







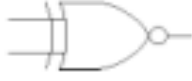
Sistemas Digitais

Registradores

Aula 04

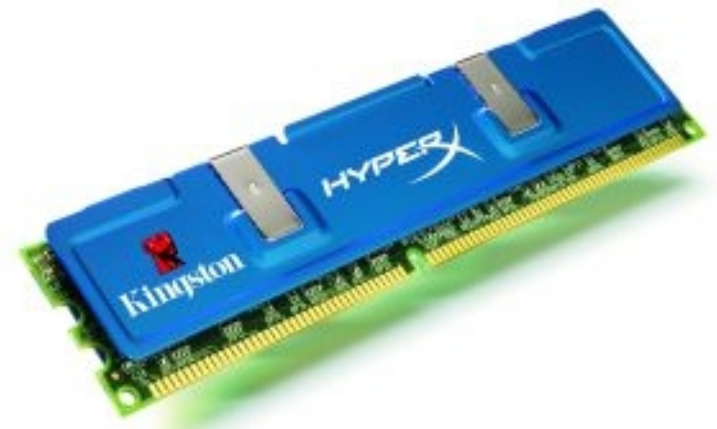
Prof. Leandro Nogueira Couto
UFU – Monte Carmelo
05/2013



<div>E</div> <div>AND</div>		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	S	0	0	0	0	1	0	1	0	0	1	1	1	<div>Função E:</div> <div>Assume 1 quando todas as variáveis forem 1 e 0 nos outros casos.</div>	<div>S=A.B</div>
A	B	S																	
0	0	0																	
0	1	0																	
1	0	0																	
1	1	1																	
<div>OU</div> <div>OR</div>		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	S	0	0	0	0	1	1	1	0	1	1	1	1	<div>Função OU:</div> <div>Assume 0 quando todas as variáveis forem 0 e 1 nos outros casos.</div>	<div>S=A+B</div>
A	B	S																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	1																	
<div>NÃO</div> <div>NOT</div>		<table><tr><th>A</th><th>S</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	S	0	1	1	0	<div>Função NÃO:</div> <div>Inverte a variável aplicada à sua entrada.</div>	<div>S=A</div>									
A	S																		
0	1																		
1	0																		
<div>NE</div> <div>NAND</div>		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	S	0	0	1	0	1	1	1	0	1	1	1	0	<div>Função NE:</div> <div>Inverso da função E.</div>	<div>S=(A.B)</div>
A	B	S																	
0	0	1																	
0	1	1																	
1	0	1																	
1	1	0																	
<div>NOU</div> <div>NOR</div>		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	S	0	0	1	0	1	0	1	0	0	1	1	0	<div>Função NOU:</div> <div>Inverso da função OU.</div>	<div>S=(A+B)</div>
A	B	S																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	0																	
<div>OU EXCLUSIVO</div>		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	S	0	0	0	0	1	1	1	0	1	1	1	0	<div>Função OU Exclusivo:</div> <div>Assume 1 quando as variáveis assumirem valores diferentes entre si.</div>	<div>S=A⊕B</div> <div>S= A.B + A.B</div>
A	B	S																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	0																	
<div>COINCIDÊNCIA</div>		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	S	0	0	1	0	1	0	1	0	0	1	1	1	<div>Função Coincidência:</div> <div>Assume 1 quando houver coincidência entre os valores das variáveis.</div>	<div>S= A⊙B</div> <div>S= A.B + A.B</div>
A	B	S																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	1																	

Circuitos Sequenciais

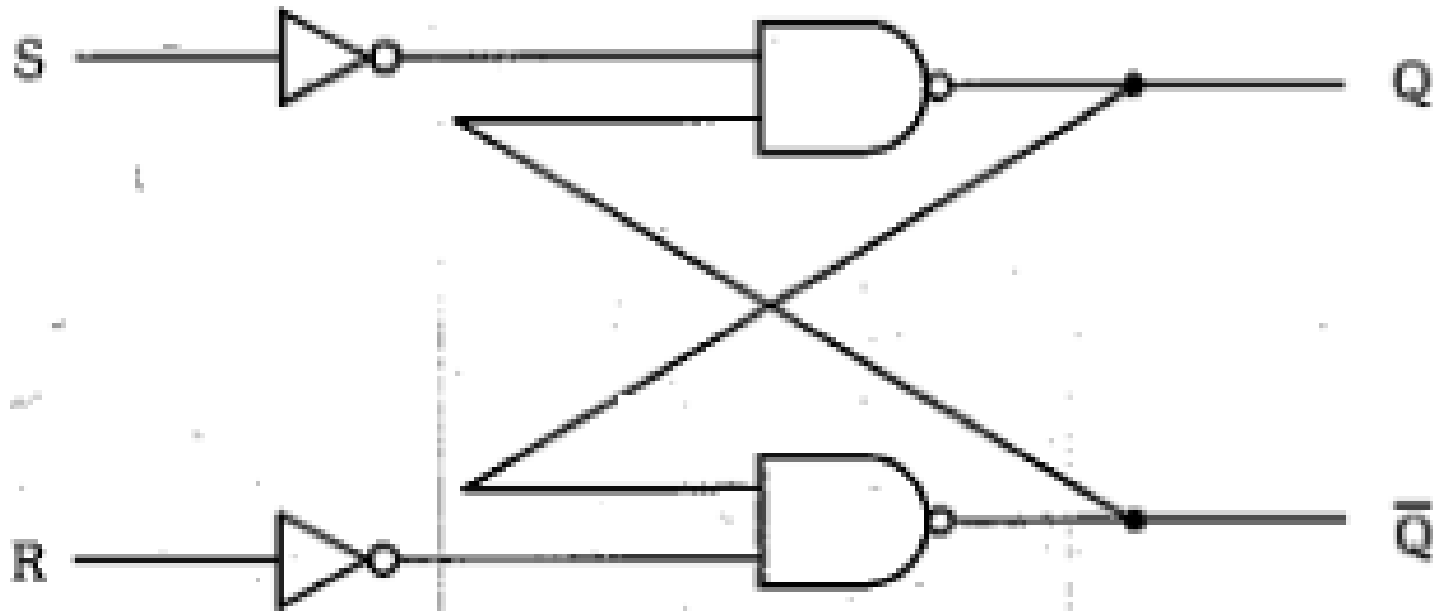
- Até aqui vimos Circuitos Combinacionais
 - Saída depende da combinação das entradas
- Agora veremos alguns Circuitos Sequenciais
 - Saída depende da combinação das entradas + estado anterior armazenado
 - Conseguir armazenar estado = **memória**



Flip-Flop

Como fazer um circuito lógico armazenar estados?

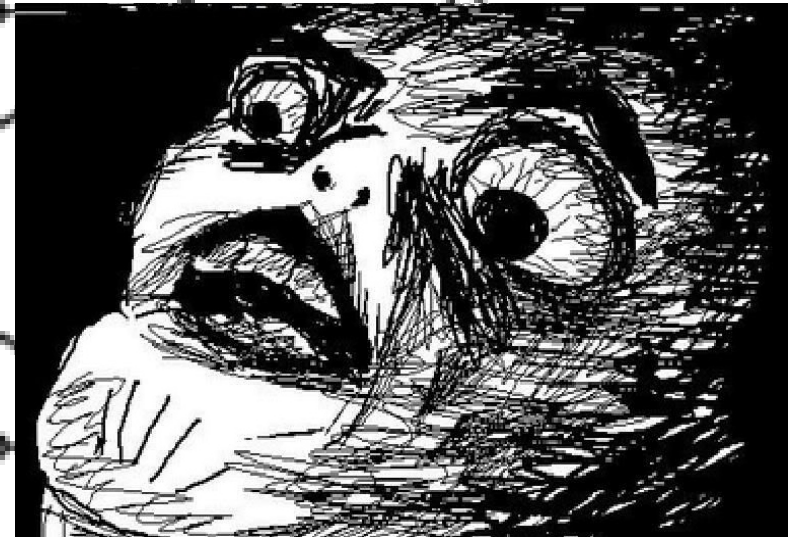
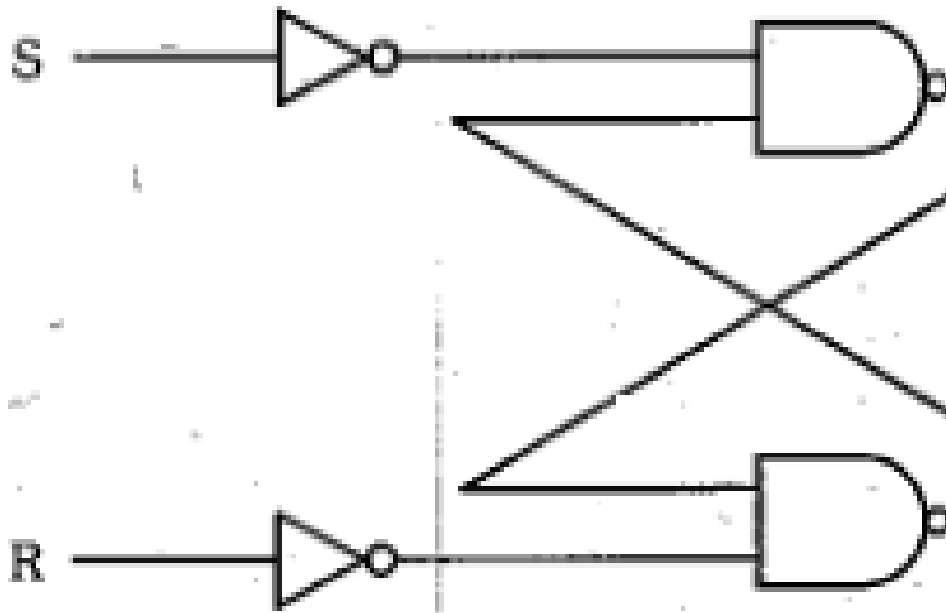
- Vamos analisar o circuito a seguir, chamado de **flip-flop** ou **latch**



Flip-Flop

Como fazer um circuito lógico armazenar estados?

- Vamos analisar o circuito a seguir, chamado de **flip-flop** ou **latch**

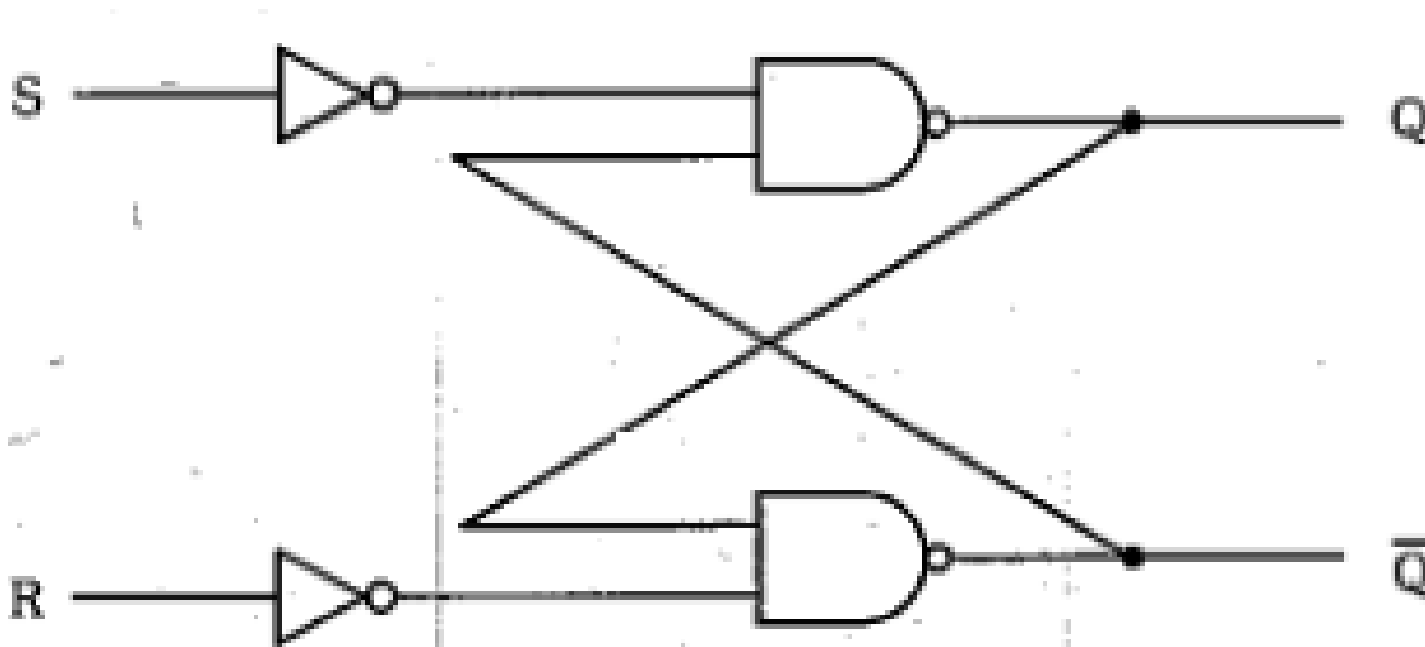


Que bruxaria é essa?

Flip-Flop

Como fazer um circuito lógico armazenar estados?

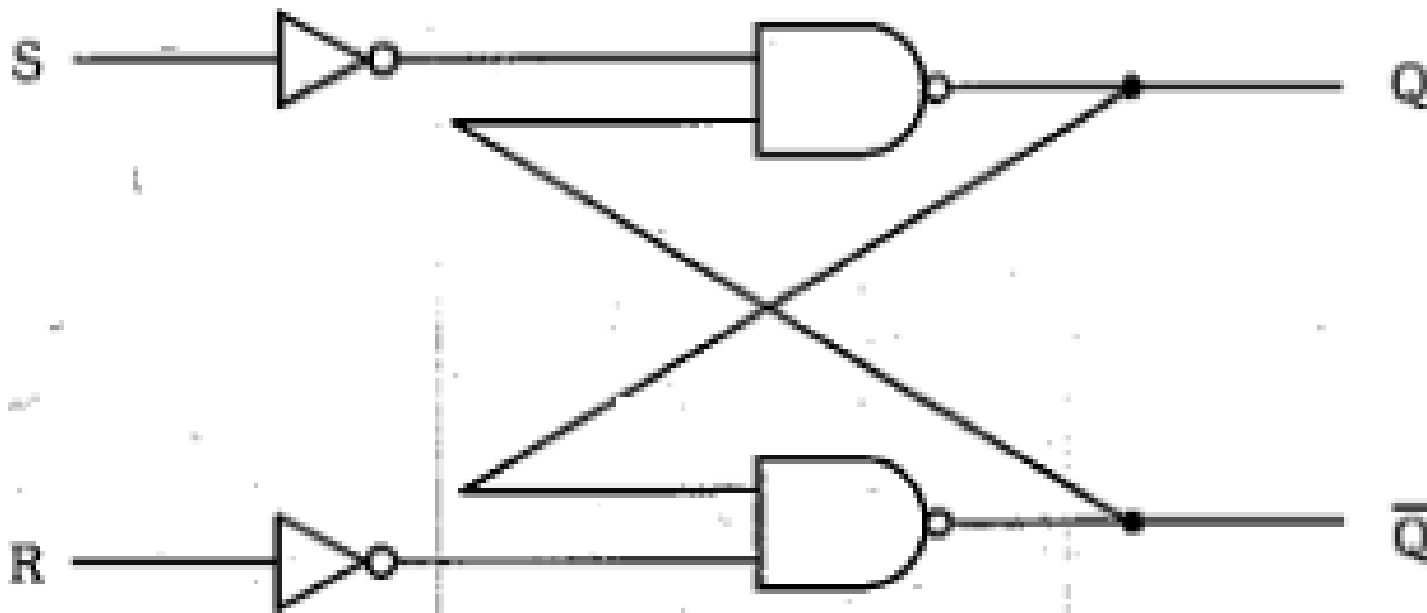
- Vamos analisar o circuito a seguir
- Para entendermos o circuito precisamos saber o valor anterior de Q, que chamaremos de Q_a



	S	R	Q_a	Q_f
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	

Flip-Flop

- Q futuro (Q_f) depende do Q anterior Q_a
- Como fica a tabela verdade de Q_f , dado S, R e Q_a ?



Flip-Flop

- Q futuro (Qf) depende do Q anterior Qa
- Como fica a tabela verdade de Qf, dado S, R e Qa?

S	R	Qa	Qf	$\bar{Q}f$	
0	0	0	0	1	} fixa Qf = Qa
0	0	1	1	0	
0	1	0	0	1	} fixa Qf em 0
0	1	1	0	1	
1	0	0	1	0	} fixa Qf em 1
1	0	1	1	0	
1	1	0	1	1	} não permitido
1	1	1	1	1	

S	R	Qf
0	0	Qa
0	1	0
1	0	1
1	1	X

Flip-Flop

- Q futuro (Q_f) depende do Q anterior Q_a
- Como fica a tabela verdade de Q_f , dado S, R e Q_a ?
- Ou seja, quando S e R são 0, Q_f não muda (igual a Q_a), quando R é 1, Q_f é 0 e quando S é 1, Q_f é 1.
Comportamento de memória!

S	R	Q_f
0	0	Q_a
0	1	0
1	0	1
1	1	


Flip-Flop

- Chamamos S e R de **Set** e **Reset**

S	R	Qf
0	0	Qa
0	1	0
1	0	1
1	1	

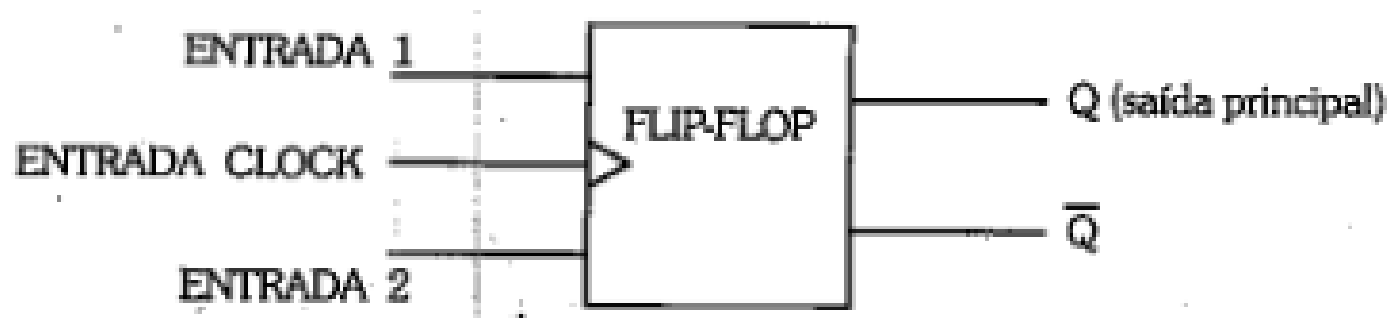
Flip-Flop

- Q futuro (Q_f) depende do Q anterior Q_a
- Como fica a tabela verdade de Q_f , dado S, R e Q_a ?
- Ou seja, quando S e R são 0, Q_f não muda (igual a Q_a), quando R é 1, Q_f é 0 e quando S é 1, Q_f é 1.
Comportamento de memória!

S	R	Q_f
0	0	Q_a
0	1	0
1	0	1
1	1	

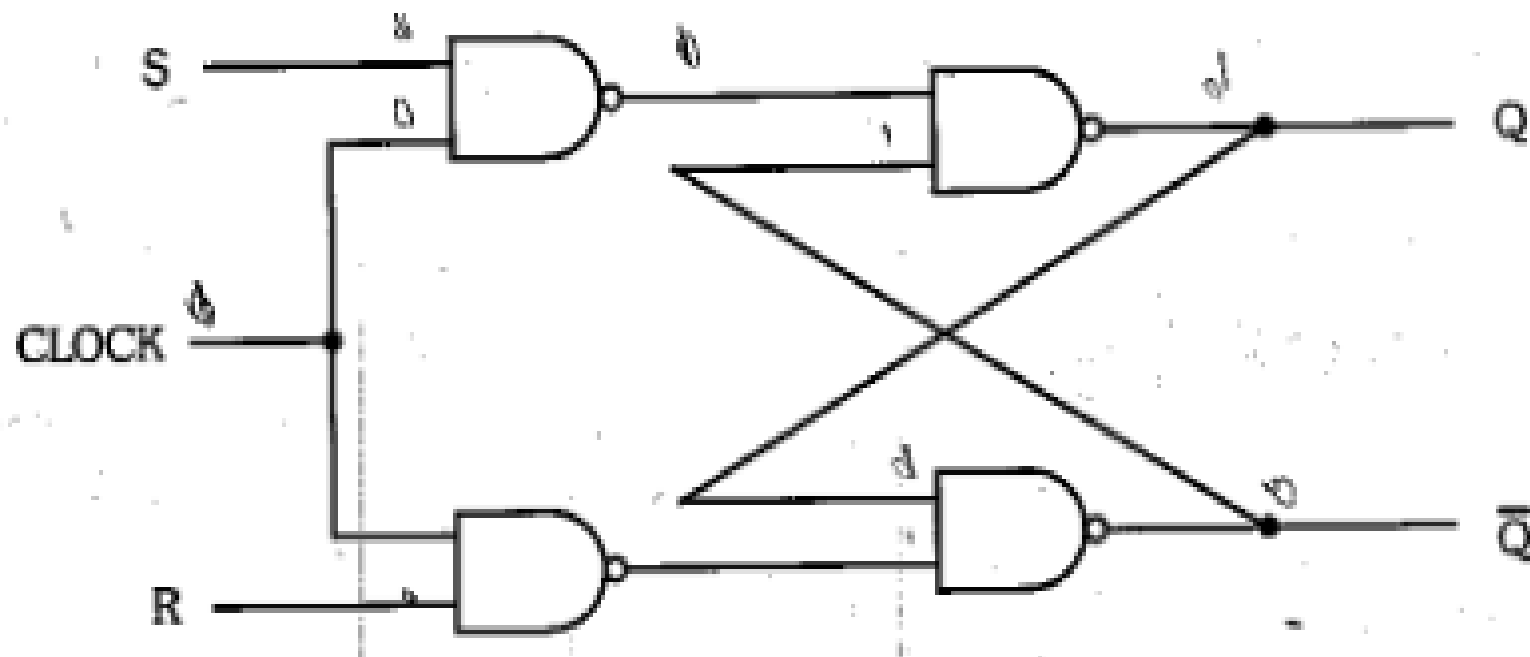
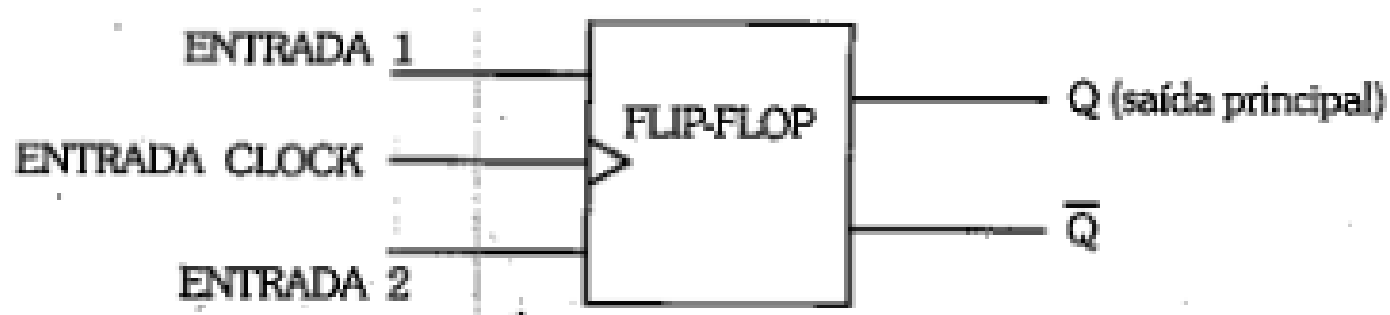
Flip-Flop

- E se quisermos controlar nosso **flip-flop** usando pulsos de **clock**?



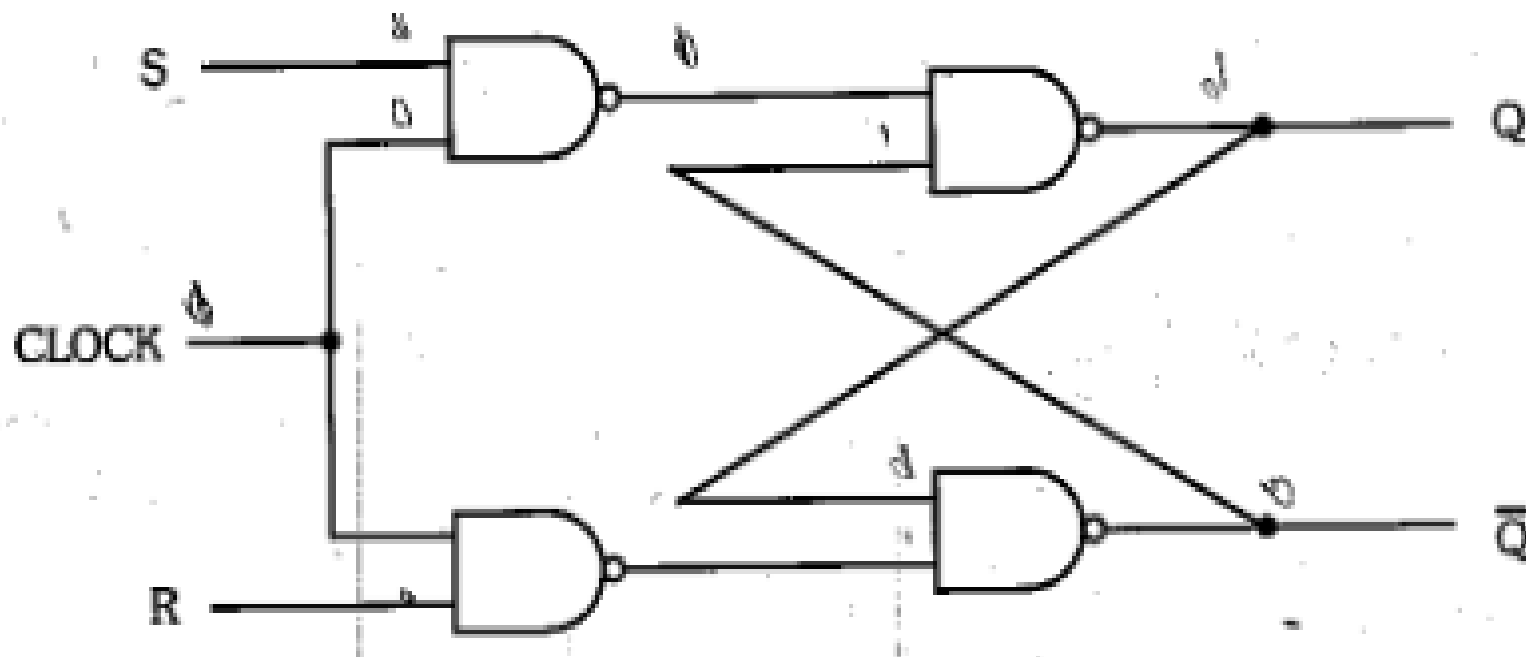
Flip-Flop

- E se quisermos controlar nosso **flip-flop** usando pulsos de **clock**?



Flip-Flop

- E se quisermos controlar nosso **flip-flop** usando pulsos de **clock**?
- Note que enquanto o Clock é 0, Q nunca muda. Só irá mudar quando ocorrer um pulso de clock.
- Também chamado de **flip-flop SÍNCRONO**



Flip-Flop

- O flip-flop síncrono funciona como um flip-flop normal quando o clock é 1, mas não muda quando o clock é 0
- Resumidamente:

CK	Qf
0	Qa
1	RS básico

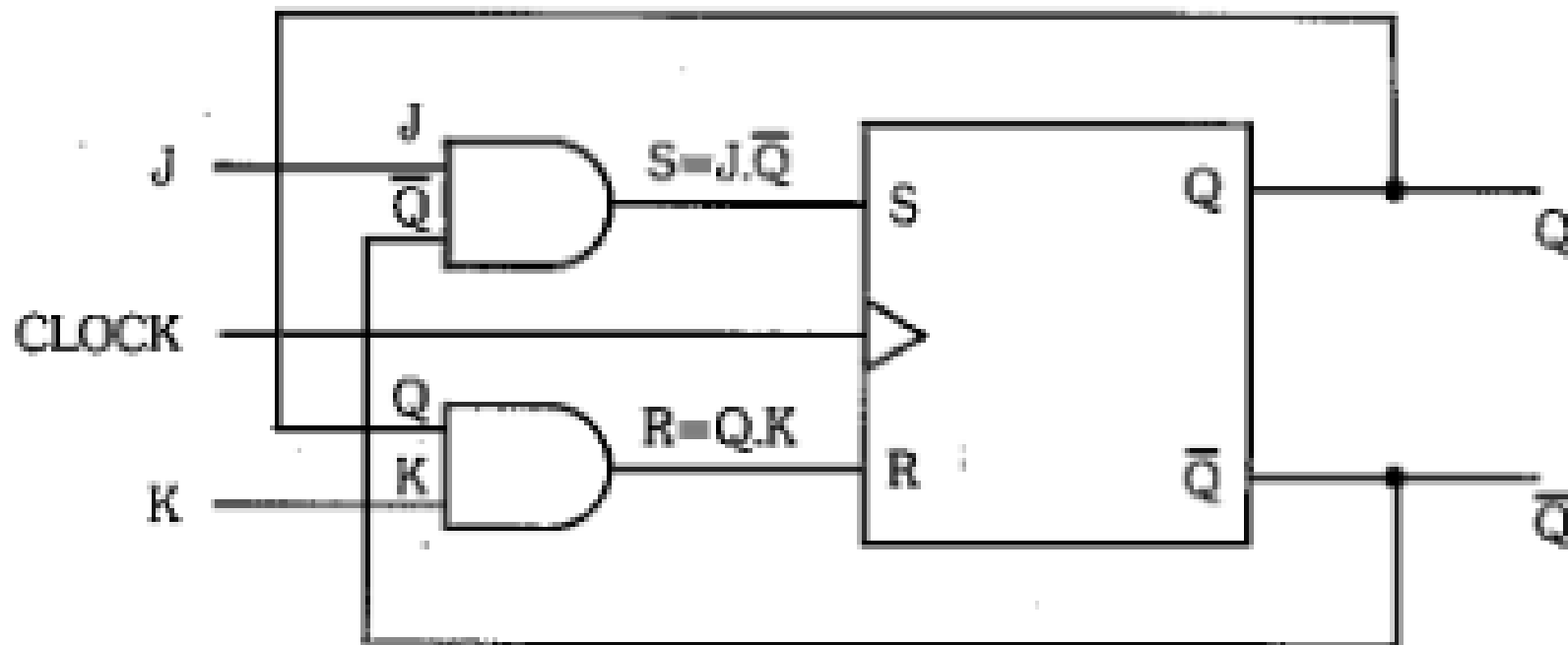
Flip-Flop

- Voltando à tabela-verdade do flip-flop RS comum, note que o caso $S=1$ e $R=1$ não é permitido pois ele acarretaria em $Q = \sim Q$
- Podemos fazer algo quanto a isso?

S	R	Qf
0	0	Qa
0	1	0
1	0	1
1	1	

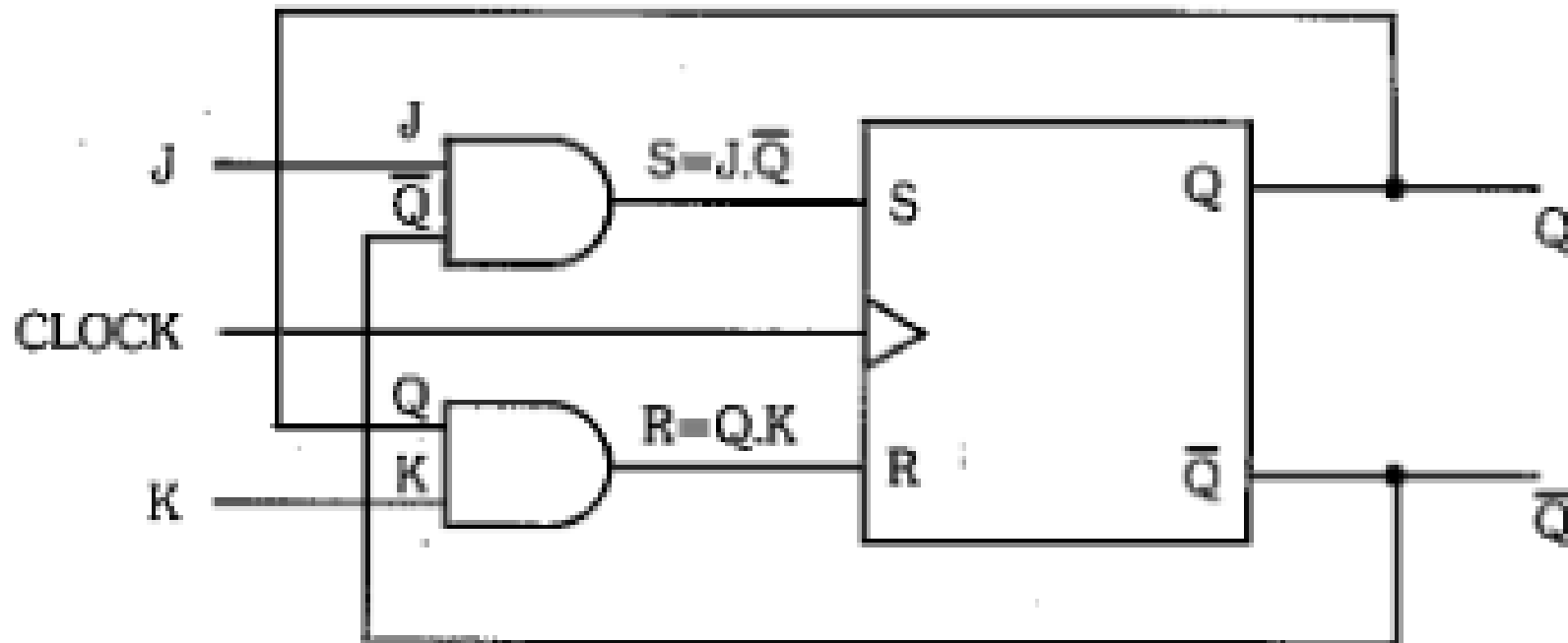
Flip-Flop JK

- Voltando à tabela-verdade do flip-flop RS comum, note que o caso $S=1$ e $R=1$ não é permitido pois ele acarretaria em $Q = \sim Q$
- Podemos fazer algo quanto a isso?
- Uma solução é o flip-flop JK



Flip-Flop JK

- Como fica a tabela verdade do JK?



Flip-Flop JK

- Como fica a tabela verdade do JK? (assumindo clock = 1)

J	K	Q_a	Q_a	S	R	Q_f	
0	0	0	1	0	0	Q_a	} Q_a
0	0	1	0	0	0	Q_a'	
0	1	0	1	0	0	$Q_a (Q_a = 0)$	} 0
0	1	1	0	0	1	0	
1	0	0	1	1	0	1	} 1
1	0	1	0	0	0	$Q_a(Q_a = 1)$	
1	1	0	1	1	0	$\bar{Q}_a(Q_a = 0)$	} \bar{Q}_a
1	1	1	0	0	1	$\bar{Q}_a(Q_a = 1)$	

Flip-Flop JK

- Basicamente, agora a combinação $J=1$ e $K=1$ faz o valor de Q inverter ($Q_f = \sim Q_a$)

J	K	Q_f
0	0	Q_a
0	1	0
1	0	1
1	1	$\overline{Q_a}$

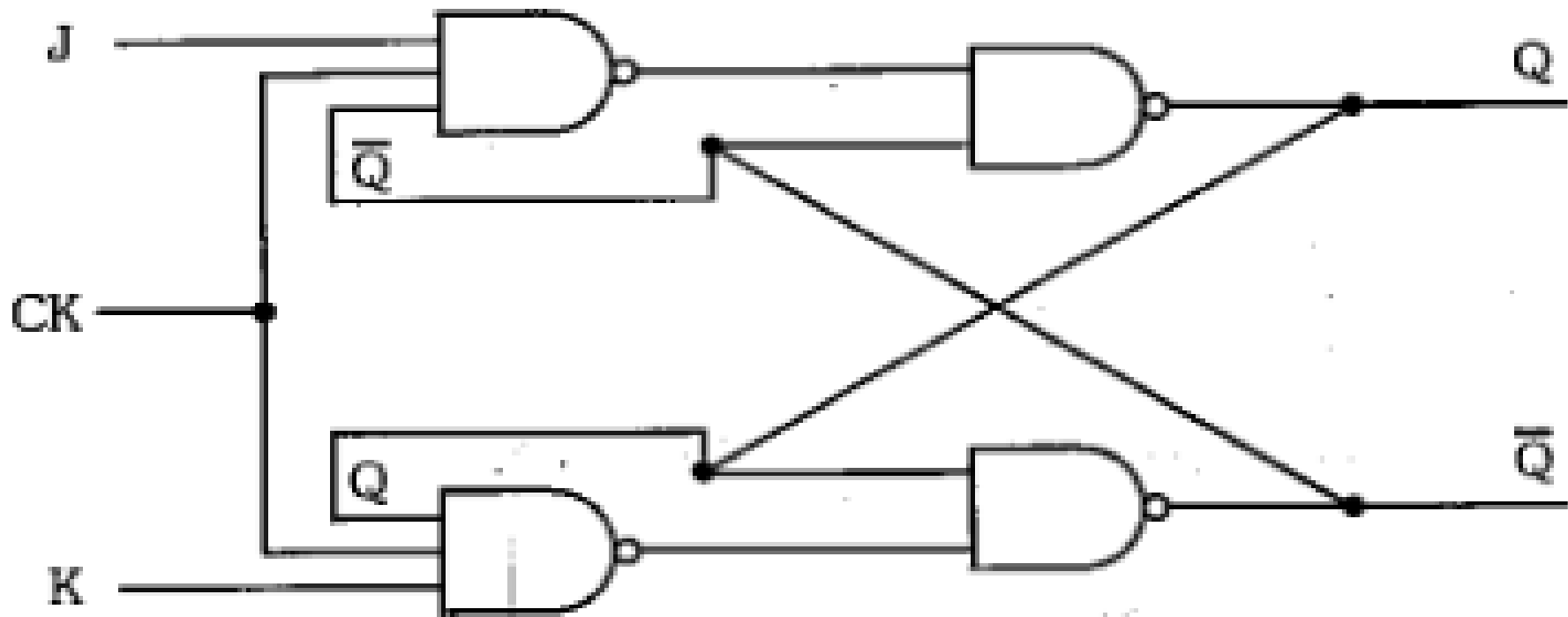
Flip-Flop JK

- Note que se J e K permanecerem ambos em 1 o flip-flop começará a oscilar (o valor vai trocar indefinidamente) então é importante controlar esse comportamento com o *timing* do clock.

J	K	Qf
0	0	Qa
0	1	0
1	0	1
1	1	\overline{Qa}

Flip-Flop JK

- Desenhando o JK completo



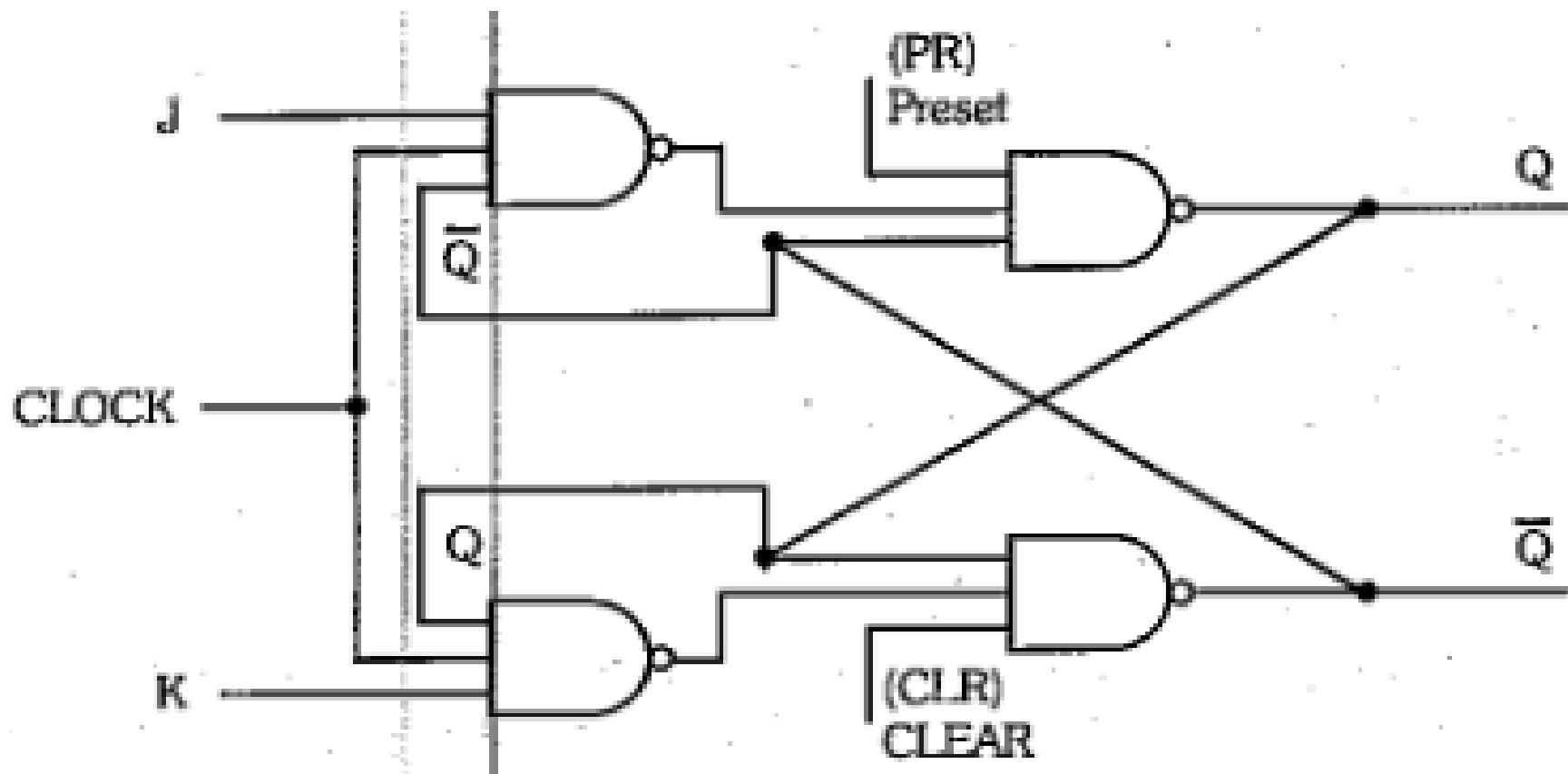
Flip-Flop JK

- Note que se J e K permanecerem ambos em 1 o flip-flop começará a oscilar (o valor vai trocar indefinidamente) então é importante controlar esse comportamento com o *timing* do clock.

J	K	Qf
0	0	Qa
0	1	0
1	0	1
1	1	\overline{Qa}

Flip-Flop JK Preset e Clear

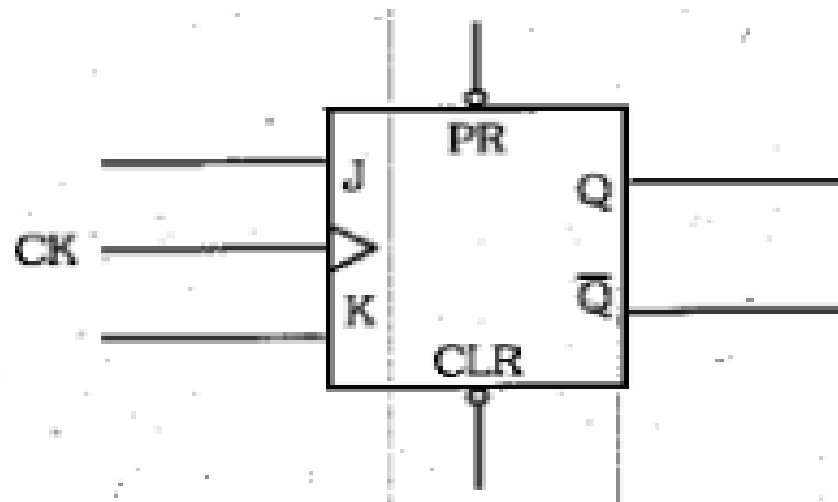
- Alguns flip-flop usam entradas **Preset** e **Clear** para setar e limpar o valor, respectivamente
- Vejam esse JK com Preset e Clear:



Flip-Flop JK Preset e Clear

- Como ele funciona?

CLR	PR	Qf
0	0	não permitido
0	1	0
1	0	1
1	1	funcionamento normal

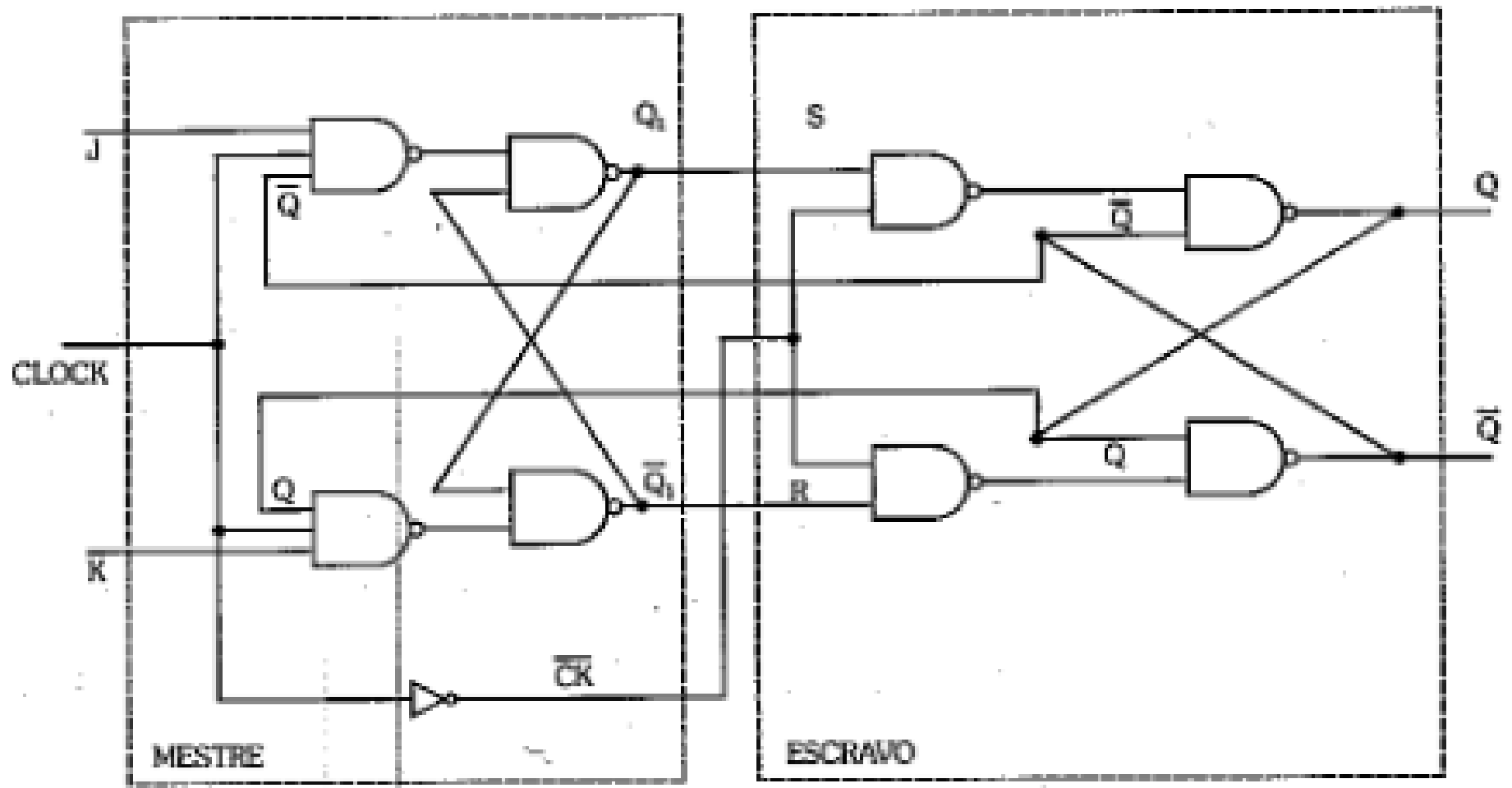


Flip-Flop JK Mestre-Escravo

- O flip-flop JK tem uma característica indesejável:
 - Quando o clock está em 0 o circuito não muda. Mas quando está 1, o circuito pode mudar indefinidamente
- Lembrem que queremos sincronizar tudo com o clock
- Como fazer para permitir apenas UMA mudança do flip-flop por pulso de clock?
 - Flip-flop **Mestre-Escravo**, ou **Master-Slave**

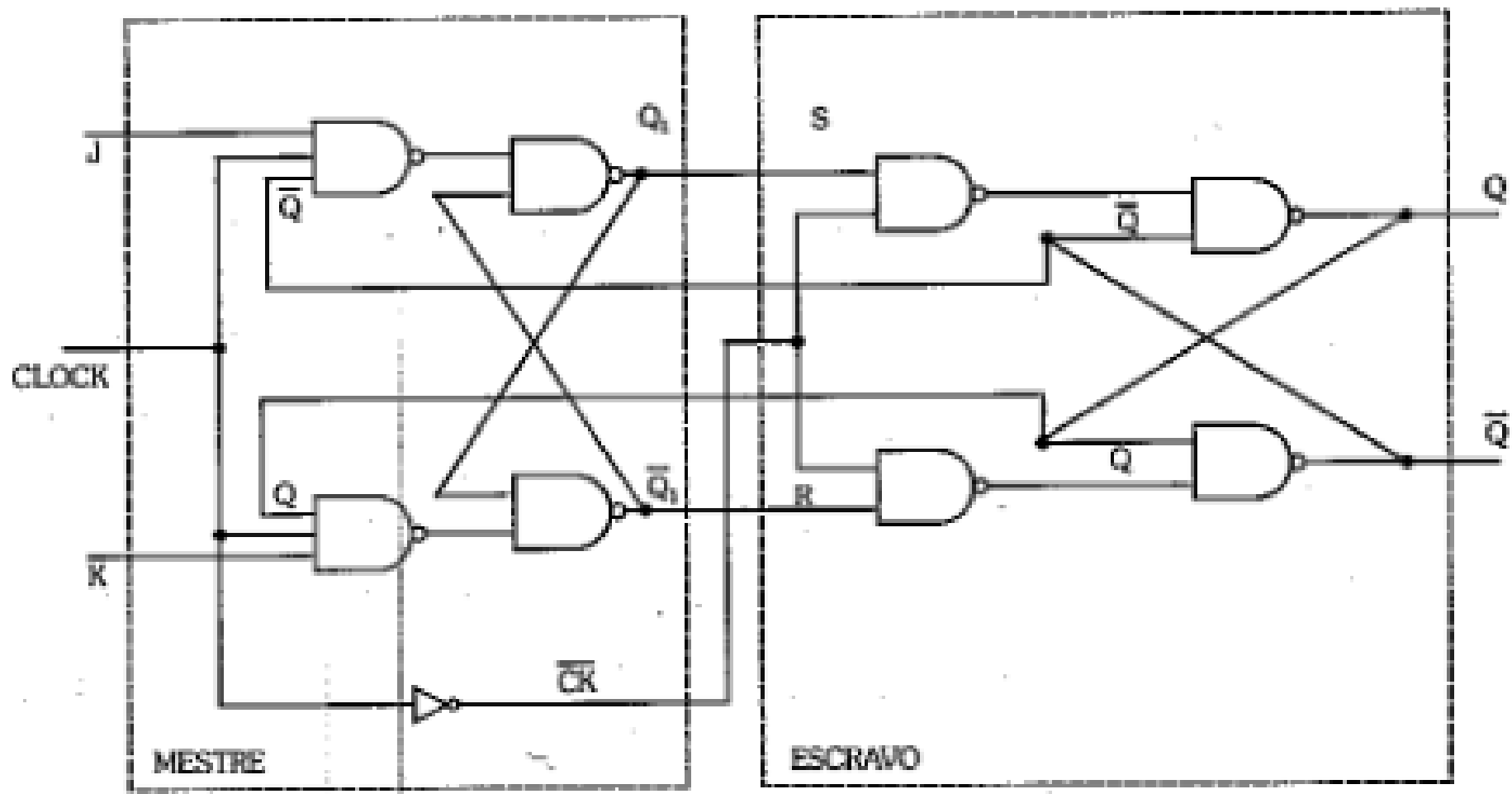
Flip-Flop JK Mestre-Escravo

- Note como ele **não permite** a passagem do valor pelos 2 circuitos de uma vez, por causa do **CLOCK negado** na parte escravo



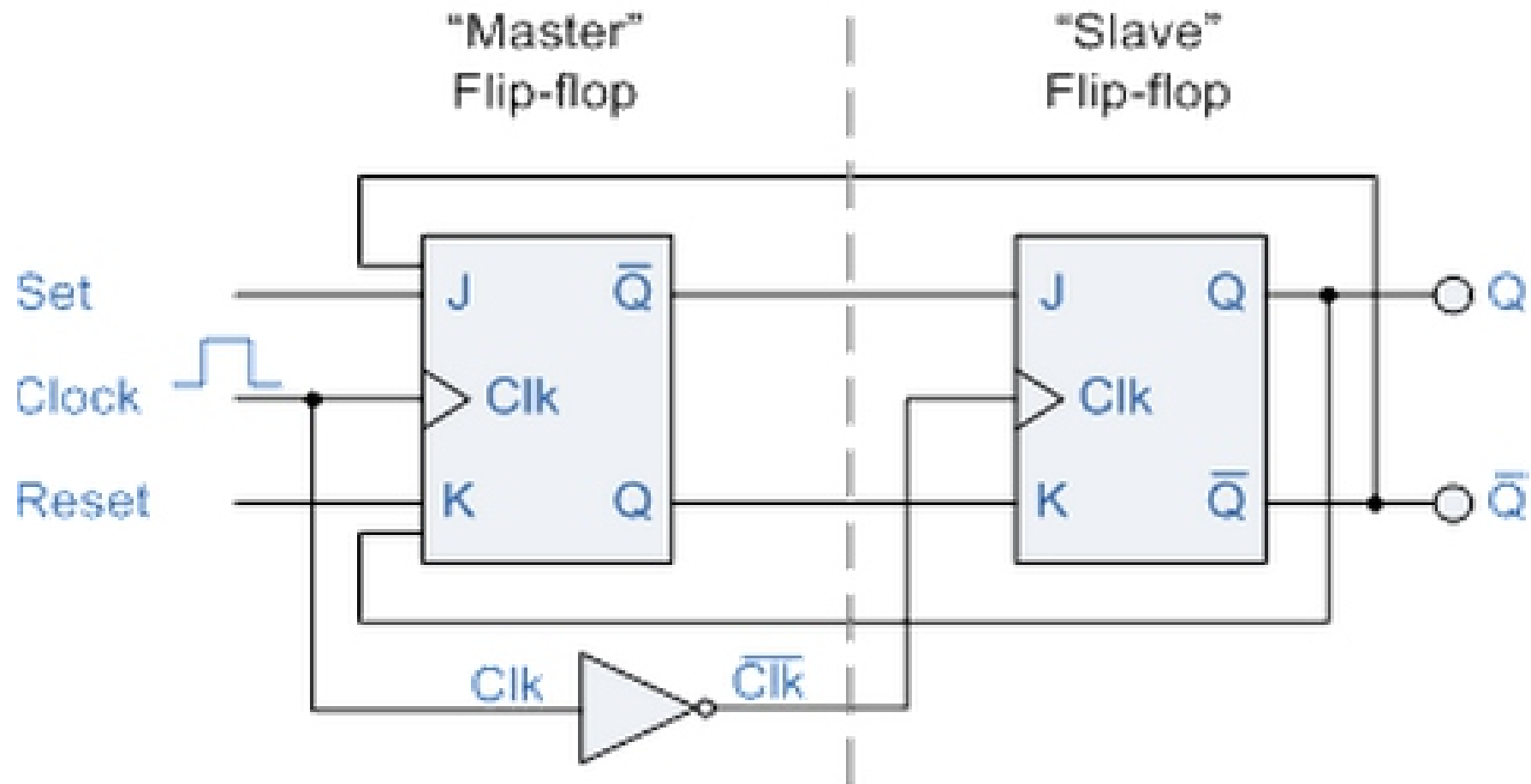
Flip-Flop JK Mestre-Escravo

- Basicamente, ele só permite a passagem de J e K quando o clock troca de 1 pra 0 (sensível à **descida de clock**)



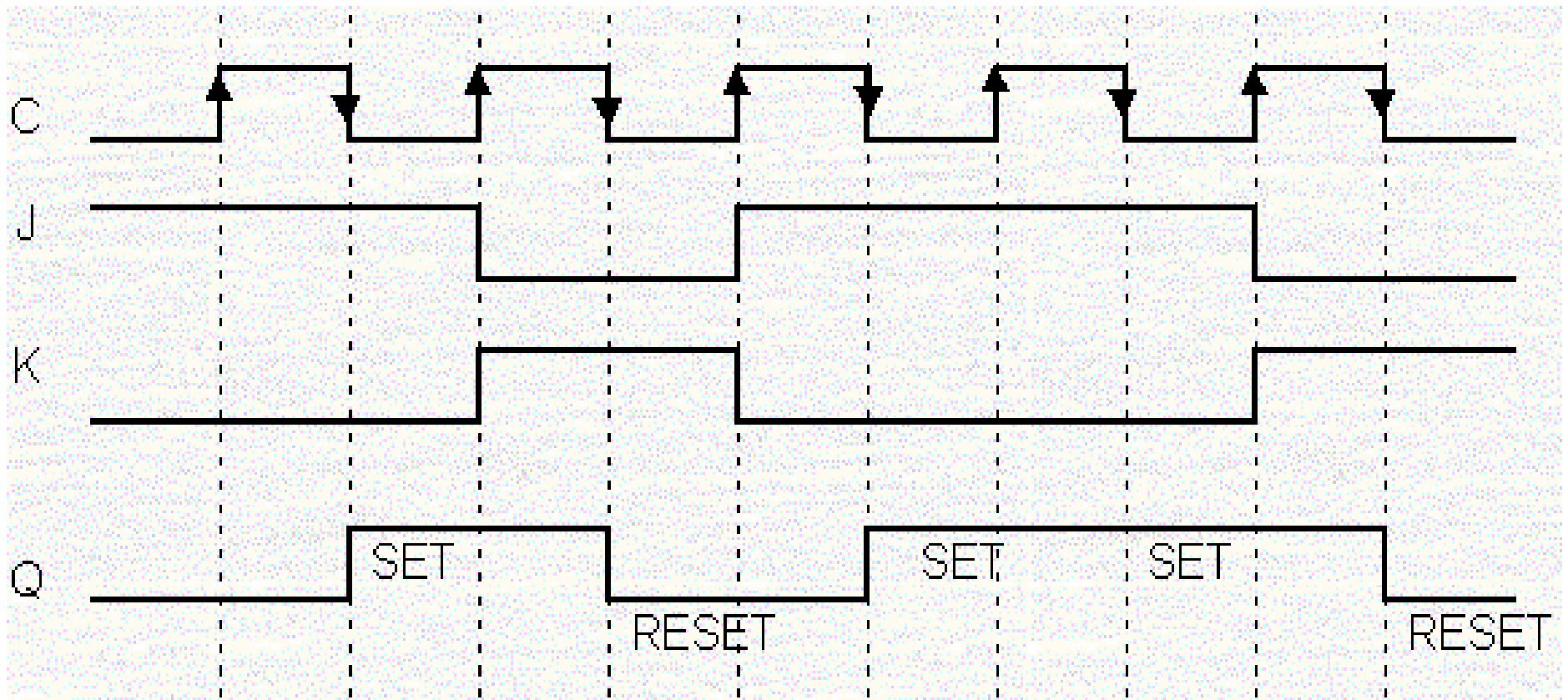
Flip-Flop JK Mestre-Escravo

- Simplificando com blocos de circuito



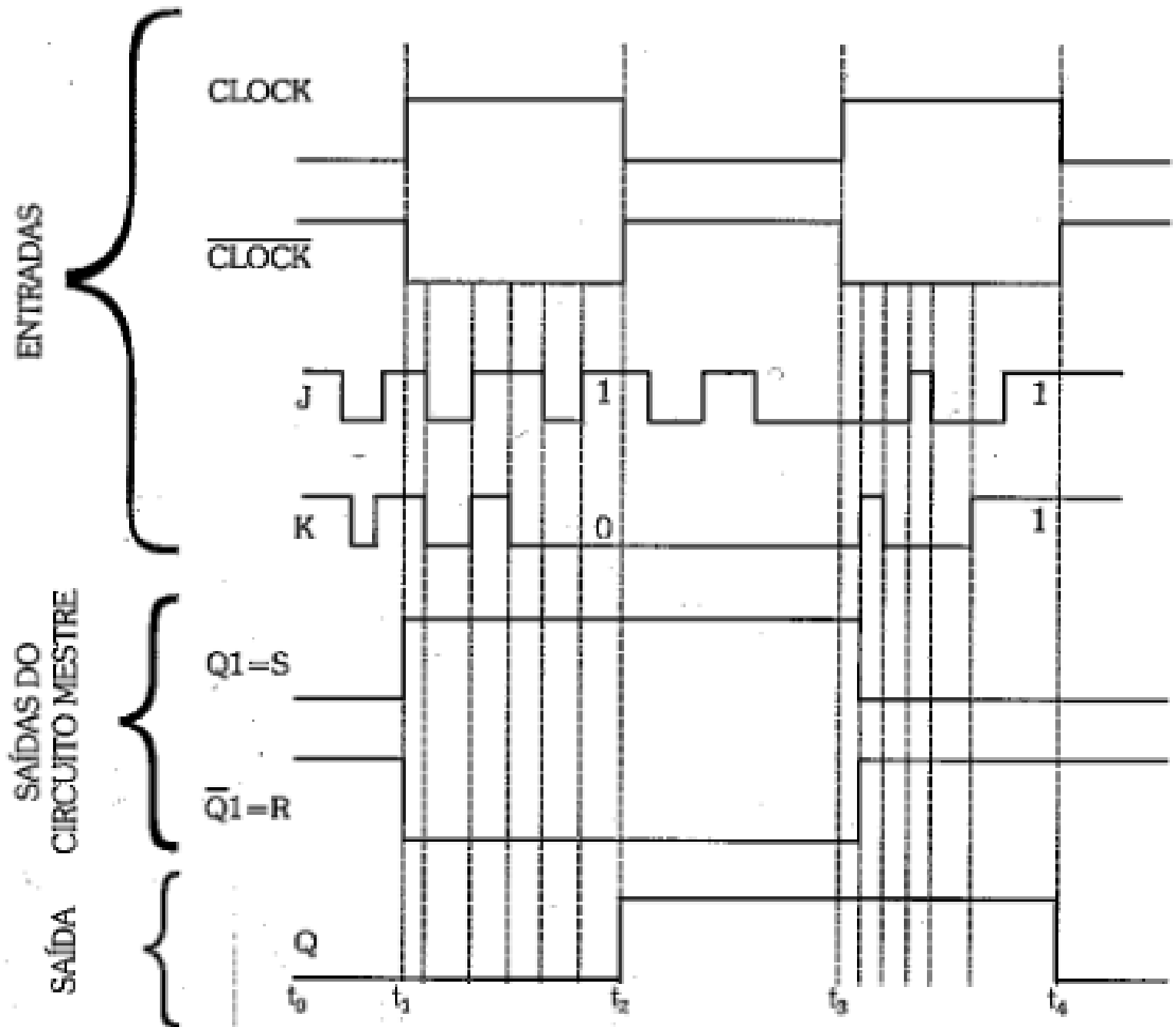
Flip-Flop JK Mestre-Escravo

- Vamos analisar uma sequência de entradas:



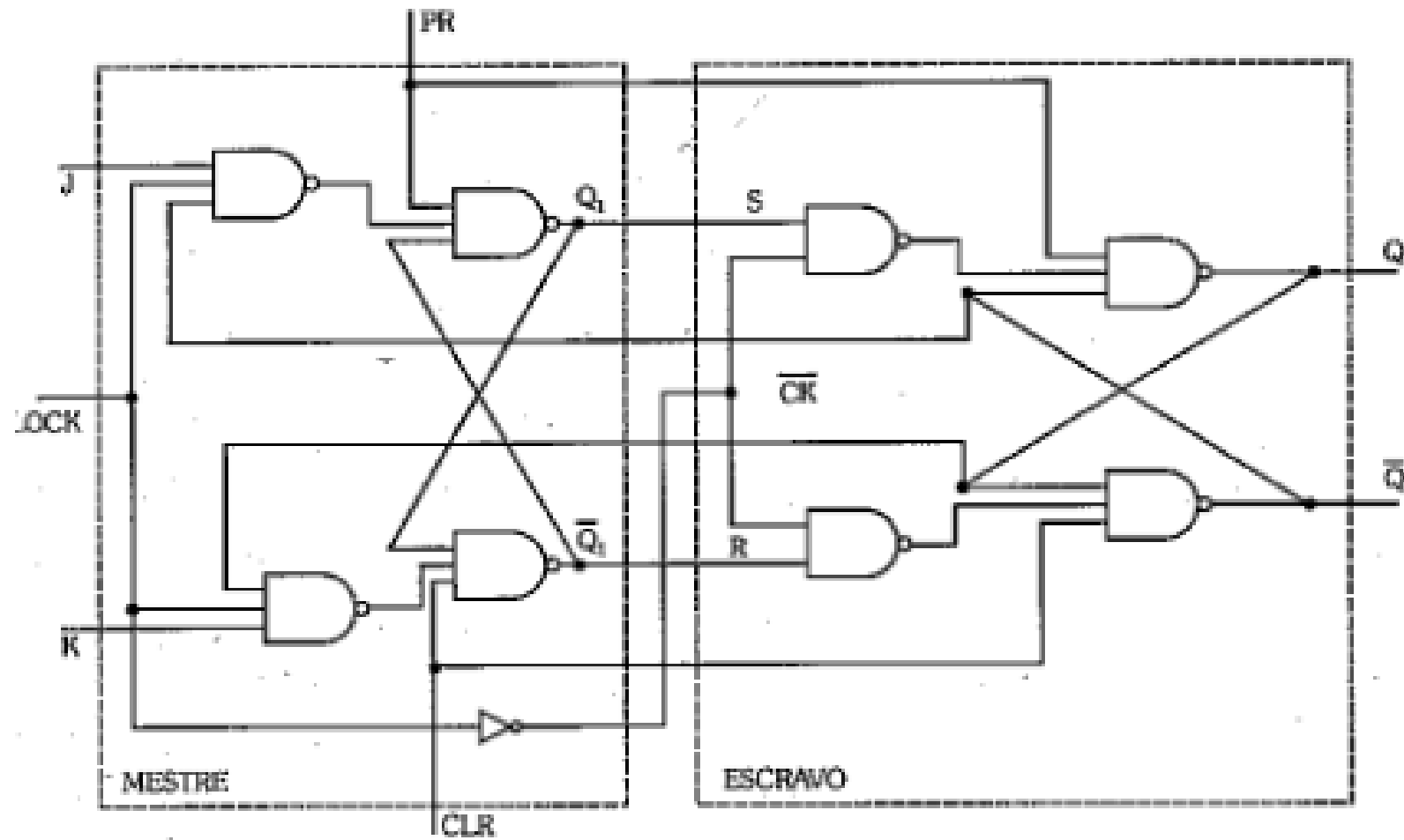
Flip-Flop JK Mestre-Escravo

- Mais uma:



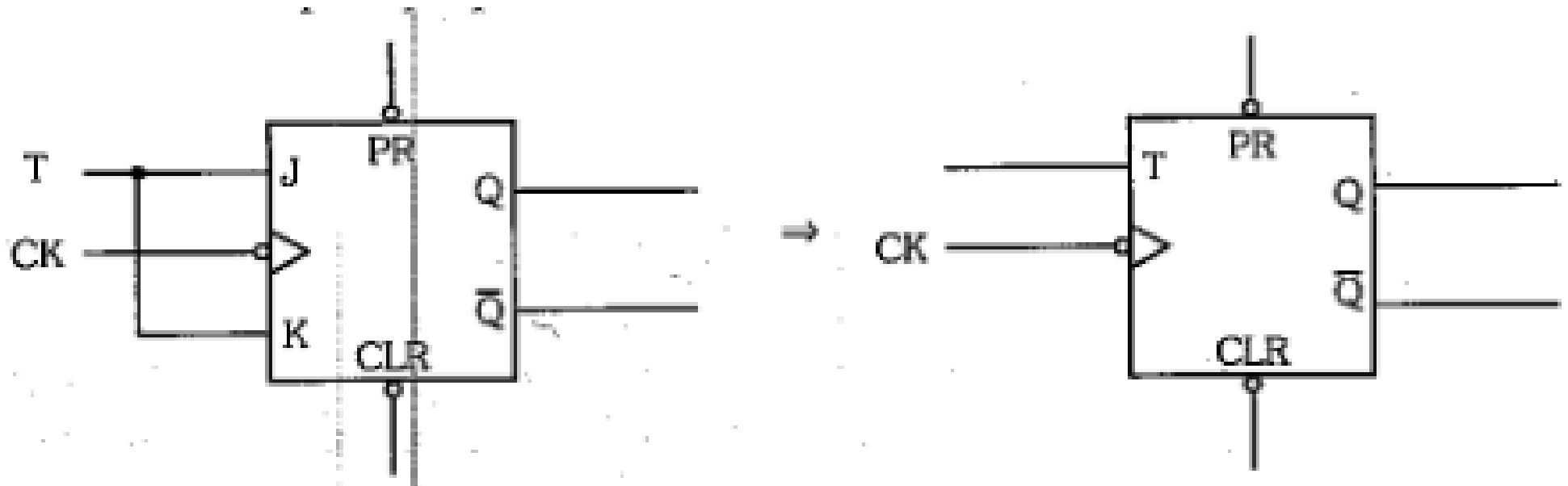
Flip-Flop JK Mestre-Escravo

- Podemos fazer um flip-flop JK Mestre-Escravo com Preset e Clear
- Note que PR e CLR estão ligados nos 2 flip-flops, e portanto são independentes do clock



Flip-Flop T

- O flip-flop T é feito a partir de um JK com as entradas curto-circuitadas (conectadas no mesmo lugar)



Flip-Flop T

- Nesse caso, é impossível acontecer $J=1$ e $K=0$, ou $J=0$ e $K=1$, é claro
- A tabela-verdade fica então assim:

J	K	T	Qf
0	0	0	Q_a
0	1	não existe	/
1	0	não existe	/
1	1	1	$\overline{Q_a}$

T	Qf
0	Q_a
1	$\overline{Q_a}$

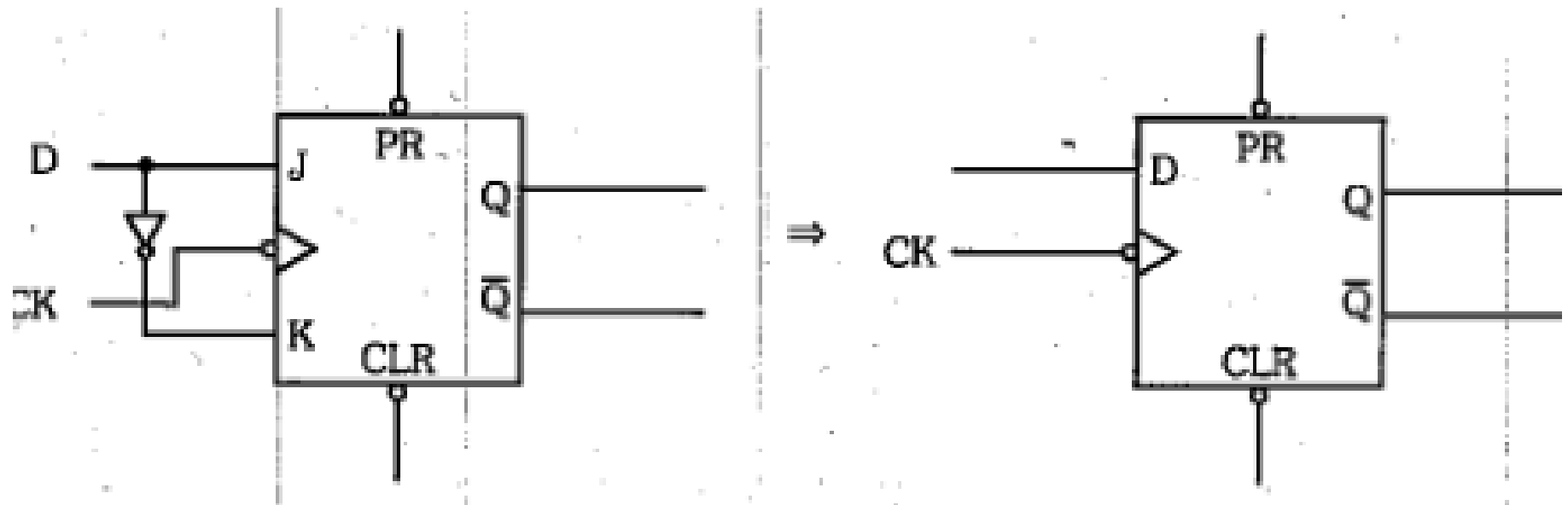
Flip-Flop T

- Ou seja, se $T=0$, flip-flop mantém estado anterior. Se $T=1$, flip-flop troca.
- O T vem de da palavra **toggle** (comutar, trocar)
- Veremos mais adiante que o T é a célula básica do contador assíncrono

T	Qf
0	Q_a
1	$\overline{Q_a}$

Flip-Flop D

- Um caso parecido ao T é se curto-circuitarmos as entradas J e K, mas colocando um inversor no K
- Chamamos isso de flip-flop D



Flip-Flop D

Nesse caso, é impossível acontecer $J=0$ e $K=0$, ou $J=1$ e $K=1$. J e K sempre serão opostos.

A tabela-verdade fica então assim:

J	K	D	Qf
0	0	não existe	/
0	1	0	0
1	0	1	1
1	1	não existe	/

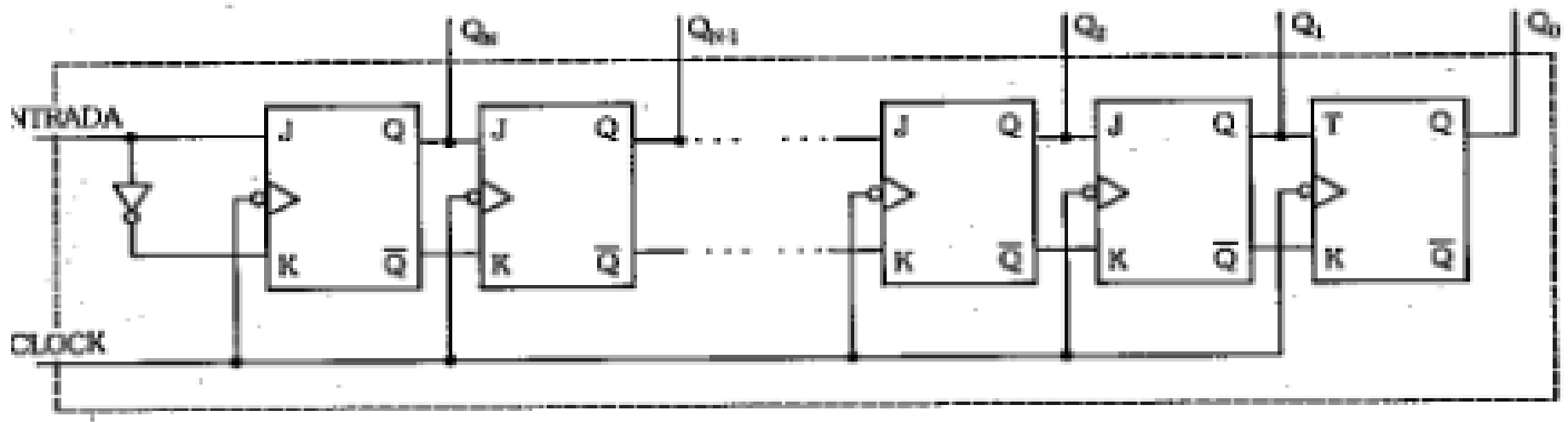
Flip-Flop D

- Ou seja, se $D=1$, flip-flop armazena valor 1. Se $D=0$, flip-flop armazena valor 0.
- O D vem de da palavra **data** (dado)
- O flip-flop D é a célula básica de **registradores de deslocamento** e diversas memórias

D	Qf
0	0
1	1

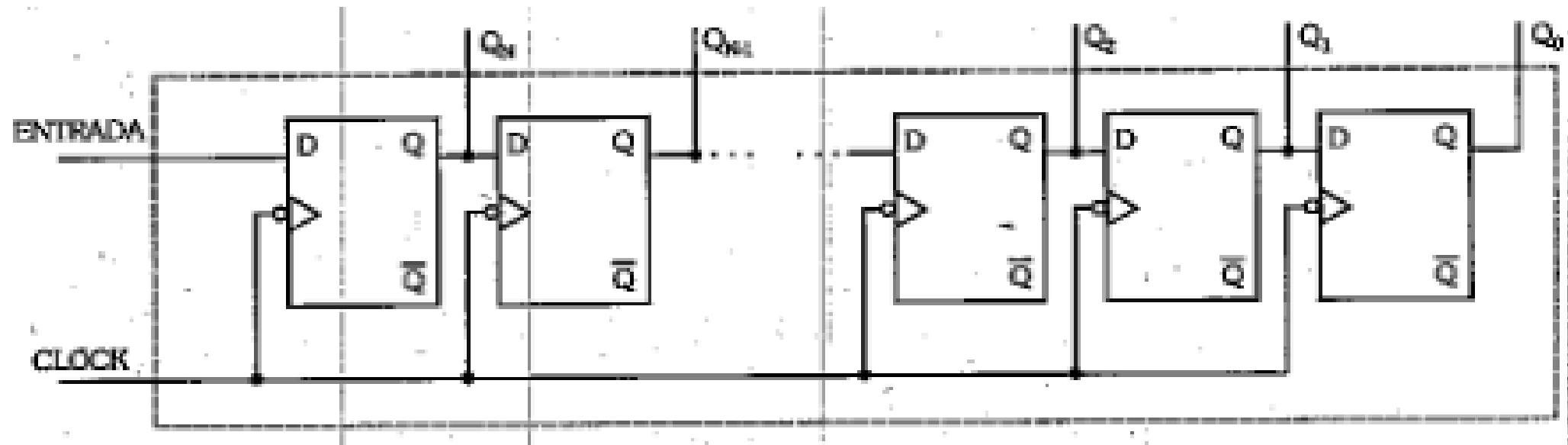
Registadores de Deslocamento

- Flip-flops só nos deixam armazenar um bit por vez. Se quisermos armazenar mais bits, precisamos combinar vários flip-flops.
- Um exemplo disso é o registrador de deslocamento, feito com flip-flops D (onde K é inverso de J)



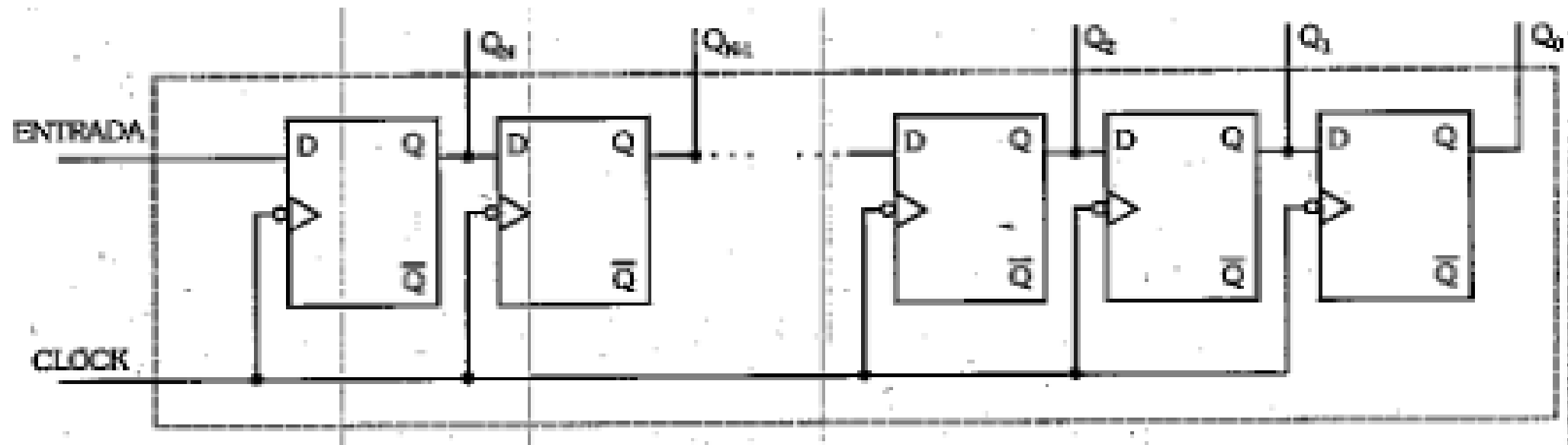
Registadores de Deslocamento

- Notação com flip-flops D:



Registradores de Deslocamento

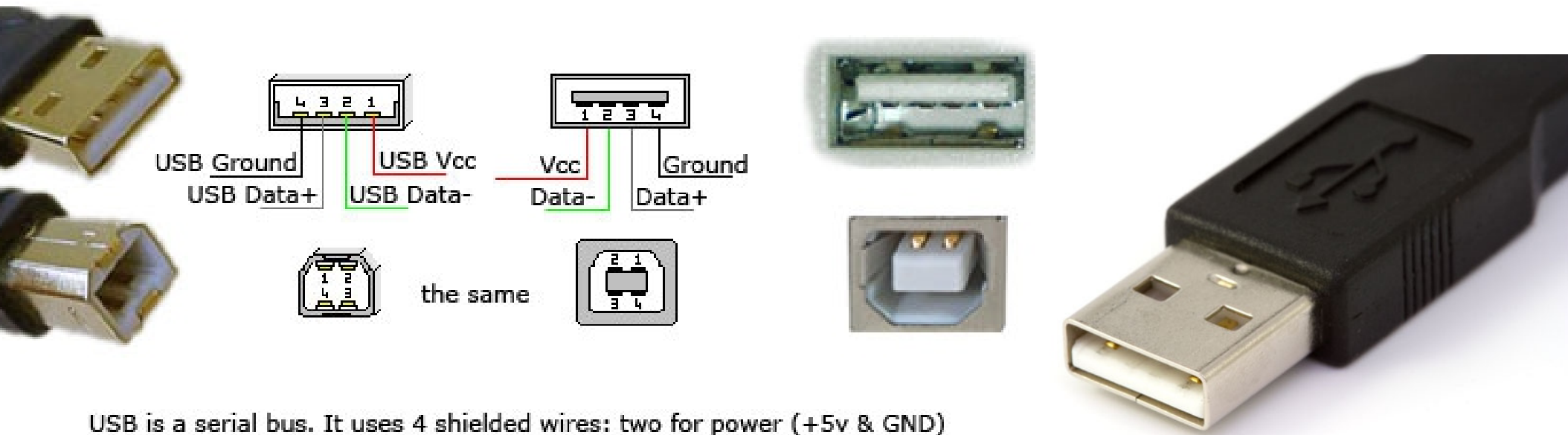
- Como funciona? Pense no registrador de deslocamento como um mestre-escravo encadeado



Registradores de Deslocamento

- O registrador de deslocamento funciona como um conversor **série-paralelo**
- **Serial:** um bit de cada vez, sequencialmente. Bom pra longas distâncias (menor custo do cabo). Ex: USB

USB pinout

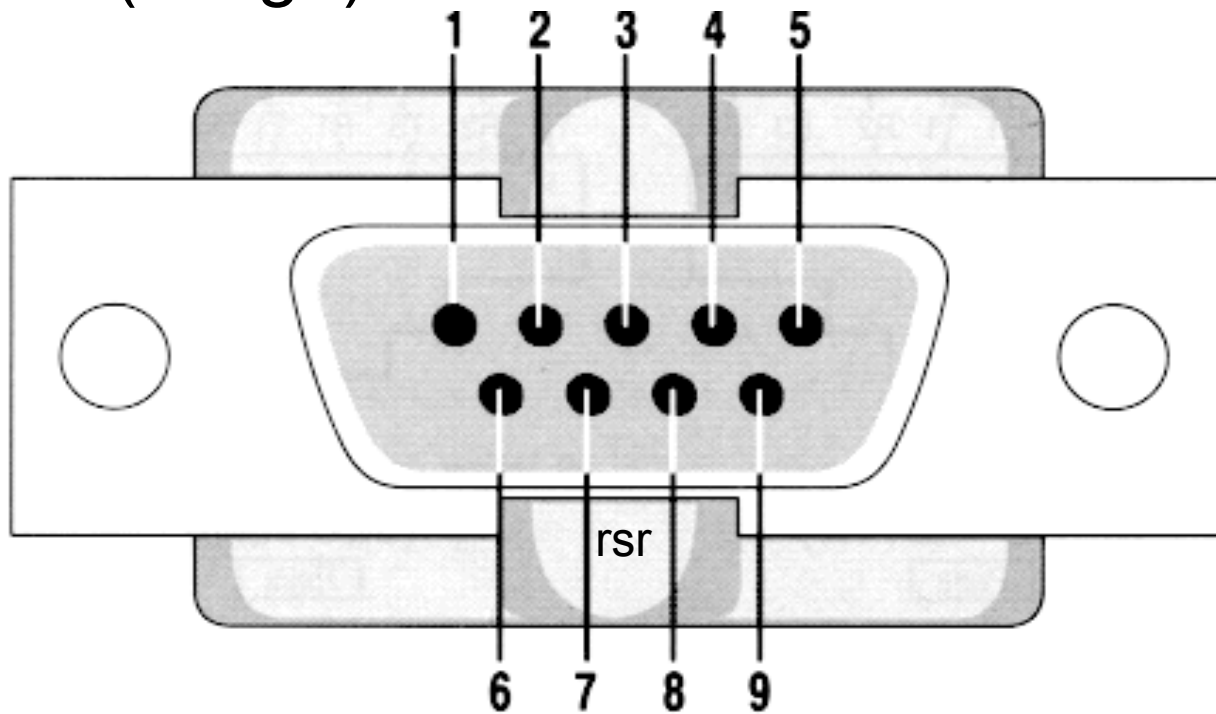


USB is a serial bus. It uses 4 shielded wires: two for power (+5v & GND) and two for differential data signals (labelled as D+ and D- in pinout)

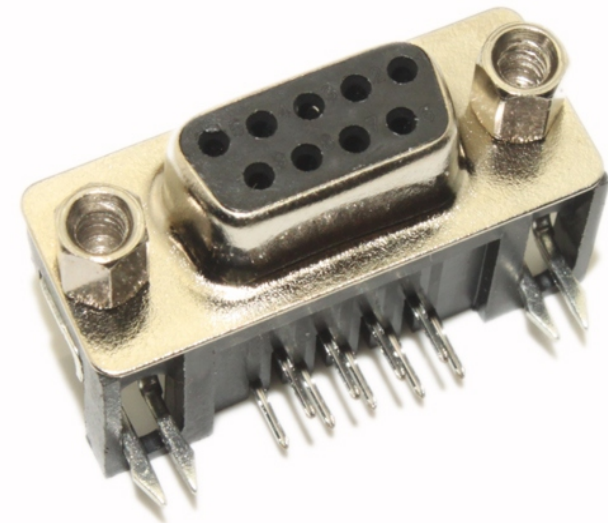
http://pinouts.ru/Slots/USB_pinout.shtml

Registradores de Deslocamento

- **Serial:** um bit de cada vez, sequencialmente. Bom pra longas distâncias (menor custo do cabo). Ex: RS-232 (antigo)

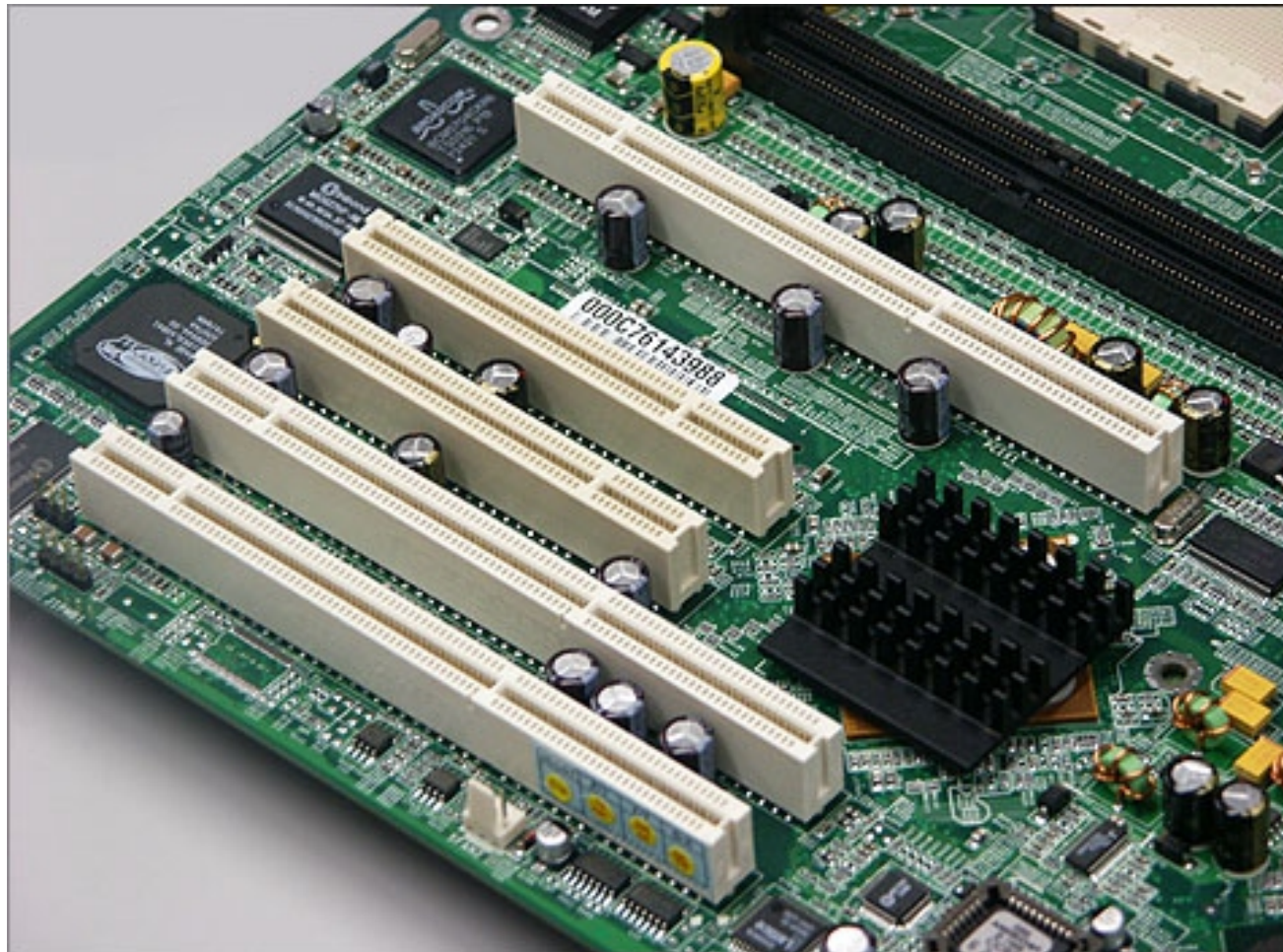


Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		



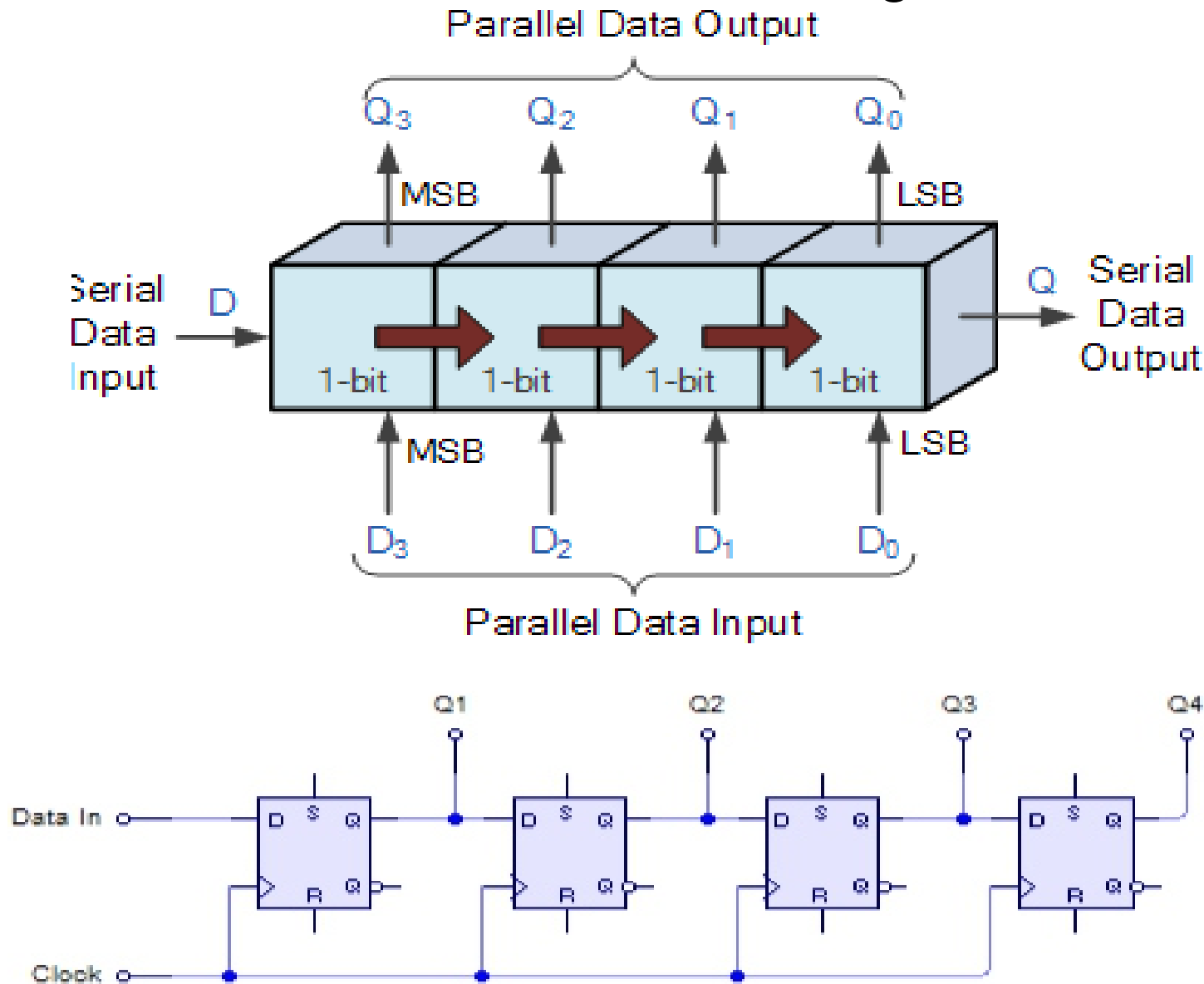
Registradores de Deslocamento

- **Paralelo:** comunica um barramento de bits simultaneamente (em paralelo). A “largura do barramento” importa. Ex: PCI (largura 32 ou 64 bits)
- Pinagem: http://en.wikipedia.org/wiki/Conventional_PCI



Registradores de Deslocamento

- Resitrador de deslocamento, em inglês: **shift register**



Registradores de Deslocamento

- Uma tabela resumindo a situação:

Informação	descidas de clock	Q_2	Q_1	Q_0	Q_{-1}
$I_0 = 0$	1ª	0	0	0	0
$I_1 = 1$	2ª	1	0	0	0
$I_2 = 0$	3ª	0	1	0	0
$I_3 = 1$	4ª	1	0	1	0

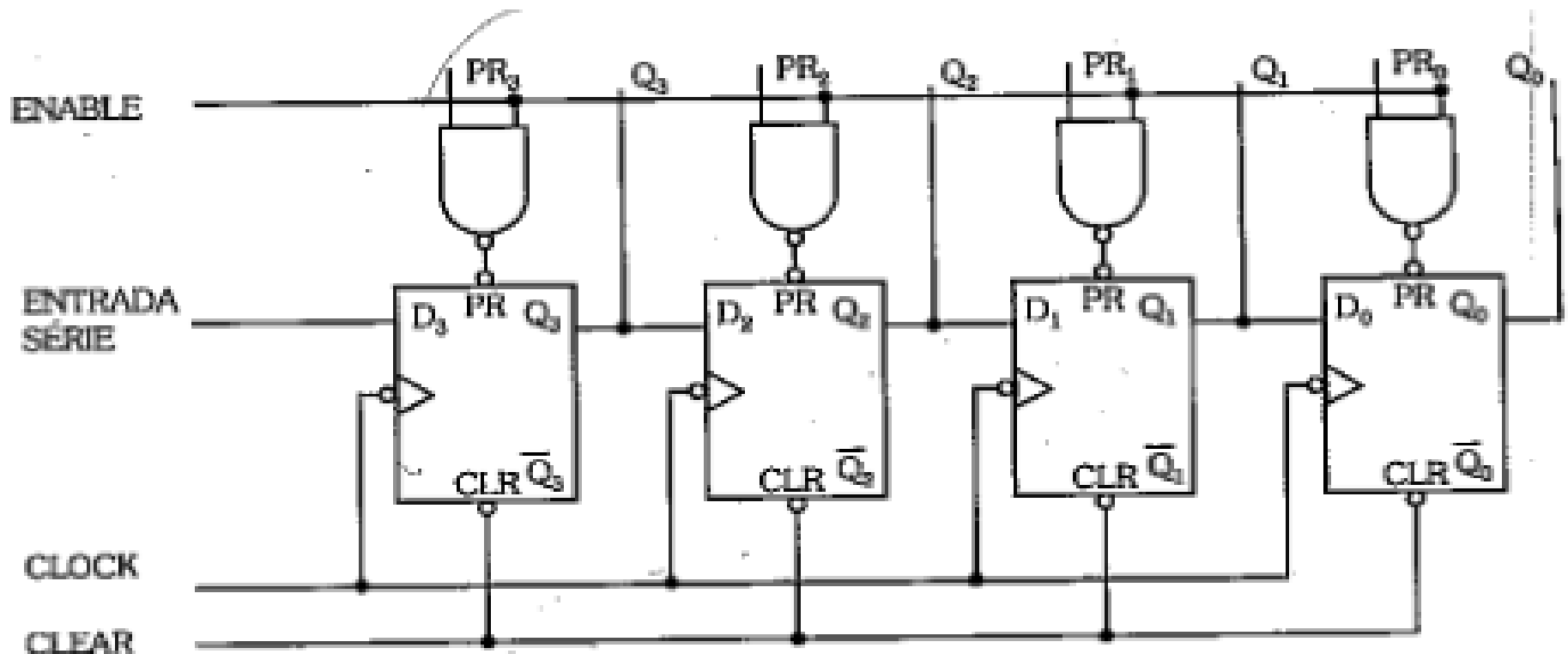
Registradores de Deslocamento

- E como faríamos um conversor paralelo-série?



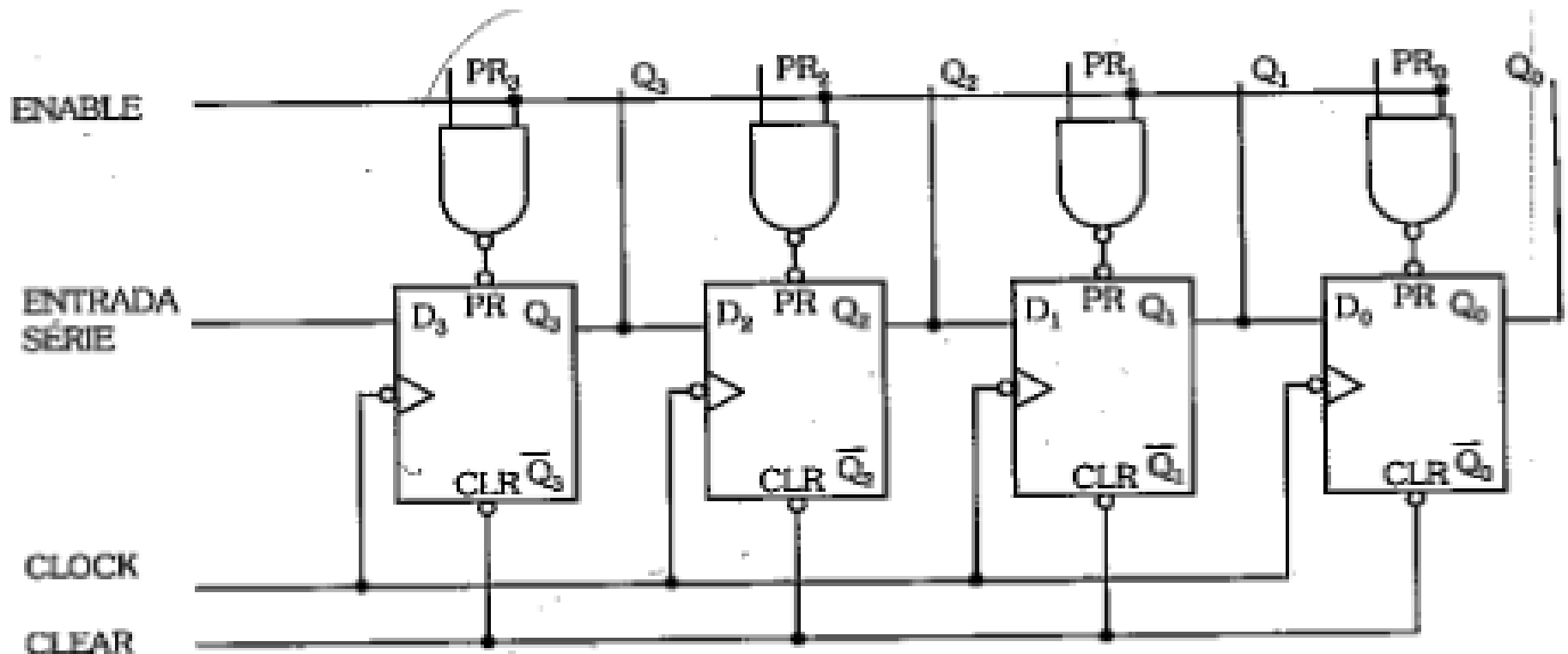
Registradores de Deslocamento

- E como faríamos um conversor paralelo-série?
 - Precisamos de registrador com Preset e Clear, pois é através deles que entraremos com a informação paralela



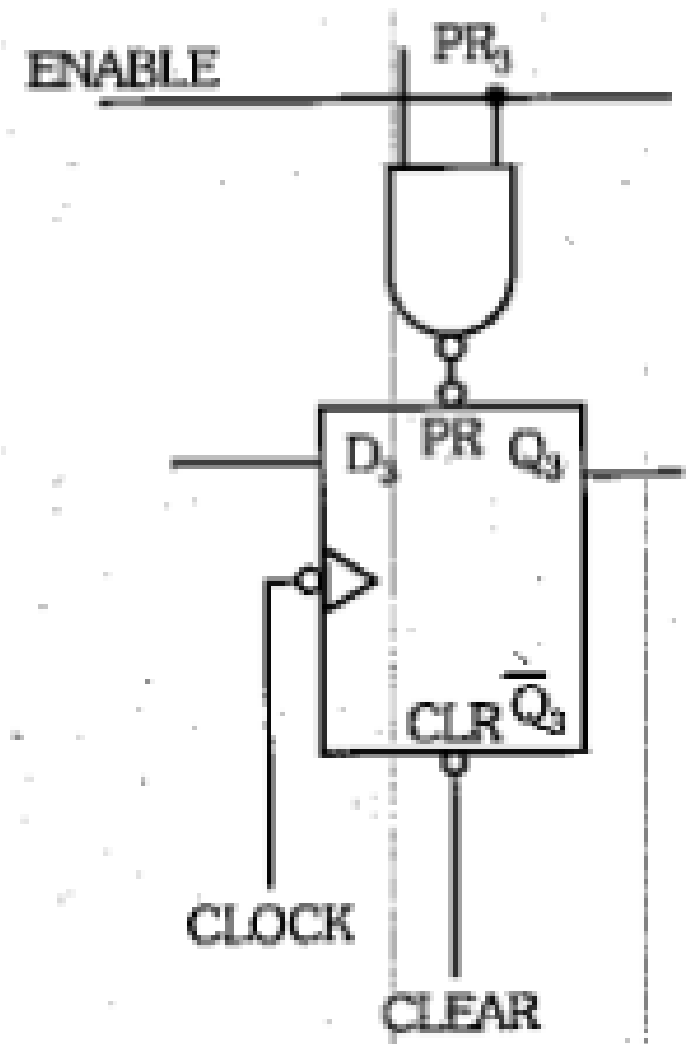
Registadores de Deslocamento

- Note o funcionamento de ENABLE
 - Se $ENABLE = 0$? $Preset = 1$.
 - Se $ENABLE = 1$? $Preset = \sim(PR_0, PR_1 \text{ ou } PR_2 \dots)$



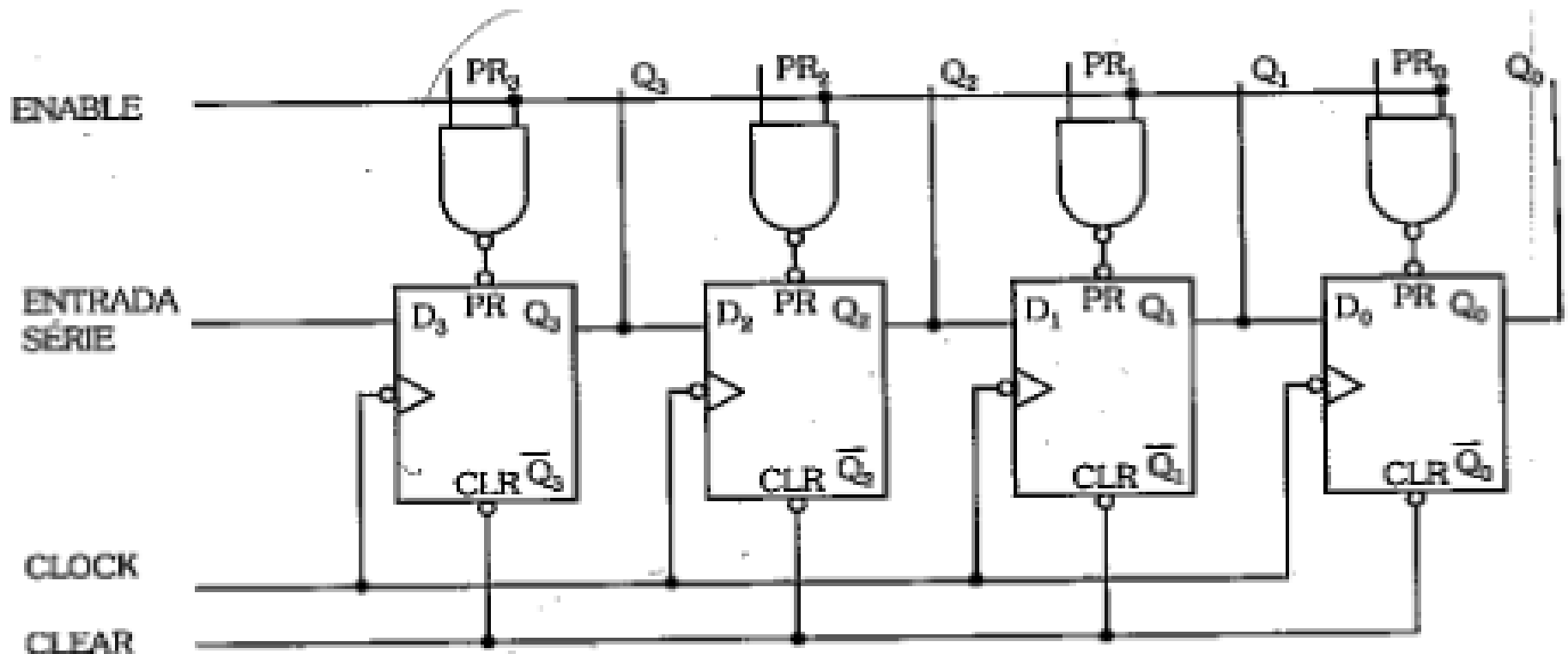
Registadores de Deslocamento

- Lembre que **Preset** e **Clear** funcionam invertido (ver tabela-verdade)
- Vamos supor que aplicamos um Clear pra começar
- Então, se $ENABLE = 0$, $Preset = 1$ e flip-flop atua normalmente
- Com $ENABLE = 1$, PR depende de PR3
 - Se $PR3 = 1$, $Q3 = 1$
 - Se $PR3 = 0$, $Q3 = 0$
- Então $ENABLE$ “ativa” entrada de PR3, que dita valor de $Q3$



Registradores de Deslocamento

- Nesse caso, Q_0 vai assumir o valor de cada entrada. Primeiro, PR_0 , depois da descida de clock PR_1 , então PR_2 e por fim PR_3 (saída em série!)



Registradores de Deslocamento

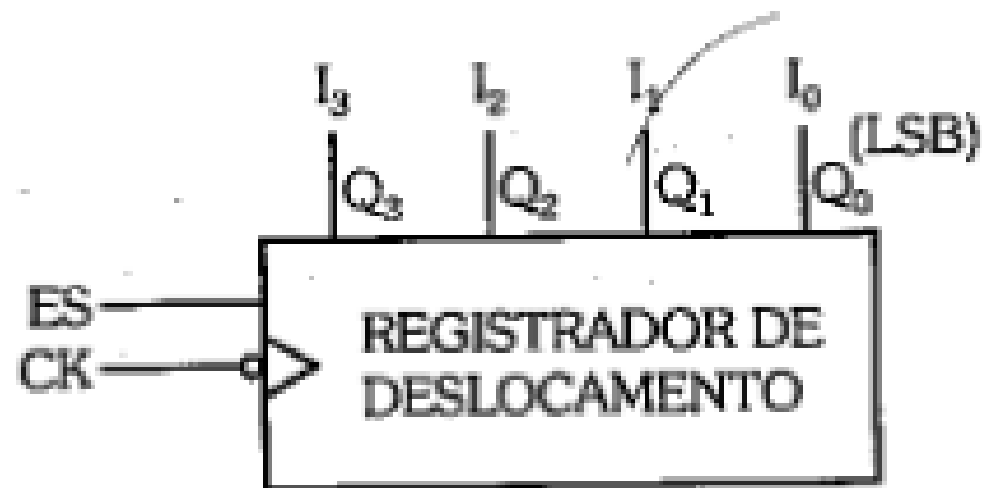
- Também podemos fazer registradores série-série e paralelo-paralelo.
- Tudo depende de onde entramos e de onde lemos os dados (se lermos em Q0, será em série, se lermos em PR0 a PR3), será em paralelo

Registradores de Deslocamento

- É possível usar o registrador de deslocamento para fazer um **multiplicador** ou **divisor** por 2.

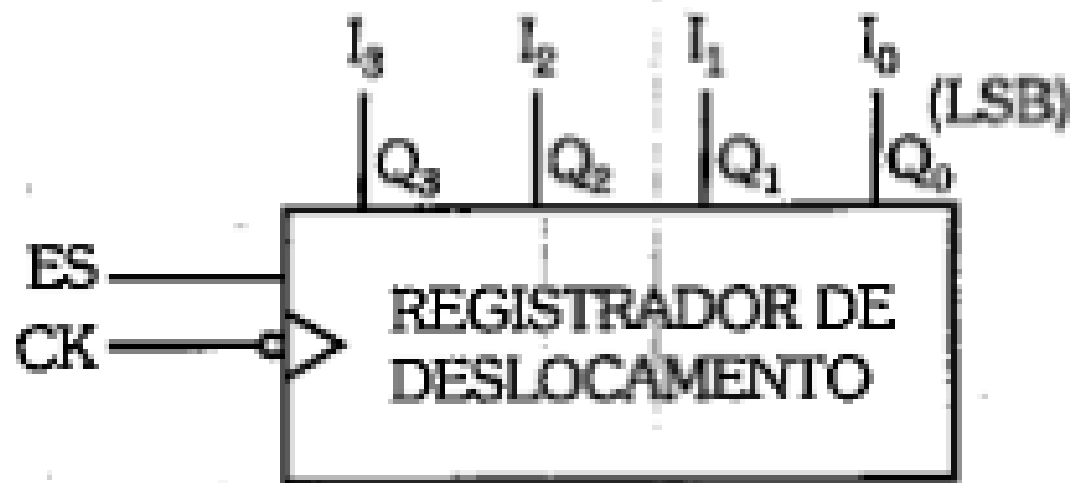
Registadores de Deslocamento

- Para **dividir por dois**:
 - Deslocamos registrador para a direita e adicionamos 0 no bit que entra pela esquerda
 - Saída em Q0 é o resto da divisão
 - Exemplos?



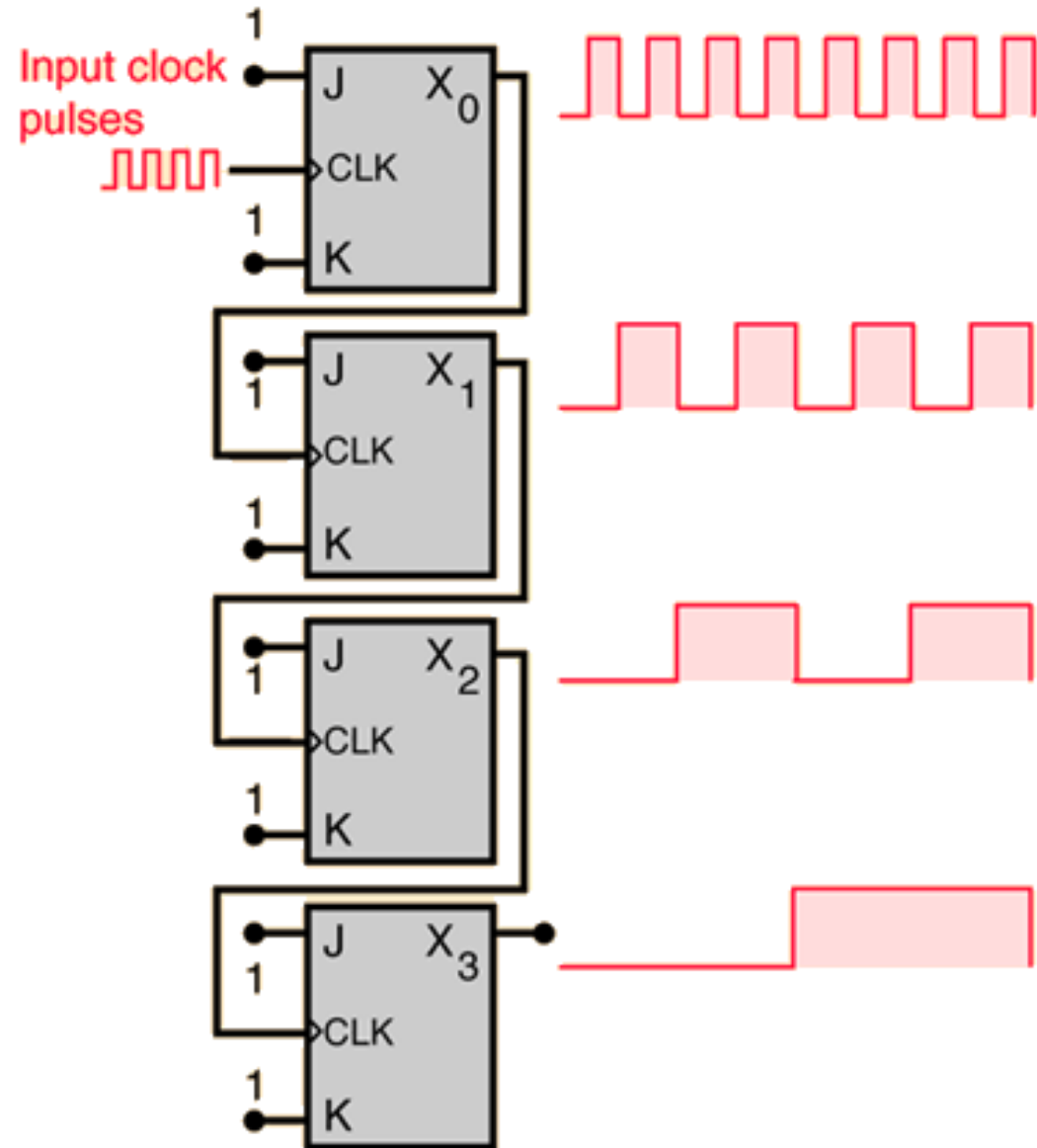
Registradores de Deslocamento

- Para **multiplicar por dois**:
 - Deslocamos registrador para a **esquerda** (temos que fazer algumas adaptações) e adicionamos 0 no bit da direita
 - Exemplos?



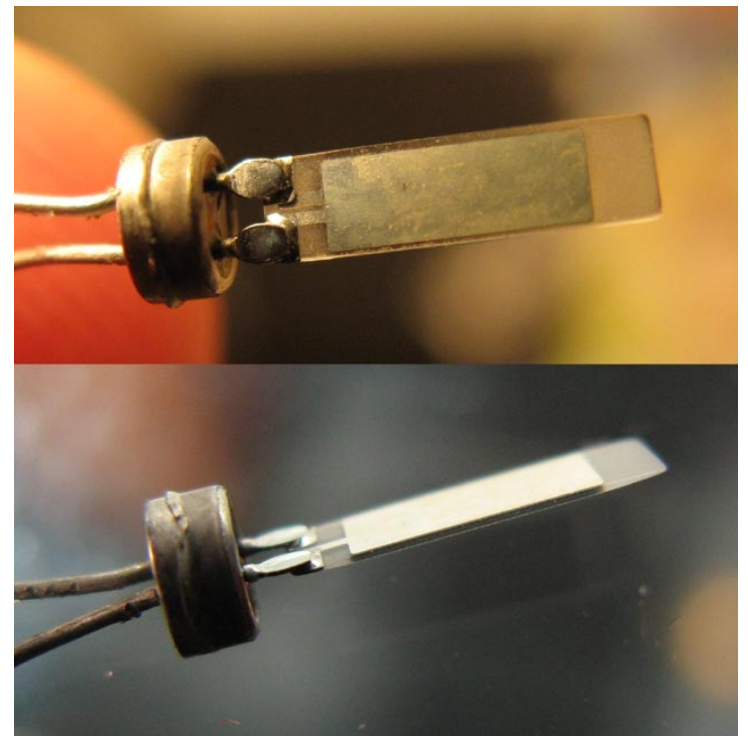
Contadores

- Contadores podem ser **assíncronos** ou **síncronos**
- Contadores assíncronos possuem uma entrada de clock no primeiro flip-flop, pra todo o sistema, e os outros clocks são derivados deste primeiro.
- Usamos **flip-flop T (toggle)**



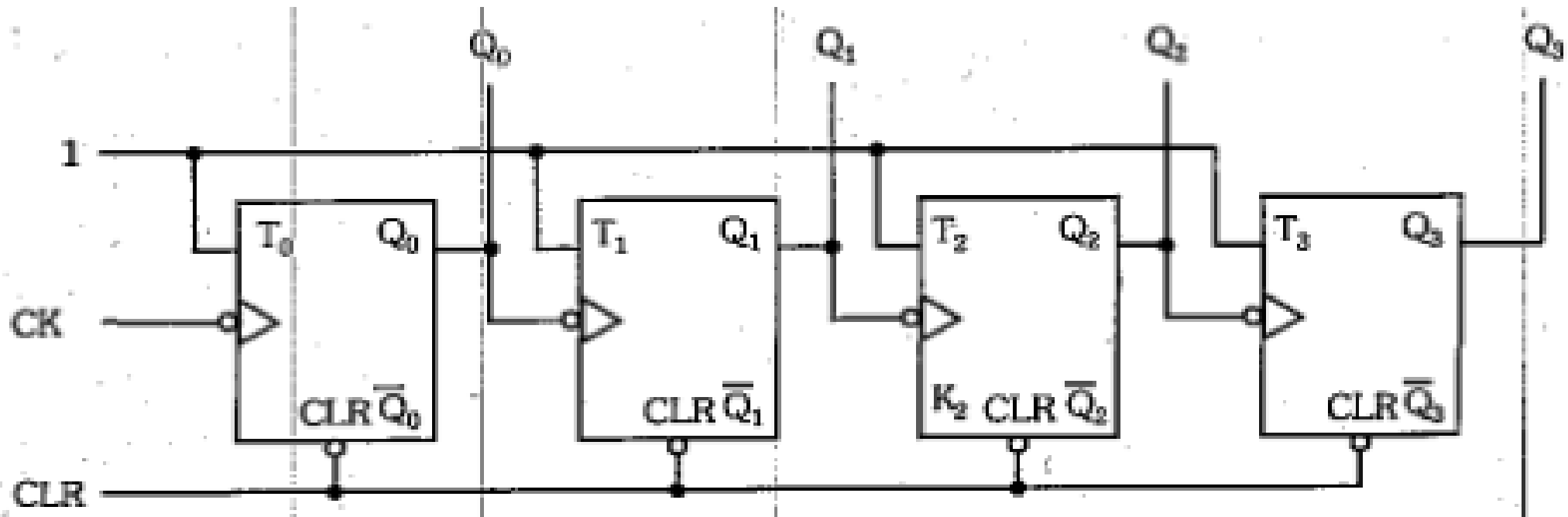
Contadores

- Aplicações:
 - Program Counter
 - Divisor de frequência
 - Relógio digital
 - Conversor A/D



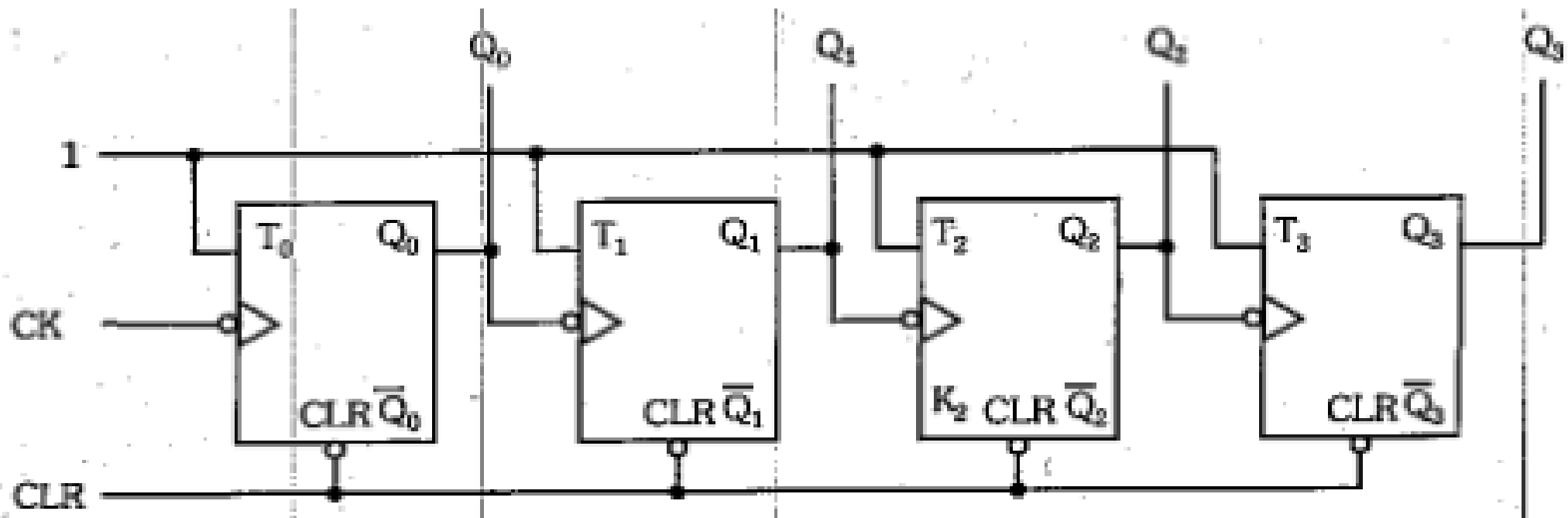
Contadores

- Objetivo do contador: colocar na saída números binários em sequência
- No flip-flop T, se $T = 1$, $Q_a = \sim Q_a$ (ou seja, permuta)
- Note que a **entrada de clock dos flip-flops**, exceto no primeiro, é a **saída do flip-flop anterior**
- Na figura: contador assíncrono (contador de *ripple*)

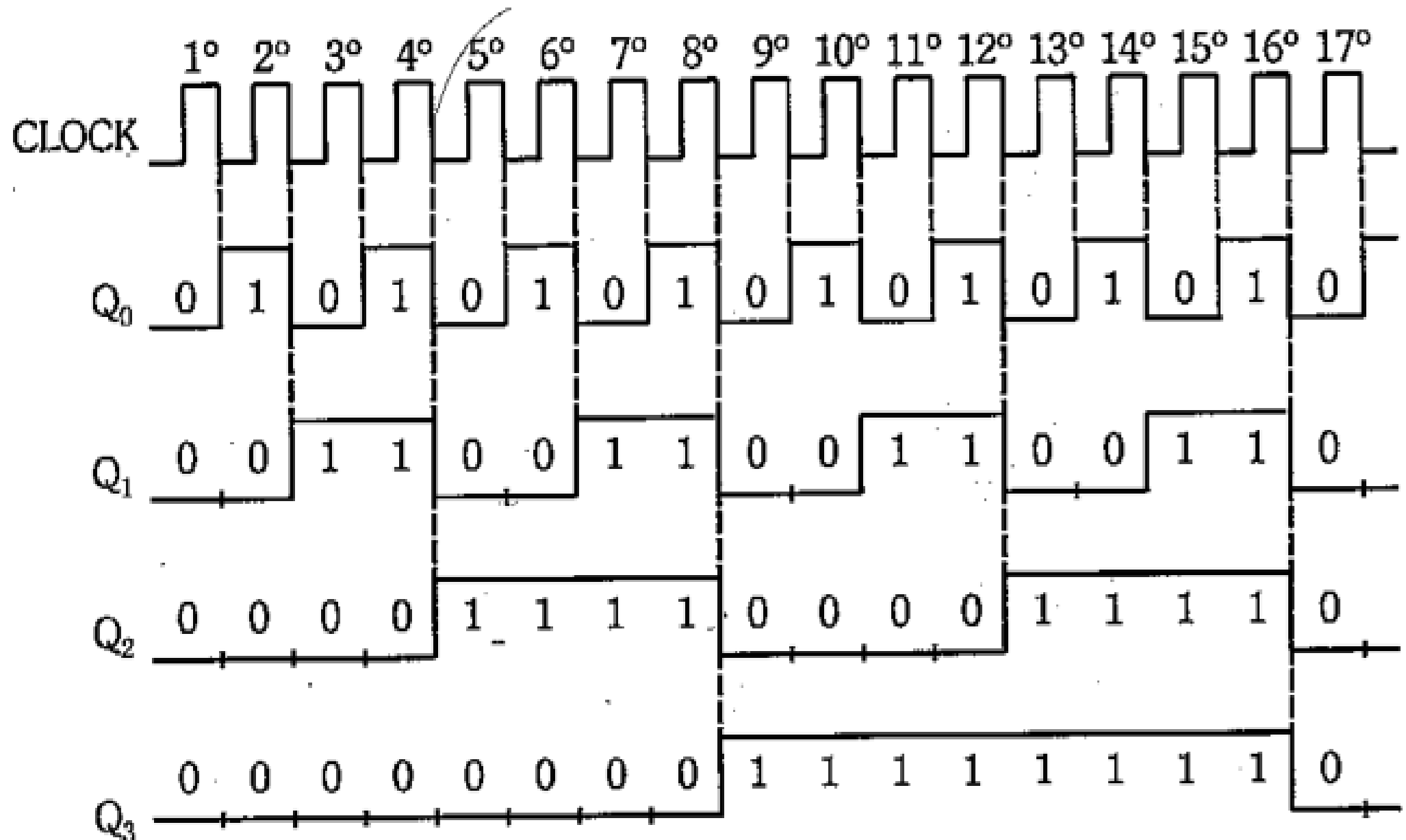


Contadores

- Vamos começar aplicando CLEAR em todos os flip-flops.
- Q0 muda com toda descida do Clock.
- Q1 muda com toda descida de Q0
- Q2 muda com toda descida de Q1 (etc.)
- Vamos desenhar as linhas do tempo de cada saída?

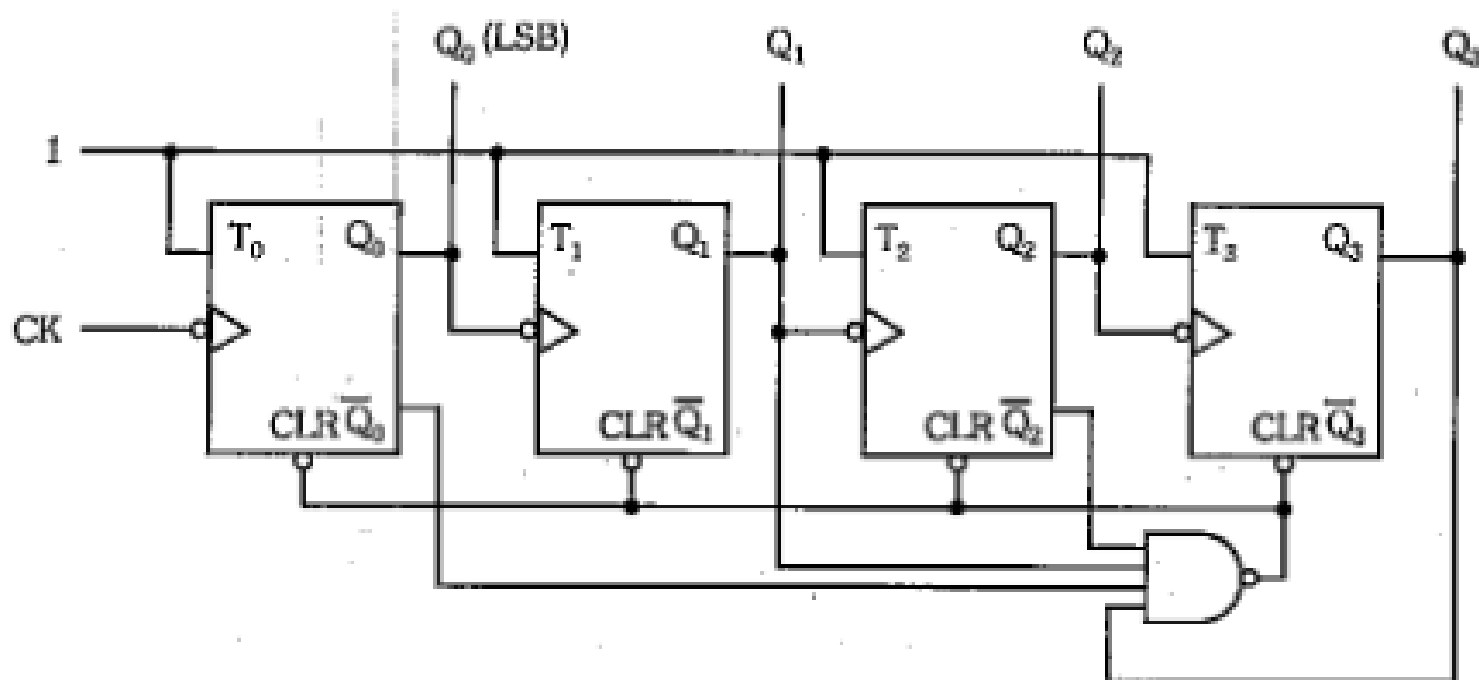


Contadores



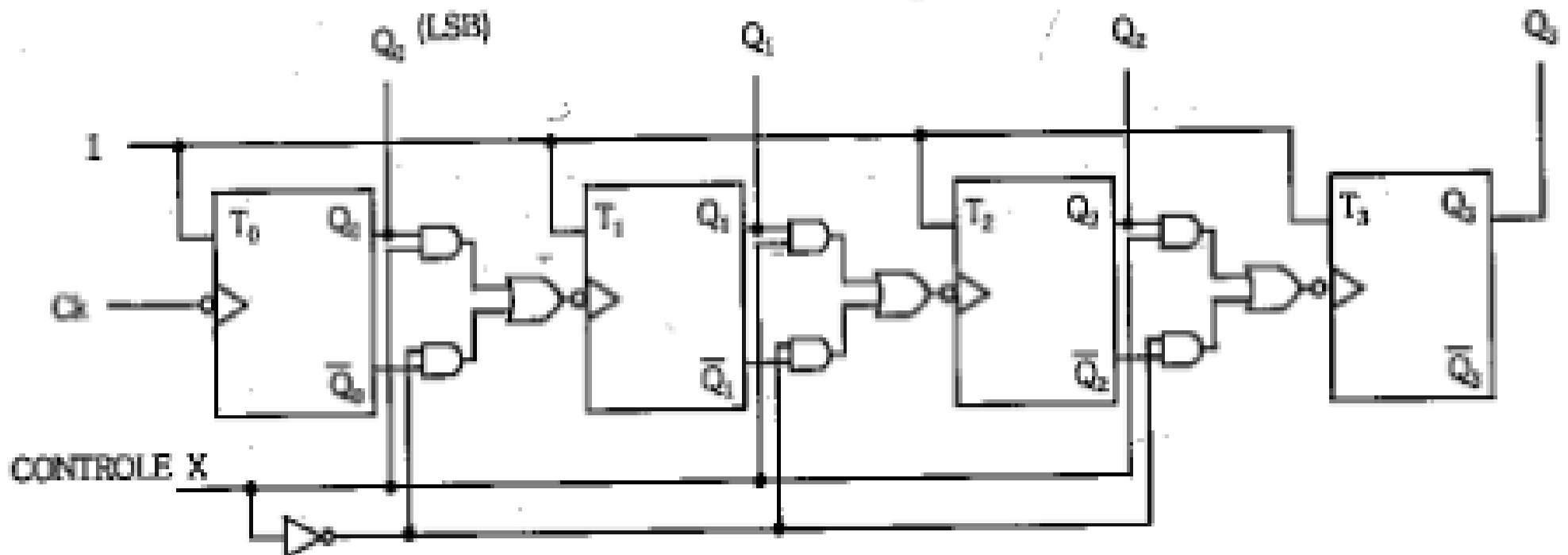
Contadores

- Contador de **década** (conta até 10)
 - Quando contador chega em 1010 (10_2), aplicamos **clear**, para voltar a contar do 0000
 - Podemos usar a mesma técnica para contar até qualquer valor que desejamos



Contadores

- Contador Assíncrono Crescente/Decrescente (**Up/Down**)
 - Como fica a **tabela** da contagem decrescente?
 - Precisamos de uma variável de controle (quando 1, crescente, quando 0, decrescente)



Contadores

- Contador **Síncrono**

- O clock entra em todos os flip-flops. É nítido que outras mudanças são necessárias pra contar.
- Lembrando a tabela-verdade do JK:

J	K	Qf
0	0	Qa
0	1	0
1	0	1
1	1	\overline{Qa}

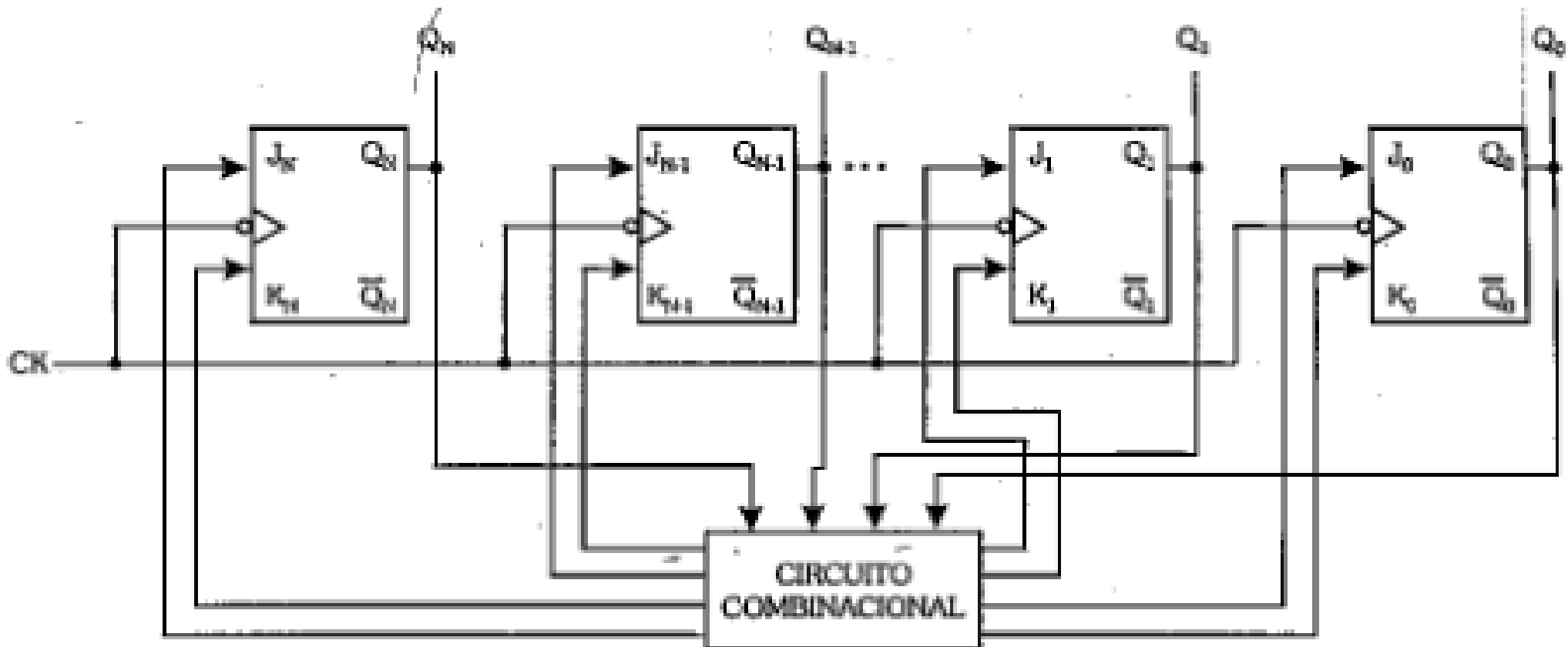
Contadores

- Contador **Síncrono**
 - Vamos fazer agora a tabela do **valor desejado futuro vs valor atual**
 - Vamos analisar caso a caso:

	Qa	Qf	J	K
1)	0	0	0	X
2)	0	1	1	X
3)	1	0	X	1
4)	1	1	X	0

Contadores

- O número futuro depende do número atual
- A partir disso podemos gerar um circuito combinacional pra controlar J e K de forma a realizar a contagem



Contadores

- Podemos contar qualquer sequência, mas vamos fazer um contador comum crescente
- Lembrando:
 - Entradas Q3, Q2, Q1 e Q0 (por ex.: 0000)
 - Saídas J3, K3, J2, K2, J1, K1, J0 e K0
- Vamos fazer a **tabela-verdade**

Contadores

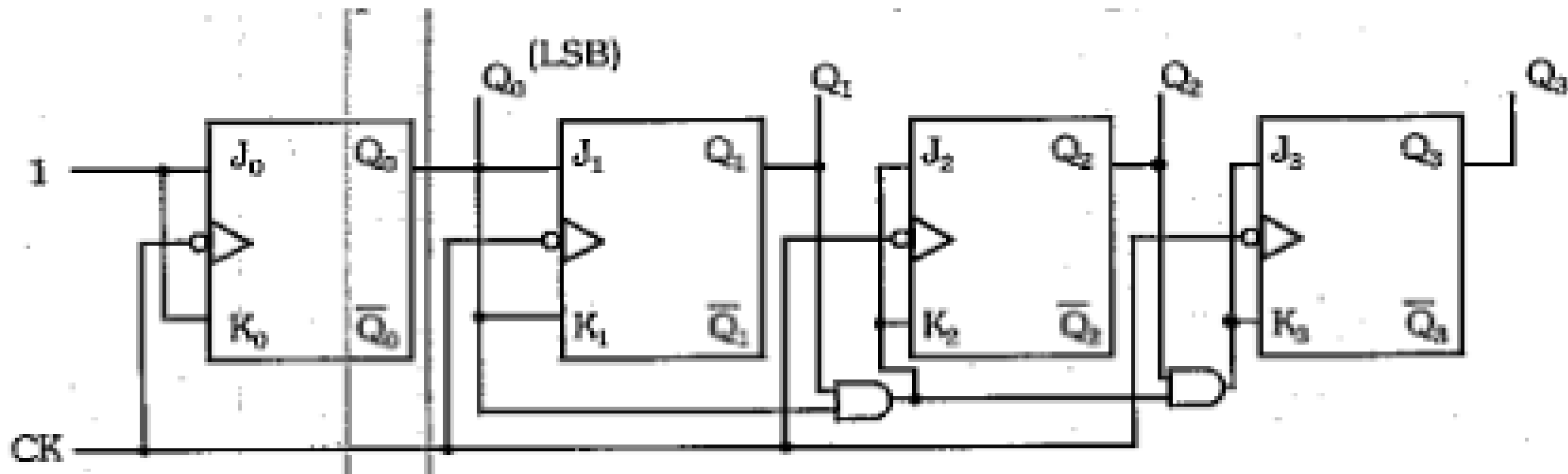
Descidas do pulso de clock	Q_3	Q_2	Q_1	Q_0	J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0
1ª	0	0	0	0	0	X	0	X	0	X	1	X
2ª	0	0	0	1	0	X	0	X	1	X	X	1
3ª	0	0	1	0	0	X	0	X	X	0	1	X
4ª	0	0	1	1	0	X	1	X	X	1	X	1
5ª	0	1	0	0	0	X	X	0	0	X	1	X
6ª	0	1	0	1	0	X	X	0	1	X	X	1
7ª	0	1	1	0	0	X	X	0	X	0	1	X
8ª	0	1	1	1	1	X	X	1	X	1	X	1
9ª	1	0	0	0	X	0	0	X	0	X	1	X
10ª	1	0	0	1	X	0	0	X	1	X	X	1
11ª	1	0	1	0	X	0	0	X	X	0	1	X
12ª	1	0	1	1	X	0	1	X	X	1	X	1
13ª	1	1	0	0	X	0	X	0	0	X	1	X
14ª	1	1	0	1	X	0	X	0	1	X	X	1
15ª	1	1	1	0	X	0	X	0	X	0	1	X
16ª	1	1	1	1	X	1	X	1	X	1	X	1

Contadores

- Vamos fazer o mapa de Karnaugh pra extrair a fórmula e desenhar o circuito combinacional pra cada saída

Contadores

- Circuito final:

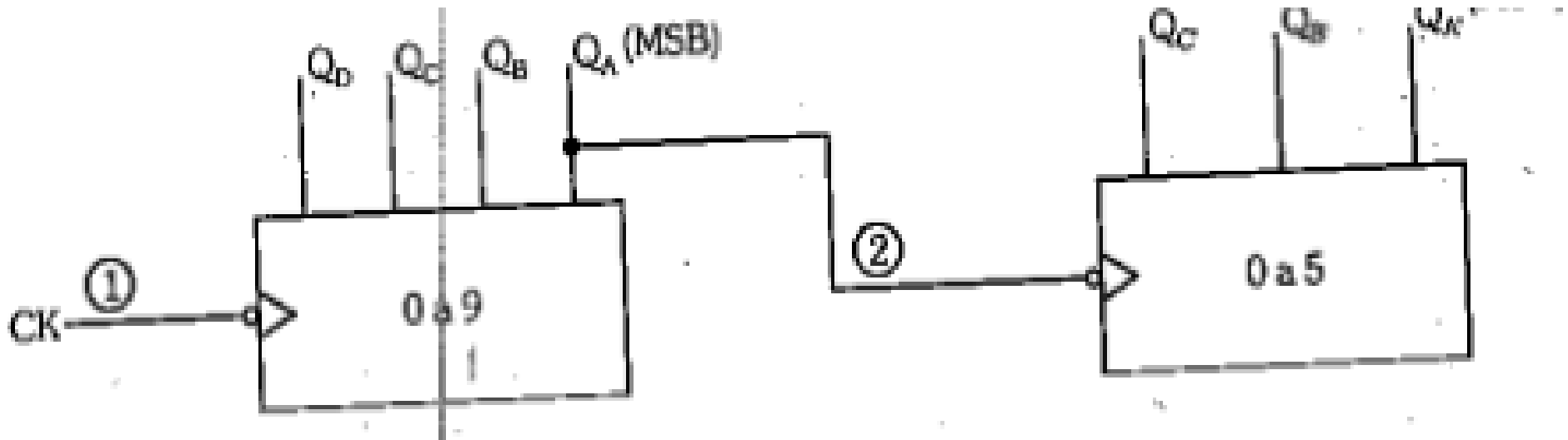


Contadores

- Outros contadores síncronos:
 - Ring Counter (Contador de anel)
 - Contador Johnson
 - Contador de Código Gray

Contadores

- E para fazermos um relógio que conta de 0 a 59?
 - Podemos usar 6 bits, ou:



- E se quisermos fazer contadores de segundos, minutos e horas?