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

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)

«___» _____20___ р.

# Дипломний проект

**на здобуття ступеня бакалавра**

по спеціальності **123 «Комп'ютерна інженерія»**

на тему: HDL-модель пристрою відновлення даних

Виконав (-ла):

студент (-ка) IV курсу, групи КВ73
(шифр групи)

Литвиненко Дмитро Олегович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник ст. викладач каф. СПіСКС Коляда К.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПіСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали ) (підпис)

Рецензент доцент кафедри ОТ КПІ ім. Ігоря Сікорського,
к.т.н. Олександр МАРКОВСЬКИЙ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2021 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 – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___» _____20___ р.

**ЗАВДАННЯ**
**на дипломний проект студента**

Литвиненка Дмитра Олеговича
(прізвище, ім'я, по батькові)

1. Тема проекту HDL-модель пристрою відновлення даних

керівник проекту Коляда Костянтин Вячеславович, ст. викладач каф. СПіКС,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 25 травня 2021 р. № N1331-С

2. Термін подання студентом проекту _____

3. Вихідні дані до проекту застосування – відновлення даних на хмарних сховищах, пристрій – модель відновлення на мові HDL, алгоритм – відновлення всіх блоків одного та трьох блоків різних сховищ, мова розробки – HDL

4. Зміст пояснювальної записки опис поняття «відновлення даних», обґрунтування необхідності відновлення даних, алгоритм побудови матриці, за допомогою якої відбувається відновлення, опис алгоритму відновлення даних, опис пристрою

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) схема електрична структурна; схема електрична функціональна; блок-схема алгоритму формування матриці

відновлення; блок схема алгоритму моделі відновлення даних; слайди презентації

6. Консультанти розділів проекту*

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Нормоконтроль | Клятченко Я.М., к. т. н., доцент каф. СП і СКС | | |
| | | | |
| | | | |

7. Дата видачі завдання _____

Календарний план

| № з/п | Назва етапів виконання дипломного проекту | Термін виконання етапів проекту | Примітка |
|---|---|---|---|
| 1. | Видача та пояснення завдання до дипломного проекту | 20.04.2021 | |
| 2. | Перегляд теоретичних матеріалів | 26.04.2021 | |
| 3. | Теоретичне представлення зберігання даних на хмарних сховищах | 14.05.2021 | |
| 4. | Вивчення алгоритмів відновлення даних | 25.05.2021 | |
| 5. | Побудова структурної схеми моделі відновлення | 27.05.2021 | |
| 6. | Побудова алгоритму матриці відновлення | 30.05.2021 | |
| 7. | Вивчення мови HDL | 01.06.2021 | |
| 8. | Проектування моделі відновлення на мові HDL | 05.06.2021 | |
| 9. | Розробка функціональної схеми моделі відновлення | 10.06.2021 | |
| 10. | Підготовка дипломних матеріалів | 10.06.2021 | |
| 11. | Огляд дипломних матеріалів на кафедрі | 10.06.2021 | |

Студент _____ Дмитро ЛИТВИНЕНКО
(підпис) (Ім'я та ПРІЗВИЩЕ)

Керівник проекту _____ Констянтин КОЛЯДА
(підпис) (Ім'я та ПРІЗВИЩЕ)

.

* Консультантом не може бути зазначено керівника дипломного проєкту.

# Анотація

Бакалаврський дипломний проект присвячений HDL моделі пристрою відновлення даних. Розглянуто актуальність відновлення даних в сучасному світі. Досліджено основні алгоритми відновлення даних, метод відновлення на базі, якого побудована модель швидкий та діючий. Розроблено схему алгоритм моделі відновлення, також побудови матриці відновлення, за допомогою якої здійснюється відновлення даних.

В дипломному проекті обґрунтовано вибір мови HDL для реалізації пристрою. Також, представлена модель збереження даних на хмарних сховищах та актуальність хмарних сховищ в даний час.

# Abstract

The bachelor's thesis project is dedicated to the HDL model of a data recovery device. The relevance of data recovery in the modern world is considered. The main algorithms of data recovery, the method of recovery on the basis of which the model is fast and effective are investigated. The scheme algorithm of the recovery model is developed, also construction of a recovery matrix by means of which data recovery is carried out.

The diploma project substantiates the choice of HDL language for the implementation of the device. Also, the model of data storage in cloud storage and the relevance of cloud storage at present is presented.

| Поз. | Формат | ПОЗНАЧЕННЯ | НАЙМЕНУВАННЯ | Кількісь аркушів | № прим. | Примітки |
|---|---|---|---|---|---|---|
| | А4 | ІАЛЦ.467200.002 ТЗ | HDL модель пристрою | 4 | | |
| | | | відновлення даних | | | |
| | | | Технічне завдання | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | А4 | ІАЛЦ.467200.003 ТП | HDL модель пристрою | 4 | | |
| | | | відновлення даних | | | |
| | | | Відомість технічного | | | |
| | | | проекту | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | А4 | ІАЛЦ.467200.004 ПЗ | HDL модель пристрою | 51 | | |
| | | | відновлення даних | | | |
| | | | Пояснювальна записка | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | А1 | ІАЛЦ.467200.005 Е1 | HDL модель пристрою | 1 | | |
| | | | відновлення даних | | | |
| | | | Схема структурна | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | ІАЛЦ.467200.001 ОА | |
| Змін | Арк. | № докум. | Підпис | Дата | | |
| Розробив | | Литвиненко Д.О | | | HDL-модель пристрою відновлення даних **Опис альбому** | Літ. / Аркуш 1 / Аркушів 2 |
| Перевірив | | Коляда К.В. | | | | |
| Консульт. | | | | | | КПІ ім. Ігоря Сікорського, ФПМ КВ-73 |
| Н. контроль | | Клятченко Я.М. | | | | |
| Зав. каф. | | Романкевич В. | | | | |

| Поз. | Формат | ПОЗНАЧЕННЯ | НАЙМЕНУВАННЯ | Кількість аркушів | № прим | Примітки |
|---|---|---|---|---|---|---|
| | A1 | ІАЛЦ.467200.006 Д2 | HDL модель пристрою | 1 | | |
| | | | відновлення даних | | | |
| | | | Схема алгоритму | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | A1 | ІАЛЦ.467200.007.Д3 | HDL модель пристрою | 1 | | |
| | | | відновлення даних | | | |
| | | | Схема алгоритму | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | A1 | ІАЛЦ.467200.008 Е2 | HDL модель пристрою | 1 | | |
| | | | відновлення даних | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | Диск CD-ROM | Текст пояснювальної | 1 | | |
| | | | записки | | | |
| | | | Графічний матеріал | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

**ЗМІСТ**

## 1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «HDL-модель пристрою відновлення даних».

Галузь застосування: відновлення даних на хмарних сховищах

## 2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування на здобуття першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

## 3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Побудова моделі відновлення даних, які зберігаються користувачем на хмарних сховищ. Описати алгоритм, за яким відбувається відновлення блоків даних. Розробити модель пристрою відновлення на мові програмування HDL.

## 4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

## 5. ТЕХНІЧНІ ВИМОГИ

### 5.1 Вимоги до моделі, що розробляється

- Підтримка Windows;

- Програмне забезпечення Active VHDL;
- 512 Мб оперативної пам’яті;
- 512 Мб фізичної пам’яті;

5.2 Вимоги до програмного забезпечення користувача

- ОС Linux, Windows, IOS, Android;
- Доступ до мережі Інтернет;
- 30 Мб фізичної пам’яті;
- 512 Мб оперативної пам’яті;

# 6. ЕТАПИ РОЗРОБКИ

| № з/п | Назва етапів виконання дипломного проекту | Термін виконання етапів проекту | Примітка |
|---|---|---|---|
| 1. | Видача та пояснення завдання до дипломного проекту | 20.04.2021 | |
| 2. | Перегляд теоретичних матеріалів | 26.04.2021 | |
| 3. | Теоретичне представлення зберігання даних на хмарних сховищах | 14.05.2021 | |
| 4. | Вивчення алгоритмів відновлення даних | 25.05.2021 | |
| 5. | Побудова структурної схеми моделі відновлення | 27.05.2021 | |
| 6. | Побудова алгоритму матриці відновлення | 30.05.2021 | |
| 7. | Вивчення мови HDL | 01.06.2021 | |
| 8. | Проектування моделі відновлення на мові HDL | 05.06.2021 | |
| 9. | Розробка функціональної схеми моделі відновлення | 10.06.2021 | |
| 10. | Підготовка дипломних матеріалів | 10.06.2021 | |
| 11. | Огляд дипломних матеріалів на кафедрі | 10.06.2021 | |

| Поз. | Формат | ПОЗНАЧЕННЯ | НАЙМЕНУВАННЯ | Кількість аркушів | № прим. | Примітки |
|---|---|---|---|---|---|---|
| | А4 | ІАЛЦ.467200.004 ПЗ | HDL-модель пристрою | 51 | | |
| | | | відновлення даних | | | |
| | | | Пояснювальна записка | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | А1 | ІАЛЦ.467200.005 Е1 | HDL-модель пристрою | 1 | | |
| | | | відновлення даних | | | |
| | | | Структурна схема | | | |
| | | | пристрою | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | А1 | ІАЛЦ.467200.006 Д2 | HDL-модель пристрою | 1 | | |
| | | | відновлення даних | | | |
| | | | Схема алгоритму | | | |
| | | | побудови матриці | | | |
| | | | відновлення | | | |
| | | | | | | |
| | | | | | | |
| | А1 | ІАЛЦ.467200.007 Д3 | HDL-модель пристрою | 1 | | |
| | | | відновлення даних | | | |
| | | | Схема алгоритму | | | |
| | | | моделі відновлення | | | |
| | | | | | | |

| Змін. | Арк. | № докум. | Підпис | Дата | ІАЛЦ.467200.003 ТП | | |
|---|---|---|---|---|---|---|---|
| Розробив | | Литвиненко Д.О. | | | HDL-модель пристрою відновлення даних<br>**Відомість технічного проєкту** | Літ. | Аркуш 1 / Аркушів 2 |
| Перевірив | | Коляда К.В. | | | | | |
| Консульт. | | | | | | КПІ ім. Ігоря Сікорського, ФПМ КВ-73 | |
| Н. контроль | | Клятченко Я.М. | | | | | |
| Зав. каф. | | РоманкевичВ.О. | | | | | |

| Поз. | Формат | ПОЗНАЧЕННЯ | НАЙМЕНУВАННЯ | Кількість аркушів | № прим. | Примітки |
|---|---|---|---|---|---|---|
| | A1 | ІАЛЦ.467200.008 Е2 | HDL-модель пристрою | 1 | | |
| | | | відновлення даних | | | |
| | | | Схема електрична | | | |
| | | | функціональна | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | A1 | Диск CD-ROM | Текст пояснювальної | 1 | | |
| | | | записки | | | |
| | | | Графічний матеріал | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | 1 | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

| Змі | Арк. | № докум. | Підпис | Дата | ІАЛЦ.467200.003 ТП | Арк. 2 |
|---|---|---|---|---|---|---|

# Пояснювальна записка
## до дипломного проекту

на тему: HDL-модель пристрою відновлення даних

Київ – 2021

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

| | |
|---|---|
| ABEL | – Advanced Boolean Expression Language |
| API | – application program interface |
| CD | – Compact Disc |
| CPLD | – Complex Programmable Logic Device |
| DDoS | – Denial of Service(відмова в обслуговувані) |
| DVD | – Digital Versatile Disc |
| FPGA | – field-programmable gate array |
| GUI | – graphic user interface |
| HDL | –Hardware Description Language |
| IaaS | – Infrastructure as a Service |
| Networks | – мережі |
| PaaS | – Platform as a Service |
| SQL | – structured query language |

# ВСТУП

На сьогоднішній день існує маса всіляких хмарних систем зберігання. У кожного свої завдання. Одні відповідають за зберігання листування у месенджерах, інші - для перенесення фотографій або резервного копіювання листів з електронної пошти. Є й універсальні, завдяки яким можна завантажувати файли будь-якого типу і з будь-яким розширенням

Залежно від масштабів обладнання, необхідне для зберігання даних, може займати як невелику кімнату, так і окремі будівлі, по площі зіставні з повноцінними ангарами для літаків. Такі місця називаються центрами обробки даних. Приклад такого центру обробки можна спостерігати на рисунку 1 – «Центр обробки даних».


Рис. 1 – «Центр обробки даних»

Для компаній, що надає дисковий простір в якості хмарного, важливо мати не тільки достатню потужність, а й подбати про декілька додаткових факторах, таких як, надмірне резервування.

Центру обробки даних не може складатися з серверів, які розраховані суто на конкретну клієнтську базу. Говорячи простіше: якщо хмарою користується 1000 осіб за тарифом 10 ГБ в хмарі, то компанії потрібно подбати про наявність не 10 ТБ ємності сховища, а про значно більшу.

Справа в тому, що серверам необхідна періодична профілактика. Адже, по суті, це залізо, яке покривається шаром пилу, перегрівається, виходить з ладу. Але клієнт повинен бути впевнений в тому, що отримає доступ до своїх файлів в будь-який час. Приклад надлишкових серверів можна бачити на рисунку 2 - «Зберігання даних на надлишкових серверах»

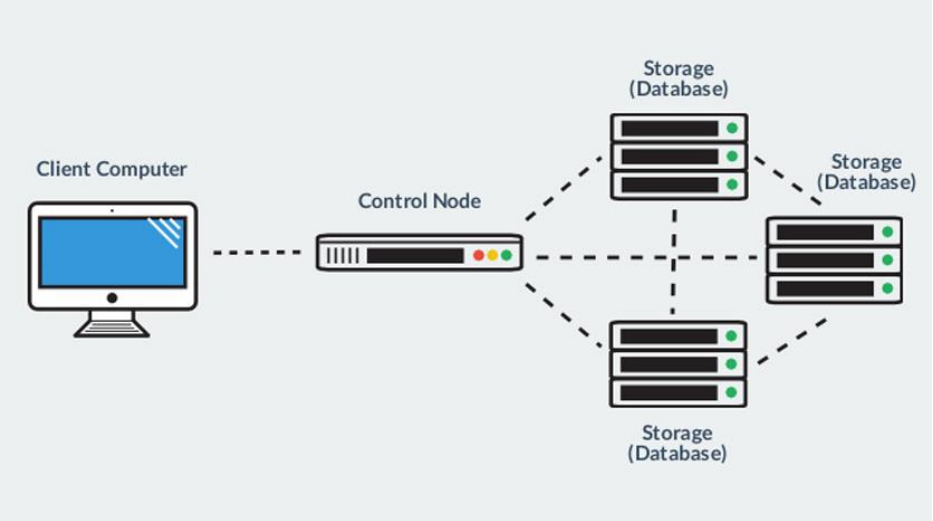


Рис. 2 - «Зберігання даних на надлишкових серверах»

Для цього застосовуються надлишкові сервера. На їх зберігаються, як би незвично це не звучало, копії ваших даних. А оригінали - у вас на компі або ж телефоні. Під час профілактики сервера відключаються в подібний черговості, щоб, як мінімум, 1 знімок даних щоразу залишалася доступною.

Розглянемо переваги віддаленого зберігання даних. Управління інформацією вважається централізованим. Сховище в хмарі робить великі здібності щодо структурування інформації. Для цього Для вас в тому числі і перестає бути важливим здобувати якісь фізіологічні приладу. Для вас досить володіти профіль однієї з фірм, яка дає пропозицію заощадження даних виділено.

Для фірм, це досить важлива річ, наприклад як заощадження даних на жорсткому диску комп'ютера загрожує їх втратою, зносом оснащення, крадіжкою або ж псуванням. Крізь постійне автоматичне резервне копіювання даних в хмара вся інформація фірми міцно захищена.

Основна маса хмарних сховищ дають можливість виявити доступ до файлів, спільної роботи над ними іншим користувачам. І всі ці маніпуляції

можливо проводити в режимі реального часу. Це збільшує продуктивність роботи фірми, дозволяє більше доцільно здійснити пролетар процес.

Шифрування - навіть якщо дані вкрадені, без ключа шифрування, отримати цінну інформацію буде неможливо.

Аутентифікація користувачів - від очевидних «ім'я користувача і пароль» в більше важких способів, завдяки яким ніхто не може отримати несанкціонований доступ до відомостей.

Призначена для користувача авторизація - процес надання окремим особам доступу до різних типів даних в залежності від їх ролі і потреб.

Поговоримо про децентралізоване та централізоване збереження даних При централізованому збереженні даних є величезний ризик втратити дані. Якраз децентралізований характер сховища цього типу в центрі обробки даних готує його більше нешкідливим простором для заощадження ваших ділових або ж власних даних, ніж локальне сховище. Фізіологічні сервери, що знаходяться у вашому кабінеті, уразливі для всіх видів загроз, починаючи від стрибків напруги, шкідливих програм, стихійних лих, тілесних збоїв і старіння оснащення / програмного забезпечення.

Основна праця в дипломному проекті присвячена відновлюванню даних. В дипломному проекті розроблено модель відновлення даних за рахунок резервних блоків, які передаються разом із даним для їх відновлення в разі певних обставин.

Відновлення втрачених під час зберігання даних здійснюється за рахунок резервних даних, які також зберігаються на віддалених розподілених системах. В роботі створено модель відновлення даних, на базі методу відновлювання всіх блоків одного та трьох блоків різних сховищ. Відновлення втрачених, або затриманих понад критичний проміжок часу даних при передачі в мережах також здійснюється шляхом передачі резервної інформації разом з основною.

Основна робота в дипломному проекті полягає в представлені моделі відновлювання даних за допомогою методу відновлювання всіх блоків одного та трьох блоків різних сховищ. Для цього складається модель збереження

інформації на віддалених сховищах та на основі цієї моделі будуть проводитися розрахунки та відновлення втрачених даних. Також дослідження полягає в роботі з матрицею резервних блоків пакетів та блоків даних користувача, які формуються разом з основними для відновлення втрачених або затриманих понад критичний проміжок часу.

# РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЄКТУ

## 1.1. Загальний опис проблеми та обґрунтування актуальності теми

В даний час існує безліч вірусів, які щодня атакують комп’ютери. За допомогою Інтернету, на сьогоднішній день можна дізнатися майже будь яку інформацію. За допомогою пошукових систем доступ до інформації став дуже легко доступним.

Але, на жаль, в цьому самому Інтернеті, який надає стільки можливостей є і пастки. Такими пастками є вірусі та шкідливі програмні забезпечення, через які дані можуть втратитися чи ушкодитися. На даний момент дуже багато людей втратили свої дані через такі віруси. За перші шість місяців 2003 року в одних тільки США було зафіксовано понад 76,4 тис. Вірусних атак, причому багато фахівців вперто вважають цю цифру заниженою[1].

Нерідко цілісність даних порушується в результаті саботажу. Певний алгоритм злочинних дій хакера приводить його до поставленої цілі – заволодіння не своїми даними. Це можуть бути шкідницька діяльність сукупності людей, які мають навички або можливість приносити злочинну діяльність.

Також, існують людські фактори. До порушення цілісності може призвести помилки в програмному забезпеченні. Збій може виникнути через неправильне налаштування додатків, невдалого оновлення програмних продуктів, некоректного форматування при установці операційної системи або помилок при її оновленні.

Всі дані піддаються ризику, будь то помилка користувача, збій системи або обладнання, шкідливе ПЗ, стихійні лиха. Вбудоване резервування хмарної архітектури та протоколи захисту даних мають велике значення для забезпечення максимальної безпеки вашої інформації.

Хмарне сховище — структура даних, де інформація зберігається в логічні об’єкти, а фізичне зберігання охоплюється кількома «хмарними» сховищами. Ці сховища належать компаніями, які надають серверні послуги Ці власники хмарних сховищ даних відповідають за схов наявної інформації й доступ до неї, та за роботу фізичного середовища, у якої користувачі зберігають дані, звісно, постачальники сховищ даних отримують матеріальний прибуток за продаж своїх послу. Вони можуть бути доступні через API на смартфоні або через інший GUI на майже на будь – якому цифровому пристрої.

1.1.1. Найчастіші технічні помилки

Закономірні поломки: накопичувач не містить помітних тілесних ушкоджень і розпізнається в системі. Проблема з'являється при спробі доступу або ж запису / зчитування даних. З'являються дані поломки в самих різних випадках. 1 з найпопулярніших підстав - невірне витяг приладу з компа. У разі закономірних поломок відновити дані можливо з підтримкою програм для відновлення даних.

Механічні ушкодження: диск перестав коректну роботу в результаті будь-якого фізіологічного впливу (падіння, попадання вологи, вигину, стиску і т.д.). Першопричина поломки, частіше за все, в несправності плати або ж руйнуванні контактів і компонент. Відновити дані можливо, в разі якщо виправити несправність: поміняти несправний складову або ж відновити порушений контакт. Ще можливо вважати дані саме з чіпа пам'яті, застосовуючи особливе оснащення.

Електричні пошкодження: першопричина електро-ушкоджень полягає в статичному ударі або ж в завданню з харчуванням. В результаті можуть згоріти стабілізатори харчування, діоди, контролери. Відновлення даних виконується, як і в попередньому випадку: підміною компонент або ж читанням з чіпів пам'яті прямо.

1.1.2. Хмара і безпека.

Без сумніву, це 1 з вагоміших завдань, при цьому її розмах весь час збільшується. Кінцевий щорічний доповідь Arbor Networks по масової інфраструктурі захищеності показує на це досить виразно. Так, потужність

DDoS-атак в 2013 році збільшилася до 309 Гбіт / с, в 2014 році дана цифра перевищила планку в 400 Гбіт / с. Тому що хмарні обчислення робляться все більше відомими, вони вважається метою безлічі шкідливих атак[2]. Ні 1 інфраструктура не вважається до кінця захищеної і вимагає суворого контролю з підтримкою комплекту дипломат. Heartbleed вважається чудовим трапилося такого, як гігантська похмура організація потрапила під удар стандартизованої структури захищеності.

Можливі витрати даних. Міграція користувачів в хмара і майбутній доступ до похмурим додатків дозволяє кінцевим користувачам працювати виділено. Втім, власне що трапляється, коли користувач починає завантажувати файли в хмару? Так як деяка інформація має можливість коштувати досить дорого. Недавній доповідь альянсу HITRUST показує на велику кількість витоків мед даних, негайно ж знижує оптимістичність порівняно хмари.

Суцільне чисельність витків: 495

Всього було викрадено записів: 21120000Загальна ціна вкрадених даних: $ 4100000000

Середній розмір витоку 42659 записів

Середня ціна: $ 8270000

Невигадливий в роботі хмари. Як правило, жодне хмара не зважає на 100% застрахована від природних лих[4] або ж інших форс-мажорних небезпек. Наприклад, потужний буря в червні 2012 року обірвав роботу великого датацентру, що є власністю Amazon, в якому, звичайно ж, були поміщені Amazon Web Services (AWS). Всі бізнеси, які сприймали AWS в даному датацентрі, припинили. Популярні похмурі сервіси, такі як Instagram, Netflix і Pinterest були недосяжні в напрямок більше 6:00. Взагалі, з 2007 року 13 найбільших похмурих постачальників переривали свою діяльність у спільній сумі на термін в 568 годин. Даний невигадливий вийшов клієнтам в $ 72 млн.

Технічні поломки призводить до втрати даних як на підприємствах наприклад і на разі 1-го юзера[4]. У провідному ці випадки з'являються ненавмисно, або ж в рідкісних випадках - крізь кримінальні дії лиходія. Виділяють кілька підстав виходу оснащення з ладу, наприклад природні

моменти. Комп'ютерні оснащення можуть отримати травми від повені, урагану, блискавки, пожежі і т.п. Оснащення прийде в неробочий стан і призведе до втрати даних в результаті аварії, будь вона техногенної або ж екологічної. Стрибок напруги, потужне збільшення температури навколишнього середовища веде до займання компонент персональних комп'ютерів або ж серверів, пошкодження жорсткого диска.

Оснащення має можливість замерзнути несправним з вини злодіїв. Це можуть бути як звичайні злочинці, наприклад і навмисне найняті зацікавленими людьми хакери. Досить велика кількість людей можуть завдати шкоди комп'ютерному обладнанню, це можуть бути в тому числі і народ, які зобов'язані для вас лагодити оснащення, а насправді - стануть завдавати шкоди. Крім такого, поломка оснащення має можливість з'явитися через поломки 1-го речовини в схемі або ж сукупності складових.

Дипломний проект приурочений до моделі відновлення даних втрачених або ж затриманих більше небезпечний зазор часу, що зберігаються на віддалених сховищах. Модель відновлення даних працює на базі методу відновлення всіх блоків 1-го і 3-х блоків всіляких сховищ. Модель описана на теоретичному рівні, і на мові програмування HDL. Розрахунки ведуться з підтримкою матриці формування запасних блоків, які складаються спільно з провідними блоками інформації, для її відновлення.

1.1.3. Шифрування

З метою захищеності і запобігання несанкціонованого доступу, всі завантажені в хмара дані шифруються з підтримкою важкого методу кодування. Отримати доступ до цих відомостей можливо лише тільки з підтримкою ключа шифрування, які вважаються тільки у юзера.

Само по собі похмурий заощадження ділиться на 3 категорії: інфраструктура як послуга (IaaS) - обстановка, в якій ці великі гравці як Amazon і Гугл надаю власні апаратні потужності в оренду іншим компаніям; перон як обслуговування (PaaS) - розміри місця в інтернеті, в яких творці

роблять програми для всіляких категорій користувачів; ПО як обслуговування (SaaS) - коли користувачі використовують програмне обеспечивание для доступу до хмари крізь онлайн.

Нарешті, ви прийняли рішення навантажити ще одну порцію фоток з відпустки в похмуре сховище. Розкривши мобільний замовник, обираєте знімки для синхронізації і обираєте «Завантажити».

З телефону крізь онлайн надходить запит на керуючий концентратор похмурого сервера. Його ще називають «розподільним сервером» або ж MasterMind-сервером. Він відповідає за обробку вашого запиту і відправку файлів буквально за адресою, тобто саме в папку, яка належить вам.[7]

Наглядну схему хмарного сховища користувача можна побачити на рисунку 3 – «Схема взаємодії користувача і хмарного сховища».

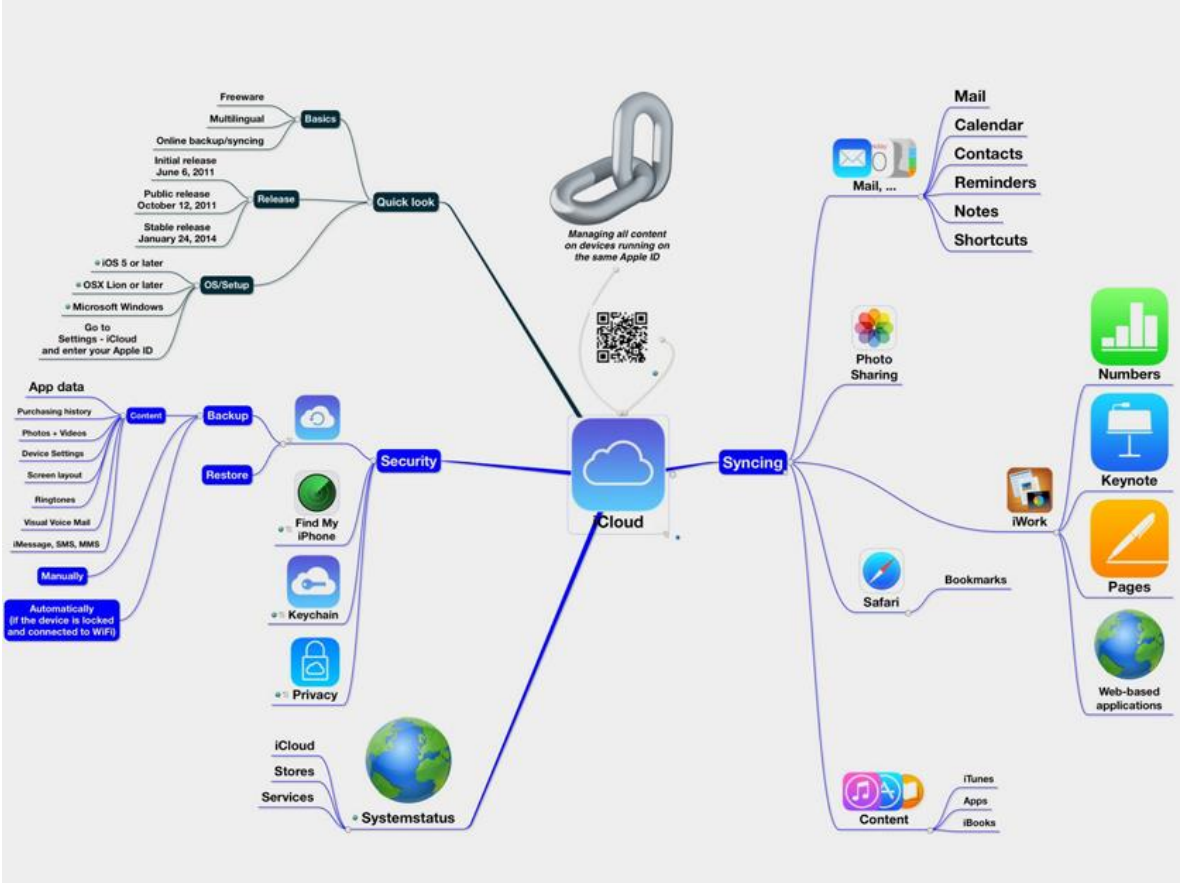


Рис. 3 – «Схема взаємодії користувача і хмарного сховища»

Нарешті, на сьогоднішній день досить велика кількість людей охороняють власні дані на похмурих сховищах. Ця ймовірність вважається

наслідком технічного прогресу і у неї є тривіальні видатні якості, такі як надійність. Справа в тому, власне що при передачі даних на віддалені носії спільно з даними передаються ще запасні дані, які надалі можуть знадобитися для відновлення втраченої інформації.

Інформація, як демонструє практика, має можливість губитися по ряду підстав, наприклад, хакерські або ж вірусні атаки. Більше тридцяти відсотків (34%) всіх атак на юридичні особи з впровадженням ВПО - це атаки троянів-шифрувальників. Ще не виключаються природні катаклізми, поломка в роботі технічних засобів в сховищах, де зберігається інформація користувачів.

В порівнянні з локальним зберіганням, віддалене має важливу перевагу надійності зберігання даних на географічно віддалених сховищах. Тобто втрата даних локалізується і дані можуть бути відновленні за рахунок резервних блоків даних, які в свою чергу, передаються разом с основними.

## 1.2. Аналіз основних алгоритмів та теоретичний опис

Було вивчено підставу втрат доступ до блоків даних користувачів. Витрати даних, викликані несправностями оснащення, старінням матеріалу носія, промахами в роботі програмного забезпеченням більш буквально описуються в математичному значенні струменями Пуассона.

### 1.2.1 Потік випадкових подій

Раз ресторанчик швидкого харчування буває в середньому 3 людини на хвилину. Втім це всього лише середній показник. Фактична сума має можливість виділятися.

Розподіл Пуассона можливо застосувати для аналізу ймовірності всіляких заходів порівняно такого, скільки покупців протікає крізь прохід. Це має можливість дозволити визначити можливість перерви в енергійності (коли 0 покупців приходять до проїзду), а ще можливість сплеску енергійності (коли 5 або ж більше покупців приходять до проїзду.). Дана інформація, в свою чергу, має можливість посприяти менеджеру спланувати ці дії з підбором персоналу і розкладом.

Напруженість струменя заходів показує, скільки в середньому трапляється цих заходів в одиницю часу. Але коли саме відбудеться будь-яка визначена захід треба кваліфікувати способами моделювання. Принципово, що, коли ми створимо, наприклад, за 200 годин 1000 заходів, їх чисельність стане точно також приблизно величині середньої інтенсивності виникнення заходів 1000/200 = 5 заходів на годину, власне що вважається статистичною величиною, що характеризує даний потік в цілому.

Напруженість струменя в конкретному значенні математичним очікуванням числа заходів в одиницю часу. Але можливо має можливість наприклад виявитися, власне що в годину буде помічений 4 дії, в іншій - 6, але в середньому виходить 5 заходів на годину, в наслідок цього однієї величини для характеристики струменя мало. 2 величиною, що характеризує як величезний розкид заходів порівняно математичного очікування, є, як і раніше, дисперсія. Саме якраз дана розмір визначає випадковість виникнення дії, кволу передбачуваність етапу його виникнення.

Для простого струменя можливість виникнення m заходів за час τ дорівнює:

$$P_m = \frac{(\lambda \cdot \tau)^m \cdot e^{-\lambda \tau}}{m!}$$

Потік подій - це послідовність однорідних подій, що входять одна за одною у випадкові проміжки часу. На осі часу ці події виглядають як показано на рис. 4.- «Графік осі часу»

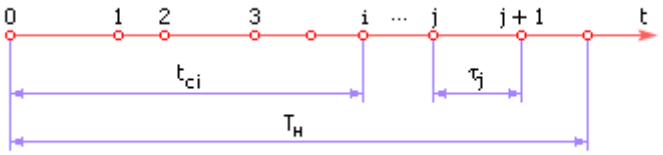


Рис. 4. – «Графік осі часу»

τj — интервал между событиями (случайная величина);

tci — момент совершения i-го события (отсчитывается от t = 0);

Тн — время наблюдения.

Трапилося струменя заходів можуть служити черговість факторів торкання злітної смуги літаками, прилітають в аеропорт.

Напруженість струменя λ - це середня кількість заходів в одиницю часу. Напруженість струменя можливо вирахувати дослідно за формулою: λ = N / Тн, де N - кількість заходів, що відбулися за час дослідження Тн.

У разі якщо перерва між заходами τj дорівнює константі або ж конкретний якийсь формулою у вигляді: tj = f (tj - 1), то потік називається детермінованим. Інакше потік називається випадковим.

Випадкові струменя трапляються:

нескладні: можливість одночасного виникнення 2-ух і більше заходів дорівнює нулю;стаціонарні: частота появи подій λ (t) = const (t);

без післядії: ймовірність появи випадкової події не залежить від моменту скоєння попередніх подій.

1.2.2. Пуассонівський потік

За еталон потоку в моделюванні прийнято брати пуассоновский потік.

Пуассонівський потік - це ординарний потік без післядії.

Як раніше було зазначено, ймовірність того, що за інтервал часу (t0, t0 + τ) відбудеться m подій, визначається із закону Пуассона рисунок 5:

$$a = \int_{t_0}^{t_0+\tau} \lambda(t) \cdot dt$$

Рис. 5 – «Закон Пуассона»

де a - параметр Пуассона.

У разі якщо λ (t) = const (t), то це стаціонарний потік Пуассона (самий). В даному випадку a = λ · t. У разі якщо λ = var (t), то це нестаціонарний потік Пуассона.[8]

Мал. 6 ілюструє залежність P0 від часу. Безперечно, власне що чим більше час дослідження, що можливість не виникнення ні 1-го дії менше. Крім такого, чим більше зміст λ, що крутіше йде графік, тобто швидше убуває можливість. Це відповідає що, власне що в разі якщо напруженість виникнення

заходів завелика, то можливість не виникнення дії швидко  зменшується зі періодом дослідження.

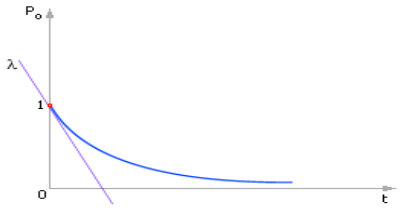


Рис. 6 – «Залежність P0 від часу»

З графіка на рис. 6 видно, власне що можливість виникнення але б 1-го дії жадає зі періодом до одиниці, тобто при відповідному нагляді дії це в обов'язковому порядку рано чи пізно станеться. Чим довше ми доглядаємо за заходом (чим більше t), постраждати від повені, урагану, блискавки, пожеж

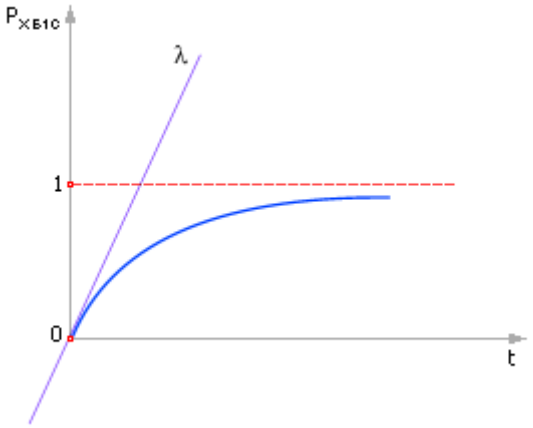


Рис. 7 – «P(t)»

За статистичними даними від Google інтенсивність, знайдена за допомогою потоку Пуассона становить близько $7 \cdot 10^{-6}$ год$^{-1}$.

1.2.3. Метод відновлювання даних

Втрата даних має можливість відбуватися за багатьма ознаками: електронні і падіння напруги, цунамі і що аналогічне. Дані можливо відновити майже всіма методами. У наданому дипломному плані стане застосовуватися спосіб відновлення даних всіх блоків 1-го і 3-х блоків всіляких сховищ. Даний спосіб працює на основі матриці формування запасних блоків, в якій по конкретним складової коштує 4-х розрядне текст. Ще за основу була взята

теоретична заощадження блоків даних виділено і формування запасних блоків даних, спільно з провідними при передачі для подальшого відновлення.

### 1.3. Постановка задачі

Задачею даного бакалаврського дипломного проекту є побудова моделі мета якої є відновлення втрачених даних при їх віддаленого зберіганні при їх втраті по ряду проблем, такі як злочинні дії, віруси, форс – мажори, людський фактор, природні катаклізми тощо. Описати та обґрунтувати формування моделі збереження інформації на віддалених сховищах. Провести та обґрунтувати розрахунки за допомогою матриці формування резервних блоків, які передаються для відновлення даних разом з основними.

Метод, який використовується для відновлення втрачених даних - метод відновлювання всіх блоків одного та трьох блоків різних сховищ. Саме на цьому методі треба буде теоретично побудувати модель відновлювання даних та описати її поведінку на мові програмування HDL.

### 1.4. Висновки

Відновлення втрачених даних при їх віддаленому зберіганні актуальна проблема сьогодення. Майже всі користуються «хмарними» сховищами, проте дані можуть бути втрачені через деякі причини - злочинні дії, віруси, форс – мажори, людський фактор, природні катаклізми тощо.

Віддалене зберігання, проте, дуже зручне через багато факторів, наприклад: можливість доступу до даних з будь-якого комп'ютера, що має вихід в інтернет , можливість організації спільної роботи з даними, висока ймовірність збереження даних навіть у разі апаратних збоїв та інші.

В цьому дипломному проекті побудовано модель відновлення даних при їх віддаленому зберіганні втрачених чи затриманих протягом критичний проміжок часу. Представлена та інформаційно обґрунтована модель збереження інформації на віддалених сховищах. Також, представлена і

обґрунтована матриця резервних блоків відновлення даних, які формуються при передачі разом з основними блоками для подальшого відновлення.

Були приведені реальні статистичні дані, щоб довести значність та актуальність проблеми втрати даних, та довести необхідність у моделях відновлення цих даних.

Описано принцип втрати даних, викликаний поломками обладнання, старінням матеріалу носія, помилками в роботі програмного забезпечення. Втрати через такі причини можна зрозуміти за допомогою принципу потоку подій Пуассона, який було описано в частині 1.2.

Було представлено теоретичну та формульну базу для розуміння принципу та необхідності відновлення даних. Теоретичне обґрунтування актуальності проблеми та широкості використання принципів відновлення втраченої чи затриманої понад критичний проміжок часу інформації.

# РОЗДІЛ 2. МЕТОД ВІДНОВЛЕННЯ, АЛГОРИТМ, БЛОК – СХЕМА

## 2.1. Поняття та роль відновлення даних у сучасному світі

Майже всі компанії бажають, щоб користувалися саме їх продуктом. Тому репутація компанії, яка надає можливість зберігати дані віддалено, дуже важлива. Компанії намагаються зробити так, щоб користувач був задоволений обслуговуванням. Звісно, обслуговування компанії складається з багатьох факторів: швидкість, доступність у використанні, ціни, максимальний об’єм хмарного сховища. Проте, одним з найважливіших критеріїв є недопустимість втрати даних.

Дані, які зберігаються виділено можуть являти собою досить велику значення для всіляких осіб, фірм і т.д. В наслідок цього в інтересах фірми влаштувати так, щоб ці взагалі не губилися. У тому числі і дані, які користувач видалив своїми руками можливо відновити.

Наприклад, одне з найвідоміших сховищ Гугл Drive. Гугл Диск вважається одним з найвідоміших похмурих сервісів, де можна зберігати до 15 гб інформації даром. Хмара від Гугл містить досить нескладну схему роботи при видаленні інформації. У разі якщо користувач прийняв рішення вислати конкретні дані, йому досить перебігти в відповідну папку, вибрати файл, натиснути по ньому правою кнопкою мишки і вибрати місце «Видалити»[5].

Згодом такого видалення файл стане переміщений в кошик, де розміщуються всі дані, які були коли-небудь видалені користувачем. Цей перелік можливостей містить ряд одноманітності зі нормальним рухом файлів в кошик з інтерфейсу Windows. Більше детально про видаленні і роботі кошика можливо визнати в замітці «Як відновити файли згодом видалення в« Кошик »і її очищення»[3].

Для відновлення інформації з кошика необхідно: перейти в розділ «Кошик», який можна виявити в лівій графі меню хмарного сховища, знайти віддалений файл, натиснути по ньому правою кнопкою миші і вибрати пункт «Відновити».

Відновлення даних - процедура вилучення інформації з запам'ятовує приладу в разі, коли вона не має можливість бути прочитана простим методикою.

Відновлення має можливість реалізуватися з будь-якого комп'ютерного носія, охоплюючи CD, DVD, строгі диски, флеш-пам'ять і т. Д Як правило, відновленню підлягають дані, що представляють конкретну значення.

В даний час є 2 провідних методу відновлення даних. Метод вибирається залежно від з'явилася поломки накопичувача. Програмно-апаратний спосіб використовується в тих випадках, коли програмний метод не дасть результату.[6]

Програмний метод - це відновлення даних без фізіологічного втручання в прилад накопичувача, а ще в функціонування прошивки і структуру модулів казенної інформації. Цей метод використовується у випадках, коли збережена функціональність самого накопичувача, але по що або ж інший основи доступ до відомостей, що зберігаються на ньому, загублений. Передумовою цього має можливість замерзнути форматування закономірних дисків, невдале зміна закономірною геометрії накопичувача, видалення інформації, вибіркове або ж абсолютне руйнування файлової системи, як інформації про структуру розміщення даних на накопичувачі. Найчастіше у випадках виходить відновити величезну частка даних, втім бачаться випадки, коли відновлення втрачених даних не можна (окремим випадком можна вважати перезапис даних). Для автоматизації процесу відновлення написано велику кількість програм, в що кількості і дармових. У разі форматування логічного диска або розділу, структура і атрибути даних не порушуються, але змінюється або інвентаризується (наводиться в початковий стан) інформація про розташування даних на даному накопичувачі.

При швидкому форматуванні оновлюється невелика частка файлової таблиці, частка казенних записів залишається, потрібно лише інтерпретувати її і прочитати дані в потрібному порядку.

Абсолютна форматування має можливість поєднувати всю файлову таблицю, в наслідок цього відновлення структури файлів і папок не всякий раз цілком ймовірно. Для відновлення даних без інформації про структуру можливо застосувати відновлення файлів по сигнатурам.

У разі якщо трапилося пошкодження файлової системи в результаті програмного збою або ж поломки носія, програми для відновлення даних можуть відновити частка інформації, знаходиться в залежності від розміру пошкоджень[9].

Модель відновлення описує схему структури даних в фізіологічному сховище; модель даних зміцнює більш значущі закономірні нюанси структури даних в основі даних. Тілесним сховищем має можливість бути локальний диск, знімний носій або ж сховище, дешеве крізь мережу.

Як правило застосовуються 2 абстрактні моделі заощадження: сховище осередків і сховище журналів. Сховище осередків має на увазі, власне що сховище вироблено з осередків схожого обсягу і власне що будь-який об'єкт поміщається буквально в 1 осередок.

Дана модель відображає фізіологічну компанію декількох носіїв інформації; початкова пам'ять пристрою організована як масив елементів пам'яті, а вторинне запам'ятовуючий прилад (наприклад, диск) організовано в секторах або ж блоках, зчитувальних і записуються як ціле єдине. узгодженість читання / запису і атомарність до або ж згодом - 2 досить бажаних якості всякий моделі заощадження і, зокрема, сховища осередків.

Сучасні бізнес-системи керують все більше гігантськими розмірами різнорідних даних. Дана різнорідність значить, власне що ціле сховище даних як правило не кращий розклад. Натомість цього нерідко ніж будь-якого іншого берегти різні типи даних в різних сховищах даних, будь-яка з яких націлене на конкретну робоче навантаження або ж шаблон застосування. Термін «сталість

поліглота» застосовується для опису висновків, в яких застосовується хитросплетіння технологій заощадження даних. В наслідок цього важливо брати головні моделі заощадження і їх компроміси.

Вибір сприятливого сховища даних для ваших домагань вважається головним дизайнерським висновком. Є практично сотки реалізацій на вибір між баз даних SQL і NoSQL. Сховища даних нерідко позначаються по що, як вони структурують дані і типам операцій, які вони підтримують. У цій статті описані кілька більш популярних моделей заощадження. Зверніть турбота, власне що певна розробка заощадження даних має можливість підтримувати деяка кількість моделей заощадження.

Наприклад, системи керування базами даних (СКБД) ще можуть підтримувати збереження ключів / значень або ж графів. Практично, є загальна тенденція до наприклад званої допомоги декількох моделей, коли 1 система баз даних підтримує кілька моделей. Але все ж здорово розуміти в всіляких моделях на найвищому рівні.

Не всі сховища даних в наданій категорії дають один і той же набір функцій. Основна маса сховищ даних дають активні здібності на стороні сервера для запиту і обробки даних. Часом ця функція вбудована в пристрій заощадження даних. В інших випадках здатності заощадження і обробки даних розбиті, і має можливість бути кілька варіантів обробки і аналізу. Сховища даних ще підтримують всілякі програмні інтерфейси та інтерфейси управління.

Інтернет-сховище даних - раз з найбільш безпечних і достовірних різновидів. Протоколи захищеності даних великого класу відстоюють вашу інформацію, а надлишкові системи гарантують, власне що в тому числі і в разі виходу з ладу конкретного оснащення у постачальника онлайн-сховища ваші дані стануть в захищеності. гігантські уразливості в онлайн-сховище даних пов'язані з клієнтським комп'ютером або ж пристроєм, який звертається до нього.

Для вас все ще потрібно йти по стопах найкращим практикам щодо комп'ютерів, які звертаються до відомостей у вашому онлайн-сховище.

Для дрібного бізнесу дані нерідко зберігаються на локальному жорсткому диску або ж суворих дисках в кабінеті. У безлічі випадках відсутня процедура резервного копіювання та вбудоване резервування. Ваші дані можуть бути втрачені через фізіологічного збою жорсткого диска, інфікування компа мікробом або ж промахи користувача. Значні дані практично жодного разу не йде по стопах берегти на локальному жорсткому диску, в разі якщо ви зможете допомогти.

2.2. Модель збереження даних

Для досягнення поставленої розроблений спосіб відновлення інформаційних блоків при їх розподіленої заощадженні але лише по r на будь-якому з віддалених сховищ, який гарантує реконструкцію всіх r блоків при втраті доступу до 1 сховища, на якому вони зберігаються, а ще 3-х блоків, які зберігаються на інших сховищах.

В рамках вивчення розглядається ця модель заощадження інформації на віддалених носіях. Юзер оберігає на віддалених серверах n інформаційних блоків B 1, B 2, .., B n. Для їх відновлення при складаються m запасних блоків R 1, R 2, ..., R m у вигляді лінійних перетворень над інформаційними блоками:

де aij ∈ {0,1} - бінарні коефіцієнти, які утворюють матрицю A і визначають метод формування запасних блоків даних:

2.3. Опис алгоритму відновлення

Для забезпечення можливості відновлення не менше 3-х втрачених від спільного кількості q = m + n блоків юзера необхідним є дотримання належних умов:

1.Мають бути запорука відновлення 1-го інформаційного блоку в історії, коли крім нього втрачено 2 запасних. Це гарантується тим, власне що будь-який стовпець матриці A зобов'язаний тримати не менше 3-х одиниць.

2. Згодом витрати 1-го запасного блоку всі стовпці матриці A зобов'язані бути різними.

3.Має бути матриця, наприклад, Δ, інтелектуальна стовпцями матриці А, при цьому будь-якими. Ще тримати під-матрицю, наприклад, θ. Завдяки цьому буває помічена ймовірність уявлення кожних 3-х інформаційних блоків у вигляді лінійної композиції інших інформаційних і запасних блоків.

Гарантуючи що найбільш ймовірність відновити 3 всілякі втрачені інформаційні блоки.

2.3.1 Спосіб формування матриці запасних блоків

Для подальшої роботи по матриці формування запасних блоків пропонується ця її зведення:

Розраховується сенс k числа двійкових розрядів для кодування номера колонки за умови, власне що їх нумерація починається з одиниці:

$$\kappa = \lceil \log_2(n+1) \rceil$$

2. В 1-і k рядків всякого стовпчика матриці А записується його порядковий номер, починаючи з одиниці.

3. В (k + 1) -му рядку тих колонок, номера яких мають парне чисельність одиниць, дізнається кількість, а мають непарну кількість одиниць - нуль.

4. В (k + 2) -му рядку тих колонок, номера яких мають 1 одиницю, дізнається кількість, а мають 2 і більше одиниць - нуль. Продемонструємо, власне що матриця А побудована згідно запропонованим способом задовольняє сформульованим вище 3 умов[10].

Тому що нумерація стовпців згідно вищевикладеного настає з одиниці, ручається але б однієї одиниці в перших k рядках всякого стовпця матриці А. Відповідно до п.4 в (k + 2) -му рядку тих колонок, номера яких мають лише тільки 1 одиницю, дізнається коли, наприклад власне що чисельність одиниць в будь-якому стовпці не має можливість бути менше 2-ух.

У разі якщо Хемінгова відстань між парою номерів дорівнює одиниці, то це цілком ймовірно лише тільки за умови, власне що чисельність одиниць в даних номерах виділяється на одиницю, тобто раз з номерів парної, а інший - непарний. Описаний вище спосіб враховує додавання одиниці в (k + 1) -му

рядку лише тільки тих стовпців, сума одиниць в k перших позиціях парна. Це означає, власне що з урахуванням (k + 1) х перших складову Хемінгова відстань між всякий двома стовпчиками матриці А гарантовано не менше 2-ух.

Всілякі 3 різних номера виділяються мінімальна кількість в 2-ух розрядах. Це означає, власне що між перших k рядків матриці А, побудованої за викладеними вище способом, для кожних 3-х стовпців є мінімальна кількість 2 рядки, складові яких не рівні між собою і ці 2 рядки різні.

У разі якщо цих рядків рівно два, то раз не вважається інверсією інше. В іншому випадку були б присутні 2 схожих номера суперечить викладеного способу. У разі якщо цих рядків рівно 2 і в даних 2-ух рядках є розряд, однаковий нулю в обох стрічках, то між інших перших k рядків матриці А слід бути рядок, що складається з одиниць, по іншому номер 1-го з стовпців дорівнює нулю, власне що заходить в протиріччя з викладеним способом.

Цим чином, в разі якщо між перших k рядків обраної трійки стовпців матриці А є рівно 2 цих, складові яких не рівні між собою і ці 2 рядки різні, але не інверсні, то можливо розглядати 2 варіанти. У першому випадку є компонента, що дорівнює нулю в обох стрічках. Але в даному випадку в обов'язковому порядку є і рядок, що складається з одиниць і який, спільно з 2-ма зазначеними вище рядками утворюють ортогональну систему θ. У другому випадку в парі різних і не інверсних рядків в обох з них є 2 одиниці і буквально є компонента, яка в обох стрічках дорівнює одиниці. У разі якщо в обраних 3-х стовпців матриці А в перших k рядках є рядок, всі складові якого дорівнюють одиниці, то дана рядок, спільно з 2-ма згаданими вище сформує ортогональну систему θ. У разі якщо в обраних 3-х номерах немає рядка, що складається з усіх одиниць, то це означає, власне що буквально раз з даних номерів містить парне чисельність одиниць. Отже, згідно викладеного способу в (k + 2) -му рядку даний стовпець має одиницю, а 2 інших - нулі. Відповідно до цього (k + 2) -й рядок спільно з 2-ма розглянутих утворюють ортогональну систему θ. Цим чином підтверджено, власне що за умови, власне що номери 3-х обраних колонок матриці А виділяються буквально в 2-ух розрядах (рядках), в матриці

А буквально є ортогональна квадратна матриця θ, стовпці якої вважаються компонентами обраних стовпців.

У разі якщо номера 3-х обраних колонок матриці А виділяються більше ніж в 2-ух розрядах, тобто не менше 3-х всіляких рядків з нерівними між собою компонентами.

За аналогією з викладеним вище можливо розглядати 2 варіанти. У першому в усіх 3-х різних рядках є компонента рівною нулю, а в другому випадку подібної складової немає.

У разі якщо в даних 3-х рядках є компонента, що дорівнює в їх нулю, то 2 інші складові утворюють пари {0,1}, {1,0}, {1,1}. При цьому в обов'язковому порядку є і рядок, що складається з одиниць. В іншому випадку треба дозволити життя нульового номера, власне що заходить в протиріччя з викладеним способом. У цій ситуації рядок обраної трійки номерів, що складається з одиниць разом з рядками, що містять пари {0,1}, {1,0} утворюють ортогональну систему θ. У другому випадку серед □ перших рядків обраної трійки стовпців матриці А існує мінімум три різні рядки, причому жодна компонента не дорівнює нулю у всіх трьох рядках, ці три рядки утворюють ортогональну систему θ якщо серед них є хоча б один з непарним числом одиниць. Якщо в трійці рядків немає рядка з непарною кількістю одиниць, то вони не утворюють ортогональної системи. Якщо серед інших рядків обраної трійки номерів є рядок, що складається з одиниць, то він утворює ортогональну систему θ з будь-якою парою наведених вище стовпців.

У разі якщо 3 різних рядки трійки номерів рівні {0,1,1}, {1,0,1} і {1,1,0} і всі інші рядки обраної трійки номерів одноманітні наведеним 3-ма або ж складаються з нулів, то між обраного номера в обов'язковому порядку стануть ці, кількість одиниць в яких парне. При цьому чисельність цих стовпців в обов'язковому порядку непарне тому, власне що кожен рядок трійки номерів має парне чисельність одиниць. Згідно викладеного способу в (k + 2) -му рядку в цих стовпцях знаходяться одиниці. Відповідно до цього, (k + 2) -му рядку знаходиться щоразу непарне чисельність одиниць. Даний рядок спільно з будь-

парою рядків {0,1,1}, {1,0,1} і {1,1,0} утворюють ортогональну систему. Наприклад, номери 7, 9 і 14 містять 4 цих рядки: {0,1,1}, {1,0,1}, {1,0,1} і {1,1,0}. Сума одиниць в номері 7 непарна, в номері 9 парна і в номері 14 - непарна. В наслідок цього (k + 2) -й рядок складається у вигляді {0,1,0}, тобто має непарне чисельність одиниць. Безперечно, власне що дана рядок спільно з рядками {0,1,1}, {1,0,1} утворюють ортогональну систему θ.

Цим чином, підтверджено, власне що матриця А сформована по викладеним способом цілком відповідає умовам, які обумовлюють гарантовану можливість всіх втрачених інформаційних блоків за умови витрати не більше 3-х від спільного кількості інформаційних і запасних блоків.

З запропонованого способу безперечно, власне що чисельність запасних блоків для гарантованого відновлення інформаційних пакетів при втраті не більш 3-х від спільного числа блоків орієнтується формулою зображеної на малюнку 8.

$$m \geq \lceil \log_2(n+1) \rceil + 2 .$$

Рис. 8 «Формула кількості резервних блоків»

На базі виконаних теоретичних досліджень запропонований спосіб, який гарантує реконструкцію всіх r блоків при втраті доступу до 1 сховища, на якому вони зберігаються, а ще 3-х блоків, які зберігаються на всіляких сховищах.

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

Рисунок 9 - «Матриця А формування резервних блоків»

При втраті всіх блоків В 9, В 10, ..., В 12, які зберігаються на 3-му сховище, належні стовпці матриці А, зображеної на малюнку 9 – «Матриця А формування резервних блоків» містять в собі ортогональну під- матрицю θ, зображену на малюнку 10:

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

Рис.10 – «Ортогональна під-матриця θ матриці А»

Відповідно, втрачені блоки, як показано на рисунку 11 відновлюються наступним порядком:

$$B_9 = R_6 \oplus B_2 \oplus B_3 \oplus B_5 \oplus B_6 \oplus B_{14}$$

$$B_{10} = R_3 \oplus B_2 \oplus B_4 \oplus B_7 \oplus B_8 \oplus B_{13} \oplus B_{14} \oplus B_{15}$$

$$B_{11} = R_4 \oplus B_1 \oplus B_3 \oplus B_5 \oplus B_7 \oplus B_8 \oplus B_{10} \oplus B_{13}$$

$$B_{12} = R_1 \oplus B_{11} \oplus B_4 \oplus B_5 \oplus B_6 \oplus B_7 \oplus B_8 \oplus B_{14}$$

Рис. 11 – «Відновлення втрачених блоків з під-матриці «вставити символ»»

При втраті трьох блоків В 1 , В 7 , В 12 , що зберігаються на різних сховищах, відповідні стовпці матриці А містять ортогональну під-матрицю □, зображену на рисунку 12:

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

Рис.12 – «Ортогональна під-матриця θ матриці А»

Відповідно, втрачені блоки, як показано на рисунку 13 відновлюються наступним порядком:

$$B_7 = R_3 \oplus B_2 \oplus B_4 \oplus B_6 \oplus B_{10} \oplus B_{13} \oplus B_{14} \oplus B_{15}$$

$$B_{12} = R_1 \oplus B_7 \oplus B_4 \oplus B_5 \oplus B_6 \oplus B_8 \oplus B_{11} \oplus B_{12} \oplus B_{14}$$

$$B_1 = R_5 \oplus B_2 \oplus B_3 \oplus B_6 \oplus B_9 \oplus B_{10} \oplus B_{12} \oplus B_{15}$$

Рис. 13 – «Відновлення втрачених блоків з під-матриці «вставити символ»»

Цим чином, запропонований спосіб формування запасних блоків і відновлення інформаційних блоків при їх розподіленої заощадженні але лише по r на будь-якому з віддалених сховищ, ручається реконструкцію всіх r блоків при втраті доступу до 1 сховища, на якому вони зберігаються, а ще кожних 3-х інформаційних блоків, які зберігаються на всіляких сховищах.

Провідною ефект запропонованого способу полягає в тому, власне що важливим чином зменшується надмірність резервування в порівнянні з популярними способами за рахунок обліку індивідуальностей витрати доступу до відомостей при їх віддаленому заощадженні. Надмірність резервування зменшується в $\mu$ раз, чисельне значення $\mu$ орієнтується формулою зображеної на малюнку 14:

$$\mu = \frac{(r-2)\cdot \log_2(n+1)+2}{r+2}.$$

Рис. 14 – «Формула для визначення надлишковості резервування»

Зокрема, для наведеного прикладу чисельність запасних блоків для відновлення 4-х, довільно локалізованих по сховищам блоків, дорівнює 10, а в запропонованому способі, який дозволяє відновлювати 4 блоки, що зберігаються на одному сховищі і три, довільно локалізованих блоків, треба 6 запасних блоків . Відповідно до цього, надмірність резервування зменшується в $\mu$ = 1.67 раз.

В результаті виконаних досліджень на теоретичному рівні обумовлено, розроблений спосіб відновлення інформаційних блоків при їх розподіленої заощадженні але лише по r на будь-якому з віддалених сховищ, що базується

на застосуванні для формування запасних блоків матриці, вважається ортогональної в межах сусідніх r стовпців, а ще всякий які 3 стовпці якої містять ортогональну під-матрицю, ручається реконструкцію всіх r блоків при втраті доступу до 1 сховища, на якому вони зберігаються, а ще 3-х блоків, які зберігаються на всіляких сховищах. Спосіб користується найменше чисельність запасних блоків, тобто містить більше невисокий ступінь надмірності бронювання в порівнянні з популярними способами, націленими на відновлення кожних r втрачених блоків. Подальше становлення запропонованого розкладу показується в дослідженні ймовірностей прискорення процесу відновлення втрачених при віддаленому заощадженні даних за рахунок оптимізації їх розміщення на віддалених сховищах по аспекту мінімізації часу доставки блоків, важливих для реконструкції втраченої інформації користувачів.

Для передових похмурих сховищ час відновлення даних втрачених або ж затриманих більше небезпечний зазор часу можна розділити на 2 елементи. 1-ша доставки провідних блоків даних, спільно з запасними, для подальшого відновлення перших. 2-а елемент - це час виконання обчислень, проведених при втраті провідних блоків. В обчисленнях провідну роль відіграють запасні блоки ще кожних 3-х інформаційних блоків, які зберігаються на всіляких сховищах.

## 2.4. Опис схеми алгоритму відновлювання даних

Схема у другому додатку описує побудови матриці формування резервних блоків, за допомогою якої, будудть відновлюватися втрачені блоки.

Спочатку задаються початкові дані r та n. Де r – загальна кількість стопчиків матриці A, яку можна бачити на рисунку 15 - «Матриця A, утворена за допомогою алгоритму»

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

Рис. 15 – «Матриця А, утворена за допомогою алгоритму»

Далі розраховується кількість фрагментів за формулою - $g = [n/r]$. Також, основною умовою є те, що перші g – 1 фрагменти мають в собі по r стовпців. А останній, g – й фрагмент містить в собі кількість столбвці, яка розраховується за формулою $n - r(g - 1)$.

Спочатку задаємо змінній j значення 0, це буде змінна наших ітерацій. Далі працюємо з рядком кожного і-го фрагменту матриці А, зображеної на малюнку 15 – «Матриця А, утворена за допомогою алгоритму», який знаходиться за формулою $(i + j)\ mod\ r$, де mod, операція ділення без остачі. Змінна і, в свою чергу, належить до проміжку $i \in \{1, 2 \dots g\}$. Кожен рядок, який ми попередньо вирахувализ заповнюється кодом, розрядність якого визначається за формулою $r - j$, що являє собою одиницю на певній позиції. Ця позиція визначається наступною формулою $(r + j - 1 - i) mod\ r$, також вона визначає циклічно наступні позиціє за нею в рамках r – бітового фрагменту нулів.

Далі, як можна бачити на блок схемі алгоритму побудови матриці резервних блоків, за допомогою якої відпувається відновлення всіх блоків одного та трьох блоків різних сховищ, відбувається інкрементація нашого лічильника, змінної j. Одразу після цього, йде перевірка чи значення r, задане на початку і чисельно характеризуюче кількість стовпців в нашій матриці формування резервних блоків, зображеної на рисунку 15 – «Матриця А,

утворена за допомогою алгоритму», більше значення j. Якщо це так, то проводовжуємо заповнювати рядки, кожного і-го фрагменту матриці А. Якщо умова не виконується, продовжуємо виконувати дії нашого алгоритму.

Потім, в алгоритмі йде заповнення перших невизначених елементів матриці А, утвореної за допомогою методу формування резервних блоків. Елементи мають знаходитися на рядках, номери яких розраховуються за наступною формулою $[log_2(n+1)]$. Також важливим параметром заповнення рядків, щоб невизначенні елементи в купі з визначеними утворювали в стовпці матриці А унількальний номер, починаючи з одиниці.

В кінці треба звернути уваги на всі стовпці матриці А, та порахувати кількість одиниць в них. Якщо кількість одиниць менша трьох – треба заповнити незаповнений елемент одиницею.

## 2.5. Висновки

В даному розділі було розкрите поняття відновлення даних та яку роль воно відіграє в сучасному світі. Також, було продемонстровано модель збереження даних та на основі цієї моделі розглянуто алгоритм відновлення даних. Досліджений алгоритм відновлення всіх блоків одного та трьох блоків різних сховищ працює за допомогою матриці резервних блоків, які передаються разом з основними при передачі інформації на хмарне сховище. Ця процедура відіграє ключову роль в відновленні даних за допомогою даного методу. Також, на основі схеми Д2 покроково розказано про алгоритм дії методу відновлення всіх блоків одного та трьох блоків різних сховищ.

# РОЗДІЛ 3.Реалізація відновлення втрачених даних на мові VHDL

## 3.1. Пояснення основних понятть мови VHDL

### 3.1.1. Представлення мови

VHDL значить мова опису оснащення VHSIC (дуже високошвидкісні інтегральні схеми). В середині 1980-х років Міністерство захисту USA і IEEE спонсорували розробку цього мови опису апаратного забезпечення з метою розробки досить швидкодіючої інтегральної схеми. В даний момент він став одним з нормальних мов, застосовуваних для опису цифрових систем. Інший широко застосовуваний мова опису оснащення - Verilog. Обидва вважаються сильними мовами, які дають можливість описувати і моделювати важкі цифрові системи. 3-ий мову HDL - це ABEL, який був навмисне розроблений для програмованих логічних приладів (PLD).[12]

Але ці мови скидаються на звичайні мови програмування, між ними є деякі значущі відмінності. Мова опису оснащення за своєю суттю вважається паралельним, тобто команди, належні закономірним складовим, виробляються (обчислюються) паралельно, як тільки-но надходить свіжий вхід. Програма HDL імітує поведінку фізіологічної, як правило цифровий, системи. Він також дозволяє підключати короткочасні властивості (затримки затвора), а ще описувати систему як зв'язок всіляких компонент.

Аналогічно що, як мови програмування високого значення дають можливість формулювати важкі концепції проектування у вигляді комп'ютерних програм, VHDL дозволяє закріплювати поведінку важких електричних схем в системі проектування для автоматичного синтезу схем або ж для моделювання системи. Аналогічно Pascal, C і C ++, VHDL підключає функції, потрібні для способів структурованого проектування, і пропонує багатий комплект функцій управління і представлення даних. На відміну від

даних інших мов програмування, VHDL дає функції, що дозволяють описувати паралельні дії.[11]

Це принципово, внаслідок того власне що апаратне забезпечення, що описується з підтримкою VHDL, за своєю суттю вважається паралельним в своїй роботі. Користувачі мов програмування PLD, таких як PALASM, ABEL, CUPL і інші, знайдуть паралельні функції VHDL досить своїми людьми. Що не менше, тим, хто програмував лише тільки з впровадженням мов програмування, буде необхідно освоїти деякі свіжі концепції.

Процес розробки VHDL настає з написання програми VHDL. Всілякі виробничі фірми, такі як XILINX і т. Д., Дають власні особисті інструменти розробки програмного забезпечення , такі як XILINX ISE, Quartus і т. Д., Для редагування, компіляції та моделювання коду VHDL. В даному коді VHDL схема описана в RTL (рівень передачі резистора).

Даний код VHDL компілюється і генерує перелік з'єднань на рівні шлюзу. Компілятор конвертує високорівневий код VHDL в RTL в ступінь шлюзу.

Даний перелік з'єднань додатково оптимізований, щоб знову отримати оптимізований перелік з'єднань на рівні воріт. Оптимізація виготовлена для найкращої швидкості і найменшого місця. На цьому етапі проводиться моделювання дизайну.

Зрештою, тілесне прилад реалізується на CPLD / FPGA, або ж безповоротна личина готується для ASIC з цього оптимізованого переліку ланцюгів по простору і програмного забезпеченню маршруту (установник). ще один остаточне прилад можливо створити і з'ясувати. Схему цієї взаємодії можна бачити на малюнку 16 – «схема проектування VHDL»

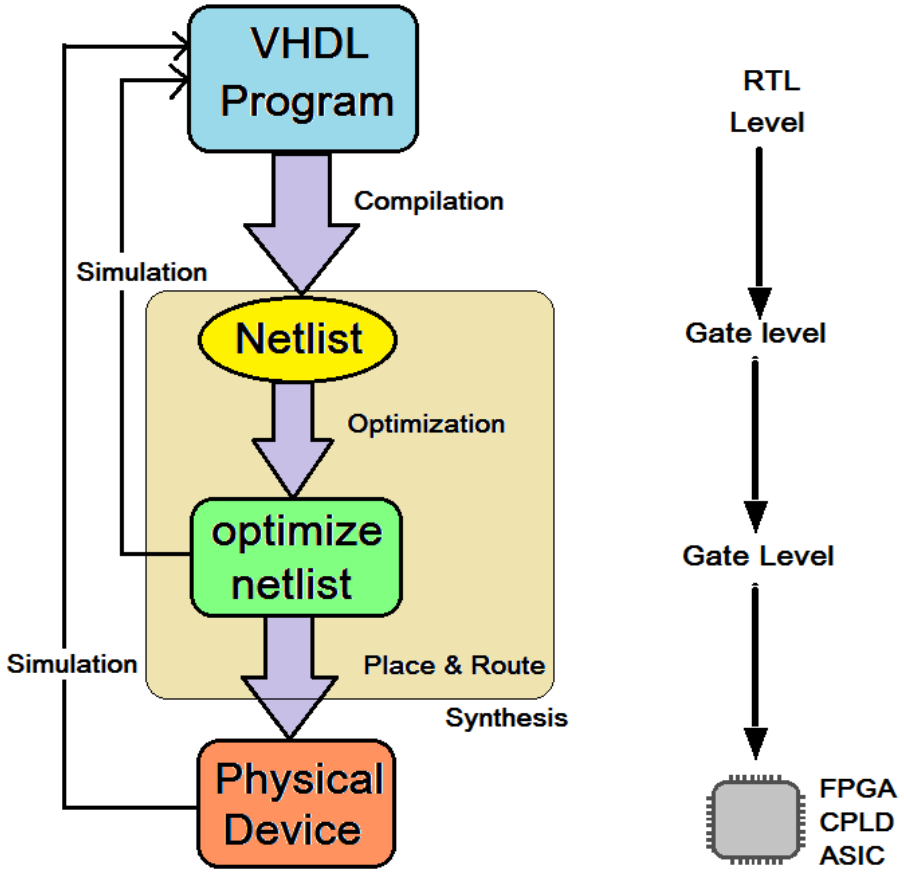


Рисунок 16 – «Схема проектування VHDL»

3.1.2. Перевага VHDL над схемним проектуванням

Проектування величезних обчислювальних приладів (ВУ) - З підтримкою VHDL легше і швидше встановити і з'ясувати величезний план. 10 термінами VHDL можна охарактеризувати як 1, наприклад і 100000 тригерів. Мікросхему з інтеграцією більше 10000 вентилів створити лише тільки з підтримкою електричних схем досить непросто по підставі громіздкості схем.

План на VHDL-об'єднання структури ВУ і методу його функціонування. Для ВУ, описаного на VHDL, необов'язково виконувати випробування коректності його функціонування, наприклад, методом його макетування. Щоб визначити, чи правильно ВУ робить даний метод, досить його VHDL-програми запустити на виконання в симуляторі VHDL. Належні САПР конвертують VHDL-опис в набір документації для приготування працездатного приладу.

План на VHDL – сам документується, тобто він не вимагає допоміжного технічного опису або ж у вигляді схем. Нечіткість і зверхність опису виключаються, наприклад як план на VHDL неважко з'ясувати[12].

Найвища надійність плану. Синтаксичний тест, моделювання і компіляція в закономірну схему моторно виявляють промахи плану.

План на VHDL - універсальний план. Створений колись обчислювальний блок має можливість бути застосований в багатьох інших планах. При цьому майже всі структурні і активні характеристики блоку можуть бути налаштованим (параметри розрядності, розміру пам'яті, елементна основа, склад блоку і конструкція з’єднань).

План на VHDL - мініатюрний план. Створений для однієї елементної бази, план ВУ без праці переноситься на іншу елементну підставу, напр. НВІС з різною технологією.

В даній манері моделювання поведінку об'єкта як комплекту операторів проводиться по черзі в позначеному порядку. Лише тільки оператори, розміщені зсередини процес, функція або процедура, вважаються послідовними.

Процеси, функції та процедури- єдині сегменти коду, які виробляються по черзі.

3.1.3. Структура програми VHDL

Структура програми VHDL зображено на малюнку 17 – «Структура програми VHDL»

```
LIBRARY library_name;
USE library_name.package_name. package_parts;

ENTITY entity_name IS
     PORT ( port_name : port_mode port_type;
            port_name : port_mode port_type;
            port_name : port_mode port_type;
                .
                .
                .
          );
END entity_name;

ARCHITECTURE archi_name OF entity_name is
   declarations
     BEGIN
          code (sequential or concurrent statements)
          .
          .
          .
END archi_name
```

Рисунок 17 – «Структура програми VHDL»

Бібліотека має цілий нерідко застосовуваний шматок коду. Це дозволить нам застосувати їх знову і знову. Ще його можна застосувати з іншими дизайнами. Він настає з головного тексту LIBRARY, за яким йде по стопах ім'я бібліотеки. У всіх кодах VHDL як правило застосовуються 3 бібліотеки. IEEE - визначає багаторівневу закономірну систему, std - книгосховище ресурсів для середовища проектування VHDL, work - застосовується для зберігання нашої роботи за планом і файлу програми (.vhd)[14].

Втім в програмному коді нам треба оголосити лише тільки бібліотеку IEEE, внаслідок того власне що 2 інші бібліотеки вважаються бібліотеками за замовчуванням.

Нині, щоб додати пакети бібліотеки і її частка, головне текст USE застосовується з ім'ям бібліотеки, пакетами бібліотеки і частинами пакета. Наприклад, в бібліотеці IEEE це згорток std_logic_1164, і щоб додати всю його частка, ми можемо скласти[13].

Суть визначає з'єднання введення-виведення цифрової схеми, з якими вона має можливість взаємодіяти з іншими компонентами / схемами.

Він заявляє чисельність входів, даних схемою, і чисельність виходів, витягнутих зі схеми.

Крім такого, він заявляє всілякі проміжні сигнали, які застосовуються в самій схемі.

Афіша об'єкта настає з головного тексту ENTITY. Користувач повинен надати бажане ім'я суті, нерідко пов'язаної зі схемою, яка замислюється як мультиплексор, декодер, суматор, лічильник і т. Д. (Практичне правило для будь-програмки VHDL - це ім'я файлу програми слід бути тим же, як ім'я об'єкта)

Зсередини об'єкта контакти введення-виведення схеми оголошуються з підтримкою головного тексту PORT.

ПОРТ (означає взаємодію контактів) оголошується з іменем порта, режимом і типом порта

port_name - визначається користувачем ім'я контакту введення-виведення

port_mode - є 4 на подобі режиму порту: IN, OUT, INOUT і BUFFER. IN означає вхідний контакт, дешевий лише тільки для читання. OUT показує вихідний контакт і його доступ лише тільки для запису. Обидва ці штифта направленим в одну частину. Висновок INOUT вважається направленим в дві частини, його можна читати і писати. Буфера застосовується для проміжного виведення.

port_type - має можливість бути бітом, бітом-вектором, логікою стд і т. д. Згодом оголошення всіх інтерфейсів афіша об'єкта завершується головним текстом END, за яким йде по стопах ім'я об'єкта.

Давайте поглянемо на зразок суті, зображеної на малюнку 18 – «Зразок суті(entity)» для закономірного речовини I з 2-ма входами.

```
ENTITY and_gate IS
   PORT (a, b : IN BIT;
            y : OUT BIT);
END and_gate;
```

Рисунок 18 – ««Зразок суті(entity)»

3.2 Опис пристрою на HDL

Перш за все розберемося з встановленої завданням. Ми маємо намір об'єднати лічильник і регістр в єдиний речовина. Наприклад як лічильник і регістр зобов'язані виконувати кілька операцій, то нам потрібно диференціювати завдання надходять новенькому складової. Для цього потрібно закодувати вхідні пам'ятці. Створимо це наступним чином як показано на таблиці 1 –«Кодування вхідних інструкцій»:

| Код інструкції | Результат Виконання |
|---|---|
| 000 | Зберігання даних |
| 001 | Загрузка даних |
| 010 | Інкрементування |
| 011 | Декрементування |
| 100 | Зсув вліво |
| 101 | Зсув вправо |
| 110 | Інвертування даних |

Рисунок 19 – «Кодування вхідних інструкцій

Після задання початкових даних для нашого пристрою, можна перейти до кодування на мові HDL. Детально розглянемо кодування пристрою – почнемо з бібліотек, які підключені до нашого розділу та зображені на малюнку 20 – «Бібліотеки пристрою»

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

Рисунок 20 – «Бібліотеки пристрою»

Гігантська частка виробників САПР в даний момент підтримує пакети numeric_bit / numeric_std. Бібліотеки IEEE вважаються більше бажаним варіантом. І для отримання найбільшої платформ коду на VHDL, варто ігнорувати застосування пакетів std_logic_signed або ж std_logic_unsigned і нерідко застосувати типи SIGNED або ж UNSIGNED (на останній випадок INTEGER) для арифметичних операцій.

Далі, подивимося на малюнок 21 – «Опис інтерфейсу пристрою»

```
entity recovery_model is
generic (
Delay: time:=20ns);
port (
CLK: in STD_LOGIC;
Reset: in STD_LOGIC;
DataIn: in STD_LOGIC_VECTOR (3 downto 0);
DataOut: out STD_LOGIC_VECTOR (3 downto 0);
Command: in STD_LOGIC_VECTOR (2 downto 0));
end entity recovery_model;
```

Рисунок 21 – «Опис інтерфейсу програми»

Суть(entity) застосовується в поєднанні з архітектурою. Спільно вони описують поведінку або ж структуру ієрархічного блоку оснащення (об'єкт проектування).Архітектура має можливість бути призначена лише тільки 1 об'єкту, але раз об'єкт має можливість бути призначений декільком архітектурам.

Суть заявляє назву дизайну. Крім такого, він визначає спільні типи, які дають статичну інформацію (наприклад, короткочасні характеристики або ж ширину шини) для плану, і порти, які забезпечують канали зв'язку між планом і його середовищем.

Оголошенню об'єкта можуть передувати пропозиції бібліотеки і застосування. Цим чином, всі оголошення, конкретні в пакеті, стануть видимі для об'єкта і всіх наданих йому архітектур.

В опису є надання часу затримки сигналу в розмірі 20нс, також є змінна вхідного порту для тактового сигналу, сигналу, скидань. Також описані інформаційні входи і виходи. Саме на них буде поступати інформація разом з резервними блока, завдяки яким будуть відновлюватися втрачені дані. В нашої моделі буде декілька сюжетів розвитку. В залежності від того, скільки блоків даних втрачено, чи зможемо ми побудувати матрицю резервних блоків, які формуються разом з основними при передачі на хмарне сховище для їх подальшого відновлення.

Тепер, подивимося на опис архітектури, зображений на рисунку 22 – «Опис архітектури ч.1»

```
architecture recovery_model of recovery_model is
begin
process(CLK,Reset)
```

Рисунок 22 – «Опис архітектури ч.1»

Як можна бачити, в наведеному вище фрагменті я використовував синхронний скидання. Якщо нам би був абсолютно не потрібен асинхронний скидання, ми би використовували замість нього синхронний скидання. Це допомагає синтезатору, оскільки є деякі спеціальні конструкції, які недоступні при асинхронному скиданні, і допомагає запобігти проблемам, коли ваш дизайн стає великим (і тригери раптово починають скидатися в різний час з-за перекосу сигналу).

Задаємо нашому вектору поведінку показану на малюнку 23 – «Опис архітектури ч.2».

```
variable
private:STD_LOGIC_VECTOR (3 downto 0);
```

Рисунок 23 – «Опис архітектури ч.2»

Як можна бачити з рисунка 23, наведеного вище, наш процес буде реагувати на зміну синхросигналу та сигналу скидання.

Далі, нам потрібно зберігати нинішній стан, для цього введемо додаткову змінну, зображеної на рисунку 24 – «Опис архітектури ч.3»

```
begin
if (Reset='1') then private:="0000";
```

Рисунок 24 – «Опис архітектури ч.4»

Якщо відбувається скидання, то значення змінної, яке зберігає нинішній стан лічильника зберігається або стає рівним нулю. Цю умову демонструє рисунок 25 – «Опис архітектури ч.5»

```
elsif (CLK'event) and (CLK='0') then
```

Рисунок 25 – «Опис архітектури ч.5»

Інакше, якщо відбувається зміна сигналу синхронізації та він стає низького рівня.

Сигнали підтримуються для синтезу за умови, власне що вони мають тип, прийнятний для інструменту закономірного синтезу.

Регістр видів сигналів покришки як правило ігнорується.

Підтримуються лише тільки конкретні дозволені типи сигналів. Основна маса інструментів розпізнають типи std_logic_1164.

Починаємо обробку події, як показано на рисунку 26 – «Обробка події»

```
case Command is
```

Рисунок 26 – «Обробка події»

Для ідентифікування інструкції керуваннями дій пристрою ми застосовуємо оператор case <name> is

Далі треба поставити умову, зображену на рисунку 27 - «Умова для перевірки вхідних даних на вході», для нашого пристрою.

```
when "000" =>
null;
```

Рисунок 27 - «Умова для перевірки вхідних даних на вході»

Якщо вхідна інформація 000, то згідно умови зберігається результат, тобто нічого. Якщо вхідна інформація 001, зберігаються дані в блоках інформації, як показано на рисунку 28 – «Збереження інформації в блоці за допомогою кодування».

```
when "001" =>
private:=datain;
```

Рисунку 28 – «Збереження інформації в блоці за допомогою кодування»

В залежності від інструкції збільшуємо або зменшуємо значення внутрішньої змінної , як показано на рисунку 29 – «Зміна внутрішньої змінної в залежності від кодування»

```
when "010" =>
private:=private+3;
when "011" =>
private:=private-3;
```

Рисунок 29 – «Зміна внутрішньої змінної в залежності від кодування»

На даному етапі здійснюється ідентифікація виду відновлення блоків, в залежності від кількості втрати блоків або взагалі – втраті хмарного сховища. Відновлення може бути наступних видів: відновлення при втраті одного блока, двох блоків, трьох, більше трьох та відновлення всього сховища. Саме ці значення ми задаємо нашій змінній. Оператор & означає конкатенацію, тобто зчеплення бітів аргументу. В першому прикладі, де ми описуємо вектор за логікою std_logic, результат формується з перших трьох бітів значення вектору і одного біту зі значенням 0.

В інших випадках нічого не відбувається, на мові HDL це виглядає так, як показано на рисунку 30 – «Всі інші випадки пристрою»

```
when others =>
null;
```

Рисунок 30 – «Всі інші випадки пристрою»

Далі відбувається процес відновлення втрачених блоків при передачі інформації в хмарне сховище. В залежності від кількості втрачених блоків будуть різні методи відновлення. В процесі відновлення беруть участь матриця формування резервних блоків, кожен елемент якої 4 розрядне слово. В матриці виділяємо під – матрицю і відновлюємо втрачені блоки за допомогою резервних, виконуємо ці дії за допомогою операції XOR. Після цього виводимо відновлені блоки інформації, як показано на рисунку 31 – «Вивід відновлених блоків»

```
end case;
end if;
DataOut<=private after Delay;
B1<=private after Delay;
B2<=private after Delay;
B3<=private after Delay;
```

Рисунок 31 – «Вивід відновлених блоків»

Після цього всі дії виконані, модель побудована на мові HDL, відновленні втрачені блоки інформації за допомогою резервних. Відновлення проводилося завдяки матриці резервних блоків, які формуються разом з

основними при передачі даних на хмарне сховище, для цього використовувалася операція XOR.

### 3.3 Тестування та роботоспосібність пристрою

Якщо вхідний сигнал 000 то у нас ніяких змін, тобто ніякої втрати інформації немає. Всі блоки які були в точці А успішно прибули в точку Б, це можна бачити на рисунку 32 – «Збереження всіх блоків»

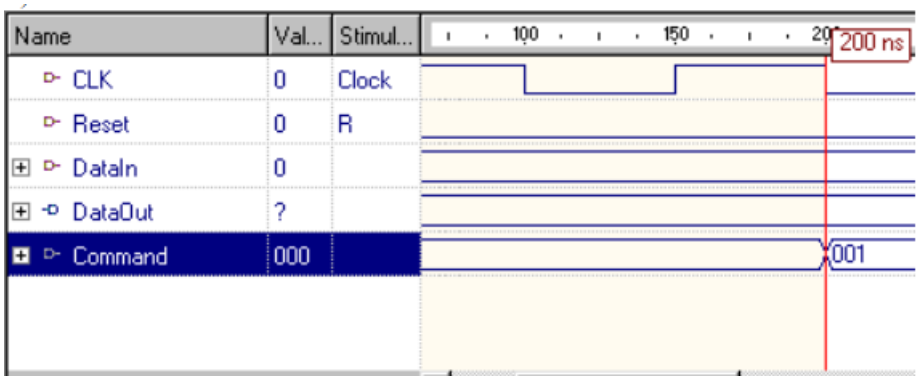


Рисунок 32 – «Збереження всіх блоків»

Якщо втрата одного інформаційного блоку, після процесу відновлення, отримаємо теж саме слово, що і на вході, як можна бачити на рисунку 33 – «Результат відновлення одного блоку»

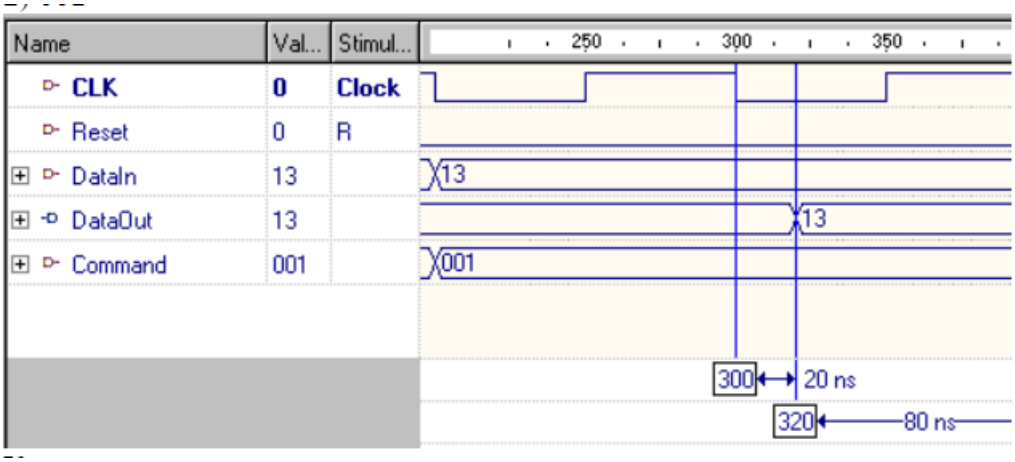


Рисунок 33 – «Результат відновлення одного блоку»

Якщо відновлення двох блоків, то відбувається додавання трьох до тринадцяти – це виходить 10000 , в залежності від розряду змінної, в якій зберігається результат, така і буде змінна результату. В нашому випадку зберігання 4-х розрядного слова – результат буде 0 («0000»). Результат

відновлення двох блоків можна бачити на рисунку 34 – «Результат правильної роботи пристрою».

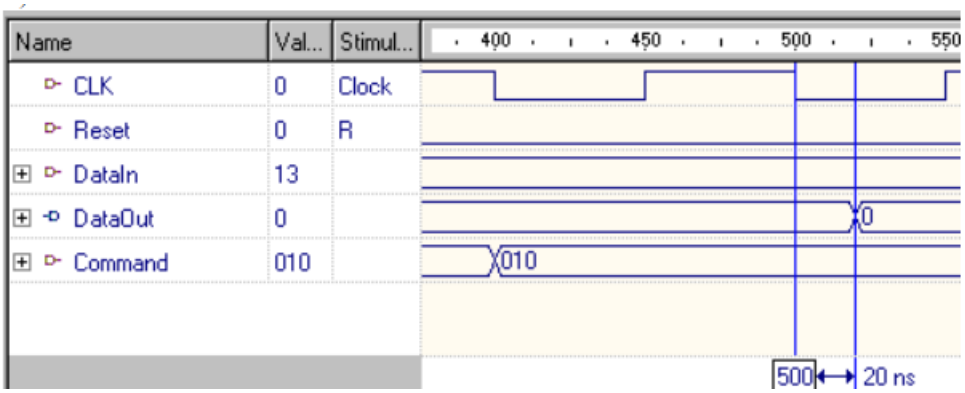


Рисунок 34 – «Результат правильної роботи пристрою»

Далі перевіряємо роботу пристрою при відновленні трьох блоків, впевнюємося в тому, що пристрій працює, подивившись на рисунок 35 – «Кінцева перевірка пристрою»

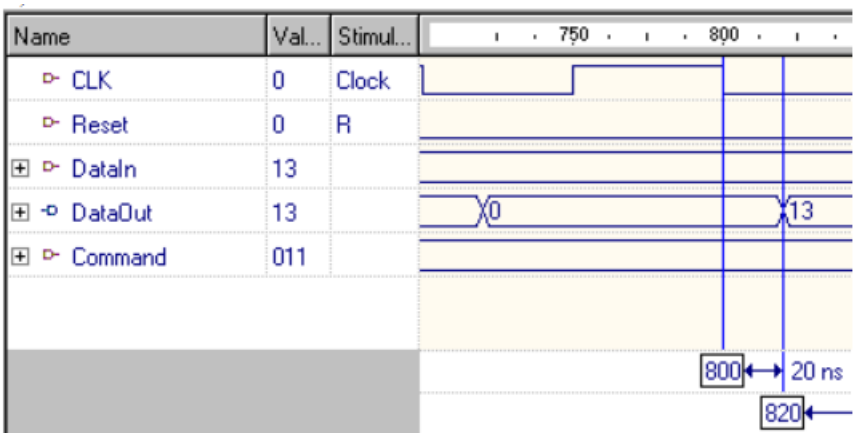


Рисунок 35 – «Кінцева робота пристрою»

Також робота пристрою на виході продемонстрована на рисунках 35 – «Відновлення сховища, скидання» та рисунку 36 – «Відновлення > 4 блоків, зсув вправо»

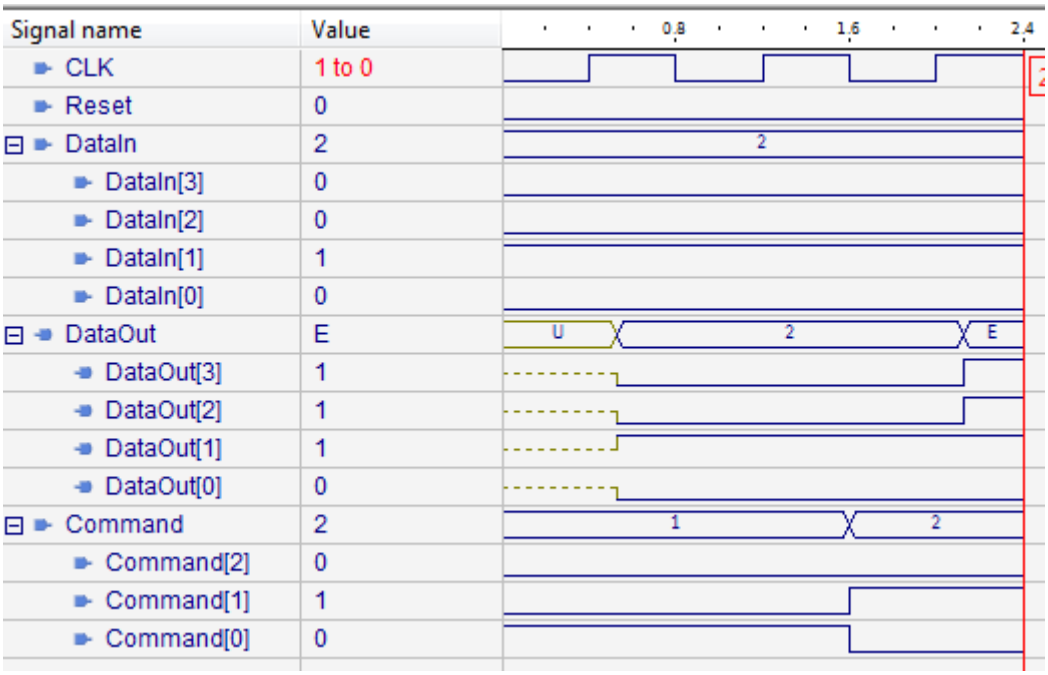


Рисунок 35 – «Відновлення сховища, скидання»

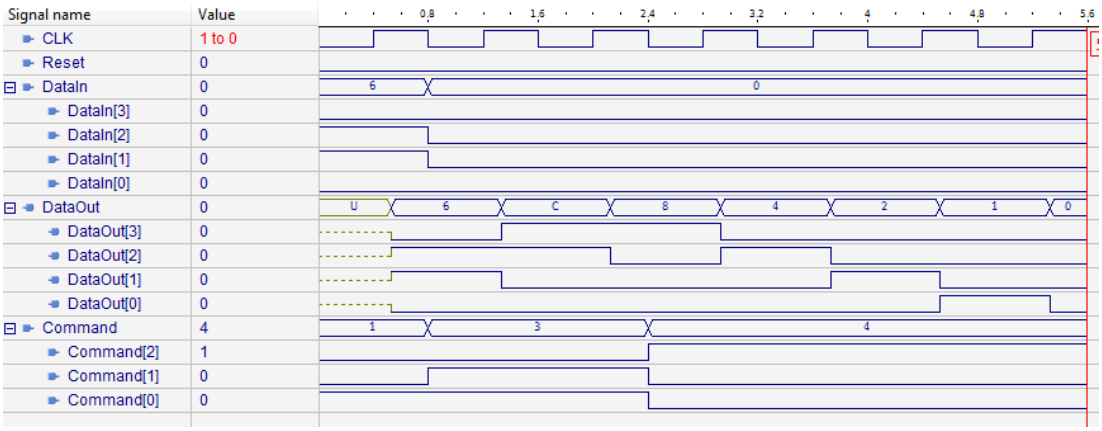


Рисунок 36 – «Відновлення > 4 блоків, зсув вправо»

## 3.4 Висновки

В цьому розділі була представлена мова розробки моделі відновлення даних – HDL, мова високого рівня для описання пристроїв. В розділі були поясненні основні поняття мови, логіка мови та принцип взаємодії та побудови пристроїв на цій мові. Також було аргументовано підтверджено вибір цієї мови для розробки пристрою та її переваги в порівняні з іншими реалізаціями. Було описано структуру програми HDL, як загальну, так і основну представлену в дипломному проеткі.

Також, було показана реалізована модель на мові програмування HDL, яка працює за методом відновлення всіх блоків одного та трьох блоків різних сховищ. Модель побудована на основі матриці формування резервних блоків, які передаються разом з основними, для подальшого відновлення. На основі зроблених тестувань, можна переконатися в роботі пристрою.

# Висновки

В даному дипломному проекті було розглянуто проблему відновлення даних. Відновлення даних відіграє велику роль в сучасному світі. Дуже багато людей користуються своїми гаджетами та іншими пристроями, завдяки яким, вони можуть зберігати та обмінюватися інформаціє та ще багато інших маніпуляцій.

Було розглянуто способи збереження даних на реальних прикладах та помилки, які допускаються при роботі з хмарними сховищами. Також, аргументовано вибір користувачів на користь хмарних сховищ – це популярно через величезну кількість факторів. Крім цього, перераховані фактори, через які корпорації втрачають дані своїх клієнтів, тобто інформація тимчасово втрачається або затримується понад критичний проміжок часу. Це може бути не провина компанії, а стихійні лиха, перепад напруги, тощо. Проте компанії, через бажання не втрачати свою репутацію відновлюють дані користувачів шляхом алгоритмами відновлення.

Проведено аналіз основних алгоритмів, які використовуються, для відновлення даних, які зберігаються на хмарних сховищах. Вибраний метод відновлення всіх блоків одного та трьох блоків різних сховищ через свою швидкість та ефективність.

Також був описаний алгоритм відновлення, зображений на блок – схемі алгоритму Д2.

Модель відновлення була побудована на основі матриці резервних блоків, які формуються разом з основними для подальшого відновлення втрачених блоків. Матриця бере за основу модель збереження даних, описану в даному дипломному проекті.

В кінці-кінців, було описано мову, на якій було реалізовано потрібну в дипломному проекті модель. Обґрунтовано і показано переваги цієї мови в даному випадку та інші переваги HDL. За допомогою кінцевих тестів, можемо

бачити відновлення втрачених або затриманих понад критичний проміжок часу блоків даних методом відновлення всіх блоків одного та трьох блоків різних сховищ.

# Список Використаних джерел


1) Актуальность восстановления данных. [Електронний ресурс]. Режим доступу: http://www.dfacto.ru/presscenter/news/informatcionnye_tehnologii_v_fokuse/aktualnost_problemy_vosstanovleniya_dannyh/
2) Відновлення даних. [Електронний ресурс]. Режим доступу: http://h-disk.blogspot.com/p/blog-page.html
3) Стаття про відновлення даних. [Електронний ресурс]. Режим доступу: https://datalabs.ru/pages/articles
4) Безпека хмарних сховищ. [Електронний ресурс]. Режим доступу: https://datami.ua/bezpeka-hmarnih-shovishh-i-tehnologij-osnovni-pravila/
5) Найпопулярніші хмарні сховища. Плюси і мінуси. [Електронний ресурс]. Режим доступу: http://ipkey.com.ua/uk/faq/942-cloud-technologies.html
6) Як забезпечити захист даних? [Електронний ресурс]. Режим доступу: https://eset.ua/ua/blog/view/93/zashchita-dannykh-v-oblake-kak-bezopasno-khranit-fayly-na-google-diske
7) Хмарні сховища та їх безпека. [Електронний ресурс]. Режим доступу: http://www.dut.edu.ua › uploads
8) Пуасонівський потік. [Електронний ресурс]. Режим доступу: http://stratum.ac.ru/education/textbooks/modelir/lection28.html
9) Відновлення файлів. [Електронний ресурс]. Режим доступу: https://starusrecovery.com.ua/articles/signature_search.html
10) Методи та алгоритми побудови хмаркових сховищ. [Електронний ресурс]. Режим доступу: https://lpnu.ua/sites/default/files/2020/dissertation/1551/disstrubyckyy2.pdf
11) VHDL tutorial. [Електронний ресурс]. Режим доступу: http://www.eecs.umich.edu/courses/doing_dsp/handout/vhdl-tutorial.pdf

12) Introduction to VHDL. [Електронний ресурс]. Режим доступу: https://ufsj.edu.br/portal2-repositorio/File/nepomuceno/fpga-vhdl/vhdl-altera.pdf
13) Basic VHDL tutorials [Електронний ресурс]. Режим доступу: https://vhdlwhiz.com/basic-vhdl-tutorials/
14) Intel about HDL. [Електронний ресурс]. Режим доступу: https://www.intel.com/content/www/us/en/programmable/support/training/course/ohdl1110.htm