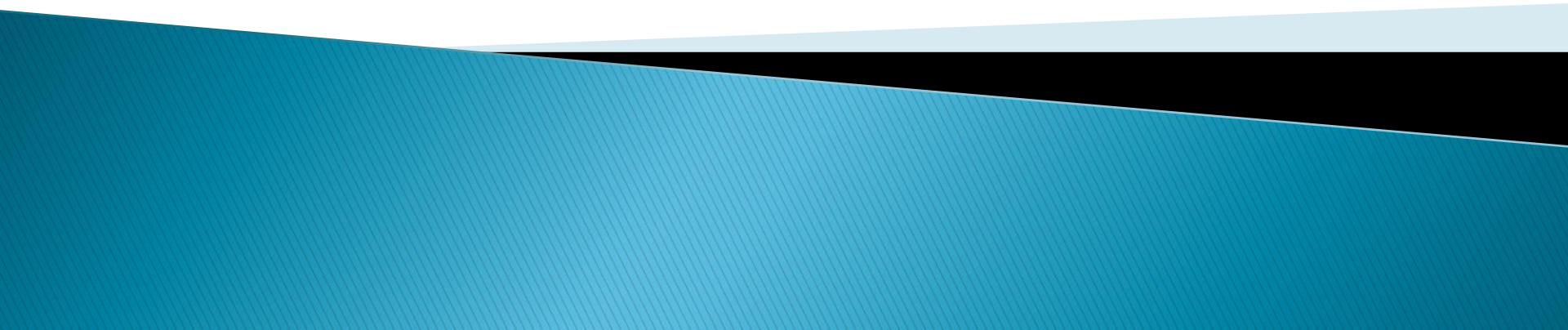


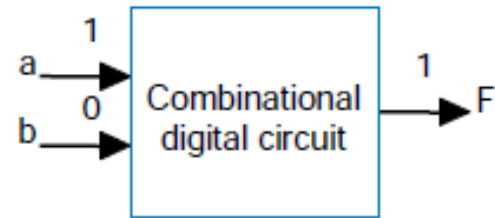
Capítulo 3–Circuitos Sequencial

Profa. Eliete Caldeira

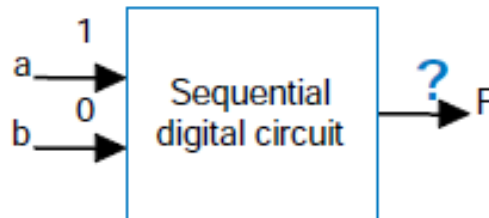


Circuitos sequenciais

- ▶ Circuitos Combinacionais: a saída no instante t depende apenas das entradas no instante t
 - Não tem memória: não podemos armazenar bits para, mais tarde, ler
 - São muito limitados em sua utilidade.




- ▶ Projetistas usam os circuitos combinacionais como parte de circuitos maiores, chamados **Circuitos sequenciais**, circuitos que realmente têm memória



A saída depende não só dos valores atuais mais também dos valores passados das entradas

Circuitos sequenciais

- ▶ Em um circuito sequencial as saídas dependem das Entradas atuais e do Estado atual
 - ▶ Estado atual:
 - É o conjunto de todos os bits armazenados no circuito
 - Depende da sequência passada de valores que apareceram nas entradas do circuito
 - ▶ Os circuito sequenciais podem ser Síncronos ou Assíncronos
 - Síncronos: mudanças acontecem sincronizadas no tempo de acordo com um relógio ou “clock”
 - Assíncronos: mudanças podem acontecer em qualquer instante
- 

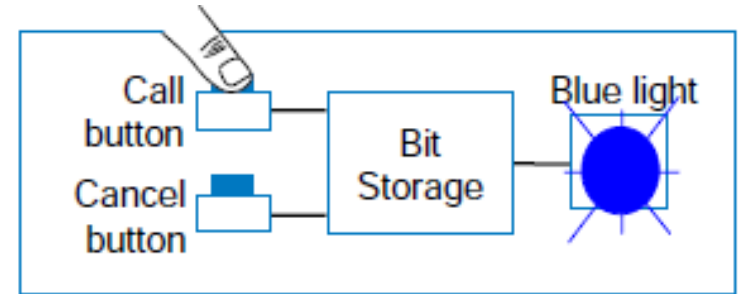
Circuitos sequenciais

▶ Exemplos:

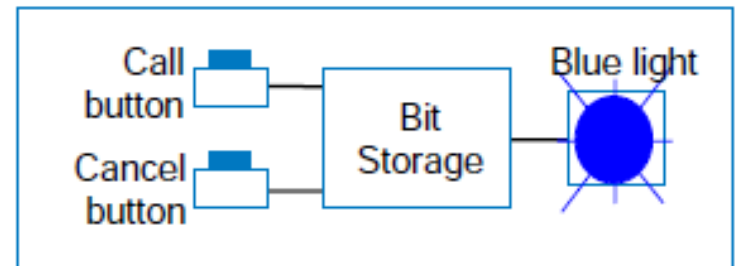
- Calculadora de bolso: deve armazenar os números fornecidos antes de operar com eles
- Câmera digital
- Controlador de semáforos
- Contadores

Exemplo – Chamar o comissário de bordo

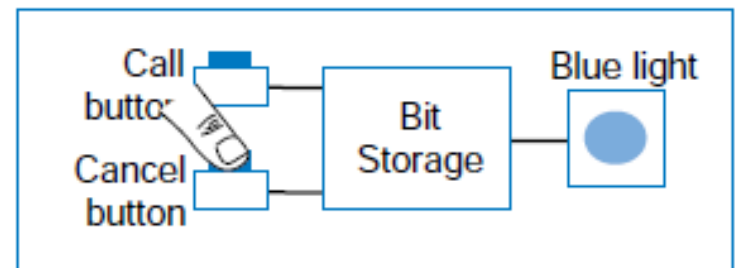
- ▶ Após pressionar o botão para chamar o comissário de bordo, a luz acende e permanece acesa mesmo que o botão seja solto.
- ▶ A luz apaga somente quando o botão de cancelamento é pressionado.



1. Call button pressed – light turns on



2. Call button released – light stays on



3. Cancel button pressed – light turns off

Armazenando um bit

- ▶ Um circuito sequencial deve armazenar uma informação (o botão de chamada do comissário de bordo foi pressionado) até que outra informação seja armazenada (o botão de cancelamento foi pressionado).

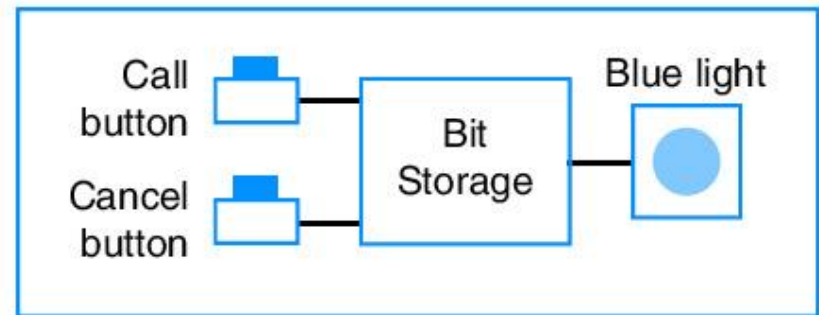
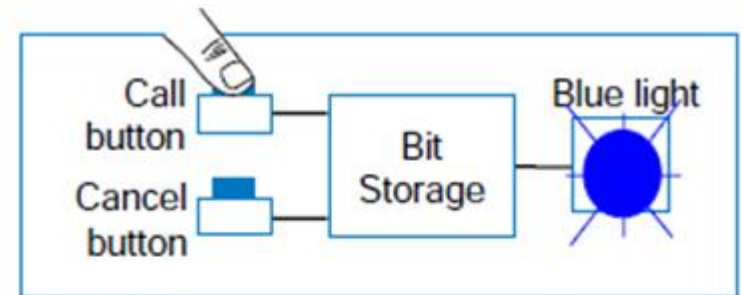


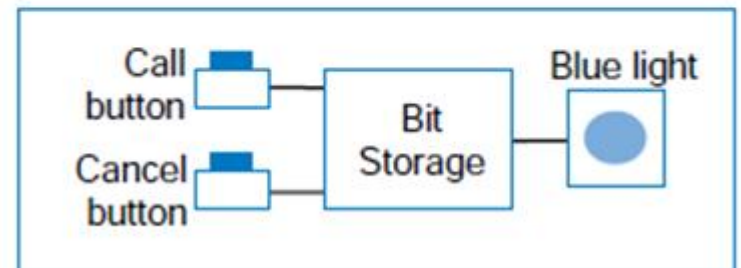
Figure 3.1 Flight attendant call-button system. Pressing Call turns on the light, which stays on after Call is released. Pressing Cancel turns off the light.

Armazenando um bit

- Se o circuito na caixa fosse implementado usando uma porta OR, a lâmpada acenderia ao pressionar o botão de chamada, mas voltaria a apagar quando o botão fosse solto



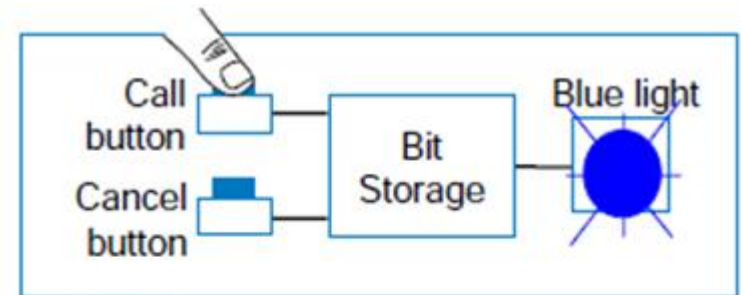
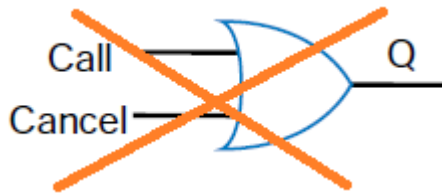
1. Call button pressed – light turns on



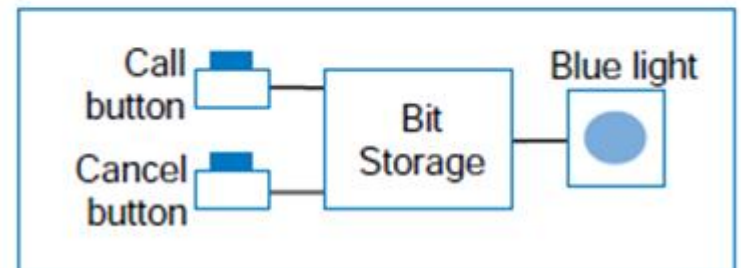
2. Call button released – light turns off

Armazenando um bit

- Se o circuito na caixa fosse implementado usando uma porta OR, a lâmpada acenderia ao pressionar o botão de chamada, mas voltaria a apagar quando o botão fosse solto



1. Call button pressed – light turns on



2. Call button released – light turns off

Armazenando um bit

- ▶ Para armazenar bits é preciso realimentar o circuito
- ▶ Primeira tentativa...

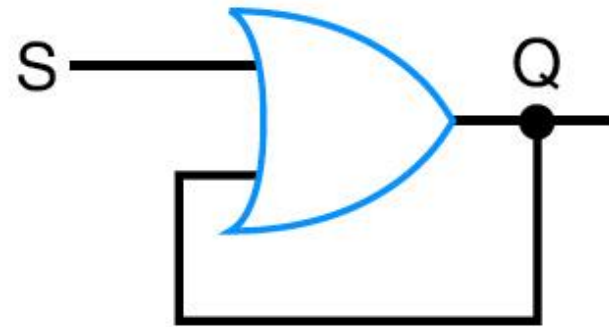


Figure 3.2 First (failed) attempt at using feedback to store a bit.

Armazenando um bit

- ▶ Acompanhando o sinal

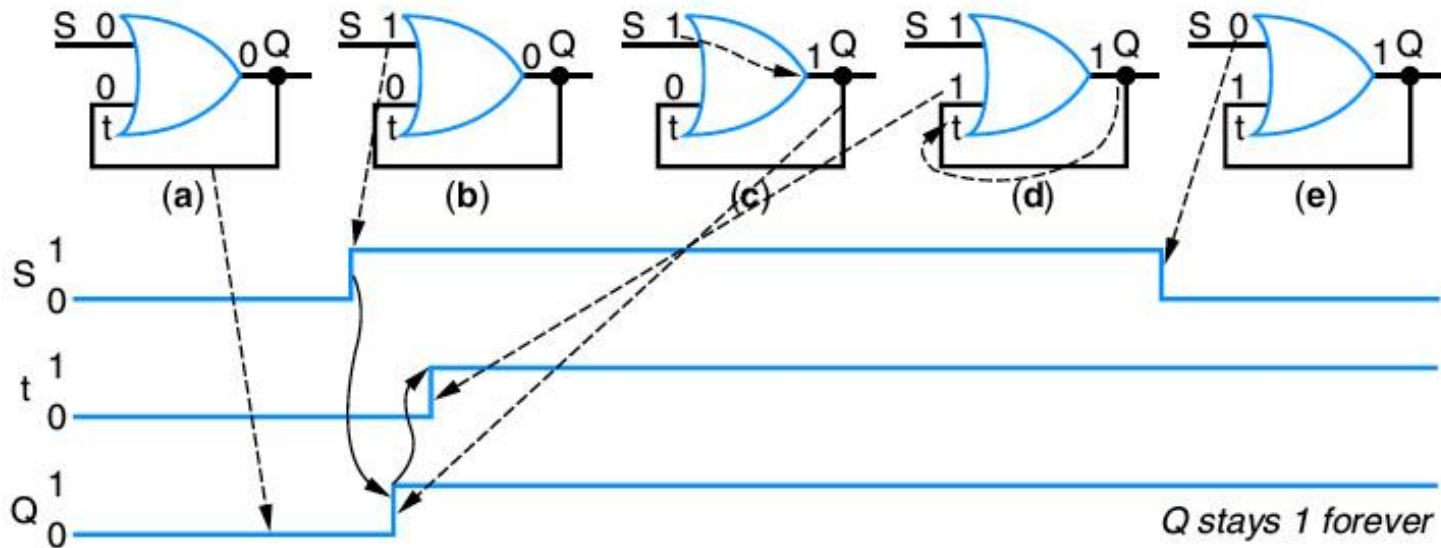


Figure 3.3 Tracing the behavior of our first attempt at bit storage.

- ▶ Resultado: uma vez que Q vai para 1, permanece em 1

Armazenando um bit

- ▶ Latch SR

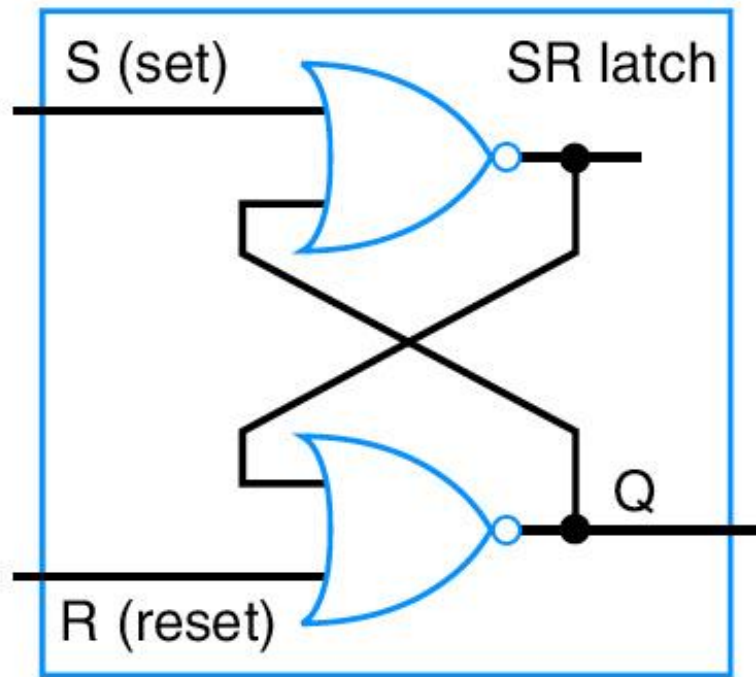


Figure 3.4 Basic SR latch.

Armazenando um bit

► Latch SR

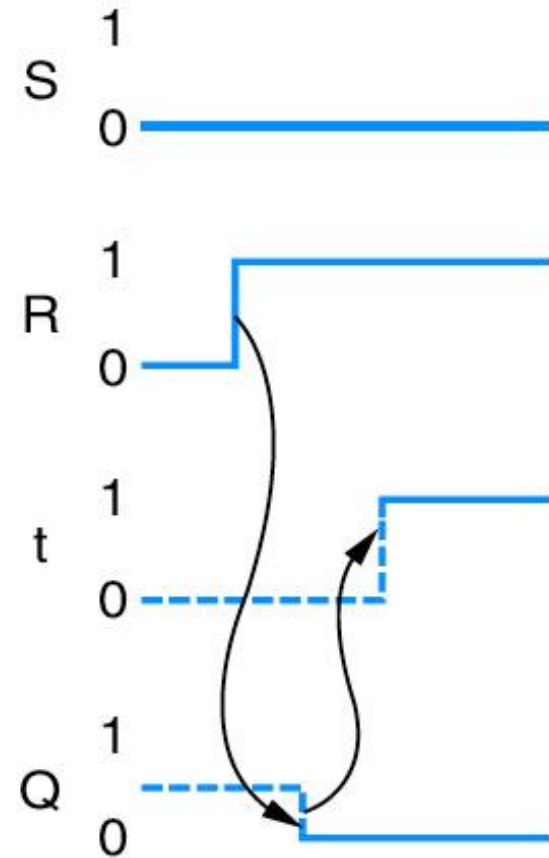
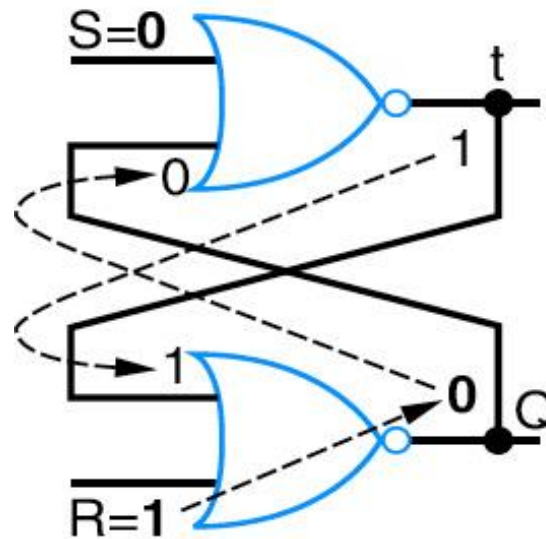


Figure 3.5 SR latch when $S = 0$ and $R = 1$.

Armazenando um bit

- ▶ Latch SR

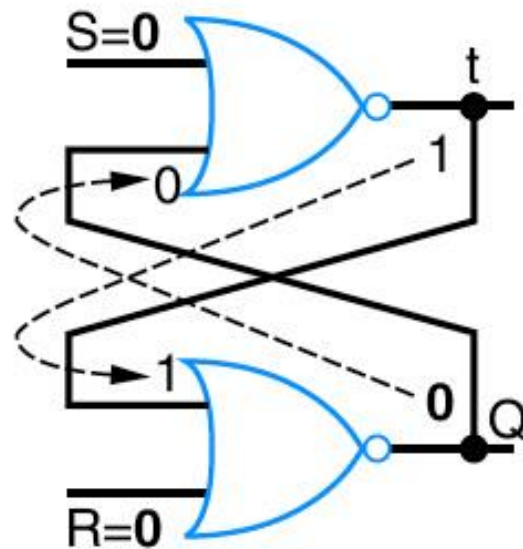
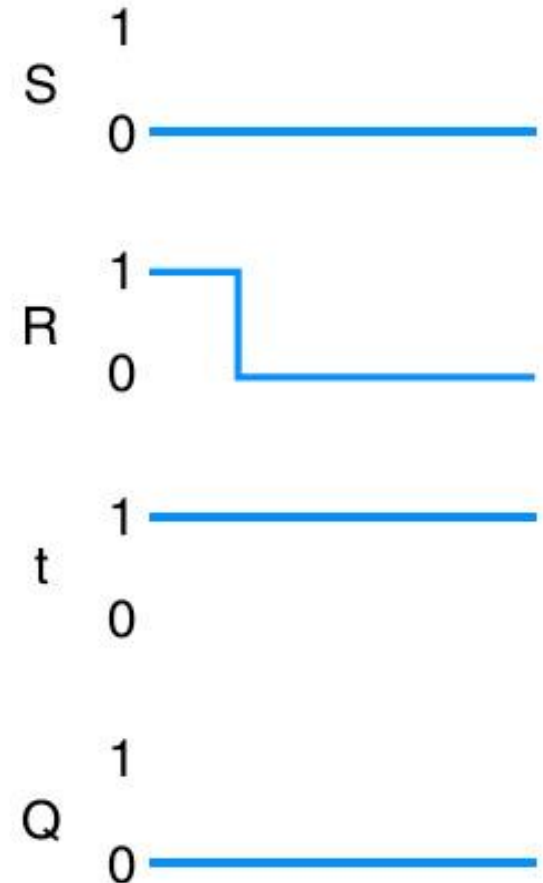


Figure 3.6 SR latch
when $S = 0$ and $R = 0$,
after R equaled 1.



Armazenando um bit

- ▶ Latch SR

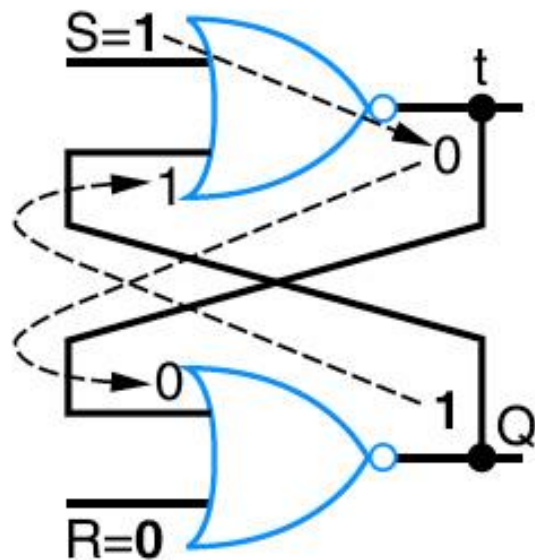
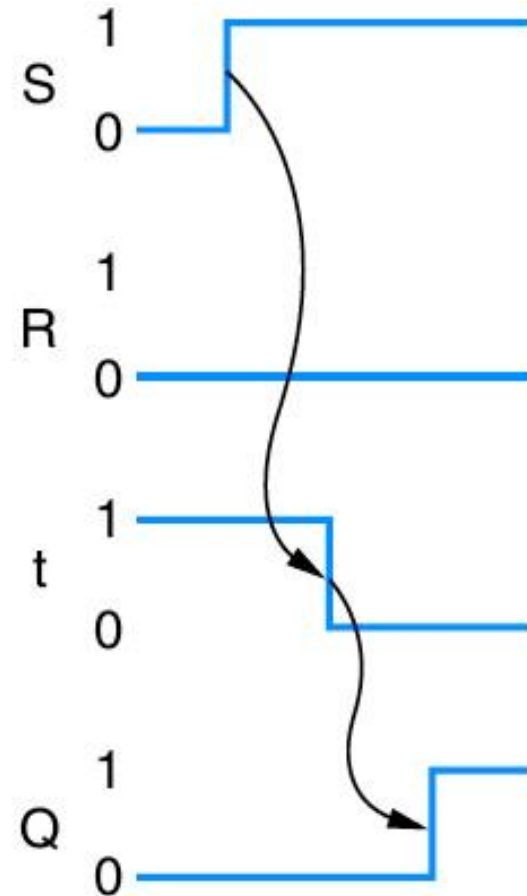


Figure 3.7 SR latch
when $S = 1$ and $R = 0$.



Armazenando um bit

► Latch SR

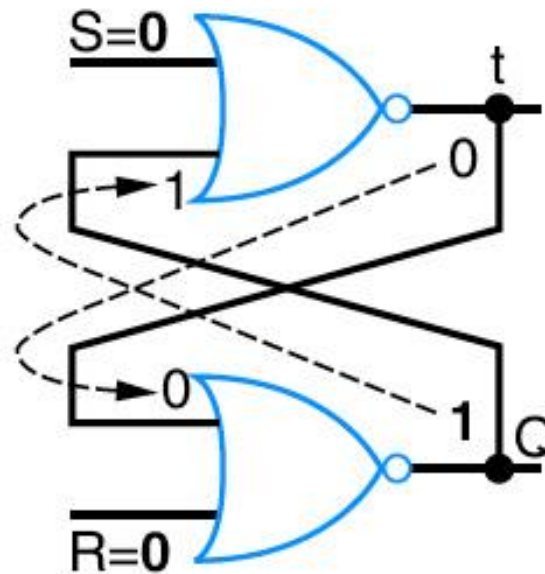
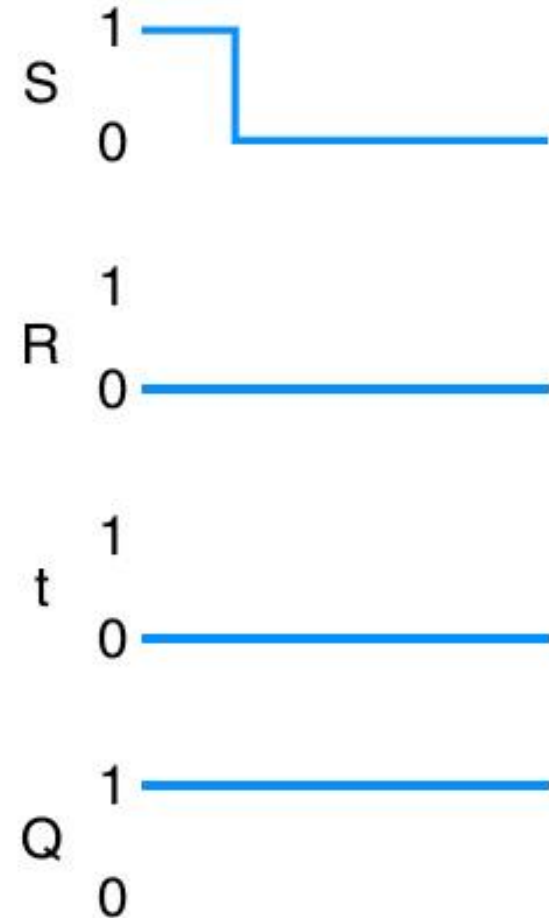


Figure 3.8 SR latch
when $S = 0$ and $R = 0$,
after S equaled 1.



Armazenando um bit

► Latch SR

S	R	Q
0	0	Mantém
0	1	0 (reset)
1	0	1 (set)
1	1	?

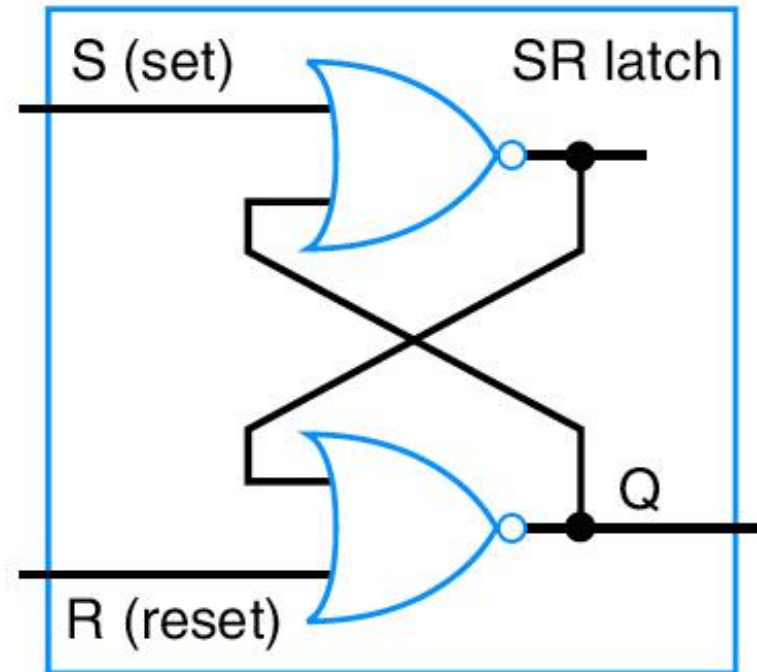
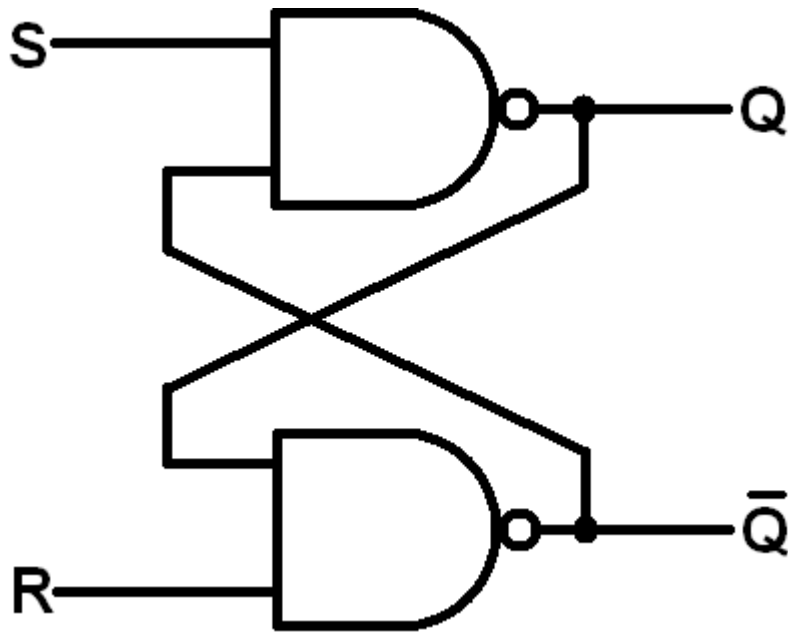


Figure 3.4 Basic SR latch.

Armazenando um bit

- ▶ Latch SR com portas NAND



S	R	Q
0	0	?
0	1	1 (set)
1	0	0 (reset)
1	1	Mantém

Armazenando um bit

- Assim, para o problema de chamada do comissário de bordo

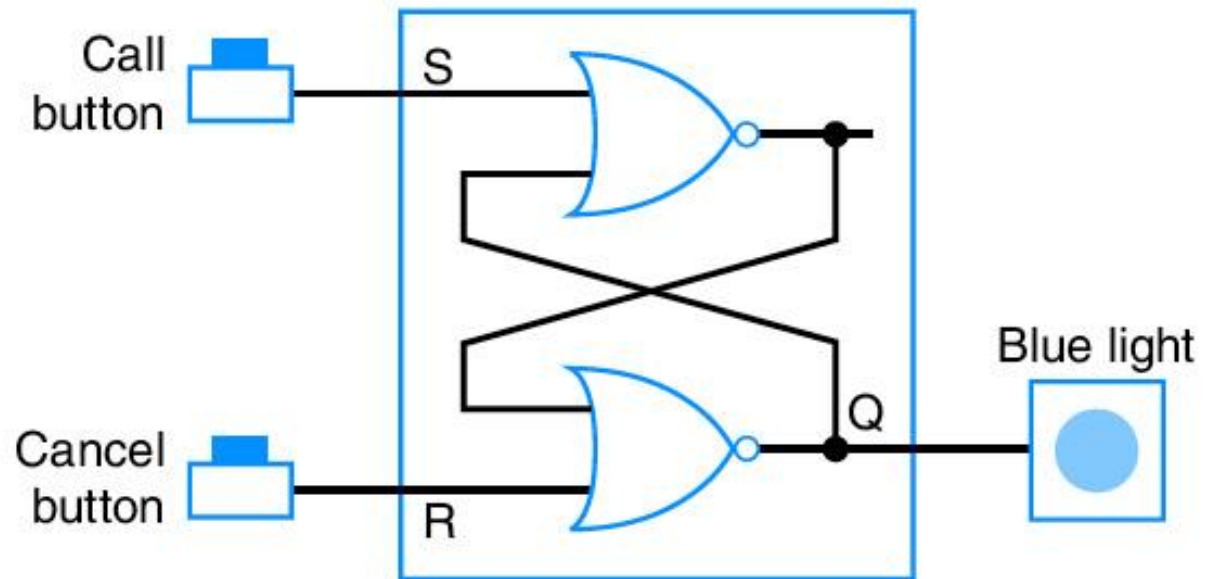


Figure 3.9 Flight attendant call-button system using a basic SR latch.

Armazenando um bit

► Problema!!! Latch SR – corrida de bits

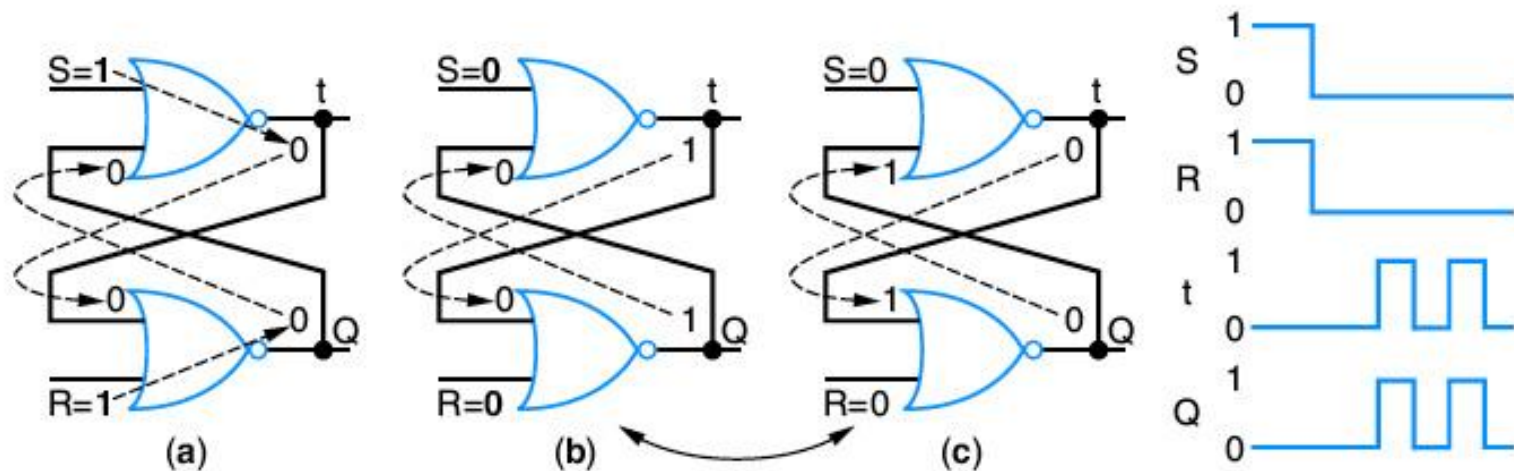


Figure 3.10 The situation of $S = 1$ and $R = 1$ causes problems— Q oscillates when SR return to 00.

- Quando $S=R=1$ ambas as saídas t e Q são 0
- Após S e R voltarem a 0 a saída vai oscilar

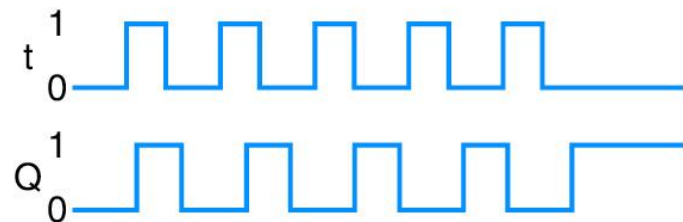


Figure 3.11 Q eventually settles to either 0 or 1, due to race condition.

Armazenando um bit

- ▶ Como evitar a corrida de bits. Primeira tentativa de impedir que S e R sejam 1 ao mesmo tempo

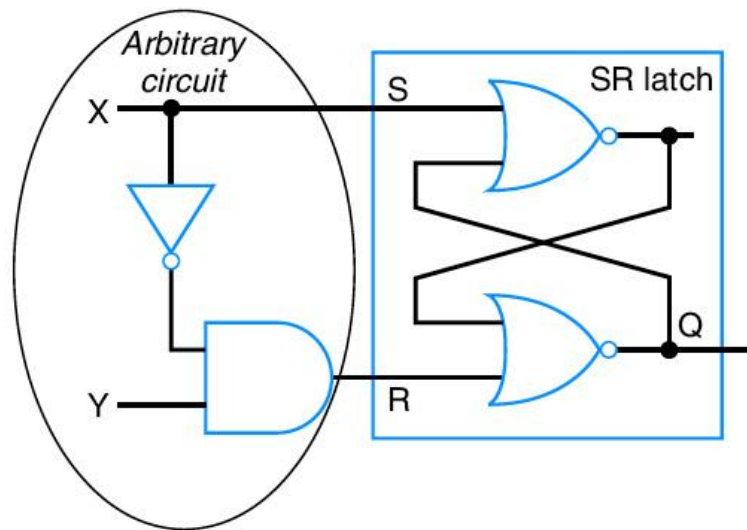


Figure 3.12 Conceptually, S and R can't both be 1 in this sample circuit. But in reality, they can, due to the delay of the inverter and AND gate.

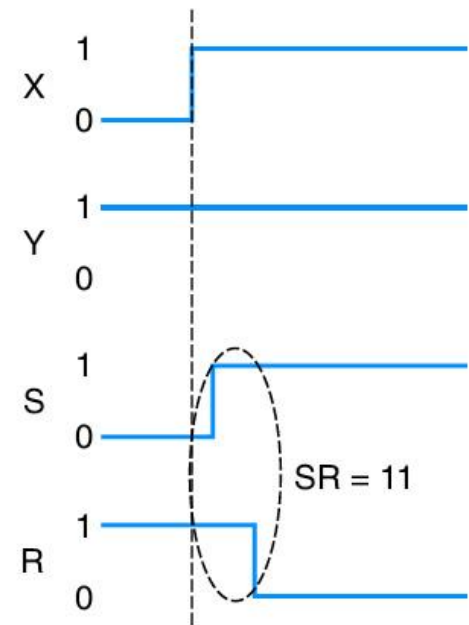


Figure 3.13 Gate delays can cause $SR = 11$.

Armazenando um bit

- ▶ Como evitar a corrida de bits. Segunda tentativa...

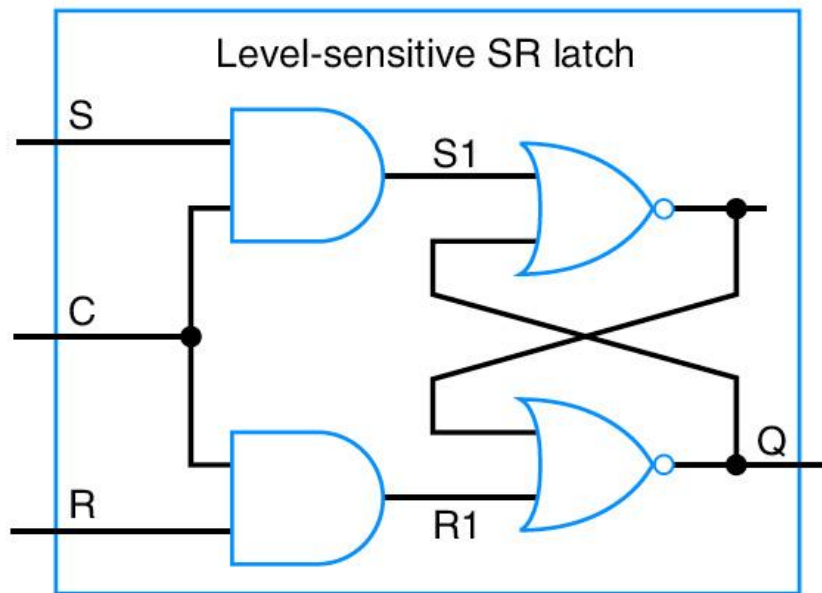


Figure 3.14 Level-sensitive SR latch—
an SR latch with enable input C.

Armazenando um bit

- ▶ Mesmo que S e R sejam 1 ao mesmo tempo, S1 e R1 não são e o latch não tem corrida de bits

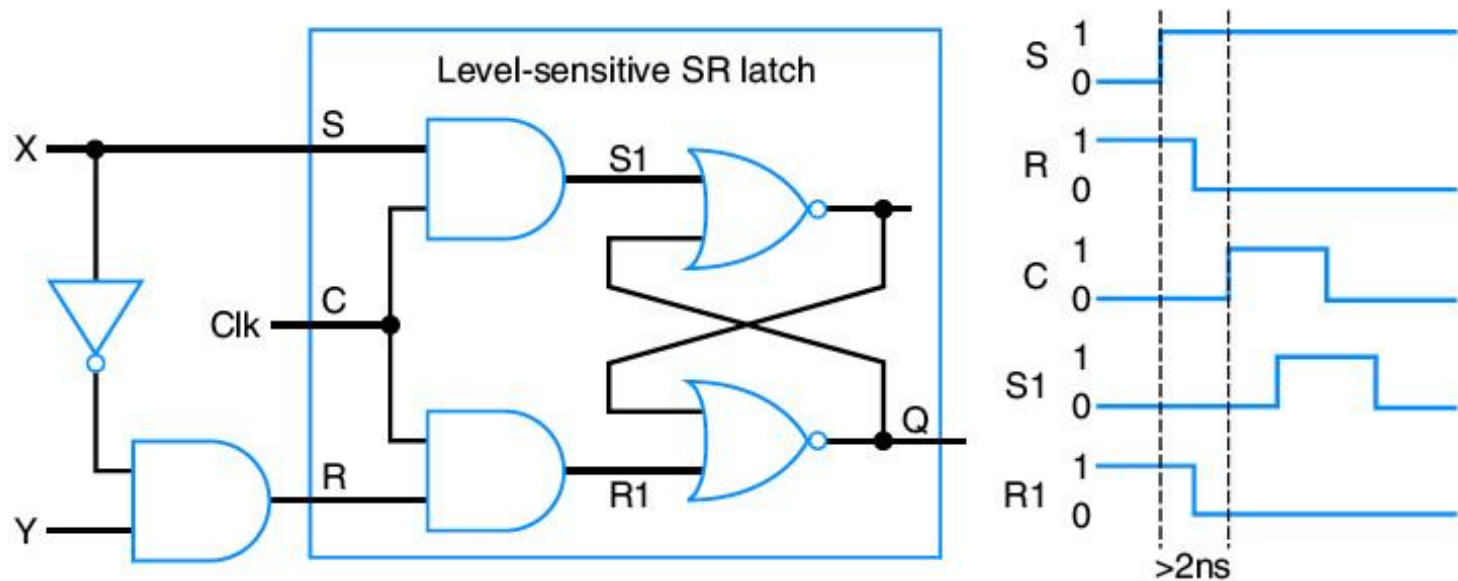


Figure 3.15 Level-sensitive SR latch—an SR latch with enable input C.

Latch SR sensível a nível

- ▶ Chamado de latch transparente
- ▶ Quando $C = 1$, o latch SR interno é transparente às entradas S e R, ou seja, estas entradas comandam o latch
- ▶ Quando $C = 0$ a saída se mantém
- ▶ As saídas são invertidas entre si

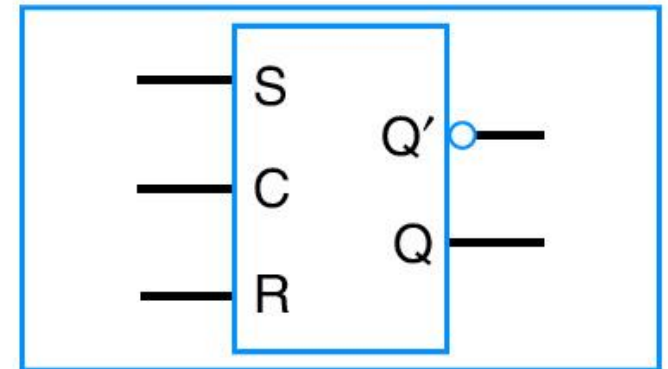


Figure 3.16 Symbol for dual-output level-sensitive SR latch.

Relógio ou clock

- ▶ Latch SR sensível a nível:
 - $C = 0$, S e R podem mudar
 - $C = 1$, S e R estáveis (não podem mudar).
- ▶ Como decidir quando tornar C igual a 1?
- ▶ Usar um sinal de habilitação que pulsa a uma taxa
 - relógio ou clock (Clk)
 - $C = \text{clk}$

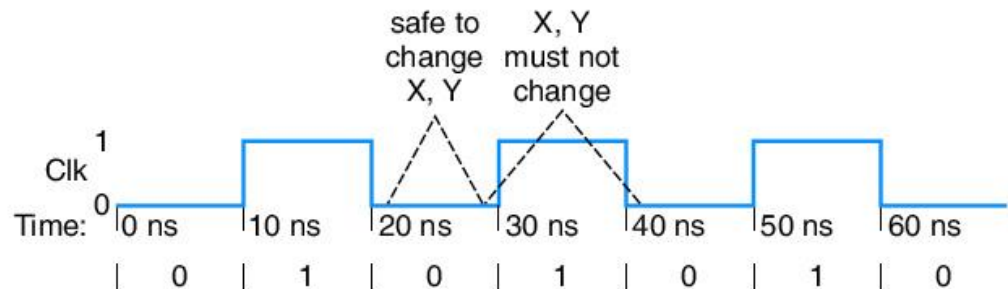


Figure 3.17 An example of a clock signal named *Clk*. Circuit inputs should only change while $\text{Clk} = 0$, such that latch inputs will be stable when $\text{Clk} = 1$.

Circuito Síncrono

- ▶ Circuito cujos elementos de armazenamento podem sofrer mudanças apenas quando um sinal de relógio está ativo
- ▶ Um circuito sequencial que não usa um relógio é chamado de circuito assíncrono
 - Não abordado neste curso.
 - Projeto digital mais avançado.
- ▶ A maioria dos circuitos sequenciais projetados e em uso atualmente são síncronos

Relógio ou clock

- ▶ Usa-se um oscilador para gerar um sinal de clk:
 - Circuito que gera uma saída que se alterna entre 1 e 0 a uma frequência constante
- ▶ Período (T): intervalo entre pulsos, medido em (s)
- ▶ Ciclo de relógio: um desses intervalos de tempo
- ▶ Frequência (F): número de ciclos por intervalo de tempo do relógio, medido em (Hz)
 - $F = 1 / T$

Freq	Period
100 GHz	0.01 ns
10 GHz	0.1 ns
1 GHz	1 ns
100 MHz	10 ns
10 MHz	100 ns

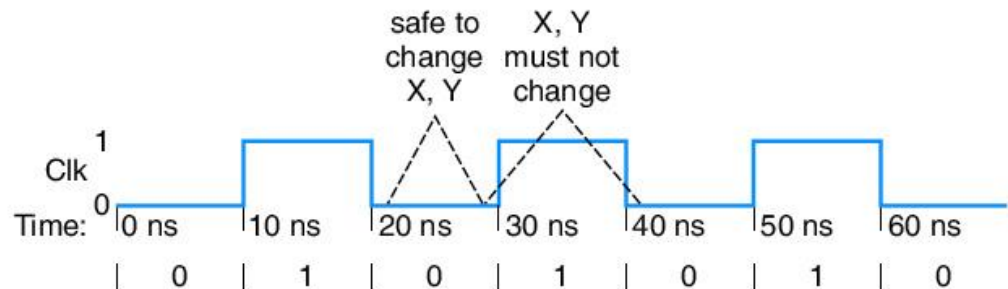


Figure 3.17 An example of a clock signal named *Clk*. Circuit inputs should only change while *Clk* = 0, such that latch inputs will be stable when *Clk* = 1.

Latch D – Sensível a Nível

- ▶ Porta inversora garante que S e R nunca serão 1 ao mesmo tempo

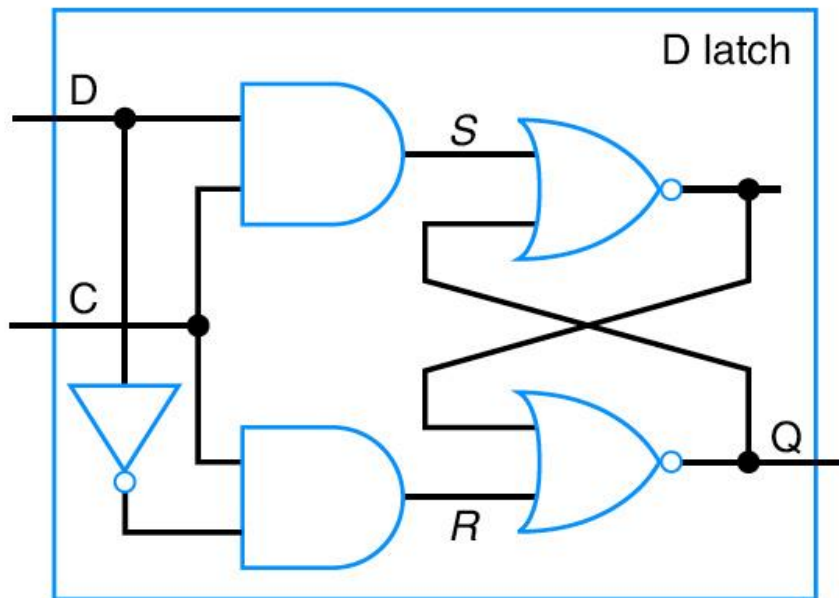


Figure 3.18 D latch internals.

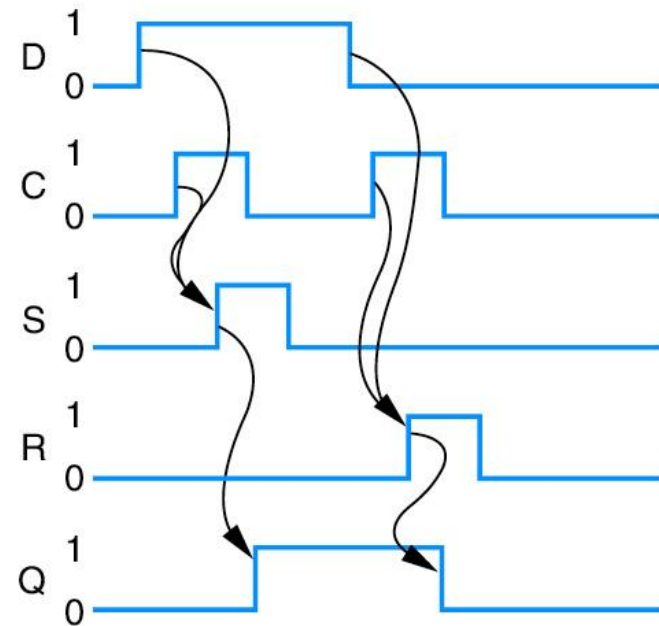
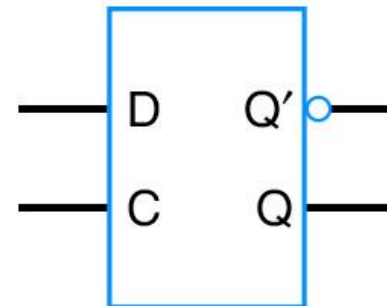


Figure 3.20
D latch symbol.



Problema dos latches sensíveis a nível

- ▶ Problema de Propagação:
- ▶ Quando $C=1$, por quantos latches o sinal vai se propagar?

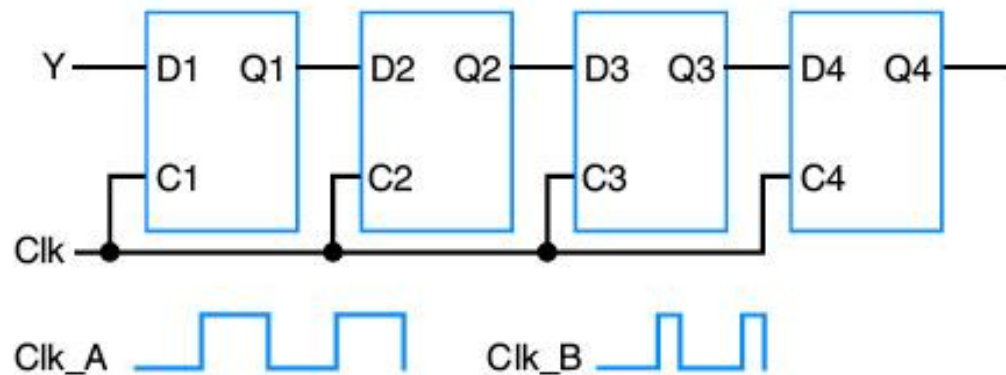


Figure 3.21 A problem with latches—through how many latches will Y propagate for each pulse of Clk_A? For Clk_B?

Problema dos latches sensíveis a nível

- ▶ Se C fica em 1 por um tempo longo demais o sinal pode se propagar por mais de um latch por vez (Clk_A)
- ▶ Se por um tempo curto demais pode não haver tempo para que o bit na entrada D do latch torne-se estável no circuito de realimentação (Clk_B)

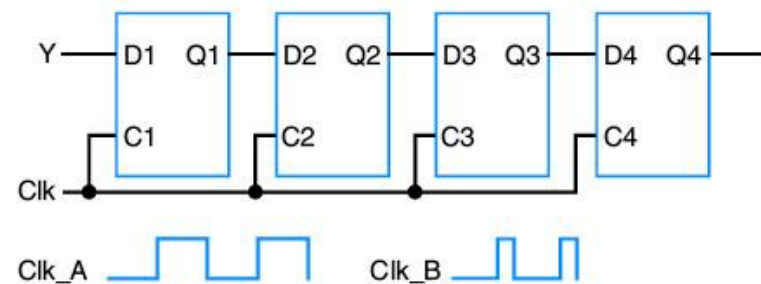


Figure 3.21 A problem with latches—through how many latches will Y propagate for each pulse of Clk_A? For Clk_B?

Problema dos latches sensíveis a nível

- ▶ **Solução:**
- ▶ Projetar um bloco sensível à borda
- ▶ Armazena o bit da entrada no instante em que o relógio sobe de 0 para 1
- ▶ O bit que é armazenado no bloco é o bit que se encontrava estável na entrada D no instante em que o relógio sobe de 0 para 1
- ▶ **Flip-flop D sensível à borda**

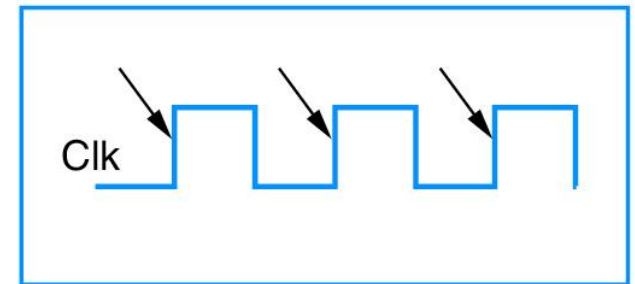


Figure 3.23 Rising clock edges.

Flip-flop D Mestre-escravo

- ▶ Para implementar um Flip-flop sensível à borda, pode-se usar dois latches
- ▶ Mestre habilitado quando $\text{Clk} = 0$ e Servo habilitado quando $\text{Clk} = 1$
 - $\text{Clk} = 0$, D é armazenado no mestre, atualizando Q_m e D_m
 - $\text{Clk} = 1$, mestre desabilitado, retém o valor do bit que estava na entrada D, antes de Clk mudar de 0 para 1
 - $\text{Clk} = 1$, servo habilitado e armazena o bit que o mestre já tinha armazenado
- ▶ Resultado: armazena-se o bit na subida do Clk

Flip-flop D Mestre-escravo

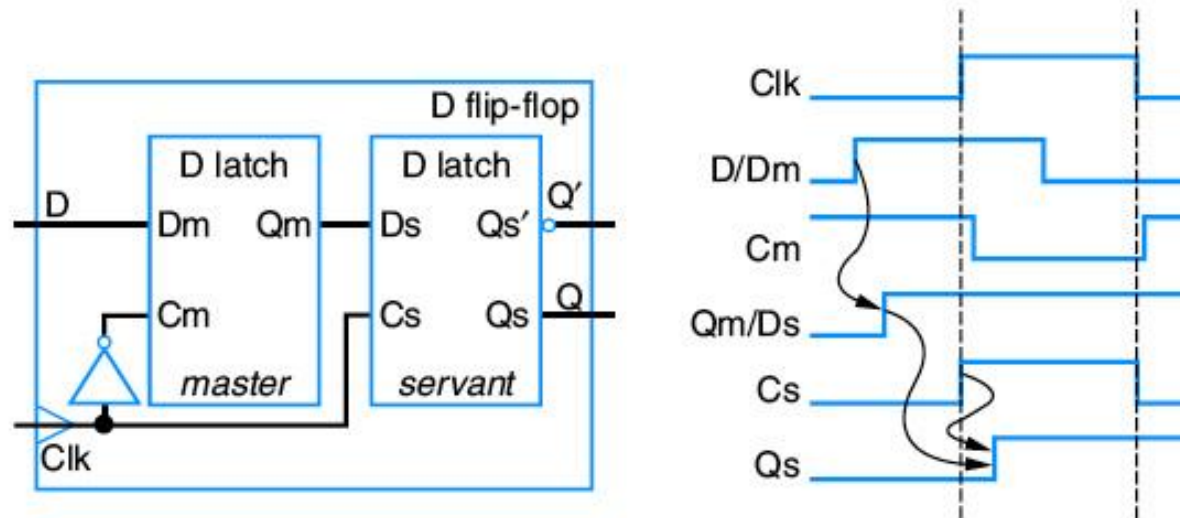
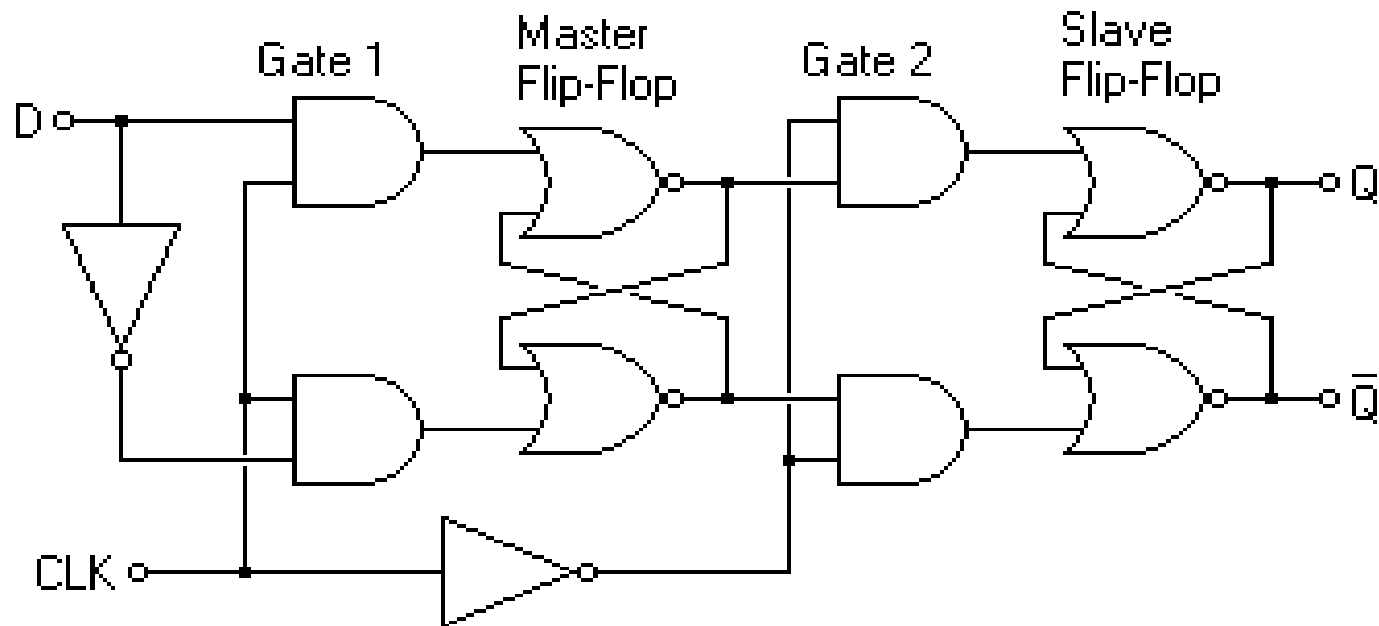


Figure 3.24 A D flip-flop implementing an edge-triggered bit storage block, internally using two latches in a master-slave arrangement. The master D latch stores its D_m input while C_lk = 0, but the new value appearing at Q_m and hence at D_s does not get stored into the servant latch, because the servant latch is disabled when C_lk = 0. When C_lk becomes 1, the servant D latch becomes enabled and thus gets loaded with whatever value was in the master latch just before C_lk changed from 0 to 1.

Flip-flop D Mestre-escravo

- ▶ Sensível à borda de subida
 - CLK = 1 muda o mestre,
 - CLK = 0 muda o escravo.



Flip-flops D

- ▶ Flip-flop D sensível à borda de subida (esquerda) e sensível à borda de descida (direita)
- ▶ O triângulo representa um flip-flop sensível à borda

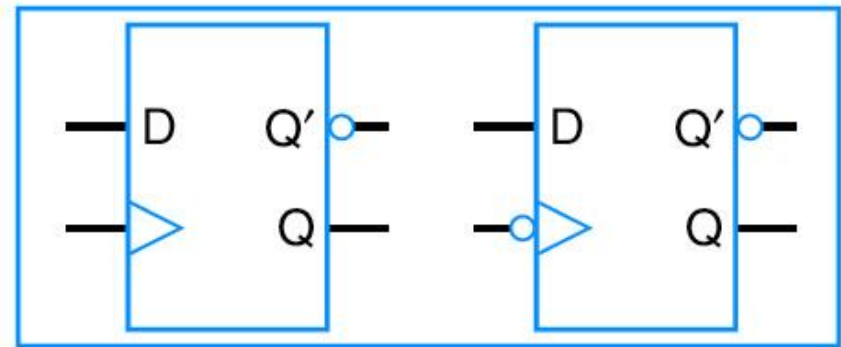
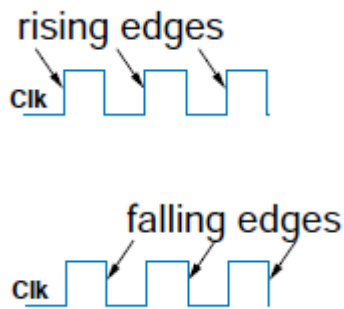


Figure 3.26 Positive (shown on the left) and negative (right) edge-triggered D flip-flops. The sideways triangle input represents an edge-triggered clock input.

Flip-flops D

- ▶ Flip-flop D sensível à borda de subida (esquerda) e sensível à borda de descida (direita)
- ▶ O triângulo representa um flip-flop sensível à borda

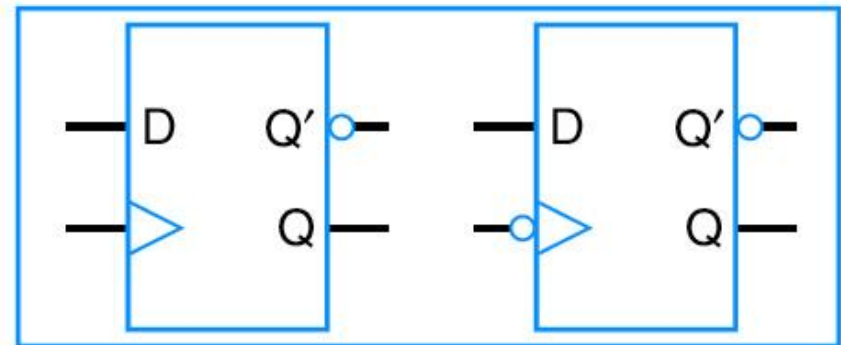
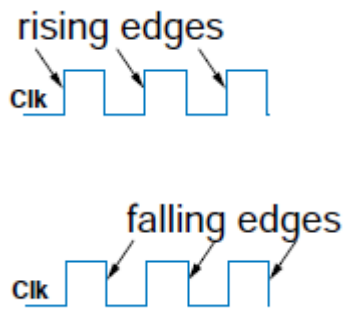


Figure 3.26 Positive (shown on the left) and negative (right) edge-triggered D flip-flops. The sideways triangle input represents an edge-triggered clock input.

Exemplo – Chamar o comissário de bordo

- ▶ Usando um flip-flop D
- ▶ Pela tabela-verdade: $D = \text{Call} + \text{cancel}' \cdot Q$

TABLE 3.1 D truth table for call-button system.

Call	Cancel	Q	D
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

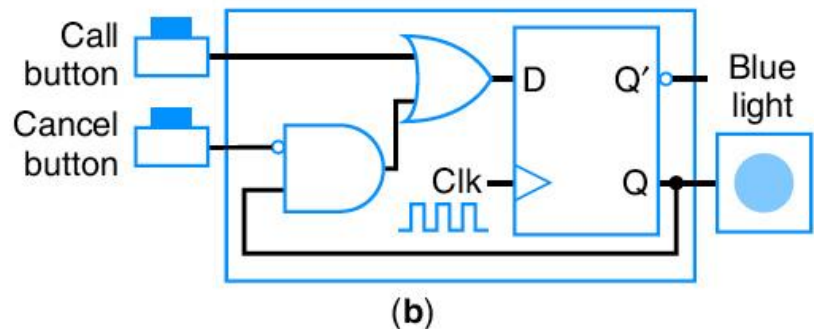
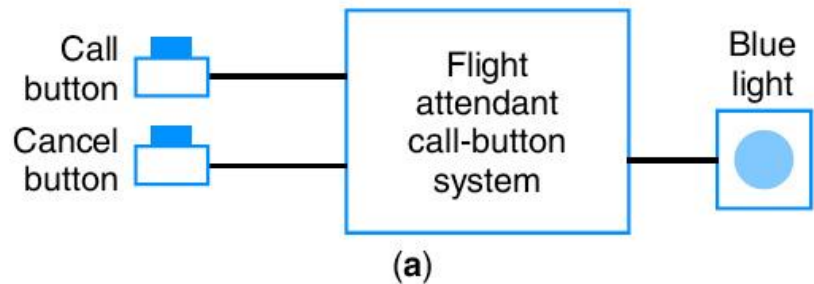


Figure 3.28 Flight attendant call-button system: (a) block diagram, and (b) implemented using a D flip-flop.

FIM