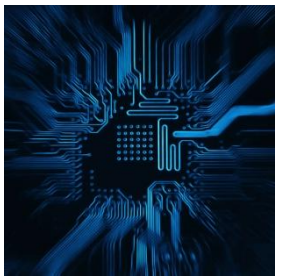
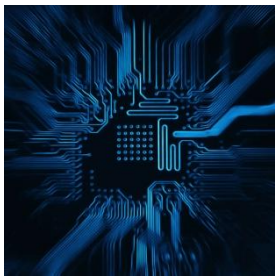
Программирование FPGA и их использование для высокопроизводительных вычислений



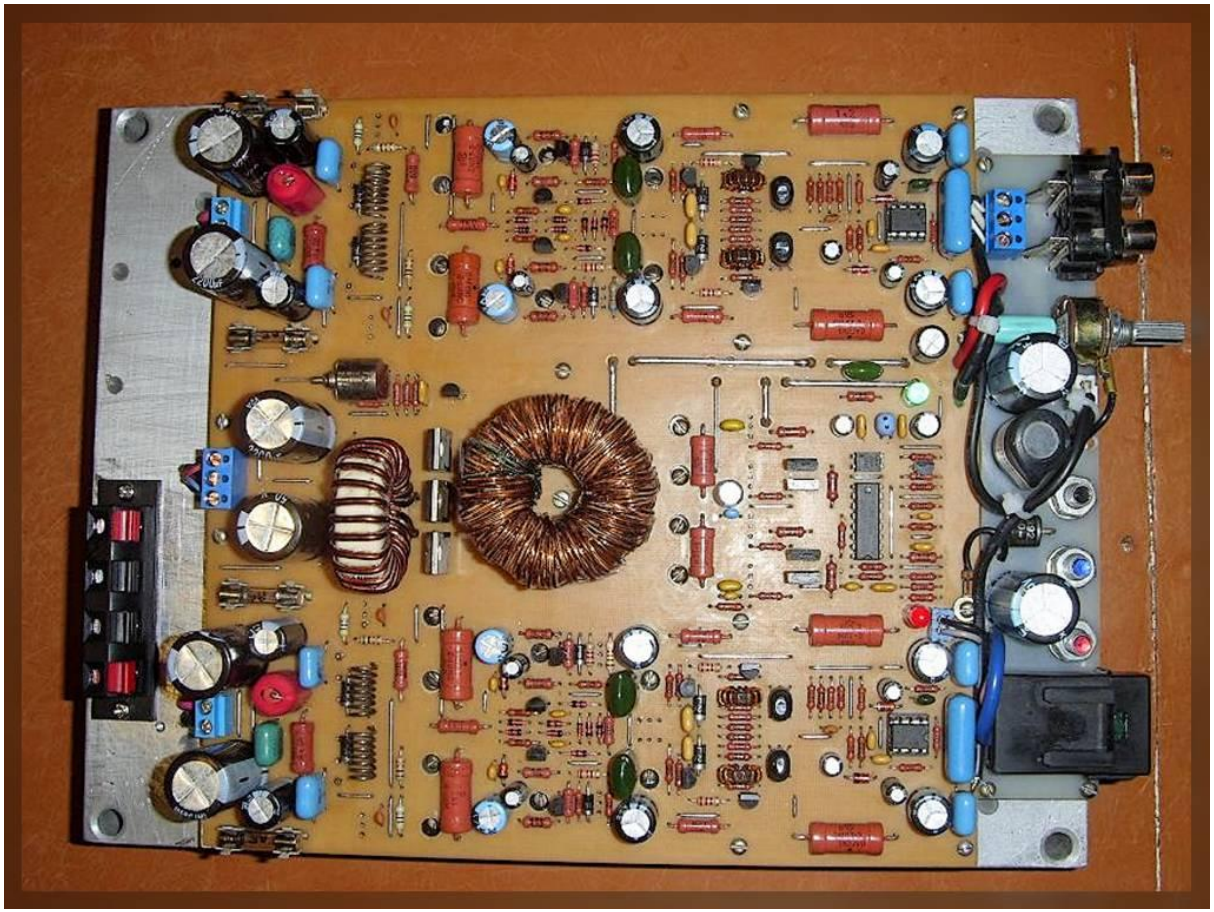
Тема 5
Схемотехника последовательных устройств

Сигнал синхронизации

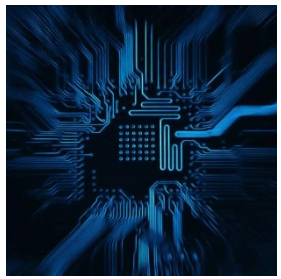


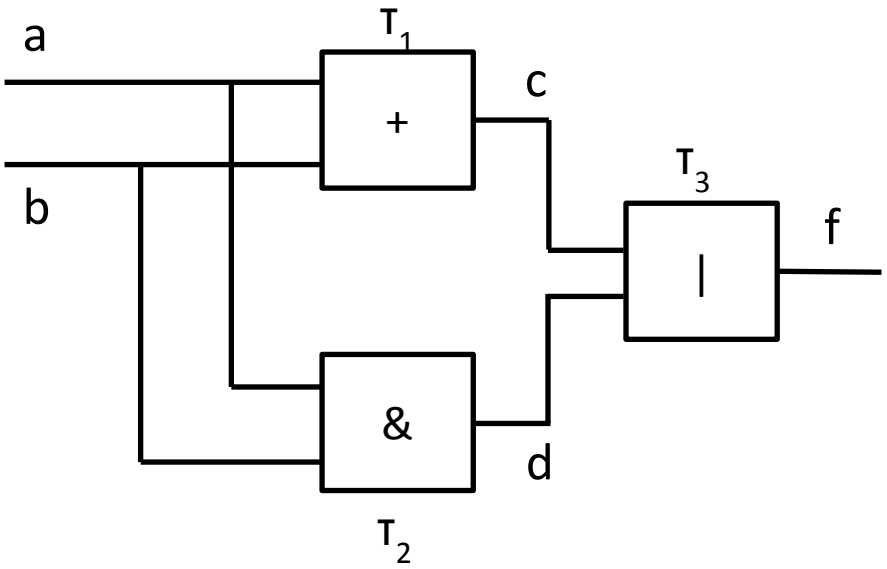
Что такое синхроимпульс и зачем он нужен?

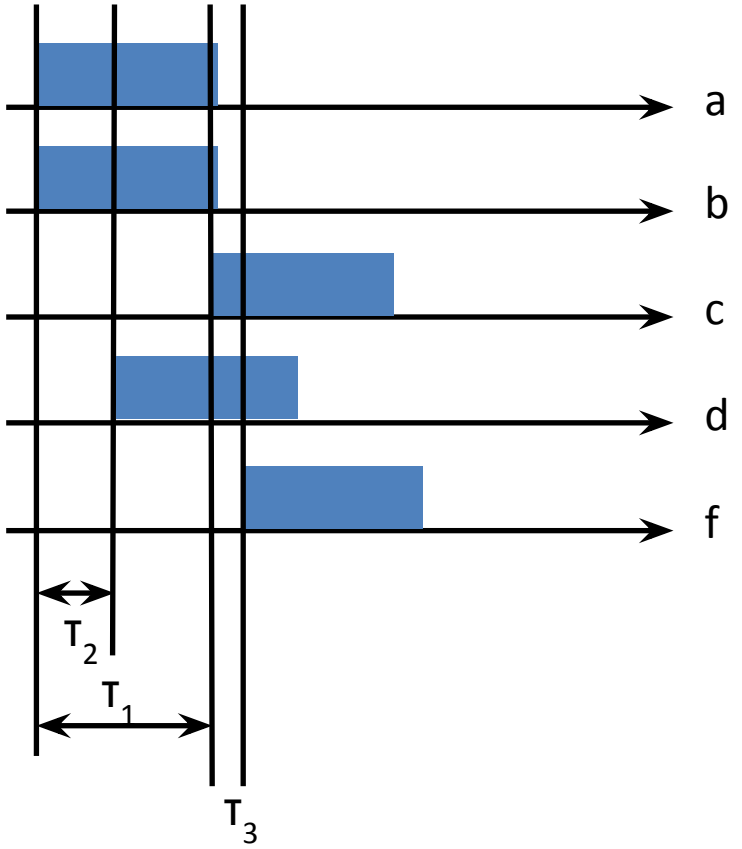
Сигнал синхронизации



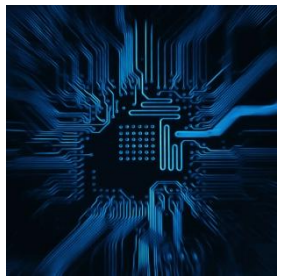
Сигнал синхронизации

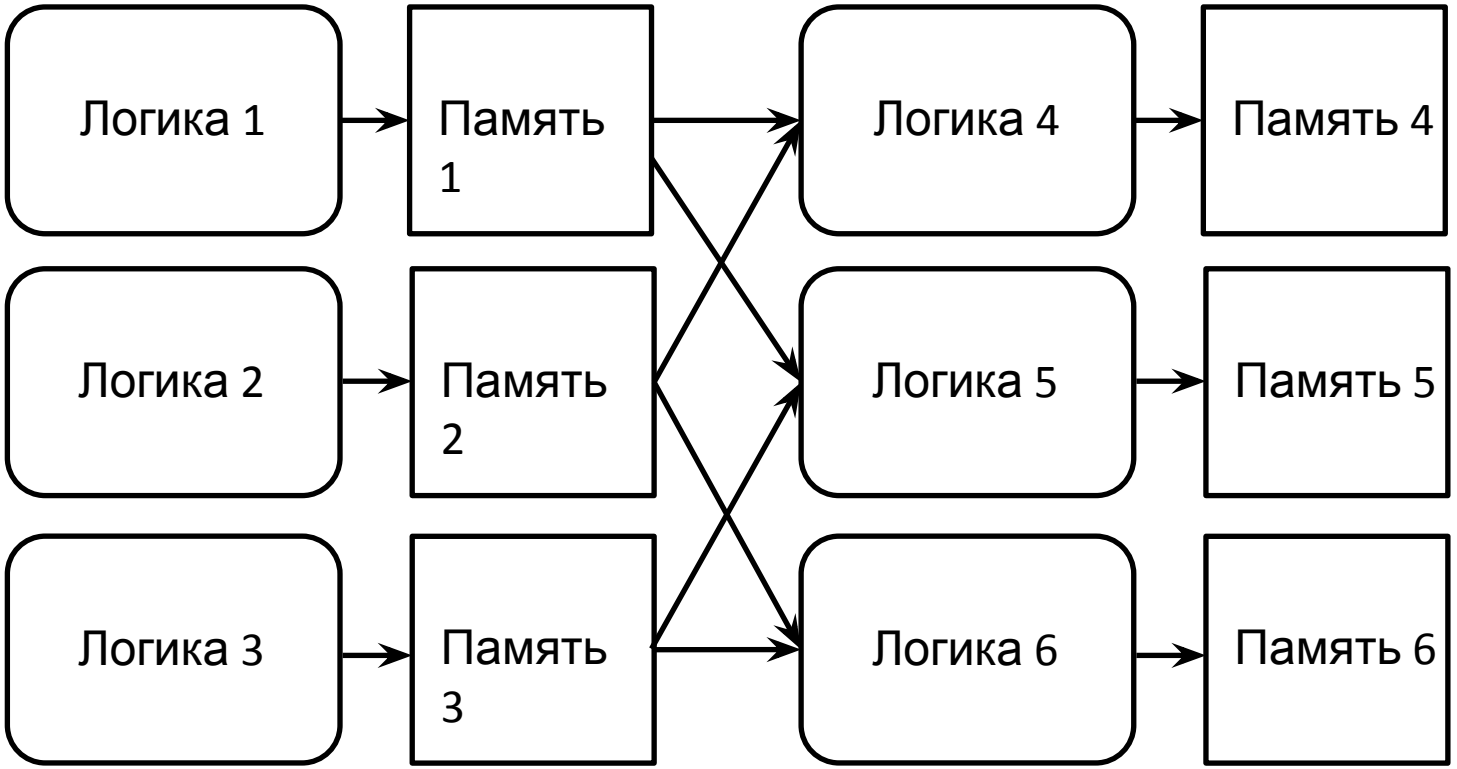




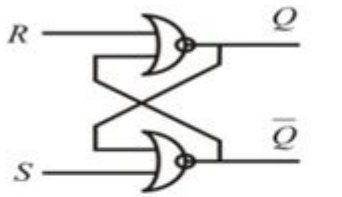


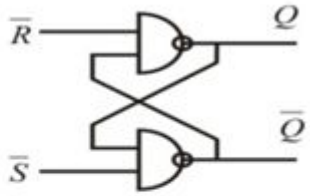
Сигнал синхронизации

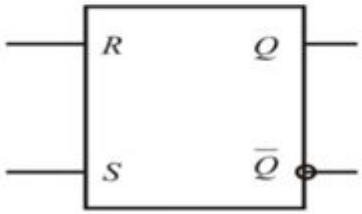


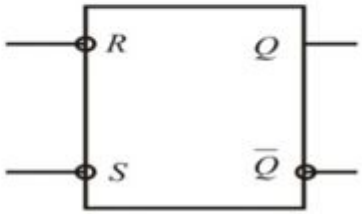


RS триггер





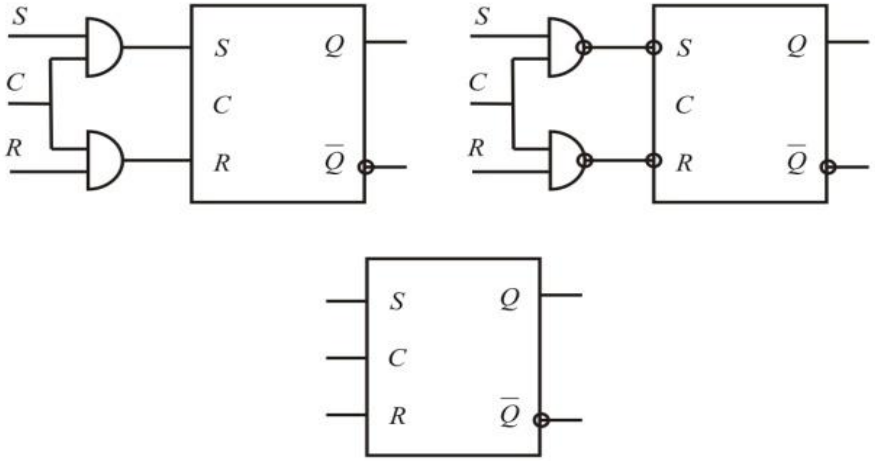


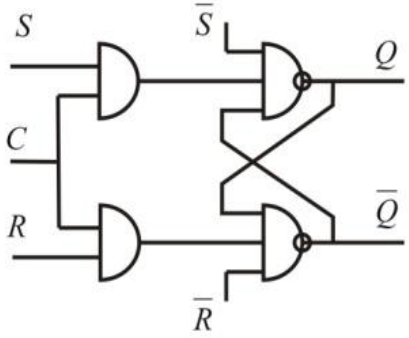


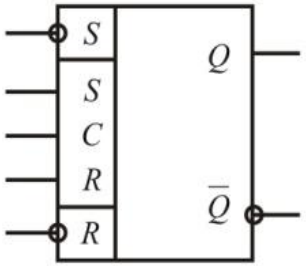
S	R	Qn+1
0	0	Qn
0	1	0
1	0	1
1	1	?

S	R	Qn+1
0	0	?
0	1	1
1	0	0
1	1	Qn

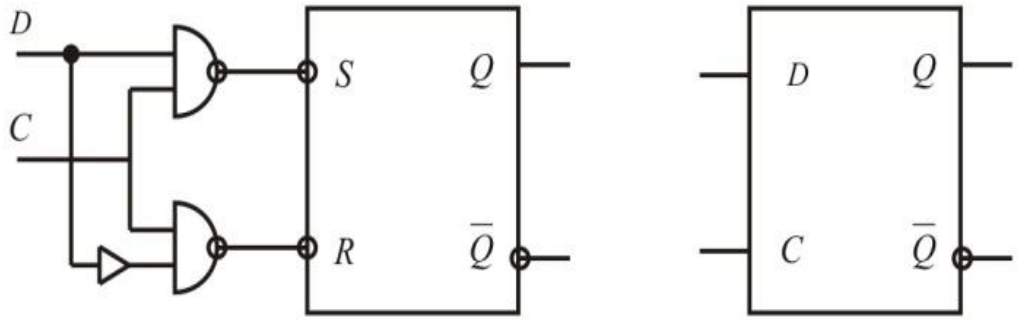
RS триггер со статической синхронизацией





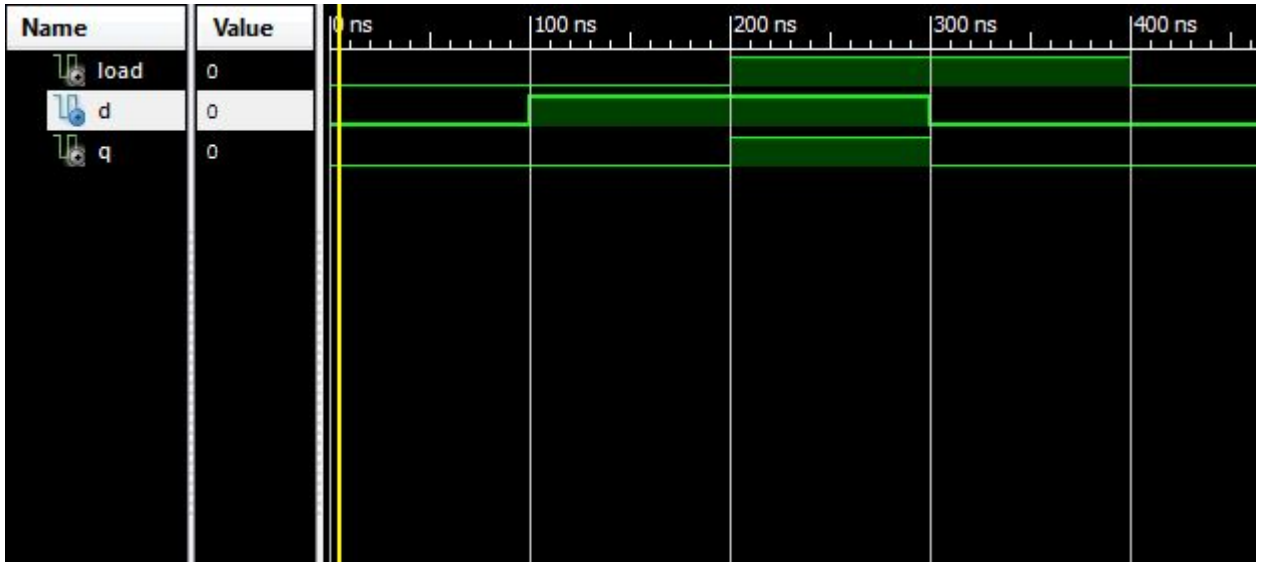


D триггер

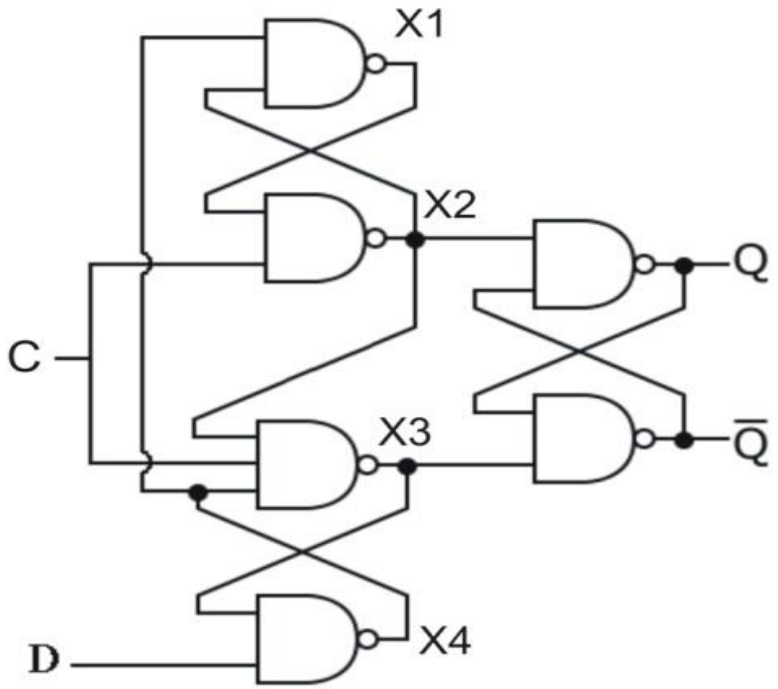


Защёлка (latch)

```
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 entity Latch is
26     Port ( load : in STD_LOGIC;
27            D : in STD_LOGIC;
28            Q : out STD_LOGIC);
29 end Latch;
30
31 architecture Behavioral of Latch is
32     signal q_tmp : std_logic := '0';
33 begin
34
35     Process (load, D    )
36     Begin
37         If (load = '1') then
38             q_tmp <= D;
39         End if;
40     End process;
41     Q <= q_tmp;
42
43 end Behavioral;
```



D триггер с динамической синхронизацией

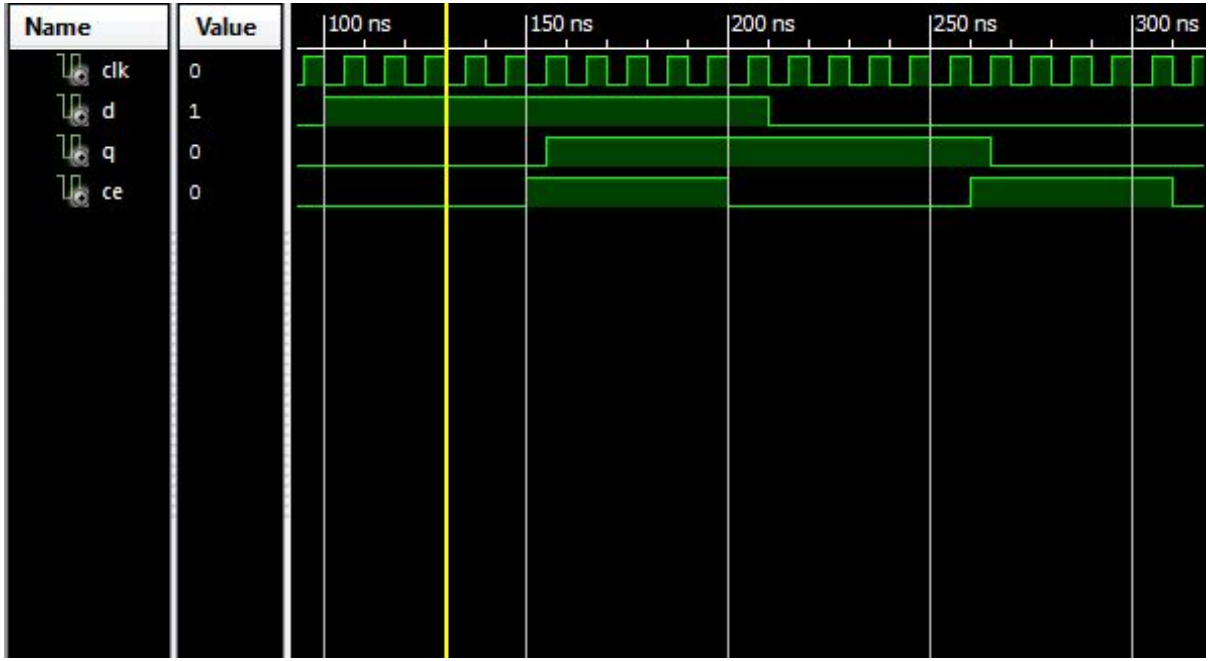


D триггер

```
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 entity Dtrig is
26     Port ( clk : in STD_LOGIC;
27            D : in STD_LOGIC;
28            Q : out STD_LOGIC);
29 end Dtrig;
30
31 architecture Behavioral of Dtrig is
32
33 begin
34
35     Process (clk)
36     Begin
37         If (rising_edge(clk)) then
38             Q <= D;
39         End if;
40     End process;
41
42 end Behavioral;
```

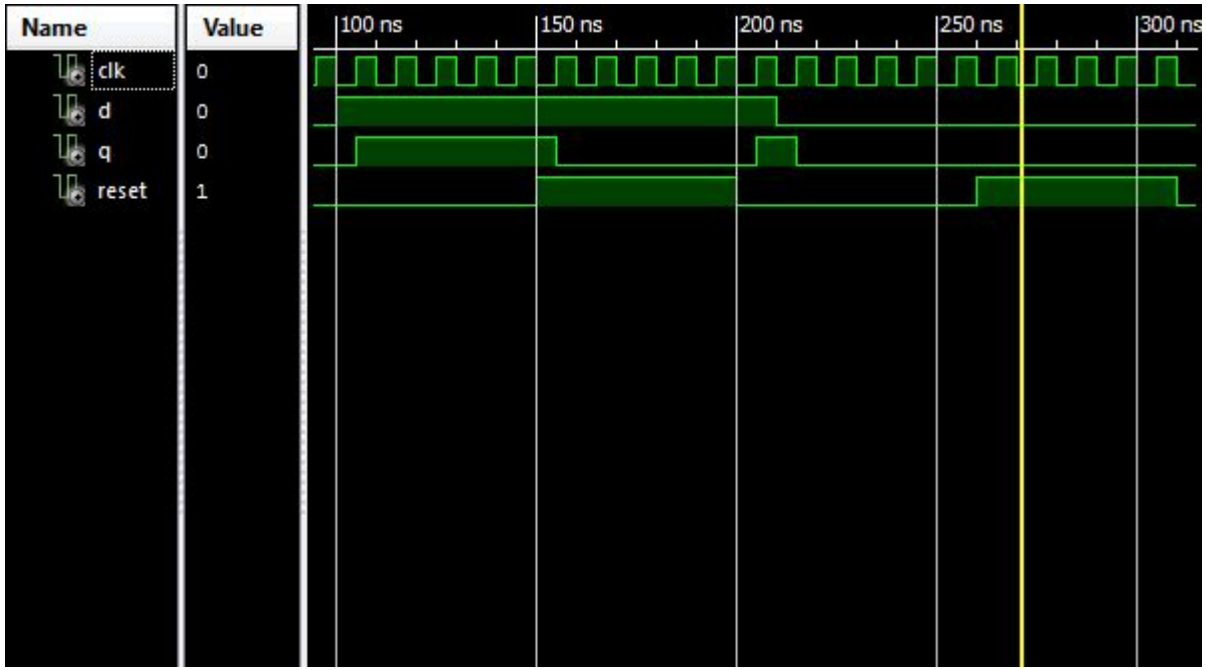
D триггер с сигналом разрешения

```
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 entity DtrigCE is
26     Port ( clk : in STD_LOGIC;
27            D : in STD_LOGIC;
28            CE : in STD_LOGIC;
29            Q : out STD_LOGIC);
30 end DtrigCE;
31
32 architecture Behavioral of DtrigCE is
33     signal Q_tmp : std_logic := '0';
34 begin
35
36     Process (clk)
37     Begin
38         If (rising_edge(clk)) then
39             If (CE = '1') then
40                 Q_tmp <= D;
41             End if;
42         End if;
43     End process;
44     Q <= Q_tmp;
45
46 end Behavioral;
```



D триггер с синхронным сбросом

```
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 entity DtrigSReset is
26     Port ( clk : in STD_LOGIC;
27            D : in STD_LOGIC;
28            reset : in STD_LOGIC;
29            Q : out STD_LOGIC);
30 end DtrigSReset;
31
32 architecture Behavioral of DtrigSReset is
33
34 begin
35
36     Process (clk)
37     Begin
38         If (rising_edge(clk)) then
39             If (reset = '1') then
40                 Q <= '0';
41             else
42                 Q <= D;
43             End if;
44         End if;
45     End process;
46
47 end Behavioral;
```

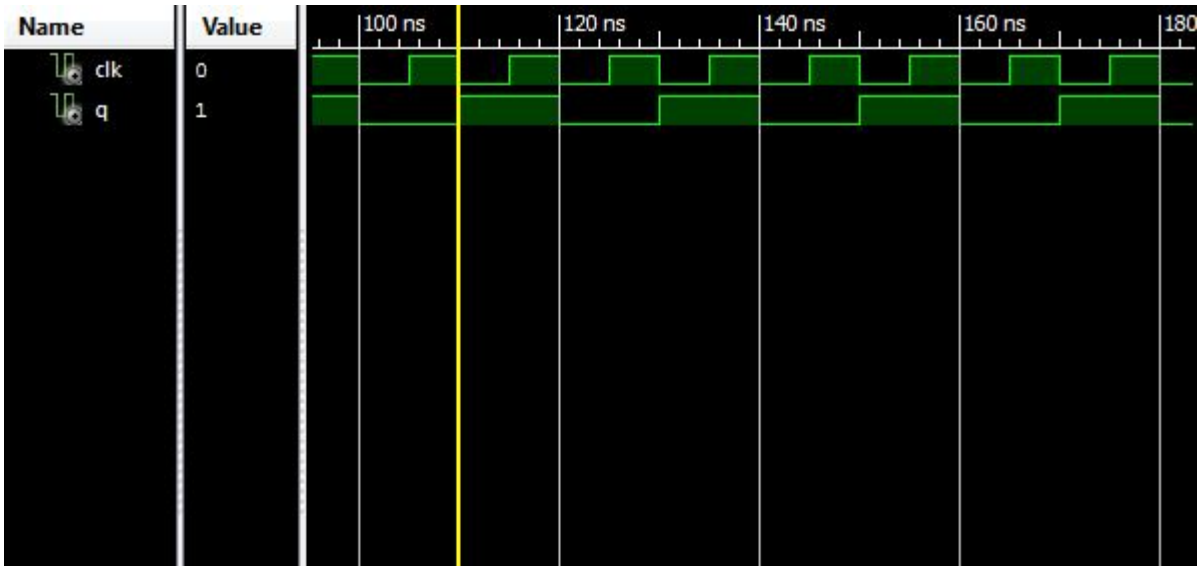


D триггер с асинхронным сбросом

```
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 entity DtrigAReset is
26     Port ( clk : in STD_LOGIC;
27            D : in STD_LOGIC;
28            reset : in STD_LOGIC;
29            Q : out STD_LOGIC);
30 end DtrigAReset;
31
32 architecture Behavioral of DtrigAReset is
33
34 begin
35
36     Process (clk, reset)
37     Begin
38         If (reset = '1') then
39             Q <= '0';
40         elsif (rising_edge(clk)) then
41             Q <= D;
42         End if;
43     End process;
44
45 end Behavioral;
```

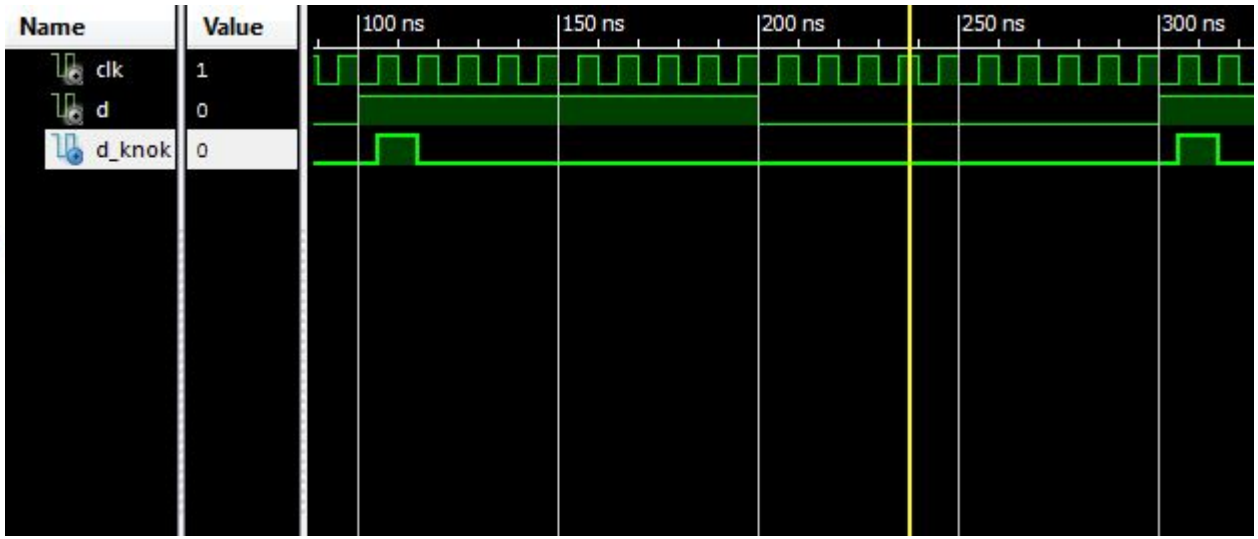
T триггер

```
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25
26 entity Ttrig is
27     Port ( clk : in STD_LOGIC;
28            Q : out STD_LOGIC);
29 end Ttrig;
30
31 architecture Behavioral of Ttrig is
32     Signal Q_tmp : std_logic := '0';
33 begin
34
35     Process (clk)
36     Begin
37         If (rising_edge(clk)) then
38             Q_tmp <= not Q_tmp;
39         End if;
40         Q <= Q_tmp;
41     End process;
42
43 end Behavioral;
```



Детектор фронта

```
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 entity EdgeDetect is
26     Port ( clk : in STD_LOGIC;
27            d : in STD_LOGIC;
28            d_knok : out STD_LOGIC);
29 end EdgeDetect;
30
31 architecture Behavioral of EdgeDetect is
32     Signal d_last : std_logic := '0';
33 begin
34
35     Process (clk)
36     Begin
37         If (rising_edge(clk)) then
38             d_last <= d;
39             d_knok <= d and not d_last ;
40         End if;
41     End process;
42
43 end Behavioral;
```

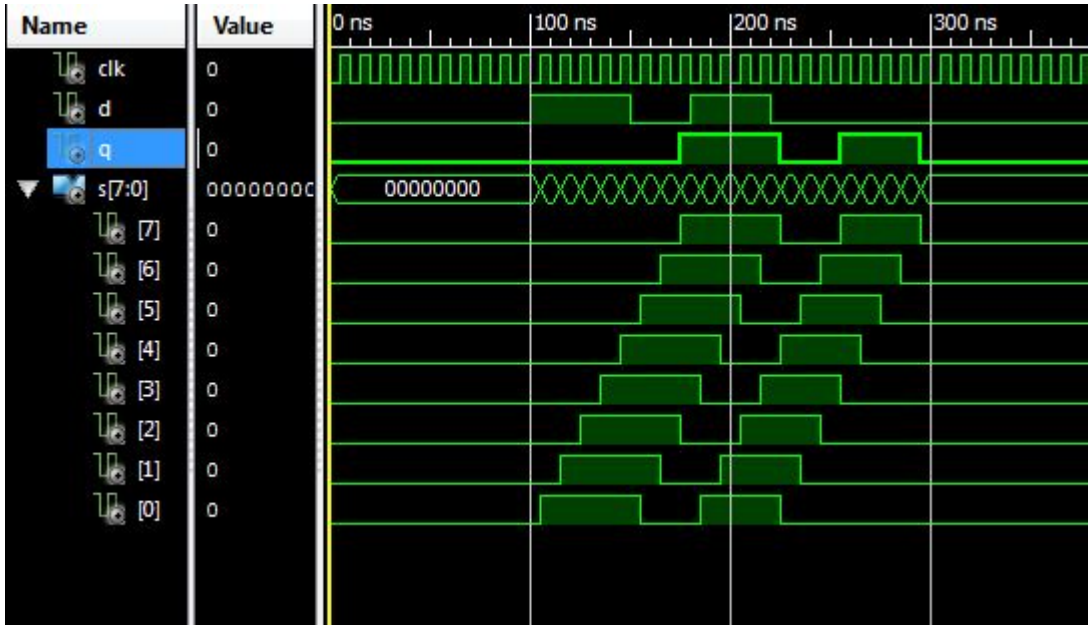


Задание

- Создать детектор спадов.
- Создать детектор фронтов и спадов.

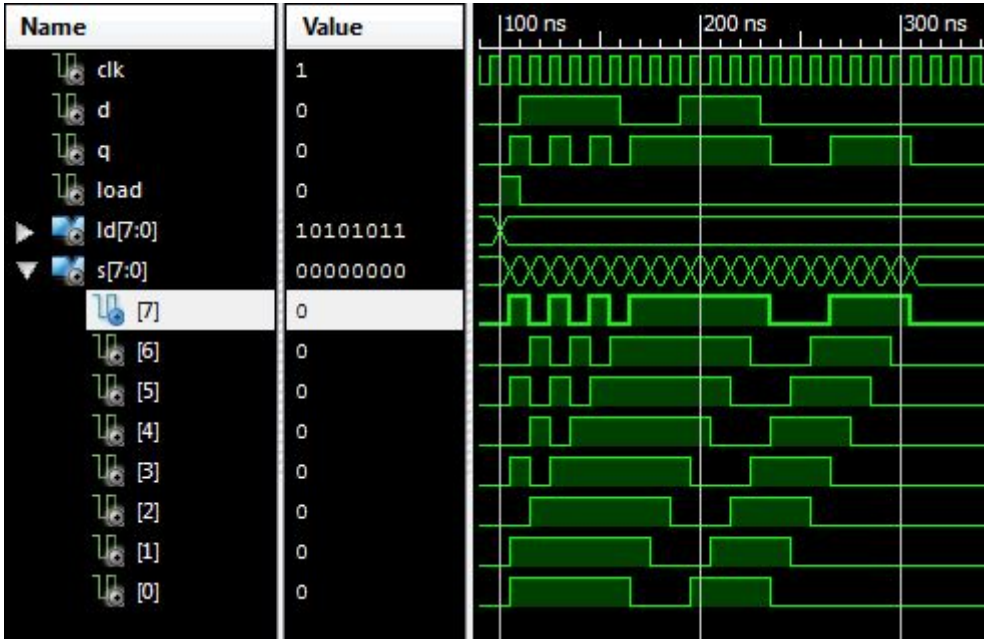
Сдвиговый регистр

```
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 entity SReg8 is
26     Port ( clk : in STD_LOGIC;
27            D : in STD_LOGIC;
28            Q : out STD_LOGIC);
29 end SReg8;
30
31 architecture Behavioral of SReg8 is
32 Signal S : std_logic_vector(7 downto 0)
33     := (others => '0');
34 begin
35     Process (clk)
36     Begin
37         If (rising_edge(clk)) then
38             S(7 downto 1) <= S(6 downto 0);
39             S(0) <= D;
40         End if;
41     End process;
42     Q <= S(7);
43 end Behavioral;
```



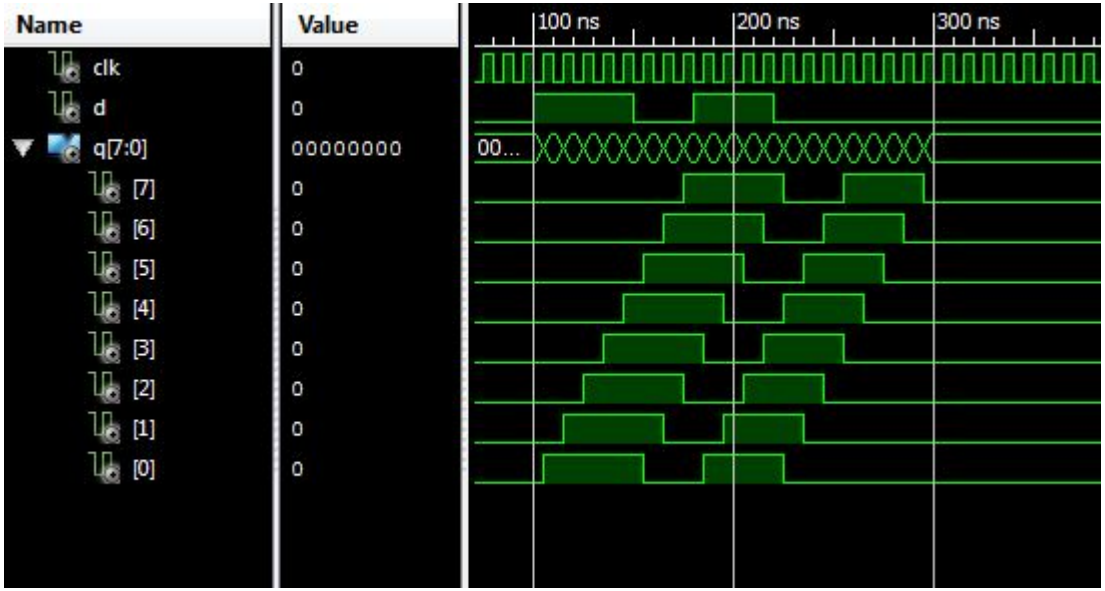
Сдвиговый регистр с параллельной загрузкой

```
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 entity SReg8_ParLoad is
26     Port ( clk : in STD_LOGIC;
27            D : in STD_LOGIC;
28            LD : in STD_LOGIC_VECTOR (7 downto 0);
29            Load : in STD_LOGIC;
30            Q : out STD_LOGIC);
31 end SReg8_ParLoad;
32
33 architecture Behavioral of SReg8_ParLoad is
34     Signal S : std_logic_vector(7 downto 0)
35         := (others => '0');
36 begin
37
38     Process (clk)
39     Begin
40         If (rising_edge(clk)) then
41             If (Load = '1') then
42                 S <= LD;
43             else
44                 S(7 downto 1) <= S(6 downto 0);
45                 S(0) <= D;
46             End if;
47         End if;
48     End process;
49     Q <= S(7);
50
51 end Behavioral;
```



Сдвиговый регистр с параллельным выходом

```
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 entity SReg8_ParOut is
26     Port ( clk : in STD_LOGIC;
27            D : in STD_LOGIC;
28            Q : out STD_LOGIC_VECTOR (7 downto 0));
29 end SReg8_ParOut;
30
31 architecture Behavioral of SReg8_ParOut is
32     Signal S : std_logic_vector(7 downto 0)
33         := (others => '0');
34 begin
35     Process (clk)
36     Begin
37         If (rising_edge(clk)) then
38             S(7 downto 1) <= S(6 downto 0);
39             S(0) <= D;
40         End if;
41     End process;
42     Q <= S;
43
44 end Behavioral;
```

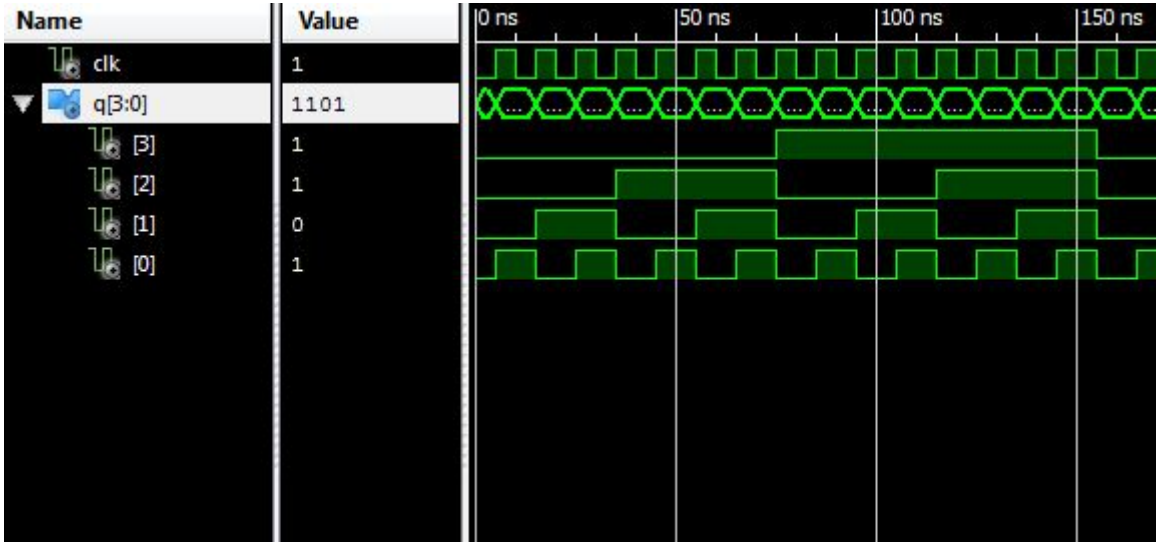


Задание

- Создать 8-ми разрядный сдвиговый регистр с сигналом разрешения на работу (CE).
- Создать 8-ми разрядный сдвиговый регистр с параллельной загрузкой.
- Создать 8-ми разрядный реверсивный сдвиговый регистр.

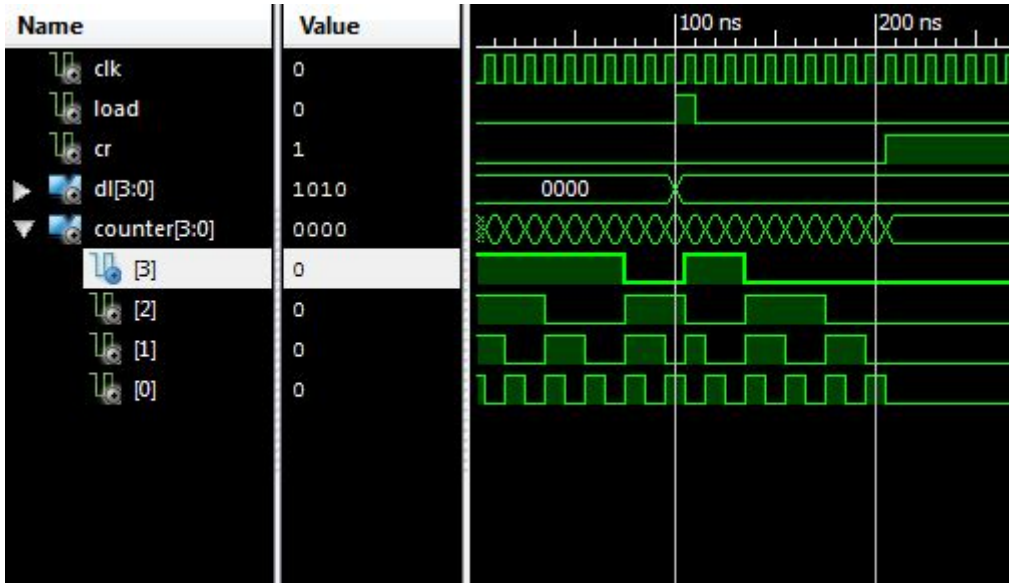
4-х разрядный счётчик

```
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24 use ieee.std_logic_arith.all;
25 use ieee.std_logic_unsigned.all;
26
27 entity Counter4 is
28     Port ( clk : in STD_LOGIC;
29            Q : out STD_LOGIC_VECTOR (3 downto 0));
30 end Counter4;
31
32 architecture Behavioral of Counter4 is
33     Signal counter : std_logic_vector(3 downto 0)
34         := (others => '0');
35 begin
36     Process (clk)
37     Begin
38         If (rising_edge(clk)) then
39             Counter <= counter + 1;
40         End if;
41     End process;
42     Q <= counter;
43 end Behavioral;
```



4-х разрядный реверсивный счётчик

```
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24 use ieee.std_logic_arith.all;
25 use ieee.std_logic_unsigned.all;
26
27 entity RevCounter4 is
28     Port ( clk : in STD_LOGIC;
29            load : in STD_LOGIC;
30            dl : in STD_LOGIC_VECTOR (3 downto 0);
31            CR : out STD_LOGIC);
32 end RevCounter4;
33
34 architecture Behavioral of RevCounter4 is
35     Signal counter : std_logic_vector(3 downto 0)
36         := (others => '1');
37 begin
38     Process (clk)
39     Begin
40         If (rising_edge(clk)) then
41             If (load = '1') then
42                 Counter <= dl;
43             Else
44                 If (counter /= 0) then
45                     counter <= counter -1 ;
46                 End if;
47             End if;
48         End if;
49     End process;
50     CR <= '1' when counter = 0 else '0';
51
52 end Behavioral;
```



Задание

- Создать двоично-десятичный счётчик на 4 десятичных разряда.
- Создать детектор фронта, выходной импульс которого длится:
  - 2 такта
  - 4 такта
  - 8 тактов
  - Может быть в пределах от 1 до 16.

Контрольные вопросы

- В чем разница между динамическим и статическим входом?
- В каких случаях у RS триггера появляются запрещённые состояния и почему?
- Каково назначение триггеров?
- Каков принцип работы сдвиговых регистров?
- Какие области применения сдвиговых регистров?