МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп’ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Пістун Мар’яна Михайлівна

Verilog-модель FIFO пристрою комп'ютерної системи / The Verilog-model of FIFO computer system appliance

напрям підготовки: 6.050102 - Комп’ютерна інженерія
фахове спрямування - Комп’ютерні системи та мережі
Бакалаврська робота

Виконав студент групи КСМ-42/1
М.М. Пістун

Науковий керівник: к.т.н.,
Дубчак Л.О.

Тернопіль - 2018

РЕЗЮМЕ

Дипломний проект містить 56 сторінок пояснюючої записки, 9 рисунків, 7 таблиць, 2 додатки. Обсяг графічного матеріалу 2 аркуші формату А3.

Метою дипломного проекту є розроблення функціонального блоку пам'яті FIFO згідно його специфікації. Це було досягнуто шляхом проектування моделі FIFO. Правильність функціональності роботи моделі підтверджено тестуванням і аналізом реальних результатів сигналів симуляції.

Проаналізовано сучасні системи обробки інформації «Перший зайшов – перший вийшов». Проаналізувавши алгоритм роботи було вибрано середовище моделювання в якому буде розроблятися модель FIFO. Вслід за цим відбулося моделювання проектованого блоку FIFO. Опісля була здійснена верифікація розробленої моделі.

Розроблена модель FIFOмає свої як позитивні так і негативні сторони До переваг можна віднести те, що запропонований проект використовує ефективну структуру масивів пам'яті та може бути додатково модифікований для роботи в додатках, де існують декілька циклів затримки між виробником даних, FIFO, і споживачем даних.

Ключові слова: FIFO, VERILOG, БЛОК ПАМ'ЯТІ.

RESUME

The diploma project contains 56 pages of explanatory note, 9 figures, 7 tables, 2 appendices. Volume of graphic material 2 sheets of A3 format.

The purpose of the diploma project is to develop a functional memory block FIFO according to its specification. This was achieved by designing the FIFO model. The correctness of the functionality of the model is confirmed by testing and analysis of real results of simulation signals.

Modern information processing systems "First come in - first come out" are analyzed. After analyzing the algorithm, the simulation environment in which the FIFO model will be developed was chosen. This was followed by modeling of the designed FIFO unit. Subsequently, the developed model was verified.

The developed FIFO model has both positive and negative sides. The advantages include the fact that the proposed project uses an efficient structure of memory arrays and can be further modified to work in applications where there are several delay cycles between the data producer, FIFO, and the data consumer. .

Keywords: FIFO, VERILOG, MEMORY BLOCK.

ЗМІСТ

					ДП.КСМ.07142/14.00.00.000 ПЗ	
Зм	*Арк*	*№ докум.*	*Підпис*	*Дата*		
Розробив		Пістун М.М.			VERILOG-МОДЕЛЬ FIFOПРИСТРОЮ КОМП'ЮТЕРНОЇ СИСТЕМИ	*Літ.* / *Аркуш* / *Аркушів*
Перевірив		Дубчак Л.О.				
Консульт.		Паздрій І.Р.				ТНЕУ. ФКІТ. КСМ-42/1
Н. Контр.		Гураль І.В.				
Затв.		Березький О.М.				

ВСТУП

«Перший зайшов – перший вийшов» (FIFO) структури пам'яті широко використовуються для буферизації передачі даних між процесорними блоками. Все більше необхідні цифрові системи високої продуктивності та високої складності для передачі даних між модулями різних та навіть незв'язаних тактових частот.

FIFO - це черга оперативної пам'яті "First – in – First - Out" з логікою управління, яка керує операціями читання та запису, створює прапори стану та забезпечує додаткові сигнали рукостискання для взаємодії з логікою користувача. Він часто використовується для управління потоком даних між джерелом та призначенням.

В якості основної мови програмування використовується мова Verilog, яка дозволяє використовувати блочно орієнтований підхід програмування. Це значить, що модулі розбиваються на окремі блоки, всередині яких міститься код на мові Verilog.Ця мова дозволяє проектувати, верифікувати та реалізувати аналогові, цифрові та змішані електронні системи на різних рівнях абстракції.

В дипломній роботі представляється загальна характеристика мотивації та прийняття рішень для надійної та масштабованої архітектури FIFO.

Запропонований проект використовує ефективну структуру масивів пам'яті та може бути додатково модифікований для роботи в додатках, де існують декілька циклів затримки між виробником даних, FIFO, і споживачем даних. У цій дипломій роботі метою є розробка, перевірка та синтезування синхронної FIFO, що використовує бінарну кодовану вказівку для читання та запису для адреси масиву пам'яті.

1 СУЧАСНІ СИСТЕМИ ОБРОБКИ ІНФОРМАЦІЇ ТИПУ «ПЕРШИЙ ЗАЙШОВ – ПЕРШИЙ ВИЙШОВ»

1.1 Сучасні методи обробки інформації

У комп'ютерному програмуванні FIFO (first-in, first-out) - це підхід до обробки робочих запитів програм з черг або стеків, коли спочатку обробляється найстаріший запит. У апаратному забезпеченні - це або масив флопів або пам'ять «читання / запис», в якій зберігаються дані, надані з одного тактового домену, і на запит, з однаковими даними, відбувається опрацювання першого в логічному вході. Часовий домен, який постачає дані в FIFO, часто називають вхідною логікою, а часовий домен, який зчитує дані з FIFO, часто називається прочитаною або вихідною логікою. FIFO використовуються у конструкціях, щоб безпечно передавати декілька бітових слів даних від одного домена до іншого або керувати потоком даних між сторонами джерела та сторонами вибору, що знаходяться водному і тому ж домені (рисунок 1.1). Якщо часові домени для читання та запису регулюються одним і тим же тактовим сигналом, FIFO називається синхронним, і якщо домени читання і запису керуються різними (асинхронними) тактовими сигналами, FIFO вважається асинхронним.

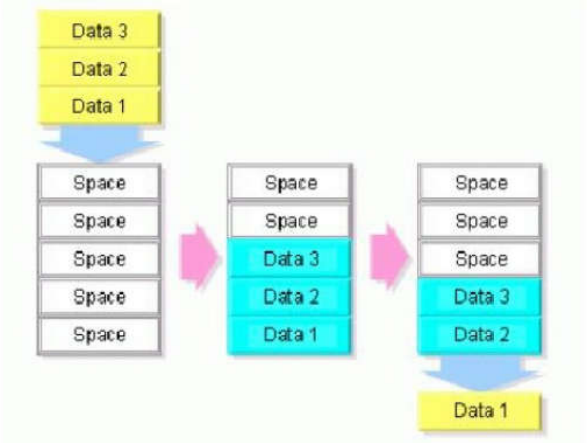

Рисунок 1.1 - Потоки даних через FIFO

FIFO - це черга оперативної пам'яті "First – in – First - Out" з логікою управління, яка керує операціями читання та запису, створює прапори стану та забезпечує додаткові сигнали рукостискання для взаємодії з логікою користувача. Він часто використовується для управління потоком даних між джерелом та призначенням.

FIFO може бути класифікований як синхронний або асинхронний в залежності від того, чи одночасні або різні (асинхронні) годинники контролюють операції читання та запису.

Загальна блок-діаграма FIFOзображена на рисунку 1.2.

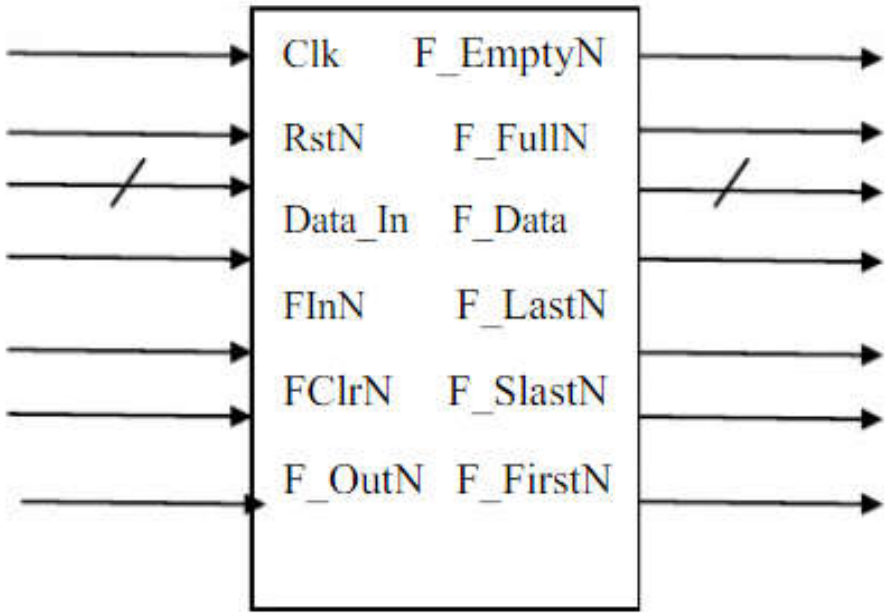

Рисунок 1.2 – Діаграма FIFO

Повні та порожні прапори FIFO генеруються і передаються логікам джерела та призначення, відповідно, для запобігання переповненню або переповненню даних. Таким чином зберігається цілісність даних між джерелом та призначенням.

Таблиця 1.1 - Розшифрування входів і виходів FIFO

Номер входу/виходу	Назва входу/виходу	Тип сигналу	Індикація
1	Clk	активний високий вхід	Тактовий сигнал забезпечується для синхронізації всіх операцій у FIFO.
2	RstN	активний низький вхід	Цей сигнал, коли заявлений низько, скидає FIFO. Усі лічильники встановлені на 0.
3	Data_In	32-бітний ввід	Це 32-розрядне введення даних в FIFO.
4	FInN	активний низький вхід	Запис сигналу в FIFO.
5	FClrN	активний низький вхід	Очищення сигналу з FIFO.
6	FOutN	активний низький вхід	Зчитування сигналу з FIFO.
7	F_EmtyN	активний низький вихід	Сигнал,який вказує на те, що FIFO порожній.
8	F_FullN	активний низький вихід	Сигнал вказує , що FIFO є повним.
9	F_LastN	активний низький вихід	Сигнал, який вказує, що FIFO має місце для одного останнього значення даних.
10	F_SlastN	активний низький вихід	Сигнал про те, що FIFO має місце для двох значень даних.
11	F_FirstN	активний низький вихід	Сигнал, який вказує, що в FIFO є лише одне значення даних.
12	F_Data	32-розрядний вихід	Це 32-розрядні дані, що виводяться з FIFO.

Більш ефективним способом побудови FIFO є створення кільцевого буфера з використанням масиву елементів пам'яті, створених таким чином, щоб дані могли бути записані та прочитані з довільних розташувань у масиві пам'яті. Ці структури також називаються паралельними FIFO і часто будуються з використанням загальних частин пам'яті SRAM або DRAM. Природний характер масивів пам'яті забезпечує низьку мінімальну затримку і високу ефективність у порівнянні з лінійними FIFO. Масштабованість також різко поліпшується, оскільки розмір годинника безпосередньо не впливає на розмір FIFO, а щільність пам'яті вище.

Синхронний FIFO відноситься до одного з виду FIFO, де значення даних записуються послідовно в масив пам'яті, використовуючи тактовий сигнал, а значення даних послідовно зчитуються з масиву пам'яті з використанням того ж тактового сигналу. У синхронному FIFO покоління порожні і повні прапори є прямими, оскільки немає перехресного домену. Враховуючи цей факт, користувач може навіть генерувати програмований частково порожній і частковий повний прапори, які потрібні в багатьох додатках.

Синхронний FIFO описує один із видів FIFO, в якому дані та інформація зберігаються в пам'яті та переміщують дані відповідним чином, використовуючи тактовий імпульс. Обидві операції читання та запису записують за схемою управління. У комп'ютерному програмуванні FIFO - це підхід до обробки запитів робочих програм з черг або стеків, щоб спочатку було оброблено найстаріший запит. У апаратному забезпеченні - це або масив флопів або пам'ять «читання/запис», що зберігає дані, надані з однієї доби годинника, і на запит із такими ж даними до іншого домену годинника, наступного за першим у вихідній логіці.

Фактично FIFO ділиться на дві категорії, такі як синхронний та асинхронний. У синхронному FIFO, запис операції в буфер FIFO і читання операції з одного і того ж буфера FIFO відбуваються в одному і тому ж

часовому домені. Але в асинхронному FIFO ці дві операції записують і читають відповідно із буферу FIFO в різні часові домени. Ясно, що домен годинника відрізняється в асинхронному FIFO. Операція запису відбувається в одному домені годинника, а операція читання перебуває в іншому часовому домені. В концепції FIFO існують обмеження для запису будь-яких даних і зчитування даних з пам'яті FIFO. Не можемо писати та читати дані без виконання певних умов, які називаються fifo_full і fifo_empty.

Асинхронний FIFO відноситься до виду FIFO, де значення даних записуються послідовно в буфер FIFO, використовуючи один часовий домен, а значення даних послідовно зчитуються з одного і того ж буфера FIFO за допомогою іншого годинника домену, де два тактові домени є асинхронними один до одного.

Однією з поширених методів проектування асинхронного FIFO є використання вказівників коду Grey, які синхронізовані в протилежний часовий домен, перш ніж створювати сигнали синхронного FIFO, повного або порожнього стану. Цікаво, що інший підхід до FIFO повного і порожнього покоління полягає в тому, щоб зробити асинхронне порівняння покажчиків і потім асинхронно встановлювати повні або порожні біти статусу.

1.2 Області застосування FIFO

Основними областями застосування пам'яті FIFO є пристрої з послідовними потоками паралельних даних (наприклад, цифрова відеокамера, мережевий концентратор та ін.). У складі цих пристроїв пам'ять FIFO може застосовуватися в якості буфера для прийому, накопичення і видачі за запитом накопичених даних. Наявність такого буфера дозволяє вивільнити ресурси

процесора від постійного сканування магістралі при прийомі повільних асинхронних потоків даних. Процесор в цьому випадку можна буде підключати до буферу FIFO тільки для вивантаження накопичених даних, користуючись інформацією про стан прапорів заповнювання.

Іншим прикладом застосування пам'яті FIFO є перетворення швидкості передачі даних в процесі взаємодії двох асинхронних пристроїв з різною пропускною здатністю. Для організації безперервного обміну потоками даних перетворенню підлягає також розрядність вхідний і вихідний шини даних.

У разі застосування RAM для вирішення аналогічних завдань потрібна наявність додаткової схеми управління, що представляє собою контролер прямого доступу до пам'яті. Це пов'язано з необхідністю формування адрес осередків в процесі виконання операцій запису і читання. Перевага в застосуванні пам'яті FIFO полягає в природному порядку проходження даних (першим надійшов - першим виводиться) та у відсутності зовнішньої схеми для формування адрес осередків.

В якості реальних прикладів застосування пам'яті FIFO можна навести такі: пристрій цифрової обробки і виведення на монітор сигналів, що надходять від рентгенівської установки; пристрій сполучення мультипроцесорного модуля на базі DSP з PCI шиною; кодек MPEG для телебачення високої чіткості.

У першому випадку мікросхеми "IDT72V21105, 256Kx18 FIFO" застосовуються для узгодження швидкості передачі даних між трьома асинхронними пристроями: джерелом цифрового рентгенівського сигналу, модулем цифрової обробки даних на базі декількох DSP процесорів і відеоконтрольного пристрою. Для забезпечення необхідної ширини пакета переданих даних мікросхеми FIFO включені в режимі нарощування інформаційної ємкості і утворюють модуль пам'яті з організацією 1,5Mx18.

У другому випадку дві односпрямовані мікросхеми "IDT72V36110, 128Kx36 FIFO" використовуються для організації двонаправленого обміну даними між 32-розрядних PCI-інтерфейсом з тактовою частотою 33 МГц і модулем, що складається з двох DSP процесорів з пропускною спроможністю шини даних 66 МГц і розрядністю 16 біт. Для підтримки обміну даними між 32-розрядної і 16-розрядної шинами використовується додаткова функція BUS MATCHING.

У кодеку MPEG-2 для телебачення високої чіткості вдалим рішенням є використання пам'яті FIFO і, зокрема, "IDT72V2113, 512Kx9". Стандарт MPEG-2 передбачає можливість гнучкого зміни швидкості передачі відеоданих в дуже широких межах, а також роботи як з чергуванням, так і з прогресивної розгортки, при частоті полів 50 або 60 Гц.

Отже, мікросхеми пам'яті FIFO можна рекомендувати до застосування:

– в якості буфера обміну даними між швидкодіючим процесором і більш повільними периферійними пристроями;

– для узгодження пристроїв з різною швидкістю передачі даних;

– для організації обміну даними між шинами з різним форматом слова.

1.3 Постановка задачі та аналіз дерева рішень

В даному дипломному проекті поставлена задача розробки Verilog–моделі FIFOпристрою комп’ютерної системи. Процес проектування складається з таких етапів, котрі зображені на рисунку 1.3.

Все більше необхідні цифрові системи високої продуктивності та високої складності для передачі даних між модулями різних та навіть незв'язаних тактових частот. Цей проект представляє загальний опис мотивації

та прийняття рішень для надійної та масштабованої архітектури FIFO. Запропонований дизайн використовує ефективну структуру масивів пам'яті та може бути додатково модифікована для роботи в програмах, де існують декілька циклів затримки між виробником даних, FIFO, і споживач даних. У цьому проекті метою є розробка, перевірка та синтезування синхронної FIFO, що використовує бінарне кодування з ознаками для читання та запису для адреси масиву пам'яті.

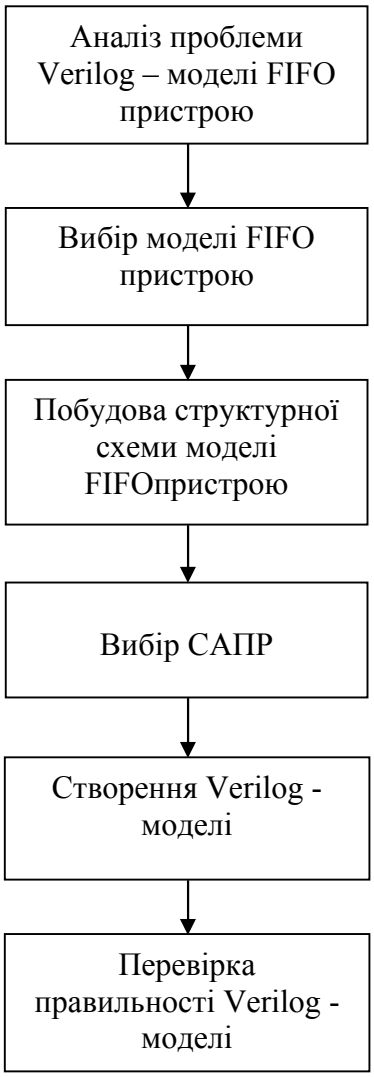

Рисунок 1.3 – Дерево рішень дипломного проектування

Дана компонента розробляється в середовищі Active-HDL. Програма дозволяє проектувати пристрій за допомогою мов опису апаратури, а також за допомогою структурних схем. Спочатку програма підтримувала тільки мову VHDL, але з часом додалася підтримка мов Verilog і SystemC. В якості основної мови програмування використовується мова Verilog, яка дозволяє використовувати блочно орієнтований підхід програмування. Це значить, що модулі розбиваються на окремі блоки, всередині яких міститься код на мові Verilog.

В якості реальних прикладів застосування пам'яті FIFO можна навести такі: пристрій цифрової обробки і виведення на монітор сигналів, що надходять від рентгенівської установки; пристрій сполучення мультипроцесорного модуля на базі DSP з PCI шиною; кодек MPEG для телебачення високої чіткості. Тому застосувати цю розробку можна в задачах даного типу.

Спочаткудля більшого розуміння опишемо подане дерево рішень дипломного проектування зображеного на рисунку 1.3. Першим кроком є – загальний аналіз проблеми створення Verilog– моделі FIFOпристрою. Наступне, що слідує, так це вибір моделі FIFO пристрою, у нашому випадку це модель FIFO пристрою розроблений на мові проектування Verilog. Третім кроком дипломного проектування є завдання побудувати структуровану схему моделі FIFO пристрою. Після цього вибираємо відповідну систему автоматизованого проектування. П’ятим кроком є створення Verilog – моделі FIFO пристрою комп’ютерної системи. Заключним етапом, зображеним на дереві рішень дипломного проектування є перевірка правильності роботи поданої Verilog – моделі.

2 АЛГОРИТМ РОБОТИ ПРОЕКТОВАНОГО БЛОКУ FIFO

2.1 Загальний алгоритм роботи FIFO

На рисунку 2.1 зображені основні конструктивні блоки синхронного FIFO: масив пам'яті, логіка управління записом і читання логіки управління. Масив пам'яті може бути реалізований як масив тригерів, так і пам'ять для читання/запису з подвійним портом.

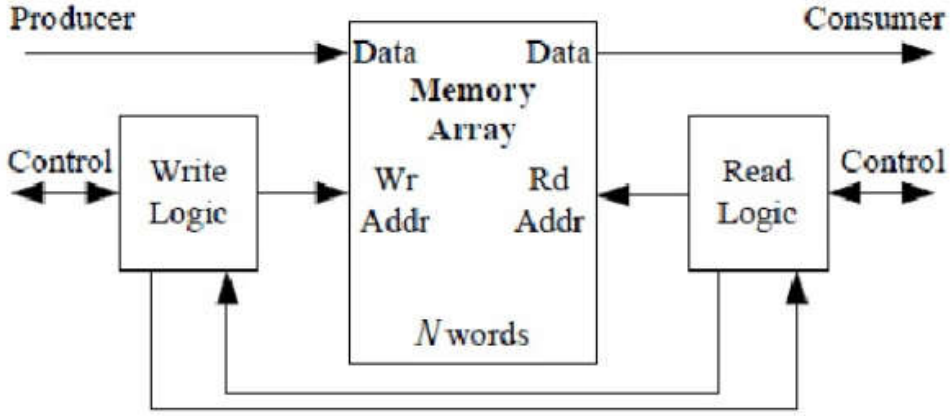

Рисунок 2.1 –Блок-схема FIFO з логікою керування записом та читанням

Обидві ці реалізації дозволяють одночасно зчитувати і записувати звернення. Цей одночасний доступ дає FIFO власну властивість синхронізації. Немає обмежень щодо часу між доступом двох портів. Це просто означає, що, хоча один з портів записує в пам'ять з однією швидкістю, інший порт може бути прочитаний за іншою швидкістю, повністю незалежною один від одного. Єдине обмеження полягає в тому, що одночасний доступ до читання та запису не повинен перебувати з/в тій же самій пам'яті (стан потоку)[3].

Синхронний FIFO має один тактовий порт Clk для операцій зчитування даних і запису даних. Дані, представлені на порті введення даних модуля Data_In, записуються в наступну вільну область пам'яті на піднімаючому краю

годинника при введенні дозволу на запис FinN високий. Повне виведення статусу F_FullN вказує, що у внутрішній пам'яті модуля не залишилося більше вільних місць. Дані можна зчитувати з FIFO через порт даних - вихід модуля F_Data в тому порядку, в якому воно було написано шляхом затвердження сигналу читання F_OutN перед підвищенням краю годинника. Вихід із порожнього стану пам'яті F_EmptyN вказує на те, що у внутрішній пам'яті модуля більше немає даних.

Варто зазначити, що контрольні блоки для читання та запису в одно часових FIFO поділяють загальний годинник, тому їх розділення являє собою функціональний поділ, не обов'язково фізичний. Схема запису контролює дані, що записуються у FIFO, і перевіряє чи пам'ять заповнена. Замкнутість читання контролює потік даних з FIFO і стосується визначення часу, коли пам'ять стає порожньою.

Керування схемами для кругових FIFO є більш складними, ніж схеми керування для лінійних FIFO через необхідність керування нестандартним доступом до запису та читання з особливими випадками, коли масив заповнений або порожній [4].

Запис логіки контролю використовується для керування операцією запису FIFO внутрішня пам'ять. Він генерує бінарним кодуванням курсор запису, який вказує на місце пам'яті, де потрібно вписати вхідні дані. Покажчик запису збільшується на один після кожної успішної операції записування. Окрім того, він генерує FIFO повну F_FullN, F_LastN (лише єдиний проміжний інтерфейс), F_SlastN (2 залишки), які, у свою чергу, використовуються для запобігання втраті даних. Наприклад, якщо запит на запис надходить після завершення FIFO. Записування логіки контролю стирає запис у пам'яті до часу F_FullN і прапорець стає затвердженим. Це означає припинення запису до джерела, не надсилаючи підтвердження у відповідь на запит.

Зчитування логіки управління використовується для керування операцією читання FIFO внутрішньої пам'яті. Він генерує вказівник для зчитування з двозарядним кодом, який вказує на місце пам'яті, звідки ці дані слід зчитати.

Показник зчитування збільшується на один після кожної успішної операції читання. Окрім того, він створює порожній FIFO і лише одне значення даних у FIFO, прапори, які, у свою чергу, використовуються для запобігання зчитування будь-яких неправильних даних. Наприклад, якщо запит на читання надходить, коли FIFO порожній, то зчитування логіки управління стирає читання з пам'яті до часу і F_EmptyN прапор стає затвердженим.

Масив пам'яті - це масив тригерів, що зберігає дані. Кількість слів даних, які може зберігати масив пам'яті, часто називають DEPTH FIFO. Довжина слова даних називається WIDTH FIFO. Крім того, флоп-масив включає в себе логіку розшифрування адреси читання та запису [6].

Функціональність масиву пам'яті відносно проста:

1.Якщо запис можливий, то сигнал є високими даними, які присутні на даних запису, вписується в ряд, вказаний адресою запису на наступному зростаючому краю синхронізуючого сигналу clk (рисунок 2.2). Запис можливий тільки, коли дійсний wdata є високим і FIFO не повний, щоб уникнути будь-якого порушення цілісності даних.

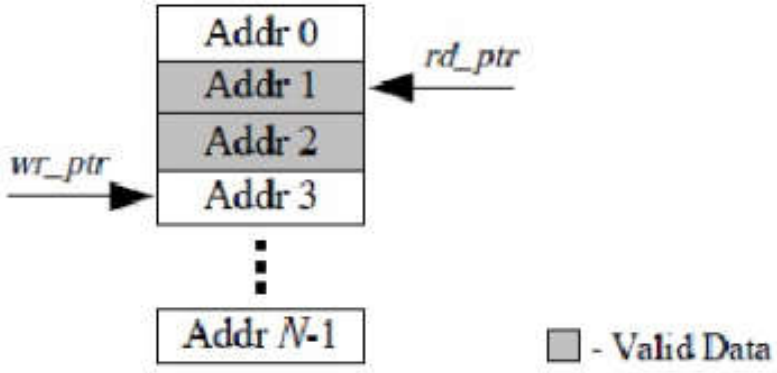

Рисунок 2.2 – Схема запису і зчитування ареси показчика

2.Якщо сигнал зчитування високий, дані, присутні в рядку, позначеному Read addr, надсилаються до шини даних для читання на наступному зростаючому краю тактового сигналу clk. Read enable підтверджується лише тоді, коли read req є високим, і FIFO не пустий, щоб уникнути передачі прохачу будь-яких неправдивих даних [8].

3. Він може обробляти одночасне читання та запис у режимі дозволу, якщо їх адреси не збігаються.

Всі паралельні FIFO повинні стежити за трьома взаємовиключними станами: 1) порожнім (також початковим станом) (рисунок 2.3); 2) повним; 3) заповненням даних між порожнім і повним.

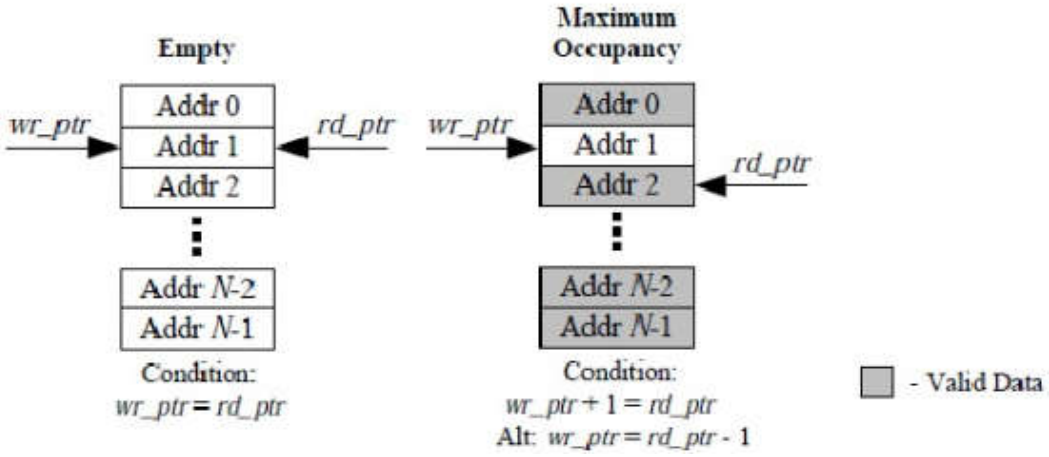

Рисунок 2.3 – FIFO визначається як порожній, якщо wr_ptr = rd_ptr

Відстеження дійсних даних у межах FIFO зазвичай здійснюється за допомогою одного з двох підходів. Перший спосіб передбачає використання N-бітного регістру (лічильника), де кожен біт являє собою дійсність даних у пам'яті, а N - кількість слів у пам'яті. Другий спосіб полягає в підтримці вказівників для читання та запису (або голови та хвоста), які вказують початок і кінець дійсного діапазону даних у пам'яті. На рисунку 2.4 показана схема, в

якій вказівник адреси запису (wr ptr) вказує місце розташування пам'яті для запису наступних даних, а вказівник для читання (rd ptr) вказує розташування наступної читаної пам'яті. Можливі інші схеми, найпоширеніші з яких зрівняли свої покажчики на одній пам'яті з наведеного тут прикладу [9].

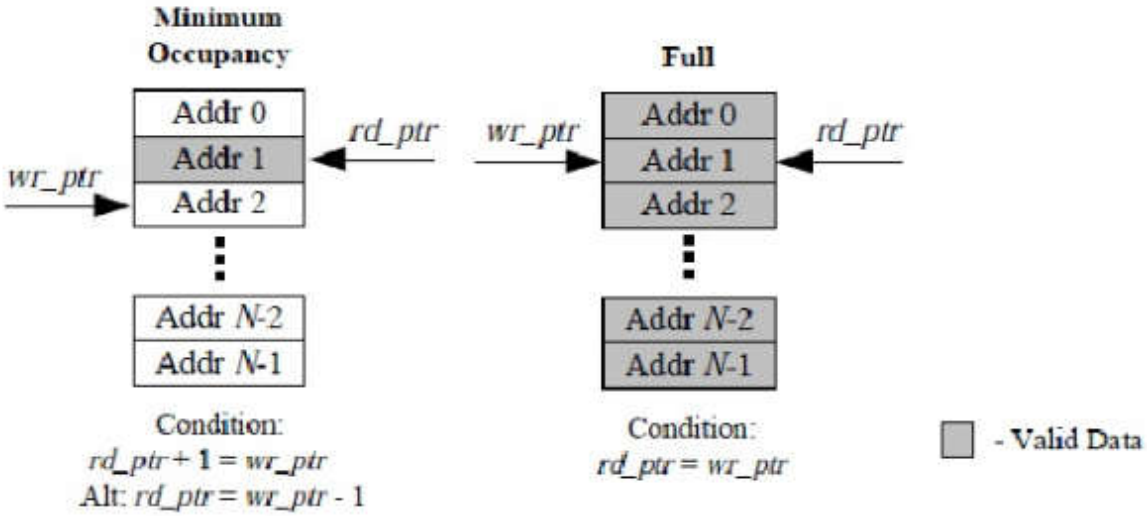

Рисунок 2.4 – FIFO визначено як повне, коли rd_ptr = wr_ptr

Враховуючи використання вказівників для читання та запису, існує два основних способи визначення порожніх і повних умов. Як показано на рисунку 2.3, перша можливість полягає у визначенні порожнього стану як такої, що відбувається, коли wr ptr дорівнює rd ptr. Стан "максимального заселення" (повний) вказується, коли wr ptr + 1 = rd ptr, або, коли ж wr ptr = rd ptr -1. Другий випадок показано на рисунку 2.4, де повна умова позначається рівністю rd ptrand wr ptr та "мінімальна заповнюваність" (один базисний) стан позначається rd ptr = wr ptr +1. Зрозуміло, що ці схеми виявляють проблеми у представленні всіх можливих станів, тому що випадки, коли rd ptr = wr ptr стає неоднозначним, якщо не відстежувати історію вказівника або не дозволяти покажчикам досягти цього стану в будь-якому повному або порожньому стані, як зазначено вище.

2.2 Мова проектування Verilog

Verilog HDL - це мова опису обладнання, що використовується для проектування та документування електронних систем. Verilog HDL дозволяє дизайнерам розробляти на різних рівнях абстракції.

Ця мова (також відомий як просто Verilog) дозволяє проектувати, верифікувати та реалізувати аналогові, цифрові та змішані електронні системи на різних рівнях абстракції.

Verilog може використовуватись для опису цифрового обладнання на трьох різних рівнях абстракції:

1) Поведінковий рівень описує як обладнання повинно вести себе без будь-якого посилання на цифрове апаратне забезпечення.

2) Register-Transfer-Level (RTL) - тут опис передбачає існування регістрів, і вони синхронізуються за тактовим сигналом. Тому цифрові дані передаються з одного регістра в інший на наступні тактові цикли.

3) Gate Level - це найнижчий рівень опису, де кожен з виходів і їх взаємозв'язок чітко визначені.

Verilog - це не тільки специфікація, яка говорить про систему CAD. Припустимо, апаратне забезпечення також включає в себе повне моделювання середовища. А компілятор Verilog робить більше, ніж відображення вашого коду на апаратне забезпечення, він також може імітувати (або виконувати) необхідний дизайн, щоб передбачити поведінку схеми. Ця мова проектування переважно використовується для розробки чіпів.

Verilog HDL виникла в автоматизованих системах інтегрованого проектування (пізніше перейменована в автоматизоване проектування шлюзів) в 1985 році. Компанія була приватною власністю в той час доктором Прабху Гоелем, винахідником алгоритму генерації тестів PODEM. Verilog HDL був

розроблений Філом Мурбі, який пізніше став головним дизайнером Verilog-XL та першим корпоративним співробітником в Cadence Design Systems. Автоматизація дизайну шлюзу швидко зростала з успіхом Verilog-XL і була нарешті придбана компанією Cadence Design Systems, Сан-Хосе, CA у 1989 році.

Verilog була винайдена як симуляційна мова. Наприкінці 1980-х рр. здавалося очевидним, що дизайнери збираються віддалятися від власницьких мов, таких як n dot, HiLo та Verilog, до стандарту US Defense Departament of Defense H.D.L., відома як мова опису апаратного забезпечення VHSIC. Сам VHSIC означає "Дуже високошвидкісний інтегральний пристрій".

Можливо, завдяки такому тиску на ринку Cadence Design Systems вирішило відкрити мову Verilog для публіки в 1990 році, і таким чином народився OVI (Open Verilog International). До цього часу Verilog HDL був власною мовою, що є власністю Cadence Design Systems. Коли OVI було створено в 1991 році, ряд невеликих компаній почали працювати на тренажерах Verilog, включаючи хронологічне моделювання, Frontline Design Automation та інші. Перший з них вийшов на ринок у 1992 році, і зараз існують зрілі симулятори Verilog, доступні з багатьох джерел.

Як наслідок, ринок Verilog значно зріс. Ринок Verilog-сумісних інструментів у 1994 році склав понад 75 мільйонів доларів, що робить його найбільш комерційно значимим на ринку описом мови опису обладнання.

Робоча група IEEE була створена в 1993 році під підкомітетом Автоматизації проектування для виробництва стандарту IEEE Verilog 1364. Verilog став стандартом IEEE 1364 у 1995 році.

Як міжнародний стандарт, ринок Verilog продовжував зростати. У 1998 році ринок тільки для симуляторів Verilog перевищив 150 000 000 доларів; продовжуючи своє панування.

Робоча група IEEE випустила переглянутий стандарт у березні 2002 року, відомий як IEEE 1364-2001. Суттєві помилки публікації затьмарили цей випуск, і виправлена версія була випущена в 2003 році, відома як IEEE 1364-2001, версія C.

Згодом була створена нова робоча група, IEEE P1800, яка побудувала мову IEEE 1364 разом із додатковими внесками від Accellera. В середині 2004 р. комітет IEEE 1364 був розформований, а робота групи IEEE 1800 була використана для підтримки стандарту.

Verilog має простий синтаксис, схожий з мовою програмування C. Мала кількість службових слів і простота основних конструкцій спрощують вивчення і дозволяють створювати ефективні програми. На опис однієї і тієї ж конструкції в Verilog потрібно в 3-4 рази менше символів, ніж в VHDL.

Verilog має конструкції для опису елементів апаратури в короткій і зрозумілій формі. Аналогічне опис на мові VHDL може бути в два рази довше.

Використовуючи Verilog, розробник повинен вивчити тільки одну мову для всіх аспектів логічного проектування. Моделювання вимагає щонайменше функціональних моделей, ієрархічних структур, тестових векторів і інтерактивної взаємодії людини і машини. На Verilog це все досягається на одній мові. Майже будь-який оператор, який можна написати в програмі, може бути виконаний з терміналу.

Цікавою особливістю Verilog є наявність стандарту PLI (Program Language Interface), який дозволяє включати функції, написані користувачем (наприклад, на C), в код симулятора.

Перевагою використання мови Verilog є зручність і простота написання програмного коду і його подальшого налагодження. Після створення блоку і написання програмного коду результат буде представлений у вигляді блоку з входами і виходами.

Ініціалізація входів і виходів блоку на мові Verilog зручніше в порівнянні з іншими мовами опису апаратури.

2.3 Алгоритм роботи проектованого FIFO

Алгоритм роботи проектованого FIFO (ДП.КСМ. 07142/14.00.00.000 А2) здійснюється за наступною послідовністю.

1. Встановлюємо значення для FWIDTH (ширина FIFO) – 32, FDEPTH (глибина FIFO) – 4, FCWIDTH (лічильник ширини FIFO) – 2. Також ініціалізуємо регістри Clk, RstN, Data_In, FClrN, FinN, FoutN як входи та F_Data, F_FullN, F_EmptyN, F_FirstN, F_SlastN, F_LastN як виходи.

2. Вводимо всі вхідні значення.

3. Поки виконується умова при якій позитивний фронт тактового сигналу частоти Clk або негативний фронт тактового сигналу частоти RstNє активними виконуємо перевірки. Якщо умова не виконується– завершуємо роботу.

4. Перевіряємо наступні умови виконання алгоритму.

4.1 Якщо виконується умова, що RstN або FClrN не є дійсними, тоді присвоюємо fcounter = 0, rd_ptr = 0, wr_ptr = 0. Після чого переходимо до пункту 5. Якщо ця умова не виконується переходимо до пункту 4.2.

4.2 Якщо виконується умова, що FoutN не є дійсним і F_EmptyN - дійсний, тоді присвоюємо F_Data = FIFO [rd_ptr], збільшуємо rd_ptr на 1, а fcounter - зменшуємо на 1. Після чого переходимо до пункту 5. Якщо ця умова не виконується переходимо до пункту 4.3.

4.3 Якщо виконується умова, що FinN не є дійсним і F_FullN - дійсний, тоді присвоюємо FIFO [wr_ptr] = Data_In, збільшуємо Wr_ptr, та

fcounter на 1. Після чого переходимо до пункту 5. Якщо ця умова не виконується переходимо до пункту 4.4.

4.4 Якщо виконується умова, що FoutN не є дійснимі F_InN - дійсний, тоді присвоюємо FIFO [wr_ptr] = Data_In, та F_Data = FIFO [rd_ptr], збільшуємо wr_ptr, та rd_ptr на 1. Після чого переходимо до пункту 5.

5. Перевіряємо наступні умови роботи алгоритму.

5.1 Якщо виконується умова, що fcounter рівний FDEPTH, тоді присвоюємо F_FullN = 0 та F_lastN = F_SlastN = F_EmptyN = F_FirstN = 1. Після того переходимо до пункту 6. Якщо ця умова не виконується переходимо до пункту 5.2.

5.2 Якщо виконується умова, що fcounter рівний нулю, тоді присвоюємо F_FullN = F_lastN = F_SlastN = F_FirstN = 1 та F_EmptyN = 0. Після того переходимо до пункту 6. Якщо ця умова не виконується переходимо до пункту 5.3.

5.3 Якщо виконується умова, що fcounter рівний 1, тоді присвоюємо F_FullN = F_lastN = F_EmptyN = F_SlastN = 1 та F_FirstN = 0. Після того переходимо до пункту 6. Якщо ця умова не виконується переходимо до пункту 5.4.

5.4 Якщо виконується умова, що fcounter рівний 2, тоді присвоюємо F_FullN = F_lastN = F_EmptyN = F_FirstN = 1 та F_SlastN = 0. Після того переходимо до пункту 6. Якщо ця умова не виконується переходимо до пункту 5.5.

5.5 Якщо виконується умова, що fcounter рівний 3, тоді присвоюємо F_FullN = F_SlastN = F_EmptyN = F_FirstN = 1 і F_LastN = 0. Після того переходимо до пункту 6, якщо ні – вихід.

6. Відбувається оновлення всіх значень вихідного сигналу. Далі переходимо до пункту 7.

7. Якщо дані підходять під умови циклу проходимо заново перевірку починаючи з пункту 4. Раптом дані не відповідають умовам перевірки, тоді – вихід.

3 МОДЕЛЮВАННЯ І ВЕРИФІКАЦІЯ МОДЕЛІ FIFO

3.1 Середовище моделювання

Єдине графічне середовище налагодження в Cadence Incisive Enterprise Simulator, Cadence SimVision Debug підтримує рівні сигналу та транзакції на основі всіх стандартів IEEE-стандартів, тестових технологій та мов опису. Він також підтримує одночасну візуалізацію обладнання, програмного забезпечення та аналогових доменів.

SimVision Debug може використовуватися для налагодження цифрових, аналогових або змішаних сигналів, написаних на мовах Verilog, SystemVerilog, VHDL та SystemC, або їх поєднання. Інтегроване налагодження в SimVision підтримує рівні сигналів і транзакцій за допомогою всіх стандартів IEEE-стандарту, на базі тестів та мов підтвердження, а також для одночасної візуалізації обладнання, програмного забезпечення та аналогових доменів.

Налагодження SimVision забезпечує уніфіковане середовище моделювання та налагодження, яке дозволяє Incisive Enterprise Simulator легко керувати кількома моделями і аналізувати як поведінку дизайну, так і тестовий стенд у будь-який момент процесу перевірки, незалежно від композиції.

Протягом потоків проектування та верифікації SimVision Debug забезпечує аналіз посилань на джерела перегляду, транзакцій та змішаних сигналів, повний аналіз покриття коду / транзакції / атестації, інтегровані відображення та налагодження енергетичної поведінки, перевірки аналізу апаратного забезпечення та безперервного підключення до потоків впровадження вниз. API на основі галузевих стандартів доступні на всіх рівнях, щоб забезпечити чітко визначені користувачем перевірки та аналіз. Моделі проектування та тестування можуть бути перекладені на будь-яку

мову та на будь-якому рівні абстракції без продуктивності та накладних витрат на інтеграцію, спричинених спільним моделюванням.

Налагодження SimVision містить кілька аналітичних вікон для вирішення складності налагодження. Деякі з цих вікон доступні у вигляді кнопок панелі інструментів та вибору меню.

Вікно Властивості, яке дає змогу керувати курсорами, маркерів, виразами та іншими об'єктами налагодження, які ви створили під час сеансу налагодження SimVision.

Вікно "Дизайн браузера", яке дозволяє відслідковувати сигнали та змінні в дизайні.

Вікно хвильової форми, яке складається з даних про моделювання уздовж осей*X* та *Y*. Дані зазвичай відображаються як значення сигналу у порівнянні з часом, але це можуть бути будь-які записані дані.

Schematic Tracer, який відображає дизайн як схему, і дозволяє простежити сигнал через дизайн.

Вікно перегляду, яке дозволяє відстежувати вибрані сигнали та змінні в дизайні у стислішій формі, ніж браузер Design.

Вікно "Реєстр", яке дозволяє вам використовувати графічний редактор вільної форми для визначення будь-якої кількості сторінок реєстру, кожна з яких містить спеціальне представлення даних моделювання.

Калькулятор виразів дозволяє визначати вирази, які поєднують сигнали з формуванням автобусів, умов та віртуальних сигналів.

Програма Memory Viewer дозволяє помітити зміни у внутрішньому стані пам'яті. Під час моделювання це також дозволяє встановлювати точки зупинки, а також сили та депозитні значення в місцях пам'яті.

Вигляд вікна з запущеним проектом зображено на рисунку 3.1.

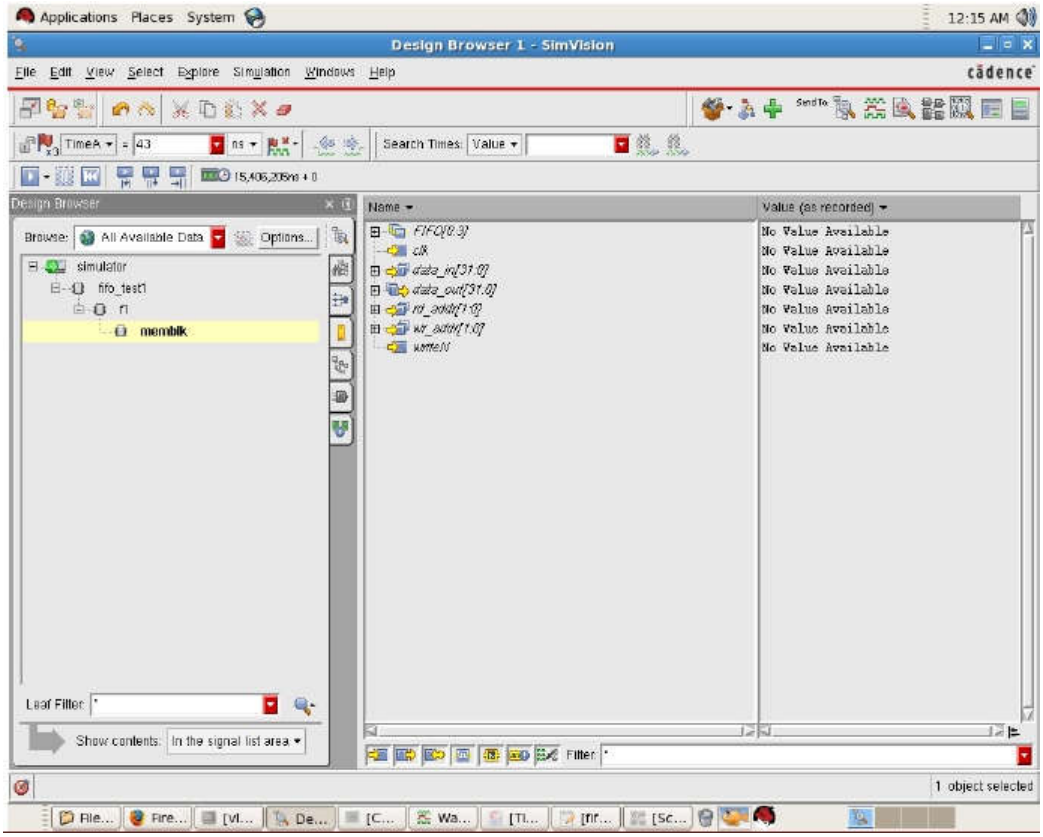

Рисунок 3.1 – Вигляд вікна з запущеним проектом

3.2 Моделювання

Архітектура системи з використанням модуля FIFO для зв’язку зображено на рисунку 3.2.

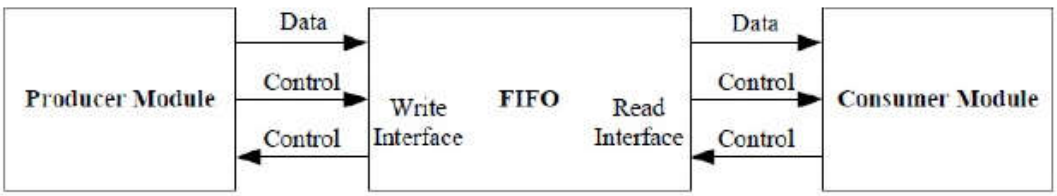

Рисунок 3.2 – Архітектура системи з використанням модуля FIFO для зв’язку

Модуль FIFO для зв'язку. FIFO використовуються для надійного передавання даних між двома асинхронними доменами. У конструкціях System-on-Chip є компоненти, які часто працюють на різних годинниках. Отже, для передачі даних з одного такого компоненту в інший потрібен ASYNCHRONOUS FIFO. Деколи, навіть якщо сторони джерела та запитувача керуються тим самим тактовим сигналом, потрібна FIFO. Це збігається з пропускною спроможністю джерела та запиту. Наприклад, в одному випадку джерело може надавати дані за курсом, який запитувач не може обробити, або в іншому випадку запитувач може розміщувати запити на дані зі швидкістю, якої джерело не може надати. Таким чином, для подолання цього розриву між джерелом та запитувальними потужностями для запису та зчитування даних використовується SYNCHRONOUS FIFO, який діє як еластичний буфер.

У процесі використання екранні терміни читання / запису FIFO для режиму читання / запису SN74LS224A на записі (WRITE CLOCK) та читання (READ CLOCK) повинні зберігатися для забезпечення належного функціонування пристроїв. Також на ринку пропонуються одночасні FIFO для читання / запису, які для їх порожнього або повного стану вимагають строків синхронізації між записами та читаннями, що забезпечують безперебійну роботу прапорів FULL та EMPTY. На додаток до мінімальної ширини імпульсів для двох сигналів (WRITE CLOCK 60 нс мінімум, READ CLOCK 30 нс мінімум), необхідно забезпечити, щоб чергування інструкцій щодо запису та читання були розділені за часом не менше ніж 50 нс. Якщо сигнали запису та читання походять з двох джерел, які працюють асинхронно один з одним, слід використовувати схему синхронізації, як показано на графічній частині (ДП.КСМ.07142/14.00.00.000 С2). Це забезпечує правильну роботу, навіть якщо два сигнали з'являються одночасно.

Підключення периферійних пристроїв до процесорів. Сучасні процесори часто значно швидше, ніж периферійні пристрої, до яких вони підключаються.

FIFO можна використовувати так, щоб швидкість обробки процесора не зменшувалася, коли він обмінюється даними з периферійним пристроєм. Якщо периферійна мережа іноді буває швидшою, ніж процесор, FIFO може знову використовуватися для вирішення проблеми (рисунок 3.3). Можливі різні варіанти схеми залежно від конкретної проблеми. У деяких випадках процесор читає вхідні дані по однонаправленій периферії, наприклад, з перетворювача A/D. FIFO може здійснювати буферизацію певної кількості вхідних даних, а потім використовувати переривання, щоб процесор читав дані. Це переривання може спрацьовувати прапором HALF FULL, ALMOST FULL або FULL.

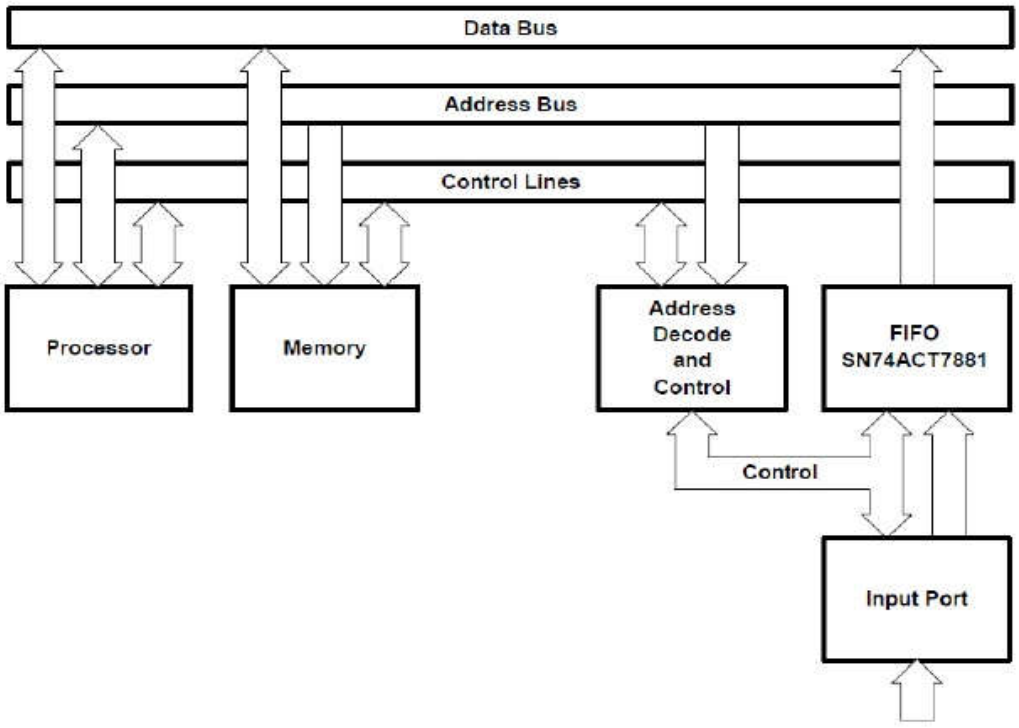

Рисунок 3.3 – Підключення FIFOдо односпрямованої периферії

Часто з'єднувані периферійні пристрої є двонаправленими, як паралельний порт, послідовний порт, контролер жорсткого диска або інтерфейс з магнітопідіймачем. У цих випадках існує можливість

використання двонаправленої FIFO типу SN74ACT2235. У цьому пристрої реалізовано два незалежних FIFO, кожен із шириною слів 9 біт і глибиною пам'яті 1024 слів. На рисунку3.4показана блок-схема для такої програми.

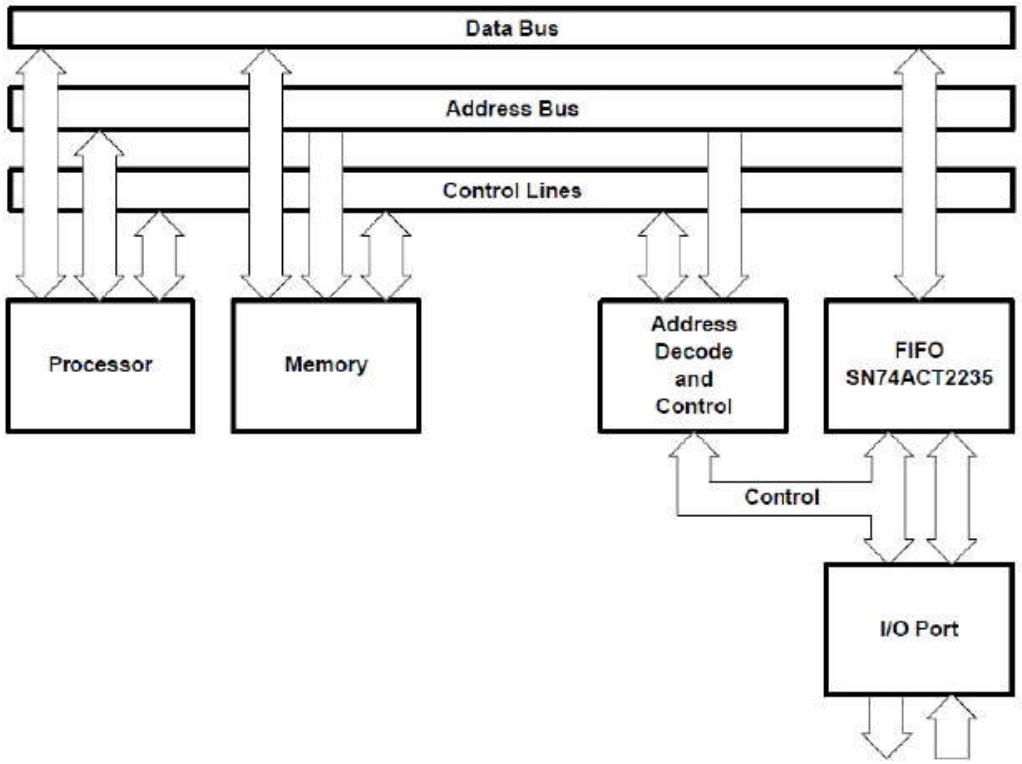

Рисунок 3.4 – Підключення FIFOдо двонаправленої периферії

Дані часто діляться на блоки та передаються по лініях даних. Приклади цієї програми - це комп'ютерні мережі та цифрові телефонні комутаційні пристрої. Якщо таке перетворення на блоки потрібно робити на дуже високій швидкості, це можливо лише за допомогою відповідного апаратного забезпечення, а не через програмне забезпечення. Простим рішенням цієї апаратної проблеми є використання FIFO (рисунок 3.5). Необхідно вибрати FIFO з прапором HALF FULL. Крім того, глибина пам'яті повинна відповідати вдвічі більшій кількості блоку. Як тільки встановлено прапорець HALF FULL, контролер відправлення надсилає блок даних. Контролер відправлення в цьому випадку складається з лічильника та деяких воріт і його дуже легко

реалізувати лише одним PAL. Написання даних в FIFO може здійснюватися постійно і не залежить від передачі блоків даних.

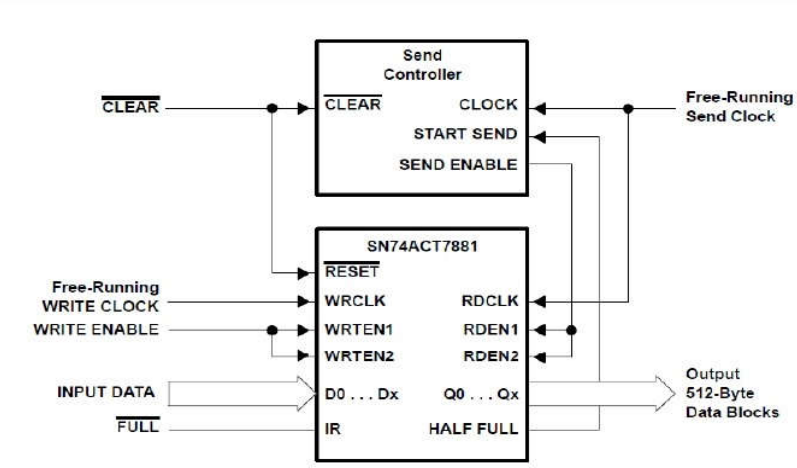

Рисунок 3.5– Передача блоків з синхронним FIFO

Завдяки FIFO, можна запровадити програмовану цифрову лінію затримки з мінімальними зусиллями. Через програмованого прапор AF / AE SN74ACT7881, лише один інвертор, крім FIFO, є обов'язковим (рисунок 3.6).

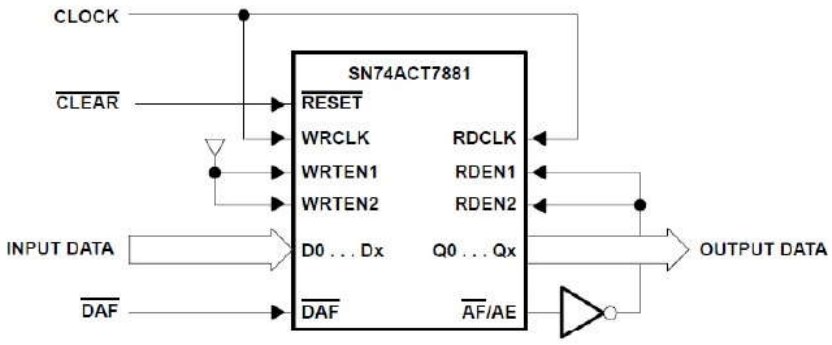

Рисунок 3.6 – Програмована затримка з FIFO

Розширюючи пристрій на рисунку 3.6, дані можуть бути зібрані як до, так і після виникнення події (рисунок 3.7). За допомогою програмованого прапору AF/AE можна вказати кількість слів даних, які збираються до і після події. До появи події, схема працює так, як показано на рисунку 3.4. Тут використовується фліп-флоп RS для керування входом WRTEN. TRIGGER

WINDOW встановлює цей фліп-флоп RS і дозволяє отримувати дані. Коли подія виникає, а сигнал TRIGGER WINDOW виходить на низький рівень, дані фіксуються, доки прапорець з введенням (IR) не стане низьким, а перемикач RS буде скинуто. Також на тракт сигналу є IO - шина від виходу AF/AE до входу RDEN1/RDEN2. Це запобігає подальшому читанню даних після події. Коли всі дані були зафіксовані, зібрані слова даних можуть бути прочитані.

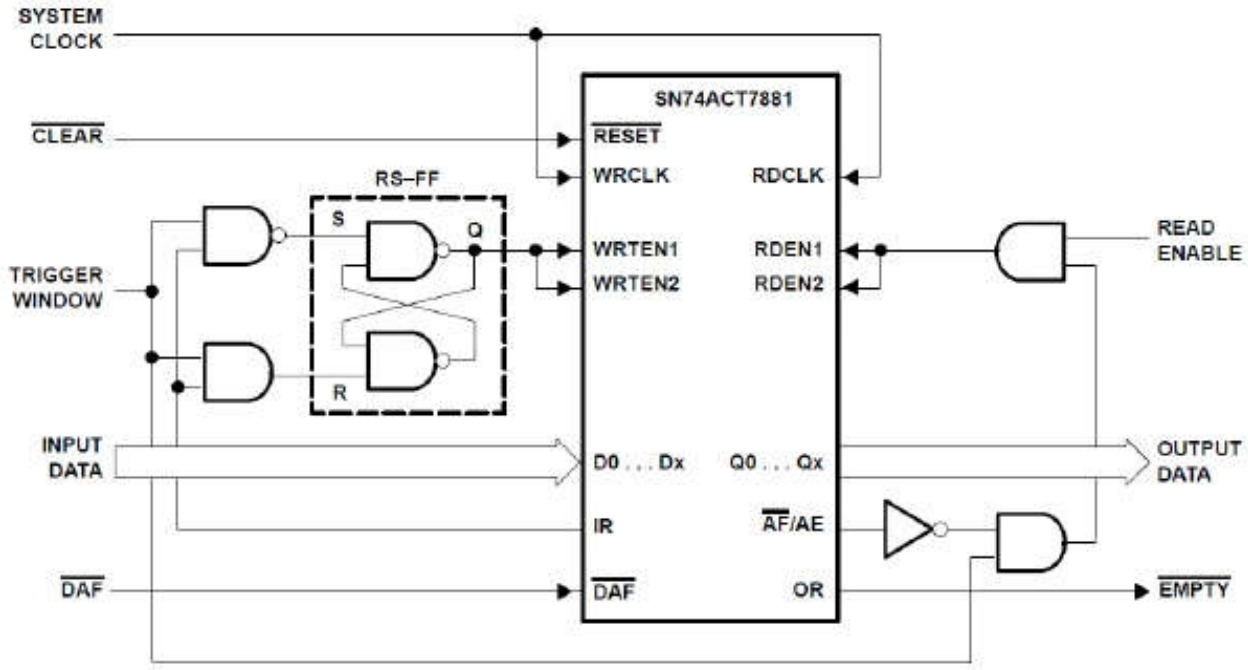

Рисунок 3.7 – Збір даних до та після події

3.3 Верифікація

Для здійснення верифікації проекту використано засіб SchematicTracer.Schematic Tracer показує відповідну схему Verilog на різних рівнях ієрархії Цей засіб відображає проектування як схему і дозволяє простежити сигнал через нього.

На рисунку 3.8 показано розроблено блок-схему тестового стенду для FIFO.

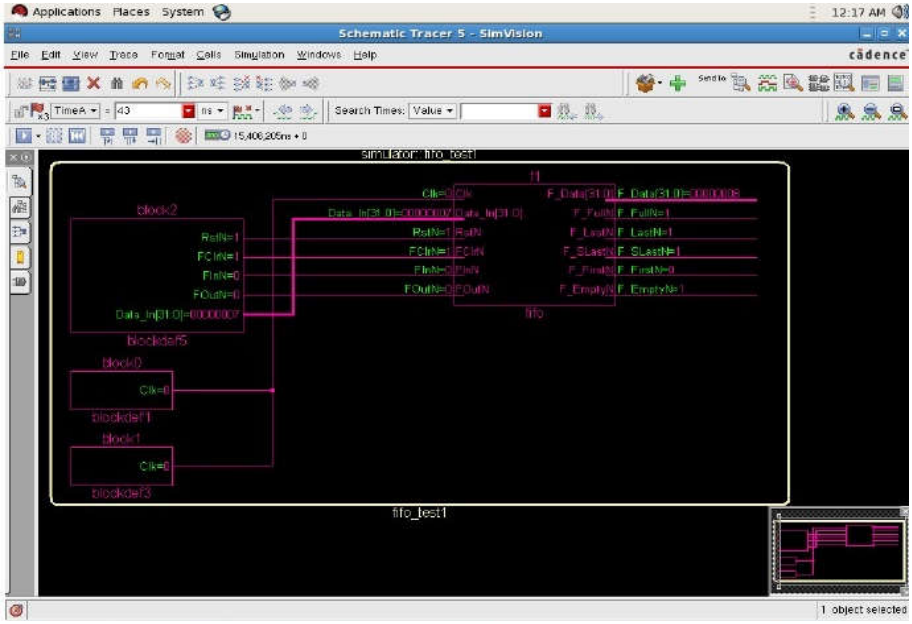

Рисунок 3.8 – Блок – схема тестового стенду для FIFO

На рисунку 3.9 показано проектовану блок-схему FIFO модуля.

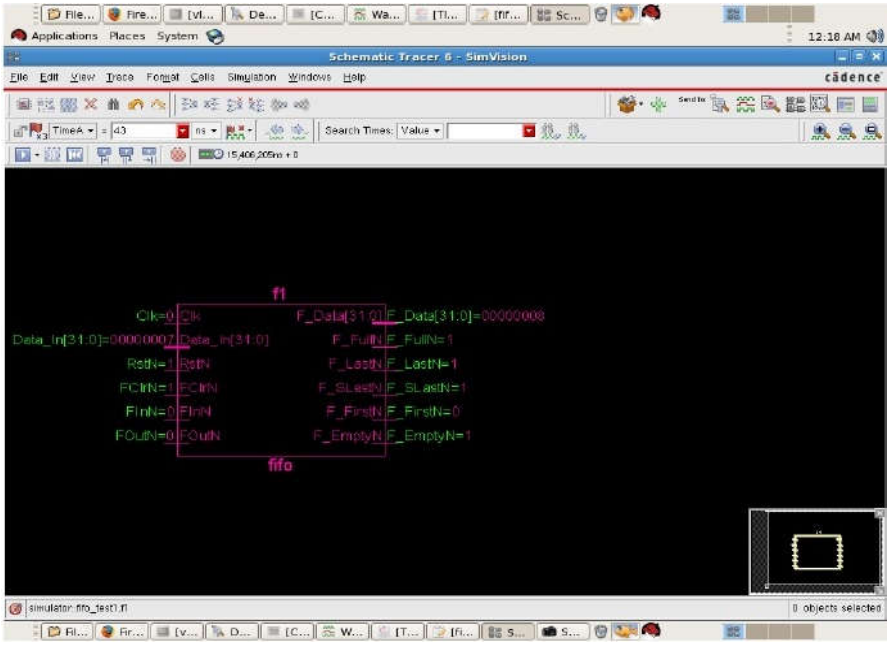

Рисунок 3.9 – Блок-схема FIFO модуля.

На рисунку 3.10 зображено блок-схему модуля блоку пам’яті.

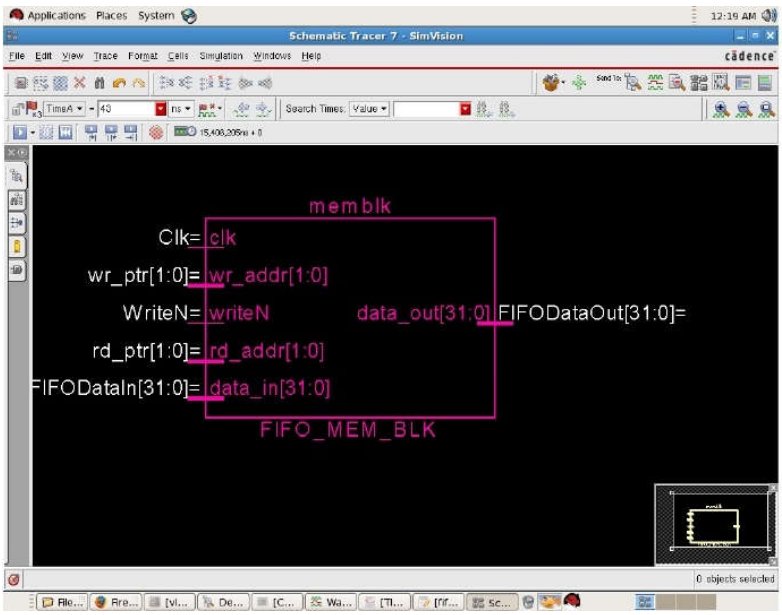

Рисунок 3.10 – Блок-схема модуля блоку пам’яті

На рисунку 3.11 подано розроблену блок-схему моделі FIFO.

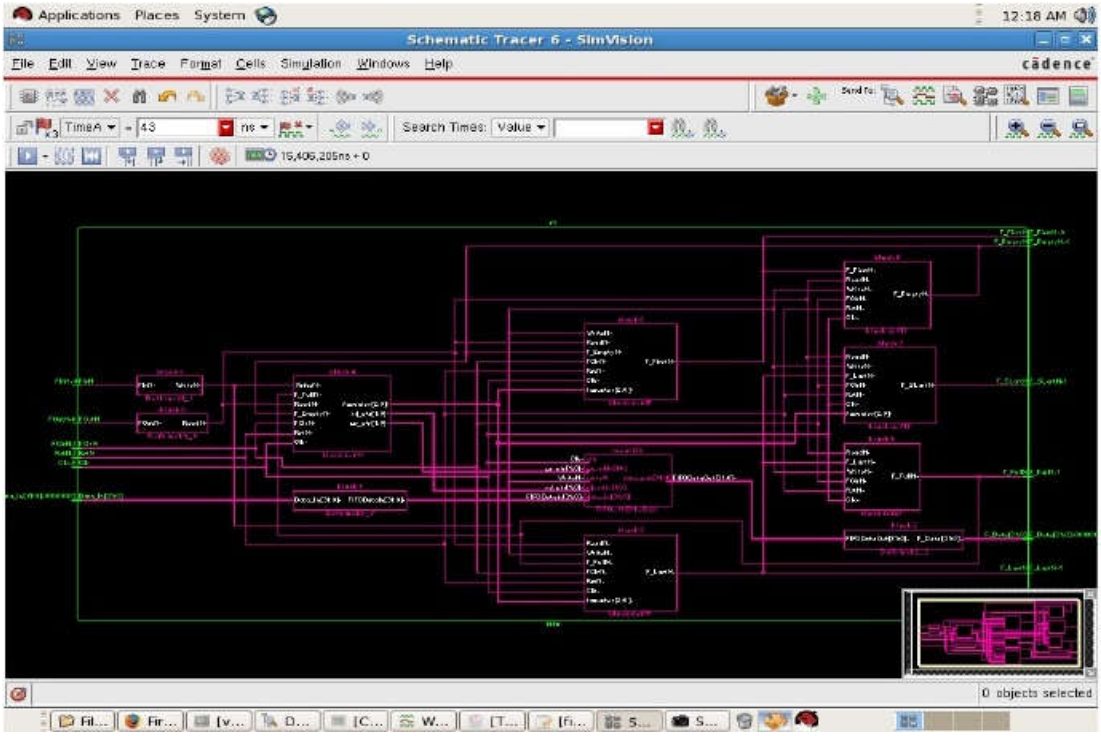

Рисунок 3.11 – Блок – схема моделі FIFO

На рисунку 3.12 зображено блок–схему деталізованого модуля блоку пам’яті.

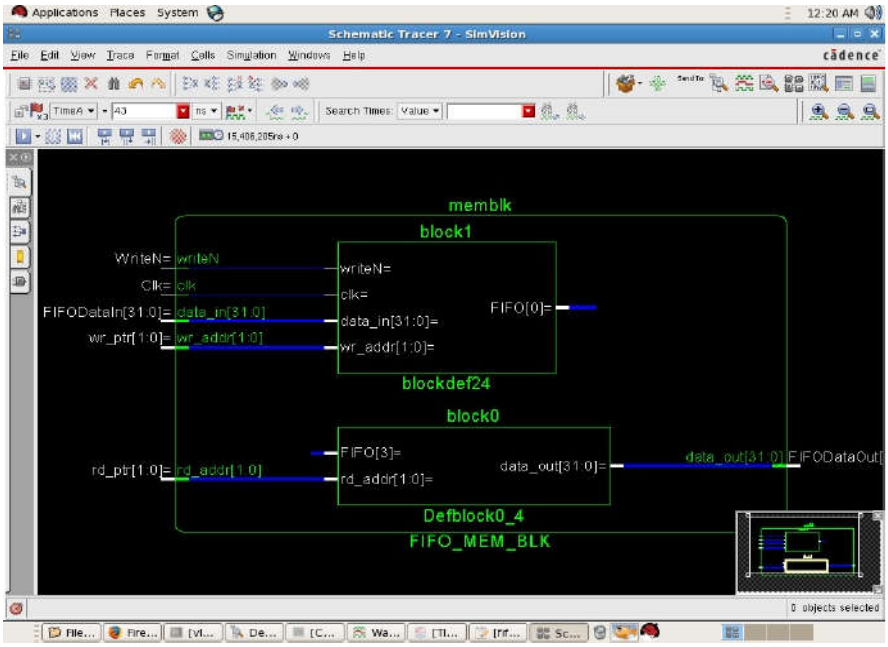

Рисунок 3.12 – Блок–схема деталізованого модуля блоку пам’яті

Для перевірки правильності роботи проекту використано засіб WaveformEditor. На рисунку 3.13 подано часову залежність виходу F_Dataвід початкових значень входів.

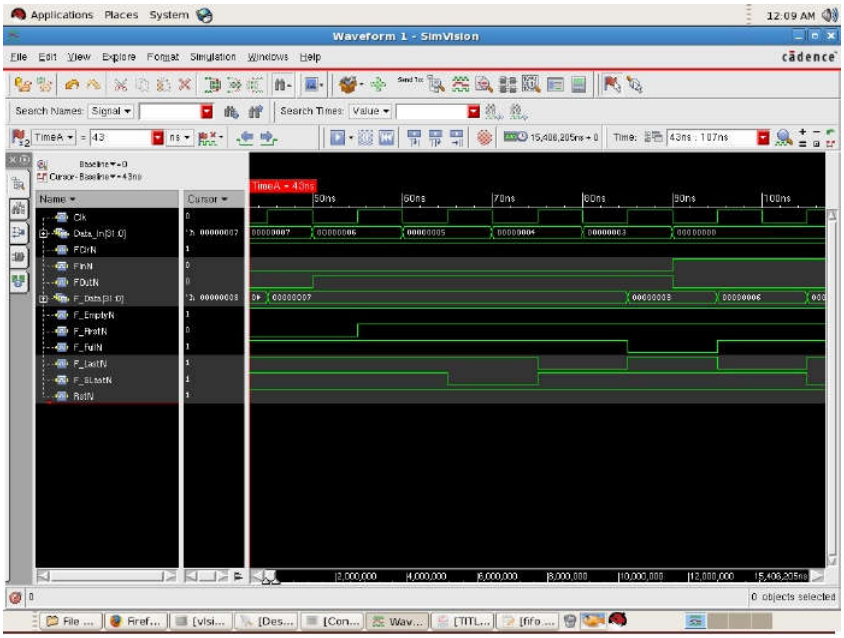

Рисунок 3.13 – Часова діаграма

Аналіз часової діаграми підтверджує правильність роботи моделі FIFO.

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ ПРОГРАМНОГО ЗАСОБУ

В цьому розділі дипломного проекту проводиться економічне обґрунтування доцільності розробки програмного забезпечення. Зокрема, здійснюється розрахунок витрат на розробку програмного забезпечення, експлуатаційних витрат, ціни споживання проектного рішення. В заключній частині визначаються показники економічної ефективності нового програмного продукту, обґрунтовуються відповідні висновки.

Розроблене програмне забезпечення призначене для візуалізації метричних ультразвукових сигналів і характеризується підвищеною ефективністю виконання алгоритму, що призводить до зменшення часу візуального представлення об’єкту дослідження.

4.1 Розрахунок витрат на розробку програмного забезпечення

У розробці проектного рішення задіяні наступні спеціалісти - розробники, а саме: керівник проекту (К); студент-дипломник (С); консультант техніко-економічного розділу (КТЕО).

Форму поділу робіт по всіх основних етапах і видах робіт, які повинні бути виконані показано в таблиці 4.1.

Таблиця 4.1 - Середній час виконання проекту та стадії технологічного процесу

№ п/п	Назва операції (стадії)	Виконавець, посада	Середній час виконання операції, год.
1	Підготовка	Студент	7
2	Розробкапроекту системи	Керівник ДП	16
		Консультант ТЕО, доцент	2
		Студент	211
3	Проектування технічної частини системи	Студент	15
4	Розробка програмного продукту системи	Студент	6
5	Встановлення та налаштування прогр. зас.	Студент	4
6	Тестування системи	Студент	4
Разом			269,5

Витрати на розробку і впровадження програмних засобів (*К*) включають:

$$K = K_1 + K_2$$

де K_1 - витрати на розробку програмних засобів, грн.;

K_2 - витрати на відлагодження і дослідну експлуатацію пристрою, грн.

Витрати на розробку програмних засобів включають:

– витрати на оплату праці розробників ($B_{ОП}$);

– витрати на покупні вироби (*Пв*);

– витрати на придбання спецобладнання для проведення експериментальних робіт (*Об*);

– накладні витрати (*Н*);

– інші витрати (*Ів*).

Витрати на оплату праці включають заробітну плату (ЗП) всіх категорій працівників, безпосередньо зайнятих на всіх етапах проектування. Розмір ЗП обчислюється на основі трудомісткості відповідних робіт у людино-днях та середньої ЗП відповідних категорій працівників.

У розробці проектного рішення задіяні наступні спеціалісти - розробники, а саме: керівник проекту; студент-дипломник; консультант техніко-економічного розділу.

Витрати на оплату праці розробників проекту визначаються за формулою:

$$B_{ОП} = \sum_{i=1}^{N} \sum_{j=1}^{M} n_{ij} \cdot t_{ij} \cdot C_{ij} , \qquad (4.1)$$

де n_{ij} – чисельність розробників *i*-ої спеціальності *j*-го тарифного розряду, осіб;

t_{ij} – затрачений час на розробку проекту співробітником *i*-ої спеціальності *j*-го тарифного розряду, год.;

C_{ij} – годинна ставка працівника *i*-ої спеціальності *j*-го тарифного розряду, грн..

Середньо годинна ставка працівника може бути розрахована за формулою:

$$C_{ij} = \frac{C_{ij}^{0}\left(1+h\right)}{РЧ_{i}} \qquad (4.2)$$

де C_{ij} – основна місячна заробітна плата розробника *i*-ої спеціальності *j*-го тарифного розряду, грн.;

h – коефіцієнт, що визначає розмір додаткової заробітної плати (при умові наявності доплат);

$РЧ_i$ - місячний фонд робочого часу працівника i-ої спеціальності j-го тарифного розряду, год. (приймаємо 168 год.).

Таблиця 4.2 - Вихідні дані для розрахунку витрат на оплату праці

№ п/п	Посада виконавців	Місячнийоклад(стипендія), грн.	Коефіцієнт Додаткової з/п	Сума
1	Керівник ДП, ,доцент,к.т.н.	5286	0,94	10254,84
2	Консультант техніко-економічного розділу, доцент	6026	1,47	14884,22
3	Студент	1287	0	1287

Звідси, загальні витрати на оплату праці ($В_{ОП}$) дорівнюють:

$$В_{ОП} = 16 \cdot \frac{10254,84}{168} + 2 \cdot \frac{14884,22}{168} + 247 \cdot \frac{1287}{168} = 3046,04 \text{ грн.}$$

Величну відрахувань у спеціальні державні фонди визначають у відсотковому співвідношенні від суми основної та додаткової заробітних плат. Згідно діючого нормативного законодавства сума відрахувань у спеціальні державні фонди складає 20,5 % від суми заробітної плати:

$$В_{Ф} = \frac{20,5}{100} \cdot 3046,04 = 624,44 \text{ грн.}$$

Матеріальні витрати — це вартість витрачених матеріалів, малоцінних та швидкозношуваних предметів на виробництво продукції, робіт або послуг, а також матеріалів і МШП, витрачених на адміністративні, збутові та інші потреби підприємства.

Загальна сума витрат на матеріальні ресурси (B_M) визначається за формулою:

$$B_M = \sum_{i=1}^{n} K_i \cdot Ц_i, \tag{4.3}$$

де K_i - витрата i-го типу матеріалу, натуральні одиниці вимірювання;

$Ц_i$ - ціна за одиницю i-го типу матеріалу, грн.;

i- тип матеріального ресурсу;

n - кількість типів матеріальних ресурсів.

Звідси, витрати на матеріальні ресурси дорівнюватимуть:

$$B_M = 390,50 \text{ грн.}$$

Проведені розрахунки занесемо у таблицю 4.3.

Таблиця 4.3- Розрахунок витрат на матеріали та комплектуючі

№ п/п	Найменування купованих виробів	Одиниця виміру	Ціна, грн	Кількість купованих виробів	Сума, грн.	Транспортні витрати (10% від суми)	Загальна сума, грн.
1	Папір (формат А4)	уп	70,00	2	140,00	14,00	154,00
2	Ручка кулькова	шт	5,00	2	10,00	1,00	11,00
3	Олівець простий	шт	3,00	2	6,00	0,60	6,60
4	Диски CD-R	шт	5,00	2	10,00	1,00	11,00
5	Зошит, 96 арк	шт	9,00	1	9,00	0,90	9,90
6	Чорнила для принтера	шт	180,00	1	180,00	18,00	198,00
Разом							390,50

Витрати на використання комп'ютерної техніки включають витрати на амортизацію комп'ютерної техніки, витрати на користування програмним забезпеченням, витрати на електроенергію, що споживається комп'ютером. Якщо для розробки КС використовується електрообладнання, то необхідно розрахувати витрати на електроенергію.

Загальна сума витрат на електроенергію розраховується за формулою:

$$B_E = \sum_{i=1}^{n} P_i \cdot k_i \cdot T_i \cdot Ц \,, \qquad (4.4)$$

де P_i - паспортна потужність i-го електрообладнання, кВт;

k_i - коефіцієнт використання потужності i-го електрообладнання (приймається 0.7 ¸ 0.9);

T_i - час роботи i-го устаткування за весь період розробки, год;

$Ц$ - ціна електроенергії, грн / кВт· год;

i - тип електрообладнання;

n - кількість електрообладнання.

Для розробки проекту даної системи використовується один ноутбук потужністю P = 0,5 кВт, який за весь період розробки працює 200 годин, та друкуючий пристрій потужністю P = 0,37 кВт, який працює 2 години.

Проміжні розрахунки на витрату електроенергії подані в таблиці 4.4.

Таблиця 4.4 - Витрати на електроенергію

Найменування устаткування	Паспортна потужність, кВт	Коефіцієнт використання потужності	Час роботи обладнання для розробки, год	Ціна електроенергії, $\frac{\text{грн}}{\text{кВт}\cdot\text{год}}$	Сума, грн.
Ноутбук	0,5	0,9	200	0,98	88,2
Принтер	0,37	0,9	2	0,98	0,653
Разом					88,85

Амортизація – це процес перенесення вартості основних фондів на вартість новоствореної продукції з метою їх повного відновлення. Амортизаційні відрахування використовуються для повної реновації, а також для їх часткового відновлення, тобто на модернізацію або капітальний ремонт.

Для визначення амортизаційних відрахувань застосуємо метод прямолінійного списання. Загальна сума амортизаційних відрахувань (B_{AM}) визначається за формулою:

$$B_{AM} = \sum_{i=1}^{n} \frac{B_i \cdot H_i}{100}, \tag{4.5}$$

де B_i - вартість i-го устаткування на початок звітного періоду, грн.;

H_i - річна норма амортизації i-го устаткування, %;

i - тип обладнання;

n - кількість устаткування.

Для проектування даної системи використовувався один ноутбук 6900 грн., та принтер вартістю 3200 грн.

Тоді:

$$B_{AM} = \frac{6900,0 \cdot 10}{100} + \frac{3200,0 \cdot 20}{100} = 1330,0 \text{ грн.}$$

Таблиця 4.5 - Амортизація основних фондів

Найменування устаткування	Вартість устаткування, грн.	Річна норма амортизації, %	Сума, грн.
Ноутбук	6900,0	10	690,0
Принтер	3200,00	20	640,0
Разом			1330,0

Транспортні витрати слід прогнозувати у розмірі 8–12 % від загальної суми матеріальних витрат.

$$B_T = 0,12 \cdot B_M , \tag{4.6}$$

де B_T – транспортні витрати.

$$B_T = 0,12 \cdot 390,50 = 46,86 \text{ грн.}$$

Накладні витрати пов’язані з обслуговуванням виробництва, утриманням апарату управління підприємства (фірми) та створення необхідних умов праці можуть становити 60–100 % від суми основної та додаткової заробітної плати працівників. Накладні витрати для даного проекту подані далі.

$$H_B = 0,7 \cdot B_{ОП} , \tag{4.7}$$

де H_B – накладні витрати.

$$H_B = 0,7 \cdot 3046,04 = 2132,23 \text{ грн.}$$

4.2 Складання кошторису витрат та розрахунок ціни проекту

Загальні витрати ($B_{КС}$) розрахуємо за формулою:

$$B_{КС} = B_{ОП} + B_{Ф} + B_{М} + B_{Е} + B_{АМ} + B_{Т} + H_{В} \quad (4.8)$$

Тобто:

$$B_{КС} = 7658{,}92 \text{ грн.}$$

Результати проведених розрахунків зведемо у таблицю 4.6.

Таблиця 4.6 - Кошторис витрат

Зміст витрат	Сума, грн.
Витрати на оплату праці (осн. і дод. ЗП)	3046,04
Відрахування на соціальні заходи	624,44
Матеріальні витрати	390,50
Витрати на електроенергію	88,85
Амортизаційні відрахування	1330
Транспортні витрати	46,86
Накладні витрати	2132,23
Разом	7658,92

Договірна ціна ($Ц_Д$) для проектних рішень розраховується за формулою:

$$Ц_Д = B_{КС} \cdot \left(1 + \frac{p}{100}\right), \quad (4.9)$$

де $B_{КС}$ – кошторисна вартість, грн.;

p - середній рівень рентабельності, % (приймаємо 30% за погодженням з керівником).

$$Ц_Д = 7658{,}92 \cdot (1 + 0{,}3) = 9956{,}6 \text{ грн.}$$

4.3 Визначення економічної ефективності і терміну окупності капітальних вкладень

Економічна ефективність — досягнення найбільших результатів за найменших затрат живої та уречевленої праці. Економічна ефективність є конкретною формою дії закону економії часу. За капіталістичного способу виробництва узагальнюючий показник економічної ефективності — норма прибутку.

Економічна ефективність (E_P) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E_P = \frac{П}{B_{KC}} \ , \qquad (4.10)$$

де $П$ – прибуток, грн.;

B_{KC} – кошторисна вартість, грн..

$$E_P = 2297{,}68 \text{ грн.} / 7658{,}92 \text{грн.} = 0{,}3.$$

Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень (T_p):

$$T_P = \frac{1}{E_P} \ . \qquad (4.11)$$

Тобто:

$$T_P = 1/0,3 = 3,3 \text{ р.}$$

Прийнятним вважається термін окупності близький до 7 років.

Розраховані економічні показники проекту занесемо до таблиці 4.7.

Таблиця 4.7 - Економічні показники розробки

№ п/п	Показник	Значення
1.	Собівартість, грн.	7658,92
2.	Плановий прибу'ток, грн.	2297,68
3.	Ціна, грн.	9956,6
4.	Економічна ефективність	0,3
5.	Термін окупності, рік	3,3

В даному розділі проведено розрахунок витрат на розробку проектного рішення. Враховуючи основі економічні показники з таблиці 4.7, можна зробити висновок, що при економічній ефективності 0,3 та терміні окупності – 3,3 роки проводити роботи по впровадженню даного пристрою є доцільним та економічно вигідним. Тому, з метою зниження вартості пристрою, варто було б здійснювати закупівлю обладнання у офіційних продавців обладнання.

ВИСНОВКИ

Основною метою дипломнї роботи – була наявність функціонального блоку пам'яті FIFO, який є правильним згідно його специфіки. Це було досягнуто шляхом проектування моделі FIFO. Правильність функціональності роботи моделі підтверджено тестуванням і аналізом реальних результатів сигналів симуляції.

1. Проаналізовано сучасні системи обробки інформації «Перший зайшов – перший вийшов».

2. Досліджено сучасні методи обробки інформації.Крім того розглянуто області застосування FIFO.

3. Обрано мову проектування – Verilog. Після того був створений алгоритм роботи проектованого FIFO.

4. Проаналізувавши алгоритм роботи було вибрано середовище моделювання в якому буде розроблятися модель FIFO. Вслід за цим відбулося моделювання проектованого блоку FIFO. Опісля була здійснена верифікація розробленої моделі.

5. Розроблена модель FIFOмає свої як позитивні так і негативні сторони До переваг можна віднести те, що запропонований проект використовує ефективну структуру масивів пам'яті та може бути додатково модифікований для роботи в додатках, де існують декілька циклів затримки між виробником даних, FIFO, і споживачем даних.

Недоліками є мінімальна затримка проходження слова даних через буфер FIFO, яка становить два такти синхросигналу з моменту запису до появи даних на виході та відносна залежність від технологічних примітивів серії і виробника елементної бази.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sutherland S. GasP: A minimal FIFO control/S.Sutherland,S. Fairbanks Advanced Research inAsynchronous Circuits and Systems. - Mar. 2001. –46 p.

2. Yantchev J.T Low-latencyasynchronous FIFO bu_ers/ J.T.Yantchev, C.G.Huang, M.B.Josephs, and M.Nedelchev, IProc. Asynchronous Design Methodologies. - May 1995.-125 p.

3. Myers C. Asynchronous Circuit Design/C.Myers,J.Wiley, Inc.- 2001.- 250 p.

4. DallyW.J. Digital Systems Engineering/ W.J.Dally, J.W.Poulton, Cambridge University Press,Cambridge, - UK, 1998.- 600 p.

5. Wakerly J.F.Digital Design: Principles and Practices, Prentice-Hall, third edition. - 1999. – 250 p.

6. Hurtado M.Ambiguous behavior of bistable elements/ M.Hurtado,D.L.Elliot Allerton Conf.on Circuit and System Theory. - Oct. 1975. – 605 p.

7. Rabaey J.MDigital Integrated Circuits/J.M.Rabaey, A.Chandrakasan, B.Nikolic A DesignPerspective, Prentice Hall, Upper Saddle River. - NJ, 2003. – 250 p.

8. Ho R. The future of wires/R.Ho, K.W.Mai, M.A.Horowitz Proceedings of the IEEE. - Apr.2001- 490 p.

9. Semeraro G. Energy-e_cient processor design using multipleclock domains with dynamic voltage and frequency scaling/ G.Semeraro, G.Magklis International Symposium onHigh-Performance Computer Architecture. - Feb. 2002 – 29 p.

10. Chapiro D. M. Globally-Asynchronous Locally-Synchronous Systems, Ph.D. thesis,StanfordUniversity, Stanford, CA. - USA, 1984. – 200 p.

11. SutherlandI. “GasP: A minimal FIFO control,”/I. Sutherland, S.Fairbanks in Advanced Research in Asynchronous Circuits and Systems. - Mar. 2001.- pp. 46–53.

12. NC–Verilog[Електронний ресурс].– Режим доступу:http://www.cadence.com/products/functional_ver/nc-verilog/index. aspx.

13. YantchevJ. T. “Low-latency asynchronous FIFO buffers”/J.T.Yantchev, C.G. Huang, M.B. Josephs, and I.M. Nedelchev in Proc. Asynchronous Design Methodologies. - May 1995. – 250 p.

14. SternheimE. Digital Design and Synthesis with Verilog HDL/E.Sternheim, Rajvir Singh, Rajeev Madhavan, Yatin Trivedi Automata Publishing Company. – 1993. – 300 p.

15. ThomasD. The Verilog Hardware Description Language, Fourth Edition/ D. Thomas and P.l Moorby Kluwer Academic Publishers. –1998. – 225 p.

16. Sutherland S. Verilog 2001A Guide to the New Features of the Verilog Hardware Description Language, Kluwer Academic Publishers, 2002. ISBN 0-7923-7568-8. Stuart Sutherland, The Verilog Pli Handbook: A User's Guide and Comprehensive Reference on the Verilog Programming Language Interface, Second Edition, Kluwer Academic Publishers. - 2002. – 325 p.

17. SmithD. HDL Chip Design: A Practical Guide for Designing, Synthesizing and Simulating ASICs and FPGAs Using VHDL or Verilog, Doone Publications, TX. - 1996. – 220 p.

18. CohenB. Real Chip Design and Verification Using Verilog and VHDL, VhdlCohen Publishing. - 2001. – 180 p.

19. IEEE 1364-2001 Standard, IEEE Standard Verilog Hardware Description Language, 2001.[Електронний ресурс].– Режим доступу:http://www.ieee.org.

20. Accellera Standard, System Verilog 3.0: Accellera's Extensions to Verilog, 2002. [Електронний ресурс].– Режим доступу:http://www.accellera.org.