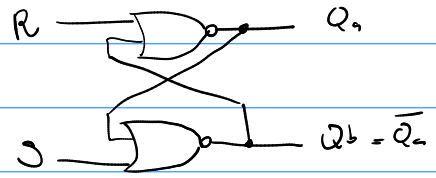
Sistemas Digitais

Simple Latch

Combinacionais vs **Sequenciais**

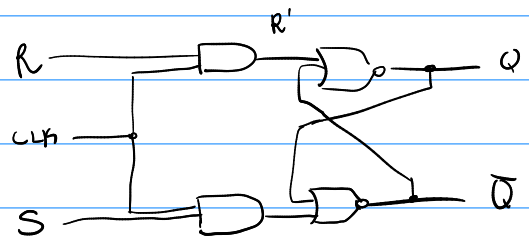
- Saída depende somente das variáveis de entrada
- Saída depende também do estado anterior do circuito
  - Tem "memórias"

Simple Latch



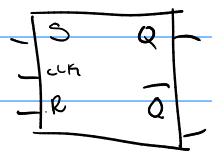
Set	Reset	Qa	Qb	
0	0	Qa	Q̄a	(no change)
0	1	0	1	
1	0	1	0	
1	1	0	0	undefined behavior

Gated SR Latch

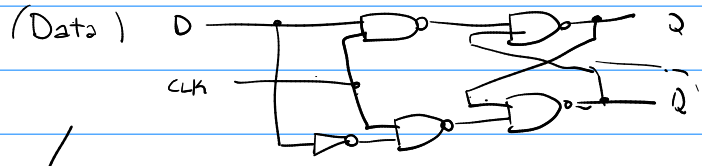


CLK	S	R		
0	x	x	→	S'R' = (0,0), No changes
1	0	0	→	S'R' = (0,0), No changes
1	0	1	→	0
1	1	0	→	1
1	1	1	→	?

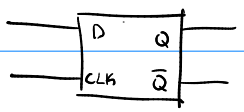
Quando CLK=0, sistema mantém a informação

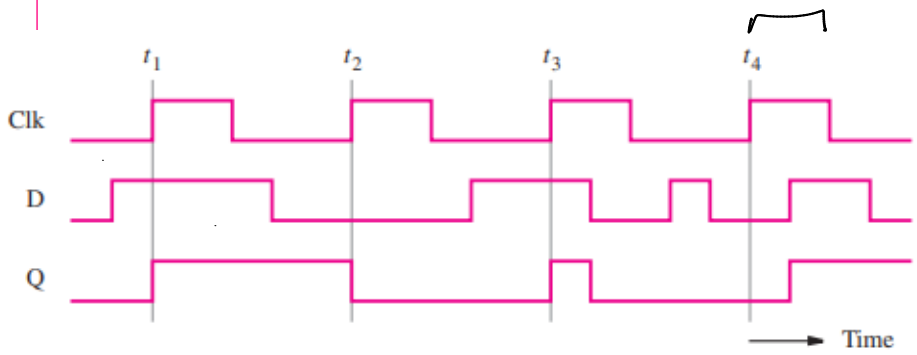


Gated D Latch



CLK	D	Q(t+1)
0	x	Q(t) (Holds)
1	0	0
1	1	1





(d) Timing diagram

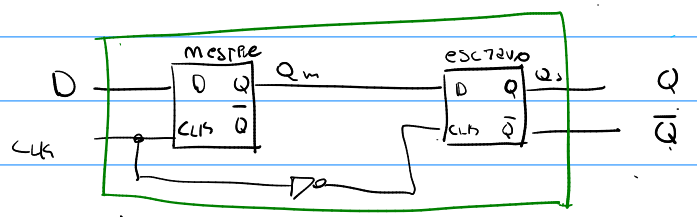
Até o momento, só é estável SE D não mudar no período crítico (mudança de clk para o nível lógico alto)

EDGE-Triggered D Flip Flops

Impedir que o estado mude mais que uma única vez por ciclo de clock → Mudar somente nas bordas

Master-Slave (Mestre Escravo)

Привет я Хосе Как мас сабот?

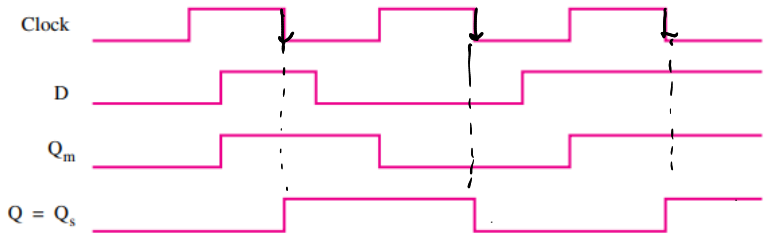


Troca o sinal final somente 1 vez! (Borda de descida)

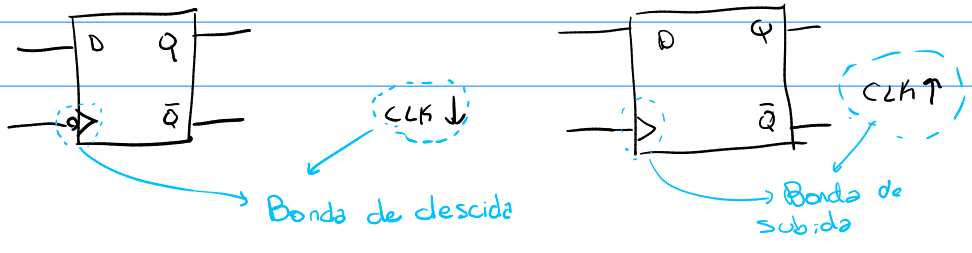
↳ Muda livremente quando CLK = 1

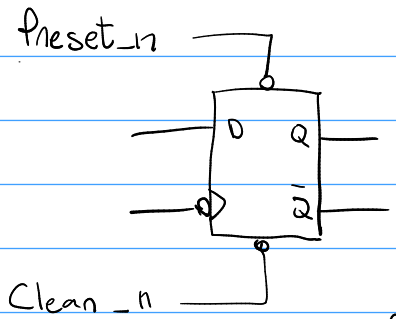
↳ Muda livremente quando o CLK = 0

↳ 1 → 0 = pega o estado do mestre



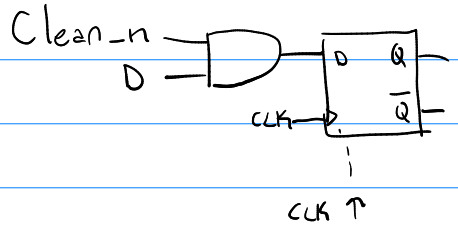
Símbolo →





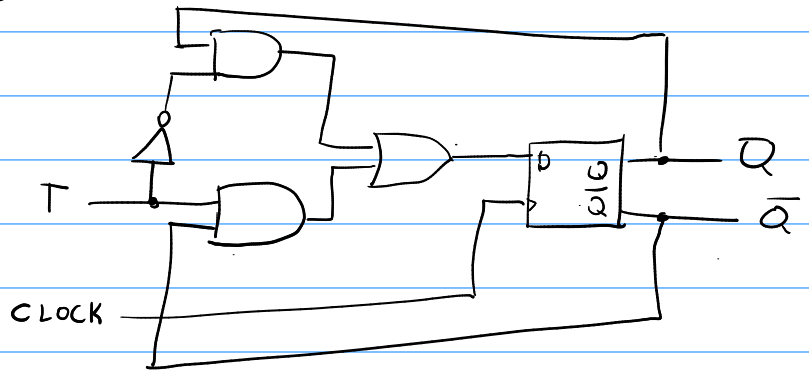
Clean assíncrono → quando clean_n = 0
↳ Q = 0 imediatamente

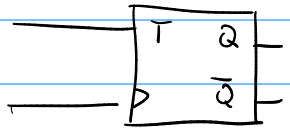
Preset → quando preset_n = 0
Q = 1 imediatamente



Clean síncrono.
clean_n = 1 → Funciona com D normalmente
clean_n = 0 → Q = 0 na próxima borda de alteração (nesse caso subida)

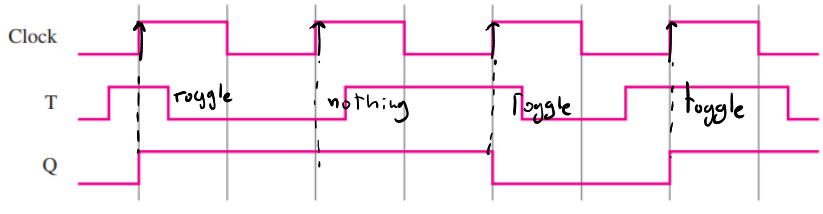
T Flip Flops





TOGGLE

T	Q(t+1)
0	Q(t)
1	Q̄(t)

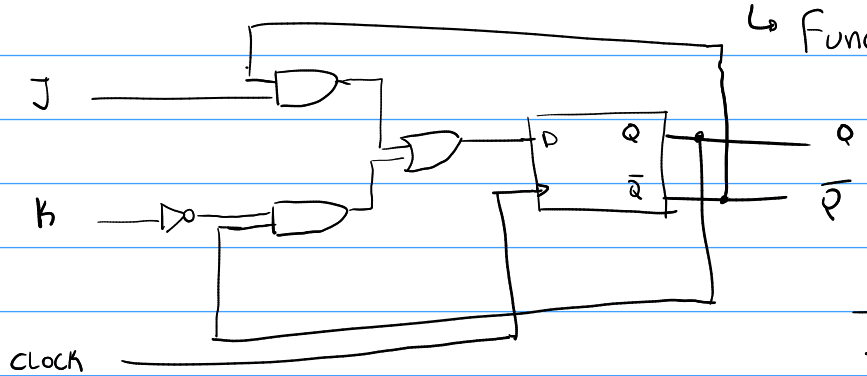


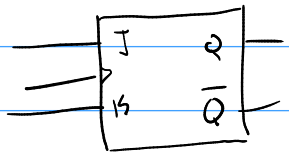
(d) Timing diagram

Flip Flop JK

Funciona como um SR (Set Reset) exceto quando J = K = 1 (caso indeterminado em SR)

↳ Funciona, então, como toggle (T)





J	K	Q(t+1)	
0	0	Q(t)	mantém } SR
0	1	0	
1	0	1	
1	1	Q̄(t)	Toggle } Toggle com T = 1

Resumo de Termos

Latch → XOR ou NAND, 1 bit de info com Set e Reset

Gated Latch → Latch com sinal de controle (CLK)

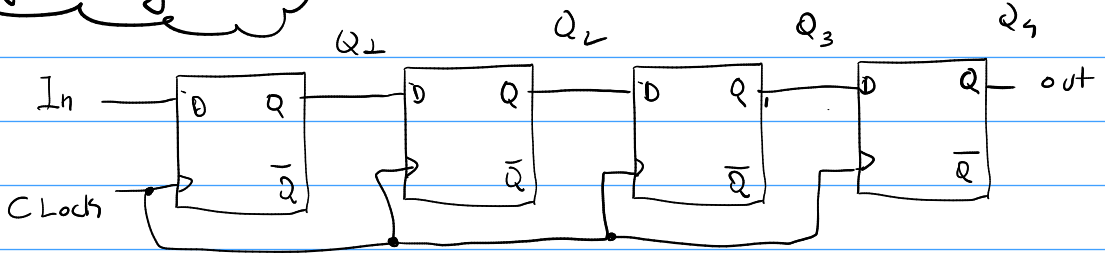
Troca somente se controle = 1

- Gated SR Latch – Set e Reset (S = R = 1 indeterminado) } enquanto controle = 1
- Gated D Latch – Saída do Latch = D } enquanto controle = 1

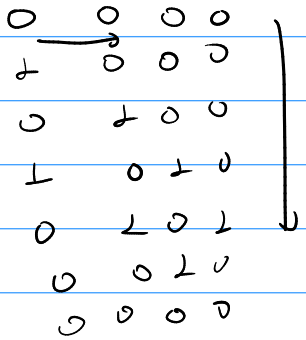
FlipFlop → Mudança somente na borda do sinal de controle

↑ CLK ou ↓ CLK

Shift Registen



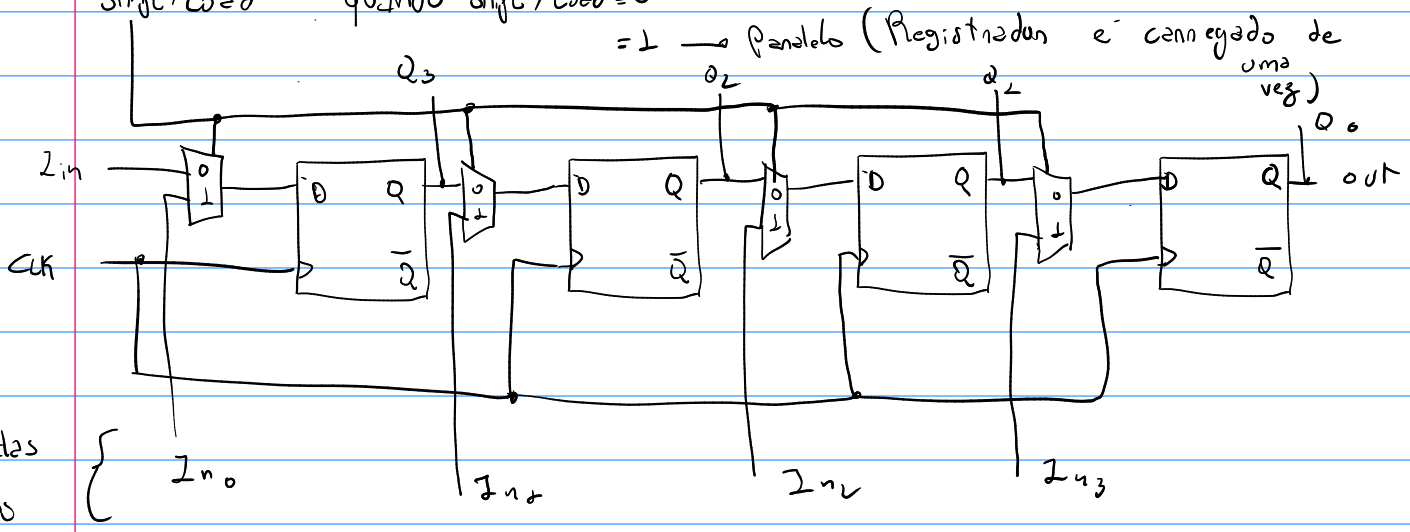
Cada entrada é propagada 1 bit para a direita a cada mudança de borda de clock (0 → 1)



$\overline{Shift}/load$ → Paralelo x Serial

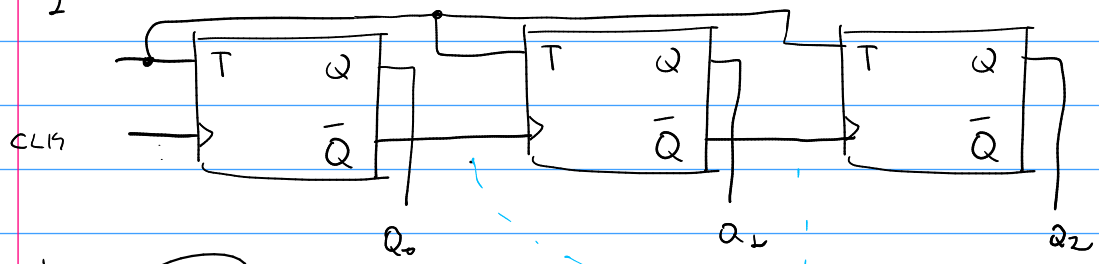
→ Shift (mux escolhe entrada serial)

$\overline{shift}/load$ quando $\overline{shift}/load = 0$ = 1 → Paralelo (Registradon é carregado de uma vez)



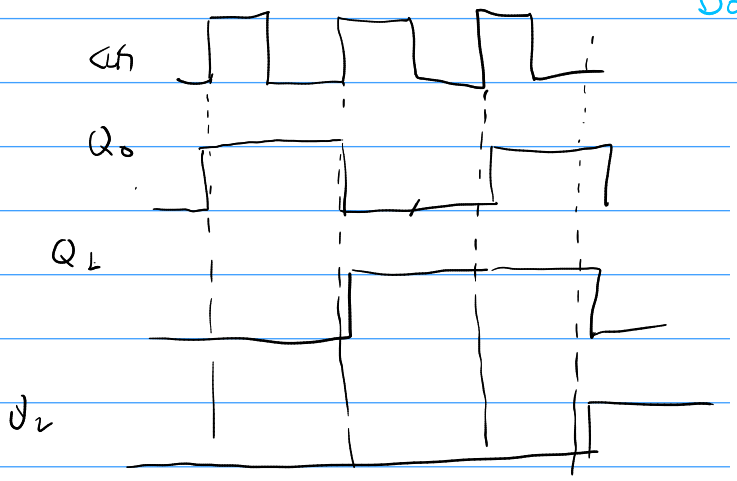
Entradas Paralelas { In_0 In_1 In_2 In_3

Contador Assíncrono



positiva (CLK ↑)

Borda de descida



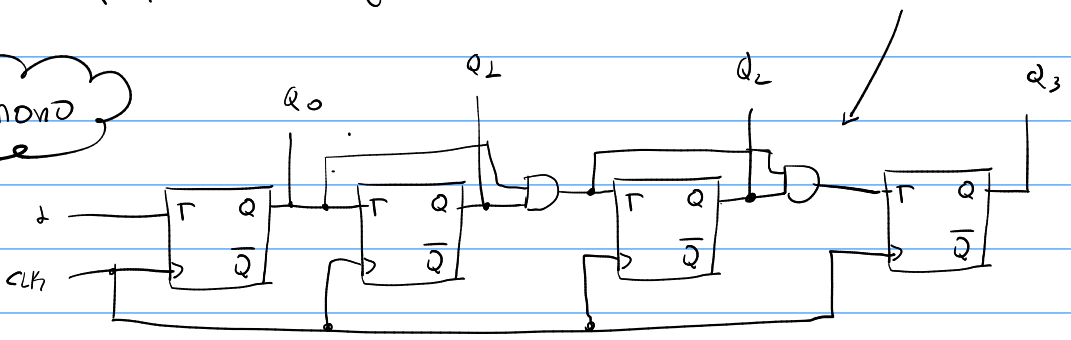
Clock cycle	Q_2 Q_1 Q_0
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1
8	0 0 0

Q_1 changes

Q_2 changes

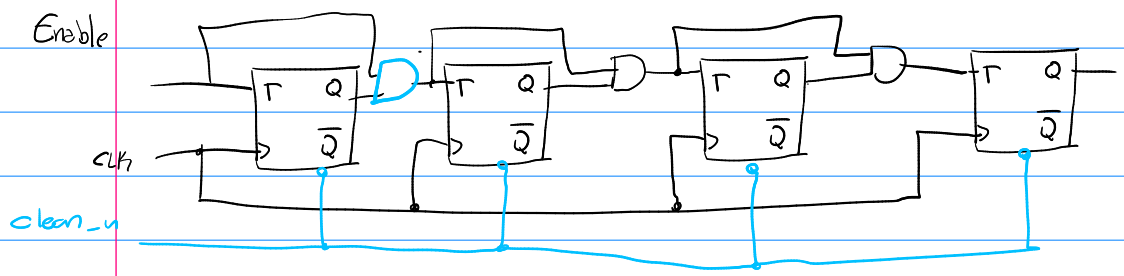
Delay causado pelo tempo de propagação é perigoso

Contador Síncrono

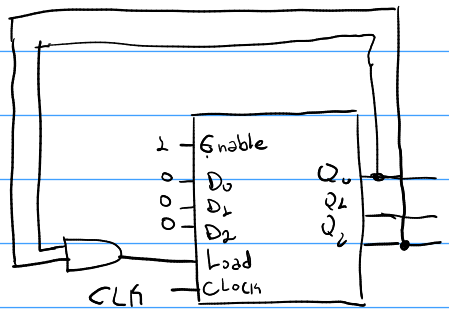


Delay se mantém p/ toda a contagem

Incluindo um sinal de controle:



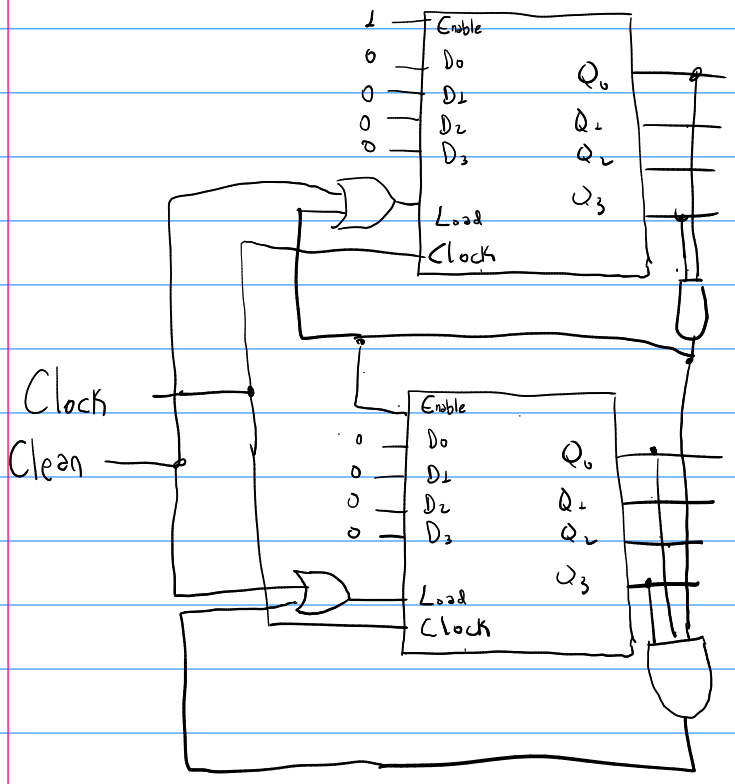
Contador Síncrono com reset síncrono → Chega em 5, daí reseta na próxima borda de clock



quando $Q_0, Q_1, Q_2 = 1, x, 1$
carrega 0,0,0
no contador

Contadores com RESET assíncrono geram problemas pois geram pulsos incompletos (pág 279 Brown)

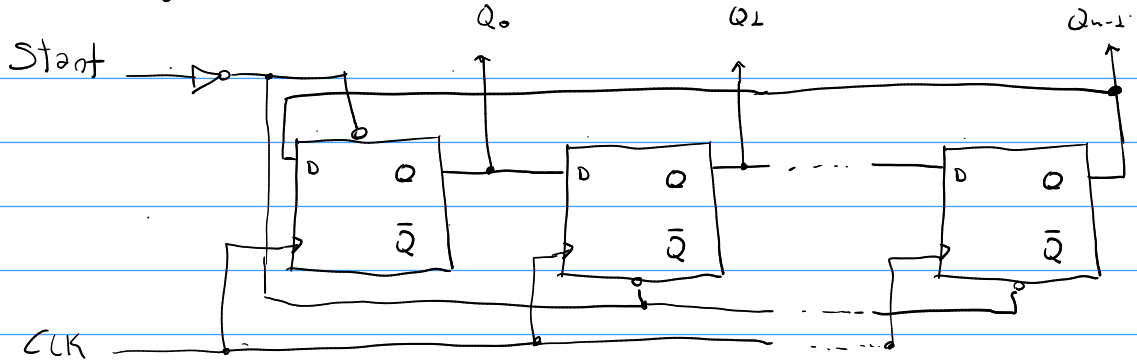
▸ BCD Counter — modulo -10 → 0, 1, 2 . . . 9 $\{Q_0 = Q_3 = 1$



← Só ativa o enable do segundo BCD quando o primeiro atinge 9 (próximo clock) ↳ reseta o primeiro

reseta o segundo quando o primeiro e o segundo = 9

Ring Counter

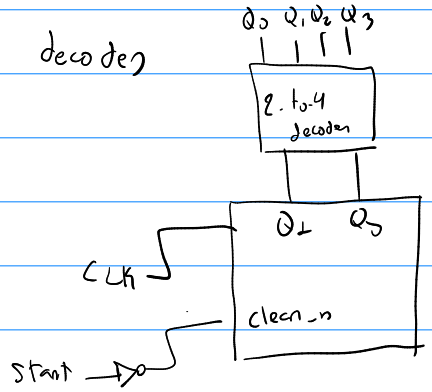


Saída do último volta ao primeiro

Start (1) → 100...0

(0) → contagem 100...0, 010...0, 001..., 000...1

Ou, Usando um 2 to 4 decoder e um contador de 2 bits:

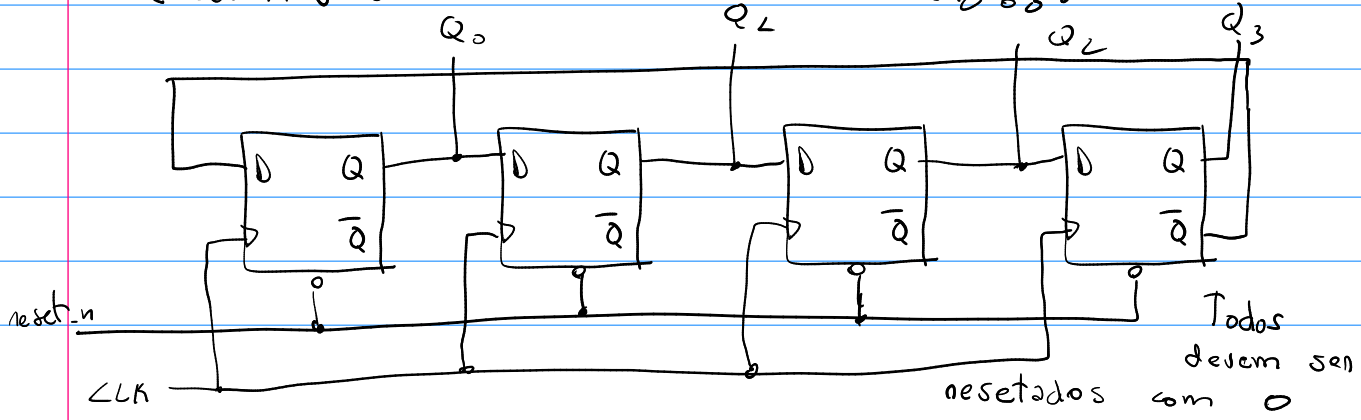


Johson Counter

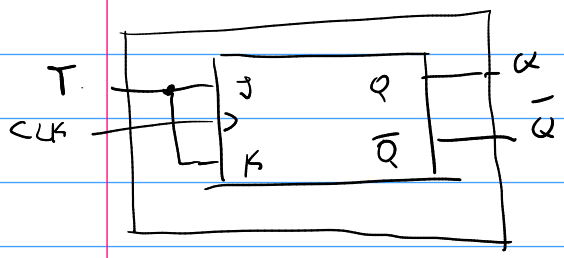
exemplo de 4 bits

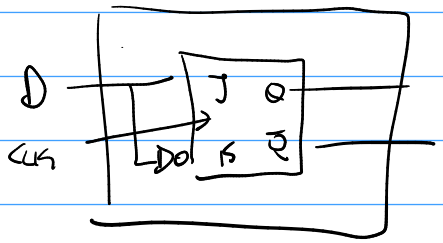
0000
1000
1100
1110
1111
0111
0011
0001
0000

Basta usar um ring-counter e realimentar o sistema com $\bar{Q}$ ao invés de Q



- Mostre como obter um FF-D e um FF-T a partir de um FF-JK



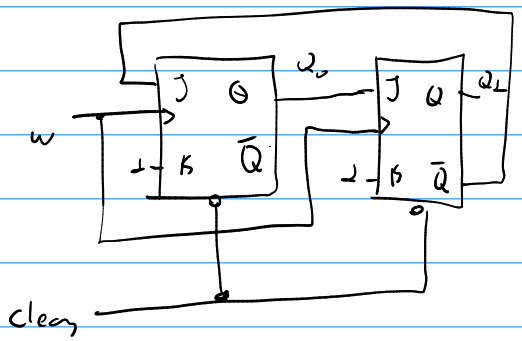


Tipo T

quando $J = K = 1$, é um tipo T

Funcionamento

	J_0	K_0	Q_0	J_1	K_1	Q_1
clear	1	1	0	0	1	0
t_1	1	1	1	1	1	0
t_2	0	1	0	0	1	1
t_3	1	1	0	0	1	0
t_5	1	1	1	1	1	0



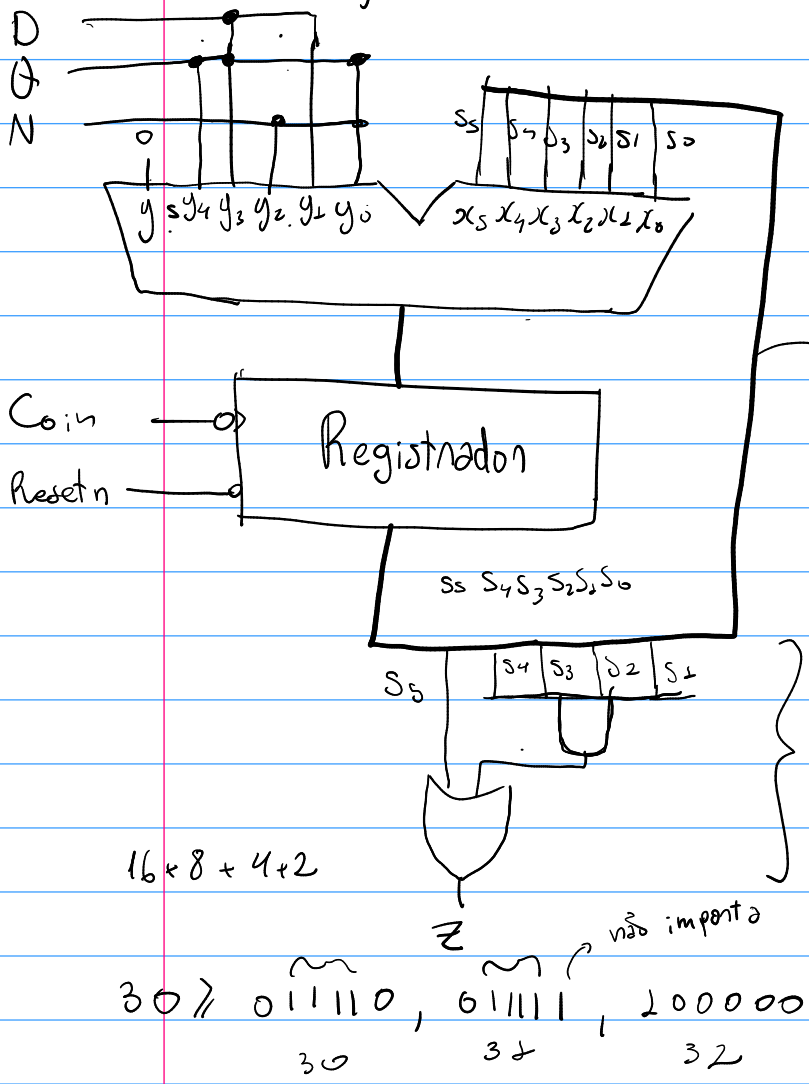
Example 5.20 **Problem:** Design a circuit that can be used to control a vending machine. The circuit has five inputs: Q (quarter), D (dime), N (nickel), *Coin*, and *Resetn*. When a coin is deposited in the machine, a coin-sensing mechanism generates a pulse on the appropriate input (Q, D, or N). To signify the occurrence of the event, the mechanism also generates a pulse on the line *Coin*. The circuit is reset by using the *Resetn* signal (active low). When at least 30 cents has been deposited, the circuit activates its output, Z. No change is given if the amount exceeds 30 cents.

Design the required circuit by using the following components: a six-bit adder, a six-bit register, and any number of AND, OR, and NOT gates.

D(dime) = 10¢ = 001010 Soma ≥ 30¢

Q(quarter) = 25¢ = 011001

N(nickel) = 5¢ = 000101



$S_{5..0}$

S[5..0]

A cada moeda: um pulso em D, Q ou N, fazendo com que o resultado da soma vá para o registrador. Entretanto, este só é carregado quando o pulso de Coin vai de 1 → 0 (↓). Isso garante que o tempo seja suficiente p/ o somador

$S \geq 30 \Leftrightarrow Z$

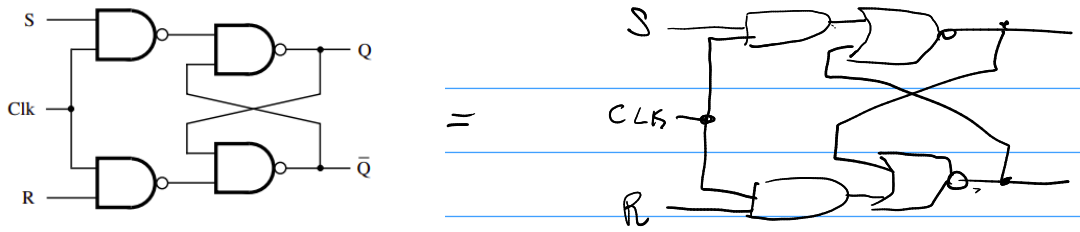
16 + 8 + 4 + 2

não importa

$30 \geq$ 011110, 011111, 100000

30 31 32

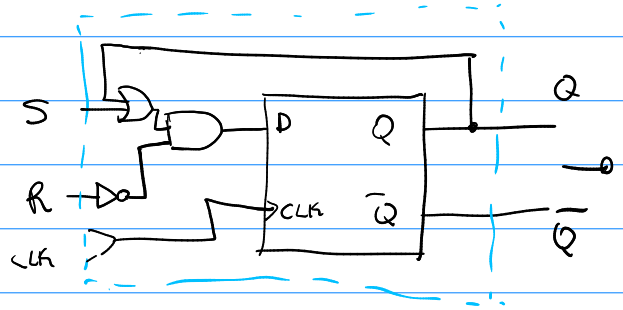
S.3 *



***5.5** An SR flip-flop is a flip-flop that has set and reset inputs like a gated SR latch. Show how an SR flip-flop can be constructed using a D flip-flop and other logic gates.

Construir flip-flop SR com flip-flop D

S,R	
0,0	Q
1,0	1
0,1	0
1,1	??? ⚠

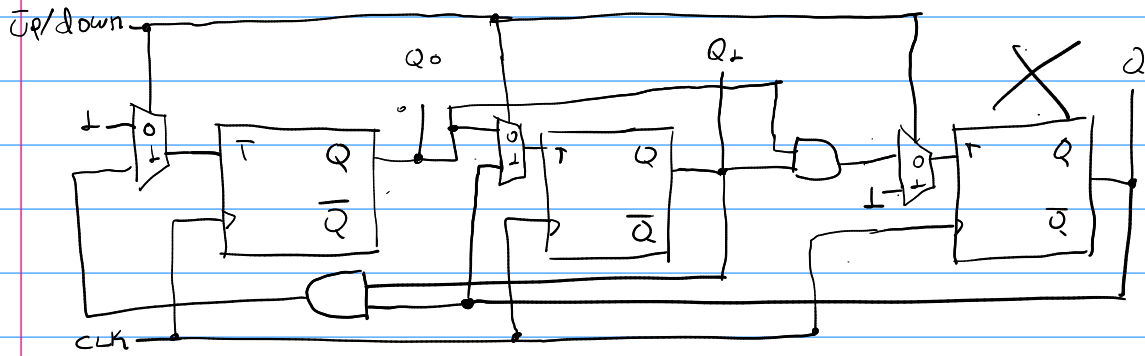


S	R	Q_{t+1}	Q_{t+1}
0	0	Q	Q
0	1	Q	0
1	0	Q	1
1	1		

D	Q
0	0
1	1

***5.15** Design a three-bit up/down counter using T flip-flops. It should include a control input called $\overline{\text{Up}}$/Down. If $\overline{\text{Up}}$/Down = 0, then the circuit should behave as an up-counter. If $\overline{\text{Up}}$/Down = 1, then the circuit should behave as a down-counter.

3 bit up/down counter com T (toggle

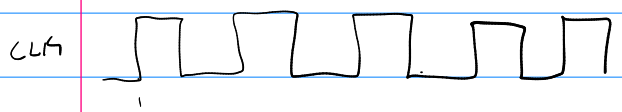


Q_0

0	0	1	0
1	0	0	0
0	1	0	0
1	1	1	0
0	0	1	1
1	0	1	1
0	1	1	1
1	1	0	1

Q_2

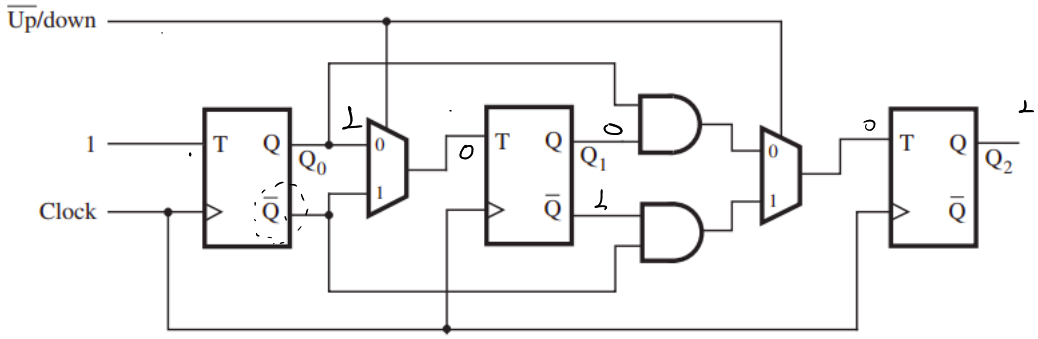
Q_2	Q_1	Q_0	$\overline{\text{Up}}$/down
0	0	1	0
0	1	0	0
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	1

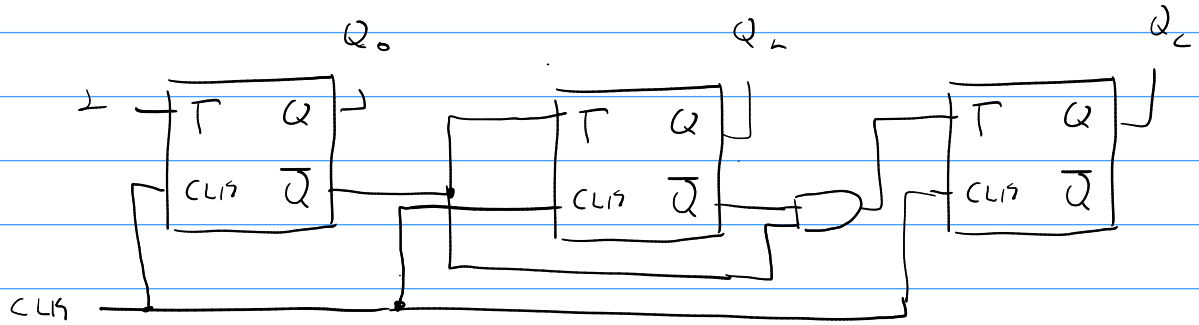


Quando $\overline{\text{Up}}$/down = 0, MUX 0 é selecionado (up counter normal).

Up/down = 1 o sistema faz o mesmo

5.15. The following circuit implements the desired counter





Toggle

***5.17** The circuit in Figure P5.3 looks like a counter. What is the counting sequence of this circuit?

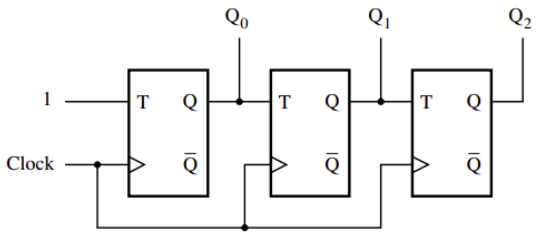


Figure P5.3 The circuit for Problem 5.17.