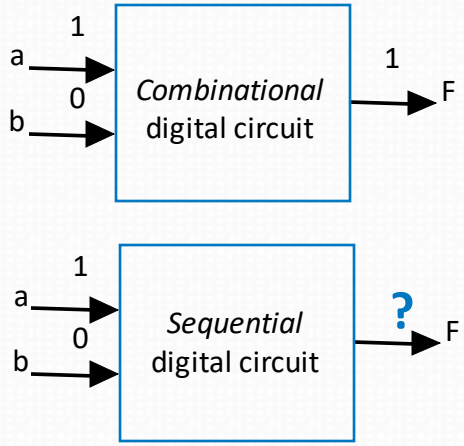
Circuitos Sequenciais

Prof. Thiago Felski Pereira, MSc.

Adaptado: Frank Vahid (Digital Design) e Paulo Roberto Valim

Introdução

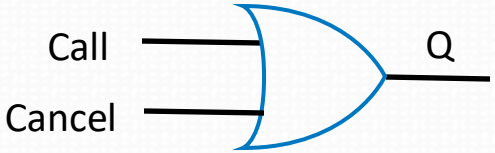
- Circuitos Sequenciais
  - A saída depente não apenas das entradas atuais (como em um circuito combinacional), mas nas sequências passadas das entradas
    - Armazenamento de bits, também conhecido como ter um estado
  - Exemplo: um circuito contador binário



Devemos saber a sequência passada para saber a saída

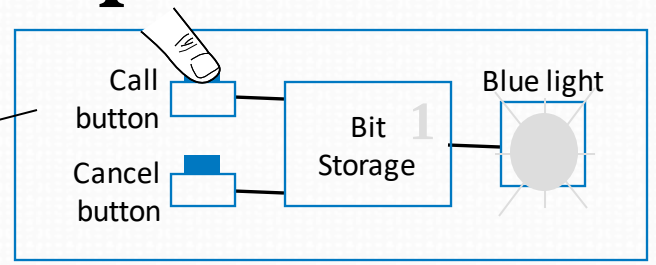
Armazenando 1 bit - Flip-Flops

- **Exemplo que requer armazenamento**
- Chamar comissária de bordo
  - Apertar chamar: luz acende
    - ***Permanece acesa*** *após soltar o botão*
  - Apertar cancelar: luz apaga
    - Permanece apagada após soltar o botão
  - Uma porta lógica resolve esse problema?

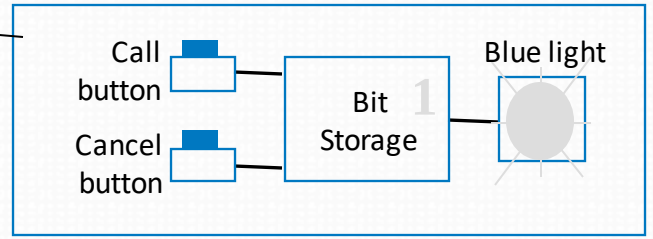


Não funciona. Q=1 quando Call=1, mas não permanece em um quando solto

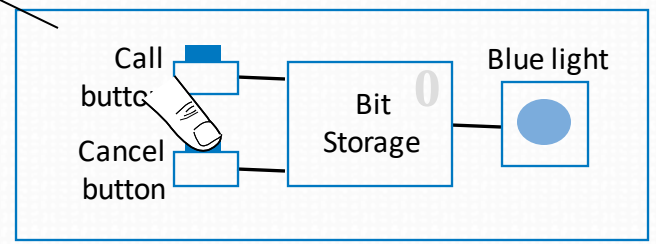
Precisa de uma forma de lembrar



1. Call button pressed – light turns on



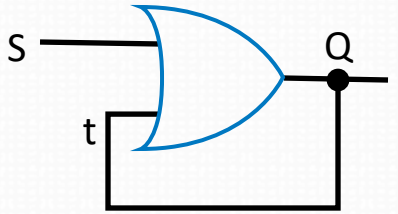
2. Call button released – light ***stays on***

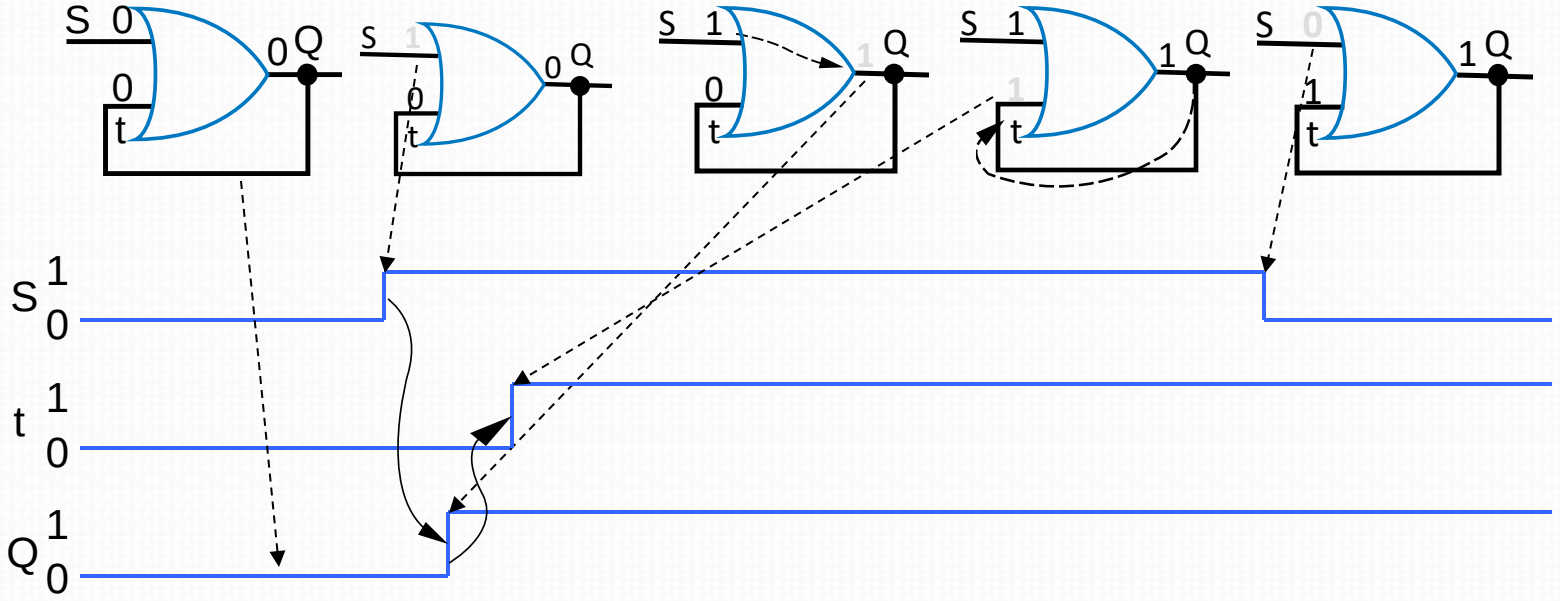


3. Cancel button pressed – light turns off

Primeira tentativa de armazenar um bit

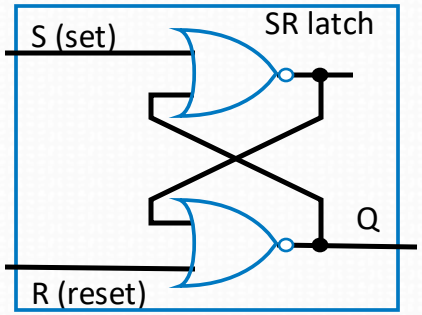
- Necessidade de armazenamento
  - O circuito da direita faz o que precisamos?
    - Não: Assim que Q torna-se 1 (quando S=1), Q fica 1 para sempre – nenhum valor de S pode mudar Q novamente para 0

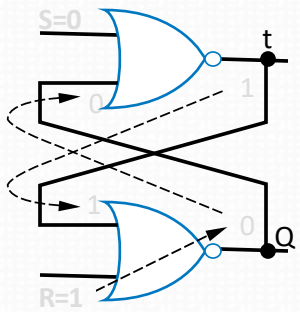


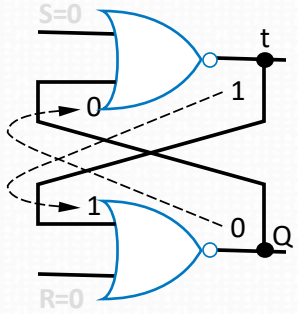


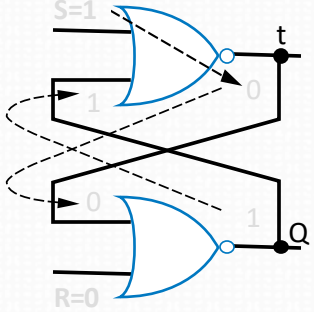
Armazenamento com Latch SR

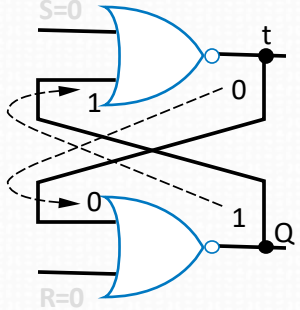
- O circuito à direita, portas NOR cruzadas, faz o que queremos?
  - Sim! Como alguém teve essa ideia? Possivelmente com tentativa e erro e um pouco de lógica.



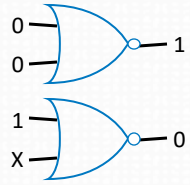


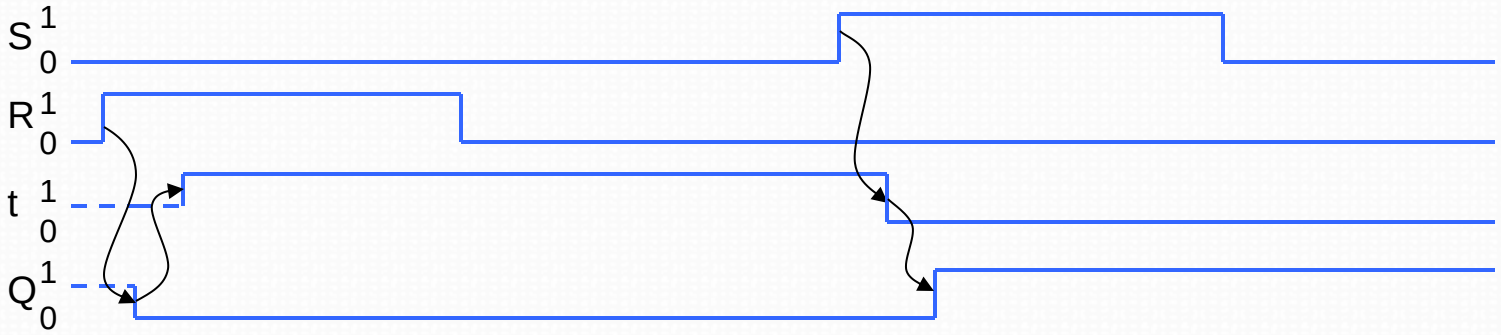






Recall NOR…

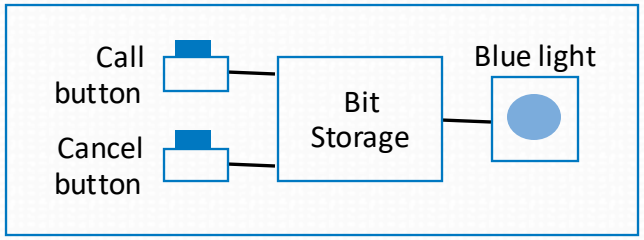


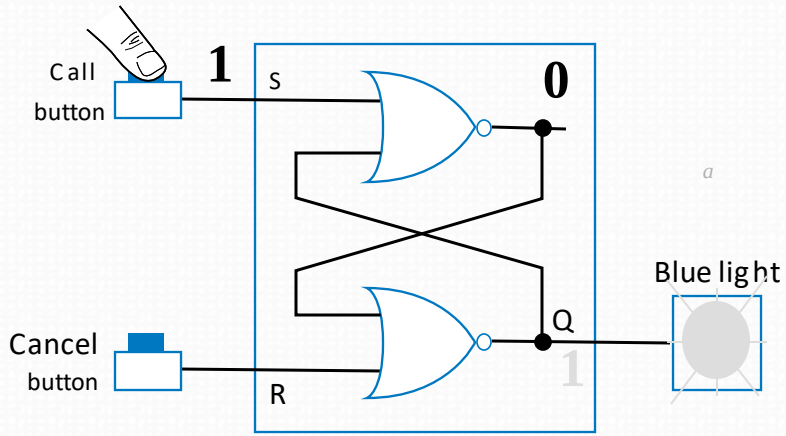


Exemplo usando Latch SR

- O Latch SR pode server como armazenamento para o exemplo anterior do botão de chamada da comissária de voo
  - Call=1 : define Q para 1
    - Q fica em 1 mesmo após Call=0
  - Cancel=1 : reinicia Q para 0

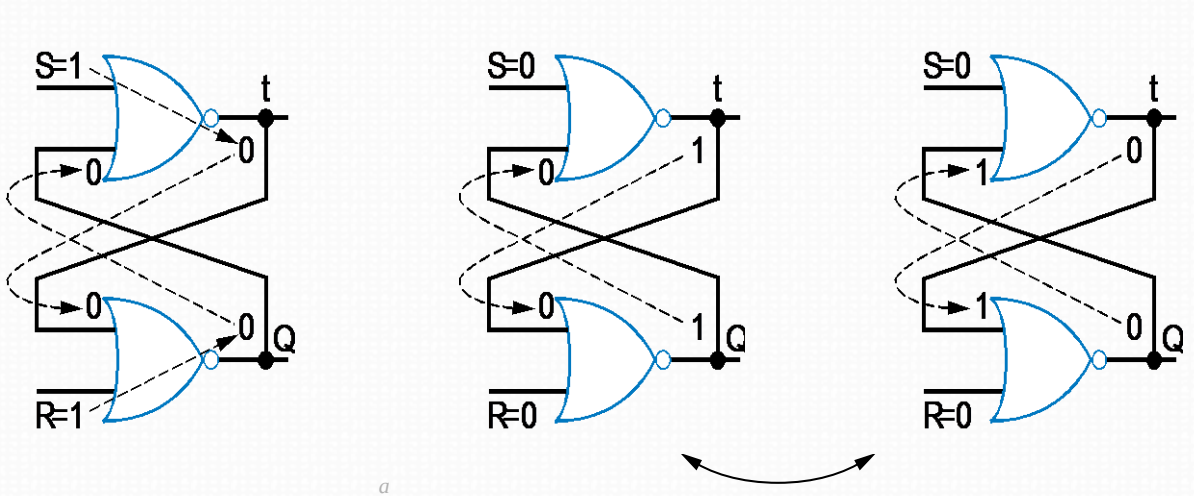
- Mas, existe um problema...

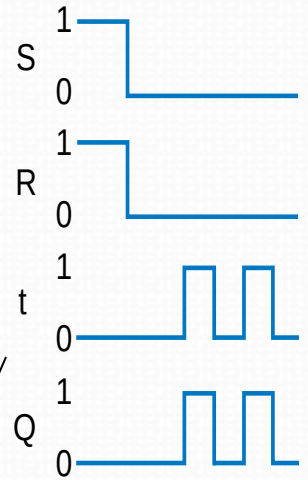




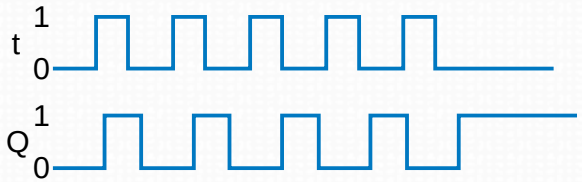
Problema com o Latch SR

- Problema
  - Se S=1 e R=1 simultaneamente, não saberemos que valor Q irá tomar



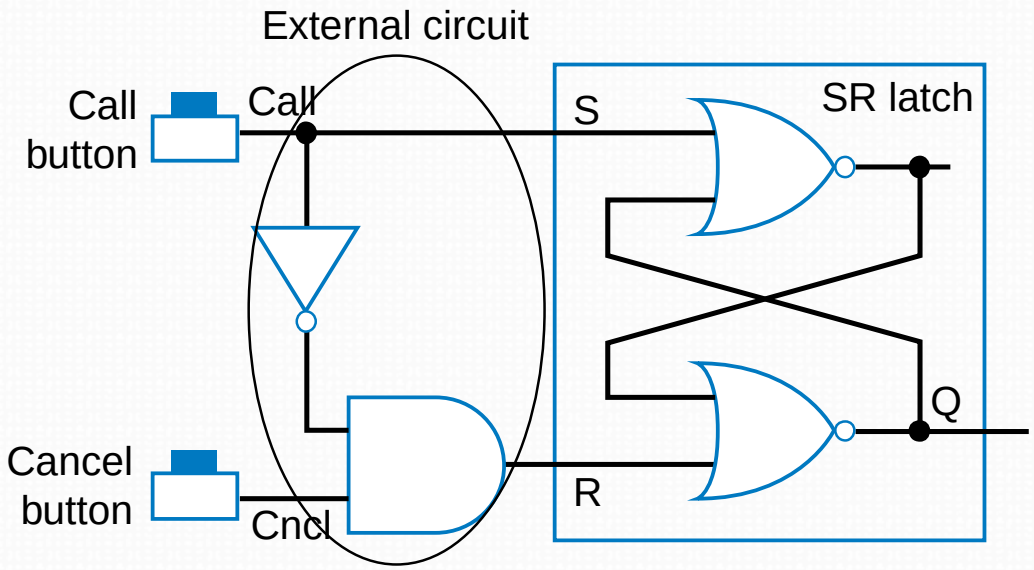


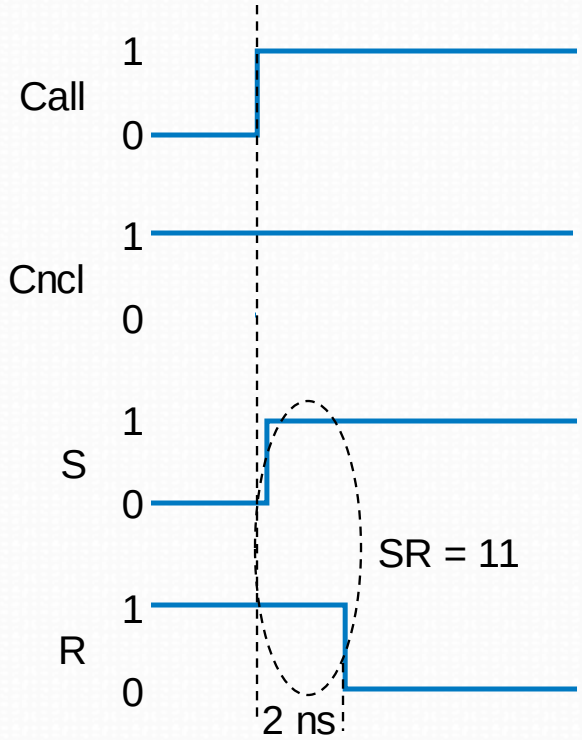
Q pode oscilar. Então, como um caminho será levemente maior do que o outro, Q irá eventualmente ficar em 1 ou zero, mas não saberemos qual deles. Conhecido como *condição de corrida*.



Problema com o Latch SR

- O projetista pode tentar evitar o problema usando um circuito externo
  - O circuito deve proteger SR contra 11
  - Mas 11 pode ocorrer por duas latências diferentes

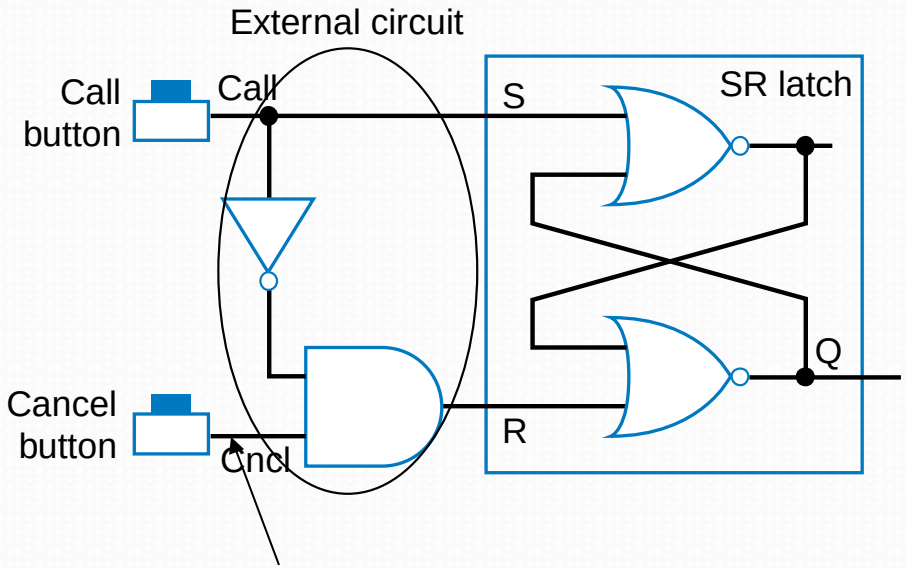




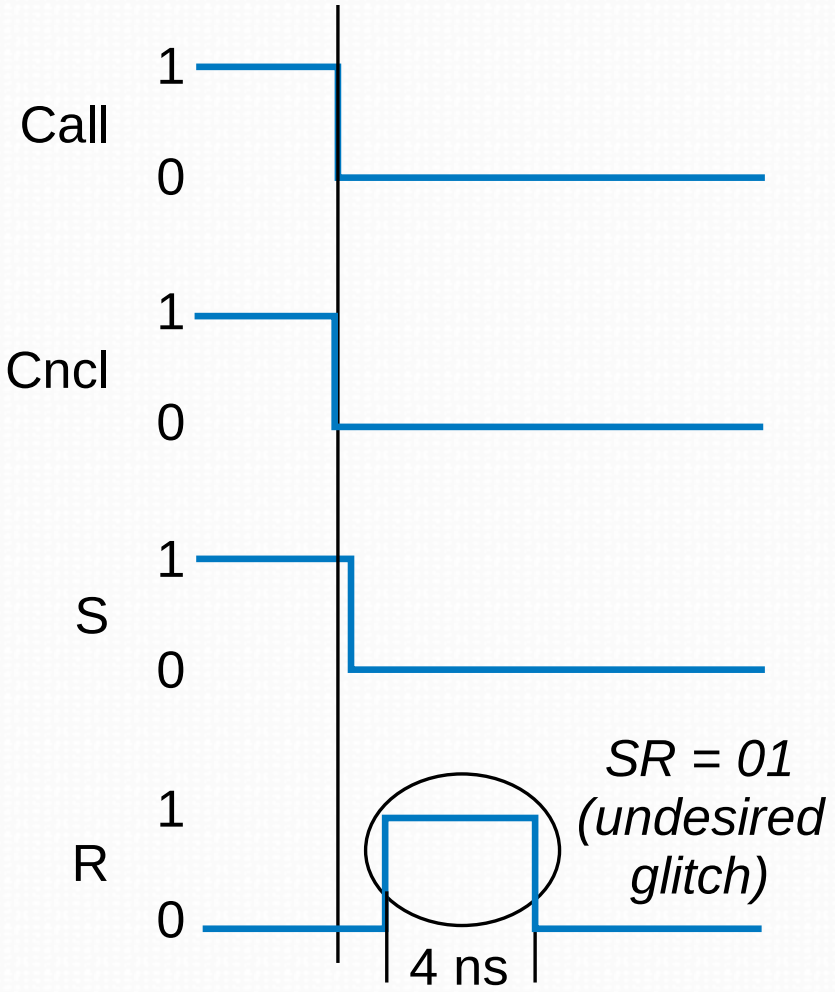
Assumindo 1 ns de latência por porta. O caminho é mais longo de Call para R do que de Call para S, causando 11 por um curto período de tempo – podendo ser longo o bastante para causar a oscilação

Problema com o Latch SR

- Pico(Glitch) pode causar uma mudança indesejada no S ou R

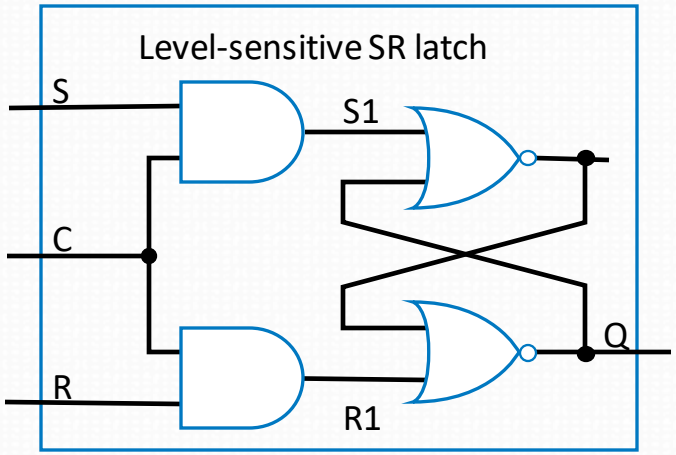


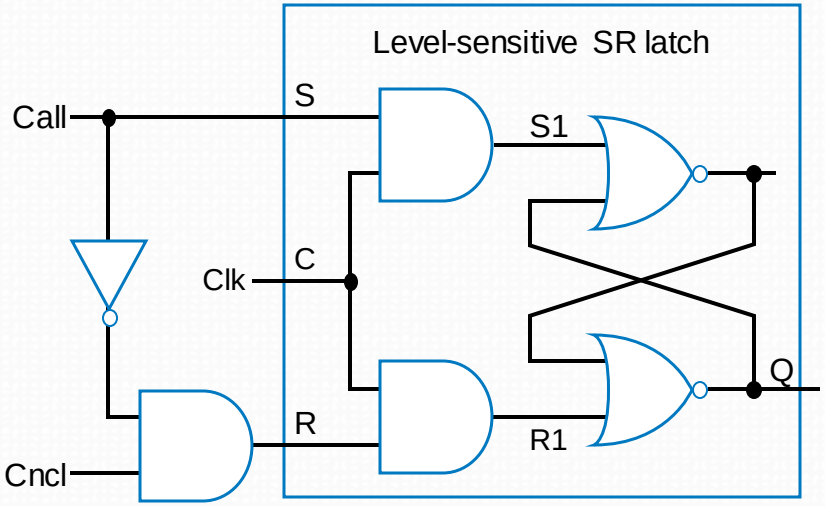
Considere uma latência de 4 ns no fio

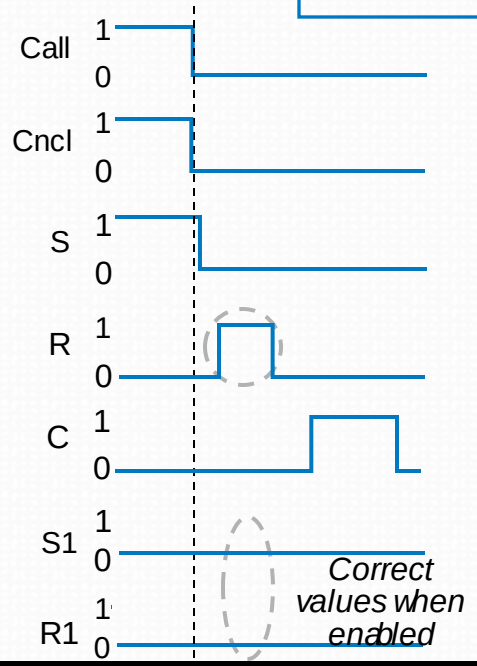


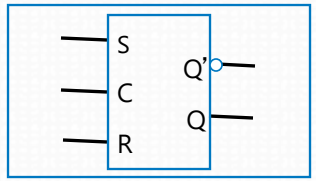
Solução: Latch SR sensível ao nível

- Adiciona-se uma porta de habilitação "C"
- S e R só mudam quando C=0
  - Protege o circuito para nunca ter SR=11, exceto por atrasos de canal
  - Coloca C=1 após S e R ficarem estáveis
  - Quando C vai para 1, os valores estáveis de S e R passam juntos pela porta AND e definem S e R ao mesmo tempo.









Level-sensitive SR latch symbol

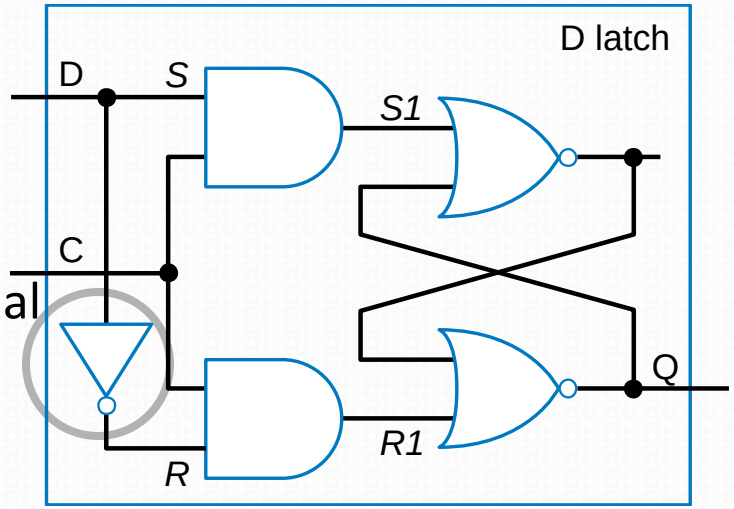
Glitch no R (ou S) não afeta R1 (ou S1)

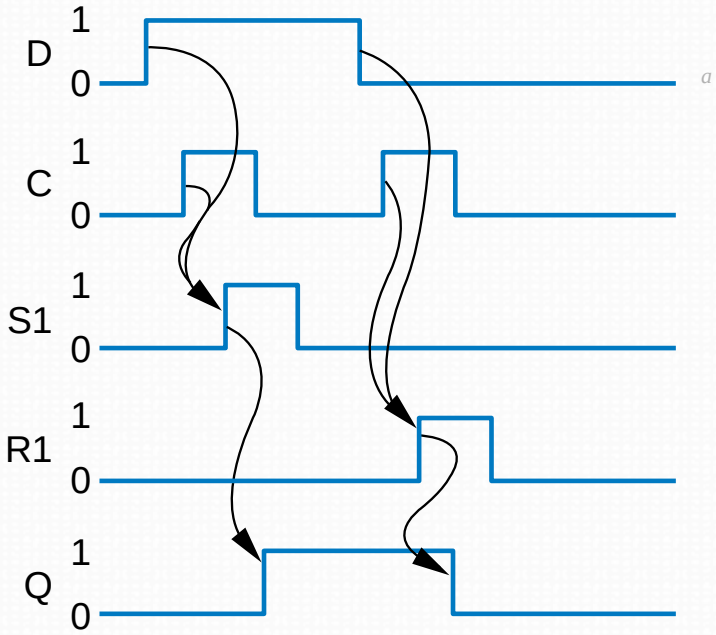
Exercícios

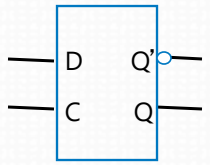
- Também é possível fazer um Latch SR com portas NAND ao invés de NOR?
- Ele funcionaria do mesmo jeito?

Latch D sensível ao nível

- O latch SR necessita de um projeto cuidadoso para que SR=11 nunca ocorra
- O latch D alivia o projetista desse fardo.
  - Um inversor inserido no R garante que S nunca seja igual a R.



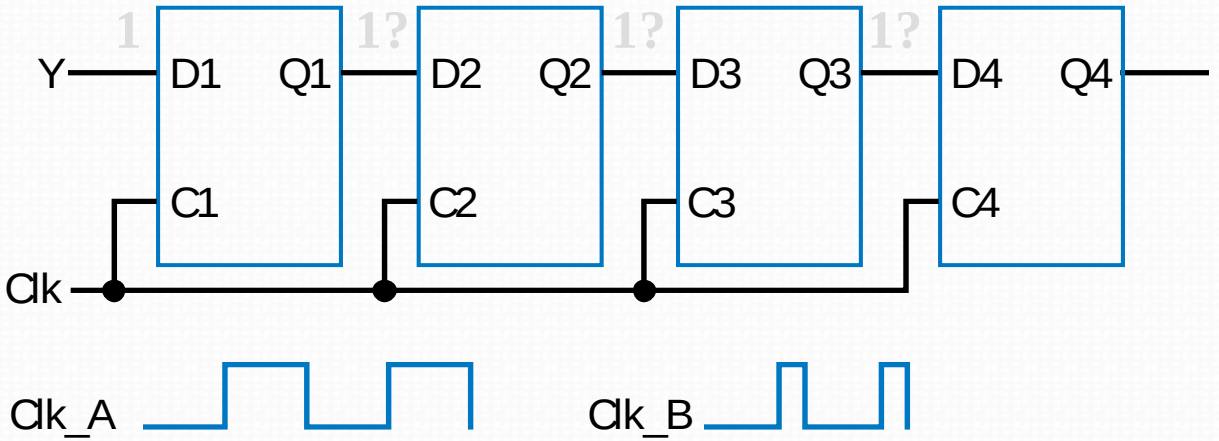




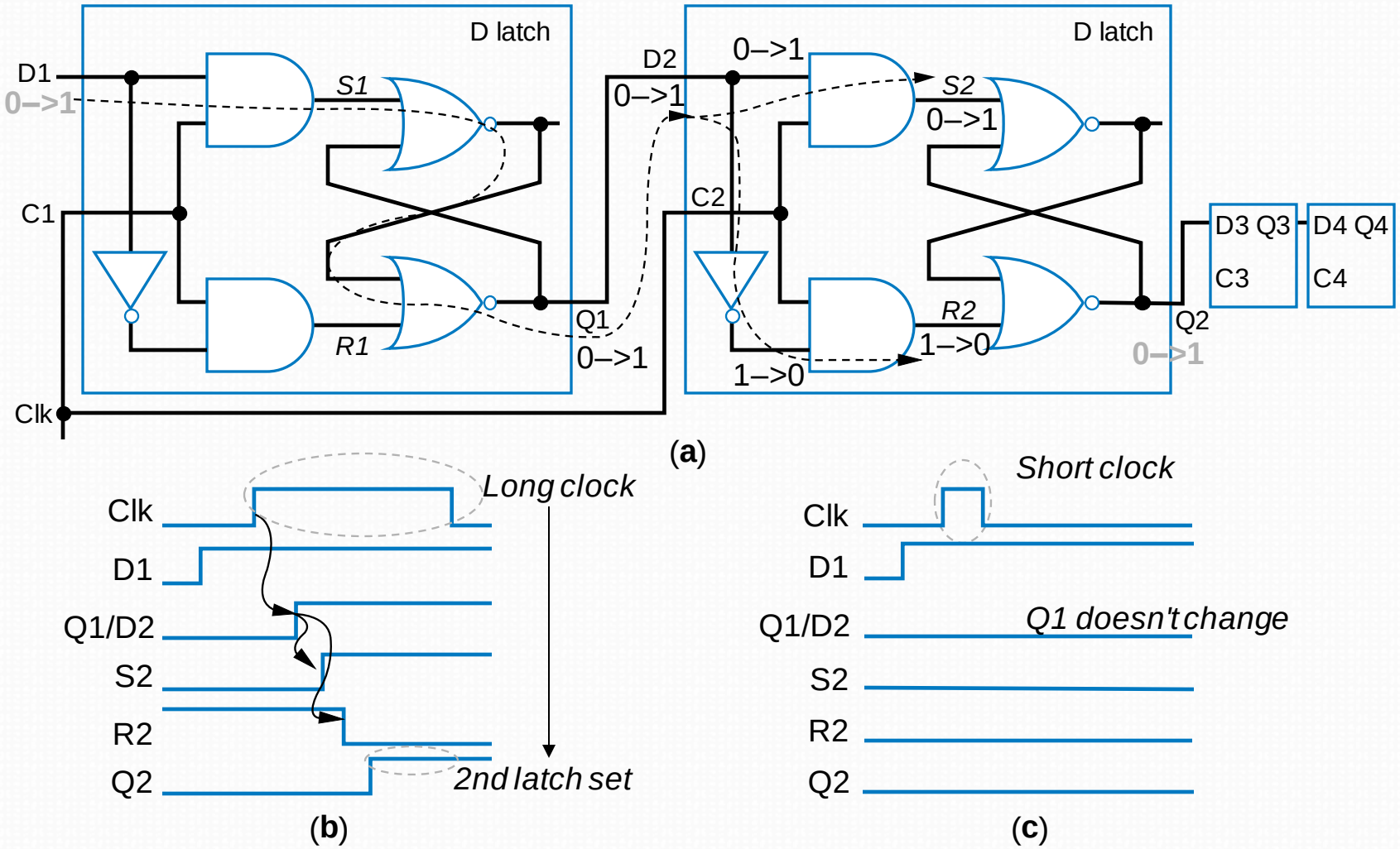
D latch symbol

Problema com Latch D sensível ao nível

- O latch D ainda tem problema (assim como o latch SR)
  - Quando C=1, por quantos latches o C vai viajar?
  - Depende quão longo C=1
    - Clk_A – o sinal deve viajar por muitos latches
    - Clk_B – o sinal deve viajar por menos latches

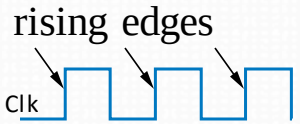


Problema com Latch D sensível ao nível

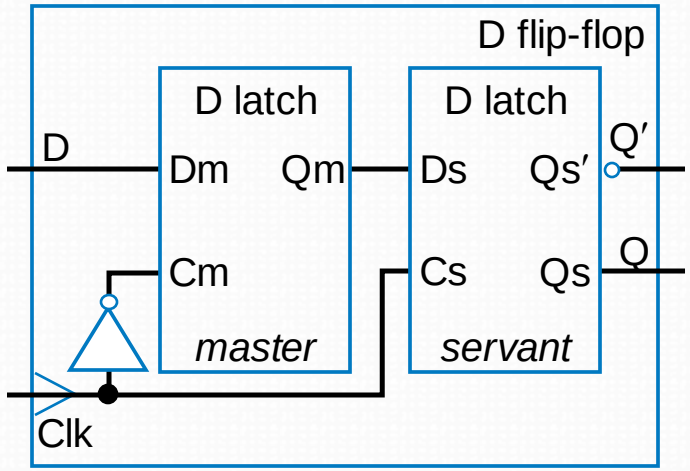


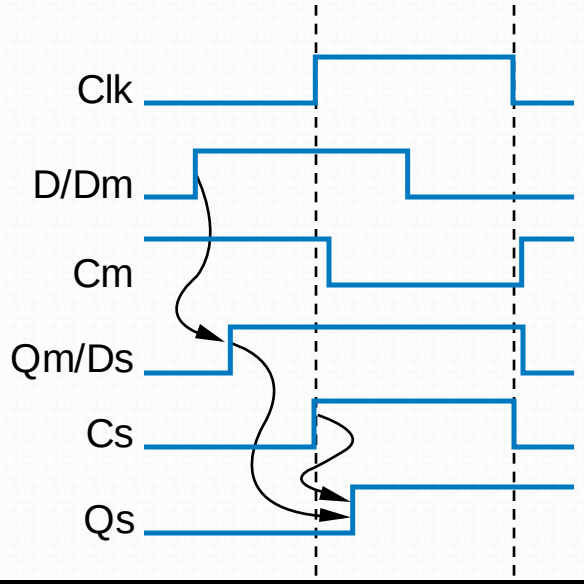
Flip-Flop D

Podemos projetar um armazenamento que só ocorra na borda de subida de um sinal de clock?



- *Flip-flop*: o armazenamento ocorre na borda do clock
- Projeto – mastre-escravo
  - Clk = 0 – Mestre habilitado, carrega D, aparece em Qm. Escravo desabilitado.
  - Clk = 1 – Mestre desabilitado, Qm permanece. Latch Escravo habilitado, carrega Qm, aparece em Qs.
  - O valor D fica armazenado em Qm e quando o clock muda de 0 para 1 é armazenado no Escravo

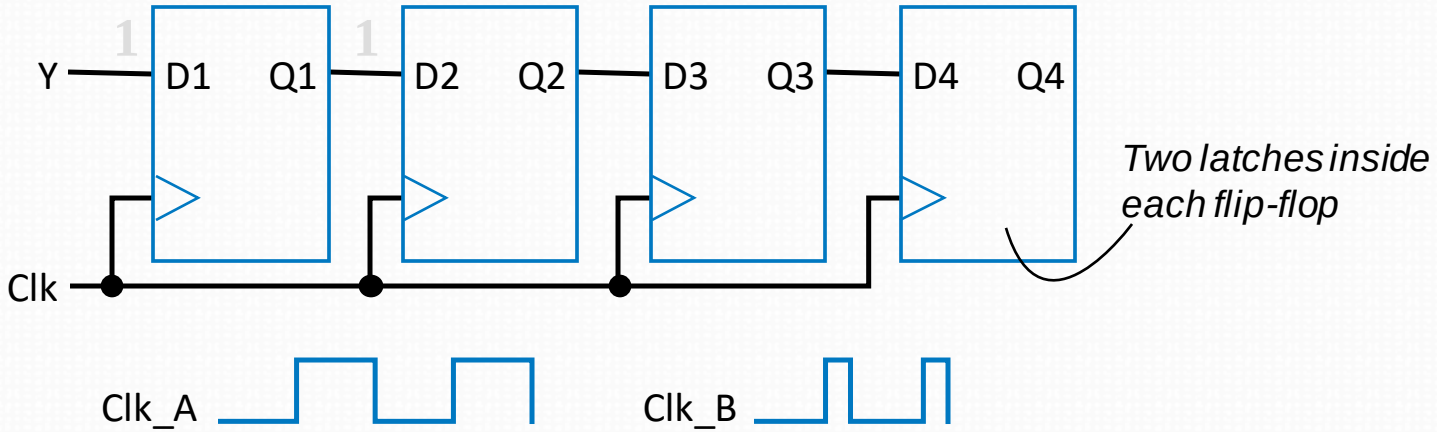




Note: Hundreds of different flip-flop designs exist

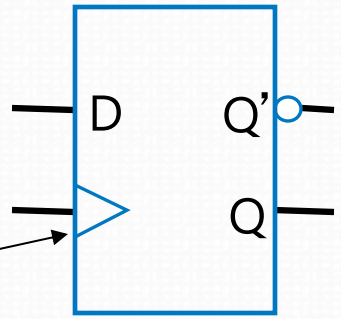
Flip-Flop D

- Soluciona o problema de não saber por quantos latches um sinal viaja quando C=1
  - Na figura abaixo, o sinal viaja por exatamente 1 flip-flop para Clk_A ou Clk_B
  - Porque? Porque na borda de subida (rising_edge) de Clk, todos os 4 flip-flops são carregados simultaneamente em seguida todos deixam de prestar atenção até a próxima borda de subida não importando por quando tempo Clk fique em 1.

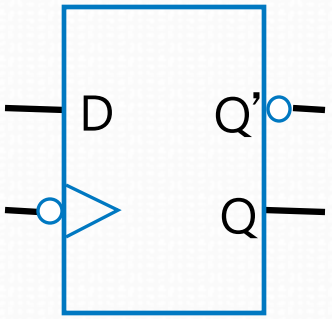


D Flip-Flop

The triangle means edge-triggered clock input

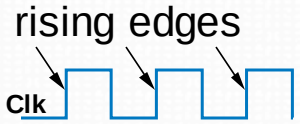


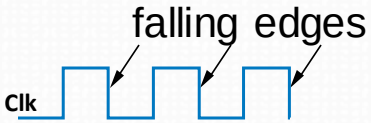
Symbol for rising-edge triggered D flip-flop



Symbol for falling-edge triggered D flip-flop

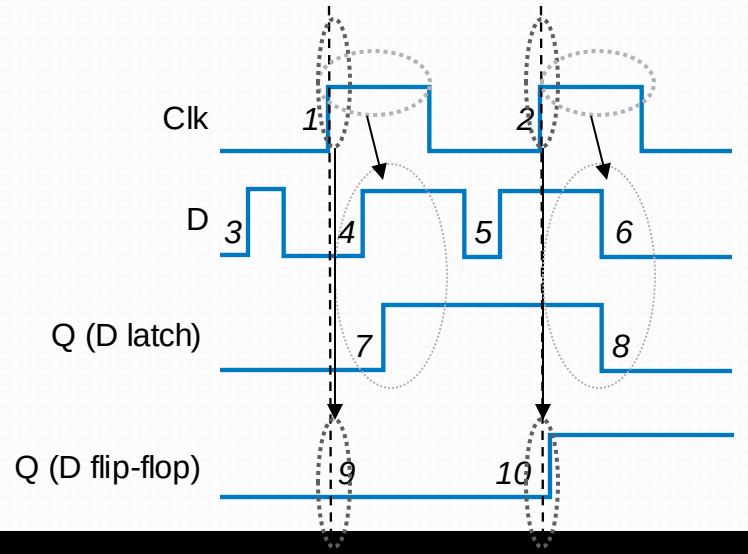
Internal design: Just invert servant clock rather than master





Latch D x Flip-Flop D

- O latch é sensível ao nível
  - Armazena D quando C=1
- O Flip-flop sensível a borda
  - Armazena D quando C muda de 0 para 1
- OBS: Dizer que um latch é sensível ao nível ou que um flip-flop é sensível a borda é redundante
- Comparando os comportamentos de um latch e de um flip-flop:



Latch follows D while Clk is 1

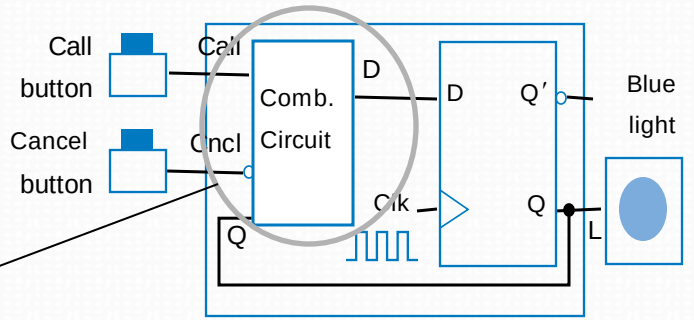
Flip-flop only loads D during Clk rising edge

Atividade

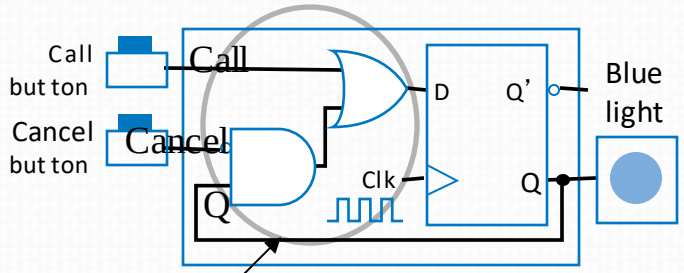
- Descubra como um oscilador com cristal de quartzo pode ser usado para gerar um sinal de relógio.
- Descreva seu funcionamento e recomende um vídeo que fale sobre o assunto.

Botão de chamada para Comissária de Bordo com D Flip-Flop

- flip-flop D irá armazenar o bit
- As entradas são Call, Cancel, Q (valor atual do flip-flop)
- Tabela verdade, a seguir



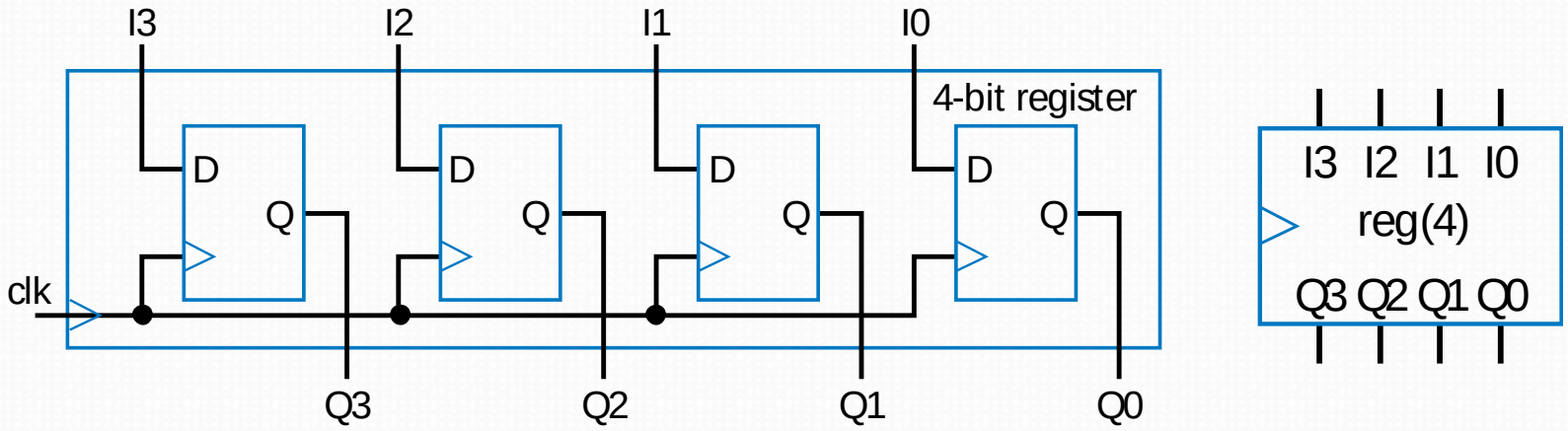
Call	Cancel	Q	D	
0	0	0	0	Preserve value: if Q=0, make D=0; if Q=1, make D=1
0	0	1	1	
0	1	0	0	Cancel -- make D=0
0	1	1	0	
1	0	0	1	Call -- make D=1
1	0	1	1	
1	1	0	1	Let's give priority to Call -- make D=1
1	1	1	1	



Circuito derivado da tabela verdade, usando o processo de projeto combinacional

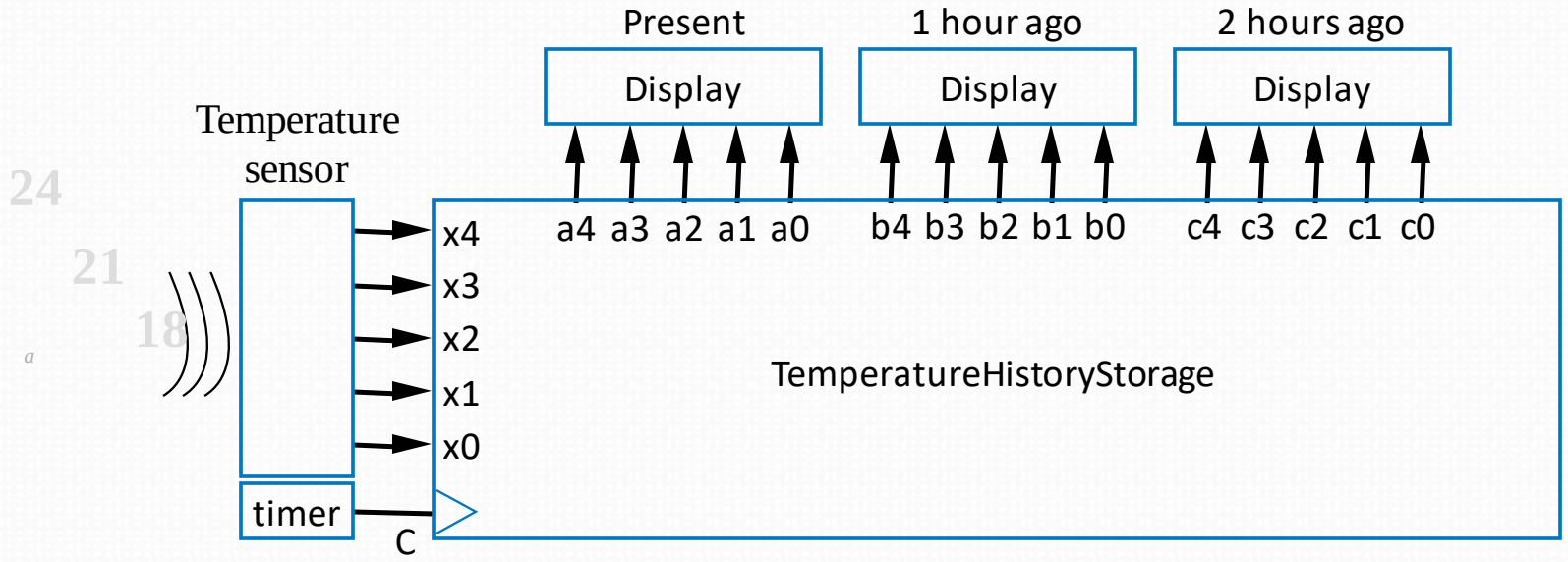
Registrador Básico

- Tipicamente armazenamos valores de múltiplos bits
  - ex., armazenando um número de 4-bits
- *Registrador*: multiplos flip-flops armazenando sinal de clock



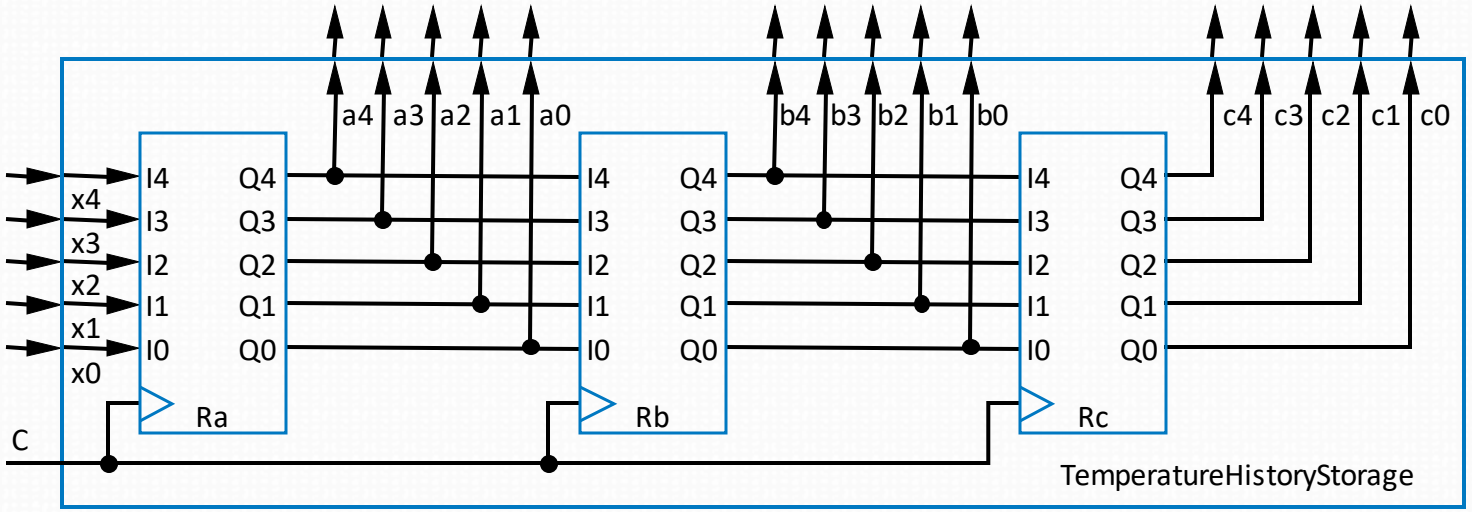
Exemplo: Display de Temperatura

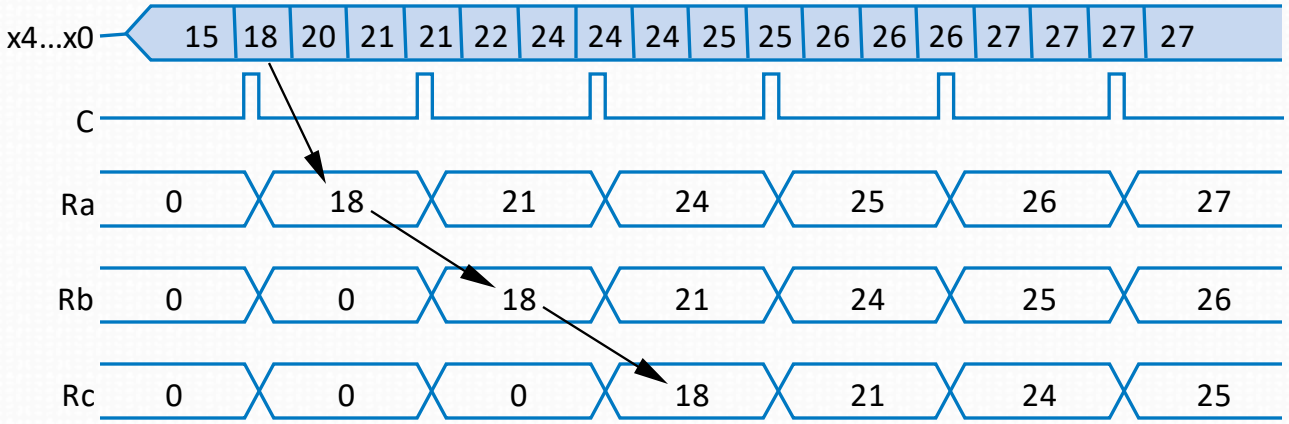
- Mostrador de histórico de temperatura
  - Saídas de temperatura número binário de 5-bits
  - O temporizador pulsa a cada hora
  - Registra a temperatura a cada pulso, mostra os últimos 3 valores registrados



Exemplo: Display de Temperatura

- Use three 5-bit registers

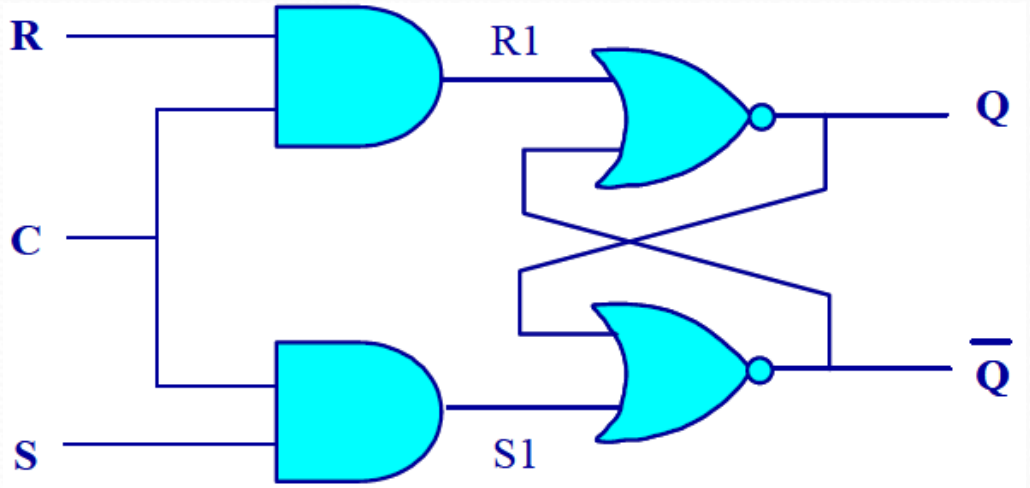




Note that registers only loaded on rising clock edges

Latch RS controlado

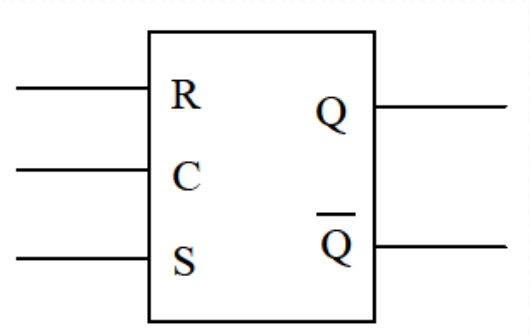
- Surgiu da necessidade de criar uma entrada que controlasse a habilitação do latch
  - Evitando a necessidade de escrita constante

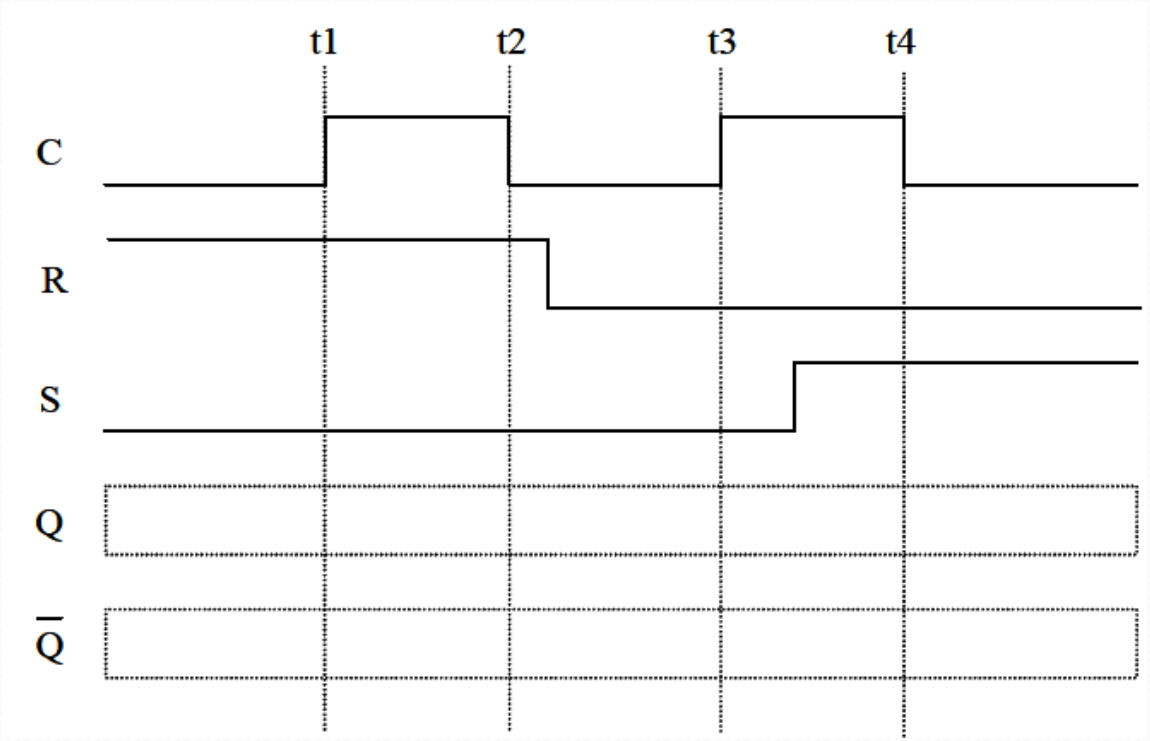


C	R	S	Q_{t+1}
0	x	x	Q_t
1	0	0	Q_t
1	0	1	Estado set
1	1	0	Estado reset
1	1	-	proibido

Atividade

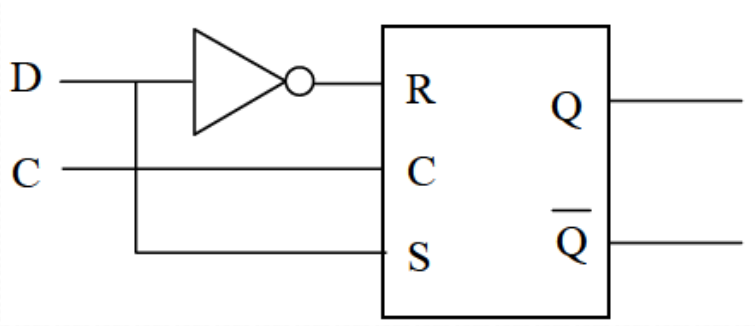
- Desenhar as formas de onda para as saídas do latch RS abaixo, a partir das formas de onda fornecidas para as entradas C, R e S





Latch D

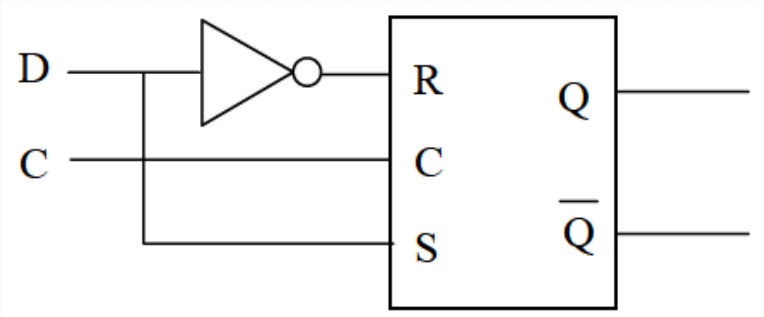
- Surgiu da necessidade de evitar a ocorrência do estado proibido



- Porque esse *Latch* evita a ocorrência de um estado proibido?

Latch D

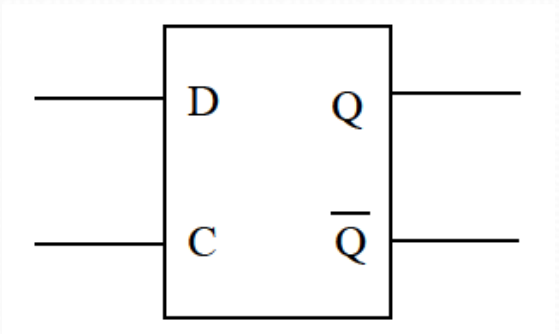
- O inversor entre as entradas S e R assegura que nunca ocorrerá o estado S=1 e R=1
  - Responsáveis pelo estado proibido

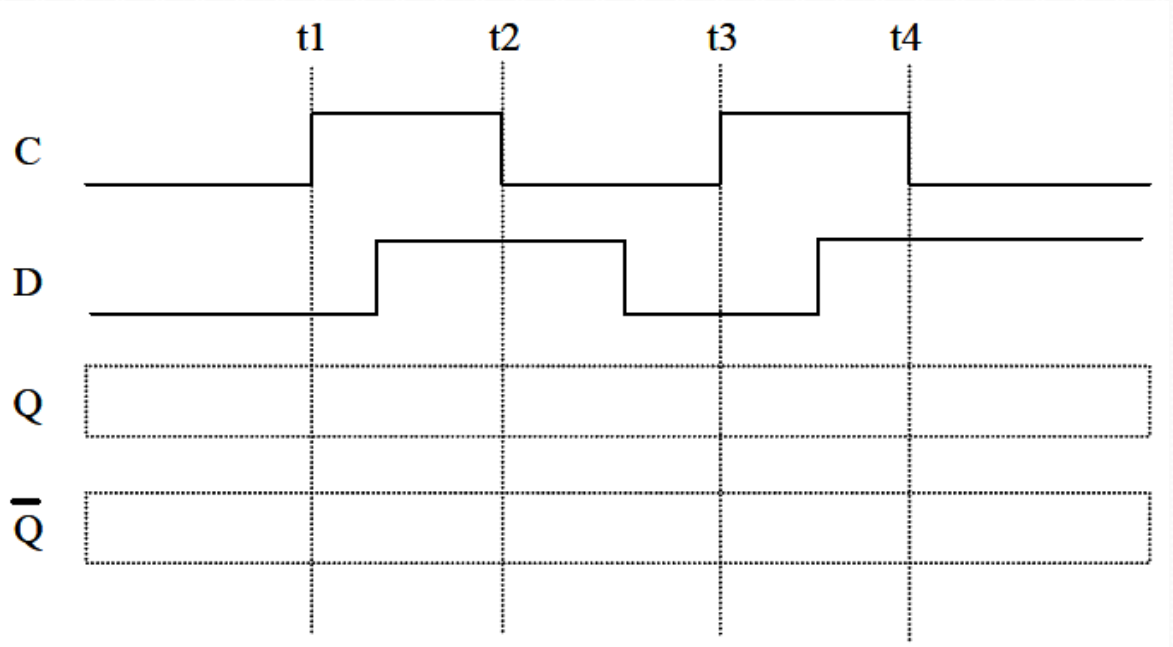


C	D	Q_{t+1}
0	x	Q_t
1	0	0
1	1	1

Atividade

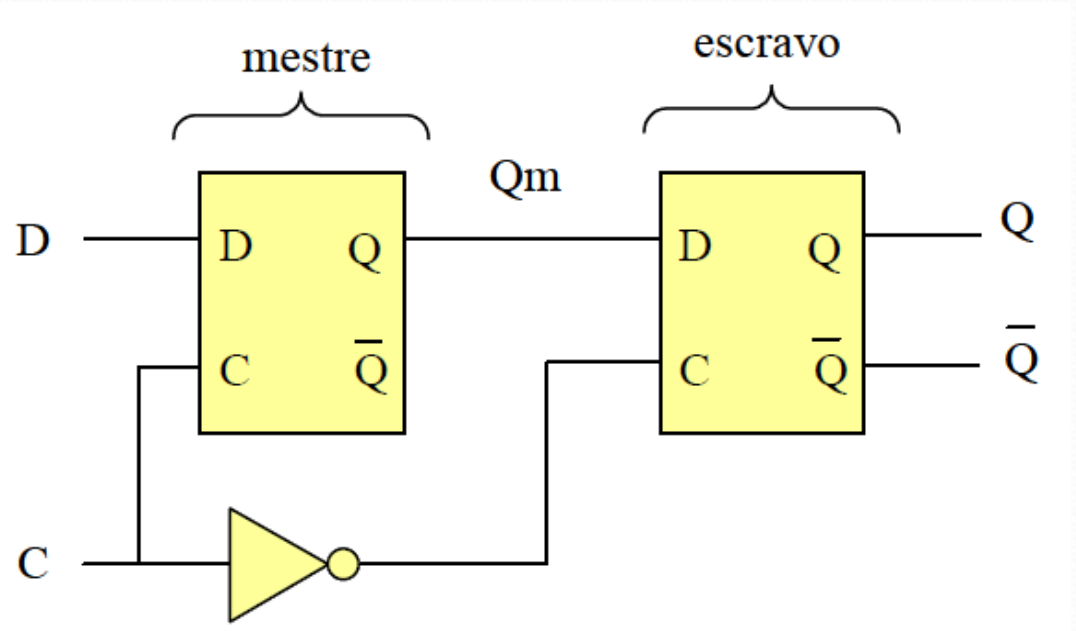
- Desenhar as formas de onda para as saídas do latch D abaixo, a partir das formas de onda fornecidas para as entradas



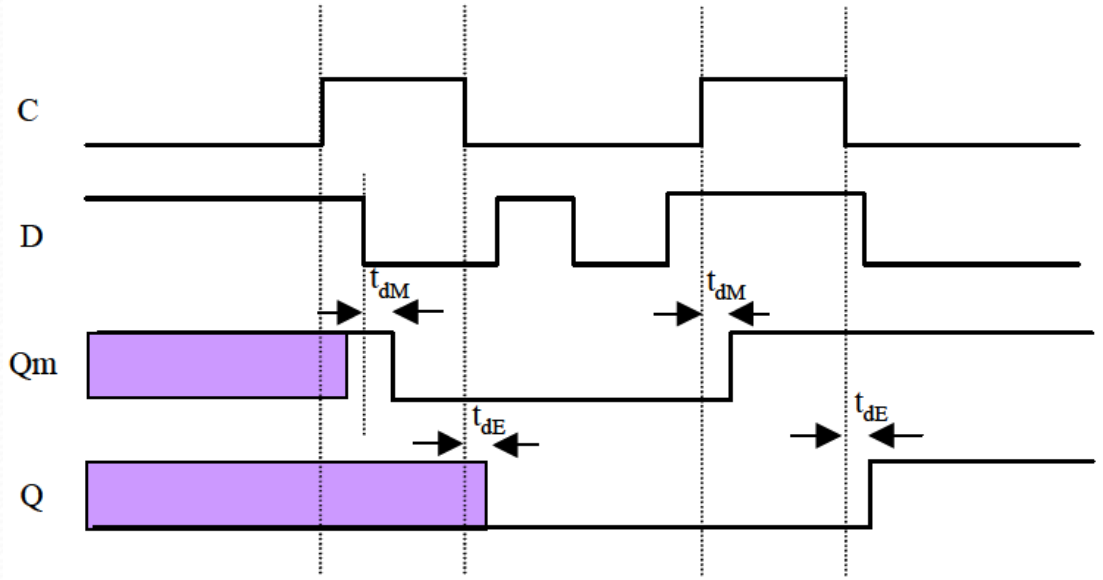


Flip-Flop D mestre-escravo

- Composto de 2 latches D conectados em cascata
  - Funcionam de forma complementar:
    - Se $Controle = 1$ então o mestre é ativado e o escravo mantém o estado anterior
    - Se $Controle = 0$ então o mestre mantém o estado anterior e o escravo é ativado



Flip-Flop D mestre-escravo

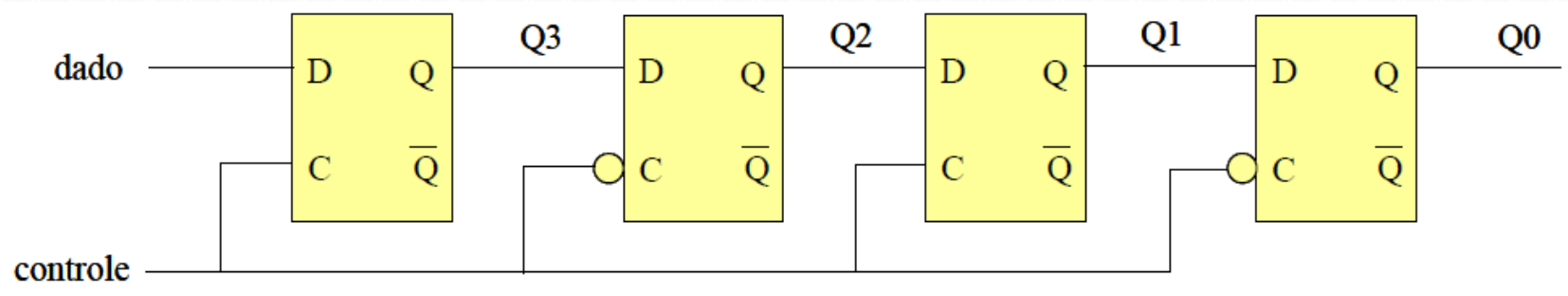


- t_{dM} = Atraso do latch mestre
- t_{dE} = Atraso do latch escravo

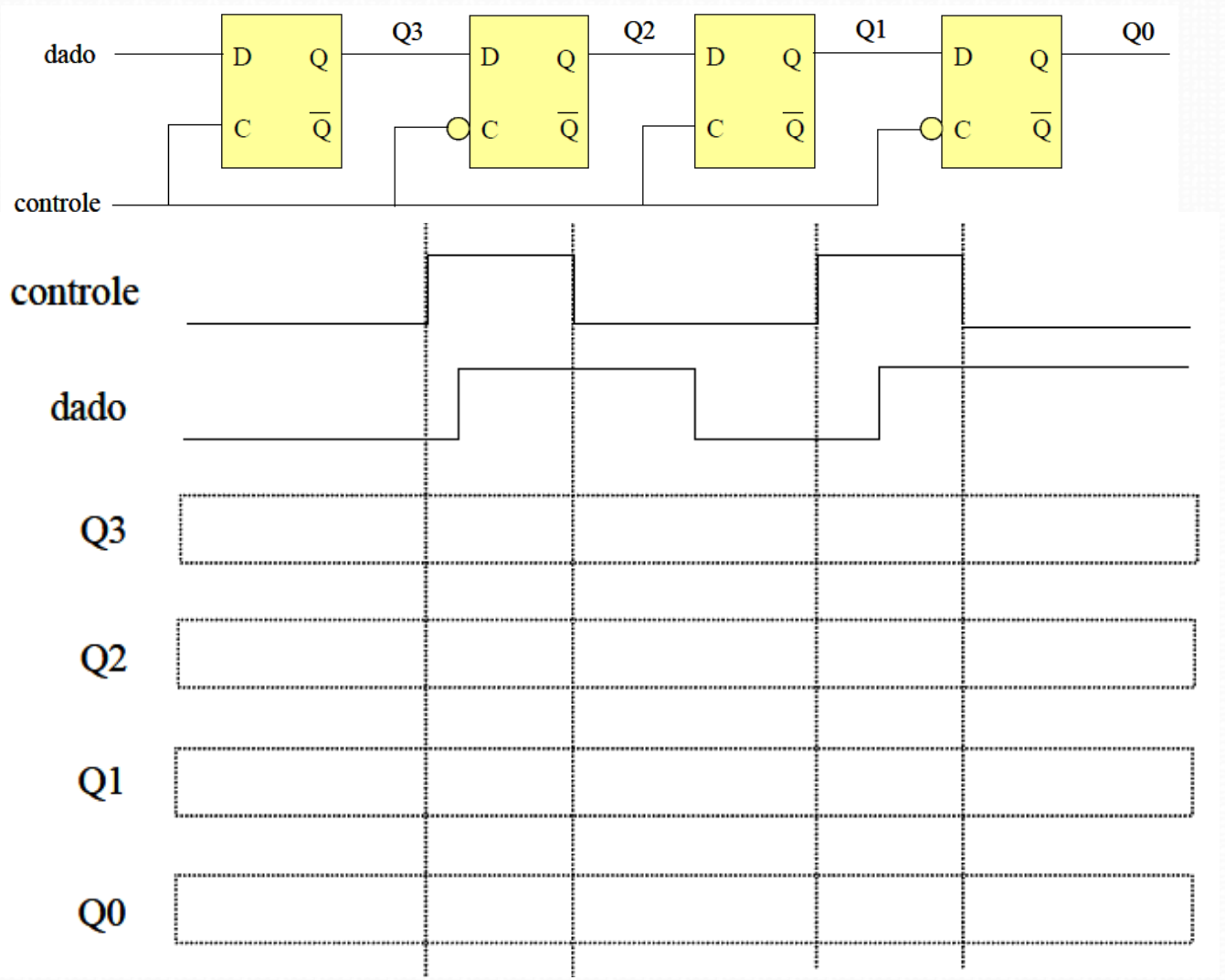
- O último valor de D amostrado pelo *latch* mestre antes da *borda descendente* fica armazenado
  - Aparecendo na saída Q do *latch* escravo após a *borda descendente*

Atividade 1

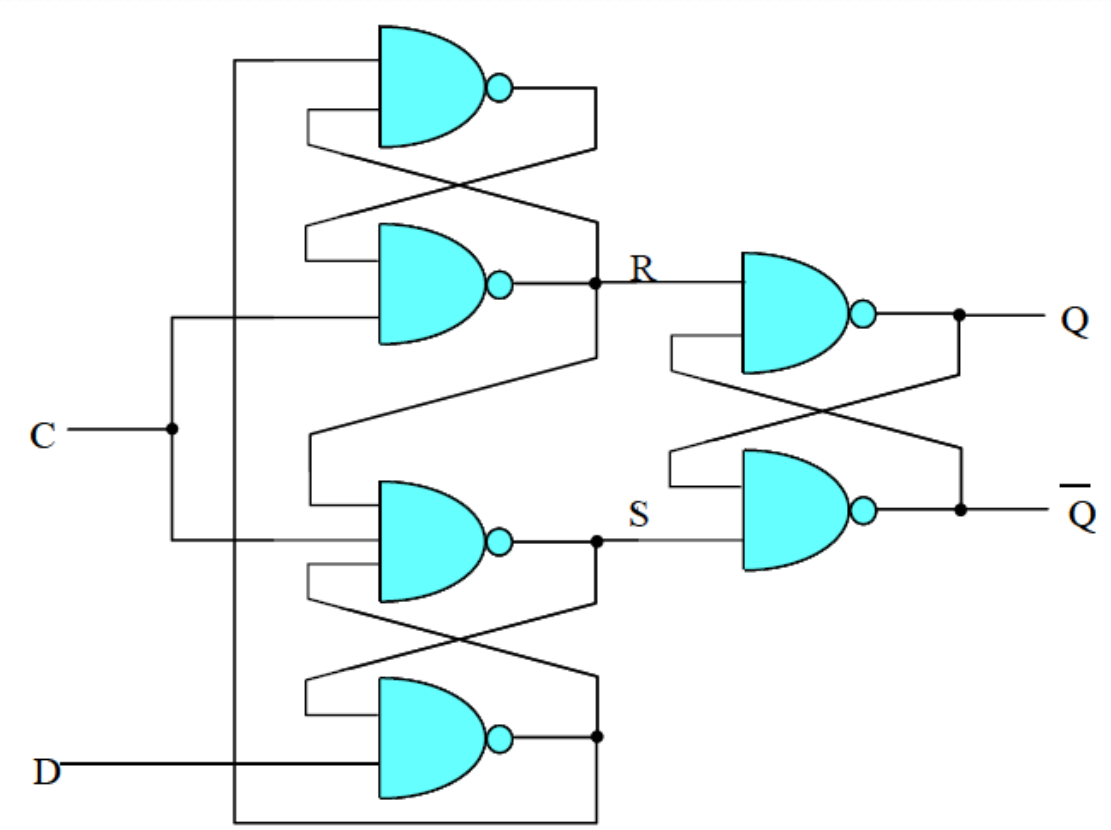
- Traçar as formas de onda para as saídas de cada um dos *latches* do circuito que segue, a partir das formas de onda fornecidas



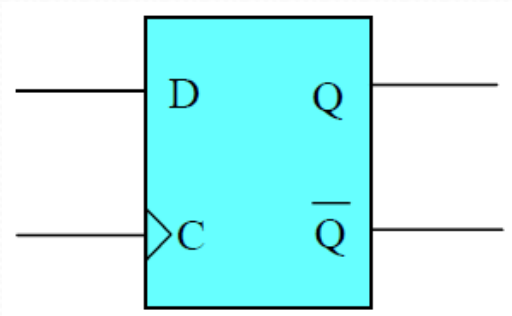
Atividade 1



Flip-flop D sensível a borda *ascendente*

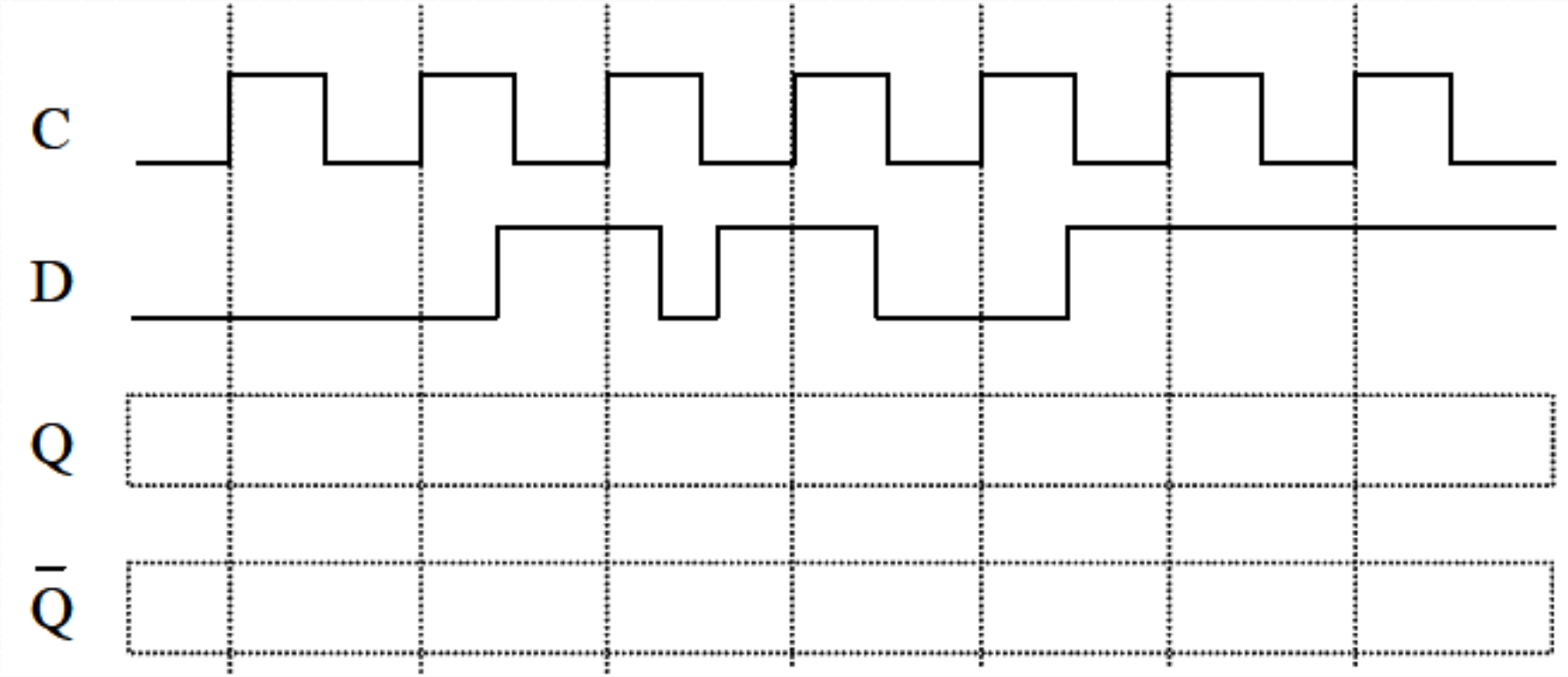


C	D	Q_{t+1}
$\neq\uparrow$	x	Q_t
$\uparrow$	0	0
$\uparrow$	1	1



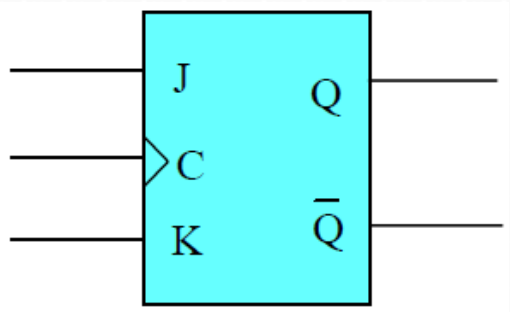
Atividade 2

- traçar as formas de onda para as saídas do flip-flop D, a partir das formas de onda fornecidas



Flip-Flop JK

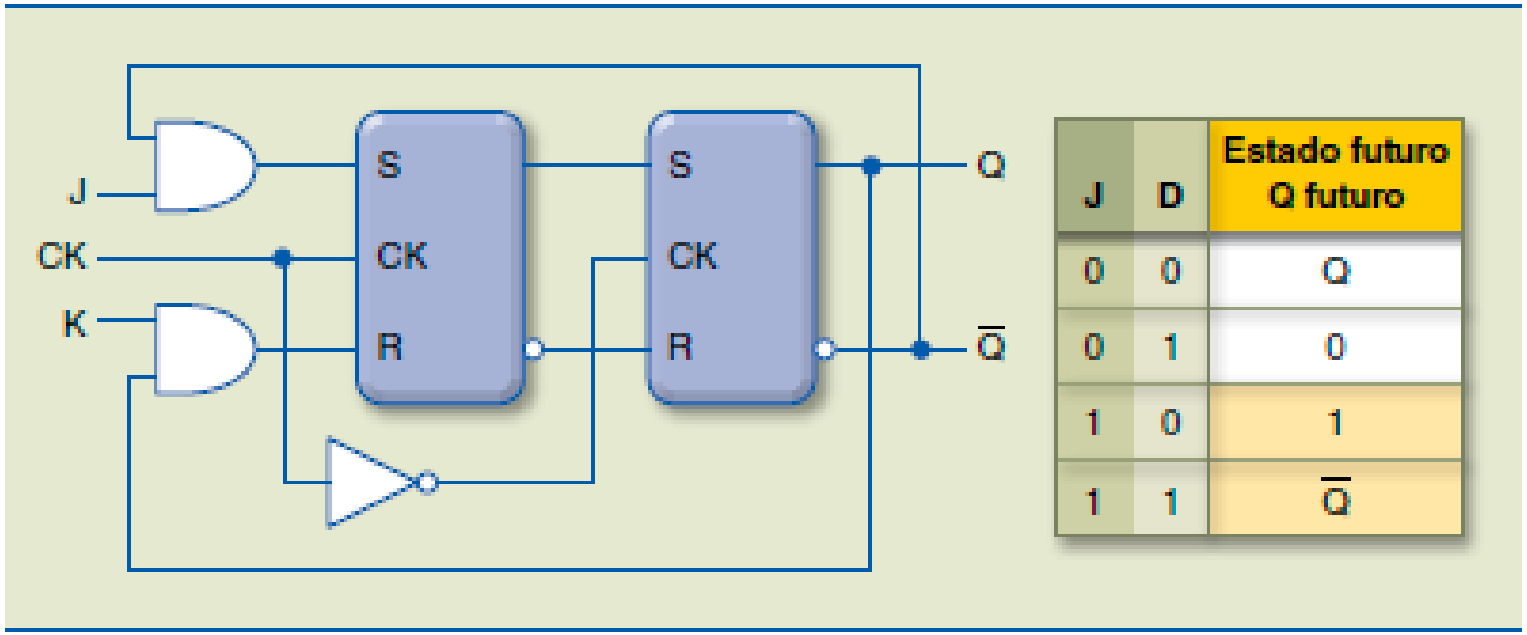
- Construído a partir de *latchs RS controlados*
  - Exceto que a combinação $J = K = 1$ não leva a um estado proibido
  - $J = K = 1$ resulta em $\overline{Q_t}$



C	J	K	Q_{t+1}	
≠↑	x	x	Q_t	Mantém estado anterior
↑	0	0	Q_t	Mantém estado anterior
↑	0	1	0	Estado reset
↑	1	0	1	Estado set
↑	1	1	$\overline{Q_t}$	Complementa estado anterior

Construção do flip-flop JK

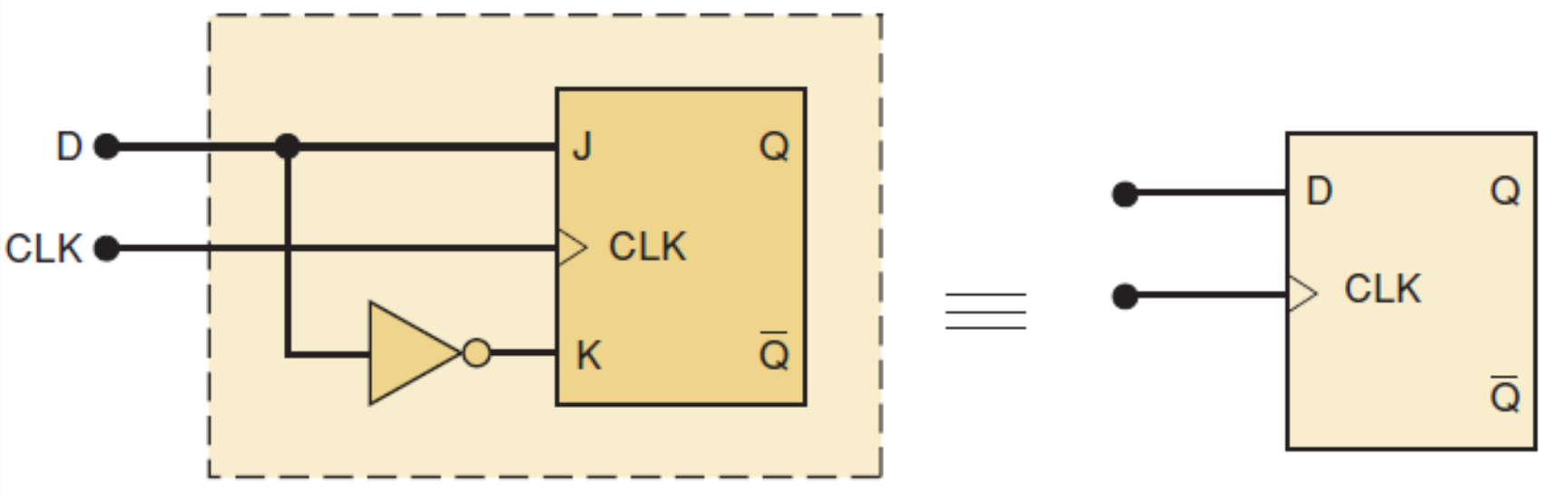
- Visão simplificada do *flip-flop JK* a partir de latchs RS controlados
  - Eletrônica: eletrônica digital. São Paulo: Fundação Padre Anchieta, 2011. (Coleção Técnica Interativa. Série Eletrônica, v. 4)
    - Fornecida pelo acadêmico Leandro



J	D	Estado futuro Q futuro
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

Flip-flop D a partir do JK

- Conforme mostra a figura da literatura da disciplina, o flip-flop D pode ser construído a partir de um flip-flop JK



Obrigado pela atenção

contato: felski@univali.br