

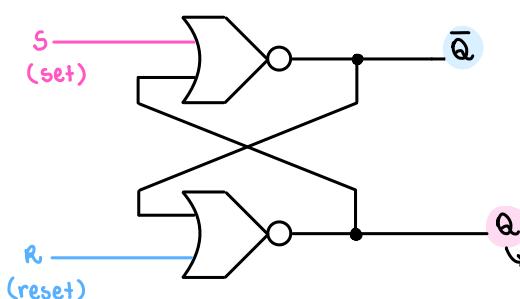
ARQ 1 - PROVA 02

- ★ A saída de um circuito combinacional é uma função que depende apenas das entradas atuais do circuito. Esse tipo de circuito não tem memória, não podemos armazenar bits em um circuito combinacional e mais tarde ler esses bits.
- ★ Um circuito sequencial tem memória. Um circuito sequencial é um circuito cujas saídas dependem não somente das entradas atuais, mas também do seu estado atual, que é o conjunto de todos os bits armazenados no circuito. O estado do circuito, por sua vez, depende da sequência passada de valores que apareceram nas entradas do circuito.

ARMAZENANDO UM BIT: FLIP-FLOPS

- ★ O método básico usado para armazenar um bit em um unidade digital é a REALIMENTAÇÃO (feedback).
- ★ Em uma porta lógica, sabemos que de algum modo precisamos alimentar a entrada inversa de volta à saída que foi anteriormente produzida, de modo que o bit armazenado fique saindo voltar, como um círculo que corre atrás de seu próprio rabo.

LATCH SR BÁSICO (NOR)



Quando a entrada S é 1, a saída Q é obrigada a se tornar 0.

Quando a entrada R é 1, a saída Q é obrigada a se tornar 1.

↳ É ESSA SAÍDA QUE IMPORTA!

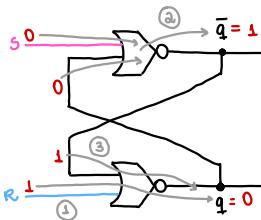
Fazer ambas S e R iguais a 0 faz com que o valor de Q, seja qual for, permaneça inalterado entre a saída e a entrada.

S "faz um set" no latch obrigaindo Q a ser 1, ou garante que R "faz um reset" no latch obrigaindo-o a ser 0.

⚠ Lembre-se de que uma porta NOR apresenta 1 na saída apenas quando todas as entradas da porta são iguais a 0. Se pelo menos uma das entradas for 1, a porta NOR dará 0 na saída.

LIVRO:

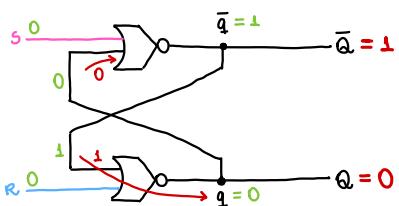
p. 101 → atrasos de tempo



Suponha que fazemos $S=0$ e $R=1$, e que inicialmente não sabíamos quais as valores de Q e \bar{Q} . Como a porta superior do circuito tem pelo menos uma entrada igual a 1 (R), a porta terá um 0 em sua saída obviamente.

No circuito, o 0 da Q realmente é a porta NOR superior, que terá ambas as entradas em 0 e, portanto, a saída igual a 1.

mas o ciclo não para por aí, ele se repete em um loop infinito até que outros valores sejam inputados em S e R . no circuito, esse 1 realmente é a porta NOR inferior, que terá, novamente, pelo menos uma das entradas igual a 1 (na realidade, ambas as entradas são 1), e assim a porta inferior continuará a ter 0 em sua saída, reiniciando

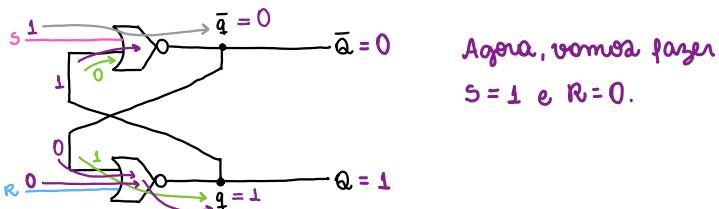


Agora, suponha que fazemos $S=0$ e $R=0$.

A NOR inferior ainda tem pelo menos um dos inputs iguais a 1 (a entrada que vem da saída da porta superior), de modo que a porta inferior continua a fornecer um 0 como output.

A porta superior continua a ter ambas as entradas iguais a 0 e continua a produzir um 1 em sua saída.

Desse modo, o $R=1$ anterior armazenou um 0 no latch SR. Isso também é conhecido como resetting do latch. Esse 0 na saída Q continuará armazenado mesmo quando fizermos R retornar a 0.



Agora, vamos fazer
 $S=1$ e $R=0$.

A porta superior do circuito tem agora a entrada S como 1, de modo que ela fornece 0 na sua saída. Zero é o que entra na NOR inferior juntamente com o input de reset, que também é 0, gerando 1 como output. Esse 1 é realimentado à NOR superior, que já tinha o 1 de S anyway, então sua saída permanece 0 e o ciclo continua.

Se fizermos $S=0$ e $R=0$ novamente, Q continuaria sendo 1. Desse modo, o $S=1$ anterior armazenou um 1 no latch SR. Isso também é conhecido como setting do latch. Esse 1 permanecerá "armazenado" em Q mesmo se fizermos S retornar a 0.

Quando $S=0$ e $R=0$, Q não muda, então chamamos essa situação de "memória".

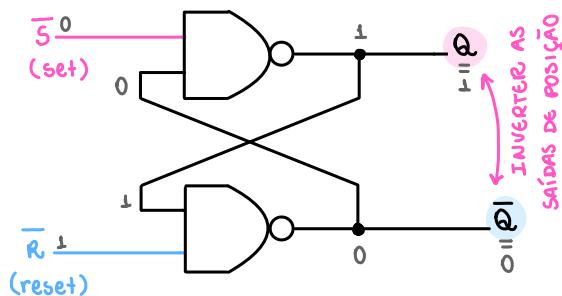
S	R	Q	\bar{Q}
0	0	Q	\bar{Q}
0	1	0	1 → resetado
1	0	1	0 → setado
1	1	?	?

As entradas S e R nunca devem sair ambas iguais a 1 em um latch SR.

USO normal

Um problema que ocorre com o latch SR báscio é que, quando ambas as entradas S e R são 1 ao mesmo tempo, pode ocorrer um comportamento indefinido - um 1 pode ter saída de memória, um 0 pode ter saída armazenada, ou ainda a saída do latch pode ter entrada em excitação e ficar trancando o output.

LATCH SR BÁSICO (NAND) (ativo em nível baixo, lógica negativa)

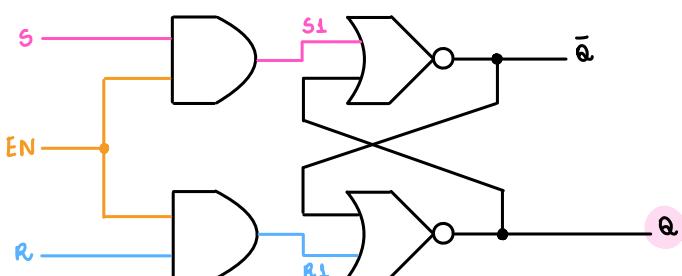


S	R	Q	\bar{Q}
0	0	?	?
0	1	1	0 → setado (memória principal é 1)
1	0	0	1 → resetado (memória principal é 0)
1	1	Q	\bar{Q} → memória (mantém a saída da jeito que estava)

A porta NAND vai produzir saída 0 quando ambas as entradas são 1. De pelo menos um das outras é 0, a NAND terá 1 como output.

LATCH SR CONTROLADO (sensível ao nível)

Uma solução parcial para lidar com a corrente proibida é acrescentar uma entrada de habilitação EN (enable) ao latch SR.



A introdução da entrada de habilitação leva-nos à ideia de tornar a habilitação igual a 1 apenas quando estivermos seguros de que S e R têm valores adequados.

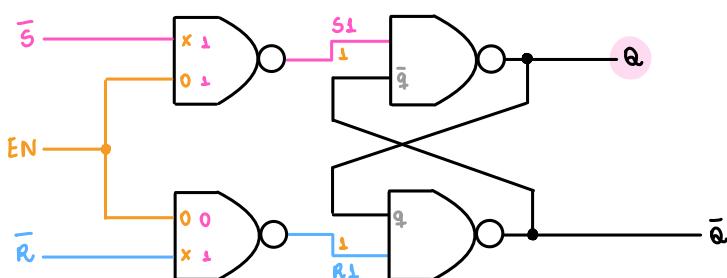
Bons EN=1, as saídas S e R propagam-se sem modificações através da sua porta AND até as entradas S1 e R1 do circuito do latch SR báscio, porque $S \cdot 1 = S$ e $R \cdot 1 = R$.

Bons EN=0, as portas AND fazem com que, inevitavelmente, S1 e R1 sejam 0, independentemente das valores de S e R. Assim, quando EN=0 temos a condição de memória, isto é, o valor de saída do latch SR báscio não pode ser alterado.

Um latch SR com uma habilitação é conhecido como latch SR sensível ao nível, porque o latch é sensível a suas entradas S e R apenas quando o nível da entrada de habilitação é 1. Também é chamado de latch transparente, porque quando $EN=1$, o latch SR interno torna-se transparente às entradas de set e reset.



LATCH SR CONTROLADO COM PORTAS NAND



Quando habilitado, possui a mesma tabela-verdade de um latch SR feito com portas NOR!
O "wearáu" não precisa inverter os valores de S e R, o circuito já está fazendo isso p/ ele. →
⊕ intuitivo

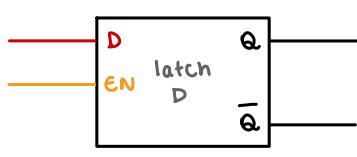
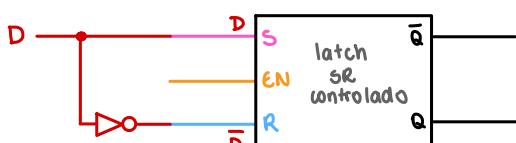
Aqui, temos a inversão da lógica negativa quando o circuito está habilitado. Quando $EN=0$, não há surpresa: S_1 e R_1 sempre serão 1, o que configura o cenário de memória, no qual independente dos valores inputados em S e R, as saídas Q e \bar{Q} não serão alteradas. No entanto, a loucura ocorre quando a habilitação do circuito se forçar $EN=1$. Nesse caso, o latch de portas NAND reage aos inputs como se fosse um latch de portas NOR! Dessa maneira, se $S=1$ e $R=0$, inverte SETAR o circuito e Q será 1, pois a primeira filera de portas NAND irá inverter os inputs para nós, produzindo $S_1=0$ (pois $\overline{1 \cdot 1} = 0$) e $R_1=1$ (pois $\overline{1 \cdot 0} = 1$)

EN	S	R	S_1	R_1	Q	\bar{Q}
0	X	X	1	1	Q	\bar{Q}
1	0	0	1	1	Q	\bar{Q}
1	0	1	1	0	0	1 → resetado
1	1	0	0	1	1	0 → setado
1	1	1	0	0	?	?

memória }
uso normal }

A não ser que JÁ DUAS entradas sejam 1, a saída de uma NAND é 1.
Inclusive, se usarmos continua sendos $R=1$ e $S=1$, para que jamais tenhamos $R_1=S_1=0$.

LATCH TIPO D



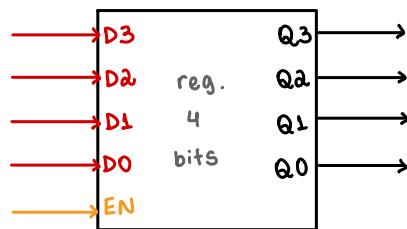
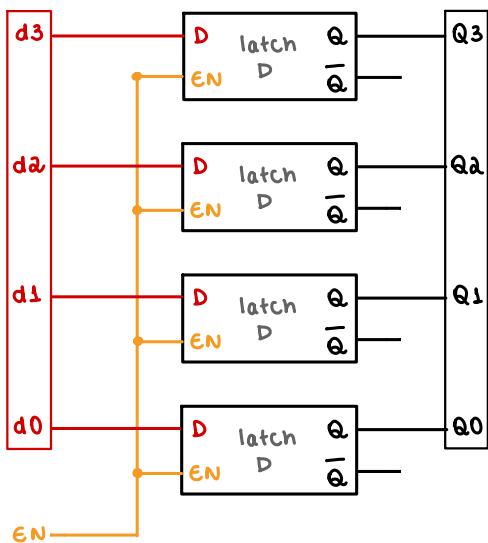
Resolva de uma vez por todas o caso proibido! Se estiver desabilitado, temos configurações de memória como de praxe, onde Q e \bar{Q} não mudam. Ao habilitar ($EN=1$), o bit inputado em D é simplesmente copiado para a saída principal.

↳ S sempre será o oposto de R!

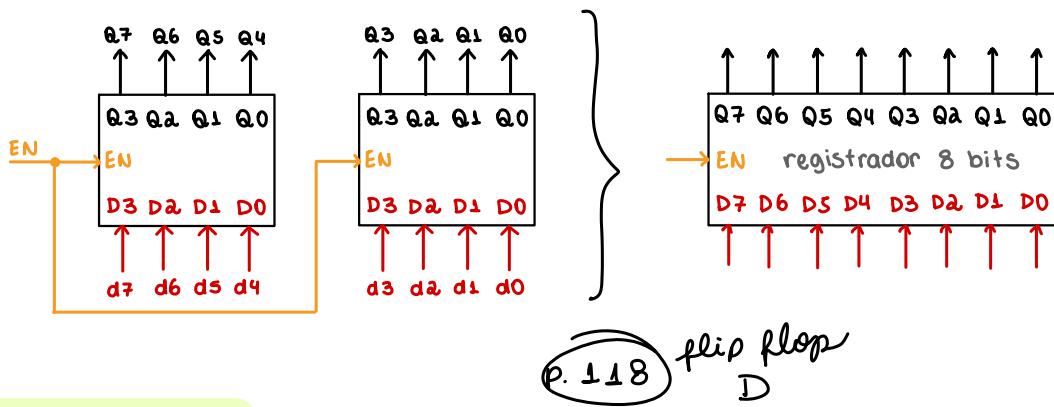
EN	D	Q	\bar{Q}
0	X	Q	\bar{Q}
1	d	d	\bar{d}

→ memória
→ cópia

REGISTRADOR DE 4 BITS



REGISTRADOR DE 8 BITS



0.118 flip flop

LATCH VS. FLIP-FLOP

- * Um latch é sensível ao nível e um flip-flop é sensível à borda.
- * Operação sensível ao nível: latches monitoram continuamente suas entradas e podem alterar suas saídas de imediato enquanto o sinal de habilitação (EN) está ativo. Quando desabilitado, o latch é "transparente", ou seja, mudanças na entrada são imediatamente refletidas na saída.
- * Operação disparada por borda: flip-flops mudam seu estado (seu output) apenas em instantes específicos, ou seja, na borda de descida ou de subida de um sinal de clock. A entrada é processada apenas no momento da transição do clock, tornando os flip-flops adequados para circuitos síncronos e previnindo alterações indesejadas devido à ruído ou glitches.
- * A transparência dos latches pode causar loops de realimentação indesejada, levando a condição de corrente.
- * O flip-flop amostra sua entrada apenas na transição do clock, ignorando mudanças de entrada em outros momentos.

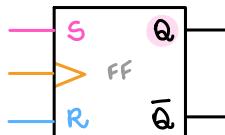
Diagrama temporal de um latch vs. um flip-flop



RELOGIOS

- * A maioria dos circuitos sequenciais simplesmente usa um sinal de habilitação que pula a taxa constante.
- * Esse sinal de habilitação pulsante é chamado de sinal de CLOCK (CLK).
- * Os projetistas normalmente usam um oscilador para gerar um sinal de relógio.
- * O período de um sinal de clock é o intervalo de tempo após o qual o sinal volta a val repetir - ou mais simplesmente, o intervalo de tempo entre 2 sucessivas.

FLIP-FLOP SR

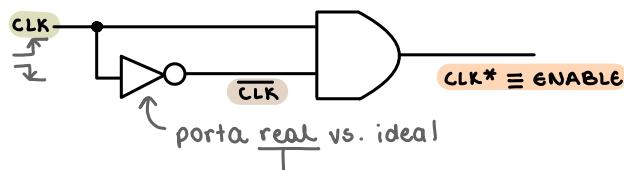


O flip-flop trabalha instantaneamente nos segundos após a saída do sinal de clock.

Circuito equivalente:



Detector de borda positiva:



possui um tempo de atraso de propagação (TP) de aprox. 10 nanosegundos.

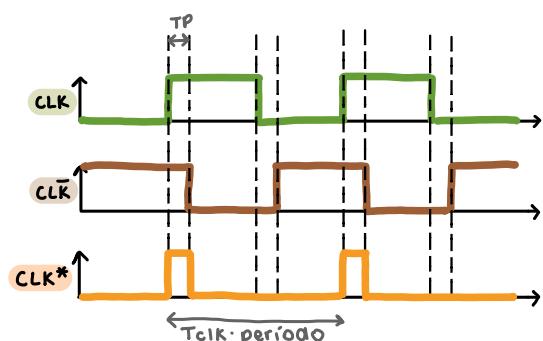
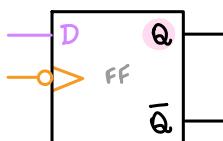


Tabela verdade do FF SR:

CLK	S	R	Q	\bar{Q}
—	X	X	Q	\bar{Q}
↑	0	1	0	1
↑	1	0	1	0
↑	1	1	?	?

→ memória
→ resetado
→ setado
→ proibido

FLIP-FLOP D



- * Possui apenas uma entrada, eliminando o uso proibido.
- * Simplesmente armazena o dado inputado, transferindo-o para a saída principal.

Círculo equivalente:

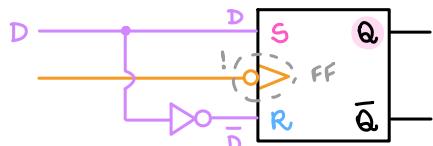


Tabela verdade:

CLK	D	Q	\bar{Q}
X	X	Q	\bar{Q}
\uparrow	0	\bar{Q}	Q

\rightarrow memória

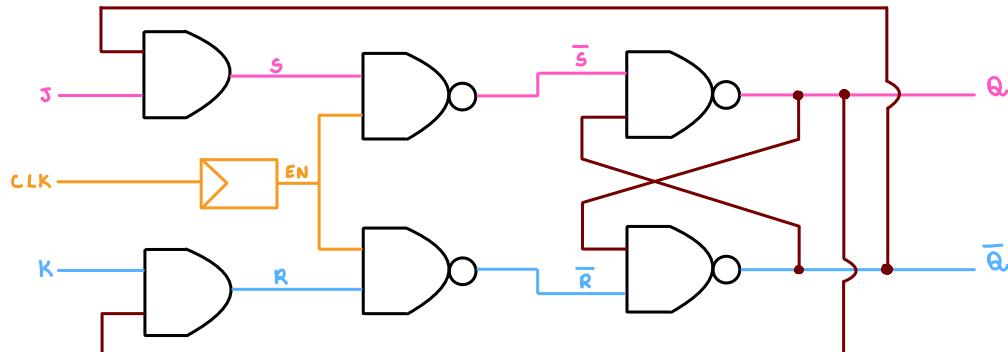
\downarrow d d \bar{d} \bar{d} \rightarrow copia dado p/ a saída

! Borda de descida: funciona quando o clock passa de 1 para zero.

FLIP-FLOP JK

- * O FF JK é muito útil pois elimina o uso proibido ao introduzir a função de troca (toggle).
- * Agora, podemos ativar o J e o K ao mesmo tempo e acionar uma funcionalidade adicional e útil. No FFSR, era simplesmente proibido que os 2 inputs fossem 1 simultaneamente.

Círculo de flip-flop JK (usamos um latch controlado de portas NAND)



CLK	J	K	Q	\bar{Q}
X	X	X	Q	\bar{Q}
\uparrow	0	0	\bar{Q}	Q
\uparrow	0	1	0	1
\uparrow	1	0	1	0
\uparrow	1	1	\bar{Q}	Q

} memória

\rightarrow resetado

\rightarrow setado

\rightarrow troca (toggle)

ENTRADAS ASSÍNCRONAS DE PRESET E CLEAR

- * Servem para definir diretamente o estado do flip-flop independentemente do clock ou das entradas J e K. Elas são usadas para forçar o flip-flop a um estado específico (set ou reset) de forma imediata, sem esperar pela próxima transição do clock.
- * Úteis em situações onde é necessário inicializar ou redefinir o estado dos flip-flops em um circuito digital.
- * Suas ações têm efeito instantâneo.

- **PRESET (PRE):** Quando ativado, força a saída Q a 1.

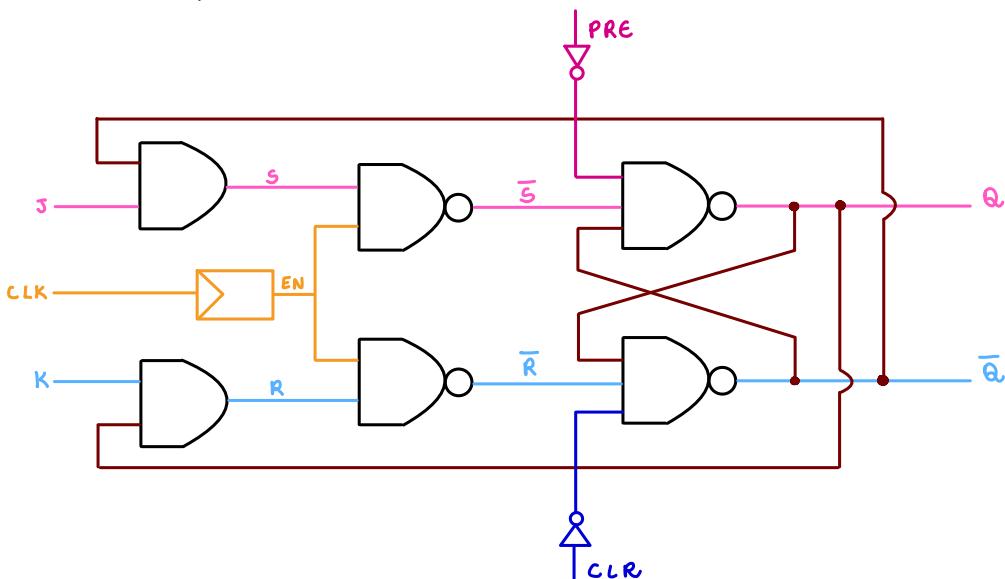
- **CLEAR (CLR):** Quando ativado, força a saída Q a 0.

Tabela verdade

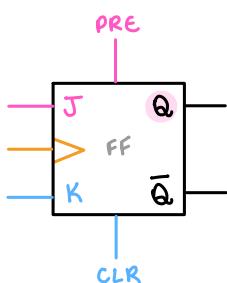
CLK	PRE	CLR	J	K	Q	\bar{Q}
—	1	0	X	X	1	0
—	0	1	X	X	0	1
—	1	1	X	X	?	?
↑	0	0	0	0	Q	\bar{Q}
↑	0	0	0	1	0	1
↑	0	0	1	0	1	0
↑	0	0	1	1	\bar{Q}	Q

* Quando CLR=0 e PRE=0, o estado da saída depende exclusivamente das entradas síncronas J e K e do clock.

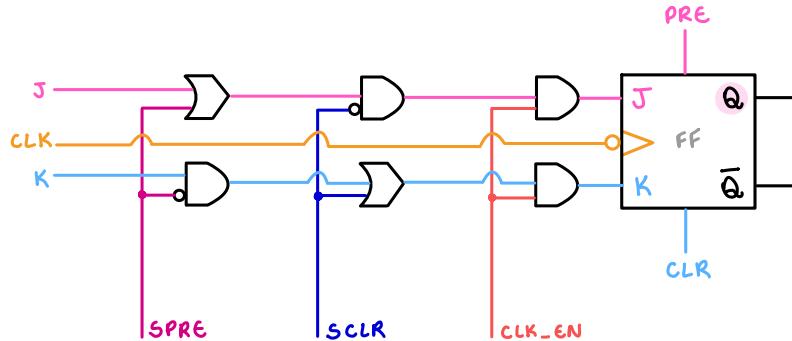
- * Estado conhecido no start-up: Ao ligar um sistema digital, é crucial que todos os flip-flops estejam em um estado conhecido. Usar a entrada CLR para resetar todos os FFs garante que o circuito comece em um estado previsível.
- * Outras utilidades são: sincronização de circuitos, interrupções emergenciais e a implementação de funções especiais.
- * Normalmente, o CLR e o PRE são ativos em nível lógico baixo. Se queremos que sejam ativados com o input=1, devemos implementar uma porta inversora.



Bloco lógico:



ADICIONANDO AS ENTRADAS SÍNCRONAS SPRE E SCLR + ENABLE CLOCK (EN-CLK)



* Esse FF é o mais flexível possível.
Nem sempre todos esses componentes serão usados.

Síncrono		assíncrono		(S) (R)					
SCLR	SPRE	CLR	PRE	CLK-EN	CLK	J	K	Q	Q̄
X	X	1	0	X	—	X	X	0 1 → CLR (resetado)	} assíncrono
X	X	0	1	X	—	X	X	1 0 → PRE (setado)	
1	0	0	0	1	—	X	X	0 1 → reset síncrono (SCLR)	
0	1	0	0	1	—	X	X	1 0 → set síncrono (SPRE)	
0	0	0	0	0	X	X	X	Q Q̄	} memória
0	0	0	0	1	—	X	X	Q Q̄	
0	0	0	0	1	—	0	0	0 1 → S	X = "don't care", não importa
0	0	0	0	1	—	1	0	1 0 → R	
0	0	0	0	1	—	1	1	1 1 → Troca	

* Se EN-CLK for 0, o flip-flop faz memória, pois é a mesma coisa do pulso do clock "não chegar" para o FF.

* As entradas síncronas de PRE e CLR são utilizadas para definir ou resetar o estado do FF de forma sincronizada com o sinal de clock. Ao contrário das entradas assíncronas, que atuam imediatamente independentemente do clock, SPRE e SCLR influenciam o estado do flip-flop somente na transição ativa do clock (borda de subida ou descida).

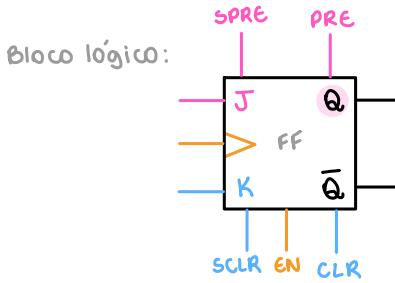
- **RESET síncrono:** Quando ativado, força a saída Q a 0 na próxima transição ativa do clock.
- **CLEAR síncrono:** Quando ativado, força a saída Q a 1 na próxima transição ativa do clock.

* Dependência do clock !!! O input dessas entradas só tem efeito na borda ativa do clock, sincronizando a alteração de estado com o restante do circuito.

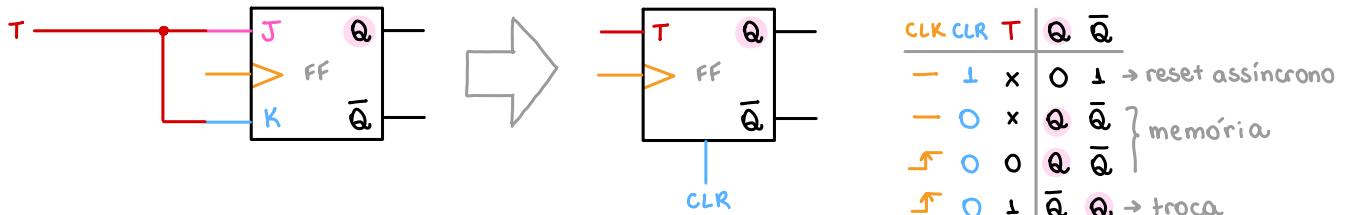
* Prioridade sobre as entradas J e K: se ativadas, SPRE e SCLR determinam o estado do flip-flop exclusivamente, não importando os valores de J e K.

- **ENABLE CLOCK:** sinal de controle que determina se o flip-flop deve capturar a entrada na próxima transição ativa do clock ou manter seu estado atual. Em suma, a habilitação é utilizada para controlar quando o FF deve responder ao sinal de clock.

* Se EN=1, o FF funciona normalmente. Na próxima borda ativa do clock, ele captura a entrada e atualiza a saída Q. Se EN=0, o FF ignora as transições do clock e mantém seu estado atual (faz memória), independentemente das mudanças nas entradas.



FLIP-FLOP T → Alterna seu estado quando sua entrada T está em nível alto (1). Se T=0, memória.

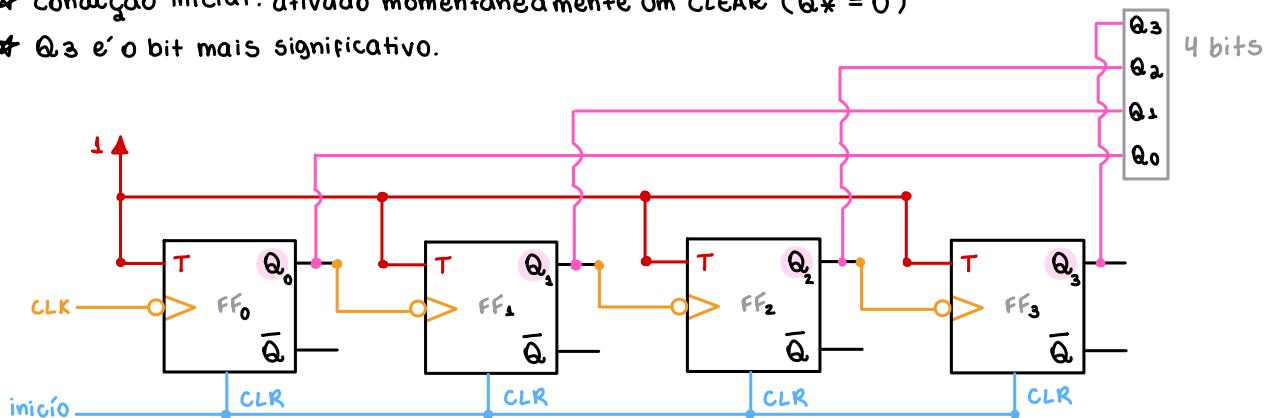


RIPPLE COUNTER (CONTADOR ASSÍNCRONO)

* Contador assíncrono, crescente, borda de descida, módulo 16 (conta de 0 a 15, pois possui 4 bits de saída).

* Condição inicial: ativado momentaneamente um CLEAR ($Q_* = 0$)

* Q_3 é o bit mais significativo.



* Nesse contador, o clock é aplicado apenas ao primeiro flip-flop. Os flip-flops subsequentes são acionados pelas saídas dos flip-flops anteriores, resultando em uma propagação do sinal de clock ao longo dos FFs (daí o nome "ripple", que significa ondulação).

① Estado inicial (contagem=0): todos os FFs estão em 0 ($Q_3 Q_2 Q_1 Q_0 = 0000$).

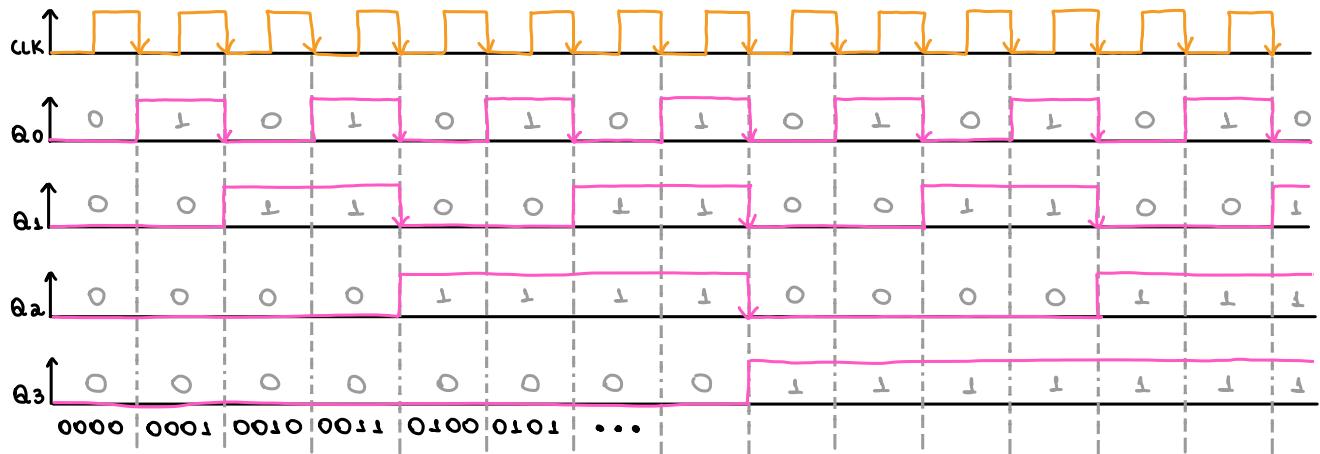
② Primeira transição (quando o clock EXTERNO desce de 1 para 0):

- FF₀: inverte seu estado, Q₀ passa de 0 para 1.
- FF₁: não sofre alteração, continua fazendo memória, pois seu clock (Q₀) não mudou ainda.
- Contagem = 0001 (1).

③ Segunda transição:

- FF₀: quando o clock externo desce novamente, Q₀ inverte de 1 para 0.
- FF₁: reage à borda de descida de Q₀. FF₁ inverte seu estado e Q₁ passa de 0 para 1.
- FF₂: não sofre alteração, pois ainda não houve clock negativo feito por Q₁, continua fazendo memória e preservando sua saída igual à zero (assim como FF₃).
- Contagem = 0010 (2).

E o processo continua...

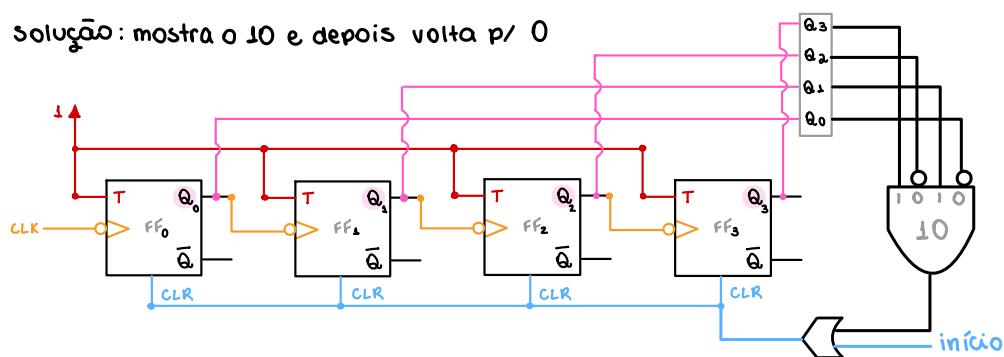


* Vantagens: simplicidade de implementação e baixo consumo de energia, não precisa de portas lógicas adicionais, só encadear um FF na frente do outro.

* Desvantagens: atraso de propagação acumulado pode causar inconsistências temporárias nas saídas (glitches) durante as transições.

RIPPLE COUNTER DE MÓDULO 10 (0 a 9) COM 4 FLIP-FLOPS

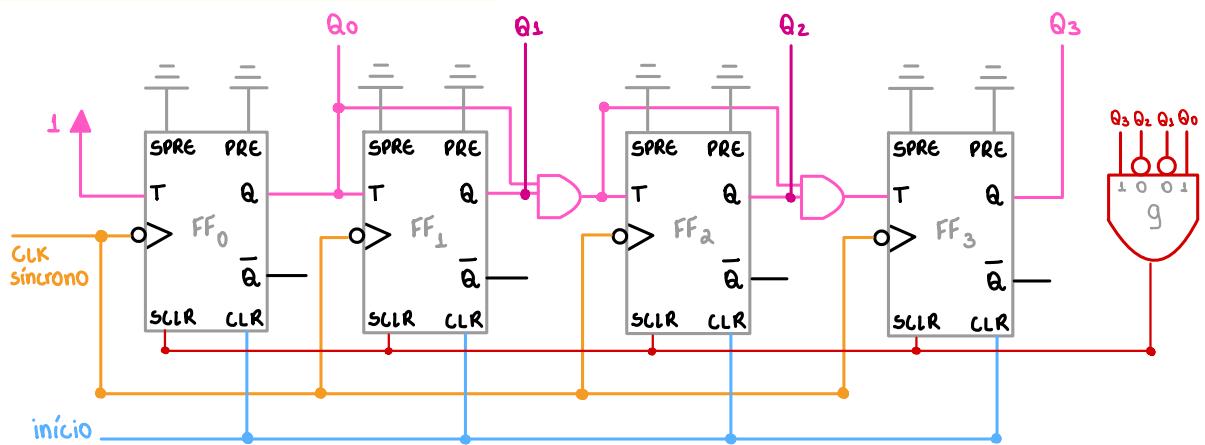
SOLUÇÃO: mostra o 10 e depois volta p/ 0



* Utilizar um clear síncrono em um ripple counter não resetará todos os flip-flops simultaneamente, pois eles não compartilham o mesmo clock. Cada FF só é acionado com a modificação da saída do FF anterior.

* Se alterarmos a conexão $Q \rightarrow CLK$ para $\bar{Q} \rightarrow CLK$, teremos um contador decrescente.

CONTADOR SÍNCRONO MÓDULO 10 (0 a 9)



CONTADOR SÍNCRONO DE 3 a 12

1100 a 1100

