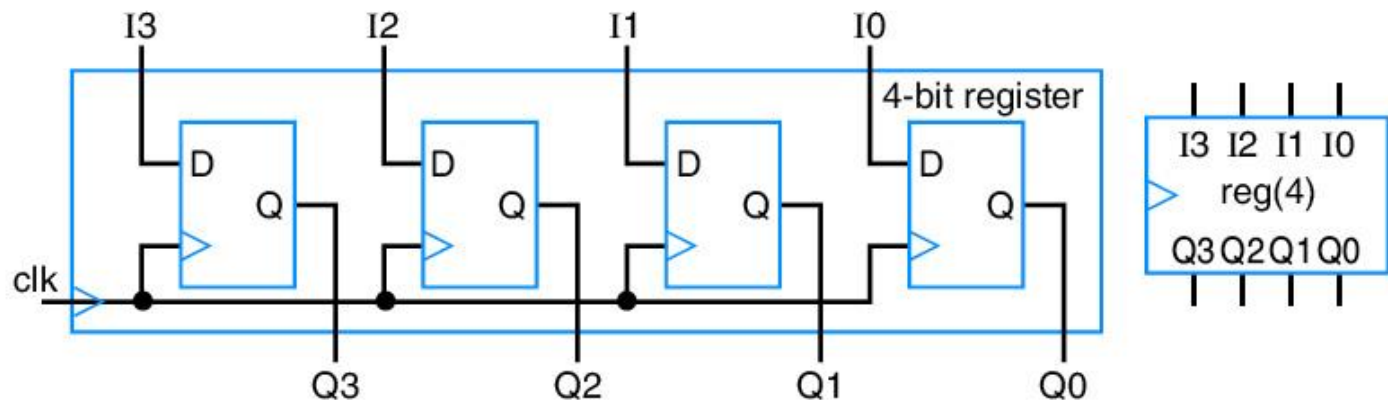


# Capítulo 3–Circuitos Sequenciais

Registradores e Máquinas de Estados  
Profa. Eliete Caldeira

# Registadores

- ▶ Componente sequencial que armazena múltiplos bits
- ▶ Construção usando múltiplos flip-flops (um para cada bit a ser armazenado) que compartilham o mesmo sinal de clock



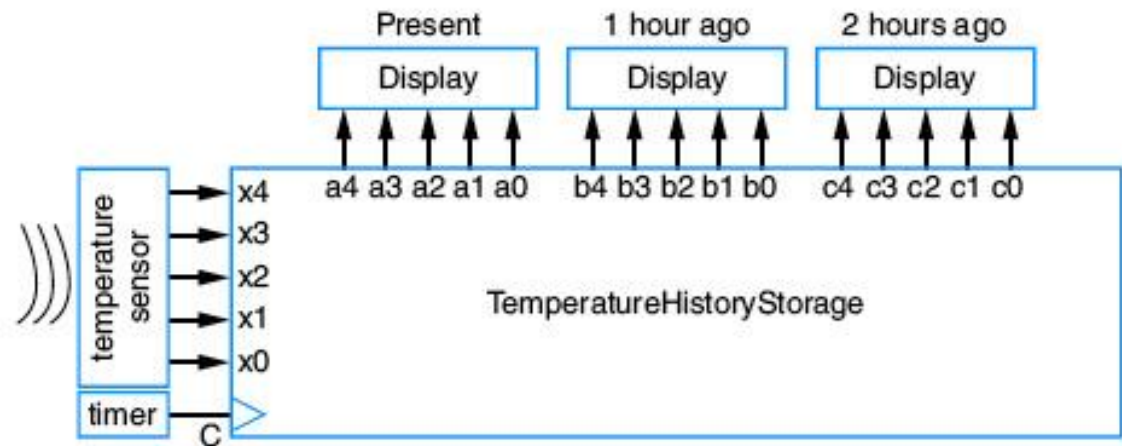
**Figure 3.30** A basic 4-bit register internal design (left) and block symbol (right).

# Registradores

- ▶ Exemplo: Projete um sistema que grava o valor da temperatura externa a cada hora e exiba as últimas três temperaturas armazenadas

# Registadores

- ▶ Exemplo: Projete um sistema que grava o valor da temperatura externa a cada hora e exiba as últimas três temperaturas armazenadas

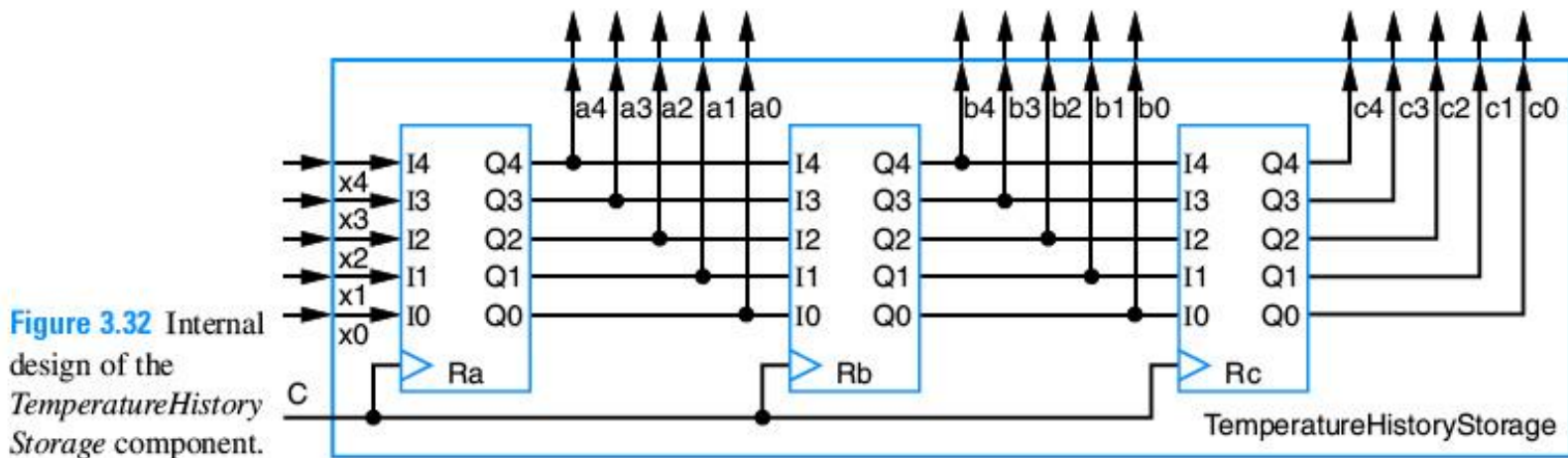


**Figure 3.31** Temperature history display system.

*(In practice, we would actually avoid connecting the timer output C to a clock input, instead only connecting an oscillator output to a clock input.)*

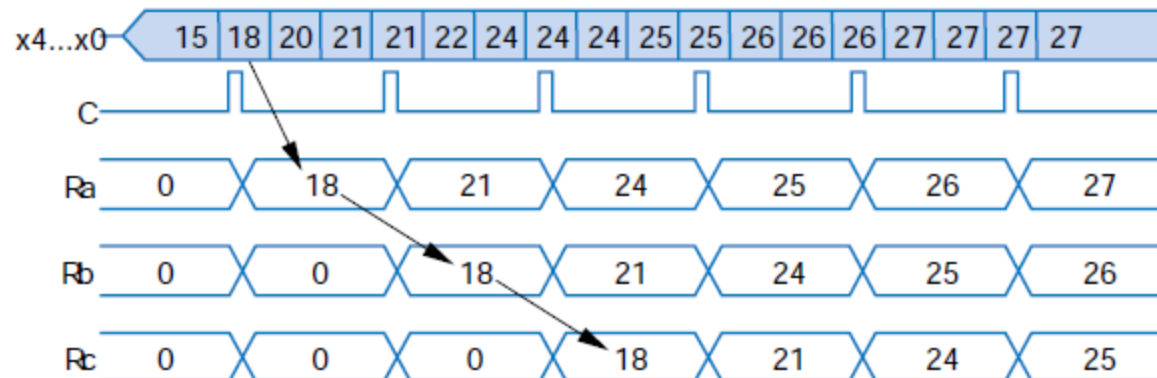
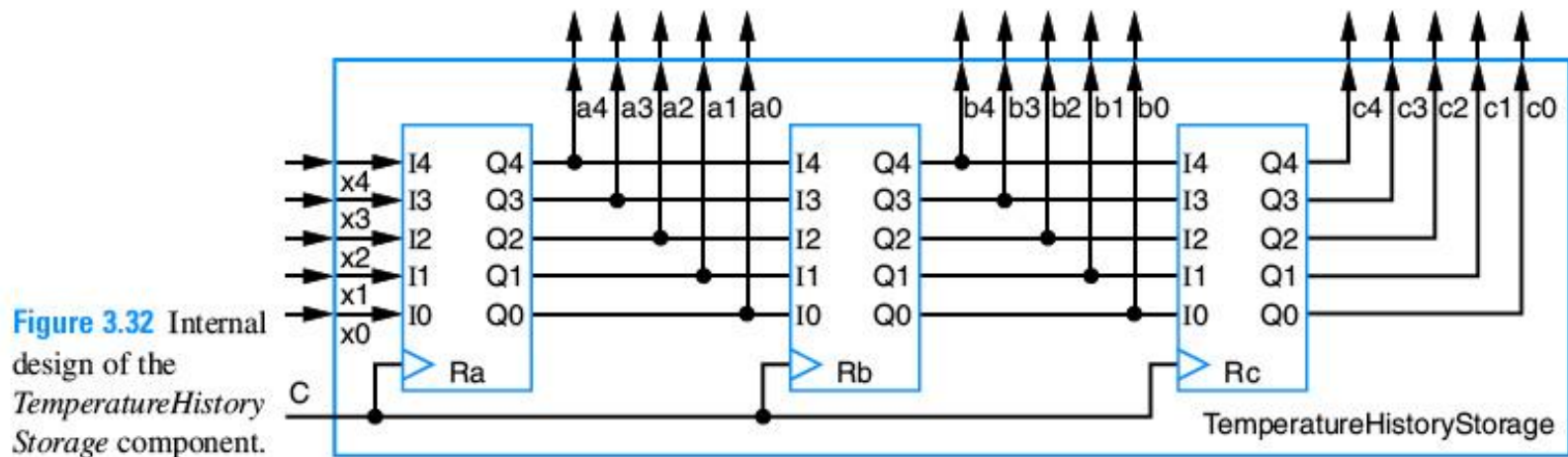
# Registadores

- ▶ Exemplo: Projete um sistema que grava o valor da temperatura externa a cada hora e exiba as últimas três temperaturas armazenadas



# Registadores

## ► Comportamento no tempo



# Circuitos sequenciais

- ▶ Em um circuito digital, os registradores armazenam bits.
- ▶ Bits armazenados significam que o circuito tem memória, o que é também conhecido com estado, resultando em circuitos sequenciais
- ▶ Podemos usar estes estados para projetar circuitos que tenham determinados comportamentos ao longo do tempo
- ▶ Por exemplo:
  - Acender uma luz durante 3 ciclos do clock após um botão ser pressionado
  - Piscar luzes em um padrão específico
  - Detectar que botões foram pressionados em uma sequência particular



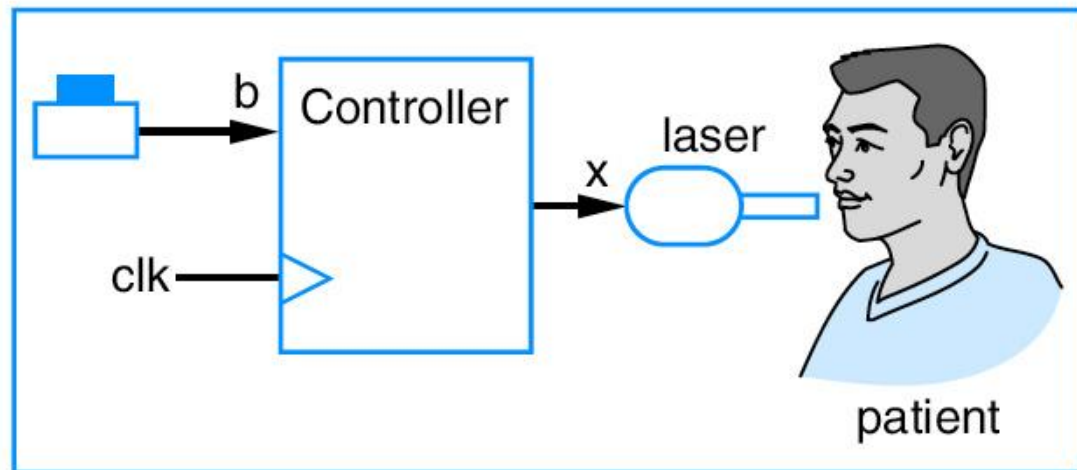
# Circuitos sequenciais

- ▶ Bloco de controle: circuito sequencial que controla saídas booleanas com base em entradas booleanas e em um comportamento específico, ordenado no tempo



# Circuitos sequenciais

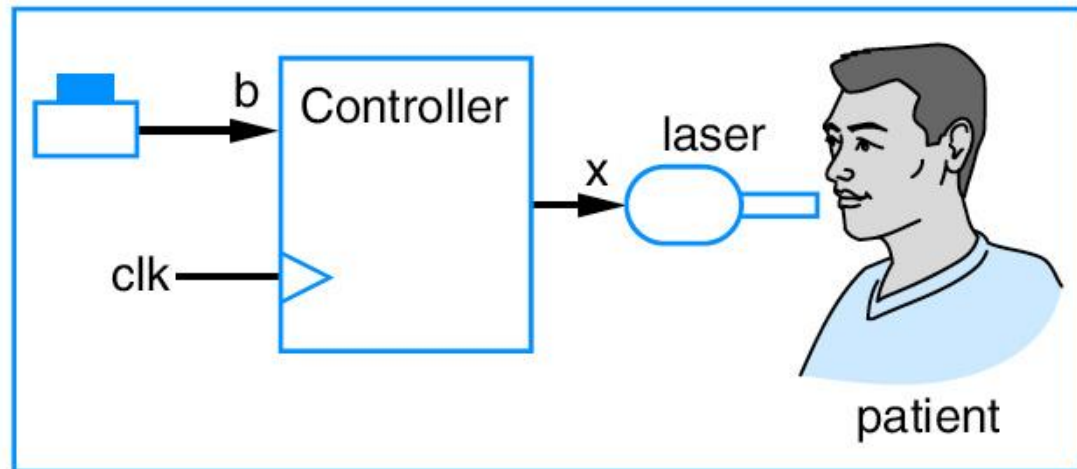
- ▶ Exemplo: Projete um sistema que aciona um LASER por 30 ns, correspondentes a 3 ciclos do clock de 100MHz, após o botão b ser pressionado.



**Figure 3.34** Laser timer system.

# Circuitos sequenciais

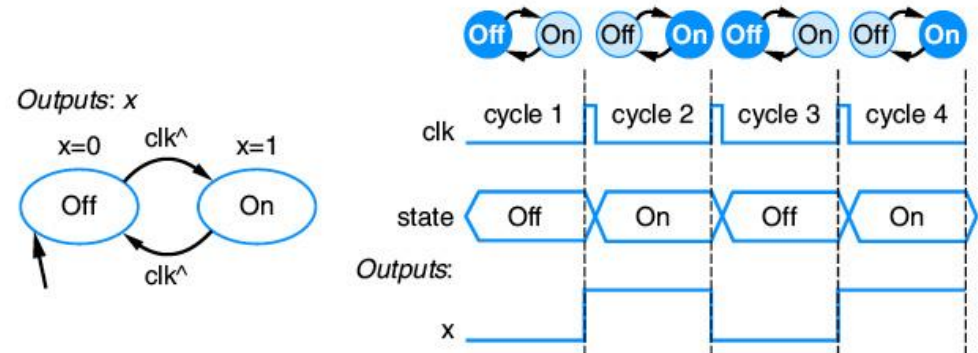
- ▶ Para resolver este problema precisamos de um formalismo que se pode obter com uma máquina de estados finitos



**Figure 3.34** Laser timer system.

# Máquinas de Estados Finitos

- ▶ Uma máquina de estados finitos (Finite State Machine – FSM) consiste em um conjunto de estados que representa todos os modos possíveis de um sistema
  - O sistema abaixo tem dois estados: ligado ( $x=0$ ) e desligado ( $x=1$ ) e muda de estado a cada ciclo de clock



**Figure 3.36** A simple state diagram (left) and the timing diagram describing the state diagram's behavior (right). Above the timing diagram, we see the FSM going from one state to the other in each clock cycle. " $clk^\wedge$ " represents the rising edge of the clock signal.

# Máquinas de Estados Finitos

- No exemplo do LASER  
Podemos pensar em um  
Sistema que fica em 1 três  
ciclos e volta a 0

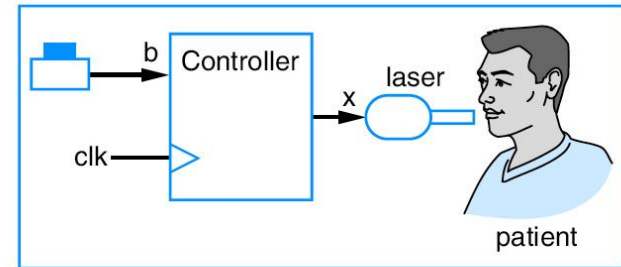


Figure 3.34 Laser timer system.

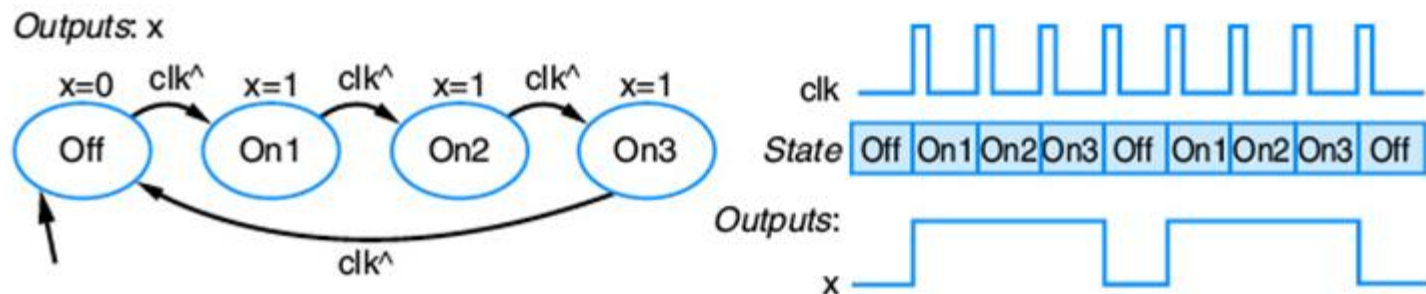


Figure 3.37 Three-cycles-high system: state diagram (left), timing diagram (right).

Mas o sistema não considera o botão!

# Máquinas de Estados Finitos

## ► Considerando o botão

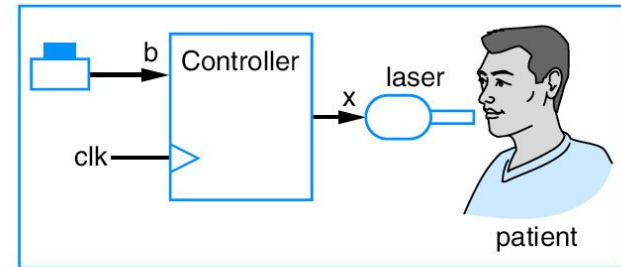


Figure 3.34 Laser timer system.

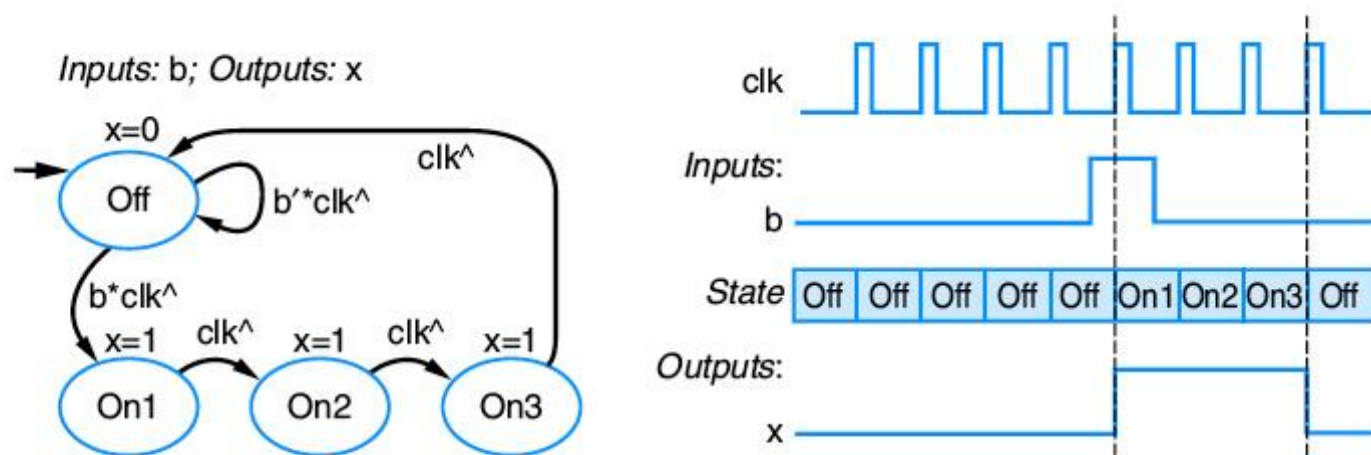


Figure 3.38 Three-cycles-high system: state diagram (left), timing diagram (right).

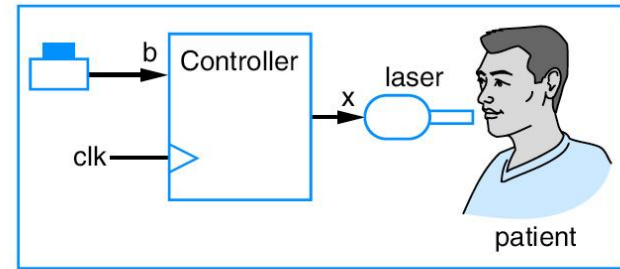
# Máquinas de Estados Finitos

- ▶ Assim, a máquina de estados consiste em:
  - Um conjunto de estados  $on1$ ,  $on1$ ,  $on3$  e  $off$
  - Um conjunto de entradas e saídas:  $b$  e  $x$
  - Um estado inicial (indicado por uma seta que não tem estado de origem):  $off$
  - Arestas orientadas que indicam para qual estado devemos seguir, com base no estado atual e nos valores de entrada. Estas arestas são conhecidas como transições.
  - Os valores de saída que devem ser gerados em cada estado e para cada condição das entradas.

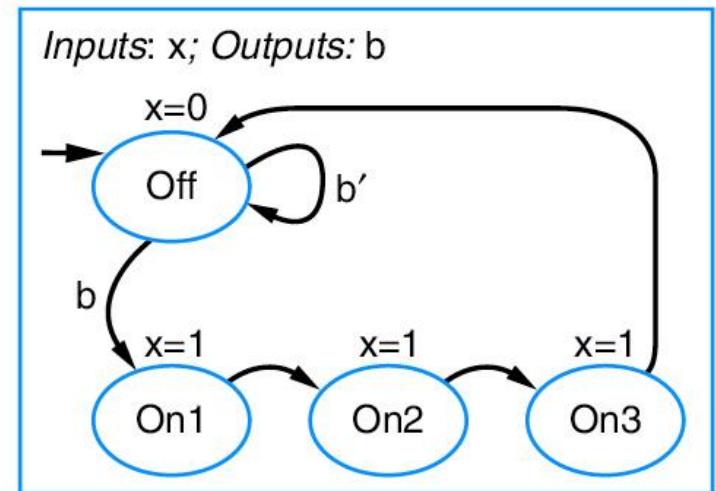


# Máquinas de Estados Finitos

- Simplificando a notação, podemos omitir a transição do clock porque ela sempre será respeitada



**Figure 3.34** Laser timer system.



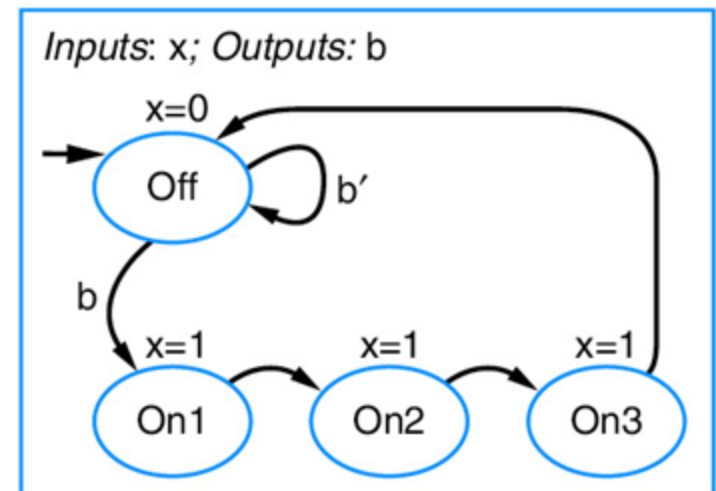
**Figure 3.39** Laser timer state diagram assuming every transition is ANDed with a rising clock.



# Máquinas de Estados Finitos

- ▶ Podemos representar máquinas de estados de várias maneiras:
  - Linguagem textual
  - Tabela de estados (esquerda)
  - Diagrama de estados (direita)

Estado Atual Entrada	Off	On1	On2	On3
<b>b=0</b>	Off	On2	On3	Off
<b>b=1</b>	On1	On2	On3	Off
<b>Saída</b>	x=0	x=1	x=1	x=1
<b>Próximo Estado</b>				



# Máquinas de Estados Finitos

- ▶ Exemplo: Projete um circuito que tem uma entrada  $a$  que deverá ser 1 quando o computador do carro estiver solicitando o ID da chave. Quando  $a = 1$  a chave deve enviar seu ID = 1011 em forma serial, começando com o bit mais à direita (bit menos significativo ou LSB – least significant bit), por meio da saída  $r$ .



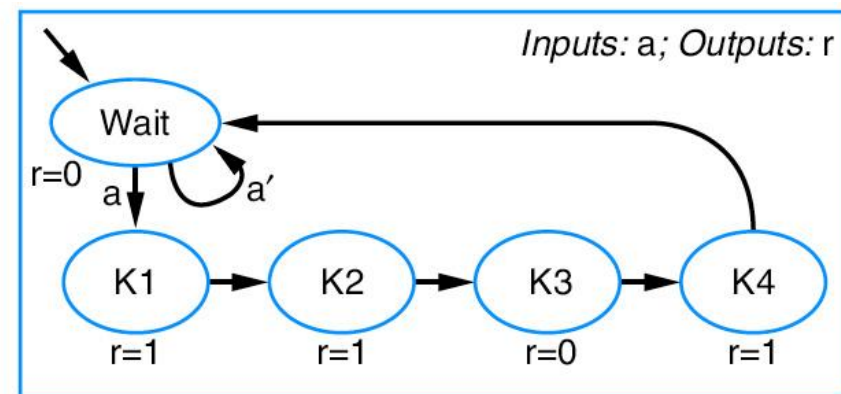
**Figure 3.40** Why are the heads of keys getting thicker? Note that the key on the right is thicker than the key on the left. The key on the right has a computer chip inside that sends an identifier to the car's computer, thus helping to reduce car thefts.

# Máquinas de Estados Finitos

- ▶ Exemplo: Projete um circuito que tem uma entrada  $a$  que deverá ser 1 quando o computador do carro estiver solicitando o ID da chave. Quando  $a = 1$  a chave deve enviar seu ID = 1011 em forma serial, começando com o bit mais à direita (bit menos significativo ou LSB – least significant bit), por meio da saída  $r$ .



**Figure 3.40** Why are the heads of keys getting thicker? Note that the key on the right is thicker than the key on the left. The key on the right has a computer chip inside that sends an identifier to the car's computer, thus helping to reduce car thefts.



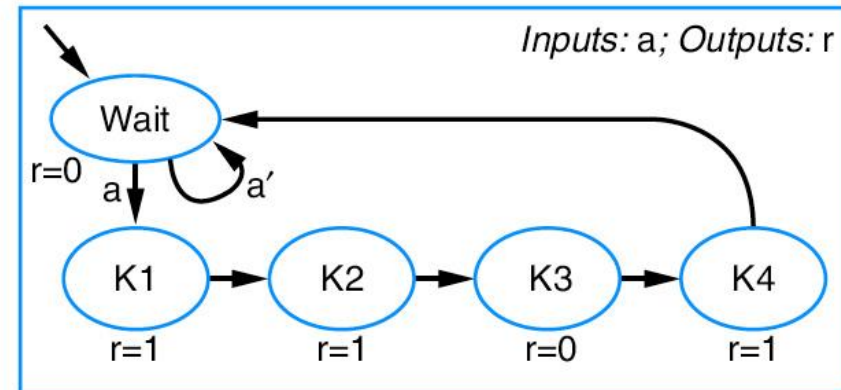
**Figure 3.41** Secure car key FSM.

# Máquinas de Estados Finitos

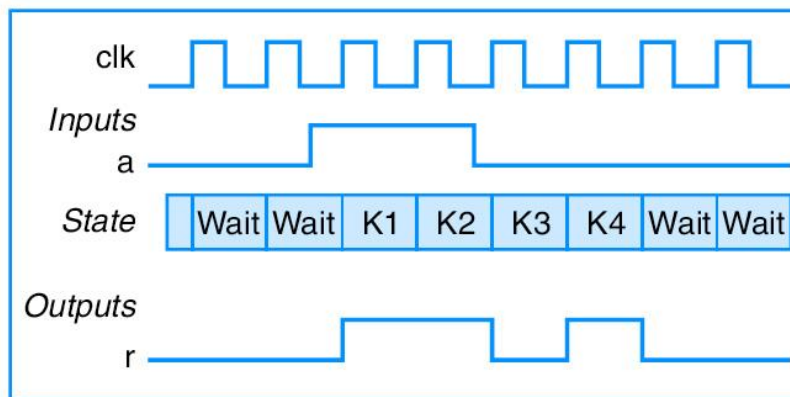
## ► Diagrama de tempo



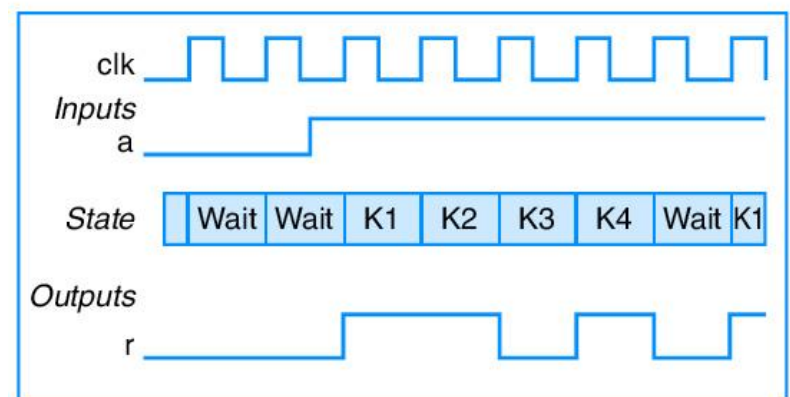
**Figure 3.40** Why are the heads of keys getting thicker? Note that the key on the right is thicker than the key on the left. The key on the right has a computer chip inside that sends an identifier to the car's computer, thus helping to reduce car thefts.



**Figure 3.41** Secure car key FSM.



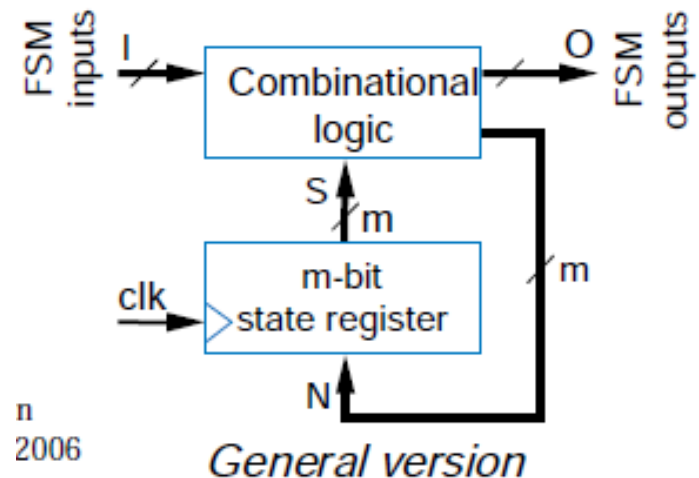
**Figure 3.42** Secure car key timing diagram.



**Figure 3.43** Secure car key timing diagram for a different sequence of values on input  $a$ .

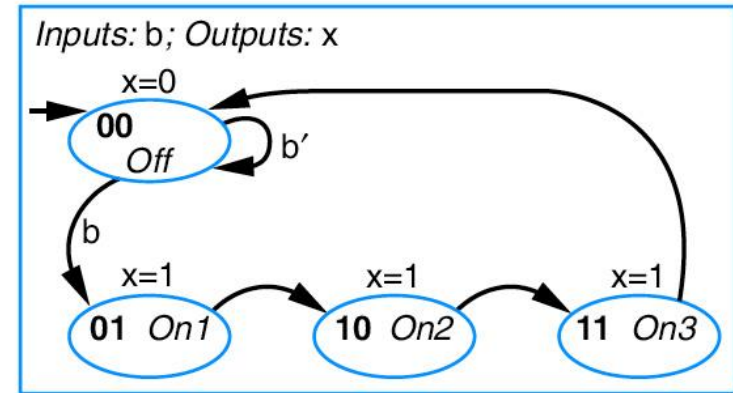
# Arquitetura padrão do bloco de controle

- ▶ Já sabemos descrever o comportamento sequencial usando uma Máquina de Estados
- ▶ Como converter?  
Máquina de Estados → Circuito Sequencial
- ▶ Método estruturado para converter uma FSM em circuito sequencial
- ▶ Usando:  
**Registrador de estado**  
+  
**lógica combinacional**

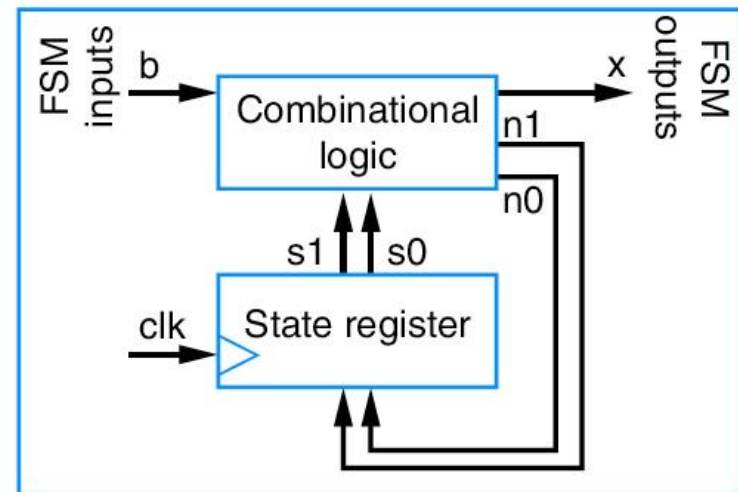


# Arquitetura padrão do bloco de controle

- ▶ Voltando ao projeto do circuito para a cirurgia com laser
  - Entrada:  $b$
  - Saída:  $x$
  - Estados:  $s1$  e  $s0$   
 $\{00, 01, 10, 11\}$
- ▶ É preciso escolher o tipo de flip-flop e projetar o circuito combinacional que faça a transição de estados e gere a saída



**Figure 3.49** Laser timer state diagram with encoded states.



**Figure 3.47** Standard controller architecture for the laser timer.



# Projeto do bloco de controle

## ▶ Passos:


1. Capture a FSM → Diagrama de bolhas ou de estados
2. Crie a arquitetura → Defina entradas, saídas e o número de flip-flops para representar estados
3. Codifique os estados → um código por estado
4. Crie a tabela de estados
5. Implemente a lógica combinacional



# Projeto do bloco de controle

- ▶ **Detalhamento dos passos:**
- ▶ **Passo 1. Capture a FSM:**
  - Crie uma FSM que descreva o comportamento desejado do circuito sequencial
- ▶ **Passo 2. Crie a arquitetura:**
  - Defina um registrador de estado com largura apropriada
  - Entradas do bloco de controle a ser projetado são os bits do registrador de estado e as entradas da FSM
  - Saídas são os bits de próximo estado e as saídas da FSM

# Projeto do bloco de controle

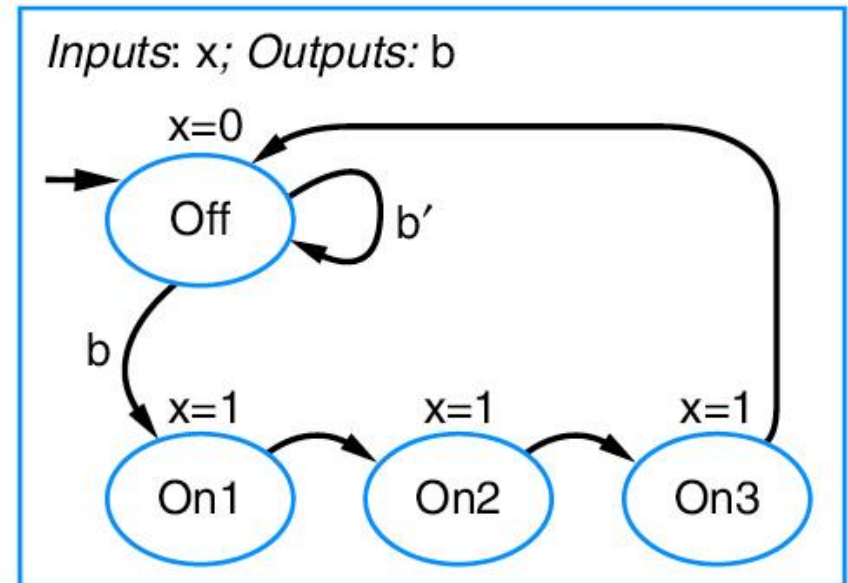
- ▶ Passo 3. Codifique os estados
    - Atribua um numero binário único a cada um dos estados.
  - ▶ Passo 4. Crie a tabela de estados
    - Crie uma tabela-verdade para a lógica combinacional
    - A lógica irá gerar as saídas e os sinais de próximo estado
    - Ordenando as entradas primeiro com os bits de estado faz com que a tabela-verdade descreva o comportamento dos estados
- 

# Projeto do bloco de controle

- ▶ Passo 5. Implemente a lógica combinacional

# Projeto do bloco de controle

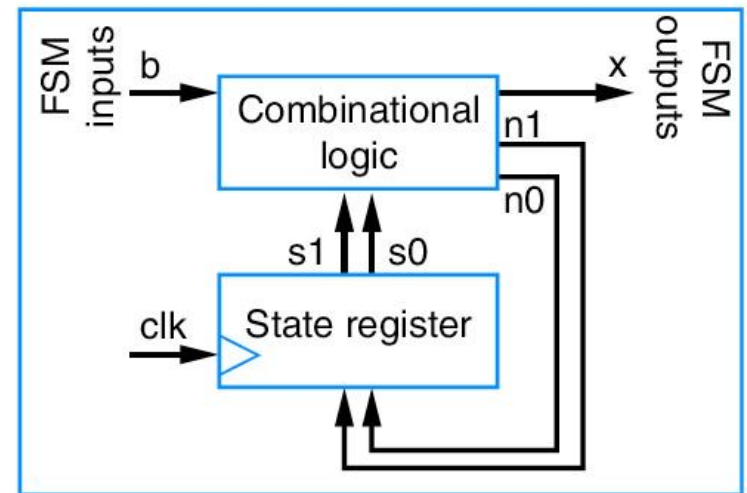
- ▶ Exemplo do LASER
- ▶ Passo 1. Capture a FSM



**Figure 3.39** Laser timer state diagram assuming every transition is ANDed with a rising clock.

# Projeto do bloco de controle

- ▶ Passo 2. Defina a arquitetura
  - 2 flip-flops (registrador de 2 bits)
  - Entrada da maquina: b
  - Entradas do bloco de controle: b e estado atual (s1 e s0)
  - Saída da máquina: x
  - Saídas do bloco de controle: x e próximo estado (n1 e n0)

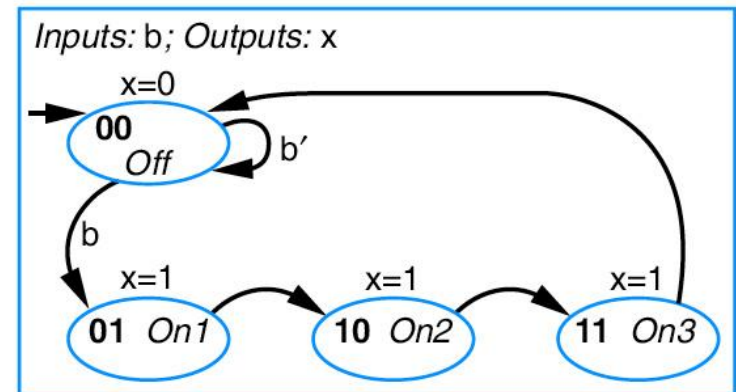


**Figure 3.47** Standard controller architecture for the laser timer.

# Projeto do bloco de controle

## ► Passo 3. Codifique os estados:

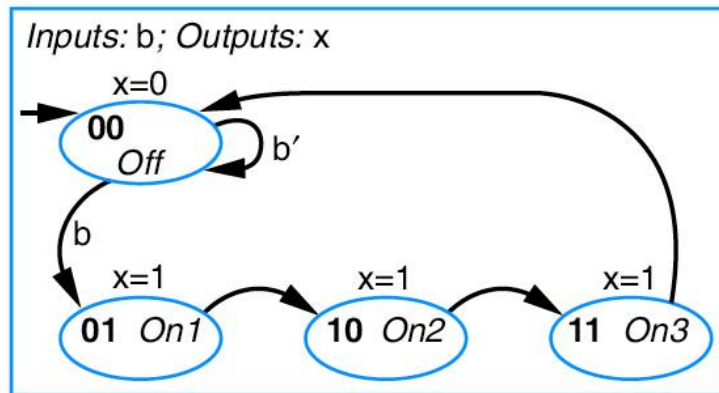
- Off: 00
- On1: 01
- On2: 10
- On3: 11



**Figure 3.49** Laser timer state diagram with encoded states.

# Projeto do bloco de controle

- ▶ Passo 4. Crie a tabela de estados



**Figure 3.49** Laser timer state diagram with encoded states.

**TABLE 3.3** State table for laser timer controller.

	Inputs			Outputs		
	s1	s0	b	x	n1	n0
<i>Off</i>	0	0	0	0	0	0
	0	0	1	0	0	1
<i>On1</i>	0	1	0	1	1	0
	0	1	1	1	1	0
<i>On2</i>	1	0	0	1	1	1
	1	0	1	1	1	1
<i>On3</i>	1	1	0	1	0	0
	1	1	1	1	0	0



# Projeto do bloco de controle

- ▶ Passo 5. Implemente a 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

$x = s1 + s0$  (note from the table that  $x=1$  if  $s1 = 1$  or  $s0 = 1$ )

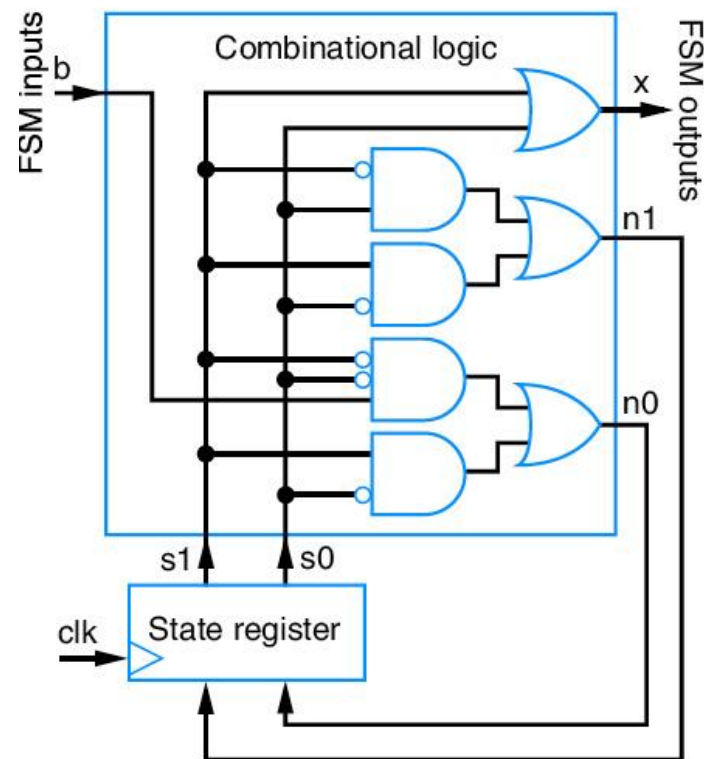
$n1 = s1's0b' + s1's0b + s1s0'b' + s1s0'b$   
 $n1 = s1's0 + s1s0'$

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

# Projeto do bloco de controle

- ▶ Passo 5. Implemente a lógica combinacional

$$\begin{aligned}x &= s1 + s0 \\n1 &= s1's0 + s1s0' \\n0 &= s1's0'b + s1s0'\end{aligned}$$



**Figure 3.50** Final implementation of the three-cycles-high laser timer controller.

# Projeto do bloco de controle

## ► Funcionamento do Projeto final

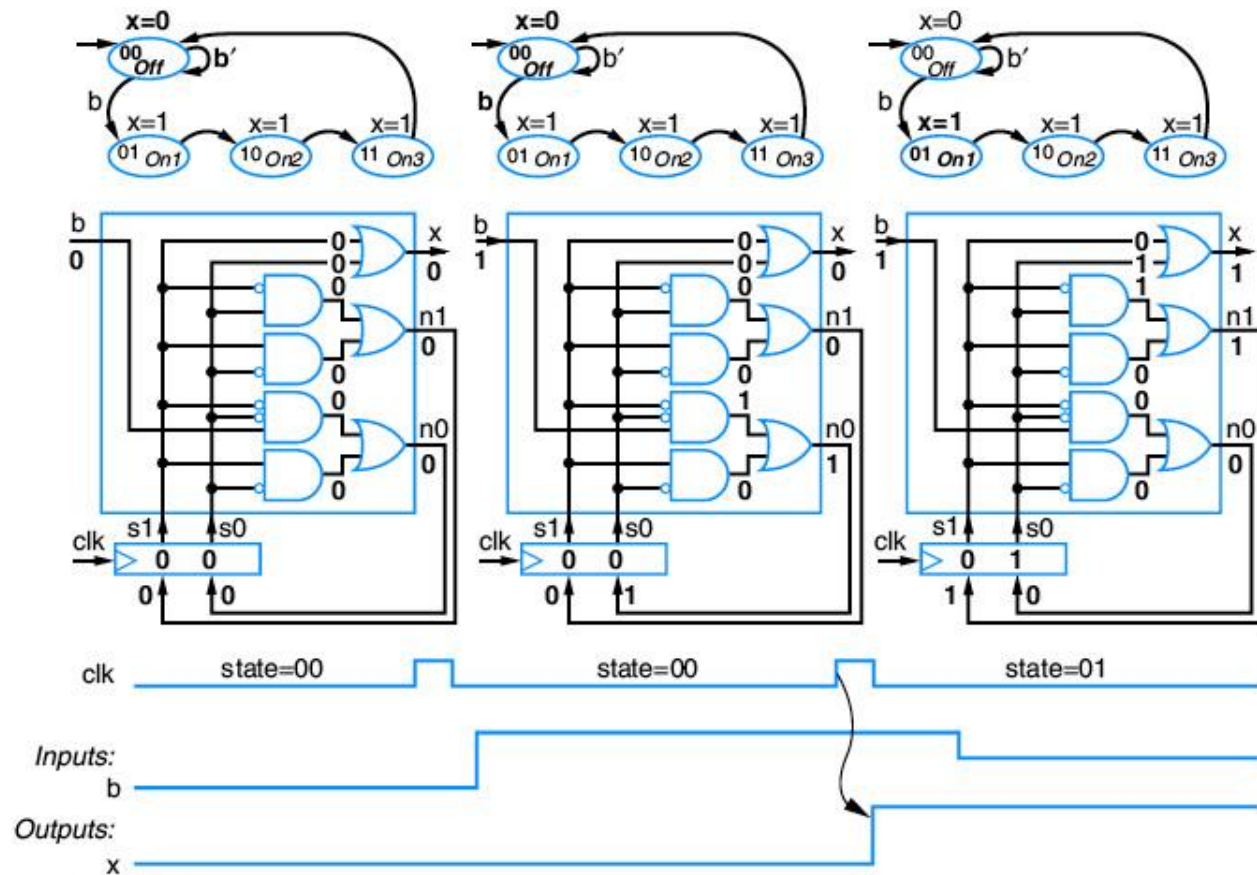
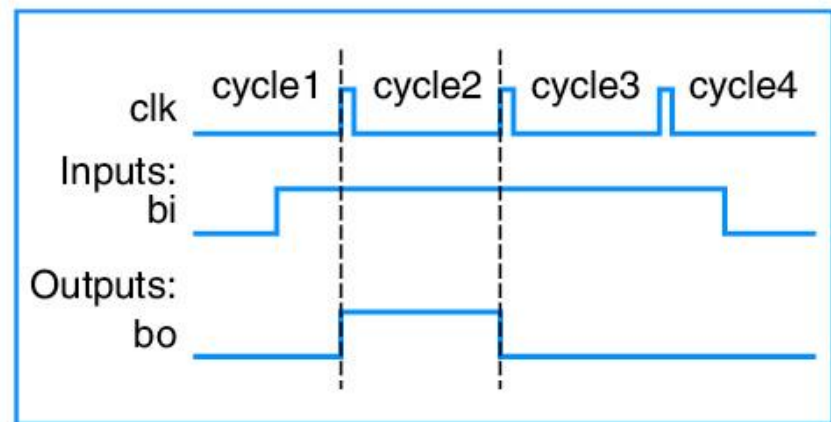
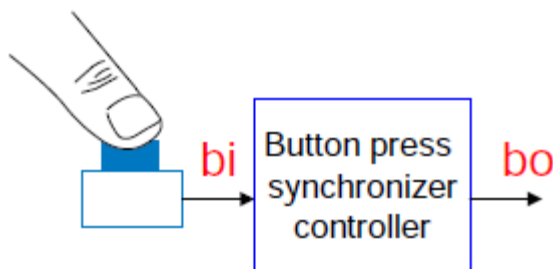


Figure 3.51 Tracing the behavior of the three-cycles-high laser timer controller.

# Exemplo: Sincronizador de aperto de botão

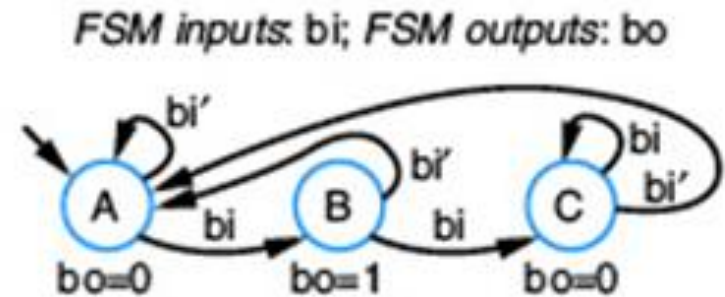
- ▶ Projetar circuito que sincroniza um aperto de botão com um sinal de clock, de tal modo que quando uma pessoa apertar o botão **bi**, o resultado será um sinal **bo** que fica em nível alto por exatamente um ciclo de clock



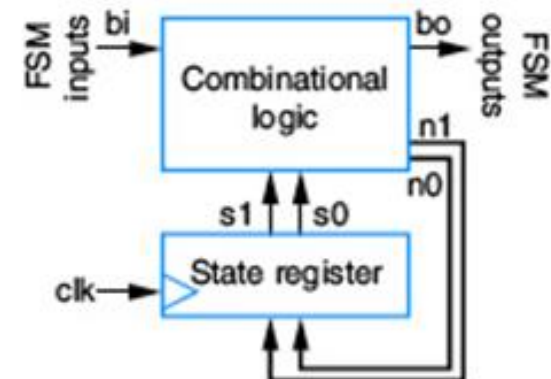
**Figure 3.52** Desired timing diagram of the button press synchronizer.

# Exemplo: Sincronizador de aperto de botão

- ▶ Passo 1. Capture a FSM

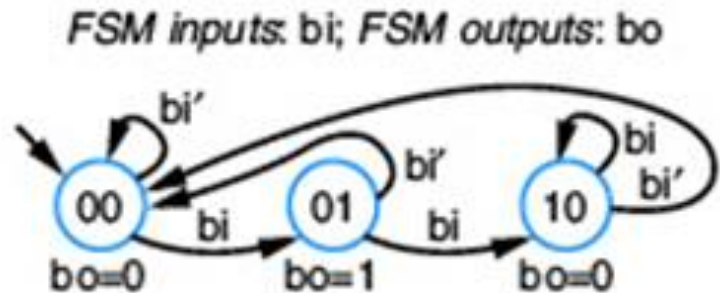


- ▶ Passo 2. Defina a arquitetura



# Exemplo: Sincronizador de aperto de botão

- ▶ Passo 3. Codifique os estados



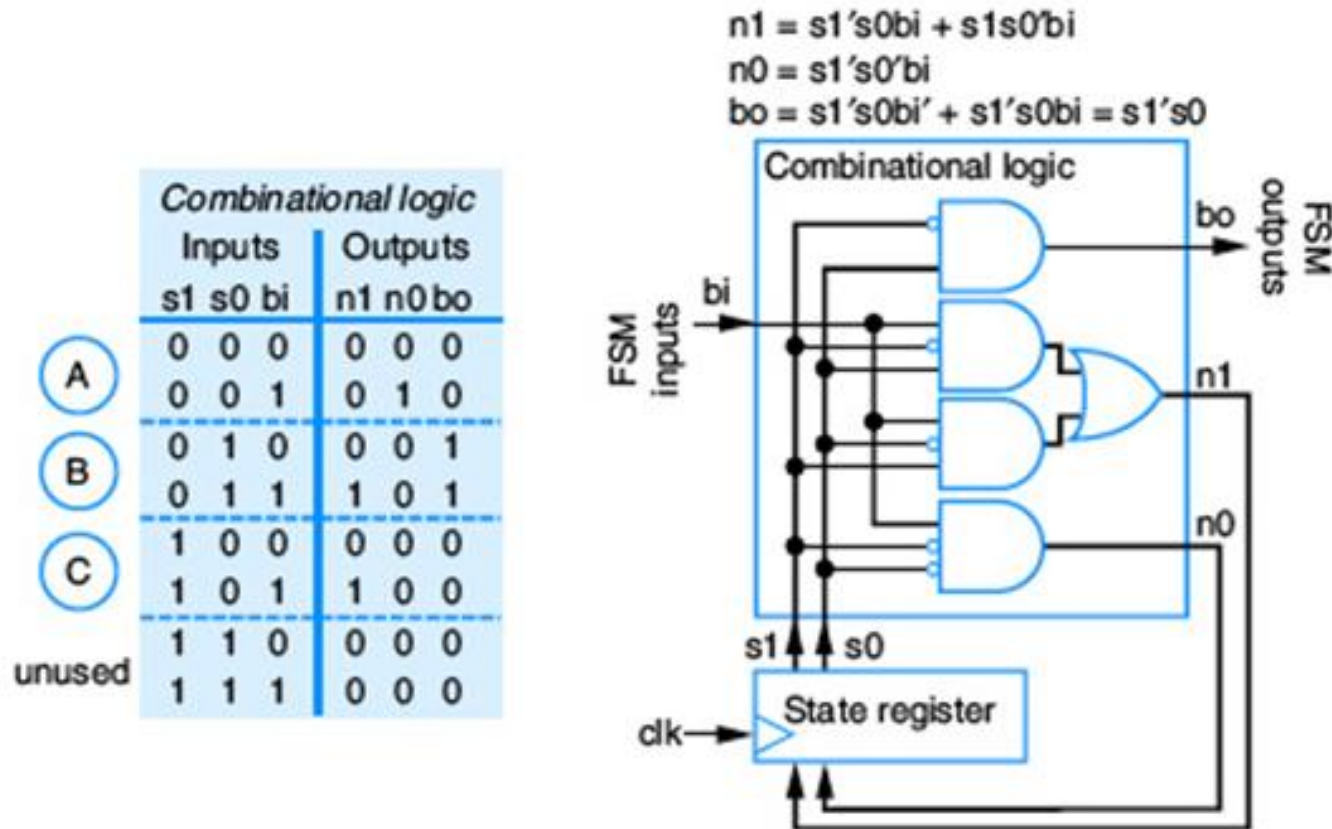
- ▶ Passo 4. Monte a tabela-verdade

Combinational logic						
Inputs			Outputs			
	s1	s0	bi	n1	n0	bo
A	0	0	0	0	0	0
	0	0	1	0	1	0
B	0	1	0	0	0	1
	0	1	1	1	0	1
C	1	0	0	0	0	0
	1	0	1	1	0	0
unused	1	1	0	0	0	0
	1	1	1	0	0	0



# Exemplo: Sincronizador de aperto de botão

## ▶ Passo 5. Projeto do circuito combinacional





# Exemplo: Gerador de sequência

- ▶ Circuito com quatro saídas: w, x, y e z
- ▶ Deve gerar a seguinte sequência:
  - 0001, 0011, 1100 e 1000
  - Depois de 1000, repetir a sequência
- ▶ Utilidade: controlar motor de passo, por exemplo

# Exemplo: Gerador de sequência

- ▶ Passo 1. Capture a FSM

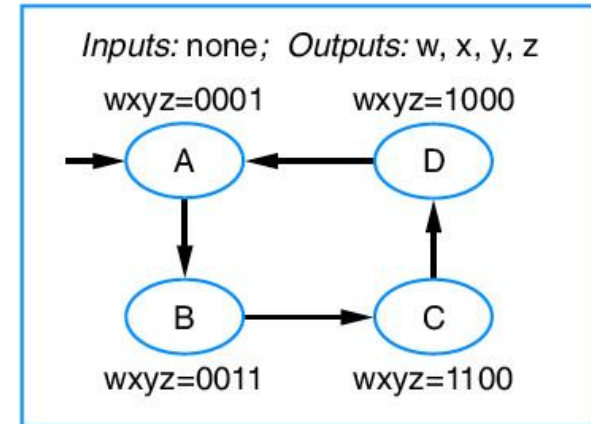


Figure 3.54 Sequence generator FSM.

- ▶ Passo 2. Defina a arquitetura

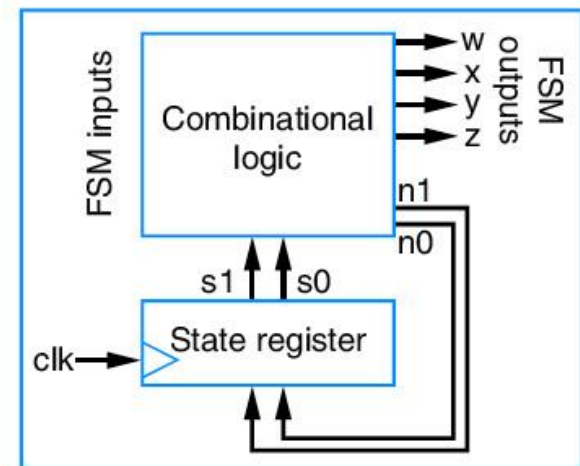
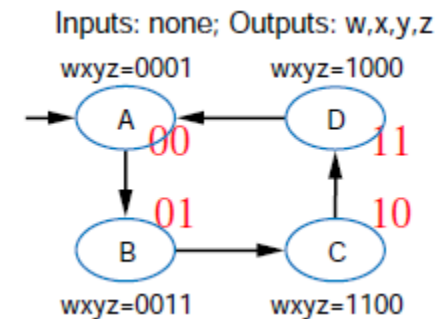


Figure 3.55 Sequence generator controller architecture.

# Exemplo: Gerador de sequência

- Passo 3. Codifique os estados



- Passo 4. Monte a tabela-verdade

**TABLE 3.4** State table for sequence generator controller.

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

# Exemplo: Gerador de sequência

## ► Passo 5. Projeto do circuito combinacional

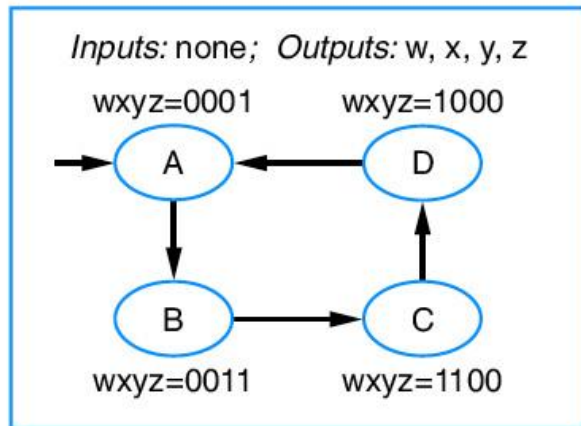


Figure 3.54 Sequence generator FSM.

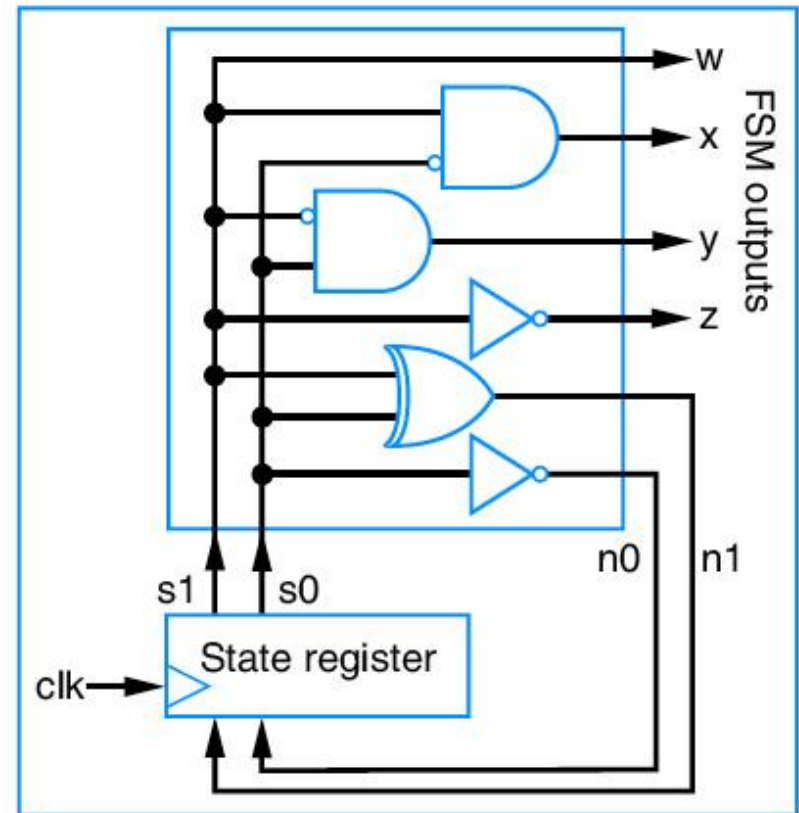


Figure 3.56 Sequence generator controller architecture.

Para ser continuado...