Язык VHDL. Типы данных.

Типы данных: Boolean, Integer, STD_LOGIC и др.

Тип данных Boolean имеет 2 состояния: 0 - ноль, 1 - единица.

Один из основных типов STD_LOGIC имеет 9 состояний:

0 - ноль, 1 - единица, Z - третье состояние,

X - неопределенное состояние и др.

Алфавит языка VHDL

A, B, . . . , Z, a, b, . . . , z, 0, 1, . . . , 9, _ .
Прописные и строчные буквы не различаются.
Признаком комментария являются два символа тире («--»).

Логические операторы

AND, OR, NAND, NOR, XOR, NOT.

Оператор конкатенации (объединения) - &.

Объекты языка VHDL

constant (константа), variable (переменная), signal (сигнал).

Декларация константы:
constant MAX1 :   integer := 10;

Декларация сигнала:
signal x1: std_logic;
signal ct: std_logic_vector(3 downto 0):=x"0";

Для указания системы счисления для констант могут быть применены спецификаторы:

- B – двоичная система счисления, например b"0011"
- O– восьмеричная система счисления, например O"3760"
- H – шестнадцатеричная система, например h"f6a0".

Операторы языка VHDL

Последовательные операторы: if, case и др.

Параллельные операторы: process и др.

Оператор присваивания:

```
x1 <= '1';
```

оператор: if

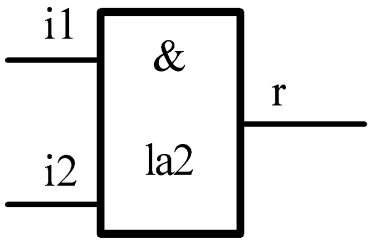
```
if  r = '1' then   -- пример 1
    ct <= 0;       -- r = 1
elsif  e = '1'  then
    ct <= ct + 1; -- r = 0,  e = 1
else
    ct <= ct;      -- r = 0,  e = 0
end if;
```

```
if e='1' then    -- пример 2
    ct <= ct+1;
end if;
```

```
if e='1' then   -- пример 3
    ct <= ct+1;
else
    ct <= ct;
end if;
```

Схема вентиля на языке VHDL

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

entity la2 is -- внешние контакты
  Port (i1: in std_logic;

        i2: in std_logic;
         r: out std_logic);
end la2;

Architecture Behavioral of la2 is --внутренние схемы
begin                                -- и связи (тело)
   r <= i1 and i2;             -- оператор присваивания
end Behavioral;
```



Вентиль 2И

Комбинационная схема на языке VHDL

Потоковое описание

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

entity p02 is -- внешние контакты
  Port (x1: in std_logic;
        x2: in std_logic;
        x3: in std_logic;
         y: out std_logic);
end p02;

Architecture Behavioral of p02 is --внутренние схемы
begin                             -- и связи (тело)
   y <= (x1 and x2) or (x2 and x3);
end Behavioral;
```

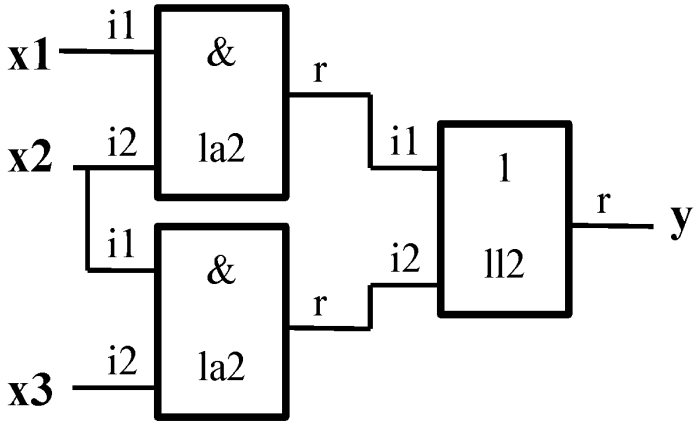
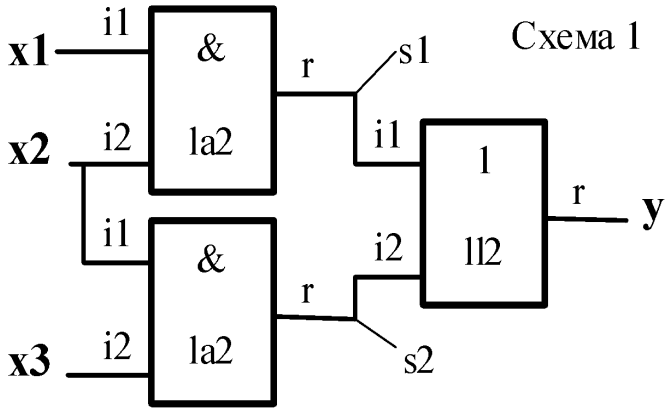


Схема 1

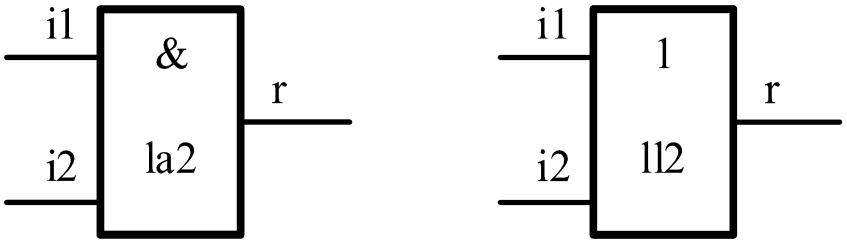
Структурное описание, главный модуль.

```
library IEEE;
  ...
entity sh2 is
 Port (x1,x2,x3:in std_logic;
       y : out std_logic);
end sh2;
```




```
architecture Structura of sh2 is
 component la2 port   -- декларация компонента la2
  (i1,i2: in std_logic; r: out std_logic);
 end component;
 component ll2 port   -- декларация компонента ll2
  (i1,i2: in std_logic; r: out std_logic);
 end component; -- продолжение на следующем слайде.
```

Компоненты:



Структурное описание, продолжение

```
signal S1, S2 :  std_logic;
begin
 u1: la2 port map(x1,x2,S1);   -- конкретизация
 u2: la2 port map(x2,x3,S2);   -- компонентов
 u3: ll2 port map(s1,s2,y);
 end Structura;
-- la2, ll2 – тип компонента,
-- u1, u2, u3 – имена узлов
--                  на схеме.
```

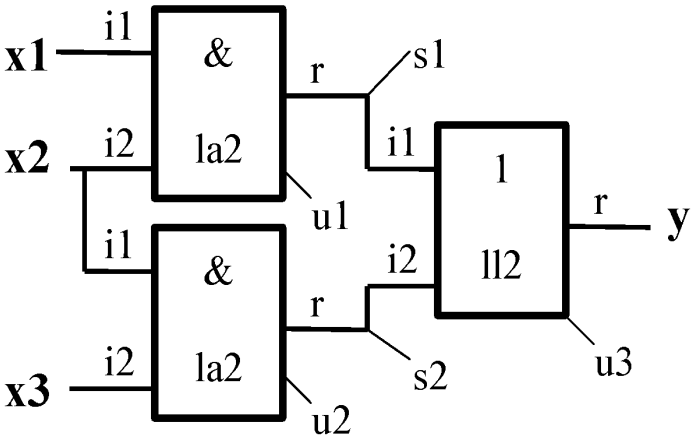
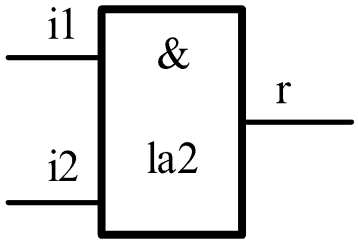
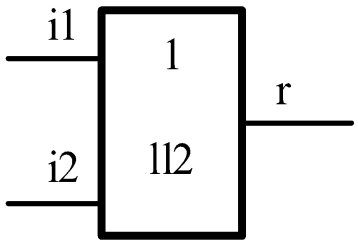


Схема 1

Компоненты



Вентиль 2И



Вентиль 2ИЛИ

Структурное и потоковое описание

```
library IEEE;  -- структурное и
   ...         -- потоковое описание
entity sh2 is -- в одном модуле
 Port (x1,x2,x3:in std_logic;
         y : out std_logic);
end sh2;
```

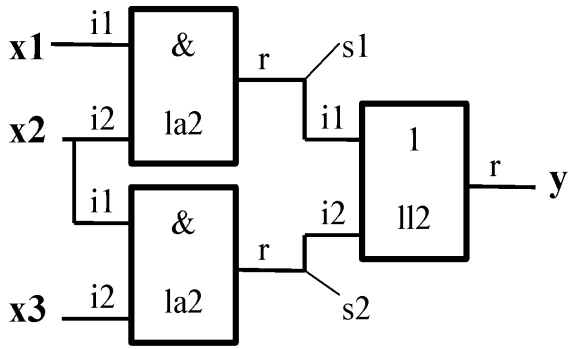
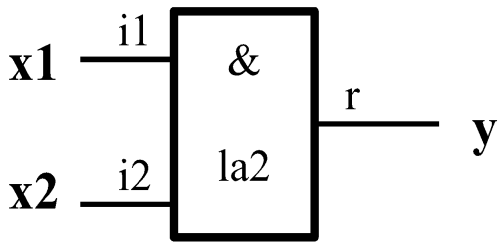


Схема 1

```
architecture Structura of sh2 is
 component la2 port   -- декларация компонента la2
  (i1,i2: in std_logic; r: out std_logic);
 end component;

signal S1, S2 :  std_logic;
begin
 u1: la2 port map(x1,x2,S1);
 u2: la2 port map(x2,x3,S2);
 y <= S1 or S2;
end Structura;
```

Способы встраивания компонента



la2 – компонент
i1, i2, r – контакты (порты)
x1, x2, y - сигналы

```
--декларация компонента
entity la2 is
 Port (i1:in std_logic;
       i2:in std_logic;
       r:out std_logic);
end la2;
```

Два способа встраивания (конкретизация компонента)

1. Позиционный принцип сопоставления контактов (портов) компонентов и сигналов:

```
la2 port map (x1, x2, y);  -- Пример
```

2. Ключевой принцип:

имя контакта (порта) => сигнал

```
la2 port map (i1=>x1, i2=>x2, r=>y); -- Пример
```

Иерархия файлов в проекте

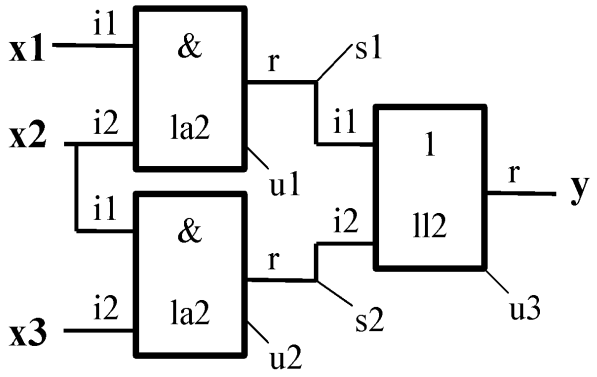
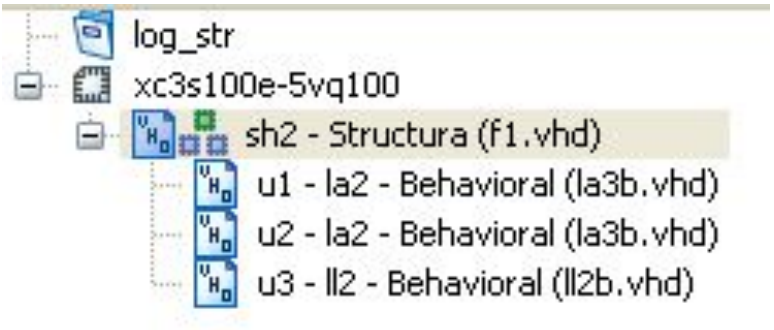
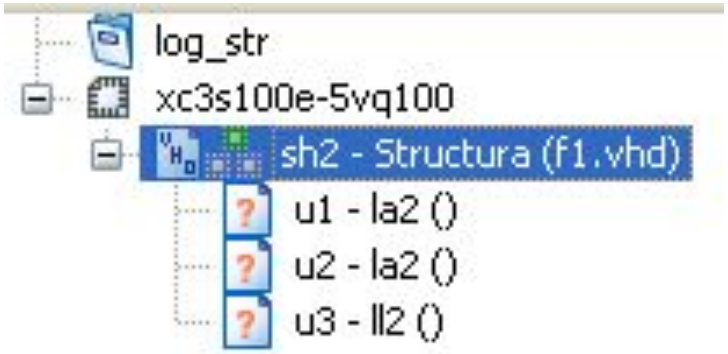


Схема 1

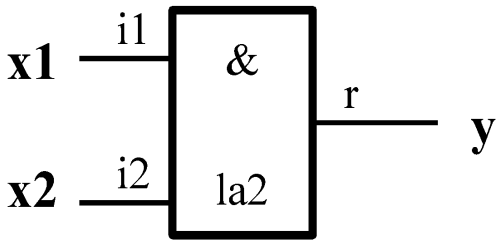


Проект без тестового модуля



Проект без компонентов

Входные сигналы

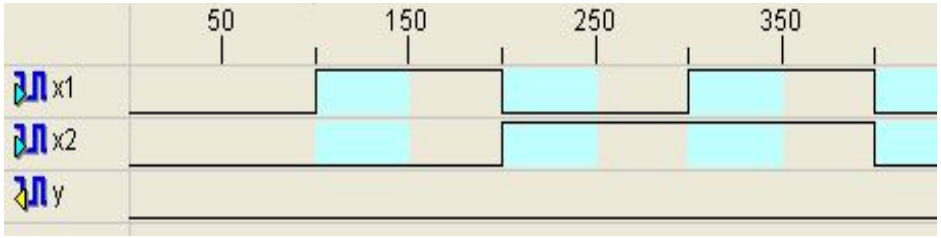


Вентиль 2И

Тестовый модуль формирует входные сигналы для проверки тестируемого модуля

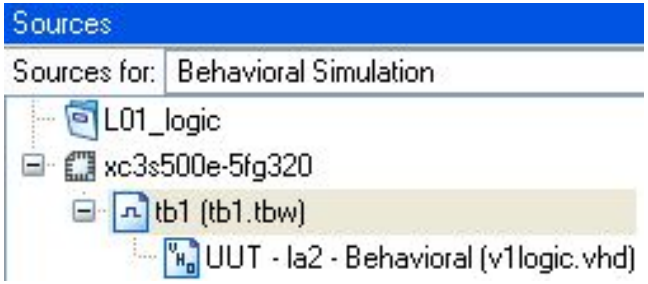
Входные сигналы задаются 2 способами:

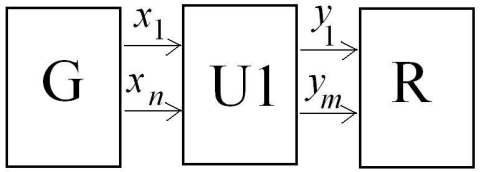
1. Графически (файл *.tbw).
2. Текстовым файлом.



i1, i2, r – контакты модуля la2;
x1, x2, y – внешние сигналы на контактах.

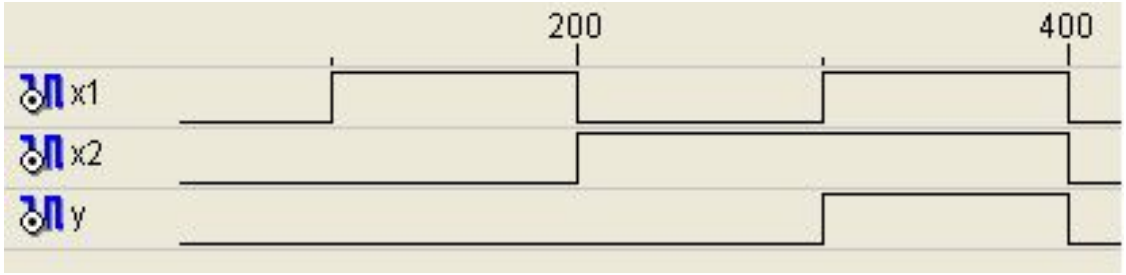
Структура проекта





U1 - настраиваемая схема
G - генератор тестовых сигналов
R - регистратор
$x_1 \ldots x_n$ - тестовые сигналы
$y_1 \ldots y_m$ - выходы U1

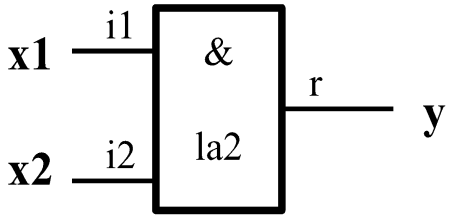
Результат моделирования



Тестовый модуль на языке VHDL

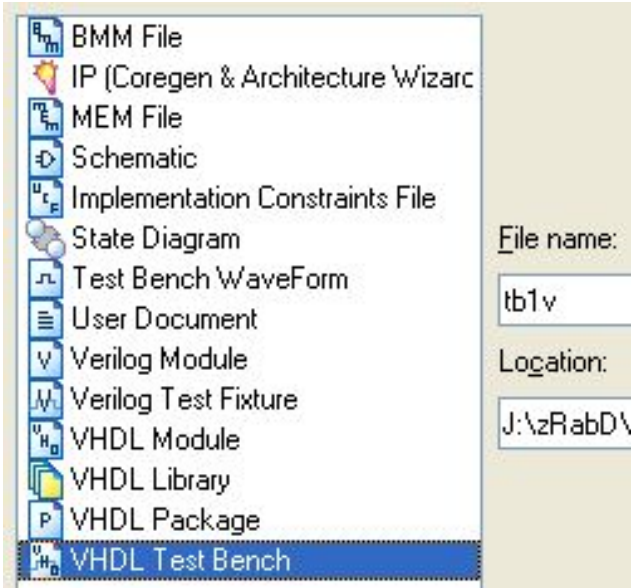
```
LIBRARY ieee;     -- библиотеки
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;
ENTITY tb1v_vhd IS -- внешние связи
END tb1v_vhd;      -- отсутствуют
Architecture behavior OF tb1v_vhd IS
-- Component Declaration ..
COMPONENT la2 -- тестируемый модуль
PORT( i1 : IN std_logic;
      i2 : IN std_logic;
   r : OUT std_logic
    );
 END COMPONENT;
 SIGNAL x1 :  std_logic := '0';
 SIGNAL x2 :  std_logic := '0';
 SIGNAL y :  std_logic;
```

Тестовый модуль формирует входные сигналы



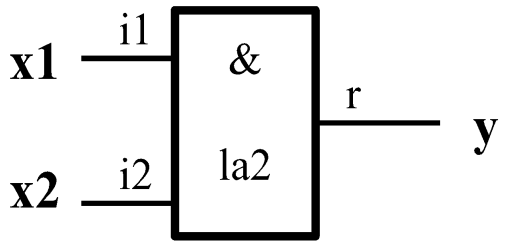
Тестируемый модуль, вентиль 2И

Выбор типа файла



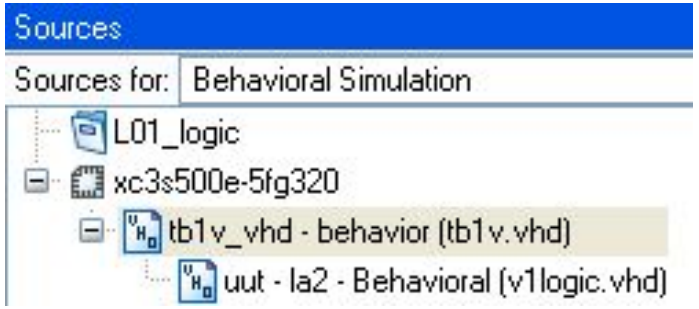
Тестовый модуль (продолжение)

```
BEGIN
-- Instantiate the Unit …
 uut: la2 PORT MAP(
        i1 => x1,
        i2 => x2,
        r => y
    );
 tb : PROCESS
 BEGIN
    wait for 100 ns;
    x1 <= '1';          -- t=100
    wait for 100 ns; -- t=200
    x1 <= '0'; x2 <= '1';
    wait for 100 ns;
    x1 <= '1';          -- t=300
    wait; -- wait forever
 END PROCESS;
END;
```

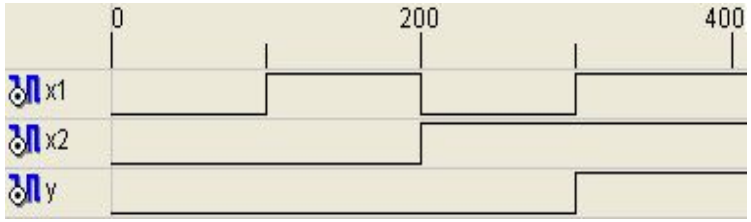


Тестируемый модуль, вентиль 2И

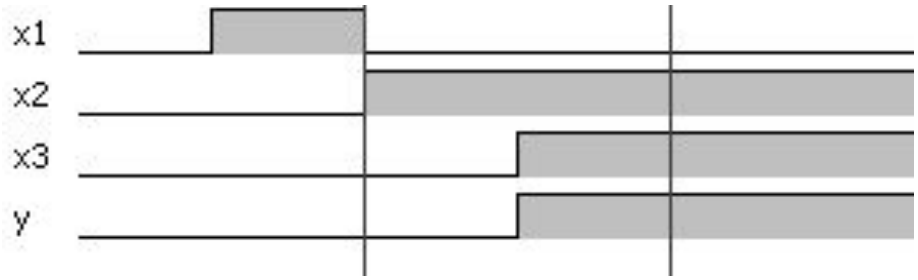
Структура проекта



Результат моделирования



Результат моделирования



Внешние сигналы

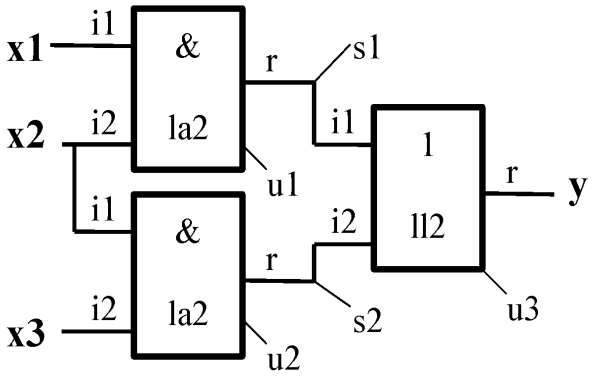
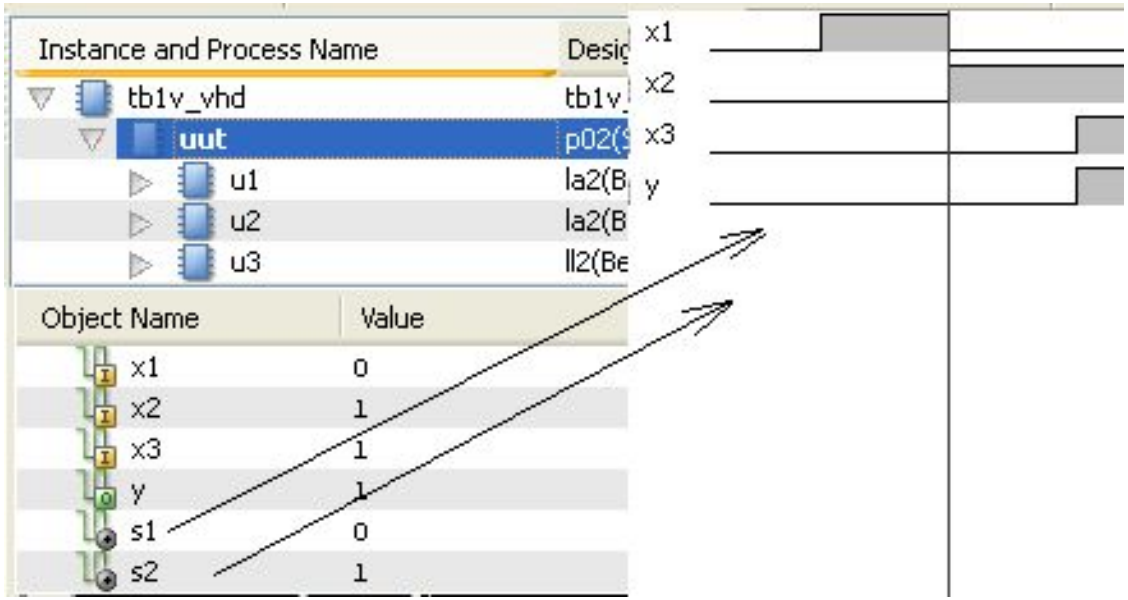


Схема 1



Добавление внутренних сигналов S1 и S2

1 – подготовка к новому запуску,

2 – запуск моделирования длительностью 1000 нс

Комбинационные и последовательные узлы.

Комбинационные узлы:

схемы на логических элементах,
мультиплексоры, дешифраторы и другие узлы, не содержащие триггеров.

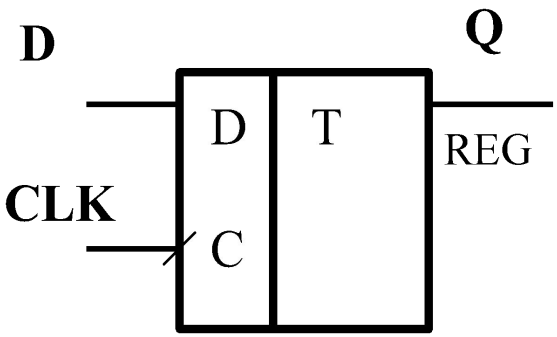
Последовательные узлы:

триггеры, регистры, счетчики и другие узлы, содержащие триггеры.

В описании на VHDL последовательные узлы содержат:

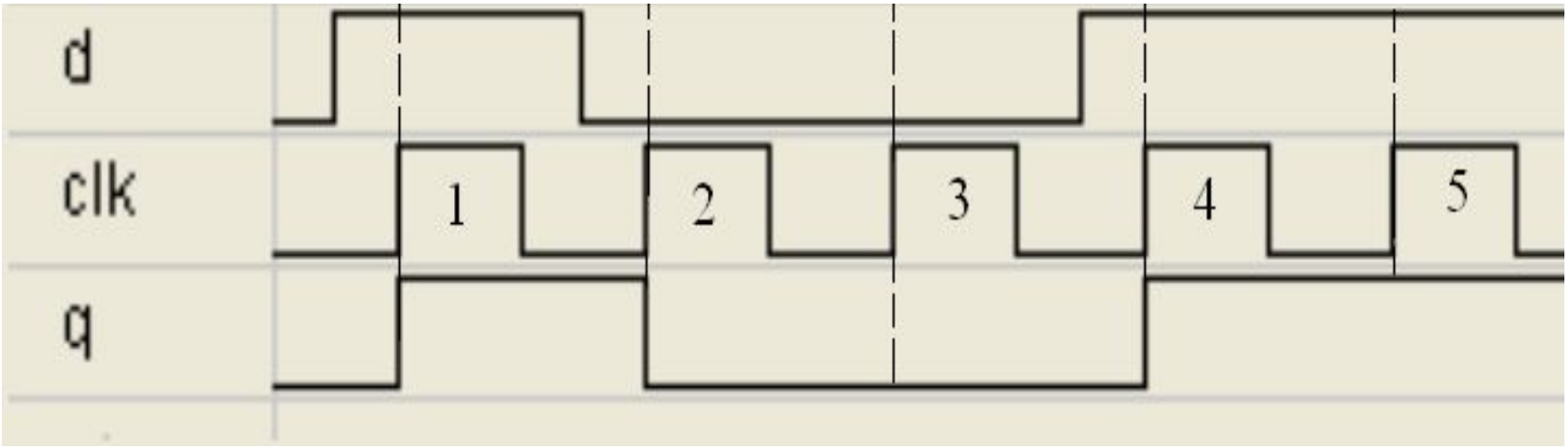
```
process (CLK)  begin                  -- CLK - синхросигнал
 if CLK'event and CLK='1' then -- передний фронт CLK
      . . .                    -- операторы присваивания
      . . .                    -- и другие операторы
 end if;
end process;
```

Схема D-триггера на языке VHDL




```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity rg1 is    -- внешн. связи
  Port (D: in std_logic;
    CLK: in std_logic; Q: out std_logic);
end rg1;
architecture Behavioral of rg1 is -- внутренние
  signal REG: std_logic:='0';     -- схемы
begin                             -- и связи (тело)
  process (CLK)  begin
     if CLK'event and CLK='1' then -- синхросигнал
        REG <= D;                  -- оператор присваивания
     end if;
  end process;
  Q <= REG;                       -- оператор присваивания
end Behavioral; -- наличие сигнала REG не обязательно
```

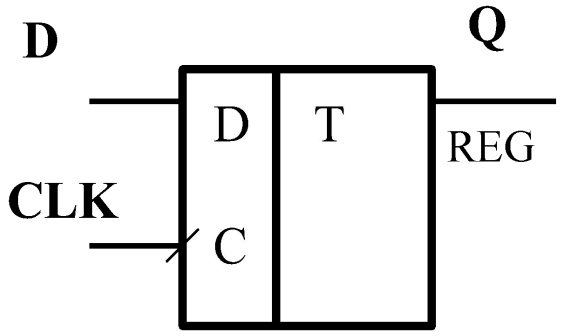
Результат работы



q □ d в момент переключения clk из 0 в 1

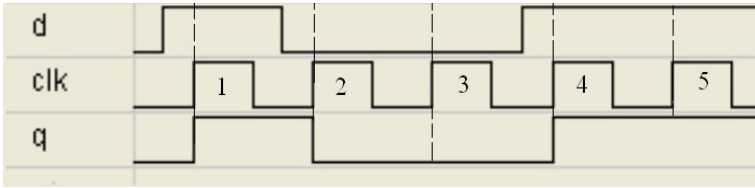
Тестовый модуль для D-триггера (ч.1)

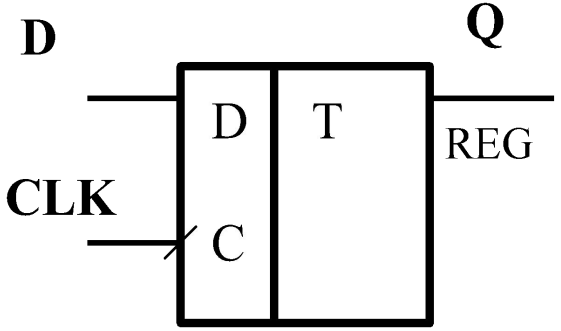
```
library IEEE;  -- тестовый модуль для D-триггера
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity tb1_vhd is  -- внешние связи отсутствуют
end tb1_vhd;

architecture behavior of tb1_vhd is   -- декларация
  component rg1 PORT(D: in std_logic; -- компонента
        CLK : in std_logic;
        Q   : out std_logic
   );
  end component;
  signal D :   std_logic := '0';
  signal CLK : std_logic := '0';
  signal Q :   std_logic;
```



Тестовый модуль для D-триггера (ч.2)

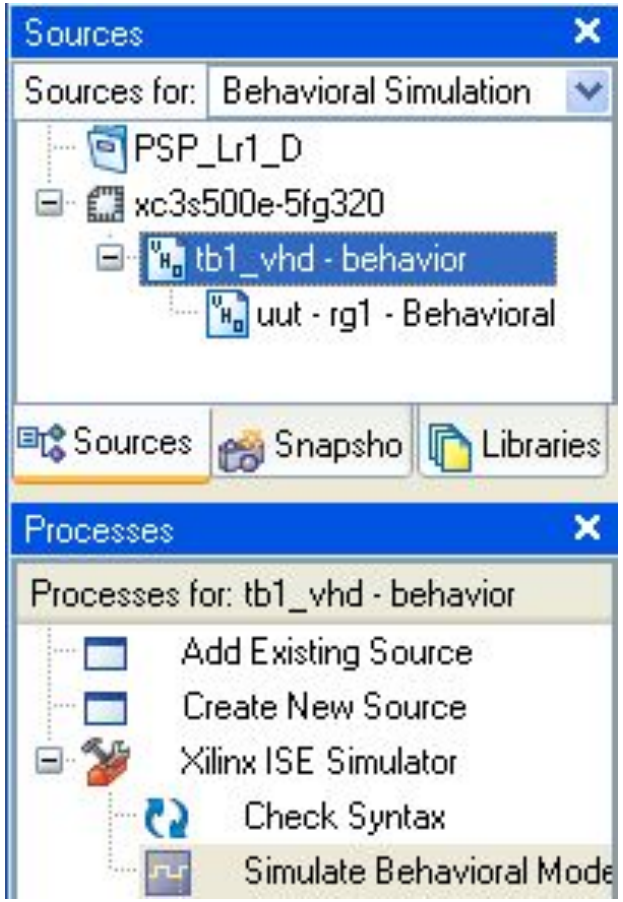
```
begin
  uut: rg1 port map( D => D, CLK => CLK, Q => Q);
   -- процессы clk_ и tb1 – параллельные операторы
   clk_process :process   begin
   clk <= '0';
      wait for 10 ns;
   clk <= '1';
      wait for 10 ns;
   end process;

   tb1 : process   begin
     wait for 5 ns;
     D <= '1';  wait for 20 ns;
     D <= '0';  wait for 40 ns;
     D <= '1';  wait for 20 ns;
     wait; -- will wait forever
    end process;
end;
```





Иерархия файлов в проекте

Моделирование



Реализация

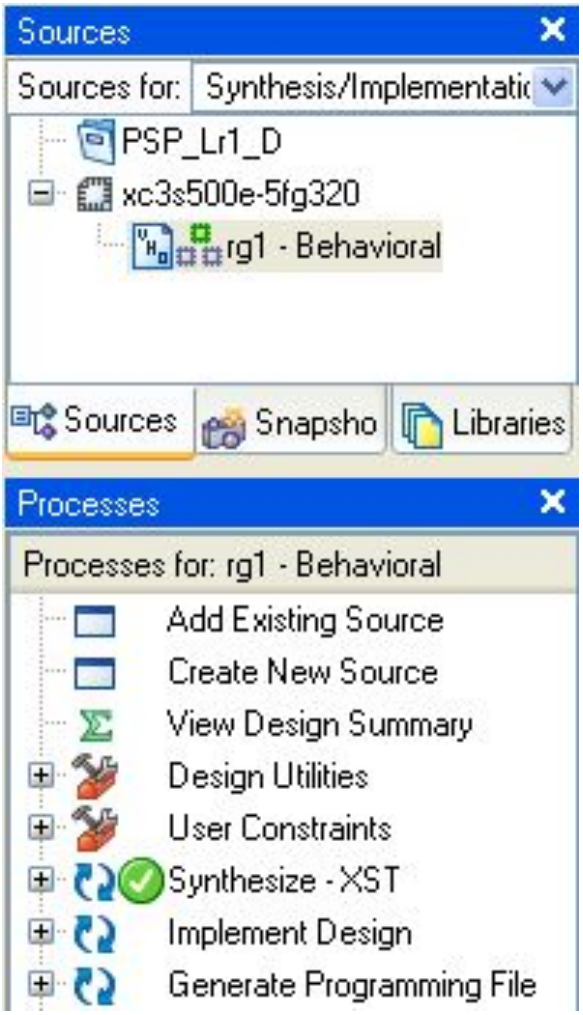
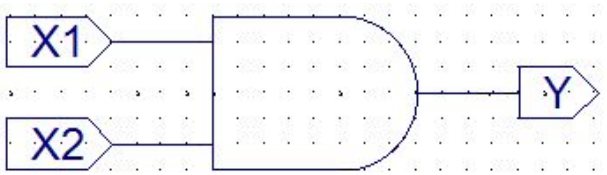


Схема вентиля на языке VHDL

```
library IEEE; -- библиотеки
  . . . .

entity la2 is -- внешн.связи
   Port (x1: in std_logic;
         x2: in std_logic;
          y: out std_logic);
end la2;

Architecture Behav of la2 is

begin
    y <= x1 and x2;
end Behavioral;
```



X1, X2 – входные сигналы.
Y – выходной сигнал,
Y=X1&X2.

Результаты моделирования не привязаны к синхросигналу

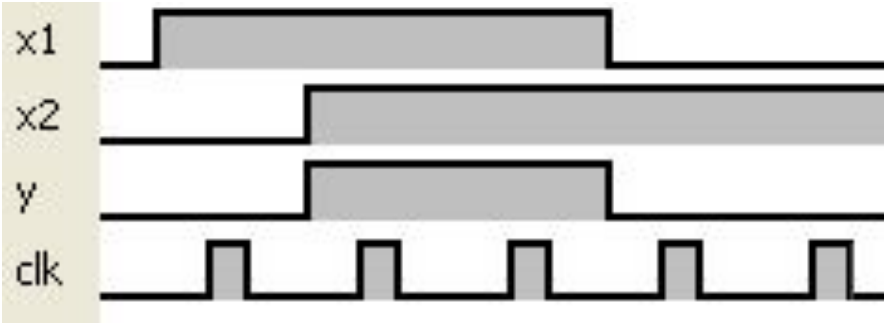
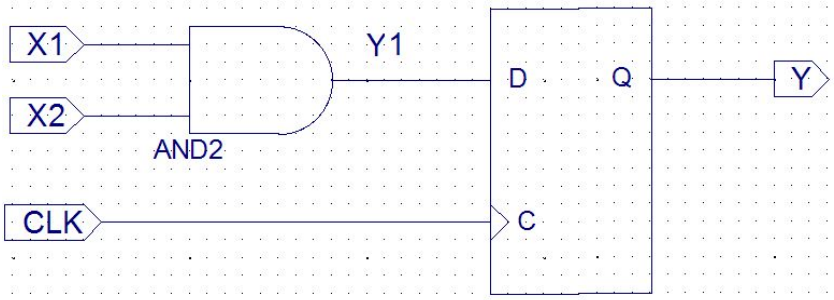


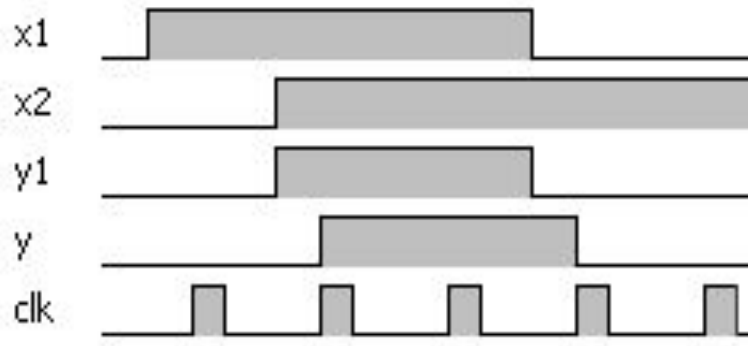
Схема вентиля с триггером на выходе

```
library IEEE; -- библиотеки
  . . . .
entity la2 is -- внешн.связи
Port (x1,x1: in std_logic;
        clk: in std_logic;
         y: out std_logic);
end la2;
Architecture B1 of la2 is
begin    --
process (CLK)  begin
 if CLK'event and CLK='1'
  then -- синхросигнал
   y <= i1 and i2;
  end if;
 end process;
end Behavioral;
```



X1, X2 – входные сигналы.
clk – синхросигнал
Y – выходной сигнал,
Y=X1&X2.

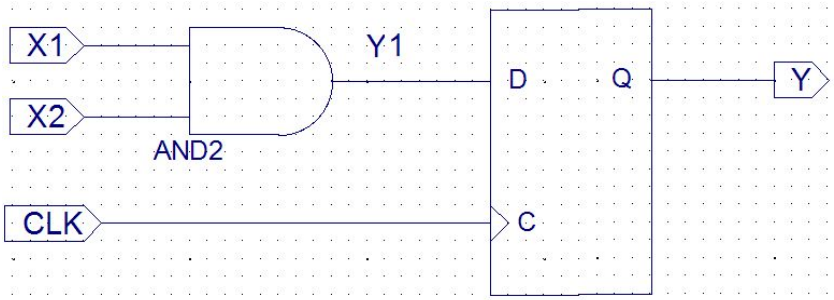
Результаты моделирования,
выходной сигнал Y
“привязан” к синхросигналу CLK




```
Architecture Behav of la2 is
begin    -- без триггера
   y <= x1 and x2;
end Behavioral;
```

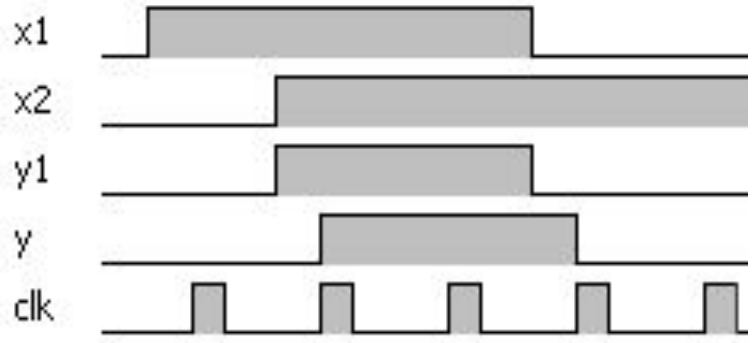
Схема вентиля с триггером на выходе (в.2)

```
library IEEE; -- библиотеки
  . . . .

entity la2 is -- внешн.связи
Port (x1,x1: in std_logic;
          clk: in std_logic;
           y: out std_logic);
end la2;

Architecture B1 of la2 is
signal y1: std_logic:='0';
begin
process (CLK)  begin
 if CLK'event and CLK='1'
  then
   y <= y1;
 end if;
end process;
y1 <= i1 and i2;
end Behavioral;
```



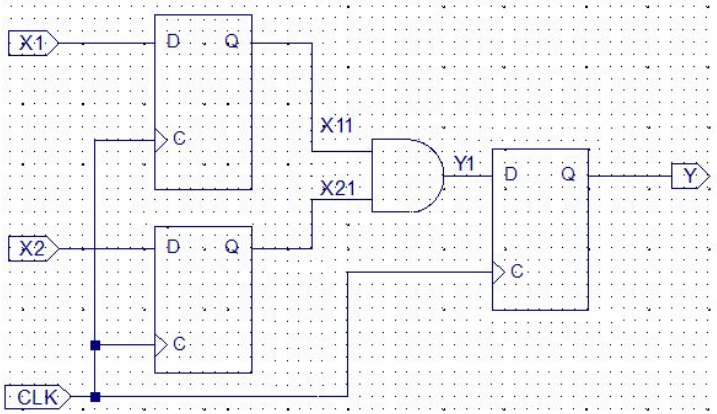
X1, X2 – входные сигналы.
clk – синхросигнал
Y – выходной сигнал,
Y=X1&X2.

Результаты моделирования,
выходной сигнал Y
“привязан” к синхросигналу CLK



Вентиль с триггерами на входах и выходе

```
library IEEE;
  . . . .
entity la2 is
Port (x1,x1: in std_logic;
         clk: in std_logic;
          y: out std_logic);
end la2;

Architecture B1 of la2 is
signal x11,x21:std_logic;
begin
process (CLK)  begin
 if CLK'event and CLK='1'
  then
   x11 <= x1;
   x21 <= x2;
   y <= x11 and x21;
  end if;
 end process;
end Behavioral;
```



x1, x2 – входные сигналы.
Y – выходной сигнал.
x11,x21 – внутренние сигналы.

Результаты моделирования,
Сигнал x11, x21 и y “привязаны” к синхросигналу CLK

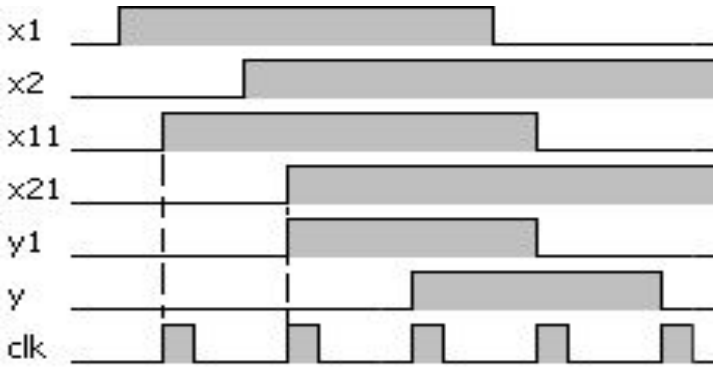
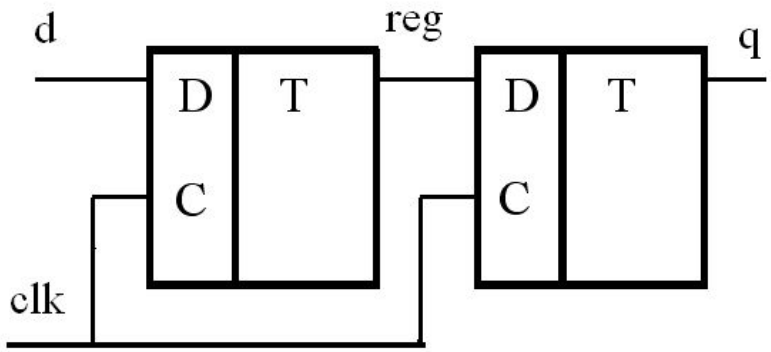
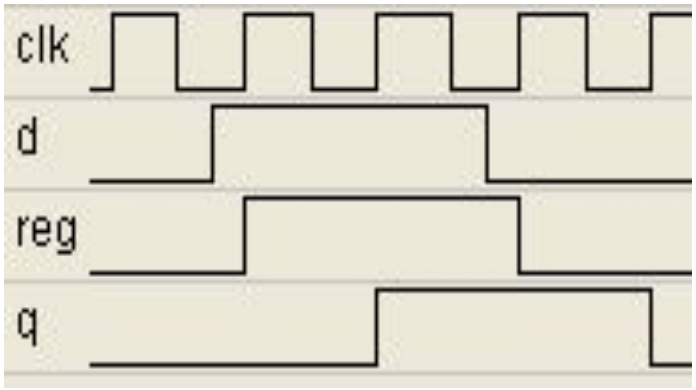


Схема задержки на языке VHDL

```
library IEEE;
  . . .
entity r1 is
  Port (d, clk: in std_logic;
        q:out std_logic);
end r1;

architecture Behavioral of r1 is
  signal reg: std_logic:='0';
begin
 process (clk)  begin
  if clk'event and clk='1' then
      reg <= d;
      q <= reg;
   end if;
  end process;
end Behavioral;
```





Если поменять местами **reg <= d** и **q <= reg**, результат не изменится

Векторные операции.

Векторные сигналы – многоразрядные сигналы, применяются в многоразрядных схемах (регистры, счетчики, сумматоры и др)

Декларация векторного сигнала:

```
signal ct: std_logic_vector(3 downto 0);
```
-- декларированы 4 сигнала: ct(3) ct(2) ct(1) ct(0)

```
signal rg: std_logic_vector(3 downto 0);
```
-- декларированы 4 сигнала: rg(3) rg(2) rg(1) rg(0)

```
signal v1: std_logic_vector(0 to 2);
```
-- декларированы 3 сигнала: v1(0) v1(1) v1(2)

Операции с векторным сигналом:

```
ct <=”0000”; ct <=x”0”;
ct <=”0010”; ct <=x”2”;
```

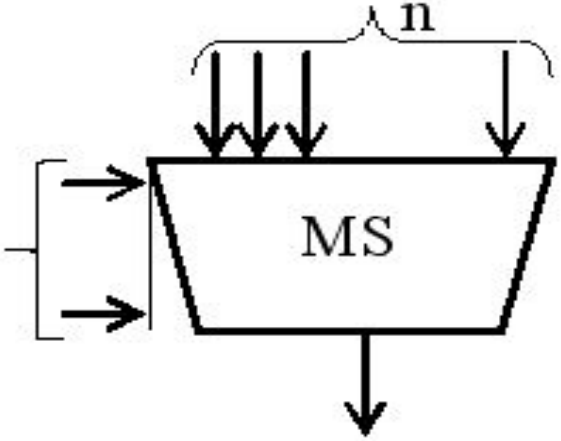
Сдвиг влево:

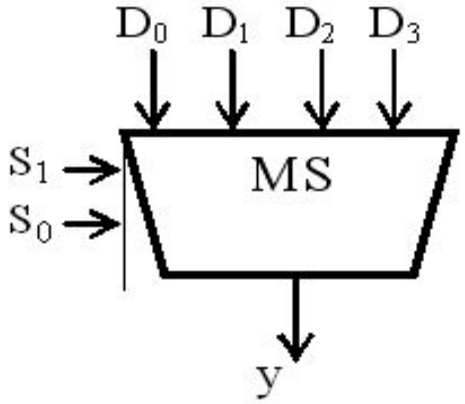
```
rg <= rg(2 downto 0) & ‘0’;
```
-- rg(2) rg(1) rg(0) 0

Декларация выходного векторного сигнала:

```
DOUT : out std_logic_vector(3 downto 0)
```

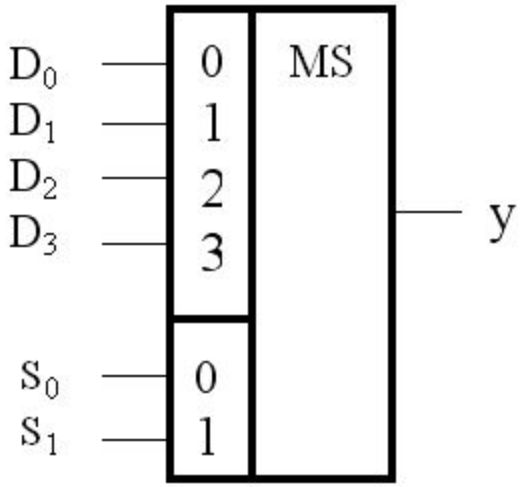
Мультиплексоры, методы реализации





s_1	s_0	y
0	0	$y=D_0$
0	1	$y=D_1$
1	0	$y=D_2$
1	1	$y=D_3$

Одноразрядный мультиплексор



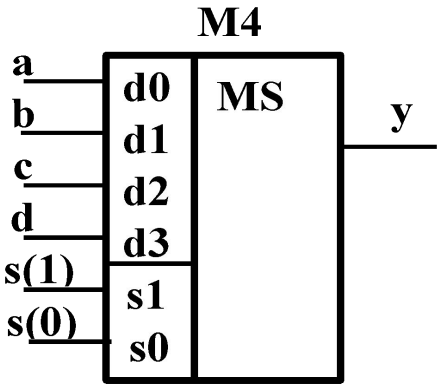
S – двухразрядный векторный сигнал

Мультиплексор (операторы if, process)

```
library IEEE;
    . . .
entity M4 is
    Port ( a,b,c,d : in  STD_LOGIC;
           s : in  STD_LOGIC_VECTOR (1 downto 0);
     y : out  STD_LOGIC);
end M4;     --   s – двухразрядный векторный сигнал
```

```
architecture Behavioral of M4 is
begin
 mux4a:process (a,b,c,d,s) begin
   if s="00" then
     y <=a;
   elsif s="01" then   y <=b;
   elsif s="10" then   y <=c;
   else        y <=d;
   end if;
 end process mux4a;
end Behavioral;
```

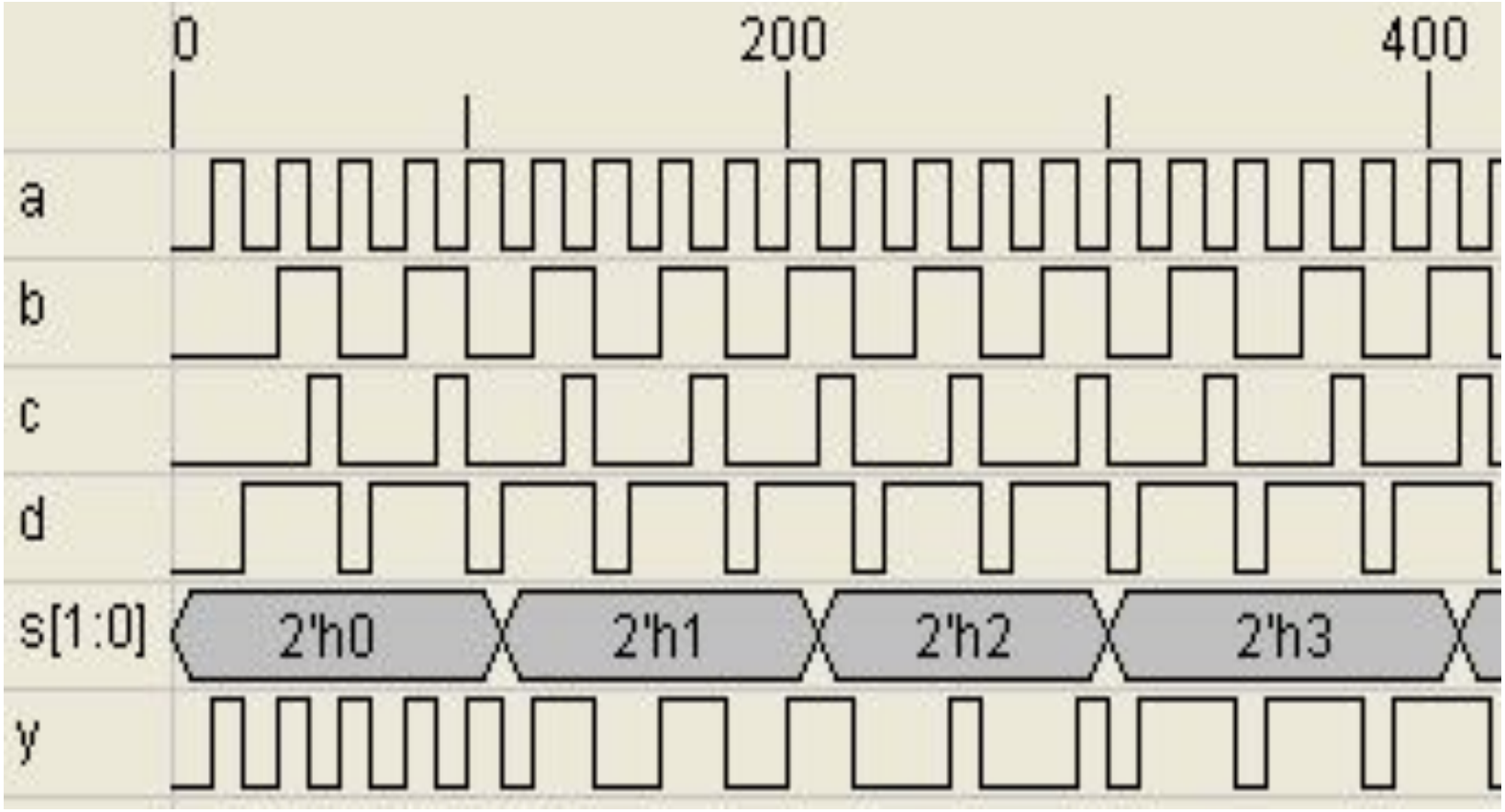
s_1	s_0	y	
0	0	$y=D_0$	a
0	1	$y=D_1$	b
1	0	$y=D_2$	c
1	1	$y=D_3$	d



Работа мультиплексора

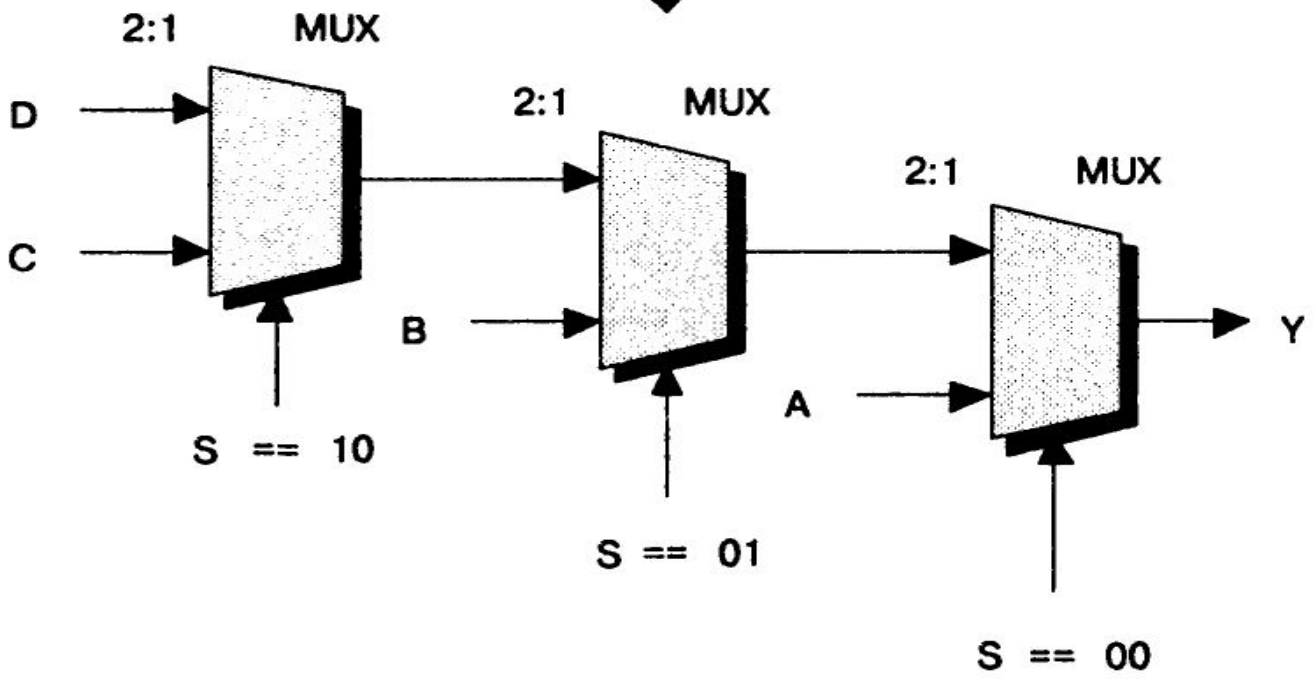
```
при s="00"  y = a
при s="01"  y = b
при s="10"  y = c
при s="11"  y = d
```

```
mux4a:process(a,b,c,d,s)
 begin
  if s="00" then
     y <=a;
 elsif s="01" then   y <=b;
 elsif s="10" then   y <=c;
 else        y <=d;
 end if;
end process mux4a;
```



Использование IF

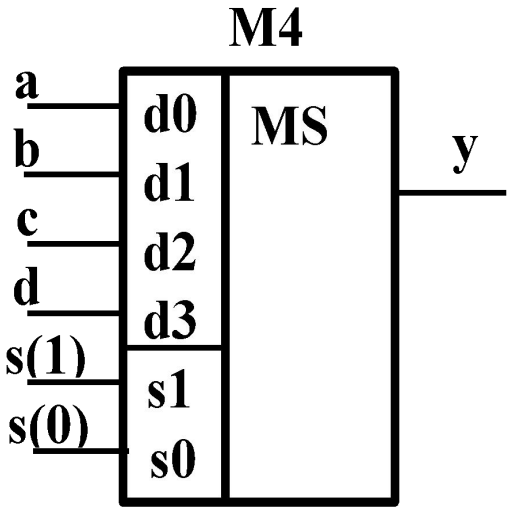
```
if      S  ==  00"  then  Y  =  A;
elseif  S  ==  01"  then  Y  =  B;
elseif  S  ==  10"  then  Y  =  C;
else                      Y  =  D;
end  if;
```



Задержка 3 элемента

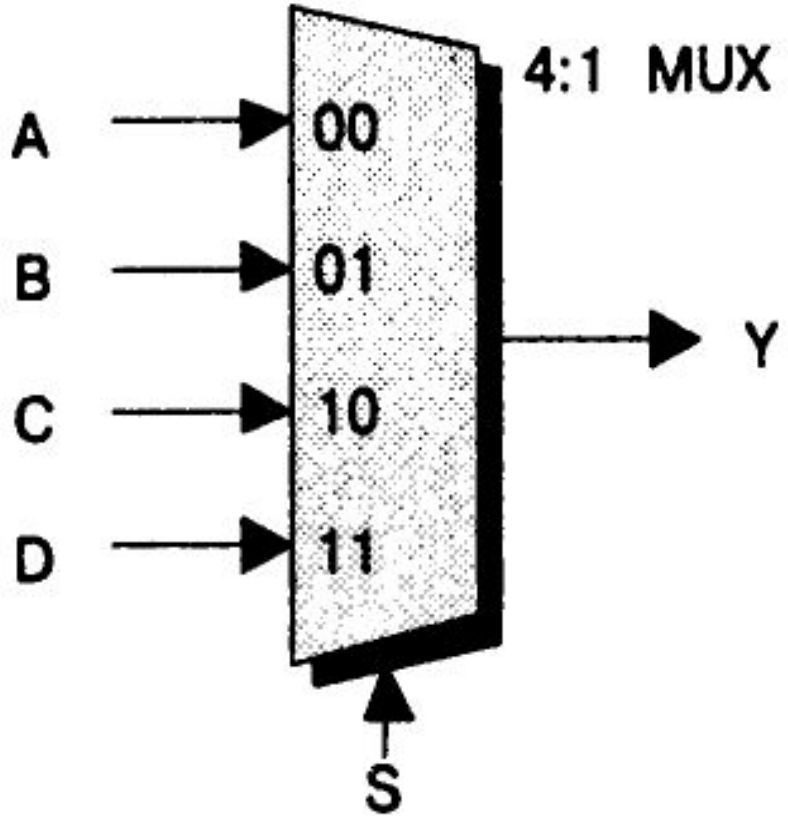
Мультиплексор, использование case

```
library IEEE;
    . . .
entity M4 is
    Port ( a,b,c,d : in  STD_LOGIC;
           s : in  STD_LOGIC_VECTOR (1 downto 0);
     y : out  STD_LOGIC);
end M4;
architecture Behavioral of M4 is
begin
 mux4a:process (a,b,c,d,s) begin
 case s is
   when "00" => y <= a;
   when "01" => y <= b;
   when "10" => y <= c;
   when others => y <= d;
 end case;
 end process mux4a;
end Behavioral;
```



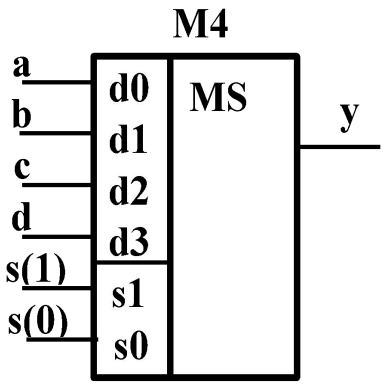
Мультиплексор, использование case

```
case   S     of;
       00":  Y = A;
       01":  Y = B;
       10":  Y = C;
otherwise:   Y = D;
end   case;
```



Задержка 1 элемент

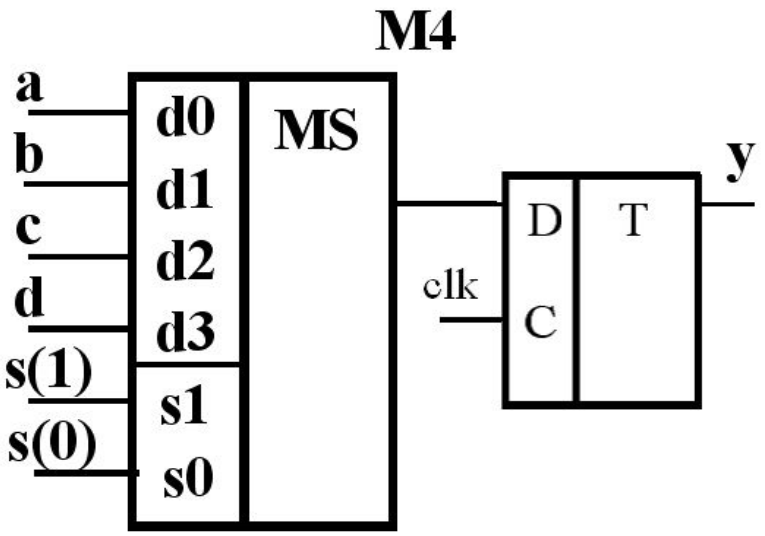
Мультиплексор, оператор when



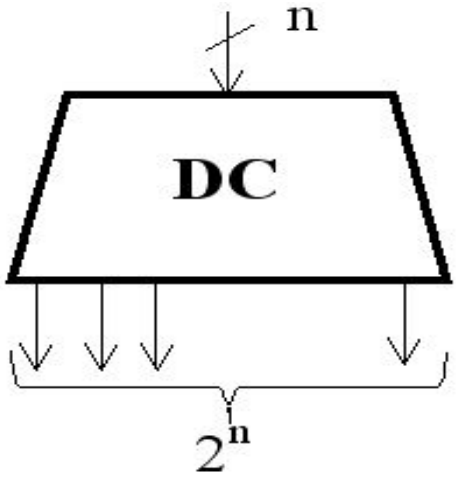

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity M4 is
    Port ( a, b, c, d : in  STD_LOGIC;
           s : in  STD_LOGIC_VECTOR (1 downto 0);
     y : out  STD_LOGIC);
end M4;

architecture Behavioral of M4 is
begin
y  <=   a when s="00"  -- может использоваться
   else b when s="01"  -- без оператора process
   else c when s="10"
   else d;
end Behavioral;
```

Мультиплексор с триггером на выходе

```
entity M4 is Port(clk,a,b,c,d :in  STD_LOGIC;
        s : in  STD_LOGIC_VECTOR (1 downto 0);
        y : out  STD_LOGIC);
end M4;    -- операторы библиотек не показаны
architecture Behavioral of M4 is
begin
process (clk) begin
 if clk='1' and clk'event then
   case s is
  when "00" => y <= a;
  when "01" => y <= b;
  when "10" => y <= c;
  when others => y <=d;
   end case;
  end if;
end process;
end Behavioral;
```

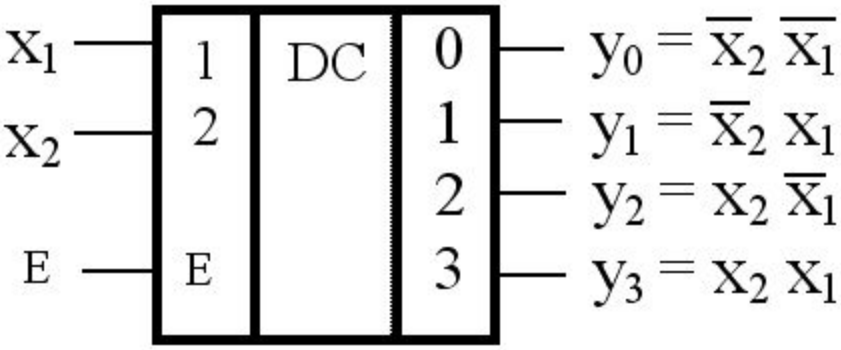


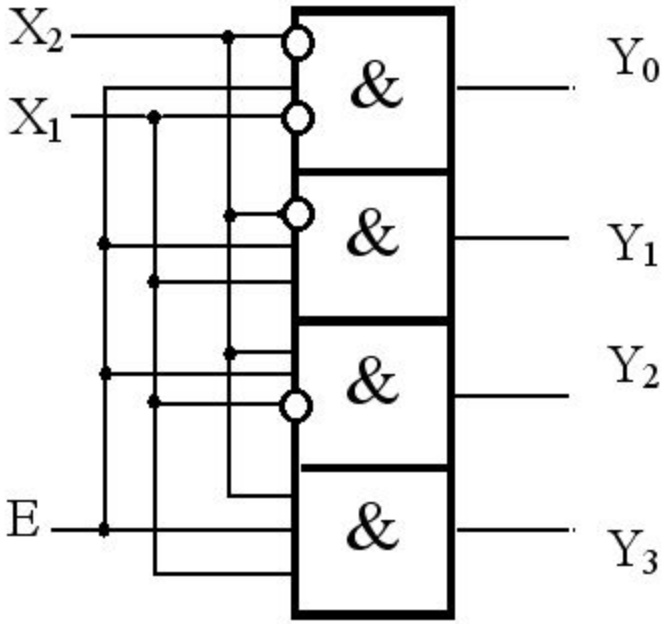
Дешифратор



Дешифратор на 2 входа

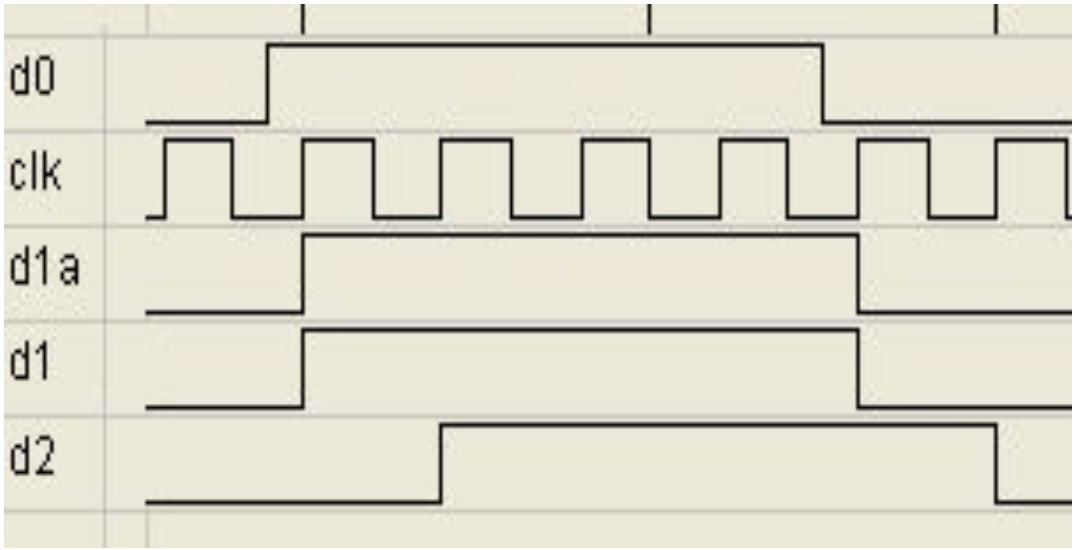
N	X_2	X_1	Y_0	Y_1	Y_2	Y_3
0	0	0	1	0	0	0
1	0	1	0	1	0	0
2	1	0	0	0	1	0
3	1	1	0	0	0	1

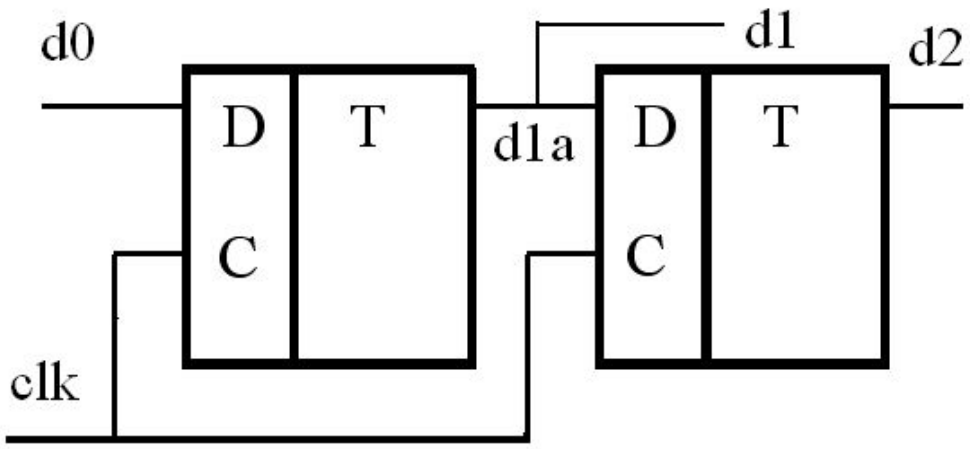




Триггерная схема

```
entity dtg2 is
    Port ( d0, clk : in  STD_LOGIC;
           d1, d2 : out  STD_LOGIC);
end dtg2;

architecture Behavioral of dtg2 is
signal d1a: std_logic:='0';
begin
 process (clk) begin
     if clk='1' and clk'event then
         d1a <= d0;
         d2 <= d1a;
   end if;
 end process;
 d1 <= d1a;
end Behavioral;
```

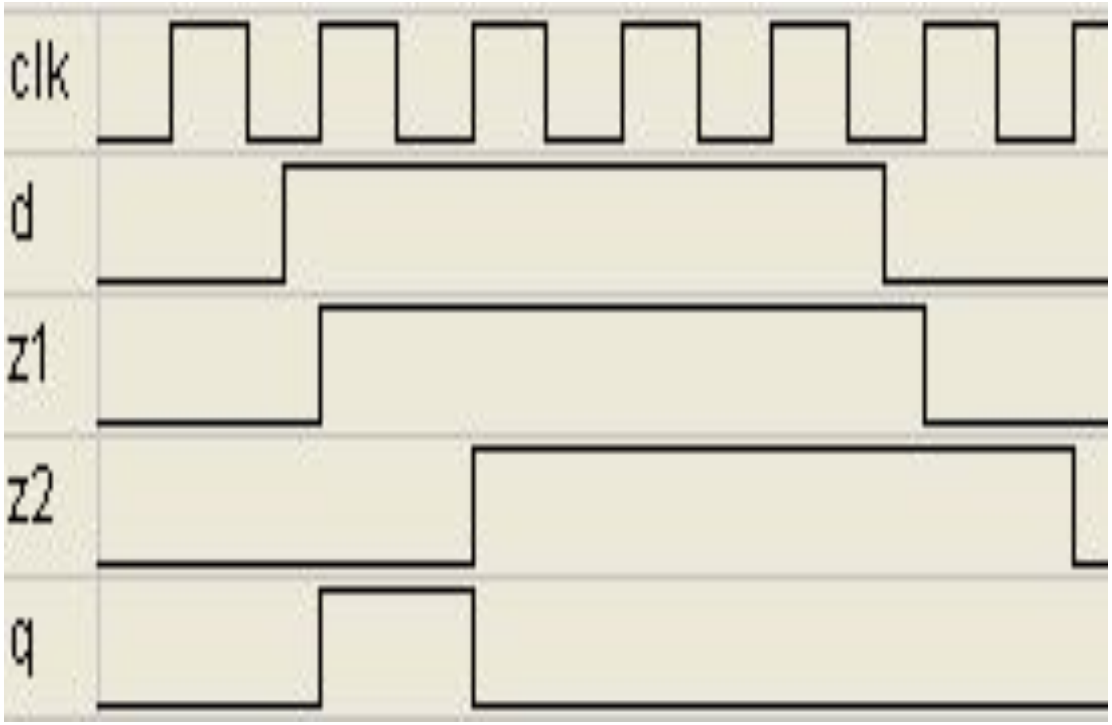




Формирователь

Описание на языке VHDL

```
entity f2a is
    Port ( d, clk : in  STD_LOGIC;
           q : out  STD_LOGIC);
end f2a;

architecture Behavioral of f2a is
signal z1,z2: std_logic:='0';
begin
 process (clk) begin
     if clk='1' and clk'event then
         z1 <= d;   z2 <= z1;
    end if;
 end process;
 q <= ( z1 and (not z2) );
end Behavioral;
```

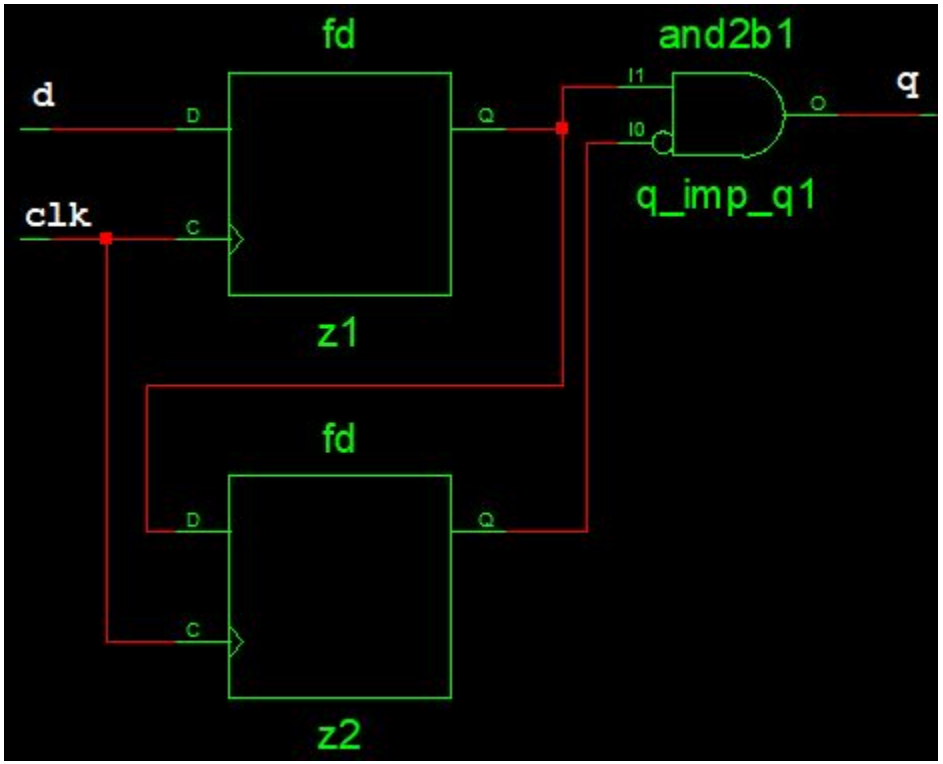


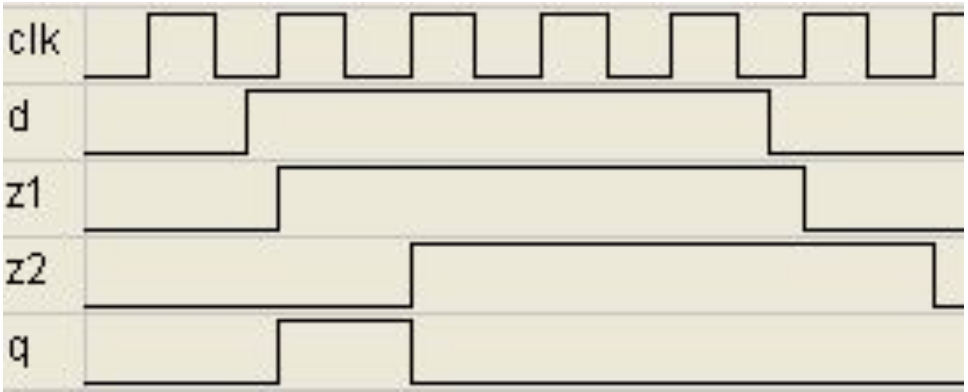
Временная диаграмма работы формирователя,
d – вход, q – выход,
z1, z2 – внутренные сигналы

Схема формирователя

Описание на языке VHDL

```
entity f2a is
    Port ( d, clk : in  STD_LOGIC;
           q : out  STD_LOGIC);
end f2a;

architecture Behavioral of f2a is
signal z1,z2: std_logic:='0';
begin
 process (clk) begin
     if clk='1' and clk'event then
         z1 <= d;   z2 <= z1;
    end if;
 end process;
 q <= ( z1 and (not z2) );
end Behavioral;
```





THE END