ЗАТВЕРДЖЕНО
Науково-методичною радою
Державного університету
«Житомирська політехніка»

протокол від 12 червня
2025 р. № 4

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

для проведення лабораторних занять
з вибіркової навчальної дисципліни фахової підготовки

«Цифрова техніка та ПЛІС»

для здобувачів вищої освіти освітнього ступеня «бакалавр»

Рекомендовано на засіданні
комп'ютерних технологій у
медицині та телекомунікаціях
26 серпня 2024 р., протокол №8

Розробник: к.т.н., доцент кафедри комп'ютерних технологій у медицині та телекомунікаціях ЦИПОРЕНКО Віталій

Житомир
2024

Зміст

1.	Моделювання мікропроцесорної системи в середовищі PROTEUS	3
2.	Дослідження побудови та функціональних можливостей тригерів	23
3.	Дослідження побудови та функціональних можливостей шифраторів, дешифраторів	34
4.	Дослідження побудови та функціональних можливостей мультиплексорів, демультиплексорів	40
5.	Синтез логічних схем малої інтеграції в ПЛІС в графічному режимі з використанням САПР Altera QUARTUS II	46
6.	Синтез логічних схем перетворювача кодів та дешифратора в ПЛІС мовою Verilog або VHDL з використанням САПР Altera QUARTUS II	59
7.	Дослідження тригерів та регістрів в ПЛІС в графічному режимі з використанням САПР Altera QUARTUS II	73
8.	Дослідження та синтез лічильників в ПЛІС в графічному режимі з використанням САПР Altera QUARTUS II	85
9.	Верифікація HDL-проекту	92
Література		103

Лабораторна робота №1
Моделювання мікропроцесорної системи в середовищі PROTEUS

1.1. Навчальні питання

1. Дослідження основних функцій та компонентів середовища PROTEUS.

2. Дослідження складання схем в середовищі PROTEUS.

1.2. Навчальна мета

Практичне ознайомлення з основними функціями та органами керування програми PROTEUS та отримання практичних навичок моделювання роботи мікропроцесорних пристроїв.

1.3. Теоретичні відомості

Симулятор електронних пристроїв PROTEUS підтримує мікроконтролери AVR, PIC, ARM, Motorola, HC11.

Для ознайомлення з основними функціями та органами керування програми розглянемо приклад: Симуляція пристрою із мікроконтролером (МК) AVR AT90S8515 та інтерфейсом "1-wire" по якому підключені "таблетка" i-button DS1990 (для домофонів) та інші "1-wire" прилади.

Корисна порада на майбутнє – не створюйте проект із нуля, відкривайте будь-який простий приклад з папки SAMPLES і модифікуйте його під свої схему та потреби.

1.4. Порядок виконання роботи

1. Встановіть програмне забезпечення від labcenter: https://www.labcenter.com/free-trial/

Запустіть PROTEUS ISIS і зайдіть у меню File – Open Design. Після в папці C:\Program Files\Labcenter Electronics\Proteus 7 Professional\SAMPLES відкрийте папку VSM for AVR.

Знайдіть в ній і відкрийте папку One-Wire, потім папку NETWORK і відкрийте файл проекту 1WIRE_NET.DSN. Результат попередніх дій зображено на рис. 1.1.

Інтерфейс PROTEUS не має лінійок прокручування. Це зроблено для максимізації робочої області.

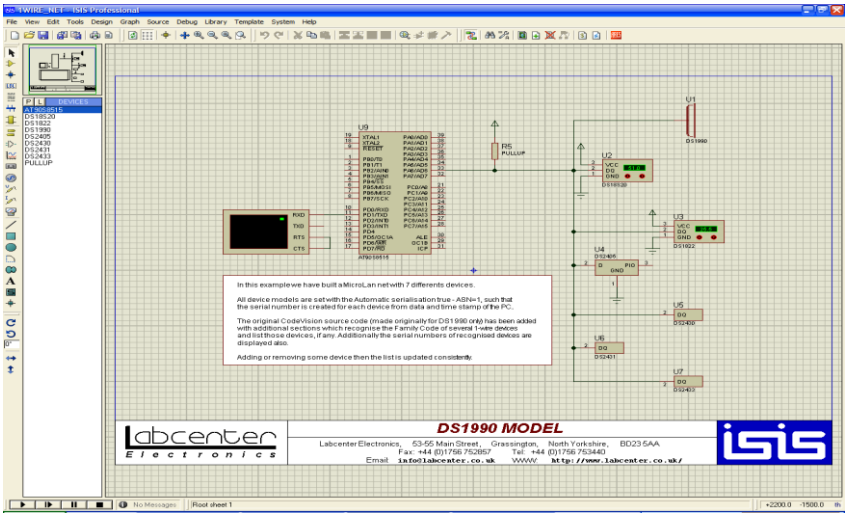


Рис. 1.1 – Головне вікно програми

Масштаб зображення можна змінювати за допомогою колеса на мишці або за допомогою інструментів "лупа +" та "лупа -" в верхній панелі інструментів.

Переміщати зображення можна, вказуючи новий центр зображення інструментом "Хрест" ліворуч від "лупа +" та переміщенням зеленої рамки по „міні макету сторінки” в лівому верхньому кутку робочої області.

Весь робочий аркуш можна побачити, кликнувши кнопку праворуч від "лупа -".

Показати певну область схеми можна, виділивши її за допомогою інструмента виділення області інструментом, що знаходиться ще правіше.

У лівій верхній області екрану ви бачите міні макет сторінки, а трохи нижче панель **DEVICES** (компоненти проекту), що зображена на рис. 1.2.

В цьому вікні відображаються **всі** елементи, які використовуються в схемі.

Кнопка з буквою "**P**" відкриває форму пошуку компонента в бібліотеках PROTEUS для розміщення на схемі.

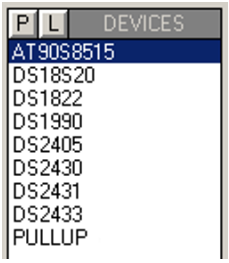


Рис. 1.2 – Панель компонентів проекту

Кнопка з буквою "**L**" відкриває менеджер бібліотек – з його допомогою ви можете підключати нові бібліотеки компонентів.

У відкритому вікні проекту знаходиться мікроконтролер AVR AT90S8515**.** Тут також розміщені декілька компонентів компанії DALLAS – MAXIM – всі вони підключені за схемою однопровідного інтерфейсу 1-Wire.

Оскільки в даній програмі сигнал неможливо передати по одному провіднику – тому у цих компонентів повинні бути з'єднані і виводи GND.

В кінці списку знаходиться компонент PULLUP – це резистор, який підтягує напругу в точці, куди він підключений до значення + живлення МК і приладів DSxxxx – звичайно це +5 вольт.

PROTEUS це інтерактивний довідник по електронним компонентам! Тут можна дізнатися, які компоненти розміщені на схемі та призначення їх виводів !

2. Клікніть кнопку з буквою **P –** відкриється меню пошуку й вибору компонентів "**Pick Devices**", рис. 1.3.

В полі – "**Keywords**" (ключові слова) уведіть **ds2** і виберіть, натиснувши мишкою, верхній з 4-х знайдених компонентів – DS2405. У полі "**Description**" (опис) бачимо "адресований перемикач". До

цього приладу можна звернутися за його адресою і "наказати" йому видати на вихід PIO логічну "1" або "0" або перевести вихід у високоімпедансний Z-стан – з дуже великим опором, що практично не проводить струму, тобто не впливає на пристрій, що до нього підключено.

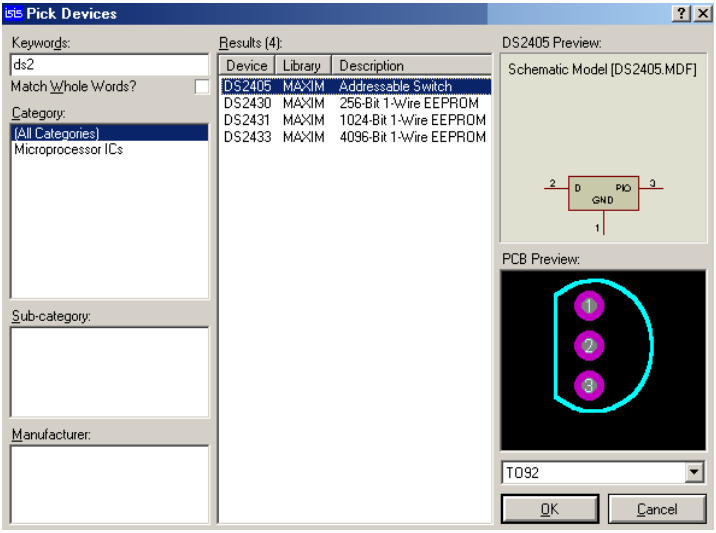


Рис. 1.3 – Вікно пошуку компонентів

У правій частині форми можна побачити назву моделі компонента, його зображення на схемі, а нижче – його "**FootPrint**" – це вигляд компонента на друкованій платі. Ще нижче – назва корпуса компонента TO92 – це маленький трьох вивідний пластиковий корпус у вигляді циліндра 5х5 мм.

Інша частина 1-Wire приладів на схемі починається з ds1 – введіть ці символи в поле ключових слів.

3. Тепер знайдено інші пристрої. Вони розташовані в 2-х категоріях.

Виберіть мишкою DS18S20 – "Високоточний 1-Wire цифровий термометр". На схемному зображенні цього компонента видно деяке

поле, що нагадує дисплей – у ньому будуть виводитися дані в процесі симуляції.

Помістити компонент на схему можна, натисканням "ОК", в результаті чого компонент з'явиться у вікні DEVICES. Виділений компонент можна помістити на аркуші розмістивши мишку в потрібне місце на аркуші схеми і натиснувши лівою кнопкою. Компонент розташується на схемі, рис. 1.4.

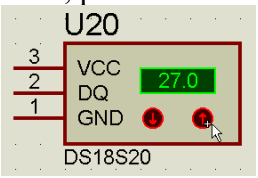


Рис. 1.4 – Розміщення компонента на схемі

Увага! Компонент інтерактивний. Червоними стрілками ви зможете в ході симуляції змінювати температуру корпуса датчика – тобто ту, яку він вимірює – клацаючи по ним мишкою.

Температура відображається на дисплеї компонента. VCC – це + живлення датчика DS18S20, GND – "загальний провідник" або "земля", DQ – лінія даних.

Компонент, що присутній на схемі можна додати і видалити, скопіювати і вставити, тільки в цьому випадку доведеться вручну призначити йому порядковий номер замість знаку питання – ?.

Виділити компонент або будь-який елемент схеми можна, натиснувши по ньому правою кнопкою миші 2 рази. При першому кліку він стане червоним і одночасно будуть виділені всі провідники, що підходять до компонента – вони стануть червоними до першого з'єднання із будь-яким іншим компонентом.

Скасувати виділення усіх виділених компонентів можна, натиснувши **правою** кнопкою миші в порожньому місці схеми.

Видалити компонент або будь-який елемент зі схеми можна двома кліками правої кнопки миші по ньому або натискання **Del** після виділення лівою кнопкою миші.

4. Ознайомтесь, в таблиці 1.1: ліворуч – верхня частина панелі інструментів середовища Proteus. В таблиці 1.2 – панель інструментів ліворуч, нижня частина.

* – дуже корисно ! Дозволяє не перетворювати схему в павутину із провідників, яку неможливо прочитати – використовуйте!

Таблиця 1.1

Призначення елементів верхньої частини панелі інструментів Proteus

	подвійним кліком лівої кнопки миші по компоненту відкриває редагування його властивостей.
	відкрити панель **DEVICES** – компоненти проекту і пошук нових;
	поставити точку з'єднання провідників вручну;
	призначити назву провіднику, натиснувши на ньому лівою кнопкою миші на вписавши придуману назву, наприклад, G1 у вікно string – **одноіменні провідники – електрично З'ЄДНАНІ*!;**
	додати текст у довільне місце схеми;
	прокласти шину – на схемі жирна темно синя лінія;
	створити підсхему – тобто якийсь блок, що містить у собі свою схему і з'єднання.

Таблиця 1.2

Призначення елементів нижньої частини панелі інструментів Proteus

	повернути на 90 градусів за годинниковою стрілкою;
	повернути на 90 градусів проти годинникової стрілки;
	повернути на довільний кут;
	відобразити горизонтально;
	відобразити вертикально.

В таблиці 1.3 – середня частина панелі інструментів. Це інструментарій моделювання роботи електронного пристрою, тобто головне в PROTEUS!

Таблиця 1.3

Призначення елементів середньої частини панелі інструментів Proteus

	***TERMINALS** – виводи живлення (power) – в полі string: +5V; значки вводу (INPUT), виводу (OUTPUT), земля (GND), з'єднання між блоками;
	додати вивід до створюваного компонента;
	графічне відображення, збереження й потужний аналіз результатів симуляції;
	"магнітофон" для запису у файл і відтворення даних;
	генератори будь-яких напруг, струмів, вивід їх з файлів даних;
	вказати точку виміру напруги на провіднику;

	вказати точку виміру струму на провіднику;
	Virtual Instruments – вимірювальні прилади;
	прокладка провідників на схемі.

Панель інструментів знизу, рис. 1.5:

1 2 3 4

Рис. 1.5 – Панель керування симуляцією

На рис. 1.5 позначено:

1 – "Пуск" – запуск симуляції або продовження припиненої симуляції;

2 – "Крок" – виконати мінімальний крок по програмі МК, звичайно це одна команда на асемблері. Цією кнопкою теж можна почати симуляцію;

3 – "Пауза" – пауза симуляції. Можна продовжити кнопками "Пуск" або "Крок";

4 – "Стоп" – зупинка симуляції. Після цього симуляція почнеться спочатку кнопками "Пуск" або "Крок".

5. Просимулюйте приклади з папки SAMPLES – VSM for AVR. Відкрити проект – 1WIRE_NET.DSN. Виділіть мікроконтролер, натиснувши по ньому правою кнопкою миші та вибравши пункт меню **Edit properties**.

У відкритій панелі редагування властивостей компонента (**Edit Component**), клацніть по кнопці "**Hidden Pins**" (сховані виводи) – відкриється додаткове меню в якому показано, як називаються вузли (провідники) схеми до яких підключено живлення МК – **VCC** і його

загальний провідник – **GND**. Зміна цих назв може знадобитися вам при живленні МК або інших компонентів різними напругами або від різних джерел, рис. 1.6.

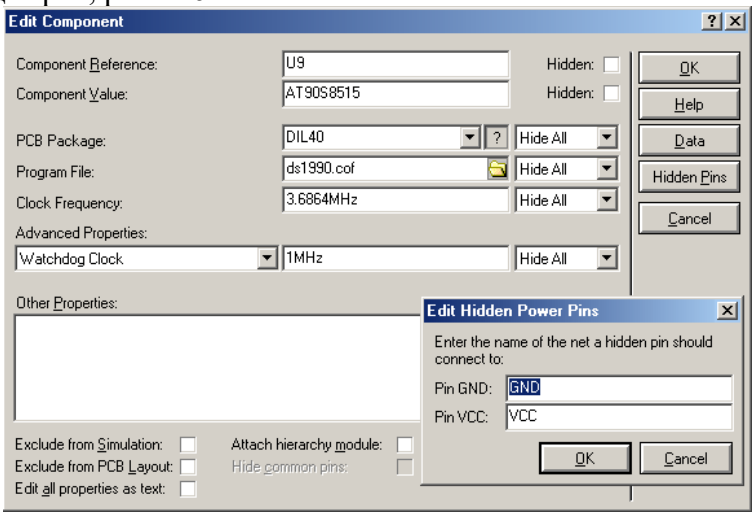


Рис. 1.6 – Редагування властивостей компонента

Підключити пристрій до довільного джерела живлення можна наступним чином:

1. Змінити назву схованого виводу МК на VCC1.
2. Створити на вільному полі вивід живлення зайшовши в TERMINALS, POWER.
3. Клацнути лівою кнопкою миші на стрілочці живлення та у полі string вікна Edit wire label вписати необхідну напругу, наприклад, +15V.
4. Подовжити для зручності провідник. Натиснути на кнопку в панелі інструментів LBL, потім натиснути мишкою на провіднику та дати йому назву VCC1, рис 1.7.
5. Інші провідники можна підключити до створеного джерела живлення давши їм назву VCC1.

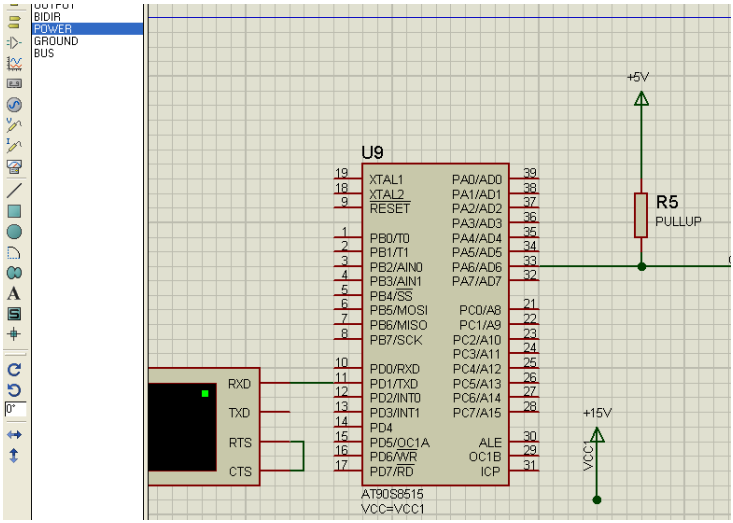


Рис. 1.7 – Приклад підключення скритих виводів до джерела напруги VCC1

Натисніть "ОК" щоб закрити панель схованих виводів і подивіться уважно на вміст панелі редагування компонента.

6. Головне для МК це програма за якою він буде працювати. В полі "**Program File**" потрібно вказати один з наступних варіантів файлу прошивки**:**

***.cof** – файл, якщо ви хочете проводити моделювання за програмою, текст якої написано мовою С.

***.hex** – файл прошивки, якщо у вас немає тексту програми – ні на С, ні на Асемблері.

***.asm** – файл, якщо ви хочете проводити моделювання за програмою, текст якої написано на Асемблері.

Зазвичай підключається файл прошивки ***.hex.**

Якщо у вас немає файлу ***.hex,** який створюється автоматично в результаті компіляції програми в середовищі **AVRStudio4**, а є текст програми ***.asm,** який можна написати в текстовому файлі блокноту – потрібно вказати назву ***.hex** файлу – ds1990.hex (або для МК Tiny 15

назву файлу T15DEMO.HEX) та через меню головної панелі інструментів: Source – Add/Remove Source Files – додати назву файлу з текстом програми на асемблері, натиснувши New і вибравши потрібний компілятор: ds1990.asm, рис. 1.8.

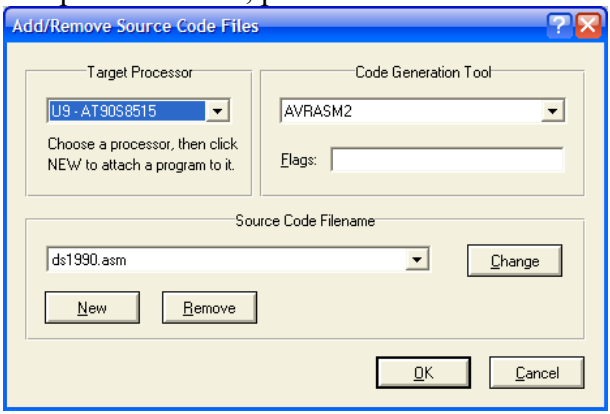


Рис. 1.8 – Додавання файлу з текстом програми в проект

Тепер при запуску симуляції буде відбуватися компіляція файлу ds1990.asm і, якщо немає помилок, то МК почне працювати вже по створеному файлу ds1990.hex.

Отже:

Щоб симулювати в PROTEUS роботу мікроконтролера (AT90S8515) досить:

1) знайти його в бібліотеках і помістити на схему;

2) вказати яку програму (ds1990.hex) він повинен виконувати (як описано вище);

3) вказати частоту тактування МК (Clock Frequency).

Досить просто.

7. Щоб побачити всі установки компонента в панелі редагування поставте галочку Edit all properties as text, рис. 1.9.

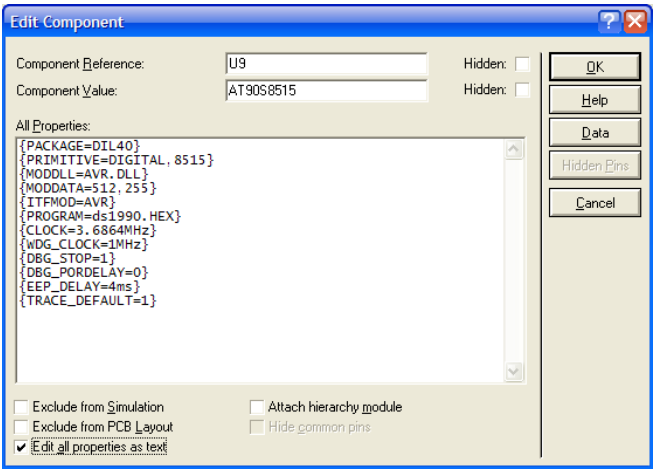


Рис. 1.9 – Редагування властивостей компонента як тексту

CLOCK – однозначно визначає частоту тактування МК при симуляції! Кварц і конденсатори не потрібні для симуляції, їх встановлюють на схему тільки для того щоб врахувати при розведенні друкованої плати пристрою.

WDG_CLOCK – показує частоту роботи генератора сторожового таймера. Хоча вона й позначена 1MHz – "собака" він не включений поки ми не додамо у властивості МК рядок: {WDGON=1} – вам не потрібно його вмикати.

Значення інших параметрів можна довідатися в довідці – кнопка F1 на клавіатурі або Help на панелі інструментів, рис. 1.10.

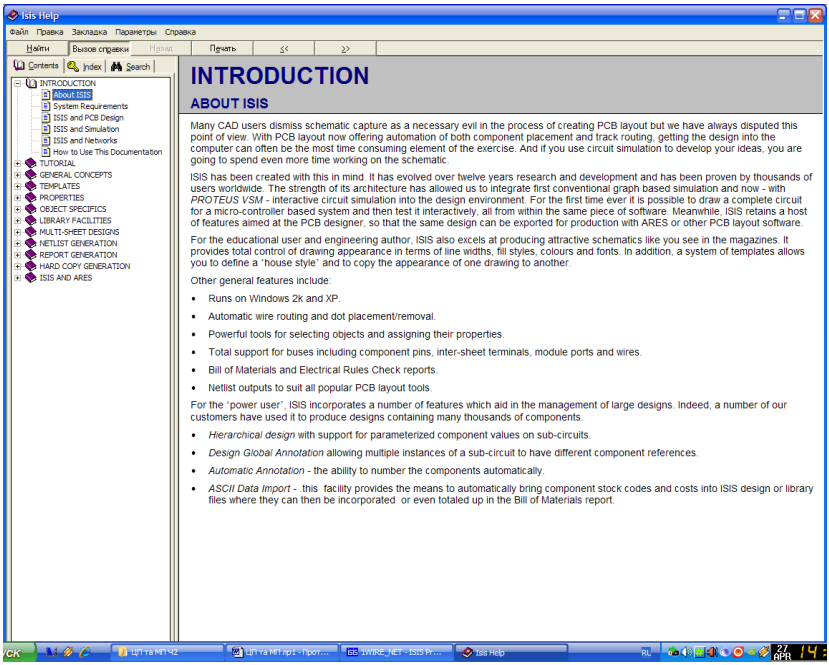


Рис. 1.10 – Довідкова система PROTEUS

8. Важливо! Для **PROTEUS** та для **AVRStudio4** потрібно встановлювати для користувача з ім'ям з латинських символів, а також усі назви проектів **(цифри та англійський алфавіт)!**

9. Запустіть симуляцію! Кнопки керування симуляцією описано вище.

Крім того управляти процесом симуляції програми на асемблері можна за допомогою клавіш F10, F11, F12 і їх комбінаціями із клавішами Alt та Ctrl, та через меню DEBUG.

На рис. 1.11 зображено меню налагодження і обрана команда, що дозволяє виконати зазначений час симуляції**.** Наприклад, 1 мікросекунду або 10.5 секунд.

При кліку по ній виводиться поле введення для вказівки скільки часу повинна відпрацювати схема:

n – нано секунди;
u – мікро секунди;
m – мілі секунди.

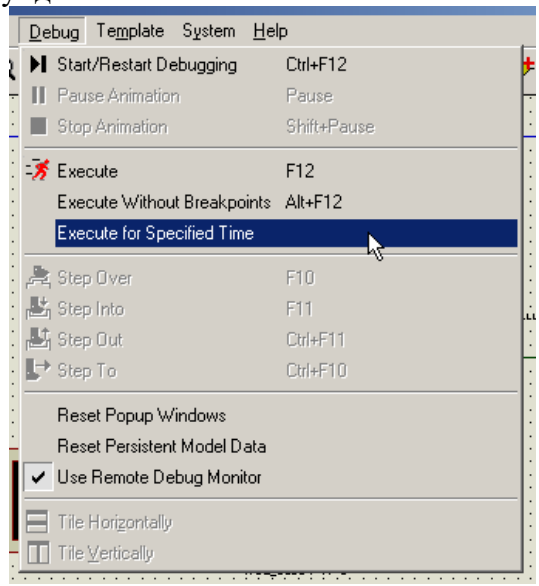


Рис. 1.11 – Меню налагодження і симуляції

Секунди можна не позначати.

Зазначений час буде просимульовано, якщо до його закінчення в програмі МК не зустрінеться точка зупинки (ВР – BreakPoint) – симуляція зупиниться на ній.

F12 – еквівалентна кнопці "Пуск" – запускає або продовжує симуляцію до ВР.

Alt+F12 – запускає або продовжує симуляцію, не зупиняється на ВР.

F10 – симулювати, не входячи в процедуру на Асм або функцію на С – вони будуть виконані як 1 крок.

F11 – симулювати крок по програмі із входом у процедуру або функцію.

Допомога по симуляції МК знаходиться в розділі – Proteus VSM Help – SOURCE LEVEL DEBUGGING WITHIN PROTEUS VSM.

При симуляції пристрою на екрані ми будемо бачити анімацію. Режим анімації можна встановити через меню **System –> Set Annimation Options,** рис. 1.12.

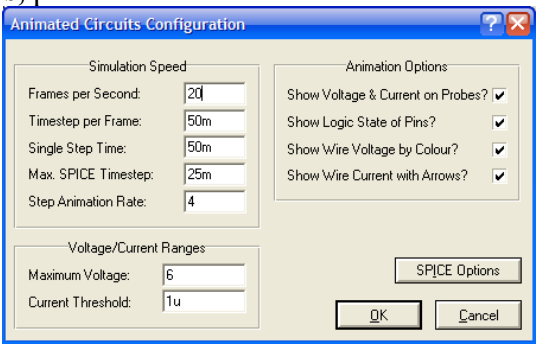


Рис. 1.12 – Вікно настроювання параметрів анімації

На рис. 1.12 зображено вікно настроювання параметрів анімації: звичайний режим – екран обновляється 20 разів за секунду.

– відображати напругу і струм на щупах.

– показувати логічні рівні на виводах.

– показати кольорами напруги на провідниках.

– показати стрілками напрямок струму в провідниках.

Максимальна напруга не обмежує напругу в схемі, що симулюється! Вона вказується для того, щоб PROTEUS зміг вирішити якими кольорами показувати логічні рівні. Крок оцифровки струму 1 мкА – ви можете зменшити цей параметр і поліпшити інші параметри для підвищення точності симуляції – але це потребує більшої обчислювальної потужності ПК.

SPICE Options – відкриває настроювання симуляції моделей.

Довідку по кожному рядку можна одержати за знаком питання.

10. Якщо у вас ще не відкритий проект 1WIRE_NET.DSN – відкрийте його.

Запустіть симуляцію кнопкою "Пуск". З'явиться вікно віртуального терміналу ПК і приблизно за 1 секунду програма МК зробить усе, що від неї було потрібно.

Вивід з UART МК на віртуальний COM порт ПК у вікно-термінал списку розпізнаних пристроїв на шині 1-Wire і їх номери на рис. 1.13.

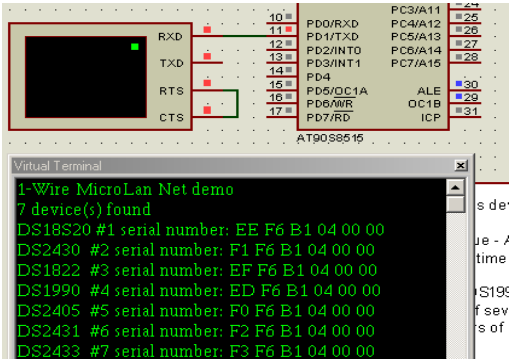


Рис. 1.13 – Симуляція роботи МК

При анімації логічні рівні на виводах вказуються кольоровими квадратиками.

Червоний – "Гарячий" – "1".

Синій – "Холодний" – "0".

Сірий – 3-й високоімпедансний вхід – Z-стан. Для визначеності PROTEUS вважає, що на виводі половина напруги живлення МК.

Номер "таблетки" для домофонів DS1990 написаний на ній.

Серійні номери приладів DSxxxx задані на заводі і не змінювані.

У PROTEUS ці номери задаються у вікні редагування властивостей компонента.

11. Натисніть кнопку "Стоп". Вікно терміналу пропаде.

Давайте відключимо один 1-Wire прилад.

Кликніть два рази правою кнопкою миші на провіднику від мікросхеми U5 – це видалить провідник зі схеми й відключить прилад від шини 1-Wire.

Натисніть кнопку "Пуск". Знову з'явиться термінал але в списку розпізнаних приладів уже не з'явиться відключена U5 і зменшиться число знайдених приладів до 6.

Відновіть вилучений провідник. Це можна зробити двома способами:

1) або натиснути кнопку скасування дії у верхній панелі інструментів.

2) або прокласти провідник заново. Виберіть інструмент – "провідник" на лівій панелі інструментів, проведіть провідник від компонента U5 до шини 1-Wire, утримуючи натиснутою кнопку миші.

Докладніше про термінал читайте по кнопці "Help" у його властивостях.

12. Практичне завдання. Складіть наступну схему цифрового термометра, рис. 1.14.

Примітка! В іноземних програмах типу MicroCup, Pcad, а також Proteus в бібліотеці пристроїв немає вітчизняних пристроїв. Тому замість них потрібно навчитись підбирати близькі за властивостями аналоги.

Схему складати, відкривши Proteus ISIS.

Необхідні компоненти розміщені в бібліотеці Pick Divices:

Відповідність назв вітчизняних та іноземних елементів:

1) Світлодіоди, лампочки – LEDs (Optoelectronics), компоненти LED, LAMP.
2) Резистори – resistors (category), також обираємо subcategory:
   – variable (trimmers) – параметричний термістор – з регульованим опором;
   – NTC – Temperature Sensor – терморезистор (TR).
3) Ключі (Switches):

– button (кнопка);
– RELAY (реле);
– SWITCH (analog primitive) – ключ.

4) Діоди – diodes (subcategory: bridge (двопівперіодний міст), switching (випрямляючі)).
5) Операційні підсилювачі: Operational Amplifiers.
6) Індикація:

– LCD – дисплеї – наприклад таймер-лічильник Milford-2X16-BKP (вхід RXD);
– 7SEG-BCD – 7-сегментна індикація десяткових чисел, що кодується двійковими числами;

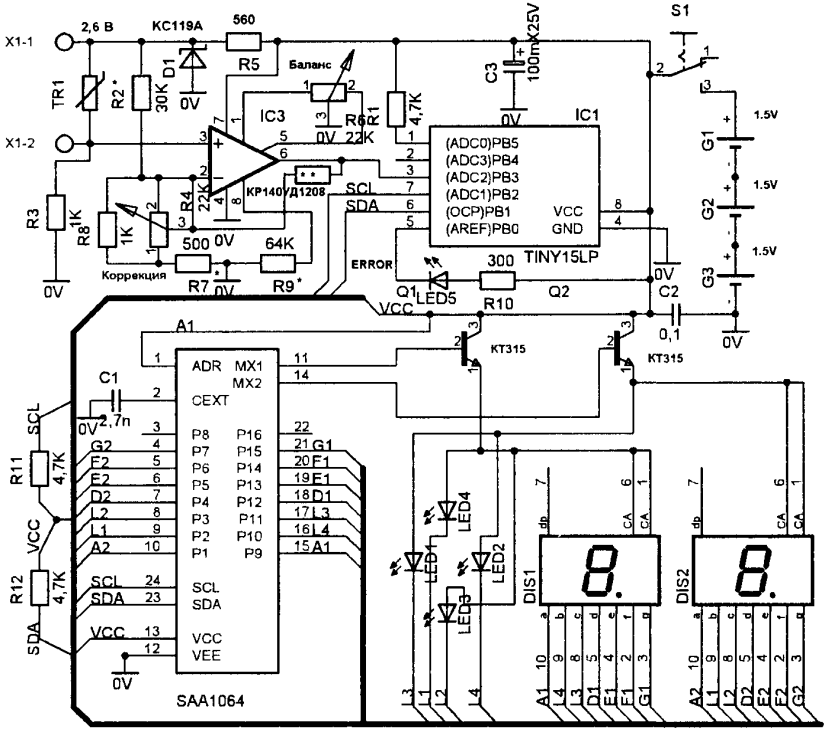


Рис. 1.14 – Досліджувана схема цифрового термометра

7) Драйвер індикації:
– BCD to 7-segment driver/decoder (дивись приклад використання в папці проектів Samples – VSM for AVR – AVR Tiny15 Demo – t15demo.DSN, мікросхема U2 разом з тригером U3).
8) Конденсатори – Capacitors.
9) Котушки індуктивності – inductors.
10) Транзистори біполярні – transistors bipolar.
11) Цифрові пристрої (TTL 74 series):
– Comparators – компаратори;
– counters – лічильники;
– Flip-Flops – тригери (RS, JK, D, T);
– Registers – регістри (складаються з тригерів);
– Multiplexers – мультиплексори;
– Gates – схема „І”.

Готові приклади знаходяться в Program files – Labcenter Electronics – Proteus 7 Professional – Samples – VSM for AVR.

1.5. Зміст звіту

Звіт повинен містити:

1. Тему роботи.

2. Результати досліджень, складену схему.

3. Зробити висновки: про об’єм виконаної роботи (повністю, або крім пунктів хх), чи досягнена мета роботи, чи практичні результати співпали з теоретичними.

1.6. Контрольні тестові питання

1. Які мікроконтролери підтримує симулятор електронних пристроїв PROTEUS?

2. Що відображає панель DEVICES?

3. Як виконати подачу живлення або певного сигналу в довільну точку схеми?

4. Який вигляд мають файли прошивки виконані на язиках с, assembler, та в машинних кодах?

5. Які дії необхідно провести, щоб просимулювати в PROTEUS роботу мікроконтролера МК?

6. Як задати частоту тактування МК при симуляції?

7. Яке призначення має віртуальний термінал?

8. Як задати необхідний час симуляції?

Лабораторна робота №2
Дослідження побудови та функціональних можливостей тригерів

2.1. Навчальні питання

1. Виконати дослідження принципів побудови та функціональних можливостей D, T, RS та JK-тригерів.
2. Дослідити можливість одночасного функціонування двох тригерів, що реалізовані в одній мікросхемі та визначити пріоритетність тригерів та їх окремих входів.

2.2. Навчальна мета

1. Експериментальне дослідження принципів побудови та функціональних можливостей інтегральних тригерів.

2.3. Теоретичні відомості

Тригери – елементарні автомати, що мають в собі елемент пам'яті (фіксатор) та схему керування. Фіксатор будується на двох інверторах» що пов'язані один з другім "навхрест" так, що вхід одного з'єднано з ВИХОДОМ ІНШОГО**.** Таке з'єднання дає коло з двома стійкими станами (рис.2.1). Дійсно, якщо на виході інвертора 1 є логічний нуль, то він забезпечує на виході інвертора 2 логічну одиницю, завдяки ЯКІЙ ВІН існує. Таке ж погодження сигналів має місце і для другого випадку, коли інвертор 1 знаходиться в одиниці, а інвертор 2 в нулі.

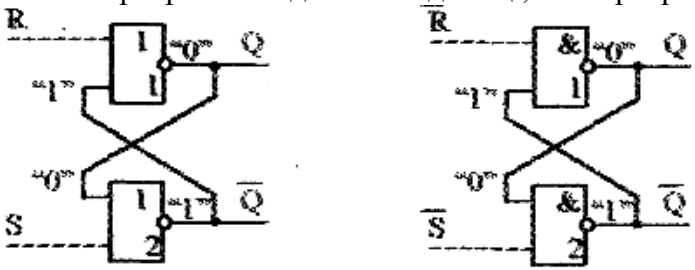


Рис. 2.1 – Схеми фіксаторів з входами керування на елементах АБО-НІ та І-НІ

Будь який з цих двох станів може існувати необмежено довго. Щоб керувати фіксатором необхідно мати в логічних елементах додаткові входи, які перетворюють інвертори в елемента І-НІ чи АБО-НІ. На входи керування подаються зовнішні керуючі сигнали. Керуючі сигналі показані на рис 2.1 штриховими лініями. Буквою R (від Reset) позначено сигнал установки в нуль (скидання), а буквою S (від Set) - установки в одиницю (установка). Стан тригера зчитують по значенню прямото виходу, що позначається як Q. Майже завжди тригер має і другий вихід з інверсним сигналом Q. Для фіксатора на елементах АБО-НІ активним рівнем є рівень логічної одиниці, тому що саме ВІН призводить до установки логічного елемента в нуль, незалежно від сигналів на інших входах. Для фіксатора на елементах І-НІ активним рівнем є рівень логічного нуля, тому що має таку саму властивість однозначно задавати стан елемента незалежно від інших входів.

Класифікація тригерів. Класифікація тригерів проводиться за ознаками логічного функціонування та за способом запису інформації (рис.2.2).

За логікою функціонування є тригери типів RS, D, T, JK.

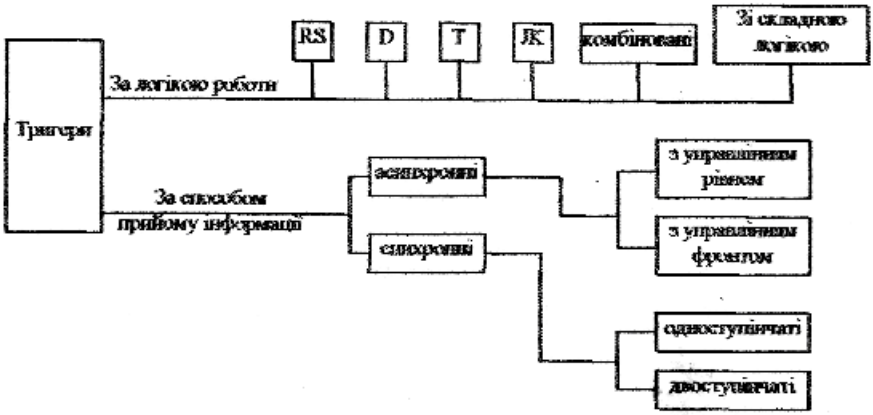


Рис 2.2 – Класифікація тригерів

Крім того використовуються комбіновані тригери, в котрих суміщаються одночасно тригера декількох типів, і тригери зі складною вхідною логікою (групами входів, пов'язаних між собою логічними залежностями).

Тригер типу RS має два входи - установки в одиницю (S), та установки в нуль (R). Одночасна подача сигналів установки та скидання не допускається (ця комбінація називається забороненою).

Тригер типу D має один інформаційний вхід. Його вихід повторює вхідний сигнал, але лише після надходження тактового сигналу.

Тригер типу T змінює свій стан кожен раз, при надходженні вхідного сигналу. Він має один вхід, і називається лічильним тригером.

Тригер типу JK універсальний (на його основі можна збудувати будь який тригер), має входи установки (J) та скидання (K), що подібні входам тригера (RS). На відміну від останнього, допускаться ситуація з одночасною подачею на сигналів на обидва цих входи (J=K=1). В цьому режимі працює як лічильний тригер відносно третього (тактового) входу.

В комбінованих тригерах суміщається декілька режимів. Наприклад тригер RST – лічильний тригер, що має входи установки та скидання.

За способом запису інформації розрізняють асинхронні (нетактовні) і синхронні (трактовані) тригер. В нетактовних перехід в інший стан викликається безпосередньою зміною вхідних інформаційних сигналів. В
тактовних, що мають спеціальний вхід, перехід відбувається лише при подачі на цей вхід тактовних сигналів. Позначаються такі входи буквою C (від Clock).

За способом сприйняття тактових сигналів тригери діляться на *керовані рівнем та керовані фронтом.* Керування рівнем означає, що при одному рівні тактового сигналу тригер сприймає вхідні сигнали та реагує на них, а при іншому - не сприймає, і залишається в

незмінному стані. При управлінні фронтом дозвіл на зміну стану дається лише в момент перепаду тактового сигналу (на його фронті чи спаді). Весь інший час, незалежно від рівня тактового сигналу тригер не сприймає вхідні сигнали і залишається в незмінному стані. Такі мікросхеми називають мікросхемами з динамічним керуванням, а тактовий вхід - динамічним.

Якщо тригер має синхронні та асинхронні входи (входи установки та скидання), то асинхронні входи є домінуючими, вплив по ним відбувається незалежно від сигналів на інших входах, які при цьому ігноруються. Якщо сигналами асинхронних входів тригер переводиться в режим зберігання, то стан тригера зберігається до першого активного синхросигналу, після чого визначається сигналами на синхронних входах.

За *характером процесу переключення* тригери ділять на одноступінчасті та двоступінчасті. В одноступінчастому тригері перехід в інший стан відбувається відразу, а в двоступінчастому - по етапам. Двоступінчасті тригери складаються з вхідної та вихідної частин, перехід в інший стан відбувається в обох частинах по черзі. Один з рівнів тактового сигналу дозволяє запис інформації у вхідну частину при незмінному виході, а інший - передачу нового стану із вхідної частики в вихідну.

Способи опису тригерів. Логічне функціонування тригерів описується способами, прийнятими для автоматів взагалі: таблицями істинності, картами Карно, характеристичними рівняннями, діаграмами станів, “словниками” (інша форма діаграм стану).

Приведемо коротку характеристику кожної з цих форм опису на прикладі JK - тригера.

В таблиці істинності (таблиці переходів - виходів) в якості змінних використовуються сигнали на входах тригера та попереднє значення виходу, а як результат - значення виходу тригера в яке він переключиться при такому впливі. Таблицю можна записати в повному табл.2.1, та скороченому, табл. 2.2, вигляді.

Таблиця 2.1

J	K	Qn-1	Qn
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	1	1	0
1	0	1	1
1	1	0	1
1	1	1	0

Таблиця 2.2

J	K	Qn
0	0	Qn-1
0	1	0
1	0	1
1	1	$\vec{Q}_{n-1}$

Через Qu позначено стан тригера після переключення, через Qu-1 – до переключення.

Карту Карно для JK - тригера показано на рис.2.3 а. Неї можна отримати з таблиці істинності (таблиці переходів - виходів). З карти Карно можна отримати характеристичне рівняння тригера:

$$Q_n = JQ_{n-1} + KQ_{n-1} \qquad (2.1)$$

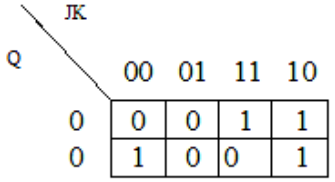
Діаграма стану (рис 2.3, б) відображає наявність у тригера двох стійких станів і умови переходу з одного стану в інший. Словник тригера (табл. 2.3) дає ту ж інформацію в аналітичному вигляді, і є інструментом проектування схем, які мають в своєму складі тригери.

Таблиця 2.3

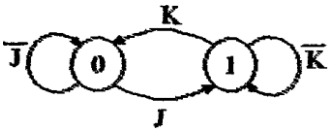
Перехід	J	
1→1	0	X
1→1	1	X

1→1	x	1
1→1	x	0

Важливим способом опису функціонування тригерів (як і інших автоматів) є часові діаграми, які відображають не лише логічне функціонування схеми, але й її поведінку в часі.



а



б

Рис. 2.3 – Карта Карно – а, діаграма станів – б для JK – тригера

Ця поведінка іншими способами не описується, тому, в ряді випадків, часові діаграми є незамінними.

Схемотехніка тригерних пристроїв. Найперше, що необхідно відзначити, це те, що між тригерами RS і D з одного боку, та T і JK з іншого, є суттєва різниця. Перші мають розімкнену структуру, а другі використовують вихідні сигнали для впливу на свої входи. Візьмемо за основу синхронний RS тригер, і розглянемо схеми інформаційних зв'язків, що створюють інші типи тригерів.

Тригер типу D виникає, якщо подавати на вхід S значення D, а на вхід R - його інверсію (рис. 2.4, а). Тригер типу T створюється на основі RS за схемою, (рис 2.4, б). В цьому випадку роль лічильного входу відіграє тактовий вхід. Дійсно, при кожному дозволі прийому інформації тригер по зворотнім зв'язкам приймає стан, протилежний поточному, тобто переключається. Тригер T аналогічним способом

можна отримати і на основі D - тригера. Схему інформаційних зв'язків, що створюють JK - тригер визначимо формальним шляхом, використовуючи дані таблиці 2.4. Ліва частина таблиці показує функціонування JK - тригера, а праві стовпчики – сигнал RS - тригера, що забезпечують необхідні переходи. З цих стовпчиків отримуємо:

$$S = J\vec{K}\vec{Q}_{n-1} + JKQ_{n-1} = J\vec{Q}_{n-1} : R = \vec{J}KQ_{n-1} + JKQ_{n-1} \qquad (2.2)$$

Таблиця 2.4

J	K	Qu-1	Qu	Перехід	S	R
0	0	0	0	0→0	0	0
0	0	1	1	1→1	0	0
0	1	0	0	0→0	0	0
0	1	1	0	1→0	0	1
1	0	0	1	0→1	1	0
1	0	1	1	1→1	0	0
1	1	0	1	0→1	1	0
1	1	1	0	1→0	0	1

Схема інформаційних зв’язків, що утворюють JK - тригер показана на рис. 2.5.

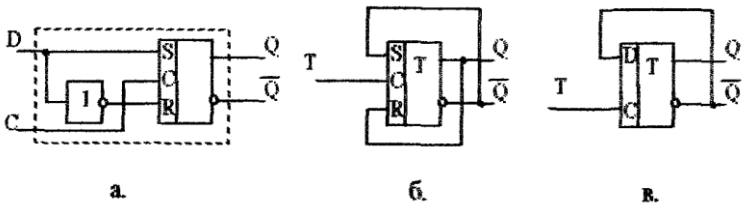


Рис.2.4 –Схеми інформаційних зв'язків, що створюють тригери D (а) і T (б, в).

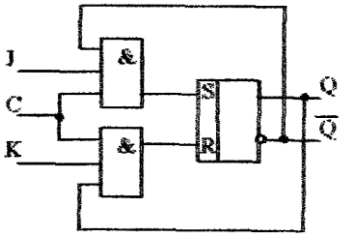


Рис.2.5 – Схеми інформаційних зв'язків, що створюють JK - тригер

Розглянемо причину існування для RS - тригера забороненої комбінації R=S=1. Що ж відбувається при подачі комбінації R=S=1? Видно, що в ньому випадку обидва виходи тригера (розглядало схему на елементах АБО-НІ) стануть нульовими. Якщо після цього на входах з'явиться комбінація 10 або 01 то тригер перейде в стан, що відповідає цій комбінації. Якщо ж після забороненої комбінації з’явиться комбінація 00 (зберігання), то виникне непередбачувана ситуація. Спочатку обидва елемента знаходяться в стані нуля. При подачі на два входи рівнів нуля, обидва елементи починають переходити в стан 1. При зміні напроти на виходах (а, значить, і входах) елементів від рівня нуля до рівня одиниці елементи проходять через точку рівноваги (вхідна напруга елемента дорівнює вихідній). Зростання напруги на виходах двох елементів в цій точці припиняється і елементи “зависають” в стані нестійкої рівноваги між нулем і одиницею. При найменшій флуктуації напроти на одному з входів напруга на виході одного з елементів починає різко зростати, а іншого - зменшуватись, і тригер перекидається в один зі стійких станів. Яка саме флуктуація, і на якому з входів, виникне сказати наперед неможливо. Тобто неможливо сказати в який саме стан переключиться тригер. Саме через це комбінація 11 вважається забороненою, тому що використовувати схему поведінка якої наперед невідома, якщо не говорити про спеціальні призначення, не дозволяється.

На рис. 2.5 показано часові діаграми, що ілюструють реакцію D - тригерів різних типів на показані вхідні сигнали.

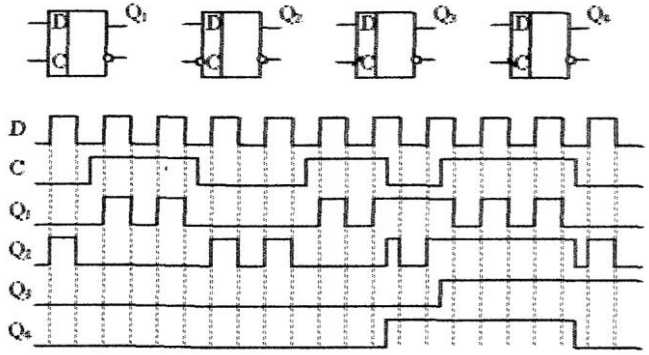


Рис. 2.6 –Часові діаграми роботи D – тригерів

2.4. Порядок виконання роботи

1. Ознайомитись з методичними вказівками. Переглянути лекційні матеріали за темою тригери.

2. Проаналізувати варіанти програмних середовищ для дослідження та моделювання схем з використанням цифрової елементної бази.

3. В середовищі Proteus створити новий проект в якому розмістити з бібліотеки (**P** – ***Pick Devices*** в режимі ***Component Mode***) досліджувані мікросхеми усіх варіантів тригерів на робочий стіл. Підключити до них з меню ***Instruments*** генератор прямокутного сигналу (***Generators*** – Pulse, DC на 2входи: +5V, 0V, частота 50Hz). Початково задавати постійну напругу +5V, що відповідає цифровій одиниці «1», напругу 0V – для «0». Дослідити діапазони значень напруг «0» та «1» для коректної роботи мікросхем.

Далі для кожного цифрового елемента та мікросхеми в зошиті складаємо таблицю істинності (переходів) для усіх можливих вхідних комбінацій цифрового сигналу: наприклад 00, 01, 10, 11.

Для дослідження тригерів окремо обираємо категорію ***Simulator Primitives*** – JKFF, DTFF, D-Type. Тригер – **Flip-Flop**.

Для дослідження мікросхем в бібліотеці робимо пошук за категорією ***Category***, наприклад беремо TTL 74F series або іншу.

4. Дослідити принципи одночасного функціонування JK-тригера та RS-тригера, що реалізовані в одній мікросхемі.

5. За допомогою необхідної комбінації значень вхідних сигналів забезпечити переведення JK-тригера в режим функціонування T-, RS-, D-тригерів забезпечивши їх експериментальну перевірку.

2.5. Обробка результатів

1. За результатами виконання завдання визначити: аналітичну форму запису функціонування RS-тригера; функціональні типи входів та виходів тригера; заборонену комбінацію RS-тригера.

2. Визначити аналітичну форму запису функціонування D-тригера; функціональні типи входів та виходів тригера; наявність або відсутність забороненої комбінації D-тригера.

3. Визначити пріоритетність входів керування RS- та D-тригерів; наявність заборонених комбінацій вхідних сигналів.

4. Визначити аналітичну форму запису алгоритму функціонування JK-тригера; заборонену комбінацію вхідних сигналів; функціональні типи входів та виходів JK-тригера.

5. Визначити пріоритетність входів керування RS- та JK-тригерів; наявність заборонених комбінації вхідних сигналів.

6. За результатами п. 4 визначити: способи переведення JK-тригера в режими функціонування T-, RS-, D-тригерів.

7. За результатами п. 11 визначити: принципи функціонування T-, D-, JK-тригерів у динамічному режимі.

8. Результати досліджень оформити у вигляді звіту, який підлягає індивідуальному захисту.

2.6. Зміст звіту. Звіт повинен містити:

1. Структурні схеми досліджуваних пристроїв.

2. Результати розрахунків.
3. Зробити висновки: про об’єм виконаної роботи, чи досягнена мета роботи, чи практичні результати співпали з теоретичними.

2.7. Контрольні тестові питання.

1. Яка пріорітетність входів керування JK-тригера?
2. Який вигляд має аналітична форма запису функціонування RS-тригера?
3. Який вигляд має аналітична форма запису функціонування JK-тригера?
4. Який вигляд має аналітична форма запису функціонування D-тригера?
5. Які є способи переводу JK-тригера в режими функціонування T-тригера?
6. Які є способи переводу JK-тригера в режими функціонування RS-тригера?
7. Які є способи переводу JK-тригера в режими функціонування D-тригера?
8. Яка пріорітетність входів керування RS-тригера?
9. Який принцип одночасного функціонування JK-тригера та RS-тригера, що реалізовані в одній мікросхемі?
10. Яке застосування в цифровій техніці мають тригери?

Лабораторна робота №3
Дослідження побудови та функціональних можливостей дешифраторів

3.1. Навчальні питання

1. Дослідити принципи функціонування та функціональні можливості двійкового дешифратора та шифратора.
2. Дослідити принципи функціонування семисегментного дешифратора в режимі ручного тактування та динамічному режимі.

3.2. Навчальна мета

1. Експериментальне дослідження принципів побудови та функціональних можливостей інтегральних цифрових дешифраторів.

3.3. Теоретичні відомості

Дешифратори відносять до перетворювачів кодів. Двійкові дешифратори перетворюють двійковий код в код "1 з N". В цьому коді лише одну позицію займає одиниця, а всі інші – нуль.

Двійковий дешифратор, що має n входів, повинен мати 2^n виходів, що відповідають числу різних комбінацій в n-розрядному двійковому коді. В залежності від вхідного двійкового коду, в дешифраторі збуджується один вихід.

Якщо частина вхідних комбінацій не використовуються, то дешифратор називають неповним, і він має менше ніж 2 виходів.

В умовному позначенні дешифратора проставляють букви DC (від англійського Decoder). Входи дешифратора прийнято позначати їх двійковими вагами. Крім інформаційних входів дешифратор зазвичай має один або декілька входів дозволу роботи, що позначається як EN (Enable). При наявності дозволу по цьому входу дешифратор працює, як описано вище, а при його відсутності всі виходи дешифратора пасивні. Якщо в ході дозволу роботи декілька, то сигнал дозволу роботи визначається як кон’юнкція окремих сигналів дозволу. Умовне графічне позначення дешифратора приведено на рис. 3.1. Часто дешифратори мають інверсні виходи. В цьому випадку лише один вихід має нульове значення, а всі інші

одиничні. При забороні роботи такого дешифратора на всіх його виходах буде присутня логічна одиниця.

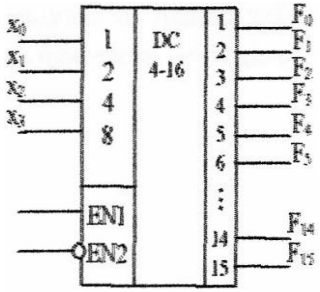


Рис. 3.1 – Умовне графічне позначення дешифратора

Нарощування розрядності дешифратора. Малорозрядність стандартних дешифраторів ставить задачу про нарощування їх розрядності. З мало розрядних дешифраторів можлива побудова схеми, еквівалентної дешифратору більшої розрядності. Для цього вхідне слово ділиться на дві частини (поля). Розрядність поля молодших розрядів відповідає числу входів тих дешифраторів, на яких необхідно побудувати схему. Поле старших розрядів, що залишилось, використовується для отримання сигналів дозволу роботи одного з дешифраторів, які декодують молодші розряди.

Як приклад, на рис. 3.2. приведено схему дешифрування п’ятирозрядного двійкового коду за допомогою дешифраторів «3 - 8» та «2 - 4».

Щоб отримати необхідні 32 виходи, складено стовпчик з чотирьох дешифраторів «3 - 8». Дешифратор «2 - 4» приймає два старших розряда вхідного коду. Збуджений вихід цього дешифратора відкриває один з дешифраторів стовпчика по його входу дозволу роботи. Вибраний дешифратор стовпчика розшифровує три молодших розряди вхідного слова.

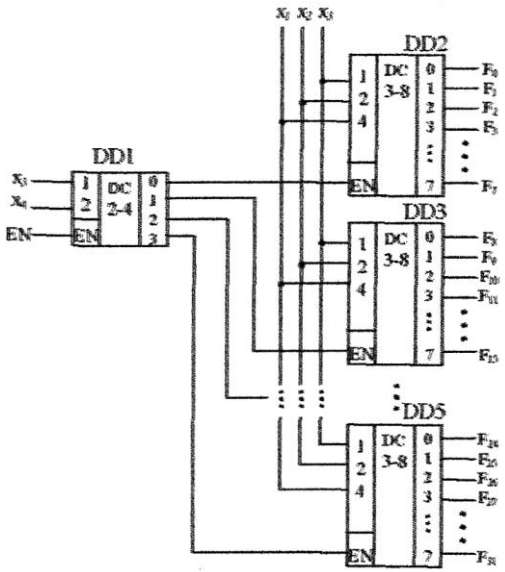


Рис. 3.2 – Схема дешифрування п’ятирозрядного двійкового коду за допомогою дешифраторів «3 - 8» та «2 - 4»

Таким чином, кожному вхідному слову відповідає збудження лише одного виходу. Наприклад, при дешифрування слова $11001_2 = 25_{10}$ на вході дешифратора першого ярусу «4 - 1» маємо код 11 (два старших розряди). Цей код збуджує вихід 3 дешифратора першого ярусу, що дозволяє роботу мікросхеми DD5 другого ярусу. На вході DD5 діє код 001 (молодша частина вхідного слова), тому одиниця з’явиться на виході 1 дешифратора DD5, тобто на виході 25 схеми в цілому, що і вимагалась. При цьому дозвіл або заборона роботи схеми в цілому здійснюється по входу Е дешифратора першого ярусу.

Дешифратор спільно зі схемами АБО можна використовувати для реалізації будь-якої логічної функції. Дійсно, на виходах дешифратора виробляються всі кон’юнктивні терми, які можна скласти з даного числа аргументів. Логічна функція в СДНФ є

диз'юнкція деякого числа таких термінів. Складаючи потрібні терми за схемою АБО можна отримати будь-яку логічну функцію.

На рис. 3.3 як приклад показано реалізацію двох функцій за допомогою дешифратора $F_1 = \vec{x}_3\vec{x}_2 \vee x_3x_1$ та $F_2 = \vec{x}_3\vec{x}_2x_1 \vee x_2\vec{x}_1$. Таке рішення може бути доцільним при необхідності отримання декількох функцій одних і тих же змінних. При цьому для отримання додаткової функції необхідно додати лише один елемент АБО. Зауважимо, що для перевірки правильності реалізації функцій їх зручно перевести в СДНФ.

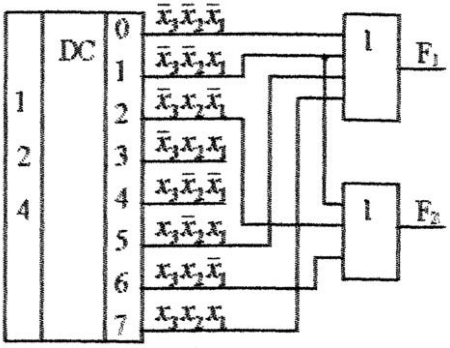


Рис. 3.3 – Схема відтворення логічних функцій за допомогою дешифратора

Шифратори. Двійкові шифратори виконують функцію обернену дешифраторам, тобто перетворення коду «1 з N» в двійковий. При збудженні одного з дешифратора на його виході формується двійковий код номера збудженої вхідної лінії. Повний двійковий шифратор має 2^n входів, та n виходів.

Пріоритетні шифратори виконують більш складну функцію. При роботі цифрового пристрою досить часто виникає потреба у визначенні пріоритетного права на використання деякого ресурсу. При цьому декілька споживачів виставляють свої запити на

обслуговування що не можуть бути задоволені одночасно. Необхідно вибрати того, кому надається право першочергового обслуговування. Найпростіший варіант розв'зання цієї задачі – присвоєння кожному з споживачів фіксованого рівня пріоритету. Наприклад, група з 8 споживачів може бути сформована так, що найвищий пріоритет має споживач № 7, а далі пріоритет зменшується від споживача до споживача. Найнижчий пріоритет має споживач №0, він буде обслуговуватись лише за відсутності інших запитів. Якщо одночасно приходить декілька запитів, то обслуговується запит з найвищим номером. Пріоритетний шифратор виробляє на виході двійковий номер найстаршого запиту.

3.4. Порядок виконання роботи

1. Ознайомитись з робочим лабораторним місцем та методичними вказівками.

2. При проведенні досліджень в середовищі Proteus створити новий проект в якому вибрати та розмістити з бібліотеки досліджувані мікросхеми різних дешифраторів та шифраторів на робочий стіл. Підключити до них з меню ***Instruments*** генератор сигналів.

Для дослідження мікросхем в бібліотеці обираємо дешифратор – DC (Decoder), шифратор – Coder.

3. Далі для кожного цифрового елемента в зошиті складаємо таблицю істинності (переходів) для усіх можливих вхідних комбінацій цифрового сигналу адреси А: наприклад 0000, 0001, 0010, 0011…, 1111.

4. Провести дослідження семисегментного дешифратора.

5. Дослідити діапазони значень напруг «0» та «1» для коректної роботи мікросхем.

3.5. Обробка результатів

1. За результатами п. 2,3 завдання визначити функціональне призначення входів двійкового дешифратора, визначити його повні функціональні можливості та функціональні типи входів і виходів.

Встановити відповідність вхідних і вихідних сигналів дешифратора, що досліджувалися, конкретним виводам мікросхеми.

2. Визначити принцип дії двійкового дешифратора і подати його в словесній та аналітичній формах.

3. Визначити принцип дії семисегментного дешифратора і подати його в словесній та аналітичній формах.

4. Синтезувати функціональну схему семисегментного дешифратора з кількістю виходів по завданню викладача.

5. Результати обробки порівняти з довідковими даними та зробити обґрунтовані висновки щодо отриманих результатів.

6. Результати досліджень оформити у вигляді звіту, що підлягає індивідуальному захисту.

3.6. Контрольні тестові питання

1. Як сформулювати в словесній формі принцип роботи дешифратора?

2. Яке визначення повного і неповного дешифраторів?

3. Як дати в словесній та аналітичній формах визначення принципів функціонування двійкових та семисегментних дешифраторів?

4. Які функціональні можливості мають сучасні інтегральні дешифратори?

5. Як реалізувати на дешифраторі 2х4 (2входи, 4виходи) двійковий дешифратор на 27 виходів?

6. Як реалізувати на основі дешифратора задану логічну функцію?

7. Яке застосування дешифраторів в цифрових пристроях?

8. Які принципи побудови двійково-десяткових дешифраторів?

9. Які принципи побудови шифраторів?

10. Яке застосування шифраторів в цифрових пристроях?

11. В якій послідовності здійснюється мінімізація логічної функції методом карт Карно?

Лабораторна робота №4
Дослідження побудови та функціональних можливостей мультиплексорів

4.1. Навчальні питання

1. Дослідити принципи функціонування та функціональні можливості шістнадцятивходового мультиплексора з п’ятьма керуючими входами і одним виходом.

2. Дослідити робочі діапазони напруг цифрового нуля та одиниці для мікросхем різних технологій виготовлення.

4.2. Навчальна мета

1. Експериментальне дослідження принципів побудови та функціональних можливостей інтегральних мультиплексорів.

4.3. Теоретичні відомості

Мультиплексори здійснюють з'єднання одного із вхідних каналів з виходом, під керуванням керуючого слова (адресного слова). Розрядності каналів можуть бути різними, мультиплексори для комутації багаторозрядннх даних складаються з однорозрядннх мультиплексорів. Входи мультиплексорів ділять на дві групи: інформаційні та адресні. Роботу мультиплексора спрощено можна уявити, як роботу багатопозиційного перемикача (рис. 4.1, а). Адресний код А задає перемикачу деяке положення, з’дниючи з виходом F один з інформаційних входів x_i. При нульовому адресному коді перемикач займає верхнє положення x_0, при збільшенні коду на одиницю, переходить в сусіднє положення x_1, потім в третє, і так далі. Схемотехнічно цифровий мультиплексор реалізує лише електронну версію вказаного перемикача, маюча на відміну від нього лише односторонню передачу даних. Робота мультиплексора описується співвідношенням :

$$F = x_0\vec{a}_{n-1}\vec{a}_{n-2}\cdots\vec{a}_1\vec{a}_0 \vee x_1\vec{a}_{n-1}a_{n-2}\cdots\vec{a}_1a_0 \vee \cdots \vee \vee x_{2^n-1}a_{n-1}a_{n-2}\cdots a_1a \quad (4.1)$$

яке іноді називають мультиплексною формулою. При будь якому значенні адресного коду всі доданки, окрім одного дорівнюють нулю, ненульовий доданок дорівнює x_i, де і - значення поточного адресного коду.

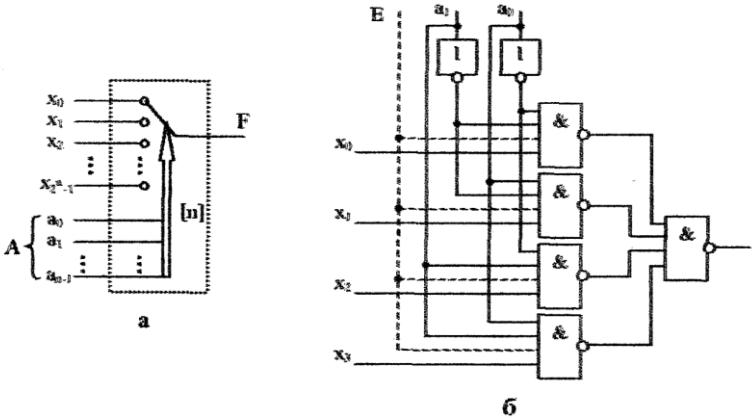


Рис. 4.1 – Спрощене уявлення мультиплексора багатопозиційним перемикачем (а) та реалізація мультиплексора на елементах І-НІ (б)

На рис. 4.1, б показано мультиплексор з чотирма інформаційними входами, двома адресними входами, та входом дозволу роботи. При відсутності дозволу роботи (E=0) вихід F залишається нульовим незалежно від інформаційних та адресних входів. В стандартних серіях мультиплексори мають розмірність не більше 16x1.

Нарощування розрядності мультиплексорів можливе за допомогою пірамідальної структури з декількох мультиплексорів. При цьому перший ярус схеми являє собою стовпчик, що містить в собі стільки мультиплексорів, скільки необхідно для отримання

потрібної кількості інформаційних входів. Всі мультиплексори стовпчика адресуються одним і тим самим кодом, складеним з відповідного числа молодших розрядів загального адресного коду (якщо число інформаційних входів схеми дорівнює 2^n, то загальне число адресних розрядів дорівнює n, молодше поле n_1 адресного коду використовується для адресації мультиплексорів першого ярусу). Старші розряди адресного коду використовуються в другому ярусі, мультиплексор котрого забезпечує почергову роботу мультиплексорів першого ярусу на загальний вихідний канал (рис. 4.2).

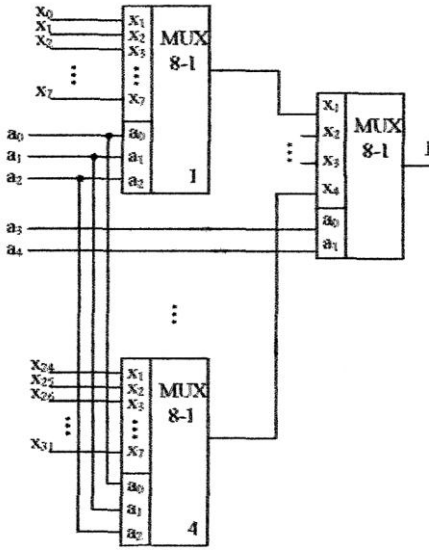


Рис. 4.2 – Схема нарощування розрядності мультиплексора

Демультиплексори виконують операцію обернену операції мультиплексорів - передають дані з одного вхідного каналу в один з декількох вихідних. Багаторозрядні мультиплексори складають з декількох одно розрядних. Неважко помітиш, що дешифратор з входом дозволу роботи буде працювати в режимі демультиплексора,

якщо на вхід дозволу роботи подавати інформаційний сигнал. Дійсно, при одиничному значенні цього сигналу подача адресного коду на входи дешифратора призведе до збудження відповідного виходу, а при нульовому – ні. А така робота відповідає передачі інформаційного сигналу в адресований вихідний канал. В зв'язку з цим, в серіях елементів може бути відсутній окремий демультиплексор, а дешифратор зі входом дозволу роботи може називатись *дєшифратор-дємультиплексор.*

За допомогою мультиплексора можлива побудова універсального логічного модуля, тобто схеми, котра може виконувати будь яку логічну функцію. Це можна зробити декількома способами, найпростіший з яких полягає у «фіксації» інформаційних входів мультиплексора. Тобто на адресні входи необхідно подати змінні від яких залежить функція, а на інформаційні входи - значення функції. Дійсно, при кожній комбінації змінних до виходу мультиплексора буде підключений відповідний інформаційний вхід, і якщо на нього подане значення функцій яке відповідає такій комбінації змінних, то мультиплексор буде реалізовувати відповідну логічну функцію. На рис. 4.3 показана реалізація функції нерівнозначної $x_0 \oplus x_1$ за допомогою мультиплексора.

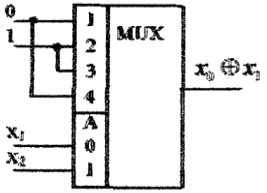


Рис. 4.3 – Реалізація логічної функції за допомогою мультиплексора

4.4. Порядок виконання роботи

1. Ознайомитись з робочим лабораторним місцем, та методичними вказівками.

2. В середовищі Proteus створити новий проект в якому вибрати та розмістити з бібліотеки досліджувані мікросхеми

мультиплексорів на робочий стіл. Підключити до них з меню ***Instruments*** генератор прямокутного сигналу (***Generator*** – Pulse, DC або DPattern, частота 1Hz). Постійна напруга +5V відповідає цифровій одиниці «1», від 0V – «0».

Далі для кожного цифрового елемента та мікросхеми в зошиті складаємо таблицю істинності (переходів) для усіх можливих вхідних комбінацій цифрового сигналу адреси A: наприклад 00, 01, 10, 11.

Для дослідження мікросхем в бібліотеці обираємо ***Category*** – TTL 74F series. Мультиплексор – MUX (наприклад mux74151).

2. Вибрати з бібліотеки та дослідити таблицю переходів для демультиплексора.

3. Дослідити робочі діапазони напруг цифрового нуля та одиниці для мікросхем різних технологій виготовлення (TTL, CMOS).

4.5. Обробка результатів

1. За результатами виконання п. 1 завдання визначити функціональне призначення входів мультиплексора, визначити його повні функціональні можливості та функціональні типи входів і виходів.

2. За результатами п. 1–3 визначити принцип дії мультиплексора та особливості функціонального призначення входів керування.

4. За результатами п.3 і довідковим даним зробити обґрунтовані висновки.

5. Реалізувати задану логічну функцію на дискретних елементах мікросхем малої інтеграції.

6. Реалізувати на мультиплексорі, що досліджується, мультиплексор з більшою кількістю інформаційних входів у відповідності до індивідуального завдання.

4.6. Зміст звіту. Звіт повинен містити:

1. Структурні схеми досліджуваних пристроїв.
2. Результати досліджень, розрахунків.
3. Зробити висновки: про об’єм виконаної роботи, чи досягнена мета роботи, чи практичні результати співпали з теоретичними.

4.7. Контрольні тестові питання

1. Як сформулювати в словесній формі принцип роботи мультиплексора?
3. Як дати в словесній та аналітичній формах визначення принципів функціонування демультиплексорів?
4. Які функціональні можливості мають сучасні інтегральні мультиплексори?
5. Як реалізувати мультиплексор на 17 входів на основі 4-х входового мультиплексора?
6. Як реалізувати на мультиплексорі задану логічну функцію?
7. Яке застосування мультиплексорів в цифрових пристроях?
8. Які принципи побудови мультиплексорів-селекторів?
9. Які принципи побудови демультиплексорів?
10. Яке застосування демультиплексорів в цифрових пристроях?

Лабораторна робота №5
Синтез логічних схем малої інтеграції в ПЛІС в графічному режимі з використанням САПР Altera QUARTUS II

5.1. Навчальні питання

1. Синтез логічних пристроїв за таблицею істинності.
2. Використання карт Карно.
3. Використання Altera Quartus II для синтезу логічних схем.

5.2. Навчальна мета

Метою роботи є знайомство з САПР для ПЛІС Altera Quartus II, а також вивчення способів синтезу логічних схем і мінімізації логічних функцій.

Посилання на сайт виробника техніки та програмного забезпечення, де можна скачати програму Quartus Prime наприклад Version… Lite Edition, рис. 5.1: https://www.intel.com/content/www/us/en/collections/products/fpga/software/downloads.html

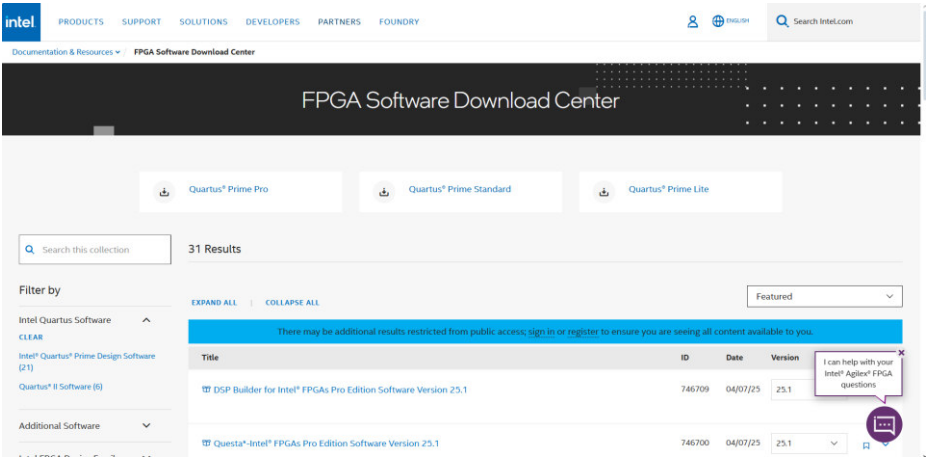


Рис. 5.1. Сайт Intel, виробника і софта для FPGA

В Quartus Prime Lite Edition скачати: Quartus Prime (includes Nios II EDS), ModelSim-Intel FPGA Edition (includes Starter Edition), Cyclone IV device support.

5.3. Теоретичні відомості

Способи синтезу логічних схем

Будь-яка логічна схема без пам'яті повністю описується таблицею істинності. Ця таблиця є вихідною інформацією для синтезу схеми на основі логічних елементів "І", "АБО", "НІ". Для розробки потрібного цифрового приладу спочатку на основі таблиці істинності записують його логічний вираз. Потім з метою спрощення цифрового пристрою мінімізують його логічний вираз і розробляють схему, яка виконує отриманий логічний вираз. Логічний вираз можна отримати двома способами:

- на основі досконалої диз'юктивної нормальної форми;
- на основі досконалої кон'юктивної нормальної форми.

Досконала диз'юктивна нормальна форма (ДДНФ)представлена сумою груп. Кожна група являє собою***мінтерм*** – добуток, в який входять всі змінні.

Наприклад,

$$F(A,B,C) = \bar{A} \cdot B \cdot C + A \cdot B \cdot \bar{C} + A \cdot B \cdot C$$

Досконала кон'юктивна нормальна форма (ДКНФ) представляється добутком груп. Кожна група являє***макстерм*** – сума, в яку входять всі змінні

Наприклад,

$$F(A,B,C) = (\bar{A} + B + C) \cdot (A + \bar{B} + C) \cdot (A + B + \bar{C})$$

Якщо схема має декілька виходів, то кожен вихід описується своєю функцією. Така система функцій називається системою власних функцій.

Приклад:

Таблиця 5.1

Таблиця істинності для функцій трьох змінних

A	B	C	Y	Мінтерм	Макстерм
0	0	0	0	$\bar{A}\bar{B}\bar{C}$	$A+B+C$
0	0	1	1	$\bar{A}\bar{B}C$	$A+B+\bar{C}$
0	1	0	0	$\bar{A}B\bar{C}$	$A+\bar{B}+C$
0	1	1	1	$\bar{A}BC$	$A+\bar{B}+\bar{C}$
1	0	0	0	$A\bar{B}\bar{C}$	$\bar{A}+B+C$
1	0	1	0	$A\bar{B}C$	$\bar{A}+B+\bar{C}$
1	1	0	1	$AB\bar{C}$	$\bar{A}+\bar{B}+C$
1	1	1	1	ABC	$\bar{A}+\bar{B}+\bar{C}$

ДДНФ складається на основі таблиці істинності шляхом додавання (логічне "АБО") всіх тих мінтермів, для яких вихід *Y* має значення 1.

$$Y = \bar{A}\cdot B\cdot C + \bar{A}\cdot B\cdot C + A\cdot B\cdot \bar{C} + A\cdot B\cdot C$$

ДКНФ складається на основі таблиці істинності шляхом добутку (логічне "І") всіх макстермів, для яких вихід Y має значення "0".

$$Y = (A+B+C)\cdot(A+\bar{B}\cdot C)\cdot(\bar{A}+B+C)\cdot(\bar{A}+B+\bar{C})$$

На основі отриманих виразів можна скласти схему пристрою, що реалізує задану функцію. Схема пристрою, отримана на основі ДДНФ, зображена на рис. 5.2, а, на основі ДКНФ на рис. 5.2, б.

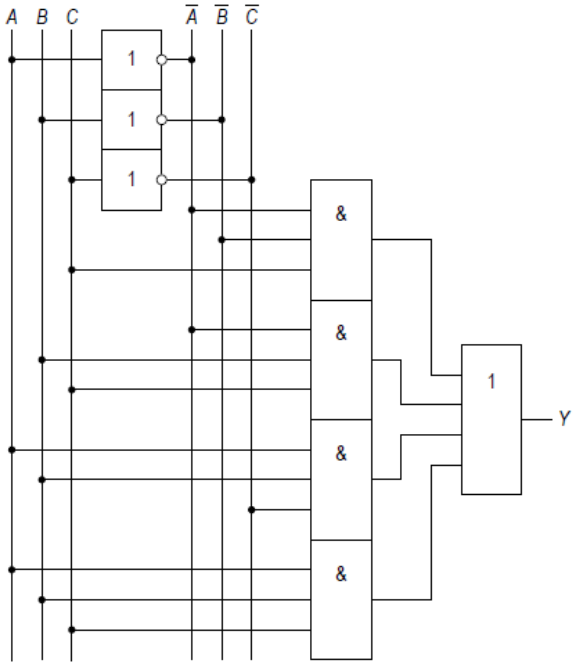


Рис. 5.2, а – Схема приладу, отриманого на основі ДДНФ

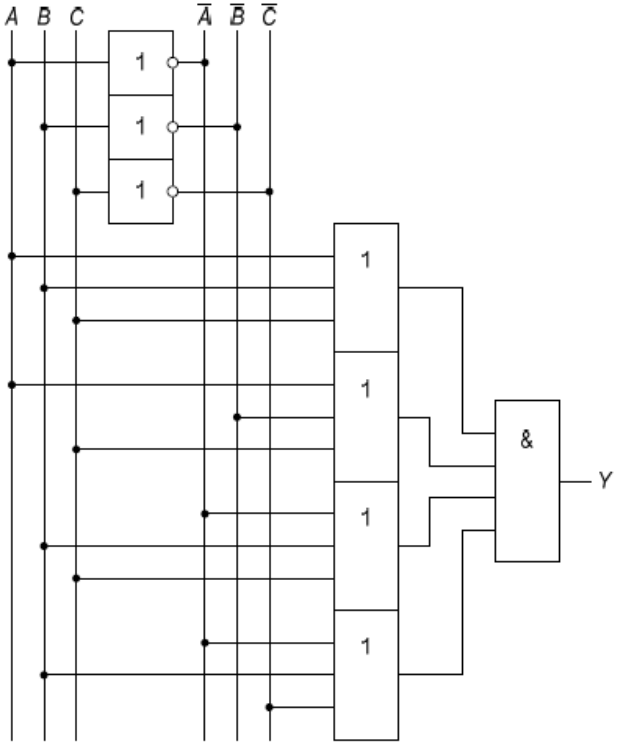


Рис. 5.2, б – Схема приладу, отриманого на основі ДКНФ

Карти Карно

З метою спрощення схеми цифрового приладу використовують мінімізацію функції. ***Карти Карно*** є наглядним методом для спрощення булевих рівнянь.

На рис. 1.3 показана карта Карно для функції, описаній в табл. 1.1. Верхній рядок карти дає 4 можливих значень для змінних *A*і *B*. Ліва колонка дає 2 можливих значень змінної *C*. Кожна клітинка карти Карно відповідає рядку таблиці істинності і містить значення

функціїY із цього рядка. Наприклад, верхня ліва клітинка відповідає першому рядку таблиці істинності і показує, що значення функції *Y*будедорівнювати0, коли *ABC* = 000. Як і кожний рядок в таблиці істинності, кожна клітинка карти Карно є визначений мінтерм.

Кожна клітинка (мінтерм) відрізняється від сусідньої зміною тільки однієї змінної. Це означає, що сусідня клітинка відрізняються тільки в значені одного літерала, значення якого "істина" в одній клітинці і "обман" в сусідній. Змінні Аі В комбінуються в верхньому рядку у вигляді ***коду Грея***: 00, 01, 10, 11. На відміну від бітового порядку по зростанню величини (00, 01, 10, 11), в коді Грея сусідні записи відрізняються тільки на один розряд.

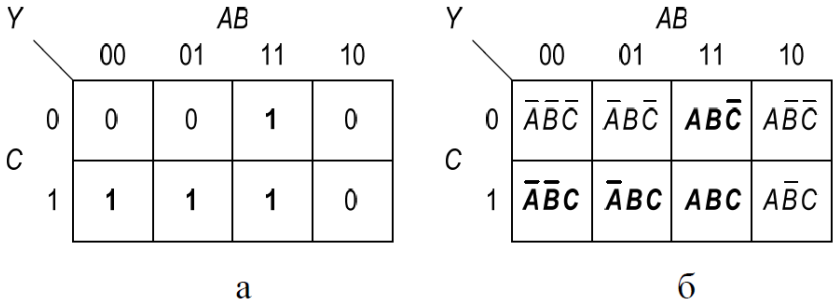


Рис.5.3 – Функція трьох змінних:
а – карта Карно, б – ката Карно з мінтермами

Карти Карно також "кільцева". Клітинка з самого правого краю таблиці є сусідньою із самою лівою, так як вона відрізняється тільки в одній змінній (*A*). Можна згорнути карту в циліндр, з'єднавши краї, і навіть в цьому випадку сусідні клітинки також будуть відрізнятися тільки в одній змінній.

На карті Карно рис. 1.3 містить чотири одиниці, що відповідають числу мінтермів в рівнянні. Читання мінтермів із карт Карно в точності відповідає читанню ДДНФ із таблиці істинності.

Карти Карно допомагають спрощувати графік, обводячи одиниці в сусідніх клітинках овалами, як це показано на рис. 1.4. Для кожного овалу пишуть і відповідну йому імпліканту. Змінні, для яких пряма і комплементарна форми попадають в один овал, виключаються.

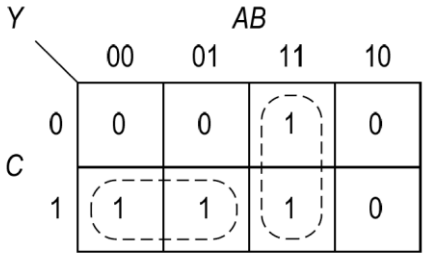


Рис. 5.4 – Мінімізація за допомогою карти Карно

Таким чином, отримуємо функцію $Y = \bar{A}C + AB$, на основі якої створимо нову схему пристрою (рис. 1.5).

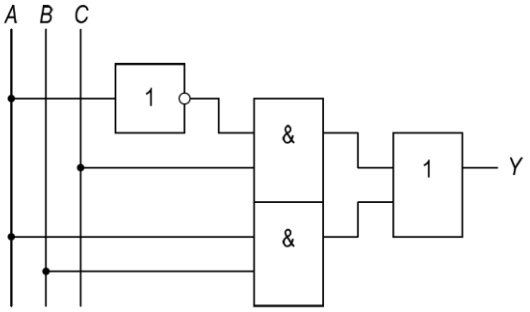


Рис. 5.5 – схема приладу, отриманого після мінімізації логічних функцій

Порівняйте з рис. 1.1 і 1.2, схема вийде на багато компактнішою.

Правила для знаходження мінімального рівняння із карт Карно наступні:

- Використовується менше всього овалів, необхідних для покриття всіх 1;
- Всі клітинки в кожному овалі обов'язково містять 1 або значення X);
- Кожен овал повинен охоплювати блок, число клітинок якого в кожному напрямку рівне степеню двійки (тобто 1, 2, 4, 8 і т.д.);
- Об'єднати клітинки можна тільки по горизонталі чи вертикалі;
- Кожен овал повинен бути на стільки великим, на скільки це можливо;
- Овал може зв'язувати краї карти Карно;
- Одиниця на карті Карно може бути обведена безліч разів, якщо це дозволяє зменшити число овалів, які будуть використовуватися.

5.4. Порядок виконання роботи

1. Синтезувати логічну схему на основі дворівневої логіки (ДКНФ або ДДНФ) за таблицею істинності свого варіанту (див табл. 5.3).
2. Запустити в середовище Altera Quartus II і створити новий проект (меню File → New Project Wizard…). Майстер створення проекту містить 5 кроків:

- Крок 1 – ім'я проекту і директорія його зберігання (рис. 5.6).

Увага! Не допускати використання в іменах файлів проекту і в шляху їх зберігання кириличних символів. Для зберігання кожного проекту використовувати окрему папку на диску.

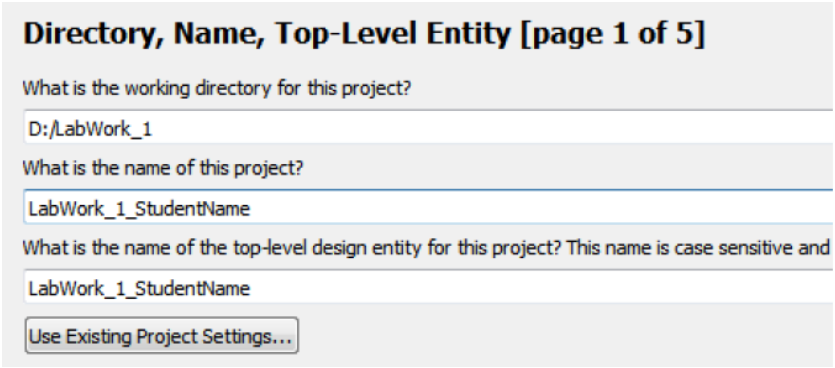


Рис. 5.6 – Вибір директорії й імені при створенні проекту

- Крок 2 – Додавання в проект других необхідних файлів. В даній роботі етап необхідно пропустити.
- Крок 3 – Вибір сімейства і назви ПЛІС, під яку створюється проект. Необхідно вказати сімейство Cyclone IV E, назву EP4CE115F29C7. Щоб прискорити процес вибору типу ПЛІС, можна використати фільтри по корпусу, числу виводів, класу швидкості, а також по імені, ввівши символи, що входять до назви (див. рис. 5.7).

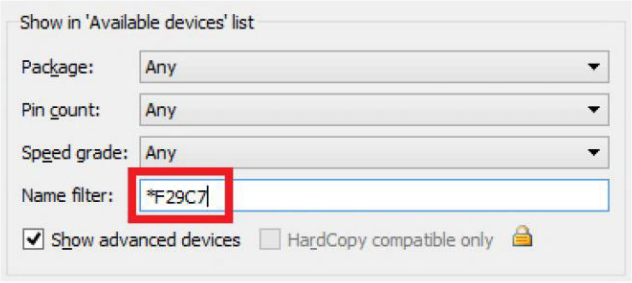


Рис. 5.7 – Використання фільтра по імені при виборі типу ПЛІС

- Крок 4 – Вибір інструментів для синтезу, налагодження і симуляції. В даній роботі етап необхідно пропустити.

- Крок 5 – На останньому кроці відображається сумарна інформація по проекту. Для завершення створення проекту потрібно натиснути кнопку "Finish".

3. Добавити в проект новий файл (File → New…) типу "Block Diagram/Schematic File" і накреслити синтезовану схему, використовуючи інструменти "Orthogonal Node Tool" (з'єднувальні провідники),"Pin Tool" (вхідні і вихідні порти), "Symbol Tool" (логічні елементи). Altera Quartus IIуже містить в своїй бібліотеці стандартні символи простих логічних елементів ("and", "ar4", "not"і т.д.) (див. рис. 5.8).

4. Зберегти файл, не змінюючи назви файлу, і виконати аналіз і синтез проекту (Processing → Start →Start Analysis & Synthesis). У випадку відсутності помилок зробити призначення виводів (меню Assignments → Pin Planner) на полі Location нижній частині вікна Pin Planner відповідно до першого і другого рядка табл. 5.2.

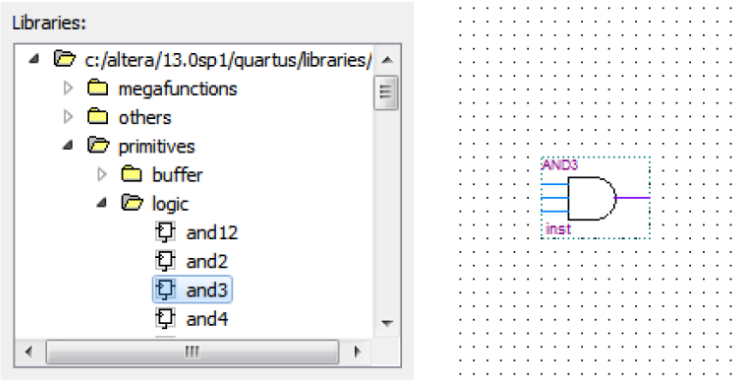


Рис. 5.8 – Бібліотека стандартних логічних елементів в Altera Quartus II

Для того щоб отримати часову діаграму потрібно (https://www.youtube.com/watch?v=a8JAkKhxlQI) вибрати File → New…University Program VWF. Далі натиснути правою кнопкою на білому полі ліворуч і вибрати Insert Node or Bus + Node Finder. Далі

List, виділити усі піни і натиснути в центрі на « > ». + Ok, Ok. Далі лівою миші на тискаємо на pin in і в меню вибираємо Edit – Value – Overwrite clock, вибираємо період прямокутного сигналу. Edite – Set End time (us - мікросекунди) – можна виставити тривалість симуляції. В кінці Simulation – Run functional simulation.

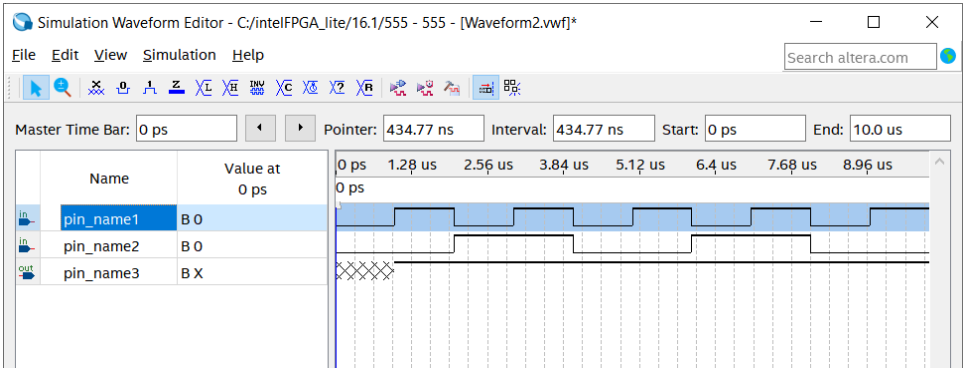


Рис. 5.9 – Вікно часової діаграми (University Program VWF)

5. Виконати повну компіляцію проекту (Processing → Start Compilation). В процесі компіляції створюється конфігураційний файл *.sof в папці з проектом в підпапці "output_files". Він використовується при програмуванні ПЛІС.

Таблиця 5.2

Назви виводів схеми

Сигнал	A	B	C	Y
Вивід ПЛІС	AC27	AC28	AB28	G19

6. В присутності викладача підключити лабораторний стенд до комп’ютеру і включити на стенді живлення.

7. Запрограмувати навчальний стенд (меню Tools → Programmer).

8. Скласти мінімізовану логічну функцію за таблицею істинності свого варіанту (див. табл. 5.3) за допомогою карт Карно. Синтезувати схему за мінімізованою функцією.

9. Створити нову схему в Altera Quartus II в тому ж файлі і виконати аналіз і синтез проекту. Потім зробити призначення виводів і повну компіляцію проекту.

10. Виконати аналіз правильності роботи схеми за часовою діаграмою вхідних і вихідного портів.

11. Зробити висновки по роботі.

Таблиця 5.3

Варіанти завдань

Варіанти			**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
A	**B**	**C**	**Y**														
0	0	0	0	1	0	0	1	0	1	0	1	0	1	0	0	0	1
0	0	1	1	0	0	1	0	0	0	1	1	1	0	1	0	1	1
0	1	0	0	1	1	1	1	0	1	0	0	1	0	0	1	0	0
0	1	1	1	1	1	0	0	1	0	1	0	0	1	0	0	1	0
1	0	0	1	0	1	0	1	1	1	0	0	0	1	1	1	0	1
1	0	1	1	0	0	1	0	1	1	1	0	0	1	0	1	0	1
1	1	0	0	1	1	0	0	0	0	1	1	1	0	1	1	1	0
1	1	1	0	0	0	1	1	1	0	0	1	1	0	1	0	1	0

5.5. Вміст звіту

1. Мета роботи.
2. Задана таблиця істинності і синтезована схема.
3. Карта Карно, синтезована логічна функція і схема.
4. Висновки по роботі.

5.6. Контрольні запитання

1. Що таке ДДНФ/ДКНФ?
2. Як записати ДДНФ/ДКНФ, використовуючи таблицю істинності?

3. Як краще синтезувати логічний пристрій (на основі ДДНФ або ДКНФ), якщо значення функції в таблиці істинності мають більше нулів, ніж одиниць?

4. Як розробити логічний пристрій, якщо він має декілька виходів?

5. Що таке мінімізація логічного виразу, і які існують способи мінімізації?

6. Яким чином здійснюється мінімізація функції за допомогою карти Карно?

7. Розкажіть, як визначити таблицю істинності логічного пристрою експериментально, використовуючи лабораторний стенд?

8. Написати таблицю істинності для основних логічних вентилів (І, АБО, НІ, виключне АБО).

9. Що таке проект Altera Quartus II? Для чого необхідна компіляція проекту в Altera Quartus II і як її виконати?

10. Для чого проводиться операція призначення виводів в Altera Quartus II?

Лабораторна робота №6
Синтез логічних схем перетворювача кодів та дешифратора в ПЛІС мовою Verilog або VHDL з використанням САПР Altera QUARTUS II

6.1. Навчальні питання

1. Синтез перетворювача кодів та дешифратора в ПЛІС.
2. Використання мови програмування VHDL.
3. Використання Altera Quartus II для синтезу логічних схем.

6.2. Навчальна мета

Метою роботи являється вивчення принципів дії комбінаційних схем: дешифратора, коду для семисегментного індикатора, суматора, а також знайомство з мовами проектування апаратури.

6.3. Короткі теоретичні відомості

Мови проектування апаратури

VHDL* – мова опису апаратури інтегрованих схем. VHDL є базовою мовою при розробці апаратури сучасних обчислюваних систем. Був розроблений в 1983 р. по замовленню Міністерства оборони США з метою формального опису логічних схем для всіх етапів розробки електронних систем, починаючи з модулів мікросхем і закінчуючи великими обчислювальними системами.

Програма на VHDL складається із одного чи декількох файлів. В одному файлі розміщується одна чи декілька пар елементів "об'єкт проекту – архітектура об'єкту". *Об'єкт проекту* – це частина інтерфейсу, в якій вказана інформація про те, як ввімкнути даний об'єкт в середині іншого об'єкту, який знаходиться на більш високому рівні ієрархії. А в *архітектурі об'єкту* описується алгоритм його функціонування. У одного об'єкт може містити декілька архітектур, відповідаючи різним алгоритмам функціонування, проектам на різних етапах проектування.

В прикладах цього і подальшого коду *курсивним шрифтом* вказані коментарі, **напівжирним** – ключові слова даної мови.

Об’єкти проекту виглядають наступним чином:

Опис архітектури:

```
Library std;                    --підключення бібліотеки STD

use std.standard.all;           -- використовуються всі типи
                                пакета
entity FUNC_NAND is
                                    STANDART бібліотеки STD
    port (
                                -- назва об’єкту - FUNC_NAND
        A : in BIT;
                                -- оголошення портів
        B : in BIT;
                                -- вхідний порт A типуBIT
        C : out BIT);
                                -- вхідний порт B типуBIT
End FUNC_NAND;
                                -- вихідний порт C типуBIT

                                -- кінець оголошення об’єкту
```

Ключове слово *port* відкриває опис входів-виходів (портів) об’єкти. Слово *in*вказує на вхід, а *out*– на вихід. BIT– це тип порту, який, згідно визначенню цього типу в пакеті *standard*, приймають значення 0 і 1.

Опис архітектури:

```
Architecture BEHAVIORAL of      -- початок архітектури
FUNC_NAND is                    BEHAVIORAL об’єкту
```

FUNC_NAND

-- початок описової (__) частини архітектури
Begin

C <= **not** (A **and** B);
-- паралельний оператор

End BEHAVIORAL;
-- кінець архітектури

Розділ архітектури складається із декларативної і описової частин. В декларативний частині оголошуються використані в середині сигналу, константи спеціальні (власні) типи, атрибути, процедури і функції. Описова частина складається із списку паралельних операторів. Всі паралельні оператори виконуються одночасно, їх порядок в списку не має значення.

Verilog*, Verilog HDL – ще одна мова описання архітектури, що використовується для описання і моделювання електронних систем. Verilog була створена в 1984 році фірмою Automated Integrated Design Systems (пізніше перейменований на GatewayDesignAutomation) як власний засіб моделювання. Після поглинання останньої фірмової Caddence, мова почала отримувати все більший попит серед розробників і стала не менш популярною, чим VHDL.

Синтаксис Verilog дуже схожий га синтаксис мови С, що спрощує його освоєння. Препроцесор Verilog дуже схожий на препроцесор мови С, і основні управляючі конструкції *if, while* та інші подібні однойменні конструкції мови С. Програма на Verilog зазвичай складається із декількох модулів. Синтаксис оголошення модуля виглядає наступним чином:

```
Module FUNC_NAND (A, B, C);        // Вхідні пори

        Input A, B;                // Вихідні порти

        output C;                  // операція і-ні

        assign C = ~ (A & B);

end module
```

У заголовку модуля описується список портів. У тілі модуля описується його функціональність.

У проекті, особливо в складному, буває багато модулів, з'єднаних між собою. Перш за все, потрібно помітить, що зазвичай в проекті завжди є один *модуль самого верхнього рівня (top level).* Він складається із декількох інших модулів. Ті в свою чергу можуть містити ще модулі і так далі. Необов'язково, щоб всі модулі були написані на одній мові описання апаратури. Зовсім навпаки. Досить зручно і наочно мати модуль самого верхнього рівня виконаним у вигляді схеми, що складається із модулів більш низького рівня. Ці модулі можуть бути написані різними розробниками,на різних мовах (Verilog, VHDL) і навіть виконанні у вигляді схеми.

System Verilog являє собою надмножину мови Verilog стандарту 2005 року, але з великими можливостями для верифікації і моделювання розробок.

Примітка – при називанні програмних об'єктів (проектів, інтерфейсів, архітектур, пакетів, сигналів, змінних, констант, компонентів і т.д.) в коді потрібно притримуватись правил, наведених в Програмі А даного посібника.

6.3.1. Дешифратор (декодер)

В спільному випадку *дешифратор* (декодер) – це пристрій, що перетворює цифровий сигнал в якому-небудь кодуванні в іншу, не закодовану форму. *Класичний дешифратор* представляє собою пристрій для перетворення N-розрядного позиційного двійкового коду в одиничний вихідний сигнал на одному із 2^N виходів. При кожній вихідній комбінації сигналів на одному із виходів з'являється логічна 1. Таким чином, по одиничному сигналів на одному із виходів можна судити про вхідній кодовій комбінації. Таблиця істинності класичного дешифратора з двома входами зображена в табл. 6.1.

Табл. 6.1

Таблиця істинності двохрозрядного дешифратора

X1	X0	Y3	Y2	Y1	Y0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Для побудови схеми дешифратора по таблиці істинності використаємо методику з лабораторної роботи №1. Наприклад, пристрій повинен мати чотири виходи. Для кожного виходу записуємо логічний вираз. На основі ДДНФ:

$$Y0 = \overline{X1} \cdot \overline{X0}, \qquad Y1 = \overline{X1} \cdot X0, \qquad Y2 = X1 \cdot \overline{X0}, \qquad Y3 = X1 \cdot X0$$

За цією системою виразів не складно побудувати схему потрібного дешифратора (рис. 6.1).

6.3.2. Перетворювач коду для семисегментного індикатора

Найбільш широкими перетворювачами кодів відомі у використанні до цифрових індикаторів. Наприклад, часто потребуються перетворювачі позиційного двійкового коду в десяткові цифри.

За допомогою семисегментного індикатора (рис. 6.2а) можна висвітити десяти арабських цифр (рис. 6.3). Такі індикатори виготовляються в двох варіантах – зі спільним анодом (ОА, рис. 6.2б) і зі спільним катодом (ОК, рис. 6.2в). Слід помітити, що "запалювання" сегменту індикатора з ОА виконується логічним нулем, ф з ОК – логічною одиницею.

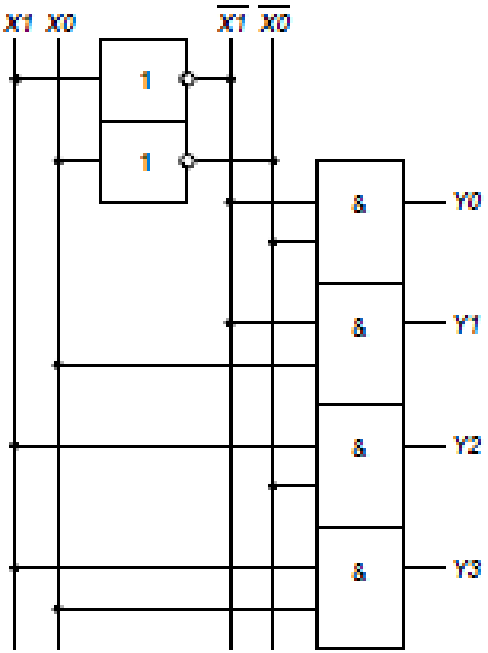


Рис. 6.1 – схема дешифратора

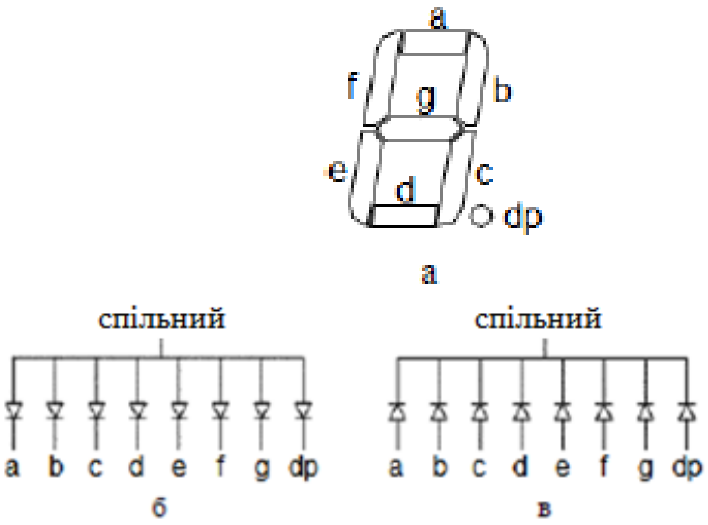


Рис. 6.2 – Семисегментний індикатор з точкою:
а – зовнішній вигляд, б – схема зі спільним анодом, в – схема зі спільним катодом

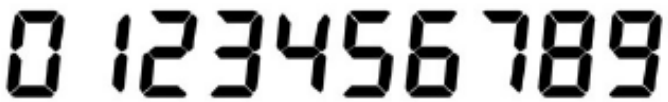


Рис. 6.3 – Приклад відображення цифр на семисегментному індикаторі

Очевидно, що двійковий код повинен мати не менше 4-х розрядів (2^4 = 16, що більше 10). Умовне графічне позначення представлене на рис. 6.4.

Таблиця 6.2

Таблиця істинності перетворювача колу

Цифра	D3	D2	D1	D0	Ha	Hb	Hc	Hd	He	Hf	Hg
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
-	Всі інші комбінації				0	0	0	0	0	0	1

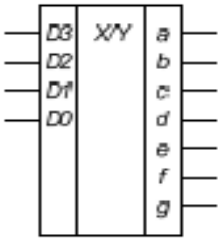


Рис.6.4 –Умовне графічне позначення перетворювача коду

6.3.3. Суматор

Напівсуматор має два входи (*A* і *B*) і виходи (*S* і $C_{вих.}$). S – це сума *A* і *B*. Якщо ж і *A*, і *B* дорівнюють 1, тоді вихід *S* повинен дорівнювати 2, але таке число не може бути представлене у вигляді одного двійкового розряду. В цьому випадку результат вказується разом з переносом $C_{вих.}$ в наступний рядок. Напівсуматор може бути збудований із елементів XOR ("виключне АБО") і AND ("логічне І"), рис. 6.5.

Таблиця 6.3

Таблиця істинності напівсуматора

A	*B*	$C_{вих.}$	*S*
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = A \oplus B \qquad C_{вых} = AB$$

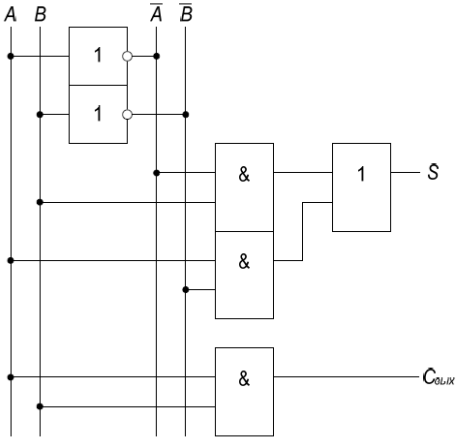


Рис. 6.5 – Схема напівсуматора

Повний суматор окрім виходу переносу має вхід переносу з попереднього розряду складань $C_{вх.}$.

$$S = A \oplus B \oplus C_{вх}$$

N-розрядний суматор складає два *N*-розрядних числа (*A*і*B*), а також вхідний переніс $C_{вх.}$ І формує *N*-розрядний результат *S* і вихідний переніс $C_{вих.}$. Такий суматор називається ***суматором із переносом, що розповсюджується,*** так як вихідний перенос одного розряду переходить в наступний рядок.

Таблиця 6.4

Таблиця істинності повного суматора

$C_{вх.}$	A	B	**$C_{вих.}$**	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

6.4. Порядок виконання роботи

Запустити середовище Altera Quartus II і створити новий проект параметрами:

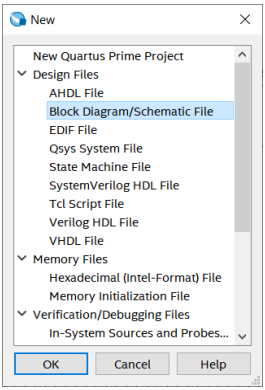
- Назва проекту: LabWork_2_StudentName
- Сімейство ПЛІС: Cyclone IV E
- Тип ПЛІС:EP4CE115F29C7

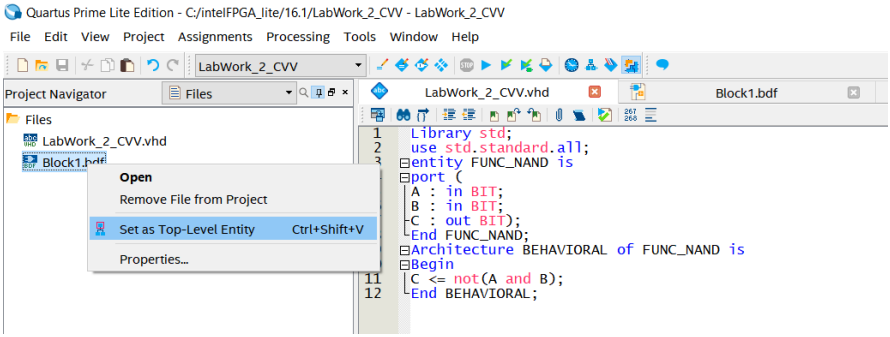
Інше залишити по замовчуванню.

1. Добавити в проект три нові файли типу "VHDL File" (або "Verilog HDL File" якщо пишеться програма мовою Verilog) і описати схеми дешифратора 2×4, перетворювача коду для семисегментного індикатора і повного суматора.

Для цього в головному меню натискаємо: File – New – VHDL File + Ok. У відкритому вікні набираємо текст програми мовою VHDL.

2. Додати до проекту файл типу "Block Diagram/Schematic File" і назначити його файлом верхнього рівня (Project → Set as Top-Level Entity):

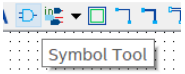


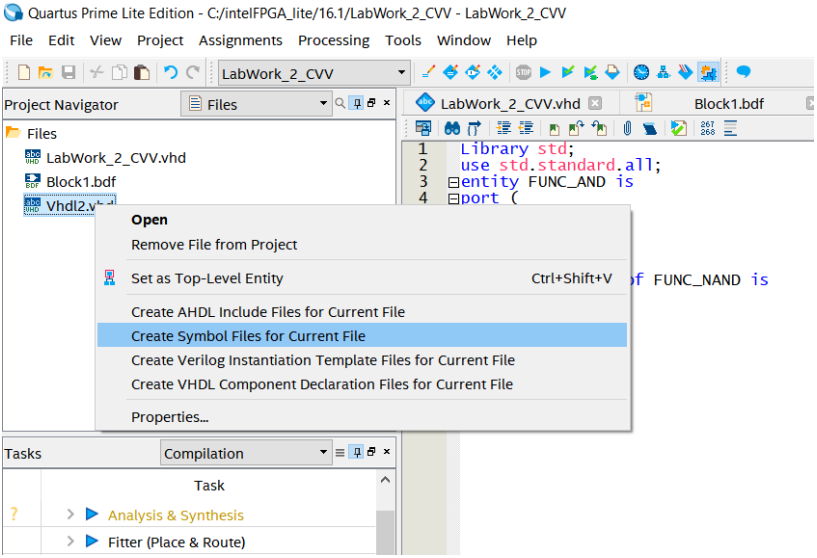


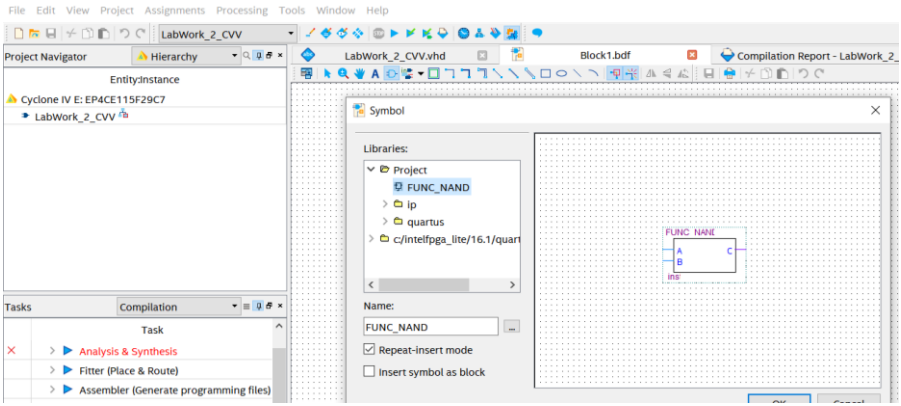
3. Виконати аналіз і синтез проекту.

4. Виконати генерацію схемного синтезу символу із HDL-опису для кожного модуля окремо(File → Create/Update → Create Symbol Files from Current File):

5. Після цієї операції символи стануть доступні в бібліотеці інструментів "Symbol Tool" в підпапці "Project":







6. В файлі схеми, створеному в п. 3, накреслити спільну схему, що містить в собі всі досліджувані модулі.

7. Виконати аналіз і синтез проекту і призначити виводи у відповідності до таблиць 6.5–6.7. Після цього виконати повну компіляцію проекту.

Таблиця 6.5

Призначення виводів дешифратора 2×4

Сигнал	**X1**	**X0**	**Y3**	**Y2**	**Y1**	**Y0**
Вивід ПЛІС	AC27	AC28	F21	E19	F19	G19

Таблиця 6.6

Призначення виводів перетворювача коду для семисегментного індикатора

Сигнал	**D3**	**D2**	**D1**	**D0**
Вивід ПЛІС	Y23	Y24	AA22	AA23

Сигнал	**Ha**	**Hb**	**Hc**	**Hd**	**He**	**Hf**	**Hg**
Вивід ПЛІС	AD17	AE17	AG17	AH17	AF17	AG18	AA14

Таблиця 6.7

Призначення виводів повного суматора

Сигнал	**$C_{вх}$**	**A**	**B**	**$C_{вих}$**	**S**
Вивід ПЛІС	AC26	AB27	AD27	E18	F18

8. Підключити до комп'ютера лабораторний стенд, ввімкнути його і запрограмувати. Подаючи різні вхідні сигнали і спостерігати за станом індикаторів, проаналізувати роботу досліджуваної схеми.

6.5. Вміст звіту

1. Мета роботи.
2. Схеми дослідження дешифратора, перетворювача коду для семисегментного індикатора, суматора.
3. Вихідні коди модулів дешифратора, перетворювача коду для семисегментного індикатора, суматора.
4. Таблиця істинності для кожної схеми.
5. Висновки по роботі.

6.6. Контрольні запитання

1. Поясніть принцип роботи дешифратора і шифратора.
2. Як синтезувати дешифратор із заданою розрядністю?
3. Як працює перетворювач коду для семисегментного індикатора? Як влаштований семисегментний індикатор?
4. Як працює мультиплікатор і демультиплікатор?
5. Як працює повний суматор? Чим він відрізняється від напівсуматора?
6. Що таке біт переносу в суматорі?
7. Що таке файл верхнього рівня в Altera Quartus II?
8. Опишіть структуру VHDL-файлу (Verilog-файлу).

Лабораторна робота №7
Дослідження тригерів та регістрів в ПЛІС в графічному режимі з використанням САПР Altera QUARTUS II

7.1. Навчальні питання

1. Синтез тригерів та регістрів в ПЛІС в графічному режимі.
3. Використання Altera Quartus II для синтезу логічних схем.

7.2. Мета роботи

Метою роботи є експериментальне дослідження роботи різних типів тригерів і регістрів.

7.3. Короткі теоретичні відомості

Тригери – прості послідовних схема, що запам'ятовують один біт інформації. Використання тригерів дозволяє реалізовувати пристрої оперативної пам'яті (тобто пам'яті, інформація в якій зберігається тільки на час обчислень). Однак тригери можуть використовуватися і для побудови деяких цифрових пристроїв з пам'яттю, таких як лічильники, перетворювачі послідовного коду в паралельний або цифрові лінії затримки.

Однією з найпростіших послідовних схем є RS-тригер, що складається, як показано на рис. 7.1, з двох перехресно включених елементів АБО-НЕ. У тригера є два входи – R і S і два виходи Q і $\overline{Q}$.

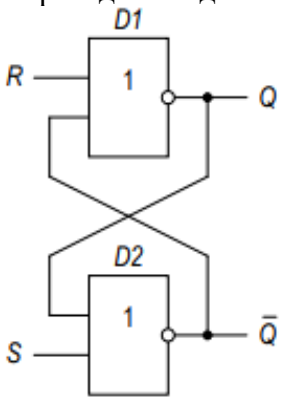


Рисунок 7.1 – Схема RS-тригера

Таблиця переходів, наведена нижче в табл. 7.1, ілюструє чотири можливих стану тригера. Входи R і S відповідають за скидання (reset) і установку (set) значення на виході Q відповідно.

Таблиця 7.1

Таблиця переходів RS-тригера

S	R	Q	$\overline{Q}$	Режим
0	0	$Q_{пред}$	$\overline{Q_{пред}}$	Режим зберігання
0	1	0	1	Режим установки 0
1	0	1	0	Режим установки 1
1	1	*	*	Заборонений режим

Заборонений режим не використовується, так як стану виходів тригера в загальному випадку не визначені і будуть залежати від його конкретної реалізації.

Розглянутий вище тригер є асинхронним.

У синхронного RS-тригера крім R і S входів є вхід синхронізації). Його схема побудови на основі асинхронного RS-тригера і умовне позначення показані на рис. 7.2.

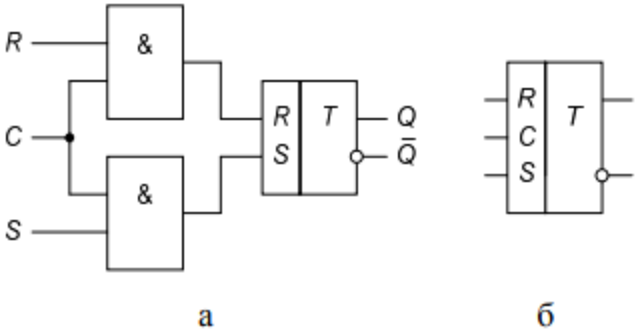


Рисунок 7.2 – Синхроний RS-тригер, а – схема, б – УГЗ

Стан виходів тактованих RS-тригера може змінюватися тільки в моменти приходу тактових імпульсів. В іншому випадку елементи І «замкнені» і не пропускають сигнали R і S. У цьому випадку говорять, що схема працює синхронно.

RS-тригер незручний через незвичайного поведінки в забороненому режимі. D-тригер (D-засувка, D-latch) (рис. 7.3) вирішує цю проблему. Він побудований на основі синхронного RS-тригера і має вхід даних D, що визначає, яким буде наступний стан, і вхід тактового сигналу C, таблиця 7.2.

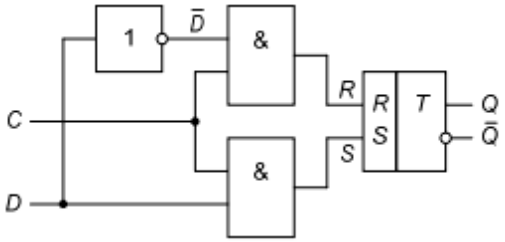


Рисунок 7.3 – Схема D-тригера

Таблиця 7.2

Таблиця істинності D-тригера

C	D	Q	$\bar{Q}$	Режим
0	X	$Q_{пред}$	$\overline{Q_{пред}}$	Режим зберігання
1	0	0	1	"Прозорий режим"
	1	1	0	

Коли C = 1, говорять, що D-тригер «прозорий», тобто він пропускає дані D на вихід Q, як якщо б він був звичайним буфером. Коли C = 0 - «непрозорий», тобто не пропускає нові дані з входу D на вихід Q, а Q зберігає своє попереднє значення. Умовне позначення D-тригера представлено на рис. 7.4

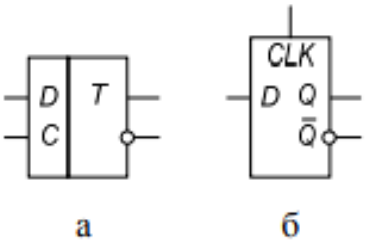


Рис. 7.4 – Умовне позначення D-тригера
а – вітчизняне, б – закордонне

Стан D-тригера змінюється безперервно, поки C = 1.

Такий триггер називається статичним. D-тригер, синхронізований фронтом (динамічний), може бути побудований з двох включених послідовно статичних D-тригерів, у яких тактові сигнали є булеві доповненнями один одного. Умовне позначення такого D-тригера наведено на рис. 7.5.

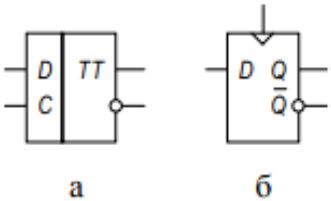


Рис. 7.5 – Умовне позначення D-тригера, який потрібно синхронізувати фронтом: а – вітчизняне, б – закордонне

Динамічний D-тригеркопіюєзначення з D на Q по задньому або передньому (залежить від реалізації тригера) фронту тактового імпульсу і пам'ятаєцей стан весь інший час. Таким чином вхід D визначає нове, майбутнє стан тригера, фронт визначає конкретний момент часу, коли стан буде оновлено.

Робота динамічного D-тригера, щопрацює по задньому фронту синхроімпульсів, представлена в таблиці 7.3.

Таблиця 3.3

Таблиця переходів динамічного D-тригера

C	D	Q	$\overline{Q}$	Режим
0	X	$Q_{пред}$	$\overline{Q_{пред}}$	Режим зберігання
1				
⌐				
⌐	0	0	1	Переключення
⌐	1	1	0	

Т-тригер - це рахунковий тригер. У даного тригера є тільки один вхід Т. Після надходження на цей вхід імпульсу, стан тригера змінюється на протилежне. Рахунковим він називається тому, що Т-тригер як би підраховує кількість імпульсів, що надійшли на його вхід. При отриманні другого імпульсу Т-тригер здатний відобразити тільки молодший розряд числа $2_{10} = 10_2$, тому на його виході знову виникає логічний нуль.

Т-тригер можна синтезувати з будь-якого двоступеневого тригера. Розглянемо приклад синтезу Т-тригера з динамічного D-тригера. Для того щоб перетворити D-тригер в рахунковий, необхідно ввести ланцюг зворотного зв'язку з інверсного виходу цього тригера на вхід, як показано на рис. 7.6.

Таблиця 7.4

Таблиця переходів Т-тригера

T	Q	$\overline{Q}$	Режим
0	$Q_{пред}$	$\overline{Q_{пред}}$	Зберігання
1			
⌐			
⌐	0	1	Переключення
⌐	1	0	

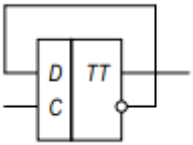


Рис. 7.6 – Схема Т-тригера на основі двоступеневого D-тригера

Існує ще одне подання Т-тригера. при розробці схем синхронних двійкових лічильників важливо здійснювати одночасний запис у всі його тригери. У цьому випадку вхід Т тригера служить тільки для дозволу зміни стану на протилежне, а синхронізація проводиться окремим входом С. Подібна схема Т-тригера приведена на рис. 7.7.

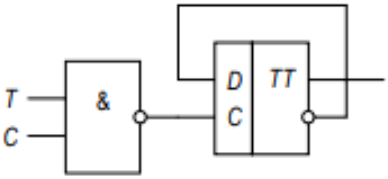


Рисунок 7.7 – Схема синхронного Т-тригера

Регістри

Регістри призначені для збереження і перетворення багаторозрядних двійкових чисел. ***N-розрядний регістр*** - набір з N тригерів із загальним тактовим сигналом. Таким чином, всі біти регістра оновлюються одночасно. Регістр є ключовим блоком при побудові більшості послідовних схем. На рис. 7.8 показана схема і позначення чотирьохрозрядного регістра зі входами D3..D0 і виходами Q3..Q0.

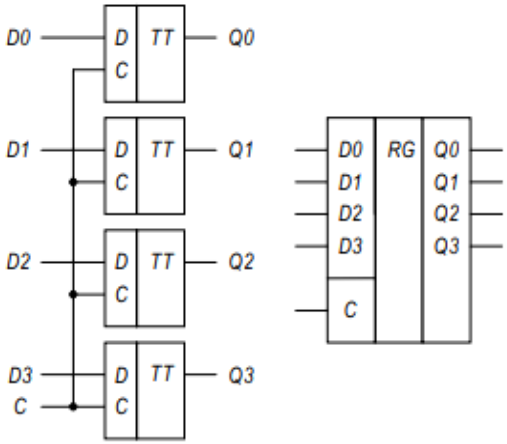


Рис. 7.8 – Чотирьохрозрядний регістр:
а – схема, б – умовне позначення

Зсувний регістр має вхід тактового сигналу, послідовний вхід Sin, послідовний вихід Sout і N паралельних виходів. По кожному задньому фронту тактового імпульсу в перший тригер регістра записується новий біт зі входу Sin а вміст наступних тригерів зсувається вперед.

Зсувний регістр може бути побудований з N послідовно з'єднаних тригерів (рис. 7.9). Такий реєстр можна розглядати як послідовно-паралельний перетворювач.

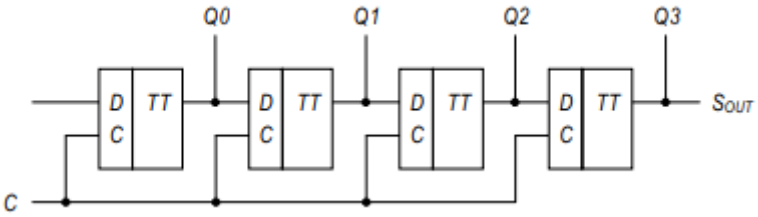


Рис. 7.9 – Схема 4-розрядного зсувного регістру

В паралельно-послідовний перетворювач паралельно завантажується N біт, які потім послідовно (по одному біти за раз) надходять на вихід.

7.4. Порядок виконання роботи

Запустити середовище Altera Quartus II і створити новий проект з параметрами:

- Ім'я проекту: LabWork_3_StudentName
- Сімейство ПЛІС: Cyclone IV E
- Тип ПЛІС: EP4CE115F29C7

Решту залишити за замовчуванням.

7.4.1. Дослідження RS-тригерів

1. Додати в проект новий файл схеми і накреслити схему асинхронного RS-тригера.

2. Виконати аналіз і синтез проекту і зробити призначення виводів відповідно до таблиці 7.5. Після цього виконати повну компіляцію проекту.

Таблиця 7.5

Призначення виводів асинхронного RS-тригера

Сигнал	R	S	Q
Вивід ПЛІС	AC27	AC28	G19
На стенді	SW2	SW1	LEDR0

3. Запрограмувати навчальний стенд і, спостерігаючи за станами світлодіодних індикаторів, проаналізувати роботу досліджуваної схеми.

4. Перетворити схему в синхронний RS-тригер, виконати аналіз і синтез проекту і зробити призначення виводів відповідно з таблицею 3.6. Після цього виконати повну компіляцію проекту.

Вхідний сигнал синхронізації виробляти з кнопки KEY3. При її натисканні надходить сигнал низького логічного рівня, а при відпуску - високого, тому в ланцюг сигналу синхронізації необхідно ввести інвертор.

Таблиця 7.6

Призначення виводів синхронного RS-тригера

Сигнал	C	R	S	Q
Вивід ПЛІС	R24	AC27	AC28	G19
На стенді	KEY3	SW2	SW1	LEDR0

5. Запрограмувати навчальний стенд і, спостерігаючи за станами світлодіодних індикаторів, проаналізувати роботу досліджуваної схеми.

7.4.2. Дослідження D-тригера

1. Накреслити схему динамічного D-тригера на основі стандартного бібліотечного елемента «DFF» (рис. 7.10).

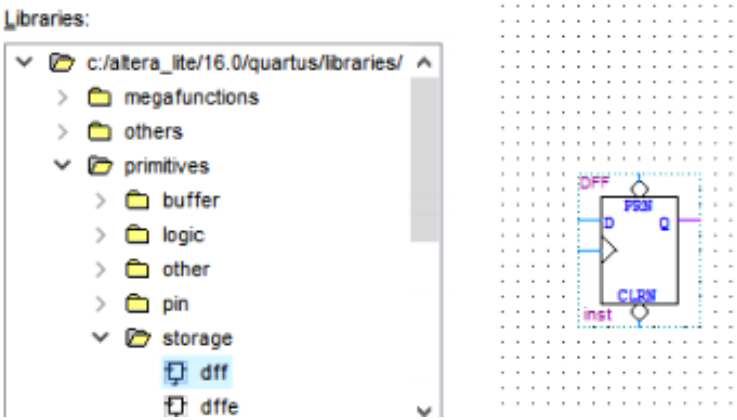


Рис. 7.10 – D-тригер в бібліотеці стандартних логічних елементів

2. Виконати аналіз і синтез проекту і зробити призначення виводів відповідно до таблиці 7.7. Після цього виконати повну компіляцію проекту.

Таблиця 7.7

Призначення виводів динамічного D-тригера

Сигнал	D	C	Q
Вивід ПЛІС	AC27	R24	G19
На стенді	SW2	KEY3	LEDR0

3. Запрограмувати навчальний стенд і, спостерігаючи за станами світлодіодних індикаторів, проаналізувати роботу досліджуваної схеми.

7.4.3. Дослідження Т-тригера

1. Накреслити схему Т-тригера на основі стандартного бібліотечного елемента «TFF».

2. Виконати аналіз і синтез проекту і зробити призначення виводів відповідно до таблиці 7.8. Після цього виконати повну компіляцію проекту.

Увага! При використанні введення інформації з кнопок лабораторного стенду виникає брязкіт контактів. Опис цього явища і спосіб боротьби з ним - модуль «button_debouncer» - наведені в Додатку А. Для того щоб використовувати модуль «Button_debouncer», необхідно перенести код, наведений в Додатку Б, в файл «debouncer.v», а потім включити його в проект. Для цього слід викликати меню «Assigment → Settings» і на вкладці Files додати файл з модулем - необхідно вказати шлях до файлу, натиснути кнопку «Add», потім «Apply» і «OK».

Таблиця 7.8

Призначення виводів Т-тригера

Сигнал	G_50MHz	T	C	Q
Вивід ПЛІС	Y2	AC27	R24	G19

3. Запрограмувати навчальний стенд. Проаналізувати роботу досліджуваної схеми.

7.4.4. Дослідження паралельного регістра

1. Накреслити схему паралельного 4-розрядного регістра (Див. рис. 7.8).

2. Виконати аналіз і синтез проекту і зробити призначення виводів відповідно до таблиці 7.9. Після цього виконати повну компіляцію проекту.

Таблиця 7.9

Призначення виводів паралельного регістра

Сигнал	**G_50MHz**	**C**	**D3**	**D2**	**D1**	**D0**
Вивід ПЛІС	Y2	R24	AD27	AC27	AC28	AB28
На стенді	50 MHz Oscillator	KEY3	SW3	SW2	SW1	SW0

Сигнал	**Q3**	**Q2**	**Q1**	**Q0**
Вивід ПЛІС	F21	E19	F19	G19

3. Запрограмувати навчальний стенд і по черзі зробити запис декількох 4-розрядних двійкових чисел в регістр.

Проаналізувати роботу досліджуваної схеми.

7.5. Дослідження регістру зсуву

1. Накреслити схему зсувного 4-розрядного регістра (рис. 7.9).

2. Виконати аналіз і синтез проекту і зробити призначення виводів відповідно до таблиці 7.10. Після цього виконати повну компіляцію проекту.

Таблиця 7.10

Призначення виводів паралельного регістра

Сигнал	**G_50MHz**	**C**	**S_{in}**
Вивід ПЛІС	Y2	R24	AB28
На стенді	50 MHz Oscillator	KEY3	SW0

Сигнал	**Q3**	**Q2**	**Q1**	**Q0**
Вивід ПЛІС	F21	E19	F19	G19

3. Запрограмувати навчальний стенд і, спостерігаючи за станами світлодіодних індикаторів, проаналізувати роботу досліджуваної схеми.

7.6. Зміст звіту

1. Мета роботи.

2. Схеми дослідження тригерів і регістрів і їх умовно-графічне позначення.

3. Таблиці переходів схем тригерів.

4. Висновки по роботі.

7.7. Контрольні запитання

1. Накреслити схему RS-тригера на логічних елементах АБО-НЕ і пояснити принцип його роботи.

2. Чим синхронний RS-тригер відрізняється від асинхронного? Як він реалізується?

3. Пояснити по таблиці переходів роботу D-тригера.

4. Якою характерною особливістю володіє періодична послідовність імпульсів на вході Т-тригера?

5. Якою перевагою володіє динамічний D-тригер?

6. Чим визначається розрядність регістрів?

7. Паралельний регістр: його схема і принцип роботи.

8. Послідовно-паралельний регістр: його схема і принцип роботи.

Лабораторна робота №8
Дослідження та синтез лічильників в ПЛІС в графічному режимі з використанням САПР Altera QUARTUS II

8.1. Навчальні питання

1. Дослідження побудови та функцій двійкового лічильника.
2. Дослідження складання та налагодження схем в середовищі Altera QUARTUS II.

8.2. Навчальна мета

Метою роботи є вивчення універсального двійкового лічильника і придбання навичок у побудові й експериментальному дослідженні лічильників.

8.3. Короткі теоретичні відомості

Лічильник – пристрій для підрахунку числа вхідних імпульсів. Вони використовуються для побудови схем таймерів або для вибірки інструкцій з ПЗУ в мікропроцесорах. Вони можуть використовуватися як подільники частоти в керованих генераторах частоти (синтезаторах). При використанні в ланцюзі фазового автопідстроювання лічильники можуть бути використані для множення частоти як в синтезаторах, так і в мікропроцесорах.

Основні параметри лічильника: - M (модуль рахунку) - число стійких станів; - E (ємність) - максимальне число, яке може бути записано в лічильник (E = M - 1);

Найпростіший асинхронний двійковий лічильник

Найпростіший асинхронний лічильник являє собою кілька послідовно включених рахункових тригерів (рис. 4.1). Нагадаємо, що за кожним вхідного імпульсу рахунковий тригер змінює свій стан на протилежне.

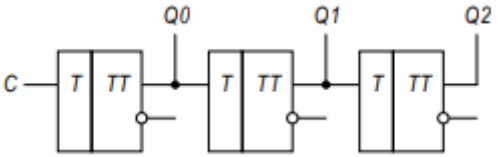


Рис. 8.1 – Найпростіший сумуючий асинхронний лічильник

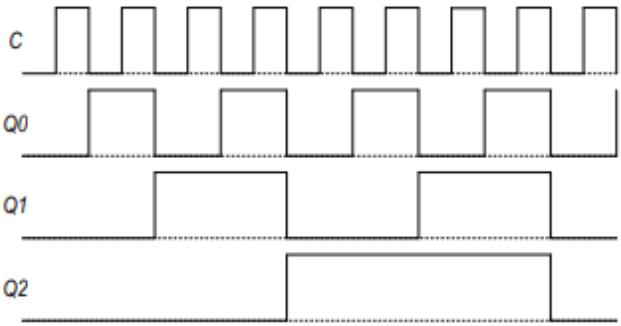


Рисунок 8.2 – Часова діаграма роботи підсумовуючого асинхронного лічильника

Для того щоб розібратися, як працює схема двійкового лічильника, скористаємося часовими діаграмами сигналів на вході і виходах цієї схеми, наведеними на рис. 8.2.

Нехай первісний стан всіх тригерів лічильника буде нульовим. Цей стан видно на часових діаграмах. Після надходження на вхід лічильника тактового імпульсу (який сприймається по задньому фронту) перший тригер змінює свій стан на протилежний, тобто одиницю. Так як по приходу першого імпульсу змінився стан першого тригера, то цей тригер містить молодший розряд двійкового числа (одиниці).

Подамо на вхід лічильника ще один тактовий імпульс. Значення першого тригера знову зміниться на прямо протилежне. На цей раз на виході першого тригера, а значить і на вході другого тригера, сформується задній фронт. Це означає, що другий тригер теж змінить свій стан на протилежний. це чітко видно на часових діаграмах, наведених на рис. 8.2.

Продовжуючи аналізувати часову діаграму, можна визначити, що на виходах наведеної схеми лічильника послідовно з'являються цифри від 0 до 7. Ці цифри записані в двійковому вигляді. При надходженні на лічильний вхід лічильника чергового імпульсу, вміст його тригерів збільшується на 1. Тому такі лічильники отримали

назву підсумовуючих лічильників. Якщо інформацію знімати з інверсних виходів тригерів, то вийде віднімаючий лічильник.

Лічильник з довільним модулем рахунку

Для побудови такого лічильника можна використовувати двійковий лічильник, у якого модуль рахунку М повинен бути більше модуля рахунку лічильника, що розробляється з довільним модулем рахунку.

Нехай потрібно зробити лічильник з М = 10. У чотирьохрозрядний лічильника модуль рахунку дорівнює 16 (більше 10). Схема лічильника являє собою 4 послідовно включених рахункових тригера, у яких є вхід скидання R (див. рис. 4.3). Число 10 в двійковій системі числення представляється 1010. Коли на виходах лічильника буде код 1010, на виході елемента «І» з'явиться логічна одиниця, яка запустить схему вимкнення. Тривалість імпульсу на виході схеми вимкнення повинна бути достатня для надійного скидання всіх тригерів лічильника в 0. Розряди числа 1010, що дорівнюють 1, подаються на схему «І» з прямих виходів тригерів, а рівні 0 – з інверсних.

Таким чином, як тільки лічильник дорахував до 10, відбудеться обнулення всіх тригерів, і рахунок продовжиться з коду 0000. Як схеми вимкнення може бути застосований RS-тригер. Сигнал на вході R лічильника буде діяти протягом одного періоду вхідних імпульсів.

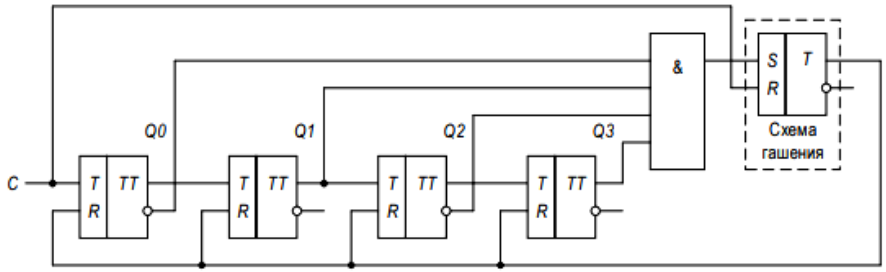


Рисунок 8.3 – Лічильник з модулем рахунку М = 10

Умовно-графічне позначення підсумовуючого довічного лічильника на принципових схемах наведено на рисунку 4.4. У лічильниках зазвичай передбачають вхід обнулення мікросхеми R, який дозволяє записати в усі тригери лічильника нульове значення. Цей стан іноді називають вихідним станом лічильника.

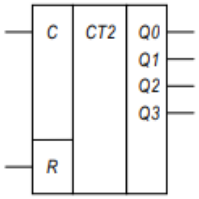


Рисунок 8.4 – Умовне графічне позначення чотирьохрозрядного двійкового лічильника

Кільцевий лічильник

У розглянутих схемах лічильників швидкодію всієї схеми визначається часом поширення сигналу від входу до виходу найстаршого розряду. При цьому виходить, що чим більше необхідний коефіцієнт ділення, тим більше двійкових розрядів лічильника потрібно для реалізації цього дільника. Тим більший час потрібно для поширення сигналу від входу синхронізації лічильника, до його виходу, і тим менше буде гранична частота, яку можна подавати на вхід цього дільника.

Щоб обійти цю неприємну особливість, потрібно, щоб лічильник готував свій новий стан в проміжках між тактовими імпульсами і тільки записував його по приходу нового імпульсу.

Проста схема, яка реалізує даний принцип, – це схема кільцевого лічильника. Такий лічильник можна побудувати на основі зсувного регістру. Схема кільцевого лічильника і часові діаграми його роботи наведені на рис. 8.5.

Як видно, при рахунку від першого розряду до останнього поширюється «хвиля» одиниць, а потім «хвиля» нулів. Для усунення

цього ефекту вводиться додаткове комбінаційний пристрій (КУ) для формування правильних чисел (Рис. 8.6).

Як перевага схеми кільцевого лічильника можна відзначити те, що її швидкодія залежить тільки від часу затримки одного тригера. Це означає, що на кільцевих лічильниках можна реалізовувати найшвидкодіючі подільники частоти.

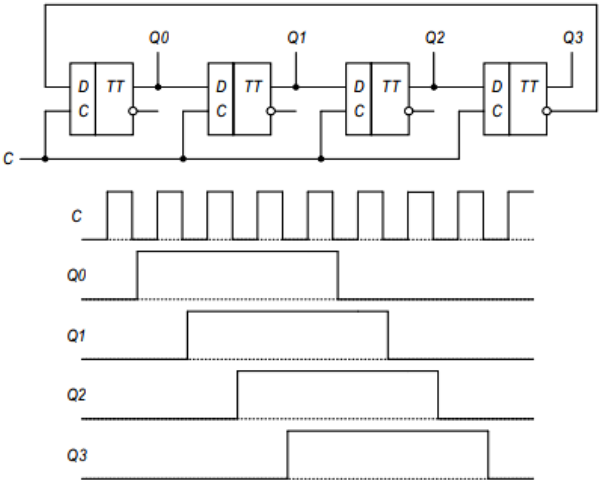


Рис. 8.5 – Схема і часові діаграми кільцевого лічильника

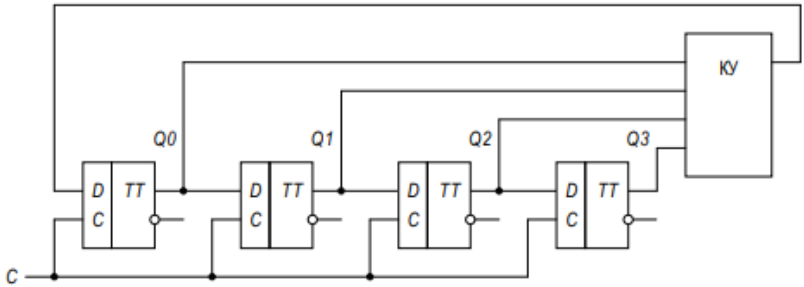


Рис. 8.6 – Схема двійкового лічильника на основі кільцевого лічильника

8.4. Порядок виконання роботи

Запустити середовище Altera Quartus II і створити новий проект з параметрами:

– Ім'я проекту: LabWork_4_StudentName

– Сімейство ПЛІС: Cyclone IV E

– Тип ПЛІС: EP4CE115F29C7

Решту залишити за замовчуванням.

8.4.1. Дослідження роботи реверсивного лічильника в режимах підсумовування, віднімання і паралельної завантаження

1. Вивчити опис і режими роботи реверсивного лічильника SN74193 (аналог КР1533ІЕ7).

2. Додати в проект новий файл схеми і накреслити схему лічильника на основі бібліотечного елемента «74193» (даний модуль є моделлю реальної мікросхеми SN74193 / КР1533ІЕ7).

3. Виконати аналіз і синтез проекту і зробити призначення виводів відповідно до таблиці 8.1. Після цього виконати повну компіляцію проекту.

Таблиця 8.1

Призначення виводів лічильника

Сигнал	**G_50MHz**	**CU**	**CD**	**D3**	**D2**	**D1**	**D0**
Вивід ПЛІС	Y2	R24	N21	AD27	AC27	AC28	AB28
На стенді	50 MHz Oscillator	KEY3	KEY2	SW3	SW2	SW1	SW0

Сигнал	**LOAD**	**RESET**	**Q3**	**Q2**	**Q1**	**Q0**
Вивід ПЛІС	M21	M23	G21	G22	G20	H21
На стенді	KEY1	KEY0	LEDG7	LEDG6	LEDG5	LEDG4

4. Запрограмувати навчальний стенд і, за часовою діаграмою проаналізувати роботу лічильника в трьох режимах – підсумовування, віднімання, паралельна завантаження. В останньому режимі на паралельний вхід лічильника необхідно завантажити число відповідно до свого варіантом (див. табл. 8.2).

8.4.2. Дослідження лічильника з довільним модулем рахунку

1. Змінити вихідну схему таким чином, щоб лічильник рахував до числа, визначеного варіантом (див. табл. 8.2).

2. Виконати повну компіляцію проекту.

3. Переконатися в правильній роботі схеми за осцилограмою.

8.5. Зміст звіту

1. Мета роботи. 2. Схема дослідження лічильника.

3. Часові діаграми вхідних і вихідних імпульсів в режимах підсумовування і віднімання.

4. Схема дослідження лічильника з довільним модулем рахунку. 5. Часові діаграми вхідних і вихідних імпульсів лічильника з довільним модулем рахунку.

6. Висновки по роботі.

8.6. Варіанти завдань

Таблиця 8.2

Варіанти завдань

Варіант	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Число	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

8.7. Контрольні питання

1. Пояснити принцип роботи підсумовуючого лічильника.

2. Зобразити часові діаграми роботи підсумовуючого лічильника.

3. Пояснити принцип роботи віднімаючого лічильника.

4. Зобразити часові діаграми роботи віднімаючого лічильника.

5. Пояснити принцип роботи лічильника з довільним модулем рахунку.

6. У чому полягає перевага кільцевого лічильника перед асинхронними?

7. Назвіть основні параметри двійкових лічильників.

8. Де застосовуються лічильники?

Лабораторна робота №9
Верифікація HDL-проекту

9.1. Мета роботи

Метою роботи є вивчення можливостей верифікації HDL-проектів в середовищі ModelSim.

9.2. Короткі теоретичні відомості

Верифікація – в загальному випадку, це перевірка збігу результатів роботи розроблювального пристрою і вимог, які до нього пред'являються.

HDL-мови мають в своєму складі необхідний інструментарно для проведення верифікації та налагодження проектів ПЛІС. В якості середовища симуляції і налагодження пропонується використовувати программуModelSim, що входить до складу САПР AlteraQuartus II.

9.2.1. Паралельний оператор PROCESS в VHDL

Оператор *process* відноситься до класу операторів паралельної обробки, що дає можливість використовувати його для проектування цифрових пристроїв комбінаторного типу. У той же час він є батьківським оператором для всіх інших виконуваних операторів, які дозволено розташовувати в його тілі. Говорячи іншими словами, всі виконувані оператори, що розташовуються в тілі оператора *process*, виконуються по послідовному принципу. Оператор *process* може використовуватися як в явному вигляді, так і в неявному.

Явно заданий оператор *process* – це основна конструкція для поведінкової форми опису проектів. Розрізняють два підкласу явно заданого оператора ***process*** : зі списком чутливості і без списку чутливості.

Список чутливості (*sensitivitylist*) – це набір сигналів, зміна значення яких викликає негайний запуск на виконання оператора *process*. Якщо список чутливості не визначений, всередині

такого процесу повинен міститися оператор *wait*, щоб надати розробнику можливість управляти запуском і зупинкою відповідного оператора *process* . Оператора без списку чутливості завжди запускається на виконання автоматично в момент початку процесу моделювання роботи проекту.

Синтаксис явно заданого оператора *process* має наступний вид (в квадратних дужках вказані необов'язкові конструктивні елементи):

[мітка_процесу:] **process** [(список_чутливості)] [**is**]
[оператори_об'явлення_процесу] *– розділ оголошень*
Begin*– розділ виконуваних операторів*
– Тут розміщуються оператори наступних типів:
– установки значень сигналів;
– присвоєння значень змінним;
– виклики процедур;
– оператори ***case, exit, if, loop, next, null*** *або* ***wait*** *;*
endprocess [мітка_процесу];

Необхідно відзначити, що присвоєння значення змінної в тілі оператора *process* позначається символами «: =», а встановлення значення сигналу символами «<=».

9.2.2. Послідовні оператори VHDL

Оператор *if* відноситься до сімейства послідовних операторів, що розташовуються в розділі виконуваних операторів явно заданого процесу. Оператор *if* відповідальний за виконання того чи іншого блоку послідовних операторів в залежності від істинності одного або декількох умов (condition). Синтаксис оператора *if* має такий вигляд:

if умова_1 **then**
блок_операторів_1;
[**elsif** умова_2 **then**
блок_операторів_2;]

...

[**else**

блок_операторів_3;]

endif;

Оператор *wait* призупиняє виконання тіла оператора process до тих пір, поки не відбудеться ще одна обставина.

Мова VHDL підтримує кілька форм оператора *wait* , а саме:

waituntil умова; - *чекати виконання умови*

waitfor час **;** - *чекати заданий час, наприклад,* ***25ns***

waiton сигнал; - *чекати змінення значення сигналу*

wait; - *нескінченне очікування*

9.2.3. Testbench

Testbench – це віртуальний «стенд тестування» цифрової схеми. Створюється середовище, всередині якого поміщається тестованасхема як компонент. Тобто файл Testbench – це такий же HDL-файл, компонентом якого є тестована схема, описана на HDL-мові.

Усередині цього середовища описується генерація вхідних сигналів для схеми, які потім подаються на вхід для подальшого аналізу виходів тестованої схеми. При цьому сам Testbench не має зовнішнього інтерфейсу.

Потім цей файл симулюється в програмі симуляторі, яка буде виконувати дії, описані в цьому файлі і представляти результат в графічному вигляді у вигляді діаграм зміни сигналів.

Змінним параметром буде час – симулятор послідовно буде збільшувати час з необхідним дозволом і послідовно обчислювати значення сигналів.

Приклад Testbench-файлу, описаного на VHDL:

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

entity temp_vhd_tst is
end temp_vhd_tst;

architecture temp_arch of temp_vhd_tst is
        signal x1: std_logic;
signal x2: std_logic;
signal x3: std_logic;
        signal y: std_logic;
- оголошення, що тестується схема
component temp
port (
x1: in std_logic;
x2: in std_logic;
                x3: in std_logic;
                y: out std_logic
        );
end component;
begin
- вставка тестованої схеми
        i1: temp
        - зв'язок між портами і сигналами
port map (
x1 => x1,
x2 => x2,
x3 => x3,
                y => y
        );

        init: process
        begin
                - одноразово виконується код
                wait ;
        end process init;

        always: process
        - список чутливості
```

begin

- код, що виконується після зміни будь-якого з сигналів зі списку чутливості

end process always;

end temp_arch;

9.2.4 Оператори перевірки умов і виведення повідомленьв консоль

Мова VHDL допускає вбудовування в файл Testbench перевіркувиконання певних умов. Це робиться за допомогою***оператора assert*** . Приклад його використання показаний нижче:

assert SUM_EXPECTED = SUM_OUT - *умова для перевірки*

report "ERROR: output SUM isincorrect" **severity** warning;

Після ключового слова *assert* вказується умова, якапередбачається істинним (при нормальній роботі модуля воно повинна виконуватися). Якщо умова не виконується, в консоль виводиться повідомлення, вказане після ключового слова ***report***. Після *severity*вказується рівень важливості перевіряють умови.

Рівень важливості може бути наступним:

• *note* - «примітка», це повідомлення не є помилкою і носить інформаційний характер;

• *warning* - «попередження», такі повідомлення вираховуються, але їх наявність не є підставою для визнання результатів некоректними;

• *error* - «помилка», результат моделювання визнається некоректним, моделювання триває;

• *failure* - «збій», аналогічно до помилки, але виконання негайно переривається.

За замовчуванням повідомлення мають рівень *error* .

Призначення рівня важливості проводиться самим розробником тесту. Передбачається, що повідомлення рівня «збій» будуть привласнювати помилки, які не можуть бути досить легко виправлені, або відносяться до «ситуацій, які ніколи не можуть виникнути ». Виникнення повідомлень такого рівня має означати, що в процесі моделювання виникли істотні проблеми, і виробляти подальший аналіз не має сенсу.

9.3. Завдання до роботи

9.3.1. Симуляція проекту вручну

1. Запустити середовище AlteraQuartusII і створити проект схеми у відповідності зі своїм варіантом (див. табл. 5.1). Слід мати на увазі, що ModelSim не підтримує симуляцію файлів схем, тому краще описати схему на одному з HDL-мов.

Примітка. Для симуляції схеми потрібно перетворити її в HDL-опис (File → Create / Update → Create HDL FilefromCurrentFile). У вікні, вибрати будь-яку зручну мову. Відкрити отриманий файл і додати його в проект (Project → AddCurrentFileto Project). Призначити його файлом верхнього рівня (Project → SetasTop-LevelEntity), а файл зі схемою (розширення * .bdf) видалити з проекту (Project → Add / RemoveFilesin Project).

2. Виконати аналіз і синтез проекту, призначити висновки і повну компіляцію проекту. У разі успішної компіляції, запустити симуляцію проекту (Tools → RunSimulationTool → RTL Simulation). При цьому слід вибрати мову, що відповідає файлу верхнього рівня проекту. AlteraQuartus II запустить програму ModelSim.

3. У вікні ModelSim запустити симуляцію свого проекту (Simulate → StartSimulation) і вибрати файл верхнього рівня проекту в каталозі «work». На панелі Objects з'являться порти і сигнали, наявні в проекті. Виділити вхідні та вихідні порти і перетягнути їх, захопивши мишею, в панель Wave.

4. Поставити сигнали на вхідні порти таким чином, щоб вони утворювали інкрементування у двійковий код (Wave → Clock ...), параметр FirstEdge задати «Falling». Встановити загальний час симуляції, достатню для симуляції всіх можливих станів входів, і запустити симуляцію (Simulate → Run → Run 100).

5. Програма проведе симуляцію заданого часу роботи схеми і відобразить результат у вигляді часової діаграми на панелі Wave. Перевірити відповідність діаграми таблиці істинності заданої функції.

6. Завершити роботу програми ModelSim.

9.3.2. Testbench

1. У AlteraQuartus II виконати генерацію шаблону файлу Testbench (Processing → Start → StartTestBenchTemplateWriter). На панелі Messages знайти рядок з інформацією про розташування згенерованого файлу (див. рис. 9.1).

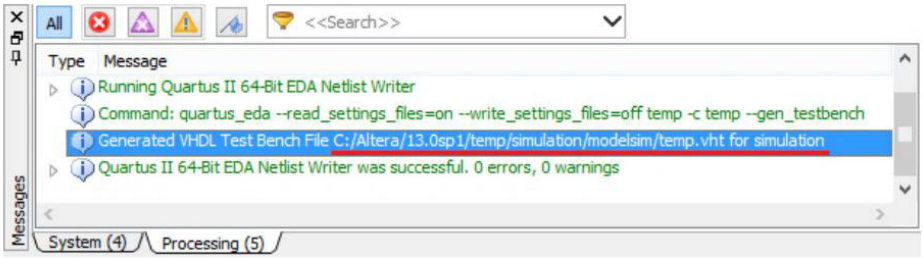


Рис. 9.1 – Приклад повідомлення Quartus II про розташування файлу Testbench

2. Виконати прив'язку файлу Testbench до проекту. Для цього відкрити налаштування проекту (Assignments → Settings), розділ Simula- tion і додати новий Testbench, виконуючи вказівки рис. 9.2.

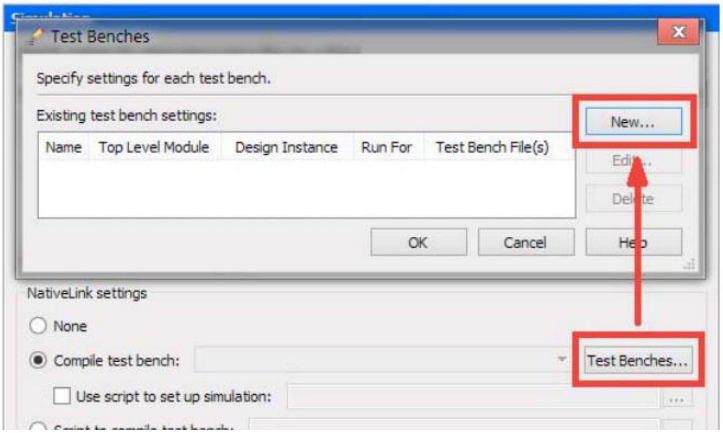


Рис. 9.2 – Налаштування симуляції, додавання Testbench

У вікні налаштувань (рис. 9.3) ввести:

• Testbenchname: довільне ім'я;

• Toplevelmoduleintestbench: ім'я файлу Testbench верхнього рівня (вказано в початковому тексті файлу Testbench як ім'я об'єкта *entity*);

• Simulationperiod: період симуляції відповідно до п. 9.3.1;

• Testbenchandsimulationfiles: вибрати Testbench-файл і нажати кнопку Add.

3. Відкрити Testbench-файл в середовищі AlteraQuartus II (File →Open) і ввести код, що описує оточення для тестування проекту: генерація вхідних сигналів і висновок консольних повідомлень про початок і завершення тесту.

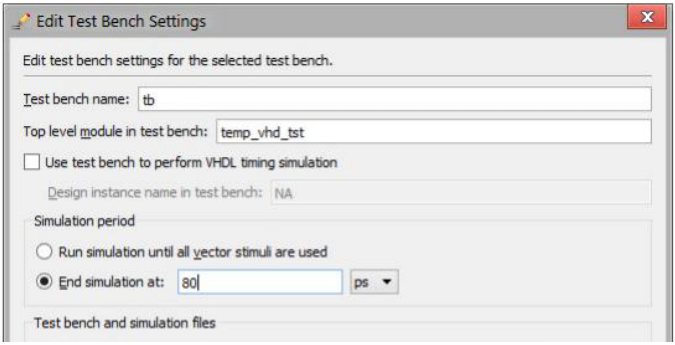


Рис. 9.3 – Налаштування Testbench

4. Виконати компіляцію проекту. У разі успішної компіляції запустити симуляцію проекту (Tools → RunSimulationTool → RTL Simulation).

5. Результат симуляції відобразиться у вигляді діаграми на панеліWave. Текстові повідомлення виводяться в панелі Transcript. Впевнитись, що виводиться повідомлення про закінчення тестування.

6. Закінчити роботу програми ModelSim.

9.4. Зміст звіту

1. Мета роботи.

2. HDL-опис схеми і її таблиця істинності за спрощеним варіантом.

3. Часова діаграма симуляції.

4. Вихідний код файлу Testbench.

5. Висновки по роботі.

9.5. Варіанти завдань

Таблиця 9.1

Варіанти завдань

Варіант	**1**	**2**	**3**	**4**	**5**
Функція	$A(B+C)$	$A+BC$	$\overline{ABC}$	$\overline{A}+\overline{BC}$	$A\oplus B\oplus C$
Варіант	**6**	**7**	**8**	**9**	**10**
Функція	$\overline{A\oplus B\oplus C}$	ABC	$A+B+C$	$\overline{AB+C}$	$\overline{A+B+C}$
Варіант	**11**	**12**	**13**	**14**	**15**
Функція	$A\oplus BC$	$\overline{A+B\oplus C}$	$\overline{A}B+C$	$\overline{A+B}+C$	$\overline{AB\otimes C}$

9.6. Контрольні питання

1. Що таке симуляція проекту? Для чого вона проводиться?

2. Поясніть принцип тестування проектів для ПЛІС із використанням файлів Testbench.

3. Опишіть структуру файлу Testbench. Чим він відрізняється від VHDL-файлу?

4. Що таке компонент в VHDL? Як він реалізується?

5. Поясніть роботу оператора *process*. Яке його призначення в мові VHDL?

6. Як працює оператор умови в мові VHDL? Наведіть приклад використання оператора *if*.

7. Наведіть приклад опису періодичного сигналу.

8. Як вивести повідомлення при тестуванні проекту?

Література

1. Цифрова схемотехніка. Навчальний посібник./ М.Г. Лорія, П.Й. Єлісєєв, О.Б. Целіщев. – Сєвєродонецьк: Вид-во Східноукр. нац. ун-ту імені Володимира Даля, 2016. – 280 с., 112 іл., 9 табл., 30 бібліогр. назв.
2. Мірошник М. А., Клименко Л. А., Корольова Я. Ю. Технології та автоматизація проєктування цифрових пристроїв складних комп'ютерних систем на ПЛІС: Навч. посібник. – Харків: УкрДУЗТ, 2021. – 220 с.
3. Основи та методи цифрової обробки сигналів: від теорії до практики: навч. посібник / Ушенко Ю.О., М.С. Гавриляк, М.В. Талах, В.В. Дворжак. – Чернівці : Чернівецький нац. ун-т ім. Ю. Федьковича. – 2021. – 308 с.
4. Лахно В.А., Гусєв Б.С., Смолій В.В., Місюра М.Д., Касаткін Д.Ю. Технології проектування комп'ютерних систем (частина 1) - К.: НУБіП України, 2019. – 205 с.
5. Alessio S.M. Digital Signal Processing and Spectral Analysis for Scientists, 1-st Edition - Switzerland: Springer Cham, 2016. – 924p.

ДОДАТОК А

ПРИДУШЕННЯ БРЯЗКІТУ КОНТАКТІВ

При використанні механічних комутаторів (кнопки, тумблери і інші ключі) завжди виникає таке явище,як брязкіт контактів.

Брязкіт контактів – явище, яке виникає в електричних і електронних перемикачах, при якому вони замість деякого стабільного перемикання виробляють випадкові багаторазові неконтрольовані замикання і розмикання контактів (відбувається в момент перемикання) (див. рис. В.1). Нижче наведено лістинг коду універсального модуля обробки сигналу з механічним перемикачем для придушення ефекту брязкоту контактів. Блок називається «debouncer». Модуль написаний мовою Verilog.

```
module button_debouncer
# (
        parameter CNT_WIDTH = 16
)
(
input clk_i,
        input rst_i,
input sw_i,
outputreg sw_state_o,
outputreg sw_down_o,
outputreg sw_up_o
);

reg [1: 0] sw_r;
always @ ( posedge rst_i or posedgeclk_i)
        if (rst_i)
                sw_r<= 2'b00;
        else
                sw_r<= {sw_r [0], ~ sw_i};

reg [CNT_WIDTH-1: 0] sw_count;

wire sw_change_f = (sw_state_o! = sw_r [1]);
wire sw_cnt_max = &sw_count;

always @ ( posedge rst_i orposedge clk_i)
        if (rst_i)
```

```
        begin
                sw_count<= 0;
                sw_state_o<= 0;
        end
        else if (sw_change_f)
        begin
                sw_count<= sw_count + 'd1;
                if (sw_cnt_max) sw_state_o<= ~ sw_state_o;
        end
        else  sw_count<= 0;

always @ ( posedge clk_i)
begin
        sw_down_o<= sw_change_f&sw_cnt_max& ~ sw_state_o;
        sw_up_o<= sw_change_f&sw_cnt_max&sw_state_o;
end

endmodule
```

Для введення сигналу безпосередньо з кнопки є вхідний порт *sw_i* . Вхід для системних тактових імпульсів, з якими синхронно працює вся інша схема - *clk_i* . Вхід для асинхронного скидання всієї схеми - *rst_i* . З метою універсальності передбачено два роздільних виходи для події «кнопка натиснута» і події «кнопка відпущена» - *sw_down_o* і *sw_up_o* . Вихід *sw_state_o* служить для відображення стану кнопки (тривалості натискання), природньо, після усунення ефекту брязкоту.

Приклад підключення схеми придушення брязкоту контактів кнопки показаний на рис. В 1. Сигнал з кнопки повинен бути пропущенийчерез модуль (подається на вхід *BUTTON*), замість того, щоб йти безпосередньо до досліджуваної схеми. Крім цього, на модуль повинен подаватися тактовий сигнал з генератора (вхід *G_50MHz*).

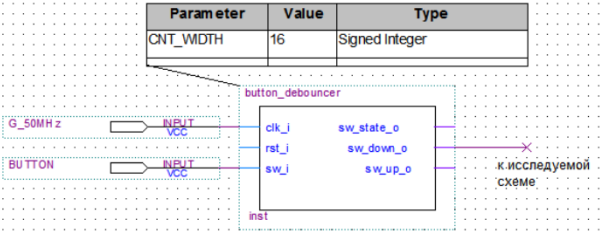


Рис. В.1 – Приклад підключення схеми придушення брязкоту контактів