PCS 3225
Sistemas Digitais II

Módulo 02 – Biestáveis

Andrade, Marco Túlio Carvalho de; Saraiva, Antônio Mauro; Simplício, Marcos Antônio.

Professores Responsáveis

Adaptado por Glauber (2018)

Conteúdo

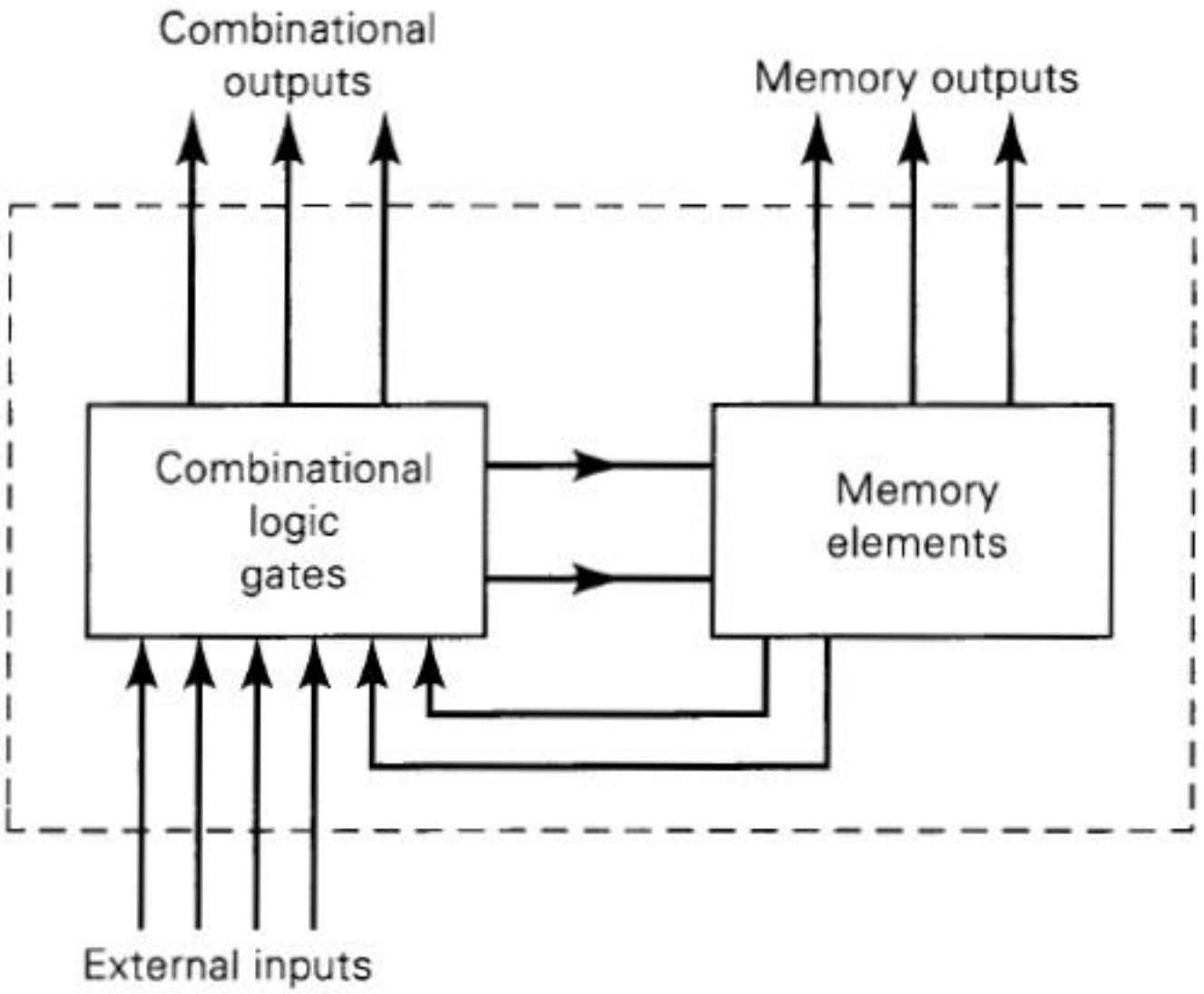
Biestáveis

1. Circuitos Lógicos Sequenciais
   - 1.1. Estado de um circuito sequencial
   - 1.2. Classes de circuitos sequenciais
   - 1.3. *Clock* ou relógio
   - 1.4. Definições: *Clock* ativo em Baixo e Alto
   - 1.5. Elementos de Memória
2. Latch RS Negativo
3. Latch RS
4. Latch RS com Clock
5. Latch D sensível ao Nível

Conteúdo

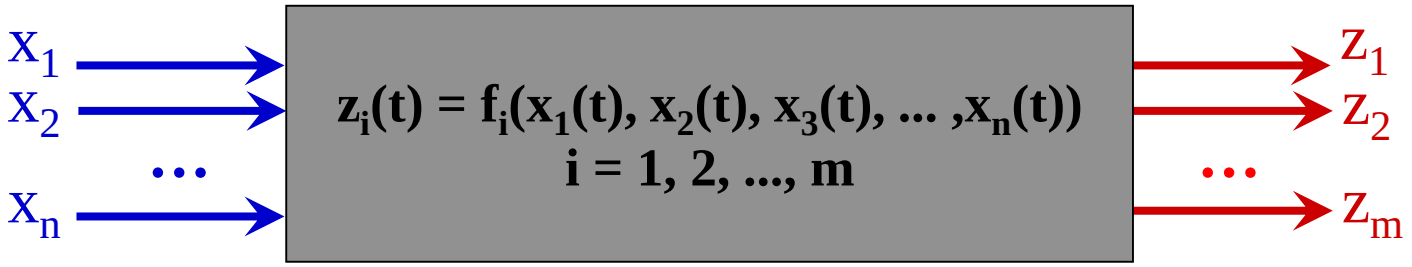
Biestáveis

1. Circuitos Lógicos Sequenciais

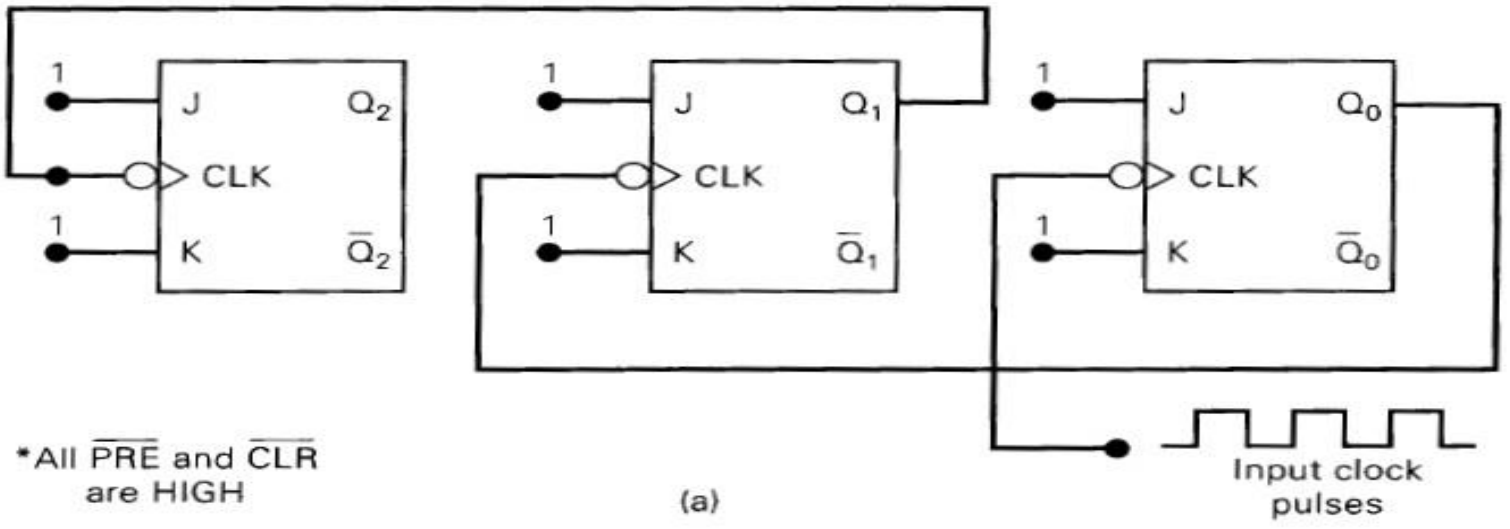


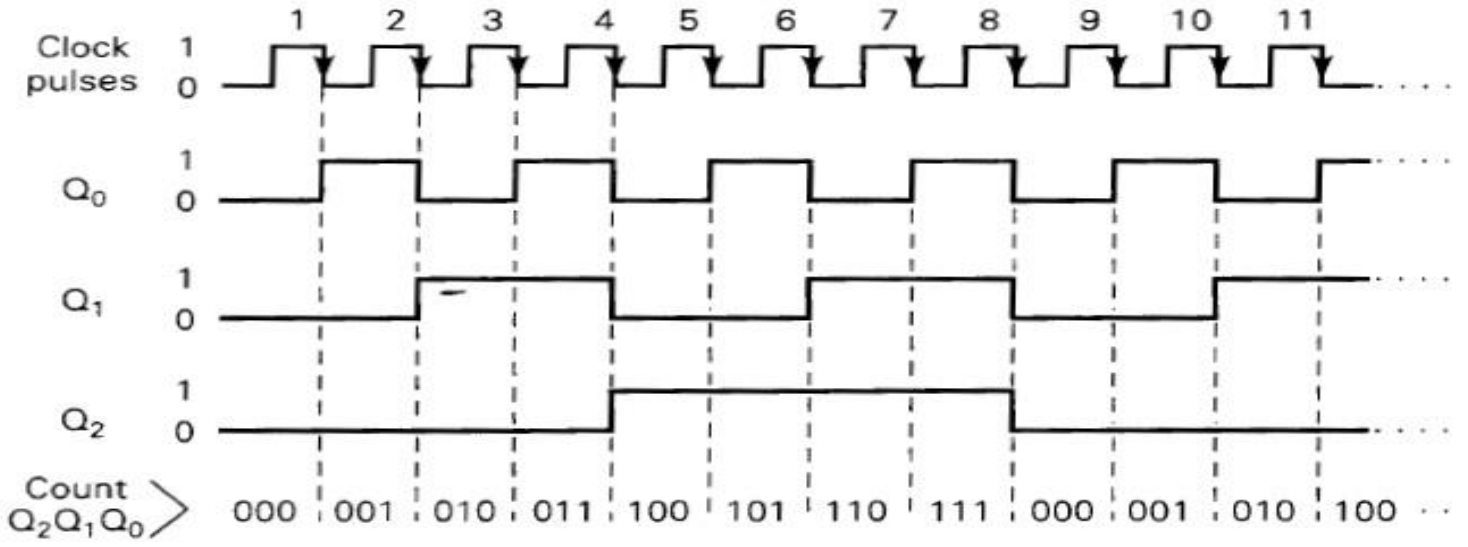
1. Circuitos Lógicos Sequenciais

- Os circuitos lógicos podem ser divididos em **duas classes**: Circuitos Combinatórios & Circuitos Sequenciais.
- Nos **circuitos combinatórios**, os valores das saídas num instante **t** dependem exclusivamente dos valores das entradas neste instante:
- Estes **não** têm "**memória**". Ex.: porta NAND
  - 00 → 1 ; 01→ 1; 10→ 1; 11 → 0.



1. Circuitos Lógicos Sequenciais





1. Circuitos Lógicos Sequenciais

- Nos **circuitos sequenciais**, os **valores** das **saídas** num **instante t dependem** dos **valores** das **entradas neste instante** e também em **instantes passados**:

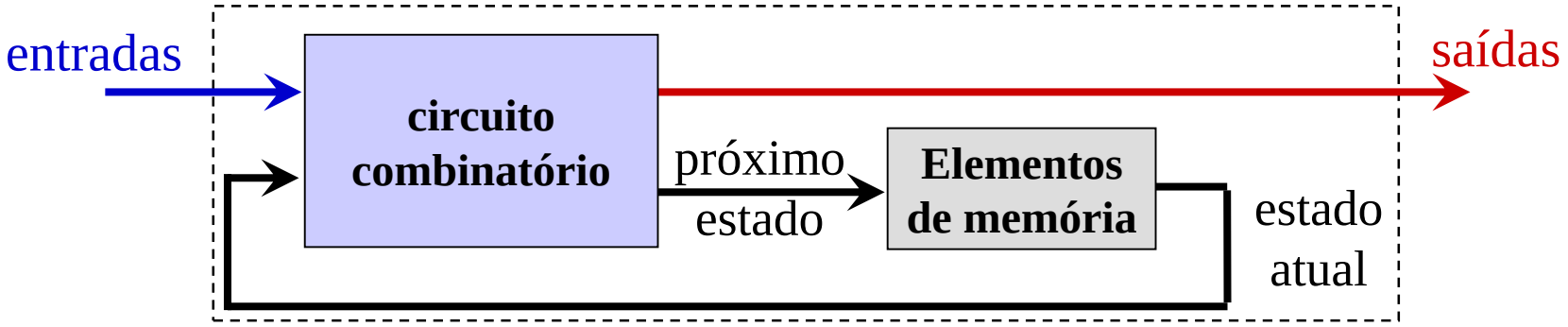
  $z_i(t) = f_i(x_1(t), x_1(t-1), x_1(t-2), ..., x_2(t), x_2(t-1),..., x_3(t), x_3(t-1), ..., x_n(t), x_n(t-1), ...)$, $i = 1, 2, ..., n$

- Exemplos:
  - cadeado de mala/bicicleta x cadeado de cofre;
  - seletor de canal de TV.

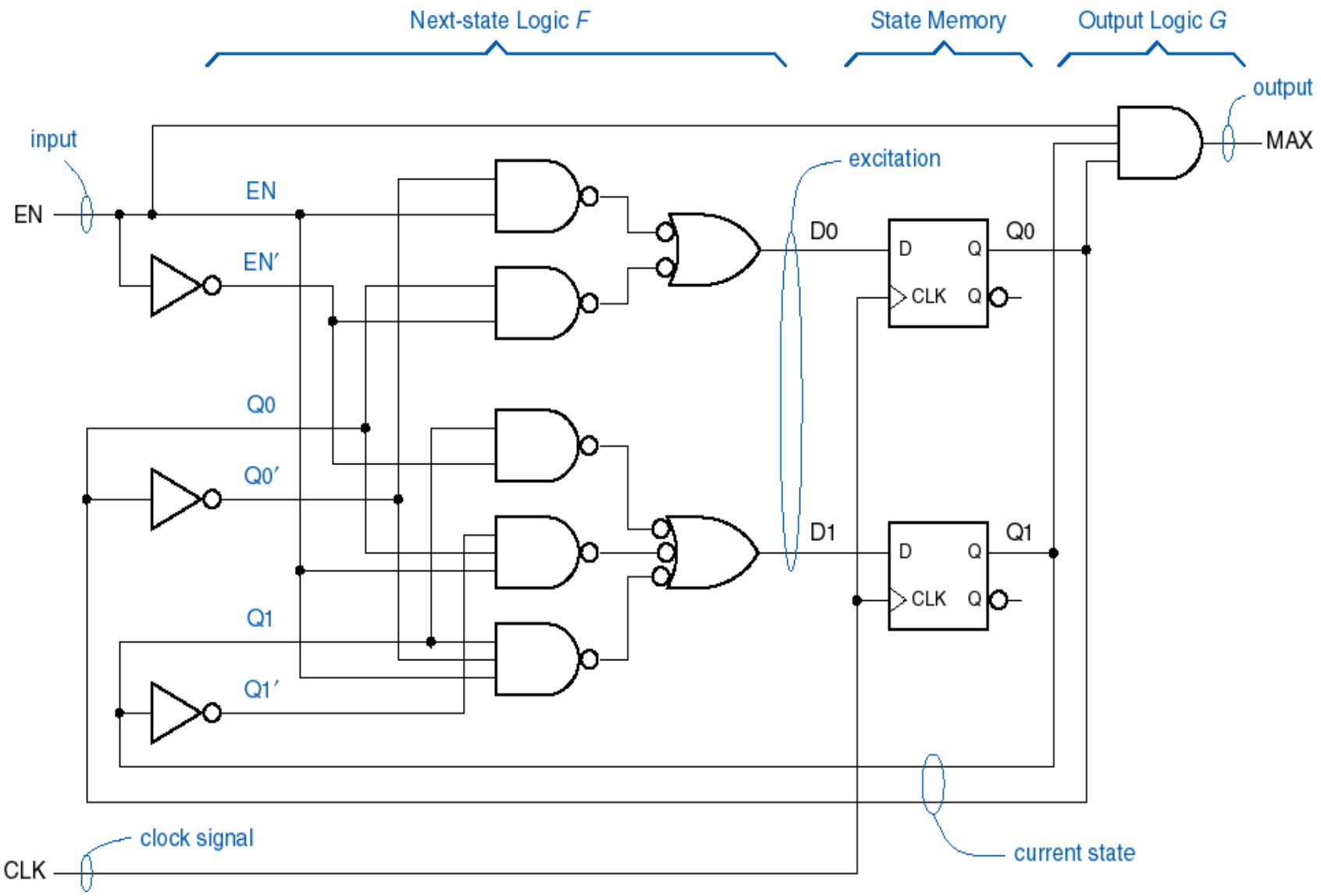
1.1 Estado de um circuito sequencial

- **Estado** de um circuito sequencial:

  É um conjunto de *valores de variáveis* –denominadas **Variáveis de Estado**– que contêm toda a informação necessária sobre o passado do circuito para descrever o seu comportamento futuro.



1. Circuitos Lógicos Sequenciais



1.2 Classes de Circuitos Sequenciais

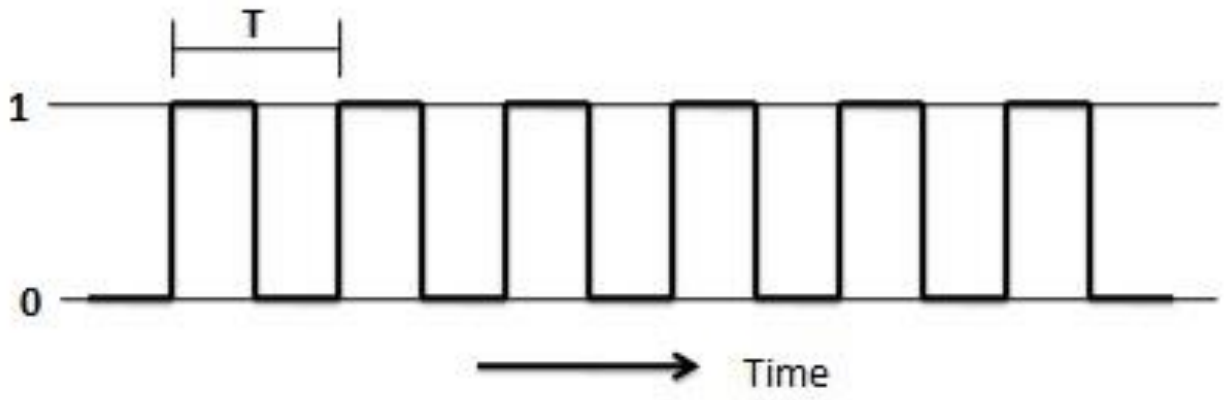
- Os circuitos lógicos sequenciais podem ser divididos em duas grandes **classes**:
  - Circuitos sequenciais **síncronos;**
  - Circuitos sequenciais **assíncronos.**
- Diferem quanto ao **instante de alteração do estado (e das saídas)** do circuito.

1.2 Classes de Circuitos Sequenciais

- Nos circuitos **síncronos**, as alterações nas variáveis de estado ocorrem em instantes específicos, sincronizados com a ocorrência de um sinal numa entrada especial denominada relógio (***clock***).
- Nos circuitos **assíncronos**, não há tal sincronismo.

1.3 *Clock* ou relógio

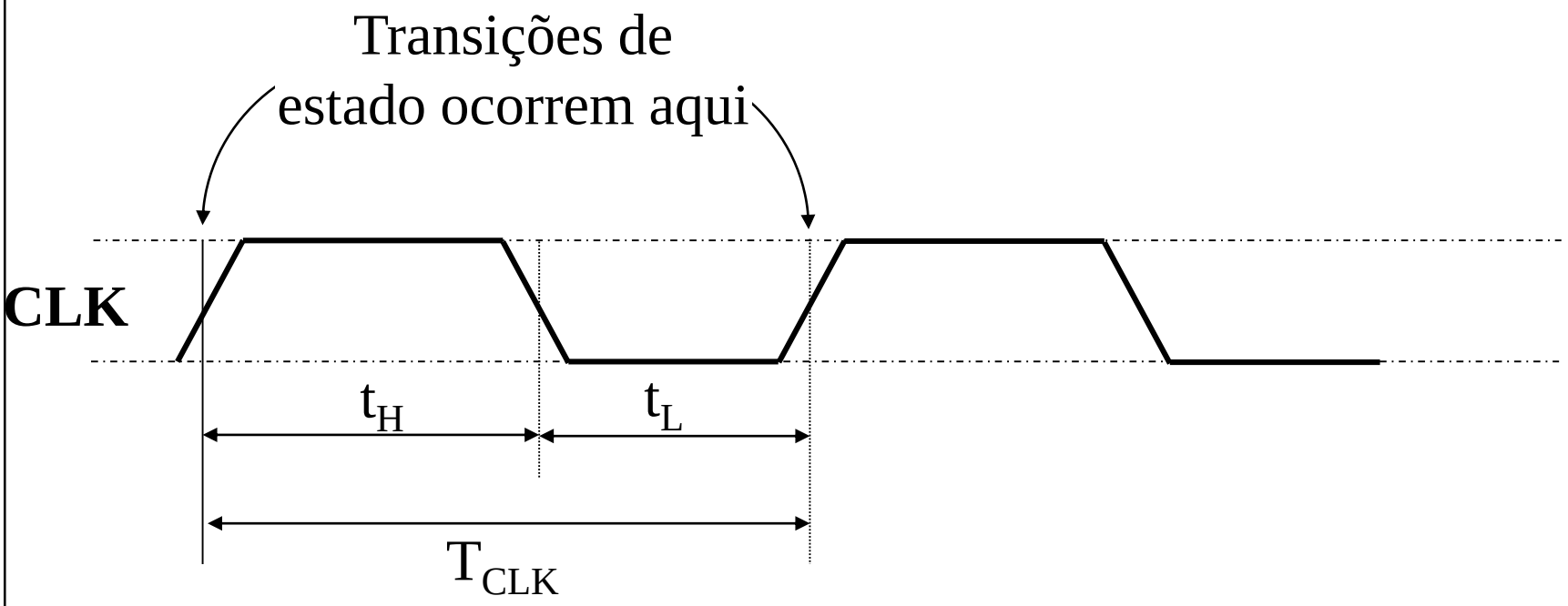
- Mudanças de estado em circuitos síncronos ocorrem em momentos especificados por um sinal de *clock*.
- Onda (periódica) retangular:



1.3 *Clock* ou relógio

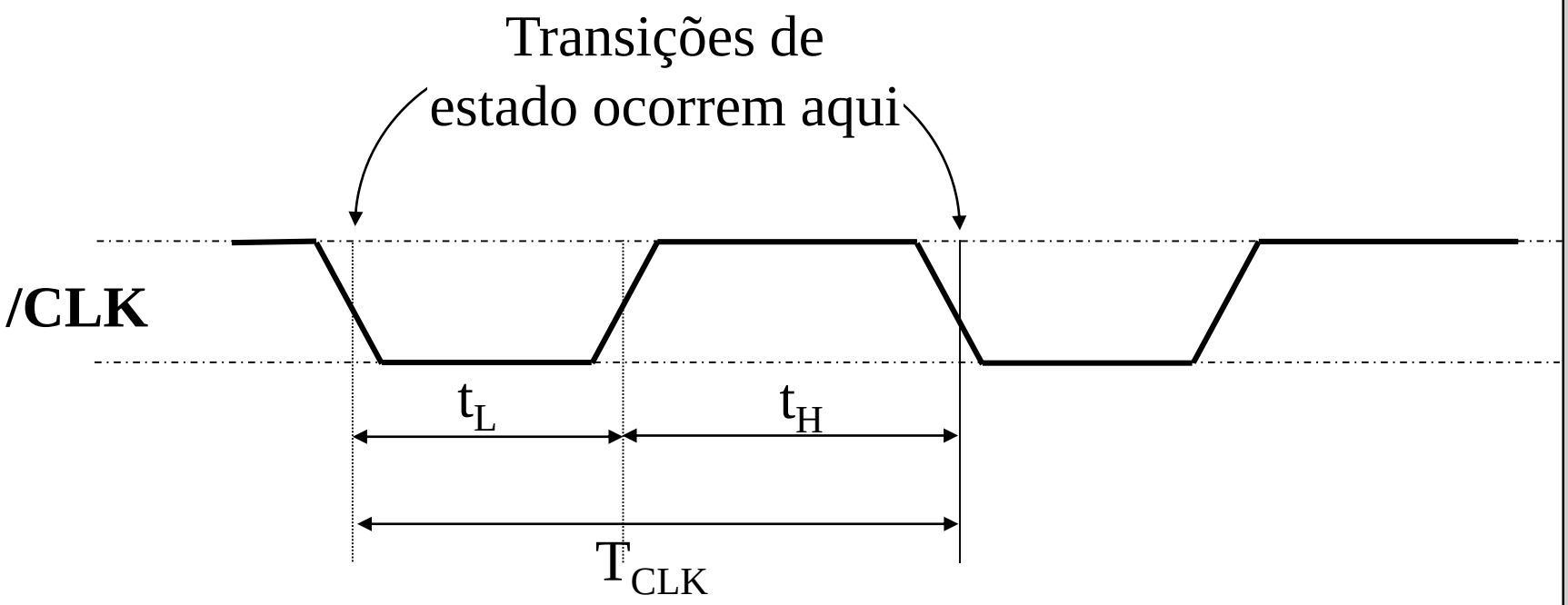
- ***Clock* ativo** em **Alto** ou **Baixo:**
  - **Ativo em Alto**, se transições de estado ocorrem na borda de subida ou quando o sinal de *clock* está em 1;
  - **Ativo em Baixo**, caso contrário.
- **Período (T):**
  - Intervalo de tempo entre duas transições do *clock* no mesmo sentido. (f = 1/T)
- **Duty Cycle**:
  - Fração do período onde sinal está ativo.

1.4 Definições: *Clock* ativo em Alto



- Período = T_{CLK}
- Frequência = $1/ T_{CLK}$
- *Duty cycle* = t_H / T_{CLK}

1.4 Definições: *Clock* ativo em Baixo



- Período = T_{CLK}
- Frequência = 1/ T_{CLK}
- *Duty cycle* = t_L / T_{CLK}

1.5 Elementos de Memória

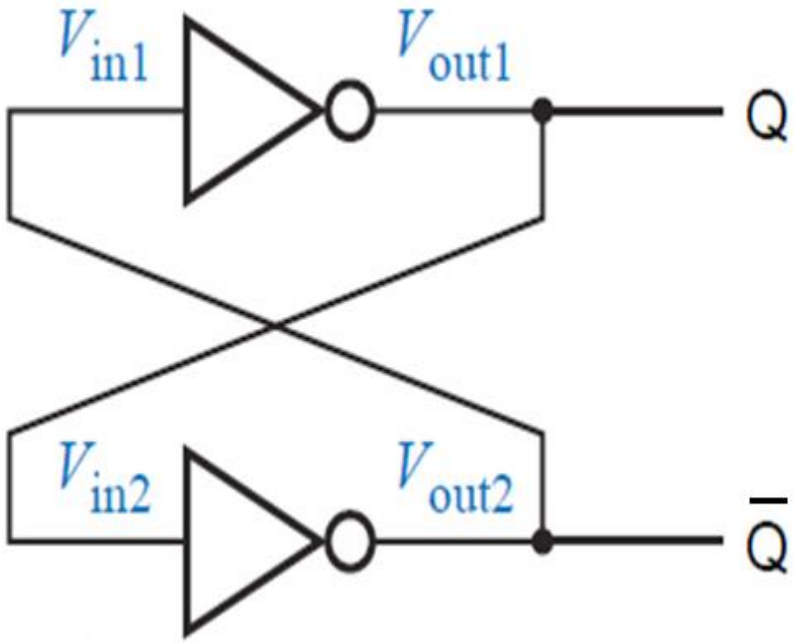
- Para guardar os valores passados das entradas, utiliza-se a noção de **estado**.
- Um **circuito sequencial síncrono** implementa uma **máquina de estados com número finito de estados**.
- **Questões**:
  - Como implementar um estado?
  - Como armazenar uma informação quando ela não está mais presente?

1.5 Elementos de Memória

- Para armazenar o estado, basta guardar o valor das variáveis (binárias) de estado.
- Precisamos de uma **memória de 1 bit**, que é um circuito sequencial.
- Com exatamente **dois estados** (0 e 1).
- Precisamos ser capazes de "escrever" 0s ou 1s nesta memória.
- Para circuitos **síncronos**, precisamos de um circuito que mude de estado apenas em um instante determinado (borda do ***clock***).

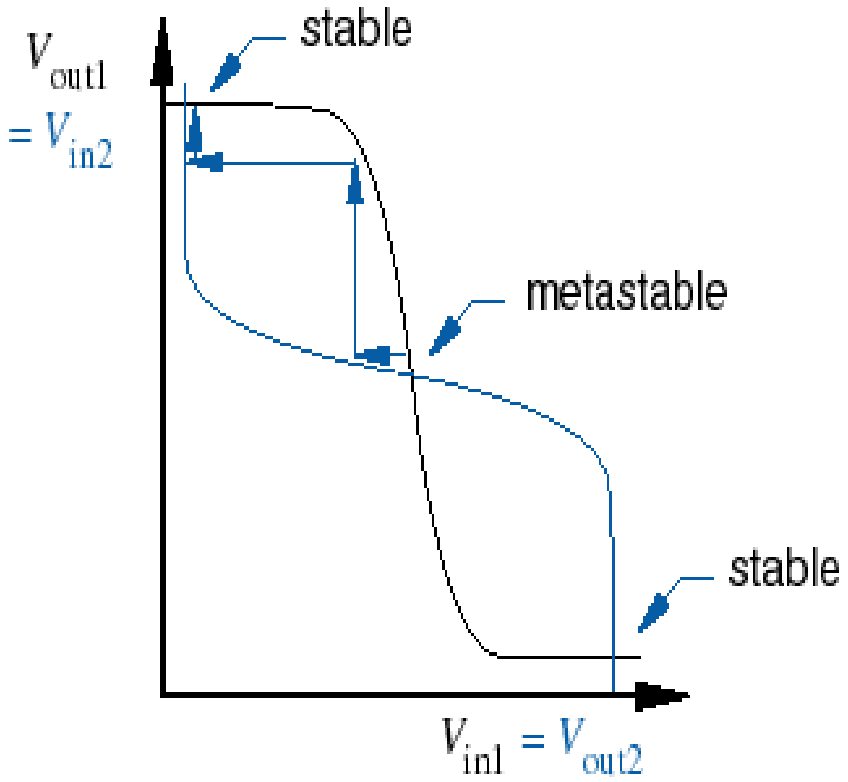
1.5 Elementos de Memória

- Circuito biestável mais simples
  - Sem entradas: sistema se estabiliza quando energizado
  - Dois estados: variável de estado Q = 0 ou 1



1.5 Elementos de Memória

- Meta estabilidade
  - Poderia permanecer nesse estado para sempre (não um estado binário/digital)



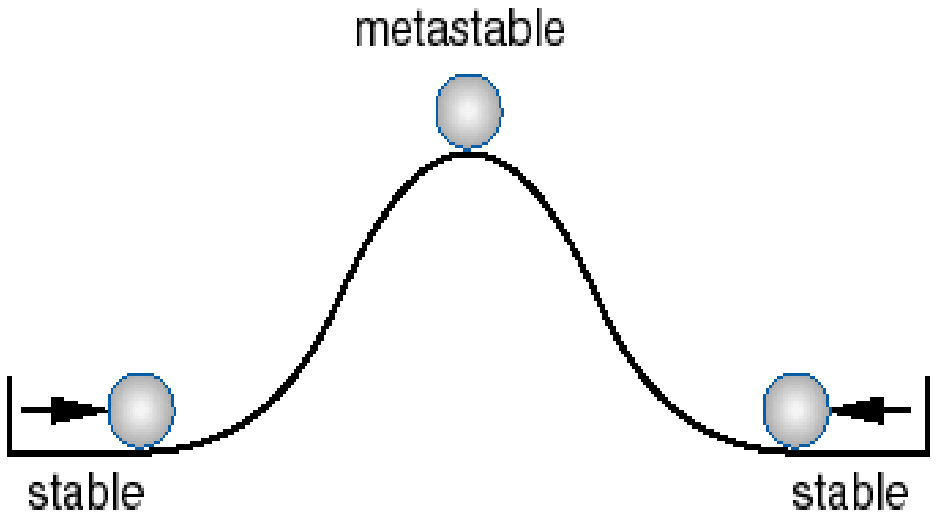
Transfer function:

$$V_{out1} = T(V_{in1})$$

$$V_{out2} = T(V_{in2})$$

1.5 Comportamento Meta-estável

- Circuito altamente realimentado
  - Realimentação leva o estado meta-estável para um dos estados estáveis
- Qualquer circuito biestável é suscetível à existência dessa meta estabilidade
- Pouca energia é suficiente para tirar do meta-estado.



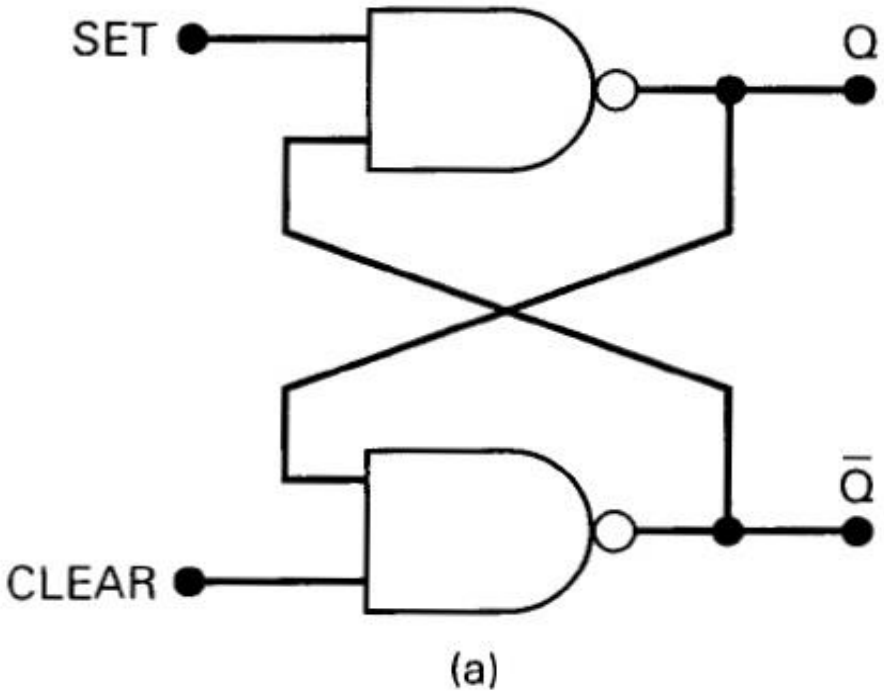
1.5 Elementos de Memória

- Os circuitos básicos que implementam a função de memória são denominados **biestáveis** ou ***flip-flops*** ou ***latches***.
- Existem diversos tipos de *flip-flops* e *latches*
  - RS (SR)
  - D
  - JK
  - T

2. Latch RS Negativo

ou Set-Reset ativo em Baixo, ou S'R'

- Entradas: S' (set') e R'(reset')
- Saídas: Q (quiescente) e Q' ou $\overline{Q}$



(a)

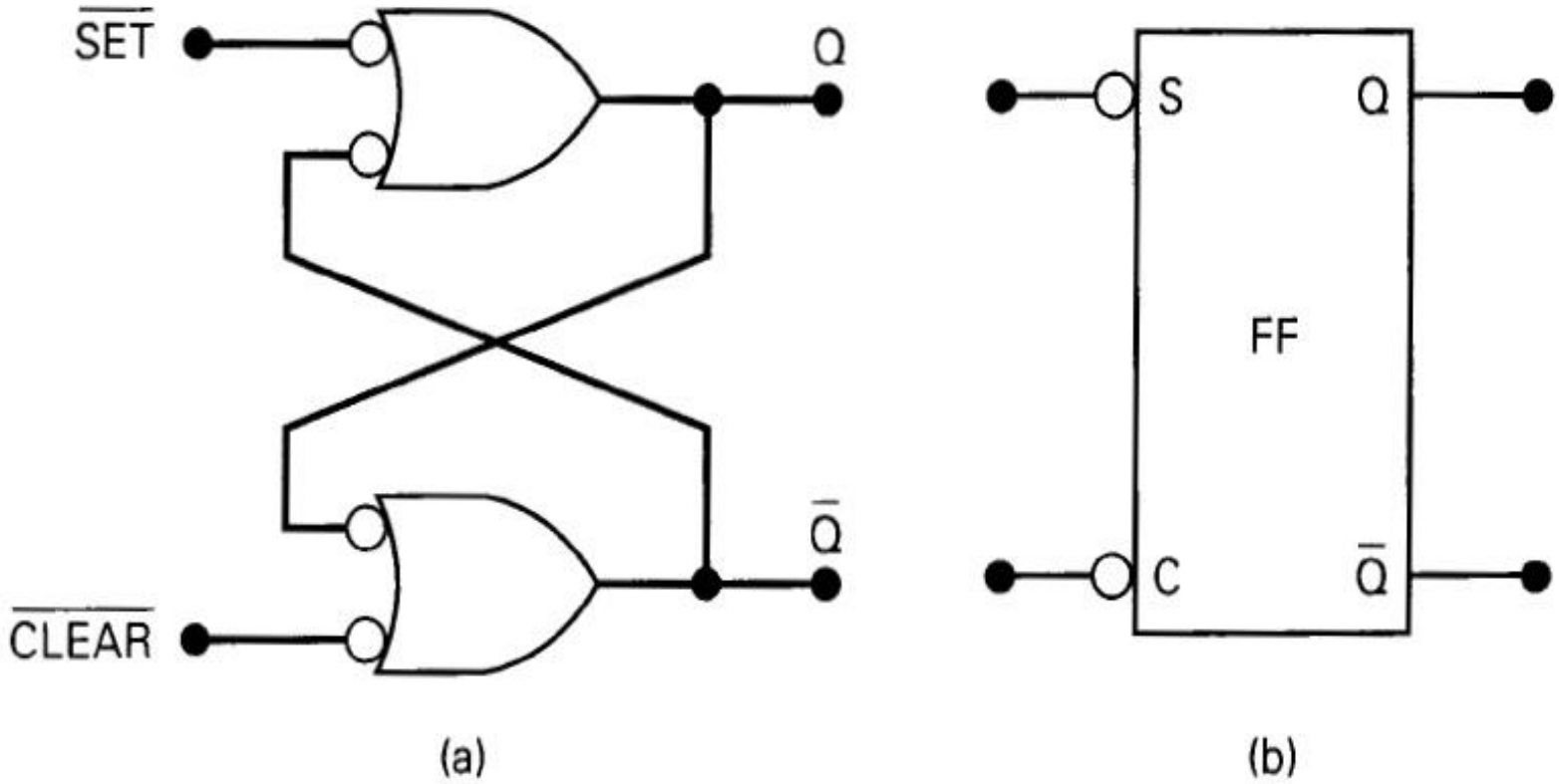
Set	Clear	Output
1	1	No change
0	1	Q = 1
1	0	Q = 0
0	0	Invalid*

*produces $Q = \overline{Q} = 1$

(b)

2. Latch RS Negativo

Latch RS Negativo – Representação Alternativa



2. Latch RS Negativo

$\overline{S}$	$\overline{R}$	Comportamento	Observação
0	1	impõe $Q=1 \quad e \quad \overline{Q}=0$	SET
1	0	impõe $Q=0 \quad e \quad \overline{Q}=1$	RESET
1	1	mantém o estado anterior	MANTÉM
0	0	estado proibido: $Q=\overline{Q}=1$	PROIBIDO

2. Latch RS Negativo

- O circuito "**lembra**" a **última entrada** que **assumiu o valor 0**.
- O valor $\overline{S} = \overline{R} = 0$ é **proibido** pois **não se deseja** o **mesmo valor** para $Q \quad e \quad \overline{Q}$
  - se após $\overline{S} = \overline{R} = 0$ tem-se $S = R = 1$, o **circuito pode oscilar** (se o atraso nas portas for idêntico), ou ter um **comportamento não determinístico** (não se sabe qual saída irá para "1" ou "0").

2. Latch RS Negativo

$Q(t)$ \ $\overline{S}(t)\,\overline{R}(t)$	00	01	11	10
0	1	1		
1	1	1	1	

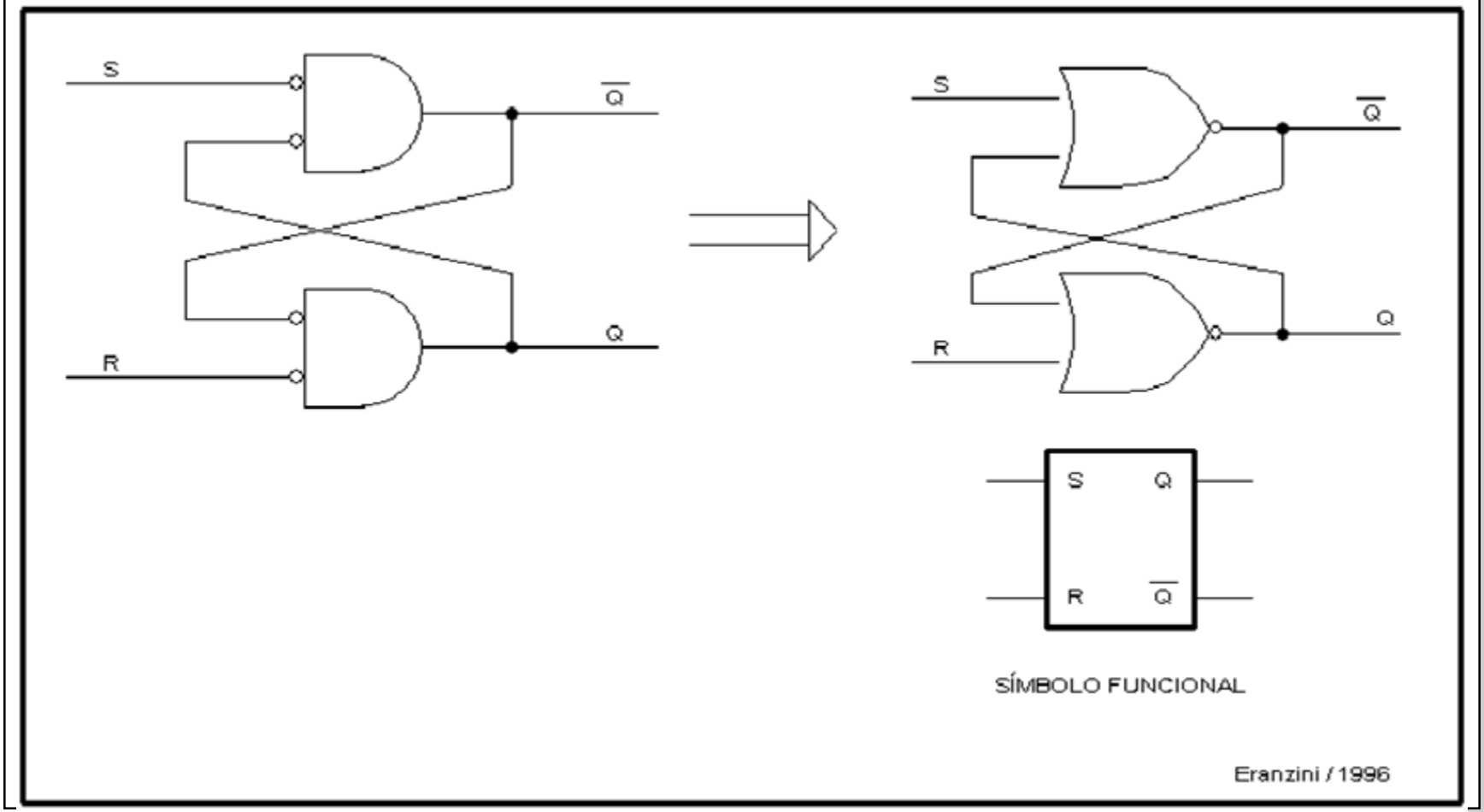
$Q(t+1)$

$$Q(t+1) = S(t) + \overline{R}(t).Q(t)$$

- Desvantagem: tem configuração proibida nas entradas.

3. Latch RS (ou Set-Reset, ou SR)

- Entradas: S (set) e $\overline{R}$ (reset)
- Saídas: **Q** e **Q'** ou $\overline{\mathbf{Q}}$



3. Latch RS (ou Set-Reset, ou SR)

S	R	Q(t)	Q(t+1)	Observação
0	0	0	0	MANTÉM
0	0	1	1	
0	1	0	0	RESET
0	1	1	0	
1	0	0	1	SET
1	0	1	1	
1	1	0	0	PROIBIDO
1	1	1	0	

3. Latch RS (ou Set-Reset, ou SR)

- O circuito "**lembra**" a **última entrada** que **assumiu o valor 1**.
- O valor $S = R = 1$ é proibido pois **não se deseja o mesmo valor** para Q e $\overline{Q}$ e se após $S = R = 1$ tem-se $S = R = 0$, o **circuito pode oscilar** (caso o **atraso nas portas seja idêntico**) ou apresentar um **comportamento não determinístico**.

3. Latch RS (ou Set-Reset, ou SR)

$Q(t)$ \ $S(t)R(t)$	00	01	11	10
0				1
1	1			1

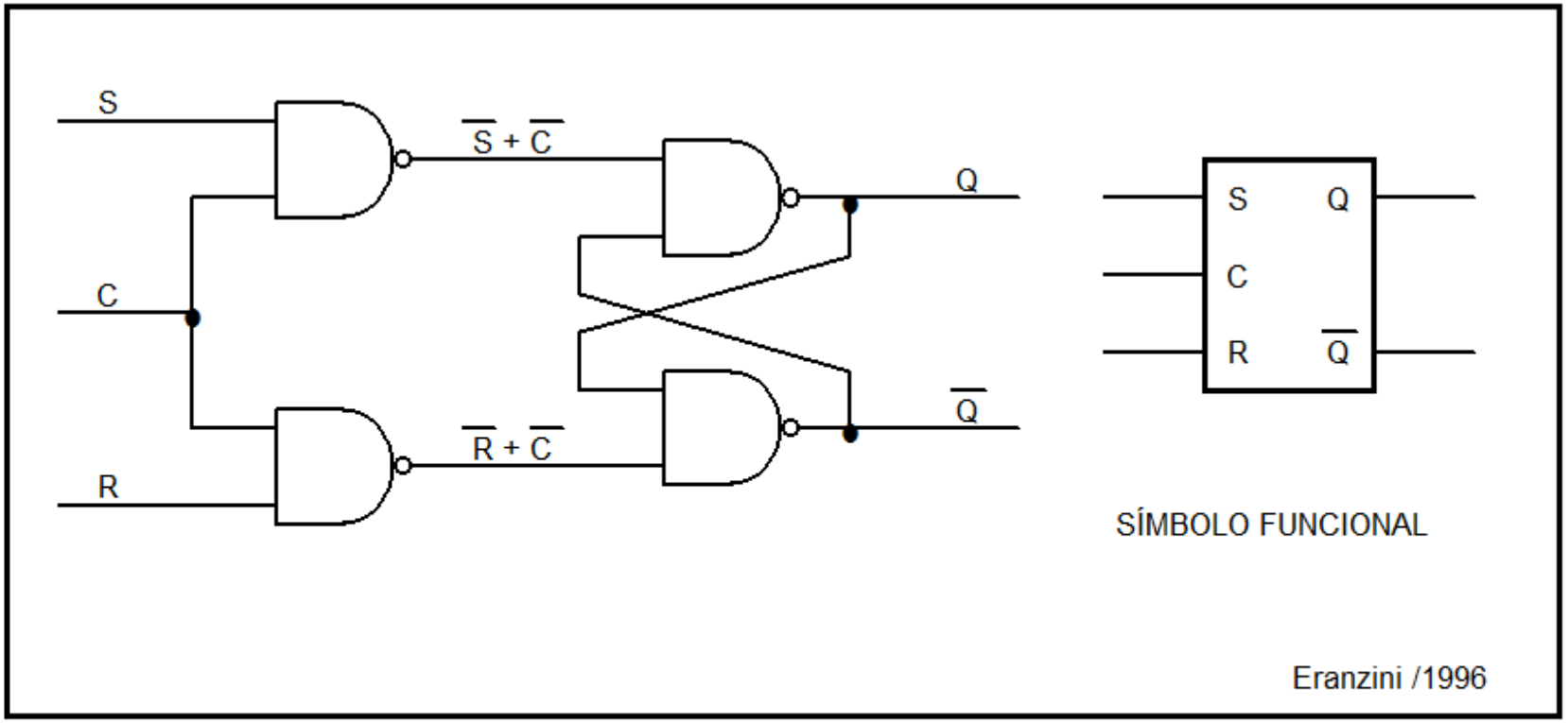
$Q(t+1)$

$$Q(t+1) = S(t).\overline{R}(t) + \overline{R}(t).Q(t)$$

– Desvantagem: também tem configuração proibida nas entradas.

4. Latch RS com Clock

- Inclui uma entrada de *Clock* ou *Enable*



4. Latch RS com Clock

$C = 0$	$\overline{S} + \overline{C} = \overline{R} + \overline{C} = 1$	Repouso na célula básica
$C = 1$	$\overline{S} + \overline{C} = \overline{S}$ $\overline{R} + \overline{C} = \overline{R}$	Funciona como Latch RS

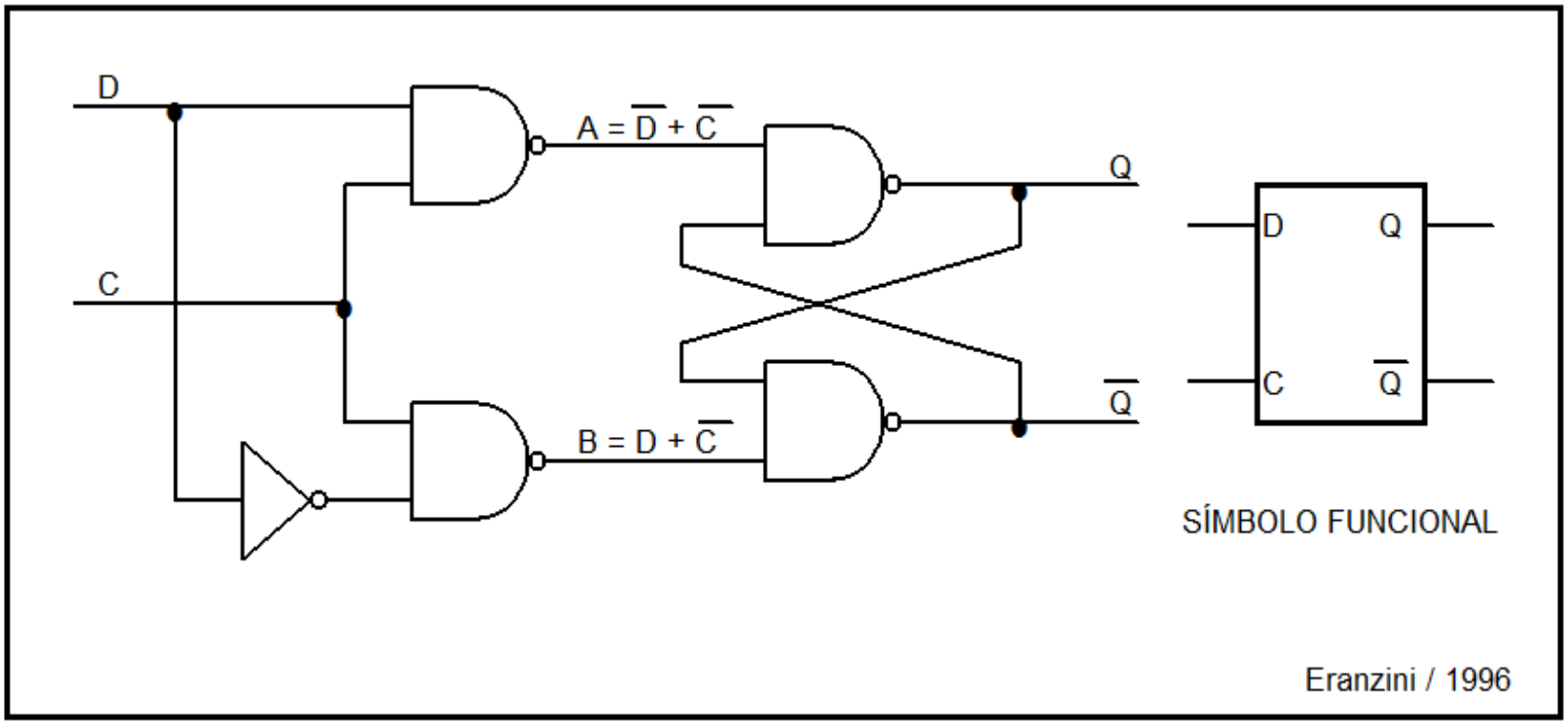
Desvantagem: quando C = 1, ainda há configuração proibida nas entradas.

4. Latch RS com clock

- Ex. Carta ou Diagrama de Tempos

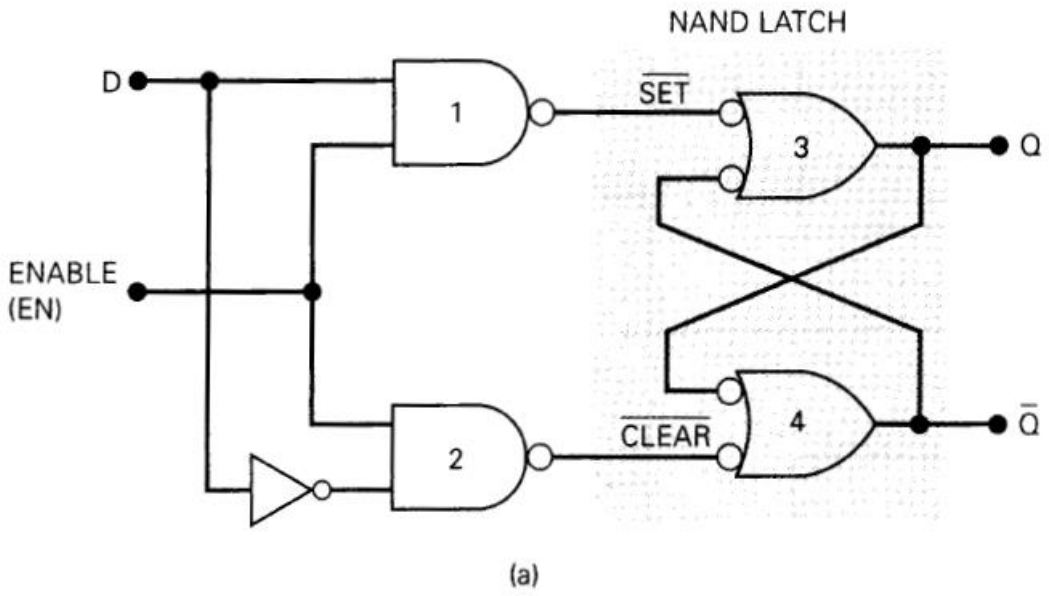
5. Latch D sensível ao Nível

- Clock ativo em 1
  → Q = D
- Clock = 0
  → Q(t) = Q(t-1)



5. Latch D sensível ao Nível

- Representação alternativa.

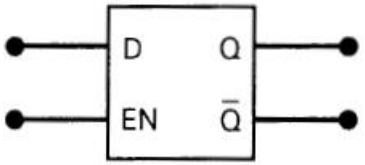


(a)

Inputs		Output
EN	D	Q
0	X	Q_0 (no change)
1	0	0
1	1	1

"X" indicates "don't care"
Q_0 is state Q just
prior to EN going LOW

(b)



(c)

5. Latch D sensível ao Nível 1

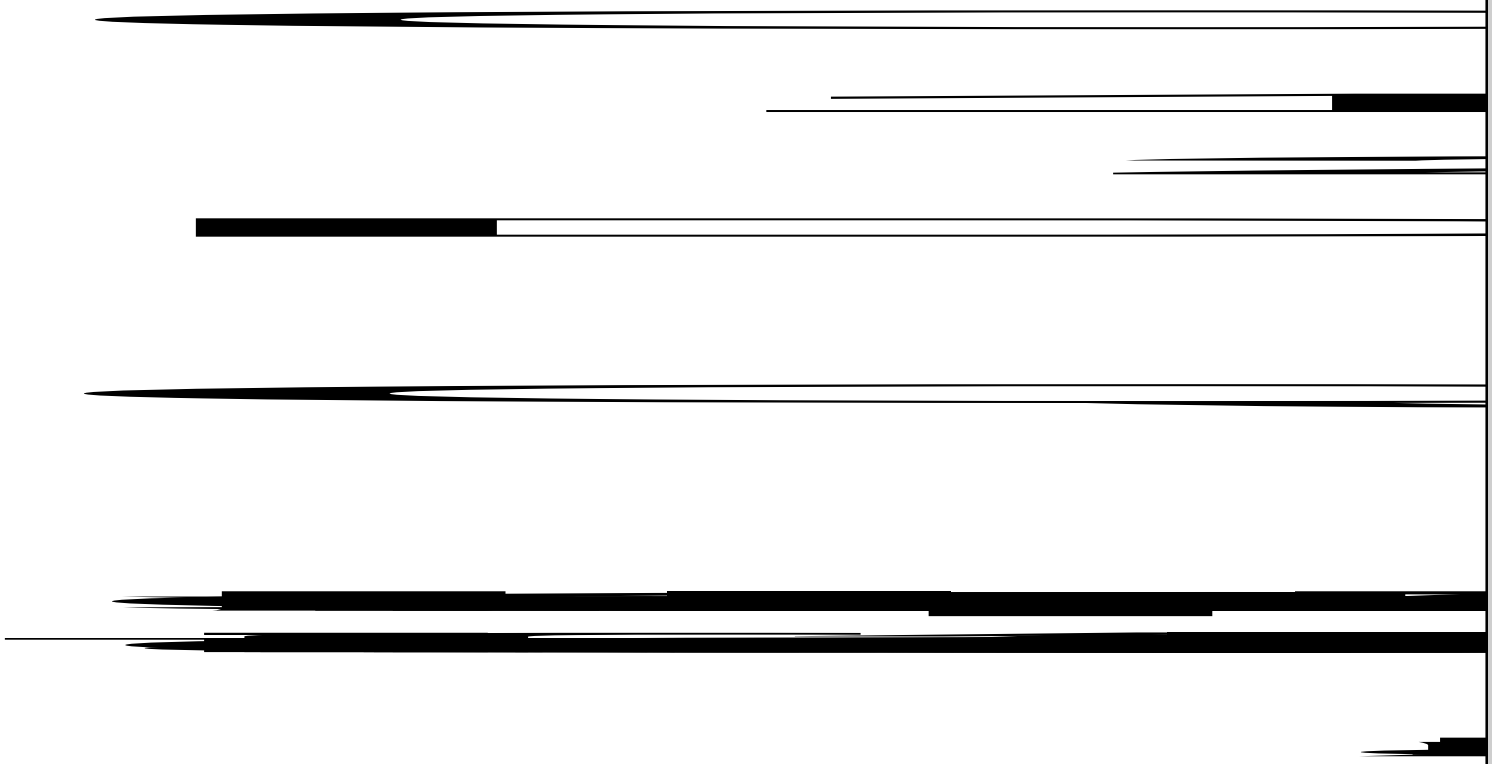
$C = 0$	$\overline{C} + \overline{D} = \overline{C} + D = 1$	Repouso na célula básica
$C = 1$	$A = \overline{C} + \overline{D} = \overline{D}$ $B = \overline{C} + D = D$	$D = 1 \rightarrow Q = 1$ $D = 0 \rightarrow Q = 0$ Copia D em Q

Não existe mais configuração proibida nas entradas

Desvantagem: não se consegue armazenar um valor preciso de D caso este se altere enquanto C = 1

5. Latch D sensível ao Nível 1

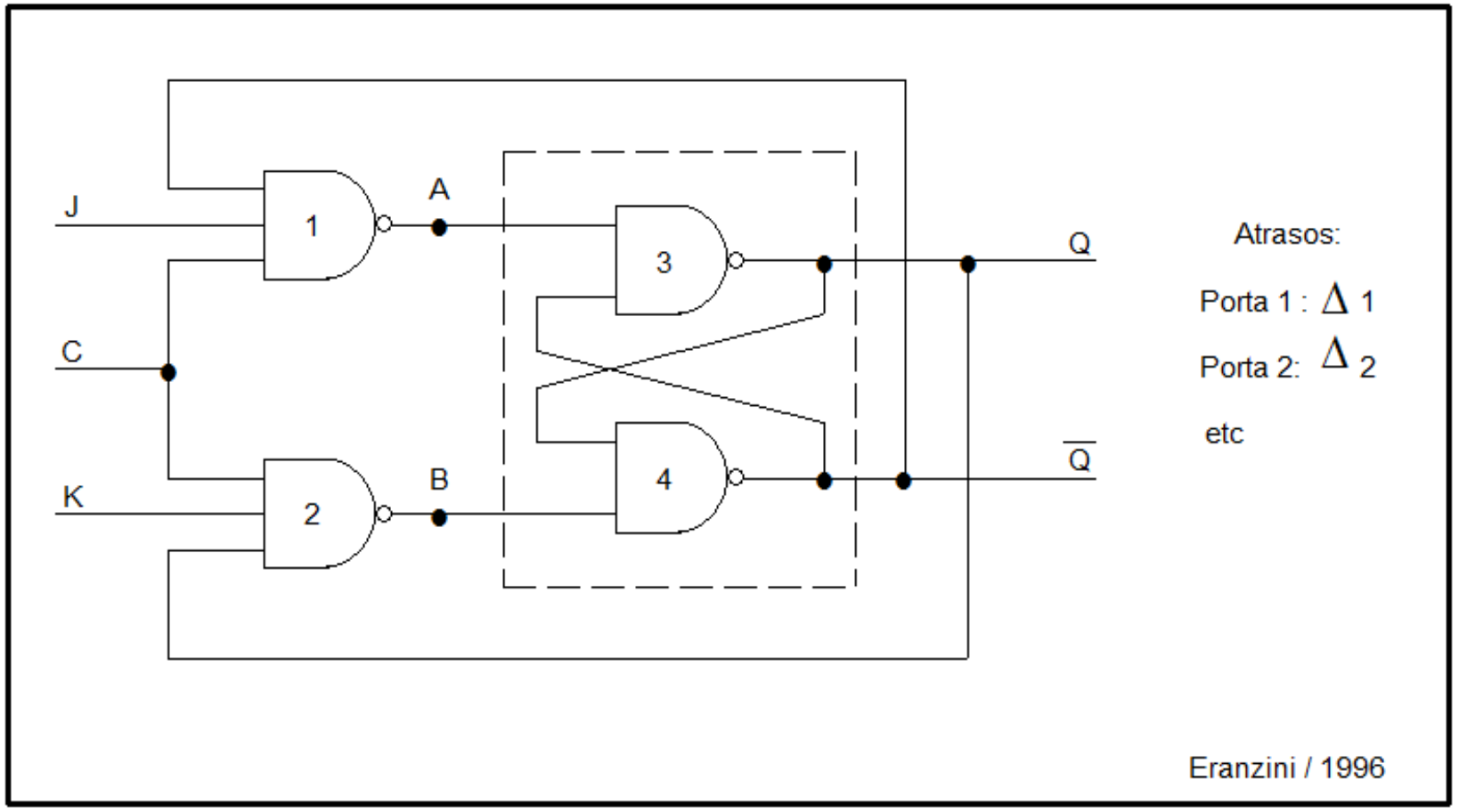
- Carta ou Diagrama de Tempos



Latch D

6. Flip-Flop J-K

- Duas entradas J e K
  - Análogas a Set e Reset



6. Flip-Flop J-K

$C = 1$	$Q(t+1) = \overline{A.\overline{Q}} = \overline{A.\overline{BQ}} = \overline{A} + BQ$ mas $A = \overline{J} + Q(t)$ $B = \overline{K} + \overline{Q}(t)$ $Q(t+1) = J\overline{Q}(t) + \overline{K}Q(t)$	
$C = 0$	$A = B = 1$	Repouso na célula básica

6. Flip-Flop J-K

J	K	Q(t)	Q(t+1)	Observação
0 0	0 0	0 1	0 1	MANTÉM
0 0	1 1	0 1	0 0	RESET
1 1	0 0	0 1	1 1	SET
1 1	1 1	0 1	1 0	INVERTE

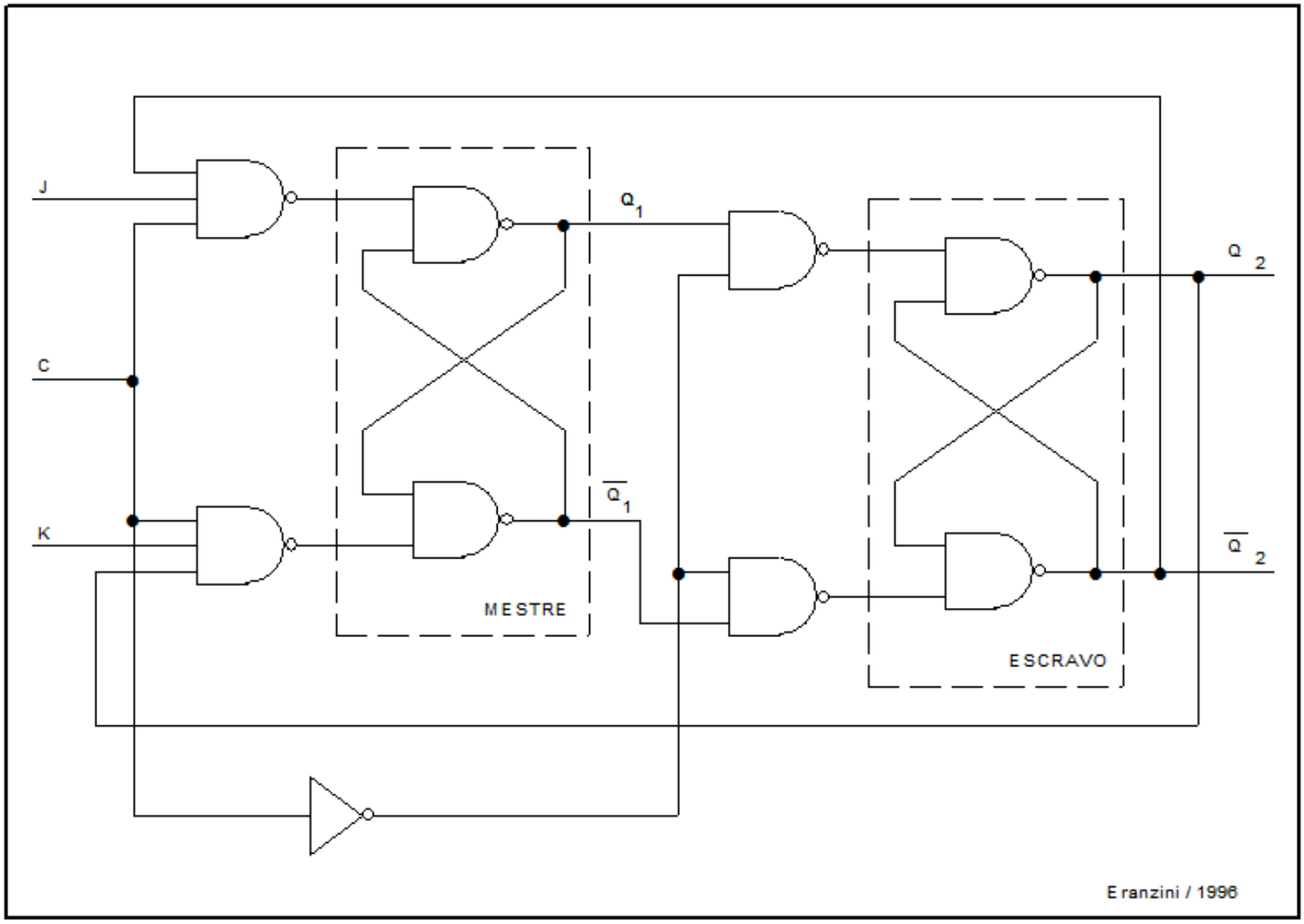
6. Flip-Flop J-K

– Aparentemente, não existe configuração proibida nas entradas, já que se J = K = 1, o *flip-flop* muda de estado

–No entanto, se o sinal C se mantiver no nível 1 durante um tempo maior do que o tempo de propagação $\Delta_1 + \Delta_3 + \Delta_4$ (ou $\Delta_2 + \Delta_3 + \Delta_4$), o *flip-flop* irá novamente mudar de estado!

<u>Desvantagem:</u> não existem valores proibidos de J e K apenas se a largura do pulso do sinal C seja suficientemente estreita

7. Flip-Flop J-K Mestre-Escravo



7. Flip-Flop J-K Mestre-Escravo

$C = 1$	Mestre Ativo Escravo Inativo	Variações de J e $\overline{K}$ alteram Q_1 e $\overline{Q_1}$
$C = 0$	Mestre Inativo Escravo Ativo	Q_1 e $\overline{Q_1}$ não se alteram e Q_2 e $\overline{Q_2}$ assumam valores que dependem de Q_1 e $\overline{Q_1}$

Não existem mais configurações proibidas nas entradas.

8. Flip-Flop's Sensíveis à Borda

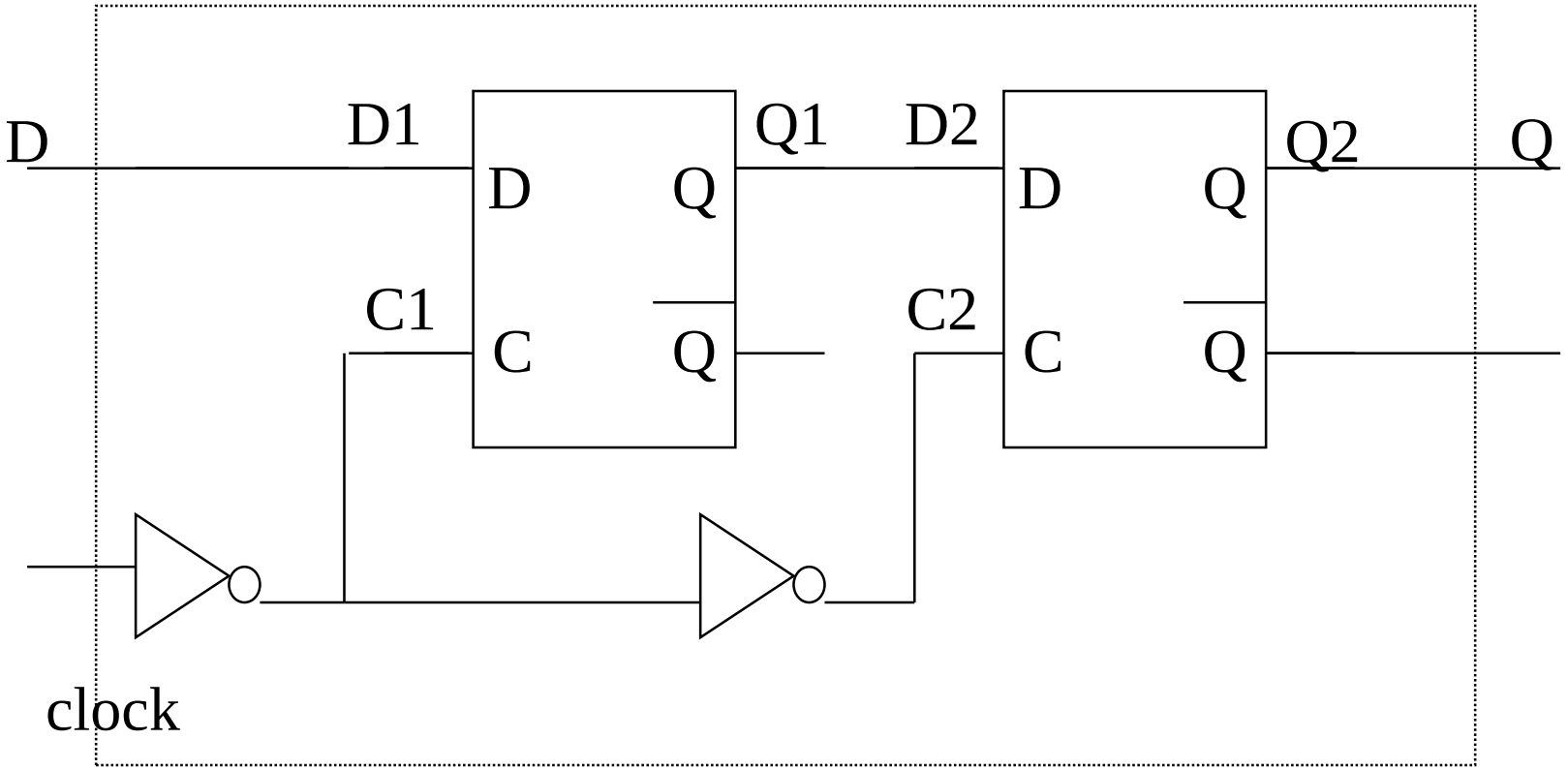
- **Problema**: ter que manter entradas constantes durante o intervalo de atuação do sinal de controle (***clock***).
- **Meta**: estabelecer um **instante preciso** para **armazenar a informação**.
- **Borda**: o instante em que um sinal digital muda de nível lógico.
  - *Borda de subida* (↑): muda de 0 para 1;
  - *Borda de descida* (↓): muda de 1 para 0.

8. Flip-Flop D Sensível à Borda de Subida

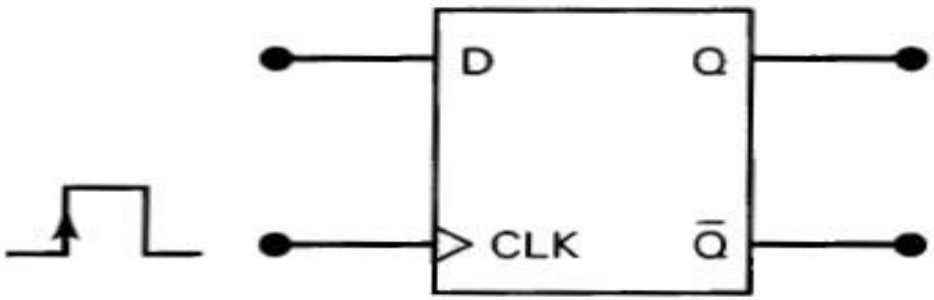
- **Ideia**: utilizar o esquema **mestre-escravo** para se obter um flip-flop tipo D sensível à borda:
  - Utilizam-se dois flip-flop's tipo D sensíveis ao nível (latches);
  - O estágio mestre atua quando *clock* = 0;
  - O estágio escravo atua quando *clock* = 1.

8.Flip-Flop D Sensível à Borda de Subida

- Mestre escravo

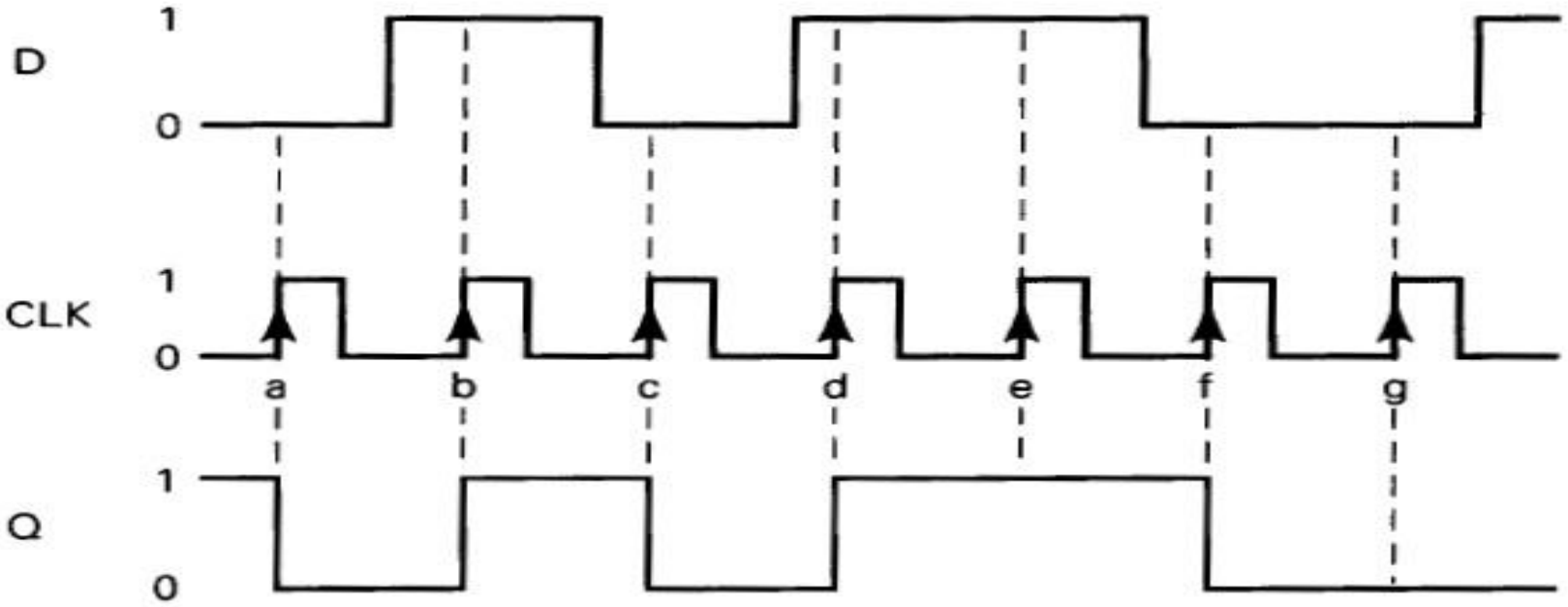


8. Flip-Flop D Sensível à Borda de Subida



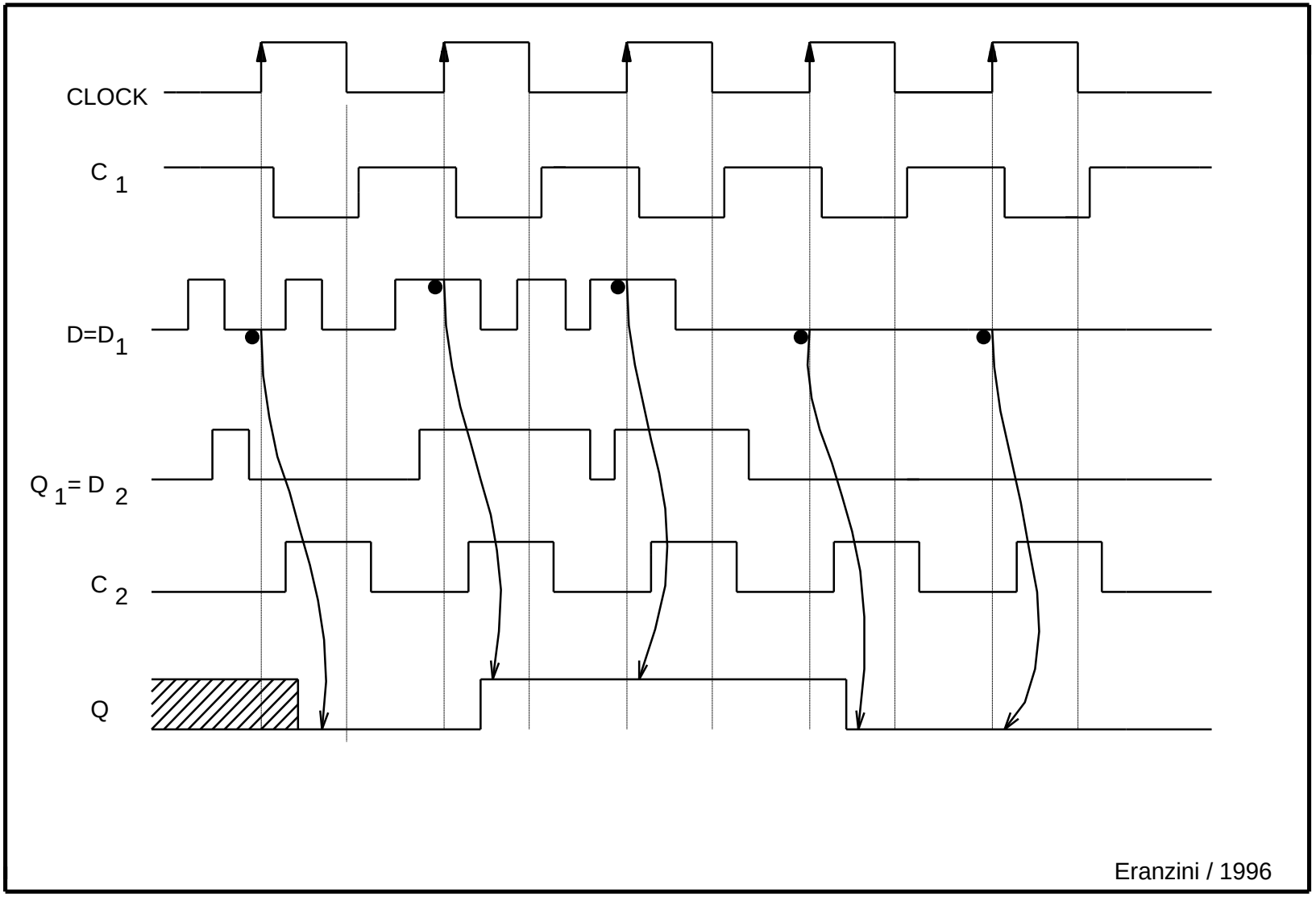
D	CLK	Q
0	↑	0
1	↑	1

(a)



(b)

8. Flip-Flop D Sensível à Borda de Subida

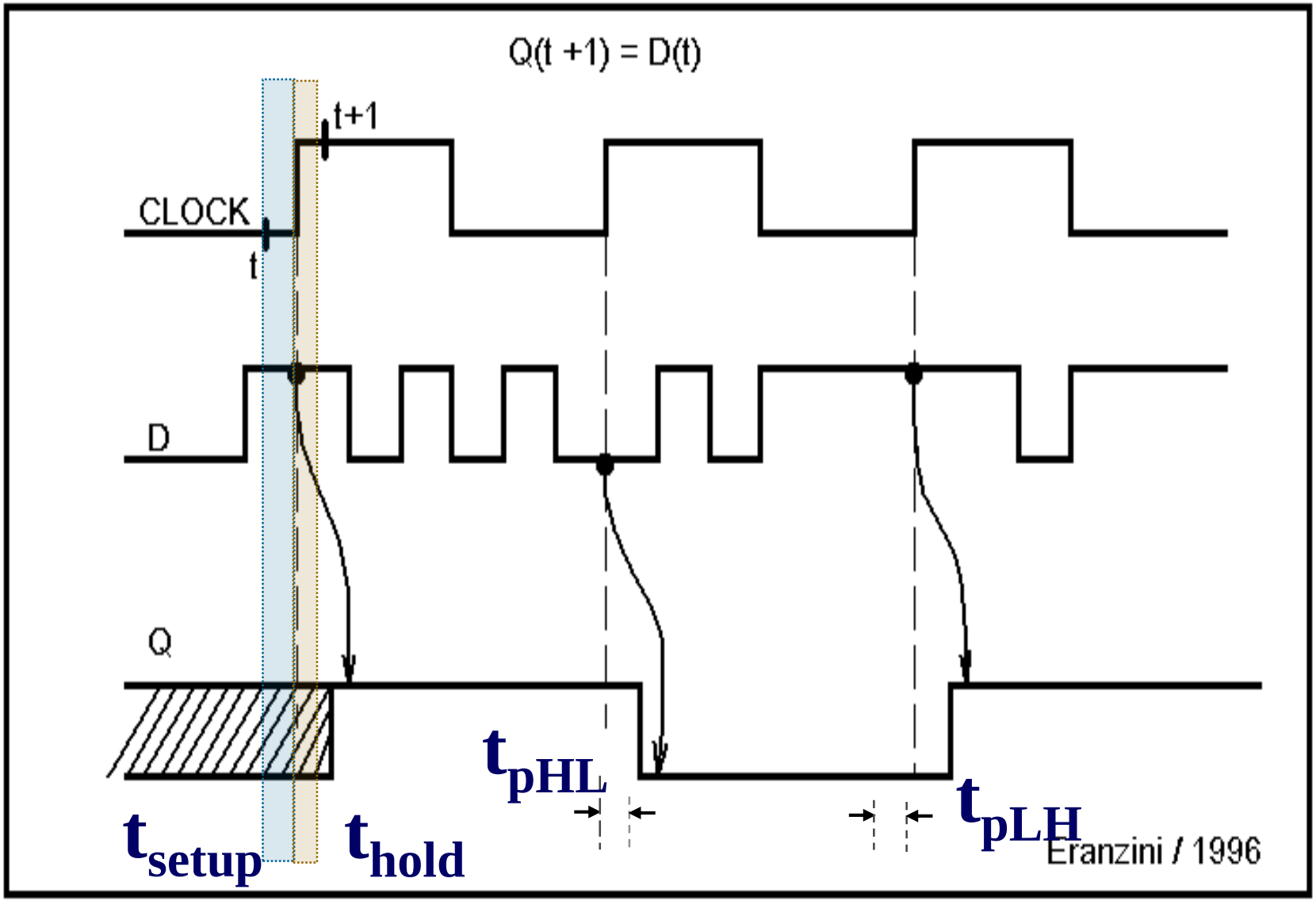


8. Flip flops sensíveis à borda

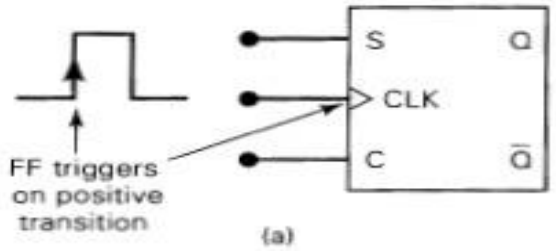
Parâmetros de temporização:

- $\mathbf{t_{pLH}}$ – Tempo de **propagação** do nível lógico 0 (***low***) para 1 (***high***);
- $\mathbf{t_{pHL}}$ – Tempo de **propagação** do nível lógico 1 (***high***) para 0 (***low***);
- $\mathbf{t_{setup}}$ $(\mathbf{t_s})$ – Tempo que os sinais de controle devem se manter **estáveis antes** da borda do ***clock***;
- $\mathbf{t_{hold}}$ $(\mathbf{t_h})$ – Tempo que os sinais de controle devem se manter **estáveis após** a borda do ***clock***;

8. Flip flops sensíveis à borda



8. Flip-Flop RS sensível à borda

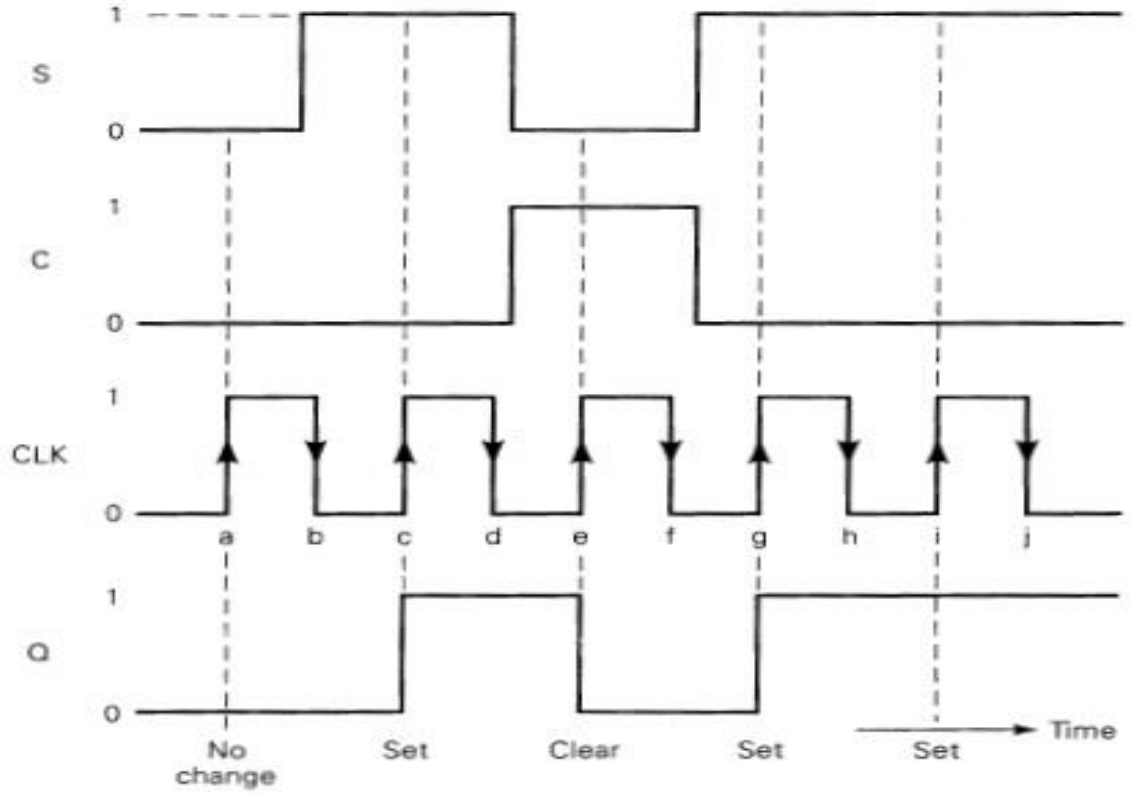


(a)

Inputs			Output
S	C	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	Ambiguous

Q_0 is output level prior to ↑ of CLK.
↓ of CLK produces no change in Q.

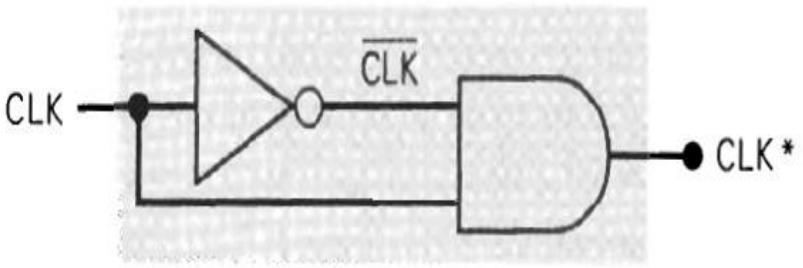
(b)

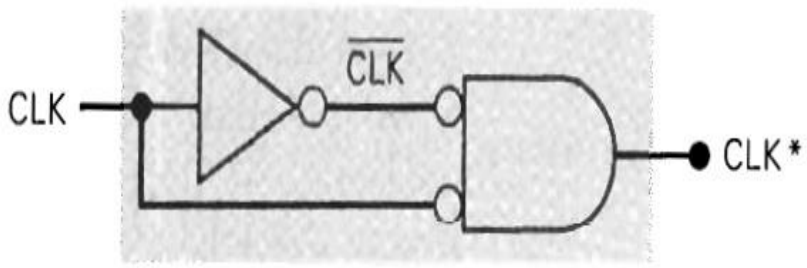


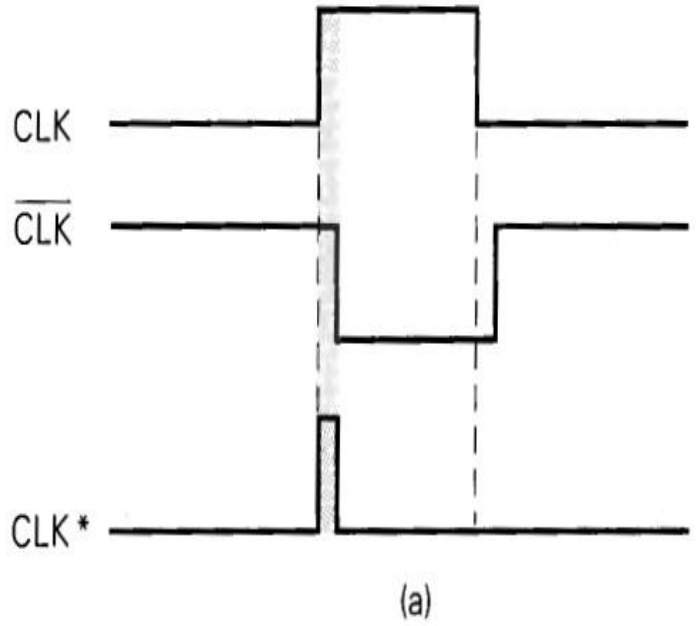
(c)

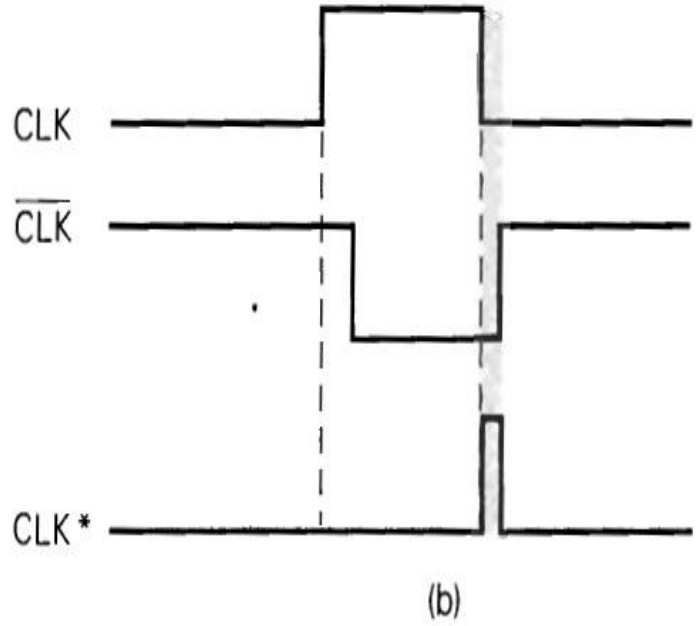
8. Flip-Flop RS sensível à borda

- **Detector de Borda** – Circuito auxiliar .

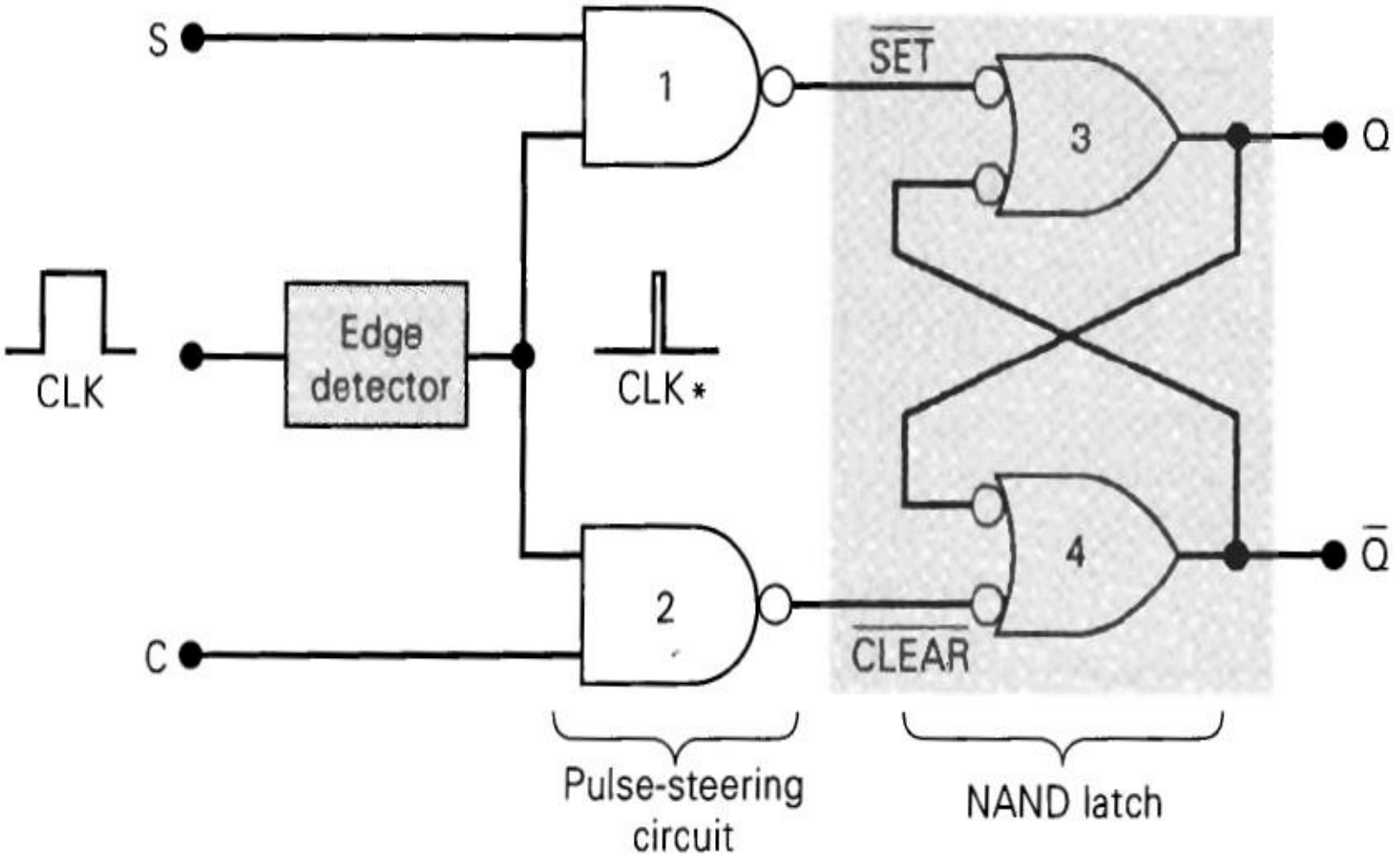




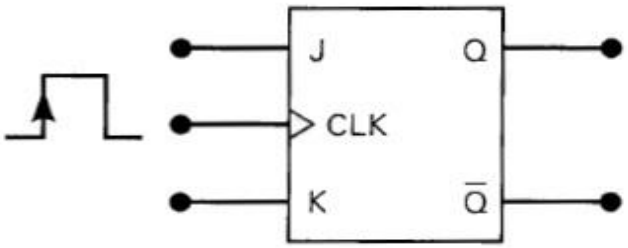




8. Flip-Flop RS sensível à borda

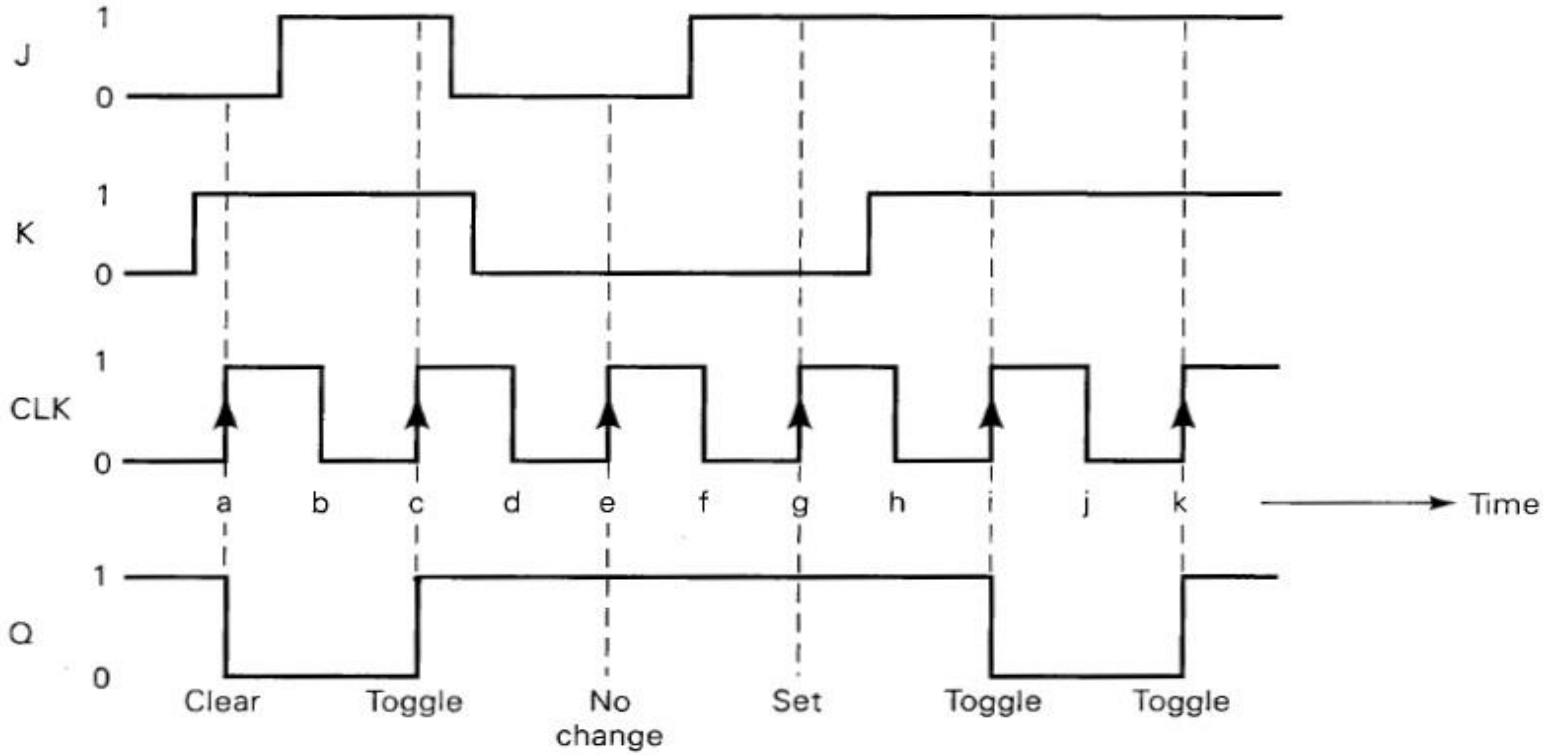


8. Flip-Flop JK sensível à borda



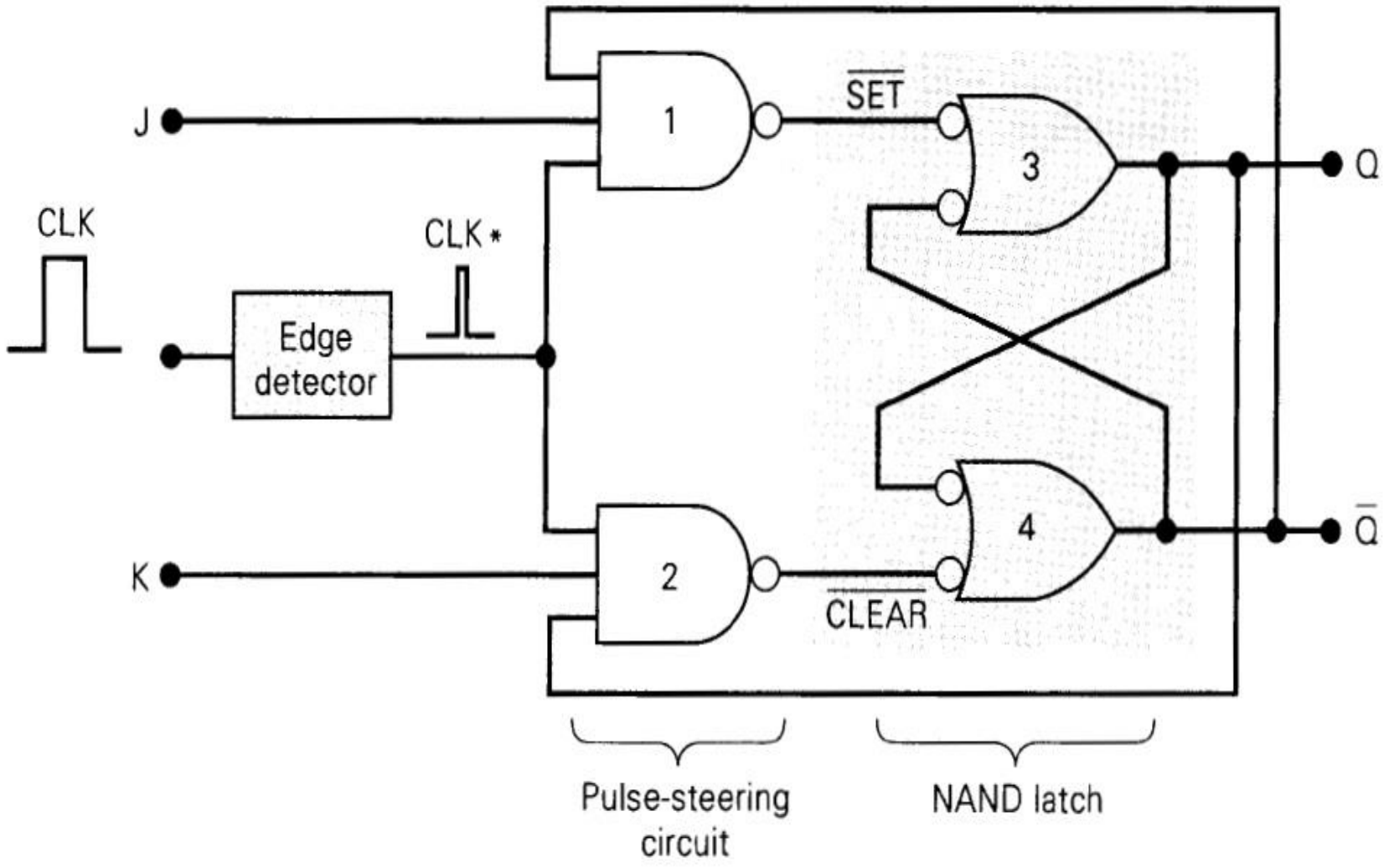
J	K	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	$\overline{Q_0}$ (toggles)

(a)

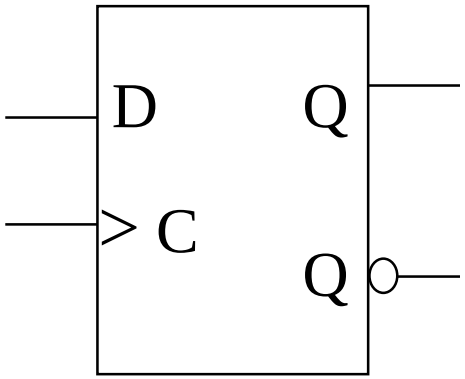


(b)

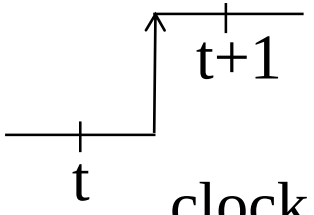
8. Flip-Flop JK sensível à borda

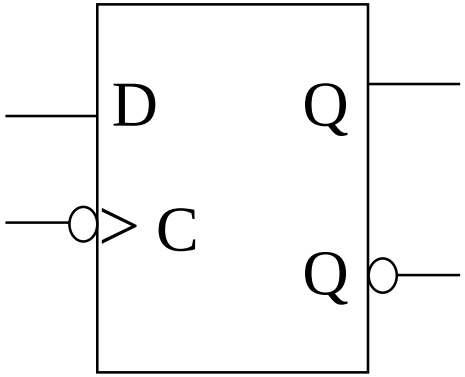


9. Representação de Flip-Flop's Sensíveis à Borda

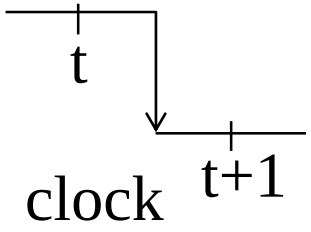


D(t)	C	Q(t+1)
0	↑	0
1	↑	1





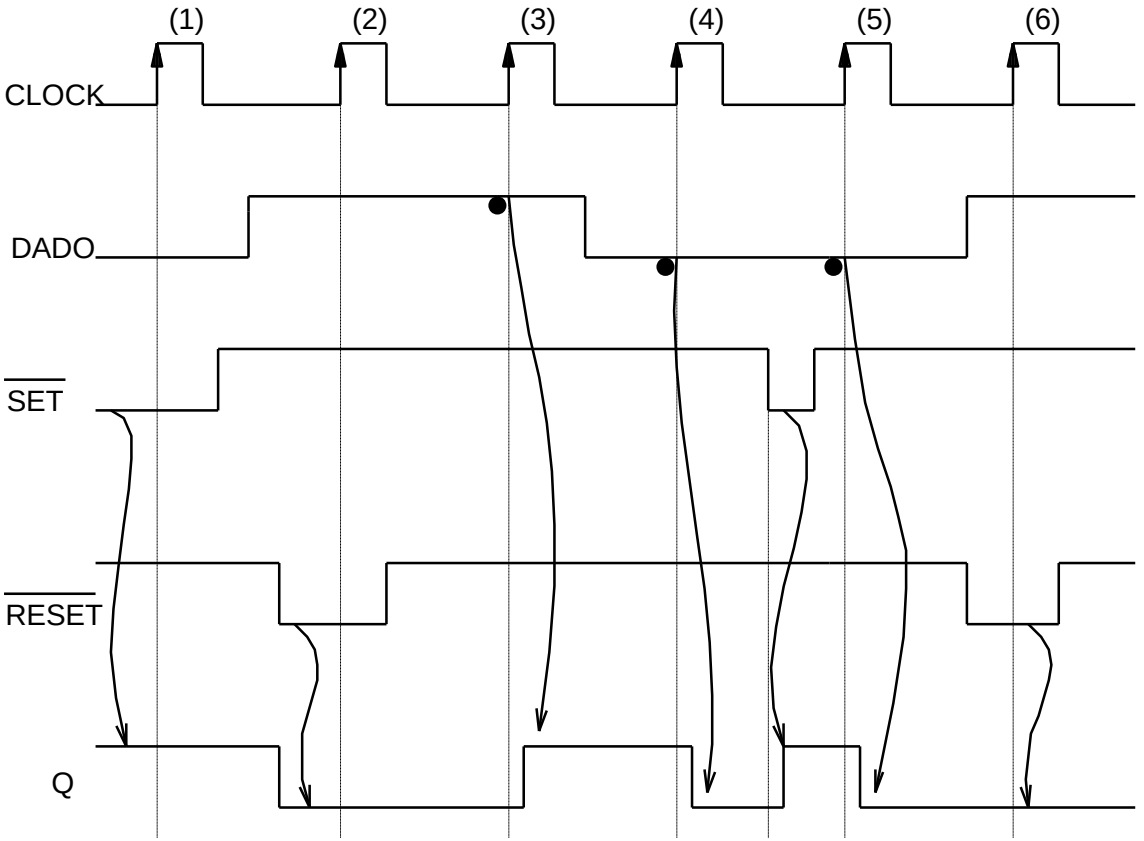
D(t)	C	Q(t+1)
0	↓	0
1	↓	1

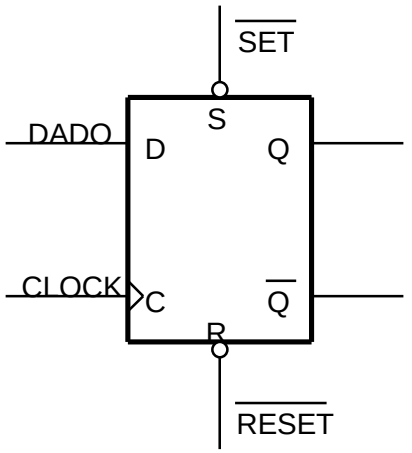


10.FF tipo D com entradas assíncronas

- Nos flip-flop's sensíveis à borda, é comum introduzir entradas do tipo set-reset (ou preset e clear) **assíncronas**, que atuam sobre a saída independentemente do sinal do *clock*.
- Tais entradas servem para impor **condições iniciais** aos flip-flops, independente do sinal de *clock*

10. FF tipo D com entradas assíncronas



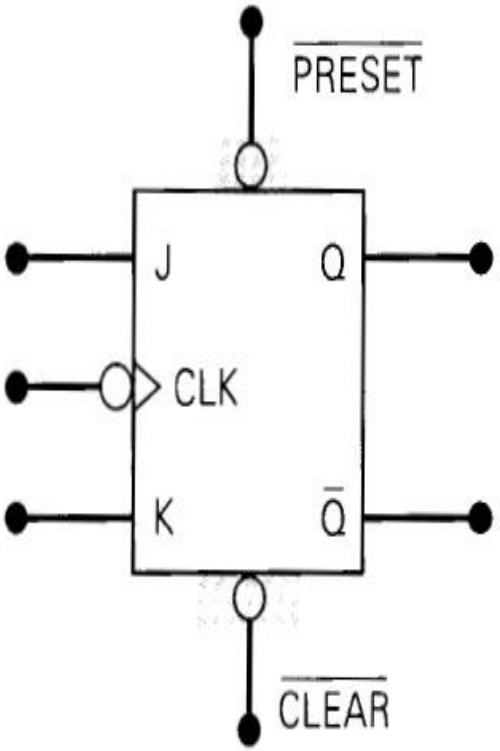


Borda (1) : CLOCK não atua pois $\overline{SET}$ acionado

Bordas (2) e (6) : CLOCK não atua pois $\overline{RESET}$ acionado

Eranzini / 1996

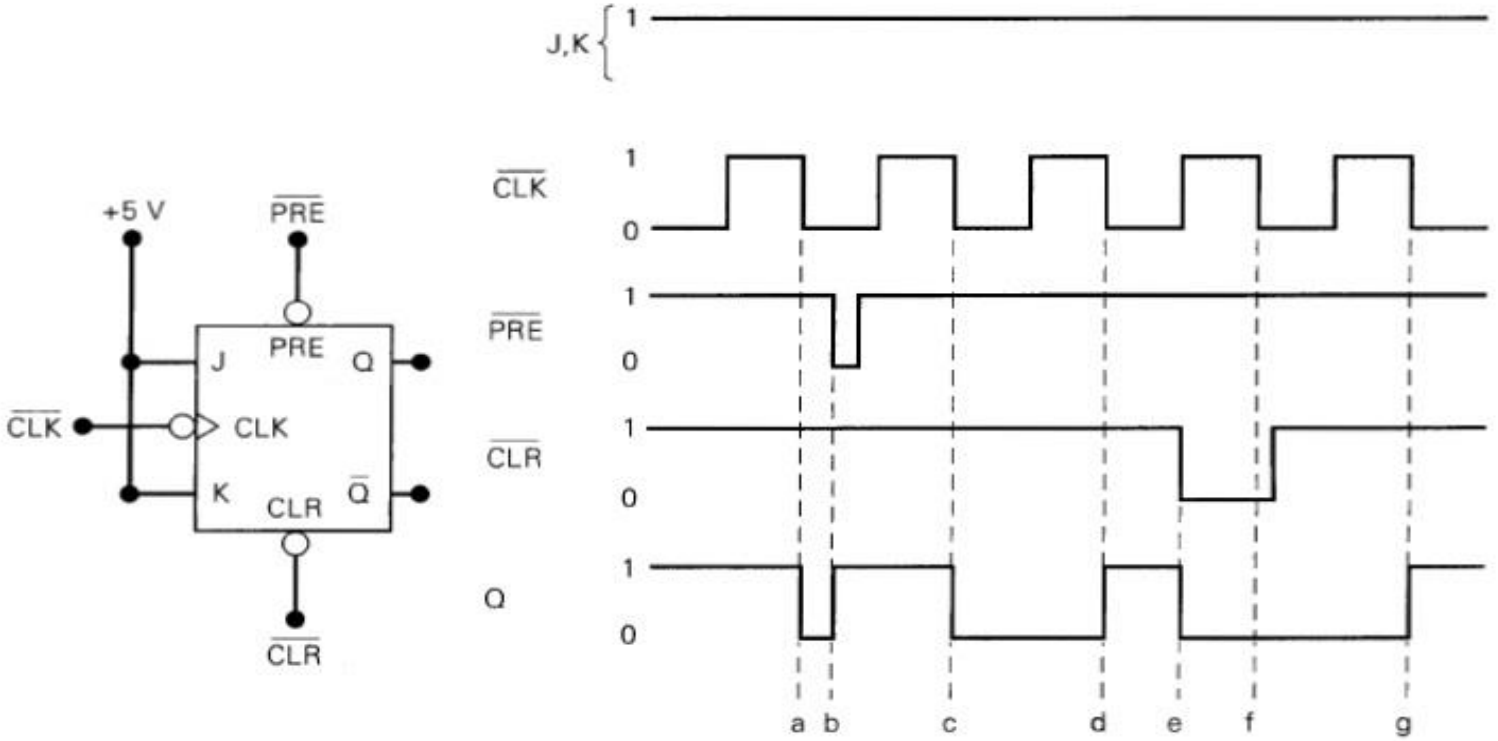
10. FF tipo JK com entradas assíncronas



$\overline{\text{PRESET}}$	$\overline{\text{CLEAR}}$	FF response
1	1	Clocked operation*
0	1	Q = 1 (regardless of CLK)
1	0	Q = 0 (regardless of CLK)
0	0	Not used

*Q will respond to J, K, and CLK

10. FF tipo JK com entradas assíncronas



(a)

Point	Operation
a	Synchronous toggle on NGT of $\overline{CLK}$
b	Asynchronous set on $\overline{PRE} = 0$
c	Synchronous toggle
d	Synchronous toggle
e	Asynchronous clear on $\overline{CLR} = 0$
f	$\overline{CLR}$ over-rides the NGT of $\overline{CLK}$
g	Synchronous toggle

(b)

11. Tabela Funcional por Tipo de Flip-Flop

Flip-Flop tipo T

Símbolo funcional

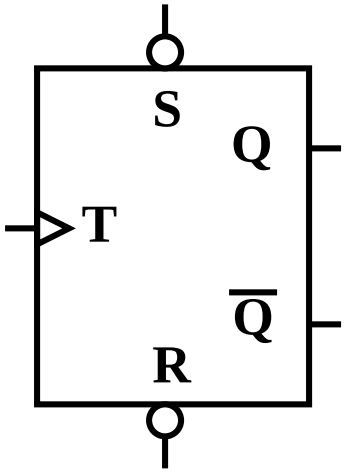


Tabela funcional

Entradas	Saídas	
T	Q	$\overline{Q}$
↑	$\overline{Q}$	Q

11. Tabela Funcional por Tipo de Flip-Flop

Flip-Flop T com Enable

Símbolo funcional

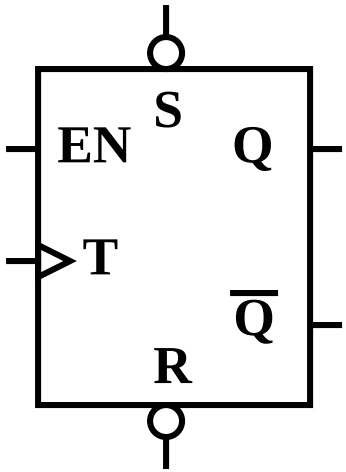


Tabela funcional

Entradas		Saídas	
EN	T	Q	$\overline{Q}$
0	↑	Q	$\overline{Q}$
1	↑	$\overline{Q}$	Q

11. Tabela Funcional por Tipo de Flip-Flop

Flip-Flop tipo D – Sensível à borda de subida do *clock*

Símbolo funcional

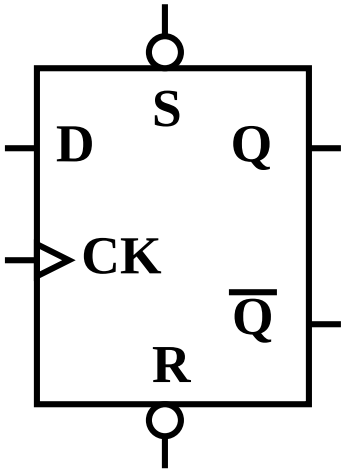


Tabela funcional

Entradas		Saídas	
D	Ck	Q	$\overline{Q}$
0	↑	0	1
1	↑	1	0

11. Tabela Funcional por Tipo de Flip-Flop

Flip-Flop tipo D – Sensível à borda de descida do *clock*

Símbolo funcional

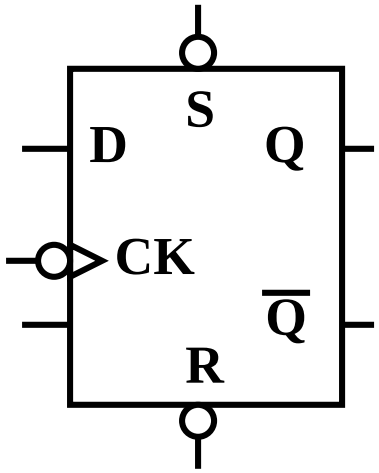


Tabela funcional

Entradas		Saídas	
D	Ck	Q	$\overline{Q}$
0	↓	0	1
1	↓	1	0

11. Tabela Funcional por Tipo de Flip-Flop

Flip-Flop tipo JK – Sensível à borda de subida do *clock*

Símbolo funcional

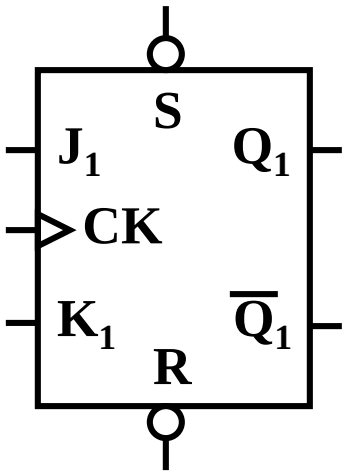


Tabela funcional

Entradas			Saídas	
J	K	Ck	Q	$\overline{Q}$
0	0	↑	Q	$\overline{Q}$
0	1	↑	0	1
1	0	↑	1	0
1	1	↑	$\overline{Q}$	Q

11. Tabela Funcional por Tipo de Flip-Flop

Flip-Flop tipo JK – Sensível à borda de descida do *clock*

Símbolo funcional

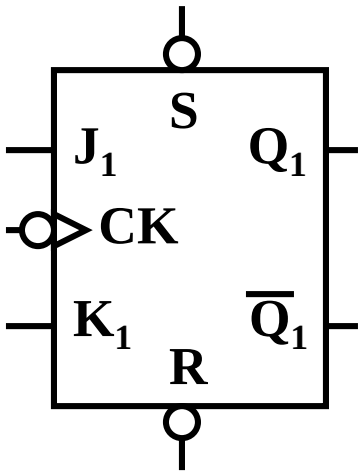
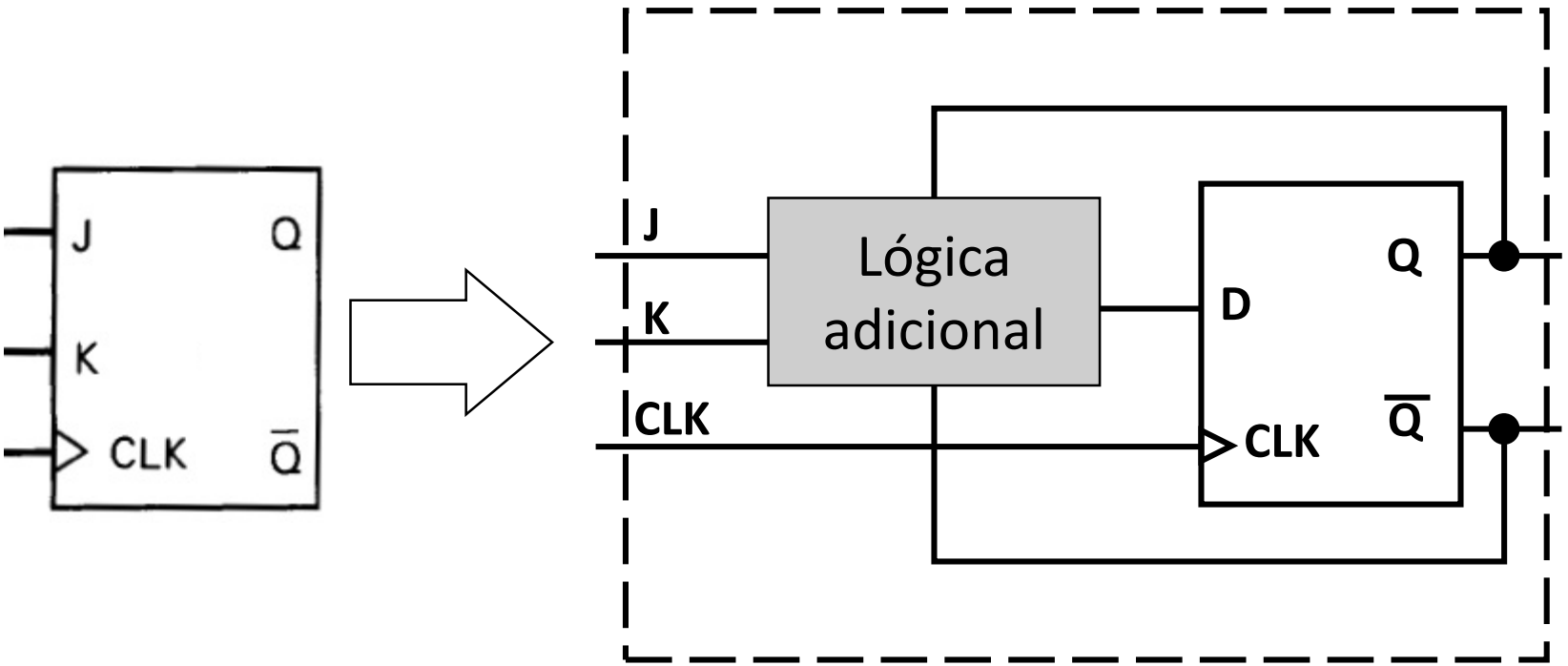


Tabela funcional

Entradas			Saídas	
J	K	Ck	Q	$\overline{Q}$
0	0	↓	Q	$\overline{Q}$
0	1	↓	0	1
1	0	↓	1	0
1	1	↓	$\overline{Q}$	Q

12. Conversão de Flip-Flops

- É possível converter um Flip-Flop (base) em outro (alvo) usando alguma lógica adicional
  - Exemplo: Flip-Flop D → Flip-Flop JK

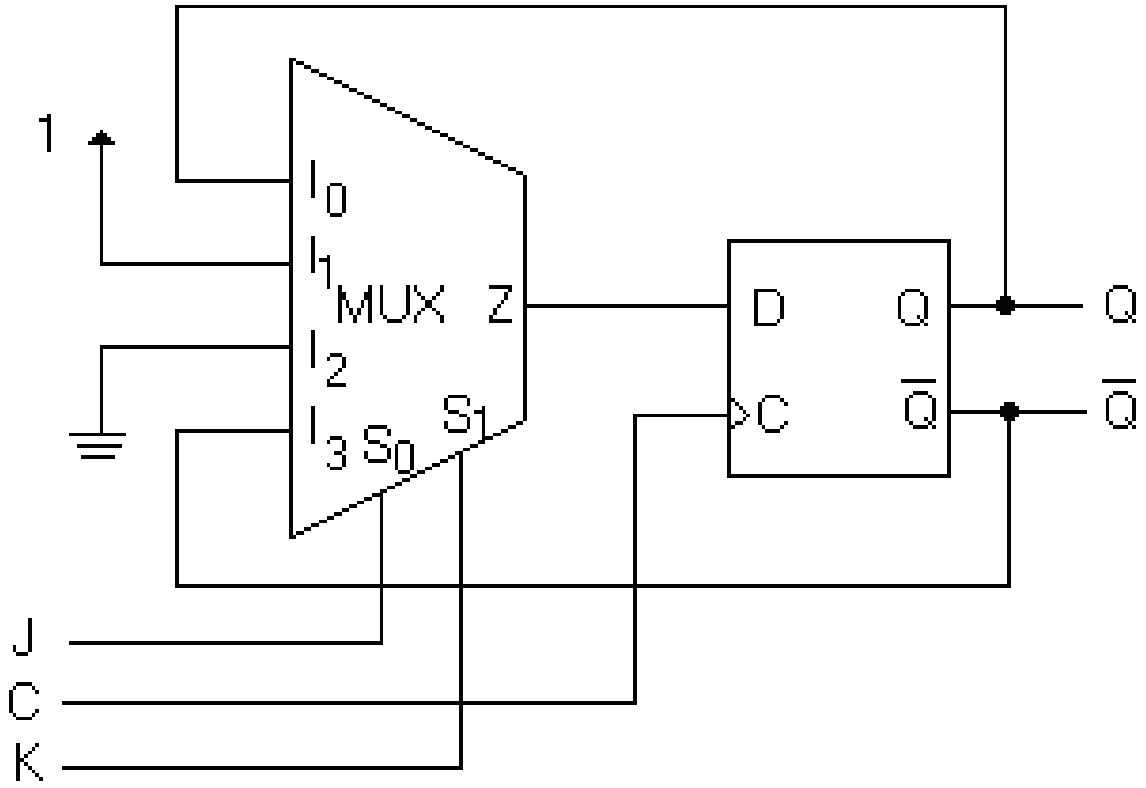


12. Conversão de Flip-Flops

- Algumas estratégias podem ajudar a definir a lógica adicional necessária.
- Exemplos
  - Definir, para cada configuração das entradas de controle do FF alvo e da saída Q, qual deve ser o valor das entradas de controle do FF base, obtendo uma tabela verdade ou expressões Booleanas.
  - Então ligar os fios, ou usar mapa de Karnaugh para soma (produto) mínima ou um multiplexador.

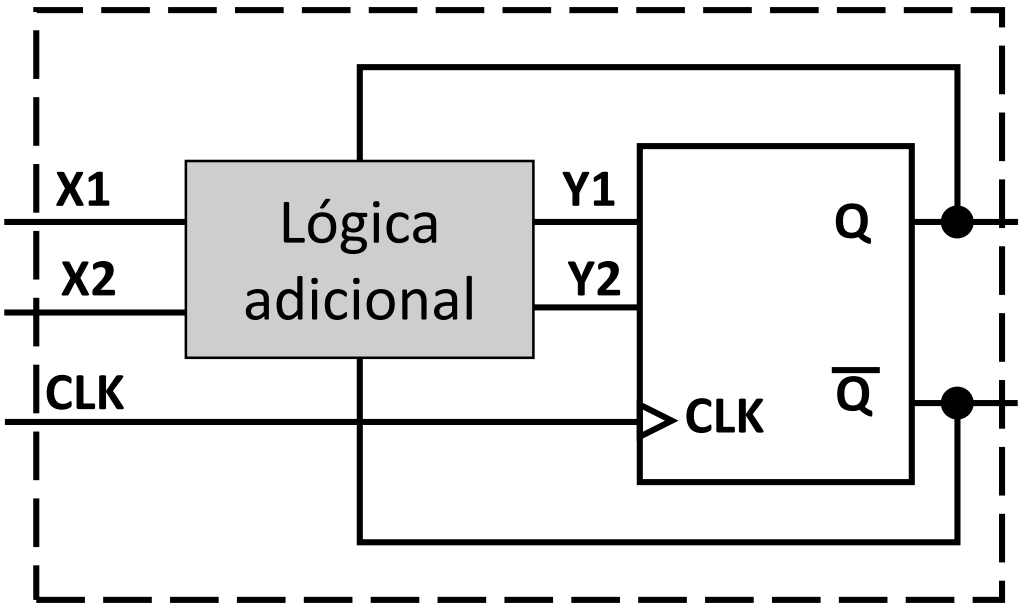
12. Conversão de Flip-Flops

- Exemplo
  - Flip-Flop tipo D → Flip-Flop tipo JK, utilizando apenas um MUX 4x1



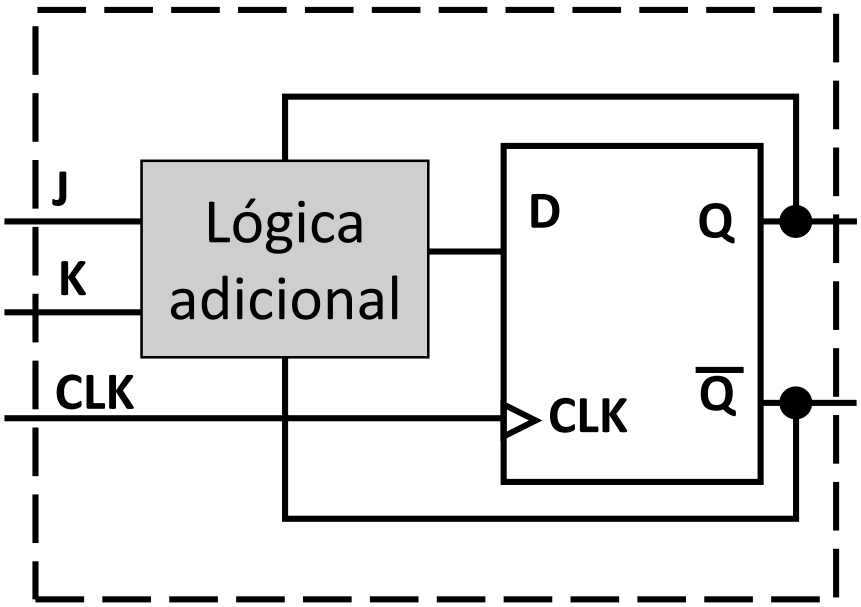
12. Conversão de Flip-Flops

- Circuito Mínimo: mapa de Karnaugh
  - Determinar lógica adicional em função do valor a ser alimentado em Y1, Y2 dados os valores de X1, X2 e Q



12. Conversão de Flip-Flops

- Método sistemático: mapa de Karnaugh para definir lógica adicional
  - Exemplo 1: Flip-Flop tipo D → Flip-Flop tipo JK
    - » Entradas: J, K, Q; Saída: D

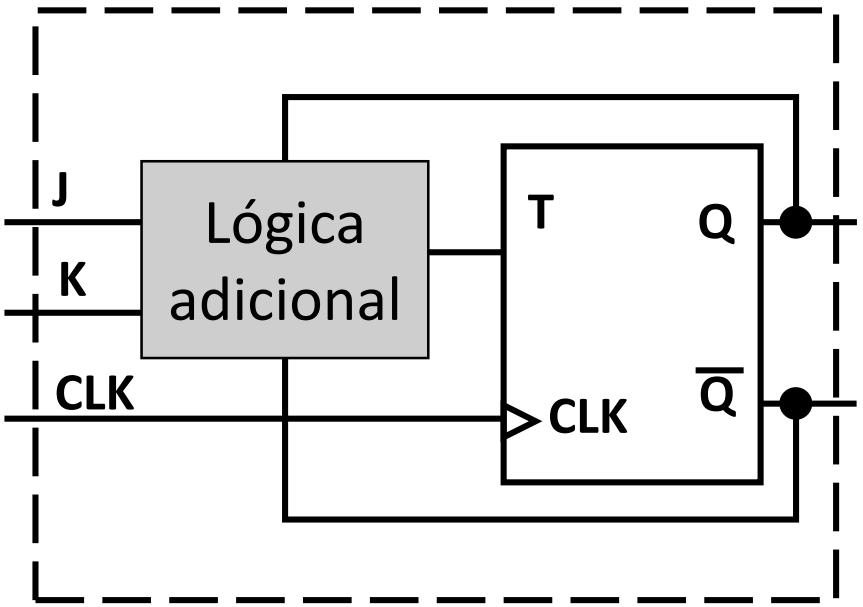


Q \ JK	00	01	11	10
0	**0**	**0**	**1**	**1**
1	**1**	**0**	**0**	**1**

Lógica adicional em D:
$J\overline{Q} + \overline{K}Q$

12. Conversão de Flip-Flops

- Método sistemático: mapa de Karnaugh para definir lógica adicional
  - Exemplo 2: Flip-Flop tipo T → Flip-Flop tipo JK
    - » Entradas: J, K, Q; Saída: T

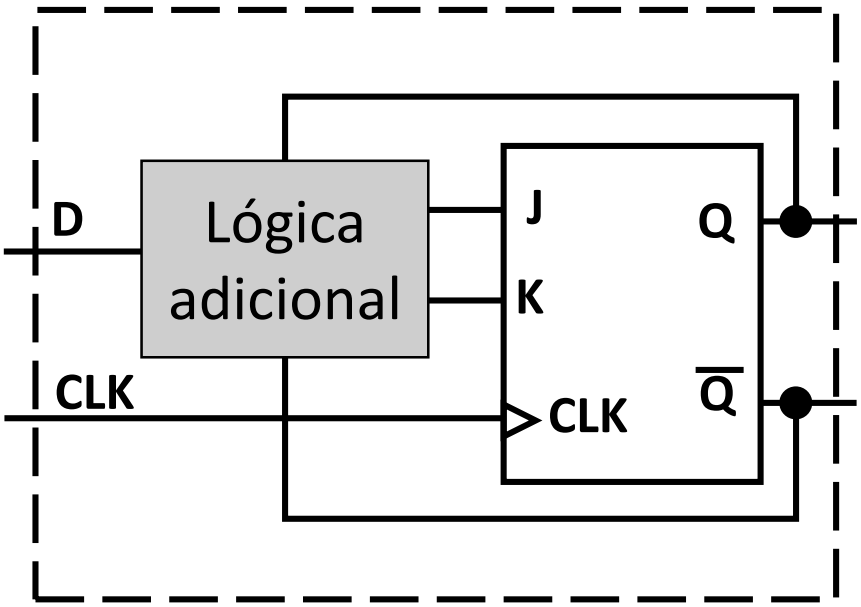


Q \ JK	00	01	11	10
0	**0**	**0**	**1**	**1**
1	**0**	**1**	**1**	**0**

Lógica adicional em T:
$J\overline{Q} + KQ$

12. Conversão de Flip-Flops

- Método sistemático: mapa de Karnaugh para definir lógica adicional
  - Exemplo 3: Flip-Flop tipo JK → Flip-Flop tipo D
    - » Entradas: D, Q; Saídas: J, K



Q \ D	0	1
0	**0x**	**1x**
1	**x1**	**x0**

Lógica adicional em JK:

J=D ; K=$\overline{D}$

Lição de Casa

- Leitura Sugerida:
  - Capítulo 7 do Livro Texto (itens 7.1, 7.2).
- Exercícios Sugeridos:
  - Capítulo 7 do Livro Texto (FFs).

Livro Texto

- Wakerly, J.F.; *Digital Design – Principles & Practices;* Fourth Edition, ISBN: 0-13-186389-4, Pearson & Prentice-Hall, Upper Saddle, River, New Jersey, 07458, 2006.