Lógica Digital

Aula-03: Circuitos Combinacionais Sequenciais

Eliseu César Miguel

Departamento de Ciência da Computação
Universidade Federal de Alfenas

August 10, 2021

Organização da Aula

Organização da Aula

Introdução

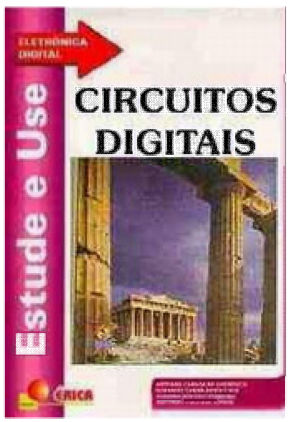
Considerações Preliminares

Este material não pretende ser completo quanto à amplitude do assunto. Aqui pretende-se apenas organizar os pontos relevantes para as aplicações dos conceitos da Lógica de Boole na disciplina de Lógica Digital, gerando um guia de estudos. Destarte, sempre consulte livros e apostilas para alcançar bons resultados em seus estudos.

Também, este material não é, em sua totalidade, de minha autoria. Ao contrário, ele contempla conteúdos de sítios de Internet e conteúdos de livros. Para tanto, cito bibliografias de textos aqui encorporados.

Boa leitura!

Bibliografia básica



Circuitos

Circuitos Combinacionais

Circuito combinacional organiza interligações entre portas lógicas para executar eletronicamente uma expressão booleana.

Circuitos

Circuitos Combinacionais

Circuito combinacional organiza interligações entre portas lógicas para executar eletronicamente uma expressão booleana.

Circuitos Combinacionais Dedicados

São cirtutos combinacionais em que as saídas dependem apenas das entradas.

Circuitos

Circuitos Combinacionais

Circuito combinacional organiza interligações entre portas lógicas para executar eletronicamente uma expressão booleana.

Circuitos Combinacionais Dedicados

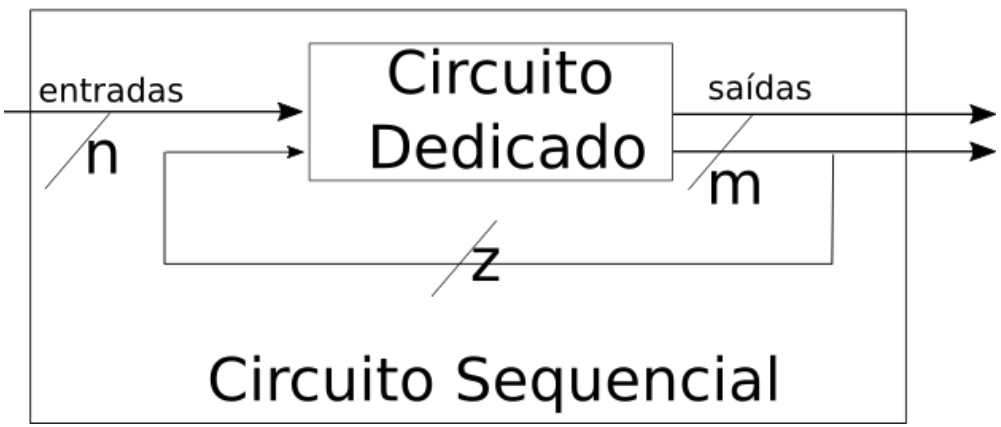
São cirtutos combinacionais em que as saídas dependem apenas das entradas.

Circuitos Combinacionais Sequenciais

São cirtutos combinacionais em que as entradas são realimentadas por algumas (ou todas) as saídas. Normalmente, a parte interna de um circuito sequencial é um circuito dedicado.

Circuitos


entradas
n
Circuito
Dedicado
saídas
m


entradas
n
Circuito
Dedicado
saídas
m
z
Circuito Sequencial

Conceito: Atraso

Os circuitos digitais são suceptíveis aos atrasos. Estes atrasos, que podem sinalizar problemas são, na verdade, a solução utilizada para fazer toda a estrutura lógica implementada funcionar corretamente. *A arquitetura de computadores não seria o que é hoje se não houvesse atraso*

Exemplo:

A saída de uma porta inversora só é atualizada após o tempo de atraso (Δt), como mostra a figura 5.3:

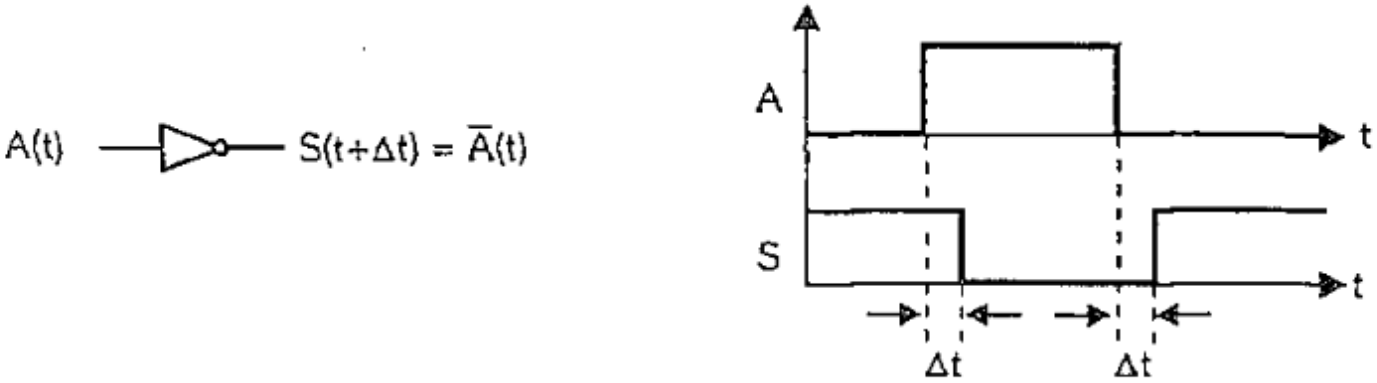


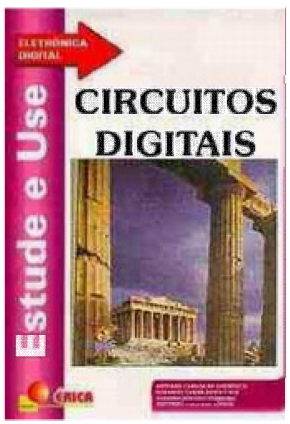
Figura 5.3 - Propagação de um Sinal numa Porta Inversora

Informação Importante

A seguir, estudaremos duas estruturas de *circuitos combinacionais sequenciais*:

- *Latch*
- *Flip-Flop*

Chamamos a atenção para o fato de que a bibliografia básica *Circuitos Digitais: Estude e Use*, Ed. 6ª da Editora Érica não faz distinção entre as duas estruturas. Destarte, caso decida por estudar a partir desta bibliografia, tenha senso crítico em associar os conceitos apresentados nesta aula ao contúdo do livro.

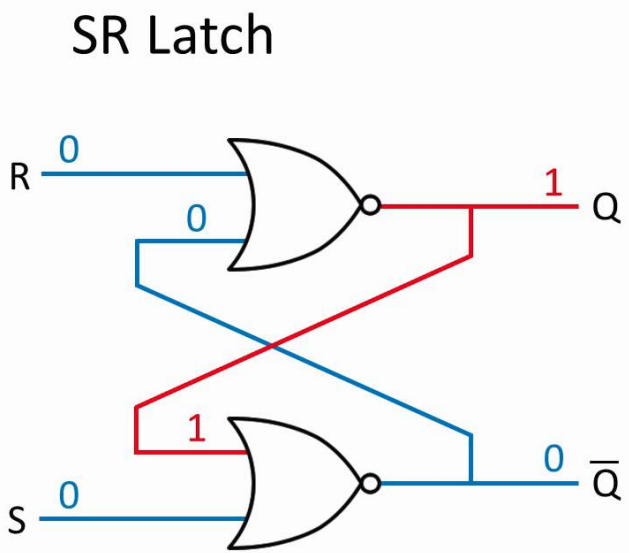


Latch SR assíncrono

Um Latch SR assincrono é um circuito combinacional sequencial que é capaz de armazenar um *bit*. O valor armazenado é configurado pelas chaves S e R.

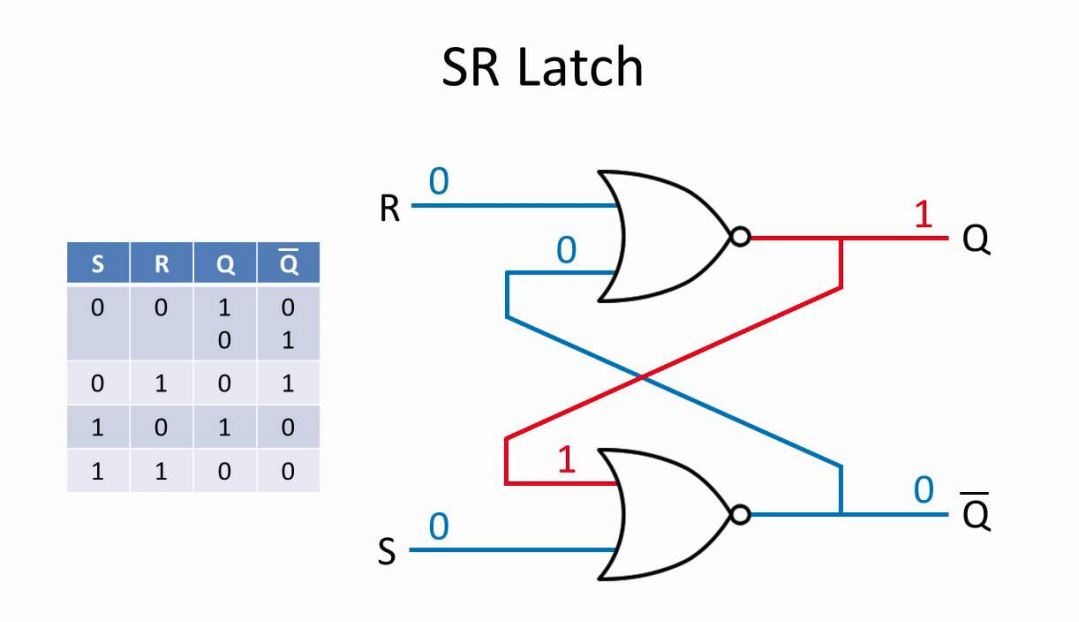
Esse Latch é assíncrono uma vez que sua atualização depende apenas do atraso de suas portas lógicas.

S	R	Q	$\overline{Q}$
0	0	1 0	0 1
0	1	0	1
1	0	1	0
1	1	0	0



Latch SR assíncrono: Exercício

Faça no *digital works* um Latch SR assincrono que tenha a mesma funcionalide do Latch da figura abaixo, mas que seja implementado com portas lógicas NAND, ao contrário de NOR.

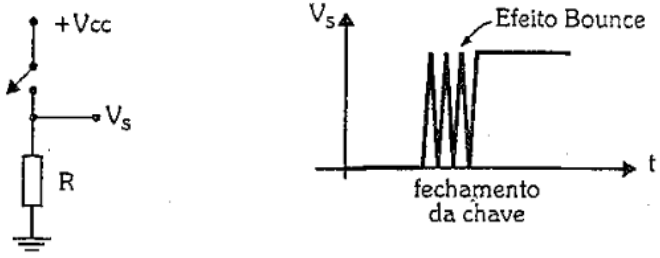


S	R	Q	$\overline{Q}$
0	0	1 0	0 1
0	1	0	1
1	0	1	0
1	1	0	0

Latch SR assíncrono: Exemplo de uso

Exemplo de Aplicação - Eliminador de Ruído (Debouncing)

Muitas vezes, o acionamento ou o controle de sistemas digitais é feito através de dispositivos mecânicos que, devido às suas características físicas de construção, apresentam vibrações ao serem acionados, gerando um ruído denominado efeito bounce, que pode ser prejudicial ao desempenho do sistema, como mostra a figura a seguir:



Por isso, muitos sistemas digitais precisam de **circuitos eliminadores de ruídos (debouncing)**, como o mostrado na figura 5.7.

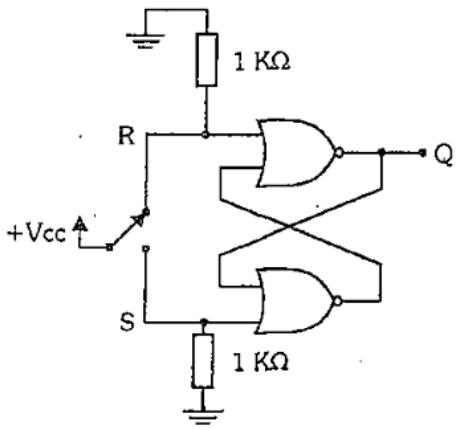
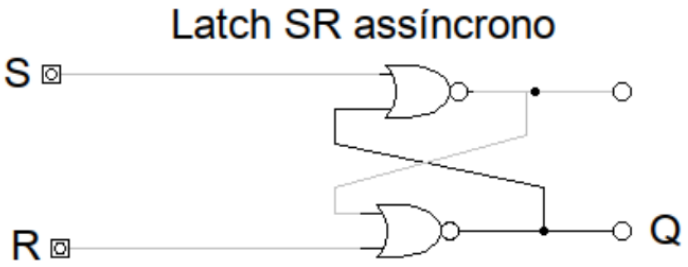


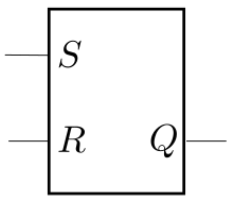
Figura 5.7 - Circuito Eliminador de Ruído (Debouncing)

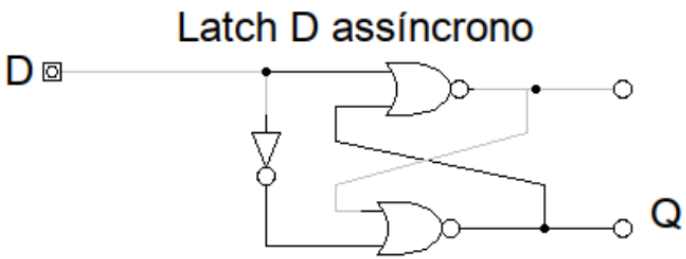
Latch SR e Latch D assíncronos

O latch SR fica estável quando $S \equiv 0$ e $R \equiv 0$, mas nada impede de ocorrer um erro lógico ($S \equiv 1$ e $R \equiv 1$). Neste caso, o latch D não permite o erro lógico.

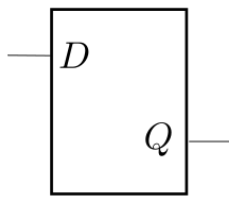


S	R	Q
0	0	Q_a
1	0	1
0	1	0
1	1	Erro



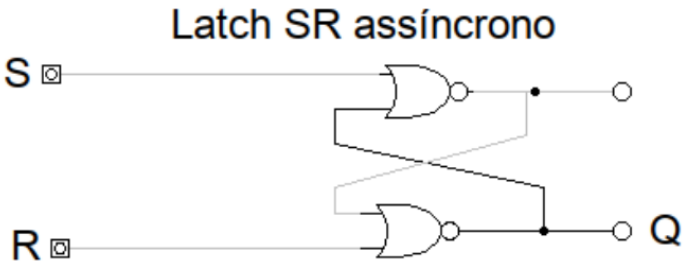


D	Q
0	0
1	1

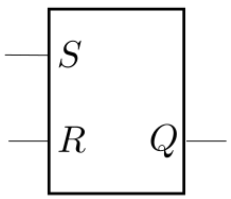


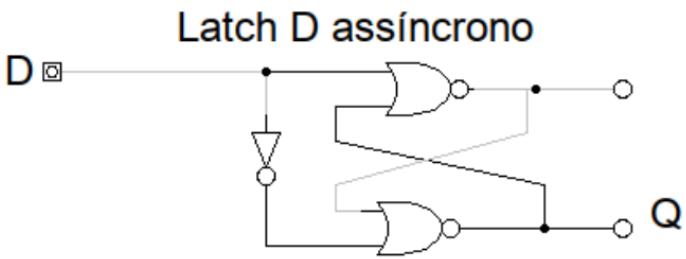
Latch SR e Latch D assíncronos

O latch SR fica estável quando $S \equiv 0$ e $R \equiv 0$, mas nada impede de ocorrer um erro lógico ($S \equiv 1$ e $R \equiv 1$). Neste caso, o latch D não permite o erro lógico.

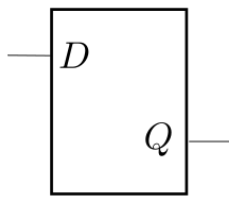


S	R	Q
0	0	Q_a
1	0	1
0	1	0
1	1	Erro





D	Q
0	0
1	1

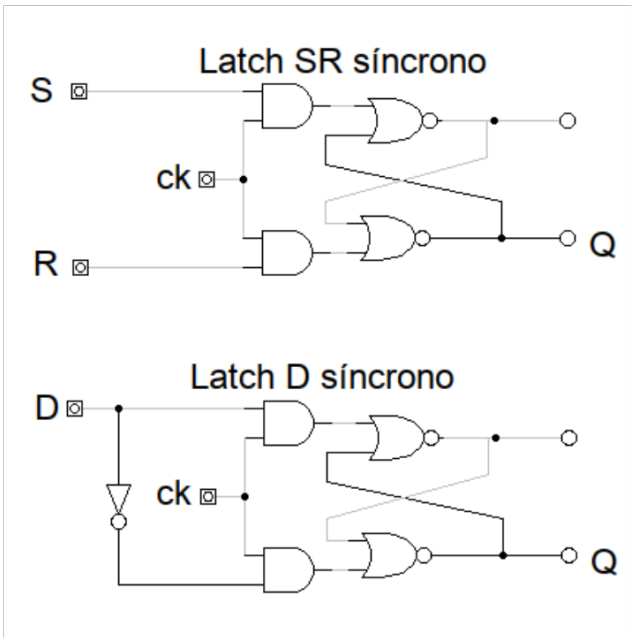


Como evitar que as variações do latch D alterem o valor armazenado?

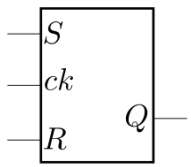
Unifal MG
Universidade Federal de Alfenas

Latch SR e Latch D síncronos

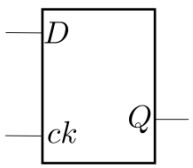
Agora, os latchs SR e D estão sincronizados à chave *ck*. Isso é bom, visto que podemos decidir quando um valor deverá ser armazenado. Esse princípio é fundamental às memórias, já que os barramentos sempre estão conectados.



ck	S	R	Q
0	x	x	Q_a
1	0	0	Q_a
	1	0	1
	0	1	0
	1	1	Erro

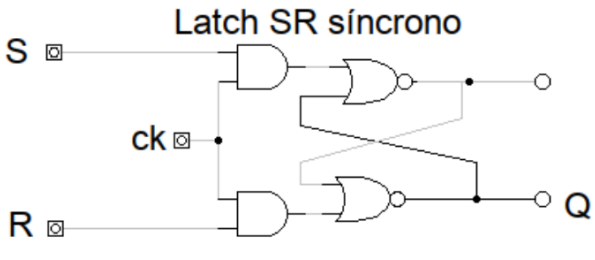


ck	D	Q
0	x	Q_a
1	1	1
	0	0

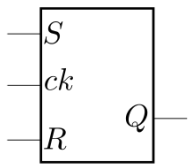


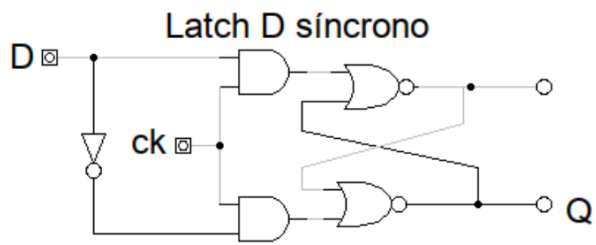
Latch SR e Latch D síncronos

Agora, os latchs SR e D estão sincronizados à chave *ck*. Isso é bom, visto que podemos decidir quando um valor deverá ser armazenado. Esse princípio é fundamental às memórias, já que os barramentos sempre estão conectados.

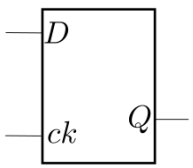


ck	S	R	Q
0	x	x	Q_a
1	0	0	Q_a
	1	0	1
	0	1	0
	1	1	Erro





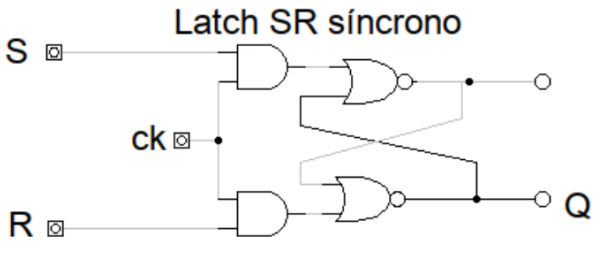
ck	D	Q
0	x	Q_a
1	1	1
	0	0



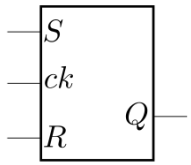
Os latchs estão sensíveis à escrita enquanto $ck \equiv 1$, e desabilitados enquanto $ck \equiv 0$.

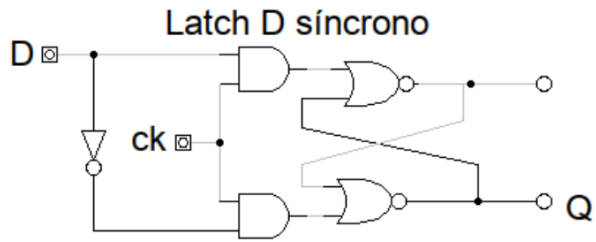
Latch SR e Latch D síncronos

Agora, os latchs SR e D estão sincronizados à chave *ck*. Isso é bom, visto que podemos decidir quando um valor deverá ser armazenado. Esse princípio é fundamental às memórias, já que os barramentos sempre estão conectados.

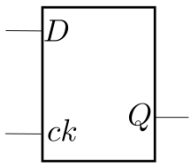


ck	S	R	Q
0	x	x	Q_a
1	0	0	Q_a
	1	0	1
	0	1	0
	1	1	Erro





ck	D	Q
0	x	Q_a
1	1	1
	0	0

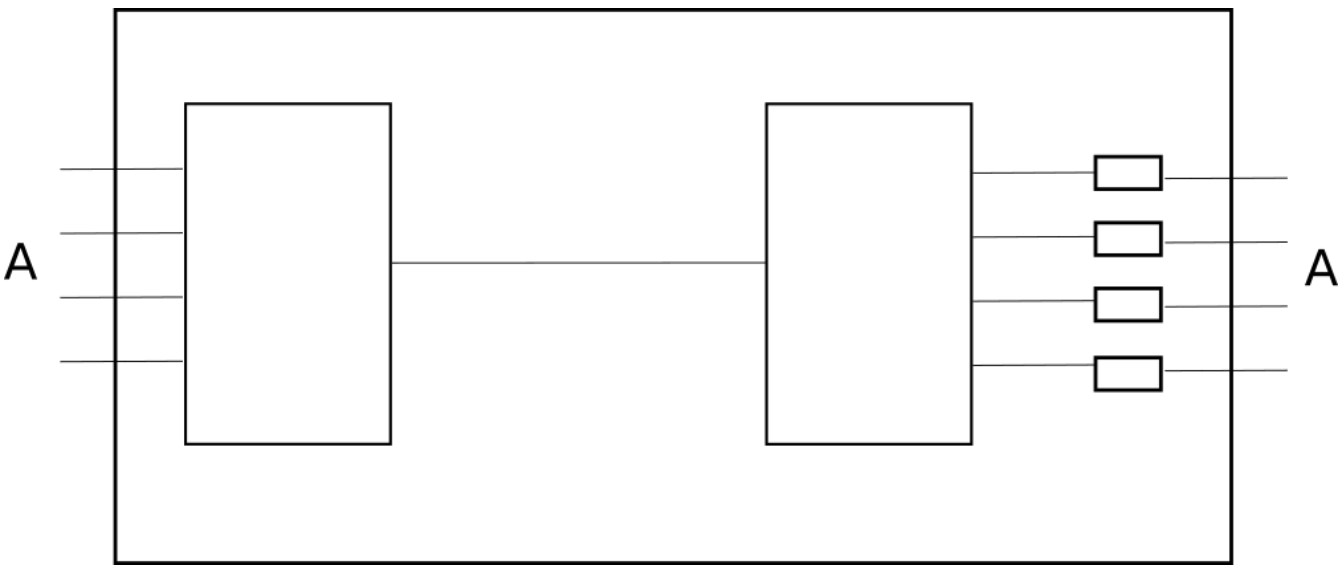


Os latchs estão sensíveis à escrita enquanto $ck \equiv 1$, e desabilitados enquanto $ck \equiv 0$.
Mas, nós sabemos mudar esse comportamento!

Latch: Exercício

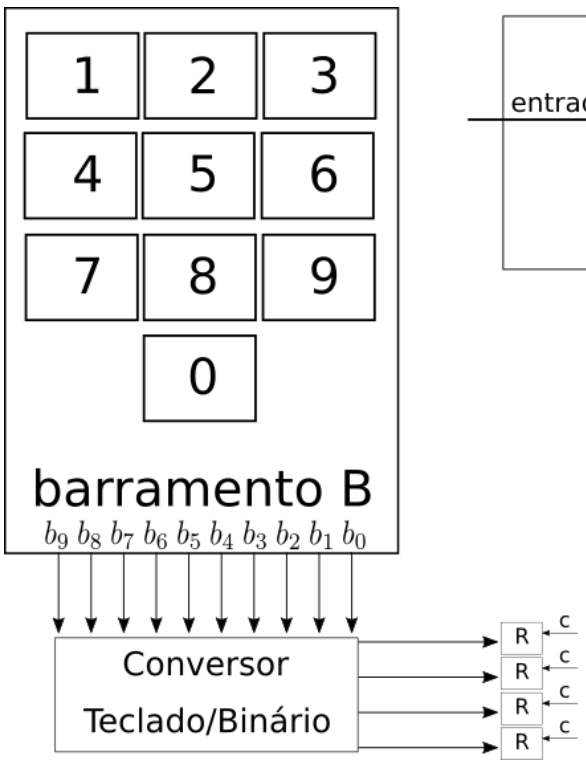
Usando o *digital works*, faça um circuito para receber uma informação A com $4bits$ paralelos, transportá-los em uma linha serial e, em seguida, registrar A na saída do circuito.

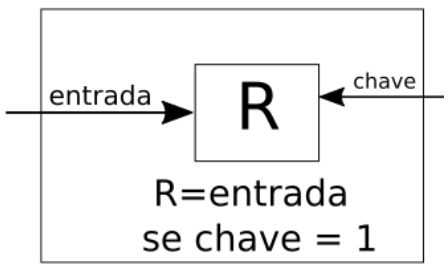
Como dica, observe que o *digital works* oferece *latch SR*. Faça uso dele, podendo também ser feita alteração por sua conta.



Latch: Exercício

Você se lembra deste exercício? Quem é *R*? Implemenete esse sistema no *digital works*





Latch e Flip-Flop *síncronos*

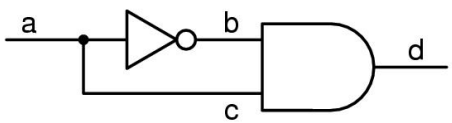
Um latch e um flip-flop têm o mesmo propósito, que é armazenar um *bit*. Contudo, eles se diferenciam quanto ao momento em que tornam-se sensíveis à escrita.
Um latch síncrono é sensível à escrita durante todo o período em que $ck \equiv 1$. Já o flip-flop, seu período de sensibilidade é durante uma transição positiva ($0 \rightarrow 1$) ou negativa ($1 \rightarrow 0$) de ck.

Mas, como gerar um pulso verdadeiro $ck \equiv 1$ durante uma transição na linha ligada à chave ck?

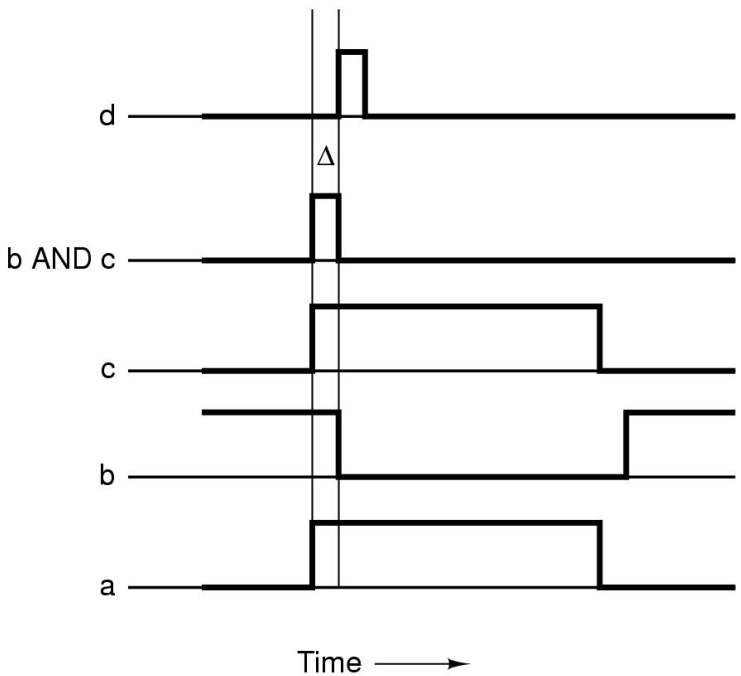
Latch e Flip-Flop *síncronos*

Um latch e um flip-flop têm o mesmo propósito, que é armazenar um *bit*. Contudo, eles se diferenciam quanto ao momento em que tornam-se sensíveis à escrita.
Um latch síncrono é sensível à escrita durante todo o período em que $ck \equiv 1$. Já o flip-flop, seu período de sensibilidade é durante uma transição positiva ($0 \rightarrow 1$) ou negativa ($1 \rightarrow 0$) de ck.

Mas, como gerar um pulso verdadeiro $ck \equiv 1$ durante uma transição na linha ligada à chave ck?

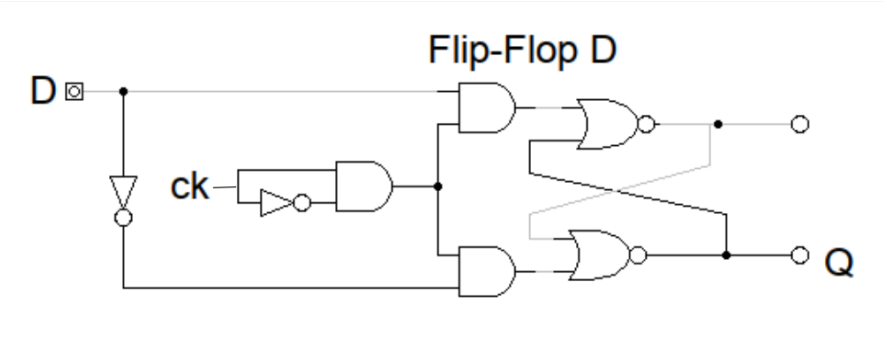


(a)

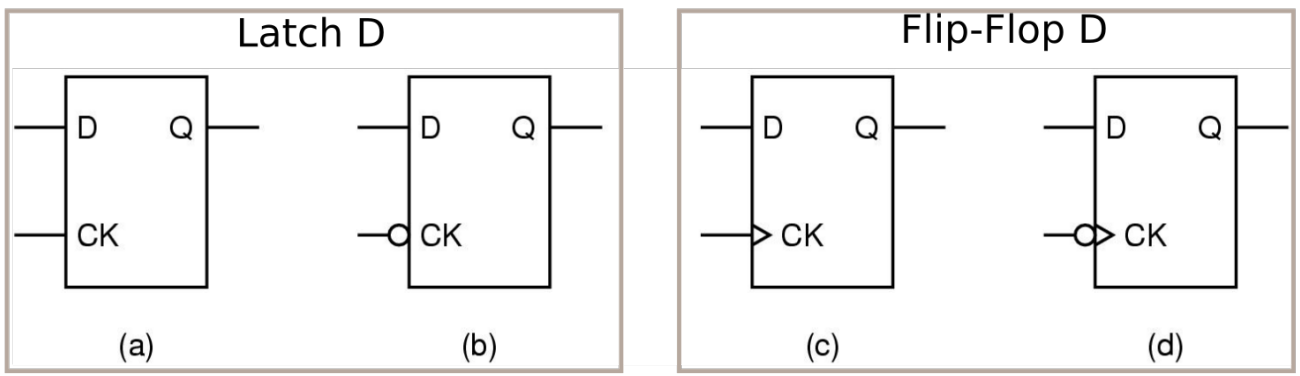


(b)

Latch-D e Flip-Flop-D síncronos



ck	D	Q
0	x	Q_a
↑	0	0
	1	1
1	x	Q_a
↓	x	Q_a

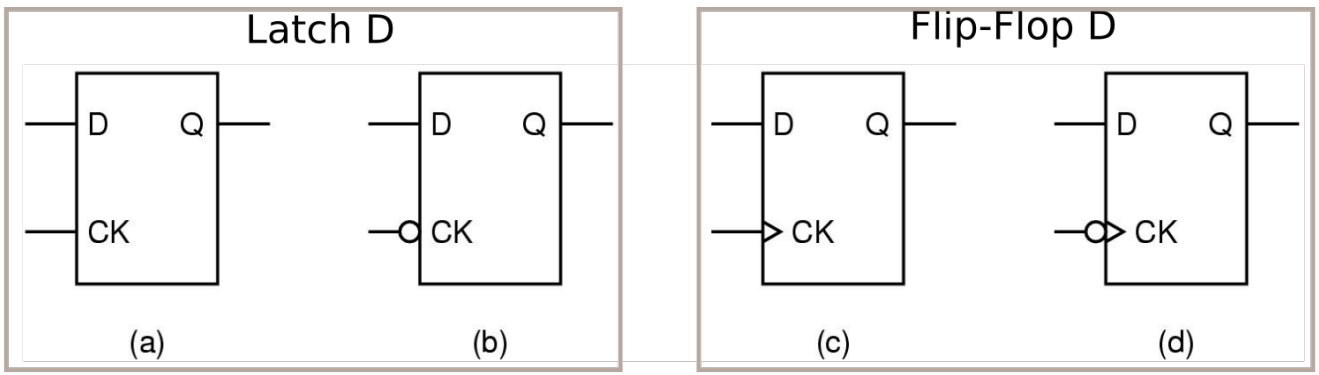


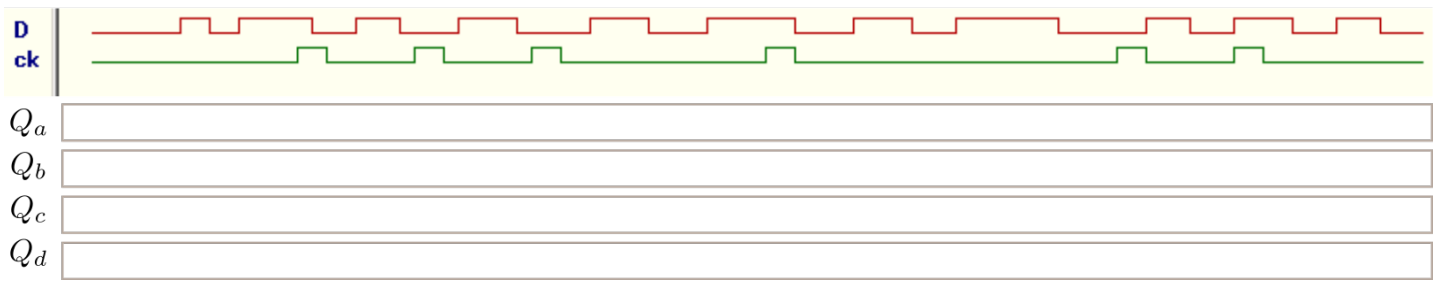
- Latch a) sensível à escrita quando $ck \equiv 1$
- Latch b) sensível à escrita quando $ck \equiv 0$
- Flip-Flop c) sensível à escrita na transição positiva de ck
- Flip-Flop d) sensível à escrita na transição negativa de ck

Unifal MG
Universidade Federal de Alfenas

Latch-D e Flip-Flop-D síncronos: Exercícios

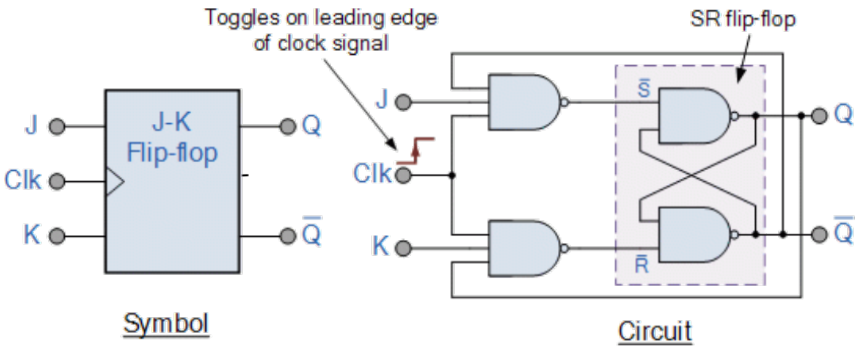
Complete as linhas de sinais para os dois latchs e os dois flip-flops da figura. Considere que cada $Q_i = 0$ em t_0, sendo t o tempo.

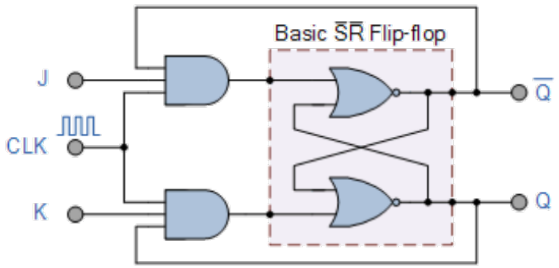




Flip-Flop JK

O flip-flop JK aproveita a possibilidade do erro ocasionado no flip-flop SR. Para isso, as saídas Q e $\overline{Q}$ são ligadas nas chaves do flip-flop. Como elas são complementares, apenas o J ou o K fornecem entrada a cada instante.



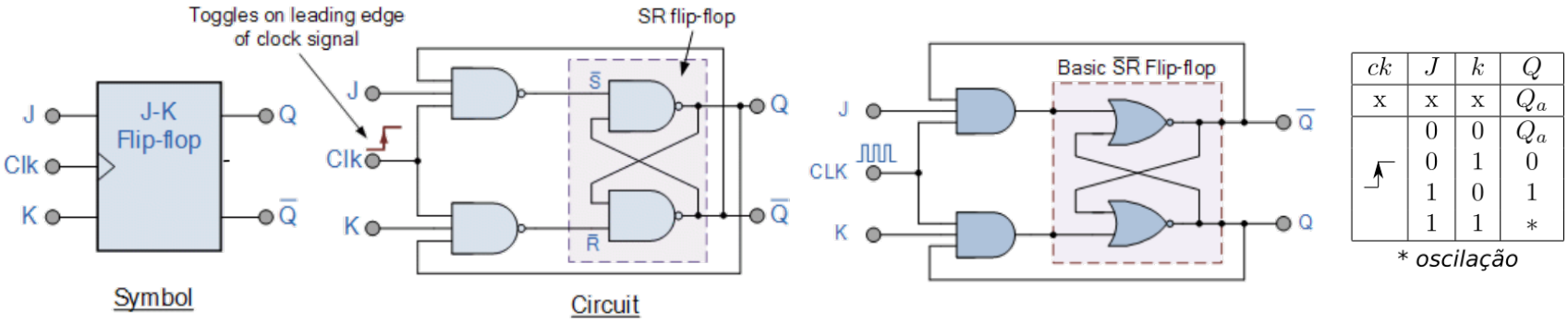


ck	J	k	Q
x	x	x	Q_a
↑	0	0	Q_a
	0	1	0
	1	0	1
	1	1	$*$

* *oscilação*

Flip-Flop JK

O flip-flop JK aproveita a possibilidade do erro ocasionado no flip-flop SR. Para isso, as saídas Q e $\overline{Q}$ são ligadas nas chaves do flip-flop. Como elas são complementares, apenas o J ou o K fornecem entrada a cada instante.



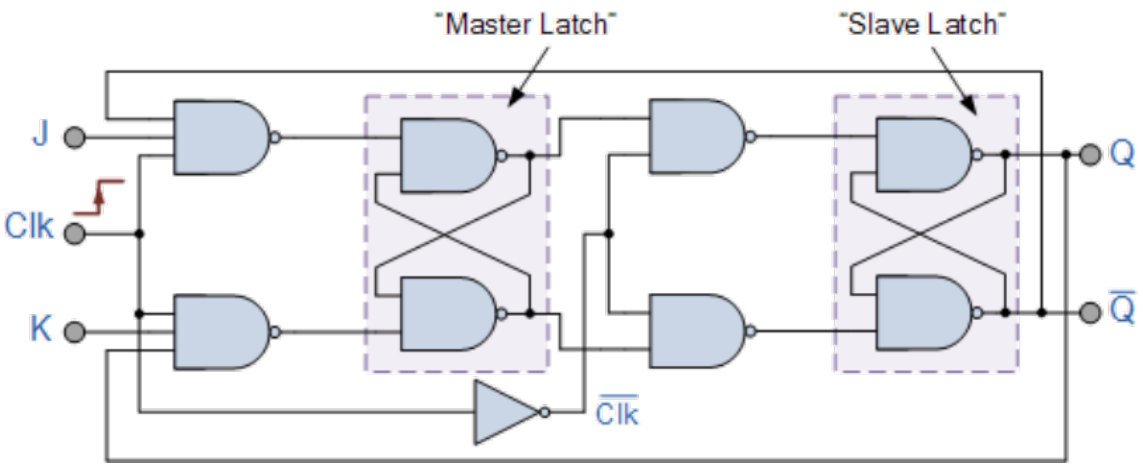
ck	J	k	Q
x	x	x	Q_a
↑	0	0	Q_a
	0	1	0
	1	0	1
	1	1	*

* *oscilação*

Quando $J \equiv 1$ e $K \equiv 1$, Q e $\overline{Q}$ controloam as chaves ligadas às entradas para que: ora J escreva 1 em Q; e ora K escreva 0 em Q. Isso gera um estado de oscilação durante o período da transição de ck.

Flip-Flop JK *Master-Slave*

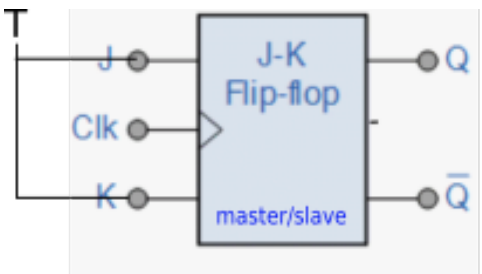
Para aproveitar a possibilidade de oscilação ocorrida quando $J \equiv k \equiv 0$, uma ideia é permitir apenas uma oscilação. Isso faria como que o *flip-flop* alteraria sua saída a cada vez que houvesse uma transição de *ck*. Isso é conveniente e tem muita utilidade. Assim, combinamos dois *flip-flops* de modo a habilitar um por vez, e capturar a oscilação em série.



ck	J	k	Q
x	x	x	Q_a
↓	0	0	Q_a
	0	1	0
	1	0	1
	1	1	$\overline{Q}_a$

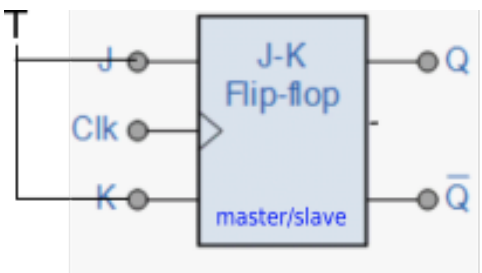
Flip-Flop JK *Master-Slave*: Exercício

Faça a tabela verdade para o *flip-flop T* construído a apartir de um *flip-flop JK Master-Slave*.



Flip-Flop JK *Master-Slave*: Exercício

Faça a tabela verdade para o *flip-flop T* construído a apartir de um *flip-flop JK Master-Slave*.



ck	T	Q
x	x	Q_a
↓	0	Q_a
	1	$\overline{Q}_a$

Flip-Flop JK *Master-Slave*

Flip-Flop JK Master-Slave com Preset e Clear

O flip-flop JK master-slave pode ser melhorado introduzindo-se duas outras entradas muito úteis, a saber, **preset (PR)** e **clear (CL)**. Estas entradas atuam diretamente nas saídas Q e $\overline{Q}$ independente do pulso de clock e do nível lógico das entradas J e K, sendo, por isso, chamadas de **assíncronas**, como mostra a figura 5.15.

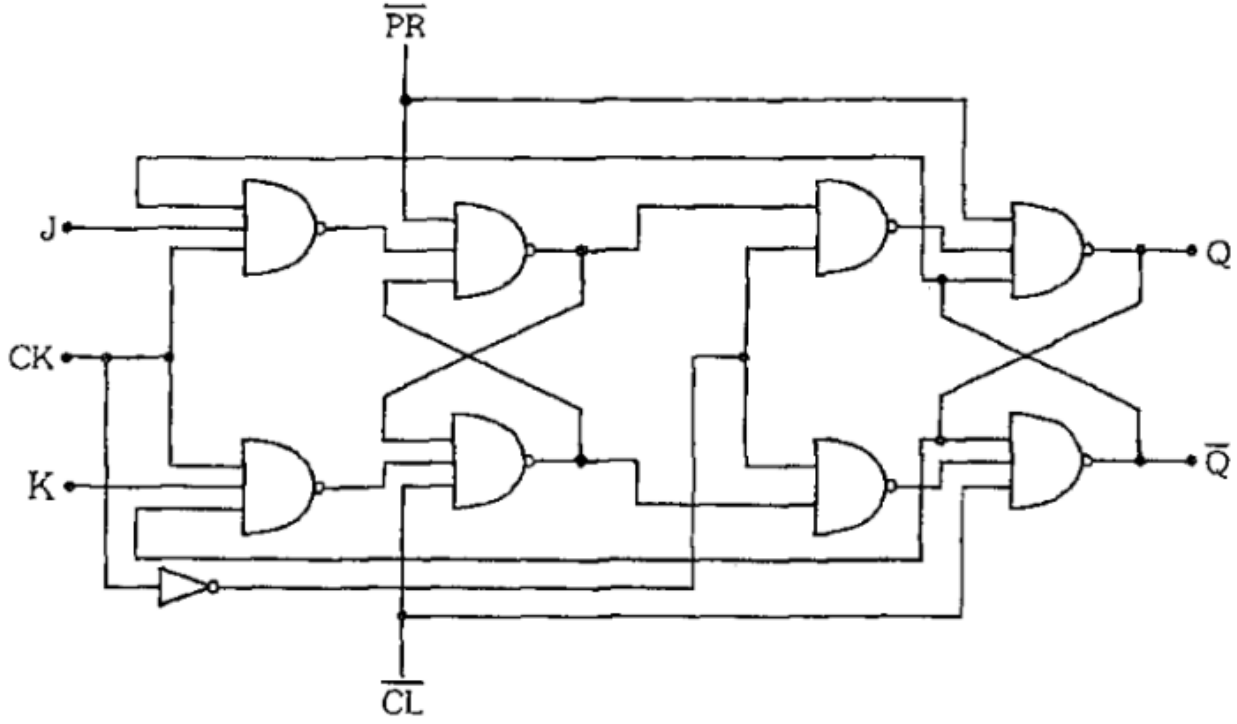


Figura 5.15 - Flip-Flop JK Master-Slave com Preset e Clear

Flip-Flop JK *Master-Slave*

Flip-Flop JK Master-Slave com Preset e Clear

O flip-flop JK master-slave pode ser melhorado introduzindo-se duas outras entradas muito úteis, a saber, **preset (PR)** e **clear (CL)**. Estas entradas atuam diretamente nas saídas Q e $\overline{Q}$ independente do pulso de clock e do nível lógico das entradas J e K, sendo, por isso, chamadas de **assíncronas**, como mostra a figura 5.15.

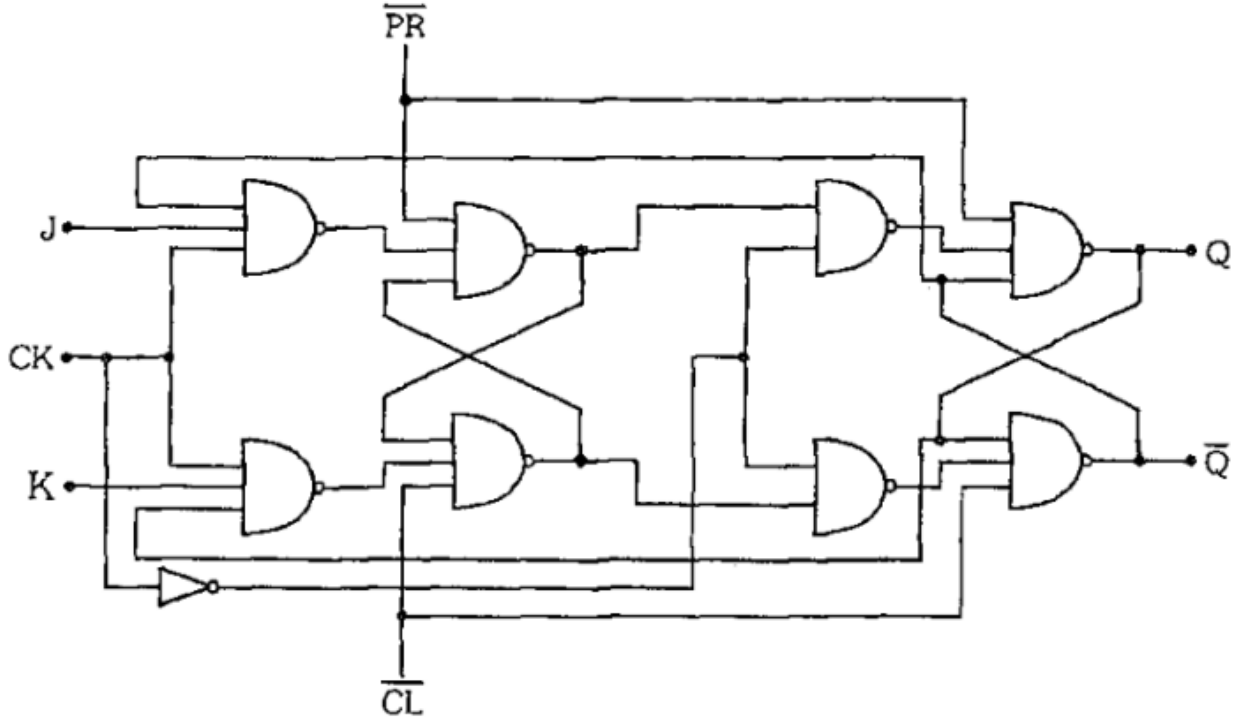
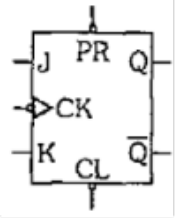


Figura 5.15 - Flip-Flop JK Master-Slave com Preset e Clear

$\overline{PR}$	$\overline{CL}$	CK	J	K	Q_t
1	0	X	X	X	0
0	1	X	X	X	1
1	1	↓	0	0	Q_0
			0	1	0
			1	0	1
			1	1	$\overline{Q}_0$



Flip-Flop JK *Master-Slave*

Exemplo de Aplicação - Divisor de Freqüência

O circuito mostrado na figura 5.21 representa dois flip-flops JK master-slave ligados em cascata, funcionando como um **divisor de freqüência**.

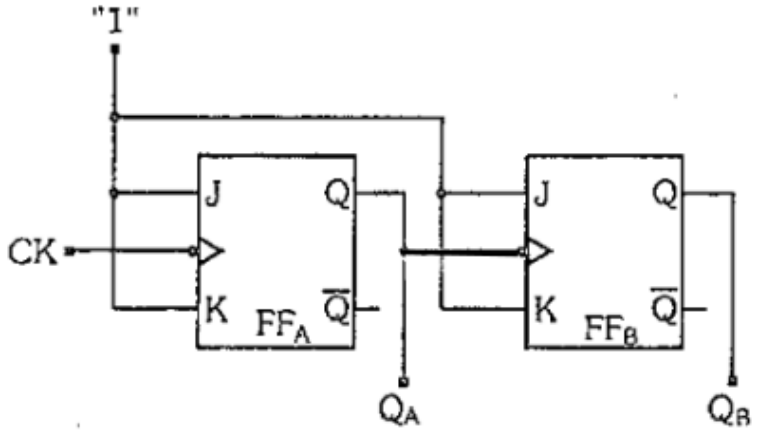


Figura 5.21 - Divisor de Freqüência

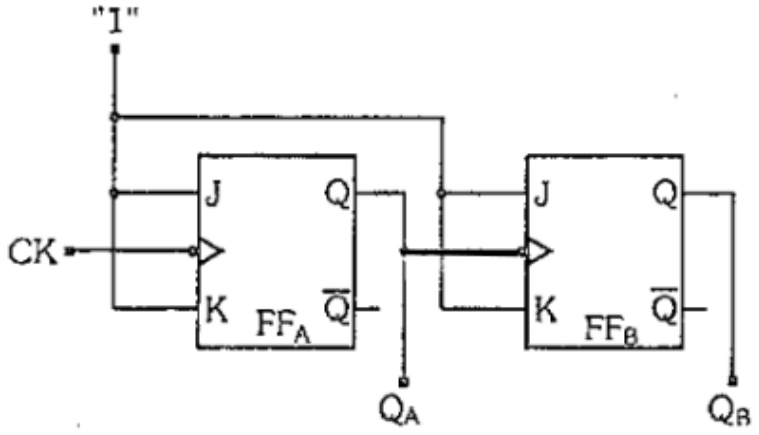
Nota-se pelo circuito que, estando os dois flip-flops com as entradas J e K em nível lógico 1, o primeiro (FF_A) complementa sua saída Q_A a cada transição negativa do pulso de clock e o segundo (FF_B) complementa sua saída Q_B a cada transição negativa da saída Q_A, como mostra o diagrama de tempos da figura 5.22.

↓

Flip-Flop JK *Master-Slave*

Exemplo de Aplicação - Divisor de Freqüência

O circuito mostrado na figura 5.21 representa dois flip-flops JK master-slave ligados em cascata, funcionando como um **divisor de freqüência**.



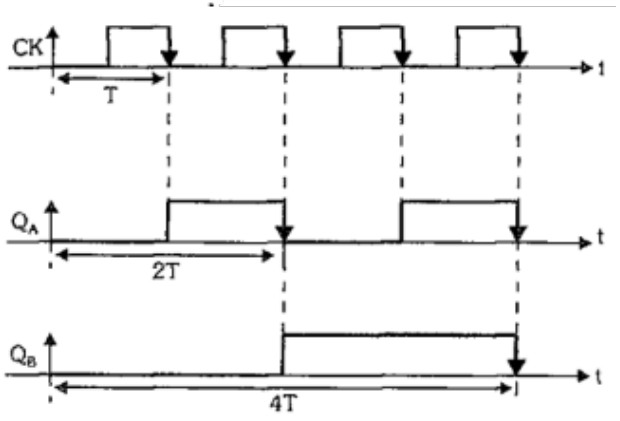


Figura 5.21 - Divisor de Freqüência

Nota-se pelo circuito que, estando os dois flip-flops com as entradas J e K em nível lógico 1, o primeiro (FF_A) complementa sua saída Q_A a cada transição negativa do pulso de clock e o segundo (FF_B) complementa sua saída Q_B a cada transição negativa da saída Q_A, como mostra o diagrama de tempos da figura 5.22.

↓

Contador Assíncrono

O Contador assíncrono tem os pulsos de *clock* não simultâneos.

Exemplo de Aplicação I - Contador Hexadecimal Assíncrono Crescente

O circuito deste contador está representado na figura 7.3.

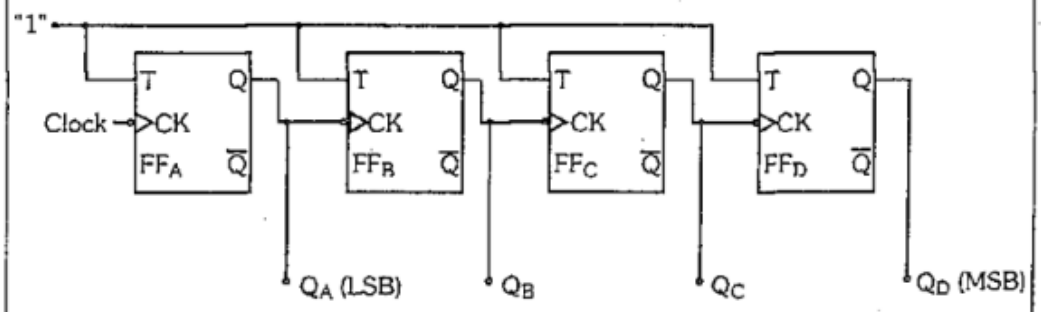


Figura 7.3 - Contador Hexadecimal Assíncrono Crescente

Contador Assíncrono

O Contador assíncrono tem os pulsos de *clock* não simultâneos.

Exemplo de Aplicação I - Contador Hexadecimal Assíncrono Crescente

O circuito deste contador está representado na figura 7.3.

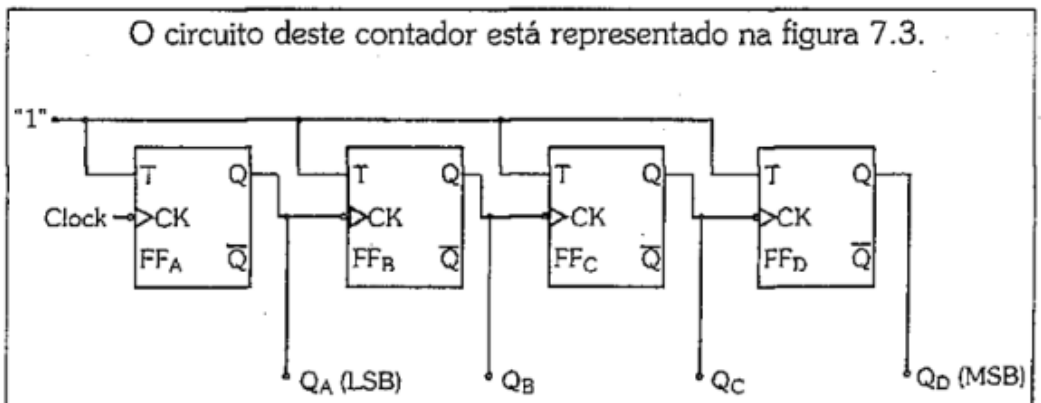
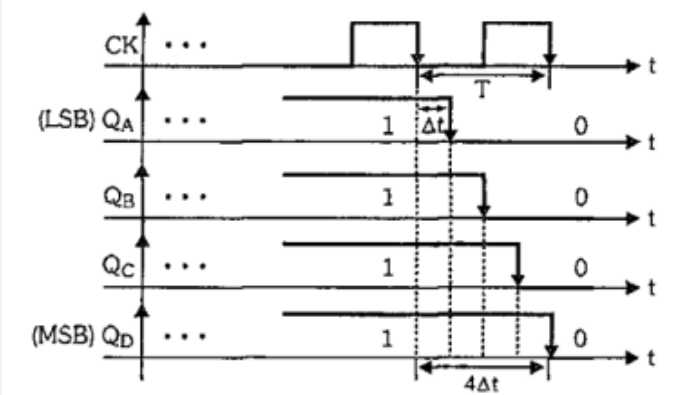


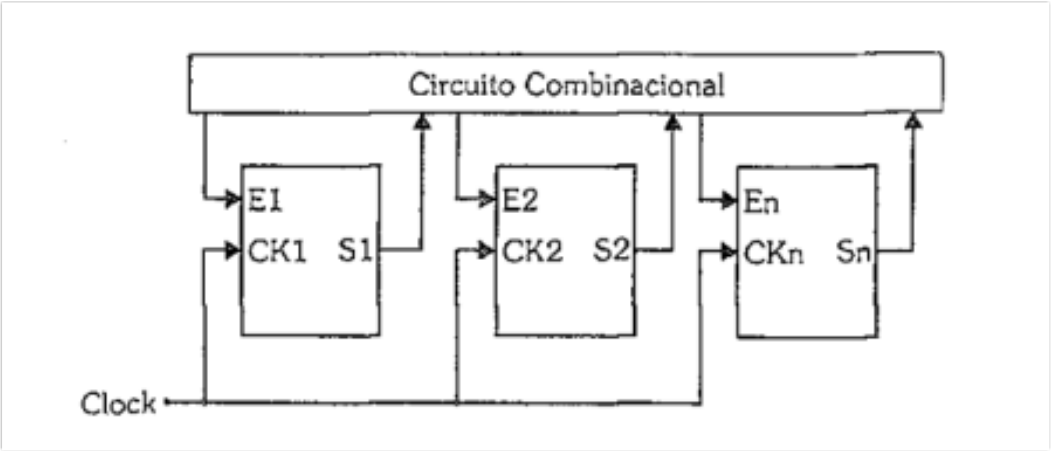
Figura 7.3 - Contador Hexadecimal Assíncrono Crescente



Contador Síncrono

O Contador síncrono tem os pulsos de *clock* simultâneos.

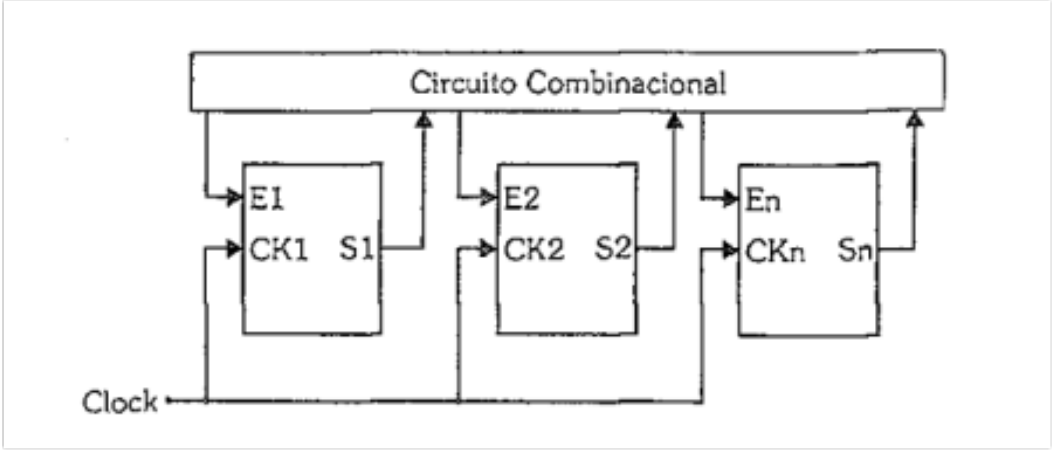
Esquema genérico de contador síncrono



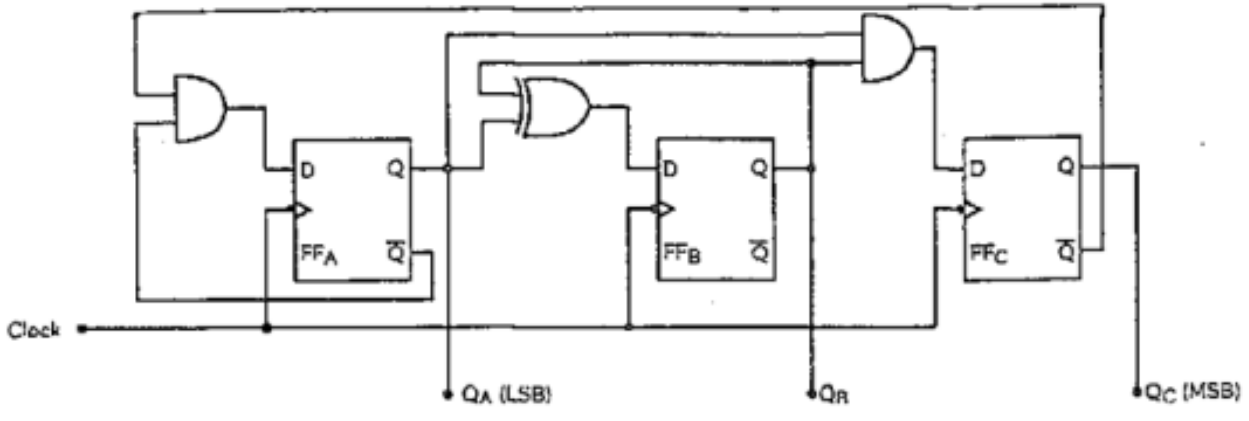
Contador Síncrono

O Contador síncrono tem os pulsos de *clock* simultâneos.

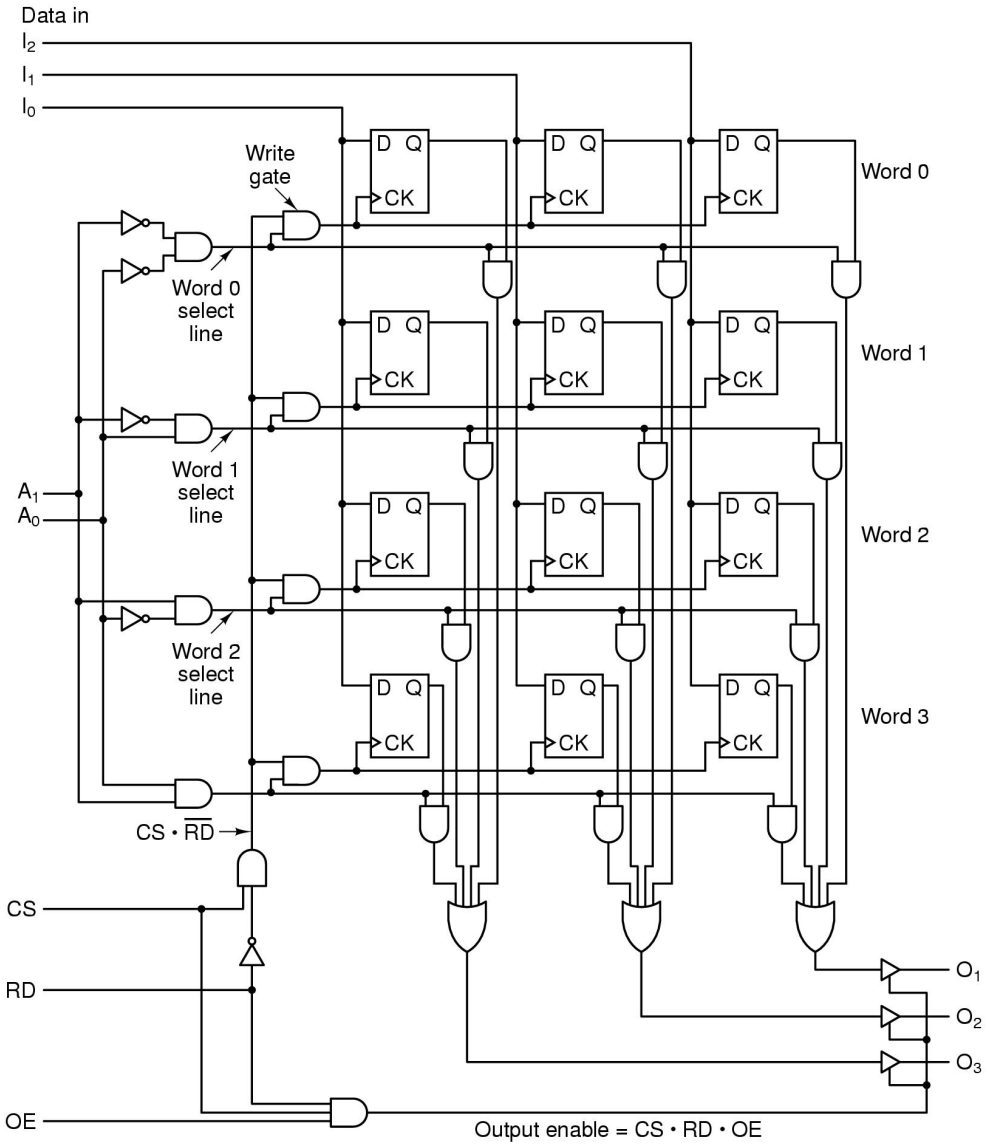
Esquema genérico de contador síncrono



Exemplo de Contador Síncrono com Flip-Flops D



Unidade básica de memória



Agradecimentos

Agradecimentos Especiais:

Agradeço a toda a comunidade LaTeX.
Em especial a *Till Tantau* pelo *Beamer*.
https://www.tcs.uni-luebeck.de/mitarbeiter/tantau/
Desta forma, tornou-se possível a escrita deste material didático.