Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧЕТ**

по лабораторной работе №3

по дисциплине «Проектирование программного обеспечения для устройств на ПЛИС»

на тему «Разработка и отладка цифровых устройств на языке VHDL»

Выполнили студенты группы 22ВВП1

Хоссейни Нежад С.А.С.М.

Беляев Д. И.

Демин М.

Приняли:

Федюнин Р.Н.

Гурин Е.И.

Пенза 2024

**Название**

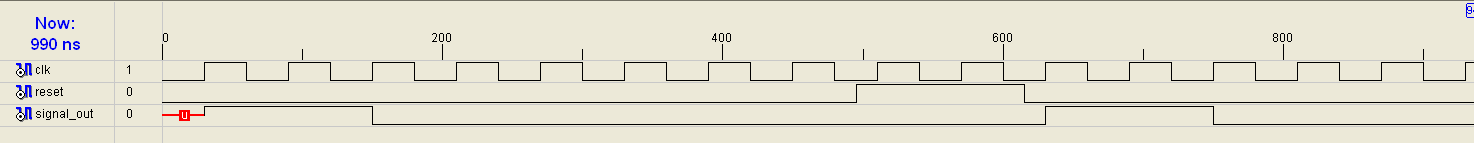
Разработка и отладка цифровых устройств на языке VHDL.

**Цель работы**

Изучение принципов разработки схем на языке VHDL

**Ход работы**

1. Разработать на языке VHDL схему формирователя  

Листинг

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity p1 is

Port (

clk : in STD\_LOGIC; -- Синхросигнал

reset : in STD\_LOGIC; -- Вход сброса

signal\_out : out STD\_LOGIC -- Выход сигнала

);

end p1;

architecture Behavioral of p1 is

signal state : INTEGER := 0; -- Состояние (0 - не сигнальное, 1 - сигнальное, 2 - завершено)

signal clk\_prev : STD\_LOGIC := '0'; -- Предыдущее состояние clk

begin

process(clk, reset)

begin

if reset = '1' then

state <= 0; -- Сброс состояния

signal\_out <= '0'; -- Выход в не сигнальное состояние

else

-- Проверяем, изменился ли сигнал clk

if clk = '1' and clk\_prev = '0' then

if state = 0 then -- Если в не сигнальном состоянии

state <= 1; -- Переход в сигнальное состояние

signal\_out <= '1'; -- Установить выход в сигнальное состояние

elsif state = 1 then -- Если в сигнальном состоянии

state <= 2; -- Переход в состояние завершения

signal\_out <= '1'; -- Выход остается в сигнальном состоянии

elsif state = 2 then -- Если в состоянии завершения

state <= 2; -- Оставаться в состоянии завершения

signal\_out <= '0'; -- Выход в не сигнальное состояние

end if;

end if;

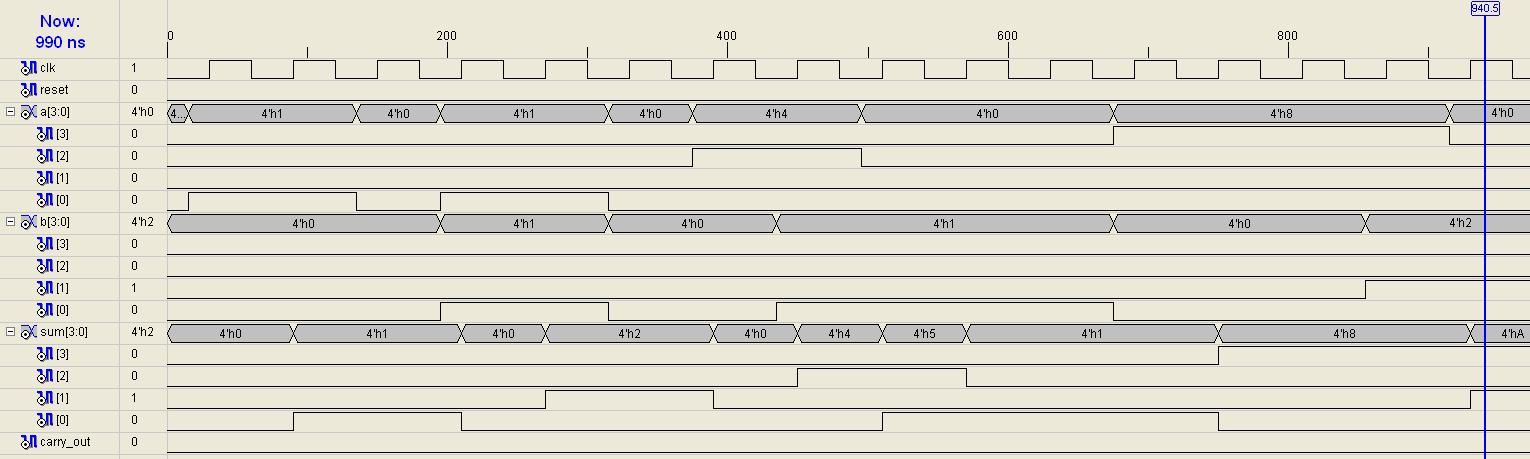
clk\_prev <= clk; -- Сохраняем текущее состояние clk

end if;

end process;

end Behavioral;

2. Разработать на языке VHDL схему четырехразрядного сумматора с регистрами на входах и выходе



Листинг 1

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity p2 is

Port (

clk : in STD\_LOGIC; -- Тактовый сигнал

reset : in STD\_LOGIC; -- Вход сброса

a : in STD\_LOGIC\_VECTOR(3 downto 0); -- Первый вход (4 бита)

b : in STD\_LOGIC\_VECTOR(3 downto 0); -- Второй вход (4 бита)

sum : out STD\_LOGIC\_VECTOR(3 downto 0); -- Выход суммы (4 бита)

carry\_out : out STD\_LOGIC -- Выход переноса

);

end p2;

architecture Behavioral of p2 is

signal reg\_a : STD\_LOGIC\_VECTOR(3 downto 0) := (others => '0'); -- Регистр для первого входа

signal reg\_b : STD\_LOGIC\_VECTOR(3 downto 0) := (others => '0'); -- Регистр для второго входа

signal reg\_sum : STD\_LOGIC\_VECTOR(3 downto 0) := (others => '0'); -- Регистр для суммы

signal carry : STD\_LOGIC := '0'; -- Перенос

begin

-- Процесс для регистрации входов

process(clk, reset)

begin

if reset = '1' then

reg\_a <= (others => '0'); -- Сброс первого регистра

reg\_b <= (others => '0'); -- Сброс второго регистра

reg\_sum <= (others => '0'); -- Сброс регистра суммы

carry <= '0'; -- Сброс переноса

elsif rising\_edge(clk) then

reg\_a <= a; -- Запись первого входа в регистр

reg\_b <= b; -- Запись второго входа в регистр

-- Суммирование с учетом переноса

reg\_sum <= reg\_a + reg\_b; -- Сумма

if (reg\_a + reg\_b) > "1111" then -- Проверка на перенос

carry <= '1'; -- Устанавливаем перенос

else

carry <= '0'; -- Сбрасываем перенос

end if;

end if;

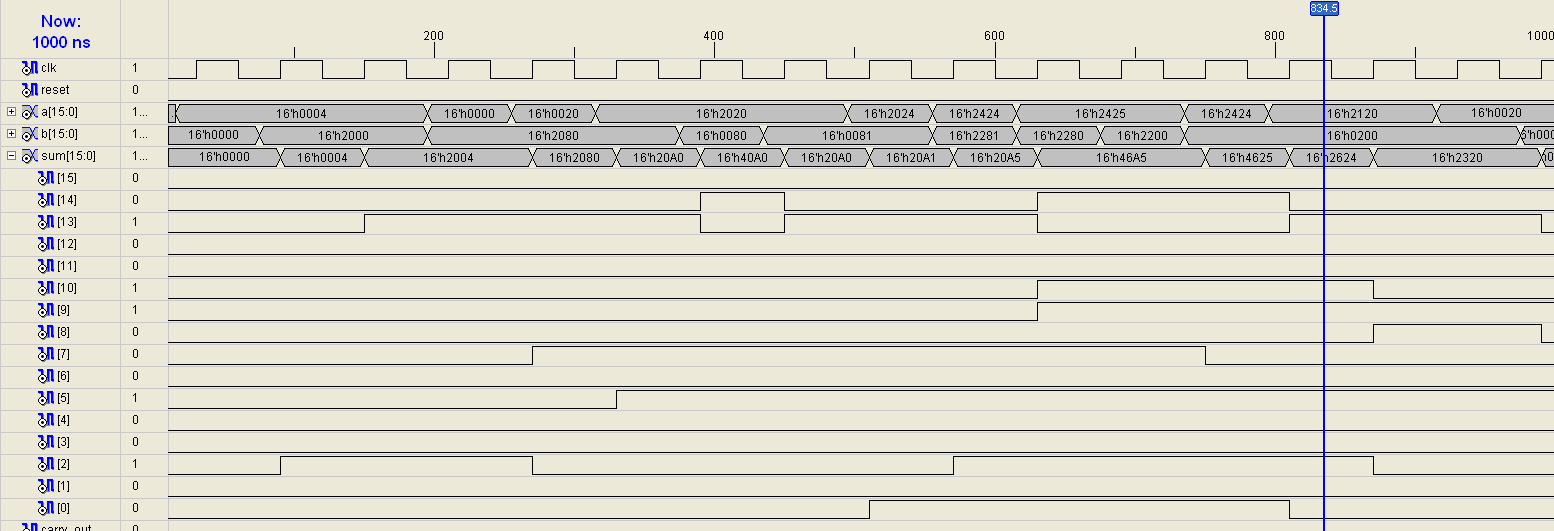
end process;

-- Вывод результата

sum <= reg\_sum; -- Вывод суммы

carry\_out <= carry; -- Вывод переноса

end Behavioral;



Листинг 2

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity p22 is

Port (

clk : in STD\_LOGIC; -- Тактовый сигнал

reset : in STD\_LOGIC; -- Вход сброса

a : in STD\_LOGIC\_VECTOR(15 downto 0); -- Первый вход (16 бит)

b : in STD\_LOGIC\_VECTOR(15 downto 0); -- Второй вход (16 бит)

sum : out STD\_LOGIC\_VECTOR(15 downto 0); -- Выход суммы (16 бит)

carry\_out : out STD\_LOGIC -- Выход переноса

);

end p22;

architecture Behavioral of p22 is

signal reg\_a : STD\_LOGIC\_VECTOR(15 downto 0) := (others => '0'); -- Регистр для первого входа

signal reg\_b : STD\_LOGIC\_VECTOR(15 downto 0) := (others => '0'); -- Регистр для второго входа

signal reg\_sum : STD\_LOGIC\_VECTOR(15 downto 0) := (others => '0'); -- Регистр для суммы

signal carry : STD\_LOGIC := '0'; -- Перенос

begin

-- Процесс для регистрации входов

process(clk, reset)

begin

if reset = '1' then

reg\_a <= (others => '0'); -- Сброс первого регистра

reg\_b <= (others => '0'); -- Сброс второго регистра

reg\_sum <= (others => '0'); -- Сброс регистра суммы

carry <= '0'; -- Сброс переноса

elsif rising\_edge(clk) then

reg\_a <= a; -- Запись первого входа в регистр

reg\_b <= b; -- Запись второго входа в регистр

-- Суммирование с учетом переноса

reg\_sum <= reg\_a + reg\_b; -- Сумма

if (reg\_a + reg\_b) > "1111111111111111" then -- Проверка на перенос

carry <= '1'; -- Устанавливаем перенос

else

carry <= '0'; -- Сбрасываем перенос

end if;

end if;

end process;

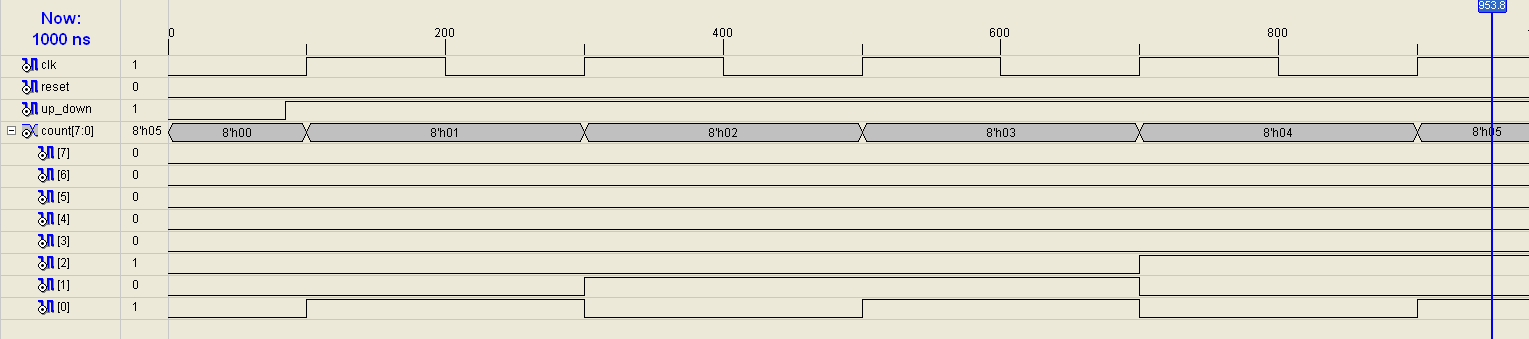
-- Вывод результата

sum <= reg\_sum; -- Вывод суммы

carry\_out <= carry; -- Вывод переноса

end Behavioral;

3. Разработать на языке VHDL и ввести схему реверсивного счетчика. Выполнить функциональное моделирование.   
Вариант – 4. № - 8



Листинг

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all; -- Для работы с арифметическими операциями над векторами

entity p3 is

port (

clk : in std\_logic; -- Тактовый сигнал

reset : in std\_logic; -- Сигнал сброса

up\_down : in std\_logic; -- Управляющий сигнал (1 - увеличивать, 0 - уменьшать)

count : out std\_logic\_vector(7 downto 0) -- Текущие значения счётчика (8 бит)

);

end entity p3;

architecture behavior of p3 is

signal count\_reg : std\_logic\_vector(7 downto 0) := "00000000"; -- Регистр для хранения текущего значения счётчика

begin

-- Процесс для управления счётчиком

process (clk, reset)

begin

if reset = '1' then

count\_reg <= "00000000"; -- Если сигнал сброса активен, сбрасываем счётчик в 0

elsif rising\_edge(clk) then

if up\_down = '1' then

count\_reg <= count\_reg + 1; -- Увеличиваем счётчик, если up\_down = 1

else

count\_reg <= count\_reg - 1; -- Уменьшаем счётчик, если up\_down = 0

end if;

end if;

end process;

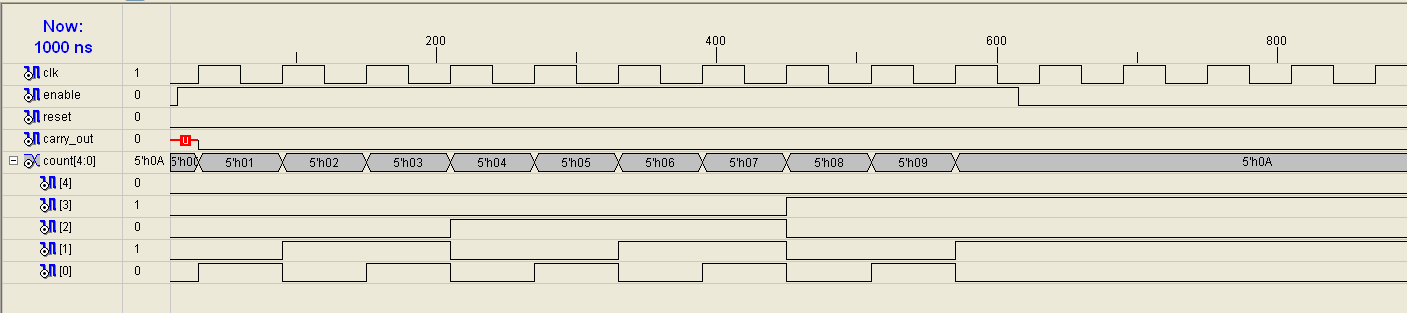
-- Присваиваем текущее значение счётчика на выход

count <= count\_reg;

end architecture behavior;

4. Разработать на языке VHDL схему n-разрядного счетчика с входами разрешения и сброса. Предусмотреть в схеме счетчика выход переноса из старшего разряда. Проверить правильность работы функциональным моделированием. Разработать на языке VHDL схему m-разрядного счетчика, используя ранее созданный n разрядный счетчик в качестве компонента. Значения m и n взять из следующей таблицы.

Вариант – 4. № - 8. M – 10



Листинг 1

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity p4 is

Port (

clk : in STD\_LOGIC; -- Тактовый сигнал

enable : in STD\_LOGIC; -- Вход разрешения

reset : in STD\_LOGIC; -- Вход сброса

carry\_out : out STD\_LOGIC; -- Выход переноса

count : out STD\_LOGIC\_VECTOR(4 downto 0) -- Выход счетчика (5 бит)

);

end p4;

architecture Behavioral of p4 is

signal count\_reg : STD\_LOGIC\_VECTOR(4 downto 0) := (others => '0'); -- Регистры счетчика

begin

process(clk, reset)

begin

if reset = '1' then

count\_reg <= (others => '0'); -- Сброс счетчика

carry\_out <= '0'; -- Сброс переноса

elsif rising\_edge(clk) then

if enable = '1' then

if count\_reg = "11111" then -- Проверка на максимальное значение (31)

count\_reg <= (others => '0'); -- Сбрасываем счетчик

carry\_out <= '1'; -- Устанавливаем перенос

else

count\_reg <= count\_reg + 1; -- Увеличиваем счетчик

carry\_out <= '0'; -- Сбрасываем перенос

end if;

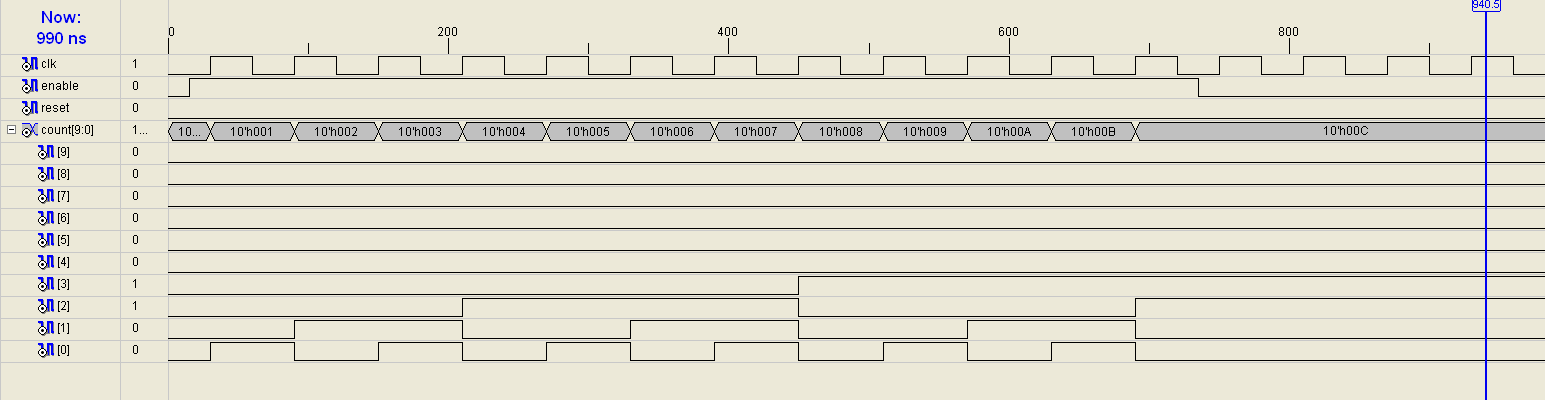
end if;

end if;

end process;

count <= count\_reg; -- Выводим текущее значение счетчика

end Behavioral;



Листинг 2

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity p42 is

Port (

clk : in STD\_LOGIC;

enable : in STD\_LOGIC;

reset : in STD\_LOGIC;

carry\_out : out STD\_LOGIC;

count : out STD\_LOGIC\_VECTOR(9 downto 0)

);

end p42;

architecture Structural of p42 is

component p4 is

Port (

clk : in STD\_LOGIC;

enable : in STD\_LOGIC;

reset : in STD\_LOGIC;

carry\_out : out STD\_LOGIC;

count : out STD\_LOGIC\_VECTOR(4 downto 0)

);

end component;

signal count\_low : STD\_LOGIC\_VECTOR(4 downto 0);

signal count\_high : STD\_LOGIC\_VECTOR(4 downto 0);

signal carry\_low : STD\_LOGIC;

begin

-- Инстанцирование младшего 5-разрядного счетчика

low\_counter : p4

port map (

clk => clk,

enable => enable,

reset => reset,

carry\_out => carry\_low,

count => count\_low

);

-- Инстанцирование старшего 5-разрядного счетчика

high\_counter : p4

port map (

clk => clk,

enable => carry\_low, -- Старший счетчик активируется, когда младший достиг максимума

reset => reset,

carry\_out => carry\_out,

count => count\_high

);

-- Объединение выходов счетчиков

count <= count\_high & count\_low; -- Соединяем старший и младший счетчики

end Structural;

5. Разработать на языке VHDL схему сумматора с регистром на выходе, сумматор выполняет суммирование числа x, поступающего с тумблеров (контакты R17, N17, L13, L14, K17, K18, H18 и G18) и числа 3\*i, где i – номер варианта. Число 3\*i задать двоичной константой, например, если 3\*i = 12, двоичная константа равна “00001100”. Выходы сумматора подключить к светодиодам (контакты R4, F4, P15, E17, K14, K15, J15 и J14). Синхросигнал ввести с контакта B8.

Выполнить моделирование работы схемы, подавая на вход x последовательно числа i, i + 7, 15 ‑ i, 15 ‑ i + 3.

Выполнить реализацию схемы и получить файл конфигурации. Записать файл конфигурации в ПЛИС на плате Nexys-2 и, последовательно подавая с тумблеров числа i, i + 7, 15 ‑ i, 15 ‑ i + 3, проверить правильность работы. Сравнить результаты работы схемы с результатами моделирования.

В этом пункте и в следующих используется ПЛИС XC3S500E‑4FG320C. В отчет по данному пункту необходимо включить подробное описание работы схемы на плате для тех же значениях, что и при моделировании.

1. Файл описания схемы (p5):

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity p5 is

Port (

x : in STD\_LOGIC\_VECTOR (7 downto 0); -- Входное число x с тумблеров

clk : in STD\_LOGIC; -- Сигнал синхронизации

q : out STD\_LOGIC\_VECTOR (7 downto 0) -- Выход, подключаемый к светодиодам

);

end p5;

architecture Behavioral of p5 is

-- Константа 3\*i (где i = 4) в двоичном виде

constant CONST\_3I : STD\_LOGIC\_VECTOR (7 downto 0) := "00001100"; -- 12

signal sum\_reg : STD\_LOGIC\_VECTOR (7 downto 0); -- Регистр для хранения результата

begin

process(clk)

begin

if rising\_edge(clk) then

-- Суммируем x и CONST\_3I и сохраняем в регистре sum\_reg

sum\_reg <= x + CONST\_3I;

end if;

end process;

-- Присваиваем выходному сигналу значение регистра

q <= sum\_reg;

end Behavioral;

2. Файл ограничений (ic5.ucf):

# Ограничения для входов (тумблеры)

NET "x[0]" LOC = "R17";

NET "x[1]" LOC = "N17";

NET "x[2]" LOC = "L13";

NET "x[3]" LOC = "L14";

NET "x[4]" LOC = "K17";

NET "x[5]" LOC = "K18";

NET "x[6]" LOC = "H18";

NET "x[7]" LOC = "G18";

# Ограничения для выходов (светодиоды)

NET "q[0]" LOC = "R4";

NET "q[1]" LOC = "F4";

NET "q[2]" LOC = "P15";

NET "q[3]" LOC = "E17";

NET "q[4]" LOC = "K14";

NET "q[5]" LOC = "K15";

NET "q[6]" LOC = "J15";

NET "q[7]" LOC = "J14";

# Ограничения для тактового сигнала

NET "clk" LOC = "B8";

NET "clk" TNM\_NET = clk;

TIMESPEC TS\_clk = PERIOD "clk" 20 ns HIGH 50%;

6 Разработать схему 31-разрядного счетчика.

1. Файл описания 31-разрядного счетчика (p6.vhd):

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity p6 is

Port (

clk : in STD\_LOGIC; -- Синхросигнал на входе

led : out STD\_LOGIC\_VECTOR (7 downto 0) -- 8 старших разрядов на светодиоды

);

end p6;

architecture Behavioral of p6 is

signal counter : STD\_LOGIC\_VECTOR (30 downto 0) := (others => '0'); -- 31-разрядный счетчик

begin

process(clk)

begin

if rising\_edge(clk) then

counter <= counter + 1; -- Инкремент счетчика

end if;

end process;

-- Присваиваем 8 старших разрядов выходам для наблюдения

led <= counter(30 downto 23);

end Behavioral;

2. Файл ограничений (ic6.ucf):

# Ограничения для выходов (старшие разряды на светодиоды)

NET "led[0]" LOC = "R4";

NET "led[1]" LOC = "F4";

NET "led[2]" LOC = "P15";

NET "led[3]" LOC = "E17";

NET "led[4]" LOC = "K14";

NET "led[5]" LOC = "K15";

NET "led[6]" LOC = "J15";

NET "led[7]" LOC = "J14";

# Ограничение для тактового сигнала

NET "clk" LOC = "B8";

7. Разработать схему «бегущие огни» и реализовать ее на Nexys-2 (Т – период работы схемы).



1. Файл описания схемы бегущих огней (p7.vhd):

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity p7 Lights is

Port (

clk : in STD\_LOGIC; -- Синхросигнал на входе

reset : in STD\_LOGIC; -- Сигнал сброса

leds : out STD\_LOGIC\_VECTOR (5 downto 0) -- 6 светодиодов

);

end p7;

architecture Behavioral of p7 is

signal counter : STD\_LOGIC\_VECTOR (2 downto 0) := "000"; -- 3-битный счетчик для кода 3

begin

process(clk, reset)

begin

if reset = '1' then

counter <= "000"; -- Сброс счетчика

elsif rising\_edge(clk) then

counter <= counter + 1; -- Инкремент счетчика

end if;

end process;

-- Генерация бегущих огней с помощью сдвига активного бита

leds <= "000001" when counter = "000" else

"000010" when counter = "001" else

"000100" when counter = "010" else

"001000" when counter = "011" else

"010000" when counter = "100" else

"100000" when counter = "101" else

"000001"; -- Начало цикла для 6 состояний

end Behavioral;

2. Файл ограничений (ic7.ucf):

# Ограничения для светодиодов (6 выводов для "бегущих огней")

NET "leds[0]" LOC = "R4";

NET "leds[1]" LOC = "F4";

NET "leds[2]" LOC = "P15";

NET "leds[3]" LOC = "E17";

NET "leds[4]" LOC = "K14";

NET "leds[5]" LOC = "K15";

# Ограничение для тактового сигнала

NET "clk" LOC = "B8";

**Вывод**

Изучили принципы разработки схем на языке VHDL