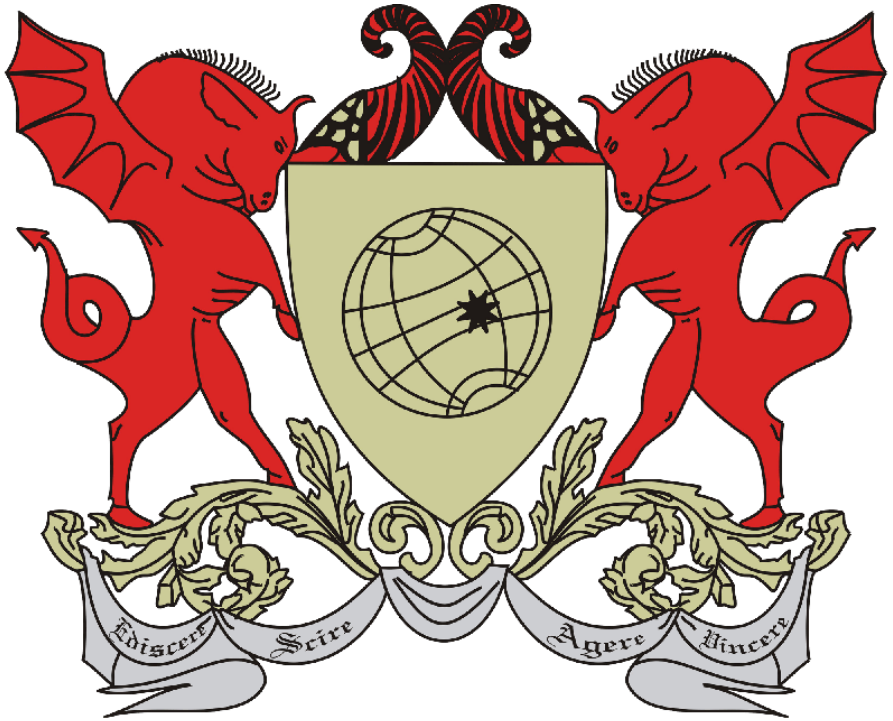


CCF 251 – Introdução aos Sistemas Lógicos

Aula 08 – Máquinas de estado finito

Prof. José Augusto Nacif – jnacif@ufv.br

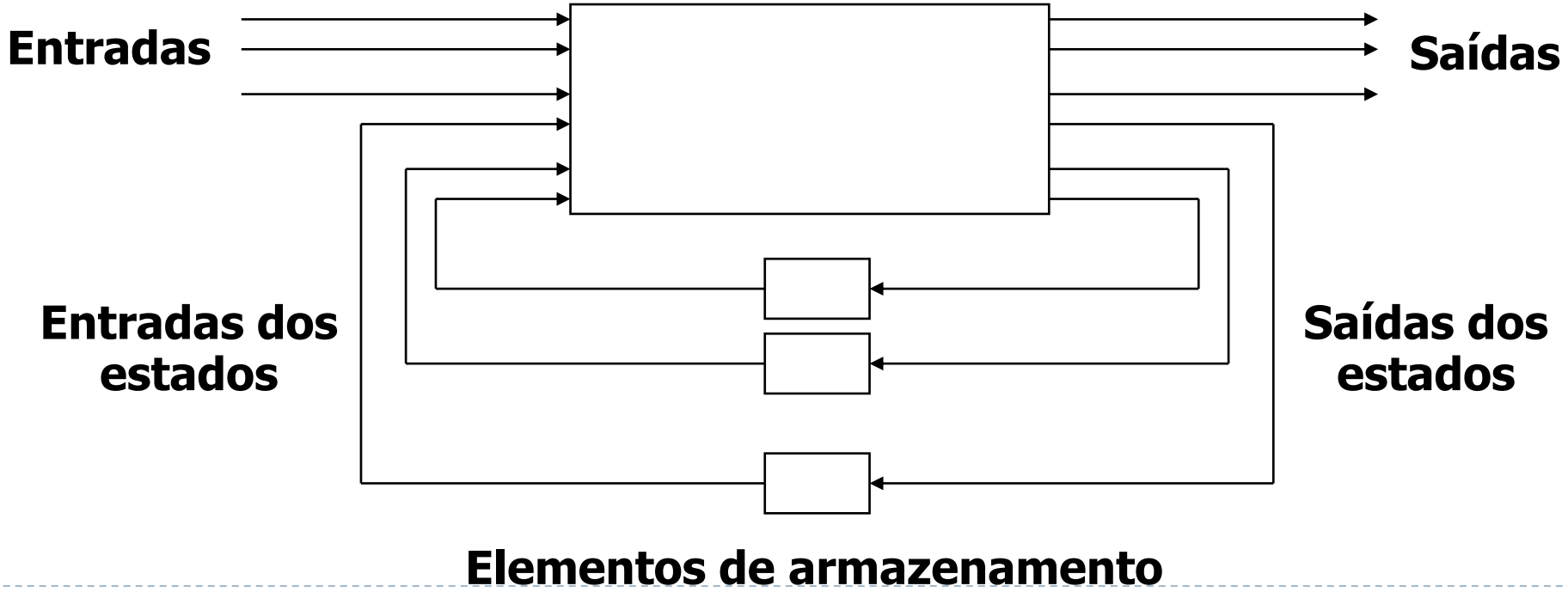
Máquinas de estado finito

- Circuitos sequenciais
  - Elementos sequenciais primitivos
  - Lógica combinacional
- Modelos de representação de circuitos sequenciais
  - Máquinas de estado finito (Moore e Mealy)
- Revisão de circuitos sequenciais básicos
  - Registradores de deslocamento
  - Contadores
- Processo de desenvolvimento
  - Diagramas de estado
  - Tabela de transição de estados
  - Funções de próximo estado

Linguagens de descrição de hardware

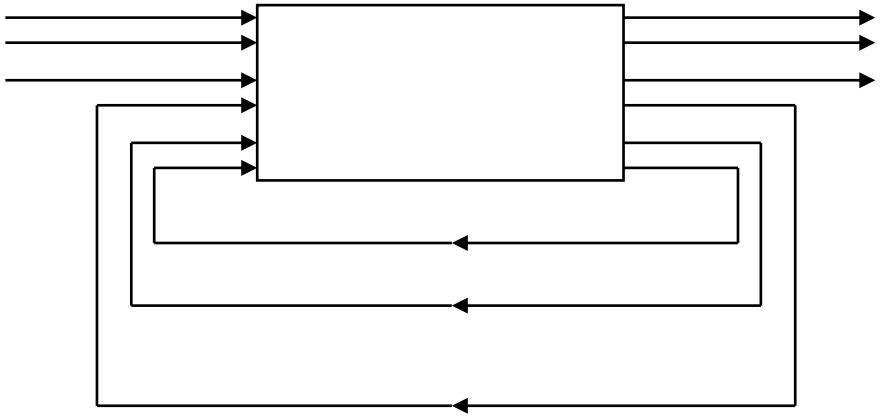
Abstração de elementos de estado

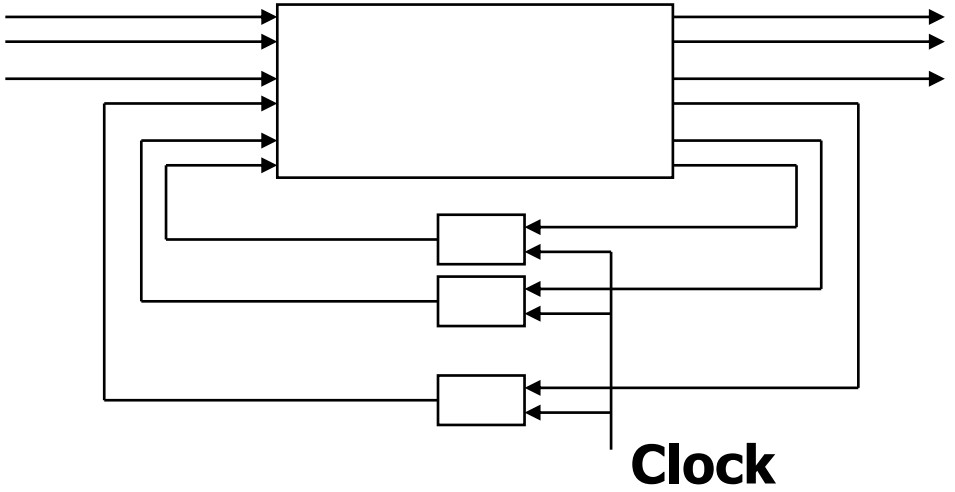
- Divide o circuito em lógica combinacional e estados
- Identifica os laços de realimentação facilitando a quebra em ciclos
- Implementação dos elementos de armazenamento leva a várias formas de lógica sequencial



Formas de lógica sequencial

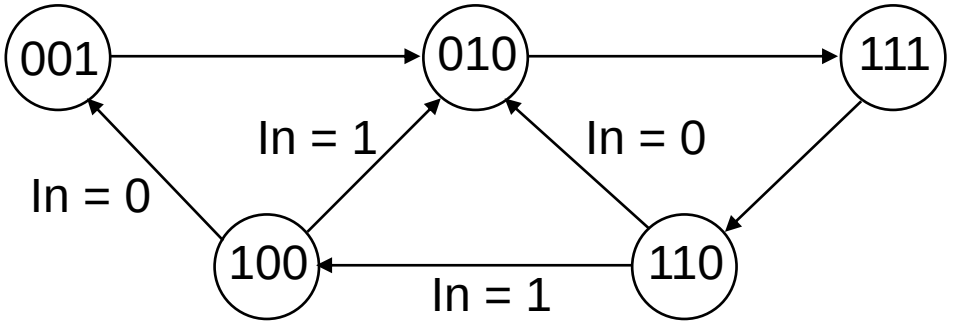
- Lógica sequencial assíncrona
  - Mudanças de estado ocorrem quando as entradas de estado se alteram
- Lógica sequencial síncrona
  - Mudanças de estado ocorrem nas transições do clock dos elementos de armazenamento





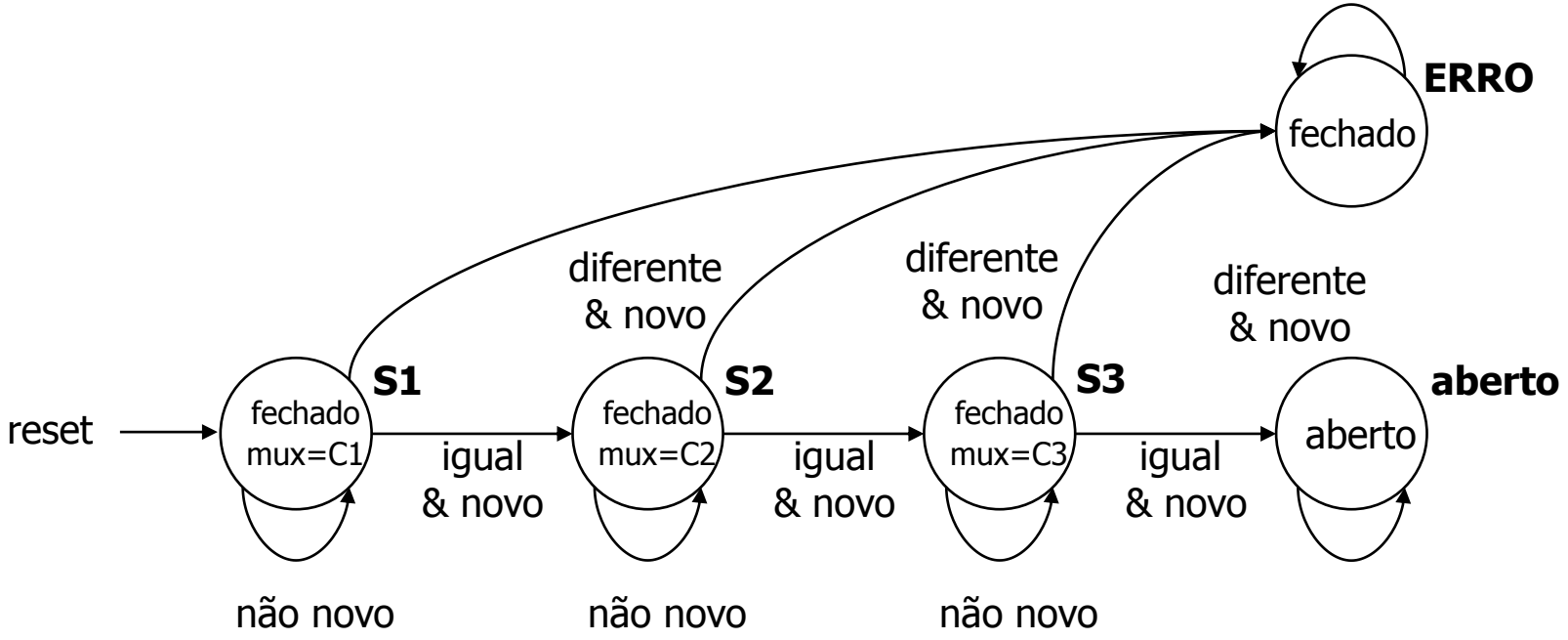
Representações de máquinas de estado finito

- Os estados são determinados pelos valores dos elementos de armazenamento
- As transições são mudanças de estado
- O clock define quando cada estado pode mudar utilizando os elementos de armazenamento
- Lógica sequencial
  - Sequencia de séries de estados
  - Baseados na sequencia de valores de entrada de sinais
  - Período de clock define elementos da sequencia



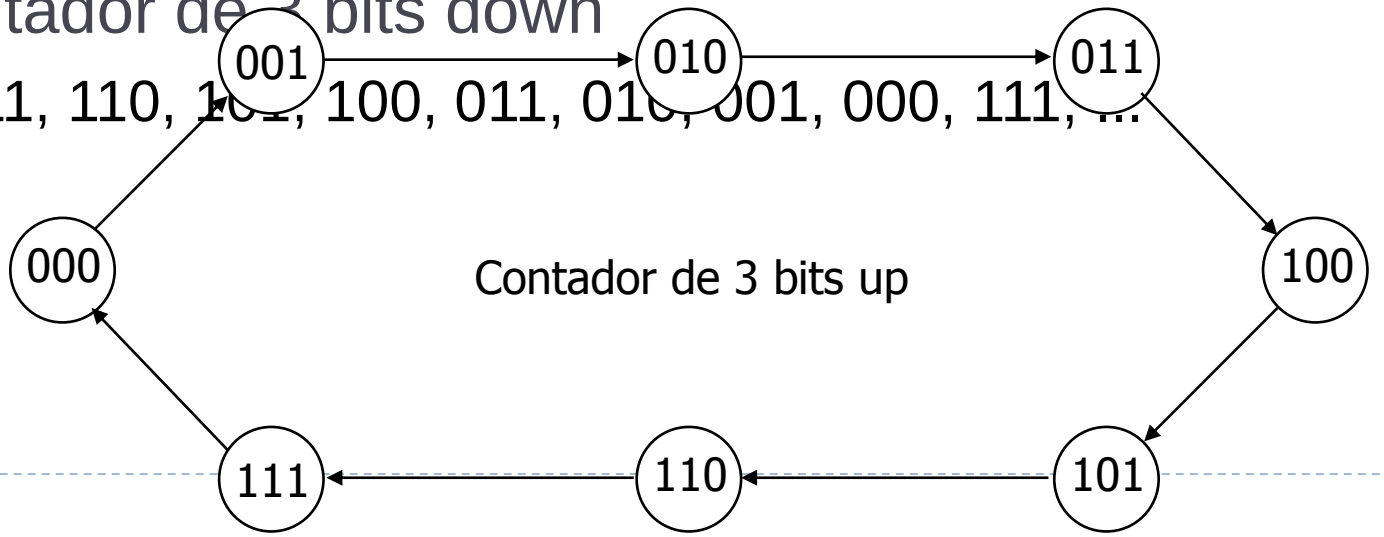
Exemplos de máquinas de estado finito

- 5 estados
- 5 transições para o próprio estado
- 6 transições para outros estados
- 1 transição de reset de todos os estados para o estado S1



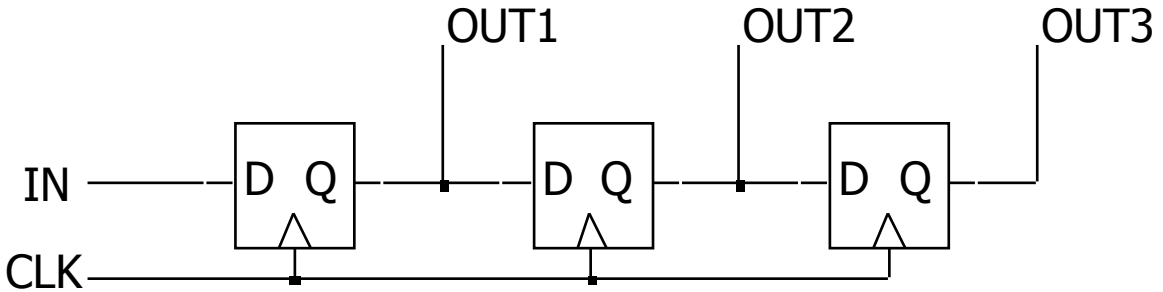
Contadores são máquinas de estados finitos simples

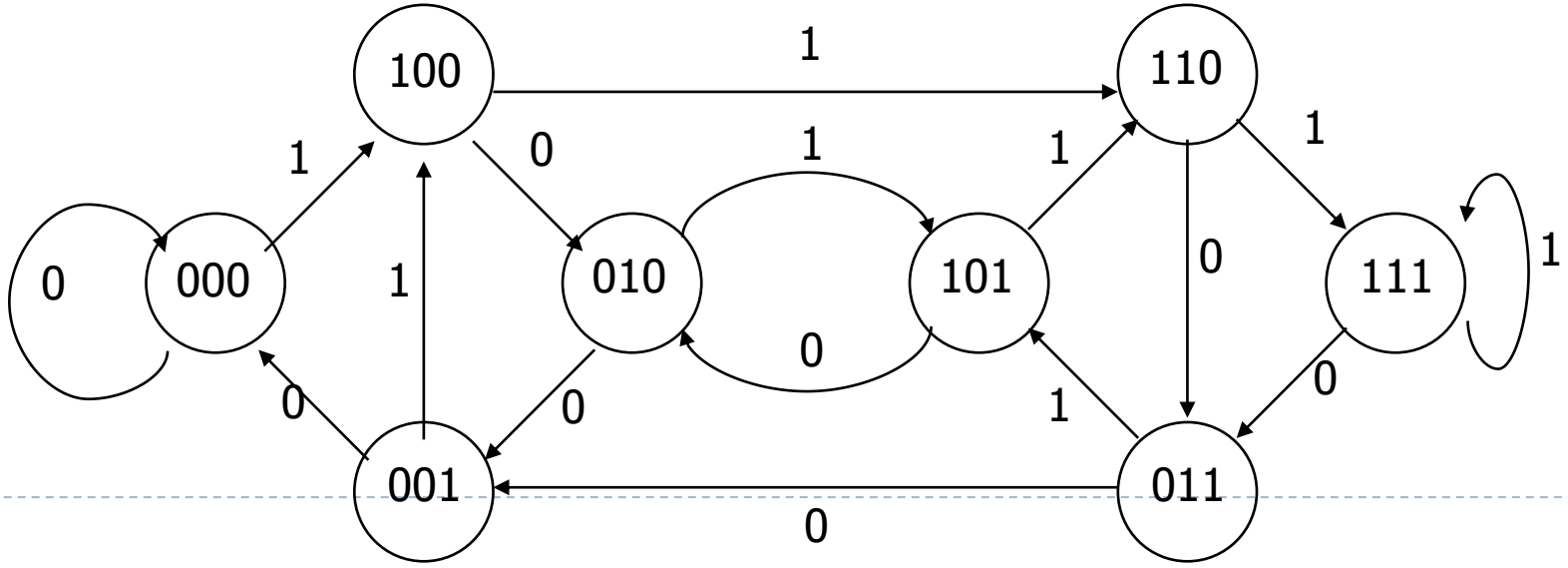
- Contadores
  - Executam sequencia pré-definida de estados em após habilitação
- Vários tipos de contadores: binário, BCD, Código Gray
  - Contador de 3 bits up
    - 000, 001, 010, 011, 100, 101, 110, 111, 000, ...
  - Contador de 3 bits down
    - 111, 110, 101, 100, 011, 010, 001, 000, 111, ...



Exemplos de máquinas de estado finito

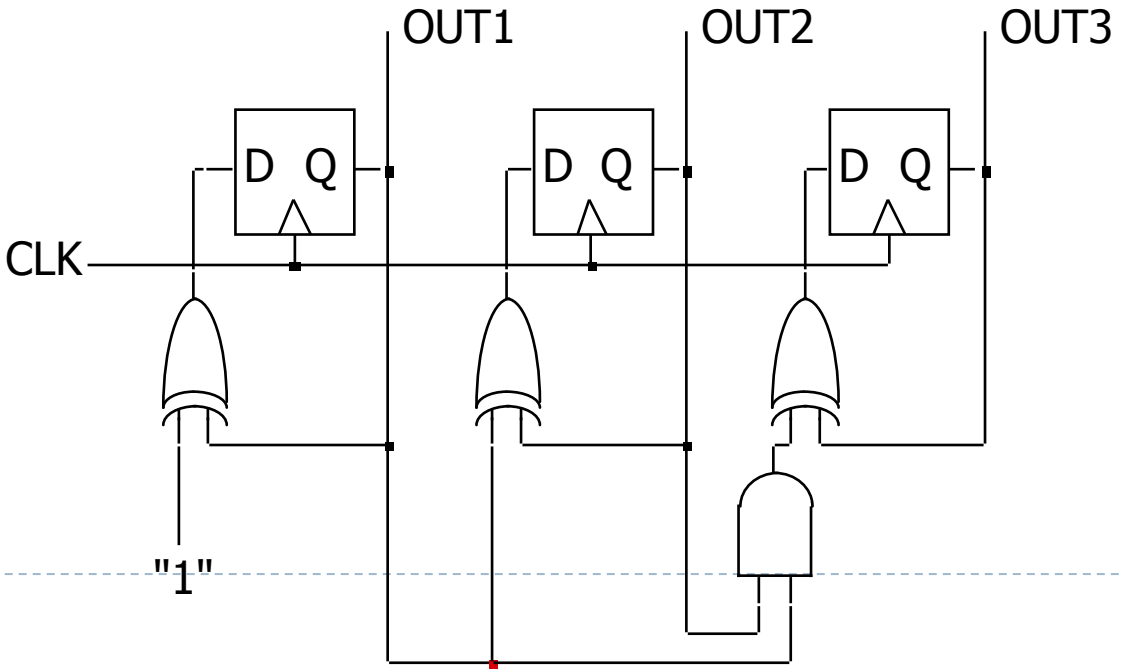
- Registrador de deslocamento
  - Valor de entrada apresentado nas transições
  - Valor da saída é apresentado no nó de estado





Como transformar um diagrama de estados em lógica?

- Contador
  - 3 flip-flops armazenam o estado
  - Lógica para calcular o próximo estado
  - Sinal de clock controla quando o flip-flop pode mudar
    - Deve esperar a lógica combinacional calcular o próximo estado
    - Não espera muito para degradar o desempenho

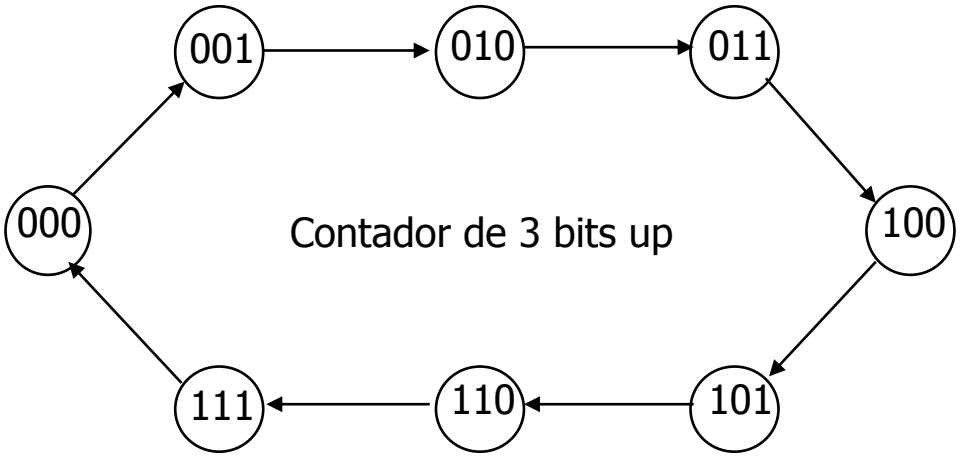


Projeto de máquinas de estado finito

- Contadores são simples
  - Saída é o próprio estado
  - Escolha do próximo estado é baseada na entrada
- Diagrama de estados e tabela de transição de estados
  - Similar a uma tabela-verdade
- Codificação dos estados
  - Define forma de representar os estados
  - Para contadores é simples, o próprio valor
- Implementação
  - 1 flip-flop para cada bit de estado
  - Lógica combinacional baseada na codificação

Projeto de máquinas de estado finito

- Transição do diagrama de estados para a tabela de transição de estados codificada
  - Similar a uma tabela-verdade
  - Codificação de estados
    - Para contadores é o próprio estados



Estado atual		Próximo estado	
0	000	001	1
1	001	010	2
2	010	011	3
3	011	100	4
4	100	101	5
5	101	110	6
6	110	111	7
7	111	000	0

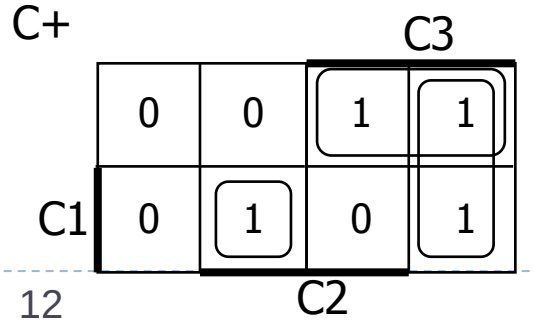
Implementação

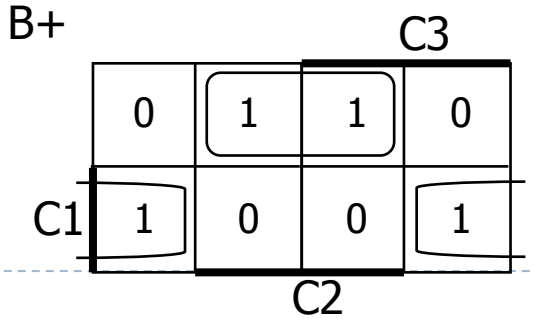
- Flip-flop D para cada bit de estado
- Lógica combinacional para codificação

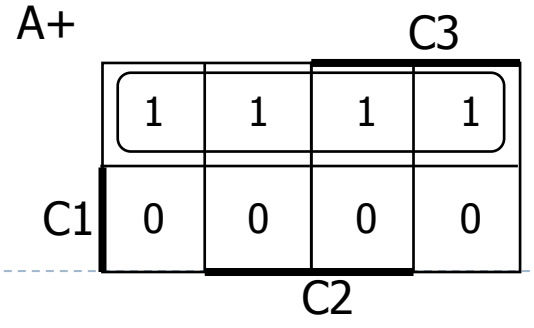
C	B	A	C+	B+	A+
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

Notação de Verilog para representar a entrada de um Flip-Flop D

A+ <= A'
B+ <= AB' + A'B
<= A xor B
C+ <= ABC' + A'C + B'C
<= (AB)C' + (A' + B')C
<= (AB)C' + (AB)'C
<= (AB) xor C

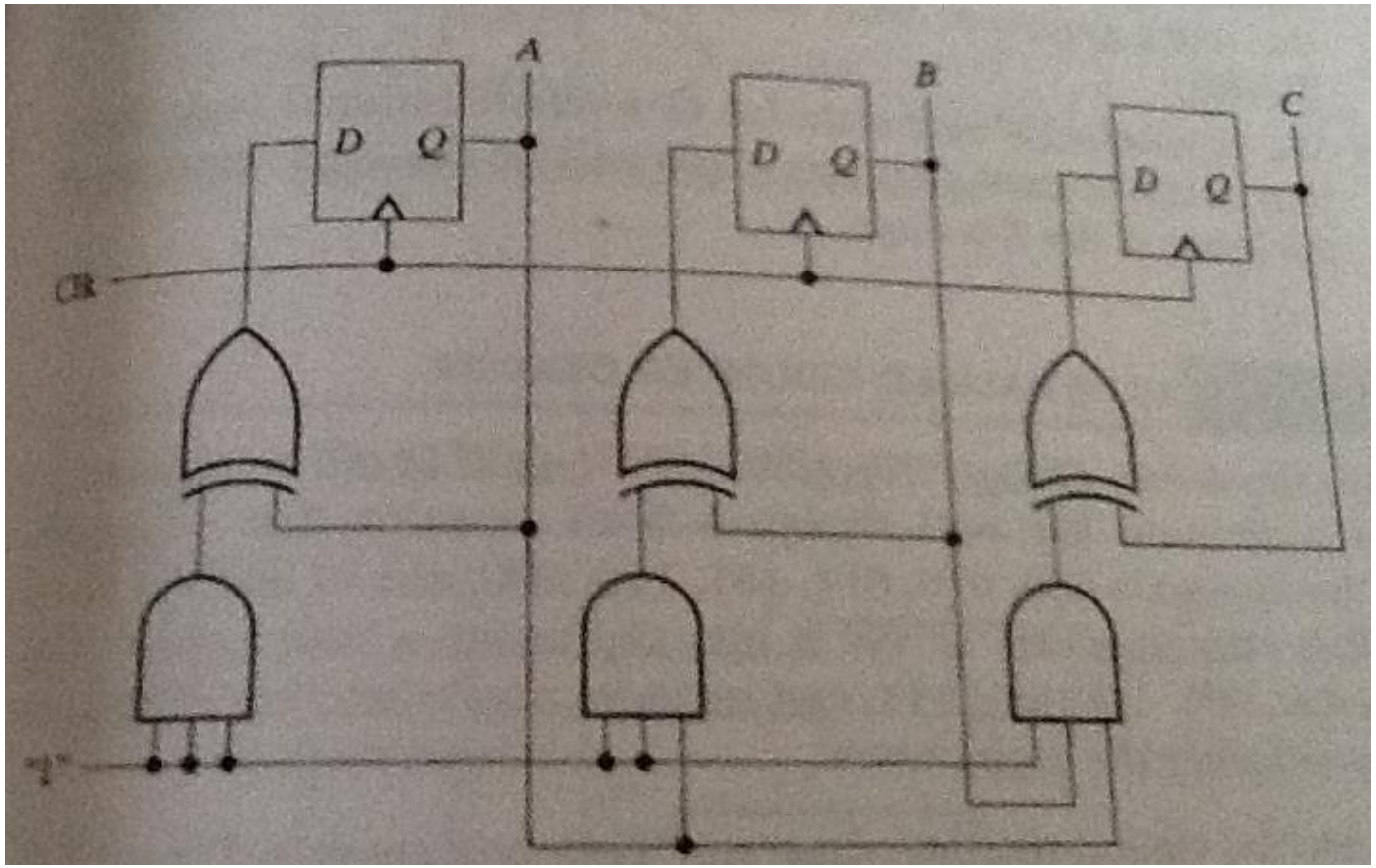






Implementação

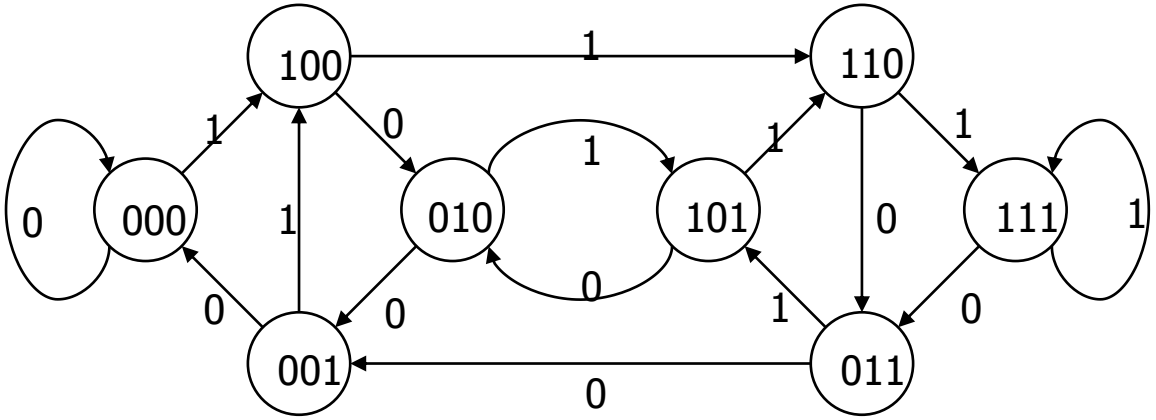
- Circuito final



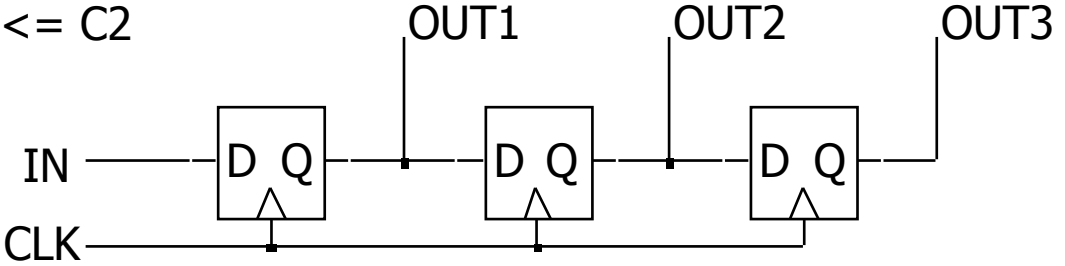
Registrador de deslocamento

- Entrada determina o próximo estado

In	C1	C2	C3	N1	N2	N3
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	0	1	0
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	0	1	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	1	0	1
1	0	1	1	1	0	1
1	1	0	0	1	1	0
1	1	0	1	1	1	0
1	1	1	0	1	1	1
1	1	1	1	1	1	1

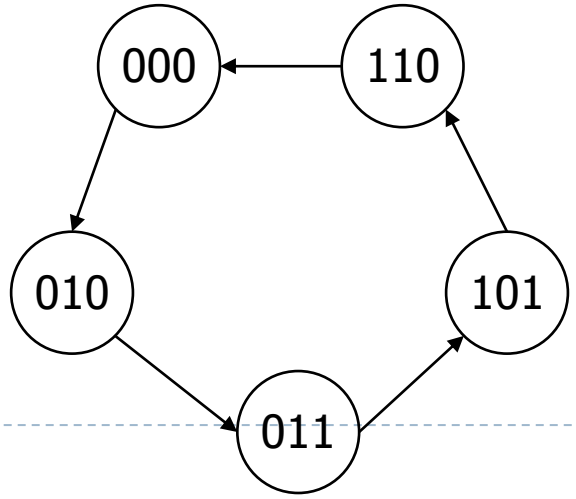


N1 <= In
N2 <= C1
N3 <= C2



Exemplo mais complexo

- Contador mais complexo
  - Repete 5 estados em sequencia
  - Não é uma representação binária
- Passo 1: Derivar o diagrama de estados
  - Sequencia de contagem: 000, 010, 011, 101, 110
- Passo 2: Derivar a tabela de transição de estados a partir do diagrama estados



Estado atual			Próximo estado		
C	B	A	C+	B+	A+
0	0	0	0	1	0
0	0	1	–	–	–
0	1	0	0	1	1
0	1	1	1	0	1
1	0	0	–	–	–
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	–	–	–

Observe que condições de don't care são causadas por códigos de estado não utilizados

Exemplo mais complexo

- Passo 3: mapas K para funções de próximo estado

C+

A \ BC	00	01	11	10
0	0	0	0	X
1	X	1	X	1

B+

A \ BC	00	01	11	10
0	1	1	0	X
1	X	0	X	1

A+

A \ BC	00	01	11	10
0	0	1	0	X
1	X	1	X	0

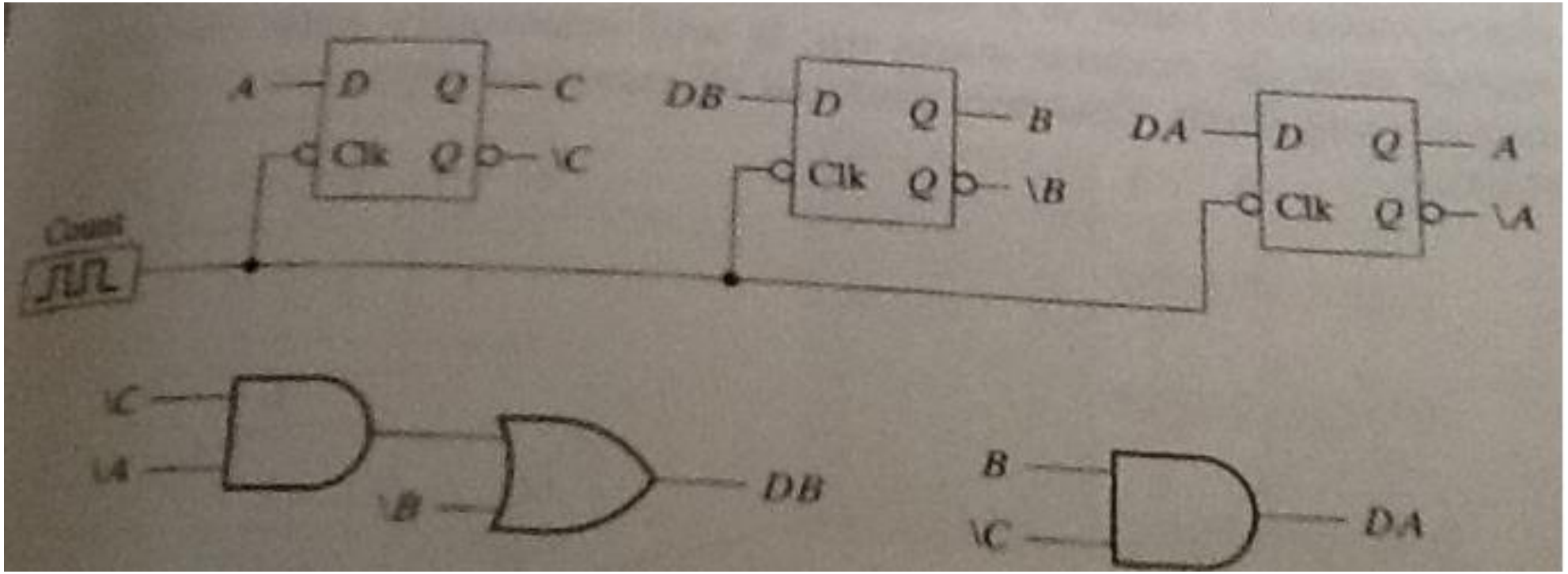
C+ <= A

B+ <= B′ + A′C′

A+ <= BC′

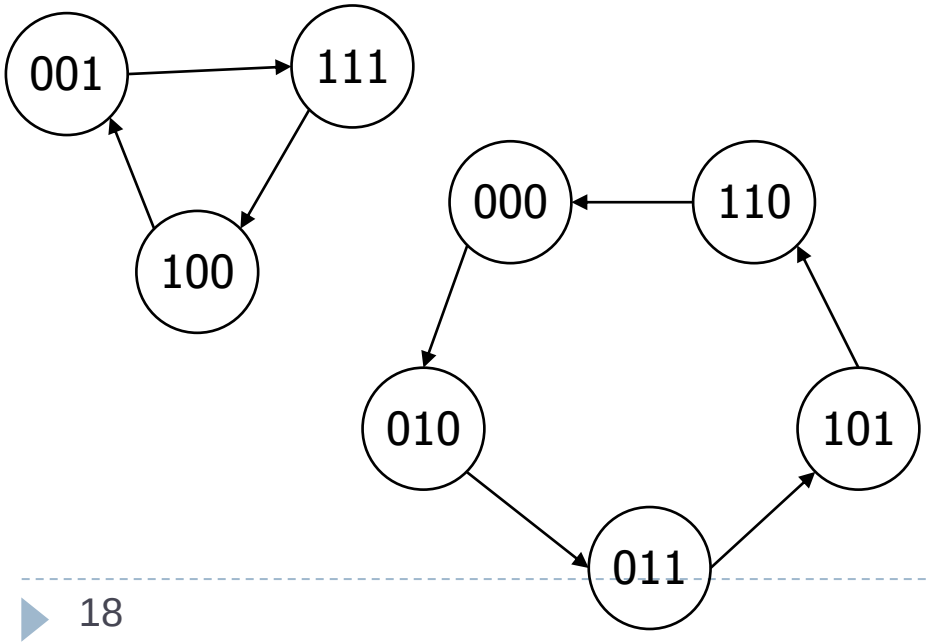
Exemplo mais complexo

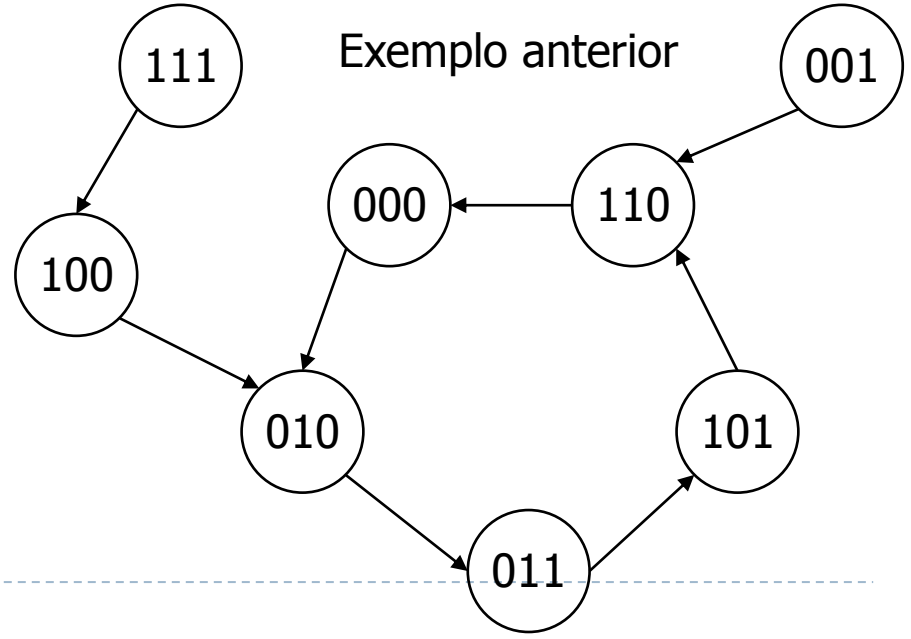
- Circuito final



Contadores auto iniciáveis

- Inicialização
  - Contador pode estar em um estado não usado ou inválido
  - Projetista deve garantir que o contador entrará em um estado válido
- Solução auto iniciável
  - Contador deve prever transições de estados inválidos para estados válidos
  - Pode limitar a utilização de don't cares





Contadores auto iniciáveis

- Refazendo a parte combinacional do contador utilizando a nova tabela verdade

C+

A \ BC	00	01	11	10
0	0	0	0	0
1	1	1	1	1

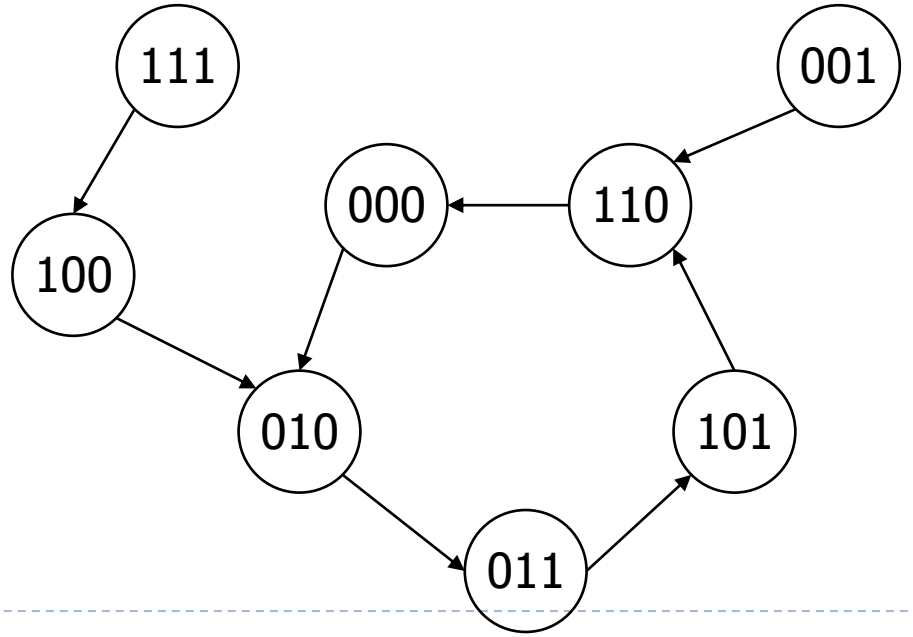
B+

A \ BC	00	01	11	10
0	1	1	0	1
1	1	0	0	1

A+

A \ BC	00	01	11	10
0	0	1	0	0
1	0	1	0	0

Estado atual C	B	A	Próximo estado C+	B+	A+
0	0	0	0	1	0
0	0	1	1	1	0
0	1	0	0	1	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	1	0	0

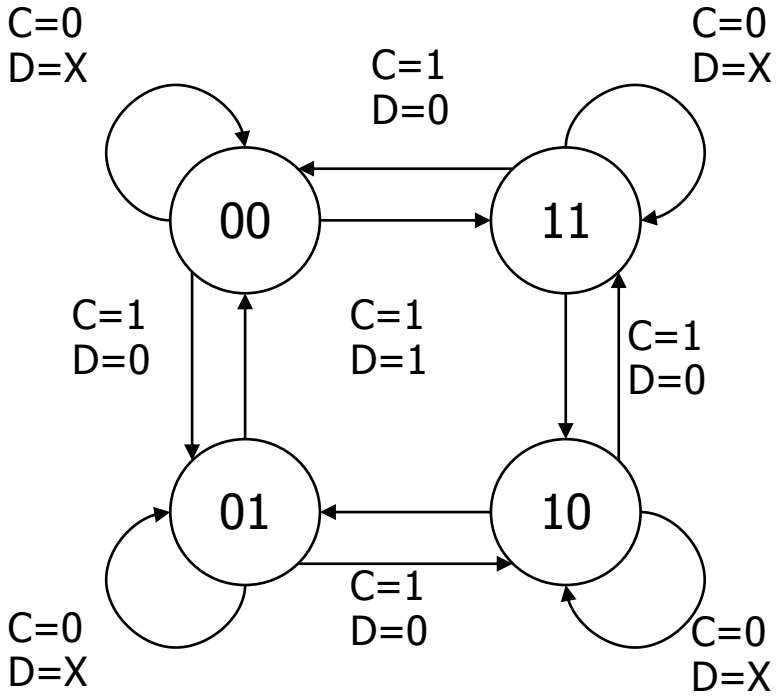


Atividade

- Contador crescente-decrescente de 2 bits
  - Direção
    - D = 0
      - Contagem crescente
    - D = 1
      - Contagem decrescente
  - Contagem
    - C = 0
      - Para contagem
    - C = 1
      - Continua a contagem

Atividade

- Contador crescente-decrescente de 2 bits
  - Direção
    - D = 0
      - Contagem crescente
    - D = 1
      - Contagem decrescente
  - Contagem
    - C = 0
      - Para contagem
    - C = 1
      - Continua a contagem
- Diagrama de estados



Atividade

- Contador crescente-decrescente de 2 bits
  - Direção
    - D = 0
      - Contagem crescente
    - D = 1
      - Contagem decrescente
  - Contagem
    - C = 0
      - Para contagem
    - C = 1
      - Continua a contagem

- Tabela de transição de estados

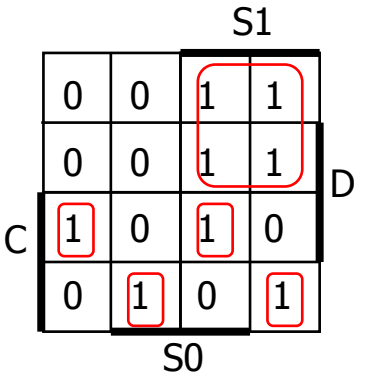
S1	S0	C	D	N1	N0
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	1	0
0	1	1	1	0	0
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	0	0
1	1	1	1	1	0

Atividade

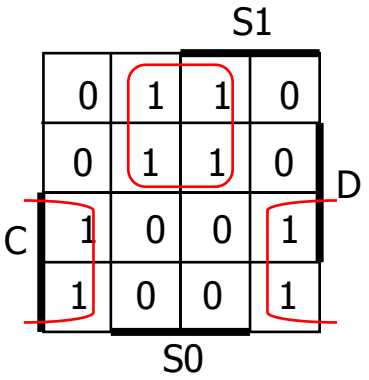
- Contador crescente-decrescente de 2 bits
  - Transição de estados

S1	S0	C	D	N1	N0
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	1	0
0	1	1	1	0	0
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	0	0
1	1	1	1	1	0

  - Mapas de Karnaugh e equações booleanas simplificadas



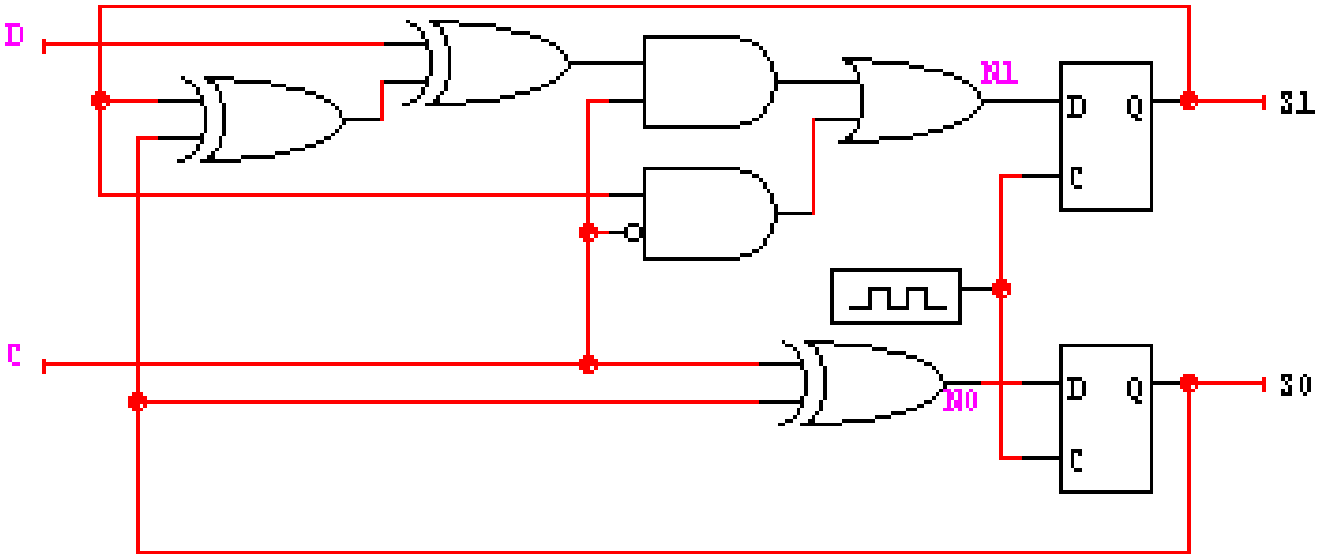
$$N1 = C'S1 + CDS0'S1' + CDS0S1 + CD'S0S1' + CD'S0'S1 = C'S1 + C(D'(S1 \oplus S0) + D(S1 \equiv S0))$$



$$N0 = CS0' + C'S0$$

Atividade

- Contador crescente-decrescente de 2 bits
  - Circ

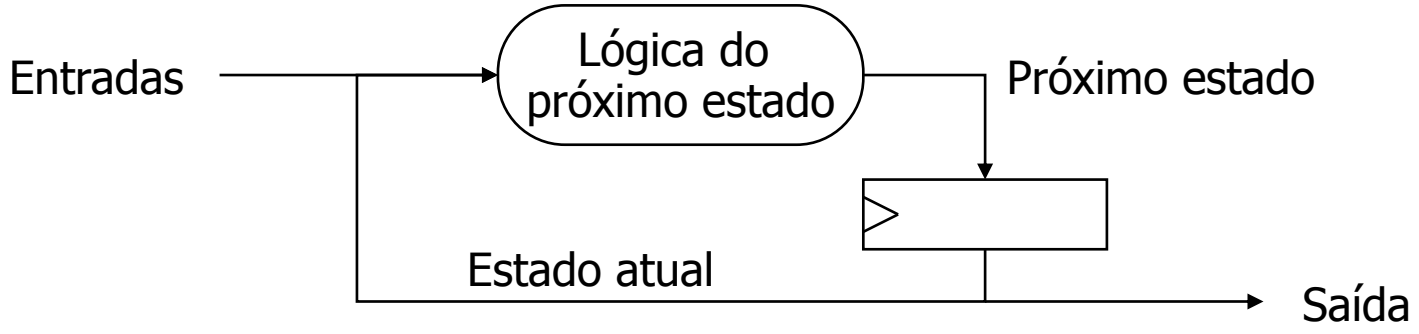


N1 = C'S1
+ CDS0'S1' + CDS0S1
+ CD'S0S1' + CD'S0'S1
= C'S1
+ C(D'(S1 ⊕ S0) + D(S1 ≡ S0))

N0 = CS0' + C'S0

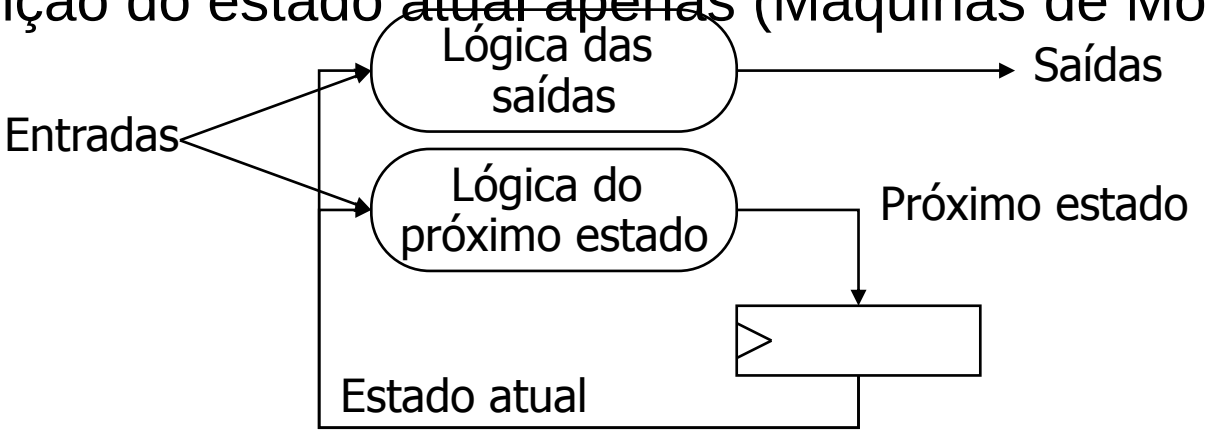
Modelo de contadores e registradores de deslocamento

- Valores armazenados representam o estado do circuito
- Lógica combinacional calcula
  - Próximo estado
    - Em função do estado atual das entradas
  - Saídas
    - Valores dos flip-flops



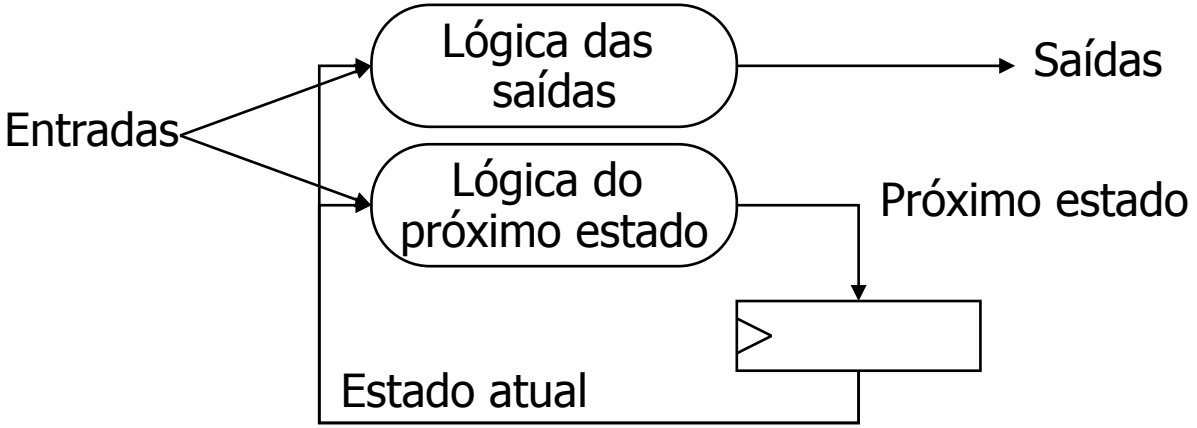
Modelo genérico de máquinas de estado

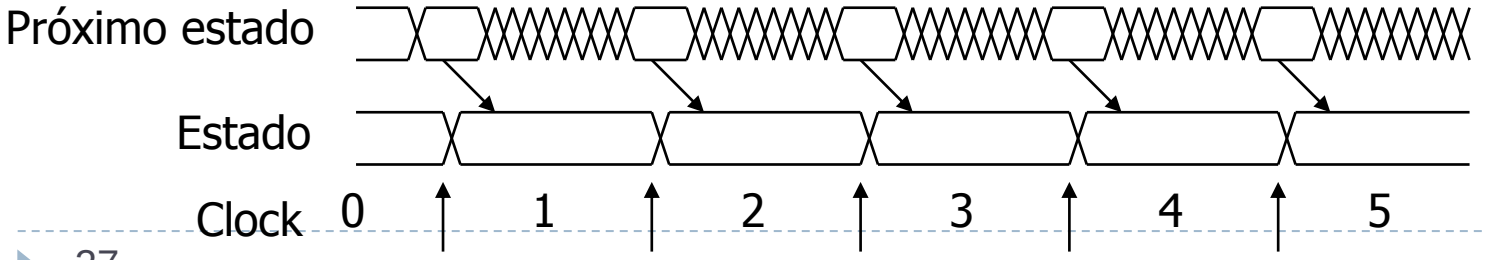
- Valores armazenados representam o estado do circuito
- Lógica combinacional calcula
  - Próximo estado
    - Em função do estado atual das entradas
  - Saídas
    - Função do estado atual e das entradas (Máquinas de Mealy)
    - Função do estado atual apenas (Máquinas de Moore)



Modelo genérico de máquinas de estado

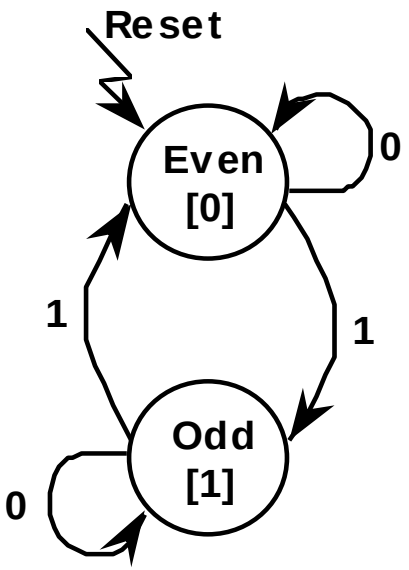
- Estados: S_1, S_2, ..., S_k
- Entradas: I_1, I_2, ..., I_m
- Saídas: O_1, O_2, ..., O_n
- Funções de transição: $F_s(S_i, I_j)$
- Função de saída: $F_o(S_i)$ or $F_o(S_i, I_j)$





Conceito de máquina de estado

- Identifica número ímpar de 1's
  - Em caso positivo, saída=1



Present State	Input	Next State	Output
Even	0	Even	0
Even	1	Odd	0
Odd	0	Odd	1
Odd	1	Even	1

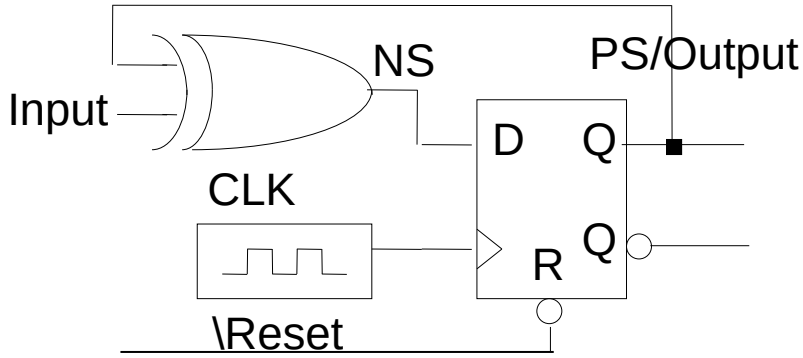
Present State	Input	Next State	Output
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	1

Conceito de máquina de estado

- Identificador de paridade ímpar

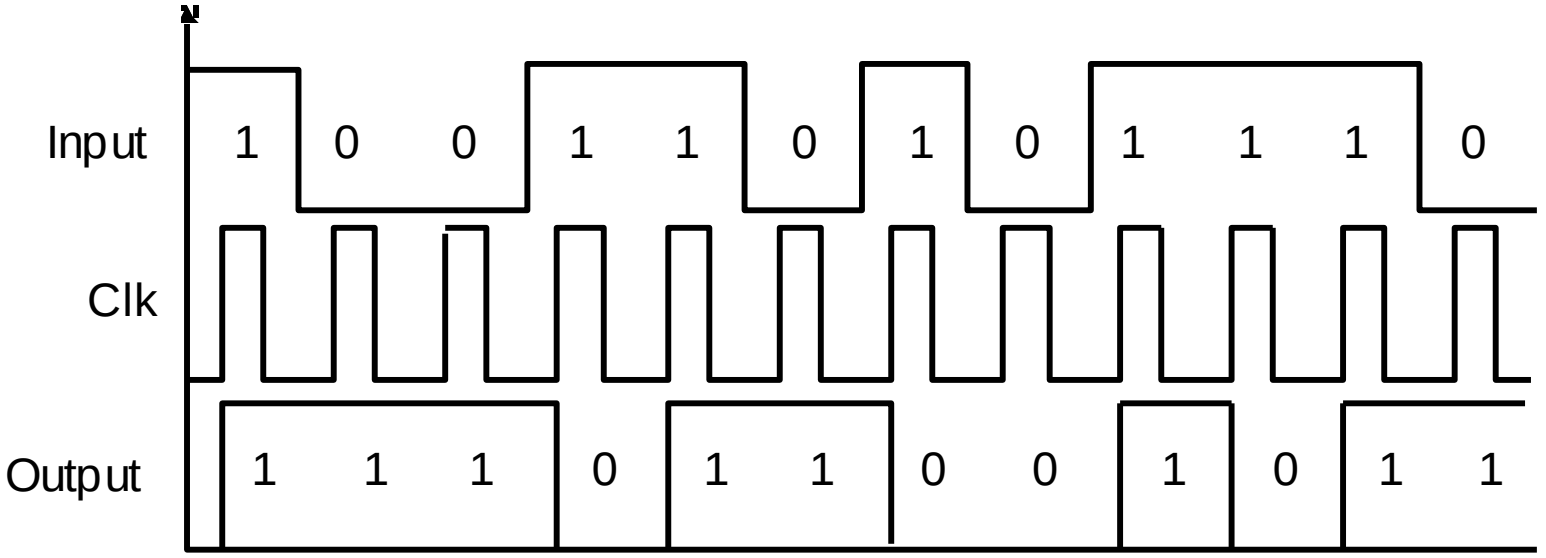
Present State	Input	Next State	Output
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	1

D_Next State



Conceito de máquina de estado

- Identificador de paridade ímpar

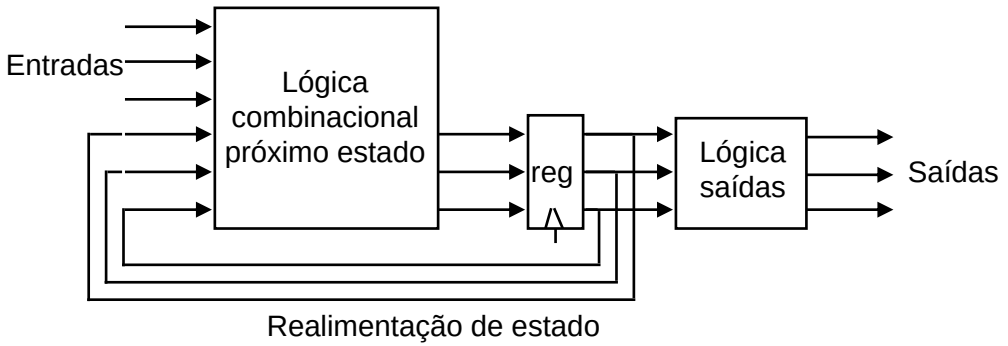


Comparação entre máquinas Mealy e Moore

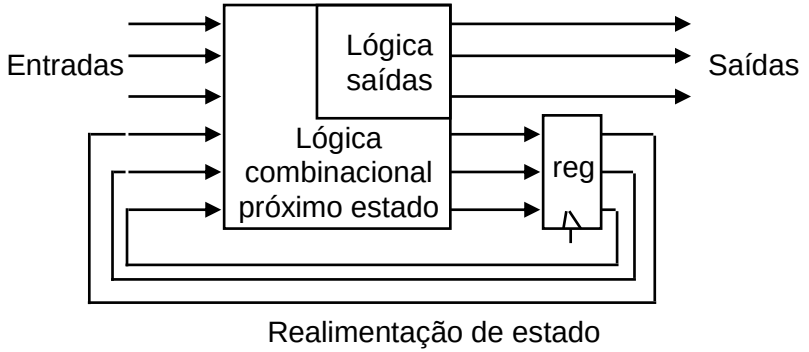
- Máquinas Mealy tendem a ter menos estados
  - Diferentes saídas em arcos (n^2) ao invés de estados (n)
- Máquinas Moore são mais seguras por utilizar
  - Mudanças nas saídas um ciclo após uma borda
  - Em máquinas Mealy, a mudanças nas entradas por causar mudanças nas saídas tão logo o cálculo da lógica seja realizado
    - Pode causar problemas quando duas máquinas estão interconectadas
- Máquinas Mealy reagem mais rapidamente às entradas
  - Reagem no mesmo ciclo, pois não dependem de clock
  - Em máquinas Moore mais lógica pode ser necessária para decodificar os estados em saídas – mais atrasos de

Comparação entre máquinas Mealy e Moore

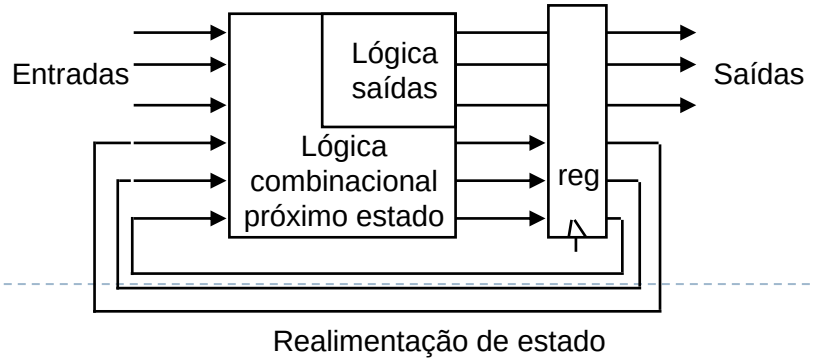
- Moore



- Mealy

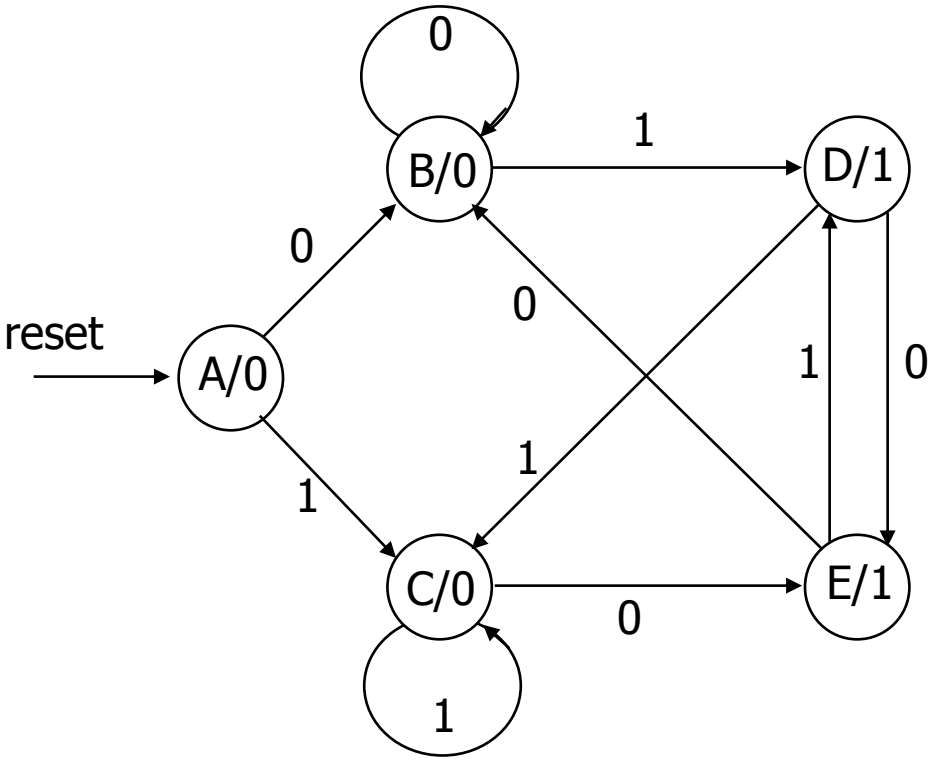


- Mealy síncrona



Especificação de saídas para uma máquina Moore

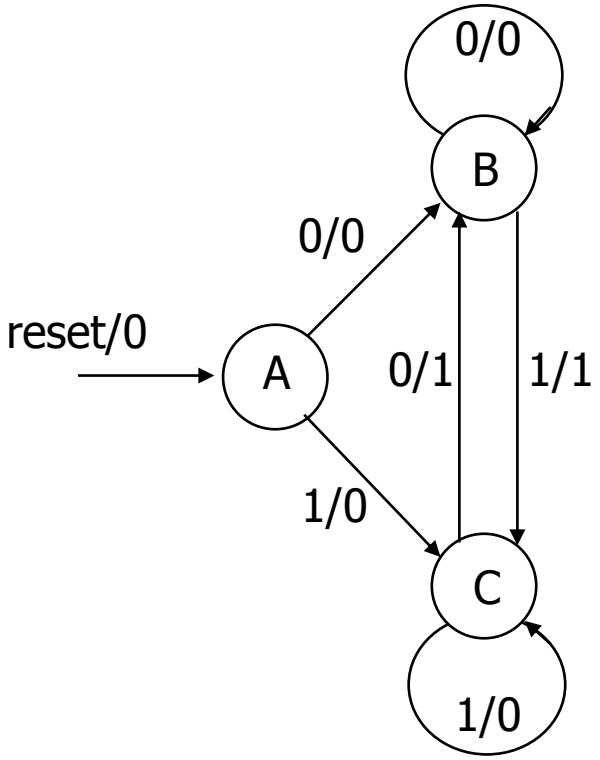
- Saída é unicamente uma função do estado
  - Especificação em diagrama de estados
  - Exemplo
    - Detector de sequencia 01 ou 10



reset	Entr.	Estado Atual	Próximo estado	saída
1	–	–	A	
0	0	A	B	0
0	1	A	C	0
0	0	B	B	0
0	1	B	D	0
0	0	C	E	0
0	1	C	C	0
0	0	D	E	1
0	1	D	C	1
0	0	E	B	1
0	1	E	D	1

Especificação de saídas para uma máquina Mealy

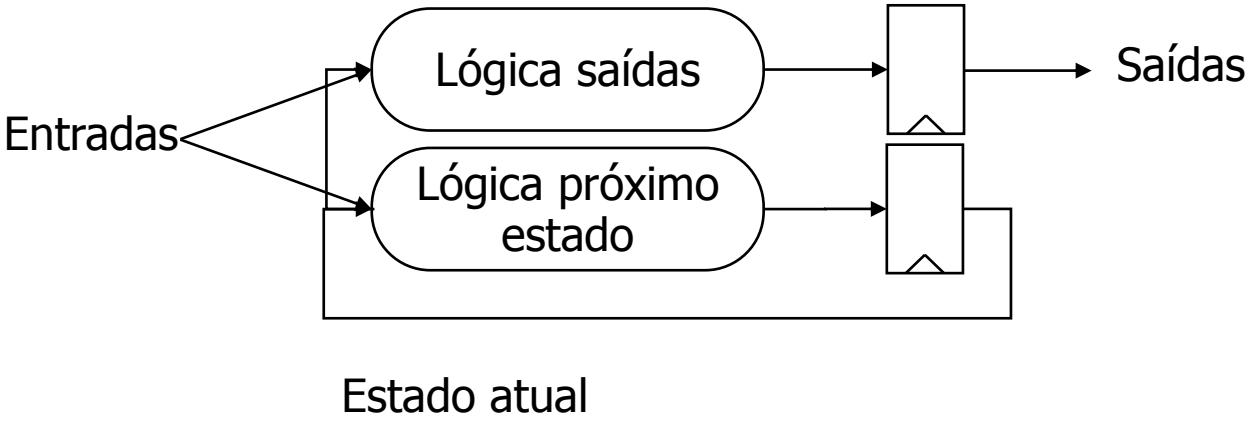
- Saída é uma função do estado e das entradas
  - Especificação em diagrama de estados
  - Exemplo
    - Detector de sequencia 01 ou 10



reset	Entr.	Estado atual	Próximo estado	saída
1	–	–	A	0
0	0	A	B	0
0	1	A	C	0
0	0	B	B	0
0	1	B	C	1
0	0	C	B	1
0	1	C	C	0

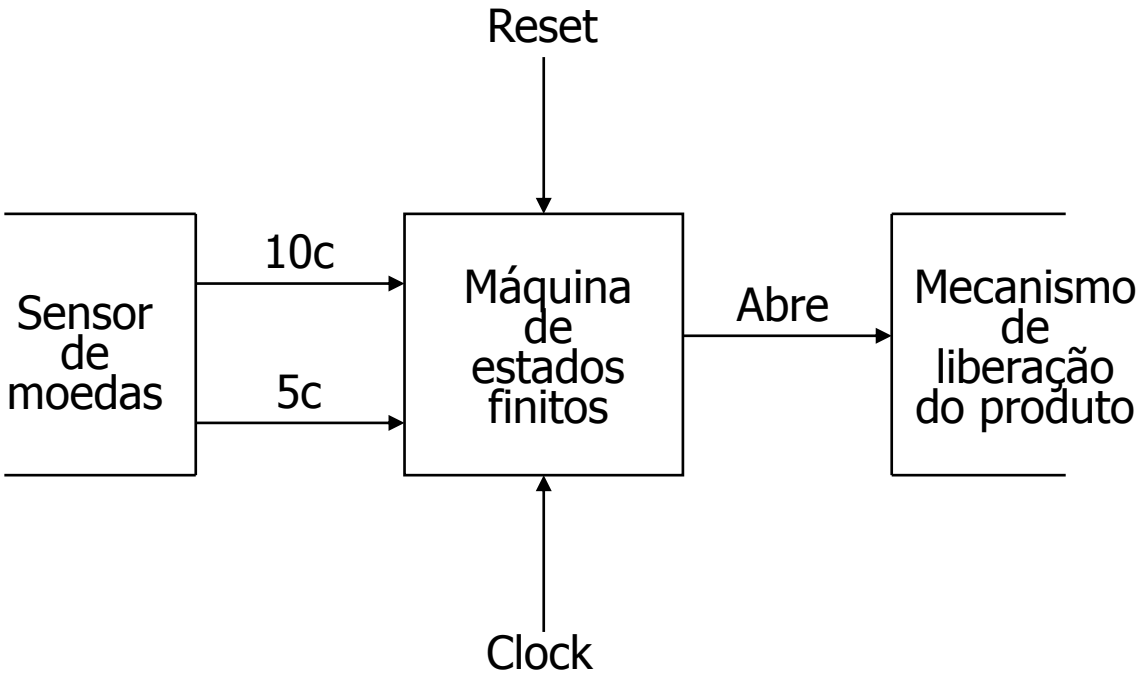
Máquina Mealy com registradores

- Máquina Mealy síncrona
  - Estados e saídas com registradores
  - Evita saídas com glitch
  - Fácil implementação em PLDs
- Máquina Moore sem decodificação de saída
  - Saídas computadas na transição do estado seguinte
  - Enxerga saídas como um vetor de estados expandido



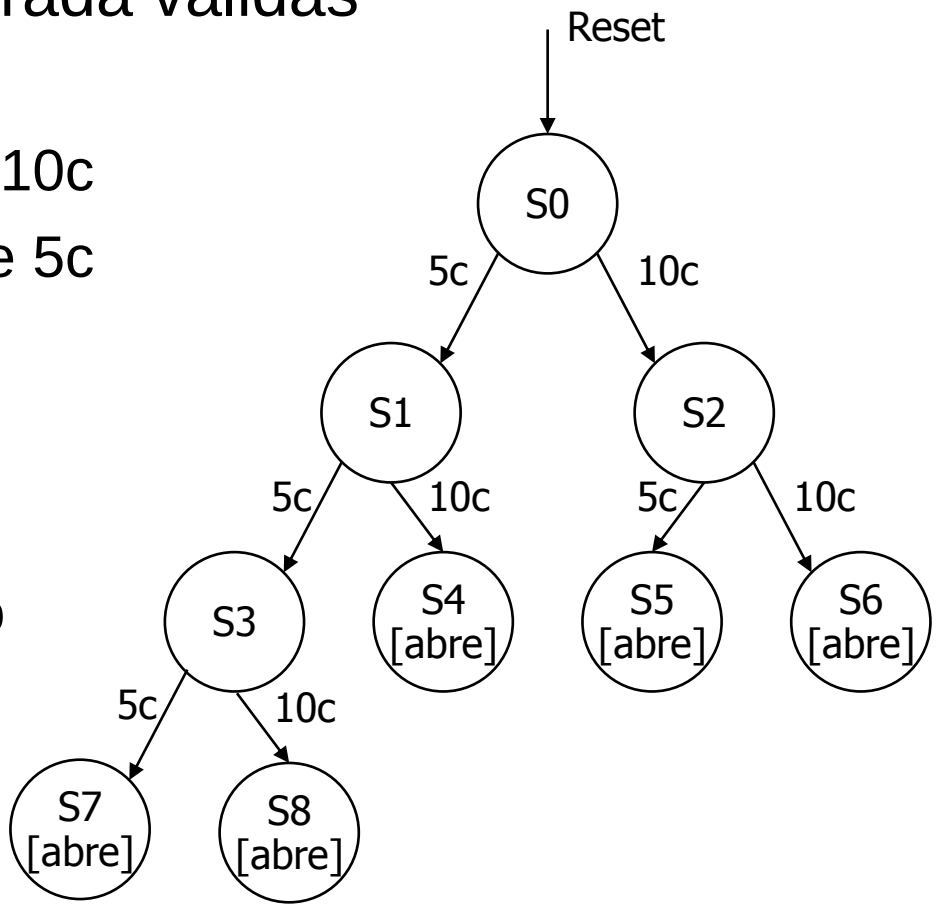
Exemplo: Máquina de vendas automática

- Libera o produto após a inserção de 15 centavos
- Moedas de 5 ou 10 centavos
- Sem troco



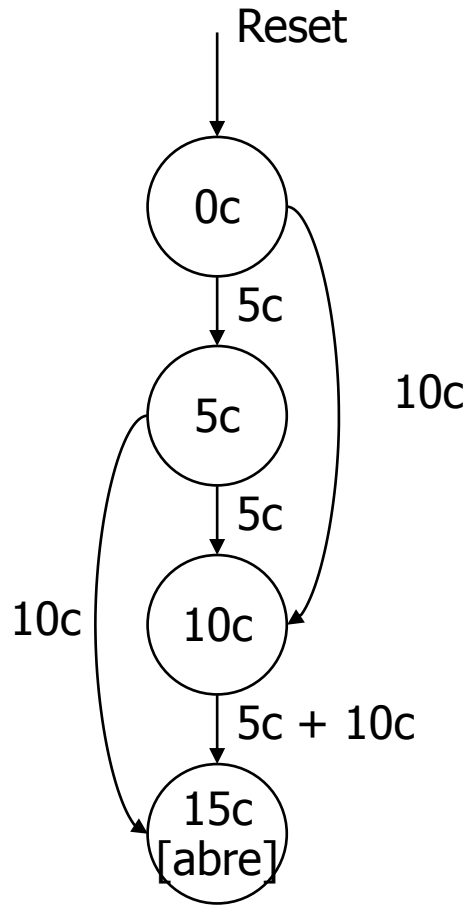
Exemplo: Máquina de vendas automática

- Representação
  - Possíveis sequências de entrada válidas
    - 3 moedas de 5c
    - 1 moeda de 5c e 1 moeda de 10c
    - 1 moeda de 10c e 1 moeda de 5c
    - 2 moedas de 10c
  - Diagrama de estados
    - Entradas: 5c, 10c, reset
    - Saídas: Liberação do produto
  - Definições
    - 5c e 10c são contabilizadas para um ciclo
    - Cada estado tem um transição para ele mesmo, representando que não foi inserida nenhuma moeda (5c = 10c = 0)



Exemplo: Máquina de vendas automática

- Minimizando o número de estados
  - Reaproveitamento de estados



Estado atual	Entradas 10c	5c	Próximo estado	Abre
0c	0	0	0c	0
	0	1	5c	0
	1	0	10c	0
	1	1	–	–
5c	0	0	5c	0
	0	1	10c	0
	1	0	15c	0
	1	1	–	–
10c	0	0	10c	0
	0	1	15c	0
	1	0	15c	0
	1	1	–	–
15c	–	–	15c	1

Tabela de transição de estados

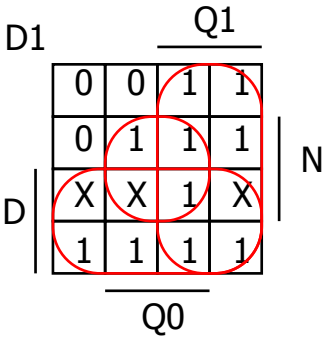
Exemplo: Máquina de vendas automática

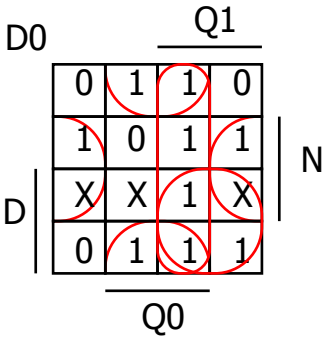
- Codificação do estados

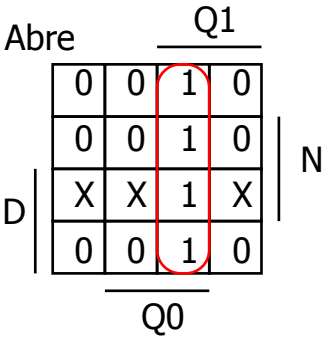
Estado atual Q1	Q0	Entradas 10c	5c	Próx. estado D1	D0	Abre
0	0	0	0	0	0	0
		0	1	0	1	0
		1	0	1	0	0
		1	1	–	–	–
0	1	0	0	0	1	0
		0	1	1	0	0
		1	0	1	1	0
		1	1	–	–	–
1	0	0	0	1	0	0
		0	1	1	1	0
		1	0	1	1	0
		1	1	–	–	–
1	1	–	–	1	1	1

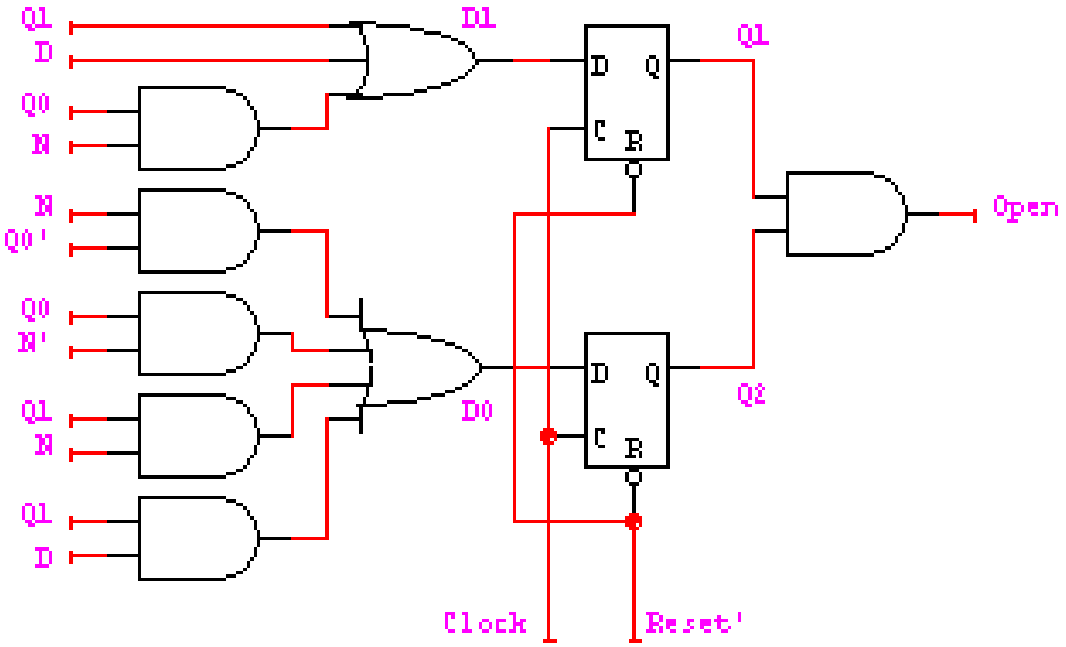
Exemplo: Máquina de Moore

- Mapeamento para lógica









D1 = Q1 + D + Q0 N

D0 = Q0′ N + Q0 N′ + Q1 N + Q1 D

OPEN = Q1 Q0

Exemplo: Máquina de vendas automática

- Codificação one-hot

Estado atual				Entradas		Próx. Estado				Saída
Q3	Q2	Q1	Q0	D	N	D3	D2	D1	D0	
0	0	0	1	0	0	0	0	0	1	0
				0	1	0	0	1	0	0
				1	0	0	1	0	0	0
				1	1	-	-	-	-	-
0	0	1	0	0	0	0	0	1	0	0
				0	1	0	1	0	0	0
				1	0	1	0	0	0	0
				1	1	-	-	-	-	-
0	1	0	0	0	0	0	1	0	0	0
				0	1	1	0	0	0	0
				1	0	1	0	0	0	0
				1	1	-	-	-	-	-
1	0	0	0	-	-	1	0	0	0	1

D0 = Q0 D' N'

D1 = Q0 N + Q1 D' N'

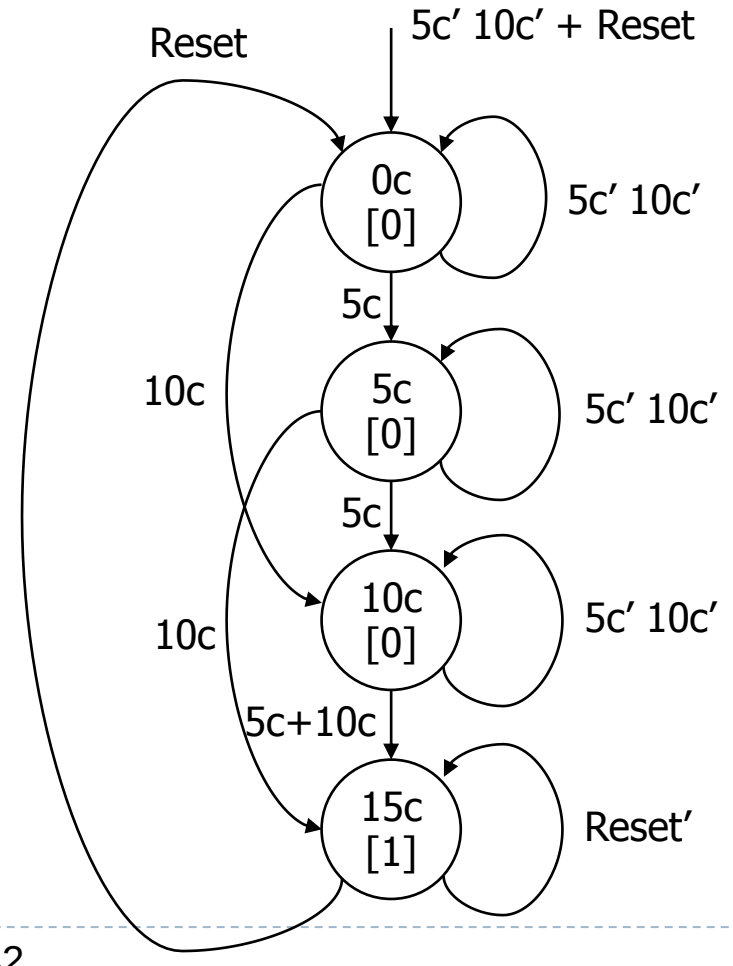
D2 = Q0 D + Q1 N + Q2 D' N'

D3 = Q1 D + Q2 D + Q2 N + Q3

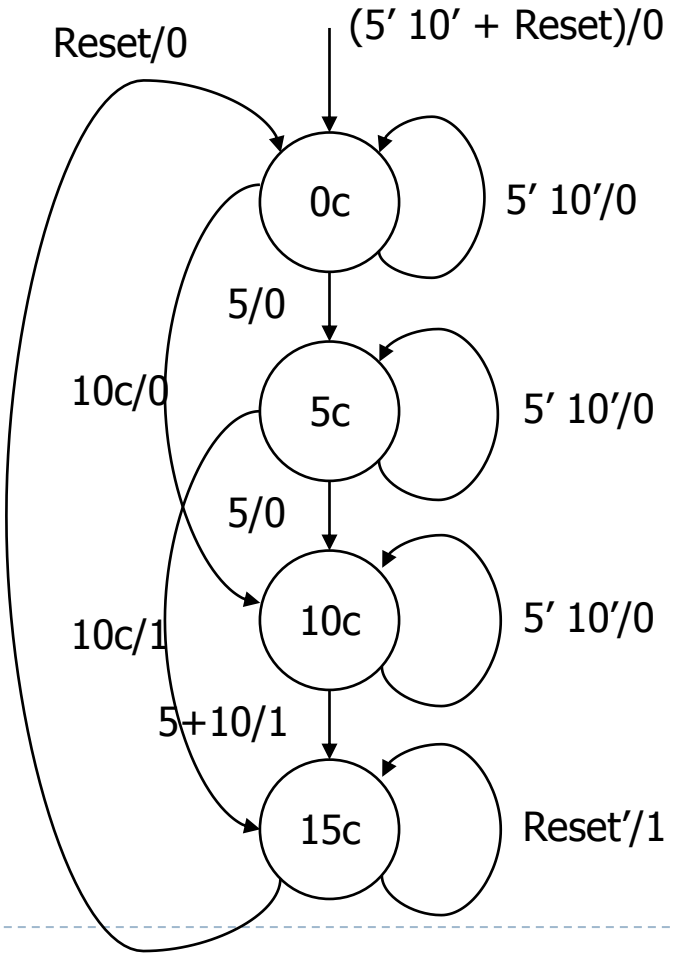
OPEN = Q3

Diagramas de estado de Moore e de Mealy

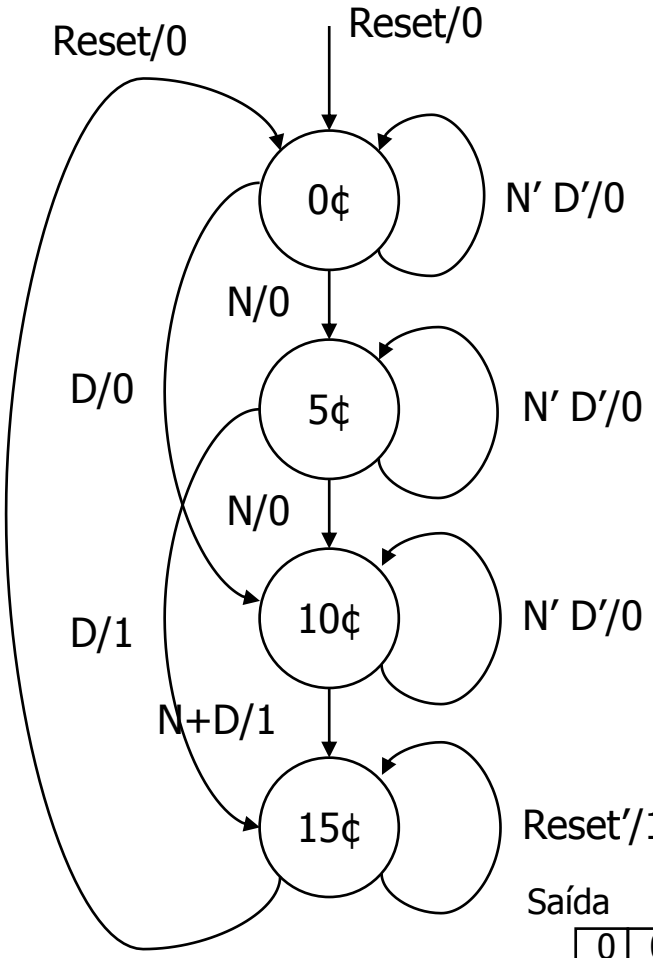
- Máquina de Moore
  - Saídas associadas com estado



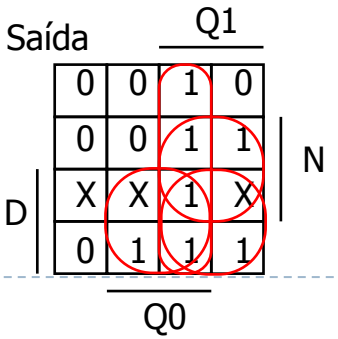
- Máquina de Mealy
  - Saídas asssociadas com transição



Exemplo: Máquina de Mealy



Estado atual Q1	Q0	Entradas D	N	Próx. est. D1	D0	Saída
0	0	0	0	0	0	0
		0	1	0	1	0
		1	0	1	0	0
		1	1	–	–	–
0	1	0	0	0	1	0
		0	1	1	0	0
		1	0	1	1	1
		1	1	–	–	–
1	0	0	0	1	0	0
		0	1	1	1	1
		1	0	1	1	1
		1	1	–	–	–
1	1	–	–	1	1	1



D0 = Q0'N + Q0N' + Q1N + Q1D

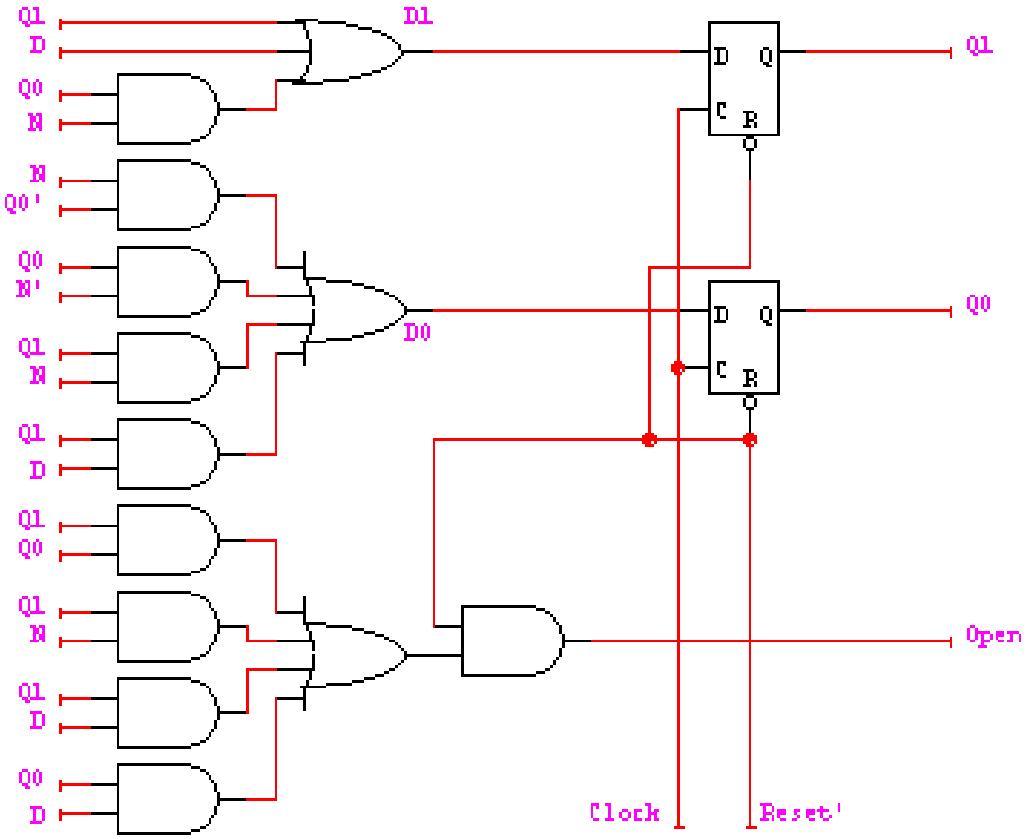
D1 = Q1 + D + Q0N

Saída = Q1Q0 + Q1N + Q1D + Q0D

Exemplo: Máquina de Mealy

D0 = Q0'N + Q0N' + Q1N + Q1D
D1 = Q1 + D + Q0N
Saída = Q1Q0 + Q1N + Q1D + Q0D

Para garantir que saída é igual a 0 durante o reset, adicione uma porta AND

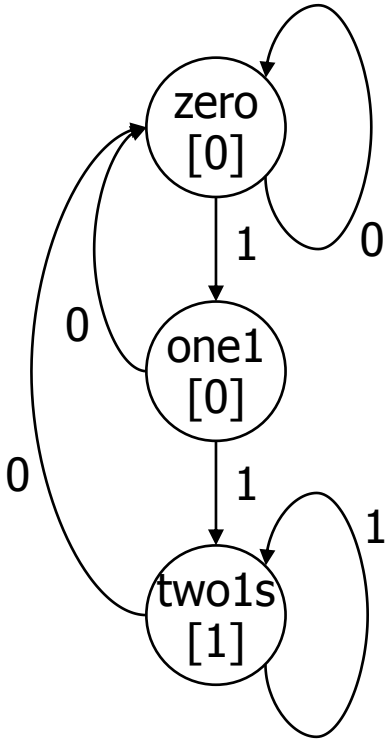


Linguagens de Descrição de Hardware e Lógica Sequencial

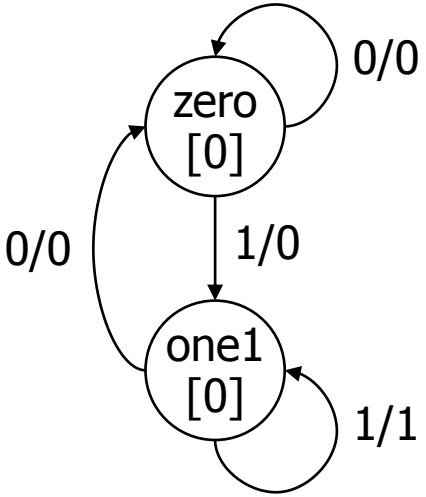
- Flip-flops
  - Representação dos clock – temporização das mudanças de estado
  - Assíncrono vs. síncrono
- Máquinas de estado finito
  - Visão estrutural (FFs separados da lógica combinacional)
  - Visão comportamental
- Caminhos de dados = Computação
  - (ALUs, comparadores) + registradores
  - Utilização de operadores aritméticos de lógicos
  - Controle de elementos de armazenamento

Exemplo: Identifica sequência com pelo menos 2 1's

Moore



Mealy

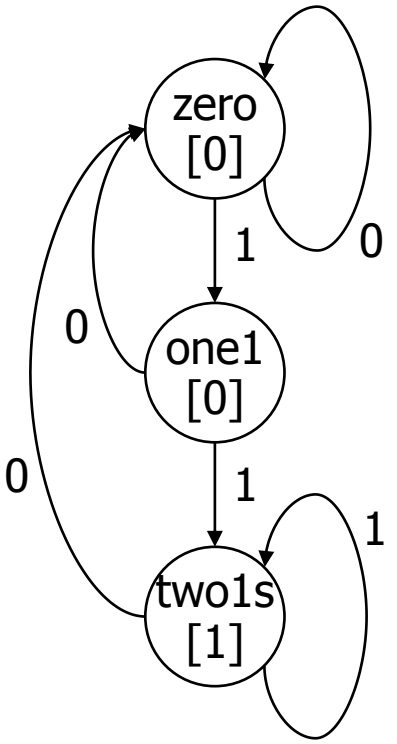


Exemplo: Identifica sequência com pelo menos 2 1's

- Código Verilog da máquina de Moore

```
module reduce (clk, reset, in, out);
  input clk, reset, in;
  output out;

  parameter zero  = 2'b00;
  parameter one1  = 2'b01;
  parameter two1s = 2'b10;

  reg out;
  reg [2:1] state;        // state variables
  reg [2:1] next_state;

  always @(posedge clk)
    if (reset) state = zero;
    else       state = next_state;
```

Assinalamento de estados



Código Verilog da máquina de Moore

```
always @(in or state)

  case (state)
    zero:
  // last input was a zero
   begin
     if (in) next_state = one1;
     else    next_state = zero;
   end
    one1:
  // we've seen one 1
   begin
     if (in) next_state = two1s;
     else    next_state = zero;
   end
    two1s:
  // we've seen at least 2 ones
   begin
     if (in) next_state = two1s;
     else    next_state = zero;
   end
  endcase
```

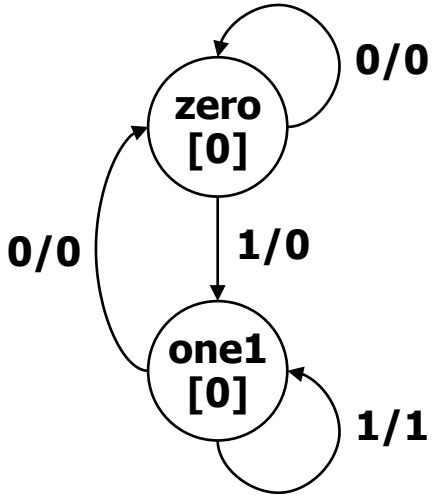
Todos os estados que são levados em consideração na determinação do estado devem ser incluídos

Observe que a saída depende apenas do estado

```
always @(state)
  case (state)
    zero: out = 0;
    one1: out = 0;
   two1s: out = 1;
  endcase

endmodule
```

Código Verilog da máquina de Mealy

```
module reduce (clk, reset, in, out);
  input clk, reset, in;
  output out;
  reg out;
  reg state; // state variables
  reg next_state;

  always @(posedge clk)
    if (reset) state = zero;
    else       state = next_state;

  always @(in or state)
    case (state)
      zero:              // last input was a zero
     begin
       out = 0;
       if (in) next_state = one;
       else    next_state = zero;
     end
      one:               // we've seen one 1
     if (in) begin
        next_state = one; out = 1;
     end else begin
        next_state = zero; out = 0;
     end
    endcase
endmodule
```



Máquina de Mealy síncrona

```
module reduce (clk, reset, in, out);
  input clk, reset, in;
  output out;
  reg out;
  reg state; // state variables

  always @(posedge clk)
    if (reset) state = zero;
    else
     case (state)
      zero:       // last input was a zero
     begin
       out = 0;
       if (in) state = one;
       else    state = zero;
     end
      one:        // we've seen one 1
     if (in) begin
         state = one; out = 1;
     end else begin
         state = zero; out = 0;
     end
    endcase
endmodule
```

Resumo

- Modelos de representação de circuitos sequenciais
  - Abstração de elementos sequenciais
  - Máquinas de estado finitos e seus diagramas de estados
  - Entradas/saídas
  - Máquinas de Mealy, Moore, Mealy síncrona
- Etapas de projeto de máquinas de estado finitos
  - Implementação do diagrama de estados
  - Implementação da tabela de transição de estados
  - Determinar funções de próximo estado e saída
  - Implementação da lógica combinacional
- Linguagens de descrição de hardware