Circuitos Lógicos Sequenciais e Máquinas de Estados

Prof. Ilan Sousa Correa

Universidade Federal do Pará (UFPA)

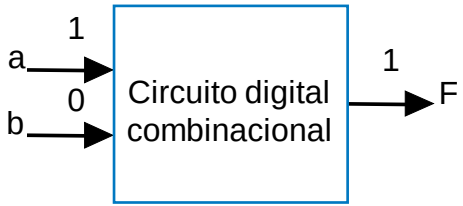
Instituto de Tecnologia (ITEC)

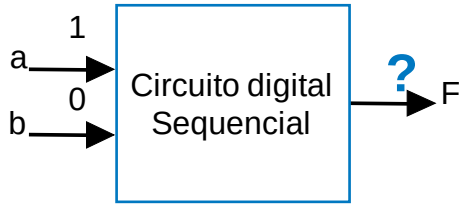
Faculdade de Eng. da Computação e Telecomunicações (FCT)

Introdução

Circuitos Sequenciais

- São circuitos para os quais o resultado da saída depende das entradas e também de valores anteriores (internos, da saída ou das entradas)
  - Armazena bits, podendo ser chamado também de “estado”
  - Exemplo: contador binário
- Geração de entradas a partir de constantes no código VHDL





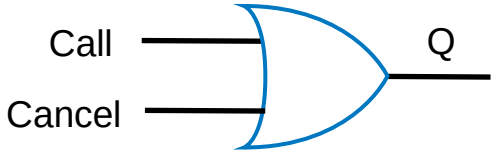
Deve-se conhecer os valores anteriores para saber o valor da saída

Exemplo de armazenamento de bit

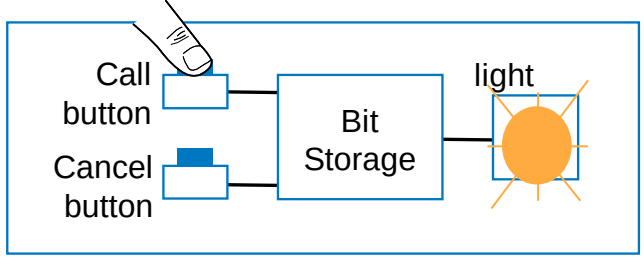
Botão de serviço

- Ao pressionar o botão “call”, uma lâmpada permanece acesa
- Ao pressionar o botão “cancel”, a lâmpada é desligada

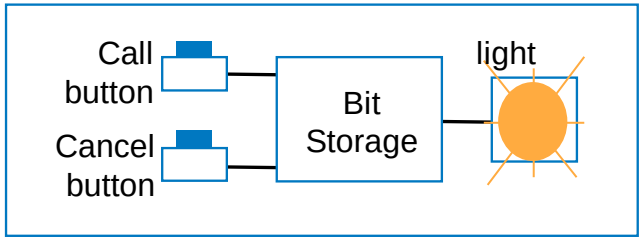
Como implementar com lógica combinacional?



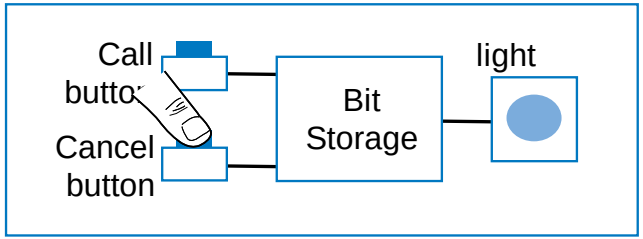
Não funciona. A saída Q não permanece com o seu valor quando os botões são liberados.



1. Call button pressed – light turns on



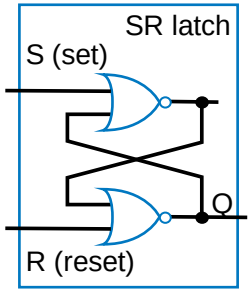
*2. Call button released – light **stays on***



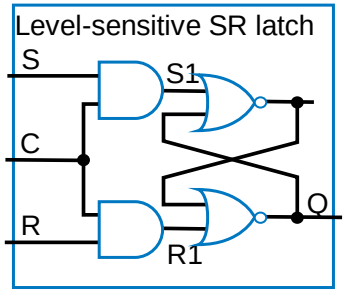
3. Cancel button pressed – light turns off

Exemplo de armazenamento de bit

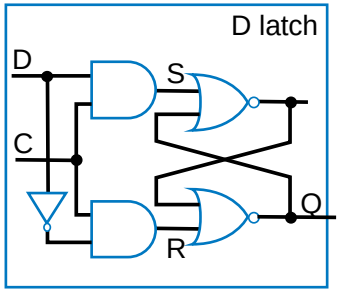
Soluções para armazenamento do valor



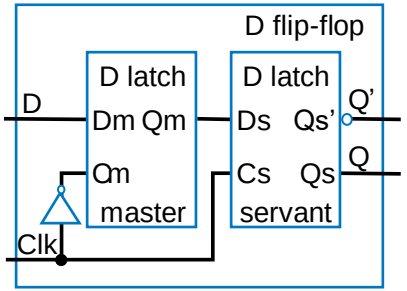
S=1 resulta em Q=1, R=1 resulta Q=0. Problema: SR=11 resulta em Q indefinido.



S e R somente surtem efeito quando C=1. Pode-se projetar um circuito externo para evitar SR=11 quando C=1.

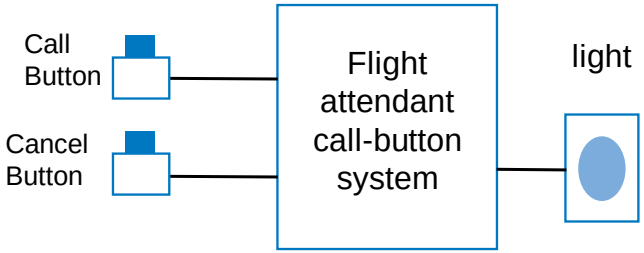


SR não pode ser 11 e passa os valores para a saída enquanto C=1



Para D o valor de D para a sáida Q na borda de subida do sinal Clk. Usa mais portas lógicas, mas resolve o problema de passar valores indefinidamente.

Exemplo de armazenamento de bit

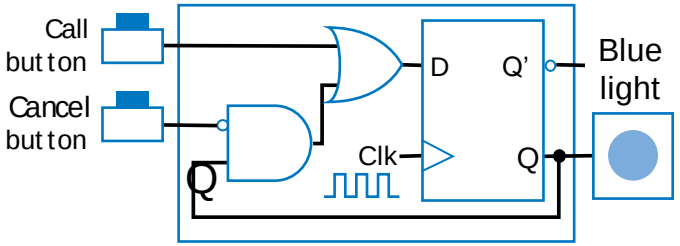


Botão de serviço

- Ao pressionar o botão “call”, uma lâmpada permanece acesa
- Ao pressionar o botão “cancel”, a lâmpada é desligada

Call	Cancel	Q	D	
0	0	0	0	Preserva o valor de Q
0	0	1	1	
0	1	0	0	Cancela, faz Q=0
0	1	1	0	
1	0	0	1	Inicia a chamada, faz Q=1
1	0	1	1	
1	1	0	1	Situação a definir. Daremos prioridade para “call”
1	1	1	1	

Implementação

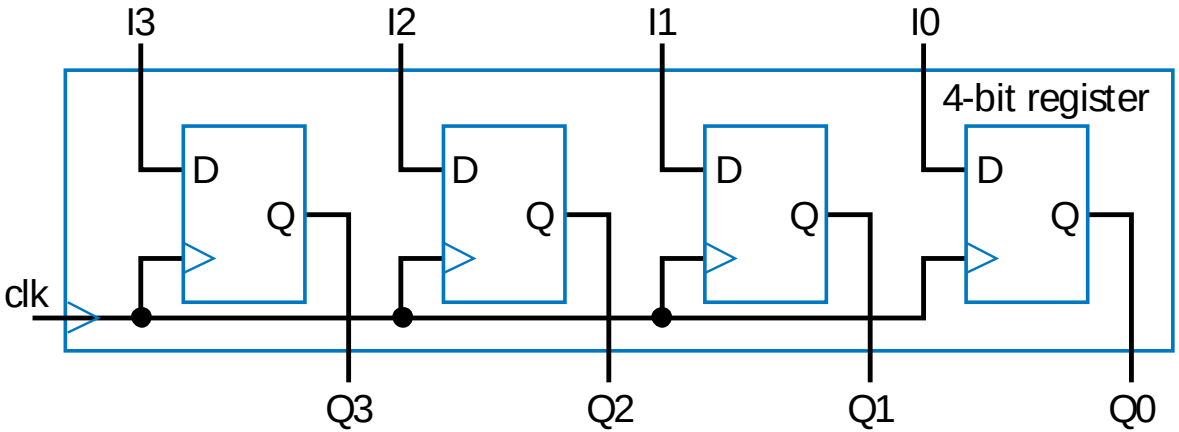


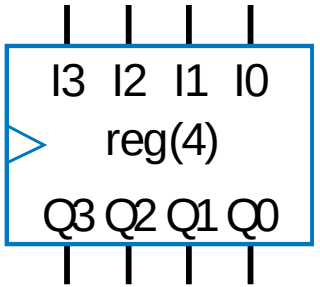
Registradores

Armazenamento de múltiplos bits

- Exemplo: armazenamento de um número binário de quatro bits

Registrador: múltiplos flip-flops compartilhando um sinal de clock

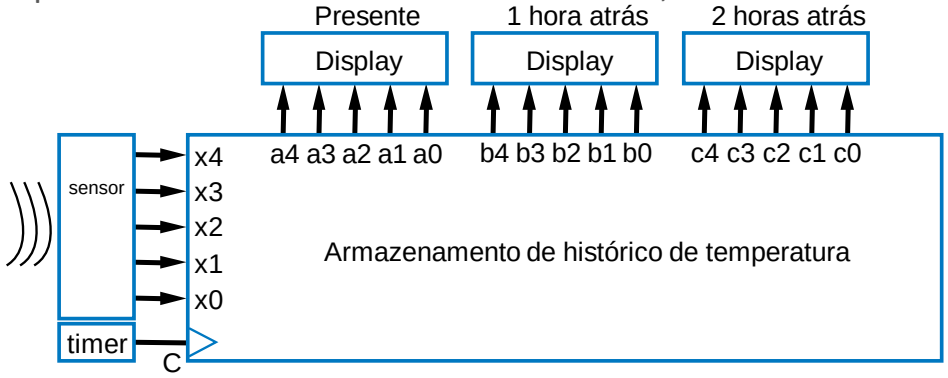




Registradores

Armazenamento de múltiplos bits

- Exemplo: mostrador de histórico de temperatura
  - Um sensor disponibiliza uma medição de temperatura como um número de 5 bits
  - Um temporizador gera um pulso C a cada hora
  - A valor de temperatura é salvo com a borda de subida C, e os três últimos valores são mostrados.

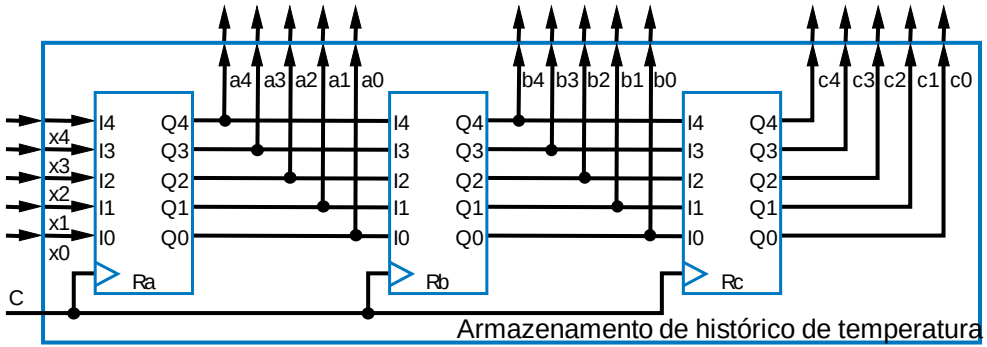


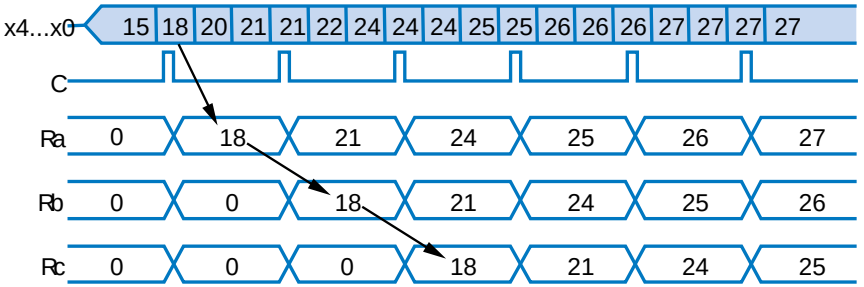
Na prática não se usa um sinal que não seja um clock (sinal periódico) na entrada C

Registradores

Armazenamento de múltiplos bits

- Exemplo: mostrador de histórico de temperatura
  - Implementação interna e propagação dos sinais

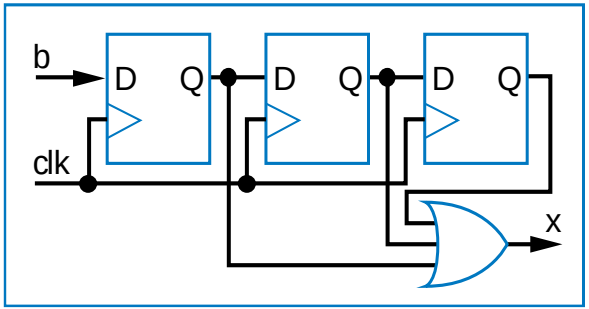




Implementação de um comportamento específico

Requisito: Necessita-se que uma determinada saída X seja 1 por três unidades de tempo

- Implementação
  - O botão b é pressionado
  - A saída X=1 por três ciclos de clock
    - b=1 é armazenado no primeiro flip-flop,
    - Então, no segundo, e então no terceiro,
    - Com uma porta OR com entradas como as saídas de cada flip-flop é possível obter o comportamento especificado?
  - Como fazer para comportamentos/sequencias mais complexas?



Máquinas de estados finitos e Controladores

Implementação de um comportamento específico

Como projetar circuitos sequenciais mais complexos?

- Tentativa e erro pode não ser uma boa abordagem
- É possível projetar circuitos dessa forma para alguns comportamentos
    - Contagem crescente (e.g., 1 até 9); Pulso (nível alto) por um ciclo de clock a cada 10 ciclos de clock; detecção da sequência binária correspondente a 1, 3 e 5 a partir de uma sequência de entradas de 3 bits.
    - É possível prever comportamentos indesejados. Para o exemplo do slides anterior, o que acontece se o botão é pressionado novamente enquanto x=1? O que acontece se o botão for mantido pressionado por mais de um ciclo de clock?

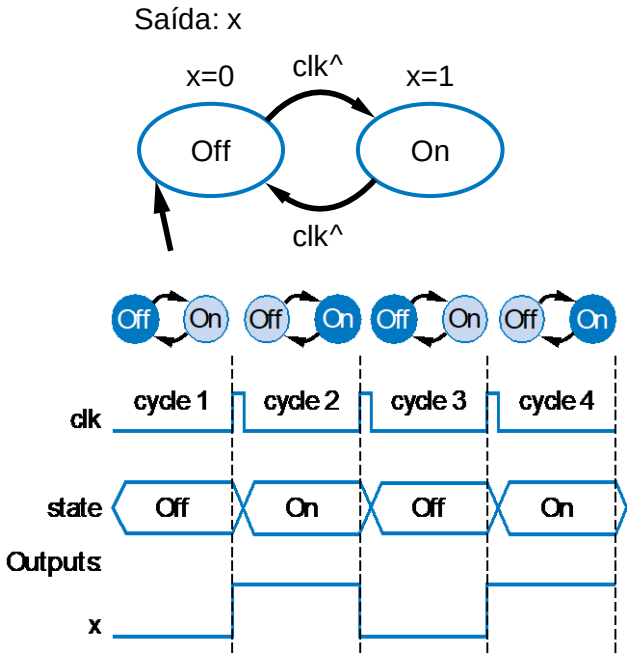
Máquina de estados finitos

Uma forma de modelar um comportamento

- Aplicável em vários tipos de modelagem
- Pode-se utilizar para modelar circuitos sequenciais

Definição: Lista de estados e transições

- Exemplo: deseja-se um circuito digital que mude o valor de uma saída de 0→1 e 1→0 a cada ciclo de clock
- Dois estados: “Off” (x=0) e “On” (x=1)
- Transições Off→On e On→Off na borda de subida do clock
- Estado inicial: transição partindo de nenhum estado



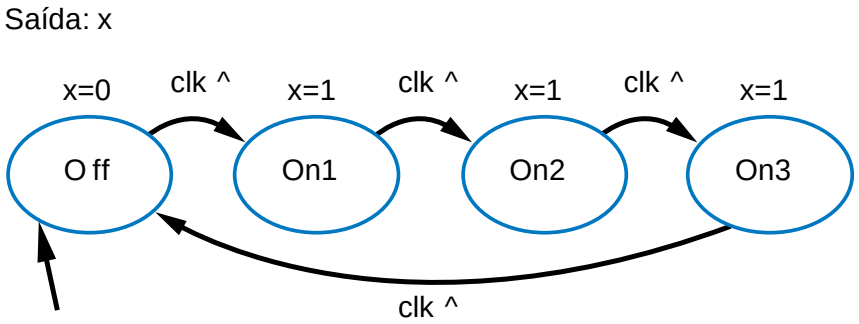
Máquina de estados finitos

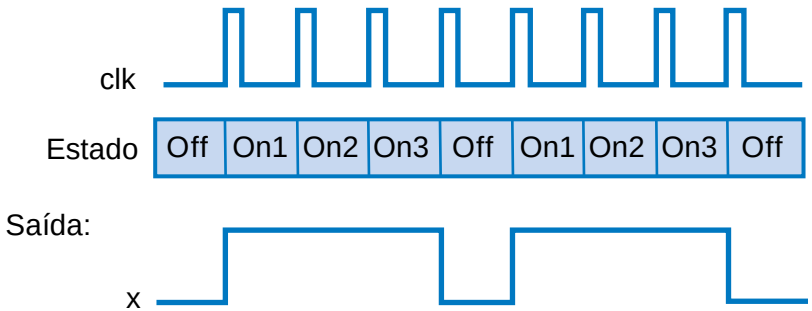
Uma forma de modelar um comportamento

- Aplicável em vários tipos de modelagem
- Pode-se utilizar para modelar circuitos sequenciais

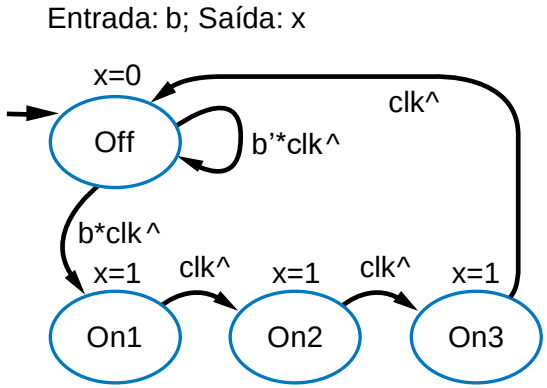
Definição: Lista de estados e transições

- Exemplo: deseja-se um circuito digital que mude ciclicamente o valor de uma saída de acordo com a sequência: 0, 1, 1, 1, 0, 1, 1, 1
- Quatro estados: “Off”, “On1”, “On2” e “On3”



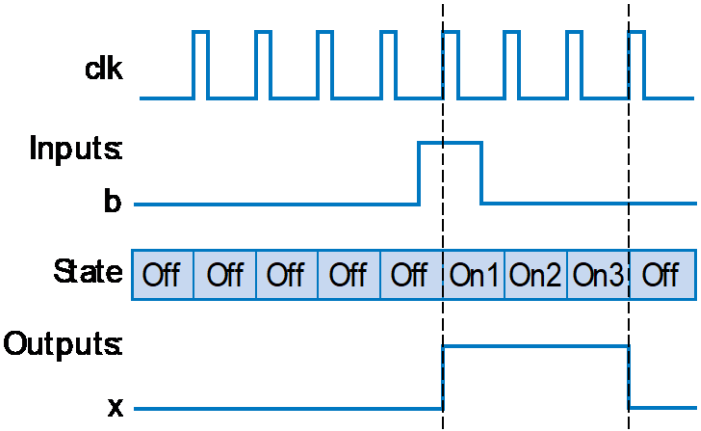


Máquina de estados finitos



Quatro estados

- Aguarda em “Off” enquanto uma entrada b=0
- Quando b=1 (na borda de subida do clock), transiciona para “On1”
  - Com X=1
  - Nas próximas duas bordas de subida, transiciona para On2 e On3, com X=1
- Resumindo, x=1 por três ciclos após um botão ser pressionado



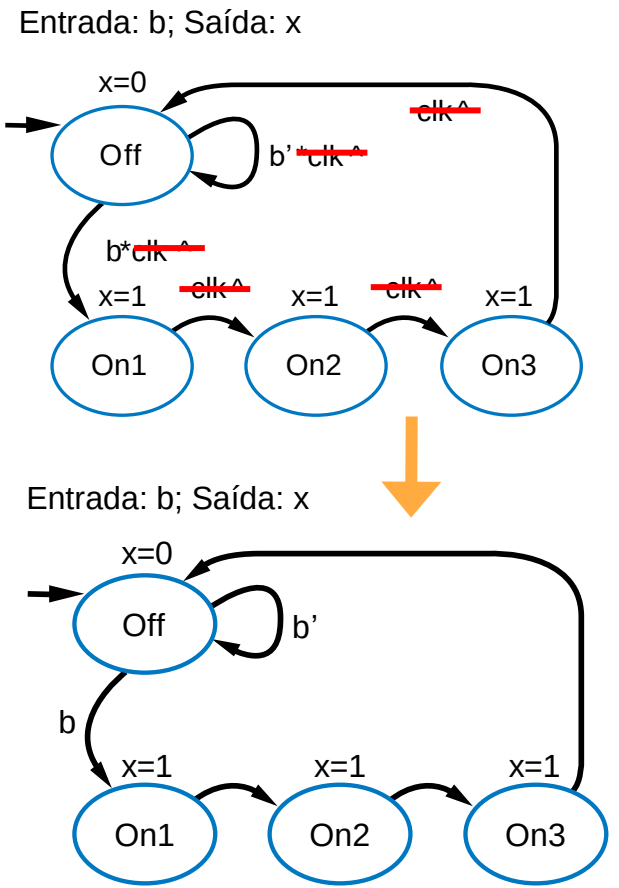
Máquina de estados finitos

Trata-se de circuitos digitais, logo, tudo será síncrono com a borda do clock (subida ou descida)

- Pode-se omitir o clock das transições
- Máquinas de estados finitos **síncrona**

Automatos finitos

- Automatos finitos determinísticos e não-deterministicos

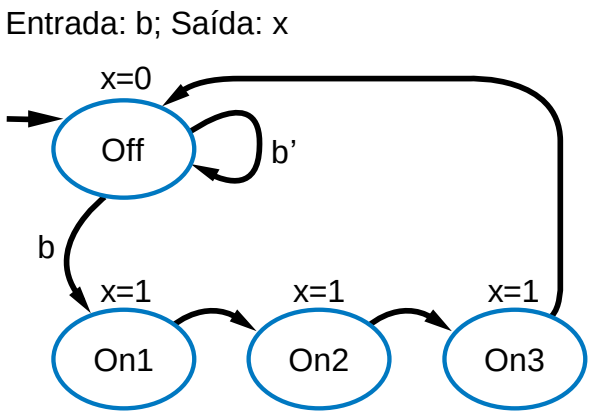


Máquina de estados finitos

Definição de uma máquina de estados

- Conjunto de estados
  - Ex: {Off, On1, On2, On3} com indicação de estado inicial
- Conjuntos de entradas e saídas
  - Ex: entradas: {x}; e saídas: {b}
- Conjunto de transições
  - Pode-se nomear as transições
- Conjunto de ações
  - Atribuições às saídas em cada estado
  - Ex: x=1, x=1, x=1, e x=1

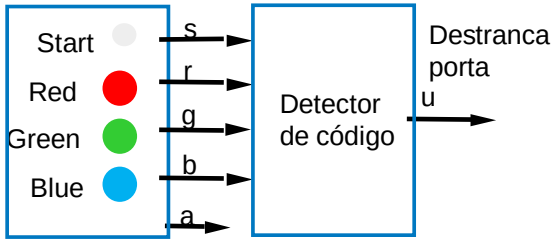
Representação gráfica de uma máquina de estados: ***Diagrama de estados.***



Exemplo mais completo: Detector de código

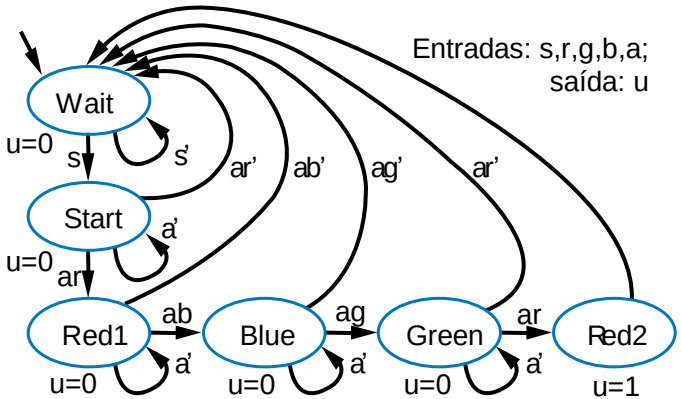
Destrancar uma porta (u=1) quando botões forem pressionados na sequência correta: start, red, blue, green, red

- Variáveis/bits indicando cada botão: s, r, g, b
- Bit auxiliar indicando qualquer outro botão pressionado: a



Máquina de estados:

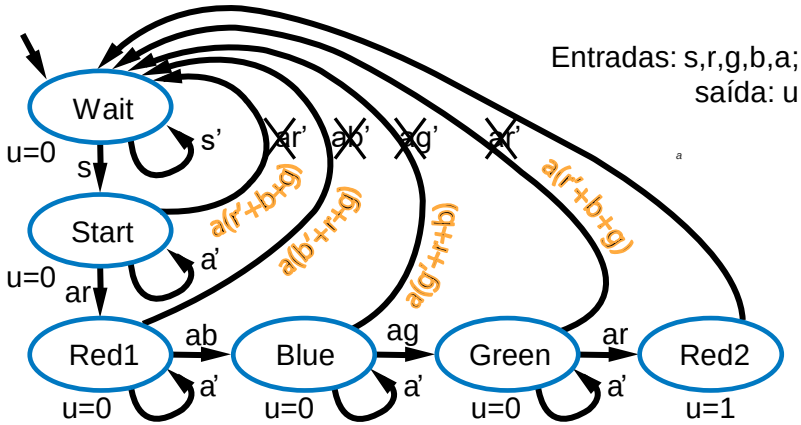
- Aguarda s=1 no estado “Wait”. Após o início
  - Se r=1, então vá para Red1
  - Então, se b=1, vá para Blue
  - Então, se g=1, vá para Green
  - Então, se r=1, vá para Red2 (neste estado destranca a porta, u=1)
  - Botão errado em qualquer estado, retorna para Wait sem abrir a porta



Exemplo mais completo: Detector de código

Destrancar uma porta (u=1) quando botões forem pressionados na sequência correta: start, red, blue, green, red

- Transições considerando o pressionamento do botão errado (sequência incorreta)
- Máquina de estados permite fornecer formalização meios para definição de comportamento desejado

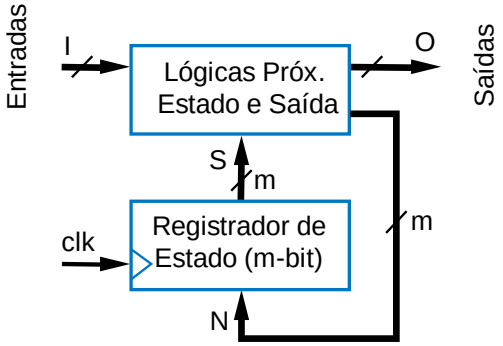


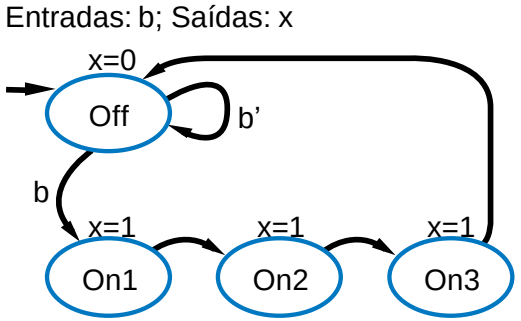
Implementação de Máquinas de estados finitos a partir de circuitos lógicos

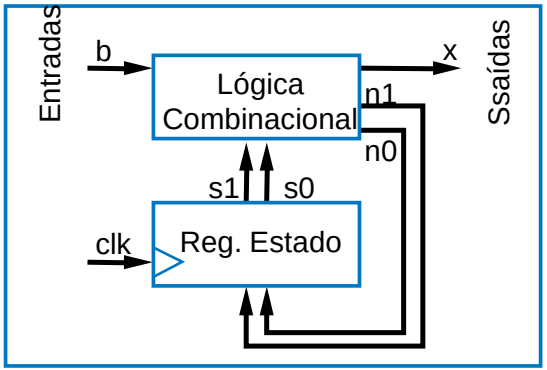
Exemplo mais completo: Detector de código

Implementação uma máquina de estados com circuitos lógicos sequenciais e combinacionais

- Formato padrão
  - Registrador de estados para armazenar o estado atual
  - Duas lógicas combinacionais para calcular o próximo estado e as saídas
    - Lógicas de **próximo estado** e de **saída**







Exemplo mais completo: Detector de código

Implementação uma máquina de estados com circuitos lógicos sequenciais e combinacionais

- Passos

	Passo	Descrição
Passo 1	Capture a FSM	Crie uma FSM que descreva o comportamento desejado do bloco de controle.
Passo 2	Crie a arquitetura	Crie a arquitetura padrão usando um registrador de estado com largura apropriada e uma lógica combinacional, cujas entradas são os bits do registrador de estado e as entradas da FSM e cujas saídas são os bits de próximo estado e as saídas da FSM.
Passo 3	Codifique os estados	Atribua um número binário único a cada um dos estados. Cada número binário que representa um estado é conhecido como uma *codificação*. Qualquer codificação poderá ser usada desde que cada estado tenha uma codificação única.
Passo 4	Crie a tabela de estados	Crie uma tabela-verdade para a lógica combinacional de modo tal que a lógica irá gerar as saídas e os sinais de próximo estado corretos para a FSM. Ordenando as entradas primeiro com os bits de estado faz com que a tabela-verdade descreva o comportamento dos estados. Assim, a tabela é uma tabela-verdade.
Passo 5	Implemente a lógica combinacional	Implemente a lógica combinacional usando qualquer método.

Exemplo mais completo: Detector de código

Implementação uma máquina de estados com circuitos lógicos sequenciais e combinacionais

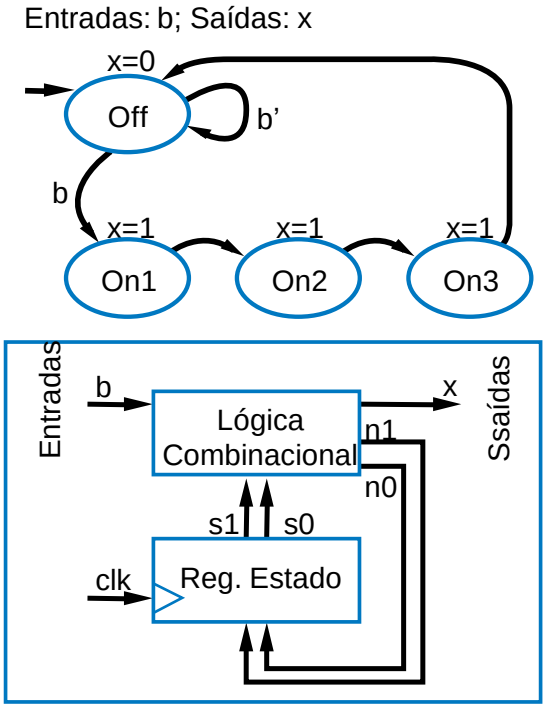
Passo 1: Captura do comportamento como um máquina de estados

Passo 2: Criação da arquitetura do circuito lógico

- Registrador de 2 bits para 4 estados
- Entrada b e saída x
- Sinais de próximo estado n1 e n0

Passo 3: codificação de estados

- Atribuição de sequências binárias únicas para cada estado

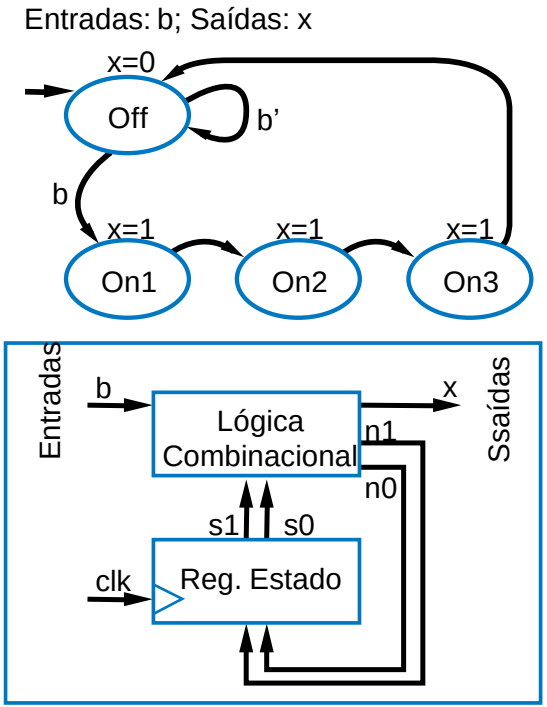


Exemplo mais completo: Detector de código

Implementação uma máquina de estados com circuitos lógicos sequenciais e combinacionais

Passo 4: Tabela de estado

	Inputs			Outputs		
	s1	s0	b	x	n1	n0
Off	0	0	0	0	0	0
	0	0	1	0	0	1
On1	0	1	0	1	1	0
	0	1	1	1	1	0
On2	1	0	0	1	1	1
	1	0	1	1	1	1
On3	1	1	0	1	0	0
	1	1	1	1	0	0



Exemplo mais completo: Detector de código

Implementação uma máquina de estados com circuitos lógicos sequenciais e combinacionais

Passo 5: Implementação da lógica combinacional

	Inputs			Outputs		
	s1	s0	b	x	n1	n0
Off	0	0	0	0	0	0
	0	0	1	0	0	1
On1	0	1	0	1	1	0
	0	1	1	1	1	0
On2	1	0	0	1	1	1
	1	0	1	1	1	1
On3	1	1	0	1	0	0
	1	1	1	1	0	0

x = s1 + s0

n1 = s1's0b' + s1's0b + s1s0'b' + s1s0'b

n1 = s1's0 + s1s0'

n0 = s1's0'b + s1s0'b' + s1s0'b

n0 = s1's0'b + s1s0'

Exemplo mais completo: Detector de código

Implementação uma máquina de estados com circuitos lógicos sequenciais e combinacionais

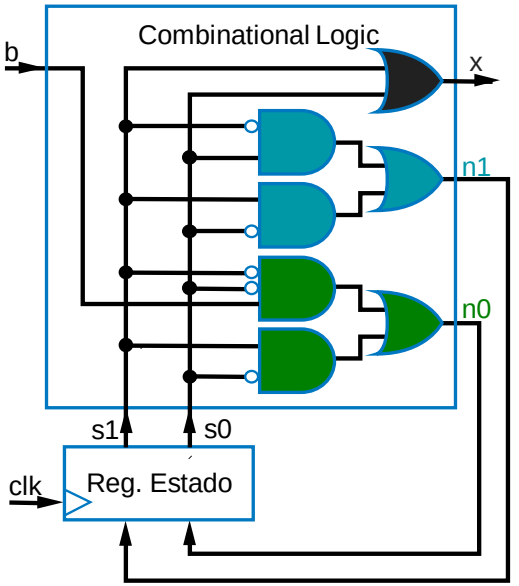
Passo 5: Implementação da lógica combinacional

	Inputs			Outputs		
	s1	s0	b	x	n1	n0
Off	0	0	0	0	0	0
	0	0	1	0	0	1
On1	0	1	0	1	1	0
	0	1	1	1	1	0
On2	1	0	0	1	1	1
	1	0	1	1	1	1
On3	1	1	0	1	0	0
	1	1	1	1	0	0

x = s1 + s0

n1 = s1's0 + s1s0'

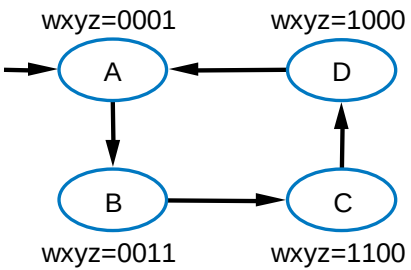
n0 = s1's0'b + s1s0'



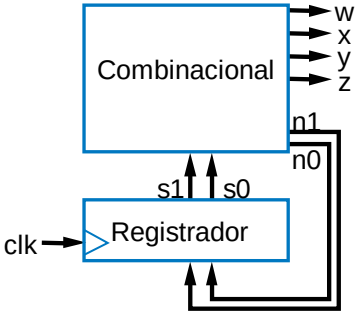
Outros exemplo: Gerador de sequência

Circuito para gerar a sequência 0001, 0011, 1100, 1000 em loop cada valor em um ciclo de clock

Entradas: -; saídas: w,x,y,z

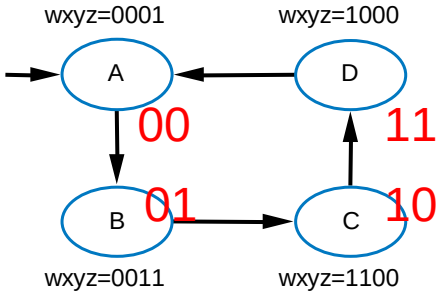


Passo 1: Captura a máquina



Passo 2: arquitetura

Entradas: -; saídas: w,x,y,z

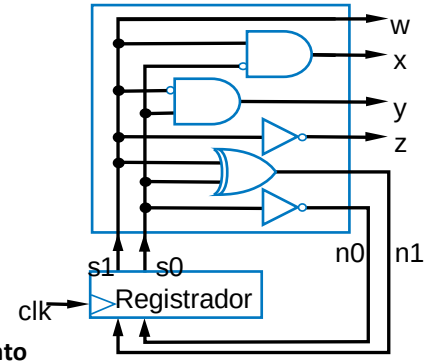


Passo 3: Codificação de estados

Passo 4: tabela de estados

	Inputs		Outputs					
	s1	s0	w	x	y	z	n1	n0
A	0	0	0	0	0	1	0	1
B	0	1	0	0	1	1	1	0
C	1	0	1	1	0	0	1	1
D	1	1	1	0	0	0	0	0

w = s1
x = s1s0'
y = s1's0
z = s1'
n1 = s1 xor s0
n0 = s0'

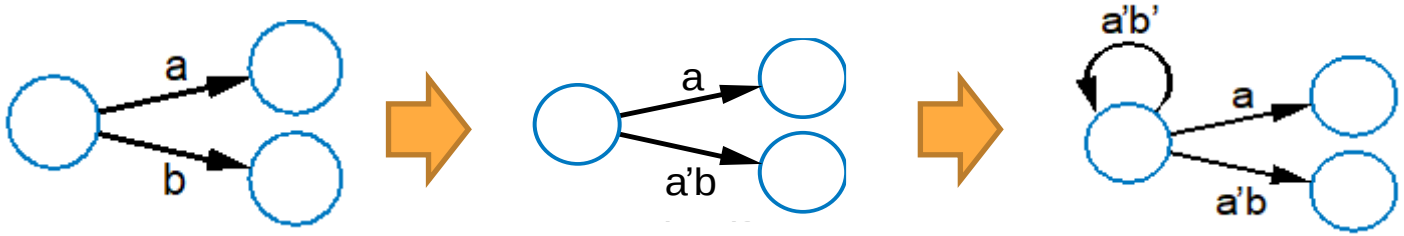


Passo 5: Circuito Combinacional

Considerações sobre as propriedades das transições

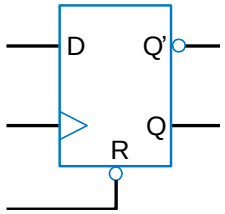
Somente uma transição deve estar ativa

- Para todas as transições partindo de um estado
- A máquina de estados deve sempre ir para um único estado



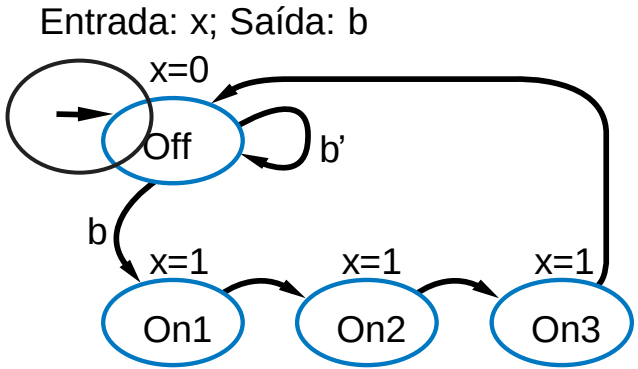
Considerações sobre estado inicioa

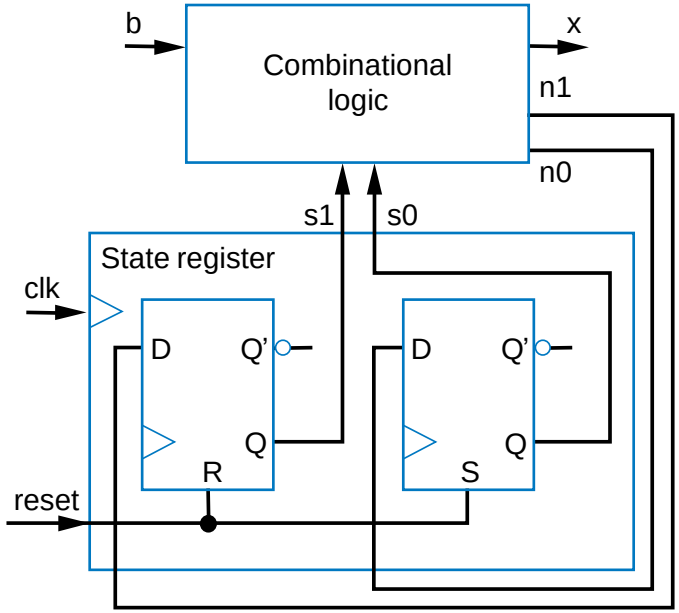
Registrador com Reset
Ativo em nível baixo



Todas as máquinas de estados devem ter um estado inicial

- Utilizar reset dos registradores



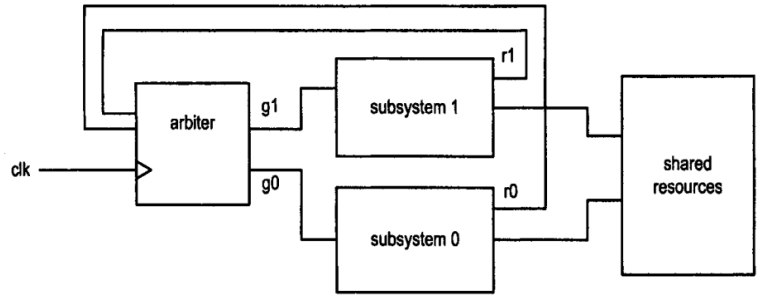


Aplicação de máquinas de estados como árbitro de barramento

Introdução à linguagem VHDL

Aplicação de máquinas de estados: árbitro de barramento

- Em um sistema computacional, um recurso pode ser compartilhado por vários subsistemas
  - Processadores podem compartilhar uma única memória
- Arbitro de barramento
  - Resolve conflitos de acesso à recursos compartilhados
- Exemplo:
  - Subsistemas comunicam-se com o árbitro para requisitar acesso (sinal rx)
  - Um subsistema somente acesso o recurso se o acesso está garantido (gx)
  - Após completar a atividade, o subsistema desabilita o requisição (rx)
  - Máquina de estados permite implementação de tal funcionalidade
  - Permite também implementação de prioridade

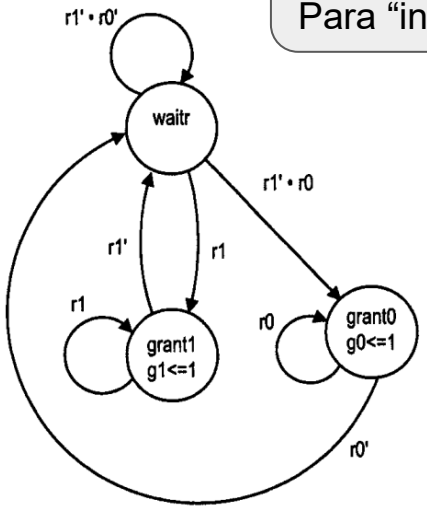


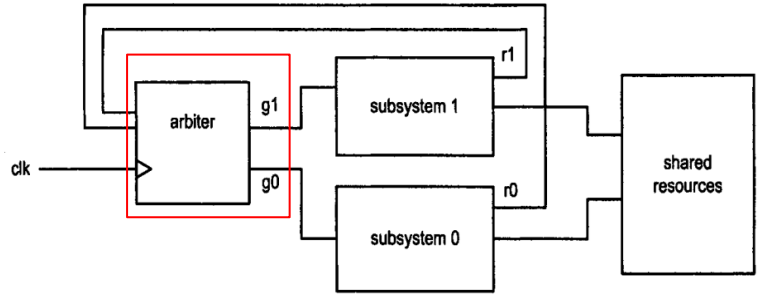
Introdução à linguagem VHDL

Aplicação de máquinas de estados: árbitro de barramento

- Exemplo:

R1 tem maior prioridade do que R0 em caso de requisição simultânea
Quando uma requisição é permitida, a outra deve esperar
Para “informar” ao árbitro sobre a finalização do acesso, Rx=0



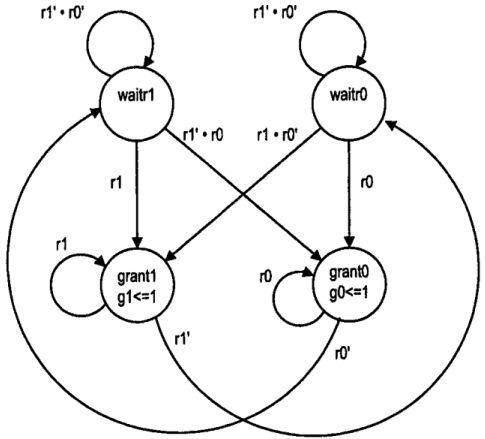


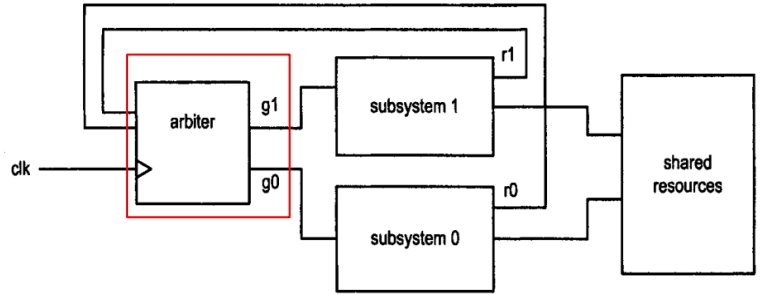
Introdução à linguagem VHDL

Aplicação de máquinas de estados: árbitro de barramento

- Exemplo:

Prioridade alternada ⇒ ao final de uma requisição, a prioridade é do outro
Quando uma requisição é permitida, a outra deve esperar
Para “informar” ao árbitro sobre a finalização do acesso, Rx=0



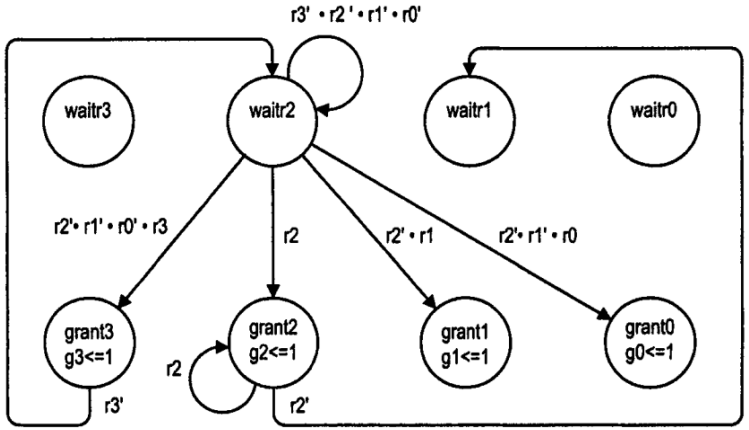


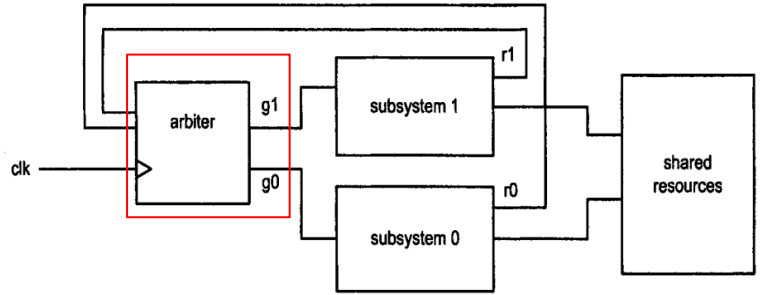
Introdução à linguagem VHDL

Aplicação de máquinas de estados: árbitro de barramento

- Exemplo:

Prioridade alternada entre quatro dispositivos
Quando uma requisição é permitida, a outra deve esperar
Para “informar” ao árbitro sobre a finalização do acesso, Rx=0



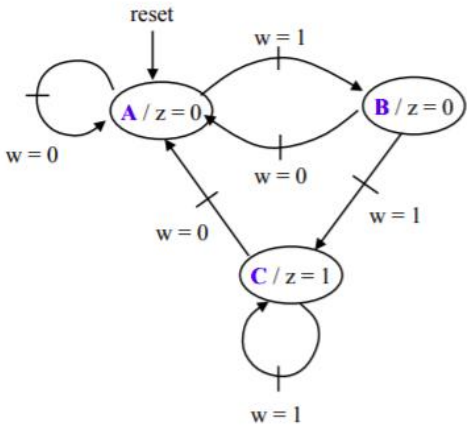


Exemplo de máquinas de estados no Quartus

Introdução à linguagem VHDL

Máquinas de estados - Exemplo no Quartus

```
4  entity fsm_aula1_6 is
5     port(
6        clk, rst, w : in  std_logic;
7        z : out std_logic
8     );
9  end fsm_aula1_6;
```



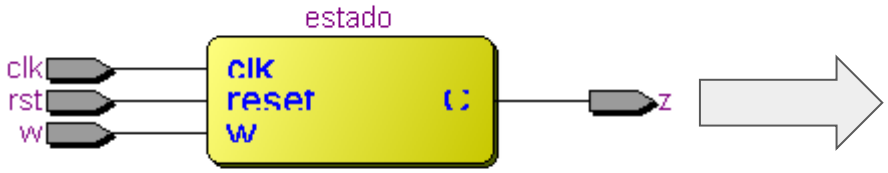

```
15    -- Circuito combinacional -> não depende de clock
16    logica_proximo_estado : process(estado, w)
17    begin
18       case estado is
19          when A =>
20             if w = '0' then
21                prox_estado <= A;
22             else
23                prox_estado <= B;
24             end if;
25          when B =>
26             if w = '0' then
27                prox_estado <= A;
28             else
29                prox_estado <= C;
30             end if;
31          when C =>
32             if w = '1' then
33                prox_estado <= C;
34             else
35                prox_estado <= A;
36             end if;
37       end case;
38    end process;
```

```
40    -- Circuito combinacional -> não depende de clock
41    logica_saida : process(estado)
42    begin
43       case estado is
44          when A => z <= '0';
45          when B => z <= '0';
46          when C => z <= '1';
47       end case;
48    end process;
49
50    registrador_estado : process(clk, rst)
51    begin
52       if rst = '1' then
53          estado <= A;
54       elsif rising_edge(clk) then
55          estado <= prox_estado;
56       end if;
57    end process;
58
59 end architecture;
```

Introdução à linguagem VHDL

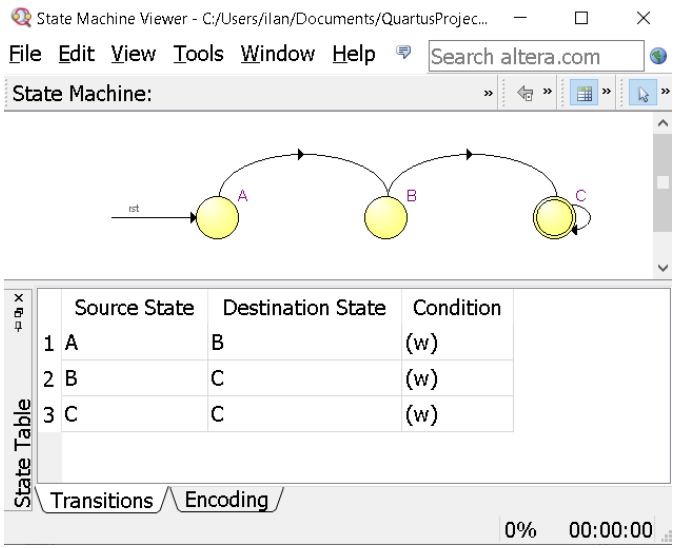
Máquinas de estados - Exemplo no Quartus

Janela RTL Viewer do Quartus
Amarelo para indicar inferência de máquina de estados



Lógicas de saída e de próximo estado são calculadas automaticamente

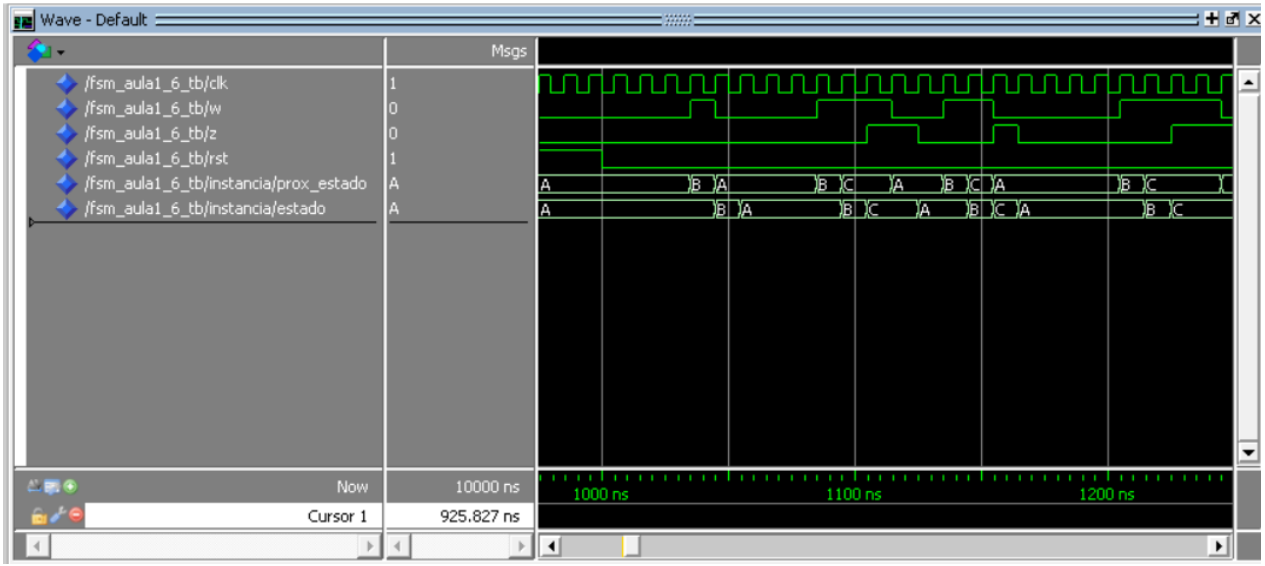
Estados codificados automaticamente



Introdução à linguagem VHDL

Máquinas de estados - Exemplo no Quartus

Simulação no Modelsim



Bibliografia

- Sistemas digitais: Projeto, Otimização e HDLs, Frank Vahid, Ed. Bookman, 1ª Ed., 2008
- Tocci, R. J., Widmer, N. S. Sistemas digitais. 7. ed. Rio de Janeiro: LTC, 1998.