Circuitos Lógicos e Organização de Computadores

Capítulo 7 – Flip-Flops e Circuitos Seqüenciais

Ricardo Pannain

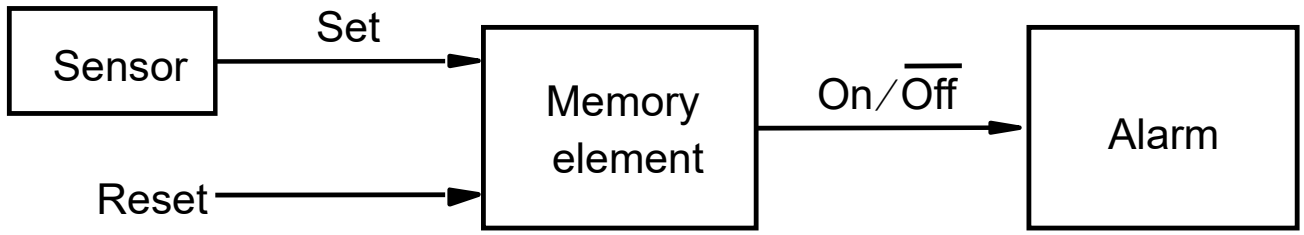
pannain@unicamp.br

Circuito Seqüenciais

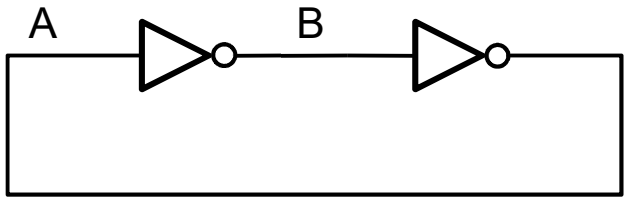
Circuito combinacional → saídas dependem apenas das entradas

Circuito sequencial → saídas dependem das entradas e do comportamento anterior do circuito

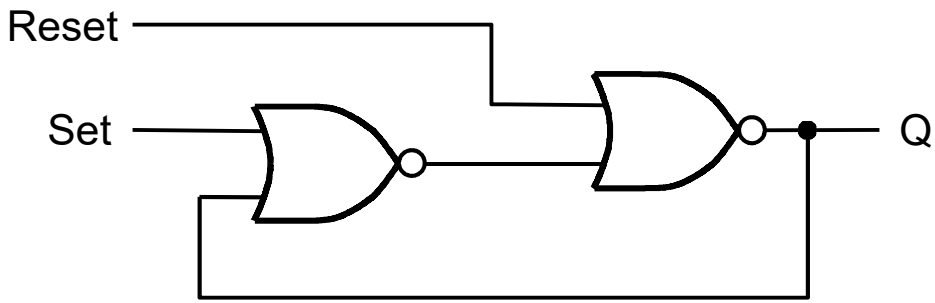
Exemplo - Controle de Sistema de Alarme



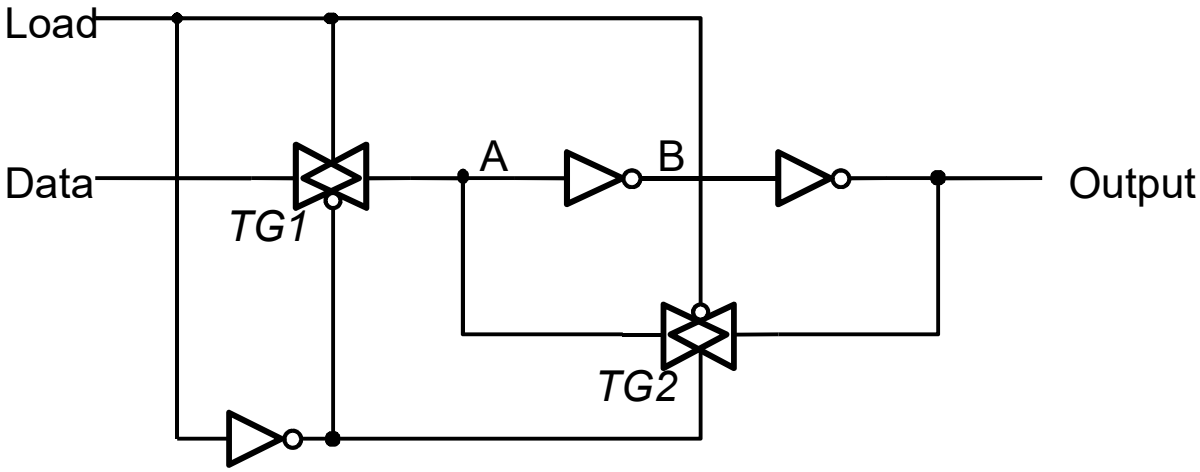
Elemento de memória simples



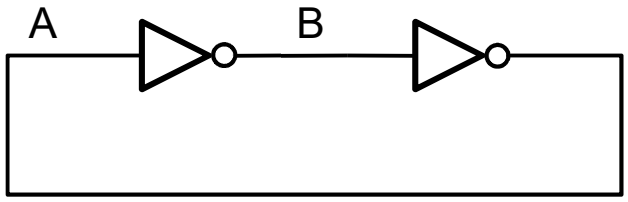
Elemento de memória com portas NOR



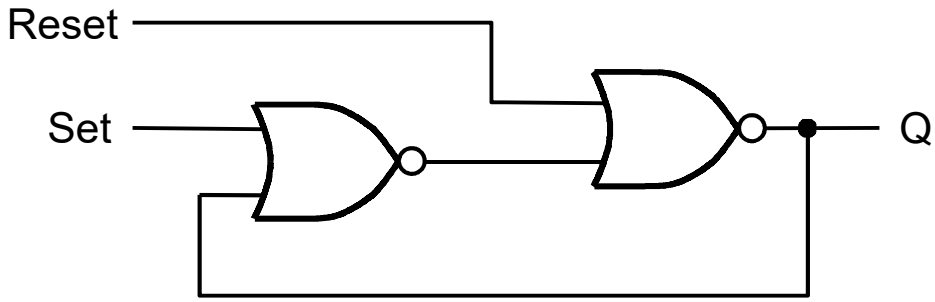
Elemento de memória controlado



Elemento de memória simples

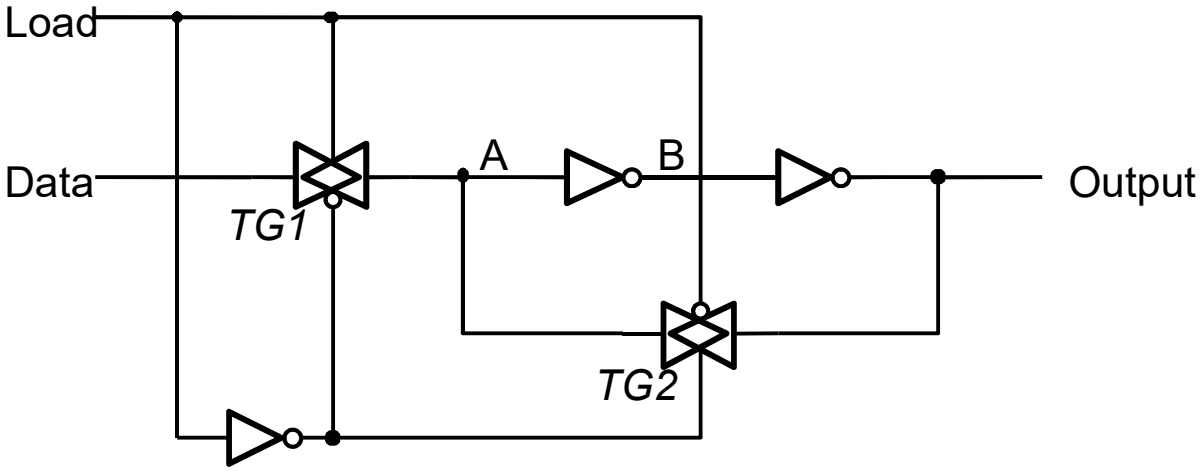


Elemento de memória com portas NOR

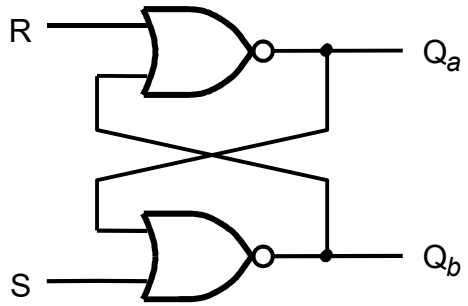


SET	RESET	Q
0	0	Q
0	1	0
1	0	1
1	1	0

Elemento de memória controlado



Latch construído com portas NOR

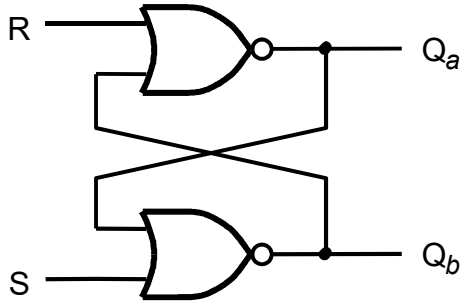


(a) Circuito

Tabela Verdade ??????

Diagrama de tempo ?????

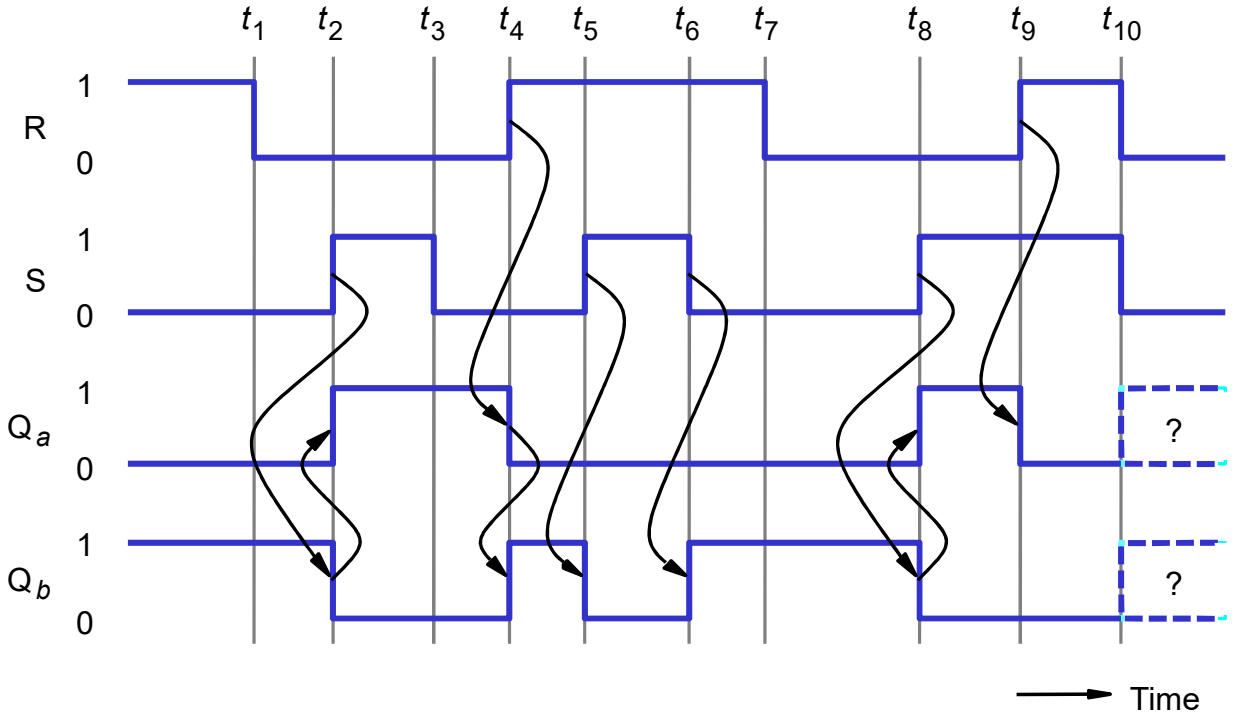
Latch construído com portas NOR



(a) Circuito

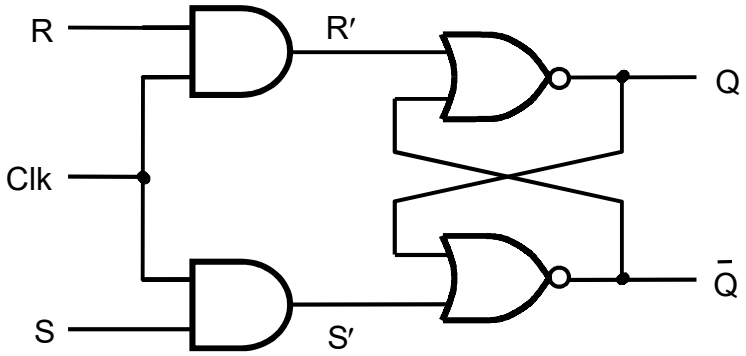
S	R	Q_a	Q_b	
0	0	0/1	1/0	(sem alteração)
0	1	0	1	
1	0	1	0	
1	1	0	0	

(b) Tabela Verdade



(c) Diagrama de Tempo

Latch SR com clock (gated)

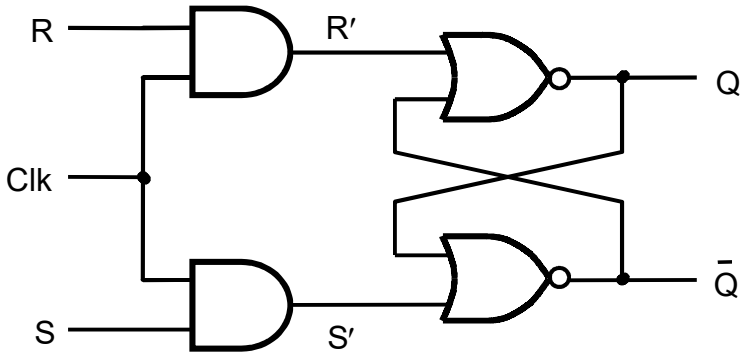


(a) Circuito

Tabela Verdade ??????

Diagrama de tempo ?????

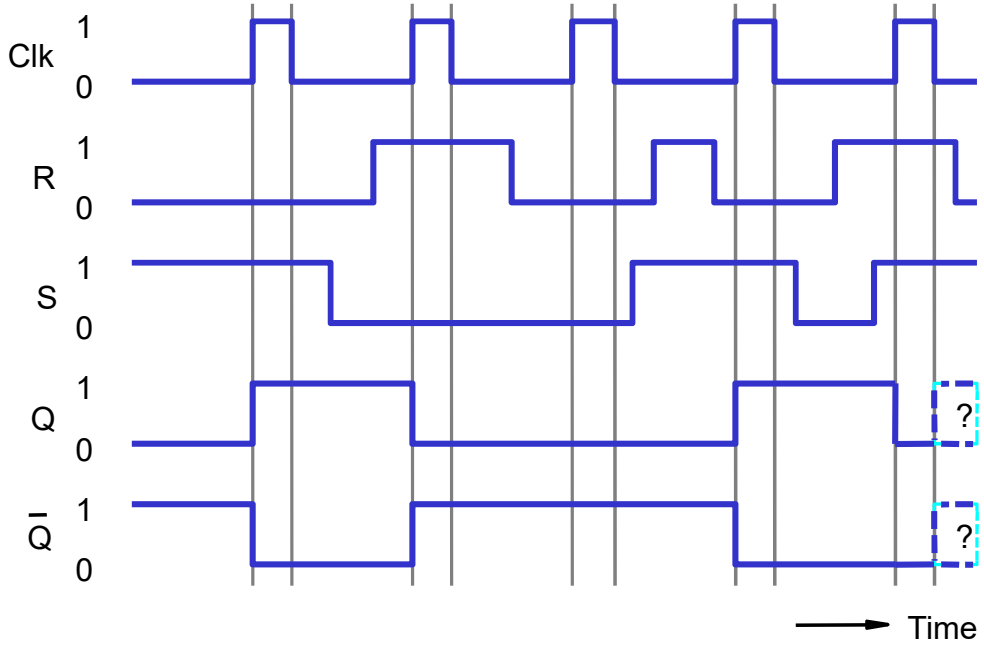
Latch SR com clock (gated)



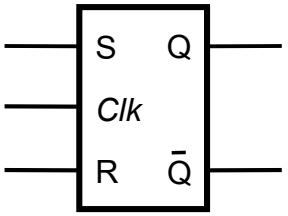
(a) Circuito

Clk	S	R	Q(*t* + 1)
0	x	x	Q(*t*) (sem alteração)
1	0	0	Q(*t*) (sem alteração)
1	0	1	0
1	1	0	1
1	1	1	x

(b) Tabela Verdade

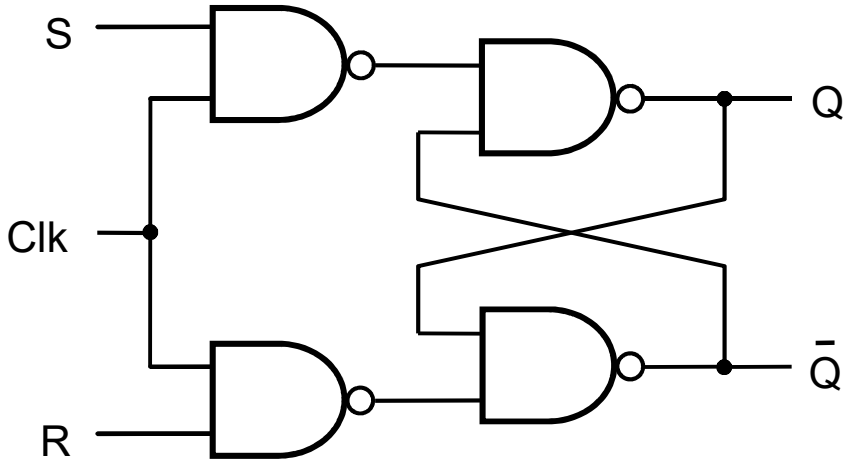


(c) Timing diagram

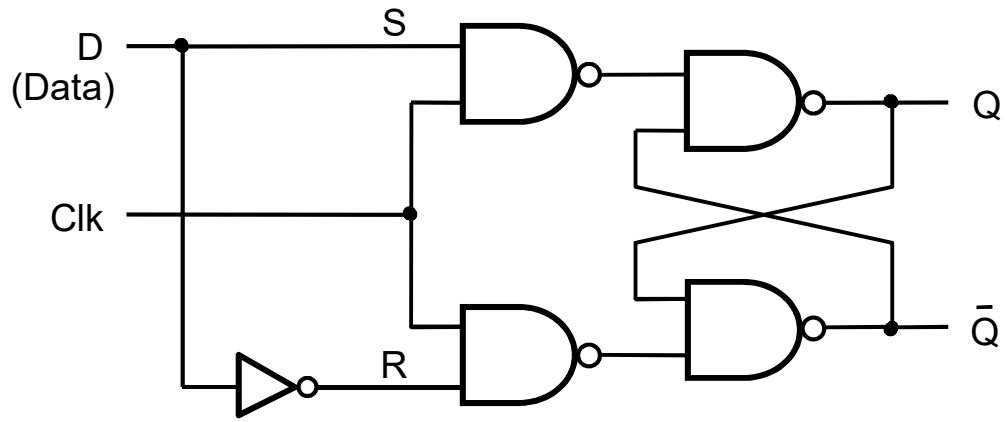


(d) Símbolo Gráfico

Latch SR Gated SR com portas NAND



Latch D Gated

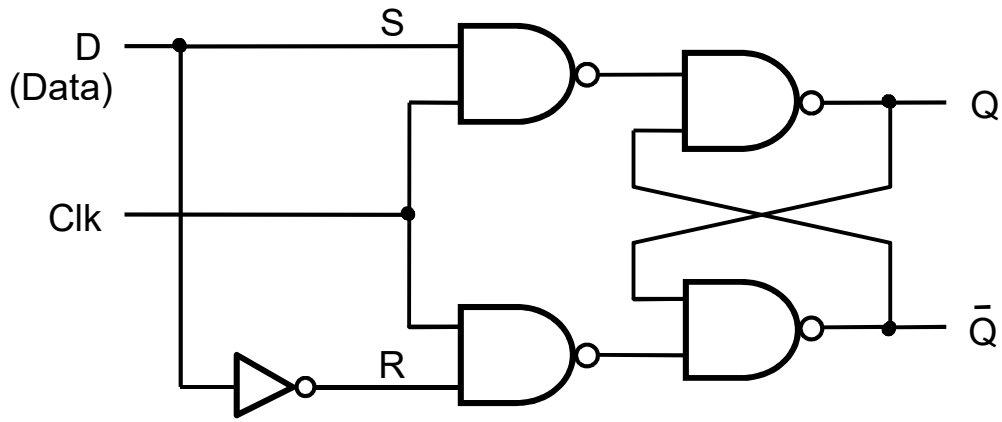


(a) Circuito

Tabela Verdade ??????

Diagrama de tempo ?????

Latch D Gated



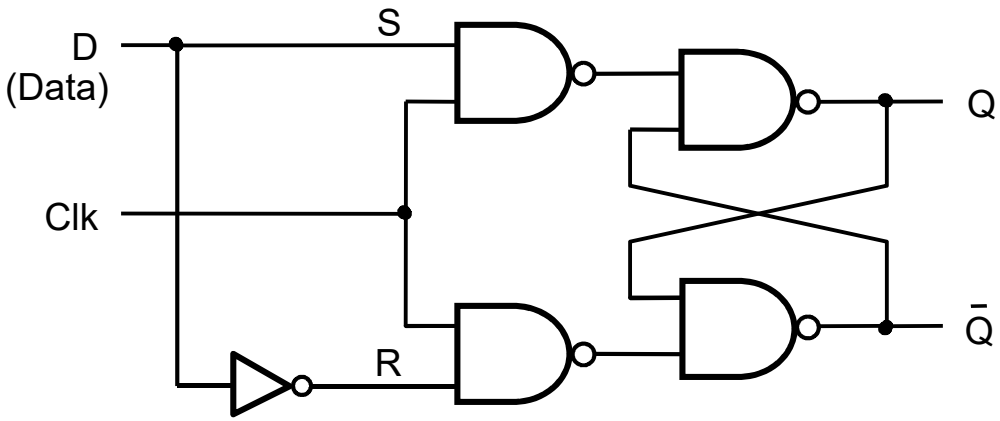
(a) Circuito

CLK	D	Qt+1
0	x	Qt
1	0	0
1	1	1

Tabela Verdade ??????

Diagrama de tempo ?????

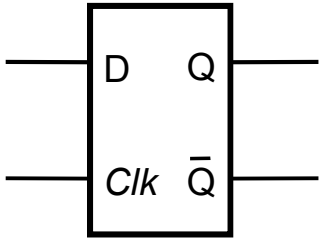
Latch D Gated



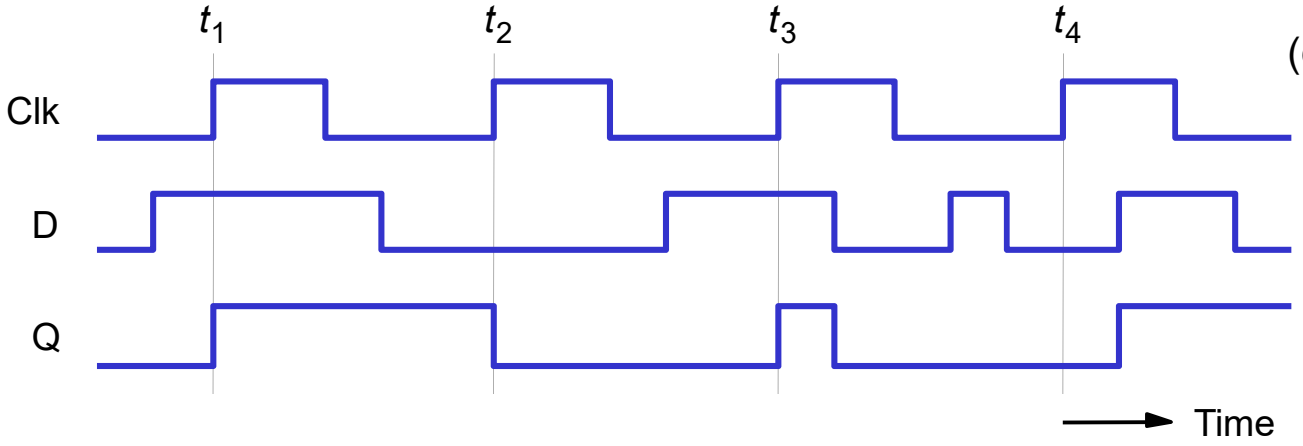
(a) Circuito

Clk	D	Q(*t* + 1)
0	x	Q(*t*)
1	0	0
1	1	1

(b) Tabela Verdade

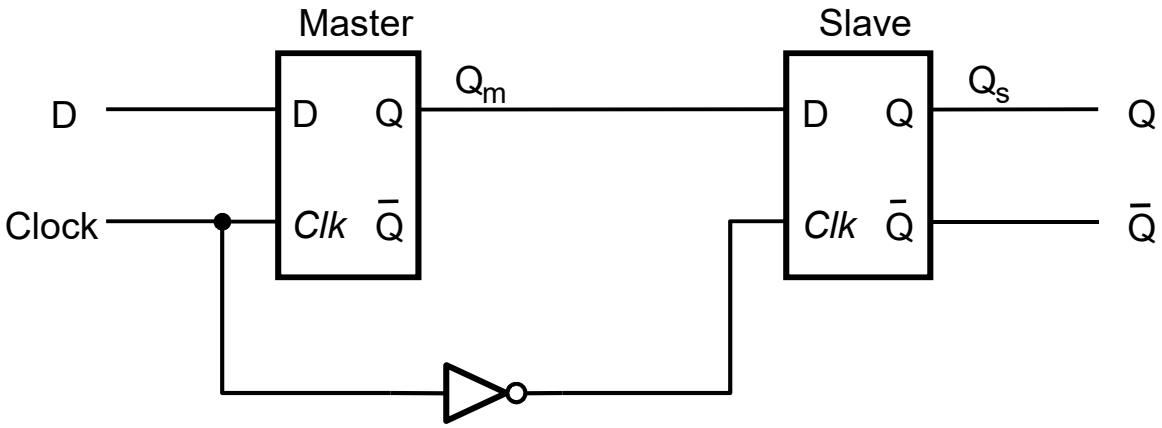


(c) Símbolo Gráfico



(d) Diagrama de Tempo

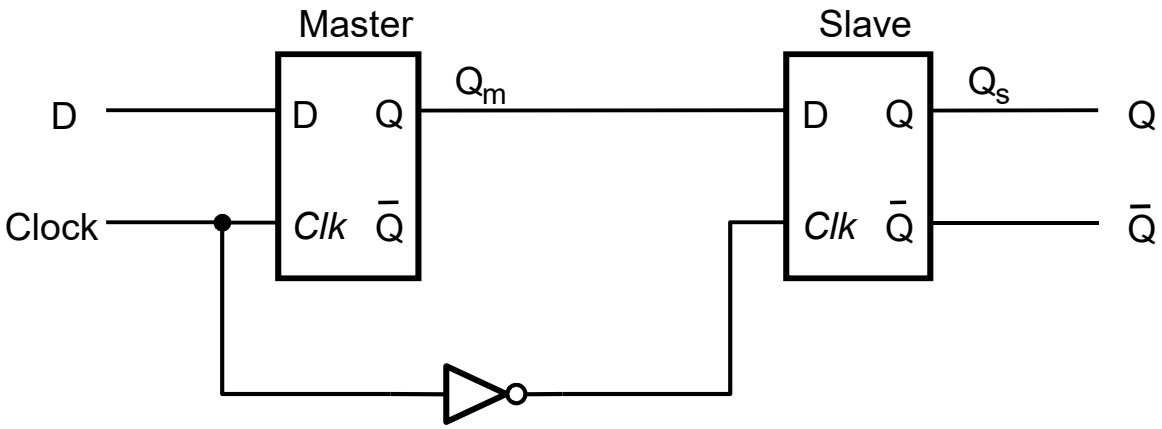
Flip-Flop D Master-slave



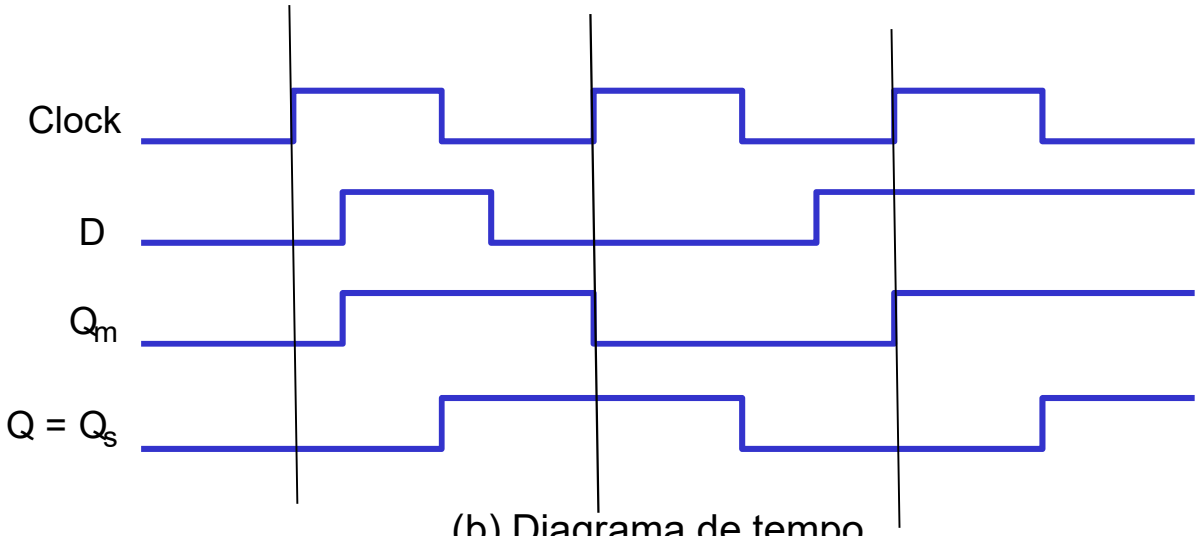
(a) Circuito

Diagrama de tempo e o comportamento?????

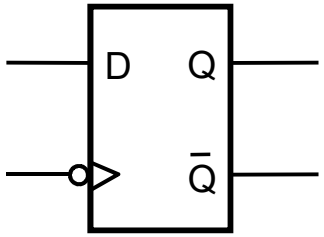
Flip-Flop D Master-slave



(a) Circuito



(b) Diagrama de tempo

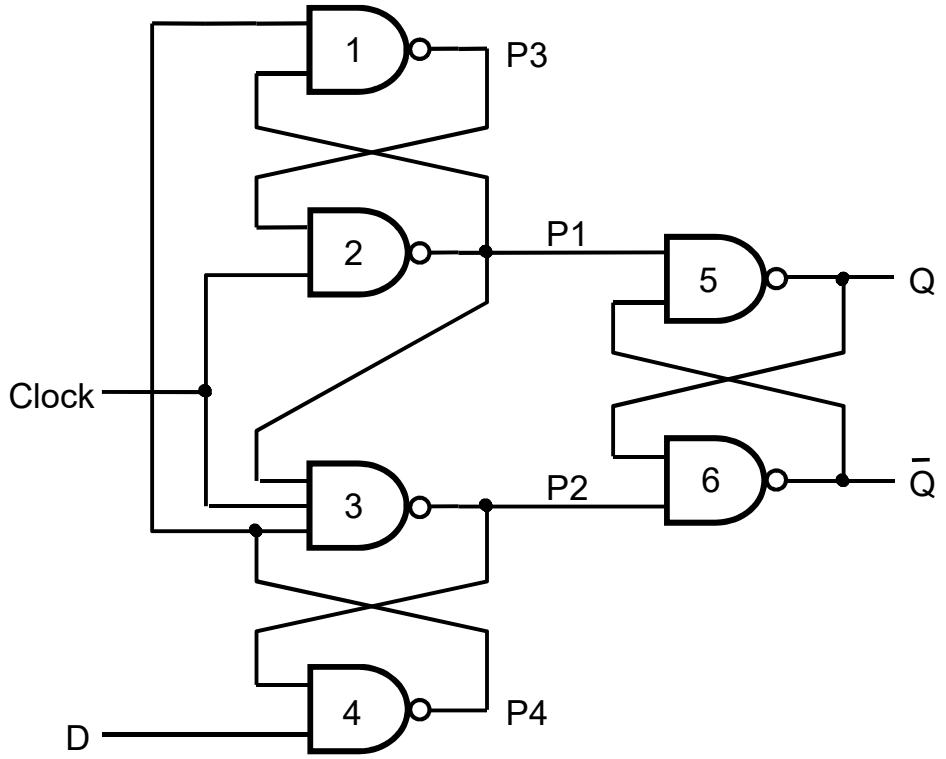


(c) Símbolo Gráfico

CLK = 1 master armazena
e slave não muda
CLK = 0 master não muda
e slave armazena

Sensível à borda de descida

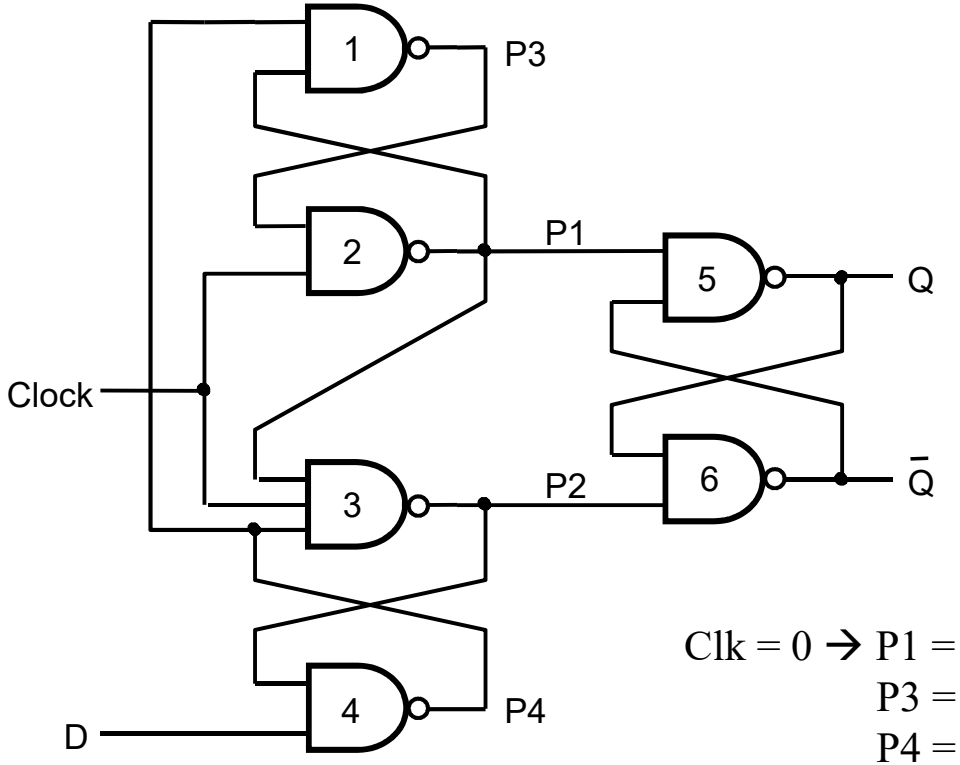
Flip-Flop D sensível à borda de subida



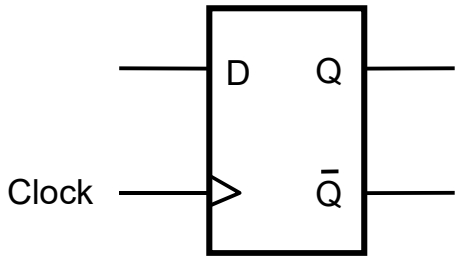
(a) Circuito

Funcionamento?????

Flip-Flop D sensível à borda de subida



(a) Circuito



(b) Símbolo Gráfico

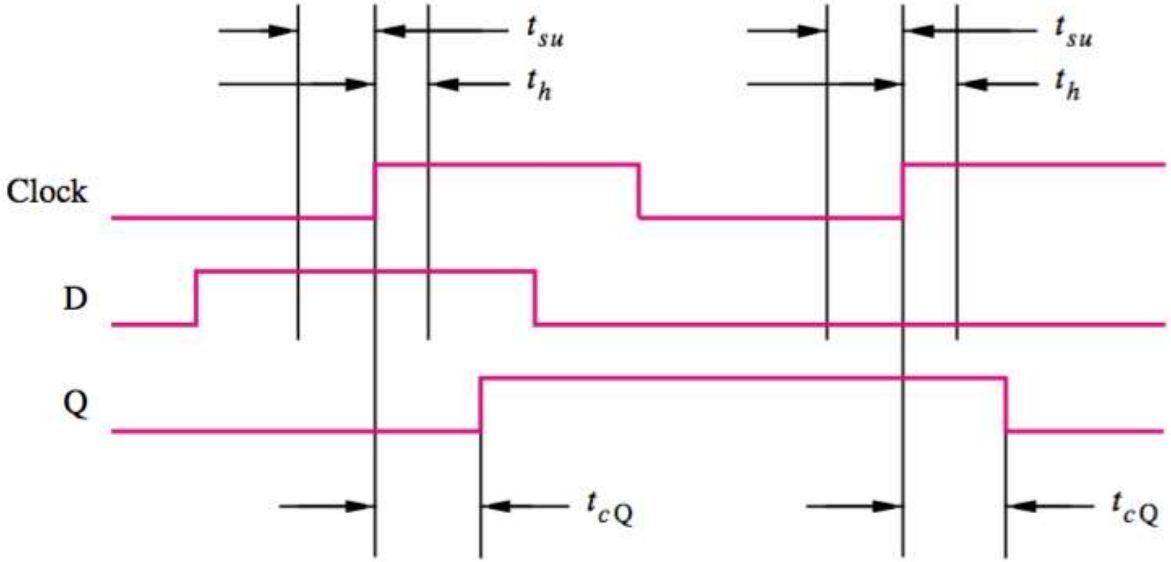
Clk = 0 → P1 = P2 = 1
P3 = D
P4 = D

Clk =1 → P3 e P4 são transmitidos através de 2 e 3

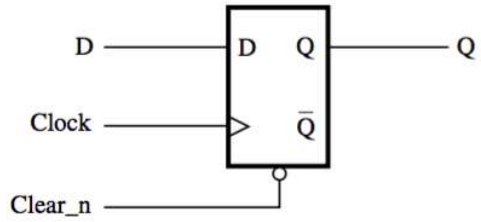
→ P1 = $\overline{D}$ e P2 = D → Q = D e $\overline{Q}$ = $\overline{D}$

Obs - P4 e P3 DEVEM ESTAR ESTÁVEIS QUANDO CLK MUDA PARA 1

Temporização



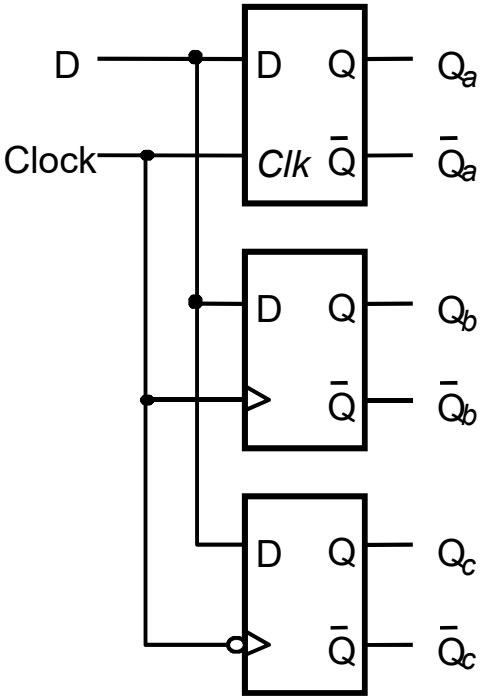
(b) Timing diagram



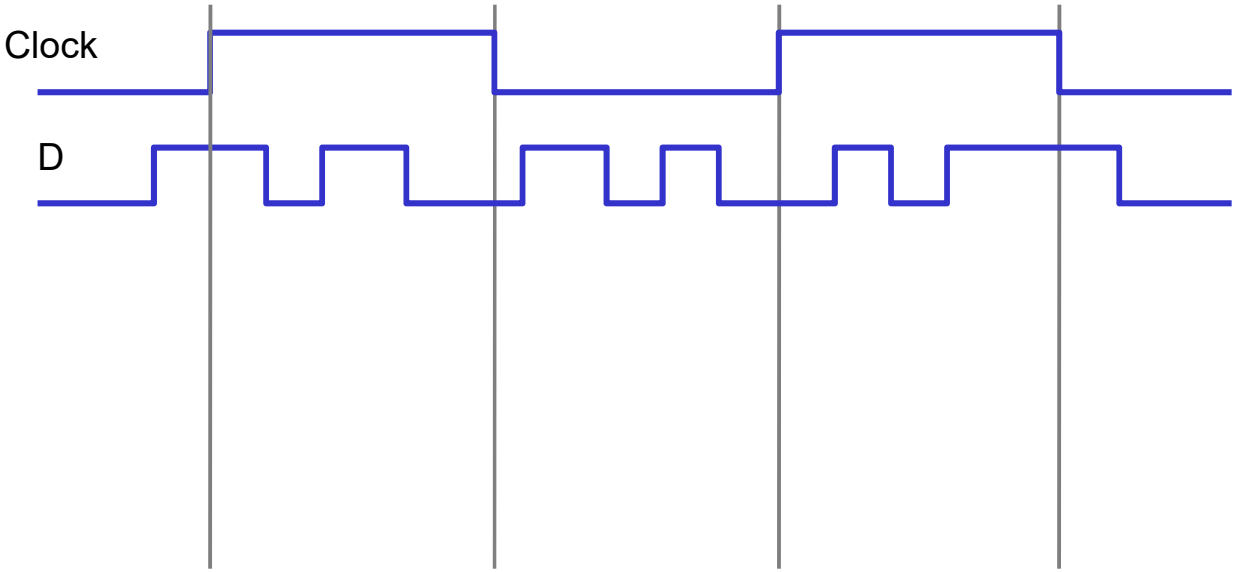
(a) D flip-flop with asynchronous clear

- Setup time (T_{su}) → tempo mínimo que D deve estar estável antes da subida do clock
- Hold time (T_h)→ tempo mínimo que D deve ser mantido estável após a subida do clock
- T_{cQ}: tempo até a saída Q mudar depois de uma borda de subida

Comparação de Flip-Flops D sensíveis a nivel e sensíveis a borda

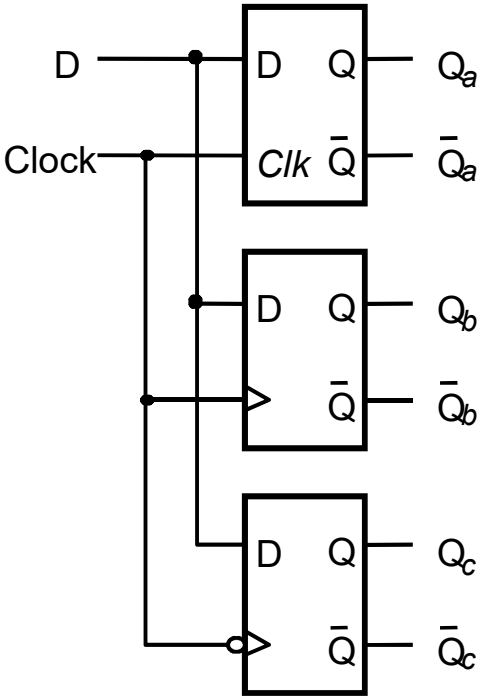


(a) Circuito

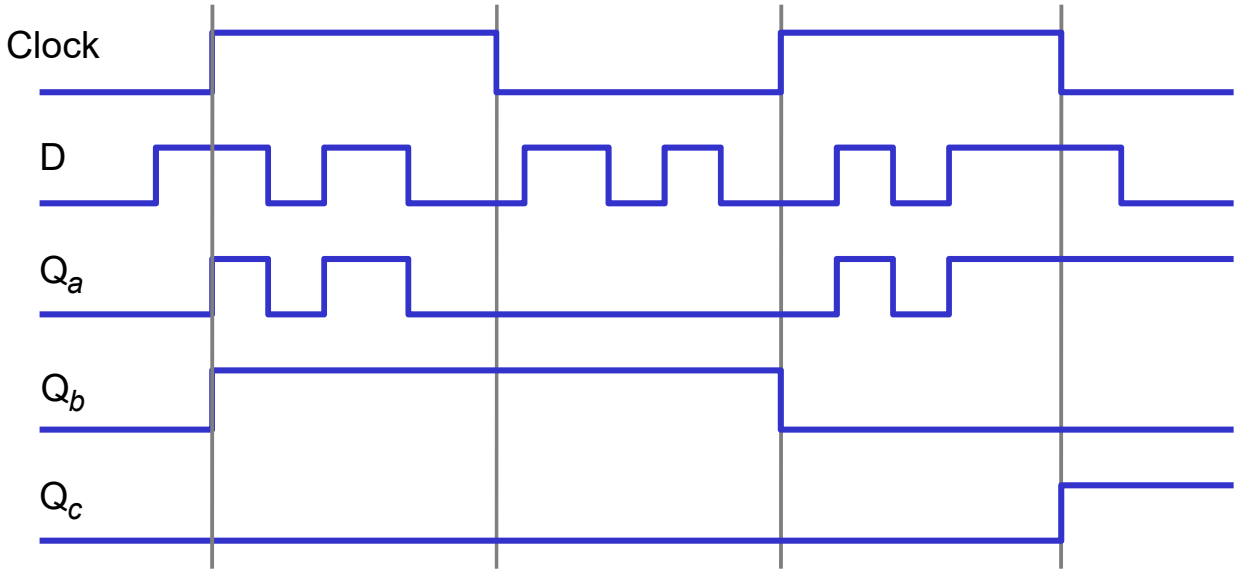


(b) Diagrama de Tempo

Comparação de Flip-Flops D sensíveis a nivel e sensíveis a borda

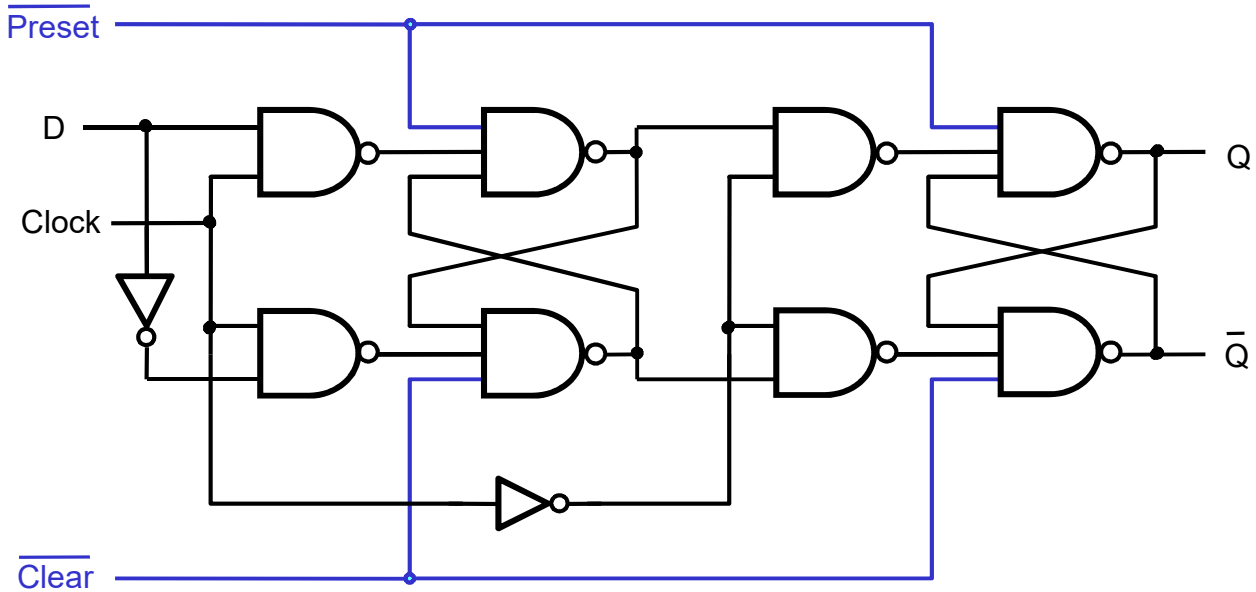


(a) Circuito

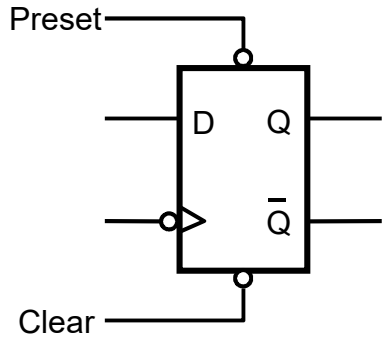


(b) Diagrama de Tempo

Flip-Flop Master-slave tipo D com *Clear* e *Preset*

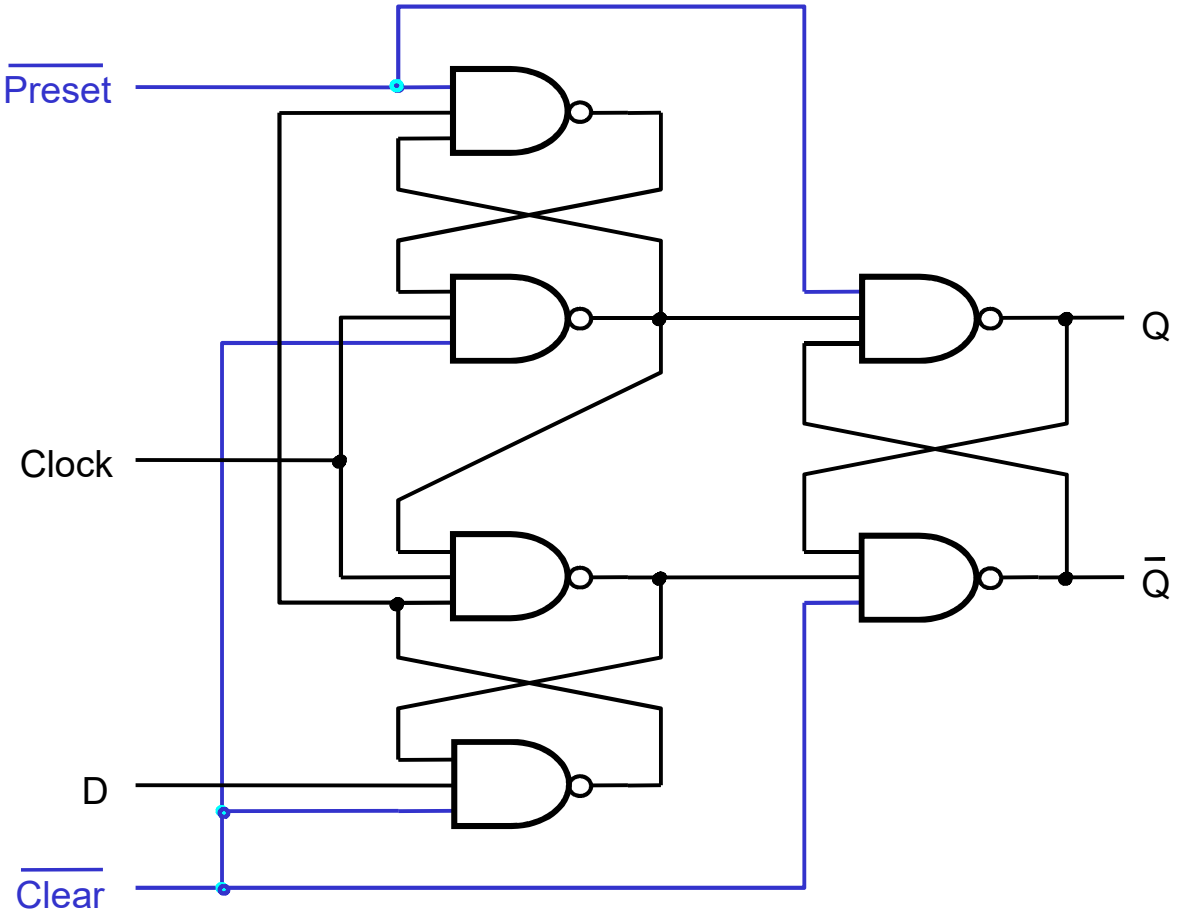


(a) Circuito

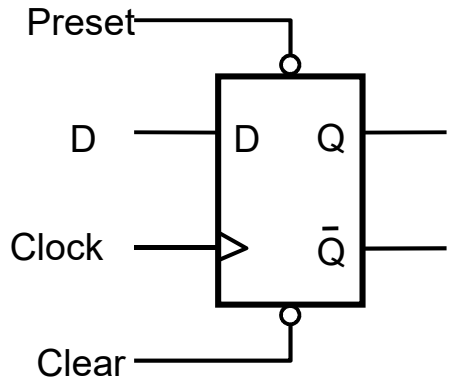


(b) Símbolo Gráfico

Flip-Flop D sensível à borda de subida com *Clear* e *Preset*

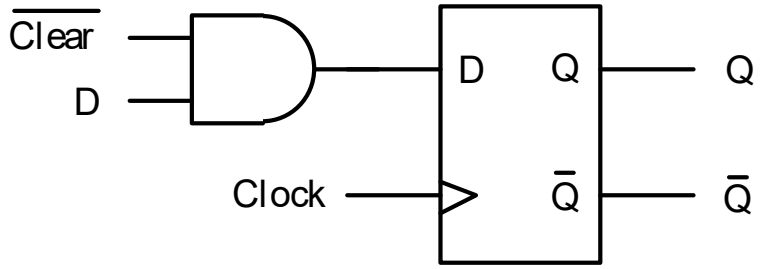


(a) Circuito

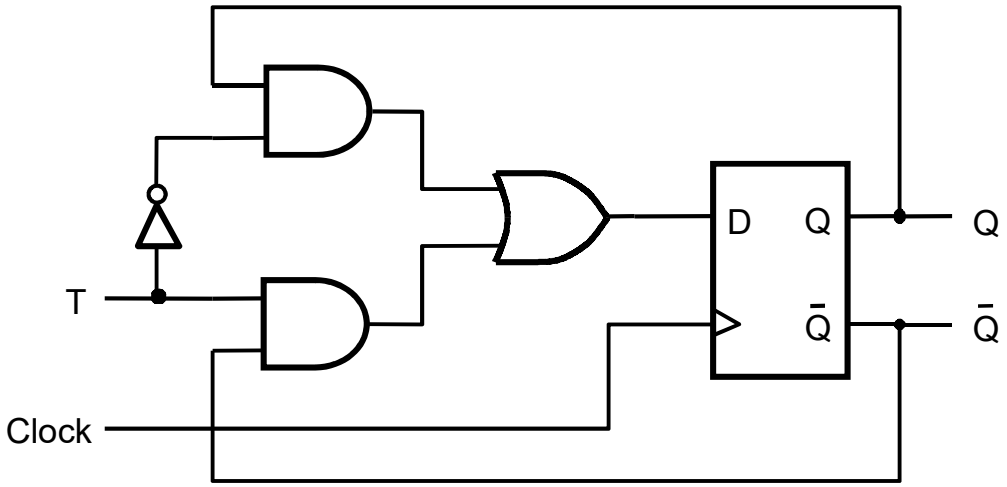


(b) Símbolo Gráfico

Flip-Flop D com reset síncrono



Flip-Flop tipo T (toogle)

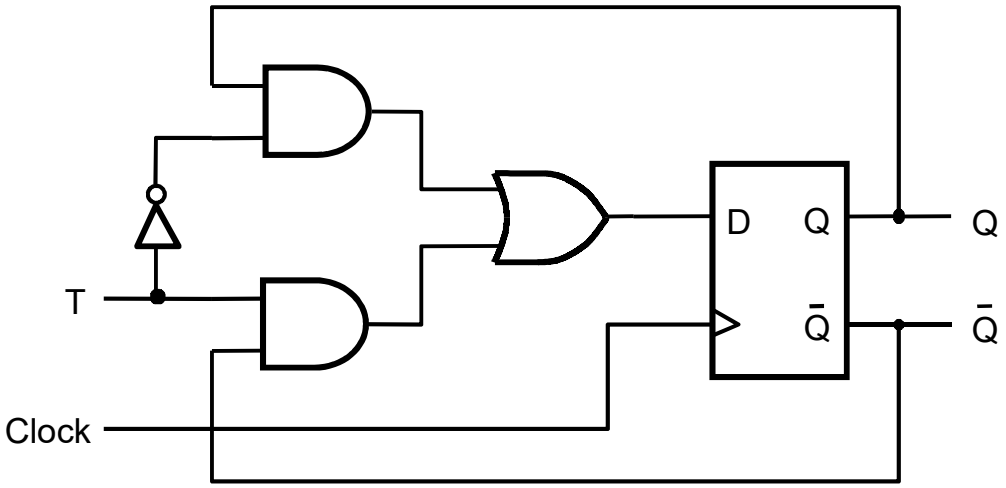


(a) Circuito

Tabela Verdade ??????

Diagrama de tempo ?????

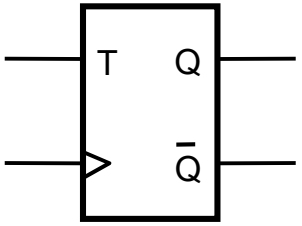
Flip-Flop tipo T (toogle)



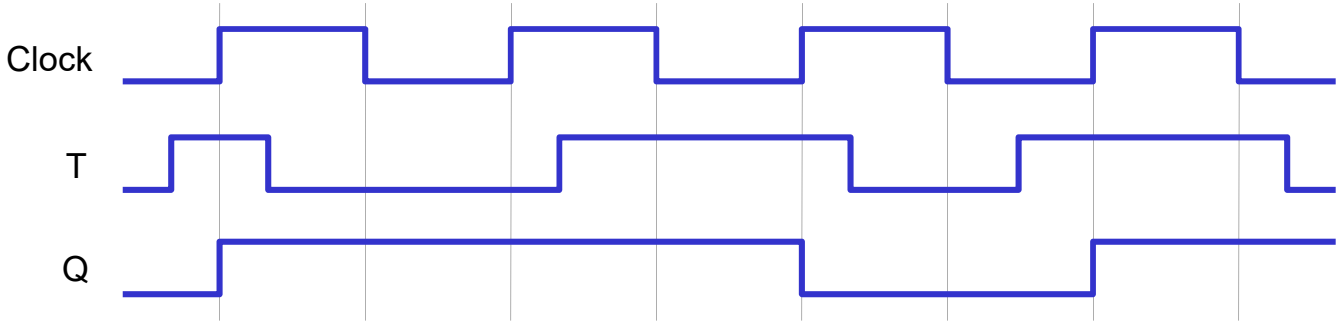
(a) Circuito

T	Q(*t* + 1)
0	Q(*t*)
1	$\bar{Q}(t)$

(b) Tabela Verdade

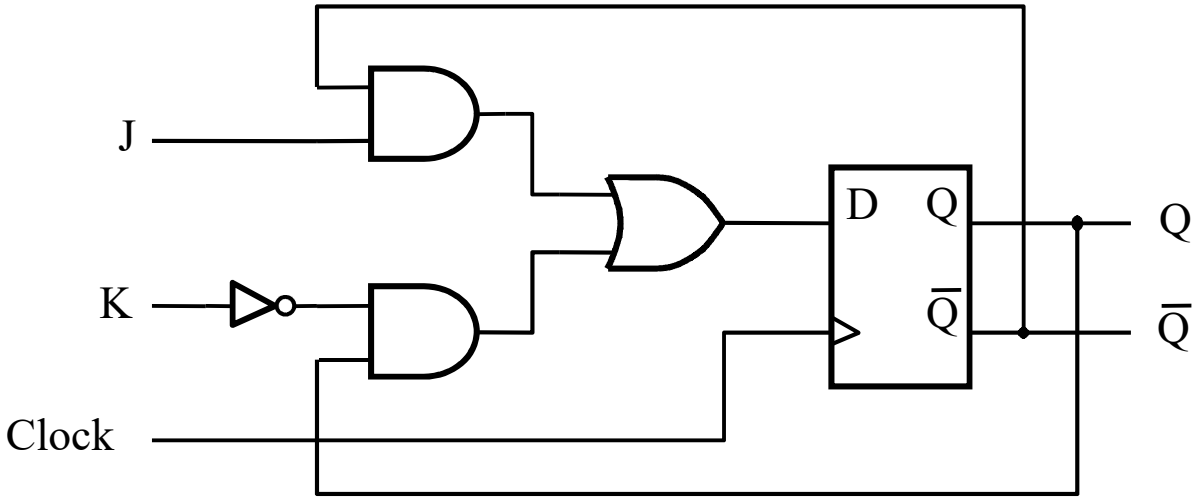


(c) Símbolo Gráfico



(d) Diagrama de tempo

Flip-Flop JK

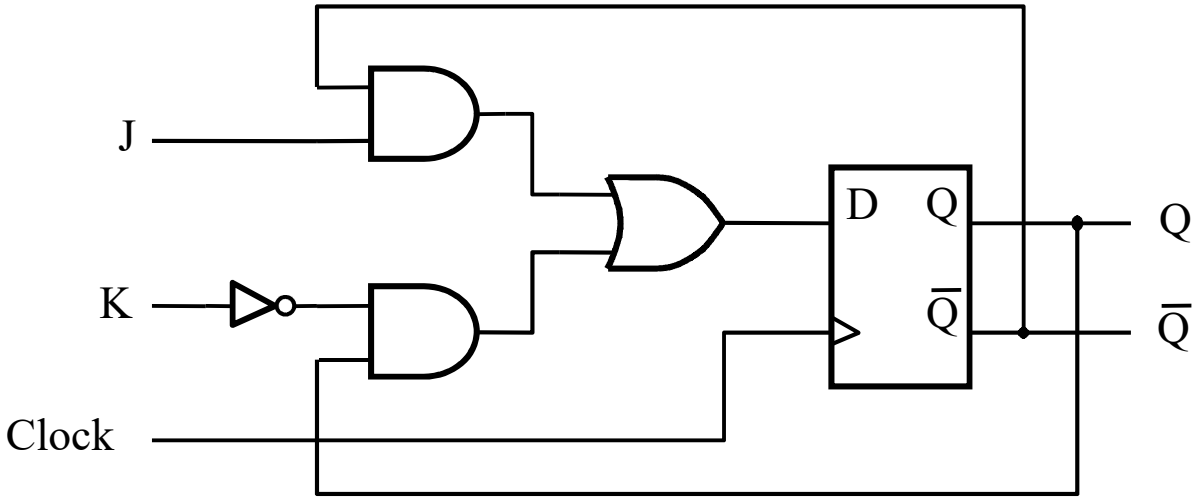


(a) Circuito

Tabela Verdade ??????

Diagrama de tempo ?????

Flip-Flop JK



(a) Circuito

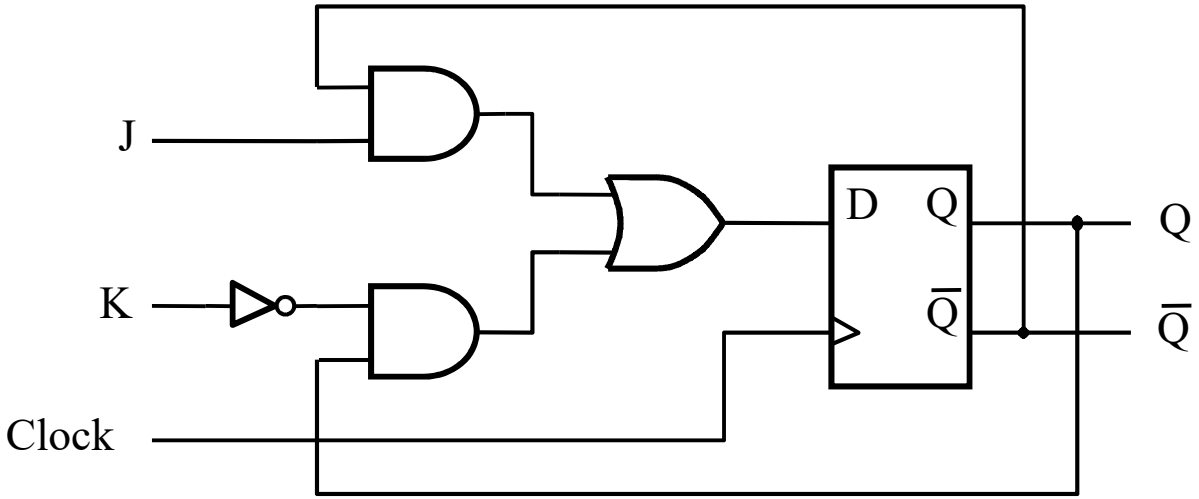
NA SUBIDA DO CLOCK

J	K	Qt+1
0	0	Qt
0	1	0
1	0	1
1	1	Q't

Tabela Verdade ??????

Diagrama de tempo ?????

Flip-Flop JK



(a) Circuito

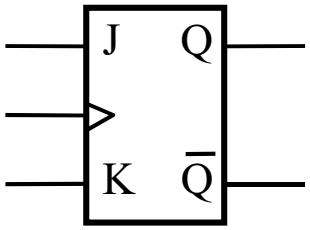
$D = J\,\overline{Q} + \overline{K}\,Q$

J e K → entradas
De controle

FFs SR e T juntos

J	K	Q (t+1)
0	0	Q (t)
0	1	0
1	0	1
1	1	$\overline{Q}$ (t)

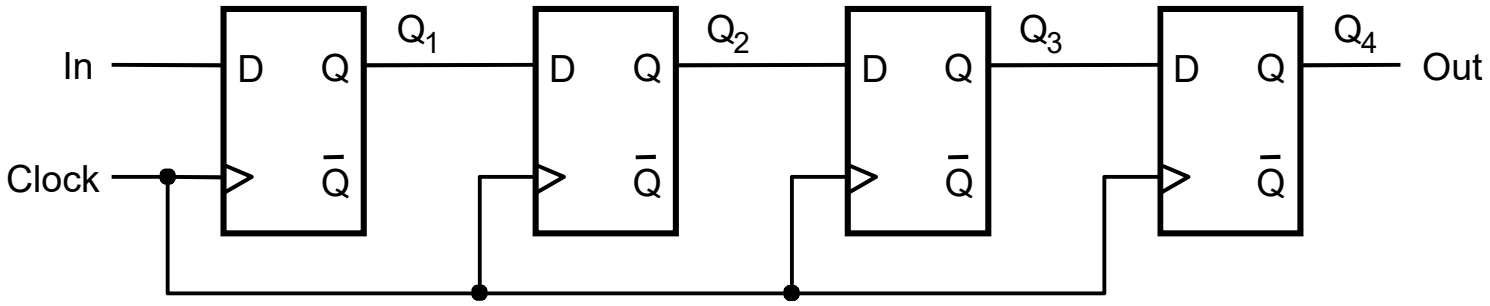
(b) Tabela Verdade



(c) Símbolo Gráfico

Registrador de deslocamento com entrada e saída serial

Registrador é um conjunto de n Flip-Flops



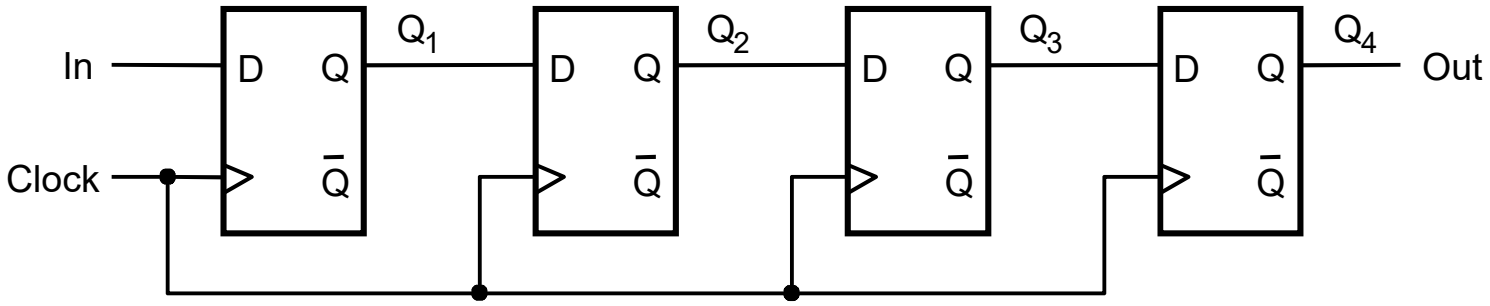
(a) Circuito

Funcionamento tempo ?????

In = 1011

Registrador de deslocamento com entrada e saída serial

Registrador é um conjunto de n Flip-Flops



(a) Circuito

Funcionamento tempo ?????

1011

t0 in = 1 →q1 = 1

t1 in = 0 → q1 = 0 q2 = 1

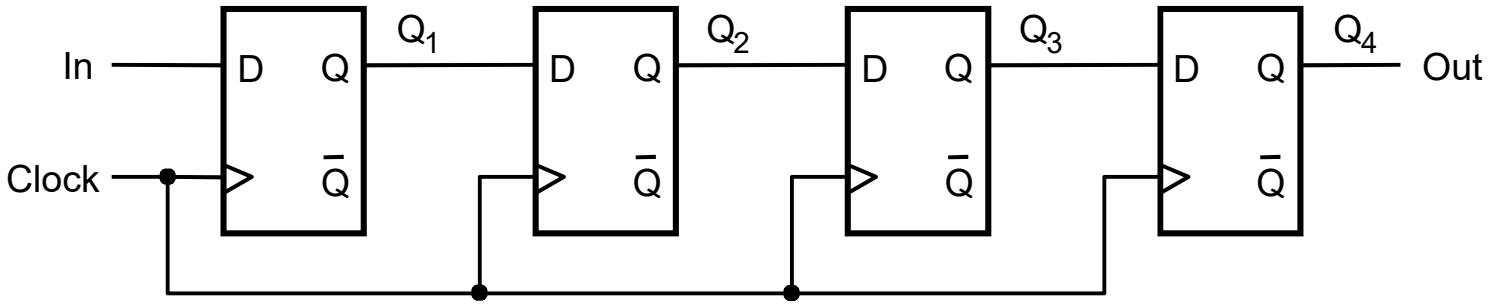
t2 in = 1 → q1 =1 q2 = 0 e q3 = 1

t3 in = 1 → q1 = 1 q2 = 1 q3 = 0 q4 = 1

←

Registrador de deslocamento com entrada e saída serial

Registrador é um conjunto de n Flip-Flops

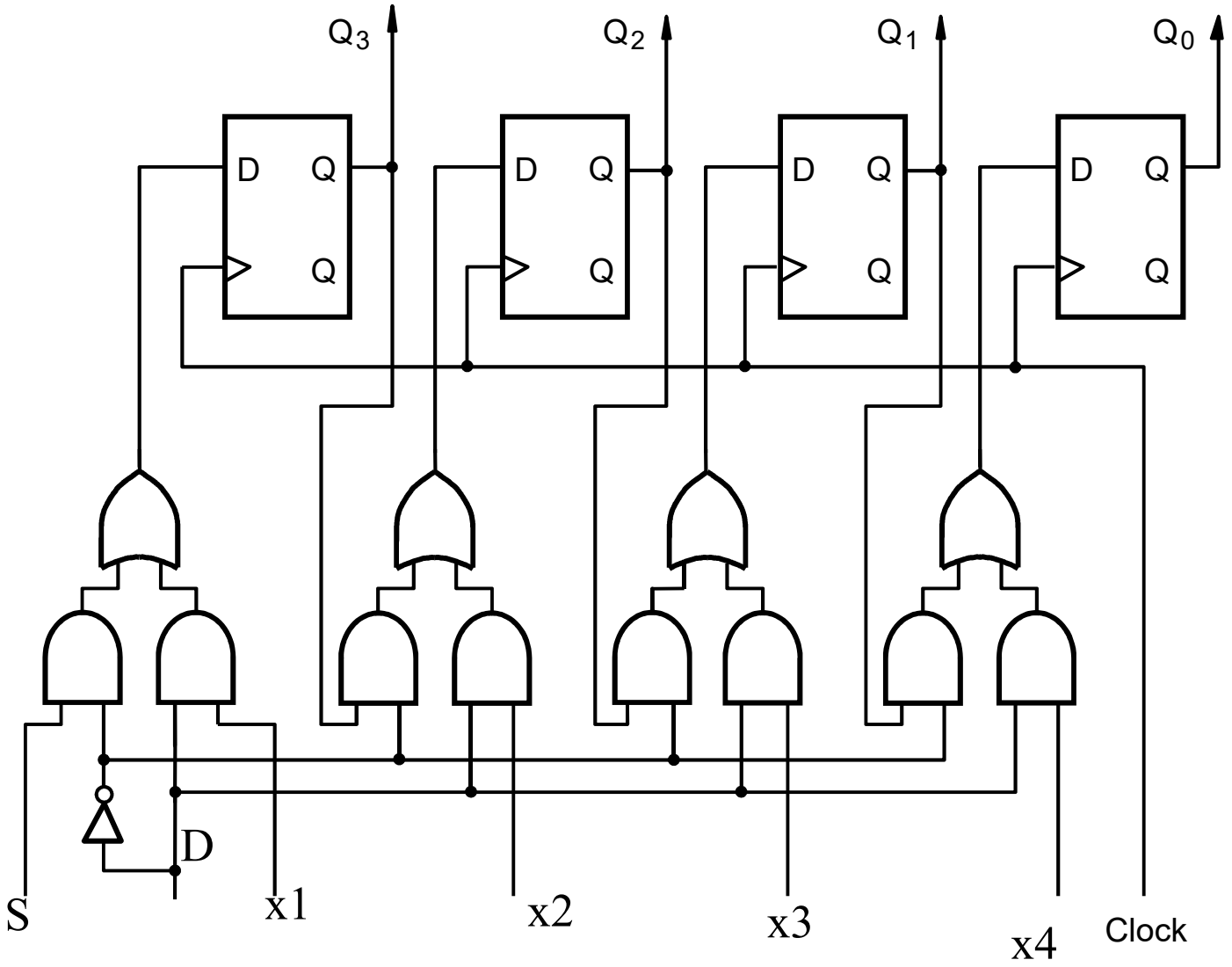


(a) Circuito

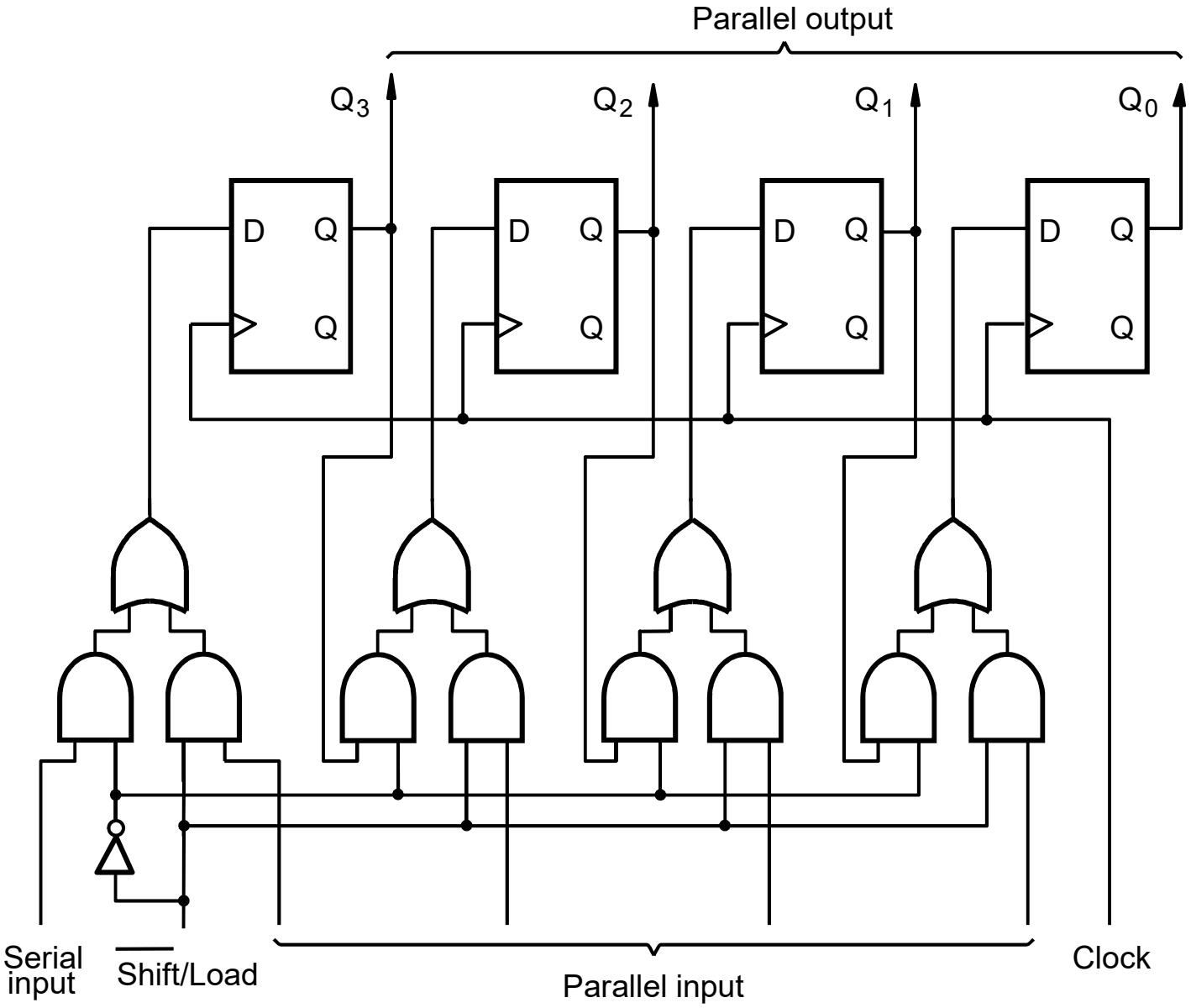
	In	Q_1	Q_2	Q_3	Q_4 = Out
t_0	1	0	0	0	0
t_1	0	1	0	0	0
t_2	1	0	1	0	0
t_3	1	1	0	1	0
t_4	1	1	1	0	1
t_5	0	1	1	1	0
t_6	0	0	1	1	1
t_7	0	0	0	1	1

(b) Exemplo de uma seqüência

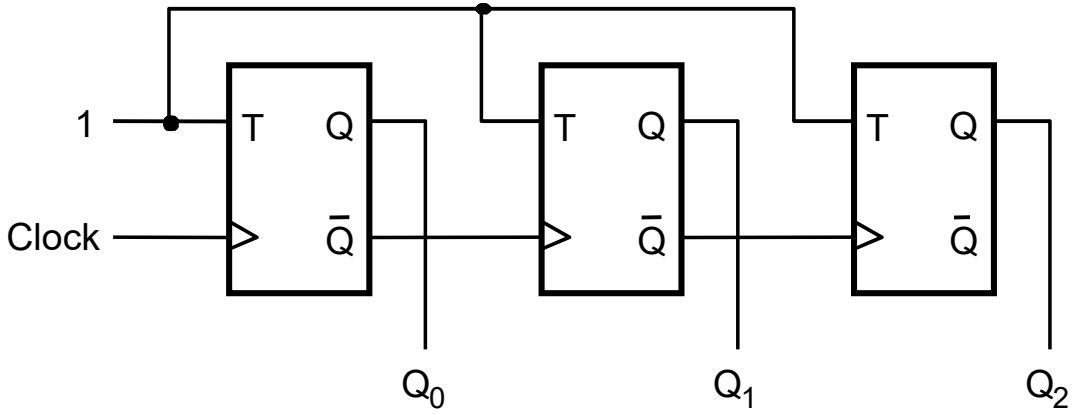
O que seria este circuito ??????????



Registrador de deslocamento com entrada paralela e serial e saída paralela



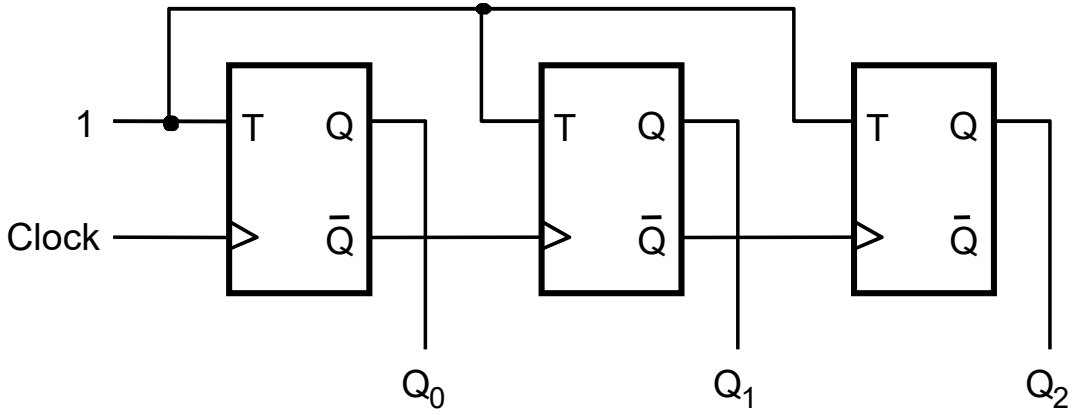
Que circuito é este ??????????



(a) Circuito

Funcionamento tempo ?????

Que circuito é este ??????????



(a) Circuito

Funcionamento tempo ?????

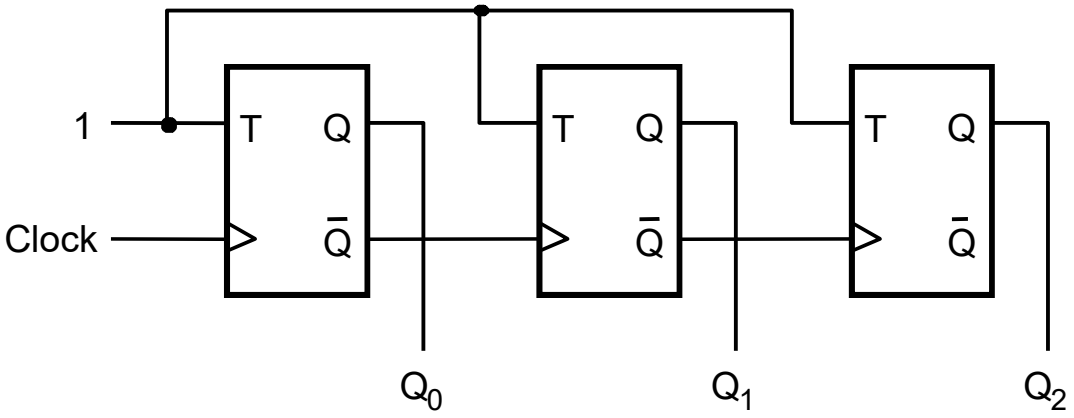
CLOCK	Q0	Q1	Q2
	0	0	0
t0	1	0	0
t1	0	1	0
t2	1	1	0
t3	0	0	1
t4	1	0	1

Q'0	Q'1	Q'2
1	1	1
0	1	1
1	0	1
0	0	1
1	1	0
0	1	0

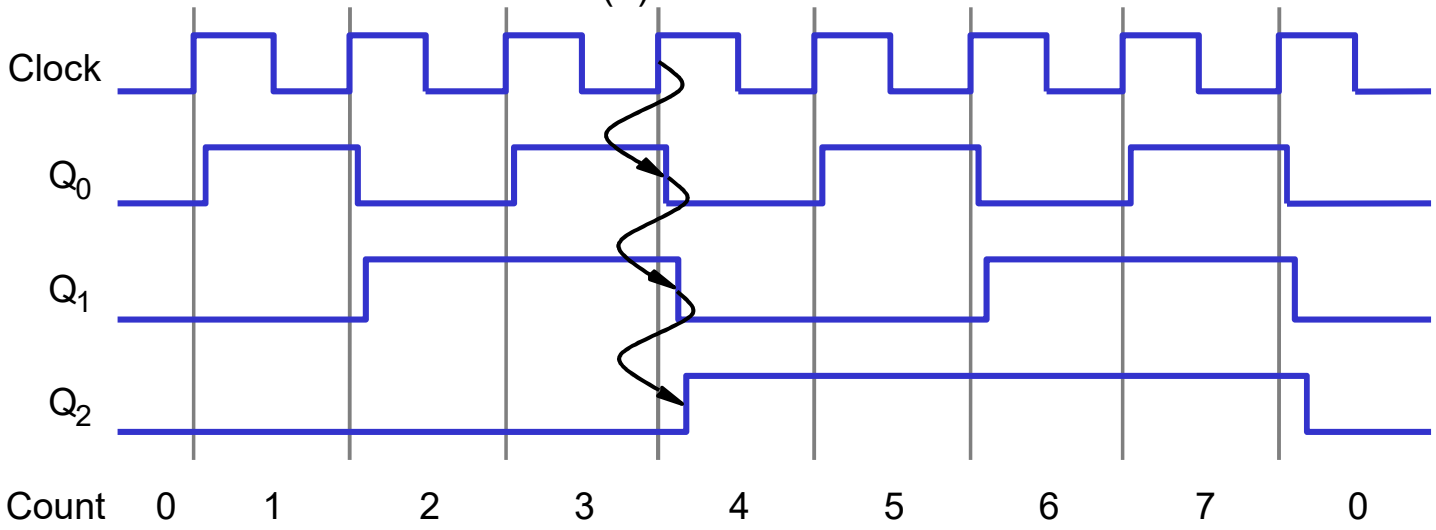
.....

Contadores

Contador assíncrono de 3 bits crescente

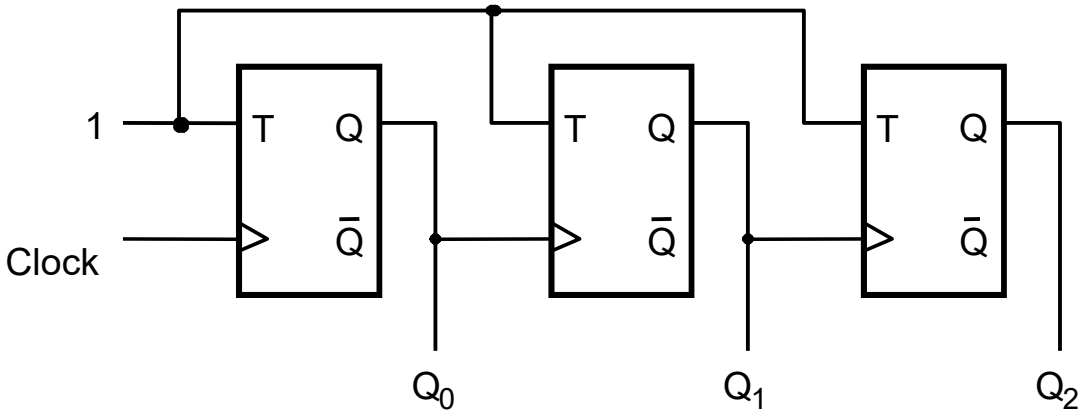


(a) Circuito



(b) Diagrama de tempo

Que circuito é este ??????????



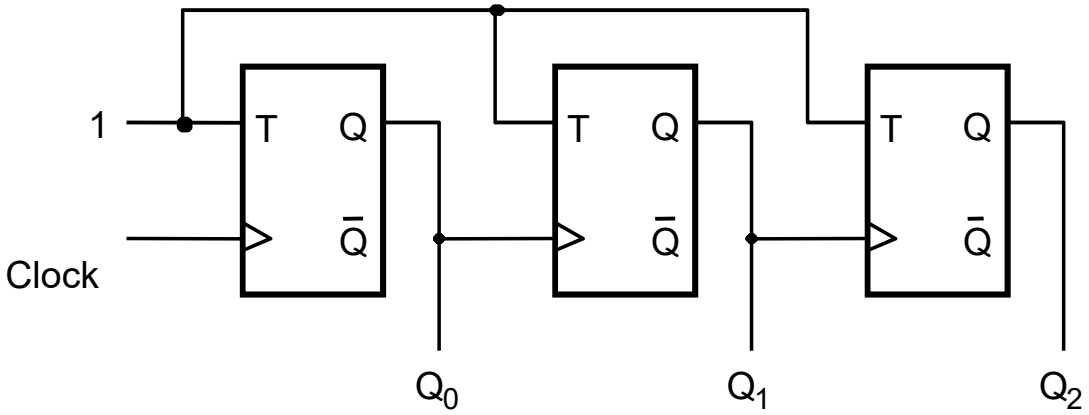
(a) Circuit

Funcionamento tempo ?????

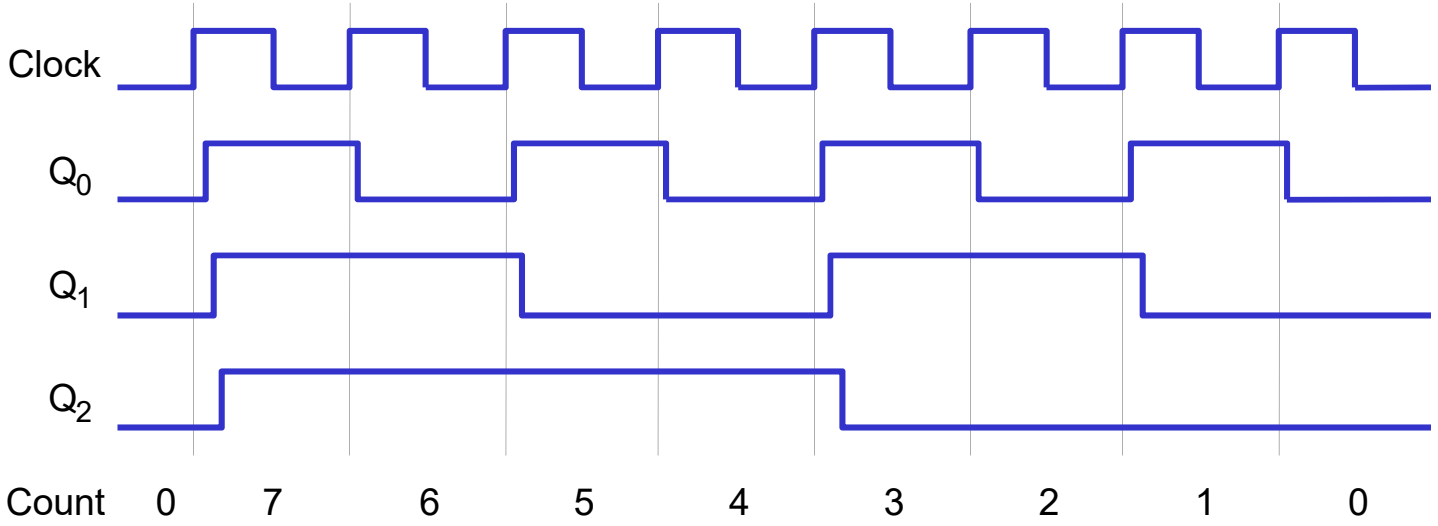
CLOCK	Q0	Q1	Q2
	0	0	0
t0	1	1	1
t1	0	1	1
t2	1	0	1
t3	0	0	1
t4			

.....

Contador assíncronos de 3 bits decrescente



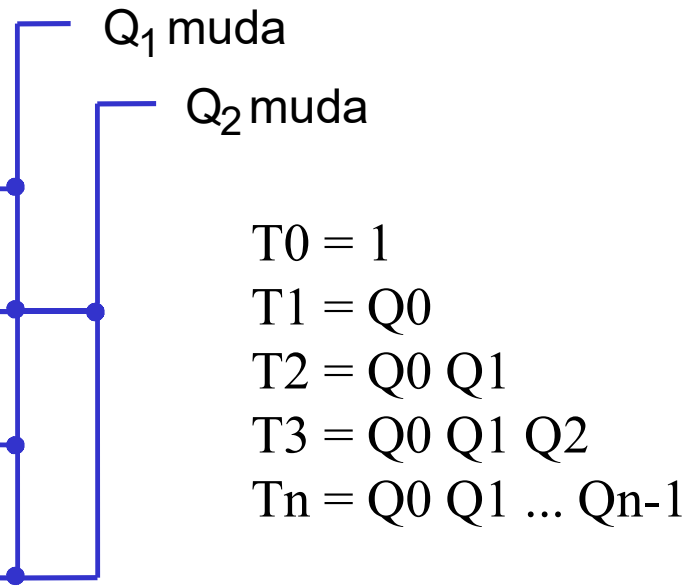
(a) Circuit



(b) Timing diagram

Contador síncrono crescente de n bits

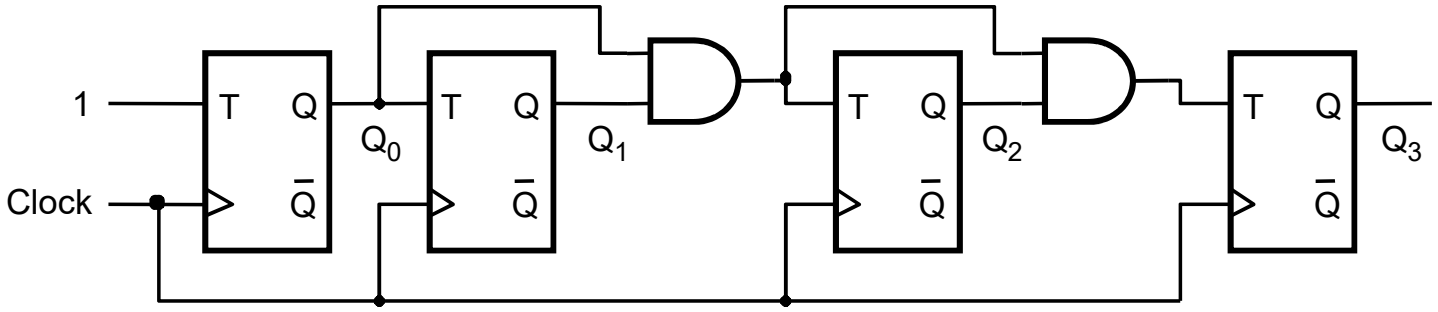
Clock cycle	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0



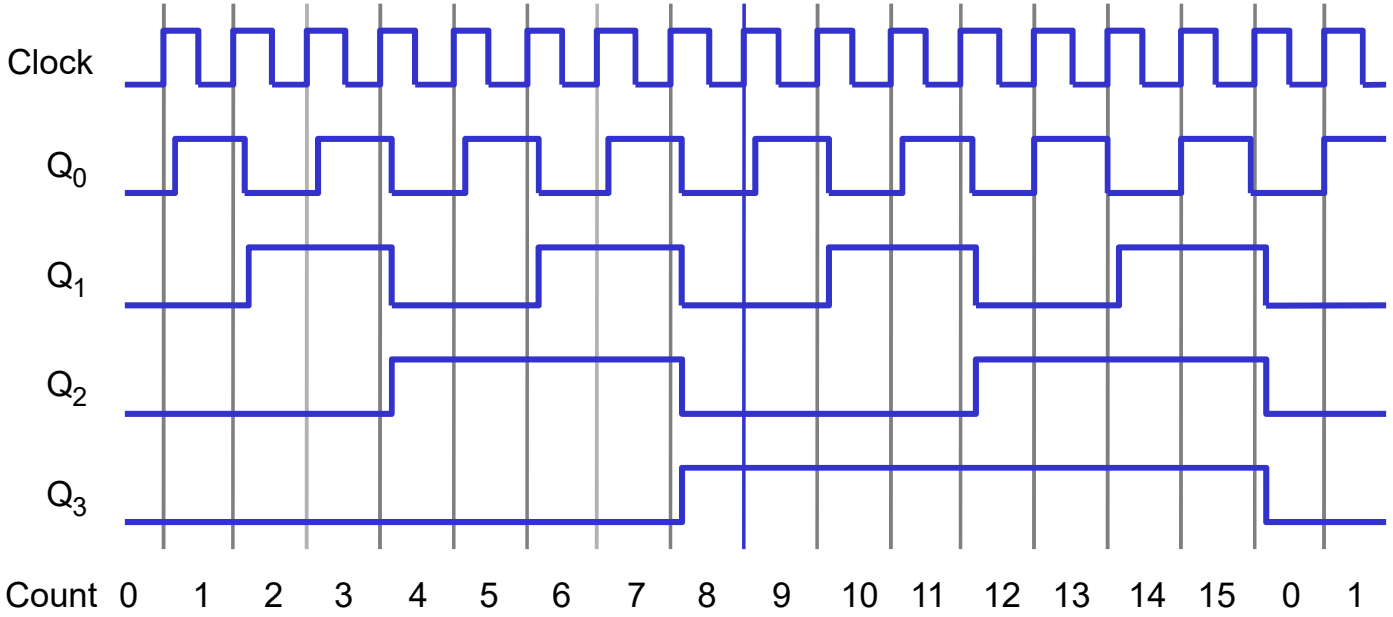
T0 = 1
T1 = Q0
T2 = Q0 Q1
T3 = Q0 Q1 Q2
Tn = Q0 Q1 ... Qn-1

Projetar um somador crescente de 4 bits

Contador síncrono de quatro-bits crescente

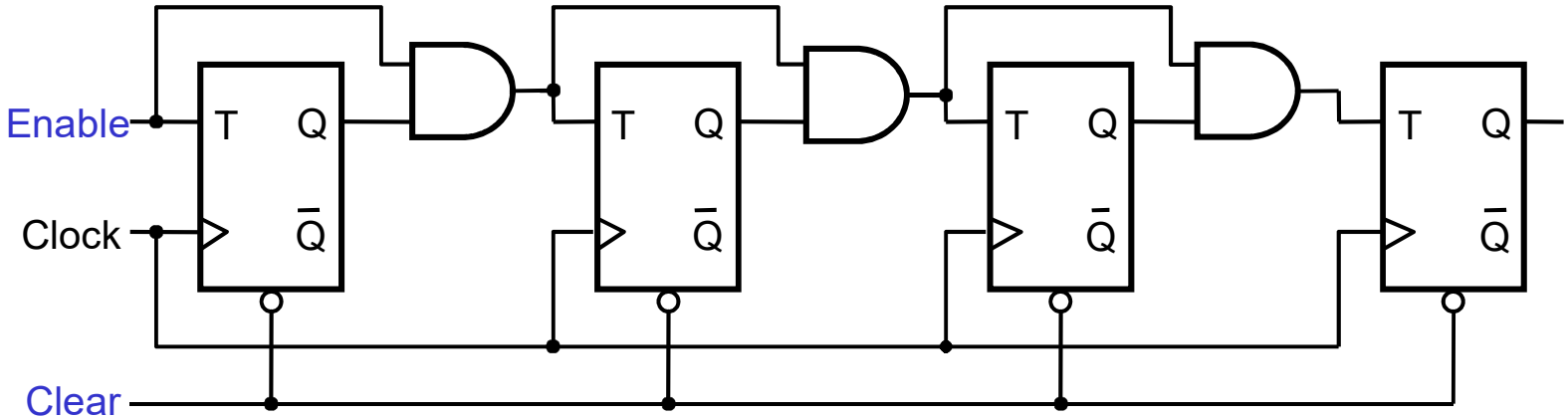


(a) Circuito



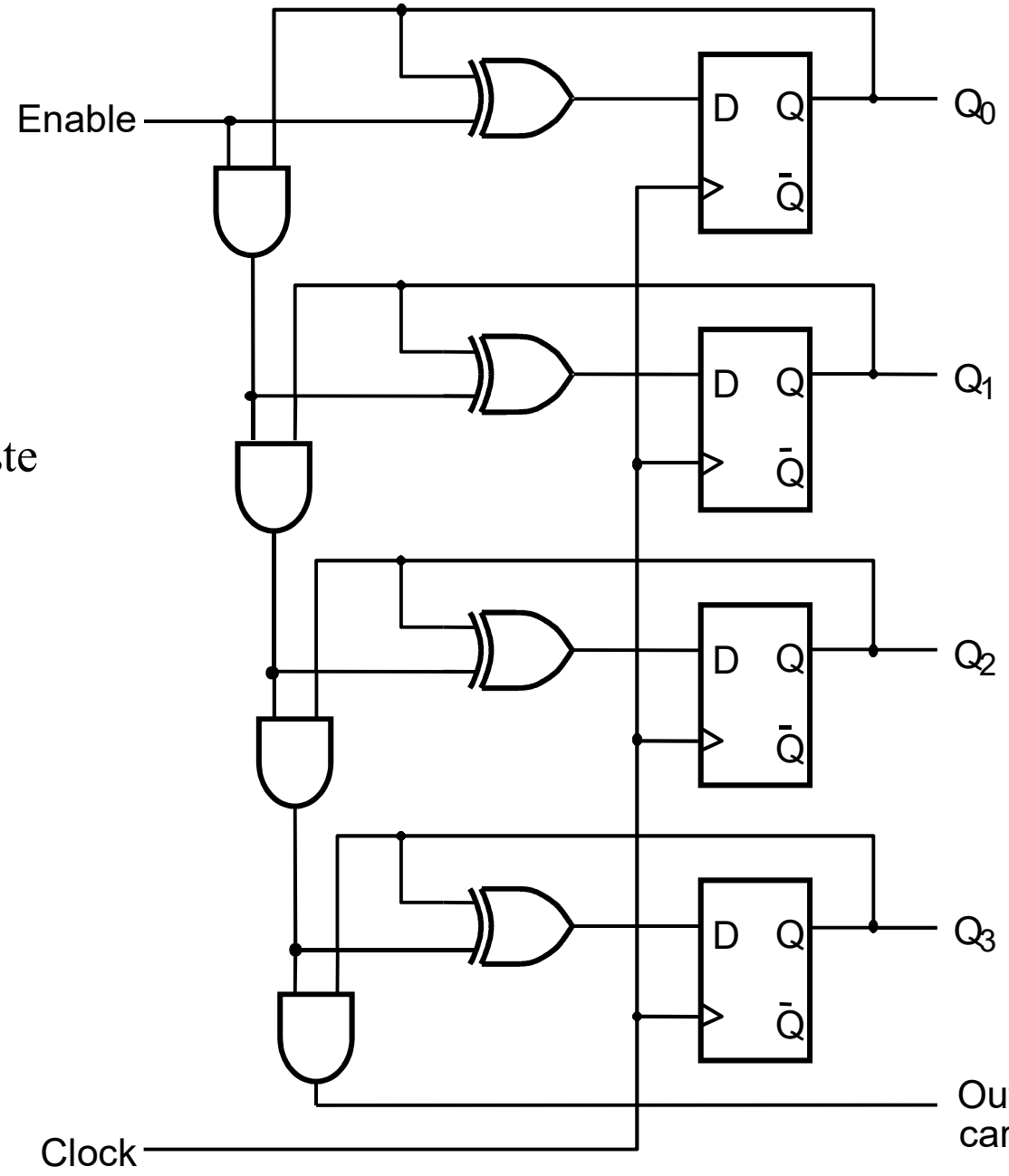
(b) Diagrama de tempo

Inclusão de sinais de enable e clear

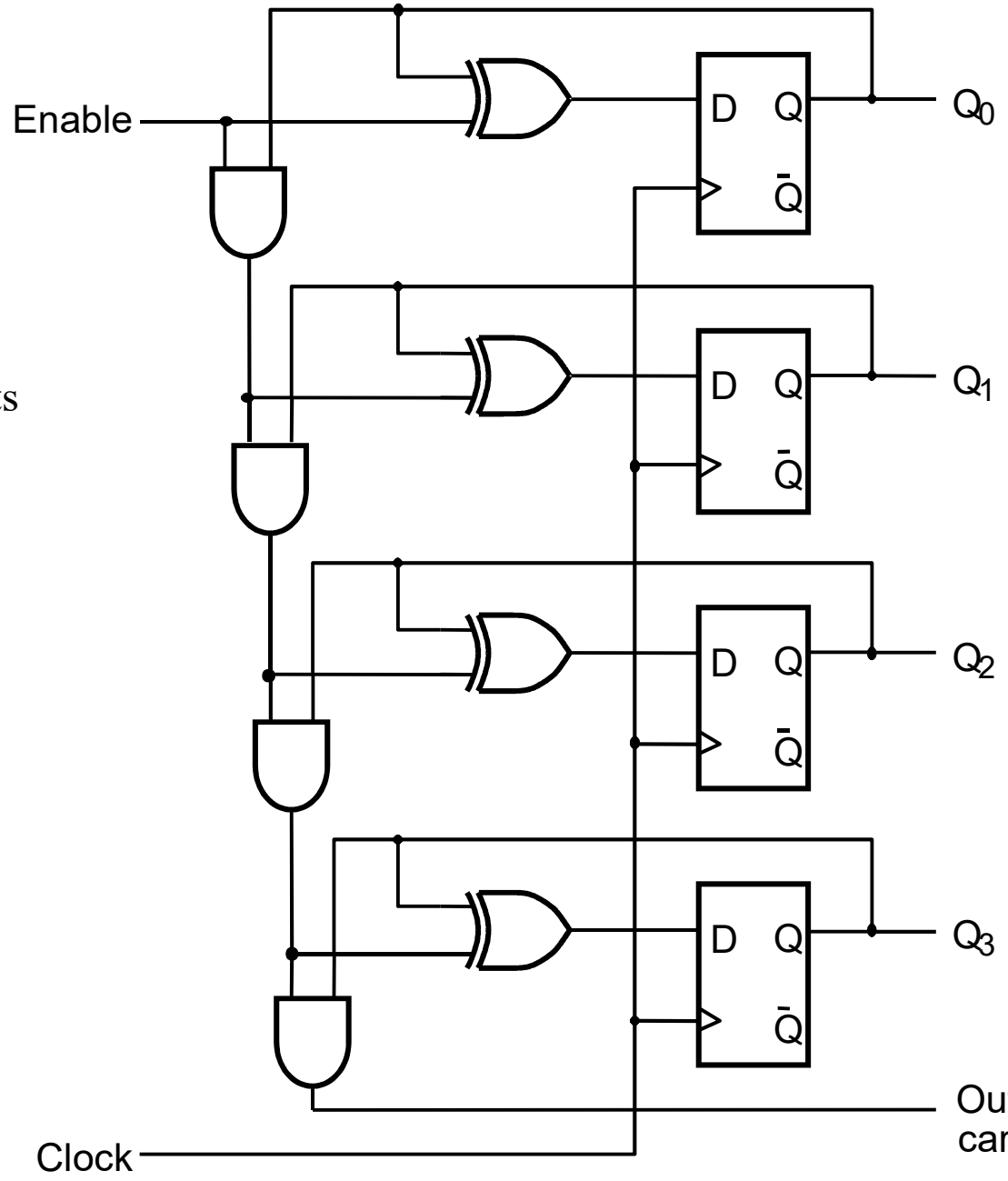


Sinais de enable e clear síncronos ou assíncronos ?

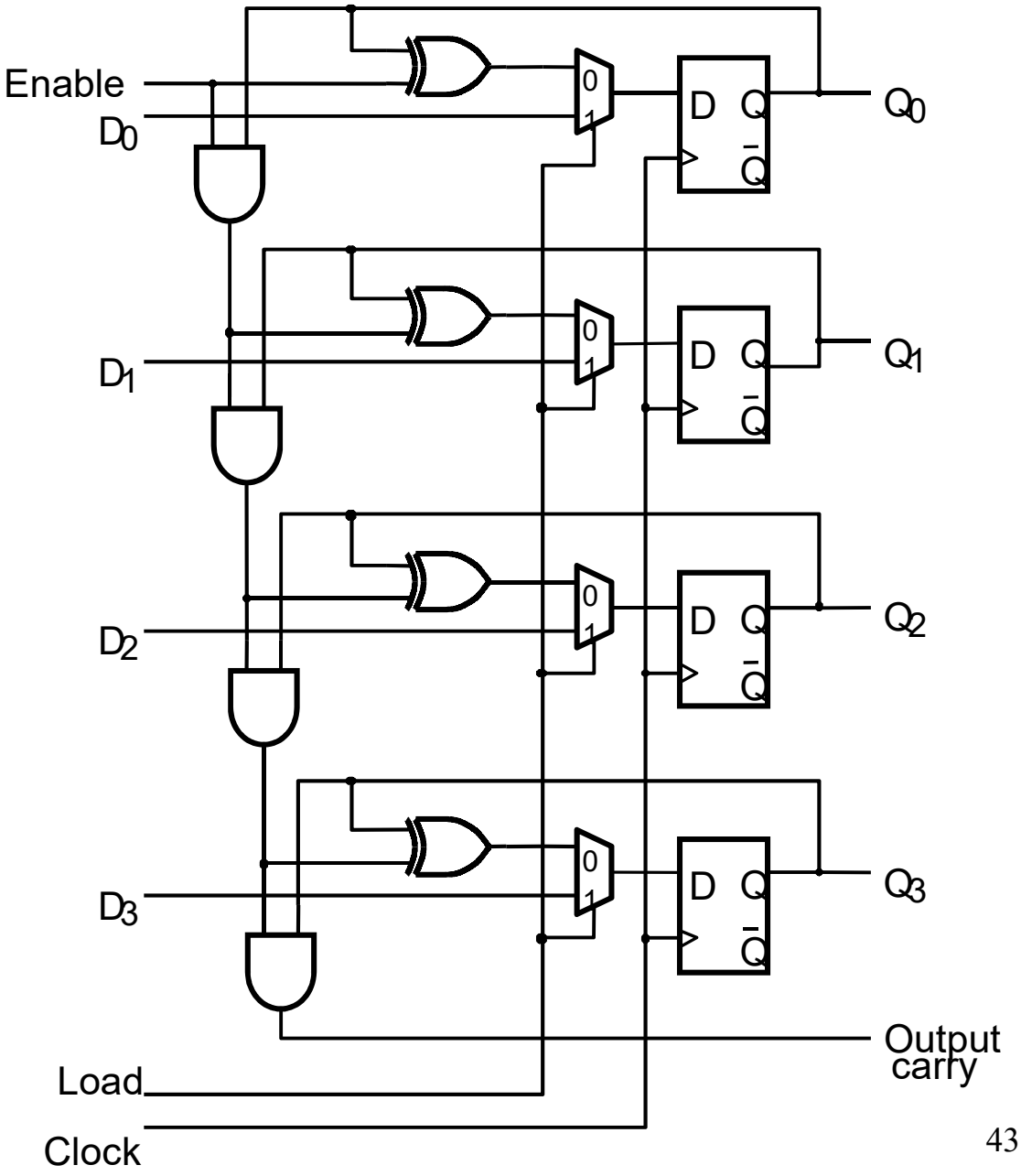
Que circuito é este
??????????



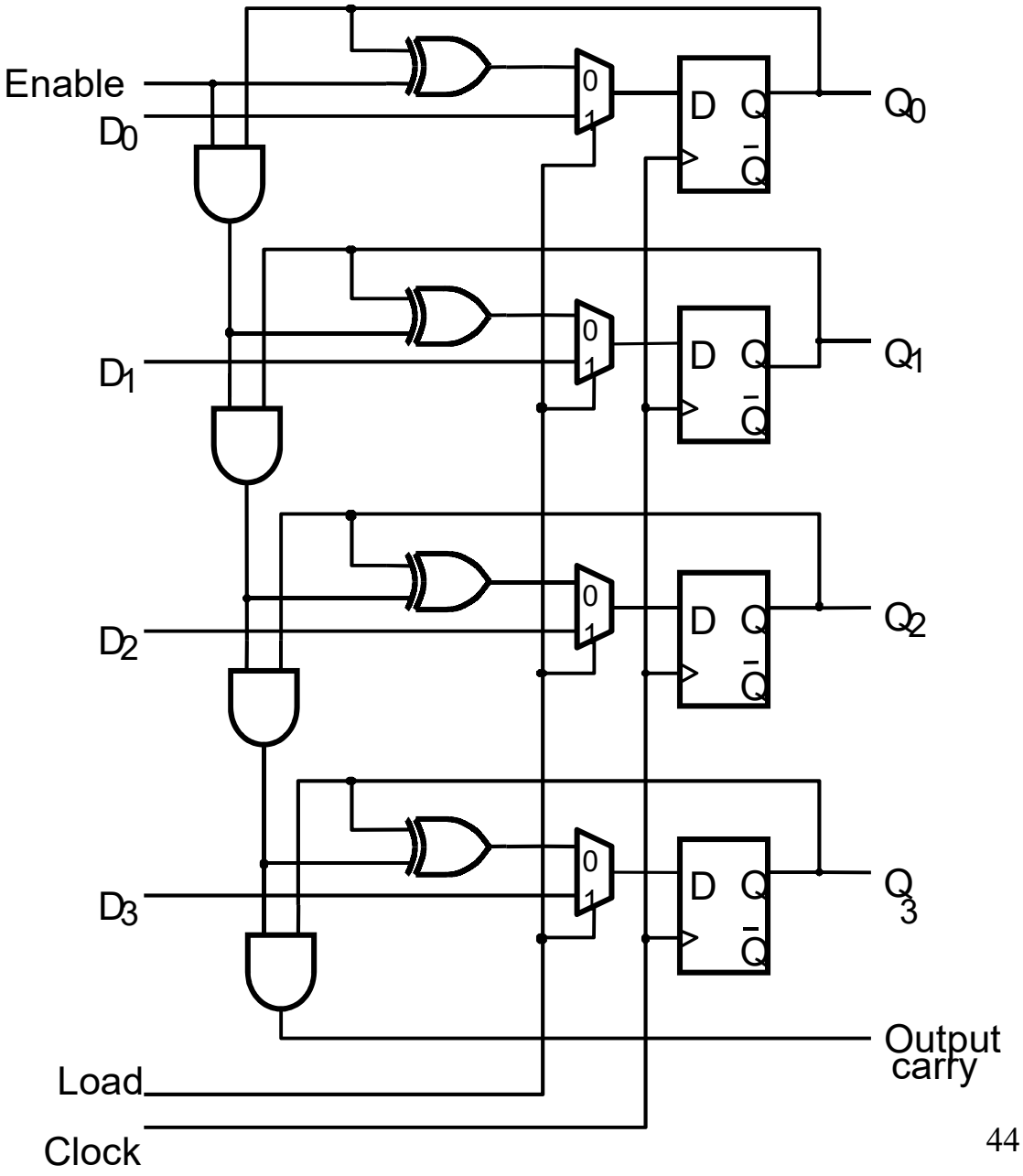
Contador síncrono de 4 bits
com FF D



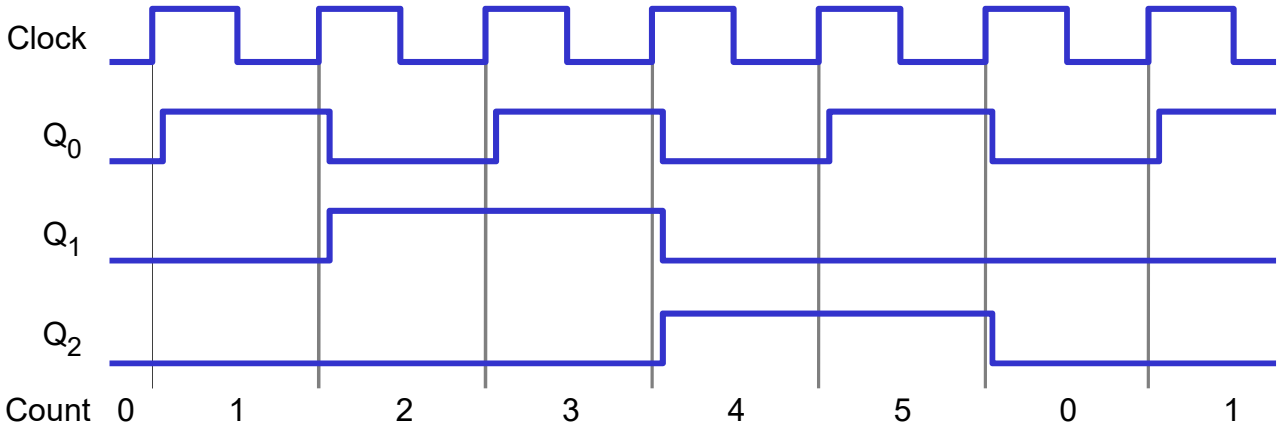
Que circuito é este
??????????



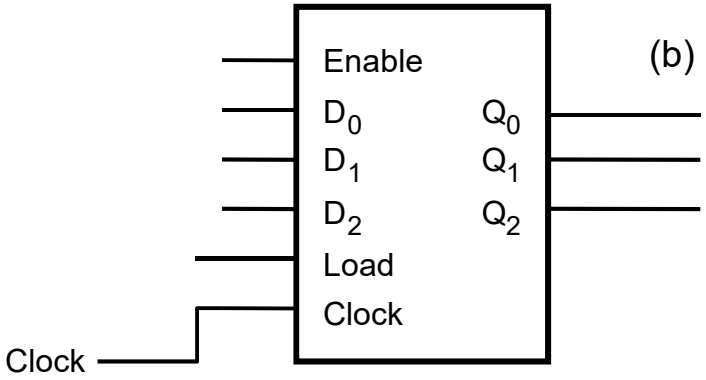
Contador síncrono de 4 bits
com entrada paralela



Projetar contador crescente modulo-6 com reset síncrono utilizando o contador de entrada paralela de 3 bits



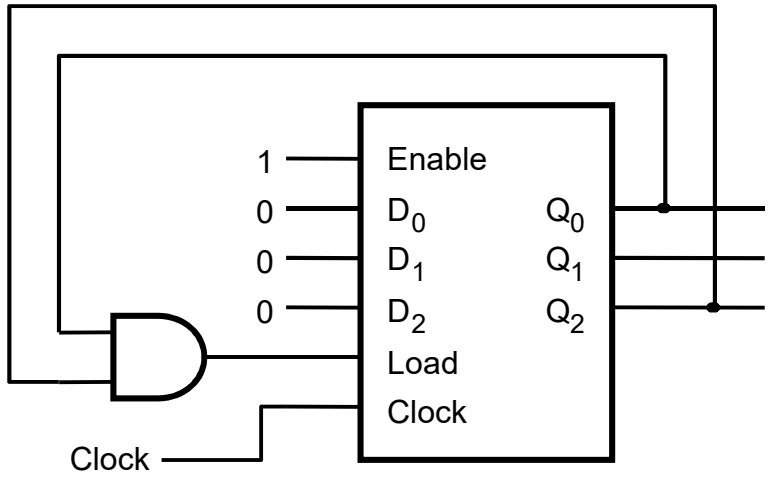
(b) Diagrama de tempo



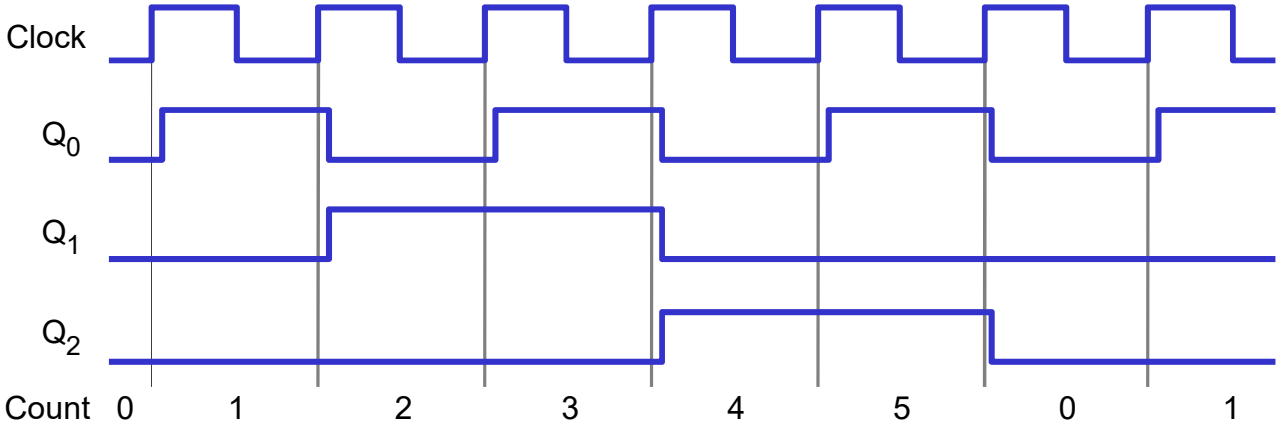
(a) Circuito

O que fazer ????? ϕγϕγϕγϕγϕ

Contador crescente modulo-6 com reset síncrono

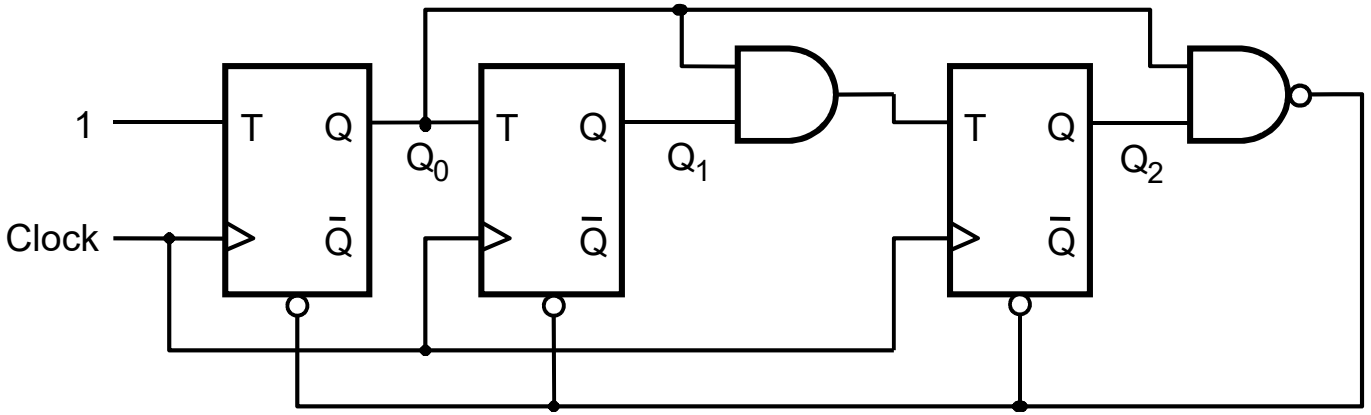


(a) Circuito

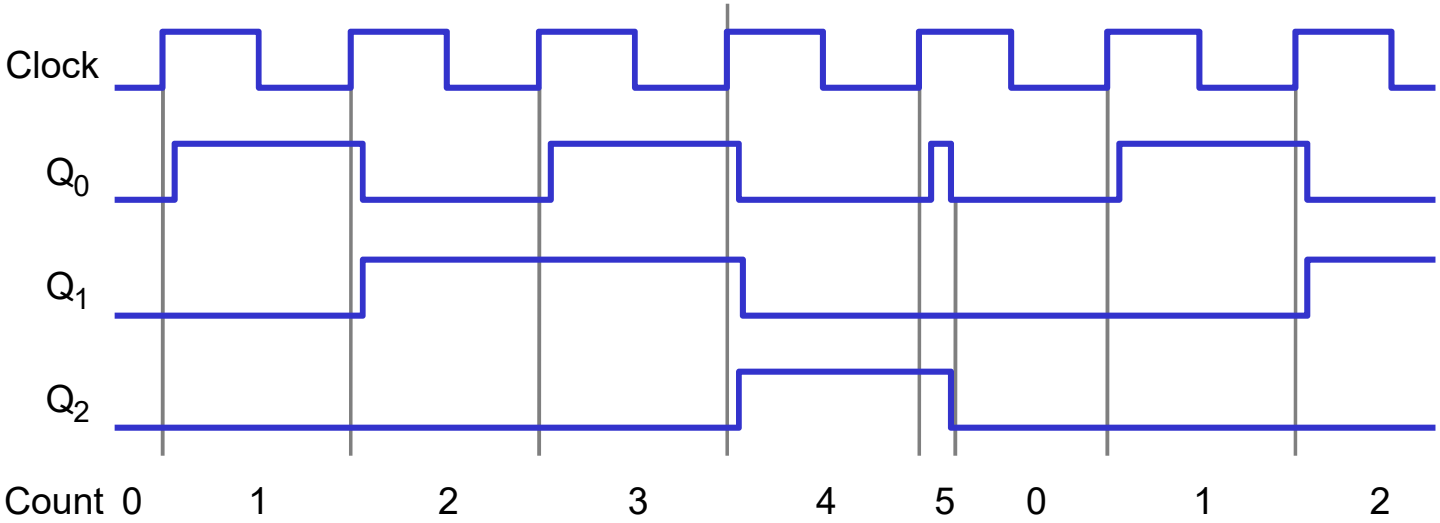


(b) Diagrama de tempo

Contador modulo-6 com reset assíncrono



(a) Circuito

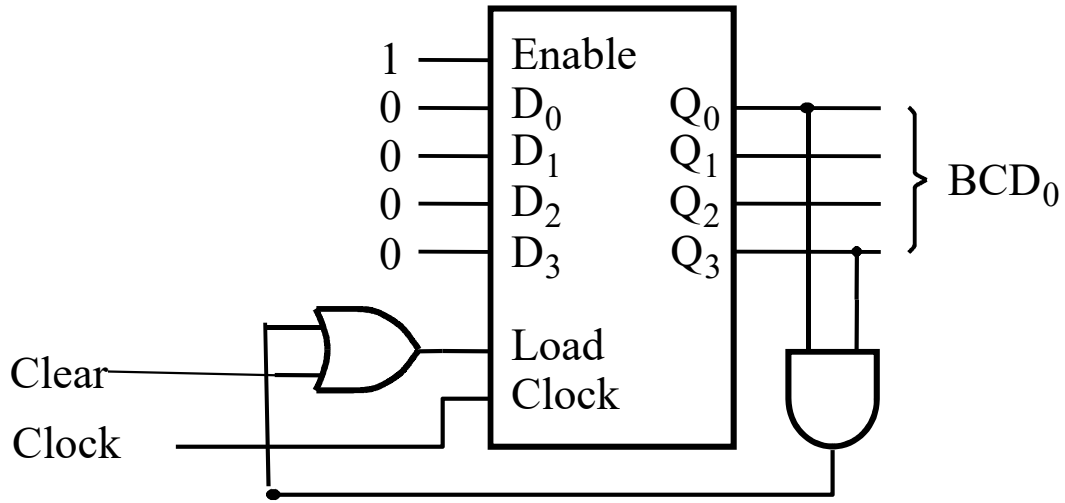


(b) Diagrama de tempo

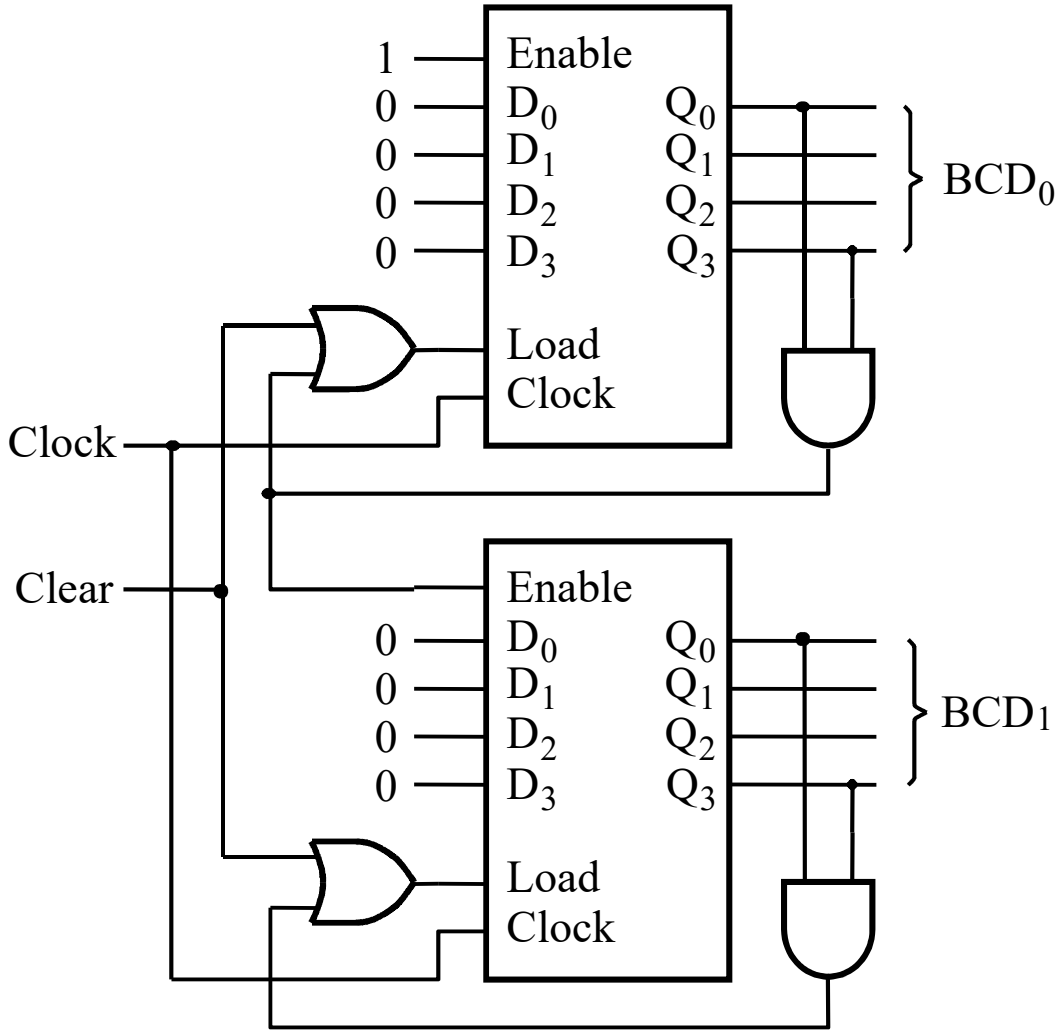
Projetar um contador BCD de 2 dígitos

O que fazer ????? O θυε φαζερ ?????

Contador BCD de 1 dígito



Contador BCD de 2 dígitos



Código para um latch D com clock

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY latch IS
      PORT (  D, Clk    : IN       STD_LOGIC ;
                  Q           : OUT    STD_LOGIC) ;
END latch ;

ARCHITECTURE Behavior OF latch IS
BEGIN
      PROCESS (D, Clk )
      BEGIN
            IF Clk = '1' THEN
                  Q <= D ;
            END IF ;
      END PROCESS ;
END Behavior ;
```

Código para um FF D sensível a borda de subida

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY flipflop IS
      PORT (   D, Clock  : IN      STD_LOGIC ;
               Q         : OUT   STD_LOGIC) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
      PROCESS (Clock )
      BEGIN
          IF Clock'EVENT AND Clock = '1' THEN
              Q <= D ;
          END IF ;
      END PROCESS ;
END Behavior ;
```

Código para um FF D sensível a borda de subida – WAIT...UNTIL

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY flipflop IS
        PORT (   D, Clock  : IN        STD_LOGIC ;
                 Q         : OUT     STD_LOGIC ) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
       PROCESS
       BEGIN
            WAIT UNTIL Clock'EVENT AND Clock = '1' ;
            Q <= D ;
       END PROCESS ;
END Behavior ;
```

Código para um FF D sensível a borda de subida reset assíncrono

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY flipflop IS
      PORT (   D, Resetn, Clock    : IN       STD_LOGIC ;
               Q                   : OUT      STD_LOGIC) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
      PROCESS ( Resetn, Clock )
      BEGIN
            IF Resetn = '0' THEN
                  Q <= '0' ;
            ELSIF Clock'EVENT AND Clock = '1' THEN
                  Q <= D ;
            END IF ;
      END PROCESS ;
END Behavior ;
```

Código para um FF D sensível a borda de subida reset síncrono

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY flipflop IS
        PORT (  D, Resetn, Clock    : IN        STD_LOGIC ;
                Q                   : OUT       STD_LOGIC) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
        PROCESS
        BEGIN
                WAIT UNTIL Clock'EVENT AND Clock = '1' ;
                IF Resetn = '0' THEN
                        Q <= '0' ;
                ELSE
                        Q <= D ;
                END IF ;
        END PROCESS ;
END Behavior ;
```

Código para um registrador de 8 bits com clear assíncrono

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY reg8 IS
    PORT (  D               : IN    STD_LOGIC_VECTOR(7 DOWNTO 0) ;
            Resetn, Clock   : IN    STD_LOGIC ;
            Q               : OUT   STD_LOGIC_VECTOR(7 DOWNTO 0) ) ;
END reg8 ;

ARCHITECTURE Behavior OF reg8 IS
BEGIN
    PROCESS ( Resetn, Clock )
    BEGIN
        IF Resetn = '0' THEN
            Q <= "00000000" ;
        ELSIF Clock'EVENT AND Clock = '1' THEN
            Q <= D ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

Código para um registrador de n bits com clear assíncrono

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY regn IS
        GENERIC ( N : INTEGER := 16 ) ;
        PORT (    D                 : IN      STD_LOGIC_VECTOR(N-1 DOWNTO 0) ;
                  Resetn, Clock     : IN      STD_LOGIC ;
                  Q                 : OUT     STD_LOGIC_VECTOR(N-1 DOWNTO 0) ) ;
END regn ;

ARCHITECTURE Behavior OF regn IS
BEGIN
        PROCESS ( Resetn, Clock )
        BEGIN
                IF Resetn = '0' THEN
                        Q <= (OTHERS => '0') ;
                ELSIF Clock'EVENT AND Clock = '1' THEN
                        Q <= D ;
                END IF ;
        END PROCESS ;
END Behavior ;
```

Código para um FF D com um MUX 2:1 na entrada D

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY muxdff IS
    PORT (   D0, D1, Sel, Clock  : IN     STD_LOGIC ;
             Q                   : OUT    STD_LOGIC ) ;
END muxdff ;

ARCHITECTURE Behavior OF muxdff IS
BEGIN
    PROCESS
    BEGIN
        WAIT UNTIL Clock'EVENT AND Clock = '1' ;
        IF Sel = '0' THEN
            Q <= D0 ;
        ELSE
            Q <= D1 ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

Código hierárquico para um registrador de deslocamento de 4 bits

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY shift4 IS
    PORT (  R              : IN      STD_LOGIC_VECTOR(3 DOWNTO 0) ;
            L, w, Clock    : IN      STD_LOGIC ;
            Q              : BUFFER  STD_LOGIC_VECTOR(3 DOWNTO 0) ) ;
END shift4 ;

ARCHITECTURE Structure OF shift4 IS
    COMPONENT muxdff
        PORT (  D0, D1, Sel, Clock : IN   STD_LOGIC ;
                Q                  : OUT  STD_LOGIC ) ;
    END COMPONENT ;
BEGIN
    Stage3: muxdff PORT MAP ( w, R(3), L, Clock, Q(3) ) ;
    Stage2: muxdff PORT MAP ( Q(3), R(2), L, Clock, Q(2) ) ;
    Stage1: muxdff PORT MAP ( Q(2), R(1), L, Clock, Q(1) ) ;
    Stage0: muxdff PORT MAP ( Q(1), R(0), L, Clock, Q(0) ) ;
END Structure ;
```

Código alternativo para um registrador de deslocamento de 4 bits

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY shift4 IS
        PORT (  R           : IN        STD_LOGIC_VECTOR(3 DOWNTO 0) ;
                Clock       : IN        STD_LOGIC ;
                L, w        : IN        STD_LOGIC ;
                Q           : BUFFER    STD_LOGIC_VECTOR(3 DOWNTO 0) ) ;
END shift4 ;

ARCHITECTURE Behavior OF shift4 IS
BEGIN
        PROCESS
        BEGIN
                WAIT UNTIL Clock'EVENT AND Clock = '1' ;
                IF L = '1' THEN
                        Q <= R ;
                ELSE
                        Q(0) <= Q(1) ;
                        Q(1) <= Q(2);
                        Q(2) <= Q(3) ;
                        Q(3) <= w ;
                END IF ;
        END PROCESS ;
END Behavior ;
```

Código para um registrador de deslocamento (esquerda para a direita) de n bits

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY shiftn IS
        GENERIC ( N : INTEGER := 8 ) ;
        PORT (      R           : IN        STD_LOGIC_VECTOR(N-1 DOWNTO 0) ;
                    Clock       : IN        STD_LOGIC ;
                    L, w        : IN        STD_LOGIC ;
                    Q           : BUFFER    STD_LOGIC_VECTOR(N-1 DOWNTO 0) ) ;
END shiftn ;
ARCHITECTURE Behavior OF shiftn IS
BEGIN
        PROCESS
        BEGIN
                WAIT UNTIL Clock'EVENT AND Clock = '1' ;
                IF L = '1' THEN
                        Q <= R ;
                ELSE
                        Genbits: FOR i IN 0 TO N-2 LOOP
                                Q(i) <= Q(i+1) ;
                        END LOOP ;
                        Q(N-1) <= w ;
                END IF ;
        END PROCESS ;
END Behavior ;
```

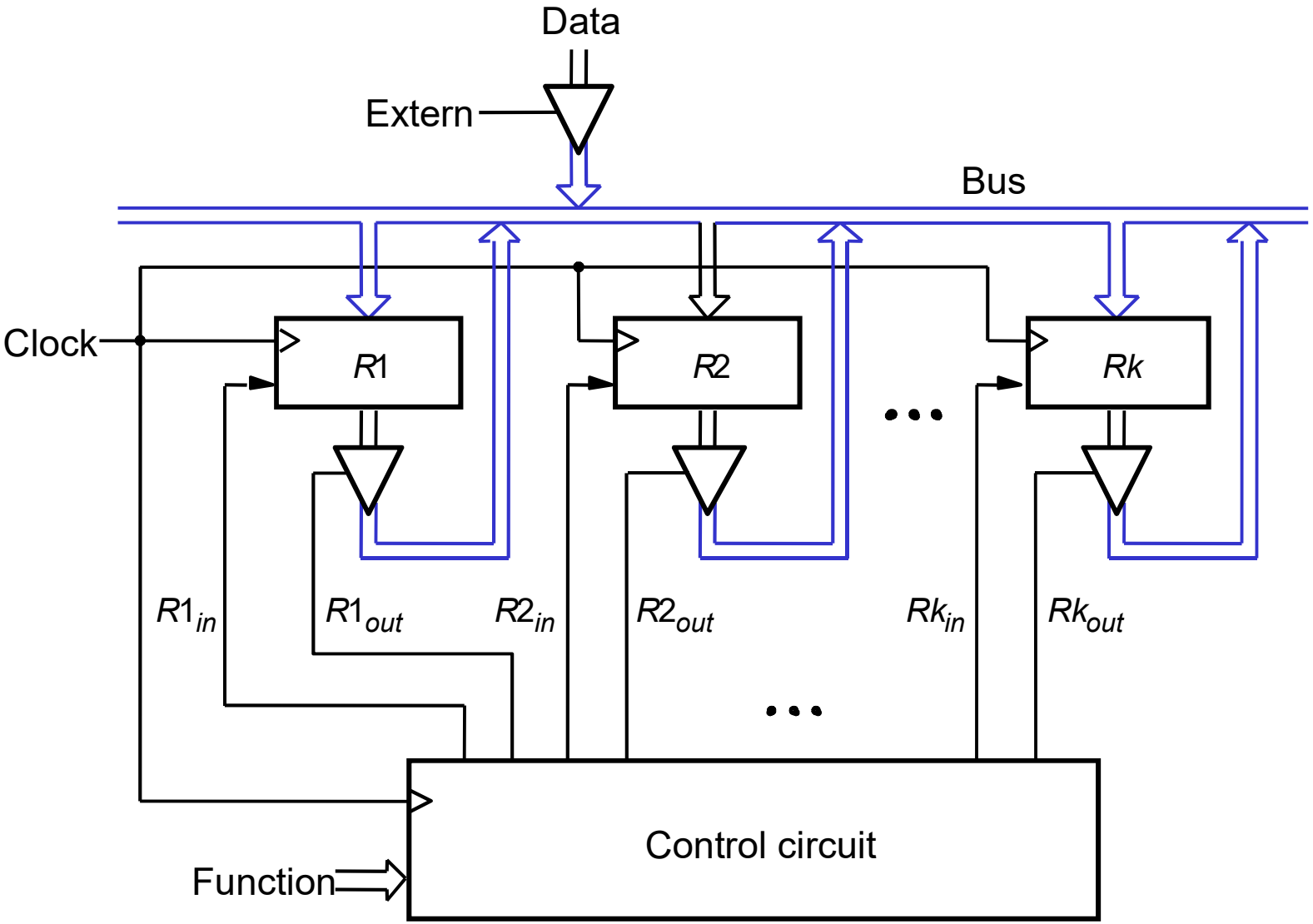
Código para um contador cresecente de 4 bits

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_unsigned.all ;
ENTITY upcount IS
        PORT (      Clock, Resetn, E    : IN      STD_LOGIC ;
                    Q                   : OUT     STD_LOGIC_VECTOR (3 DOWNTO 0)) ;
END upcount ;

ARCHITECTURE Behavior OF upcount IS
        SIGNAL Count : STD_LOGIC_VECTOR (3 DOWNTO 0) ;
BEGIN
        PROCESS ( Clock, Resetn )
        BEGIN
                IF Resetn = '0' THEN
                        Count <= "0000" ;
                ELSIF (Clock'EVENT AND Clock = '1') THEN
                        IF E = '1' THEN
                                Count <= Count + 1 ;
                        ELSE
                                Count <= Count ;
                        END IF ;
                END IF ;
        END PROCESS ;
        Q <= Count ;
END Behavior ;
```

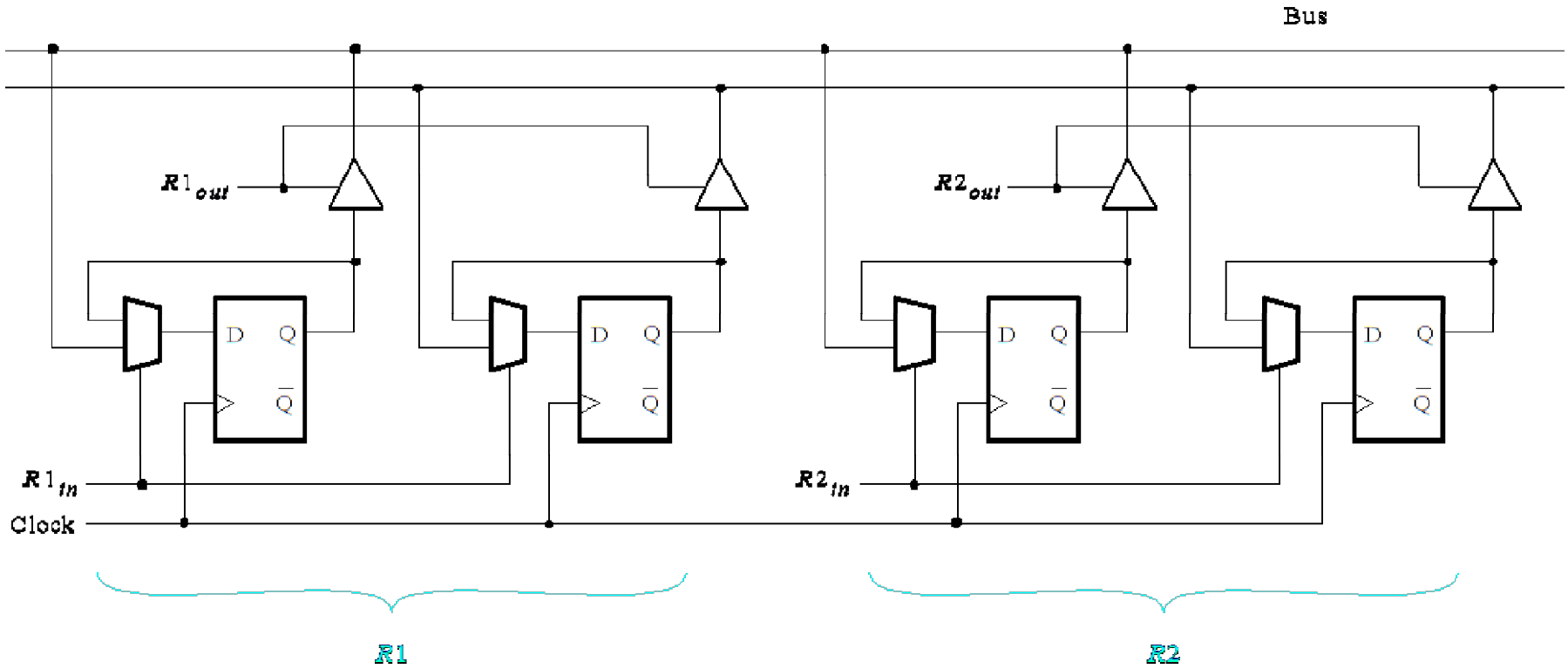
Contador de 4 bits com carga paralela, usando sinais INTEGER

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY upcount IS
        PORT (  R                   : IN        INTEGER RANGE 0 TO 15 ;
                Clock, Resetn, L    : IN        STD_LOGIC ;
                Q                   : BUFFER    INTEGER RANGE 0 TO 15 ) ;
END upcount ;

ARCHITECTURE Behavior OF upcount IS
BEGIN
        PROCESS ( Clock, Resetn )
        BEGIN
                IF Resetn = '0' THEN
                        Q <= 0 ;
                ELSIF (Clock'EVENT AND Clock = '1') THEN
                        IF L = '1' THEN
                                Q <= R ;
                        ELSE
                                Q <= Q + 1 ;
                        END IF;
                END IF;
        END PROCESS;
END Behavior;
```

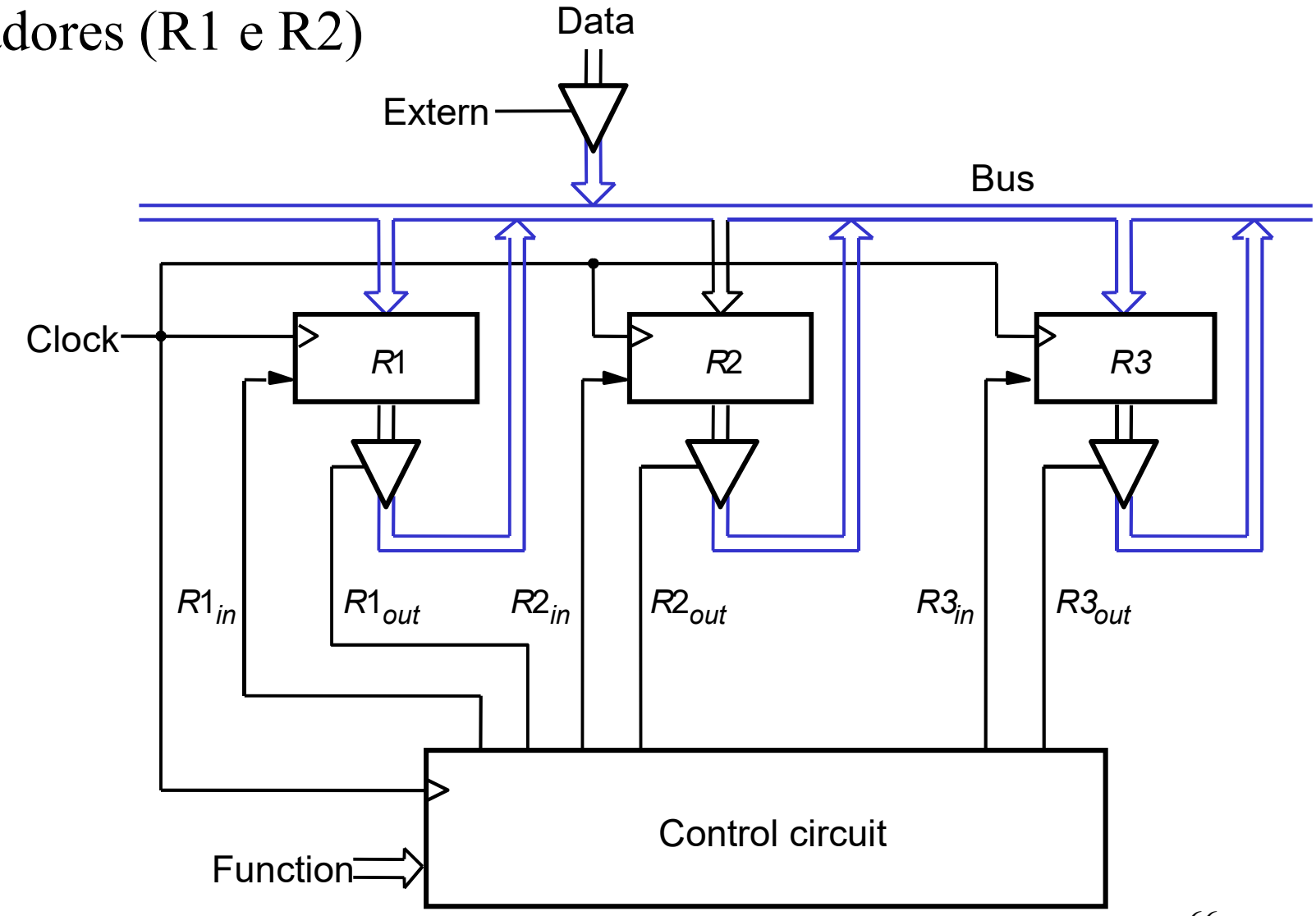
Sistema digital com *k* registradores



Conexão dos registradoes ao barramento



Sistema digital com 3 registradores- Projetar o cicuito de controle para executar a troca entre o conteúdo de 2 registradores (R1 e R2)



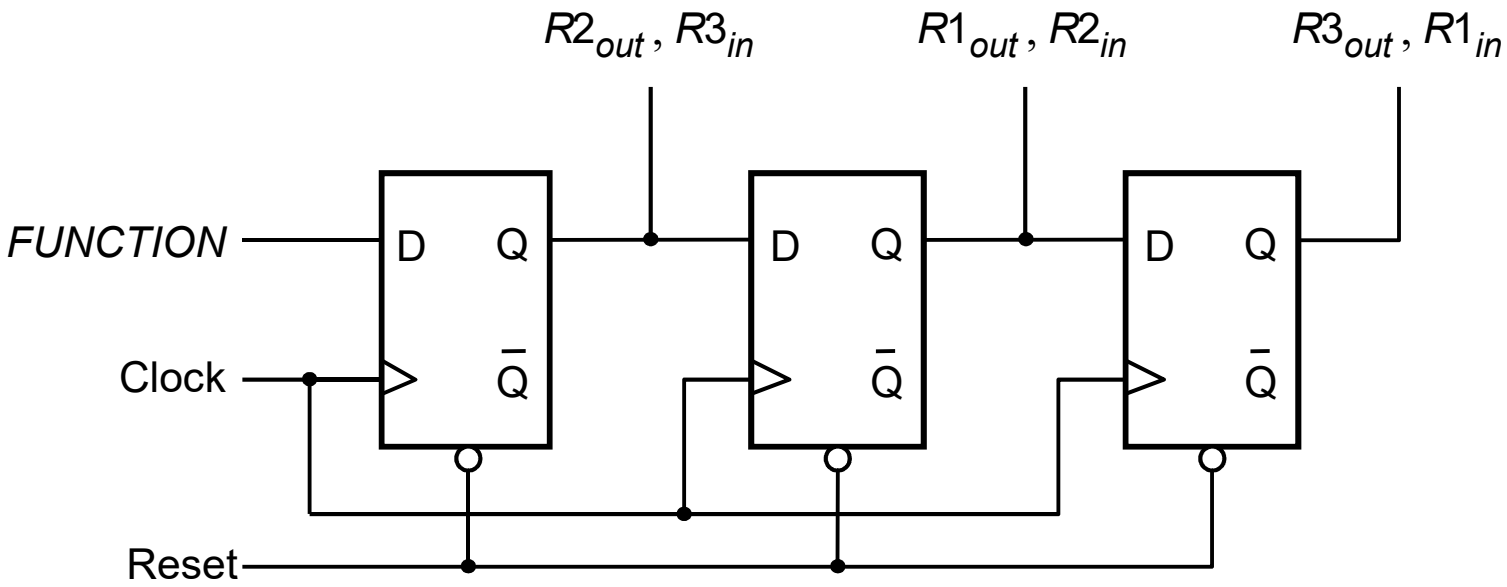
Circuito de controle deve gerar sinais que façam:

R3 ← R2
R2 ← R1
R1 ← R3

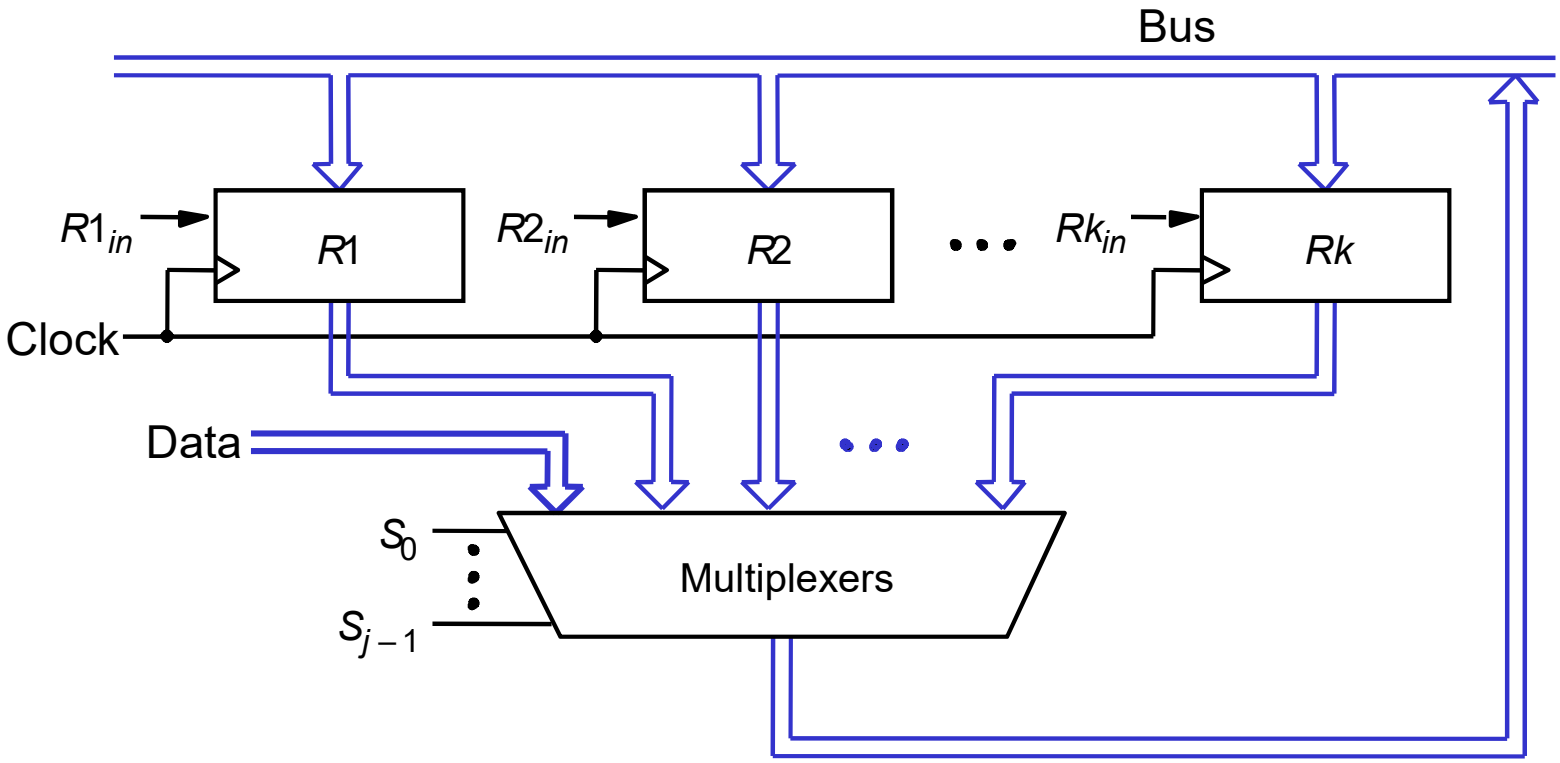
Portanto gerar, em cada ciclo de clock:

T1 →R3in,R2out
T2 →R2in, R1out
T3 →R1in, R3out

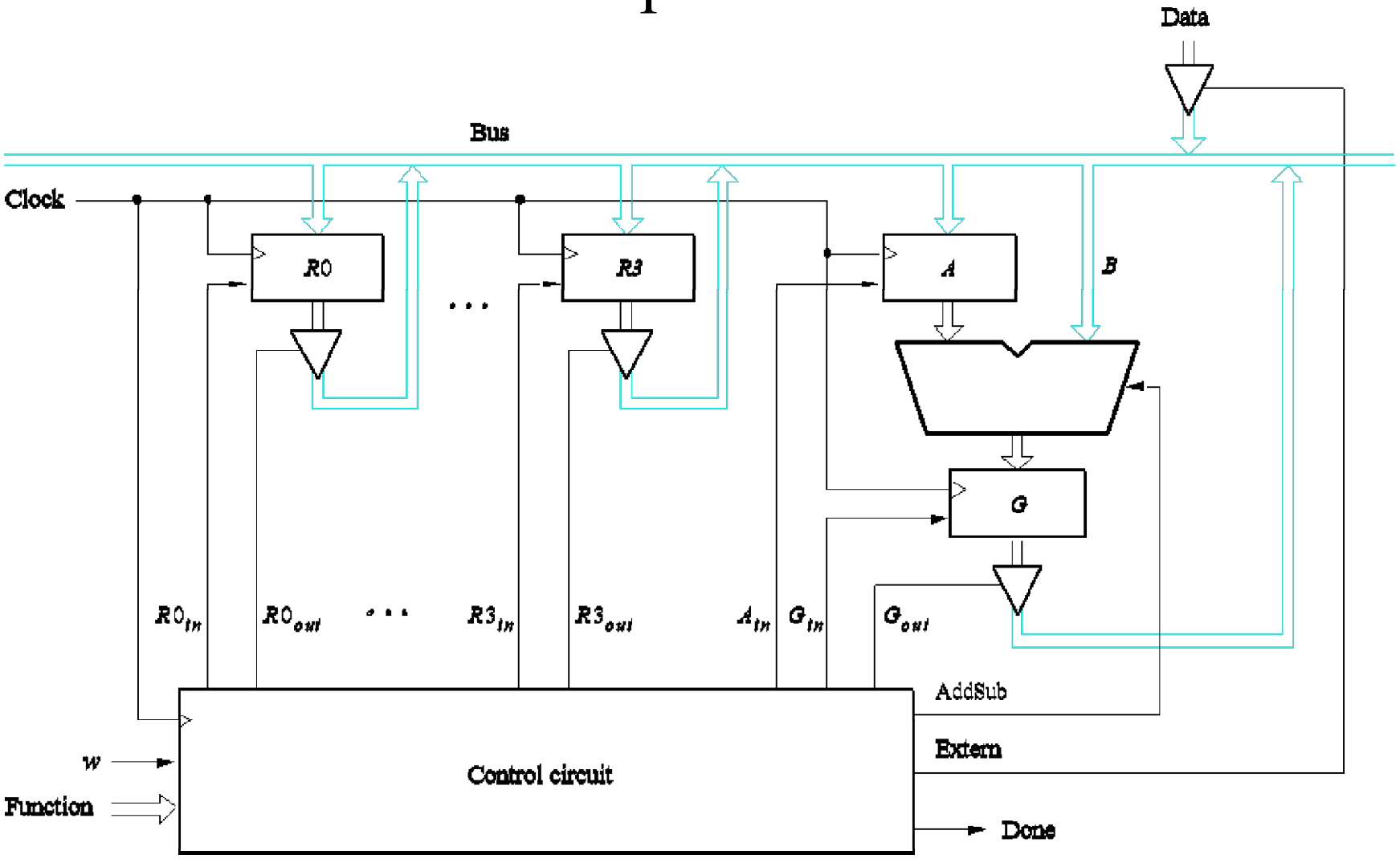
Circuito de controle com um shift-register



Usando multiplexadores para implementação de um barramento



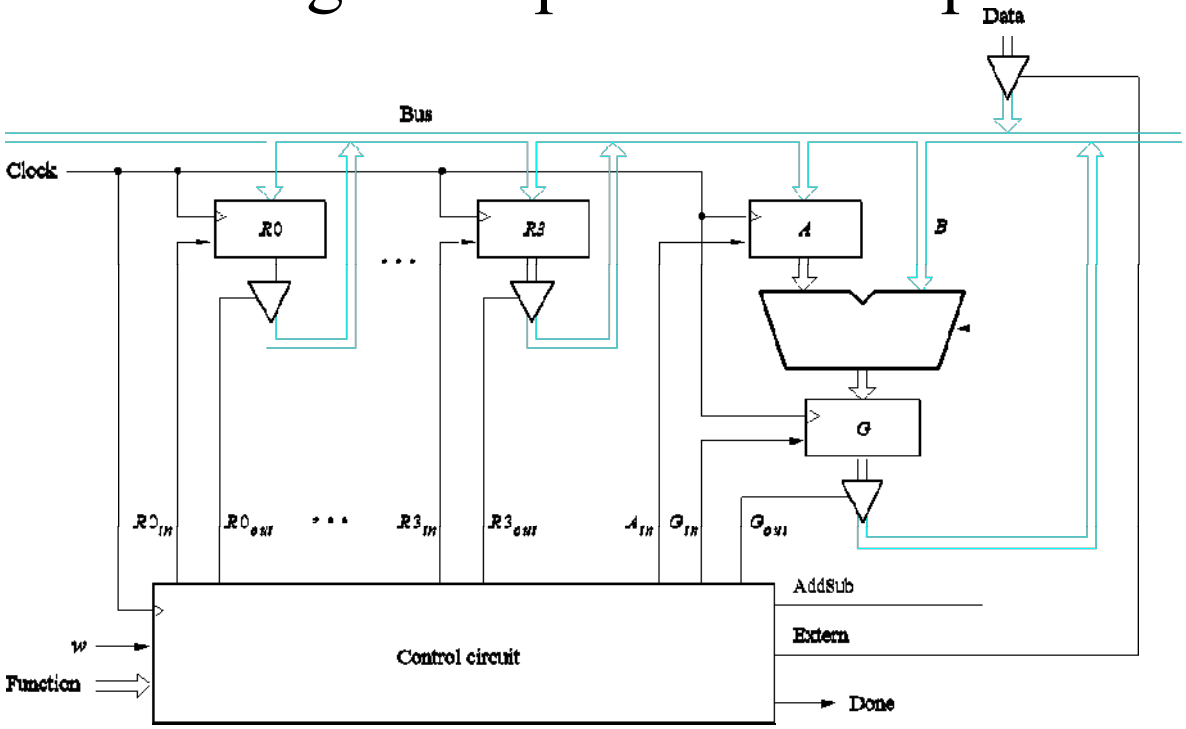
Sistema digital implementa um processador simples



Operacões executadas pelo processador

Operation	Function performed
Load Rx,$Data$	$Rx \leftarrow Data$
Move Rx, Ry	$Rx \leftarrow [Ry]$
Add Rx, Ry	$Rx \leftarrow [Rx] + [Ry]$
Sub Rx, Ry	$Rx \leftarrow [Rx] - [Ry]$

Sistema digital implementa um processador simples



Operation	Function performed
Load Rx,$Data$	$Rx \leftarrow Data$
Move Rx, Ry	$Rx \leftarrow [Ry]$
Add Rx, Ry	$Rx \leftarrow [Rx] + [Ry]$
Sub Rx, Ry	$Rx \leftarrow [Rx] - [Ry]$

MOVE RX,RY
T1 → RYout,RXin, Done

LOAD RX,RY
T1 → Extern, RXin, Done

ADD RX,RY
T1 → RXout, Ain,
T2→ Ryout, AddSub=0,Gin
T3→ Gout, RXin, Done

SUB RX,RY
T1 → RXout, Ain,
T2→ Ryout, AddSub=1,Gin
T3→ Gout, RXin, Done

Valores dos sinais de controle para cada operação/time step

	T_1	T_2	T_3
(Load): I_0	$Extern$, $R_{in} = X$, $Done$		
(Move): I_1	$R_{in} = X$, $R_{out} = Y$, $Done$		
(Add): I_2	$R_{out} = X$, A_{in}	$R_{out} = Y$, G_{in}, $AddSub = 0$	G_{out}, $R_{in} = X$, $Done$
(Sub): I_3	$R_{out} = X$, A_{in}	$R_{out} = Y$, G_{in}, $AddSub = 1$	G_{out}, $R_{in} = X$, $Done$