НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

«На правах рукопису»
УДК __004.033__

До захисту допущено:
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
«___»_____202__ р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-науковою програмою «Системне програмування та спеціалізовані комп'ютерні системи»

зі спеціальності 123 «Комп'ютерна інженерія»

на тему: «Спосіб відновлення блоків даних при віддаленому розподіленому зберіганні»

Виконав:
студент II курсу, групи КВ-01мн
Антонюк Антон _____

Науковий керівник:
ст. вик. каф. СПСКС, к.т.н.
Коляда Костянтин Вячеславович _____

Консультант з нормоконтролю:
Доцент кафедри СПСКС, к.т.н., с.н.с
Боярінова Юлія Євгенівна _____

Рецензент:
_____
_____ _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2022 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра системного програмування

і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – другий (магістерський)

Спеціальність – 123 «Комп‘ютерна інженерія»

Освітньо-наукова програма «Системне програмування та спеціалізовані комп‘ютерні системи»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ

«___»_____202__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Антонюку Антону Андрійовичу

1. Тема дисертації «Спосіб відновлення блоків даних при віддаленому розподіленому зберіганні», науковий керівник дисертації Коляда Костянтин Вячеславович, старший викладач кафедри СПСКС, кандидат технічних наук затверджені наказом по університету від «05»_травня___ 2022 р. №НС/201/2022

2. Термін подання студентом дисертації 10.05.2021

3. Об’єкт дослідження – вивчення методів відновлення даних на віддалених сховищах та застосування цих методів у системах розподіленого зберігання інформації.

4. Вихідні дані: розроблений спосіб відновлення даних в віддалених розподілених системах зберігання інформації, поведінкова модель системи відновлення даних.

5. Перелік завдань, які потрібно розробити: аналіз існуючих методів та заходів резервування та відновлення даних при віддаленому розподіленому зберіганні; створення способу відновлення даних на основі одного з існуючих методів.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу – презентація.

7. Орієнтовний перелік публікацій: тези опубліковані у збірнику VII міжнародної науково-практичної конференції «Актуальні питання сучасності

наука, суспільство і освіта» та стаття опублікована в журналі «Вісник Хмельницького національного університету» серія «Технічні науки» №3 2022.

8. Дата видачі завдання 09.10.2020

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Вивчення матеріалів за темою дисертації	06.12.2021	
2	Підготовка матеріалів першого розділу	21.01.2022	
3	Підготовка матеріалів другого розділу	21.02.2022	
4	Розробка способу відновлення даних за допомогою обраного методу	18.02.2022	
5	Розробка поведінкової моделі системи відновлення даних при віддаленому розподіленому зберіганні	25.03.2022	
6	Підготовка матеріалів третього розділу	14.04.2022	
7	Симуляція роботи системи та аналіз отриманих результатів	10.05.2022	
8	Підготовка матеріалів четвертого розділу	20.05.2022	
9	Попередній розгляд магістерської дисертації на кафедрі	27.05.2022	

Студент Антон Антонюк

Науковий керівник Костянтин Коляда

РЕФЕРАТ

Актуальність теми. При зберіганні даних можуть виникати проблеми як технічного так і катастрофи техногенного чи природного походження, які не можуть бути передбачені заздалегідь. В сучасному світі зменшення ризиків втрати інформації забезпечується її розділеним зберіганням, що є природнім і зрозумілим. Актуальним залишається питання застосування методів забезпечення резервування та відновлення інформації в системах розподіленого збереження інформації. Існує багато розроблених та застосованих методів вирішення вищеописаних проблем на різних рівнях: на рівні логічних масивів (розділів одного носія), на рівні носіїв інформації. Однак, проблема втрати одного з декількох серверів з даними всеодно може спричинити загрозу цілісності та повноти інформації. З огляду на ситуацію в світі та нові військові конфлікти, виникає гостра необхідність у створенні рішення для подолання цих проблем.

Мета роботи: створення нового способу відновлення блоків даних, при їх віддаленому розподіленому зберіганні за одним із відомих методів.

Об'єктом дослідження є вивчення методів відновлення даних на віддалених сховищах та застосування цих методів у системах розподіленого зберігання інформації.

Предметом дослідження є способи використання різних методів для систем відновлення блоків даних при їх віддаленому розподіленому зберіганні.

Методи дослідження. Для досягнення поставленої мети було використано наступні наукові методи:

1. абстрагування – виокремлено проблематику резервування даних при віддаленому розподіленому зберіганні;
2. аналіз і синтез – було проаналізовано існуючі методи вирішення проблеми резервування даних та було синтезовано новий спосіб з використанням одного з проаналізованих методів;

3. моделювання – було створено поведінкову модель та проведено досліди з її роботи.

Наукова новизна одержаних результатів полягає в наступному:

1. запропоновано новий спосіб відновлення блоків даних, , що вирішує проблеми третього рівня розподіленого зберігання даних;
2. дістала подальшого розвитку поведінкова модель системи, що використовує різні способи відновлення блоків даних, при їх віддаленому розподіленому зберіганні.

Практичне значення одержаних результатів полягає в тому, що запропонований спосіб в системах відновлення блоків даних дозволяє забезпечити захист від втрати даних при локальних природніх чи техногенних катастрофах, що доведено практичним використанням модифікованої поведінкової HDL-моделі.

Особистий внесок магістранта полягає в теоретичному обґрунтуванні одержаних результатів, експериментальній їх перевірці та дослідженні на базі HDL-моделювання. Всі результати, що наведені в дисертації, отримані автором самостійно.

Апробація результатів дисертації.

Загальні положення та пропозиція застосування розробленої системи опубліковані у збірнику VII міжнародної науково-практичної конференції «Актуальні питання сучасності наука, суспільство і освіта».

Публікації.

Опис системи із застосуванням розробленого способу та приклади роботи системи було опубліковано в статті «Поведінкова модель системи відновлення блоків даних при їх віддаленому розподіленому зберіганні» в журналі «Вісник Хмельницького національного університету» серія «Технічні науки» №3 2022.

Структура та обсяг дисертації.

Магістерська дисертація складається чотирьох розділів, висновків по кожному розділу та загальних висновків по роботі в цілому, списку використаних літературних джерел (25 найменувань).

У вступі сформульована актуальність досліджень, мета і задачі досліджень.

У першому розділі подано загальну інформацію щодо напрямку досліджень та актуального стану проблематики.

У другому розділі проаналізовано існуючі методи резервування їх переваги та недоліки та обрано оптимальний метод для використання його в запропонованій системі.

У третьому розділі наведено опис моделі використання запропонованого способу для системи відновлення інформації на мові VHDL.

У четвертому розділі проводиться аналіз роботи моделі, що заснована на запропонованому способі із використанням обраного методу відновлення даних на віддалених сховищах, та наведено результати роботи симуляції системи.

У висновках представлені результати роботи над темою магістерської дисертації.

Повний обсяг дисертації – 128 сторінок, у тому числі 81 сторінок основного тексту, містить 3 додатки, 25 джерел, 49 рисунків, 1 таблицю.

Ключові слова: системи віддаленого розподіленого зберігання даних, резервування даних, система відновлення даних, хмарні технології.

ABSTRACT

Actuality of theme. Data storage can cause both technical and man-made or natural disasters that cannot be foreseen. In today's world, reducing the risk of information loss is ensured by its separate storage, which is natural and understandable. The issue of the application of methods of ensuring backup and recovery of information in distributed information storage systems remains relevant. There are many developed and applied methods for solving the above problems at different levels: at the level of logical arrays (sections of one medium), and the level of media. However, the problem of losing one of several data servers can still compromise the integrity and completeness of the information. Given the world situation and new military conflicts, there is an urgent need to find solutions to overcome these problems.

Purpose: to create a new way to recover data blocks, with their remote distributed storage by one of the known methods.

The object of research is to study the methods of data recovery in remote storage and the application of these methods in distributed storage systems.

The subject of the study is ways to use different methods for data block recovery systems for their remote distributed storage.

Research methods

The following scientific methods were used to achieve this goal:

1. Abstraction - the issue of data backup in remotely distributed storage;
2. Analysis and synthesis - the existing methods of solving the problem of data backup were analyzed and a new method was synthesized using one of the analyzed methods;
3. Modeling - a behavioral model was created, and experiments were conducted on its work.

The scientific novelty of the obtained results is as follows:

1. A new method of restoring data blocks is proposed, which solves the problems of the third level of distributed data storage;

2. Behaved further behavioral model of the system, which uses different ways to recover data blocks, with their remote distributed storage.

The practical significance of the obtained results is that the proposed method in data block recovery systems allows protection against data loss in local natural or man-made disasters, as evidenced by the practical use of a modified behavioral HDL model.

The personal contribution of the master consists in the theoretical substantiation of the obtained results, their experimental verification, and research based on HDL modeling. The author himself obtained all the results presented in the dissertation.

Approbation of dissertation results.

General provisions and proposals for the application of the developed system are published in the Proceedings of the VII International Scientific and Practical Conference "Current Issues of Science, Society, and Education".

Publications.

A description of the system using the developed method and examples of system operation was published in the article "Behavioral model of data block recovery system for their remote distributed storage" in the journal "Bulletin of Khmelnytsky National University" series "Technical Science" №3 2022.

Structure and scope of the dissertation. The master's dissertation consists of four chapters, conclusions on each section and general conclusions on the work as a whole, a list of used literature sources (25 titles).

The *introduction* formulates the relevance of research, the purpose and objectives of research.

The *first section* provides general information on the direction of research and the current state of the issue.

The *second section* analyzes the existing methods of redundancy, their advantages and disadvantages and selects the best method for its use in the proposed system.

The *third section* describes the model of using the proposed method for the information recovery system in VHDL.

The *fourth section* analyzes the operation of the model based on the proposed method using the selected method of data recovery in remote storage. It presents the results of the system simulation.

The *conclusions* present the work results on the master's dissertation topic.

The full volume of the dissertation – 128 pages, including 81 pages of the main text, contains 3 appendices, 25 sources, 49 figures, 1 table.

Keywords: remote distributed data storage systems, data backup, data recovery system, cloud technologies.

ЗМІСТ

СПИСОК ТЕРІМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ЗП – запам’ятовуючий пристрій;

Кластер, вузол даних чи інформації – об’єднання носіїв пристроїв за будь-яким принципом (наприклад географічним, прикладом є дата центр);

Носій інформації – пристрій на якому зберігається інформація;

ПЗ – програмне забезпечення;

Симуляція – в Active-HDL це можливість перевірки створеного HDL-коду за допомогою комп’ютерного модулювання для різних вхідних даних, зокрема реальних, також існує можливість відслідковувати роботу симуляції по тайм кодах;

Фрагментація – це розділення одного цільного об’єму інформації на фрагменти, які будуть зберігатися окремо;

Хмарні сервіси – сервіси, що надають можливість віддаленого збереження даних та обчислення на віддалених ЕОМ;

Active-HDL – це середовище розробки та моделювання проєктів для програмованих логічних інтегральних схем, створене компанією Aldec;

XOR (виключне або) – логічна операція результатом якої є логічна одиниця лише коли відрізняються операнди.

ВСТУП

Кожного дня розвиток технологій невпинно набирає свої оберти, що дозволяє з кожним днем робити все більш складні технології, більше того, про які вчора не можна було навіть мріяти. Задля новітньої, інноваційної розробки будь-чого ще більш складнішого з кожним разом використовується ще більше накоплених людством даних, що в свою чергу ставить питання про зберігання та передачу накопичених знань.

За останні десятки років відбувся інформаційний вибух, адже зберігання даних на цифрових носіях тепер було доступно для кожної людини. Зберігання даних локально не дозволяло захистити важливу інформацію від її втрати, адже можуть виникнути різні проблеми, захистити від яких можуть лише новітні технології. Окрім того, доступ до інформації є важливим чинником, тому що якщо важливі дані збережені локально, тоді виникає проблема завжди мати з собою носій цих даних. Розвиток глобальної мережі Інтернет та швидкості передачі інформації дозволив вирішити всі ці питання появою хмарних сервісів.

Кожна людина може зберігати свої дані у хмарі, адже по суті це є віддаленою розподіленою мережею збереження даних, доступ до якої може бути з будь-якої точки світу, з будь-якого пристрою, умовою є лише стабільний доступ в Інтернет [1]. В цьому випадку вирішенням проблем забезпечення цілісності даних та обчислень займаються професіонали, на яких користувачі покладають надію, адже не мають більше жодного впливу на процес зберігання даних. Звичайно, загрози технічного, природного чи техногенного характеру залишаються, однак можливості їх вирішення істотно збільшуються.

Реалізація системи зберігання даних вирішується безпосередньо тими хто ці послуги надає. З часом з використанням загальнолюдського досвіду було прийнято рішення використовувати системи розподіленого зберігання даних, задля того, щоб зменшити потенційні ризики втрати всієї інформації одномоментно. Цілком логічно, якщо зберігати інформацію в одному місці, то

будь-яка зовнішня загроза, як війна, природній катаклізм, тощо, може спричинити втрату всіх даних, що є неприпустимим. Тим не менш, ризик втрати навіть частини даних є небажаним, саме тому залишається активним питання резервування та відновлення даних з систем віддаленого розділеного зберігання даних.

З кожним роком новітні дослідження просувають ефективність відновлення інформації вперед. Результатом цього є низка заходів, які ефективно працюють в рамках одного сховища даних, однак, питання резервування та відновлення даних в рамках всього вузла є не вирішеним.

Отже, метою даної дисертації є підвищення ефективності резервування та відновлення даних, шляхом створення способу відновлення даних в віддалених розподілених системах збереження інформації.

РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМАТИКИ ВІДНОВЛЕННЯ ДАНИХ ПРИ ВІДДАЛЕНОМУ ЗБЕРІГАННІ

Будь-яка система збереження інформації зустрічається з проблеми завадостійкості: здатності ідентифікації, локалізації помилок та здатності відновлення втрачених даних не нова, і нас сьогодні існує певна кількість засобів, які розроблялися з метою вирішення цих проблем, адже забезпечення цілісності даних – є головною метою.

Цілісність даних – це поняття, під яким розуміється збереження точності та узгодженості інформації під час їх життєвого циклу. Цілісність даних забезпечує захист від втрат та пошкодження, викликаних апаратним, програмним забезпеченням або збоєм мережі [2]. Окрім цілісності даних в хмарних сервісах також можуть бути використані засоби задля збереження цілісності обчислень. Цілісність обчислень – це забезпечення захисту процесу виконання програм без спотворень шкідливими програмами, постачальниками хмарних сервісів чи іншими користувачами, і гарантія того, що будь-які неправильні обчислення будуть виявлені [3].

1.1 Аналіз проблеми надійності зберігання інформації при віддаленому зберіганні

Попри наявність низки методів резервування та відновлення втраченої інформації тема залишається актуальною та важливою з огляду на особливості динамічного розвитку інформаційних технологій, адже пошук оптимального рішення завжди буде на часі. Будь-який метод відновлення втрачених блоків даних можна характеризувати декількома параметрами:

- кількість відновлюваної інформації;
- надлишковість;
- обчислювальна складність процедури відновлення даних з використанням резервних носіїв.

Цілком зрозуміло, що вищенаведені параметри будуть перебувати в контр-позиції, адже при використанні простих методів зі збільшенням кількості відновлюваних даних ми збільшуємо кількість резервних блоків, що збільшує надлишковість, швидкість при цьому буде великою. Однак, за використання складних обчислень, що дозволяють відновити відносно велику кількість втрачених даних за відносно малої надлишковості, ми втрачаємо в швидкості вирішення цієї задачі, через складні математичні обчислення.

Інформація, що зберігається на будь-яких засобах збереження даних може бути втрачена, або втрачена цілісність даних в результаті дій наступних чинників:

- вірусні чи хакерські атаки;
- непрогнозовані помилки в роботі ПЗ;
- відмови технічного характеру ЗП;
- катаклізми різного характеру.

Незалежно від виду ЗП, себто на магнітних, твердотільних чи оптичних пристроях може виникати кожна з вищезазначених помилок, що призведе до втрати даних. Можуть бути втраченими як окремі блоки даних, які, як правило, відновлюються, якщо це можливо, безпосередньо на сховищі, або втрата даних на кластері цілком. Саме через наявні чинники виникла ідея рознесення даних та їх фрагментація з метою мінімізації втрачених даних.

Завдяки останнім досягненням в технологіях широко почали використовуватись розподілене зберігання даних на віддалених сховищах [4]. Практика надання простору на хмарному сервісі для збереження інформації завдяки переходу на комерційну основу дає наступні переваги:

- необмежений простір для зберігання даних – там де є попит є й пропозиція, тому за додатковий кошт, який сплачує низка користувачів компанія має можливість розширення наявного цифрового простору;

- фаховий підхід до забезпечення цілісності даних – профільні інженери та інші спеціалісти будуть працювати, щоб збільшувати надійність систем:
    - створювати захист від атак вірусів на рівні вузлів;
    - імплементація новітніх методів резервування та відновлення інформації на рівні кластерів.
- дослідження для поліпшення резервування даних – компанії зацікавлені в створенні додаткової вартості на свої послуги, а тому будуть інвестувати в розробку інноваційних систем резервування даних;
- дослідження для поліпшення передачі даних – компанії зацікавлені в швидкій передачі інформації, задля покращення користувацього досвіду, а тому інвестуватимуть в дослідження.

Зрозуміло, що окрім безпосередньої втрати даних вузол збереження даних може бути недоступний і з інших причин, зокрема:

- завантаженість великою кількістю запитів;
- оновлення чи зміна ПЗ;
- ремонтні чи профілактичні роботи.

Тим не менш, існують інструменти для маніпулювання цими проблемами та їх вирішення. Як правило, компанія є зацікавленою в збільшенні та збереженні кількості свої користувачів, а тому вона попереджає заздалегідь про ті чи інші чинники, які залежать від неї, адже прагне зберегти довіру до свого сервісу. Окрім того, існує практика проводити сервісні роботи над розподіленими сховищами в найменш активні години користування ними, щоб мінімізувати вплив на безпосереднього користувача.

Через вище зазначені чинники, останнім часом, можна спостерігати стійке збільшення попиту на хмарні сервіси та хмарного простору збереження інформації. Користувачі надають перевагу хмарним технологіям через те, що вони надають можливість зручно та швидко обмінюватись та розповсюджувати

інформацію, окрім того збереження даних в хмарі дає більшу ймовірність їх збереження.

Найбільш розповсюдженою ситуацією є втрата даних на одному з носіїв інформації, через такі причини, як виходу їх з ладу, чи стирання інформації на них [5]. Тому на даному етапі технологічного розвитку підвищення надійності віддаленого зберігання інформації, ключовим фактором є багаторівневість. Можна виділити наступні рівні:

- бінарний масив – на цьому рівні виконується локалізація та виправлення помилок [6] безпосередньо в запам’ятовуючих пристроях;
- масив накопичувачів даних – цей рівень характеризується, як правило, використанням систем Intermemory[7] та RAID [8].

З огляду на сучасний світ залишається ключовим питанням можливість втратити повністю вузол інформації внаслідок будь-якого місцевого катаклізму, бойових дій, тощо, що призведе до повної втрати частини даних, або ж втрати доступу до них. Задля того, щоб користувач не відчув втрати зв’язку з певним сервером даних через будь-яку причину, необхідно забезпечити можливість відновлення інформації на рівні вузлів даних. Адже це єдиний чинник, який здатний на практиці убезпечити захист від втрат інформації. Хоч і сучасні технології передачі інформації через мережу Інтернет забезпечують відносно швидкий доступ до віддаленого сховища даних, що дозволяє повною мірою використовувати можливості розподілених систем, але існує негативний момент в використанні такого підходу – через збирання необхідних даних з декількох вузлів існує затримка в часі.

Для розуміння проблеми виправлення помилок необхідно розглянути більш детально існуючі засоби резервування та відновлення даних, а також їх практичне застосування.

Рівень бінарного масиву, або ж перший рівень забезпечення надійності зберігання інформації характеризується засобами, що працюють в рамках секторів магнітних чи оптичних ЗП. Коли відбувається форматування

відбувається контрольне зчитування, якщо було спотворено 2 і більше байтів, то запис в сектор блокується. На цьому рівні використовуються корегуючі коди, які, як правило, виконуються відновлення даних в два етапи: локалізація та відновлення спотворених бітів. На практиці використовується код Ріда-Соломона (512, 4) [9], який забезпечує можливість відновлення двох байтів, або виявлення помилки більшої кратності.

На другому рівні використовується резервні накопичувачі. На практиці широко застосовується система RAID. Початково створена для прискорення роботи з недорогими носіями, її розвиток був продовжений в напрямку резервування та відновлення даних, що породило цілу низку версій пристроїв, що мали передові на той час засоби відновлення інформації. Загалом вирізняються такі версії системи RAID [10]:

- RAID-1 – резервування забезпечувалось повним дублюванням даних, що забезпечувало максимальну швидкість відновлення;
- RAID-2 – використовувався один накопичувач, інформація на якому була сформована кодом Хемінга [11];
- RAID-3 та RAID-4– використовувався один накопичувач в якому були сформовані дані функцією XOR всіх інших накопичувачів;
- RAID-5 – принцип в розподіленні інформаційних блоків та резервних по n+1 накопичувачах, резервні блоки формуються функцією XOR;
- RAID-6 – використовує n+2 накопичувачі, та здатна відновлювати інформацію з будь-яких двох втрачених блоків.

В сучасних системах RAID використовуються MDS-коди [12].

Попри значний досвід використання RAID технології не є ефективним до flash-носіїв, які набули значного поширення в останній час. Це зумовлено тим, що твердотільні пристрої, хоч і мають значно більшу швидкість роботи, однак мають менший час збереження інформації та залежать від кількості циклів перезапису даних [10].

Використання додаткового рівня резервування інформації має сенс, ґрунтуючись на попередні твердження, адже це можливість уникнути втрати інформації на кластері даних цілком – чого не передбачає архітектура RAID, зокрема. Окрім того, звертаючи увагу на те, що твердотільним носіям необхідна час від часу процедура поновлення даних, у разі їх втрати, було б логічно в рамках запланованого сервісного обслуговування, або в разі втрати даних робити відповідні поновлення, для всього вузла інформації. У випадку локального катаклізму це також зможе уберегти інформацію від втрати.

Наразі відома низка системи віддаленого зберігання даних, які працюють на потенційно ненадійних кластерах даних. Зокрема до таких систем відносять: Facebook‘s codes Hadoop, Google Colossus, Microsoft Azure [13]. В той час як ті самі архітектури RAID прогресували з часом, теперішній час резервування даних на рівні вузлів інформації залишився на рівні RAID-1 – себто повного дублювання вузла. Це є вкрай неефективним з огляду на постійний приріст нових користувачів, а тому постає питання в використанні нової системи на основі роботи відновлюючих кодів для забезпечення оптимального рівня резервування та відновлення інформації.

Отже, виникає потреба в модифікації існуючих систем резервування та відновлення інформації в віддалених розподілених системах зберігання даних. Вирішенням цієї проблеми може бути новий спосіб, який можна буде імплементувати до існуючих розподілених систем, з такими ключовими завданнями:

1) можливість відновлення даних на кластері інформації при втраті доступу до нього чи втраті його фізично;
2) можливість оперативного відновлення даних «на льоту» задля того, щоб надати користувачу необхідну йому інформацію, якщо це можливо, до того як буде проведено сервісне обслуговування та відновлення даних;
3) знизити надлишковість існуючої системи.

1.2 Огляд наявних технологій резервування та відновлення втрачених даних

В основі розглянутих існуючих методів резервування та відновлення інформації, є застосування MDS-кодів. Це застосовується для того, щоб в розподілених системах зберігання даних досягти максимально можливого теоретично можливості відновлення інформації, при мінімальному обсязі надлишкових блоків даних [14].

MDS (Maximum Distance Separable)-коди є в основі будь-якого з відмовостійких блочних кодів.

В блочних кодах кодові слова є послідовністю символів довжини з n розрядів, що має назву блок ($n = k + r$).

В двійкових блокових кодах будь-який розряд слова може приймати одне з двох значень: 0 або 1.

В лінійних двійкових кодах контрольні розряди можна вибрати таким чином, щоб кожний з них утворювався функцією XOR певних інформаційних розрядів. При виявленні і виправленні помилок по значенню інформаційних розрядів визначаються поточні значення резервних розрядів, які додаються по модулю два з однойменним резервними розрядами блоку. Якщо отриманий синдром має ненульове значення – це свідчить про наявність помилки в ліченому блоці. Також для кодів, що окрім виявлення ще й виправляють помилки, визначає місце помилки [15].

Кількість помилок, яка може бути ідентифікована та виправлена відмовостійкими кодами, визначається за мінімальною кодовою відстанню d [14]. В коригуючих кодах використовується кодова відстань Хемінга – $d(B_i, B_j)$ між комбінаціями A_i, A_j. дорівнює Вона дорівнює числу розрядів, в яких ці комбінації відрізняються одна від одної. Мінімальна відстань, серед всіх пар можливих комбінацій коду називається мінімальною кодовою відстанню:

$$d = \min d(A_i, A_j),$$

де d – мінімальна Хемінгова відстань між комбінаціями A_i, A_j.

Для описання властивостей і можливостей кодів використовують перевірочні Н-матриці. Н-матриця лінійного коду містить r рядків та n колонок. Кожній колонці матриці ставиться в відповідність один інформаційний розряд n – розрядного слова. Рядок Н-матриці містить значення логічної одиниці в позиціях, що відповідають інформаційним розрядам блоку, значення котрого додаються задля генерації відповідного резервного розряду. Щоб виконати підрахунок резервного розряду, треба щоб відповідна колонка матриці містила логічну одиницю в розглянутому рядку і нулі в інших позиціях.

Наприклад, Н-матриця для коду (20,25), має вигляд:

$$H = H'_{5,20}|I_5 = \left|\begin{matrix} 0000000000 & 0111111111 & 10000 \\ 0000111111 & 1000111111 & 01000 \\ 0111000111 & 1000001111 & 00100 \\ 0111000111 & 1001110011 & 00010 \\ 1101101010 & 1010101010 & 00001 \\ A_{20} & & C_5 \end{matrix}\right|$$

Задля того, щоб отримати значення розряду синдрому необхідно додати по модулю два інформаційні та резервні розряди вибраного блоку, яким відповідають логічні одиниці в рядку Н-матриці. Якщо значення синдрому набуває нульового значення – помилка відсутня. У випадку, якщо в значення синдрому не дорівнює нулю, а дорівнює одній з колонок Н-матриці, то це свідчить про наявність помилки в тому розряді, у якій колонці знаходиться значення синдрому. Якщо виникла двократна помилка, синдром набуває значення сумі по модулю два колонок Н-матриці, що відповідають помилковим розрядам. Тому приклади викривлення розрядів та значення синдрому наведені в таблиці 1.

Таблиця 1 – Приклад таблиці синдромів для виявлення двократної помилки

Вектор помилки	1001000...	1100000...	0100100...	0001100...	1000010...
Синдром	00110	00110	01110	01110	01001

Засновуючись на результатах створення синдромів можна дійти до висновку, що наявний код дозволяє виявити двократні помилки та виправити одно розрядні помилки.

Колонки Н-матриці, що розглядається різняться. Окрім того, колонки, що відповідають інформаційним розрядам слова, містять не менше двох одиниць. Через це код, що задається цією Н-матрицею, має мінімальну кодову відстань $d = 3$, тобто дозволяє виправляти одинарні чи виявляти незалежні подвійні помилки.

Будь-які Н-матриці мають такі властивості:

- число логічних одиниць в кожній з колонок Н-матриці, що відповідають інформаційним розрядам блоку , має бути не менше $d - 1$;
- коли відбувається перестановка колонок Н-матриці корегуючі властивості коду у щодо незалежних похибок в розрядів блоків не змінюється;
- коли відбувається перемноження рядків Н-матриці на нульові елементи і додаванні одного рядка з іншим, корегуючі властивості коду не змінюються.

ВИСНОВКИ ДО РОЗДІЛУ 1

В даному розділі було розглянуто проблематику надійності збереження даних при їх віддаленому розподіленому зберіганні. Можна стверджувати, що сучасні технології передачі інформації через мережу Інтернет забезпечують швидкий доступ до віддаленого сховища даних, що робить використання розподілених систем актуальним, однак це зумовлює негативний ефект – який являє затримку через збирання необхідних даних з декількох вузлів.

Отже використання методів відновлення даних на рівні вузлів інформації – є актуальним, як для відновлення втрачених, так і для відновлення тих кластерів, які не мають зв’язку, або мають затримки зі зв’язком.

РОЗДІЛ 2 ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ВІДНОВЛЕННЯ ДАНИХ ПРИ ВІДДАЛЕНОМУ РОЗПОДІЛЕНОМУ ЗБЕРІГАННІ

Аналіз ефективності резервування даних у хмарних середовищах, проведений у першому розділі даної роботи, дозволяє стверджувати, що на сьогоднішній день системи розподіленого віддаленого зберігання інформації потребують модифікації.

Модифікація з огляду на аналіз проведений вище пропонується саме на рівні кластерів інформації, адже саме в цьому аспекті існує декілька невирішених питань по відновленню даних, зокрема:

- зменшення надлишковості;
- тимчасова відсутність зв’язку з вузлом інформації;
- катаклізми природнього, техногенного, політичного чи воєнного характеру.

Під час виконання роботи було проаналізовано низку джерел [1], [17] – [25], які представляють ті чи інші надбання у вирішенні цього питання. Зокрема в роботах [17] – [19] запропоновано побудувати резервування та відновлення інформації на основі діагональних контрольних сум. Хоч наведений там метод забезпечує теоретичний мінімум надлишковості резервування, однак залишається проблема, що у випадку втрати резервних блоків даних буде неможливо відновити інформацію. В роботі [20] запропоновано використовувати метод зворотного атомарного відновлення даних, однак він не забезпечує відновлення всіх даних на вузлі інформації. В роботах [1], [16], [21] – [23] також наведені доволі ефективні методи для вирішення поставлених задач, однак вони дозволяють відновити лише фіксовану кількість втрачених блоків даних.

Існує наразі ще два методи резервування та відновлення даних в системах віддаленого розподіленого зберігання інформації, які були детально проаналізовані, та які найближче підходять до вирішення поставленої задачі:

метод відновлення даних з довільної заданої кількості носіїв [24] та метод відновлення всіх блоків даних на одному сховищі та трьох блоків даних на різних сховищах [25].

Для обох методів присутнє поняття блоку даних, або фрагменту даних – це поняття не прив'язано до будь-якого певного фізичного значення чи природи, блоком даних може бути розділ жорсткого диску чи іншого ЗП, або носій в цілому.

2.1 Опис методу відновлення даних з довільної заданої кількості носіїв [24]

Цей метод характеризується тим, що дозволяє відновити інформацію з будь-якої непарної кількості носіїв віддаленого зберігання, до яких втрачено доступ тимчасово, або вони втрачені повністю.

Під час застосування системи розробник зобов'язаний визначити граничну кількість носіїв даних які можуть бути відновлені, адже задля функціонування методу необхідна така ж кількість резервних ЗП. За таких умов метод забезпечує теоретичний мінімум інформаційної надлишковості інформаційного резервування. Таким чином ми маємо m запам'ятовуючих пристроїв з n фрагментами у кожному та p резервних носіїв з n фрагментами у кожному. Їх можна зобразити у вигляді матриць S та D:

$$S = \begin{pmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,n} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ s_{m,1} & s_{m,2} & \cdots & s_{p,n} \end{pmatrix}, \quad D = \begin{pmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,n} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ d_{p,1} & d_{p,2} & \cdots & d_{p,n} \end{pmatrix}.$$

В таких матрицях стовпець містить однойменні фрагменти всіх носіїв, а кожен рядок це фрагменти одного носія.

Резервні носії кількістю p штук, використовуються для резервування та відновлення інформації, для цього вони поділяються на три різні групи:

В першу групу поділяють носії, які мають індекси $i \in \{1,2,\dots,(p-1)/2\}$. Фрагменти резервних носіїв для цієї групи потрібно формувати так, щоб вони утворювали суму по модулю два з фрагментів інформаційних носіїв, що лежать на висхідних діагоналях матриці під різними кутами, тобто:

$$\forall\, i \in \left\{1,2,\dots,\frac{p-1}{2}\right\}, \forall\, j \in \{1,2,\dots,m\cdot i\}:$$

$$d_{(p-1)/2\,-i+1,j} = \bigoplus_{q\,=\,1}^{[j/i]} S_{[j/i]-q+1,(j-1)mod\ i+1+i(q-1)}$$

$$\forall\, i \in \left\{1,2,\dots,\frac{p-1}{2}\right\}, \forall\, j \in \{m\cdot i+1,\dots,n\}:$$

$$d_{(p-1)/2\,-i+1,j} = \bigoplus_{q\,=\,1}^{m} S_{m-q+1,i\cdot q+j-i\cdot m}$$

В другу групу поділяють носій, який має індекс $i \in \{(p+1)/2\}$. Фрагменти резервних ЗП для цієї групи потрібно формувати так, щоб він дорівнював сумі по модулю два з фрагментів інформаційних носіїв, що належать стовпцям, тобто:

$$\forall\, j \in \{1,2,\dots,n\}:\ d_{(p+1)/2,j} = \bigoplus_{q\,=\,1}^{m} S_{q,j}$$

В третю групу поділяють носії, які мають індекси $i \in \{(p+3)/2,\dots,p\}$. Фрагменти носіїв для цієї групи потрібно формувати з фрагментів інформаційних носіїв, що лежать на низхідних діагоналях матриці під різними кутами, тобто:

$$\forall\, i \in \left\{1,2,\dots,\frac{p-1}{2}\right\}, \forall\, j \in \{1,2,\dots,m\cdot i\}:$$

$$d_{i+(p+1)/2,j} = \bigoplus_{q\,=\,1}^{[j/i]} S_{m-[j/i]+q,(j-1)mod\ i+1+i(q-1)}$$

$$\forall\, i \in \left\{1,2,\dots,\frac{p-1}{2}\right\}, \forall\, j \in \{m\cdot i+1,\dots,n\}:$$

$$d_{i+(p+1)/2,j} = \bigoplus_{q\,=\,1}^{m} S_{q,j-i\cdot m+i\cdot q}$$

Метод дозволяє відновити майже всі фрагменти носіїв доступ до яких відсутній. Значення кількості не відновлених фрагментів даних варіюється від 0 в найкращому випадку до $(m-p)/2$ в найгіршому випадку. Не відновлюються останні однойменні фрагменти носіїв.

Втрачені інформаційні блоки даних позначаються як $X = (x_1, x_2, \dots x_p)$, де $x_1 < x_2 < \cdots < x_p$. Процес відновлення передбачає впорядковані методи відновлення інформації. Існує два етапи:

1) відновлення першого фрагменту з носія другої групи, та, якщо треба, всіх інших необхідних фрагментів з інших недоступних носіїв для відновлення першого;
2) поступове відновлення кожного носія по одному фрагменту, аж поки не будуть відновлені всі фрагменти.

Для розв'язання задачі відновлення визначаються додаткові масиви, зокрема: $B = (b_1, b_2, \dots, b_p)$ – кількість фрагментів на кожному недоступному носії, яка необхідна для відновлення даних з першого фрагменту ЗП $x_{(p+1)/2}$, $A = (a_1, a_2, \dots, a_{p-1})$ – різниці індексів сусідніх пристроїв збереження даних, де $a_i = (x_{i+1} - x_i)$; C= $(c_1, c_2, \dots, c_{p-1})$ – останній відновлений фрагмент кожного недоступного носія. За рахунок цього було отримано рівність:

$$b_i = \begin{cases} a_i \cdot (\left(p - \frac{1}{2} - i\right) + b_{i+1}, i < (p-1)/2 \\ 1, (p-1)/2, \leq i \leq (p+3)/2 \\ a_{i-1} \cdot \left(i - \frac{p+3}{2}\right) + b_{i-1}, i > (p+3)/2 \end{cases}$$

В першому етапі алгоритм виглядає наступним чином:

1) задаємо крок $i = b_1$;
2) задаємо номер носія для відновлення $j = 1$;
    a. якщо не виконується $b_j \geq i$, переходимо в пункт d);
    b. інкрементуємо $C_j = C_j + 1$;
    c. відновлення блоку S_{X_j, C_j};
    d. інкрементуємо $j = j + 1$;
3) якщо виконується умова $j < (p+1)/2$ переходимо в пункт a)
4) змінюємо значення $i = i - 1$;
5) якщо $i \geq 1$, переходимо в пункт 2)
6) оновлюємо крок $i = b_p$;

7) оновлюємо номер ЗП для відновлення $j = p$;
   a. якщо $b_j < i$, переходимо в d);
   b. інкрементуємо c_j;
   c. відновлення блоку S_{X_j,C_j};
   d. декремінуємо $j = j - 1$;
8) якщо $j > (p+1)/2$ виконується – переходимо в a);
9) декремінуємо $i = i - 1$;
10) якщо $i \geq 1$ переходимо в пункт 11);
11) інкрементуємо $C_{(p+2)/2} = C_{(p+2)/2} + 1$;
12) відновлення блоку $S_{X_{(p+1)/2},C_{(p+1)/2}}$.

Другий етап дещо простіший, він складається з p ітерацій – по одному фрагменту з кожного пристрою доступ до яких було втрачено. Головним чином процес полягає у відновленні блоків даних носія з найменшим порядковим номером за допомогою першої групи резервних блоків на непарних кроках алгоритму. А на парних кроках відбувається реконструкція блоків даних тих пристроїв, які мають найбільший порядковий номер за допомогою третьої групи резервних блоків. Останнім відновлюється останній фрагмент ЗП другої групи.

2.2 Опис методу відновлення всіх блоків даних на одному сховищі та трьох блоків даних на різних сховищах [25]

Метод описаний в даній роботі характеризується тим, що дозволяє відновити дані з будь-яких трьох блоків даних та з усіх блоків даних, які згруповані за певним принципом.

Аналогічно до першого методу, під час застосування методу розробник повинен визначити яку кількість блоків даних необхідно зберігати в одному сховищі, адже на основі цих значень буде будуватись матриця формування резервних блоків. Тобто за умови, що ми маємо систему в якій в нас є n інформаційних блоків B_1, B_2 ... B_n та m резервних блоків R_1, R_2 ... R_m, то кожен резервний блок буде визначатись наступний чином:

$$\forall\, i \in \{1,2,\dots,m\}:\ R_i = \bigoplus_{j=1}^{n} a_{i,j}\, B_j$$

де $a_{i,j}$ – коефіцієнти, що є елементи матриці формування резервних блоків, та приймають значення логічного «0» чи «1».

В цьому методі кількість мінімально необхідних резервних блоків даних для гарантованого резервування даних на різних носіях, якщо втрачено не більше трьох, обраховується за наступною формулою:

$$m \geq\ [\log_2(n+1)] + 2$$

Для досягнення можливостей, відновлення вузла інформації, до якого було втрачено тимчасово чи повністю доступ, в методі пропонується формувати матрицю формування відновлювальних блоків особливим чином. Значення, яке розробник визначив для кількості блоків даних r, що зберігаються на одному кластері інформації, необхідне для підрахунку кількості угрупувань, на які поділяється матриця $g = [n/r]$. Формування повинно відбуватись таким чином, щоб в перших $g-1$ групах було по r стовпців, а в останньому фрагменті було $n - r(g-1)$ стовпців.

Створення та заповнення коефіцієнтами матриці формування резервних блоків виконується за наступним алгоритмом:

1) індекс j переводимо в нуль: j =0;
2) для кожного рядка i-тої групи, що визначається номером $(j+i)\ mod\ r$, а $i\ \in \{0, 1, \dots g-1\}$ відбувається наступне:
   a. на позицію, індекс якої визначається $(r+j-1-i)\ mod\ r$ встановлюється логічна одиниця;
   b. на наступних $(r-1-j)$ позиціях в рамках цього фрагменту циклічно встановлюються значення логічного нуля.
3) значення індексу j інкрементується: $j = j+1$. За умови, що $j < r$ необхідно повернутися до виконання пункту 2);

4) елементи матриці, які не були заповнені необхідно сформувати таким чином, щоб мінімальною Хемінговою відстанню між стовпцями було 1 та щоб серед стовпців не було нульового.
5) останні два рядки матриці заповнюються таким чином, щоб кожен кількість одиниць в кожному стовпці була не меншою за три.

Наприклад, результатом алгоритму створення матриці для системи $r = 4$, $n = 16$ буде наступна матриця:

$$A = \begin{vmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

Відновлення втрачених даних відбувається в декілька етапів:

1) за індексами втрачених блоків даних знаходяться стовпчики, які формують собою проміжну матрицю;
2) з проміжної матриці викреслюються рядки, індекси яких співвідносяться з індексам втрачених резервних блоків даних;
3) в проміжній матриці виконується пошук ортогональної підматриці;
4) виконується кроки для пошуку синдрому помилки:
    a. рядки ортогональної підматриці співвідносяться як частинки формування резервних блоків, а одиниці в стовпцях як інформаційні блоки, що їх утворювали. Розв’язується система лінійних рівнянь, для пошуку відповідностей;
    b. з матриці формування резервних блоків беруться рядки, що відповідають за частинки резервних блоків, та виконується пошук синдрому кожної частинки.
5) за допомогою отриманих синдромів знаходимо втрачені блоки даних.

2.3 Аналіз обраних методів та вибір оптимального для розробки способу

Після проведеного аналізу вибраних методів, необхідно повернутись до критичних показників які були сформульовані в першому розділі, а саме будь-який метод можна характеризувати такими властивостями:

- кількість відновлюваної інформації;
- надлишковість;
- обчислювальна складність процедури відновлення даних з використанням резервних носіїв.

Для наближення сприятливих реалій для порівняння методів приймемо, що вони відновлюють дані з n інформаційних носіїв, де під інформаційним носієм мається на увазі масиви дисків великої ємності об'єднаних в сервер. Також, нехай в одному вузлі інформації (кластері) знаходиться по p серверів. Кількість резервних носіїв позначається буквою m.

Метод відновлення даних з довільної заданої кількості носіїв дозволяє розробнику системи вирізнити кількість носіїв, які можна буде відновити p, інший метод теж дозволяє коригувати максимальну кількість носіїв, які можна буде відновити, якщо вони були груповані. В даному порівнянні кожен з методів в якості максимального числа носіїв для відновлення буде мати p.

В такому разі перший метод буде мати $m = p$ кількість резервних носіїв даних, в той же час другий метод, через формування матриці мусить мати:

- якщо $p \leq [\log_2(n+1)]$, то $m = [\log_2(n+1)] + 2$;
- якщо $p > [\log_2(n+1)]$, то $m = p + 2$.

Аналізуючи та порівнюючи ці значення можна зробити висновок, що перший метод є менш надлишковим, а саме на $q = [\log_2(n+1)] - p$.

Якщо роздивитися питання щодо відновлення даних, то ситуація наступна. Перший метод за сприятливих умов може відновити всі фрагменти в усіх p носіях, однак існує несприятливий сценарій, при якому втрачається $(n-p)/2$ однойменних фрагментів. Існує ймовірність, що втрачені дані нам не були потрібні і ми можемо їх використовувати, однак якщо саме втрачені дані були нам необхідні, то не буде можливості використання відновлених даних одразу.

У випадку якщо доступ до вузла даних тимчасово було втрачено, це негативно вплине на час роботи користувача, однак у випадку якщо було фізично втрачено кластер повністю і ми маємо частину помилок – ці проблеми вже ніяк не вирішити.

Як результат, перший метод не може сам по собі повністю забезпечити цілісність даних в найгіршому випадку, хоч і надає змогу в частині випадків відновити всі фрагменти втрачених носіїв.

Другий метод хоч і дозволяє відновити лише фіксовану частину носіїв даних, якщо вони були втрачені різних вузлах, однак від дозволяє гарантовано відновити втрачений кластер пам’яті.

Метод відновлення даних з довільної заданої кількості носіїв не потребує в додатковому збережені матриць, на відміну від іншого методу. Також, в методі відновлення всіх блоків даних на одному сховищі та трьох блоків даних на різних сховищах відбувається розв’язання системи рівнянь для пошуку синдрому.

З огляду на вимоги висунуті під час аналізу проблематики було вирішено доцільним використовувати при розробці способу відновлення даних при розподіленому їх зберіганні саме другий метод, адже він гарантовано відновлює дані з втраченого вузла даних, що безперечно збільшує надійність системи збереження інформації. Це дозволить запобігти втратам на випадок локальних конфліктів, війн тощо. Окрім того, при втраті даних, які можна відновити, не треба завбачливо перевіряти чи реконструювались вся необхідна інформація.

Також, надлишковість другого методу над першим не є високою, зокрема якщо наприклад об’єднувати дані за географічною ознакою, та прийняти, наприклад що у нас існує $n = 1000000$ носіїв даних, то другий метод мусить сформувати лише $m \geq [\log_2(+1)] + 2$, тобто 22 резервні блоки, то лише при формування кластерів $p < 20$, різниця між методами буде більше 2.

ВИСНОВКИ ДО РОЗДІЛУ 2

В даному розділі було висвітлено наявні методи відновлення даних при розподіленому віддаленому їх зберіганні. Виявлено переваги та недоліки методів. Також було описано деталі провідних методик резервування даних.

Проведено детальний порівняльний аналіз двох передових методів резервування інформації. Було обрано метод відновлення всіх блоків даних на одному сховищі та трьох блоків даних на різних сховищах на підставі його кращої спроможності забезпечити цілісність даних при втраті доступу до вузла інформації.

3. ОПИС ТА ВІДЛАГОДЖУВАННЯ ПОВЕДІНКОВОЇ МОДЕЛІ СИСТЕМИ ВІДНОВЛЕННЯ БЛОКІВ ДАНИХ ПРИ ВІДДАЛЕНОМУ РОЗПОДІЛЕНОМУ ЗБЕРІГАННІ

Під час виконання магістерської дисертації було синтезовано систему з використанням розробленого способу на основі обраного методу. Оптимальним методом вирішення поставленої задачі – було використання середовища розробки Active-HDL та використання мови опису апаратури інтегральних схем VHDL, як мову опису пристроїв системи. Таке рішення було зумовлене тим, що VHDL (**V**ery high speed integrated circuits **H**ardware **D**escription **L**anguage) вже довгий час є базисом для розробки різної апаратури сучасних обчислювальних систем. Окрім того, середовище розробки Active-HDL дає змогу не тільки розробляти різні системи на основі інтегральних схем, апаратури та пристрої, а й відлагоджувати, моделювати та відслідковувати поведінку розробленого продукту всередині середовища. Для перевірки справності роботи системи на основі розробленого способу ці аспекти програмного середовища є ключовими.

Задля побудови поведінкової моделі було використано стандарт IEEE 1164, окрім того в сутностях ControlUnit.vhd, MissedPartsAnalyzeUnit.vhd, DetectOrthogonalMatrixUnit.vhd, було використано бібліотеку IEEE Numeric_std.

Для оптимального та ефективного використання обраного методу пропонується спосіб його використання в системі. Однією з ключових вимог поставлених до розв’язання існуючої проблеми є можливість відновлення даних «на льоту», що дозволить надати користувачу необхідні дані одразу, навіть якщо вони були втрачені або затримані під час передачі, або інформація була втрачена фізично. Потенційно при такому способі використання даного методу зростає не тільки ефективність відновлення інформації при втраті блоків даних чи одного вузла даних, а й при звичайній роботі забезпечується швидший доступ до даних, шляхом відновлення тих, які ще в шляху по мережі. Ця можливість досягається тим, що робота по відновленню даних не залежить від розміру самих блоків

даних, а залежить тільки від конфігурації системи при налаштуванні. У випадку, коли дані дійсно було втрачено через ту чи іншу причину користувач зможе використовувати потрібну йому інформацію до того моменту, коли не відбудуться роботи по повноцінному відновленню даних на вузлі чи вузла цілком.

3.1 Структурний опис системи

Задля забезпечення використання переваг способу пропонується безпосереднє встановлення системи в розрив загальної шини даних. Адже, як зазначалось раніше, таке розташування дозволить непомітно для користувача відновити втрачені блоки даних, якщо це можливо, або ж сповістить про неможливість відновлення даних до того, як користувач матиме змогу користуватись невірними даними. Звичайно, в цьому випадку відбувається постійний контроль втрат даних, що може призвести до затримок, однак складність та час аналізу на наявність помилок зумовлена лише однократним прочитанням масиву основних даних та резервних даних, що надійшли та однократного прочитання масиву індексів, що зумовлює жваву відповідь у випадку, коли не має жодних помилок серед отриманих блоків інформації. Система компактно розміщена в одному структурному блоці.

3.2 Функціональний опис системи

Система може виконувати наступні базові функції для забезпечення справної роботи вибраного алгоритму та відновлення блоків даних максимально непомітно для користувача, де це можливо:

- створення та збереження матриці формування резервних блоків;
- прийом блоків даних та резервних блоків;
- аналіз та відновлення блоків даних, якщо це можливо;
- сповіщення про неможливість відновлення даних.

Було вирішено розподілити роботу між логічними пристроями, що в сукупності будуть собою представляти систему на основі вищеописаного

способу. Всередині пристроїв було виділено функціонально окремі частини логіки, які були сепаровані в під програми. Функціональна схема містить наступні пристрої:

- CU – Control Unit – блок керування;
  - RM – RotateMatrix;
  - FG – FirstGo;
  - FGIM – FillGapsInMatrix;
  - BMA – BuildMatrixA.
- MPIU – Missed Parts Identify Unit – блок виявлення втрачених блоків даних;
- MPAU – Missed Parts Analyze Unit – блок аналізу втрачених блоків даних;
- DOMU – Detect Orthogonal Matrix Unit – блок пошуку ортогональної матриці;
  - CDHNZ – CheckIsMainDiagonalHasNoZero;
  - CMULT – CheckIsMatrixUpperLowTriangle;
  - CM – CalculateMatrix;
  - RMR – ReplaceMatrixRows;
  - CIMO – CheckIfMatrixOrthogonal;
  - SMRACD – SortMatrixRowsAndClearDublicates;
  - NS – NextSet.
- SOECU – System Of Equations Calculate Unit – блок вирішення системи рівнянь;
  - RDMR – ReplaceDataMatrixRows;
  - RBUMR – ReplaceBackUpMatrixRows;
  - CM – CalculateMatrix;
  - SMR – SortMatrixRows;
  - FSFC – FormSyndromeForCalculation.

- MDGU – Missed Data Generator Unit – блок генерації втрачених блоків даних.

Для створення матриці формування резервних блоків подається сигнал на блок Control Unit на вхід Initialize, окрім того на входи NumberOfDataBlocks та NumberOfBackUpBloks подається відповідна кількість блоків, які будуть використовуватись в роботі системи. На основі цих даних синтезується відповідна матриця формування резервних блоків, яка буде зберігатися в блоці Control Unit та буде використовуватись в роботі пристроїв Missed Parts Analyze Unit та System Of Equations Calculate Unit.

Всередині пристрою Control Unit за побудову матриці відповідає процедура BuildMatrixA, яка заповнює основні позиці по кластерам. Після цього процедура RotateMatrix перетворює матрицю початкових значень в матрицю того виду яку використовують всі інші пристрої. Далі позначаються вже утворені значення матриці в процедурі FirstGo. Останню вирішальну роль грає процедура FillGapsInMatrix, яка відповідає за проставляння інших значень в матриці.

Для синтезу матриці було модифіковано та пристосовано до автоматичного використання метод створення, який зазначений в методі, що ліг в основу роботи способу відновлення даних. Кроки генерації матриці формування резервних блоків наступні:

1) формується масив з прото-значеннями окремих кластерів інформації:
    a. задається значення J = 0;
    b. задається значення $I \in \{0, 1, 2, \dots, g-1\}$, де g – це індекс кластеру;
    c. кожний $(I+J)\ mod\ r$, де r – кількість блоків даних в кластері, заповнюється кодом з $(r-J)$ розрядів, з логічною «1» на позиції $(r+J-1-I)$, а також в рамках цього бітового фрагменту $(r-1-J)$ логічних «0»;

   d. значення J збільшується на одиницю. Якщо $J < R$, то ми знову переходимо в пункт c), в іншому випадку масив прото-значень вважається сформованим.

2) у випадку, якщо $m - 2 > r$, де m – кількість резервних блоків даних, r – кількість блоків даних в кластері, ми вимушені заповнити пустий рядок матриці послідовністю з $\frac{[\log_2 n]}{2} - 1$ кількості логічних нулів та відповідно послідовність з $n - \frac{[\log_2 n]}{2} - 1$ кількості логічних одиниць. Якщо $m - 2 = r$ просто переходимо в пункт 3);
3) перевіряємо чи наявні у нас вже сформовані числа в вигляді логічних векторів, якщо такі наявні – помічаємо їх в буферний масив;
4) кожен рядок матриці аналізується наступним чином:
   a. підраховується кількість нулів, яка вже проставлена в цьому рядку;
   b. помічаються двійкові масиви, які являють собою частину стовпця матриці довжини $m - 2$, які за умова проставлення логічного значення в цьому рядку матриці набудуть свого числового значення;
   c. аналізуються двійкові масиви, що були помічені в попередньому пункті:
      i. в пропущену комірку масиву (яка знаходиться на рядку, що ми аналізуємо) задаємо значення логічного нуля;
      ii. перевіряємо чи сформоване число вже помічене в буферному масиві, якщо так, то переходимо в пункт iii), інакше в пункт iv);
      iii. в пропущену комірку масиву задаємо значення логічної одиниці;

iv. зберігаємо значення в комірці.

d. в усі інші незаповнені комірки зліва направо спочатку заповнюються значеннями логічного нулю, доки кількість нулів у рядку не буде дорівнювати $\frac{[\log_2 n]}{2} - 1$, всі інші значення заповнюються нулями.

5) останні два рядки заповнюються таким чином, щоб в кожному стовпці було щонайменше 3 логічні одиниці.

Формування матриці є доволі гнучким, адже в даній моделі можна формувати матриці менших розмірів, при цьому потрібна нам матриця буде розташовуватися в лівій верхній частині великої матриці, всі інші комірки будуть приймати значення логічного нуля.

Під час роботи самої системи по відновленню блоків даних спершу працює пристрій Missed Parts Identify Unit. На вхід DataBlocks та BackUpDataBlocks подаються блоки даних та резервні блоки даних відповідно. Пристрій аналізує сигнали що прийшли та формує вихідні сигнали MissedBlockIndexes та BackUpMissedBlockIndexes які представляють собою бітовий масив, в якому на місцях індексів втрачених чи пошкоджених блоків даних та резервних блоків відповідно, стоїть логічна «1», у випадку якщо блок не втрачений логічний «0».

Наступний пристрій – Missed Parts Analyze Unit, використовується для аналізу отриманих бітових масивів індексів втрачених блоків даних з першого пристрою та формування зміненої матриці формування резервних блоків. Спочатку пристрій виконує аналіз отриманих масивів на спроможність відновлення втрачених блоків даних. Тобто якщо в масиві індексів блоків даних не було виявлено жодної помилки, то ми подаємо на вихід NoMissedData сигнал про те, що не було виявлено жодної помилки. В іншому випадку, якщо було виявлено більше 3-х помилок на різних кластерах інформації, то відповідно ми не можемо їх відновити за допомогою обраного методу, а тому ми подаємо на вихід Error сигнал про помилку. Однак, якщо ці втрачені блоки даних знаходяться на одному кластері інформації, або було виявлено 3 або менше

помилки на різних кластерах, то ми можемо їх відновити, а тому ми переходимо до наступного етапу роботи пристрою. В фінальному етапі роботи пристрою формується змінена матриця формування резервних блоків даних, де значення логічних «1» та «0» залишається тільки в тих рядках де є невтрачені резервні блоки даних та в тих стовбцях де є втрачені блоки даних. Отримана матриця подається на вихід ResultNewMatrix.

Detect Orthogonal Matrix Unit – пристрій, що виконує задачі формування підматриці для пошуку синдромів для відновлення втрачених блоків даних, пошуку ортогональної підматриці з вищезазначеної матриці, а також сповіщає про помилку, якщо в ході роботи вона виникла. Для формування підматриці для пошуку синдрому використовується змінена матриця формування резервних блоків, де додаються пропущені рядки з втраченими резервними блоками. Пошук ортогональної матриці відбувається в декілька етапів:

1) шукаємо нове унікальне розміщення рядків матриці;
2) аналізуємо перші n рядків матриці (де n – це кількість стовпців матриці) на предмет ортогональності:
    a. перевіряємо чи кожен рядок та кожен стовбець мають хоча б по одній логічній «1»;
    b. перевіряємо чи є матриця нижньо- чи верхньотрикутною, якщо так переходимо в пункт e), інакше – в пункт c).
    c. переставляємо рядки матриці так, щоб в першому елементі матриці була логічна «1», а також на головній діагоналі було як найбільше одиниць;
    d. зводимо матрицю до верхньотрикутної.
    e. перевіряємо чи всі елементи головної діагоналі матриці є логічними «1», якщо так – матриця ортогональна, якщо ні – ні.
3) якщо матриця не є ортогональною – йдемо в пункт 1), інакше – закінчуємо пошук.

Якщо під час пошуку ортогональної матриці були перебрані всі варіанти розміщення рядків, а жодна з матриць не була ортогональною, то на вихід Error подається сигнал про помилку, адже в такому разі відновити дані є неможливим.

В пристрої Detect Orthogonal Matrix Unit процедура CheckIsMainDiagonalHasNoZero робить перевірку матриць на наявність одиниць на головній діагоналі квадратних матриць, які передаються в процедуру. Перевірка матриці на належність до класу верхньотрикутних чи нижньотрикутних матриць проводиться в процедурі CheckIsMatrixUpperLowTriangle. Для зведення квадратної матриці до класу верхньотрикутної матриці відбувається в процедурі CalculateMatrix. В свою чергу процедура ReplaceMatrixRows відповідає за зміну рядів матриці місцями, за вказаними індексами. CheckIfMatrixOrthogonal – це функціональна одиниця, яка використовує всі попередні процедури для того, щоб визначити чи є отримана квадратна матриця ортогональною. Після роботи вона видає вердикт, який обробляє пристрій. В парі з попередньою процедурою працює процедура NextSet, для того, щоб надати новий варіант квадратної підматриці з матриці помилок який буде досліджений на ознаки ортогональності. SortMatrixRowsAndClearDublicates використовується над матрицею помилок для того, щоб відсортувати рядки матриці за значенням та видалити з цієї матриці рядки дублікати, що в рази пришвидшить роботу пристрою, адже скоротить кількість рядків для варіацій розміщень.

System Of Equations Calculate Unit – це пристрій, що під час своєї роботи виконує наступні ключові функції:

- створює та вирішує систему рівнянь для знаходження способу відновлення втрачених блоків даних;
- формує синдром для знаходження втрачених блоків з використання не страчених блоків.

Пристрій аналізує отримані ортогональну матрицю та підматрицю для пошуку синдромів та знаходить відповідність чистин тих резервних блоків, які були використані в ортогональні матриці, рисунки 3.1 – 3.2.

D1	D7	D12
0	1	1
0	1	0
1	0	1

Рисунок 3.1 – Ортогональна матриця

	D1	D7	D12
R1	0	1	1
R2	1	1	1
R3	0	1	0
R4	1	1	0
R5	1	0	1
R6	0	0	0
R7	0	0	0

Рисунок 3.2 – Підматриця для пошуку синдромів

Пристрій формує систему рівнянь, що складається з двох матриць: нашої ортогональної матриці та матриці резервних блоків як на рисунку 3.3, та вирішує її. В результаті отримають матрицю значень резервних блоків, як на рисунку 3.4.

R1	R2	R3	R4	R5	R6	R7		D1	D7	D12
1	0	0	0	0	0	0		0	1	1
0	0	1	0	0	0	0	=	0	1	0
0	0	0	0	1	0	0		1	0	1

Рисунок 3.3 – Матричний вигляд системи рівнянь, що застосовується в роботі системи

R1	R2	R3	R4	R5	R6	R7		D1	D7	D12
1	0	1	0	1	0	0		1	0	0
0	0	1	0	0	0	0	=	0	1	0
1	0	1	0	0	0	0		0	0	1

Рисунок 3.4 – Вирішена система рівнянь

Після вирішення рівняння отримана матриця використовується для того, щоб визначити рядки матриці формування резервних блоків, які необхідно скласти за модулем два, щоб отримати синдром відомих блоків даних, які необхідні будуть для відновлення втрачених блоків.

Процедура SortMatrixRows спочатку сортує матрицю даних таким чином, щоб по максимуму заповнити головну діагональ логічними одиницями, а також обов’язково щоб залишилась одиниця в першій позиції на першому рядку. Процедури ReplaceDataMatrixRows та ReplaceBackUpMatrixRows відповідають за перестановку рядків матриці, що відповідає за дані, та матриці, що відповідає за резервні блоки даних, відповідно. Виконується це за рахунок переданих індексів. CalculateMatrix – це процедура, що відповідає за вирішення системи рівнянь, приклад якої зазначений на рисунку 3.4. Методами перестановки рядків та сумування по сумі два ця процедура зводить матрицю даних до одиничної, і в цьому випадку в матриці резервних блоків буде вирішення. Останньою процедурою виступає FormSyndromeForCalculation, яка з отриманого рішення рівняння виводить синдром, за допомогою якого можна відновити втрачені блоки даних. Синдром формується наступним чином, що для кожного пропущеного блоку даних:

1) проходимо по відповідному рядку рішення системи рівнянь, коли знаходимо значення логічної одиниці, то зберігаємо значення індексу резервного блоку даних;

2) з матриці формування резервних блоків даних додаємо рядки по модулю два за знайденими індексами.

В результаті буде отримана перша частина синдрому, що складається з індексів виключно блоків даних для відновлення втрачених даних. Другою частиною синдрому і є розв’язок системи рівнянь, адже там вже зазначено які самі резервні блоки треба брати під час процесу відновлення.

Missed Data Generator Unit – пристрій призначений для відновлення втрачених блоків на основі проведених обчислень в попередніх блоках. Також, у випадку коли не було виявлено жодної помилки на вхід пристрою NoMissedData, буде отриманий сигнал про відсутність помилки, тоді на вихід DataBlocks буде направлений отриманий сигнал звідти ж. Під час відновлення втрачених блоків даних пристрій використовує синдром отриманий з входу SyndromeOfCalculation, матрицю резервних блоків з входу BackupDataMatrix, непошкоджені блоки даних та резервних блоків.

Важливо зазначити, що розроблений спосіб використовує перевагу методу, на якому заснований, та не зважає на розмір блоків даних, задля яких нам треба побудувати інфраструктуру відновлення. Це може бути як одне бітове слово, так і flash-носій, головним чином система працює не з самими даними, а з їх індексами, утворюючими матрицями та підматрицями. Відповідно до цього можна зробити теоретичне твердження, що різниця між роботою даного способу на маленьких обсягах даних та між роботою на великих обсягах даних буде залежати виключно від швидкості читання та запису даних в шину.

3.3 Загальний опис поведінкової моделі системи на основі методу

Під час роботи було прийнято рішення виконувати окреме моделювання кожного логічного пристрою всередині системи окремо. Таким чином було проведено моделювання, тестування та відлагоджування кожного елемента системи.

Розроблена модель містить наступні файли:

- ControlUnit.vhd;

- MissedPartsIdentifyUnit.vhd;
- MissedPartsAnalyzeUnit.vhd;
- DetectOrthogonalMatrixUnit.vhd;
- SystemOfEquationsCalculateUnit.vhd;
- MissedDataGeneratorUnit.vhd;
- Model.vhd.

Усі вищевказані файли в рамках мови VHDL є сутностями. Сутність Model є комбінуючою сутністю, тобто виконує функцію об’єднання всіх інших частин в рамках одної системи.

3.4 Опис та відлагодження сутностей поведінкової моделі системи та приклади їх роботи

Для зручності подання та перевірки даних, які обробляють та видають ті чи інші пристрої було обрано наступні налаштування:

- розмір одного блоку даних – 8 біт;
- кількість блоків даних – 19;
- кількість резервних блоків даних – 7.

Компонент ControlUnit.vhd, що виконує функції однойменного розробленого пристрою, призначений для генерації матриці формування резервних блоків для заданих параметрів, зберігання цієї матриці та видачі повідомлень про помилку якщо така виникла.

В даній моделі присутні наступні входи та виходи:

- Error – вхід;
- Initialize – вхід;
- NumberOfDataBlocks – вхід;
- NumberOfBackUpBloks – вхід;
- ErrorFound – вихід;
- MatrixA – вихід.

Приклади роботи даної сутності наведені нижче.

На рисунку 3.5 показано результати роботи при тому, що на входи подавались наступні дані:

- Error = 1 (на 20 нс);
- Initialize = 1 (на 10 нс);
- NumberOfDataBlocks = 19 (на 10 нс);
- NumberOfBackUpBloks = 7 (на 10 нс).

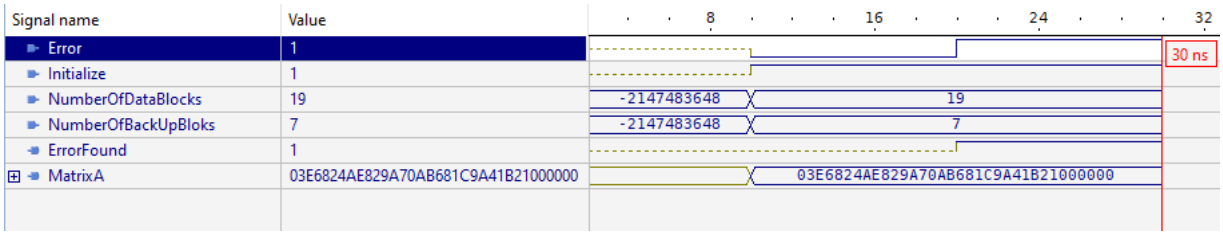

Рисунок 3.5 – Перша симуляція та відлагодження компоненти ControlUnit.vhd

На рисунку 3.6 можна побачити роботу приладу за наступних вхідних даних:

- Error = 1 (на 100 нс та 300 нс);
- Initialize = 1 (на 0 нс, 200 нс та 400 нс);
- NumberOfDataBlocks = 15 (на 0 нс), 19 (на 400 нс);
- NumberOfBackUpBloks = 6 (на 0 нс), 7 (на 400 нс).

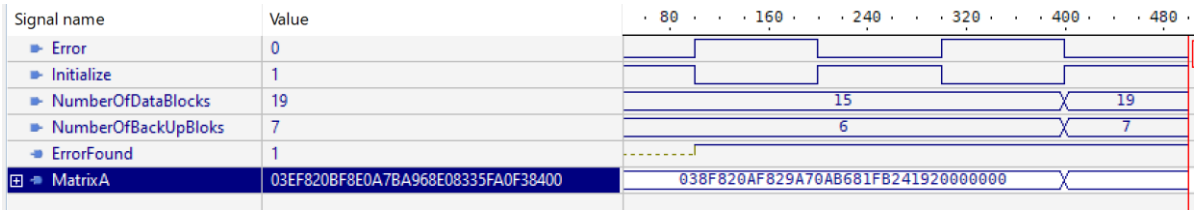

Рисунок 3.6 – Друга симуляція та відлагодження компоненти ControlUnit.vhd

На рисунку 3.7 проілюстрована робота приладу за наступних вхідних даних:

- Initialize = 1 (на 0 нс, 20 нс та 40 нс);

- NumberOfDataBlocks = 19 (на 0 нс, 50 нс), 17 (на 20 нс), 13 (на 40 нс);
- NumberOfBackUpBloks = 7 (на 0 нс, 50 нс), 6 (на 40 нс).

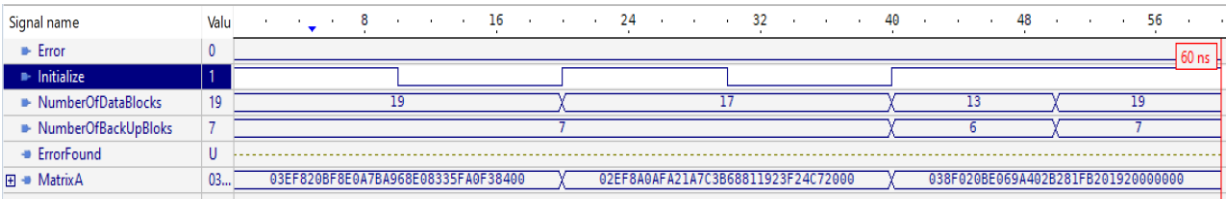

Рисунок 3.7 – Третя симуляція та відлагодження компоненти ControlUnit.vhd

Як видно з останньої симуляції, якщо значення Initialize не було переведене в логічну одиницю з логічного нуля, то генерація матриці не відбувається.

MissedPartsIdentifyUnit.vhd – це сутність, яка виконує функції пристрою, що призначений для виявлення індексів втрачених блоків даних.

В даній моделі присутні наступні входи та виходи:

- DataBlocks – вхід;
- BackUpDataBlocks – вхід;
- SizeOfOneDataBlock – вхід;
- MissedBlocksIndexes – вихід;
- BackUpMissedBlocksIndexes – вихід.

Приклади роботи даної сутності наведені нижче.

На рисунку 3.8 показано результати роботи при тому, що на входи подавались наступні дані:

- DataBlocks = 0102030405060708UUUUUUUU0D0E0F1011121316;
- BackUpDataBlocks = 01UU0F0406050016;
- SizeOfOneDataBlock = 8.

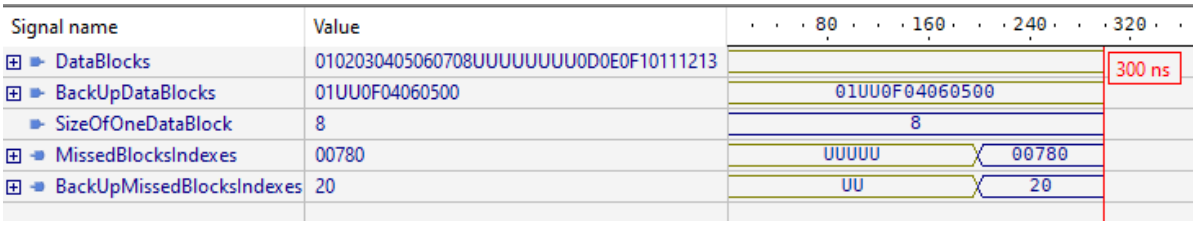

Рисунок 3.8 – Перша симуляція та відлагодження компоненти MissedPartsIdentifyUnit.vhd

На рисунку 3.9 проілюстрована робота приладу за наступних вхідних даних:

- DataBlocks = UU0203040506UU08090A0BUU0D0E0F1011121316;
- BackUpDataBlocks = 010E0F0406050016;
- SizeOfOneDataBlock = 8.

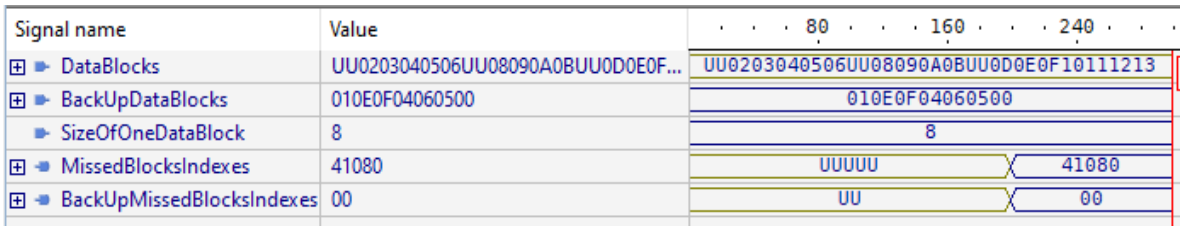

Рисунок 3.9 – Друга симуляція та відлагодження компоненти MissedPartsIdentifyUnit.vhd

На рисунку 3.10 зображена робота приладу за наступних вхідних даних:

- DataBlocks = UUUU03040506UU08090A0BUU0D0E0F1011121316;
- BackUpDataBlocks = 010E0F0406050016;
- SizeOfOneDataBlock = 8.

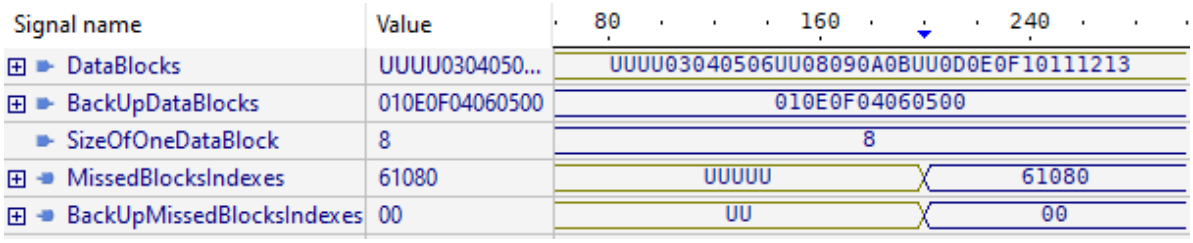

Рисунок 3.10 – Третя симуляція та відлагодження компоненти MissedPartsIdentifyUnit.vhd

Для аналізу кількості помилок та створення зміненої матриці формування резервних блоків використовується компонента MissedPartsAnalyzeUnit.vhd.

В даній моделі присутні наступні входи та виходи:

- MatrixA – вхід;
- MissedDataIndexes – вхід;
- MissedBackupIndexes – вхід;
- ResultNewMatrix – вихід;
- NumberOfMissedData – вихід;
- Error – вихід;
- NoMissedData – вихід.

Приклади роботи даної сутності наведені нижче.

На рисунку 3.11 в якості вхідних даних використовувались наступні значення:

- MatrixA = 03E6824AE829A70AB681C9A41B2100000016;
- MissedDataIndexes = 4108016;
- MissedBackupIndexes = 20_{16}.

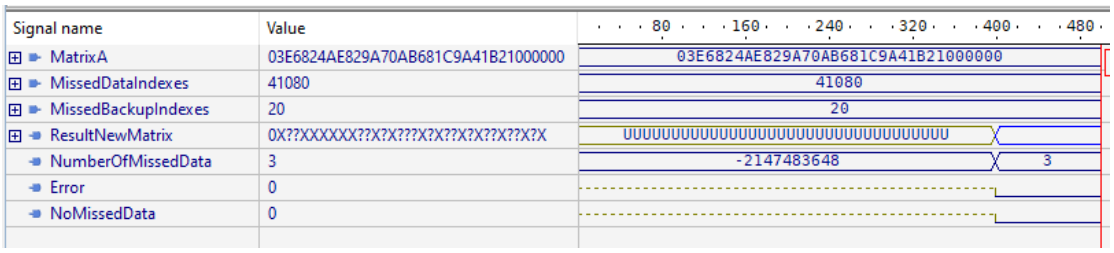

Рисунок 3.11 – Перша симуляція та відлагодження компоненти MissedPartsAnalyzeUnit.vhd

На рисунку 3.12 показано результати роботи при тому, що на входи подавались наступні дані:

- MatrixA = 03E6824AE829A70AB681C9A41B2100000016;
- MissedDataIndexes = 0078016;
- MissedBackupIndexes = 20_{16}.

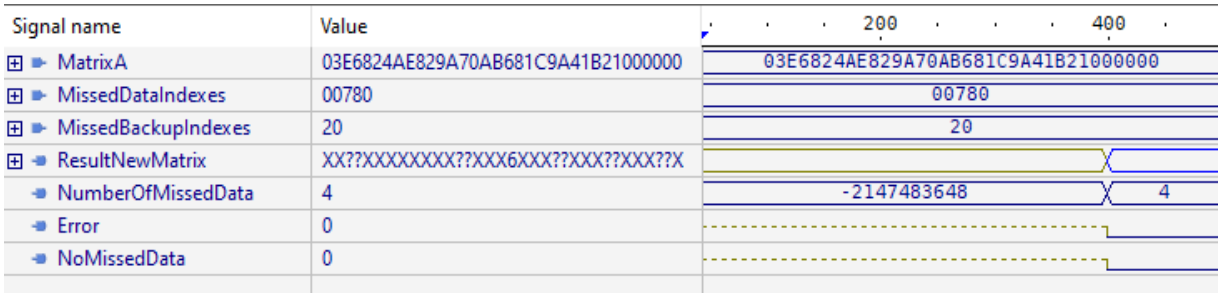

Рисунок 3.12 – Друга симуляція та відлагодження компоненти MissedPartsAnalyzeUnit.vhd

На рисунку 3.13 проілюстрована робота приладу за наступних вхідних даних:

- MatrixA = 03E6824AE829A70AB681C9A41B2100000016;
- MissedDataIndexes = 0000016;
- MissedBackupIndexes = 00_{16}.

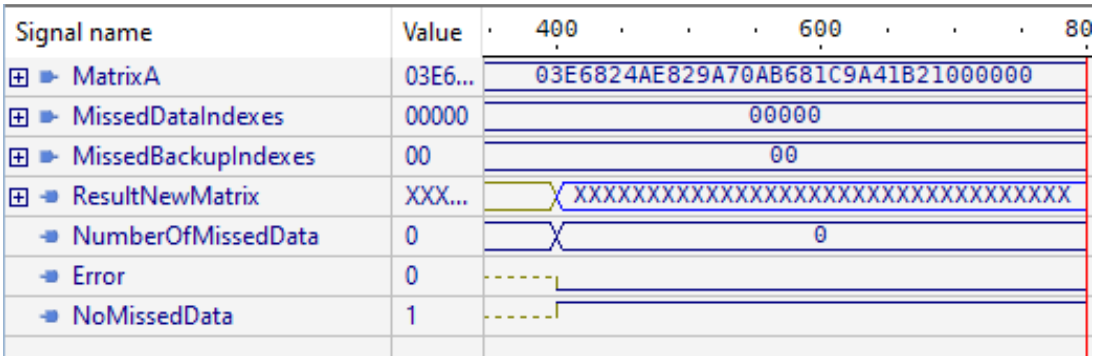

Рисунок 3.13 – Третя симуляція та відлагодження компоненти MissedPartsAnalyzeUnit.vhd

На рисунку 3.14 показана робота приладу за наступних вхідних даних:

- MatrixA = 03E6824AE829A70AB681C9A41B2100000016;
- MissedDataIndexes = 4604016;
- MissedBackupIndexes = 00_{16}.

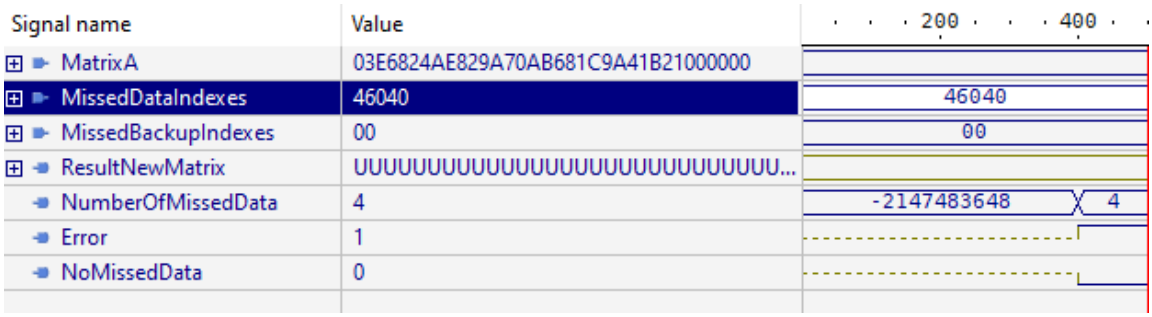

Рисунок 3.14 – Четверта симуляція та відлагодження компоненти MissedPartsAnalyzeUnit.vhd

Сутність DetectOrthogonalMatrixUnit.vhd призначена для виявлення ортогональної матриці.

В даній моделі присутні наступні входи та виходи:

- ChangedAMatrix – вхід;
- NumberOfMissedData – вхід;
- MissedBackupIndexes – вхід;
- ErrorMatrix – вихід;
- OrthogonalMatrix – вихід;
- Error – вихід.

Тестові запуски сутності наведені нижче.

На рисунку 3.15 в якості вхідних даних використовувались наступні значення:

- ChangedAMatrix = матриця де значення тільки в 1,7,12 стовпцях;
- NumberOfMissedData = 3;
- MissedBackupIndexes = 20_{16}.

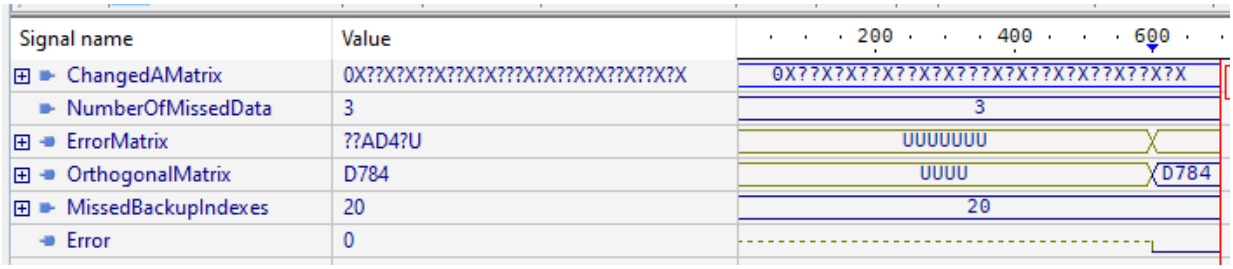

Рисунок 3.15 – Перша симуляція та відлагодження компоненти DetectOrthogonalMatrixUnit.vhd

На рисунку 3.16 проілюстрована робота приладу за наступних вхідних даних:

- ChangedAMatrix = матриця де значення тільки в 8-12 стовпцях;
- NumberOfMissedData = 4;
- MissedBackupIndexes = 20_{16}.

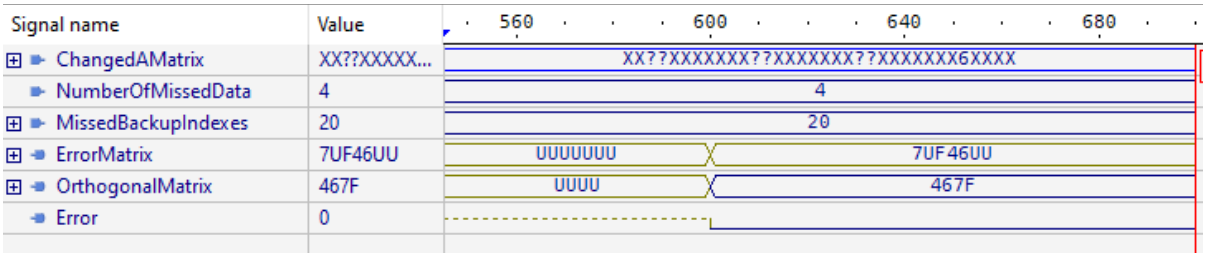

Рисунок 3.16 – Друга симуляція та відлагодження компоненти DetectOrthogonalMatrixUnit.vhd

На рисунку 3.17 продемонстрована робота приладу за наступних вхідних даних:

- ChangedAMatrix = матриця де значення тільки в 8-12 стовпцях;
- NumberOfMissedData = 4;
- MissedBackupIndexes = $5F_{16}$.

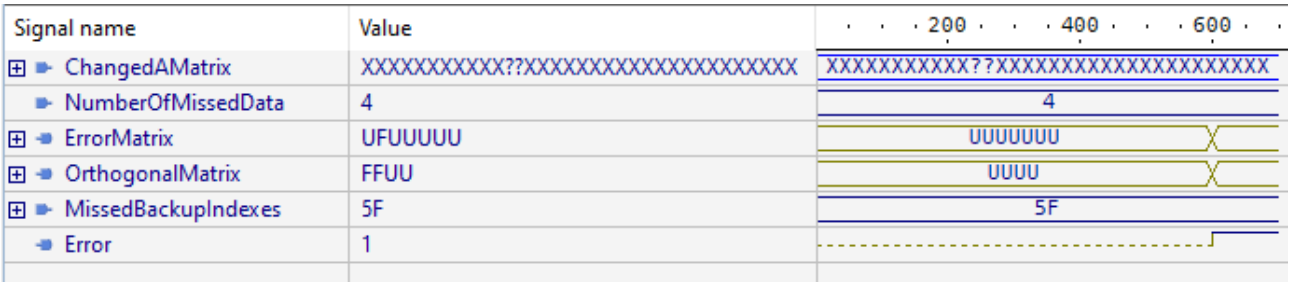

Рисунок 3.17 – Третя симуляція та відлагодження компоненти DetectOrthogonalMatrixUnit.vhd

Компонент SystemOfEquationsCalculateUnit.vhd виконує функції однойменного розробленого пристрою. Ця сутність призначена для вирішення системи рівнянь та формування синдрому для того, щоб відновити втрачені блоки даних.

В даній моделі присутні наступні входи та виходи:

- AMatrix – вхід;
- ErrorMatrix – вхід;
- OrthogonalMatrix – вхід;
- MissedDataArray – вхід;
- SyndromeOfCalculation – вихід;
- BackupDataMatrix – вихід.

Приклади роботи даної сутності наведені нижче.

На рисунку 3.18 проілюстрована робота приладу за наступних вхідних даних:

- AMatrix = 03E6824AE829A70AB681C9A41B2100000016;
- ErrorMatrix = $7D6A_{16}$ 00_2;
- OrthogonalMatrix = 6A16 12;
- MissedDataArray = 4108016;

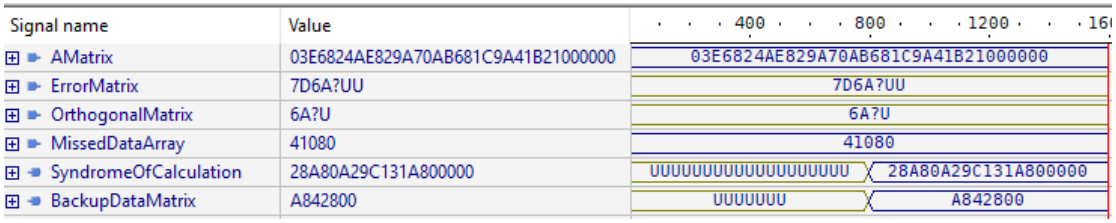

Рисунок 3.18 – Перша симуляція та відлагодження компоненти SystemOfEquationsCalculateUnit.vhd

На рисунку 3.19 в якості вхідних даних використовувались наступні значення:

- AMatrix = 03EF820BF8E0A7BA968E08335FA0F3840016;
- ErrorMatrix = 7E42816 02;
- OrthogonalMatrix = 8D1116;
- MissedDataArray = 4108016;

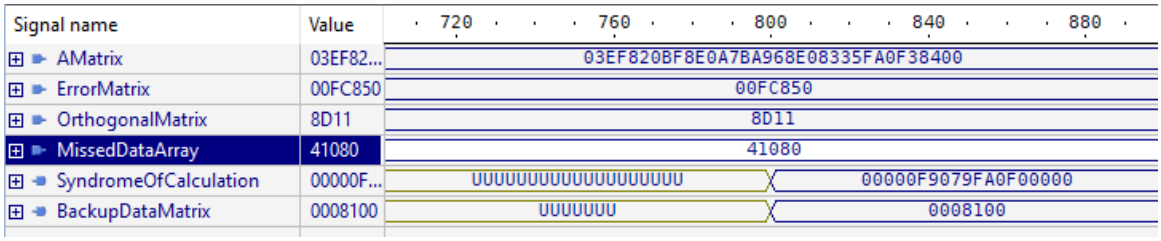

Рисунок 3.19 – Друга симуляція та відлагодження компоненти SystemOfEquationsCalculateUnit.vhd

MissedDataGeneratorUnit.vhd виконує функції однойменного розробленого пристрою та призначений для відновлення втрачених блоків за допомогою синдрому.

В даній моделі присутні наступні входи та виходи:

- NoMissedData – вхід;
- BackUpDataBlocks – вхід;
- SizeOfOneDataBlock – вхід;
- MissedDataArray – вхід;
- SyndromeOfCalculation – вхід;

- BackupDataMatrix – вхід;
- DataBlocks – вхід/вихід.

Приклади роботи даної сутності наведені на Рисунку 3.20. В якості даних використовувались наступні значення:

- NoMissedData = 1;
- BackUpDataBlocks = 01UU0F0406050016;
- SizeOfOneDataBlock = 8;
- MissedDataArray = 4108016;
- SyndromeOfCalculation = 28A80A29C131A80000016;
- BackupDataMatrix = A84280016;
- DataBlocks = UU0203040506UU08090A0BUU0D0E0F10111213$_{16}$.

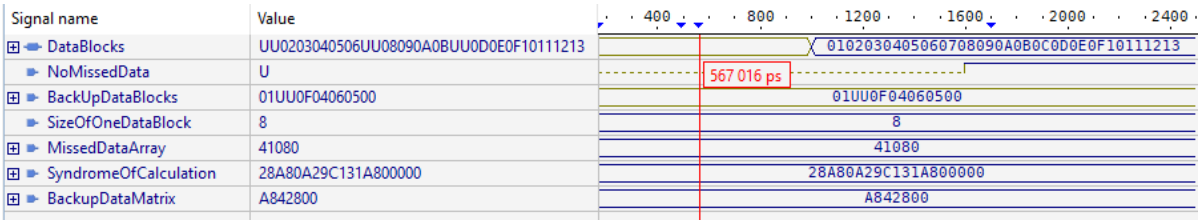

Рисунок 3.20 – Перша симуляція та відлагодження компоненти MissedDataGeneratorUnit.vhd

На рисунку 3.21 проілюстрована робота приладу за наступних вхідних даних:

- NoMissedData = 0;
- BackUpDataBlocks = 061D1C1315140C16;
- SizeOfOneDataBlock = 8;
- MissedDataArray = 4108016;
- SyndromeOfCalculation = 414F6F907987BF0000016;
- BackupDataMatrix = 200A100$_{16}$;
- DataBlocks = UU0203040506UU08090A0BUU0D0E0F10111213$_{16}$.

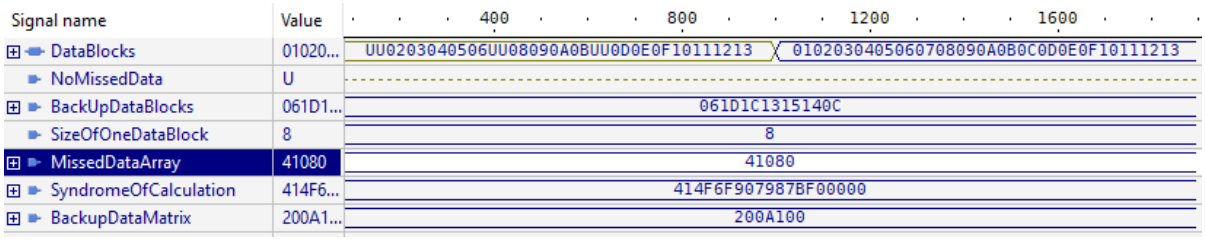

Рисунок 3.21 – Друга симуляція та відлагодження компоненти MissedDataGeneratorUnit.vhd

На рисунку 3.22 проілюстрована робота приладу за наступних вхідних даних:

- NoMissedData = 1;
- SizeOfOneDataBlock = 8;
- DataBlocks = $0102030405060708090A0B0C0D0E0F10111213_{16}$.

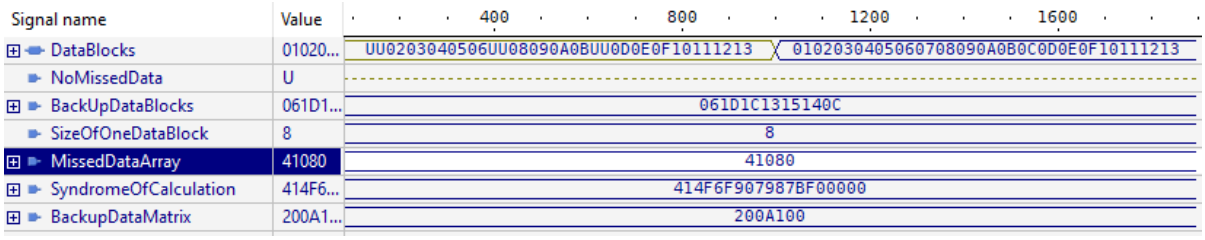

Рисунок 3.22 – Третя симуляція та відлагодження компоненти MissedDataGeneratorUnit.vhd

Компонент Model.vhd, є необхідною частиною моделювання, адже виконує важливу роль компонуючою сутністю. Всередині такої сутності, як правило відбувається об’явлення всіх необхідних дочірніх елементів, та безумовно, як сигнали об’являються змінні, які виконують роль конекторів між дочірніми об’єктами.

Всередині моделі наявні такі конектори:

- OrthogonalMatrix – зберігає дані ортогональної матриці;
- BackupDataMatrix – використовується задля передачі синдрому резервних блоків, що одночасно є вирішенням системи рівнянь та є необхідним для відновлення втрачених даних.

- ErrorMatrix – використовується задля передачі підматриці в якій є лише елементи, що знаходяться у стовпцях втрачених блоків даних та рядках тих резервних блоків що залишилися.
- SyndromeOfCalculation – використовується задля передачі синдрому для обрахування втрачених блоків даних;
- MatrixA – використовується задля передачі інформації згенерованої матриці формування резервних блоків;
- NumberOfMissedData – використовується задля передачі значення про кількість помилок;
- Error – використовується для передавання сигналу про помилку, або її відсутність;
- NoMissedData – використовується для передавання сигналу про відсутність помилок;

Окрім того, так як цей пристрій репрезентує систему в цілому, то всі його виходи є аналогічними виходам, що має система, а саме:

- BackUpDataBlocks – вхід;
- NumberOfDataBlocks – вхід;
- NumberOfBackUpBlocks – вхід;
- SizeOfOneDataBlock – вхід;
- Initialize – вхід;
- DataBlocks – вхід/вихід;
- ErrorFound – вихід.

Приклад роботи даної сутності наведені на Рисунку 3.23. В якості даних використовувались наступні значення:

- BackUpDataBlocks = 061D1C1315140C16;
- NumberOfDataBlocks = 19;
- NumberOfBackUpBlocks = 7;
- SizeOfOneDataBlock = 8 bit;
- Initialize = 1;
- DataBlocks = UU0203040506UU08090A0BUU0D0E0F10111213$_{16}$.

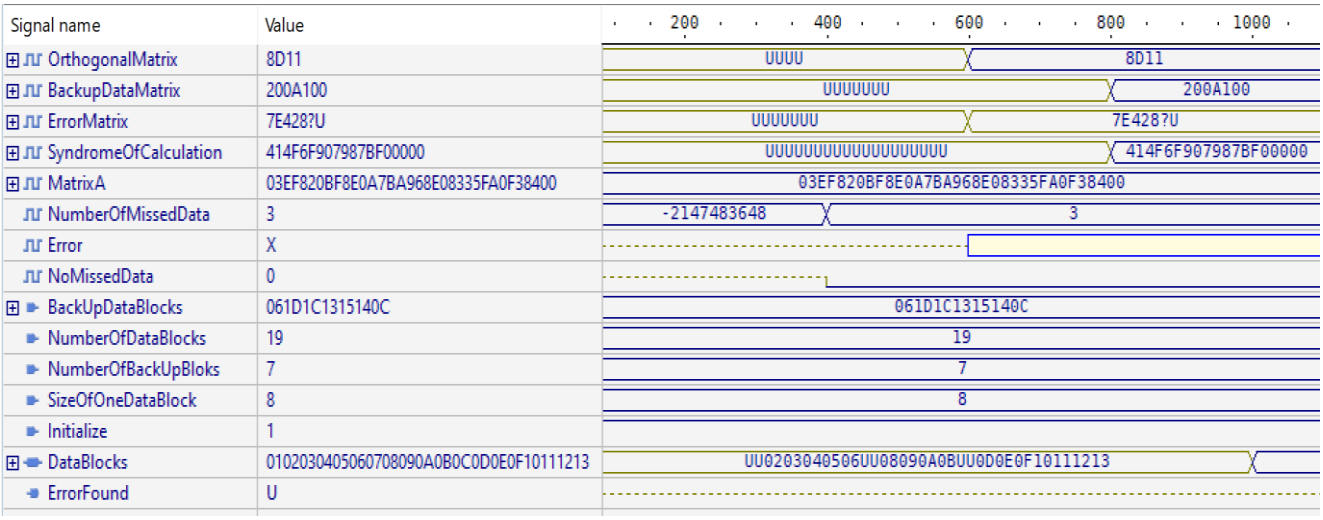

Рисунок 3.23 – Симуляція та відлагодження компоненти Model.vhd

ВИСНОВКИ ДО РОЗДІЛУ 3

В даному розділі було описано розроблений спосіб відновлення блоків даних у системі із розподіленим зберіганням на основі методу, вибраного в розділі 2. Було приділено увагу використання даної системи на різних етапах проходження даних.

Наведено детальний опис функціональних частин системи та принципи їх роботи. Було проведено тестування та відлагодження компонентів системи із запропонованим способом, та наведені приклади їх роботи.

РОЗДІЛ 4 АНАЛІЗ РОБОТИ ТА ПОВЕДІНКИ СИСТЕМИ ІЗ ЗІСТОСУВАННЯМ РОЗРОБЛЕНОГО СПОСОБУ

Для відпрацювання, відлагодження системи із застосування розробленого способу та надбання даних для аналізу її роботи було прийнято рішення провести низку симуляцій з тестовими даними. Для вирішення цього питання було використано ПЗ Active-HDL.

В системі, що буде симулюватися, для зручності перегляду отриманих результатів та легкості їх перевірки було використано наступні налаштування:

- розмір одного блоку даних – 8 біт;
- кількість максимальних блоків даних – 19;
- кількість максимальних резервних блоків даних – 7;
- кількість блоків даних що групуються у вузли даних – 4.

4.1 Тестування роботи системи при різних ситуаціях

Для проведення ґрунтовного аналізу поведінки даної системи було змодельовано ситуації в яких вона мала себе так чи інакше повести. Тут описано коротко про ітерації моделювання, про використання тих чи інших тестових даних, та описи результатів.

Задля першого моделювання методу було використано такі налаштування системи:

- NumberOfDataBlocks =13;
- NumberOfBackUpDataBlocks = 6.

На рисунку 4.1 показано симуляцію, під час якої були введені наступні вхідні данні:

- BackUpDataBlocks = 070F0F0F0A0E0016;
- DataBlocks = UUUU03040506UU08090A0BUU0D0E0F10111213$_{16}$.

В результаті під час роботи було отримано помилку, що й було передбачувано, адже система не може відновити дані з більш ніж трьох різних сховищ інформації.

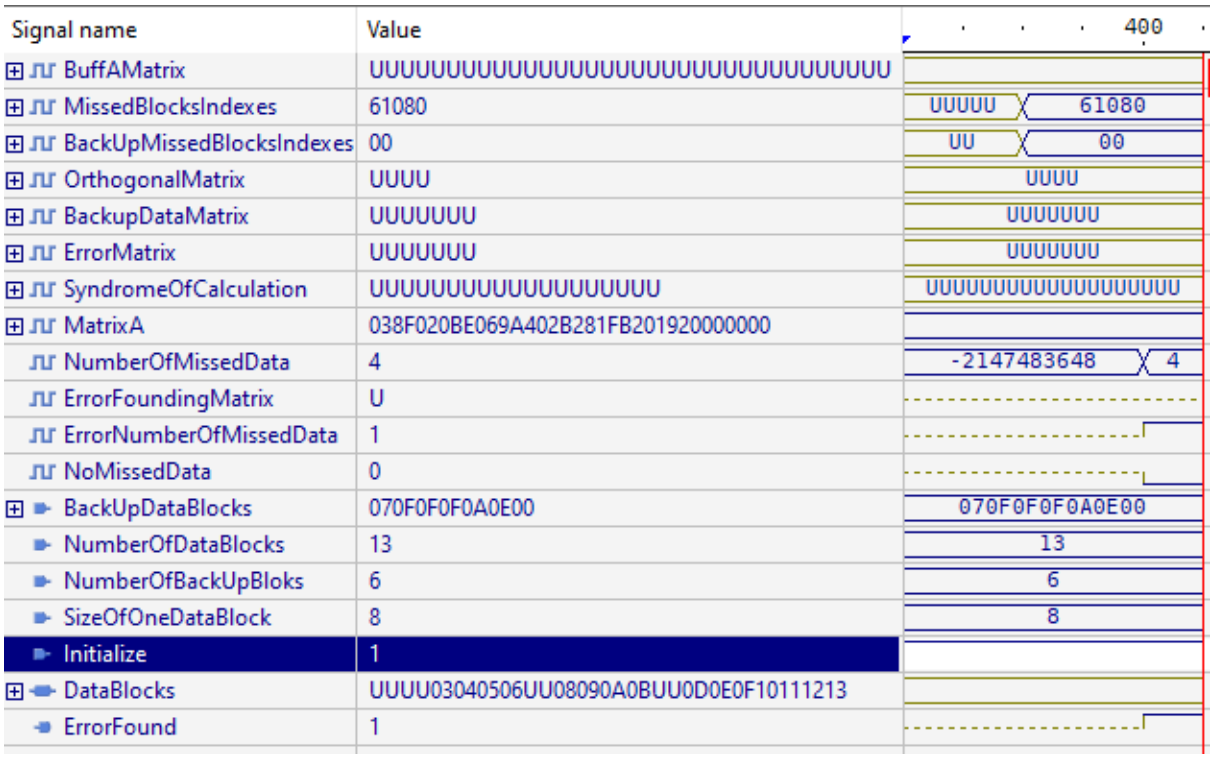

Рисунок 4.1 – Перша симуляція системи (13,6)

В результаті під час роботи було отримано помилку, що й було передбачувано, адже система не може відновити дані з більш ніж трьох різних сховищ інформації.

На рисунку 4.2 зображені результати симуляції, вхідними даними під час якої були:

- BackUpDataBlocks = 070F0F0F0A0E0016;
- DataBlocks = 01UU03040506UU08090A0BUU0D0E0F10111213$_{16}$.

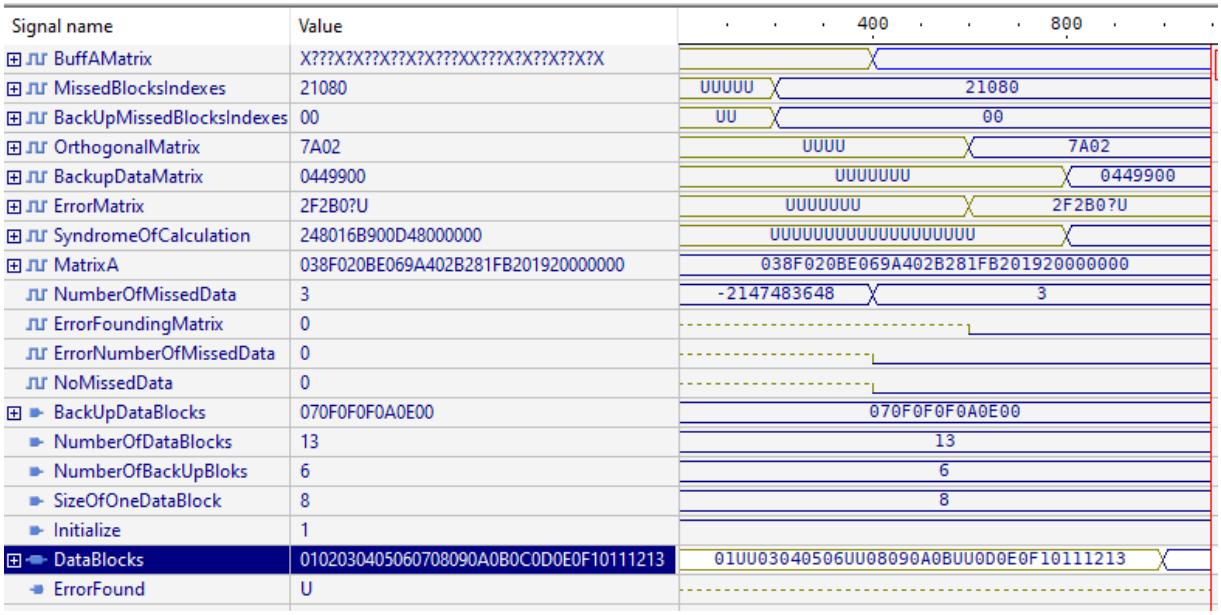

Рисунок 4.2 – Друга симуляція системи (13,6)

Метод чудово впорався з поставленою задачею, відновивши всі втрачені блоки.

Наступну симуляцію зображено на рисунку 4.3, де в якості вхідних даних було використано:

- BackUpDataBlocks = 070F0F0F0A0E0016;
- DataBlocks = UUUUUUUU05060708090A0C0D0E0F10111213$_{16}$.

Рисунок 4.3 – Третя симуляція системи (13,6)

Моделювання показало, що система із використанням розробленого способу використала одну з своїх функцій та змогла відновити дані з одного з кластерів інформації.

Останню симуляцію цієї конфігурації системи зображено на рисунку 4.4. Вхідні данні були наступні:

- BackUpDataBlocks = UUUU0F0F0A0E0016;
- DataBlocks = 01UU03040506UU08090A0UUC0D0E0F10111213$_{16}$.

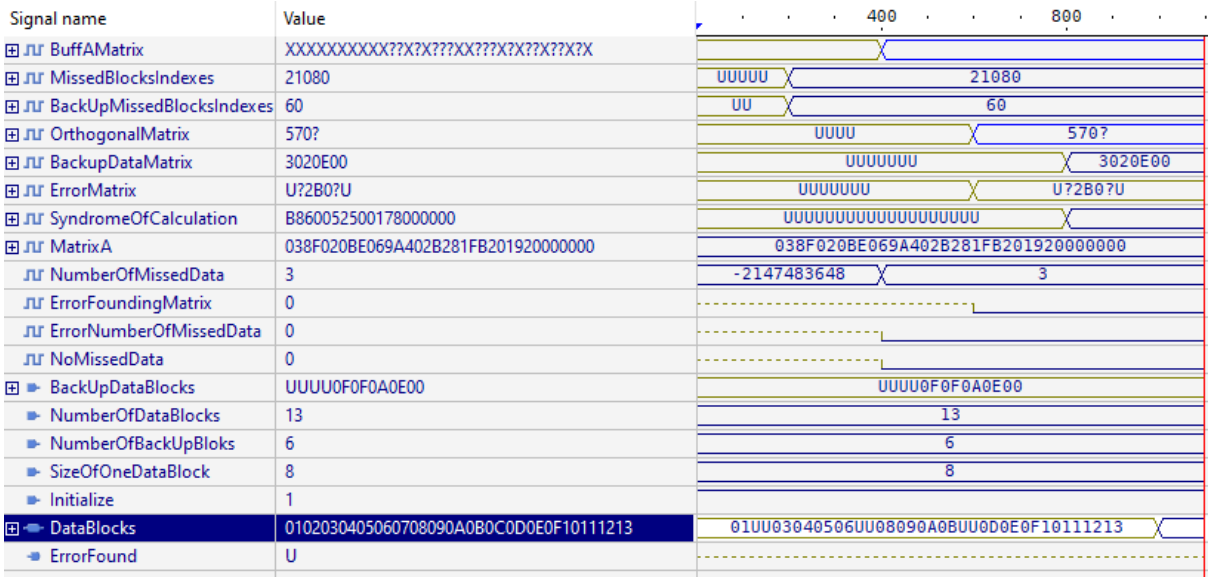

Рисунок 4.4 – Четверта симуляція системи (13,6)

Для моделювання другої конфігурації були обрані наступні значення налаштувань:

- NumberOfDataBlocks =15;
- NumberOfBackUpDataBlocks = 7.

Перше моделювання цієї системи в даній конфігурації можна спостерігати на рисунку 4.5. Тут в якості вхідних даних використовувались наступні значення:

- BackUpDataBlocks = 04040103060D0E;
- DataBlocks = 01UU03040506UU08090A0BUU0D0E0F10111213$_{16}$.

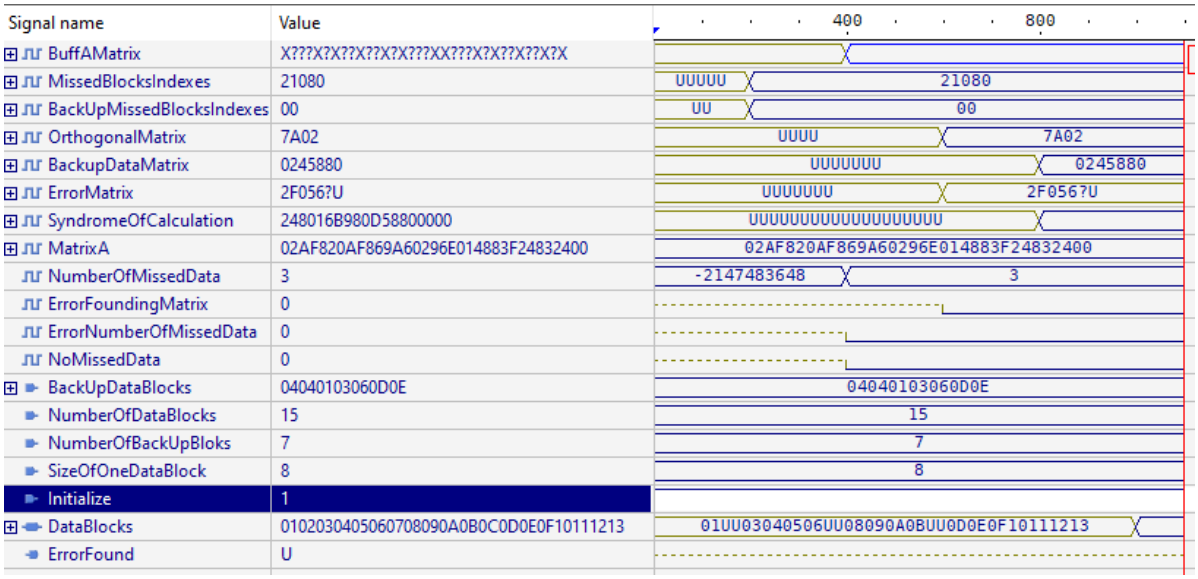

Рисунок 4.5 – Перша симуляція системи (15,7)

Система відновила данні, як і передбачалось.

Наступне моделювання зображене на рисунку 4.6, під час якого було використані наступні тестові значення:

- BackUpDataBlocks = 04040103060D0E;
- DataBlocks = UUUUUUUU05060708090A0C0D0E0F10111213$_{16}$.

Як і було задумано – система відновила втрачені інформаційні блоки з втраченого середовища.

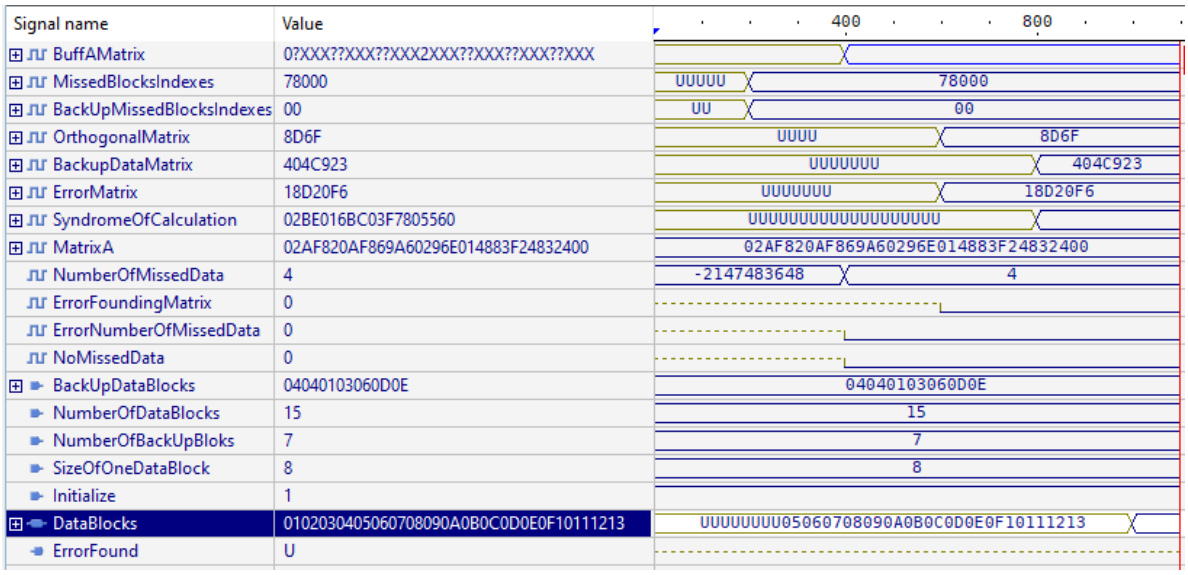

Рисунок 4.6 – Друга симуляція системи(15,7)

Останнє моделювання виконане в рамках цієї конфігурації системи проілюстрована на рисунку 4.7. Використані були наступні дані:

- BackUpDataBlocks = UU0401030UUD0E;
- DataBlocks = 01UU03040506UU08090A0BUU0D0E0F10111213$_{16}$.

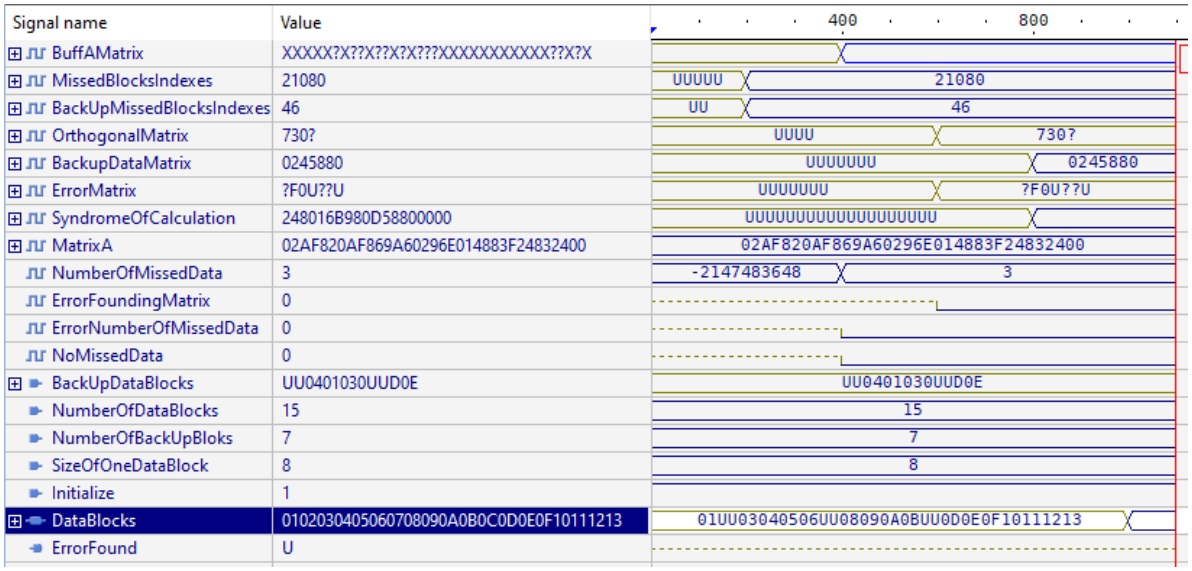

Рисунок 4.7 – Третя симуляція системи(15,7)

З огляду на вхідні дані, було втрачено аж три резервних блоки даних, та три інформаційних, однак метод зміг відновити ці дані.

Для моделювання третьої конфігурації були обрані наступні значення налаштувань:

- NumberOfDataBlocks =16;
- NumberOfBackUpDataBlocks = 7.

При першому моделюванні, що зображено на рисунку 4.8, були використані наступні тестові дані:

- BackUpDataBlocks = 04041B01161DO616;
- DataBlocks = UUUU03040506UU08090A0BUU0D0E0F10111213$_{16}$.

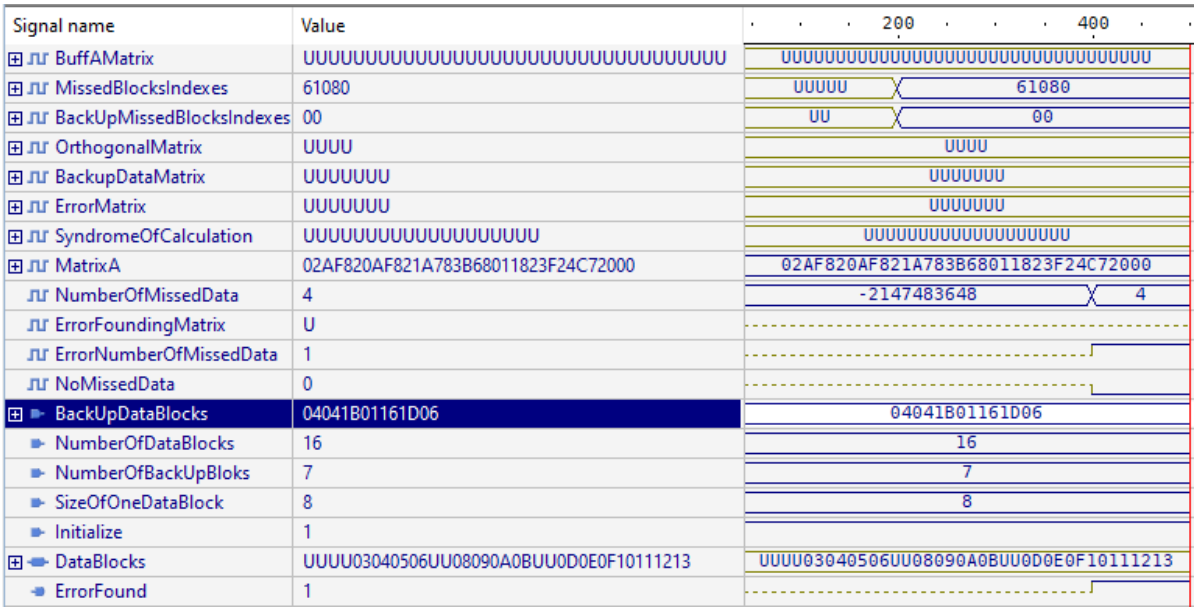

Рисунок 4.8 – Перша симуляція системи(16,7)

Результатами першого моделювання цієї конфігурації є помилка, адже система повідомляє про велику кількість помилок, що не розташована на одному вузлі інформації.

На рисунку 4.9 проілюстровано друге моделювання цієї конфігурації, вхідними даними були значення:

- BackUpDataBlocks = 04041B01161DO616;
- DataBlocks = 01UU03040506UU08090A0BUU0D0E0F10111213$_{16}$.

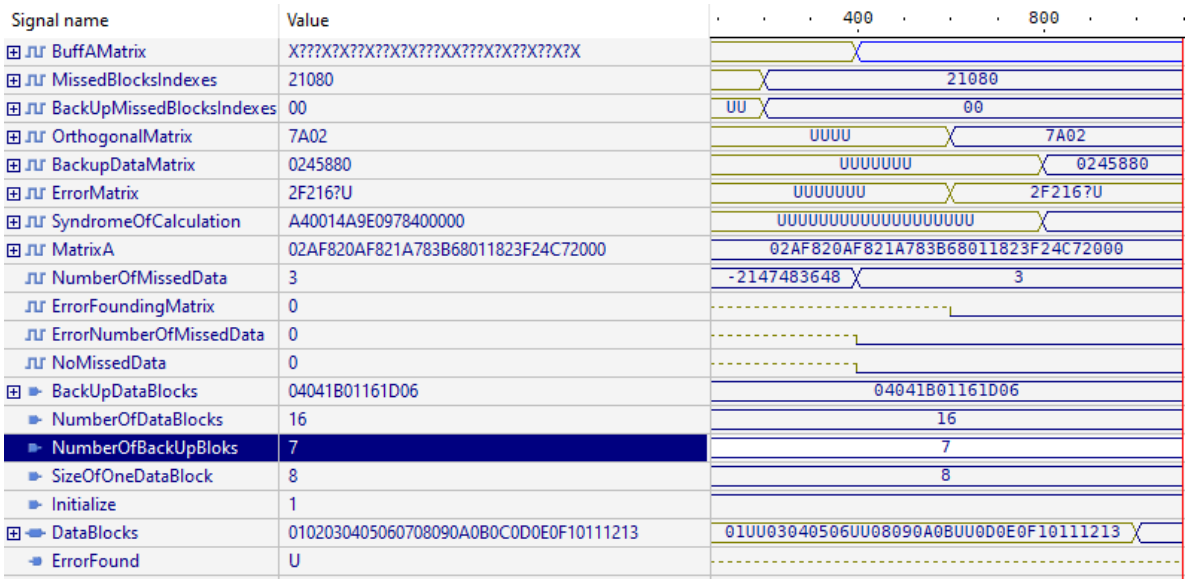

Рисунок 4.9 – Друга симуляція системи(16,7)

Метод зміг відновити три втрачені блоки даних, як і передбачалось.

Під час наступної симуляції, яку можна побачити на рисунку 4.10, були використані дані:

- BackUpDataBlocks = 04041B01161DO616;
- DataBlocks = UUUUUUUU05060708090A0C0D0E0F10111213$_{16}$.

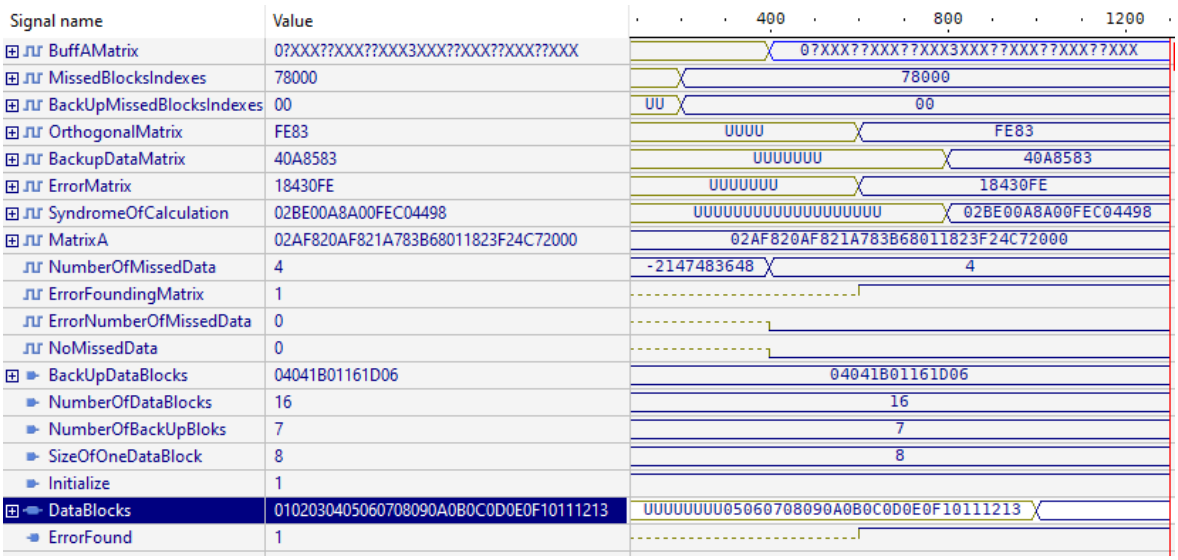

Рисунок 4.10 – Третя симуляція системи(16,7)

Метод успішно відновив кластер інформації.

Під час останньої симуляції цієї конфігурації, зображеної на рисунку 4.11, були використані дані:

- BackUpDataBlocks = 04UUUUUUU61DO616;
- DataBlocks = UUUUUUUU05060708090A0C0D0E0F10111213$_{16}$.

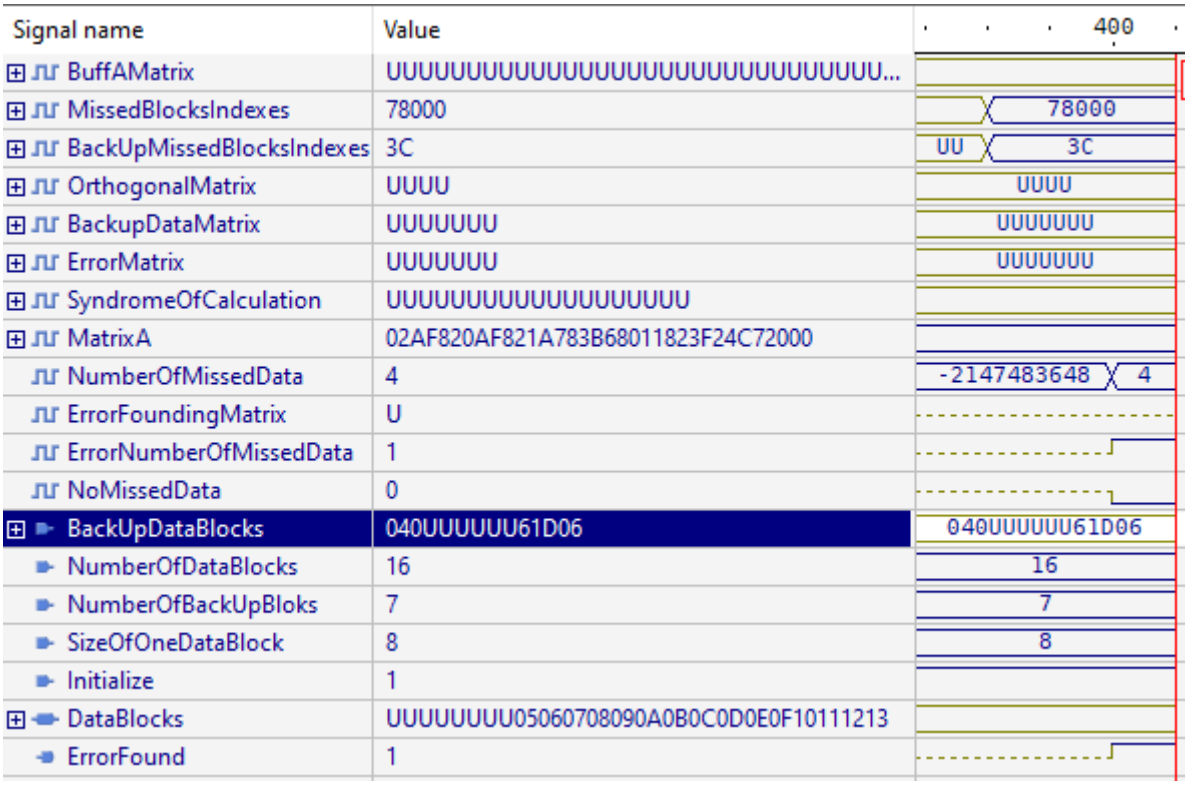

Рисунок 4.11 – Четверта симуляція системи(16,7)

Через велику кількість втрачених резервних блоків даних метод не зміг відновити кластер інформації.

Для моделювання четвертої конфігурації системи були обрані наступні значення:

- NumberOfDataBlocks =18;
- NumberOfBackUpDataBlocks = 7.

Задля першої симуляції даної конфігурації, результати якої зображені на рисунку 4.12 було використано наступні значення даних:

- BackUpDataBlocks = UU061F0600UUUU16;
- DataBlocks = $01UU03040506UU08090A0BUU0D0E0F10111213_{16}$.

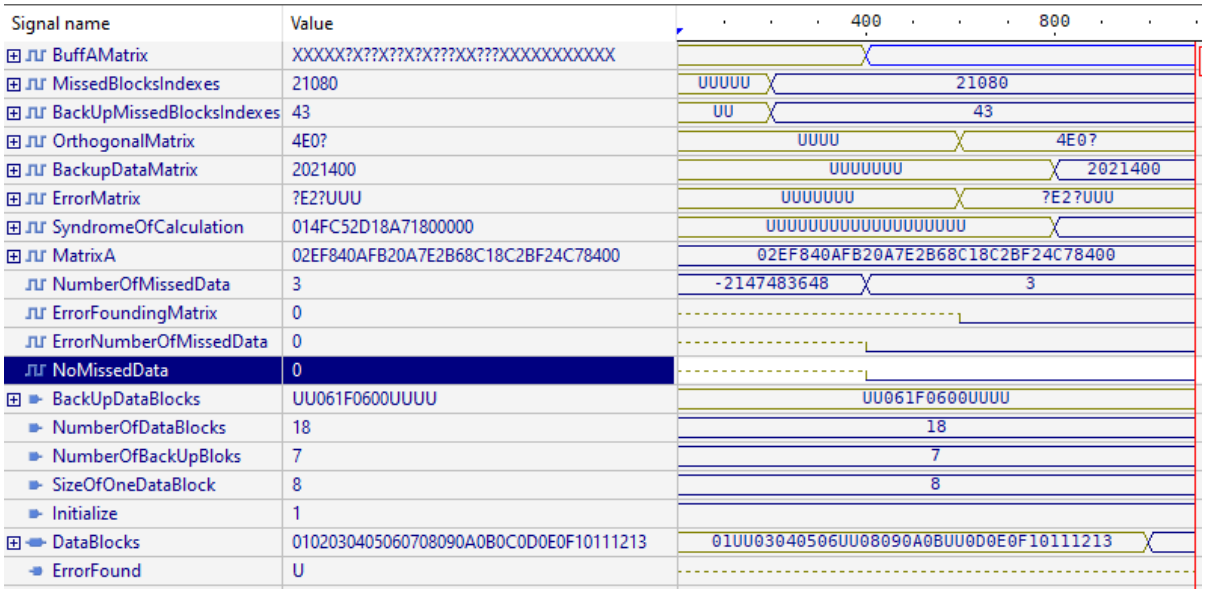

Рисунок 4.12 – Перша симуляція системи(18,7)

Як результат, система відновила втрачені блоки даних навіть при втрачених трьох резервних блоках інформації.

Під час другої симуляції зображеної на рисунку 4.13 було використано наступні значення:

- BackUpDataBlocks = UUUU1F0600UUUU16;
- DataBlocks = 01UU03040506UU08090A0BUU0D0E0F10111213$_{16}$.

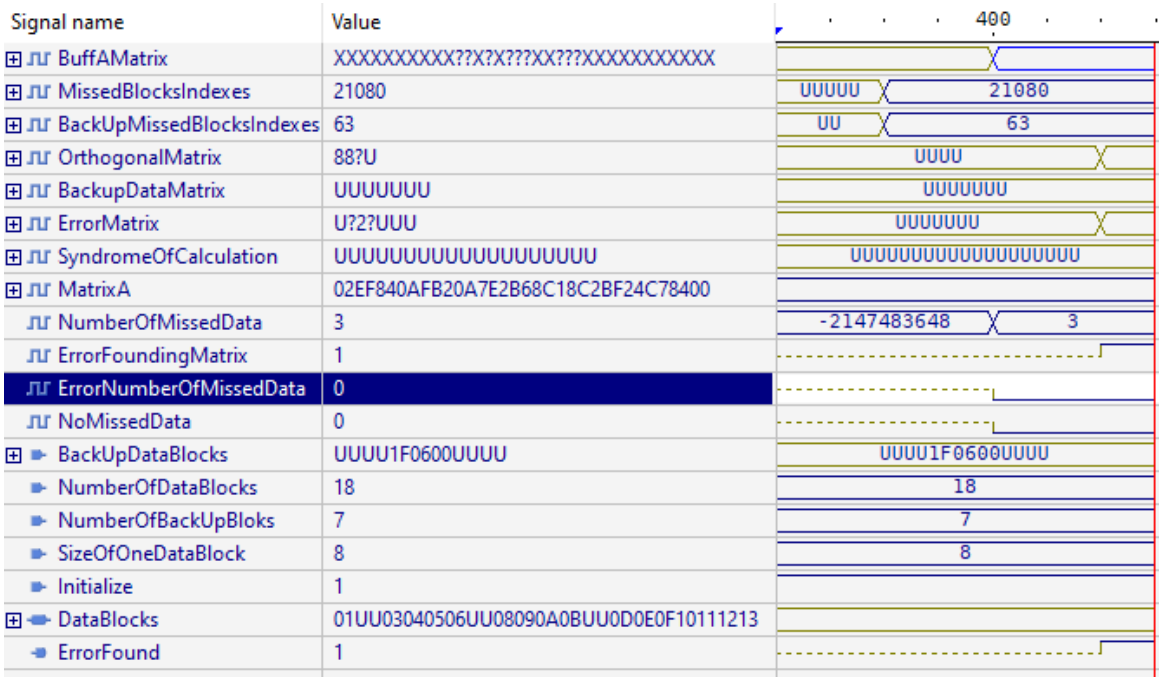

Рисунок 4.13 –Друга симуляція системи(18,7)

Як результат можна побачити, що при великій втраті резервних блоків існує ситуація, як в нашому випадку, коли знайти ортогональну підматрицю в матриці помилок неможливо. Саме про це ми і отримали сповіщення.

На рисунку 4.14 зображено процес симуляції, під час якого використовувались такі дані:

- BackUpDataBlocks = 10061F06001D0D 16;
- DataBlocks = 0102030405060708090A0C0D0E0F10UUUUUU$_{16}$

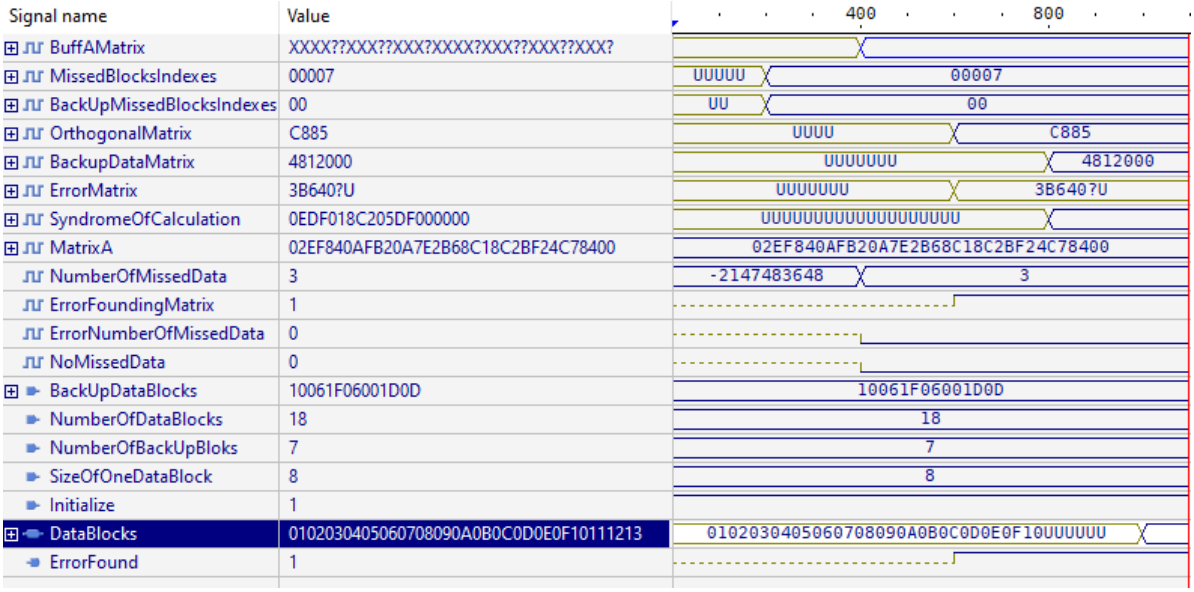

Рисунок 4.14 – Третя симуляція системи(18,7)

Як результат були відновлені всі блоки даних на останньому кластері інформації.

Для останньої симуляції системи при заданій конфігурації, що зображена на рисунку 4.15, використовували наступні дані:

- BackUpDataBlocks = 10061F06001D0D 16;
- DataBlocks = 0102UU0405060708090A0C0D0E0F10UUUUUU $_{16.}$

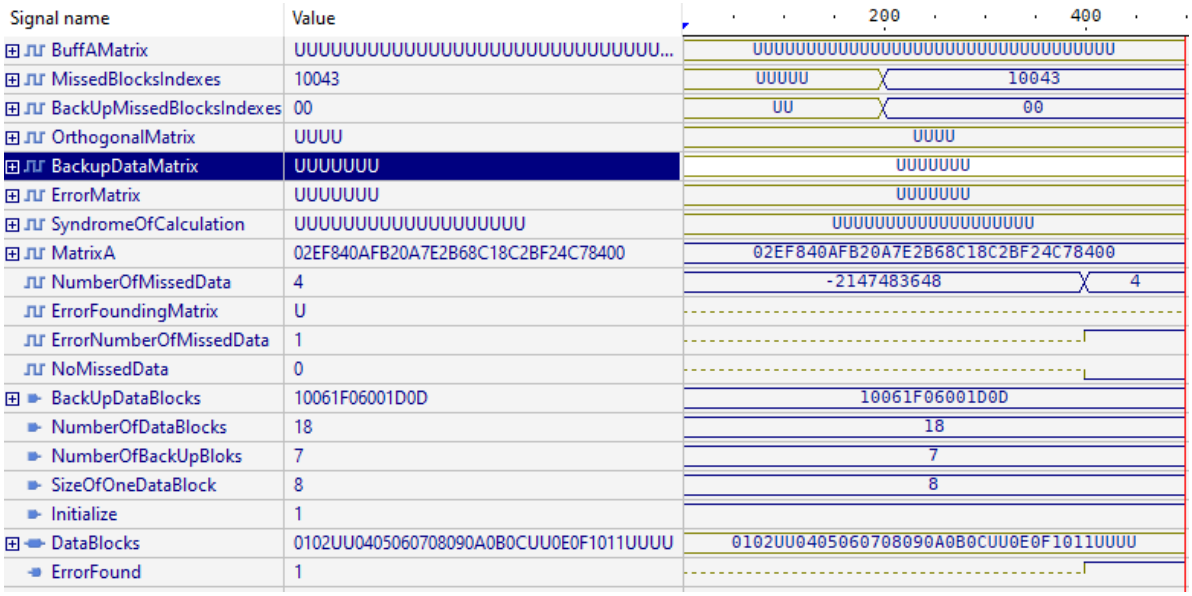

Рисунок 4.15 – Четверта симуляція системи(18,7)

Як результат ми отримали помилку, адже намагались відновити 4 блоки даних на різних середовищах, що і передбачалось.

Для моделювання останньої конфігурації були обрані наступні значення налаштувань:

- NumberOfDataBlocks =19;
- NumberOfBackUpDataBlocks = 7.

На рисунку 4.16 зображено симуляцію системи при наступних значеннях:

- BackUpDataBlocks = 061D1C1315140C16;
- DataBlocks = 0102UU0405060708090A0C0D0E0F10UUUUUU $_{16}$.

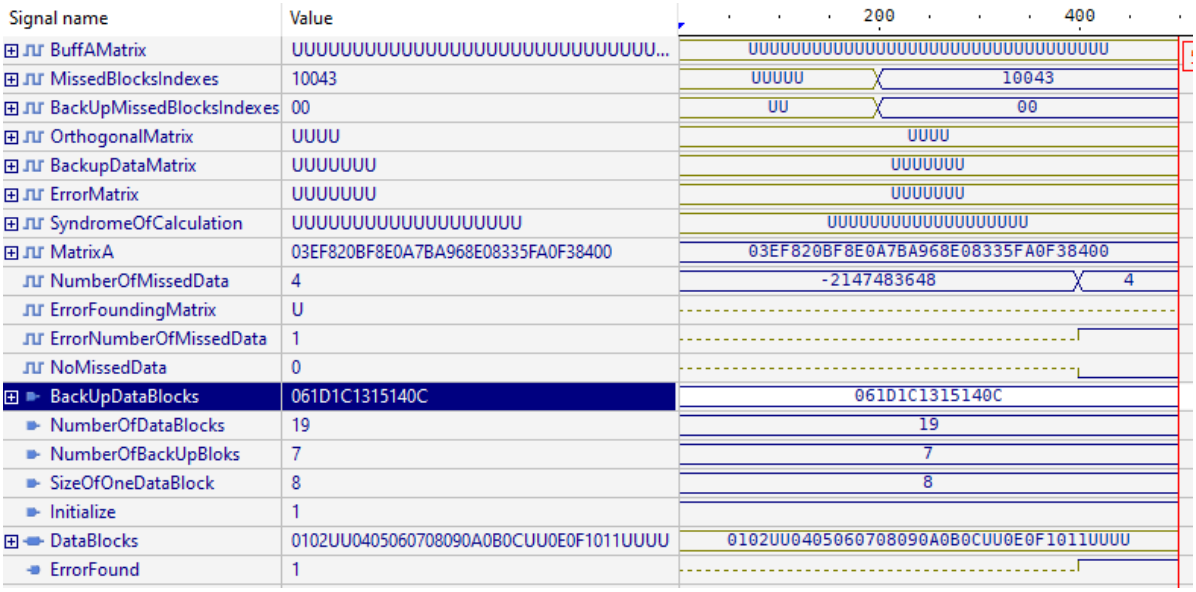

Рисунок 4.16 – Перша симуляція системи(19,7)

Як результат ми отримали таку ж помилку як і при минулій симуляції, адже система на такі випадки не розрахована.

При наступній симуляції, що зображена на рисунку 4.17, використовувались наступні значення:

- BackUpDataBlocks = 061D1C1315140C16;
- DataBlocks = 0102030405060708UUUUUUUU0D0E0F10111213 16.

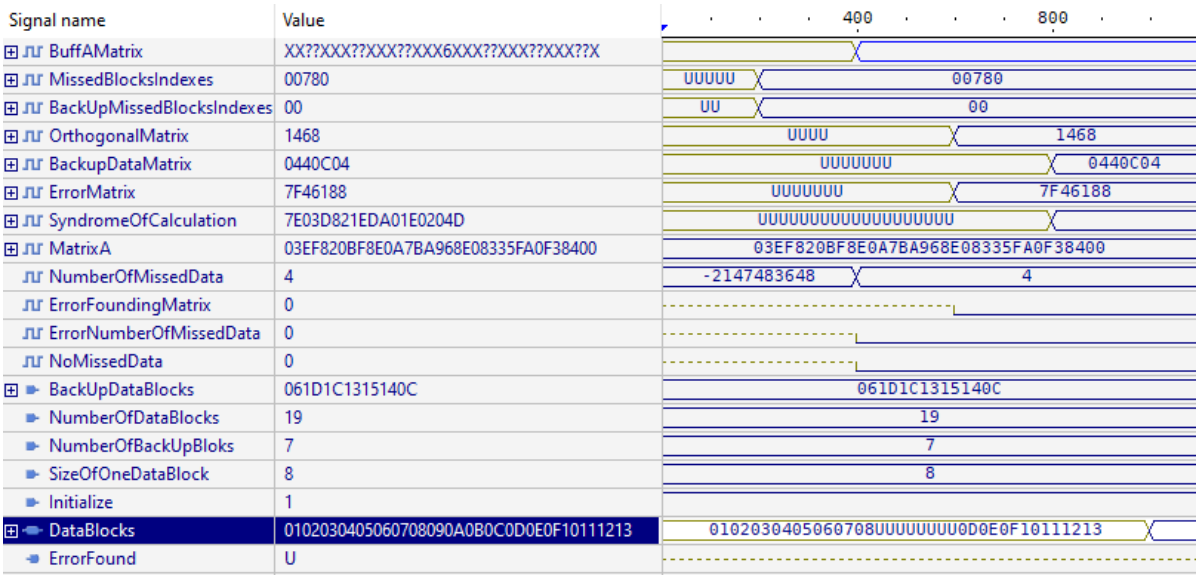

Рисунок 4.17 – Друга симуляція системи(19,7)

Як результат, були відновленні всі дані з втраченого кластеру, що і передбачалось.

Під час третьої симуляції, зображеної на рисунку 4.18, були використані наступні значення:

- BackUpDataBlocks = 06UUUU1315140C16;
- DataBlocks = 0102030405060708090A0C0D0E0F10UUUUUU $_{16}$.

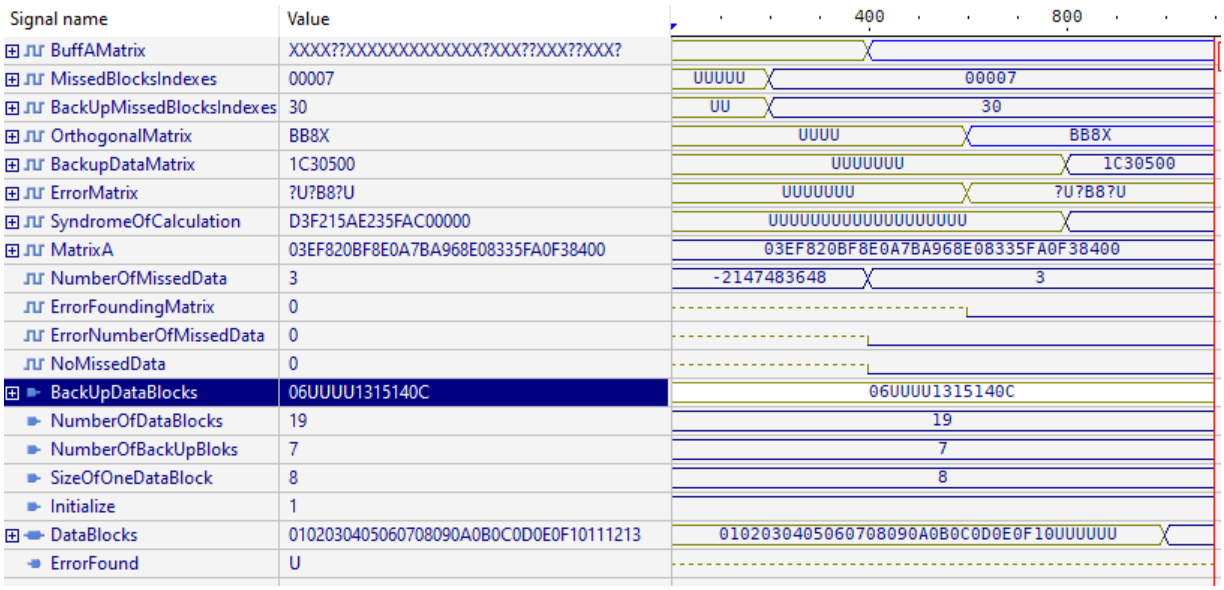

Рисунок 4.18 – Третя симуляція системи(19,7)

Дані були відновлені з втраченого вузла інформації, що було цілком передбачуваним.

Під час останньої симуляції цієї конфігурації системи, що зображена на рисунку 4.19, були використані наступні значення:

- BackUpDataBlocks = 06UUUU1315140C16;
- DataBlocks = UUUU03040506UU08090A0BUU0D0E0F10111213$_{16}$.

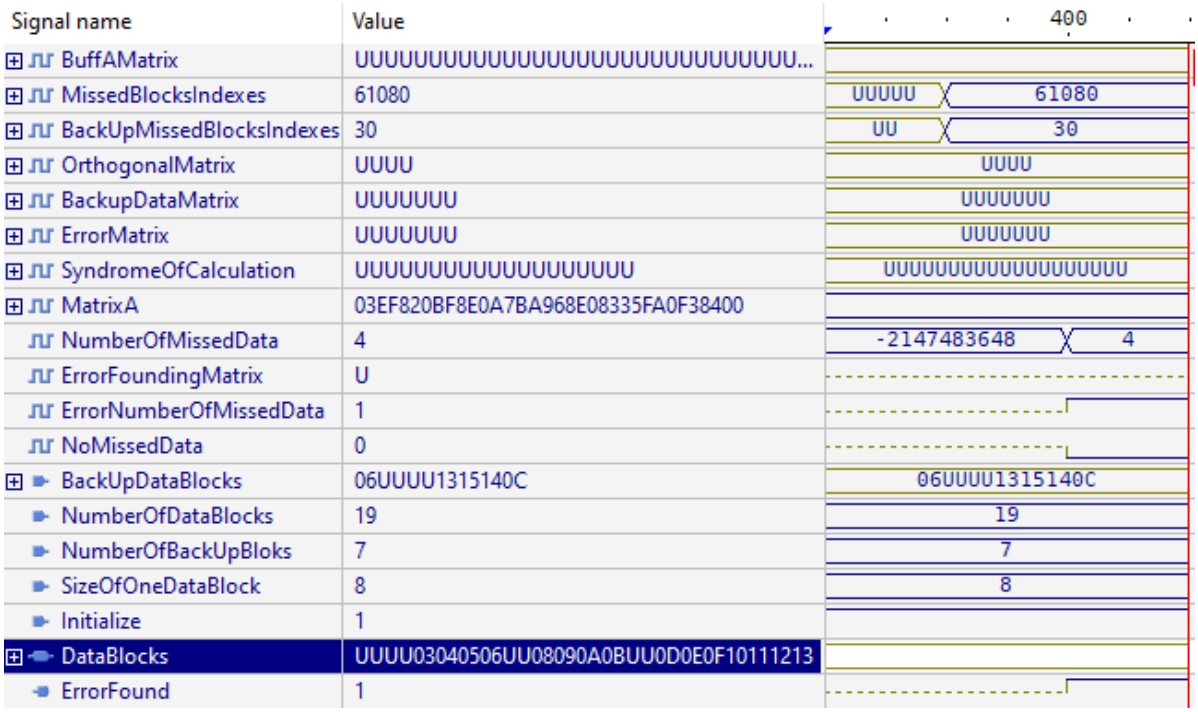

Рисунок 4.19 – Четверта симуляція системи(19,7)

Як результат можна відзначити, що через втрату більше трьох накопичувачів інформації системі не вдалося відновити втрачені дані.

Також особливо треба звернути увагу на симуляції генерації матриці формування резервних блоків даних та створення резервних блоків даних.

На рисунку 4.20 зображено симуляцію з використанням наступної інформації:

- NumberOfDataBlocks =15;
- NumberOfBackUpDataBlocks = 7;
- Initialize – 1;
- DataBlocks = $0102030405060708090A0B0C0D0E0F10111213_{16.}$

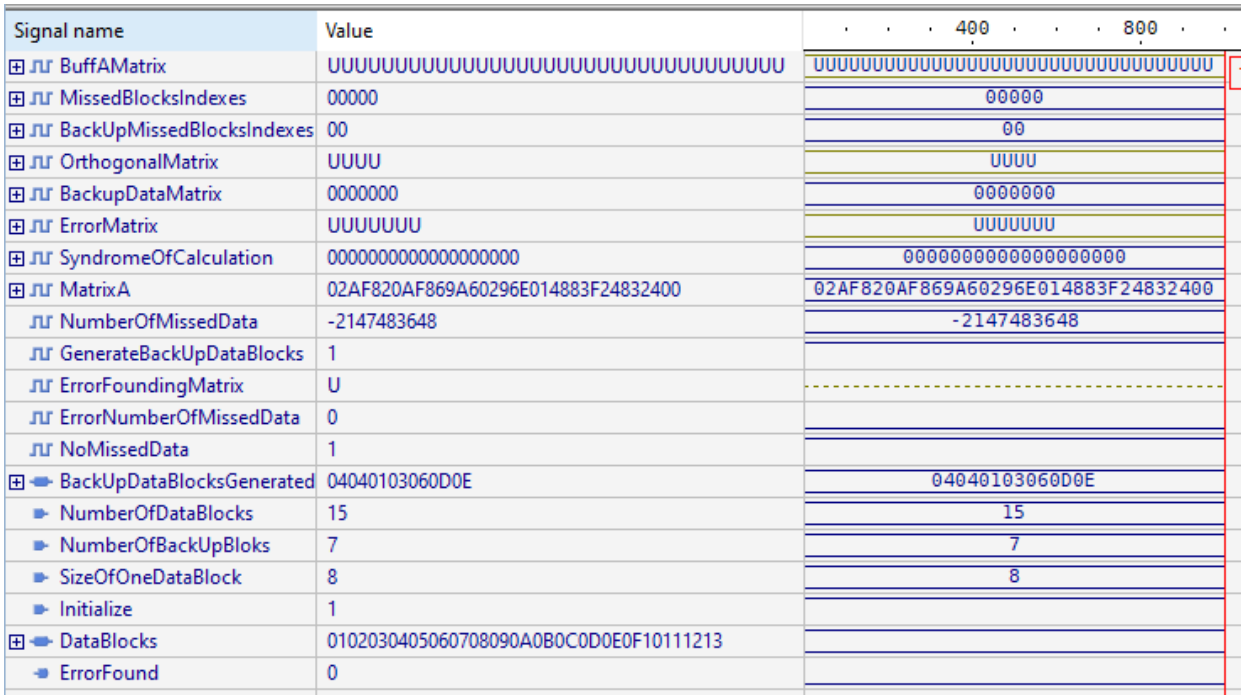

Рисунок 4.20 – генерація матриці А та резервних блоків за конфігурації(15,7)

Матриця та резервні блоки сформовані успішно та правильно, що було доведено попередніми симуляціями.

На рисунку 4.21 ілюструється симуляція з використанням наступної інформації:

- NumberOfDataBlocks =15;
- NumberOfBackUpDataBlocks = 6;
- Initialize – 1;
- DataBlocks = $0102030405060708090A0B0C0D0E0F10111213_{16}$.

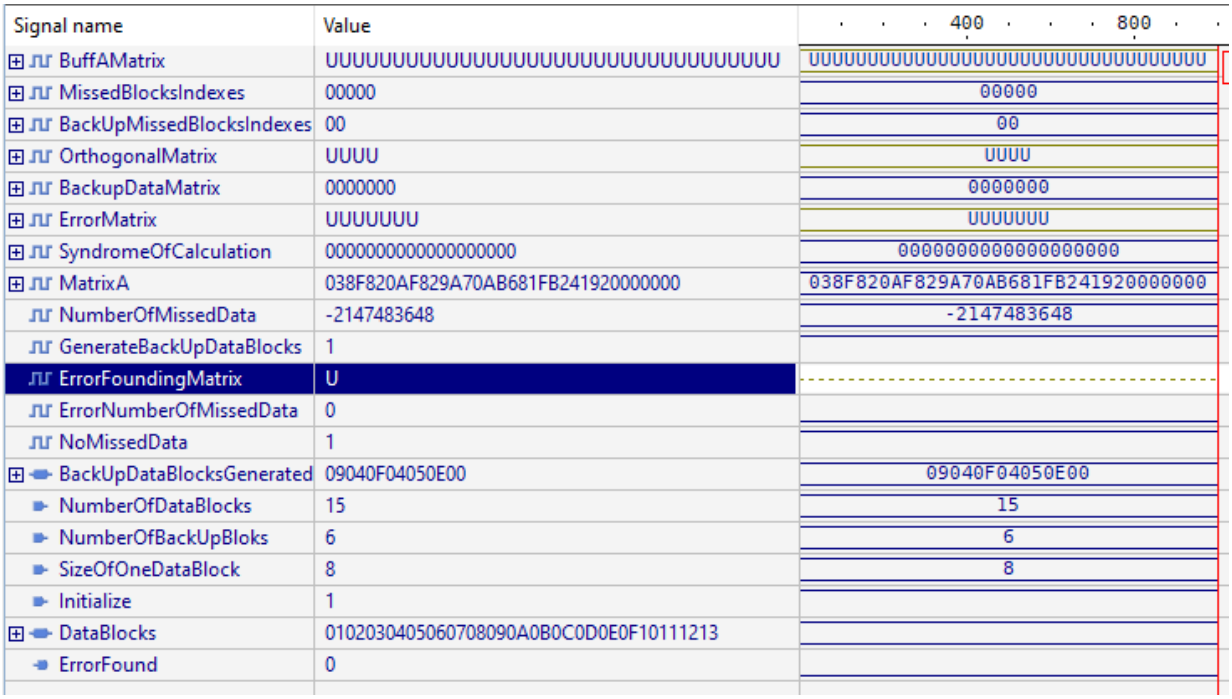

Рисунок 4.21 – генерація матриці А та резервних блоків за конфігурації(15,6)

Всі дані успішно створено та сформовано, в їх вірності дозволяють переконатися попередні симуляції.

На рисунку 4.22 зображено симуляцію з використанням наступної інформації:

- NumberOfDataBlocks =19;
- NumberOfBackUpDataBlocks = 7;
- Initialize – 1;
- DataBlocks = $0102030405060708090A0B0C0D0E0F10111213_{16}$.

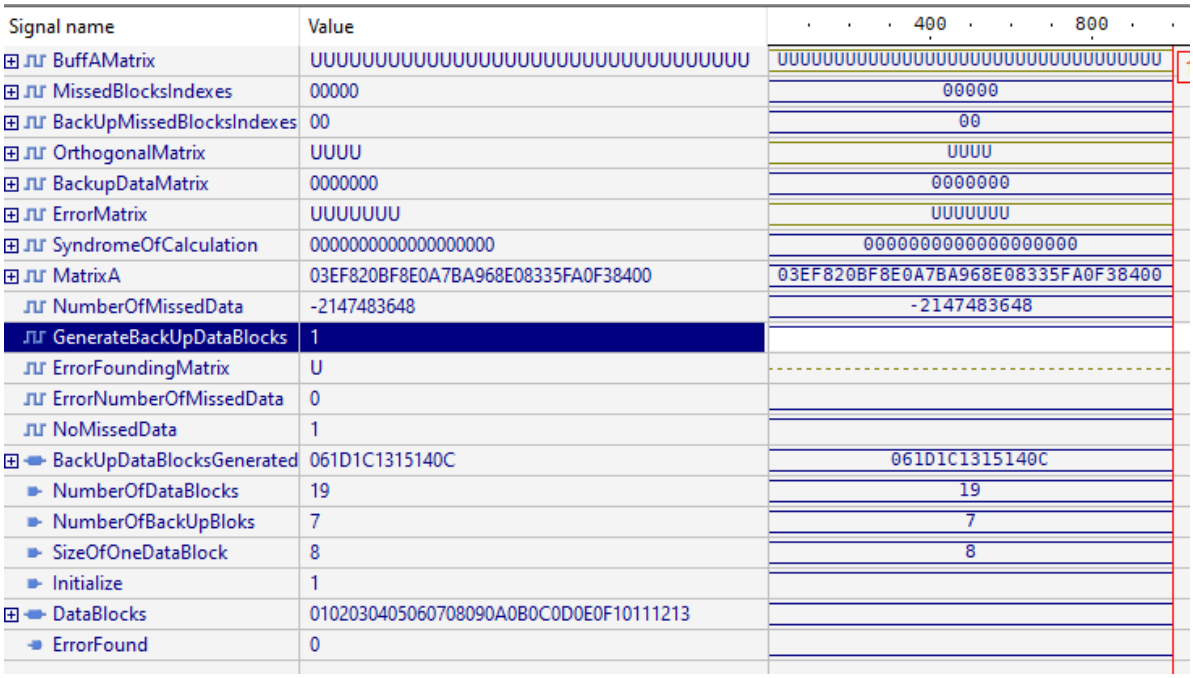

Рисунок 4.22 – генерація матриці А та резервних блоків за конфігурації(19,7)

Матриця та резервні блоки сформовані успішно та правильно, що було доведено попередніми симуляціями.

На рисунку 4.23 ілюструється симуляція з використанням наступної інформації:

- NumberOfDataBlocks =13;
- NumberOfBackUpDataBlocks = 6;
- Initialize – 1;
- DataBlocks = $0102030405060708090A0B0C0D0E0F10111213_{16}$.

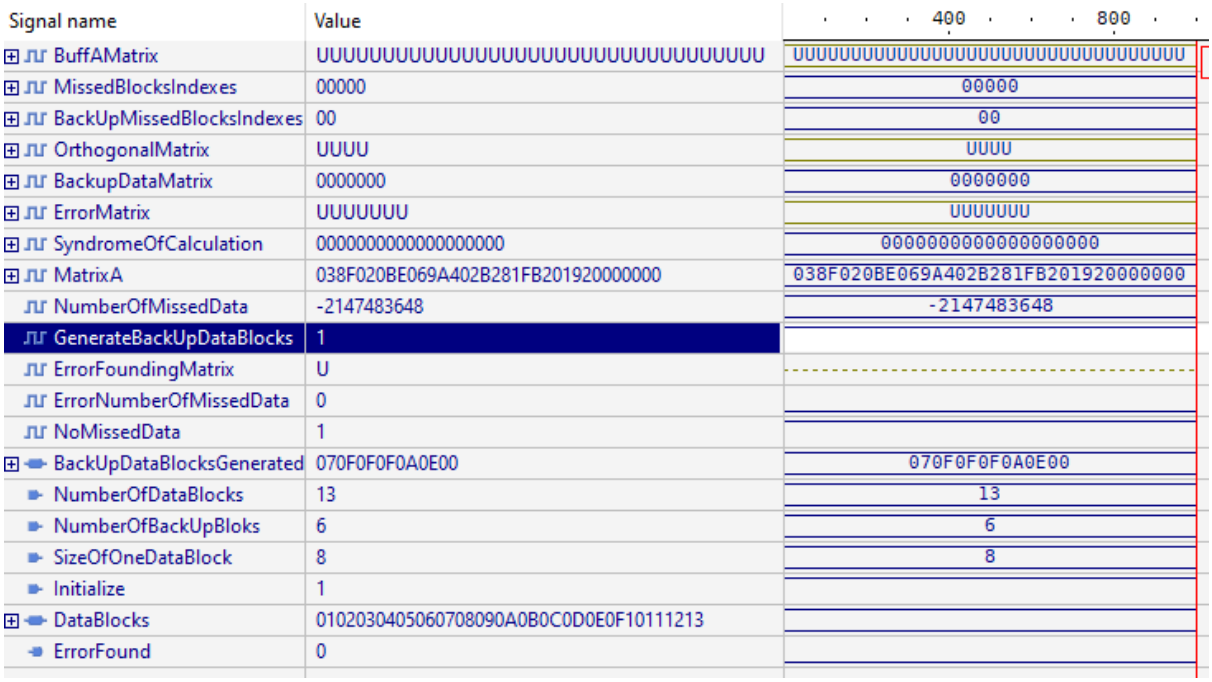

Рисунок 4.23 – генерація матриці А та резервних блоків за конфігурації(13,6)

Всі дані успішно створено та сформовано, в їх вірності дозволяють переконатися попередні симуляції.

На рисунку 4.24 зображено симуляцію з використанням наступної інформації:

- NumberOfDataBlocks =19;
- NumberOfBackUpDataBlocks = 7;
- Initialize – 1;
- DataBlocks = $0102030405060708090A0B0C0D0E0F10111213_{16}$.

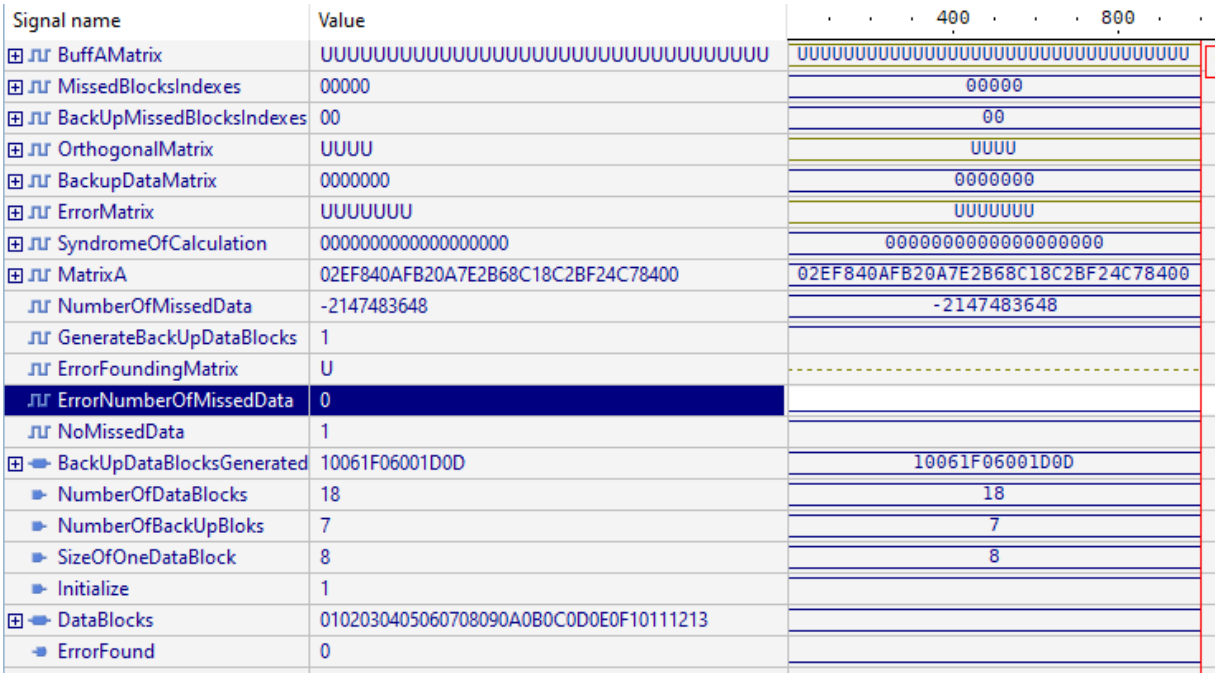

Рисунок 4.24 – генерація матриці А та резервних блоків за конфігурації(18,7)

Матриця та резервні блоки сформовані успішно та правильно, що було доведено попередніми симуляціями.

Також під час симуляції було проведено тестування створення матриць довільної форми, що задовольняють вимогам методу.

На рисунку 4.25 зображено симуляцію з використанням наступної інформації:

- NumberOfDataBlocks =19;
- NumberOfBackUpDataBlocks = 7;
- Initialize – 1;
- DataBlocks = $0102030405060708090A0B0C0D0E0F10111213_{16.}$

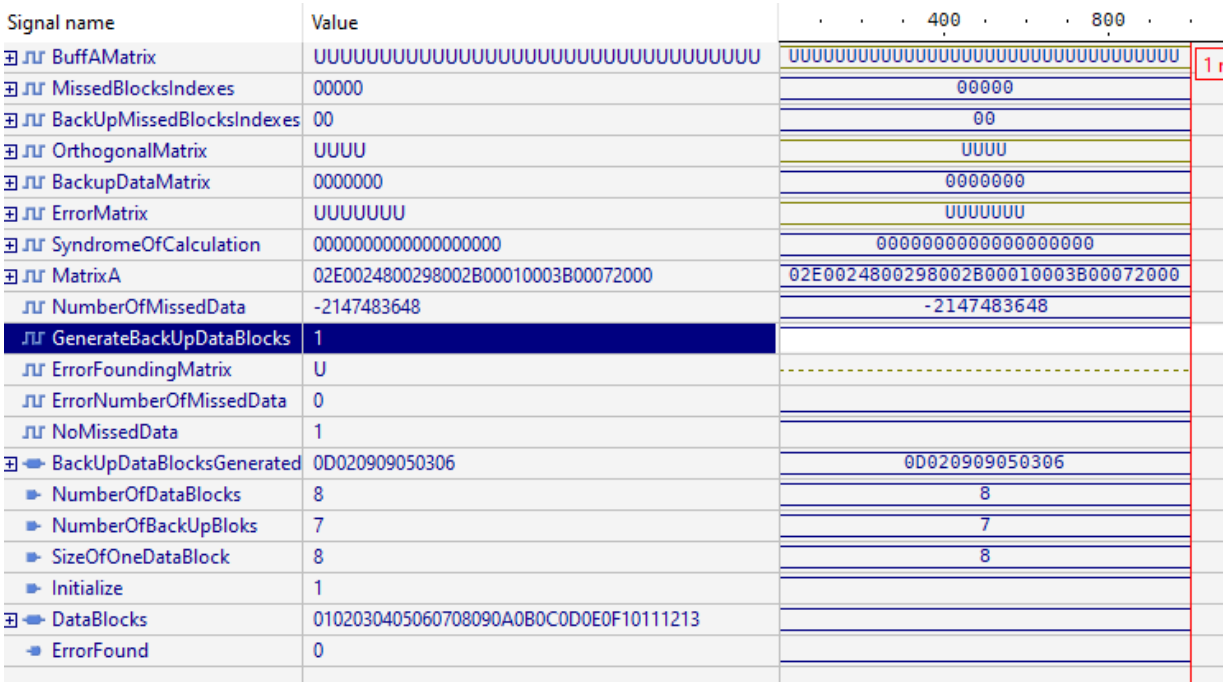

Рисунок 4.25 – генерація матриці А та резервних блоків за конфігурації(8,7)

Матриця успішно сформована, та перевірена на відповідність практичними підрахунками.

На рисунку 4.26 зображено симуляцію з використанням наступної інформації:

- NumberOfDataBlocks =14;
- NumberOfBackUpDataBlocks = 7;
- Initialize – 1;
- DataBlocks = $0102030405060708090A0B0C0D0E0F10111213_{16}$

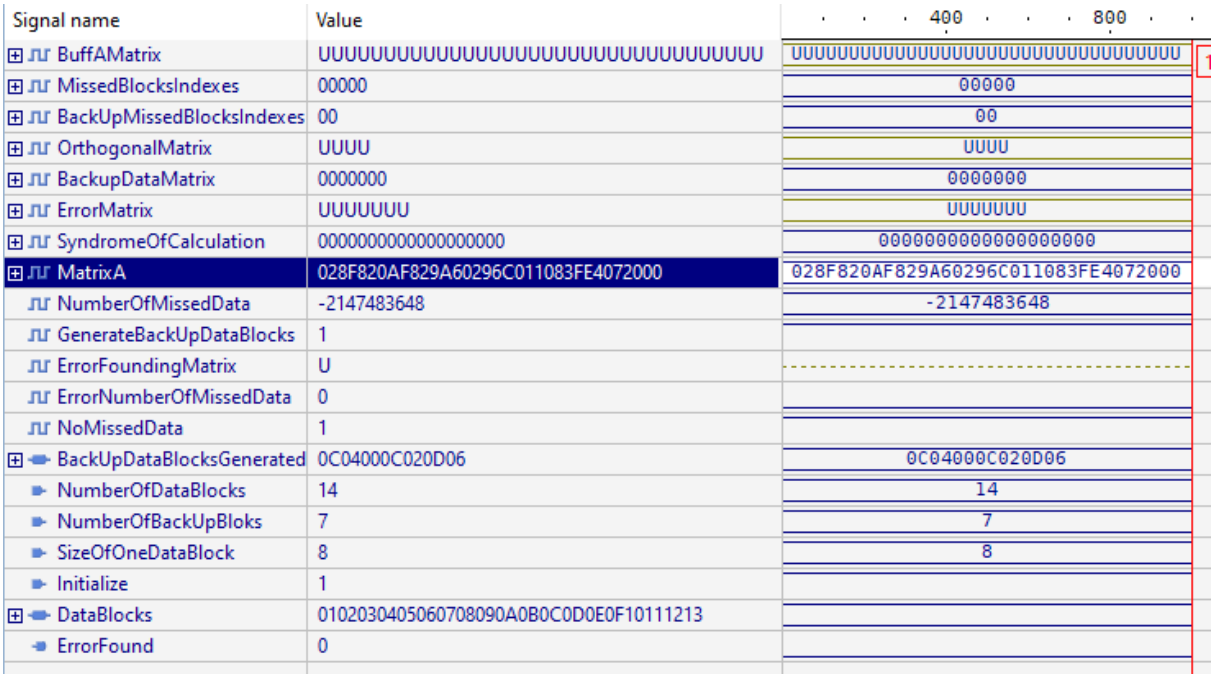

Рисунок 4.26 – генерація матриці А та резервних блоків за конфігурації(14,7)

4.2 Аналіз отриманих результатів роботи та поведінки системи із застосуванням розробленого способу

З огляду на наявні результати можна безперечно зазначити, що розроблена система виконує всі поставлені перед нею ключові завдання:

- Забезпечує гарантоване відновлення інформації з максимум трьох блоків даних, що розташовані віддалено;
- Забезпечує гарантоване відновлення будь-якої кількості блоків даних з одного середовища зберігання даних, за умови що всі помилки локалізовані;
- Відновлення інформації відбувається доволі швидко та непомітно для користувача;
- Істотно зменшується надлишковість порівняно з існуючими методами резервування даних на рівні вузлів [13].

Окрім того, варто зазначити, що система сама утворює матрицю формування резервних блоків та може сама формувати резервні блоки на основі згенерованої матриці. Звичайно, зберігання таких масивів резервної матриці

неможливе в системі, тому це питання потребує вирішення. Адже як правильно зберегти резервні блоки, для того, щоб їх також не спіткала доля втраченого вузлу даних, це не є тривіальна задача. Тим не менш, система із використанням розробленого способу є доволі самостійною і не потребує додаткових засобів для відновлення даних, лише самі дані та їх резервні дані.

Швидкість роботи даного пристрою зумовлена такими факторами:

1) Швидкістю зчитування даних, що прийшли до системи;
2) Швидкістю розв'язання задачі пошуку ортогональної матриці;
3) Швидкістю розв'язання системи лінійних рівнянь;
4) Швидкістю записування даних.

Також під час практичних досліджень слід відмітити, що при використанні даної системи є оптимальним використання групування по кількості $\mathrm{m}-2$. Адже в разі збереження в одному вузлі даних r блоків інформації може виникнути ситуація, коли кількість резервних блоків буде перевищувати мінімально необхідну кількість. З іншого боку, якщо робити об'єднання у вузли великої кількості то кількість резервних блоків вже буде зумовлена не кількістю даних в цілому, а кількістю збереження даних на одному сховищі. Тому використання $\mathrm{m}-2$ блоків даних на одному сховищі є оптимальним рішенням.

З огляду на наявні характеристики розробленої системи слід зрозуміти, що її ефективність може бути збільшена, на випадок того, що ми будемо використовувати прогнозоване віддалене резервування даних. Цілком зрозуміло, що досягти прогнозованості використання хмарного сховища даних від користувача, яким є звичайна людина доволі складно, адже структура та наповненість даними може бути абсолютно різна та й питання конфіденційності даних з кожним роком постає ще гостріше. Саме тому на використання прогнозованого резервування в практиці роботи хмарних сервісів для широкого загалу можна очікувати не скоро.

На відміну від фізичної особи, юридичні особи мають більш прогнозований підхід до всіх процесів, зокрема й до збереження та управління інформацією. З огляду на специфіку конкретного замовника з бізнесу чи

державної структури комерційні хмарні сервіси можуть вдатись до прогнозованого резервування. Наприклад, серед сучасних хмарних сервісів можна взяти . хмарні CRM системи, зокрема Oracle, SAP AG, Salesforce.com, тощо, то можна спробувати знайти покращене застосування.

Зокрема, якщо взяти за приклад Salesforce, то цей сервіс пропонує як збереження інформації у хмарі для компаній, так і дозволяє виконувати обчислення в хмарі. В свою чергу якщо компанія-замовник має більше ніж один сервер в рамках цієї хмарної системи, то на фізично можливим є варіант рознесення інформації на географічно різні вузли зберігання інформації, що надає компанія постачальник послуг. В такому разі можна виокремити критичну для побудованої бізнесом системи, або всередині системи інформацію, та продублювати її резервування для кожної матриці формування резервних блоків даних. В такому разі забезпечиться відновлення цих критичних даних при втраті будь якого кластеру.

ВИСНОВКИ ДО РОЗДІЛУ 4

В даному розділі було проведено симуляцію системи із розробленим способом відновлення даних при їх віддаленому розподіленому зберіганні. Поведінкова модель пройшла низку тестувань та виправдала очікування.

Система виконує заявлені функції, а саме аналіз блоків даних, що надходять та швидке їх відновлення, окрім того система сигналізує, якщо під час її роботи сталась будь-яка помилка. З огляду на наявні результати, та на те, що система з даним способом має можливість відновлювати і ті блоки даних, які затримуються і по мережі, було запропоновано використовувати цю систему в системах CRM.

ВИСНОВКИ

В розділі 1 даної роботи було досліджено проблематику надійності збереження інформації при віддаленому її зберіганні. На основі проведених досліджень можна стверджувати, що сучасні технології передачі інформації забезпечують швидкий доступ до віддаленого сховища даних, що робить використання розподілених систем актуальним. Окрім того, було зазначено, що основною прогалиною в рамках забезпечення надійності є рівень відновлення тих кластерів, які не мають зв’язку, або мають затримки зі зв’язком.

В розділі 2 було досліджено наявні методи резервування та відновлення даних при віддаленому та розподіленому зберіганні. В розділі було проведено порівняльний аналіз двох методів, а саме: метод відновлення всіх блоків даних на одному сховищі та трьох блоків даних на різних сховищах та метод відновлення даних з довільної заданої кількості носіїв, які найбільше підходять для вирішення поставленої задачі, та обрано оптимальний за критеріями якості відновлення інформації та надлишковості.

Розділ 3 ніс в собі інформацію, щодо структурного та функціонально складу системи із запропонованим способом. Опис всіх елементів моделі системи. Цей розділ містив детальні відомості про етапи функціонування системи, опис входів та виходів. Також наведені результати тестування та відлагодження окремих функціональних частин системи із запропонованим способом, та приклади їх роботи.

В розділі 4 було подано результати проведення поведінкових симуляцій системи із запропонованим способом для перевірки її на відповідність вимогам поставленим на початку роботи. Цей розділ несе в собі інформацію щодо результатів роботи змодельованої системи відновлення даних при їх віддаленому розподіленому зберіганні. Модель пройшла низку тестувань та можна заявити, що використання її як методу забезпечення резервування даних на рівні вузлів

даних є оптимальним рішенням. Також в розділі було наведено рекомендації щодо використання даної системи на практиці.

В результаті можна зробити висновок, що розроблений спосіб відновлення даних при їх віддаленому розподіленому зберіганні, задовольняє сучасні потреби та вимоги, щодо забезпечення резервування та відновлення даних на рівні кластерів даних, тобто вузлів інформації, які в наш час можуть бути від’єднанні чи зруйновані внаслідок локальних катаклізмів.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Иванов Д. Г. Организация резервирования в системах распределенного хранения данных // Вісник Національного технічного університету України ”КПІ” Інформатика, управління та обчислювальна техніка, – Київ: ВЕК+ – 2012 – № 56. с.160-164.
2. Kahate, A. Cryptography and Network Security. New Delhi : Tata McGraw-Hill Publishing Company, 2008 2.
3. Juare, S. Survey on Data Security and Integrity Issues in Cloud Computing. Nagpur (MS) : IOSR Journal of Computer Engineering, 2016.
4. XIAO and XIAO: Security and Privacy in Cloud Computing, IEEE Communications Surveys & tutorials, January 2013.
5. Plank J. S., Thomasson M.G., On Practical Use of LDPC Erasure Codes for Distribute Storage Application: Technical Report UT-CS-03-510. – Department of Computer Science, University of Tennessee. – 2004.
6. Plank J. S., Thomasson M.G., On Practical Use of LDPC Erasure Codes for Distribute Storage Application: Technical Report UT-CS-03-510.- Department of Computer Science, University of Tennessee. - 2004.
7. Луцкий Г.М., Иванов Д.Г. Метод восстановления данных на основе скошенных матриц в системах распределенного хранения // Известия высших учебных заведений. Проблемы полиграфии и издательского дела. Информационные технологии. - М.: УПИПК МГУП им. И. Федорова. - 2013.- № 2. - С.47-52 3.
8. Heuert U., Ivanov D. Paladin: Secure and redundant cloud storage // Herald of the Merseburg University of Applied Sciences. - Merseburg: Elbe Drucketei Wittenberg GmbH.- 2011.-№ 8.-S.220-228.
9. Blaum M., James Lee Hafner, Steven Hetzler (2013) Partial MDS Codes and Their Application to RAID Type of Architectures, IEEE Trans. Inf. Theory. vol.59, no.7, P. 4510-4519.

10. Im S. Flash-aware RAID techniques for dependable and high-performance flash memory SSD/ S. Im, D. Shin // IEEE Trans. Comput. -2011. - Vol.C-60, - No.1, - P. 80-92.
11. Vaushampayan V.A. Reability of Erasure Codes Storage Systems: A Combinatorial-Geometric Approach / V.A. Vaushampayan, A. Campello // IEEE Transaction on Information Theory. - 2015.- Vol. 61.- No.11.- P. 5795- 5809.
12. Li M. C-codes: Cyclic lower-density MDS array codes constructed using starters for RAID-6 / M. Li, J. Shu // IBM New York USA. Res. Report RC25218.- 2011.- 273 P.
13. Calder B. Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency / B. Calder // ACM SOSP. - 2011.- № 2. - P.62-70.
14. Li J. A Generic Transformation to Enable Optimal Repair in MDS Codes for Distribution Storage Systems. / J. Li, X. Tang, C. Tian // IEEE Transaction on Information Theory.- 2019.- Vol. 65.- No. 9.- P. 6257-6267.
15. Березюк Н.Т. Кодирование информации. Двоичные коды. Харків: Вища школа, 1978. – 252 с.
16. Johnny M. A Multi-Level Encoding and Decoding Strategy for Binary Erasure Channel / M. Johnny, M.R. Aref // IEEE Transaction on Information Theory.- 2019.- Vol. 65.- No. 7.- P. 4143-4151.
17. Sohn J.Y. Capacity of Clustered Distributed Storage / J.Y. Sohn, B. Choi, S.W. Yoon, J. Moon // IEEE Transaction on Information Theory. - 2019.- Vol. 65.- No. 1.- P. 81-107.
18. Chen H.C.H. NCCloud: A network coding based storage systems in cloud-of cloud / H.C.C. Chen, Y. Hu, P.P.C. Li, Y. Tang // IEEE Transaction on Computers. - 2014.- Vol.63.-No.1.- P.31-44.
19. Stones R. J. K-Plex 2-Erasure Codes and Blackburn Partial Latin Squares / R. J. Stones // IEEE Transactions on Information Theory. – 2020.-vol. 66.- № 6.- P.3704-3713.
20. Challagidad P. Local and Remote Recovery of Cloud Services Using Backward Atomic Backup Recovery Technique for High Availability in Strongly Consistent Cloud Service: Recovery of Cloud Service for High Availability / Praveen Challagidad.

// International Journal of Advanced Pervasive and Ubiquitous Computing. – 2019. – №11. – С. 16–33.

21. Иванов Д.Г. Метод восстановления данных на основе скошенных матриц в системах распределенного хранения / Иванов Д.Г. Луцкий Г.М. // Известия высших учебных заведений. Проблемы полиграфии и издательского дела. Информационные технологии. – М.:УПИПК МГУП им. И. Федорова. – 2013. – № 2. – С.47-52.
22. Марковський О.П. Організація резервування та відновлення даних при їх віддаленому зберіганні / Марковський О.П., Іванов Д.Г., Ванчугов Б.Ю. // Вісник Національного технічного університету України ”КПІ” Інформатика, управління та обчислювальна техніка, – Київ: ВЕК+ – 2013. – № 59. – С. 50-55.
23. Марковський О.П. Організація резервування та відновлення даних при їх віддаленому зберіганні / Марковський О.П., Іванов Д.Г., Ванчугов Б.Ю. // Вісник Національного технічного університету України ”КПІ” Інформатика, управління та обчислювальна техніка, – Київ: ВЕК+ – 2013. – № 59. – С. 50-55.
24. Марковський О. П. Метод резервування та прискореного відновлення даних в системах їх віддаленого зберігання / О. П. Марковський, М. М. Великий // Вісник Національного технічного університету України "КПІ". Інформатика, управління та обчислювальна техніка. - 2015. - Вип. 63. - С. 65-71.
25. Коляда К.В., Романкевич В.О., Орлова М.М., Марковський О.П. Метод відновлення даних при їх розподіленому зберіганні на віддалених сховищах // Комп‘ютерно-інтегровані технології: освіта, наука, виробництво. – ғ 40. – 2020. – С.44-50.