Álgebra Booleana e Circuitos Digitais

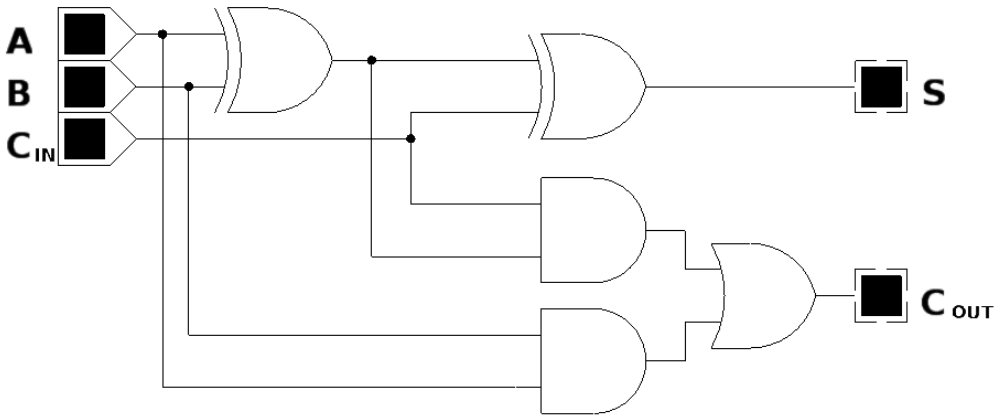
Revisão para Prova 2

Conteúdo

- Somadores, subtratores e comparadores
  - somador completo (tabela verdade e circuito)
  - subtrator completo (tabela verdade e circuito)
  - subtração em complemento de 2
  - comparador de igualdade e de magnitude (tabela verdade e circuito)
- Multiplexadores e demultiplexadores
  - tabela verdade e circuito
  - ampliação de MUX e DEMUX
  - multiplexador como gerador de funções lógicas
  - bit de paridade
- Latches e flip-flops
  - circuito combinacional vs sequencial
  - diferenças entre latches e flip-flops
  - biestáveis tipos SR, D, JK e T (tabela verdade e diagramas de tempo, somente – não cai circuito!)
  - sincronismo (níveis alto e baixo, borda de subida e descida)
  - entradas Preset e Clear
- Registradores e contadores
  - tipos de registradores
  - shift register (multiplicação e divisão)
  - contadores assíncronos e síncronos (de 0 a n, crescentes ou decrescentes)
  - contadores de década
- Máquinas de estado
  - contadores síncronos (qualquer sequencia)

Somador completo - tabela verdade e circuito lógico

A	B	C_{IN}	S	C_{OUT}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



$$S = \bar{A}\bar{B}C_{IN} + \bar{A}B\overline{C_{IN}} + A\bar{B}\overline{C_{IN}} + ABC_{IN} = A \oplus B \oplus C_{IN}$$

$$C_{OUT} = \bar{A}BC_{IN} + A\bar{B}C_{IN} + AB\overline{C_{IN}} + ABC_{IN} = C_{IN}(A \oplus B) + AB$$

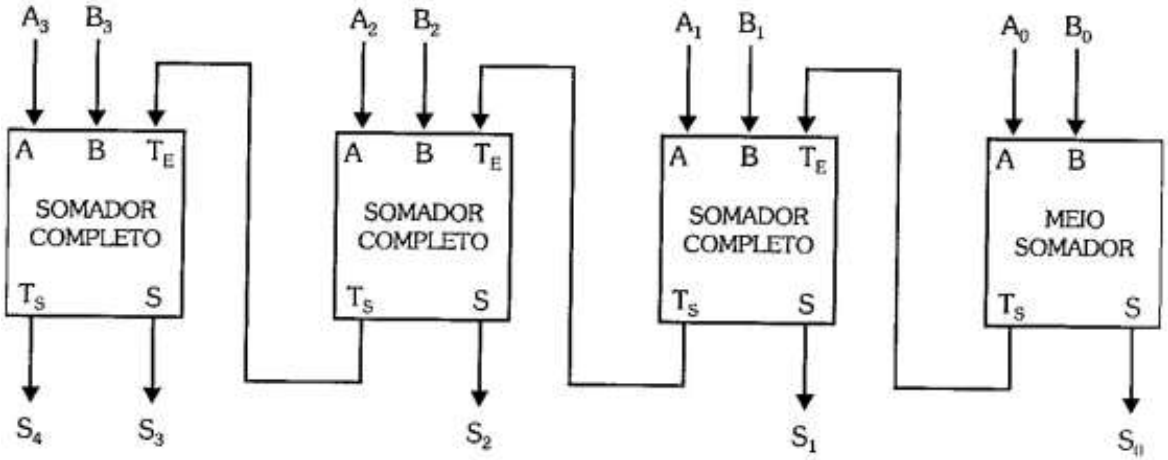
Fonte: microcontrollerslab.com

Somador completo paralelo de 4 bits

Soma binário completo paralelo de 4 bits:

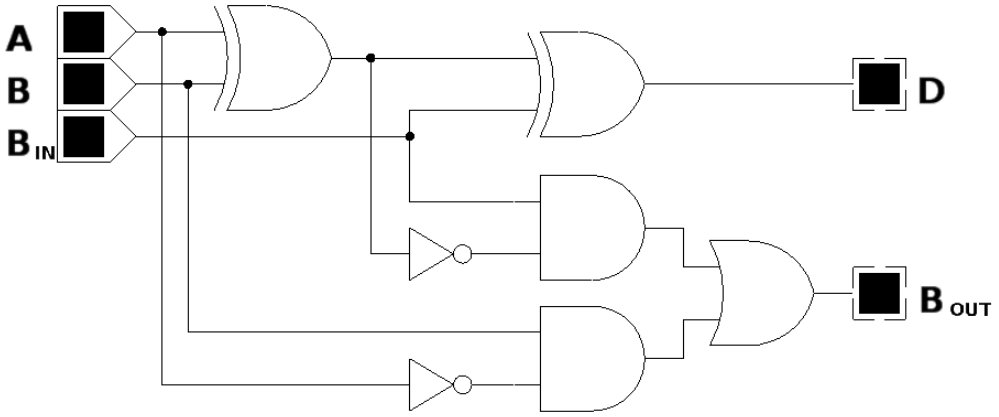
$$\begin{array}{cccccc} & A_3 & A_2 & A_1 & A_0 \\ + & B_3 & B_2 & B_1 & B_0 \\ \hline S_4 & S_3 & S_2 & S_1 & S_0 \end{array}$$

O somador paralelo completo de 4 bits:



Subtrator completo - tabela verdade e circuito lógico

A	B	B_{IN}	D	B_{OUT}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



$$D = \bar{A}\bar{B}B_{IN} + \bar{A}B\overline{B_{IN}} + A\bar{B}\overline{B_{IN}} + ABB_{IN} = A \oplus B \oplus B_{IN}$$

$$B_{OUT} = \bar{A}\bar{B}B_{IN} + \bar{A}B\overline{B_{IN}} + \bar{A}BB_{IN} + ABB_{IN} = \bar{A}B_{IN} + \bar{A}B + BB_{IN} = \bar{A}B + B_{IN}\left(\overline{A \oplus B}\right)$$

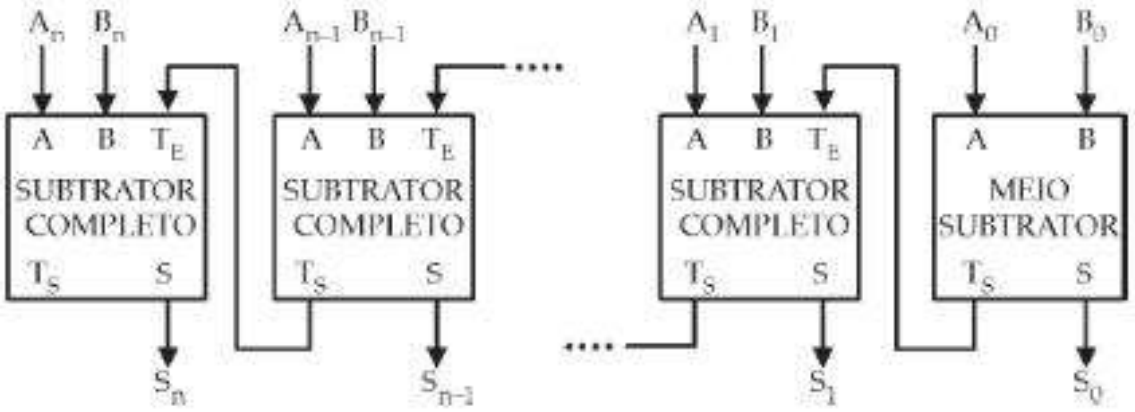
Fonte: de-iitr.vlabs.ac.in

Subtrator completo paralelo de 4 bits

Subtrator completo paralelo binário de 4 bits:

$$\begin{array}{cccc} A_3 & A_2 & A_1 & A_0 \\ B_3 & B_2 & B_1 & B_0 \\ \hline S_3 & S_2 & S_1 & S_0 \end{array}$$

O subtrator paralelo completo de 4 bits:



Subtrator em complemento de 2

Subtrai um número binário (subtraendo) de outro número binário (minuendo) seguindo os passos:

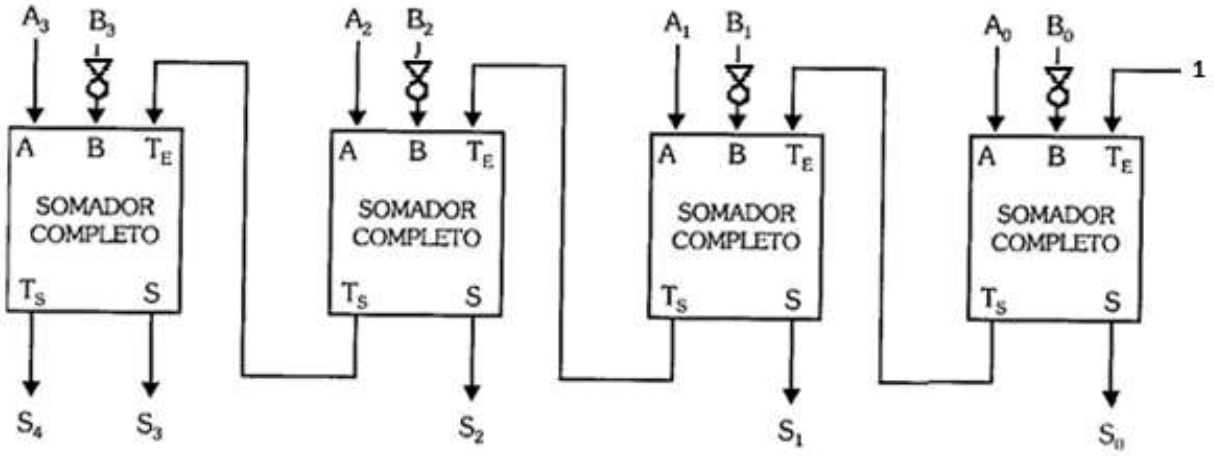
1. Complemente de 1 o subtraendo: mude o subtraendo complementando cada bit que compõe o subtraendo;
2. Adicione 1 ao subtraendo;
3. Some minuendo e subtraendo: o resultado será a diferença entre o subtraendo e minuendo.

O *overflow* indica que a resposta é positiva. Ignore o *overflow*.

Se não houver *overflow*, a resposta é negativa.

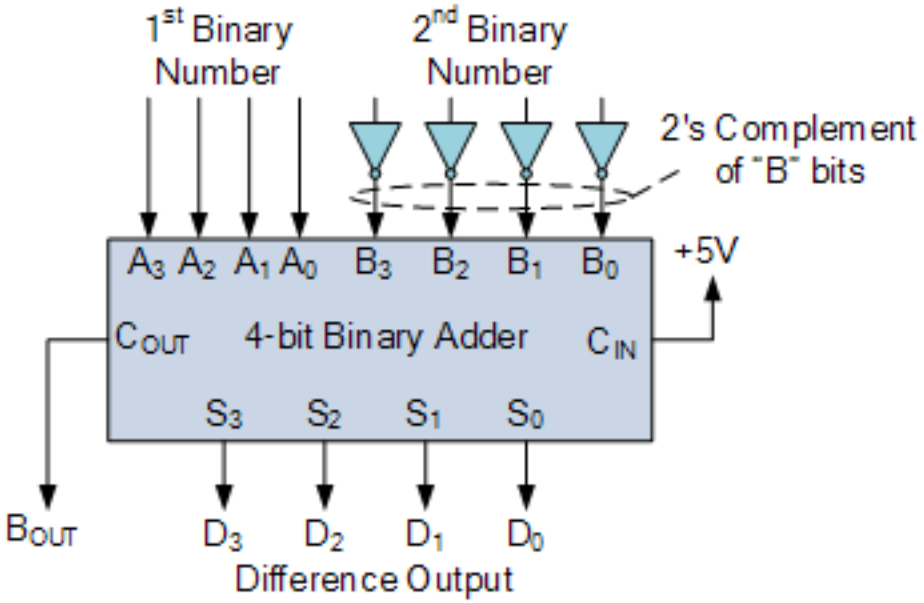
Subtrator em complemento de 2

Subtrator paralelo de 4 bits em complemento de 2:



Subtrator em complemento de 2 - circuito

Para construir um circuito subtrator, é necessário combinar um somador completo com inversores, que irão fazer o papel de complemento de 1 do subtraendo. O complemento de 2 é alcançado com a entrada C_{IN} conectada em nível lógico alto.



Não existe um circuito integrado TTL dedicado a realizar subtrações somente.

Fonte: electronics-tutorials.ws

Comparador digital

Um comparador digital pode ser do tipo:

- **comparador de identidade**, quando se deseja verificar se duas entradas são iguais; ou
- **comparador de magnitude**, quando se deseja verificar, além da igualdade, se uma entrada é maior ou menor que a outra entrada.

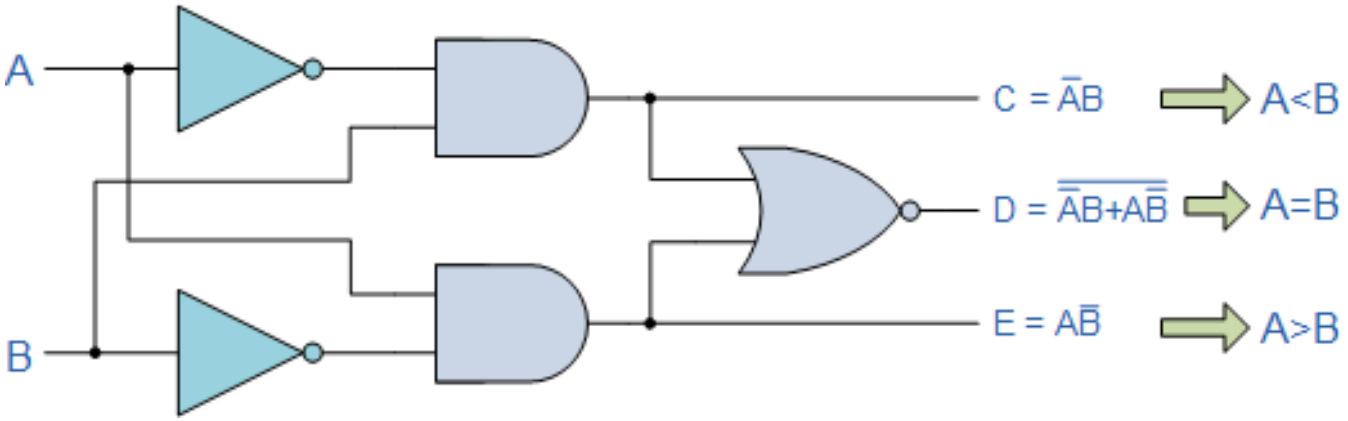
Os comparadores digitais são usados em unidades centrais de processamento (CPU) e unidades de microcontroladores (MCU).

Exemplos de comparadores digitais incluem o CMOS 4063 e 4585 e o TTL 7485 e 74682.

Fonte: electronics-tutorials.ws

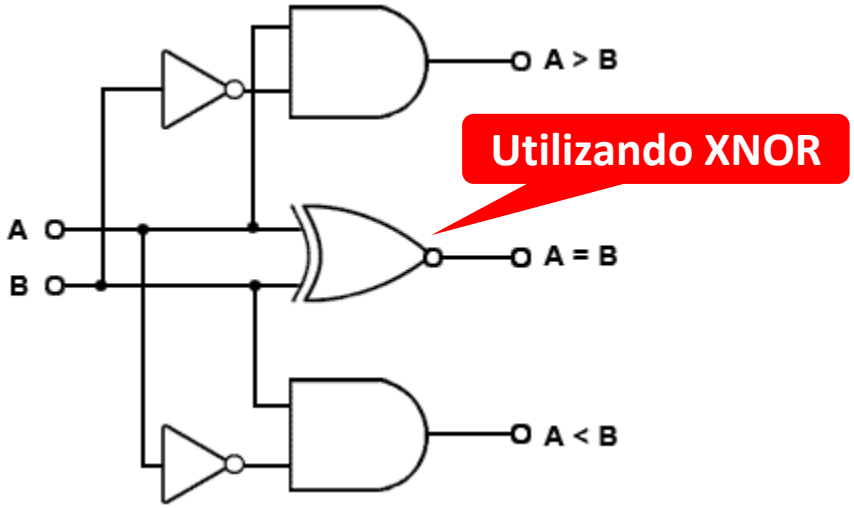
Comparador digital - tabela verdade e circuito

A	B	F(A<B)	F(A=B)	F(A>B)
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

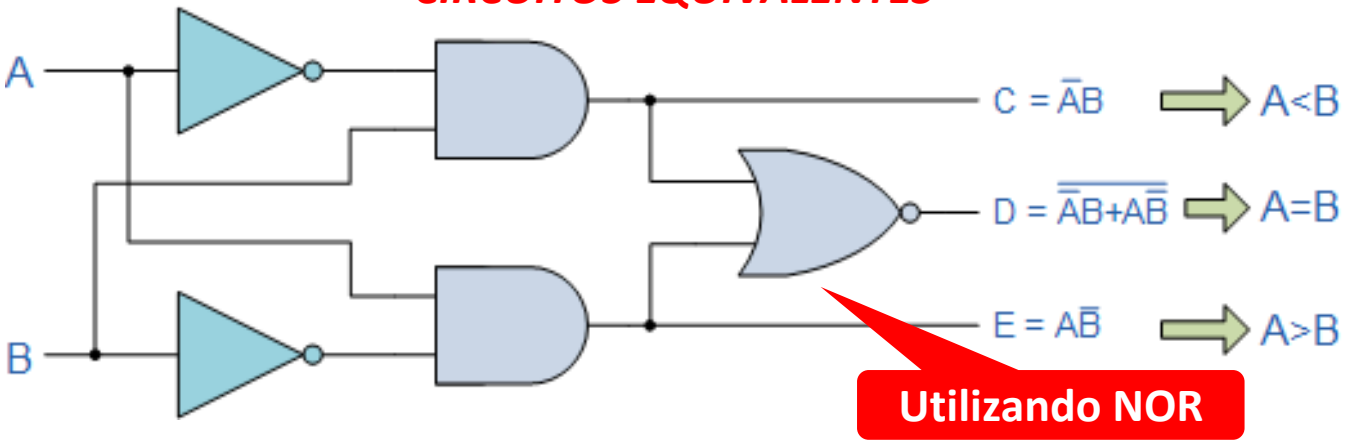


Fonte: electronics-tutorials.ws

Comparador digital - circuitos equivalentes



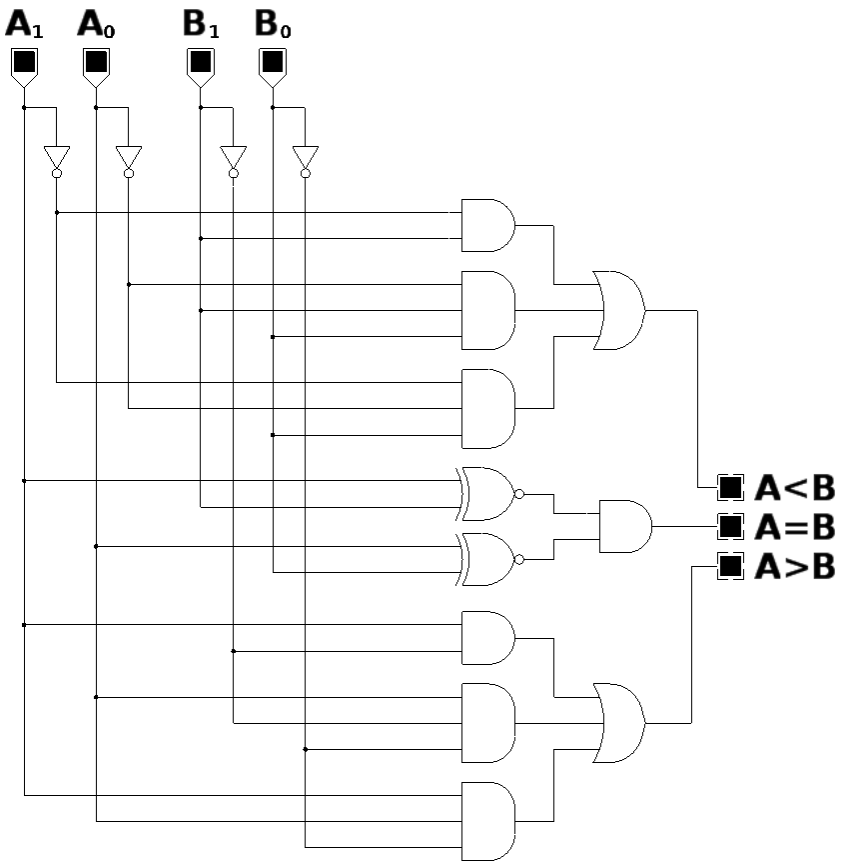
CIRCUITOS EQUIVALENTES



Fonte: learnabout-electronics.org; electronics-tutorials.ws

Comparador digital 2 bit
Tabela verdade e circuito

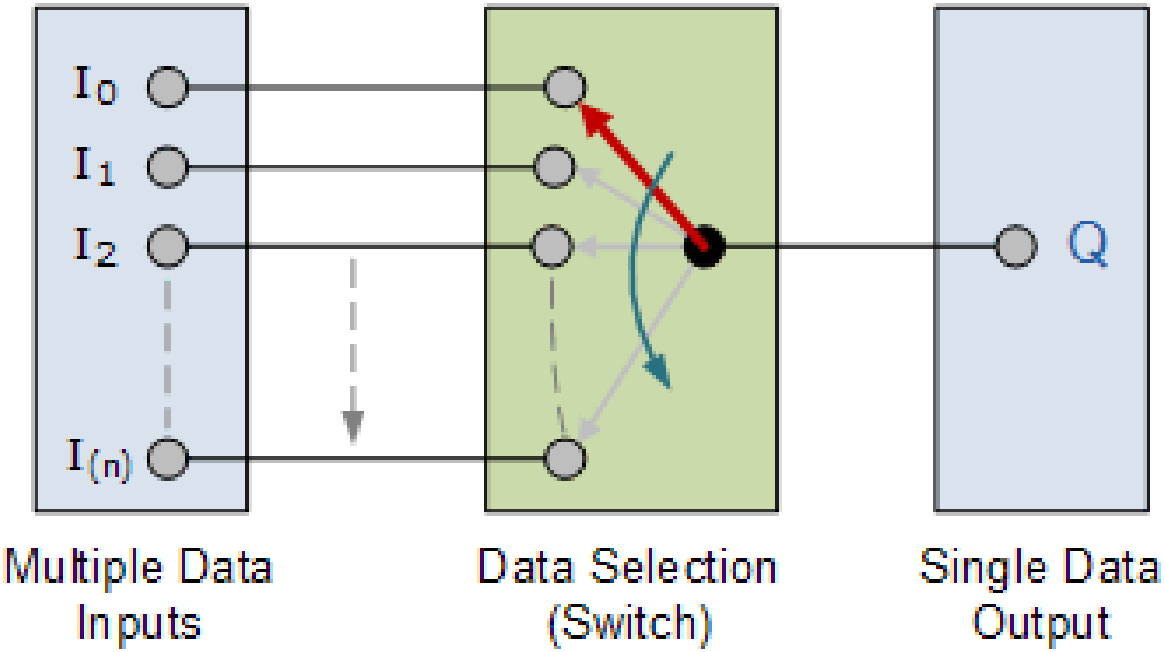
A_1	A_0	B_1	B_0	A<B	A=B	A>B
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0



Fonte: de-iitr.vlabs.ac.in/exp/comparator-using-logic-gates/theory.html

Multiplexador

Os multiplexadores são circuitos lógicos combinacionais projetados para alternar uma das várias linhas de entrada para uma única linha de saída comum pela aplicação de um sinal de controle.

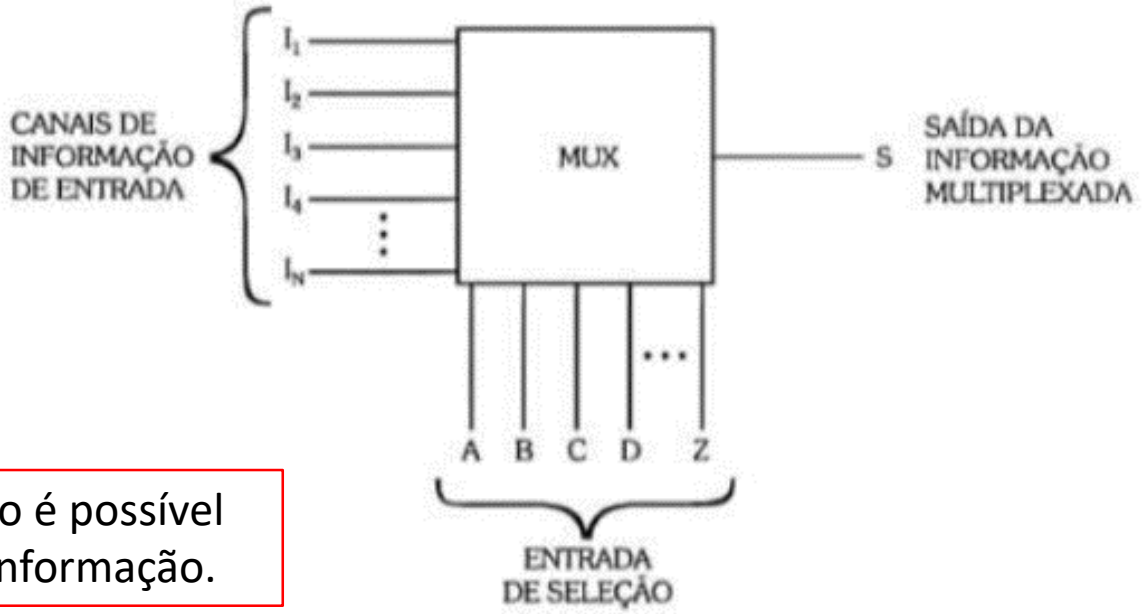


Fonte: electronics-tutorials.ws

Multiplexador

Para conectar a informação I_1 na saída S, basta selecionar a posição 1 da chave seletora. Para conectar à saída a informação I_2, é necessário selecionar a posição 2 e assim sucessivamente.

A entrada de seleção tem como finalidade escolher qual dos canais de informação deve ser ligado à saída.



Com ***n*** entradas de seleção é possível multiplexar **2^n** canais de informação.

Multiplexador - 2x1

No MUX básico para 2 canais de entrada I_0 e I_1, temos uma variável de seleção A, que quando for igual a 0, teremos na saída S a mesma informação que a entrada I_0; se I_0 for igual a 0, S será igual a 0 e se I_0 for igual a 1, S será igual a 1.

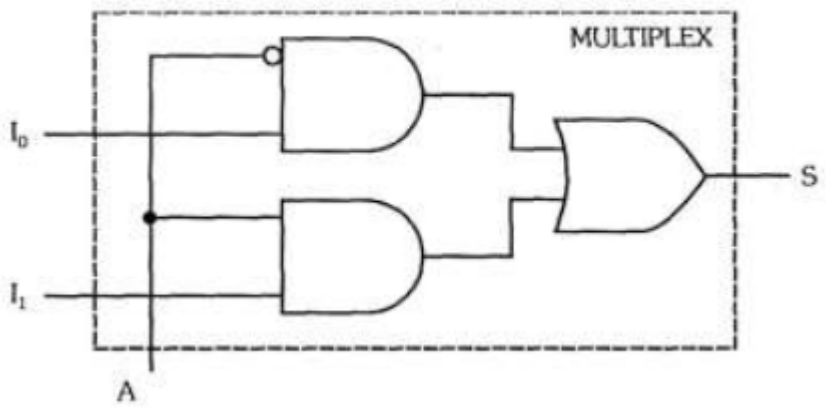
Neste caso, o canal I_1 será bloqueado pela porta AND referente a I_1, pois o outro terminal deste estará ligado em A que valerá 0.

Quando A for igual a 1, I_0 será bloqueado e, analogamente, a informação do canal I_1 aparecerá na saída.

TABELA VERDADE – MULTIPLEXADOR 2x1

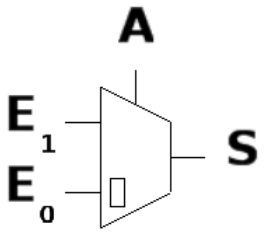
SELETOR	ENTRADAS		SAÍDA
A	I_0	I_1	S
0	I_0	1	$\bar{A}I_0$
1	1	I_1	AI_1

CIRCUITO LÓGICO - MULTIPLEXADOR 2x1

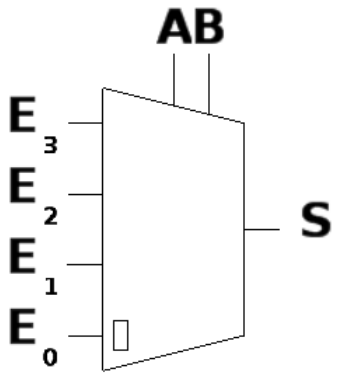


Bloco multiplexador no CEDAR

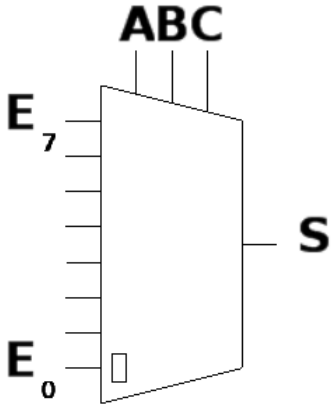
MSB <-- [ABCD] --> LSB



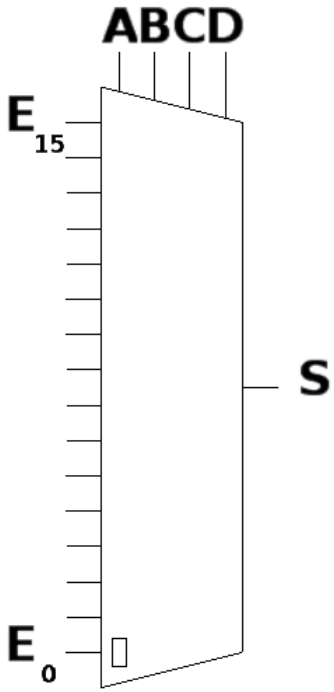
MUX 2x1



MUX 4x1



MUX 8x1

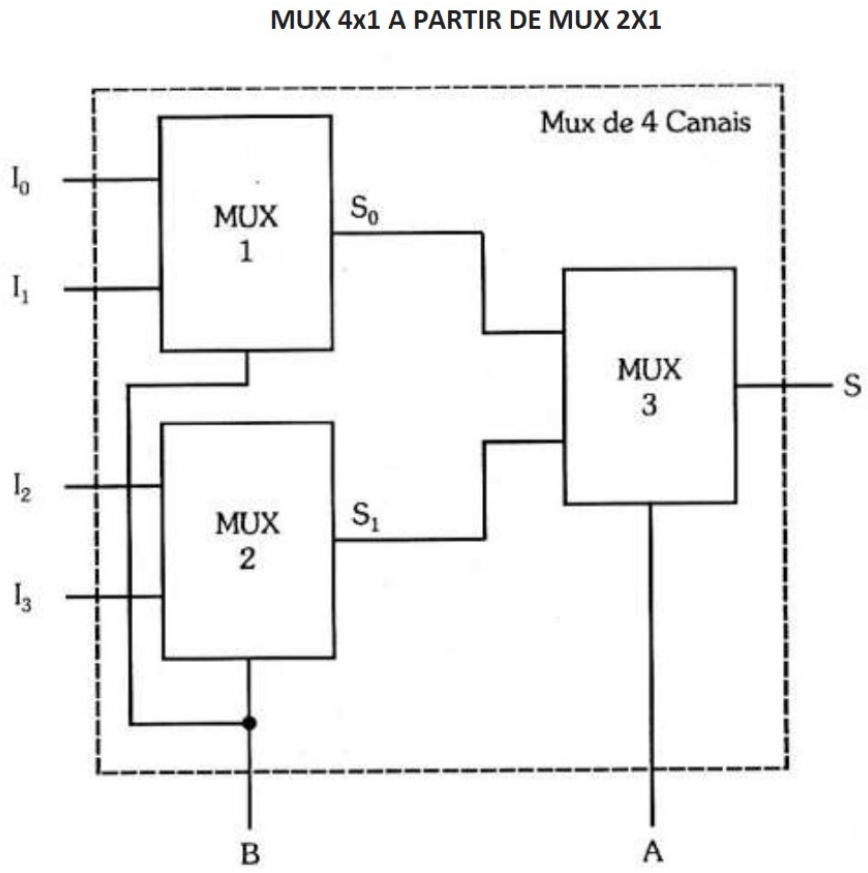


MUX 16x1

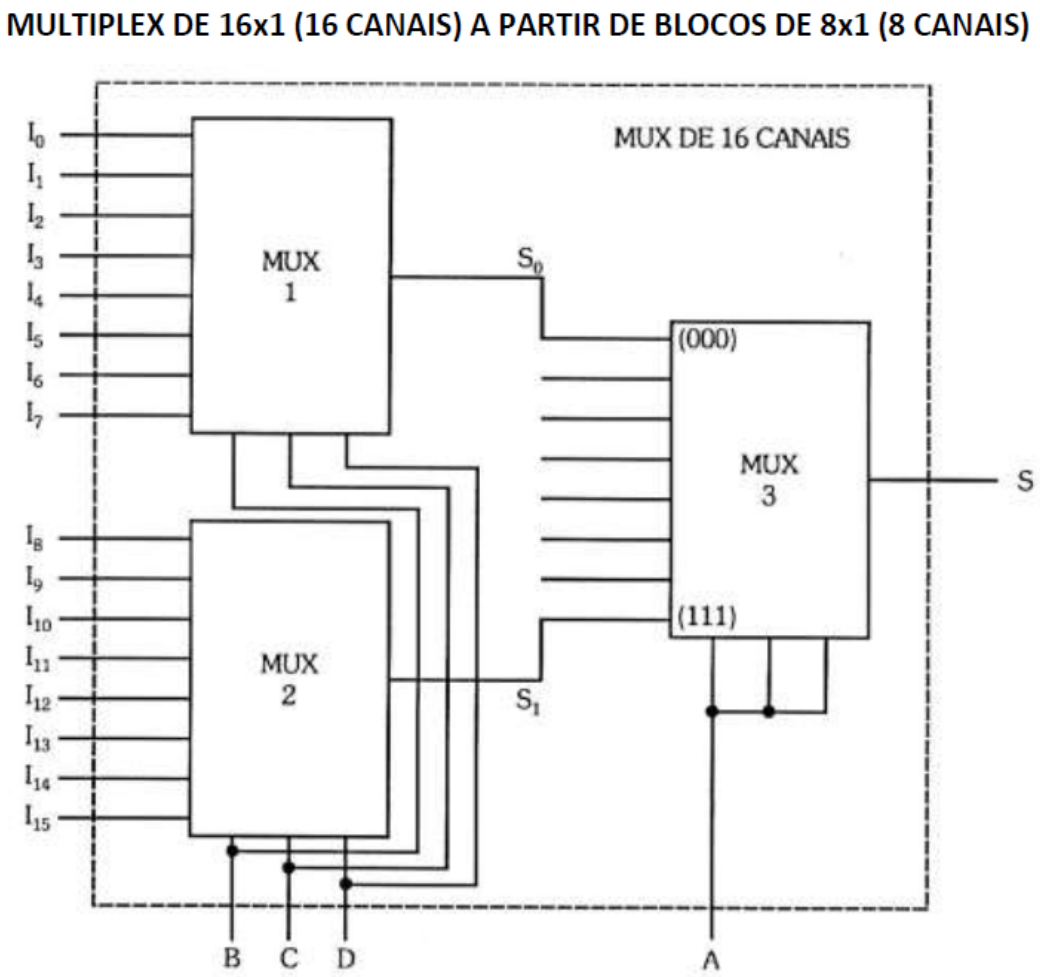
Ampliação de um multiplexador

A partir de um multiplexador de baixa capacidade é possível formar outros para um maior número de informações de entrada.

Como exemplo, é possível montar um multiplexador de 4 canais de informação a partir de outros de apenas 2 canais de informação.



Ampliação de um multiplexador



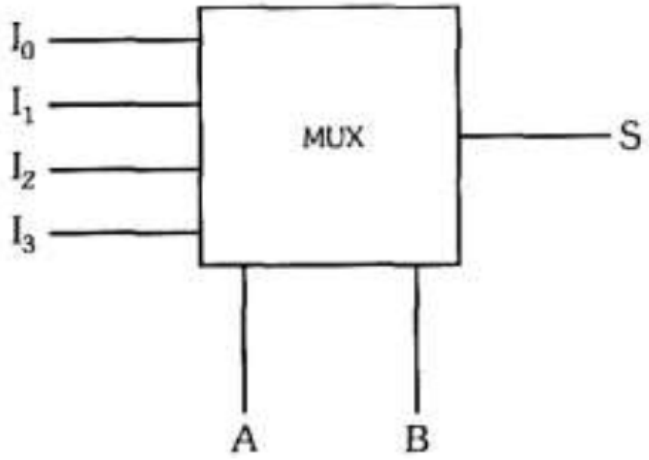
Multiplexador como gerador de funções lógicas

Além de selecionar sinais, o multiplexador pode ser usado para a implementação de funções lógicas.

EXPRESSÃO LÓGICA

$$S = \overline{A}\overline{B}\,I_0 + \overline{A}BI_1 + A\overline{B}I_2 + ABI_3$$

BLOCO LÓGICO – MUX 4x1



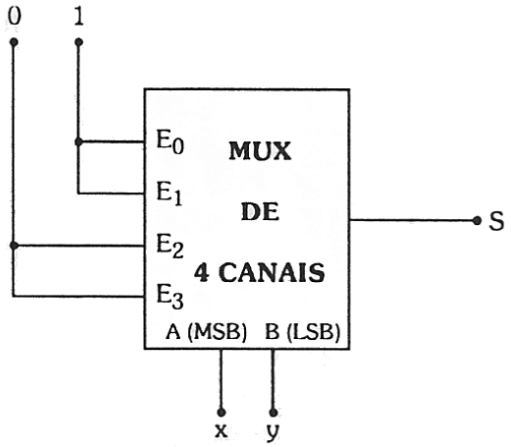
Multiplexador como gerador de funções lógicas - exemplo 1

Deseja-se implementar o circuito lógico correspondente à seguinte expressão booleana:

$$S = \bar{X} \cdot \bar{Y} + \bar{X} \cdot Y$$

Primeiro levanta-se a tabela verdade relativa à expressão que se deseja implementar, e em seguida o circuito lógico correspondente:

X	Y	S	Entrada do MUX
0	0	1	E_0
0	1	1	E_1
1	0	0	E_2
1	1	0	E_3



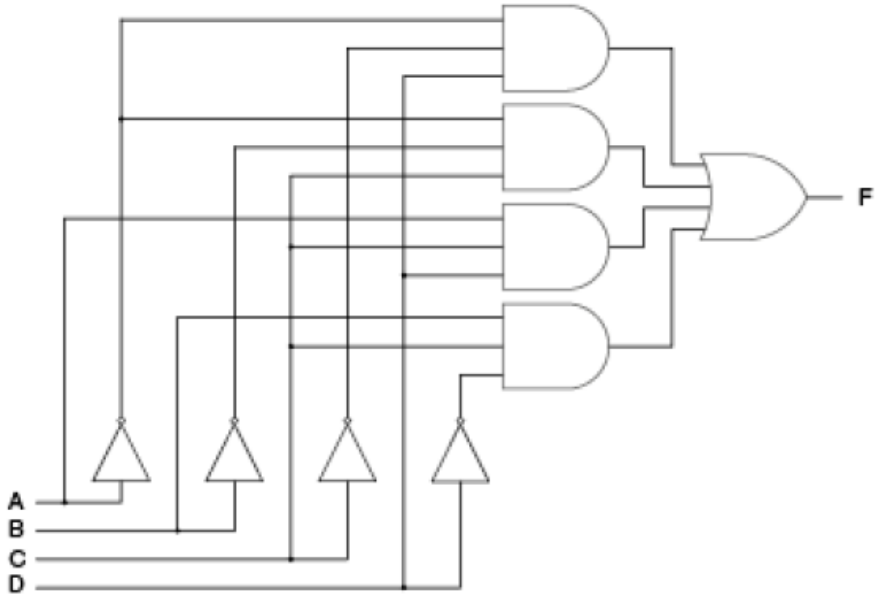
Fonte: LOURENÇO, A.C., et al. Circuitos Digitais. São Paulo: Érica, 1996

Multiplexador como gerador de funções lógicas - exemplo 2

EXPRESSÃO LÓGICA

$$F = \bar{A}\,\bar{C}D + \bar{A}\bar{B}C + ACD + BC\bar{D}$$

CIRCUITO LÓGICO

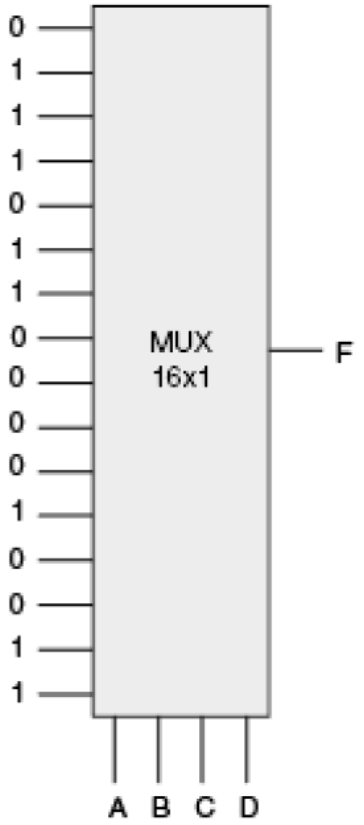


Multiplexador como gerador de funções lógicas - exemplo 2

REDUÇÃO - SOLUÇÃO ATRAVÉS DE MUX 16x1

$$F = \bar{A}\,\bar{C}D + \bar{A}\bar{B}C + ACD + BC\bar{D}$$

CD \ AB	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	1	0	1	1
10	1	1	1	0

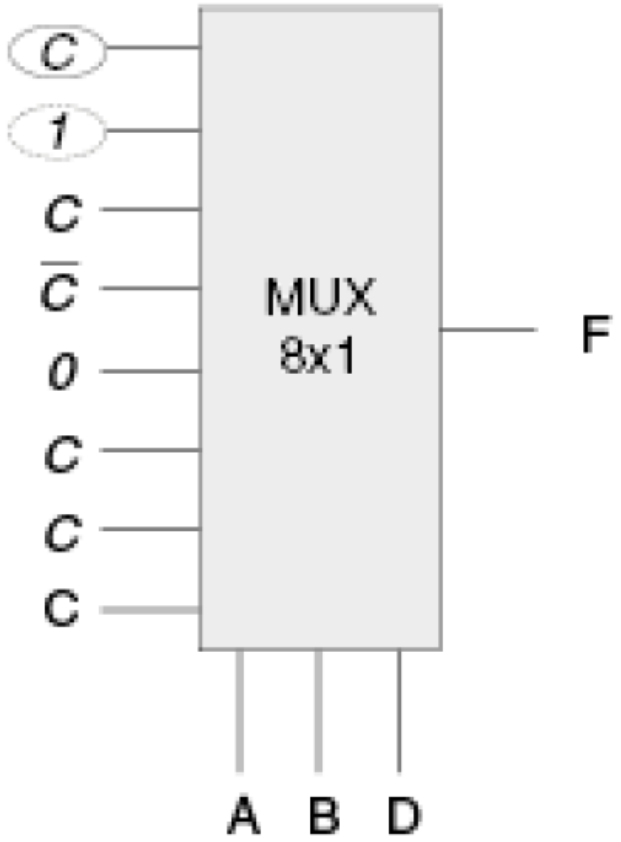


Multiplexador como gerador de funções lógicas - exemplo 2

REDUÇÃO - SOLUÇÃO ATRAVÉS DE MUX 8x1

$$F = \bar{A}\,\bar{C}D + \bar{A}\bar{B}C + ACD + BC\bar{D}$$

CD \ AB	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	1	0	1	1
10	1	1	1	0

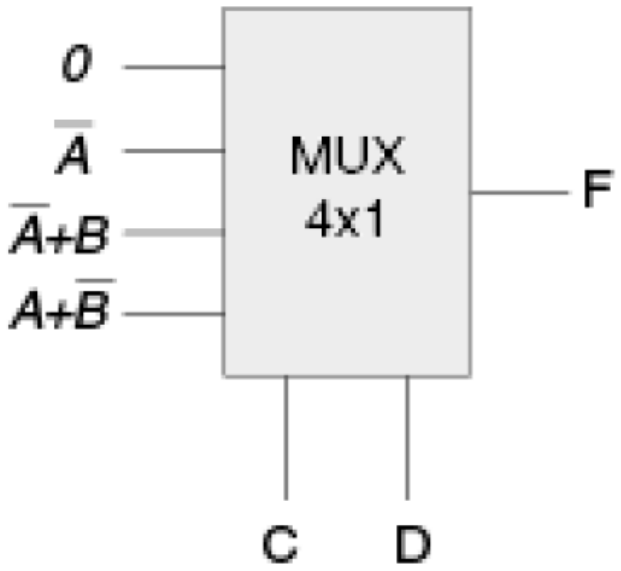


Multiplexador como gerador de funções lógicas - exemplo 2

REDUÇÃO - SOLUÇÃO ATRAVÉS DE MUX 4x1

$$F = \bar{A}\,\bar{C}D + \bar{A}\bar{B}C + ACD + BC\bar{D}$$

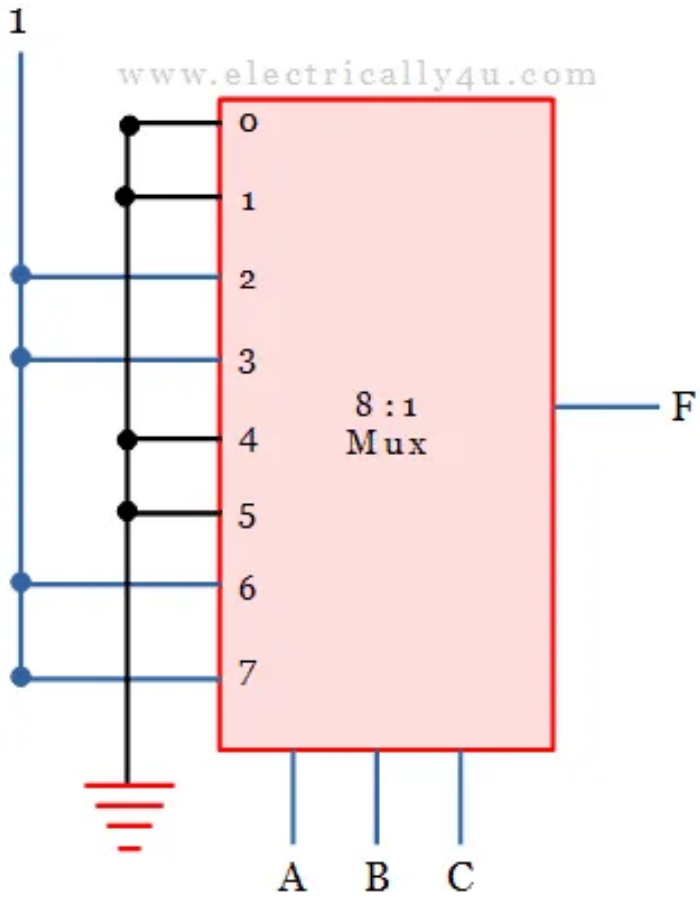
CD \ AB	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	1	0	1	1
10	1	1	1	0



Multiplexador como gerador de funções lógicas - exemplo 3

Deseja-se implementar o circuito lógico correspondente à seguinte expressão booleana expressa na forma de mintermos:

$$F(A,B,C) = \sum m(2,3,6,7)$$

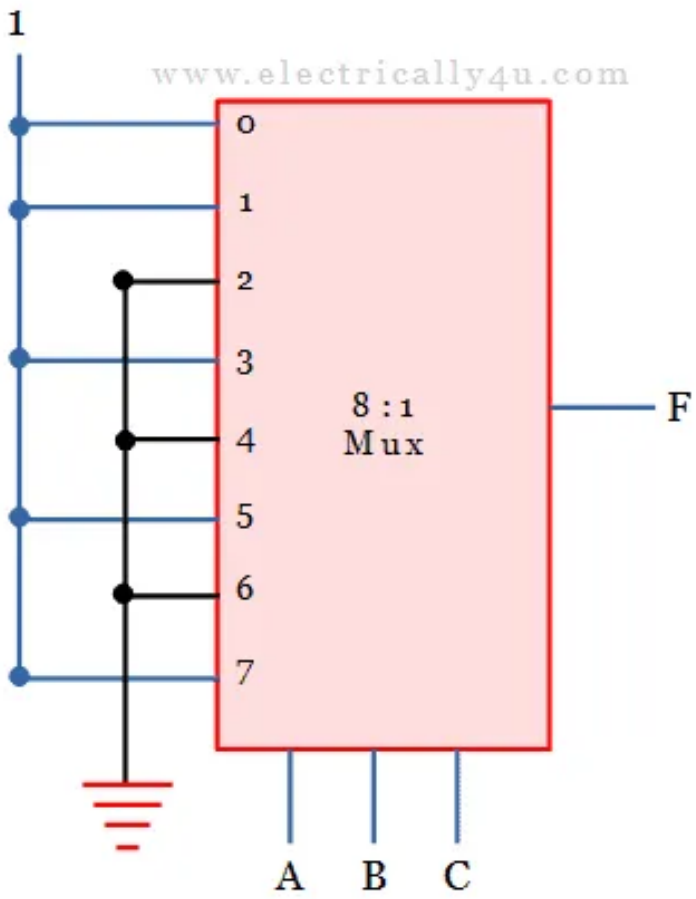


Fonte: electrically4u.com

Multiplexador como gerador de funções lógicas - exemplo 4

Deseja-se implementar o circuito lógico correspondente à seguinte expressão booleana expressa na forma de mintermos:

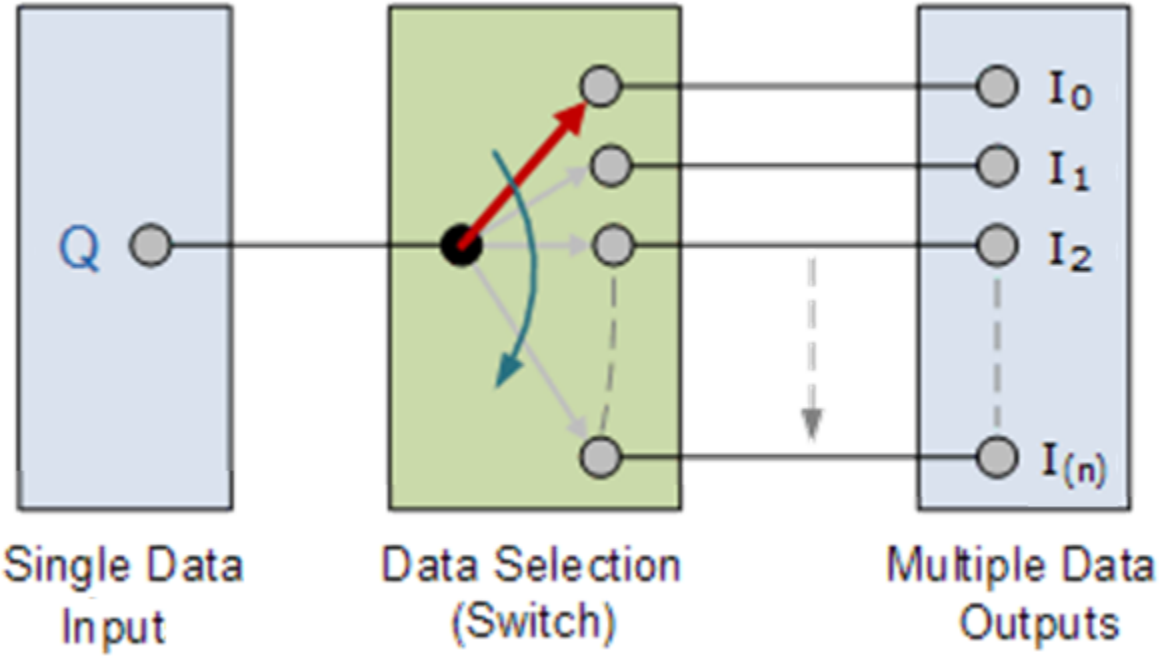
$$F(A,B,C) = \sum m(0,1,3,5,7)$$



Fonte: electrically4u.com

Demultiplexador

Os demultiplexadores são circuitos lógicos combinacionais projetados para alternar uma única linha de entrada comum para várias linhas de saída pela aplicação de um sinal de controle.

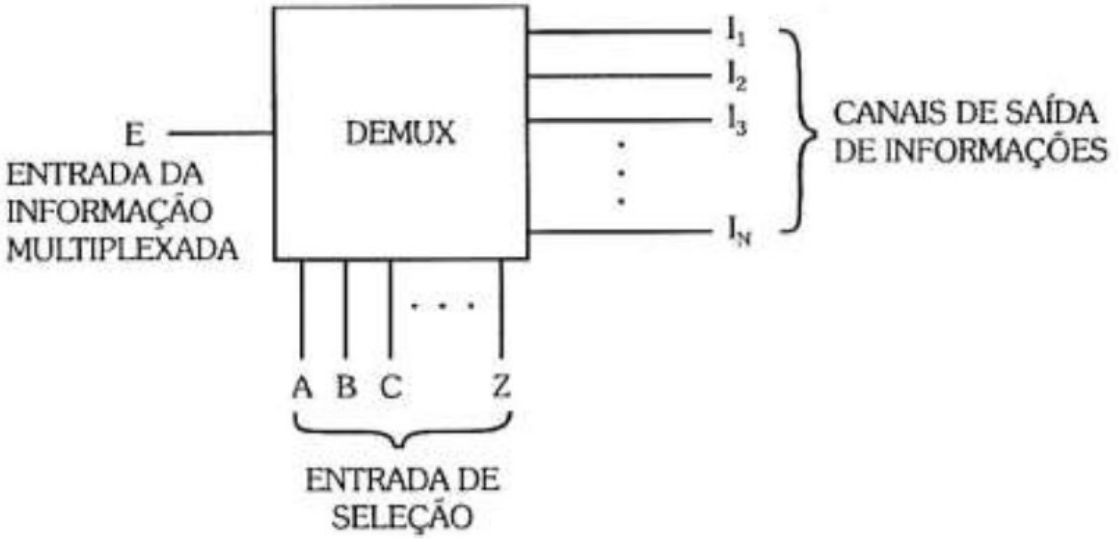


Fonte: electronics-tutorials.ws

Demultiplexador

Para conectar a informação E na saída I_1, basta selecionar a posição 1 da chave seletora. Para conectar a informação à saída I_2, é necessário selecionar a posição 2 e assim sucessivamente.

A entrada de seleção tem como finalidade escolher qual dos canais de informação deve ser ligado à entrada.



Demultiplexador - 1x2

No caso do DEMUX básico para 2 de informações de saída I_0 e I_1, temos uma variável de seleção A, que quando for igual a 0, teremos na saída a mesma informação que a entrada I_0; se I_0 for igual a 0, S será igual a 0 e se I_0 for igual a 1, S será igual a 1.

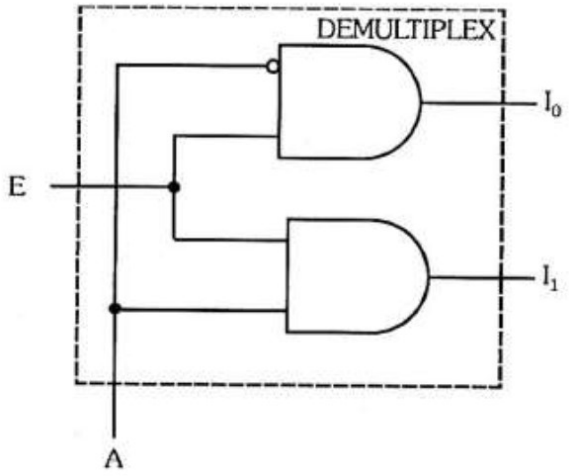
Neste caso, a informação I_1 será bloqueada pela porta E referente a I_1, pois o outro terminal desta estará ligado em A que valerá 0.

Quando A for igual a 1, I_0 será bloqueado e, analogamente, a informação I_1 aparecerá na saída.

TABELA VERDADE - DEMULTIPLEXADOR 1x2

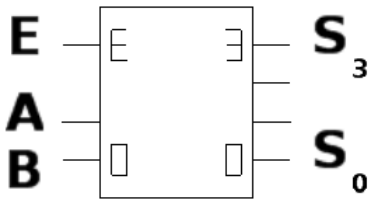
SELETOR	ENTRADAS	SAÍDAS	
A	E	I_0	I_1
0	E	$\bar{A}E$	AE
1	E	$\bar{A}E$	AE

DEMULTIPLEXADOR 1x2

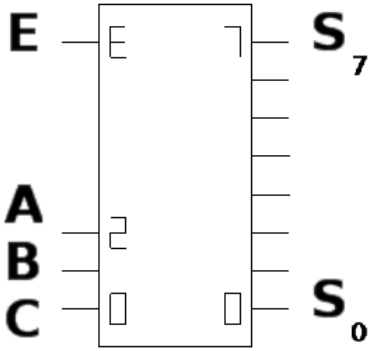


Bloco demultiplexador no CEDAR

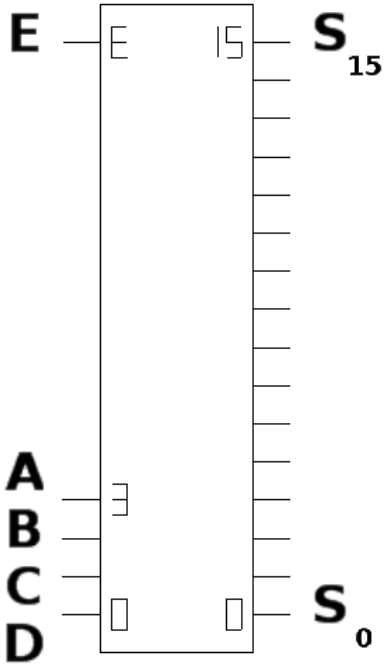
MSB <-- [ABCD] --> LSB



DEMUX 1x4



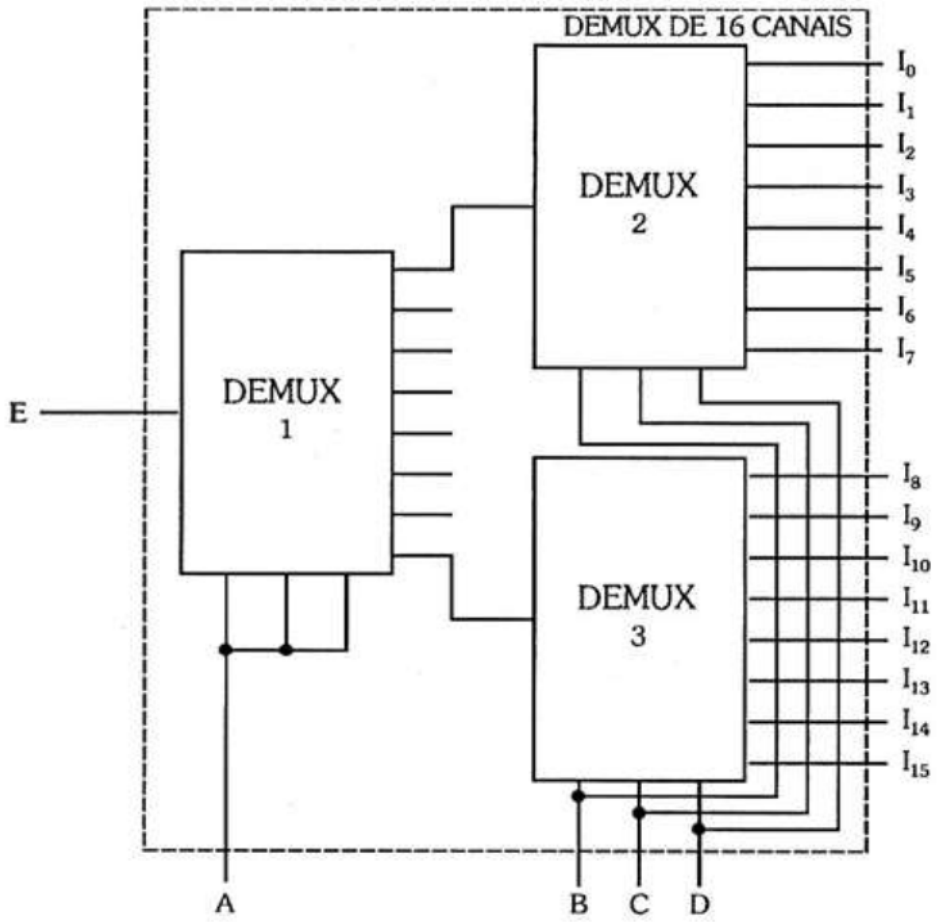
DEMUX 1x8



DEMUX 1x16

Ampliação de um demultiplexador

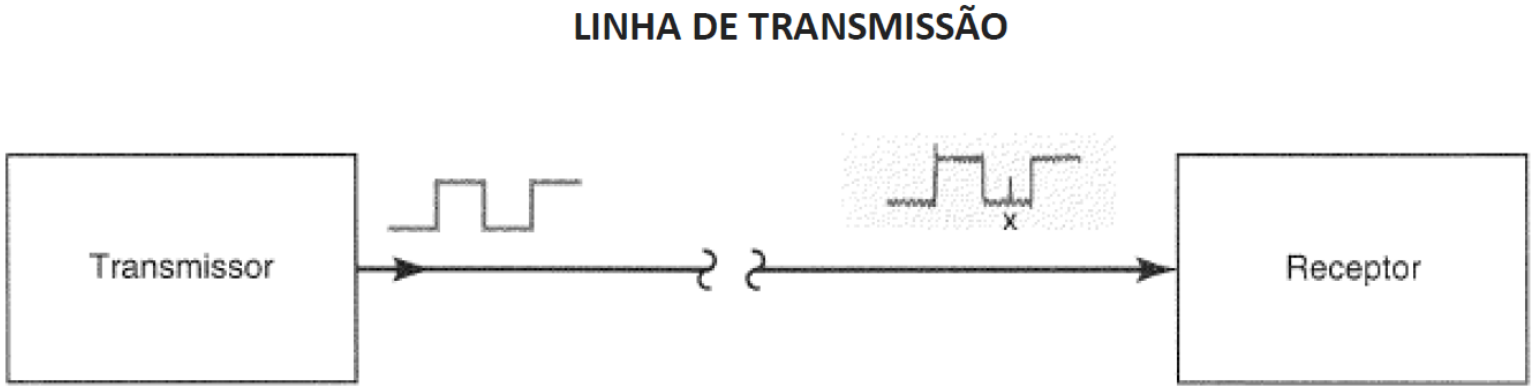
DEMULTIPLEXADOR DE 1x16 (16 CANAIS) A PARTIR DE DEMUX 1x8 (8 CANAIS)



Bit de paridade

Um bit de paridade é um bit extra que é anexado ao grupo de bits do código que está sendo transferido de um lugar a outro.

O bit de paridade pode ser 0 ou 1, dependendo do número de 1s contido no grupo.



Existem dois métodos de cálculo da paridade: par e ímpar.

Bit de paridade - método par

No método de paridade par, o valor do bit de paridade é escolhido de tal modo que o número total de 1s no grupo de bits do código (incluindo o bit de paridade) seja um número par.

Exemplo 1:

O grupo 1000011 corresponde ao caracter “C” em ASCII e possui três 1s, portanto adiciona-se um bit de paridade 1 para fazer com que o número total de 1s no grupo seja par.

O novo grupo será: **1**1000011

bit de paridade

Se o grupo de bits do código já conter inicialmente um número par de 1s, o bit de paridade assume o valor zero.

Bit de paridade - método par

Exemplo 2:

O grupo 1000001 corresponde ao caracter “A” em ASCII e possui dois 1s, portanto adiciona-se um bit de paridade 0 para fazer com que o número total de 1s no grupo seja par.

O novo grupo será: **0**1000001

Bit de paridade - método ímpar

No método de paridade ímpar, o valor do bit de paridade é escolhido de tal modo que o número total de 1s no grupo de bits do código (incluindo o bit de paridade) seja um número ímpar.

Exemplo 1:

O grupo 1000011 corresponde ao caracter “C” em ASCII e possui três 1s, portanto adiciona-se um bit de paridade 0 para fazer com que o número total de 1s no grupo seja ímpar.

O novo grupo será: **0**1000011

bit de paridade

Se o grupo de bits do código já conter inicialmente um número ímpar de 1s, o bit de paridade assume o valor zero.

Bit de paridade - método ímpar

Exemplo 2:

O grupo 1000001 corresponde ao caracter “A” em ASCII e possui dois 1s, portanto adiciona-se um bit de paridade 0 para fazer com que o número total de 1s no grupo seja ímpar.

O novo grupo será: **1**1000001

Circuito digitais - combinacional *vs* sequencial

No **circuito combinacional**, a saída depende unicamente do estado das entradas.

No **circuito sequencial**, a entrada é retroalimentada pela saída atual Qa, de tal modo que o estado futuro da saída Qf depende das entradas e do estado atual Qa.

Combinacional
A saída depende apenas das entradas

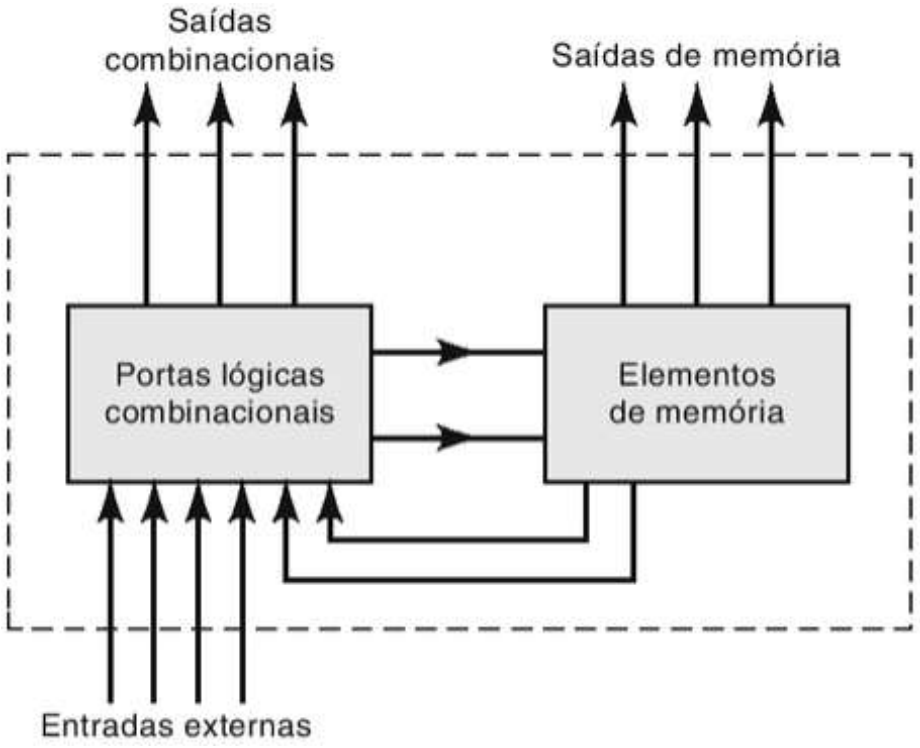
Sequencial
A saída futura depende das entradas e da saída atual

Figura: notas de aula do Prof. Onur Mutlu

Circuito sequenciais

Denomina-se estado interno a realimentação do sistema.

As condições atuais da entrada e do estado interno determinam a condição futura da saída.



Latches e flip-flops

Latches e flip-flops são dispositivos de armazenamento temporário (volátil). Eles apresentam dois estados estáveis (0 e 1) na saída, portanto denominados de biestáveis.

Existem dois tipos básicos de biestáveis:

- **Latches:** permite a troca de estado de maneira assíncrona ou síncrona, ou seja, muda o seu estado por ação de um pulso de disparo de relógio interno (clock);
- **Flip-flops:** permite a troca de estado apenas de maneira síncrona por borda de subida ou descida, ou seja, somente durante a rampa de mudança de estado do relógio interno (clock).

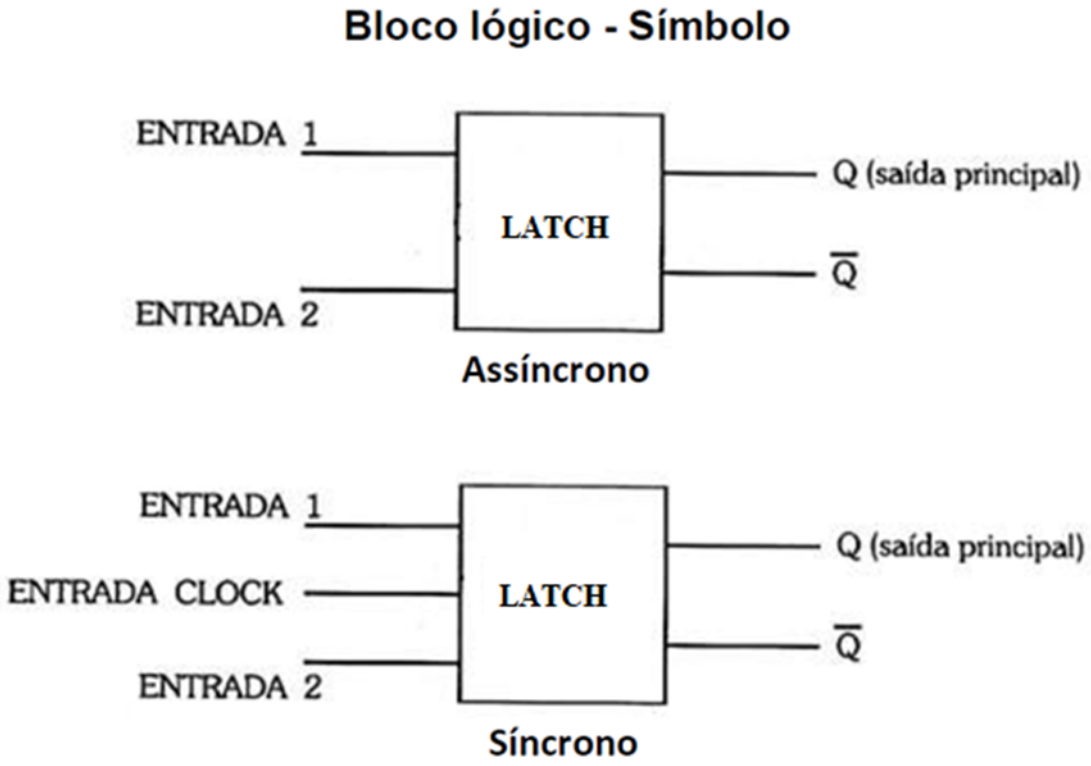
Devido a característica de manter um determinado estado, os biestáveis são denominados também de elementos de memória.

Latches são mais utilizados como memória e os flip-flops como contadores e registradores.

Latches e flip-flops

Existem basicamente quatro tipos de biestáveis:

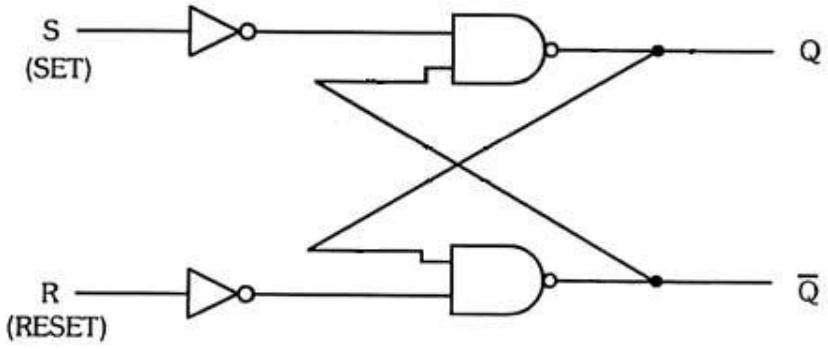
- Tipo SR;
- Tipo D;
- Tipo JK;
- Tipo T.



Latch SR (set/reset)

Latch SR é um tipo de dispositivo lógico biestável ou multivibrador e sua entrada é ativada em nível alto (nível 1).

Pode ser composto com duas portas NAND tendo um acoplamento cruzado (retroalimentação).



Bloco lógico

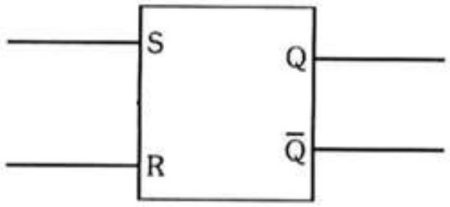


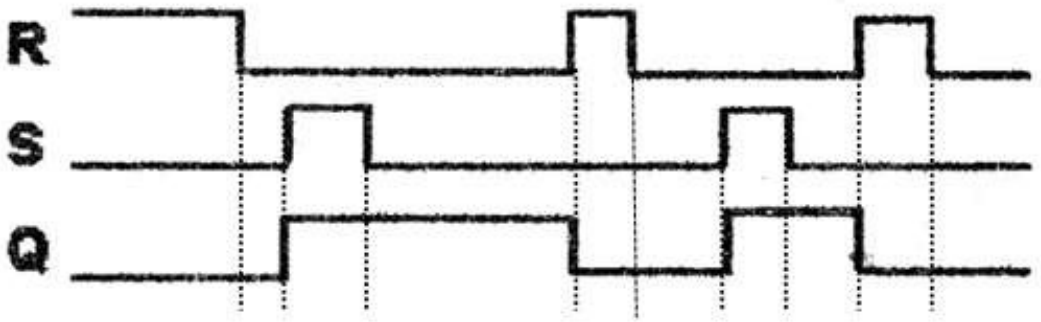
Tabela Verdade

S	R	Q
0	0	Mantém
0	1	0
1	0	1
1	1	Proibido

Latch SR (set/reset) - diagrama de tempo

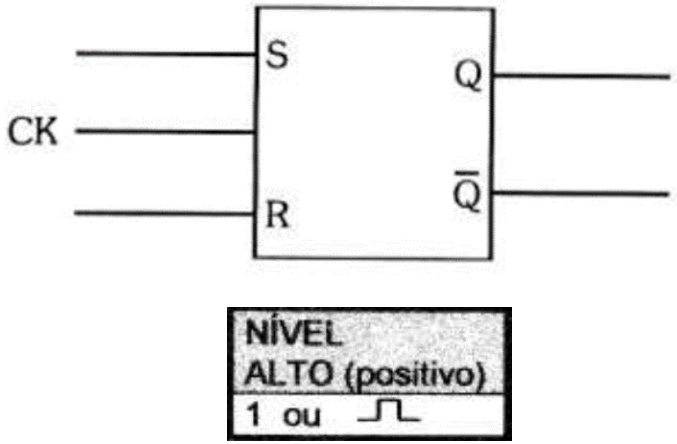
Diagramas ou cartas de tempo apresentam as alterações das saídas do circuito lógico em função das alterações das entradas ao longo do tempo.

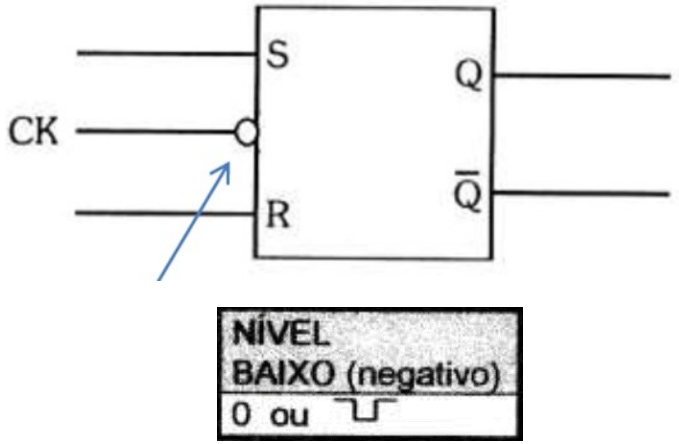
Para o latch SR temos o seguinte diagrama de tempo:

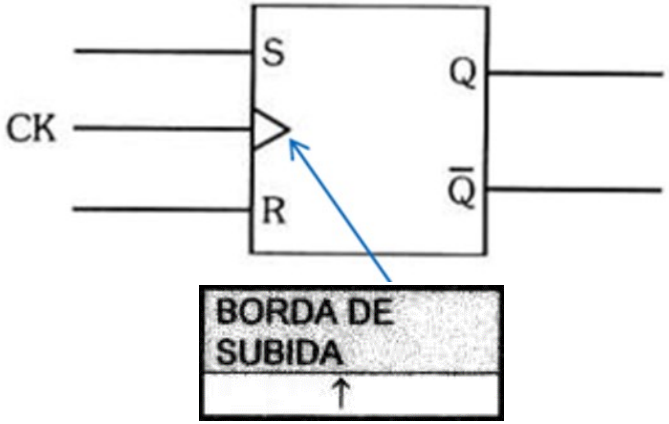


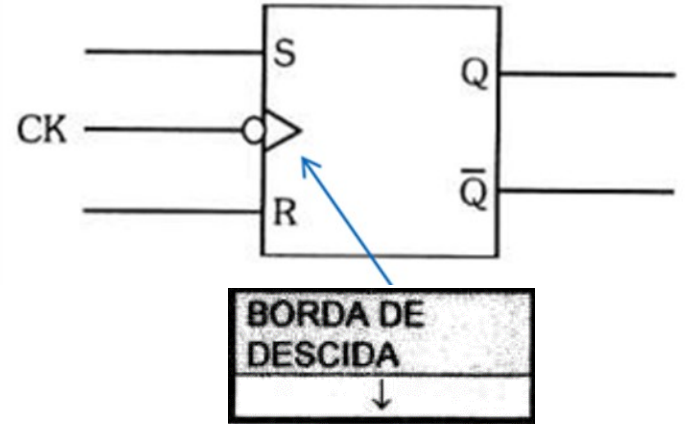
Observe que o circuito do latch SR irá muda apenas no instante em que ocorrem mudanças nas variáveis de entrada.

Sincronismo - resumo









Flip-flop SR - síncrono borda de subida com Preset e Clear

Tabela Verdade:

A tabela verdade neste caso apresenta 5entradas (PRESET, CLEAR, clock, S e R).

Quando a saída é forçada através do PRESET ou do CLEAR, independentemente do que se tenha nas entradas S ou R (ou mesmo no clock), a mesma fica inalterada.

TABELA VERDADE PARA FLIP-FLOP BORDA DE SUBIDA

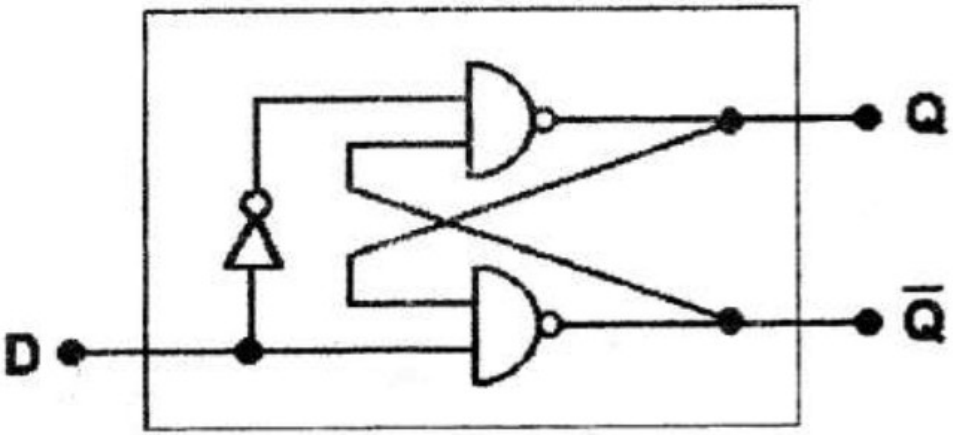
PR	CL	CK	R	S	Q
0	0	X	X	X	proibido
0	1	X	X	X	1
1	0	X	X	X	0
1	1	0	X	X	mantém
1	1	1	X	X	mantém
1	1	↓	X	X	mantém
1	1	↑	0	0	mantém
1	1	↑	0	1	1
1	1	↑	1	0	0
1	1	↑	1	1	proibido

Latch D

O sinal que estiver em sua entrada (D) é o mesmo que vai para a saída (Q).

A sigla D vem de *data* (dado), termo original em inglês.

Circuito baseado em Portas NAND



Simbologia

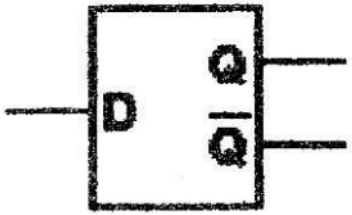
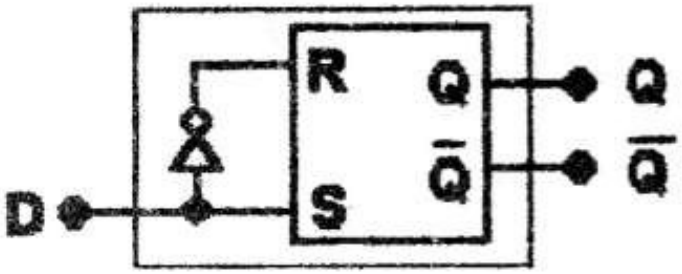


Tabela Verdade

D	Q
0	0
1	1

Latch D

O biestável tipo D pode ser construído através do latch SR utilizando uma porta inversora.

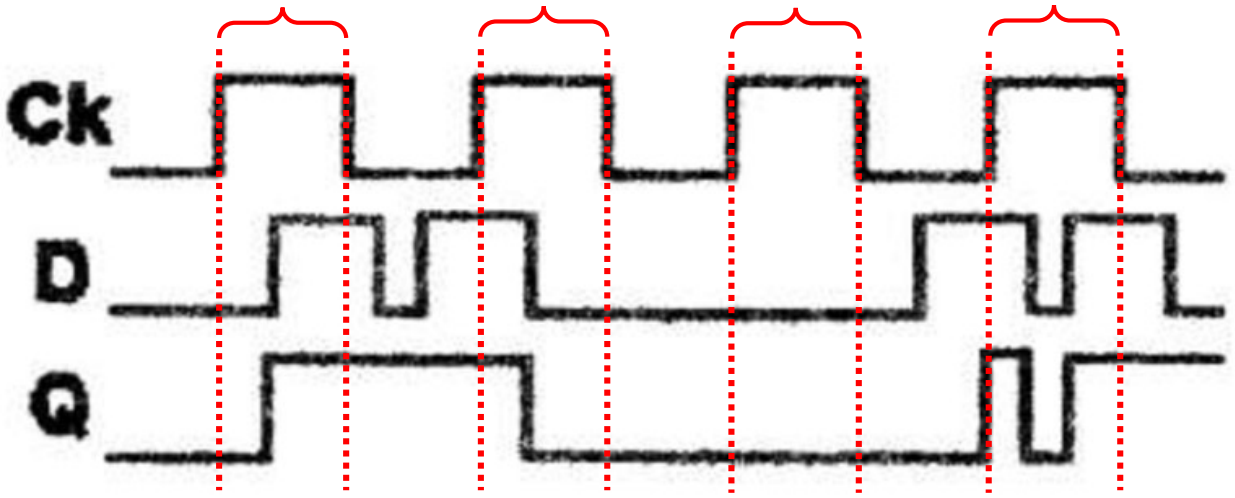


⚠ Esta é uma das formas de se eliminar a condição de estado proibido, uma vez que as entradas S e R sempre serão complementares. Outra forma é implementando o flip-flop do tipo JK, visto adiante.

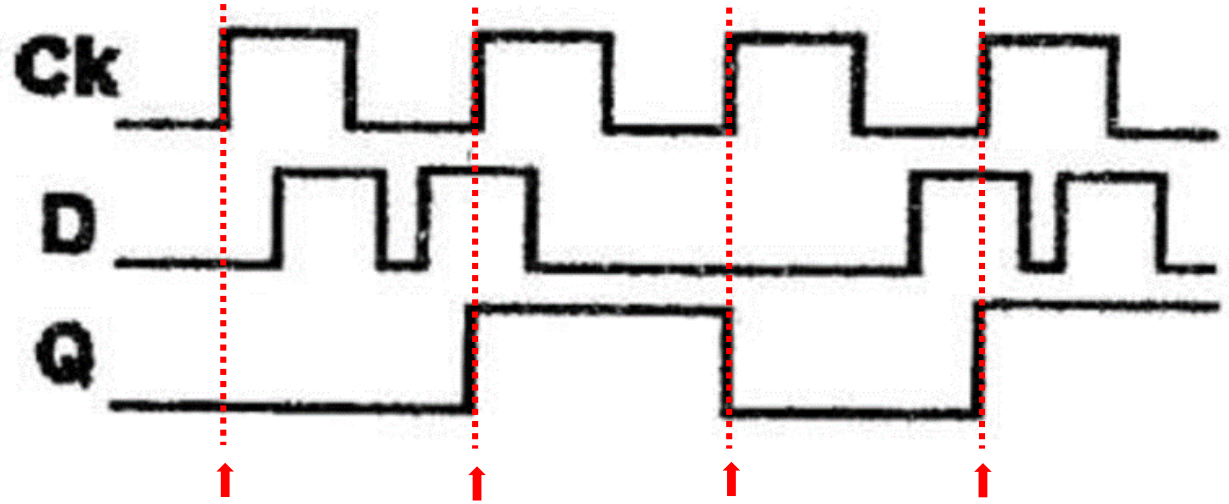
⚠ Também é possível construir este mesmo biestável utilizando o flip-flop do tipo JK.

Latch D - comparação

NÍVEL ALTO



BORDA DE SUBIDA



Flip-flop JK

O biestável JK só existe na configuração borda de subida ou descida. Não é possível utilizá-lo no modo assíncrono ou a nível.

Simbologia

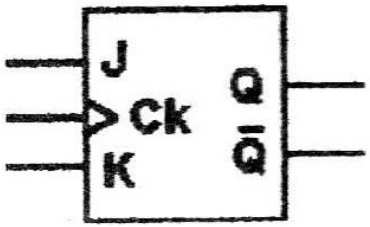
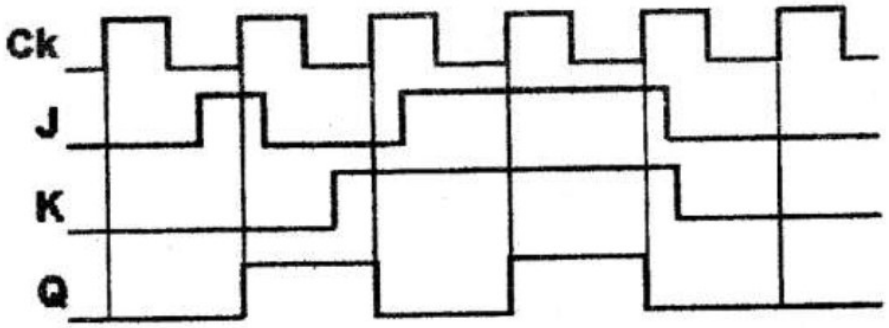


Tabela Verdade

Ck	J	K	Q
↑	0	0	Mantém
↑	0	1	0
↑	1	0	1
↑	1	1	Inverte

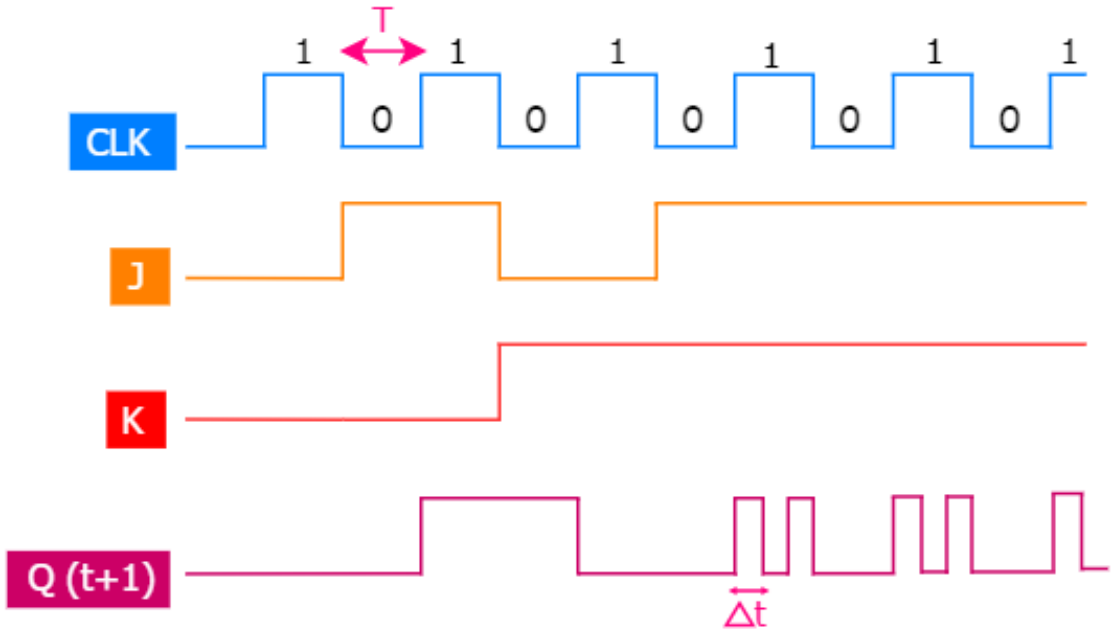
Diagrama de Tempos



Flip-flop JK - corrida em torno da condição

No entanto, a saída Q pode apresentar uma indeterminação, resultado de uma condição conhecida como corrida em torno da condição ou *race around condition*.

Esta condição ocorre quando J, K e o clock são iguais a “1” e o tempo T do pulso de clock em “1” é maior que o tempo de propagação Δt da saída Q para Q’.



Fonte: learn.circuitverse.org

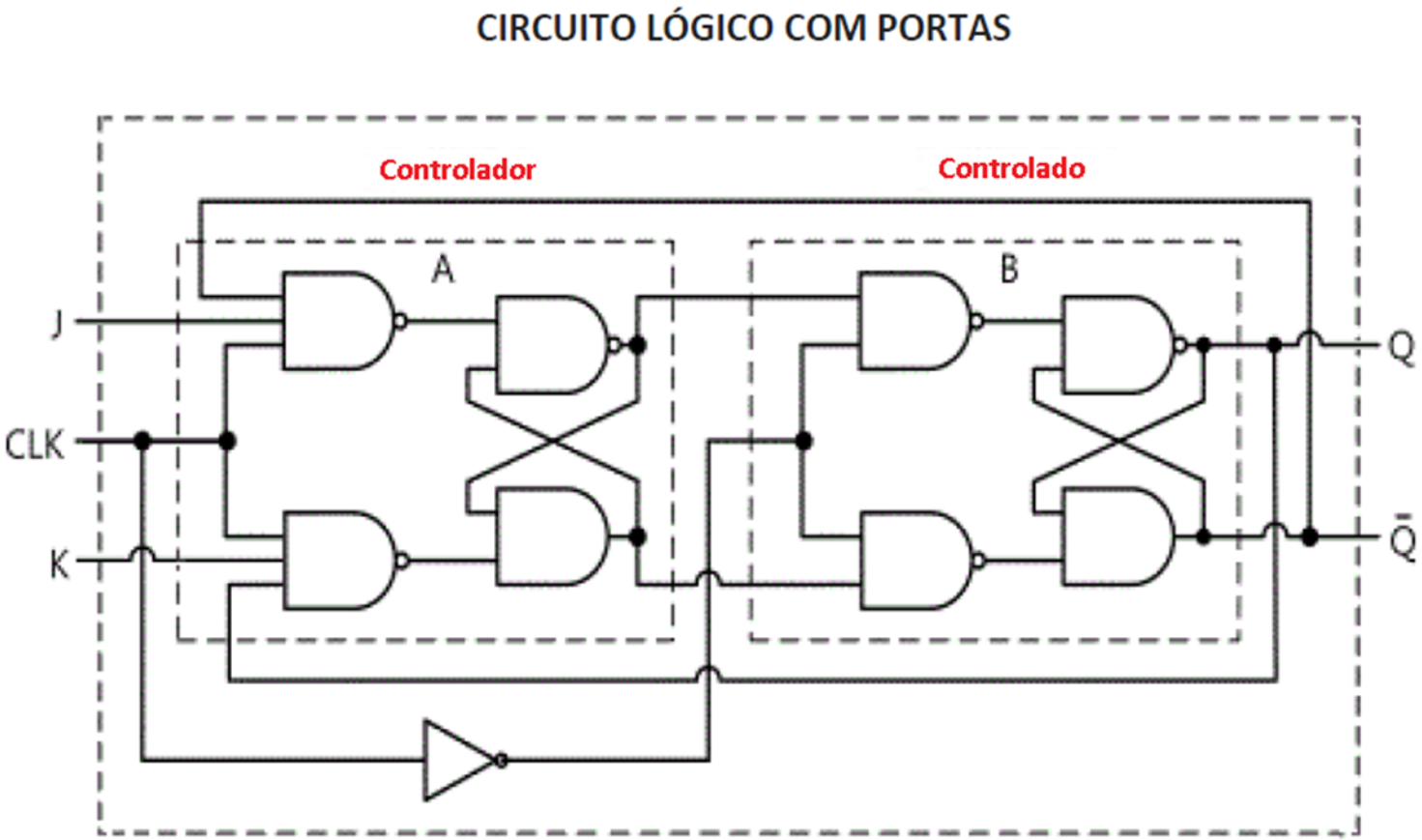
Flip-flop JK - corrida em torno da condição

Para contornar esta condição, pode-se adotar três métodos:

- **Aumentar o tempo de atraso do flip-flop:** o atraso de propagação Δt deve ser maior que a duração T do pulso de clock, mas isso diminuirá a velocidade do sistema;
- **Usar flip-flop disparado por borda:** ao utilizar um flip-flop disparado por borda em vez de usar o flip-flop disparado por nível, o clock ficará alto por um intervalo de tempo menor que o atraso de propagação do flip-flop;
- **Uso do flip-flop JK controlador/controlado:** trata-se de um flip-flop JK modificado que elimina o problema da corrida em torno da condição.

Fonte: learn.circuitverse.org

Flip-flop JK controlador/controlado



Flip-flop T

É obtido a partir de um flip-flop JK com as entradas J e K curto-circuitadas (uma ligada à outra), e só existe na configuração borda de subida ou borda de descida.

A sigla T vem de *toggle* (comutado).

Simbologia

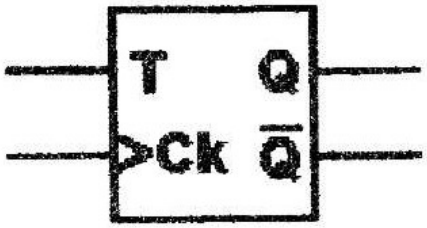
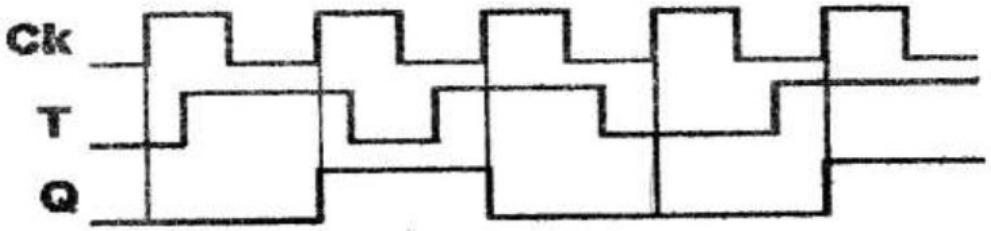


Tabela Verdade

Ck	T	Q
↑	0	mantém
↑	1	inverte

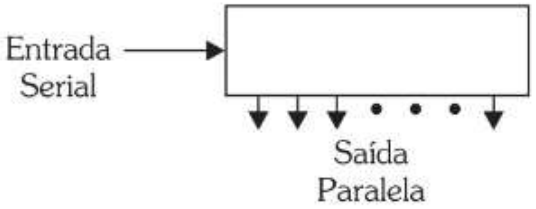
Diagrama de Tempos



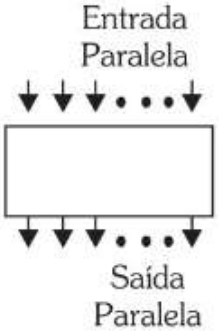
Registradores

a) Registrador Série-Série

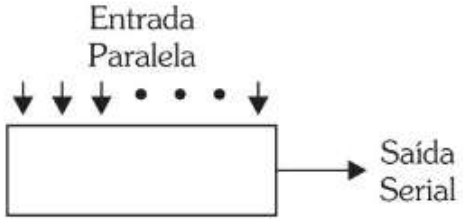
b) Registrador Série-Paralelo



c) Registrador Paralelo-Paralelo

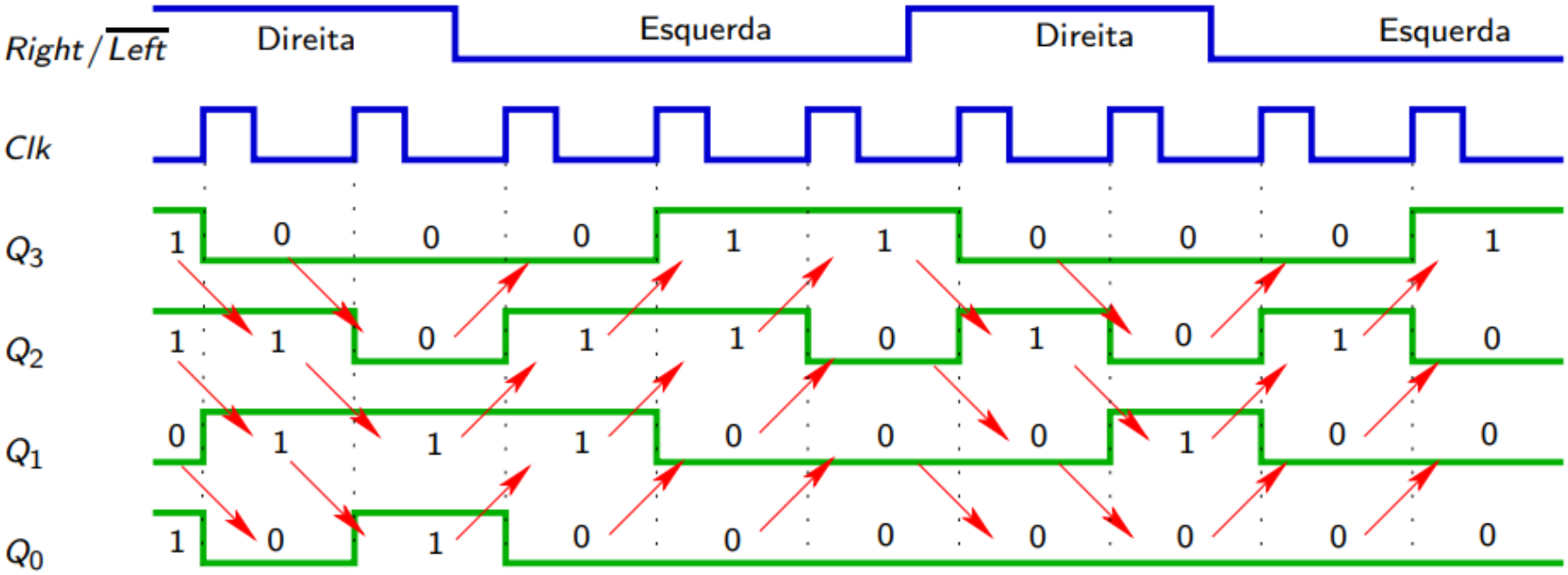


d) Registrador Paralelo-Série



Registradores de deslocamento bidirecionais

Nos registradores, os dados podem ser deslocados para direita (shift to right) ou para a esquerda (shift to left) nos registradores.



Contadores

São utilizados principalmente para contagens diversas, divisão de frequência, medição de frequência e tempo, geração de formas de onda e conversão de analógico para digital.

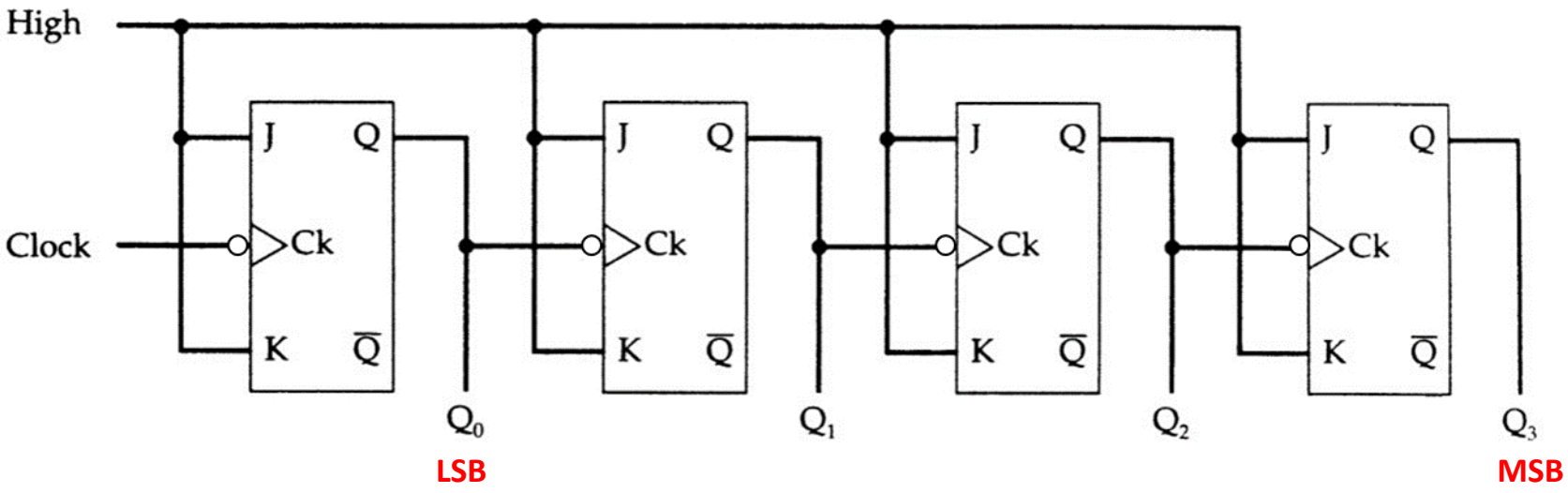
Podem ser divididos em duas categorias:

- **Assíncronos:** as transições dos flip-flops não são simultâneas;
- **Síncronos:** as transições dos flip-flops são simultâneas e geradas pelo mesmo sinal de clock.

Um contador pode realizar uma contagem de forma crescente ou decrescente.

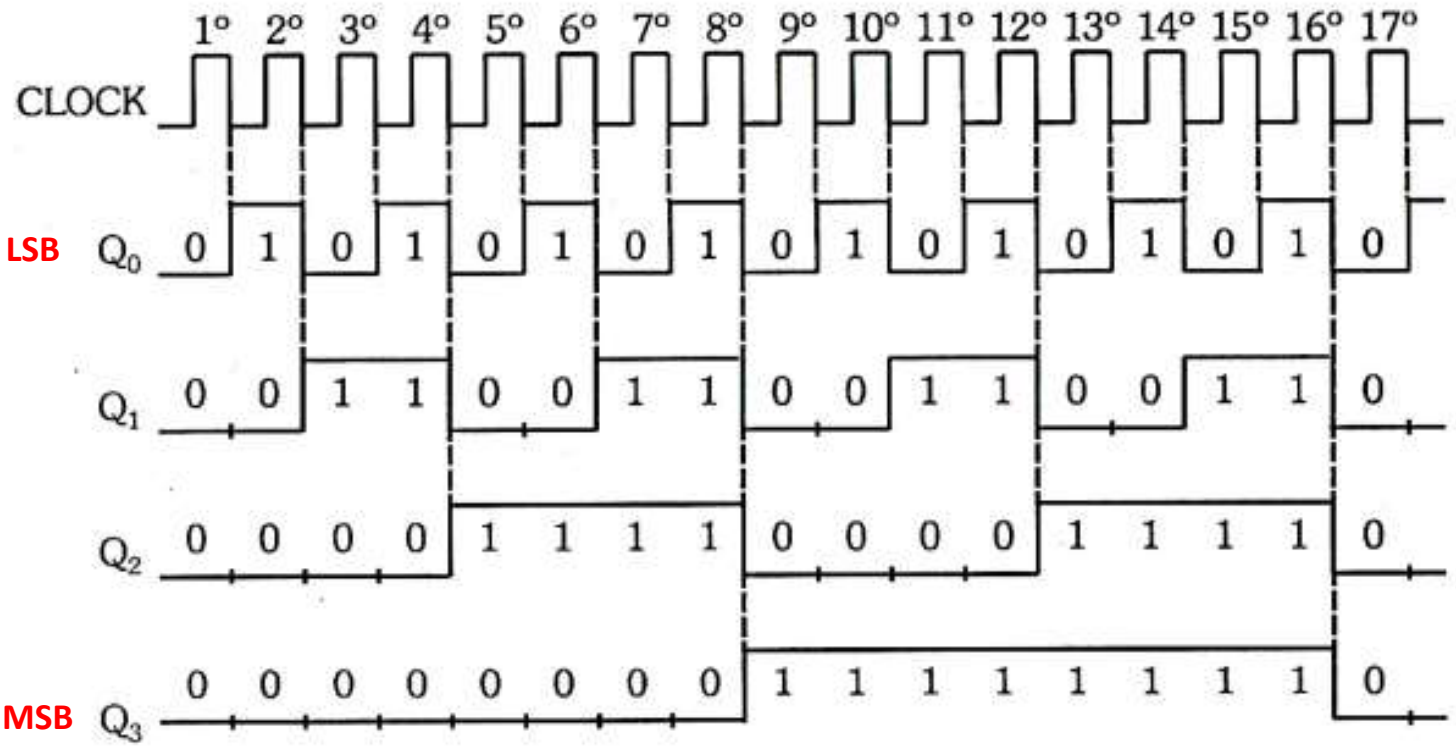
Contador assíncrono crescente

Seu circuito básico apresenta um grupo de quatro flip-flops do tipo T ou JK em que as entradas são sempre iguais a 1, originando na saída $Q_f = \overline{Q_a}$ a cada descida de clock.



Contador assíncrono crescente

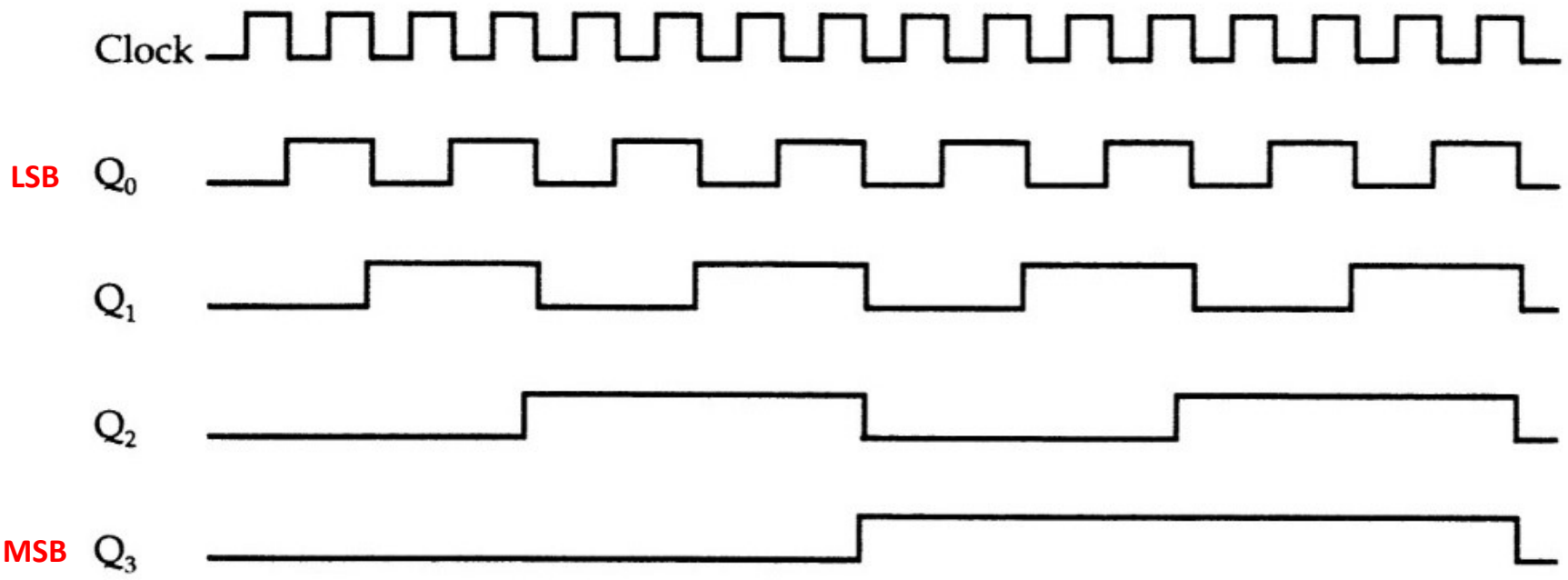
A principal característica de um contador de pulsos é apresentar nas saídas o sistema binário em sequência.



Contador assíncrono crescente
Tabela verdade

Descidas de clock	Saídas Q_0	Q_1	Q_2	Q_3	
1ª	0	0	0	0	(Estado inicial, imposto por CLR = 0)
2ª	1	0	0	0	(Após a 1ª descida de clock: Q_0=1)
3ª	0	1	0	0	(Após a 2ª descida: Q_0=0 e Q_1=1, obtido pela descida de Q_0)
4ª	1	1	0	0	(Q_0=1 e Q_1 permanece igual a1)
5ª	0	0	1	0	(Q_0=0 ⇒ Q_1=0 ⇒ Q_2=1)
6ª	1	0	1	0	(Q_0=1, Q_1 e Q_2 permanecem)
7ª	0	1	1	0	(Q_0=0 ⇒ Q_1=1)
8ª	1	1	1	0	(Q_0=1)
9ª	0	0	0	1	(Q_0=0 ⇒ Q_1=0 ⇒ Q_2=0 ⇒ Q_3=1)
10ª	1	0	0	1	(Q_0=1)
11ª	0	1	0	1	(Q_0=0 ⇒ Q_1=1)
12ª	1	1	0	1	(Q_0=1)
13ª	0	0	1	1	(Q_0=0 ⇒ Q_1=0 ⇒ Q_2=1)
14ª	1	0	1	1	(Q_0=1)
15ª	0	1	1	1	(Q_0=0 ⇒ Q_1=1)
16ª	1	1	1	1	(Q_0=1)
17ª	0	0	0	0	(Q_0=0 ⇒ Q_1=0 ⇒ Q_2=0 ⇒ Q_3=0)

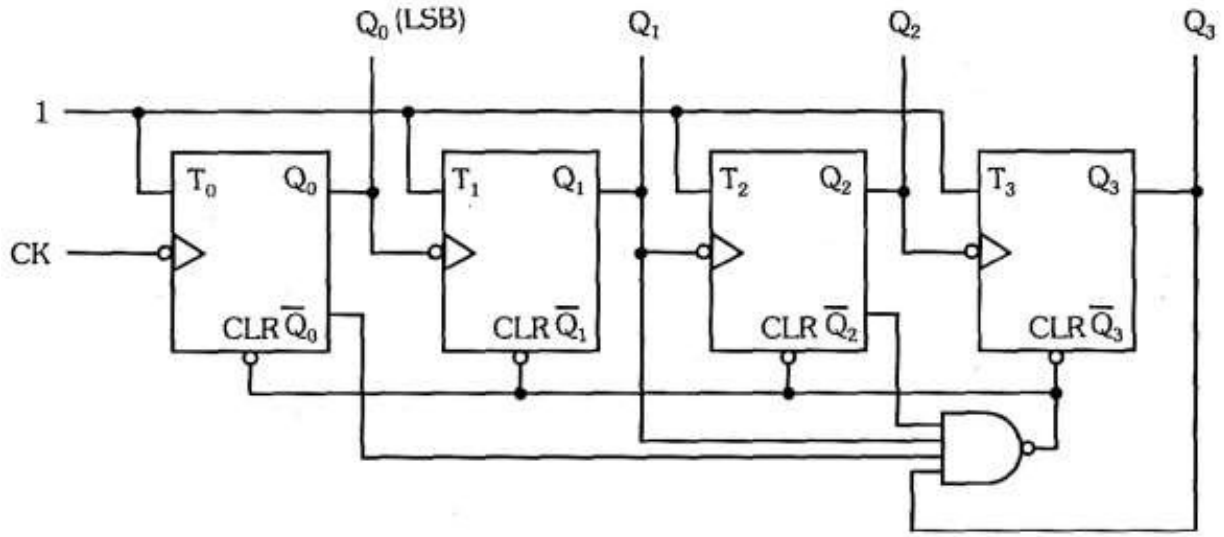
Contador assíncrono crescente
Diagrama de tempo



Contador assíncrono de década

O contador de década é o circuito que efetua a contagem em números binários de 0 a 9 (10 algarismos).

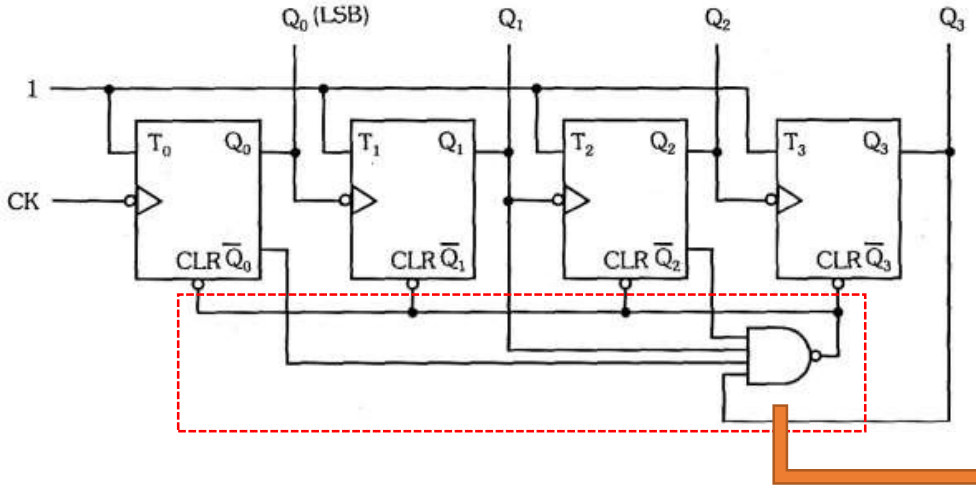
Isso significa acompanhar a sequência do código BCD 8421 de 0000 a 1001.



Contador assíncrono de década

Para que o contador conte somente de 0 a 9, deve-se colocar um nível 0 na entrada clear assim que surgir o caso 10 (1010), ou seja, no 10º pulso.

Após a 10ª descida de clock, o contador tende a assumir o estado $Q_0 = 0$, $Q_1 = 1$, $Q_2 = 0$ e $Q_3 = 1$, neste instante, a entrada CLR (clear) vai para 0, zerando o contador e reiniciando a contagem.

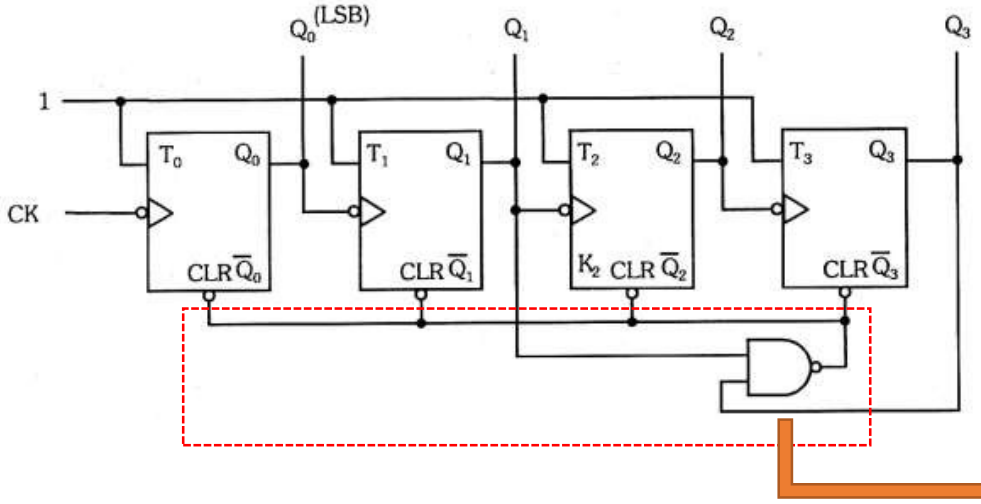


Descidas de clock	Q_3	Q_2	Q_1	Q_0	CLR
1ª	0	0	0	0	1
2ª	0	0	0	1	1
3ª	0	0	1	0	1
4ª	0	0	1	1	1
5ª	0	1	0	0	1
6ª	0	1	0	1	1
7ª	0	1	1	0	1
8ª	0	1	1	1	1
9ª	1	0	0	0	1
10ª	1	0	0	1	1
	1	0	1	0	0

Contador assíncrono de década NAND de 2 entradas

Uma outra forma de obter o mesmo clear ou reset no caso 1010 é utilizar uma porta NAND com menos entradas, ligando apenas Q_3 e Q_1 nesta, pois só serão iguais a 1 simultaneamente neste caso, zerando as saídas do mesmo jeito.

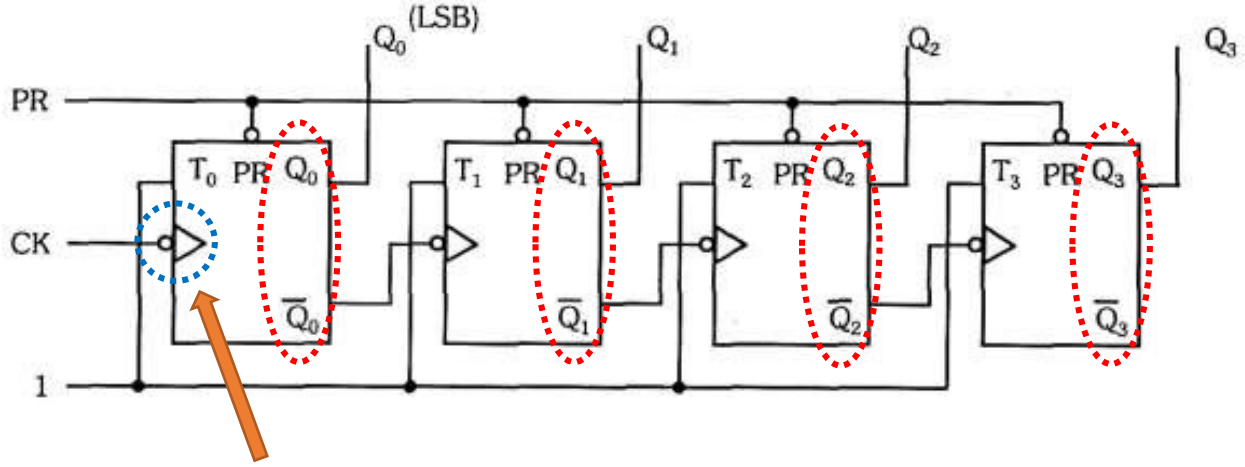
Este contador poderá ser utilizado como divisor de frequência por 10 para uma onda quadrada aplicada à entrada clock, pois possui 10 estados de saída.



Descidas de clock	Q_3	Q_2	Q_1	Q_0	CLR
1ª	0	0	0	0	1
2ª	0	0	0	1	1
3ª	0	0	1	0	1
4ª	0	0	1	1	1
5ª	0	1	0	0	1
6ª	0	1	0	1	1
7ª	0	1	1	0	1
8ª	0	1	1	1	1
9ª	1	0	0	0	1
10ª	1	0	0	1	1
	1	0	1	0	0

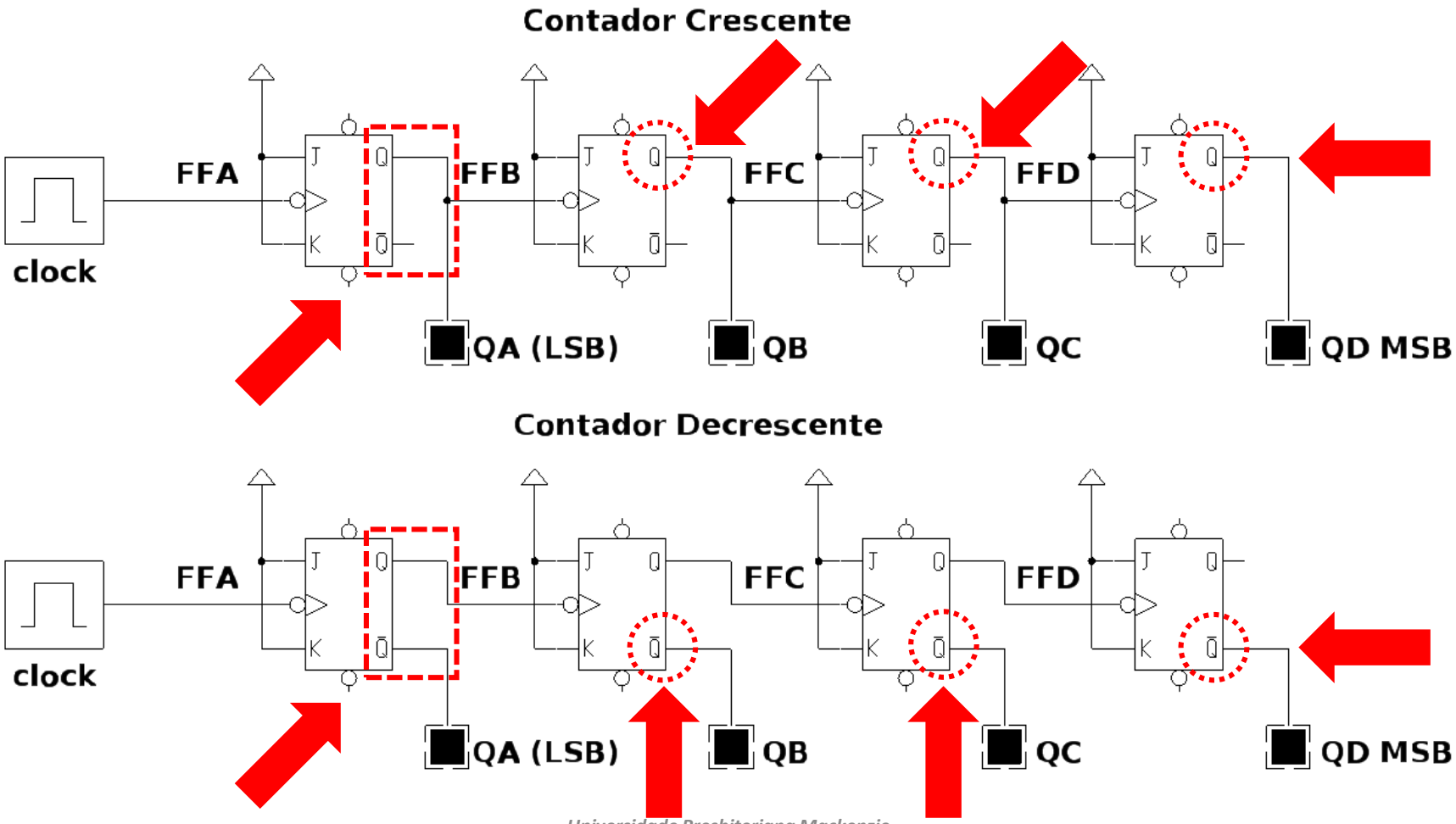
Contador assíncrono decrescente

Um outro modo de montar um contador decrescente é injetando nas entradas de pulso de clock dos flip-flops as saídas complementares, ou seja, pulso de borda de descida ao invés de borda de subida, ou vice-versa, dependendo se as saídas utilizadas são Q ou $\bar{Q}$.



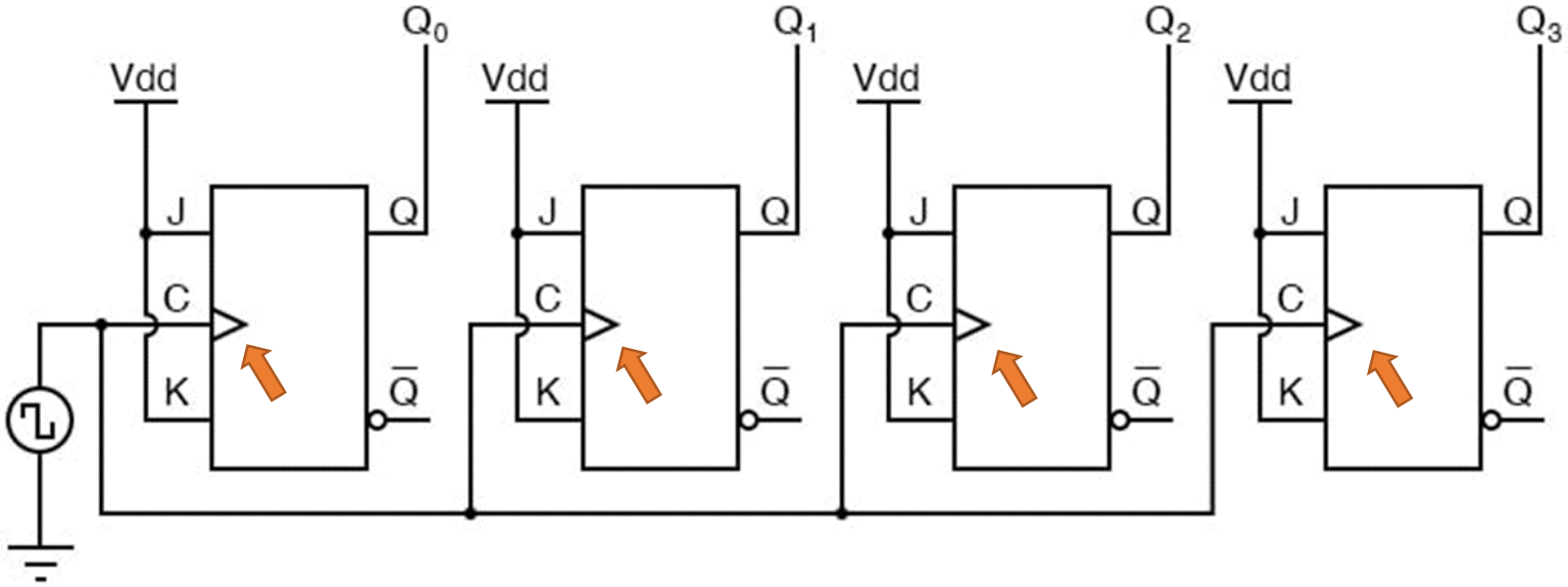
Alterar a forma do pulso de clock do primeiro flip-flop não altera o comportamento do contador!

Contador assíncrono crescente *vs* decrescente



Contador síncrono

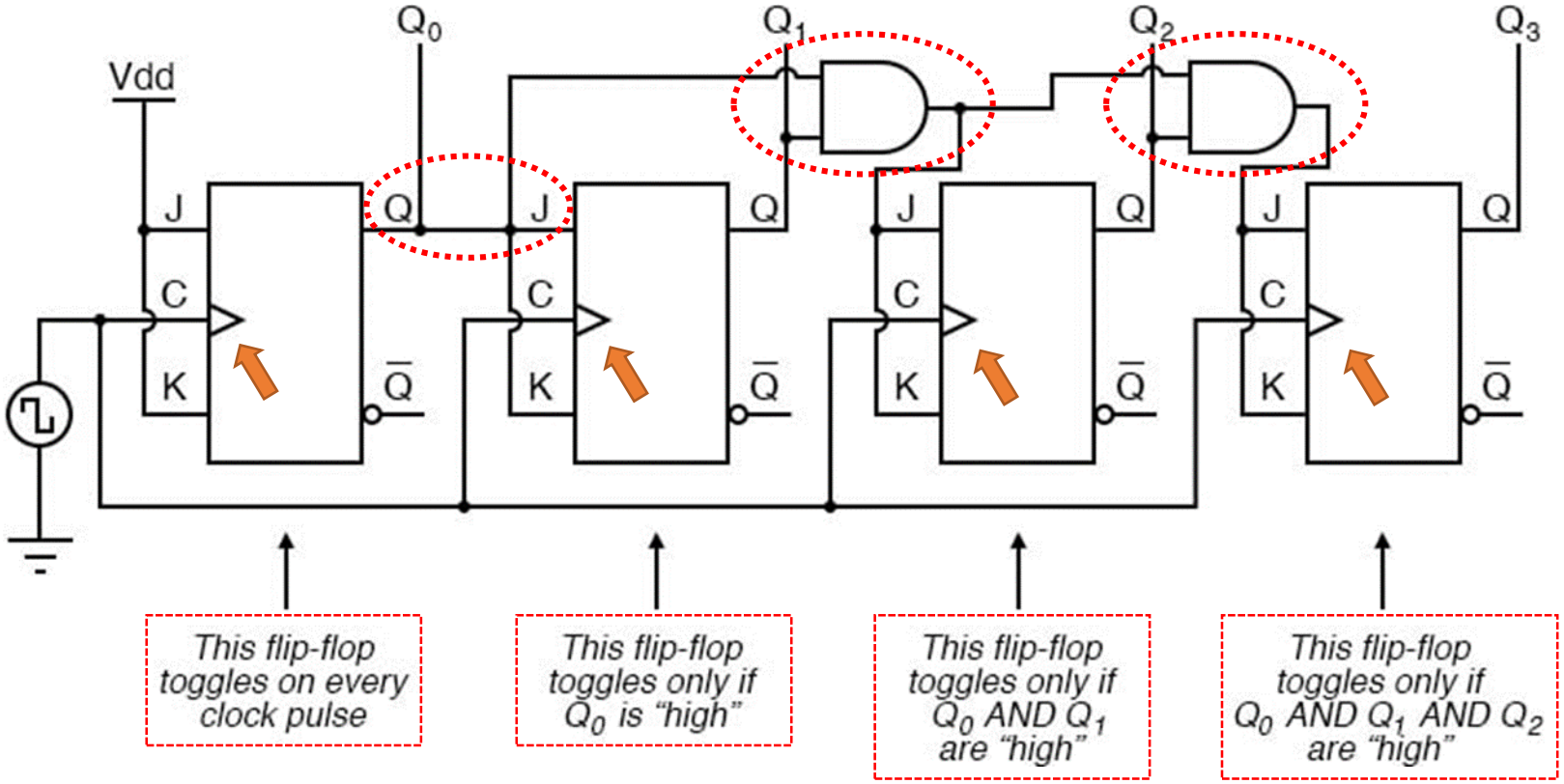
No contador síncrono, todas as entradas de clock dos flip-flops são alimentadas pela mesma fonte de sinal, eliminando assim o problema de tempo de propagação do sinal.



⚠ No entanto, este circuito como está não irá funcionar como um contador, sendo necessário acrescentar portas AND para trocar o estado da entrada T (J e K).

Fonte: allaboutcircuits.com

Contador síncrono crescente

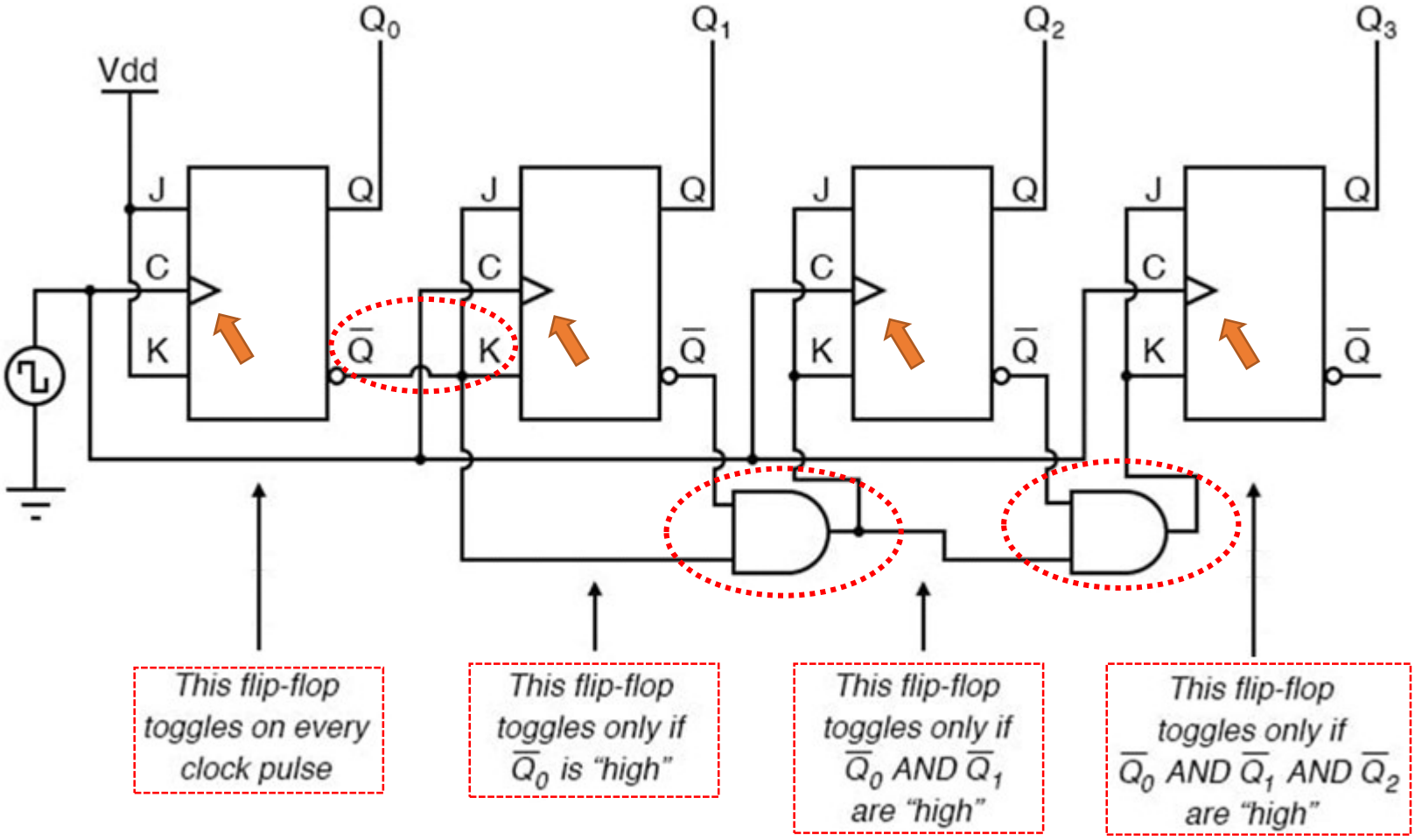


⚠ Alterar a forma do pulso de clock não altera o comportamento do contador!

⚠ Para configurar o contador, é necessário trocar o estado das entradas T (J e K).

Fonte: allaboutcircuits.com

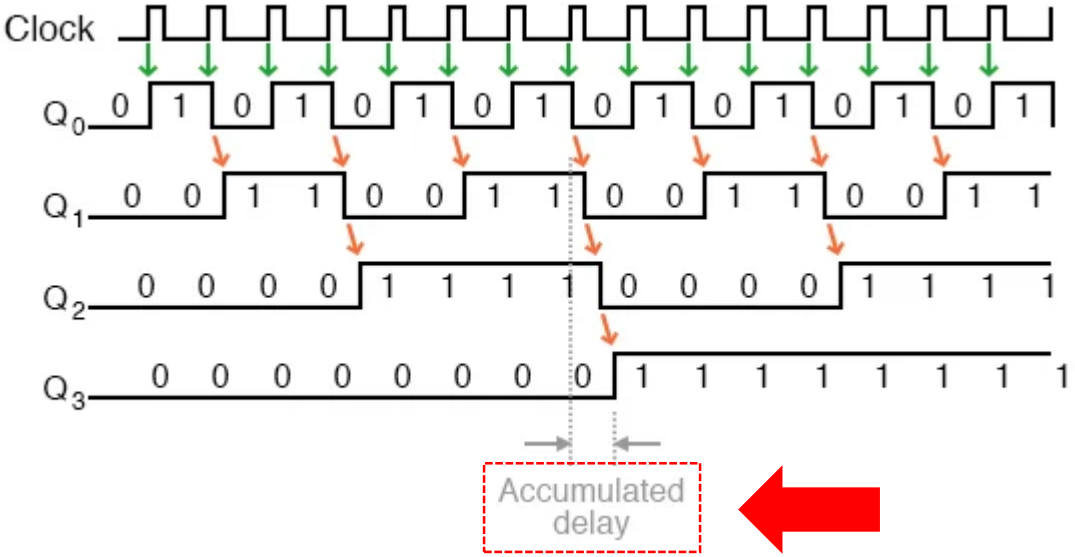
Contador síncrono decrescente



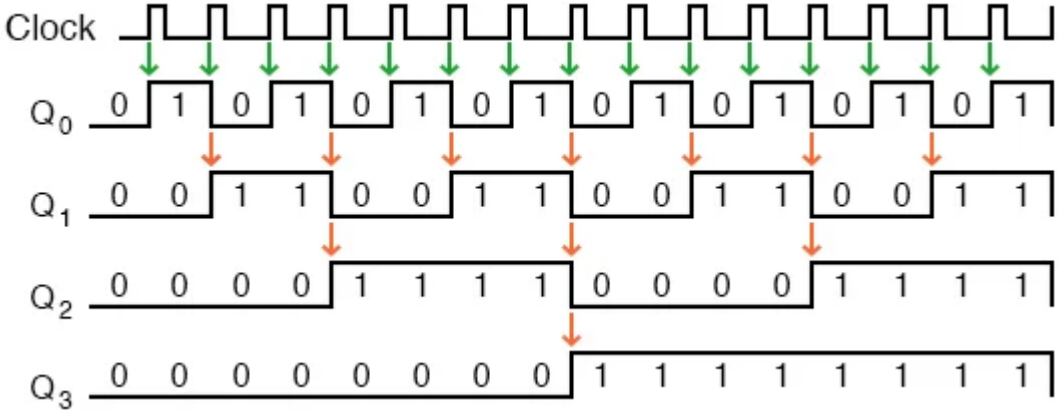
Fonte: allaboutcircuits.com

Contadores assíncronos *vs* síncronos

Contador Assíncrono



Contador Síncrono



Fonte: allaboutcircuits.com

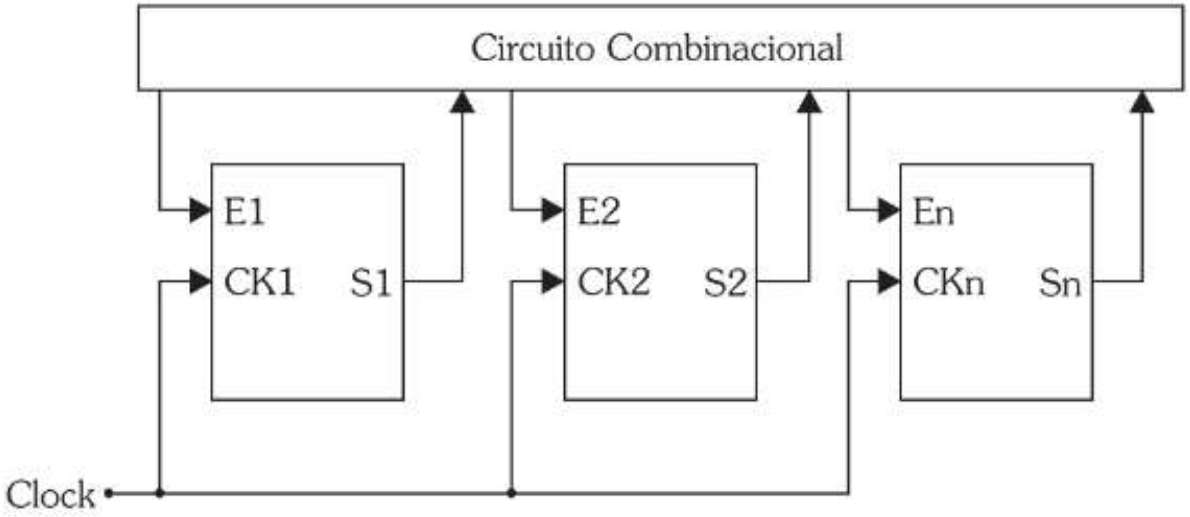
Projeto de circuitos sequenciais síncronos

O projeto de circuitos sequenciais síncronos pode ser obtido seguindo-se estes passos:

1) Elaborar um diagrama de estados;
2) Minimizar o número de estados no diagrama de estados, se necessário;
3) Escrever a tabela de estados, com os estados atuais e futuros;
4) Atribuir a cada estado uma combinação de variáveis de estado (flip-flops) em função dos estados atuais e futuros;
5) Construir a tabela de transição de estados relacionando os níveis lógicos do tipo de flip-flops em uso em função dos estados atuais e futuros;
6) Relacionar os níveis lógicos que as entradas dos flip-flops devem assumir, após o pulso de clock, gerando as saídas futuras previstas (as entradas dos flip-flops que são as saídas do circuito combinacional);
7) Montar o mapa de Veitch-Karnaugh para cada uma das entradas dos flip-flops do circuito, com o auxílio da tabela de excitação de cada tipo de flip-flop em uso;
8) Obter as expressões lógicas das saídas do circuito combinacional através de mapas de Veitch-Karnaugh;
9) Elaborar o diagrama lógico do circuito em que todos os elementos de memória (flip-flops) recebem o mesmo sinal de relógio.

Contadores síncronos

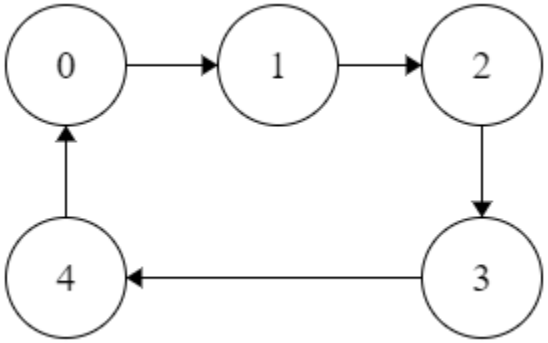
Possuem as entradas de clock curto-circuitadas, acionando assim todos os flip-flops simultaneamente.



Contadores síncronos - exemplo 1

Criação de um contador módulo 5 síncrono crescente com flip-flops tipo D sensível a borda de descida.

Como se trata de um contador módulo 5 (cinco estados), três flip-flops são suficientes para o projeto, já que sua sequencia é formada apenas pelos estados 0 a 4 (000 a 100).



O diagrama de estados não apresenta em sua malha principal todos os estados possíveis do contador (estão faltando os estados 5, 6 e 7). Estes estados representam estados secundários que serão analisados posteriormente.

Fonte: LOURENÇO, A.C., et al. Circuitos Digitais. São Paulo: Érica, 1996

Contadores síncronos - exemplo 1

		Entradas do circuito combinacional						Saídas do circuito combinacional		
Estado atual	**Estado futuro**	**Saídas atuais dos flip-flops**			**Saídas futuras dos flip-flops**			**Entradas dos flip-flops**		
		Q_C	**Q_B**	**Q_A**	**Q_C**	**Q_B**	**Q_A**	**D_C**	**D_B**	**D_A**
0	**1**	0	0	0	0	0	1	0	0	1
1	**2**	0	0	1	0	1	0	0	1	0
2	**3**	0	1	0	0	1	1	0	1	1
3	**4**	0	1	1	1	0	0	1	0	0
4	**0**	1	0	0	0	0	0	0	0	0
5	**X**	1	0	1	X	X	X	X	X	X
6	**X**	1	1	0	X	X	X	X	X	X
7	**X**	1	1	1	X	X	X	X	X	X

Fonte: LOURENÇO, A.C., et al. Circuitos Digitais. São Paulo: Érica, 1996

Contadores síncronos - exemplo 1

Q_C \ $Q_B Q_A$	00	01	11	10
0	**0**	**0**	**1**	**0**
1	**0**	**X**	**X**	**X**

$$D_C = Q_B \cdot Q_A$$

Q_C \ $Q_B Q_A$	00	01	11	10
0	**0**	**1**	**0**	**1**
1	**0**	**X**	**X**	**X**

$$D_B = \overline{Q_B} \cdot Q_A + Q_B \cdot \overline{Q_A}$$
$$D_B = Q_B \oplus Q_A$$

Q_C \ $Q_B Q_A$	00	01	11	10
0	**1**	**0**	**0**	**1**
1	**0**	**X**	**X**	**X**

$$D_A = \overline{Q_C} \cdot \overline{Q_A}$$

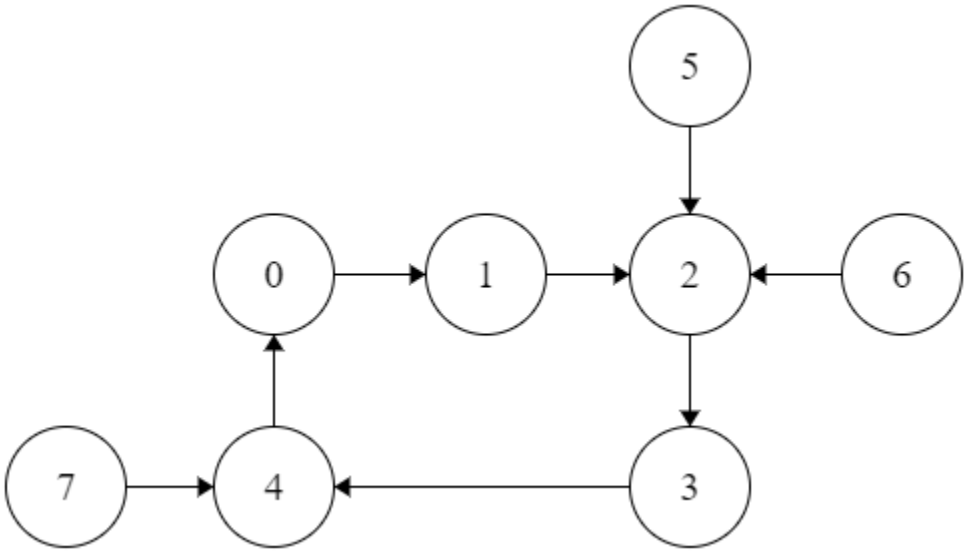
Fonte: LOURENÇO, A.C., et al. Circuitos Digitais. São Paulo: Érica, 1996

Contadores síncronos - exemplo 1

		Entradas do circuito combinacional						Saídas do circuito combinacional		
Estado atual	Estado futuro	Saídas atuais dos flip-flops			Saídas futuras dos flip-flops			Entradas dos flip-flops		
		Q_C	Q_B	Q_A	Q_C	Q_B	Q_A	D_C	D_B	D_A
0	**1**	0	0	0	0	0	1	0	0	1
1	**2**	0	0	1	0	1	0	0	1	0
2	**3**	0	1	0	0	1	1	0	1	1
3	**4**	0	1	1	1	0	0	1	0	0
4	**0**	1	0	0	0	0	0	0	0	0
5	**2**	1	0	1	0	1	0	0	1	0
6	**2**	1	1	0	0	1	0	0	1	0
7	**4**	1	1	1	1	0	0	1	0	0

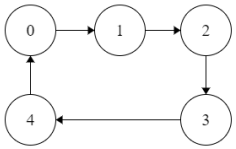
Fonte: LOURENÇO, A.C., et al. Circuitos Digitais. São Paulo: Érica, 1996

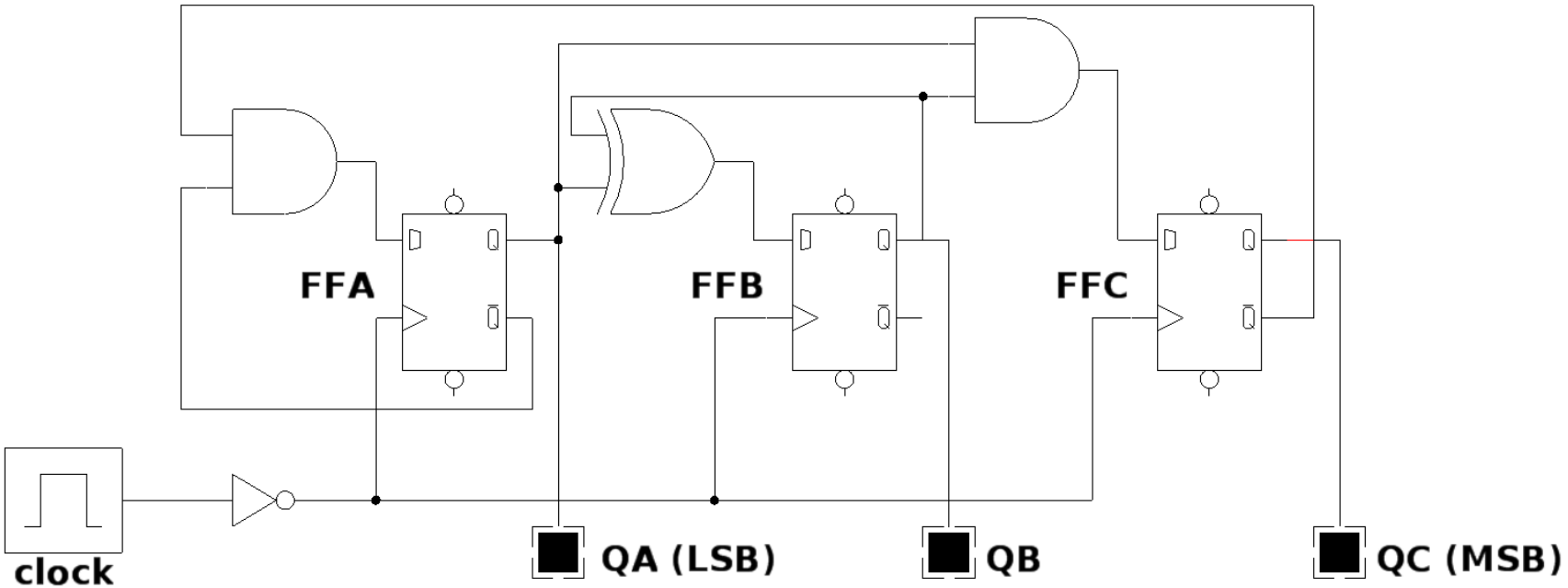
Contadores síncronos - exemplo 1



Fonte: LOURENÇO, A.C., et al. Circuitos Digitais. São Paulo: Érica, 1996

Contadores síncronos - exemplo 1



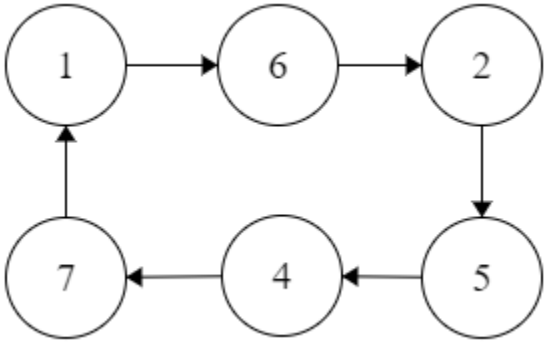


Fonte: LOURENÇO, A.C., et al. Circuitos Digitais. São Paulo: Érica, 1996

Contadores síncronos - exemplo 2

Criação de um contador síncrono com flip-flops tipo D sensível a borda de subida.

Como se trata de um contador cujo maior número é o 7, três flip-flops são suficientes para o projeto.



O diagrama de estados não apresenta em sua malha principal todos os estados possíveis do contador (estão faltando os estados 0 e 3). Estes estados representam estados secundários que serão analisados posteriormente.

Contadores síncronos - exemplo 2

		Entradas do circuito combinacional						Saídas do circuito combinacional		
Estado atual	Estado futuro	Saídas atuais dos flip-flops			Saídas futuras dos flip-flops			Entradas dos flip-flops		
		Q_C	Q_B	Q_A	Q_C	Q_B	Q_A	D_C	D_B	D_A
0	**X**	0	0	0	X	X	X	X	X	X
1	**6**	0	0	1	1	1	0	1	1	0
2	**5**	0	1	0	1	0	1	1	0	1
3	**X**	0	1	1	X	X	X	X	X	X
4	**7**	1	0	0	1	1	1	1	1	1
5	**4**	1	0	1	1	0	0	1	0	0
6	**2**	1	1	0	0	1	0	0	1	0
7	**1**	1	1	1	0	0	1	0	0	1

Contadores síncronos - exemplo 2

Q_C \ $Q_B Q_A$	00	01	11	10
0	X	1	X	1
1	1	1	0	0

$$D_C = \overline{Q_C} + \overline{Q_B}$$

Q_C \ $Q_B Q_A$	00	01	11	10
0	X	1	X	0
1	1	0	0	1

$$D_B = \overline{Q_C} \cdot Q_A + Q_C \cdot \overline{Q_A}$$
$$D_B = Q_C \oplus Q_A$$

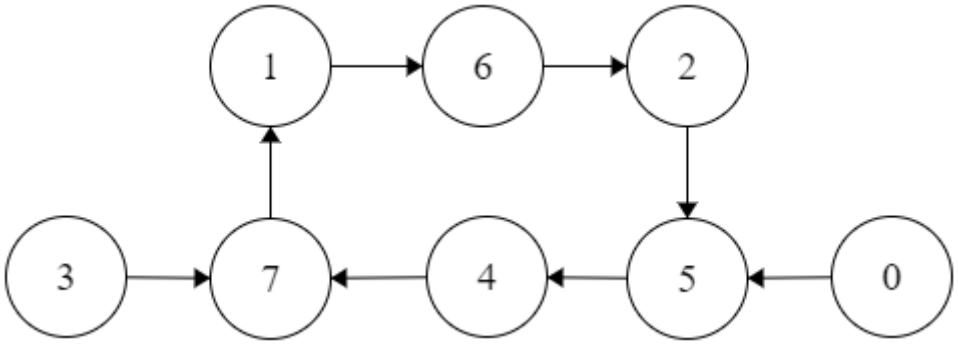
Q_C \ $Q_B Q_A$	00	01	11	10
0	X	0	X	1
1	1	0	1	0

$$D_A = \overline{Q_C} \cdot Q_B + \overline{Q_B} \cdot \overline{Q_A} + Q_B \cdot Q_A$$

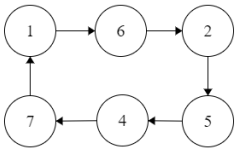
Contadores síncronos - exemplo 2

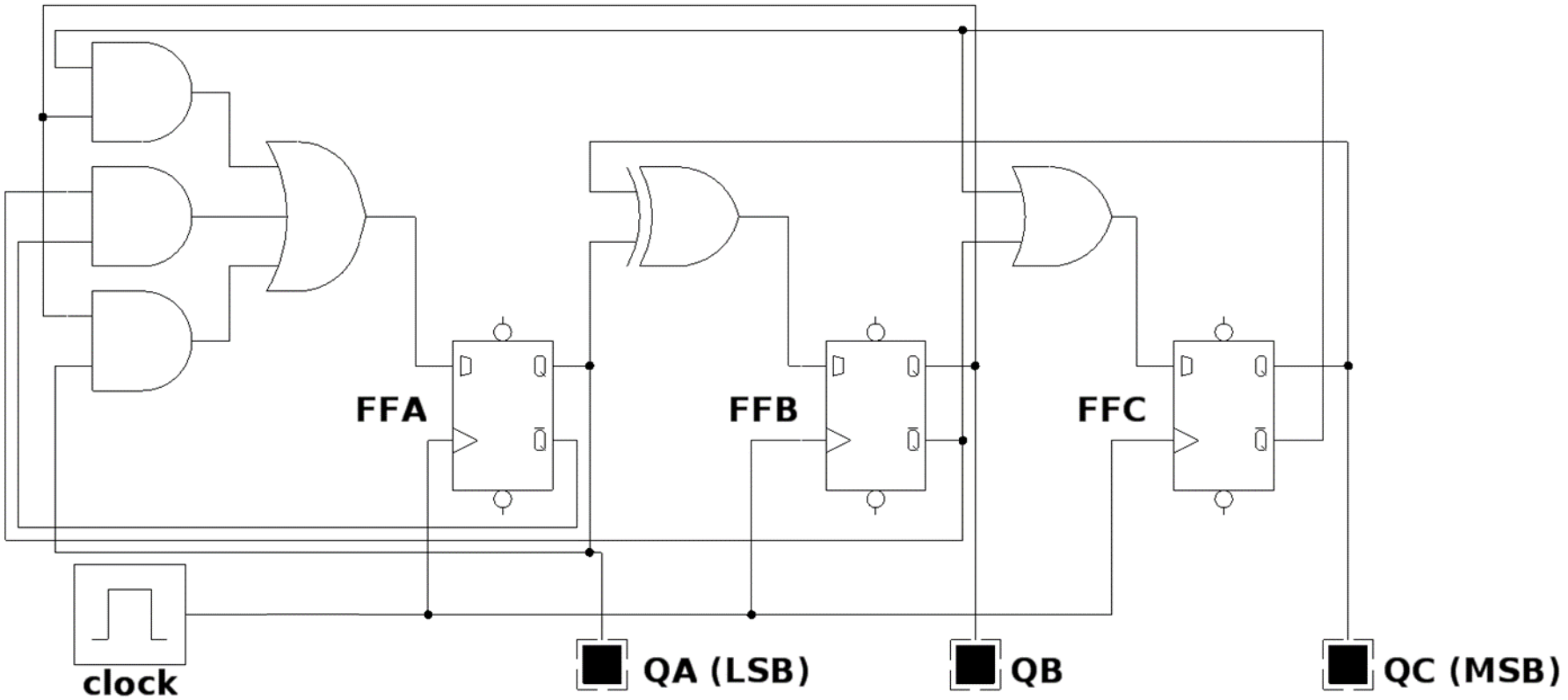
		Entradas do circuito combinacional						Saídas do circuito combinacional		
Estado atual	Estado futuro	Saídas atuais dos flip-flops			Saídas futuras dos flip-flops			Entradas dos flip-flops		
		Q_C	Q_B	Q_A	Q_C	Q_B	Q_A	D_C	D_B	D_A
0	**5**	0	0	0	1	0	1	1	0	1
1	**6**	0	0	1	1	1	0	1	1	0
2	**5**	0	1	0	1	0	1	1	0	1
3	**7**	0	1	1	1	1	1	1	1	1
4	**7**	1	0	0	1	1	1	1	1	1
5	**4**	1	0	1	1	0	0	1	0	0
6	**2**	1	1	0	0	1	0	0	1	0
7	**1**	1	1	1	0	0	1	0	0	1

Contadores síncronos - exemplo 2



Contadores síncronos - exemplo 2





FIM