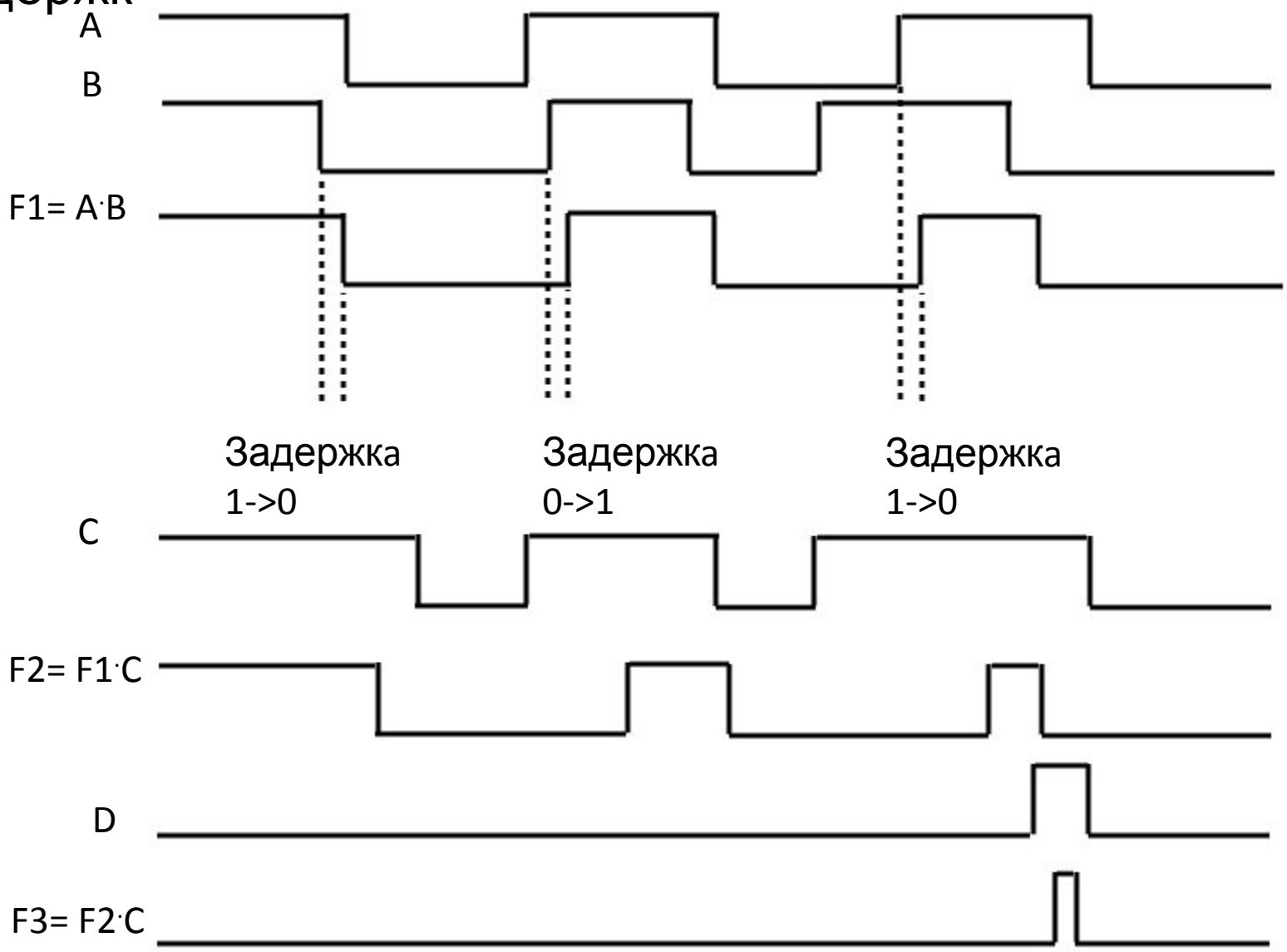
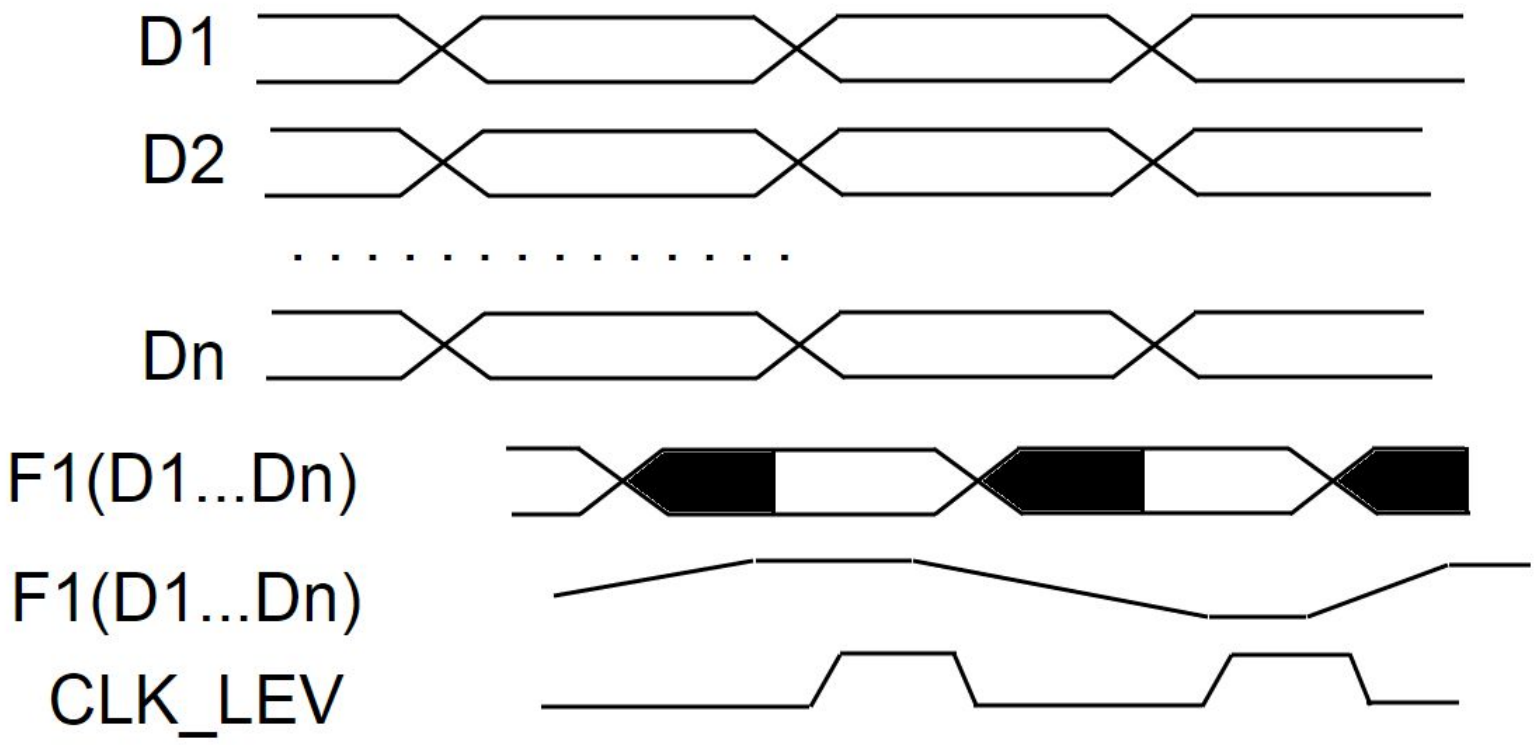
Лекция 4. Дискретные устройства. Синхронизаторы, триггеры, регистры, счетчики

Распространение сигналов. Быстродействие. Понятие о задержке



Сигналы A, B, C, D заданы, сигналы F1, F2, F3 получаются из них

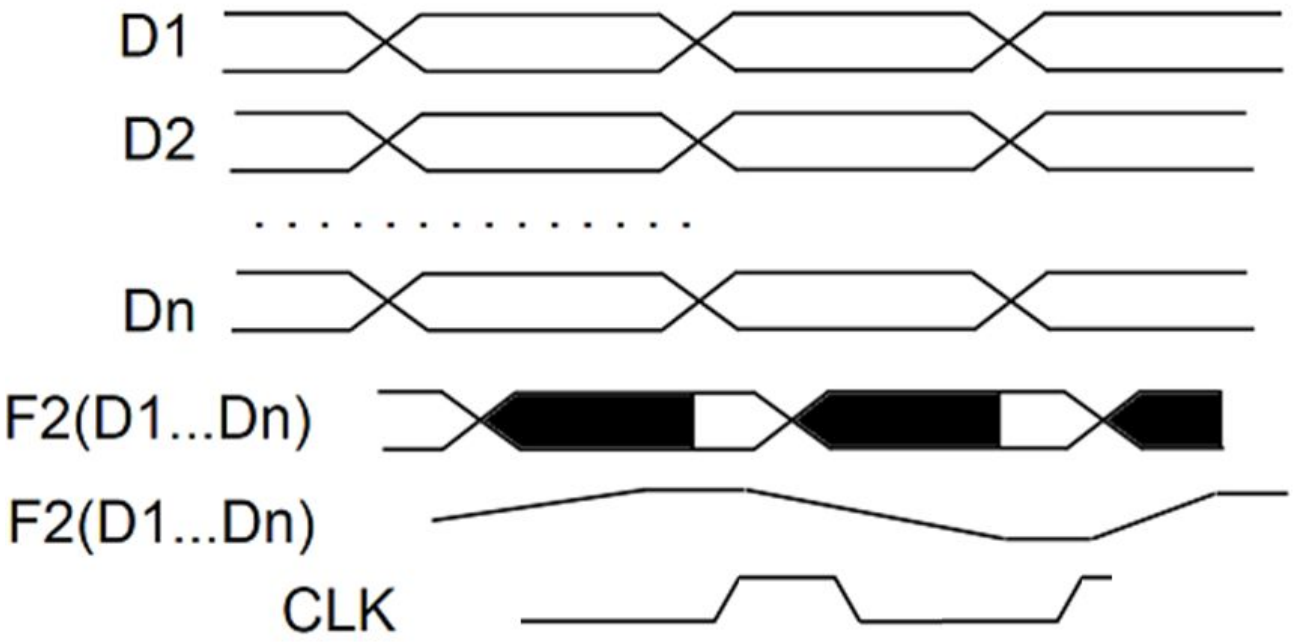
Распространение сигналов. Быстродействие. Понятие о задє



Короткий по времени фронт позволяет зафиксировать короткий сигнал

Фиксация сигнала по уровню будет или ресурсоемкой, или ненадежной

Распространение сигналов. Быстродействие. Понятие о задержке

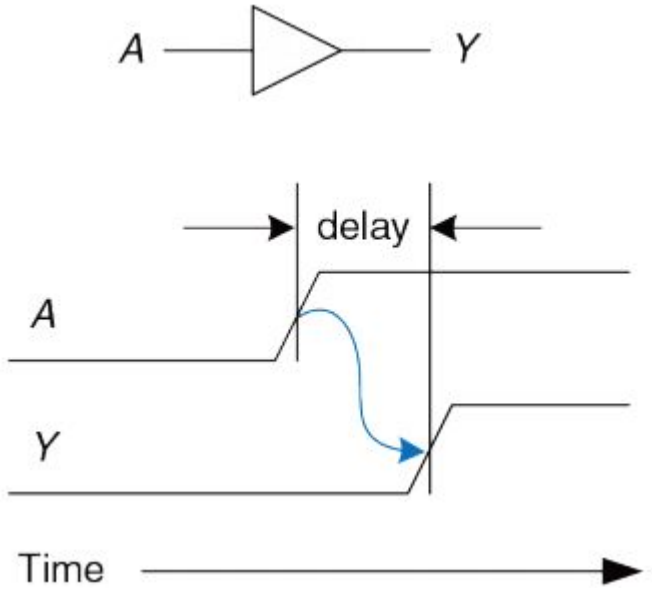


Короткий по времени фронт позволяет зафиксировать короткий сигнал

Фиксация сигнала по уровню будет или ресурсоемкой, или ненадежной

Распространение сигналов. Быстродействие. Понятие о задержке

Переход от НИЗКОГО уровня к ВЫСОКОМУ называется положительным перепадом или фронтом. Аналогично, переход от ВЫСОКОГО уровня к НИЗКОМУ называется соответственно отрицательным перепадом или срезом.

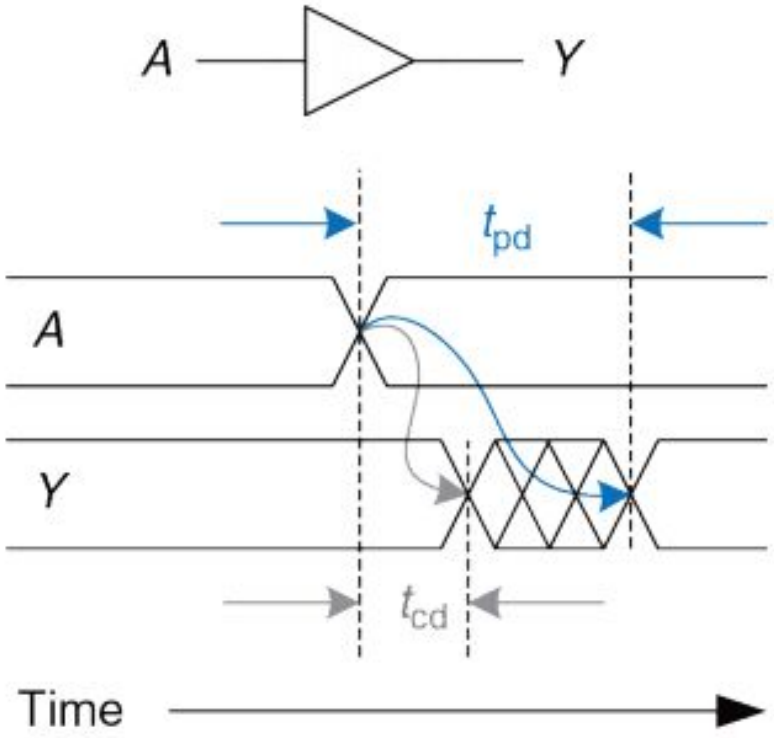


Распространение сигналов.
Быстродействие.
Понятие о задержке. Критические цепи

Комбинационная логика характеризуется задержкой распространения (propagation delay) и задержкой реакции, или отклика (contamination delay).

Задержка распространения t_{pd} – максимальное время от начала изменения входа до момента, когда все выходы достигнут установившихся значений.

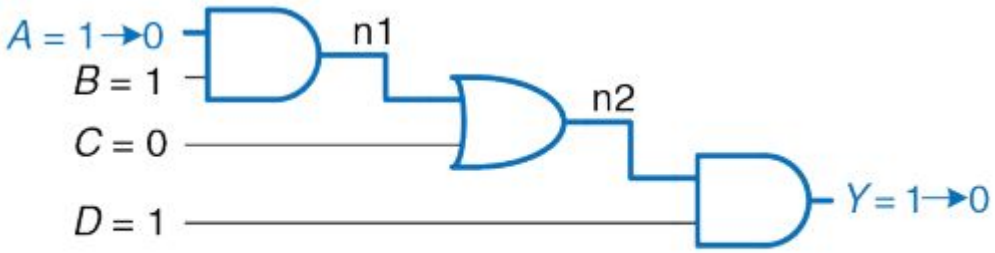
Задержка реакции t_{cd} – минимальное время от момента, когда вход изменился, до момента, когда любой из выходов начнет изменять свое значение

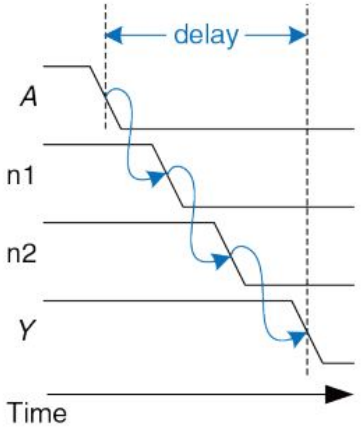


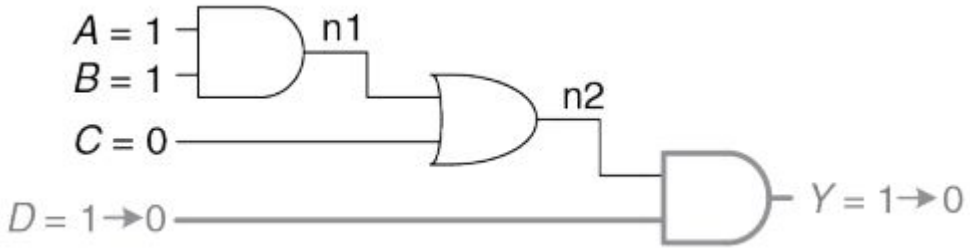
Распространение сигналов.

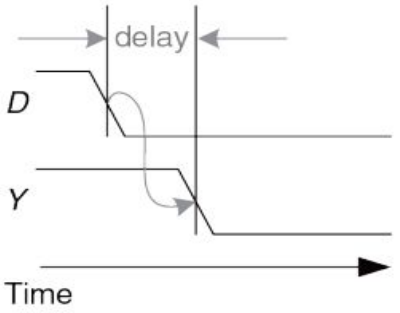
Быстродействие.

Понятие о задержке. Критические цепи.
Различные задержки в логическом устройстве









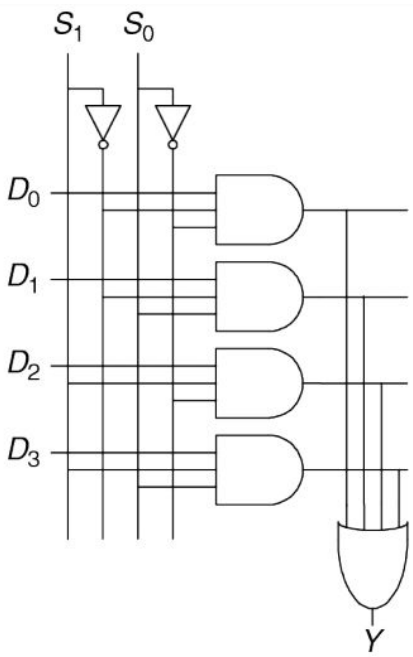
Распространение сигналов.

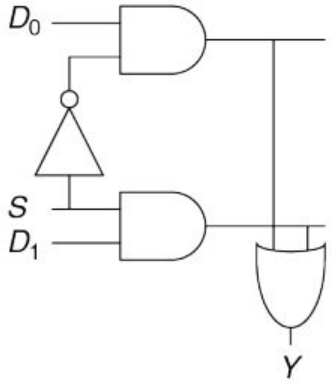
Быстродействие.

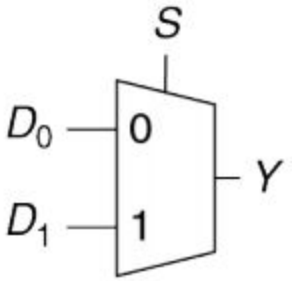
Понятие о задержке. Критические цепи.

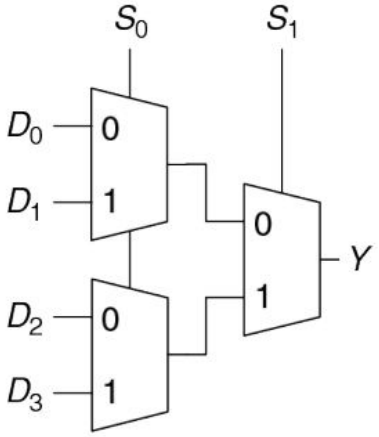
Варианты реализации мультиплексора:

где больше задержка?

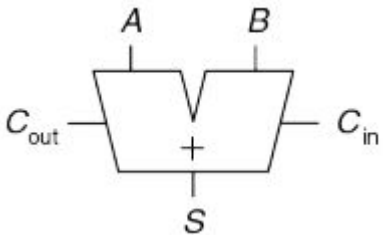








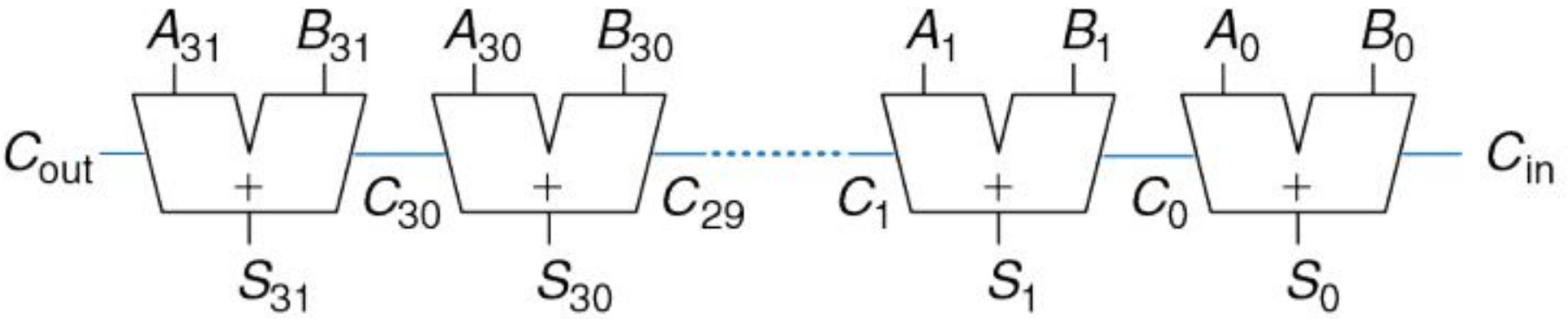
Критические цепи. Критической является цепь с самой
большой задержкой, например:



C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

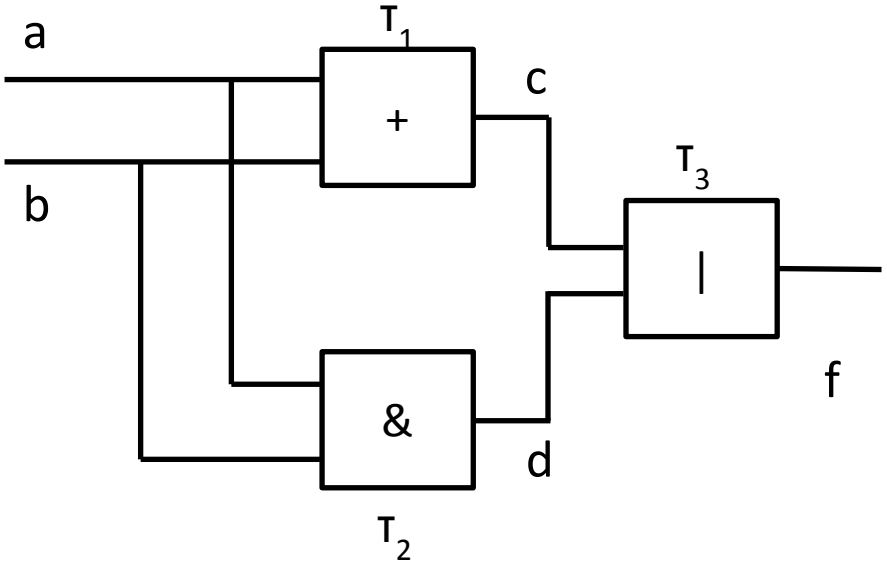
$$S = A \oplus B \oplus C_{in}$$

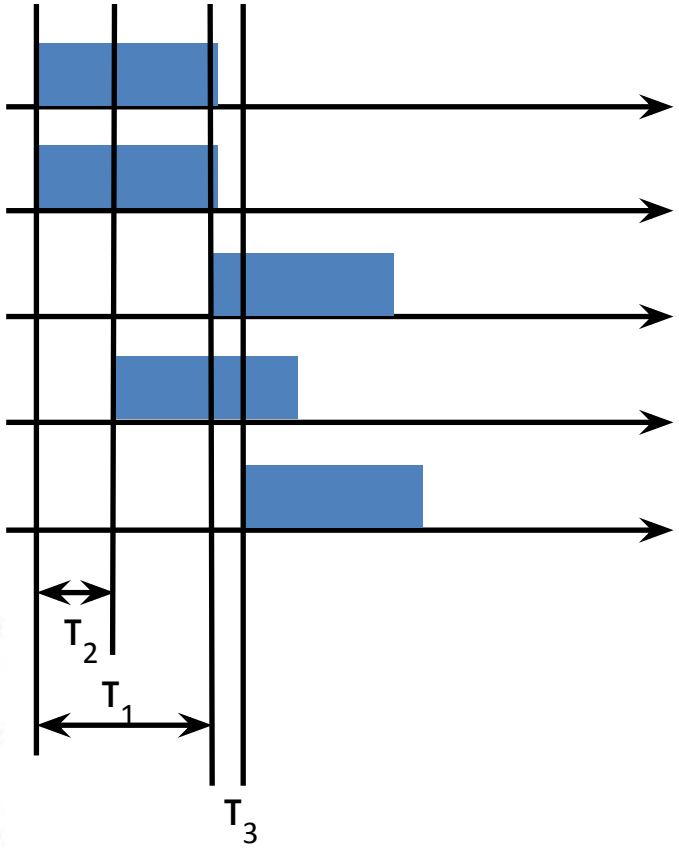
$$C_{out} = AB + AC_{in} + BC_{in}$$

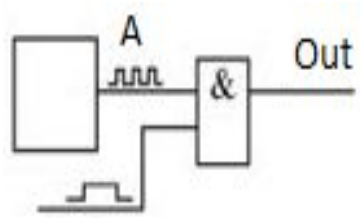


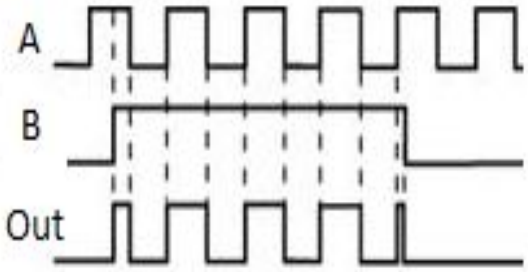
Цепь переноса в сумматоре является критической: в ней последовательно включено 2*n элементов, где n-разрядность.

Проблемы, требующие синхронизации:

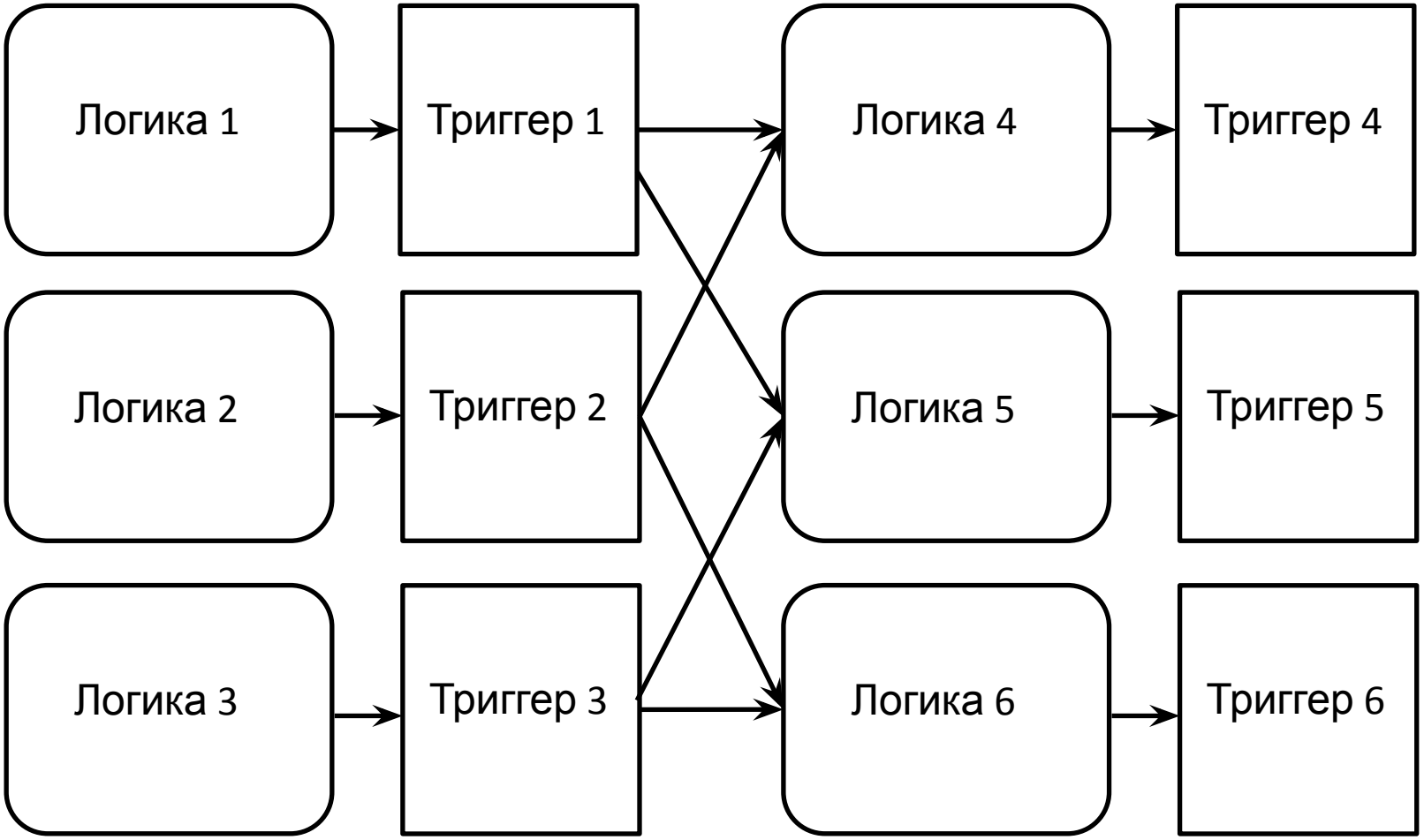




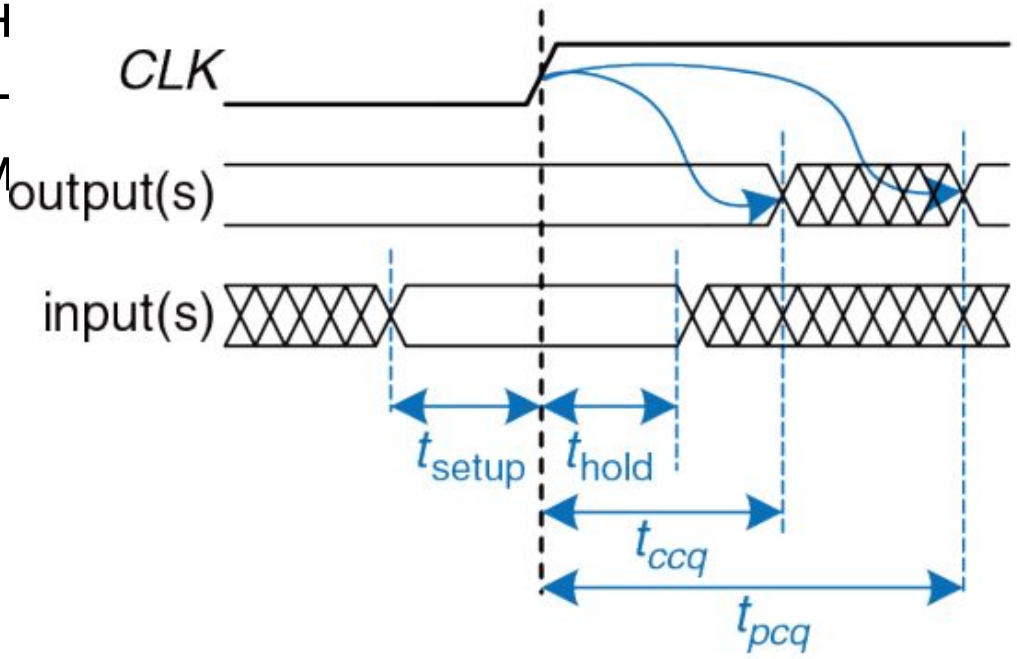




Базовый принцип синхронизации:



Динамическая дисциплина:
Входы синхронной последовательной схемы должны быть стабильны в течение времени предустановки до и
времени удержания после фронта тактового импульса.
Выполнение этих требований гарантирует, что в процессе
фиксации зн
триггером он
не будет изм



Метастабильность

Сам фронт сигнала тактовой частоты немного растянут во времени, да и триггер переключается не мгновенно, на его переключение требуется время.

Чтобы успешно зафиксировать входное значение в D-триггере, входной сигнал должен быть стабилен некоторое время до фронта и после фронта тактовой частоты.

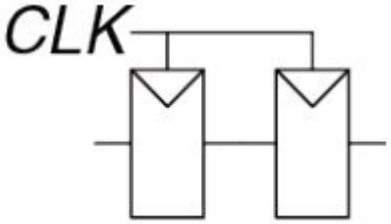
Если этого не произошло, триггер может оказаться в «промежуточном» состоянии, называемом метастабильным.

Метастабильное состояние триггера не является устойчивым.

После выхода из метастабильного состояния триггер оказывает произволь

Метастабильность

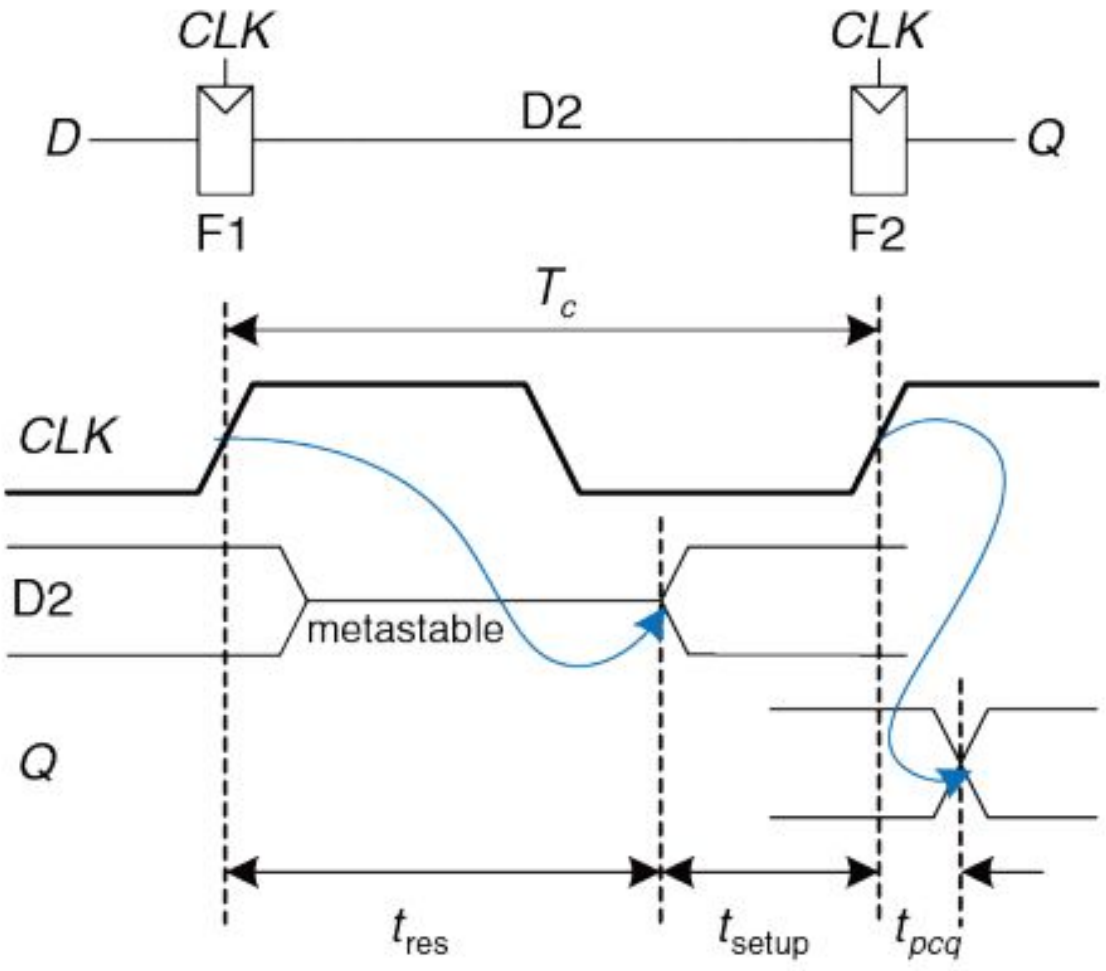
Попадание триггера в метастабильное состояние- вероятностный процесс, причем вероятность связана как с параметрами микросхемы, так и с тактовой частотой.
Вероятность этого события можно уменьшить, используя два последовательно включенных триггера, при этом вероятность того, что оба триггера попадут в метастабильное состояние, равна квадрату исходной вер



Если первый триггер и «поймает» метастабильное состояние, то вероятно оно должно пройти к моменту фиксации сигнала во втором триггере.

Простой синхронизатор и временная диаграмма его функционирования.

Триггер F1 фиксирует значение входного сигнала D по переднему фронту тактового сигнала CLK. Если D изменяется в апертурное время, его выход D2 на некоторое время может стать метастабильным. Если период тактового сигнала достаточно велик, то с высокой вероятностью до конца периода D2 придет к корректному логическому уровню. Триггер F2 затем фиксирует D2, который теперь стабилен, и формирует



Вероятность сбоя синхронизатора

Если D изменяется N раз за секунду, то вероятность ошибки за секунду составит

$$P(\text{failure})/\text{sec} = N\frac{T_0}{T_c}e^{-\frac{T_c - t_{setup}}{\tau}}$$

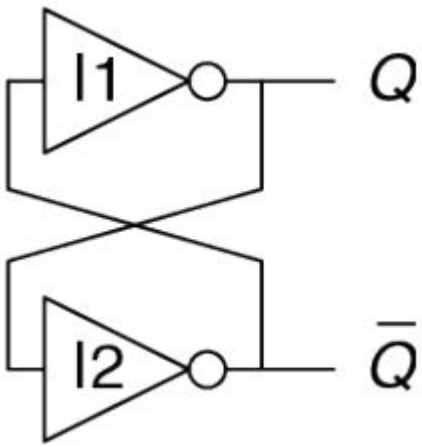
А среднее время наработки на отказ (mean time between failures, MTBF) составит

$$MTBF = \frac{1}{P(\text{failure})/\text{sec}} = \frac{T_c e^{\frac{T_c - t_{setup}}{\tau}}}{NT_0}$$

MTBF растет экспоненциально с ростом времени ожидания синхронизатора T_c . Для большинства систем синхронизатор, который ожидает один период тактового сигнала, обеспечивает достаточную величину MTBF. В высокоскоростных системах может понадобиться ожидание на большее количество периодов тактового сигнала.

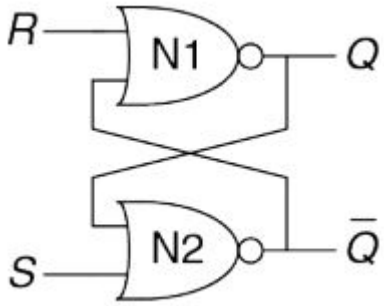
Дискретные устройства- триггеры
Дискретное (последовательное) цифровое устройство:
Выходное состояние устройства зависит от текущего состояния и
предыдущих входных воздействий.

Бистабильная защел

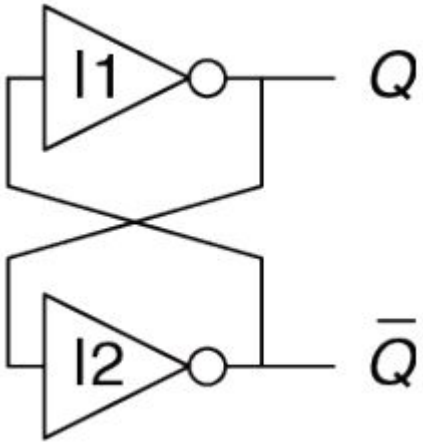


Дискретные устройства- триггеры
Дискретное (последовательное) цифровое устройство:
Выходное состояние устройства зависит от текущего состояния и
предыдущих входных воздействий.

R-S триггер:

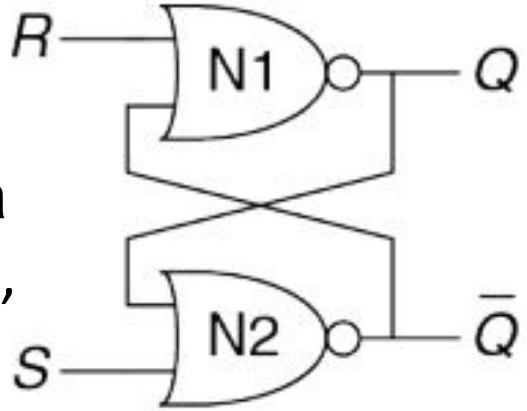


Случай 1: Q=0
На вход поступает сигнал Q = 1. I2 инвертирует сигнал и подает
на вход I1 сигнал Q^{-} = 1. Соответственно, на выходе I1 – логический 0. В рассмотренном случае схема
в стабильном состоянии.
Случай II: Q=1
На вход I2 поступает 1 (Q). I2 инвертирует сигна
и подает на вход I1 «0».
Соответственно, на выходе I1 – логическая 1.
В этом случае схема также находится в стабильном состоянии.

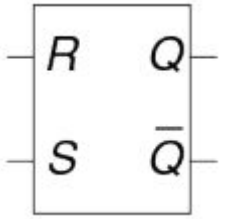


Так как инверторы, включенные перекрестно, имеют два стабильных состояния Q = 0 и Q = 1, то говорят,
что схема бистабильна.

R-S триггер



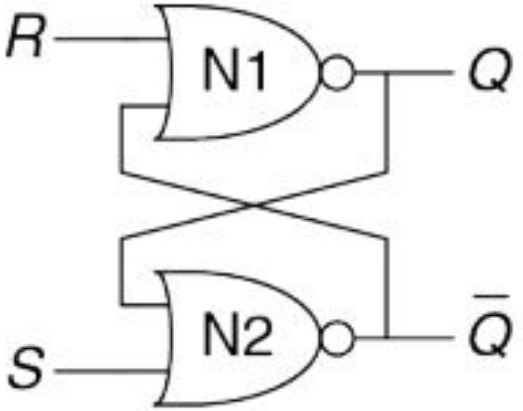
Случай I: R=1, S=0
На входе N1 как минимум одна единица – вход R, следовательно, выход Q=0. Оба входа N2 – в состоянии логического нуля (Q=0 и S=0), поэтому выход =1.

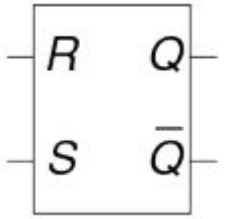


Случай II: R=0, S=1
На вход N1 поступает 0 и Q^- . Так как мы еще не знаем значения Q^- , мы не можем определить значение Q.
На вход N2 поступает как минимум одна единица S, поэтому на выходе Q^- нуль. Теперь можно вернуться к определению состояния выхода элемента N1.
Мы знаем, что на обоих его входах 0, следовательно, Q=1.

R-S триггер

Случай III: R=1, S=1
Запрещенная комбинация входных воздействий.
Как на входе N1, так и на входе N2 как минимум
по одной единице (R и S), поэтому на выходе каждой защелки – логический 0.
Следовательно, Q=0 и Q^{-} =0.





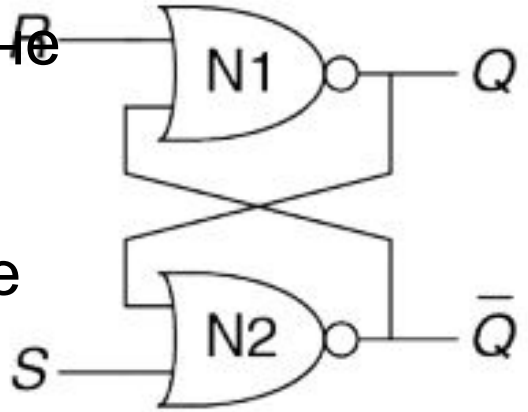
Случай IV: R=0, S=0

На вход N1 поступает 0 и Q^- . Так как мы еще не знаем значения Q^-, мы не можем определить значение на выходе элемента N1.

На вход N2 поступает 0 и Q. Так как мы еще не знаем значения Q, мы не можем определить значение на выходе элемента N2.

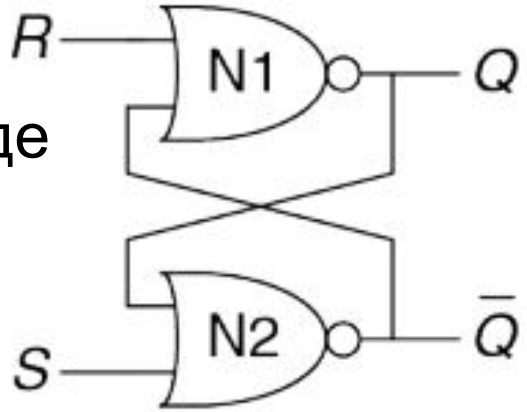
Этот случай аналогичен случаю с двумя перекрестно
включенными инверторами. Мы знаем, что Q должен
быть равен либо 0, либо 1.

Рассмотрим каждый из этих двух случаев.



Случай IVa: Q=0

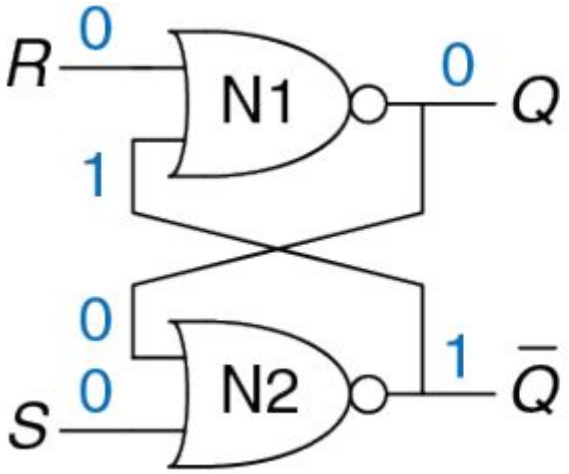
Так как S и Q равны 0, то на выходе N2 будет логическая 1, Q^- =1. Теперь на входе N1 есть одна единица – Q^-, поэтому на его выходе Q=0.

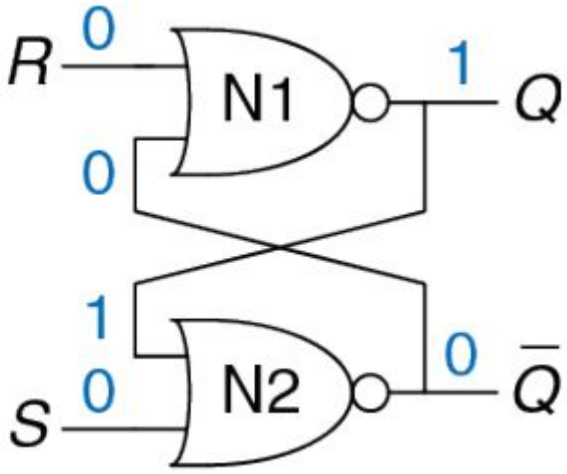


Случай IVb: Q=1

Так как Q = 1, то на выходе N2 будет 0, Q^- =0. Теперь на обоих входах N1 нули (R и Q^-), поэтому на его выходе логическая 1, Q=1.

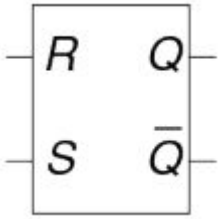
Устрой





R-S триггер, УГО и таблица истинности

Case	S	R	Q	$\overline{Q}$
IV	0	0	Q_{prev}	$\overline{Q}_{prev}$
I	0	1	0	1
II	1	0	1	0
III	1	1	0	0

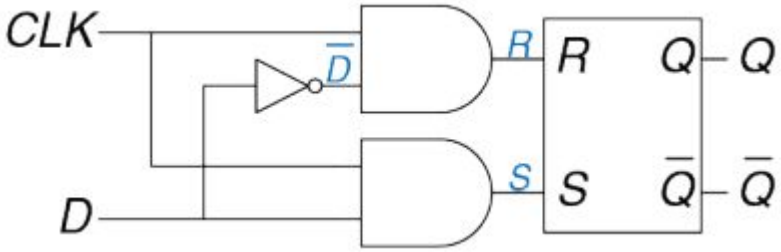


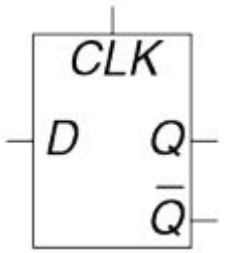
RS-триггер- бистабильный элементом с одним битом состояния,
хранящимся в Q.
Состоянием можно управлять при помощи входов R и S.
Когда на R поступает высокий уровень, выход сбрасывается в 0.
Когда высокий уровень приходит на S, выход устанавливается в 1.

D триггер с синхронизацией по уровню

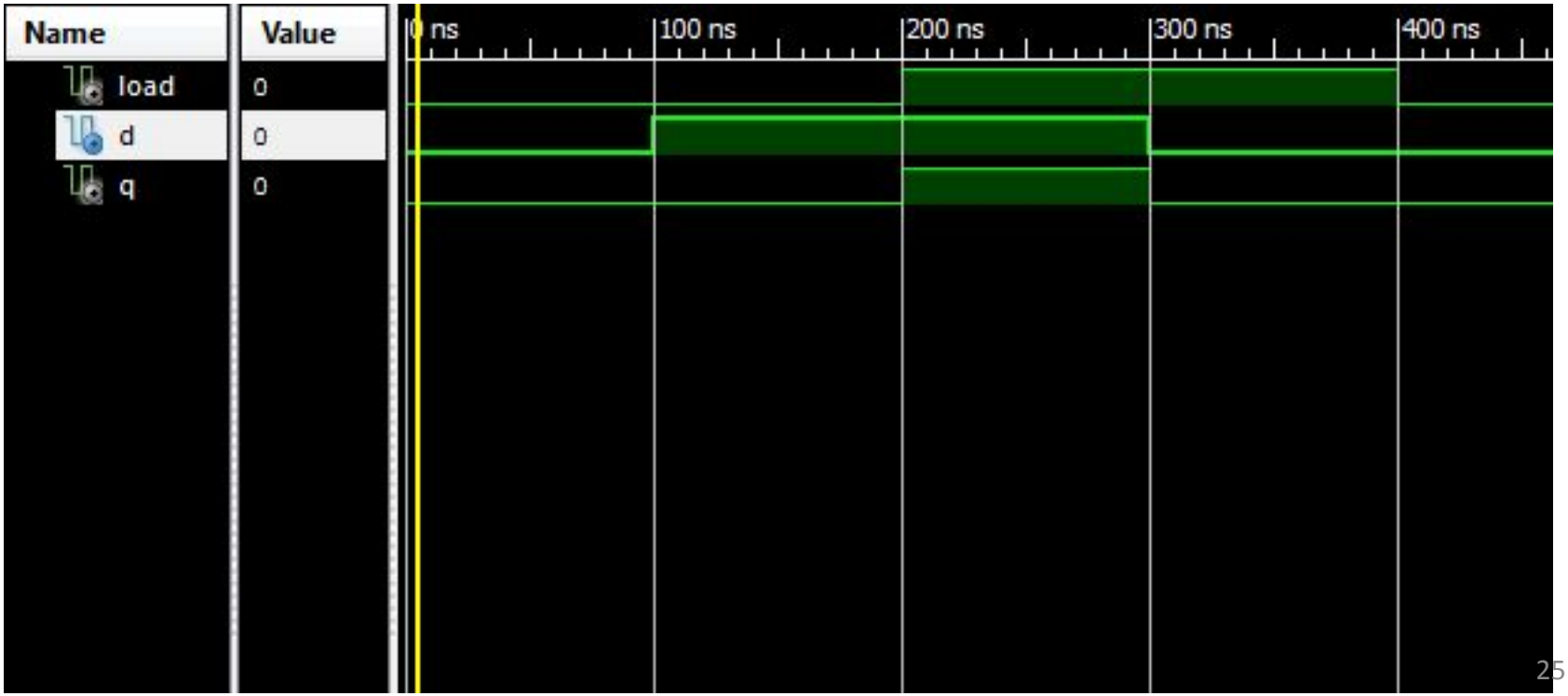
Сигнал D записывается в R-S триггер

по разрешению от сигнала CLK





CLK	D	D	S	R	Q	Q
0	X	$\overline{X}$	0	0	Q_{prev}	$\overline{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

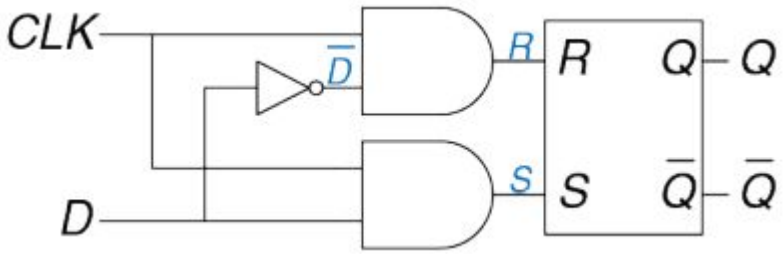


D триггер с синхронизацией по уровню

Сигнал D записывается в R-S триггер

- сигнала CLK

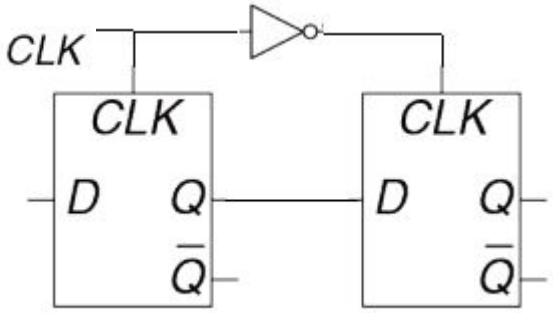
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Latch is
    Port ( load : in STD_LOGIC;
           D : in STD_LOGIC;
           Q : out STD_LOGIC);
end Latch;

architecture Behavioral of Latch is
    signal q_tmp : std_logic := '0';
begin

    Process (load, D    )
    Begin
        If (load = '1') then
            q_tmp <= D;
        End if;
    End process;
    Q <= q_tmp;

end Behavioral;
```

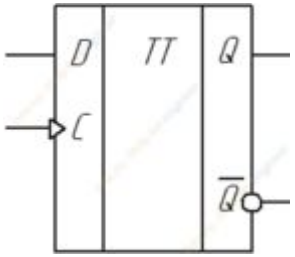


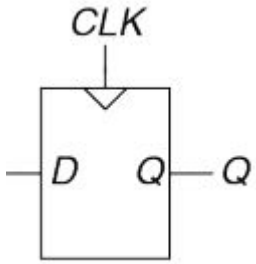
D триггер работающий по фронту

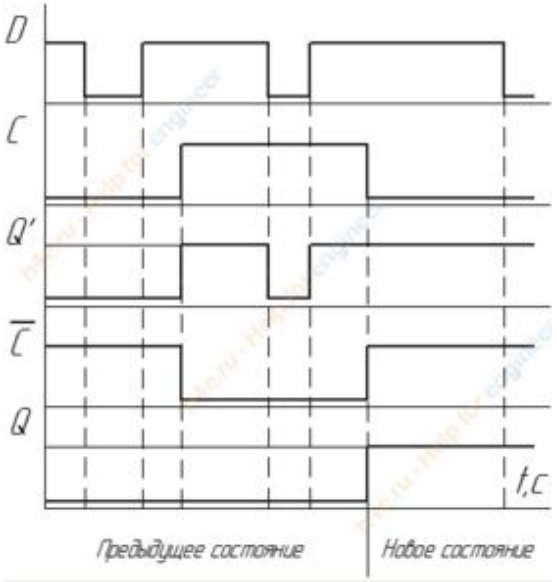
Из двух D триггеров, работающих по уровню, можно построить D триггер, работающий по фронту тактового сигнала.

Полученное устройство будет переключаться по фронту нарастания сигнала.



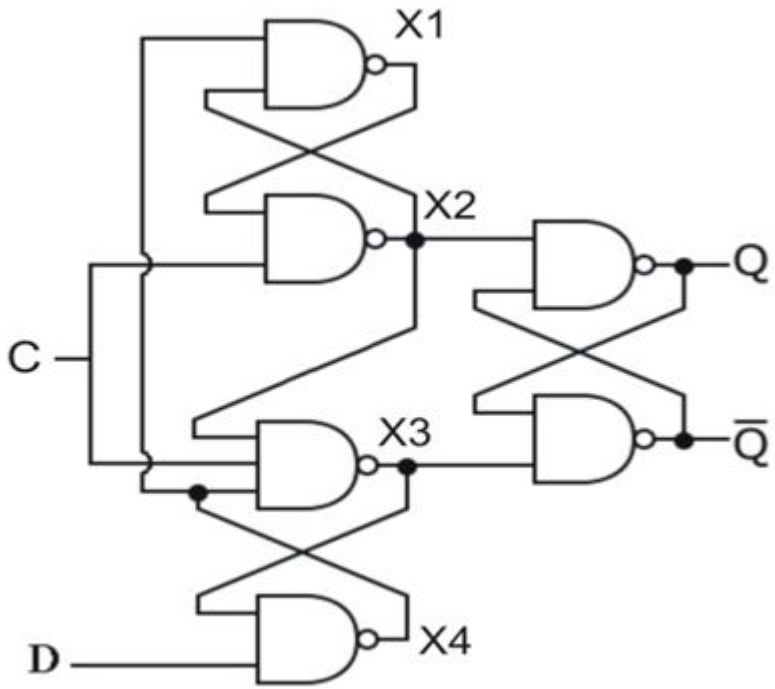






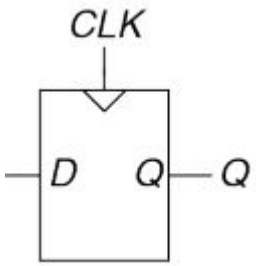
D триггер работающий по фронту

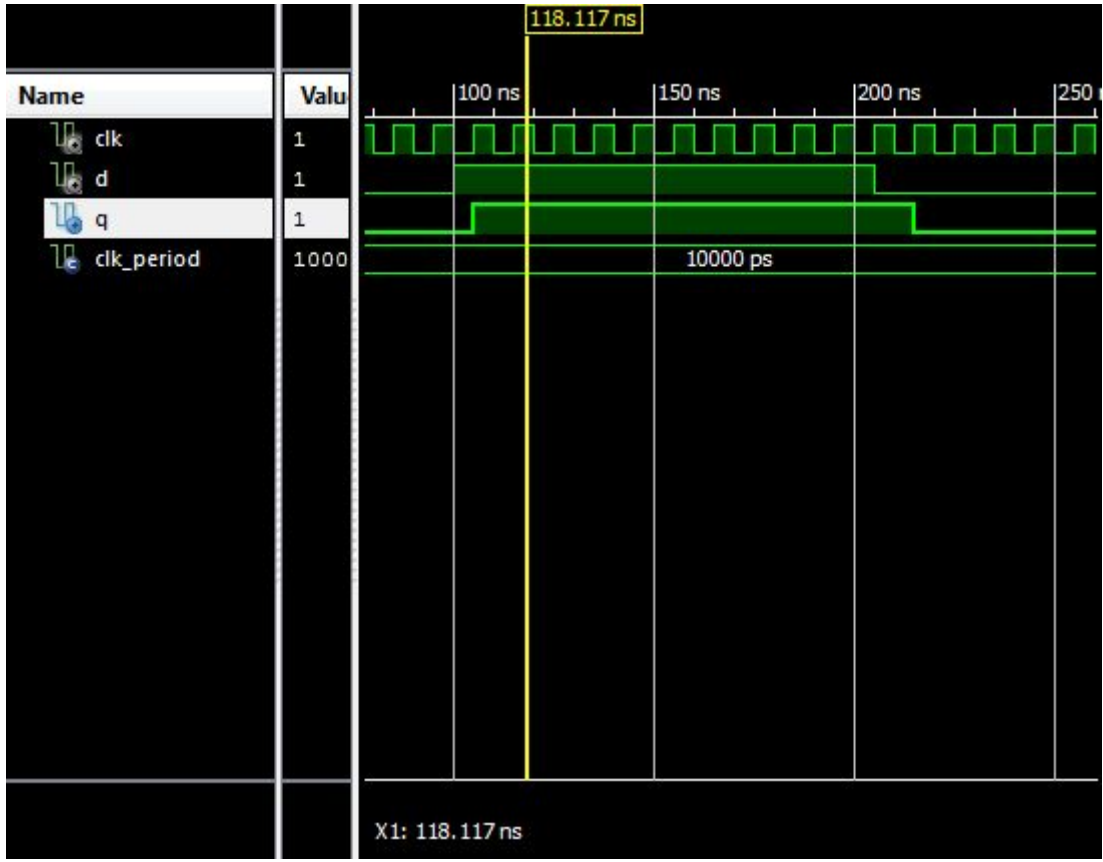
Другой вариант реализации



D триггер работающий по фронту, VHDL описание и временная диаграмма

```
Process (clk)
  Begin
   if (clk='0' and c'event) then
   q<=d;
   End if;
 End process;
```

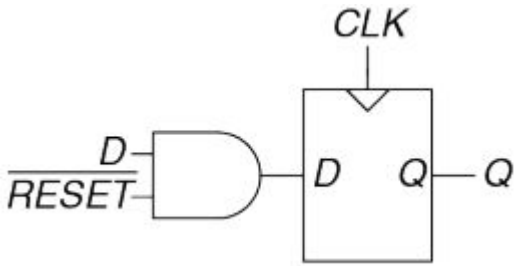


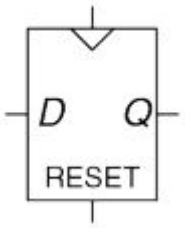


D триггер со сбросом

В триггере с функцией сброса добавляется вход RESET (сброс). Когда на RESET подан 0, триггер ведет себя как обычный D-триггер.
Когда на RESET подана 1, такой триггер игнорирует вход D и сбрасывает выход в 0.

```
process (clk) begin
 if (clk'event and clk = '1') then
   if (reset = '1') then
     q <= '0';
     else q <= data;
   end if;
  end if;
end process;
```



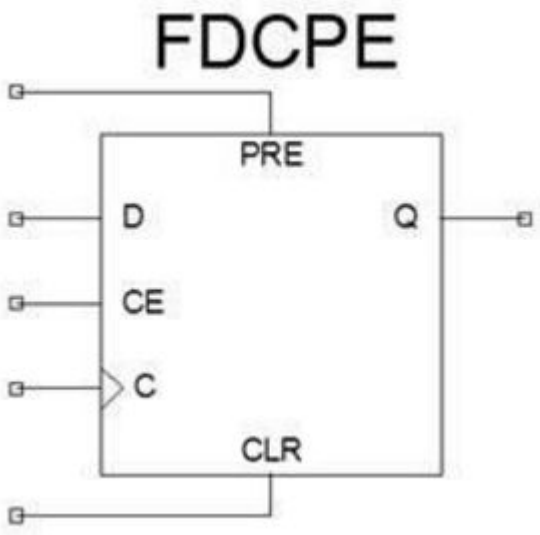


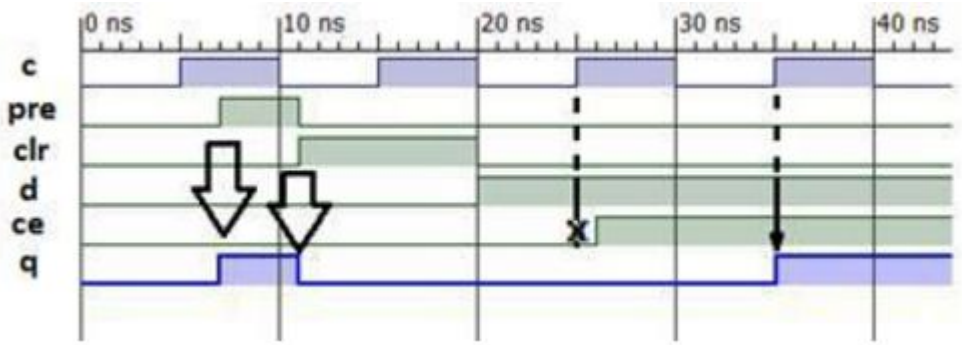
D триггер со сбросом и установкой

В триггере с функцией установки добавляется вход SET. Когда на SET подан 0, триггер ведет себя как обычный D-триггер. Когда на SET подан 1, такой триггер игнорирует вход D и устанавливает выход в 1.

```
process (clk) begin
if (clk'event and clk = '1') then
   if (reset = '1') then  q <= '0';
   if (set = '1') then  q <= ‘1';
     else q <= data;
-- r=q=0 – хранение данных
end if; end if; end if;
end process;
```

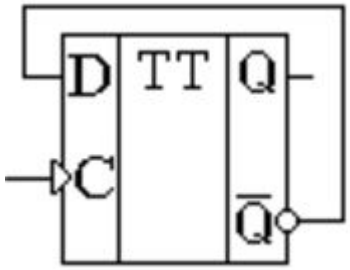
D триггер со сбросом, установкой и разрешением (вход CE-разрешение на запись)

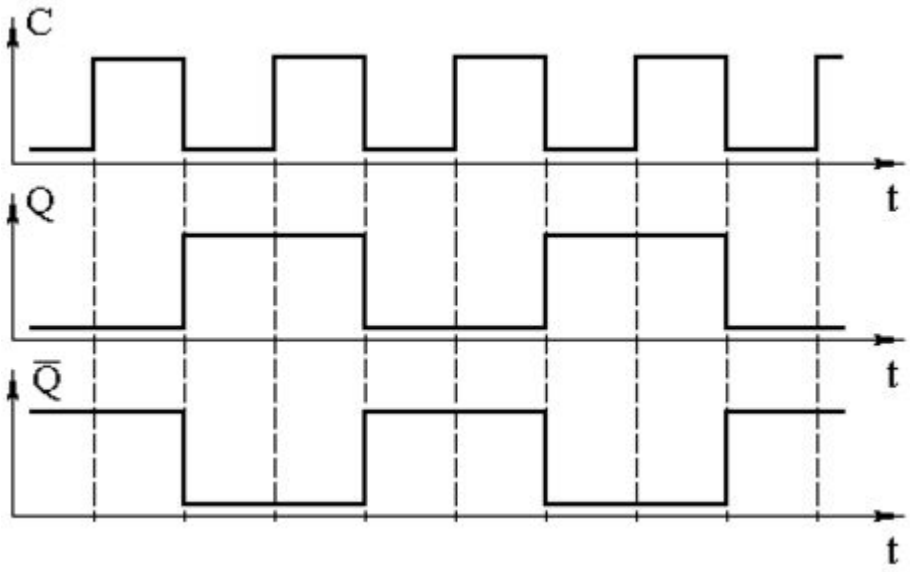




Счетный триггер

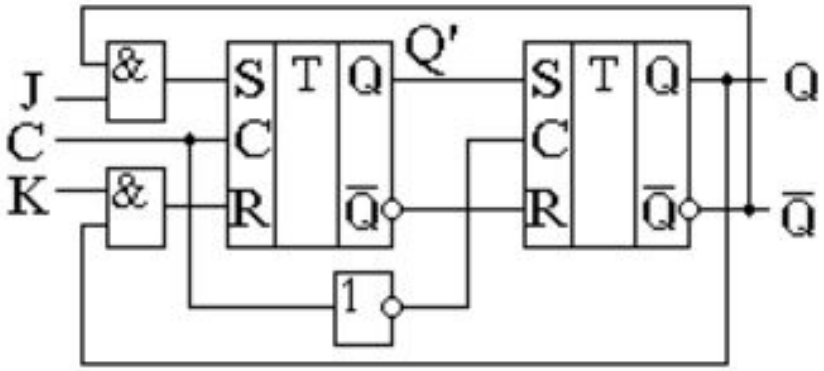
D- триггер, у которого инверсный выход подключен к входу, называется Т-триггером. После поступления на этот вход импульса, состояние Т-триггера меняется на прямо противоположное. Счётным он называется потому,
что подсчитывает количество импульсов, поступивших на его
вход. Считать этот триггер умеет
только до одного. При поступлении второго импульса

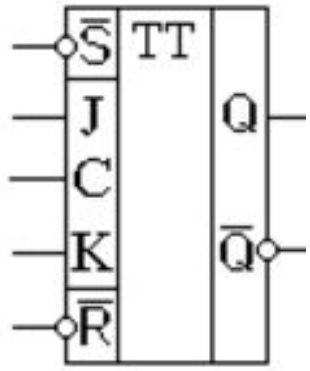




JK- триггер.

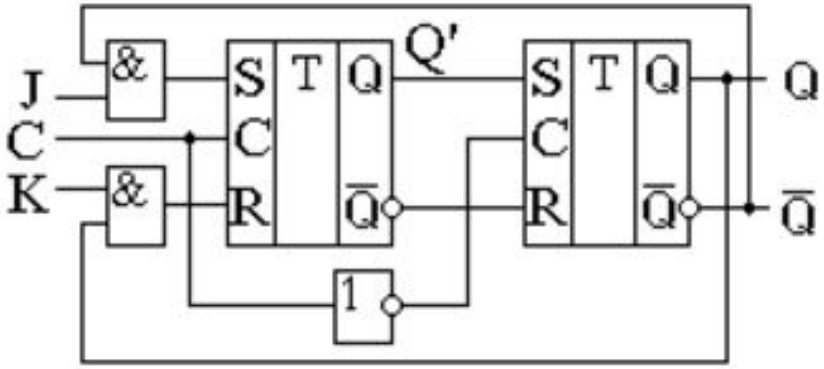
Таблица истинности JK-триггера практически совпадает с таблицей истинности синхронного RS-триггера. Для того чтобы исключить запрещённое состояние, схема триггера изменена таким образом, что при подаче двух единиц JK-триггер превращается в счётный триггер. Это означает, что при подаче на тактовый вход C импульсов JK-триггер изменяет своё состояние на противоположное.



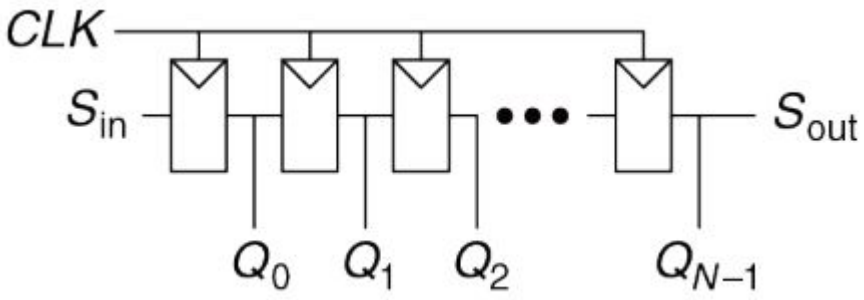


K	J	Q(t)	Q(t+1)	Пояснения
x	x	0	0	Режим хранения информации
x	x	1	1	
0	0	0	0	Режим хранения информации
0	0	1	1	
0	1	0	1	Режим установки единицы J=1
0	1	1	1	
1	0	0	0	Режим записи нуля K=1
1	0	1	0	
1	1	0	1	K=J=1 счетный режим триггера
1	1	1	1	

JK- триггер.



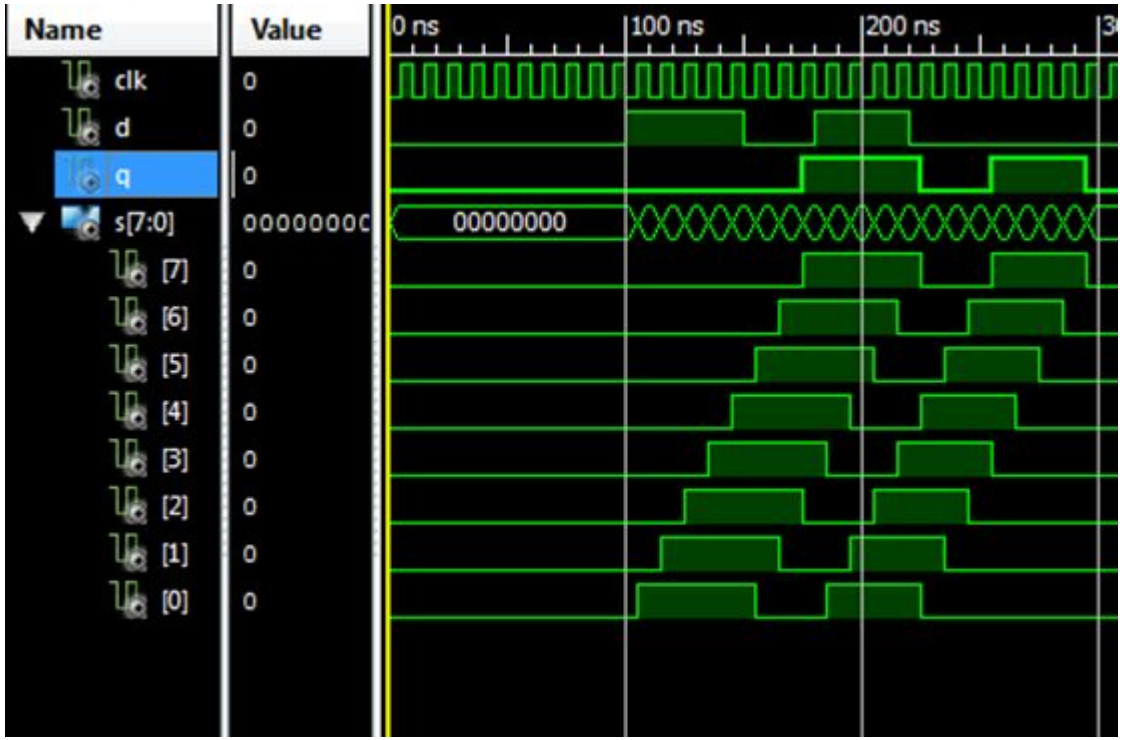
Сдвиговый регистр



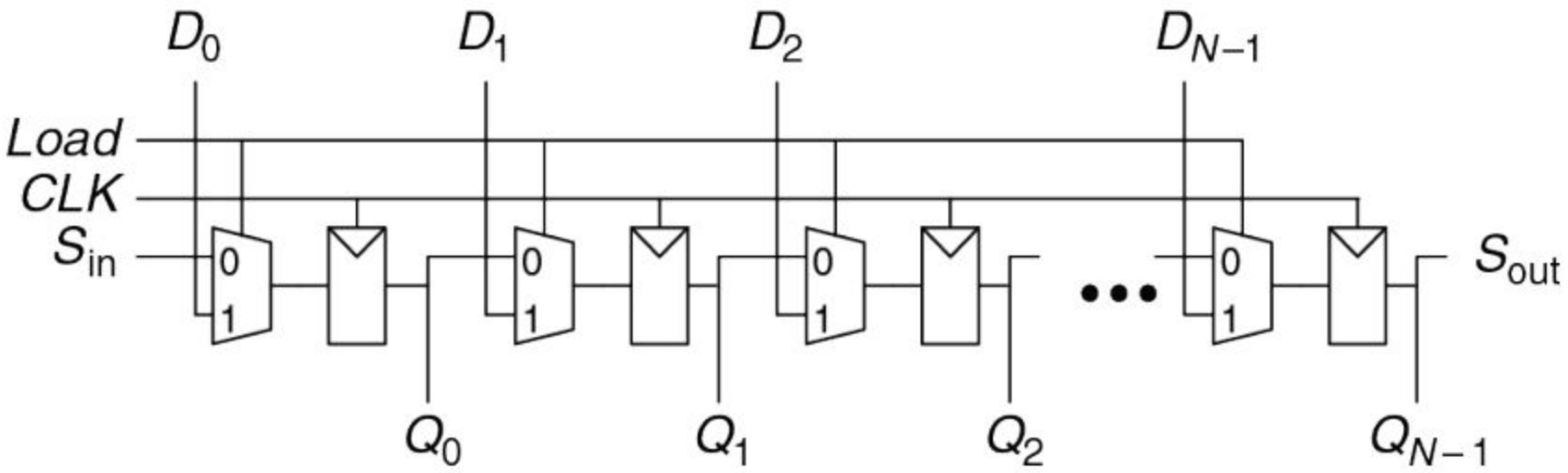
Сдвиговые регистры представляют собой последовательно соединенную цепочку триггеров. Основной режим их работы - это сдвиг разрядов кода, записанного в эти триггеры, То есть по тактовому сигналу содержимое каждого предыдущего триггера переписывается в следующий по порядку в цепочке триггер. Сдвиг бывает двух видов: вправо - это основной режим, который есть у всех сдвиговых регистров, и влево - этот режим есть только у некоторых реверсивных сдвиговых

Сдвиговый регистр

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

entity SReg8 is
    Port ( clk : in STD_LOGIC;
           D : in STD_LOGIC;
           Q : out STD_LOGIC);
end SReg8;

architecture Behavioral of SReg8 is
Signal S : std_logic_vector(7 downto 0)
    := (others => '0');
begin
    Process (clk)
    Begin
        If (rising_edge(clk)) then
            S(7 downto 1) <= S(6 downto 0);
            S(0) <= D;
        End if;
    End process;
    Q <= S(7);
end Behavioral;
```



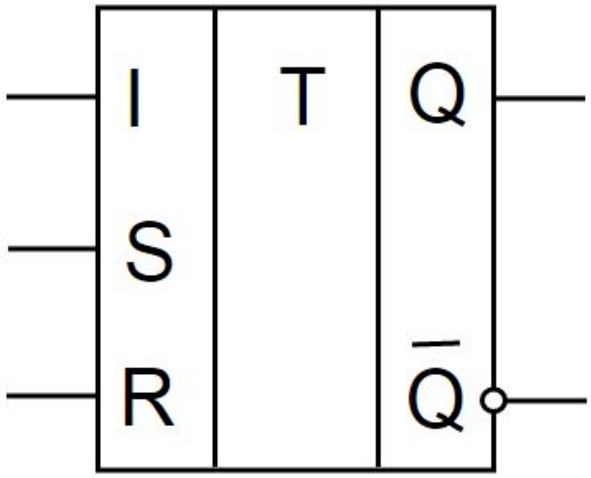
Сдвиговый регистр с параллельной загрузкой



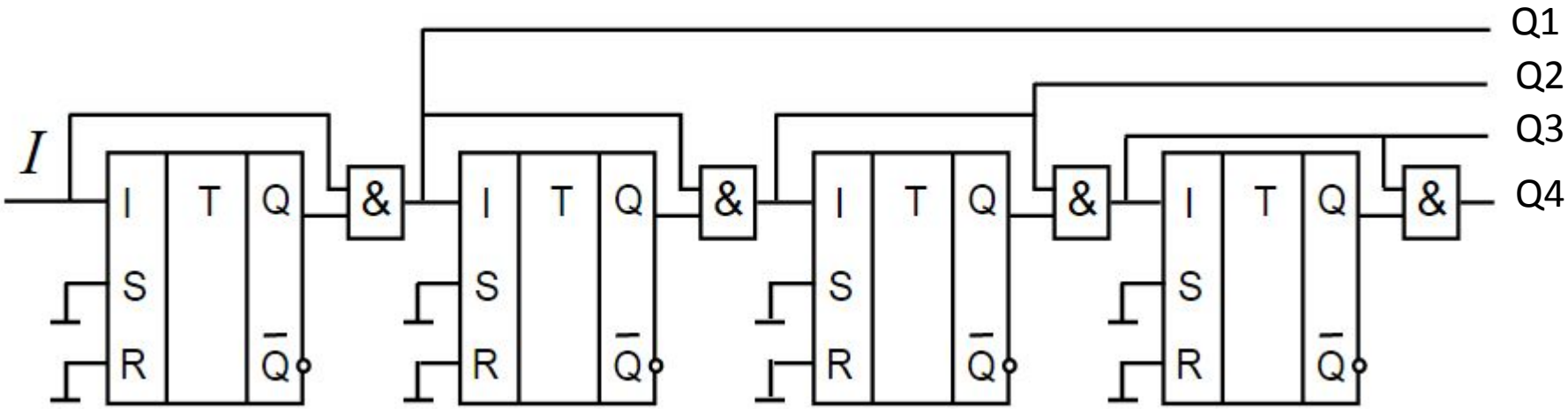
Сдвиговый регистр с параллельной загрузкой

```
library IEEE; use IEEE.STD_LOGIC_1164.ALL;
entity shiftreg is
 generic(N: integer := 8);
 port(clk, reset: in STD_LOGIC;  load, sin: in STD_LOGIC;
 d: in STD_LOGIC_VECTOR(N–1 downto 0);
 q: out STD_LOGIC_VECTOR(N–1 downto 0);
 sout: out STD_LOGIC); end;
architecture synth of shiftreg is
 process(clk, reset) begin
 if reset = ‘1’ then q <= (OTHERS => ‘0’);
 elsif rising_edge(clk) then
 if load then q <= d;
 else q <= q(N–2 downto 0) & sin;
 end if;
 end if;
 end process;
 sout <= q(N–1);
end;
```

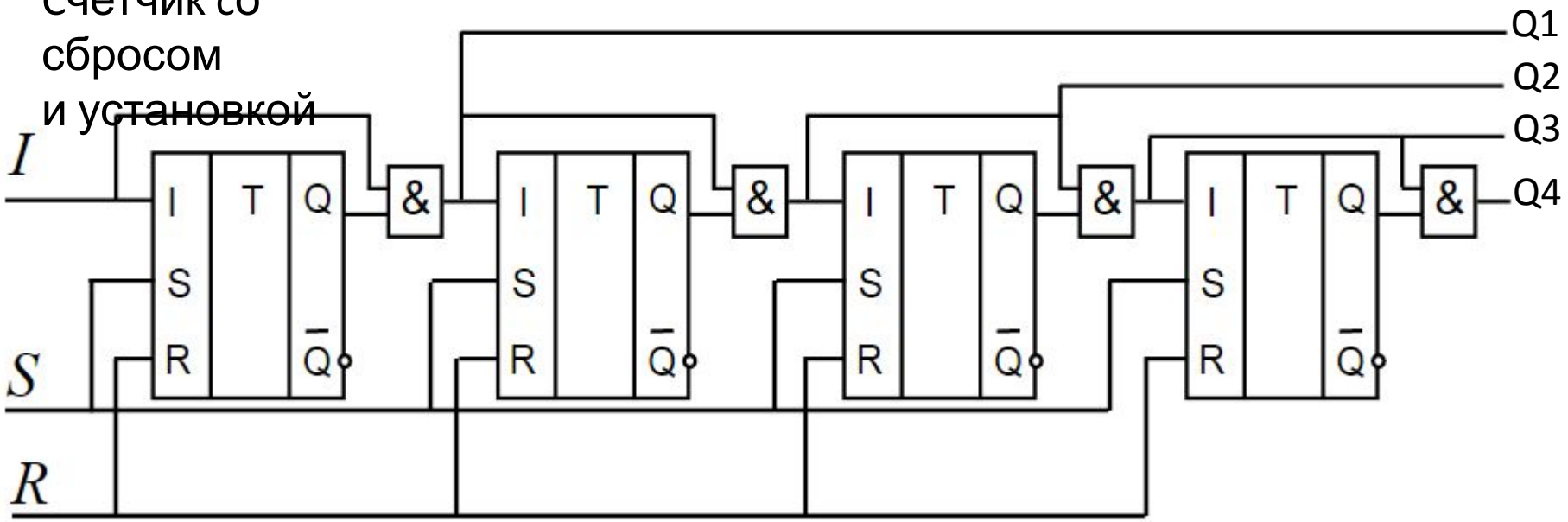
$$Q = ((\overline{Q} \cdot I \cup S) \cdot \overline{R})$$



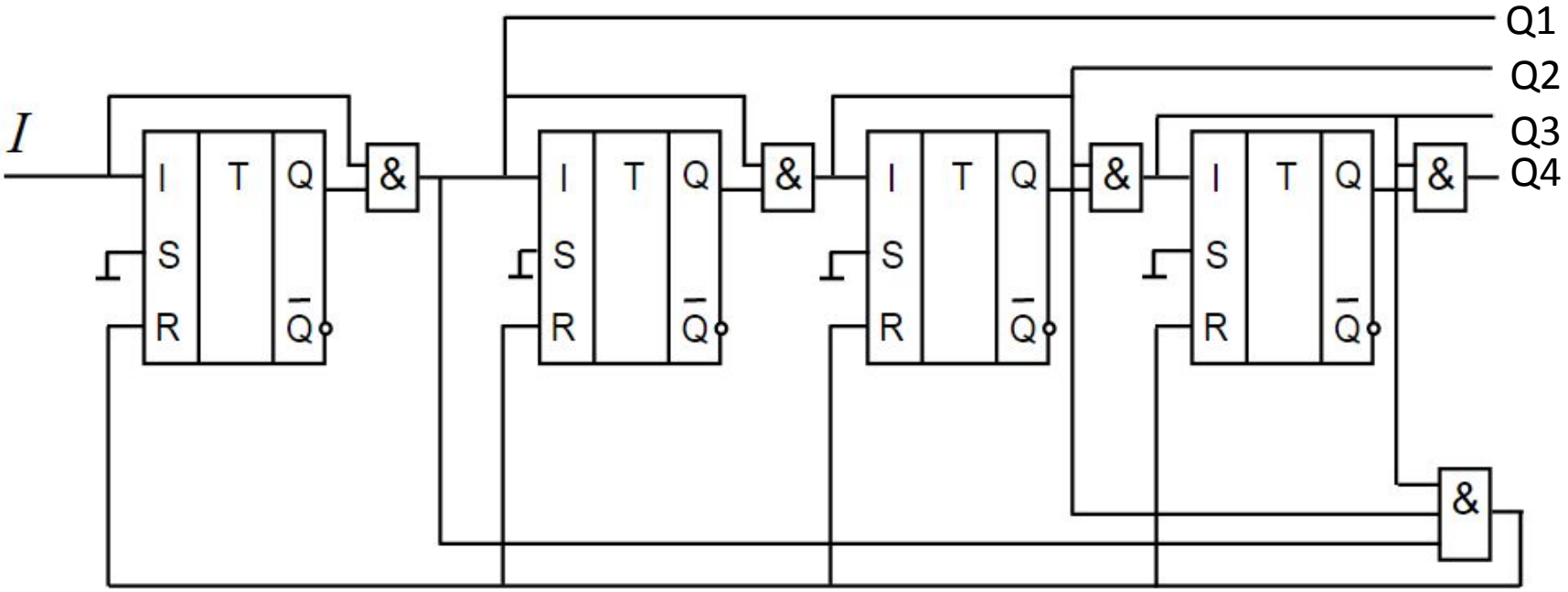
Счетчик



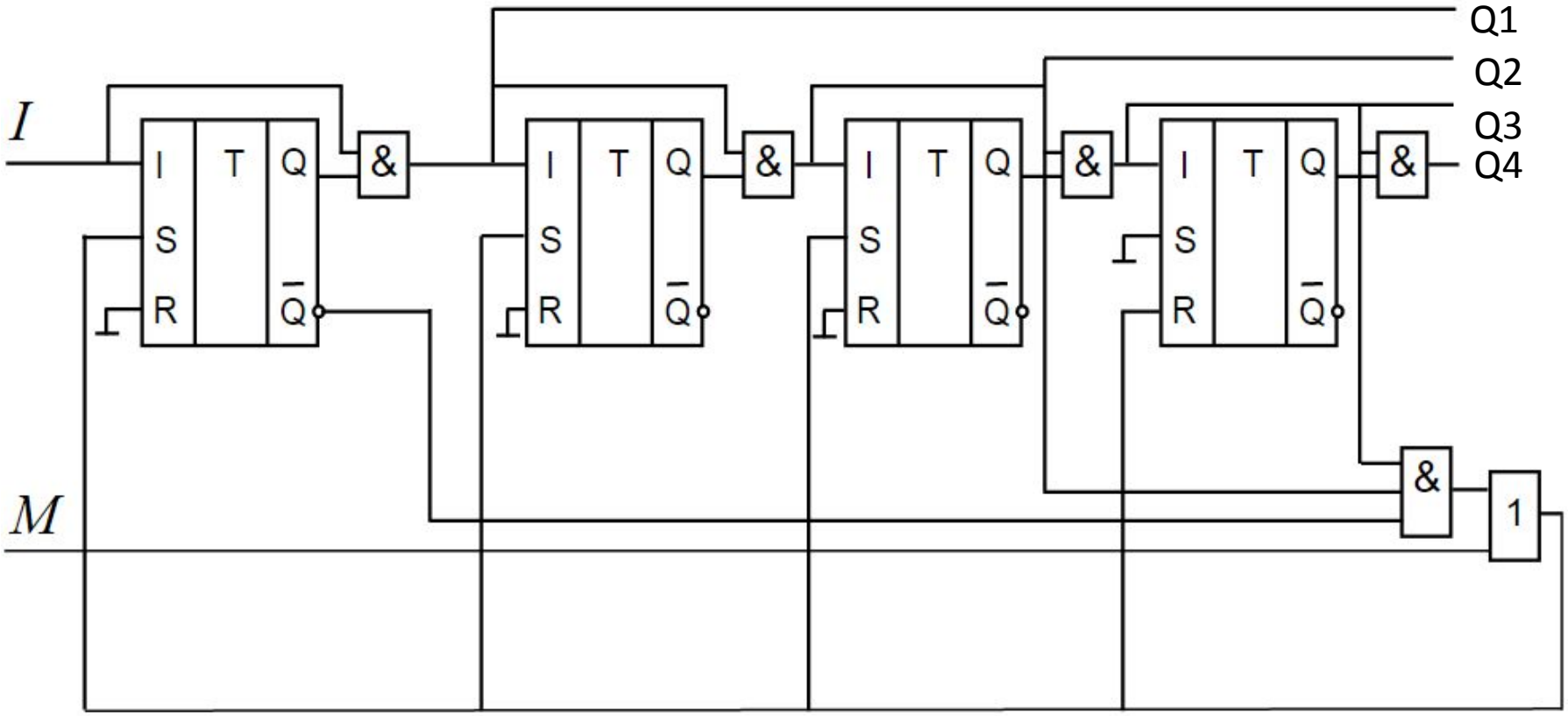
Счетчик со сбросом и установкой



Счетчик с промежуточным сбросом (из какого состояния?)



Счетчик с пропуском состояний и внешней установкой в «0111»

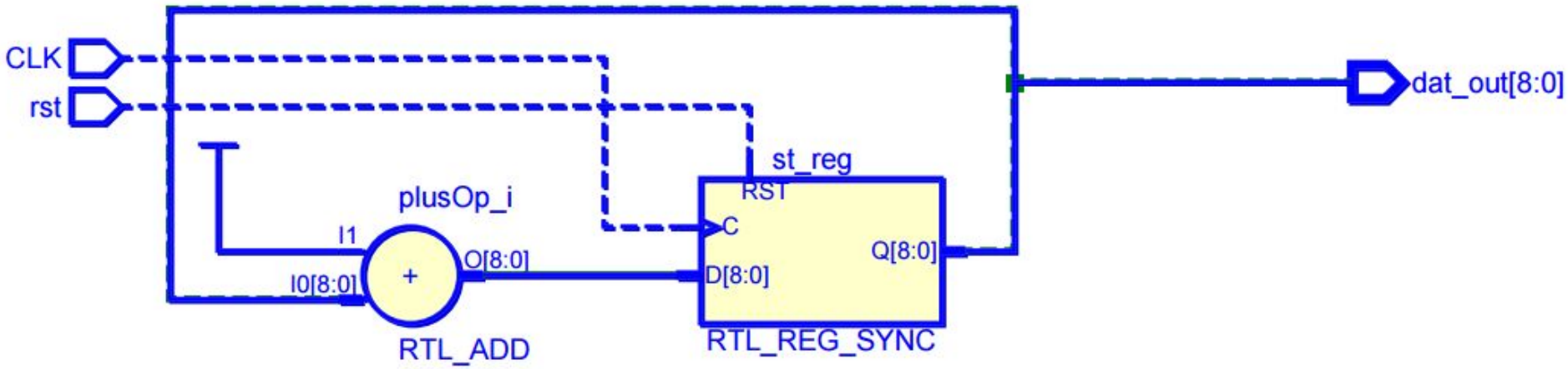


Счетчик: Пример поведенческого описания и результат моделирования

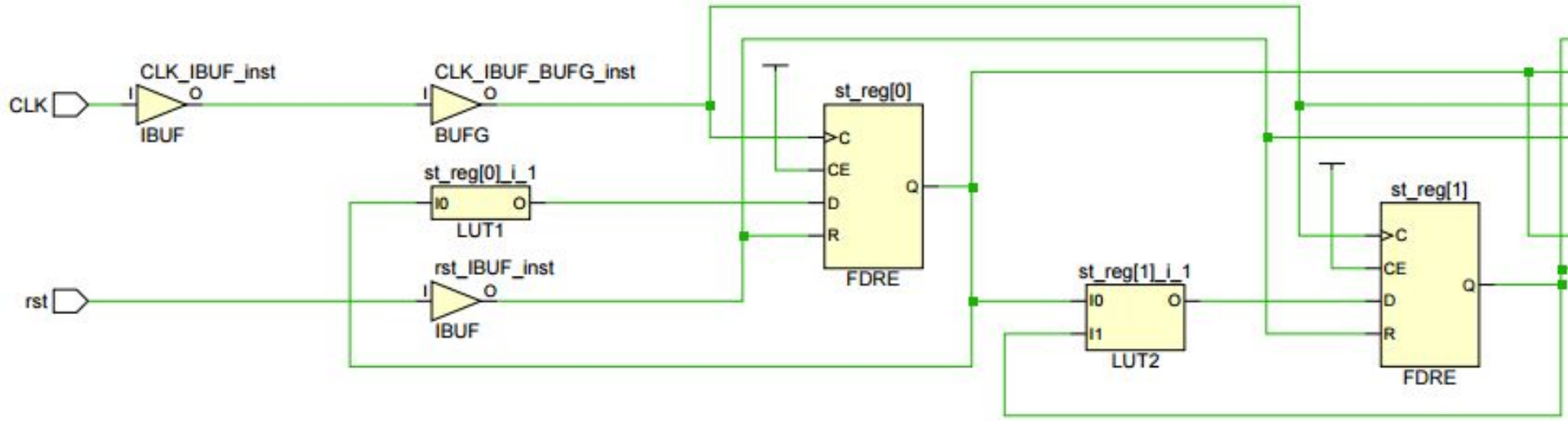
```
-- Простой счетчик  со сбросом
process(CLK)
begin
if CLK 'event and CLK = '1' then
if (rst) = '1' then
    st <= "000000000";
else
st <= st + 1;
end if;
```

Счетчик: Пример поведенческого описания схема-«калька» с него

```
-- Простой счетчик  со сбросом
process(CLK)
begin
if CLK 'event and CLK = '1' then
if (rst) = '1' then
    st <= "000000000";
else
st <= st + 1;
end if;
```

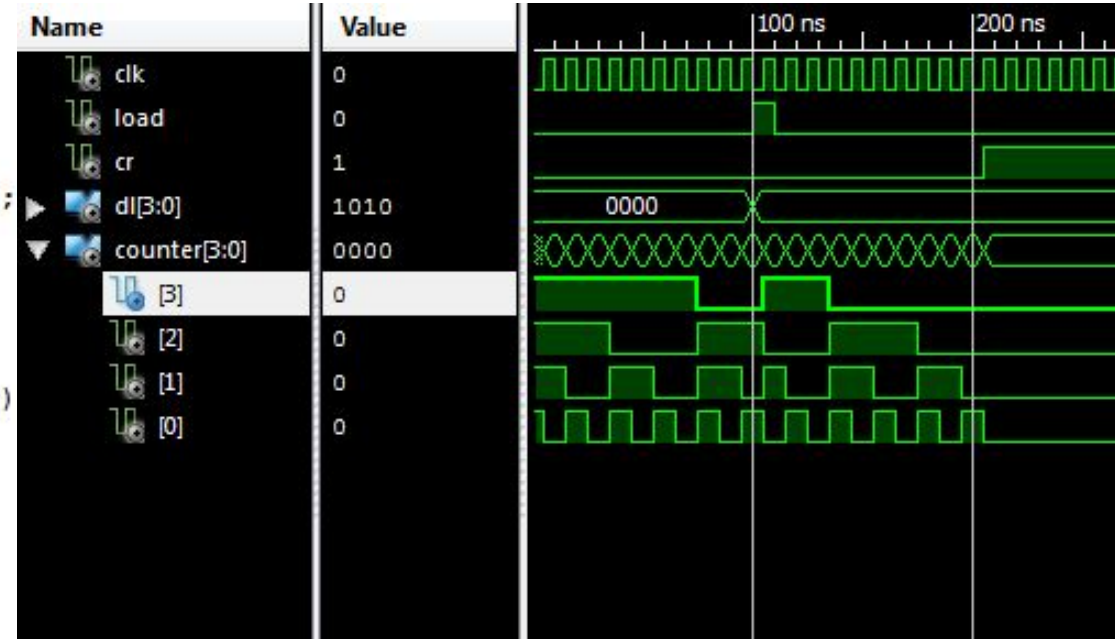


Счетчик: поведенческое описание, переведенное САПР
на элементы ПЛИС (фрагмент)



Реверсивный счетчик: пример поведенческого описания и результат моделирования

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity RevCounter4 is
    Port ( clk : in STD_LOGIC;
           load : in STD_LOGIC;
           dl : in STD_LOGIC_VECTOR (3 downto 0);
           CR : out STD_LOGIC);
end RevCounter4;

architecture Behavioral of RevCounter4 is
    Signal counter : std_logic_vector(3 downto 0)
        := (others => '1');
begin
    Process (clk)
    Begin
        If (rising_edge(clk)) then
            If (load = '1') then
                Counter <= dl;
            Else
                If (counter /= 0) then
                    counter <= counter -1 ;
                End if;
            End if;
        End if;
    End process;
    CR <= '1' when counter = 0 else '0';

end Behavioral;
```



Реверсивный счетчик считает от исходного
набора например 1111 до 0000.

Благодарю за внимание!