



# MC 602

## Circuitos Lógicos e Organização de Computadores

IC/Unicamp

Prof Mario Côrtes

### Capítulo 7

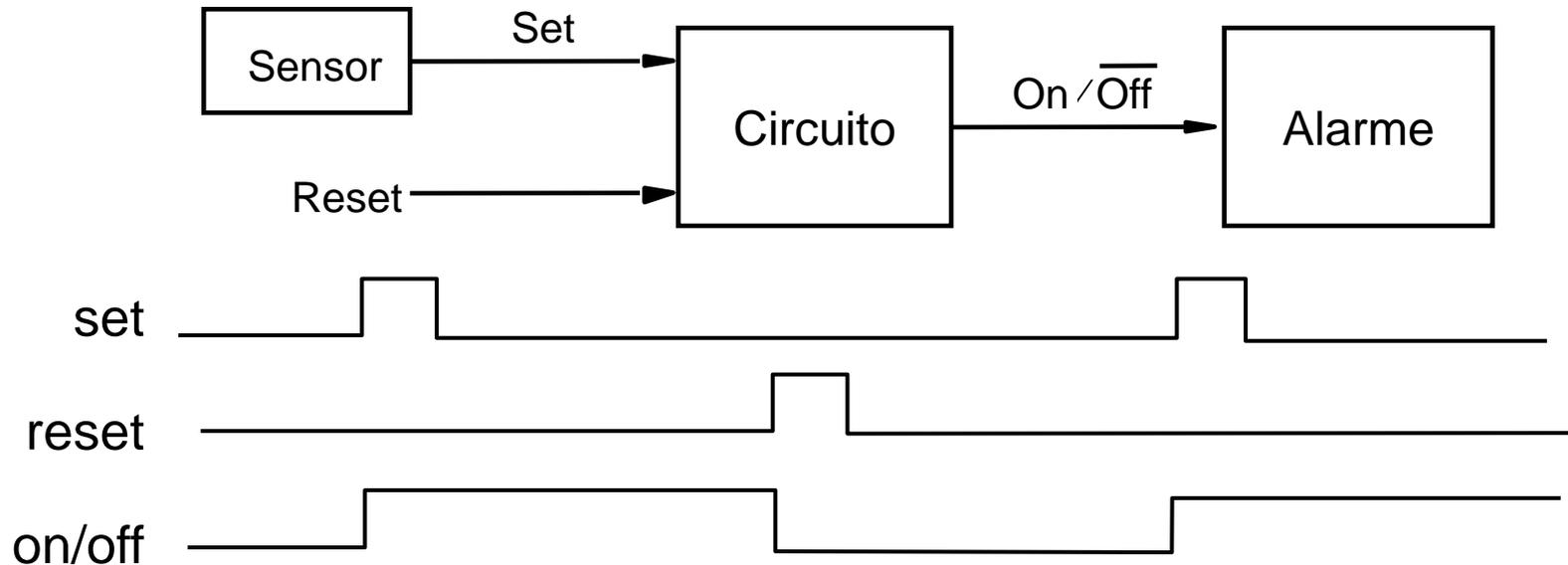
Circuitos sequenciais: latches, flip-flops, registradores, contadores

# Tópicos

- Elementos de armazenamento: latches e flip-flops
- Registradores
- Contadores

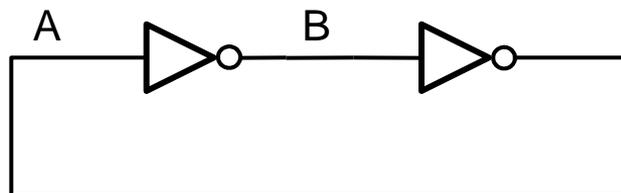
# Circuitos combinacionais / sequenciais

- Circuitos combinacionais
  - $y(t) = f[x(t)] \rightarrow$  não há dependência de  $x(t-1)$  ou  $x(t-2) \rightarrow$  não há memória ou histórico
- Alguns problemas não podem ser resolvidos com circuitos combinacionais



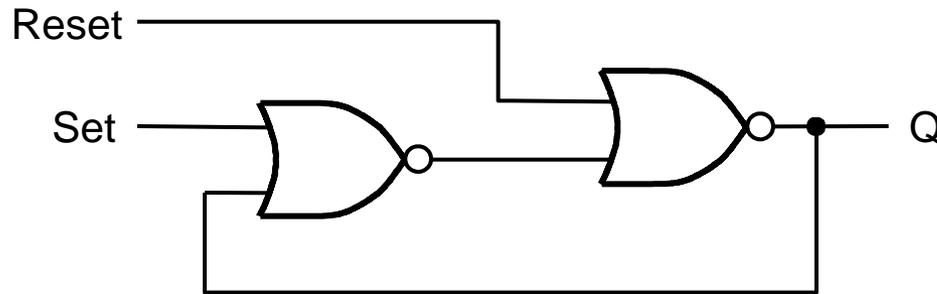
# Circuitos sequenciais

- Circuitos sequenciais
  - $y(t) = f[x(t-1), x(t-2), \dots]$   $\rightarrow$  depende do histórico de entradas  $\rightarrow$  novos conceitos: estado ou memória
- Elemento básico: elemento de memória
- O mais simples e rudimentar:



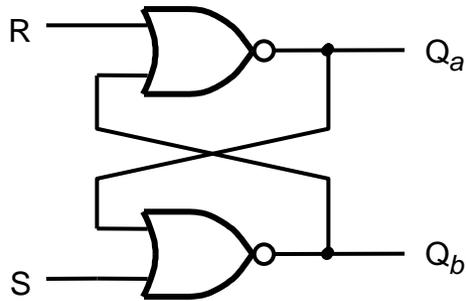
- Elemento de memória com dois estados estáveis:  
 $A=0$  e  $B=1$  ou  $A=1$  e  $B=0$

# Um elemento básico feito com NORs



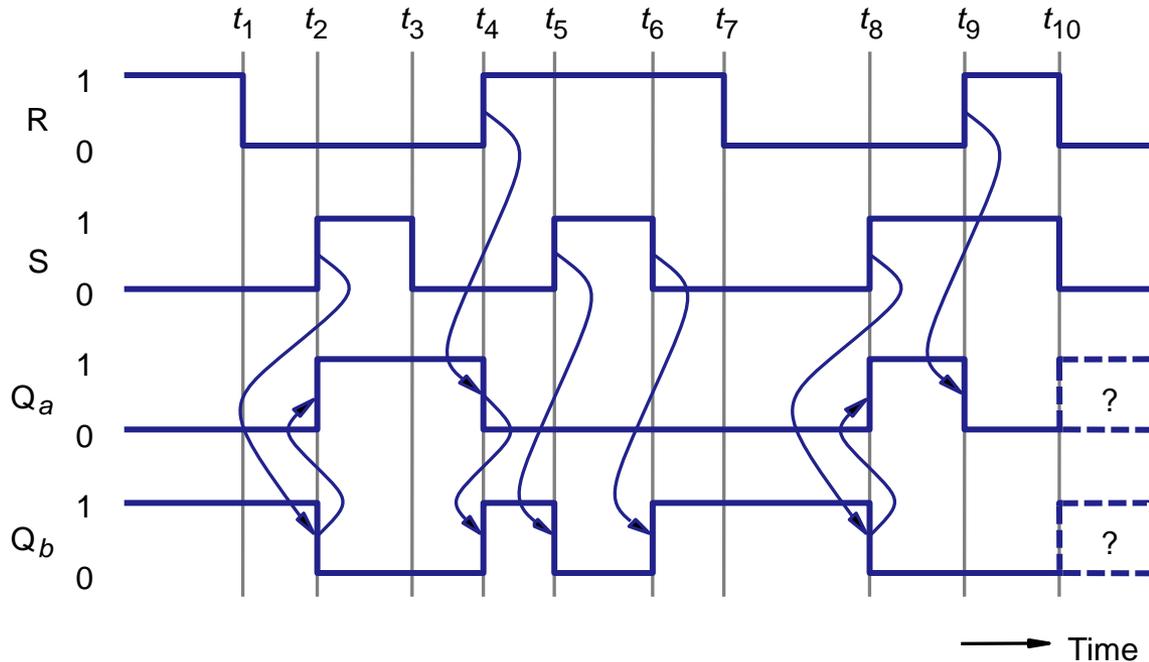
- Extensão dos inversores em anel
  - permite “carga” de dados:
  - set  $\rightarrow$  1
  - reset  $\rightarrow$  0
- Base para o Latch Set Reset = Latch SR

# Latch SR com NORs



S	R	$Q_a$	$Q_b$
0	0	$Q_a(t-1)$	$Q_b(t-1)$
0	1	0	1
1	0	1	0
1	1	0	0

sem mudança

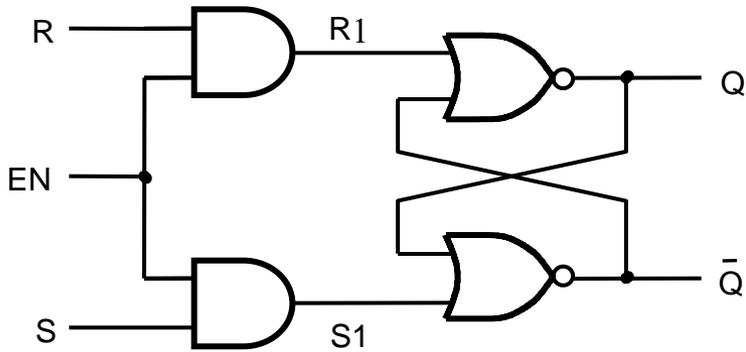




# Latch SR: considerações

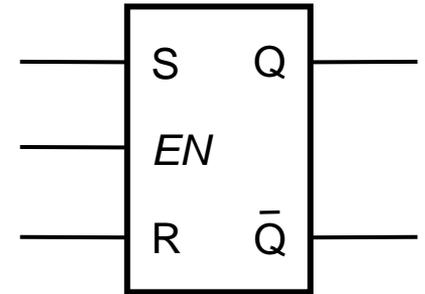
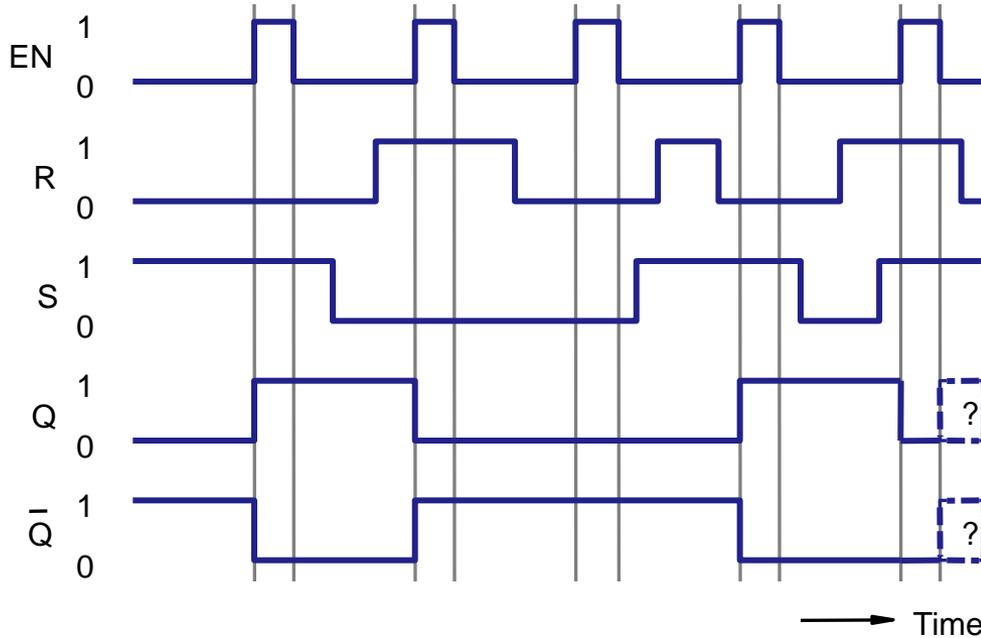
- Forma de onda assume atrasos de propagação = 0
- Situação de entrada SR=11 é indesejável
- Mas circuito pode ser usado no sistema de alarmes
  
- Pode ser interessante ter a função de habilitar (enable) a entrada de dados ou não no Latch SR → Latch SR chaveado ou com enable

# Latch SR chaveado

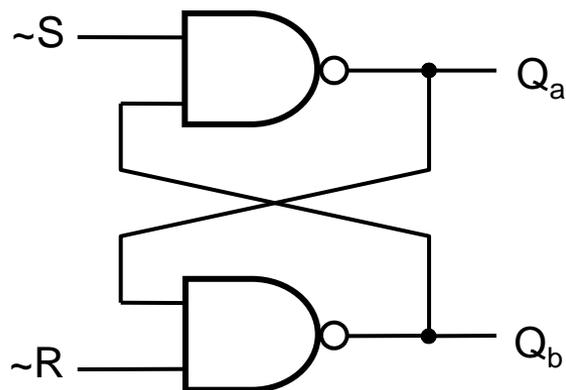


Clk	S	R	$Q(t+1)$
0	x	x	$Q(t)$ (no change)
1	0	0	$Q(t)$ (no change)
1	0	1	0
1	1	0	1
1	1	1	x

Por que?



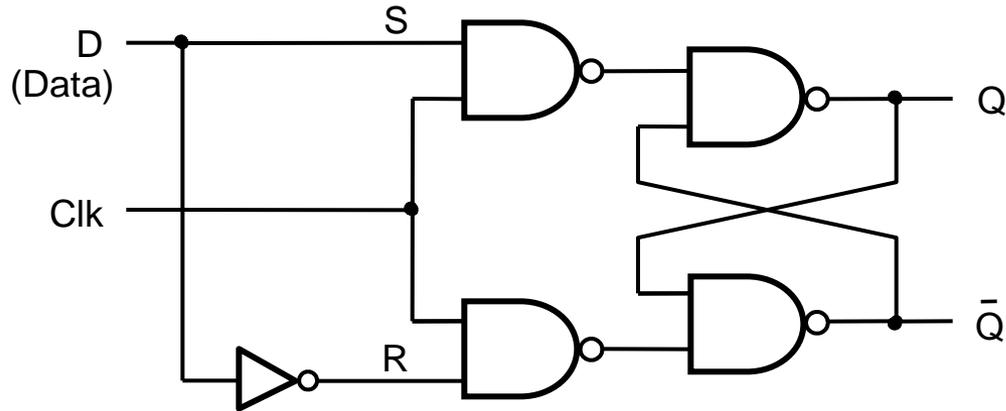
# Latch SR com NANDs



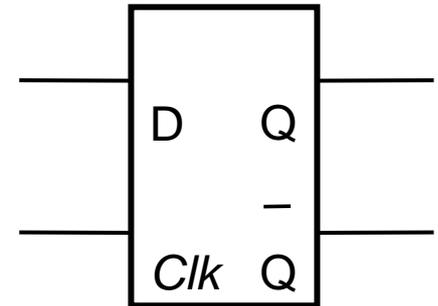
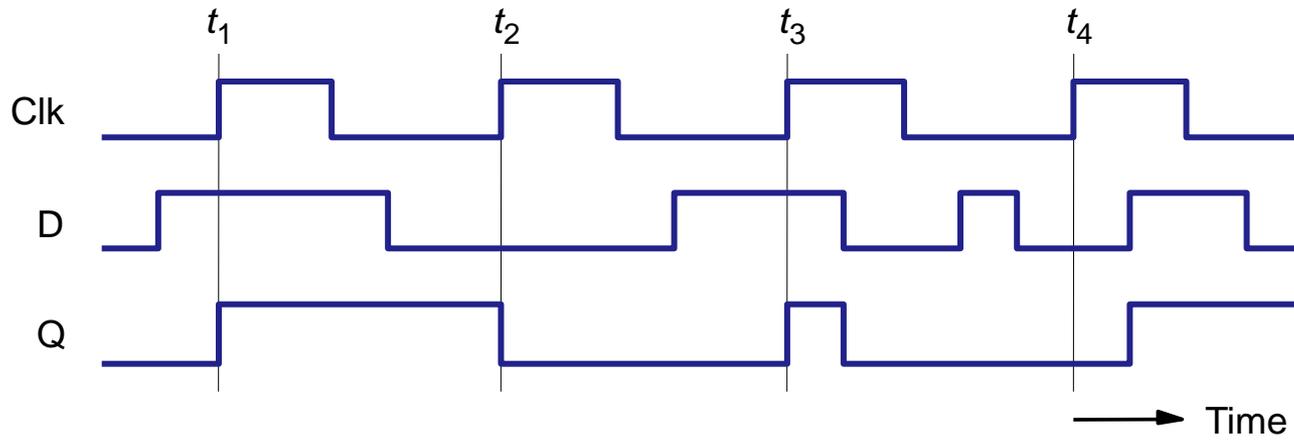
$\sim S$	$\sim R$	$Q_a$	$Q_b$	
1	1	$Q_a(t-1)$	$Q_b(t-1)$	sem mudança
0	1	1	0	
1	0	0	1	
0	0	0	0	

- Observar entradas “ativo-baixo”

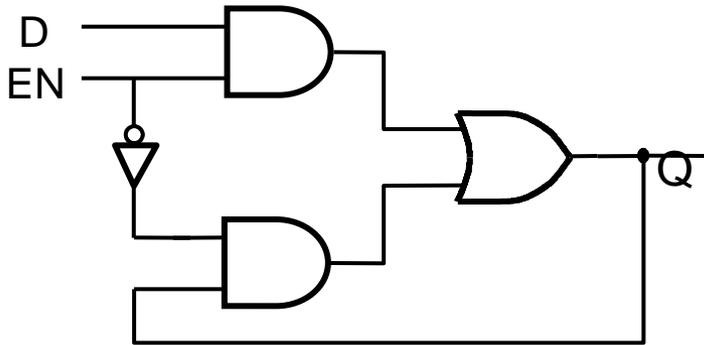
# Latch tipo D chaveado



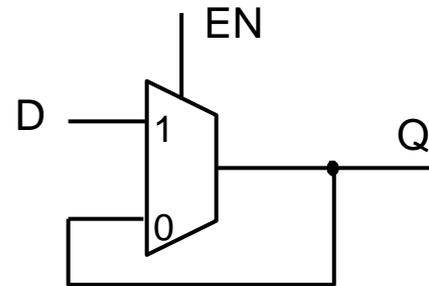
Clk	D	$Q(t+1)$
0	x	$Q(t)$
1	0	0
1	1	1



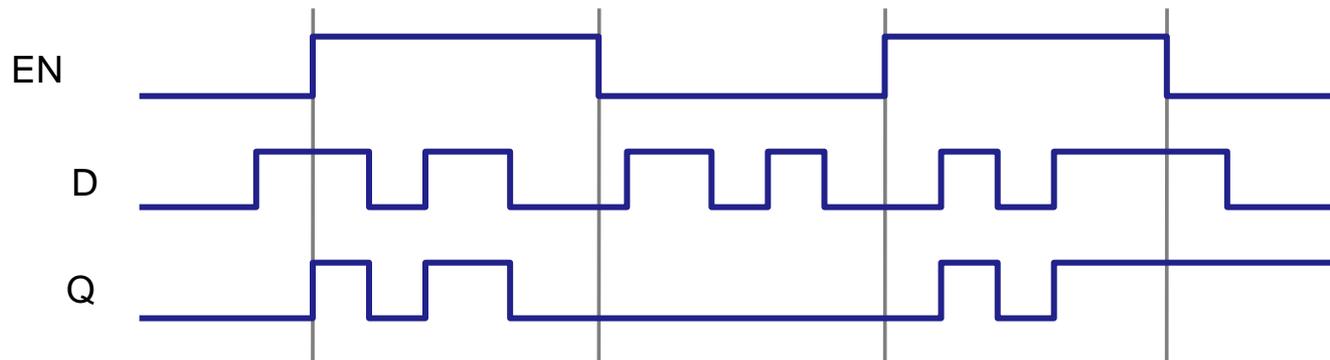
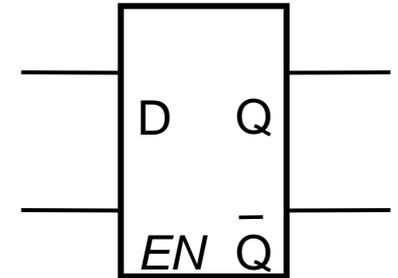
# Latch tipo D: outra implementação



Equivalente a



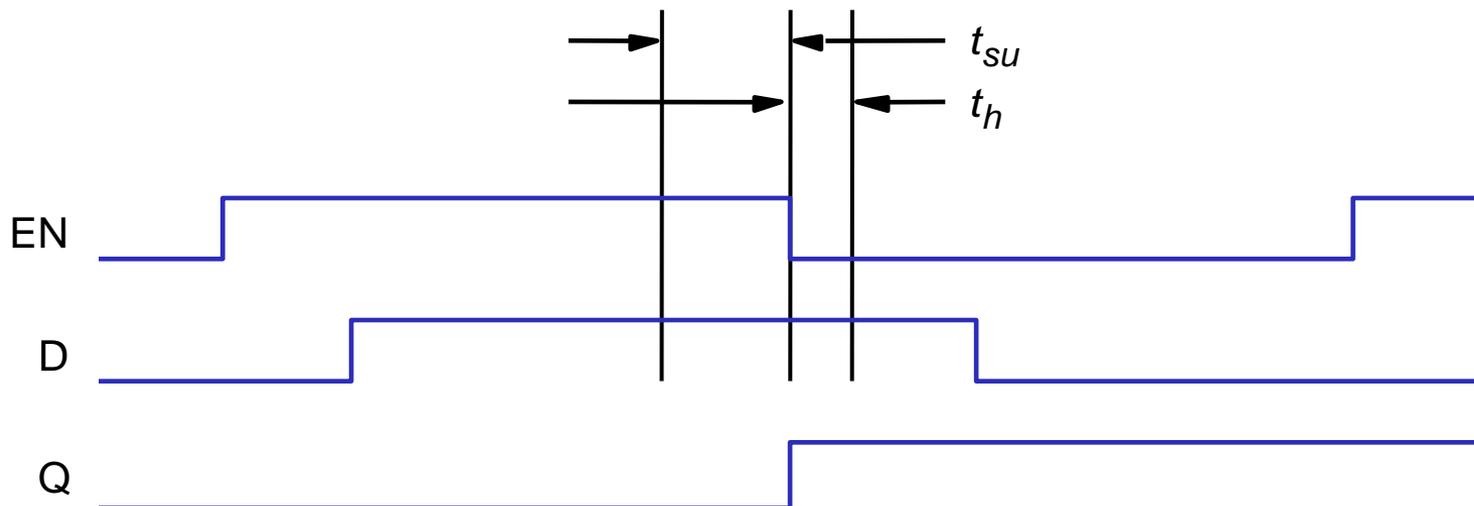
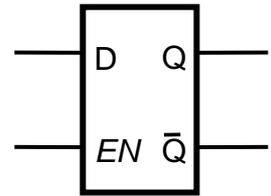
Símbolo



Analisar efeito do atraso do inversor nesta configuração e na configuração com  $\sim EN$

# Tempos de setup e hold em latches

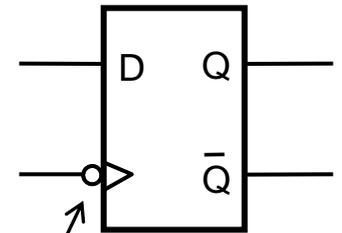
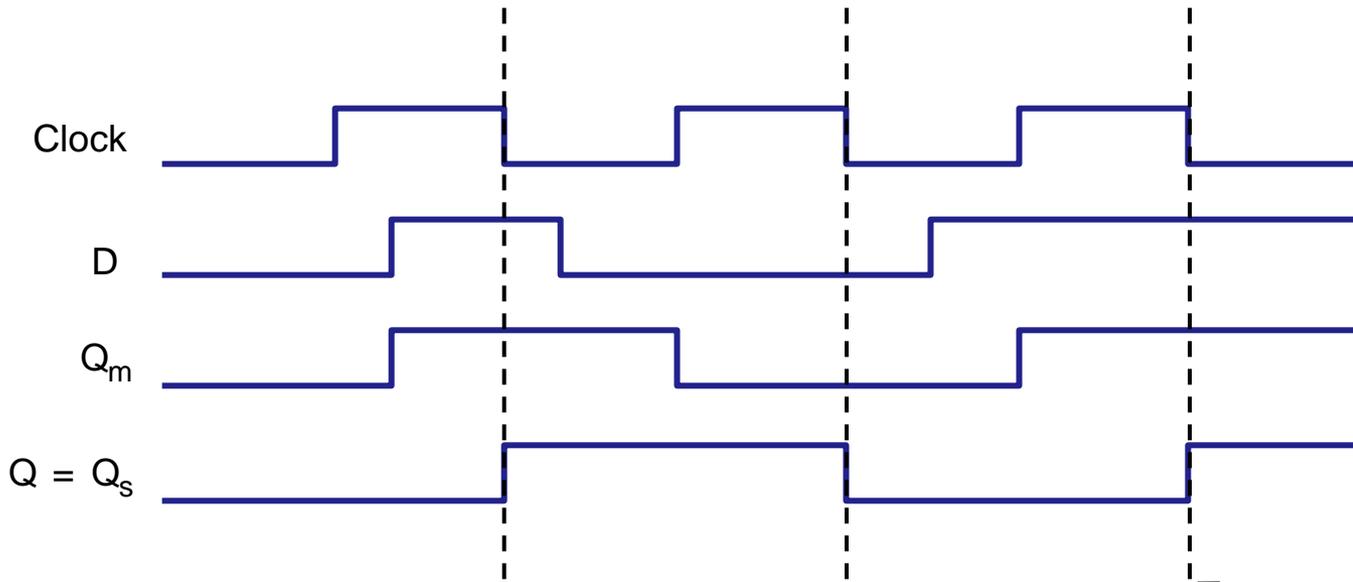
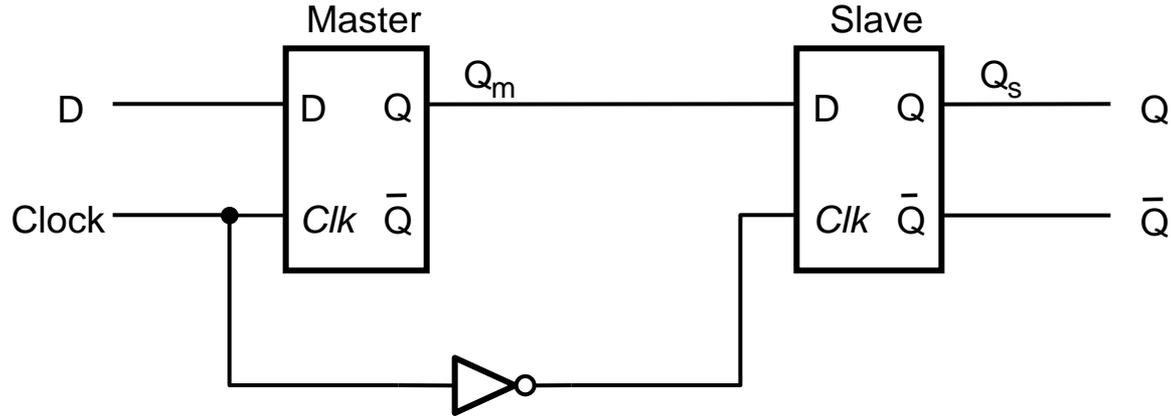
- Valor armazenado (travado) quando  $EN \rightarrow 0$
- O que acontece se D está mudando?
  - valor travado é imprevisível
- Tempos de guarda
  - antes: tempo de setup (preparação)
  - depois: tempo de hold (manutenção)
  - dependem dos atrasos internos e da tecnologia
  - ver datasheets



# Flip-flops

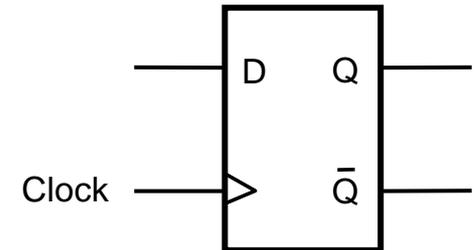
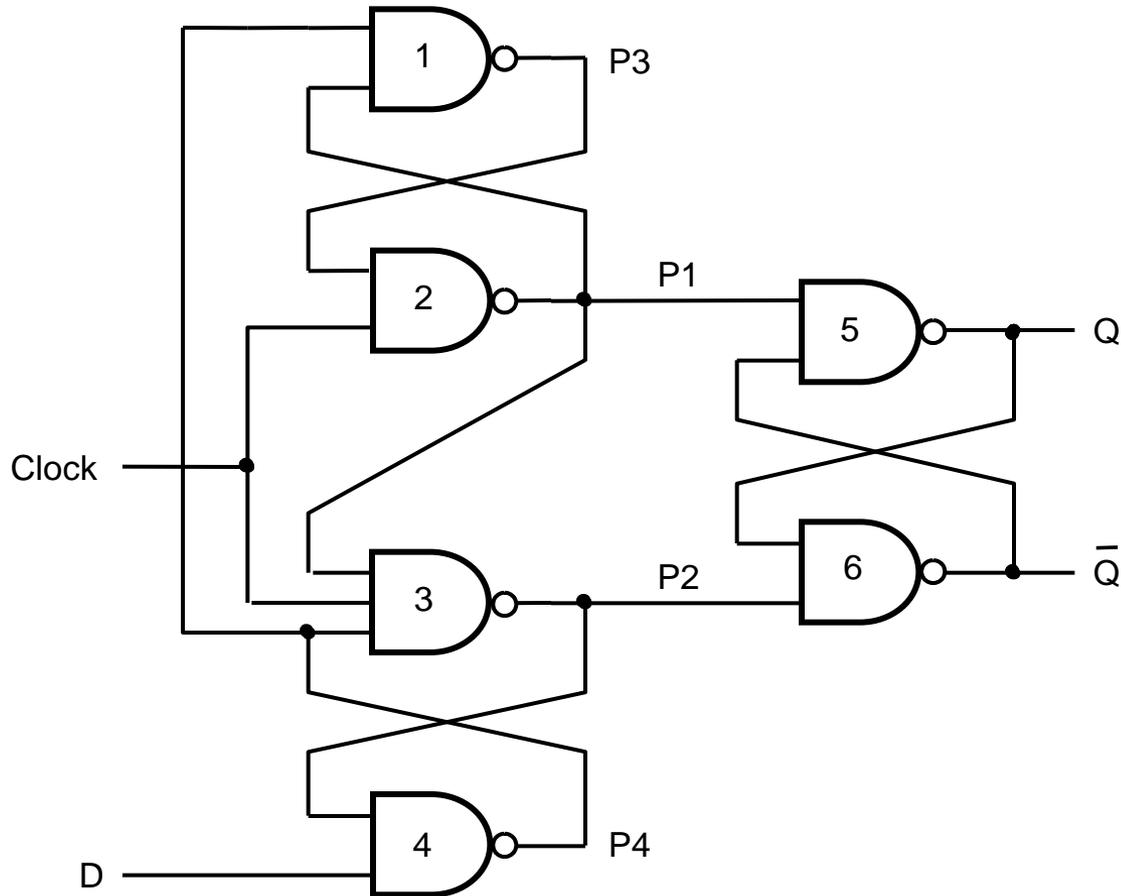
- Latches são sensíveis ao nível: o estado pode mudar várias vezes durante  $EN=1$
- É interessante ter circuitos que mudam de estado apenas uma vez (nas transições do EN)
- Flip-Flops
  - Mestre-Escravo: composto por dois latches controlados por clocks (enables) complementares
  - Edge-triggered / sensível à borda do clock

# Flip-Flop Mestre Escravo tipo D



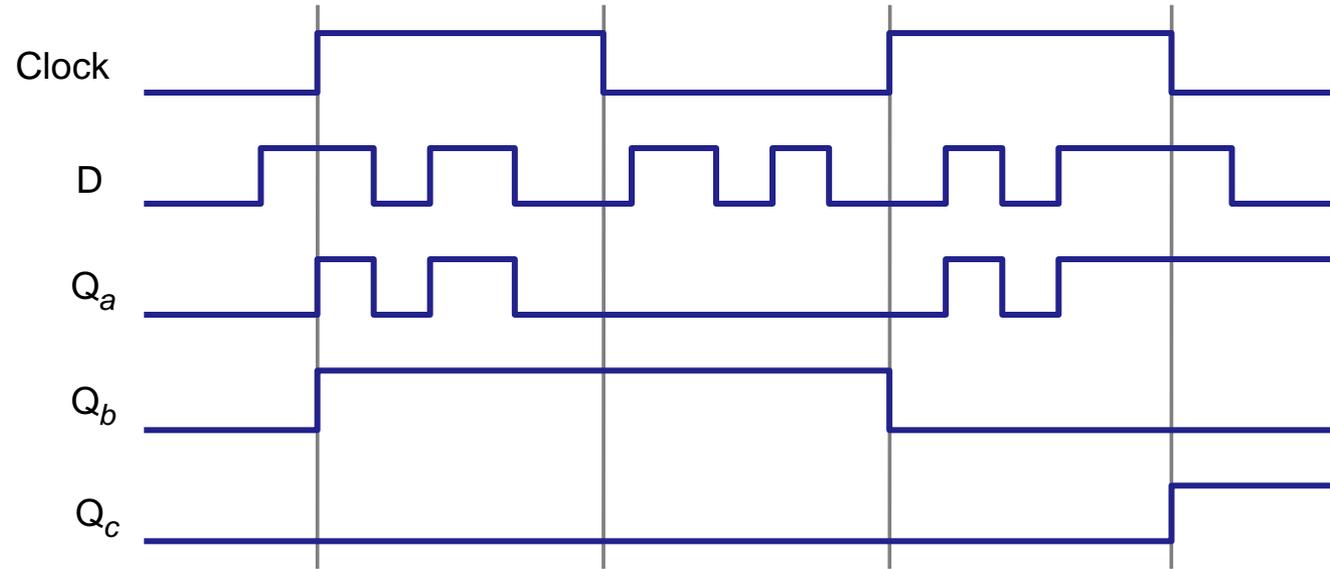
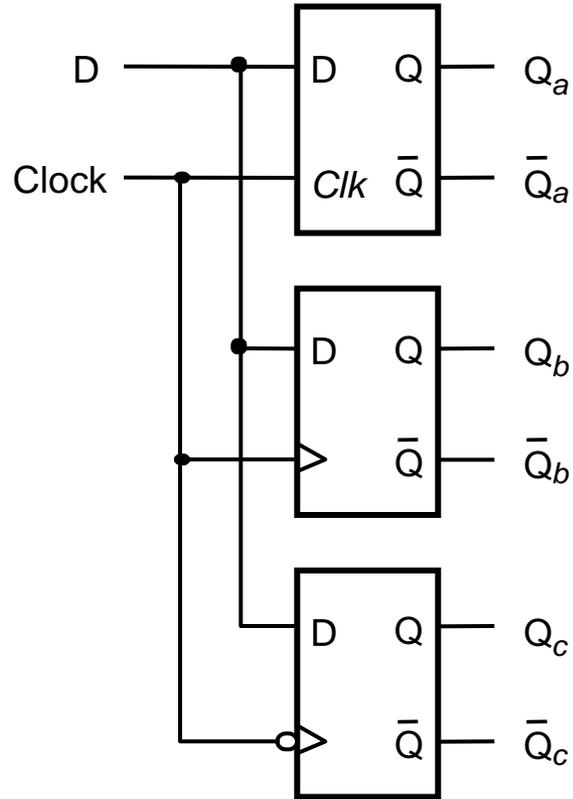
Sensível à borda de DESCIDA

# Um Flip-Flop tipo D clássico (sensível à borda)



- ver análise de funcionamento no livro texto

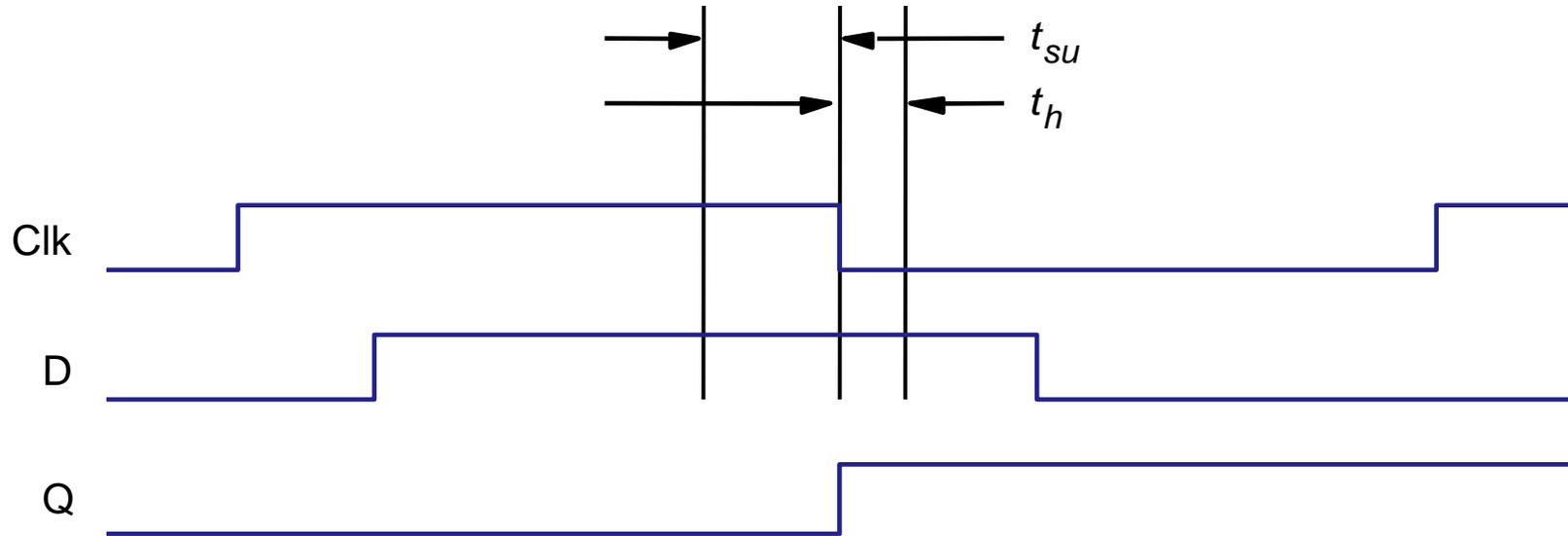
# Latch e FF: comportamento comparado



# Latches e Flip-Flops: diferenças

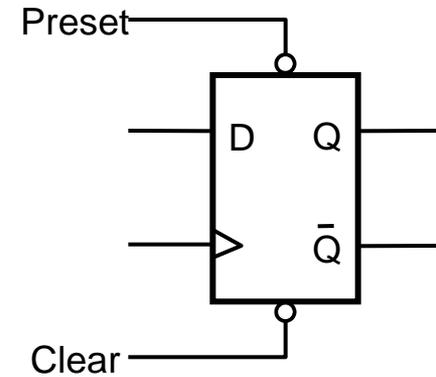
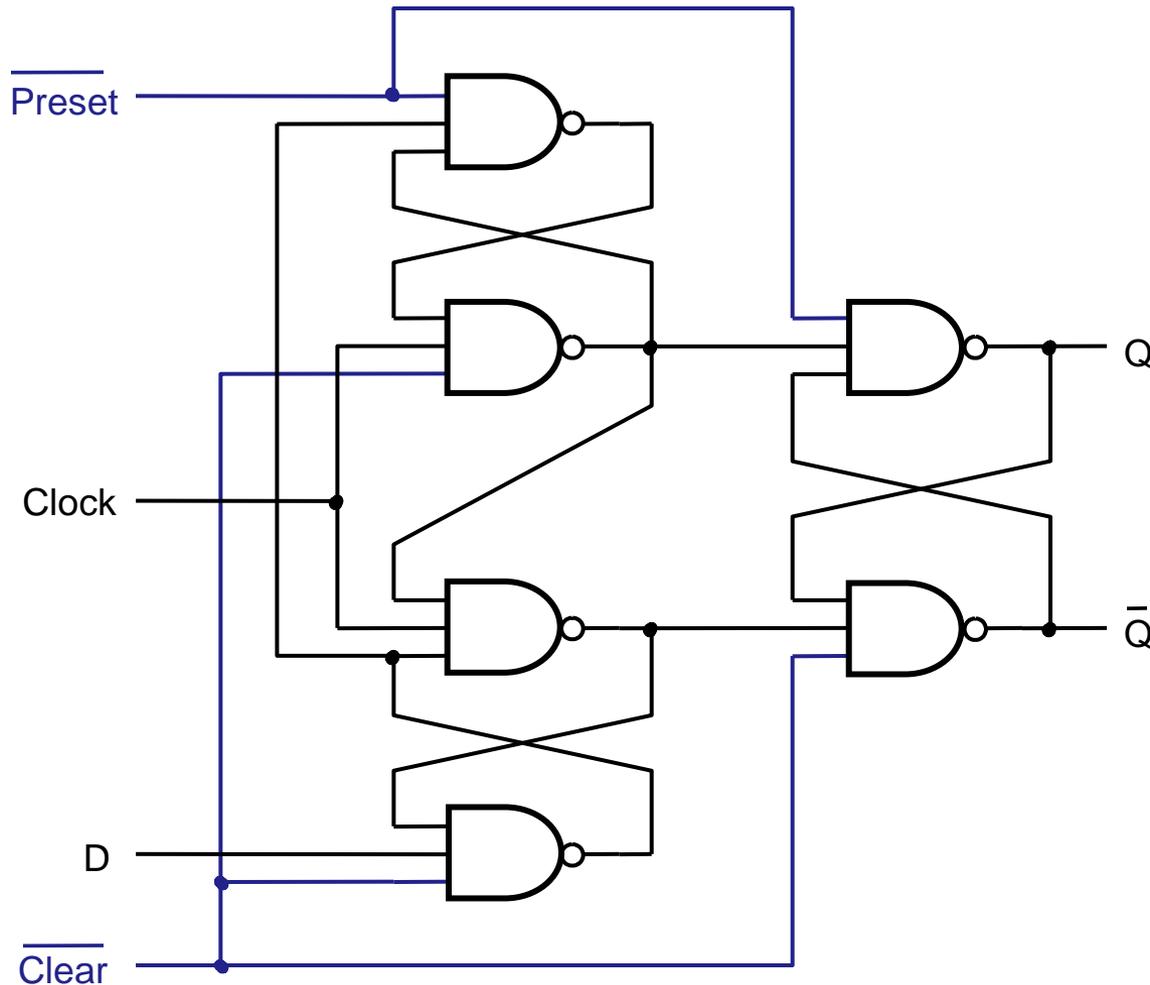
- Manifestação da saída Q em função de variações na entrada D:
  - Latch: transparente durante EN (ou Ck) ativos, ou seja, entrada D passa diretamente para a saída Q
  - Flip-Flop: na borda do Clock, o valor presente na entrada D é transferido para Q
- Instante em que o valor da entrada D é armazenado
  - Latch: valor armazenado é o presente na entrada D no instante em que EN (ou Ck) é desativado (operação de latch ou travamento)
  - Flip-Flop: na borda do Clock, o valor presente na entrada D é armazenado

# Tempos de setup e hold

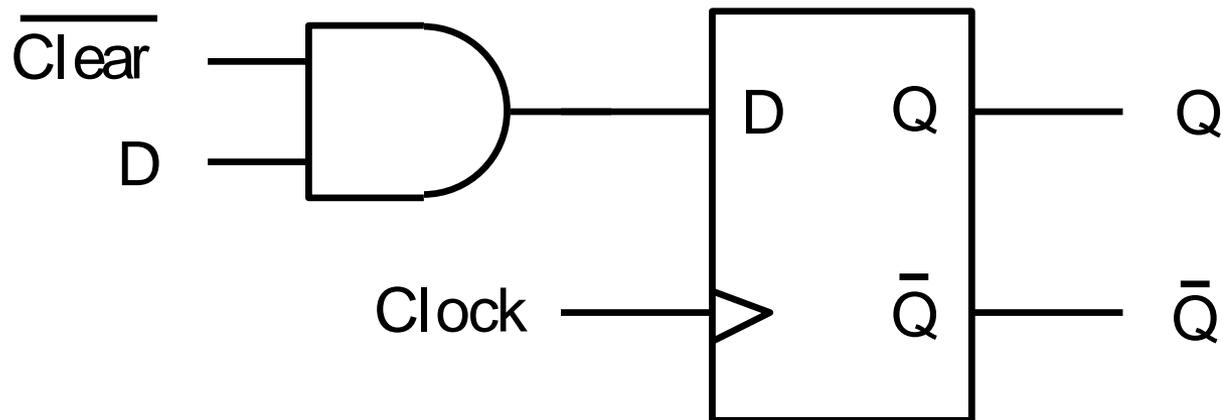


- $T_{su}$ : tempo de guarda antes da borda do clock (de descida, no exemplo) durante o qual a entrada D não deve mudar
- $T_h$ : idem, para depois da borda do clock

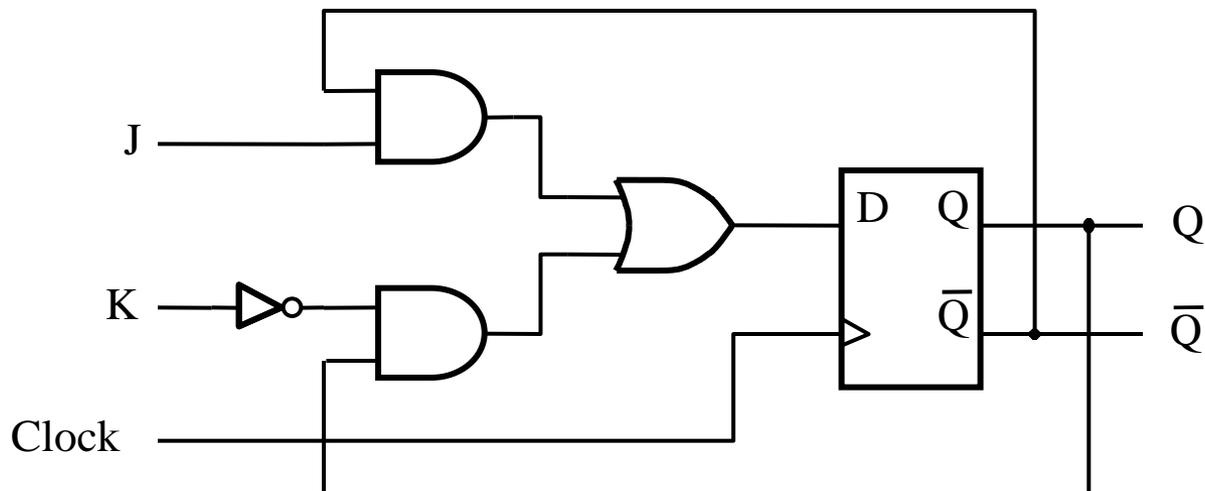
# FF D: borda de subida, com Preset e Clear assíncronos



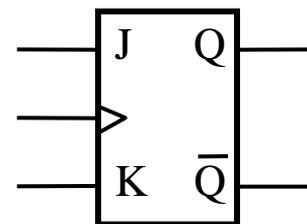
# FF D com Clear síncrono



# Flip-Flop JK

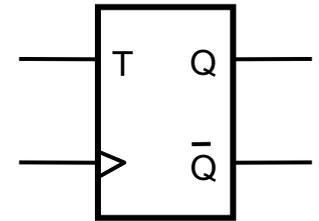
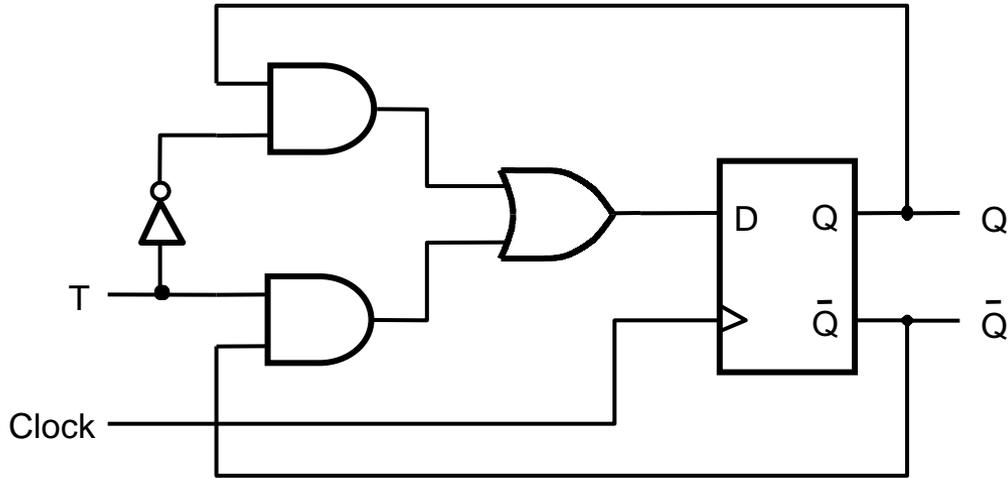


J	K	$Q(t+1)$
0	0	$Q(t)$
0	1	0
1	0	1
1	1	$\bar{Q}(t)$

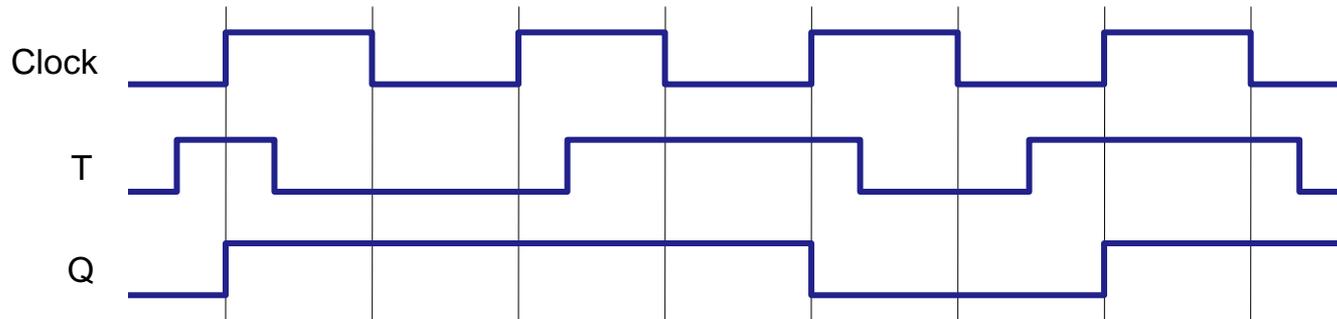
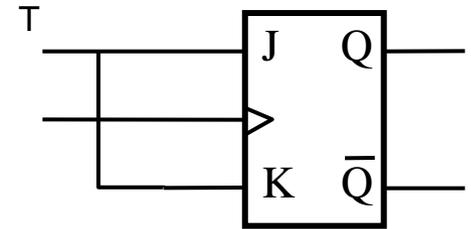


Resolve problema do SR

# Flip-Flop tipo T

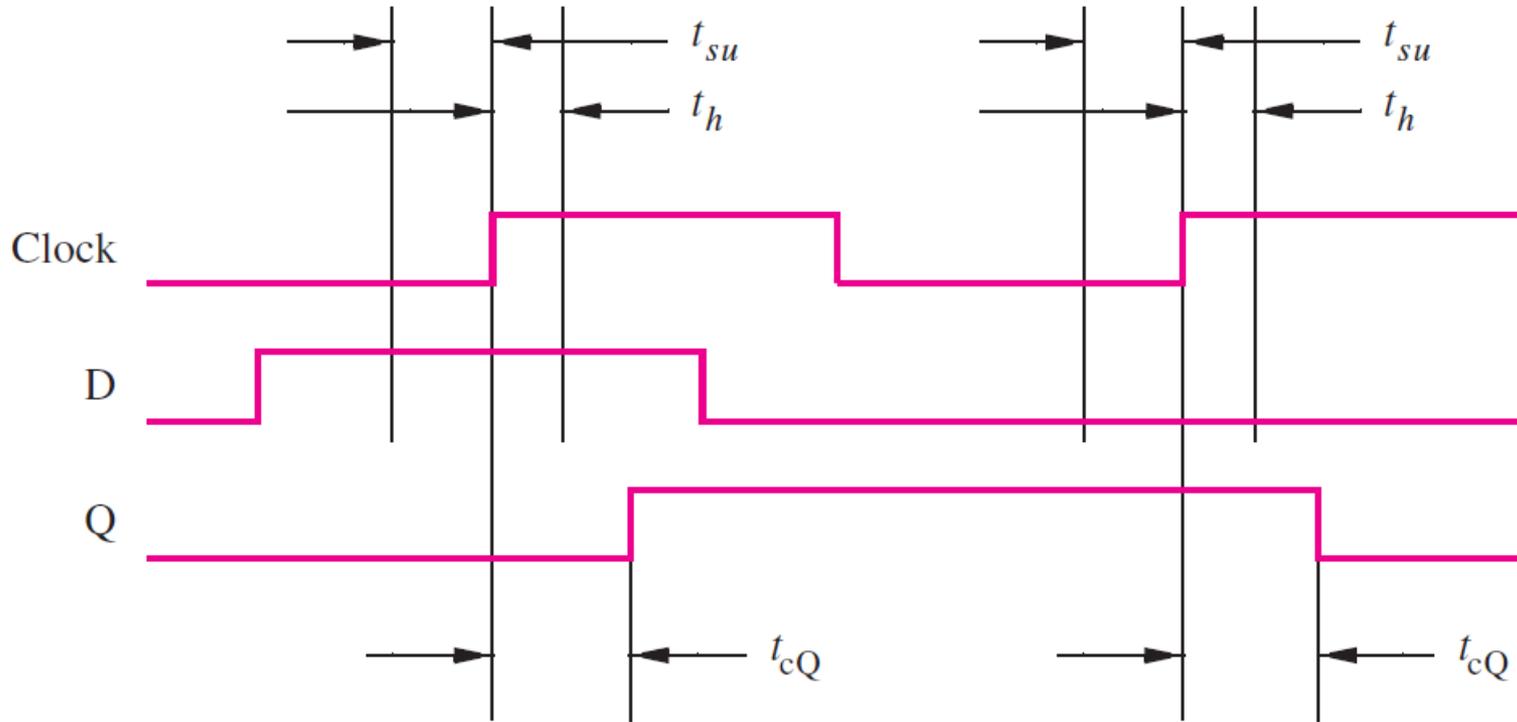


Equivalente a

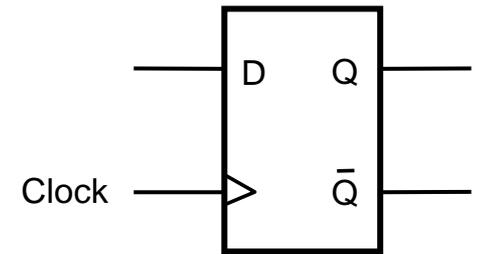


T	$Q(t+1)$
0	$Q(t)$
1	$\bar{Q}(t)$

# Parâmetros de timing de Flip-Flops



- $t_{su}$ ,  $t_h$ : tempos de setup e hold
- $t_{cQ}$ : atraso de propagação do clock até Q





# MC 602

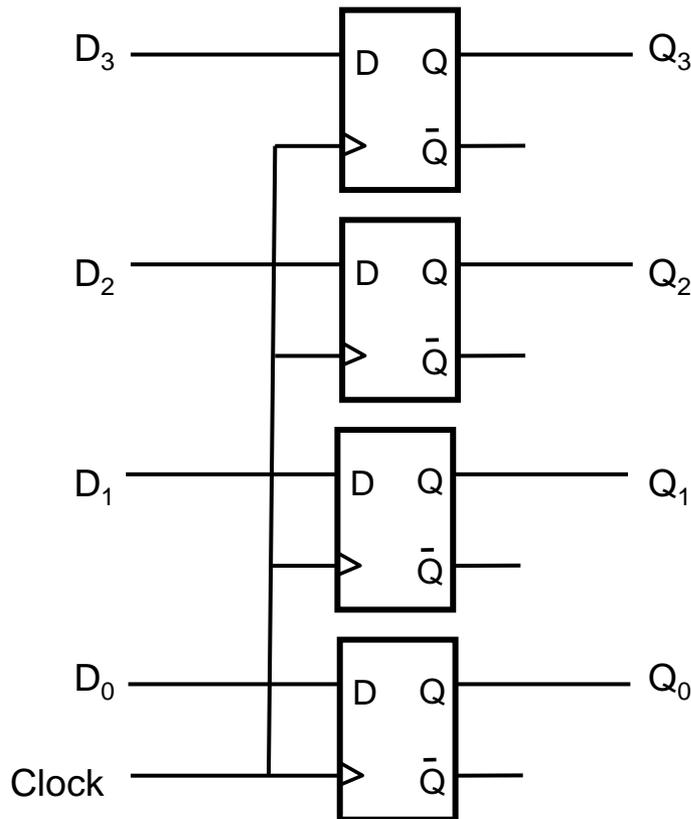
## Registradores e Contadores

# Tópicos de Registradores

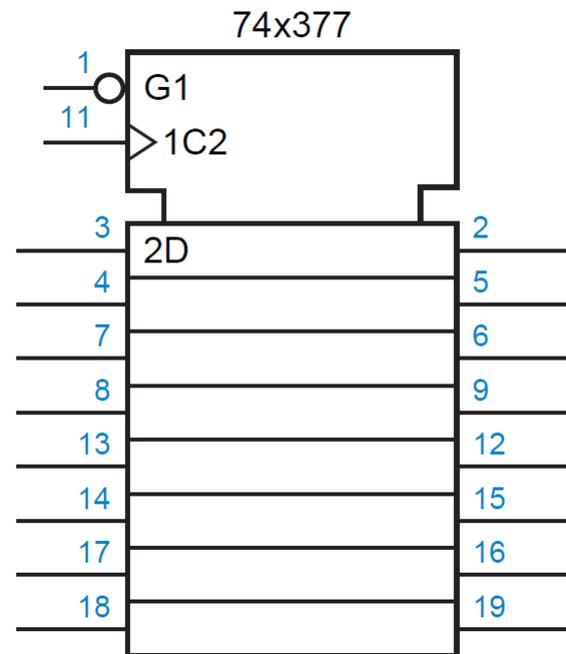
- Construção usando flip-flops
- Clear assíncrono e Enable
- Registradores deslocamento
- Carga paralela
- Registrador deslocamento universal
- Exemplo de uso em barramento

# Registadores

- Conjunto de elementos de memória (flip-flops) utilizados para armazenar n bits a cada borda do clock
- Utilizam em comum os sinais de clock e controle

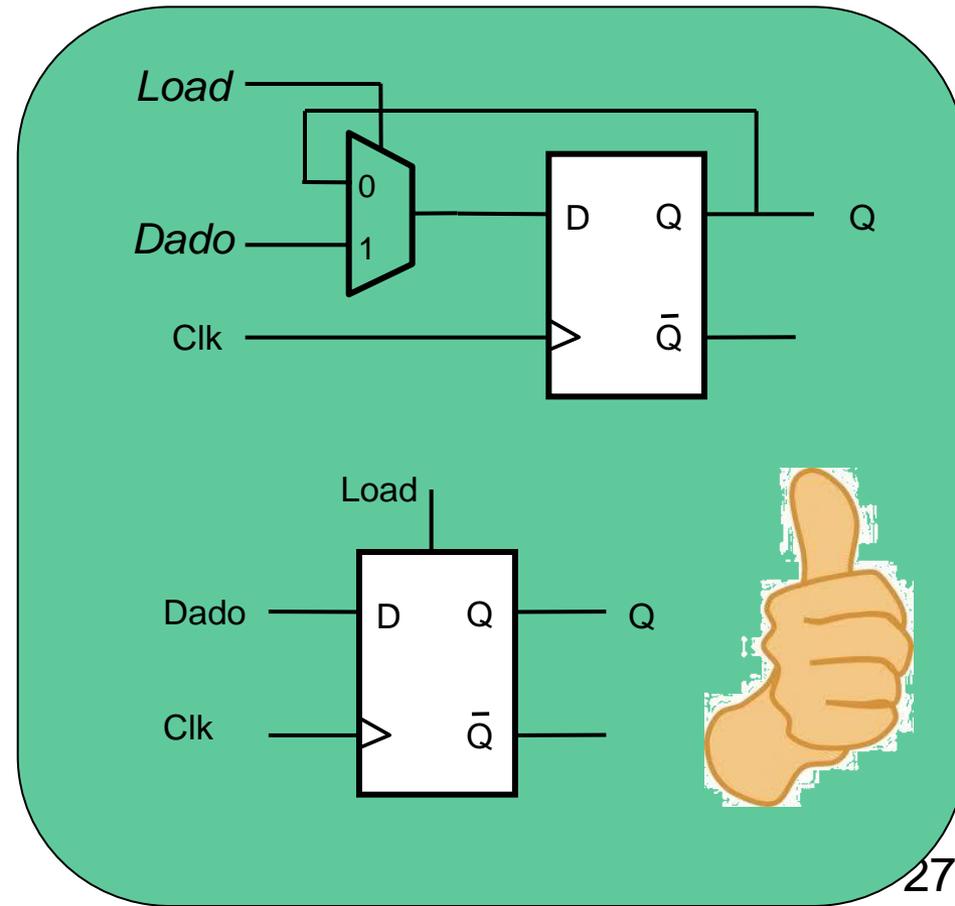
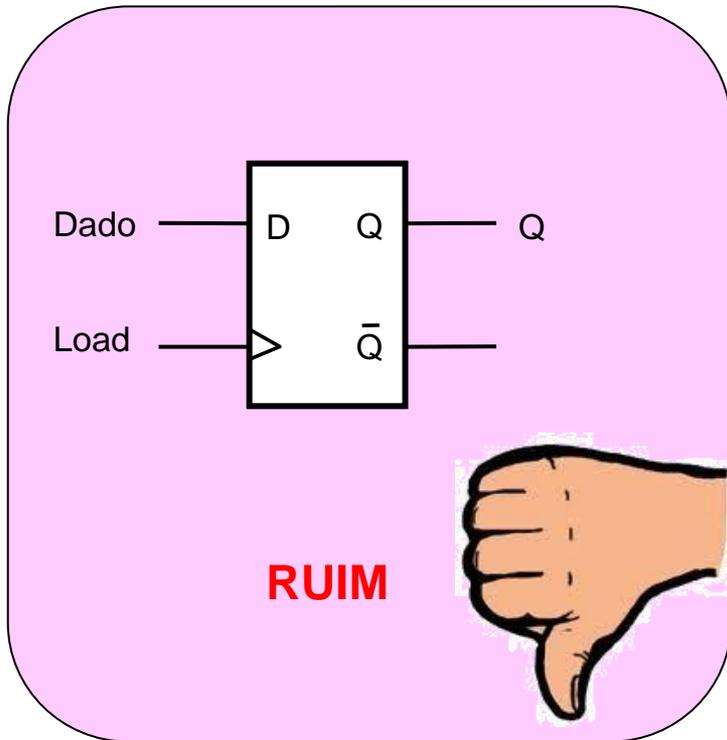


Símbolo ANSI/IEEE Std 91



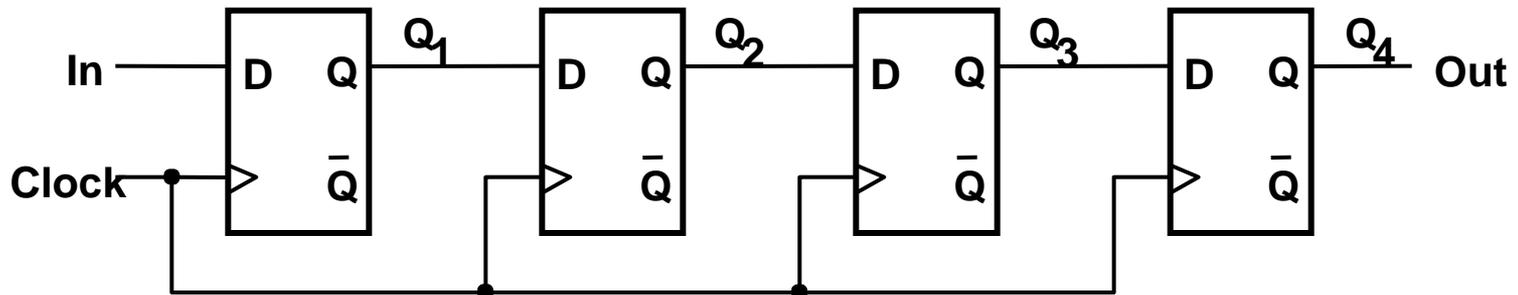
# Registrador (Flip-Flop) com enable

- O registrador apresentado carrega o dado a cada borda do clock
- Desejável carga controlada por sinal “load”



# Shift Register – Registrador de deslocamento

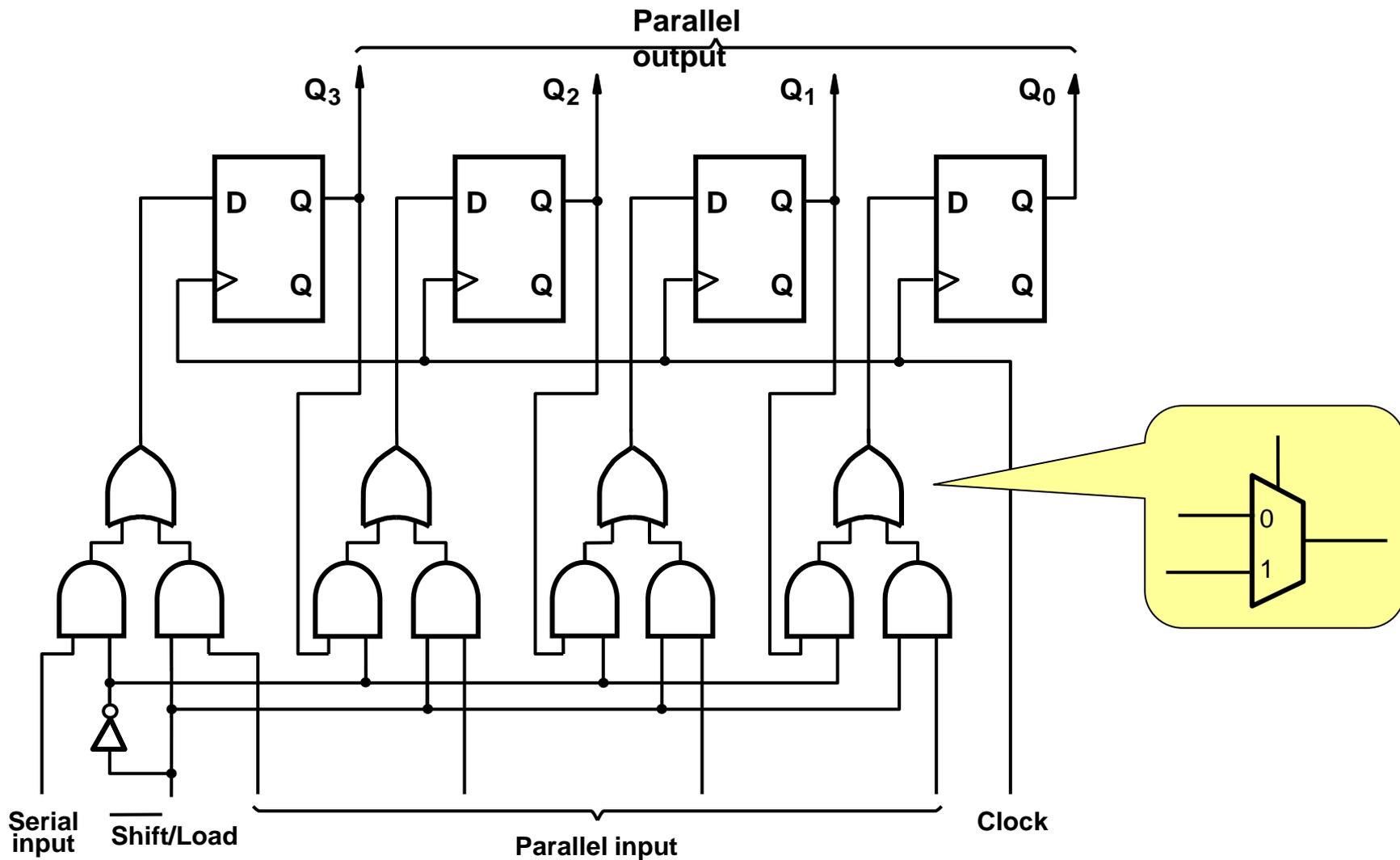
	In	Q <sub>1</sub>	Q <sub>2</sub>	Q <sub>3</sub>	Q <sub>4</sub> = Out
$t_0$	1	0	0	0	0
$t_1$	0	1	0	0	0
$t_2$	1	0	1	0	0
$t_3$	1	1	0	1	0
$t_4$	1	1	1	0	1
$t_5$	0	1	1	1	0
$t_6$	0	0	1	1	1
$t_7$	0	0	0	1	1



# Usos do registrador de deslocamento

- Conversão série  $\rightarrow$  paralelo
- Conversão paralelo  $\rightarrow$  série
  - (precisa de carga paralela)
- Multiplicador / divisor por potência de 2

# Shift Register com Carga Paralela



# Shift Register Universal

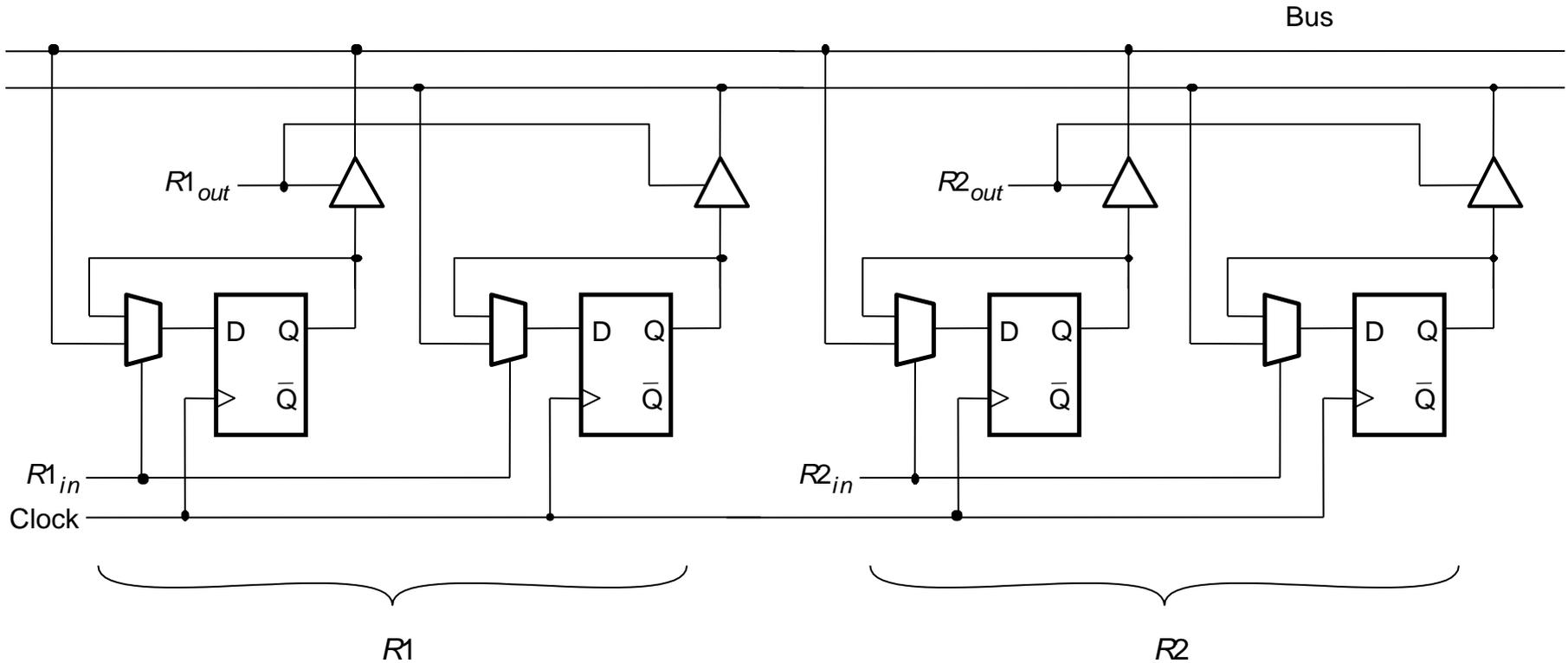
- Entrada Serial
  - Deslocamento a Esquerda
  - Deslocamento a Direita
- Carga Paralela
- Saída Paralela

## Exercício:

Diagrama do Shift Register Universal de 4 bits com entradas de controle: NOP (mantém valor), SHL (esquerda), SHR (direita), LD (carga paralela)

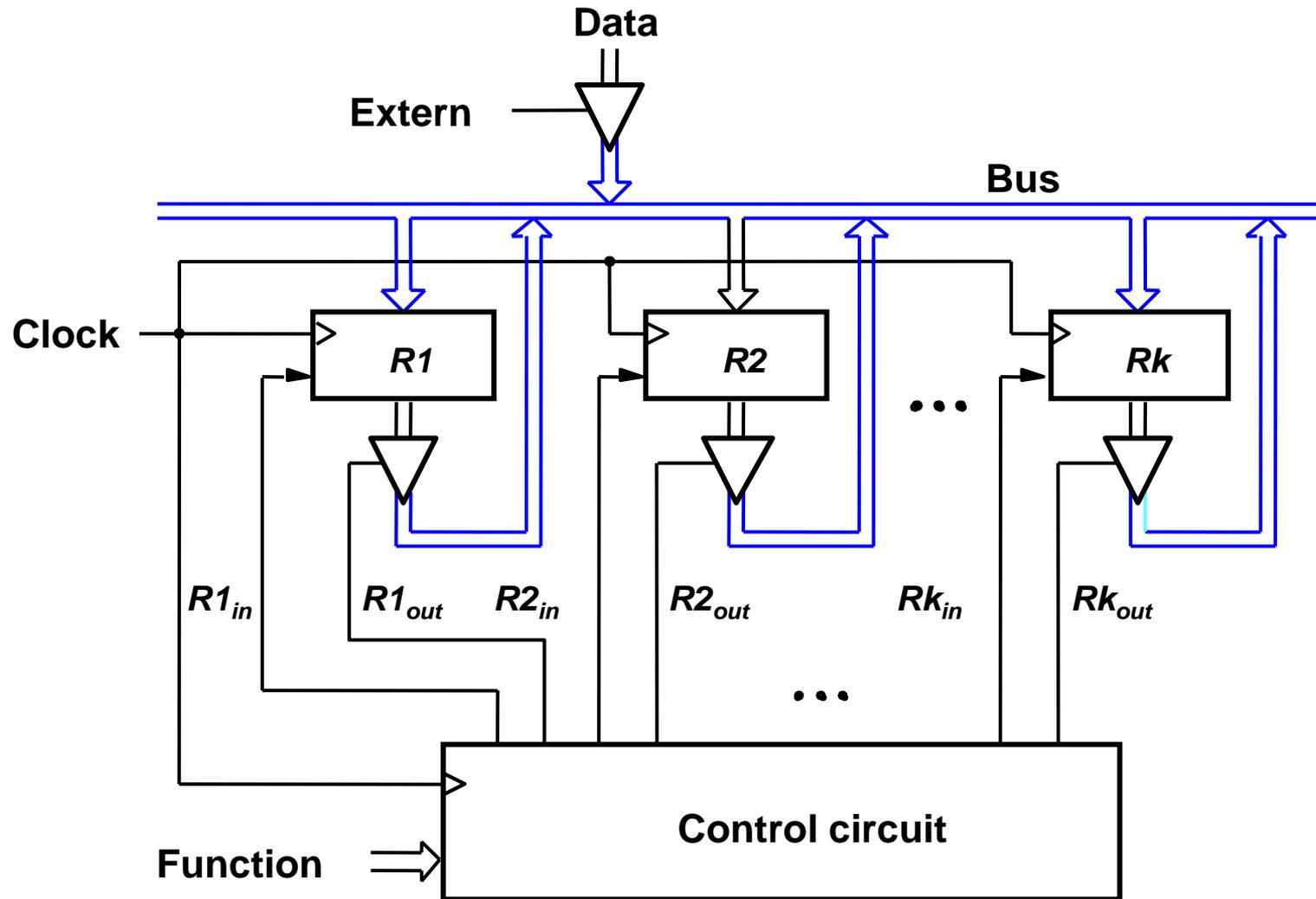


# Registadores em um Barramento (3ST)

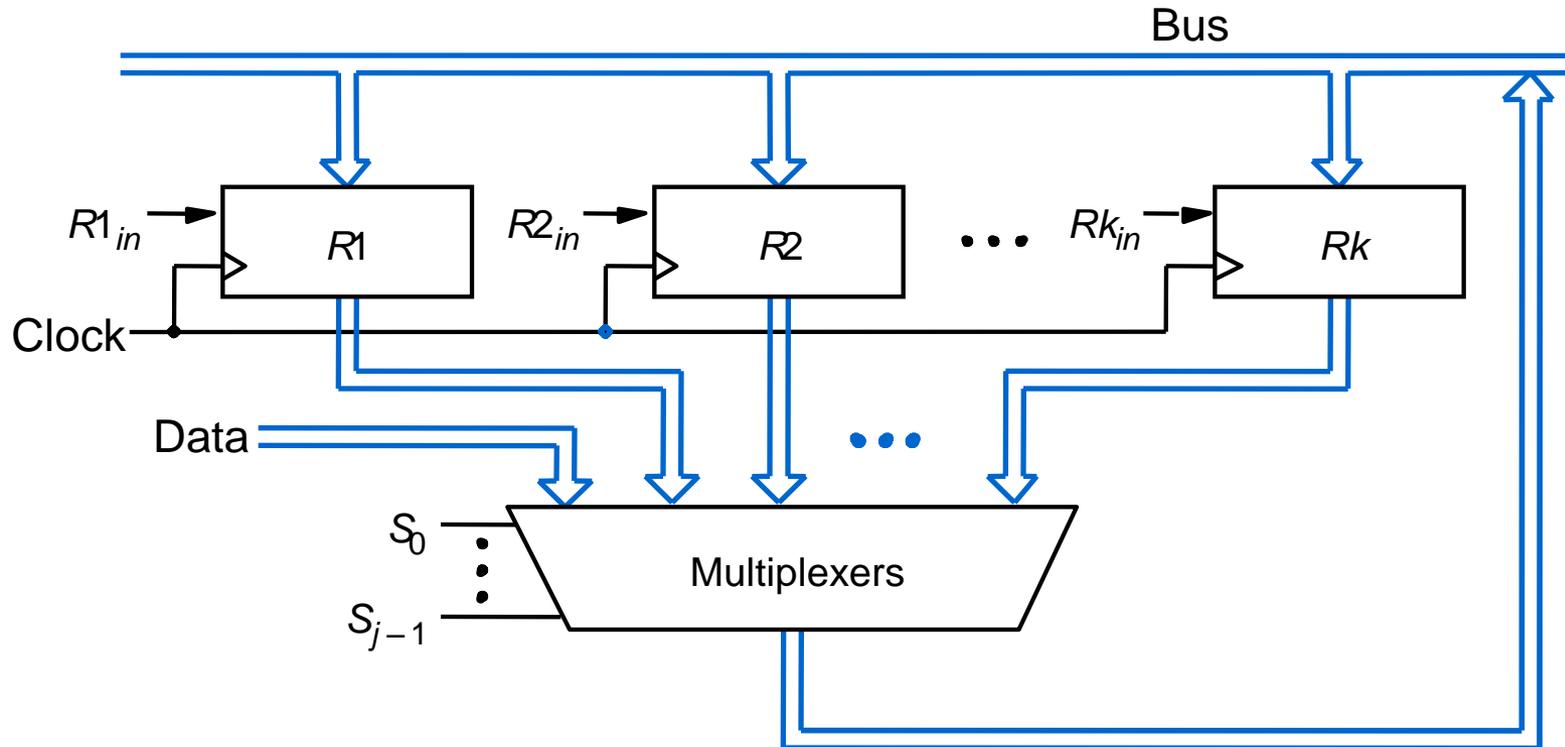


- Dois registradores (R1 e R2) de 2 bits cada ligados a um barramento
- Operação  $R2 \rightarrow R1$ : ativar sinais  $R2_{out}$  e  $R1_{in}$
- Transferência completada na borda de subida do clk

# Registadores em um Barramento



# Barramento com MUX



- “Escrita” no barramento (inclusive entrada externa Data) → saído do MUX
- Função semelhante à implementação 3ST

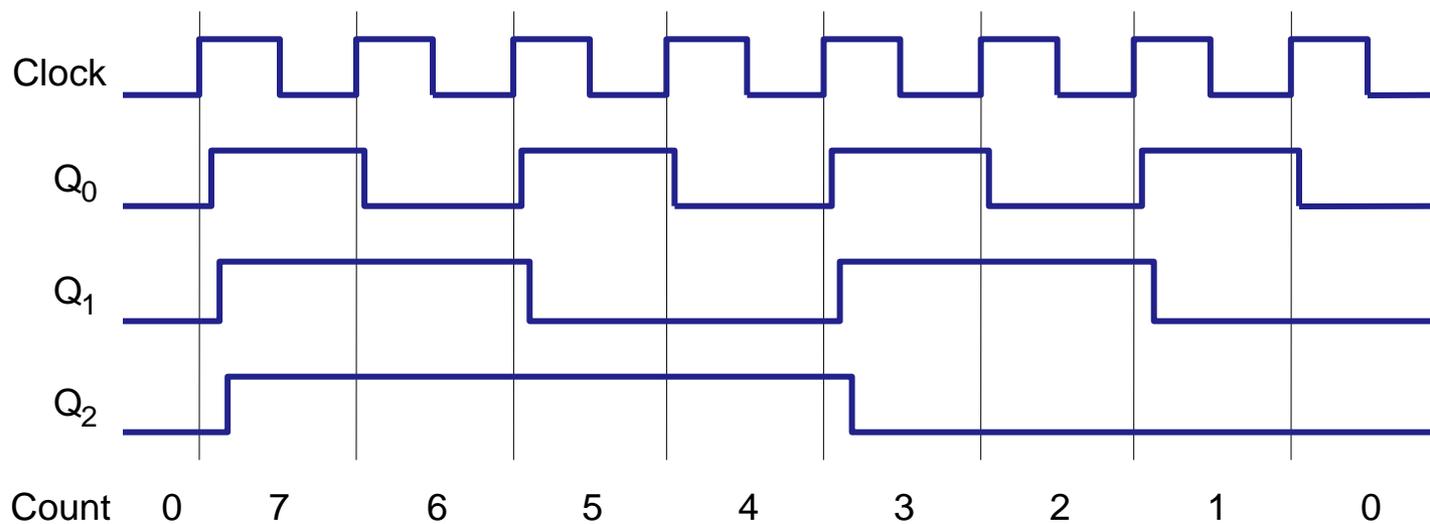
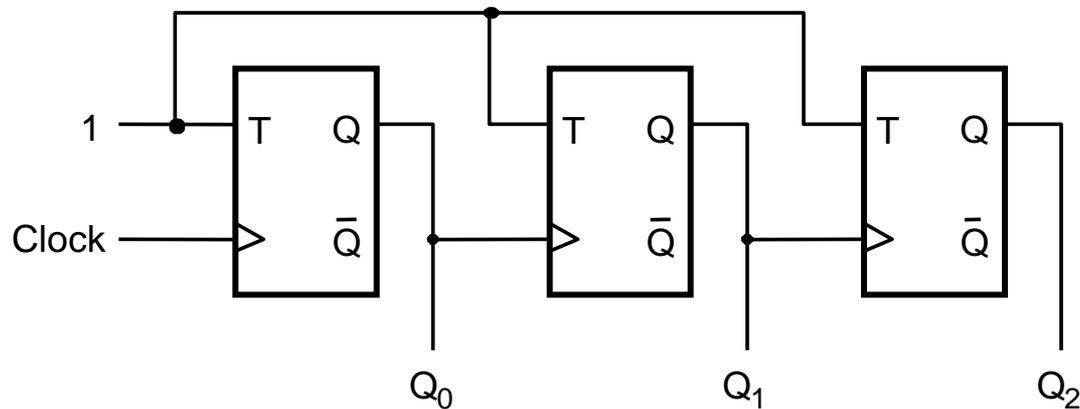
# Tópicos de Contadores

- Contadores síncronos e assíncronos
- Contadores de módulo configurável
- Contadores em anel e Johnson
- Preset e Clear síncronos e assíncronos

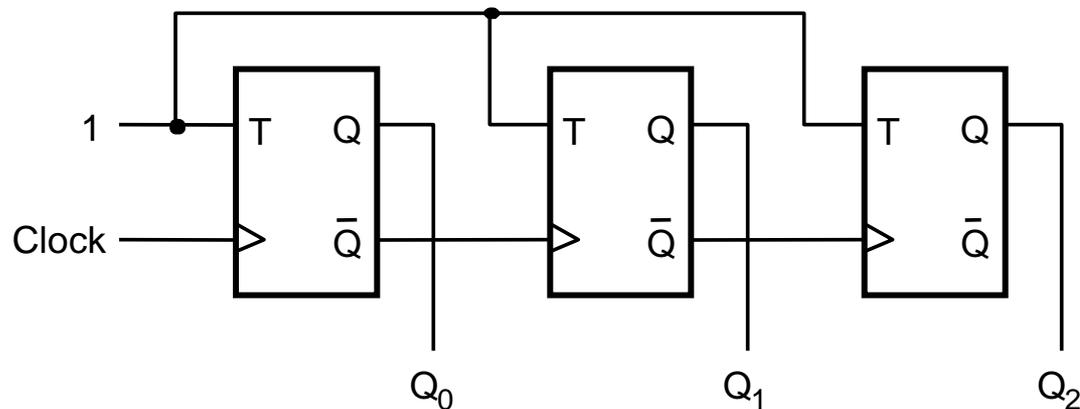
# Contadores síncronos / assíncronos

- Contadores assíncronos
  - Entrada de clock dos FFs recebe saída de estágios anteriores
  - Estado do contador: transições dos estágios não simultâneas (em ripple)
  - Circuito mínimo mas requer cuidados com decodificação
- Contadores síncronos
  - Entrada de clock dos FF recebe apenas sinal externo de clock
  - Estado do contador: transições sincronizadas (razoavelmente simultâneas)

# Contador assíncrono: 3 bits DOWN

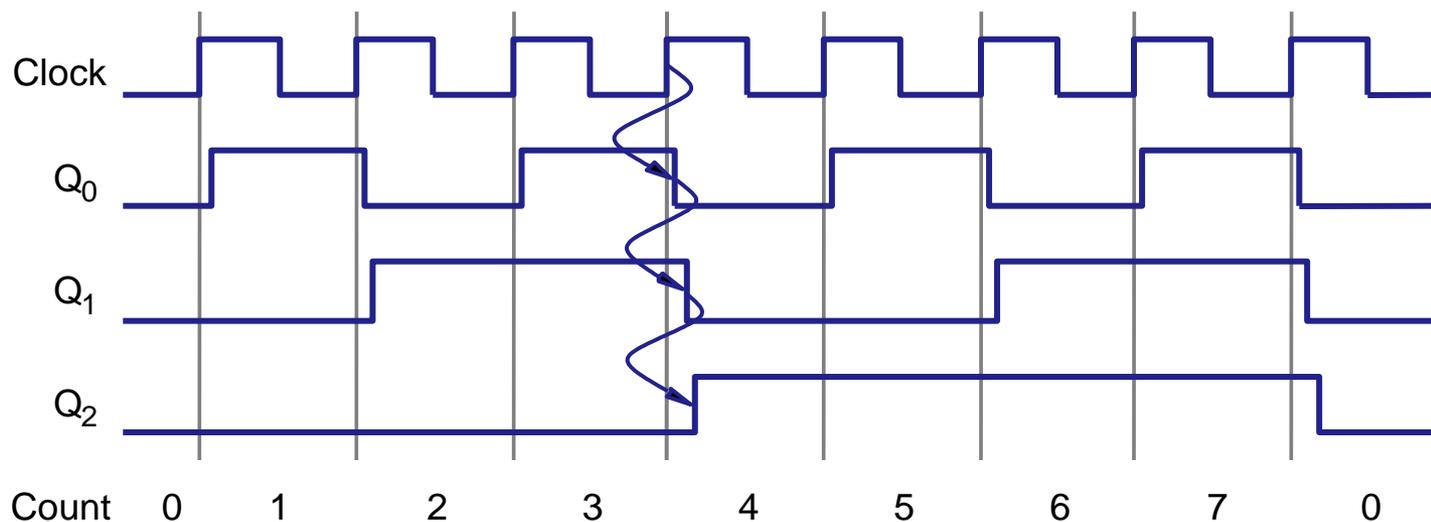


# Contador assíncrono: 3 bits UP



Alternativas para UP/DOWN:

- FF sens. borda de descida
- saída =  $\sim Q$



Como fazer um contador Up/Down?

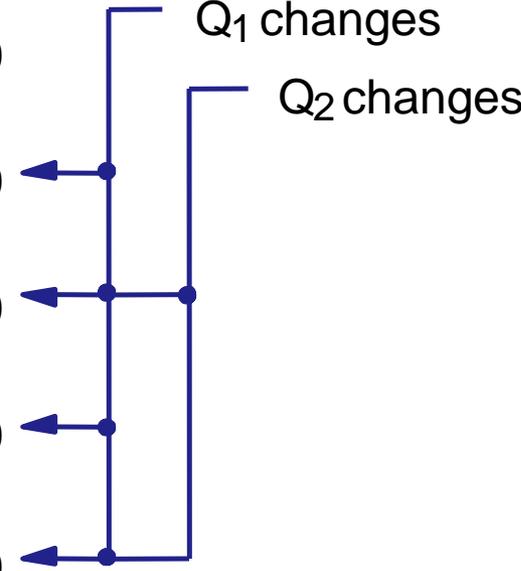


# Problemas com contadores assíncronos

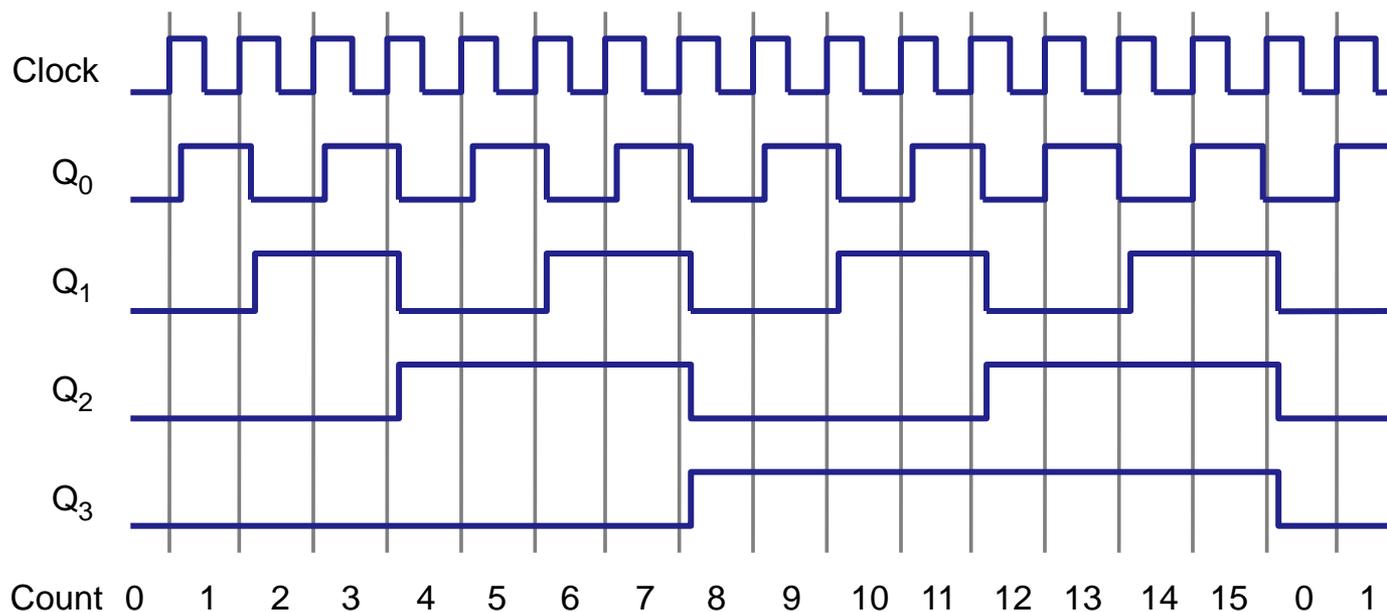
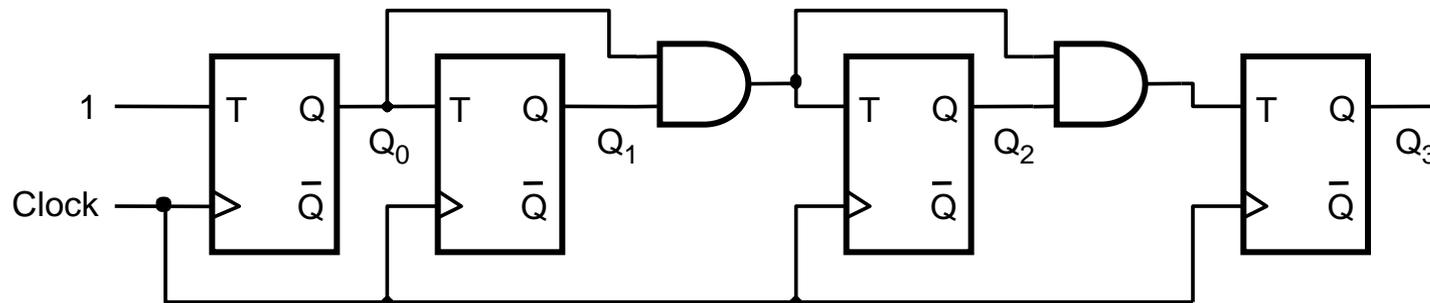
- Saídas dos FFs (estado do contador) com atrasos cumulativos LSB  $\rightarrow$  MSB
  - ver simulação em modo gráfico (vwf) e tabular (tbl)
- Se o estado do contador está sendo decodificado para detectar valor de contagem que dispara alguma ação (ex: reset)  $\rightarrow$  problema
- Ver análise de timing
- Voltar ao assunto: contador módulo N

# Projeto de contador síncrono: 3 bits

Clock cycle	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>
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

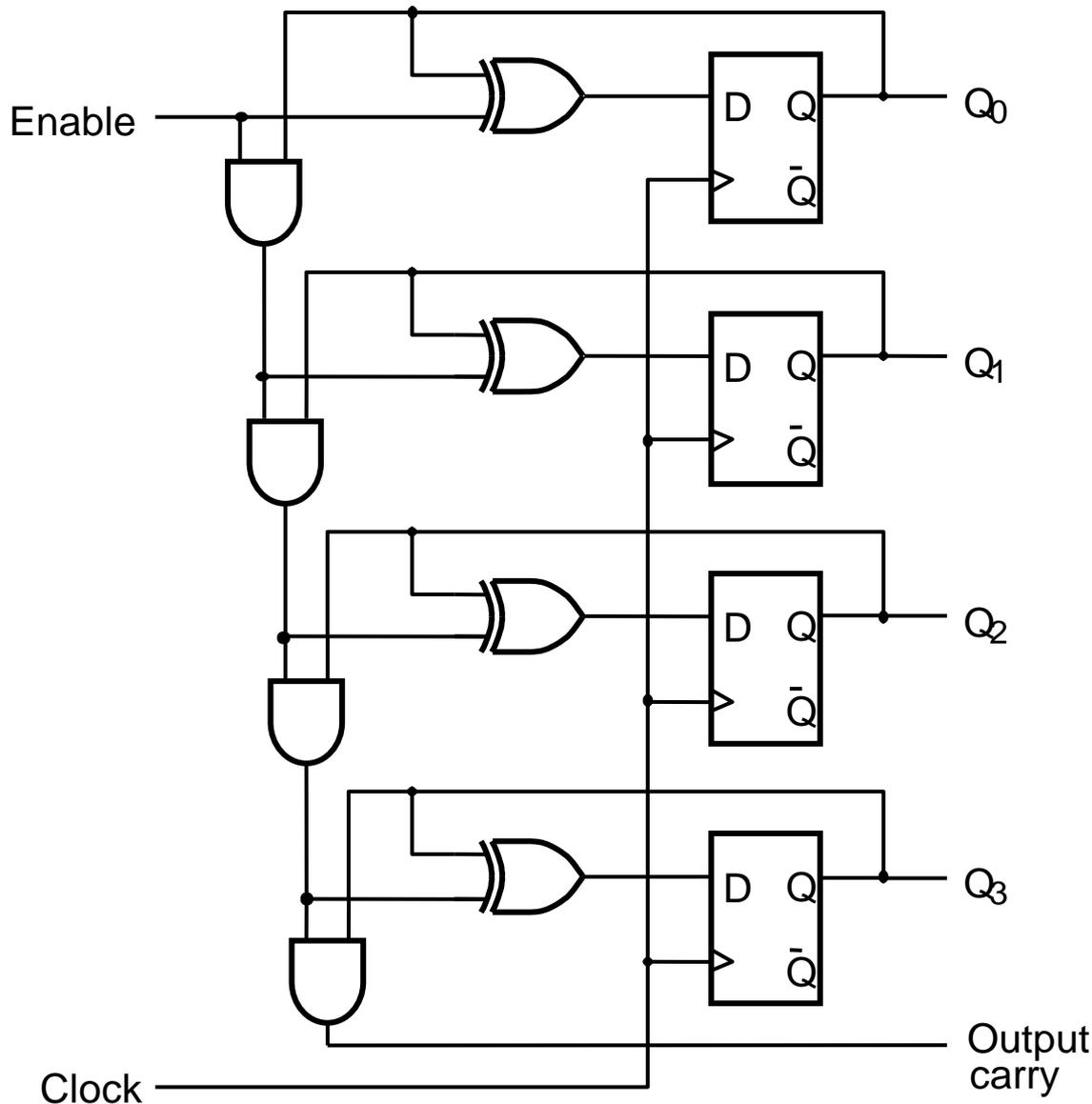
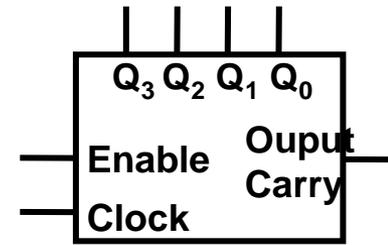


# Contador (up) síncrono: 4 bits





# Contador síncrono com FF D



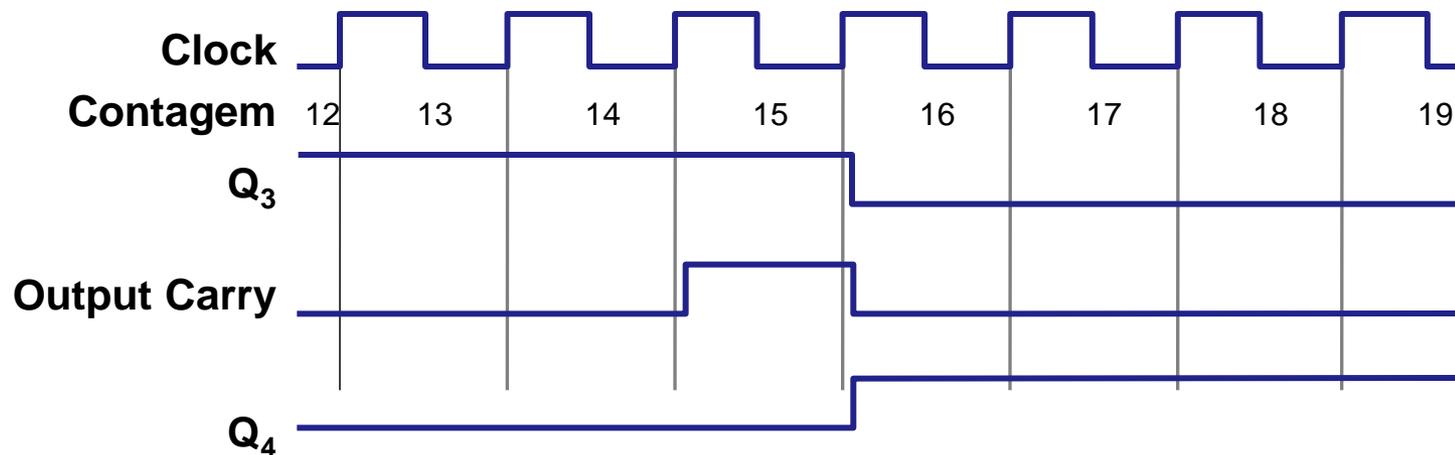
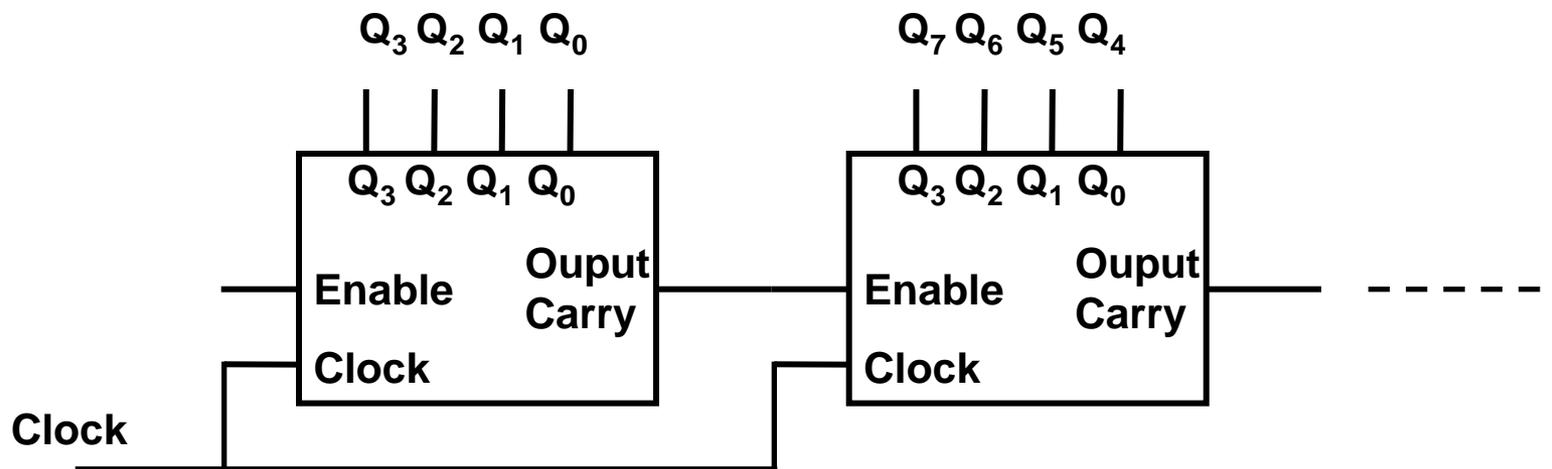
$$D_0 = \overline{Q_0} = 1 \oplus Q_0$$

$$D_1 = Q_1 \oplus Q_0$$

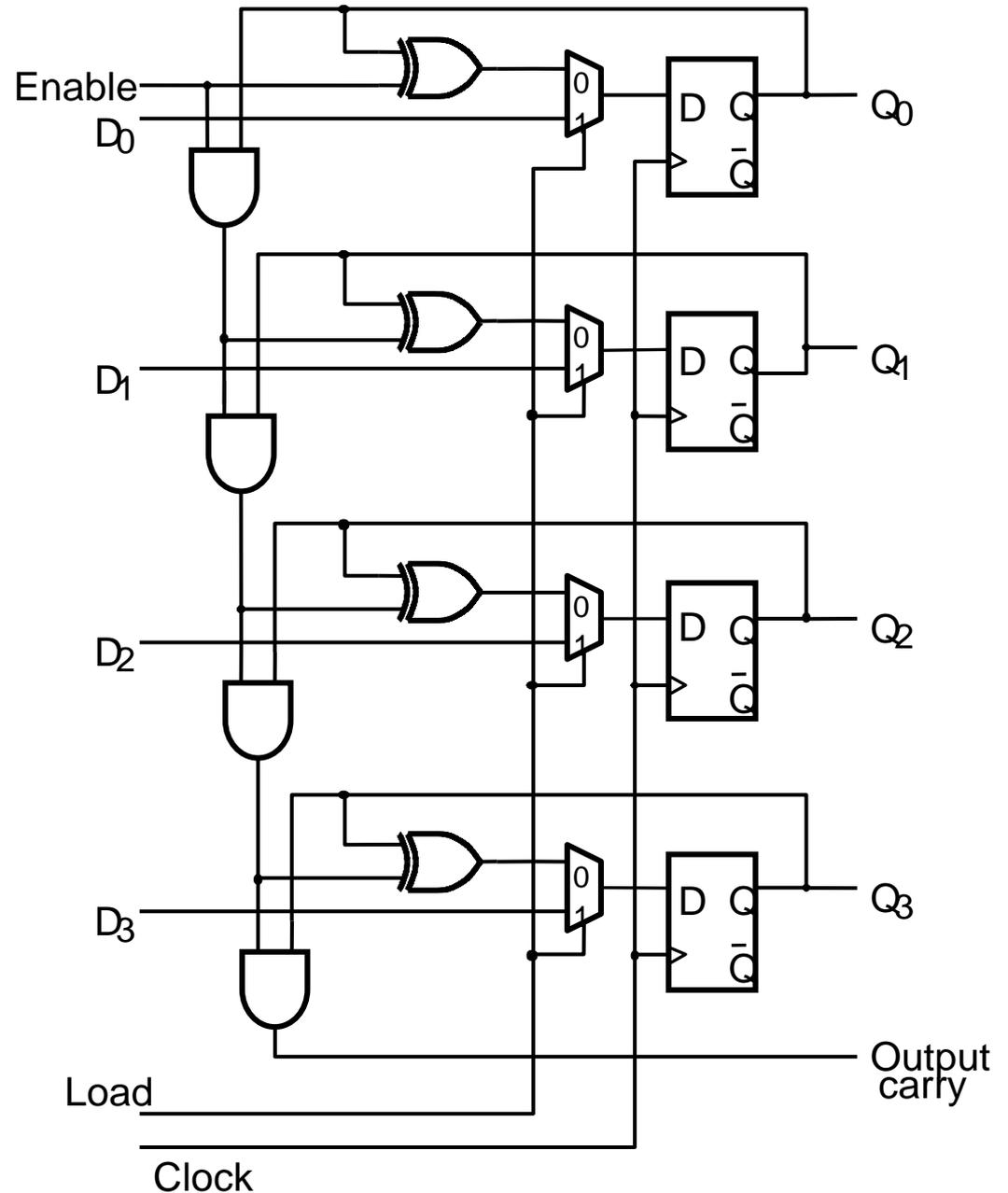
$$D_2 = Q_2 \oplus Q_1 Q_0$$

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

# Contadores em cascata



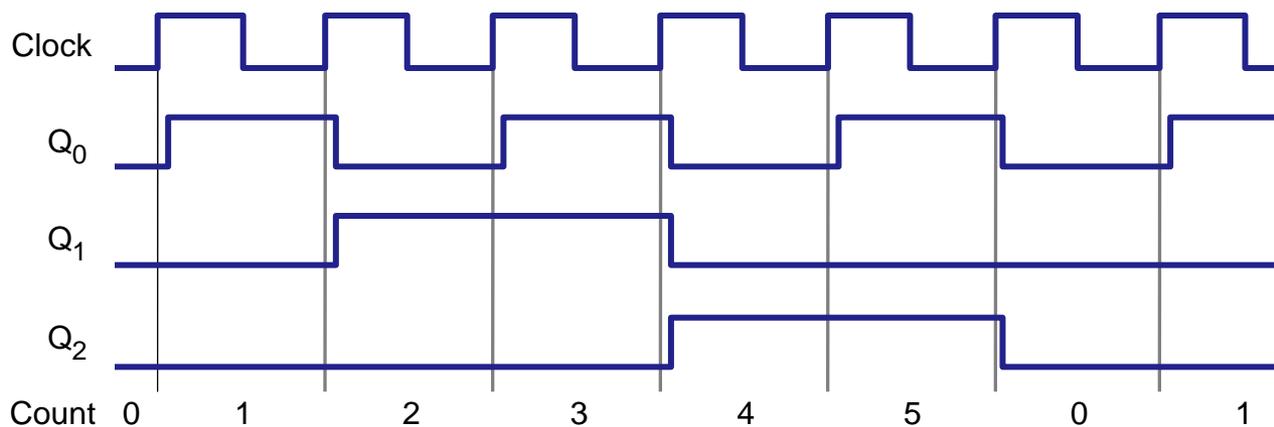
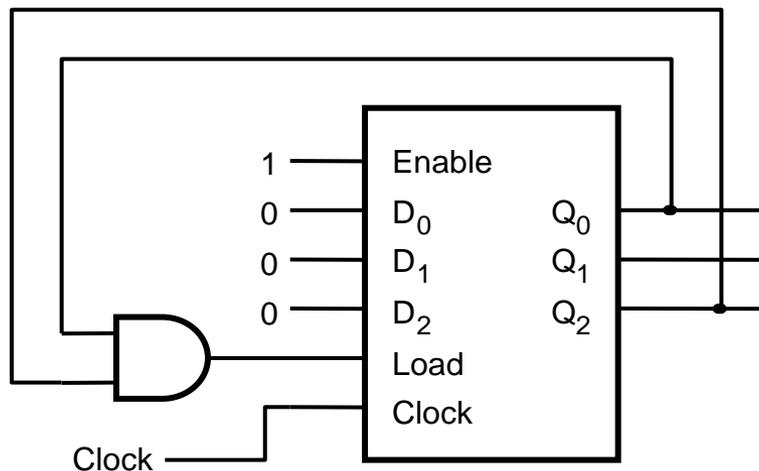
# Inclusão de Load



# Contadores de módulo $\neq 2^n$

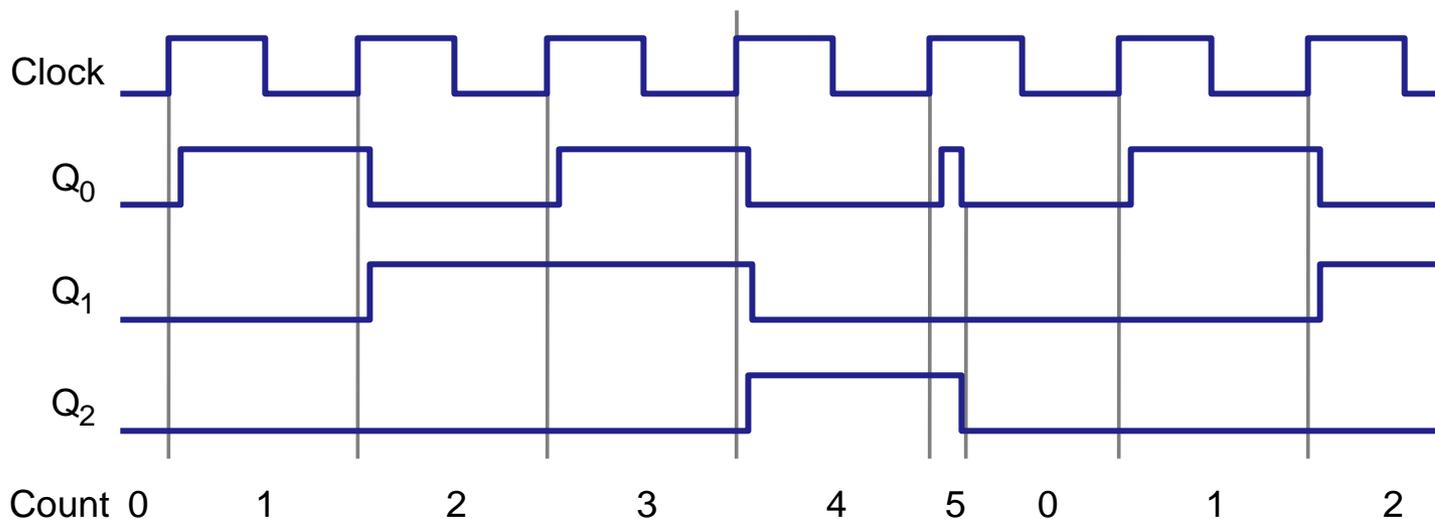
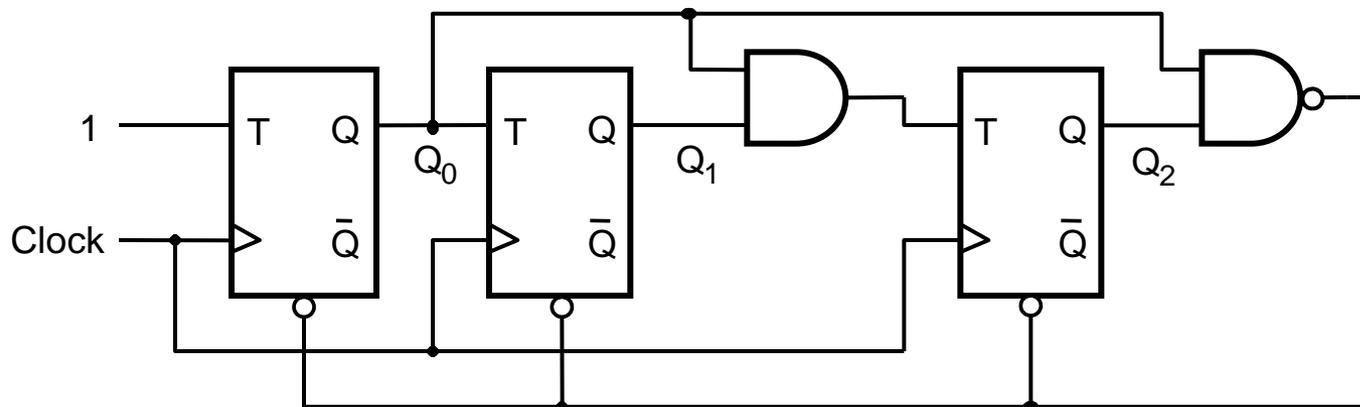
- Algumas possibilidades para módulo M:
  - Com o uso de clear/preset
    - Contador UP:
      - ao atingir valor = M  $\rightarrow$  clear
    - Contador DOWN:
      - ao atingir valor =  $2^n - M \rightarrow$  preset (1111111)
  - Com o uso de Load paralelo
    - Contador UP:
      - ao atingir contagem máxima  $\rightarrow$  load valor ( $2^n - M$ )
    - Contador DOWN:
      - ao atingir contagem = 0  $\rightarrow$  load valor M
- **ATENÇÃO:** projeto muda se os controles clear/preset/load são síncronos ou assíncronos

# Contador mod-6 com Load síncrono



Obs: o load síncrono de 000 faz o que seria um reset síncrono

# Contador mod-5 com Reset assíncrono



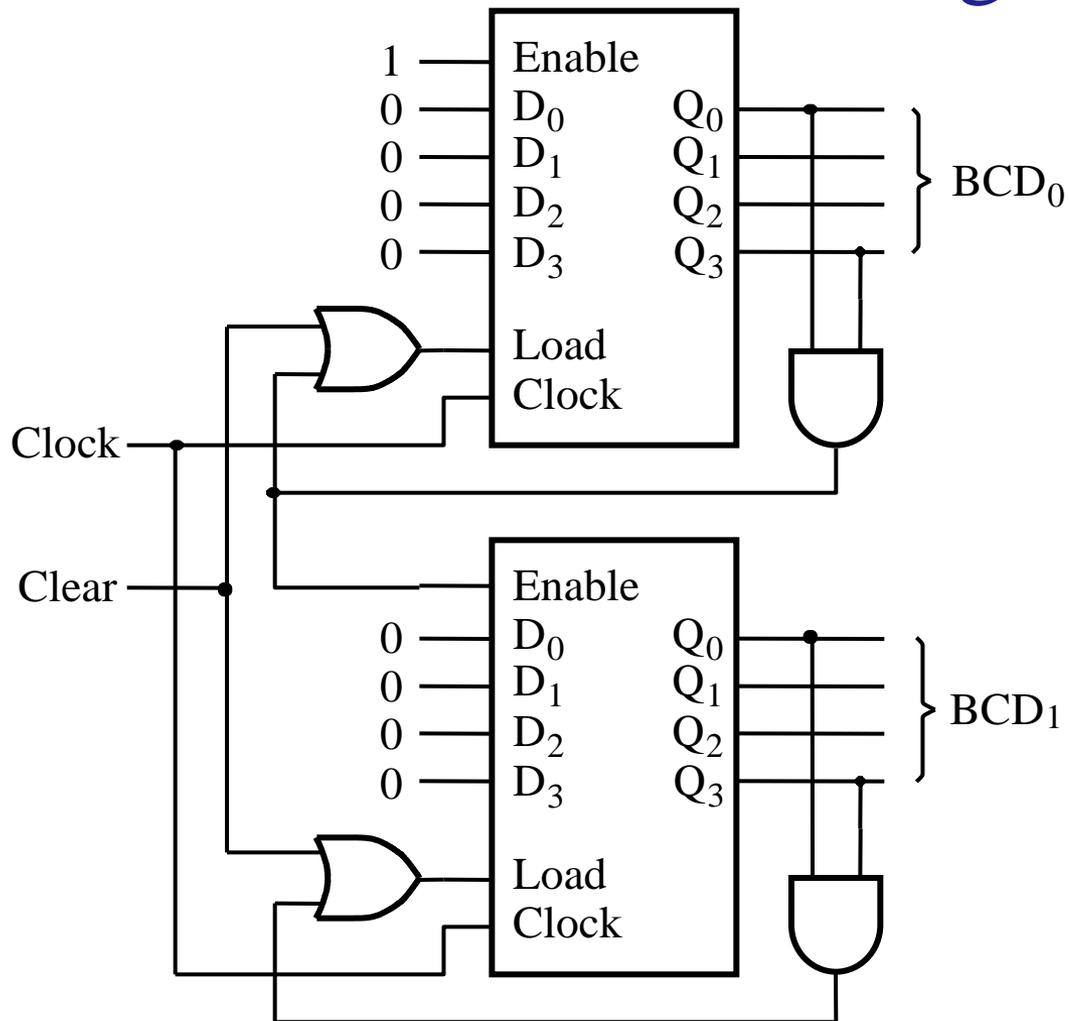
Obs: apesar do estado final a ser detectado ser o mesmo do caso anterior, estado 101 (5), o módulo deste contador é 5 (contagem 0 1 2 3 4 0 1 2 3 4)



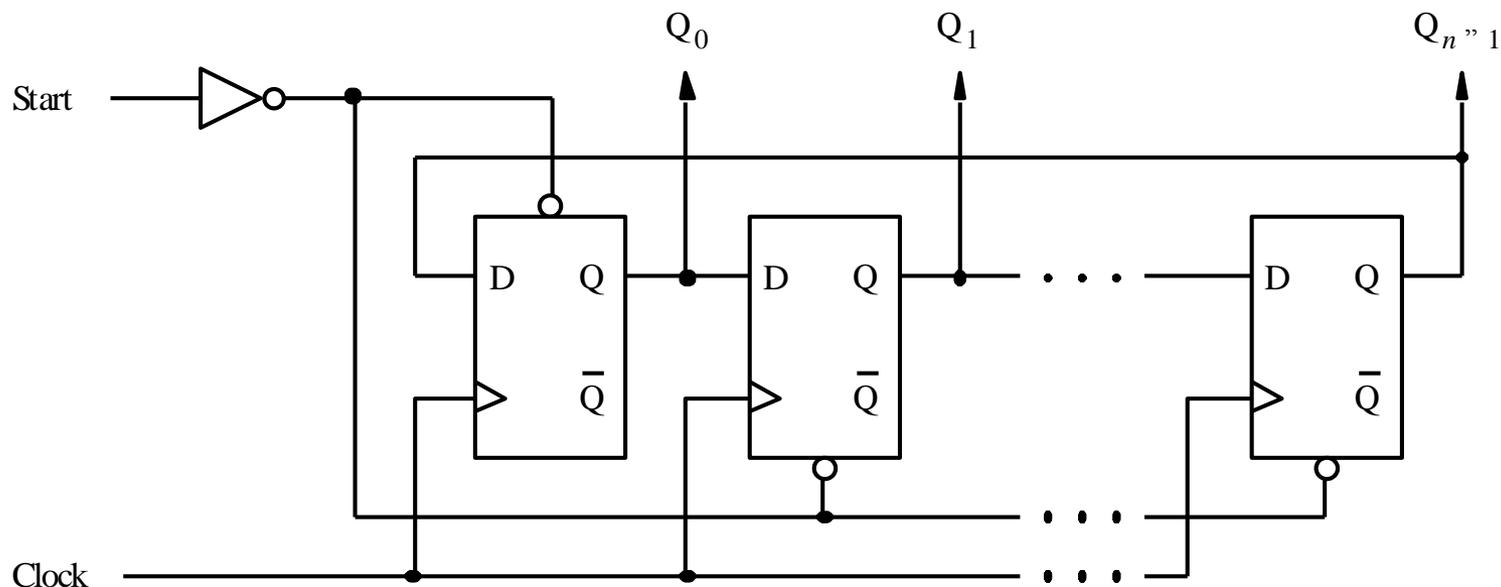
# Problemas com contadores assíncronos

- Como as saídas dos flip-flops são desalinhadas no tempo, vários estados intermediários podem ocorrer
  - ver simulação em modo gráfico (vwf) e tabular (tbl)
- A decodificação do estado desejado para fins de acionar clear/preset/load assíncronos pode acontecer de forma espúria durante estados transientes.
- Uso de clear/preset/load síncronos resolvem o problema

# Contador BCD de 2 dígitos



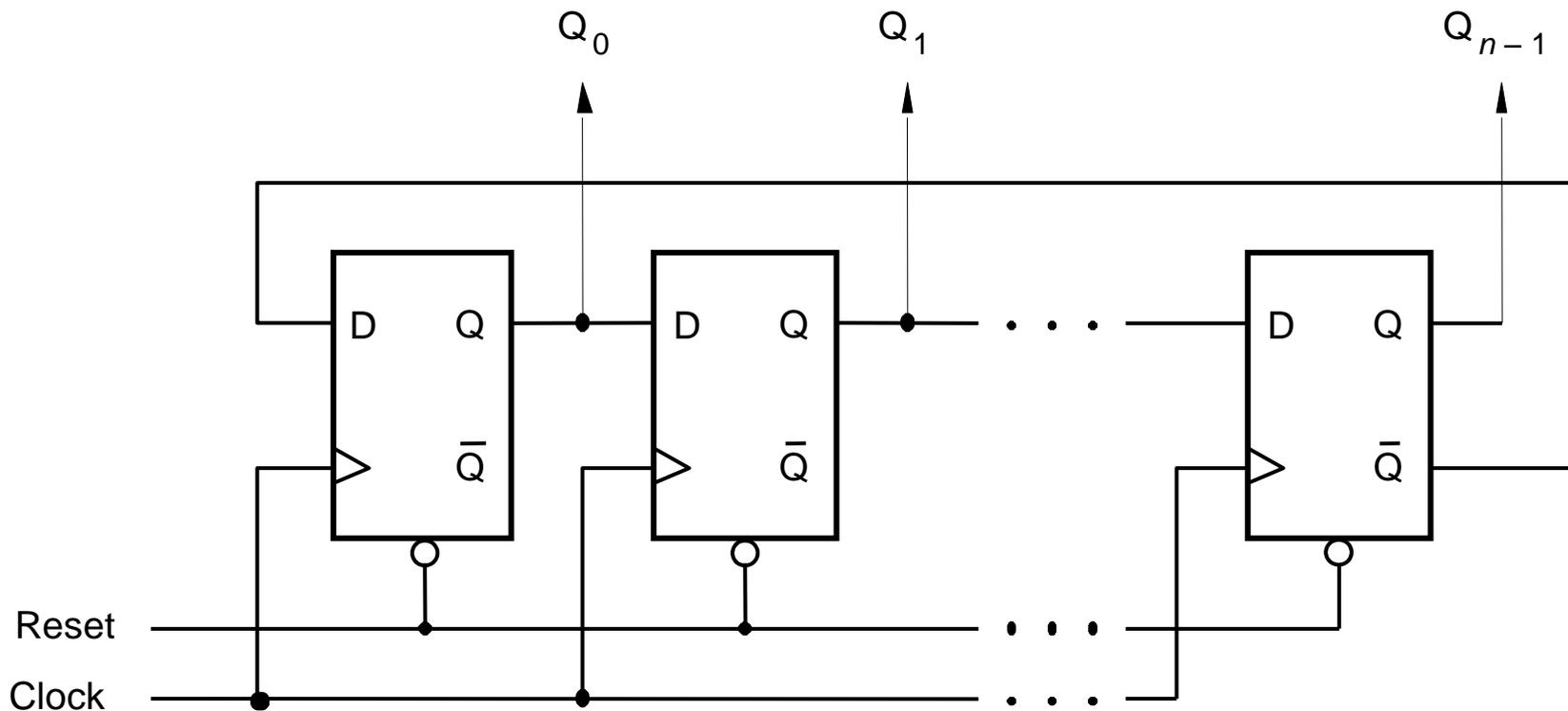
# Contador em anel



Contagem p 4 bits: 1000 0100 0010 0001 1000 0100 0010 0001 ....

Módulo?

# Contador Johnson



Para um contador de 4 estágios:

- Qual é a sequência de contagem
- Qual é o módulo do contador?