Circuitos Combinacionais:

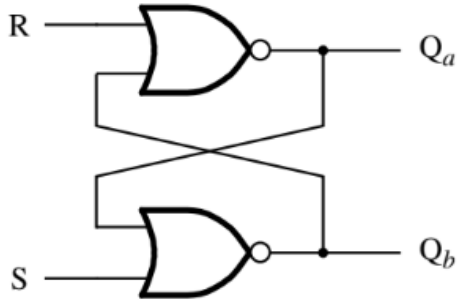
- O que vimos até o semestre anterior
- Depende somente das entradas

Circuitos sequenciais:

- Depende das entradas e do estado anterior do cirucito
- Contem memória
- Dependendo das entradas ele pode mudar o estado ou continuar no anterior

Latch (aquela imagem q n ta no livro)

Latch báisco:



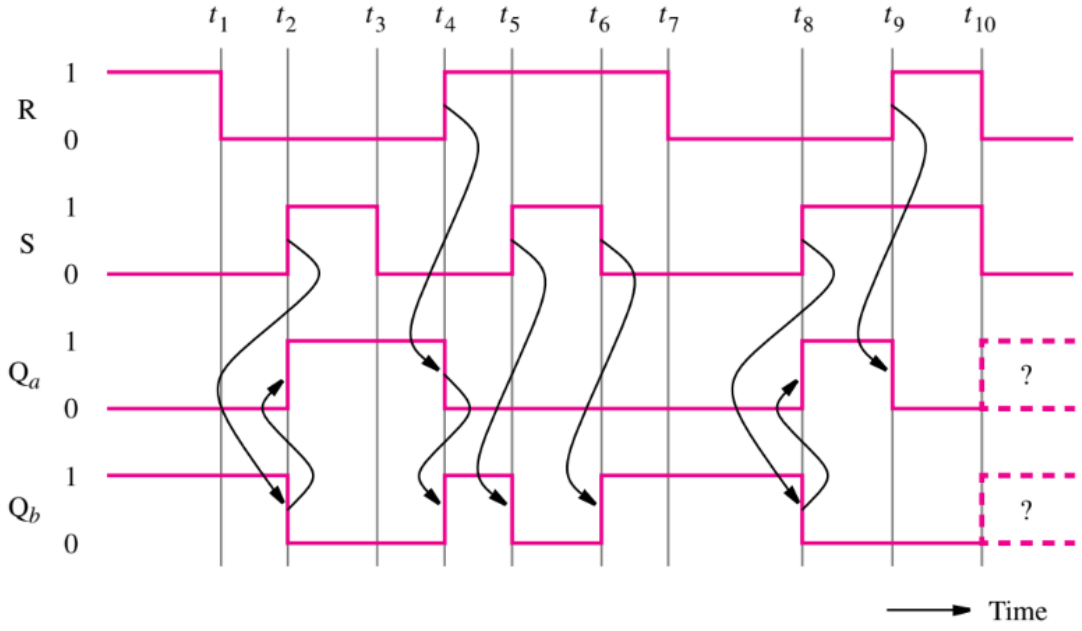
(a) Circuit

S	R	Q_a	Q_b	
0	0	0/1	1/0	(no change)
0	1	0	1	
1	0	1	0	
1	1	0	0	

(b) Characteristic table

Quando S (set) e R (reset) são 0 ele mantem o mesmo estado

Quando 1 das entradas do NOR for 1 ele já ira produzir 0.



O estado 0 0 nunca deve ocorrer como primeiro estado, pois você não sabe o que havia antes no circuito.
Quando S = R = 0 e somente 1 deles muda de valor, podemos perceber que há um certo delay acontecendo. Em t2, quando S vai para 1, isso automaticamente faz com que Qb seja

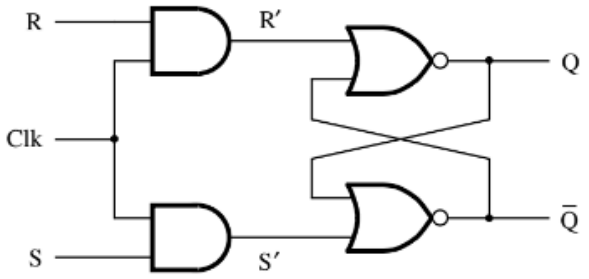
0, ai, com Qb sendo zero podemos inferir o valor de que Qa, que neste caso será 1. Ou seja, tem um certo delay até que o Qb seja definido para ai posteriormente o Qa ser definido. O mesmo ocorre em t4, Quando R vira 1 isso implica Qa a ser 0 que ai irá fazer com que Qb seja 1, neste caso.
Portanto, pode-se perceber a existencia desse delay e isso se torna um problema quando vamos do instante 1 1 para o 0 0 (duas mudanças acontecem ao mesmo tempo), isso, pois, ambos vao mudar de valor, mas por causa desse delay, você nao sabe quem ira mudar primeiro, fazendo com que o resultado do circuito seja indeterminado. Mesmo se as mudanças fossem instantâneas isso faria com que Qa = Qb = 1 o que faria com que Qa = Qb = 0 assim entrando em um looping de indefinição.existencia

Mudanças de 0 0 para 1 1 ou 1 1 para 0 0 são inseguras!
O estado 1 1 é inseguro

Gated SR Latch

O estado do Latch anterior sempre vai mudar quando as mudanças nos sinais acontecerem. Mas se nao tivermos controle dessas mudanças, não saberemos quando o Latch irá mudar de estado.

A seguir terá um latch com um controle (Clock), no qual só irá mudar de estado quando clock for 1. Se não sempre manterá seu estado anterior.
Agora o circuito virou somente Q e -Q, invés de Qa e Qb, isso, pois, como o caso 1 1 1 sempre deve ser evitado, o circuito irá sempre se comportar dessa forma, tendo um sendo o negado do outro.
E é considerado que quando Set é 1 Q é 1 e quando Reset é 1 Q = 0.

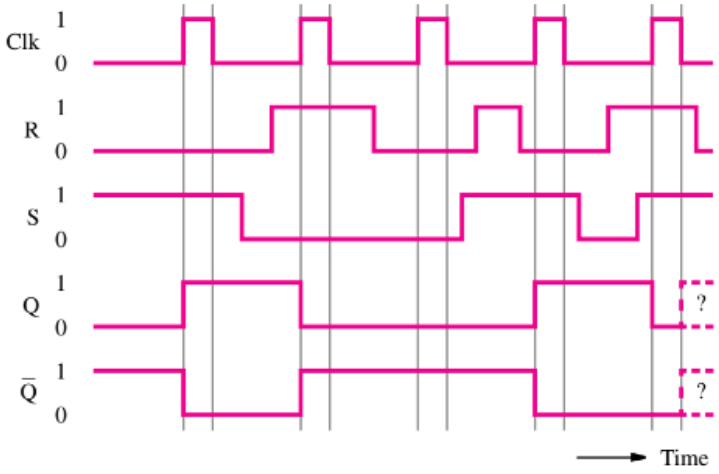


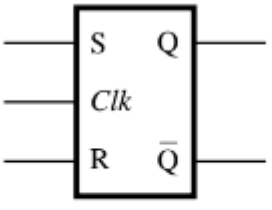
(a) Circuit

Clk	S	R	Q(t + 1)
0	x	x	Q(t) (no change)
1	0	0	Q(t) (no change)
1	0	1	0
1	1	0	1
1	1	1	x

(b) Characteristic table

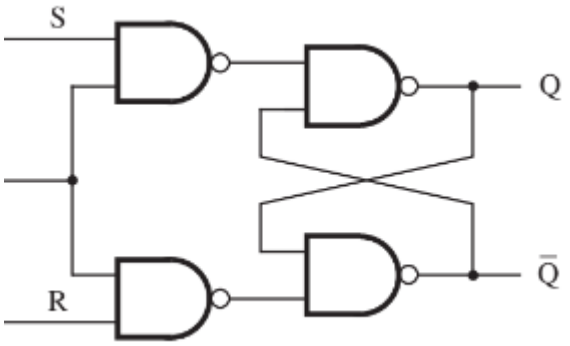
X singnifica que não importa se é 0 ou 1 para a primeira linha, e na última linha ele é X para mostrar que o estado é indefinido.





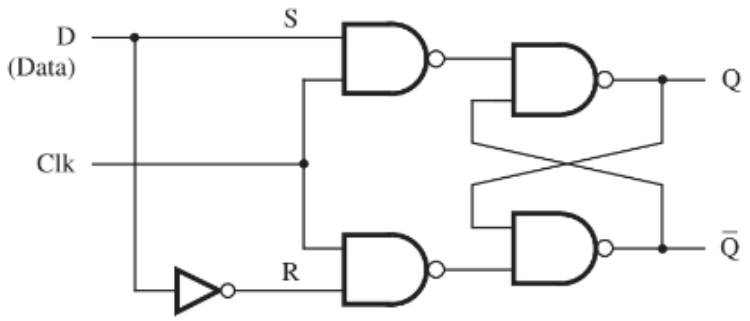
(d) Graphical symbol

Circutio com NAND:



Gated D Latch

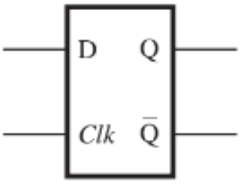
Imagina que queremos fazer uma soma usando o ripple-carry e também desejamos armazenar o resultado dessa soma. Utilizar um simples Latch seria um problema. Quando o ripple-carry está fazendo as suas operações, suas outpus ficam mudando constantemente por causa do atraso das das portas lógicas, sendo assim, seus otuputs vao ficar alternando entre 0 e 1. Portanto, no caso em que o bit é temporariamente 1, logo, 1 é passado para o Set do latch, porém, se, no final, esse bit mudar para 0, 0 será mandado para o Set, o que fará com que o Latch fique com o estado anterior invés de atualizar seu valor.
Isso foi resolvido com o clock. Ele é 0 até terminar a operação da soma, ai irá virar 1 para armazenar o valor e depois zero novamente. Entretanto, antes de adicionar um novo valor no Latch teriamos que reseta-lo para apagar o que ele tinha anteriormente. E esse problema é corrigido com o Latch tipo D.



(a) Circuit

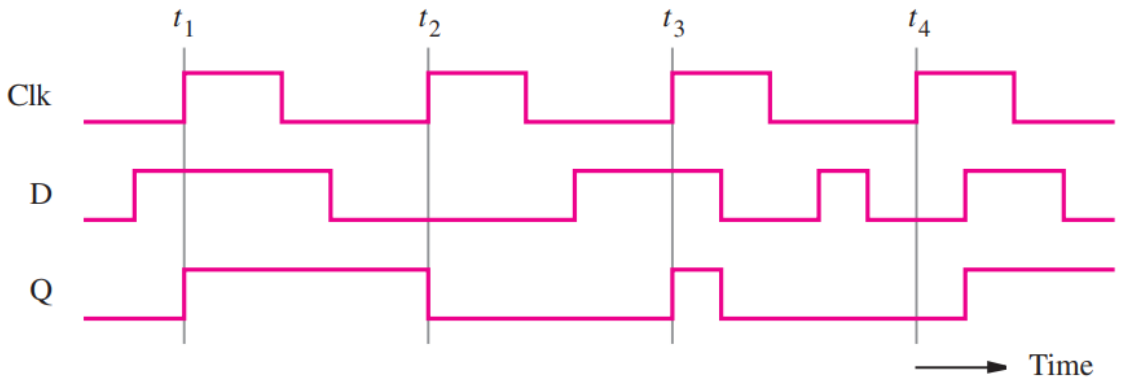
Clk	D	Q(t + 1)
0	x	Q(t)
1	0	0
1	1	1

(b) Characteristic table



(c) Graphical symbol

Neste Latch só existe o input D, considerando que o clock é 1, se D for 1 ele irá forçar S ser 1 e R ser 0 o que irá forçar Q ser 1. Se D for 0 ele irá forçar S ser 0 e R ser 1 o que irá fazer Q ser zero.
Note que nesse tipo de Latch, R é sempre o oposto do S o que evita o caso 1 1 que causava indeterminação.



(d) Timing diagram

O diagrama de tempo mostra que quando Clock for 0, não importa o valor de D, o Q sempre vai se manter igual, entretanto, quando o Clock for 1 Q = D.

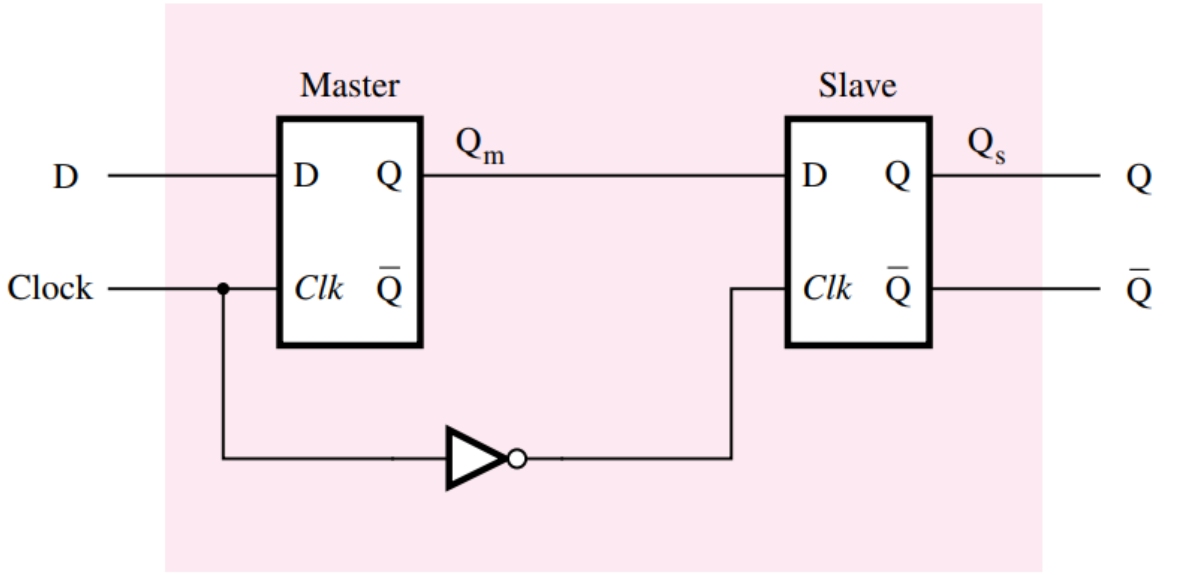
É dito que o Latch é level sensitive, ou seja, ele é sensível ao sinal do clock.

Tempo de Setup e Hold

Se o D variar no exato momento que o clock muda de valor e logo após variar novamente, não tem como se ter certeza de qual será o valor do Q, pois, conforme o Clock altera o valor o D também está alterando. (isso faz mais sentido em flip-flop que é nas bordas, mas tudo bem né).

Com isso existe um tempo de setup, que é quanto tempo antes do clock mudar de valor o D já tem que estar com o seu valor estável e o tempo de Hold que é quanto tempo depois do clock o D deve manter o seu valor.

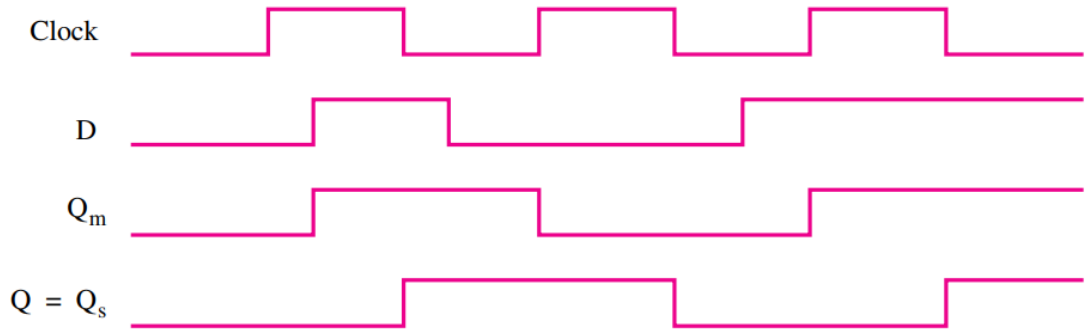
Master-Slave D Flip-Flop



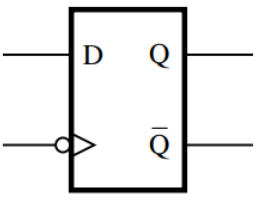
(a) Circuit

O primeiro, Master, muda seu estaado quando o Clock é 1, já o Slave muda quando o clock for 0.
Portanto, quando o Clock for 1, Qm será igual a D. Logo após o clock ir para 0, Qm não vai mais mudar de valor e Qs será agora igual a Qm.

Essa situação, do master só alterar quand o clock for 1, fará com que na hora da subida do clock ele comesse a ler o valor do D, mas não irá alterar nada, enquanto, na hora que o clock descer para 0, ele irá ignorar o valor do D e atuailizará o valor do Q baseado do Qm.



(b) Timing diagram

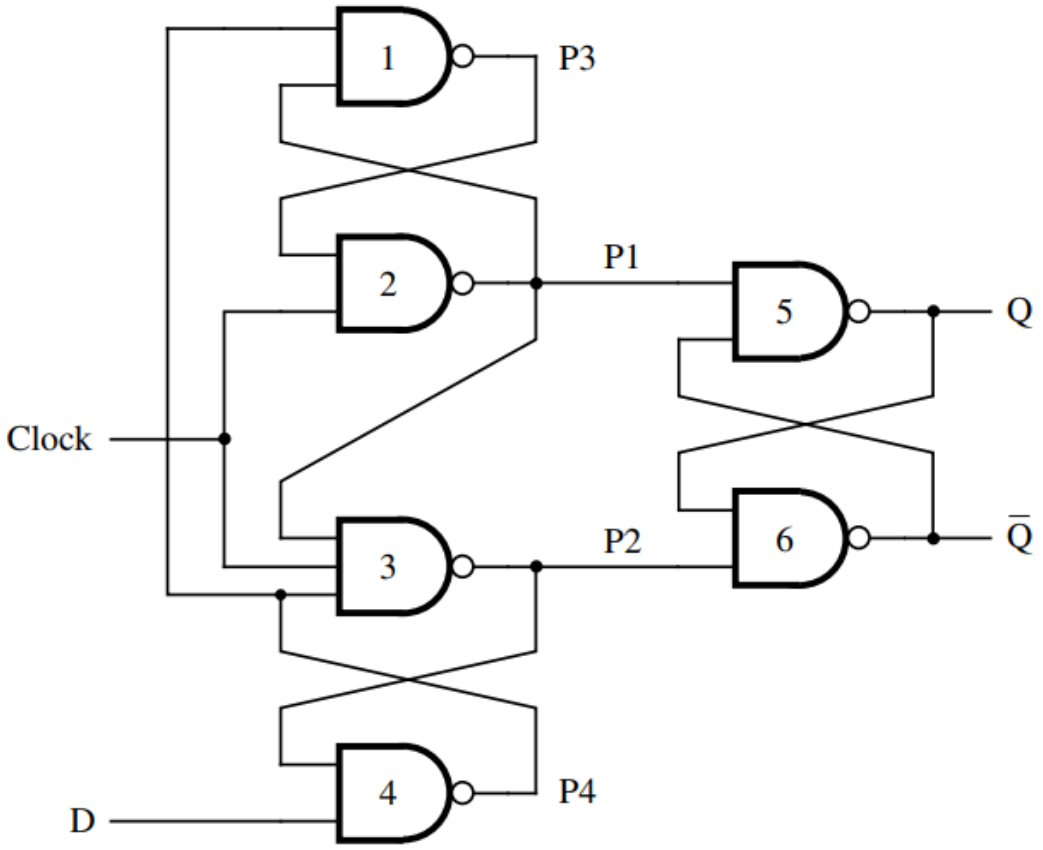


Esse será um flip-flop tipo D com borda de descida (bolinha de NOT no clock)

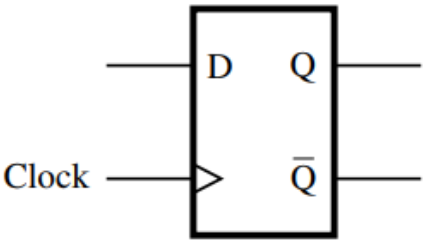
O fato de agora se ter um > invés de um CLK é para mostrar que a alteração é feita nas bordas.

Para ser um flip-flop tido D com borda de subida deve-se fazer com que o Master atualize com clock = 0 e o Slave com clock = 1.

Outros Flip-Flop tipo D



(a) Circuit



Podemos ver que ele só depende de 6 NANDS, enquanto um Master-Slave precisa de 2 Gated D latch, cada um com 4 NANDS, ficando assim com 8 NANDS.

Clock 0:

- P1 = P2 = 1
    - Mantem o estado anterior do circuito
- P4 = -(1.D) = -D
- P3 = -(1.-D) = D

Mudança de Clock para 1:

- P1 vai depender de p3 (anterior) TEMPO DE SETUP
    - p3 = D
    - P1 = -p3 = -D
- P2 vai depender de p4 (anterior) TEMPO DE SETUP
    - p4 = -D
    - P2 = -(P1.1.p4)
    - P2 = -(-D . -D) = D
- Com isso, Q = D
- P4 = -(D.P2) = -D
- P3 = -(P4.P1) = -(-D.-D) = D

A mudança do clock não altera o valor de P4 e P3

Mudança de D para -D:

- P1 = p1 = -D
- P2 = p2 = D
- P4 = -(-D.P2) = 1
- P3 = -(1.P1) = D
- P1 = -(P3) = -D
- P2 = -(P1.P4) = D

A mudança do input D não altera o valor de P1 e P2, por isso que a mudança de Q só ocorre na borda de subida, pois se o clock se manter em 1 e você alterar o valor do D isso não irá influenciar em nada.

Tempo de setup:

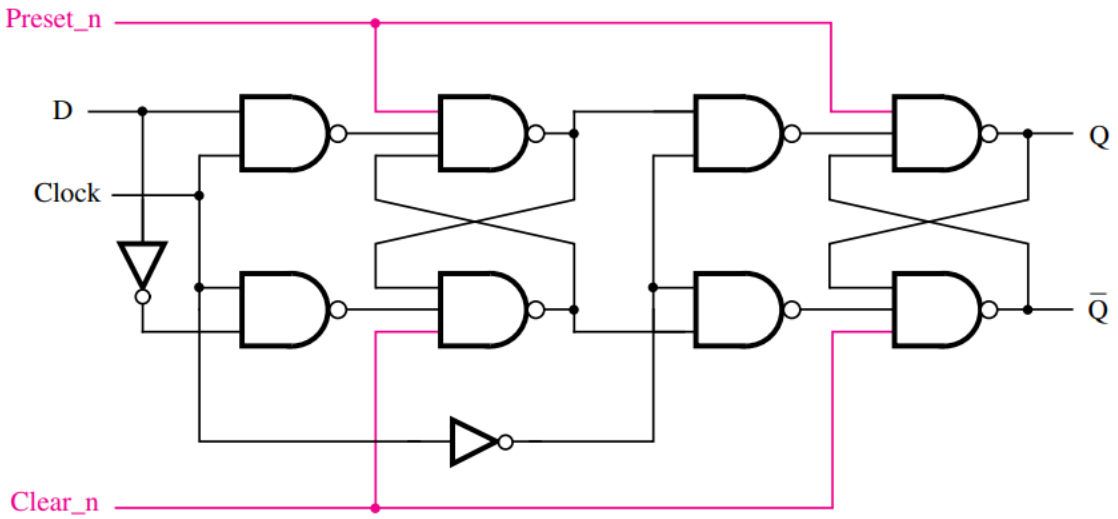
- Quando se faz a altreação do clock de 0 para 1, P1 e P2 vão depender de P3 e P4, do estado anterior, porém, no estado anterior P3 é D e P4 e -D, portanto, se vc mudar o valor de D e logo depois fazer a mudança no clock, antes que de tempo de o sinal passar pelos gastes 1 e 2 vc vai se foder.
- Com isso, podemos concluir que o tempo de Setup é dado pelos gates 1 e 4.

Tempo de hold:

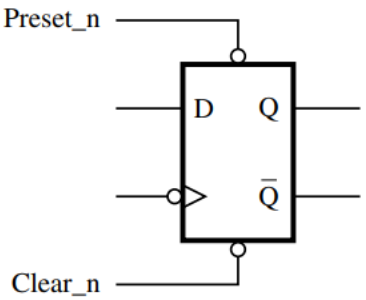
- É dado no gate 3, pois, quando o P2 é estável mudanças no D não importam mais

Preset e Celar

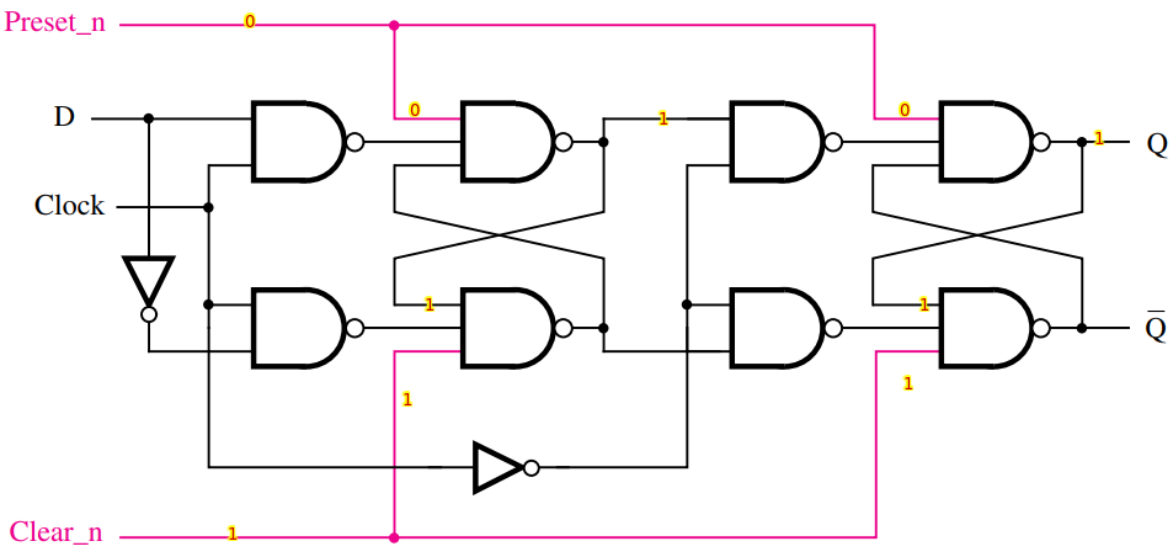
Master e Slave Flip-Flop



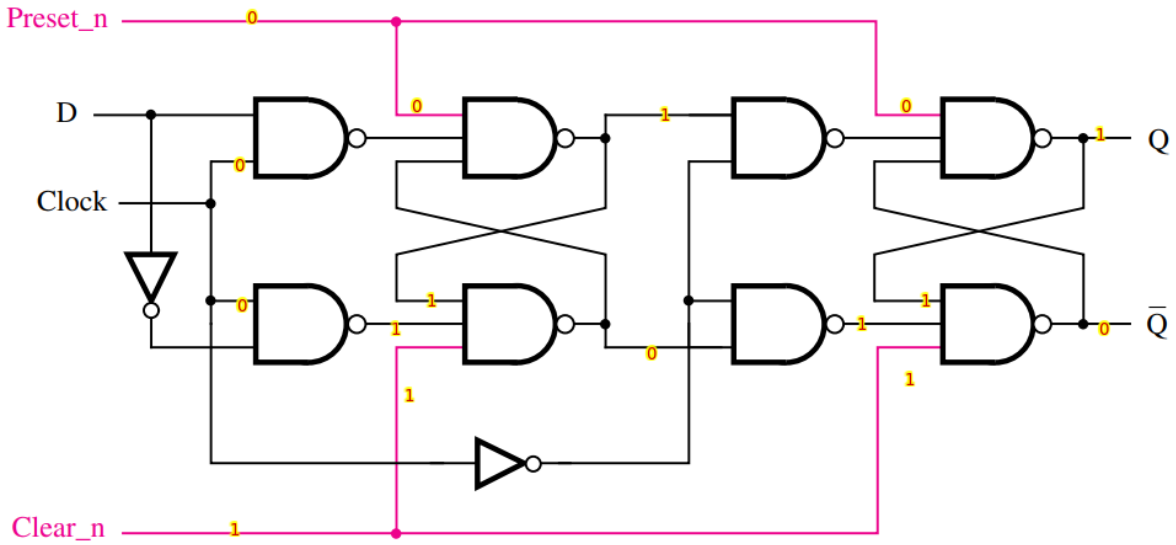
(a) Circuit



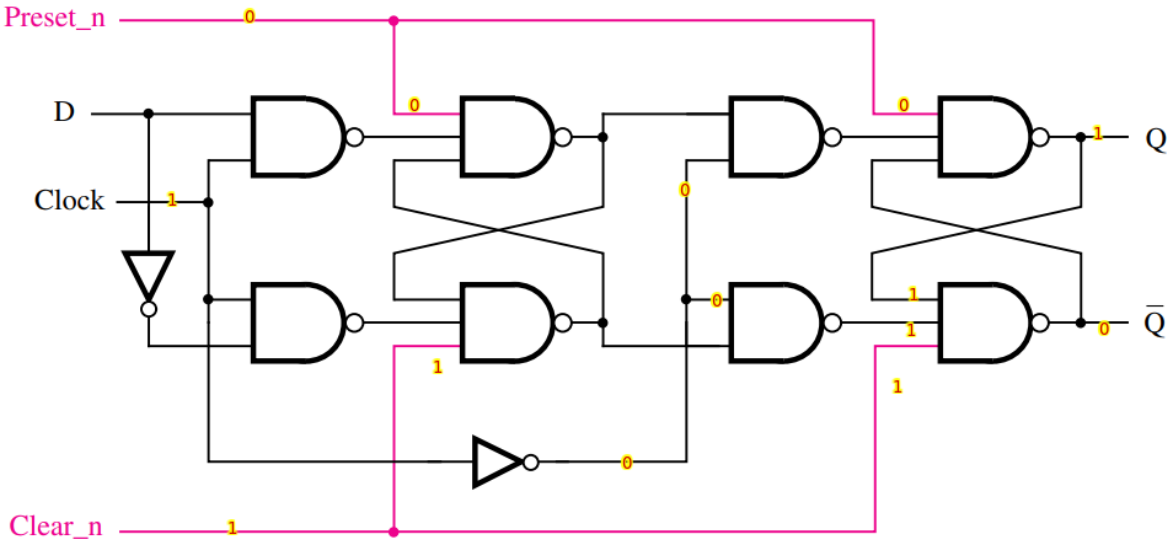
Esse flip-flop tipo D na verdade é a junção de um Gated D Latch e um Gated SR Latch.



Ao ver isso, se pode ter a impressao que ele irá depender do Clock e do D, mas isso não é verdade. Tomamos por exemplo o Clock como 0:



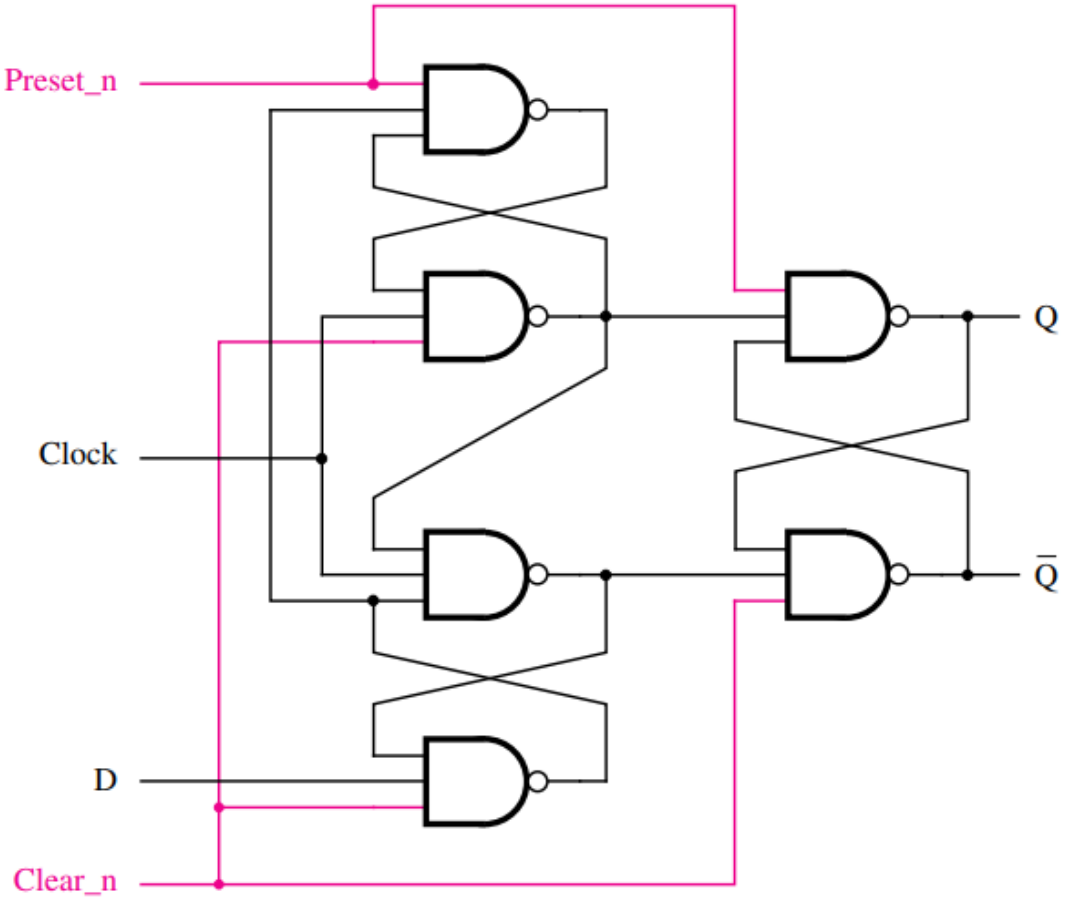
E agora o Clock como 1:

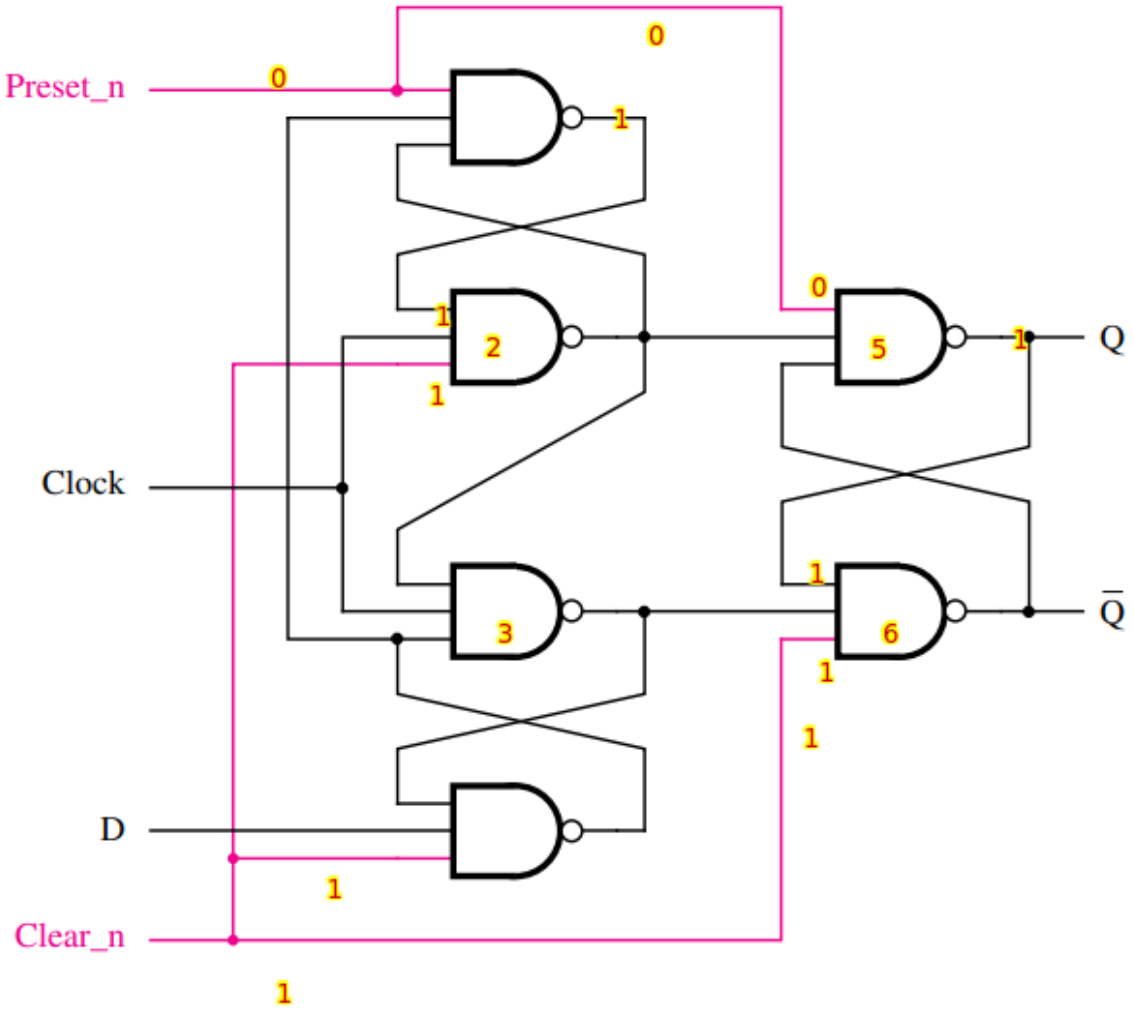


Por causa dessa implementação, de o Master receber o Clock e o slave receber Not Clock faz com que isso sempre irá funcionar.

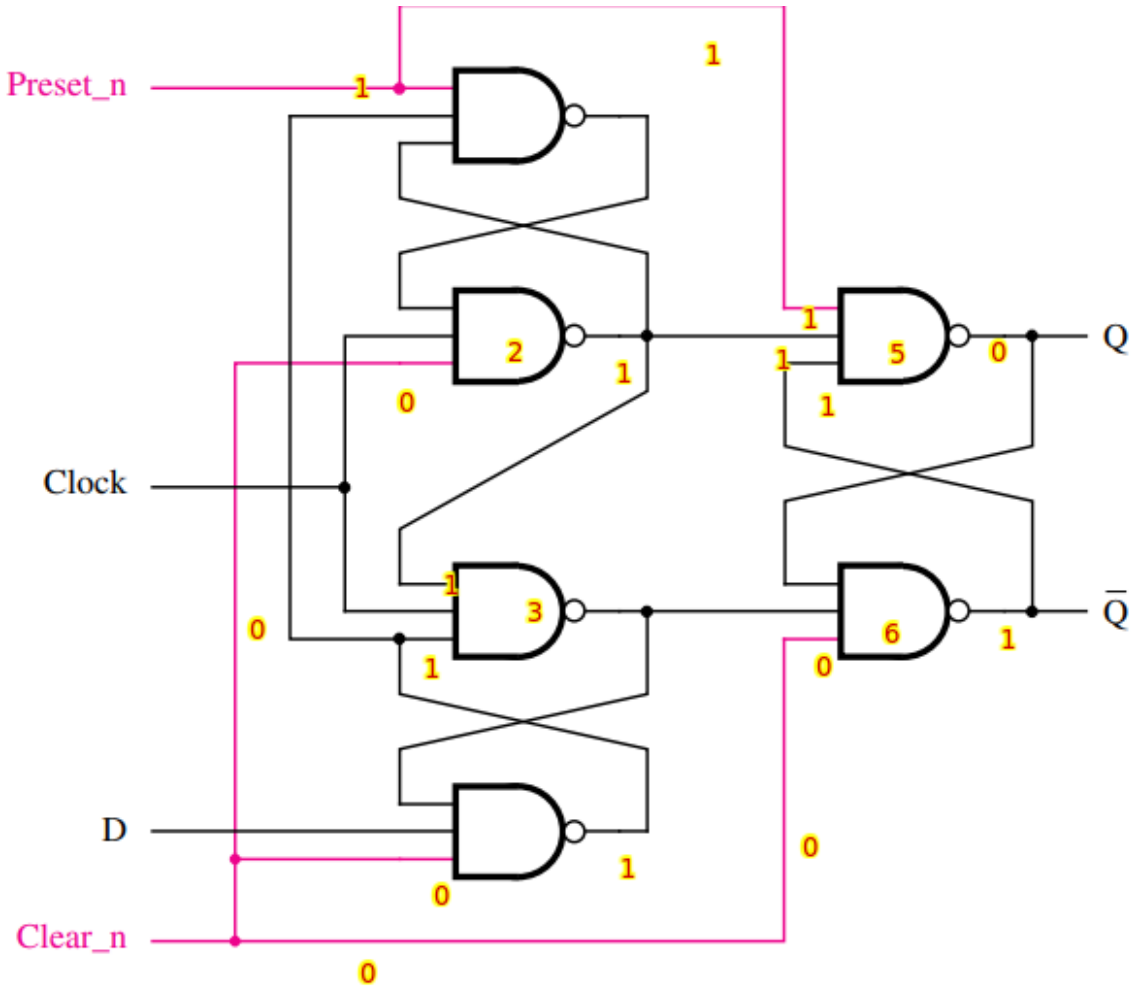
E a mesma lógica segue para o Clear.

O outro Flip-Flop tipo D:

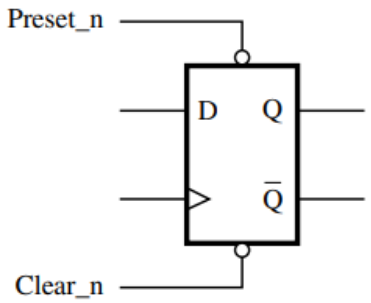




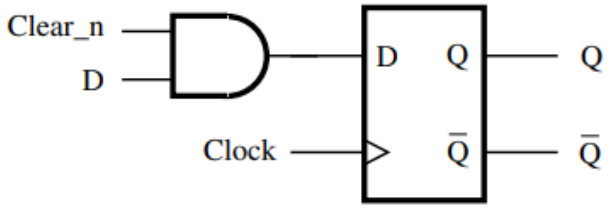
Analisando para o Preset, podemos ver que a mágica está no gates 2 e 3. Se o clock for 1, o gate 2 irá produzir 0 que então será passado para o gate 3, que produzirá 1 que irá para o gate 6 dando o output que queremos. Quando o clock for 0, o 0 irá direto para o gate 3 que produzirá 1 para para o 6.



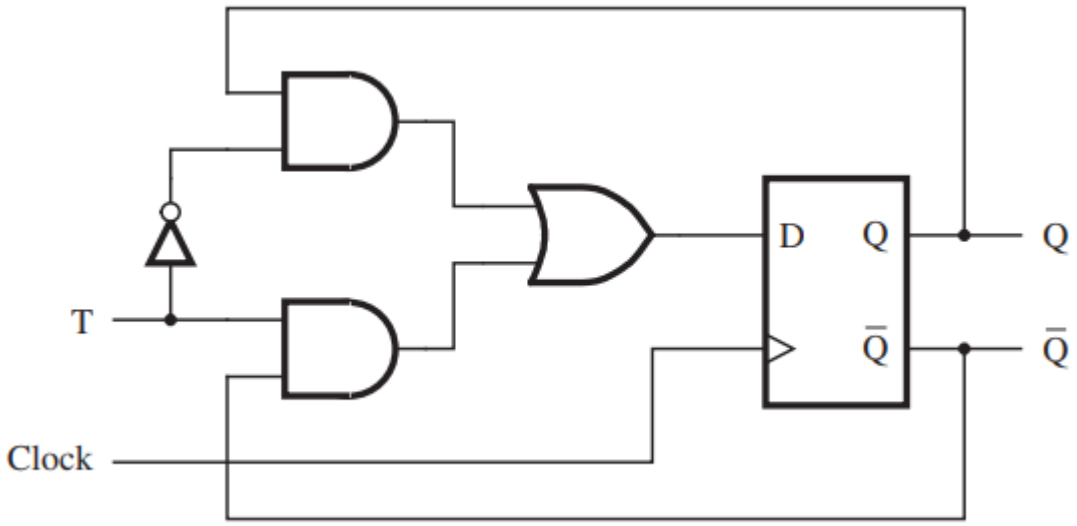
Para o Clear, ele sempre irá produzir independendo



O preset e o Clear são assincronos, não depende do clock. Entretanto, as vezes é necessário te-los de uma maneira sincrona, que pode ser representada da seguinte forma:



Flip-Flop tipo T



O circuito faz com que o D receba ou Q ou -Q dependendo do T.

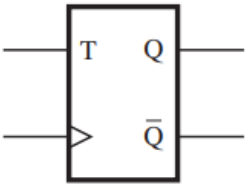
T = 0, então D = Q, logo o estado do sistema não muda.
T = 1, entao D = -Q, logo o estado do sistema muda.

Portanto, o Flip-Flop tipo T mantem o estado quando T for 0 e muda o estado quando T for 1.

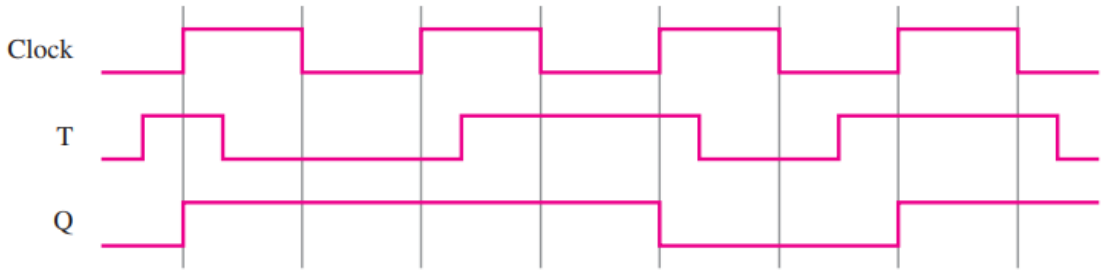
D será dado por: -TQ + T-Q

T	Q(t + 1)
0	Q(t)
1	Q̄(t)

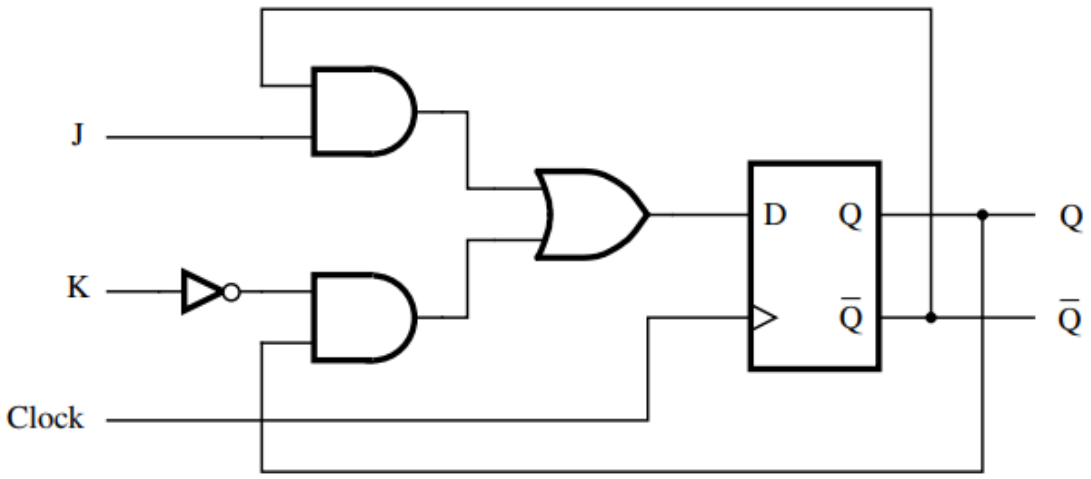
(b) Characteristic table



(c) Graphical symbol



Flip-Flop JK



(a) Circuit

É praticamente o mesmo circuito do Flip-Flop tido T, entretanto, o D agora não depende somente do T, também dependendo do J e do K.

O D será dado por: $D = J\overline{Q} + \overline{K}Q$

Note que a mudança não ocorre somente na subistituição do T pelo J e o K, mas também pelas mudanças de onde se pega o Q.

Antes, o Q ia com o -T e o -Q ia com o T. A lógica agora será a mesma, o Q terá que estar multiplicando algo negado e -Q terá que estar multiplicando algo não negado. Por isso que será J-Q + -KQ.

O flip-flop JK funcionará tipo o SR, onde J = S e K = R menos no caso J = K = 1, pois neste caso, o Flip-Flop vai funcionar exatamente como o Flip-Flop tipo T.

O Flip-Flop JK é um modelo versátil, pois ele pode funcionar como um Flip-Flop tido D e também como um Flip-Flop tipo T (J = K).

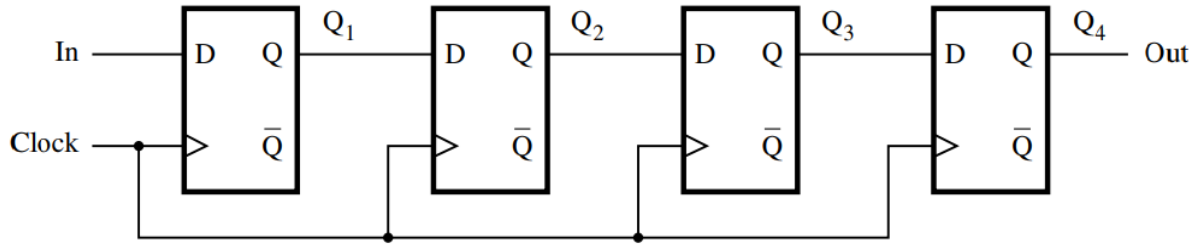
Quando J = K = 0 ele funciona exatamento como um tipo T quando T é 0, ou seja, ele mantem o estado dele. Já quando J = K = 1 ele funciona com o T sendo 1, mudando o estado.

Quando J = 1 ele é um Set, fazendo o Q ir para 1.
Quando J = 0 ele é um Reset, fazendo Q ir para 0.

Registrador

É uma junção de Flip-Flops para poder armazenar mais de 1 bit.

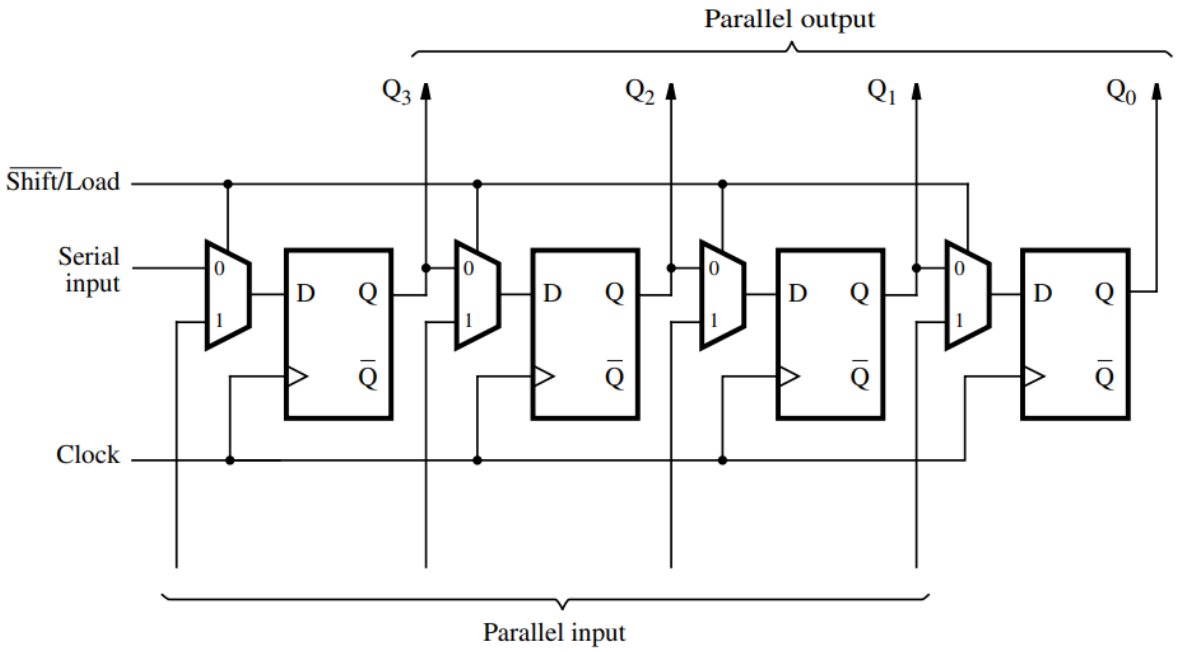
Registrador Shifter (serial):



Como a próxima entrada recebe a saida do anterior, podemos ver como isso será um shifter, pois, a cada ciclo de block, o próximo Flip-Flop irá receber o que tinha previamente no Flip-Flop anterior.

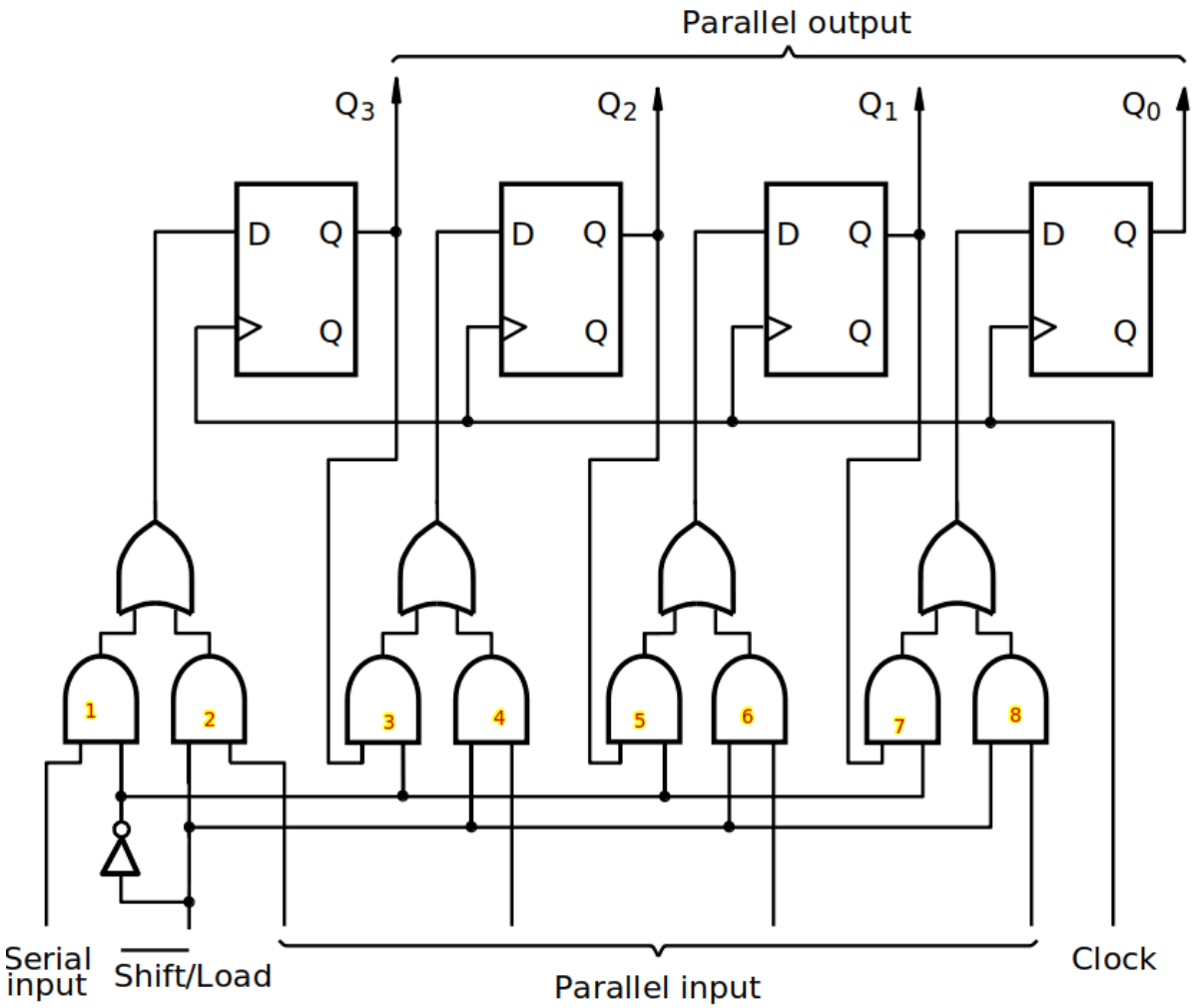
	In	Q_1	Q_2	Q_3	Q_4 = Out
t_0	1	0	0	0	0
t_1	0	1	0	0	0
t_2	1	0	1	0	0
t_3	1	1	0	1	0
t_4	1	1	1	0	1
t_5	0	1	1	1	0
t_6	0	0	1	1	1
t_7	0	0	0	1	1

Registrador Shifter (paralelo):



Como se pode perceber, esse circuito funciona como um shifter, mas, no meio dele, você pode ir e alterar o que está em cada Flip-Flop dando um load.

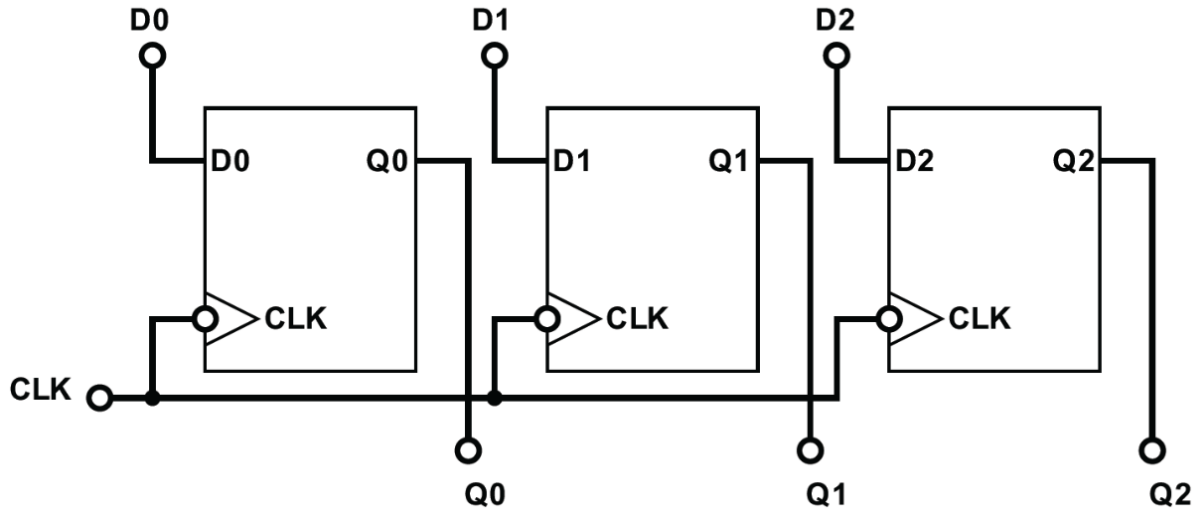
A entrada de cada Flip-Flop vem de um Mux, no qual, a chave é o -Shift/Load, se a chave for 0 ele irá funcionar exatamente como o circuito anterior, entretanto, se a chave for 1, 4 inputs Paralelos serão passados para o Registrador.



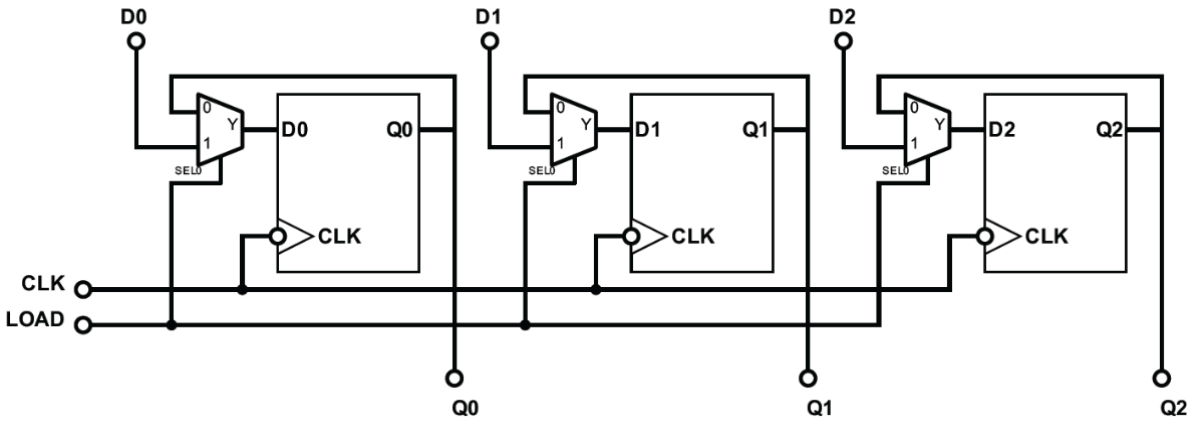
É o mesmo circuito, porem mais complexo e da forma que o vanderlei passou kkkk.

Quando o -Shift/Load é 0 todos os ANDS impares são acionados, consequentemente, o que irá para os inputs Ds dos Flip-Flops será os Qs, ou seja, quando -Shift/Load é 0 ele irá funcionar como um Shifter.
Já, quando -Shift/Load é 1, todos ANDS pares serão acionados, os quais recebem os inputs paralelos, portanto os Flip-Flops (quando vier o sinal de clock) serão atualizados para os novos inputs passados.

Registrador Paralelo:

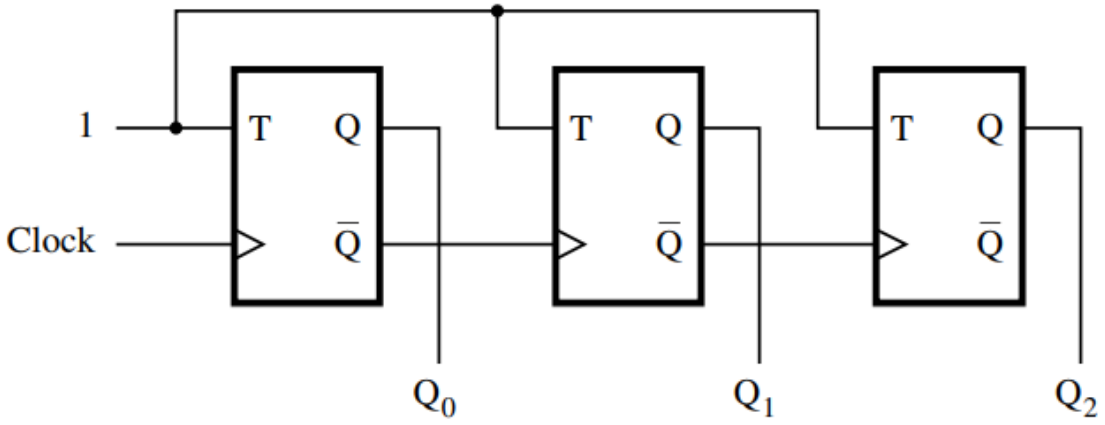


Esse registrador não é um shifter, e sim, somente um registrador normal. Ou seja, durante um ciclo de clock ele estará guardando as informações presentes nos Flip-Flops, entretanto, logo quando uma nova borda (no caso da imagem, uma de descida) vem ele irá atualizar o seu valor.
Portanto, para resolver este problema, precisaremos adicionar um novo input que será o Enable, que irá dizer se devemos ou não atualizar o registrador quando se tem uma nova borda de clock.



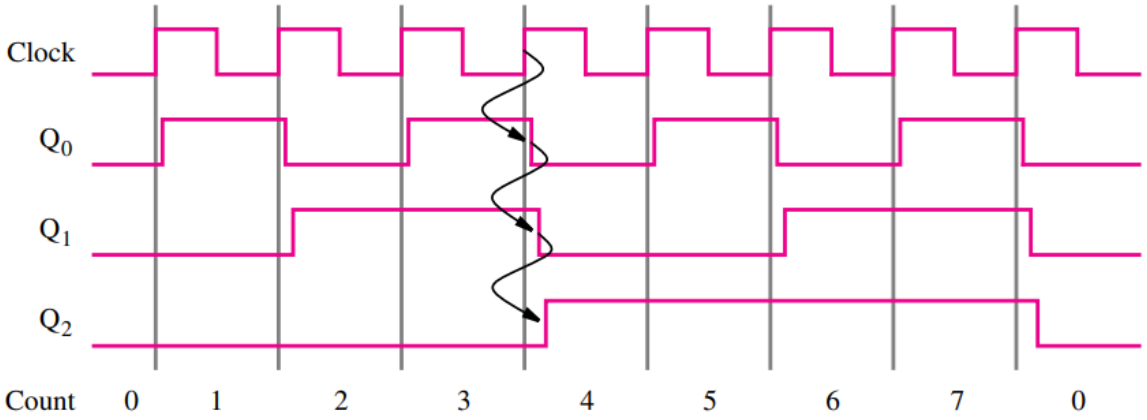
É utilizado este mux tendo o LOAD como chave, pois quando o LOAD é 1 ele irá atuailizar o valor contendo nos inputs na nova borda de clock, entretanto, quando LOAD é 0 o mux irá mandar para o D o valor que se tinha previamente no Q, ou seja, quando vier uma nova borda de clock ele irá atualizar seu valor para o valor que já tinha, ou seja, nada irá mudar.

Contador Assíncrono



Cada input T está ligado a 1, ou seja, na borda de subida esse Flip-Flop terá seu estado alterado.

O primeiro Flip-Flop funcionará normalmente, trocando seu estado a cada borda de clock, entretanto, os seguintes só irão trocar de estado quando o Q do Flip-Flop anterior mudar de 1 para 0.

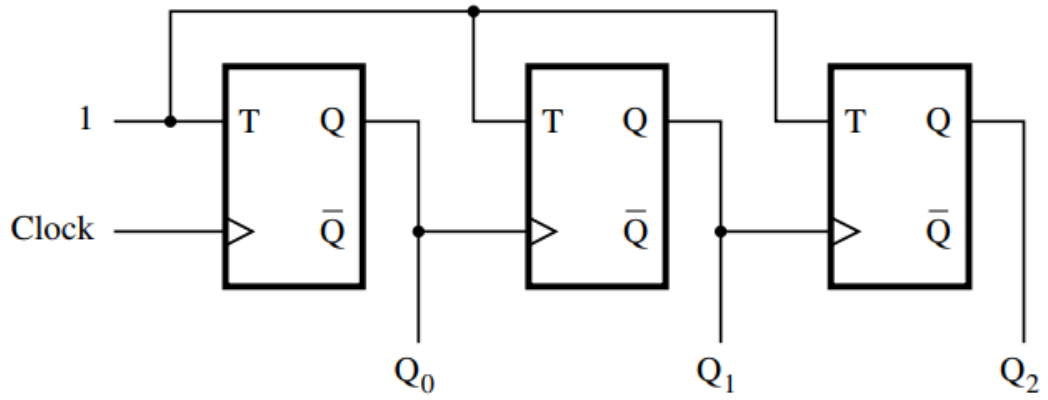


Tempo	Clock	Q0	Q1	Q2
0	0	0	0	0
1	1	1	0	0
2	0	1	0	0
3	Borda de Clock	0	1	0
4	Borda de Clock	1	1	0
5	Borda de Clock	0	0	1

6	Borda de Clock	1	0	1
7	Borda de Clock	0	1	1
8	Borda de Clock	1	1	1
9	Borda de Clock	0	0	0

Como se pode ver, a cada Borda de Clock o valor está aumentando pelo fator de 1.
Como só se tem 3 Flips-Flops ele conseguirá contar até 7, como visto na table,a pois no tempo 9 ele já volta para 0 0 0 e começará tudo novamente.

Esse circuito é conciderado um circuito de módulo 8 (olha a mat discreta ai).



Esse novo circuito é muito parecido com o antigo, entretanto, o clock para os demias Flips-Flops são passados pelo Q, e não pelo -Q. Portanto, um Flip-Flop, menos o primeiro, só teŕa seu estado alterado quando o seu antecessor passar de 0 para 1.
Isso faz com que esse contador tenha uma reposta oposta ao anterior, sendo da seguinte forma: 0, 7, 6, 5, 4, 3, 2, 1, 0, 7, 6 …

Contadores Síncronos

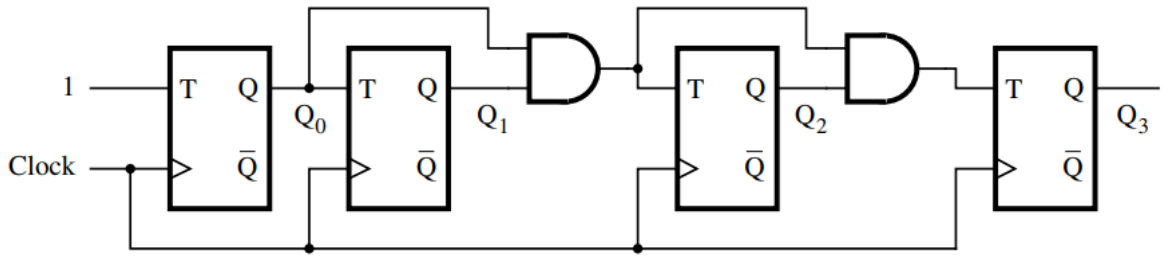
Os contadores assincronos são mt lentos, pois para um grande número de flips-flops o delay vai ficar gigante. Por isso os contadores síncronos são mais rápidos.

Table 5.1 Derivation of the synchronous up-counter.

Clock cycle	Q_2 Q_1 Q_0	
0	0 0 0	
1	0 0 1	
2	0 1 0	← Q_1 changes
3	0 1 1	
4	1 0 0	← Q_1 changes, Q_2 changes
5	1 0 1	
6	1 1 0	← Q_1 changes
7	1 1 1	
8	0 0 0	← Q_1 changes, Q_2 changes

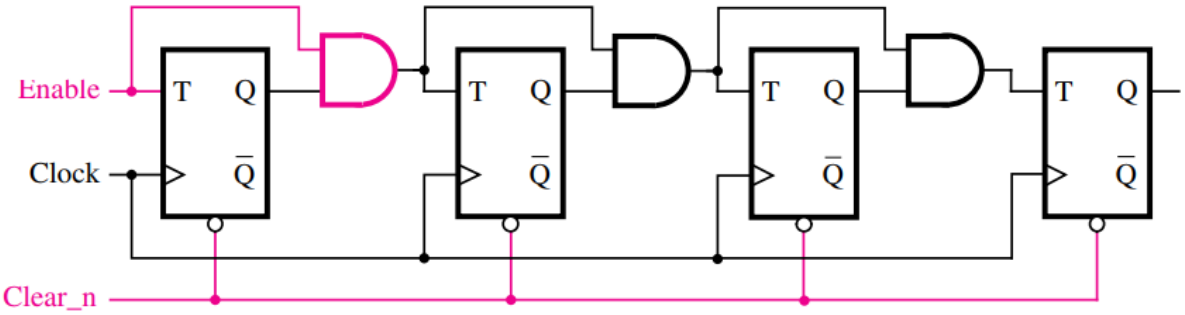
Analisando a Tabela de um contador de módulo 8 podemos perceber que Q0 sempre muda de valor, Q1 só muda de valor quando Q0 = 1 e Q2 só muda quando tanto Q0 quando Q1 mudam de valor. Portanto, utilizando Flips-Flopts tipo T, os inputs seriam dados por:
T0 = 1
T1 = Q0
T2 = Q0Q1

O que nos leva ao seguinte circuito:



Ainda existe um tempo de propagação nos ANDS, esse tempo não pode ser maior que o Clock, na realidade, não pode ser mais que o Clock menos o tempo de Setup.

Também podemos adicionar um Enable e Clear no circuito.

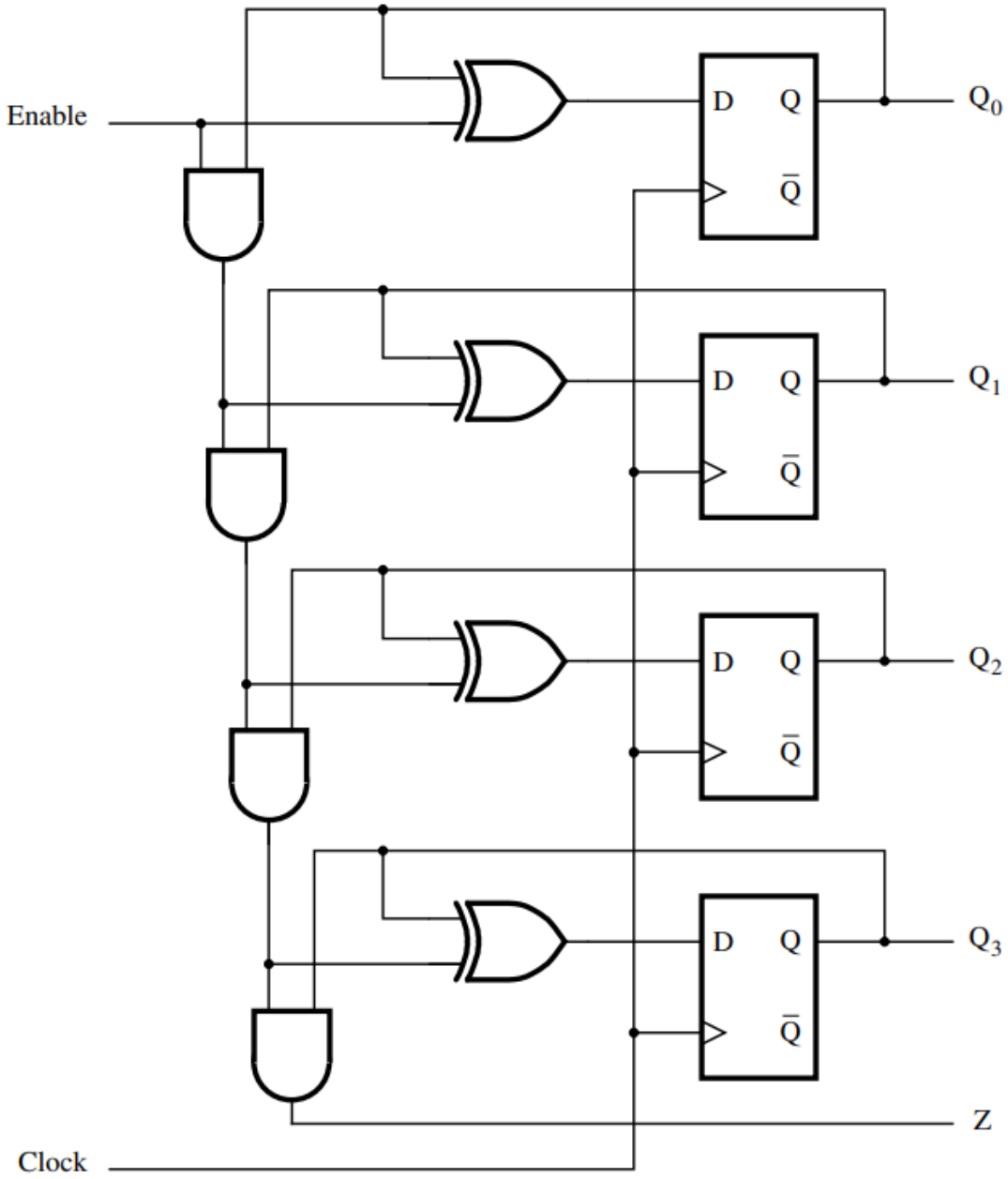


Agora invés de T0 ser simplesmente 1 ele é Enable e, além disso, T1 é dado por um AND entre o Enable o Q0, onde, se Enable for 1 o circuito vai funcionar normalmente, mas se o Enable for 0, esse zero será propagado por todos os ANDs o que irá ocasionar em todos os Flips-Flops terem o T como 0 que fará com que o circuito nunca mude de estado.

Esse AND no T0 entre o Enable e o Q0 deve existir, pois se o Enable for 0 o Flip-Flop irá manter o seu estado anterior, o que fará com que Q0 possa ser 1, propagando 1 pelo resto do circuito que é algo que não queremos. Logo, este AND cuidará disso.

E o clear serve para poder colocar todos os Flips-Flops como zero.

Contadores com Flip-Flop tipo D



Se assumirmos que Enable é 1, os Ds serão dados dessa forma:

$$D_0 = Q_0 \oplus 1 = \overline{Q}_0$$

$$D_1 = Q_1 \oplus Q_0$$

$$D_2 = Q_2 \oplus Q_1 Q_0$$

$$D_3 = Q_3 \oplus Q_2 Q_1 Q_0$$

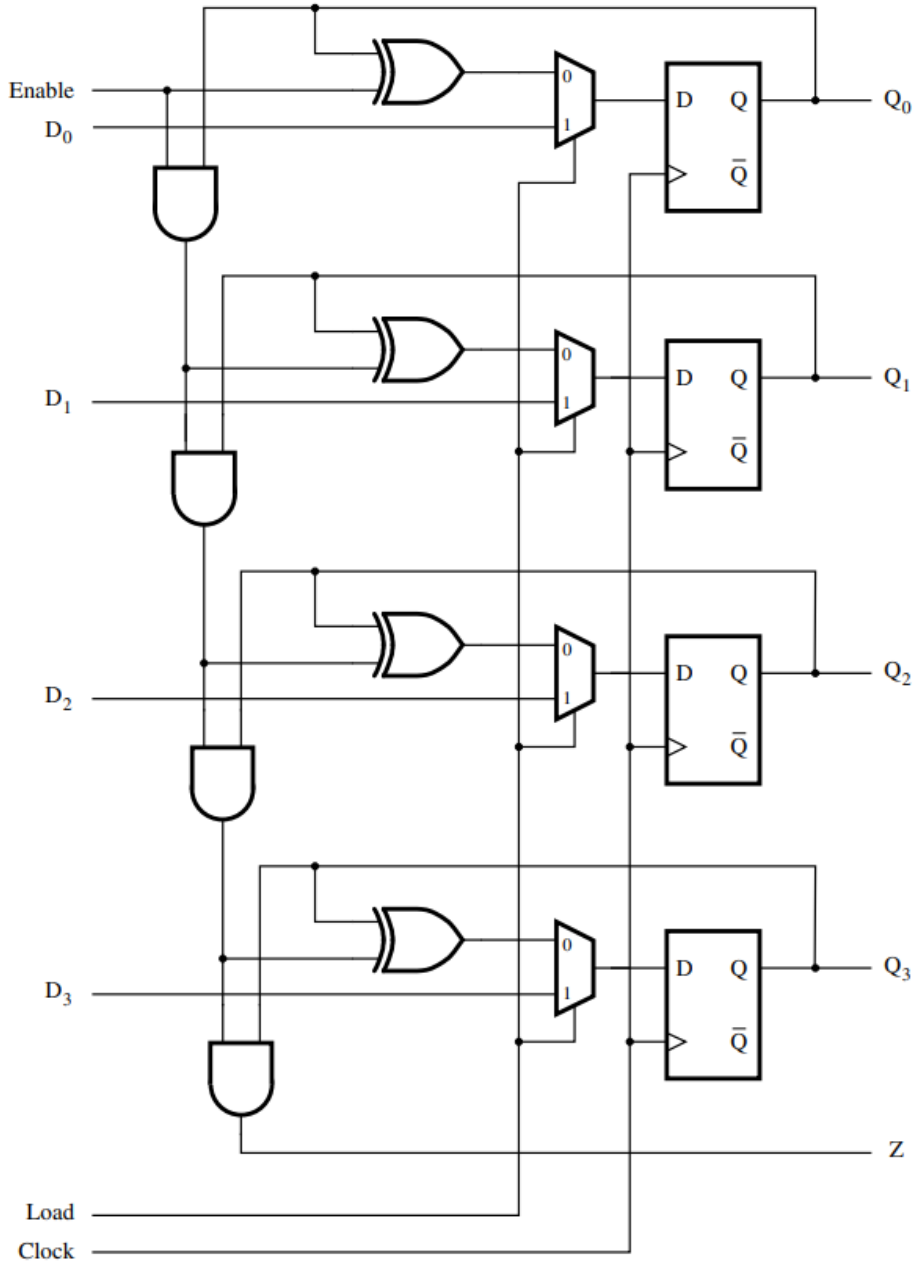
Portanto, a fórmula para o D é dada da seguinte maneira:

$$D_i = Q_i \oplus Q_{i-1} Q_{i-2} \cdots Q_1 Q_0$$

Essa fórmula para o D vem do fato de que, quando analisamos a tabela verdade para o contador, vimos que o um Q só muda se todos os Q anteriores forem 1, logo, se o AND entre todos os Qs forem 1, isso signifca que o Flip-Flop tem que mudar, logo, o xor vai fazer com que o D seja -Q. E se o resultado do AND for 0 isso sinifica que o estado tem que manter o mesmo, sendo assim, será um XOR entre Q e 0 o que resultará em Q e consequentemente o estado se manterá o mesmo.

Existe também um AND adicional que da como output um Z, esse Z é importante, pois ele facilita uma junção com um novo contador e também é útil para saber se o contador chegou no seu valor máximo (todos os Qs sendo um) e virará 0 no próximo ciclo de Clock.

Contadores com Carregamento paralelo

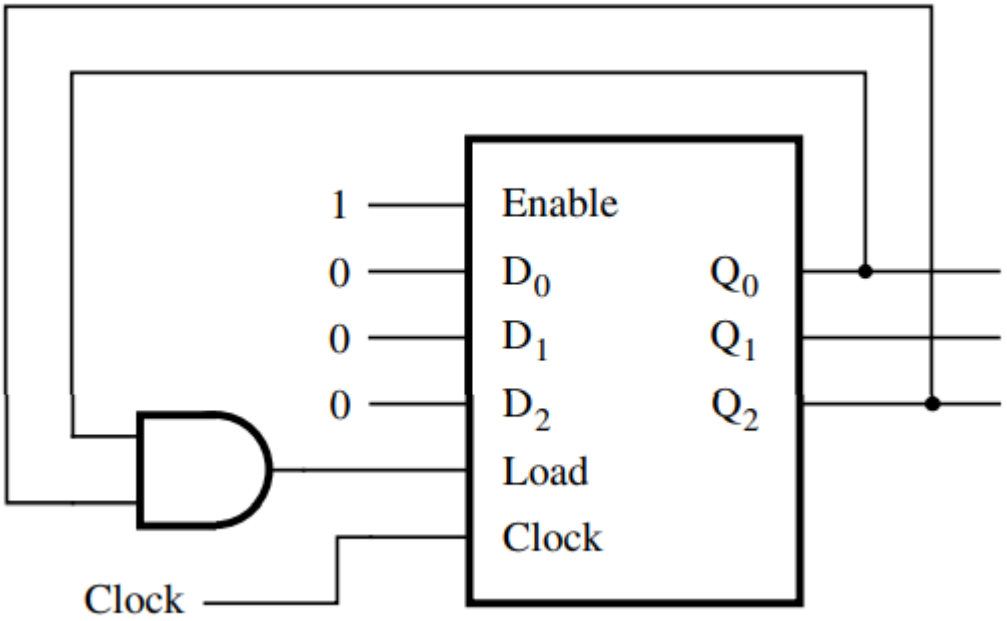


Novamente, quando se fala de carregamentos paralelos a estratégia é a mesma. Será o mesmo circuito, porem com um MUX para escolher entre o circuito normal ou os inputs paralelos.

Contadores fora da base 2

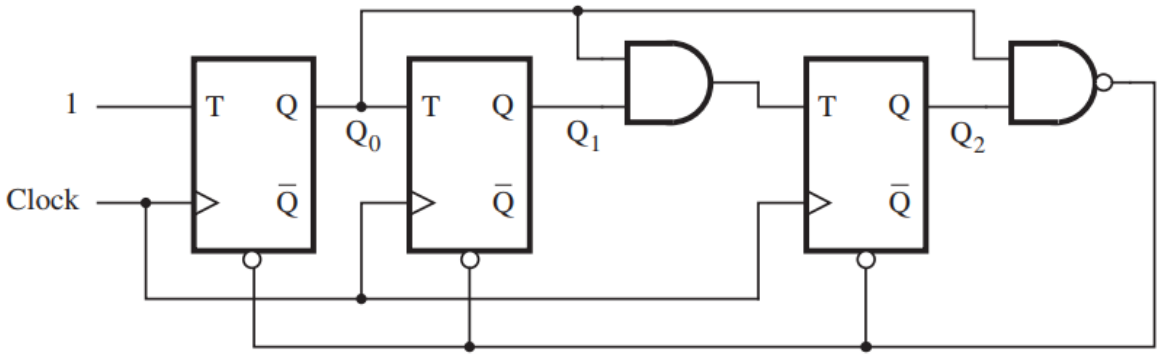
Os contadores sempre contam de 1 em 1, mas acabam em um número de base 2. Mas e se quissemos fazer um contador que fosse mod 5?
O jeito é utilizar o Load de uma forma tal que quando chegue em certo estado você reste todas as operações para 0.

No caso do mod 5, quando Q0 = Q2 = 1 isso implica que já chegou em 5, logo, o contador deve resetar para 0, o que, em um circuito fica dessa forma:



Pelo fato de o reset acontecer na borda do clock, dizemos que esse tipo de contador tem um reset síncrono.

Reset Assícrono



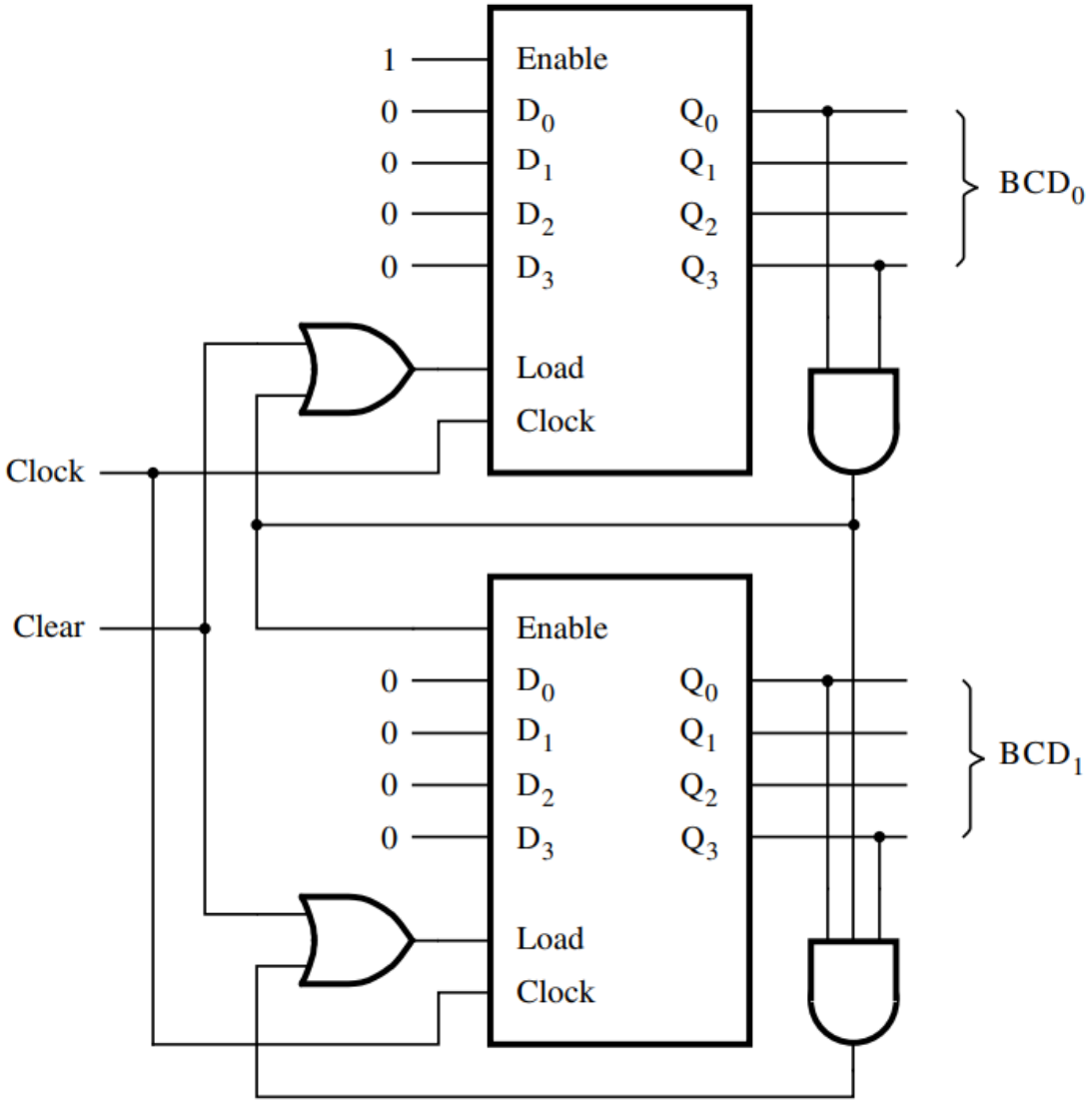
(a) Circuit

Dessa vez, pelo reset ser no clear do Flip-Flop ele é assincrono e funciona da mesma forma do anterior, pegando quando Q0 = Q2 = 1 ele usa um NAND para poder ativar o clear e restar o contador.

Essa ideia talvez não seja muito pertinente, pois se tem um sistema que é totalmente sincronizado, ter um reset que é assincrono pode foder com ele.

E o reset assíncrono só pode colocar todos em 0, já o sícrono pode alterar este valor para um pré determinado.

BCD Counter



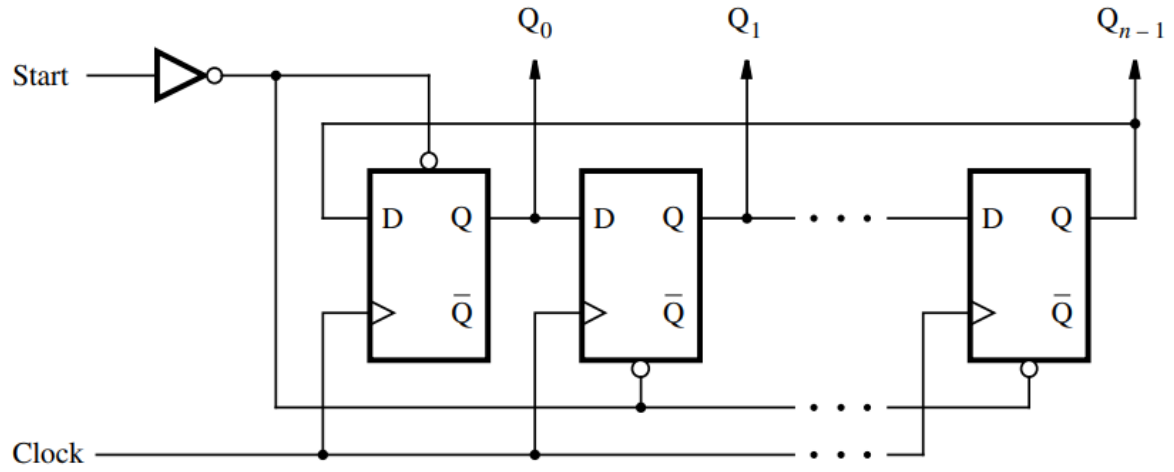
Ele consiste de dois módulos 10.

O segundo só é ativado quando o primeiro chegar em 9.

Note que um sinal Clear, que é ativado quando for 1 é mandado para ambos os contadores, podendo coloca-los como 0 a qualquer instante.

Ring Counter

Nos contadores anteriores, para saber o valor do circuito é necessário detectar qual que é o valor de cada Flip-Flop.
É possível desenvolver um contador tal qual cada Flip-Flop só fique no estado Qi = 1 for exatamente um count, ou seja, em todos os outros counts Qi = 0. Logo, Qi vai indicar diretamente a ocorrencia de um count correspondente. Pelo fato disso não representar números binários, é melhor dizer que os outputs dos Flips-Flops representam um código. Tal circuito pode ser construído de um simples registrador de shift.



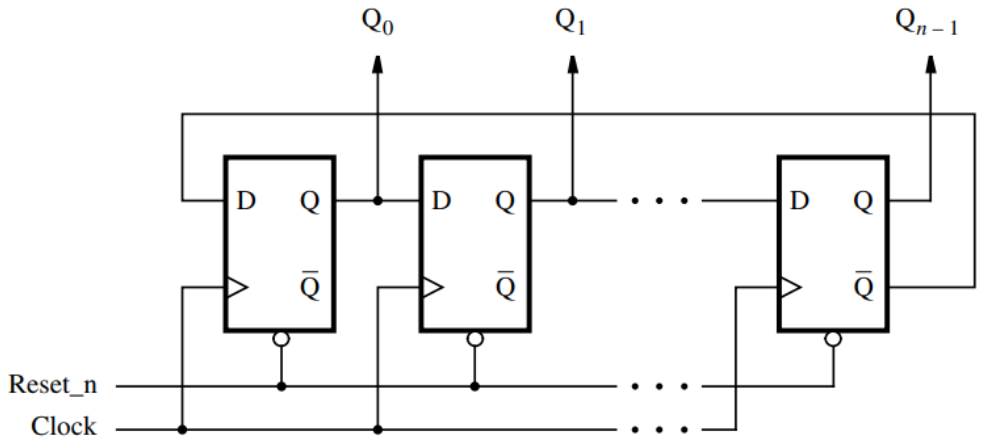
O nome ring vem do fato de que o output do último Q irá para o D do primeiro Flip-Flop.

Quando dado Start, o primeiro Flip-Flop terá valor 1, o que fará com que o Contador fique da seguinte forma:

1000 -> 0100 -> 0010 -> 0001 -> 1000 …
Pode-se perceber que o tamanho da sequência é de n, que nessa caso é 4

Johnson baby Counter

Uma variação do Ring Counter é o Johson Counter. No ring Counter, pegavamos o Q do último Flip-Flop e passávamos para o D do primeiro, enquanto no Joson Counter, invés de passarmos o Q para o D iremos passar o -Q do último Flip-FLop para o D do primeiro.



1000 -> 1100 -> 1110 -> 1111 -> 0111 -> 0011 -> 0001 -> 1000
1 -> 2 -> 7 -> 15 -> 14 -> 12 -> 8 -> 1

Ao contrário do Ring Counter o Johnson Counter tem um tamanho de sequência de 2n, que nesse caso é 8.

Disponível nos próximos capítulos de Vanderlei Bonato

Obrigado aos leitores! Desconsiderem os inumeros erros de portugues, tipo o contido nesta frase. ☃️