Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____ д.т.н., проф. С. І. Шматков
«___» грудня 2023 р.

Пояснювальна записка

до кваліфікаційної роботи
магістра

на тему: «**Методи зменшення апаратурних витрат для суміщених мікропрограмних автоматів на мікросхемах програмованої логіки**»

Захищено на засіданні
Атестаційної комісії № 40
протокол № __ від __.12.2023 р.
Оцінка _____ / _____
Голова Атестаційної комісії
_____ **СКОБ Ю.О.**
(підпис)

Виконав:
Студент групи КІ – 61
спеціальність: 123 – «Комп'ютерна інженерія»
Галузь знань: 12 – Інформаційні технології

ПРОКОФ'ЄВ
Сергій Володимирович

Керівник:
д.т.н. професор
МІРОШНИК
Марина Анатоліївна

Рецензент:

Харків – 2023

АНОТАЦІЯ

Пояснювальна записка до магістерської атестаційної роботи складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і чотирьох додатків. Загальний обсяг роботи складає 79 сторінок, із яких 55 сторінок основної частини, списку використаних джерел та чотирма додатками.

Кваліфікаційна робота присвячена вивченню алгоритмів програмування логічних інтегральних схем, на яких базуються суміщені мікропрограмні автомати,для зменшення апаратурних витрат.

Під час виконання кваліфікаційної роботи були проаналізовані різні підходи до зменшення апаратурних витрат, але універсального підходу не було визначено. Тому для досягнення мети у роботі було запропоновано розробити алгоритм, що дозволить досягнути мети.

Об'єкт дослідження: методи програмування ПЛІС для подальшого зменшення апаратурних витрат.

Предмет дослідження: структура спеціалізованого пристрою на основі ПЛІС та алгоритми його програмування.

Мета роботи – визначення способу організації та реалізації паралельної обробки даних на основі ПЛІС для зменшення апаратурних витрат.

КЛЮЧОВІ СЛОВА: ПРОГРАМОВАНІ ЛОГІЧНІ ІНТЕГРАЛЬНІ СХЕМИ, СХЕМОТЕХНІКА, МІКРОПРОГРАМНІ АВТОМАТИ, АЛГОРИТМИ, ПРОГРАМУВАННЯ

ANNOTATION

The explanatory note for the master's certification work consists of an introduction, four sections, conclusions, a list of used sources, and four appendices. The total volume of the work is 79 pages, including 55 pages of the main part, a list of used sources, and four appendices.

The qualification work is dedicated to the study of algorithms for programming logical integrated circuits that serve as the basis for combined microprogrammed automata to reduce hardware costs. During the execution of the qualification work, various approaches to reducing hardware costs were analyzed, but a universal approach was not identified. Therefore, to achieve the goal of the work, an algorithm was proposed to facilitate reaching the objective.

Research Object: Programming methods for FPGA to further reduce hardware costs.

Research Subject: The structure of a specialized device based on FPGA and algorithms for its programming.

Work Objective: Determining the method for organizing and implementing parallel data processing based on FPGA to reduce hardware costs.

KEYWORDS: PROGRAMMABLE LOGIC INTEGRATED CIRCUITS, CIRCUITRY, MICROPROGRAMMED AUTOMATA, ALGORITHMS, PROGRAMMING.

ЗМІСТ

СПИСОК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

ПЛІС – Програмована Логічна Інтегральна Схема

НВІС – НадВелика інтегральна схема

ASIC – інтегральна схема для специфічного застосування

FPSLIC – Field Programmable System Level Integration Circuits, що можна перекласти як «програмовані схеми системного рівня інтеграції»

VHDL – VHSIC Hardware Description Language

AHDL – Altera Hardware Description Language

БІС – великі інтегральні схеми,

ШІМ – широтно імпульсна модуляція

ЕОМ – Електронна обчислювальна машина

ВСТУП

Методи дослідження включають математичний опис алгоритмів, аналіз та порівняння роботи алгоритмів.

Перелік завдань, які потрібно розробити полягає в огляді та аналізі методів обробки даних на основі ПЛІС для зменшення апаратурних витрат на мікропрограмних автоматах.

Наукова новизна полягає у у виявленні оптимальних засобів програмування ПЛІС та розробці більш ефективного алгоритму порівняно із існуючими методами.

Практична цінність виявляється у можливості використання розробленого алгоритму для ефективної обробки даних на ПЛІС у різних галузях, таких як медична діагностика, авіаційна промисловість, телекомунікації та інші.

Результати досліджень будуть апробовані та впроваджені в практику у співпраці зі спеціалістами відповідних галузей.

Останнім часом особливо актуальною темою є забезпечення безпеки технологічних процесів на різних виробничих об'єктах (особливо на хімічних заводах та АЕС). Для визначення несправностей за показаннями від різних датчиків призначено багато автоматизованих пристроїв. Більшість із них діють за принципом порівняння отримуваного значення з якоюсь «уставкою». З 2008 року намітилася тенденція щодо збільшення кількості впроваджуваних на виробництво вбудованих засобів контролю, реалізованих на програмованих логічних інтегральних схемах (ПЛІС), проте більшість підприємств в Україні все ще користуються застарілими методами та технологіями діагностування несправностей (в т.ч., заснованими на мікроконтролерах, ASIC, спеціалізовані комп'ютери або базові матричні кристали).

ПЛІС - це електронний компонент, що використовується для створення цифрових інтегральних схем. На відміну від звичайних цифрових мікросхем, логіка роботи ПЛІС не визначається під час виготовлення, а задається за

допомогою програмування (проектування). Для цього використовуються налагоджувальні середовища, що дозволяють задати бажану структуру цифрового пристрою у вигляді принципової електричної схеми або програми спеціальними мовами опису апаратури Verilog, VHDL, AHDL та ін., що забезпечує бажану гнучкість як при створенні, так і при застосуванні пристрою на об'єктах: передбачена можливість коригування деяких властивостей пристрою під специфіку конкретного виробництва (кількість входів від датчиків, кількість виходів, бажаний коефіцієнт надійності та інше).

Альтернативою ПЛІС під час створення засобів контролю є базові матричні кристали, що потребують заводського виробничого процесу для програмування; ASIC – спеціалізовані замовні БІС , які при дрібносерійному та одиничному виробництві суттєво дорожчі; спеціалізовані комп'ютери, процесори (наприклад, цифровий сигнальний процесор) або мікроконтролери, які повільніше ПЛІС через програмний спосіб реалізації алгоритмів.

Деякі виробники ПЛІС пропонують програмні процесори для ПЛІС, які можуть бути модифіковані під конкретне завдання, а потім вбудовані в ПЛІС. Тим самим забезпечується зменшення місця на друкованій платі та спрощення проектування самої ПЛІС. Найпоширеніші їх розглядаються у цій роботі.

Останні роки характеризуються різким зростанням щільності упаковки елементів на кристалі, багато провідних виробників розпочали серійне ПЛІС з еквівалентною ємністю понад 1 млн. логічних вентилів, при цьому ціни на програмовані логічні схеми йдуть вниз (середня ціна опустилася від $2500 в 2005 до $00 що ще більше збільшує конкурентоспроможність цього виду пристроїв для забезпечення контролю на великих підприємствах.

РОЗДІЛ 1.
ОГЛЯД ПРОГРАМОВАНОЇ ЛОГІКИ

1.1 Передумови розвитку програмованої логіки. Переваги ПЛІС

Загальною тенденцією розвитку елементної бази цифрової схемотехніки, починаючи з появи перших інтегральних мікросхем на початку 60-х років і до теперішнього часу є безперервне підвищення числа логічних елементів (ЛЕ), що розміщуються на кристалі, з одночасним зниженням питомої вартості одного елемента. Збільшення числа ЛЕ безперервно відкриває можливості створення все більш складних цифрових пристроїв, що розміщуються на одному кристалі. До основних (далеко не повних) позитивних результатів цієї тенденції можна віднести:

− постійне розширення функціональних можливостей та покращення споживчих властивостей кінцевих виробів;

− зменшення габаритів та споживаної потужності;

− підвищення надійності. Відображенням цієї тенденції у складі елементної бази цифрової

схемотехніки став перехід від інтегральних мікросхем (ІМС) малого та середнього ступеня інтеграції до великих (ВІС) та надвеликих (НВІС) інтегральних мікросхем. Одним із найбільш революційних результатів розвитку мікроелектроніки стала можливість створення перших мікропроцесорів (початок 70-х років), що дало потужний поштовх до впровадження цифрових технологій обробки інформації у всіх сферах людської діяльності.

Однак далеко не всі практичні завдання цифрової схемотехніки

можуть бути вирішені лише з використанням одних мікропроцесорів. Це зумовлено органічно властивою всім мікропроцесорам особливістю, пов'язаною з тим, що вирішення будь-якої задачі мікропроцесором завжди складається з

послідовності кроків кінцевої тривалості, у той час як для вирішення багатьох завдань (у тому числі пов'язаних із забезпеченням роботи самих мікропроцесорів) потрібні пристрої з мінімальною затримкою виконання логічних функций. Існує три основні способи задовольнити цю потребу:

- використання наборів стандартної дискретної цифрової логіки загального застосування, наприклад, наборів логічних мікросхем 74-ї серії (ТТЛ, КМОП) і типових периферійних БІС;

– використання замовних НВІС;

- Використання програмованих логічних інтегральних схем (ПЛІС). Набори дискретної цифрової логіки різних серій тривалий час були основний елементної базою розробки цифрових устройств. До складу таких наборів входить велика кількість окремих мікросхем, призначених як для виконання базових логічних функцій (І, АБО, НЕ), так і для виконання функцій типових цифрових пристроїв, таких як тригери, регістри, лічильники, мультиплексори, дешифратори і т.д. , що дозволяє використовувати їх розробки більш складних функціонально закінчених цифрових устройств. Основний недолік дискретної логіки полягає в тому, що для розробки кінцевих виробів зазвичай потрібна велика кількість мікросхем. Наслідком є велика кількість зовнішніх з'єднань, складність конструкції і великі габарити друкованих плат, велика довжина з'єднувальних провідників, складність побудови пристроїв з високою тактовою частотою, низька надійність. Для зменшення кількості мікросхем при проектуванні мікропроцесорних систем було розроблено ряд периферійних БІС, що являють собою спеціалізовані цифрові пристрої, призначені для виконання деяких типових функцій у складі

мікропроцесорних систем, такі як контролери динамічних ОЗП, контролери переривань, контролери прямого доступу в пам'ять, шини контролери .д. Однак, навіть застосування периферійних БІС не дозволяє повністю подолати основні недоліки дискретної цифрової логіки.

Найбільш кардинально проблема габаритів, швидкодії, спрощення конструкцій друкованих плат та забезпечення надійності вирішується шляхом розробки та виготовлення замовних НВІС (класичний приклад – чіпсети материнських плат та карт розширення персональних комп'ютерів). На жаль, цей шлях економічно виправданий тільки за великосерійного виробництва однотипних кінцевих виробів, внаслідок високої вартості та тривалих термінів підготовки виробництва замовних НВІС. Крім того, при використанні замовних НВІС можлива модифікація виробів потребує суттєвих додаткових матеріальних та тимчасових витрат.

У той самий час практично досить часто виникають потреби у створенні оригінальних цифрових пристроїв і виробів, не розрахованих на великосерійне виробництво, котрим розробка замовних НВІС не прийнятна або з економічних причин, або за термінами виконання. Протягом тривалого часу єдиним шляхом для вирішення таких завдань було використання інтегральних мікросхем дискретної логіки та периферійних БІС, адже можливості перших поколінь простих ПЛІС були дуже обмежені, а ціна складних ПЛІС дуже високою, крім того, були певні складнощі і з проектуванням цифрових пристроїв на ПЛІС .

В останні роки, однак, стався різкий прорив як у технології виготовлення ПЛІС, так і в розробці інструментальних засобів, призначених для проектування цифрових пристроїв на ПЛІС та випуску готових виробів. Технологічний прорив характеризується різким збільшенням числа еквівалентних логічних вентилів, що розміщуються на одному кристалі (до 10 млн вентилів у найближчій перспективі),

підвищенням робочої частоти (до 400 МГц) з різким одночасним зниженням як питомої, так і абсолютної вартості. Так, за даними [1], вартість ПЛІС ємністю 100 000 знизилася з 1500 ... 3000 до 100 ... 350 у. е. і ця тенденція є стійкою. Крім того, в [2] зазначається, що програмована логіка за темпами розвитку почала випереджати інші напрями цифрової електроніки (універсальних мікропроцесорів, сигнальних процесорів, мікроконтролерів та замовних ВІС). Наприклад, один зі світових лідерів з виробництва ПЛІС фірма «Xilinx» у І кварталі 2007 р. (до фінансової кризи) збільшила обсяг продажів на 26% порівняно з І кварталом 2006 р. При цьому вартість ПЛІС обсягом до кількох тисяч вентилів на українському ринку , представленому цілим рядом постачальників електронних компонентів, становить лише кілька у. е.

Основними фірмами-виробниками ПЛІС в даний час є такі фірми, як "Altera", "Atmel", "Csypres", "Lattice", "Lucent", "Xilinx". Надалі ми розглянемо найперспективніші з них.

1.2 Порівняльний аналіз ПЛІС, НВІС та мікроконтролерів

Техніко-економічні показники сучасних ПЛІС досягли такого рівня, який забезпечує при випуску партій виробів до кількох сотень нижчу вартість кінцевих виробів, ніж застосування замовних НВІС. Очевидно, що найближчим часом цей поріг підвищуватиметься, забезпечуючи економічну доцільність застосування ПЛІС та у великосерійному виробництві. Крім суто економічних передумов, цьому сприяє низка додаткових переваг сучасних ПЛІС, які полягає в тому, що, зберігаючи всі переваги однокристального рішення, властивого замовним НВІС, пристрої на основі ПЛІС можуть дуже швидко і з малими витратами піддаватися модернізації (upgrade). Завдяки наявності у багатьох типах ПЛІС вбудованих систем програмування та конфігурування, що

дозволяють перепрограмувати їх прямо на місці без використання зовнішніх програматорів, пристрої на ПЛІС можуть модернізуватися навіть перебуваючи у постійній експлуатації у замовника. З цією метою провідні виробники програмованої логіки включають засоби проектування пристроїв на ПЛІС підтримку оновлення версії ПЛІС через інтернет. Крім того, терміни проектування і випуску готової продукції на ПЛІС незмірно менше, ніж розробка та виробництво замовних НВІС, що в умовах ринку, що динамічно змінюється, іноді може мати вирішальне значення. Важливою обставиною є те, що для випуску готової продукції не потрібне складне і дороге технологічне обладнання, яке потрібне для виробництва замовних НВІС. Остання обставина відкриває середнім і навіть дрібним фірмам шлях ринку виробів сучасної електронної техніки, який був раніше доступний лише гігантам електронної промисловості (див. таблицю 1.1).

Таблиця 1.1

Відносні рейтинги ефективності застосування стандартної логіки, замовних НВІС та ПЛІС

Показники	Стандартна логіка	Замовні НВІС	ПЛІС
Швидкодія	2	5	5
Щільність упаковки елементів	2	4	5
Вартість	2	4	3
Час розробки	3	2	5
Час налагодження	2	2	4
Час виробництва	3	2	5
Можливості модернізації	3	2	5
Ризик виробника	3	2	5
Ступінь автоматизації процесу проектування	2	4	5

При однаковій кількості логічних елементів у проекті вартість сумарного числа стандартних мікросхем перевищує вартість ПЛІС, а перевага вартості замовлених НВІС проявляється тільки при дуже великому обсязі виробництва ідентичних мікросхем.

До недавнього часу мікропроцесори та ПЛІС являли собою два незалежні напрямки в цифровій мікроелектроніці. Насущні потреби у подальшому підвищенні ступеня інтеграції сприяли розробці НВІС класу SOC (Systems On Chip) – тобто "систем на кристалі".

Сутність такого підходу полягає у злитті двох шляхів розвитку складних універсальних мікросхем (мікропроцесорів та ПЛІС) в єдине ціле шляхом розміщення на одному кристалі пов'язаних між собою мікропроцесорного ядра та масиву вентилів ПЛІС. Піонером у створенні подібних пристроїв стала фірма «Atmel», яка в жовтні 1999 р. випустила перше сімейство НВІС FPSLIC .

Типова мікросхема FPSLIC фірми «Atmel» містить швидкісне процесорне ядро AVR RISC з продуктивністю понад 30 MIPS, масив FPGA об'ємом до 4000 еквівалентних логічних вентилів, 10 кбайт х 16 програмної пам'яті, 4х послідовних порту, три таймери-лічильники, інтерфейс I^2C, два паралельні порти). До складу сімейства на сьогодні входять три мікросхеми типу AT94K10, AT94K20, AT94K40 з обсягом FPGA в 1000, 2000 та 4000 вентилів відповідно [3]. Через очевидні переваги мікросхем цього класу можна припустити, що НВІС класу SOC найближчим часом інтенсивно розвиватимуться і отримають широке застосування в цифровій схемотехніці.

Неабияку роль у розширенні сфер застосування ПЛІС, скороченні часу та зниженні трудовитрат на проектування відіграли і значні успіхи у створенні інструментальних засобів для розробки та випуску кінцевих виробів на ПЛІС, основу яких складають спеціальні пакети програм, що

забезпечують весь виробничий цикл зі створення цифрових пристроїв на ПЛІС, від розроблення схем до випуску готових виробів.

Висновки за розділом 1

У підсумку, технологічні та економічні параметри сучасних ПЛІС досягли рівня, коли для виробництва партій виробів в кількості кількох сотень вони пропонують меншу вартість, ніж замовні ІС. Гнучкість ПЛІС для перепрограмування та скорочені терміни дизайну і виробництва роблять їх привабливим вибором, демократизуючи доступ до ринку електроніки для середніх і малих підприємств.

Підсумовуючи, використання периферійних ІС не повністю вирішує основні недоліки дискретної цифрової логіки. Великі труднощі залишаються щодо розмірів, швидкості, спрощення конструкцій печатних плат та надійності. Замовні інтегральні схеми (ІС), такі як чіпсети для материнських плат та карт розширення в особистих комп'ютерах, вирішують ці проблеми, але економічно виправдані тільки для великосерійного виробництва через високі витрати та тривалі терміни підготовки виробництва.

У даній кваліфікаційній роботі буде розглядатись один з методів , ще більшого збільшення ефективності ПЛІС, а саме покращення алгоритму програмування.

РОЗДІЛ 2.

ОГЛЯД СТРУКТУРИ ПРОГРАМОВАНОЇ ЛОГІКИ

2.1 Компаратори LM311 та LM339

Безумовно, існує безліч компараторів, які заточені під конкретні потреби. Безумовно є і вітчизняні аналоги різних мікросхем. Один з прикладів це компанія ‘‘Квазар-Мікро’’

Рисунок 2.1 – Зовнішній вигляд вітчизняних аналогів мікросхем

На даний момент безпосередньо на сайті можна побачити декілька моделей різних мікроконтролерів, таких як:

- AT89C1051-24PI
- AT89C2051-24PU
- AT89C51RC2-RLTIM

Програмована логіка:

- EPC1LC20N

• EPC2LC20N

• EPF81188AQC208-4

Зовнішній вигляд ви можете побачити на рисунку 2.1

Такі мікросхеми часто зустрічаються, зокрема, на системних платах ЕОМ, а також у системах управління ШІМ контролерів у блоках перетворення напруги (наприклад, у комп'ютерних блоках живлення із системою живлення ATX). Їхня вартість також невелика: в межах одногодвох доларів, проте функції цих компараторів обмежені і вони можуть бути застосовані тільки в спеціалізованих областях або з додатковими пристроями. Крім того, як неважко помітити, кількість входів і виходів у них строго задано, вони не мають можливості запам'ятовувати значення сигналів, щоб порівняти, скажімо, числа, що подаються в двійковому коді, у них немає багатьох найпростіших функцій. Такі компаратори можуть тільки порівнювати кілька значень напруги, що одночасно подаються, і вибирати з них менше / більше.

2.2 Компаратори на операційних підсилювачах

Також компаратори будуються з урахуванням мікросхем операційного підсилювача. Порівняльні мікросхеми таких пристроїв призначені для зіставлення двох аналогових сигналів та надання інформації про більший з них. Розміри таких схем – 5 сантиметрів. Для прикладу були представлені аналогові компаратори TS3702IDT та LM393DT на рисунку 2.2.

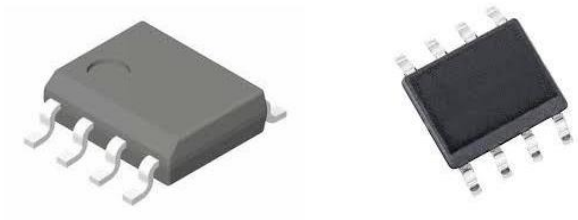


Рисунок 2.2. – Зовнішній вигляд аналогових компараторів (TS3702IDT та LM393DT)

Принцип роботи даної мікросхеми заснований на «впізнанні», тобто в такому випадку важлива різниця характеристики сигналів, а не пікові рівні, проте максимальні все ж таки необхідно враховувати, щоб вони знаходилися в межах допустимого рівня для конкретної мікросхеми. Порівняльна мікросхема побудована за типовим граничним принципом, тобто стан схеми компаратора змінюється, коли сигнал на виході досягає певного рівня. Дані мікросхеми застосовуються як самостійні елементи у схемах різних регуляторів, або як складова частина складнішої мікросхеми – таймер, що використовуються у генераторах імпульсів різних видів (див. рисунок 2.3)

Як компаратор можна використовувати практично будь-який операційний підсилювач, для чого досить просто виключити зворотний зв'язок. Для цих цілей бажано підбирати підсилювачі з максимальними характеристиками посилення і з великою швидкістю наростання сигналу. Слід зазначити, що ОУ можна перетворити на порівняльну мікросхему, а ось навпаки – ні, тому що не може виконувати функцію посилення чи поділу, вона для цього не призначена.

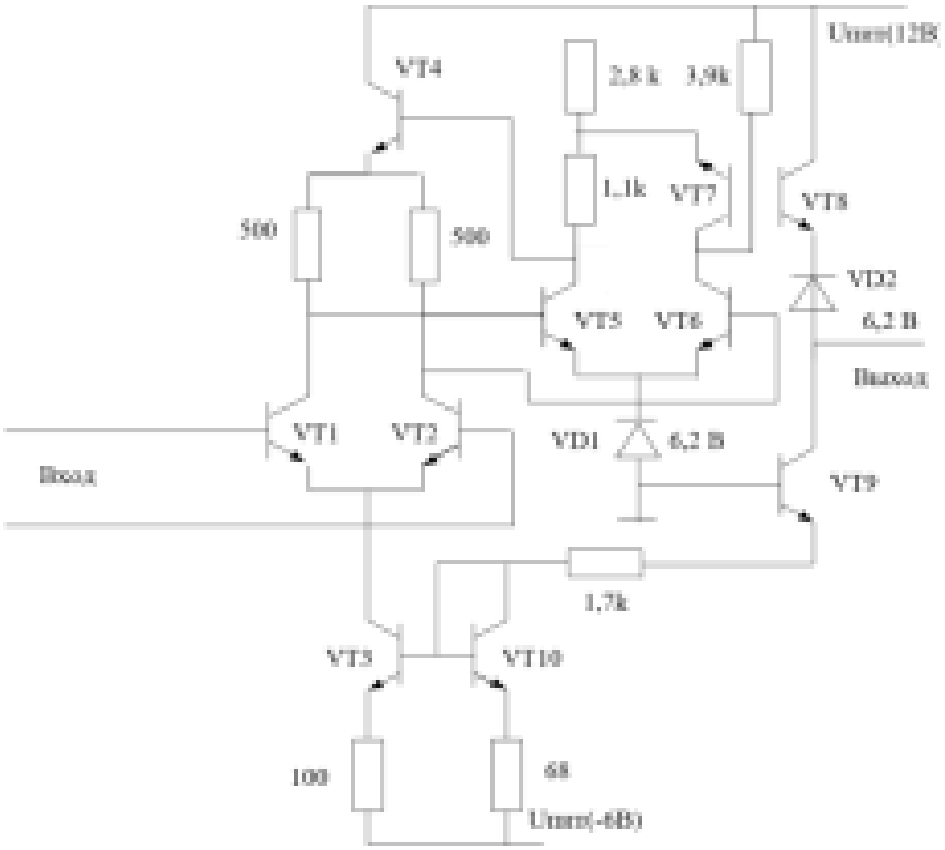


Рисунок 2.3 – Схема компаратора на операційному підсилювачі (mA710)

Найчастіше у схемотехніці використовуються мікросхеми-компаратори К554СА3 та К521СА3. Ці мікросхеми ідентичні за своєю структурою та характеристиками: мають потужний вихід (до 50 мА), високу стабільність та низьку вартість. Завдяки тому, що в принципову схему компаратора включений транзистор з відкритим колектором та емітером, мікросхему можна підключати залежно від необхідності або із загальним емітером, або із загальним колектором. Наприклад, якщо потрібно узгодити вихідний рівень із рівнем інших логічних мікросхем, компаратор підключається за схемою з відкритим колектором.

Більшість порівняльних мікросхем розраховано на використання при двополярній напрузі живлення ±15 В, але так само можуть працювати і при однополярному від 5 до 15 В. Більшість порівняльних мікросхем працюють досить ефективно, але в деяких ситуаціях, наприклад при подачі на вхід сигналу, що повільно змінюється, з невеликим

високочастотним імпульсом, з'являється специфічний шум, так зване деренчання. Брусок виникає в результаті множинних перемикань компаратора на межі порогового рівня. Щоб усунути деренчання, можна встановлювати його між виходами малої ємності (від 10 до 1000 пФ), або за допомогою рознесення порогів. Перший спосіб не завжди підходить, тому що через ємність суттєво падає швидкодія мікросхеми. Другий спосіб полягає в установці різних порогових напруг на включення та вимкнення, наприклад, коли мікросхема спрацьовує при більш високій напрузі. Подібний принцип рознесення порогів (гістерезис) використовується у формувачі сигналу тригера Шмітта (див. рис. 2.4.).

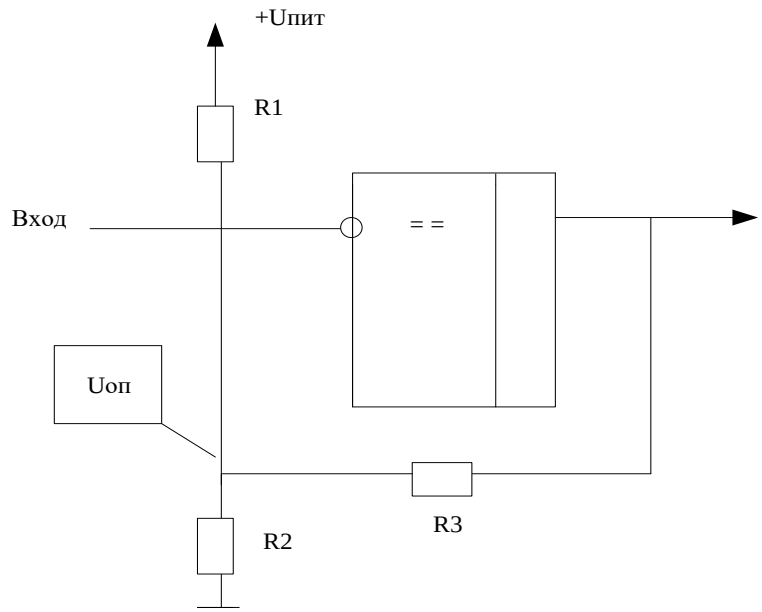


Рисунок 2.4 – Введення гістерези в рахунок позитивного зворотного зв'язку

Гістерезис у схему компаратора вводиться шляхом підключення трьох транзисторів: так як на виході напруга постійно коливається резистор (R3) зрушує потенціал неінвертуючого виводу в напрямку резистора (R1), то в напрямку резистора (R2). Таким чином, створюється додатковий струм, призводить до падіння напруги. Зауважимо, що схема підключення на

транзисторах істотно впливає збільшення швидкодії спрацьовування.

2.3 Компаратори серії Р300Х

Також, як приклад, можна навести компаратор напруги Р3003, який все ще використовується в деяких лабораторіях і навіть виробництвах країн СНД та Азії (Індія, Монголія, Пакистан). Компаратори напруги Р3003, Р3003М1-1, Р3003М1-2 були розроблені і випускалися до кінця двадцять першого століття. В даний час випуск Р3003, Р3003М1-1, Р3003М1- 2 припинено, проте їх ще повсюдно використовують у метрології та електротехніці (див. рис. 2.5).

Рисунок 2.5 – Зовнішній вигляд компаратора Р3003

Цей компаратор призначений для:

– компарування та вимірювання напруги постійного струму; –видачі каліброваних напруг від 10 нВ до 11,111110 В;

– посилення напруги від 20 нВ до 10 В з виходами на цифровий та аналоговий прилади (10 В, 5 мА);

- харчування прецизійних електричних ланцюгів (10 В, 10-25 мА). При компаруванні та вимірі напруги спільно функціонують вбудований калібратор, диференціальний вимірювальний підсилювач, диференціальний мікровольтметр і дільник вхідної напруги – речі, які досить просто підключаються до будь-якого іншого компаратора практично без витрат. Але

подивимося на характеристики цього диво-приладу:

Клас точності: 0,0005 (для порівняння, у цифрових компараторів точність практично обмежена лише точністю вхідних сигналів). Межа основної похибки компарування, що допускається, для різних значень напруги, дорівнює, мкВ:

±(5U+1) (U=11,111110 D);

±(5U+0,1) (U=1,1111110);

±(10U+0,04) (U=0,1111110);

±(50U+4) (U=111,11110);

Здатність навантаження по виходу АВ: 10 В; 5 мА

Живлення від мережі змінного струму: 220±22 В; 50, 60 Гц

Маса: 13 кг (тоді як маса цифрових і навіть деяких аналогових компараторів не перевищує кількох грам).

Габарити: 488х170х380 мм (габарити деяких мікроконтролерів не перевищують 10х2х5 мм).

Як ми переконалися, можливостей у Р3003 справді небагато. На базі ПЛІС, до яких ми прийдемо пізніше, можна спроектувати пристрій у сотні тисяч разів складніший, а його собівартість (виключаючи оплату праці програміста) буде у вісім разів меншою.

2.4 Порівняльні пристрої, створювані програмними методами

Створити цифровий компаратор, який відповідає вимогам конкретного виробництва, найпростіше програмним шляхом. Це дозволить задати йому необхідні характеристики, а також при необхідності відносно безболісно змінити їх, злегка підкоригувавши програму. Далі розглядаються два найбільш поширені варіанти створення такого компаратора.

Порівняльний пристрій на мікроконтролері

Всі розглянуті компаратори (крім Р3003 та його аналогів) є недорогими, проте вузькими у своїй спеціалізації, а також сучасні засоби дозволяють самому програмувати пристрій під свої конкретні потреби. Наприклад, зробити на мікроконтролер компаратор під задану кількість входів досить просто можна в програмі CodeVisionAVR. Наприклад, напишемо програму, яка виводитиме інформацію про порівняння значень напруги на двох ніжках різних портів на рідкокристалічний дисплей (Код у додатку Д) .

Собівартість такого компаратора (або іншого, оптимізованого) буде, за винятком оплати праці програміста, дорівнювати вартості мікроконтролера, тобто, для ATMega16 - в межах 4 $. Однак навіть створені програмно, заточені під конкретні потреби компаратори, в які будуть закладені (якщо розглядати приклад атомної станції) функції контролю та стійкість до відмов, у будь

якому випадку будуть обмежені можливостями мікроконтролера – як за напругою, так і за кількістю ніжок (порівнюваних одночасно сигналів). Однак можна створювати ще складніші і, відповідно, функціональні пристрої порівняння.

Порівняльні пристрої на ПЛІС

Програмовані логічні інтегральні схеми дозволяють створювати пристрої практично необмеженої складності, з величезною кількістю входів і виходів (бувають пліс із більш ніж 1000 висновків («пінів)), з величезною

швидкістю передачі даних, з великими значеннями вхідної напруги, великим обсягом пам'яті (до 540 кбіт), допустиму дозу опромінення (що важливо при застосуванні на атомних станціях, де, у разі виникнення небезпеки, електроніка швидко виходить з ладу і врятувати ситуацію може лише якась стійка схемка, на зразок нашого компаратора, який зрозуміє, що допустима напруга перевищена) – більше трьох мільйонів (3000000) РАД (у той час, як летальна доза для людини, при якій смерть настає через кілька годин – 10000 РАД, а первинна променева хвороба – 200 РАД), висока швидкодія (аж до 5 наносекунд) та багато іншого.

Проектування на ПЛІС можна здійснювати схемотехнічним методом, тоді простий компаратор, який з'ясовує, більше, менше або дорівнює сигнал, що прийшов з А, сигналу з В виглядатиме так, як зображено на рисунку 2.6.

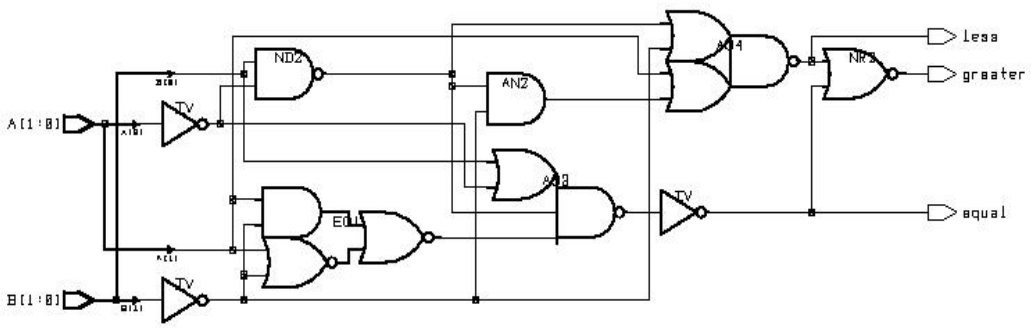


Рисунок 2.6 – Простий компаратор серед MAX+plus II

Неважко помітити, що схема навіть найпростішого компаратора не така легка і читається, як того хотілося б. У Quartus і навіть Max+plus (якщо використовувати бібліотеки, що підключаються) компаратор можна поставити окремим блоком, проте це буде той же компаратор, який ми розглядали раніше: рівня LM311 або схожого. При збільшенні кількості входів схема швидко ускладнюватиметься. Також суттєвим мінусом є те, що в тілі такої схеми-програми не можна залишати коментарі, що ще більше

ускладнює проектування складних пристроїв. На щастя, у програмних середовищах фірми Altera існує можливість створити необхідний пристрій однією з мов програмування Hardware Description Language: VHDL або

Verilog HDL. Ось як виглядає той самий компаратор, написаний мовою VHDL: –

```
– n-bit Comparator (ESD book figure 2.5)
– by Levichev Artem, 03/2011
– this simple comparator has two n-bit inputs &
– three 1-bit outputs
– library ieee; use ieee.std_logic_1164.all;
– entity Comparator is
generic (n: natural:=2); – описание портов входа и выхода
port (A: in std_logic_vector (n-1 downto 0); B: in
std_logic_vector (n-1 downto 0); less: out std_logic;
equal: out std_logic; greater: out std_logic); end
Comparator;
– architecture behv of Comparator is begin process (A, B) begin if
  (A<B) then  less <= '1'; equal <= '0'; greater <= '0'; elsif (A=B)
  then less <= '0'; equal <=  '1'; greater <= '0'; else less <= '0'; equal
  <= '0'; greater <= '1'; end if; end  process; end behv;
–
```

Ця реалізація, як помітно, є набагато більш читабельною та зрозумілою. Також важливо, що при збільшенні кількості портів і кількості входів така програма не так швидко захаращуватиметься і, в будь-якому випадку, в її тілі завжди можна буде написати пояснення для себе або майбутніх розробників.

Висновки за розділом 2

Розглянувши приклади різних приладів можна зробити висновки , що з часом програмована логіка перетерпіла великих змін, які в свою чергу мали вплив на зовнішній вигляд та структуру. Завдяки цьому сильно підвищилась ефективність мікросхем. У зв’язку з цим потреба у старих моделях зникла так як з’явились більш вигідні та компактні варіанти під одні і ті самі вимоги.

РОЗДІЛ 3.

РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ ДЛЯ ТЕСТУВАННЯ АЛГОРИТМУ

3.1 Створення схеми пристрою

Проектуємо пристрій, який відповідатиме нашим потребам: компаратор із додатковою системою перевірки його на правильність функціонування. Надалі розробка вестиметься на ПЛІС.

Структурна схема пристрою

Створюємо схематичний проект майбутнього пристрою. У режимі роботи на схему порівняння надходитимуть два цифрові сигнали, з яких, залежно від того, чи рівні вхідні значення, буде формуватися один вихідний. У режимі тестування на компаратор повинен подаватися один сигнал (увімкнення режиму тестування), і, залежно від результатів, що демонструються цифровим компаратором, на вихід подаватиметься сигнал «Норма» (пристрій функціонує нормально) або «Ненорма» (помічений збій у роботі пристрою)

Первинний варіант електричної принципової схеми системи

Цей варіант був створений спочатку, щоб визначитися з основними напрямками розробки. Тут є кілька помилок, і в результаті досвідчений зразок за певних умов видавав неправильні значення.

Останній варіант електричної принципової схеми системи

Після детального розгляду першого варіанта аналізу схожих пристроїв і можливостей реалізації окремих компонентів знайшлися шляхи спрощення схеми. По-перше, замість двох лічильників з чотирма висновками цілком можна помістити один лічильник, і його сигнали роздвоювати на входи A і B. Ще простіше замість цього лічильника помістити чотири простих тригера (це, в принципі, і є схема лічильника) – адже тоді в схемою будуть використовуватися лише однотипні, взаємозамінні елементи.

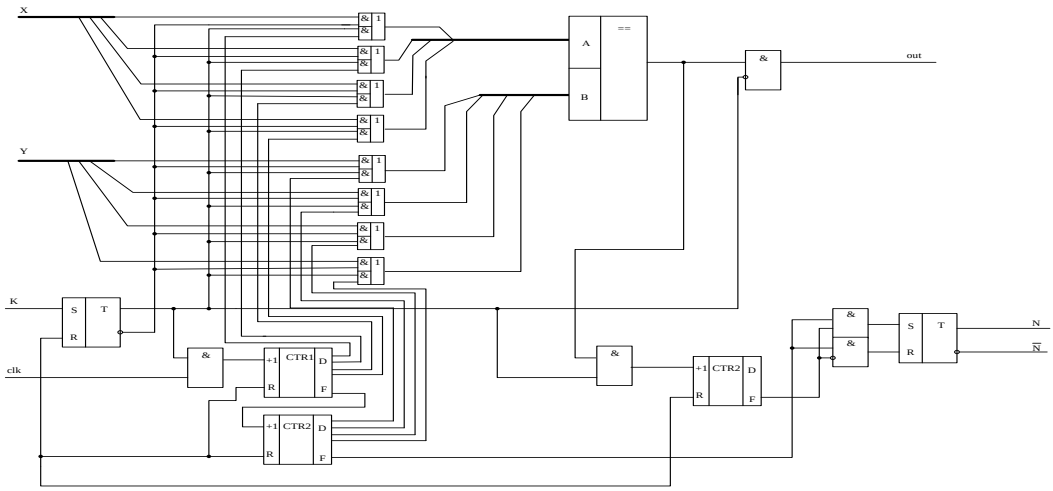


Рисунок 3.2 – Попередня електрична принципова схема системи

Значно спрощується права частина схеми: знайшовся спосіб помістити лише один елемент «І» замість лічильника, двох елементів «І» та тригера. Після всіх цих спрощень, схема, виконуючи ті самі функції, набуває значно більш простий і читабельний вигляд (див. Рисунок 3.3).

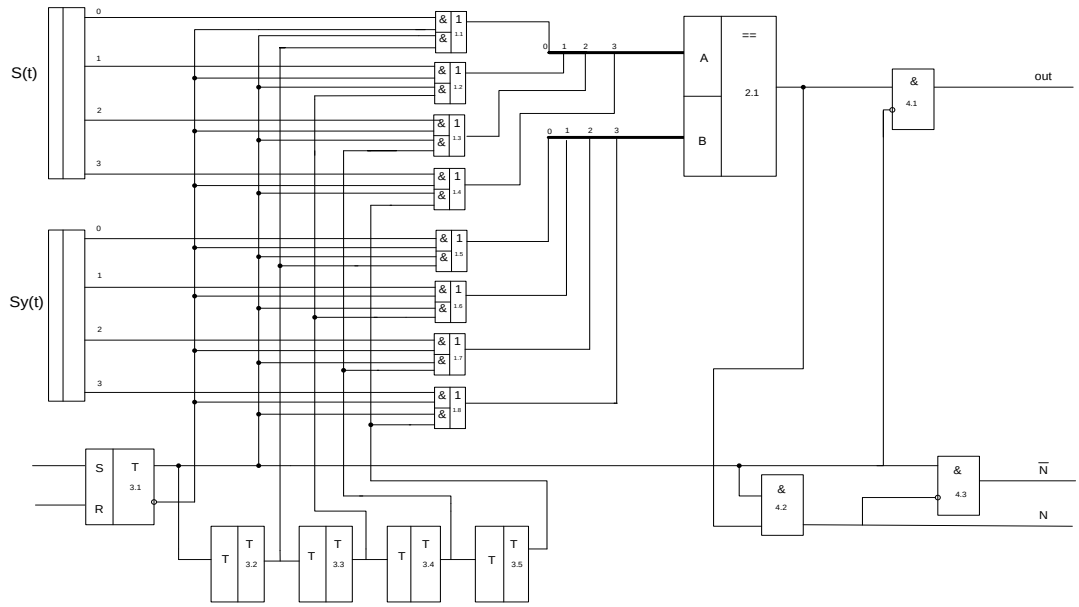


Рисунок 3.3 – ЕП схема контролю компаратора

Виробляємо набір даної схеми Quartus. Деякі елементи цього середовища проектування зображуються по-іншому. Наприклад, замість одного компаратора на вісім входів потрібно помістити чотири однотипні компаратори по два входи. Необхідно також вказати повну схему підключення тригерів, також змінюються деякі входи (див. рисунок 3.4).

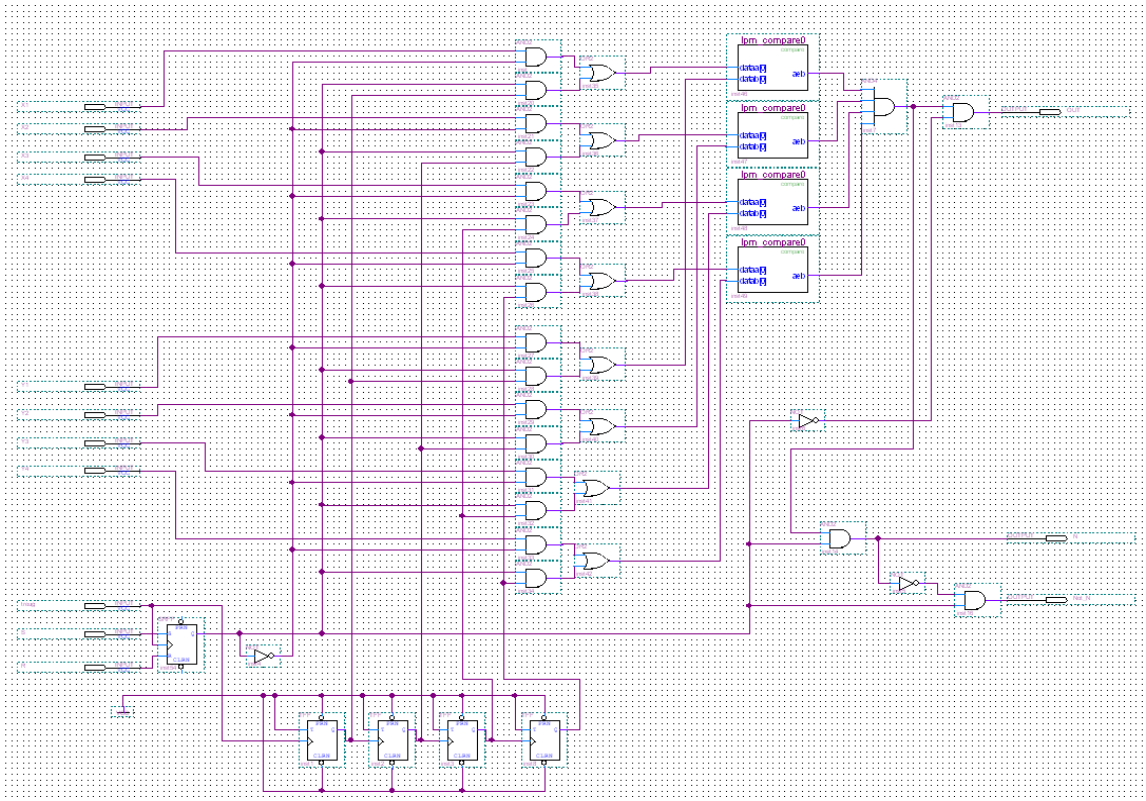


Рисунок 3.4 - ЕП схема системи, набрана в Quartus

Перевіряємо правильність роботи системи: створюємо Vector Waveform file, задаємо необхідні для перевірки входи та виходи, симулюємо та дивимось на результат (див. Рисунок 3.5).

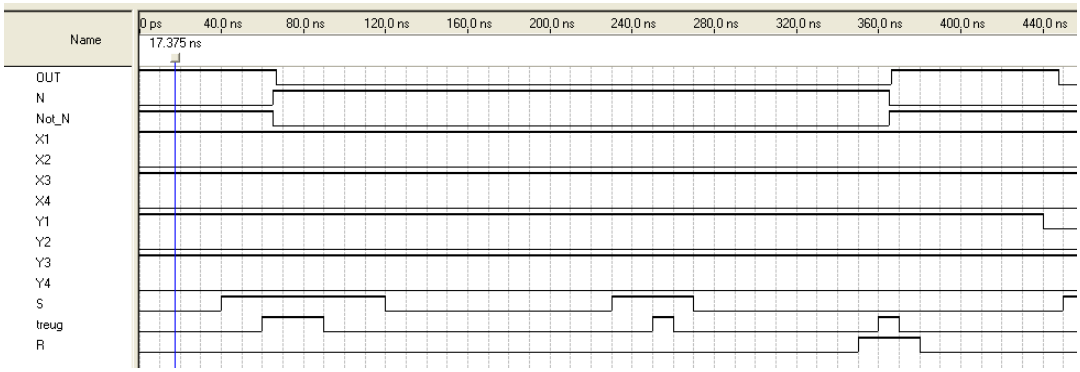


Рисунок 3.5 – Симулювання роботи системи

Три верхніх сигналу являють собою виходи пристрою, інші - входи, що задаються. Як можна побачити і компаратор, і система перевірки працюють справно. Затримка між входом та виходом становить лише 9 наносекунд. Набрану схему середовище проектування та програмування «Quartus» для себе представляє в наступному вигляді (див. рисунок 3.6). Це проміжний вигляд між створеною програмою та тим набором елементів «І» та «АБО», який буде занесений безпосередньо на ПЛІС.

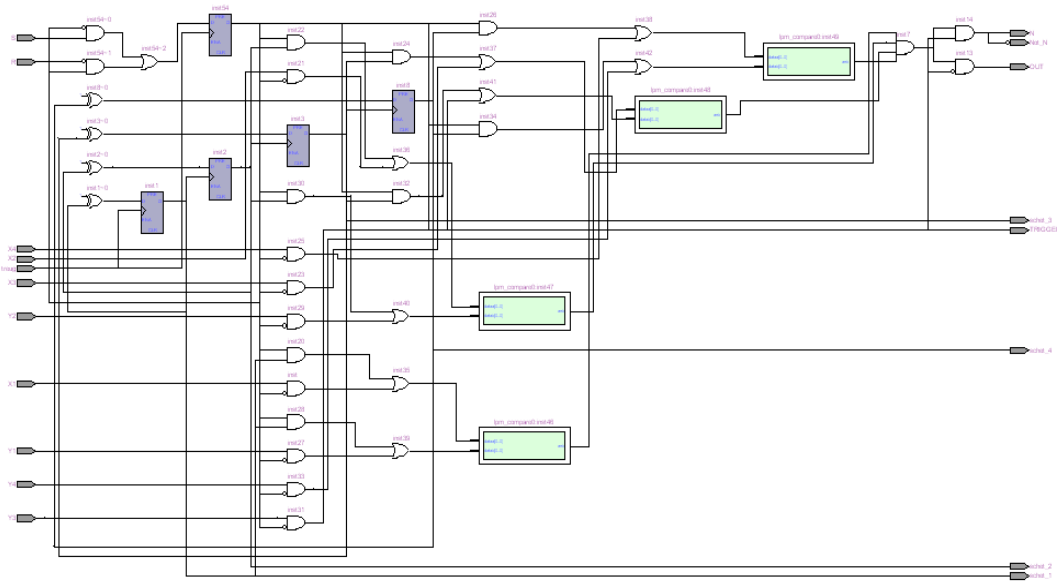


Рисунок 3.6 – ЕП схема створеного пристрою RTL Viewer

3.2 Огляд та порівняння основних світових виробників ПЛІС

Як відомо, при виборі елементної бази для високотехнологічних систем зазвичай керуються такими критеріями відбору:

– швидкодія;

– логічна ємність, достатня для реалізації алгоритму;

– схемотехнічні та конструктивні параметри ПЛІС, надійність, робочий діапазон температур, стійкість до іонізуючих випромінювань тощо;

– вартість володіння засобами розробки, що включає як вартість програмного забезпечення, так наявність та вартість апаратних засобів налагодження;

– вартість обладнання для програмування ПЛІС чи конфігураційних ПЗП;

– наявність методичної та технічної підтримки;

–наявність та надійність російських постачальників;

– вартість мікросхем.

Розглянемо із цих позицій продукцію провідних світових виробників ПЛІС, доступну на українському ринку.

Компанія Altera

Фірма Altera Corporation (101 Innovation Drive, San Jose, CA 95134, USA, www.altera.com) була заснована в червні 1983 року. Нині High-end продуктом цієї фірми є сімейство APEX20K, особливості архітектури якого згадувалися вище, а табл. 3.1 наведено основні параметри ПЛІС цього сімейства (крім того, Altera випускає CPLD сімейств APEX20K, MAX7000, MAX9000 та FPGA сімейств FLEX10K, FLEX8000, FLEX6000).

Таблиця 3.1

Основні характеристики ПЛІС сімейства APEX20K фірми ALTERA

	EP20K100	E20K160	EP20K200	EP20K300	EP20K400	EP20K600	EP20K1000
Максимальна кількість еквівалентних вентилів	263 000	404 000	526 000	728 000	1 052 000	1 537 000	2 670 000
Число логічних елементів	4 160	6 400	8 320	11 520	16 640	24 320	42 240
Вбудовані блоки пам'яті	26	40	52	72	104	152	264
Максимальний обсяг пам'яті, біт	53 248	81 920	106 496	147 456	212 992	311 296	540 672
Число макроосередків	416	640	832	1 152	1 664	2 432	4 224
Число висновків користувача	252	320	382	420	502	620	780

Компанія Altera є основним конкурентом компанії Xilinx, причому за всіма основними напрямками. Головне з них - це виробництво ПЛІС як типу FPGA, так і типу CPLD. У травні 2008 року Altera представила нове сімейство із серії Stratix високопродуктивних мікросхем типу FPGA – Stratix

IV, що працюють на 40-нм архітектурі. Для менш ресурсомістких завдань компанія Altera пропонує серію ПЛІС FPGA Cyclone, а як компроміс між продуктивними Stratix і недорогими Cyclone – серію Arria. Для мобільних пристроїв випускається серія Max на основі ПЛІС типу CPLD.

Також на додаток до цих мікросхем компанія випускає серію ASIC мікросхем HardCopy, розроблених як спеціалізовані аналоги відповідних FPGA Stratix.

Починаючи з серії Stratix III, у ПЛІС використовується технологія Programmable Power Technology, яка дозволяє варіювати режим роботи і, відповідно, споживану потужність логічних осередків залежно від необхідності швидкого виконання поставленого завдання.

Мікросхеми компанії Altera активно застосовуються в багатьох областях, наприклад, на ринку бездротових та провідних комунікацій, у військових технологіях, в галузі телемовлення, а також у різних мобільних пристроях.

Компанія Altera займається розробкою різноманітного програмного забезпечення для роботи з їх мікросхемами, серед яких основним програмним продуктом є пакет програм Quartus II, розглянутий у даній роботі, який надає різні засоби для проектування та аналізу структури мікросхем, а також для оптимізації витрат на споживану потужність.

Додатковим фактором при виборі ПЛІС Altera (крім того, що вони мають можливість перепрограмування безпосередньо в системі; і відносно високий ступінь інтеграції, що дозволяє розмістити цифровий пристрій в одному кристалі і тим самим знизити час і витрати на трасування та виробництво друкованих плат) є наявність достатньо розвинених безкоштовних версій САПР.

Крім того, ПЛІС фірми Altera випускаються з можливістю програмування у системі безпосередньо на платі. Для програмування та завантаження конфігурації пристроїв опубліковано схему завантажувального кабелю ByteBlaster і ByteBlasteMV. Слід зазначити, що нові конфігураційні ПЗП EPC2 дозволяють програмувати за допомогою цього пристрою, тим самим відпадає потреба в програматорі, що знижує вартість володіння технологією.

ПЛІС фірми Altera випускаються у комерційному та індустріальному

діапазоні температур.

Компанія Xilinx

Компанія Xilinx Inc. (2100 Logic Drive, San Jose, CA 95124-3400, USA, www.xilinx.com) була заснована в лютому 1984, її High-end продуктом є ПЛІС сімейства Virtex, основні характеристики яких представлені в таблиці 3.2.

Таблиця 3.2

Основні характеристики ПЛІС сімейства Virtex фірми XILINX

	XCV50	XCV100	XCV150	XCV200	XCV300	XCV400	XCV600	XCV800	XCV1000
Максимальна кількість еквівалентних вентилів	57 906	108 904	164 674	236 666	322 970	468 252	661 111	888 439	1 124 022
Число логічних елементів	1 728	2 700	3 888	5 292	6 912	10 800	15 552	21 168	27 648
Максимальний обсяг пам'яті, біт	24 576	38 400	55 296	75 264	98 304	153 600	221 184	301 056	393 216
Число висновків користувача	180	180	260	284	316	404	512	512	512

Архітектура сімейства Virtex характеризується широкою різноманітністю високошвидкісних трасувальних ресурсів, наявністю виділеного блокового ОЗУ, розвиненою логікою прискореного перенесення. ПЛІС цієї серії забезпечують високі швидкості міжкристального обміну – до 200 МГц (стандарт HSTL IV). Кристали серії Virtex за рахунок розвиненої технології виробництва та вдосконаленого процесу верифікації мають досить

низьку вартість (до 40% від еквівалентної вартості серії XC4000XL

Крім сімейства Virtex, Xilinx випускає FPGA сімейств XC3000A, XC4000E, Spartan, XC5200, а також CPLD XC9500 і серію CoolPLD, що мало споживає. Існує безкоштовна версія САПР – WebPACK, що підтримує CPLD XC9500 та CoolPLD, введення опису алгоритму за допомогою мови опису апаратури VHDL. Слід також зауважити, що Xilinx суттєво оновив модельний ряд як своїх ПЛІС, так і програмного забезпечення, яке розробляється за участю фірми Synopsys. Для ВНЗ передбачені значні знижки на ПЗ. ПЛІС Xilinx випускаються в комерційному та індустріальному діапазоні температур з військовою (Military) та космічною (Space) прийманням.

Корпорація Actel

Компанія Actel Corporation (955 East Arques Avenue, Sunnyvale, CA 94086-4533, USA, www.actel.com) була заснована в 1985 році. Особливістю ПЛІС Actel є застосування так званої Antifuse-технології, що є створенням металізованої перемички при програмуванні. Дана технологія забезпечує високу надійність та гнучкі ресурси трасування, а також не потрібне конфігураційне ПЗП. За цією технологією випускаються сімейства ACT1, ACT2, 1200XL, а також нові сімейства 54SX, A40MX і A42MX (з вбудованими модулями пам'яті), що мають хороші показники ціна / логічна ємність (ПЛІС, що замінює 300-350 корпусів ТТЛ, коштує 10 > 250 МГц). Дані ПЛІС є гарною альтернативою БМК при середньосерійному виробництві.

Нове сімейство ProASIC фірми Actel має ємність до 500 000 еквівалентних логічних вентилів. Його відмінною особливістю є енергонезалежність, завдяки застосуванню FLASH-технології, і наявність інтегрованого на кристалі пристрою.

На жаль, мікросхеми Actel, що випускаються за Antifuse-технологією, вимагають застосування спеціального програматора, вартість якого поки що дуже висока. Однак їх відрізняє висока надійність, тому вони є досить

перспективною базою для спеціальних застосувань. Так, ПЛІС серії RH1280 мають такі характеристики:

- допустима доза опромінення 300 000 РАД;
- логічна ємність 16 000 еквівалентних вентилів;
- швидкодія до 135 МГц.

ПЛІС даного типу були застосовані в марсоході в системі управління та обробки зображення цифрової відеокамери робота-марсохода та у формувачі кадру для передачі інформації на Землю. В даний час випущені радіаційностійкі ПЛІС нових сімейств. Компанія Actel робить ставку на виробництво невеликих та недорогих мікросхем типу FPGA, орієнтуючись на надійність таких ПЛІС. В цілому мікросхеми, що випускаються компанією Actel, можна розбити на два типи:

- із використанням flash-пам'яті;
- з одноразово програмованою пам'яттю (antifuse технологія).

Обидва типи мікросхем забезпечують високий рівень захищеності інформації як від несанкціонованого доступу, і від альфа- і нейтронного випромінювання. Також великою перевагою таких мікросхем є той факт, що вони не змінюються – не вимагають завантаження конфігурації архітектури ПЛІС щоразу при включенні живлення. Це означає, що вони готові до роботи відразу після запуску обладнання.

Оскільки компанія Actel займається виробництвом компактних і недорогих ПЛІС (у лютому цього року Actel стала випускати мікросхеми серії ProASIC3 за рекордно низькою ціною 99 центів), основними її покупцями є компанії, що займаються різними портативними пристроями та автомобільною промисловістю. Також завдяки описаним властивостям високої надійності та моментальної готовності до роботи мікросхеми компанії Actel використовуються у військовій та аерокосмічній областях.

Висновки за розділом 3

Завдяки поглибленому дослідженню структури програмованої логіки була розроблена схема пристрою для майбутніх тестів алгоритму. Був проведений аналіз та порівняння продукції передових виробників програмованої логіки на випадок. На базі цієї інформації можна зрозуміти, що на даний момент існують виробники програмованої логіки під усі потреби.

РОЗДІЛ 4.

РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ АЛГОРИТМУ ДЛЯ МІКРОПРОГРАМНОГО АВТОМАТУ

4.1 Вибір середовища програмування

На жаль, ПЛІС кожної фірми вимагають застосування своїх програмних пакетів, ступінь доступності яких також різний. Повні версії програмних продуктів всіх без винятку фірм є комерційними продуктами з вартістю від кількох сотень до кількох тисяч. е. Однак деякі фірми надають безкоштовні версії своїх програмних продуктів з деякими обмеженнями можливостей.

Основні характеристики пакету WebPACK ISE фірми Xilinx Програмні засоби WebPACK ISE є системою наскрізного проектування, яка реалізує всі етапи створення цифрового пристрою на базі ПЛІС, включаючи програмування кристала: розробка проекту, синтез, моделювання, трасування і завантаження в кристал. Версія 3.3WP8.0 САПР WebPACK ISE призначена для проектування цифрових пристроїв на базі ПЛІС виробництва Xilinx, що відносяться як сімейства CPLD: XC9500, XC9500XL, XC9500XV, XCR22V10, XCR3000 (XPLA1_0 та FPGA: Spartan™-II, Virtex™-E (тільки кристал XCV300E), Virtex-II (кристали 2V40, 2V80 та 2V250).

Можливості цього пакету:

- підтримка різних методів опису проектованих пристроїв

(графічних та текстових);

- можливість використання проектів, підготовлених в інших системах проектування, зокрема серед пакета Altera MAX+PlusII™; - наявність схемотехнічного редактора, укомплектованого набором великих бібліотек;

- інтелектуальні засоби створення HDL (Hardware Description Language) - описів, що формують шаблони на основі інформації, що

надається користувачем, для мов опису апаратури VHDL, Verilog™ і ABEL™ HDL;

- високоефективні засоби синтезу HDL-проектів, що підтримують мови VHDL, Verilog та ABEL HDL, з можливістю оптимізації;

- автоматичні засоби трасування проекту в кристали різних 1

сімейств ПЛІС Xilinx з урахуванням оптимізації проекту за різними параметрами; - засоби програмування кристалів сімейств ПЛІС Xilinx, виконаних за різною технологією (CPLD та FPGA), що підтримують кілька типів завантажувальних кабелів JTAG-інтерфейсу;

- наявність інтегрованого з пакетом САПР набору інструментів та утиліт інших фірм, що надають додаткові зручності у процесі проектування, що включає утиліту генерації тестових сигналів HDL Bencher™, програму моделювання ModelSim XE Starter™ та редактор діаграм станів StateCAD™ (рис. 4.1).

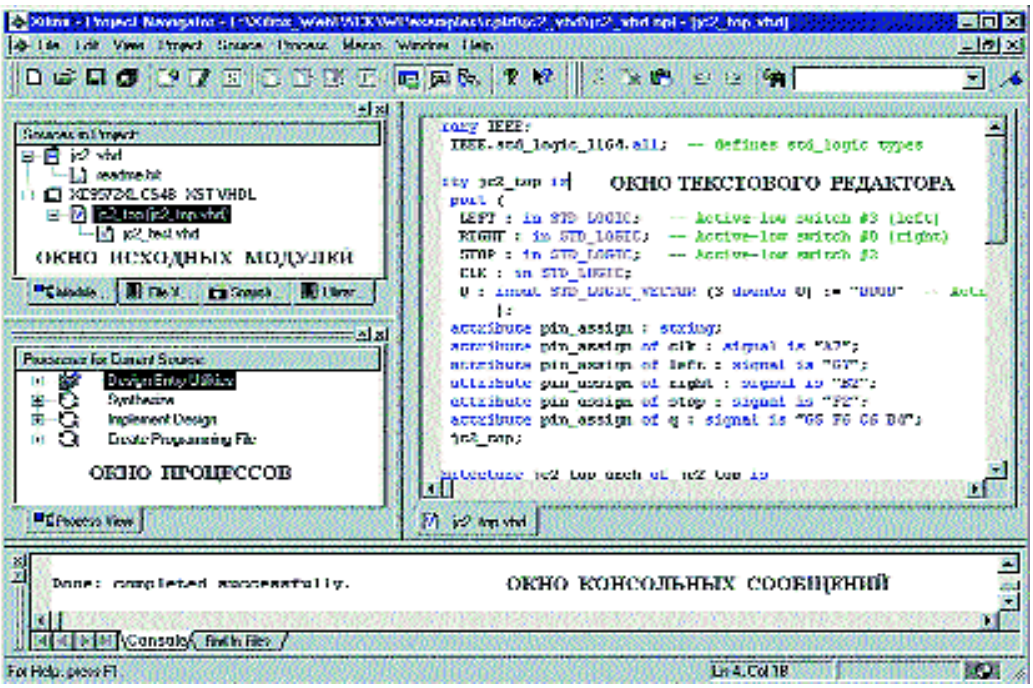


Рисунок 4.1– Основне вікно Навігатора проекту пакету WebPACK ISE

Для отримання програмного забезпечення WebPACK ISE необхідно зареєструватися на сайті www.xilinx.com. Після реєстрації виконайте процедуру копіювання модулів пакета на ПК, який буде використаний для розгортання САПР. Дистрибутив пакета виконаний у вигляді набору модулів, кожен з яких являє собою архів, що саморозпаковується. Після копіювання слід по черзі запустити виконання кожен із отриманих файлів. Після закінчення розпакування кожного архіву автоматично запускається програма встановлення відповідних модулів. Користувач повинен послідовно виконати всі інструкції з кожної програми інсталяції. Зверніть увагу на те, що після встановлення програми моделювання ModelSim XE Starter слід отримати файл ліцензії. Для цього необхідно запустити програму Licensing Wizard, яка збере необхідну для отримання ліцензійного коду інформацію про ПК. Файл із цією інформацією має бути надісланий електронною поштою. У той же час, пакет Quartus II, що розглядається далі, при вищій продуктивності, не вимагає таких довгих налаштувань для запуску.

Програмні пакети фірми Altera

Для розробки цифрових пристроїв на ПЛІС фірми Altera зараз використовується інтегроване середовище розробки цифрових пристроїв, комерційна версія якої називається Quartus II. Її безкоштовна та комерційна версії нічим не відрізняються функціонально. Відмінність між ними полягає лише в числі типів пристроїв, що підтримуються. Природно, що основна версія підтримує весь спектр ПЛІС, що випускаються фірмою, і навіть кілька типів ПЛІС конкуруючих фірм, тоді як безкоштовна не містить, наприклад, нове сімейство ПЛІС Arria II GZ.

Типи ПЛІС, що підтримуються безкоштовними версіями пакета Quartus 2, можна переглянути в таблиці 4.1. Програма має можливість змінити набрану інформацію та підлаштуватися практично під будь-яку з даних ПЛІС.

Таблиця 4.1

ПЛІС, що підтримуються середовищем програмування Quartus

Сім'я	Підтримувані ПЛІС
Max II, MAX 3000 A, MAX 7000B(S), Arria GX, Cyclone I–IV, FLEX 6000, ACEX	Усе
СТРАТІКС III	EP3SE50, EP3SL70
СТРАТІКС II	EP2S15
APEX II	EP2A15
APEX 20KE	EP20K30E, EP20K60E, EP20K100E, EP20K200E
FLEX 10 KE	EPF10K50S, EPF10K200S

За кількістю доступних ПЛІС Quartus значно випереджає інші системи програмування ПЛІС, у той час, як за іншими не менш ніж відповідає їхньому рівню

Таблиця 4.2

Основні характеристики сучасних ПЛІС

Параметри	МАКСИМУМ 3000	МАКСИМУМ 7000	FLEX 6000	ACEX 1K
Тип структури	CPLD	CPLD	FPGA	FPGA
Логічна ємність еквівалентних вентилів	600…5000	600…5000	10 000…24 000	10 000…100 000
Число програмованих висновків	34…158	36…164	102…218	130…330
Максимальна тактова частота, МГц	150…190	125…175	200	200
Орієнтовна вартість, в. е.	5…80	7…160	17…45	35…120

Як випливає з наведених у таблиці 4.2 даних, номенклатура мікросхем, що підтримуються безкоштовними версіями системи Quartus II, охоплює практично всю лінійку ПЛІС. Інтерфейс даного середовища проектування та програмування зображено рисунку 4.2.

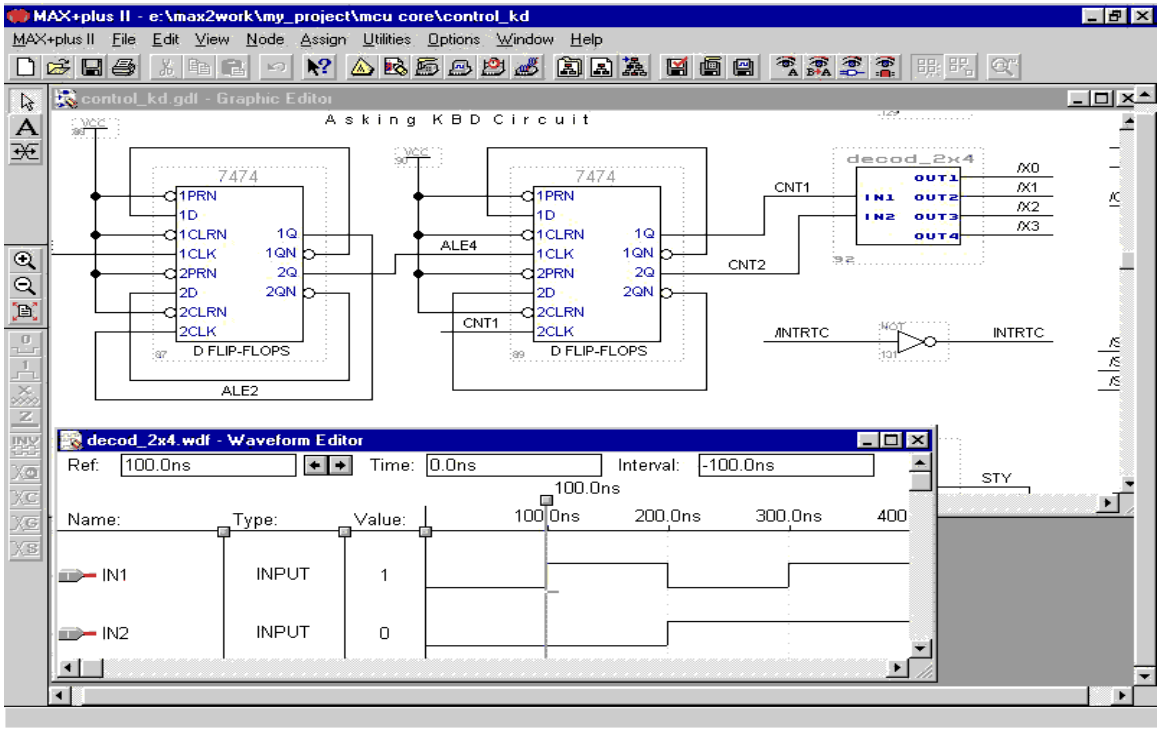


Рисунок 4.2 – Інтерфейс середовища програмування «Quartus»

Усі пакети інтегрованого середовища розробки цифрових пристроїв на ПЛІС фірми Altera мають такі загальносистемні властивості: 1. Забезпечують повний виробничий цикл випуску готових цифрових пристроїв на ПЛІС, що включає:

-розробку проекту пристрою (завдання необхідної логіки функціонування пристрою);

- перевірку коректності проекту та локалізацію помилок;

- синтез внутрішньої структури пристрою із мінімізацією необхідних ресурсів;

- компіляцію проекту (створення файлу для програмування чи конфігурування ПЛІС);

- моделювання процесу функціонування пристрою, тимчасовий аналіз та, нарешті, програмування (конфігурування) ПЛІС.

-Мають розвинені та зручні у користуванні засоби розробки проектів, до складу яких входять:

- редактор схем (Graphic Editor), дещо схожий на редактори САПР друкованих плат (ORCAD, PCAD), але набагато зручніший у роботі; - редактор тимчасових діаграм (Waveform Editor);

- текстовий редактор проектів мовою AHDL (Text Editor), найпотужніший, а й найскладніший засіб створення проектів; - (Всі редактори можуть використовуватися для створення різних частин основного проекту, який у цьому випадку має створюватись лише за допомогою редактора схем.)

-Мають велику бібліотеку елементів різного виду (логічних примітивів, аналогів дискретної логіки 74 серії, параметризованих логічних функцій), що дозволяють створювати проекти цифрових пристроїв будь-якої складності.

-Дозволяють розробляти проекти у вигляді багаторівневої ієрархії вкладених функціональних модулів та створювати за допомогою різних редакторів власні бібліотеки модулів, які можуть використовуватись у різних проектах.

-Забезпечують оптимальний синтез та мінімізацію використовуваних для реалізації проекту ресурсів мікросхем. 4. Забезпечують перевірку та локалізацію помилок при створенні вихідного проекту та при компіляції з урахуванням формальних та емпіричних правил проектування цифрових пристроїв та достатності наявних ресурсів, гарантуючи працездатність успішно скомпільованого проекту (але не гарантують від помилок при завданні розробником алгоритму функціонування пристрою).

-Мають можливість автоматичного вибору найбільш підходящої мікросхеми необхідного обсягу чи розподілу проекту між кількома мікросхемами малого обсягу.

-Забезпечують можливість закріплення призначених компілятором висновків мікросхем для постійного прив'язування до зовнішніх компонентів цільового пристрою або перепризначення висновків.

-Мають вбудовані засоби функціонального та тимчасового моделювання, що забезпечують швидку верифікацію та налагодження проектів.

-Забезпечують програмування та перепрограмування мікросхем, що мають вбудовану систему програмування, безпосередньо у складі кінцевого виробу через спеціальний кабель, що підключається або до LPTпорту (Byte Blaster), або до COM-порту (Bit Blaster) комп'ютера та технологічного 10-контактного з'єднувача, що встановлюється на платі виробу.

Схеми кабелів можна знайти на сайті фірми Altera в розділі Development Tools. Якщо на платі виробу встановлюється кілька ПЛІС з вбудованими системами програмування, всі вони можуть програмуватися через один технологічний роз'єм. Для програмування інших мікросхем необхідно додатково використовувати зовнішній програматор, який може підключатися до COM- або LPT-порту.

Всі пакети мають однаковий стандартний для Windows-додатків інтерфейс користувача. Вигляд робочого вікна з фрагментами проекту практичної схеми сканування матричної клавіатури, створеного з використанням редактора схем та модуля декодера 2x4, створеного в редакторі часових діаграм.

Резюмуючи викладене вище, можна сказати, що обидва програмні пакети придатні та зручні в експлуатації, проте Quartus перевершує конкурентів як за кількістю можливих для програмування ПЛІС, так і за гнучкістю структури, що дозволяє використовувати різні мови програмування високого та середнього рівня. Інтегровані середовища розробки цифрових пристроїв фірми Altera, і навіть їх безкоштовні версії є дуже ефективними інструментами для проектування багатьох типів ПЛІС.

4.2 Створення програми

Після створення пристрою «на блоках»: засобами пакета Block Diagram/Schematic files системи Quartus, спробуємо реалізувати той же пристрій програмними засобами (мова VHDL) і порівняємо їх за ефективністю кінцевого результату і трудомісткості.

Створення блок-схеми програми.

Для початку створимо «кістяк» – алгоритм, за яким працюватиме програма, а потім нарощуватимемо на нього інші частини.

Написання коду програми

Після розробки алгоритму, нарощуємо програму, точно дотримуючись створеного плану:

1. Початок створення програми:

Заходимо до Quartus, створюємо новий проект, створюємо у проекті VHDL-файл.

2. Підключення бібліотек:

```
library ieee; use
ieee.std_logic_1164.all;
```

3. Призначення входів та виходів компаратора з відмовостійкістю:

```
entity comparator is generic
(bits: natural:=4);
port (X: in bit_vector (bits-1 downto 0);
Y: in bit_vector (bits-1 downto 0);
K: in bit_vector (0 to 0);
output: out bit;
N: out bit;
```

notN: out bit); end comparator;

"X" і "Y" - чотирибітні входи порівнянь, "K" - однобітний ('так' або 'ні') вхід включення перевірки компаратора, "output" - вихід (рівний одиниці, якщо порівнювані значення рівні і нулю, якщо вони не рівні), "N" - вихід "норма" ('0' або '1'), "notN" - вихід "ненорма" ('0' або '1').

4. Створення функції з переведення бітових значень у числові:

function bin_to_int (signal rand: bit_vector (0 to bits-1))

return integer is variable sum: integer:=0; begin for i in 0

*to bits-1 loop if (rand(i)='1') then sum:=sum+2**(bits-1-i);*

else null; end if; end loop; return sum; end bin_to_int;

Ця функція потрібна нам для того, щоб ми могли порівняти один з одним багатобітні сигнали (для чотирьох біт завдання ще може бути вирішена іншим способом, без використання цієї функції, але якщо потрібно, скажімо, порівняти один з одним 32-бітні або 64-бітові сигнали, це буде найпростіший метод).

5. Призначення сигналів, які курсують в архітектурі пристрою:

signal A:integer; signal B:integer;

signal Z: bit_vector (bits-1 downto 0);

signal P: bit_vector (bits-1 downto 0);

Сигнали «A» і «B» задані у числовому форматі, адже саме через них ми порівнюватимемо значення, що надходять із різних джерел. Це будуть саме входи компаратора. Сигнали «Z» і «P» будуть сигналами, завдяки яким виконується функція контролю працездатності пристрою, тому їх розмірність аналогічна розмірності сигналів, що надходять ззовні «X» і «Y».

6. Перевірка на те, чи надійшов сигнал на вхід увімкнення режиму тестування:

if (K(0)<'1') or (K(0)>'1') then

7. Якщо попередня умова виконується, тобто режим тестування не включений, то нам потрібно виконати перетворення значень входів у числовий формат:

A<=bin_to_int(X); B<=bin_to_int(Y);

і порівняти їх:

if (A<B) then output <= '0'; notN<='0'; N<='0';

elsif (A>B) then output <='0'; notN<='0'; N<='0';

else output <= '1'; notN<='0'; N<='0'; end if;

Тут же виводяться значення виходів пристрою (№8 в блок-схемі). 9. Якщо ж режим тестування включений:

elsif (K(0)='1')

то проводиться перебір усіляких значень наших входів:

for i in 1 to 16 loop

if i=1 then z<= «0000»; end if; if i=2 then z<= «0001»; end if; if i=3 then z<= «0010»; end if; if i=4 then z<= «0011»; end if; if i=5 then z<= «0100»; end if; if i=6 then z<= «0101»; end if; if i=7 then z<= «0110»; end if; if i=8 then z<= «0111»; end if; if i=9 then z<= «1000»; end if; if i=10 then z<= «1001»; end if; if i=10

then z<= «1010»; end if; if i=11 then z<= «1011»; end if; if i=12 then z<= «1100»; end if; if i=13 then z<= «1101»; end if; if i=14 then z<= «1110»; end if; if i=15 then z<= «1111»; end if;

А потім привласнення тих же значень іншої змінної:

p<=z;

10, 11, 12, 13, 14.

Порівнюємо два значення через ті ж змінні «A» та «B»:

A<=bin_to_int(P); B<=bin_to_int(Z); if (A<B) then output <= '0'; notN<='1'; N<='0'; findi:=i; elsif (A>B) then output <='0'; notN<='1'; N<='0'; findi:=i; else output <= '0'; notN<='0'; N<='1';

Закриваємо весь цикл, якщо знайшли помилку в компараторі (видається значення, у якому виникла ця помилка), або якщо перебір значень закінчено:

exit loop when (findi>0) or (i=16);

15. Завершуємо написання програми, закриваючи всі дужки:

end if;

end loop;

end if; end

process; end

diplom;

Перевірка правильності функціонування програми

Для того, щоб перевірити правильність роботи програми, створюємо в Квартусі векторний файл (Vector Waveform file), налаштовуємо параметри симуляції і задаємо значення вхідних сигналів. Після проведення симуляції на виході ми маємо саме те, що хотіли (див. Рисунок 4.4).

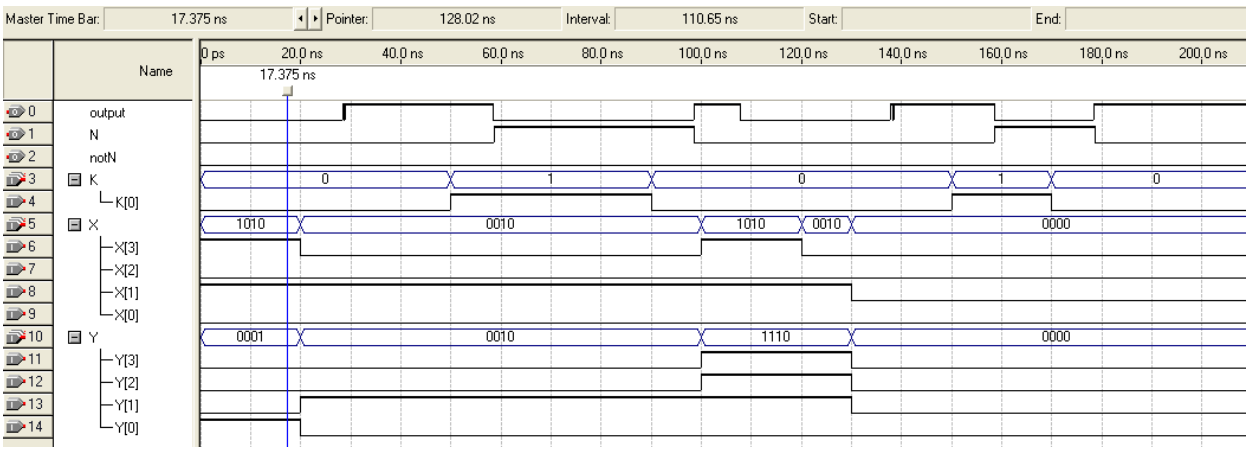


Рисунок 4.4 – Симулювання роботи програми

Спочатку всі три вихідні сигнали (вони виведені нагору) дорівнювали нулю, адже на вхід тестування «K» значення не подавалося, а порти «X» і «Y»

були нерівні ('1010' і '0001' відповідно). Потім значення входів стали однаковими ('0010'), і вихід пішла одиниця. Потім увімкнули вхід тестування, і він показав «норму». Загалом, як можна поспостерігати, програма виконує свої функції безпомилково. Затримка виходу від входу зісовує близько 9 наносекунд - так само, як і у програми, створеної схемотехнічно. Також можна поглянути на електричну принципову схему створеної програми у переглядачі Register Transfer Level. (рисунок 4.5)

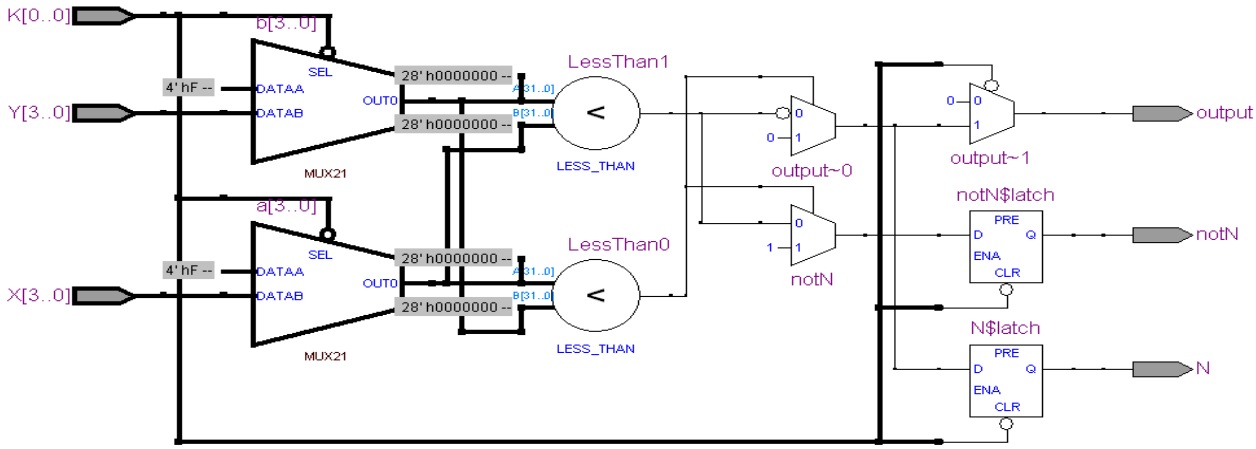


Рисунок 4.5 – Схема створеної програми у RTL Viewer

Як можна помітити за цим рисунком, схема, створена за програмою, значно компактніша, ніж розроблена (втім, деякі елементи в ній досить громіздкі). Це дозволяє судити про ефективність програми та її якість. Те, що створений проект досить хороший, підтверджує хоча б той факт, що якщо замінити кілька рядків на, здавалося б, абсолютно аналогічні вирази, а саме:

if (z= «1111») then z<= «0000»; end if; if (z= «1110») then z<= «1111»; end if; if (z= «1101») then z<= «1110»; end if; if (z= «1100») then z<= «1101»; end if; if (z= «1011») then z<= «1100»; end if; if (z= «1010») then z<= «1011»; end if; if (z= «1001») then z<= «1010»; end if; if (z= «1000») then z<= «1001»; end if; if (z= «0111») then z<= «1000»; end if; if (z= «0110») then z<= «0111»; end if; if (z= «0101») then z<= «0110»; end if; if (z= «0100») then z<= «0101»; end if; if (z= «0011») then z<= «0100»; end if; if (z= «0010») then z<= «0011»; end if;

if (z= «0001») then z<= «0010»; end if; if (z= «0000») then z<= «0001»; end if;

або змінну «Z» замість bit_vecor'a зробити стандартним сигналом і змінювати його за функцією (для і в циклі від 1 до 16

z:=z+1;), то в результаті в RTL Viewer наша схема з простої та компактної перетворюється на вкрай громіздку – понад 23 сторінки в RTL. До таких наслідків приводить не тільки ця, але й багато інших, здавалося б, рівнозначних замін у тілі циклу. Отже, можна сказати, що програма є досить компактною (хоча це й не було пріоритетним напрямом під час її розробки).

Висновки за розділом 4

Була розроблена програма та протестована в симуляції, що допомогло довести те, що програма є компактною, досить ефективною та може бути змінена під різні вимоги.

ВИСНОВОК

У ході кваліфікаційної роботи було проведено дослідження засобів паралельної обробки даних на програмованих логічних інтегральних схемах (ПЛІС). Було проведено аналіз існуючих методів паралельної обробки даних на ПЛІС та розроблено оптимальний алгоритм обробки даних на ПЛІС з використанням сучасних методів та технологій.

Результати проведеного дослідження показали, що використання програмованих логічних інтегральних схем для паралельної обробки даних є ефективним і має широкий потенціал у різних галузях, таких як медична діагностика, авіаційна промисловість, телекомунікації та інші.

Розроблений алгоритм обробки даних на ПЛІС дозволяє досягати значного прискорення обробки даних порівняно зі стандартними методами, що є важливим в практичній роботі.

Одним з ключових результатів роботи є розробка алгоритму обробки даних на ПЛІС, який є більш ефективним порівняно з існуючими методами. Це дає змогу використовувати розроблений алгоритм для ефективної обробки даних на ПЛІС у різних галузях.

Таким чином, дослідження засобів паралельної обробки даних на програмованих логічних інтегральних схемах є актуальним і важливим напрямом досліджень у галузі інформаційних технологій. Розроблений алгоритм обробки даних на ПЛІС може бути використаний для практичних цілей і внесе значний внесок у розвиток цієї галузі.

Апробація роботи була представлена на КМНТ-2023. IX Міжнар. наук.-техн. конф. Харків : ХНУ імені В.Н. Каразіна,.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Говорущенко Т.О. Типові цифрові схеми комп’ютерів. [Текст] Харків, 2019. стор. 12.

2. Методи обробки даних на ПЛІС. [Електронний ресурс]. – 2019.Режим доступу: http://ela.kpi.ua/bitstream/123456789/19096/1/Klymenko_aref.pdf. – Дата доступу: квітень 2023.

3. Однокристальні мікрокомп’ютери. [Електронний ресурс]. - 2019. Режим доступу: http://sh.instone.com.ua/tema2.6.php. – Дата доступу: квітень 2023.

4. Типові цифрові схеми комп’ютерів. [Електронний ресурс]. – 2019. Режим доступу: https://msn.khnu.km.ua/pluginfile.php/147595/mod_resource/content/2/Lekz10.pdf. – Дата доступу: квітень 2023.

5. Топ-5 найбільш небезпечніших підприємств України. [Електронний ресурс]. - 2020. Режим доступу: http://firtka.if.ua/blog/view/top5najbilsnebezpecnih-virobnictv-v-ukraini4013. – Дата доступу: квітень 2023.

6. Потенційно небезпечні виробництва та вимоги до їх розміщення. [Електронний ресурс]. 2021. Режим доступу: https://studopedia.org/420102.html. – Дата доступу: квітень 2023.

7. Характеристики ПЛІС. [Електронний ресурс]. 2020. Режим доступу: https://msn.khnu.km.ua/pluginfile.php/147595/mod_resource/content/2/Lekz10.pdf. – Дата доступу: квітень 2023.

8. Вступ до Алгоритмів 1990(перше видання) 2019(видання українською)-Дата доступу: квітень 2023

9. Xilinx Documentation [Електронний Ресурс]-2023- Дата доступу: квітень 2023

10. Зеленева И.Я. Сравнительный анализ способов реализации управляющих алгоритмов в базисе FPGA / И.Я. Зеленева, С.С. Грушко, А.А. Котенко, В.В. Зеленько // Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: Тези доповідей IX Міжнародної науково-практичної конференції [Електронний ресурс] Запоріжжя : ЗНТУ, 2018. с. 91 – 92

11. Грушко С.С. Аналіз методів зменшення апаратурних витрат при реалізації схем суміщених мікропрограмних автоматів на CPLD / С.С. Грушко, І.Я. Зеленьова // Матеріали Міжнародної науково-практичної конференції «Електротехнічні та комп'ютерні системи: теорія та практика», Одеса, 29 травня – 1 червня, 2018

12. Грушко С.С. Оптимізація апаратурних витрат при реалізації алгоритмів керування в мікросхемах XILINX / С.С. Грушко, А.О. Федько //Комп'ютерні інтелектуальні системи та мережі. Матеріали XI Всеукраїнської науково практичної WEB конфе-ренції аспірантів, студентів та молодих вчених (21-23 березня 2018 р.). – Кривий Ріг: ДВНЗ «Криворізький національний університет», 2018. 250 с. С.100-103.

13. Грушко С.С. Класифікація структур суміщених мікропрограмних автоматів при реалізації в базисі FPGA / С.С. Грушко // Вчені записки Таврійського національного університету імені В.І.Вернадського. Серія: Технічні науки, Київ, 2018, Том 29 (68) №1 2018. Частина 1. – С. 131 –136.

14. Грушко С.С. Аналіз методів зменшення апаратурних витрат при реалізації схем суміщених мікропрограмних автоматів на CPLD / С.С. Грушко, І.Я. Зеленьова // Електротехнічні та комп'ютерні системи. 2018. №2. С. 161 – 169.

15. Грушко С.С. Реалізація об'єднаного мікропрограмного автомата на програмованих логічних мікросхемах / С.С. Грушко// Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: 20 Тези доповідей IX Міжнародної науково-практичної конференції [Електронний ресурс] Запоріжжя : ЗНТУ, 2018. с. 87 – 89.

16. Реалізація на програмованій вентильній матриці об'єднаного скінченного автомата з лічильником / А. Баркалов, Л. Тітаренко, І. Зеленева, С. Грушко // Збірник тез доповідей 2018 року 9-ї міжнародної конференції IEEE з надійних систем, служб та технологій DESSERT'2018, Україна, Київ, 24-27 травня 2018 року.

17. Altera Documentation [Електронний ресурс] 2023-Дата доступу квітень 2023

18. "Digital Design and Computer Architecture" by David Harris, Sarah L. Harris:2019-Дата доступу квітень 2023

19. "Microprocessor Systems Design: 68000 Family Hardware, Software, and Interfacing" by Alan Clements: 2019-Дата доступу квітень 2023

20. FPGA Prototyping by Verilog Examples" by Pong P. Chu: 2019-Дата доступу квітень 2023

ДОДАТОК А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **Магістр**
Галузь знань: **12 – Інформаційні технології**
Спеціальність: **123 «Комп'ютерна інженерія»**

ЗАТВЕРДЖУЮ
Завідувач кафедри теоретичної та прикладної системотехніки
_____**д.т.н., проф. Шматков С. І.**

«08 » грудня 2022 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЕКТ)

Прокоф'єв Сергій Володимирович
(прізвище, ім'я, по батькові студента)

1. Тема роботи: «Методи зменшення апаратурних витрат для суміщених мікропрограмних автоматів на мікросхемах програмованої логіки»

керівник роботи Мірошник Марина Анатоліївна, д.т.н, професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «10» листопада 2023 року № 4101-5/3197

2. Строк подання студентом роботи 28 .11.2023

3. Перелік питань, які потрібно розробити:
   1) Огляд та аналіз методів паралельної обробки даних на основі ПЛІС;
   2) Розгляд аналогів на ринку;
   3) Розробка схеми ПЛІС для визначених потреб;
   4) Програмна реалізація методу.

2

План роботи

№ з/п	Назви етапів роботи	Термін виконання виконання етапів роботи
1	Підбір та узгодження літератури за темою роботи	09.12.2022-10.05.2022
2	Аналіз ринку на наявність рішення обраної проблеми	09.12.2022-13.06.2023
3	Дослідження методів оптимізації алгоритму для ПЛІС	14.06.2023-21.09.2023
4	Підбір програмного середовища для подальшої роботи	22.09.2023-01.10.2023
5	Розробка оптимізованого алгоритму для ПЛІС	02.10.2023-04.10.2023
6	Дослідження платформи для тестування	04.10.2023-10.10.2023
7	Розробка схеми ПЛІС для майбутнього тестування	05.10.2023-12.10.2023
8	Розробка архітектури програми	12.10.2023-17.10.2023
9	Тестування програми	27.10.2023-31.10.2023
10	Виправлення помилок в роботі програми	01.11.2023-14.11.2023
11	Розробка та оформлення пояснювальної записки	15.11.2023-28.11.2023
12	Подання дипломної роботи керівнику та рецензентам	29.11.2023-04.12.2023

Дата видачі завдання 06.12.2022р

Студент С.В. Прокоф'єв _____
ініціали, прізвище підпис

Керівник роботи М.А. Мірошник _____
ініціали, прізвище підпис

ДОДАТОК Б

Технічне завдання

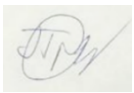
На розробку програмного виробу: “Методи зменшення апаратурних витрат для суміщених мікропрограмних автоматів на мікросхемах програмованої логіки**”**

Назва розділу	**Назва і зміст підрозділу**
1. Введення	1.1. Назва програмного виробу: комп’ютерна модель алгоритму програмування ПЛІС для мікропрограмних автоматів. 1.2. Галузь застосування: мобільні технології, інтернет речі, вбудовані системи.
2. Підстава для розробки	2.1. Навчальний план за спеціальністю 123 – «Комп’ютерна інженерія» 2.2. Завдання на кваліфікаційну роботу магістра затверджено наказом ректора No 4101-5/3197 від 10.11.2023.
3. Призначення розробки	3.1. Мета розробки програмного виробу: розробка покращенного алгоритму для зменшення апаратурних витрат. 3.2. Призначення програмного виробу: для підвищення ефективності використовуваних мікропрограмних автоматів на мікросхемах програмованої логіки.

4. Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик : мінімальна похибка при роботі, ефективність 4.2. Вимоги до надійності: запобігання помилкам: мінімізація помилкових позитивних та негативних результатів. 4.3. Вимоги до умов експлуатації: сумісність з різними платформами: модель має бути сумісною з різними середовищами для розробки, 4.4. Вимоги до складу і параметрів технічних засобів: обладнання для симуляції моделі, реальний аналог ПЛІС для тестування. 4.5. Вимоги до інформаційної та програмної сумісності: інтеграція з іншими системами: здатність налаштування під різні вимоги. 4.6. Вимоги до транспортування і зберігання: умови зберігання: вказівки для зберігання унікальності персональних даних. 4.7. Спеціальні вимоги:

5. Вимоги до програмної документації.	Програмною документацією до виробу «комп'ютерна модель покращеного алгоритму роботи ПЛІС для мікропрограмних автоматів» вважати: 1) Справжнє Технічне завдання на розробку програмного виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи). 2) Програму і методику випробувань розробленого програмного виробу (представити у вигляді Додатку В до пояснювальної записки до кваліфікаційної роботи). 3) Опис програмного виробу (представити в розділі 2 пояснювальної записки до кваліфікаційної роботи). 4) Текст програми (представити в Додатку Г до пояснювальної записки до кваліфікаційної роботи).	
6. Техніко-економічні показники	Орієнтовна оцінка ефективності та якості виконуваного аналізу: мінімальне підвищення ефективності має бути не менш 20%.	
7. Стадії і етапи розробки	Огляд методів програмування ПЛІС	08.12.2022
	Вибір та обґрунтування методу Програмування	01.01.2023
	Вибір та обґрунтування Програмного середовища	10.02.2023
	Розробка алгоритму програмування ПЛІС	20.03.2023
	Тестування та симуляція алгоритму	01.05.2023
	Опис програмного забезпечення для вирішення задачі	05.07.2023
	Порівняння даних ефективності розробленого алгоритму	10.08.2023
	Огляд та аналіз отриманих результатів	15.09.2023

	Підготовка та розробка програмної документації	10.10.2023
	Розробка та оформлення пояснювальної записки	22.11.2023
8. Порядок контролю і приймання	В даному розділі повинні бути вказані загальні вимоги до приймання розробленого програмного виробу наприклад: 1) Перевірка ходу розробки програмного виробу. Керівнику робіт виконувати 1 раз в 3 тижні. 2) Випробування програмного виробу відповідно до Програми і методики випробувань провести на базі комп'ютерного класу. 3) Захист розробленого програмного виробу провести на засіданні атестаційної комісії. 4) Пояснювальну записку надати на паперових носіях в одному примірнику, в електронному вигляді - на CD-диску в одному екземплярі.	

Виконавець: студент групи КІ-61 , Прокоф’єв Сергій Володимирович 

Замовник: д.т.н , професор ,Мірошник Марина Анатоліївна 

ДОДАТОК В

Програма і методика випробувань

програмного виробу

«Модель алгоритму програмування ПЛІС для суміщених мікропрограмних автоматів з ціллю зменшення апаратурних витрат»

1. Об’єкт випробувань

1.1 Найменування випробовуваного програмного виробу: «Модель алгоритму програмування ПЛІС для суміщених мікропрограмних автоматів з ціллю зменшення апаратурних витрат»

1.2 Галузь застосування: оцінювання структури комп’ютерних мереж та прогнозування їх продуктивності і ефективності

1.3 Умовне позначення розробки (при необхідності): відсутнє

2. Мета випробувань

Зменшення апаратурних витрат за допомогою покращення алгоритму програмування

3. Загальні положення

3.1 Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії.

3.2 Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться дистанційно в період роботи атестаційної комісії.

3.3 Обсяг випробувань

Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї програми і методики випробувань.

3.4 Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання за участю Замовника, Виконавця та інших осіб, присутніх на засіданні в дистанційному режимі.

4. Вимоги до програми або програмного виробу

Модель повинна:

1) Представляти з себе програмний алгоритм роботи ПЛІС

2) Мати можливість модифікації

3) Забезпечувати можливість використання на різних платформах

4.2 Вимоги до надійності:

1) Програма повинна видавати правильний результат

2) Програма дозволяє змінювати компоненти без втрати працездатності

4.3 Вимоги до умов експлуатації:

1) Умови експлуатації відповідають умовам експлуатації будь-якого персонального комп'ютера, який сумісний з програмою

1) Для використання програми потрібно ознайомитись із особливостями формату даних, і як змінювати гіперпараметри мережі

4.4 Вимоги до складу і параметрів технічних засобів: комп'ютер або ноутбук із 8 ГБ оперативної пам'яті та процесором не нижче Intel Core i3 7-го покоління.

4.5. Вимоги до інформаційної та програмної сумісності: ОС Linux або Windows 8/10, підтримка WEBPack ISE, RTL Viewer.

4.6. Вимоги до маркування та упаковки: відсутні

4.7. Вимоги до транспортування і зберігання: відсутні

4.8. Спеціальні вимоги: відсутні.

5. Вимоги до програмної документації

Програмною документацією до виробу «Модель алгоритму програмування ПЛІС для суміщених мікропрограмних автоматів з ціллю зменшення апаратурних витрат» вважати:

1) Справжнє Технічне завдання на розробку програмного виробу (представити у вигляді Додатку Г до пояснювальної записки до

кваліфікаційної роботи).

2) Програму і методику випробувань розробленого програмного виробу (представити у вигляді Додатку В до пояснювальної записки до кваліфікаційної роботи).

3) Опис програмного виробу (представити в розділах 3 пояснювальної записки до кваліфікаційної роботи).

4) Текст програми (представити в Додатку Г до пояснювальної записки до кваліфікаційної роботи).

6. Засоби і порядок випробувань

6.1 Засоби випробувань

Для проведення випробувань необхідний проект для розробки програмної моделі буде введен в програмне середовище для подальших випробувань.

Випробовування проводиться на технічних засобах, таких як персональний комп'ютер в повній комплектації, чи ноутбук, який має не менше ніж 8Гб оперативної пам'яті, та рекомендований мінімальний процесор Intel Core І3-7 го покоління. Підтримовані операційні системи "Windows" та "Linux".

6.2 Порядок проведення випробувань

1) Перевірка програмної докуметації

1.1 Перевірка складу програмної документації. Перевірку здійснюваи за критерієм наявності, представленої в ТЗ документації.

1.2 Критерієм успішності тесту вважати відповідність наявної документації згідно зі списком в ТЗ та відповідність якості наявної документації згідно з вимогами ЕСПД.

1.3. Перевірка якості програмної документації. Перевірку здійснювати за критерієм відповідності вимогам ЕСПД.

1.4. Модель працює відповідно до умов експлуатації ОС «Windows» та «Linux» із встановленим пакетом WEBPack ISE

Тест 1:

Написання коду програми

Після розробки алгоритму, нарощуємо програму, точно дотримуючись створеного плану:

1. Початок створення програми:

Заходимо до Quartus, створюємо новий проект, створюємо у проекті VHDL-файл.

2. Підключення бібліотек:

library ieee; use

ieee.std_logic_1164.all;

3. Призначення входів та виходів компаратора з відмовостійкістю:

entity comparator is generic

(bits: natural:=4);

port (X: in bit_vector (bits-1 downto 0);

Y: in bit_vector (bits-1 downto 0);

K: in bit_vector (0 to 0);

output: out bit;

N: out bit;

notN: out bit); end comparator;

"X" і "Y" - чотирибітні входи порівнянь, "K" - однобітний ('так' або 'ні') вхід включення перевірки компаратора, "output" - вихід (рівний одиниці, якщо порівнювані значення рівні і нулю, якщо вони не рівні), "N" - вихід "норма" ('0' або '1'), "notN" - вихід "ненорма" ('0' або '1').

4. Створення функції з переведення бітових значень у числові:

function bin_to_int (signal rand: bit_vector (0 to bits-1))

return integer is variable sum: integer:=0; begin for i in 0

*to bits-1 loop if (rand(i)='1') then sum:=sum+2**(bits-1-i);*

else null; end if; end loop; return sum; end bin_to_int;

Ця функція потрібна нам для того, щоб ми могли порівняти один з одним багатобітні сигнали (для чотирьох біт завдання ще може бути вирішена іншим способом, без використання цієї функції, але якщо потрібно, скажімо, порівняти один з одним 32-бітні або 64-бітові сигнали, це буде найпростіший метод).

5. Призначення сигналів, які курсують в архітектурі пристрою:

signal A:integer; signal B:integer;

signal Z: bit_vector (bits-1 downto 0);
signal P: bit_vector (bits-1 downto 0);

Сигнали «A» і «B» задані у числовому форматі, адже саме через них ми порівнюватимемо значення, що надходять із різних джерел.

Це будуть саме входи компаратора. Сигнали «Z» і «P» будуть сигналами, завдяки яким виконується функція контролю

працездатності пристрою, тому їх розмірність аналогічна розмірності сигналів, що надходять ззовні «X» і «Y».

6. Перевірка на те, чи надійшов сигнал на вхід увімкнення режиму тестування:

if (K(0)<'1') or (K(0)>'1') then

7. Якщо попередня умова виконується, тобто режим тестування не включений, то нам потрібно виконати перетворення значень входів у числовий формат:

A<=bin_to_int(X); B<=bin_to_int(Y);

і порівняти їх:

if (A<B) then output <= '0'; notN<='0'; N<='0';

elsif (A>B) then output <='0'; notN<='0'; N<='0';

else output <= '1'; notN<='0'; N<='0'; end if;

Тут же виводяться значення виходів пристрою (№8 в блок-схемі). 9. Якщо ж режим тестування включений:

elsif (K(0)='1')

то проводиться перебір усіляких значень наших входів:

for i in 1 to 16 loop

if i=1 then z<= «0000»; end if; if i=2 then z<= «0001»; end if; if i=3 then z<= «0010»; end if; if i=4

```
then z<= «0011»; end if; if  i=5 then z<= «0100»; end
if; if i=6 then z<= «0101»; end if; if  i=7 then z<=
«0110»; end if; if i=8 then z<= «0111»; end if; if  i=9
then z<= «1000»; end if; if i=10 then z<= «1001»; end
if; if  i=10 then z<= «1010»; end if; if i=11 then z<=
«1011»; end if; if  i=12 then z<= «1100»; end if; if i=13
then z<= «1101»; end if; if  i=14 then z<= «1110»; end
if; if i=15 then z<= «1111»; end if;
```

А потім привласнення тих же значень іншої змінної:

```
p<=z;
```

10, 11, 12, 13, 14. Порівнюємо два значення через ті ж змінні «A» та «B»:

```
A<=bin_to_int(P);   B<=bin_to_int(Z);   if
(A<B) then output  <= '0'; notN<='1'; N<='0';
findi:=i; elsif (A>B) then
output <='0'; notN<='1'; N<='0'; findi:=i;
else output <=  '0'; notN<='0'; N<='1';
```

Закриваємо весь цикл, якщо знайшли помилку в компараторі (видається значення, у якому виникла ця помилка), або якщо перебір значень закінчено:

```
exit loop when (findi>0) or (i=16);
```

15. Завершуємо написання програми, закриваючи всі дужки:

end if;

end loop;

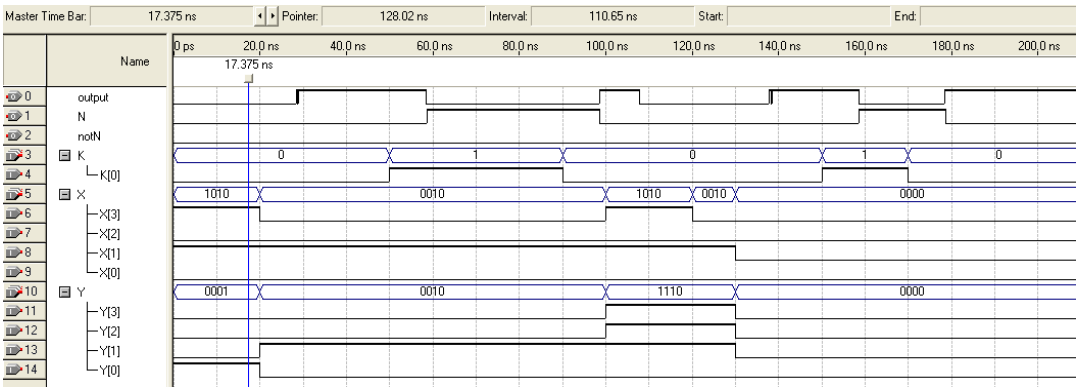
end if; end

process; end

diplom;

Перевірка правильності функціонування програми

Для того, щоб перевірити правильність роботи програми, створюємо в Квартусі векторний файл (Vector Waveform file), налаштовуємо параметри симуляції і задаємо значення вхідних сигналів. Після проведення симуляції на виході ми маємо саме те, що хотіли (див. Рисунок 4.4).

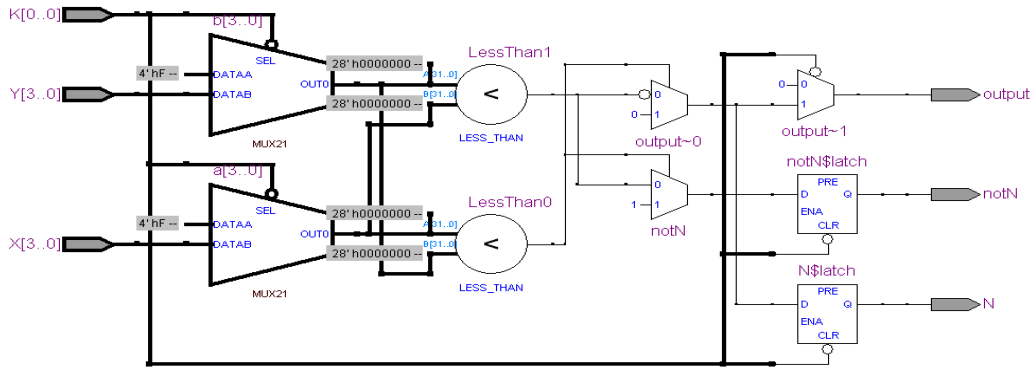


В1 – Симулювання роботи програми

Спочатку всі три вихідні сигнали (вони виведені нагору) дорівнювали нулю, адже на вхід тестування «К» значення не подавалося, а порти «Х» і «Y»

були нерівні ('1010' і '0001' відповідно). Потім значення входів стали однаковими ('0010'), і вихід пішла одиниця. Потім увімкнули вхід тестування, і він показав «норму». Загалом, як можна поспостерігати, програма виконує свої функції безпомилково. Затримка виходу від входу зісовує близько 9 наносекунд - так само,

як і у програми, створеної схемотехнічно. Також можна поглянути на електричну принципову схему створеної програми у переглядачі Register Transfer Level:



B2 – Схема створеної програми у RTL Viewer

Як можна помітити за цим рисунком, схема, створена за програмою, значно компактніша, ніж розроблена (втім, деякі елементи в ній досить громіздкі). Це дозволяє судити про ефективність програми та її якість. Те, що створений проект досить хороший, підтверджує хоча б той факт, що якщо замінити кілька рядків на, здавалося б, абсолютно аналогічні вирази, а саме:

if (z= «1111») then z<= «0000»; end if; if (z= «1110») then z<= «1111»; end if; if (z= «1101») then z<= «1110»; end if; if (z= «1100») then z<= «1101»; end if; if (z= «1011») then z<= «1100»; end if; if (z= «1010») then z<= «1011»; end if; if (z= «1001») then z<= «1010»; end if; if (z= «1000») then z<= «1001»; end if; if (z= «0111») then z<= «1000»; end if; if (z= «0110») then z<= «0111»; end if; if (z= «0101») then z<= «0110»; end if; if (z= «0100») then z<= «0101»; end if; if (z= «0011») then z<= «0100»; end if; if (z= «0010») then z<= «0011»;

end if;

if (z= «0001») then z<= «0010»; end if; if (z= «0000») then z<= «0001»; end if;

або змінну «Z» замість bit_vecor'a зробити стандартним сигналом і змінювати його за функцією (*для і в циклі від 1 до 16 z:=z+1;*), то в результаті в RTL Viewer наша схема з простої та компактної перетворюється на вкрай громіздку – понад 23 сторінки в RTL.

До таких наслідків приводить не тільки ця, але й багато інших, здавалося б, рівнозначних замін у тілі циклу. Отже, можна сказати, що програма є досить компактною (хоча це й не було пріоритетним напрямом під час її розробки).

Висновки: працездатність коду підтверджена, компактність програми підтверджує її ефективність.

ДОДАТОК Г

ЛІСТИНГ КОДУ

```
library ieee;
use ieee.std_logic_1164.all;
entity comparator is
generic (bits: natural:=4);
port (X:       in bit_vector (bits-1 downto 0);
Y:      in bit_vector (bits-1 downto 0);
K: in bit_vector (0 to 0);
output:        out bit;
N: out bit;
notN:          out bit); end comparator;
function bin_to_int (signal rand: bit_vector (0 to bits-1))
return integer is
variable sum: integer:=0;
begin for i in 0 to bits-1 loop
if (rand(i)='1') then sum:=sum+2**(bits-1-i);
else null;      end if; end loop;    return sum;  end bin_to_int;
signal A:integer;
signal B:integer;
signal Z: bit_vector (bits-1 downto 0);
signal P: bit_vector (bits-1 downto 0);
if (K(0)<'1') or (K(0)>'1') then
A<=bin_to_int(X); B<=bin_to_int(Y);
if (A<B) then output <= '0'; notN<='0'; N<='0';
elsif (A>B) then output <='0'; notN<='0'; N<='0';
else    output <= '1'; notN<='0'; N<='0';
end if;
```

```
elsif (K(0)='1')
for i in 1 to 16 loop
if i=1 then z<= «0000»; end if; if i=2 then z<= «0001»; end if;
if i=3 then   z<= «0010»; end if; if i=4 then z<= «0011»; end if;
if i=5 then z<= «0100»; end if; if i=6 then   z<= «0101»; end if;
if i=7 then   z<= «0110»; end if; if i=8 then z<= «0111»; end if;
if i=9 then   z<= «1000»; end if; if i=10 then z<= «1001»; end if;
if i=10 then z<= «1010»; end if; if i=11 then z<= «1011»; end if;
if i=12 then z<= «1100»; end if; if i=13 then z<= «1101»; end if;
if i=14 then z<= «1110»; end if; if i=15 then z<= «1111»; end if;
p<=z;
A<=bin_to_int(P); B<=bin_to_int(Z);
if (A<B) then output <= '0'; notN<='1'; N<='0'; findi:=i;
elsif (A>B) then    output <='0'; notN<='1'; N<='0'; findi:=i;
else   output <= '0'; notN<='0'; N<='1';
exit loop when (findi>0) or (i=16);
end if;
end loop;
end if;
end process;
end diplom;
\\
\\
```

Приклад коду для програми яка виводитиме інформацію про порівняння значень напруги на двох ніжках різних портів на рідкокристалічний дисплей:

```
/*****************************************
```

```
**** ****  Project: COMPARATOR Chip type: ATmega16
*********************************************
**** ***/  #include <mega16.h>
#include <delay.h>
#include <stdio.h>
#asm
equ __lcd_port=0x15; ПОРТ
#endasm
#include <lcd.h> void
main(void) {
lcd_heat(16); //ініціалізація LCD lcd_gotoxy
(0,0); lcd_putsf («Razrabotal:»);
delay_ms(500);
lcd_gotoxy (0,1); lcd_putsf («Левичев Артем»); delay_ms(2000);
lcd_clear(); lcd_gotoxy (0,0); delay_ms(2000); if (PORTA.5>PORTB.5) {sprintf (lcd_buffer, «На п'яту ніжку порту А
подається більша напруга, ніж на п'яту ніжку порту В»);
delay_ms(2000); lcd_clear(); lcd_gotoxy (0,0);}
if (PORTA.5<PORTB.5) {sprintf (lcd_buffer, «На п'яту ніжку порту В подається більша напруга, ніж на п'яту ніжку порту А»);
delay_ms(2000); lcd_clear(); lcd_gotoxy (0,0);} if (PORTA.5=PORTB.5) {sprintf (lcd_buffer, «На п'яту ніжку порту А
подається така ж напруга, як на п'яту ніжку порту В»);
delay_ms(2000); lcd_clear(); lcd_gotoxy (0,0);}
```

lcd_clear();};}

Або без LCD, чистий код:

if (PORTA.1>PORTB.1) {PORTC=0b11111111}; // на порті C – значення «255» у двійковому коді if (PORTA.1 <PORTB.1) {PORTC = 0b00000111}; // на порті C – сімка if (PORTA.1 = PORTB.1) {PORTC = 0b00000000}; // Вихід (порт C)

дорівнює 0.