Министерство образования Республики Беларусь

Учреждение образования

Белорусский государственный университет информатики и радиоэлектроники

Кафедра РТС

Отчет по лабораторной работе №2

«РЕАЛИЗАЦИЯ КОМБИНАЦИОННЫХ УСТРОЙСТВ НА ПЛИС»

Выполнил:

ст.гр. 240102
shlom41k

Проверил:

xxxxxxxxx

Минск 2015

Цель работы

1. Углубление и закрепление теоретических знаний по схемотехническому проектированию цифровых устройств на логических элементах и описание их работы на языке высокого уровня.

2. Формирование практических навыков создания цифровых устройств путем схемотехнического описания и описания работы устройства на языке VHDL, а также компьютерного моделирования их работы.

3. Приобретение практических навыков работы с реальными устройствами на базе ПЛИС и контрольно-измерительными приборами.

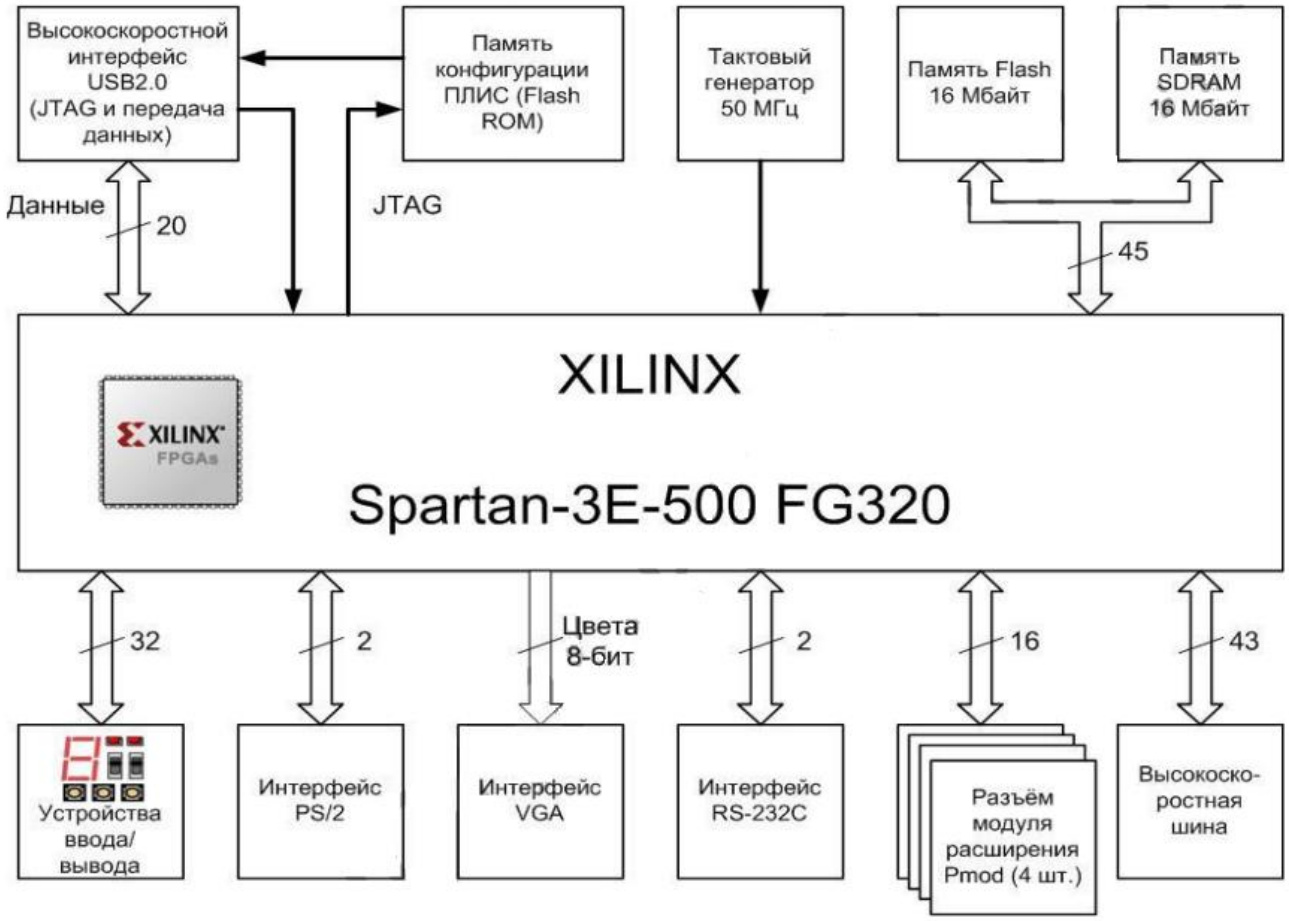

Рисунок *1* – Структура отладочной платы

Условия индивидуального задания

Реализовать на основе ПЛИС устройство, которое находит заданную двоичную последовательность во входном сигнале, формируемую генератором ПСП. Искомая последовательность задается переключателями. Организовать подсчет количества найденных последовательностей и индикацию их числа на семисегментном индикаторе.

Листинг программы

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

entity qqq is

   Port ( clock_en      : in STD_LOGIC;                    -- Разрешение счета
          clk           : in  STD_LOGIC;                   -- Входная частота
          reset         : in  STD_LOGIC;                   -- Сброс
          InStr         : in  std_logic_vector (3 downto 0);  -- Задаваемая искомая последовательность
          F_out         : out  STD_LOGIC;                  -- Выходная частота 15 Гц
          dout          : out  STD_LOGIC                   -- Выход ПСП
          OutInd        : out  std_logic_vector (7 downto 0);  -- ABCDEFGDP
          OutAN         : out  std_logic_vector (3 downto 0);  -- Разряды
          );
end qqq;

architecture Behavioral of qqq is

        signal counter15:       std_logic_vector (20 downto 0);     -- Счетчик на 15 Гц
        signal counter30:       std_logic_vector (20 downto 0);     -- Счетчик на 30 Гц
        signal F_out_temp:      std_logic;                          -- Выходной сигнал 15 Гц
        signal reg:             std_logic_vector (8 downto 0);      -- LFSR регистр
        signal InStrTemp:       std_logic_vector (3 downto 0);      -- Искомая последовательность
        signal stec:            std_logic_vector (3 downto 0);      -- Регистр, где хранится текущее
                                                                    -- значение последовательности
        signal VP:              std_logic_vector (7 downto 0);      -- Переменная для преобразования из 2
                                                                    -- в 10
        signal AN:              std_logic_vector (3 downto 0);      -- Регистр для сдвига текущего разряда
        signal CountStrBuf:     std_logic_vector (3 downto 0);      -- Регистр для хранения текущего
                                                                    -- значения 2-го числа
        signal CountStr0:       std_logic_vector (3 downto 0);      -- Счетчик 0-го разряда
        signal CountStr1:       std_logic_vector (3 downto 0);      -- Счетчик 1-го разряда
        signal CountStr2:       std_logic_vector (3 downto 0);      -- Счетчик 2-го разряда
        signal CountStr3:       std_logic_vector (3 downto 0);      -- Счетчик 3-го разряда

begin

process (clk, reset)

begin

        if reset = '1' then

                counter15 <= (others => '0');
                reg <= "010101110";                 -- Начальное значение регистра LFSR
                AN <= "1110";                       -- Начальное значение регистра текущего разряда
                stec <= "0000";                     -- Начальное значение регистра для хранения
                                                    -- последовательности
                CountStr0 <= "0000";                -- Обнуление счетчиков разрядов
                CountStr1 <= "0000";                --//--
                CountStr2 <= "0000";                --//--
```

```
            CountStr3 <= "0000";                    --//--

elsif clk = '1' and clk'event then

    if InStrTemp /= InStr then              -- Если произошло изменение входной последовательности, то

            CountStr0 <= "0000";            -- обнуляем счетчики разрядов
            CountStr1 <= "0000";            --//--
            CountStr2 <= "0000";            --//--
            CountStr3 <= "0000";            --//--
    end if;

            if clock_en = '1' then

                    counter15 <= counter15 + 1;
                    counter30 <= counter30 + 1;
            end if;

if counter15 = "11001011011100110101O" then         -- 15 Гц

            InStrTemp <= InStr;                     -- Считываем значение входной комбинации в регистр
            counter15 <= (others => '0');
            F_out_temp <= not F_out_temp;           -- Меняем значение (частота 15 Гц)
            reg <= (reg(4) xor reg(0)) & reg(8 downto 1);   -- Генерируем новый бит
            stec <= reg(0) & stec(3 downto 1);      -- Заносим сгенерированный бит в стек

                    if stec = InStrTemp then        -- Нашли нужную последовательность

                    CountStr0 <= CountStr0 +1;      -- +1 к счетчику 0-го разряда

                    end if;

                    if CountStr0 = "1010" then      -- 0-й разряд переполнен

                            CountStr0 <= "0000";    -- Сброс счетчика 0-го разряда
                            CountStr1 <= CountStr1 + 1;     -- +1 к счетчику 1-го разряда

                    end if;

                    if CountStr1 = "1010" then      -- 1-й разряд переполнен

                            CountStr1 <= "0000";    -- Сброс счетчика 1-го разряда
                            CountStr2 <= CountStr2 + 1;     -- +1 к счетчику 2-го разряда

                    end if;

                    if CountStr2 = "1010" then      -- 2-й разряд переполнен

                            CountStr2 <= "0000";    -- Сброс счетчика 2-го разряда
                            CountStr3 <= CountStr3 + 1;     -- +1 к счетчику 3-го разряда

                    end if;

                    if CountStr3 = "1010" then      -- 3-й разряд переполнен
```

```
                    CountStr0 <= "0000";            -- Сброс счетчика 0-го разряда
                    CountStr1 <= "0000";            -- Сброс счетчика 1-го разряда
                    CountStr2 <= "0000";            -- Сброс счетчика 2-го разряда
                    CountStr3 <= "0000";            -- Сброс счетчика 3-го разряда

                end if;

        end if;

            if counter30 = "00011001010110111001101"  then          -- 30 Гц

                counter30 <= (others => '0');
                AN <= AN(0) & AN(3 downto 1);          -- Вычисляем значение текущего разряда

            end if;

        end if;

case AN is                                              -- Если текущий разряд

        when "1110" => CountStrBuf <= CountStr0;      --0, заносим в буфер содержимое 0-го разряда
        when "1101" => CountStrBuf <= CountStr1;      --1, заносим в буфер содержимое 1-го разряда
        when "1011" => CountStrBuf <= CountStr2;      --2, заносим в буфер содержимое 2-го разряда
        when "0111" => CountStrBuf <= CountStr3;      --3, заносим в буфер содержимое 3-го разряда

end case;

case CountStrBuf is                                     -- Если в буфере индикации

        when "0000" => VP <= "00000011";              -- 0, преобразуем его в семисегментный код
        when "0001" => VP <= "10011111";              -- 1, преобразуем его в семисегментный код
        when "0010" => VP <= "00100101";              -- 2, преобразуем его в семисегментный код
        when "0011" => VP <= "00001101";              -- 3, преобразуем его в семисегментный код
        when "0100" => VP <= "10011001";              -- 4, преобразуем его в семисегментный код
        when "0101" => VP <= "01001001";              -- 5, преобразуем его в семисегментный код
        when "0110" => VP <= "01000001";              -- 6, преобразуем его в семисегментный код
        when "0111" => VP <= "00011111";              -- 7, преобразуем его в семисегментный код
        when "1000" => VP <= "00000001";              -- 8, преобразуем его в семисегментный код
        when "1001" => VP <= "00001001";              -- 9, преобразуем его в семисегментный код

end case;


        F_out <= F_out_temp;            -- Выходная частота 15 Гц
        dout <= reg(0);                 -- Выходной бит ПСП (15 Гц)
        OutInd <= VP;                   -- Вывод семисегментного кода на индикатор
        OutAN <= AN;                    -- Подсвечиваем текущий разряд

end process;

end Behavioral;
```

Используя осциллограф, измерили временные диаграммы формируемого генератором ПСП сигнала.

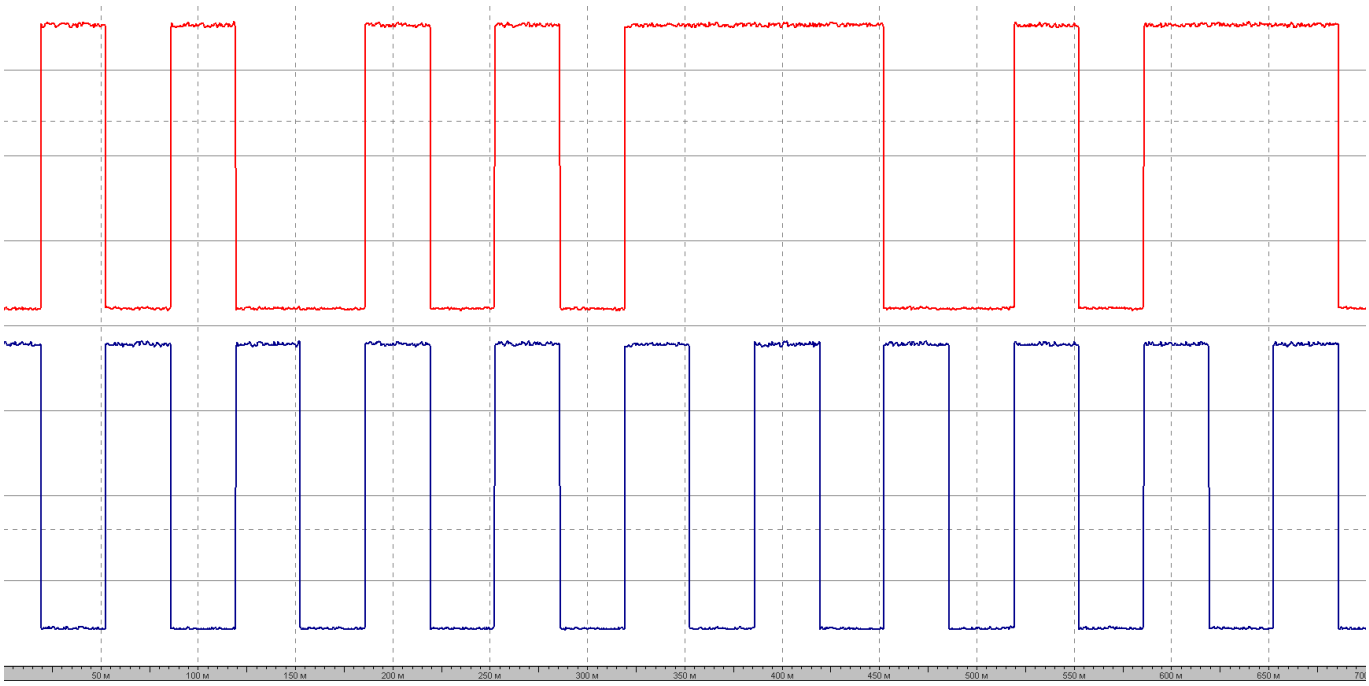

Рисунок *2* – Временные диаграммы сигналов (*F* = 15 Гц)

Вывод

В данной лабораторной работе углубили и закрепили теоретические знания по схемотехническому проектированию цифровых устройств на логических элементах и описание их работы на языке высокого уровня. Сформировали практические навыки создания цифровых устройств путем схемотехнического описания и описания работы устройства на языке VHDL, а также компьютерного моделирования их работы. Приобрели практические навыки работы с реальными устройствами на базе ПЛИС и контрольно-измерительными приборами.