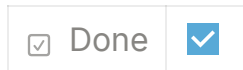


Лабораторная работа 3 (Дешифратор)



Цель

Ознакомиться с примерами дешифраторов. Написать на языках VHDL и SystemVerilog программу для заданного варианта дешифратора.

Задание

Реализовать на языке VHDL и SystemVerilog дешифратор, предложенный вариантом.

Выполнение

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

New Project Wizard

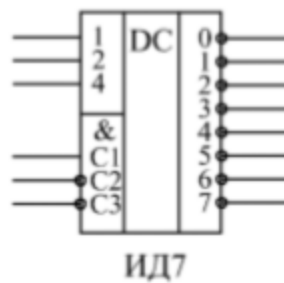
Summary

When you click Finish, the project will be created with the following settings:

| | |
|---------------------------------|---------------------------------|
| Project directory: | E:/QuartusLab/lab3/lab3_decoder |
| Project name: | lab3_decoder |
| Top-level design entity: | lab3_decoder |
| Number of files added: | 0 |
| Number of user libraries added: | 0 |
| Device assignments: | |
| Design template: | n/a |
| Family name: | Cyclone IV E |
| Device: | EP4CE6E22C8L |
| Board: | n/a |
| EDA tools: | |
| Design entry/synthesis: | <None> (<None>) |
| Simulation: | Custom (VHDL) |
| Timing analysis: | 0 |
| Operating conditions: | |
| VCCINT voltage: | 1.0V |
| Junction temperature range: | 0-85 °C |

< Back Next > **Finish** Cancel Help

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

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity decode_VHDL is
    port (
        x1: in STD_LOGIC;
        x2: in STD_LOGIC;
        x4: in STD_LOGIC;
        c1: in STD_LOGIC;
        c2: in STD_LOGIC;
        c3: in STD_LOGIC;
        y0: out STD_LOGIC;
        y1: out STD_LOGIC;
        y2: out STD_LOGIC;
        y3: out STD_LOGIC;
        y4: out STD_LOGIC;
        y5: out STD_LOGIC;
        y6: out STD_LOGIC;
        y7: out STD_LOGIC
    );
end decode_VHDL;

architecture Behavioral of decode_VHDL is
    signal enable: STD_LOGIC;
    signal x_combined: STD_LOGIC_VECTOR(2 downto 0);
begin
    enable <= c1 and not c2 and not c3;
    x_combined <= x1 & x2 & x4;

    process (x_combined, enable)
    begin
        y0 <= '0';
        y1 <= '0';
        y2 <= '0';
        y3 <= '0';
    end process;
end Behavioral;

```

```

y4 <= '0';
y5 <= '0';
y6 <= '0';
y7 <= '0';

if enable = '1' then
    case x_combined is
        when "000" => y0 <= '1';
        when "001" => y1 <= '1';
        when "010" => y2 <= '1';
        when "011" => y3 <= '1';
        when "100" => y4 <= '1';
        when "101" => y5 <= '1';
        when "110" => y6 <= '1';
        when "111" => y7 <= '1';
    end case;
end if;
end process;
end Behavioral;

```

```

module decoder_Verilog (
    input x1, x2, x4,
    input c1, c2, c3,
    output reg y0, y1, y2, y3, y4, y5, y6, y7);

    assign enable = (c1 & ~c2 & ~c3);
    always @(*) begin
        // Сбрасываем выходы по умолчанию
        y0 = 0;
        y1 = 0;
        y2 = 0;
        y3 = 0;
        y4 = 0;
        y5 = 0;
        y6 = 0;

```

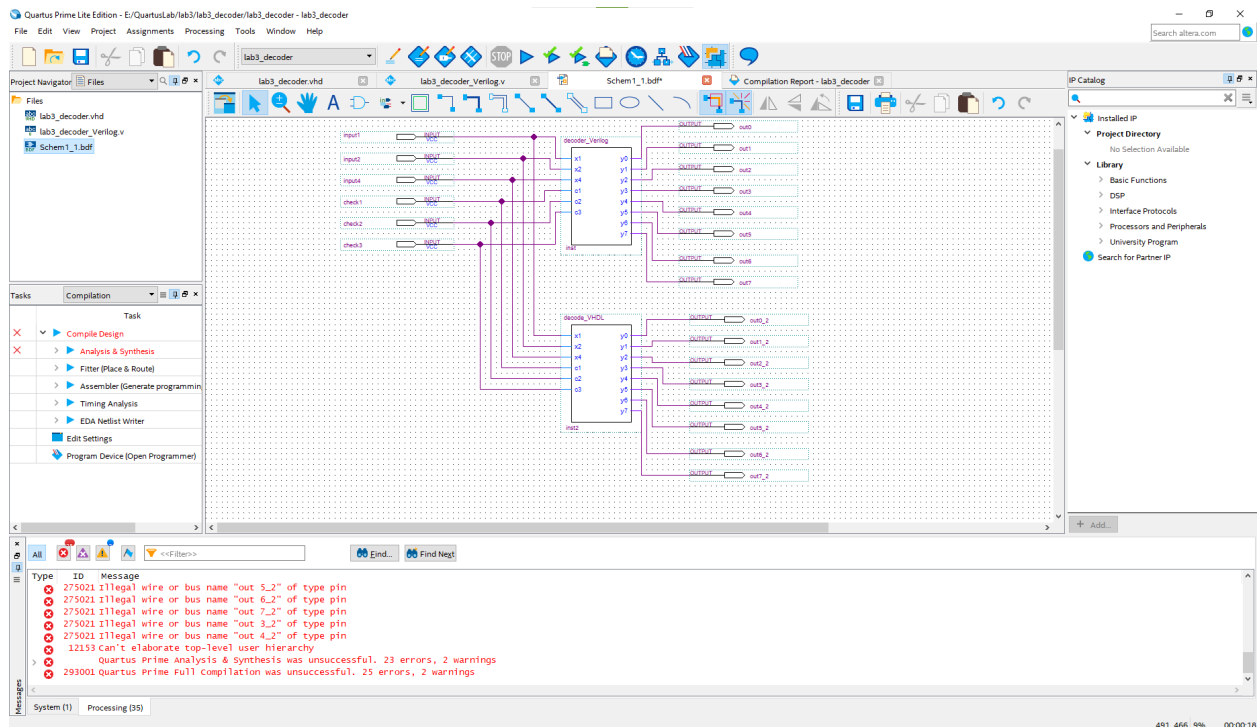
```

y7 = 0;

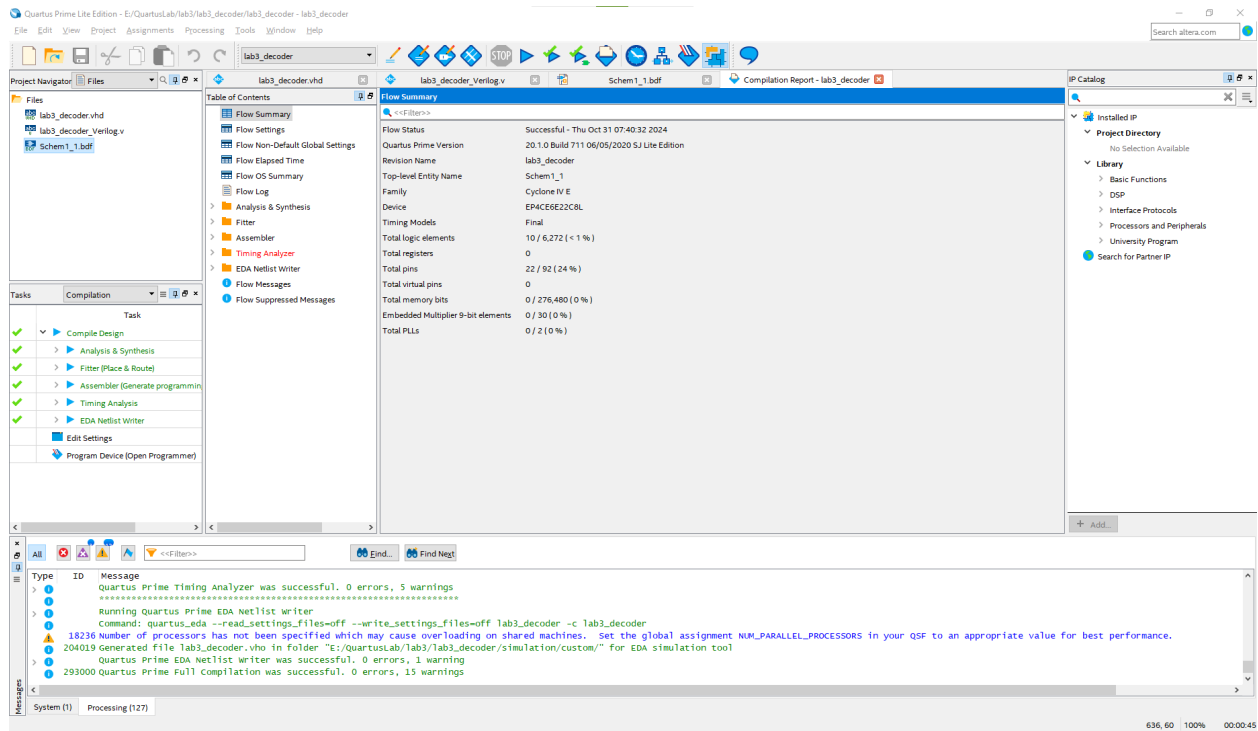
// Проверка enable и выбор выхода на основе входов
if (enable) begin
    case ({x1, x2, x4})
        3'b000: y0 = 1;
        3'b001: y1 = 1;
        3'b010: y2 = 1;
        3'b011: y3 = 1;
        3'b100: y4 = 1;
        3'b101: y5 = 1;
        3'b110: y6 = 1;
        3'b111: y7 = 1;
    endcase
end
end
endmodule

```

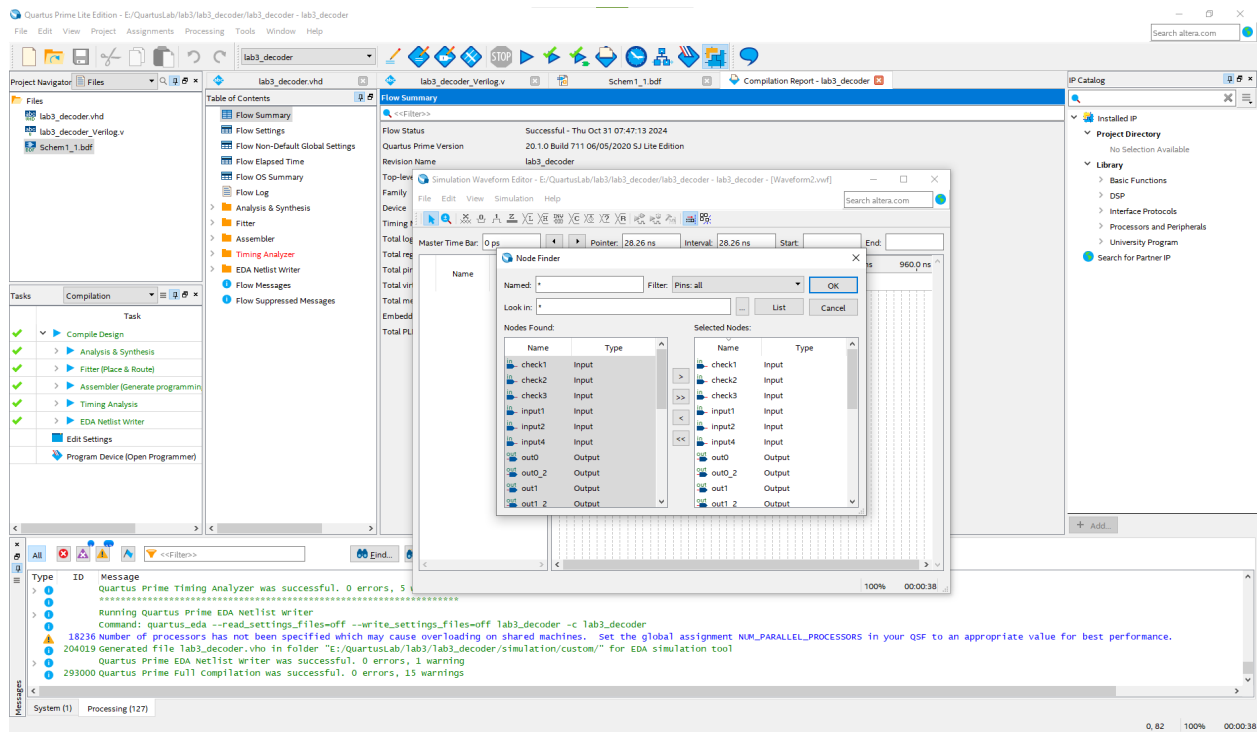
3. Компилируем их и добавляем на Block файл



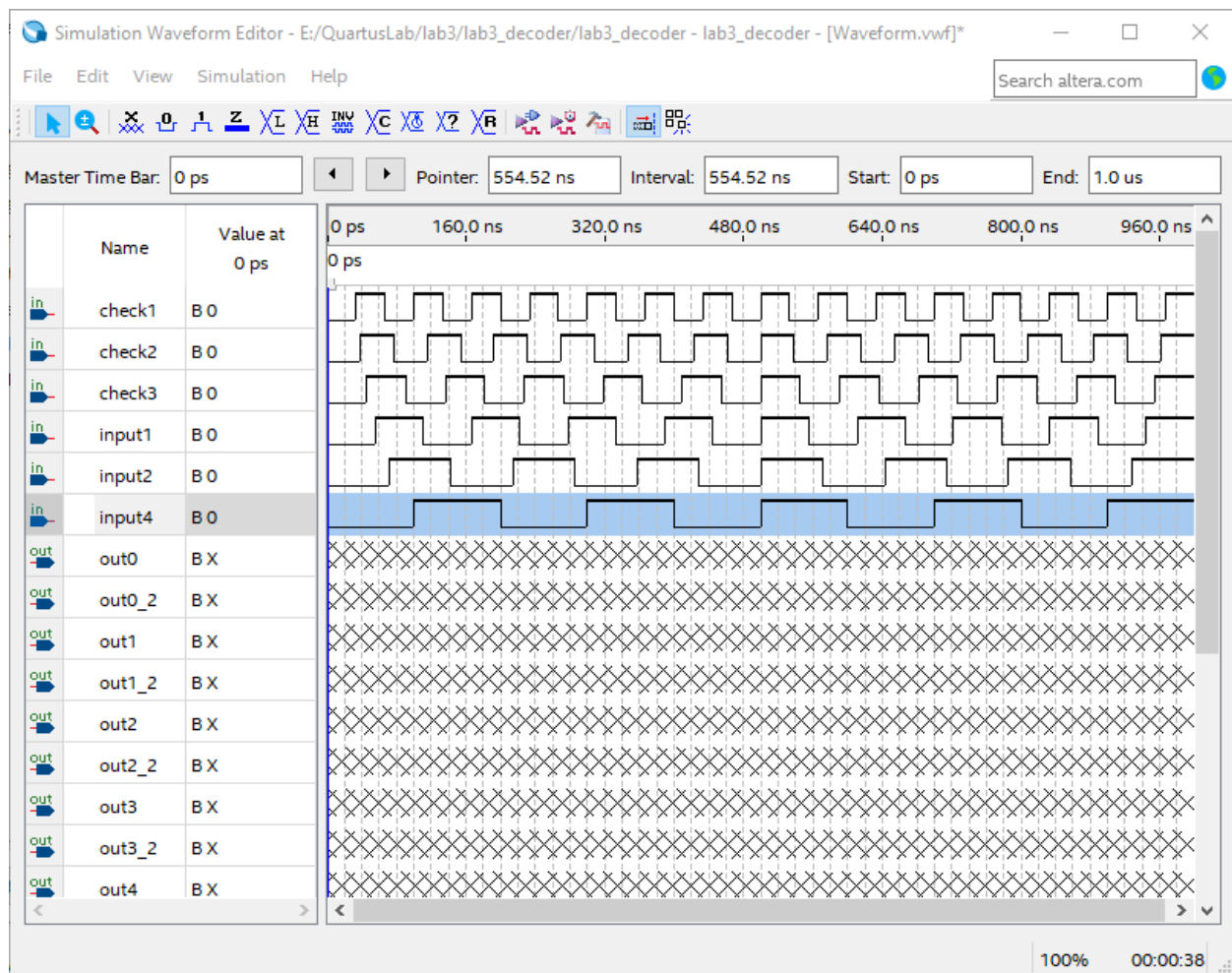
4. Отправляем нашу схему на верхний уровень и запускаем компиляцию проекта, дожидаясь успешного завершения.



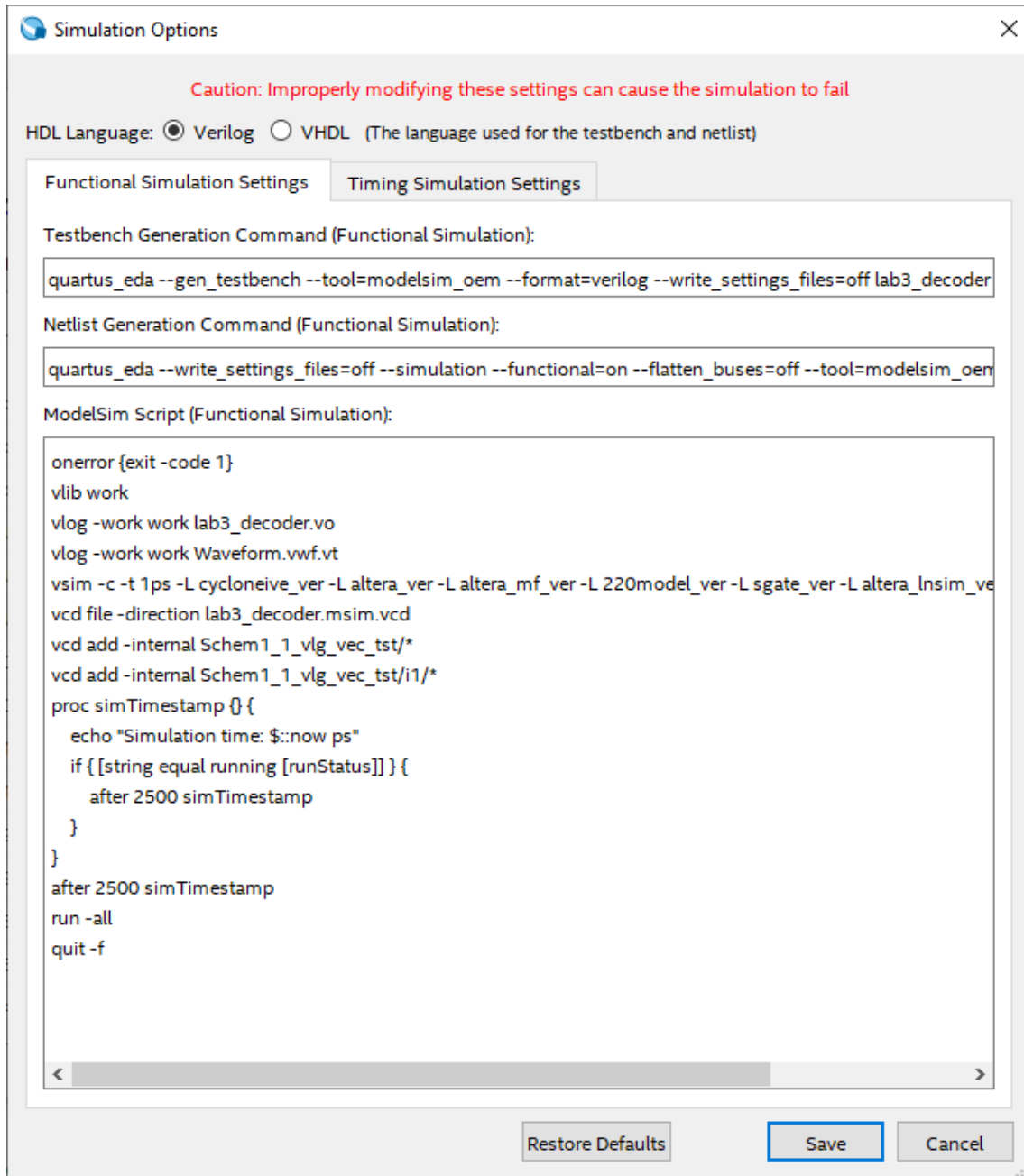
5. На странице добавления узлов в модуляцию ищем все наши узлы и добавляем их



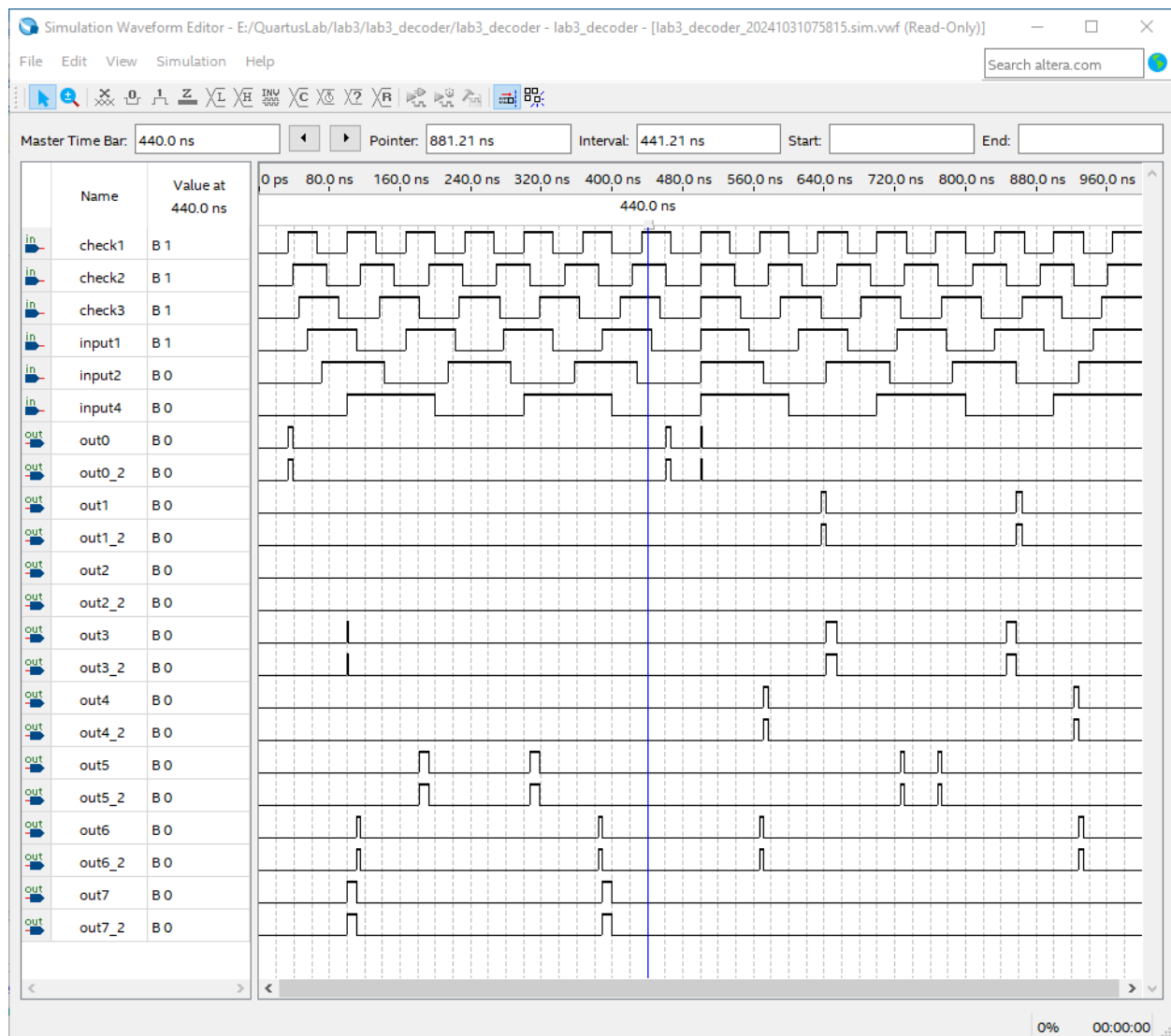
6. Для каждого узла выставляем разную частоту от 15 до 5 MHz



7. Убираем **novopt** из параметра оптимизации и запускаем симуляцию



8. Проверяем результат



Вывод

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