МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Чернігівська політехніка»

ПРОЕКТУВАННЯ ЦИФРОВИХ ПРИСТРОЇВ У САПР INTEL QUARTUS PRIME

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторного практикуму та самостійної роботи
для здобувачів першого (бакалаврського) рівня вищої освіти з дисципліни

«Комп'ютерна логіка та основи схемотехніки»

для студентів спеціальності123 – "Комп'ютерна інженерія"

ЗАТВЕРДЖЕНО
на засіданні кафедри
інформаційних та комп'ютерних систем
протокол № 7 від 06.06.24

Чернігів 2024

Проектування цифрових пристроїв у САПР Intel Quartus Prime. Методичні вказівки до виконання лабораторних робіт з для здобувачів першого (бакалаврського) рівня вищої освіти дисципліни “Комп’ютерна логіка та основи схемотехніки” для студентів денної форми навчання напряму підготовки 123 – “Комп'ютерна інженерія”. / Укл. Красножон О.В., Роговенко А.І., Красножон А.В. – Чернігів: НУ «Чернігівська політехніка», 2024 113 стр.

Укладачі: Красножон Олексій Васильович, канд. техн. наук, доцент кафедри інформаційних та комп'ютерних систем;
Роговенко Андрій Іванович, канд. техн. наук, доцент кафедри інформаційних та комп'ютерних систем;
Красножон Андрій Васильович, канд. техн. наук, доцент, доцент кафедри електричної інженерії та інформаційно-вимірювальних технологій

Відповідальний за випуск: В.М. Базилевич, завідувач кафедрою інформаційних та комп'ютерних систем, к.е.н., доцент.

Рецензент: І.В. Білоус, канд. техн. наук, доцент, завідувач кафедри інформаційних технологій та програмної інженерії Національного університету “Чернігівська політехніка”

ЗМІСТ

ВСТУП

Під час розробки звичайних або спеціалізованих цифрових пристроїв вже давно використовують програмовані логічні інтегральні схеми (ПЛІС) або надвеликі інтегральні схеми програмованої логіки (НВІС ПЛ, PLD – Programmable Logic Device). НВІС ПЛ використовуються в різних галузях для створення спеціалізованих пристроїв, вони виявляються поза конкуренцією при створенні високопродуктивних спеціалізованих цифрових пристроїв. Апаратні рішення різноманітних задач забезпечують паралелізацію процесів обробки та збільшення продуктивності в десятки разів у порівнянні, наприклад, із програмними, аналогічну гнучкість модифікації, як і у будь-яких програмних рішень. Розвиток елементної бази НВІС ПЛ дозволив створювати на кристалі стандартні процесорні ядра (так звані Soft Processors) і вирішувати практично будь-які задачі побудови програмно-апаратних систем на одній мікросхемі із використанням єдиних засобів проектування та відлагодження.

Розробник спеціалізованого цифрового пристрою, використовуючи інструментарій систем автоматизованого проектування (САПР, CAD – Computer Aided Design) НВІС ПЛ, у звичному вигляді (структурна або принципова схема, текстовий опис) описує необхідний пристрій та отримує файл, який використовується для конфігурації ПЛІС. Програмування полягає у встановленні необхідних параметрів для функціональних перетворювачів ПЛІС та зв’язків між ними. Такий цикл проектування/виготовлення займає незначний час, при цьому зміни можуть вноситися на будь-якій стадії. Впровадження нових засобів проектування на початковому етапі практично не потребує матеріальних витрат завдяки низькій вартості мікросхем та наявності безкоштовних повнофункціональних версій САПР із відкритим кодом.

Компанія Altera, що зараз є структурним підрозділом у складі Intel, виробляла і виробляє НВІС ПЛ декількох сімейств, що принципово відрізняються: MAX3000A, MAX7000, MAX9000, FLEX10K, FLEX20K, ACEX1K, APEX20K, Stratix I, II, III, Cyclone I, II, III, IV, V. НВІС ПЛ, що входять у ці сімейства відрізняються:

– ступенем інтеграції (логічною ємністю);
– архітектурою функціонального перетворювача (ФП);
– внутрішньою структурою та структурою матриці з’єднань ФП;
– типом програмованого елемента, що використовується;
– наявністю внутрішньої оперативної пам’яті;
– наявністю спеціалізованих вбудованих модулів.

Сучасна елементна база передбачає використання нових технологій проектування та сучасних засобів проектування. Компанія Altera (Intel) пропонує розробникам систему автоматизованого проектування цифрових пристроїв на базі мікросхем програмованої логіки Quartus Prime. Це САПР, що підтримує роботу з усіма новими сімействами НВІС ПЛ, забезпечує доступ до всіх ресурсів мікросхеми та дозволяє проводити проектування цифрових пристроїв та систем будь-якого рівня складності.

1 ОСНОВНІ ВІДОМОСТІ ПРО САПР QUARTUS PRIME

Програмне забезпечення Quartus Prime від компанії Altera являє собою повноцінне кросплатформенне середовище проектування (як для ОС MS Windows, так і для різних версії ОС Linux), яке забезпечує всі стадії проектування цифрового пристрою на ПЛІС за допомогою різних мов опису цифрових пристроїв. Програмне забезпечення Quartus Prime включає засоби для всіх фаз проектування із застосуванням ПЛІС як FPGA (Field Programmable Gate Array), так і CPLD (Complex Programmable Logic Device) структур. Нижче, на рисунку 1.1 представлено узагальнену структуру процесу проектування в середовищі Quartus Prime.

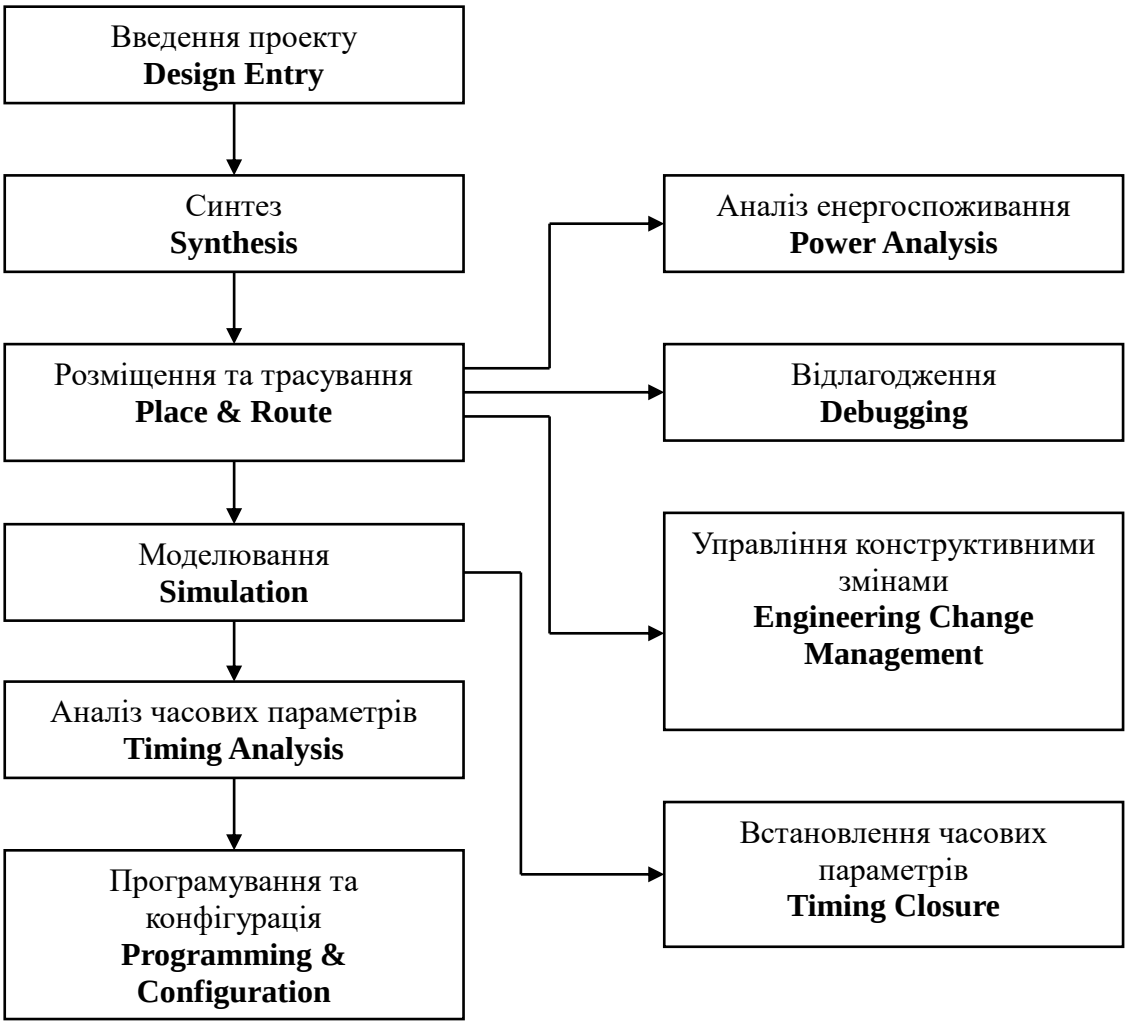


Рисунок 1.1 – Узагальнена структура процесу проектування у середовищі Quartus Prime

Quartus Prime відноситься до САПР ПЛІС (EDA Tool), тобто підтримує весь цикл проектування цифрових пристроїв на основі програмованої логіки, що забезпечує можливість реконфігурації внутрішньої структури (наприклад, для всіх мікросхем ПЛІС всіх сімейств Cyclone).

САПР Quartus Prime містить у своєму складі:

– різні засоби розробки цифрових пристроїв (графічний редактор, редактор схем, редактор мов опису VHDL та Verilog, тощо);

– компілятор та редактор для розміщення розробленої схеми на логіку цільового пристрою;

– засоби аналізу часових характеристик пристрою (час затримки розповсюдження сигналу між входом та виходом, максимальна тактова частота роботи пристрою, тощо);

– редактор часових діаграм для тестування та налагодження пристрою;

– програматор для перенесення конфігурації пристрою з проекту в мікросхему ПЛІС.

Аналогами та конкурентами САПР Quartus Prime є Quarus II, MAX+PLUS II (попередник Quartus II), ISE компанії Xilinx та Active HDL компанії Aldec.

Додатково Quartus Prime надає можливість використовувати графічний інтерфейс користувача, EDA або інтерфейс командного рядка в кожній фазі розробки. Ви можете використовувати будь-який інтерфейс для всього процесу проектування або різні налаштування для кожної окремої фази розробки.

1.1 Можливості САПР Quartus Prime

До основних можливостей САПР Quartus Prime можна віднести:

– різні способи введення поведінкових та структурних описів пристроїв;

– інтегровані засоби допомоги для створення складних проектів MegaWizard® & SOPC Builder;

– підсистему синтезу;

– підсистему розміщення внутрішніх ресурсів та трасування з’єднань всередині ПЛІС;

– підсистему моделювання;

– підсистему часового аналізу та аналізу споживання енергії;

– підсистему програмування НВІС;

– підсистему оптимізації швидкодії проекту – LogicLockTM;

– підсистему підтримки інтеграції з іншими засобами автоматизації проектування – NativeLink®;

– систему проектування блоків цифрової обробки сигналів – DSP Builder;

– інтегровані засоби розробки програмного забезпечення для вбудованих систем;

– підтримка використання IP-модулів;

– вбудовані засоби відлагодження ПЛІС у складі системи Signal Tap Logic Analyzer® II & SignalProbeTM;

– підтримку операційних систем Widows, Solaris та Linux;

– підтримку різних схем ліцензування (nodelocked, network, gpl, lgpl).

1.2 Особливості САПР Quartus Prime

САПР Quartus Prime – це наступний крок у проектуванні пристроїв із високим ступенем інтеграції, включаючи розробку замкнених систем на одному

кристалі (SOPC – System-on-a-Programmable-Chip).

Quartus Prime поєднує в собі проектування, синтез, розміщення елементів, трасування з'єднань та верифікацію, зв'язок із системами проектування інших виробників.

Розробка систем на кристалі вимагає від розробників ефективної командної роботи: внесення змін в одній із частин проекту повинно створювати мінімальний вплив на інших членів команди. Програмне забезпечення Quartus Prime – це комплексне середовище для розробки систем на кристалі SOPC, доступне в даний час. Нові особливості СПАР Quartus Prime:

1. **LogicLock** – це блокова методологія проектування. Quartus Prime спільно із LogicLock – єдине програмне забезпечення для розробки пристроїв на основі програмованої логіки, яке включає блочну методологію проектування як стандартну функцію, що допомагає збільшити ефективність роботи розробників, зменшити час на проектування та верифікацію. LogicLock дозволяє проектувати та перевіряти кожен модуль окремо. Розробники можуть інтегрувати готові модулі у проект верхнього рівня, зберігаючи продуктивність кожного модуля у процесі об'єднання. LogikLock знижує час розробки та верифікації, оскільки кожен модуль оптимізується лише один раз.

2. **NativeLink** – забезпечує зв'язок між інструментами розробки Quartus Prime та програмним забезпеченням інших виробників. NativeLink дозволяє засобам синтезу сторонніх виробників перетворювати свої примітиви безпосередньо на примітиви пристроїв компанії Altera (Intel). Пряме перетворення скорочує час компіляції та звільняє від використання додаткових бібліотек трансляції перетворень, які можуть обмежити продуктивність, досягнуту засобами проектування сторонніх виробників. Застосування NativeLink в процесі розробки дозволяє розробникам використовувати Quartus Prime для розміщення елементів, а засоби проектування інших виробників – для оптимізації стратегій синтезу.

3. **Технологія PowerFit розміщення елементів та трасування з'єднань** у Quartus Prime використовує часові параметри, задані розробником, для оптимального складання схеми та розміщення логічних елементів. Інтелектуальний алгоритм трасування за часовими параметрами приділяє першочергову увагу з'єднанням, критичним до часових параметрів. Такі з'єднання оптимізуються в першу чергу для зменшення затримок і досягнення максимальної продуктивності (fMAX). Подальше покращення параметра fMAX може досягатися шляхом використання більш новітні архітектур мікросхем ПЛІС.

4. **Верифікація** або перевірка проекту може стати найтривалішою стадією в процесі розробки високопродуктивних систем на кристалі (SOPC). Однак, використовуючи Quartus Prime, можна скоротити цей час, оскільки програмне забезпечення має набір власних засобів верифікації, інтегрованих з останніми засобами верифікації сторонніх виробників.

5. **Аналіз**. Компанія Altera розробила два методи, щоб допомогти розробникам проаналізувати стан внутрішніх точок та входів/виходів пристрою. Це налагоджувальний засіб SignalProbe та логічний аналізатор

SignalTap. Технології SignalTap та SignalProbe можуть працювати спільно із засобами синтезу сторонніх виробників та не вимагають внесення змін до вихідного HDL файлу проекту.

Доступна технологія апаратного налагодження SignalProbe дозволяє користувачам послідовно з’єднувати внутрішні точки пристрою із вільними зарезервованими виводами для аналізу за допомогою осцилографа або логічного аналізатора.

Для багатьох розробників, які використовують мікросхеми в корпусах BGA (Ball Grid Array) із великою кількістю вхідних/вихідних контактів, верифікація системного рівня займає багато часу і є суттєво ускладненою. Логічний аналізатор SignalTap здійснює верифікацію за допомогою інтеграції функціональності логічного аналізатора в програмному забезпеченні. SignalTap дозволяє розробникам зібрати дані з будь-яких внутрішніх точок та входів/виходів пристрою в режимі реального часу під час роботи системи. Quartus Prime вставляє у проект мегафункцію, що містить логічний аналізатор. Дані збираються та зберігаються у блоках вбудованої пам'яті пристрою та направляються у програмне забезпечення Quartus Prime через завантажувальний кабель. Розробники також можуть і подавати внутрішні сигнали на виводи пристрою з метою подальшого моніторингу. Логічний аналізатор SignalTap дозволяє суттєво знизити час верифікації, що дозволяє у більш короткий термін випускати нові пристрої.

6. **Технологія PowerGauge** являє собою інтегрований засіб аналізу енергоспоживання. Вона використовує файли, створені в процесі моделювання, щоб зв'язати оцінку споживання енергії із заданими параметрами пристрою. Використовуючи симулятор Quartus Prime або симулятори інших виробників, інтегрований аналізатор енергоспоживання дозволяє розробникам виміряти та оптимізувати профіль споживання енергії на більш ранній стадії процесу розробки.

2 ПОНЯТТЯ ПРОЕКТУ В САПР QUARTUS PRIME

Під терміном “проект” у САПР Quartus Prime мається на увазі набір файлів, пов’язаних із проектованим цифровим пристроєм/системою, в якому виділяються дві групи файлів:

– логічні, що описують структуру або алгоритм роботи (Design Files);

– допоміжні (Ancilary Files).

Проект може містити як один логічний файл, так і їх сукупність, що утворюють ієрархічний опис модуля/пристрою/системи, що проектується. При ієрархічному описі серед множини логічних файлів виділяють:

– файл верхнього рівня в ієрархії опису (Top-level Design File);

– файли нижчих (одного чи кількох) рівнів (Low-level Design files).

У файлі верхнього рівня ієрархії задається архітектура модуля, визначається набір модулів, що входять до його складу як компоненти, та їх взаємозв’язки. Описи цих модулів містяться у логічних файлах нижчого рівня ієрархії. В свою чергу, до їх складу у вигляді компонентів можуть входити модулі, описи яких наведено в логічних файлах ще більш низького рівня ієрархії, і т.д.

Ім’я проекту має збігатися з іменем модуля верхнього рівня в ієрархії описів, а отже, і з іменем логічного файлу, в якому міститься цей опис. Імена модулів нижніх рівнів ієрархії, в свою чергу, повинні збігатися з іменами файлів, в яких вони реалізовані.

До логічних можуть відноситися файли, що мають один із наведених нижче типів:

1. **Block Diagram / Schematic File** (*.bdf), містить структурну схему пристрою, створену в САПР Quartus Prime.
2. **Graphic Design File** (*.gdf), містить принципову електричну схему, створену в САПР.
3. **AHDL Text Design File** (*.tdf), містить текстовий опис модуля мовою AlteraHDL.
4. **Waveform Design File** (*.wdf), містить часові діаграми вхідних та вихідних сигналів, створених в Quartus Prime.
5. **VHDL Design File** (*.vhd), містить текстовий опис модуля мовою VHDL.
6. **Verilog Design File** (*.v), містить текстовий опис модуля мовою Verilog HDL.
7. **Orcad Schematic Files** (*.sch), містить схему, створену в САПР ORCAD.
8. **EDIF Input Files** (*.edf), містить опис EDIF 200 або 300.
9. **Xilinx Netlist Format File** (*.xnf), містить опис модуля, отриманий в САПР фірми Xilinx.

Додаткові файли зберігають додаткову інформацію про проект. Їх назви збігаються із назвою проекту, а розширення можуть бути різними.

3 СТРАТЕГІЯ ПРОЕКТУВАННЯ

Середовище проектування Quartus Prime дозволяє реалізовувати як стратегії висхідного, так і низхідного структурного проектування.

Як перша, так і друга стратегії включають в себе використання поведінкових та структурних описів модулів/пристроїв. При структурному описі модуль/пристрій представляється у вигляді сукупності взаємозалежних компонентів більш низького рівня ієрархії описів. При поведінковому описі задається алгоритм їх роботи.

Висхідне проектування застосовується в тому випадку, коли для пристрою, що розробляється, вже існує детальний структурний опис (зазвичай, це схема електрична принципова, яку реалізовано на папері і яка складається із мікросхем середнього ступеня інтеграції), виконаний в елементному базисі, відмінному від базису, що є в розпорядженні розробника ПЛІС.

В такому випадку розробник вирішує наступні завдання:

– створення функціональних аналогів елементів, використаних в існуючому структурному описі;

– налаштування створених компонентів;

– інтеграцію створених компонентів у єдиний модуль;

– моделювання та відлагодження роботи пристрою в цілому.

Таким чином, у процесі проектування розробник спочатку створює модулі нижнього рівня в ієрархії описів, а потім модуль верхнього рівня. Звідси й назва стратегії проектування.

Стратегія низхідного проектування застосовується у випадку, коли задано алгоритм роботи (у вигляді опису поведінки) пристрою, що розробляється, і набір системних вимог (максимальна тактова частота роботи, затримка на розповсюдження сигналів від входів до виходів, споживання енергії, вартість та ін.). При цьому поведінковий опис може бути як формалізованим (тобто, представлятися у вигляді схеми алгоритму, графа, таблиці переходів, таблиці виходів, тощо), так і неформалізованим (текстовий опис). Реалізація низхідного проектування базується на ітераційному виконанні структурної декомпозиції.

У спрощеному вигляді (орієнтуючись на функціональні можливості САПР Quartus Prime), процедура низхідного проектування виглядає наступним чином:

– розробка архітектури НВІС: поведінковий опис перетворюється на структурний (структурну схему), елементами якої є архітектурні модулі;

– архітектурні модулі або описуються на поведінковому рівні (наприклад, за допомогою мов AHDL, VHDL, Verilog), або здійснюється їх структурна декомпозиція та створюється структурний опис, елементами якого є функціональні модулі (далі описана процедура циклічно повторюється до тих пір, поки всі функціональні модулі не буде описано на поведінковому рівні);

– після цього здійснюється функціональне моделювання роботи модулів, що мають поведінкові описи;

– функціональне моделювання роботи модулів, що мають структурний опис (оскільки модулі, що мають поведінковий опис, входять до них як

компоненти);

– моделювання та відлагодження роботи пристрою в цілому.

Таким чином, в процесі проектування розробник опускається від верхнього рівня ієрархії описів, тобто, рівня HBIC, до нижніх рівнів. Звідси й назва стратегії проектування.

Варто відмітити, що стратегія низхідного проектування має безумовні переваги як за витратами часу на розробку, так і за якістю опрацювання проекту.

Незалежно від вибору стратегії проектування, для реалізації структур та алгоритмів роботи модулів/пристроїв доцільно використовувати текстовий опис, створений мовами опису апаратури високого рівня (AHDL, VHDL, Verilog).

Нижче розглядаються основні етапи проектування та особливості налаштувань САПР Quartus Prime у кожній конкретній частині процесу розробки, а також загальна методологія проектування.

Спочатку створюється загальна структурна схема верхнього рівня ієрархії, далі виконується опис окремих модулів пристрою, після чого файли проекту об’єднуються, тобто реалізується стратегія низхідного проектування.

4 ВВЕДЕННЯ ОПИСУ ПРОЕКТУ

Як зазначалося раніше, проект у САПР Quartus Prime являє собою повний набір, який складається із наступних файлів: призначення, моделювання, системних налаштувань та інформації про ієрархічну структуру проекту.

4.1 Запуск САПР Quartus Prime

Запуск Quartus Prime виконується або за допомогою іконки, розташованої на робочому столі ОС комп'ютера, або за допомогою меню Пуск (Пуск => Всі програми => Altera => Quartus Prime), або із командного рядка (консолі), після виклику якої треба ввести Quartus і натиснути клавішу Enter.

Після запуску на моніторі з’явиться головне (стартове) вікно САПР, вигляд якого зображено нижче, на рисунку 4.1, воно складається із наступних областей:

– заголовок вікна, що містить назву проекту та його робочий каталог (позначено цифрою I);

– головне меню САПР (позначено цифрою II);

– панель інструментів САПР (позначено цифрою III);

– навігатор проекту, що відображає ієрархію проекту, файли проекту та команди швидкого запуску (позначено цифрою IV);

– вікно стану процедури компіляції проекту (позначено цифрою V);

– вікно повідомлень (або консоль), що виводить інформацію про виконання операцій, помилки та попередження (позначено цифрою VI);

– головне вікно, що відображає звіти, файли проекту та інші елементи проекту (позначено цифрою VII).

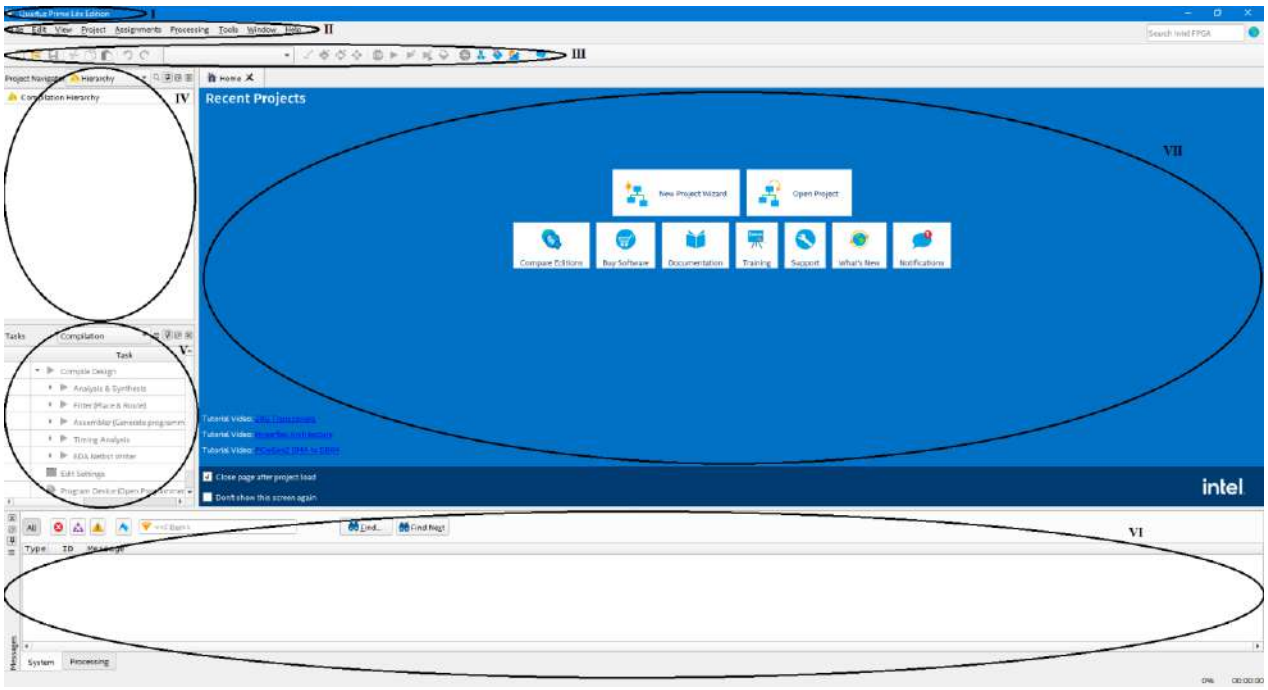


Рисунок 4.1 – Загальний вигляд головного вікна при першому запуску середовища Quartus Prime

Варто зазначити, що вікно, зображене на рисунку 4.1 буде з’являтися при кожному запуску САПР Quartus Prime. Як видно із рисунку 4.1, області IV (Project Navigator) і VI (Messages) є порожніми, оскільки жодного проекту зараз не відкрито для роботи, крім того, область V (Tasks) є недоступною (виділена сірим кольором), оскільки немає задач для компіляції. Список Recent Projects в області VII вікна може містити посилання на проекти, які раніше відкривалися (не більше чотирьох посилань). Натискання на відповідне посилання дозволить швидше відкрити необхідний проект.

З метою порівняння і пошуку відмінностей нижче, на рисунку 4.2 зображено приклад головного вікна САПР Quartus Prime із відкритим проектом.

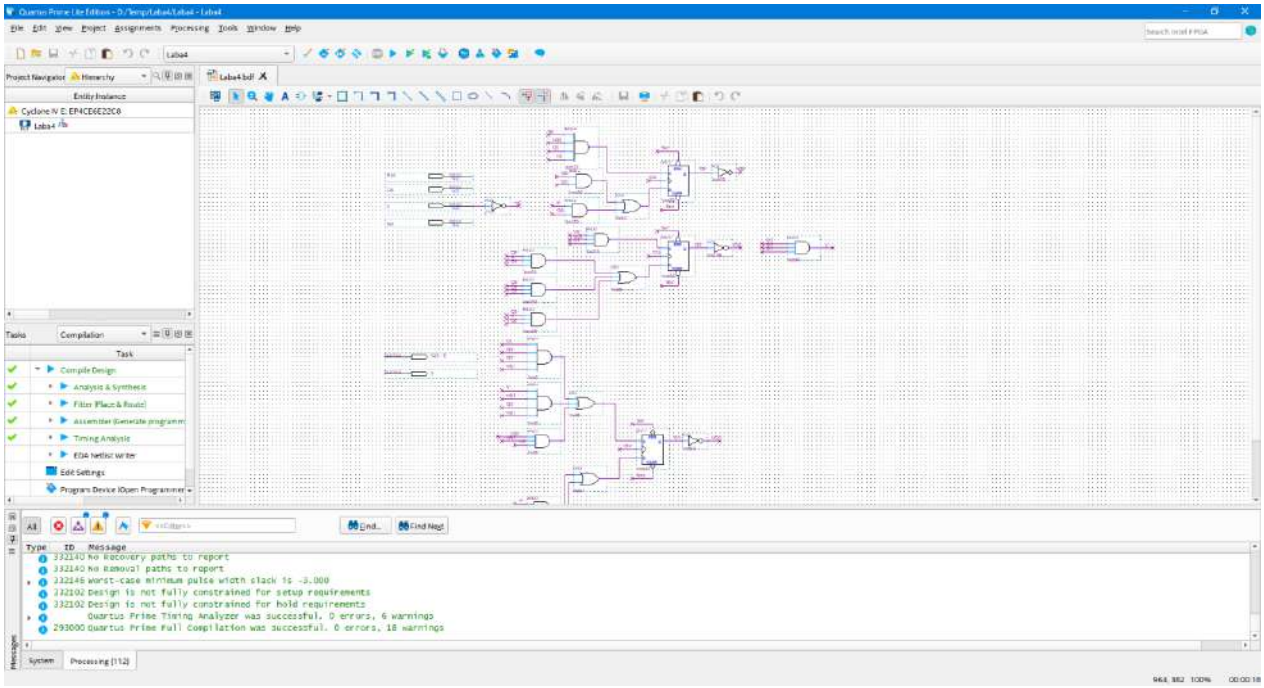


Рисунок 4.2 – Загальний вигляд головного вікна при роботі над проектом в середовищі Quartus Prime

4.2 Визначення нзави проекту та його основних параметрів

Існує два шляхи для створення нового проекту в САПР Quartus Prime:

– створення необхідних файлів проекту окремо з їх подальшим додаванням і зв’язуванням вручну;

– використання вбудованого інструменту – майстра створення проекту (New Project Wizard).

Очевидно, перший шлях є досить складним і довгим, тому, розглянемо детальніше другий шлях. Запустити New Project Wizard можна двома способами:

– натиснути на кнопку з аналогічною назвою, що розташовується в області VII головного вікна САПР (дивись рисунок 4.1);

– активувати пункт із аналогічною назвою, що знаходиться в меню File (область 2 на рисунку 4.1).

Створення проекту за допомогою майстра New Project являє собою виконання зазначеної послідовності дій:

1. В робочій області жорсткого диску створити каталог, в якому буде зберігатися проект. Увага! Quartus Prime самостійно не створює каталог для кожного окремого проекту! крім того, для всіх нових проектів буде пропонуватися однакове місце їх розміщення, що через певний час призведе до плутанини і проблем при копіюванні, переносі або видаленні проектів! Наполегливо рекомендується давати назву для каталогу проекту тільки латинськими (англійськими) літерами, використання кирилічних символів (українських або російських) може призводити до проблем при маніпуляціях із проектом. Рекомендується не робити великий рівень вкладеності (глибину) при розміщенні каталогу проекту на жорсткому диску, і в шляху до цього каталогу краще також уникати не латинських символів.

2. При першому запуску майстра створення проекту відображається вікно представлення, в якому зазначено всю необхідну послідовність дій, що супроводжує створення і відкриття нового проекту (рисунок 4.3 нижче). Після ознайомлення із нею можна перейти безпосередньо до створення самого проекту. Щоб перейти до першого вікна створення нового проекту треба натиснути клавішу Next.

3. У вікні першого етапу створення проекту (рисунок 4.4) необхідно вказати назву і шлях до робочого каталогу проекту (той, що було створено на першому кроці), назву проекту та назву файлу верхнього рівня ієрархії (таблиця 4.1). Для переходу до наступного етапу – натиснути Next.

Таблиця 4.1 – Призначення полів уведення

Найменування поля вікна	**Призначення**
Робочий каталог проекту (**working directory for this project**)	Каталог повинен містити всі файли проекту та інші пов'язані з цим проектом файли. Для вибору існуючого каталогу, необхідно натиснути кнопку «…», розташовану праворуч поля введення
Назва проекту (**name of this project**)	Задається назва файлу верхнього рівня проекту. Назву тут краще вписати вручну, причому, вона повинна співпадати із назвою робочого каталогу проекту
Назва блоку верхнього рівня ієрархії проекту (**name of the top-level design entity for this project**)	Quartus Prime автоматично створює налаштування компіляції та моделювання для блоку, зазначеного у цьому вікні. Після створення проекту Ви можете додати інші об’єкти верхнього рівня ієрархії та створити для них налаштування компіляції та моделювання за допомогою меню Processing. Вказана назва проекту автоматично надається блоку верхнього рівня. Однак допустимо задавати назви, відмінні від назви блоку верхнього рівня ієрархії, хоча це не рекомендується робити

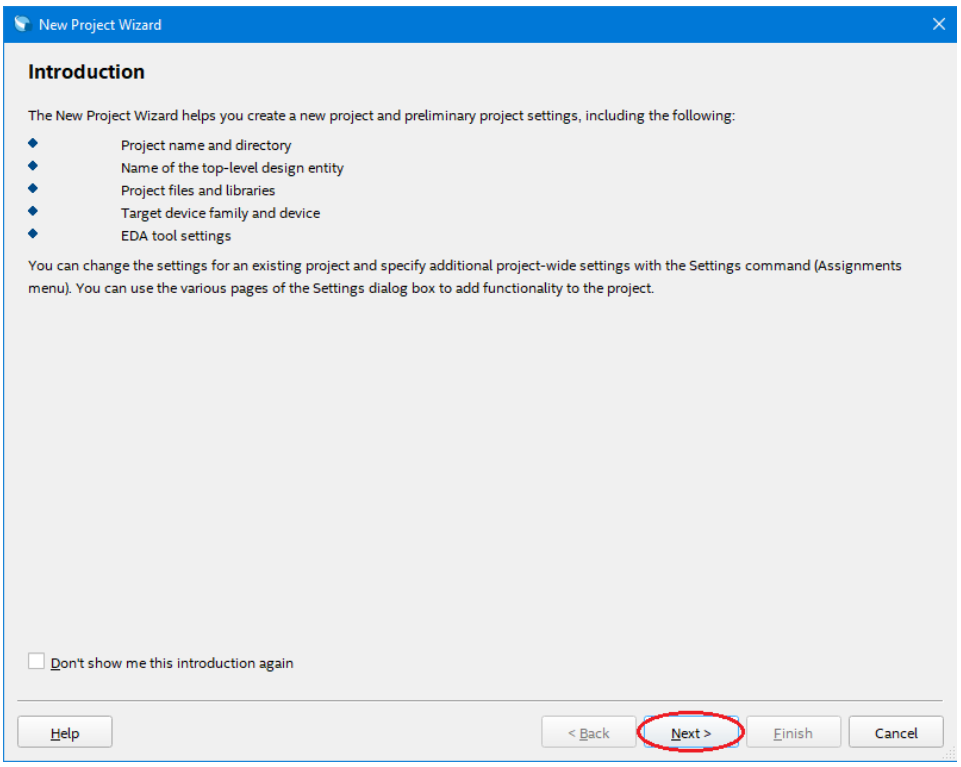


Рисунок 4.3 – Загальний вигляд вікна представлення (Introduction) при запуску майстра створення проекту New Project Wizard

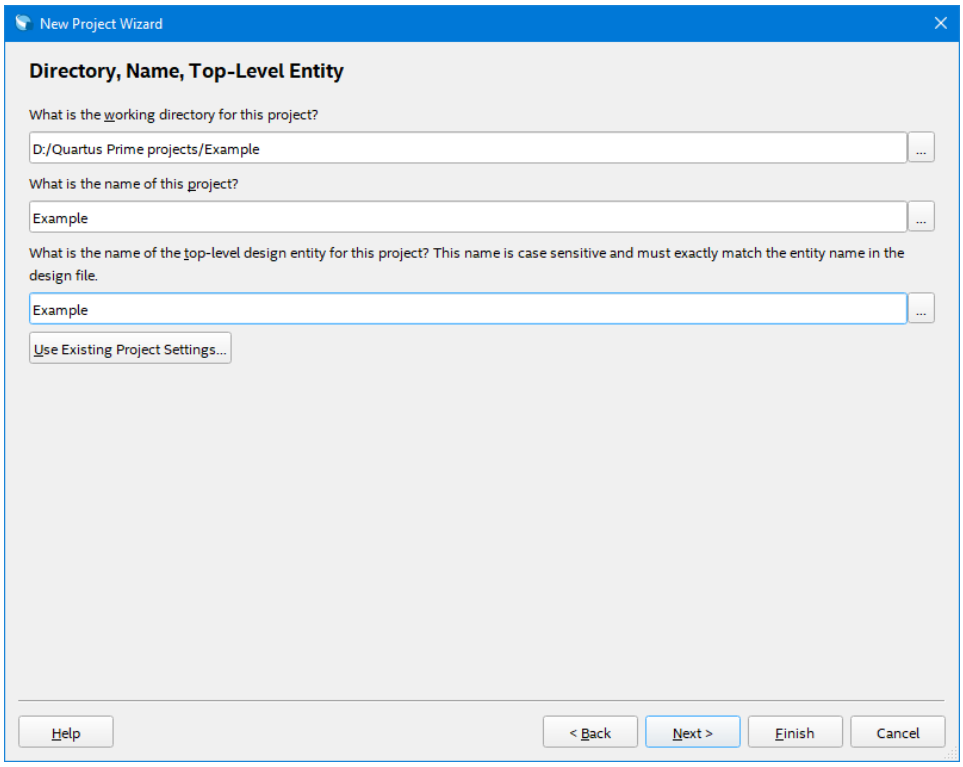


Рисунок 4.4 – Загальний вигляд вікна першого етапу створення нового проекту за допомогою New Project Wizard

4. У вікні другого етапу створення проекту (рисунок 4.5) необхідно обрати тип створюваного проекту. Призначення полів вікна описано у таблиці 4.2. При створенні перших проектів, у даному вікні рекомендується обирати Empty Project, оскільки використання шаблонів вимагає наявності певних

знань, вмінь і навичок роботи із САПР. Для переходу до наступного етапу – натиснути кнопку Next, для повернення на перший етап – кнопку Back.

Таблиця 4.2 – Призначення полів введення

Найменування поля вікна	**Призначення**
Порожній проект (**Empty project**)	Створити новий проект шляхом визначення файлів і бібліотек, конкретного цільового пристрою та його сімейства, а також налаштувань інструментарію розробника ПЛІС (EDA tools)
Шаблонний проект (**Project Template**)	Створити проект на основі існуючого шаблону. Ви можете самостійно обирати серед шаблонів, які було встановлено разом із поточною версією САПР Quartus Prime, або завантажити їх за відповідним посиланням

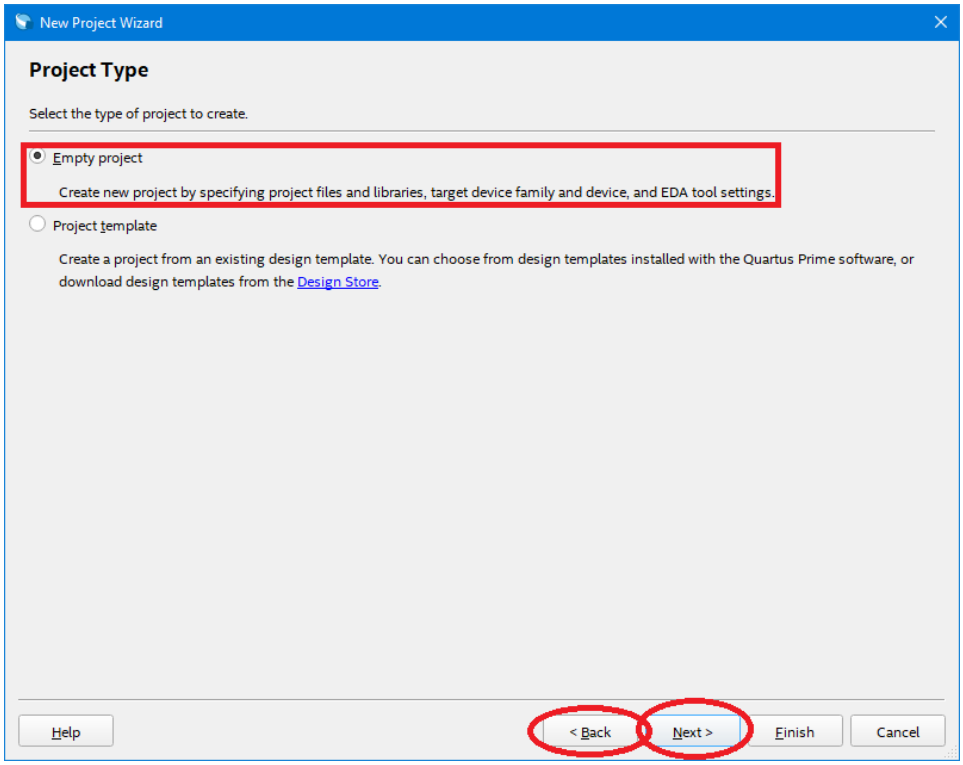


Рисунок 4.5 – Загальний вигляд вікна другого етапу створення нового проекту за допомогою New Project Wizard

5. Вікно третього етапу створення проекту (рисунок 4.6) призначено для підключення до проекту раніше створених файлів і бібліотек (якщо такі існують). Призначення полів вікна описано у таблиці 4.3. При створенні перших проектів, дане вікно рекомендується просто пропускати. Для переходу до наступного етапу – натиснути кнопку Next, для повернення на другий етап – кнопку Back.

Таблиця 4.3 – Призначення полів введення

Найменування поля вікна	**Призначення**
Назва файлу (**File name**)	Виберіть імена файлів, які потрібно підключити до проекту. Натисніть кнопку **Add All**, щоб додати до робочого каталогу проекту всі файли. Якщо Ви не маєте файлів проекту в інших каталогах або файлах, назва яких відрізняється від назви вашого проекту, то додавати файли в цьому вікні не обов'язково. Шляхи до файлів, що додатково підключаються, та їх імена можуть бути знайдені за допомогою кнопки (…)
Вибір додаткових бібліотек (**User libraries...**)	Якщо проект включає бібліотеки спеціалізованих функцій, вкажіть шлях до них

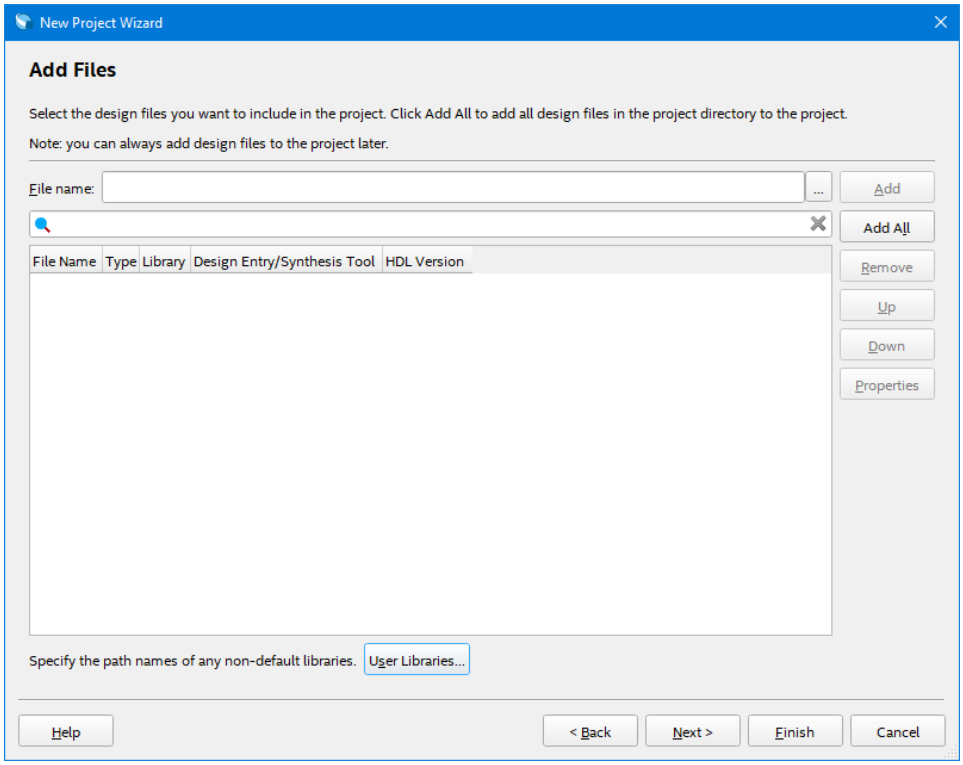


Рисунок 4.6 – Загальний вигляд вікна третього етапу створення нового проекту за допомогою New Project Wizard

6. Вікно четвертого етапу створення проекту (рисунок 4.7) призначено для вибору сімейства та типу мікросхеми ПЛІС, яка буде використовуватися при компіляції проекту. В принципі, на цьому кроці можна вибрати будь-яку мікросхему серед будь-яких доступних сімейств (або просто доручити вибір на розсуд трасувальника встановивши налаштування Auto device selected by the fitter), але слід пам’ятати, що від цього вибору безпосередньо залежить час

компіляції, її складність і результат. Оскільки, в навчально-відлагоджувальному стенді використовується мікросхема EP4CE6E22C8 сімейства Cyclone IV E, потрібно вибирати саме її. Для переходу до наступного етапу – натиснути кнопку Next, для повернення на третій етап – кнопку Back.

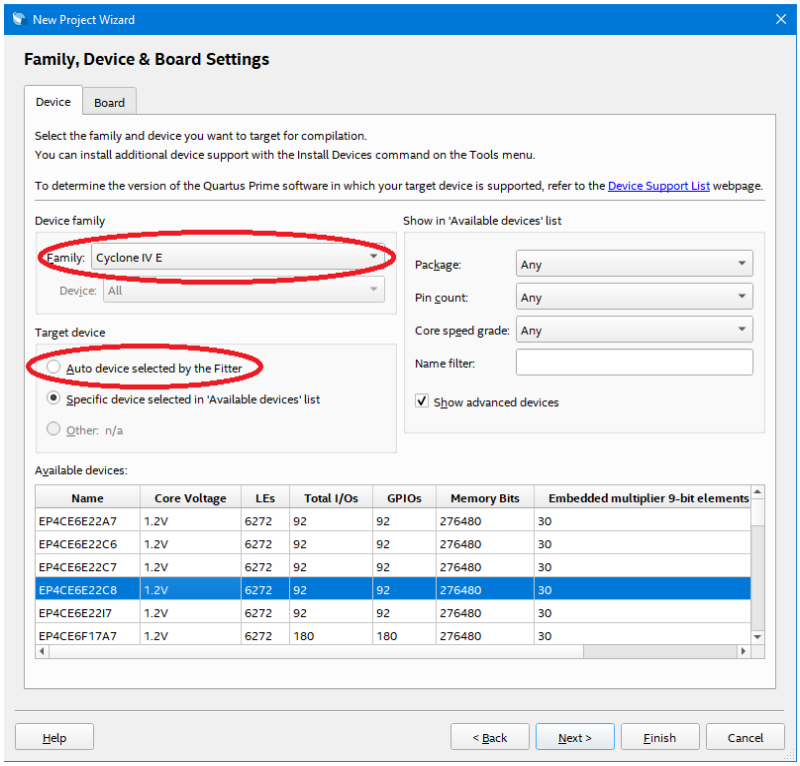


Рисунок 4.7 – Загальний вигляд вікна четвертого етапу створення нового проекту за допомогою New Project Wizard

7. Вікно п’ятого етапу створення проекту (рисунок 4.8) надає можливість підключення до САПР Quartus Prime під час створення цього проекту інших сторонніх засобів (EDA – Electronic Design Automation). Якщо розробка проекту не потребує додаткових інструментів EDA, не потрібно їх вибирати. При створенні перших проектів, дане вікно рекомендується просто пропускати. Для переходу до наступного етапу – натиснути кнопку Next, для повернення на четвертий етап – кнопку Back.

8. Вікно заключного шостого етапу (рисунок 4.9) містить повну підсумкову інформацію стосовно всіх встановлених раніше налаштувань. При необхідності внесення змін завжди можна повернутися до попередніх вікон натисканням клавіші Back необхідної кількості разів. Для завершення процесу створення нового проекту необхідно натиснути кнопку Finish.

Після натискання кнопки Finish і закриття вікна Summary процес створення нового проекту закінчиться. Результат створення нового проекту можна побачити в області IV (Project Navigator) головного вікна САПР (рисунок 4.1). Там з’явиться: назва сімейства мікросхеми ПЛІС, яка буде використовуватися, її точне найменування, і назва блоку верхнього рівня ієрархії проекту. Одночасно в області I (рисунок 4.1) головного вікна також з’явиться шлях до каталогу проекту, його назва та найменування блоку

верхнього рівня ієрархії.

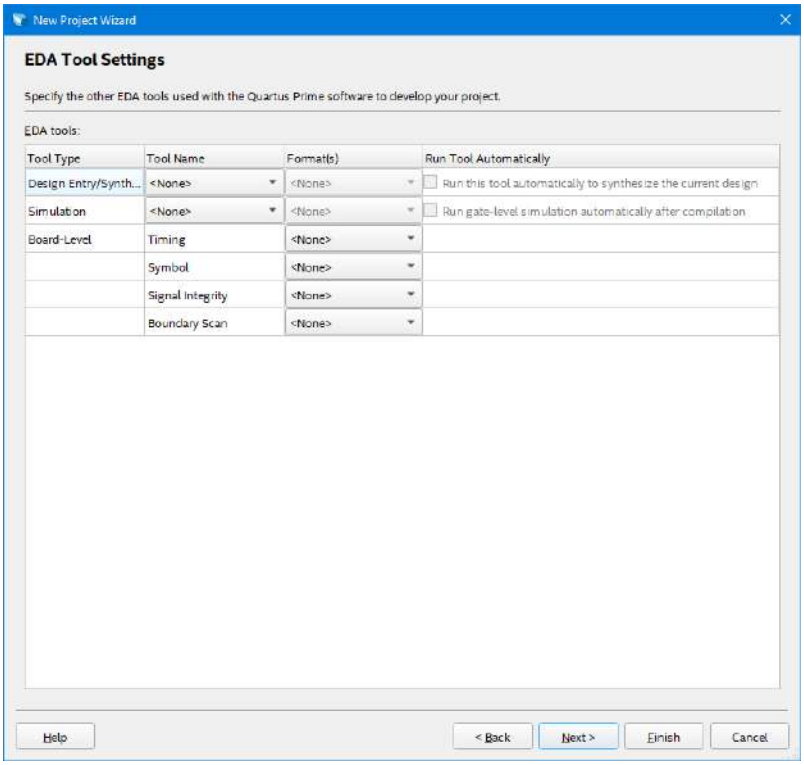


Рисунок 4.8 – Загальний вигляд вікна п’ятого етапу створення нового проекту за допомогою New Project Wizard

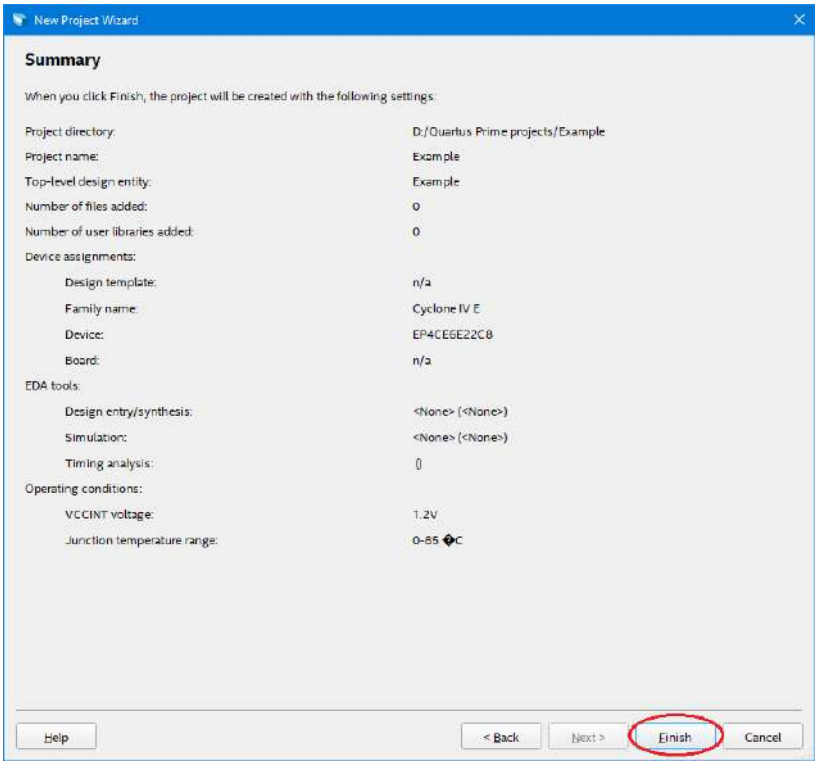


Рисунок 4.9 – Загальний вигляд вікна заключного етапу (Summary) створення нового проекту за допомогою New Project Wizard

4.3. Створення структурної схеми проекту

Створення структурної схеми проекту починається зі створення відповідного файлу схеми.

4.3.1 Створення файлу структурної схеми проекту

Для створення файлу структурної схеми проекту необхідно виконати наступну послідовність дій:

1. В області II (рисунок 4.1) натиснути на меню File (операції з файлами), де із випадного списку вибрати пункт New (новий файл). Аналогічний результат можна отримати після натискання комбінації клавіш Ctrl+N. В результаті повинно з'явитися вікно New (рисунок 4.10), яке дозволяє вибрати необхідний тип файлу.

2. У цьому вікні необхідно знайти та натиснути на пункт Design Files (файли розробника) і у списку, що відкриється, вибрати рядок Block Diagram/Schematic File. Після натискання кнопки OK, відкриється вікно редактора структурної схеми (рисунок 4.11).

3. Після цього в меню File вибрати пункт Save As і ввести назву файлу, що створюється. За потреби, встановити мітку біля пункту Add file to current project (додати файл до поточного проекту) як це показано на рисунку 4.12, і натиснути на кнопку "Зберегти". У головному вікні САПР з'явиться область із заданою назвою файлу, а у каталозі проекту, що використовується, з'явиться файл із розширенням *.bdf. Слід зазначити, що САПР автоматично використовує назву проекту як назву файлу верхнього рівня ієрархії проекту.

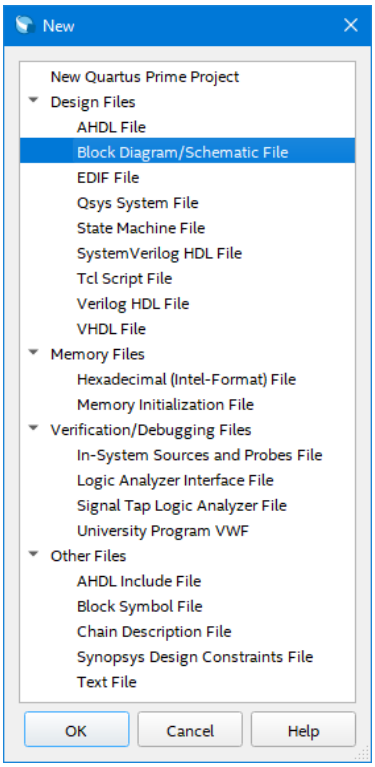


Рисунок 4.10 – Загальний вигляд вікна вибору файлів для створеного проекту

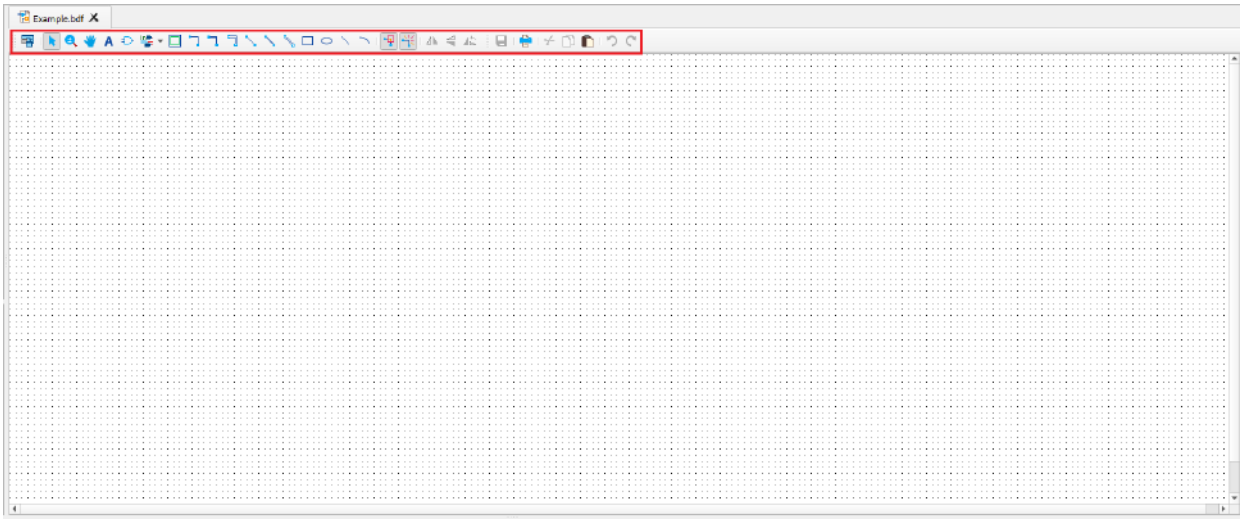


Рисунок 4.11 – Загальний вигляд вікна редактора структурних схем САПР Quartus Prime

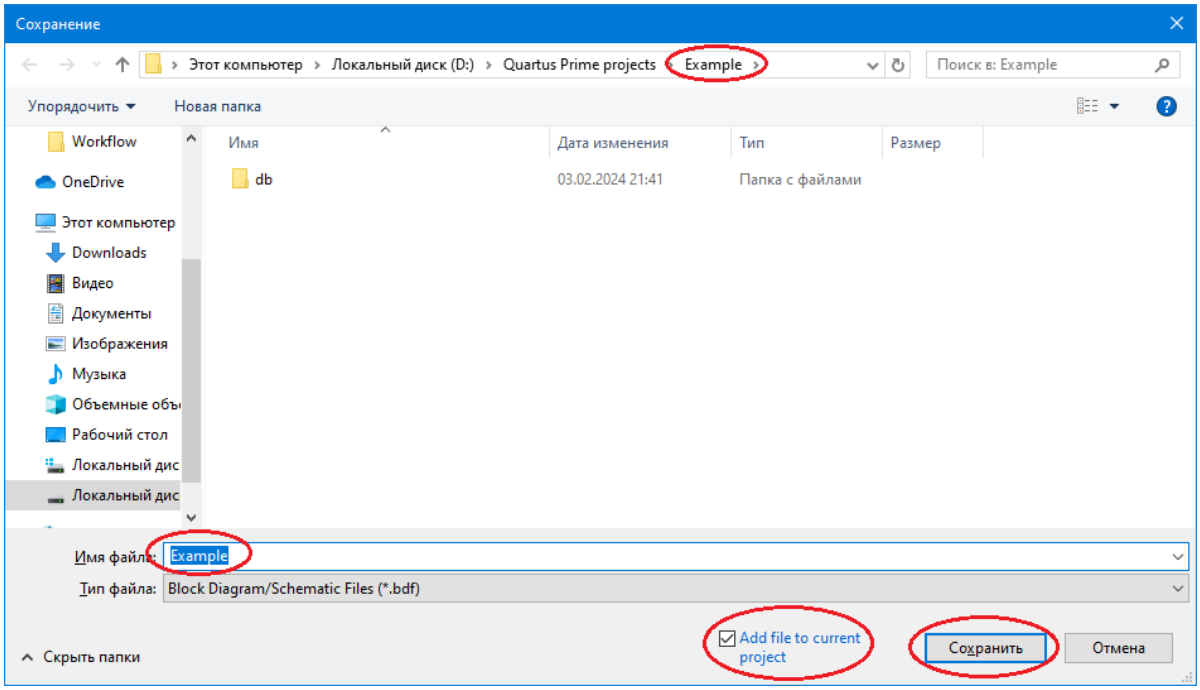


Рисунок 4.12 – Загальний вигляд вікна вибору файлів для створеного проекту

4.3.2 Введення структурної схеми проекту

Створення структурної схеми проекту здійснюється за допомогою панелі інструментів (рисунок 4.11) вікна редактора структурних схем САПР Quartus Prime. Призначення та опис цих інструментів наведено в таблиці 4.4.

Таблиця 4.4 – Опис інструментів, що знаходяться на панелі вікна редактора структурних схем САПР Quartus Prime

Режим	Призначення
Редагування та опитування	
 Від’єднати вікно **(Detach Window)**	Від’єднання вікна робочої області від САПР, при повторному натисканні – приєднання назад
 Вказівник виділення **(Selection Tool)**	Інструмент вибору об’єктів для виконання операцій. Окремий об’єкт вибирається натисканням миші. Щоб додати об’єкт до групи – натиснути Shift
 Введення тексту **(Text Tool)**	Інструмент для введення тексту. Нанесення на схему текстових написів: назв сигнальних кіл, опису моделей компонентів, коментарів, тощо
 Введення компонента **(Symbol Tool)**	Інструмент для введення умовного графічного позначення функціонального елемента (подвійний клік у полі схеми виконає аналогічну дію)
 Введення блоку **(Block Tool)**	Інструмент для введення блоку. Дозволяє створити деяку сукупність функціональних елементів і об’єднати їх
 Ортогональний провідник **(Orthogonal Node Tool)**	Інструмент для створення ортогональних провідників (сигнальних кіл)
 Ортогональна шина **(Orthogonal Bus Tool)**	Інструмент для створення ортогональних шин (сукупності провідників однакового призначення)
 Ортогональний канал **(Orthogonal Conduit Tool)**	Інструмент для створення ортогональних каналів зв’язку (передачі даних)
 “Гумові” лінії зв’язку **(Use Rubberbanding)**	Використання “гумових” ліній зв’язку. При переносі компонентів схеми автоматично тягне за ними ортогональні провідники, канали та шини
 Частковий вибір ліній **(Use Partial Line Selection)**	Використання часткового вибору ліній. Вибір необхідної ділянки зв’язку незалежно від її розміру (а також кількох ліній зв’язку одночасно)
 Масштабування схеми **(Zoom Tool)**	Вибір масштабу для відображення об’єктів структурної схеми

Продовження таблиці 4.4

Режим	Призначення
Редагування та опитування	
 Віддзеркалення по горизонталі **(Flip Horizontal)**	Віддзеркалення виділеного об’єкта в горизонтальній площині
 Відображення по вертикалі **(Flip Vertical)**	Віддзеркалення виділеного об’єкта у вертикальній площині
 Поворот проти годинникової стрілки на 90° **(Rotate Left 90)**	Поворот виділеного об’єкта схеми на 90 градусів проти годинникової стрілки (вліво)
 Еліпс **(Oval Tool)**	Інструмент для створення еліптичних форм
 Лінія **(Line Tool)**	Інструмент для створення ліній.
 Дуга **(Arc Tool)**	Інструмент для створення дуг.

4.3.3 Налаштування та зміна властивостей екрану

У режимі редактора структурних схем можна змінити основні налаштування головного вікна САПР Quartus Prime, для цього необхідно в області II (рисунок 4.1) натиснути меню Tools, потім – вибрати команду Options... У вікні, що з’явилося (рисунок 4.13) у списку Category необхідно вибрати пункт Block/Symbol Editor. Вміст саме цього вікна дозволяє редагувати властивості головного вікна САПР, опис яких наведено у таблиці 4.5.

Таблиця 4.5 – Властивості екрану

Властивість	Призначення
Show guidelines	Показувати лінії сітки
Guidelines spacing	Крок ліній сітки
Snap to grid (applies only to Symbol Editor)	Прив’язка компонентів і ліній до сітки (тільки у режимі редактора символів)
Use rubberbanding	Використання “гумових” ліній зв’язку
Use partial line selection	Використання часткового вибору ліній зв’язку
Double click to show property sheet	Дозволити виклик вікна властивостей компонента подвійним кліком миші

Продовження таблиці 4.5

Властивість	**Призначення**
Show parameter assignment	Показувати значення параметрів
Show block I/0 tables	Показувати таблиці входів/виходів для блоків
Show mapper table	Показувати відображення таблиць
Show location assignments	Показувати розташування виводів і компонентів у ПЛІС
Show I/O standard and reserve pin assignments	Показувати призначення стандартних і зарезервованих ліній вводу/виводу ПЛІС
Show connection rectangle	Показувати прямокутники у місцях з'єднання провідників і компонентів
Show page breaks	Показувати розриви сторінок схеми
Include a border when printing	Додавати рамки сторінок під час друку
Automatically open as detached window	Автоматично відкривати як окреме вікно (від'єднане від САПР)

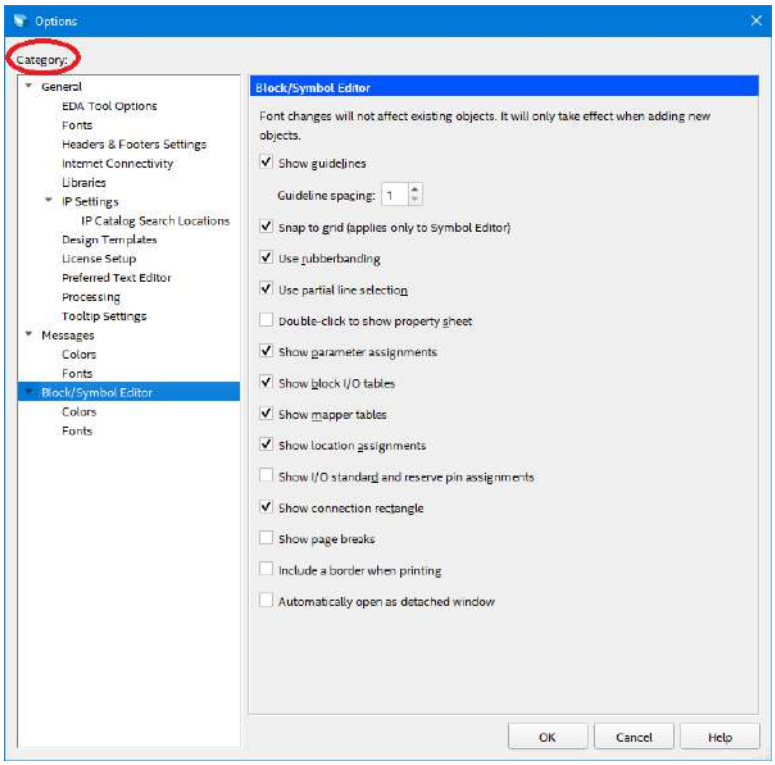


Рисунок 4.13 – Загальний вигляд вікна налаштувань для головного вікна САПР Quartus Prime

Редагування кольорової гами і відображення шрифтів у редактора структурних схем можна здійснити за допомогою відповідних пунктів Colors і Fonts вікна Options.

4.3.4 Створення функціонального блоку

Для створення функціональних блоків структурної схеми необхідно виконати наступну послідовність кроків:

1. На панелі інструментів редактора структурних схем (рисунок 4.11) вибрати інструмент Block Tool (введення блоку).

2. Натиснути в довільному місці поля редактора ліву кнопку миші, і утримуючи її, потягнути мишу, виділяючи прямокутну область, розміри якої відповідають блоку, що вводиться. Коригувати свої дії, можна використовуючи команди Undo (відмінити останню дію) та Redo (повторити останню дію) меню Edit (редагування) області II головного вікна САПР. Натискання комбінацій клавіш Ctrl+Z та Ctrl+Y виконують команди Undo та Redo відповідно.

3. Задіяти на панелі інструментів редактора вказівник виділення (Selection Tool), після чого навести вказівник миші на створений блок і натиснути праву клавішу. Відкриється контекстне меню, в якому треба вибрати пункт Properties, після чого відкриється діалогове вікно Block Properties (Властивості блоку), зображене на рисунку 4.14. Воно призначене для вводу назви блоку та позначення його вхідних/вихідних сигналів.

4. Вибравши вкладку General (загальні налаштування) у вікні Block properties, в верхньому рядку Name ввести назву блоку. Назву у нижньому рядку (Instance name) залишити без зміни – inst.

5. Вибрати вкладку I/Os. У стовпчику Name ввести назву першого вхідного контакту блоку. У вікні Type (тип) вказати тип виводу (INPUT – вхідний, OUTPUT – вихідний, BIDIR – двонаправлений). Після натискання клавіші Enter у вікні з’явиться відповідний запис. Виконуючи описані дії, ввести назви всіх необхідних вхідних і вихідних контактів блоку.

7. Після закінчення введення назв і типів всіх контактів блоку, натиснути кнопку “ОК”. Тепер введені назви можна побачити в таблиці блоку, яка розміщується в робочій області редактора схеми.

Аналогічним чином вводяться імена і типи контактів для всіх інших блоків проекту.

В результаті введення імен і типів контактів функціональних блоків може виявитися, що через малі розміри таблиць блоків вони повністю не відображаються. Збільшити розмір таблиці блоку можна двома способами:

1. Вибрати на панелі інструментів вказівник виділення, клікнути по потрібному блоку лівою кнопкою миші і розтягнути його до потрібного розміру за квадрати редагування, розміщені по контуру виділення.

2. Виділити зазначеним вище способом потрібний блок, клікнути правою клавішею миші і вибрати у контекстному меню команду AutoFit. Таблиця блоку автоматично збільшиться до потрібного розміру.

За необхідності, створені блоки можна розставляти на полі редактора структурної схеми найбільш зручним чином. Для цього на панелі інструментів необхідно вибрати інструмент вказівник виділення. Навести вказівник миші на блок, який треба перемістити, натиснути ліву кнопку миші і, утримуючи її, перемістити блок на нове місце.

Після виконання всіх описаних дій, необхідно зберегти зміни проекту (вибрати команду Save із меню File, або натиснути Ctrl+S).

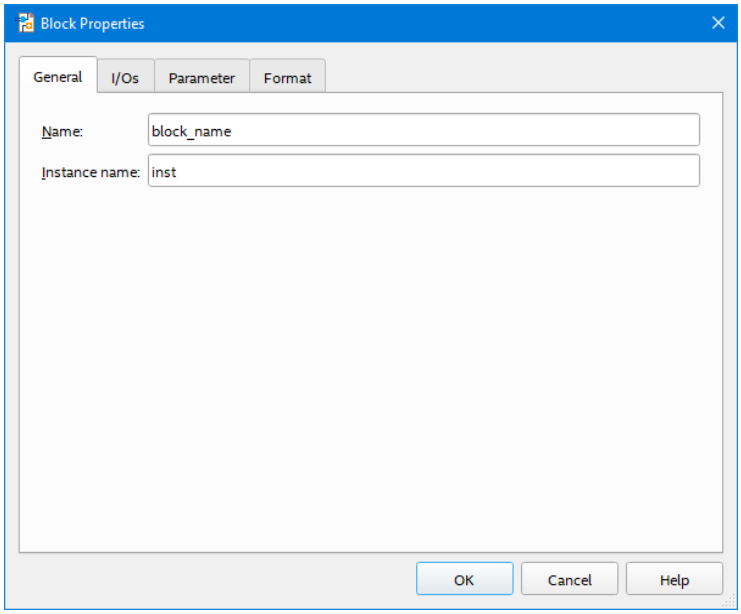


Рисунок 4.14 – Загальний вигляд вікна налаштувань властивостей блоку

4.3.5 Створення вхідних і вихідних контактів

Для створення вхідних/вихідних контактів на структурній схемі пристрою, необхідно на панелі інструментів вибрати Symbol Tool (введення компонента), в результаті чого з’явиться вікно Symbol (рисунок 4.15).

У лівій верхній частині вікна (Libraries) необхідно вказати шлях до бібліотеки із потрібними примітивами для чого із списку у вигляді дерева треба розкрити кореневий вузол, потім вузол primitives, а потім – вузол pin. Після вибору відповідного компонента бібліотеки (наприклад, Input), у правій частині вікна з'явиться зображення вибраного примітиву. У даному випадку це вхідний контакт. Після натискання кнопки ОК обраний символ з’явиться в основному вікні редактора схеми. При такому введенні автоматично вмикається режим Repeat-insert-mode (автоматичного повторного введення), при якому один символ можна вставляти в декілька місць проекту. Вибраний компонент бібліотеки прив’язується до курсора миші за лівий верхній кут. Тепер при натисканні лівої кнопки миші компонент встановлюється на вказане місце схеми. Далі його можна перенести в інше місце схеми і там його зафіксувати аналогічним способом. Для завершення вставки достатньо натиснути клавішу ESC або праву кнопку миші.

Якщо найменування компоненту відомо, а його місцезнаходження серед бібліотек і пакетів ні, тоді можна скористатися системою пошуку. Для цього достатньо вписати назву компоненту у поле Name (рисунок 4.15). Якщо такий існує – він одразу відобразиться у полі справа, а в дереві бібліотек буде відкрито його місцезнаходження.

Аналогічним чином на схему додаються всі контакти, які будуть використовуватися в проекті. Після введення всіх контактів необхідно

перезаписати файл проекту, щоб зберегти зміни.

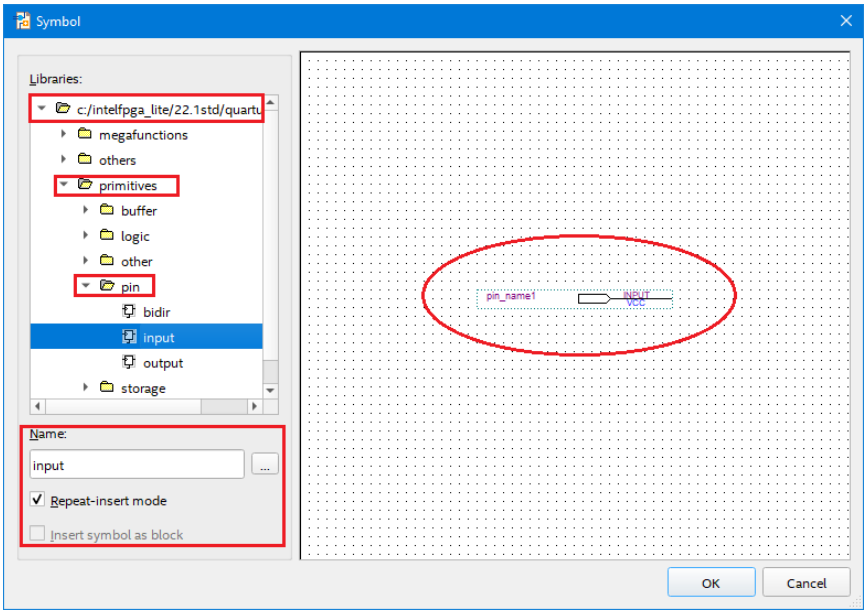


Рисунок 4.15 – Загальний вигляд вікна вибору готових компонентів із бібліотек САПР Quartus Prime для пристрою

4.3.6 Іменування контактів

Присвоєння назв вхідним/вихідним контактам схеми здійснюється у наступній послідовності:

1. Навести вказівник миші на необхідний контакт та двічі клікнути лівою кнопкою миші.

2. На екрані з’явиться вікно Pin Properties (властивості контакту), яке показано на рисунку 4.16. У цьому вікні треба вибрати вкладку General.

3. У полі Pin name(s) (назва контакту) ввести його назву.

4. У рядку Default value (значення по замовчуванню) виставляється вихідне значення, яке буде триматися в початковий момент або за відсутності керуючого сигналу; може приймати одне із двох значень: VCC – логічна одиниця, або GND – логічний нуль.

5. За допомогою вкладки Format можна змінювати кольори ліній та текстових написів.

6. Завершення вводу і налаштування контакту здійснюється натисканням кнопки OK.

Аналогічним чином вводяться імена всіх вхідних та вихідних контактів проекту.

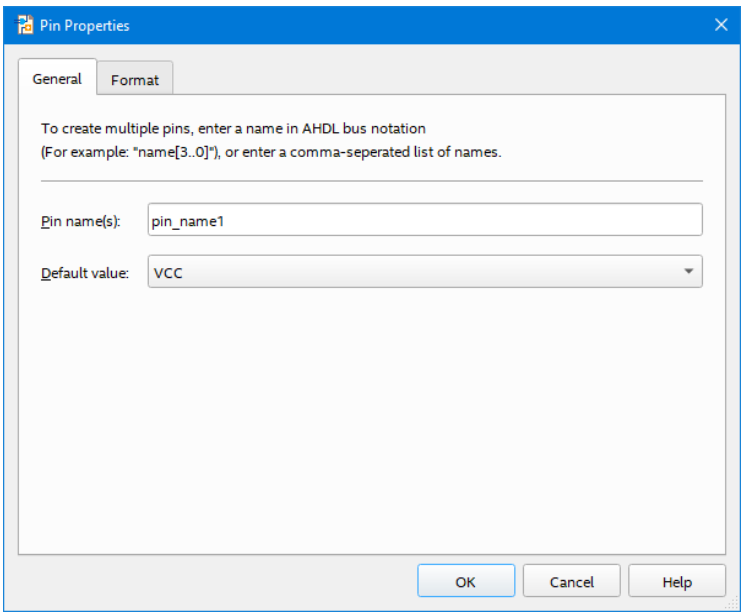


Рисунок 4.16 – Загальний вигляд вікна налаштувань властивостей контакту

4.3.7 З'єднання блоків пристрою між собою

Реалізація фізичних зв'яків між функціональними вузлами пристрою може здійснюватися за рахунок:

– сигнальних ліній (одиночних провідників) – команда Orthogonal Node Tool панелі інструментів редактора схем;

– шин (груп провідників) – команда Orthogonal Bus Tool панелі інструментів редактора схем;

– каналів передачі – команда Orthogonal Conduit Tool панелі інструментів редактора схем.

Крім того, більшість з'єднань між функціональними блоками можна виконувати за допомогою Selection and Smart Drawing Tool (автоматичного вказівника виділення). Цей вказівник (стрілка) автоматично перемикається в режим створення необхідних зв'язків при наведенні на позначення будь-якого контакту або виводу блока. Наприклад, після наведення стрілки на позначення контакту автоматично включається вказівник створення ортогональних ліній, а при наведенні на межу блоку – режим створення ортогональних шин. Ці ж режими можна вибирати на панелі інструментів.

САПР Quartus Prime автоматично створює зв'язки між з'єднаними блоками, а також між сигналами шини та входами/виходами блоків. Таким чином, шина виступає у ролі каналу для будь-якої кількості сигналів.

Для створення шини або каналу необхідно виконати наступні дії:

1. На панелі інструментів редактора схем вибрати необхідний інструмент (Orthogonal Bus Tool).

2. Для визначення місця початку шини клікнути лівою кнопкою миші на вхідний контакт, який потрібно з'єднати.

3. Утримуючи ліву кнопку миші, перемістити вказівник до межі необхідного блоку. При з'єднанні шини із будь-яким блоком на краю останнього автоматично з'явиться символ квадратик (mapper). Розподільник

дозволяє призначати вхідні/вихідні контакти блоку певним сигналам шини. Аналогічно з'єднуються і вихідні контакти.

Перевірити правильність виконання з'єднань можна за допомогою наступної послідовності дій:

1. У режимі Selection Tool навести вказівник на необхідну ділянку каналу зв'язку.

2. Натиснути праву клавішу миші і в контекстному меню вибрати пункт Properties, після чого відкриється вікно Conduit Properties (властивості каналу).

3. У цьому вікні треба вибрати команду Signal, після чого з'явиться вікно, в якому буде перераховано всі сигнали, що проходять через канал.

4.3.8 Трасування сигналів між блоками

САПР Quartus Prime надає різні шляхи трасування сигналів (map signals) між окремими блоками (blocks) та функціональними елементами (symbols) схеми. Опис доступних методів трасування представлено нижче, у таблиці 4.6.

Таблиця 4.6 – Методи трасування сигналів між блоками

Метод трасування	**Опис**
Автоматичне з'єднання **(Smart Connection)**	Якщо назви вхідних/вихідних сигналів одного блоку збігаються з іменами відповідних сигналів іншого, з'єднання формуються автоматично. Не треба створювати лінії для таких шин або каналів, а лише зазначити, які сигнали блоків повинні залишатися не приєднаними (якщо це вимагається). Це необхідно для запобігання автоматичному їх з'єднанню
З'єднання за спільними іменами сигналів або шин **(Connection by name)**	Якщо назви вхідних/вихідних контактів блоків, що з'єднуються, є різними, їх можна з'єднати з каналом або шиною по імені. Для цього необхідно призначити каналу назву, що відповідає імені входу/виходу, що приєднується. Цей спосіб є зручним при з'єднанні каналів без окреслення їх ліній на схемі. Для створення з'єднання між такими каналами і блоками спочатку необхідно перевірити, щоб кожен канал з'єднувався з блоком одним кінцем і не мав фізичного з'єднання із іншим блоком або функціональним елементом. З'єднання двох каналів по імені можна здійснити, надавши їм однакові імена. У такий же спосіб можна з'єднувати по імені й окремі провідники

Продовження таблиці 4.6

Метод трасування	Опис
Призначення з’єднань у явному вигляді за допомогою розподільників сигналів **(Mappers)**	Якщо назви входів/виходів блоків, що з’єднуються, різні, але блоки фізично з’єднані – розподіл сигналів можна задати в явному вигляді. Для цього необхідно присвоїти імена контактів блоків сигналам шини, а потім розподілити сигнали за контактами блоків, що з’єднуються

Щоб призначити назви сукупності сигналів для забезпечення можливості їх з’єднання по імені або логічно, необхідно зробити наступне:

1. У режимі вказівника виділення виділити необхідний провідник.

2. Натиснути праву кнопку миші і у контекстному меню вибрати рядок Properties.

3. У вікні Node Properties (властивості вузла з’єднання), що з’явилося (рисунок 4.17) перейти на вкладку General.

4. У полі Name ввести назву сигналу (наприклад, Signal_1.

5. Натиснути кнопку ОК, після чого сигнал буде автоматично додано до каналу, а його назва відобразиться над раніше виділеним провідником.

Для забезпечення передачі сигналів між блоками у випадку, якщо їх назви не збігаються, необхідно:

1. В режимі вказівника виділення навести курсор миші на вказівник вихідного контакта (mapper) таким чином, щоб поруч із курсором утворилося зображення таблиці і двічі клікнути лівою кнопкою миші.

2. У вікні із назвою Mapper Properties (властивості карти з’єднань) вибрати вкладку General.

3. У списку Type обрати функціональне призначення цього виводу.

4. Перейти на вкладку Mapping (складання карти з’єднань) та у списку I/0 on block (входи та виходи блоку) вибрати необхідний сигнал.

5. У списку Signals in conduit (сигнали в каналі) вибрати назву сигналу, який має бути присутнім у каналі зв’язку.

6. Для завершення з’єднання необхідно натиснути кнопку Add. При цьому, введена інформація з’явиться у полі Existing mappings (існуючі з’єднання).

7. Для виходу необхідно натиснути кнопку ОК. При цьому на структурній схемі пристрою, що розробляється, з’явиться відповідна таблиця, що відображає введену інформацію про призначені з’єднання.

8. Зберегти оновлення в файлі проекту, використовуючи команду Save.

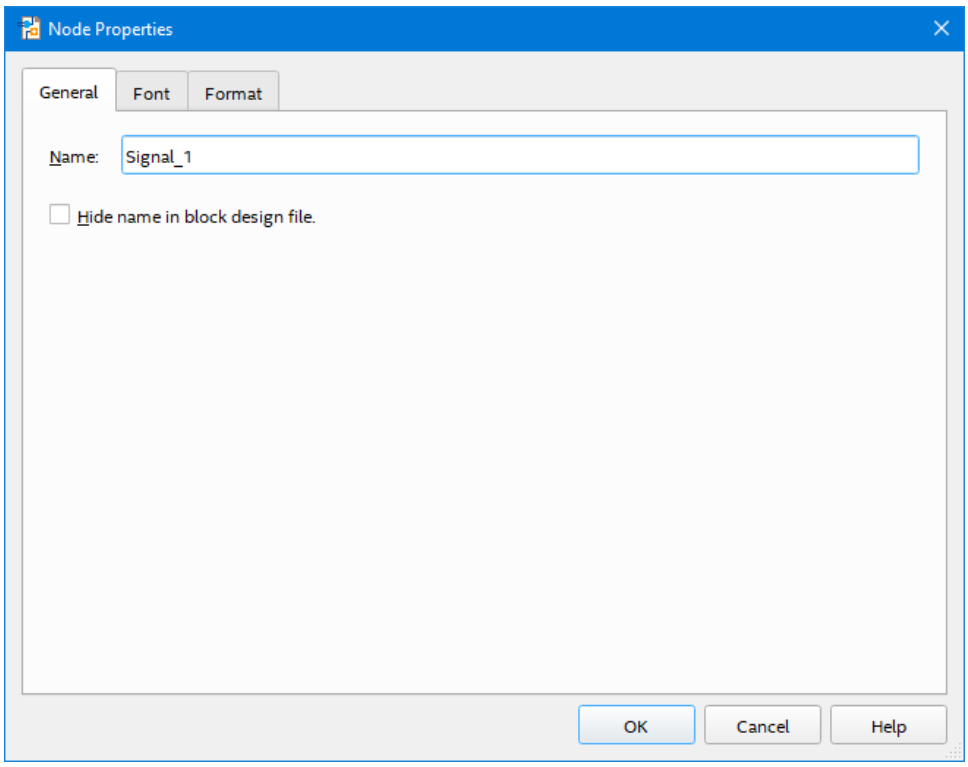


Рисунок 4.17 – Загальний вигляд вікна налаштувань властивостей з’єднання

5 СТВОРЕННЯ ОПИСІВ ОКРЕМИХ БЛОКІВ ПРОЕКТУ

Для того, щоб створити файли опису окремих блоків проекту, попередньо необхідно виконати наступну послідовність дій:

1. Запустити САПР Quartus Prime.

2. В області II (рисунок 4.1) натиснути на меню File, в якому вибрати команду Open Project (відкрити проект).

3. В випадному меню вибору типу файлів (правий нижній кут вікна, що з'явилося) вибрати розширення файлу, що відповідає Quartus Prime Project File (*.qpf або *.par, або *.qar), вказати шлях до проекту, а у вільному полі вибрати назву проекту та натиснути кнопку відкриття.

4. У другому вікні вибрати назву потрібного файлу верхнього рівня ієрархії проекту та натиснути кнопку відкриття.

5. У головному вікні САПР з'явиться вибраний файл верхнього рівня проекту, що містить структурну схему пристрою.

6. Виділити лівою кнопкою миші на структурній схемі проекту блок, опис якого необхідно ввести, та натиснути праву кнопку миші.

7. У вікні, що з'явилося, курсором миші зробити вибір рядка Create Design File from Selected Block (створити файл проекту на основі виділеного блоку) і натиснути ліву кнопку миші.

8. Вікно, яке з'явилося, забезпечує можливість введення або поведінкового опису алгоритму роботи блоку мовами високого рівня (AHDL, VHDL або Verilog), або його схемотехнічного подання (Schematic). Залежно від вибраного типу опису, Quartus Prime створить файл опису блоку із відповідним розширенням. Назва файлу, що створюється, автоматично задається як назва блоку, який було виділено спочатку. При виборі мови опису AHDL буде створено файл *.tdf, VHDL – файл *.vhd, при Verilog – *.v, а при виборі Schematic – *.bdf. Необхідно звернути увагу, що в меню повинен бути встановлений прапорець біля напису Add the new design file to the current project (додати новий файл до поточного проекту). Після вибору типу опису файлу необхідно натиснути кнопку ОК.

9. САПР Quartus Prime відповідним написом, що з'явиться на екрані, повідомляє про успішне відкриття потрібного файлу та відкриває його у своєму головному вікні. Тепер можна вводити опис блоку.

5.1 Загальні принципи схемотехнічного опису поведінки блоку

Процес створення схемотехнічного опису зводиться до складання в графічному редакторі структурно-логічної схеми блоку наявними у бібліотеках САПР Quartus Prime стандартних компонентів. Для цього необхідно виконати стандартну процедуру, яка складається із наступних дій:

1. Сформулювати алгоритм роботи блоку.

2. На основі сформульованого опису, формалізувати алгоритм роботи блоку та записати відповідні функції алгебри логіки.

3. На основі записаних логічних функцій із використанням наявних

бібліотечних компонентів, скласти структурно-логічну схему пристрою та ввести її у вікно редактора схем, який інтегровано в САПР Quartus Prime.

При створенні схем можна використовувати наступні види компонентів:

– найпростіші логічні елементи (кон'юнктори, диз'юнктори, інвертори), тригери, вхідні/вихідні контакти та ін.;

– мегафункції, тобто, модулі, створені компанією Altera;

– компоненти , що були створені раніше у текстовому чи графічному редакторі.

5.2 Загальні принципи опису із використанням мов високого рівня

Мова опису апаратури (HDL – Hardware Description Languages) є формальним записом, який може бути використано на всіх етапах розробки цифрових електронних пристроїв. Це можливо тому, що мова легко сприймається як машиною, так і людиною. Вона може використовуватися на етапах проектування, верифікацій, синтезу та тестування апаратури, так само легко, як і для передачі даних про проект, його модифікації та супровід. Мови опису апаратури тривалий час були прерогативою досить вузького класу розробників спеціалізованих інтегральних схем. З появою такої елементної бази як ПЛІС стрімко розширилося коло користувачів, зацікавлених у використанні сучасних способів опису проекту.

Мови опису апаратури можна досить умовно поділити на мови високого та низького рівня. До першого виду прийнято відносити VHDL і Verilog, а до другого – AHDL, Abel HDL та ін. Мови високого рівня дозволяють забезпечити певну мобільність опису при міграції на іншу елементну базу, тоді як мови низького рівня орієнтовані на використання архітектурних особливостей мікросхем ПЛІС конкретних виробників.

При вводі опису блоку мовою опису апаратури, текстовий редактор забезпечує деякі наступні переваги:

– нумерація рядків коду;

– використання заздалегідь створених мовних конструкцій і шаблонів;

– відображення ключових слів мови вибраними кольорами;

– підказка про необхідність збереження вмісту файлу.

6 КОМПІЛЯЦІЯ ПРОЕКТУ

Процес компіляції проектів в САПР Quartus Prime складається із кількох етапів (кожен із яких виконується окремим спеціалізованим модулем):

– перевірка проекту на наявність помилок;

– логічний синтез;

– розміщення та трасування зв'язків на кристалі ПЛІС;

– генерація вихідних файлів для моделювання проекту;

– аналіз часових параметрів і характеристик;

– програмування.

На початку компіляції проекту з нього видобувається інформація про ієрархічні зв'язки між складовими файлами, а опис проекту перевіряється на наявність синтаксичних та лексичних помилок. Потім створюється організаційна карта проекту, і всі файли перетворюються на єдину базу даних, із якою потім і буде працювати САПР.

Компіляція може виконуватися із урахуванням вимог, заданих користувачем, до яких належать:

– забезпечення потрібних часових характеристик;

– підвищення швидкодії (робочої частоти цифрових пристроїв);

– оптимізація використання ресурсів ПЛІС.

В процесі роботи компілятор створює файли для програмування та конфігурування ПЛІС.

Проміжні та остаточні результати компіляції у САПР можна переглянути на вкладці Compilation Report (звіт про компіляцію).

Програмування та конфігурування ПЛІС фірми Altera може бути здійснено як за допомогою вбудованих засобів САПР, так і за допомогою стандартних промислових засобів програмування.

6.1 Налаштування компілятора

Quartus Prime дозволяє виконати компіляцію як всього проекту, так і будь-якої його частини. Зокрема, під час налаштування компілятора визначають:

– фокус компіляції (Compilation focus) – частину проекту, що підлягає компіляції;

– тип компіляції;

– сімейство та тип ПЛІС;

– інші додаткові параметри.

При створенні нового проекту, САПР встановлює значення всіх необхідних параметрів за замовчуванням, але такі параметри можна перевизначити відповідно до поставлених вимог. Крім того, є можливість вибору різних параметрів налаштування безпосередньо в процесі компіляції.

Нижче розглянуто методику налаштування основних параметрів компіляції, яка включає:

– перегляд основних налаштувань компілятора;

– визначення сімейства та типу ПЛІС;

– визначення режиму компіляції;

– визначення та налаштування параметрів логічного синтезу та трасування з’єднань;

– визначення параметрів верифікації проекту на етапі компіляції.

Як зазначалося вище, компілятор Quartus Prime є модульним, до його складу входять наступні (модулі, позначені як * є опціональними, тобто, їх наявність може залежить від налаштувань або версії САПР):

– модуль аналізу та синтезу проекту (Analysis & Synthesis);

– модуль розміщення і трасування (Fitter);

– модуль транслятора для програматора ПЛІС (Assembler);

– модуль аналізу часових параметрів і характеристик (Timing Analyzer);

– помічник проектувальника (Design Assistant)*;

– редактор списку з’єднань і кіл (EDA Netlist Writer);

– інтерфейс бази даних компілятора (Compiler Database Interface)*.

Повну компіляцію проекту можна запустити, вибравши із меню Processing (область II на рисунку 4.1) пункт Start Compilation або натиснувши комбінацію клавіш Ctrl+L. При цьому будуть послідовно запущені усі модулі компілятора. Також можна запускати потрібні модулі компілятора окремо. Для цього необхідно вибрати пункт Start меню Processing і потім вибрати модуль, який необхідно запустити для виконання.

Після виконання всього циклу повної компіляції, Quartus Prime повідомляє про її завершення і відображає результати у вигляді кількості знайдених помилок, зроблених попереджень або критичних попереджень, тощо.

Схему алгоритму виконання етапів типової компіляції проекту у САПР Quartus Prime представлено нижче, на рисунку 6.1.

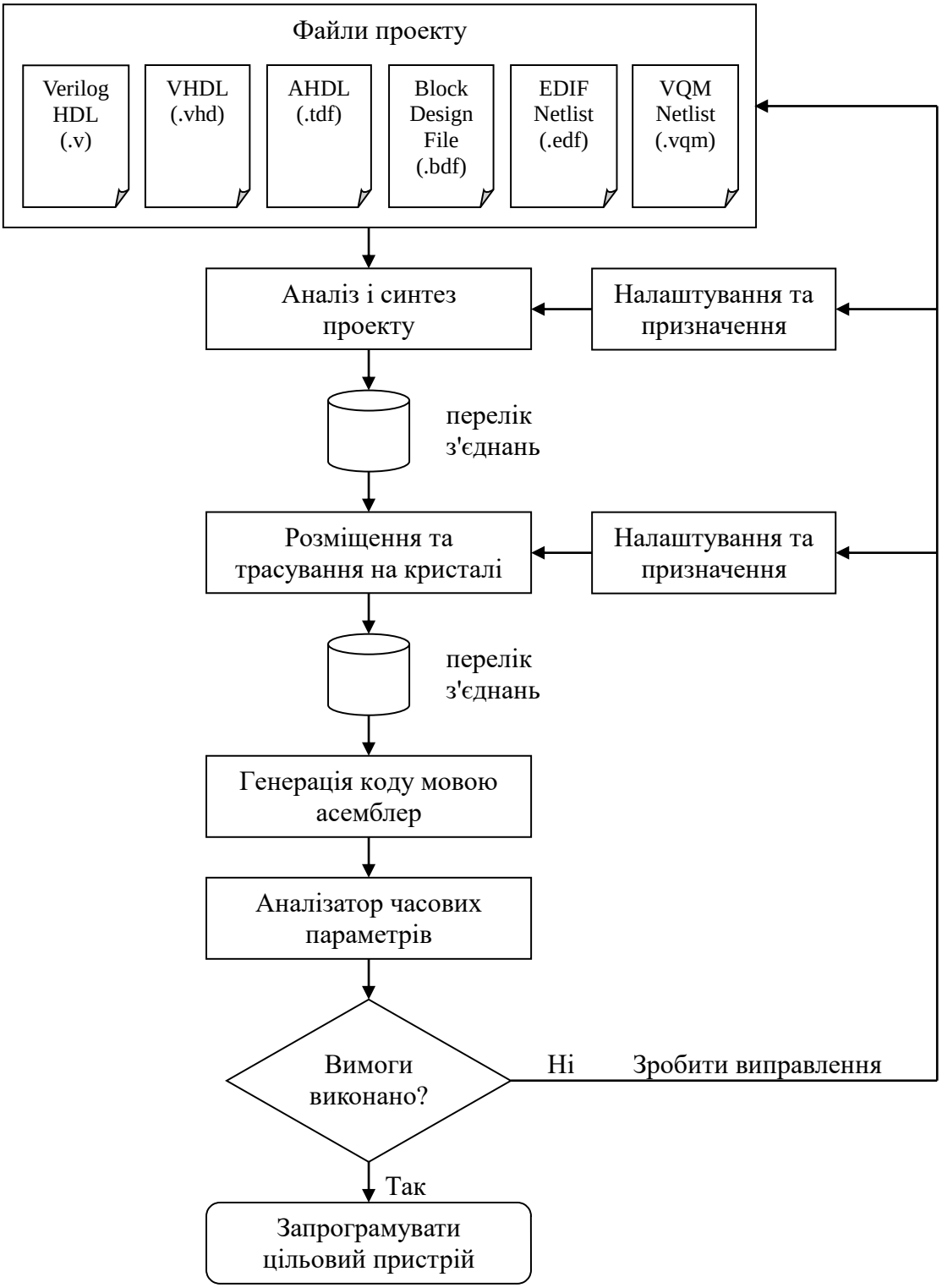
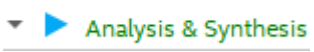
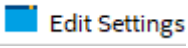
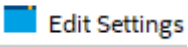
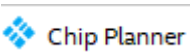
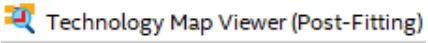
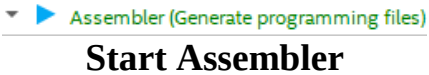
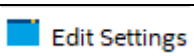
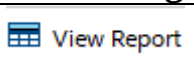


Рисунок 6.1 – Послідовність етапів типової компіляції проекту в Quartus Prime

Запустити окремий модуль компілятора можна за допомогою кнопок, розміщених під назвами відповідних модулів (таблиця 6.1).

Таблиця 6.1 – Компіляція модулів проекту

Модуль компіляції	**Призначення**	
Analysis & Synthesis	Модуль аналізу та синтезу проекту: – перевіряє файли на помилки; – створює і наповнює базу даних, яка інтегрує всі файли в єдину ієрархію; – синтезує та оптимізує результати розробки; – виробляє технологічну відповідність Вашої розробки конкретному пристрою ПЛІС.	
	 **Start Analysis & Synthesis**	Запуск модуля аналізу та синтезу проекту
	 **Edit Settings**	Відкрити вікно налаштувань цього модуля
	 **View Report**	Відкрити файл звітності цього модуля
Fitter (Place & Route)	Модуль “складальника” розміщує логіку проекту в мікросхему ПЛІС, використовуючи результати роботи модуля аналізу та синтезу	
	 **Start Fitter**	Запуск модуля “складальника”
	 **Edit Settings**	Відкрити вікно налаштувань цього модуля
	 **View Report**	Відкрити файл звітності цього модуля
	 **Chip Planner**	Відкрити топологічну структуру проекту, його розміщення у ПЛІС
	 **Technology Map Viewer (Post-Fitting)**	Відкрити структуру внутрішніх ресурсів у ПЛІС, задіяних у проекті
Assembler	Асемблер завершує компіляцію проекту, перетворюючи результати роботи “складальника” для програмування ПЛІС (один або декілька файлів)	
	 **Start Assembler**	Запуск модуля асемблера
	 **Edit Settings**	Відкрити вікно налаштувань цього модуля
	 **View Report**	Відкрити файл звітності цього модуля

Продовження таблиці 6.1

Модуль компіляції	**Призначення**	
Timing Analysis	Часовий аналізатор призначено для аналізу, налаштування та оцінювання часових затримок (продуктивності) всієї логіки проекту. Використовує результати роботи модулів аналізу та синтезу, а також “складальника”, однак є можливість зробити часовий аналіз на ранніх етапах (без “складальника”)	
	Timing Analysis **Start Timing Analysis**	Запуск модуля часового аналізатора
	Edit Settings **Edit Settings**	Відкрити вікно налаштувань цього модуля
	View Report **View Report**	Відкрити файл звітності цього модуля

Список у лівій частині вікна звіту компілятора (Compilation Report) дозволяє отримати детальний звіт про результати виконання компіляції (як це показано на рисунку 6.2).

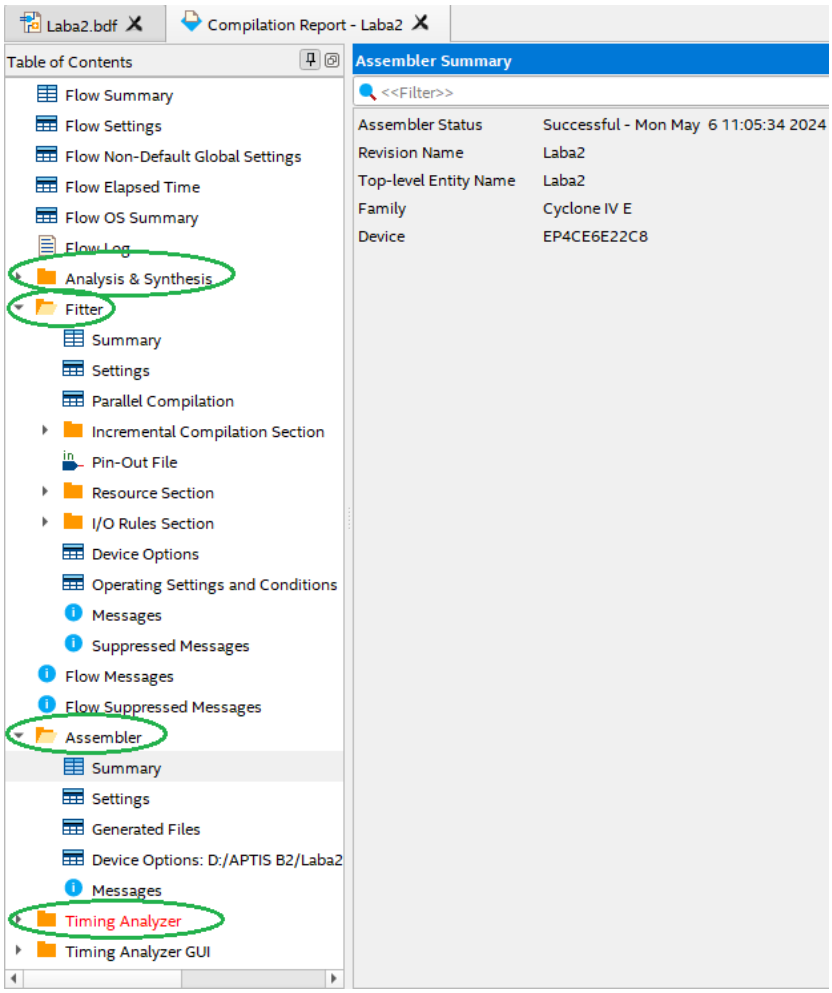


Рисунок 6.2 – Вміст вікна звіту компілятора (Compilation Report)

6.1.1 Перегляд та налаштування основних властивостей компілятора

Переглянути та налаштувати основні параметри компілятора можна:

– за допомогою вікна Settings (налаштування проекту), яке можна викликати із розділу Assignments (призначення), розташованого в області II (дивись рисунок 4.1), командою Settings (установки);

– безпосередньо за модулями компілятора, із використанням вікна Tasks (область V на рисунку 4.1), активуючи пункт Edit Settings відповідного модуля.

Обидва способи дозволяють відкрити вікна властивостей відповідних модулів компілятора і, за необхідності, внести відповідні зміни параметрів компіляції. У лівій частині вікна, що відкривається (рисунок 6.3), показано загальне дерево можливих параметрів проекту, а в його правій частині – параметри обраного розділу призначень.

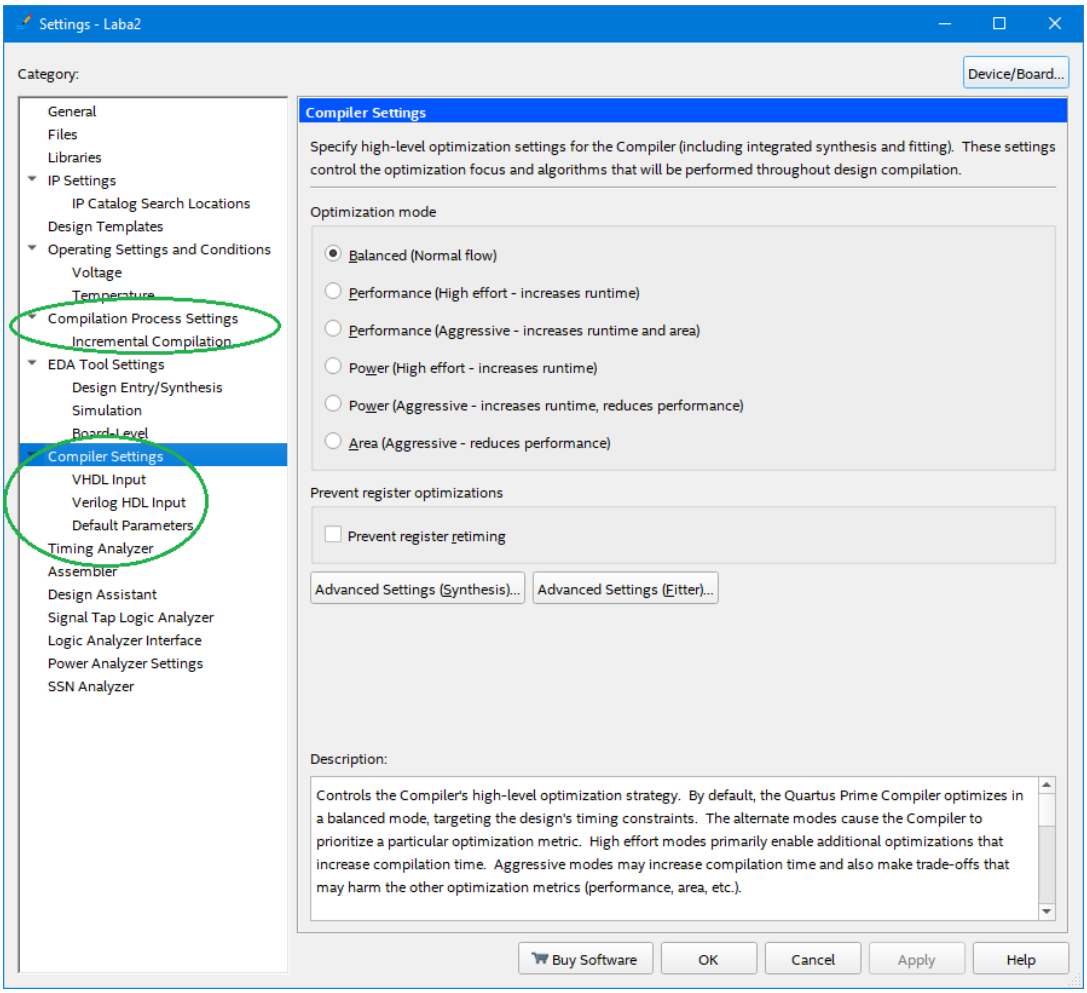


Рисунок 6.3 – Вміст вікна Settings налаштувань компілятора САПР

6.1.2 Визначення сімейства та типу ПЛІС

Вибрати сімейство та конкретний тип ПЛІС, призначений для реалізації проекту, можна за допомогою відповідного вікна, яке викликається із розділу Assignments (призначення) командою Device (пристрій). В результаті буде відкрито вікно Device, вміст якого аналогічний зображеному на рисунку 4.7.

У верхній частині цього вікна необхідно вибрати сімейство ПЛІС (випадне меню Family (сімейство)).

Вибір конкретного типу ПЛІС можна зробити одним із двох способів:

– довірити вибір відповідного пристрою, що задовольнятиме всім критеріям вимог, на розсуд модуля “складальника” компілятора, вибравши пункт “Auto device selected by the Fitter”;

– вручну, вибравши пункт “Specific device selected in “Available devices” list”.

Правіше вікна зі списком доступних приладів, під написом Show in “Available devices” list (показати у списку доступних приладів) можна вибрати тип корпусу (Package), кількість виводів (Pin count) та градацію швидкодії (Speed Grade) ПЛІС, обраної для використання. Якщо ці параметри спеціально не обговорюються, то у всіх пунктах можна встановити Any (будь-який).

Слід зазначити, що вибір конкретних параметрів ПЛІС у розділі “Available devices” list призводить до зміни списку доступних приладів у вікні Available devices. Тому можна або вибрати тип пристрою зі списку Available devices, задавши у всіх рядках розділу Show in “Available devices” list вибір довільних параметрів (Any), або зробити у розділі Show in “Available devices” list вибір конкретних параметрів ПЛІС, що призведе до звуження переліку доступних пристроїв. Таким чином, якщо в розділі Show in “Available devices” list задати вказані для ПЛІС параметри, то у списку доступних приладів залишаться лише відповідні пристрої. Вибір параметрів у вікні Show in “Available devices” list зручний у випадку, якщо вже відомі конструктивні параметри пристрою, що розробляється.

У розділі Available devices додатково вказано напругу живлення ядра обраної ПЛІС (стовпчик Core voltage).

6.2 Виконання компіляції проекту

Під час своєї роботи компілятор використовує поточні налаштування, автоматично розпізнає та обробляє всі файли, що стосуються об'єкта компіляції, зокрема:

- ***.inc** – файли включення, що містять опис функцій мовою AHDL;
- ***.mif** – файли ініціалізації пам’яті;
- ***.hex** – файли ініціалізації пам’яті у шістнадцятковому форматі Intel;
- ***.psf, *.esf, *.and, *.csf**- файли параметрів проекту та компіляції.

Попередження (Warnings), що генеруються в процесі компіляції, та повідомлення про помилки (Error messages) відображаються у вікні процесора повідомлень (Messages), якому відповідає область VI на рисунку 4.1.

6.2.1. Запуск компілятора

Для запуску компілятора необхідно в меню Processing вибрати команду Start Compilation (запустити компіляцію). Починається процес компіляції. При цьому активуються наступні вікна:

– Tasks (задачі), в якому відображається процес та результати проходження необхідних етапів компіляції;

– Compilation Report (звіт про компіляцію), в якому фіксується витрачений час.

Процес компіляції виконується у фоновому режимі, тому при тривалій компіляції можлива робота із іншими вікнами САПР Quartus Prime або іншими запущеними під операційною системою програмами.

Після завершення компіляції на екрані з'являється відповідний напис, що свідчить про завершення процесу та кількість знайдених помилок та інформаційних повідомлень (область VI на рисунку 4.1).

6.2.2 Локалізація джерела повідомлень

Під час компіляції у вікні процесора повідомлень (Messages) з’являються інформаційні повідомлення, повідомлення про помилки, попередження та критичні попередження, зроблені компілятором. Ці записи зберігаються у певному місці файлу проекту або іншому вихідному файлі.

Щоб знайти (локалізувати) частину проекту, що є джерелом повідомлення необхідно двічі клацнути на повідомлення, що цікавить вас, після чого в головному вікні системи з'явиться файл, до якого відноситься дане повідомлення. До того ж, на екрані буде кольором безпосередньо виділено джерело повідомлення.

6.2.3 Перегляд звіту про компіляцію

Інформація про результати компіляції проекту виводиться на вкладці Compilation Report (звіт про компіляцію). Безпосередньо після завершення компіляції у правій частині вкладки з’являється зведений звіт (Flow Summary), що містить у собі:

– інформацію про дату та час проведення компіляції;

– версію САПР Quartus Prime;

– назву контрольного та файлу верхнього рівня проекту;

– сімейство та тип використаної мікросхеми ПЛІС;

– тип компіляції (проміжна або остаточна);

– відповідність проекту заданим вимогам щодо часових параметрів, кількості використаних логічних блоків, виводів та обсягу пам’яті.

У лівій частині вікна є впорядкований за ієрархією деревоподібний каталог повідомлень окремих блоків компілятора. Тут можна отримати повну інформацію про всі етапи компіляції, включаючи попередньо зроблені установки і результати роботи окремих блоків компілятора. Додатково надається можливість друку вибраних частин звіту.

7 ВЕРИФІКАЦІЯ ТА ТЕСТУВАННЯ ПРОЕКТУ ШЛЯХОМ МОДЕЛЮВАННЯ

Моделювання дозволяє визначити реакцію розробленого цифрового пристрою на задані вхідні сигнали, тобто, переконатися в коректності його функціонування.

Вихідними даними для моделювання є зовнішні впливи, задані як деякий вхідний вектор (набір кодових слів). Підсистема моделювання (Simulator), вбудована у САПР Quartus Prime, синтезує вихідні сигнали проекту, що відповідають його реакції на задані вхідні впливи (згідно алгоритму), які є дуже близькими до реакції реальної ПЛІС. У типових задачах розробник задає набори вхідних векторів та аналізує отримані моделюванням вихідні сигнали.

Залежно від поставленої мети підсистема моделювання дозволяє виконати:

– функціональне моделювання проекту (Functional Simulation), при якому перевіряється правильність опису та логічного функціонування проекту;

– моделювання з урахуванням часових параметрів реальної ПЛІС (Timing Simulation), що дозволяє перевірити не тільки правильність логічного функціонування проекту, але і його роботу з урахуванням реальних параметрів обраної ПЛІС у найжорсткіших умовах експлуатації.

У більш ранніх версіях САПР Quartus (I і II) підсистема моделювання була повною і дозволяла робити гнучкі налаштування процесу моделювання і побудови часових діаграм. Однак, у версіях Quartus Prime, принаймні в безкоштовних (Lite Edition), даний симулятор отримав жорсткі обмеження тривалості часу моделювання (в 1 мс), що може бути недостатнім навіть при тестуванні навчальних проектів. Натомість, альтернативним способом побудови часових діаграм для верифікації створених цифрових пристроїв є використання середовища Modelsim (або його більш просунутої версії Questa).

Розглянемо послідовність дій для верифікації цифрових пристроїв у середовищі Modelsim більш детально.

7.1 Можливості середовища Modelsim

Modelsim є середовищем опису та моделювання, що розробляється і супроводжується компанією Mentor Graphics, містить величезну кількість різноманітних бібліотек моделей компонентів та розповсюджується під ліцензіями GPL (General Public License) та LGPL (Lesser GPL).

ModelSim – це багатомовне середовище, що надає користувачу набір інструментів для моделювання систем цифрової електроніки за допомогою мов опису апаратури, таких як VHDL, Verilog і SystemC, забезпечує верифікацію та тестування функціональності і синхронізацію систем без необхідності розробки фізичного прототипу.

Робота із середовищем Modelsim, зокрема, створення графічного інтерфейсу для користувача, здійснюється за допомогою скриптової мови високого рівня Tcl (Tool command language).

ModelSim пропонує широкий набір функцій, серед яких можна виділити:

– графічний інтерфейс користувача для створення, редагування та відлагодження коду, написаного різними мовами опису;

– підтримку ієрархічної структури моделі, що забезпечує зручність при роботі із великими та складними пристроями/системами;

– перевірку часу для виявлення проблем із синхронізацією та продуктивністю;

– інструменти аналізу форми та параметрів сигналу для візуалізації результатів моделювання;

– можливість розширення функціональності за допомогою API (Application Programming Interface), який дозволяє підключення і використання скриптів і плагінів, створених користувачем.

ModelSim підтримує 3 основні мови опису апаратури:

– VHDL (Very High Speed Integrated Circuit Hardware Description Language) – мова опису високошвидкісних пристроїв, що використовується для опису як цифрових, так і аналогових схем.

– Verilog – мова опису апаратури (схожа на мову С), що широко використовується в промисловості для моделювання та верифікації цифрових систем;

– SystemC – мова опису апаратури для моделювання та розробки систем на кристалі (SoC – System on Chip) та вбудованих систем (Embedded systems).

Процес моделювання в ModelSim включає наступні кроки:

1. Створення моделі шляхом її опису однією із мов.

2. Компіляцію моделі, тобто, перетворення HDL-коду в виконуваний.

3. Налаштування моделі, тобто, налаштування параметрів моделювання (тестові стимули, сигнали синхронізації та критерії збою, та ін.).

4. Моделювання, що включає оцінку поведінки моделі та генерацію звітів із результатами.

5. Аналіз результатів, тобто, оцінка правильності функціонування та виявлення існуючих помилок/проблем.

Переваги використання середовища Modelsim:

– підтримка декількох мов опису апаратури для забезпечення більшої гнучкості процесу моделювання;

– вбудований графічний відлагоджувач для швидкого виявлення та усунення помилок;

– сумісність із іншими інструментами від Mentor Graphics для забезпечення комплексного циклу проектування;

– широкий набір функцій для всебічного моделювання та верифікації;

– потужна спільнота користувачів для підтримки та обміну досвідом.

Недоліки середовища Modelsim:

– може знадобитися час, щоб ознайомитися із особливостями інтерфейсу і функціональністю (може бути складним для початківців);

– висока вартість ліцензії для професійної версії ModelSim;

– моделювання великих і складних систем займає багато часу;

– ефективність моделювання залежить від апаратного забезпечення ЕОМ;

– надмірно складний і важкий для невеликих і простих моделей.

Сферамии застосування ModelSim є:
– проектування цифрових пристроїв різної складності;
– проектування і розробка систем на кристалі;
– проектування і розробка вбудованих систем;
– верифікація пристроїв на системному рівні.
Альтернативними до ModelSim середовищами є наступні:
– Xilinx Vivado Simulator;
– Cadence Incisive;
– Synopsys VCS;
– GHDL;
– Verilator.

7.2 Послідовність запуску процесу моделювання цифрових пристроїв в середовищі Modelsim

Завдяки використанню САПР Quartus Prime, процес моделювання значно спрощується. Фактично, для запуску Modelsim і побудови часових діаграм роботи пристрою необхідно запустити виконуваний скриптовий файл для інтерпретатора MS Windows, що має розширення *.bat. Рекомендується виконувати запуск цього файлу натисканням комбінації клавіш Shift+Enter, оскільки в цьому випадку вікно консолі не буде закриватися, що буде корисним, оскільки дозволить побачити помилки і проблеми, які можуть виникнути при запуску. Нижче показано вміст типового скриптового файлу.

```
rem recreate a temp folder for all the simulation files
rd /s /q sim
md sim
cd sim

rem start the simulation
vsim -do ../modelsim_script.tcl

rem return to the parent folder
cd ..
```

Як можна побачити, окрім допоміжних команд для створення окремого каталогу моделювання (sim) і його наповнення, основною дією є запуск на виконання файлу modelsim_script.tcl, в якому знаходиться скрипт, написаний мовою Tcl. Розглянемо більш детально вміст цього файлу.

```
# create modelsim working library
vlib work

# compile all the Verilog sources
vlog  ../testbench.v ../../Laba2.v

# open the testbench module for simulation
vsim work.testbench

# add all testbench signals to time diagram
add wave sim:/testbench/*

# run the simulation
run -all
```

```
# expand the signals time diagram
wave zoom full
```

Як видно із прикладу, для коректної роботи скрипта необхідна наявність двох файлів, написаних мовою Verilog: “testbench.v” і “Laba2.v”, які будуть підлягати компіляції і симуляції. Варто зазначити, що перед назвою кожного із них стоїть один або кілька специфікаторів “../”, які вказують на відносність шляху розташування файлу та рівень його вкладеності в каталозі проекту САПР Quartus Prime.

Файл testbench.v (слово testbench перекладається як тестовий стенд, або стенд для випробувань) містить т.зв. “план тестування”, тобто описує найменування, розрядність і кількість вхідних/вихідних сигналів, правила і порядок формування вхідних тестових сигналів, а також правила і порядок відображення вихідних результуючих сигналів та ін. Слід зауважити, що назва “testbench” є зарезервованою і наперед встановленою, тому, використання файлу із іншою назвою призведе до помилки. В якості прикладу, нижче наведено вміст файлу testbench.v для лабораторної роботи №2.

```
//testbench is a module which only task is to test another module
//testbench is for simulation only, not for synthesis
module testbench;
    //input and output test signals
    reg  [3:0] x;
    wire y;

    //creating the instance of the module we want to test
    //lab2 - module name
    //dut - instance name ('dut' means 'device under test')
    Laba2 dut (x, y);

    // do at the beginning of the simulation
    initial
        begin
            x = 4'b0000;      //set test signals value
            #10;              //pause
            x = 4'b0001;      //set test signals value
            #10;              //pause
            x = 4'b0010;      //set test signals value
            #10;              //pause
            x = 4'b0011;      //set test signals value
            #10;              //pause
            x = 4'b0100;      //set test signals value
            #10;              //pause
            x = 4'b0101;      //set test signals value
            #10;              //pause
            x = 4'b0110;      //set test signals value
            #10;              //pause
            x = 4'b0111;      //set test signals value
            #10;              //pause
            x = 4'b1000;      //set test signals value
            #10;              //pause
            x = 4'b1001;      //set test signals value
            #10;              //pause
            x = 4'b1010;      //set test signals value
            #10;              //pause
            x = 4'b1011;      //set test signals value
            #10;              //pause
            x = 4'b1100;      //set test signals value
            #10;              //pause
            x = 4'b1101;      //set test signals value
```

```
            #10;                //pause
            x = 4'b1110;        //set test signals value
            #10;                //pause
            x = 4'b1111;        //set test signals value
            #10;                //pause
        end

    //do at the beginning of the simulation
    //print signal values on every change
    initial
        $monitor("x=%b y=%b", x, y);

    //do at the beginning of the simulation
    initial
        $dumpvars;  //iverilog dump init
endmodule
```

На відміну від “testbench.v”, вміст якого створюється і наповнюється розробником або тестувальником, “Laba2.v” є файлом, який автоматично генерується за допомогою САПР Quartus Prime. Він містить опис мовою Verilog цифрового пристрою, реалізованого у вигляді структурної схеми. Для його створення слід виконати наступну послідовність дій:

1. Завершити процес проектування, тобто, закінчити роботу над проектом і зберегти всі зміни, які вносилися у файли.

2. Запустити процес повної компіляції, дочекатися її закінчення і переконатися у відсутності помилок та критичних попереджень.

3. Закрити звіт про результати компіляції і повернутися до вкладки із файлом, в якому розміщується схема пристрою (*.bdf).

4. У розділі File головного меню (область II на рисунку 4.1) вибрати пункт “Create / Update”, де із випадного меню вибрати пункт “Create HDL Design File from Current File...”.

5. Після того, як на екрані з’явиться вікно, зовнішній вигляд якого показано нижче, на рисунку 7.1, встановити точно таке ж налаштування, як і на рисунку (тобто, вибрати мову Verilog для опису створеного пристрою, а також перевірити правильність місця розташування майбутнього файлу опису). Натиснути кнопку “OK”.

6. Дочекатися результатів створення описового файлу і пересвідчитися у відсутності помилок або попереджень. В якості прикладу далі наведено вміст файлу “Laba2.v”.

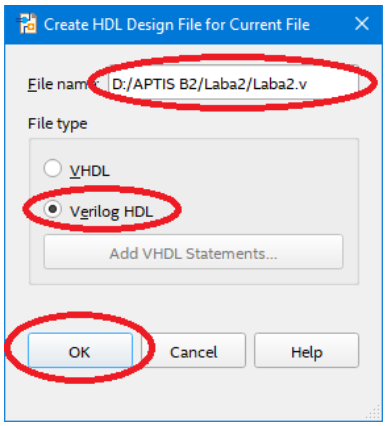


Рисунок 7.1 – Налаштування генератора коду мови Verilog

```
//Copyright (C) 2022  Intel Corporation. All rights reserved.
//Your use of Intel Corporation's design tools, logic functions
//and other software and tools, and any partner logic
//functions, and any output files from any of the foregoing
//(including device programming or simulation files), and any
//associated documentation or information are expressly subject
//to the terms and conditions of the Intel Program License
//Subscription Agreement, the Intel Quartus Prime License Agreement,
//the Intel FPGA IP License Agreement, or other applicable license
//agreement, including, without limitation, that your use is for
//the sole purpose of programming logic devices manufactured by
//Intel and sold by Intel or its authorized distributors.  Please
//refer to the applicable agreement for further details, at
//https://fpgasoftware.intel.com/eula.

//PROGRAM         "Quartus Prime"
//VERSION         "Version 22.1std.0 Build 915 10/25/2022 SC Lite Edition"
//CREATED         "Tue May 14 22:11:43 2024"

module Laba2(
      x,
      y
);

input wire  [3:0] x;
output wire [4:1] y;

wire  a;
wire  [3:0] nx;
wire  SYNTHESIZED_WIRE_0;
wire  SYNTHESIZED_WIRE_1;
wire  SYNTHESIZED_WIRE_2;
wire  SYNTHESIZED_WIRE_3;
wire  SYNTHESIZED_WIRE_4;
wire  SYNTHESIZED_WIRE_5;
wire  SYNTHESIZED_WIRE_6;
wire  SYNTHESIZED_WIRE_7;
wire  SYNTHESIZED_WIRE_8;
wire  SYNTHESIZED_WIRE_9;
wire  SYNTHESIZED_WIRE_10;
wire  SYNTHESIZED_WIRE_11;
wire  SYNTHESIZED_WIRE_12;

assign      nx[3] = ~(x[3] | x[3]);
assign      nx[2] = ~(x[2] | x[2]);
assign      nx[1] = ~(x[1] | x[1]);
assign      nx[0] = ~(x[0] | x[0]);
assign      a = ~(nx[2] | x[3]);
assign      SYNTHESIZED_WIRE_0 = ~(nx[3] | x[2] | nx[0]);
assign      SYNTHESIZED_WIRE_1 = ~(nx[3] | nx[1] | nx[2] | x[0]);
assign      y[1] = ~(a | SYNTHESIZED_WIRE_0 | SYNTHESIZED_WIRE_1);
assign      SYNTHESIZED_WIRE_2 = ~(x[3] | x[2] | nx[0]);
assign      SYNTHESIZED_WIRE_4 = ~(nx[3] | nx[2] | nx[0]);
assign      SYNTHESIZED_WIRE_3 = ~(nx[2] | nx[1] | x[0]);
assign      SYNTHESIZED_WIRE_5 = ~(nx[3] | nx[1] | x[0]);
assign      SYNTHESIZED_WIRE_7 = ~(nx[3] | x[2] | x[0]);
assign      SYNTHESIZED_WIRE_6 = ~(x[3] | nx[2] | x[0]);
assign      y[2]    =    ~(SYNTHESIZED_WIRE_2    |    SYNTHESIZED_WIRE_3    |
SYNTHESIZED_WIRE_4     |     SYNTHESIZED_WIRE_5     |     SYNTHESIZED_WIRE_6     |
SYNTHESIZED_WIRE_7);
assign      SYNTHESIZED_WIRE_8 = ~(nx[0] | x[3]);
assign      SYNTHESIZED_WIRE_9 = ~(nx[3] | nx[1] | x[2] | x[0]);
assign      y[3] = ~(a | SYNTHESIZED_WIRE_8 | SYNTHESIZED_WIRE_9);
assign      SYNTHESIZED_WIRE_10 = ~(nx[0] | nx[3]);
```

```
	assign	SYNTHESIZED_WIRE_11 = ~(x[2] | nx[1] | x[0]);
	assign	SYNTHESIZED_WIRE_12 = ~(x[3] | x[2] | x[0]);
	assign	y[4]	=	~(SYNTHESIZED_WIRE_10	|	SYNTHESIZED_WIRE_11	|
SYNTHESIZED_WIRE_12);
	endmodule
```

Якщо всі описані раніше етапи було виконано успішно, відбудеться запуск Modelsim, застосування “плану тестування” до описаного пристрою і побудова результуючих часових діаграм. Вікно із результатами моделювання має вигляд, зображений на рисунку 7.2.

Аналогічно, як і САПР Quartus Prime, Modelsim містить вікно “Transcript”, що являє собою консоль, в якій додатково дублюються всі стани моделювання цифрового пристрою (значення вхідних і вихідних сигналів під час моделювання), а у випадку помилок – зміст самих помилок.

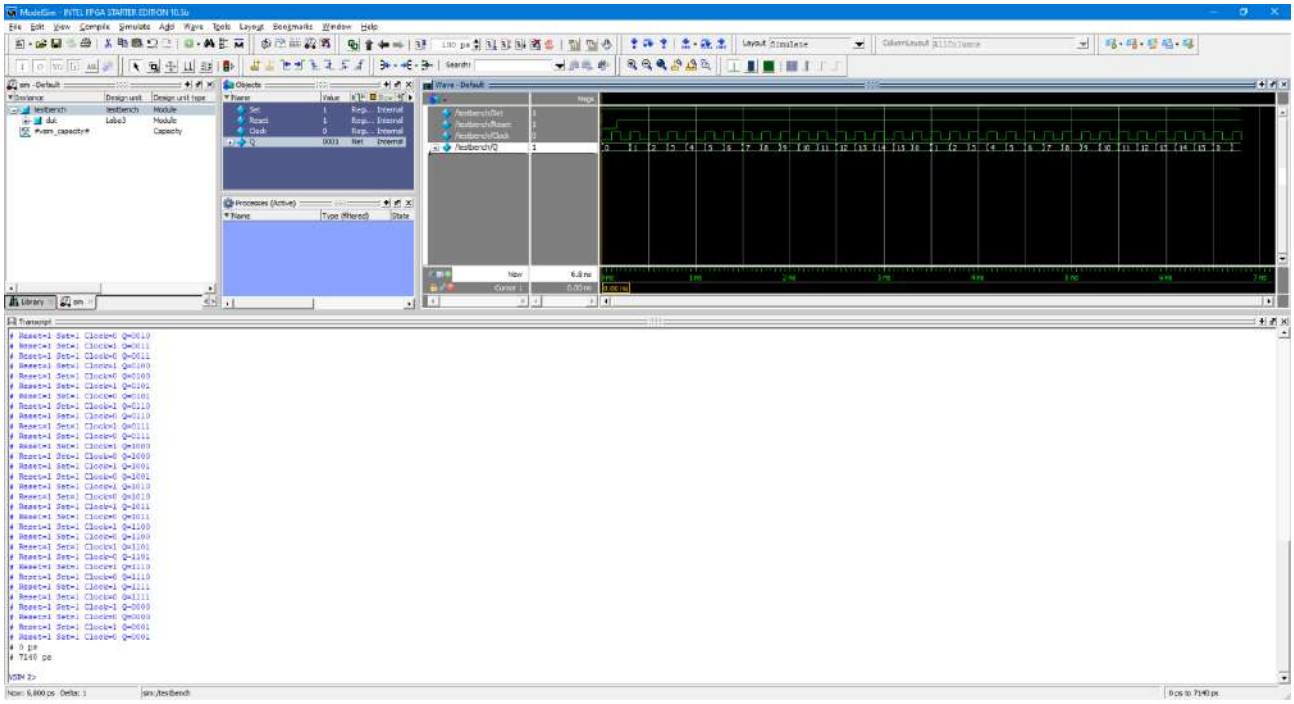


Рисунок 7.2 – Вікно середовища Modelsim із результатами моделювання

7.3 Особливості роботи із Modelsim

В процесі моделювання і аналізу побудованих часових діаграм, варто враховувати наступні зауваження і рекомендації:

1. Як зазначалося вище, файл із описом створеного цифрового пристрою (“Laba2.v”) автоматично генерується засобами САПР Quartus Prime, тому внесення будь-яких змін або виправлень в нього не має ніякого сенсу, оскільки вони будуть втрачені після наступної регенерації.

2. У випадку, якщо отримані часові діаграми демонструють непрацездатність цифрового пристрою, або невідповідність його заданим вимогам, необхідно вносити зміни у його структурну схему (або інші файли проекту). У випадку успішної повторної компіляції проекту, файл із описом мовою Verilog змінюватися не буде! Дану процедуру необхідно виконувати вручну після кожної модифікації проекту, інакше Modelsim буде

використовувати застарілий опис, тобто, помилки на часових діаграмах так і залишаться.

3. При написанні/редагуванні файлу “testbench.v” слід звертати особливу увагу на рядок, де відбувається виклик модуля, який буде тестуватися:

```
Laba2 dut (x, y);
```

Назва модуля має в точності співпадати із назвою модуля верхнього рівня ієрархії проекту САПР Quartus Prime, а передача параметрів повинна відповідати їх типу, розмірності і порядку декларування в файлі-описі (“Laba2.v”).

4. При відображенні часових діаграм функціонування пристрою або системи, часто буває незручно відображати векторний багаторозрядний сигнал у вигляді одиночних бітів (наприклад, якщо такий сигнал подається на вхід суматора багаторозрядних чисел або кодує поточний стан лічильника). Саме для таких випадків Modelsim дозволяє задавати спосіб інтерпретації набору сигналів.

Для того, щоб змінити відображення сигналу необхідно виділити його на часовій діаграмі, натиснувши ліву клавішу миші, потім натиснути праву кнопку миші, і у контекстному меню, яке випаде, вибрати пункт Radix. Даний пункт міститиме у собі перелік усіх доступних способів інтерпертації значень сигналів, зокрема:

– Binary – у вигляді двійкового числа;
– Octal – у вигляді числа восьмирічної системи числення;
– Decimal – у вигляді десяткового числа;
– Hexadecimal – у вигляді числа вшістнадцяткової системи числення;
– Unsigned – беззнакове ціле десяткове число;
– ASCII – символ згідно вказаної кодової таблиці.

8 РЕКОМЕНДАЦІЇ І ВИМОГИ ЩОДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ І ОФОРМЛЕННЯ ЗВІТІВ

8.1 Загальні методичні рекомендації щодо проектування

Вивчення дисципліни “Комп’ютерна логіка та основи схемотехніки” ефективне лише тоді, коли поряд із оволодінням теорії здобувачі вищої освіти в умовах проведення лабораторного практикуму набувають навичок синтезу цифрових пристроїв різного призначення за допомогою САПР, що застосовуються в промисловості.

Запропоновані методичні вказівки присвячено вивченню методів проектування електронних пристроїв за допомогою САПР Quartus Prime, який відрізняється від інших відомих САПР:

– простотою та зручністю у користуванні, особливо при графічному способі введення схеми;

– доступністю для користувача, мало знайомого з обчислювальною технікою;

– наявністю основних видів аналізу електронних пристроїв;

– розвиненою бібліотекою компонентів.

Під час виконання лабораторних робіт здобувачі вищої освіти мають:

– освоїти методики синтезу електронних пристроїв;

– отримати знання про методи проектування електронних пристроїв на персональних ЕОМ;

– глибше засвоїти основи теорії електронних пристроїв та методи їх синтезу.

Підготовка до лабораторної роботи передбачає обов’язкове вивчення студентами теоретичного матеріалу.

Лабораторну роботу рекомендується розпочинати з опрацювання прикладів схем цифрових пристроїв, наведених у кожній лабораторній роботі. Після цього слід переходити до дослідження:

– синтезувати цифровий електронний пристрій з урахуванням встановленого варіанта завдання;

– задати параметри та межі моделювання цифрового пристрою.

Звіт до кожної лабораторної роботи повинен містити:

– назву лабораторної роботи;

– мету роботи;

– короткі теоретичні відомості;

– порядок виконання роботи;

– результати дослідження та аналізу параметрів та характеристик досліджуваного пристрою;

– особливості функціонування САПР Quartus Prime, виявлені під час виконання лабораторної роботи;

– висновки.

8.2 Вимоги до оформлення звіту про виконання лабораторної роботи

Викладені нижче вимоги щодо оформлення звіту про виконання будь-якої лабораторної роботи є обов'язковими до виконання.

1. Для оформлення звіту використовуються стандартні листи формату А4 (210х297 мм), що мають наступні поля: ліве – не менше 25 мм; верхнє – не менше 15 мм; нижнє – не менше 15 мм; праве – не менше 10 мм.

2. Текст звіту (включаючи заголовки, назви розділів пунктів і підпунктів, підписи рисунків і таблиць) оформлюється шрифтом Times New Roman, 14 pt, одинарний міжрядковий інтервал без додаткових виступів до або після. У випадку представлення у звіті коду програми (будь-якою мовою), коду скрипта, тощо, використовується шрифт Courier New, 10 pt, всі інші вимоги – аналогічні.

3. Порядковий номер лабораторної роботи, її тема, назви всіх розділів верхнього рівня, підписи рисунків, тіла рисунків і таблиць вирівнюються по центру, без абзацного відступа. Виключення складають лише підписи таблиць, оскільки вони вирівнюються по ширині із абзацного відступу.

4. Підпис рисунка і його тіло є нерозривними, тому категорично забороняється розривати їх (переносити на різні сторінки) або розділяти будь-якими пропусками.

5. При переносі таблиці на іншу сторінку вгорі обов'язково ставиться напис "Продовження таблиці Х.У", а також дублюється її шапка.

6. Нумерація таблиць, рисунків і формул виконується наскрізно в межах кожного розділу звіту і складається із двох цифр: перша – номер розділу, до якого відноситься елемент, друга – порядковий номер елемента в межах розділу. Наприклад, напис "Таблиця 2.4" означає, що ця таблиця відноситься до розділу 2 і є четвертою по порядку слідування в межах розділу.

7. При нумерації пунктів і підпунктів розділу категорично не рекомендується використовувати рівень вкладеності глибший, ніж 3.

8. Після тексту перед назвою розділу/підрозділу, тілом рисунку або підписом таблиці, а також після назви розділу/підрозділу, підпису рисунка або тіла таблиці перед наступним текстом робиться пропуск в 1 рядок. Категорично забороняється робити пропуски рядків у суцільному тексті.

9. Рисунки і таблиці, що мають різний порядковий номер, не можуть мати ідентичні пояснюючі підписи. Забороняється залишати тільки нумерацію без пояснюючих підписів.

10. Часові діаграми із результатами моделювання роботи синтезованих цифрових пристроїв повинні обов'язково містити вісь часу.

8.3 Вимоги до змісту звіту про виконання лабораторної роботи

Наполегливо рекомендується дотримуватися описаного нижче порядку нумерації та іменування розділів звіту при виконанні кожної лабораторної роботи.

Звіт про виконання лабораторної роботи повинен відповідати вимогам академічної доброчесності і містити наступні обов'язкові елементи і розділи:

1. Порядковий номер лабораторної роботи. Пишеться вгорі першої сторінки, по центру, без абзацних відступів великими літерами напівжирним шрифтом.

2. Тема лабораторної роботи. Пишеться вгорі першої сторінки, по центру, без абзацних відступів великими літерами напівжирним шрифтом без крапки в кінці.

3. Мета роботи. Виділяється напівжирним шрифтом із двокрапкою, після чого пишеться звичайним шрифтом, починаючи із абзацного відступу, текст вирівнюється по ширині.

4. Короткі теоретичні відомості. Це перший по порядку розділ звіту, його загальний обсяг складає: мінімально одну повну сторінку, максимально – дві повні сторінки формату А4. Матеріал, який тут наводиться, може бути запозичено у методичних вказівках, презентаціях/конспектах лекцій, книжках, журналах, інтернет-ресурсах (окрім Wikipedia), тощо. Запозичений матеріал повинен прямо або опосередковано стосуватися теми лабораторної роботи або інструментальних засобів, які будуть використовуватися, і має бути зрозумілим автору звіту.

5. Хід виконання роботи. Обов’язково повинен містити схеми синтезованих цифрових пристроїв згідно варіанту завдання, часові діаграми із результатами моделювання їх роботи, а також текстові описи, пояснення чи розрахунки, якщо це є у вимогах до конкретної лабораторної роботи.

6. Особливості функціонування САПР Quartus Prime, виявлені в ході виконання лабораторної роботи. Містить текстовий опис спостережень за особливостями, пов’язаними із функціонуванням або застосуванням САПР при виконанні лабораторної роботи. Даний розділ повинен містити виключно власні спостереження виконавця (здобувача вищої освіти). Категорично забороняється наводити загальні відомості стосовно функціональних можливостей САПР, які, наприклад, можуть міститися в інструкціях користувача, тощо. Якщо жодних особливостей виявлено не було, це також треба відмітити.

7. Висновки. Повинні містити не тільки відповіді на пункти мети виконання лабораторної роботи, але й відмітки про виконання чи невиконання поставлених завдань із зазначенням причин неможливості досягнути поставленої мети, описом виявлених особливостей синтезу схеми пристрою, або застосування САПР та ін.

9 ЛАБОРАТОРНА РОБОТА №1
ВИВЧЕННЯ МОЖЛИВОСТЕЙ САПР INTEL QUARTUS PRIME ДЛЯ СИНТЕЗУ ТА МОДЕЛЮВАННЯ ЦИФРОВИХ ПРИСТРОЇВ

Мета роботи: ознайомитися із функціональними можливостями і інструментальними засобами САПР Intel Quartus Prime, навчитися створювати проекти та синтезувати прості цифрові пристрої за допомогою САПР.

9.1 Порядок виконання роботи

9.1.1 Запустити САПР Quartus Prime.

9.1.2 Вивчити основні можливості САПР Quartus Prime.

9.1.3 Дослідити призначення основних команд, що відносяться до створення електричних принципових схем цифрових пристроїв та аналізу їх роботи.

9.1.4 Синтезувати схему простого цифрового пристрою з урахуванням встановленого варіанта завдання згідно таблиці 9.1.

Таблиця 9.1 – Варіанти завдання 1

№ варіанту завдання	**Найменування логічного елемента**	
1	and4	or4
2	and4	or3
3	and4	or2
4	and3	or4
5	and3	or3
6	and3	or2
7	and2	or4
8	and2	or3
9	and2	or2
10	and4	not
11	or4	not
12	and3	not
13	or3	not
14	and2	not
15	or2	not
Примітки. 1. Номер варіанту завдання обирається у відповідності до порядкового номера студента згідно списку у журналі академічної групи. 2. Якщо порядковий номер студента перевищує 15, то від нього віднімається число 15, а отримана різниця – і буде номером варіанта.		

9.1.5 В якості прикладу, на рисунку 9.1 наведено синтезовану схему простого цифрового пристрою.

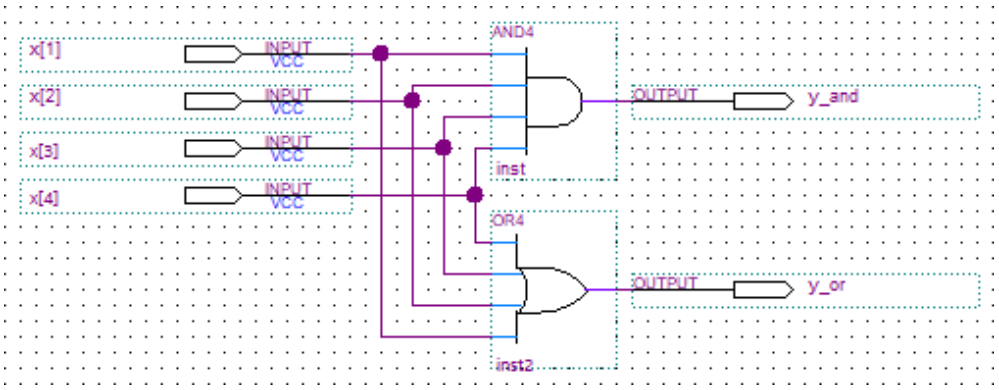


Рисунок 9.1 – Результат синтезу простого цифрового пристрою

9.1.6 Дослідити часову діаграму роботи синтезованого цифрового пристрою при всіх можливих наборах логічних змінних за допомогою пакету ModelSim.

На рисунку 9.2, в якості прикладу, наведено результат моделювання роботи простого цифрового пристрою, схему якого наведено на рисунку 9.1.

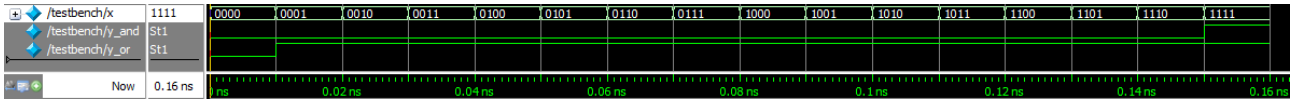


Рисунок 9.2 – Часова діаграма функціонування синтезованого цифрового пристрою

9.1.7 На основі отриманої часової діаграми зробити висновки стосовно того, яку логічну операцію виконує кожен із компонентів у складі синтезованого цифрового пристрою.

9.1.8 Синтезувати комбінаційну схему логічної функції з урахуванням встановленого варіанта завдання згідно таблиці 9.2 за допомогою САПР Quartus Prime.

Таблиця 9.2 – Варіанти завдання 2

№ варіанту завдання	**Аналітичний вираз логічної функції $y=f(x_1, x_2, x_3, x_4)$**
1	$y= x_1 \vee x_2 \vee x_3 \vee x_4 \wedge (\sim x_1 \vee \sim x_2)$
2	$y= x_1 \vee x_2 \vee x_3 \wedge x_4 \wedge (\sim x_1 \vee \sim x_3)$
3	$y= x_1 \vee x_2 \wedge x_3 \wedge x_4 \wedge (\sim x_1 \vee \sim x_4)$
4	$y= x_1 \wedge x_2 \wedge x_3 \wedge x_4 \vee (\sim x_1 \wedge x_2)$
5	$y= x_1 \wedge x_2 \wedge x_3 \vee x_4 \vee (\sim x_1 \wedge x_3)$
6	$y= x_1 \wedge x_2 \vee x_3 \vee x_4 \vee (\sim x_1 \wedge x_4)$
7	$y= x_1 \vee x_2 \wedge x_3 \vee x_4 \wedge (x_1 \vee \sim x_2)$
8	$y= x_1 \wedge x_2 \vee x_3 \wedge x_4 \vee (x_1 \wedge \sim x_3)$
9	$y= x_1 \vee x_2 \vee x_3 \vee x_4 \wedge (x_1 \wedge \sim x_4)$
10	$y= x_1 \wedge x_2 \wedge x_3 \wedge x_4 \vee (x_1 \vee x_2)$
11	$y= x_1 \wedge x_2 \vee x_3 \wedge x_4 \vee (x_1 \wedge x_3)$
12	$y= \sim x_1 \vee \sim x_2 \vee \sim x_3 \vee \sim x_4 \wedge (x_1 \vee x_2)$

Продовження таблиці 9.2

№ варіанту завдання	**Аналітичний вираз логічної функції $y=f(x_1, x_2, x_3, x_4)$**
13	$y= \sim x_1 \vee \sim x_2 \vee \sim x_3 \wedge \sim x_4 \wedge (x_1 \vee x_3)$
14	$y= \sim x_1 \vee \sim x_2 \wedge \sim x_3 \wedge \sim x_4 \wedge (x_1 \vee x_4)$
15	$y= \sim x_1 \wedge \sim x_2 \wedge \sim x_3 \wedge x_4 \vee (x_1 \wedge x_2)$
16	$y= \sim x_1 \wedge \sim x_2 \wedge \sim x_3 \vee x_4 \vee (x_1 \wedge x_3)$
17	$y= \sim x_1 \wedge \sim x_2 \vee \sim x_3 \vee x_4 \vee (x_1 \wedge x_4)$
18	$y= \sim x_1 \vee \sim x_2 \wedge x_3 \vee x_4 \wedge (x_1 \vee \sim x_2)$
19	$y= \sim x_1 \wedge \sim x_2 \vee x_3 \wedge x_4 \vee (x_1 \wedge \sim x_3)$
20	$y= \sim x_1 \vee \sim x_2 \vee x_3 \vee x_4 \wedge (x_1 \wedge \sim x_4)$
21	$y= \sim x_1 \wedge x_2 \wedge x_3 \wedge x_4 \vee (x_1 \vee x_2)$
22	$y= \sim x_1 \wedge x_2 \vee x_3 \wedge x_4 \vee (x_1 \wedge x_3)$
23	$y= \sim x_1 \vee x_2 \wedge x_3 \vee x_4 \vee (x_1 \vee x_4)$
24	$y= x_1 \vee \sim x_2 \vee x_3 \vee \sim x_4 \wedge \sim (x_1 \vee x_2)$
25	$y= x_1 \vee \sim x_2 \vee x_3 \wedge \sim x_4 \wedge \sim (x_1 \vee x_3)$
26	$y= x_1 \vee \sim x_2 \wedge x_3 \wedge \sim x_4 \wedge \sim (x_1 \vee x_4)$
27	$y= \sim x_1 \vee x_2 \vee \sim x_3 \vee x_4 \wedge \sim (x_1 \vee x_2)$
28	$y= \sim x_1 \vee x_2 \vee \sim x_3 \wedge x_4 \wedge \sim (x_1 \vee x_3)$
29	$y= \sim x_1 \vee x_2 \wedge \sim x_3 \wedge x_4 \wedge \sim (x_1 \vee x_4)$
30	$y= \sim x_1 \vee \sim x_2 \vee \sim x_3 \vee \sim x_4 \wedge x_1 \wedge x_2 \wedge x_3 \wedge x_4$
Примітки. 1. Номер варіанту завдання обирається у відповідності до порядкового номера студента згідно списку академічної групи. 2. В таблиці застосовано наступні позначення логічних операцій: “$\vee$” – диз’юнкція, “$\wedge$” – кон’юнкція, “$\sim$” – заперечення.	

9.1.9 Дослідити часову діаграму роботи синтезованої комбінаційної схеми при всіх можливих наборах логічних змінних за допомогою пакету ModelSim. Після аналізу отриманої часової діаграми, заповнити правий стовпчик таблиці істинності булевої функції у вигляді, наведеному в таблиці 9.3.

Таблиця 9.3 – Таблиця істинності булевої функції

x_4	x_3	x_2	x_1	y
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	

Продовження таблиці 9.3

x_4	x_3	x_2	x_1	y
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

9.2 Вимоги до змісту звіту

Звіт про виконання лабораторної роботи повинен містити:

1. Номер і тему роботи.
2. Мету виконання роботи.
3. Короткі теоретичні відомості.
4. Порядок виконання роботи.
5. Результати виконання роботи у вигляді:

– схеми електричної принципової, яку було отримано в результаті виконання завдання 1;

– часової діаграми, яку було отримано в результаті виконання завдання 1;

– схеми електричної принципової, яку було отримано в результаті виконання завдання 2;

– часової діаграми, яку було отримано в результаті виконання завдання 2;

– результати виконання пунктів 9.1.7 та 9.1.9 порядку виконання роботи.

6. Особливості функціонування САПР Intel Quartus Prime, виявлені під час виконання роботи.
7. Висновки.

9.3 Контрольні питання для перевірки знань

9.3.1 Наведіть таблицю істинності кон'юнктора, що має 3 входи.

9.3.2 Наведіть таблицю істинності диз'юнктора, що має 3 входи.

9.3.3 Наведіть таблицю істиності інвертора.

9.3.4 Перелічіть основні логічні операції (кон'юнкція, диз'юнкція, інверсія) в порядку зростання пріоритету їх виконання.

9.3.5 Поясніть, чому в бібліотеках САПР Intel Quartus Prime не можна знайти кон'юнктори і диз'юнктори із кількістю входів більшою, ніж 12, хоча зазначені операції не мають обмежень на максимальну кількість логічних змінних, що поєднуються?

9.3.6 Поясніть, які переваги і недоліки може мати заміна одного кон'юнктора із більшою кількістю входів декількома кон'юнкторами із меншою кількістю (каскадування)?

9.3.7 На який логічний елемент перетвориться кон'юнктор або диз'юнктор, якщо на всі його входи подать один і той же сигнал?

10 ЛАБОРАТОРНА РОБОТА № 2
РЕАЛІЗАЦІЯ БУЛЕВИХ ФУНКЦІЙ НА ЛОГІЧНИХ ЕЛЕМЕНТАХ І–НЕ, АБО–НЕ, ВИКЛЮЧНЕ АБО

Мета роботи: вивчити методи проектування комбінаційних схем на логічних елементах типу І–НЕ, АБО–НЕ, ВИКЛЮЧНЕ АБО.

10.1 Короткі теоретичні відомості

Проектування будь-якої комбінаційної схеми завжди складається із трьох етапів.

На першому етапі, виходячи із таблиці істинності, яка описує роботу комбінаційної схеми, що синтезується, знаходять мінімальну диз'юнктивну нормальну форму (МДНФ) та/або мінімальну кон'юнктивну нормальну форму (МКНФ) функції.

Якщо функція, яка описує роботу схеми, що синтезується, не є повністю визначеною (тобто, її значення існують не для всіх можливих наборів аргументів), то здійснюють її доповнення таким чином, щоб функція мала більш просту МДНФ (або МКНФ). На цьому перший етап закінчується.

На другому етапі функцію записують у так званій операторної формі, тобто у вигляді суперпозиції операторів логічних елементів. Оператором логічного елемента називають функцію, яка реалізується цим елементом. Таких форм є вісім. Якщо кількість входів логічних елементів є обмеженою, то для представлення функції в операторній формі використовують наступні співвідношення:

$$x_1 \wedge x_2 \wedge \ldots \wedge x_n = (x_1 \wedge \ldots \wedge x_p) \wedge (x_{p+1} \wedge \ldots \wedge x_{2p}) \wedge \ldots \wedge (x_{mp+1} \wedge \ldots \wedge x_n) \quad (10.1)$$

$$x_1 \vee x_2 \vee \ldots \vee x_n = (x_1 \vee \ldots \vee x_p) \vee (x_{p+1} \vee \ldots \vee x_{2p}) \vee \ldots \vee (x_{mp+1} \vee \ldots \vee x_n) \quad (10.2)$$

$$\overline{x_1 \wedge x_2 \wedge \ldots \wedge x_n} = \overline{\overline{(x_1 \wedge \ldots \wedge x_p)} \wedge \overline{(x_{p+1} \wedge \ldots \wedge x_{2p})} \wedge \ldots \wedge \overline{(x_{mp+1} \wedge \ldots \wedge x_n)}} \quad (10.3)$$

$$\overline{x_1 \vee x_2 \vee \ldots \vee x_n} = \overline{\overline{(x_1 \vee \ldots \vee x_p)} \vee \overline{(x_{p+1} \vee \ldots \vee x_{2p})} \vee \ldots \vee \overline{(x_{mp+1} \vee \ldots \vee x_n)}} \quad (10.4)$$

На заключному (третьому) етапі на основі отриманих операторних представлень функцій складається шукана комбінаційна схема.

Задачу синтезу комбінаційної схеми із декількома виходами може бути зведено до синтезу комбінаційної схеми з одним виходом, якщо функцію кожного виходу синтезувати окремо. В цьому випадку задана система, що складається із m функцій, буде реалізована у вигляді m не зв'язаних між собою комбінаційних схем, однак це може призводити до дублювання логічних елементів.

Існуючі методи синтезу комбінаційних схем з декількома входами базуються на ідеї використання однієї функції (або її частини) для отримання іншої функції. При невеликому значенні m вельми наочним є метод мінімізації, заснований на використанні карт Карно. Для кожної із булевих функцій карти будуються окремо, а потім (шляхом порівняння карт) відзначаються

однойменні набори, на яких всі або декілька функцій приймають одиничні значення.

10.2 Порядок виконання роботи

10.2.1 З метою визначення свого варіанту булевої функції перевести число і місяць свого народження, а також порядкового номера згідно списку у журналі академічної групи у двійковий код і записати у вигляді слова ($a_{13} – a_0$):

a_{13} a_{12} a_{11} a_{10} a_9	a_8 a_7 a_6 a_5	a_4 a_3 a_2 a_1 a_0
число народження	місяць народження	порядковий номер

Наприклад, якщо студент народився 28.05 і має порядковий номер 15 згідно списку, то слово буде мати наступний вигляд:

$$28_{10} = 11100_2;$$
$$05_{10} = 0101_2;$$
$$15_{10} = 01111_2;$$

1	1	1	0	0	0	1	0	1	0	1	1	1	1
a_{13}	a_{12}	a_{11}	a_{10}	a_9	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

10.2.2 Для заданої логічної функції згідно таблиці 10.1 та з урахуванням підстановки значень розрядів (a_i) отриманого слова, виконати та письмово представити наступні результати.

10.2.3 Знайти за допомогою карт Карно МДНФ логічної функції. Перетворити знайдену МДНФ за допомогою законів “заперечення заперечення” і “де Моргана” таким чином, щоб її було представлено у базисі І-НЕ.

10.2.4 Побудувати комбінаційну схему, що реалізує задану функцію на основі багатовходових елементів І–НЕ.

10.2.5 Знайти за допомогою карт Карно МКНФ логічної функції. Перетворити знайдену МКНФ за допомогою законів “заперечення заперечення” і “де Моргана” таким чином, щоб її було представлено у базисі АБО-НЕ.

10.2.6 Побудувати комбінаційну схему, що реалізує задану функцію на основі багатовходових елементів АБО–НЕ.

10.2.7 Для кожної із отриманих схем порахувати та вказати параметри:

1) N – кількість елементів схеми (рахуються тільки ті, що виконують логічні операції);

2) L – рівень схеми (максимальне число послідовно з’єднаних елементів).

10.2.8 Аналогічно до того, як викладено в 10.2.3 – 10.2.7, побудувати на логічних елементах І–НЕ та АБО–НЕ схеми перетворювача кодів згідно таблиці 10.2. Визначити для кожної із отриманих схем значення параметрів N і L.

УВАГА! Під час створення схем електричних принципових слід використовувати логічні елементи І-НЕ (nand) та АБО-НЕ (nor), що містяться виключно в бібліотеці primitives/logic. Використання аналогічних елементів бібліотеки others/maxplus2, категорично заборонено, оскільки призведе до

проблем при отриманні часових діаграм. Якщо кількість входів обраного логічного елемента (бібліотеки primitives/logic) перевищує кількість змінних, які він повинен поєднати, можна: зробити дублювання існуючого сигналу, тобто, подавати однаковий сигнал на кілька входів логічного елемента; подати фіксований логічний рівень на всі "зайві" входи, але так, щоб це не змінювало результат операції (високий – на кон'юнктор, низький – на диз'юнктор).

Таблиця 10.1 – Варіант завдання для логічної функції чотирьох змінних

x_3	x_2	x_1	x_0	y
0	0	0	0	a_{13}
0	0	0	1	a_{12}
0	0	1	0	a_{11}
0	0	1	1	a_{10}
0	1	0	0	a_9
0	1	0	1	a_8
0	1	1	0	a_7
0	1	1	1	1
1	0	0	0	a_6
1	0	0	1	a_5
1	0	1	0	a_4
1	0	1	1	a_3
1	1	0	0	a_2
1	1	0	1	a_1
1	1	1	0	a_0
1	1	1	1	0

Таблиця 10.2 – Варіант завдання для системи логічних функцій

x_3	x_2	x_1	x_0	y_1	y_2	y_3	y_4
0	0	0	0	1	1	1	0
0	0	0	1	1	0	0	1
0	0	1	0	1	1	1	0
0	0	1	1	1	0	0	1
0	1	0	0	0	0	0	1
0	1	0	1	0	1	0	1
0	1	1	0	0	0	0	1
0	1	1	1	0	1	0	1
1	0	0	0	a_{13}	a_9	a_5	a_1
1	0	0	1	0	1	1	0
1	0	1	0	a_{12}	a_8	a_4	0
1	0	1	1	0	1	1	0
1	1	0	0	a_{11}	a_7	a_3	1
1	1	0	1	1	0	1	0
1	1	1	0	a_{10}	a_6	a_2	a_0
1	1	1	1	1	0	1	0

10.2.9 Синтезувати комбінаційну схему цифрового пристрою в базисі Жегалкіна {використовується 3 логічні функції: ∧, ⊕, 1} згідно варіанту, наведеного в таблиці 10.3. Для цього логічну функцію, задану таблицею істинності після підстановки значень a_i, представити поліномом Жегалкіна. Отримати поліном Жегалкіна двома способами: використовуючи канонічне правило переходу із Булевого базису, а також методом трикутника Паскаля.

10.2.10 Визначити для отриманої схеми значення параметрів N і L.

Таблиця 10.3 – Завдання для синтезу логічної функції в базисі Жегалкіна

x_2	x_1	x_0	y
0	0	0	a_7
0	0	1	a_6
0	1	0	a_5
0	1	1	a_4
1	0	0	a_3
1	0	1	a_2
1	1	0	a_1
1	1	1	a_0

10.3 Приклад реалізації системи чотирьох логічних функцій на елементах І-НЕ, АБО-НЕ

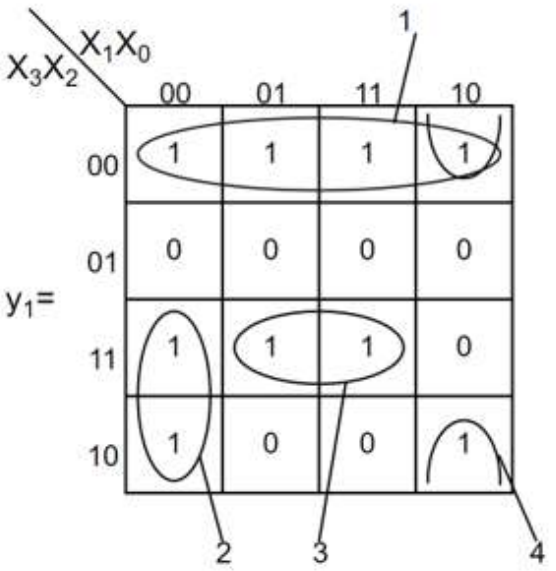
Для розгляді процесу синтезу системи логічних функцій (таблиця 10.2) візьмемо поліном, сформований у пункті 10.2.1. В результаті підстановки усіх a_i, було отримано таблицю істинності системи функцій (таблиця 10.4).

Таблиця 10.4 – Таблиця істинності системи із чотирьох булевих функцій

x_3	**x_2**	**x_1**	**x_0**	**y_1**	**y_2**	**y_3**	**y_4**
0	0	0	0	1	1	1	0
0	0	0	1	1	0	0	1
0	0	1	0	1	1	1	0
0	0	1	1	1	0	0	1
0	1	0	0	0	0	0	1
0	1	0	1	0	1	0	1
0	1	1	0	0	0	0	1
0	1	1	1	0	1	0	1
1	0	0	0	1 (a_{13})	0 (a_9)	1 (a_5)	1 (a_1)
1	0	0	1	0	1	1	0
1	0	1	0	1 (a_{12})	0 (a_8)	0 (a_4)	0
1	0	1	1	0	1	1	0
1	1	0	0	1 (a_{11})	1 (a_7)	1 (a_3)	1
1	1	0	1	1	0	1	0
1	1	1	0	0 (a_{10})	0 (a_6)	1 (a_2)	1 (a_0)
1	1	1	1	1	0	1	0

На рисунку 10.1 зображено процес та результат отримання МДНФ булевої логічної функції y_1 згідно таблиці 10.4. На рисунку 10.2 представлено результат створення комбінаційної схеми цифрового пристрою, реалізованої в базисі І–НЕ, засобами САПР Quartus Prime. На рисунку 10.3 – часові діаграми функціонування отриманого комбінаційного цифрового пристрою.

1. Отримання МДНФ для функції y_1:



$$y_1 = \underbrace{\overline{X_3}\overline{X_2}}_{1} \cup \underbrace{X_2\overline{X_1}\overline{X_0}}_{2} \cup \underbrace{X_3X_2X_0}_{3} \cup \underbrace{\overline{X_2}X_1\overline{X_0}}_{4}$$

Застосуємо закон "заперечення заперечення" до отриманої МДНФ функції y_1, отримаємо:

$$y_1 = \overline{\overline{X_3}\overline{X_2} \cup X_3\overline{X_1}\overline{X_0} \cup X_3X_2X_0 \cup \overline{X_2}X_1\overline{X_0}};$$

Тепер застосуємо закон де Моргана:

$$y_1 = \overline{\overline{X_3}\overline{X_2} \cap \overline{X_3\overline{X_1}\overline{X_0}} \cap \overline{X_3X_2X_0} \cap \overline{\overline{X_2}X_1\overline{X_0}}};$$

Рисунок 10.1 – Отримання МДНФ булевої функції y_1 методом карт Карно

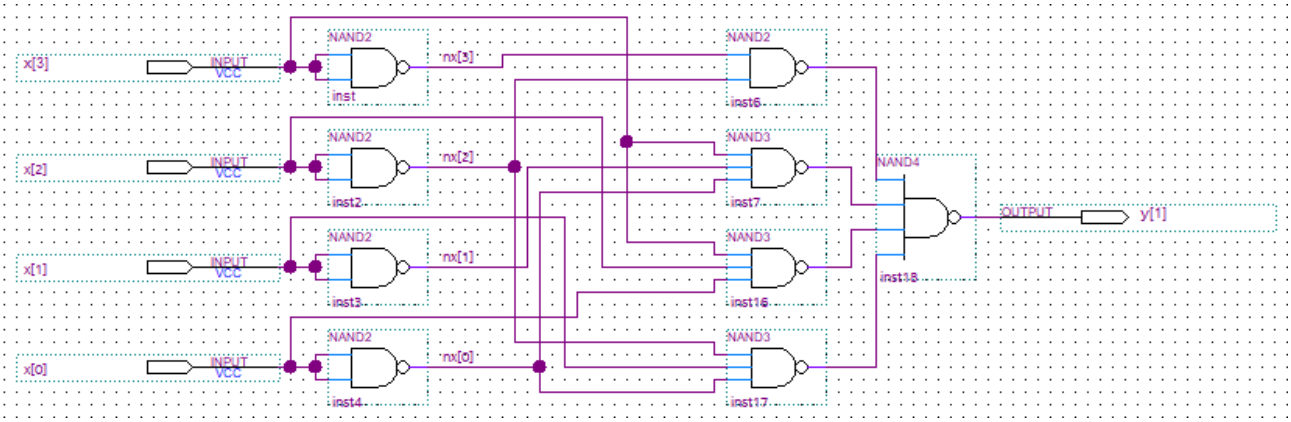


Рисунок 10.2 – Схемотехнічна реалізація булевої функції y_1 в базисі І–НЕ засобами САПР Quartus Prime

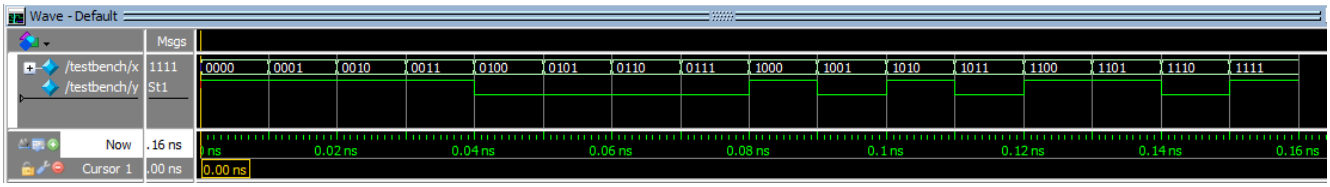


Рисунок 10.3 – Результат симуляції роботи комбінаційної схеми, наведеної на рисунку 2.4, засобами ModelSim

Аналізуючи схему на рисунку 10.2 легко бачити, що кількість елементів схеми (N) становить 9 (із них 4 елементи І-НЕ виконують роль інверторів), а її рівень (L) дорівнює 3.

Аналізуючи рисунок 10.3 легко бачити, що результат роботи схеми, реалізованої в базисі І-НЕ, у відповідності до отриманого опису булевої функції y_1, повністю співпадає із її таблицею істинності (таблиця 10.4).

Процес створення логічної схеми, зображеної на рисунку 10.2, можна істотно спростити та прискорити, якщо використати можливості САПР Quartus Prime утворювати електричні з'єднання компонентів за допомогою однакових назв сигналів. Результат реалізації булевої функції y_1 в базисі І–НЕ за допомогою іменування показано на рисунку 10.4, а часова діаграма її функціонування – на рисунку 10.5.

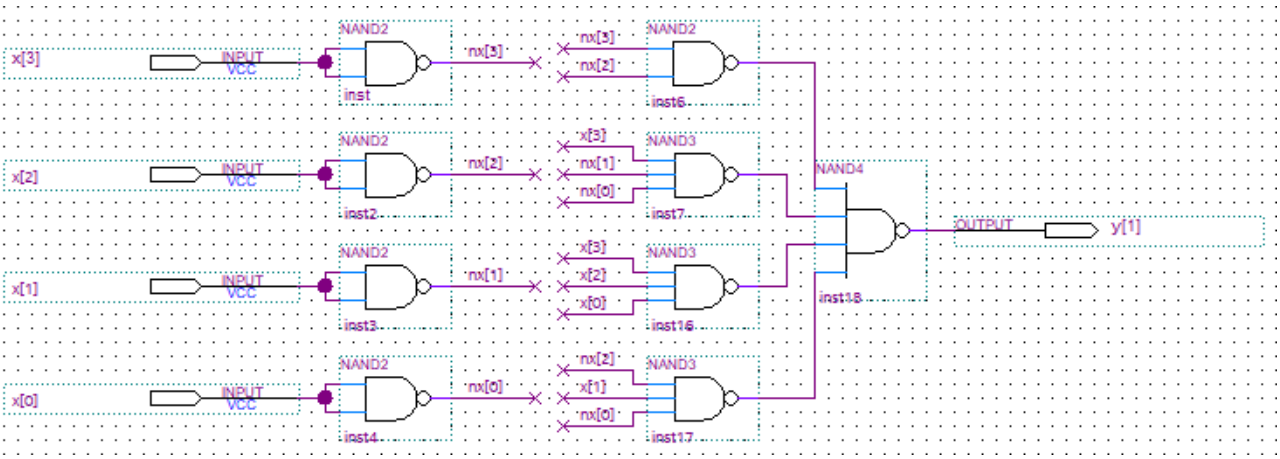


Рисунок 10.4 – Схемотехнічна реалізація булевої функції y_1 в базисі І–НЕ із використанням інструменту іменування

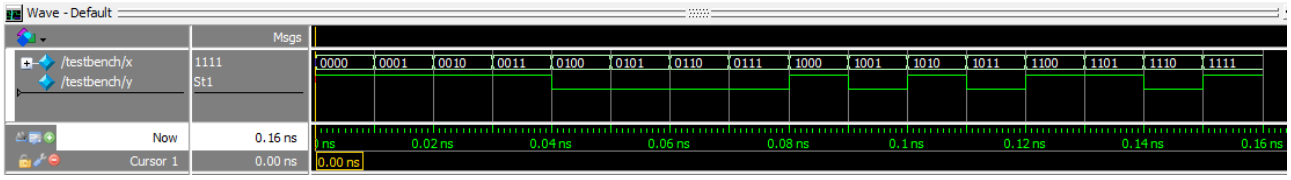


Рисунок 10.5 – Результат симуляції роботи комбінаційної схеми, наведеної на рисунку 10.4, засобами ModelSim

Порівнюючи між собою часові діаграми на рисунках 10.3 та 10.5, можна побачити, що вони ідентичні, тобто, схеми на рисунках 10.3 та 10.5 є тотожними. Однак, схема, зображена на рисунку 10.4, є більш компактною і наочною, в ній істотно знижується ризик утворення небажаних вузлів під час

введення схеми. Тут важливо використовувати унікальні і зрозумілі назви сигналів (наприклад, логічна змінна x[1] після проходження інвертора – елемента nand2 – перетворюється на nx[1]).

Спосіб зв'язування шляхом іменування є зручним і широковживаним, тому усі подальші наведені схеми будуть використовувати саме його.

Нижче, на рисунку 10.6 зображено процес та результат отримання МКНФ булевої логічної функції y_1 у відповідності до таблиці 10.4.

На рисунках 10.7 і 10.8 представлено результат створення комбінаційної схеми цифрового пристрою, реалізованої в базисі АБО–НЕ та часова діаграма його функціонування.

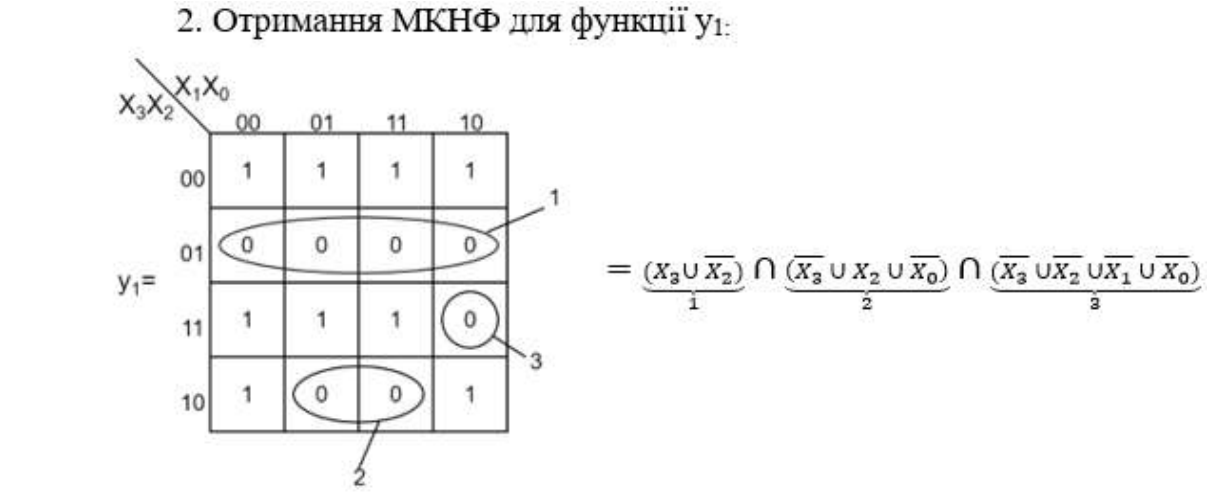


Рисунок 10.6 – Отримання МКНФ булевої функції y_1 методом карт Карно

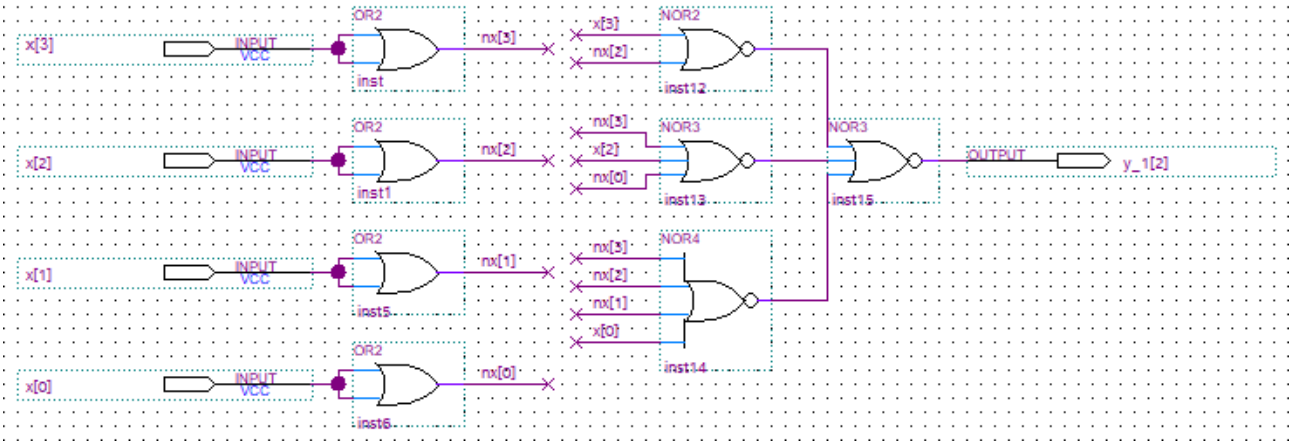


Рисунок 10.7 – Схемотехнічна реалізація булевої функції y_1 в базисі АБО–НЕ із використанням інструменту іменування

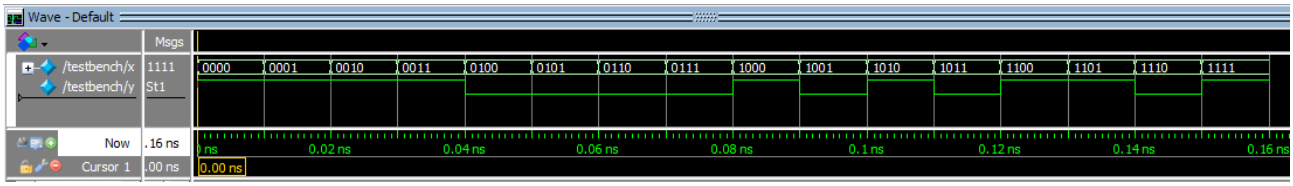
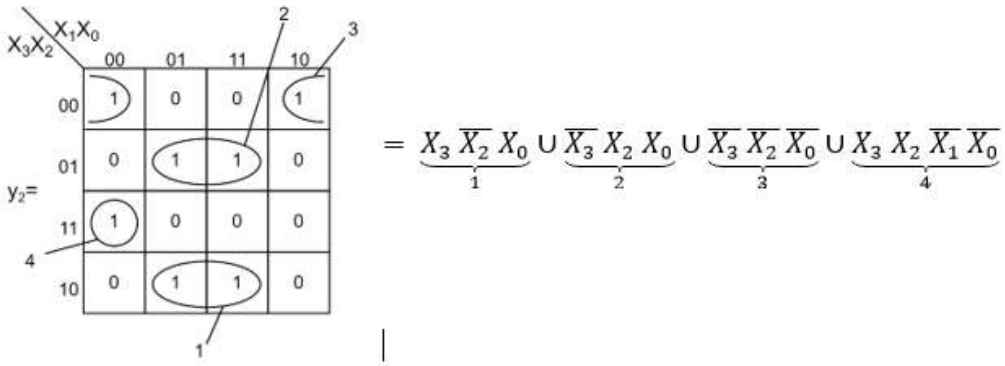


Рисунок 10.8 – Результат симуляції роботи комбінаційної схеми, наведеної на рисунку 10.8, засобами ModelSim

Часові діаграми, зображені на рисунках 10.3 (або 10.5) та 10.8, повинні повністю співпадати між собою, оскільки форма представлення логічної функції жодним чином не впливає на її таблицю істинності (таблиця 10.4).

Нижче, на рисунках 10.9 – 10.14 зображено процес та результат отримання МДНФ та МКНФ булевих логічних функцій y_2, y_3, y_4 у відповідності до їх таблиць істинності (таблиця 2.4), а також аналітичний вигляд функцій, приведених до базисів І–НЕ та АБО–НЕ.

3. Отримання МДНФ для функції y_2:



Застосовуємо закон "заперечення заперечення" до отриманої МДНФ функції y_2, отримаємо:

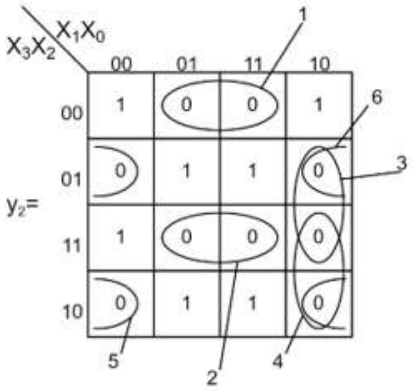
$$y_2 = \overline{X_3\,\overline{X_2}\,X_0 \cup \overline{X_3}\,X_2\,X_0 \cup \overline{X_3}\,\overline{X_2}\,\overline{X_0} \cup X_3\,X_2\,\overline{X_1}\,\overline{X_0}}$$

Тепер застосуємо закон де Моргана:

$$y_2 = \overline{\overline{X_3\,\overline{X_2}\,X_0} \cap \overline{\overline{X_3}\,X_2\,X_0} \cap \overline{\overline{X_3}\,\overline{X_2}\,\overline{X_0}} \cap \overline{X_3\,X_2\,\overline{X_1}\,\overline{X_0}}}$$

Рисунок 10.9 – Отримання МДНФ булевої функції y_2 методом карт Карно

4. Отримання МКНФ для функції y_2:



$$= \underbrace{(X_3 \cup X_2 \cup \overline{X_0})}_{1} \cap \underbrace{(\overline{X_3} \cup \overline{X_2} \cup \overline{X_0})}_{2} \cap \underbrace{(\overline{X_2} \cup \overline{X_1} \cup X_0)}_{3} \cap \underbrace{(\overline{X_3} \cup \overline{X_1} \cup X_0)}_{4} \cap \underbrace{(\overline{X_3} \cup X_2 \cup X_0)}_{5} \cap \underbrace{(X_3 \cup \overline{X_2} \cup X_0)}_{6}$$

Застосуємо закон "заперечення заперечення" до отриманої МКНФ функції y_2, отримаємо:

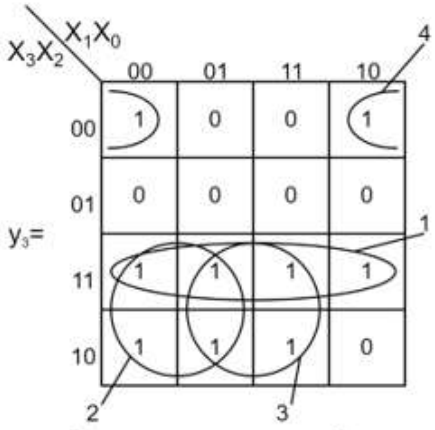
$$y_2 = \overline{\overline{(X_3 \cup X_2 \cup \overline{X_0}) \cap (\overline{X_3} \cup \overline{X_2} \cup \overline{X_0}) \cap (\overline{X_2} \cup \overline{X_1} \cup X_0) \cap (\overline{X_3} \cup \overline{X_1} \cup X_0) \cap (\overline{X_3} \cup X_2 \cup X_0) \cap (X_3 \cup \overline{X_2} \cup X_0)}};$$

Тепер застосуємо закон де Моргана:

$$y_2 = \overline{\overline{X_3 \cup X_2 \cup \overline{X_0}} \cup \overline{\overline{X_3} \cup \overline{X_2} \cup \overline{X_0}} \cup \overline{\overline{X_2} \cup \overline{X_1} \cup X_0} \cup \overline{\overline{X_3} \cup \overline{X_1} \cup X_0} \cup \overline{\overline{X_3} \cup X_2 \cup X_0} \cup \overline{X_3 \cup \overline{X_2} \cup X_0}};$$

Рисунок 10.10 – Отримання МКНФ булевої функції y_2 методом карт Карно

5. Отримання МДНФ для функції y_3



$$= \underbrace{X_3 X_2}_{1} \cup \underbrace{X_3 \overline{X_1}}_{2} \cup \underbrace{X_3 X_0}_{3} \cup \underbrace{\overline{X_3}\,\overline{X_2}\,\overline{X_0}}_{4}$$

Застосуємо закон "заперечення заперечення" до отриманої МДНФ функції y_3, отримаємо:

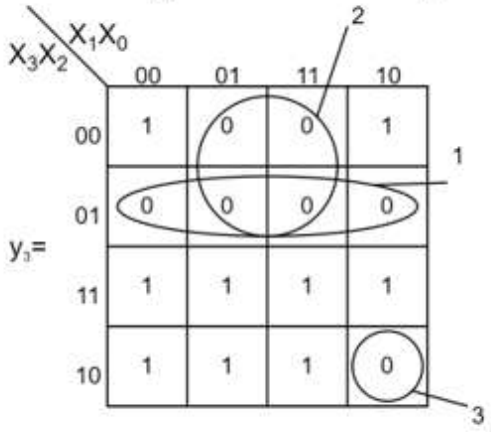
$$y_3 = \overline{\overline{X_3 X_2 \cup X_3 \overline{X_1} \cup X_3 X_0 \cup \overline{X_3}\,\overline{X_2}\,\overline{X_0}}};$$

Тепер застосуємо закон де Моргана:

$$y_3 = \overline{\overline{X_3 X_2} \cap \overline{X_3 \overline{X_1}} \cap \overline{X_3 X_0} \cap \overline{\overline{X_3}\,\overline{X_2}\,\overline{X_0}}};$$

Рисунок 10.11 – Отримання МДНФ булевої функції y_3 методом карт Карно

6. Отримання МКНФ для функції y_3:



$$= \underbrace{(X_3 \cup \overline{X_2})}_{1} \cap \underbrace{(X_3 \cup \overline{X_0})}_{2} \cap \underbrace{(\overline{X_3} \cup X_2 \cup \overline{X_1} \cup X_0)}_{3}$$

Застосуємо закон "заперечення заперечення" до отриманої МКНФ функції y_3, отримаємо:

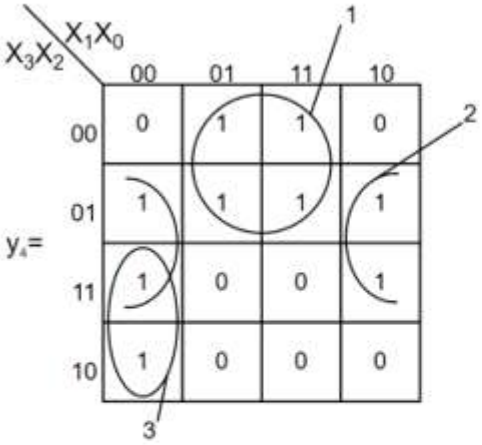
$$y_3 = \overline{\overline{(X_3 \cup \overline{X_2}) \cap (X_3 \cup \overline{X_0}) \cap (\overline{X_3} \cup X_2 \cup \overline{X_1} \cup X_0)}};$$

Тепер отримаємо закон де Моргана:

$$y_3 = \overline{\overline{X_3 \cup \overline{X_2}} \cup \overline{X_3 \cup \overline{X_0}} \cup \overline{\overline{X_3} \cup X_2 \cup \overline{X_1} \cup X_0}};$$

Рисунок 10.12 – Отримання МКНФ булевої функції y_3 методом карт Карно

7. Отримання МДНФ для функції y_4:



$$= \underbrace{\overline{X_3}\, X_0}_{1} \underbrace{X_2\, \overline{X_0}}_{2} \underbrace{X_3 \overline{X_1}\, \overline{X_0}}_{3}$$

Застосуємо закон "заперечення заперечення" до отриманої МДНФ для функції y_4, отримаємо:

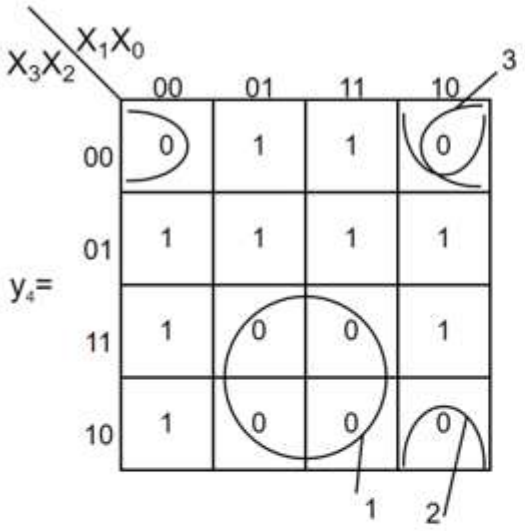
$$y_4 = \overline{\overline{X_3 X_0 \cup X_2 \overline{X_0} \cup X_3 \overline{X_1}\, \overline{X_0}}};$$

Тепер застосуємо закон де Моргана:

$$y_4 = \overline{\overline{X_3 X_0} \cap \overline{X_2 \overline{X_0}} \cap \overline{X_3 \overline{X_1}\, \overline{X_0}}};$$

Рисунок 10.13 – Отримання МДНФ булевої функції y_4 методом карт Карно

8. Отримаємо МКНФ для функції y_4:



$$= \underbrace{(\overline{X_3} \cup \overline{X_0})}_{1} \underbrace{(X_2 \cup \overline{X_1} \cup X_0)}_{2} \underbrace{(X_3 \cup X_2 \cup X_0)}_{3}$$

Застосуємо закон "заперечення заперечення" до отриманої МКНФ для функції y_4, отримаємо:

$$y_4 = \overline{\overline{(\overline{X_3} \cup \overline{X_0}) \cap (X_2 \cup \overline{X_1} \cup X_0) \cap (X_3 \cup X_2 \cup X_0)}};$$

Тепер застосуємо закон де Моргана:

$$y_4 = \overline{\overline{X_3} \cup \overline{X_0} \cup \overline{X_2 \cup \overline{X_1} \cup X_0} \cup \overline{X_3 \cup X_2 \cup X_0}};$$

Рисунок 10.14 – Отримання МКНФ булевої функції y_4 методом карт Карно

В результаті послідовно виконання описаних вище операцій, було отримано наступні аналітичні вирази для булевих логічних функцій $y_1 – y_4$ у базисах І–НЕ та АБО–НЕ відповідно (дивись рисунок 10.15).

Проаналізувавши аналітичний вигляд функцій $y_1 – y_4$ в базисі І–НЕ, можна прийти до висновку: деякі функції містять ідентичні поєднання логічних змінних (терми). Так функції y_1 та y_4 містять спільний терм (позначений як "a"), а функції y_2 та y_3 – спільний терм "b". В базисів АБО–НЕ лише функції y_1 та y_3 містять один спільний терм ("a"). Наявність спільних термів означає, що процес створення логічних схем можна спростити і прискорити шляхом реалізації цих термів лише один раз та подальшого використання результатів обчислення значень ними.

На рисунках 10.16 і 10.17 представлено результат створення комбінаційної схеми для системи булевих функцій $y_1 – y_4$, реалізованої в базисі І–НЕ та часові діаграми її роботи відповідно. На рисунках 10.18 і 10.19 представлено результат створення комбінаційної схеми для системи булевих функцій $y_1 – y_4$, реалізованої в базисі АБО–НЕ та часова діаграма її роботи відповідно.

В базисі І-НЕ:

$$y_1 = \overline{\overline{X_3}\,\overline{X_2}} \cap \underbrace{\overline{X_3}\,\overline{X_1}\,\overline{X_0}}_{a} \cap \overline{X_3}\,\overline{X_2}\,\overline{X_0} \cap \overline{\overline{X_2}\,X_1\,\overline{X_0}};$$

$$y_2 = \overline{\overline{X_3}\,\overline{X_2}\,X_0} \cap \overline{\overline{X_3}\,X_2\,X_0} \cap \underbrace{\overline{\overline{X_3}\,\overline{X_2}\,\overline{X_0}}}_{b} \cap \overline{X_3}\,\overline{X_2}\,\overline{X_1}\,\overline{X_0};$$

$$y_3 = \overline{\overline{X_3}\,\overline{X_2}} \cap \overline{X_3}\,\overline{\overline{X_1}} \cap \overline{X_3}\,\overline{X_0} \cap \underbrace{\overline{\overline{X_3}\,\overline{X_2}\,\overline{X_0}}}_{b};$$

$$y_4 = \overline{\overline{X_3}X_0} \cap \overline{X_2}\overline{X_0} \cap \underbrace{\overline{X_3}\overline{X_1}\,\overline{X_0}}_{a};$$

В базисі АБО-НЕ:

$$y_1 = \underbrace{\overline{X_3 \cup \overline{X_2}}}_{a} \cup \overline{\overline{X_3} \cup X_2 \cup \overline{X_0}} \cup \overline{\overline{X_3} \cup \overline{X_2} \cup \overline{X_1} \cup X_0};$$

$$y_2 = \overline{X_3 \cup X_2 \cup \overline{X_0}} \cup \overline{\overline{X_3} \cup \overline{X_2} \cup \overline{X_0}} \cup \overline{\overline{X_2} \cup \overline{X_1} \cup X_0} \cup \overline{\overline{X_3} \cup \overline{X_1} \cup X_0} \cup \overline{\overline{X_3} \cup X_2 \cup X_0} \cup \overline{X_3 \cup \overline{X_2} \cup X_0};$$

$$y_3 = \underbrace{\overline{X_3 \cup \overline{X_2}}}_{a} \cup \overline{X_3 \cup \overline{X_0}} \cup \overline{\overline{X_3} \cup X_2 \cup \overline{X_1} \cup X_0};$$

$$y_4 = \overline{\overline{X_3} \cup \overline{X_0}} \cup \overline{X_2 \cup \overline{X_1} \cup X_0} \cup \overline{X_3 \cup X_2 \cup X_0};$$

Рисунок 10.15 – Аналітичний опис логічних функцій y_1 – y_4 у різних базисах

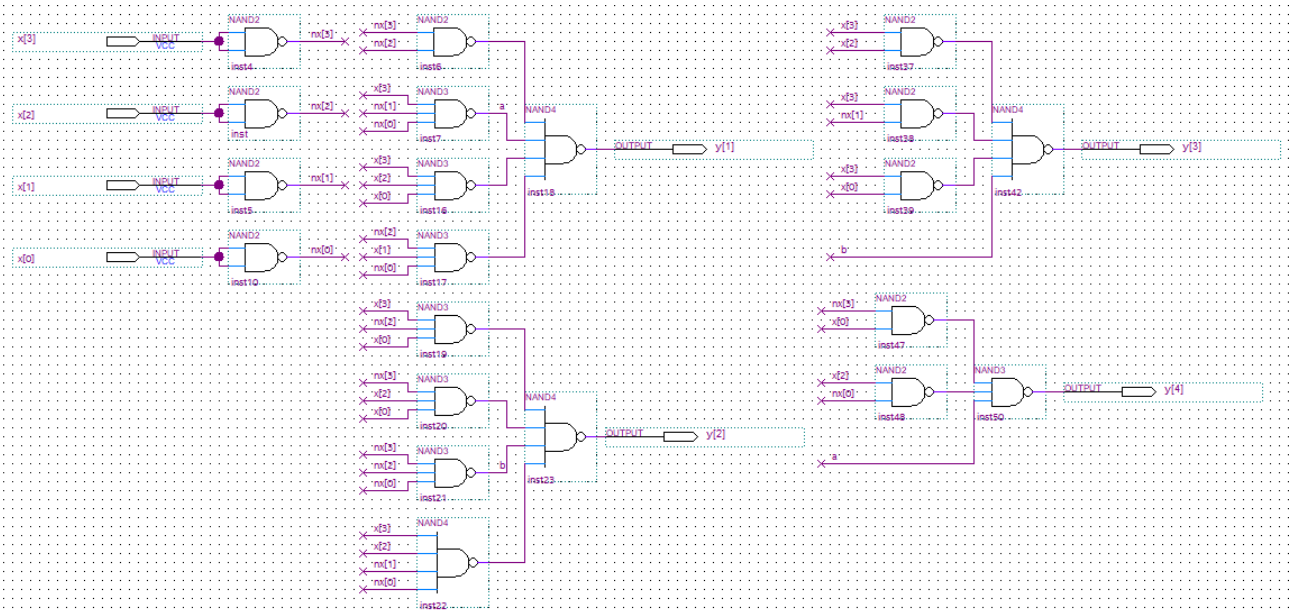


Рисунок 10.16 – Результат схемотехнічної реалізація системи булевих функцій y_1 – y_4 в базисі І–НЕ з урахуванням повторюваності термів

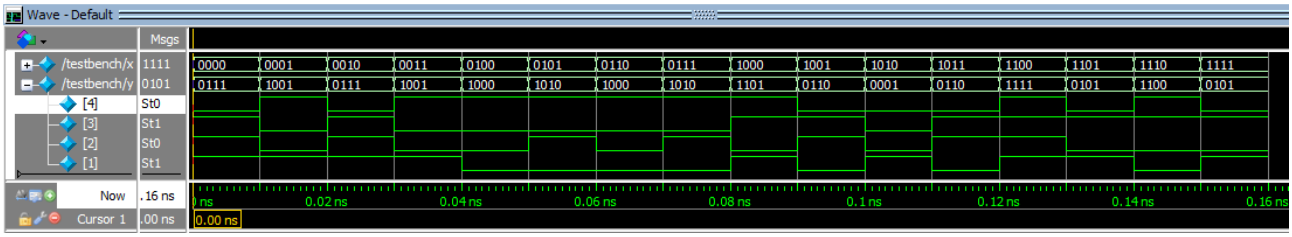


Рисунок 10.17 – Результат симуляції роботи комбінаційної схеми, наведеної на рисунку 10.14, засобами ModelSim

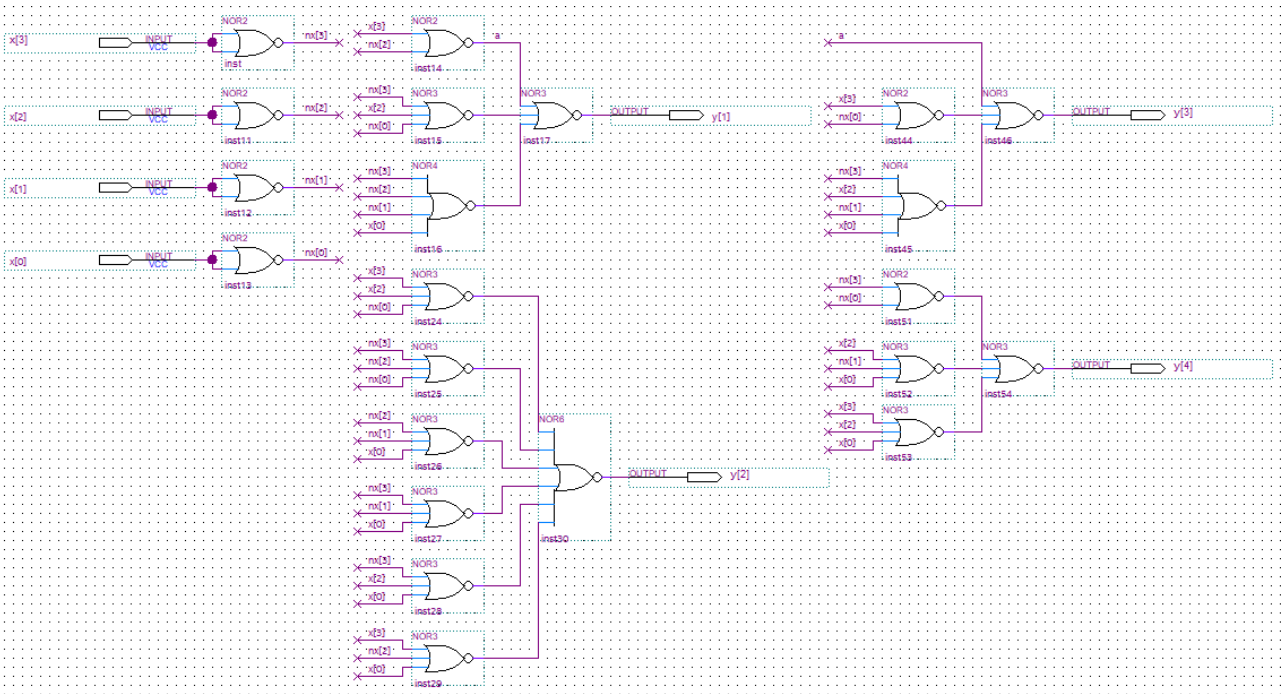


Рисунок 10.18 – Результат схемотехнічної реалізація системи булевих функцій $y_1 – y_4$ в базисі АБО–НЕ з урахуванням повторюваності термів

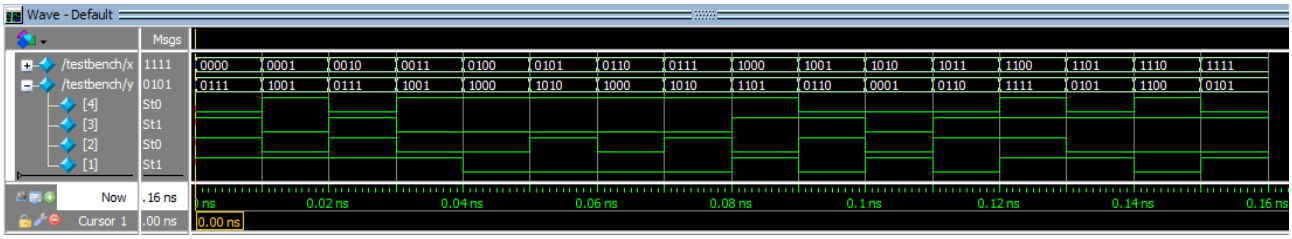


Рисунок 10.19 – Результат симуляції роботи комбінаційної схеми, наведеної на рисунку 2.18, засобами ModelSim

Комбінаційну схему, що відповідає реалізації системи булевих логічних функцій $y_1 – y_4$ в різних базисах рекомендується виконувати у різних проектах САПР Quartus Prime.

10.4 Приклад синтезу комбінаційного пристрою в базисі Жегалкіна

В якості прикладу при розгляді процесу синтезу комбінаційної схеми у базисі Жегалкіна (згідно таблиці 10.3) візьмемо поліном, який було отримано у пункті 10.1. В результаті підстановки необхідних значень a_i, було отримано

представлену нижче (в таблиці 10.5), початкову таблицю істинності булевої функції.

Таблиця 10.5 – Таблиця істинності функції

x_2	x_1	x_0	y
0	0	0	1 (a_7)
0	0	1	0 (a_6)
0	1	0	1 (a_5)
0	1	1	0 (a_4)
1	0	0	1 (a_3)
1	0	1	1 (a_2)
1	1	0	1 (a_1)
1	1	1	1 (a_0)

На рисунках 10.20 і 10.21, зображено процес та результат отримання поліному Жегалкіна для заданої булевої функції шляхом застосування канонічного правила переходу від базису Буля до базису Жегалкіна.

X_2	X_1	X_0	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

1. Застосування канонічного правила переходу від базису Буля до базису Жегалкіна:

а) записати ДДНФ функції y (тобто, без мінімізації):

$$y = \overline{X_2}\,\overline{X_1}\,\overline{X_0} \cup \overline{X_2}\,X_1\,\overline{X_0} \cup X_2\,\overline{X_1}\,\overline{X_0} \cup X_2\,\overline{X_1}\,X_0 \cup X_2\,X_1\,\overline{X_0} \cup X_2\,X_1\,X_0;$$

Б) замінити всі диз'юнкції операціями "виключне або" ($\oplus$ або XOR):

$$y = \overline{X_2}\,\overline{X_1}\,\overline{X_0} \oplus \overline{X_2}\,X_1\,\overline{X_0} \oplus X_2\,\overline{X_1}\,\overline{X_0} \oplus X_2\,\overline{X_1}\,X_0 \oplus X_2\,X_1\,\overline{X_0} \oplus X_2\,X_1\,X_0;$$

в) перетворити всі заперечення змінних за правилом ($\overline{X_i} = X_i \oplus 1$)

Рисунок 10.20 – Послідовність кроків застосування канонічного правила переходу від базису Буля до базису Жегалкіна

$$y = \underbrace{(X_2 \oplus 1)(X_1 \oplus 1)(X_0 \oplus 1)}_{1} \oplus \underbrace{(X_2 \oplus 1)X_1(X_0 \oplus 1)}_{2} \oplus \underbrace{X_2(X_1 \oplus 1)(X_0 \oplus 1)}_{3}$$
$$\oplus \underbrace{X_2(X_1 \oplus 1)X_0}_{4} \oplus \underbrace{X_2X_1(X_0 \oplus 1)}_{5} \oplus \underbrace{X_2X_1X_0}_{6}$$

г) розкрити дужки і здійснити приведення усіх утворених доданків за правилом $(X_i \oplus X_i = 0)$:

$$y = \underbrace{(X_2X_1 \oplus X_2 \oplus X_1 \oplus 1)(X_0 \oplus 1)}_{1} \oplus \underbrace{(X_2X_1 \oplus X_1)(X_0 \oplus 1)}_{2} \oplus$$
$$\oplus \underbrace{(X_2X_1 \oplus X_2)(X_0 \oplus 1)}_{3} \oplus \underbrace{X_2X_1X_0 \oplus X_2X_0}_{4} \oplus \underbrace{X_2X_1X_0 \oplus X_2X_1}_{5} \oplus \underbrace{X_2X_1X_0}_{6} =$$
$$= X_2X_1X_0 \oplus X_2X_1 \oplus X_2X_0 \oplus X_2 \oplus X_1X_0 \oplus X_1 \oplus X_0 \oplus 1 \oplus X_2X_1X_0 \oplus X_2X_1 \oplus$$
$$\oplus X_1X_0 \oplus X_1 \oplus X_2X_1X_0 \oplus X_2X_1 \oplus X_2X_0 \oplus X_2 \oplus X_2X_1X_0 \oplus X_2X_0 \oplus X_2X_1 =$$
$$= \boxed{1 \oplus X_0 \oplus X_2X_0}$$

Рисунок 10.21 – Результат отримання канонічного поліному Жегалкіна для булевої логічної функції, заданої таблицею 10.5

На рисунку 10.22 зображено процес та результат отримання поліному Жегалкіна для заданої булевої функції шляхом застосування методу трикутника Паскаля.

2. Застосування методу трикутника Паскаля:

X_2	X_1	X_0	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Трикутник Паскаля															Доданки
①		0		1		0		1		1		1		1	1
	①		1		1		1		0		0		0		X_0
		0		0		0		1		0		0			X_1
			0		0		1		1		0				X_1X_0
				0		1		0		1					X_2
					①		1		1						X_2X_0
						0		0							X_2X_1
							0								$X_2X_1X_0$

$$y = \boxed{1 \oplus X_0 \oplus X_2X_0}$$

Рисунок 10.22 – Результат отримання поліному Жегалкіна для булевої логічної функції, заданої таблицею 10.5 методом трикутника Паскаля

Порівнюючи між собою результати, показані на рисунках 10.21 і 10.220, можна зробити висновок про правильність отримання канонічного поліному Жегалкіна для булевої логічної функції, заданої таблицею 10.5.

На рисунках 10.23 і 10.24 представлено результат створення комбінаційної схеми для булевої функції, реалізованої в базисі Жегалкіна, та часова діаграма її функціонування відповідно.

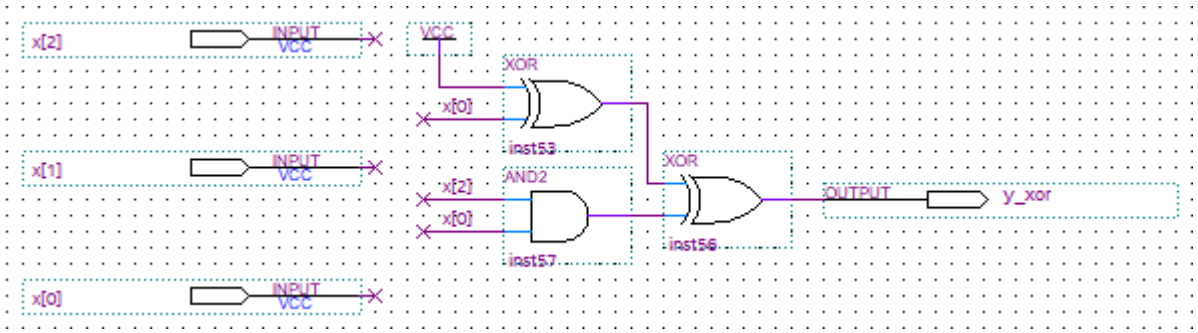


Рисунок 10.23 – Результат схемотехнічної реалізація булевої функції в базисі Жегалкіна

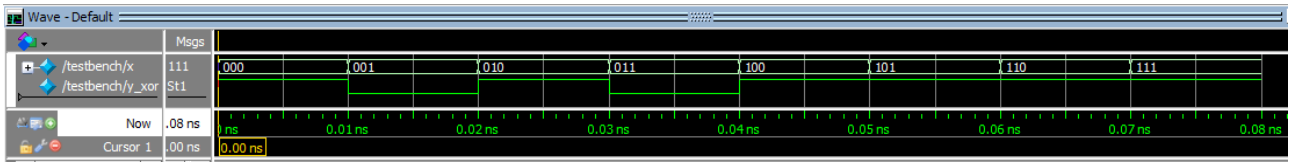


Рисунок 10.24 – Результат симуляції роботи комбінаційної схеми, наведеної на рисунку 10.21, засобами ModelSim

На рисунку 10.25 для реалізації конституенти “1” було використано компонент “VCC” бібліотеки САПР Quartus Prime. У випадку, якщо необхідно реалізувати конституенту “0” слід використовувати компонент GND, зовнішній вигляд якого наведено на рисунку 10.23.

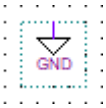


Рисунок 10.25 – Зовнішній вигляд компоненту “GND”

10.5 Вимоги до змісту звіту

Звіт про виконання лабораторної роботи повинен містити:

1. Номер і тему роботи.
2. Мету виконання роботи.
3. Короткі теоретичні відомості.
4. Порядок виконання роботи.
5. Результати виконання роботи у вигляді:

– утвореного двійкового слова згідно пункту 10.2.1;

– результати виконання пунктів 10.2.2 – 10.2.10, в яких представити початкові таблиці істинності функцій, весь процес отримання МДНФ та МКНФ, перетворення їх до базису І-НЕ та АБО-НЕ, схеми електричні принципові та часові діаграми роботи синтезованих цифрових пристроїв.

6. Особливості функціонування САПР Intel Quartus Prime, виявлені під час виконання роботи.
7. Висновки.

10.6 Контрольні питання для перевірки знань

10.6.1 Сформулюйте визначення логічної функції, конституенти 1 або 0, імпліканти і простої імпліканти функції.

10.6.2 Що таке досконала, скорочена, тупикова і мінімальна ДНФ (або КНФ)?

10.6.3 Дайте визначення функціональної повноти булевих функцій.

10.6.4 У чому полягає сутність мінімізації булевих функцій?

10.6.5 Охарактеризуйте основні етапи синтезу комбінаційних схем.

10.6.6 В чому полягає специфіка синтезу комбінаційних схем з декількома виходами?

10.6.7 Яким чином здійснюється розв’язка схем у відповідності до коефіцієнтів об’єднання і розгалуження логічних елементів?

11 ЛАБОРАТОРНА РОБОТА № 3
ПРОЕКТУВАННЯ ТА ДОСЛІДЖЕННЯ ТРИГЕРІВ, ЛІЧИЛЬНИКІВ, РЕГІСТРІВ

Мета роботи: вивчити принципи функціонування різних типів тригерів, лічильників і регістрів, а також методи їх синтезу.

11.1 Короткі теоретичні відомості

11.1.1 Загальні відомості про тригери та їх класифікації

Тригер являє собою пристрій з двома стійкими станами і широко використовується в цифрових схемах як елемент пам'яті. Тригери відрізняються як за функціональним призначенням, так і за способом запису інформації.

Серед усіх існуючих типів тригерів, найбільш широке застосування знайшли наступні: T, D, JK, DV, RS, RSR, RSS, RSE. Функціонування тригерів представляється таблицею переходів, яка характеризує стани входів і виходів тригера в поточний момент часу t^s (до його перемикання) та в наступний (t^{s+1}), вже після перемикання.

В таблиці 11.1 представлено скорочену таблицю переходів стандартного RS-тригера, а також всіх його модифікацій (RSR, RSS, RSE). Із неї випливає, що:

– RS-тригер не змінює свого стану (зберігає попередній) в момент часу t^{s+1}, тобто $Q^{s+1} = Q^s$, якщо в момент часу t^s на його входи подають сигнали $R^s = S^s = 0$;

– RS-тригер здійснює встановлення, тобто $Q^{s+1} = 1$ при комбінації на вході $R^s = 0$ та $S^s = 1$;

– RS-тригер здійснює скидання, тобто $Q^{s+1} = 0$ при комбінації на вході $R^s = 1$ та $S^s = 0$;

– при комбінації вхідних сигналів $R^s = 1$ та $S^s = 1$ вихідний стан тригера в наступний момент є невизначеним, тобто, $Q^{s+1} = *$, це означає, що така комбінація для нього є забороненою.

RSR-тригер (або RS with Reset) відрізняється від стандартного RS тим, що при комбінації вхідних сигналів $R^s = 1$ та $S^s = 1$ він переходить у стан скидання, тобто $Q^{s+1} = 0$.

RSS-тригер (або RS with Set) в зазначеному випадку перейде у встановлення, тобто $Q^{s+1} = 1$.

RSE-тригер (RS with Enable) в такому випадку буде зберігати попередній стан, тобто $Q^{s+1} = Q^s$.

Таблиця 11.1 – Таблиця переходів RS-, RSR-, RSS- та RSE-тригерів

Момент часу t^s		**Момент часу t^{s+1}**			
R^s	**S^s**	**Q^{s+1}**			
		RS	**RSR**	**RSS**	**RSE**
0	0	Q^s	Q^s	Q^s	Q^s
0	1	1	1	1	1
1	0	0	0	0	0
1	1	*	0	1	Q^s

D-тригер ще називають тригером затримки, оскільки він здійснює затримку появи вихідного сигналу відносно вхідного, що подається на інформаційний вхід D. Скорочену таблицю переходів D-тригера наведено нижче, в таблиці 11.2. Як бачимо, для нього є справедливою наступна рівність:

$$Q^{s+1} = D^s \qquad (11.1)$$

Таблиця 11.2 – Таблиця переходів D-тригера

Момент часу t^s	**Момент часу t^{s+1}**
D^s	**Q^{s+1}**
0	0
1	1

DV-тригер відрізняється від D-тригера наявністю додаткового входу V. Скорочену таблицю переходів DV-тригера наведено нижче, в таблиці 11.3. Як бачимо, вхід V грає роль входу дозволу на перемикання, що працює за наступним алгоритмом:

– якщо $V^s = 1$, то DV-тригер буде перемикати свої стани у відповідності до таблиці 11.2;

– якщо $V^s = 0$, тригер переходить в режим зберігання свого попереднього стану.

Таблиця 11.3 – Таблиця переходів DV-тригера

Момент часу t^s		**Момент часу t^{s+1}**
D^s	**V^s**	**Q^{s+1}**
0	0	Q^s
0	1	0
1	0	Q^s
1	1	1

T-тригер також називають лічильним тригером, оскільки він здійснює підрахунок по модулю 2 кількості одиниць, які надходять на вхід T, що легко бачити із його скороченої таблиці переходів (таблиця 11.4).

Таблиця 11.4 – Таблиця переходів Т-тригера

Момент часу t^s	**Момент часу t^{s+1}**
T^s	**Q^{s+1}**
0	Q^s
1	notQ^s

JK-тригер ще по-іншому називають універсальним, оскільки при комбінації вхідних сигналів $J^s = K^s = 1$ він змінює свій вихідний стан на протилежний, повністю аналогічно до того, як це робить лічильний тригер Т, при інших комбінаціях він працює як D- або RS-тригер. Скорочену таблицю переходів JK-тригера наведено нижче, в таблиці 11.5.

Таблиця 11.5 – Таблиця переходів JK-тригера

Момент часу t^s		**Момент часу t^{s+1}**
J^s	**K^s**	**Q^{s+1}**
0	0	Q^s
0	1	0
1	0	1
1	1	notQ^s

У відповідності до того, в який спосіб тригери записують вхідні дані, вони поділяються на 2 види:

– асинхронні;

– синхронні.

Асинхронні тригери мають лише інформаційні входи. Запис інформації в них здійснюється безпосередньо при надходженні інформаційних сигналів.

Синхронні тригери мають входи тактування. Сигнали синхронізації (або тактування) задають частоту зміни інформації в дискретні моменти часу.

Виділяють наступні типи синхронних тригерів:

– статичні, які керуються рівнем тактового сигналу;

– динамічні, які керуються перепадом тактового сигналу.

Тригери першого типу при утримуванні активного рівня тактового сигналу можуть перемикатися стільки разів, скільки змінюються інформаційні сигнали. Іншими словами, синхронні статичні тригери при активному рівні тактового сигналу поводяться подібно до асинхронних.

У тригерах другого типу (динамічних) вихідні сигнали, що відповідають новому стану, з’являються тільки в момент переходу тактового сигналу із логічного “0” в “1” (такий момент називається переднім фронтом, або просто фронтом) або навпаки, тобто із логічної “1” в “0” (такий момент називається заднім фронтом, або зрізом).

11.1.2 Загальні відомості про лічильники

Лічильником називається послідовнісний пристрій, який призначено для підрахунку і зберігання числа імпульсів, поданих у визначеному часовому інтервалі на його лічильний вхід.

Виділяють наступні класифікації лічильників:

I. В залежності від напрямку лічби.

II. За способом організації схеми переносу.

III. За коефіцієнтом перерахунку.

IV. За наявністю синхронізації перемикання лічильників.

В залежності від напрямку лічби виділяють наступні види:

– лічильники, що додають (інкрементні);

– лічильники, що віднімають (декрементні);

– реверсивні (двонаправлені).

За способом організації схеми переносу:

– лічильники із послідовним переносом;

– лічильники із наскрізним переносом;

– лічильники із паралельним переносом;

– лічильники із груповим (комбінованим) переносом.

За коефіцієнтом перерахунку виділяють:

– двійкові лічильники;

– двійково-десяткові лічильники;

– із довільним постійним коефіцієнтом перерахунку;

– зі змінним коефіцієнтом перерахунку.

За наявністю синхронізації перемикання лічильників:

– синхронні;

– асинхронні.

Основними специфічними технічними параметрами лічильників є:

1. Коефіцієнт перерахунку – максимальне число одиничних імпульсів, які можна полічити без переповнення.

2. Швидкодія, що характеризується:

а) максимальною частотою надходження лічильних імпульсів (f_{MAX}), які лічильник може реєструвати без збоїв;

б) часом встановлення ($t_{ВСТ}$) станів лічильника, який визначається як максимальний інтервал часу від моменту надходження лічильного імпульсу і до моменту переходу всіх його розрядів у новий стійкий стан;

в) максимальною частотою зміни станів ($f_{С.MAX}$), що визначається за найгіршим часом встановлення $t_{ВСТ}$ з урахуванням часу дешифрації (t_0).

11.1.3 Загальні відомості про регістри та їх класифікації

Регістром називається послідовнісний пристрій, який призначено для прийому, зберігання, простих перетворень і передачі двійкових чисел.

Під простими перетвореннями розуміють зсув числа на задану кількість розрядів, а також перетворення послідовного коду на паралельний і навпаки.

Базовими елементами є тригери, які доповнюють комбінаційними логічними елементами для реалізації різноманітних зв’язків між тригерами і для керування прийомом і передачею інформації.

Основне функціональне призначення – оперативна пам’ять для зберігання багаторозрядних двійкових чисел.

Всі регістри класифікують за способом прийому і передачі інформації:

– паралельні (паралельно-паралельні);

– послідовні (послідовно-послідовні);

– послідовно-паралельні;

– паралельно-послідовні;

– універсальні (можуть працювати як декілька різних видів).

В паралельних регістрах ввод і вивод всіх розрядів числа здійснюється одночасно, за 1 такт. Такі регістри є основним функціональним елементом оперативних запам’ятовуючих пристроїв (ОЗП).

В послідовних регістрах ввод і вивод інформації здійснюється через один інформаційний вхід і один вихід порозрядно із зсувом числа. Тому їх ще називають зсувними. Зсувний регістр, що реалізує зсув вправо або вліво в залежності від керуючих сигналів, називається реверсивним.

Послідовно-паралельні регістри мають один інформаційний вхід для послідовного вводу числа в режимі зсуву і вихідні елементи для видачі n-розрядного числа паралельним кодом.

В паралельно-послідовних регістрах інформація вводиться паралельним кодом за 1 такт через вхідні елементи, що тактуються, а виводиться послідовно по одному розряду в кожному тактовому періоді.

Універсальні регістри поєднують в собі можливості всіх типів регістрів, крім того, забезпечують режими вимикання входів і виходів регістра від інформаційної шини, перекомутацію місцями входів і виходів регістра, керування напрямком зсуву, тощо.

11.2 Порядок виконання роботи

11.2.1 Синтезувати схеми для дослідження синхронних динамічних тригерів згідно варіанту завдання, наведеного в таблиці 11.1.

Таблиця 11.1 – Варіант завдання №1

№ варіанту завдання	Найменування тригера за функціональним призначенням	Назва примітиву, що моделює роботу тригера
1	D	dff
		dffe
2	JK	jkff
		jkffe
3	T	tff
		tffe

Продовження таблиця 11.1

№ варіанту завдання	Найменування тригера за функціональним призначенням	Назва примітиву, що моделює роботу тригера
4	RS	srff
		srffe
5	JK	jkff
		jkffe
6	T	tff
		tffe
7	RS	srff
		srffe
8	D	dff
		dffe
9	T	tff
		tffe
10	RS	srff
		srffe
11	D	dff
		dffe
12	JK	jkff
		jkffe
13	RS	srff
		srffe
14	D	dff
		dffe
15	JK	jkff
		jkffe
16	T	tff
		tffe

Примітки.

1. Номер варіанту завдання обирається у відповідності до порядкового номера студента згідно списку у журналі академічної групи.

2. Якщо порядковий номер студента перевищує 16, то від нього віднімається число 16, а отримана різниця – і буде номером варіанта.

11.2.2 Побудувати часові діаграми функціонування кожної із отриманих схем.

Правильно побудованою часовою діаграмою вважається така, за допомогою якої можна перевірити усі переходи, зазначені у відповідній таблиці для заданого типу тригера.

Нижче, на рисунках 11.1 та 11.2 показано результат створення схеми для дослідження D-тригера на основі примітива dff САПР Quartus, а також часову діаграму його функціонування.

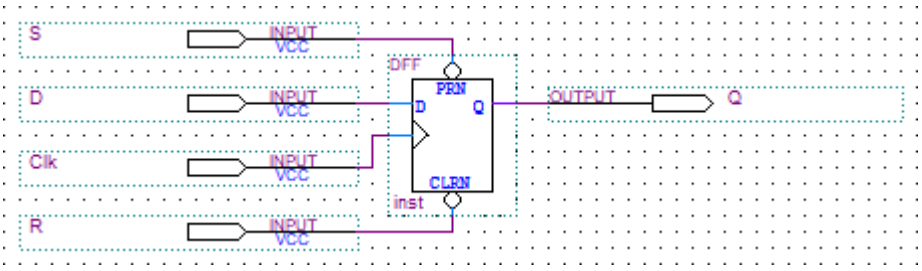


Рисунок 11.1 – Схема для дослідження роботи D-тригера

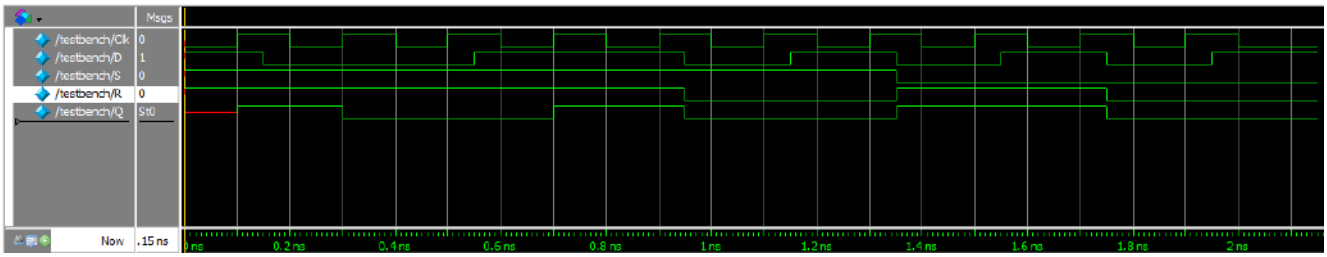


Рисунок 11.2 – Часова діаграма функціонування досліджуваного тригера

11.2.3 На основі отриманих часових діаграм з'ясувати та описати:

– функціональне призначення кожного із входів і виходів тригера (для тригерів, в назві яких міститься літера e – тільки призначення входу ENA);

– при якій події тактового сигналу відбувається перемикання тригера (зміна його стану);

– вплив сигналів !CLRN, !PRN на функціонування зазначеного тригера, їх активні рівні та режим перемикання (синхронний/асинхронний);

У разі виникнення сумнівів щодо правильності виконання пункту 2.3 скористатися онлайн системою допомоги за посиланням: https://www.intel.com/content/www/us/en/programmable/quartushelp/17.0/index.htm#quartus/gl_quartus_welcome.htm. Після запуску якої вказати назву шуканого тригера у полі Search (наприклад, dff).

11.2.4 Синтезувати схему чотирьохрозрядного асинхронного лічильника у відповідності до варіанта завдання згідно таблиці 11.2.

Таблиця 11.2 – Варіант завдання №2

№ варіанту завдання	**Тип переносу**	**Напрямок лічби**	**Найменування тригера для побудови лічильника**
1	послідовний	що додає	dff
2			tff
3			jkff
4		що віднімає	dff
5			jkff
6			tff
7		реверсивний	tff
8			dff
9			jkff

Продовження таблиці 11.2

№ варіанту завдання	Тип переносу	Напрямок лічби	Найменування тригера для побудови лічильника
10	наскрізний	що додає	tff
11			jkff
12			dff
13		що віднімає	jkff
14			tff
15			dff
16		реверсивний	jkff
17			dff
18			Tff

Примітки.

1. Номер варіанту завдання обирається у відповідності до порядкового номера студента згідно списку у журналі академічної групи.

2. Якщо порядковий номер студента перевищує 18, то від нього віднімається число 18 стільки разів, скільки необхідно для отримання додатного числа, менше, ніж 18 а отримана різниця – і буде номером варіанта.

11.2.5 Побудувати часову діаграму функціонування отриманої схеми.

Правильно побудованою часовою діаграмою вважається така, за допомогою якої можна перевірити як мінімум 2 випадки переповнення/антипереповнення лічильника, якщо він лічить в одному напрямку (додає/віднімає), або як мінімум дві зміни напрямку лічби у випадку реверсивного лічильника.

Нижче, на рисунках 11.3 – 11.6 попарно відображено результати синтезу схем асинхронних лічильників із послідовним та наскрізним переносом, а також часові діаграми їх функціонування.

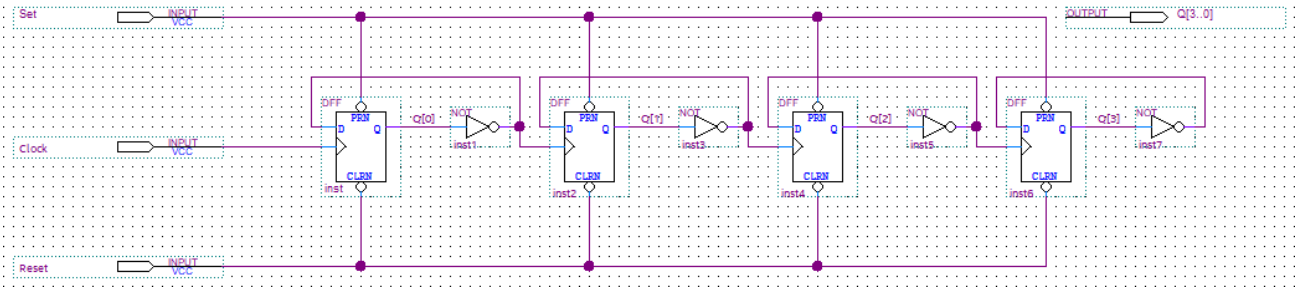


Рисунок 11.3 – Схема асинхронного лічильника, що додає, із послідовним переносом на D-тригері

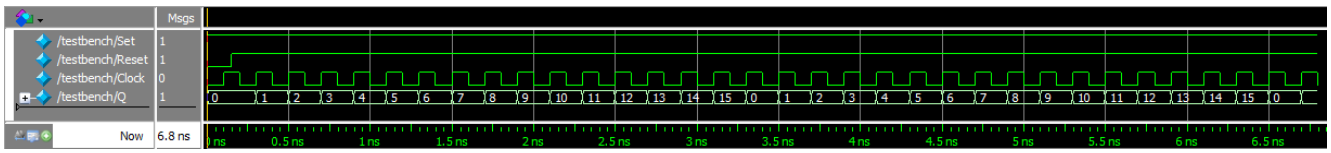


Рисунок 11.4 – Часова діаграма функціонування асинхронного лічильника, що додає, із послідовним переносом на D-тригері

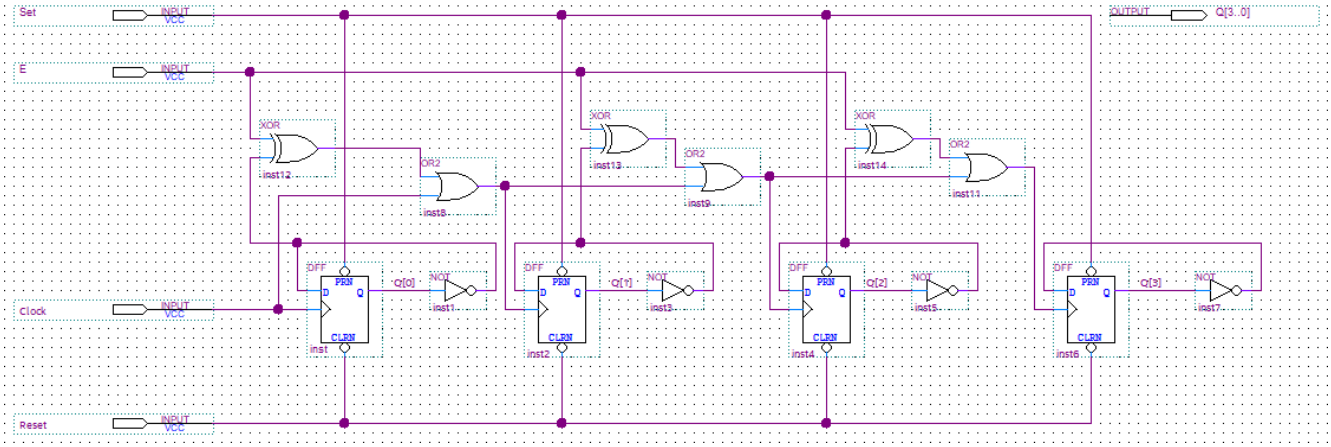


Рисунок 11.5 – Схема асинхронного реверсивного лічильника із наскрізним переносом на D-тригері

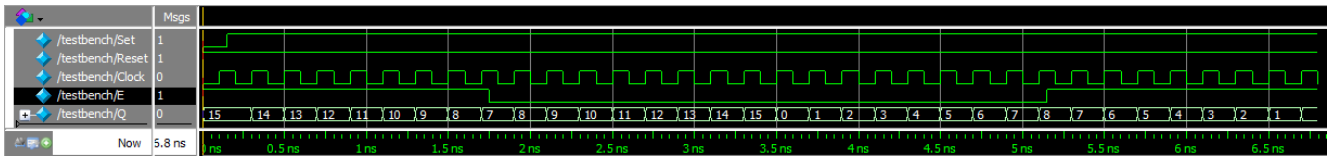


Рисунок 11.6 – Часова діаграма функціонування асинхронного реверсивного лічильника із наскрізним переносом на D-тригері

11.2.6 Синтезувати схеми чотирьохрозрядного синхронного лічильника у відповідності до варіанта завдання згідно таблиці 11.3.

Таблиця 11.3 – Варіант завдання №3

№ варіанту завдання	**Тип переносу**	**Напрямок лічби**	**Найменування тригера для побудови лічильника**
1	наскрізний	що додає	tff
2	наскрізний	що додає	jkff
3	наскрізний	що додає	dff
4	наскрізний	що віднімає	jkff
5	наскрізний	що віднімає	tff
6	наскрізний	що віднімає	dff
7	наскрізний	реверсивний	jkff
8	наскрізний	реверсивний	dff
9	наскрізний	реверсивний	tff
10	паралельний	що додає	dff
11	паралельний	що додає	tff

Продовження таблиці 11.3

№ варіанту завдання	Тип переносу	Напрямок лічби	Найменування тригера для побудови лічильника
12	паралельний	що додає	jkff
13	паралельний	що віднімає	dff
14	паралельний	що віднімає	jkff
15	паралельний	що віднімає	tff
16	паралельний	реверсивний	tff
17	паралельний	реверсивний	dff
18	паралельний	реверсивний	Jkff

Примітки.

1. Номер варіанту завдання обирається у відповідності до порядкового номера студента згідно списку у журналі академічної групи.

2. Якщо порядковий номер студента перевищує 9, то від нього віднімається число 9 стільки разів, скільки необхідно для отримання додатного числа, менше, ніж 9 а отримана різниця – і буде номером варіанта.

11.2.7 Побудувати часову діаграму функціонування отриманої схеми.

Правильно побудованою часовою діаграмою вважається така, за допомогою якої можна перевірити як мінімум 2 випадки переповнення/антипереповнення лічильника, якщо він лічить в одному напрямку (додає/віднімає), або як мінімум дві зміни напрямку лічби у випадку реверсивного лічильника.

Нижче, на рисунках 11.7 – 11.10 попарно відображено результати синтезу схем синхронних лічильників із наскрізним та паралельним переносом, а також часові діаграми їх функціонування.

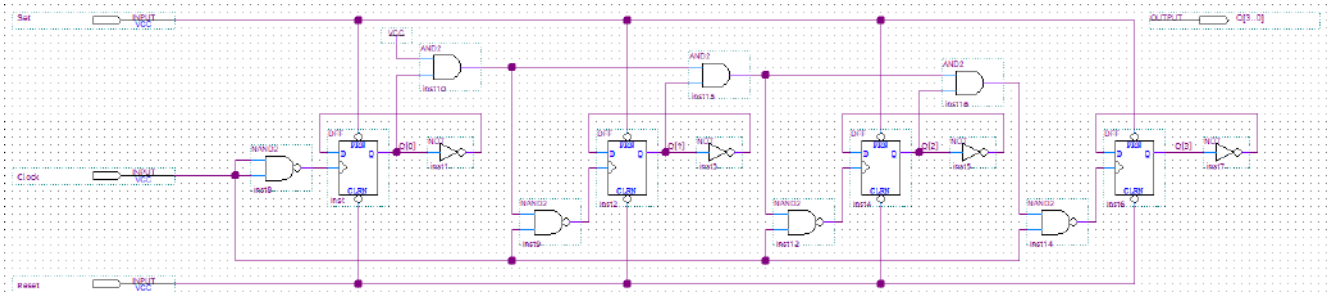


Рисунок 11.7 – Схема синхронного лічильника, що додає, із наскрізним переносом на D-тригері

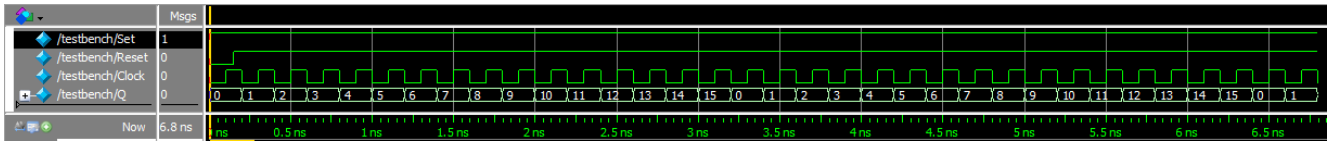


Рисунок 11.8 – Часова діаграма функціонування синхронного лічильника, що додає, із наскрізним переносом на D-тригері

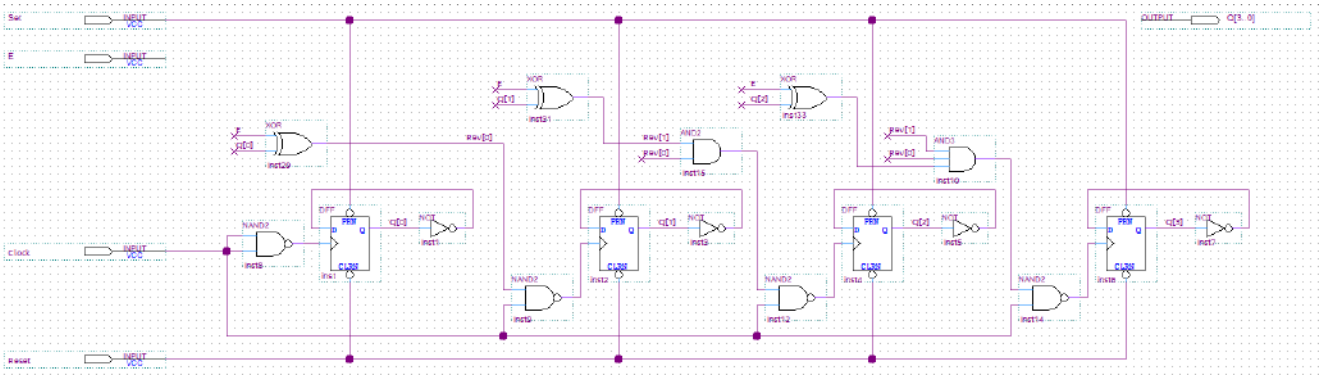


Рисунок 11.9 – Схема синхронного реверсивного лічильника із паралельним переносом на D-тригері

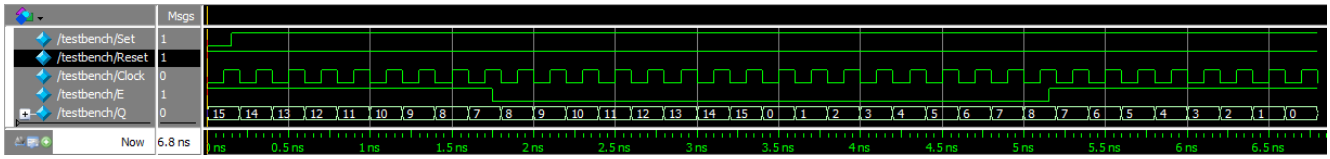


Рисунок 11.10 – Часова діаграма функціонування синхронного реверсивного лічильника із паралельним переносом на D-тригері

11.2.8 Синтезувати схему чотирьох розрядного регістра у відповідності до варіанту завдання, наведеного в таблиці 11.4 із врахуванням заданого типу тригера згідно функціонального призначення.

Таблиця 11.4 – Варіант завдання №4

№ варіанту завдання	**Тип регістра за способом запису/читання інформації**	**Найменування тригера за функціональним призначенням**
1	паралельний	tff
2	послідовно-паралельний	
3	послідовний	
4	паралельно-послідовний	
5	паралельний	jkff
6	послідовно-паралельний	
7	послідовний	
8	паралельно-послідовний	
9	паралельний	srff
10	послідовно-паралельний	
11	послідовний	
12	паралельно-послідовний	

Примітки.

1. Номер варіанту завдання обирається у відповідності до порядкового номера студента згідно списку у журналі академічної групи.

2. Якщо порядковий номер студента перевищує 12, то від нього віднімається число 12 стільки разів, скільки необхідно для отримання додатного числа, менше, ніж 12 а отримана різниця – і буде номером варіанта.

Нижче, на рисунках 11.11 – 11.18, попарно наведено базові схеми регістрів, а також часові діаграми їх функціонування.

11.2.9 Побудувати часову діаграму функціонування синтезованого регістру і порівняти її з наведеною нижче (на рисунках 11.12, 11.14, 11.16, 11.18) для аналогічного типу регістру.

11.2.10 На основі проведених порівнянь пояснити, яким чином відбувається запис інформації в регістр, її читання із нього, а також те, який сигнал дозволяє запис інформації.

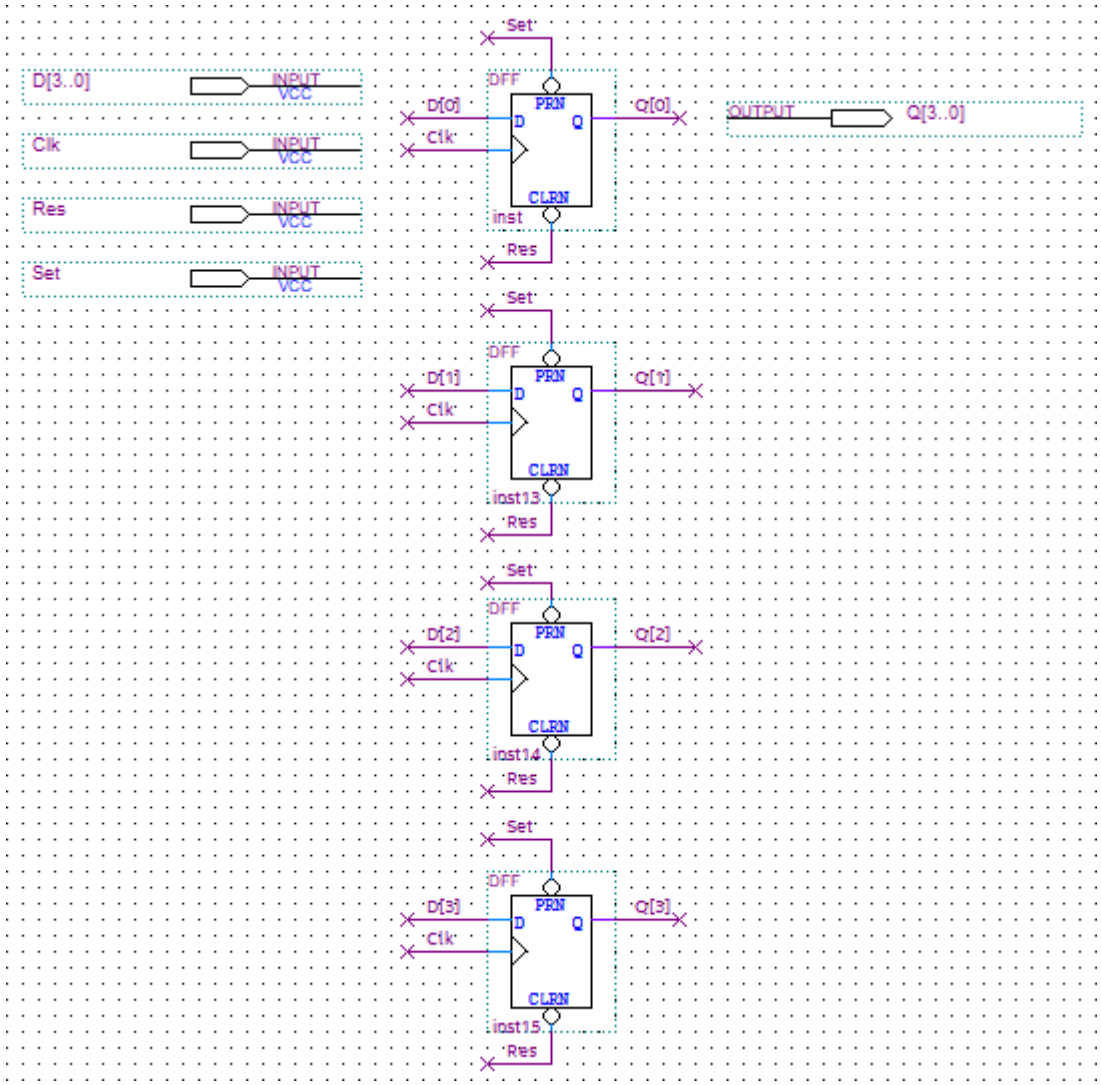


Рисунок 11.11 – Схема паралельно-паралельного регістру на D-тригері

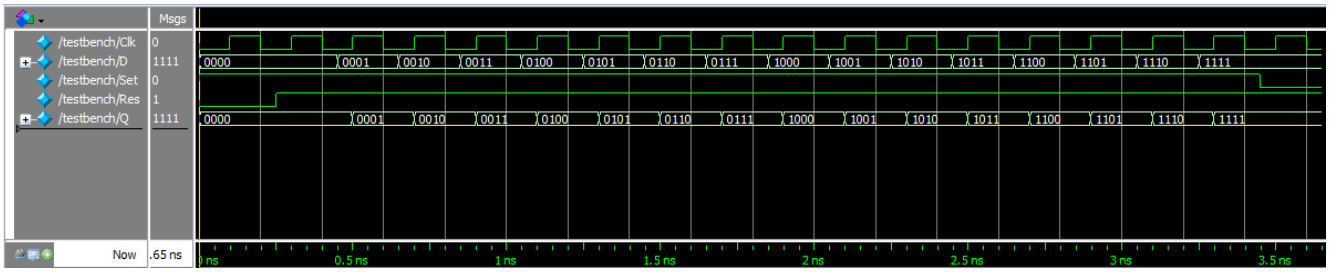


Рисунок 11.12 – Часова діаграма функціонування паралельного регістра на D-тригері

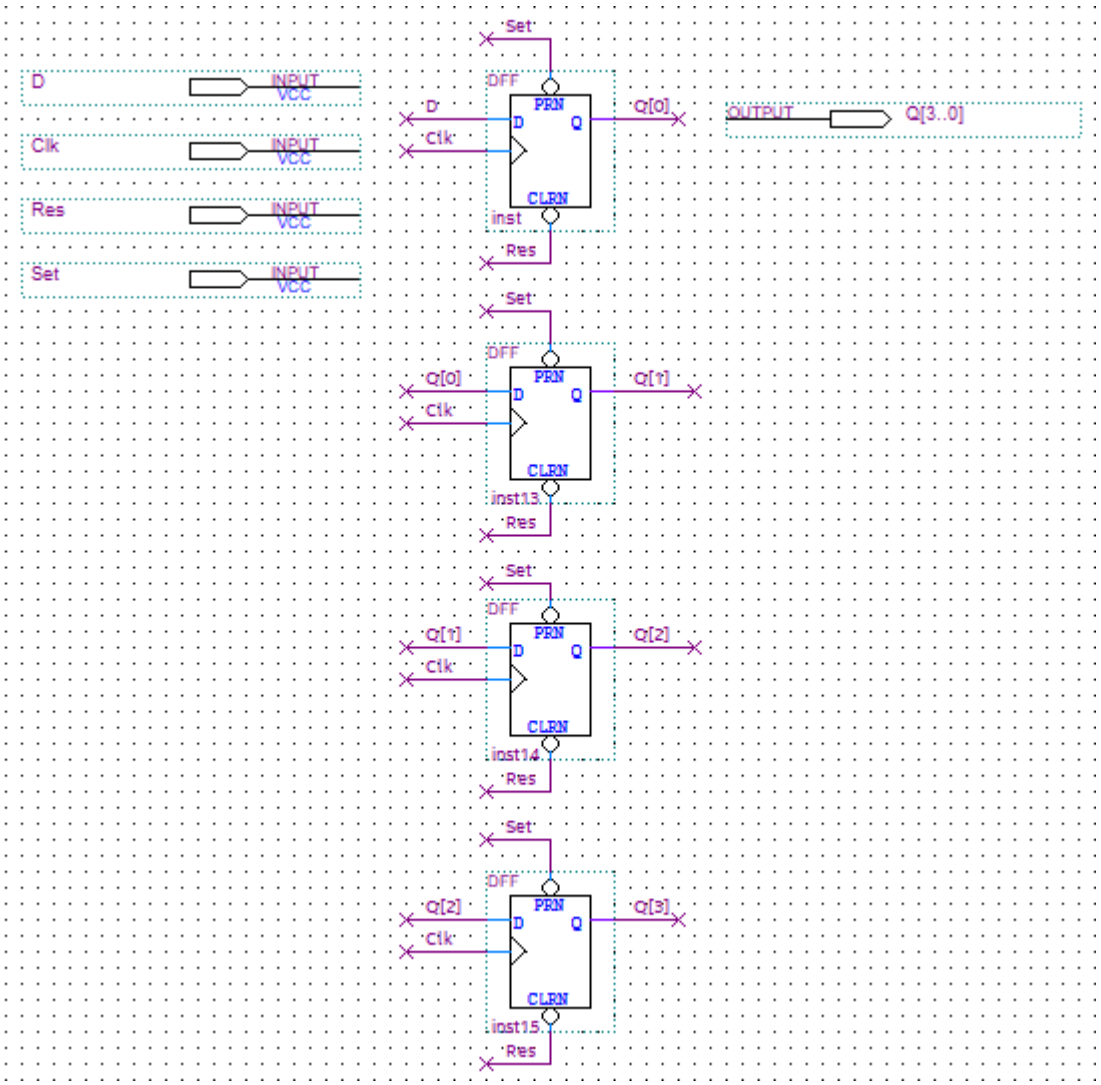


Рисунок 11.13 – Схема послідовно-паралельного регістра на D-тригері

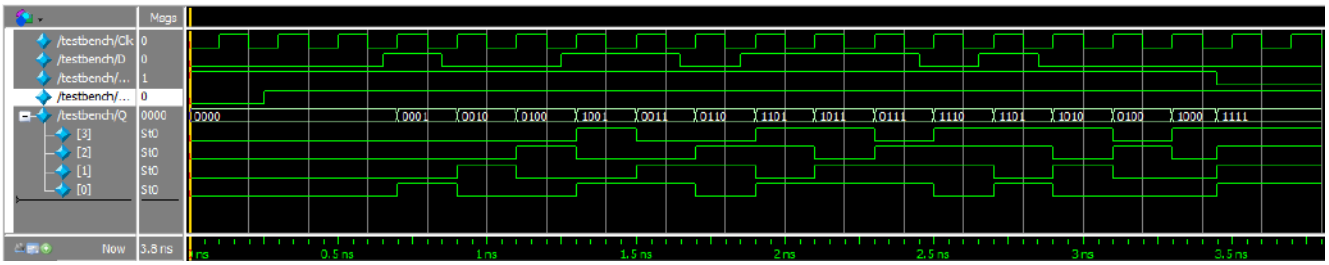


Рисунок 11.14 – Часова діаграма функціонування послідовно-паралельного регістра на D-тригері

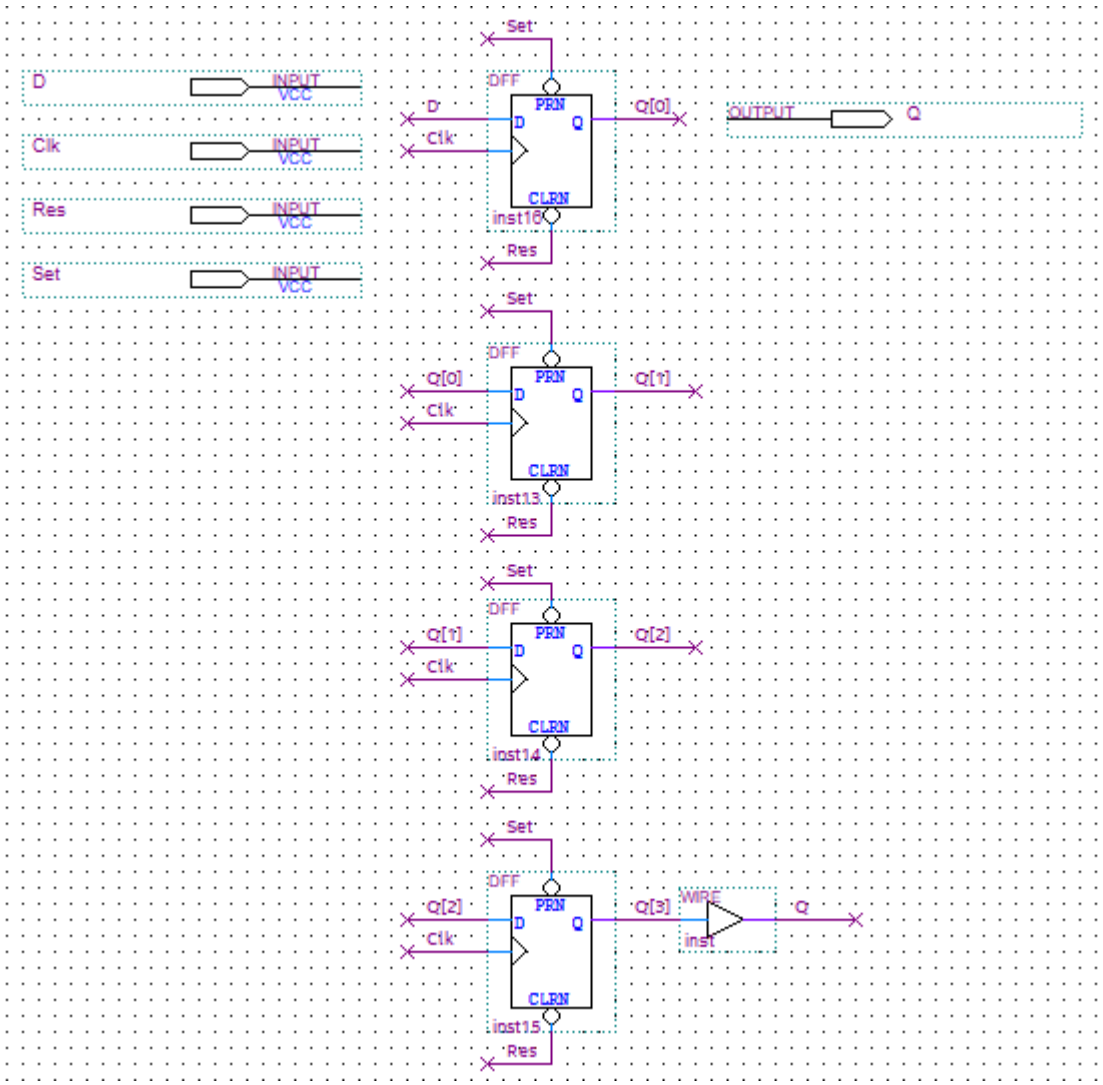


Рисунок 11.15 – Схема послідовного регістра на D-тригері

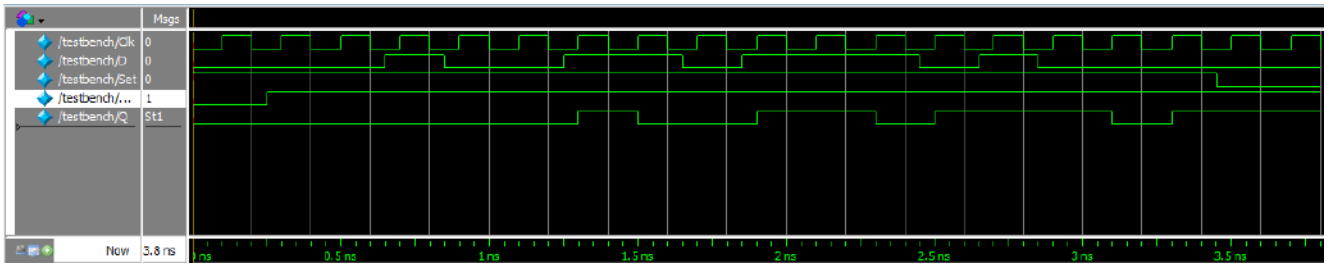


Рисунок 11.16 – Часова діаграма функціонування послідовного регістра на D-тригері

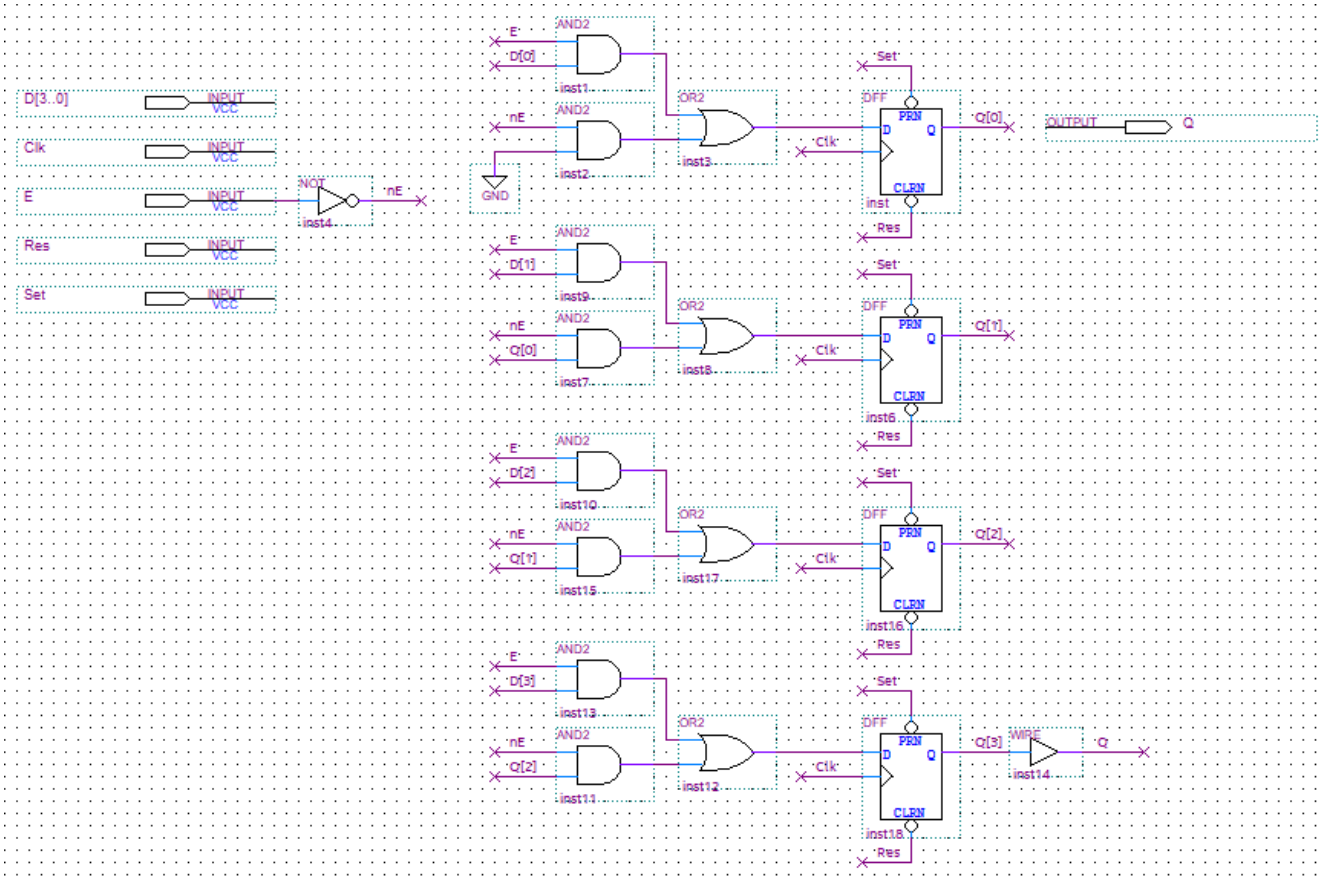


Рисунок 11.17 – Схема паралельно-послідовного регістра на D-тригері

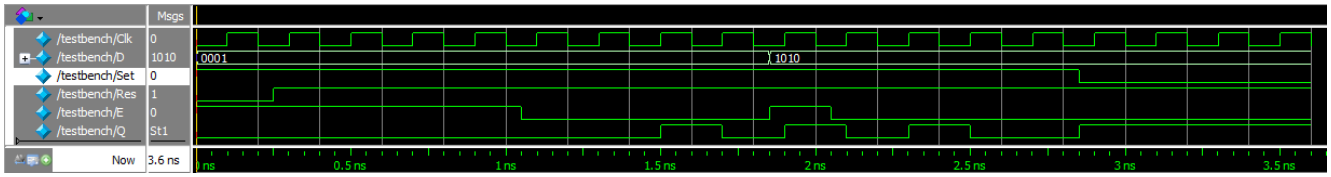


Рисунок 11.18 – Часова діаграма функціонування паралельно-послідовного регістра на D-тригері

11.3 Вимоги до змісту звіту

Звіт про виконання лабораторної роботи повинен містити:

1. Номер і тему роботи.
2. Мету виконання роботи.
3. Короткі теоретичні відомості.
4. Порядок виконання роботи.
5. Результати виконання роботи у вигляді:

– результат виконання пунктів 11.2.1 – 11.2.3, в яких представлено схеми електричні принципові та часові діаграми роботи синтезованих цифрових схем із необхідними текстовими поясненнями;

– результати виконання пунктів 11.2.4 – 11.2.7, в яких представлено схеми електричні принципові та часові діаграми роботи синтезованих лічильників із необхідними текстовими поясненнями;

– результат виконання пунктів 11.2.8 – 11.2.10, в яких представлено схему електричну принципову та часову діаграму роботи синтезованого регістра із необхідними текстовими поясненнями.

6. Особливості функціонування САПР Intel Quartus Prime, виявлені під час виконання роботи.

7. Висновки.

11.4 Контрольні питання для перевірки знань

11.5.1 Складіть таблиці переходів для RS-, RSR-, RSS- та RSE-тригерів.

11.5.2 Складіть таблиці переходів для D- та DV-тригерів.

11.5.3 Складіть таблиці переходів для T- та JK-тригерів.

11.5.4 Поясніть, у чому саме полягає різниця між синхронними та асинхронними тригерами.

11.5.5 Поясніть, у чому саме полягає різниця між синхронними статичними та динамічними тригерами.

11.5.6 Поясніть, у чому полягає відмінність між схемами організації переносу в лічильниках.

11.5.7 Поясніть, у чому полягає відмінність між синхронними і асинхронними лічильниками.

11.5.8 Поясніть, у чому полягає різниця між лічильниками, що додають, що віднімають і реверсивними. Вкажіть шляхи, яким чином можна змінити один напрямок лічби у інший.

11.5.9 Наведіть класифікації регістрів за способом запису і читання інформації.

11.5.10 Поясніть, у чому полягає відмінність між зсувними, реверсивними і кільцевими регістрами.

11.5.11 Поясніть, яким чином можна перетворити D-тригер на T-тригер або на JK-тригер.

11.5.12 Поясніть, яким чином можна перетворити JK-тригер на D- або T-тригер.

12 ЛАБОРАТОРНА РОБОТА № 4
ПРОЕКТУВАННЯ ТА ДОСЛІДЖЕННЯ СИНХРОННИХ ЦИФРОВИХ АВТОМАТІВ

Мета роботи: вивчити принципи функціонування синхронних цифрових автоматів Мілі і Мура, а також методи їх абстрактного та структурного синтезу.

12.1 Короткі теоретичні відомості

12.1.1 Мінімізація числа станів цифрового автомата

Для визначення абстрактного автомата необхідно описати всі елементи множини $S = \{A, X, Y, a_0, \delta, \lambda\}$, тобто:

– алфавіт станів, в яких може знаходитися автомат $A = \{a0, ..., a_k\}$;

– початковий стан автомата a_0;

– вхідний алфавіт $X = \{x_1, ..., x_n\}$;

– вихідний алфавіт $Y = \{y_1, ..., y_m\}$;

– функцію переходів $\delta = f(X, A)$;

– функцію виходів $\lambda = f(A)$ для автомата Мура або $\lambda = f(X, A)$ для автомата Мілі.

Для опису абстрактного автомата найчастіше використовують табличний або графічний методи. На цьому завершується перший етап синтезу цифрового автомата (ЦА).

Однак, ЦА може містити зайві внутрішні стани, тому необхідно провести мінімізацію числа станів автомата.

Основна ідея одного із найпоширеніших методів мінімізації полягає у розбитті всіх станів вихідного автомата на попарно не пересічні класи еквівалентних станів та заміні кожного класу еквівалентності одним станом. Утворений в результаті таких дій мінімальний ЦА, має стільки ж внутрішніх станів, на скільки класів еквівалентності розбиваються стани вихідного автомата.

Два стани a_m та a_s називаються еквівалентними ($a_m \equiv a_s$), якщо $(a_m, \varepsilon) = (a_s, \varepsilon)$ для всіх можливих вхідних слів ε. Якщо a_m і a_s не є еквівалентними, вони є такими, що розрізняються.

Введемо поняття k еквівалентності. Два стани a_m і a_s є k-еквівалентними, якщо $\lambda (a_m, \varepsilon_k) = \lambda (a_s, \varepsilon_k)$ для всіх можливих вхідних слів довжиною k, якщо стани не k-еквівалентні, вони є k-розрізненими. Наведені відношення еквівалентності та k-еквівалентності використовуються для розбиття множини станів автомата на класи, що попарно не перетинаються (класи еквівалентності). Відповідні розбиття на класи еквівалентності та k-еквівалентності, зазвичай, позначаються як π та π_k. Саме розбиття π і дозволяє визначити надлишкові елементи у множині станів автомата.

Мінімізація числа станів автомата складається з наступних кроків.

1. Знаходяться послідовні розбиття π_1, π_2, ..., π_k, π_{k+1} множини станів на класи до тих пір, поки на якомусь k+1 кроці не виявиться, що $\pi_{k+1} = \pi_k$.В цьому випадку $\pi_k = \pi$, тобто, k-еквівалентні стани є у цьому випадку еквівалентними.

2. У кожному класі еквівалентності розбиття π вибирається по одному елементу, які утворюють нову множину станів мінімального автомата.

3. Для визначення функцій переходів та виходів викреслюються рядки, що не ввійшли в множину станів мінімального автомата. А в рядках таблиці переходів, що залишилися, всі стани замінюються на еквівалентні з нової мінімальної множини станів автомата.

12.1.2 Канонічний метод структурного синтезу

Канонічний метод структурного синтезу дозволяє звести завдання синтезу довільного ЦА до завдання синтезу комбінаційних схем. Цей метод синтезу можна умовно поділити на такі послідовні етапи:

– кодування;

– вибір елементів пам’яті;

– побудову булевих функцій збудження елементів пам'яті;

– побудову булевих функцій виходів автомата;

– побудову функціональної схеми автомата.

При переході на структурний рівень синтезу кожна літера x вхідного алфавіта X, кожна літера y вихідного алфавіта Y, кожна літера a_i алфавіту станів A перетворюється на деякий двійковий вектор.

Кількість сигналів, необхідних, наприклад, для кодування внутрішніх станів автомата, можна визначити за формулою:

$$k_c \geq \log_2 |A|, \qquad (12.1)$$

де $|A|$ – потужність (кількість елементів, що утворюють множину) алфавіту станів автомата, наприклад, якщо $A = \{a_1, a_2, a_3\}$, то $|A| = 3$.

Таким чином, якщо ЦА має 3 стани, то $k \geq 2$. Іншими словами, кожну букву a_i із множини A можна закодувати двійковим вектором, що складається не менше, ніж із двох компонентів, наприклад, A = {00, 01, 10}.

При канонічному методі структурного синтезу ЦА в якості елементів пам’яті використовуються елементарні автомати Мура із двома стійкими станами або іншими словами – тригери. Як такі елементарні автомати зазвичай використовують синхронні динамічні D-, T-, RS- та JK-тригери.

Очевидно, що кількість тригерів (елементів пам’яті) в точності дорівнює кількості компонентів, що описують вектор його станів.

Під час структурного синтезу дуже зручно мати спеціальну таблицю, в якій описано, яке значення повинен приймати сигнал збудження, щоб відповідний тригер перейшов із стану Q^s в Q^{s+1}. Такі дані представлено нижче, в таблиці 12.1.

Таблиця 12.1 - Підграфи переходів тригерів

Qs	Qs+1	D	T	R	S	J	K
0	0	0	0	—	0	0	—
0	1	1	1	0	1	1	—
1	0	0	1	1	0	—	1
1	1	1	0	0		—	0

12.1.3 Особливості процесу кодування станів ЦА

Різні варіанти кодування станів автомата призводять до різних виразів функцій виходів, внаслідок чого виявляється, що складність комбінаційних схем автомата істотно залежить від обраного методу кодування.

Розглянемо приклад застосування методу евристичного кодування станів автомата, що дозволяє мінімізувати комбінаційну схему автомата.

З цією метою візьмемо автомат, граф якого має 5 станів (рисунок 12.1).

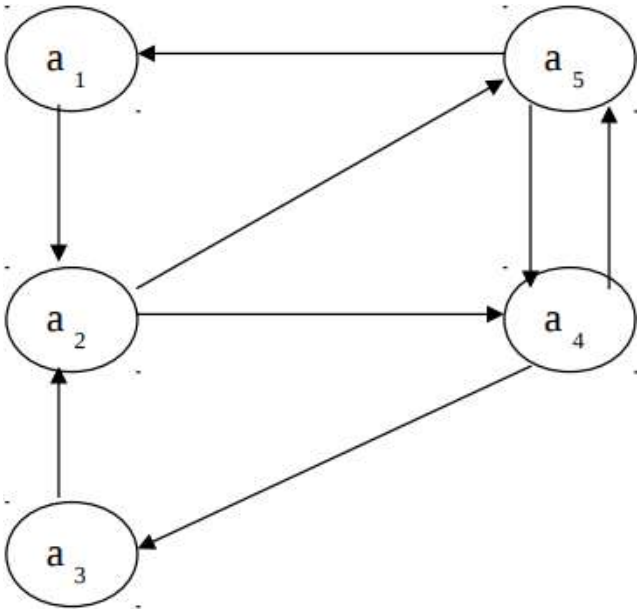


Рисунок 12.1 – Приклад графа ЦА

Алгоритм складається з наступних кроків:

1. Побудуємо матрицю, що складається з усіх різних пар номерів станів, таких, що в ЦА є переходи для пар. Матриця будується таким чином, щоб хоча б один з елементів і-го рядка містився б у якомусь із попередніх рядків.

$$M = \begin{vmatrix} 1 & 2 \\ 2 & 4 \\ 2 & 5 \\ 3 & 2 \\ 4 & 3 \\ 4 & 5 \\ 5 & 4 \\ 5 & 1 \end{vmatrix}$$

2.Закодуємо стани із першого рядка матриці наступним чином: $a_1 = 000$; $a_2 = 001$. Кодування будемо ілюструвати картою Карно (рисунок 12.2).

$Q_1 \backslash Q_2Q_3$	00	01	11	10
0	1	2		
1				

Рисунок 12.2 – Кодування початкових станів матриці M

3. Викреслимо із матриці M перший рядок, що відповідає закодованим станам a_1 і a_2. Отримаємо матрицю M' виду:

$$M' = \begin{vmatrix} 2 & 4 \\ 2 & 5 \\ 3 & 2 \\ 4 & 3 \\ 4 & 5 \\ 5 & 4 \\ 5 & 1 \end{vmatrix}$$

4. У початковому рядку матриці M' закодовано один елемент. Виберемо із першого рядка M' незакодований елемент і позначимо його v, тобто, v = 4.

5. Побудуємо матрицю M_4, вибравши з матриці M' рядки, що містять 4. B_4 – це множина елементів матриці M', які вже закодовані. У нашому випадку закодований тільки другий елемент.

$$M_4 = \begin{vmatrix} 2 & 4 \\ 4 & 3 \\ 4 & 5 \\ 5 & 4 \end{vmatrix}$$

6. Для кожного стану знайдемо множину кодів, сусідніх із кодом стану і ще незайнятих для кодування станів автомата: $C'_2 = \{101, 011\}$. Таким чином, множина вільних кодів, які можуть використовуватися для кодування стану 4: $C'_2 = \{101, 011\}$.

7. Для вибору коду вводимо вагову функцію, що характеризує відстань між кодами станів (відстань Хеммінга), що дорівнює кількості елементів пам'яті, що змінюють свій стан при переході:

$$W_{101} = |101 - 001| = 1;$$
$$W_{011} = |011 - 001| = 1.$$

Вибираємо довільним чином, що $a_4 = 101$ та відмічаємо у карті Карно (рисунок 12.3).

$Q_1 \backslash Q_2Q_3$	00	01	11	10
0	1	2		
1		4		

Рисунок 12.3 – Кодування третього стану ЦА

8.Викреслимо із матриці M' 1-й рядок, тоді вона матиме вигляд:

$$M'' = \begin{vmatrix} 2 & 5 \\ 3 & 2 \\ 4 & 3 \\ 4 & 5 \\ 5 & 4 \\ 5 & 1 \end{vmatrix}$$

9. Виберемо із першого рядка незакодований елемент та позначимо його v = 5.

10. Побудуємо матрицю M_5, вибравши з M" рядки, що містять стан 5.

$$M_5 = \begin{vmatrix} 2 & 5 \\ 4 & 5 \\ 5 & 4 \\ 5 & 1 \end{vmatrix}$$

Множина елементів з матриці, які вже закодовано $B_5 = \{2, 4, 1\}$.

11. Для кожного стану з множини B_5 знайдемо сусідні коди, що ще не використовуються для кодування: $C'_2 = \{011\}$, $C'_4 = \{100, 111\}$, $C'_1 = \{100, 010\}$. Таким чином, множина вільних кодів, яка може використовуватися для кодування стану 5: $D'_5 = \{011, 100, 111, 010\}$.

12.Визначимо вагову функцію для кожного коду

$W_{011} = |011 - 001| + |011 - 101| + |011 - 101| + |011 - 000| = 1+2+2+2 = 7,$

$W_{100} = |100 - 001| + |100 - 101| + |100 - 101| + |100 - 000| = 2+1+1+1 = 5;$

$W_{111} = |111 - 001| + |111 - 101| + |111 - 101| + |111 - 000| = 2+1+1+3 = 7;$

$W_{010} = |010 - 001| + |010 - 101| + |010 - 101| + |010 - 000| = 2+3+3+1 = 9.$

Таким чином, $W_{100} = \min \{W_{011}, W_{100}, W_{111}, W_{010}\} = 5$. Відмітимо у карті Карно (рисунок 12.4) результат кодування.

$Q_1 \backslash Q_2Q_3$	00	01	11	10
0	1	2		
1	5	4		

Рисунок 12.4 – Кодування четвертого стану ЦА

При розрахунку вагових функцій враховувалися ті стани, з якими пов'язаний стан 5 (дивися граф переходів на рисунку 12.1). Стан 4 враховувався двічі, оскільки вони зв’язані двома дугами.

13. Викреслимо з матриці M" перший рядок і побудуємо нову матрицю M"', в яку не входять рядки, де стани вже закодовані.

$$M''' = \begin{vmatrix} 3 & 4 \\ 2 & 3 \end{vmatrix}$$

14. Виберемо із першого рядка незакодований елемент v = 3.

15. Побудуємо матрицю M_3, вибравши із M"' рядки, що містять 3.

$$M_3 = \begin{vmatrix} 3 & 4 \\ 2 & 3 \end{vmatrix}$$

16. Множина елементів, які вже закодовано, $B_3 = \{2,4\}$.

17.Знайдемо сусідні коди для кожного стану: $C'_2 = \{011\}$, $C'_4 = \{111\}$. Таким чином, множина вільних кодів, які можуть використовуватися для кодування стану 3: $D'_3 = \{011, 111\}$.

18. За допомогою вагової функції вибираємо код для аналізованого стану:

$$W_{011} = \begin{vmatrix} 011 - 001 \end{vmatrix} + \begin{vmatrix} 011 - 101 \end{vmatrix} = 1+2 = 3;$$
$$W_{111} = \begin{vmatrix} 111 - 001 \end{vmatrix} + \begin{vmatrix} 111 - 101 \end{vmatrix} = 2+1 = 3.$$

Вибираємо $a_3 = 011$ та відмічаємо в карті Карно (рисунок 12.5).

$Q_1 \backslash Q_2Q_3$	00	01	11	10
0	1	2	3	
1	5	4		

Рисунок 12.5 – Кодування останнього стану ЦА

Таким чином, в результаті виконання усіх зазначених вище кроків і було здійснено кодування станів ЦА.

12.2 Порядок виконання роботи

12.2.1 Синтезувати структурну схему синхронного ЦА Мура (без накладання) згідно варіанту завдання, наведеного в таблиці 12.2.

Таблиця 12.2 – Варіанти завдання №1

№ варіанту завдання	Послідовність бітів, яку треба відшукати у вхідному потоці	Тип тригера (примітива) для реалізації блоку пам’яті
1	00001	D (dff)
2	00010	T (tff)
3	00011	JK (jkff)
4	00100	RS (srff)
5	00101	D (dff)
6	00110	T (tff)
7	00111	JK (jkff)
8	01000	RS (srff)
9	01001	D (dff)
10	01010	T (tff)
11	01011	JK (jkff)
12	01100	RS (srff)
13	01101	D (dff)
14	01110	T (tff)
15	01111	JK (jkff)
16	10000	RS (srff)

Продовження таблиці 12.2

№ варіанту завдання	Послідовність бітів, яку треба відшукати у вхідному потоці	Тип тригера (примітива) для реалізації блоку пам'яті
17	10001	D (dff)
18	10010	T (tff)
19	10011	JK (jkff)
20	10100	RS (srff)
21	10110	D (dff)
22	10111	T (tff)
23	11000	JK (jkff)
24	11001	RS (srff)
25	11010	D (dff)
26	11011	T (tff)
27	11100	JK (jkff)
28	11101	RS (srff)
29	11110	D (dff)
30	11111	T (tff)
Примітка. Номер варіанту завдання обирається у відповідності до порядкового номера студента згідно списку у журналі академічної групи.		

12.2.2 Побудувати часові діаграми функціонування отриманої схеми синтезованого ЦА. Правильно побудованою часовою діаграмою вважається така, за допомогою якої можна перевірити, що ЦА точно знаходить задану послідовність, а також не реагує на будь-яку із 5 послідовностей, кожна із яких відрізняється від шуканої лише у одному біті.

12.2.3 Синтезувати структурну схему синхронного ЦА Мілі (без накладання) згідно варіанту завдання, наведеного в таблиці 12.3.

Таблиця 12.3 – Варіанти завдання №2

№ варіанту завдання	Послідовність бітів, яку треба відшукати у вхідному потоці	Тип тригера (примітива) для реалізації блоку пам'яті
1	00001	JK (jkff)
2	00010	RS (srff)
3	00011	D (dff)
4	00100	T (tff)
5	00101	JK (jkff)
6	00110	RS (srff)
7	00111	D (dff)
8	01000	T (tff)
9	01001	JK (jkff)
10	01010	RS (srff)
11	01011	D (dff)

Продовження таблиці 12.3

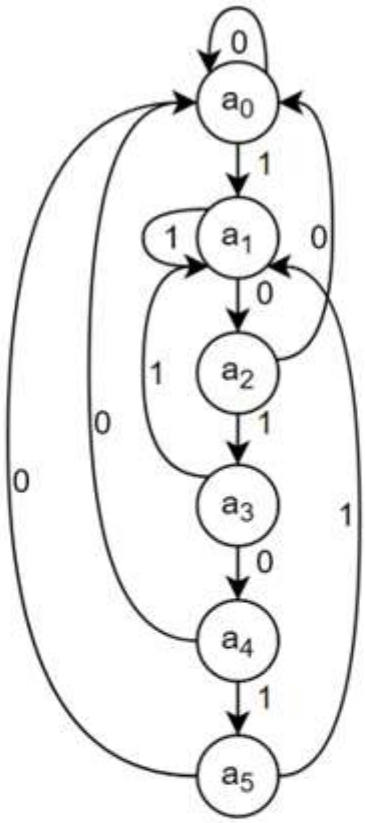
12	01100	T (tff)
13	01101	JK (jkff)
14	01110	RS (srff)
15	01111	D (dff)
16	10000	T (tff)
17	10001	JK (jkff)
18	10010	RS (srff)
19	10011	D (dff)
20	10100	T (tff)
21	10110	JK (jkff)
22	10111	RS (srff)
23	11000	D (dff)
24	11001	T (tff)
25	11010	JK (jkff)
26	11011	RS (srff)
27	11100	D (dff)
28	11101	T (tff)
29	11110	JK (jkff)
30	11111	RS (srff)
Примітка. Номер варіанту завдання обирається у відповідності до порядкового номера студента згідно списку у журналі академічної групи.		

12.3 Приклад синтезу синхронних цифрових автоматів Мура і Мілі

В якості прикладу розглянемо послідовність синтезу синхронного цифрового автомата Мура (без накладання), який шукає послідовність 10101_2 у вхідному потоці бітів, а його блок пам’яті будується на JK-тригерах. Опис абстрактного ЦА Мура у вигляді графа переходів і таблиць переходів і виходів, а також усі кроки канонічного методу структурного синтезу показано нижче, на рисунках 12.6 – 12.14.

Синтез автомата Мура (без накладання) для пошуку послідовної 10101_2

1. Будуємо граф ЦА.



2. Складемо таблиці переходів і виходів ЦА Мура по відомому графу переходів

Поточний стан	Вхідний сигнал	
	x = 0	x = 1
a_0	a_0	a_1
a_1	a_2	a_1
a_2	a_0	a_3
a_3	a_4	a_1
a_4	a_0	a_5
a_5	a_0	a_1

Поточний стан	Вихідний сигнал y
a_0	0
a_1	0
a_2	0
a_3	0
a_4	0
a_5	1

Рисунок 12.6 – Граф переходів, таблиці переходів і виходів абстрактного ЦА Мура

3. Визначаємо кількість сигналів, необхідних для кодування внутрішніх станів автомата A = {a_0, a_1, a_2, a_3, a_4, a_5}; тоді $|A| = 6$; тоді $K \geq \log_2|A|$; $K \geq \log_2 6$; $K \geq 2{,}585$; $\boxed{K \geq 3}$

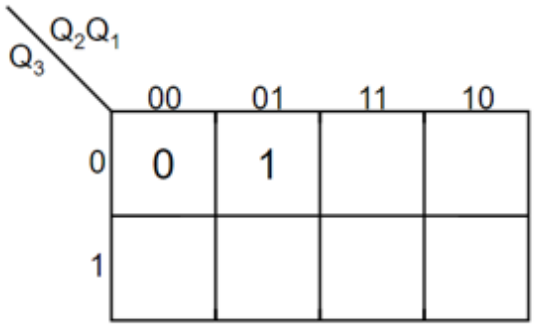
4. Застосуємо евристичний алгоритм кодування станів ЦА:

$$M = \begin{vmatrix} 0 & 1 \\ 2 & 1 \\ 0 & 3 \\ 4 & 1 \\ 0 & 5 \\ 0 & 1 \end{vmatrix}$$

а) будуємо матрицю, що складається із усіх різних пар номерів станів

б) довільним чином закодуємо обидва стани, що написані у верхньому рядку матриці, тобто $a_0 = 000$ і $a_1 = 001$;

Побудуємо карту Карно:



в) викреслюємо із початкової матриці M перший рядок із закодованими станами, отримуємо матрицю M' виду:

Рисунок 12.7 – Розрахунок кількості розрядів для кодування станів ЦА Мура і застосування евристичного алгоритму кодування (1)

$$M' = \begin{vmatrix} 2 & 1 \\ 0 & 3 \\ 4 & 1 \\ 0 & 5 \\ 0 & 1 \end{vmatrix}$$

г) виберемо із першого рядка M' незакодований елемент 2 і позначимо його як V = 2;

д) побудуємо матрицю M_2, вибравши із M' усі рядки, що містять стан 2, отримаємо

$M = |2 \quad 1|$

е) для кожного стану знайдемо множину кодів, що є його сусідами, але ще не зайняті для кодування, тобто $C_2' = \{101,\ 011\}$;

є) для визначення коду будемо використовувати відстань Хеммінга, тоді отримаємо:

$W_{101} = |101 - 001| = 1$; оскільки відстань однакова;

$W_{011} = |011 - 001| = 1$; довільним чином вибираємо, що $a_2 = 101$;

Відмітимо це карті Карно:

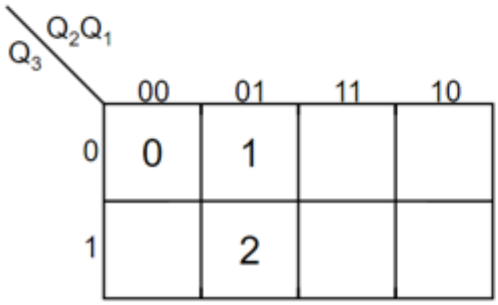


Рисунок 12.8 – Застосування евристичного алгоритму кодування станів ЦА Мура (2)

ж) викреслюємо із матриці M' закодований рядок, отримаємо:

$$M'' = \begin{vmatrix} 0 & 3 \\ 4 & 1 \\ 0 & 5 \\ 0 & 1 \end{vmatrix}$$

з) повторюємо усю послідовність дій, розглянутих у пунктах г) – є) раніше, тобто:

$M_3 = |0 \quad 3|$ v = 3;

$C_3' = \{010,\ 100\}$;

$W_{010} = |010 - 000| = 1$; оскільки відстань однакова;

$W_{100} = |100 - 000| = 1$; довільним чином вибираємо, що $a_3 = 100$;

Q_3 \ Q_2Q_1	00	01	11	10
0	0	1		
1	3	2		

і) викреслюємо із матриці M'' закодований рядок, отримаємо:

$$M''' = \begin{vmatrix} 4 & 1 \\ 0 & 5 \\ 0 & 1 \end{vmatrix}$$

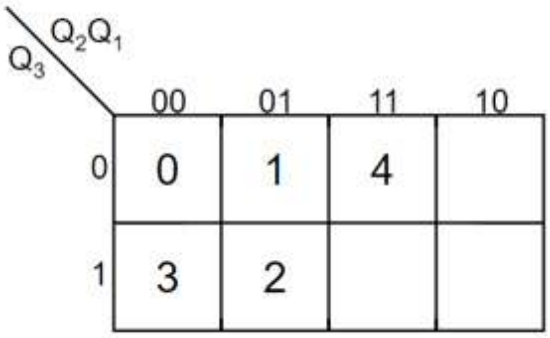
к) виберемо із першого рядка матриці M'''

Рисунок 12.9 – Застосування евристичного алгоритму кодування станів ЦА Мура (3)

незакодований елемент і позначимо його як v = 4;

л) побудуємо матрицю M_4, вибравши із M''' усі рядки, що містять стан 4, отримаємо: $M_4 = |4 \quad 1|$

м) як бачимо із карт Карно, стан 1 має 2 зайнятих сусіди із трьох можливих, тому $a_4 = 011$;



н)

$$M'''' = \begin{vmatrix} 4 & 1 \\ 0 & 5 \\ 0 & 1 \end{vmatrix} \quad v = 5 \quad M_5 = |0 \quad 5|$$

як бачимо із карт Карно, стан 0 має 2 зайняті сусіди із трьох можливих, тому a_5=010;

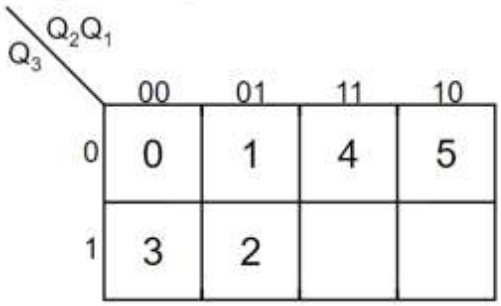


Рисунок 12.10 – Застосування евристичного алгоритму кодування станів ЦА Мура (4)

В результаті евристичного кодування станів ЦА було отримано наступні результати:

$a_0 = 000$;

$a_1 = 001$;

$a_2 = 101$;

$a_3 = 100$;

$a_4 = 011$;

$a_5 = 010$;

Перепишемо таблиці переходів і виходів в закодованому вигляді:

Стан	X = 0	X =1
0 0 0	0 0 0	0 0 1
0 0 1	1 0 1	0 0 1
1 0 1	0 0 0	1 0 0
1 0 0	0 1 1	0 0 1
0 1 1	0 0 0	0 1 0
0 1 0	0 0 0	0 0 1
$Q_3Q_2Q_1$	$Q_3Q_2Q_1$	$Q_3Q_2Q_1$

Стан	y
000	0
001	0
101	0
100	0
011	0
010	1

5. В якості прикладу будемо синтезувати ЦА Мура, БП якого будується на ІК = тригерах.

Виведемо підграф його переходів:

I	K	Q^S	Q^{S+1}	
0	0	0	0	I)
0	0	1	1	IV)
0	1	0	0	I)
0	1	1	0	III)
1	0	0	1	II)
1	0	1	1	IV)
1	1	0	1	II)
1	1	1	0	III)

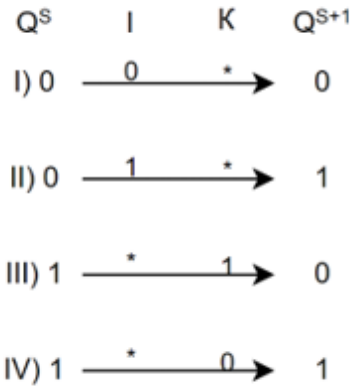
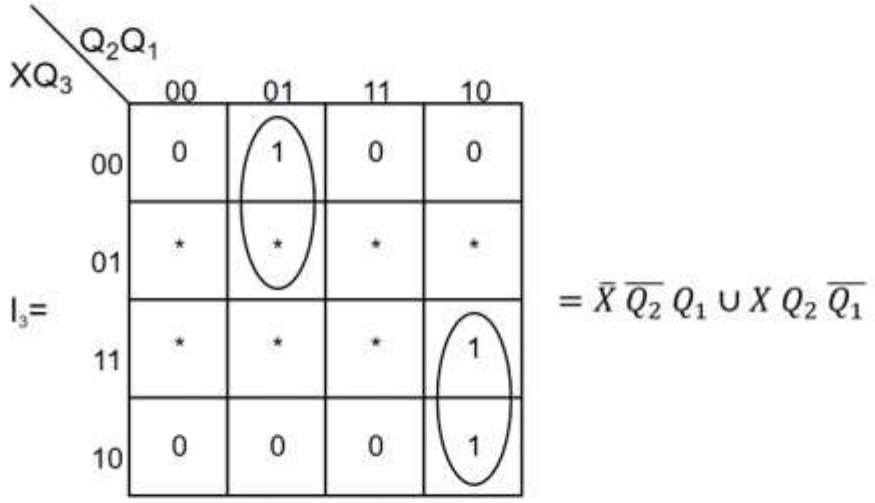


Рисунок 12.11 – Синтез функцій збудження для блоку пам’яті ЦА Мура (1)

6. Синтезуємо КС1 на основі закодованої таблиці переходів:

Стан	X = 0						X = 1					
000	0	*	0	*	0	*	0	*	0	*	1	*
001	1	*	0	*	*	0	0	*	0	*	*	0
101	*	1	0	*	*	1	*	0	0	*	*	1
100	*	1	1	*	1	*	*	1	0	*	1	*
011	0	*	*	1	*	1	0	*	*	0	*	1
010	0	*	*	1	0	*	0	*	*	1	1	*
$Q_3Q_2Q_1$	I3	K3	I2	K2	I1	K1	I3	K3	I2	K2	I1	K1

Отримаємо карти Карно для функцій керування кожним із тригерів окремо:



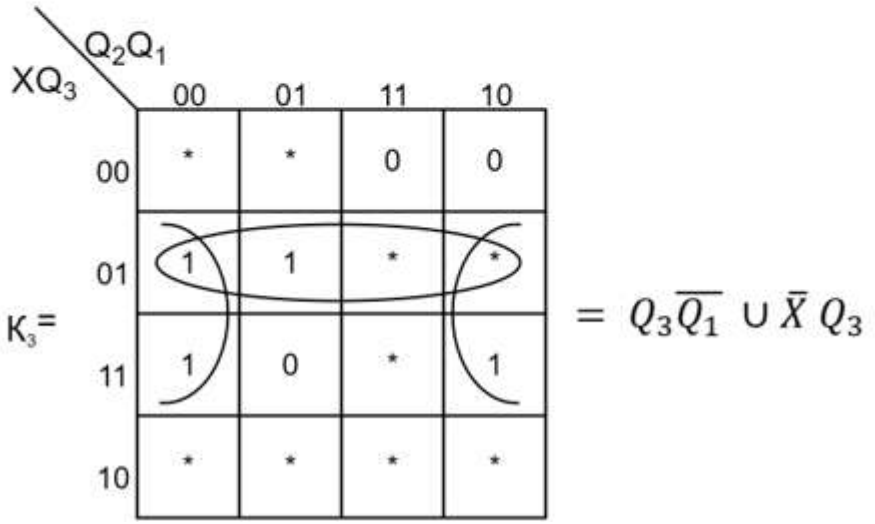


Рисунок 12.12 – Синтез функцій збудження для блоку пам’яті ЦА Мура (2)

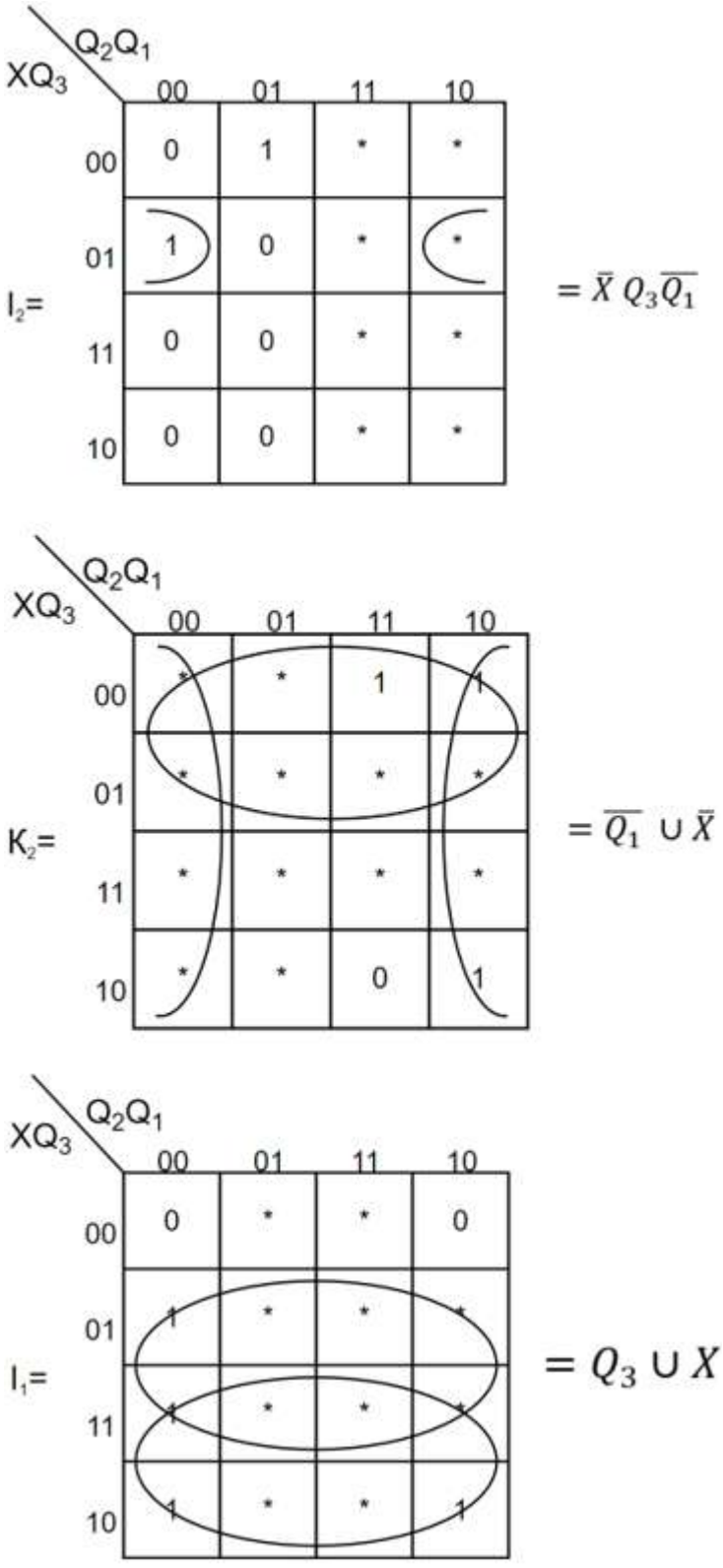
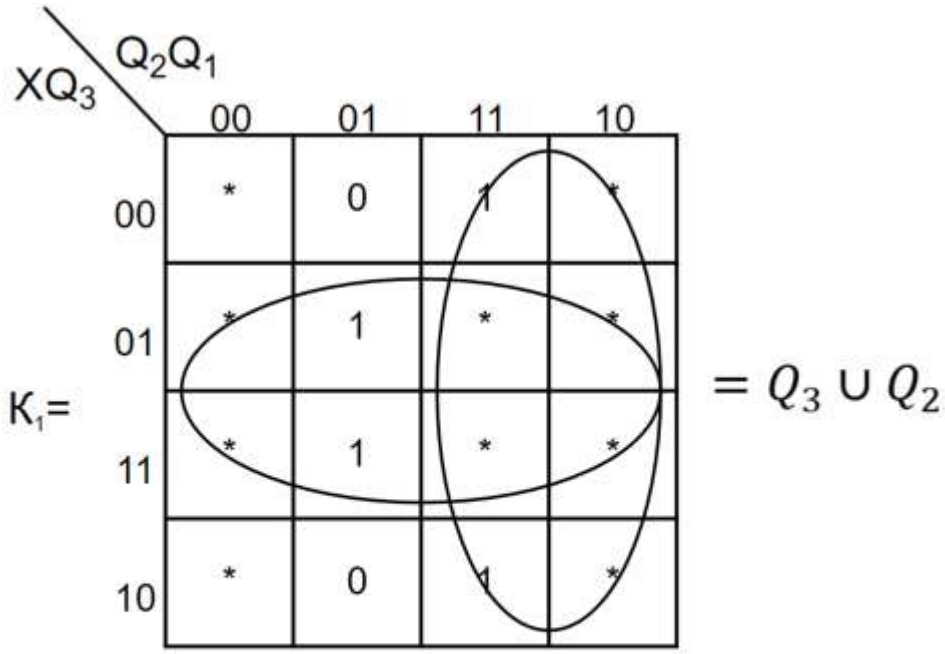


Рисунок 12.13 – Синтез функцій збудження для блоку пам’яті ЦА Мура (3)



7. Оскільки ЦА Мура повертає 1 на виході, тоді і тільки тоді, коли знаходиться у стані a_5, то його функція виходів є очевидною: $y = \overline{Q_3}Q_2\overline{Q_1}$ (бо a_5=010);

8. Тепер можна переходити до збірки схеми у САПР, отриманню та аналізу часових діаграм.

Рисунок 12.14 – Синтез функції виходів ЦА Мура

Результат створення структурного ЦА Мура засобами САПР Quartus Prime показано нижче, на рисунку 12.15, а часова діаграма його функціонування – на рисунку 12.16. Із аналізу часової діаграми видно, що синтезований синхронний ЦА Мура знаходить необхідну послідовність у вхідному потоці, про що свідчить логічна “1” на виході Y при досягненні кінцевого стану.

Опис абстрактного ЦА Мілі у вигляді графа переходів і суміщеної таблиці переходів-виходів, а також усі кроки канонічного методу структурного синтезу показано нижче, на рисунках 12.17 – 12.19.

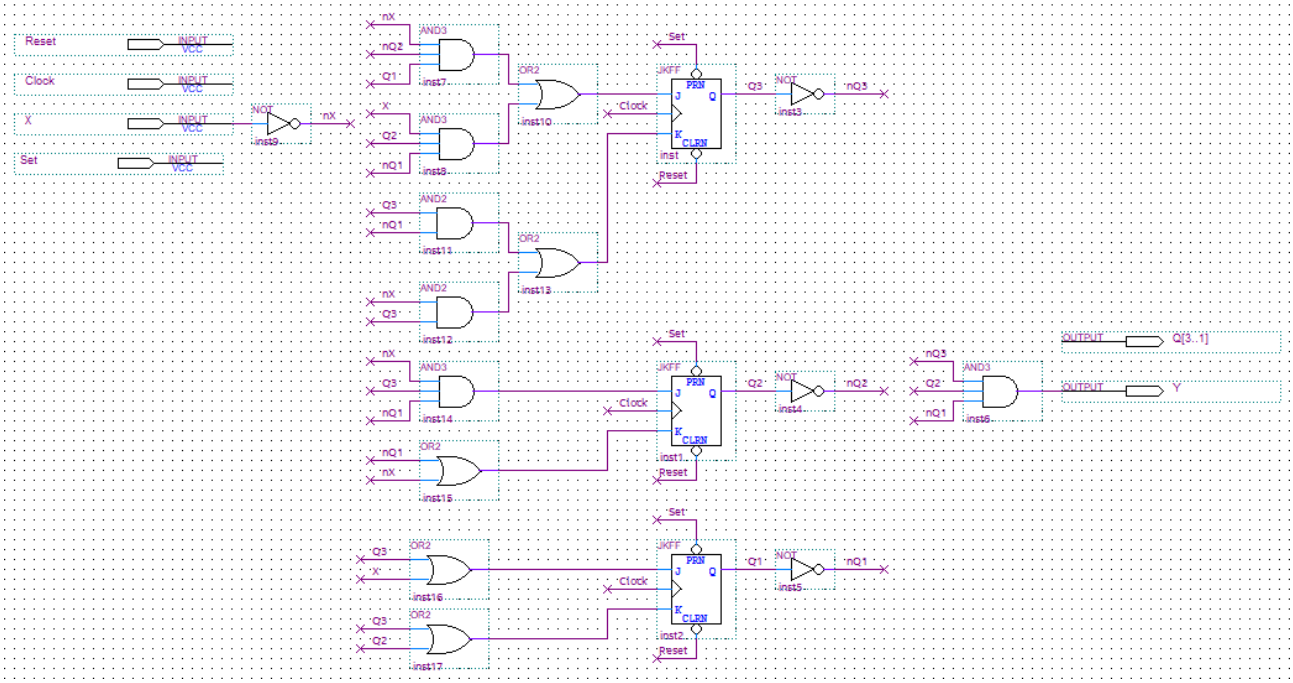


Рисунок 12.15 – Схема синхронного ЦА Мура без накладання

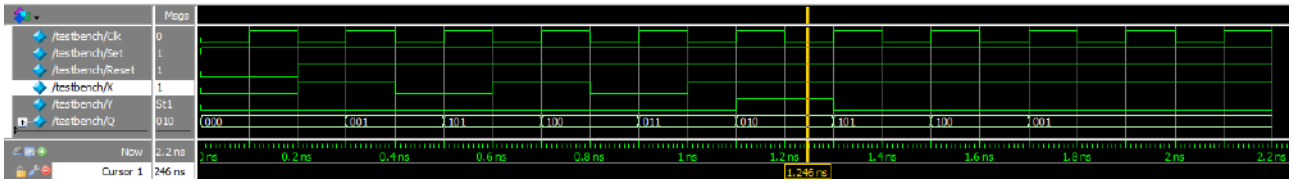
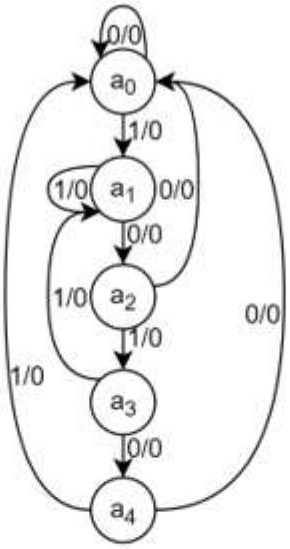


Рисунок 12.16 – Часова діаграма функціонування синтезованого синхронного ЦА Мура без накладання

Синтез автомата Мілі (без накладання) для пошуку послідовності 10101_2.

1. Будуємо граф ЦА.



2. Складемо суміщену таблицю переходів-виходів по відомому графу переходів.

Поточний стан	Вхідний сигнал	
	X = 0	X = 1
a_0	$a_0/0$	$a_1/0$
a_1	$a_2/0$	$a_1/0$
a_2	$a_0/0$	$a_3/0$
a_3	$a_4/0$	$a_1/0$
a_4	$a_0/0$	$a_0/1$

3. Визначаємо кількість сигналів, необхідних для кодування внутрішніх станів автомата: $A = \{ a_0, a_1, a_2, a_3, a_4\}$, тоді $|A| = 5$, тоді:

Рисунок 12.17 – Граф переходів, таблиця переходів-виходів абстрактного ЦА Мілі

$K \geq \log_2|A|; \qquad K \geq \log_2 5; \qquad K \geq 2{,}322; \qquad \boxed{K \geq 3}$

4. Оскільки кількість розрядів (тригерів у блоці пам'яті) не змінилася, немає сенсу застосовувати евристичний алгоритм кодування, тобто:

$a_0 = 000;$

$a_1 = 001;$

$a_2 = 101;$

$a_3 = 100;$

$a_4 = 011;$

Перепишемо таблицю переходів-виходів у закодованому вигляді:

Стан	X = 0	X = 1	X = 0						X = 1					
000	000	001	0	*	0	*	0	*	0	*	0	*	1	*
001	101	001	1	*	0	*	*	0	0	*	0	*	*	0
101	000	100	*	1	0	*	*	1	*	0	0	*	*	1
100	011	001	*	1	1	*	1	*	*	1	0	*	1	*
011	000	000	0	*	*	1	*	1	0	*	*	1	*	1
$Q_3Q_2Q_1$	$Q_3Q_2Q_1$	$Q_3Q_2Q_1$	I3	K3	I2	K2	I1	K1	I3	K3	I2	K2	I1	K1

Рисунок 12.18 – Розрахунок кількості розрядів для кодування станів ЦА Мілі і синтез функцій збудження для блоку пам'яті

5. Оскільки таблиця станів ЦА Мілі скоротилася відносно аналогічної для ЦА Мура, треба виконувати синтез БП повторно:

6. Оскільки ЦА Мілі повертає 1 на виході тоді і тільки тоді, коли знаходиться у переході між станами, його функція виходів не є очевидною, її треба синтезувати на основі суміщеної таблиці:

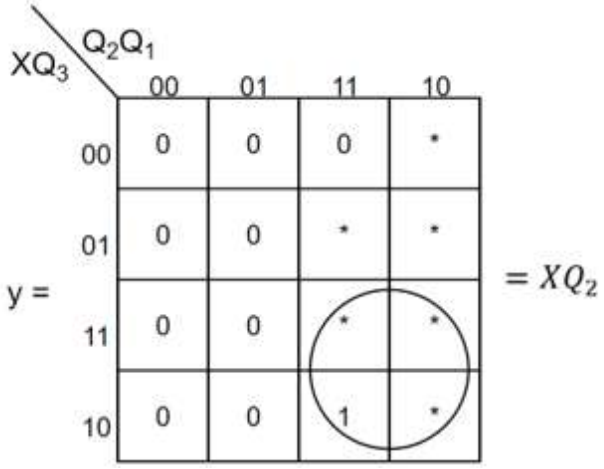


Рисунок 12.19 – Синтез функцій збудження для блоку пам'яті та функцій виходів ЦА Мілі

Результат створення структурного ЦА Мілі засобами САПР Quartus Prime показано нижче, на рисунку 12.20, а часова діаграма його функціонування – на рисунку 12.21. Із аналізу часової діаграми видно, що синтезований синхронний ЦА Мілі знаходить необхідну послідовність у вхідному потоці, про що свідчить логічна "1" на виході Y при переході від кінцевого стану у початковий.

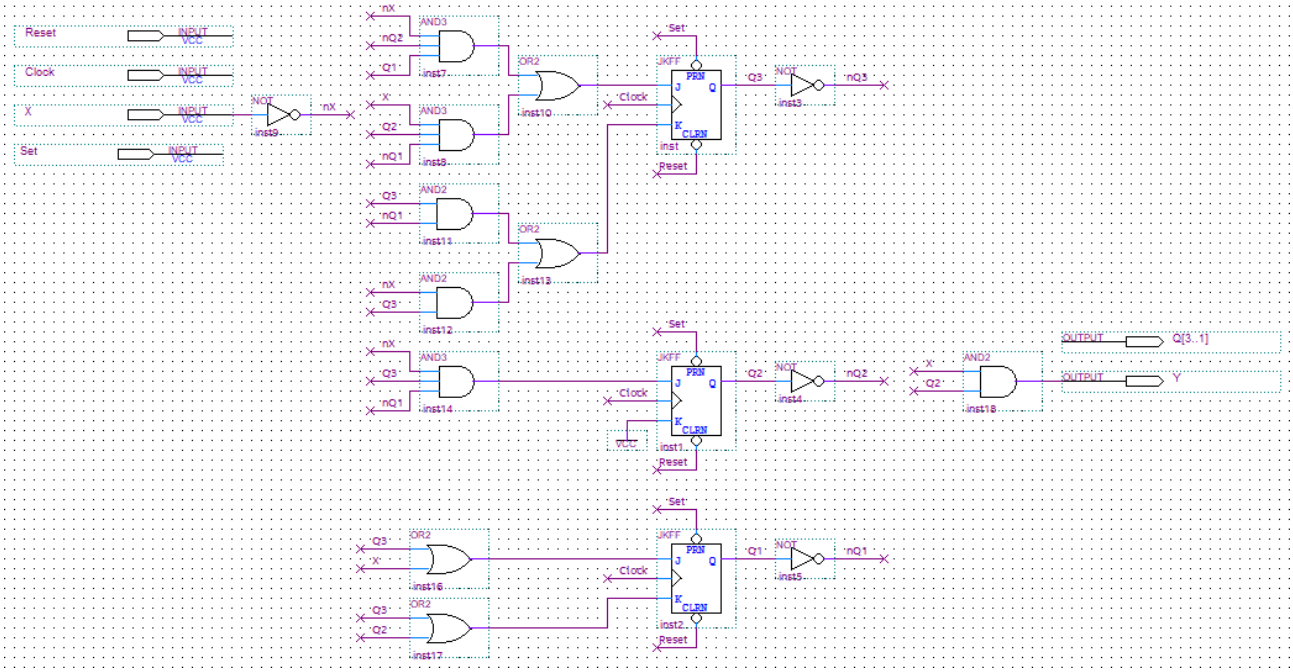


Рисунок 12.20 – Схема синхронного ЦА Мілі без накладання

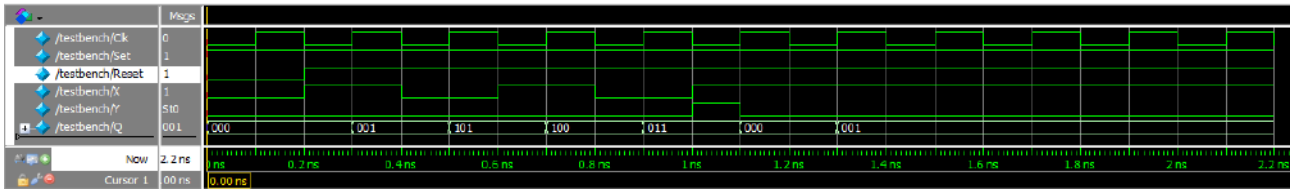


Рисунок 12.21 – Часова діаграма функціонування синтезованого синхронного ЦА Мілі без накладання

12.4 Вимоги до змісту звіту

Звіт про виконання лабораторної роботи повинен містити:

1. Номер і тему роботи.
2. Мету виконання роботи.
3. Короткі теоретичні відомості.
4. Порядок виконання роботи.
5. Результати виконання роботи у вигляді:

– результати виконання пунктів 12.2.1 і 12.2.2, в яких представлено висхідний граф переходів ЦА Мура, таблиці переходів і виходів, застосування канонічного методу структурного синтезу ЦА, схему електричну принципові та часові діаграми роботи синтезованого ЦА при різних комбінаціях вхідних сигналів;

– результати виконання пунктів 12.2.3 і 12.2.4, в яких представлено висхідний граф переходів ЦА Мілі, таблицю переходів-виходів, застосування канонічного методу структурного синтезу ЦА, схему електричну принципові та часові діаграми роботи синтезованого ЦА при різних комбінаціях вхідних сигналів.

6. Особливості функціонування САПР Intel Quartus Prime, виявлені під час виконання роботи.
7. Висновки.

12.5 Контрольні питання для перевірки знань

12.5.1 Опишіть процедуру мінімізації числа станів автомата.
12.5.2 Покажіть ефективність раціонального кодування станів.
12.5.3 В чому полягає важлива відмінність автоматів Мура та Мілі?
12.5.4 Поясніть рівняння, що описують роботу синхронних автоматів Мура та Мілі.
12.5.5 Назвіть етапи при проектуванні синхронних цифрових автоматів.
12.5.6 Що називається відстанню Хеммінга і яким чином вона розраховується?
12.5.7 В чому полягає різниця в зображеннях графів ЦА Мура і Мілі?

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Проектування комп’ютерних систем на основі мікросхем програмованої логіки: монографія / С.А. Іванець, Ю.О. Зубань, В.В. Казимир, В.В. Литвинов. – Суми: Сумський державний університет, 2013. – 313 с. ISBN 978-966-657-491-9

2. Taylor A. How to implement state machines in your FPGA//XCell, 2012. – №2. – P. 52 – 57.

3. Quartus II Handbook. Version 9.1. Altera, 2009. – 1820 p.

4. IEEE Std 1076-2000 // IEEE Standard VHDL. Language Reference Manual. – New York: IEEE, 2000/ – 290 c.

5. Intel® Quartus® Prime Standard Edition Handbook Volume 1. Design and Synthesis. – 1103 p.

6. Технології проектування комп’ютерних систем. Частина 1. Проектування цифрових систем у САПР Quarus II та лабораторному стенді DE0. Навчальний посібник із дисципліни «Технології проектування комп’ютерних систем» для студентів спеціальності 123 «Комп’ютерна інженерія» / Лахно В.А., Гусєв Б.С., Смолій В.В., Місюра М.Д., Касаткін Д.Ю. – Київ: Компрінт, 2019. – 250 с.