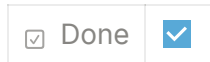


Лабораторная работа 5 (Мультиплексор)



Цель

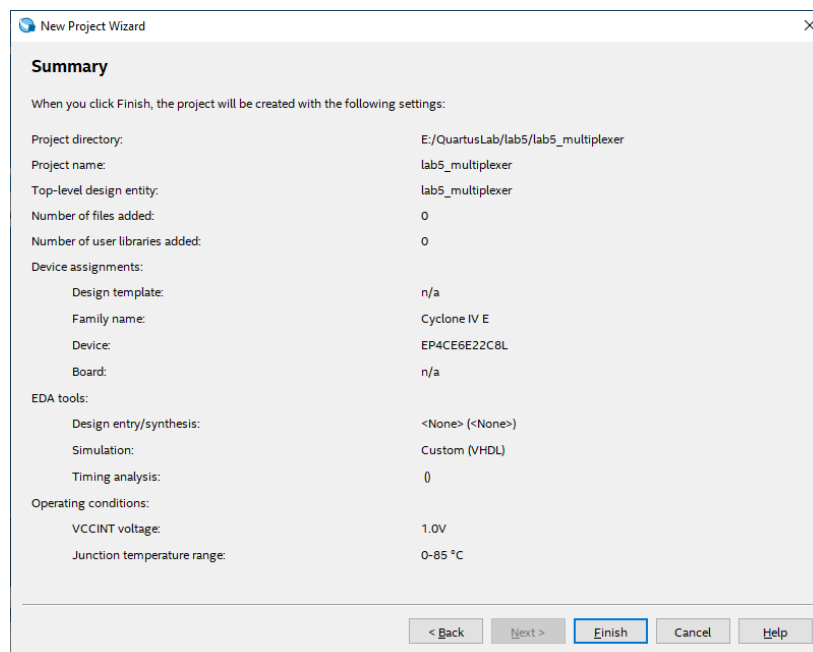
Ознакомиться с примерами мультиплексоров, их схемами и работой. Написать на языках VHDL и SystemVerilog программу для заданного варианта мультиплексора.

Задание

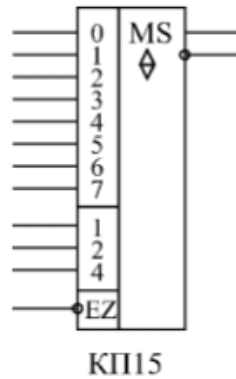
Реализовать на языке VHDL и SystemVerilog мультиплексор, предложенный вариантом.

Выполнение

1. Создаем пустой проект с такими же параметрами как и лабораторная №1



2. Выбираю свой вариант



3. Создаем **Verilog HDL File** , **Block Diagram/Schematic File** , **VHDL File** и в разделе **Verification/Debugging File** выбираем **University Program VwF** и заполняем их кодом

```
module multiplexer_Verilog (
    input X0, X1, X2, X3, X4, X5, X6, X7,
    input A1, A2, A4,
    input EZ,
    output reg Y0, Y1
);

always @(*) begin
    if (EZ == 1) begin
        Y0 = 1'bz;
        Y1 = 1'bz;
    end else begin
        case ({A1, A2, A4})
            3'b000: begin
                Y0 = X0;
                Y1 = ~X0;
            end
            3'b001: begin
                Y0 = X1;
                Y1 = ~X1;
            end
            3'b010: begin
                Y0 = X2;
                Y1 = ~X2;
            end
            3'b011: begin
                Y0 = X3;
            end
        endcase
    end
end
```

```

        Y1 = ~X3;
    end
    3'b100: begin
        Y0 = X4;
        Y1 = ~X4;
    end
    3'b101: begin
        Y0 = X5;
        Y1 = ~X5;
    end
    3'b110: begin
        Y0 = X6;
        Y1 = ~X6;
    end
    3'b111: begin
        Y0 = X7;
        Y1 = ~X7;
    end
    default: begin
        Y0 = 0;
        Y1 = 1;
    end
endcase
end
endmodule

```

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity multiplexer_VHDL is
    Port (
        X0, X1, X2, X3, X4, X5, X6, X7 : in STD_LOGIC;
        A1, A2, A4 : in STD_LOGIC;
        EZ : in STD_LOGIC;
        Y0, Y1 : out STD_LOGIC
    );
end multiplexer_VHDL;

architecture Behavioral of multiplexer_VHDL is

```

```

begin
  process (X0, X1, X2, X3, X4, X5, X6, X7, A1, A2, A4, EZ)
    variable sel : STD_LOGIC_VECTOR(2 downto 0); -- Изменено имя на s
  begin
    sel := A1 & A2 & A4;
    Y0 <= '0';
    Y1 <= '1';

    if EZ = '0' then
      case sel is
        when "000" =>
          Y0 <= X0;
          Y1 <= not X0;
        when "001" =>
          Y0 <= X1;
          Y1 <= not X1;
        when "010" =>
          Y0 <= X2;
          Y1 <= not X2;
        when "011" =>
          Y0 <= X3;
          Y1 <= not X3;
        when "100" =>
          Y0 <= X4;
          Y1 <= not X4;
        when "101" =>
          Y0 <= X5;
          Y1 <= not X5;
        when "110" =>
          Y0 <= X6;
          Y1 <= not X6;
        when "111" =>
          Y0 <= X7;
          Y1 <= not X7;
        when others =>
          Y0 <= '0';
          Y1 <= '1';
      end case;
    else
      Y0 <= 'Z';
      Y1 <= 'Z';
    end if;
  end process;
end;

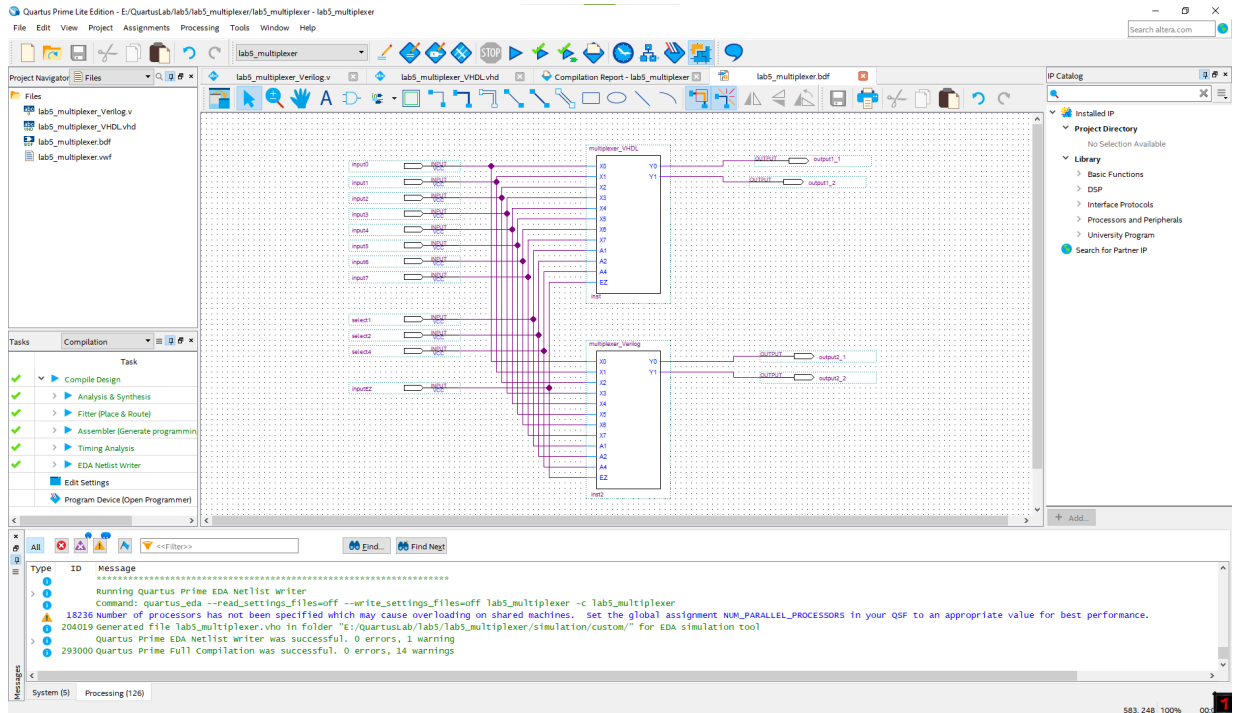
```

```

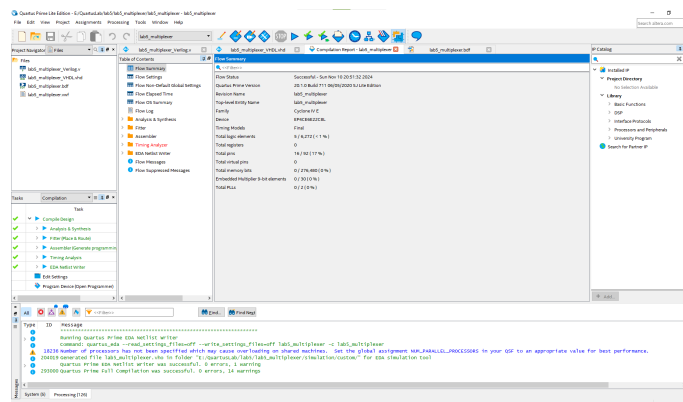
end if;
end process;
end Behavioral;

```

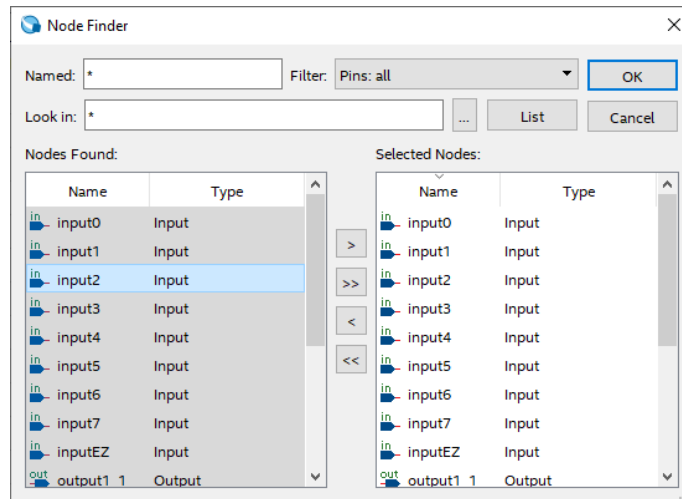
4. Компилируем их и добавляем на Block файл



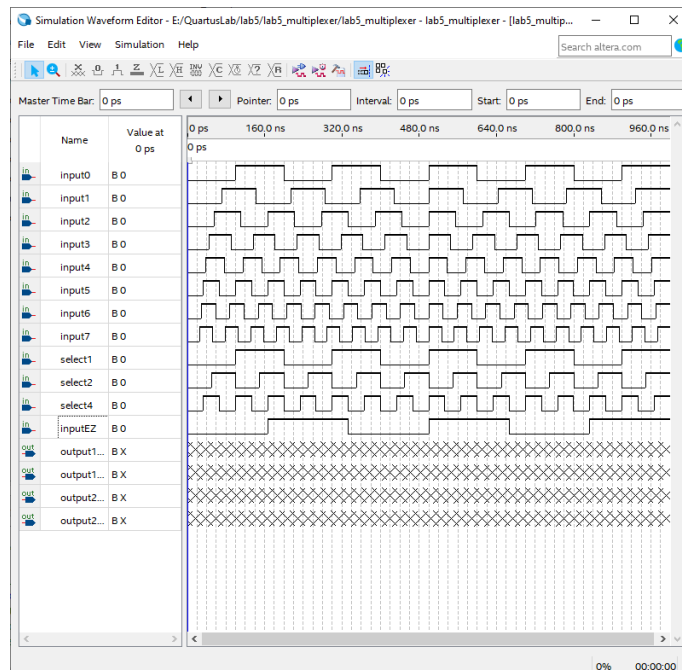
5. Отправляем нашу схему на верхний уровень и запускаем компиляцию проекта, дожидаясь успешного завершения.



6. На странице добавления узлов в модуляцию ищем все наши узлы и добавляем их



7. Для каждого узла выставляем разную частоту от 1 до 15 MHz



8. Удаляем `-novopt` из параметров модуляции

Simulation Options

Caution: Improperly modifying these settings can cause the simulation to fail

HDL Language: ☒ Verilog ☐ VHDL (The language used for the testbench and netlist)

Functional Simulation Settings Timing Simulation Settings

Testbench Generation Command (Functional Simulation):

```
quartus_eda --gen_testbench --tool=modelsim_oem --format=verilog --write_settings_files=off lab3_decoder
```

Netlist Generation Command (Functional Simulation):

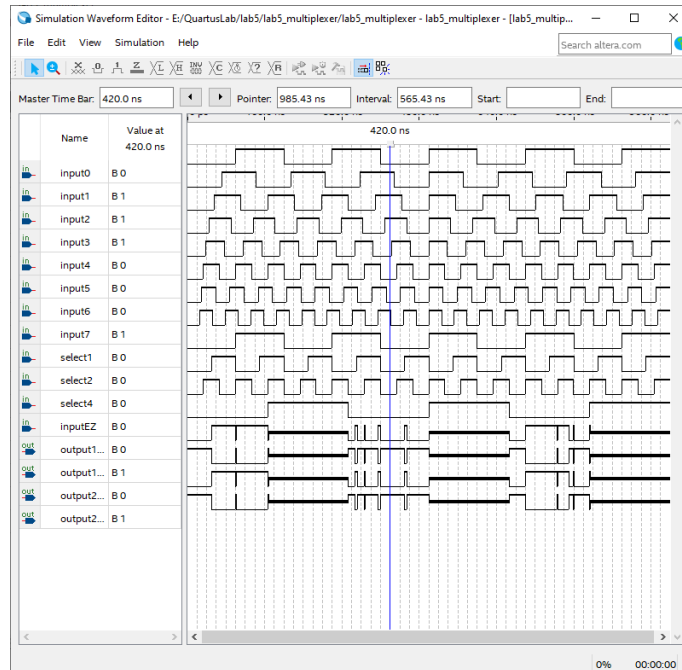
```
quartus_eda --write_settings_files=off --simulation --functional=on --flatten_buses=off --tool=modelsim_oem
```

ModelSim Script (Functional Simulation):

```
onerror {exit -code 1}
vlib work
vlog -work work lab3_decoder.vo
vlog -work work Waveform.vwf.vt
vsim -c -t 1ps -L cycloneive_ver -L altera_ver -L altera_mf_ver -L 220model_ver -L sgate_ver -L altera_insim_ver
vcd file -direction lab3_decoder.msim.vcd
vcd add -internal Schem1_1_vlg_vec_tst/*
vcd add -internal Schem1_1_vlg_vec_tst/i1/*
proc simTimestamp {} {
    echo "Simulation time: $::now ps"
    if { [string equal running [runStatus]] } {
        after 2500 simTimestamp
    }
}
after 2500 simTimestamp
run -all
quit -f
```

Restore Defaults Save Cancel

9. Проверяем результат



Сверяем с таблицей истинности

EZ	A1	A2	A4	Y0	Y1
1	X	X	X	Z	Z
0	0	0	0	X0	~X0
0	0	0	1	X1	~X1
0	0	1	0	X2	~X2
0	0	1	1	X3	~X3
0	1	0	0	X4	~X4
0	1	0	1	X5	~X5
0	1	1	0	X6	~X6
0	1	1	1	X7	~X7

Вывод

В ходе данной работы мы познакомились с построением мультиплексора, а также запрограммировали заданный мультиплексор на языках VHDL и SystemVerilog и проверили работу нашего кода с помощью составления схемы и запуска симуляции работы.