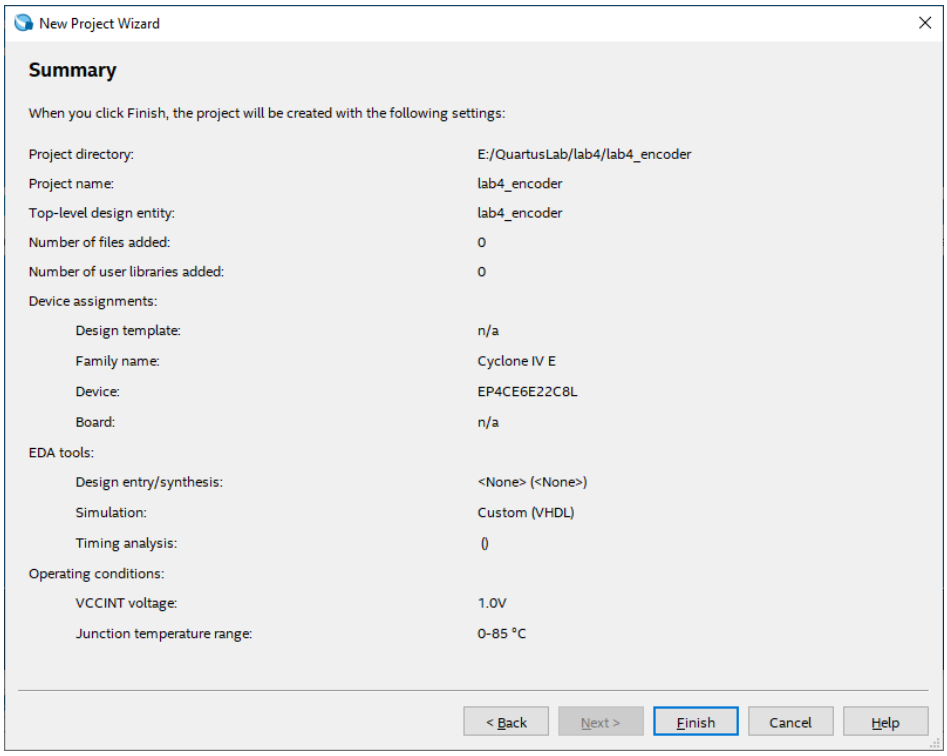
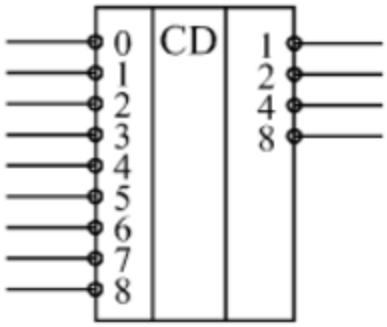
Лабораторная работа 4 (Шифратор)

Done ☑

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



2. Выбираю свой вариант



ИВ3

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

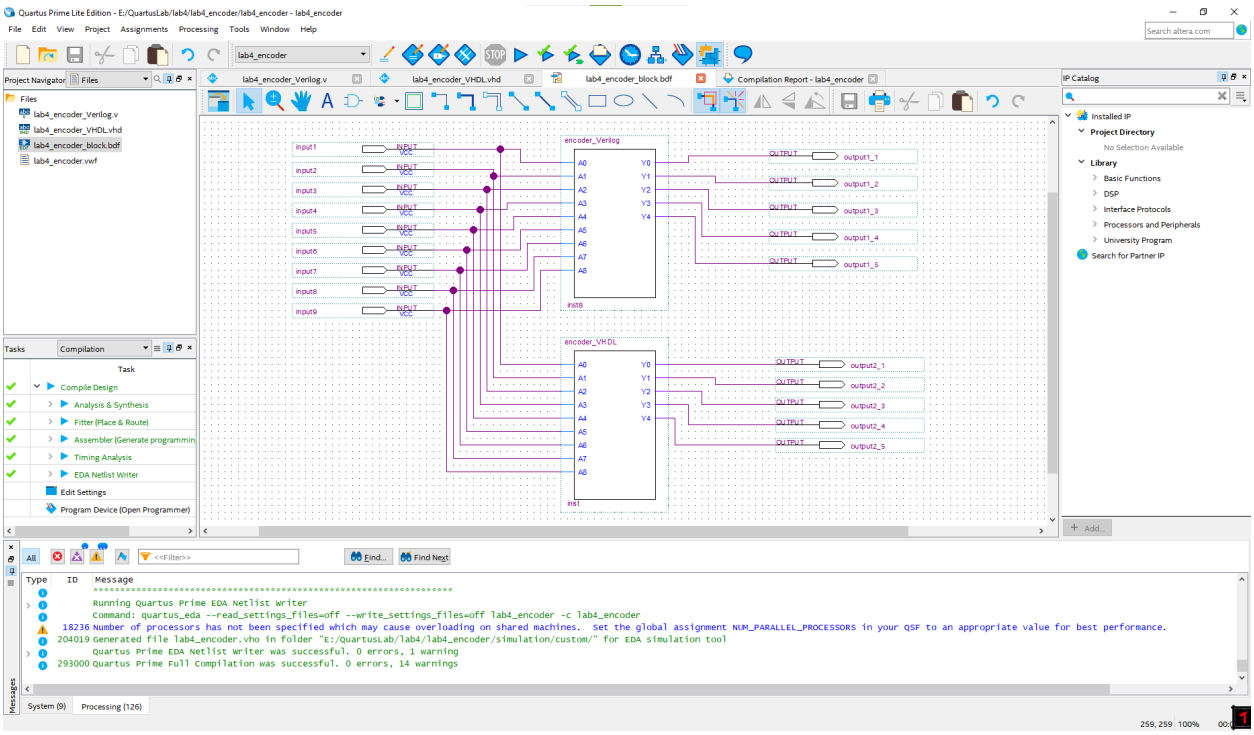
```
module encoder_Verilog (
    input A0, A1, A2, A3, A4, A5, A6, A7, A8,
    output reg Y0, Y1, Y2, Y3, Y4
);
    always @(*) begin
        Y0 = 1'b0;
        Y1 = 1'b0;
        Y2 = 1'b0;
        Y3 = 1'b0;
        Y4 = 1'b0;

        if (~A0) begin
            {Y4, Y3, Y2, Y1, Y0} = 5'b00000;
        end
        else if (~A1) begin
            {Y4, Y3, Y2, Y1, Y0} = 5'b00001;
        end
        else if (~A2) begin
            {Y4, Y3, Y2, Y1, Y0} = 5'b00010;
        end
        else if (~A3) begin
            {Y4, Y3, Y2, Y1, Y0} = 5'b00011;
```

```
        end
        else if (~A4) begin
            {Y4, Y3, Y2, Y1, Y0} = 5'b00100;
        end
        else if (~A5) begin
            {Y4, Y3, Y2, Y1, Y0} = 5'b00101;
        end
        else if (~A6) begin
            {Y4, Y3, Y2, Y1, Y0} = 5'b00110;
        end
        else if (~A7) begin
            {Y4, Y3, Y2, Y1, Y0} = 5'b00111;
        end
        else if (~A8) begin
            {Y4, Y3, Y2, Y1, Y0} = 5'b01000;
        end
        else begin
            {Y4, Y3, Y2, Y1, Y0} = 5'b10000;
        end
    end
endmodule
```

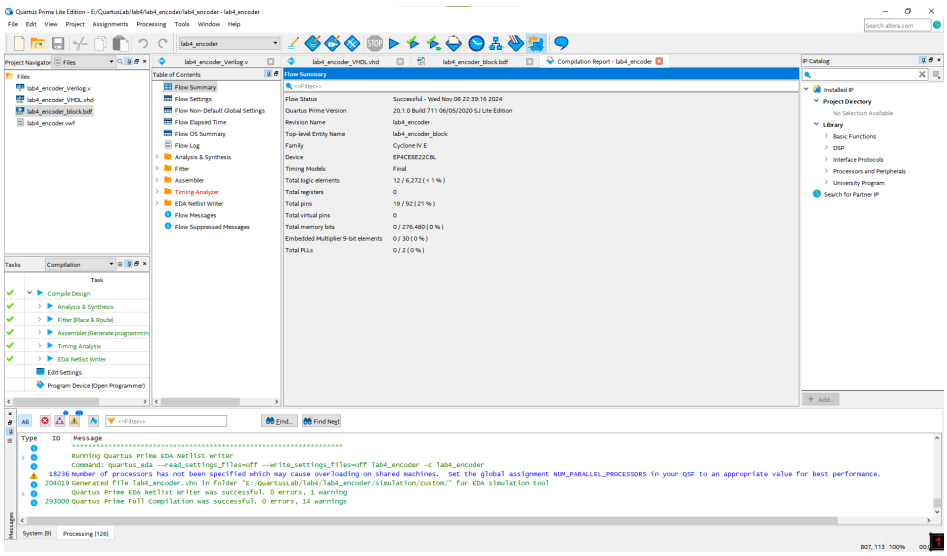
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity encoder_VHDL is
    Port (
        A0, A1, A2, A3, A4, A5, A6, A7, A8 : in STD_LOGIC;
        Y0, Y1, Y2, Y3, Y4 : out STD_LOGIC
    );
end encoder_VHDL;

architecture Behavioral of encoder_VHDL is
begin
    process (A0, A1, A2, A3, A4, A5, A6, A7, A8)
```

```
    begin
        Y0 <= '0';
        Y1 <= '0';
        Y2 <= '0';
        Y3 <= '0';
        Y4 <= '0';

        if (not A0 = '1') then
            Y0 <= '0'; Y1 <= '0'; Y2 <= '0'; Y3 <= '0'; Y4 <= '(
        elsif (not A1 = '1') then
            Y0 <= '1'; Y1 <= '0'; Y2 <= '0'; Y3 <= '0'; Y4 <= '(
        elsif (not A2 = '1') then
            Y0 <= '0'; Y1 <= '1'; Y2 <= '0'; Y3 <= '0'; Y4 <= '(
        elsif (not A3 = '1') then
            Y0 <= '1'; Y1 <= '1'; Y2 <= '0'; Y3 <= '0'; Y4 <= '(
        elsif (not A4 = '1') then
            Y0 <= '0'; Y1 <= '0'; Y2 <= '1'; Y3 <= '0'; Y4 <= '(
        elsif (not A5 = '1') then
            Y0 <= '1'; Y1 <= '0'; Y2 <= '1'; Y3 <= '0'; Y4 <= '(
        elsif (not A6 = '1') then
            Y0 <= '0'; Y1 <= '1'; Y2 <= '1'; Y3 <= '0'; Y4 <= '(
        elsif (not A7 = '1') then
            Y0 <= '1'; Y1 <= '1'; Y2 <= '1'; Y3 <= '0'; Y4 <= '(
        elsif (not A8 = '1') then
            Y0 <= '0'; Y1 <= '0'; Y2 <= '0'; Y3 <= '1'; Y4 <= '(
        else
            Y0 <= '0'; Y1 <= '0'; Y2 <= '0'; Y3 <= '0'; Y4 <= '1
        end if;
    end process;
end Behavioral;
```

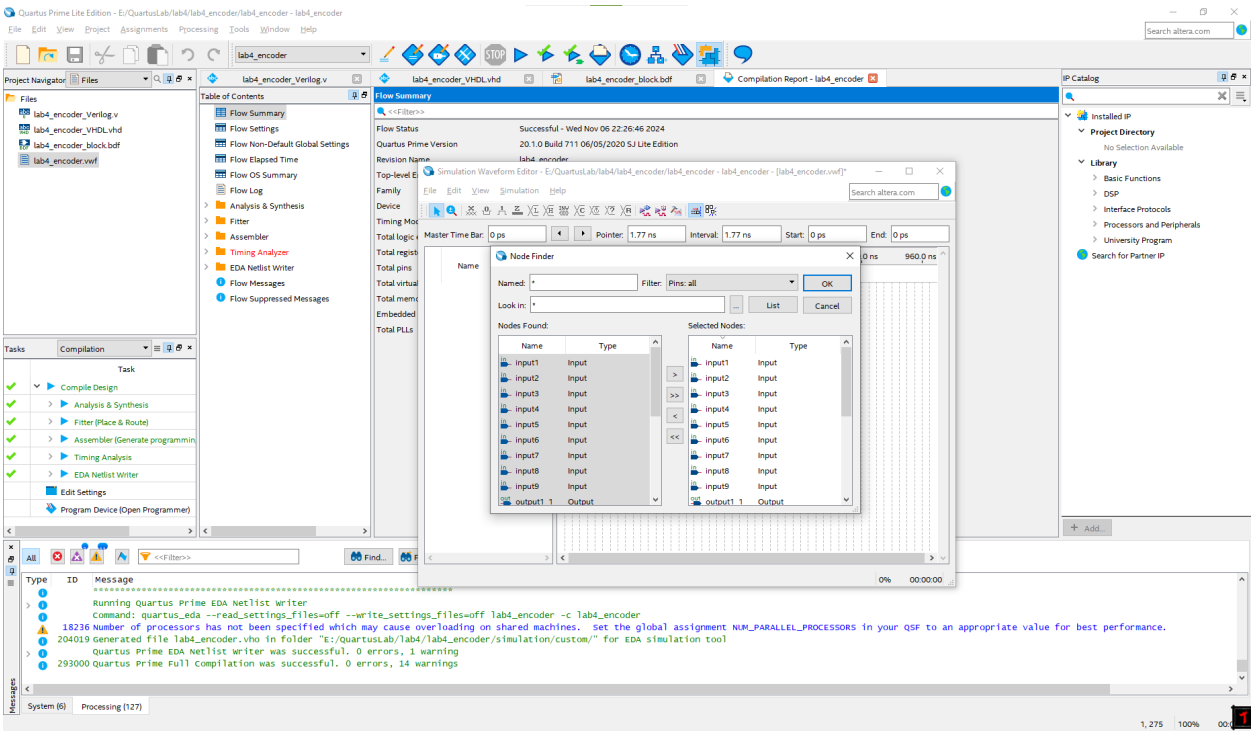
4. Компилируем их и добавляем на Block файл



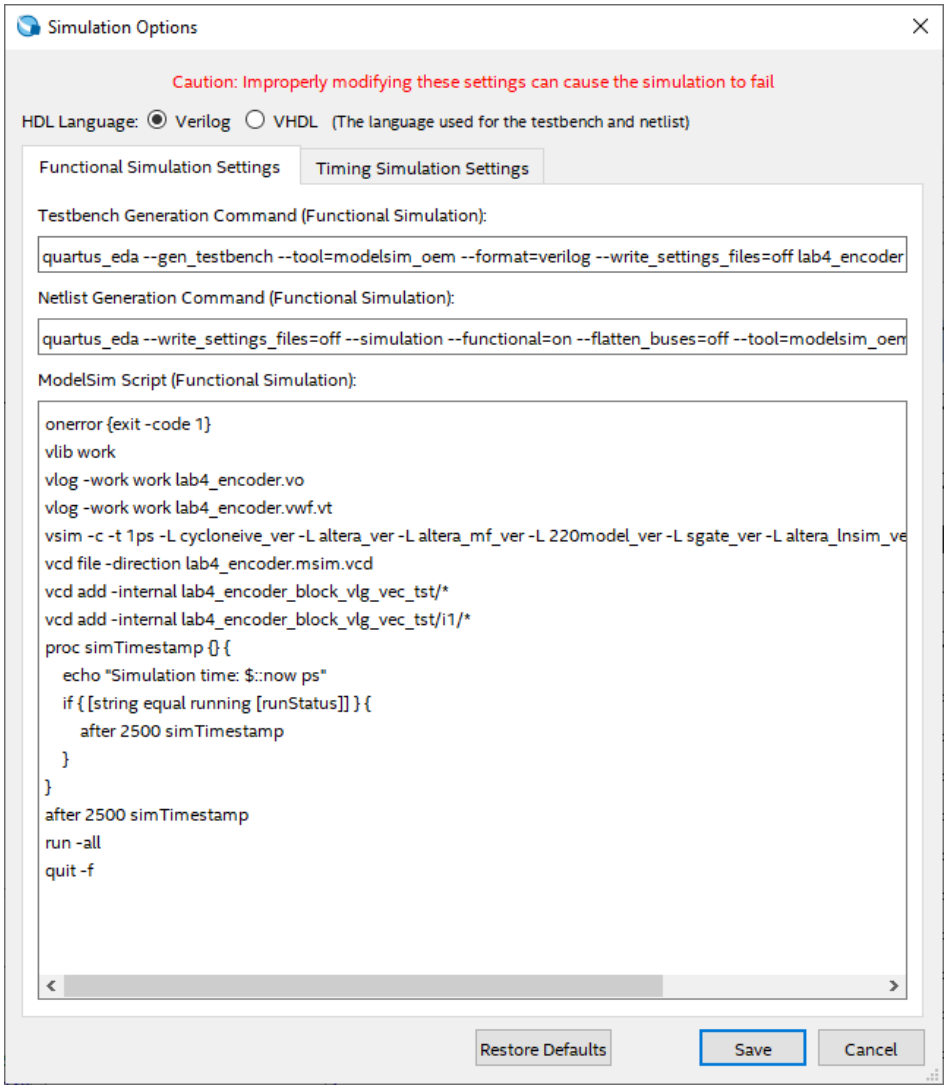
5. Отправляем нашу схему на верхний уровень и запускаем компиляцию проекта, дожидаемся успешного завершения.



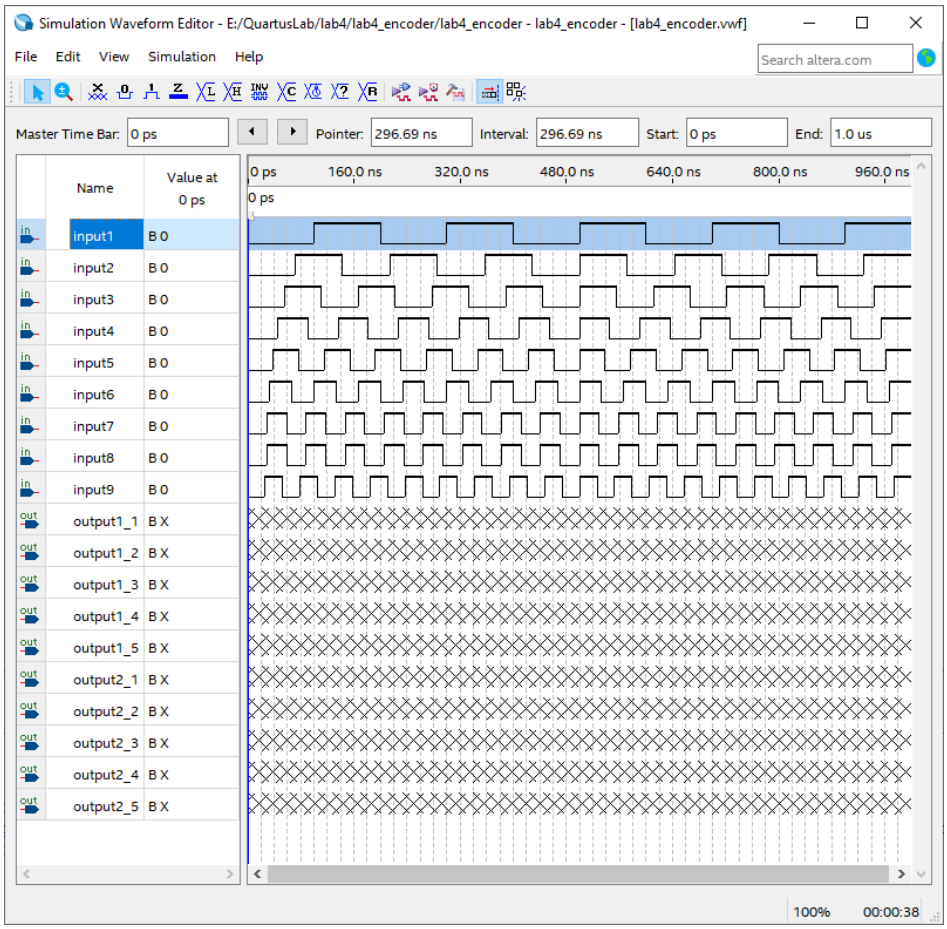
6. На странице добавления узлов в модуляцию ищем все наши узлы и добавляем их



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



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



9. Проверяем результат

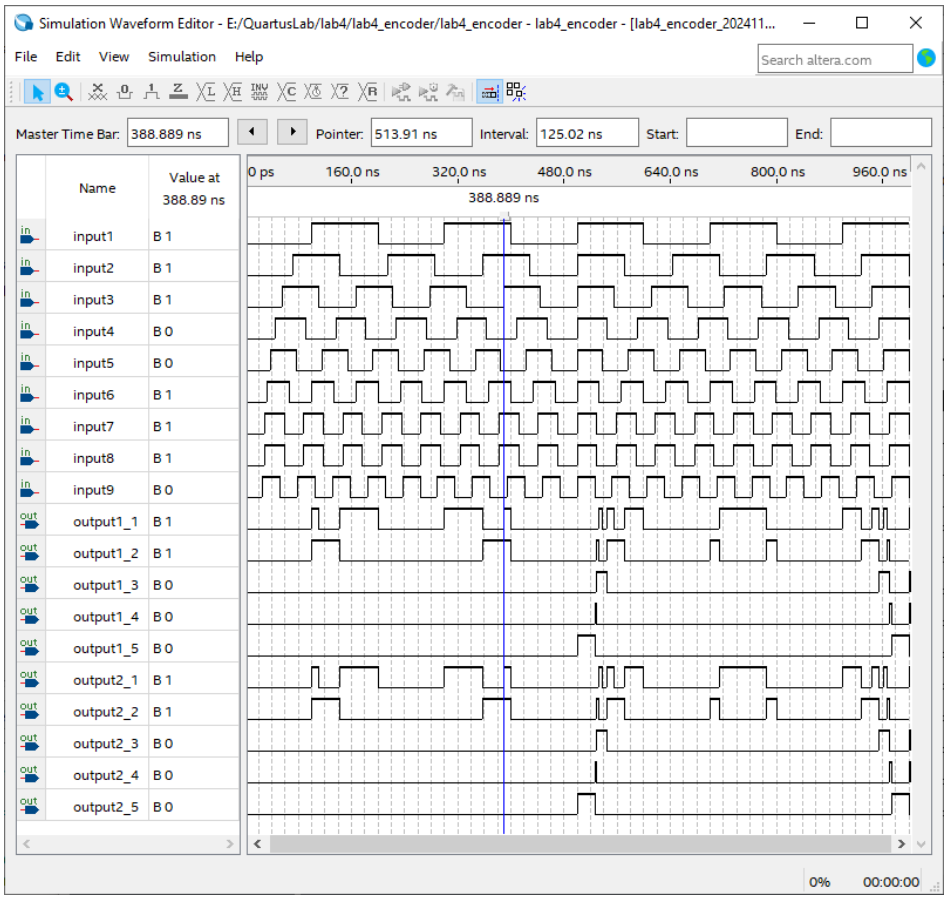


Таблица истинности:

Входы (инвертированные)	Выходы
`A0` = 0, остальные 1	00000
`A1` = 0, остальные 1	00001
`A2` = 0, остальные 1	00010
`A3` = 0, остальные 1	00011
`A4` = 0, остальные 1	00100
`A5` = 0, остальные 1	00101
`A6` = 0, остальные 1	00110
`A7` = 0, остальные 1	00111
`A8` = 0, остальные 1	01000
Все 1 (неактивное состояние)	10000