1 Цель работы

Создание проекта на ПЛИС в Quartus II. Создание VHDL файла для описания логических элементов и триггеров на языке VHDL. Проверка работоспособности созданного символа при помощи утилиты Waveform.

2 Задание по лабораторной работе

В соответствии с заданием необходимо описать на языке описания аппаратуры VHDL $\overline{RS}$ – триггер, а также устройство, имеющее следующую логическую функцию:

$$f = \overline{a \wedge b \vee c}$$

3 Выполнение работы

Начнем с реализации устройства, логическая функция которого имеет вид $f = \overline{a \wedge b \vee c}$. Для того, чтобы описать это устройство на языке описания аппаратуры VHDL необходимо составить таблицу истинности данного устройства, которая представлена в таблице 3.1.

Таблица 3.1 – Таблица истинности описываемого устройства

a	b	c	$a \wedge b$	$a \wedge b \vee c$	$\overline{a \wedge b \vee c}$
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	0	1	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	1	0
1	1	1	1	1	0

В соответствии с таблицей 3.1 опишем данное устройство с помощью поведенческого стиля описания на языке описания аппаратуры VHDL. В процессе описания устройства воспользуемся операторами process и if – else. Листинг программы представлен на рисунке 3.1

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  entity logic is
4
5     port(a,b,c : in std_logic;
6          Y : out std_logic);
7
8  end logic;
9
10 architecture beh of logic is
11    begin
12       process (a,b,c)
13          begin
14
15             if (a = '0' and b = '0' and c = '0') then
16                Y <= '1';
17             elsif (a = '0' and b = '0' and c = '1') then
18                Y <= '0';
19             elsif (a = '0' and b = '1' and c = '0') then
20                Y <= '1';
21             elsif (a = '0' and b = '1' and c = '1') then
22                Y <= '0';
23             elsif (a = '1' and b = '0' and c = '0') then
24                Y <= '1';
25             elsif (a = '1' and b = '0' and c = '1') then
26                Y <= '0';
27             elsif (a = '1' and b = '1' and c = '0') then
28                Y <= '0';
29             elsif (a = '1' and b = '1' and c = '1') then
30                Y <= '0';
31             end if;
32       end process;
33 end beh;
```

Рисунок 3.1 – Листинг программы описываемого устройства

В коде программы описываемого устройства мы используем оператор process, в списке чувствительности которого расположены входы a, b и c. Таким образом оператор process начинает свою работу в процессе изменения сигнала на одном из данных входов.

Внутри оператора process, с помощью оператора if – else мы описываем логическую работу устройства в соответствии с таблицей истинности, представленной в таблице 3.1.

Скомпилируем проект, результат успешной компиляции проекта представлен на рисунке 3.2.

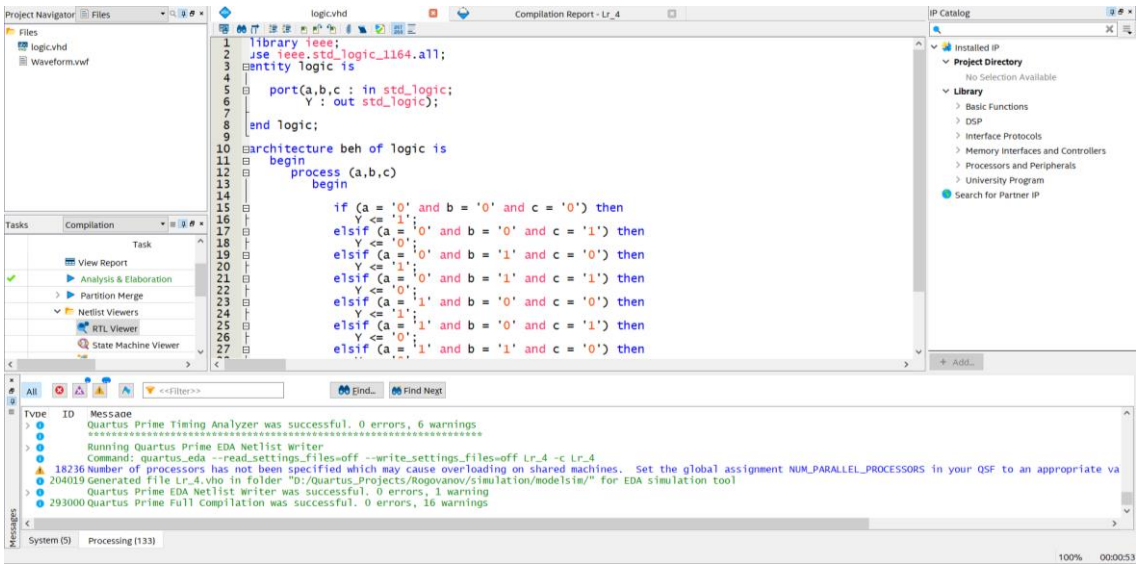

Рисунок 3.2 – Результат успешной компиляции проекта

После успешной компиляции проекта, проанализируем сигналы синтезируемого устройства с помощью утилиты Simulation Waveform. Результаты анализа представлены на рисунке 3.3.

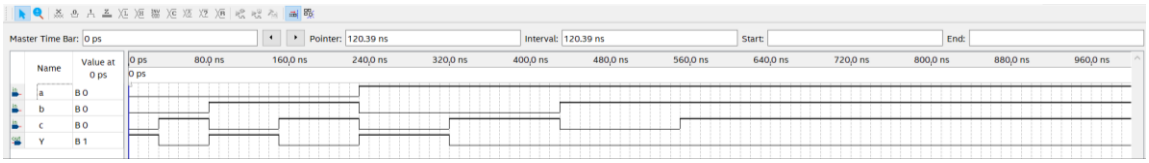

Рисунок 3.3 – Окно Simulation Waveform при анализе описанного устройства

Действительно, как видно из рисунка 3.3, описанное устройство функционирует в соответствии с таблицей истинности, представленной в таблице 3.1.

Также можно описать и в одну строку без использования оператора process, листинг программы представлен на рисунке 3.4.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  entity logic is
4
5     port(a,b,c : in std_logic;
6          Y : out std_logic);
7
8  end logic;
9
10 architecture beh of logic is
11    begin
12    Y <= not ((a and b) or c);
13 end beh;
```

Рисунок 3.4 – Листинг программы описываемого устройства в одну строку

В данной программе, мы, пренебрегая оператором process, просто описываем логику выходного сигнала Y в соответствии с данными входами.

Скомпилируем проект, результат компиляции представлен на рисунке 3.5.

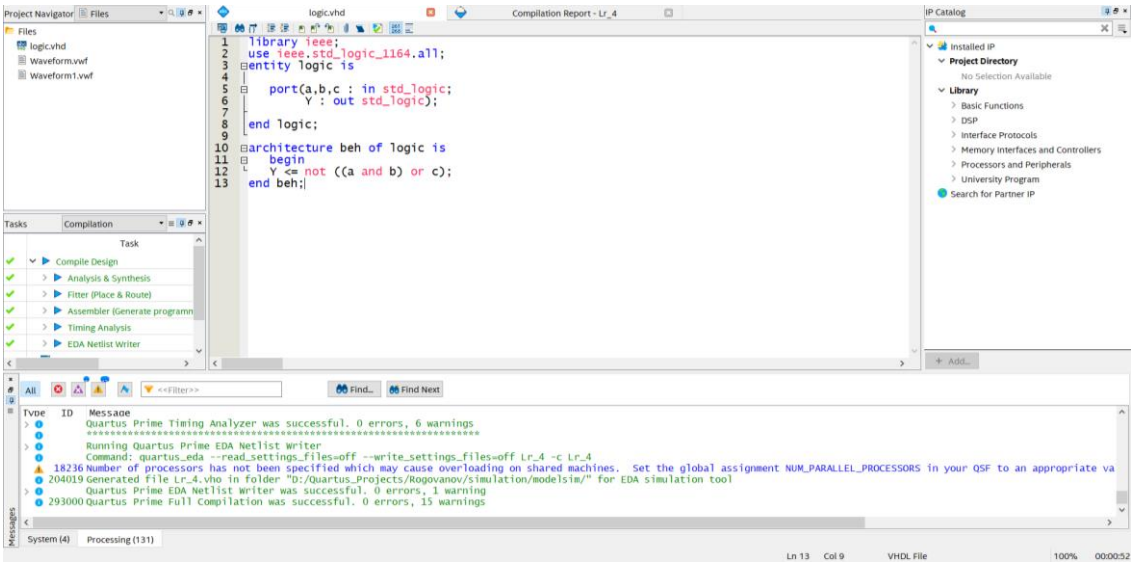

Рисунок 3.5 – Результат успешной компиляции проекта

После успешной компиляции проекта обратимся к встроенной утилите RTL Viewer для того, чтобы увидеть, в какой схеме пакет Quartus синтезировал описанное устройство. Результат анализа в RTL Viewer представлен на рисунке 3.6.

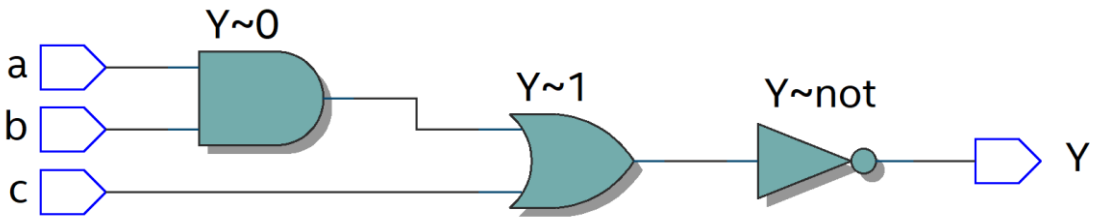

Рисунок 3.6 – Окно утилиты RTL Viewer

Действительно, как мы можем заметить из рисунка 3.6, в процессе синтеза Quartus синтезировал устройство на основе его логической функции. Схема состоит из элемента «И», на вход которого подключены входы a и b, а также элемента «ИЛИ», на вход которого подается выход с элемента «И» и вход c. Перед выходом устройства стоит инвертор.

После успешной компиляции проекта, проанализируем сигналы синтезируемого устройства с помощью утилиты Simulation Waveform. Результаты анализа представлены на рисунке 3.7.

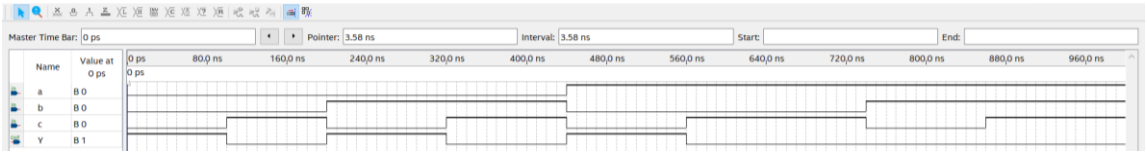

Рисунок 3.7 – Окно Simulation Waveform при анализе описанного устройства в одну строку

Действительно, как можно заметить из рисунка 3.7, временные диаграммы устройство полностью совпадают с временными диаграммами, продемонстрированными на рисунке 3.3, что свидетельствует о работе устройства в соответствии с таблицей истинности.

Следующим этапом выполнения лабораторной работы будет описание $\overline{RS}$ – триггера на языке описания аппаратуры VHDL. Для этого, в первую очередь, составим таблицу истинности $\overline{RS}$ – триггера. Таблица истинности представлена в таблице 3.2.

Таблица 3.2 – Таблица истинности $\overline{RS}$ – триггера

S	R	Q
0	0	$Q-1$
0	1	0
1	0	1
1	1	0

Как можно заметить из таблицы 3.2, при подаче сигнала активного уровня на вход R выход триггера сбрасывается и на выходе «0», в случае подачи сигнала активного уровня на вход S, триггер устанавливается в единицу, случай, когда на оба входа S и R триггера подается сигнал активного уровня, называется запрещенным, в таком случае на выходе триггера будет «0». Когда на входы S и R не подаются сигналы активного уровня, на выходе триггера остаются предыдущие значения триггера. Такое состояние называется режимом хранения.

Теперь, на основании таблицы истинности, представленной в таблице 3.2, опишем устройство на языке VHDL с помощью поведенческого стиля описания с использованием операторов process и if – else. Листинг программы представлен на рисунке 3.8.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity sr_ff is
5
6     port(R, S : in std_logic;
7          Q : buffer std_logic);
8
9  end sr_ff;
10
11 architecture beh of sr_ff is
12    begin
13       process (R,S)
14          begin
15             if (R = '1' and S = '0') then
16                Q <= '0';
17             elsif (R = '0' and S = '1') then
18                Q <= '1';
19             elsif (R = '1' and S = '1') then
20                Q <= '0';
21             elsif (R = '0' and S = '0') then
22                Q <= Q;
23             end if;
24       end process;
25 end beh;
```

Рисунок 3.8 – Листинг программы описанного $\overline{RS}$ – триггера

Из кода программы, представленного на рисунке 3.8, можно заметить, что в архитектуре устройства используется оператор process, в список чувствительности которого внесены входы R и S. В данном случае оператор process будет реагировать на изменение сигналов на данных входах. Внутри оператора процесс используется оператор if – else, с помощью которого описывается логика работы устройства в соответствии с таблицей истинности. Однако в случае, когда триггер переходит в режим хранения, то есть на обоих его входа S и R сигнал отсутствует, то на выходе триггер хранится предыдущее состояние, за этот режим отвечают 21 и 22 строчки кода, представленные на рисунке 3.8. Так как выходу Q присваивается предыдущее значение на выходе Q, то в таком случае микросхеме нужно считать сигнал, хранящийся на выходе Q, в таком случае, в сущности устройства (entity), при объявлении портов, тип порта Q указан не как out, а как buffer.

Скомпилируем проект, результат компиляции представлен на рисунке 3.9.

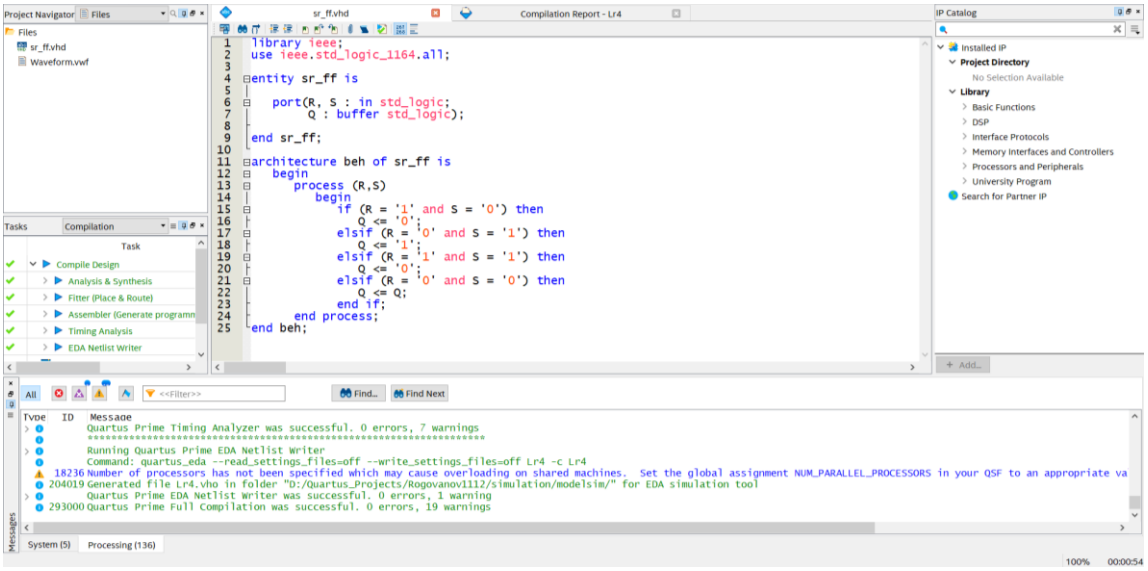

Рисунок 2.9 – Результат успешной компиляции проекта

После успешной компиляции проекта, проанализируем сигналы синтезируемого устройства с помощью утилиты Simulation Waveform. Результаты анализа представлены на рисунке 3.10.

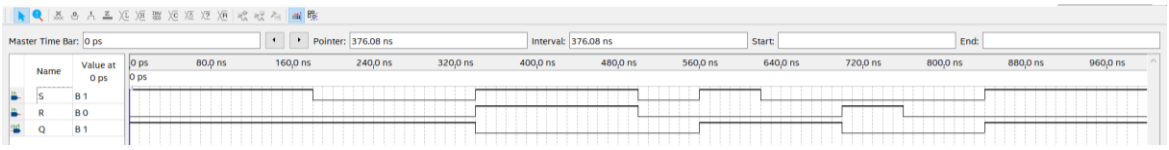

Рисунок 3.10 – Окно Simulation Waveform при анализе описанного $\overline{RS}$ – триггера

Исходя из данных, полученных на рисунке 3.10, можно заметить, что при подаче на вход S сигнала активного уровня, на выходе Q устройства также наблюдается сигнал активного уровня. Далее, в следствии отсутствия сигналов на обоих входах, триггер переходит в режим хранения и на его выходе по-прежнему наблюдается сигнал активного уровня. Если подать сигнал активного уровня на входы S и R одновременно, то на выходе Q устройства сигнала не будет, так как триггер находится в запрещенном режиме. Если на вход R подать сигнал активного уровня, то состояние триггера будет сброшено.

Таким образом описанный $\overline{RS}$ – триггер работает в соответствии с таблицей истинности.

Выводы

По результатам выполнения лабораторной работы были освоены навыки описания логических элементов с помощью языка описания аппаратуры VHDL на поведенческом уровне. В данной работе я познакомился с такими синтаксическими конструкциями языка, как if, else, then и т.д.

Способы описания устройств, представленные в данной лабораторной работе, не являются единственными. Помимо поведенческого стиля описания устройства могут быть использованы также структурный тип описания и потоковый тип описания, а также различные операторы, с помощью которых можно описать устройство.