# 2 Лабораторна робота № 2

## Дослідження дешифраторів

Мета: вивчити принципи побудови дешифраторів та їх опис на мові Verilog

## Опис дешифратора на мові Verilog

Повний дешифратор 2х4 описується таблицею істинності, показаною в таблиці 2.1

Таблиця 2.1 – Таблиця істинності дешифратора

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

В результаті логічні рівняння, що описують дешифратор буть мати такий вигляд:



Для опису дешифратора будемо використовувати логічні рівняння. В результаті отримаємо наступний код на мові Verilog (Лістинг 2.1)

module DC1

(

input [1:0] x\_i,

output [3:0] y\_o

);

assign y\_o[0] = !x\_i[1] && !x\_i[0];

assign y\_o[1] = !x\_i[1] && x\_i[0];

assign y\_o[2] = x\_i[1] && !x\_i[0];

assign y\_o[3] = x\_i[1] && x\_i[0];

endmodule

Лістинг 2.1 – Дешифратор 2х4

В результаті компіляції ми отримаємо у **RTL Viewer** схему, що описує дешифратор 2х4 у відповідності до логічних рівнянь.

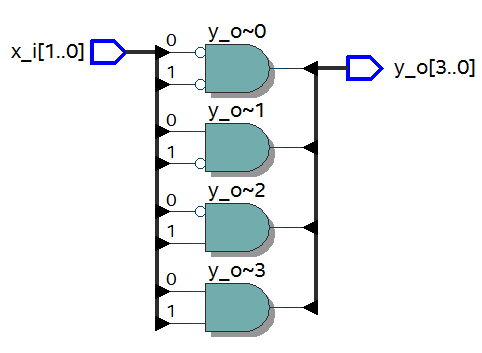


Рисунок 2.1 – Результат компіляції лістингу 2.1 у RTL Viewer

Для тестування дешифратора будемо використовувати тестбенч, наведений у лістингу 2.2.

**`timescale** **1**ns / **1**ps

**module** testbench;

// input and output test signals

**reg** [**1**:**0**] x\_i;

**wire** [**3**:**0**] y\_o;

// creating the instance of the module

// we want to test

// DC1 - module name

// dut - instance name

//('dut' means 'device under test')

**DC1** dut ( x\_i, y\_o );

// do at the beginning of the simulation

**initial**

**begin**

x\_i = **2'b00**; // set test signals value

#**10**; // pause

x\_i = **2'b01**; // set test signals value

#**10**; // pause

x\_i = **2'b10**; // set test signals value

#**10**; // pause

x\_i = **2'b11**; // set test signals value

#**10**; // pause

**end**

// do at the beginning of the simulation

// print signal values on every change

**initial**

$monitor("x\_i=%b y\_o=%b", x\_i, y\_o);

**endmodule**

Лістинг 2.2 – Тестбенч для тестування дешифратора 2х4

В результаті симуляції будуть отримані часові діаграми, наведені на рисунку 2.2.

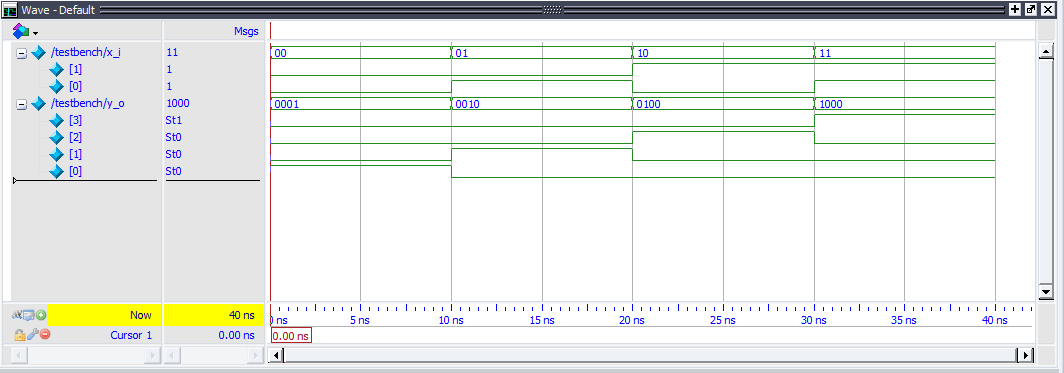


Рисунок 2.2 – Результат симуляції дешифратора 2х4

Як видно з рисунку 2.2 робота дешифратора, що описується лістингом 2.1 повністю відповідає таблиці істинності дешифратора 2х4.

## Опис дешифратора за допомогою оператора case

В цьому випадку ми будемо використовувати оператор **case**, який розміщений всередині блока **always**. В кожному рядку оператора **case** буде виконуватись блокуюче присвоювання (**blocking assignment**) вихідному сигналу.

Блок **always** виконується в тому разі, коли відбувається зміна сигналів у списку чутливості (**sensitivity list**). Для комбінаційної схеми у список чутливості необхідно включати всі сигналу, що є входами комбінаційної схеми. Виходи в список чутливості не включаються. В нашому лістингу ми використовуємо знак **\***, який говорить про те, що в список чутливості включаються всі сигнали, які впливають на результат блоку **always**.

Дешифратор буде описуватись кодом, наведеним у лістингу 2.3.

module DC2

(

input [1:0] x\_i,

output reg [3:0] y\_o

);

always @ (\*)

begin

case( x\_i )

2'h0 : y\_o = 4'h1;

2'h1 : y\_o = 4'h2;

2'h2 : y\_o = 4'h4;

2'h3 : y\_o = 4'h8;

endcase

end

endmodule

Лістинг 2.3 – Опис дешифратора за допомогою оператора case

В результаті компіляції ми отримаємо у **RTL Viewer** схему, яка показує дешифратор. Тобто компілятор пакету **Quartus Prime** розпізнав опис за допомогою оператора **case**, як опис дешифратора.

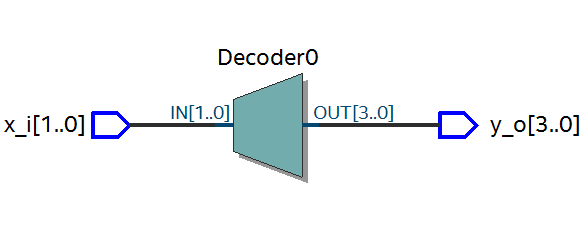


Рисунок 2.3 – Результат компіляції лістингу 2.3 у RTL Viewer

Тестбенч залишається той самий і результат симуляції буде збігатися з показним вище.

## Дешифратор з входом дозволу роботи

Дешифратор з входом дозволу роботи буде описуватись за допомогою таблиці істинності, показаної в таблиці 2.2

Таблиця 2.2 – Таблиця істинності дешифратора із входом дозволу роботи

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| OE | X1 | X0 | Y0 | Y1 | Y2 | Y3 |
| 0 | X | X | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 |

Дешифратор буде описуватись кодом, наведеним у лістингу 2.4.

module DC3

(

input [1:0] x\_i,

input ena\_i,

output reg [3:0] y\_o

);

always @ (\*)

begin

y\_o = 0;

if( ena\_i ) begin

case (x\_i)

2'h0 : y\_o = 4'h1;

2'h1 : y\_o = 4'h2;

2'h2 : y\_o = 4'h4;

2'h3 : y\_o = 4'h8;

endcase

end

end

endmodule

Лістинг 2.4 – Опис дешифратора із входом дозволу роботи

В результаті компіляції ми отримаємо у **RTL Viewer** схему, яка показує дешифратор та мультиплексор. Керування мультиплексором відбувається за допомогою входу **ena\_i**. У випадку, коли сигнал **ena\_i** дорівнює 0 на вихід подаються всі нулі, у випадку 1 на виході буде сигнал, що відповідає нормальній роботі дешифратора. Тобто робота буде повністю співпадати з логікою, описаною таблицею 2.2.

Для моделювання дешифратора ми використали тестбенч, описаний у лістингу 2.5.

`timescale 1ns / 1ps

module testbench;

reg [1:0] x\_i;

reg ena\_i;

wire [3:0] y\_o;

DC3 dut ( x\_i, ena\_i, y\_o );

initial

begin

ena\_i = 0;

x\_i = 2'b00;

#10;

x\_i = 2'b01;

#10;

x\_i = 2'b10;

#10;

x\_i = 2'b11;

#10;

ena\_i = 1;

x\_i = 2'b00;

#10;

x\_i = 2'b01;

#10;

x\_i = 2'b10;

#10;

x\_i = 2'b11;

#10;

end

initial

$monitor("x\_i=%b ena\_i=%b y\_o=%b", x\_i, ena\_i, y\_o);

endmodule

Лістинг 2.5 – Тестбенч для симуляції дешифратора із входом дозволом роботи

Результат симуляції дешифратора показаний на рисунку 2.4. З рисунку видно, що дешифратор працює у повній відповідності до таблиці 2.2.

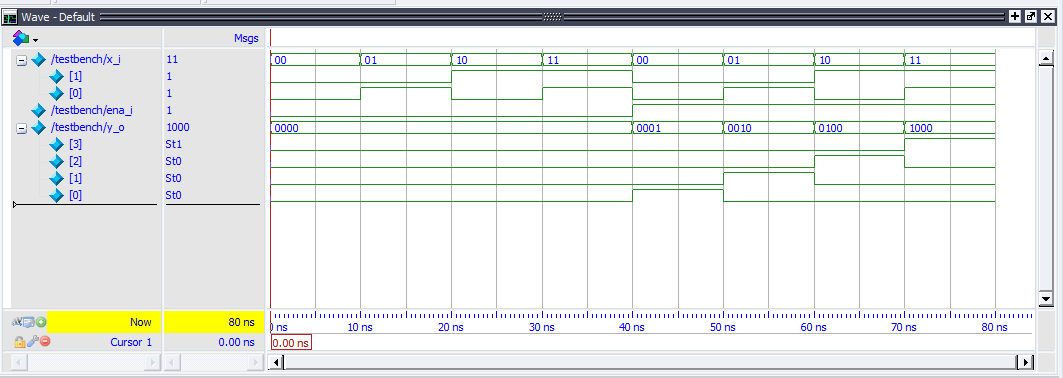


Рисунок 2.4 – Результат симуляції дешифратора із входом дозволу роботи

## Хід роботи

1. Отримати у викладача завдання на лабораторну роботу відповідно до номера свого варіанту. Якщо в таблиці не вказана кількість керуючих входів, то її необхідно визначити самостійно.

Таблиця 2.3 – Шифратори та дешифратори

|  |  |  |
| --- | --- | --- |
| № варіанта | Тип пристрою | Додаткові входи |
|  | Дешифратор 2х4, рухома «1» | Скидання: активний рівень «1» |
|  | Дешифратор 2х4, рухома «1» | Скидання: активний рівень «0» |
|  | Дешифратор 2х4, рухома «1» | Дозвіл роботи: 2 входи, об’єднані по функції «І» |
|  | Дешифратор 2х4, рухома «1» | Дозвіл роботи: 2 входи, об’єднані по функції «АЛЕ» |
|  | Дешифратор 2х4, рухома «1» | Установка всіх виходів в «1» |
|  | Дешифратор 2х4, рухомий «0» | Скидання: активний рівень «1» |
|  | Дешифратор 2х4, рухомий «0» | Скидання: активний рівень «0» |
|  | Дешифратор 2х4, рухомий «0» | Дозвіл роботи: 2 входи, об’єднані по функції «І» |
|  | Дешифратор 2х4, рухомий «0» | Дозвіл роботи: 2 входи, об’єднані по функції «АЛЕ» |
|  | Дешифратор 2х4, рухомий «0» | Установка всіх виходів в «1» |
|  | Дешифратор на 3 виходи, рухома «1» | Скидання: активний рівень «1» |
|  | Дешифратор на 3 виходи, рухома «1» | Скидання: активний рівень «0» |
|  | Дешифратор на 3 виходи, рухома «1» | Дозвіл роботи: 2 входи, об’єднані по функції «І» |
|  | Дешифратор на 3 виходи, рухома «1» | Дозвіл роботи: 2 входи, об’єднані по функції «АЛЕ» |
|  | Дешифратор на 3 виходи, рухома «1» | Установка всіх виходів в «1» |
|  | Дешифратор на 3 виходи, рухомий «0» | Скидання: активний рівень «1» |
|  | Дешифратор на 3 виходи, рухомий «0» | Скидання: активний рівень «0» |
|  | Дешифратор на 3 виходи, рухомий «0» | Дозвіл роботи: 2 входи, об’єднані по функції «І» |
|  | Дешифратор на 3 виходи, рухомий «0» | Дозвіл роботи: 2 входи, об’єднані по функції «АЛЕ» |
|  | Дешифратор на 3 виходи, рухомий «0» | Установка всіх виходів в «1» |

1. Створити таблицю істинності дешифратора.
2. Допуском до виконання лабораторної роботи є розроблена електрична принципова схема та часові діаграми її роботи, побудовані без врахування затримок. При побудові часових діаграм проглянути всі режими роботи схеми.
3. На мові Verilog написати програму, яка буде реалізовувати отриману схему у двох варіантах: опис дешифратора за допомогою логічних рівнянь та поведінковий опис дешифратора.
4. Провести симуляцію обох проектів в пакеті **ModelSim**. Побудувати часові діаграми роботи проекту. Порівняти з отриманими при виконанні домашнього завдання.
5. Створити проект, в якому зібрати схему, видану викладачем. Входи схеми підключити до кнопок або мікроперемикача, а виходи – до світлодіоду.
6. Перевірити роботу дешифратора на відлагоджувальній платі у відповідності до отриманого завдання.