

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт з дисципліни
“ПРОЕКТУВАННЯ МІКРОПРОЦЕСОРНИХ СИСТЕМ”

Частина 2: Проектування цифрових схем в пакеті ActiveHDL

для студентів спеціальності 123 «Комп’ютерна інженерія»
всіх форм навчання

Методичні вказівки до виконання лабораторних робіт з дисципліни «Проектування мікропроцесорних систем», частина 2: Проектування цифрових схем в пакеті ActiveHDL. Для студентів спеціальності 123 «Комп'ютерна інженерія» всіх форм навчання. /Укл.: І.Я. Зеленьова, С.С. Грушко. – Запоріжжя: ЗНТУ, 2018.– 50 с.

Укладачі: І.Я. Зеленьова, к.т.н., доцент

С.С. Грушко, асистент

Рецензент: В.В. Шкарупило, к.т.н., доцент

Відповідальний за випуск: С.С. Грушко

Затверджено на засіданні кафедри
комп'ютерних систем та мереж

Протокол № 9
від «17» _травня_ 2018 р.

Рекомендовано до видання
НМК факультету КНТ

Протокол № 10_
від «_28_» _травня_ 2018 р.

ЗМІСТ

1 Лабораторна робота №2	5
ПРОЕКТУВАННЯ ТА ДОСЛІДЖЕННЯ ЦИФРОВИХ КОМПАРАТОРІВ З ВИКОРИСТАННЯМ МОВИ VHDL	5
1.1 Проектна частина роботи.....	5
1.1.1 Теоретичні відомості.....	5
1.1.2 Приклад виконання проектної частини роботи	8
1.1.3 Варіанти завдань	10
1.2 Дослідницька частина роботи.....	11
1.2.1 Коротка теорія про середовище Active-HDL.....	11
1.2.2 Порядок виконання роботи	14
1.3 Зміст звіту з лабораторної роботи	23
1.4 Контрольні питання	23
2 Лабораторна робота №3	24
ПРОЕКТУВАННЯ ТА ДОСЛІДЖЕННЯ СХЕМ БАГАТОФУНКЦІОНАЛЬНИХ РЕГІСТРІВ ЗА ДОПОМОГОЮ САПР ACTIVE-HDL. СТРУКТУРНИЙ ПІДХІД.....	24
2.1 Проектна частина роботи.....	24
2.1.1 Теоретичні відомості з проектування регістрів.....	24
2.1.2 Приклад виконання проектної частини лабораторної роботи: синтез багатофункціонального регістра (БФР).....	29
2.1.3 Варіанти індивідуальних завдань	33
2.2 Дослідницька частина роботи. Моделювання та верифікація схеми БФР за допомогою САПР Active-HDL структурним способом	34
2.2.1 Короткі теоретичні відомості.....	34

2.2.2 Хід роботи. Приклад виконання дослідницької частини лабораторної роботи	36
2.3 Зміст звіту з лабораторної роботи.....	50
2.4 Контрольні питання	50

1 Лабораторна робота №2

ПРОЕКТУВАННЯ ТА ДОСЛІДЖЕННЯ ЦИФРОВИХ КОМПАРАТОРІВ З ВИКОРИСТАННЯМ МОВИ VHDL

Мета: вивчити принципи функціонування та проектування цифрових компараторів (схем порівняння), навчитися програмувати функціональні особливості схем в Active-HDL, проводити моделювання роботи мікросхем.

1.1 Проектна частина роботи

1.1.1 Теоретичні відомості

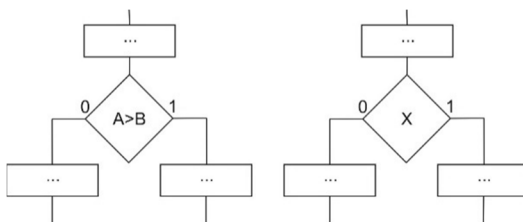
Цифрові компаратори (схеми порівняння) обчислюють значення результату відносини двох двійкових величин. Значення може бути true (1) або false (0).

Схеми порівняння – це комбінаційні схеми, які призначені для реалізації логічних умов в мікропрограмах, а також використовуються у всіх випадках, коли необхідно виконати порівняння двох чисел.

Важливо!!! Порівняння чисел виконується порозрядно, починаючи зі старших розрядів.

Результати порівняння можуть бути використані як інформативні (вхідні) сигнали для керуючих автоматів або як «прапорці» в командах передачі управління. Компаратори також виконують важливу функцію в схемах контролю і діагностики.

Наприклад:



Теоретичною основою для побудови функцій порівняння служить функція рівнозначності (рівність). Її інверсією, тобто

функцією нерівнозначності (нерівності) є функція XOR. З цього робимо висновок, що *якщо*:

$$\begin{aligned} A \neq B &\rightarrow \text{функція нерівнозначності (XOR)} \rightarrow \\ Q_i &= \bar{a}_i \cdot b_i \vee a_i \cdot \bar{b}_i = a_i \oplus b_i, \\ \text{то } A=B &\rightarrow \text{функція рівнозначності } (\overline{\text{XOR}}) \rightarrow \\ R_i &= a_i \cdot b_i \vee \bar{a}_i \cdot \bar{b}_i = \overline{a_i \oplus b_i}. \end{aligned}$$

Для порівняння чисел *необхідно і достатньо* виконати:

- $R = \bigwedge_{i=1}^n R_i$, для рівнозначності (рівності чисел) необхідно виконати порівняння за всіма розрядами і визначити значення множення результатів. Результат буде дорівнювати 1, якщо числа рівні.

- $Q = \bigvee_{i=1}^n Q_i$, для нерівнозначності (нерівності чисел), виконувати порівняння результатів порозрядного порівняння, до отримання хоча б одного позитивного результату.

(1) Наприклад, щоб визначити рівність двурозрядних чисел $A=B$, тобто $[a_1 \ a_2]=[b_1 \ b_2]$, необхідно і достатньо перевірити результат множень:

$$\begin{aligned} R &= R_1 \cdot R_2 \\ R_1 &= a_1 \cdot b_1 \vee \bar{a}_1 \cdot \bar{b}_1 \\ R_2 &= a_2 \cdot b_2 \vee \bar{a}_2 \cdot \bar{b}_2 \\ R &= (a_1 \cdot b_1 \vee \bar{a}_1 \cdot \bar{b}_1) \& (a_2 \cdot b_2 \vee \bar{a}_2 \cdot \bar{b}_2) \end{aligned}$$

(2) Наприклад, щоб визначити нерівність двурозрядних чисел $A \neq B$, тобто $[1 \ 1] \neq [1 \ 0]$, необхідно рахувати функцію нерівнозначності порозрядно, доки не буде отриманий позитивний результат:

$$\begin{aligned} Q_1 &= \bar{a}_1 \cdot b_1 \vee a_1 \cdot \bar{b}_1 = 0 \cdot 1 \vee 1 \cdot 0 = 0 \\ Q_2 &= \bar{a}_2 \cdot b_2 \vee a_2 \cdot \bar{b}_2 = 0 \cdot 0 \vee 1 \cdot 1 = 1 \end{aligned}$$

Порівняння чисел на $>$, $<$, $\geq$, $\leq$ (більше, менше, більше та дорівнює, менше та дорівнює). Таке порівняння можна виконати,

наприклад, в операційному автоматі, з використанням суматора і визначити знак різниці $(A-B)$, якщо знак «0», то $A \geq B$, якщо знак «1», то $A < B$. При цьому наступним кроком потрібно виконати уточнююче порівняння $A=B$.

Очевидно, що цей процес займає досить великий час. Простіше використовувати спеціальну комбінаційну схему порівняння. Визначимо функцію, на підставі якої будується схема, де, наприклад, $A > B$. Ті для цієї операції буде виглядати так:

a_i	b_i	y_i
0	0	0
0	1	0
1	0	1
1	1	0

y_i – результат в i -му біті.

В результаті отримаємо таку функцію: $y_i = a_i \cdot \bar{b}_i$

У разі, якщо $A < B$, використовуємо комбінаційну схему з такою ТІ відповідно:

a_i	b_i	y_i
0	0	0
0	1	0
1	0	1
1	1	0

В результаті отримаємо функцію: $y_i = b_i \cdot \bar{a}_i$

Продовження для $A > B$: далі необхідно скласти повну функцію «>» для всіх розрядів за формулою:

$$Y_{(A>B)} = y_i \vee (R_i \cdot y_{i-1}) \vee (R_i \cdot R_{i-1} \cdot y_{i-2}) \vee \dots \vee (R_i \cdot \dots \cdot R_1 \cdot y_0)$$

Ми знаємо, що $y_i = a_i \cdot \bar{b}_i$, а також, що $R_i = a_i \cdot b_i \vee \bar{a}_i \cdot \bar{b}_i$

Оскільки $Y = f(a_i, b_i)$, підставимо аргумент у формулу та отримаємо розгорнуту функцію:

$$Y_{(A>B)} = a_i \cdot \bar{b}_i \vee ((a_i \cdot b_i \vee \bar{a}_i \cdot \bar{b}_i) \cdot (a_{i-1} \cdot \bar{b}_{i-1})) \vee \dots$$

Очевидно, що загальна формула досить велика. Це справедливо для універсальних повнорозрядних формул порівняння двох змінних.

Схеми порівняння з «0» і «1» будуть простіше, тому що у виразі деякі терми скоротяться за рахунок множення на 0. У загальному випадку схема порівняння з константою теж буде простіше з тих же причин.

Нестрогі нерівності $A \geq B$, $A \leq B$ виконуються шляхом комбінації розглянутих прийомів:

- $A \geq B \rightarrow \overline{A < B}$ («більше або дорівнює» – це «не менше»). Функцією, на підставі якої будується схема, буде функція МЕНШЕ, від якої потім просто візьмемо інверсію;

- $A \leq B \rightarrow \overline{A > B}$ («менше або дорівнює» – це «не більше»). Функцією, на основі якої будується схема, буде функція БІЛЬШЕ, від якої потім теж просто візьмемо інверсію.

1.1.2 Приклад виконання проєктної частини роботи

Завдання: синтезувати компаратор (схему порівняння) чотирирозрядного двійкового числа А з константою В.

$$A \leq B, \text{ де } A = [a_3 \ a_2 \ a_1 \ a_0] \text{ та } B = [1 \ 0 \ 1 \ 1]$$

Рішення:

1) Перетворимо нестрогу нерівність у строгу:

$$A \leq B \rightarrow \overline{A > B} \text{ («менше або дорівнює» – це «не більше»)}$$

2) Визначимо базову функцію "більше" ("more" $\rightarrow$ М) для і-го розряду, склавши ТІ:

$$m_i = a_i \cdot \bar{b}_i$$

a_i	b_i	y_i
0	0	0
0	1	0
1	0	1
1	1	0

3) Загальну формулу для функції «не більше» (Not_M) отримуємо на основі базової функції «більше», а потім інвертуємо обидві частини рівності.

$$\text{Not_M} = \bar{M} = \overline{m_3 \vee R_3 \cdot m_2 \vee R_3 \cdot R_2 \cdot m_1 \vee R_3 \cdot R_2 \cdot R_1 \cdot m_0}$$

4) Виконаємо підстановку відомих значень числа В у кожен і-ий розряд, попередньо визначивши окремі (приватні) формули R_i і базові функції m_i

$$m_3 = a_3 \cdot \bar{b}_3 = a_3 \cdot 0 = 0$$

$$m_2 = a_2 \cdot \bar{b}_2 = a_2 \cdot 1 = a_2$$

$$m_1 = a_1 \cdot \bar{b}_1 = a_1 \cdot 0 = 0$$

$$m_0 = a_0 \cdot \bar{b}_0 = a_0 \cdot 0 = 0$$

$$R_3 = a_3 \cdot b_3 \vee \bar{a}_3 \cdot \bar{b}_3 = a_3 \cdot 1 \vee \bar{a}_3 \cdot 0 = a_3$$

$$R_2 = a_2 \cdot b_2 \vee \bar{a}_2 \cdot \bar{b}_2 = a_2 \cdot 0 \vee \bar{a}_2 \cdot 1 = \bar{a}_2$$

$$R_1 = a_1 \cdot b_1 \vee \bar{a}_1 \cdot \bar{b}_1 = a_1 \cdot 1 \vee \bar{a}_1 \cdot 0 = a_1$$

$$R_0 = a_0 \cdot b_0 \vee \bar{a}_0 \cdot \bar{b}_0 = a_0 \cdot 1 \vee \bar{a}_0 \cdot 0 = a_0$$

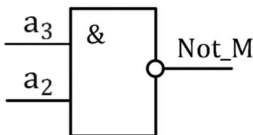
Зверніть увагу, що R_0 не бере участі в загальній формулі Not_M, але ми з його допомогою покажемо, як значення розрядів константи відображаються безпосередньо на результаті порівняння, тобто:

$$B = [1\ 0\ 1\ 1] \rightarrow R = [a_3\ \bar{a}_2\ a_1\ a_0]$$

При підстановці в загальну формулу вона набуде вигляду:

$$\begin{aligned} \text{Not_M} &= \bar{M} = \\ &= a_3 \cdot 0 \vee a_3 \cdot (a_2 \cdot 1) \vee a_3 \cdot \bar{a}_2 \cdot (a_1 \cdot 0) \vee a_3 \cdot \bar{a}_2 \cdot a_1 \cdot (a_0 \cdot 0) = \bar{a}_3 \cdot a_2 \end{aligned}$$

5) Синтез логічної схеми компаратора:



Важливо!! Якщо термів виходить більше після скорочення формули, ніж в прикладі, то і схема буде складатися з більшої кількості логічних елементів відповідно.

6) Тестова таблиця для перевірки коректності роботи схеми:

A	B	A ≤ B
a ₃ a ₂ a ₁ a ₀	b ₃ b ₂ b ₁ b ₀	Not M
0 1 1 1	1 0 1 1	1
1 0 1 0	1 0 1 1	1
1 0 1 1	1 0 1 1	1
1 1 0 0	1 0 1 1	0

Тестову таблицю потрібно приводити для загального випадку. У лабораторній роботі досить перевірити 4-5 тестових комбінацій для перевірки всіх можливих випадків (більше, менше, дорівнює).

7) Висновки: В даному простому прикладі очевидно, що нерівність не буде виконуватися тільки в тих випадках, коли обидва старших розряди числа A рівні 1.

1.1.3 Варіанти завдань

Виконати синтез схеми порівняння деякого двійкового числа з константою.

		Вид нерівності							
		A > B		A < B		A ≥ B		A ≤ B	
Варіанти	1	2	3	4	5	6	7	8	
	9	10	11	12	13	14	15	16	
	17	18	19	20	21	22	23	24	
	25	26	27	28	29	30	31	32	
		A=70	B=75	A=71	B=76	A=77	B=74	A=78	B=73
		Значення десяткової константи							

1.2 Дослідницька частина роботи

1.2.1 Коротка теорія про середовище Active-HDL

Active-HDL – середовище розробки, моделювання та верифікації проектів для програмованих логічних інтегральних схем, розроблена фірмою Aldec.

Програма дозволяє описувати пристрої за допомогою мов програмування апаратури, а також за допомогою структурних схем. Спочатку програма підтримувала тільки мову VHDL, але з часом додалася підтримка мов Verilog і SystemC. За допомогою програми можна графічно проектувати кінцеві автомати, а також конвертувати HDL опис в графічні структурні схеми і назад.

Компоненти Active-VHDL

Всі компоненти Active-VHDL об'єднані у єдину графічну середу, що є основним каркасом. Каркас забезпечує робочу область вікна і зв'язуючу середу для всіх елементів системи. Крім ядра моделювання, кожен Active-VHDL інструмент виконаний в окремому вікні. У таблиці представлено короткий опис компонентів Active-VHDL.

Таблиця 1.1 – Опис компонентів Active-VHDL

Компонент	Опис
Console	Вікно консолі – це інтерактивне вікно, призначене для введення-виведення тексту. Воно забезпечує введення команд Active-VHDL і виведення повідомлень, згенерованих Active-VHDL інструментальними засобами.
Design Browser	Вікно перегляду проекту показує поточне утримання проекту, включаючи: - файли ресурсів, приєднаних до проекту; - зміст файлів робочих бібліотек визначаються за замовчуванням; - структуру проєктованих пристроїв обраних для моделювання; - сигнали і змінні, оголошені всередині обраної області поточного проекту.
HDL Editor	HDL редактор – текстовий редактор, розроблений для створення вихідних файлів VHDL. Він показує певні

	категорії синтаксису в різних кольорах. Редактор глибоко інтегрований з моделюючим пристроєм, що дозволяє легко налагоджувати вихідний текст.
Language Assistant	Мовний помічник є допоміжним інструментом, який забезпечує ряд типових VHDL шаблонів і їх логічне перетворення в функціональні блоки. Він об'єднаний з HDL редактором так, щоб можна було автоматично вставити бажаний шаблон в відредагований вихідний файл.
State Machine Editor	Редактор автоматів з кінцевими станами – графічний інструмент, розроблений для редагування діаграм кінцевого автомата. Редактор здійснює автоматичний переклад графічних примітивів в коди VHDL.
Waveform Viewer	Переглядач тимчасових діаграм показує результати моделювання, під дією тестових сигналів. Це дозволяє графічно редагувати форму сигналу, для створення потрібних випробувальних векторів.
List	Вікно списку показує результати моделювання, виконані в зведеному в таблиці текстовому форматі.
Watch	Вікно засобів спостереження показує поточні значення вибраних сигналів і змінних протягом моделювання.
Processes	Вікно процесів показує поточний стан одночасних процесів в розробленому проекті протягом моделювання.
Library Manager	Бібліотечний менеджер – розроблений, щоб управляти VHDL бібліотеками та їх змістом.
Design Explorer	Провідник проекту полегшує управління Active-VHDL проектами. Він дозволяє не запам'ятовувати фізичну локалізацію файлів проекту.
Script Editor	Редактор сценарію – текстовий редактор з вмонтованим отладчиком. Він розроблений, щоб редагувати VBA Basic сценарії і виконувати команди Active-VHDL.

Структура файлів

При реалізації проекту ряд файлів створюється в каталозі проектів. Це файли конфігурації проекту, вихідні файли, файли форми сигналу, Active-VHDL проекти. Розширення файлів представлені в таблиці.

Таблиця 1.2 – Розширення файлів в проекті

*.VHD	Джерело VHDL код
*.VHQ	Джерело VHDL код, згенерований з кореневого файлу
*.ASF	Вихідний файл діаграми станів
*.DO	Файл сценарію
*.BAS	Вихідний командний файл
*.ADF	Файл опису проекту
*.WSP	Файл робочої області вікна
*.WFV	Файл форми сигналу
*.LOG	Файл звіту
*.ERF	Помилки і повідомлення компілятора
*.EPR	Список вихідних файлів для компіляції
*.SDF	Стандартний файл затримки
*.INI	Бібліотечний індексний файл
*.MFG	Бібліотечний головний файл
*.DAT	Файл даних
*.OID	Допоміжний файл Майстри HDL
*.TXT	Текстовий файл

Структура проекту

Кожен проект записаний в окремому каталозі, названому каталогом проектів. І каталог проектів, і проект має ту ж саму назву. Кожен каталог проектів містить підкаталоги SRC I GENERIC а також файли опису проектів, які мають те ж саме ім'я, як і назва проекту, тільки з розширенням .ADF.

Підкаталог SRC містить вихідні файли VHDL або файли діаграм автоматів з кінцевими станами з розширенням .VHD або .ASF.

Підкаталог GENERIC містить файл робочої області вікна з .WSP розширенням і робочої бібліотеки проектів з .VLB розширенням. Обидва мають ту ж саме назву, як і ім'я проекту. Підкаталог GENERIC також містить ряд робочих файлів, які використовуються в процесі синтезу і моделювання.

Конфігурація проекту і його стан описані в наступних файлах:

- * .ADF – файл опису проекту, що містить інформацію щодо структури проекту: вихідні файли і бібліотеки, які формують проект.

- * .WSP – файл робочої області вікна, що містить дані щодо поточного стану проекту. Він завжди відновлює всі параметри після закриття проекту і повторного його відкриття. Поточний статус містить інформацію про те, які файли були відкриті, і в якому місці тексту розташований курсор, які закладки були обрані, які вікна розкриті, інше.

1.2.2 Порядок виконання роботи

На підставі даних, отриманих в першій частині лабораторної роботи, необхідно синтезувати проект на мові VHDL, що працює за заданим алгоритмом (наприклад, який реалізує функцію Not_M, як в ПРИКЛАДІ п. 1.2.).

У програмі Active-HDL створюємо новий робочий простір (рис. 1.1).

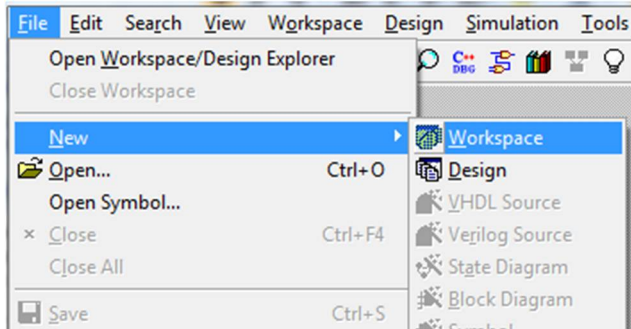


Рисунок 1.1 – Новий робочий простір в Active-HDL

В полі Type the workspace name вводимо ім'я робочого простору і натискаємо OK (рис. 1.2).

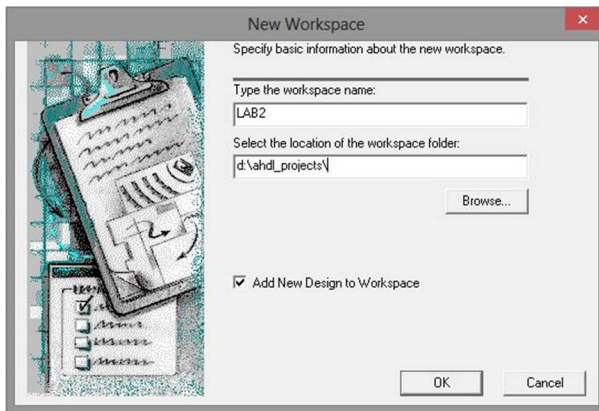


Рисунок 1.2 – Ім'я робочого простору в Active-HDL

У вікні New Design Wizard вибираємо Create an empty design і натискаємо Next (рис. 1.3).

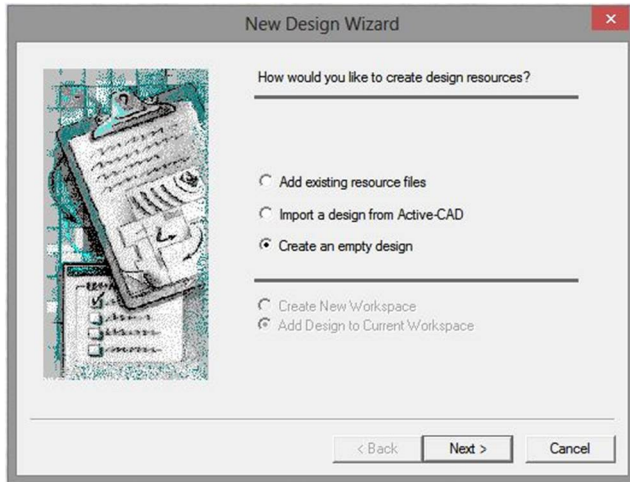


Рисунок 1.3 – Створення пустого проекту в Active-HDL

У списку Default HDL Language вибираємо мову VHDL і натискаємо Next (рис. 1.4).

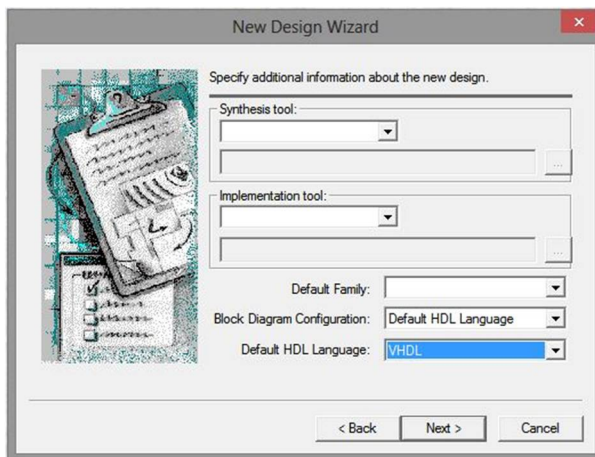


Рисунок 1.4 – Вибір мови

Відкриється вікно New Design Wizard (Майстер нового проекту) – інструмент, який допомагає в створенні нових проектів і додаванні

нових файлів в процесі проектування. В полі Type the design name необхідно ввести назву схеми (не повинно починатися з цифри) і натиснути Next (рис. 1.5).

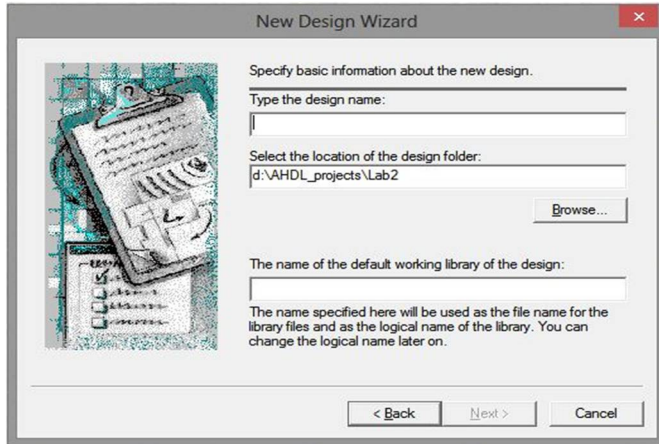


Рисунок 1.5 – Назва мікросхеми в Active-HDL

У вікні натискаємо Finish.

На панелі design flow вибираємо HDE (або на панелі Design Browser (зліва на екрані) вибираємо Add New File, і на вкладці Wizards вибираємо VHDL Source Code) (рис. 1.6).

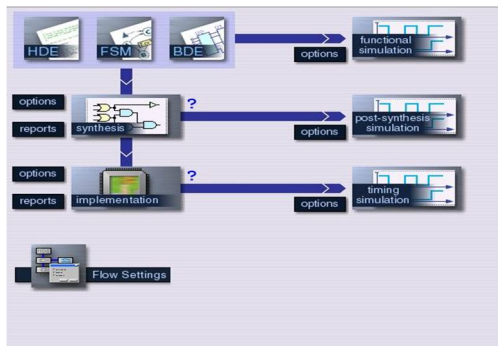


Рисунок 1.6 – Розширений маршрут проектування

Обираємо шаблон мови VHDL і натискаємо ОК (рис 1.7).

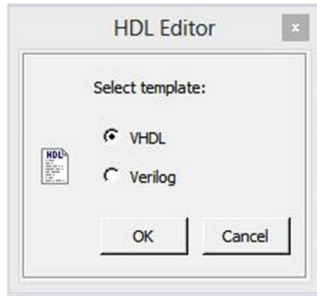


Рисунок 1.7 – Вибір мови проектування

У вікні New Source File Wizard залишаємо галочку на Add the generated file to the design і натискаємо Next (рис. 1.8).

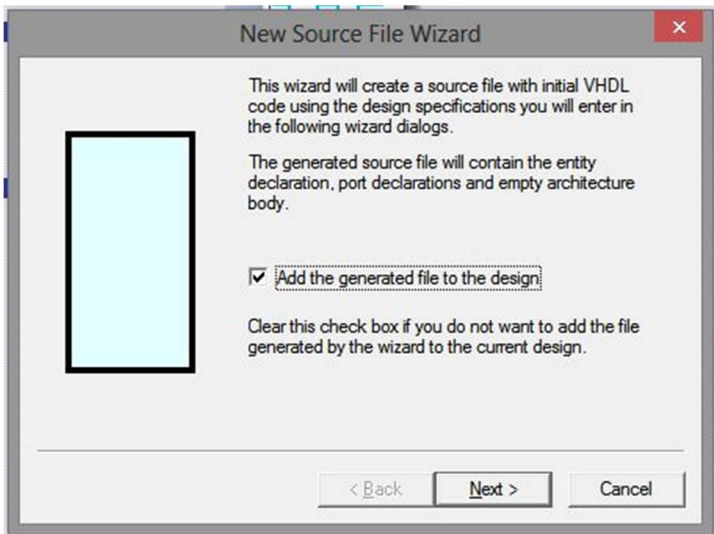


Рисунок 1.8 – Додавання файлу, що генерується, до проекту

Далі необхідно вказати назву файлу, що створюємо, і натиснути Next.

У вікні New Source File Wizard - Ports необхідно вказати входи і виходи мікросхеми. Для цього потрібно натиснути на кнопку New, ввести ім'я порту в поле Name і вказати в меню Port Direction тип порту (in-вхід, out-вихід, inout-двонаправлений порт).

Залежно від розрядності портів, що одночасно входять і виходять, в розділі Array Indexes вказується їх розрядність (відлік починається з 0).

У прикладі у нас порівнюються чотирирозрядні числа, тому вказуємо саме такі входні і вихідні порти мікросхеми, як на рис. 1.9. У лабораторній роботі ви вкажете порти виходячи з вашого завдання.

Натиснути кнопку Finish для створення скелетного файлу (рис. 1.9).

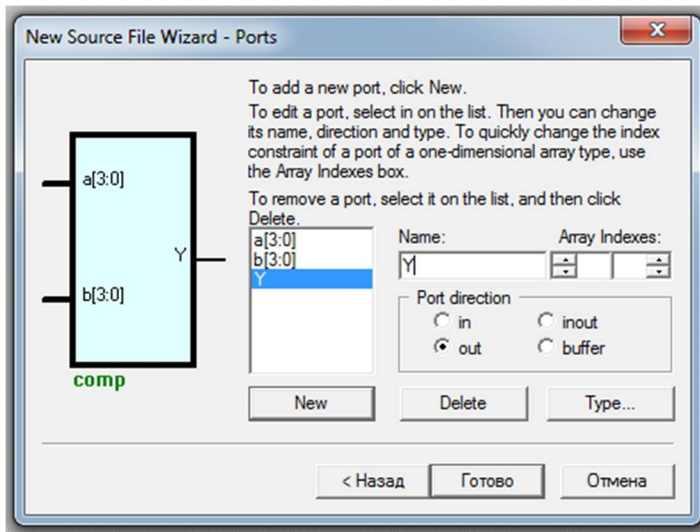


Рисунок 1.9 – Введення даних про інтерфейсні порти проекту (зверніть увагу, що а і b задаються у вигляді масивів, де зліва вказаний старший розряд)

В результаті створено файл-шаблон з розширенням *.vhd.

На основі проектної частини потрібно написати код роботи мікросхеми. У пункті 4 проектної частини ми отримали шляхом скорочень формулу для повної функції (в нашому прикладі – « $\leq$ »). Саме цю формулу потрібно представити за допомогою логічних операцій I, АБО, НІ і вписати в генерований код з розширенням .vhd у блок entity.

Важливо! Знак $\leq$ мовою VHDL виступає, як символ присвоєння.

Важливо! Назва блоку entity залежить від того, як названа мікросхема.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity comp is
port(
    a : in STD_LOGIC_VECTOR(3 downto 0);
    b : in STD_LOGIC_VECTOR(3 downto 0);
    Y : out STD_LOGIC
);
end comp;
architecture comp of comp is
begin
    Y<=not(a(2) and a(3));
end comp;
```

Компілюємо файл, натиснувши **Compile All**.

Для перевірки правильності роботи схеми необхідно створити випробувальний стенд – Test Bench. Для цього в панелі інструментів вибираємо Tools / Generate Test Bench.

У вікнах вибираємо Далі / Далі / Готово.

На панелі Design Browser у вкладці Files відкриваємо папку Test Bench, натискаємо правою кнопкою миші на файлі, з розширенням .do і вибираємо Execute (рис.1.10).

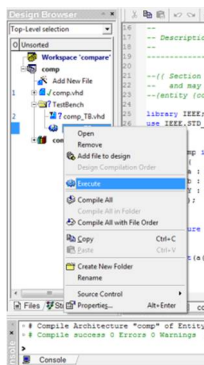


Рисунок 1.10 – Файл з розширенням .do на панелі Design Browser

У файлі часових діаграм, який з'явився, потрібно задати вхідні сигнали (рис. 1.11).

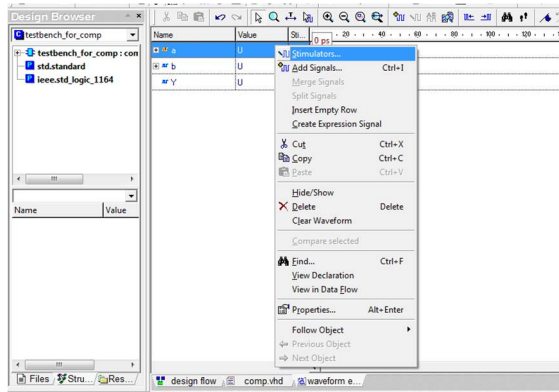


Рисунок 1.11 – Спосіб задання вхідних сигналів

Значення $a(3:0)$ задаємо за допомогою лічильника (рис 1.12).

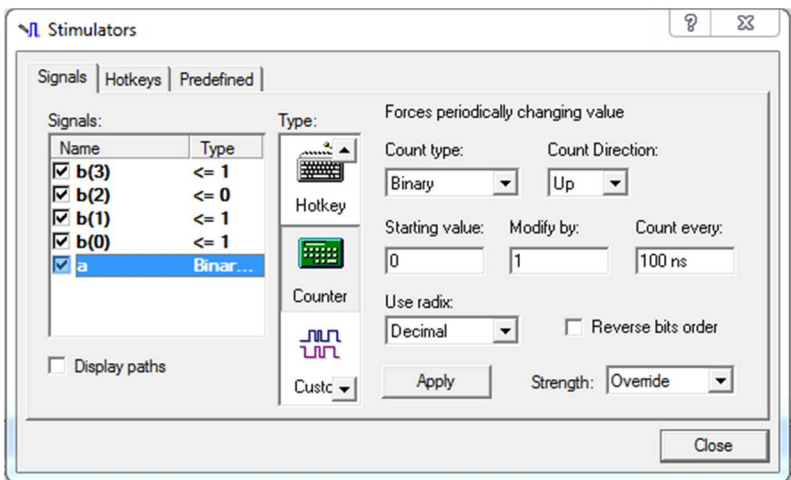


Рисунок 1.12 – Параметри задання значення лічильником

Значення $b(3:0)$ задаємо порозядно, відповідно до завдання (рис. 1.13).

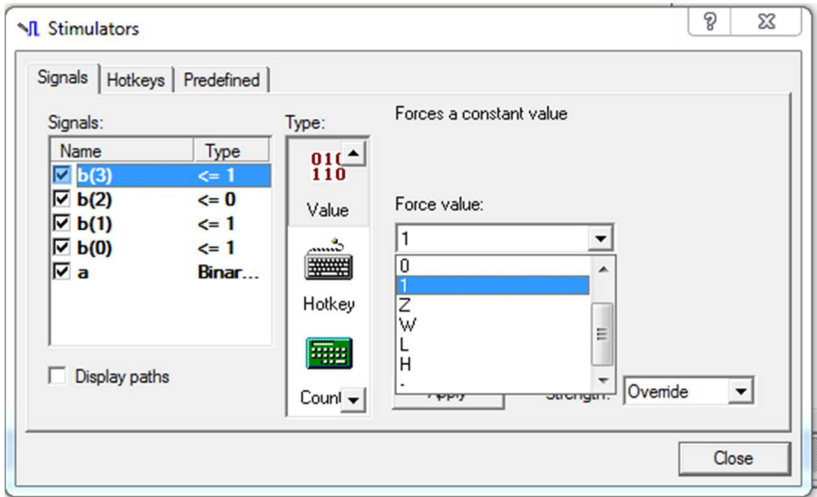


Рисунок 1.13 – Параметри порозрядного задання значення

Запускаємо процес симуляції Simulation / Run. Результати моделювання представлені на рис. 1.14.

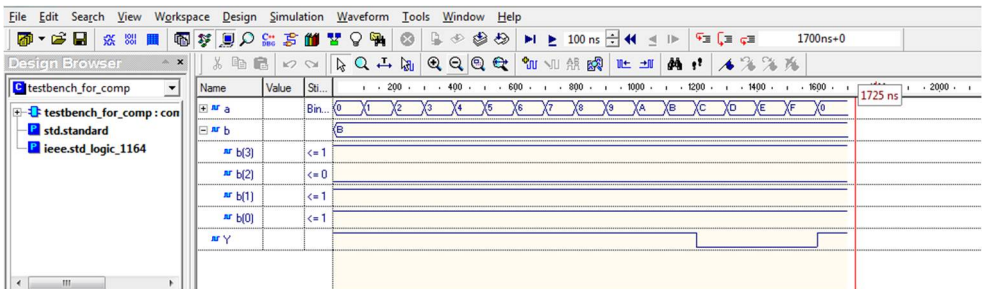


Рисунок 1.14 – Часова діаграма результатів моделювання

Проаналізував часову діаграму переконаємося, що задана в завданні схема компаратора працює правильно:

Число В – «В» в 16-річній системі.

Доки $A \leq B$, $Y = 1$

Як тільки A стає $> B$, Y переходить в 0.

1.3 Зміст звіту з лабораторної роботи

1.3.1 Привести докладний опис процесу синтезу логічної схеми компаратора, відповідно до заданого варіанту завдання (див. Приклад п.1.2).

1.3.2 Скласти текст програми з описом схеми заданого за спрощеним варіантом компаратора на VHDL.

1.3.3 Результати тестування отриманої схеми привести в звіті у вигляді часових діаграм. Зробити висновки по роботі (див. Приклад п.2.1).

1.4 Контрольні питання

1.4.1 Що таке Active-HDL?

1.4.2 Основні компоненти програмного середовища Active-HDL?

1.4.3 Що ви знаєте про мову VHDL?

1.4.4 У яких файлах описані конфігурація проекту і його стан?

1.4.5 Для чого використовується Waveform Viewer?

1.4.6 Що таке TestBench? Для чого він використовується?

2 Лабораторна робота №3

ПРОЕКТУВАННЯ ТА ДОСЛІДЖЕННЯ СХЕМ БАГАТОФУНКЦІОНАЛЬНИХ РЕГІСТРІВ ЗА ДОПОМОГОЮ САПР ACTIVE-HDL. СТРУКТУРНИЙ ПІДХІД

Мета: засвоїти основні правила проектування схем багатофункціональних регістрів та виконати верифікацію отриманої схеми способом моделювання за допомогою схемного редактора Active-HDL.

2.1 Проектна частина роботи

2.1.1 Теоретичні відомості з проектування регістрів

Регістри - найбільш широко використовувані пристрої в складі цифрових схем. Регістри оперують зв'язаними даними, що складають слово інформації.

Головним критерієм класифікації регістра є спосіб прийому і видачі даних. За цим критерієм регістри розділяються на:

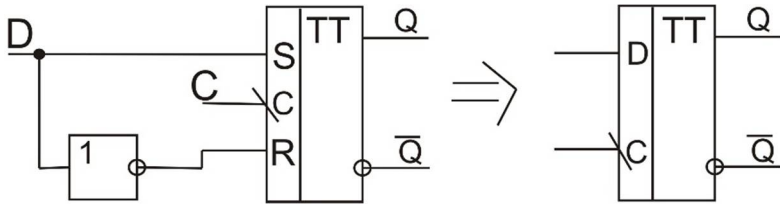
- паралельні (статичні);
- послідовні (зсувні);
- послідовно-паралельні.

Крім зазначених основних класів існують **багатофункціональні регістри (БФР)**, призначенням яких може бути виконання наступних функцій:

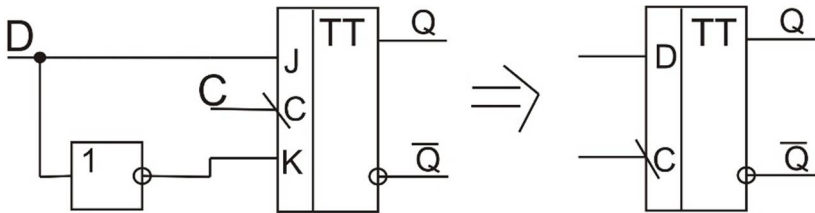
- прийом даних з одного або декількох портів;
- збереження даних;
- видача даних;
- праві і ліві зсуви;
- порозрядні логічні операції з даними.

Склад регістрів: тригери із загальною синхронізацією (зазвичай використовуються двоступінчаті тригери). Найбільш розповсюдженим випадком є синтез регістра на D - тригері (тригер даних). Якщо в якості базового заданий тригер іншого типу, то для спрощення задачі синтезу потрібно сформувати схему заміщення D - тригера на базі заданого і далі вирішувати задачу, як для D – тригера. Наприклад:

А) D - тригер на базі RS – тригера:



Б) D – тригер на базі JK – тригера:



Рівнобіжні (паралельні) регістри: прийом і видача інформації відбувається паралельно на усіх входах (виходах), при цьому синхросигнал подається той самий на всі синхровходи тригера (загальна синхронізація).

Послідовні регістри: прийом інформації відбувається по одному біту послідовно, у кожному такті. Крім того, для звільнення розряду, приймаючого інформацію, в кожному такті відбувається зсув в сусідній розряд.

Видача інформації може відбуватися послідовно або паралельно. Такий регістр називають **послідовно - паралельним**.

Синтез рівнобіжного статичного регістра.

Нехай потрібно синтезувати логічну схему простого - рівнобіжного статичного – регістру, який виконує тільки одну мікрооперацію (позначимо відповідний керуючий сигнал y) – прийом числа A (рис. 2.1). Візьмемо, наприклад, чотирьох-розрядний регістр.

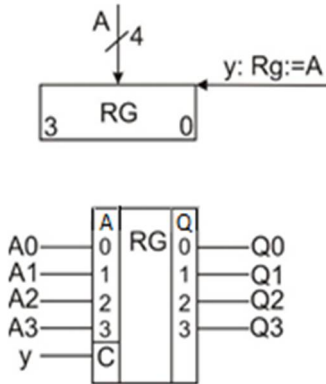


Рисунок 2.1 - Загальна структура регістра та його умовно-графічне позначення

Як базовий тригер, візьмемо D - тригер. Оскільки всі біти інформації в даному регістрі формуються однаково, то досить провести синтез схеми для i -го біта. Визначимо функції активації кожного тригера Q_i та функцію загальної синхронізації C_i :

$$Q_i = A_i \cdot y$$

$$C_i = y$$

Користуючись цими функціями можна виконати синтез логічної схеми регістра (рис. 2.2).

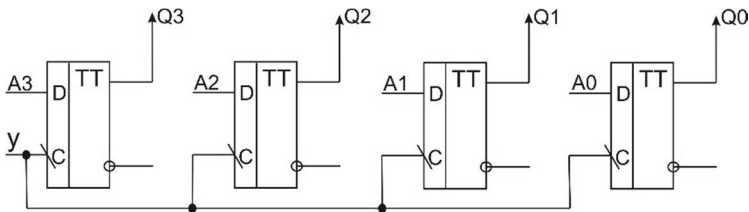


Рисунок 2.2 - Логічна схема регістра

Синтез регістра для прийому з декількох напрямків.

Покажемо, як виконати синтез регістра із двома портами прийому, наприклад, для чотирьох-розрядних чисел A та B (рис. 2.3).

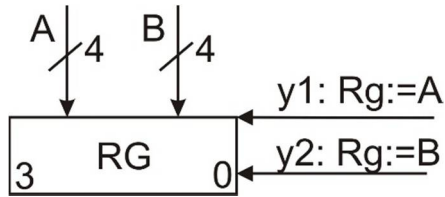


Рисунок 2.3 – Загальна структура регістру для прийому двох чисел

У складі регістра, крім тригера, буде **комбінаційна схема**, яка повинна забезпечувати запис числа А або числа В.

Для синтезу пристрою необхідно скласти **таблицю істинності регістра (таблицю мікрооперацій)**.

Мікрооперація - це спеціальний керуючий сигнал, що дозволяє виконання елементарної дії, що теж називається мікрооперацією (МО).

Таблиця 2.1 -Таблиця мікрооперацій регістра

МО	D ₃	C ₃	D ₂	C ₂	D ₁	C ₁	D ₀	C ₀
y ₁	A ₃	y ₁	A ₂	y ₁	A ₁	y ₁	A ₀	y ₁
y ₂	B ₃	y ₂	B ₂	y ₂	B ₁	y ₂	B ₀	y ₂

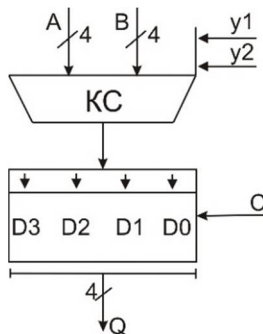


Рисунок 2.4 – Розгорнута структурна схема регістра для прийому двох чисел

Визначимо функцію активації тригера в кожному розряді (всі аналогічні), а також функцію синхронізації:

$$D_i = A_i \cdot y_1 \vee B_i \cdot y_2 \quad (\text{рис. 3.5})$$

$$C_i = y_1 \vee y_2 \quad (\text{рис. 3.6})$$

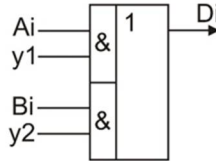


Рисунок 2.5 – Реалізація функції пам'яті синхронізації тригерів

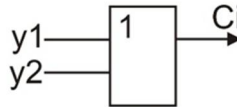


Рисунок 2.6 – Реалізація функції активації тригерів

Очевидно, що сигнали C_i і D_i сформуються за різний проміжок часу. Для забезпечення стійкості роботи схеми необхідно ввести в формулу сигнал глобальної синхронізації CLK :

$C_i = (y_1 \vee y_2) \wedge CLK$ тоді схема синхронізації виглядає, як на рис. 3.7. Зауважимо, що пауза сигналу CLK повинна бути більше часу формування сигналів D_i .

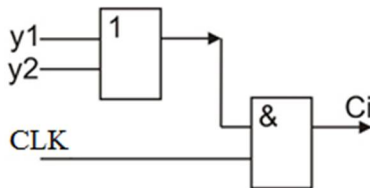


Рисунок 2.7 – Уточнена схема синхронізації регістра

Із урахуванням всіх необхідних зовнішніх сигналів схеми розробимо умовне графічне позначення отриманого регістра:

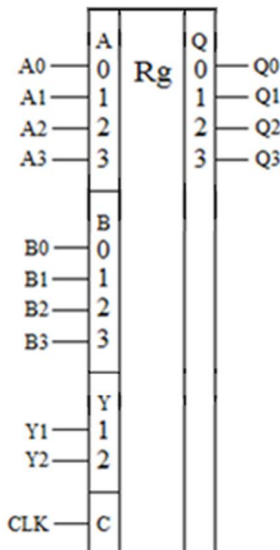


Рисунок 2.8 - Умовне графічне позначення регістра для прийому двох чисел

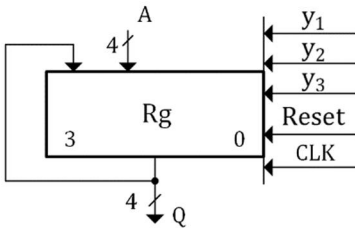
Таким чином, **методика синтезу** логічної схеми будь-якого нетривіального регістру включає етапи:

- аналіз постановки задачі та розробка структури регістра;
- формування таблиці мікрооперацій регістра;
- запис функцій активації тригерів та загальної синхронізації;
- синтез логічної схеми регістра та визначення його умовного графічного позначення.

Висновки. Кожний регістр, що є не тривіальним (тобто крім функції прийому, збереження і видачі одного числа має також інші функції), складається з комбінаційної схеми і набору тригерів, що реалізують пам'ять. Таким чином, будь-який багатофункціональний регістр являє собою найпростіший операційний автомат

2.1.2 Приклад виконання проектної частини лабораторної роботи: синтез багатофункціонального регістра (БФР)

Завдання: виконати проектування логічної схеми багатофункціонального регістра на основі даних за варіантом (табл. 2.2). Наприклад, вказаний нижче набір мікрооперацій (рис. 2.9):



Мікрооперації:

$$y_1 \# Rg := \bar{A}$$

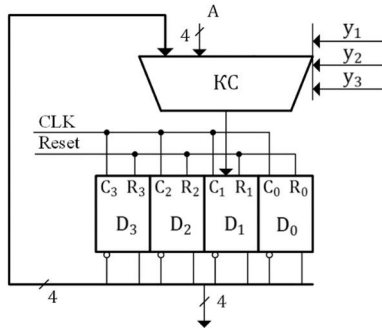
$$y_2 \# Rg := A \wedge Rg$$

$$y_3 \# Rg := LC_2 \wedge (Rg)$$

Рисунок 2.9 - Приклад набору мікрооперацій

Рішення:

1) Розробка розгорнутої структурної схеми БФР:



2) Таблиця мікрооперацій:

Мікро-операція	D ₃	C ₃	D ₂	C ₂	D ₁	C ₁	D ₀	C ₀
y ₁	$\bar{a}_3$	y ₁	$\bar{a}_2$	y ₁	$\bar{a}_1$	y ₁	$\bar{a}_0$	y ₁
y ₂	$Q_3 \wedge a_3$	y ₂	$Q_2 \wedge a_2$	y ₂	$Q_1 \wedge a_1$	y ₂	$Q_0 \wedge a_0$	y ₂
y ₃	Q ₁	y ₃	Q ₀	y ₃	Q ₃	y ₃	Q ₂	y ₃

3) Рівняння для синтезу комбінаційної схеми регістра:

а) Функції збудження тригерів:

$$D_3 = \bar{a}_3 \wedge y_1 \vee (Q_3 \wedge a_3) \wedge y_2 \vee Q_1 \wedge y_3$$

$$D_2 = \bar{a}_2 \wedge y_1 \vee (Q_2 \wedge a_2) \wedge y_2 \vee Q_0 \wedge y_3$$

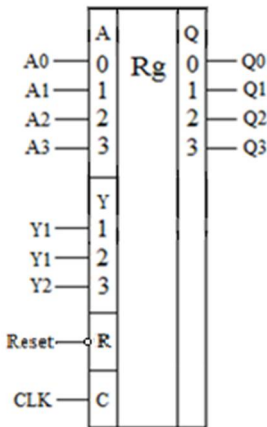
$$D_1 = \bar{a}_1 \wedge y_1 \vee (Q_1 \wedge a_1) \wedge y_2 \vee Q_3 \wedge y_3$$

$$D_0 = \bar{a}_0 \wedge y_1 \vee (Q_0 \wedge a_0) \wedge y_2 \vee Q_2 \wedge y_3$$

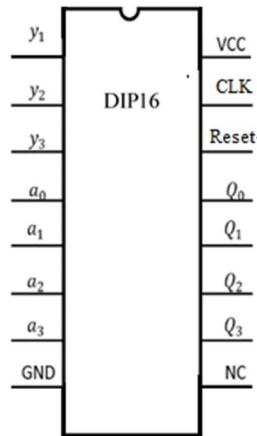
б) Рівняння для сигналу синхронізації:

$$C_i = \text{CLK} \wedge (y_1 \vee y_2 \vee y_3)$$

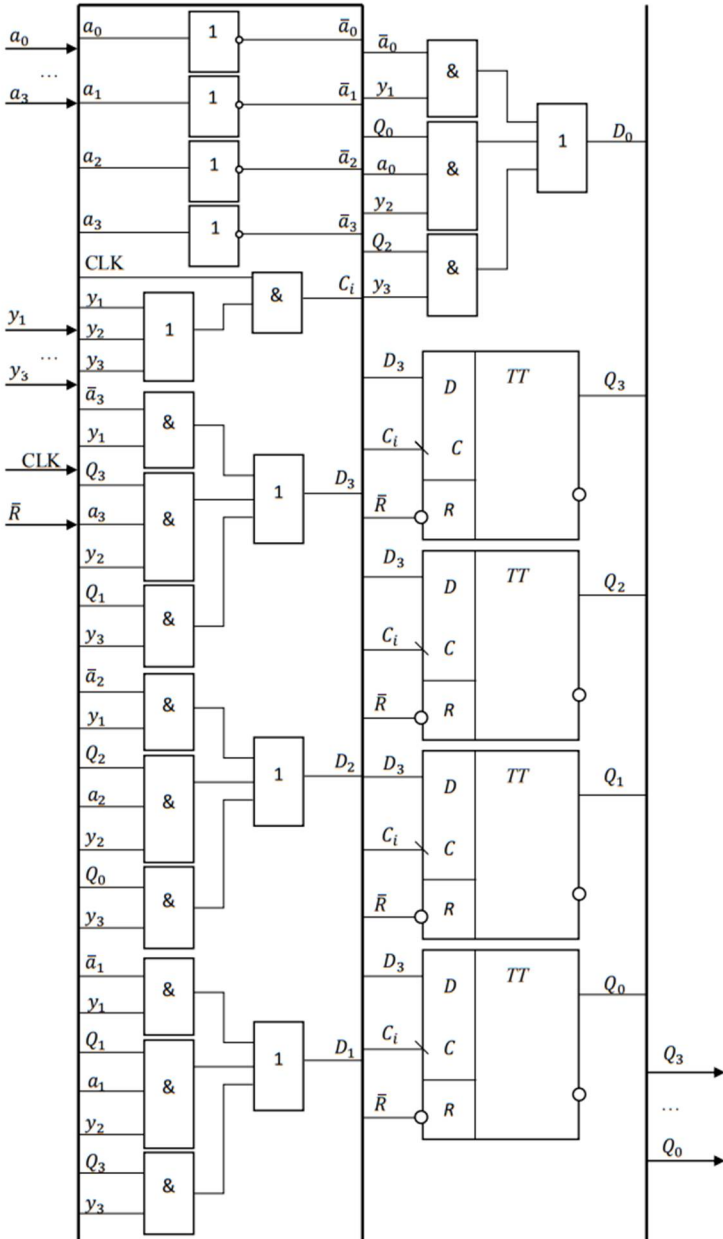
4) Умовне графічне позначення отриманого БФР:



5) Цоколівка БФР в корпусі DIP16:



6) Синтез логічної схеми заданого БФР. У виходах D-тригера присутній вихід NotQ, який у наступній схемі не виводиться на шину, адже він не використовується (що можна побачити у розрахунках). Якщо у ваших розрахунках такий вихід присутній, його обов'язково необхідно виводити.



2.1.3 Варіанти індивідуальних завдань

№ непар- ного варіанту	Набір мікрооперацій y_i			
	1,7,15	2,4,8	1,5,9	2,6,10
	1	3	5	7
	9	11	13	15
	17	19	21	22
	25	27	29	31
№ парного варіанту	2	4	6	8
	10	12	14	16
	18	20	22	24
	26	28	30	32
	13,15, 11	14,6,12	13,4, 7	14,5,8
	Набір мікрооперацій y_i			

Познач.	Зміст мікрооперацій
y1	$Rg := \text{NOT}(A)$
y2	$Rg := \text{NOT}(B)$
y3	$Rg := Rg \wedge A$
y4	$Rg := Rg \vee B$
y5	$Rg := Rg \text{ xor } A$
y6	$Rg := \text{NOT}(Rg)$
y7	$Rg := \text{LC2}(Rg)$
y8	$Rg := \text{RC2}(Rg)$
y9	$Rg := \text{LC3}(Rg)$
y10	$Rg := \text{RC3}(Rg)$
y11	$Rg := \text{R2}(Rg)$
y12	$Rg := \text{L2}(Rg)$
y13	$Rg := A$
y14	$Rg := B$
y15	$Rg := \text{LC1}(A)$

2.2 Дослідницька частина роботи. Моделювання та верифікація схеми БФР за допомогою САПР Active-HDL структурним способом

2.2.1 Короткі теоретичні відомості

Система проектування Active-HDL є сьогодні провідною середовищем проектування для створення проектів та їх моделювання для всіх сімейств програмованих інтегральних схем ПЛІС, забезпечуючи гнучкість підходу і надаючи розвинені функції підтримки найбільш складних сучасних проектів. Середою проектування Active-HDL дає розробникам незалежність у використанні інших засобів проектування з єдиного, повністю інтегрованого оточення.

Тісна інтеграція всіх додатків забезпечує абсолютний контроль над проектом від специфікації на всьому шляху до його фізичної реалізації. Active-HDL також має інтерфейси з усіма провідними програмними продуктами, надаючи розробникам свободу у використанні тих засобів проектування, які найбільш повно відповідають вимогам кожного конкретного проекту.

Блок управління проектом Active-HDL дозволяє розробнику виконувати всі модифікації та операції над проектом з єдиного оточення; така інтеграція дає кращий контроль над процесом проектування і зберігає час, усуваючи необхідність у запуску безлічі віконних інтерфейсів і процесів.

Головним принципом підходу до перевірки задач програмування за допомогою засобів САПР є максимальне використання функціональних можливостей пакету САПР для автоматизації процесу перевірки рішення задач в системах тестування. Такий підхід дозволяє звільнити дорогоцінний час викладача, який раніше витрачався на рутинну перевірку таких завдань.

Розширений маршрут проектування

Маршрут проектування компанії Aldec не залежить від виробника ПЛІС і налаштовується на будь-які комбінації засобів логічного синтезу і топологічний реалізації ПЛІС. Розробник цифрових систем може робити проекти на ПЛІС будь-якого виробника на ринку з єдиної інтегрованої середовища.

Робоча середою проектування

Розробники можуть одночасно відкривати декілька проектів і інтегрувати їх в один супер-проект. Мультипроектна робоча область надає середовище проектування, що дозволяє користувачам керувати

всіма завантаженими проектами, перемикається між ними, редагувати їх ресурси і конфігурувати їх незалежно один від одного. Всі модулі можуть розроблятися окремо один від одного і потім інтегруватися разом як один проект верхнього рівня.

Групової розробки

Збільшення розмірів проектів змушує багатьох розробників ПЛІС використовувати методи групової розробки при проектуванні. Active-HDL надає засоби підвищення продуктивності, наприклад такі, як засіб управління завданнями для задач, що займають велике час виконання, таких як моделювання, логічних синтез і топологічна реалізація. Кожна задача може бути призначена на віддалений сервер або кластер інших комп'ютерів для виконання відповідного завдання, таким чином вивільняючи комп'ютер розробника для виконання інших завдань. Система Active-HDL також надає інтерфейс до більш ніж 15 найбільш популярних систем управління версіями (RCS).

Управління проектом

Система управління проектами забезпечує швидке та ефективне управління всіма ресурсами проектів. Розробники можуть використовувати її для:

- додавання, видалення, перегляду, модифікації або виконання інших операцій над файлами ресурсів проекту;
- перегляду вмісту робочої бібліотеки, бібліотеки результатів логічного синтезу і бібліотеки часових параметрів поточного проекту;
- перегляду сконструйованої структури проектного модуля;
- перегляду об'єктів, визначених всередині окремих областей проектного модуля.

Управління бібліотеками

Система управління бібліотеками надає ефективне управління всіма бібліотеками в середовищі проектування Active-HDL. Користувачі можуть виконувати такі операції над бібліотеками та їх вмістом:

- приєднання, від'єднання і видалення бібліотек;
- редагування логічних імен бібліотек;
- стиснення і звільнення бібліотек;
- перегляд вмісту бібліотек;
- перегляд вихідних файлів певних бібліотечних модулів;
- видалення певних бібліотечних модулів;
- пошук проектних модулів в бібліотеках.

Графічний редактор в системі Active-HDL

Графічний редактор в системі Active-HDL призначений для розробки апаратури шляхом інтерактивного введення графічних елементів схеми з можливістю подальшого однозначного перетворення отриманої моделі в код на мовах VHDL або Verilog.

Складовими частинами графічного редактора є основне меню і меню редактора тексту і графіки, поле розробки (конструювання) з урахуванням вимог ЕСКД і редагування схеми, меню елементів діаграми.

Схема модельованого пристрою може містити як елементи, відповідні структурному опису (компоненти, шини тощо), так і процеси. У разі процесів використовуються спеціальні текстові блоки (Graphical Process), які допускається змінювати безпосередньо у вікні графічного редактора схем.

Розроблена схема може використовуватися в проєкті на будь-якому рівні ієрархії розроблюваної системи, тобто схемою може бути представлений як окремий блок або вузол, так і пристрій в цілому. Система Active-HDL дозволяє створювати змішані проєкти, в яких використовується як схемне представлення блоків, так і будь-яке інше опис (текстове або у вигляді кінцевого автомата).

2.2.2 Хід роботи. Приклад виконання дослідницької частини лабораторної роботи

Для початку роботи необхідно створити новий проєкт.

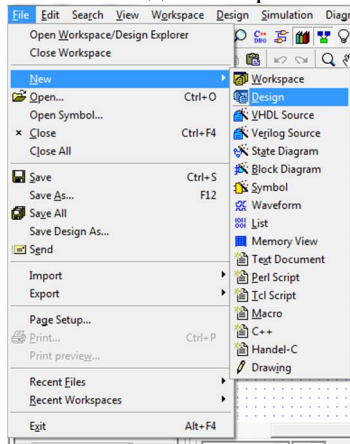


Рисунок 2.10 – Створення нового проєкту

У полі **Type the workspace name** треба ввести ім'я робочої області й натиснути **OK** (рис. 2.11).

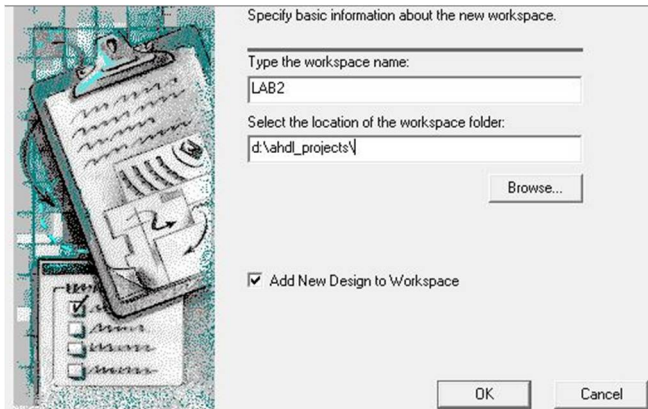


Рисунок 2.11 – Ім'я робочої області в Active-HDL

У вікні **New Design Wizard** обрати **Create an empty design** та натиснути **Next** (рис. 2.12).

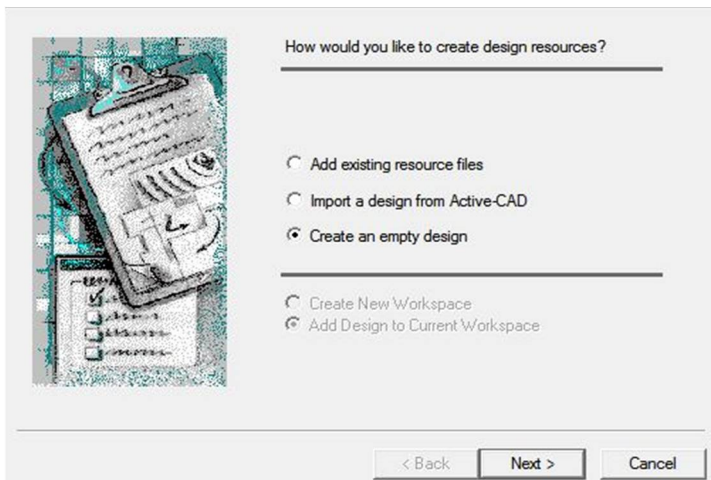


Рисунок 2.12 – Створення пустого проекту в Active-HDL

Далі у випадяючому списку **Default HDL Language** обрати мову **VHDL** та натиснути **Next** (рис. 2.13).

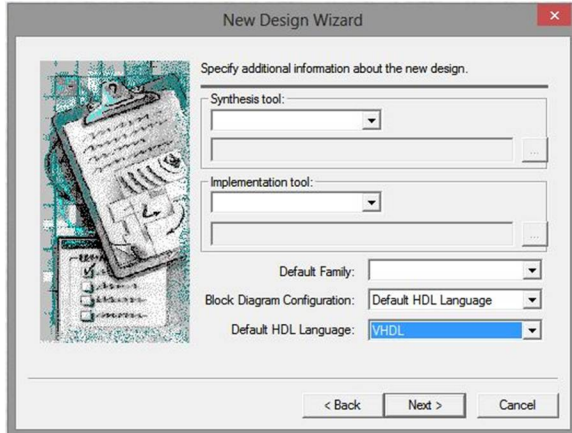


Рисунок 2.13 – Вибір мови

Відкривається вікно **Майстер Нового Проекту** (New Design Wizard). У полі **Type the design name** необхідно ввести назву мікросхеми й натиснути **Next** (рис. 2.14).

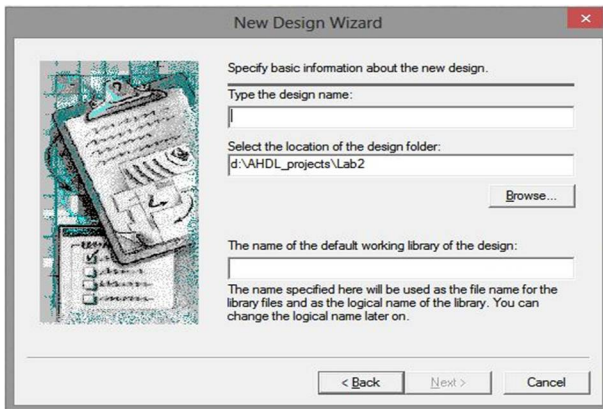


Рисунок 2.14 – Назва мікросхеми в Active-HDL

У вікні, що з'явилося натиснути **Finish**.

Створення схеми бібліотечного D-тригера

Спосіб 1 – Моделювання схеми D-тригера за допомогою VHDL Source Code. (Підходить для Active-HDL версії 9.1 та новіше)

Двічі клацніть на піктограмі **Add New File** у вікні **Design Browser**, перейти на вкладку **Wizards** та обрати пункт **VHDL Source Code** (рис. 2.15).

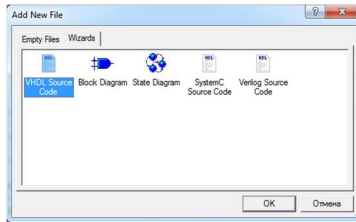


Рисунок 2.15 – Створення нового файлу

Наступним кроком треба задати ім'я мікросхеми (рис. 2.16).

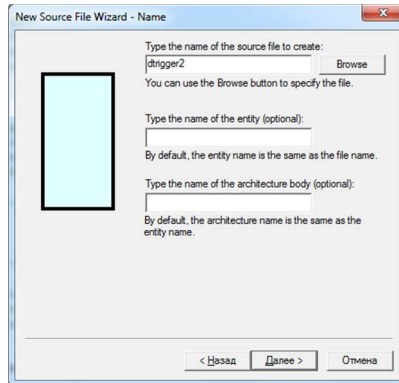


Рисунок 2.16 – Завдання імені мікросхеми

У вікні **New Source File Wizard – Ports** необхідно вказати входи та виходи мікросхеми (рис. 2.17). Для цього потрібно

натиснути на кнопку **New**, ввести ім'я порту в поле **Name** і вказати в меню **Port Direction** тип порту (in-вхід, out-вихід, inout-двонаправлений порт).

В розділі **Array Indexes** можна задавати розрядність портів (відлік починається з 0). Натиснути кнопку **Finish** для генерації вхідних та вихідних портів мікросхеми.

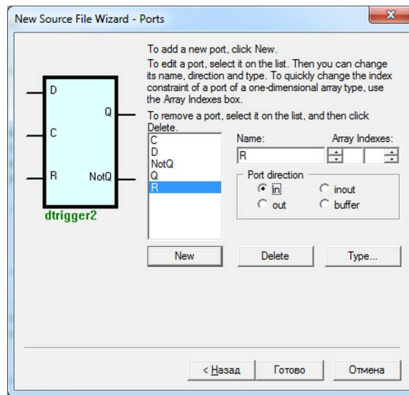


Рисунок 2.17 – Входи та виходи мікросхеми D-тригера

У отриманому файлі із розширенням .vhd у блоці entity прописується наступний код, враховуючи назви портів.

```
begin
    process (C, R) begin
        if (R='0') then
            Q<='0';
            NotQ<='1';
        elsif (C'event and C = '1') then
            Q<=D;
            NotQ<=not(D);
        end if;
    end process;
```

Важливо!!! C'event вказує на роботу за переднім фронтом імпульсу.

Компілюємо файл, натиснувши **Compile All**.

Спосіб 2 – Моделювання схеми D-тригера за допомогою Block Diagram Wizard (версія 1). (Підходить для Active-HDL версії 6.1.)

Двічі клацніть на піктограмі **Add New File** у вікні **Design Browser** та на вкладці **Wizards** обрати пункт **Block Diagram Wizard**.

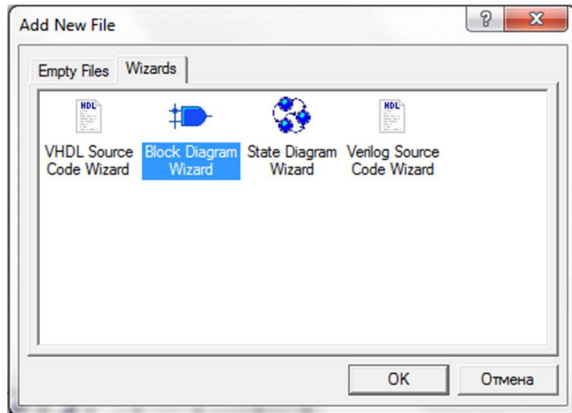


Рисунок 2.18 – Запуск майстра Block Diagram Wizard

У вікні **New Source Wizard –Language** обрати мову VHDL та натиснути **Next**.

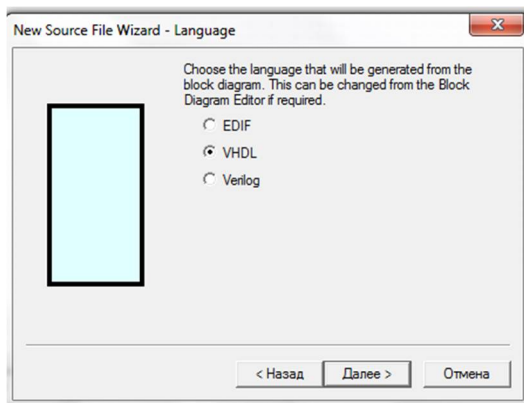


Рисунок 2.19 – Вибір мови програмування

У вікні **New Source File Wizard – Ports** необхідно вказати входи та виходи мікросхеми (рис. 2.17).

У головному вікні середовища Active-HDL з'являється нова закладка з ім'ям відкритого в цьому редакторі файлу **.bde**. Це схемний файл. Він доданий і в вікно перегляду проекту **Design Browser**.

У робочій області схемного редактора видно тільки графічні зображення портів. Для розробки схеми розгорнемо вікно редактора **Block Diagram Editor** і відшукаємо інструментальну панель **Diagram Items - VHDL** - «Елементи схеми VHDL» (рис. 2.20).



Рисунок 2.20 – Інструментальна панель **Diagram Items – VHDL**

На першому кроці необхідно знайти і розмістити на кресленні схеми графічні зображення потрібних елементів.

Натиснути по іконці **Show Symbols Toolbox** або на гарячу клавішу S. З'явиться діалогова панель (рис. 2.21). У режимі, заданому за замовчуванням, для малювання схеми доступна тільки одна бібліотека з вбудованими символами **Built-in symbols**.

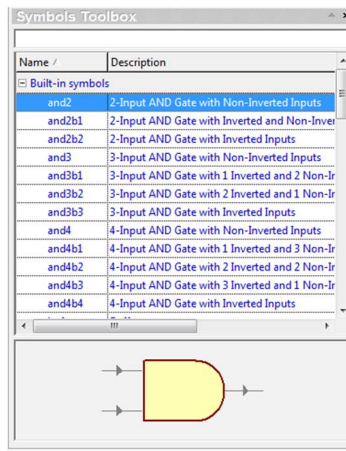


Рисунок 2.21– Діалогова панель **Symbols Toolbox**

Далі необхідно обрати для побудови мікросхеми елементи та перетягнути їх до робочої області.

Після цього елементи схеми необхідно з'єднати провідниками відповідно структури обраної мікросхеми. Для цього необхідно натиснути на панелі інструментів **Diagram Items** кнопку **Wire**. Помістіть курсор в точку, з якої потрібно почати провідник, та натисніть лівою кнопкою миші (ЛКМ). Перемістіть курсор в точку призначення знову натисніть ЛКМ. Треба зібрати схему D-тригера (рис. 2.22).

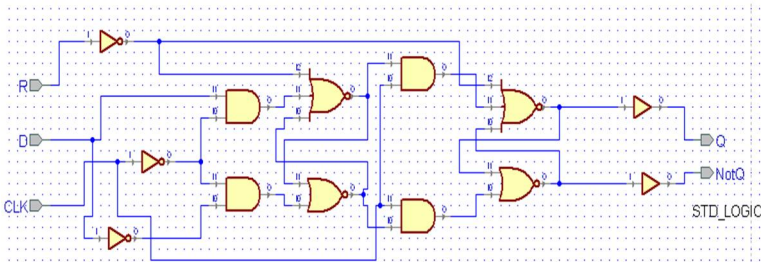


Рисунок 2.22 - схема D-тригера

Перед виходами встановлені **буфери зворотного зв'язку** для узгодження швидкостей передачі даних в лініях зв'язку. Їх потрібно обов'язково встановлювати, інакше тригер не працюватиме.

Спосіб 3 – Моделювання схеми D-тригера за допомогою Block Diagram Wizard (версія 2). Поточна схема підходить для Active-HDL версії 9.1.

Послідовність дій така ж сама, як у другому способі, тільки схема тригера буде виглядати так:

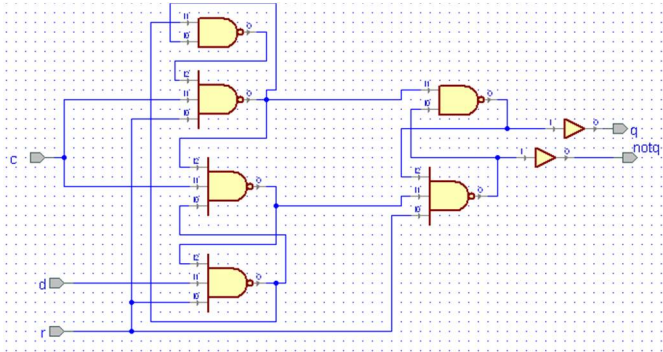


Рисунок 2.23 - схема D-тригера

Перед виходами також встановлені буфери зворотного зв'язку.

Після створення D-тригера треба перевірити правильність його роботи. Для цього необхідно створити випробувальний стенд – Test Bench (послідовність етапів згідно другої лабораторної роботи). У файлі часових діаграм задаємо характеристики входних сигналів в згідно рисункам 2.24 – 2.26.

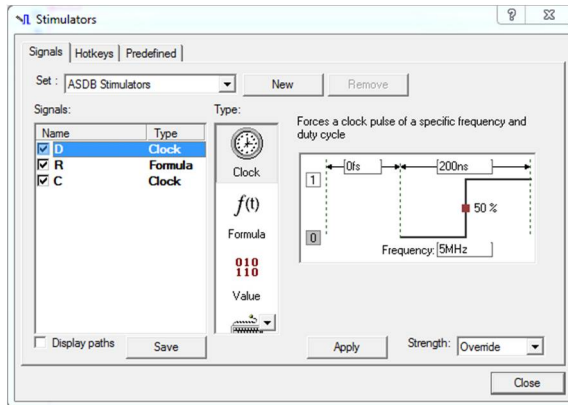


Рисунок 2.24 – Значення сигналу D

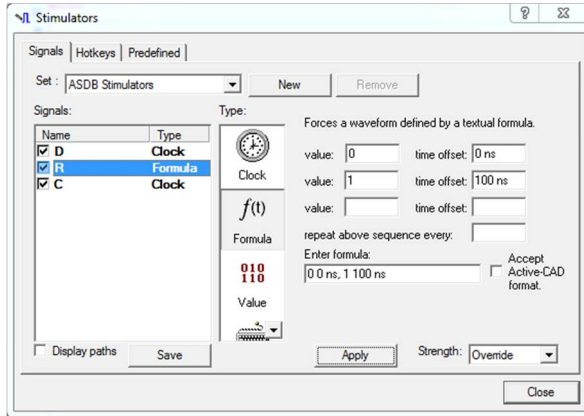


Рисунок 2.25 – Значення сигналу R

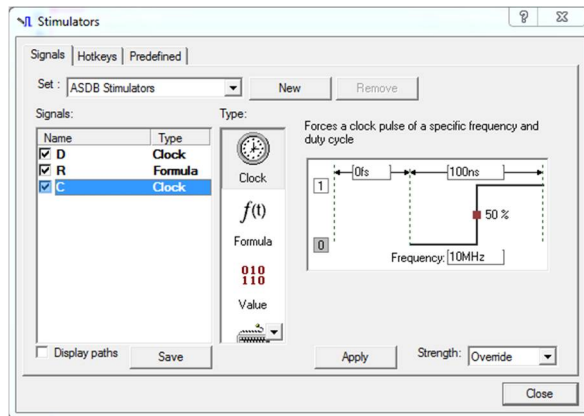


Рисунок 2.26 – Значення сигналу C

В результаті отримаємо часову діаграму роботи D-тригера, яка зображена на рис. 2.27. Проаналізувавши діаграму, зробимо висновок, що D-тригер працює по передньому фронту синхроімпульсу.

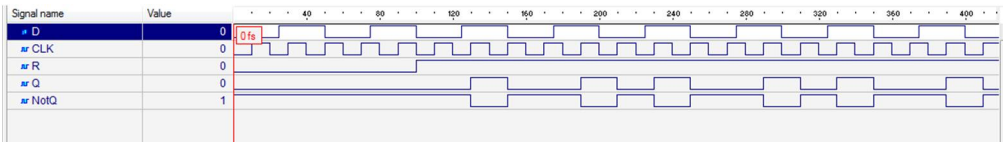


Рисунок 2.27 – Часова діаграма роботи D-тригера

Створення схеми БФР

Початок роботи такий самий, як у другому способі створення D-тригера.

Створюється **Block Diagram Wizard**, та збирається схема згідно варіанта. У вікні **New Source File Wizard – Ports** необхідно вказати входи та виходи мікросхеми (рис. 2.28).

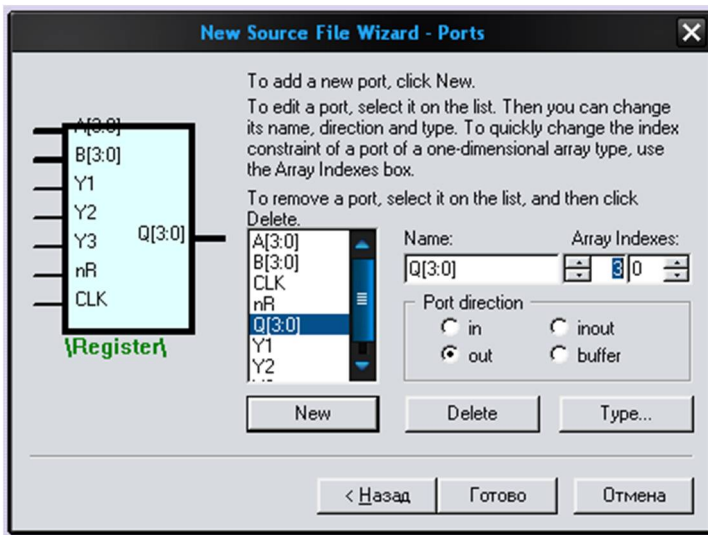


Рисунок 2.28 – Вхідні та вихідні порти регістра

Схема регістра згідно прикладу:

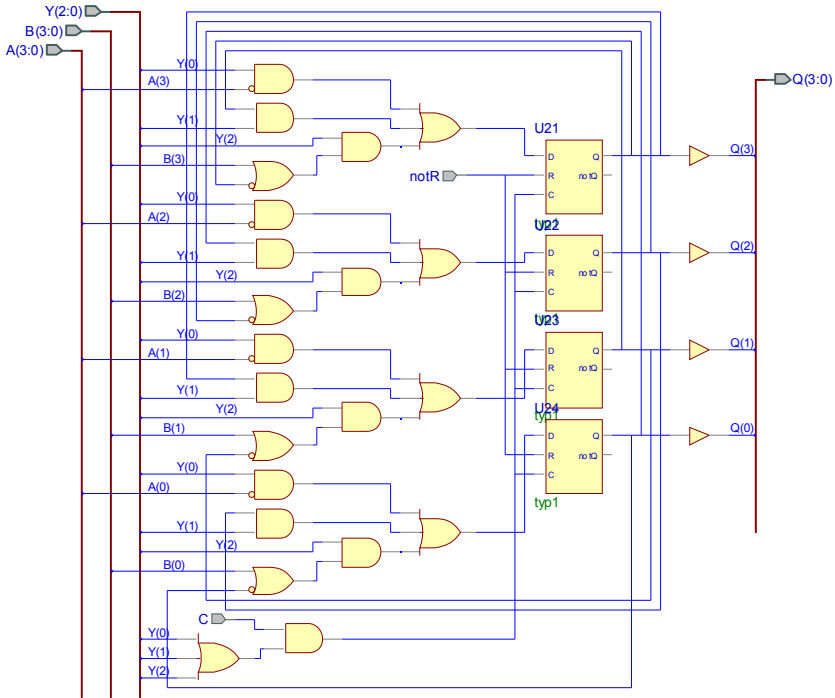


Рисунок 2.29 – Приклад закінченої схеми

Важливо! NotQ у заданих операціях не використовується, тому ці виходи не виводяться шину.

Важливо! Зверніть увагу на наявність буферів зворотного зв'язку на виходах, як у лабораторній №2.

Після підключення всіх логічних елементів необхідно дати назву з'єднанням які виходять і входять в шину.

Важливо! Назву (ідентифікатор) потрібно встановлювати одразу на виході з шини.

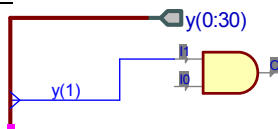


Рисунок 2.30 – Приклад встановлення ім'я з'єднання

Щоб зробити ім'я видимим на екрані монітора, досить двічі клацнути на провіднику мишею, потім на діалоговій панелі **Wire Properties**, що відкрилася, вибрати закладку **View Text** і встановити прапорець **NAME**.

Після цього корисно виконати електричний контроль схеми (команда **Check Diagram** в випадаючому меню **Diagram**). Система не тільки проінформує вас про число помилок і попереджень, а й покаже ті місця на схемі, де вони виявлені.

Для перевірки правильності роботи мікросхеми рекомендується створити випробувальний стенд (**TestBench**). Алгоритм створення **TestBench** аналогічний алгоритму у лабораторній роботі №2.

Задаємо CLK, notR, A, B та Y. Якщо, наприклад, $Y1: Rg := \text{NOT}(A)$, то ми задаємо Y(001) і, наприклад, A(0101) і на виході Q повинні отримати (1010). Після перевірки Y1 аналогічним чином перевіряємо Y2 та Y3. Якщо всі значення сходяться то можна зробити висновок, що схема побудована правильно.

******Додаткове завдання:*

На наступному кроці графічне описання проекту (у нашому прикладі - схема) повинно бути конвертовано в текстовий опис (в нашому прикладі - в **формат мови VHDL**). Виконується така робота командою **Generate HDL Code**, яку можна активізувати клацанням по іконці або з випадаючого меню **Diagram**. Виконавши її, ви виявите в вікні перегляду проекту **Design Browser** новий файл з розширенням **.vhd**.

Зміст цього файлу можна переглянути двічі натиснувши на ньому у вікні **Design Browser**. Порівнюючи графічне і текстове описання мікросхеми, виявите їх повну відповідність.

Натискаємо на цьому файлі правою кнопкою миші та виконуємо команду **Compile**.

Необхідно ще задати часові діаграми на входах мікросхеми, без них симулятор відмовляється працювати.

Часові діаграми зовнішніх впливів задаються у спеціальному вікні редактора сигналів **Waveform Editor**, який викликається клацанням по піктограмі **New Waveform** на інструментальній панелі **Standard**.

Далі необхідно натиснути праву кнопку миші у лівій частині вікна та вибрати в контекстному меню команду **Add Signals**.

Натисканням кнопки **Add** перенесемо потрібні сигнали у вікно редактора **Waveform Editor**.

Виділимо один або кілька програмованих сигналів і натисканням на іконці **Stimulators** виконаємо однойменну команду **Stimulators**. У вікні **Signals** виділимо перший сигнал, в вікні **Type** визначимо для нього тип стимулятора. Натиснемо кнопку **Apply** і, не закриваючи панель, перейдемо до програмування інших сигналів.

Виконаємо ініціалізацію схеми (команда **Initialize Simulation** з меню **Simulation**). Поряд з іменами сигналів в стовпці **Value** з'являться початкові значення всіх сигналів.

Запускаємо схему на моделювання натисканням на гарячу клавішу **F5** або команду **Run For** з меню **Simulation**.

Для перевірки правильності роботи мікросхеми рекомендується створити випробувальний стенд (**TestBench**). Алгоритм створення **TestBench** аналогічний алгоритму у лабораторній роботі №2.

Задаємо CLK, notR, A, B та Y. Якщо, наприклад, $Y1: Rg := NOT(A)$, то ми задаємо $Y1=1, Y2=0, Y3=0$ (або $Y = 001$) і, наприклад, $A(0101)$. На виході Q повинні отримати (1010) . Після перевірки Y1 аналогічним чином перевіряємо Y2 ($Y1=0, Y2=1, Y3=0$ або $Y = 010$) та Y3 ($Y1=0, Y2=0, Y3=1$ або $Y = 100$). Якщо всі значення сходяться то можна зробити висновок, що схема побудована правильно. Приклад часової діаграми представлений на рис. 2.31.

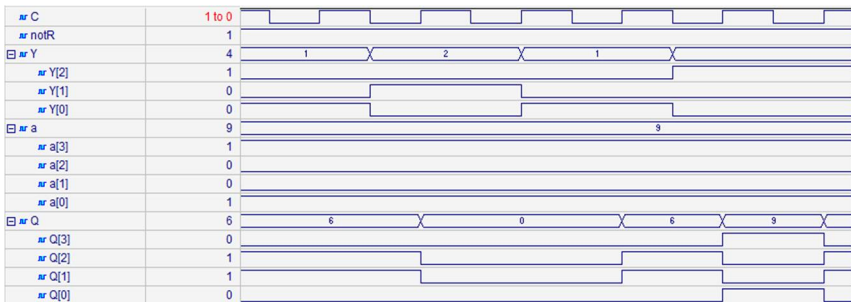


Рисунок 2.31 – Приклад часової діаграми

Висновки: засвоїли основні правила проектування схем багатофункціональних регістрів та виконали верифікацію отриманої схеми способом моделювання за допомогою схемного редактора Active-HDL. Проаналізувавши часову діаграму роботи мікросхеми робимо висновок, що схема працює правильно і відповідно заданим мікроопераціям.

2.3 Зміст звіту з лабораторної роботи

2.3.1 Логічна схема пристрою, що проектується, з описом вихідних даних та **всіх етапів проектування** (див. ПРИКЛАД виконання проектної частини роботи).

2.3.2 Схема (або код), перевірка та часова діаграма роботи D-тригера.

2.3.3 Логічна схема БФР, спроектована у середовищі Active-HDL.

2.3.4 Часова діаграма результатів моделювання схеми БФР.

2.3.5 Висновки щодо проектування та правильності роботи спроектованої схеми БФР.

2.4 Контрольні питання

2.4.1 Що дозволяє блок управління проектом Active-HDL?

2.4.2 Що є головним принципом підходу до перевірки задач програмування за допомогою засобів САПР?

2.4.3 Для чого призначений графічний редактор Active-HDL?

2.4.4 Як вказати вхідні та вихідні порти мікросхеми?

2.4.5 Де знаходяться логічні елементи для побудови заданої мікросхеми?

2.4.6 Для чого необхідний TestBench?