# Лабораторна робота № 3

## Дослідження мультиплексорів

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

## Опис мультиплексора 2в1 на мові Verilog

Мультиплексор 2в1 описується логічним рівнянням, показаним нижче:



Для опису мультиплексора 2в1 будемо використовувати декілька варіантів опису:

* логічні рівняння,
* тернарний оператор
* оператор **if**
* оператор **case**.

При описі за допомогою логічних рівнянь отримаємо наступний код на мові Verilog (Лістинг 3.1)

module b1\_mux\_2\_1\_comb

(

input d0,

input d1,

input sel,

output y

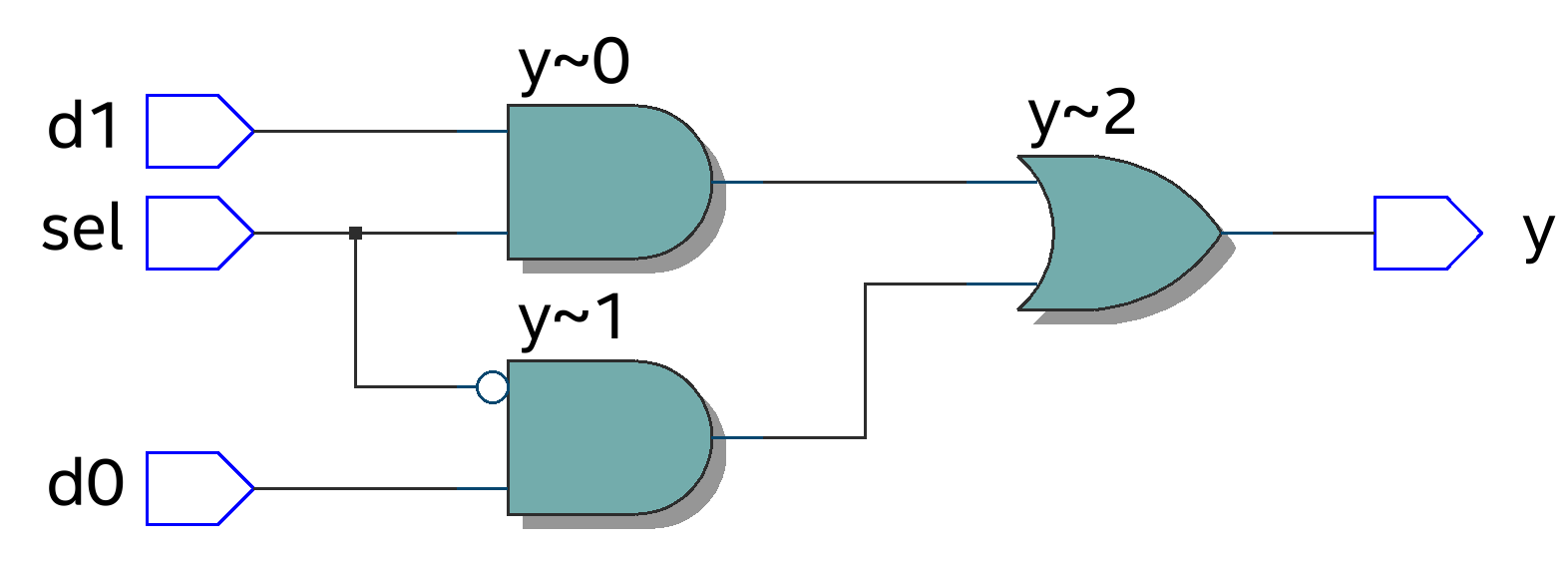
);

assign y = (sel & d1) | ((~sel) & d0);

endmodule

Лістинг 3.1 - Комбінаційний мультиплексор 2в1

Результат синтезу даного мультиплексора показаний на рисунку 3.1. Як видно, мультиплексор реалізовується на логічних елементах так, як описано у лістингу 3.1.



*Рисунок 3.1 – Результат компіляції мультиплексора, описаного логічними рівняннями, у RTL Viewer*

Для симуляції мультиплексора використаємо тестбенч, описаний нижче:

`timescale 1 ns / 100 ps

module testbench;

reg a;

reg b;

reg sel;

wire y;

b1\_mux\_2\_1\_comb b1\_mux\_2\_1\_comb a, b, sel, y);

initial

begin

a = 1'b0;

b = 1'b1;

sel = 1'b0;

#10;

sel = 1'b1;

#10

b = 1'b0;

#5

b = 1'b1;

#5;

end

initial

$monitor("a=%b b=%b sel=%b y=%b", a, b, sel, y);

endmodule

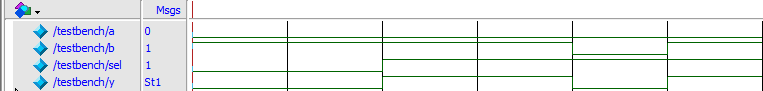
Лістинг 3.2 - Тестбенч для моделювання різних реалізацій однобітних мультиплексорів 2в1

Тестбенч містить чотири внутрішніх сигнали: *a*, *b*, *sel* і *y*, де *a* і *b* – 1-бітні вхідні сигнали, *sel* – 1-бітний сигнал вибору, а *y* – 1-бітний вихідний сигнал мультиплексора.

Блок *initial* ініціалізує входи *a*, *b* і *sel* і змінює значення *sel* після затримки в 10 одиниць часу. Потім він змінює значення вхідного параметра *b* через ще 10 одиниць часу, а потім знову змінює значення *b* після затримки в 5 одиниць часу.

Інструкція *monitor* друкує значення *a*, *b*, *sel* і *y* щоразу, коли відбувається зміна в одному з цих сигналів.

Результат тестування мультиплексору за допомогою тесбенчу Лістинг 3.2 показаний на рисунку 3.2.



*Рисунок 3.2 – Результат симуляції 1-бітного мультиплексора*

Якщо кількість входів мультиплексора і їх розрядність більше, ніж у наведеному прикладі, то синтез комбінаційної функції, яка описує мультиплексор, стає більш складним і мультиплексори реалізують іншими способами.

Одне з можливих рішень – використання тернарного оператора (потрійного вибору). Воно показане у наступному прикладі:

module b1\_mux\_2\_1\_sel

(

input d0,

input d1,

input sel,

output y

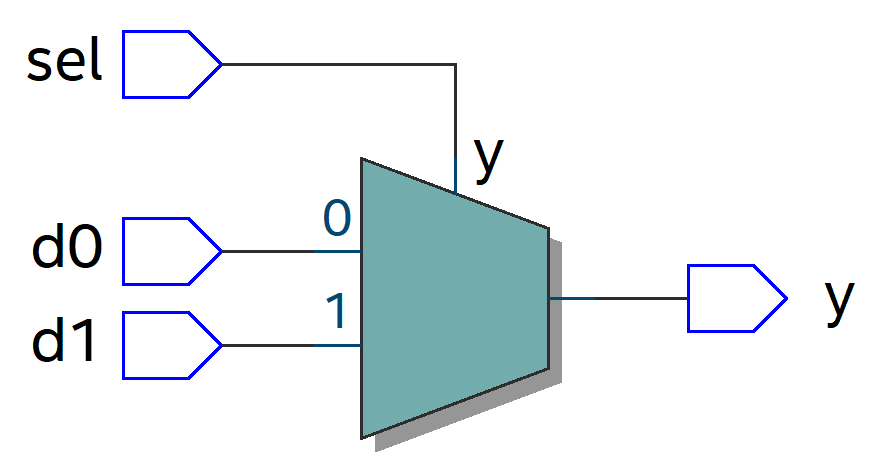
);

assign y = sel ? d1 : d0;

endmodule

Лістинг 3.3 - Мультиплексор 2в1 на основі тернарного оператора

Результат синтезу даного мультиплексора у **RTL Viewer** показаний на рисунку 3.3. Як видно, мультиплексор розпізнається компілятором пакету **Quartus Prime** саме як мультиплексор.



*Рисунок 3.3 – Результат компіляції мультиплексора, описаного тернарним оператором, у RTL Viewer*

Цей же мультиплексор можна реалізувати і за допомогою умовного оператора (*if*):

module b1\_mux\_2\_1\_if

(

input d0,

input d1,

input sel,

output reg y

);

always @ (\*)

begin

if(sel)

y = d1;

else

y = d0;

end

endmodule

Лістинг 3.4 - Мультиплексор 2в1 на основі умовного оператора

Результат синтезу мультиплексора, описаного за допомогою оператора *if*, у **RTL Viewer** буде такий самий як і показаний на рисунку 3.3 – мультиплексор розпізнається компілятором пакету **Quartus Prime** як мультиплексор.

Найкращою реалізацією мультиплексора є використання оператора множинного вибору (*case*), оскільки він дозволяє найпростіше описувати складні мультиплексори з великою кількістю входів:

module b1\_mux\_2\_1\_case

(

input d0,

input d1,

input sel,

output reg y

);

always @ (\*)

begin

case (sel)

0: y = d0;

1: y = d1;

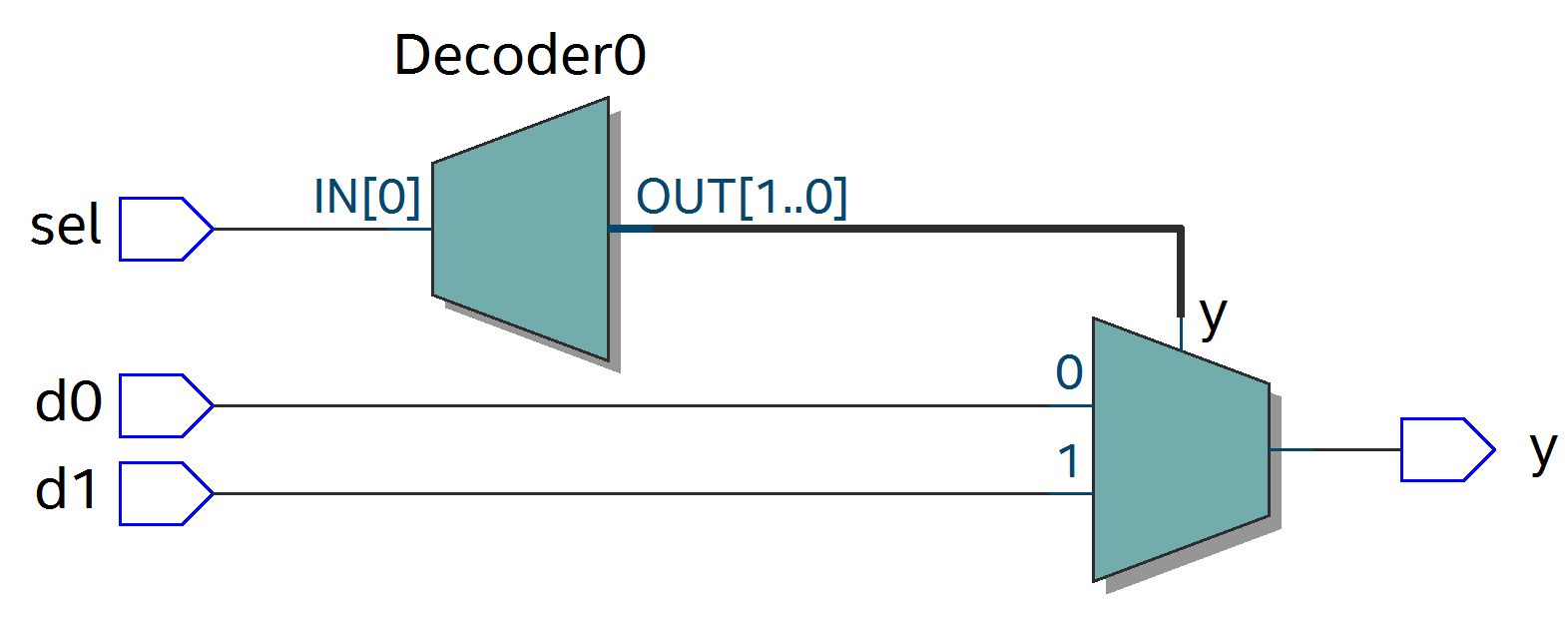
endcase

end

endmodule

Лістинг 3.5 - Мультиплексор 2в1 на основі оператора множинного вибору

Результат синтезу мультиплексора, описаного за допомогою оператора *case*, у **RTL Viewer** показаний на рисунку 3.4. Як видно, мультиплексор розпізнається компілятором пакету **Quartus Prime** саме як мультиплексор, але на вхід керування надходить декодований однобітний сигнал.



*Рисунок 3.4 – Результат компіляції мультиплексора, описаного за допомогою оператора case, у RTL Viewer*

Проте, логіка роботи всіх реалізацій мультиплексорів однакова, а при синтезі будь-який з мультиплексорів буде реалізований однаково як кобінаційна схема за допомогою таблиці перекодування (Look-Up Table) логічного елементу мікросхеми ПЛІС.

## Двохбітний мультиплексор 4в1

Коли кількість входів збільшується, збільшується також і ширина вхідного сигналу *select*. Розмір такого сигналу залежить від кількості інформаційних входів і може визначатися рівнянням:

N\_sel = ⌈log\_2 N⌉.

Такі мультиплексори найлегше реалізуються за допомогою оператора *case*:

module b2\_mux\_4\_1\_case

(

input [1:0] d0, d1, d2, d3,

input [1:0] sel,

output reg [1:0] y

);

always @(\*)

case (sel)

2'b00: y = d0;

2'b01: y = d1;

2'b10: y = d2;

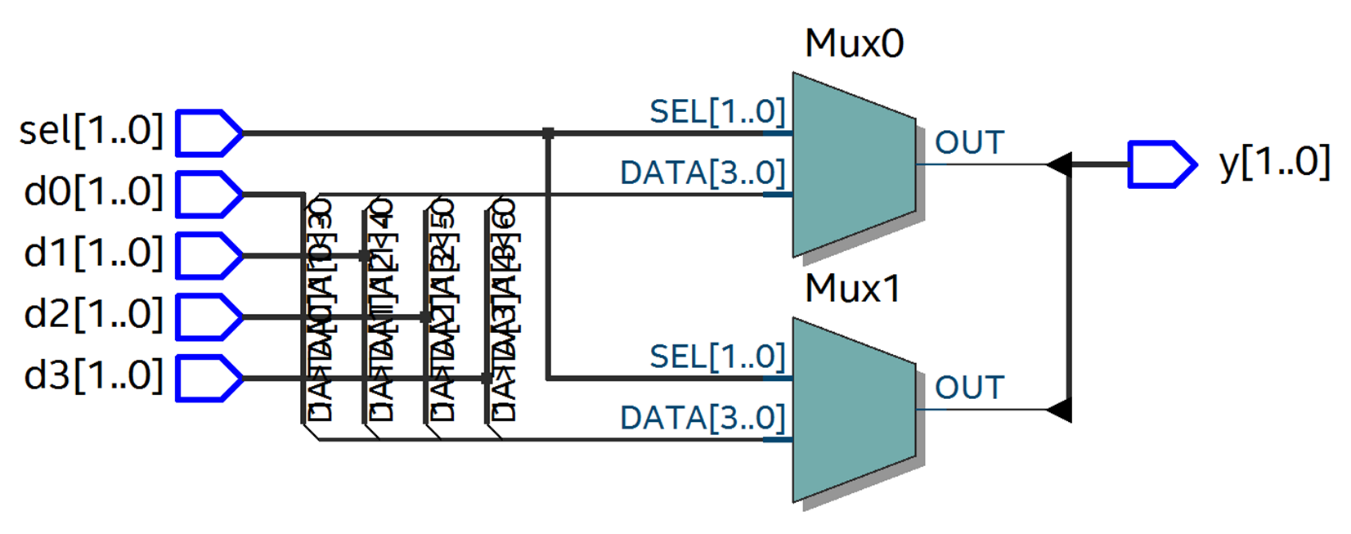
2'b11: y = d3;

endcase

endmodule

***Лістинг 3.6 - Двохбітний мультиплексор 4в1, реалізований за допомогою оператора case***

Результат синтезу двохбітного мультиплексора 4в1, описаного за допомогою оператора *case*, у **RTL Viewer** показаний на рисунку 3.5.



*Рисунок 3.5 – Результат компіляції двохбітного мультиплексора 4в1 у RTL Viewer*

Результати симуляції двохбітного мультиплексора 4в1 показані на рисунку 3.6 всіх представлених варіантів мультиплексорів збігаються:

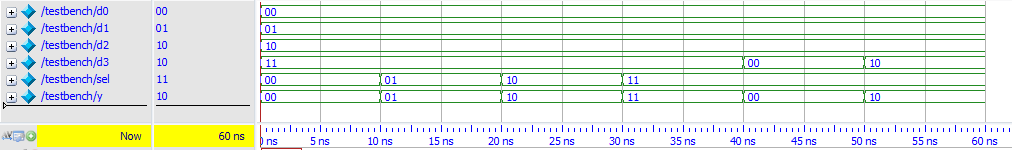


Рисунок 3.6 - Результат симуляції двохбітного мультиплексора 4в1

## Неповний мультиплексор 3в1

До цього моменту всі реалізації мультиплексорів були повними, тобто кількість вхідних інформаційних сигналів відповідала кількості комбінацій, можливих на адресному вході. Розглянемо реалізацію неповного мультиплексора з трьома інформаційними входами (відповідно, адресний вхід буде двохбітним). Вхід d0 відповідає комбінаційній адресі 2'b00, d1 - 2'b01 і d2 - 2'b10.

Реалізуємо цей мультиплексор за допомогою оператора **case**:

module b2\_mux\_3\_1\_case\_bad

(

input [1:0] d0, d1, d2,

input [1:0] sel,

output reg [1:0] y

);

always @(\*)

case (sel)

2'b00: y = d0;

2'b01: y = d1;

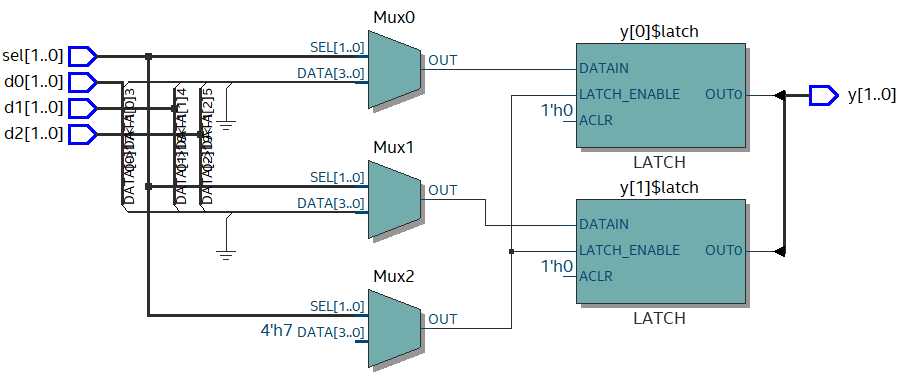
2'b10: y = d2;

endcase

endmodule

***Лістинг 3.7 - Мультиплексор 3в1 з дозволом роботи***

Для комбінацій адресних входів 2'b00, 2'b01 і 2'b10 мультиплексор поводиться так, як очікувалося, комутуючи відповідну інформацію з входів на вихід. Поведінка мультиплексора з недозволеним комбінаційною адресним входом 2'b11 наступна: пристрій зберігає свій попередній стан. Це призводить до появи регістра засувки (latch) на виході, що зазвичай є неприпустимою ситуацією. Результат синтезу мультиплексора у **RTL Viewer** показаний на рисунку 3.7.



*Рисунок 3.7 – Результат компіляції неповного мультиплексора у RTL Viewer*

На рисунку 3.8 показані результати симуляції неповного мультиплексора, а у лістингу 3.8– результат роботи оператора $monitor. Зверніть увагу, що 3 і 4 та 5 і 6 рядки мість однакове значення виходу, хоча значення входу sel змінюється на 2’b11. Саме це говорить про наявніть засувки, що зберігає останнє вихідне значення.

# d0=00 d1=01 d2=10 sel=00 y=00

# d0=00 d1=01 d2=10 sel=01 y=01

# d0=00 d1=01 d2=10 sel=10 y=10

# d0=00 d1=01 d2=10 sel=11 y=10

# d0=00 d1=01 d2=10 sel=01 y=01

# d0=00 d1=01 d2=10 sel=11 y=01

***Лістинг 3.8 - Консоль пакету ModelSim***

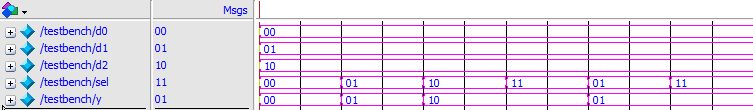


Рисунок 3.8 – Результати моделювання мультиплексора 3в1

Це поширена помилка при використанні оператора *case* з неповним набором комбінацій; та ж проблема виникатиме в неповному мультиплексорі, реалізованому за допомогою операторів *if*.

Ця помилка виправляється шляхом додавання рядка *default*, яка описує вибір за замовчуванням, як в наступному прикладі:

module b2\_mux\_3\_1\_case\_correct

(

input [1:0] d0, d1, d2,

input [1:0] sel,

output reg [1:0] y

);

always @(\*)

case (sel)

2'b00: y = d0;

2'b01: y = d1;

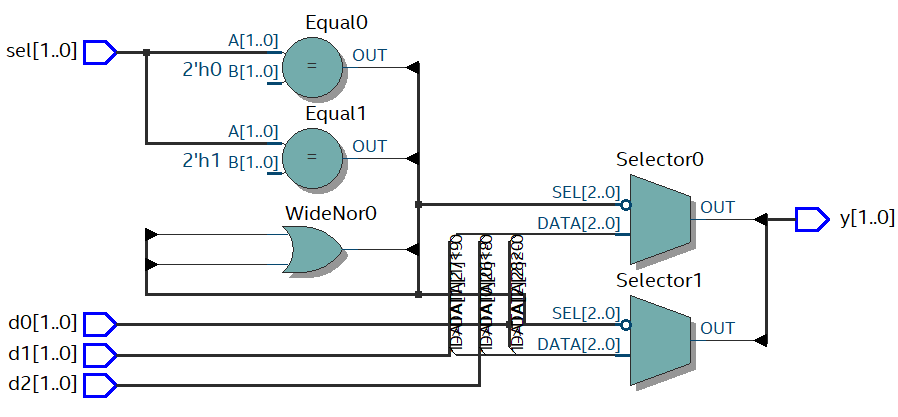
default: y = d2;

endcase

endmodule

***Лістинг 3.11 - Мультиплексор 3в1 без засувки***

Результат компляції мультиплексора у **RTL Viewer** показаний на рисунку 3.8. Для формування сигналів керування мультиплексорами використовуютється комбінаційна схема з компараторами та логічним елементом АБО. Вона забезпечує коректну роботу мультиплексора: коли комбінації 2'b10 і 2'b11 надходять на адресний вхід *sel*, сигнал з входу *d2* буде переключено на вихід.



*Рисунок 3.9 – Результат компіляції неповного мультиплексора у RTL Viewer*

На рисунку 3.10 показані результати симуляції неповного мультиплексора. Видно, що при надходженні комбінації 2'b11 на адресний вхід *sel* вихід перемикається на передачу даних з входу *d2.*

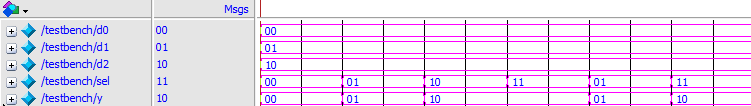


Рисунок 3.10 – Результати моделювання мультиплексора 3в1

## 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 3.1 – Мультиплексори

|  |  |  |
| --- | --- | --- |
| № варіанта | Тип пристрою | Додаткові входи |
|  | Мультиплексор 4х1 | Дозвіл роботи |
|  | Мультиплексор 6х2 | Дозвіл роботи |
|  | Мультиплексор 5х1 | Дозвіл роботи: 2 входи, об’єднані по функції «І» |
|  | Мультиплексор 6х3 | Скидання |
|  | Мультиплексор 4х2 | Дозвіл роботи: 2 входи, об’єднані по функції «АЛЕ» |
|  | Мультиплексор 3х1 | Дозвіл роботи |
|  | Мультиплексор 4х2 | Дозвіл роботи |
|  | Мультиплексор 6х2 | Дозвіл роботи |
|  | Мультиплексор 6х3 | Дозвіл роботи |
|  | Мультиплексор 5х1 | Скидання |
|  | Мультиплексор 6х3 | Дозвіл роботи: 2 входи, об’єднані по функції «АЛЕ» |
|  | Мультиплексор 4х2 | Скидання |
|  | Мультиплексор 3х1 | Скидання |
|  | Мультиплексор 6х2 | Скидання |

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.