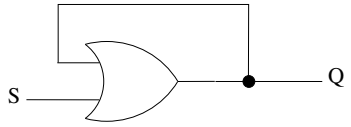
Memória

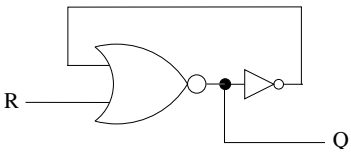
Última revisão em 11 de maio de 2016

Um importante componente dos computadores é a memória. Seria possível termos circuitos que funcionam como memória, no sentido de "armazenar" um determinado dado? Como fazer com que esse dado permaneça inalterado até decidirmos modificá-lo? Circuitos com capacidade de armazenamento podem ser obtidos considerando-se realimentação, ou seja, fazendo com que as saídas de um circuito alimentem parte de suas entradas. Essa ideia pode ser vista inicialmente nos dois exemplos a seguir.

Exemplo 1: Considere uma porta OU com ambas as entradas inicialmente em 0. Nesta situação, a saída é também 0. Suponha que de alguma forma conectamos a saída em uma das entradas (ver figura abaixo). O circuito mantém se em estado estável. Em seguida, suponha que mudamos o valor da outra entrada (S) para 1. O que acontece com a saída Q? O que acontece se, em seguida, colocamos o valor de S de volta para 0?



Exemplo 2: Considere uma porta NÃO-OU cuja saída está conectada a um inversor. Este circuito é uma outra forma de representar a função lógica OU. Suponha que inicialmente ambas as entradas da porta NÃO-OU estão em 0. Nesta situação, a saída Q da porta NÃO-OU é 1 e a saída do inversor é 0. Suponha que a saída do inversor é conectada em uma das entradas da porta NÃO-OU (ver figura abaixo). O circuito mantém-se em estado estável. Suponha que em seguida mudamos o valor da outra entrada (R) da porta NÃO-OU para 1. O que acontece com a saída Q? O que acontece se, em seguida, colocamos o valor de R de volta para 0?



Os exemplos acima são chamados de *set latch* e *reset latch*, respectivamente. No primeiro, após o estado Q passar para 1 (set), ele não pode mais ser alterado. No segundo, após Q passar para 0, não mais pode ser alterado. Pelo fato de não podermos mudar o estado desses dispositivos mais de uma vez, eles tem possibilidade de uso muito limitado. Podemos, porém, usar idéia similar para construir dispositivos que permitem alterar o estado repetidas vezes. Tais circuitos, descritos em seguida, são conhecidos por *set-reset latches* ou *set-reset flip-flops*.

1 Latches SR

O nome SR vem de *set-reset*. Alguns autores usam a sigla RS em vez de SR. Na figura 1, à esquerda encontra-se o circuito de um *latch* SR que usa portas NÃO-OU, e à direita o seu comportamento (saídas Q e $\overline{Q}$) em função das entradas (S e R).

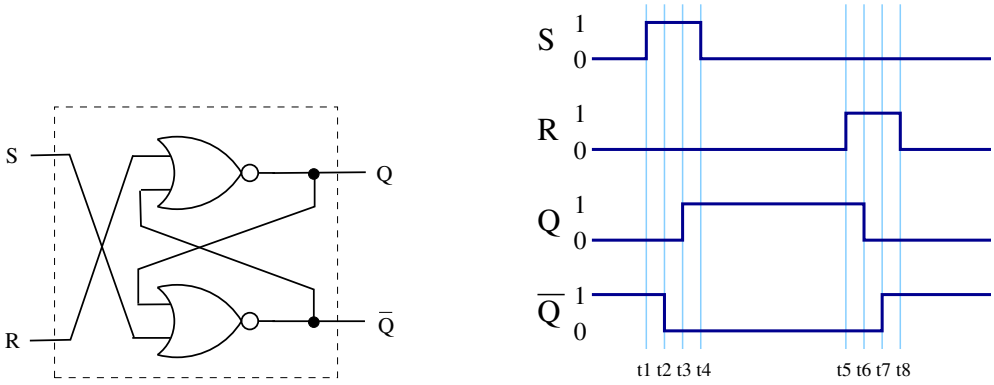
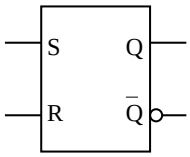


Figura 1: Operação de um *latch* SR (S=sinal set; R=sinal reset; Q=saída (estado)).

Inicialmente, ambas as entradas, S e R estão em 0, e a saída Q está também em 0 (consequentemente, a saída $\overline{Q}$ está em 1). Esse estado é consistente. No instante t_1, o sinal S vai a 1. Depois de um certo retardo, no instante t_2, $\overline{Q}$ vai a 0 e depois de outro retardo, em t_3, Q vai a 1. Quando, em t_4, S volta a 0, nenhuma mudança ocorre nos demais sinais. Em t_5, o sinal R vai a 1, e depois de um certo retardo, em t_6, Q vai a 0 e, em seguida, no instante t_7, $\overline{Q}$ vai a 1. No instante t_8, R volta a 0, porém nemhum dos outros sinais é modificado.

Para simplificar as ilustrações, um *latch* SR será representado pelo seguinte diagrama:



A relação entrada-saída desse circuito pode ser representada por uma equação. Para tanto, considere as seguintes notações:

S_i denota o valor do sinal que alimenta a entrada S num certo instante de tempo t_i.

R_i denota o valor do sinal que alimenta a entrada R num certo instante de tempo t_i.

Q_i denota a saída ou estado do *latch* num certo instante de tempo t_i.

Q_{i+1} denota a saída (ou próximo estado) do *latch* em decorrência de termos, no instante de tempo t_i, o estado Q_i e os valores S_i e R_i nas entradas S e R, respectivamente.

O comportamento do *latch* SR pode ser descrito pela seguinte tabela-verdade:

S_i	R_i	Q_i	Q_{i+1}	
0	0	0	0	Nenhuma mudança
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	?	proibido
1	1	1	?	

Resumindo, significa que quando a entrada S vai para 1, realiza-se a operação *set* (ou seja, o estado Q do *latch* vai para 1) e quando a entrada R vai a 1, realiza-se a operação *reset* (ou seja, o estado Q do *latch* vai para 0). Observe que o pulso nos sinais de entrada deve ter uma duração suficiente para que essas operações se completem e se estabilizem. Se a duração for muito curta, o comportamento poderá ser diferente.

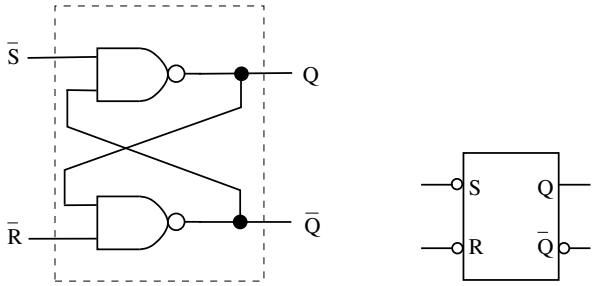
A situação $R = S = 1$ não é permitida por duas razões. Primeiro, se tivermos ambas as entradas em 1, teremos $Q = \overline{Q} = 0$, violando a condição de que uma saída é o complemento da outra. Segundo, ao supormos que $Q = \overline{Q} = 0$ é aceitável, pode ocorrer uma situação em que ambas as entradas S e R estão em 1 e passam simultaneamente para 0. Em tal situação, a saída das portas NÃO-OU passa a 1 e, em seguida, se as duas portas NÃO-OU funcionarem de forma exatamente iguais, o estado voltará para 0, o que faz com que em seguida passe para 1 e depois novamente para 0 e assim por diante. Isso levaria o estado do circuito a oscilar (ou seja, a saída não se estabilizaria). Se as duas portas não funcionarem de forma exatamente iguais, aquela que responde primeiro ditará o comportamento do circuito. Como na prática é razoável supormos que uma das portas responderá antes da outra e como numa realização física não se tem controle de qual porta responde primeiro, o comportamento será imprevisível.

Pelas razões descritas acima, consideramos que a entrada $R_i = S_i = 1$ é proibida e, portanto, na descrição do comportamento do circuito via uma expressão lógica, ela será tratada como *don't care*. Desta forma, a expressão resultante é:

$$Q_{i+1} = S_i + Q_i \, \overline{R}_i$$

que pode ser facilmente obtida desenhando-se o mapa de Karnaugh correspondente à tabela-verdade dada acima.

O *latch* SR pode também ser realizado com portas NÃO-E, da seguinte forma:



Neste caso, $\overline{S} = 0$ faz o estado ir a 1 (Q=1) e $\overline{R} = 0$ faz o estado ir a 0 (Q=0). Ou seja, este é um *latch* SR ativado por sinal baixo (e isso é indicado com a bolinha nas entradas S e R).

Cada *latch* pode ser pensado como uma unidade de memória capaz de armazenar um *bit*. Em geral, um sistema é composto por várias dessas unidades de memória e a mudança de estado dessas unidades precisa ser coordenada. Para isso, existem os *latches* controlados como o da figura 2. Note que existe um terceiro sinal de entrada C; quando $C = 0$, a saída de ambas as portas E é 0 e portanto mudanças no valor de R e S não tem efeito nenhum sobre o estado do *lathc*. Quando $C = 1$, temos o mesmo comportamento descrito na tabela acima. A literatura denomina em geral esses tipos de dispositivos de *latches* quando não há uma entrada para o sinal de controle e de *flip-flops* quando há.

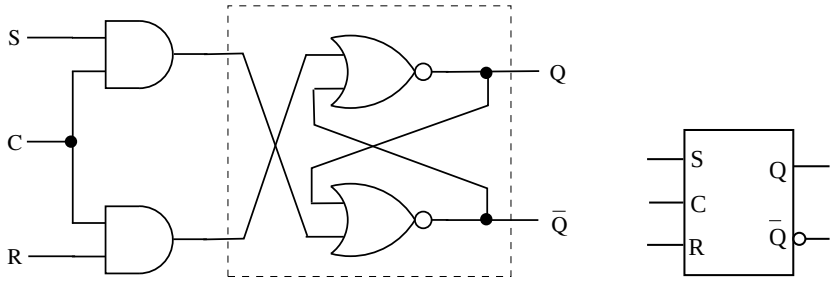


Figura 2: Um flip-flop SR (há uma entrada para o sinal de controle).

Um sinal de controle usado para coordenar a mudança de estado dos *flip-flops* é o sinal *clock*. O sinal de *clock* é um sinal periódico de frequência fixa (que determina os passos dos instantes de tempo). Note que nos *flip-flops*, antes que a entrada C torne-se ativa, é esperado que os sinais para as demais entradas estejam devidamente preparadas.

Daqui em diante, sempre que mencionarmos *flip-flop* SR, estaremos considerando a existência de entrada para o sinal de controle. Considerando a entrada C para o sinal de controle, a expressão do próximo estado de um *flip-flop* SR é dada por

$$Q_{i+1} = S_i\, C_i + Q_i\, \overline{R}_i + Q_i\, \overline{C}_i\,.$$

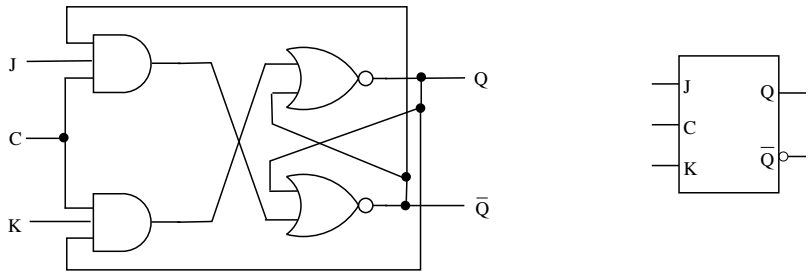
Se $C_i = 0$, temos $Q_{i+1} = Q_i$ e se $C_i = 1$ então temos $Q_{i+1} = S_i + Q_i\, \overline{R}_i$, a mesma equação do *latch* SR.

Observação: o termo *flip-flop* é usado por alguns autores para se referir tanto aos dispositivos denominados neste texto por *latches* quanto por *flip-flops*.

2 *Flip-flop* JK

Vimos que nos *flip-flops* SR, se ambas as entradas estiverem em 1 simultaneamente, eles podem apresentar comportamento imprevisível. Isso, do ponto de vista prático, é indesejável pois o projetista do circuito precisaria garantir que as duas entradas nunca ficarão ativas simultaneamente.

Os *flip-flops* JK são uma evolução do SR pois não apresentam esse problema. No JK, quando ambas as entradas ficarem em 1 simultaneamente, o estado do circuito se inverte, isto é, se a saída era 1, então passa a ser 0 e vice-versa. A figura a seguir mostra uma possível realização do *flip-flop* JK e a respectiva representação diagramática.



O comportamento do *flip-flop* JK é igual ao do *flip-flop* SR, exceto quando $J = K = 1$. Vejamos o que ocorre quando tivermos $J = K = 1$. Suponha, inicialmente, que $J = K = 0$ e que então ambas passam a 1 e em seguida C é ativada (faz-se $C = 1$): (a) se tínhamos $Q_i = 0$, então a saída da porta E de cima vai para 1 e, em consequência, teremos $Q = 1$ e $\overline{Q} = 0$; (b) se tínhamos $Q_i = 1$, teremos (similarmente) $Q = 0$ e $\overline{Q} = 1$ em seguida. Ou seja, $J = K = 1$ tem o efeito de inverter o estado do *flip-flop*.

Tabela-verdade do *flip-flop* JK:

J_i	K_i	C_i	Q_{i+1}	
×	×	0	Q_i	não muda
0	0	1	Q_i	mantém
0	1	1	0	*reset*
1	0	1	1	*set*
1	1	1	$\overline{Q}_i$	inverte

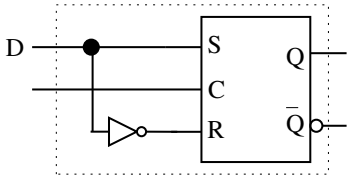
Para $C_i = 1$ a equação do próximo estado é dada por

$$Q_{i+1} = J_i\overline{Q}_i + \overline{K}_iQ_i\,.$$

Note que o sinal em C deve ser apenas um pulso (valor 1 por um pequeno instante de tempo) para que a saída das portas E não seja alterada pela segunda vez (em consequência da alteração do valor de Q). Mais adiante veremos que se o sinal que alimenta C permanecer em 1 por um longo período de tempo, o estado Q poderá oscilar algumas vezes.

3 *Flip-flops* D e T

***Flip-flop* D:** é um *flip-flop* que, quando o controle está ativo, copia a entrada D para a saída Q. Uma possível implementação é alimentar a entrada S de um *flip-flop* SR com o sinal D e R com o seu inverso, como na figura a seguir. Pode-se também usar um *flip-flop* JK em vez de um *flip-flop* SR.

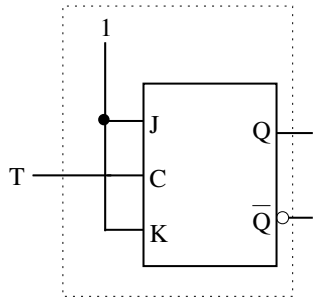


A expressão do próximo estado do *flip-flop* D é dada por

$$Q_{i+1} = D_i\, C_i + \overline{C}_i\, Q$$

ou seja, se $C_i = 0$, $Q_{i+1} = Q_i$ e, se $C_i = 1$ então $Q_{i+1} = D_i$.

***Flip-flop* T:** é um *flip-flop* que, quando o sinal de entrada T passa a 1, inverte o estado. Uma possível implementação é alimentar J e K de um *flip-flop* JK com 1 e alimentar a entrada C dele com o sinal T.



A expressão do *flip-flop* T é dada por

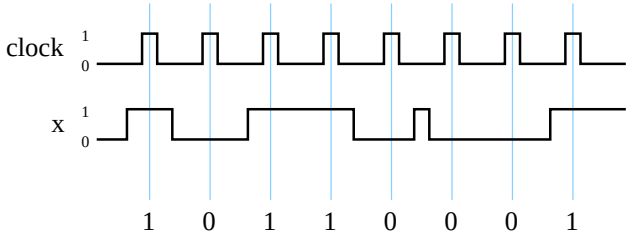
$$Q_{i+1} = \overline{Q}_i\,.$$

4 *Flip-flops* mestre-escravo

Os *flip-flops* vistos acima podem mudar de estado várias vezes enquanto o sinal que alimenta o controle C estiver alto. Em geral, o sinal que alimenta a entrada C é o sinal de *clock*. Em condições ideais, poderíamos fazer com que a duração de um pulso de um sinal de clock seja menor que o tempo necessário para que ocorram mais de uma mudança de estado nos *flip-flops*. No entanto, na prática, este tipo de controle é difícil e pode nem ser possível.

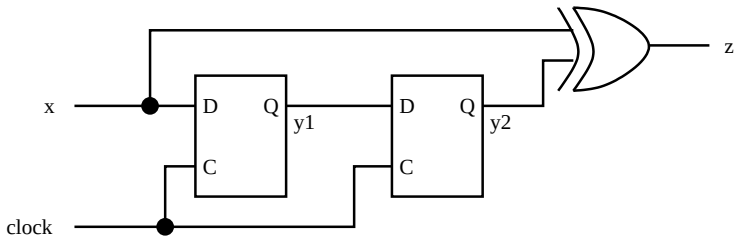
Exemplo: Para ilustrar o problema acima, considere uma sequência de *bits* x_i, $i = 1, 2, \ldots$. Deseja-se gerar uma outra sequência de *bits* z_i, $i = 1, 2, \ldots$, tal que $z_i = 1 \iff x_i \neq x_{i-2}$. Vamos supor que $x_i = 0$ para $i < 1$. Por exemplo, se a sequência x é dada por 10110001 então a sequência z deverá ser 10011101.

Na prática, sinais que alimentam um circuito são sinais contínuos no domínio do tempo. Tais sinais contínuos podem ser interpretados como uma sequência de *bits*, tomando-se o valor do sinal (baixo=0 e alto=1) nos instantes de tempo nos quais o sinal de *clock* está alto. Ou seja, para produzir uma determinada sequência de *bits*, basta sincronizarmos de forma adequada a subida e descida do sinal de entrada com o sinal de *clock*. Um exemplo é mostrado na figura a seguir. O sinal de *clock* é um sinal periódico, formado por pulsos (subida seguido de descida) de pequena duração[1]. As linhas verticais no gráfico correspondem aos instantes de tempo nos quais o sinal de *clock* está alto. Logo, a sequencia de *bits* é 10011101, que são os valores do sinal x nos instantes que o sinal de *clock* está alto. Observe que o sinal x apresenta um pulso entre os quinto e sexto pulsos do sinal de *clock*. Porém, como trata-se de um pulso num instante em que o sinal de *clock* está baixo, não deve ser encarado como um *bit* adicional na sequência x_i.

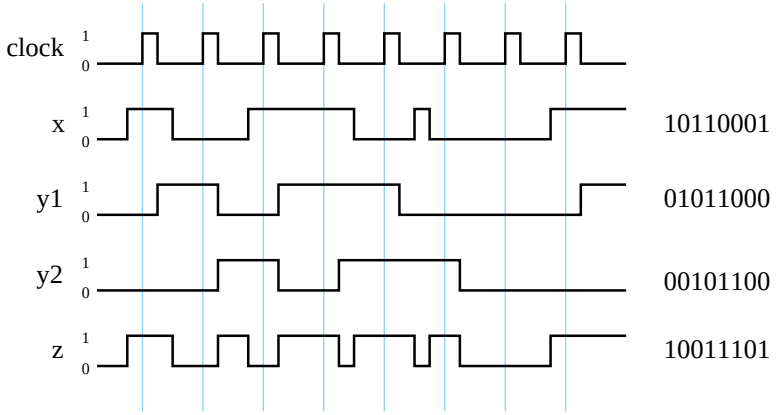


Um circuito sequencial para, dada uma sequência x, produzir a sequência z é mostrado na figura a seguir:

[1] As ilustrações apresentadas mostram os sinais "quadrados" idealizados. Na prática, eles não são "quadrados".



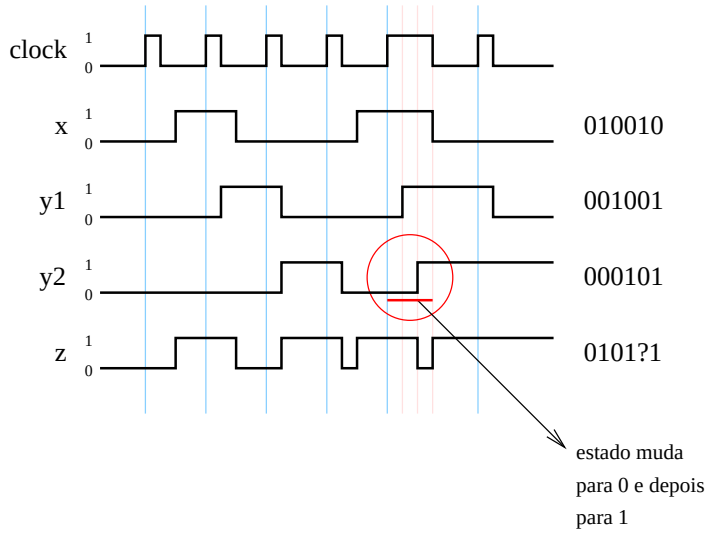
O *flip-flop* D copia a entrada atual quando o *clock* está alto para a respectiva saída Q. A simulação do comportamento dos sinais desse circuito ao longo do tempo é mostrada na figura a seguir:



Observe que o primeiro *flip-flop* funciona como um deslocador, ou seja, sua saída y_1 é exatamente igual à entrada x, exceto pelo fato de que os *bits* estão defasados em um instante. Similarmente, o segundo *flip-flop* funciona como um deslocador em relação ao sinal y_1, ou seja, y_2 é exatamente igual a x, exceto pelo fato de estar defasado em dois instantes de tempo. Logo, ao se computar o XOR entre x e y_2, obtém-se z ($z_i = 1 \iff x_i \neq x_{i-2}$). Observe que z apresenta um pulso entre o segundo e terceiro pulsos do *clock* bem como descidas entre o quarto e quinto pulsos e entre o quinto e sexto pulsos do sinal de *clock*, que não são levados em consideração na sequência z_i.

Observe também que cada *flip-flop* muda de estado apenas uma vez numa subida do sinal de *clock* e que ambos os *flip-flops*, quando mudam de estado, mudam simultaneamente. Na simulação acima, estamos supondo que o retardo do sinal de entrada do *flip-flop* para se propagar até a saída é exatamente igual à duração de um pulso do sinal de *clock*. Quanto à porta XOR, estamos supondo que não há retardo nenhum.

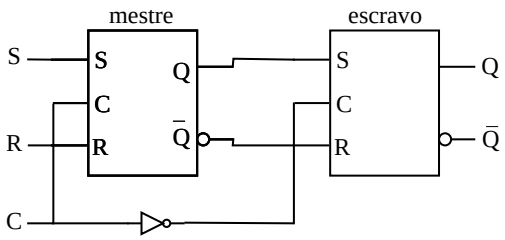
Suponha agora que, por alguma razão, um pulso de *clock* pode durar mais do que o tempo normal em algumas situações. O diagrama da figura a seguir mostra a simulação do mesmo circuito acima, para um outro sinal de entrada x. Observe que o quinto pulso do sinal de *clock* tem uma duração três vezes maior que o normal.



Na evolução do sinal y_2, a dupla mudança de estado aparece em destaque em t_5 (no quinto pulso do *clock*). A subida do *clock* em t_5 faz com que o sinal y_1 também suba logo em seguida. Essa subida de y_1 só deveria afetar o sinal y_2 na próxima subida do *clock*, em t_6. Porém, como o pulso do *clock* em t_5 é demorado, acaba afetando y_2, ou seja, y_2 também acaba subindo em t_5. Isto significa que, o sinal y_2 cujo estado anterior em t_4 era 1 passou para 0 em t_5 e ainda em t_5 passou novamente para 1, ou seja, mudou de estado duas vezes em t_5. Isso compromete toda a evolução subsequente do sinal de saída.

Para contornar esse tipo de problema, existem os chamados *edge-triggered flip-flops* que são aqueles que mudam de estado somente na transição 1 para 0 (descida) ou na transição 0 para 1 (subida) do sinal de controle. Um exemplo desse tipo de *flip-flop* são os ***flip-flops* mestre-escravo**.

A figura a seguir mostra o esquema de um ***flip-flop* SR mestre-escravo**.



Quando o *clock* está baixo, mudanças nas entradas S e R não têm efeito no *flip-flop* mestre. O *flip-flop* escravo está habilitado para mudanças (sua entrada C é 1), mas como nenhuma mudança ocorre nas suas entradas S e R (que vem das saídas do *flip-flop* mestre) seu estado também não muda.

Quando o *clock* sobe, o mestre pode mudar de estado de acordo com as entradas S e R enquanto o escravo fica desabilitado (deve-se apenas garantir que o escravo fique

desabilitado antes que ocorra qualquer mudança na saída do mestre). Quando o *clock* desce, o mestre fica desabilitado (e portanto "congela" a saída dele) e a mudança de estado no escravo fica habilitada (assim, a saída do mestre é copiada para a saída do escravo). Desde o momento em que o *clock* subiu, a saída do mestre pode ter oscilado algumas vezes, porém a saída do *flip-flop* como um todo (que é a saída do escravo) só muda quando o *clock* baixa. Este é, portanto, um exemplo de *flip-flop* que muda de estado na descida do sinal de *clock*.

Similarmente, pode-se construir *flip-flops* que mudam de estado apenas na subida do sinal de controle. Pode-se também construir tais *flip-flops* usando-se como base o *flip-flop* JK ou ainda outros tipos de circuitos com realimentação.

A figura a seguir mostra a representação diagramática de *flip-flops* SR mestre-escravo com mudança de estado respectivamente na subida e na descida do sinal de controle. O triângulo na entrada C indica que o *flip-flop* é *edge-triggered*; a bolinha indica que a mudança de estado ocorre na descida do sinal de controle, enquanto a ausência de bolinha indica que a mudança de estado ocorre na subida do sinal de controle.