Лабораторная работа 7 (Сумматор)

Done

Цель

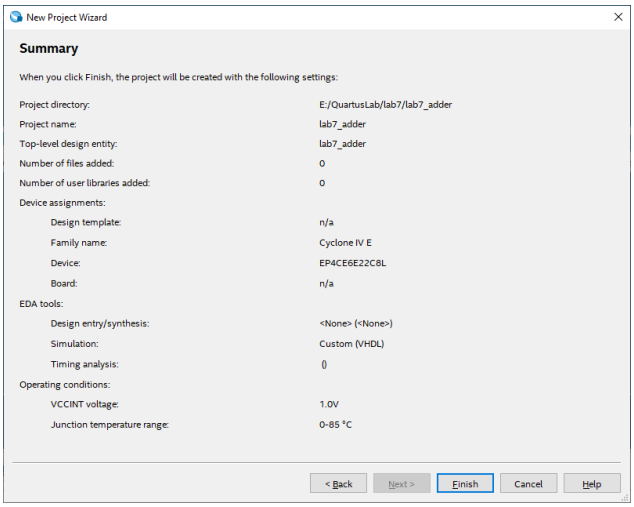
Ознакомиться с примерами сумматоров, их схемами и работой. Написать на языках VHDL и SystemVerylog программу для заданного вариантом сумматора.

Задание

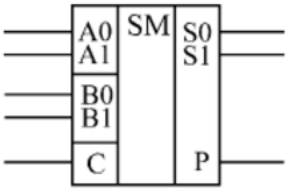
Реализовать на языке VHDL и SystemVerylog сумматор, предложенный вариантом.

Выполнение

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



2. Выбираю свой вариант



ИМ2

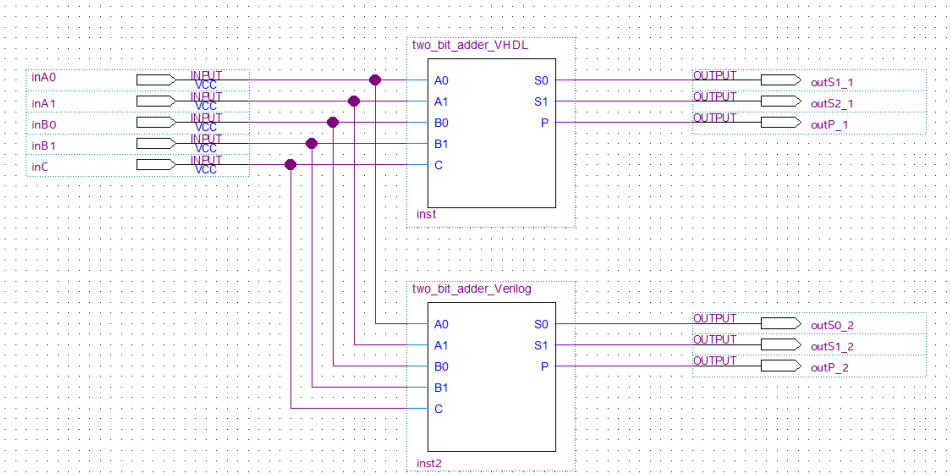
3. Создаем `Verilog HDL File`, `Block Diagram/Schematic File`, `VHDL File` и в разделе Verification/Debugging File выбираем `University Program VWF` и заполняем их кодом

```
module two_bit_adder_Verilog (
    input A0, A1, // Два разряда числа A
    input B0, B1, // Два разряда числа B
    input C,      // Входной перенос
    output S0, S1, // Сумма
    output P       // Перенос
);
```

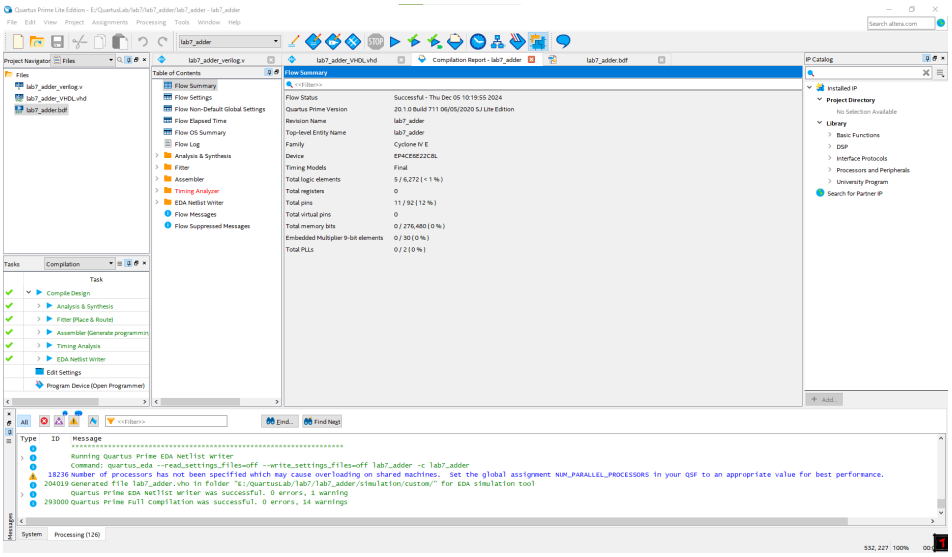
```
    wire carry0, carry1;

    // Первый разряд
    assign S0 = A0 ^ B0 ^ C;
    assign carry0 = (A0 & B0) | (A0 & C) | (B0 & C);

    // Второй разряд
    assign S1 = A1 ^ B1 ^ carry0;
    assign carry1 = (A1 & B1) | (A1 & carry0) | (B1 & carry0);

    // Перенос
    assign P = carry1;
endmodule
```

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

entity two_bit_adder_VHDL is
    Port (
        A0, A1 : in STD_LOGIC; -- Два разряда числа A
        B0, B1 : in STD_LOGIC; -- Два разряда числа B
        C      : in STD_LOGIC; -- Входной перенос
        S0, S1 : out STD_LOGIC; -- Сумма
        P      : out STD_LOGIC  -- Перенос
    );
end two_bit_adder_VHDL ;

architecture Behavioral of two_bit_adder_VHDL is
    signal carry0, carry1: STD_LOGIC;
begin
    -- Первый разряд
    S0 <= A0 XOR B0 XOR C;
    carry0 <= (A0 AND B0) OR (A0 AND C) OR (B0 AND C);

    -- Второй разряд
    S1 <= A1 XOR B1 XOR carry0;
    carry1 <= (A1 AND B1) OR (A1 AND carry0) OR (B1 AND carry0);

    -- Перенос
    P <= carry1;
end Behavioral;
```

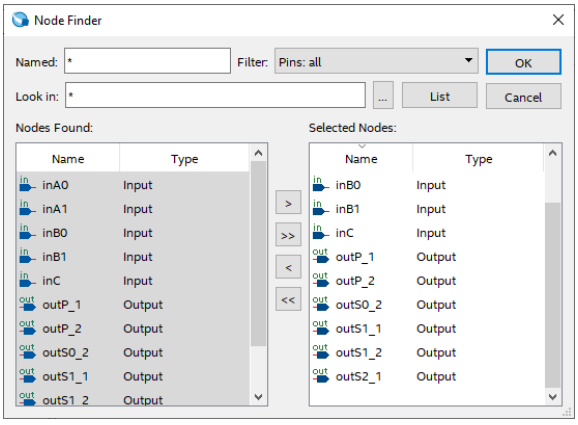
4. Компилируем их и добавляем на Block файл



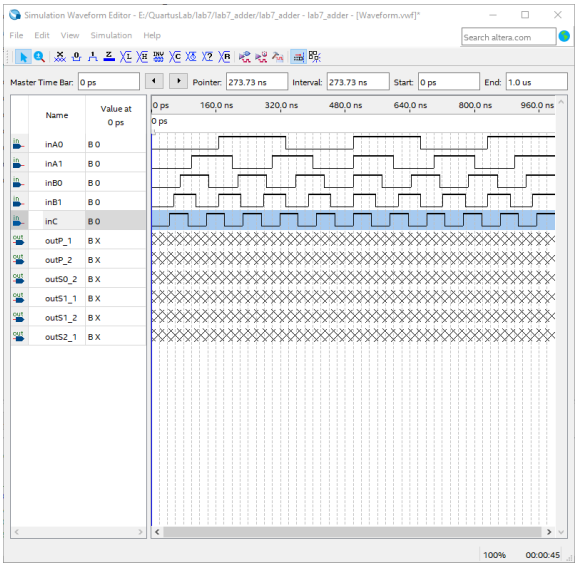
5. Отправляем нашу схему на верхний уровень и запускаем компиляцию проекта, дожидаемся успешного завершения.



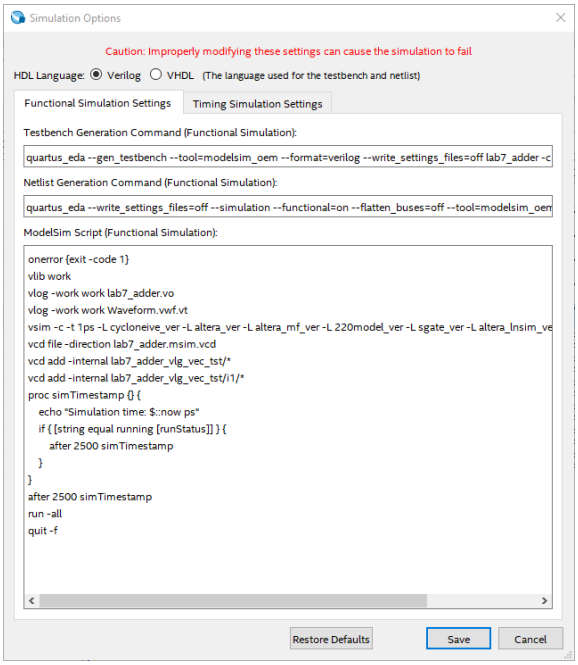
6. На странице добавления узлов в модуляцию ищем все наши узлы и добавляем их



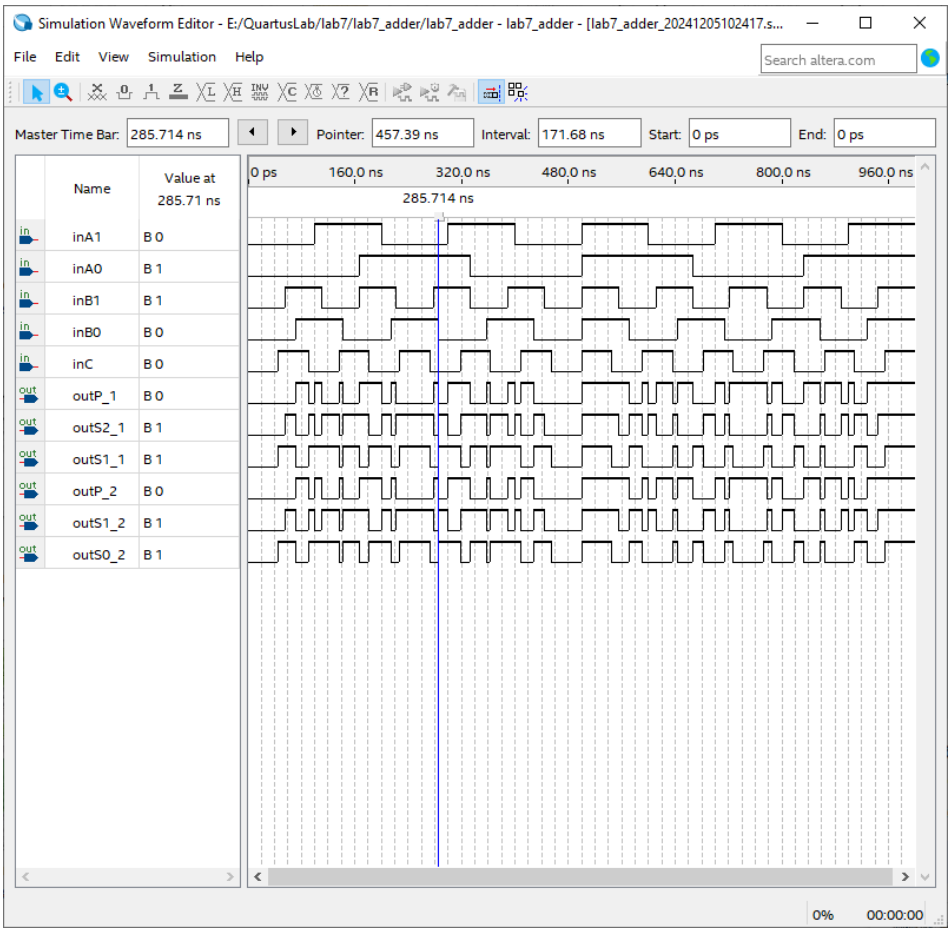
7. Для каждого узла выставляем разную частоту от 3 до 11 MHz



8. Убираем `-novop`t из параметров симуляции



9. Проверяем результат



Входы				Выходы			
				C=0			C=1
A1	A0	B1	B0	P	S1	S0	P
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0
0	0	1	0	0	1	0	0
0	0	1	1	0	1	1	1
0	1	0	0	0	0	1	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	1
1	0	0	0	0	1	0	0
1	0	0	1	0	1	1	1
1	0	1	0	1	0	0	1
1	0	1	1	1	0	1	1
1	1	0	0	0	1	1	1
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1	1	1	1	0	1

Вывод

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