БГУИР

Кафедра ЭВМ

Отчет по лабораторной работе № 2
Тема: «Проектирование функциональных узлов последовательного типа на языке VHDL в среде WEBPACK»
Вариант 7

Выполнил:

Проверил:

МИНСК 2024

1 ЦЕЛЬ РАБОТЫ

Получить навыки структурного описания устройств на языке VHDL.

2 ИСХОДНЫЕ ДАННЫЕ К РАБОТЕ

Задание для лабораторной работы состоит из двух частей.

1. В соответствии с полученным вариантом задания разработать на языке VHDL синхронный динамический триггер. Для его моделирования разработать тестовое воздействие в соответствии с рис. 2.1. Тестовое воздействие можно реализовать с помощью встроенного в САПР WebPACK графического редактора. Выполнить поведенческое (Behavioral Simulation) моделирование работы заданного триггера.

2. В соответствии с полученным вариантом задания на базе триггера, разработанного в первой части лабораторной работы, реализовать на языке VHDL функциональный узел последовательного типа. Описание функционального узла должно быть выполнено в структурном виде, то есть отражать структуру, а не поведение данного устройства. Функциональные схемы устройств последовательного типа, отражающие их структуру, предоставляются преподавателем.

3. Для реализованного функционального узла последовательного типа Библиотека БГУИР разработать по возможности полное тестовое воздействие и провести поведенческое (Behavioral Simulation) моделирование его работы. Данное тестовое воздействие должно проверять (моделировать) работу заданного устройства во всех режимах его работы. На рис. 2.1 представлены условные графические обозначения (УГО) синхронных динамических триггеров и диаграммы их работы.

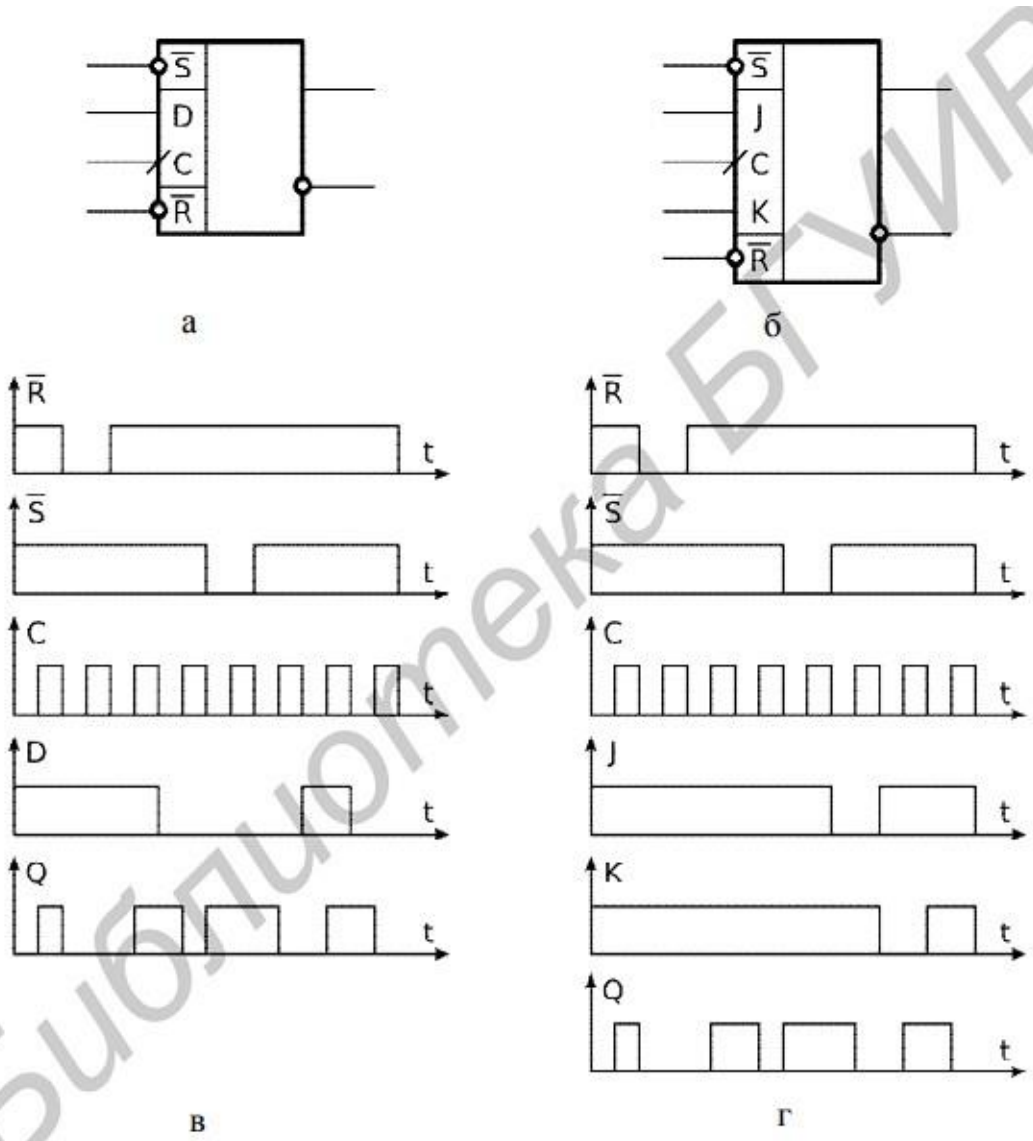

Рис. 2.1. УГО и диаграммы работы динамических триггеров :
а – синхронный динамический D-триггер с асинхронными входами сброса и установки; б – синхронный динамический JK-триггер с асинхронными входами сброса и установки; в – диаграммы работы D-триггера;
г – диаграммы работы JK-триггера

Варианты заданий представлены в таблице 2.1.

Таблица 2.1 – Варианты заданий

Номер варианта	Тип триггера	Имя файла описания функционального узла
1	D-триггер	1.pdf
2	JK-триггер	2.pdf
3	D-триггер	3.pdf
4	JK-триггер	4.pdf
5	D-триггер	5.pdf
6	JK-триггер	6.pdf
7	D-триггер	7.pdf
8	JK-триггер	8.pdf
9	D-триггер	9.pdf
10	JK-триггер	10.pdf
11	D-триггер	11.pdf
12	JK-триггер	12.pdf
13	D-триггер	13.pdf
14	JK-триггер	14.pdf
15	D-триггер	15.pdf
16	JK-триггер	16.pdf
17	D-триггер	17.pdf
18	JK-триггер	18.pdf
19	D-триггер	19.pdf
20	JK-триггер	20.pdf
21	D-триггер	21.pdf
22	JK-триггер	22.pdf
23	D-триггер	23.pdf
24	JK-триггер	24.pdf
25	D-триггер	25.pdf

3 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

3.1. Атрибуты в языке VHDL

Атрибуты – это данные скалярного типа, отражающие некоторые свойства объектов, используемых в описаниях на языке VHDL. Атрибут имеет

имя и тип. Атрибуты подразделяются на предопределенные и определенные пользователем. Атрибуты агрегатов удобно использовать для перебора их элементов в цикле. Атрибуты сигналов являются эффективным средством анализа поведения сигнала во времени и используются, например, для определения момента перехода сигнала из одного состояния в другое. Эта возможность, в свою очередь, часто используется при описании синхронных динамических функциональных узлов, то есть тех, которые изменяют свое состояние при переходах синхросигнала из 0 в 1 и наоборот. Ниже приведен пример использования атрибута сигнала для описания схемы, которая по фронту синхросигнала (clk), то есть при изменении его значения с 0 на 1, присваивает значение сигнала x сигналу y.

3.2. Виды представления описаний проектов на языке VHDL

Классически принято выделять три различных стиля описания аппаратных архитектур на зыке VHDL:

— структурное описание;

— потоковое описание;

— поведенческое описание.

При структурном описании (structural description) объекта проекта его архитектура представляется в виде иерархии связанных компонентов. Каждый экземпляр компонента представляет собой часть проекта, которая также может быть описана объектом проекта низшего уровня, состоящим из связанных компонентов. Таким образом может быть построена иерархия объектов, представляющих в конечном итоге весь проект.

Структурный тип описания отражает декомпозицию проекта на компоненты и делает акцент на соединениях, которые должны быть проведены между компонентами. При этом сами компоненты могут быть абстрактными функциональными устройствами или могут представлять отдельный вентиль, микросхему, плату или целую подсистему. Иерархия может представлять как структурное разбиение, так и функциональную декомпозицию разрабатываемого устройства.

При потоковом описании (data-flow description) объекта проекта его архитектура представляется в виде множества параллельных регистровых операций, каждая из которых управляется вентильными сигналами. Потоковое описание соответствует стилю описания, используемому в языках регистровых передач. В потоковом описании акцент делается на потоке информации между элементами с вентильным управлением. Этот поток информационного обмена регулируется и направляется управляющими элементами, которые логически отделены от путей прохождения данных. Так как пути данных показаны явно, то при потоковом описании не уделяется внимание структуре возможных реализаций.

Поведенческий стиль описания определяет последовательно выполнимый код процедурного типа, подобный коду на языках программирования общего назначения. Основная причина использования

поведенческого описания заключается в том, что данный стиль описания определяет с любой желаемой степенью точности функционирование устройства без определения его структуры. Большая часть способов описания комбинационных устройств из первой лабораторной работы представляла собой именно поведенческое описание.

Для использования объекта, описанного отдельным модулем entity, необходимо выполнить следующие действия.

1. В декларативной части проектного модуля architecture объявить прототип используемого объекта, воспользовавшись оператором объявления компоненты.

2. В операторной части проектного модуля architecture создать экземпляр требуемого объекта, воспользовавшись параллельным оператором создания экземпляра компонента.

4 ВЫПОЛНЕНИЕ РАБОТЫ

Ниже представлен текст программы для счетчика:

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

entity priority_encoder is
    Port (
        inputs : in std_logic_vector(8 downto 0);
        outputs : out std_logic_vector(3 downto 0)
    );
end priority_encoder;

architecture Behavioral of priority_encoder is
begin
    process(inputs)
    begin
        -- Инициализация выходов
        outputs <= "0000";

        -- Приоритетный кодировщик
        if inputs(8) = '1' then
            outputs <= "1001"; -- 9
        elsif inputs(7) = '1' then
            outputs <= "1000"; -- 8
        elsif inputs(6) = '1' then
            outputs <= "0111"; -- 7
        elsif inputs(5) = '1' then
            outputs <= "0110"; -- 6
        elsif inputs(4) = '1' then
            outputs <= "0101"; -- 5
        elsif inputs(3) = '1' then
            outputs <= "0100"; -- 4
        elsif inputs(2) = '1' then
```

```
            outputs <= "0011"; -- 3
        elsif inputs(1) = '1' then
            outputs <= "0010"; -- 2
        elsif inputs(0) = '1' then
            outputs <= "0001";  -- 1
        else outputs <= "0000"; -- 0
        end if;
    end process;
end Behavioral;
```

Testbench для проверки правильности работы:

```
library IEEE;
use  IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use std.textio.all;
use IEEE.std_logic_textio.all;

entity testbench is
end testbench;

architecture behavior of testbench is

    signal input_lines : std_logic_vector(8 downto 0);
    signal output_lines : std_logic_vector(3 downto 0);
    signal expected_output : std_logic_vector(3 downto 0);

    component bcd_priority_encoder
        port (
            input_lines : in std_logic_vector(8 downto 0);
            output_lines : out std_logic_vector(3 downto 0)
        );
    end component;

    -- Объявление файлового типа
    file file_input : text; -- Правильное объявление файла

begin

    uut: bcd_priority_encoder port map (
        input_lines => input_lines,
        output_lines => output_lines
    );

    process
        variable line_input : line;
        variable temp_input : std_logic_vector(8 downto 0);
        variable temp_expected : std_logic_vector(3 downto 0);
        variable temp_output : std_logic_vector(3 downto 0);
    begin
        -- Открытие файла для чтения
        file_open(file_input, "input_data.txt", read_mode);

        while not endfile(file_input) loop
            readline(file_input, line_input);
            read(line_input, temp_input);
            read(line_input, temp_expected);
            read(line_input, temp_output);


            input_lines <= temp_input;
            expected_output <= temp_expected;
```

```
            output_lines <= temp_output;

            wait for 10 ns; -- Задержка для симуляции

            assert output_lines = expected_output
            report "Ошибка: выходные данные не совпадают с ожидаемыми"
            severity error;
        end loop;

        file_close(file_input);

        wait;
    end process;

end behavior;
```

На рисунке 4.1 показана временная диаграмма, полученная в результате выполнения программы счетчика.

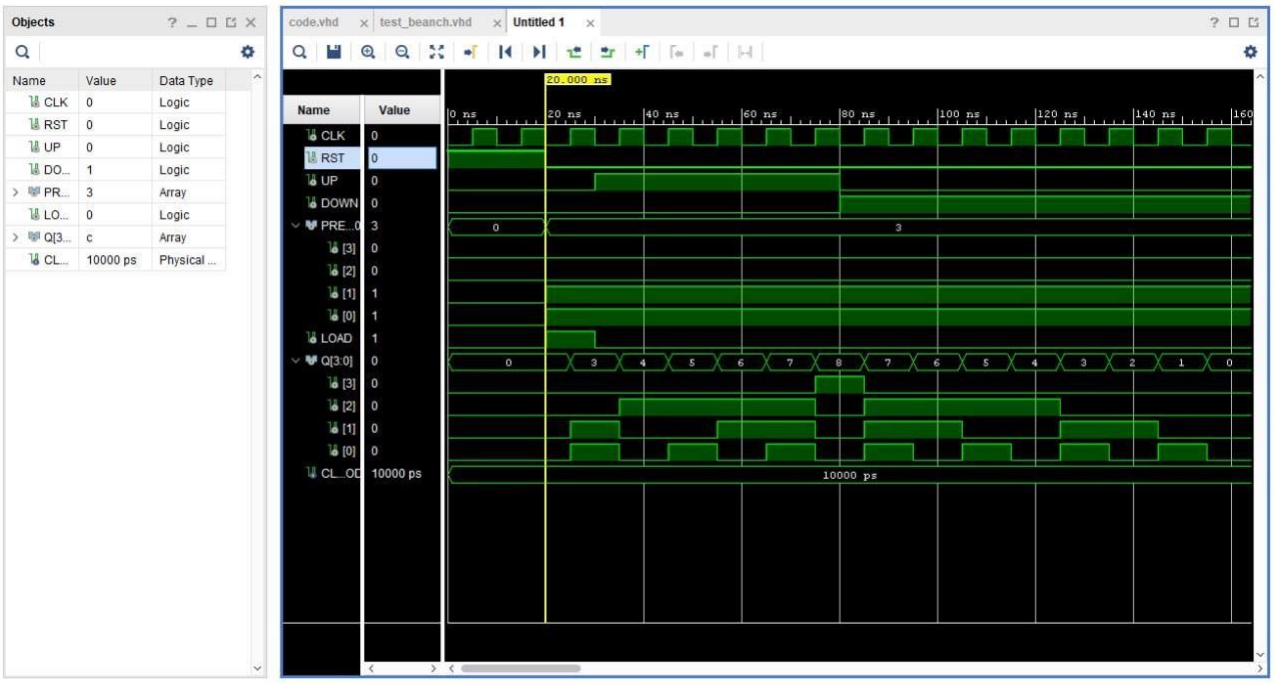

Рисунок 4.1 – Временная диаграмма счетчика

Ниже представлен текст программы для D- триггер:

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

entity priority_encoder is
    Port (
        inputs : in std_logic_vector(8 downto 0);
        outputs : out std_logic_vector(3 downto 0)
    );
end priority_encoder;

architecture Behavioral of priority_encoder is
begin
    process(inputs)
    begin
```

```
        -- Инициализация выходов
        outputs <= "0000";  -- Значение по умолчанию

        -- Приоритетный кодировщик с использованием операторов
        case inputs is
            when "100000000" =>  -- 9
                outputs <= "1001";
            when "010000000" =>  -- 8
                outputs <= "1000";
            when "001000000" =>  -- 7
                outputs <= "0111";
            when "000100000" =>  -- 6
                outputs <= "0110";
            when "000010000" =>  -- 5
                outputs <= "0101";
            when "000001000" =>  -- 4
                outputs <= "0100";
            when "000000100" =>  -- 3
                outputs <= "0011";
            when "000000010" =>  -- 2
                outputs <= "0010";
            when "000000001" =>  -- 1
                outputs <= "0001";
            when others => -- Все нули
                outputs <= "0000";
        end case;
    end process;
end Behavioral;
```

Testbench для проверки правильности работы:

```
library IEEE;
use  IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use std.textio.all;
use IEEE.std_logic_textio.all;

entity testbench is
end testbench;

architecture behavior of testbench is

    signal input_lines : std_logic_vector(8 downto 0);
    signal output_lines : std_logic_vector(3 downto 0);
    signal expected_output : std_logic_vector(3 downto 0);

    component bcd_priority_encoder
        port (
            input_lines : in std_logic_vector(8 downto 0);
            output_lines : out std_logic_vector(3 downto 0)
        );
    end component;

    -- Объявление файлового типа
    file file_input : text; -- Правильное объявление файла

begin

    uut: bcd_priority_encoder port map (
        input_lines => input_lines,
        output_lines => output_lines
    );
```

```
    process
        variable line_input : line;
        variable temp_input : std_logic_vector(8 downto 0);
        variable temp_expected : std_logic_vector(3 downto 0);
        variable temp_output : std_logic_vector(3 downto 0);
    begin
        -- Открытие файла для чтения
        file_open(file_input, "input_data.txt", read_mode);

        while not endfile(file_input) loop
            readline(file_input, line_input);
            read(line_input, temp_input);
            read(line_input, temp_expected);
            read(line_input, temp_output);


            input_lines <= temp_input;
            expected_output <= temp_expected;
            output_lines <= temp_output;

            wait for 10 ns; -- Задержка для симуляции

            assert output_lines = expected_output
            report "Ошибка: выходные данные не совпадают с ожидаемыми"
            severity error;
        end loop;

        file_close(file_input);

        wait;
    end process;

end behavior;
```

На рисунке 4.2 показана временная диаграмма, полученная в результате выполнения программы D-триггера.

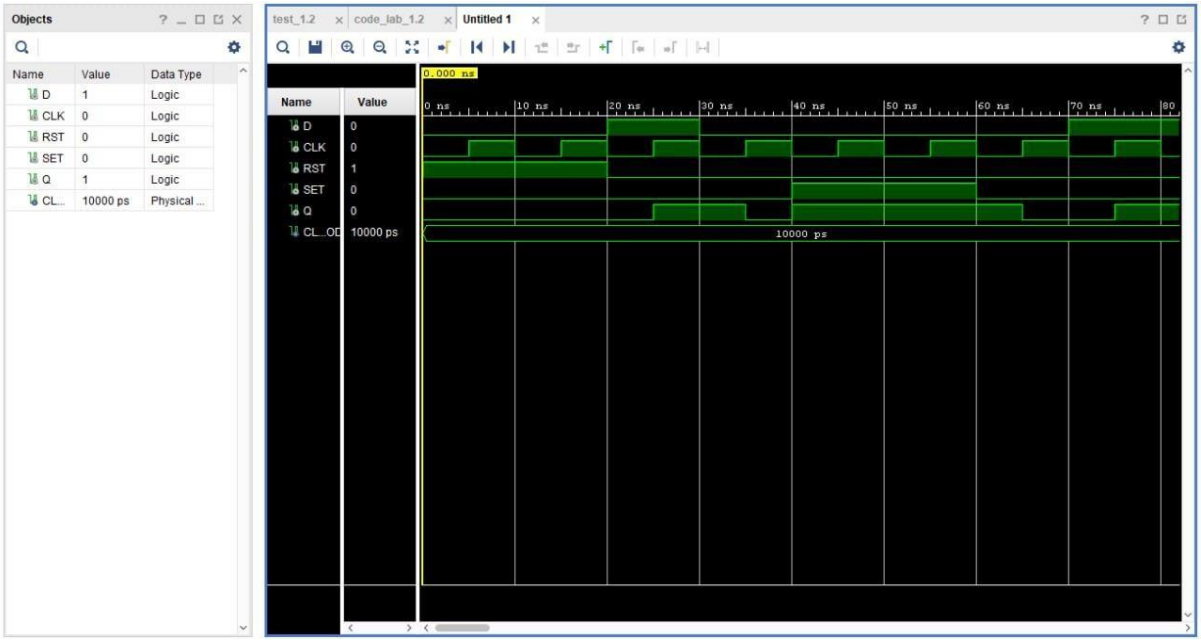

Рисунок 4.2 – Временная диаграмма D-триггера

5 ВЫВОДЫ

В ходе выполнения лабораторной работы были получены навыки структурного описания устройств на языке VHDL.