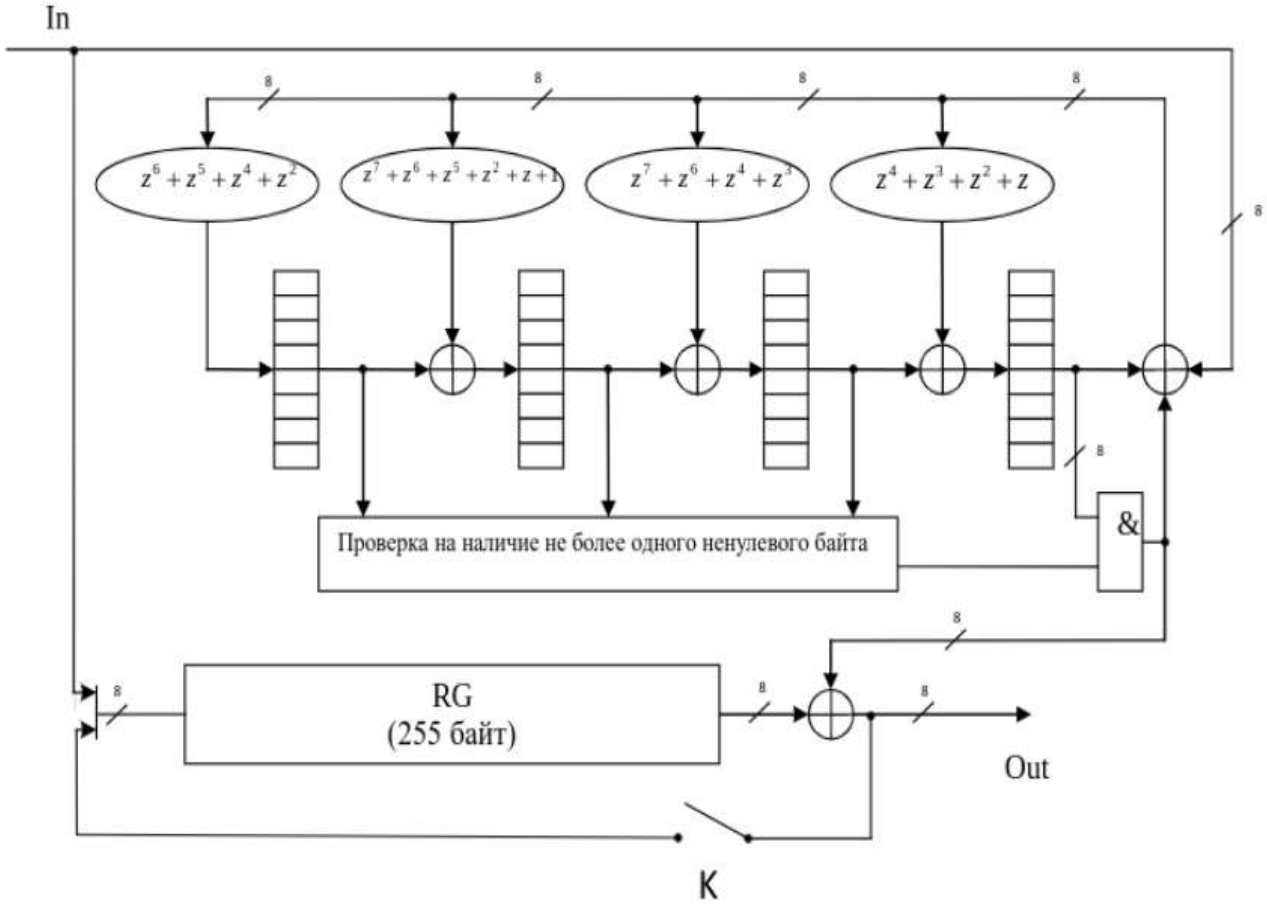


Рисунок 2.3 - Схема декодеру RS(255, 251)

Описаний вище декодер може виправити два потенційно помилкових байти за 3n циклів і демонструє наступну послідовність операцій:

- протягом перших n тактів генерується синдром і кодове слово зберігається у буферному регістрі;

- протягом наступних n циклів виправляється один з двох потенційно помилкових байтів. Синдром виправляється і кодове слово знову записується в буферний регістр;

- протягом останніх n тактів виправляється другий потенційно помилковий байт.

При конфігуруванні дешифратора на основі двійкових логічних елементів, множник повинен бути виражений в константних термінах як суматор по модулю два. Це означає, що використовуються операції додавання (сума) та додавання за модулем 2.

Однак недоліком таких декодерів є те, що робота кодера і декодера не є послідовною. Кодеру доводиться чекати 2n тактів, перш ніж декодер виправить помилку. Цю проблему можна вирішити за допомогою конвеєрної реалізації, яка дозволяє кодуванню та декодуванню працювати паралельно, зменшуючи загальний час роботи системи. Крім того, для побудови дешифратора на двійкових логічних елементах потрібен елемент поля у загальному форматі. Таким елементом поля є біт n [30].

Для цього потрібен елемент поля в загальній формі:

$$a_7 - z^7 + a_6 z^6 + a_5 z^5 + a_4 z^4 + a_3 z^3 + a_2 z^2 + a_1 z + a_0 \qquad (2.35)$$

Для отримання відповідних коефіцієнтів виконуються операції множення на відповідні коефіцієнти та ділення на p(z). Залишкові коефіцієнти, отримані в результаті такого ділення, можна інтерпретувати як суматор у вигляді множника.

$$R_7 = (a_7 z + a_5 z\, a_4 z + a_3 z + a_2 z + a_1 z)\, z^7\, mod p(z) =$$
$$(a_6 + a_5 + a_4 + a_0)z^7 + (a_5 + a_4 + a_3)z^6 +$$
$$+ (a_7 + a_4 + a_3 + a_2)z^5 + (a_7 + a_3 + a_2 + a_1)z^4 +$$
$$+ (a_5 + a_4 + a_2 + a_1)z^3 + (a_6 + a_5 + a_3 + a_1)z^2 +$$
$$+ (a_7 + a_6 + a_2)z^1 + (a_7 + a_6 + a_5)$$
$$R_6 = (a_7 + a_6 + a_5 + a_2)z^7 + (a_6 + a_5 + a_4)z^6 +$$

$$+ (a_5 + a_4 + a_3)z^5 + (a_7 + a_4 + a_3 + a_2)z^4 +$$
$$+ (a_6 + a_5 + a_3 + a_2)z^3 + (a_7 + a_6 + a_4 + a_2)z^2 +$$
$$+ (a_7 + a_3)z^1 + (a_7 + a_6 + a_2)$$
$$R_5 = (a_7 + a_6 + a_2)z^7 + (a_7 + a_6 + a_5 + a_1)z^6 +$$
$$+ (a_6 + a_5 + a_4 + a_0)z^5 + (a_5 + a_4 + a_3)z^4 +$$
$$+ (a_6 + a_4 + a_3)z^3 + (a_7 + a_5 + a_3)z^2 + a_5 z + a_5$$
$$R_4 = (a_7 + a_3)z^7 + (a_7 + a_6 + a_2)z^6 +$$
$$+ (a_7 + a_6 + a_5 + a_1)z^5 + (a_6 + a_5 + a_4 + a_0)z^4 +$$
$$+ (a_7 + a_5 + a_4)z^3 + (a_7 + a_6 + a_4)z^2 + a_5 z + a_4$$
$$R_3 = a_4 z^7 + (a_7 + a_3)z^6 + (a_7 + a_6 + a_2)z^5 + (a_7 + a_6 + a_5 + a_1)z^4 +$$
$$+ (a_6 + a_5 + a_0)z^3 + (a_7 + a_5)z^2 + (a_7 + a_6)z + a_5$$
$$R_2 = a_5 z^7 + a_4 z^6 + (a_7 + a_3)z^5 + (a_7 + a_6 + a_2)z^4 +$$
$$+ (a_7 + a_6 + a_1)z^3 + (a_6 + a_0)z^2 + a_7 z + a_6$$
$$R_1 = a_6 z^7 + a_5 z^6 + a_4 z^5 + (a_7 + a_3 z)z^4 + (a_7 + a_2 z)z^3 + (a_7 + a_1)z^2 + a_0 z + a_7$$
$$R_0 = a_7 z^7 + a_6 z^6 + a_5 z^5 + a_4 z^4 + a_3 z^3 + a_2 z^2 + a_1 z + a_0 \qquad (3.36)$$

Після знаходження залишку від ділення можна побудувати схему постійного множення. Інші елементи кодера і декодера для апаратної реалізації коду Ріда-Соломона на основі двійкових логічних елементів (255, 251) не потребують спеціальних методів розрахунку або побудови. Описані вище методи декодування і перетворення множника можуть бути застосовані не тільки до цього коду, але і до будь-якого коду Ріда-Соломона, що виправляє одинарні або подвійні помилки з чергуванням і/або скороченням символів. На закінчення варто відзначити, що для генерації синдрому коду Ріда-Соломона, реалізованого апаратно, необхідно замінити множник на константу з використанням примітивних елементів суматора за модулем 2.

2.4 Висновок

В рамках роботи були обговорені теми пов'язані з кодувальниками та декодувальниками Ріда-Соломона, моделюванням та проектуванням завадостійких

кодерів на основі FPGA, а також способи представлення кодів Ріда-Соломона. Також в контексті моделювання та проектування завадостійких кодерів на основі FPGA зазначено, що структура схеми постійного множення базується на знаходженні залишку від операції ділення і може бути реалізована за допомогою суматорів. Також зазначено, що елементи кодера та декодера для апаратної реалізації кодів Ріда-Соломона на елементах двійкової логіки не потребують спеціальних обчислювальних або конфігураційних методів. В контексті декодера Ріда-Соломона було розглянуто процес виправлення двох можливих помилкових байтів за 3n тактів, що включає генерацію синдрому, виправлення помилкових байтів та використання констант і множників у вигляді суматорів. Також зазначається, що розглянуті методи декодування та факторизації застосовуються для виправлення непарних або парних помилок, має символьне чергування та усічення.

3 АПАРАТНА РЕАЛІЗАЦІЯ КОДУ РІДА-СОЛОМОНА

3.1 Програмована користувачем вентильна матриця (FPGA)

Програмована користувачем вентильна матриця (FPGA) є спеціальним типом інтегральної схеми, яка може бути налаштована або програмована після її виготовлення, відтак і назва "що програмована полем". Зазвичай конфігурація FPGA визначається за допомогою мови опису апаратного забезпечення (HDL), яка схожа на ту, що використовується для проектування спеціалізованих інтегральних схем (ASIC) для конкретних застосувань. Раніше для визначення конфігурації FPGA використовувались принципові схеми, але з появою електронних засобів автоматизації проектування це стає все менш поширеним підходом.

FPGA містять масив програмованих логічних блоків і ієрархію реконфігурованих взаємоз'єднань, які дозволяють з'єднувати ці блоки між собою. Логічні блоки в FPGA можуть бути налаштовані для виконання складних комбінаційних функцій або виступати як прості логічні вентилі, такі як І (AND) та XOR (ексклюзивне "або").

У більшості FPGA логічні блоки мають елементи пам'яті, які можуть бути простими тригерами або складнішими блоками пам'яті. Багато FPGA можуть бути перепрограмовані, що дозволяє реалізувати різні логічні функції і гнучко переконфігурувати обчислення, подібно до того, як це виконується в комп'ютерному програмному забезпеченні.

FPGA відіграють важливу роль у розробці вбудованих систем, оскільки вони дозволяють почати розробку системного програмного забезпечення (SW) одночасно з апаратним забезпеченням (HW). Вони надають можливість моделювання продуктивності системи на ранніх етапах розробки та взаємодію з різними системними компонентами (SW і HW).

FPGA мають широке застосування у різних галузях. 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. Ці пристрої особливо важливі в середовищах, де важлива реального часу інформація для користувачів. Наприклад, домашня камера безпеки повинна миттєво передавати високоякісне зображення на розумні пристрої власника будинку з мінімальною затримкою. Очікування щодо швидкості та низької затримки тільки зростатимуть, оскільки споживачі стають все більше залежати від миттєвого доступу до необхідної інформації.

3.2 Мова проектування VHDL

Very High-Speed Integrated Circuit Hardware Description Language (VHSIC HDL, VHDL) - це мова програмування, яка використовується для опису апаратних засобів. Вона широко використовується в автоматизації електронного проектування для моделювання та розробки змішаних сигналів і цифрових систем, таких як інтегральні схеми і FPGA. VHDL був створений в початку 1980-х років як результат дослідницького проекту в галузі високошвидкісних інтегральних схем, який фінансувався Міністерством оборони США.

Під час програми VHSIC, дослідники стикалися з викликом описувати схеми великого масштабу і вирішувати складні проблеми проектування, які вимагали співпраці команди інженерів. Оскільки на той час доступні були лише інструменти проектування на рівні вентилів, виявилася потреба у розробці кращих та більш структурованих методів та інструментів проектування. Для вирішення цієї проблеми Міністерство оборони уклало контракт з командою інженерів з трьох компаній - IBM, Texas Instruments та Intermetrics, щоб завершити специфікацію та впровадити новий метод опису дизайну на основі мови. Перша загальнодоступна версія VHDL, версія 7.2, була випущена в 1985 році.

VHDL є мовою, яка надає багато функцій для детального опису поведінки електронних компонентів на різних рівнях деталізації. Вона може бути використана для опису від простих логічних вентилів до повноцінних мікропроцесорів і

користувацьких мікросхем. Особливості VHDL дозволяють точно визначити електричні аспекти поведінки ланцюга, такі як час наростання і спаду сигналів, затримки через вентилі та функціональна робота. Це дозволяє інженерам докладно описати різні електричні характеристики та забезпечити правильну роботу електронних компонентів у системі.

Отримані імітаційні моделі VHDL можуть бути використані як будівельні блоки для створення більших схем на системному рівні, використовуючи схеми, блок-схеми або описи VHDL. Це дозволяє проводити моделювання системи. Аналогічно до того, як мови програмування високого рівня дозволяють виражати складні концепції проектування у вигляді комп'ютерних програм, VHDL дозволяє описати поведінку складних електронних схем в системі проектування для автоматичного синтезу схем або моделювання системи. Подібно до мов програмування, таких як Pascal, C і C++, VHDL має функції, що сприяють структурованому проектуванню, а також широкий набір функцій управління та представлення дани

Відмінністю VHDL від інших мов програмування є наявність функцій, які дозволяють описувати одночасні події. Це має велике значення, оскільки обладнання, що описується за допомогою VHDL, має природу одночасності. Користувачі мов програмування програмованих логічних пристроїв (PLD), таких як PALASM, ABEL, CUPL та інші, знайдуть відомими функції одночасності, які надає VHDL. Однак, тим, хто має досвід програмування лише у програмних мовах, можуть бути необхідні пояснення деяких нових концепцій.

Одним з ключових аспектів VHDL, який часто недооцінюється, є його можливість використовувати специфікацію продуктивності схеми у вигляді тестового стенду. Це означає, що VHDL дозволяє описати тестові умови і параметри, які допомагають перевірити працездатність та продуктивність схеми. Цей підхід дозволяє розробникам ефективно валідувати та випробовувати свої схеми на ранніх етапах проектування.

Тестові стенди - це описи у VHDL, які містять стимули для ланцюга сигналів та очікувані результати, що перевіряють поведінку схеми з плином часу. Вони є

необхідною складовою частиною кожного проекту на VHDL і мають бути створені паралельно з описами самої схеми. Тестові стенди дозволяють розробникам валідувати та перевіряти правильність роботи схеми, забезпечуючи надійне тестування і відладку.

VHDL є потужною мовою, яка може бути використана для розробки нових проектів на високому рівні. Проте, вона також має значну користь як низькорівневий інструмент для зв'язку різних компонентів у комп'ютерному середовищі проектування. Завдяки своїм структурним особливостям, VHDL може бути ефективно використана як мова для опису мережі, замінюючи або доповнюючи інші мови, такі як EDIF, які використовуються для цього ж метою.

Переваги VHDL включають:

- підтримку різних методологій проектування, таких як верхньо-вниз і нижньо-вгору підходи, що дозволяє гнучко підходити до проектування.є фільтр грубої очистки;
- забезпечення гнучкої мови дизайну, що дозволяє виразно виражати задуми та вимоги проекту;
- можливість кращого керування дизайном, що дозволяє точно визначати поведінку та функціональність схеми;
- підтримку багаторівневої абстракції, що дозволяє описувати схеми на різних рівнях деталізації;
- забезпечення тісного зв'язку з нижчими рівнями конструкції, що дозволяє ефективно інтегрувати різні компоненти системи;
- підтримку всіх інструментів CAD, що сприяє ефективному проектуванню та аналізу схем;
- можливість повторного використання коду та спільного використання, що спрощує розробку та підтримку проектів.

Недоліки VHDL включають:

- вимогу до специфічних знань про структуру та синтаксис мови, що може бути викликом для новачків і вимагає додаткового навчання.;

- складність візуалізації та виявлення помилок у дизайні, оскільки VHDL зазвичай використовується для опису складних систем, що може ускладнити процес налагодження та виявлення проблем;

- Деякі програми VHDL не можуть бути синтезовані, тобто перетворені на фізичну апаратну реалізацію, що може обмежити можливості розробки;

- VHDL вимагає часу та зусиль для його вивчення, оскільки це мова зі своїм власним набором правил та концепцій.

У VHDL існують три типи моделювання:

- потокове моделювання даних: Цей тип моделювання описує, як дані переміщуються від входу до виходу. Використовуючи булеві вирази, він працює паралельно, дозволяючи одночасне виконання операцій на різних сигналах;

- поведінкове моделювання: Поведінкове моделювання використовується для послідовного виконання операторів. Воно показує, як система працює відповідно до поточного оператора. Цей тип моделювання може містити оператори процесу, послідовні оператори, оператори присвоєння сигналу та оператори очікування;

- структурне моделювання (підключення підмодулів): Структурне моделювання використовується для визначення функціональності та структури схеми. Цей тип моделювання містить оголошення сигналів, екземпляри компонентів та карти портів у екземплярі компонента. Він дозволяє збирати компоненти разом і визначати, як вони взаємодіють між собою.

У VHDL використовуються три типи об'єктів:

- константи: Константа є об'єктом, який містить фіксоване значення, яке не може змінюватися протягом виконання коду. Наприклад, константа number_of_bytes integer:=8;

- змінні: Змінна також містить значення певного типу, але це значення може бути змінене під час виконання моделі за допомогою оператора присвоєння. Змінні використовуються в процесах та підпрограмах. Приклад: variable index: integer :=0;

- сигнали: Сигнали оголошуються в архітектурі і можуть бути використані у будь-якому місці цієї архітектури. Значення сигналів призначається за допомогою оператора присвоєння "<=". Наприклад, Signal sig1: std_logic; Sig1 <= '1'

У мові VHDL існує кілька типів даних, які дозволяють визначати різні характеристики змінних та структур для моделювання електронних схем і систем.

Основні типи даних включають скалярні типи, такі як цілочисельні (Integer), числа з плаваючою комою (Floating point), перерахування (Enumeration) та фізичні типи (Physical).

Скалярні типи даних використовуються для представлення цілих чисел, чисел з плаваючою комою та наборів можливих значень. Наприклад, тип Integer дозволяє зберігати додатні і від'ємні цілі числа, а тип Floating point використовується для чисел з плаваючою комою, включаючи додатні та від'ємні значення. Тип Enumeration полегшує читабельність коду, оскільки дозволяє визначити набір можливих значень, обмежених певним списком.

Фізичні типи даних дозволяють описувати фізичні величини, такі як час, довжина або напруга, з використанням специфічних одиниць вимірювання.

Композитні типи даних включають масиви (Array) і записи (Record). Масиви використовуються для зберігання кількох значень одного типу під одним ідентифікатором і дозволяють зберігати та керувати колекціями даних. Записи використовуються для визначення групи елементів, кожен з яких має свою назву та тип даних.

Вони дозволяють створювати складні структури даних, які можуть містити різні типи інформації. Використання цих типів даних у мові VHDL допомагає визначати різні види змінних та структур для моделювання електронних схем і систем.

У мові VHDL можна використовувати різні типи операторів для виконання різних операцій та керування поведінкою даних. Основні типи операторів

включають логічні оператори, оператори порівняння, арифметичні оператори та оператори байтового зсуву.

Логічні оператори виконують операції "і", "або", "і негація", "або негація", "виключне або" та "виключне або негація". Наприклад, оператор "AND" повертає TRUE, якщо всі вхідні значення TRUE, оператор "OR" повертає TRUE, якщо хоча б одне з вхідних значень TRUE, а оператор "XOR" повертає TRUE, якщо тільки одне з вхідних значень TRUE.

Оператори порівняння використовуються для порівняння значень і повертають TRUE або FALSE залежно від умови порівняння. Наприклад, оператор "=" перевіряє, чи два значення дорівнюють одне одному, а оператор "<" перевіряє, чи одне значення менше за інше.

Арифметичні оператори включають додавання, віднімання, множення, ділення, конкатенацію, цілочисельне ділення, залишок від цілочисельного ділення, модуль, зведення в степінь. Вони використовуються для виконання математичних операцій над числами.

Оператори байтового зсуву використовуються для зсуву бітів вліво або вправо, а також для повороту бітів вліво або вправо. Вони дозволяють зміщувати біти у числі, що може бути корисним у деяких ситуаціях. Використання цих операторів у мові VHDL дозволяє виконувати різні операції з даними та керувати їхнім поведінкою.

3.3 Опис розробки кодера та декодера

За результатами аналізу, що був проведений у попередньому розділі, було створено кодер та декодер Ріда-Соломона, використовуючи мову VHDL.

Приклад фрагменту коду 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)) ;
```

Приклад фрагменту коду 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;
```

3.4 Створення моделі, яка відображає взаємодію кодера і декодера

У цьому розділі проводиться аналіз взаємодії кодера і декодера Ріда-Соломона. Схема системи представлена на рисунку 3.1 і включає джерело даних, кодер, канал зв'язку, декодер і осцилограф.

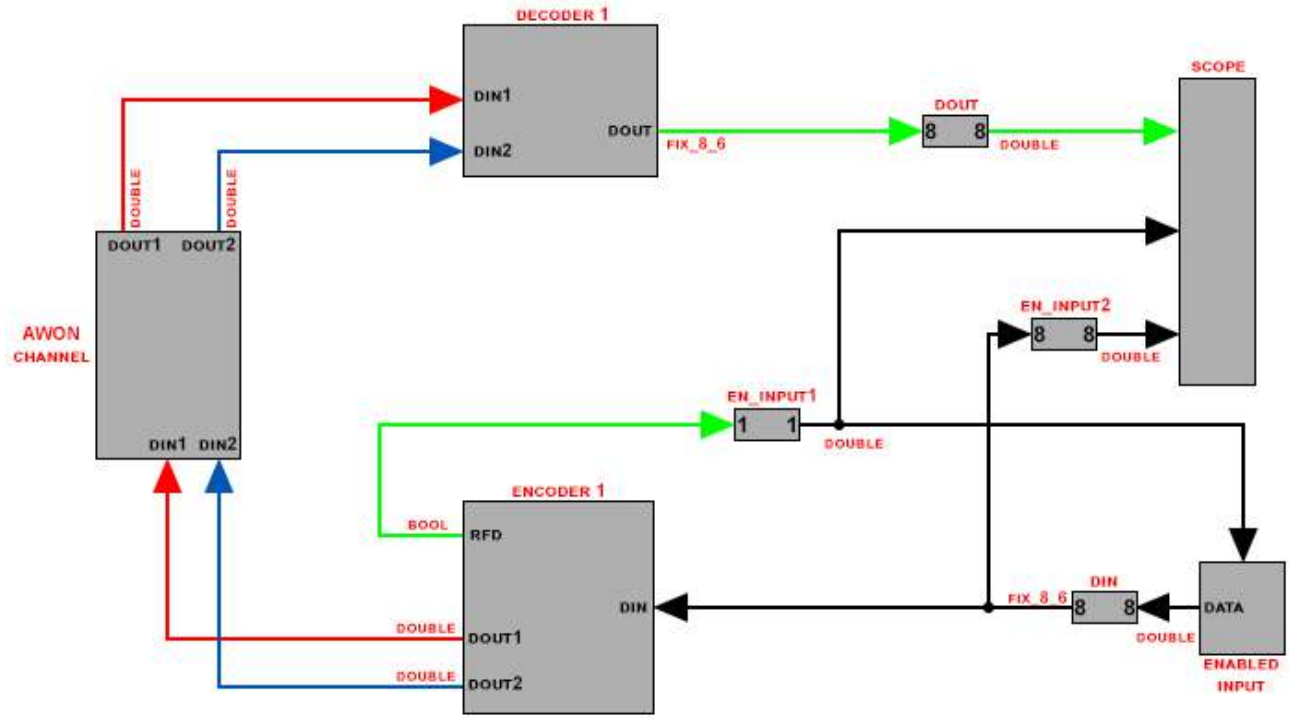


Рисунок 3.1 – Схема роботи системи з корекцією помилок

Джерело генерує дискретизований і квантований сигнал, який має синусоїдальну форму. Блок кодера Encoder1 складається з кількох функціональних блоків. Ці блоки включають кодер Ріда-Соломона (OuterEncoder), перемішувач (Interleaver), згортковий кодер (InnerEncoder) і вибірковий кодер (Puncturing). Блок затримки (EnabledDelay) використовується для врахування часу виконання процесу кодування. У кодері повного коду Ріда-Соломона (255, 251) формується блок OuterEncoder.

3.5 Синтез і симуляція програмно-технічного модулю

Програмне забезпечення, яке використовується для компіляції та синтезу, — Altera Quartus ii. Тест стенд створено для виконання моделювання. У тестовому модулі введення десяткової дроби створено числа від 1 до 204. Передати вхідні дані в кодер після обчислення, отриманий результат є кодовим

словом. Програмне забезпечення, яке використовується для моделювання кодера Modelsim на рисунку 3.2.

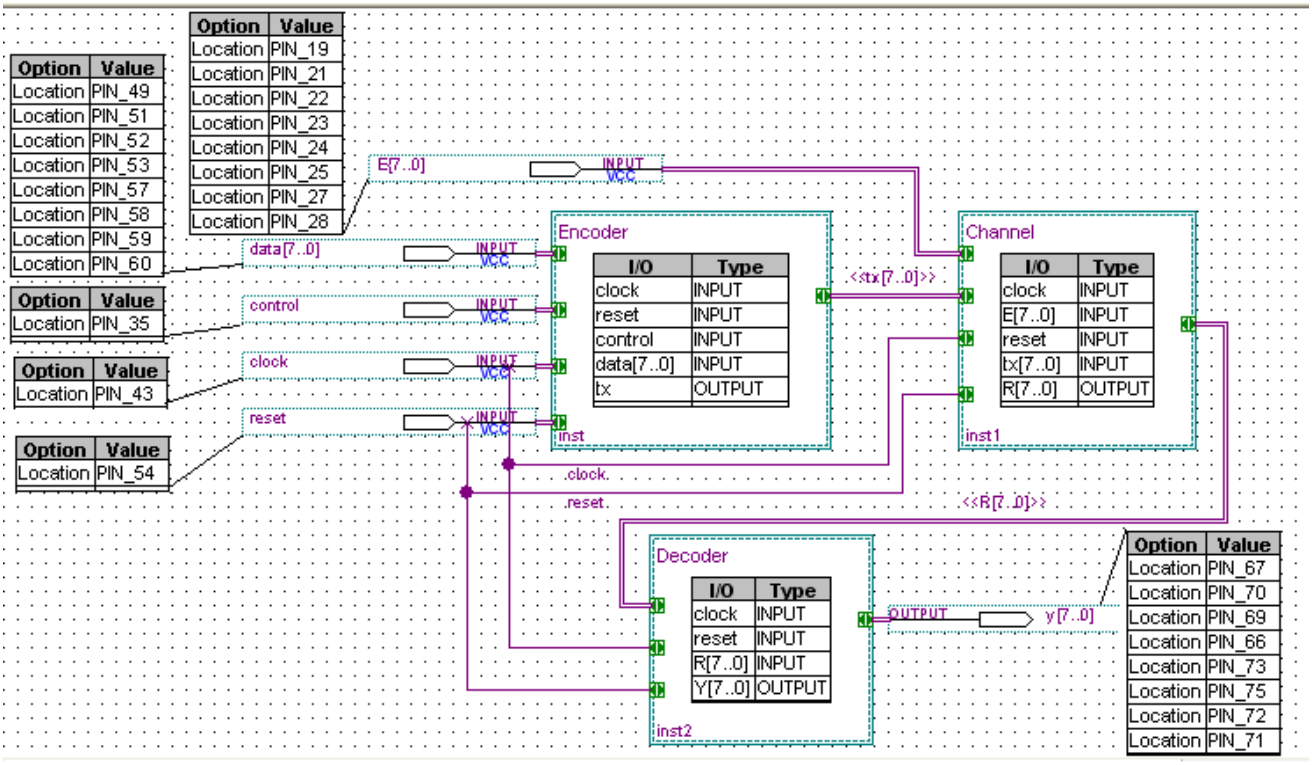


Рисунок 3.2 – Архітектура кодера/декодера Ріда-Соломона

Для зручного порівняння використовується програма MATLAB для створення контрольних бітів парності. Програма написана таким чином:

```
msg=ones(1, 239);
for i=1:51;
msg(i)=0;
end
for j=52:239;
msg(j)=(j-51);
end
m=8;n=255;k=239;b=0;
genpoly=rsgenpoly(n, k, [], b);
msg_p=gf(msg, m);
```

cwd=rsenc(msg_p, n, k, genpoly)

Шістнадцять символів перевірки парності: 195, 231, 90, 194, 142, 112, 85, 171, 63, 242, 251, 154, 1, 82, 33, 222.

Моделюємо схему в Modelsim, щоб отримати форму сигналу моделювання, як показано нижче рисунок 3.3:

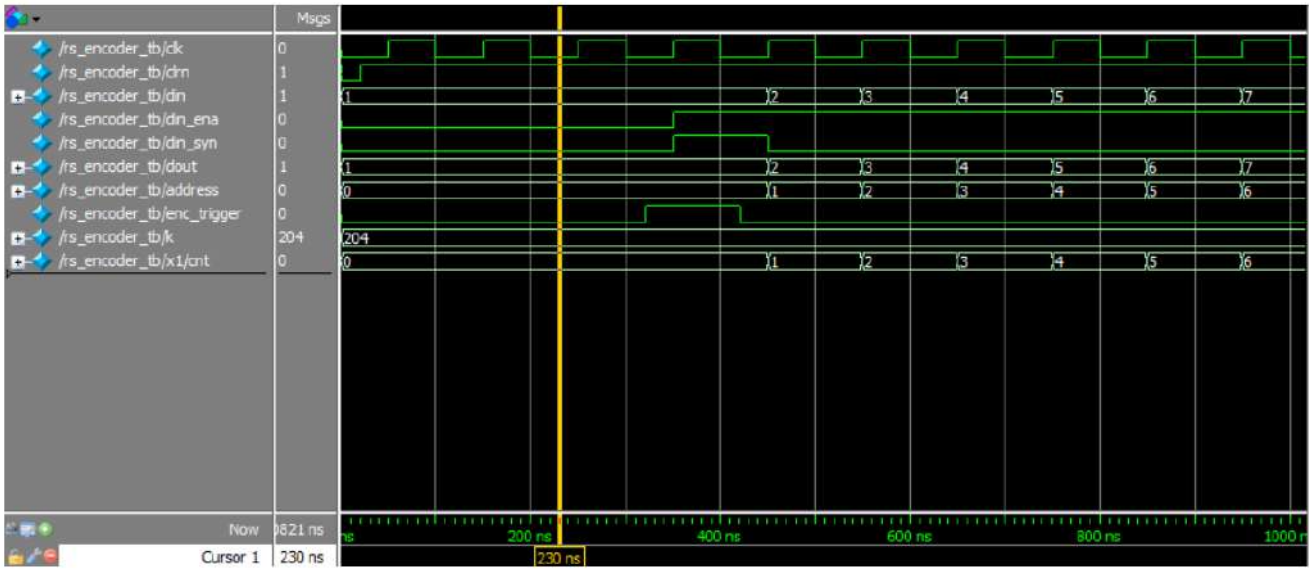


Рисунок 3.3 – Сигнал кодера (1)

На рисунку 3.3 тестуються сигнали «clrn», «din_ena» і «cnt». Від хвилі, коли «clrn» дорівнює 0, «cnt» дорівнює 0; коли «din_ena» змінювався з низького на високий, «cnt» залишався 0. Отже, три сигнали задовольняють вимоги конструкції.

На рисунку 3.4 показано стан схеми після кількох тактових циклів. У вигляді хвильової форми "din_ena" та "clrn" дорівнюють 1. Кодер почав працювати. Передано перші 188 інформаційних символів в систему. Згідно з теорією кодування, виходи повинні бути такими ж, як ці 188 вхідних символів. З рисунка видно, що перші 188 виходів співпадають з вхідними символами.

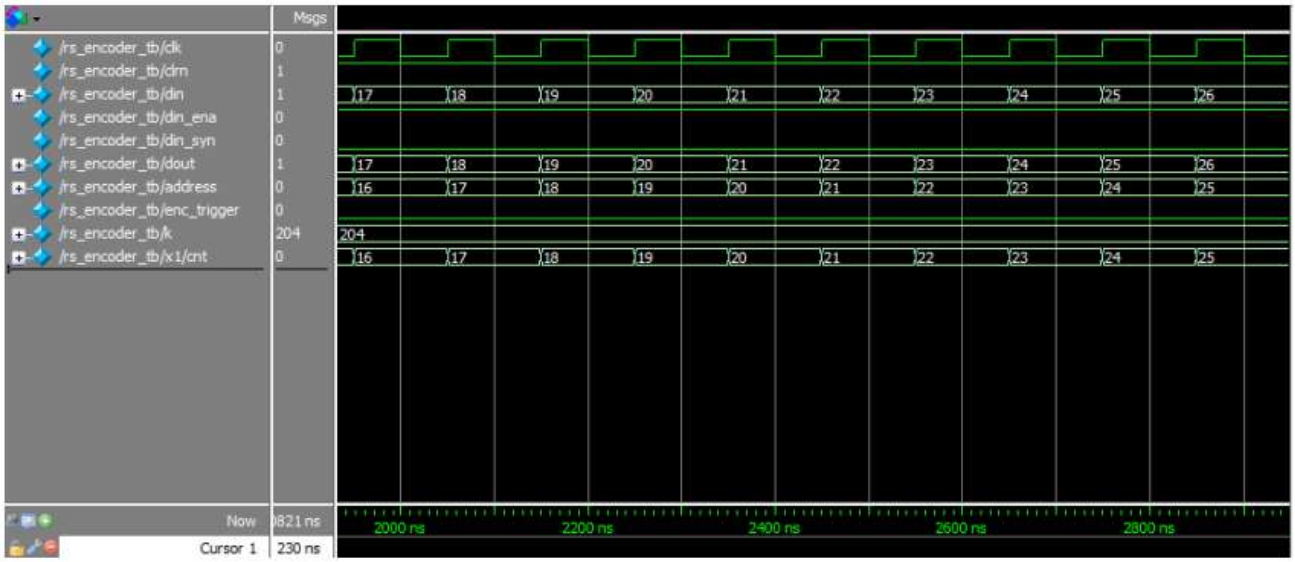


Рисунок 3.4 – Сигнал кодера (2)

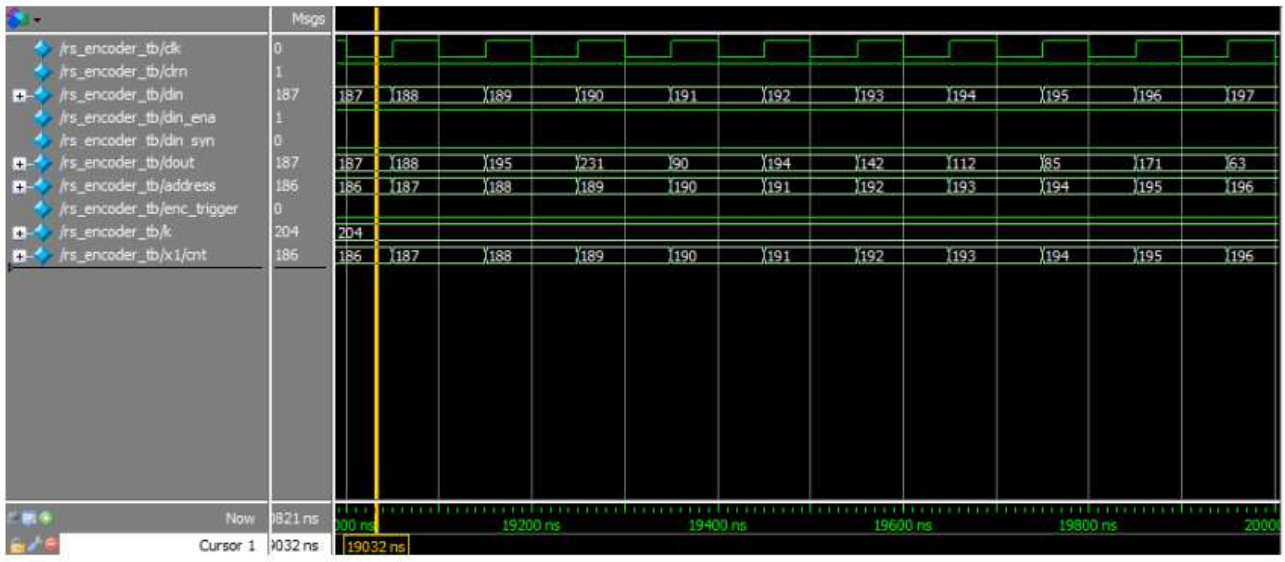


Рисунок 3.5 – Сигнал кодера (3)

Вищенаведені два рисунки відображають кодування останніх 16 вхідних символів. З хвильової форми видно, що 188-й символ "dout" має значення 188, а наступні 16 символів мають значення 195, 231, 90, 194, 142, 112, 85, 171, 63, 242, 251, 154, 1, 82, 33, 222, що узгоджується з результатами програми MATLAB.

На основі проведеного аналізу можна зробити висновок, що розроблений кодер має здатність правильно генерувати кодові слова.

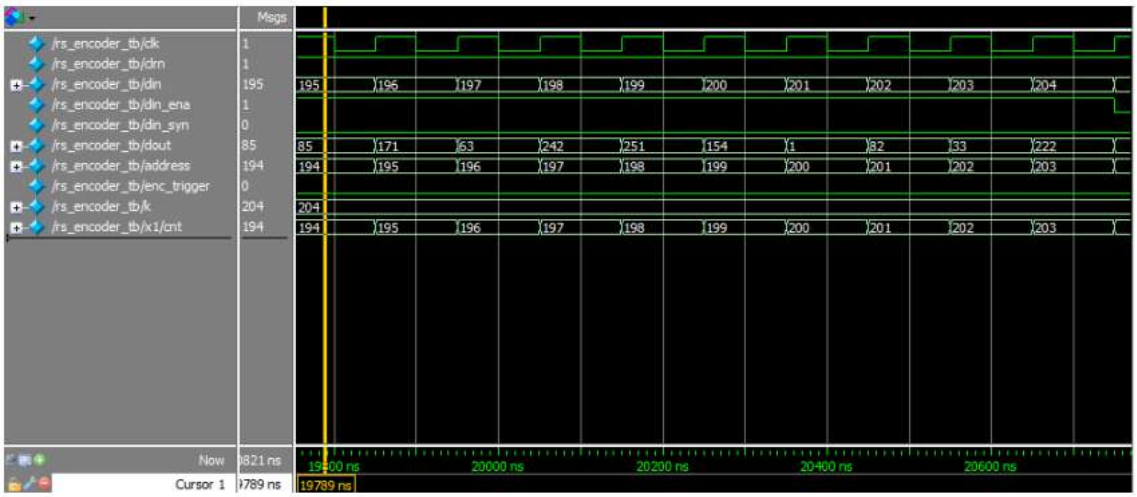


Рисунок 3.6 – Сигнал кодера (4)

Апаратне проектування кодера реалізовано з використанням EP2S15F484C3 Stratix II. Аналіз представлено на рисунку 3.7.

Quartus II Version	9.0 Build 132 02/25/2009 SJ Full Version
Revision Name	Encoder
Top-level Entity Name	encode
Family	Stratix II
Met timing requirements	Yes
Logic utilization	2 %
Combinational ALUTs	177 / 12,480 (1 %)
Dedicated logic registers	136 / 12,480 (1 %)
Total registers	136
Total pins	20 / 343 (6 %)
Total virtual pins	0
Total block memory bits	0 / 419,328 (0 %)
DSP block 9-bit elements	0 / 96 (0 %)
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 2 (0 %)
Device	EP2S15F484C3
Timing Models	Final

Рисунок 3.7 – Аналіз компонентів кодера

Для декодера створено векторний файл хвильової форми та кілька файлів ".mif" для перевірки функціональності декодера. Використовувані програмні засоби - Altera Quartus ii та MATLAB. Векторний файл хвильової форми був створений у Quartus ii. Файли ".mif" створені за допомогою MATLAB.

Наступні три рисунки 3.8, 3.9, 3.10, показують вхідний сигнал у безпомилковій ситуації. У хвильових формах "clk" - це тактовий сигнал; "rst" - це кнопка скидання; "sync" - це кнопка включення; "Din" - це вхід для декодера; "Dout" - це вихід.

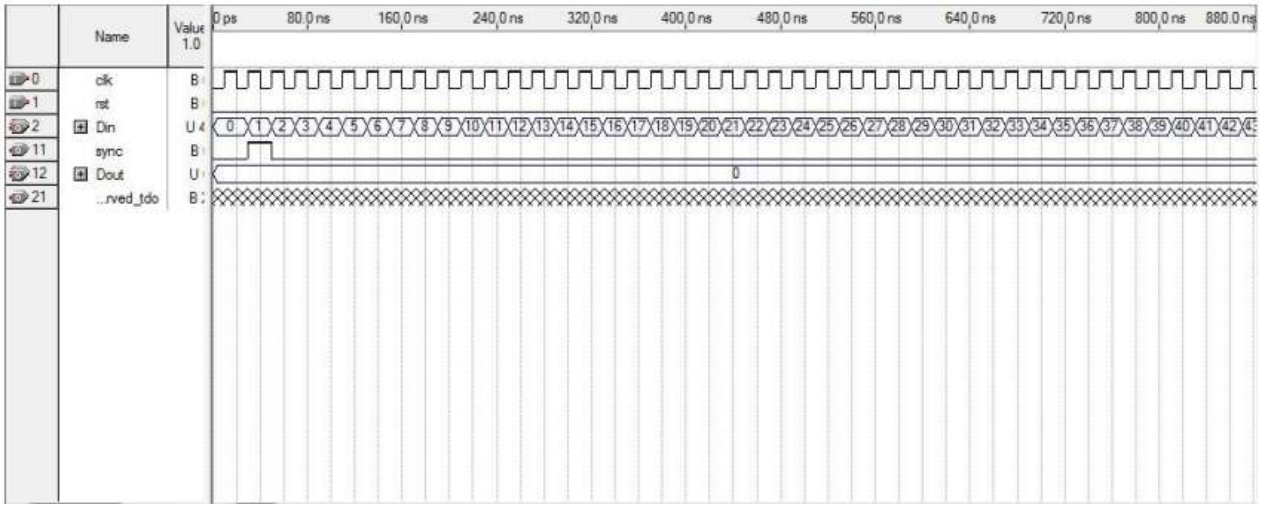


Рисунок 3.8 – Часова схема вхідного сигналу без помилок для декодера (1)

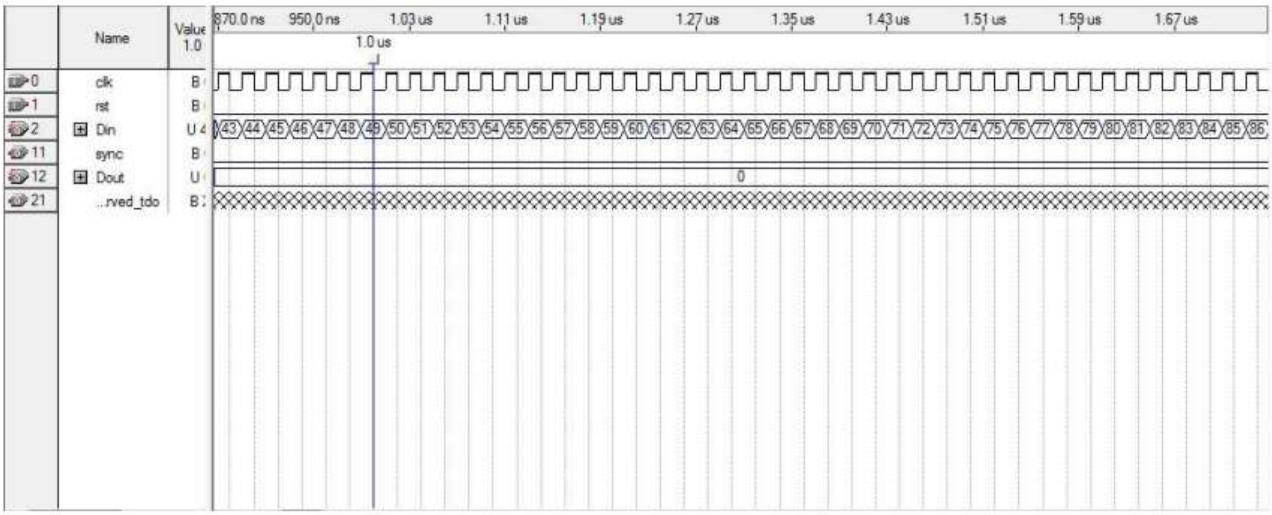


Рисунок 3.9 – Часова схема вхідного сигналу без помилок для декодера (2)

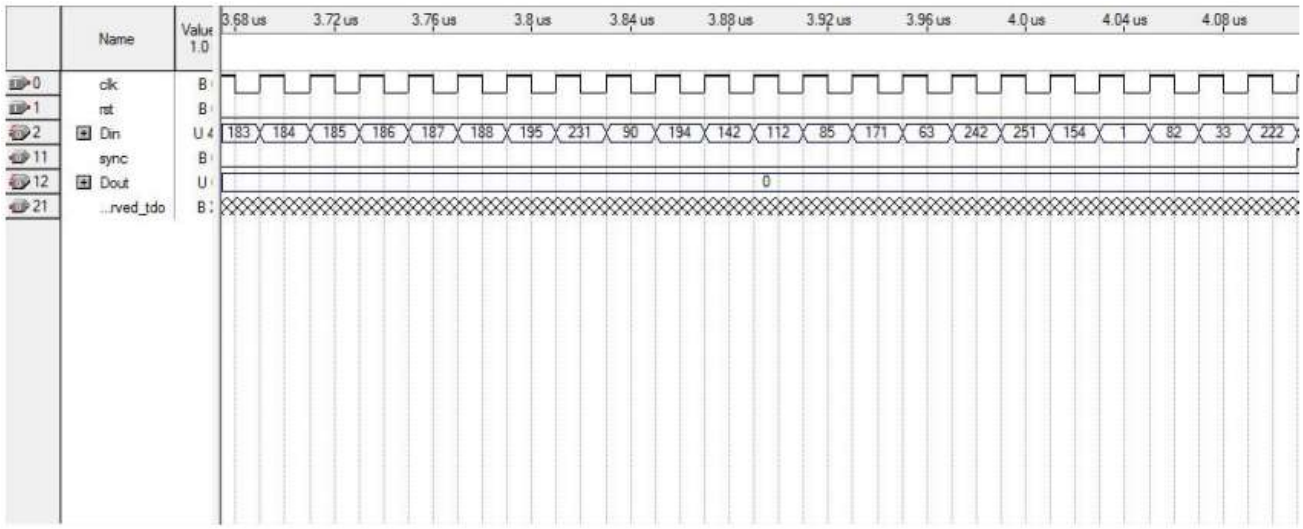


Рисунок 3.10 – Часова схема вхідного сигналу без помилок для декодера (3)

Наступні три рисунки показують вихід декодера у випадку відсутності помилок.

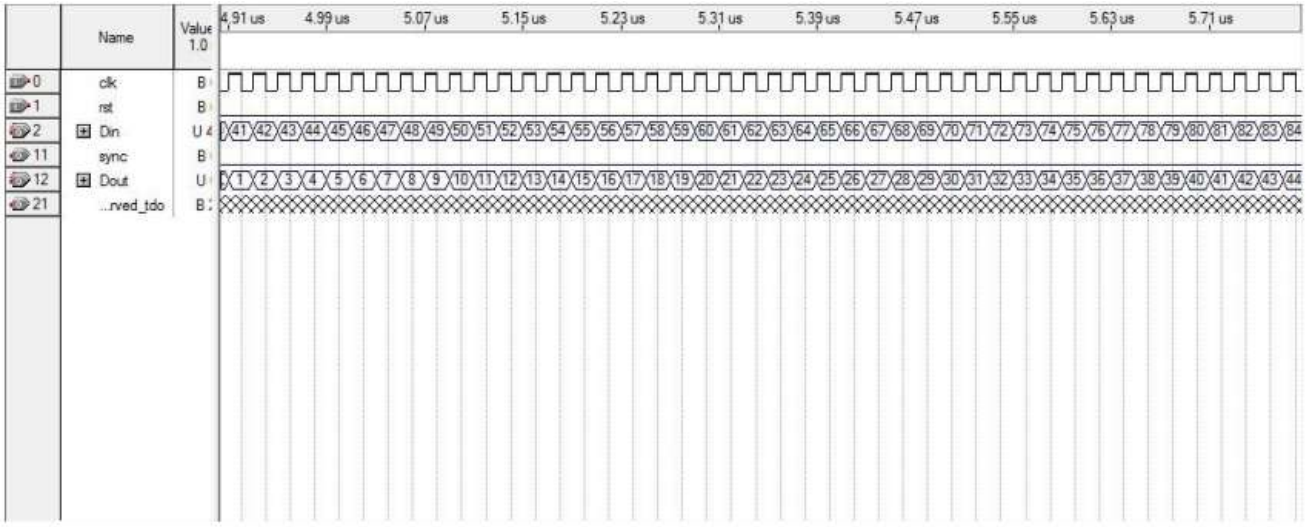


Рисунок 3.11 – Часова схема вихідного сигналу без помилок для декодера (1)

Часова схема вихідного сигналу без помилок для декодера Ріда-Соломона відображає послідовність дискретних часових відліків на осі часу. Нижче описані три рисунки, що демонструють вихідний сигнал декодера при відсутності помилок

Перший рисунок показує часову схему вихідного сигналу без помилок для декодера. На горизонтальній осі зображений час, а на вертикальній осі - значення вихідного сигналу. Вихідний сигнал підтверджує правильне декодування вхідного кодового слова і має стабільну високу амплітуду протягом всього часу.

Другий рисунок показу схеми вихідного сигналу без помилок для декодера Ріда-Соломона. На цьому рисунку видно, що вихідний сигнал декодера має більш складну структуру, що відображає послідовність бітів, які передаються з вхідного кодового слова. Кожен біт відображається як високий або низький рівень сигналу протягом відведеного часового інтервалу.

На третьому рисунку можна побачити більш деталізовану часову схему вихідного сигналу без помилок для декодера Ріда-Соломона. Цей рисунок показує послідовність бітів на вертикальній осі, а по горизонтальній осі відображений час. Кожен біт відображається як високий або низький рівень сигналу протягом певного часового інтервалу. За відсутності помилок вихідний сигнал декодера має точно відповідати вхідному кодовому слову.

Ці часові схеми демонструють як правильне функціонування декодера Ріда-Соломона при відсутності помилок. Вихідний сигнал відображає точне відтворення вхідного кодового слова, що підтверджує ефективність і надійність алгоритму декодування Ріда-Соломона. При належному проектуванні та реалізації декодера можна досягти високої якості декодування та відсутності помилок при передачі та отриманні даних.

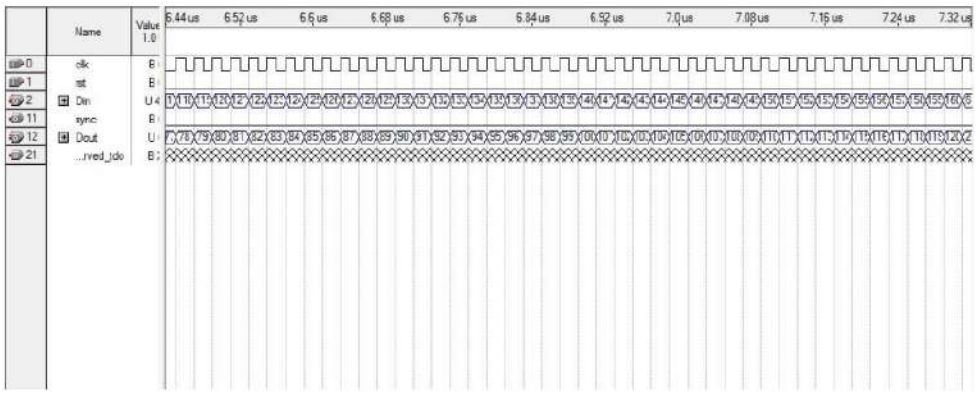


Рисунок 3.12 – Часова схема вихідного сигналу без помилок для декодера (2)

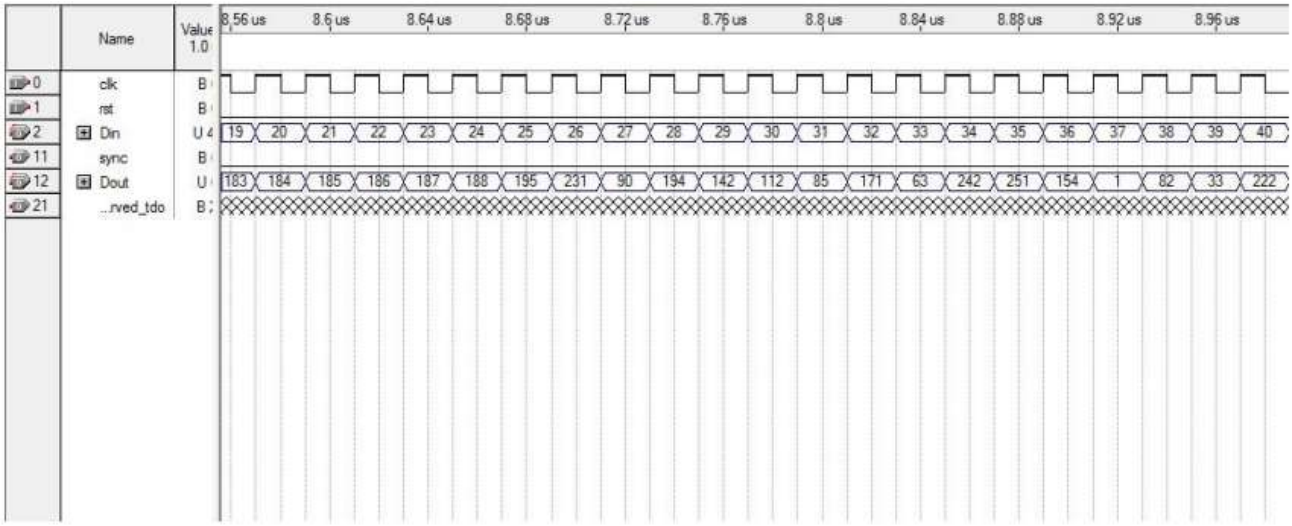


Рисунок 3.13 – Часова схема вихідного сигналу без помилок для декодера (3)

З вищезазначених сигналів вимірювання можна побачити, що декодер може правильно розкодувати без помилок.

Нижче наведено часові схеми, що показують вхідні та вихідні дані декодера, коли в кодовому слові є 6 помилок.

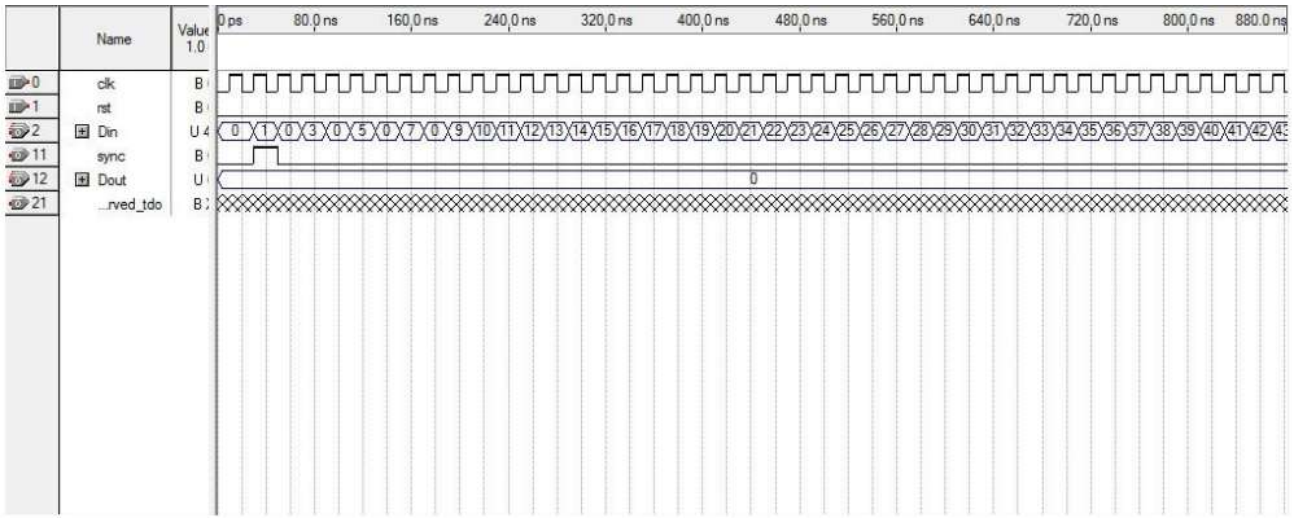


Рисунок 3.14 – Часова схема вхідного/вихідного сигналу з 6 помилками для декодера (1)

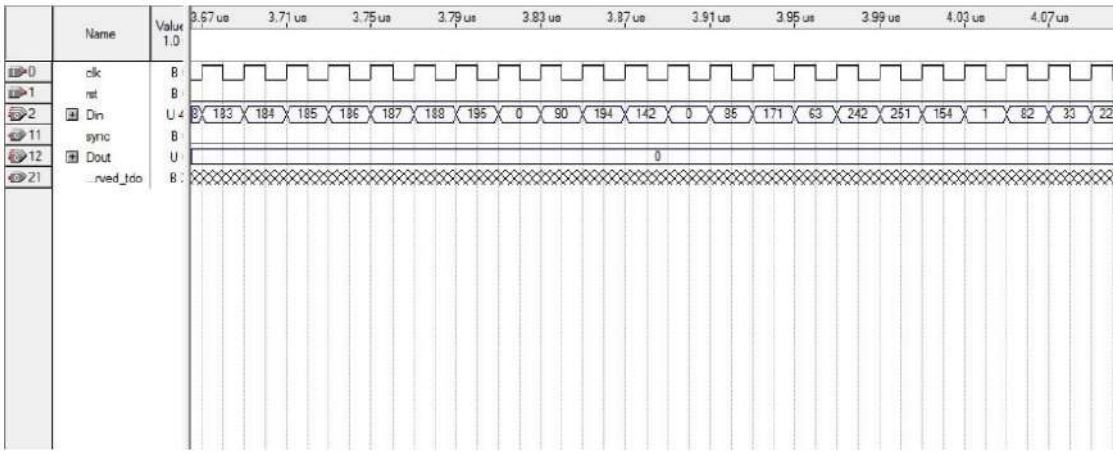


Рисунок 3.15 – Часова схема вхідного/вихідного сигналу з 6 помилками для декодера (2)

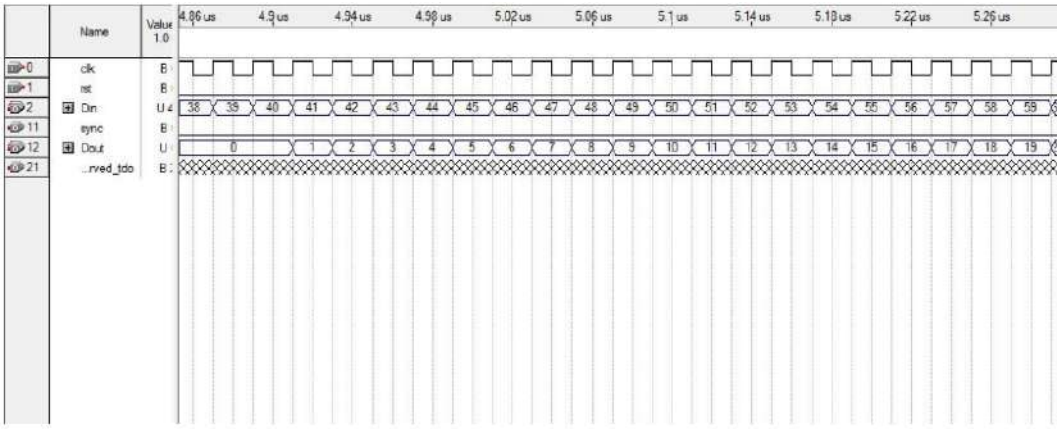


Рисунок 3.16 – Часова схема вхідного/вихідного сигналу з 6 помилками для декодера (3)

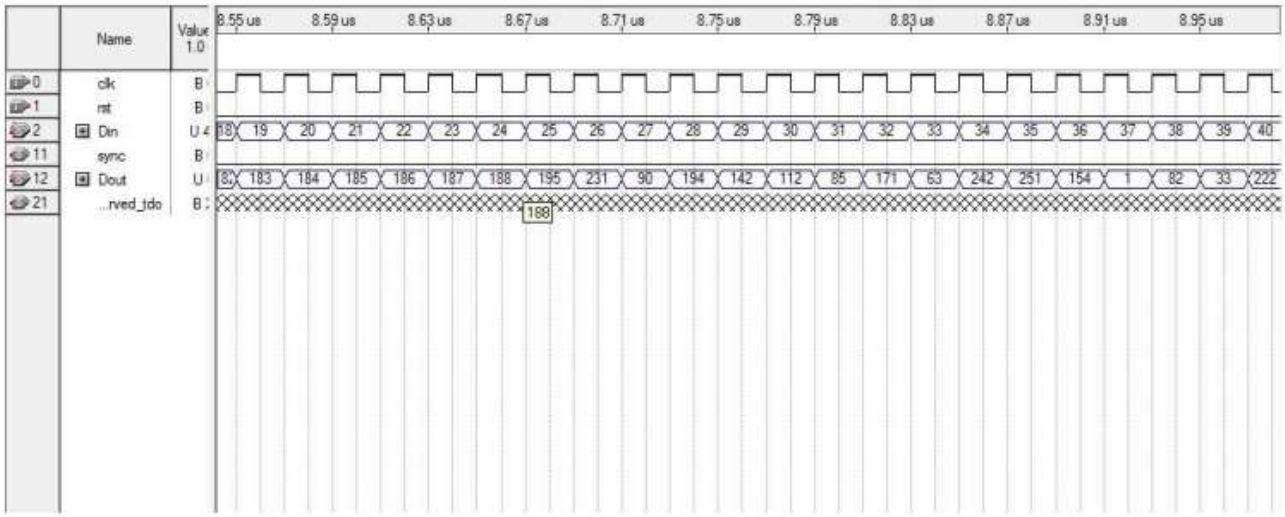


Рисунок 3.17 – Часова схема вхідного/вихідного сигналу з 6 помилками для декодера (4)

З вищезазначених рисунків видно, що декодер виправив 6 помилок (4 в символах інформації та 2 в символах перевірки парності), розташованих у кодовому слові.

Наступний набір даних містить 8 помилок у вхідних символах.

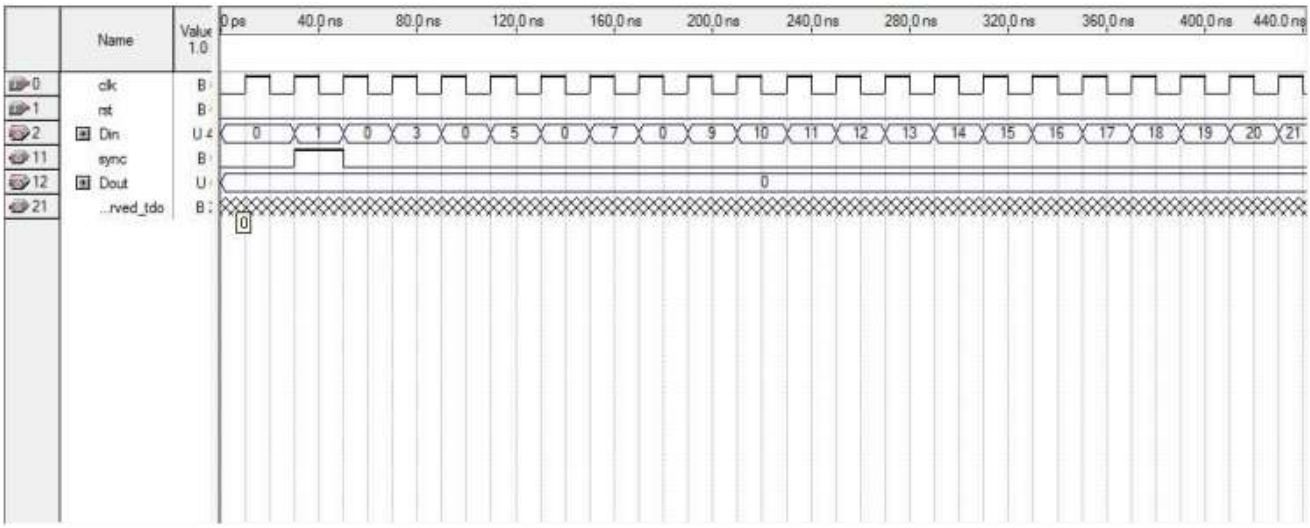


Рисунок 3.18 – Часова схема вхідного/вихідного сигналу з 8 помилками для декодера (1)

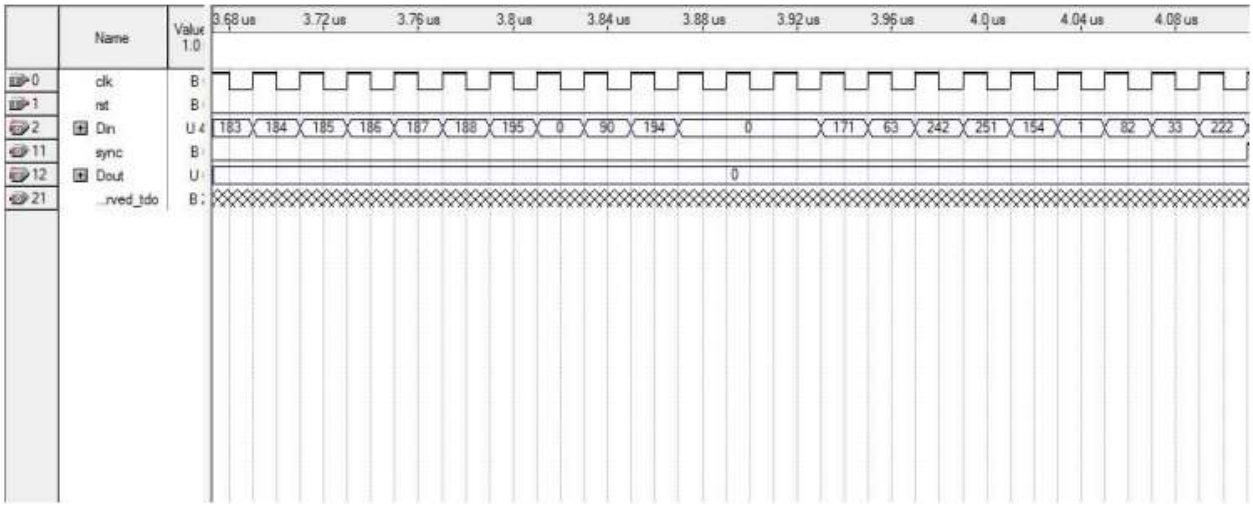


Рисунок 3.19 – Часова схема вхідного/вихідного сигналу з 8 помилками для декодера (2)

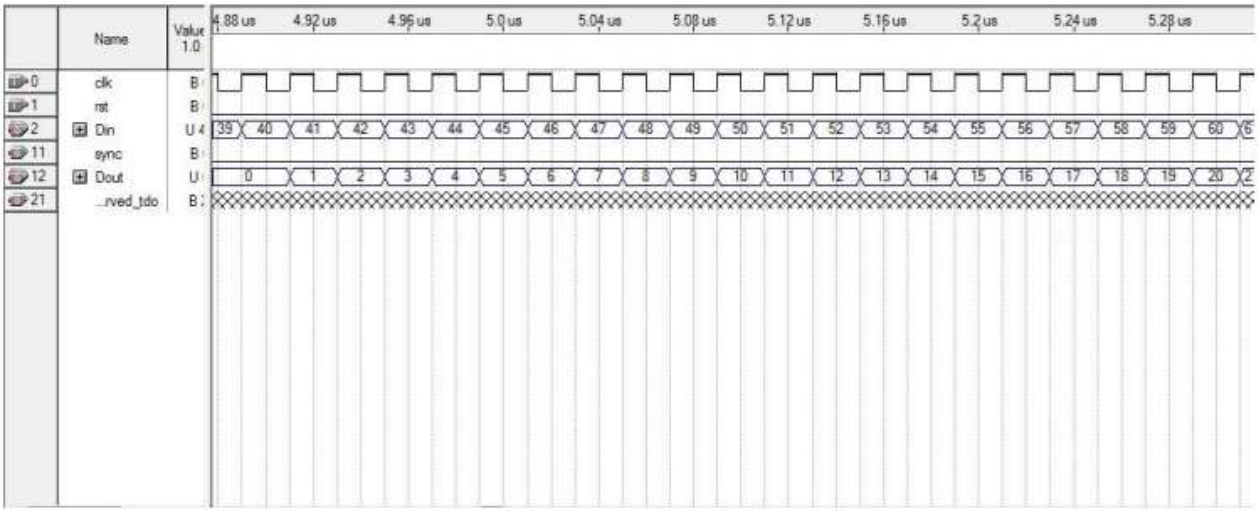


Рисунок 3.20 – Часова схема вхідного/вихідного сигналу з 8 помилками для декодера (3)

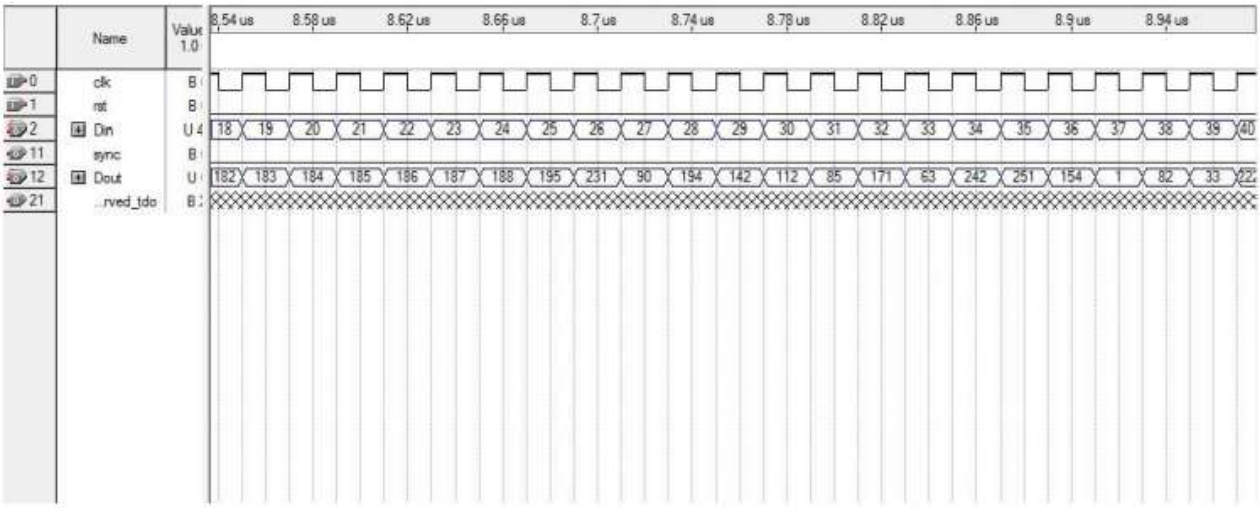


Рисунок 3.21 – Часова схема вхідного/вихідного сигналу з 8 помилками для декодера (4)

За допомогою вищенаведених форм сигналу можна зробити висновок, що коли в кодовому слові не більше 8 символів з помилками, декодер може точно виконати корекцію помилок.

Наступне тестування підтвердило, що коли кількість помилок перевищує 8, корекція помилок буде невдалою.

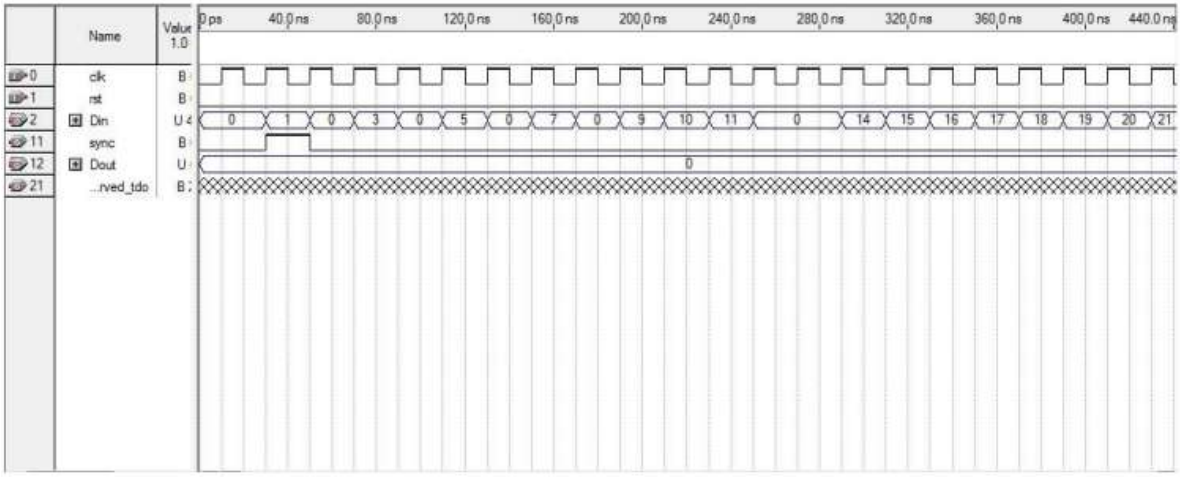


Рисунок 3.22 – Часова схема вхідного/вихідного сигналу з 9 помилками для декодера (1)

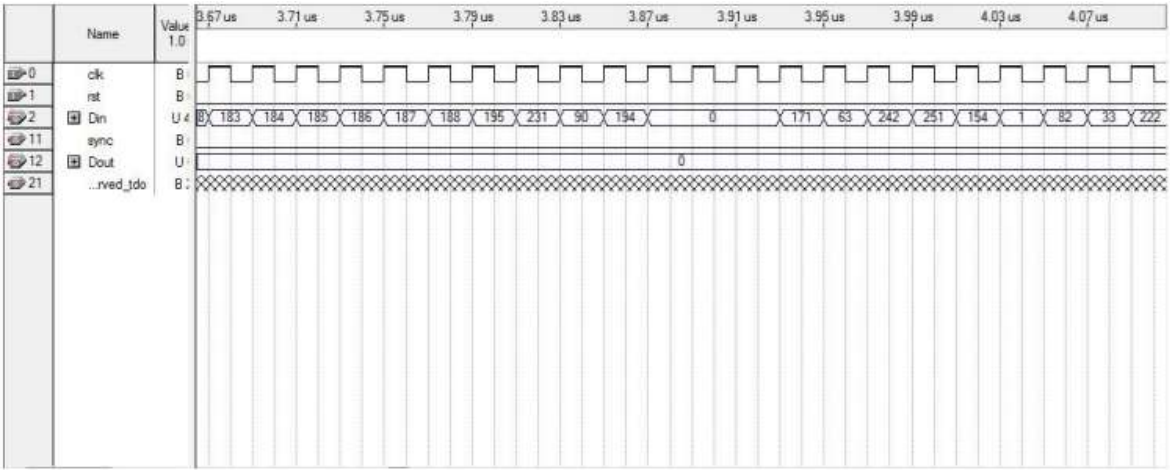


Рисунок 3.23 – Часова схема вхідного/вихідного сигналу з 9 помилками для декодера (2)

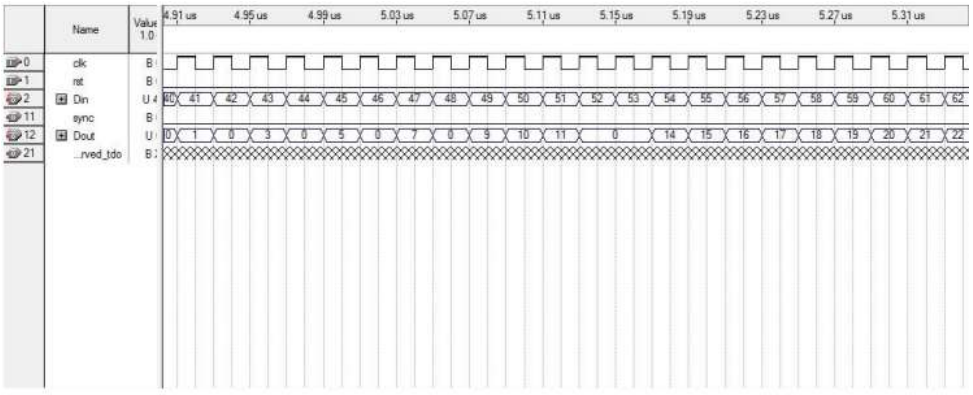


Рисунок 3.24 – Часова схема вхідного/вихідного сигналу з 9 помилками для декодера (3)

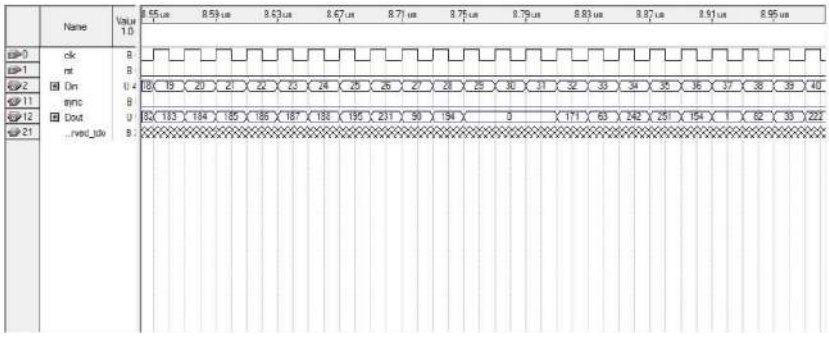


Рисунок 3.25 – Часова схема вхідного/вихідного сигналу з 9 помилками для декодера (4)

З усіх вищевказаних часових схем можна зрозуміти, що коригуюча здатність коду Ріда-Соломона (204, 188) становить 8 символів. Апаратне проектування кодера реалізовано за допомогою EP2C35F672C6 Cyclone II. Аналіз показано на рисунку 3.26.

Quartus II Version	9.0 Build 132 02/25/2009 SJ Full Version
Revision Name	rs_decoder204_188
Top-level Entity Name	rs_decoder204_188
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	Yes
Total logic elements	4,492 / 33,216 (14 %)
Total combinational functions	4,049 / 33,216 (12 %)
Dedicated logic registers	1,981 / 33,216 (6 %)
Total registers	1981
Total pins	19 / 475 (4 %)
Total virtual pins	0
Total memory bits	26,088 / 483,840 (5 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

Рисунок 3.7 – Аналіз компонентів декодера

3.6 Висновок

За допомогою програмованих користувачем вентильних матриць (FPGA) та мови проектування VHDL була здійснена апаратна реалізація коду Ріда-Соломона. FPGA забезпечують гнучкість та перепрограмовуваність, що дозволяє реалізувати різні логічні функції та ефективно взаємодіяти з різними системними компонентами. Мова VHDL надає засоби для детального опису поведінки

апаратних компонентів на різних рівнях деталізації, що сприяє точному моделюванню та розробці електронних систем.

Створений кодер і декодер Ріда-Соломона були успішно використані для кодування та декодування даних. Взаємодія між кодером і декодером була вивчена та проаналізована за допомогою системної схеми, яка включає джерело даних, кодер, канал зв'язку, декодер і осцилограф. Також була проведена симуляція програмно-технічного модулю з використанням Altera Quartus II та Modelsim. Це дозволило перевірити правильність роботи кодера та декодера та зрозуміти їхню коригуючу здатність. Апаратне проектування кодера було здійснено з використанням FPGA EP2C35F672C6 Cyclone II. Загальною висновкою є те, що апаратна реалізація коду Ріда-Соломона з використанням FPGA та мови VHDL є ефективним та гнучким рішенням. Вона дозволяє забезпечити надійну передачу та коригування даних в системах з помилками. Подальше дослідження та вдосконалення в області апаратної реалізації коду Ріда-Соломона можуть сприяти розвитку та вдосконаленню комунікаційних систем.

ВИСНОВКИ

Як особливий тип кодів для контролю помилок, коди Ріда-Соломона через свою високу здатність коригувати як випадкові, так і групові помилки широко застосовуються в різних типах цифрових комунікаційних систем, особливо в системах цифрового відеомовлення (DVB), системах високої чіткості (HDTV), супутникових комунікаційних системах та системах зберігання даних.

Під час виконання бакалаврського дипломного проєкту був розроблений пристрій для корекції помилок на базі коду Ріда-Соломона з використанням FPGA, а також було проведено моделювання його роботи. В ході проєкту були досліджені коригуючі коди, їх властивості та типи. Було висвітлено основні тези щодо використання та необхідності коригуючих кодів. Розроблений пристрій базується на коді Ріда-Соломона для корекції незалежних помилок. Він забезпечує кодування та декодування інформаційних слів довжиною до 255 бітів і може виправляти будь-які дві незалежні помилки в будь-якому слові. Під час розробки проєкту була створена функціональна схема пристрою, вибрана оптимальна матриця декодування і розроблений алгоритм його роботи. Для підтвердження працездатності пристрою була створена його модель, і проведені тести на контрольних наборах даних. При побудові моделі використовувалися мови програмування C++ та VHDL.

В результаті виконання бакалаврського дипломного проєкту було підтверджено, що використання коригуючих кодів для збільшення надійності передачі даних та захисту від пошкодження є ефективним засобом.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Науменко М.І., Стасев Ю.В., Кузнецов О.О. Теоретичні основи та методи побудови алгебраічних блокових кодів: навч. посіб. Харків: ХУПС, 2005. 267 с.

2. Сидоров Г. І. Основи теорії передачі: конспект лекцій. Харків: ХНУРЄ, 2006. 78 с.

3. Волочій Б. Ю. Передавання сигналів у інформаційних системах. Частина 1. Навчальний посібник. Львів: Видавництво Львівської політехніки, 2005.

4. Shu Lin and Daniel J. Costello, Error Control Coding: Fundamentals and Applications, Prentice-Hall, Inc., Englewood Cliffs. N.J. 2004.

5. I.S.Reed and G.Solomon, Polynomial Code over Certain Finite Fields, Journal of the Society for Industrial and Applied Mathematics, 1960, Vol. 8, No. 2, P 300-304.

6. Thomas M. Cover and Joy A. Thomas, Elements of Information Theory, Wiley, July, 2006.

7. K.Y. Liu, Architecture for VLSI Design of Reed-Solomon Encoder. IEEE Transactions on Information Theory, 1982, Vol. IT28, No. 6, P 869-874.

8. Gadiel Seroussi, A Systolic Reed-Solomon Encoder: IEEE Transactions on Information Theory, 1991, Vol. 4, No. 4, P 1217-1220.

9. M.A. Hasan and V.K. Bhargava, Architecture for a Low Complexity RateAdaptive Reed-Solomon Encoder. IEEE Transactions on Computers, 1995, Vol. 44, No. 7, P 938-942.

10. E.R. Berlekamp, Algebraic Coding Theory, McGraw-Hill, New York, 1967.

11. J.L. Massey, Shift-Register Synthesis and BCH Decoding, IEEE Transactions on Information Theory, 1968, No. 15,P 122-127.

12. H.M. Shao and I.S. Reed, A Systolic VLSI Design of a Pipeline Reed-Solomon Decoder, TDA Progress Report, 42-76J, 99-113, 1983.

13. K.Y. Liu, Architecture for VLSI Design of Reed-Solomon Decoders, IEEE Transactions on Computers, 1984, Vol. C, No. 33, P 178-189.

14. A. Shiozaki, K. Okuno, and K. Suzuki, A Hybrid ARQ Scheme with Adaptive Forward Error Correction for Satelite Communications. IEEE Transactions on Computers, 1991, Vol. 39, No. 4, P 482-484.

15. D. Dabiri and I.F. Blake, Fast Parallel Algorithms for Decoding Reed-Solomon Codes Based on Remainder Polynomials, IEEE Transactions on Information Theory, 1995, Vol. 41, No. 4, P 873-885.

16. G.D. Forney, Generalized Minimum Distance Decoding, IEEE Transactions on Information Theory, Vol. IT-12, No. 2, P 125-131.

17. C. R. H. Morelos-Zaragoza, The Art of Error Correcting Coding, p. 1, 74, John Wiley & Sons Ltd, Chichester (2006).

18. A. Betten, Error-Correcting Linear Codes: Classification by Isometric and Applications, p. 7, 4, Springer, Berlin (2006).

19. G.D. Forney, On Decoding BCH Codes, IEEE Transactions on Information Theory, 549-557.

20. Min-An Song and Sy-Yen Kuo, A Low Complexity Design of Reed Solomon Code Algorithm for Advanced RAID System, IEEE, 265-273, 2007.

21. Ufuk Demir and Ozlem Akta, Raptor versus Reed Solomon Forward Error Correction Codes, IEEE ISCN, 264-269, 2006.

22. National Aeronautics and Space Administration (NASA) technical reports, “Tutorial on Reed-Solomon error correction coding”, Lyndon B. Johnson Space Center Houston, 1990.

23. Aqib. Al Azad, Minhazul. Huq, Iqbalur, Rahman Rokon, “Efficient Hardware Implementation of Reed Solomon Encoder and Decoder in FPGA using Verilog”, International Conference on Advancements in Electronics and Power Engineering (ICAEPE'2011) Bangkok Dec., 2011.

24. Bhawna Tiwari, Rajesh Mehra, “Design and Implementation of Reed Solomon Decoder for 802.16 Network using FPGA” 978-1-4673-1318, IEEE 2012.

25. Donald G. Bailey, “Design for Embedded Image Processing on FPGA’S” 2011.

26. J. C. Moreira and P. G. Farrell, Essentials of Error-Control Coding, p. 2, 166, 3, 166 , John Wiley & Sons Ltd, Chichester (2006).

27. M. A. Ingale, Error Correcting Codes in Optical Communication Systems, Gothenburg (2003).

28. T. Le-Ngoct, M. T. Vot, B. Mallettt and V. K. Bhargava, in Military Communications Conference, 1990. MILCOM '90, Conference Record, A New Era/1990, p. 121-125, A gate-arraybased programmable Reed-Solomon codec:structureimplementation- applications, Monterey, CA (1990).

29. A. Halbutogullari and Ç. K. Koç, in IEEE Transactions on Computers/2000, p. 503-518, Mastrovito Multipler for General Irreducible Polynomials, Corvallis (2000).

30. G.C. Cardarilli, S. Pontarelli, M. Re, and A. Salsano, Analysis of Errors and Erasures in Parity Sharing RS Codecs, IEEE, 1721-1726, 2007.

31. Lionel Biard, Dominique Noguet, “Reed-Solomon Codes for Low Power Communications”, Journal of Communications, Vol. 3, No. 2, April 2008, pp. 13-21.

32. Priyanka Dayal and Rajeev Kumar Patial, “FPGA Implementation of Reed-Solomon Encoder and Decoder for Wireless Network 802.16”, International Journal of Computer Applications, April 2013, Volume 68, No.16, P 42-45.

33. Petrus Mursanto, “Generic Reed Solomon encoder”, Makara, Sains, 2006, Vol. 10, No. 2, P 58-62.

34. Amandeep Singh and Mandeepkaur, “Study of Reed Solomon Encoder”, International Journal of Innovative Research in Computer and Communication Engineering Vol. 1, Issue 2, April 2013.

35. J. Jittawutipoka and J. Ngarmnil, “Low complexity reed solomon encoder using Globally optimized finite field multipliers”, IEEE region 10 conference, vol. 4, Nov. 2004, pp. 423-426.

36. M. Song, S. Kuo, I. Lan, in Consumer Electronics, IEEE Transactions, p. 265-273, A Low Complexity Design of Reed Solomon Code Algorithm for Advanced RAID System, Rosemont, IL (2007).

37. M. A. Ingale, Error Correcting Codes in Optical Communication Systems, Gothenburg (2003).

38. B. Sklar. Reed-Solomon Codes, (2001).

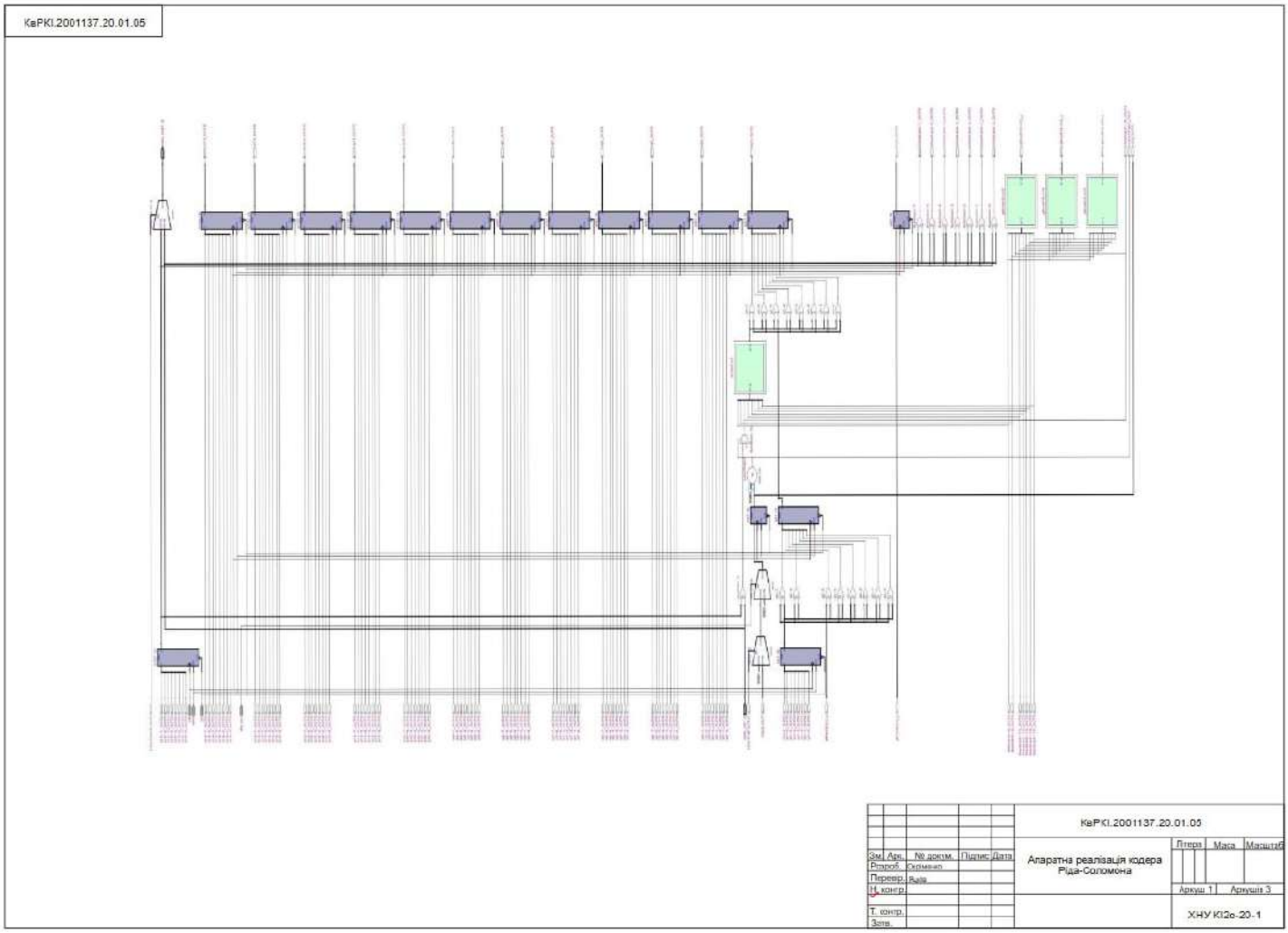
39. G. Mitchell, “Investigation of Hamming , Reed-Solomon , and Turbo Forward Error Correcting Codes,” Army Research Laboratory, Adelphi, ARL-TR-4901, Tech. Rep. Jul., 2009.

40. Peter J. Ashenden, “The Designer’s Guide to VHDL”, Second Edition, Morgan Kaufmann Publishers, 2004.

Додаток А

(обов’язковий)

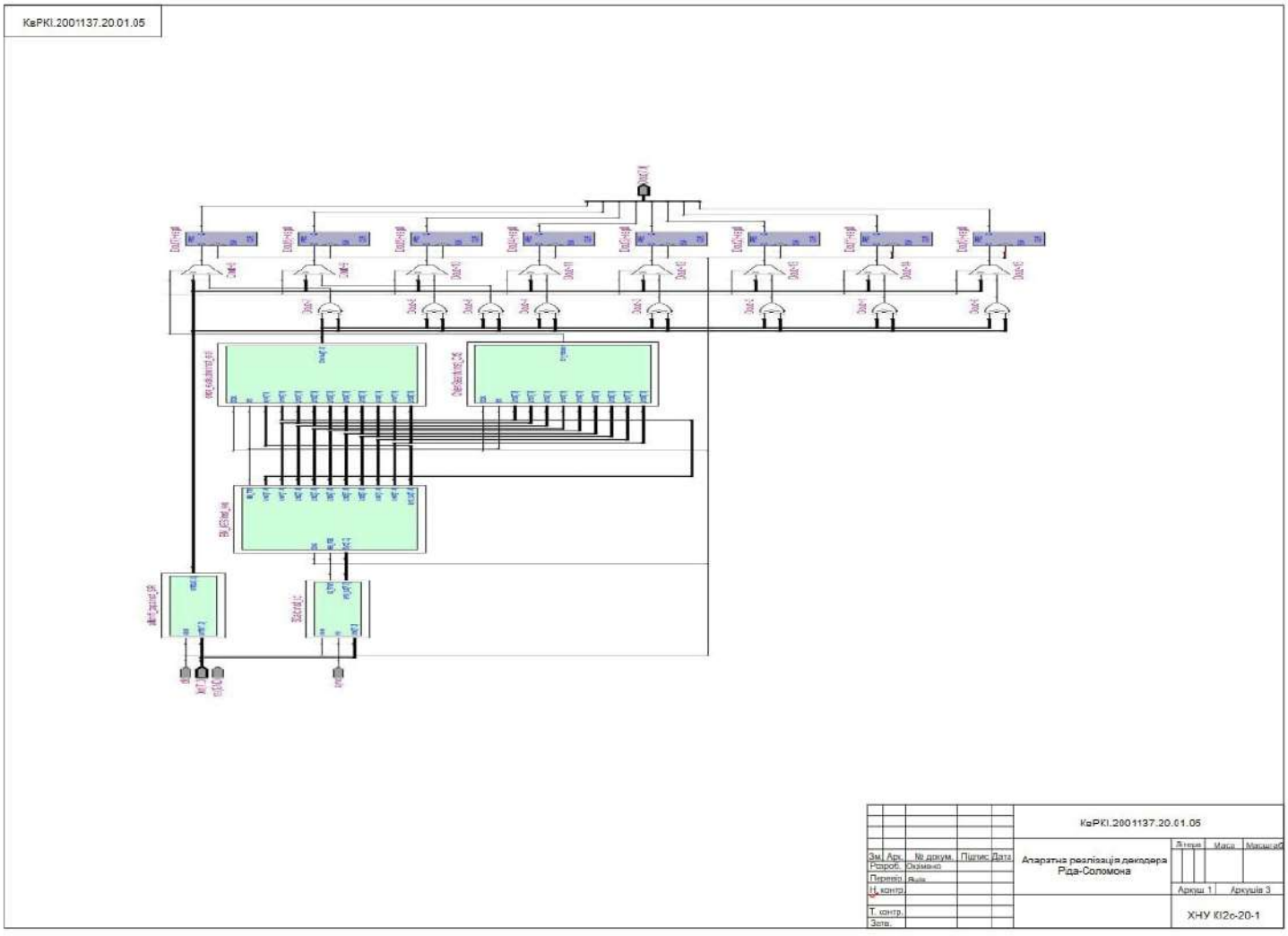
Копія креслення «Апаратна реалізація кодера Ріда-Соломона»



Додаток Б

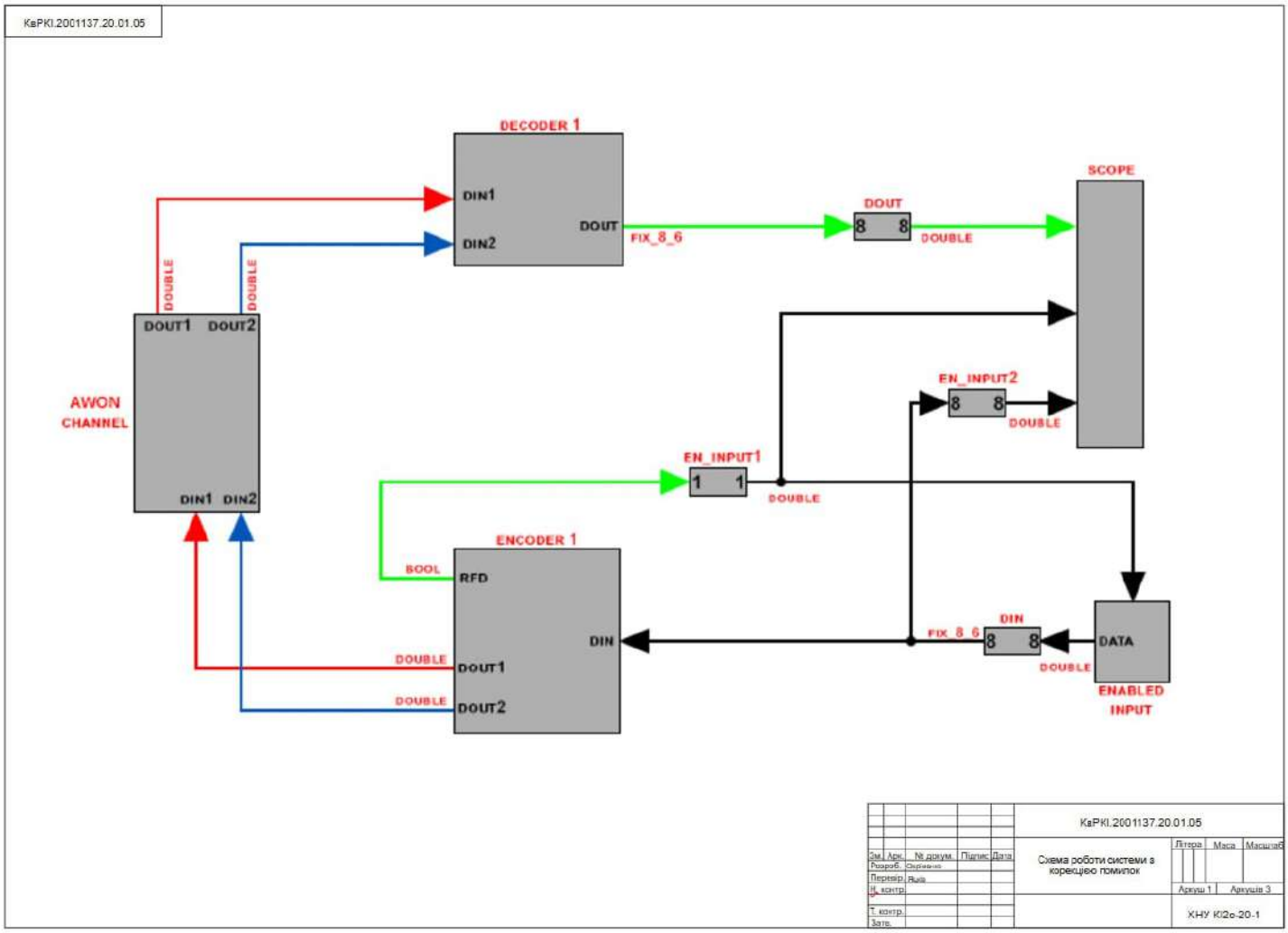
(обов’язковий)

Копія креслення «Апаратна реалізація декодера Ріда-Соломона»



Додаток В

Копія креслення «Функціональна схема системи»



UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КІ

ID перевірки:
1015684954

Дата перевірки:
23.06.2023 19:19:56 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
23.06.2023 19:23:13 EEST

ID користувача:
100005591

Назва документа: **Охріменко_Програмно-технічний модуль завадозахищеного кодування даних на базі FPGA**

Кількість сторінок: **76** Кількість слів: **11967** Кількість символів: **87762** Розмір файлу: **4.61 MB** ID файлу: **1015329044**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

21.2%
Схожість

Найбільша схожість: **18.4%** з джерелом з Бібліотеки (ID файлу: **1011493128**)

20.4% Джерела з Інтернету	192	Сторінка 78
19.6% Джерела з Бібліотеки	141	Сторінка 80

0% Цитат

Цитати	3	Сторінка 81
Посилання	1	Сторінка 81

0%
Вилучень

Немає вилучених джерел

Модифікації

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

Замінені символи	38
Підозріле форматування	16 сторінок

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 9.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 11%**

ID: 117970 Назва: БКР Програмно-технічний модуль завадозахищеного кодування даних на базі FPGA Додано в БД: 2023-06-23 Автора: І. С. Охріменко Керівники: В. В. Яцків Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	74285	642	7021 (9%)	81 (13%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми
106566	Назва: Програмно-технічний модуль завадозахищеного кодування даних на базі FPGA Додано в БД: 2022-06-22 Автора: В.Є Голубович Керівники: В. В. Яцків Консультанти: Опоненти:	6554 (9.0%)	72 (11.0%)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Охріменко Ілля Сергійович

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 ___56__

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є розробка програмно-технічного модулю завадозахищеного кодування даних 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 було проаналізовано та розглянуто технології, які забезпечують надійне кодування, яке стійке до перешкод. Визначено класифікацію методів кодування, описано завадостійкі коди та код Ріда-Соломона. В другому розділі кваліфікаційної роботи були визначені етапи проектування та складові програмно-технічного модуля завадозахищеного кодування даних на базі FPGA, а саме методи представлення коду Ріда-Соломона, кодера та декодеру. В третьому розділі кваліфікаційної роботи виконано апаратну реалізацію коду Ріда-Соломона з програмованою користувачем вентильною матрицею (FPGA). Також створена модель, яка відображає взаємодію кодера і декодера. Проведений синтез і симуляція програмно-технічного модулю.

4. Позитивні сторони роботи: Проведено моделювання розробленої схеми кодера в середовищі Modelsim, MATLAB, яке підтвердило його коректну роботу.

5. Негативні сторони роботи: недостатня увага практичної реалізації.

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 належному науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: заловільно

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Стецюк Михайло Васильович старший викладач кафедри кібербезпеки

"26" червня 2023 р. _____(підпис)

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Говорущенко Т. О.

Охріменка Іллі Сергійовича
ПІБ здобувача вищої освіти

ФІТ, 3 курсу, групи КІ2с-20-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, 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 плагіат оповіщений(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.

2023 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

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: Охріменко Ілля Сергійович

Спеціальність: 123 – Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Яцків Василь Васильович, д.т.н, професор

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, розмішені 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 розмішені 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 чотирьохрозряних 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 21.2% і адресується до 333 першоджерел, 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 КІІС _____ Т. О. Говорущенко