Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Моделювання інтерфейсів комп'ютерних систем для навчання студентів

(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІ-21-4
Андрій ШИНГУР
(власне ім’я, прізвище)

Спеціальність
123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп’ютерна інженерія
(повна назва освітньої програми)

Керівник: ст.викл. Дмитро РОСІНСЬКИЙ
(посада, власне ім’я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім’я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Шингуру Андрію Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Моделювання інтерфейсів комп'ютерних систем для навчання студентів

затверджена наказом по університету від “ 26 ” травня 2025 р. № 424 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 17 червня 2025 р.

3. Вхідні дані до роботи

1) інтерфейси: PCIe, DMI, Chipset
2) середовище розробки Visual Studio
3) графічна платформа WPF

4. Перелік питань, що потрібно опрацювати у роботі

1) Аналіз проблеми та огляд існуючих рішень
2) Огляд та аналіз обраних системних інтерфейсів
3) Розробка ПЗ для спрощеного моделювання роботи інтерфейсів КС
4) Опис інтерфейсу та функціоналу створеного додатку
5) Завантаження повного коду програми у відкритий доступ
6) Висновки по роботі

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайд–презентація – 12 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз проблеми та її актуальності. Огляд	27.05.25-30.05.25	
	існуючих аналогів		
2	Огляд та аналіз обраних інтерфейсів	31.05.25-02.06.25	
3	Визначення модельованого функціоналу та	03.06.25-04.06.25	
	компонентів інтерфейсів		
4	Моделювання роботи інтерфейсів	05.06.25-07.06.25	
5	Опис функціоналу та інтерфейсу застосунку	08.06.25-09.06.25	
6	Оформлення матеріалів кваліфікаційної роботи	10.06.25-11.06.25	
7	Подання кваліфікаційної роботи керівникові	12.06.25-13.06.25	
	та її попередній захист		
8	Подання кваліфікаційної роботи на рецензу-	14.06.25-16.06.25	
	вання		

Дата видачі завдання " 26 " травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. Дмитро РОСІНСЬКИЙ
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 88 с., 16 рис., 2 табл., 1 дод., 10 джерел.

СИСТЕМНИЙ ІНТЕРФЕЙС, НАВЧАННЯ, МОДЕЛЮВАННЯ, НАВЧАЛЬНА МОДЕЛЬ, КОМП'ЮТЕРНІ СИСТЕМИ, ФУНКЦІОНАЛ ІНТЕРФЕЙСУ, C#, WPF

Метою кваліфікаційної роботи є розробка засобів моделювання роботи сучасних інтерфейсів комп'ютерних систем для навчання студентів.

У ході виконання кваліфікаційної роботи проведено аналіз проблеми недостатньої кількості інтерактивних засобів для вивчення внутрішніх інтерфейсів комп'ютерних систем. Обґрунтовано актуальність створення навчального програмного забезпечення, яке дозволяє моделювати роботу таких інтерфейсів для студентської аудиторії технічних спеціальностей. Визначено чотири ключові інтерфейси для розгляду: PCI Express, DMI, South та North Bridge. Проведено огляд існуючих аналогічних систем та навчальних рішень, а також виконано глибокий аналіз кожного з обраних інтерфейсів з точки зору архітектури, функціональності та ролі в комп'ютерній системі.

На основі проведеного аналізу визначено основні функціональні компоненти, які доцільно моделювати для кожного інтерфейсу. Розроблено концепцію навчального застосунку з графічним інтерфейсом, реалізованого на базі технологій WPF та C#. В додатку користувач має змогу візуально ознайомитись із структурою обраних інтерфейсів, а також спостерігати за моделюванням процесів передачі даних, ініціалізації зв'язків та обробки запитів.

ABSTRACT

Bachelor’s thesis: 88 pages, 16 figures, 2 tables, 1 appendices, 10 sources.

SYSTEM INTERFACE, LEARNING, MODELING, LEARNING MODEL, COMPUTER SYSTEMS, INTERFACE FUNCTIONALITY, C#, WPF

The major goal of this thesis is to develop tools for modeling the operation of modern computer system interfaces for teaching students.

In order of analysing the problem of insufficient interactive tools for studying the internal interfaces of computer systems was carried out. The relevance of creating educational software that allows modeling the operation of such interfaces for students of technical specialties was substantiated. Four key interfaces were identified for consideration: PCI Express, DMI, South and North Bridge. A review of existing similar systems and educational solutions was conducted, and an in-depth analysis of each of the selected interfaces was performed in terms of architecture, functionality, and role in the computer system.

Based on the analysis, the main functional components that are appropriate to simulate for each interface were identified. A concept for a training application with a graphical interface, implemented on the basis of WPF and C# technologies, was developed. In the application, the user can visually familiarize themselves with the structure of the selected interfaces, as well as observe the modeling of data transfer processes, connection initialization, and request processing.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ПК – персональний комп’ютер.

ЦП – центральний процесор (англ. CPU, Central Processing Unit).

ACK – сигнал, який вказує на успішне отримання даних. Він інформує відправника, що повідомлення було успішно отримано та оброблено (англ. Acknowledge).

ACPI – відкритий промисловий стандарт, який визначає спільний інтерфейс для виявлення апаратного забезпечення, керування живленням та конфігурації материнської плати та пристроїв (англ. Advanced Configuration and Power Interface).

AGP – спеціалізована 32-розрядна системна шина для відеокарти (англ. Accelerated Graphics Port).

CRC – алгоритм обчислення контрольної суми, призначений для перевірки цілісності даних. CRC є практичним додатком завадостійкого кодування, заснованому на певних математичних властивостях циклічного коду (англ. Cyclic Redundancy Check).

DMA – прямий доступ до пам'яті, що дозволяє пристроям обмінюватися даними з оперативною пам'яттю без участі центрального процесора (англ. Direct Memory Access).

DMI – програмний інтерфейс, призначений для збору даних про характеристики комп’ютера (англ. Direct Media Interface).

DRAM – один із видів комп'ютерної пам'яті із довільним доступом (англ. Dynamic Random Access Memory).

FIS – первинні пакети даних, що використовуються для передачі команд, даних та інформації про стан між хостом і пристроєм (англ. Frames Information Structure).

Flash ROM – тип довготривалої комп'ютерної пам'яті, яка може зберігати дані навіть після вимкнення живлення (англ. Flash Read-Only

Memory).

FSB – шина, що забезпечує з'єднання між x86-сумісним центральним процесором і внутрішніми пристроями (англ. Front Side Bus).

GPU – окремий або вбудований пристрій, який відповідає за обробку та виведення графічної інформації на комп'ютері чи іншому пристрої (англ. Graphical Processing Unit).

IDE – паралельний інтерфейс для підключення накопичувачів (жорстких дисків, оптичних приводів) до комп'ютера (англ. Integrated Drive Electronics).

ISA – 8- або 16-розрядна шина введення/виведення IBM PC-сумісних комп'ютерів. Служить для підключення плат розширення стандарту ISA (англ. Industry Standard Architecture).

LPC – шина, використовувана в IBM PC-сумісних персональних комп'ютерах для підключення до центрального процесора пристроїв, що не вимагають великої пропускної здатності (англ. Low Pin Count).

LTSSM – це механізм, який використовується в PCIe для встановлення та підтримки зв'язку між пристроями (англ. Link Training and Status State Machine).

Nak – Сигнал, який вказує на те, що повідомлення не було успішно отримано або було отримано з помилками. Він інформує відправника, що необхідно повторити передачу повідомлення (англ. Negative Acknowledge).

Native Command Queuing – технологія, що використовується в жорстких дисках SATA для підвищення продуктивності.

PCH – сімейство мікросхем, представлених Intel в 2008 році разом з Nehalem (англ. Platform Controller Hub).

PCIe – комп'ютерна шина, що використовує програмну модель шини PCI і високопродуктивний фізичний протокол, заснований на послідовній передачі даних (англ. Peripheral Component Interconnect Express).

Plug–n–play – технологія, призначена для швидкого визначення і конфігурування пристроїв в комп'ютері та інших технічних пристроях.

RAID – технологія віртуалізації даних, яка об'єднує кілька дисків в логічний елемент для надійності збереження інформації та підвищення продуктивності накопичувачів (англ. Redundant Array of Independent Disks).

SSD – комп'ютерний запам'ятовувальний пристрій на основі мікросхем пам'яті та контролера керування ними, що не містить рухомих механічних частин (англ. Solid-State Drive).

Super I/O – клас інтегральних мікросхем, що використовуються на материнських платах комп'ютерів для управління низькошвидкісними периферійними пристроями (англ. Super Input/Output).

ВСТУП

У процесі вивчення будови та архітектури комп'ютерів студенти часто стикаються з проблемою розуміння того, як працюють внутрішні інтерфейси комп'ютерних систем. Наявні засоби навчання в цій галузі є застарілими, надто спрощеними або недостатньо інтерактивними, щоб задовольнити потреби сучасного навчання.

Очевидно, що потрібні спеціальні навчальні засоби, які дозволять студентам вивчати компоненти системного рівня. Такі інструменти повинні поєднувати теоретичні дані з інтерактивною візуалізацією, щоб сприяти розумінню та залученню.

Рішенням цієї проблеми може стати розгляд та поєднання можливостей існуючих засобів або ж створення власного, що б демонструвало, як працюють внутрішні інтерфейси комп'ютера. Отримане рішення може містити інтерактивні діаграми, пояснення у вигляді фреймів та графічні симуляції відповідно до потреб студентів.

Запропонований проєкт заповнить прогалину між теорією апаратного забезпечення та практичним розумінням шляхом моделювання вибраних системних інтерфейсів. Проєкт буде одночасно і дослідницьким внеском, і прототипом концепції для майбутнього освітнього програмного забезпечення.

Насамкінець, проєктування та дослідження такого роду систем є дуже актуальним для студентів комп'ютерної інженерії. Це не тільки ілюструє складну архітектуру, але й підвищує рівень технічної освіти в цій галузі в цілому.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка проблеми та її актуальність

Стрімкий розвиток цифрових технологій та складність сучасних обчислювальних систем радикально підвищили вимоги до якості технічної освіти. У комп'ютерній інженерії розуміння структури та роботи внутрішніх системних інтерфейсів, таких як PCI Express, DMI, концентратори чипсетів та контролери переривань, є першочерговим для отримання низькорівневого розуміння того, як функціонують комп'ютери. Однак спеціалізованих освітніх ресурсів, які дозволяють учням інтерактивно вивчати ці інтерфейси, дуже мало. Більшість існуючих рішень є або застарілими, або спрощеними, або спрямованими на інші рівні абстракції, такі як програмування чи мережеві конфігурації.

1.1.1 Проблеми, з якими стикаються студенти при вивченні архітектури комп'ютерів

У більшості навчальних програм теми системних інтерфейсів спочатку вивчаються в абстрактному вигляді, можливо, доповнені статичними схемами або описами з підручників. Хоча це створює концептуальну основу, це не завжди забезпечує інтуїтивне відчуття того, як компоненти взаємодіють у реальному часі. Брак візуалізації та інтерактивності може ускладнити розуміння учнями того, як у комп'ютері проходять потоки даних і керуючої інформації, як генеруються переривання і визначаються їхні пріоритети, або як центральний процесор взаємодіє з периферійними пристроями через різні шини і контролери. Без інструментів, які демонструють динамічну поведінку, студенти можуть в кінцевому підсумку запам'ятати теорію, але не розвинути розуміння апаратних систем.

Відсутність доступного, сучасного і спеціально орієнтованого навчального програмного забезпечення для моделювання інтерфейсів системного рівня є серйозною проблемою для студентів. Існуючі інструменти мають тенденцію зосереджуватися або на вищих рівнях абстракції, таких як мови програмування чи операційні системи, або на вузьких технічних спеціалізаціях як, наприклад, моделювання мереж, програмування мікроконтролерів, тощо. Існує явний дефіцит засобів, які ілюструють та пояснюють роботу внутрішніх шин, контролерів та інтерфейсів – областей, які є надзвичайно важливими для розуміння архітектурного дизайну сучасних комп'ютерів.

1.1.2 Важливість інтерактивності та візуалізації в освіті

Для того, щоб вирішити цю проблему, необхідно розробити або концептуально спроєктувати освітню програму, яка спеціалізується саме на системних інтерфейсах в комп'ютері. Відповідна програма повинна надавати студентам поєднання пояснювальної теорії, інтерактивних діаграм та імітаційного моделювання, яке графічно демонструє, як взаємодіють різні підсистеми [1].

Такий підхід представляє особливий інтерес для студентів, оскільки він допомагає подолати прірву між теорією і практикою – можливо, найбільш важливий аспект інженерної освіти. По–друге, інтерактивне програмне забезпечення співзвучне сучасній педагогічній теорії, яка вимагає активного навчання та візуального мислення. Завдяки більшій інтерактивності та доступності, таке програмне забезпечення може допомогти студентам подолати концептуальні бар'єри та розвинути більш міцний фундамент у проєктуванні систем на апаратному рівні.

1.2 Характеристика та класифікація проблеми

Описана вище проблема належить до широкої категорії навчально–методичних проблем технічних наук. Більш конкретно її можна описати як недостатню кількість навчальних матеріалів на практичному та графічному рівні при вивченні апаратного інтерфейсу на системному рівні. Проблема полягає не у відсутності теоретичних матеріалів – підручники, лекції та наукові статті з архітектури комп'ютерів широко доступні – а у складності перекладу цієї теорії в інтуїтивне розуміння та практичний досвід. Таким чином, це і педагогічна, і технологічна проблема, яка потребує вирішення питання про те, як поширювати знання і те, за допомогою яких засобів розширювати це робити.

З точки зору класифікації, ця проблема знаходиться на межі розробки освітнього програмного забезпечення, моделювання комп'ютерних систем та дизайну інтерактивного навчання. Це питання, зачіпає як учнів, так і викладачів. Студентам потрібні інструменти для експериментів і вивчення таких понять, як шина, потік даних чи управління перериваннями. У той же час, викладачам не вистачає сучасних рішень, які можна було б використовувати як віртуальні посібники на лекціях, практичних чи лабораторних заняттях. Це свідчить про наявну прогалину – сучасні методи викладання не передбачають використання симуляції, взаємодії чи візуального зворотного зв'язку, що є життєво важливими в інженерній освіті.

Проблема також відображає елементи асиметрії у здобутті знань. У той час як високорівневе програмування, мережеві технології та дизайн програмного забезпечення часто підтримуються надійними середовищами розробки та тренажерами, апаратні інтерфейси системного рівня опановуються за допомогою статичних підручників або абстрактних моделей. Це створює деякий дисбаланс: студенти почуваються комфортніше із завданнями на програмному рівні, але залишаються невпевненими чи відстороненими, коли мають справу з архітектурою апаратного рівня.

Зрештою, це може призвести до зниження рівня знань з таких тем, як вбудовані системи, інтеграція обладнання чи оптимізація продуктивності.

Технологічна застарілість є ще однією проблемою в цій ситуації. Деякі з існуючих інструментів та симуляторів, які спочатку були орієнтовані на низькорівневе проєктування, не оновлювалися і не підтримувалися протягом останніх кількох років. Всі вони або вже не здатні підтримувати сучасні апаратні стандарти, такі як PCI Express, DMI або розширену маршрутизацію переривань за допомогою APIC, або роблять це неповноцінно. Це робить їх менш актуальними для навчання сучасним архітектурам і зменшує їхню корисність у допомозі студентам працювати з реальними системами.

Цільова аудиторія для таких навчальних інструментів може розширитися і включати студентів університетів, студентів технічних коледжів, індивідуальних учнів та молодших спеціалістів у навчальних програмах. Тому характер і функціонування навчального програмного забезпечення для цієї групи має бути гнучким і пристосованим до мінливих рівнів знань і контекстів навчання. Багаторівневий дизайн – з простими візуальними представленнями, а також функціональними моделями вищого рівня – може забезпечити більший діапазон сценаріїв використання та типів користувачів.

1.3 Огляд існуючих систем навчання

Розширення комп'ютерної освіти призвело до появи широкого спектру методів навчання, придатних для різних типів учнів і навчальних цілей. У сфері технічної та інженерної освіти, особливо в галузі інформатики та електроніки, ці стратегії часто ґрунтуються на інтерактивності, візуалізації та проблемно – орієнтованому навчанні. Теоретичний зміст все частіше доповнюється динамічними презентаціями та симуляціями, за допомогою яких учні можуть взаємодіяти з абстрактним матеріалом у більш експериментальний спосіб.

Усі описані нижче методи надають перевагу активному залученню та індивідуальному темпу навчання, а не пасивному споживанню контенту. Вони зміщують фокус технічної освіти назад до експериментів і дій, що особливо корисно при вивченні складних предметів, таких як архітектура комп'ютера і внутрішні системні інтерфейси. Завдяки використанню інтерактивності, зворотного зв'язку та візуалізації ці навчальні системи допомагають зменшити когнітивне навантаження, яке зазвичай пов'язане з абстрактними або інформаційно насиченими темами.

1.3.1 Навчання, що базується на іграх та гейміфікація

Серед фаворитів – ігрове навчання, коли навчальний матеріал подається у вигляді гри або інтерактивного сценарію. Цей метод підвищує мотивацію, заохочує до спроб і помилок і може включати рівні, цілі або системи зворотного зв'язку. Ігрове навчання з технічних дисциплін використовується для імітації таких видів діяльності, як написання коду, вирішення проблем або створення систем, і винагороджується за правильний вибір, заохочуючи при цьому критичне мислення.

1.3.2 Моделювання та віртуальні середовища

Іншим поширеним методом є використання віртуальних лабораторій, які відтворюють функціональність реальних лабораторій у програмному середовищі. Ці системи дозволяють учням проводити експерименти або технічні процедури без доступу до реального обладнання. Для інженерних дисциплін віртуальні лабораторії регулярно використовуються для моделювання електроніки, роботи ланцюгів або функцій процесора, даючи учням можливість спостерігати за змінами в роботі системи на основі введених користувачем даних або конфігурації.

1.3.3 Інтерактивні діаграми та моделювання

Діаграмне навчання та інтерактивне моделювання – ще один ефективний метод опису поведінки складних систем. Він використовує графічні схеми та анімацію, які пояснюють, як частини системи взаємодіють одна з одною в часі. При використанні в комп'ютерних системах ці моделі можуть відображати потік даних, доступ до пам'яті, зв'язок по шині або обробку переривань. Маніпулюючи інтерактивними моделями, учні можуть досліджувати причинно – наслідкові зв'язки та поглиблювати свої знання про архітектурні залежності.

1.3.4 Візуальна розповідь та покрокове навчання

Візуальне розповідання історій та навчання за сценаріями також набувають популярності. Цей підхід використовує поетапний процес через діяльність, що іноді відображає реальні сценарії, щоб провести учнів через навчальні потреби. Кожен крок випливає з попереднього, іноді з використанням візуальних пояснень, запитань для вирішення проблем і контрольних точок. Це особливо корисно при поясненні учням поведінки системи або кроків конфігурації.

1.3.5 Навчання на основі коду зі зворотним зв'язком

У середовищах, орієнтованих на програмування, поширеним є навчання на основі коду та негайного зворотного зв'язку. Цей підхід дозволяє учням вводити код, випробовувати його та експериментувати з налагодженням коду в навчальному середовищі без необхідності виходити з нього, отримуючи миттєві підказки чи описи помилок. Хоча це більш поширено на просунутих рівнях програмування, реакція в реальному часі на введення даних студентами також може застосовуватися для симуляції на

апаратному рівні та системного моделювання.

1.3.6 Модульний контент з адаптивним розвитком

Іншою сильною концепцією є адаптивне і модульне подання контенту, яке структурує навчання на окремі чітко визначені блоки або модулі. Модулі можуть поступово розкриватися на основі успішності або виконання студентом попередніх модулів. Така гнучкість дозволяє студентам навчатися у власному темпі, переглядати проблемні місця або пропускати зайвий матеріал відповідно до їхнього рівня підготовки.

1.4 Приклади та опис існуючих інтерфейсів

Для підтримки навчання в комп'ютерній інженерії та суміжних галузях розроблено різноманітні освітні програми та симулятори. Ці інструменти використовують різні навчальні підходи для представлення технічного контенту, починаючи від симуляцій і візуального моделювання і закінчуючи інтерактивними сценаріями. Хоча багато з них є високоефективними у своїх галузях, відносно небагато з них зосереджені на інтерфейсах системного рівня, таких як внутрішні шини, зв'язок з чипсетом та обробка переривань. Нижче наведено кілька прикладів інструментів, які відображають описані вище методи викладання та демонструють можливі напрямки подальшого розвитку.

Усі наведені нижче приклади відображають широкий спектр освітніх стратегій та реалізацій. Деякі з них зосереджені на моделюванні системи, інші – на програмуванні чи поведінці обладнання. У всіх випадках включення візуальних елементів і взаємодія з користувачем є ключовим для перетворення абстрактних понять на доступний навчальний досвід. Інструменти, які пропонують цільовий контент на внутрішніх комп'ютерних інтерфейсах, таких як програма для ПК, особливо актуальні для усунення

прогалин, визначених у попередніх розділах.

1.4.1 Cisco Pakcet Tracer

Одним з найкраще розроблених інструментів у сфері технічного навчання є Cisco Packet Tracer – мережева симуляція, яка дозволяє користувачам створювати та перевіряти віртуальну мережеву інфраструктуру [2]. Він має інтуїтивно зрозумілий інтерфейс (рисунок 1.1), за допомогою якого студенти можуть підключати пристрої, налаштовувати параметри та спостерігати за поведінкою трафіку між вузлами.

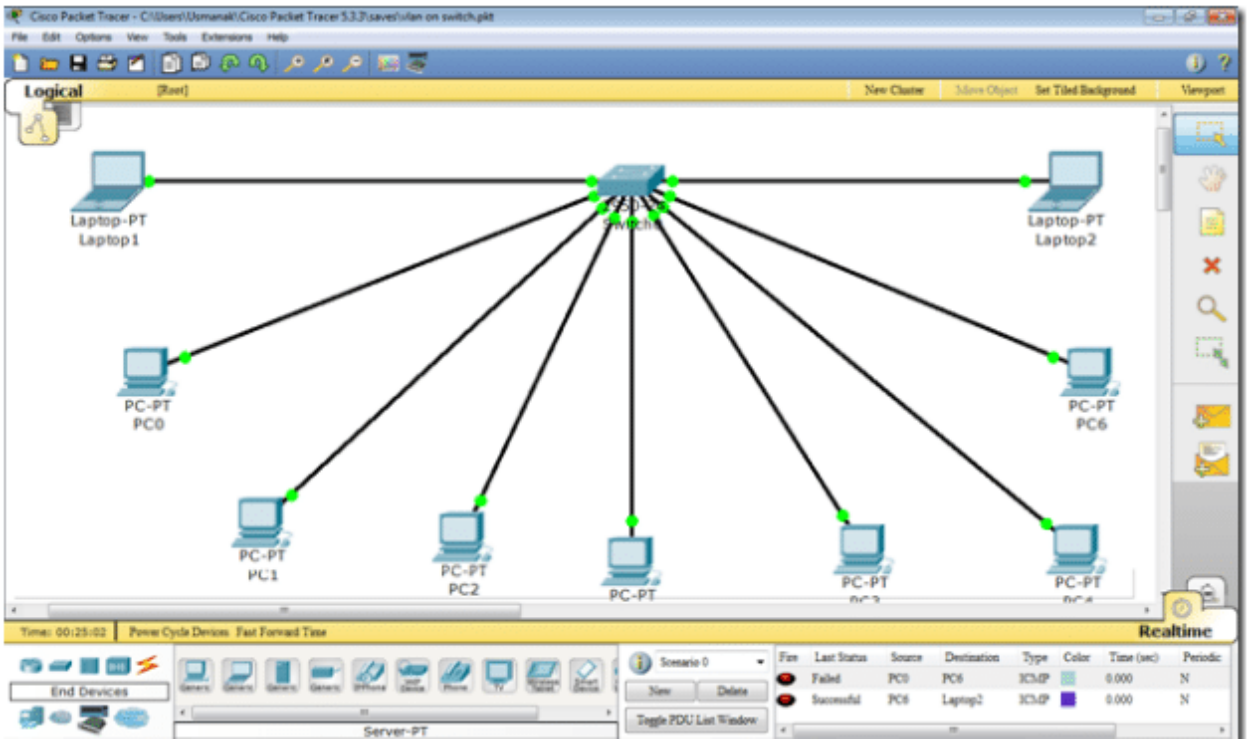


Рисунок 1.1 – Інтерфейс застосунку Cisco Packet Tracer

Платформа підтримує конфігурацію командного рядка і коментарі в реальному часі, що заохочує дослідження роботи. Хоча її основною метою є мережева комунікація, а не внутрішня IT – архітектура, вона є ефективним прикладом інтуїтивно зрозумілої симуляції, яка може покращити розуміння складних систем.

1.4.2 CodeCombat та MakeCode

У сфері інтерактивного навчання програмуванню такі платформи, як Codecombat і Microsoft MakeCode (рисунок 1.2), забезпечують середовище, в якому студенти вивчають синтаксис, логіку і структуру управління завдяки завданням, інтегрованим у сценарій.

а) б)

Рисунок 1.2 – Інтерфейси навчальних платформ, що використовують медод «гейміфікації» навчання: а) CodeCombat; б) Microsoft MakeCode

Ці інструменти часто поєднують програмування на основі блоків і документів на основі негайного зворотного зв'язку та моніторингу процесу. Хоча вони не призначені для фізичного виховання, вони успішно продемонстрували, що взаємодія та ігри можуть підвищити зацікавленість та мотивацію учнів.

1.4.3 VisUAL та EduCPU

Інший тип інструментів включає симулятори збірки та спрощені моделі процесорів. Таким прикладом є Visual, візуальна збірка ARM (рисунок 1.3), яка дозволяє студентам писати, переглядати та налагоджувати низькорівневий код. Інтерфейс зазвичай включає режими реєстрації, карти пам'яті та контрольні схеми.

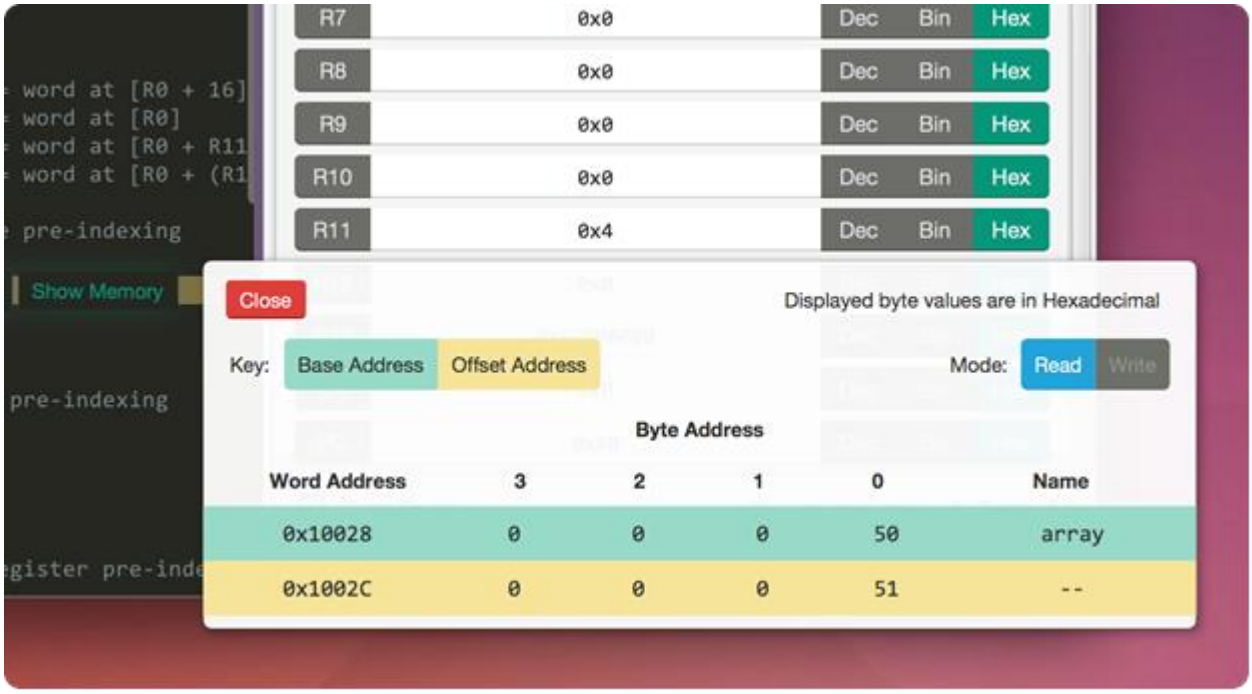


Рисунок 1.3 – Інтерфейс застосунку VisUAL ARM

Іншим прикладом є EducationCPU, більш абстрактна, адже не має графічного інтерфейсу симуляція процесора, яка часто використовується для ознайомлення студентів з процесом обробки та логікою роботи процесора. Ці інструменти корисні для розуміння роботи внутрішнього процесора, але часто обмежуються ізольованою поведінкою і не забезпечують візуалізацію повного системного зв'язку, наприклад, передачу даних через системні шини або переривання через контролери.

1.4.4 Logisim

Середовище моделювання схем є ще одним прикладом візуальної освіти. Такий інструмент, як logisim (рисунок 1.4), дозволяє студентам створювати логічні схеми з віртуальними компонентами, такими як логічні двері та трансплантація каналів.

Користувачі можуть моделювати поведінку схеми в реальному часі і спостерігати за зміною вхідних і вихідних сигналів у відповідь на різні

умови. Хоча цей метод зосереджений на цифровій електроніці, його можна налаштувати для відображення архітектурних блоків вищого рівня та внутрішніх маршрутів передачі даних.

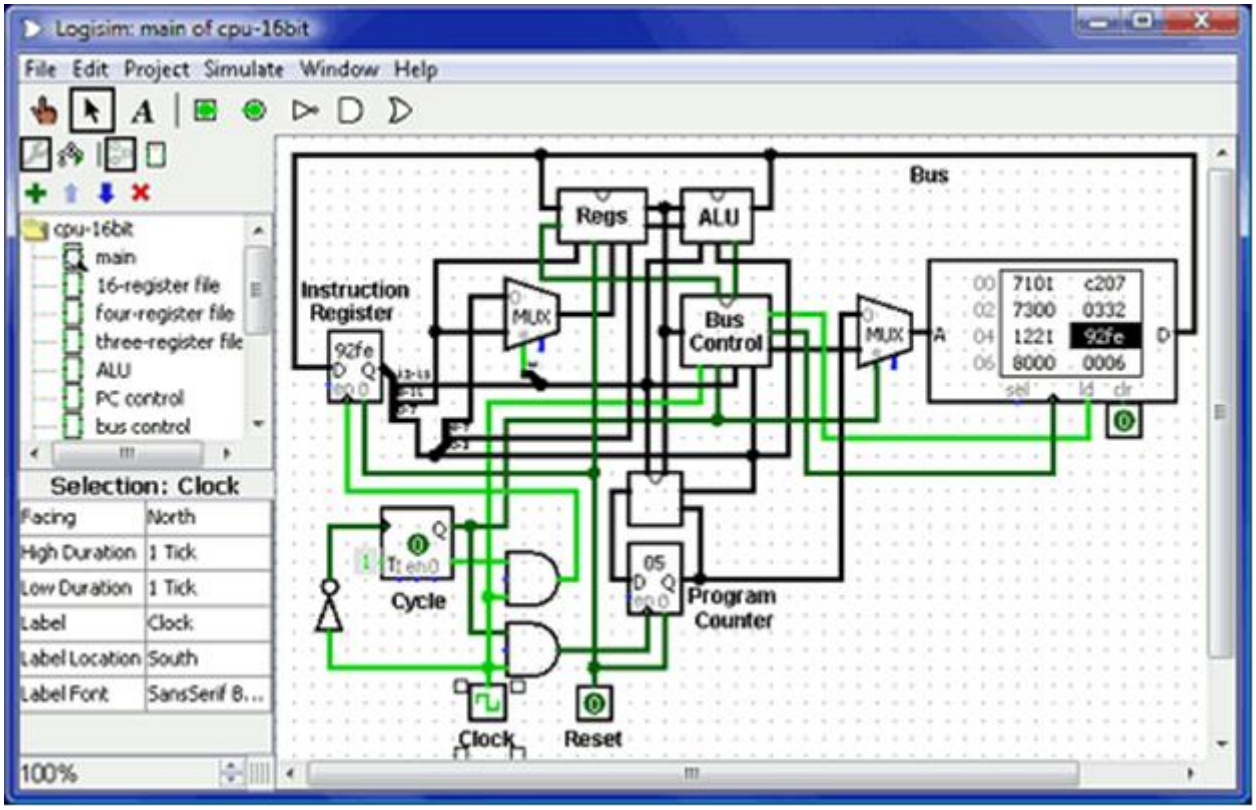
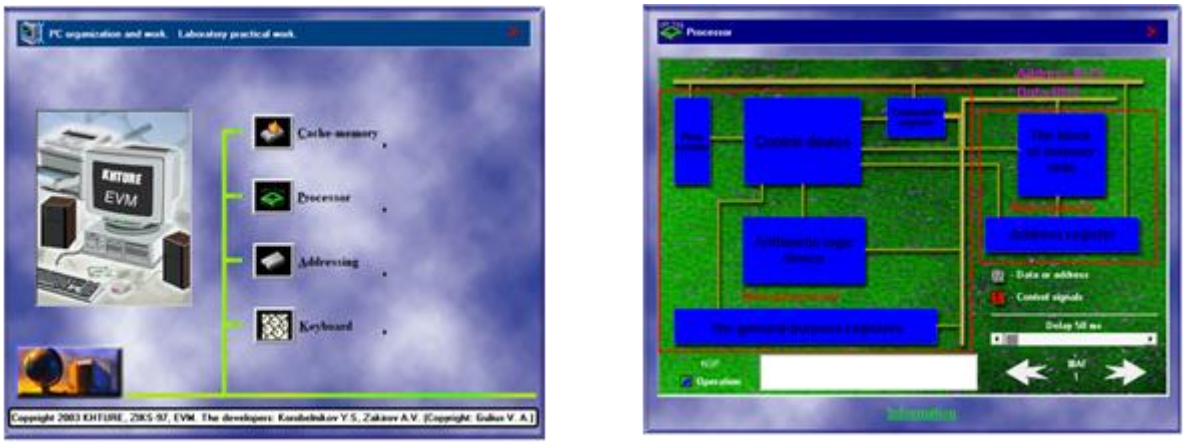


Рисунок 1.4 – Інтерфейс та функціонал застосунку Logisim

1.4.5 Застосунок, виконаний студентами ХНУРЕ (PC)

На додаток до загальнодоступних інструментів, деякі спеціально розроблені освітні програми призначені для висвітлення конкретних тем. Одним з таких прикладів є застосунок для Windows під назвою PC, що був розроблений студентами Харківського національного університету радіоелектроніки з метою вивчення внутрішніх компонентів системи за допомогою інтерактивних елементів у рамках курсу «Архітектура комп’ютерів». Ця програма містить модулі, які представляють частини комп'ютерної системи, такі як процесор, пам'ять і периферійні інтерфейси. Кожен компонент включає інформаційний розділ та розділ моделювання, що дозволяє користувачам вивчати як теорію, так і роботу частини системи.

Інтерфейс комп'ютерної програми (рисунок 1.5) базується на графічному макеті з кнопками, що ведуть до підмодулів та візуальних зображень. Наприклад, у розділі адресації або процесора користувач може отримати доступ до інформації про операнди, типи адресації та операції процесора. Деякі компоненти містять фонові зображення з інтерактивними зонами, тоді як інші надають текстові пояснення або файли підказок, щоб провести користувача через функціональну логіку. Дизайн відображає принципи модульного навчання, візуального моделювання та покрокового вивчення. Хоча програма є відносно компактною, вона демонструє можливість створення тематичних інструментів, які підтримують глибшу взаємодію з архітектурою системи.



а) б)

Рисунок 1.5 – Інтерфейс програми PC: а) Головна сторінка; б) Моделювання роботи ЦП

1.5 Основні ідеї, викладені в сучасних інформаційних джерелах

Дослідження того, як студенти взаємодіють з цифровими освітніми системами, розвивалися паралельно з розвитком педагогічного дизайну та інтерфейсу. Сучасні документи в цій галузі підкреслюють, що успіх навчального середовища визначається не лише інформацією, яку воно містить, але й структурою, тобто структурованістю, інтуїтивною

зрозумілістю та поширенням цієї інформації через користувацький інтерфейс. У технічних галузях – особливо в таких сферах, як комп’ютерна архітектура – це занепокоєння стає ще більш вираженим, оскільки від студентів часто вимагається розуміння складних абстрактних процесів без переваги безпосередньої фізичної взаємодії.

Періодично в джерелах зустрічається думка про те, що навчання є найбільш ефективним, коли модель системи візуально показана і розділена на частини. Це особливо стосується матеріальних понять і системних рівнів. Як зазначається в одному з джерел, когнітивне перевантаження може бути зменшене, якщо навчальний інтерфейс задовольняється непомітними і важливими одиницями, направляючи учнів через прогресуючу складність [3]. Це узгоджується з необхідністю структурованого і нагромадженого підходу в навчальному програмному забезпеченні для моделювання інтерфейсів комп'ютерної системи.

Інший аспект, що обговорюється, стосується переважно важливості дизайну користувацького інтерфейсу. Дослідники вважають, що навчальні платформи повинні бути адаптовані відповідно до очікувань і моделей студентів. У дослідженні «розумного» користувацького інтерфейсу автори стверджують, що освітнє середовище адаптується до особистого процесу, а стиль навчання значно покращує прихильність і підтримку. Цей принцип адаптації особливо стосується технічної освіти, куди студенти приходять з різним базовим рівнем знань. Забезпечення гнучкого інтерфейсу – детального і пояснювального – дозволяє новачкам і користувачам покращити свої переваги від використання одного і того ж інструменту.

У літературі також обговорюється роль взаємодії у сприянні глибшому навчанню. Системи дозволяють студентам досліджувати, моделювати або маніпулювати компонентами для підтримки того, що часто називають «позитивною обізнаністю». Публікаційна примітка: учні повинні не тільки спостерігати за моделлю, але й взаємодіяти з нею – відповідно до вибору, введення або гіпотези – щоб побудувати значуще розуміння. Цей принцип

об'єднує багато інструментів, заснованих на моделюванні, і є головним аргументом для поєднання діаграм і взаємодії з додатками, розробленими для цієї тези.

В останніх дослідженнях було виявлено, що інтеграція моніторингу погляду, адаптивного ШІ та надання динамічного контенту є засобами для покращення коментарів та підтримки уваги користувачів. Наприклад, у статті про українську мову пропонується система коригування потоку контенту на основі інтуїтивного фокусу та візуального розкладу студента, стверджується, що моніторинг управління поглядом у реальному часі дозволяє освітнім системам природно реагувати на ритм і фокус уваги учнів [4]. Хоча такі технології ще тільки з'являються, концепція інтерфейсу реакції застосовується навіть у простішій формі – наприклад, коригування контенту на основі результатів навігаційних шляхів головоломки або користувачів.

У деяких статтях і книгах також піднімаються питання, пов'язані з розривом між теоретичною освітою і практичним розумінням, особливо в таких галузях, як ІТ – інженерія. Деякі автори вказують на те, що студенти можуть дізнатися більше про шини даних, сигнали переривань та архітектуру системи з конференцій та інструкцій з використання, але немає візуальної або динамічної підтримки, ці теми все ще далекі з точки зору концепцій. Ця прогалина потребує більше статичних діаграм – потрібно, щоб системи відображали реальну поведінку, потоки даних та інтерактивні моделі.

Дизайн інтерфейсу в освітніх інструментах завжди описується не лише як технічний виклик, але й як освітній. Добре розроблений інтерфейс, згідно з дослідженням, повинен бути достатньо прозорим, щоб при необхідності зникнути, але повністю структурованим, щоб спрямовувати студентів на правильний шлях. Цей парадокс – між баченням і простотою – визначає мистецтво дизайну освітнього інтерфейсу. При моделюванні чогось складного, як, наприклад, навчальної комп'ютерної системи, цей баланс стає особливо важливим, щоб уникнути поглинання користувачів.

Саме тому міністерство освіти підтримує запропоновані основні

дизайнерські рішення проєкту. Вони підкреслюють ясність, адаптивність, мозаїчність та взаємодію – все, що є базовим для створення інструменту, який є чимось більшим, ніж сучасна теорія. Натомість він уможливлює візуальне дослідження, пояснення та особисту траєкторію навчання, засновану на найкращих освітніх практиках та науці про інтерфейси.

1.6 Висновки аналізу

Проведений аналіз підкреслює складність проблеми, пов'язаної з викладанням інтерфейсів у внутрішній комп'ютерній системі. Хоча теоретична платформа комп'ютерної архітектури та матеріальних систем добре показана в академічних програмах, методи, що використовуються для надання цих знань, часто не мають візуальних та інтерактивних компонентів. Це особливо важливо при вивченні поведінки інтерфейсів на системному рівні, що важко уявити за допомогою традиційних засобів.

Існуючі освітні інструменти пропонують цінний досвід в таких областях, як програмування, мережеве з'єднання і цифрова логіка, але відносно мало спеціальної адаптації до комунікації внутрішніх компонентів. Існуючі методи часто зосереджені на ізольованих концепціях або використовують застарілі уявлення, які більше не відповідають сучасним матеріальним стандартам. Розгляд існуючих систем і методів показує, що візуалізація, взаємодія інструкцій і прогресуюче ускладнення є одним з найбільш ефективних методів в технічному навчальному середовищі.

Література підтримує ідею, що концепція освітнього інтерфейсу повинна базуватися на переконанні та педагогіці. Адаптивна структура навчання, безпосереднє інформування користувача та сегмент взаємодії підкреслюються як основні елементи систем для підтримки навчання та підтримки знань, отриманих під час самонавчання. Інтеграція цих принципів в контексті комп'ютерно – технічного навчання все ще обмежена, що створює можливості для розробки професійного і цілеспрямованого рішення.

Результати, описані в цьому розділі, створюють міцну основу для подальшої роботи. Наступним кроком дослідження буде розробка цілей і вимог до прототипу освітнього додатку, визначення його структури і функціональних компонентів, а також пропозиція концепції інтерфейсу, призначеного для підтримки досліджень внутрішніх взаємодій комп'ютерної системи. Ця концептуальна модель базуватиметься як на викликах, визначених у процесі аналізу, так і на методах, які довели свою ефективність у споріднених навчальних системах.

2 ОГЛЯД ТА АНАЛІЗ ОБРАНИХ ІНТЕРФЕЙСІВ

2.1 Інтерфейс PCI Express

PCI Express (PCIe) є одним з найважливіших внутрішніх інтерфейсів, що використовуються в сучасних IT – системах. Як висококласний стандарт шини, PCIe замінив успадковані шини PCI та AGP, забезпечивши вищу швидкість передачі даних, розширення та гнучкість. Він функціонує на основі топології «точка – точка», де кожен пристрій має пряме з'єднання з кореневим комплексом або комутатором, що усуває конфлікти шини та дозволяє здійснювати паралельну передачу даних.

З архітектурної точки зору, PCIe базується на класовій моделі, розділяючи функції на три основні рівні: транзакційний рівень, рівень передачі даних і фізичний рівень. Така структура забезпечує чіткіше розподілення обов'язків та покращену модульність. Транзакційний клас відповідає за створення і обробку вимог, таких як читання пам'яті або елемента. Рівень передачі даних забезпечує надійний розподіл пакетів за допомогою таких механізмів, як CRC і протоколів ACK / Nak [5]. Нарешті, фізичний рівень передачі даних шифрується на різних парах, відповідаючи за еволюційні конфігурації від x1 до x16.

Однією з головних особливостей PCIe є його протокол зв'язку на основі пакетів, який є значно надійнішим, ніж старі паралельні шини. Кожна транзакція, незалежно від типу, інкапсульована у формат пакета, який чітко визначений. Ця узгодженість не тільки покращує цілісність даних, але й спрощує управління різними типами транзакцій на деяких класах. На відміну від шини PCI, яка дозволяла необмежені передачі, PCIe вимагає пакетів фіксованого розміру, які легше перевіряти та відстежувати [6].

Інтерфейс також поєднує механізм управління потоком на основі

кредитів. Кожен пристрій передачі даних повинен отримати кредит на початку, перш ніж розпочати транзакцію. Це запобігає переповненню буфера і дозволяє динамічно і справедливо розподіляти ресурси між декількома віртуальними каналами. Ці віртуальні канали, в свою чергу, підтримують концепцію якості обслуговування, що дозволяє встановлювати пріоритети трафіку даних, що є необхідним в системах, які працюють з чутливими до часу даними, такими як аудіо – або відеопотоки [5].

Ініціалізація каналу PCIe здійснюється за допомогою Link Training and Status State Machine, яка забезпечує узгодження обома кінцевими точками таких параметрів, як ширина каналу, швидкість та інверсія полярності. Після завершення синхронізації та переходу каналу в стан L0 можна розпочинати звичайну передачу даних. Фізичний рівень також може обробляти такі функції, як зміна напрямку руху та стани управління живленням (L0s, L1, L2, L3), що підвищує сумісність та зменшує споживання енергії.

З точки зору системного інтегратора, модульна природа PCIe забезпечує значну гнучкість. Такі компоненти, як графічні процесори, SSD–накопичувачі та мережеві карти, можуть використовувати один і той же стандартний фізичний та електричний інтерфейс, а програмне забезпечення здійснює виявлення та перелік за допомогою регістрів конфігураційного простору. Ця модель plug–and–play спрощує складання та розширення системи [5].

Окрім основних команд читання та запису, PCIe підтримує переривання з повідомленням та різні системні контрольні повідомлення, що забезпечують точну та ефективну координацію між апаратними компонентами. Крім того, архітектура PCIe розроблена з урахуванням сумісності з майбутніми версіями [7]. Майбутні версії з більшою пропускною здатністю (наприклад, PCIe 5.0 та 6.0) часто можуть підтримуватися за допомогою оновлень на фізичному рівні без значної переробки логіки верхнього рівня.

Таким чином, дизайн PCIe є прикладом сучасної архітектури

інтерфейсу, адже, він є модульним та масштабованим, а також здатен розвиватися разом з вимогами до апаратного забезпечення. Для цілей навчання студентів PCIe пропонує комплексний приклад сучасного високошвидкісного інтерфейсу, де багаторівневість протоколу, цілісність даних, арбітраж ресурсів і динамічна конфігурація системи є критично важливими оперативними функціями.

2.2 Direct Media Interface (DMI)

Direct Media Interface (DMI) – це важлива технологія міжсистемного з'єднання в системах на базі Intel, яка служить високошвидкісним каналом зв'язку між чіпсетом (часто званим Platform Controller Hub або PCH) і процесором. Впроваджена в рамках еволюції архітектури чіпсетів Intel, DMI полегшує комунікацію між процесором і різними периферійними пристроями, підключеними до PCH, такими як пристрої зберігання даних, USB–контролери та мережеві інтерфейси. На відміну від інтерфейсу Peripheral Component Interconnect Express (PCIe), який безпосередньо з'єднує пристрої з високою пропускною здатністю, такі як графічні процесори, з центральним процесором, DMI діє як вузьке місце для пристроїв, підключених до чіпсету, впливаючи на загальну продуктивність системи в додатках з інтенсивним введенням – виведенням.

DMI працює як послідовний інтерфейс «точка – точка», використовуючи протокол, схожий на PCIe, але пристосований для зв'язку між чіпсетом і центральним процесором. У своїй початковій версії, DMI 1.0, інтерфейс забезпечував пропускну здатність приблизно 1 ГБ/с в кожному напрямку, використовуючи чотириканальну (x4) конфігурацію зі швидкістю передачі даних 2,5 ГТ/с, подібну до PCIe 1.0 [8]. Наступні версії значно покращили цю пропускну здатність: DMI 2.0 подвоїв швидкість передачі до 5,0 ГТ/с, пропонуючи до 2 ГБ/с в кожному напрямку, а DMI 3.0, представлений з чіпсетами Intel серії 100, ще більше збільшив пропускну

здатність до 3,94 ГБ/с в кожному напрямку, використовуючи ту ж чотириканальну конфігурацію, але зі швидкістю 8,0 ГТ/с, що відповідає швидкості PCIe 3.0 [8]. Ця еволюція відображає зростаючий попит на вищу пропускну здатність вводу – виводу, особливо з появою високопродуктивних рішень для зберігання даних, таких як SSD NVMe, які можуть легко наситити пропускну здатність DMI.

У контексті систем глибокого навчання обмеження DMI стають особливо очевидними. Як підкреслювалося в попередніх дослідженнях, вузьке місце в передачі даних між пристроями зберігання даних і CPU/GPU часто виникає через обмежену пропускну здатність DMI, незважаючи на прогрес у продуктивності SSD. Наприклад, у дослідженні з перенесення навчання з обробкою в сховищі зазначається, що навіть з високопродуктивними SSD NVMe, здатними до послідовного читання зі швидкістю до 3500 МБ/с, сукупна пропускна здатність в системах Intel обмежена 3,94 ГБ/с через обмеження DMI 3.0. Це вузьке місце призводить до недовикористання ресурсів ЦП, оскільки завантаження даних не може встигати за обчислювальними вимогами. Це явище спостерігається в системах, що до прикладу використовують GPU NVIDIA RTX 2080 Ti, де загальна пропускна здатність була обмежена 6200 зображеннями на секунду, незважаючи на здатність GPU обробляти понад 30000 зображень на секунду [8].

Архітектурні наслідки DMI є значними для проєктування систем. У типовій системі Intel графічні процесори та інші пристрої з високою пропускною здатністю підключаються безпосередньо до центрального процесора через слоти PCIe, отримуючи переваги від більш високих швидкостей передачі даних і масштабованих конфігурацій ліній (наприклад, x16 для графічних процесорів). На відміну від цього, пристрої зберігання даних, такі як SATA або M.2 SSD, підключаються до PCH, покладаючись на DMI для передачі даних до центрального процесора для подальшої обробки. Ця ієрархічна структура вводить обмеження затримки та пропускної

здатності, особливо коли кілька SSD об'єднані в конфігурації RAID–0, оскільки їхня сукупна пропускна здатність часто перевищує пропускну здатність DMI. Наприклад, вісім SSD Samsung 860 EVO, кожен з яких здатний передавати 550 МБ/с, теоретично можуть досягти 4,4 ГБ/с, але DMI 3.0 обмежує цю швидкість до 3,94 ГБ/с, що призводить до вузького місця в продуктивності.

Щоб пом'якшити обмеження DMI, були досліджені альтернативні підходи, такі як підключення SSD–накопичувачів NVMe безпосередньо до слотів PCIe для обходу PCH. Однак це рішення є неоптимальним, оскільки слоти PCIe зазвичай зарезервовані для високопродуктивних пристроїв, таких як графічні процесори або карти InfiniBand, і їх виділення для SSD–накопичувачів може погіршити продуктивність системи для обчислювально–інтенсивних завдань. Цей компроміс підкреслює необхідність інноваційних рішень, таких як обробка в сховищі, що переносить завдання з інтенсивним використанням даних на сам пристрій зберігання, зменшуючи обсяг даних, що передаються через DMI.

Розуміння ролі DMI також впливає на проєктування майбутніх систем, спрямованих на збалансування обчислювальної потужності та продуктивності вводу – виводу. У міру подальшого вдосконалення продуктивності графічних процесорів – про що свідчить десятикратне підвищення продуктивності графічних процесорів NVIDIA для настільних комп'ютерів за десятиліття – пропускна здатність DMI повинна відповідно масштабуватися, щоб запобігти виникненню постійного вузького місця. Майбутні ітерації DMI або альтернативні технології міжсистемних з'єднань можуть потребувати використання ширших конфігурацій ліній або вищих швидкостей передачі даних, щоб задовольнити зростаючі вимоги до вводу–виводу вимог глибокого навчання та інших додатків, що інтенсивно використовують дані. Для цілей цього моделювання, моделювання DMI як обмеженого зв'язку між PCH і CPU може надати цінну інформацію про його вплив на продуктивність системи.

2.3 Південний міст (South Bridge)

Південний міст, важливий компонент архітектури сучасних IT – систем, виступає в ролі основного інтерфейсу та відповідає за управління низкою функцій вводу і виводу , необхідних для роботи периферійних пристроїв. Його основне призначення – виступати посередником між центральним процесором (CPU) і різними низькошвидкісними або застарілими пристроями, забезпечуючи безперебійну комунікацію та координацію в системі [9]. Цей інтерфейс виконує такі завдання, як управління USB–портами, контролерами SATA, аудіокодеками, мережевими інтерфейсами та іншими периферійними пристроями, що не вимагають високої швидкості, що безпосередньо підтримуються північним мостом або його сучасними аналогами. Знімаючи ці обов'язки з ЦП і Північного мосту, Південний міст підвищує ефективність системи, дозволяючи процесору зосередитися на обчислювальних завданнях, тоді як Південний міст контролює більш буденні, але життєво важливі операції управління периферійними пристроями. Такий розподіл обов'язків історично був основою дизайну чіпсетів, відображаючи стратегічний підхід до балансування продуктивності та розподілу ресурсів в персональних комп'ютерах.

Еволюція Південного мосту протягом багатьох років відображає швидкий прогрес у комп'ютерних технологіях та зміну вимог користувачів і додатків. Спочатку представлений як частина архітектури чіпсету на початку 1990–х років, Південний міст з'явився разом із Північним мостом, щоб вирішити проблему зростаючої складності конструкцій ПК, особливо з появою шини PCI . У своїх найперших ітераціях, таких як чіпсет Intel 430FX, випущений у 1995 році, південний міст був окремим фізичним компонентом, який керував повільнішими пристроями вводу–виводу, такими як шина ISA, контролери IDE та базові паралельні або послідовні порти. З розвитком

технологій Південний міст адаптувався до підтримки нових стандартів, таких як USB із впровадженням USB 1.0 у 1996 році, а пізніше USB 2.0, що значно розширило його роль у підключенні сучасних периферійних пристроїв. Згодом інтеграція таких функцій, як SATA, Gigabit Ethernet та вдосконалена обробка аудіо, ще більше трансформувала Південний міст, відображаючи рух галузі до вищих швидкостей передачі даних та більш досконалої підтримки пристроїв. До середини 2000–х років, з появою серії ICH (I/O Controller Hub) від Intel, Південний міст почав зливатися з Північним мостом в більш уніфіковану конструкцію чіпсета, і ця тенденція досягла свого апогею в розробці Platform Controller Hubs (PCH) приблизно в 2011 році. Ця еволюція підкреслює перехід від автономного компонента до інтегрованого рішення, зумовлений потребою в покращенні продуктивності, зменшенні затримки та енергоефективності в сучасних системах.

Походження Південного мосту можна простежити до початку 1990–х років, періоду, що ознаменувався переходом від однокристальних рішень до більш модульних архітектур в персональних комп'ютерах. У цю епоху відбувся підйом шини PCI як заміни старішої шини ISA, що вимагало чіткого розподілу обов'язків всередині чіпсету для задоволення різноманітних вимог до швидкості системних компонентів. Intel, що є гігантом галузі, представив цю концепцію у своїх чіпсетах серії 430, де Південний міст був офіційно встановлений як окремий елемент для обробки функцій вводу–виводу. Цей розвиток був відповіддю на зростаючу складність конструкцій ПК, оскільки виробники прагнули підтримувати все більшу кількість периферійних пристроїв, зберігаючи сумісність із застарілими системами. Таким чином, створення південного мосту стало важливою віхою в розробці чіпсетів, заклавши основу для масштабованих і гнучких архітектур, які домінували в галузі протягом наступних десятиліть.

З точки зору архітектури, південний міст характеризується високоінтегрованою конструкцією, що охоплює декілька контролерів та інтерфейсів, які координуються для полегшення комунікації між процесором,

чіпсетом та периферійними пристроями. У своїй основі південний міст підключений до північного мосту або PCH через високошвидкісний канал, такий як DMI в системах Intel, який служить основним каналом для обміну даними. Це з'єднання дозволяє Південному мосту отримувати команди та дані від центрального процесора, які він потім розподіляє між відповідними контролерами вводу–виводу. Архітектура зазвичай включає інтерфейс концентратора, який управляє потоком інформації, забезпечуючи синхронну роботу таких пристроїв, як USB–порти, SATA–накопичувачі та мережеві адаптери. Крім того, Південний міст має функції управління живленням, такі як підтримка ACPI що дозволяє регулювати енергоспоживання підключених периферійних пристроїв. Включення підтримки застарілих технологій, таких як LPC для старих пристроїв, ще більше підвищує його універсальність, роблячи його мостом між сучасними та застарілими технологіями. Ця складна архітектура відображає баланс між оптимізацією продуктивності та зворотньою сумісністю, що є критично важливим фактором при проєктуванні споживчих та корпоративних систем.

Функціональність південного мосту є настільки ж багатогранною і охоплює широкий спектр завдань, які є незамінними для роботи комп'ютерної системи. Він служить основним контролером для низькошвидкісних пристроїв вводу–виводу, керуючи передачею даних між процесором і периферійними пристроями, такими як клавіатури, миші та зовнішні пристрої зберігання даних через інтерфейси USB і SATA. Південний міст також контролює мережеве підключення, інтегруючи контролери Ethernet, що забезпечує доступ до Інтернету та комунікацію в локальній мережі. Ще однією ключовою функцією є обробка аудіо, оскільки Південний міст часто містить кодеки, що обробляють вхідний і вихідний звук, – функція, яка еволюціонувала від базового моноаудіо до підтримки об'ємного звуку високої чіткості. Крім того, він забезпечує підтримку застарілих інтерфейсів, гарантуючи сумісність зі старим обладнанням через LPC або подібні шини, що є особливо цінним у промислових або

спеціалізованих додатках. Здатність південного мосту керувати запитами на переривання та системними таймерами ще більше підсилює його роль центрального координатора, забезпечуючи роботу всіх периферійних пристроїв у межах часових обмежень системи. У сучасних реалізаціях, де Південний міст інтегрований у PCH, його функціональність розширилася і включає такі вдосконалені функції, як підтримка Thunderbolt або додаткових портів USB 3.0, що відображає його постійну адаптацію до нових технологій [9]. Ця комплексна функціональність підкреслює важливу роль Південного мосту в підтримці згуртованості та ефективності комп'ютерної екосистеми.

Підсумовуючи, Південний міст є фундаментальним елементом дизайну чіпсету, який еволюціонував від окремого компонента в 1990–х роках до інтегрованої частини сучасних PCH. Його призначення – управління операціями вводу–виводу, в поєднанні з адаптивною архітектурою та різноманітною функціональністю, зробило його важливим інтерфейсом у розвитку персональних комп'ютерів. У міру розвитку технологій спадщина Південного мосту в підтримці інновацій та сумісності, ймовірно, впливатиме на майбутні конструкції, забезпечуючи його актуальність у постійно мінливому ландшафті комп'ютерної архітектури.

2.4 Північний міст (North Bridge)

Північний міст, який історично був фундаментальним компонентом архітектури чіпсету персональних комп'ютерів, відігравав ключову роль у забезпеченні високошвидкісного зв'язку між центральним процесором та критичними системними ресурсами, забезпечуючи ефективну передачу даних та продуктивність системи. Його основним призначенням було служити високошвидкісним посередником, з'єднуючи CPU безпосередньо з основною пам'яттю системи і пристроями з високою пропускною здатністю, такими як відеокарта, через інтерфейси, такі як AGP або пізніший PCI Express. Керуючи цими критично важливими з'єднаннями, північний міст

забезпечував швидкий обмін даними, що було необхідним для виконання завдань, що вимагають високої продуктивності, таких як ігри, обробка мультимедіа та інші обчислювальні навантаження. На відміну від південного мосту, що обробляв повільніші пристрої вводу–виводу, Північний міст був розроблений з пріоритетом на швидкість та ефективність, виступаючи центральним вузлом для найбільш вимогливих компонентів системи. Такий розподіл обов'язків між північним та південним мостами дозволив збалансувати розподіл навантаження в чіпсеті, оптимізуючи загальну продуктивність системи шляхом ізоляції високошвидкісних операцій від завдань управління периферійними пристроями, які більш толерантні до затримок.

Історія розвитку North Bridge відображає динамічну еволюцію архітектури комп'ютерів, зумовлену прогресом у розробці процесорів, технологіях пам'яті та попитом на вищу продуктивність. Коли північний міст вперше з'явився на ринку, він був окремим фізичним чіпом на материнській платі, і така конструкція зберігалася з початку 1990–х до кінця 2000–х років [9]. Протягом цього періоду він зазнав значних змін, щоб пристосуватися до нових технологій, таких як перехід від підтримки SDRAM, до пам'яті DDR, яка забезпечувала більшу пропускну здатність для задоволення потреб сучасних додатків. Впровадження PCIe в середині 2000–х років ще більше розширило його роль, оскільки північний міст став відповідальним за управління цим високошвидкісним інтерфейсом для графічних карт, замінивши старий стандарт AGP. Однак Північний міст почав зникати з материнських плат приблизно в 2011 році, що було спричинено інтеграцією його основних функцій безпосередньо в процесор. Ця зміна була в основному обумовлена архітектурою Intel Nehalem, представленою в 2008 році, яка інтегрувала контролер пам'яті – ключову функцію північного мосту в сам процесор, зменшивши затримку і підвищивши ефективність. AMD вже застосувала цей підхід у своїй архітектурі K8 у 2003 році, створивши основу для переходу всієї галузі до рішень з інтегрованим процесором. В результаті

традиційний Північний міст був фактично виведений з обігу, а його функції перейшли до процесора та нового компонента, відомого як Platform Controller Hub, який об'єднав решту функцій чіпсету в одному чіпі.

Походження Північного мосту можна простежити до початку 1990–х років, коли зростаюча складність конструкцій персональних комп'ютерів вимагала більш модульного підходу до архітектури чіпсетів. Впровадження шини PCI як заміни старішої шини ISA підкреслило необхідність створення спеціального компонента для управління високошвидкісними з'єднаннями, що призвело до розробки North Bridge як частини чіпсетів Intel серії 430, починаючи з 430FX у 1995 році. Цей період став важливим поворотним моментом, оскільки північний міст був створений для задоволення зростаючих потреб у системній пам'яті та графічній обробці, які ставали критично важливими для зростаючого ринку споживчих ПК. Відокремивши високошвидкісні завдання від повільніших операцій вводу–виводу, що управляються південним мостом, ця конструкція дозволила виробникам більш ефективно масштабувати продуктивність системи, підтримуючи швидке поширення мультимедійних додатків і ранніх 3D–ігор, що було характерно для кінця 1990–х і початку 2000–х років.

Архітектура північного мосту була розроблена з пріоритетом на швидкість і пряме підключення, що відображало її роль як високопродуктивного хаба системи. Зазвичай вона мала контролер пам'яті, який безпосередньо взаємодіяв із системною оперативною пам'яттю, забезпечуючи доступ з низькою затримкою для центрального процесора. Крім того, північний міст керував фронтальною шиною, високошвидкісним каналом, що з'єднував процесор з чіпсетом, сприяючи швидкій передачі даних між процесором і пам'яттю. У пізніших версіях Північний міст також підтримував графічні інтерфейси, спочатку через слот AGP, а згодом через PCIe–лінії, що забезпечувало пропускну здатність, необхідну для сучасних графічних карт. Північний міст з’єднувався з південним мостом через спеціальний канал, такий як Intel Hub Link або AMD HyperTransport,

забезпечуючи скоординовану роботу чіпсету. Ця архітектура дозволяла північному мосту виступати в ролі вузького місця для високошвидкісних даних, що було ефективним рішенням на той час, але зрештою стало обмеженням, оскільки швидкість процесора і пам'яті перевищила можливості FSB і зовнішніх контролерів пам'яті.

Функціонально Північний міст відповідав за управління потоком даних між процесором, системною пам'яттю та периферійними пристроями з високою пропускною здатністю, забезпечуючи злагоджену роботу цих компонентів. Він відігравав важливу роль у доступі до пам'яті, оскільки процесор покладався на контролер пам'яті Північного мосту для читання та запису даних в оперативну пам'ять, що безпосередньо впливало на продуктивність системи. Північний міст також обробляв графічні дані, маршрутизуючи інформацію між процесором і графічною картою через інтерфейс AGP або PCIe, що ставало все більш важливим із зростанням популярності графічно інтенсивних додатків. Завдяки прямому контролю над цими високошвидкісними з'єднаннями Північний міст мінімізував затримки і максимізував пропускну здатність, що робило його важливим компонентом систем, призначених для виконання завдань, критичних до продуктивності. Однак його залежність від FSB вводила обмеження, оскільки шина ставала вузьким місцем при збільшенні швидкості процесора і зростанні вимог до пропускної здатності пам'яті, що в кінцевому підсумку сприяло його застаріванню, оскільки виробники шукали більш ефективні рішення.

У своїй остаточній формі Північний міст фактично зник із сучасних материнських плат, а його функції були інтегровані в процесор і контролер платформи (PCH). Станом на зараз, спадщина північного мосту очевидна в архітектурі сучасних систем, де процесори, такі як серія Intel Core або процесори AMD Ryzen, містять вбудовані контролери пам'яті та контролери PCIe. PCH, який замінив традиційну модель чіпсета, поєднує функції північного та південного мостів в одному чіпі, керуючи всіма входами/виходами та периферійними підключеннями, тоді як процесор

безпосередньо обробляє пам'ять та графіку. Ця інтеграція усунула необхідність в окремому Північному мості, зменшивши затримку, енергоспоживання та складність материнської плати. PCH підключається до процесора через високошвидкісний канал, такий як Intel DMI (або AMD Infinity Fabric, забезпечуючи більш оптимізовану та ефективну архітектуру, що відповідає вимогам сучасних обчислень, від високопродуктивних ігор до робочих навантажень штучного інтелекту. Хоча північний міст більше не існує як окремий компонент, його історичне значення як інструменту високошвидкісних міжз'єднань продовжує впливати на дизайн сучасних і майбутніх систем.

Підсумовуючи, північний міст колись був наріжним каменем архітектури комп'ютерів, керуючи критично важливими високошвидкісними з'єднаннями між процесором, пам'яттю та графічними пристроями. Його еволюція від окремого чіпа до інтегрованої функції в процесорі підкреслює невпинне прагнення галузі до продуктивності та ефективності. Хоча він зник із сучасних материнських плат, його спадщина зберігається в інтегрованих конструкціях сучасних процесорів і чіпсетів, підкреслюючи його тривалий вплив на розвиток технології персональних комп'ютерів.

3 ЕЛЕМЕНТИ І ФУНКЦІЇ, ЩО МОДЕЛЮЮТЬСЯ

3.1 Моделювання інтерфейсу PCIe на основі симуляції

Інтерфейс PCI Express є основою сучасних обчислювальних систем, забезпечуючи високошвидкісний послідовний обмін даними між процесорами та периферійними пристроями, такими як графічні процесори, твердотільні накопичувачі та мережеві інтерфейсні карти. Його багаторівнева архітектура, надійні механізми обробки помилок та масштабована пропускна здатність підкреслюють його значення в проєктуванні систем. Моделюючи ключові аспекти комунікації PCIe, включаючи як одноканальні (x1), так і чотириканальні (x4) конфігурації, симуляція пропонує візуальне та інтерактивне представлення роботи протоколу [5].Завдяки цьому програма не тільки ілюструє механізми роботи PCIe, але й є навчальним інструментом для розуміння його поведінки в ієрархічній топології.

В основі симуляції лежить побудована модель топології PCIe, візуалізована на полотні, де дані передаються між взаємопов'язаними компонентами. Кореневий комплекс, що втілює хост (зазвичай це процесор або чіпсет), ініціює комунікацію, слугуючи джерелом транзакцій, таких як запити на читання або запис пам'яті. Цей компонент, зображений у вигляді прямокутника, інкапсулює транзакцію, канал передачі даних і фізичні рівні, кожен з яких обробляє пакети перед передачею. З кореневого комплексу пакети проходять через PCIe-комутатор, центральний посередник, який маршрутизує трафік до різних кінцевих точок, забезпечуючи збереження розгалуженої структури протоколу. Комутатор, ще один прямокутний елемент, забезпечує безперебійну комунікацію, направляючи пакети до основної кінцевої точки, графічного процесора, який активно обробляє транзакції та відповідає пакетами завершення. Для підвищення реалістичності додано декоративні кінцеві точки SSD та NIC, їхня нижня

прозорість вказує на їхню пасивну роль, але вони збагачують зображення типової конфігурації сервера в симуляції і можуть бути реалізовані та змодельовані у майбутньому. Ці компоненти з'єднані лініями, що представляють PCIe-канали, причому модель x1 має один канал зі швидкістю 2,5 Гб/с (PCIe 2.1), а модель x4 – чотири канали зі швидкістю 30 Гб/с (PCIe 2.0), що підкреслює масштабованість протоколу.

Моделювання відображає суть комунікації PCIe за допомогою анімованих пакетів, які відображаються у вигляді кольорових еліпсів, що перетинають канали. Ці пакети – пакети рівня транзакцій та пакети рівня каналу передачі даних – виконують різні ролі, кожна з яких позначена унікальним кольором для полегшення інтерпретації. У міру переміщення цих пакетів індикатори рівня для Root Complex і GPU змінюються на чорні, сигналізуючи про активну обробку на транзакційному, каналі передачі даних і фізичному рівнях, таким чином ілюструючи багаторівневу архітектуру PCIe. Індикатор контролю потоку відстежує доступні кредити, забезпечуючи відповідність передачі даних механізму PCIe на основі кредитів, запобігаючи переповненню буфера. На додаток до цих візуальних елементів, журнал стану фіксує кожну подію, від передачі пакетів до відновлення після помилки, пропонуючи текстовий опис, що пояснює хід симуляції.

Процеси, змодельовані в рамках симуляції, відображають критичні механізми PCIe, починаючи з ініціалізації зв'язку. Ця початкова фаза імітує навчання фізичного рівня зв'язку, під час якого Root Complex і Switch виявляють один одного, узгоджують ширину смуги пропускання – одинарну для x1, чотири для x4 – і встановлюють швидкість зв'язку 2,5 GT/s або 5 GT/s відповідно. Журнал стану ретельно фіксує ці кроки, активуючи кнопки управління після завершення, що сигналізує про робочий зв'язок. Передача даних є основною частиною симуляції, де конфігурація x1 анімує пакети по одній смузі, наприклад, TLP зчитування пам'яті, що надсилається з Root Complex до GPU через Switch, а потім TLP завершення та ACK DLLP у відповідь. На відміну від цього, конфігурація x4 використовує чотири смуги,

випадково розподіляючи пакети для демонстрації паралельної передачі та збільшеної пропускної здатності, а журнал стану фіксує конкретну смугу та тип пакета. Операції запису пам'яті, що запускаються спеціальною кнопкою, анімують TLP, за якими слідують ACK DLLP, імітуючи опубліковані транзакції. Обробка помилок демонструється за допомогою, помилкового TLP зчитування пам'яті, що спонукає GPU надіслати NAK DLLP, запускаючи повторну передачу – процес, що реєструється для підкреслення надійності PCIe. Кожна анімація, що триває одну секунду на сегмент з затримками обробки, покращує візуальну чіткість цих операцій.

Візуальні та інтерактивні елементи симуляції поєднуються, утворюючи комплексну освітню платформу. Ієрархічна топологія з чітким розмежуванням кореневого комплексу, комутатора та кінцевих точок відображає реальні системи PCIe. Анімації пакетів, що проходять по одній або декількох лініях, підкреслюють послідовний та масштабований характер протоколу. Індикатори рівнів та механізм контролю потоку дають уявлення про внутрішню роботу PCIe, а журнал стану та легенда забезпечують доступність, зіставляючи кольори з типами пакетів для розуміння користувачем. Моделюючи конфігурації x1 та x4, симуляція не тільки демонструє основні операції PCIe, але й його здатність до підвищення продуктивності за допомогою масштабування смуг, пропонуючи відчутний зв'язок між теоретичними концепціями та практичним застосуванням.

Підсумовуючи, ця симуляція узагальнює суть архітектури та функціональності PCIe, моделюючи її основні компоненти та процеси, щоб забезпечити динамічний інструмент навчання. Включення конфігурацій x1 та x4 підкреслює універсальність PCIe, роблячи симуляцію цінним ресурсом для освітніх та дослідницьких цілей, що сприяє поглибленню розуміння цього ключового інтерфейсу в сучасних обчислювальних системах.

3.2 Моделювання інтерфейсу DMI на основі симуляції

Direct Media Interface, являє собою основне з'єднання між процесором і чіпсетом у сучасних обчислювальних системах, забезпечує високошвидкісну точкову комунікацію, необхідну для координації передачі даних до периферійних компонентів. Працюючи у версії 3.0, DMI забезпечує пропускну здатність до 3,94 ГБ/с; однак це обмеження часто стає вузьким місцем в операціях з інтенсивним введенням-виведенням, як свідчать чисельні дослідження та практичні експерименти [8]. Моделюючи основні компоненти та процеси DMI, симуляція візуалізує його оперативну динаміку, підкреслюючи ініціалізацію з'єднання, передачу даних та обмеження пропускної здатності, одночасно включаючи обробку в пам'яті. Ця робота не тільки ілюструє механіку DMI, але й служить освітнім інструментом для розуміння її ролі в сучасних обчисленнях.

Модель відтворює основні компоненти архітектури на основі DMI, ілюструючи поведінку та обмеження інтерфейсу. Топологія включає процесор, який ініціює операції з пам'яттю, безпосередньо підключений до пам'яті DDR4, та чіпсет, який забезпечує зв'язок з такими пристроями, як графічні процесори, SSD-накопичувачі, USB-порти та мережеві карти. Ці елементи візуально представлені на полотні: прямокутники символізують апаратні одиниці, а лінії, що їх з'єднують, моделюють потік даних. Чорна пунктирна лінія з написом «DMI 3.0 (3,94 ГБ/с)» підкреслює обмежену пропускну здатність інтерфейсу. SSD інтегровано для моделювання обробки в пам'яті, посилаючись на систему SSD-Off [8], яка зменшує передачу даних шляхом локальної обробки функцій.

Для демонстрації операцій DMI в моделюванні використовуються анімовані пакети різних кольорів. Жовті пакети позначають запити на читання пам'яті, фіолетові – записи в пам'ять, зелені – оброблені дані, повернуті SSD, а червоні– події, що створюють вузьке місце. Кожен пакет проходить через архітектуру, активуючи індикатори для транзакційного,

канального та фізичного рівнів, що представляють багаторівневу модель обробки: створення пакета, послідовність, перевірка помилок та фізична передача. Індикатор пропускної здатності на екрані повторює обмеження 3,94 ГБ/с, а журнал у реальному часі описує ключові події моделювання. Легенда пояснює значення кольорів пакетів для підтримки навчальних цілей.

Моделювання починається з ініціалізації з'єднання, імітуючи фазу навчання зв'язку фізичного рівня. Процесор і чіпсет ідентифікують один одного, домовляються про чотирисмугову конфігурацію і встановлюють швидкість передачі 8 ГТ/с, з часовими мітками, що імітують 1,5-секундне налаштування. Після завершення починаються послідовності передачі: надсилаються запити на читання, і у відповідь SSD повертає менші пакети, що представляють витягнуті функції, ефективно демонструючи економію пропускної здатності. Операції запису відбуваються за подібним процесом із використанням пакетів запису в пам’ять. Червоні пакети імітують затримки передачі, щоб візуалізувати вплив обмеженої пропускної здатності DMI під високим навантаженням.

Ключовим вдосконаленням симуляції є інтеграція обробки в пам'яті. Завдяки можливості SSD витягувати функції локально, система значно зменшує обсяг даних, що проходять через канал DMI. Пакети, що символізують оброблені дані, демонструють майже 38-кратне зменшення обсягу даних, що зменшує навантаження на пропускну здатність DMI в 3,94 ГБ/с [8]. Журнал відстежує цю оптимізацію, записуючи послідовності пакетів і участь SSD, а візуальні індикатори підкреслюють координуючу роль чіпсета. Це підкреслює не тільки архітектуру DMI, але й практичну стратегію пом'якшення її обмежень.

Підсумовуючи, симуляція пропонує всебічне та динамічне зображення інтерфейсу DMI. Вона моделює реалістичну топологію системи, демонструє багаторівневу обробку пакетів та візуально передає обмеження інтерфейсу та можливі оптимізації. Інтеграція налаштування з'єднання, інтерактивної анімації та логіки, що враховує пропускну здатність, робить симуляцію

технічно інформативною та педагогічно ефективною.

3.3 Моделювання інтерфейсу South Bridge та процесів комунікації

Південний міст, як один з основних елементів традиційної архітектури чіпсетів для ПК, відіграє фундаментальну роль у координації обміну даними між різними низькошвидкісними периферійними інтерфейсами та процесором [9]. Хоча сучасні архітектури часто інтегрують багато його функцій в єдиний контролер платформи, концептуальна та освітня цінність південного мосту залишається значною. Студентам і дослідникам часто важко уявити, як цей компонент з'єднується з декількома підсистемами в рамках більш широкої архітектури комп'ютера і керує ними. З огляду на це, моделювання Південного мосту в розробленій програмі зосереджується не тільки на демонстрації структурної топології, але й на моделюванні окремих ключових процесів, що відбуваються в комунікаційній області Південного мосту.

З топологічної точки зору, модельована система представляє візуальну структуру, яка організовує Південний міст у центрі комунікації, розташований безпосередньо під Північним мостом і утворює хаб для декількох підключених інтерфейсів. Ця схема відображає ієрархічні деревоподібні схеми, які традиційно використовуються в діаграмах архітектури системи, підкріплюючи ідею централізованого контролю трафіку. Програма чітко окреслює, як Південний міст взаємодіє з високошвидкісними та низькошвидкісними периферійними інтерфейсами. Це включає підключення до ліній PCI Express, портів SATA, інтерфейсів USB, інтегрованого аудіо, Flash ROM та Super I/O – причому інтерфейси PCIe, SATA та USB моделюються детально, а інші представлені візуально для збереження архітектурної точності.

Що стосується конкретних структурних елементів, програма використовує інтерфейсні блоки, розташовані навколо Південного мосту,

щоб відобразити їх реальну взаємодію. PCIe, SATA та USB, які є одними з найважливіших і найпоширеніших інтерфейсів у сучасних системах, моделюються як інтерактивні компоненти. Вони реалізовані не просто як візуальні заповнювачі, а як кнопки, які направляють користувача до детальних вікон симуляції. Кожне вікно симуляції ілюструє основні процеси комунікації між південним мостом і підключеним пристроєм, надаючи можливість детальніше розглянути реальні потоки даних, зберігаючи при цьому доступність для учнів.

У рамках моделювання PCI Express, користувачеві представлена чітка ієрархія, що показує, як дані передаються від процесора до кінцевих точок, таких як графічні процесори, через комутатор, з візуалізацією контролю потоку та обробки помилок за допомогою пакетів та індикаторів рівнів. Однак у контексті моделювання South Bridge доступ до цього самого інтерфейсу здійснюється з іншої архітектурної позиції – він представляє лінії PCIe, що простягаються від самого South Bridge, а не від кореневого комплексу. Ця подвійність підкріплює ідею, що PCIe-канали можуть виходити з декількох компонентів системи, і підкреслює гнучкість топології PCIe.

На відміну від цього, інтерфейс SATA використовує нову розроблену симуляцію, яка моделює пряме з'єднання між південним мостом і пристроєм SATA, таким як твердотільний накопичувач. Ця симуляція зосереджена на демонстрації різних типів пакетів передачі даних, які є фундаментальними для протоколу SATA. Включено три основні типи пакетів: Register FIS, який ініціює комунікацію через обмін даними про стан і управління; Data FIS, який передає фактичні дані; та Set Device Bits FIS, який сигналізує про оновлення операційного стану, що зазвичай пов'язано з такими функціями, як Native Command Queuing. Програма демонструє ці транзакції за допомогою чітких візуальних індикаторів та анімації руху, що дозволяє користувачам інтуїтивно зрозуміти, як інформація проходить через інтерфейс. Моделювання також включає контекстну інформацію, таку як ефективна

швидкість передачі даних SATA 3.0, що надає кількісну довідку на додаток до візуального представлення.

Інтерфейс USB також моделюється у спрощеній, але концептуально точній формі. Він зосереджується на демонстрації двонаправленої пакетної комунікації, яка визначає роботу шини USB. Моделювання включає транзакції токенів IN і OUT, підкреслюючи їх спрямованість і функцію. Наприклад, токен IN запитує дані від підключеного пристрою, викликаючи відповідь у вигляді пакета даних, за яким слідує підтвердження. З іншого боку, токен OUT представляє ініційований хостом перенос даних на пристрій, що завершується негативним підтвердженням, щоб проілюструвати умови, коли пристрій не готовий приймати нові дані. Ці потоки представлені за допомогою анімованих пакетів і спрямованих переходів, що дозволяє учням побачити механіку USB на основі токенів, не заглиблюючись у складні електричні сигнали або арбітраж шини.

Хоча лише три інтерфейси – PCIe, SATA та USB – моделюються з усіма деталями, візуальна присутність інших інтерфейсів, таких як інтегрований аудіо, Flash ROM та Super I/O, забезпечує топологічну точність Південного мосту як багатофункціонального компонента. Кожен з цих додаткових елементів розміщений відповідно до його очікуваного архітектурного розташування та підключений до Південного мосту. Це підсилює просторові взаємозв'язки в системі, зберігаючи при цьому чіткий та зрозумілий візуальний макет. Навіть без повної анімації для кожного з'єднання, візуалізація того, як кожен інтерфейс відгалужується від південного мосту, допомагає зміцнити цілісне розуміння структури чіпсету.

З точки зору моделювання процесів, мета полягає не в тому, щоб відтворити кожну деталь на рівні електротехніки або протоколу, а в тому, щоб запропонувати інтуїтивно зрозумілу топологію основних взаємодій. Пакети, що відображаються в кожній симуляції – незалежно від того, чи представляють вони читання пам'яті, запис, команди управління або сигнали підтвердження – є символічними аналогами своїх реальних відповідників і

призначені для відображення динаміки на рівні протоколу в доступній формі. Анімації слідують логічним шляхам, що представляють потік даних, час відгуку та сигналізацію команд, дозволяючи користувачам побудувати функціональну модель того, як ці системи працюють на практиці.

Багаторівнева структура також зберігається в кожній симуляції, щоб відобразити багаторівневий характер протоколів інтерфейсу. Наприклад, навіть у спрощених симуляціях SATA та USB існує неявне розуміння логіки управління хостом, транспорту та фізичних рівнів, хоча вони можуть бути візуально не розділені. На відміну від цього, у симуляції PCIe рівні явно вказані на обох кінцях каналу зв'язку, що допомагає користувачам співвідносити такі дії, як ініціалізація зв'язку або управління потоком, з конкретними рівнями протоколу.

Підсумовуючи, частина моделювання South Bridge виконує подвійну функцію в загальній структурі проєкту. По-перше, він перетворює теоретичні знання про архітектуру чіпсету та функції інтерфейсу в конкретну візуальну форму. По-друге, він служить для динамічного моделювання конкретних процесів комунікації, заповнюючи прогалину між статичними діаграмами та активною поведінкою системи. Завдяки можливості вибіркової взаємодії з ключовими інтерфейсами при збереженні точної топологічної карти, модель забезпечує як структурну чіткість, так і функціональну глибину.

3.4 Моделювання інтерфейсу North Bridge та процесів обміну даними

Наостанок хотілось би звернути увагу на моделювання Північного мосту, важливого компонента традиційних архітектур комп'ютерів. Південний міст часто асоціюється з периферійним контролем і операціями з даними на низькій швидкості, тоді як Північний міст історично відповідальний за високошвидкісну комунікацію між процесором, оперативною пам'яттю та графічними інтерфейсами [9]. З огляду на його центральну роль в управлінні продуктивністю системи та затримкою доступу

до пам'яті, його включення в це імітаційне середовище має як освітню, так і аналітичну цінність.

Моделювання починається з топологічної схеми, яка відображає типову архітектуру материнської плати, з Північним мостом, розміщеним у центрі високошвидкісного зв'язку. Розташований над Південним мостом, він підтримує пряме з'єднання через широкий канал взаємозв'язку, що відображає інтерфейси, такі як Direct Media Interface від Intel або ж це можуть бути більш ранні варіанти, такі як PCI або LPC. Ліворуч від північного мосту розташований процесор, а праворуч – модуль оперативної пам'яті. Ця структура не тільки візуально чітка і симетрична, але й підкреслює ключову роль північного мосту як посередника між виконанням процесора і доступом до пам'яті, одночасно полегшуючи зв'язок з периферійними пристроями через південний міст [9].

Щоб візуалізувати цю взаємодію, симуляція представляє три ключові процеси, які відображають типові сценарії, в яких бере участь Північний міст. Перший процес, моделює взаємодію між ЦП і Північним мостом через фронтальну шину. Ця транзакція починається, коли ЦП ініціює запит інструкції, надсилаючи адресу через FSB. Потім модель відображає другий пакет – «Read Command» – який означає фазу інструкції запиту. Нарешті, надсилається пакет «Запит даних», що символізує запит процесора на всю лінію кешу. Після обробки запиту північним мостом симуляція повертає пакет з позначкою «Інструкція» назад до процесора. Цей потік фіксує фази адреси, контролю та даних, характерні для протоколу FSB, і відображає тип пакетних транзакцій або пакетних структур, що зустрічаються в архітектурах на основі HyperTransport.

Другий процес, змодельований у моделі, демонструє комунікацію між північним мостом і оперативною пам'яттю. Ця послідовність починається, коли північний міст, отримавши запит на доступ до пам'яті від центрального процесора або іншої підсистеми, активує відповідний рядок DRAM. Далі відбувається передача пакета «Column Address», що представляє метод

сигналізації, який використовується для вибору точної комірки пам'яті. Третя фаза, «Read Command», також надсилається для завершення структури команди. Після невеликої модельованої затримки, що представляє внутрішній час доступу до оперативної пам'яті, оперативна пам'ять надсилає назад пакет даних, що передається з оперативної пам'яті до північного мосту. Цей сценарій точно відображає основну роботу синхронної DRAM, у описуваному моделюванні представлена DDR3-1600, яка здійснює зв'язок за допомогою заздалегідь визначених циклів і пакетів 64-бітних або 128-бітних даних. Модель ефективно узагальнює складність роботи контролера пам'яті, який видає низькорівневі команди DRAM, і перетворює їх на зрозумілі для користувача анімації.

Третій процес моделює роль Північного мосту в забезпеченні зв'язку між Південним мостом і системною пам'яттю, як правило, під час транзакції прямого доступу до пам'яті. У моделюванні процес починається з пакета «DMA Req», що надходить від Південного мосту, який імітує запит на доступ до пам'яті від пристрою SATA або мережевої карти. Цей запит передається Південним мостом до Північного мосту, що відображає спосіб передачі даних через DMI або подібні міжз'єднання. Після отримання цього запиту Північний міст записує запитувані дані в оперативну пам'ять. Після завершення операції з пам'яттю Північний міст повертає сигнал підтвердження до центрального процесора, закриваючи цикл DMA. Цей обмін моделює, як сучасні чіпсети управляють асинхронними передачами даних, обходячи участь центрального процесора, щоб зменшити затримку і розвантажити системні ресурси.

Візуально кожна транзакція представлена у вигляді анімованого руху пакетів між компонентами, з використанням кольорових еліпсів та описових міток для передачі типу даних або команди, що надсилається. Ці візуальні підказки доповнюються легендою, яка дозволяє користувачам швидко визначити значення кожного пакета, позначеного кольором, тим самим підвищуючи освітню зрозумілість інтерфейсу. Крім того, журнал стану

реєструє кожен крок транзакції, надаючи доступний текстовий розбір процесу для користувачів, які бажають детально простежити протокол.

Симуляція Північного мосту, хоча і дещо спрощена, але точно відображає принципи взаємодії між компонентами в традиційних архітектурах ПК. Вона об'єднує аспекти пакетної комунікації, логіки управління пам'яттю та обробки переривань. Водночас вона знайомить студентів із впливом на продуктивність та структурою високошвидкісних шинних систем, таких як FSB, шини пам'яті та DMI. Візуалізуючи ці процеси таким чином, що відображає реальну поведінку системи, симуляція слугує візуальним сполученням, що допомагає учням засвоїти складні концепції, пов'язані з часовими параметрами та транзакціями, які в підручниках пояснюються абстрактно.

4 ОПИС ФУНКЦІОНАЛУ ТА ІНТЕРФЕЙСУ СТВОРЕННОГО ПРОГРАМНОГО РІШЕННЯ

4.1 Загальна архітектура програмного забезпечення

Головною метою створення програмного забезпечення була розробка для навчальних цілей, в першу чергу для початківців, які тільки починають вивчати архітектуру комп'ютерних систем. Часто в теоретичних курсах такі терміни, як «PCI Express», «South Bridge» або «DMI», залишаються абстрактними або погано зрозумілими. Тому існувала очевидна потреба в інструменті, який міг би допомогти подолати розрив між теорією та практичним, візуальним розумінням. Основна ідея застосунку полягає в тому, щоб дати користувачам можливість вивчити базову теоретичну інформацію про комп'ютерні інтерфейси, візуалізувати їх топології та моделювати спрощені моделі обміну даними через ці інтерфейси. Як для студентів, так і для викладачів, додаток може використовуватися як повноцінний інтерактивний навчальний інструмент або частково – наприклад, для вилучення знімків екрана для використання в лекціях або презентаціях. Це робить програму придатною як для демонстрацій в аудиторії, так і для самостійного навчання.

На зображенні нижче (рисунок 4.1) представлена структура проєкту. Ця структура чітко ілюструє модульну організацію: кожен інтерфейс – PCIe, DMI, South Bridge North Bridge – реалізований у вигляді окремого вікна, що містить окрему інформаційну сторінку, сторінку симуляції та відповідні анімаційні та інтерфейсні компоненти. Все це організовано в папки відповідно до їх функцій, що забезпечує чіткий і зручний макет проєкту. Так як програмна реалізаціє є спрощеною, програма має моноархітектуру, тобто реалізована як одношарова програма WPF. Вся логіка – включаючи елементи управління інтерфейсом, анімації та процедури симуляції – міститься на

одному рівні.

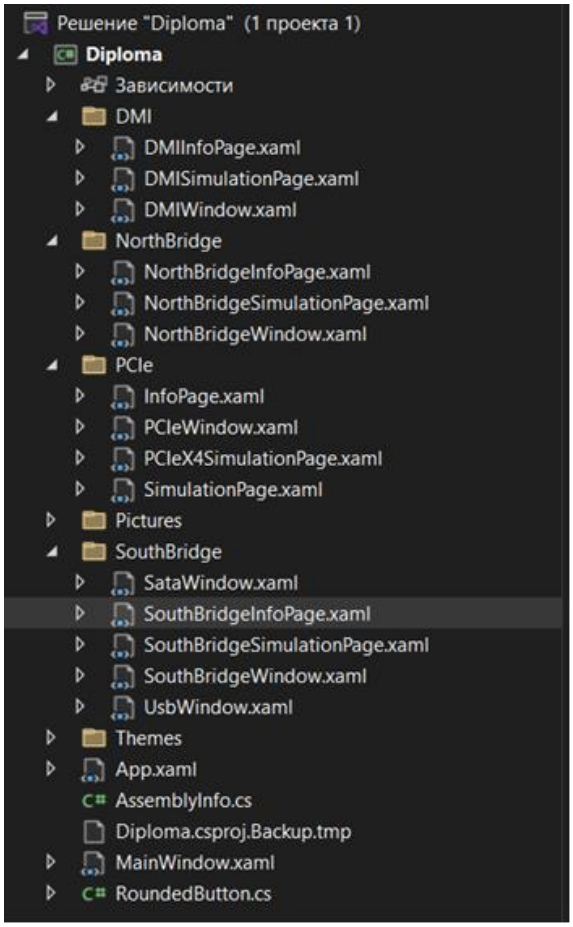


Рисунок 4.1 – Структура проєкту у середовищі Visual Studio

Іншим ключовим архітектурним елементом є система навігації між вікнами програми. Вона представлена діаграмою переходу між меню та формами, яка показує, як користувач взаємодіє з додатком, переходячи з однієї сторінки на іншу (рисунок 4.2). Користувач завжди починає з головного вікна, яке служить домашнім екраном і містить кнопки, що ведуть до кожного модуля інтерфейсу. Після натискання кнопки відкривається нове вікно з двома основними вкладками у верхній частині: «Інформація» та «Моделювання», а також кнопкою «Назад». Ця послідовна структура використовується у всіх модулях інтерфейсу і створює єдину, інтуїтивно зрозумілу логіку навігації.

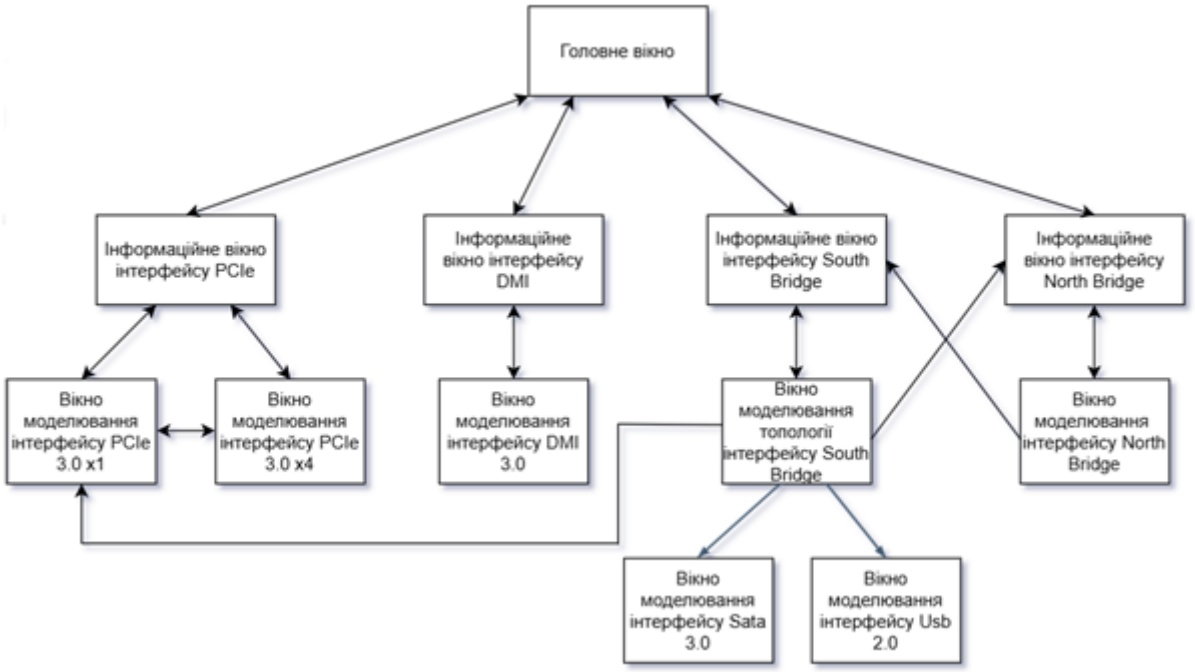


Рисунок 4.2 – Діаграма топології програмного застосунку

З головного меню користувач може перейти до будь-якого реалізованого інтерфейсу. У кожному вікні інтерфейсу користувач може вибрати перегляд загальної інформації про інтерфейс або перейти до його симуляції. Кожне вікно симуляції містить статичну візуальну топологію (що показує, як інтерфейс з'єднується з іншими компонентами системи), інтерактивні кнопки для запуску анімацій, легенди для пакетів, індикатори рівнів та журнал стану, що відображає всі події системи. Ці елементи працюють разом, щоб симулювати реальні процеси в спрощеній формі – показуючи, як передаються пакети, як протоколи працюють на різних архітектурних рівнях і як вузькі місця або помилки можуть впливати на поведінку системи.

Завдяки послідовній архітектурній логіці, яка використовується у всіх модулях, програмне забезпечення є зручним для користувачів і легкодоступним для студентів. Більше того, воно забезпечує міцну основу для потенційного розширення, чи то шляхом впровадження нових інтерфейсів, чи то шляхом додавання більш глибоких рівнів

функціональності до існуючих. Таким чином, система не тільки відповідає цілям поточної дипломної роботи, але й служить перспективною платформою для подальшого розвитку та інтеграції в освітній процес.

4.2 Головна сторінка застосунку

Головне вікно програми служить відправною точкою для взаємодії з користувачем, пропонуючи простий, але структурований вхід в освітнє середовище. Його макет мінімалістичний та інтуїтивно зрозумілий, містить чотири основні кнопки навігації, які направляють користувача до різних модулів інтерфейсу: PCI Express, Direct Media Interface, South Bridge та North Bridge. Кожна з цих кнопок відкриває нове вікно, присвячене відповідному інтерфейсу, де користувач може вибрати між читанням теоретичних відомостей або запуском інтерактивної симуляції (рисунок 4.3). Такий дизайн допомагає забезпечити логічний хід навчання та чітке розділення теорії та практичних експериментів.

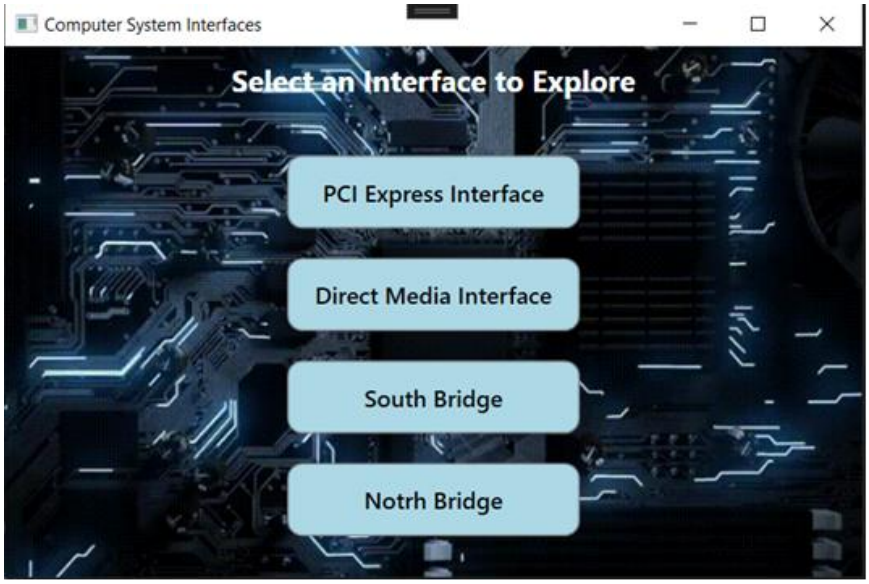


Рисунок 4.3 – Головна сторінка застосунку

Щоб підвищити візуальну привабливість і сучасний вигляд програми, головне вікно містить динамічну анімацію фону у вигляді циклічного GIF-

файлу (лістинг 4.1). Ця анімація створює фон, не заважаючи читабельності та зручності використання основних елементів інтерфейсу. Для реалізації цієї функції в рамках WPF проєкт використовує зовнішню бібліотеку – WpfAnimatedGif. Ця бібліотека забезпечує підтримку вбудовування та відображення анімованих GIF-файлів у програмах WPF, долаючи вбудовані обмеження платформи в обробці відтворення GIF-файлів.

Використовуючи клас ImageBehavior, наданий WpfAnimatedGif, програма може відображати плавну, повторювану анімацію, яка завантажується під час виконання і безперебійно відтворюється у фоновому режимі. Включення анімації не тільки покращує користувацький досвід, але й свідчить про більш сучасний стандарт інтерфейсу користувача, що особливо актуально для освітнього програмного забезпечення, орієнтованого на молодшу аудиторію.

Лістинг 4.1 – Розміщення анімованого гіф-зображення на задньому фоні вікна (файл MainWindow.xaml)

```
<VisualBrush>
    <VisualBrush.Visual>
        <Image
gif:ImageBehavior.AnimatedSource="/Pictures/back6.gif"
       gif:ImageBehavior.AutoStart="True"
       gif:ImageBehavior.RepeatBehavior="1"/>
    </VisualBrush.Visual>
</VisualBrush>
```

Таким чином, головне вікно виконує функцію як центрального центру навігації, так і естетичного вступу до навчального середовища. Воно ефективно направляє студента до цільових навчальних модулів, зберігаючи при цьому візуально привабливий, інтерактивний вигляд.

4.3 Інформаційні сторінки застосунку

Кожен інтерфейс, представлений у програмі супроводжується спеціальною інформаційною сторінкою, призначеною для ознайомлення з

теоретичними основами вибраного інтерфейсу перед переходом до моделювання. Ці сторінки призначені як перший крок у процесі навчання і пропонують користувачам, особливо студентам, які мають мінімальний досвід роботи з архітектурою комп'ютерів, структуроване і доступне пояснення того, що таке кожен інтерфейс, де він знаходиться в більш широкій топології системи і яку роль він відіграє.

Інформаційні сторінки мають однаковий інтерфейс та структуру, що забезпечує узгодженість у всій програмі. У верхній частині кожної сторінки заголовок представляє інтерфейс. Далі йде один або кілька абзаців пояснювального тексту, написаного спрощеним академічним стилем, щоб зробити зміст доступним для новачків. Ці тексти описують основні функції інтерфейсу, його місце в системі, а також історичний або практичний контекст, який може допомогти в розумінні його значення – наприклад, той факт, що North Bridge в сучасних архітектурах в основному витіснений, але залишається важливим для розуміння застарілих систем.

Під текстовою інформацією на кожній сторінці відображається одна або кілька топологічних діаграм. Ці діаграми є статичними зображеннями, взятими або адаптованими з загальнодоступних матеріалів, що наочно показують, як інтерфейс підключається до інших компонентів комп'ютерної системи. Наприклад, у випадку PCI Express схеми ілюструють, як CPU з’єднується з GPU або пристроєм зберігання даних через PCIe-комутатор. Для інтерфейсу DMI основна увага приділяється точковому з'єднанню між процесором і чіпсетом. На сторінках South Bridge і North Bridge показано кілька топологій, щоб проілюструвати як старі, так і більш сучасні інтерпретації структури системи.

Зображення розміщені поруч у горизонтальному форматі, тому вони відображаються в одному рядку. Це дозволяє користувачеві легко порівнювати різні версії топології без надмірного прокручування (рисунок 4.4). Формат динамічно налаштовується, щоб забезпечити візуальну збалансованість зображень, а поля застосовуються для забезпечення чіткості

та інтервалів.

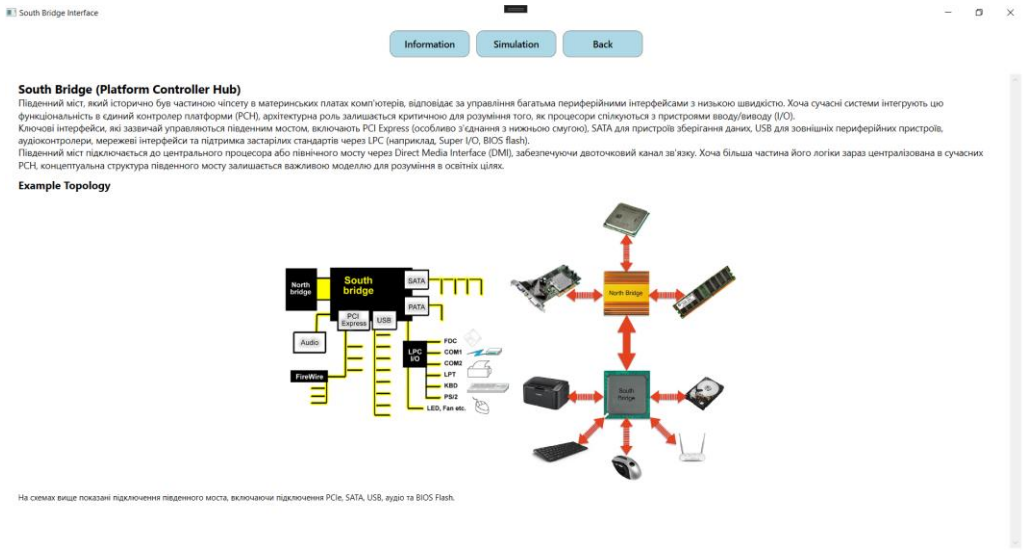


Рисунок 4.4 – Інформаційне вікно інтерфейсу South Bridge

Дизайн цих сторінок поєднує простоту та чіткість. На цих сторінках немає інтерактивних компонентів або анімацій; мета полягає в тому, щоб надати статичний, читабельний контекст перед переходом у режим моделювання. Таке розділення гарантує, що користувачі можуть повністю засвоїти необхідну теорію, перш ніж перейти до візуального моделювання або анімації на основі пакетів.

Поєднуючи візуальні матеріали та лаконічний, але інформативний текст, інформаційні сторінки допомагають сформувати базове розуміння кожного інтерфейсу. Такий підхід підтримує як самостійне навчання – оскільки студенти можуть самостійно орієнтуватися в інтерфейсах, – так і навчання в аудиторії, де викладач може проєктувати інформаційні сторінки або використовувати знімки екрана під час пояснення. Незалежно від того, чи використовуються вони самостійно, чи разом із практичними демонстраціями в симуляторі, ці інформаційні сторінки слугують важливим з'єданнням між абстрактними поняттями та прикладним візуальним моделюванням.

4.4 Моделювання роботи інтерфейсу PCI Express з однією та чотирма лініями передачі даних

Вікно моделювання PCI Express (рисунок 4.5) є однією з найбільш докладних і детальних частин програми, призначеною для імітації поведінки інтерфейсу PCIe у візуально зрозумілій та освітній формі. Ця імітація забезпечує інтерактивну візуалізацію того, як високошвидкісні дані передаються між центральним процесором, що представлений як кореневий комплекс і кінцевим пристроєм, таким як графічний процесор, через комутатор – типову конфігурацію в сучасних ПК і серверах.

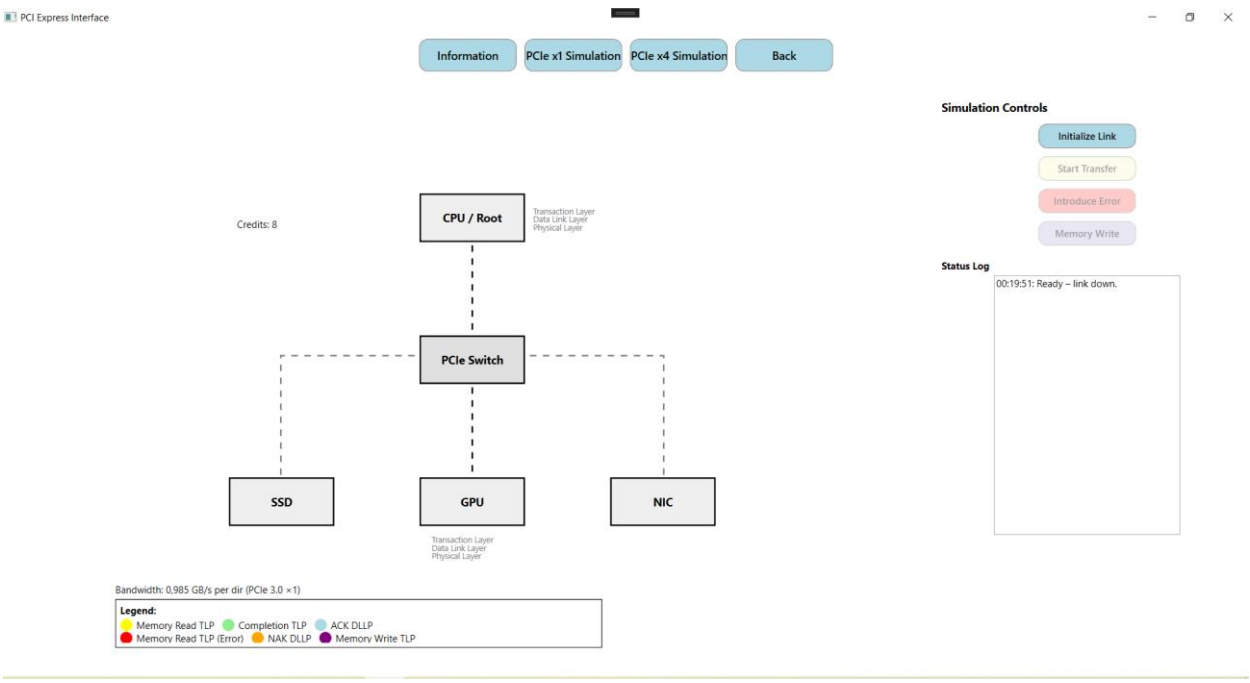


Рисунок 4.5 – Вікно моделювання інтерфейсу PCI Express x1

Після запуску моделювання PCIe з головного вікна користувач переходить до спеціальної сцени, яка представляє спрощену, але реалістичну топологію. У верхній центральній частині полотна знаходиться ЦП, точка походження всіх транзакцій. Трохи нижче знаходиться PCIe Switch, який виконує функцію маршрутизатора. У нижній частині екрану відображається графічний процесор як основна кінцева точка зв'язку. Зліва і справа

розташовані два декоративні пристрої – SSD і NIC – включені для візуального реалізму і для підкреслення масштабованості комутаторів PCIe, хоча вони не є інтерактивними і не функціонують у поточній версії програми.

Основна освітня функція цієї симуляції реалізується за допомогою набору інтерактивних кнопок, розміщених на правій панелі керування. Кожна кнопка запускає певний процес в інтерфейсі PCIe, і кожен процес візуалізується за допомогою анімованих пакетів, що переміщуються по смугах між компонентами. Ці анімації не є довільними; кожен пакет імітує конкретний тип транзакції PCIe, а відповідний журнал стану праворуч відстежує кожен етап операції у вигляді тексту з позначкою часу.

Кнопка «Ініціалізація зв'язку» запускає симуляцію, моделюючи процес узгодження фізичного рівня. Це включає виявлення елементів, узгодження ширини смуги та конфігурацію швидкості. Після завершення цього процесу в журнал стану записується повідомлення із зазначенням пропускної здатності, а решта кнопок симуляції стають активними. Якщо користувач спробує ініціювати транзакцію до ініціалізації посилання, програма попередить в журналі стану, зберігаючи реалістичність протоколу.

Після ініціалізації кнопка «Читання пам’яті» дозволяє процесору відправити запит на читання пам'яті до графічного процесора. У наведеному лістингу 4.2 демонструється анімація пакету запису в пам’ять, що надсилається з ЦП до графічного процесору.

Лістинг 4.2 – Анімація пакету «Memory Read» (файл SimulationPage.xaml.cs)

```
await AnimatePacketAsync(440, 440, 118, 207, "Memory Read TLP",
Colors.Yellow, isCPUtoGPU: true);
await AnimatePacketAsync(440, 440, 263, 382, "Memory Read TLP",
Colors.Yellow, isCPUtoGPU: true);
```

Жовтий пакет з написом «Memory Read» анімовано переміщується від кореневого комплексу до графічного процесора через комутатор. Після досягнення графічного процесора повертається зелений пакет «Completion»,

що символізує відправлення даних назад. Незабаром після цього відправляється блакитний пакет «ACK» як підтвердження. Симуляція виділяє активні архітектурні рівні – транзакційний, каналу передачі даних і фізичний – тимчасово змінюючи їх колір, ілюструючи, як запит обробляється на всіх рівнях стека PCIe.

Для подальшої демонстрації обробки помилок кнопка «Симуляція несправності» вводить червоний пакет «Error» або «Throttled», який імітує несправну або затриману транзакцію. Це викликає негативне підтвердження NAK, за яким слідує повторна передача. Цей процес моделює надійний механізм виправлення помилок каналу PCIe, включаючи його реакцію на пошкодження пакета, виявлене за допомогою CRC або невідповідності послідовності.

Журнал стану є важливим компонентом симуляції. Він записує кожну значущу подію послідовно – від ініціалізації каналу, передачі пакетів, обробки рівнів до оновлення кредитів. Цей поточний коментар забезпечує прозорість симуляції та підсилює освітню мету, дозволяючи користувачам стежити за кожним кроком у зрозумілій для людини формі.

Симуляція чітко розрізняє різні типи пакетів за допомогою кольорів:

- жовтий колір для TLP зчитування пам'яті;
- зелений колір для TLP завершення;
- блакитний колір для ACK DLLP;
- фіолетовий колір для TLP запису пам'яті;
- червоний колір для пакетів з помилками або обмежених пакетів;
- помаранчевий колір для NAK DLLP.

Кожен колір і його функція пояснюються в панелі легенди, розташованій у нижньому лівому куті полотна. Це гарантує, що користувачі, які не знайомі з термінологією PCIe, все одно зможуть інтерпретувати поточну анімацію та зрозуміти поведінку симуляції.

На додаток до описаної вище конфігурації x1, програма також включає варіант PCIe x4 (рисунок 4.6). У режимі x4 базова поведінка моделювання

залишається незмінною – застосовуються ті самі типи пакетів, функції кнопок та механізми контролю потоку – але на екрані відображаються чотири паралельні анімовані смуги замість однієї, що візуально вказує на збільшення ширини смуги. Ця невелика, але потужна зміна дозволяє студентам інтуїтивно зрозуміти, як PCIe масштабує продуктивність, об'єднуючи кілька смуг без зміни поведінки основного протоколу.

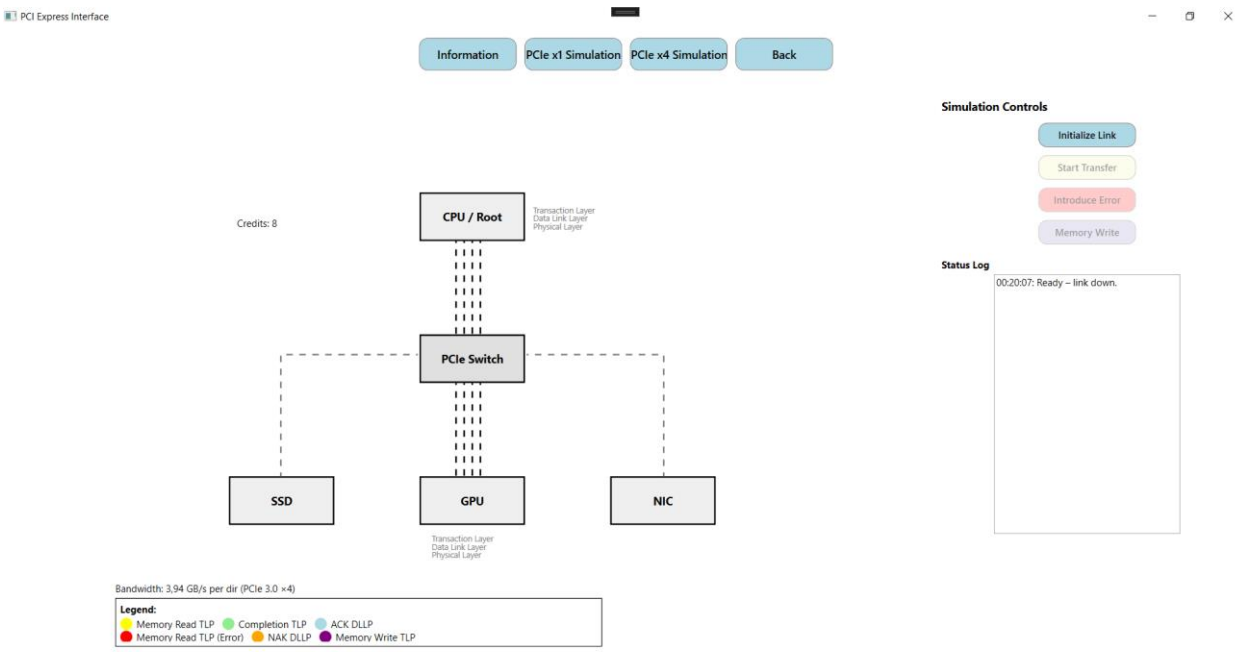


Рисунок 4.6 – Вікно моделювання інтерфейсу PCI Express x4

Загалом, симуляція PCIe служить динамічним навчальним інструментом, який у спрощений, але точний спосіб демонструє активність на рівні протоколу. Поєднуючи анімацію, інтерактивні елементи керування, реєстрацію в режимі реального часу та багаторівневість протоколу, вона дозволяє студентам-початківцям візуалізувати те, що зазвичай залишається невидимим у роботі апаратного забезпечення. Водночас вона надає викладачам гнучкий навчальний ресурс.

4.5 Моделювання роботи інтерфейсу DMI

Вікно моделювання DMI (рисунок 4.7) є одним з найбільш наочних прикладів того, як програма поєднує теоретичні принципи інтерфейсу з практичним інтерактивним моделюванням. Ця частина програми присвячена моделюванню роботи інтерфейсу DMI 3.0, важливого високошвидкісного з'єднання, яке пов'язує центральний процесор із чіпсетом. Інтерфейс відіграє центральну роль у сучасних комп'ютерних системах, полегшуючи комунікацію між процесором і різними пристроями вводу-виводу, включаючи накопичувачі, периферійні пристрої USB і мережеві карти, які підключені через чіпсет.

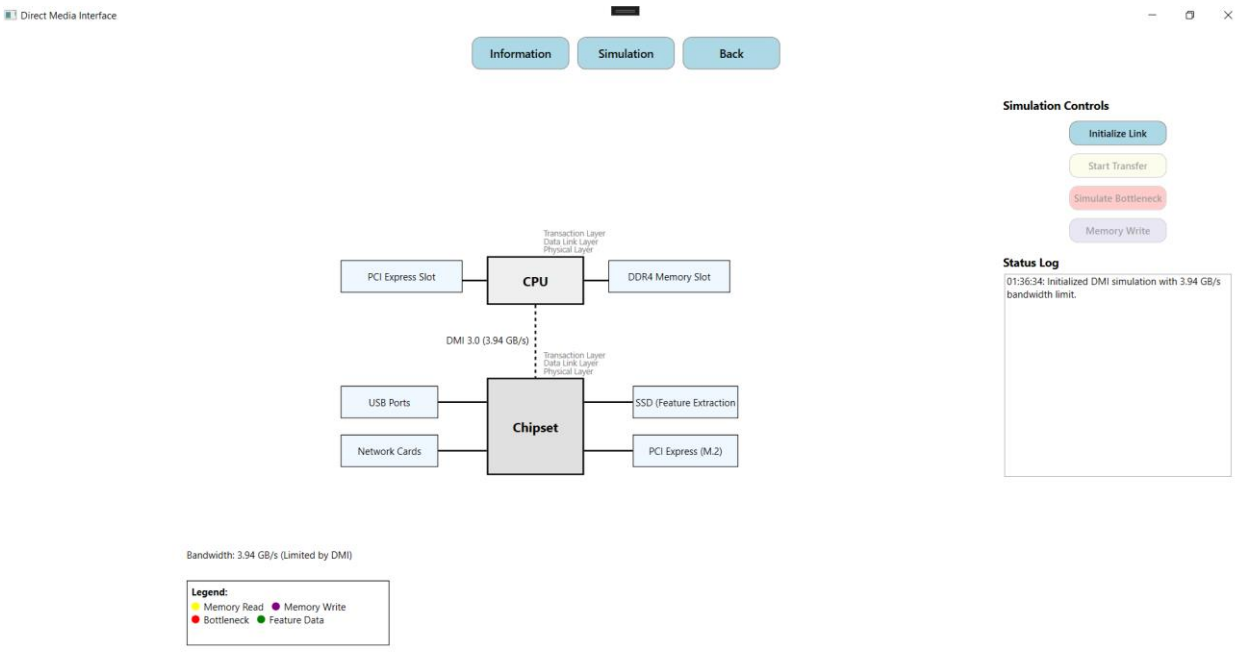


Рисунок 4.7 – Вікно моделювання інтерфейсу DMI

Візуальна структура цієї симуляції організована у вигляді топологічної діаграми, яка точно відтворює спрощену сучасну архітектуру ПК. У верхній частині вікна розташований процесор, який служить відправною точкою для більшості транзакцій. По обидва боки знаходяться пам'ять DDR4 і PCI Express, що позначають високопродуктивну пам'ять і інтерфейси

розширення. У нижній центральній частині розташований чіпсет, основний компонент, відповідальний за маршрутизацію та управління підключеннями до підсистем вводу-виводу з нижчою швидкістю. Декілька компонентів, таких як USB-порти, мережеві адаптери, SATA, M.2 та SSD-модулі, розташовані навколо чіпсету, щоб відобразити реальні периферійні підключення. Вони не є лише декоративними: у цій симуляції SSD відіграє ключову роль у демонстрації розвантаження обробки пам'яті.

Характерною особливістю цієї топології є вертикальна пунктирна чорна лінія, що з'єднує процесор і чіпсет, з позначкою «DMI 3.0 (3,94 ГБ/с)». Цей елемент не тільки графічний, але й символічно підкреслює основну освітню ідею симуляції – обмеження пропускної здатності інтерфейсу DMI. Мета полягає в тому, щоб допомогти студентам уявити, чому ця високошвидкісна міжсистемна мережа може стати вузьким місцем у додатках з інтенсивним введенням-виведенням, таких як ті, що передбачають передачу великих обсягів даних або обчислення на базі SSD.

У правій частині вікна користувачам надаються чотири інтерактивні кнопки, кожна з яких відповідає конкретному сценарію транзакції. Ці кнопки призначені не тільки для запуску анімації, але й для передачі чіткої послідовності операцій на рівні протоколу, з позначенням рівня та реєстрацією в режимі реального часу.

Перша кнопка, «Ініціалізувати з'єднання», імітує послідовність навчання DMI – процес, схожий на узгодження фізичного шару в реальному апаратному забезпеченні. Після натискання програма послідовно реєструє етапи пошуку партнерів, узгодження ширини смуги та конфігурації швидкості, що дає ефективну пропускну здатність 3,94 ГБ/с. Успішна ініціалізація активує інші кнопки керування та відображає статичний індикатор, що підтверджує встановлену швидкість і пропускну здатність. Це дозволяє користувачеві оцінити можливості інтерфейсу, одночасно визнаючи його верхню межу.

Після ініціалізації каналу, кнопка «Читання пам’яті» демонструє

операцію зчитування пам'яті. Ця симуляція є особливо детальною. Спочатку процесор надсилає пакет зчитування пам’яті до чіпсету, ініціюючи стандартну операцію зчитування. Далі відбувається ключовий навчальний момент – чіпсет виконує функціональне розвантаження на SSD, імітуючи вилучення функцій з пам'яті або обчислювальне розвантаження, як це відбувається в сучасних системних архітектурах. Це візуалізується зеленим пакетом «Дані функцій», який переміщується від SSD до чіпсету і, нарешті, назад до процесора. Цей процес демонструє, як обробка ближче до сховища може зменшити навантаження на канал DMI, що часто згадується в дослідженнях, присвячених високопродуктивним обчисленням і системним вузьким місцям [8].

Щоб ще більше підкреслити обмеження, кнопка «Симуляція Bottleneck’у» дозволяє користувачам моделювати сценарій, в якому великий обсяг передачі даних перевантажує пропускну здатність 3,94 Гб/с. Система реагує на це, анімуючи червоний пакет «Bottleneck» в обох напрямках, супроводжуючи його штучними затримками до двох секунд (лістинг 4.3), що символізують піки затримки через перевантаження. Ці затримки реєструються та чітко описуються на панелі стану, що дозволяє студентам зрозуміти, як навіть, здавалося б, високошвидкісні інтерфейси можуть стати обмеженими під навантаженням.

Лістинг 4.3 – Анімація руху пакету «Bottleneck» з затримкою в дві секунди (файл DMISimulationPage.xaml.cs)

```
await AnimatePacketAsync(415, 415, 160, 250, "Bottleneck",
"Red", true, 2000);
await AnimatePacketAsync(415, 415, 250, 160, "Bottleneck",
"Red", false, 2000);
```

Остання інтерактивна кнопка, що представлена у цьому вікні «Запис в пам’ять», виконує операцію запису в пам'ять. Фіолетовий пакет запису надсилається з процесора до чіпсету, а потім направляється до контролера пам'яті, завершуючи стандартну двосторонню транзакцію. Як і в разі інших

кнопок, активність на кожному рівні протоколу відображається за допомогою анімованого виділення: рівень транзакцій формує команду, рівень каналу передачі даних обробляє цілісність пакета, а фізичний рівень обробляє фактичну передачу.

Журнал стану симуляції функціонує як детальний трекер подій. Кожне натискання кнопки, анімація пакета та дія на рівні протоколу відображаються з часовою міткою, пропонуючи текстовий підсумок поточних процесів. Наприклад, під час циклу читання реєструються такі записи, як «Рівень транзакцій ЦП: формування читання пам'яті» або «Витяг функцій обробки чіпсету через SSD», що відображає те, як реальний стек інтерфейсу може реєструвати свої операції. Це підтримує як покрокове пояснення, так і аналіз після виконання для самостійних учнів.

Таблиця 4.1 – Інформація про використані пакети даних

Колір пакету	Назва пакету	Опис	Походження	Призначення
Жовтий	Memory Read	Ініціює запит на читання від процесора до пам'яті через чіпсет	ЦП	DDR4 через чіпсет
Зелений	Feature Data	Представляє оброблені дані, повернуті після вивантаження. Ініціює запит на читання	SSD	ЦП через чіпсет
Червоний	Bottleneck	Імітує перевантажений канал передачі даних через обмеження пропускної здатності	ЦП	DDR4 через чіпсет
Фіолетовий	Write to Memory	Транзакція запису в пам'ять	ЦП	DDR4 через чіпсет

Кожен тип пакета в симуляції має свій колір і роль, що також чітко визначено в легенді в нижньому лівому куті екрана (таблиця 4.1), що забезпечує інтуїтивне розуміння без необхідності запам'ятовування технічної термінології.

Візуалізація на рівні шарів, яка стала характерною особливістю всього додатка, також присутня і тут. У міру проходження кожного пакета його шлях через транзакційний, канальний і фізичний рівні позначається зміною кольору – чорне підсвічування сигналізує про активність. Ця абстрактна концепція, яка зазвичай використовується в підручниках і специфікаціях, стає зрозумілою для студентів завдяки візуальному зворотньому зв'язку.

Загалом, симуляція DMI досягає декількох педагогічних цілей. Вона ілюструє архітектуру загального системного інтерфейсу, підкреслює його характеристики продуктивності, моделює реальні обмеження, такі як обмеження пропускної здатності, та представляє сучасні концепції, такі як вивантаження функцій з SSD. У поєднанні з чіткими візуальними елементами, інтуїтивною інтерактивністю та журналом стану, додаток може служити як навчальним інструментом для студентів, так і допоміжним засобом для викладачів у поясненні складної поведінки шляху передачі даних у доступній формі.

4.6 Моделювання топології інтерфейсу South Bridge та її окремих елементів

Вікно симуляції South Bridge (рисунок 4.8) представляє візуально структуровану модель підсистеми Південного мосту в рамках традиційної архітектури чіпсету ПК. Основною метою цієї сторінки є не симуляція детальних механізмів протоколу безпосередньо в цьому центральному блоці, а контекстуалізація South Bridge в рамках більш широкої топології системи та підкреслення його ролі як хаба для повільніших периферійних інтерфейсів. Цей підхід має педагогічне обґрунтування: багато студентів-

початківців на початку навчання стикаються з такими термінами, як «South Bridge», не маючи чіткого уявлення про його фізичну роль або схему підключення. Щоб вирішити цю проблему, симуляція зосереджується на топологічній точності та ієрархічній природі конструкції чіпсету.

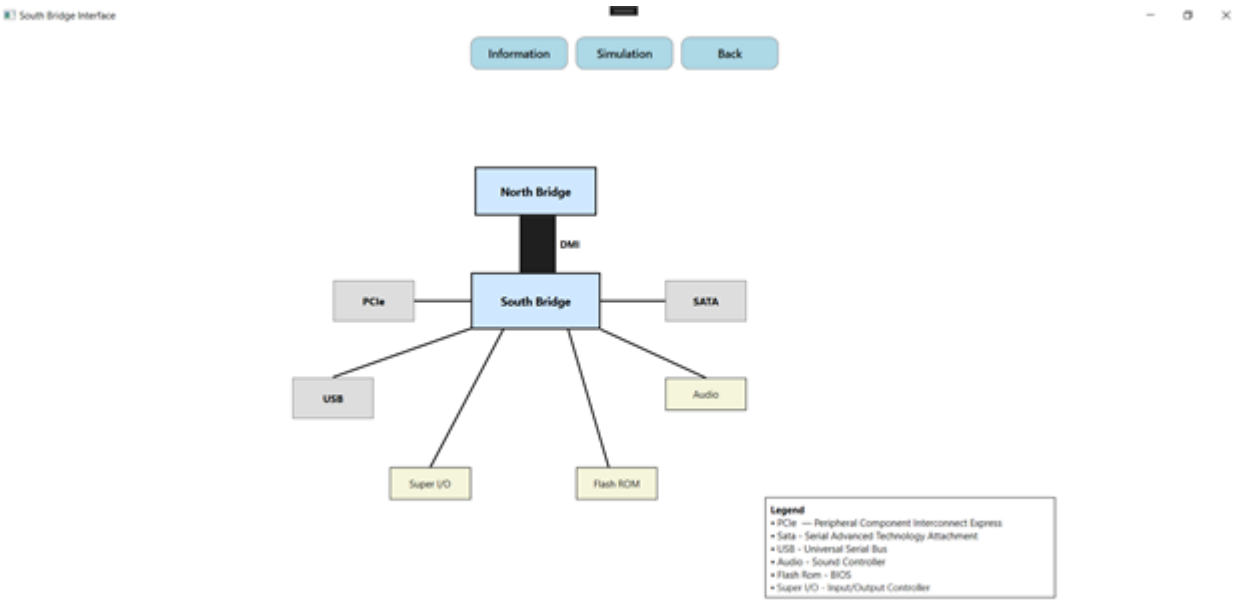


Рисунок 4.8 – Моделювання топології South Bridge

У центрі вікна розташований блок South Bridge, який виконує функцію візуального та логічного центру для підключених інтерфейсів. Прямо над ним розташований блок North Bridge, з'єднаний вертикальною лінією, яка символічно та візуально представляє Direct Media Interface – з'єднання з високою пропускною здатністю, яке з'єднує дві підсистеми. Лінія DMI виділяється своєю шириною та яскравим кольором, підкреслюючи важливість міжмостового з'єднання для продуктивності системи.

Від південного моста в усіх напрямках відходять кілька інших з’єднань, кожне з яких закінчується позначеним блоком, що представляє конкретний інтерфейс. До них належать PCI Express і SATA, USB і аудіо та Super I/O і BIOS/Flash ROM. Ця структура утворює деревоподібну топологію, що відповідає стилю, який використовується в усьому додатку, де ієрархічна зв'язність підкреслюється для кращого розуміння організації системи.

Хоча не всі інтерфейси є інтерактивними, два з них – PCIe, SATA та USB – виступають точками входу до більш детальних симуляцій. Вони представлені у вигляді кнопок у топології (лістинг 4.4) або як окремі елементи керування в інтерфейсі користувача.

Лістинг 4.4 – Реалізація елементу топології у вигляді натискаємої кнопки (файл SouthBridgeSimulationPage.xaml)

```
<Button Content="PCIe" Canvas.Left="190" Canvas.Top="170"
        Width="100" Height="50" FontWeight="Bold" FontSize="13"
        Click="PCIeButton_Click" Cursor="Hand"/>
```

Натиснення на кнопку PCIe відкриває вже реалізовану симуляцію PCIe x1, а кнопка SATA та USB відкривають нові спрощені моделі шин. Ця функціональність ілюструє, як South Bridge управляє декількома інтерфейсами і виступає в ролі контролера трафіку для даних, що надходять до периферійних пристроїв з нижчою швидкістю і від них. Крім того, елемент топології North Bridge теж функціонує як кнопка, що дозволяє користувачам переходити безпосередньо до вікна Північного мосту, підсилюючи архітектурний зв’язок між двома мостами.

Таким чином, це вікно відіграє важливу роль у поєднанні теоретичного розуміння функцій South Bridge та практичної реалізації периферійного моделювання. Воно також слугує центральним вузлом у навігації програми, забезпечуючи безперебійний доступ до більш спеціалізованих середовищ моделювання інтерфейсів, одночасно представляючи цілісну архітектурну картину положення South Bridge у комп'ютерній системі.

4.6.1 Моделювання роботи інтерфейсу SATA

Вікно моделювання SATA (рисунок 4.9) служить спрощеною візуальною демонстрацією інтерфейсу Serial ATA 3.0, підкреслюючи принцип високошвидкісного обміну даними між південним мостом і

пристроєм SATA. Візуальний дизайн відповідає загальній естетиці та послідовності макета програми, забезпечуючи звичний для користувача досвід.

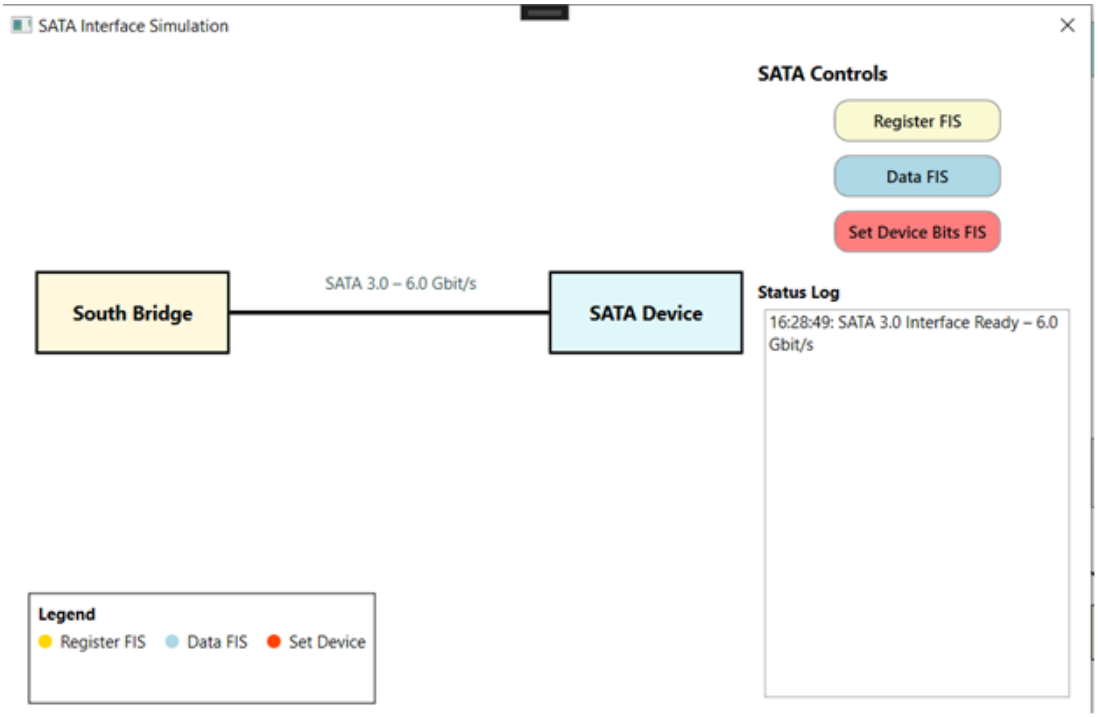


Рисунок 4.9 – Вікно моделювання інтерфейсу SATA 3.0

У лівій частині вікна Південний міст представлений у вигляді прямокутного блоку, з'єднаного з пристроєм SATA. Ця лінія символічно ілюструє з'єднання SATA 3.0. Моделювання зосереджено на анімації на рівні пакетів і відстеженні стану, що дає студентам уявлення про послідовний потік комунікації через шину SATA. Анімовані еліпси імітують пакети даних, що рухаються по каналу, кожен з яких супроводжується плаваючою міткою, що описує тип пакета та контекст передачі.

Панель керування праворуч екрана має три інтерактивні кнопки, кожна з яких імітує різну FIS – фундаментальну концепцію протоколу SATA. Кнопка «Ініціалізація FIS» ініціює типовий потік контрольних повідомлень, де South Bridge спочатку надсилає контрольний або командний сигнал до пристрою SATA. Після короткої затримки з пристрою SATA повертається другий пакет, що імітує відповідь про стан або контроль. Ця двофазна

взаємодія наочно демонструє структуру команд, ініційованих хостом, що є властивою для комунікацій SATA.

Наступна кнопка, «Передача FIS», запускає просту симуляцію передачі корисного навантаження даних. Після активації по лінії з'єднання анімується блакитний пакет, що ілюструє рух фактичних даних, який зазвичай відбувається у вигляді послідовностей імпульсів під час операцій читання або запису.

Остання кнопка, «Оновлення хосту FIS», зосереджена на сценаріях сигналізації про стан та черги вбудованих команд. Ця симуляція ілюструє, як пристрій SATA оновлює хост за допомогою прапорців або бітів стану, зазвичай для сигналізації про хід виконання або завершення поточної команди. Анімація відображає односторонній перенос даних від пристрою до хосту, підкріплюючи концепцію асинхронного повідомлення про стан у рамках структури SATA.

Кожна дія супроводжується детальними записами в журналі стану в правій панелі журналу, яка відстежує етапи моделювання в режимі реального часу, включаючи типи пакетів, напрямок і події обробки. Ці повідомлення містять часові мітки для наочності навчання, що допомагає студентам відстежувати часову послідовність транзакцій SATA.

Панель легенди розташована в нижньому лівому куті сторінки і пояснює кольорові коди пакетів, що використовуються в симуляції: жовтий колір позначає запити Register FIS, зелений – відповіді, блакитний – корисні дані FIS, а помаранчевий – біти стану пристрою. Ця легенда забезпечує швидкий довідник, що підсилює розуміння візуальних підказок.

Загалом, вікно симуляції SATA узагальнює принципи комунікації між хостом і пристроєм інтерфейсу SATA 3.0. Завдяки абстрагуванню складних електричних і часових рівнів протоколу в інтуїтивно зрозумілі анімовані взаємодії, воно надає студентам початкового рівня доступне і добре запам'ятовуване введення в цю широко використовувану шину зберігання даних.

4.6.2 Моделювання роботи інтерфейсу USB

Вікно моделювання USB (рисунок 4.10) забезпечує спрощене, але наочне моделювання комунікації USB 2.0 між контролером USB-хоста та пристроєм USB. Ця візуалізація зосереджується на обміні протоколами низького рівня в типових транзакціях USB, знайомлячи студента з поняттями токен-пакетів, пакетів даних та відповідей на рукостискання, які є основними для роботи шини USB.

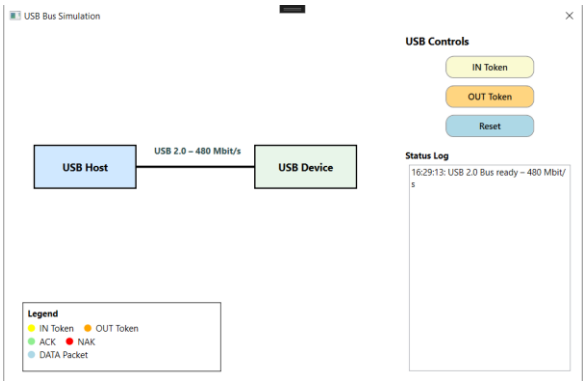


Рисунок 4.10 – Вікно моделювання інтерфейсу USB 2.0

Топологія навмисно мінімалістична: блок USB Host розміщений у лівій частині вікна, а блок USB Device вирівняний праворуч. Вони з'єднані лінією, що символізує шину USB 2.0, з позначкою «USB 2.0 – 480 Мбіт/с», що вказує на теоретичну максимальну пропускну здатність стандарту. Таке розташування чітко відображає характер USB як системи «головний-підлеглий», де хост завжди ініціює зв'язок, а пристрій відповідає відповідно.

В основі симуляції лежить набір з трьох кнопок управління, які імітують основні операції USB: IN Token, OUT Token і Reset. Кожна кнопка запускає заздалегідь визначену послідовність передач пакетів, які анімуються у вигляді кольорових еліпсів, що рухаються між хостом і пристроєм.

Натискання кнопки «IN токен» починає трифазний обмін, який імітує те, як хост запитує дані у пристрою. Жовтий пакет з позначкою «IN»

надходить від хоста до пристрою, ініціюючи запит. Потім пристрій відповідає блакитним пакетом «DATA», який рухається в протилежному напрямку. Нарешті, хост повертає зелений пакет «ACK», щоб підтвердити успішне отримання. Це відображає реальну поведінку USB під час опитування вводу – наприклад, коли USB-клавіатура надсилає натискання клавіші.

На відміну від цього, кнопка «OUT токен» моделює ситуацію, коли хост надсилає дані на пристрій. Помаранчевий пакет «OUT» анімується від хоста до пристрою, за яким одразу слідує блакитний пакет даних. У цій симуляції пристрій зайнятий або не може обробити запит, тому відповідає червоним пакетом «NAK». Це не тільки вводить поняття готовності пристрою, але й навчає користувачів, як USB реалізує логіку повторної спроби через відповіді на рукостискання.

Третя кнопка, «Очистка журналу», очищає журнал стану та скидає симуляцію для нового тесту. Журнал стану, розташований у правій частині інтерфейсу, фіксує кожен крок обміну в хронологічному порядку з часовими мітками, надаючи детальний аудиторський слід для освітнього аналізу та налагодження. Ці журнали підсилюють розуміння фаз USB-комунікації та підтримують причинно-наслідковий зв'язок між діями користувача та активністю шини.

У нижньому лівому куті сторінки також розміщена панель з кольоровими позначками. Вона ідентифікує всі типи візуальних пакетів: жовтий колір для токенів IN, помаранчевий для токенів OUT, блакитний для пакетів даних, зелений для відповідей ACK і червоний для відповідей NAK. Ця панель допомагає студентам швидко інтерпретувати анімацію під час більш складних транзакцій.

Це вікно моделювання є особливо цінним, оскільки воно розбиває протокол USB на зрозумілі одиниці, дозволяючи студентам спостерігати за обома напрямками передачі даних і розуміти ключові механізми узгодження. Воно візуально відображає різницю між передачами IN і OUT та знайомить із

базовою логікою відповіді пристрою. Абстрагуючись від електричних і часових складнощів протоколу USB, ця симуляція пропонує педагогічний інструмент, який висвітлює структуру та потік управління USB 2.0 у компактній і доступній формі.

4.7 Моделювання топології інтерфейсу North Bridge

Вікно моделювання Північного мосту (рисунок 4.11) демонструє важливий сегмент більш старих комп'ютерних архітектур, візуалізуючи взаємодію між ЦП, оперативною пам’ятю і Південним мостом через чіп North Bridge. Хоча сучасні системи інтегрують функціональність North Bridge в процесор, класична архітектура залишається важливою для навчальних цілей завдяки чіткості ілюстрації потоку даних по системних шинах. Ця симуляція розроблена, щоб зробити цей процес доступним і зрозумілим для студентів, які вперше стикаються з проєктуванням апаратного забезпечення та архітектурою комп'ютерів.

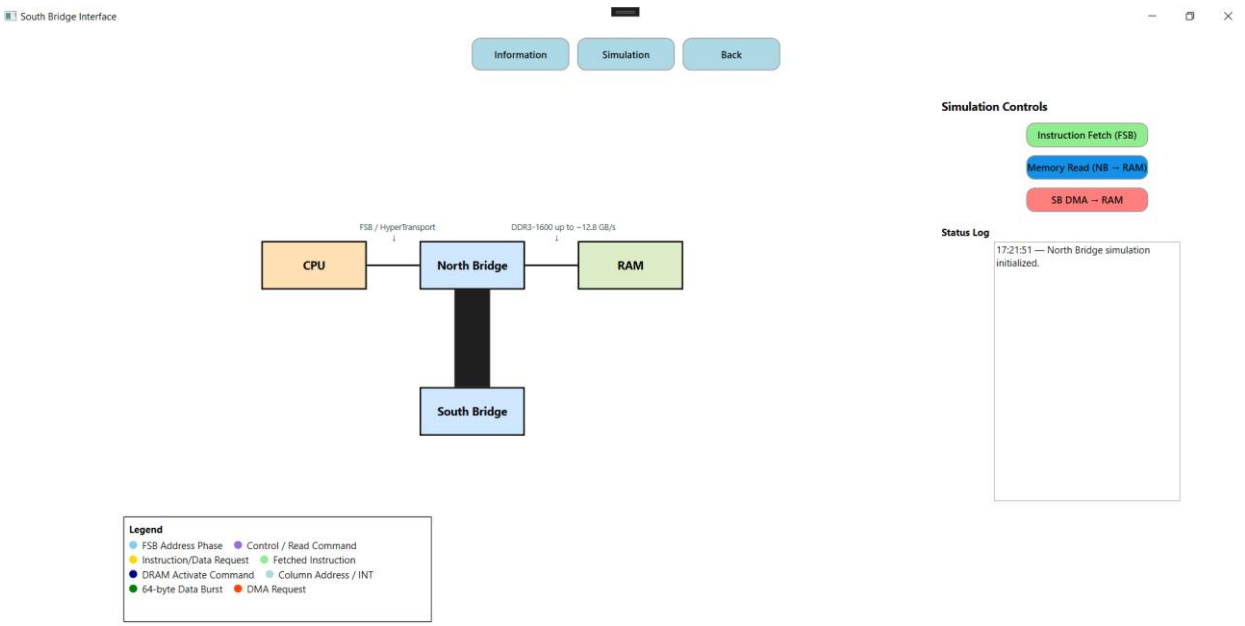


Рисунок 4.11 – Вікно топології та моделювання інтерфейсу North Bridge

Візуальне оформлення сторінки симуляції побудовано навколо центрального елемента – Північного мосту, представленого блоком з написом, розташованим у верхній центральній частині вікна. Ліворуч від нього знаходиться процесор, а праворуч – оперативна пам'ять. Під Північним мостом розташований Південний міст, з'єднаний вертикальним каналом, що відображає внутрішній канал з високою пропускною здатністю.

Ця проста, але точна топологія допомагає користувачам візуалізувати роль Північного мосту як контролера трафіку для високошвидкісного обміну даними, зокрема для доступу до пам'яті та делегування вводу-виводу.

У правій частині інтерфейсу вертикальна панель містить елементи керування моделюванням. Кожна кнопка на цій панелі представляє реальну транзакцію, що відбувається за участю Північного мосту, і запускає анімовану послідовність, яка імітує рух даних по системі. Перша кнопка, «Виконання команди», симулює повну транзакцію між ЦП та оперативною пам’ятю через North Bridge. Анімація починається з блакитного еліпса, що представляє фазу адреси, за яким слідує фіолетовий еліпс, що позначає фазу контролю, а потім золотий еліпс, що символізує фазу даних, яка запитує 64-байтову лінію кешу. Після короткої затримки світло-зелений еліпс повертається з північного мосту до процесора, щоб доставити отриману інструкцію. Ця послідовність наочно показує етапи доступу до інструкцій через передню шину, даючи уявлення про те, як затримка і структура шини впливають на продуктивність.

Наступна кнопка, «Читання пам’яті», показує операцію читання пам'яті, що походить від Північного моста до оперативної пам'яті. Послідовність включає активацію рядка DRAM, відправку адреси стовпця, видачу команди читання і отримання зеленого пакета даних – все це візуалізується за допомогою еліпсів, позначених кольорами. Ці кроки імітують чутливу до часу сигналізацію шини пам'яті DDR, включаючи використання RAS і CAS для вибору місць у пам'яті. Ця модель спрощує складний процес вилучення пам'яті, відображаючи його у вигляді лінійної

анімації з супровідними повідомленнями журналу стану.

Остання наявна кнопка, «ПМ DMA → RAM», вводить симуляцію запиту DMA, виданого Південним мостом. Цей сценарій є більш складним, оскільки передбачає надсилання Південним мостом запиту DMA вгору до Північного мосту через вертикальний канал. Анімація доповнена візуальними корегуваннями, які забезпечують читабельність міток, незважаючи на темний фон каналу. Після отримання запиту північний міст надсилає пакет для запису 4 КБ в оперативну пам'ять і, нарешті, відповідає процесору блакитним сигналом переривання. Ця взаємодія демонструє, як периферійні пристрої обходять процесор для прямого доступу до пам'яті, що є механізмом, який зазвичай використовується для оптимізації продуктивності в системах з інтенсивним введенням-виведенням.

Кожна з цих дій записується в журнал стану, велике текстове поле для читання під кнопками керування. Журнал фіксує кожен крок симуляції з часовими мітками, включаючи фази керування, видачу команд, відповіді пам'яті та сигнали підтвердження. Цей текстовий зворотний зв'язок доповнює візуальну анімацію, підкріплюючи те, що представляє кожен пакет і куди він переміщується.

Важливим компонентом цього вікна є легенда, розташована в нижньому лівому куті екрану. Вона надає візуальну довідку для інтерпретації пакетів, позначених кольорами, які використовуються протягом усього моделювання. Для підвищення зрозумілості та забезпечення послідовного розуміння анімованих елементів легенда представлена у вигляді таблиці 4.2, де кожен тип пакета пояснюється за допомогою його кольору, назви, короткого функціонального опису та напрямку руху – від джерела до пункту призначення.

Блок Південного мосту також є інтерактивним елементом. Користувачі можуть натиснути на нього, щоб безперешкодно перейти до вікна South Bridge. Така інтеграція забезпечує безперервність між симуляціями та підкреслює взаємопов'язаність компонентів системи. Вона також має

педагогічну цінність, допомагаючи учням зрозуміти архітектуру мосту як ієрархічну систему, в якій Північний міст відіграє центральну роль.

Таблиця 4.2 – Інформація про використані пакети даних

Колір пакету	Назва пакету	Опис	Походження	Призначення
Світло-блакитний	Фаза адресації	Містить адресу пам'яті	ЦП	Північний міст
Фіолето-вий	Команда читання або управління	Повідомляє про команду читання	ЦП	Північний міст
Золотий	Запит даних або інструкцій	Процесор запитує 64-б. лінію кеш-пам'яті даних	ЦП	Північний міст
Світло-зелений	Виконана інструкція	Результуючий 64-б. пакет даних	Північний міст	ЦП
Синій	Команда активації DRAM	Активація рядка сигналів в RAM	Північний міст	Оперативна пам’ять
Блакитний	Адреса стовпця	Вказує стовпець пам'яті	Пн/Пд мости	ОП/ЦП
Зелений	64-байтовий пакет даних	Основні дані, повернуті з оперативної пам'яті	Оперативна пам’ять	Північний міст
Помаранчевий	Запит DMA	Південний міст запитує доступ до пам'яті	Південний міст	Північний міст

Моделюючи реалістичні процеси, такі як виклик інструкцій, доступ до пам'яті та передача даних за допомогою DMA, це вікно відображає, як

Північний міст функціонував як критично важливий комунікаційний хаб у класичній архітектурі ПК. Кожен елемент – від кольору пакета до запису в журналі – спеціально розроблений для формування у студентів інтуїтивного розуміння того, як маршрутизуються та управляються дані процесора та периферійних пристроїв. Анімації, хоч і спрощені, відповідають структурі реальних протоколів на основі пакетів і слугують ефективним навчальним посібником як для викладачів, так і для учнів.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було виявлено проблему недостатньої візуалізації та сучасних навчальних засобів для вивчення інтерфейсів комп'ютерних систем. Багато студентів-початківців стикаються з проблемами розуміння того, як функціонують і взаємодіють в реальній системі різні інтерфейси ком’пютера. Традиційні матеріали часто не можуть чітко проілюструвати їх топологічне розміщення або функціональні ролі, залишаючи прогалини як у розумінні, так і в залученні.

Щоб вирішити цю проблему, було досліджено існуючі аналоги та обрано інтерфейси, які є релевантними та знайомими студентам. Метою було створення моделі, яка підтримує як самостійне навчання, так і використання викладачами. В результаті було розроблено додаток на основі WPF, який поєднує базову теоретичну інформацію з анімованими візуальними симуляціями, що дає можливість спостерігати та краще розуміти процеси обміну даними між компонентами системи.

Ця робота забезпечує міцний фундамент для майбутніх удосконалень. Додаток можна масштабувати і розширювати новими моделями та функціями, що в кінцевому підсумку сприятиме модернізації навчальних інструментів з комп'ютерної архітектури.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Мосіна Ю. THE ROLE OF VISUALIZATION IN TEACHING AND LEARNING PROCESS. Інноватика у вихованні. 2020. Т. 2, № 11. С. 203–208. URL: https://doi.org/10.35619/iiu.v2i11.212 (дата звернення: 27.05.2025).

2. Packet Tracer [Електронний ресурс] // Wikipedia – The Free Encyclopedia. – Режим доступу: https://en.wikipedia.org/wiki/Packet_Tracer – Дата звернення: 28.05.2025.

3. Human-Computer Interaction (3rd Edition) / A. Dix та ін. 3-тє вид. Prentice Hall, 2003. 834 с.

4. Designing the User Interface: Strategies for Effective Human – Computer Interaction (6th Edition) / Shneiderman B. та ін. 6-те вид. Pearson, 2016. 672 с.

5. Anderson D., Inc M., Shanley T. PCI System Architecture (4th Edition) (PC System Architecture Series). Addison-Wesley Professional, 1999. 832 с.

6. Всё, что вы хотели знать о PCI Express [Електронний ресурс] // ServerFlow.com – Режим доступу: https://serverflow.com/blog/stati/vse-chto-vy-khoteli-znat-pci-express/ – Дата звернення: 30.05.2025.

7. Эволюция PCI Express: от PCIe 1.0 до PCIe 6.0 [Електронний ресурс] // Fibermall.com – Режим доступу: https://www.fibermall.com/blog/pcie-development.htm – Дата звернення: 03.06.2025.

8. Jin G. Toward Fast and Scalable Transfer Learning with In – storage Processing. – Master’s Thesis. – Daegu Gyeongbuk Institute of Science and Technology (DGIST), Department of Information and Communication Engineering, 2021. – 36 с.

9. Lau J. H. Chiplet Communications (Bridges). *Flip Chip, Hybrid Bonding, Fan-In, and Fan-Out Technology*. Singapore, 2024. С. 427–469.

10. Посилання на GitHub репозиторій з вихіднимм кодом [Електронний ресурс] // GitHub.com – Режим доступу:

https://github.com/fastik822/DiplomaThesisProgram.git – Дата звернення: 17.06.2025.