Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення)

Кафедра обчислювальної техніки
(повна назва кафедри (предметної, циклової комісії)

Пояснювальна записка
до магістерської кваліфікаційної роботи
магістр
(освітньо-кваліфікаційний рівень)

на тему Засоби покращення відмовостійкості у напівпровідниковій пам'яті

Виконав: студент 2 курсу, групи 2КІ-18м
напряму підготовки (спеціальності)
123 Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)

Войналович Олександр Юрійович
(прізвище та ініціали)
Керівник Семеренко В.П.
(прізвище та ініціали)
Рецензент Поплавський А.В.
(прізвище та ініціали)

ВНТУ - 2019

Семеренко В. П.,Войналович О. Ю.

ЗАСОБИ ПОКРАЩЕННЯ ВІДМОВОСТІЙКОСТІ У НАПІВПРОВІДНИКОВІЙ ПАМ'ЯТІ

РЕФЕРАТ

Магістерська дипломна робота присвячена дослідженню засобів покращення завадостійкості напівпровідникових пристроїв пам'яті, методів та способів підвищення контролю напівпровідникових пристроїв пам'яті. Дані методи дозволяють виправляти помилки в напівпровідникових пристроях пам'яті за допомогою корегуючих кодів. В роботі проведено аналіз критеріїв коректувальної здатності лінійних блокових кодів. Показано, що для ітеративних детермінованих кодів мінімальна кодова відстань зростає зі збільшенням кількості ітерацій. Запропоновано спосіб підвищення коректувальної здатності на прикладі кода Елаєса.

Ключові слова:ітеративнікоди, коди Елаєса, мінімальна кодова відстань.

Semerenko V.P., Voinalovich O.Y.

MEANS OF IMPROVING THE DEFICIENCY OF Y SEMICONDUCTOR MEMORY

Abstract

The master's thesis is devoted to the research of MEANS of improvement of noise immunity of semiconductor memory devices, methods and methods of Increasing control of semiconductor memory devices. These methods allow for error correction in semiconductors to accommodate memory using corrective codes.The analysis of the criteria for the error-correcting capability of linear block codes is carried out. It is shown that for iterative deterministic codes, the minimum code distance increases with an increase of the number of iterations. A method for improvingof error-correcting capability using the example of the Elias codeis proposed.

Keywords: iterative deterministic codes, Elias code, minimum code distance.

ЗМІСТ

ВСТУП

Актуальність роботи

Напівпровідникові запам'ятовуючі пристрої пам'яті (НЗП) реалізується наодній чи кількох напівпровідникових схемах. При роботі з такою пам'яттю можлива поява помилок. Для того щоб захистити пам'ять від помилок можна застосовувати циклічні коди які прості в реалізації і при невисокій надлишковості мають хороші властивості виявлення спотворень [1]. З метою підвищення коректувальної здатності завадостійких кодів необхідні розробки нових методів їх декодування, зокрема, ітеративних методів декодування.

Метою роботи є підвищення надійності при зберіганні інформації в напівпровідникових пристроях.

Об'єктом дослідження є процеси контролю напівпровідникової пам'яті на основі завадостійкого кодування.

Предмет дослідження – є теоретичні моделі,методи та програмно-апаратні засоби підвищення коректувальної здатності модифікованих кодів Елаєса та кодів Ріда-Соломона на основі принципу ітеративності.

Методи дослідження, що були використані для вирішення поставленої задачі: теорія завадостійкого кодування, теорія інформації, теорія автоматів, теорія алгоритмів, цифрова схемотехніка, об'єктно-орієнтоване програмування.

Наукова новизна одержаних результатів полягає в розвитку детермінованої теорії завадостійкого кодування, яка базується на математичному апараті лінійних послідовнісних схем та ітеративних методів декодування кодів Елаєса і кодів CRC. Запропоновані методи підвищення коректувальної здатності завадостійких кодів, які орієнтовані на особливості контролю напівпровідникової памяті.

Публікації.

1. Семеренко В. П. Войналович О. Ю. Контроль напівпровідникових пристроїв памяті за допомогою циклічних кодівМатеріали конференції «XLVII Науково-технічна конференція підрозділів Вінницького національного технічного університету (2018)», Вінниця, 2018, 2018, с. 965-966.

2. Семеренко В. П. Войналович О. Ю. Оцінка коректувальної здатності ітеративних завадостійких кодів. IV Міжнародна науково-практична конференція "Потенціал сучасної науки" м. Київ, 10-11грудня 2019 р.

1. АНАЛІЗ ВИКОРИСТАННЯ ЗАВАДОСТІЙКИХ КОДІВ В НАПІВПРОВІДНИКОВІЙ ПАМ'ЯТІ

1.1 Класифікація сучасної напівпровідникової пам'яті

Пам'яттю комп'ютера називається сукупність різних пристроїв, призначених для прийому, зберігання та видачі двійкової інформації. Окремий пристрій називається запам'ятовуючим (ЗП) абооперативною пам'яттю (ОП). Термін "запам'ятовуючий пристрій" вживають тоді, коли потрібно підкреслити принцип його побудови: на магнітнихосердях, напівпровідниках тощо. Пам'ять сучасних комп'ютерів класифікують за функціональним призначенням, видом носія інформації, способом організації доступу до інформації.[2]

В наш час ОП реалізується на напівпровідникових ВІС ЗП (рис. 1). У процесі роботи інформація із зовнішньої пам'яті при необхідності переписується в оперативний ЗП.

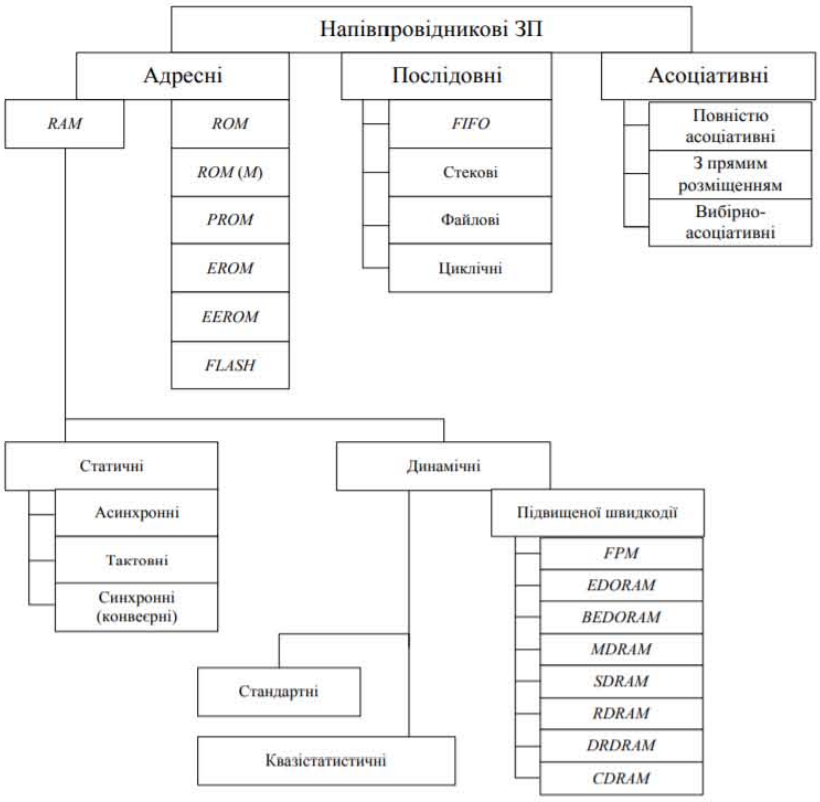


Рисунок 1.1 – Класифікація напівпровідникових ЗП

ОПтакож називається RAM (рис. 1.1) (Random Access Memory – пам'ять з довільним доступом).ОП підрозділяється на статичну пам'ять (Static RAM – SRAM) і динамічну (Dynamic RAM – DRAM). SRAM має швидкий доступ до інформації і не вимагає регенерації, однак коштує трохи дорожче, ніж DRAM. Використовується в основному для кеш–пам'яті нарегістрах.

1.2 Проблеми сучасної напівпровідникової пам’яті

Доосновних проблем напівпровідникової пам’яті належить контроль достовірності інформації, яка в ній зберігається. Розглянемо спочатку види дефектів у цьому типу памяті. Дефекти напівпровідникової пам’яті поділяють на відмови(жорсткі дефекти) та збої (м’які дефекти).

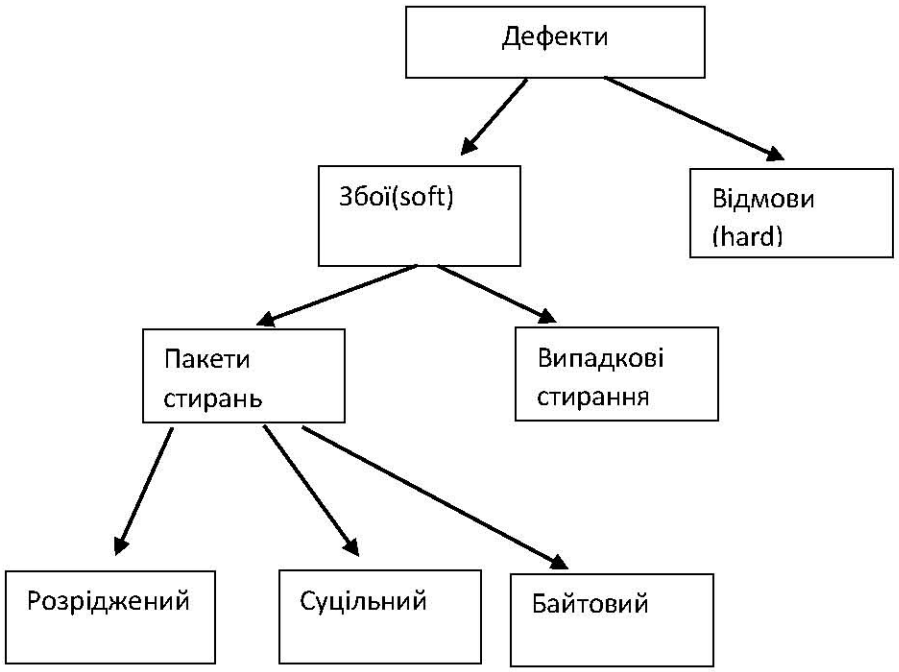


Рисунок 1.3 – Дефекти в НПП.

Причиною відмов можуть бути виробничі несправності чи фізичні спотворення мікросхем. Відмови, зазвичай, наперед відомі (тобто, перед практичним використанням мікросхем) і тому їх можна компенсувати зарахунок резервних комірок пам’яті кристалу НЗП. Збої – короткочасні порушення інформації в комірках пам’яті.

Причиною відмов можуть бути виробничі несправності чи фізичні спотворення мікросхем. Відмови, зазвичай, наперед відомі (тобто, перед практичним використанням мікросхем) і тому їх можна компенсувати за рахунок резервних комірок пам'яті кристалу НЗП. Збої – короткочасні порушення інформації в комірках пам'яті. Динамічні НЗП чутливі до опромінення світлом і радіацією. Звичайна цифрова техніка перестає працювати після 5000 рад. Це найбільша проблема космічної електроніки – важкі заряджені частинки мають таку високу енергію, що «пробивають» мікросхему наскрізь, і залишають за собою «шлейф» заряду, який може привести до програмної помилку (0 стати 1 або навпаки – single–event upset, SEU або до тиристорному замикання (single–event latchup, SEL)(таблиця 1.4).

Таблиця 1.1 Вплив космічного випромінювання на НПП

Тип ефекту	Ефект	Позначення	Опис
Катастрофічний (HARD)	Вторичний пробій	SEB (Single-Event Burnout)	Вторичний пробій p-n переходу який призводить до його руйнування
	"Прокол" діелектрика	SEGR (Single-Event Gate Rupture)	Пробій затвору діелектрика вздовж треку ядерної частки
	Одиночне пошкодження біта	SEDU	"Залипання"пікселя або біта в стані 0 або 1
	"Прокол" діелектрика	SEDR (Single-Event Dielectric Rupture)	Пробій діелектрика під дією іонізуючих частинок
	Вторинний пробій транзистора	SB (Second Breakdown)	Лавиноподібне збільшення струму на локальній ділянці стоку транзистора
	Тиристорний ефект	SEL (Single-Event Latchup)	Ввімкнення чотирьохслойної p-n-p-n структури яке призводить до різкого збільшення струму в ланцюгу живлення
	Тиристорний ефект	SES (Single-Event Snapback)	Аналогічно SEL але в n-переході
Функціональ-ний (SOFT)	Одиночне функціональне переривання	SEFI (Single-Event Functional Interrupt)	Інверсія логічного стану комірки пам'яті яке призводить до порушення виконання програми
	Одиничний збій	SEU	Інверсія логічного стану комірки пам'яті або тригера
Залишковий (SOFT)	Багаторазові збої	Multi-Cell and Multi-Bit Upsets (MCU and MBU)	Інверсія логічного стану декількох сусідніх комірок пам'яті або тригера
	"Голка"	DSET	Короткочасний імпульс на виході елементу цифрової ІС
Коротко-часний		ASET	Короткочасний імпульс на виході елементу аналогової ІС

Як приклад такої аварії можна навести ситуацію з «Фобос-Грунтом», коли через збої в пам'яті космічний аппарат не зміг виконати свою задачу.

Щоб домогтися досягнутої на даний момент величезної щільності упаковки елементів, в мікросхемахреалізовано багато механізмів компенсації збоїв. В DRAM, наприклад, завдяки постійномуоновленню зберігаються безперервно витікаючі дані, коди корекції помилок дозволяють відновлювати значення помилково переданих бітів, ау флеш-пам'яті є рівень трансляції (Flash Translation Layer, FTL) який керує даними, що зберігаються на флеш - пам'яті і здійснює зв'язок з комп'ютером або електронним пристроєм. Контролери флеш-пам'яті можуть бути призначені для роботи з SD картами, CompactFlash картами, або іншими аналогічними засобами масової інформації для використання в цифрових камерах, кишенькових комп'ютерів, мобільних телефонів тощо. USB флеш-накопичувачі використовують контролери флеш-пам'яті, призначені для зв'язку з особистим комп'ютери через порт USBв низькомуробочому циклі. Контролери флеш-пам'яті також можуть бути розроблені для середовищ з більш високим робочим циклом, таких як твердотільнінакопичувачі SSD(Solid-state drive), що використовуються як сховища даних для переносних комп'ютерних систем.[3]

Як в DRAM, так і у флеш-пам'яті використовується заряд - електрони заносяться в комірці для зберігання даних, а зчитування відбувається при подачі напруги. Для переміщення електронів досить низьких енергій, щоодночасно і добре, і погано - висока мобільність носіїв дозволяє швидко зчитувати і записувати біти з малими витратами енергії, але робить складним довгострокове утримання інформації, оскільки електрони можуть легко змінити свій логічний стан в комірці. У флеш-пам'яті це неможливо зарахунок товстого шаруоксидного ізолятора, який утримує заряд довше, однак для програмування комірок потрібно «пробивати» цей шар 0 можуть утримувати все менший заряд - відповідно, на заряд одиночного електрона доводиться все більша частказагального заряду комірки, і навіть якщо невелика кількість електронів покине її або проникне в неї, збережене значення зіпсується. У міру зменшення розміру комірки проблему посилює виробничий брак - неоднорідність комірок. Відхилення зачіпають геометрію і склад комірки, в результаті деякі з них

можуть утримувати набагато менший в порівнянні з нормою заряд абооксид в нихруйнується набагатораніше, ніж в інших.[4]

Історично індустрія пам'яті вирішувала ці проблеми шляхом використання нових матеріалів, виробничих процесів чи геометрій комірки, але все частіше для підвищення щільності деякими важливими характеристиками доводилося жертвувати. Наприклад, замість зберігання одного біта в кожній однорівневійкомірці (single-level cell, SLC) перейшли на багаторівневі комірки (multi-level cell, MLC), що зберігають кілька біт зарахунок поділу діапазону напруг, використовуваних для читання і запису. В результаті підвищується щільність пам'яті, але прискорюється знос. Ще один приклад - регенерація DRAM: якщо її не буде, DRAM не здатна зберігати записані значення, тому комірки доводиться періодично зчитувати і перезаписувати, щоб підтримувати необхідний рівень заряду. Однак безперервне оновлення призводить до погіршення швидкодії і вимагає додаткової витрати енергії.

При відсутності ефективного вирішення для утримання електронів в більш компактних комірках DRAM непродуктивні витрати на компенсацію помилок пам'яті можуть стати недозволено високими. Наприклад, підвищена частотаоновлення може віднімати так багато тактів пам'яті, що нареальні операції зчитування і запису їх просто не залишиться. А вдосконалення механізмів корекції помилок - ще одне стандартне рішення для серверної DRAM - підвищує складність конструкції чіпа і вимагає додаткового простору для зберігання кодів корекції, внаслідок чого надбавка в щільності зменшується і зростає ціна. Крім того, корекція помилок в робочій пам'яті може збільшувати затримку - вимоги до швидкодії робочої пам'яті набагато вище, ніж до довготривалої, арішення для підвищення надійності, наприклад додавання трансляційного шару, не підходять для робочої пам'яті через велику затримку, яку вони створюють.

Ще одна складність, яка стосується робочої пам'яті, особливо серверної, полягає в тому, щоб виділити достатню пропускну здатність пам'яті процесорним ядрам. Вонаросте не так швидко, як число ядер в процесорах, і

необхідність ділити між ними вузьку смугу пропускання призводить до зниження продуктивності. Для подолання згаданих труднощів розробляються альтернативні види пам'яті (Табл. 1.1).

Виробники DRAM і флеш-пам'яті йдуть по шляху вертикальногорозміщення пам'яті в кілька ярусів таким чином, збільшується щільність комірок наодиницю площі, знижується вартість на біт і підвищується швидкодія. Атомні (резистивні) технології пам'яті засновані на зміні фізичної конфігурації атомів або вакансій в комірці, зарахунок чого змінюється її опір. Магнітні (резистивні) схеми пам'яті засновані на зміні орієнтації феромагнітного матеріалу з плаваючою полярністю по відношенню до матеріалу з фіксованою, що теж впливає наопір комірки. Вертикальна флеш-пам'ять, найімовірніше, буде застосовуватися в якості довгострокової, оскільки процес її виробництва сьогодні найближче до використовуваного для випуску флеш-пам'яті.

Збільшена завдяки переходу в третій вимір щільність дозволить виробникам спочатку задіяти більші комірки, внаслідок чого тимчасово зменшаться проблеми з надійністю. Якийсь час підвищення щільності може тривати зарахунок збільшення числа ярусів в комірці, але в кінцевому підсумку буде досягнутообмеження, коли витрати на виробництво стануть непропорційно високими.

Після цієї межі виробникам доведеться повернутися до зменшення розмірів комірки з усіма супутніми складнощами. Потім, ймовірно, відбудеться перехід на атомні технології, якщо вартість великомасштабного виробництва такої пам'яті вдасться знизити до рівнів вартості нинішньої флеш-пам'яті.

Прогнозувати розвиток рішень в області робочої пам'яті складніше через більш строгих вимог до її швидкодії. Ймовірно, в короткостроковій перспективі в серверах почнуть застосовувати НМС-масиви Консоль апаратного забезпечення (НМС) - це апаратна система, за допомогою якої можна налаштовувати і контролювати одну або кілька керованих систем. НМС з'єднується з керованими системами, збирає в них інформацію і відправляє її в

службу технічної підтримки для аналізу, що забезпечують більш високу продуктивність, однак, через меншу ємності, технологія HMC, швидше за все, не зможе витіснити модулі DRAM, а буде застосовуватися разом з ними. Що стосується більш віддаленої перспективи, у кожної з технологій-кандидатів є недоліки, які поки не подолані. У магнітної пам'яті присутні всі необхідні характеристики, однак високої щільності такої пам'яті досягти важко, і поки вона не порівнянна з DRAM. Атомні технології пам'яті, схоже, мають хороші перспективи підвищення щільності, але, так як їх принцип збереження даних заснований на зміні атомної конфігурації комірки, час і енергія, які потрібні на запис даних в такі комірки, можуть виявитися неприпустимо високими.

Крім того, під час запису деформується матеріал комірки, це викликає деградацію і призводить до передчасного зносу, внаслідок якого комірка «застряє» на низькому або високому опорі, що може погіршити довговічність системи в цілому. Механізми, розроблені для DRAM, наприклад коди корекції помилок, ще більше посилюють проблему, зменшуючи термін служби атомної пам'яті.

Підвищення щільності пам'яті зі збереженням стабільності стає справою настільки складною і витратною, що скоро може знадобитися радикальний відхід від традиційних технологій, а це означає серйозні наслідки для розробників ПЗ (Програмне забезпечення) і системних інженерів. В процесі того як комірки пам'яті стають все меншими, їх стабільність погіршується, і навіть незначний вплив на заряд здатний змінити стан комірки.

Таблиця 1.2 - Альтернативні технології виготовлення пам’яті

Технологія	Тип	Механізм зберігання

Hybrid Memory Cube	Електронна	Декілька “поверхів”DRAM розташовані поверх слою логіки високої швидкодії. Загальна ємність пам’яті приноситься в жертву швидкодії.
Вертикальна флеш-пам’ять	Електронна	Заряди захвачуються в пастку транзистором з плавуючим затвором. Комірки орієнтовані вертикально зарахунок чого значно збільшується щільність і є можливість збільшити габарити комірок.
Phase-charge memory	Атомна	Матеріал комірки може кристалізуватися і переходити в аморфний вид під дією контрольованого нагріву таохолодження.
Мемристори	Атомна	В мемристорах використовується тонка плівка діоксиду титану яка під дією струму змінює опір.
Conductive-bridging RAM,CBRAM	Атомна	При подачі струму іони металу в комірці переставляються надаючи провідність непровідниковому матеріалу.

Середможливих способів поліпшення стабільності пропонуються дорогі технології виготовлення: вертикальна «штабелювання» комірок, альтернативні методи зберігання, перехід на гетерогенні ієрархії пам'яті, а також більш

досконалі технології корекції, що дозволяють надійно працювати з нестабільною пам'яттю. Пам’ять з блоком перевірки помилок

ECC(Error correcting code)-пам'ять - тип комп'ютерної пам'яті, якаавтоматичнорозпізнає і виправляє спонтанно виниклі зміни (помилки) бітів пам'яті. Пам'ять, яка не підтримує корекцію помилок, позначається як non-ECC (Рис. 1.2).

Як правило, пам'ять з корекцією помилок може виправляти зміни одного біта в одному машинному слові. Це означає, що при читанні одного машинного слова з пам'яті буде прочитано те ж значення, що було до цього записано, навіть якщо в проміжку між записом і читанням один біт був випадково змінений (наприклад, під дією космічних променів). Звичайна пам'ять, як правило, не здатна визначити, чи була помилка, хоча деякі види пам'яті з контролем парності здатні визначити, що сталася помилка, але не здатні її виправити.[5]

Пам'ять з корекцією помилок використовується в більшості комп'ютерів, для яких важлива безперебійнаробота, в тому числі в більшості серверів. Для роботи пам'яті в режимі корекції помилок потрібна підтримка з боку контролераоперативної пам'яті, який може бути складовою частиною чіпсетаабо вбудовуватися в систему на кристалі, єдину з обчислювальними ядрами.

На практиці широко застосовується DDR SDRAM (Double Data Rate Synchronous Dynamic Random Access Memory).ECC-пам'ять для серверів з кодом класу SECDED (виправлення одиночних і виявлення подвійних помилок). На модулях пам'яті на кожні 8 мікросхем додається ще поодній мікросхемі, яка зберігає ECC-коди розміром 8 біт на кожні 64 бітаосновний пам'яті.ECC забезпечують кодування каналу у вигляді контрольних бітів парності для захисту даних користувачів. Потім ,коли слово зчитаэться з памяті ,контрольні біти можна перерахувати та порівняти проти раніше написаних бітів,. Будь-які неспівпадіння призведуть до розбіжностей між двома наборами контрольних бітів, також відомих як синдром.

Синдром може бути використаний для виправлення або виявлення помилок залежно від складності коду.Цей процес показаний на малюнку 1. Слідом за генерацією шаблону певного синдрому, його можна розшифрувати в місці помилки. Цей вектор розташування помилок може потім застосуватись проти первісно отриманого кодового слова для виправлення будь-яких помилок. Для визначення того, чи є омилка, використовується додаткова комбінаційна логіка виправлення або виявлення. типи помилок, які можна виправити або виявити, залежать від кодування контрольних бітів, і витонченість цих типів помилок має прямий вплив на накладні витрати схем.Виправлення помилок, що використовуються в ранніх системах пам'яті комп'ютера, було розроблено за допомогою клас кодів SEC-DED . Код SEC-DED здатний виправити одну однобітну помилку та виявити дві однобітні помилки в кодовому слові. Можливості виявлення подвійних помилок служать для захисту від втрати даних. У 1970 році Сяо покращили ефективність цих кодів шляхом введення непарних кодів ваги стовпців до SEC-DED .Хоча коди Hamming та Hsiao SEC-DED здатні виправити одну помилку і виявивши всі можливі подвійні помилки, стає очевидним, що вони не зможуть забезпечити адекватний захист від MBU(Multi-BitUpset) в сучасних масштабах SRAM .Щоб вирішити цю проблему, потрібні коди, що використовують більш високий ступінь контрольних бітів та / або логіку кодера декодера. Ефективний підхід до пом'якшення MBU - через багатобітні коди виправлення помилок. Коди Bose Chaudhuri Hocquenghem (BCH) , коди ReedSolomon (RS) , коди Golay , перевірка паритету евклідової геометрії низької щільності (EG-LDPC) та двовимірні коди помилок були розроблені для роботи декількома бітовими помилками в пам'яті. Завдяки високій складності декодування та надмірності, проте реалізація цих кодів не є популярною Питання полягає в тому, що, хоча

існують більш потужні ECC, вони отримують занадто високий штраф у порівнянні з Коди SEC-DED для щоденних застосувань . Наприклад, коди BCH для подвійного виправлення помилок, що визначають потрійну помилку (DEC-TED), вимагають вдвічі більше контрольних бітів і приблизно на 60% збільшення затримки порівняно з кодом SEC-DED. Тому питання розширення можливостей захисту кешу ECC для використання в мікропроцесорах нового покоління є все ще відкритим.

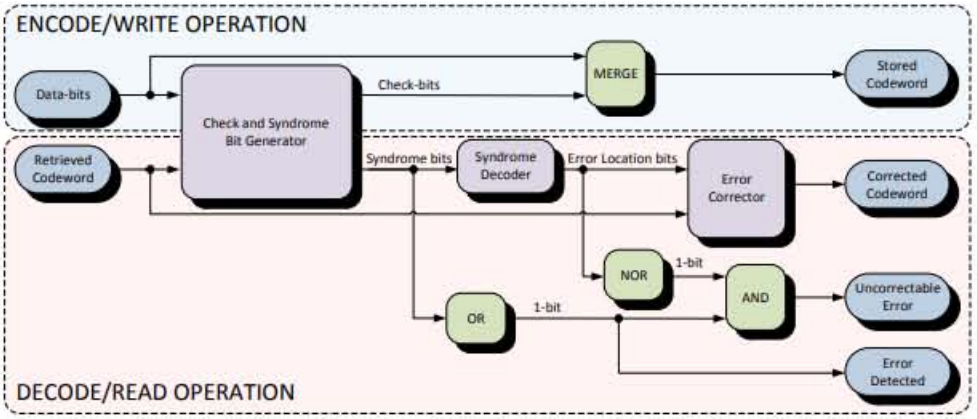


Рисунок 1.2 – Діаграма ECC

ECC-захисту даних можуть застосовуватися для вбудованої в мікропроцесори пам'яті: кеш-пам'яті, регістровогофайлу. Іноді контроль також додають в обчислювальні схеми.

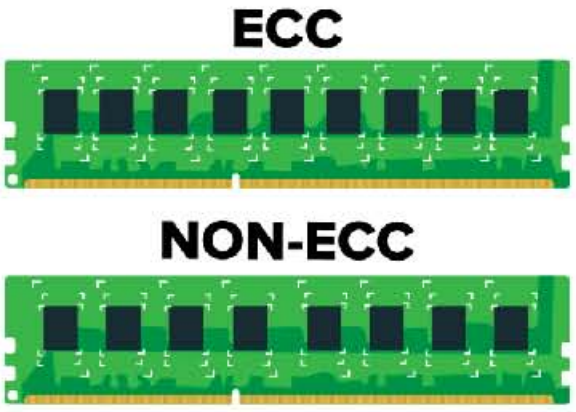


Рисунок 1.3 – ECC та NON-ECC пам’ять

Однією із складових частин ініціативи X-Architecture корпорації IBM стала технологія виправлення помилок Chipkill. Вона забезпечує захист серверів від відмов окремих мікросхем і багаторозрядних помилок в модулях пам'яті. Ця технологія, перенесена IBM з великих систем, істотно скорочує середній час простою серверів і забезпечує більш надійну платформу для клієнт-серверних обчислень на базі мікропроцесорів Intel. Вона покликана підвищити надійність систем, доступність і зручність в експлуатації - ключові характеристики серверів масштабу підприємства, які обслуговують критично важливі програми. Архітектура Chipkill дозволяє системі безболісно сприймати помилки, які в звичайних умовах призводять до аварійних збоїв, тим самим забезпечуючи збереження даних і високу доступність системи.[6]

У системах високої доступності, таких як сервери масштабу підприємства IBM, проблема багаторозрядних помилок пам'яті DRAM відсутня завдяки особливій архітектурі оперативної пам'яті. Підсистема пам'яті сконструйована так, що відмова окремої мікросхеми, незалежно від її розрядності, не зачепить більше одного розряду в будь-якому з декількох слів ECC. Наприклад, в 4-розрядної мікросхемі DRAM окремі біти з усієї четвірки потрапляють в різні слова ECC тобто в різні адресні простори пам'яті. Тому навіть у разі повної відмови мікросхеми кількість помилкових розрядів в словах ECC не перевищить одиницю, а таку помилку механізм ECC усуває автоматично. Дана архітектура забезпечує відмовостійкість всієї підсистеми пам'яті.

Продумана конструкція мейнфреймів захищає їх від збоїв мікросхем пам'яті. У кожному модулі пам'яті розрядність мікросхем дорівнює числу розрядів, захищених механізмом ECC. Нинішні сервери на базі мікропроцесорів Intel такого механізму не підтримують - адже ринок вимагає дешевої пам'яті, змушуючи проектувальників створювати дуже щільні мікросхеми пам'яті, які підтримують галузевий стандарт (іншими словами, виключно ECC).

Chipkill - це механізм, що дозволяє пам'яті протистояти багаторозрядним помилок наокремих мікросхемах DRAM, в тому числі збою всіхрозрядів даних. У механізмі Chipkill є дваосновні методи виправлення помилок, причому вони можуть застосовуватися спільно. Ці методи базуються на певному наборі мікросхем і особливої апаратної архітектурі системи - їх підтримку неможливо забезпечити простим оновленням ПЗ.

У першому методі кожен біт даних модуля пам'яті розміщується в окремому "слові ECC". (Слово ECC - це набіррозрядів даних і контрольнихрозрядів, в якому виявлення та виправлення помилок забезпечується алгоритмом ECC.) Припустимо, щорозрядність системи пам'яті становить 32 байт (або 256 розрядів). Біти ECC додаються так, щоб загальна ширина блоку (і контрольні, і біти даних) складала 288 розрядів. Чотири слова ECC, кожне з яких складається з 64 розрядів даних і 8 контрольнихрозрядів ECC, підтримують механізм SEC / DED. Ці чотири слова ECC розподіляються по DRAM-модулів. Наприклад, якщо DIMM містить модулі x4 DRAM, чотири біта кожного пристрою розподіляються зарізними словами ECC (рис. 2). Збій всіх чотирьох бітів - це всього лише чотири поодинокі помилки в чотирьох словах ECC, і вони усуваються автоматично. У цьому прикладі механізм Chipkill підтримується тільки на DIMM-модулях, що складаються з мікросхем x4 DRAM.

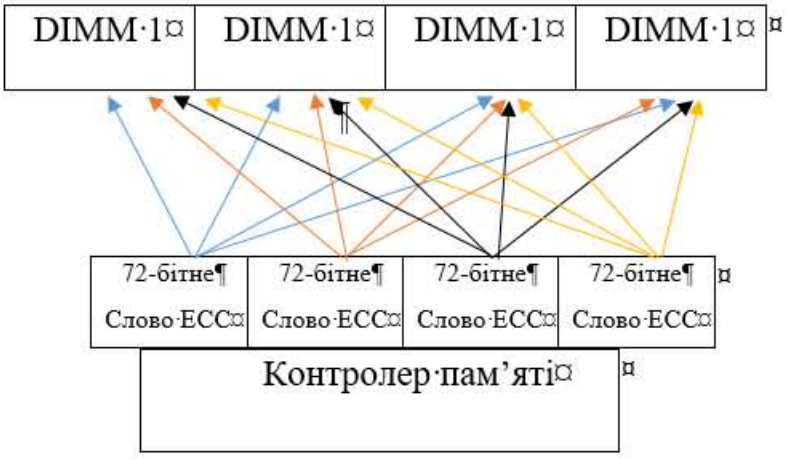


Рисунок 1.4 – Механізм Chipkill

Другий метод полягає в наданні механізму ECC більшого числарозрядів для зберігання контрольних кодів, щоб забезпечити виправлення не одного, а декількохрозрядів. При цьому використовуються відповідні математичні алгоритми усунення багаторозрядних помилок при певній кількості контрольних бітів ECC і бітів даних. Наприклад, 144-розрядне слово ECC, що складається з 128 розрядів даних і 16 бітів ECC, дозволяє виправляти помилки, які охоплюють до 4 розрядів даних. (Для виправлення збою чотирьох біт необхідно, щоб вони були суміжними, а не розташовувалися випадково.) Співвідношення розрядів ECC і розрядів даних таке ж, як і в попередньому прикладі (16/128 і 8/64), проте довший слово ECC дозволяє застосувати більш ефективний алгоритм виявлення та виправлення помилок.

Спільне використання цих двох методів забезпечує корекцію за механізмом Chipkill на DIMM-модулях з мікросхемами x8 DRAM. Два 144-розрядних слова ECC розподіляються так, щоб на кожному DRAM в першому і другому словах ECC виправлялися по 4 розряду. Цей метод забезпечує підтримку механізму Chipkill при використанні DIMM-модулів, що складаються як з мікросхем x4 DRAM, так і з x8 DRAM.

На серверах Netfinity (на зміну яким прийшли машини eServer xSeries) інженери IBM вирішили цю проблему, розмістивши надлишковий масив недорогих мікросхем DRAM (Redundant Array of Inexpensive DRAM, RAID) безпосередньо на DIMM-модулях. При кожному доступі до пам'яті RAID-мікросхемаобчислює контрольний код ECC для всього набору мікросхем і зберігає результат в резервної пам'яті на захищається DIMM-модулі. Уразі відмови однієї мікросхеми DIMM-модуля збережена на RAID-масиві інформація застосовується для відновлення втрачених даних, забезпечуючи безперебійнуроботу всього сервера Netfinity. Така RAID-технологія нагадує RAID-механізми, що застосовуються для захисту даних в дискових масивах, і називається Chipkill DRAM.

Фахівці компанії IBM розробили замовну спеціалізовану мікросхему для Chipkill, яка називається ECC ASIC (Application-Specific Integrated Circuit) і виконує виправлення помилок без уповільнення доступу до високопродуктивної EDO-пам'яті з часом доступу 50 нс. Таким чином, при установці DIMM-модулів IBM Chipkill на сервери Netfinity, обладнані процесором Intel Xeon, модифікація плат пам'яті не потрібна, а нові модулі не викликають зниження продуктивності.

Результати досліджень показали, що на серверах, обладнаних 32 Мбайт пам'яті з перевіркою парності, частота збоїв пам'яті за 3 роки склала 7 відмов на 100 серверів. На серверах, обладнаних 1 Гбайт пам'яті з підтримкою ECC, частота збоїв за 3 роки дорівнює 9 відмов на 100 серверів. Аось на серверах, обладнаних 4 Гбайт пам'яті з підтримкою Chipkill, за 3 роки число збоїв не перевищило 6 відмов на 10 тис. Серверів. Крім того, виявилося, що частота відмов через помилки пам'яті на серверах з 1 Гбайт ECC-пам'яті вище, ніж на серверах з 32 Мбайт пам'яті з перевіркою парності. Причина тут в тому, що при тому величезному числі DRAM-мікросхем ємністю 64 Мбіт, яке потрібно, щоб скласти 1 Гбайт пам'яті, ймовірність відмови окремих мікросхем істотно вище, ніж вірогідність виникнення одиничних помилок парності в 32 Мбайт пам'яті з перевіркою парності. Технологія IBM Chipkill Memory дозволила знизити числовідмов серверів Netfinity через збої підсистеми пам'яті до неймовірно низькогорівня.

1.3 Аналіз класичних завадостійких кодів для діагностики напівпровідникової пам’яті

Суть контролю НПП завадостійкими кодами така ж, як і в системах зв’язку: формування r контрольних символів і їх запис разом з k інформаційними символами, а після зчитування додаткові символи дозволяють виявити в n–розрядному кодовому слові τ стирань в залежності від вибраного ступеня надлишковості. Тому завадостійкі коди, які були створені для систем зв’язку, можуть з успіхом застосовуватись і для контролю НПП.

Поряд з очевидною аналогією процесів контролю в обох випадках доцільно також використати особливості структурної організації пам'яті, що дозволить підвищити ефективність використання кодів.

Наприклад, поширеною структурою оперативної пам'яті є побітоваорганізація, коли всі біти кодового слова зберігаються в окремих кристалах. В цьому випадку дефект навіть багатьох комірок пам'яті одного кристалу буде спотворювати не більше одного біту в різних кодових словах. Досягти такого ефекту в системах зв'язкуможна лише зарахунок операції перемежування кодових слів. Для виявлення та виправлення такого типу стирань найбільш придатними будуть коди типу SEC–DED (виправлення поодинокої стирання або виявлення подвійної стирання).

В сучасних НПП великої ємності використовується також і b–байтова ($b = 2 \div 8,16,32$) організація, коли в одному кристалі зберігається b послідовних біт. Дефекти одного кристала вже можуть викликати пакети стирань довжиною b. В цьому випадку вже необхідно використовувати коди типу SbEC–DbED (виправленняодного b–бітового байту стирань або виявлення подвійного b–бітового байту стирань). Як вже відзначалося, байти можуть мати різну структуру: розріджену (sparse byte errors) або суцільну (Spotty byte error).

Суцільний пакет стирань довжиною τ біт може бути частиною розрідженого пакету стирань довжиною b. Для таких конфігурацій стирань відповідають коди типу $S\tau/bEC-D\tau/bED$ (виправлення τ сусідніх біт в одному b–бітовому байті стирань або виявлення τ сусідніх біт в двох b–бітових байтах стирань).

Однією з фундаментальних проблем в теорії завадостійкого кодування є правильна інтерпретація результатів декодування, оскільки різні типи стирань можуть давати однаковий ненульовий синдром. Лише з математичних позицій цю задачу неможливоrозв'язати, необхідно враховувати додаткові фактори, зокрема тип каналу, по якому передаються дані.

Практика та експериментальні дослідження, довели, м'які дефекти в НПП найбільш адекватноописуються суцільними пакетами стирань НПП з бітовою організацією або суцільними байтовими пакетами стирань для НПП з байтовою організацією. Тому надалі будеморозглядати завадостійкі коди саме для таких типів стирань.

В системах зв'язку зазначені типи стирань окремо не розглядались, тому і не існувало для них спеціалізованих кодів. Останнім часом пропонують використання для цієї мети відомих завадостійких кодів, які не завжди враховують особливості оперативної пам'яті.

Зокрема, швидкодіючі кристали НПП мають фіксований і дуже малий час циклу запису–зчитування. З цієї причини непридатними є турбо–коди і коди LDPC, які використовують ітеративне багатоциклове декодування. Такого недоліку позбавлені циклічні коди, які дозволяють здійснити виявлення та виправлення стирань протягом однієї–двох ітерацій.

Різноманітні підкласи циклічних кодів вже давно використовуються для контролю оперативної памяті. Наприклад, циклічні коди Хемінга використовуються як коди SEC–DED, а коди БЧХ – як коди DEC–TED (подвійні стирання виправляються, а потрійні – виявляються). Для виявлення тавиправлення пакетів стирань з успіхом використовуються коди Ріда–Соломона. [7]

Модульні коди необхідні для корекції стирань в пристроях з модульною структурою коли пам'ять будується на пристроях з багатозарядною організацією. При цьому всі повідомлення довжиною n діляться на N груп символів по b символів в кожній групі.

Вихід з ладуодного пристрою пам'яті пристрою спотворює лише модуль кодового слова, кордони якого відомі на відмінну від пакету стирань.

До класичних завадостійких кодів для діагностування стирань пам'яті можна віднести коди Ріда–Соломона, БЧХ, циклічні коди Хеммінга, Файра.

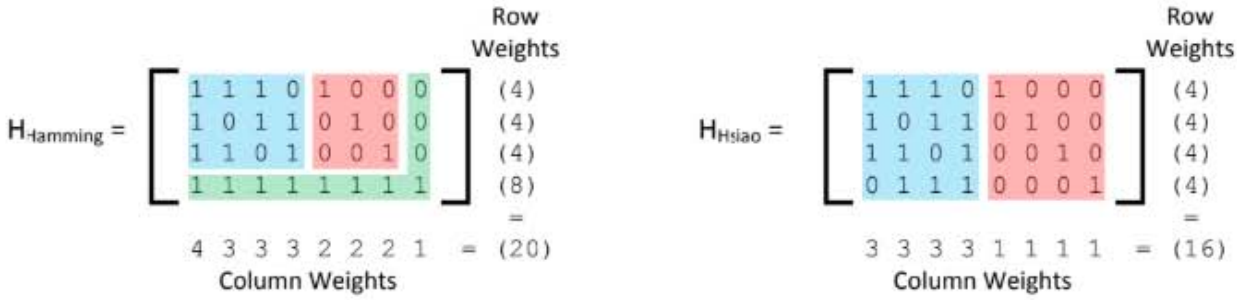
Ріда–Соломона – недвійкові циклічні коди, що дозволяють виправляти стирання в блоках даних. Елементами кодового вектора є не біти, а групи бітів (блоки).

Важливою і часто використовуваною підмножиною кодів БЧХ є коди Ріда–Соломона. Це такі коди БЧХ, в яких мультиплікативний порядок алфавіту символів кодового слова ділиться на довжину коду. Таким чином, m =1 і поле символів GF(q) збігається з полем локаторів стирань ( ) m GF q. Звичайно ми будемо вибирати α примітивним, тоді n = q −1 = q −1 m. Мінімальний багаточлен над GF(q) елемента β , взятого з того самого поля, дорівнює fβ (. x) = x – β Оскільки поле символів і поле локаторів стирань збігаються, всі мінімальні багаточлени лінійні. [6]

Коди БЧХ(Боуза — Чоудхури —Хоквингема) – в теорії кодування це широкий клас циклічнихкодів, що застосовуються для захисту інформації від стирань. Відрізняється можливістю побудови коду із заздалегідь визначеними коригувальними властивостями, а саме, мінімальною кодовою відстанню.

Принцип побудови коду БЧХ Як породжувальний поліном для коду БЧХ вибирається поліном, що являє собою добуток двох поліномів: B(x) = M1(x)·M3(x). Ці поліноми вибираються таким чином, що синдром, отриманий для M1(x) S_1 і синдром, отриманий для M3(x) S3 у випадку виникнення одиночної стирання зв'язані між собою співвідношеннями: S_3i = S_1i 3. У випадку подвійної стирання, що займає позиції i і j: S_1 = S_1i + S_1j S_3 = S_3i + S_3j S_3i = S_1i 3 S_3j = S_1j. Таким чином, одержали системурівнянь із двома невідомими: S_1 = S_1i + S_1j S3 = S_1i 3 + S_1j 3 вирішивши яку, можна визначити номера позицій стирань i і j, аотже, виправити стирання.[7]

Коди Hsiao, також називають кодами з непарною вагою, подібні модифікованим кодам Хемніга, забезпечують однаковий рівень ефективності кодування в порівнянні з кодом Хемінга з точки зору використання ресурсів, але забезпечують кращу швидкодію, та надійніші в декодуванні. Замість того щоб покладатися на перший рядок H-матриці коди Hsiao використовують стовпчики Хемінга для пошуку подвійних стирань.[16]



(a) Hamming (b) Hsiao

Рис. 1.5 - Порівняння матриць кодів Хемінга та Hsiao

Матриця Hsiao має меншу загальну вагу (16) в порівнянні з матрицею Хемінга (20) (рис. 2.1). Це призводить до зменшення загальної кількості XOR (Hsiao = 12, Хемінг = 16) і нижча кількість логічних рівней в XOR дереві декодеру (Hsiao = $[\log_2(4)]=2$, Хемінг = $[\log_2(8)]=3$). Це дозволяє зменшити затримку при реалізації Hsiao кодів в порівнянні з кодами Хемінга.[17]

При наявності потрійної стирання ймовірність помилкової корекції модифікованим кодом Хемінга становить 75,9%, в той час як у кода Hsiao 56,3%, при наявності чотирикратної стирання можливість знайти її у кода Хемінга 98,9%, в кода Hsiao 99,2 (Таблиця 2.6).

Таблиця 1.3 Порівняльна характеристика кодів для інформаційного слова з 64 біт

Вид коду	Код Хемінга	Модифікований код Хемінга	Код Hsiao
Інформаційне слово, біт	64	64	64
Кодове слово, біт	71	72	72
Коефіцієнт збільшення площі яку займає комірка пам'яті	1.11	1.13	1.13
Площа кодеру-декодеру	1	1.33	1.32

Час критичного шляху в кодері	1	1.32	0.96
Час критичного шляху в декодері	1	1.49	1.14

Якщо існує низька ймовірність появи стирань з кратністю 2 доцільно використовувати коди Хемінга, якщо ж ймовірність висока тоді краще використати коді Hsiao.

Спочатку практика обмежувалася тільки перевіркою на парність, яка виявляла будь-яку помилку непарної кратності. Найчастіше в З.П. виникають поодинокі стирання тому стали необхідні коди, що виправляють тільки одиночні помилки.

Двійковий лінійний блоковий (n,k)-код представляє собою k-мірний підпростір n-мірного простору двійкових векторів. Кодове слово (вектор) довжини nзберігає kінформаційних та $r=n-k$перевірочних символів. Для опису коду слугує перевірочна $r\times n$-матриця H.

Слово ділиться на байти довжини b. Одно-байтова помилка може містити будь-яке число стирань, обмежених одним байтом. З іншого боку, пакет стирань довжини b - будь-яке число стирань, обмежених b сусідніми позиціями, код здатний до виявлення (або виправлення) пакетів довжини b, а також здатний до виявлення (або виправлення) всіх пакетів менше, ніж b.[15]

Декодер працює наступним чином:

Якщо синдром дорівнює нулю то стирань немає і зчитане кодове слово залишається без змін. Якщо синдром не дорівнює нулю, подається сигнал про знаходження стирання і дешифратор видає вектор-помилку, якщо її можна виправити то помилка зчитується з кодового слова, якщо ні, подається сигнал що помилка є, але її не можна виправити. При виправленні одиноких стирань коли t=1 та $d\geq 3$, також знаходиться дві інші помилки, що призводить до зберігання цілісності коду.

Для виправлення однієї та знаходження подвійної стирання можуть використовуватись скорочені коди Хемінга які мають мінімальну надлишковість

Коди SEC-DED не здатні до виправлення або виявлення байтових стирань або пакетів стирань. З іншого боку, коди SBEC і коди SbEC-DbED можуть виправити байтові помилки, але вимагають великої надлишковості. SEC-DED-SbED клас кодів практичний з точки зору вимоги введення маленької додаткової надлишковості до існуючих кодів SEC-DED. Можна сказати, що коди SEC-DED-SbED є спеціальним класом коду SEC-DED-BED в тому сенсі, що помилка може виникнути в межах одного байта.Коди SEC-DED-SbED важливі з практичної точки зору, і тому вони активно вивчаються.

Dutta SEC-DED-DAEC (виправлення одиночної помилки, знаходження подвійної помилки, виправлення двох сусідніх стирань) це компроміс між низькими витратами SEC-DED кодів та високою корекційною здатністю БЧХ кодів та кодів Ріда-Соломона. В них використовується така ж кількість контрольних бітів як і в кодах Хемінга та Hsiao, але існує додаткове обмеження на процедуру вибору стовпця *H*-матриці (рисунок 2.2).[18]

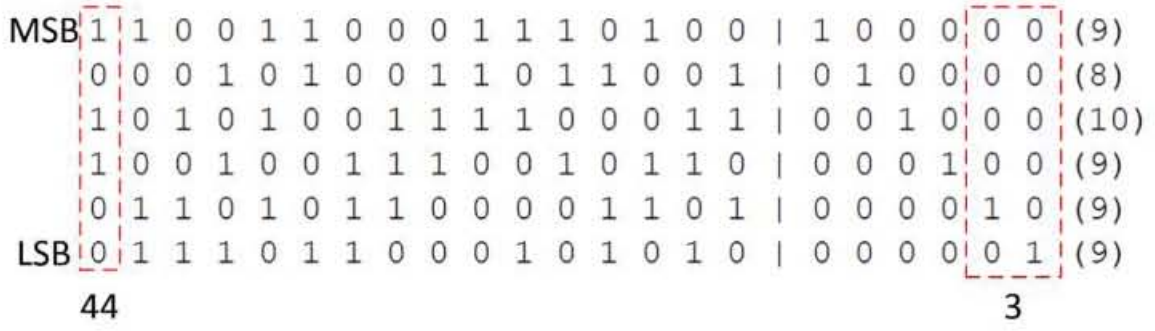


Рисунок 1.6 - Матриця SEC-DED-DAEC Dutta коду

Це обмеження дає можливість виправляти дві сусідні помилки. Для роботи всіх SEC-DED-DAEC кодів існують такі обмеження:

1. Жоден із стовпців не повинен складатись з нульового вектору.
2. Всі стовпці повинні відрізнятись.

3. Додавання по модулю 2 будь яких двох стовбців не повинне дорівнювати якомусь із існуючих стовбців.

4. Лінійна комбінація будь-яких двох сусідніх стовбців не дорівнює жодному з окремих стовбців або лінійній комбінації будь-яких інших двох сусідніх стовбців.

Перші три обмеження ідентичні коду Хеммінга, а четверте обмеження, запропоноване Dutta, вимагає, щоб лінійні комбінації будь-яких двох суміжних вектори стовпчика $h_i \oplus h_{i+1}$ відрізняються один від одного і від кожного окремого вектору стовпця h_i. Саме четверте обмеження дозволяє виправляти дві сусідні помилки.

Циклічні коди Хеммінга – це коди які забезпечують виявлення двобітних стирань і виправлення однобітних стирань.

Коди Хеммінга, що виправляють одиночну помилку (d_{min}= 3) Для кодів Хеммінга, що виправляють одиночну помилку, відразу будується перевірочнаматриця, що містить ррядків і n стовпців. Кожен стовпець перевірочної матриці дорівнює номерові позиції стовпця в двійковому виді. Якщо стовпець містить однуодиницю, він відповідає перевірочному символові. Всі інші стовпці від– повідають інформаційним символам. Приклад: вихідні дані для побудови кодуХеммінга s=1, dmin= 3, k = 4 p ≥ [log2{(k + 1) + [log2 (k + 1)]}], p = 3 n = k+p = 7 Будуємо перевірочну матрицю для коду.[8]

$$\mathbf{H} = \begin{vmatrix} b_1 & b_2 & a_3 & b_4 & a_5 & a_6 & a_7 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{vmatrix}$$

Коди Файра – це коди, які виявляють і виправляють пакети стирань. Під пакетом стирань довжиною b розуміють такий вид комбінації перешкоди, у якій між крайніми розрядами, ураженими перешкодами, міститься b–2 розряду.

Для контролю сучасної напівпровідникової пам'ятінайчастіше використовуються байтові коди, які орієнтовані на пошук та виправлення помилок в окремих байтах (табл. 1.2).

SEC–DED код тобто виправлення однієї стирання і виявлення подвійної стирання. Як приклад можна навести модифікований код Хеммінга типу SEC–DED. Модифікований код Хеммінга відрізняється від немодифікованого додатковим перевірочним бітом, який є загальним бітом правдивості всього кодового словаХеммінга. Однак, при появі стирання кратністю більше 2 (особливо непарної кратності 3,5,7…) існує ймовірність появи помилкової корекції.

SbEC код виправляє один b–бітовий помилковий байт.

SbEC–DbED код виправляє один *b*–бітовий помилковий байт або знаходить два помилкових *b*–бітових байта найкраще підходить для корекції стирань в високошвидкісних системах пам'яті з використанням байт–організованих мікросхем RAM. Даний код має меншу надлишковість і більш високу здатність контролю стирань, ніж існуючі коди.

Таблиця 1.3 - Байтові коди

Код	Позначення
SEC–DED	Single Error Correction Double Error Detection
SbEC	Single b–bit Byte Error Correcting Codes
SbEC–DbED	Single b–bit Byte Error Correcting and Double b–bit Byte Error Detection
SbEC–Spxb/BED	Single b–bit Byte Error Correcting and Single p–byte within a B–bit Block Error detecting codes
SbEC–DEC	Single b–bit Byte Error Correcting and Double–Bit Error Correcting Codes
SbEC–DED	Single b–bit Byte Error Correcting and Double–Bit Error Detecting Codes

St/bEC	Single Spotty Byte Error Correcting Codes
St/bEC–SbED	Single Spotty Byte Error
St/bEC–Dt/bED	Single Spotty Byte Error Correcting and Double Spotty Byte Error Detecting

SbEC–DbED код виправляє один *b*–бітовий помилковий байт або знаходить два помилкових*b*–бітових байта найкраще підходить для корекції стирань в високошвидкісних системах пам'яті з використанням байт–організованих мікросхем RAM. Даний код має меншу надлишковість і більш високу здатність контролю стирань, ніж існуючі коди.

SbEC–Spxb/BED код який виправляє один *b*–бітовий помилковий байт або виявляє помилку з p байт в блоці довжиною B біт.

SbEC–DEC код який виправляє один помилковий *b*–бітовий байт або виявляє двапомилкових біта.

SbEC–DED код який виправляє один помилковий *b*–бітовий байт або виявляє двапомилкових біта.

St/bEC код який виправляє будь–які t помилкових біта в одному*b*–бітовому байті.

St/bEC–SbED код виправляє будь–які t<b помилкових біт в одному b–бітовому байті або виявляє помилковий байт якщо t<b.

St/bEC–Dt/bED код виправляє будь–які t<b помилкових біт в одному b–бітовому байті або виявляє два помилкових байта якщо t<b.

В даномурозділі розглянуто класифікацію напівпровідникових пристроїв пам'яті, розглянуто їххарактеристику, розвиток та проблеми які можуть виникнути у пам'яті. Розглянуті завадостійкі коди для діагностики стирань пам'яті. Розглянуті загально відомі коди для контролю напівпровідникової пам'яті, а саме коди Ріда-Соломона, модифіковані коди Хеммінга, та байтові

коди. Описано сучасні технології захисту пам’яті, такі як ECC-пам’ять та технологія Chipkill.

2. РОЗРОБКА МОДИФІКОВАНИХ ІТЕРАТИВНИХ МЕТОДІВ ДЕКОДУВАННЯ ЗАВАДОСТІЙКИХ КОДІВ

2.1 Обґрунтування виборузавадостійких кодів для напівпровідникової пам'яті.

В наш час відбувається бурхливий розвиток комп'ютерних і телекомунікаційних мереж. Під час передавання, обробки і збереження даних в цих мережах можливі різноманітні спотворення і втрати даних. Для боротьби з цими завадами та захисту даних від спотворень створені спеціальні завадостійкі коди, які суттєво підвищують достовірність операцій з даними.

При виборі завадостійких кодів необхідно враховувати різноманітні кодові характеристики: тривалість кодування і декодування, складність програмно-апаратної реалізації тощо. Найважливішою характеристикою коду є його коректувальна здатність: чим більше помилок може виявити та виправити код, тим він краще. При цьому головною проблемою є оптимальний вибір самого критерію оцінювання. Зароки існування теорії завадостійкого кодування склались два таких критерії.

Для ймовірнісних кодів (кодів LDPC, турбо-кодів) зазвичай використовується ступінь наближеннядо межі Шеннона графіка залежності ймовірності появи помилкового біта в переданому повідомленні до спектральної густини потужності шуму в каналі зв'язку: чим ближче до межі Шеннона, тим краще.

Для детермінованих лінійних кодів використовується параметр dmin мінімальної кодової відстані, який безпосередньо визначається кількістю виявлених tdабо виправлених tc помилок:$dmin=2_{tc}+1= td+1$ [1].

Розглянемо детальніше сенс параметраd_{min} для заданого лінійного (n,k)-коду. При збільшенні числаr $(r=n-k)$ контрольнихрозрядів коду пропорційно зростає кількість виявлених та виправлених помилок, однак зменшується обсяг переданої по каналу корисної інформації і значноускладнюється процес декодування.

Оптимальним розв'язанням зазначеної проблеми стало використання в сучасних завадостійких кодах принципу ітеративності, який дозволяє розбити складну процедуру декодування наряд відносно простих кроків (ітерацій). Що ж відбувається з параметром dmin при ітеративному декодуванні?

Параметр dmin був запропонований ще для опису перших завадостійких кодів, в яких декодування відбувалось лише заодну ітерацію. Тому dminза визначенням став статичною та незмінною характеристикою конкретного лінійного (n,k)-коду [9].

Під час ітеративного декодування dmin може змінюватись, тобто стає динамічною характеристикою коду. Починаючи із початкового статичного значення dmin поступово зростає, що еквівалентно збільшенню числа помилок, які можна виявити і виправити. Дослідженню цього процесу і становить суть даної роботи.

Підвищити коректувальну здатність коду під час його ітеративного декодування можна лише при використанні альтернативних кодових перевірок. Такі перевірки дозволяють формувати перевіряльні рівняння з різними розрядами кодового слова. В найпростішому варіанті це можуть бути перевірки парності різних груп розрядів кодових слів, а в більш складному варіанті – перестановки розрядів за допомогою спеціальних перемежувачів.

Як приклад розглянемо код Елаєса, в якому (n×m)-розрядне кодове слово записується у вигляді (n×m)-матриці, в якій використовуються перевірочні рівняння по парності порядкам та по стовпцям. Якщо перевірка по парності кодового слова в послідовному форматі дозволяє лише виявити помилки непарної кратності, тоді як перевірки по парності в матричному форматі дозволяють вже виправити поодинокі помилки і виявити помилки подвійної та непарної кратності. Одна, і це ще не межа, якщо сформувати додатковуальтернативну перевірку наоснові контрольної суми CRC для кодового словау послідовному форматі. Для точної локалізації подвійних помилок необхідно знову повернутись до послідовного формату таким чином, щоб кожній конфігурації помилок відповідалоодне кодове слово з відповідними

помилками. Таким чином, основні етапи алгоритму ітеративного декодування коду Елаєса будуть такими.

1. З каналу зв'язкуотримати послідовно (побітово) n-розрядне ($n=n_1\times n_2$) кодове словоZi еталонне значення CRC.

2. Сформувати зі словаZкодову ($n_1\times n_2$)-матрицю M.

3. Виконати всі контрольні перевірки порядкам і по стовпцям матриці M.

4. Сформувати h можливих шаблонів помилок для ненулевих перевірок.

5. Для i-го шаблону помилок сформувати послідовне кодове словоZ_i, обчислити для ньогофактичне значення i-гоCRC і порівняти з еталоннимCRC.

6. Збіг значень однієї із вказаних парCRC буде свідчити про знайдену конфігурацію помилковихрозрядів в слові Z_i, а значить, і в слові Z.

2.2Модифікація кодів Елаеса для декодування помилок в напівпровідниковій пам'яті.

Ітеративні коди,якщо вони орієнтовані на виправлення однократних помилок, являють собою, як правило, двомірні лінійні коди з кодуваннямрядків і стовпців завадостійкими кодами з перевіркою на парність . Такі ітеративні коди мають мінімальну кодову відстань d_{min}= 4і урежимі виправленняпомилок дозволяють виправити на практиці використовувати коди з числом перевірочних елементів8, 9 та 16. Для коду зr = 8 використовують блокінформаційнихелементіврозмірами3 ´ 4 (зk_1= 3рядкамита k_2 =4 стовпцями). Прицьомучисло інформаційних елементівk= k_1 ´ k_2=3 ´ 4= 12,число перевірочних–r= 8, n = 20. Для коду зr = 9берутьk =16, n =25;для коду зr = 16:або k = 56,n = 72абоk = 56,n = 72. При виправленні помилки у декодері визначають рядок і стовпець, для яких не виконуються умови парності.

Спотворений інформаційний елемент, розташований на місці перетинурядка і стовпця, для яких не виконується перевірка на парність, інвертується.[10]

Передачу символів такого коду зазвичай здійснюють послідовно символ за символом, від одногорядка до іншого, або паралельно цілими рядками. Декодування починають відразу, не чекаючи надходження всього блоку інформації. Перевіркарівнянь при декодуванні дозволяє виявити будь-непарне число спотворених символів, розташованих в одномурядкуабо в одному стовпці. Перевіркаодиночної помилки порядку, вказує на наявність помилки в ній, а перевірка по стовпцю - конкретний символ. Однак цим кодом не можуть бути виявлені помилки, які мають парне число спотворених символів, як порядках, так і по стовпцях.

Недоліком ітеративних кодів з перевірками на парність порядках і стовпцях є їх порівняно високанадлишковість, яка становить 15 - 20% і значно перевищує за іншихрівнихумов надлишковість циклічних кодів. Дійсно, наприклад, при ітерації двох кодів Хеммінга з dmin 3, що забезпечують виправлення одноразових помилок, виходить код з dmin 3 - 3 9, який повинен виправляти помилки кратності tu 4 і менше.

Надлишковість двомірних ітеративних кодів:

дляr= 8 $R = r / n$=2 / 5;

дляr=9 R = 9 / 25;

дляr= 16R = 2 / 9.

Висновокітеративного методу

Якщо помилки знаходяться в горизонтальному та вертикальному стовпчику то мінімальна кратність помилки дорівнює 2.

Якщо дві помилки знаходяться лише в горизонтальномурядку то можна лише виявити ці помилки але не можна виправити.

Якщо дві помилки знаходяться лише в вертикальномурядку томожна лише виявити ці помилки але не можна виправити.

Бувають такі варіанти помилок:

Варіантиодиночних помилок:

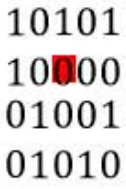


Рисунок 2.2 – Варіанти одиночних помилок

Синдром: одиночнаабо потрійна помилка.

Варіанти подвійних помилок

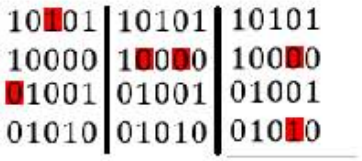


Рисунок 2.3 – Варіанти подвійних помилок

Типи потрійних помилок

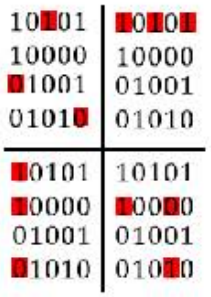


Рисунок 2.4 – Варіанти потрійних помилок

Як видно з класифікації помилок, для виправлення деяких подвійних та потрійних помилок необхідне підтвердження місцезнаходження цих помилок за допомогою CRC коду.Циклічний надлишковий код або CRC-код — алгоритм обчислення контрольної суми, призначений для перевірки цілісності даних. CRC є практичним додатком завадостійкого кодування, заснованому на певних математичних властивостях циклічного коду.У загальному вигляді контрольна сума являє собою деяке значення, обчислене за певною схемою на основі кодованого повідомлення. Перевірочна інформація при систематичному

кодуванні приписується до переданих даним. На приймаючій стороні абонент знає алгоритм обчислення контрольної суми: відповідно, програма має можливість перевірити коректність прийнятих даних.CRC засновані на теорії циклічних кодів, що виправляють помилки. Використання систематичних циклічних кодів, які кодують повідомлення, додаючи контрольну величину фіксованої довжини, з метою виявлення помилок у мережах зв'язку, вперше було запропоновано У. Веслі Петерсоном у 1961 році. Циклічні коди не просто прості у здійсненні, але мають вигоду, що вони особливо добре підходять для виявлення помилок всплеску: суміжні послідовності помилкових символів даних у повідомленнях. Це важливо, оскільки помилки розриву є поширеними помилками передачі в багатьох каналах зв'язку, включаючи магнітні та оптичні пристрої зберігання даних. Зазвичай n-бітний CRC, застосований до блоку даних довільної довжини, буде виявляти будь-яку одиночну помилку помилок не довше n біт, а частка всіх довших помилок, які вона буде виявляти, становить (1 - 2 − n). Специфікація коду CRC вимагає визначення так званого генератора многочлена. Цей многочлен стає дільником на багаточленному поділі, який приймає повідомлення як дивіденд і в якому коефіцієнт відкидається, а решта стає результатом. Важливим застереженням є те, що коефіцієнти поліномів обчислюються відповідно до арифметики кінцевого поля, тому операція додавання завжди може виконуватися порозрядно-паралельно (немає переносу між цифрами). На практиці всі широко використовувані CPC використовують поле Галуа з двох елементів, GF (2). Обидва елементи зазвичай називаються 0 і 1, що зручно відповідають архітектурі комп'ютера. CRC називається n-бітною CRC, коли її контрольне значення n біт. Для даного n можливі кілька CRC, кожен з яких має різний многочлен. Такий многочлен має найвищий ступінь n, що означає, що він має n + 1 доданків. Іншими словами, многочлен має довжину n + 1; його кодування вимагає n + 1 біт. Зауважте, що більшість специфікацій поліномів або відміняють MSB, або LSB, оскільки вони завжди є 1. CRC та пов'язаний

многочлен зазвичай мають назву форми CRC-n-XXX, як показано в таблиці нижче.

Найпростіша система виявлення помилок, біт парності, насправді є 1-бітовою CRC: вона використовує генератор полінома x + 1 (два доданки) і має назву CRC-1.

При передачі пакетів з мережевого каналу вихідна інформація може бути спотворена через різні зовнішні впливи: електричні перешкоди, негоду та багато інших. Суть цієї методики полягає в тому, що при хороших властивостях контрольної суми у переважній кількості випадків, помилка в повідомленні змінює контрольну суму. Якщо вихідні та обчислені суми не збігаються, отримані дані будуть недостовірними, і необхідно подати запит на повторну передачу пакета.

Алгоритм CRC базується на властивостях ділення із залишком довічних многочленів, тобто многочленів над кінцевим полем.

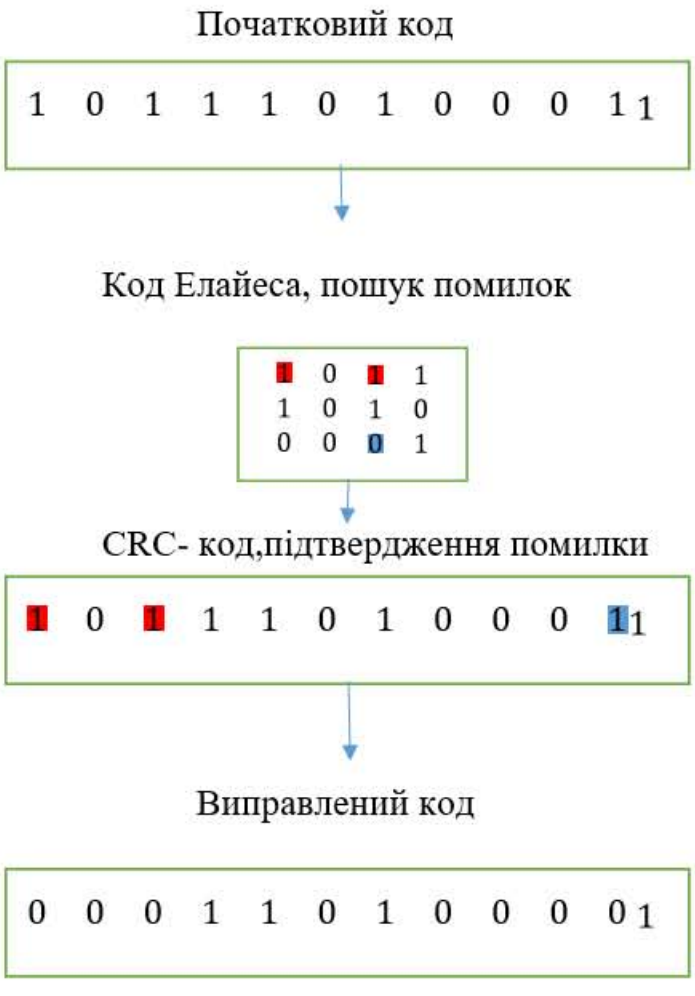


Рисунок 2.1 - Принцип виправлення помилок за допомогою модифікованого коду Елайеса

2.3 Кодування та декодування кодів Ріда-Соломона

Коди Ріда-Соломона є окремим випадком кодів БЧХ. Основна відмінність коду Ріда-Соломона полягає в тому, що символ не є двійковим символом, а є елементом Галуа. Поліном генеративної помилки коду Ріда-Соломона повинен містити 2s: {i 0, i 0 +1, i 0 +2, ..., i 0 + 2s-1}, де j_0 - конструктивний параметр, j_0 зазвичай дорівнює 1. Тоді множина поліноміальних коренів має вигляд {a, a_2, a_3 ... α_{2s}} ... Для виправлення помилки генеруючий многочлен Ріда-Соломона має вигляд: RS (x) = (x - α) (x - α_2) (x - α_3)… (x - α_{2s}), очевидно, що генеруючий многочлен має набір коренів {a, α_2, α_3… α_{2s}} ... Для коду Рід-Соломона, з одним виправленням помилок (s = 1) генеруючим многочленом є RS (x) = (x - α) (x - α_2). Існує кілька форм генеративних поліноміальних позначень для кодів Ріда-Соломона. Розширення GF (2) примітивним поліномом p(z) зазвичай використовуються для побудови кодів Ріда - Соломона. Тому замість написання примітивного елемента ми можемо використовувати z: RS (x) = x_2 + (a + a_2) · x + a_3 = x_2 + (z + z_2) · x + z_3. Друга форма буде залежати від того, яке поле використовується для побудови коду Ріда-Соломона. Давайте розглянемо цю форму як приклад. Приклад. Побудуємо поле Галуа GF (8) як розширення поля GF (2) над примітивним многочленом p (z) = z_3 + z + 1. Елементи поля можуть бути представлені в різних позначеннях.

Тому для коду Рід-Соломона виходять різні форми генеративних многочленів, які виправляють окремі помилки: RS (x) = (x - a) (x - a_2) при j0 = 1 - не використовуються для генерації таких схем; RS (x) = (x - a (x - a_3) для j0 = 2 - не використовується в проектуванні схем; RS (x) = x_2 + (z + z_2) · x + z_3 для j0 = 1 - незалежно від конкретного поля 4) RS (x) = x_2 + (z_3 + z_2) x + z_5 при j0 = 2 - не залежить від конкретного поля Галуа і може бути використаний для побудови

схем кодування та декодування ; RS (x) = x_2 + (z + z_2)x + z + 1 для j0 = 1 - залежить від конкретного поля Галуа і може бути використаний для побудови схем кодера і декодера; RS (x) = x_2 + (z_2 + z + 1) x + z_2 + z + 1 для j0 = 2 - залежить від конкретного поля Галуа і може використовуватися для формування схем пристрою кодування та декодування; RS (x) = x_2 + a_4 x + a_3 при j0 = 1 - залежить від конкретного поля Галуа і може використовуватися для збирання схем кодера і декодера; RS (x) = x_2 + i_5 x + i 5 при j0 = 2 - залежить від конкретного поля Галуа і може бути використаний для побудови схем кодера і декодера. але основними параметрами є параметри корекції, довжина коду n, кількість інформації k та валідація nk = p однакові. Зазвичай 7, 8 варіантів генерування поліномів використовуються для представлення функціональних схем пристроїв кодування та декодування, тип схеми є найбільш компактним, але побудова таких поліномів вимагає побудови поля Галуа. Для розробки схематичних діаграм бажано використовувати 3-6 варіантів поліномів, оскільки вони не потребують побудови поля Галуа, і 5, 6 варіантів є кращими. Розглянемо приклади. Для поля GF (8) та генеративного полінома RS (x) = x2 + i 4 x + та 3 код коду виправлення помилок ReedSolomon (одна потрійна - три двійкові символи) має такі параметри: довжина коду n = 7, кількість тестових символів p = degRS (x) = 2, кількість інформаційних символів k = np = 5. Таким чином, ми реалізуємо код Ріда-Соломона (7, 5), а цифри 7 і 5 вказують на кількість трьох двійкових символів. Кодер цього коду аналогічний датчику циклічного коду Хеммінга. Як і у випадку з кодами Хеммінга, код системного коду Ріда-Соломона - це схема множення поліномів і генеративний поліноміальний поділ; Несистемний кодер Ріда-Соломона - це схема, що генерує поліном. Різниця полягає в різній інтерпретації символу кодового слова і відповідно елементів пам'яті та множників постійною. Схема декодера аналогічна декодерудля циклічного коду Хеммінга, за винятком того, що кожен елемент затримки в цьому випадку має не одну комірку, а три.

У рівнянні представлена найбільш поширена форма кодів Ріда Соломона через параметри n, k, t і деяке позитивне число t> 2. Наведемо це рівняння повторно:

(n, k) = (2m — 1,2m - 1-2t), (2.1)

Тут n - k = 2t - число контрольних символів, а t - кількість помилкових бітів в символі, які може виправити код. Поліном для коду Ріда-Соломона має наступний виглад:

$$g(X) = g_0 + g_1X + g_2X^2 + \dots + g_{2r-1}X^{2r-1} + X^{2r} \quad (2.2)$$

Ступінь поліноміального генератора дорівнює числу контрольних символів. Зв'язок між ступенем поліноміального генератора і числом контрольних символів, як і в кодах БХЧ, не повинна виявитися несподіванкою. Оскільки поліноміальний генератор має порядок 2t, ми повинні мати в точності 2t послідовні ступеня a, які є початком полінома.

Позначимо корені g (X) як: α , $\alpha^2 \dots \alpha^{2r}$. Немає необхідності починати саме з кореня α , це можна зробити за допомогою будь-якого ступеня α . Візьмемо до прикладу код (7, 3) з можливістю корекції однорядкове резюме помилок. Ми висловимо поліноміальний генератор через 2t = n – k=4 кореня наступним чином:

$$g(X) = (X - \alpha)(X - \alpha^2)(X - \alpha^3)(X - \alpha^4) = (X^2 - (\alpha + \alpha^2)X + \alpha^3)(X^2 - (\alpha^3 + \alpha^4)X + \alpha^7) = (X^2 - \alpha^4 X + \alpha^3)(X^2 - \alpha^6 X + \alpha^0)$$

$$= X^4 - (\alpha^4 + \alpha^6)X^3 + (\alpha^3 + \alpha^{10} + \alpha^0)X^2 - (\alpha^4 + \alpha^9)X + \alpha^3 =$$

$$= X^4 - \alpha^3\alpha^3 + \alpha^0 X^2 - \alpha^1 X + \alpha^3.$$

Помінявши порядок розташування членів полінома на зворотний і замінивши знаки "мінус" на "плюс", так як над двійковим полем +1 = -1, генератор g (X) можна буде представити в такий спосіб:

$$g(X) = \alpha^3 + \alpha^1 X + \alpha^0 X^2 + \alpha^3 X^3 + X^4 \quad (2.3)$$

Декодування кодів Ріда-Соломона значно складніше кодування. очевидно, що першим кроком необхідно виконати поділ полінома на який породжує поліном g(x). Якщо залишок дорівнює нулю, то повідомлення не спотворено і декодування тривіально: слід просто виділити з повідомлення коефіцієнти з N-k + 1 до N-1 - це і буде основне повідомлення. Уразі ж присутності помилки доведеться виконати наступні дії. Декодування засноване на побудові многочлена синдрому помилки S(x) і знаходженні відповідного йому многочлена локаторів L (x). Локатори помилок - це елементи поля Галуа, ступінь яких збігається з позицією помилки. Так, якщо спотворений коефіцієнт при x^4 , То локатор цієї помилки дорівнюєa^4 , якщо спотворений коефіцієнт приx^7 то локатор помилки дорівнюватимеa^4. Многочлен локаторів L(x) - це многочлен, коріння якого протилежні локаторам помилок. Таким чином, многочлен L(x) повинен мати вигляд L (x) = (1 + xX1) (1 + xX2) ... (1 + xXi), де X1, X2, Xi - локатори помилок. Зрозуміло, що якщо цей многочлен буде знайдений, то ми легкозможемо визначити локатори помилок - для цього буде потрібно лише визначити його коріння, що легко зробити звичайним перебором. [11]

Декодер, що працює по авторегресівному спектральним методом декодування, послідовно виконує наступні дії:

1) Обчислює синдром помилки;
2) Будує поліном помилки;
3) Знаходить коріння даного полінома;
4) Визначає характер помилки;
5) Виправляє помилки.

Обчислення синдрому помилки виконується синдромним декодером, який ділить кодове слово на породжує многочлен. Якщо при розподілі виникає залишок, то в слові є помилка. Залишок від ділення є синдромом помилки.

лений синдром помилки не вказує на цей помилок. Ступінь полінома синдрому дорівнювати 2t, що багато разів студенти кодового слова n. Для достовірності між помилкою та її наявністю у представленому майбутньому

полімінієвілок. Поліном помилок реально використовується для використання алгоритму Берлекемпа - Мессі, або для використання алгоритму Евкліда. Алгоритм Евкліда має можливість реалізувати реалізацію, але вимагає великих ресурсів. Завдяки цьому існує більший склад, але але затриманий і алгоритм Берлекемпа-Мессі. Коефіцієнти знайдені поліноми, що досягають великих коефіцієнтів, притаманні символам у кодовому слові.

Обчислений синдром помилки не вказує на місце розташування помилок. Ступінь многочлена синдрому становить 2t, що набагато менше, ніж ступінь кодового слова n. Журнал помилок складається у повідомленні, яке відповідає помилці та її розташуванню. Протокол помилок реалізований за допомогою алгоритму Берлекампа-Мессі або алгоритму Евкліда. Алгоритм Евкліда легко здійснити, але вимагає багато ресурсів. Тому частіше використовується складніший, але дешевший алгоритм Берлекампа-Мессі. Коефіцієнти знайденого многочлена безпосередньо відповідають коефіцієнтам помилок символів у кодовому слові.

Обчислений синдром помилки не вказує на місце розташування помилок. Ступінь многочлена синдрому становить 2т, що значно менше ступеня кодового слова n. Складається журнал помилок, який відповідає помилці та її розташуванню. Журнал помилок реалізований за допомогою алгоритму Берлекамп-Мессі або алгоритму Евкліда. Алгоритм Евкліда легко здійснити, але вимагає багато ресурсів. Тому частіше використовується складніший, але дешевший алгоритм Берлекампа-Мессі. Коефіцієнти знайденого многочлена безпосередньо відповідають коефіцієнтам помилок символів у кодовому слові.Далі шукаються корені полінома помилки, що визначають положення перекручених символів в кодовому слові. Реалізується за допомогою процедури Ченя, равносильной повного перебору. У поліном помилок послідовно підставляються всі можливі значення, коли поліном наближається до нуля - коріння знайдені.

За синдрому помилки і знайденим коріння полінома за допомогою алгоритму Форне визначається характер помилки і будується маска

перекручених символів. Однак для кодів РС існує більш простий спосіб відшукання характеру помилок.

Далі знайдена маска накладається на кодове слово і спотворені символи виправляються, а перевірочні символи відкидаються та виходить відновлене інформаційне слово.[12]

2.4Модифікація кодів Ріда-Соломона для декодування помилок в напівпровідниковій пам'яті

2.4.1 Представлення кодів Ріда-Соломона в полях Галуа

Коди Ріда–Соломона (РС) – недвійкові циклічні коди, що дозволяють виправляти стирання в блоках даних. Елементами кодового вектора є не біти, агрупи бітів (блоки).Коди Ріда-Соломона мають високу корегувальну здатність і широко використовуються в різних сферах: від систем передачі даних до захистуоптичних дисків.

Коди РС є підкласом (n,k)-кодів БЧХ над недвійковим полем GF(q), для яких довжина блокускладає $n=q-1$ ($q=p^m$, $m=2,3,4, \ldots$). Далі будеморозглядати лише випадок $p=2$. Для побудови породжувального багаточлена кодуРС використаємо ту методику, що і для кодів БЧХ.

Спочатку для заданої довжини n коду формуємо поле GF(2^m), де $m=log_2(n+1)$. Для обчислення елементів поля $GF(2^m)$,будемовикористовувати ЛПС над полем $GF(2)$, яку будемо також йменувати двійковою. Ця ЛПС визначається за допомогою формул 2.3 2.4.

$$1) \quad S(t+1) = A \times S(t) + B \times U(t) \qquad (2.4)$$

$$2) \quad Y(t) = C \times S(t) + D \times U(t) \qquad (2.5)$$

Матриця A двійкового ЛПС містить коефіцієнти примітивного багаточлена:

h(x) = $1+h_1x+h_2x+\ldots +h_{m-2}x^{m-2}+h_{m-1}x^{m-1}+x^m$ який використовується при побудові поля GF(2).

Нульового елементу поля $GF(2^m)$ буде відповідати слово нульового стану S(0) двійкової ЛПС. Решту елементів поля знаходимо за формулою

$a^i = A \times a^{i-1}$, $i = 1 \dots 2^m - 1$. (2.6)

Далі для поля GF(2^m) будуємо циклотомічні класи. Таких класів є $(n-1)$, причому кожний з них містить тільки один елемент поля:

$C_a^i = \{a^i\}$, $i = 0 \dots 2^m - 2$. (2.7)

Отже, кожному циклічному класу відповідатиме один мінімальний багаточлен:

$f_i(x) = (x - a^i)$, $i = 1 \dots 2^m - 1$. (2.8)

Таким чином, породжувальний багаточлен кодуРС, який дозволяє виправити τ_{min} помилок, матиме вигляд

$g(x) = (x-a)(x-a^2) \dots (x-a^{2\tau_{min}})$, $GF(q)$. (2.9)

Оскільки степінь породжувального багаточлена коду дорівнює $2\tau_{min}$, тому для виправлення τ_{min} випадкових помилок над полем GF(q) необхідно $r = 2\tau_{min}$ перевірочних символів.

Доведено, що коди РС є МДВ-кодами, тобто при фіксованих n і k не існує коду, у якого d_{min} більше, ніж для кодуРС:

$d_{min} = n-k+1$.

Для розуміння принципів кодування і декодування недвійкових кодів, таких як коди Ріда-Соломона, потрібно зрозуміти поняття поля Галуа(Galois fields - GF). Для будь-якого числа p існує поле Галуа, яке позначається GF(p) і зберігає p елементів. Поняття GF(p) можна застосувати на поле з p^m елементів, які називаються полем розширення GF(p), це поле позначається GF(p^m), де m – ціле число.

Кожен не нульовий елемент в GF(2^m) можна представити як степінь a. Нескінченна кількість елементів ,F, з'являється з стартової множини {0,1,a}і генерується додатковими елементами. Цього можна досягти послідовним множенням останнього запису наa.

$F = \{0,1,a,a^2,\dots,a^j,\dots\} = \{0,a^0,a^1,a^2,\dots,a^j,\dots\}$

2.4.2Операція додавання та множення в полі зрозширення GF(2^m)

Полем називається набір елементів, що визначають дві операції. Один називається додаванням і позначається a + b, а другий помножується на a позначається a, b, хоча ці операції не є загальним складанням і множенням. Для того, щоб набір елементів був полем операцій додавання та множення, для кожної з цих операцій необхідно було мати комутативність (a + b = b + i ab = ba), асоціативність (a + (b + c) = (a + b) + caa (bc) = (ab) c) і закон розподілу також був дотриманий, тобто для всіх трьох елементів a, b, c, (b + c) = ab + ac a (b + c) a = b + co. Слід зазначити, що властивості групи для операції множення дійсні для всіх ненульових елементів поля. Поля з обмеженою кількістю елементів q називаються полями Галуа по імені їх першого дослідника, Галуа Евариста, і відносяться до GF (q). Кількість елементів масиву q називається порядком масиву. Таке поле повинно містити 2 окремих елементи: 0 для операції додавання та 1 для операції множення. Це GF (2) або двійкове поле. Залежно від q, є поодинокі або розширені поля. Поле називається простим, якщо q - простим. Символ p використовується для позначення простих чисел. Просте поле складається з модулів p чисел: 0, 1, 2, p-1, а операції додавання та множення виконуються за модулем r. Якщо поле складається з елементів q^m, то це поле називається розширенням. m поля над GF (p) або розширеним полем. Він містить елементи p^m і позначається GF (p^m). Далі ми розглянемо поля GF (2^m). Будь-яке кінцеве поле GF (2^m) є m-мірним вектором над полем GF (2). Поліном f (t) степені m над полем GF (2) є многочленом вигляду f (t) = tm + fm-1tm-1 + ... + f0, де коефіцієнти полінома fi належать GF (2). Операції на таких многочленах виконуються як операції на звичайних многочленах, тільки поля з коефіцієнтом виконуються в полі GF (2). 106 Поліном f (t) з ненульовим ступенем називається непридатним над полем GF (2^m), якщо він повністю розділиться через це поле на себе і на поліноми нульового ступеня. Елемент x кінцевого поля GF (2^m) називається коренем многочлена f (t), якщо f (x) = 0. Якщо x - корінь невідворотного многочлена p (x) ступеня m, то елементи (xm-1,

..., 1) складають основу підсумкового поля GF (2^m) як векторного простору над полем GF (2). Цю основу називають многочленом. Поліном, корінь якого є примітивним елементом, називається примітивним многочленом. Якщо ми виберемо примітивний многочлен ступеня m, який не зводиться над полем GF (2), як P (x), то отримаємо поле GF (2m) з усіх 2m двійкових послідовностей довжиною m. Основою многочлена є примітивний многочлен. Елементи кінцевого поля в основі многочлена представлені поліномами ступеня не більше m-1, або однаково бінарними рядами довжини m, що складаються з коефіцієнтів таких многочленів.

Кожен з 2^m елементів поля Галуа GF(2^m) можна представити як окремий поліном степені *m-1* або менше. Позначимо кожен ненульовий елемент GF(2^m) поліномом a_I(X), в якому хоча б *m* коефіцієнтів ненульові. Для *i* = 0,1,2,…, 2^m-2,

$$a_I=a_I(X)=a_{I,0}+a_{I,1}X+a_{I,2}X^2+\cdots+a_{I,m-1}X^{m-1}$$

Відзначимо, що в базовому простому полі GF (2) для елемента 0 зворотним елементом по додаванню є сам елемент 0, також як і для елемента 1 зворотним елементом по додаванню є сам елемент 1. Відповідно, як складання, так і віднімання елементів простого поля GF (2) фактично зводяться до однієї і тієї ж операції підсумовування по модулю 2. Тоді, при додаванні і відніманні елементів поля GF (2m) ми маємо складання відповідних коефіцієнтів многочленів по модулю 2 (при поданні у вигляді многочленів) або побітове додавання по модулю 2 відповідних розрядів двійкових чисел (при поданні у вигляді двійкових чисел).

2.4.2 Виконання операцій додавання і множення в простих скінченних полях GF(q)

Сума обчислюється суматором Z=(X+Y)q, де число Z є залишком від ділення суми X+Y на число q. Числа Z, X, Y, і q мають розрядність n та зобраені в двійковій формі. Розглянемо суму

S=(X+Y)+(2^n-q)де S має n+1 двійкових розрядів. Зрозуміло, що сума Sв заленості від X та Yприймає значенняS=2n. Якщо сума S=2n , то (n+1) розряд

числа S дорівнює 1 і з співвідношеня (3) випливає, що X+Y>=q, а отже, Z=X+Y-q.В даному розділі обгрунтовано вибір завадостійких кодів для напівпровідникової пам'яті, запропоновано модифікацію коду Елайеса для декодування помилок в напівпровідниковій пам'яті.

3. ПРОГРАМНО-АПАРАТНАРЕАЛІЗАЦІЯ ЗАПРОПОНОВАНИХ МЕТОДІВ

3.1 Апаратнареалізація модифікованого коду Елайеса

Для реалізації роботи будь-якого даного методу необхідно мати напівпровідникову пам’ять з блоком усунення помилок (Error-correcting code memory , ECC-пам'ять).Інформація подається у пам’ять одночасно закодованою методо Елайеса та CRC-кодером, під час зберігання данних в них можуть виникнути помилки. Найбільш поширений код корекції помилок у ECC-пам’яті, SECDED код Хеммінга, дозволяє виправити однобітні помилки, запропонований модифікований метод Елайеса дозволяє виправляти подвійні та потрійні помилки. Звичайний метод Елайеса не зможе правильно визначити помилки зображені нарисунку 3.2 тахибно виправить не той розряд кодового слова.

Рисунок 3.2 – Потрійна помилка яка буде визначена як одиночна

Також неможливо встановити місцезнаходження подвійних помилок які знаходяться в одномурядкуабо стовпці(Рисунок 3.3).

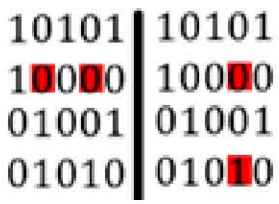


Рисунок 3.3 – Помилки які неможливо точно визначити

В даному випадку буде визначено лише стовпчик тарядок з помилками.

Для вирішення даних проблем було запропоновано використовувати CRC код. Сам по собі CRC-коди не можуть виправляти помилки але вони здатні їх знаходити. Принцип роботи модифікованого коду Елайеса показано нарисунку 3.4, аалгоритм роботи у додаткуВ.

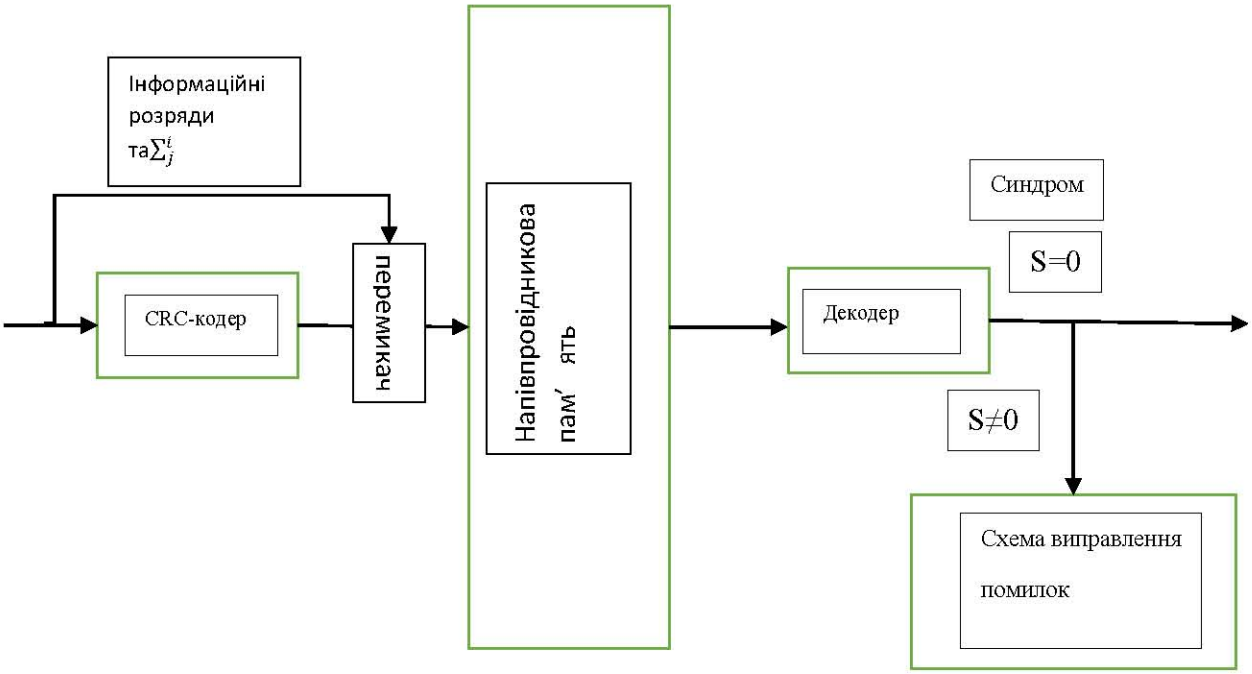


Рисунок 3.4 – Принцип роботи модифікованого методу Елайеса

Початковий код кодується методом Елайеса та паралельно перевіряється CRC кодом який підтверджує наявність виявлених помилок та вказує їх місцезнаходження, після цього можливе встановлення правильного синдрому та виправлення помилок.

Ітеративність даного методу показана нарисунку 3.5 та заключається в наступних кроках:

1) З початкового коду, який передається послідовно, формується матриця.

2) Код Елайеса перевіряє кожен рядок та стовпець матриці на наявність помилки, та контролюєтьсяCRC-кодом, це відбувається в циклі доки не зацінчиться матриця.

3) Виправляються знайдені помилки.

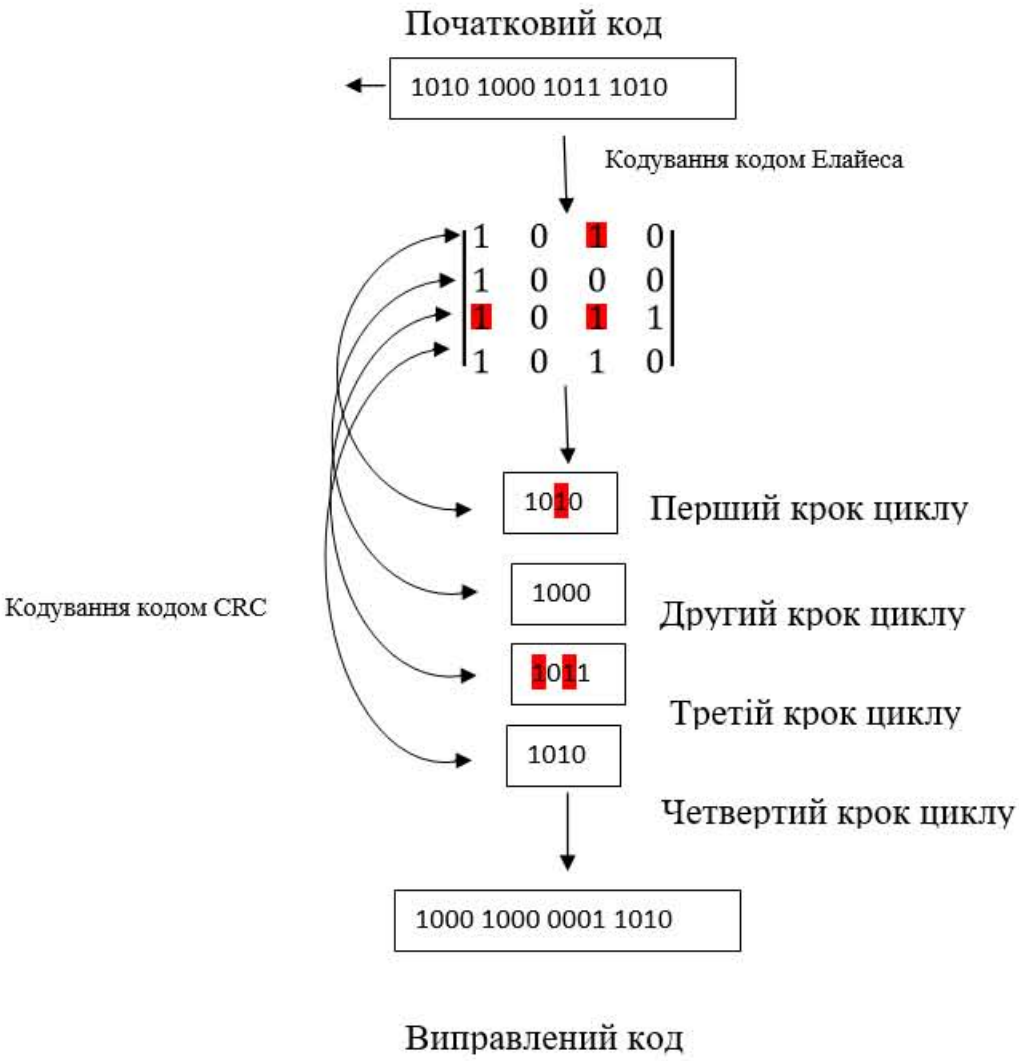


Рисунок 3.5 – Ітеративність модифікованого коду Елайеса

3.2 Опис роботи алгоритму модифікованого коду елайеса

Алгоритм модифікованого коду Елайеса представлений у додатку Д.

Можна виділити такі комбінації помилок:

1) m =2, w=2 подівйна помилка в різних стовпцях і рядках;
2) m=2, w=0 подвійна помилка в одному рядку;
3) m=0, w=2 подвійна помилка в одному стовпці ;

4) m=1, w=1 одиночна, або потрійна помилка;

5) m=3, w=1 потрійна помилка в одному рядку ;

6) m=1,w=3 потрійна помилка в одному стовпці;

Де m це кількість помилок в перевіряльному рядку, wце кількість помилок в перевіряльному стовпці, i_nномер розрядку перевіряльного рядкаj_n номер розряду перевіряльно стовпця.

Перша комбінація помилок виправляється інвертуванням розрядку на перетині і-того рядка та j-того стовпця. Друга комбінація –інвертуванням h_iпар розрядів в і-тому та j-тому розрядах перевіряльно рядка. Третя комбінація - інвертуванням h_iпар розрядів в j_1-тому та j_2-тому розрядах перевіряльно стовпця. Четверта – CRCкод дозволить підтвердити наявність одиночної чи потрійної помилки і в залежності від цього виправити їх. П'ята та шоста комбінації - інвертуванням розрядів на перетині і-того рядків та j-того стовпців.

3.3 Апаратнареалізація модифікованого кодуРіда-Соломона

Модифікований код Ріда-Соломонатакож можуть використовуватись у блоках контролю напівпровідникової пам'яті.

В таблиці 3.1 показані типи кодів які можуть використовуватись.

Код Хемінга який зазвичай використовується блоках контролю пам'ять має такі параметри: n=15, k=11, r=4. Де n – кількість розрядів, k – кількість інформаційних розрядів, r – кількість перевірочних розрядів t – кількість помилок які виправляються.

Таблиця 3.1 - Різновид кодів РС

n	k	r	t
7	3	4	2
15	9	6	3
31	23	8	4

63	53	10	5
15	11	4	2

Для модифікації було обрано 15,11 код Ріда-Соломона де GF(2^4). Принцип роботи модифікованого коду Ріда-Соломона показаний на рисунку 3.6.

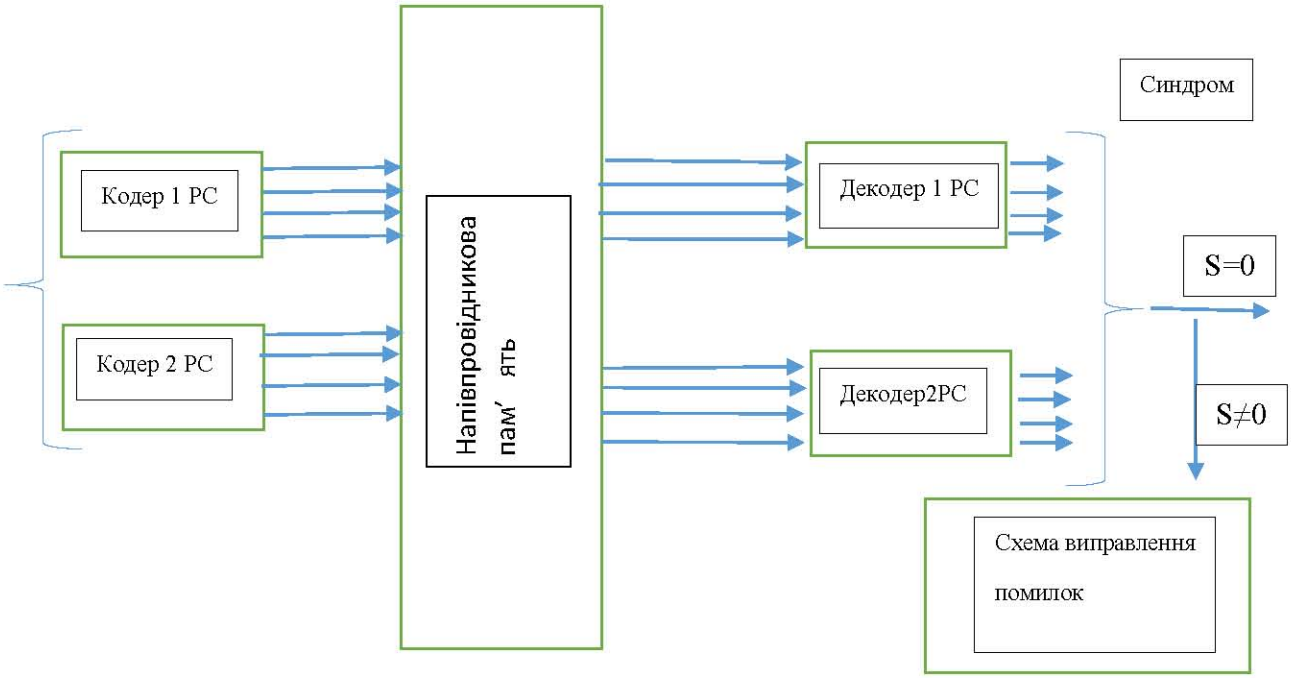


Рисунок 3.6 – Принцип роботи модифікованого коду Ріда-Соломона

З рисунку 3.6 видно що використовується два кодери Ріда-Соломона які паралельно передають інформацію в пам'ять по 4 потоки кожен. Це дозволяє випраляти 4 помилки або 8 помилкових біт (Рисунок 3.7), також можна було використати один кодер Ріда-Соломона з 8 потоками данних але кількість виправлених помилок зменшилась би до 3.

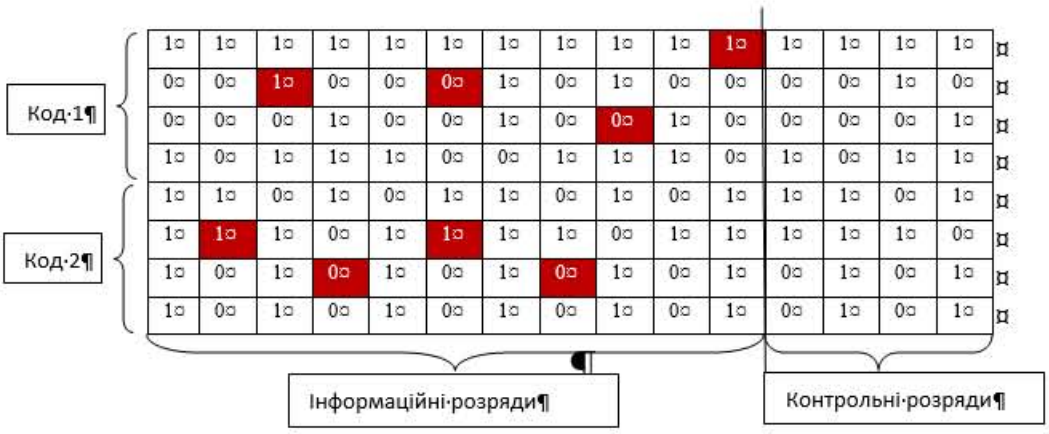


Рисунок 3.7 – Виправлення помилок кодом Ріда-Соломона

Для виправлення 8 помилкових біт знадобилось два декодера для прик

ладу щоб виправити 8 помилкових біт кодомХемінга (15,11) нам знадобиться 8 декодорів що і видно на малюнку 3.8.

Рисунок 3.8 – Виправлення помилок кодом Хемінга

В даному розділі було показано апаратну реалізацію модифікованого коду Елайеса та модифікованого коду ріда-Соломона, описано принципи роботи даних кодів, описано алгоритм роботи модифікованого коду Елайеса. Зроблено порівняльну характеристику модифікованого коду Елайеса з модифікованим кодом Ріда-Соломона, а також модифікованого окду Ріда-Соломона з кодом Хемніга.

4. ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Результатом магістерської кваліфікаційної роботи "засобів покращення відмовостійкості напівпровідникової пам'яті" є розробка методів контролю напівпровідникової пам'яті. Для проведення технологічного аудиту залучено трьох незалежних експертів. Оцінювання комерційного потенціалу буде здійснене за критеріями, що наведені в таблиці 4.1.

Таблиця 4.2 - Результати оцінювання комерційного потенціалу розробки

Критерії	Експерти		
	1	2	3
	Бали, виставлені експертами:		
1	3	2	2
Ринкові переваги (недоліки):			
2	2	2	3
3	3	4	3
4	4	3	3
5	3	4	3
Ринкові перспективи			
6	3	4	4
7	3	4	3
Практична здійсненність			
8	4	4	3
9	3	3	4
10	2	4	3
11	3	4	2
12	4	4	4
Сума балів	$СБ_1$=37	$СБ_2$=42	$СБ_3$=37
Сума	38		

За даними таблиці 4.2 можна зробити висновок, щодо рівня комерційного потенціалу розробки. Зважимо на результат й порівняємо його з рівнями комерційного потенціалу розробки, що представлено в таблиці 4.3.

Таблиця 4.3 - Рівні комерційного потенціалу розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Рівень комерційного потенціалу розробки, становить 38 балів, що відповідає рівню «вище середнього».

Проаналізуємо суть технічної проблеми та розглянемо аналоги.

Напівпровідникові запам’ятовуючі пристрої пам’яті (НЗП) реалізується на одній чи кількох напівпровідникових схемах. При роботі з такою пам’яттю можлива поява помилок. Для того щоб захистити пам’ять від помилок можна застосовувати циклічні та ітеративні коди які прості в реалізації і при невисокій надлишковості мають хороші властивості виявлення спотворень.

В якості аналога для розробки було обрано ECC-пам’ять. ECC-пам'ять використовується в більшості комп'ютерів, на яких пошкодження даних не може бути допущено ні за яких обставин, наприклад, для наукових або фінансових обчислень. Найчастіше використовують алгоритм корекції помилок на основі коду Хеммінга.

Основними недоліками аналога є те що даний контроль пам’яті здатний виправляти одиночні помилки. У розробці використовуються ітеративні коди Елайєса та CRC-коди які дозволяють знаходити та випраляти до 3 помилок.

У таблиці 4.4 наведені основні технічні показники аналога і нового програмного продукту.

Таблиця 4.4 - Основні технічні показники аналога і нового програмного продукту

Показники	Аналог	Нова розробка	Відношення параметрів нової розробки до параметрів аналога
Довжина коду	середня	велика	краще
Складність декодера	мала	середня	гірше
d_{min}	3	5	краще
Виправлення пакетів помилок	немає	присутнє	краще
Ітеративність	немає	присутня	краще

Ітеративні кодиявляють собою, як правило, двомірні лінійні коди з кодуванням рядків і стовпців завадостійкими кодами з перевіркою на парність . Такі ітеративні коди мають мінімальну кодову відстань d_{min} = 5 і дозволяють виправити коди з числом перевірочних елементів 8, 9 та 16. При виправленні помилки у декодері визначають рядок і стовпець, для яких не виконуються умови парності.

Передачу символів такого коду зазвичай здійснюють послідовно символ за символом, від одного рядка до іншого, або паралельно цілими рядками. Декодування починають відразу, не чекаючи надходження всього блоку інформації. Перевірка рівнянь при декодуванні дозволяє виявити будь-непарне число спотворених символів, розташованих в одному рядку або в одному стовпці. Перевірка одиночної помилки по рядку, вказує на наявність помилки в ній, а перевірка по стовпцю - конкретний символ. Для виправлення помилок які знаходяться в одному стовпці або рядку використовують CRC-коди для виявлення «скритих» помилок.

В даний час продукт знаходиться на фінальній стадії розробки, тобто готовий алгоритм модифікованого коду Елайеса. Даний алгоритм можна

використовувати в блоках контролю напівпровідникової пам'яті які вбудовані в пам'ять домашнього комп'ютера, та компаніях які використовують напівпровідникові пристрої пам'яті.

В перспективі використання модифікованого коду Елайеса можливо в космічній сфері для захисту пам'яті від впливу радіації.

Для реклами продукту пропонується створення сайту з інформацією про контроль напівпровідникових пристроїв пам'яті та представлення готового алгоритму та програми для блоків контролю пам'яті.

4.2 Прогнозування витрат на виконання науково-дослідної, та конструкторської технологічної роботи

Для розробки нового програмного продукту необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою (4.1):

$$З_о = \frac{М}{Т_р} \cdot t, \quad (4.1)$$

де М- місячний посадовий оклад конкретного розробника;

$Т_р$ - кількість робочих днів у місяці, $Т_р$ = (21…23) дні;

t - число днів роботи розробника

Зроблені розрахунки бажано звести до таблиці 4.2.

Таблиця 4.2 - Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	7000	304	3	912
Інженер-програміст	5500	296	40	11868
Всього:				12780

Додаткова заробітна плата $З_д$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$З_д = (0{,}1...0{,}12) \cdot З_о . \quad (4.2)$$

В даному випадку

$$З_д = 0.1 \cdot 12780 = 1278 (грн).$$

Нарахування на заробітну плату операторів $Н_{зп}$ розраховується як 22% від суми їхньої основної та додаткової заробітної плати:

$$Н_{зп} = (З_о + З_р) \cdot \frac{\beta}{100}, \quad (4.3)$$

$$Н_{зп} = (12780+1278) \cdot 0{,}22 = 3092{,}76 \text{ (грн.)}.$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12}, \quad (4.4)$$

де Ц – балансова вартість обладнання, грн;

$Н_а$ – річна норма амортизаційних відрахувань % (для нашого випадку 10…25%);

Т – Термін використання ,місяці.

Таблиця 4.3 - Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань,грн
Персональний комп'ютер	10000	25	3	625
Всього:				625

Також потрібно врахувати витрати на доступ до мережі Інтернет, що використовувався під час виконання роботи.

Витрати за доступ до Інтернет можна розрухувати за формулою:

$В_{ді}=Ц_{ді}·Т$ [грн],

де Цді – це ціна доступу за місяць;

Т – кількість місяців використання доступу до мережі.

Отже, витрати на доступ до мережі Інтернет становлять:

$В_{ді}$ = 150 2 = 300 (грн).

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$K = \sum_1^n Н_і \cdot Ц_і \cdot К_і, \quad (4.5)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 4.4 - Витрати на комплектуючі, що були використані для розробки методу

Для розробки алгоритму контролю напівпровідникової пам'яті використовувалась оперативна пам'ять ціною 860 грн.

Витрати на силову електроенергію розраховуються за формулою:

$$В_е=В\cdot П\cdot Ф\cdot К_п ; \quad (4.6)$$

де В – вартість 1кВт-години електроенергії (В=2,5 грн/кВт);

П – установлена потужність комп'ютера (П=0,6кВт);

Ф – фактична кількість годин роботи комп'ютера;

$К_п$ – коефіцієнт використання потужності ($К_п < 1$, $К_п = 0,85$).

$$В_е= 2,5 \cdot 0,6 \cdot 240 \cdot 0,85 = 306 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$В_{ін} = (1..3) \cdot (З_о + З_р). \quad (4.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 \cdot (12780+1278) = 14058 \text{ (грн).}$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в$$

$$В = 12780+1278+3092,76+625+300+860+306 +14058= 33299,71 \text{ (грн.)}$$

Розрахуємо загальну вартість наукової роботи $В_{заг}$ за формулою:

$$В_{заг} = \frac{В_{ін}}{\alpha} \quad (4.7)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{33005,21}{1} = 33005,21$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta} \quad (4.8)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{33005,21}{0,8} = 41256,51 \text{ (грн.)}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки.

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \Delta N)_i \quad (4.9)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться на 25 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 25 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 200 користувачів, протягом другого року – на 175 користувачів, протягом третього року – 150 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 1 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 200 грн.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_1 = 25 \cdot 1 + (200 + 25) \cdot 200 = 45025 \text{ грн.}$$

Протягом другого року:

$$\Delta П_2 = 25 \cdot 1 + (200 + 25) \cdot (200 + 175) = 84400 \text{ грн.}$$

Протягом третього року:

$$\Delta П_3 = 25 \cdot 1 + (200 + 25) \cdot (200 + 175 + 150) = 34150 \text{ грн.}$$

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Рисунок 4.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t} \qquad (4.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

τ – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{45025}{(1+0,1)^0} + \frac{84400}{(1+0,1)^2} + \frac{43150}{(1+0,1)^3} = 147196,29(грн.)$$

Тоді розрахуємо $E_{абс}$:

$$Е_{абс} = 147196,29 - 41256,51 = 105939,78 \text{ грн.}$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою:

$$Е_в = \sqrt[Т]{1 + \frac{Е_{абс}}{PV}} - 1 \qquad (4.11)$$

де $Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$Е_в = \sqrt[3]{1 + \frac{105939,78}{41256,51}} - 1 = 0,44 \text{ або } 44\ \%$$

Далі, розраховану величина $Е_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть.

$$\tau = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0{,}2 + 0{,}1 = 0{,}3$$

Оскільки $E_в = 44\% > \tau_{мін} = 0{,}3 = 30\%$, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}$$

$$T_{ок} = \frac{1}{0{,}44} = 2{,}27 \text{ року}$$

4.5 Висновок до розділу

У четвертому розділі було проведено розрахунок ефективності вкладених інвестицій та періоду їх окупності. Приведена вартість всіх чистих прибутків склала 45025 грн. Абсолютна ефективність вкладених інвестицій – 105939,78грн. Було обраховано відносну ефективність вкладених коштів в наукову розробку інвестицій. Вона складає 44%. Оскільки вона менша за мінімальну ставку дисконтування, що складає 30%, інвестор будезацікавлений у фінансуванні даної наукової розробки. Також, оскільки термін окупності складає 2,27 роки, що є меншим за три роки, фінансування такої розробки є доцільним.

ВИСНОВОК

В магістерській кваліфікаційні роботі проведено аналіз напівпровідникових пристроїв пам'яті, помилок які можуть в ній виникнути, а також завадостійких кодів. Обгрунтовано вибір завадостійких кодів для напівпровідникової пам'яті. Проаналізовано завадостійкі коди Хемінга, байт-коди, коди Елайеса, Ріда-Соломона. Запропоновано модифікований код Елайеса та модифікований код Ріда-Соломона для декодування помилок в напівпровідниковій пам'яті. Розглянуто декодування кодів Ріда-Соломона.

Показано апаратну реалізацію модифікованого коду Елайеса та Ріда-Соломона. Побудовано алгоритм декодування помилок методом модифікованого коду Елайеса.

В економічній частині було розраховано необхідні витрати на розробку, доведено доцільність розробки та її фінансування.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. A. Neale and M. Sachdev, \A New SEC–DED Error Correction Code Subclass for Adjacent MBU Tolerance in Embedded Memory," Device and Materials Reliability, IEEE Transactions on , vol.13, no.1, pp.223,230, March 2013.

2. Конопелько В. К. Анализ возможности применения БЧХ кодов для коррекции зависимыхошибок / В. К. Конопелько, О. Г. Смо– лякова, А. В. Шкиленок // Доклады БГУИР. – 2007. – № 5. – С. 17–22.

3. E. Fujiwara. Code Design for Dependable Systems – Theory and Practical Applications. John Wiley and Sons, Inc., 2006.

4. Metzner J. J. On Correcting Bursts (and Random Errors) in Vector Symbol (n, k) Cyclic Codes / J. J. Metzner // IEEE Trans. Inform. Theory. – April, 2008. – Vol. 54. – No. 4. – P. 1795–1807.

5. Bossert M.j Sidorenko V. Singleton–Type Bounds for Blot–Correcting Codes / / IEEE Trans. Inform. Theory. 1996. V. 42. №. 3. P. 1021–1023

6. P.E. Dodd and L.W. Massengill. Basic Mechanisms and Modeling of Single–Event Upset in Digital Microelectronics. Nuclear Science, IEEE Transactions on, 50(3):583– 602, June 2003.

7. Семеренко В. П. Войналович О. Ю. Оцінка коректувальної здатності

ітеративних завадостійких кодів. IV Міжнародна науково-практична конференція "Потенціал сучасної науки" м. Київ, 10-11грудня 2019 р.

8. Стівен В. Міллер. Пам'ять і технології зберігання. — Монтвейл: AFIPS Press, 1977. — ISBN.

9. Семеренко В. П. Теорія циклічних кодів на основі автоматних моделей : монографія [Текст] / В. П. Семеренко. – Вінниця : ВНТУ, 2015. – 444 с.

10. Конопелько В. К. Надежное хранение информации в полупроводниковых запоминающих устройствах / В. К. Конопелько, В. В. Лосев. — М. : Радио и связь, 1986. — 240 с.

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

12. Блейхут Р. Теория и практика кодов, контролирующих ошибки = Theory and Practice of Error Control Codes. — М.: Мир, 1986. — 576 с.

13. Хемминг Р. В. Теория кодирования и теория информации: Пер. с англ. – М.: Радио и связь, 1983. – 176 с., ил

14. Боссерт М., Брайтбах М., Зяблов В. В., Сидорен– ко В. Р. Коды, исправляющие множество пятен оши– бок или стираний // Проблемы передачи информа– ции, 1997, т. 33, № 4, с. 15–25.

15. Metzner J. J. On Correcting Bursts (and Random Errors) in Vector Symbol (n, k) Cyclic Codes / J. J. Metzner // IEEE Trans. Inform. Theory. – April, 2008. – Vol. 54. – No. 4. – P. 1795–1807.

16. Бородин ГА., Иванов В А. Методы расчета надежности запоминающих устройств // Зарубежная радиоэлектроника. 2003. № 2. С. 92–111.

17. Блейхут Р. Теория и практика кодов, контролирующих ошибки / Блейхут Р. – М.: Мир, 1986. – 576 с.

18. Bossert M.j Sidorenko V. Singleton–Type Bounds for Blot–Correcting Codes / / IEEE Trans. Inform. Theory. 1996. V. 42. №. 3. P. 1021–1023

19. Сагалович Ю. Л. Об одном свойстве кода Хэмминга // Проблемы передачи информации. 1988. Т. 24. № 1. С. 102–105.

20. P.E. Dodd and L.W. Massengill. Basic Mechanisms and Modeling of Single–Event Upset in Digital Microelectronics. Nuclear Science, IEEE Transactions on, 50(3):583– 602, June 2003.