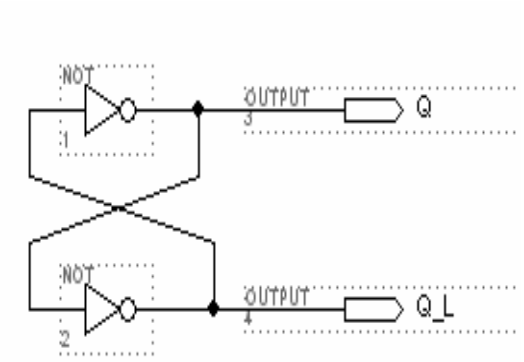
# Circuitos Sequenciais

- A este circuito dá-se o nome de **bi-estável** porque possui dois estados (situações) estáveis:
- Pode usar-se uma única variável de estado (sinal **Q**) para definir o estado do circuito.
- Logo,há 2 estados possíveis: Q=0 e Q=1.



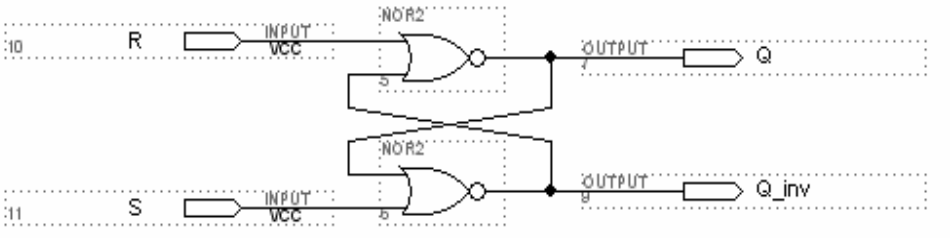

As ***latches*** e os ***flips-flops*** são os blocos elementares com os quais se constrói a maior parte dos circuitos sequenciais.

Um ***flip-flop*** é um dispositivo sequencial que amostra as suas entradas e que altera as suas saídas apenas em instantes determinados por um sinal de **relógio**.

Uma ***latch*** é um dispositivo sequencial que observa todas as suas entradas continuamente e altera as suas saídas em qualquer momento, **independentemente** de qualquer sinal de **relógio**.

# O Bloco elementar *latch*

- Uma ***latch*** é um dispositivo sequencial que observa todas as suas entradas continuamente e altera as suas saídas em qualquer momento, **independentemente** de qualquer sinal de **relógio**.
- O ***latch RS***



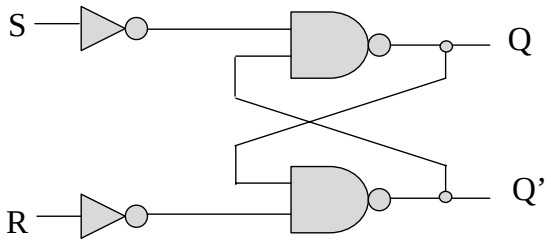

**S**=set, **R**=reset)

# Flip-Flop

- Um flip-flop é um latch, controlado pelo clock. Isto significa que o valor atual da saída Q do elemento de memória não é relacionado com o valor atual da entrada.
- São circuitos que possuem dois estados estáveis, ou seja, são circuitos biestáveis:
  - Flip = atirar ao alto ou movimento rápido
    Circuito assume estado lógico alto
  - Flop = queda brusca ou repentina
    Circuito assume estado lógico baixo
- Um flip-flop é um circuito digital básico que armazena um bit de informação.
- A saída de um flip-flop só muda de estado durante a transição do sinal de clock.
- Existem vários tipos:
  - Flip-Flop D, Flip-Flop D com reset assíncrono
  - Flip-Flop D com reset síncrono, Flip-Flop D com clock enable
  - Flip-Flop T, SR, JK
  - outros.....

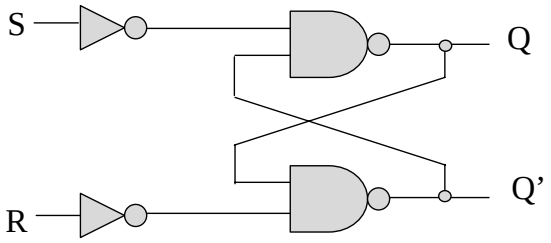
# Flip-Flop RS básico (Assíncrono)

- São construídos a partir de portas que tem a seguinte configuração:




- A realimentação mostra que o novo estado na saída depende do estado anterior que está injetado na entrada.

# Flip-Flop RS básico (Assíncrono)



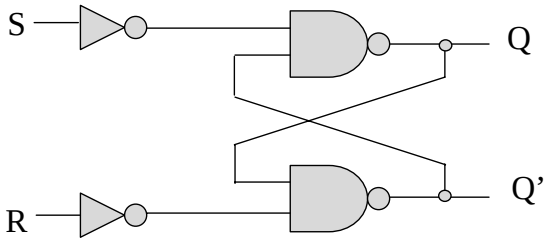

- Tabela da verdade

| S | R | $Q_A$ | Q | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | → $Q = Q_A$ |
| 0 | 0 | 1 | 1 | → $Q = Q_A$ |
| 0 | 1 | 0 | 0 | → $Q = 0$ |
| 0 | 1 | 1 | 0 | → $Q = 0$ |
| 1 | 0 | 0 | 1 | → $Q = 1$ |
| 1 | 0 | 1 | 1 | → $Q = 1$ |
| 1 | 1 | 0 | X | → não permitido |
| 1 | 1 | 1 | X | → não permitido |

Resumindo:

| S | R | Q |
|---|---|---|
| 0 | 0 | $Q_A$ |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | X |

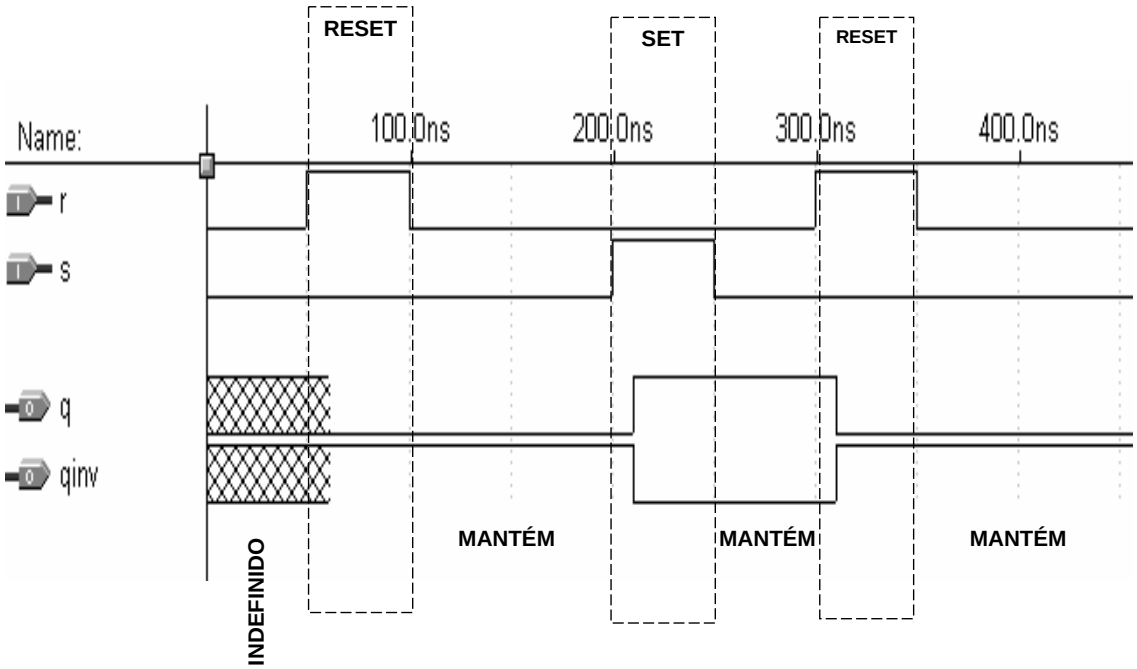
# Flip-Flop RS básico (Assíncrono)




| S | R | Q |
|---|---|---|
| 0 | 0 | $Q_A$ |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | X |

- Desvantagens
  - As saídas mudam imediatamente após as entradas mudarem
    - Isto é um problema lógico, pois as mudanças de dados nas células serão casuais, não sendo possível controlar a operação. A solução é FF RS Síncrono
  - Quando as entradas estão em nível alto as saídas serão indefinidas
    - Este problema será resolvido com um FF JK

# Simulação/Latch RS




# *Simulação/Estudo de caso*

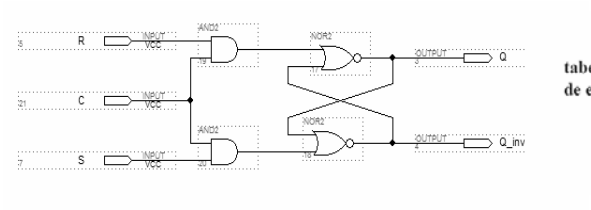
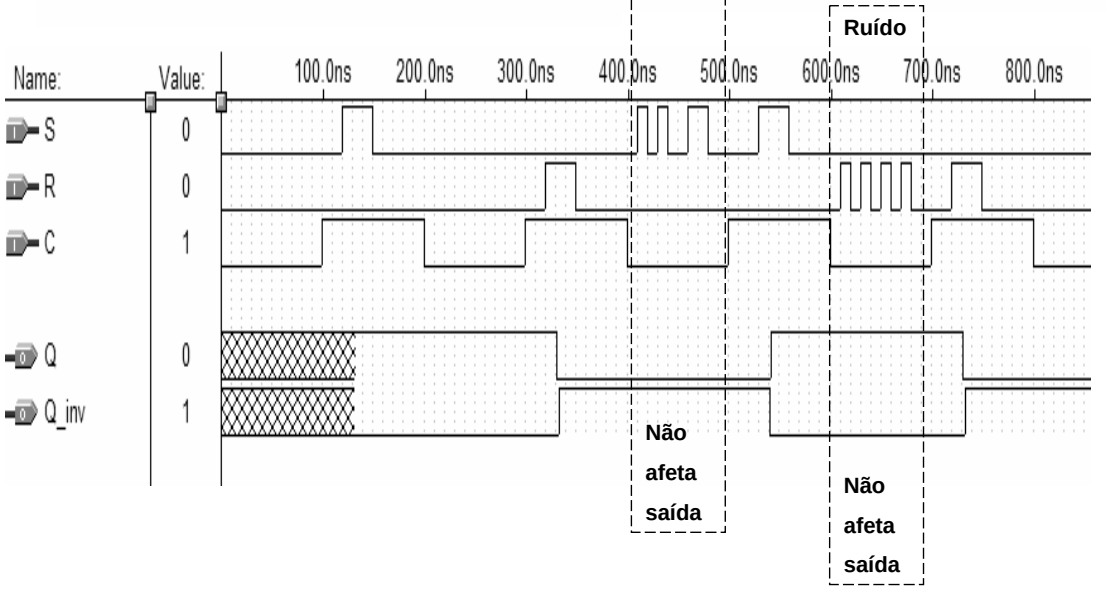



tabela de transição de estados

| C | R | S | $Q_{t+1}$ | comentário |
|---|---|---|---|---|
| 0 | X | X | $Q_t$ | mantém estado anterior |
| 1 | 0 | 0 | $Q_t$ | mantém estado anterior |
| 1 | 0 | 1 | 1 | estado set |
| 1 | 1 | 0 | 0 | estado reset |
| 1 | 1 | 1 | - | proibido |

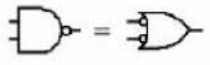


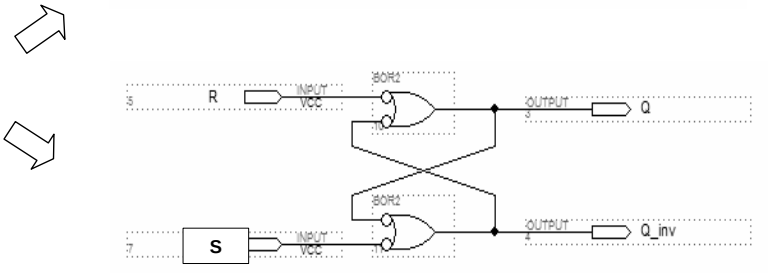
# O *latch RS/Outras implementações*

Teoremas de DeMorgan:

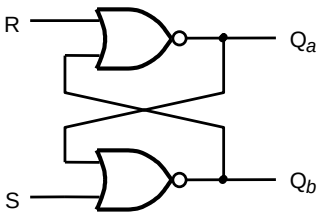
Teorema 2:

Ir $\overline{X \cdot Y} = \overline{X} + \overline{Y}$





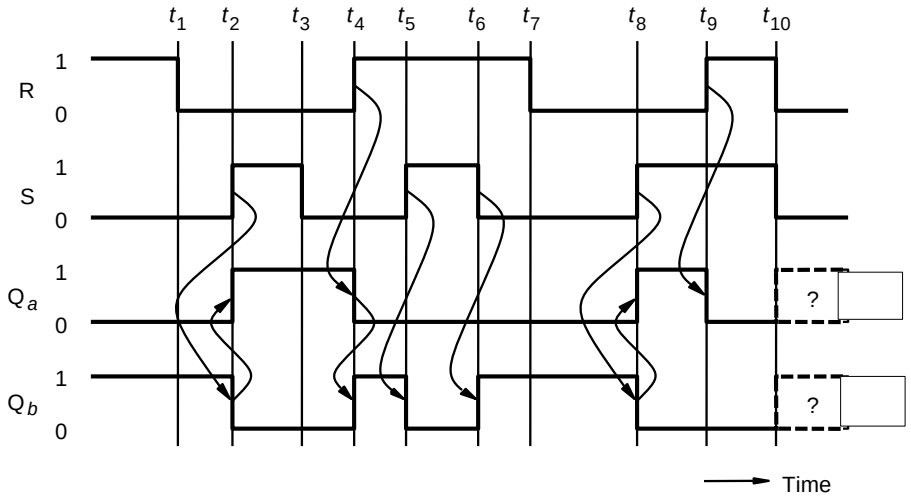

# Latch construído com portas NOR




(a) Circuito

| S | R | $Q_a$ | $Q_b$ | |
|---|---|---|---|---|
| 0 | 0 | 0/11/0 | | (sem alteração) |
| 0 | 1 | 0 | 1 | |
| 1 | 0 | 1 | 0 | |
| 1 | 1 | 0 | 0 | |

(b) Tabela Verdade

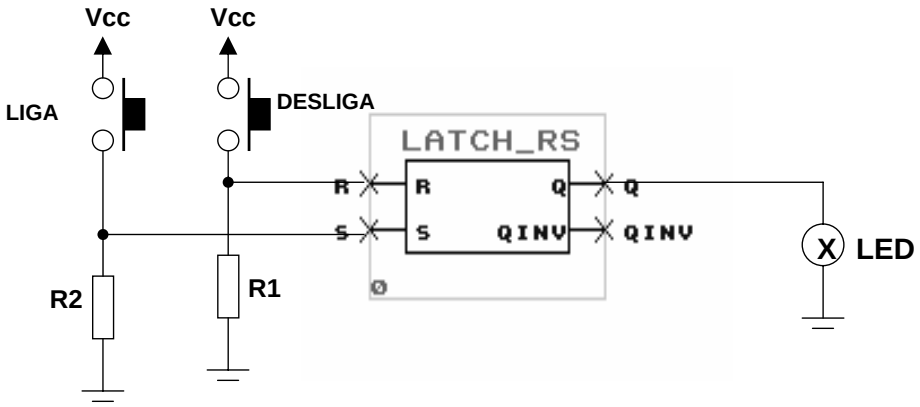



(c) Diagrama de Tempo

# O *latch RS/Exemplo*

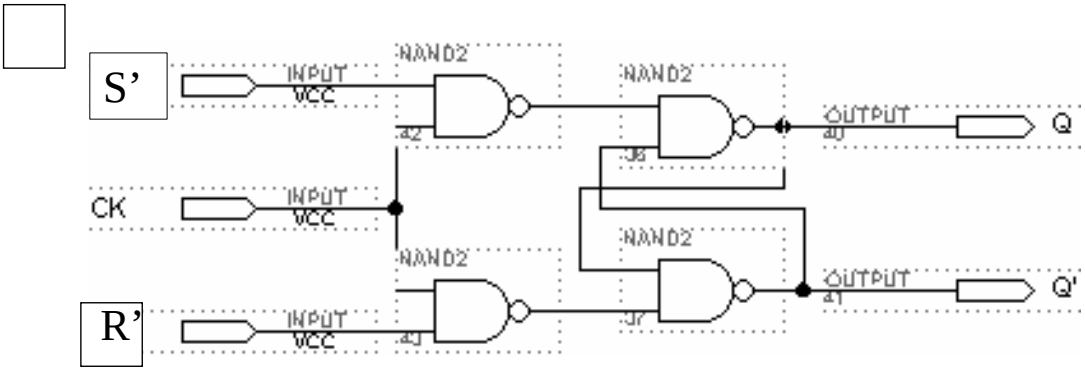
**•Circuito para controle de LED**

Para ligar o LED pressiona-se o botão LIGA → O Latch RS passa para o estado de "SET" → R = '0' e S = '1' => Q = '1'
Para desligar o LED pressiona-se o botão DESLIGA → O Latch RS passa para o estado de "RESET" → R = '1' e S = '0' => Q = '0'



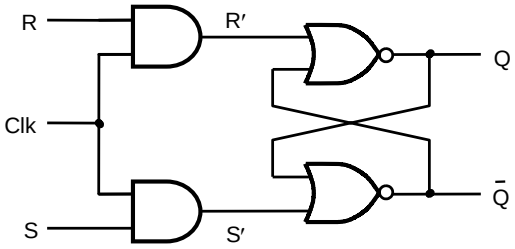

# Flip-Flops RS Síncrono

- Tem como característica um terceira entrada denominada pulso de controle (*clock* ou CK) agregada a um estágio de entrada adicional.
- O clock faz com que o flip-flop RS atualize seus estados.



# Flip-Flops RS Síncrono

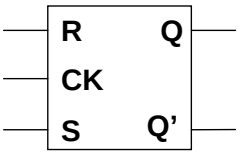
- Com o clock em nível zero (CK=0), as saídas anteriores são mantidas.
- Com o clock em nível um (CK=1), o flip-flop RS síncrono opera como um flip-flop RS básico.



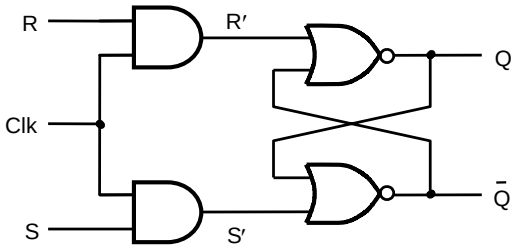

(a) Circuito

| CK | S | R | Q | Q' |
|---|---|---|---|---|
| 0 | X | X | Qa | Qa |
| 1 | 0 | 0 | 0/1 | 0/1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | ? | ? |

FlipFlop RS Sinc



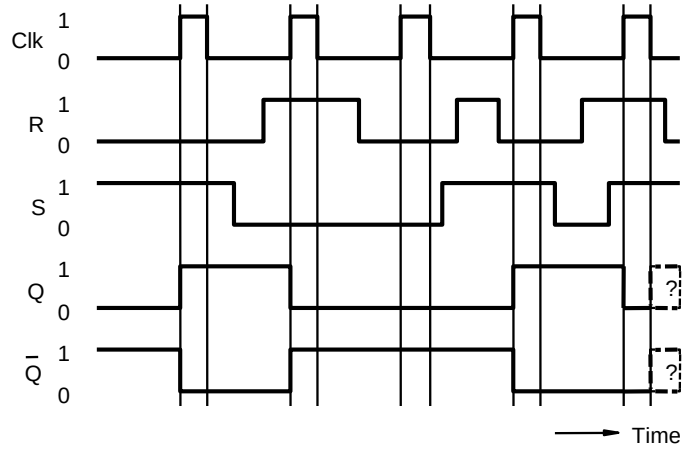

# Latch SR com clock (gated)



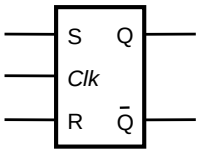

(a) Circuito

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

(b) Tabela Verdade



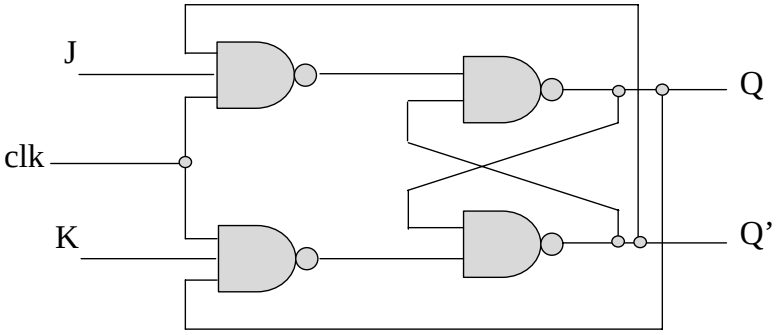

(c) Timing diagram



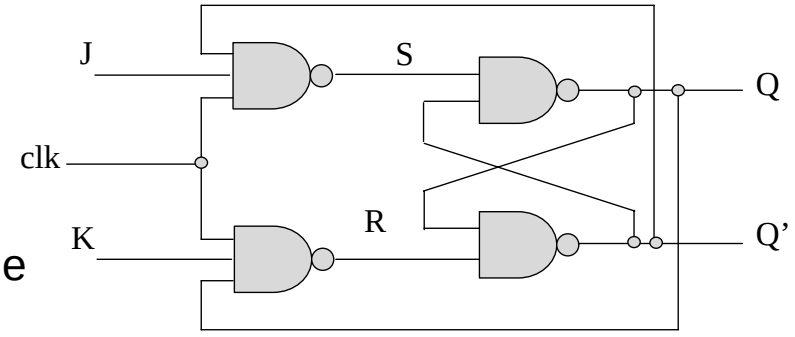

(d) Símbolo Gráfico

# Flip-Flop JK

- Tipo de flip-flop RS aprimorado, onde o erro lógico foi eliminado.




# Flip-Flop JK




- Tabela da Verdade

| 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 | $Q_A$ | $= 0$ |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | |
| 1 | 0 | 1 | 0 | 0 | 0 | $Q_A$ | $= 1$ |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | $= Q'_A$ |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | $= Q'_A$ |

1

$Q'_A$

# Flip-Flop JK

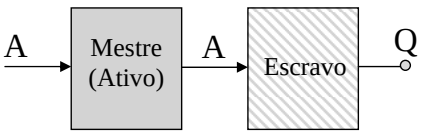
- Obs: Para J=K=1
  - Para obter a última opção da tabela da verdade é necessário que os atrasos das portas sejam convenientes, caso contrário pode haver oscilação das saídas.
  - O problema é totalmente resolvido com FF JK mestre escravo.
  - Obs: atualizar transparencias

Tabela JK Simplificada

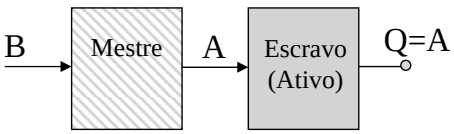
| CK | J | K | Q | Q' |
|---|---|---|---|---|
| 0 | X | X | Qa | Qa' |
| 1 | 0 | 0 | Qa | Qa' |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | $Q_A'$ | $Q_A$ |

# Flip-Flop Mestre-Escravo

- Possui este nome devido aos dois blocos internos com os quais ele é formado
- Estes dois blocos representam dois circuitos separados de latch.
- O latch mestre é utilizado para aceitar a entrada do bit de dado A no flip-flop
- O valor de A é armazenado no mestre e, então, transferido para o escravo em um tempo posterior.
- Ambas as entradas são sincronizadas pelo sinal de clock



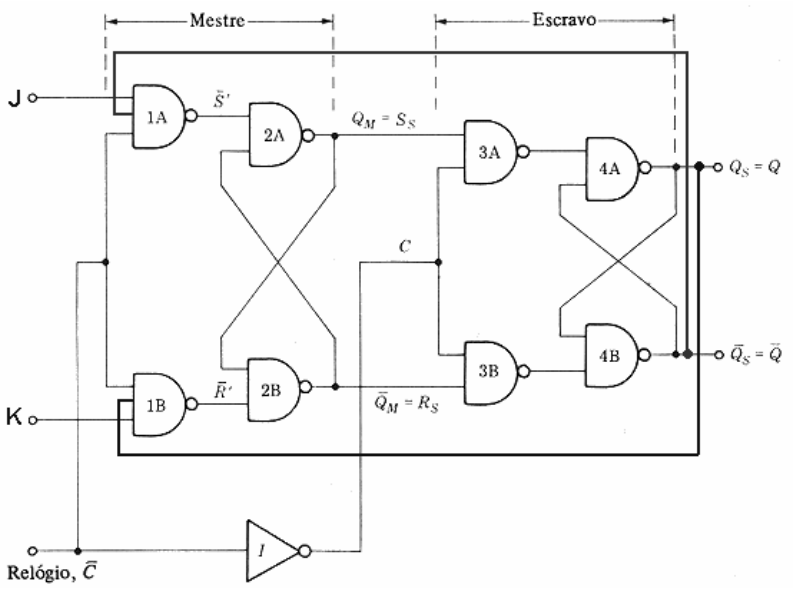

Mestre ativo




Escravo ativo

# Flip-Flop JK Mestre-Escravo

- Para eliminar a oscilação do flip-flop JK, foram combinados dois flip-flops RS como no circuito a seguir, denominado flip-flop JK Master-Slave (Mestre-Escravo).




# Flip-Flop JK Mestre-Escravo

- Quando o clock for ‘0’ o circuito de entrada está inativo, logo as entradas do escravo não serão alteradas e a saída do flip-flop JK – MS não será alterada.
- Quando o clock for ‘1’ o mestre operará como um flip-flop JK normal, mas o escravo estará inativo e as saídas não serão alteradas.
- Quando o clock voltar para o nível ‘0’ o circuito mestre para de funcionar. O circuito escravo volta a funcionar (habilitado) e as saídas do mestre no instante que o clock volta a zero são transferidas para o escravo.
- Isto é muito interessante porque não vai haver mais que uma mudança na saída do flip-flop JK mestre-escravo por ciclo de clock.

# Flip-Flop JK Mestre-Escravo

- Tabela da verdade:

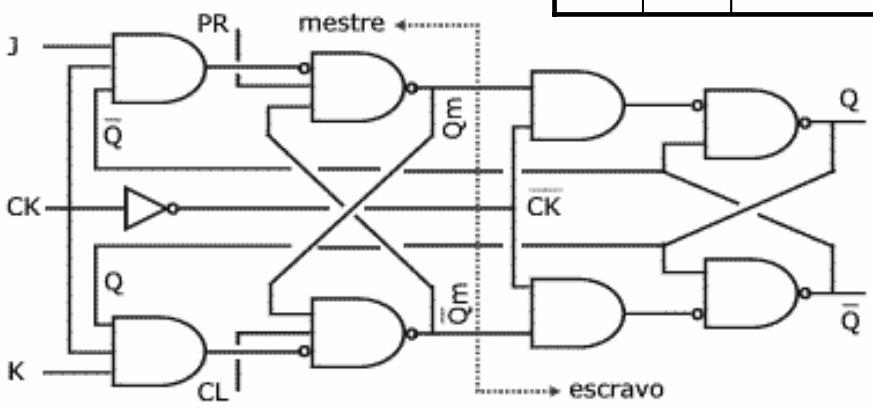
| J | K | Q |
|---|---|---|
| 0 | 0 | $Q_A$ |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | $Q'_A$ |

Note que este é um circuito sensível à descida do clock. Para continuarmos um que seja sensível à subida do clock, basta colocarmos um inversor na entrada do clock.

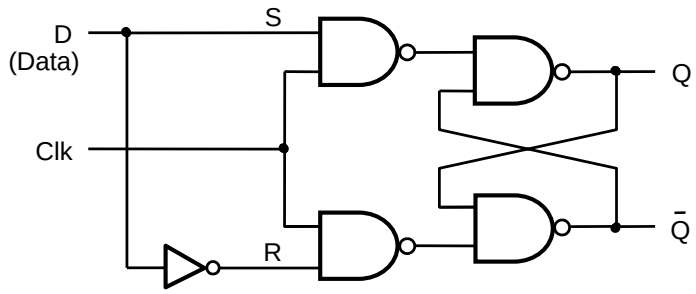
# Flip-Flop JK Mestre-Escravo

- Com entrada Preset e Clear

| CL | PR | Q |
|---|---|---|
| 0 | 0 | Não permitido |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | Funcionamento normal |



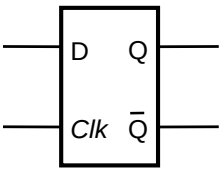
# Latch D Gated



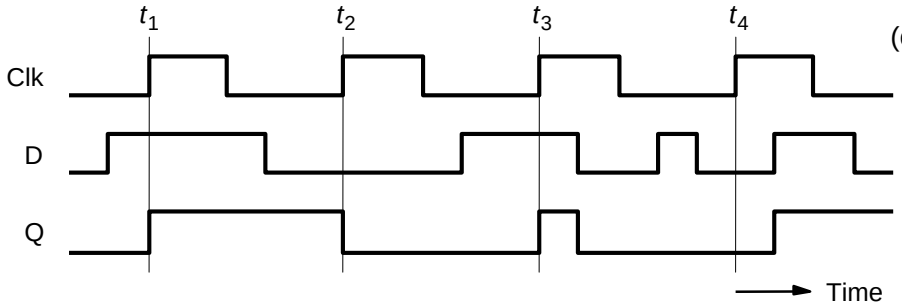

(a) Circuito

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

(b) Tabela Verdade

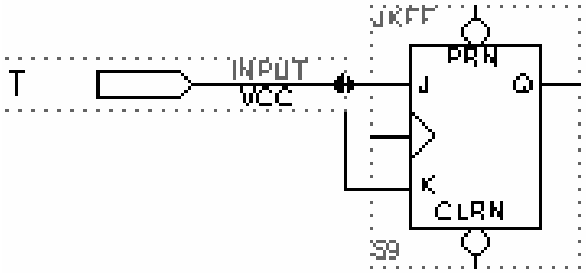



(c) Símbolo Gráfico



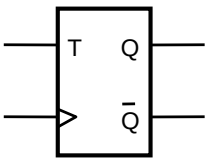

(d) Diagrama de Tempo

# Flip-Flop tipo T (toogle)

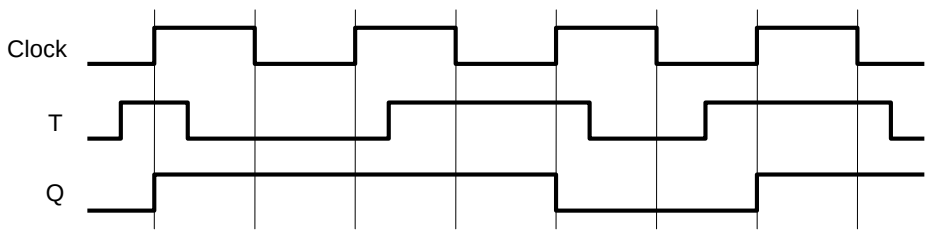



| T | Q(*t* + 1) |
|---|---|
| 0 | Q(*t*) |
| 1 | $\bar{Q}$(*t*) |

(b) Tabela Verdade



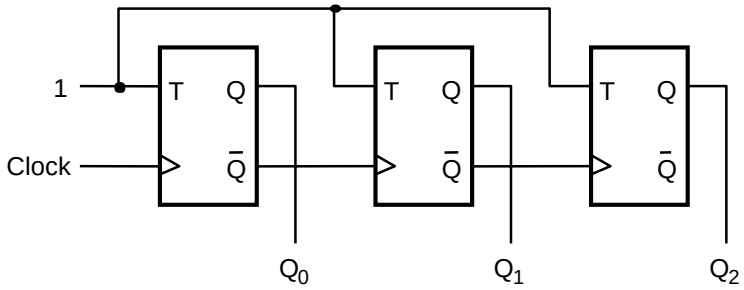

(c) Símbolo Gráfico



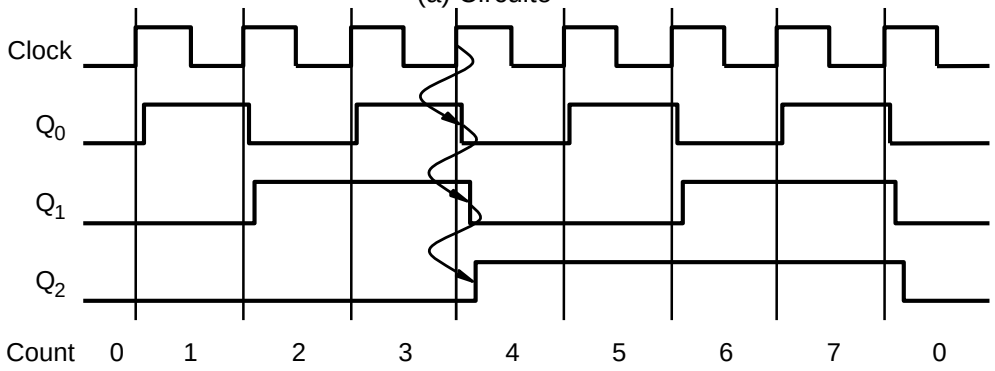

(d) Diagrama de tempo

# Contadores

## Contador de 3 bits up-counter

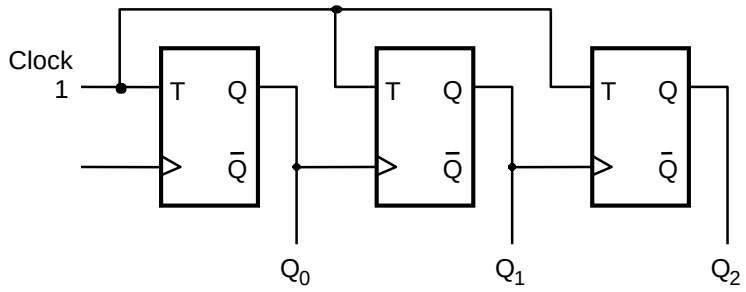



(a) Circuito

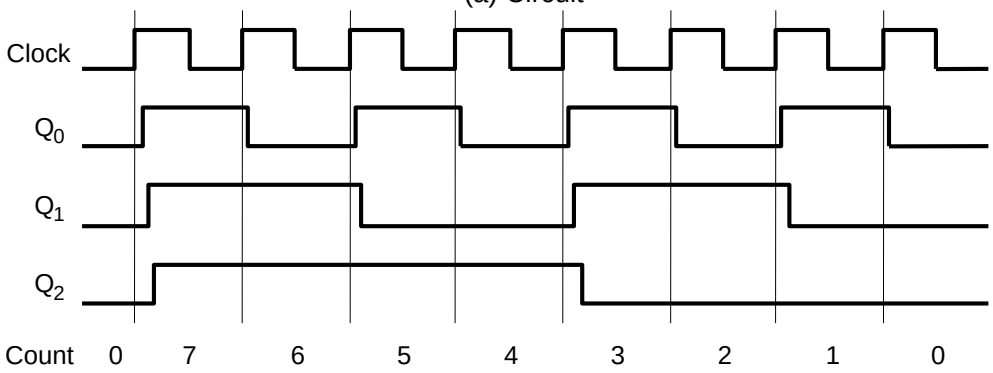



(b) Diagrama de tempo

## Contador de 3 bits down-counter




(a) Circuit



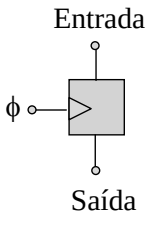

(b) Timing diagram

# Registrador

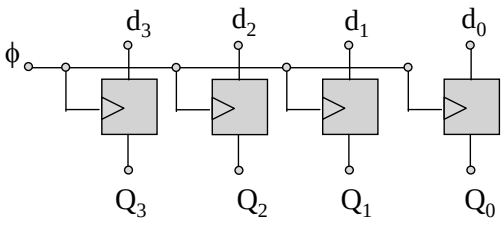
- Um registrador é um elemento lógico utilizado para armazenar uma palavra binária de n-bits.
- Praticamente, todos os grandes sistemas digitais utilizam registradores para armazenar dados importantes.

# Registrador básico

- Pode-se construir um registrador com a conexão de n células de armazenamento, de um único bit, em paralelo, possibilitando ler ou escrever em todas as células simultaneamente.
- Vamos examinar um registrador de 4 bits que pode armazenar e bits de dado.



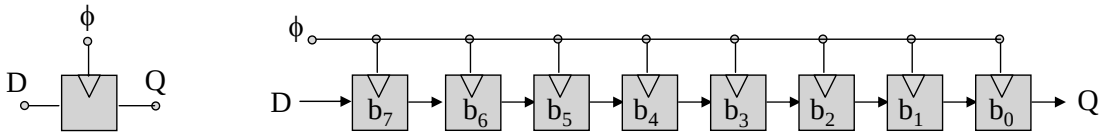

Célula individual




Registrador de 4 bits

# Registrador de Deslocamento

- Um registrador de deslocamento é projetado para mover bits para as células vizinhas, enquanto houver pulsos de clock.

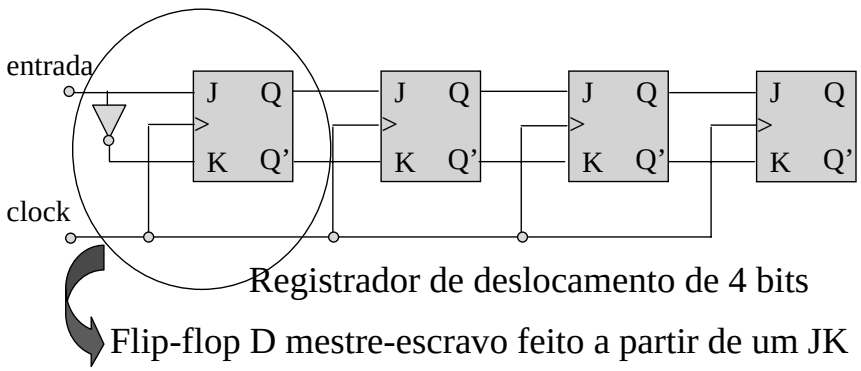



  - Como todas as células são controladas pelo mesmo sinal de clock $\phi$ , todas elas são carregadas ao mesmo tempo

# Registrador de Deslocamento

| J | K | Q |
|---|---|---|
| 0 | 0 | $Q_A$ |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | $Q'_A$ |

- Como sabemos o flip-flop pode armazenar um estado de saída, até que um novo pulso chegue ao clock.
- Registrador de deslocamento trata-se de um certo número de flip-flops RS ou JK mestre-escravo, sendo cada J ou S ligada à saída Q do flip-flop anterior e cada entrada K ou R ligada à saída Q' do flip-flop anterior.
- O primeiro flip-flop é ligado com um flip-flop tipo D

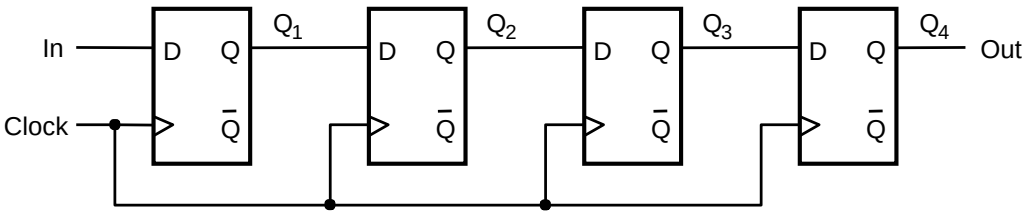



Registrador de deslocamento de 4 bits

Flip-flop D mestre-escravo feito a partir de um JK

Registrador de deslocamento com entrada e saída serial

Registrador é um conjunto de n Flip-Flops




(a) Circuito

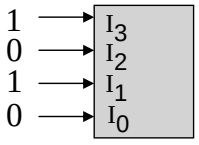
| | In | $Q_1$ | $Q_2$ | $Q_3$ | $Q_4$ = Out |
|---|---|---|---|---|---|
| $t_0$ | 1 | 0 | 0 | 0 | 0 |
| $t_1$ | 0 | 1 | 0 | 0 | 0 |
| $t_2$ | 1 | 0 | 1 | 0 | 0 |
| $t_3$ | 1 | 1 | 0 | 1 | 0 |
| $t_4$ | 1 | 1 | 1 | 0 | 1 |
| $t_5$ | 0 | 1 | 1 | 1 | 0 |
| $t_6$ | 0 | 0 | 1 | 1 | 1 |
| $t_7$ | 0 | 0 |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

(b) Exemplo de uma seqüência

# Conversor Série-Paralelo

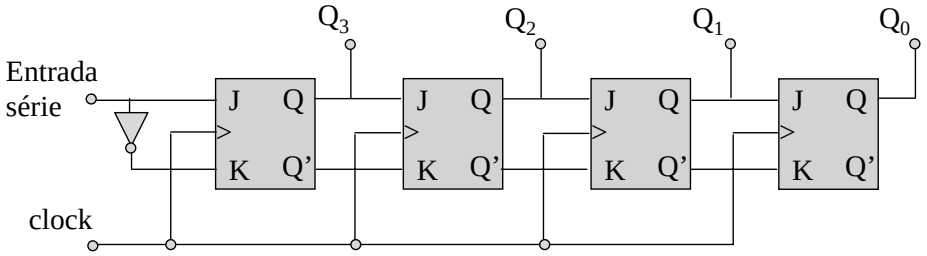
- Informação Paralela : os bits se apresentam simultaneamente.
  - Apresenta tantos fios quanto o número de bits
  - Caso: Queremos transmitir a informação

$$I_3\ I_2\ I_1\ I_0 = 1010$$

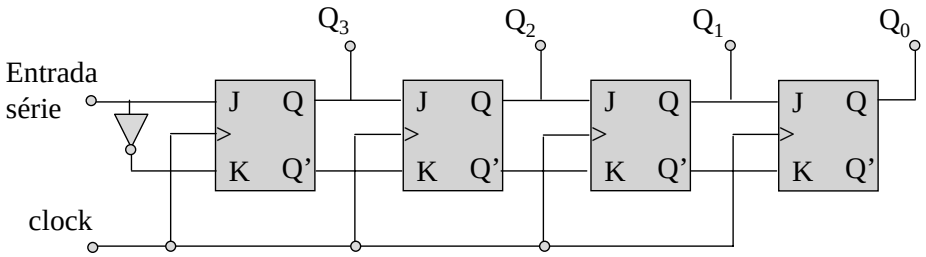


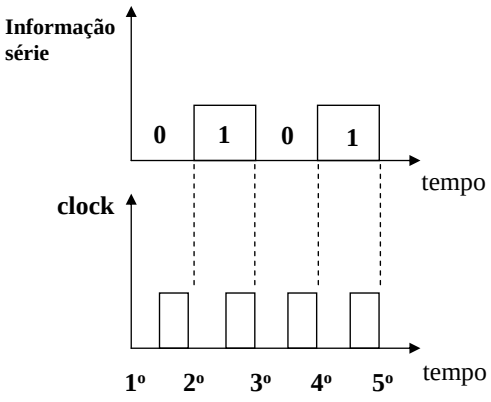
# Conversor Série-Paralelo

- Informação Serial : os bits são transmitidos por um único fio.
  - Podemos construir um conversor sério-paralelo com um registrador de deslocamento




# Conversor Série-Paralelo




| Informação | descidas do clock | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|---|---|
| $I_0 = 0$ | 1º pulso | 0 | 0 | 0 | 0 |
| $I_1 = 1$ | 2º pulso | 1 | 0 | 0 | 0 |
| $I_2 = 0$ | 3º pulso | 0 | 1 | 0 | 0 |
| $I_3 = 0$ | 4º pulso | 1 | 0 | 1 | 0 |

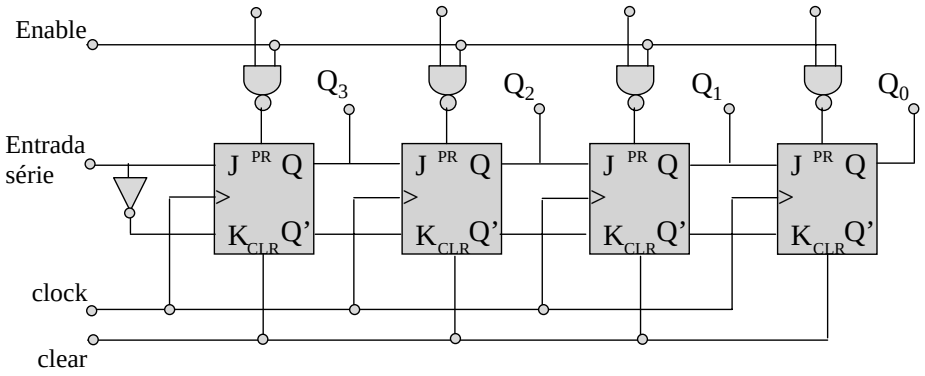
Obs: A cada descida do clock as informações são deslocadas para o próximo flip-flop

# Conversor Paralelo-Série

| J | K | Q |
|---|---|---|
| 0 | 0 | $Q_A$ |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | $Q'_A$ |

| CL | PR | Q |
|---|---|---|
| 0 | 0 | Não permitido |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | Funcionamento normal |

- Podemos realizar o processo inverso, ou seja, entrar com uma palavra com n bits e “retirar” bit a bit com pulsos de clock.
- Para isto utilizamos as funções Preset e Clear dos flip-flops JK mestre-escravo.
- O circuito utilizado está mostrado abaixo:

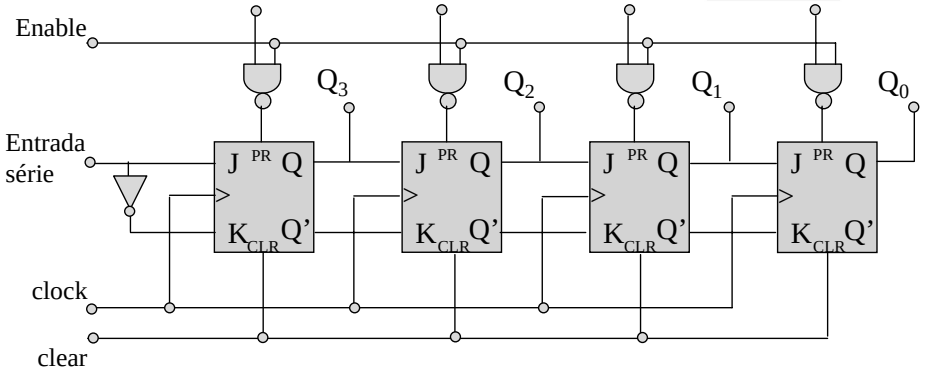




# Conversor Paralelo-Série

| J | K | Q |
|---|---|---|
| 0 | 0 | $Q_A$ |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | $Q'_A$ |

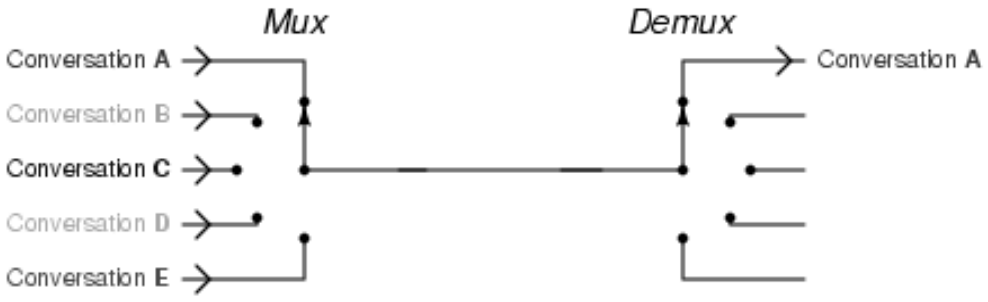
| CL | PR | Q |
|---|---|---|
| 0 | 0 | Não permitido |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | Funcionamento normal |


Para inserirmos a palavra paralela:

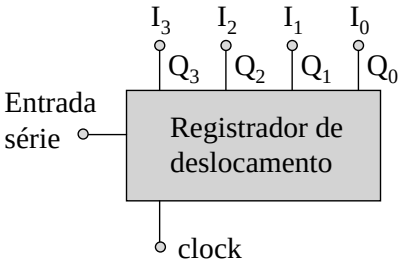
- Primeiro damos um pulso em zero no Clear e todas as saídas vão para estado zero.
- Contudo, enquanto o enable estiver em zero, a entrada preset estará em ‘1’, e as saídas permanecerão em zero.
- Quando a entrada ENABLE for para ‘1’, as saídas assumirão os valores presentes nas entradas PRESET, ou seja, quando ENABLE=‘1’ $\rightarrow$ $Q_3$ = PR3; $Q_2$ = PR2; $Q_1$ = PR1; $Q_0$ = PR0
- Então para externar serialmente os bits, aplica-se pulsos ao clock.

# Multiplexador

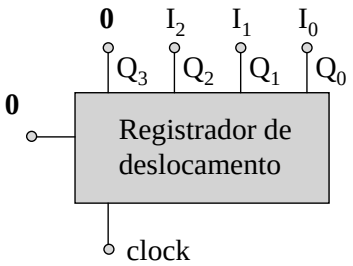



# Divisor por 2

- Considere o estado inicial:




  - Agora considere a entrada série em zero e um pulso de clock aplicado. Assim, temos:



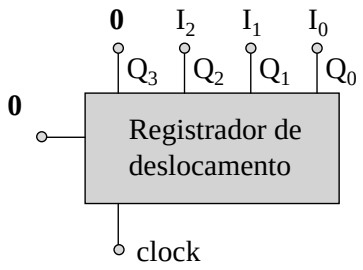

**Ou seja, o número na saída é uma potência de 2 menor.**

# Exemplo: (divisão por 2)

- Considere: $I_3I_2I_1I_0 = (1010)_2 = (10)_{10}$

Depois do pulso de clock teremos:

$$0I_2I_1I_0 = 0101 = (5)_{10}$$




# Outros registradores

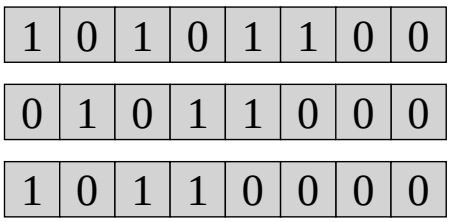
- Um registrador de deslocamento de carregamento paralelo permite ao usuário armazenar toda a palavra em uma transição de clock.
- Este registrador pode possuir duas operações de deslocamento controladas pelos sinais SHR(*Shift Right*) e SHL(*Shift Left*).
- Operação Shift Right:
  - A operação de deslocamento a direita é iniciada fazendo SHR=1.
  - Neste caso cada bit é movido uma posição para a direita
  - O bit mais a esquerda é forçado para o valor 0 (zero).

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | (a) Condição inicial |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | (b) Após um SHR |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | (c) Após mais um SHR |

Obs: A operação SHR pode ser utilizada para dividir o conteúdo do registrador por uma potência de 2

# Outros registradores

- Operação Shift Left:
  - A operação de deslocamento a direita é iniciada fazendo SHL=1.
  - Neste caso cada bit é movido uma posição para a esquerda
  - O bit mais a direita é forçado para o valor 0 (zero).




(a) Condição inicial

(b) Após um SHL

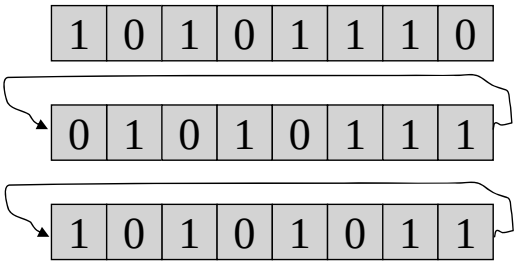
(c) Após mais um SHL

Obs: A operação SHL pode ser utilizada para multiplicar o conteúdo do registrador por uma potência de 2

Obs: As operações de divisão e multiplicação usando os operadores SHL e SHR só funciona corretamente para números decimais pares inteiros.

# Outros registradores

- Operação ROR (Rotate Right)
  - Rotacionamento para a direita
  - Ao contrário do SHR o bit mais a direita é transferido para o outro lado do registrador.



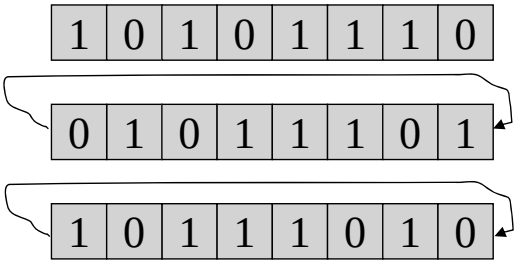

(a) Condição inicial

(b) Após um ROR

(c) Após mais um ROR

# Outros registradores

- Operação ROL (Rotate Left)
  - Rotacionamento para a esquerda
  - Ao contrário do SHL o bit mais a esquerda é transferido para o outro lado do registrador.




(a) Condição inicial

(b) Após um ROL

(c) Após mais um ROL