Лабораторная работа № 3.
Исследование комбинационного цифрового устройства.

Цель работы: получение основных навыков проектирования схем в редакторе пакета **Quartus15**. Изучение функционирования коммутационных схем: мультиплексора и демультиплексора. Моделирование схемы КЦУ, состоящего из кодопреобразователя и демультиплексора.

Задание на работу в лаборатории.

Часть 1. Программное проектирование коммутационных схем.

1. Создать проект. Открыть VerilogHDL файл и записать **программу 3.1**, отражающую функционирование мультиплексора на 2 адресных входа. Сохранить файл с названием **ms**, установить его старшим в иерархии и откомпилировать.

2. Пользуясь «Приложением 2», получить диаграммы при **интервалах импульса на адресной шине – 20ns, на шине данных в течение первых 50ns записать 1011, а затем установить на 100ns – 0110.**

Программа 3.1.

```
module  ms
(input wire [3:0] data,
input wire [1:0] adr,
output wire line);
reg out;
assign line=out;
always@(data,adr)
begin
        if(adr==2'b00)
        begin
        out=data[0];
        end
                else if (adr==2'b01)
                begin
                out=data[1];
                end
                        else if (adr==2'b10)
                        begin
                        out=data[2];
                        end
                                else
                                begin
```

```
                    out=data[3];
                    end
end
endmodule
```

3. Открыть новый Verilog HDL файл и записать **программу 3.2**, отражающую функционирование демультиплексора на 2 адресных входа. Сохранить файл под новым именем - **dms**, установить его старшим в иерархии и откомпилировать.

4. Получить временные диаграммы для устройства, задав параметры для **данных - 20нс, адреса – 80нс.**

Программа 3.2

```
module  dms
(output wire [3:0] data,
input wire [1:0] adr,
input wire line);
reg [3:0]out;
assign data=out;
always@(line,adr)
begin
        if(adr==2'b00)
        begin
        out[0]=line;
        out[1]=1'bz;
        out[2]=1'bz;
        out[3]=1'bz;
        end
                else if (adr==2'b01)
                begin
                out[1]=line;
                out[0]=1'bz;
                out[2]=1'bz;
                out[3]=1'bz;
                end
                        else if (adr==2'b10)
                        begin
                        out[2]=line;
                        out[1]=1'bz;
                        out[0]=1'bz;
                        out[3]=1'bz;
                        end
                                else
```

```
                    begin
                    out[3]=line;
                    out[1]=1'bz;
                    out[2]=1'bz;
                    out[0]=1'bz;
                    end
end
endmodule
```

Часть 2. Моделирование КЦУ, состоящего из 2-х узлов.

5. Создать проект под названием **seg_4**. Открыть Verilog HDL файл и записать **программу 3.3**, отражающую функционирование преобразователя кода четырехразрядного двоичного числа в соответствующий ему символ на семисегментном индикаторе. Сохранить файл с названием **coder**, установить его старшим в иерархии и откомпилировать.

6. Пользуясь «Приложением 2», получить диаграммы при **интервалах импульса на входной шине – 20нс. Поразрядно шину не разворачивать!**

Программа 3.3.

```
module  coder
(input wire [3:0] data,
output wire [6:0] seg);
reg [6:0]code;
assign seg=code;
always @*
        case(data)
        4'b0000: code = 7'b1000000;
        4'b0001: code = 7'b1111001;
        4'b0010: code = 7'b0100100;
        4'b0011: code = 7'b0110000;
        4'b0100: code = 7'b0011001;
        4'b0101: code = 7'b0010010;
        4'b0110: code = 7'b0000010;
        4'b0111: code = 7'b1111000;
        4'b1000: code = 7'b0000000;
        4'b1001: code = 7'b0010000;
        4'b1010: code = 7'b0001000;
        4'b1011: code = 7'b0000011;
        4'b1100: code = 7'b1000110;
        4'b1101: code = 7'b0100001;
```

```
        4'b1110: code = 7'b0000110;
        4'b1111: code = 7'b0001110;
        endcase
endmodule
```

7. Открыть новый Verilog HDL файл и записать **программу 3.4**, отражающую функционирование объединенного демультиплексора для 7 (семи) входных линий. Сохранить файл под именем **dms7_4,** установить его старшим в иерархии и откомпилировать.
8. Получить временные диаграммы для устройства, задав параметры для **символов 20нс, адреса – 80нс.**

Программа 3.4

```
module  dms7_4
(output wire [6:0] data0,
output wire [6:0] data1,
output wire [6:0] data2,
output wire [6:0] data3,
input wire [1:0]adr,
input wire [6:0]line);
reg [6:0]out0;
reg [6:0]out1;
reg [6:0]out2;
reg [6:0]out3;
assign data0=out0;
assign data1=out1;
assign data2=out2;
assign data3=out3;
always@(line,adr)
begin
        if(adr==2'b00)
        begin
        out0=line;
        out1=7'bzzzzzzz;
        out2=7'bzzzzzzz;
        out3=7'bzzzzzzz;
        end
                else if (adr==2'b01)
                begin
                out1=line;
                out0=7'bzzzzzzz;
                out2=7'bzzzzzzz;
                out3=7'bzzzzzzz;
                end
```

```
                else if (adr==2'b10)
                begin
                out2=line;
                out1=7'bzzzzzzz;
                out0=7'bzzzzzzz;
                out3=7'bzzzzzzz;
                end
                        else
                        begin
                        out3=line;
                        out1=7'bzzzzzzz;
                        out2=7'bzzzzzzz;
                        out0=7'bzzzzzzz;
                        end
end
endmodule
```

9. Открыть новый Verilog HDL файл и записать **программу 3.5**, отражающую функционирование кодопреобразователя с демультиплексором на выходе. Сохранить файл под новым именем – **seg_4**, установить его старшим в иерархии и откомпилировать.
10. Получить временные диаграммы для устройства, задав параметры для **символов 20нс, адреса – 80нс.**

Программа 3.5

```
module  seg_4
(input wire [3:0]tmb,
input wire [1:0]adrss,
output wire [6:0]hex0,
output wire [6:0]hex1,
output wire [6:0]hex2,
output wire [6:0]hex3);
wire [6:0]bs;
coder  ccc(.data(tmb), .seg(bs));
dms7_4  ddd(.line(bs), .adr(adrss), .data0(hex0), .data1(hex1), .data2(hex2),
.data3(hex3));
endmodule
```

11. Пользуясь «Приложением 3» произвести разводку выводов схемы для работы в макете таким образом, чтобы ввод **числа** осуществлялся с тумблеров **SW9,SW8,SW7,SW6**(**SW9** – старщий разряд), ввод **адреса** – с тумблеров **SW1,SW0**(**SW1** – старший разряд). Вывод производить на **сегментные индикаторы** с **0-го** по **3-ий. После компиляции файла планировщика еще раз откомпилируйте файл верхнего уровня!**

12. Пользуясь «Приложением 4» произвести программирование кристалла FPGA макета.
13. Проверить работу устройства**. Порядок следующий: устанавливаем адрес, устанавливаем любое число. Убедившись, что вывод производится, устанавливаем новый адрес и повторяем вывод любого числа.**

Продемонстрировать работу преподавателю.

Отчет должен содержать программы функционирования устройств и диаграммы их работы.