БГУИР

Кафедра ЭВМ

Отчет по лабораторной работе № 4
Тема: «Разработка и реализация цифровых устройств на базе макетной платы SET-StarterKit»

Вариант №7

Выполнил:

Проверил:

Минск
2024

1 ЦЕЛИ ЛАБОРАТОРНОЙ РАБОТЫ

Цели:

1. Изучить полный цикл проектирования цифровых устройств на базе ПЛИС с использованием САПР WebPACK.

2. Получить навыки проектирования и отладки цифровых устройств на базе ПЛИС.

2 ИСХОДНЫЕ ДАННЫЕ К РАБОТЕ

На базе своего варианта функционального узла последовательного типа, разработанного во второй лабораторной работе, спроектировать и реализовать на базе макетной платы одно из следующих устройств.

1. Счетчик, увеличивающийся (уменьшающийся) на 1 с частотой, заданной преподавателем. При этом в качестве компонента обязательно использовать счетчик, разработанный в лабораторной работе №2. Выходы данного счетчика подключить к выводам микросхемы FPGA, соединенным со светодиодами, а управляющие входы счетчика – к выводам микросхемы FPGA, подключенным к переключателям.

2. Регистр, сохраняющий значения с частотой, заданной преподавателем. В качестве компонента обязательно использовать регистр, разработанный в лабораторной работе №2. Выходы регистра соединить с выводами микросхемы FPGA, подключенными к светодиодам, а управляющие входы – с выводами микросхемы FPGA, подключенными к переключателям.

Для разработанного устройства создать тестовое воздействие и выполнить его функциональное (поведенческое) моделирование.

Выполнить синтез спроектированного устройства. При необходимости внести коррективы в его описание.

Провести функциональное моделирование описания устройства, полученного после его синтеза.

Выполнить реализацию (получение физического описания) устройства на базе микросхемы FPGA Spartan II XC2S200-6FG256.

Провести временное моделирование полученного описания устройства и добиться его корректной работы. При необходимости внести изменения в описание проектируемого устройства.

Сгенерировать файл конфигурационной последовательности для микросхем FPGA и FLASH-памяти.

Выполнить программирование микросхемы на макетной плате и убедиться в корректной работе спроектированного устройства.

3 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

3.1 Полный цикл проектирования цифровых устройств в САПР WebPACK

Процесс проектирования цифровых устройств на базе FPGA состоит из следующих этапов: ввод описания, синтез, преобразование в физическую реализацию, создание конфигурационного файла и его передача в микросхему FPGA (её программирование). Моделирование описания, включающее в себя как функциональное, так и временное моделирование, выполняется на разных этапах процесса проектирования для контроля полученного описания устройства. Далее более подробно опишем назначение каждого этапа в отдельности.

3.1.1 Ввод описания

Ввод описания разрабатываемого устройства состоит из следующих этапов:

— создание проекта;

— создание новых или добавление в проект уже существующих исходных файлов проектов, включая файлы временных и топологических ограничений;

— ввод описания проектируемого устройства;

— установка значений временных и топологических ограничений.

В предыдущих лабораторных работах выполнялись все этапы ввода проекта, кроме последнего, так как информация, устанавливаемая на этом этапе, не обходима только начиная с синтеза проекта.

3.1.2 Синтез проекта

После ввода проекта и выполнения необязательного функционального моделирования необходимо выполнить синтез проекта. Это можно сделать через окно исходных модулей проекта, воспользовавшись командой Synthesize XST. Синтез может осуществляться как встроенным синтезатором, так и внешней программой, поставляемой в составе других САПР, выполняющих те же функции. В лабораторных работах будет использоваться встроенная программа-синтезатор XST. Во время синтеза выполняется преобразование исходного описания устройства в описание на базе логических блоков и функциональных узлов, для которых впоследствии может быть создана эффективная физическая реализация на базе микросхем FPGA. Фактически выполняется синтез устройства на базе стандартных функциональных схемотехнических узлов.

Результат синтеза выводится в консоль, а также может быть выведен на экран в отдельном окне с помощью команды View Synthesis Report. Кроме этого, результат синтеза можно посмотреть и в графической форме, воспользовавшись командами окна процессов View RTL Schematic и View Technology Schematic.

Описание устройства, полученное на этом этапе, в большинстве случаев является еще аппаратно независимым, то есть не зависит от микросхемы, на которой оно будет реализовано.

3.1.3 Преобразование в физическую реализацию

Результат синтеза описания устройства используется для получения его физической реализации на заданной микросхеме FPGA. Получение физической реализации выполняется в несколько этапов:

1) Translate;
2) Map;
3) Place and Route;
4) создание конфигурационного файла.

Выполнить данные этапы можно, воспользовавшись соответствующими командами из окна процессов САПР WebPACK (Processes).

На первом этапе (Translate) выполняется преобразование описания, по лученного после синтеза проекта, в описание устройства на базе примитивов WebPACK, каждый из которых помещается в базу данных используемых эле ментов.

На втором этапе (Map) каждый элемент из базы данных, полученной на предыдущем этапе, реализуется на компонентах, входящих в состав заданной микросхемы FPGA.

На третьем этапе (Place and Route) выполняется размещение и развод ка описания устройства, полученного на предыдущем этапе на заданной микросхеме.

На последнем этапе выполняется генерация конфигурационного файла (файла прошивки). Для этого используется команда Generate Programming File окна процессов программы ProjectNavigator.

3.1.4 Загрузка конфигурационного файла в микросхему FPGA

После того как получен конфигурационный файл для прошивки FPGA, не обходимо загрузить его в микросхему.

Лабораторная работа выполняется с использованием платы SET-StarterKit, ядром которой является микросхема FPGA Spartan II XC2S200. Данная плата не позволяет напрямую загружать конфигурационную информацию в FPGA. Вместо этого необходимо сформировать файл прошивки для микросхемы FLASH памяти XC18V02, размещенной на этой же плате, и загрузить его в эту микросхему, воспользовавшись интерфейсом JTAG. После включения питания либо Библиотека БГУИР 39 при нажатии кнопки «Program» конфигурационная информация из этой микро схемы будет автоматически загружена в FPGA.

Для формирования файла прошивки FLASH-памяти можно воспользоваться командой Generate PROM, ACE or JTAG File окна процессов программы ProjectNavigator, при этом загрузится специализированный модуль САПР WebPACK – IMPACT, предназначенный для формирования файлов прошивок конфигурационной последовательности и загрузки этих файлов в микросхемы посредством использования различных интерфейсов.

После загрузки модуля IMPACT автоматически запускается мастер, позволяющий облегчить процесс выполнения необходимой операции, которая может быть выбрана на первой вкладке данного мастера. Для формирования файла прошивки FLASH-памяти на первом этапе необходимо выбрать операцию Prepare a PROM File.

На втором этапе работы мастера (вкладка Prepare PROM Files) необходимо указать устройство, для которого будут сформированы прошивка (Xilinx PROM), тип файла прошивки (MCS), имя файла прошивки (PROM File Name) и путь, куда будет записан сформированный файл.

На третьем этапе (вкладка Specify Xilinx PROM Device) создается спи сок микросхем памяти, для которых будет генерироваться файл прошивки. Здесь необходимо выбрать тип используемой микросхемы FLASH-памяти (XC18V02) и добавить ее в список.

На последнем этапе выводится для проверки общая информация по выполняемой операции, после чего запрашивается имя файла конфигурационной последовательности, сгенерированного на предыдущих этапах. После выбора необходимого файла и отрицательного ответа на запрос о добавлении другого устройства необходимый файл прошивки FLASH-памяти будет сгенерирован модулем IMPACT, о чем будет сообщено всплывающим информационным окном.

После получения файла прошивки FLASH-памяти можно приступать к его загрузке в микросхему. Для этого в модуле IMPACT необходимо перейти в ре жим работы с использованием JTAG интерфейса (Boundary Scan). После этого загрузка файла прошивки в микросхему FLASH-памяти также осуществляется в несколько этапов. На первом этапе необходимо запустить процедуру JTAG сканирования для определения на макетной плате микросхем, поддерживающих JTAG. В случае успешного завершения этой операции будут обнаружены две микросхемы: ПЛИС FPGA Spartan II (XC2S200) и конфигурационная FLASH-память (XCV18V02).

На втором этапе на запрос файла для программирования FLASH-памяти (файл с расширением MCS) указывается файл, сгенерированный ранее. При этом на запрос файла прошивки для FPGA (файл с расширением BIT) необходимо нажать кнопку BYPASS, что означает отсутствие конфигурационного файла для FPGA.

После этого необходимо, воспользовавшись контекстным меню по нажатию правой кнопки мыши на изображении FLASH-памяти, вызвать процедуру ее программирования.

После успешного завершения процедуры загрузки конфигурационной последовательности во FLASH-память можно воспользоваться кнопкой «Program» для загрузки конфигурационной информации в FPGA.

3.1.5 Подключение FPGA на плате SET-StarterKit

Ядром платы SET-StarterKit является микросхема ПЛИС FPGA Spartan II (XC2S200), тактируемая сигналом внешнего тактового генератора с

частотой 50 Мгц. Кроме этого, на плате есть набор светодиодов, подключенных к выводам микросхемы FPGA, работающим в режиме выходов, и набор переключателей, подключенных к выводам микросхемы FPGA, работающим в режиме входов. Схема подключения описанных выводов микросхемы FPGA изображена на рисунке 3.1.

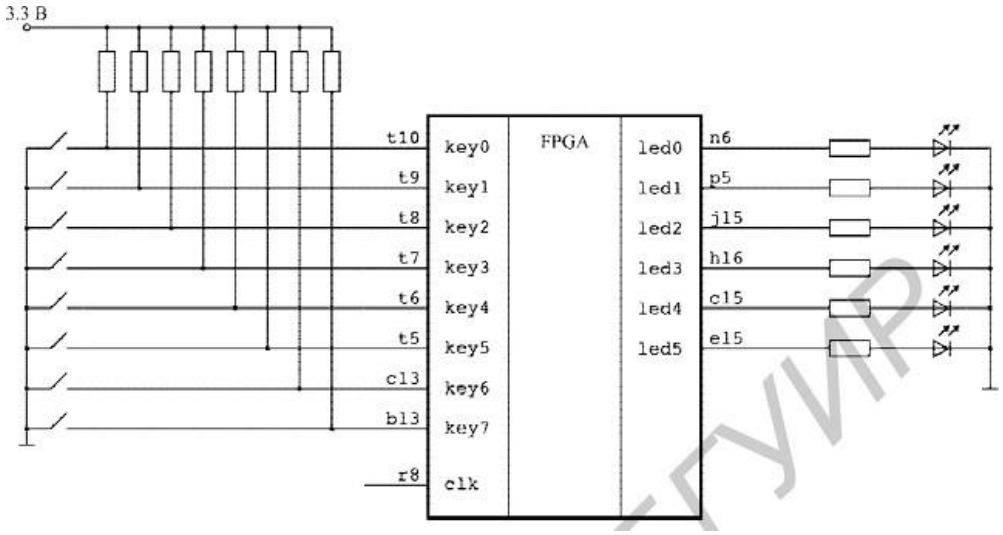

Рисунок 3.1 – Схема подключения микросхемы FPGA на макетной плате

3.1.6 Файл временных и топологических ограничений

Файл временных и топологических ограничений содержит дополнительную информацию для программ синтеза, размещения и трассировки. Этот файл имеет текстовый формат, каждая строка которого представляет собой выражение, описывающее соответствующий параметр. Чаще всего файл временных и топологических ограничений используют для задания соответствия между входами/выходами разработанного устройства и выводами микросхемы FPGA.

4 ЛИСТИНГ ПРОГРАММЫ

Обёртка:

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

entity artix is
    Port(
        ledsmain: out std_logic_vector(3 downto 0);     --светодиоды
        ledsboard: out std_logic_vector(3 downto 0);    --светодиоды
        pushbuttons:in std_logic_vector(4 downto 0);    --кнопки
        dipswitch:in std_logic_vector(3 downto 0);      --переключатели
        sysclk_p:in std_logic;                          --принимают дифф
тактовый сигнал
        sysclk_n:in std_logic
    );
```

```
end artix;

architecture Behavioral of artix is

     component UpDownCounter
            port (
                 RST : in STD_LOGIC;
                 CLK : in STD_LOGIC;
                 UP  : in  STD_LOGIC;
                 DOWN: in  STD_LOGIC;
                 LOAD: in  STD_LOGIC;
                 PRESET   : in  STD_LOGIC_VECTOR(3 downto 0);
                 Q: out STD_LOGIC_VECTOR(3 downto 0)
            );
       end component;

    component ibufds
    port (
        I, IB : in std_logic;   --
        O     : out std_logic); --
    end Component;

    component clk_div is    --делитель частоты
    port (
        clk_in  : in std_logic;
        clk_out : out std_logic);
    end Component;

    signal PRESET : std_logic_vector(3 downto 0);
    signal Q: std_logic_vector(3 downto 0);
    signal clk_no_div: std_logic;
    signal clk_divided: std_logic;
begin

    counter_uut: UpDownCounter
    Port Map (
      RST => pushbuttons(1), --левая
      CLK => clk_divided,
      LOAD => pushbuttons(0), --верхняя
      PRESET => PRESET,
      UP => pushbuttons(2), --центральная
      DOWN=> pushbuttons(3), --правая
      Q => Q
    );

    PRESET(3 downto 0) <= dipswitch(3 downto 0);
    ledsmain(3 downto 0) <= Q(3 downto 0);

    buff: ibufds
    Port Map (
        I => sysclk_p,
        IB=> sysclk_n,
        O => clk_no_div
    );

    divider_uut: clk_div
    Port Map (
        clk_in => clk_no_div,
        clk_out => clk_divided
    );

end Behavioral;
```

Счётчик:

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

entity UpDownCounter is
    Port ( CLK     : in  STD_LOGIC;
           RST     : in  STD_LOGIC;  -- Асинхронный сброс
           UP      : in  STD_LOGIC;  -- Счет вверх
           DOWN    : in  STD_LOGIC;  -- Счет вниз
           PRESET   : in  STD_LOGIC_VECTOR(3 downto 0); -- Предустановка
           LOAD       : in   STD_LOGIC;   -- Загрузка предустановленного
значения
           Q       : out STD_LOGIC_VECTOR(3 downto 0)); -- Выход счетчика
end UpDownCounter;

architecture Behavioral of UpDownCounter is
    signal count : STD_LOGIC_VECTOR(3 downto 0) := (others => '0');
begin

    process(CLK, RST)
    begin
        if (RST = '1') then
            count <= (others => '0');  -- Сброс счетчика
        elsif rising_edge(CLK) then
            if (LOAD = '1') then
                count <= PRESET;  -- Загрузка предустановленного значения
            elsif (UP = '1' and DOWN = '0') then
                count <= count + 1;  -- Счет вверх
            elsif (DOWN = '1' and UP = '0') then
                count <= count - 1;  -- Счет вниз
            end if;
        end if;
    end process;

    Q <= count;  -- Выход счетчика

end Behavioral;
```

Делитель частоты:

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

entity clk_div is
    Port ( clk_in: in std_logic;
           clk_out: out std_logic);
end clk_div;

architecture Behavioral of clk_div is
    signal temp_clk : std_logic := '0';

    constant divide_value: integer := 200000000 / 2; -- Делитель частоты
begin
    process (clk_in)
       variable i  : integer := 0;
    begin
        if clk_in'event and clk_in = '1' then
            if (i = divide_value) then
                i := 0;
```

```
                temp_clk <= not temp_clk;
            else
                i := i + 1;
            end if;
        end if;
    end process;

    clk_out <= temp_clk;
end Behavioral;
```

Пины:

```
# LEDs on AC701 main board
set_property PACKAGE_PIN M26 [get_ports ledsmain[0]]
set_property IOSTANDARD LVCMOS33 [get_ports ledsmain[0]]
set_property PACKAGE_PIN T24 [get_ports ledsmain[1]]
set_property IOSTANDARD LVCMOS33 [get_ports ledsmain[1]]
set_property PACKAGE_PIN T25 [get_ports ledsmain[2]]
set_property IOSTANDARD LVCMOS33 [get_ports ledsmain[2]]
set_property PACKAGE_PIN R26 [get_ports ledsmain[3]]
set_property IOSTANDARD LVCMOS33 [get_ports ledsmain[3]]

# LEDs on external board
set_property PACKAGE_PIN F25 [get_ports ledsboard[0]]
set_property IOSTANDARD LVCMOS25 [get_ports ledsboard[0]]
set_property PACKAGE_PIN G25 [get_ports ledsboard[1]]
set_property IOSTANDARD LVCMOS25 [get_ports ledsboard[1]]
set_property PACKAGE_PIN G26 [get_ports ledsboard[2]]
set_property IOSTANDARD LVCMOS25 [get_ports ledsboard[2]]
set_property PACKAGE_PIN H26 [get_ports ledsboard[3]]
set_property IOSTANDARD LVCMOS25 [get_ports ledsboard[3]]

# Pushbuttons
set_property PACKAGE_PIN P6 [get_ports pushbuttons[0]]
set_property IOSTANDARD LVCMOS15 [get_ports pushbuttons[0]]
set_property PACKAGE_PIN R5 [get_ports pushbuttons[1]]
set_property IOSTANDARD SSTL15 [get_ports pushbuttons[1]]
set_property PACKAGE_PIN U6 [get_ports pushbuttons[2]]
set_property IOSTANDARD SSTL15 [get_ports pushbuttons[2]]
set_property PACKAGE_PIN U5 [get_ports pushbuttons[3]]
set_property IOSTANDARD SSTL15 [get_ports pushbuttons[3]]
set_property PACKAGE_PIN T5 [get_ports pushbuttons[4]]
set_property IOSTANDARD SSTL15 [get_ports pushbuttons[4]]

# DIP switches
set_property PACKAGE_PIN R8 [get_ports dipswitch[0]]
set_property IOSTANDARD SSTL15 [get_ports dipswitch[0]]
set_property PACKAGE_PIN P8 [get_ports dipswitch[1]]
set_property IOSTANDARD SSTL15 [get_ports dipswitch[1]]
set_property PACKAGE_PIN R7 [get_ports dipswitch[2]]
set_property IOSTANDARD SSTL15 [get_ports dipswitch[2]]
set_property PACKAGE_PIN R6 [get_ports dipswitch[3]]
set_property IOSTANDARD SSTL15 [get_ports dipswitch[3]]

# System clock 200 MHz
set_property PACKAGE_PIN R3 [get_ports sysclk_p]
set_property IOSTANDARD LVDS_25 [get_ports sysclk_p]
set_property PACKAGE_PIN P3 [get_ports sysclk_n]
set_property IOSTANDARD LVDS_25 [get_ports sysclk_n]
```

4. ЗАКЛЮЧЕНИЕ

3. В ходе выполнения лабораторной работы мы изучили полный цикл проектирования цифровых устройств на базе ПЛИС с использованием САПР WebPACK, а также получили навыки проектирования и отладки цифровых устройств на базе ПЛИС.