Цифровые устройства и микропроцессоры I часть

Лекция 15

Лектор: Богаченков Алексей Николаевич

e-mail: microproc@mail.ru

Тема лекции:

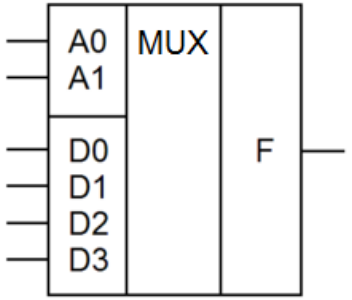
Примеры описаний комбинационных и последовательностных устройств на языке VHDL

Все примеры подготовлены в среде разработки Xilinx ISE Design Suite 14.7

Селективная установка значений сигналов

1. Операторы when … else

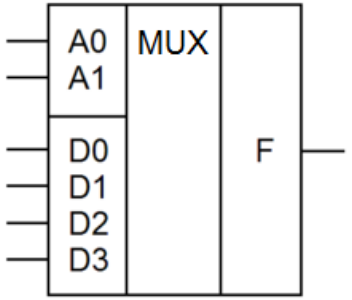
```
library ieee;
use ieee.std_logic_1164.all;

entity mux_4x1 is
    port ( A : in  std_logic_vector(1 downto 0);
           D : in  std_logic_vector(3 downto 0);
           F : out std_logic);
end mux_4x1;

----------------ВАРИАНТ 1  (when...else)-----------------------

architecture mux_arch of mux_4x1 is
begin

F <= D(0) when A = "00" else
     D(1) when A = "01" else
     D(2) when A = "10" else
     D(3);

end mux_arch;
```



Селективная установка значений сигналов

2. Операторы select … when

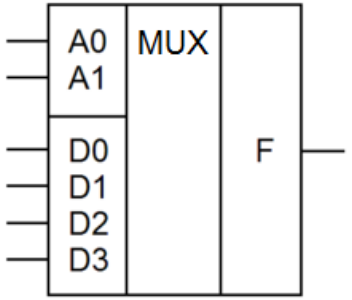
```
library ieee;
use ieee.std_logic_1164.all;

entity mux_4x1 is
    port ( A : in  std_logic_vector(1 downto 0);
           D : in  std_logic_vector(3 downto 0);
           F : out std_logic);
end mux_4x1;

----------------ВАРИАНТ 2  (select..when)----------------------

architecture mux_arch2 of mux_4x1 is
begin

with A select
F <= D(0) when "00",
     D(1) when "01",
     D(2) when "10",
     D(3) when others;

end mux_arch2;
```



Селективная установка значений сигналов

3. Операторы if … then … else

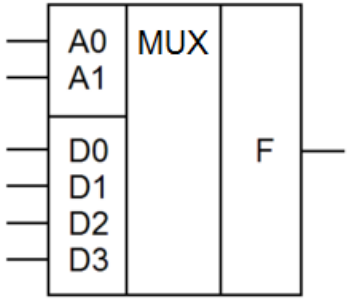
```
library ieee;
use ieee.std_logic_1164.all;

entity mux_4x1 is
    port ( A : in  std_logic_vector(1 downto 0);
           D : in  std_logic_vector(3 downto 0);
           F : out std_logic);
end mux_4x1;

----------------ВАРИАНТ 3  (if...then...else + "process")-------

architecture mux_arch3 of mux_4x1 is
begin

process (A, D)                  -- Оператор "process" со списком чувствительности
begin
   if    A = "00" then F <= D(0);
   elsif A = "01" then F <= D(1);
   elsif A = "10" then F <= D(2);
   else F <= D(3);
   end if;
end process;

end mux_arch3;
```



Селективная установка значений сигналов

4. Операторы case … when

```
library ieee;
use ieee.std_logic_1164.all;

entity mux_4x1 is
    port ( A : in  std_logic_vector(1 downto 0);
           D : in  std_logic_vector(3 downto 0);
           F : out std_logic);
end mux_4x1;

----------------ВАРИАНТ 4  (case...when + "process")-----------

architecture mux_arch4 of mux_4x1 is
begin

process (A, D)
begin
   case conv_integer(A) is
      when  0     => F <= D(0);
      when  1     => F <= D(1);
      when  2     => F <= D(2);
      when  3     => F <= D(3);
      when others => F <= D(0);  -- Хотя данная строка "лишняя",
   end case;                     --  рекомендуется ее всегда включать
end process;

end mux_arch4;
```

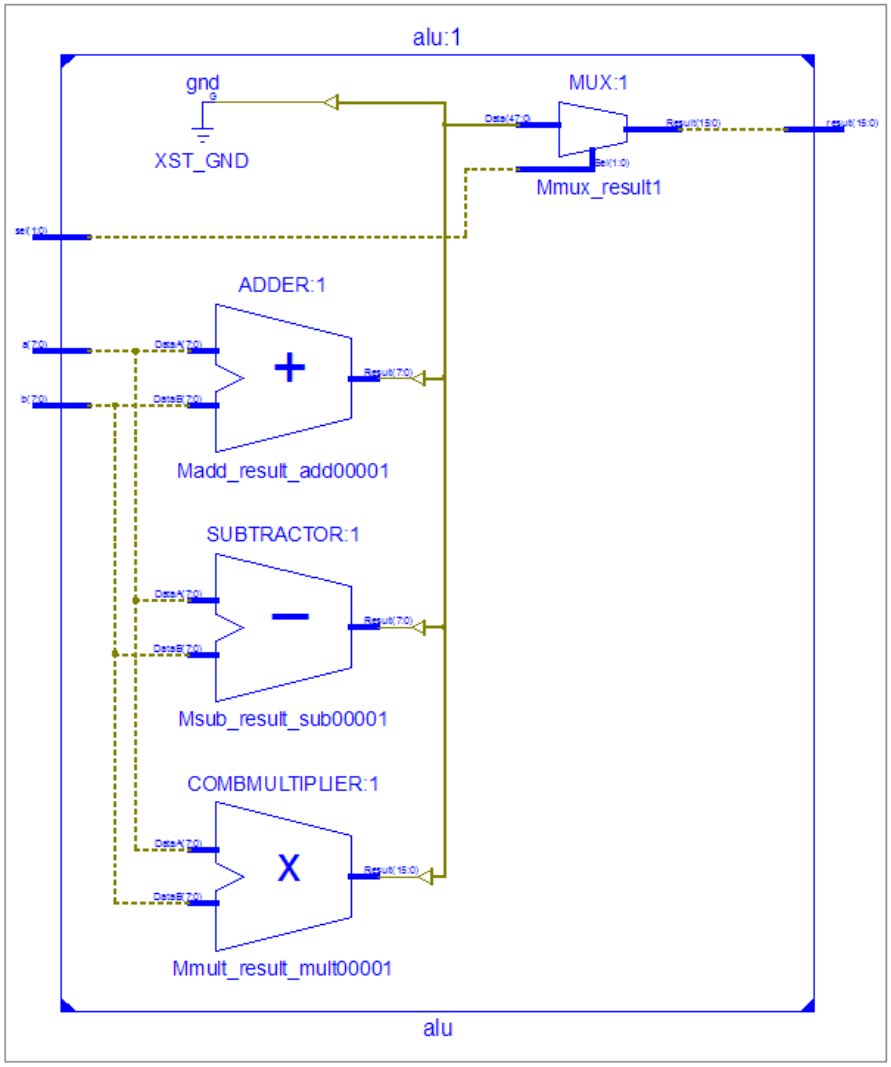


Арифметические операции

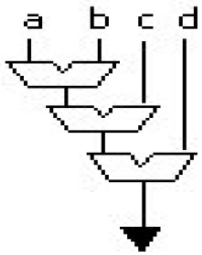
```
library ieee;                                   -- Объявление библиотек
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity alu is                                   -- Начало описания объекта, его имя
   generic
   (
      DATA_WIDTH : natural := 8                 -- Объявление настроечного параметра
   );
   port
   (  a, b      : in  std_logic_vector(DATA_WIDTH-1 downto 0);   -- Вх.операнды
      result    : out std_logic_vector(DATA_WIDTH*2-1 downto 0); -- Результат
      sel       : in  std_logic_vector(1 downto 0)               -- Тип операции
   );
end entity;

architecture rtl of alu is
begin

process (a, b, sel)
begin
   if sel = "00" then
      result <= ext(a + b, DATA_WIDTH*2); -- Сложение, дополнение результата
   elsif sel = "01" then                  --  старшими нулями
      result <= sxt(a - b, DATA_WIDTH*2); -- Вычитание, знаковое расширение
   elsif sel = "10" then
      result <= a * b;                    -- Умножение
   else
      result <= (others => '-');          -- Неопределенный результат
   end if;

end process;

end rtl;
```

Арифметические операции

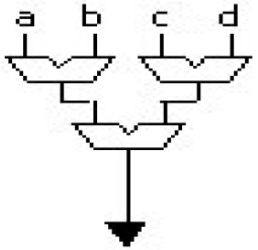
Результат синтеза



Пример, как синтезируемая структура зависит от выражения

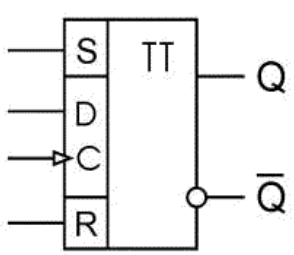


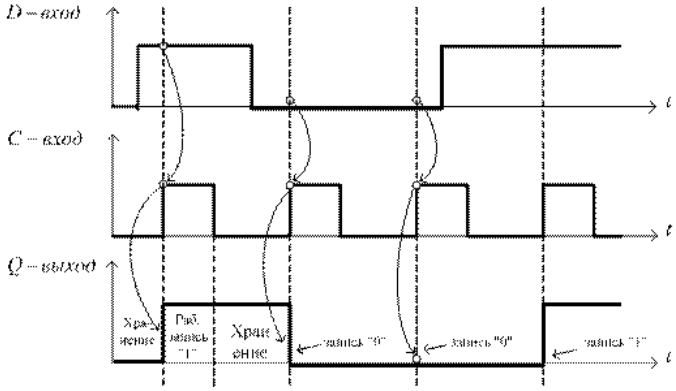
a+b+c+d



(a+b)+(c+d)

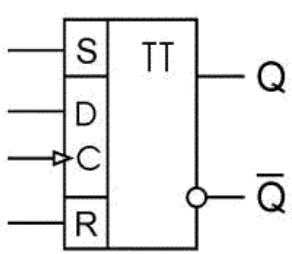
Последовательностные схемы. Триггер, защелка

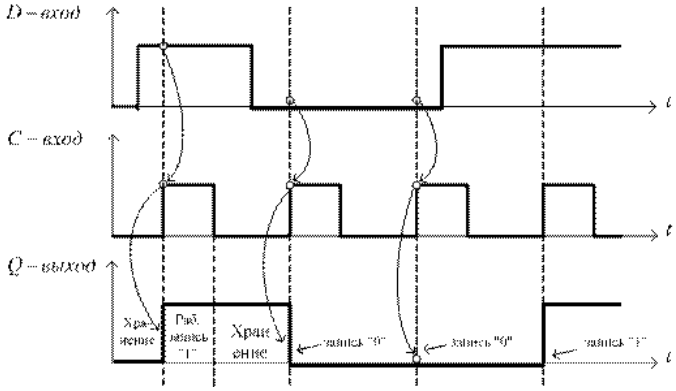





```
library ieee;
use ieee.std_logic_1164.all;

entity trig is
   port
   (  C       : in  std_logic;      -- Вход тактирования (записи)
      S, R    : in  std_logic;      -- Входы установки и сброса
      D       : in  std_logic;      -- Вход данных
      Q, QN   : out std_logic       -- Прямой и инверсный выходы данных
   );
end entity;

-------------------------------------------------------------
-- ЗАЩЕЛКА (сигнал записи с управлением по уровню)
-------------------------------------------------------------
architecture latch_a of latch is
begin

process (C, D)
  begin
    if (C = '1') then
       Q <= D; QN <= not D;         -- Задание поведения при C = 1,
    end if;                         -- не задано для C = 0 (это плохо!)
end process;

end latch_a;
```

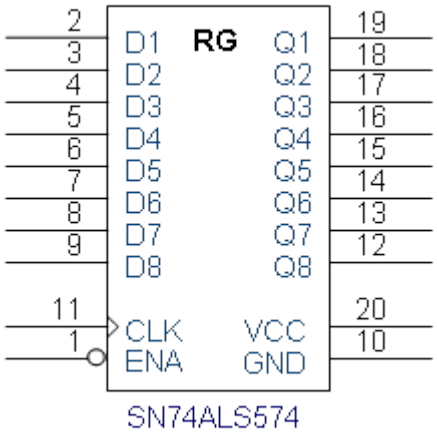
Последовательностные схемы. Триггер, защелка






```
library ieee;
use ieee.std_logic_1164.all;

entity trig is
   port
   (  C       : in  std_logic;      -- Вход тактирования (записи)
      S, R    : in  std_logic;      -- Входы установки и сброса
      D       : in  std_logic;      -- Вход данных
      Q, QN   : out std_logic       -- Прямой и инверсный выходы данных
   );
end entity;

-------------------------------------------------------------
-- ТРИГГЕР (с переключением по фронту и асинхронным сбросом)
-------------------------------------------------------------
architecture trig_a of trig is
begin

process (R, C)
  begin
    if  R = '1'  then  Q <= '0'; QN <= '1';
    elsif  C'event  and  C = '1'  then
       Q <= D; QN <= not D;
    end if;
end process;

end trig_a;
```

Последовательностные схемы. Регистр

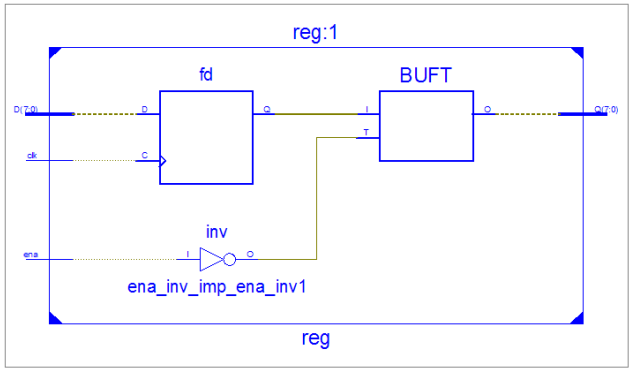



```
library ieee;
use ieee.std_logic_1164.all;

entity reg is
   port
   (  clk  : in  std_logic;                      -- Вход тактирования (записи)
      ena  : in  std_logic;                      -- Вход разрешения работы выхода
      D    : in  std_logic_vector(7 downto 0); -- Входные данные (8 бит)
      Q    : out std_logic_vector(7 downto 0)  -- Выходные данные (8 бит)
   );
end entity;

architecture reg_a of reg is
                                              -- Внутренний регистр памяти
signal mem : std_logic_vector(7 downto 0) := (others => '0');
begin
                                              -- Разрешение работы выходного каскада
Q <= mem when ena = '0' else (others => 'Z');

process (clk)
begin
   if rising_edge(clk) then mem <= D;      -- Защелкивание данных по фронту CLK
   end if;
end process;

end reg_a;
```

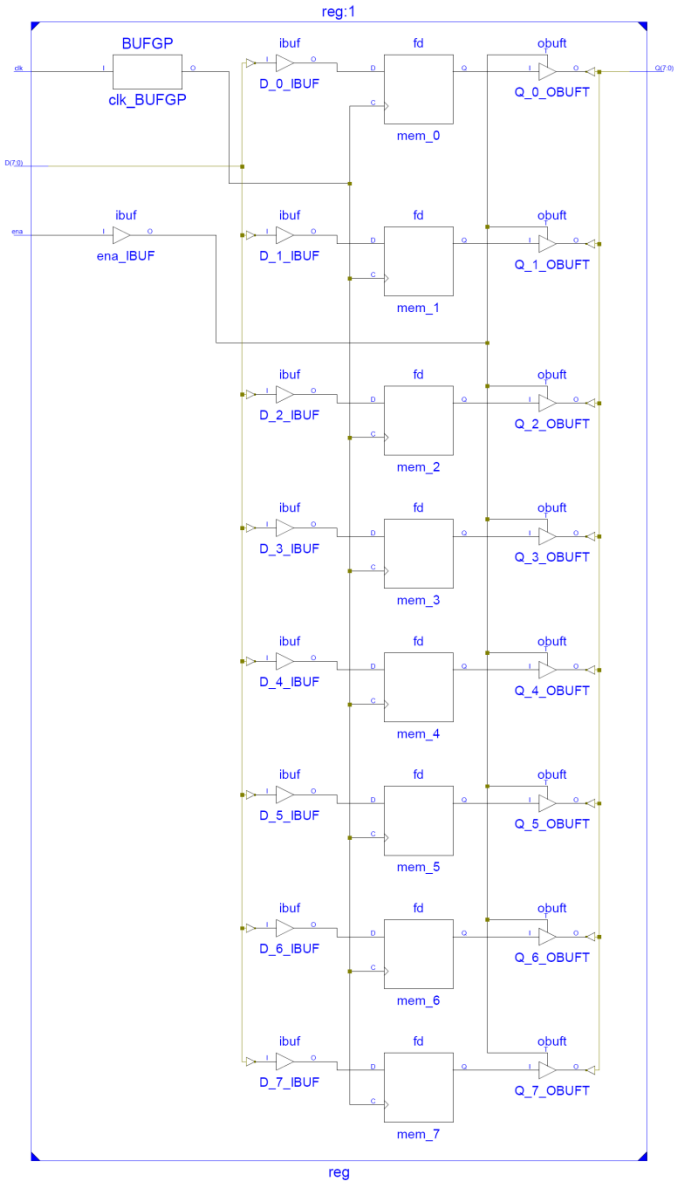
Последовательностные схемы. Регистр

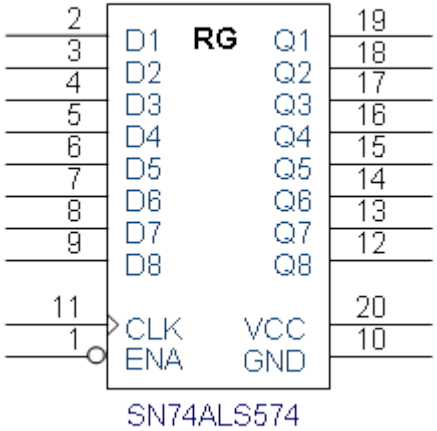
Результат синтеза

RTL схема
(на уровне регистровых передач)

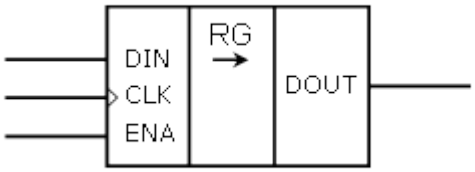


Технолог. схема (на уровне ячеек)





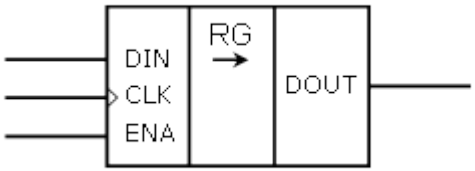
Последовательный регистр



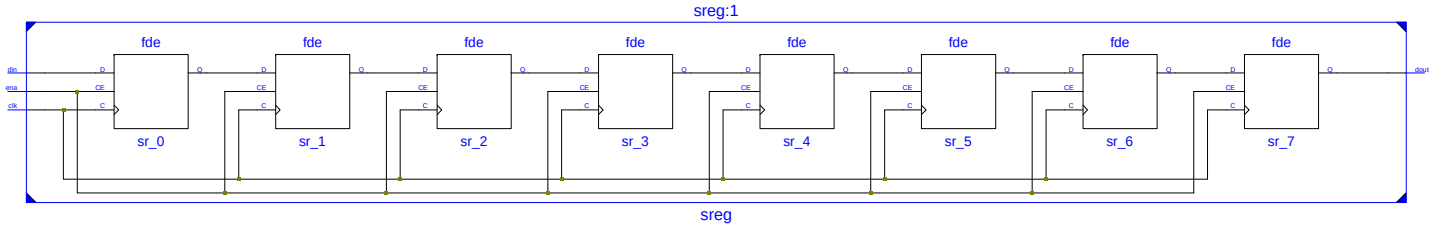

```
library ieee;
use ieee.std_logic_1164.all;

entity sreg is
   port
   (  clk  : in  std_logic;                  -- Вход тактирования (сдвига)
      ena  : in  std_logic;                  -- Вход разрешения сдвига
      din  : in  std_logic;                  -- Входные данные
      dout : out std_logic                   -- Выходные данные
   );
end entity;

architecture Behavioral of sreg is
                                             -- Внутренний регистр, изначально
                                             -- инициализируется нулями
signal sr : std_logic_vector(7 downto 0) := x"00";

begin

   process (clk)
   begin
      if rising_edge(clk) and ena = '1' then
         sr <= sr(6 downto 0) & din;         -- Сдвиг влево, вдвижение входных
      end if;                                -- данных в младший разряд
   end process;

   dout <= sr(7);                            -- Выход со старшего разряда

end Behavioral;
```

Последовательный регистр

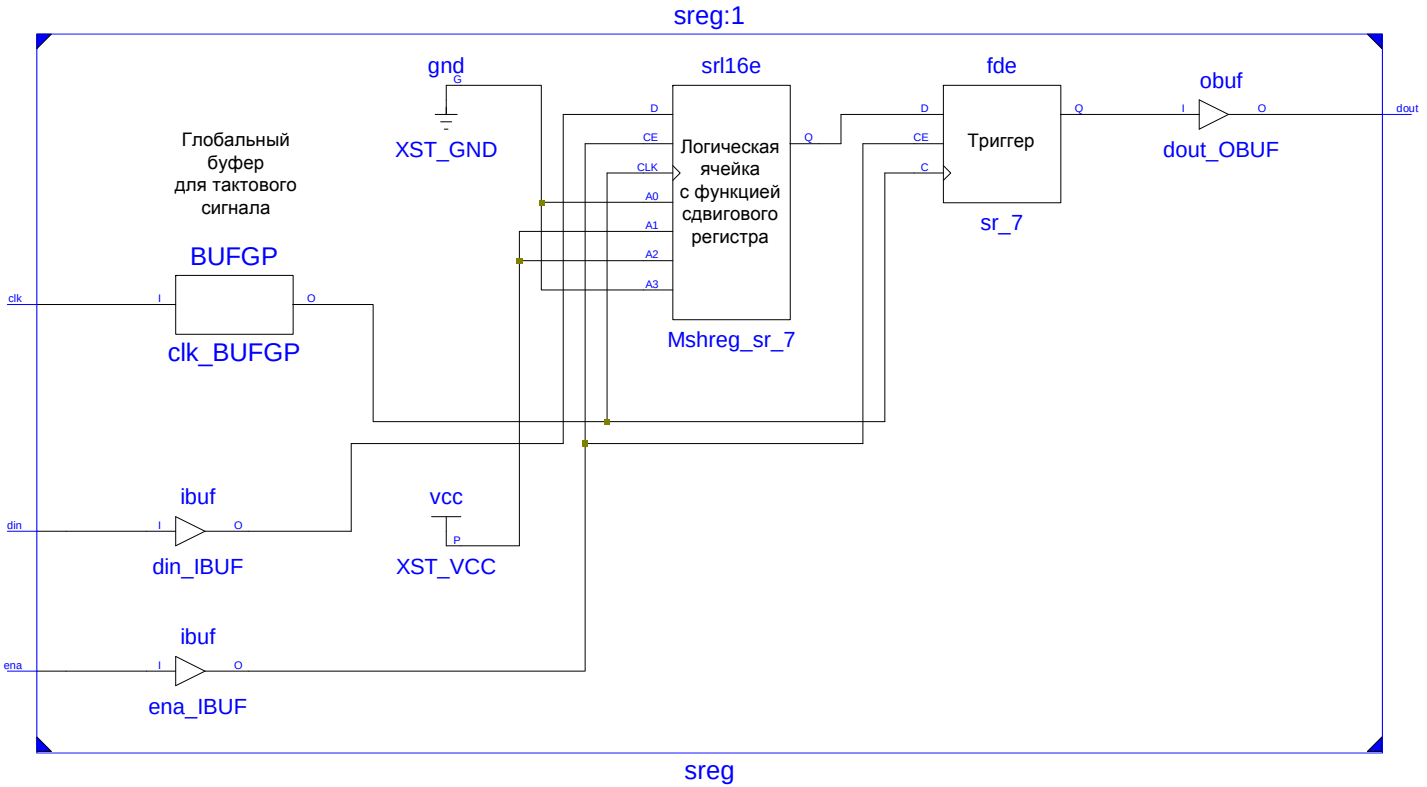
Результат синтеза



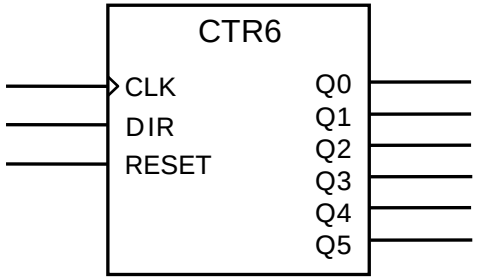
RTL схема (на уровне регистровых передач)



Технологическая схема (на уровне логических ячеек)



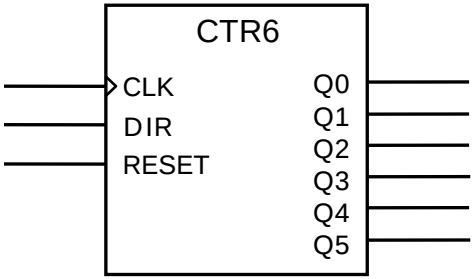
Реверсивный счетчик



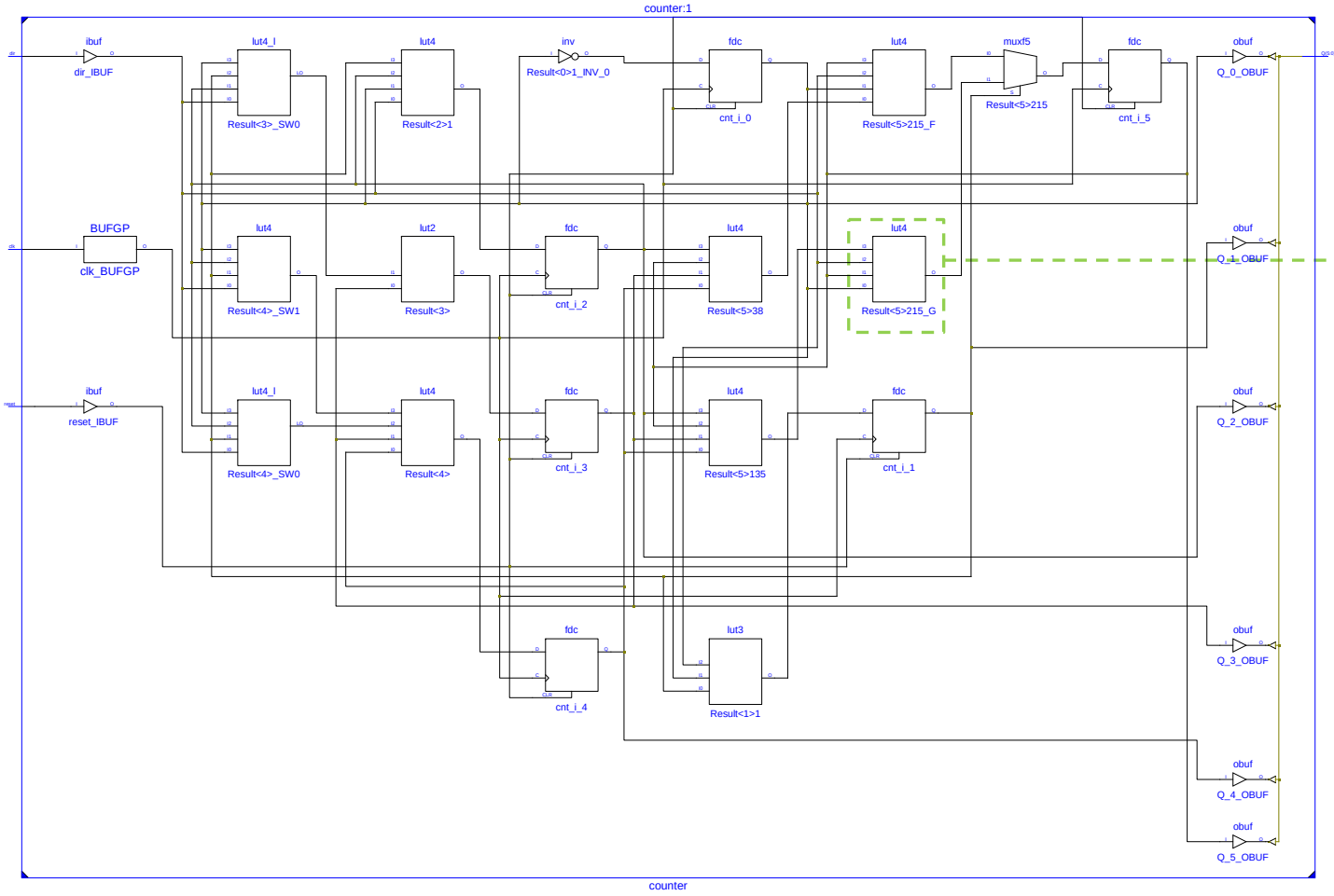

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity counter is
   port
   (  clk   : in  std_logic;                    -- Вход тактирования
      reset : in  std_logic;                    -- Вход сброса (установки в 0)
      dir   : in  std_logic;                    -- Вход задания направления счета
      Q     : out std_logic_vector(5 downto 0) -- Выходные данные (6 бит)
   );
end entity;

architecture Behavioral of counter is

-- Создание внутренней копии состояния счетчика, т.к. непосредственно с выходным
-- кодом Q не допускаются операции чтения / преобразования
signal cnt_i : std_logic_vector(5 downto 0) := (others => '0');

begin

   process (clk, reset)
   begin
      if reset = '1' then                       -- Проверка сигнала сброса
         cnt_i <= (others => '0');              -- Асинхронный сброс
      elsif rising_edge(clk) then               -- Следующие операции - синхронные
         if dir = '1' then                      -- Проверка направления
            cnt_i <= cnt_i + 1;                 -- Суммирование
         else
            cnt_i <= cnt_i - 1;                 -- Вычитание
         end if;
      end if;
   end process;

   Q <= cnt_i;                                  -- Копирование на выход

end Behavioral;
```

Реверсивный счетчик

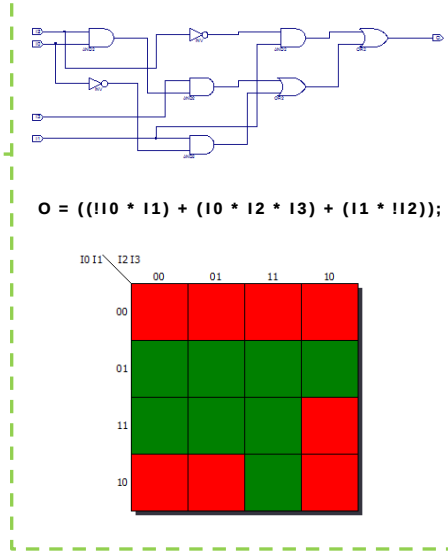
Результат синтеза



Технологическая схема (на уровне логических ячеек)

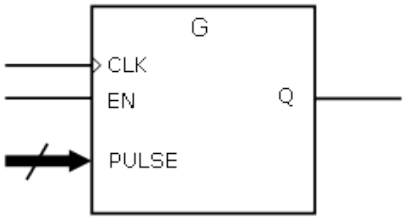


Одна из логических ячеек



Генератор ШИМ сигнала

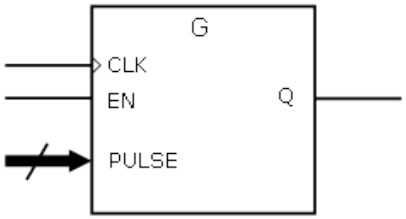
Описание устройства

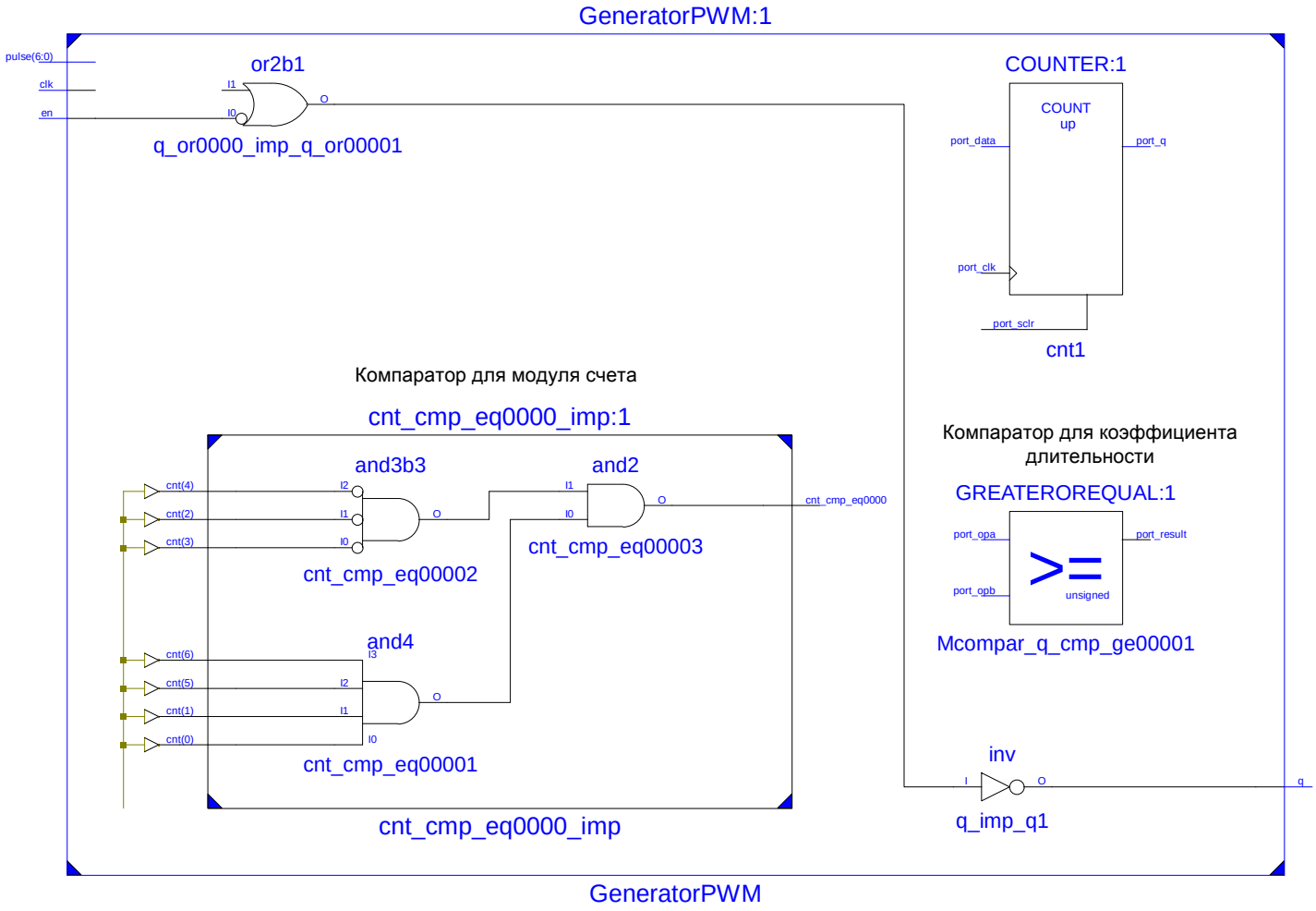



```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity GeneratorPWM is
   port
   (  clk   : in  std_logic;                  -- Тактовая частота
      en    : in  std_logic;                  -- Разрешение генерации
      pulse : in  integer range 0 to 100;     -- Коэффициент длительности
      q     : out std_logic                   -- Выходной сигнал
   );
end entity;

architecture Behavioral of GeneratorPWM is

signal cnt : integer range 0 to 99 := 0;     -- Счетчик с модулем счета 100

begin

   process (clk)
   begin
      if rising_edge(clk) then               -- Операции по фронту тактового сигнала
         if cnt = 99 then                    -- Если в счетчике максимум,
            cnt <= 0;                        -- его сброс
         else
            cnt <= cnt + 1;                  -- Текущий инкремент счетчика
         end if;
      end if;
   end process;

   -- Формирование выходного сигнала:
   --   1, если в счетчике < pulse,
   --   0, если в счетчике >= pulse или запрет генерации
   q <= '0' when (cnt >= pulse or en = '0') else '1';

end Behavioral;
```

Генератор ШИМ сигнала

Результат синтеза

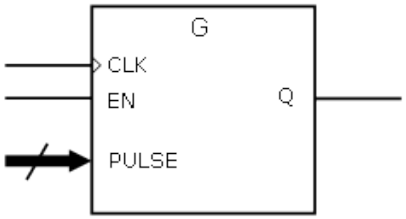




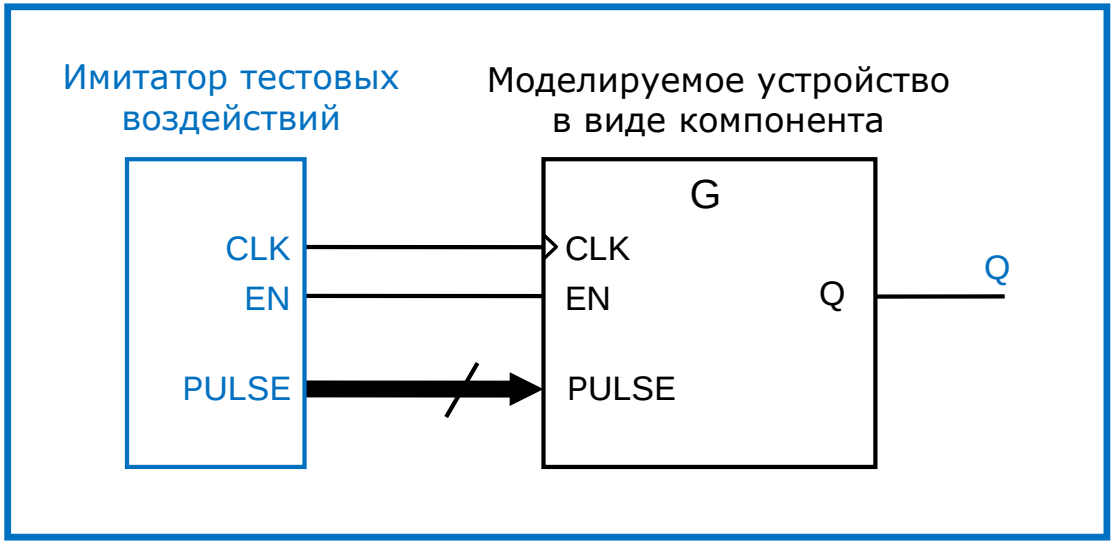
Генератор ШИМ сигнала

Функциональное моделирование (симуляция)

Структура тестового модуля



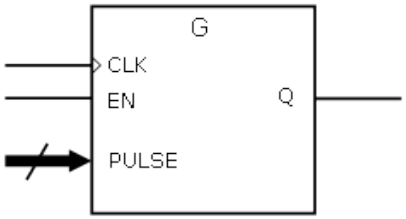
Test Bench



Генератор ШИМ сигнала

Функциональное моделирование (симуляция)

Описание тестового модуля. Начало

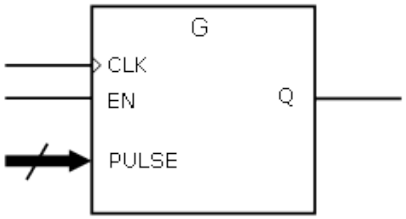



```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

-- Формальное объявление интерфейса тестовой оболочки
ENTITY top_tb IS
END top_tb;

ARCHITECTURE behavior OF top_tb IS

   -- Объявление моделируемого устройства как компонента UUT (Unit Under Test)
   COMPONENT GeneratorPWM
   PORT(
         clk   : IN  std_logic;
         en    : IN  std_logic;
         pulse : IN  integer;
         q     : OUT std_logic
       );
   END COMPONENT;

   -- Имитируемые входные сигналы
   signal clk : std_logic := '0';
   signal en : std_logic := '0';
   signal pulse : integer range 0 to 100 := 0;

   -- Имитируемый выходной сигнал
   signal q : std_logic;

   -- Период тактового сигнала (1 мкс - 1 МГц)
   constant clk_period : time := 1000 ns;
```

Генератор ШИМ сигнала

Функциональное моделирование (симуляция)

Описание тестового модуля. Окончание

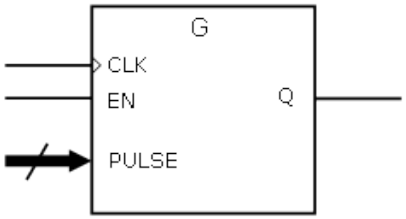



```
BEGIN

   -- Связи с тестируемым компонентом
   uut: GeneratorPWM PORT MAP (
          clk   => clk,
          en    => en,
          pulse => pulse,
          q     => q
        );

   -- Процесс симуляции тактового сигнала
   clk_process: process
   begin
      clk <= '1';               -- Лог. 1
      wait for clk_period/2;    -- Задержка на полпериода
      clk <= '0';               -- Лог. 0
      wait for clk_period/2;    -- Задержка на полпериода
   end process;

   -- Процесс симуляции других входных воздействий
   stim_proc: process
   begin

      pulse <= 0;               -- Начальное значение коэффициента
      en <= '1';                -- Разрешение генерации

      for i in 1 to 10 loop     -- Цикл по изменению коэффициента
         wait for clk_period*500;
         pulse <= pulse + 10;
      end loop;

      wait;                     -- Остановка
   end process;

END;
```

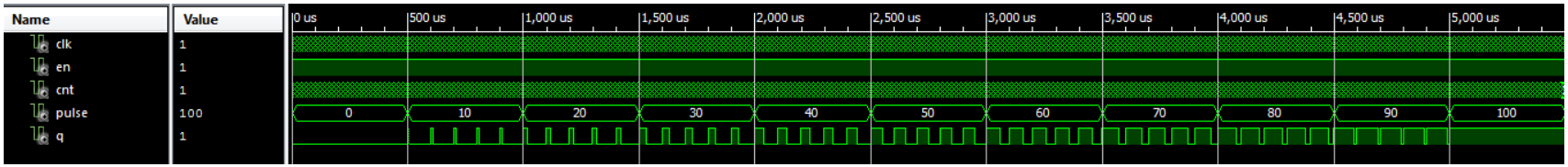
Генератор ШИМ сигнала

Функциональное моделирование (симуляция)

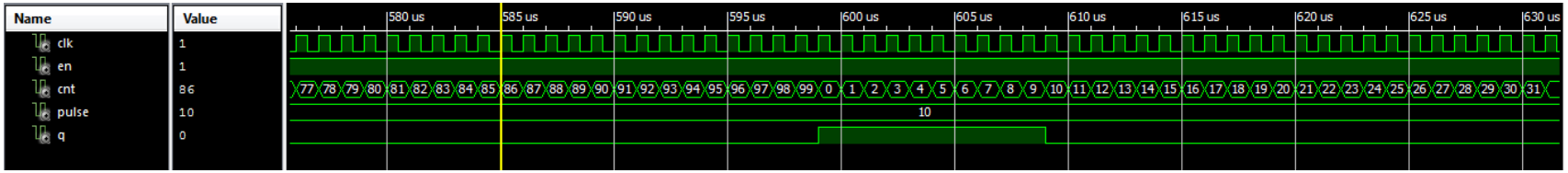
Скриншоты процесса



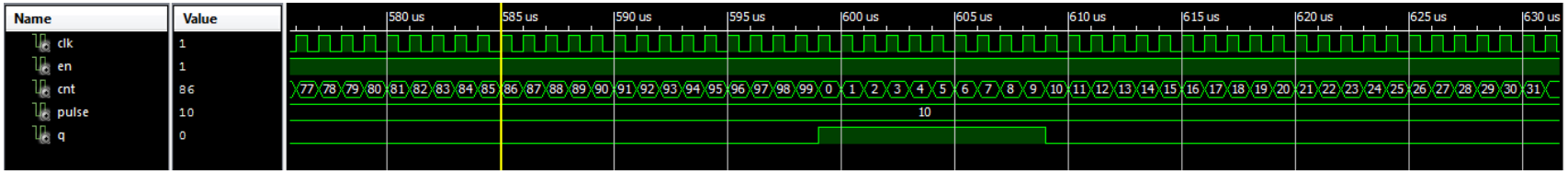
Полный цикл изменения коэффициента: 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100



Выходной импульс для коэффициента 10



Выходной импульс для коэффициента 20



Выходной импульс для коэффициента 90