



FEDERAL UNIVERSITY  
OF SANTA CATARINA

## EEL5105 – Circuitos e Técnicas Digitais

### Aula 8

---

Prof. Héctor Pettenghi

[hector@eel.ufsc.br](mailto:hector@eel.ufsc.br)

<http://hectorpettenghi.paginas.ufsc.br>

Material desenvolvido com apoio de arquivos de apresentação do livro de Frank Vahid

## **8. Projeto de Circuitos Sequenciais**

---

**8.1. Projeto de FSM (continuação)**

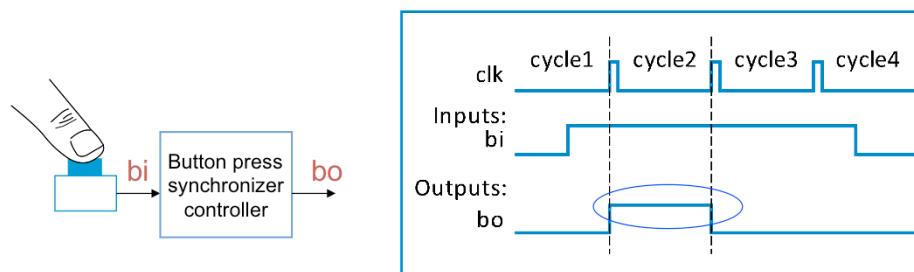
**8.2. FSM com estados de bloqueio (Lock-out)**

**8.3. Engenharia reversa de projeto FSM**

Nesta aula veremos a continuação do projeto FSM, com alguns exemplos para melhor compreensão do assunto; FSM com estados de bloqueio; e engenharia reversa de projeto FSM.

## 8.1. Projeto de FSM (continuação)

- **Exemplo 1:** Projetar um sincronizador de aperto de botão (circuito que converte um toque no botão em um pulso com duração de um ciclo de *clock*, independentemente do tempo que o botão fica apertado).



O primeiro exemplo é um sincronizador de aperto de botão. Como construir este circuito?

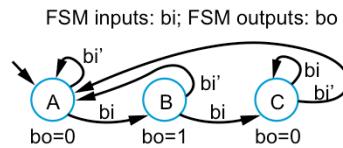
## 8.1. Projeto de FSM (continuação)

- **Exemplo 1:** Projetar um sincronizador de aperto de botão (circuito que converte um toque no botão em um pulso com duração de um ciclo de *clock*, independentemente do tempo que o botão fica apertado).
  - Especificação formal: [Capturar a FSM](#)

A primeira coisa a se fazer é capturar a FSM.

## 8.1. Projeto de FSM (continuação)

- **Exemplo 1:** Projetar um sincronizador de aperto de botão (círcuito que converte um toque no botão em um pulso com duração de um ciclo de *clock*, independentemente do tempo que o botão fica apertado).
  - Especificação formal: *Capturar a FSM*



Este deve ser o resultado.

## 8.1. Projeto de FSM (continuação)

- **Exemplo 1:** Projetar um sincronizador de aperto de botão (circuito que converte um toque no botão em um pulso com duração de um ciclo de *clock*, independentemente do tempo que o botão fica apertado).
  - **Passo 1:** *Codificar os estados*

Agora, vamos codificar os estados.

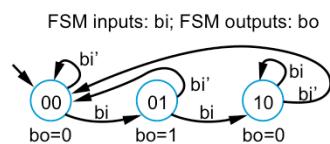
## 8.1. Projeto de FSM (continuação)

- **Exemplo 1:** Projetar um sincronizador de aperto de botão (círcuito que converte um toque no botão em um pulso com duração de um ciclo de *clock*, independentemente do tempo que o botão fica apertado).

- **Passo 1:** *Codificar os estados*

**Codificação binária**

	s1	s0
A	0	0
B	0	1
C	1	0



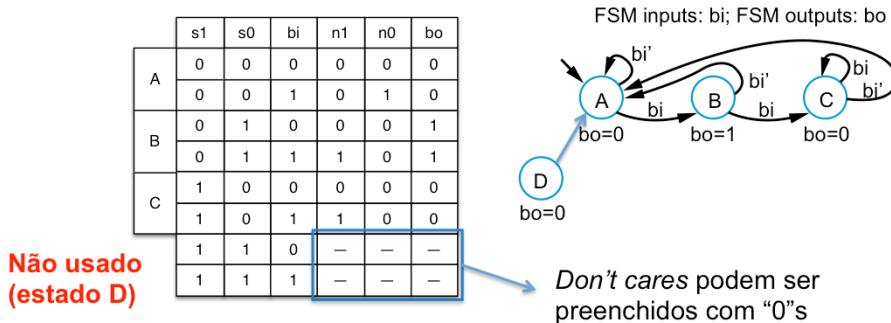
- **Passo 2:** *Criar uma tabela de estados*

Utilizando a codificação binária, precisamos de apenas dois bits para fazer isso.  
Próximo passo: criar uma tabela de estados.

## 8.1. Projeto de FSM (continuação)

- **Exemplo 1:** Projetar um sincronizador de aperto de botão (círcuito que converte um toque no botão em um pulso com duração de um ciclo de *clock*, independentemente do tempo que o botão fica apertado).

- **Passo 2:** Criar uma tabela de estados



Sendo  $s_1$  e  $s_0$  os bits em que fica codificado o estado atual e  $n_1$  e  $n_0$  os que codificam o próximo estado, obtemos esta tabela. Observe que sobra um código para os estados, de forma que, por enquanto, não importa o que é produzido pelo circuito neste caso.

## 8.1. Projeto de FSM (continuação)

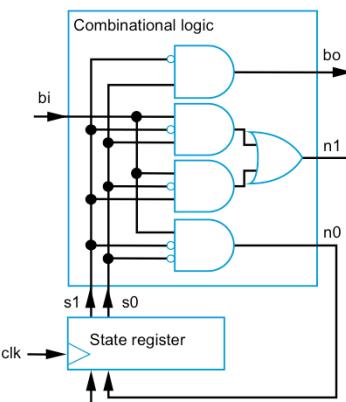
- **Exemplo 1:** Projetar um sincronizador de aperto de botão (circuito que converte um toque no botão em um pulso com duração de um ciclo de *clock*, independentemente do tempo que o botão fica apertado).
  - **Passo 3:** *Implementar...*

Próximo passo: implementação.

## 8.1. Projeto de FSM (continuação)

- **Exemplo 1:** Projetar um sincronizador de aperto de botão (circuito que converte um toque no botão em um pulso com duração de um ciclo de *clock*, independentemente do tempo que o botão fica apertado).
  - **Passo 3:** *Implementar...*

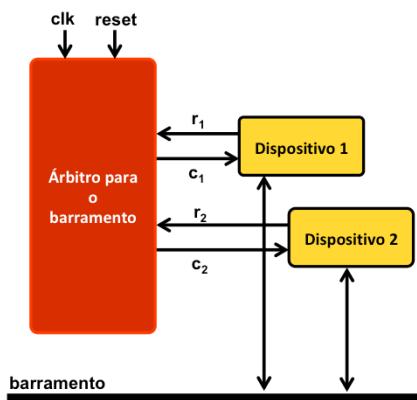
$$\begin{aligned}n_1 &= s_1's_0 b_i + s_1 s_0'b_i \\n_0 &= s_1's_0' b_i \\b_o &= s_1's_0 b_i' + s_1 s_0 b_i = s_1's_0\end{aligned}$$



Através da tabela de transição de estados que acabamos de produzir, obtemos as funções lógicas apresentadas neste slide. A forma final deste circuito está desenhada ao lado.

## 8.1. Projeto de FSM (continuação)

- **Exemplo 2:** Faça o projeto de um circuito árbitro para o acesso a um barramento que funcione conforme descrito a seguir.

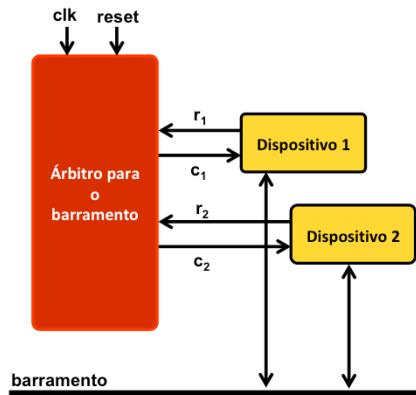


- O árbitro recebe requisições para o uso do barramento de 2 dispositivos diferentes
- O “Dispositivo 1” tem prioridade sobre o “Dispositivo 2” para o uso do barramento
- Somente um dispositivo por vez pode ser autorizado a usar o barramento
- Para sinalizar que o dispositivo pode usar o barramento, o árbitro faz o sinal de concessão igual a 1 ( $c_1 = 1$  ou  $c_2 = 1$ , dependendo do dispositivo)

Vejamos o próximo exemplo.

## 8.1. Projeto de FSM (continuação)

- **Exemplo 2:** Faça o projeto de um circuito árbitro para o acesso a um barramento que funcione conforme descrito a seguir.



- Uma vez que o dispositivo recebe a concessão para usar o barramento, ele permanece com esta concessão durante todo o tempo que ele necessitar usar o barramento (por isso, o dispositivo mantém seu sinal de requisição em nível alto durante o período de uso do barramento)
- Faça o projeto do árbitro...

## 8.1. Projeto de FSM (continuação)

- **Exemplo 2:** Faça o projeto de um circuito árbitro para o acesso a um barramento que funcione conforme descrito a seguir.
  - **Solução:**
    - Em um determinado momento, o circuito poderá então estar em um dentre os seguintes estados:
      - **Nenhum** dispositivo com acesso concedido ao barramento
      - Concessão de acesso para o **Dispositivo 1**
      - Concessão de acesso para o **Dispositivo 2**
    - Como são 3 os estados possíveis, precisamos de 2 variáveis de estado para codificar tais estados e, portanto, usaremos 2 flip-flops.
    - Com isso, podemos especificar as variáveis:
      - Entradas:  $r_1$  e  $r_2$
      - Saídas:  $c_1$  e  $c_2$
      - Variáveis de estados:  $s_1$  e  $s_0$  para estado atual e  $n_1$  e  $n_0$  para próximo estado.

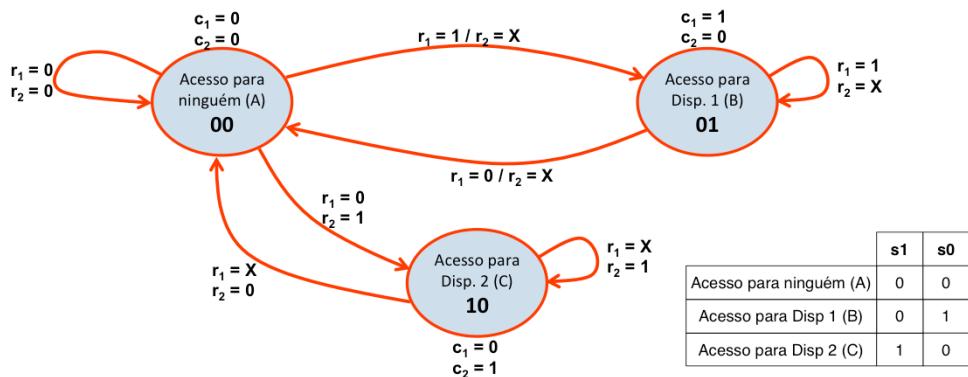
Para construir o diagrama de estados, consideremos os seguintes pontos.

## 8.1. Projeto de FSM (continuação)

- Exemplo 2:** Faça o projeto de um circuito árbitro para o acesso a um barramento que funcione conforme descrito a seguir.

- Solução:**

- Construindo o diagrama de estados, temos:



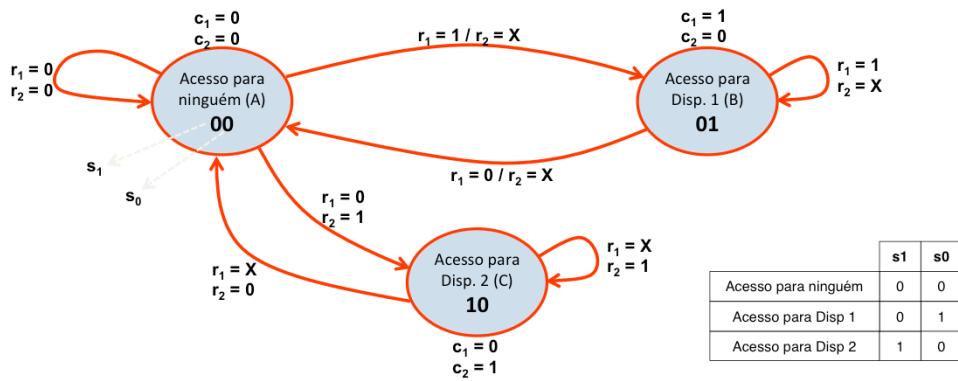
O diagrama de estados obtido a partir disso é o seguinte.

## 8.1. Projeto de FSM (continuação)

- Exemplo 2:** Faça o projeto de um circuito árbitro para o acesso a um barramento que funcione conforme descrito a seguir.

- Solução:**

- Construindo o diagrama de estados, temos:



Veja que a codificação dos estados já está embutida neste diagrama.

## 8.1. Projeto de FSM (continuação)

- **Exemplo 2:** Faça o projeto de um circuito árbitro para o acesso a um barramento que funcione conforme descrito a seguir.
  - **Solução:**
    - Agora, a tabela:

Não usado  
(estado D)

	Entradas				Saídas			
	s1	s0	r1	r2	n1	n0	c1	c2
A	0	0	0	0	0	0	0	0
	0	0	0	1	1	0	0	0
	0	0	1	—	0	1	0	0
B	0	1	0	—	0	0	1	0
	0	1	1	—	0	1	1	0
C	1	0	—	0	0	0	0	1
	1	0	—	1	1	0	0	1
	1	1	—	—	—	—	—	—

Don't cares podem ser usados para maior simplificação

A partir disso, obtemos esta tabela de transição dos estados. Aqui, novamente, temos um código não utilizado e, como antes, os "don't cares" podem ser usados para maior simplificação da equações lógicas.

## 8.1. Projeto de FSM (continuação)

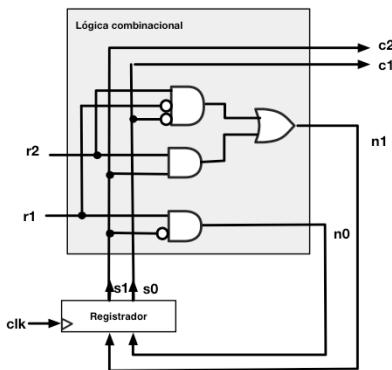
- **Exemplo 2:** Faça o projeto de um circuito árbitro para o acesso a um barramento que funcione conforme descrito a seguir.

- **Solução:**

- E as equações:

- $n_1 = s_1 r_2 + \bar{r}_1 r_2 \bar{s}_0$
- $n_0 = r_1 \bar{s}_1$
- $c_1 = s_0$
- $c_2 = s_1$

- Agora basta montar o circuito...



Neste slide temos tais equações e também o circuito já montado.

## PROBLEMAS

**Problema 8.1.** Projete o circuito gerador da sequência (0001, 0011, 1100, 1000, repete...).

**Solução:**

EA	q1	q0
A	0	0
B	0	1
C	1	0
D	1	1

Entradas				Saidas				
q1	q0	Q1	Q0	w	x	y	z	
0	0	0	1	0	0	0	1	
0	1	1	0	0	0	1	1	
1	0	1	1	1	1	0	0	
1	1	0	0	1	0	0	0	

Estado atual (EA)= $q_1 q_0$

Próximo Estado (PE)= $Q_1 Q_0$

Fazendo os mapas de Karnaugh  
obtenho:

$$Q_1 = q_1 q_0' + q_0' q_1$$

$$Q_0 = q_0'$$

$$w = q_1$$

$$x = q_1 q_0'$$

$$y = q_1' q_0$$

$$z = q_1'$$

## 8. Projeto de Circuitos Sequenciais

---

8.1. Projeto de FSM (continuação)

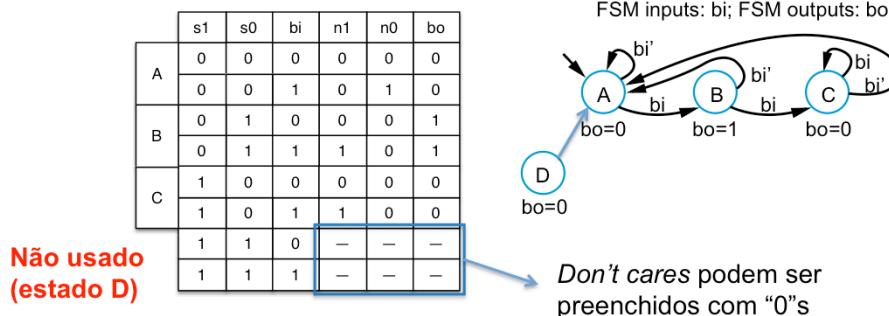
**8.2. FSM com estados de bloqueio (Lock-out)**

8.3. Engenharia reversa de projeto FSM

Agora veremos sobre os estados de bloqueio que podem haver em uma FSM.

## 8.2. FSM com estados de bloqueio (Lock-out)

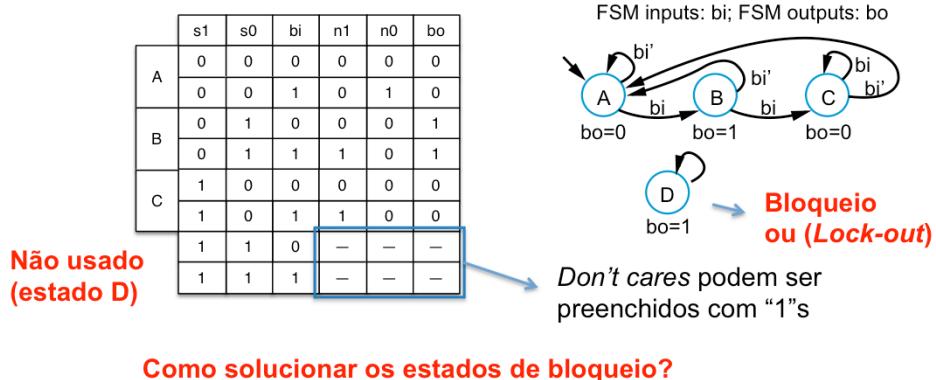
- **No exemplo 1:** Projetar um sincronizador de aperto de botão (circuito que converte um toque no botão em um pulso com duração de um ciclo de *clock*, independentemente do tempo que o botão fica apertado).



Vimos no exemplo 1 que havia um código não utilizado para os estados. Foi orientado que os "don't cares" fossem preenchidos com 0's.

## 8.2. FSM com estados de bloqueio (Lock-out)

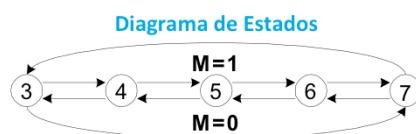
- **No exemplo 1:** Projetar um sincronizador de aperto de botão (circuito que converte um toque no botão em um pulso com duração de um ciclo de *clock*, independentemente do tempo que o botão fica apertado).



Mas caso os "don't cares" fossem preenchidos com 1's teríamos um estado de bloqueio, ou "lock -out". Assim, se, por alguma falha, o circuito cair no estado D, o mesmo permanece sempre neste mesmo estado. Como solucionar este problema?

## 8.2. FSM com estados de bloqueio (Lock-out)

- Exemplo 3:** Projecto de um Contador Ascendente/Descendente Síncrono de Módulo 5 (PADM5)



**Codificação de Estados**

	s2	s1	s0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

**Tabela de Transição de Estados**

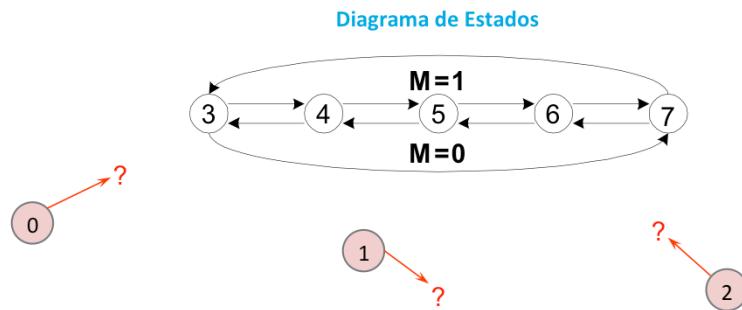
	Entradas			Saídas			
	s2	s1	s0	M	n2	n1	n0
3	0	1	1	0	1	1	1
	0	1	1	1	1	0	0
4	1	0	0	0	0	1	1
	1	0	0	1	1	0	1
5	1	0	1	0	1	0	0
	1	0	1	1	1	1	0
6	1	1	0	0	1	0	1
	1	1	0	1	1	1	1
7	1	1	1	0	1	1	0
	1	1	1	1	0	1	1
	0	0	0	—	—	—	—
	0	0	1	—	—	—	—
	0	1	0	—	—	—	—

Não usados  
(estado 0, 1 e 2)

Para explicar isso, vejamos outro exemplo: um contador ascendente/descendente síncrono de módulo 5. Neste exemplo, como podemos ver no slide, não são usados os estados 0, 1 e 2.

## 8.2. FSM com estados de bloqueio (Lock-out)

- **Exemplo 3:** O que acontece se a máquina transitar para um estado fora da gama prevista (lock-out)?

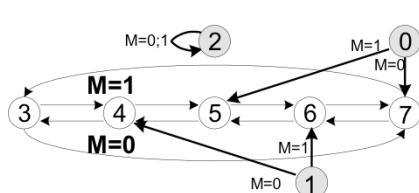


O que acontece se a máquina transitar para um estado fora da gama prevista (ou seja, entrar em um estado de lock-out)?

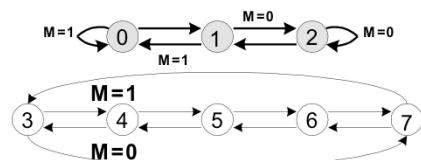
## 8.2. FSM com estados de bloqueio (Lock-out)

- “Estados de LOCK-OUT: no caso de não serem utilizados todos os estados disponíveis, pode ocorrer a situação do contador se encontrar num estado não desejado (fora da sequência de contagem) devido a ruído no circuito ou à não imposição de estado inicial.
- Nessa situação o contador entra na sequência de contagem pretendida ou fica indefinidamente no exterior (Lock-Out).

Exemplo de Lock-Out (1)



Exemplo de Lock-Out (2)

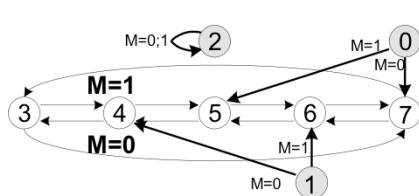


Uma vez que o circuito está em um desses estados pode tanto voltar para a sequência pretendida como permanecer indefinidamente em lock-out, como podemos ver nos exemplos deste slide. Para entrar em lock-out basta um ruído no circuito ou a não imposição de estado inicial.

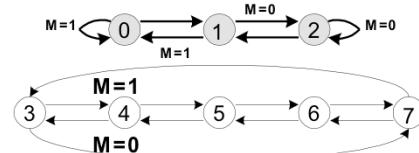
## 8.2. FSM com estados de bloqueio (Lock-out)

- **Solução 1:** impor a transição de qualquer estado externo para um estado da sequência de contagem
- **Solução 2:** considerar uma entrada extra, de inicialização, que coloque o sistema num dos estados de contagem pretendido.

Exemplo de Lock-Out (1)



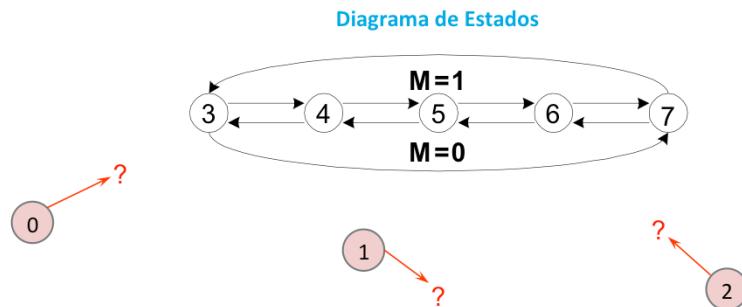
Exemplo de Lock-Out (2)



Há duas soluções para este problema. A primeira consiste em impor a transição de qualquer estado externo para um estado da sequência de contagem. Na segunda, consideramos uma entrada extra, de inicialização (reset), que coloque o sistema num dos estados de contagem pretendido.

## 8.2. FSM com estados de bloqueio (Lock-out)

- **Exemplo 3:** O que acontece se a máquina transitar para um estado fora da gama prevista (lock-out)?



Voltando ao nosso exemplo, vamos utilizar a primeira solução.

## 8.2. FSM com estados de bloqueio (Lock-out)

- **Exemplo 3:** O que acontece se a máquina transitar para um estado fora da gama prevista (lock-out)?

$$\begin{aligned}
 n_2 &= (\overline{M} + \overline{s}_2 + \overline{s}_1 + \overline{s}_0) \cdot (M + s_1 + s_0) \\
 n_1 &= M \overline{s}_1 \overline{s}_0 + M \overline{s}_1 s_0 + \overline{M} s_1 s_0 + M s_2 s_1 \\
 n_0 &= M s_2 s_1 + \overline{s}_0 + \overline{M} \overline{s}_2
 \end{aligned}$$

Estados fora da gama prevista definidos para evitar lock-outs

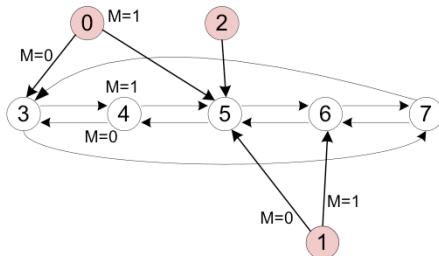
Tabela de Transição de Estados

	s2	s1	s0	M	n2	n1	n0
3	0	1	1	0	1	1	1
	0	1	1	1	1	0	0
4	1	0	0	0	0	1	1
	1	0	0	1	1	0	1
5	1	0	1	0	1	0	0
	1	0	1	1	1	1	0
6	1	1	0	0	1	0	1
	1	1	0	1	1	1	1
7	1	1	1	0	1	1	0
	1	1	1	1	0	1	1
0	0	0	0	0	0	1	1
	0	0	0	1	1	0	1
	0	0	1	0	1	0	1
	0	0	1	1	1	1	0
2	0	1	0	—	1	0	1

Assim, a tabela de transição de estados com os estados fora da gama prevista já definidos é a do slide. Aqui também estão definidas as equações correspondentes às saídas.

## 8.2. FSM com estados de bloqueio (Lock-out)

- **Exemplo 3:** O que acontece se a máquina transitar para um estado fora da gama prevista (lock-out)?

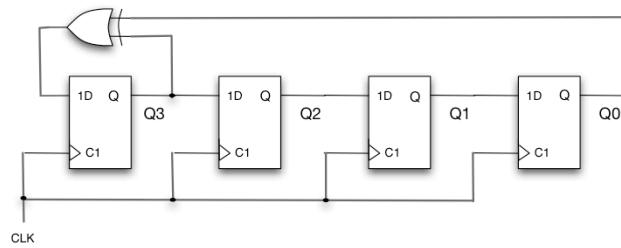


Os estados, não considerados para efeito da contagem (estados 0, 1 e 2), permitem passar a máquina para a sequência de contagem pretendida ao fim de um ciclo de relógio.

Com isso definido, os estados, não considerados para efeito da contagem, permitem passar a máquina para a sequência de contagem pretendida ao fim de um ciclo de relógio.

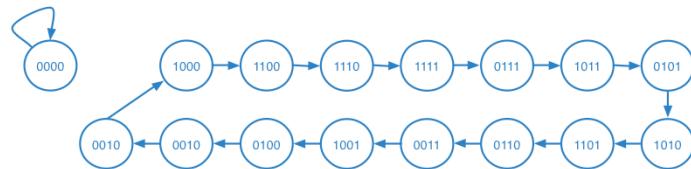
## PROBLEMAS

**Problema 8.2.** Considere o circuito da figura e obtenha a sequencia que gera partindo do estado inicial  $Q_3Q_2Q_1Q_0=1000$ . Utilizando portas lógicas modifique o circuito de forma de evitar situações de bloqueio (*lock-outs*).

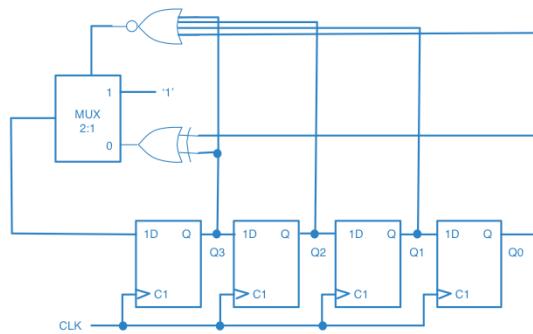


## PROBLEMAS

Solução:



Existe *Lock-Out* no estado 0000, para解决-lo existem varias formas, por exemplo podemos colocar um Set assincrono nos *Flip-flops D*. Outra solução possível é forçar a que o estado 0000 continue para o estado 1000 sem alterar o funcionamento do sistema. Para isso, podemos usar lógica combinatória ou Multiplicadores como mostrado no figura a seguir.



## **8. Projeto de Circuitos Sequenciais**

---

**8.1. Projeto de FSM (continuação)**

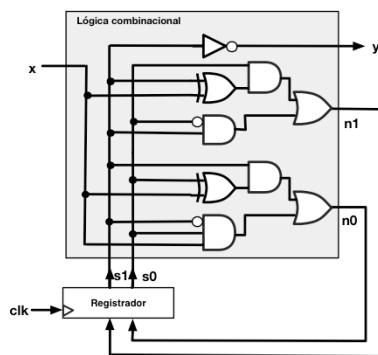
**8.2. FSM com estados de bloqueio (Lock-out)**

**8.3. Engenharia reversa de projeto FSM**

Veremos agora como fazer a engenharia reversa de um projeto FSM.

### 8.3. Engenharia reversa de projeto FSM

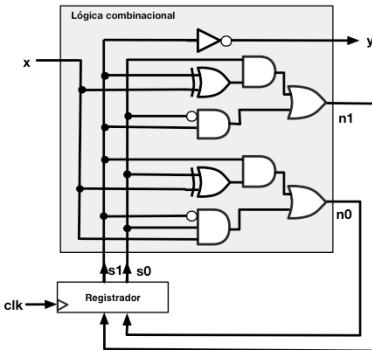
- Engenharia reversa de projeto FSM consiste em, a partir do circuito obter o diagrama de estados.
- **Exemplo 4:** Projete o diagrama de estados associado ao circuito sequencial.



A engenharia reversa de um projeto FSM consiste em, a partir do circuito obter o diagrama de estados. Vejamos como fazer isso através deste novo exemplo.

### 8.3. Engenharia reversa de projeto FSM

- **Exemplo 4:** Projete o diagrama de estados associado ao circuito sequencial.



$$n_0 = \overline{s_1} \overline{s_0} x + (x \oplus s_1) s_0$$

$$n_1 = \overline{s_1} s_0 + (x \oplus s_0) \overline{s_1}$$

$$y = s_1$$

1. Obter funções lógicas

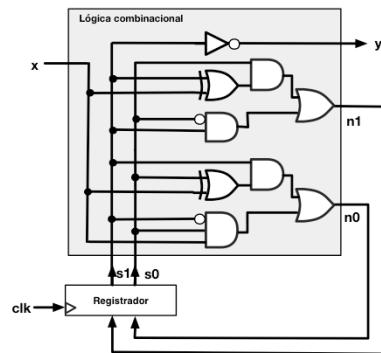
De modo contrário ao projeto FSM, primeiro, obtemos as funções lógicas implementadas no circuito.

### 8.3. Engenharia reversa de projeto FSM

- **Exemplo 4:** Projete o diagrama de estados associado ao circuito sequencial.

**2. Obter  
Tabela de  
transição de  
estados**

s1	s0	x	n1	n0	y
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	1	1	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	0	0	0



$$n_0 = \overline{s_1} \overline{s_0} x + (x \oplus s_1) s_0$$

$$n_1 = \overline{s_1} s_0 + (x \oplus s_0) s_1$$

$$y = s_1$$

**Obter funções lógicas**

A partir disso conseguimos obter a tabela de transição de estados.

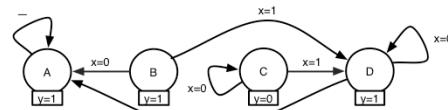
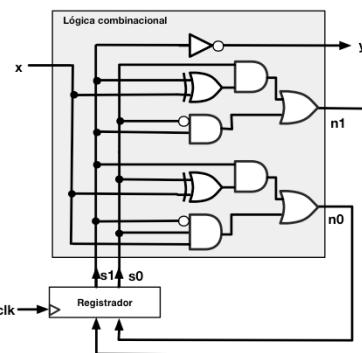
### 8.3. Engenharia reversa de projeto FSM

- Exemplo 4: Projete o diagrama de estados associado ao circuito sequencial.

**2. Obter Tabela de transição de estados**

ESTADOS		A	C		
s1	s0	x	n1	n0	y
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	1	1	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	0	0	0

**3. Decodificar os estados**



**4. Obter diagrama de estados**

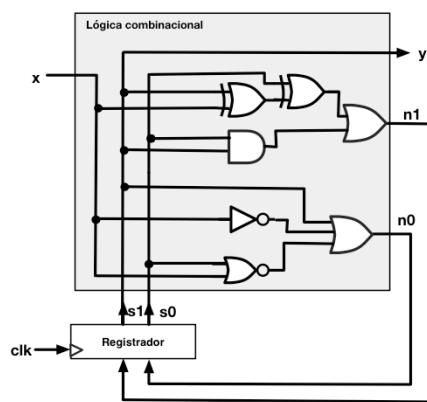
**1. Obter funções lógicas**

$$\begin{aligned}n_0 &= \overline{s_1} \overline{s_0} x + (x \oplus s_1) s_0 \\n_1 &= \overline{s_1} s_0 + (x \oplus s_0) s_1 \\y &= s_1\end{aligned}$$

Em seguida, decodificamos os estados, definindo nomes arbitrários para eles. A partir a tabela de transição de estados já conseguimos também obter o diagrama de estados.

## PROBLEMAS

**Problema 8.3.** Projete o diagrama de estados associado ao circuito sequencial.



## PROBLEMAS

### Solução:

A partir de circuito obtenho:

$$n_1 = s_1 s_0 + x \oplus s_1 \oplus s_0$$

$$\begin{aligned} n_0 &= s_1 + x' + (x + s_0)' = s_1 + x' + x's_0' = \\ &= s_1 + x'(1 + s_0') = s_1 + x' \end{aligned}$$

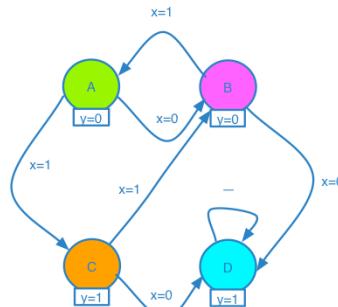
$$y = s_1$$

$$\text{Estado atual (EA)} = s_1 s_0$$

$$\text{Proximo Estado (PE)} = n_1 n_0$$

		Entradas		Saídas	
s <sub>1</sub>	s <sub>0</sub>	x	n <sub>1</sub>	n <sub>0</sub>	y
0	0	0	0	1	0
0	0	1	1	0	0
0	1	0	1	1	0
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	0	1	1
1	1	0	1	1	1
1	1	1	1	1	1

Codificação de estados		
EA	q <sub>1</sub>	q <sub>0</sub>
A	0	0
B	0	1
C	1	0
D	1	1





FEDERAL UNIVERSITY  
OF SANTA CATARINA

## EEL5105 – Circuitos e Técnicas Digitais

### Aula 8

---

Prof. Héctor Pettenghi

[hector@eel.ufsc.br](mailto:hector@eel.ufsc.br)

<http://hectorpettenghi.paginas.ufsc.br>

Material desenvolvido com apoio de arquivos de apresentação do livro de Frank Vahid

## Exercícios

---

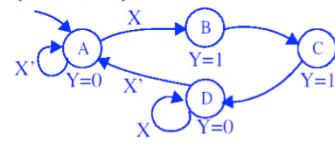
- Exercícios do Livro do **Frank Vahid**
  - **3.23** até **3.27**
  - **3.29** e **3.30**
  - **3.32** e **3.33**
  - **3.38** até **3.42**
- **A versão digital do livro do Frank Vahid está disponível no site da BU**
  - Mais especificamente em:  
[http://150.162.4.10/pergamon/biblioteca\\_s/php/login\\_pearson.php](http://150.162.4.10/pergamon/biblioteca_s/php/login_pearson.php)

## Exercícios

- Respostas Livro do Frank Vahid

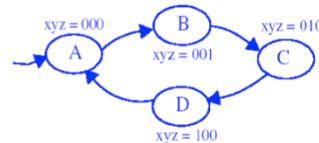
3.23

*Inputs: X, Outputs: Y*



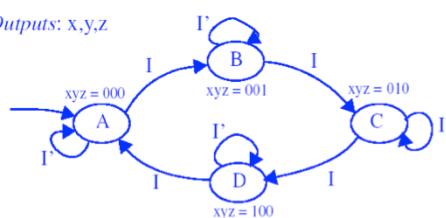
3.24

*Inputs: None, Outputs: x,y,z*



3.25

*Inputs: I, Outputs: x,y,z*

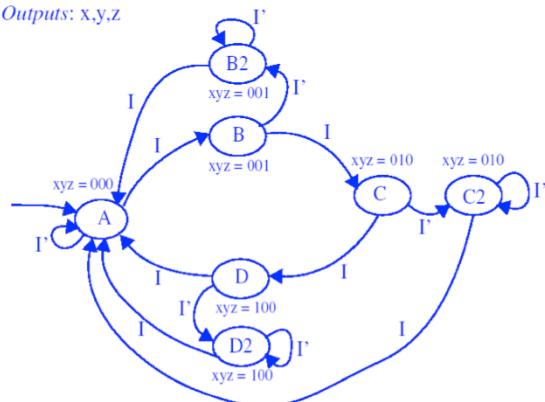


## Exercícios

- Respostas Livro do Frank Vahid

3.26

*Inputs: I, Outputs: x,y,z*

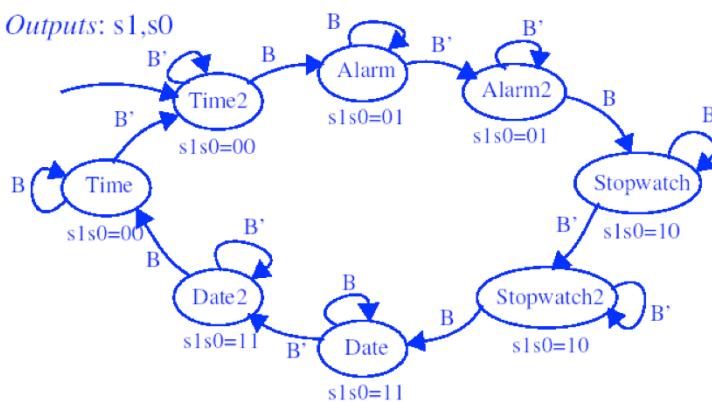


## Exercícios

- Respostas Livro do Frank Vahid

3.27

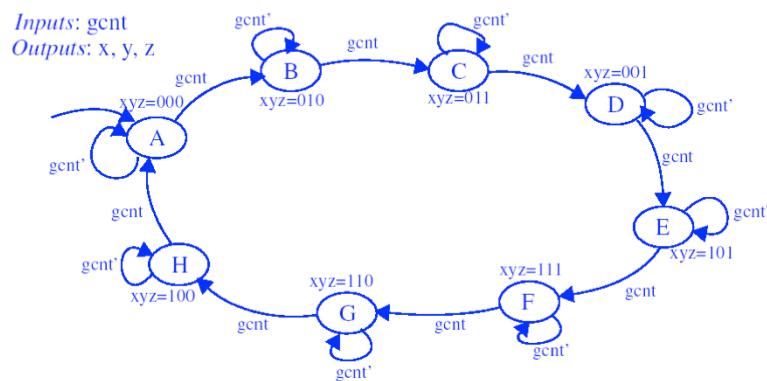
Inputs: B, Outputs: s1,s0



## Exercícios

- Respostas Livro do Frank Vahid

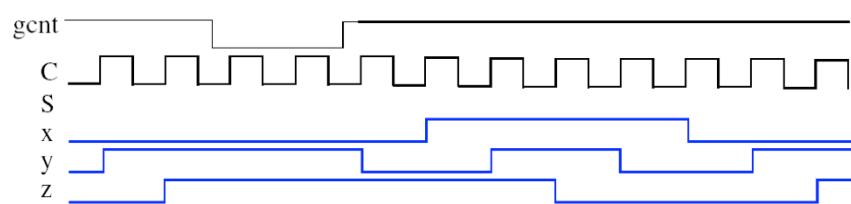
3.29



## Exercícios

- Respostas Livro do Frank Vahid

3.30



## Exercícios

---

- Respostas Livro do **Frank Vahid**

3.32 a) 2 bits

b) 3 bits

c) 4 bits

d) 5 bits

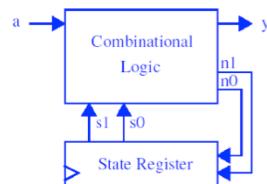
e) 10 bits

3.33

$2^{16} = 65,536$  possible states

## Exercícios

- Resposta do 3.38



A straightforward encoding is A=00, B=01, C=10, D=11.

Inputs			Outputs		
s1	s0	a	n1	n0	y
0	0	0	0	1	0
0	0	1	0	0	0
0	1	0	0	1	1
0	1	1	1	0	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	1	0	0	0

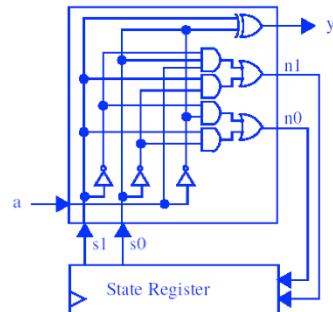
## Exercícios

- Resposta do 3.38

$$n1 = s1's0a + s1s0'a' + s1s0'a = s1's0a + s1s0'$$

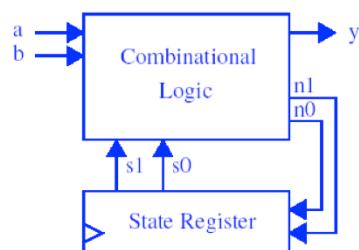
$$n0 = s1's0'a' + s1's0a' + s1s0'a' + s1s0'a = s1'a' + s1s0'$$

$$y = s1's0a' + s1's0a + s1s0'a' + s1s0'a = s1's0 + s1s0' = s1 \text{ xor } s0$$



## Exercícios

- Resposta do 3.39



A straightforward encoding is A=00, B=01, C=10, D=11.

Inputs				Outputs		
s1	s0	a	b	n1	n0	y
0	0	0	0	1	0	0
0	0	0	1	0	1	0
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	1
0	1	1	1	1	0	1
1	0	0	0	1	0	1
1	0	0	1	1	1	1
1	0	1	0	1	0	1
1	0	1	1	1	1	1
1	1	0	0	0	0	0
1	1	0	1	0	0	0
1	1	1	0	0	0	0
1	1	1	1	0	0	0

## Exercícios

- Resposta do 3.39

### Step 5 - Implement the combinational logic

$$n1 = s1's0'a'b' + s1's0a + s1s0'$$

$$n0 = s1's0'a'b + s1's0a' + s1s0'b$$

$$y = s1's0 + s1s0'$$

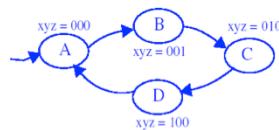
*Note: The above equations can be minimized further.*

## Exercícios

- Resposta do 3.40

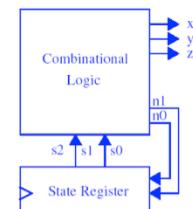
Step 1 - Capture the FSM

*Inputs: None, Outputs: x,y,z*



The FSM was created during Exercise 3.24.

Step 2 - Create the architecture



Step 3 - Encode the states

A straightforward encoding is A=00, B=01, C=10, D=11.

Step 4 - Create the state table

Inputs		Outputs				
s1	s0	n1	n0	x	y	z
0	0	0	1	0	0	0
0	1	1	0	0	0	1
1	0	1	1	0	1	0
1	1	0	0	1	0	0

Step 5 - Implement the combinational logic

$$n1 = s1's0 + s1s0' = s1 \text{ XOR } s0$$

$$n0 = s1's0' + s1s0' = s0'$$

$$x = s1s0$$

$$y = s1s0'$$

$$z = s1's0$$