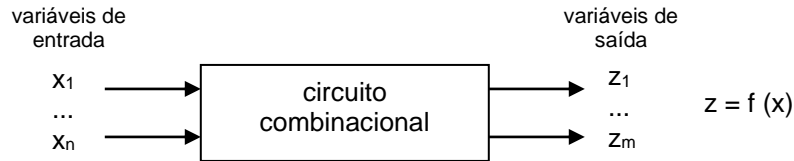
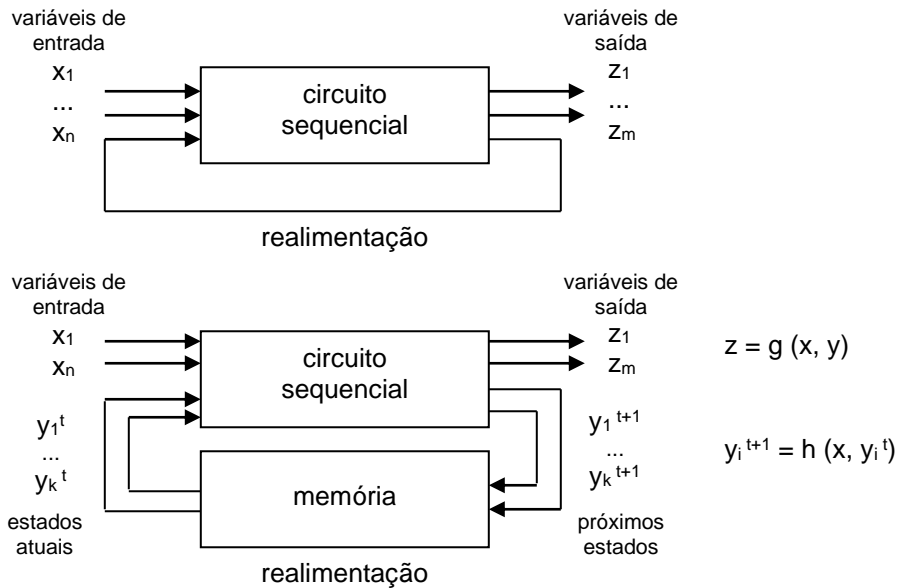


## Circuitos sequenciais

Um *circuito combinacional* é aquele em que a(s) saída(s) depende(m) de uma combinação das entradas.



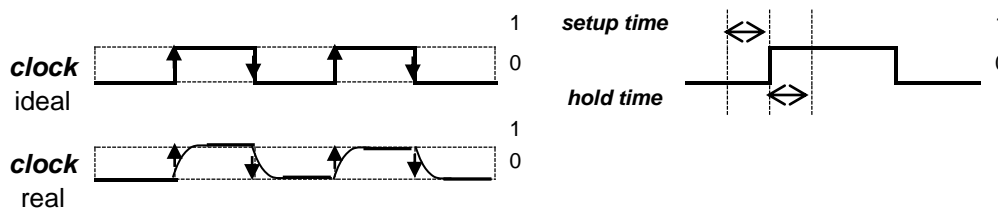
Um *circuito sequencial*, além de uma combinação das entradas, depende de uma combinação de outras variáveis que definem o estado em que o sistema se encontrava. Isto significa que um sistema deverá ter *memória*; para passar ao próximo estado, precisará guardar informações sobre o estado atual.



Basicamente, há dois tipos de circuitos sequenciais:

- assíncronos - em que os estados podem mudar a qualquer instante
- síncronos - em que os estados mudam em instantes bem determinados

As mudanças de estados que ocorrerão em instantes determinados serão orientadas por um sinal de temporização (**clock**). Se ocorrerem as transições ocorrerem durante uma variação de 0 para 1 (↑ - borda de subida), o sistema será dito de *nível alto*; caso contrário, durante uma variação de 1 para 0 (↓ - borda de descida), o sistema será dito de *nível baixo*. As especificações de tempo para circuitos sequenciais também incluirão o tempo para a transição se estabilizar (**setup time**) e o tempo, após a transição, em que o sinal deve se manter constante (**hold time**).



### Máquinas de estados finitos (**Finite State Machines**)

Uma *máquina de estados finitos*, ou simplesmente *autômato finito*, é um modelo de comportamento composto de estados, transições e ações. Um estado armazena uma informação sobre a história de um sistema (reflete como as mudanças nas entradas trouxeram o sistema até o estado atual). Uma transição indica uma mudança de estado e é descrita por uma condição que a permite. Uma ação é a descrição de uma atividade executada em certo instante.

Máquinas de estados finitos podem ser usadas para descrever circuitos sequenciais pois suas saídas e seus novos estados são funções de suas entradas e de seus estados atuais.

Diagrama de estados

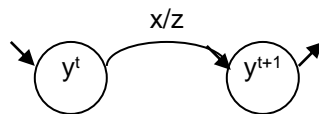
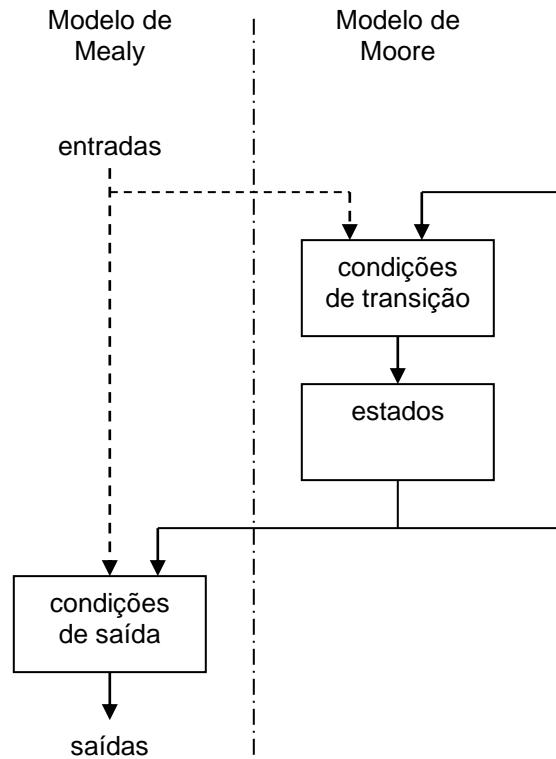


Tabela de estados

		entrada	
estado\ atual	y	x	

Os modelos de Mealy e Moore são comumente usados para descrever máquinas de estados finitos. Se a saída for função apenas do estado, o modelo de Moore é melhor empregado. Caso a saída seja função do estado da máquina e de suas entradas, então o modelo de Mealy é melhor empregado. Na prática, esses dois modelos são combinados para oferecer uma descrição do funcionamento de uma máquina de estados finitos.



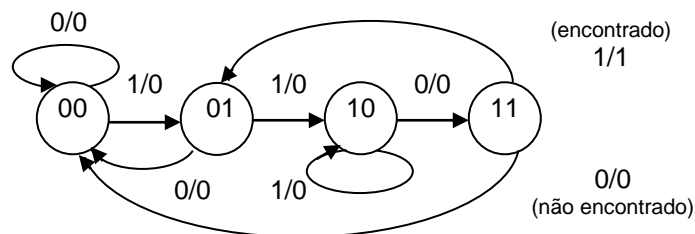
Exemplo:

Considerar um circuito capaz de identificar a sequência binária (abcd=1101).

Modelo de Mealy:

estado atual	\ código y	entradas nome	Tabela de Estados x	
			x	
			x=0	x=1
	0 0	início	início / s=0	id1 / s=0
	0 1	id1	início / s=0	id11 / s=0
	1 0	id11	id110 / s=0	id11 / s=0
	1 1	id110	início / s=0	id1 / s=1

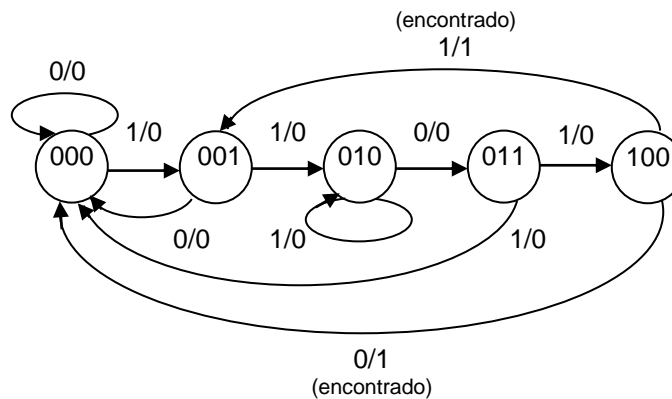
Diagrama de estados (Mealy)



Modelo de Moore:

estado atual	\ código y	entradas nome	Tabela de Estados x	
			x	
			x=0	x=1
	0 0 0	início	início / s=0	id1 / s=0
	0 0 1	id1	início / s=0	id11 / s=0
	0 1 0	id11	id110 / s=0	id11 / s=0
	0 1 1	id110	início / s=0	id1101 / s=0
	1 0 0	id1101	início / s=0	id1 / s=1

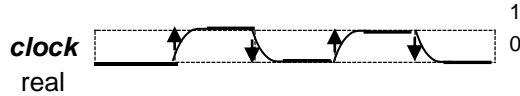
Diagrama de estados (Moore)



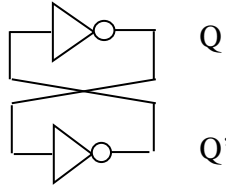
## Latches

Um **latch** é uma estrutura lógica capaz de armazenar um **bit**. Constitui-se de um circuito que muda de estado apenas devido às variações das entradas.

Os **latches** são geralmente empregados para se construir chaveadores sem ressaltos (**debounced**). Os ressaltos são variações de natureza oscilatória que podem ocorrer durante uma transição, e poderão ser interpretados erroneamente por um circuito.



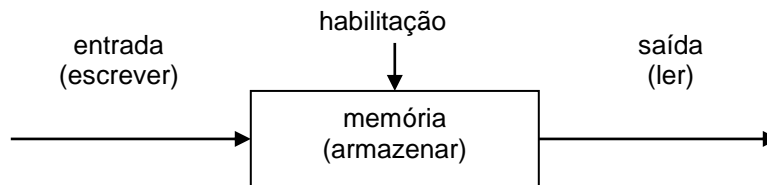
O esquema abaixo ilustra a composição de um **latch** implementado com dois inversores.



Se for imposto que a saída ( Q ) tenha valor igual a ( 1 ), o seu complemento ( Q' ) terá valor igual a ( 0 ). Caso contrário, se ( Q ) for igual a ( 0 ), então ( Q' ) será igual a ( 1 ). Dessa forma, uma unidade mínima de informação ( **bit** ) poderá ser guardada.

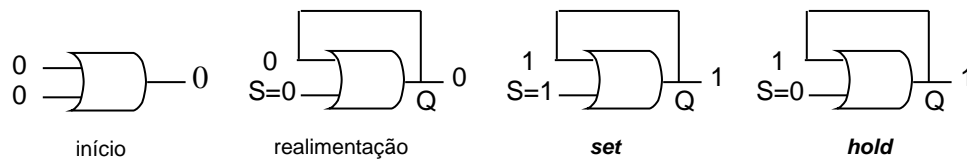
Um **latch** constitui um elemento básico de memória e opera enquanto determinado nível de sinal ( 0 ou 1 ) for mantido.

De modo geral, um elemento de memória pode ser descrito como um circuito capaz de receber (escrita), armazenar um valor binário e fornecer cópia (leitura) do valor armazenado.

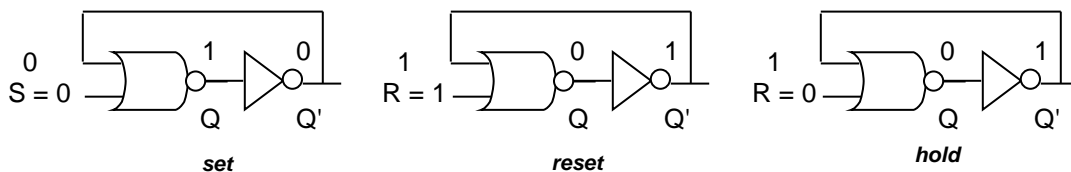


Construção de um **latch** do tipo **set-reset**:

- Com porta OR:



- Substituindo por porta NOR:



- No mapa de Karnaugh:

Q \ S R	<b>set</b>			
	0 0	0 1	{ 1 1	1 0 }
0	0	0	X	1
1	1	0	X	1
	0 0	{ 0 1	1 1 }	1 0
	<b>reset</b>			

$$\begin{aligned}
 S = 0 : (S' \cdot Q')' &= (1 \cdot Q')' = Q \\
 R = 0 : (Q' \cdot Q)' &= (1 \cdot Q)' = Q' \\
 S=0, R=1 : Q &= (S' \cdot Q')' = (0' \cdot 1)' = 0 \\
 S=1, R=0 : Q' &= (R' \cdot Q)' = (0' \cdot 1)' = 0 \\
 S = 1 : (S' \cdot Q')' &= (0 \cdot Q')' = 1 \\
 R = 1 : (Q' \cdot Q)' &= (0 \cdot Q)' = 1
 \end{aligned}$$

- Na tabela de estados:

S	R	Q <sup>t</sup>	Q <sup>t+1</sup>	
0	0	0	0	<b>hold</b>
0	0	1	1	
0	1	0	0	<b>reset</b>
0	1	1	0	
1	0	0	1	<b>set</b>
1	0	1	1	
1	1	0	X	<b>unused</b>
1	1	1	X	

### Latch SR

Um **latch** SR (**Set-Reset**) pode ser construído com portas NOR.

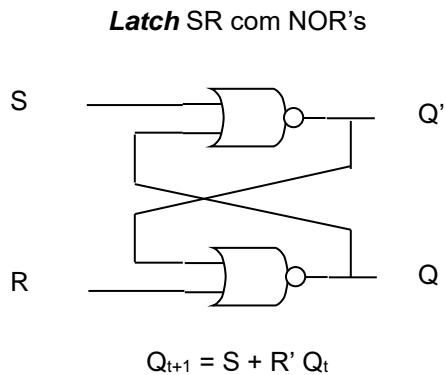


Tabela característica				
S	R	Q <sub>t+1</sub>	Q' <sub>t+1</sub>	Obs.:
0	0	Q <sub>t</sub>	Q' <sub>t</sub>	<b>hold</b>
0	1	0	1	<b>reset</b>
1	0	1	0	<b>set</b>
1	1	?	?	<b>unused (X)</b>

Equação característica

Um **latch** SR (**Set-Reset**) também pode ser construído com portas NAND.

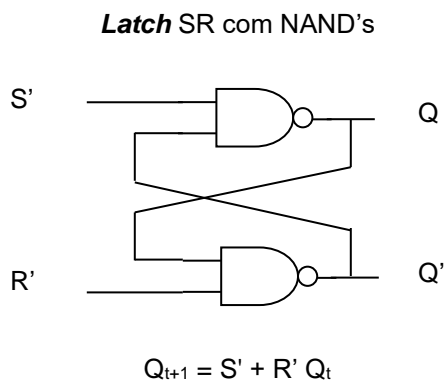
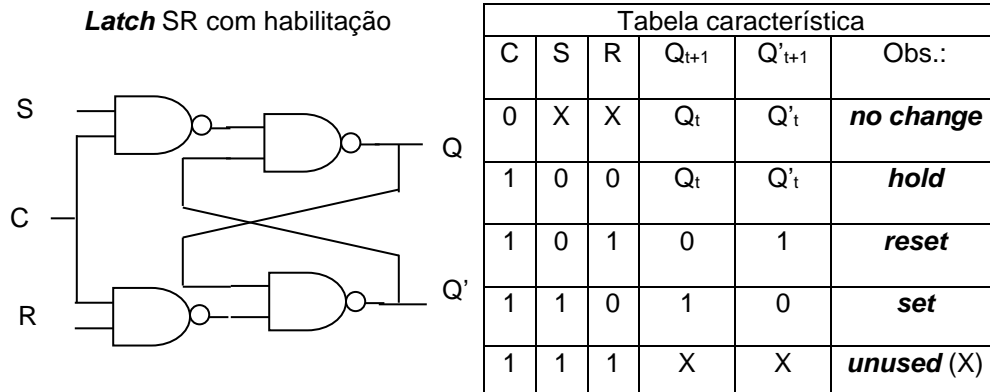


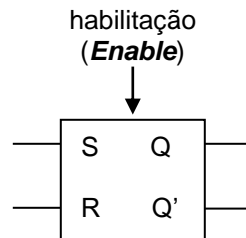
Tabela característica				
S'	R'	Q <sub>t+1</sub>	Q' <sub>t+1</sub>	Obs.:
0	0	?	?	<b>unused (X)</b>
0	1	1	0	<b>set</b>
1	0	0	1	<b>reset</b>
1	1	Q <sub>t</sub>	Q' <sub>t</sub>	<b>hold</b>

Equação característica

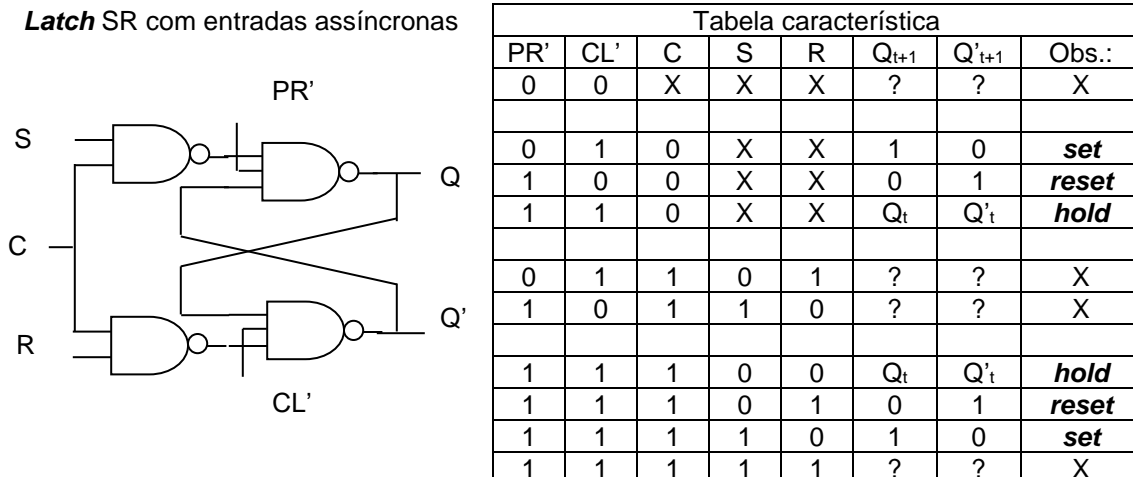
Um problema que pode ser observado pelas tabelas é a indicação de estado indefinido (não utilizado ou X). Uma tentativa de solução é o acréscimo de uma entrada de controle (habilitação), cujo objetivo é tentar evitar que valores das entradas ( S e R ) possam levar ao estado não definido ( X ).



A representação genérica do **latch** SR com habilitação pode ser a mostrada abaixo.



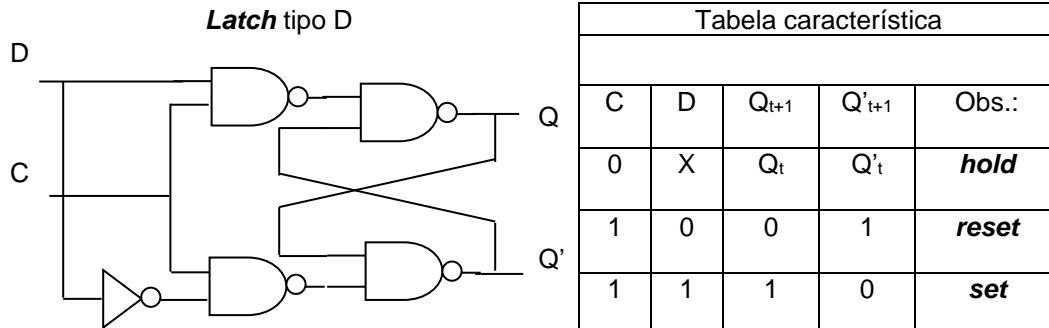
O **latch** SR também pode ser construído com mais duas entradas assíncronas: uma para estabelecer certo valor inicial ( PReset ), e outra para limpar ( CLear ) o conteúdo armazenado.



A configuração com habilitação tem como vantagem não necessitar de uma configuração especial dos sinais ( S ) e ( R ) para manter o estado anterior. Contudo, ainda há desvantagem: a existência de um estado não utilizado. O circuito a seguir procura resolver esse problema vinculando as entradas.

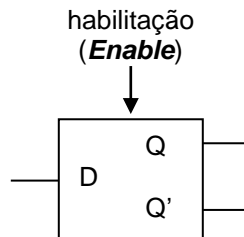
**Latch** tipo D

Um **latch** tipo D mantém a vantagem de manter o estado atual ( **hold** ), mesmo que desabilitado, além disso, é capaz de impedir a ocorrência do estado inválido por vincular as entradas ( S e R ) em uma única ( D ) e sua inversão.



Há, entretanto, uma desvantagem: se o sinal de controle for mantido em ( 1 ) e o sinal de entrada flutuar, as saídas também o farão. O **latch** tipo D não oferece estabilidade.

A representação genérica do **latch** tipo D pode ser a mostrada abaixo.

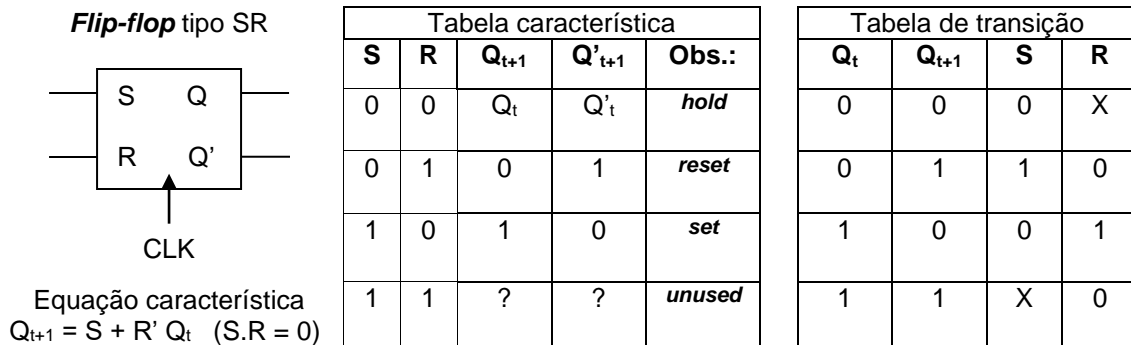


## Flip-flops

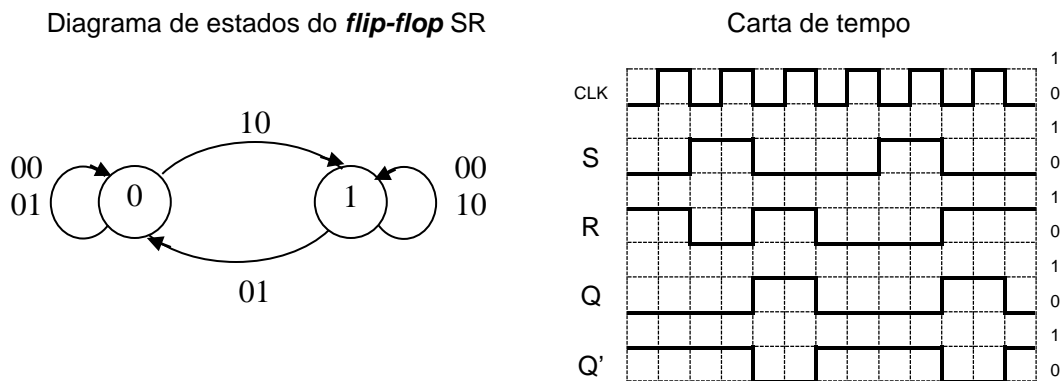
**Flip-flop** é um elemento de memória cujas atualizações de estado ocorrem somente durante as transições (positiva/subida ou negativa/descida) de um sinal de temporização (**clock**). Isso permite que os sinais possam ter pequenas variações arbitrárias, sem que isto possa afetar seus estados. Dessa forma, é possível ditar, com maior precisão, o momento em que os dados poderão ser armazenados no dispositivo. A detecção de variação de borda do pulso é normalmente associada ao sinal de habilitação.

### Flip-flop SR

A representação genérica do **flip-flop** SR com **clock** (CLK) pode ser a mostrada abaixo.



O diagrama de estados do **flip-flop** SR está mostrado abaixo.

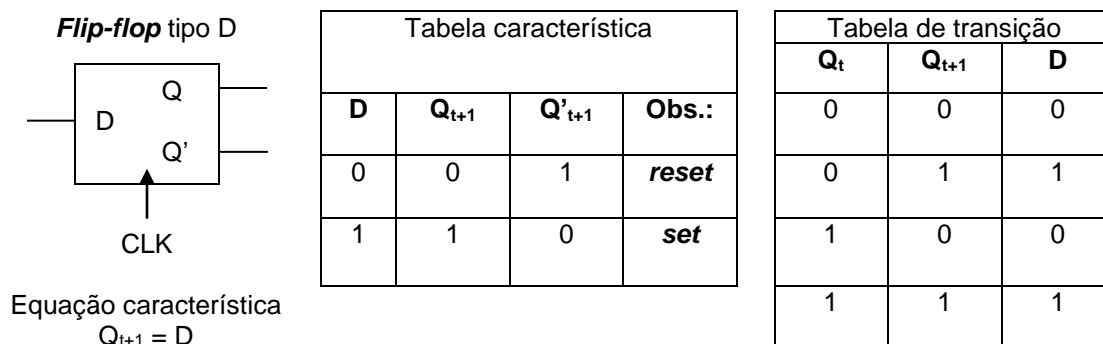


Obs.: Carta de tempo obtida com **flip-flop** sensível à borda de descida, sem atraso (*lag*).



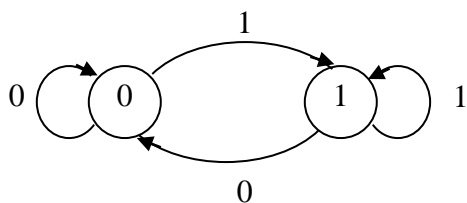
**Flip-flop D**

A representação genérica do **flip-flop** tipo D com **clock** pode ser a mostrada abaixo.

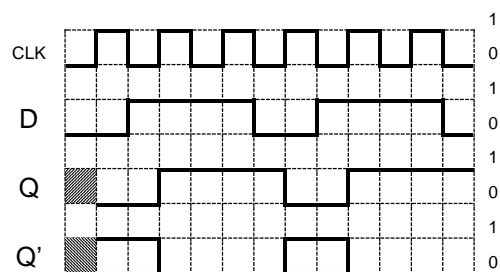


O diagrama de estados do **flip-flop** tipo D está mostrado abaixo.

Diagrama de estados do **flip-flop** tipo D



Carta de tempo

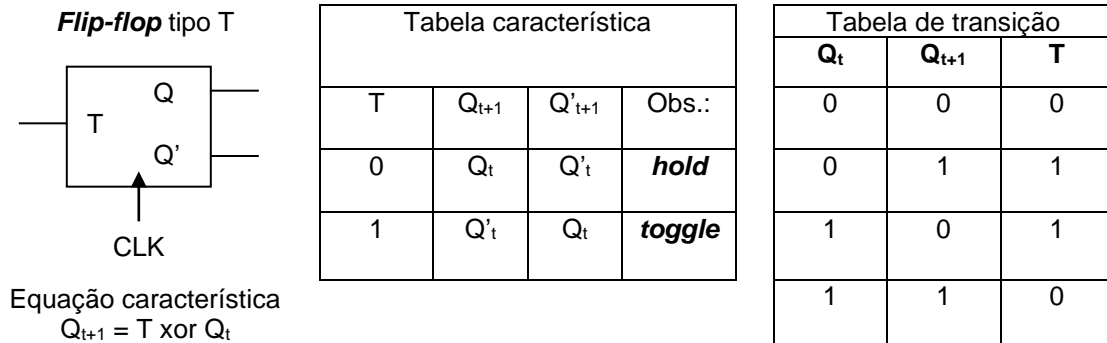


Obs.: Carta de tempo obtida com **flip-flop** sensível à borda de subida, sem atraso (*lag*).

**Flip-flop T (toggle)**

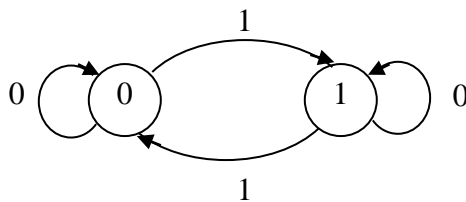
O **flip-flop** tipo T é um circuito com apenas uma entrada. Toda vez que a entrada for igual a 1, ou seja, quando houver variação de **clock** de 0 para 1, as saídas serão invertidas.

A representação genérica do **flip-flop** tipo T com **clock** pode ser a mostrada abaixo.

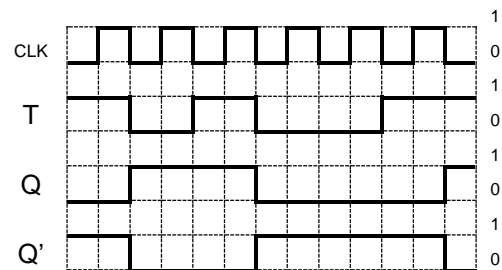


O diagrama de estados do **flip-flop** tipo T está mostrado abaixo.

Diagrama de estados do **flip-flop** tipo T



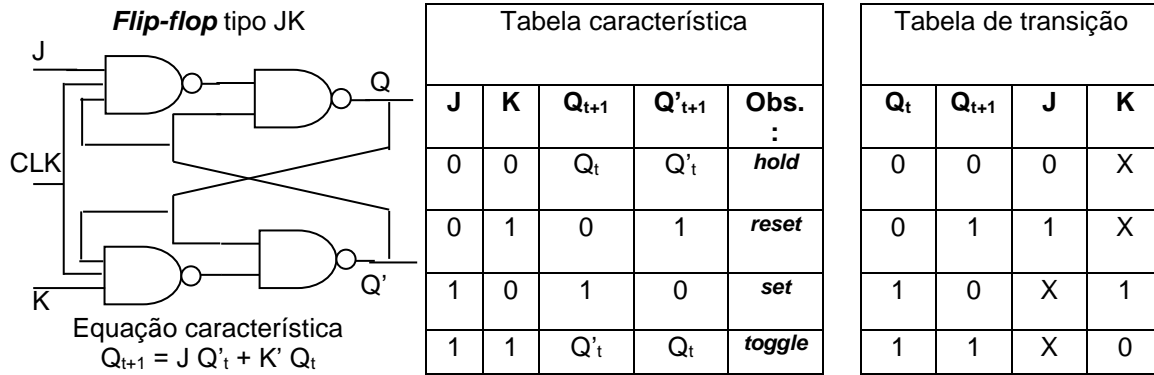
Carta de tempo



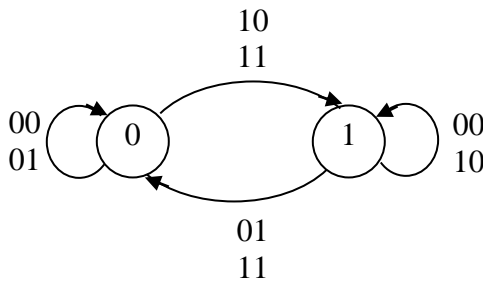
Obs.: Carta de tempo obtida com **flip-flop** sensível à borda de descida, sem atraso (**lag**).

**Flip-flop JK**

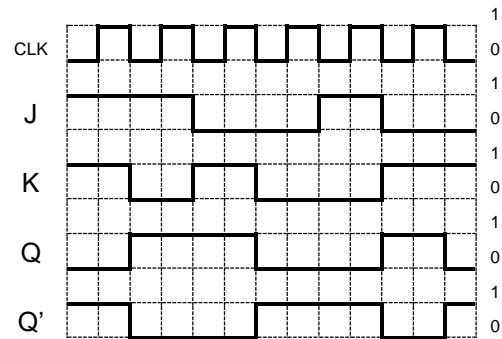
O **flip-flop** tipo JK é uma modificação do flip-flop SR que é capaz de inverter (**toggle**) as entradas quando ambas forem iguais a 1.



O diagrama de estados do **flip-flop** tipo JK está mostrado abaixo.

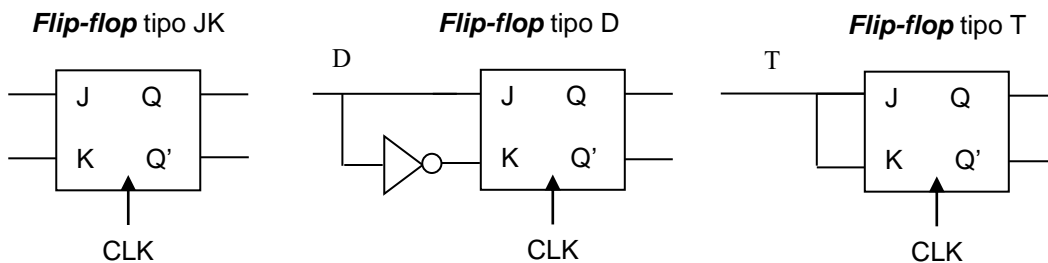
Diagrama de estados do **flip-flop** JK

Carta de tempo



Obs.: Carta de tempo obtida com **flip-flop** sensível à borda de descida, sem atraso (**lag**).

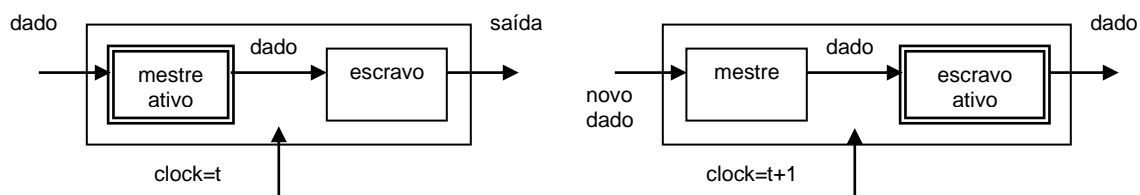
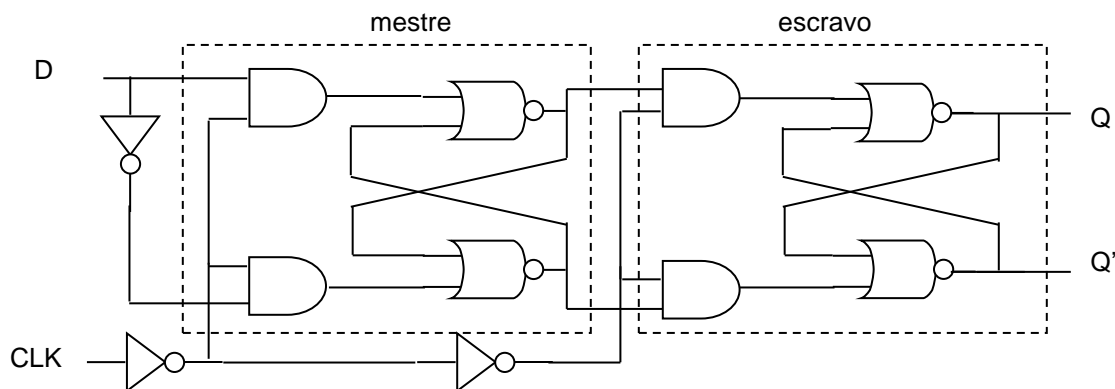
O **flip-flop** tipo JK é considerado modelo universal e usado para construir os outros tipos.



O **flip-flop** tipo JK também pode apresentar um comportamento indesejado quando entra no estado de **toggle**: as saídas Q e Q' podem entrar em oscilação até que as entradas voltem a zero. Isso pode ser resolvido através do uso de arranjos do tipo mestre-escravo (a seguir).

**Flip-flop** mestre-escravo

Em um **flip-flop** mestre-escravo, o primeiro bloco (mestre) é utilizado para receber uma entrada de dados e armazená-lo. Em um instante posterior, o dado será transferido ao escravo. Ambas as ações são sincronizadas pelo sinal de **clock**.

**Flip-flop** mestre-escravo tipo D**Flip-flop** mestre-escravo tipo JK