SD132 – Circuitos Digitais II
A-206 Máquinas de Estado Mealy

Autores	
Gilmar Silva Beserra (UnB)	Daniel Muñoz Arboleda (UnB)
Nome 3 (INSTITUIÇÃO)	Nome 4 (INSTITUIÇÃO)

Histórico de revisões		
10/02/2025	V1.0	Versão inicial

Tópicos

- Moore x Mealy
- Projeto/Síntese e Análise de uma FSM (revisão)
- Máquinas de Estado Mealy
- Implementação em Verilog
- Implementação em VHDL

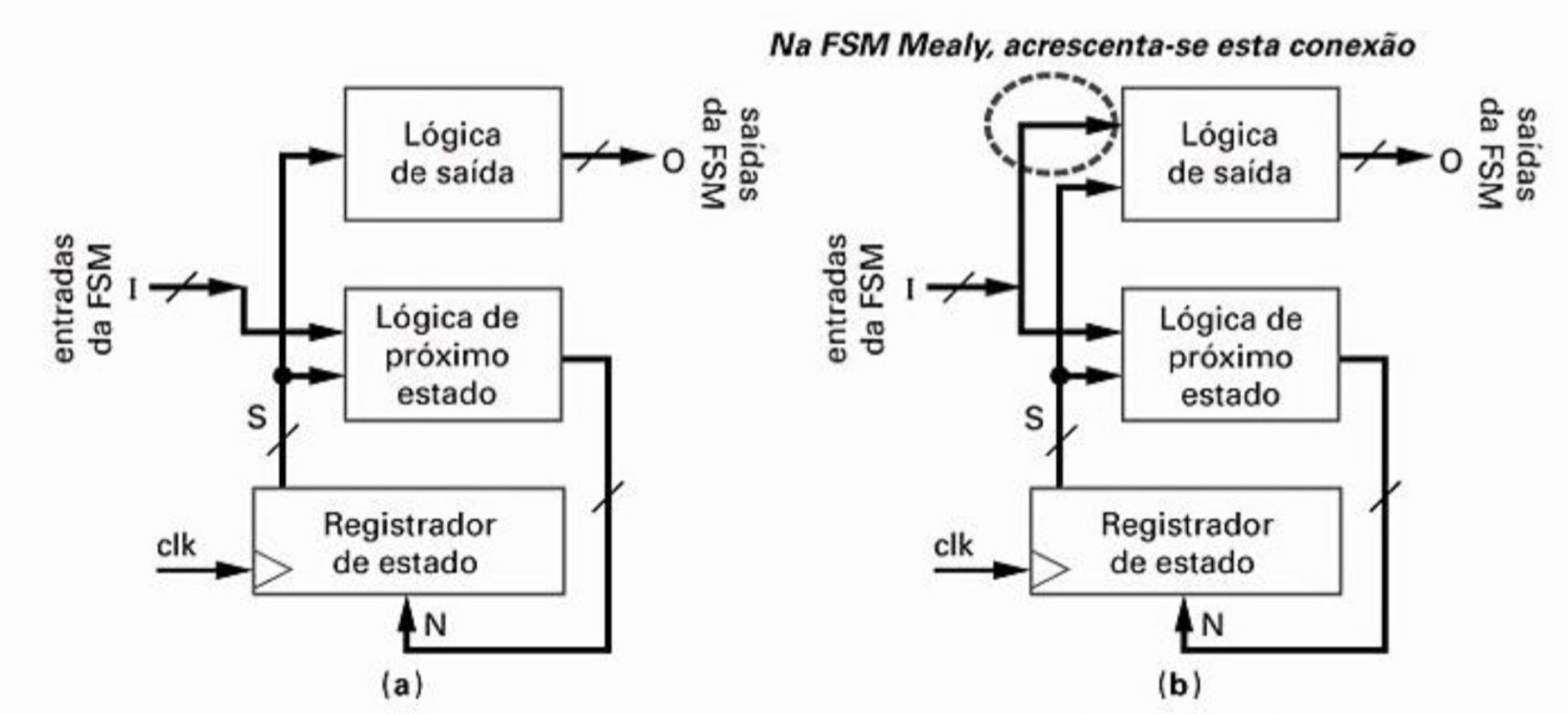
Página em branco.

Aula 18
Moore x Mealy

Objetivos

- Entender a diferença entre os modelos de Moore e Mealy
- Revisar os processos de projeto/síntese e análise de FSMs (Moore e Mealy)

Moore x Mealy

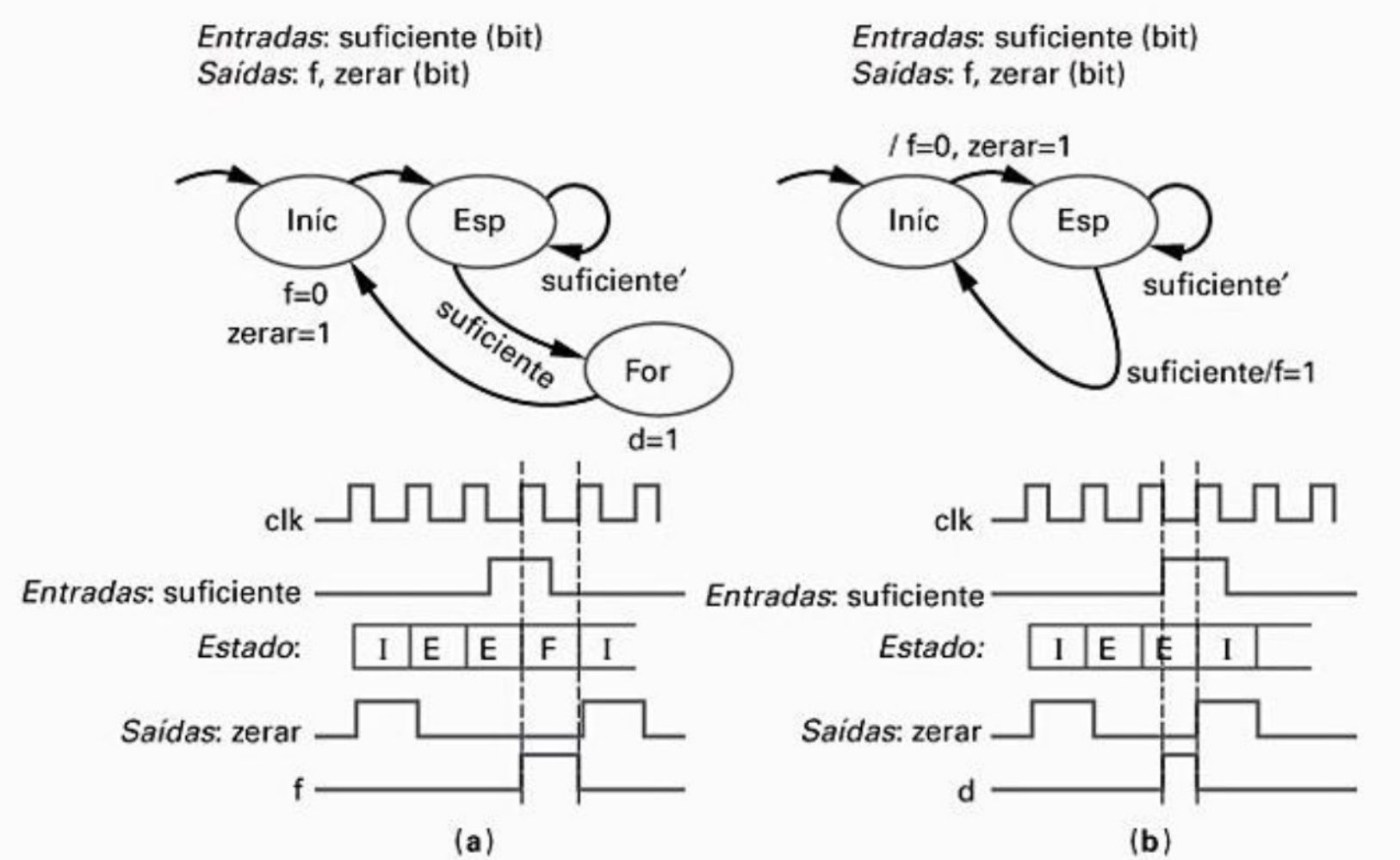


Exemplo

- **Máquina de refrigerante**
- **Entrada**: *suficiente* (1 bit)
- **Saídas**: *f* (1 bit), *zerar* (1 bit)

a) **Moore** (estados *Início, Esperar* e *Fornecer)*

b) **Mealy** (estados *Início* e *Esperar)*



Exemplo

- **Moore**
  - Estado **Iníc**: f = 0 e zerar = 1 (zera um dispositivo usado para contar o valor depositado em moedas)
  - Estado **Esp**: FSM aguarda a entrada suficiente, que informa se o valor depositado em moedas é igual ou maior ao valor do refrigerante
  - Estado **For**: saída d = 1, indicando que um refrigerante foi fornecido
  - Saídas sincronizadas com as bordas de relógio
- **Mealy**
  - Não precisa do estado **For** para fazer f = 1
  - Redução de estados
  - Saídas não estão sincronizadas com as bordas de relógio: podem ser modificadas quando uma entrada é alterada
  - Possível problema: se as entradas apresentarem glitches, o mesmo pode ocorrer com as saídas

Projeto/Síntese de FSMs

- **Descrição funcional detalhada**
  - Anotar as especificações do sistema (geralmente são variáveis e dados numéricos)
  - Quanto mais detalhado, melhor
  - Geralmente é um processo iterativo para aprimorar o conhecimento do problema, mas deve ser concluído antes de iniciar o desenvolvimento
  - Ao final da descrição é possível prever quais vão ser os estados necessários, o que faz cada estado, quantos bits (quantos flip-flops) serão necessários para registrar o estado e qual a saída em cada estado

Projeto/Síntese de FSMs

- **Diagrama de Estados**
  - Adotar um modelo (**Mealy** ou **Moore**)
  - Separar em estados cada etapa/tarefa/função do sistema, processo ou algoritmo
  - Indicar o estado inicial
  - Indicar condições de transição de entradas
  - Indicar saída(s) em cada transição ou estado
  - Tomar cuidado com condições de transição mal formuladas:
    - A partir de cada estado, uma e apenas uma transição deve ser válida
    - A partir de cada estado, deve ter pelo menos uma transição válida
    - A verificação dessas condições será objeto de estudo futuramente

Projeto/Síntese de FSMs

- **Tabela de Transição de Estados e Saídas**
  - Transformar o diagrama de estados em uma tabela de transição
  - Na coluna esquerda, separar por linhas o estado atual
  - Para cada combinação das entradas, colocar próximo estado
  - Dependendo se o modelo é de Mealy ou Moore, colocar a(s) saída (s) dependendo da combinação de entradas ou apenas do estado atual

Projeto/Síntese de FSMs

- **Tabela de Transição Reduzida**
  - Na tabela de transição, verificar se tem linhas redundantes
  - Quando há duas linhas iguais, ou seja, a partir do mesmo estado atual temos o mesmo próximo estado E a(s) mesma(s) saída(s), então temos estados redundantes
  - Eliminar as linhas que correspondem ao(s) estado(s) redundante(s)

Projeto/Síntese de FSMs

- **Codificação de Estados**
  - Definir número de bits para codificar o estado
  - Se número de estados = $\mathbf{2^n}$ então serão necessários **n** flip-flops
  - Se número de estados < $\mathbf{2^n}$ então serão necessários **n** flip-flops, porém existirão estados não codificados, com comportamento não definido
  - Adotar um código binário (binario, Gray, Johnson, BCD, etc.)
  - Codificar cada estado

Projeto/Síntese de FSMs

- **Circuito Combinacional para Transição de Estados**
  - Para cada bit do estado, obter um mapa de Karnaugh (mapa K)
  - Associar elementos adjacentes do mapa K no intuito de obter as equações booleanas de transição de estado
  - Pode-se usar soma de produtos (associação de **‘1’s**) ou produtos de somas (associação de **‘0’s**)
  - **Dicas**: adotar a associação de **‘1’s**, a não ser que o problema solicite o contrário, e maximizar as associações (por exemplo, é melhor associar uma quadra do que usar duas duplas, pois dessa forma o circuito combinacional terá menos portas lógicas)

Projeto/Síntese de FSMs

- **Circuito Combinacional para Saídas**
  - Para cada bit da saída, obter um mapa de Karnaugh (mapa K)
  - Associar elementos adjacentes do mapa K no intuito de obter as equações booleanas das saídas
  - Pode-se usar soma de produtos (associação de **‘1’s**) ou produtos de somas (associação de **‘0’s**)
  - **Dicas**: adotar a associação de **‘1’s**, a não ser que o problema solicite o contrário, e maximizar as associações (por exemplo, é melhor associar uma quadra do que usar duas duplas, pois dessa forma o circuito combinacional terá menos portas lógicas)

Projeto/Síntese de FSMs

- **Diagrama Lógico do Circuito**
  - A partir das equações booleanas de transição de estado e saída, obter um circuito equivalente
  - Iniciar desenhando os flip-flops. Cada bit de estado precisa de um flip-flop
  - A entrada do flip-flop é a saída de uma equação booleana (um bit), ou seja, o próximo estado
  - A saída do flip-flop é o bit de estado registrado, ou seja, o estado atual
  - Usar portas AND, OR, NAND, NOR ou XOR, dependendo da equação booleana

Análise de FSMs

- **Procedimento**:
  1) Analisar o diagrama lógico do circuito
  2) Obter as equações booleanas
  3) Obter a tabela de transição de estados e saídas
  4) Codificar estados
  5) Obter arquitetura padrão
  6) Obter diagrama de estados

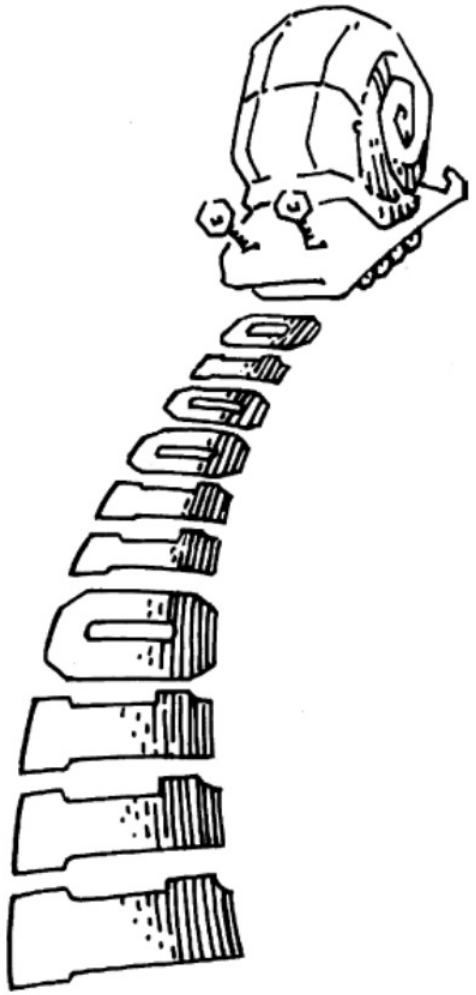
  Finalmente, interpretar funcionamento!

Página em branco.

Aula 19
Máquinas de Estado Mealy

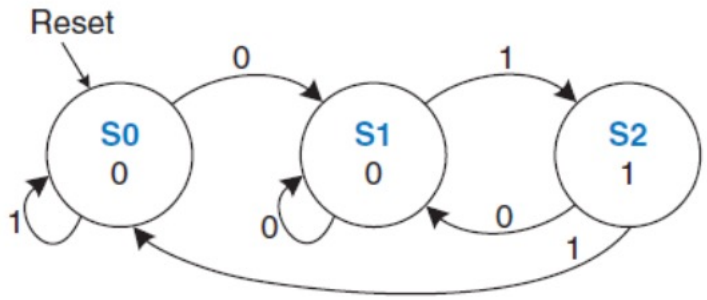
Exemplo de Projeto/Síntese

- Descrição detalhada:
  - Caracol robótico com cérebro FSM rastejando a partir da esquerda para a direita ao longo de uma fita de papel contendo uma sequência de 1’s e 0’s. Em cada ciclo de clock, o caracol rasteja para o p’roximo bit. Quando os dois últimos dígitos são 01, o caracol sorri.
  - Considerando que a entrada A é a parte de baixo das antenas do caracol que lê os dígitos e a saída Y = 1 quando o caracol sorri, projetar uma FSM (Moore e Mealy) para calcular quando o caracol deve sorrir.



Exemplo de Projeto/Síntese

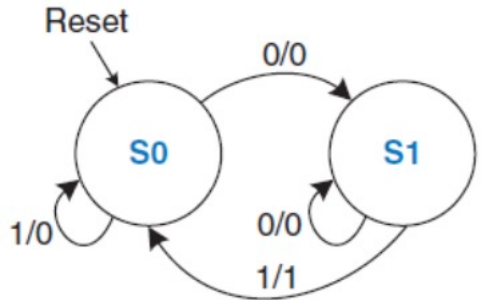
- Moore



Current State S	Input A	Next State S'
S0	0	S1
S0	1	S0
S1	0	S1
S1	1	S2
S2	0	S1
S2	1	S0

Current State S	Output Y
S0	0
S1	0
S2	1

- Mealy



Current State S	Input A	Next State S'	Output Y
S0	0	S1	0
S0	1	S0	0
S1	0	S1	0
S1	1	S0	1

Exemplo de Projeto/Síntese

- Moore
  - S0 = 00, S1 = 01, S2 = 10
  - Estado 11 não existe:
    - Risco mínimo: próximo estado e saídas iguais a zero
    - Custo mínimo: próximo estado e saídas iguais a *don't care*
- Mealy
  - S0 = 0
  - S1 = 1

Current State S_1	Current State S_0	Input A	Next State S'_1	Next State S'_0
0	0	0	0	1
0	0	1	0	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0

Current State S_1	Current State S_0	Output Y
0	0	0
0	1	0
1	0	1

Current State S_0	Input A	Next State S'_0	Output Y
0	0	1	0
0	1	0	0
1	0	1	0
1	1	0	1

Exemplo de Projeto/Síntese

- Moore
  - Equações com FF D

$$S_1' = S_0 A$$
$$S_0' = \bar{A}$$
$$Y = S_1$$

Current State S_1	Current State S_0	Input A	Next State S_1'	Next State S_0'
0	0	0	0	1
0	0	1	0	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0

Current State S_1	Current State S_0	Output Y
0	0	0
0	1	0
1	0	1

- Mealy
  - Equações com FF D

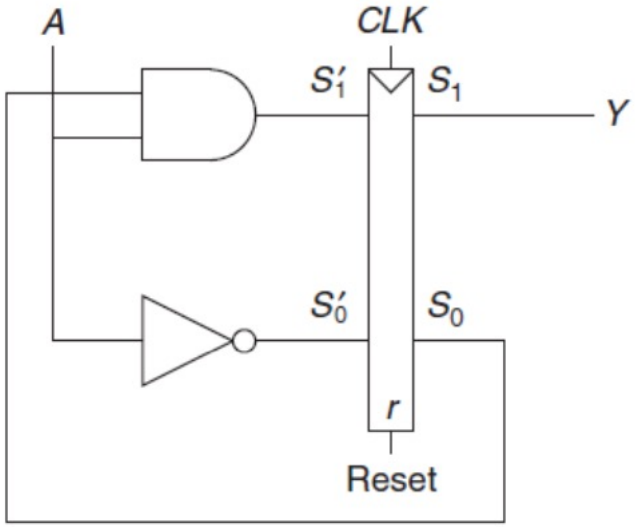
$$S_0' = \bar{A}$$
$$Y = S_0 A$$

Current State S_0	Input A	Next State S_0'	Output Y
0	0	1	0
0	1	0	0
1	0	1	0
1	1	0	1

Exemplo de Projeto/Síntese

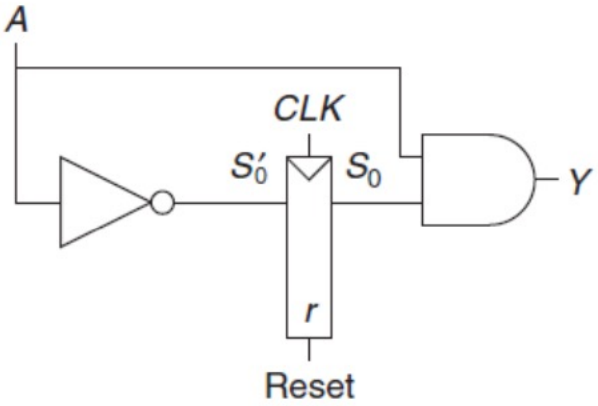
- Moore
  - Equações com FF D

$$S_1' = S_0 A$$
$$S_0' = \bar{A}$$
$$Y = S_1$$



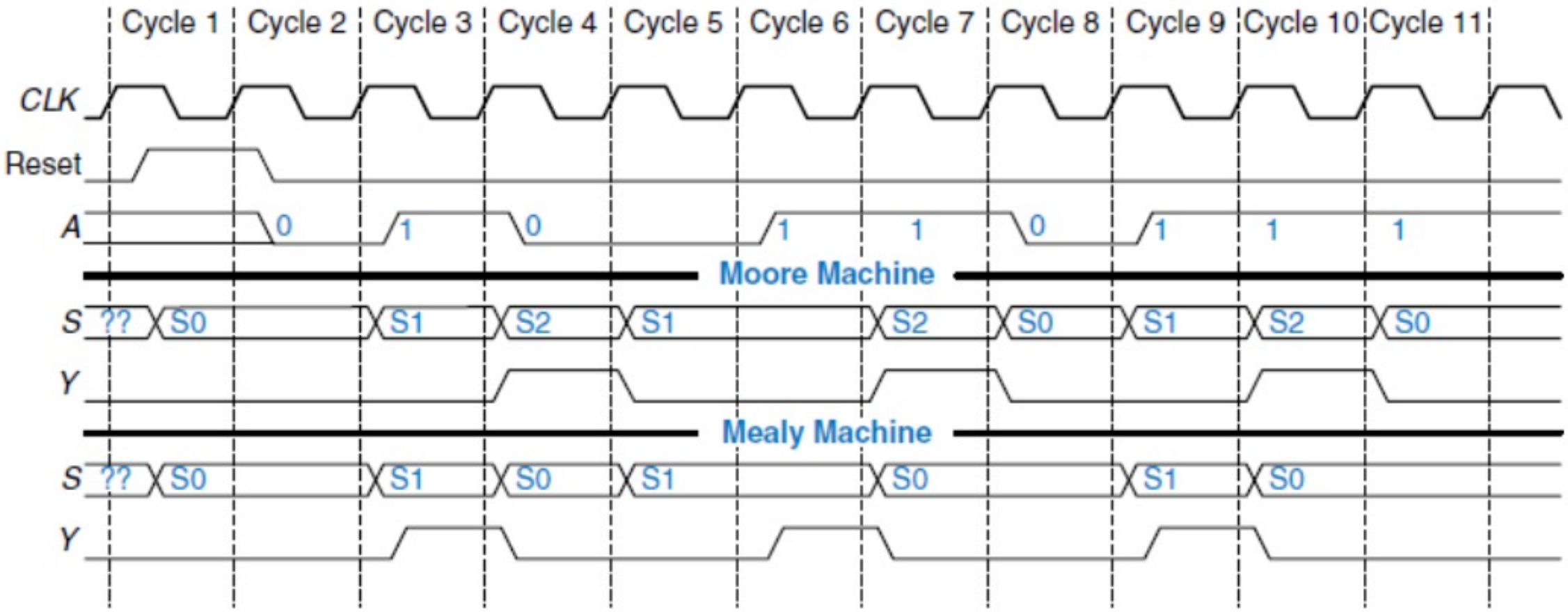
- Mealy
  - Equações com FF D

$$S_0' = \bar{A}$$
$$Y = S_0 A$$



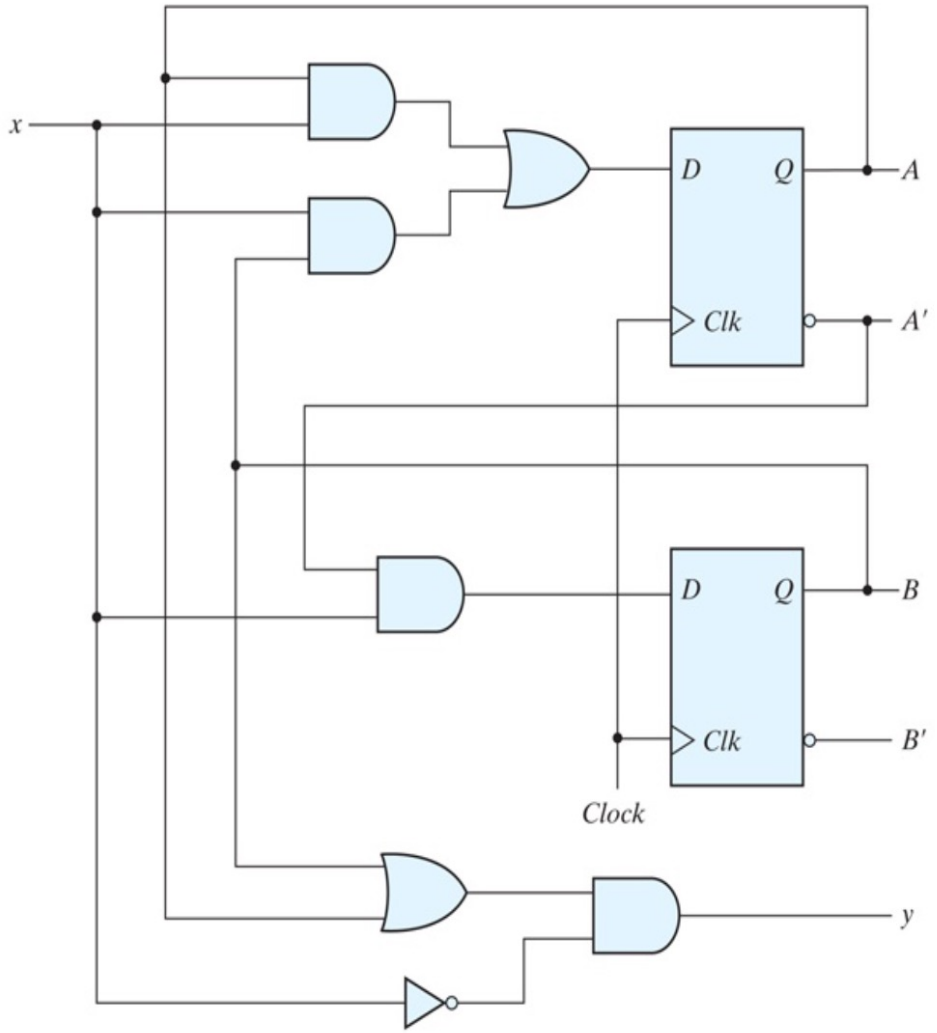
Exemplo de Projeto/Síntese

- Diagrama de tempo



Exemplo de Análise

- Analisar o diagrama do circuito ao lado
- Identificar
  - O registrador de estados
  - A lógica combinacional de transição de estados
  - A lógica combinacional para a saída



Exemplo de Análise

- Obter as equações booleanas a partir dos circuitos combinacionais e a tabela de transição e de saída

Próximo estado:

D1 = x.Q1 + x.Q0

D0 = x.Q1'

Saída:

y = (Q1 + Q0).x'

Present State		Next State				Output	
		$x = 0$		$x = 1$		$x = 0$	$x = 1$
Q1	Q0	D1	D0	D1	D0	y	y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

Exemplo de Análise

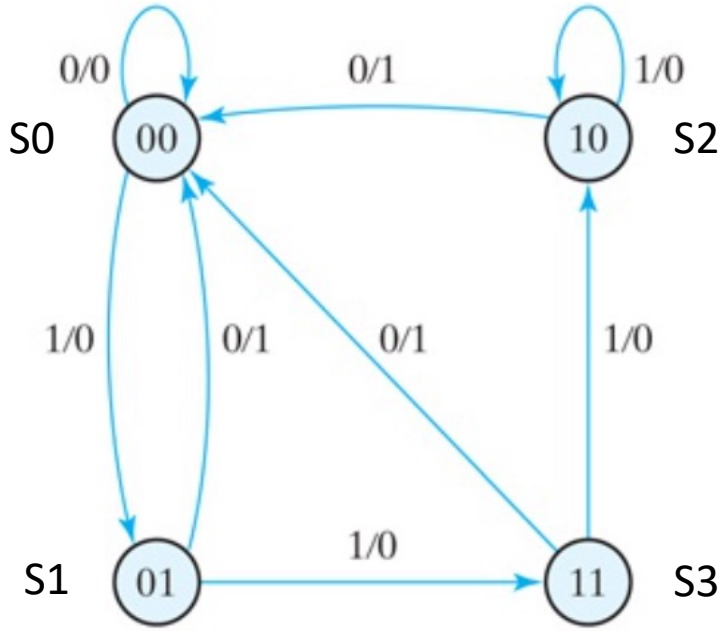
- Codificar os estados, usando nomes simbólicos para os estados enumerados
- Obter diagrama de estados

S0 = 00

S1 = 01

S2 = 10

S3 = 11

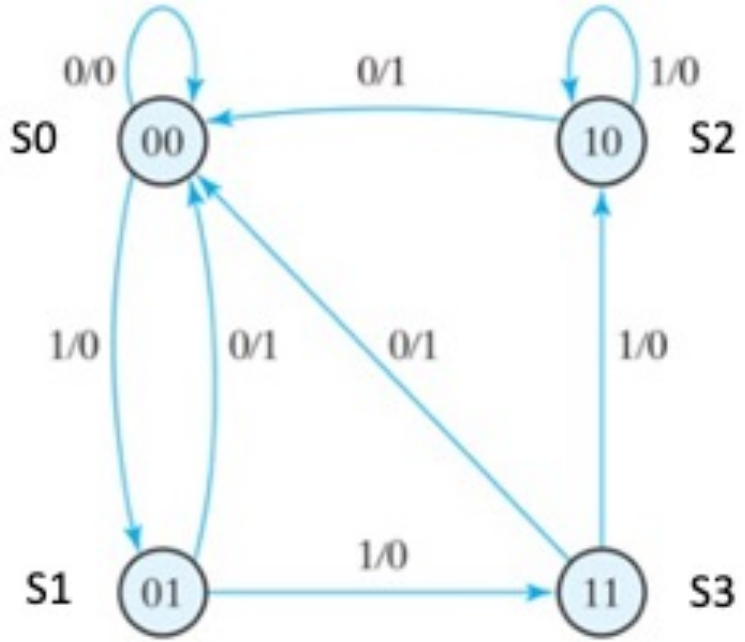


Página em branco.

Aula 20
Implementação de Máquinas de Estados Mealy em HDL

Exemplo (Verilog)

- Funcionalidade pode ser baseada no diagrama de estados.
- Exemplo: detector de zero



Exemplo (Verilog)

- Entradas: *x_in, clock, reset.* Saída: *y_out*

```
//Verilog 2001, 2005 syntax
module Mealy_Zero_Detector (
output reg y_out,
input x_in, clock, reset
);
reg [1: 0] state, next_state;
parameter S0 = 2'b00, S1 = 2'b01,
S2 = 2'b10, S3 = 2'b11;
```

- Estados:

```
reg [1: 0] state, next_state;
parameter S0 = 2'b00, S1 = 2'b01,
S2 = 2'b10, S3 = 2'b11;
```

Exemplo (Verilog)

- Processos de estado atual e próximo estado

```
always @ ( posedge clock, negedge reset)
    if (reset == 0) state <= S0;
    else state <= next_state;

always @ (state, x_in) // Next state
    case (state)
    S0: if (x_in) next_state = S1; else next_state = S0;
    S1: if (x_in) next_state = S3; else next_state = S0;
    S2: if (~x_in) next_state = S0; else next_state = S2;
    S3: if (x_in) next_state = S2; else next_state = S0;
    endcase
```

Exemplo (Verilog)

- Processo de saída

```
always @ (state, x_in) // Mealy output
case (state)
S0: y_out = 0;
S1, S2, S3: y_out = ~x_in;
endcase

endmodule
```

Exemplo (Verilog)

- *Testbench*

```
module t_Mealy_Zero_Detector;
 wire   t_y_out;
 reg    t_x_in, t_clock, t_reset;
Mealy_Zero_Detector M0 (t_y_out, t_x_in, t_clock, t_reset);
initial #200 $finish;
initial begin t_clock = 0; forever #5 t_clock = ~t_clock; end
initial fork
   t_reset = 0;
 #2 t_reset = 1;
 #87 t_reset = 0;
 #89 t_reset = 1;
 #10 t_x_in = 1;
 #30 t_x_in = 0;
 #40 t_x_in = 1;
 #50 t_x_in = 0;
 #52 t_x_in = 1;
 #54 t_x_in = 0;
 #80 t_x_in = 1;
 #100 t_x_in = 0;
 #120 t_x_in = 1;
 join
endmodule
```

Exemplo (VHDL)

- Entradas: *x_in, clock, reset.* Saída: *y_out*

```
entity Mealy_Zero_Detector_vhdl is
 port (y_out: out std_logic; x_in, clock, reset: in std_logic);
end Mealy_Zero_Detector_vhdl;
```

- Estados:

```
architecture Behavioral of Mealy_Zero_Detector_vhdl is
 type state_type (S0, S1, S2, S3);  -- machine states
 signal state, next_state : state_type;
```

- Processo de estado atual:

```
process (clock, reset) begin  -- Synchronous state transitions
  if (reset'event and reset = '0' then state <= S0;
  else if clock'event and clock = '1' then state <= next_state
 end if;
end process;
```

Exemplo (VHDL)

- Processos de próximo estado:

```
process (state, x_in) begin  -- Next state
 case (state) is
  when  S0 =>   if x_in = '1' then next_state = S1; else next_s
                else end if;
  when  S1 =>   if x_in = '1' then next_state = S3; else next_s
                else end if;
    when        S2 =>   if x_in = '0' then next_state = S0; els
                else end if;
  when  S3 =>   if x_in = '1' then next_state = S2; else next_s
                else end if;
  when others => next_state <= S0;
 end case;
end process;
```

Exemplo (VHDL)

- Processos de saída:

```
 process (state, x_in) begin  -- Output
  case (state) is
   when S0 => y_out = '0';
   when  S1 => y_out = not x_in;
   when  S2 => y_out = not x_in;
   when  S3 => y_out = not x_in;
  end case;
 end process;
end Behavioral;
```

Exemplo (VHDL)

- *Testbench*

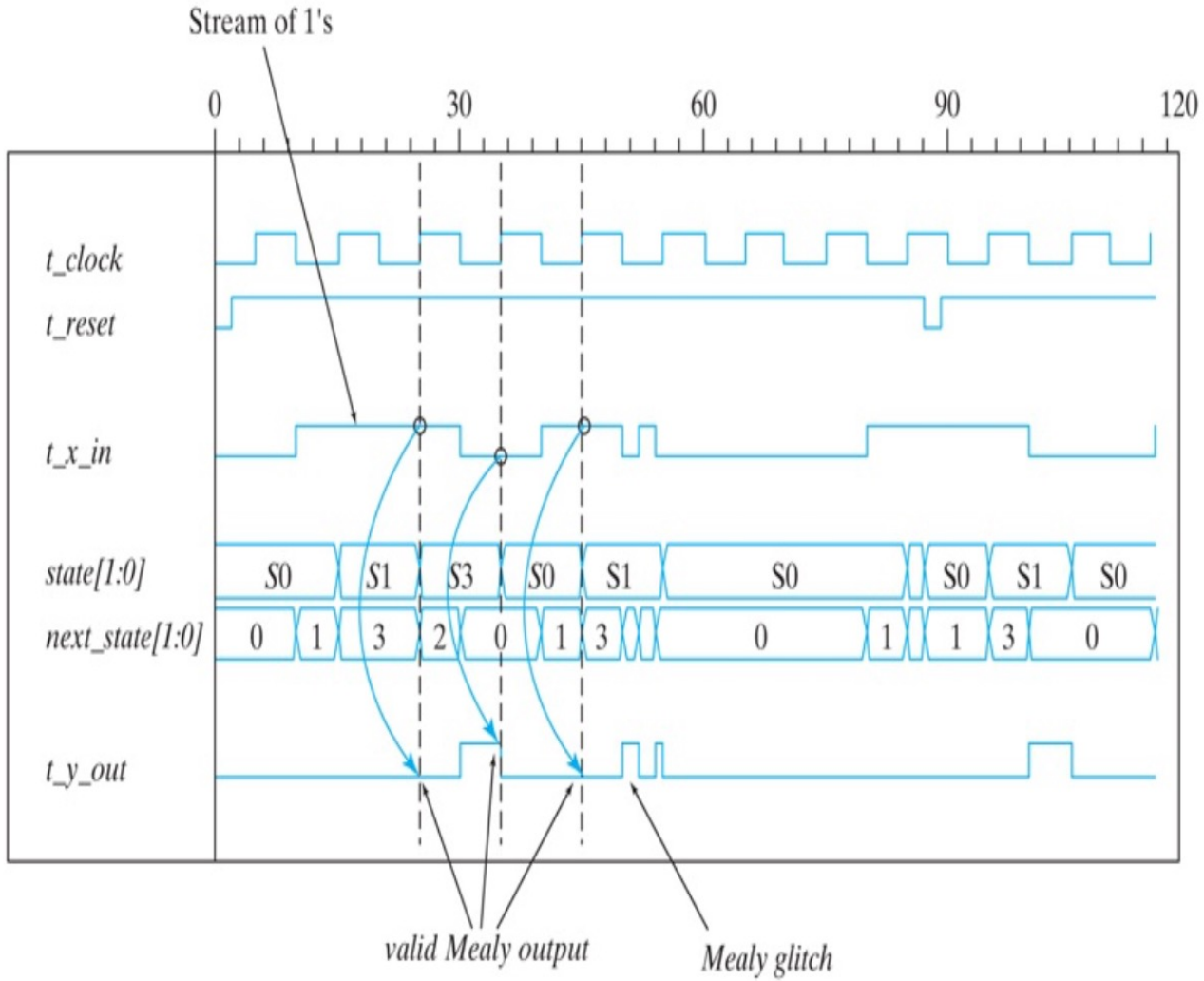
```
entity t_Meally_Zero_Detector_vhdl is
end Mealy_Zero_Detector_vhdl;

architecture Behavioral of t_Mealy_Zero_Detector_vhdl is
 signal t_y_out: std_logic;
 signal t_x_in: std_logic;
begin
  -- Instantiate the UUT
  UUT: Mealy_Zero_Detector_vhdl port map (y_out => t_y_out, x_i
 -- Create free-running clock signal;
process (clock) begin
 clock <= not clock after 5 ns;
end process;
```

Exemplo (VHDL)

- *Testbench*

```
-- Specify stimulus signal signals
 process begin
  t_reset   <= '0';
  t_reset   <= '1' after 2 ns;
  t_reset   <= '0' after 87 ns;
  t_reset   <= '1' after 89 ns;
  t_x_in <= '1' after 10 ns;
  t_x_in <= '0' after 30 ns;
  t_x_in <= '1' after 40 ns;
  t_x_in <= '0' after 50 ns;
  t_x_in <= '1' after 52ns;
  t_x_in <= '0' after 54 ns;
  t_x_in <= '1' after 70 ns;
    t_x_in <= '0' after 80 ns;
  t_x_in <= '1' after 90 ns;
  t_x_in <= '0' after 100 ns;
  t_x_in <= '1' after 120 ns;
  t_x_in <= '0' after 160 ns;
  t_x_in <= '1' after 170 ns;
 end process ;
end Behavioral;
```

Exemplo - Simulação

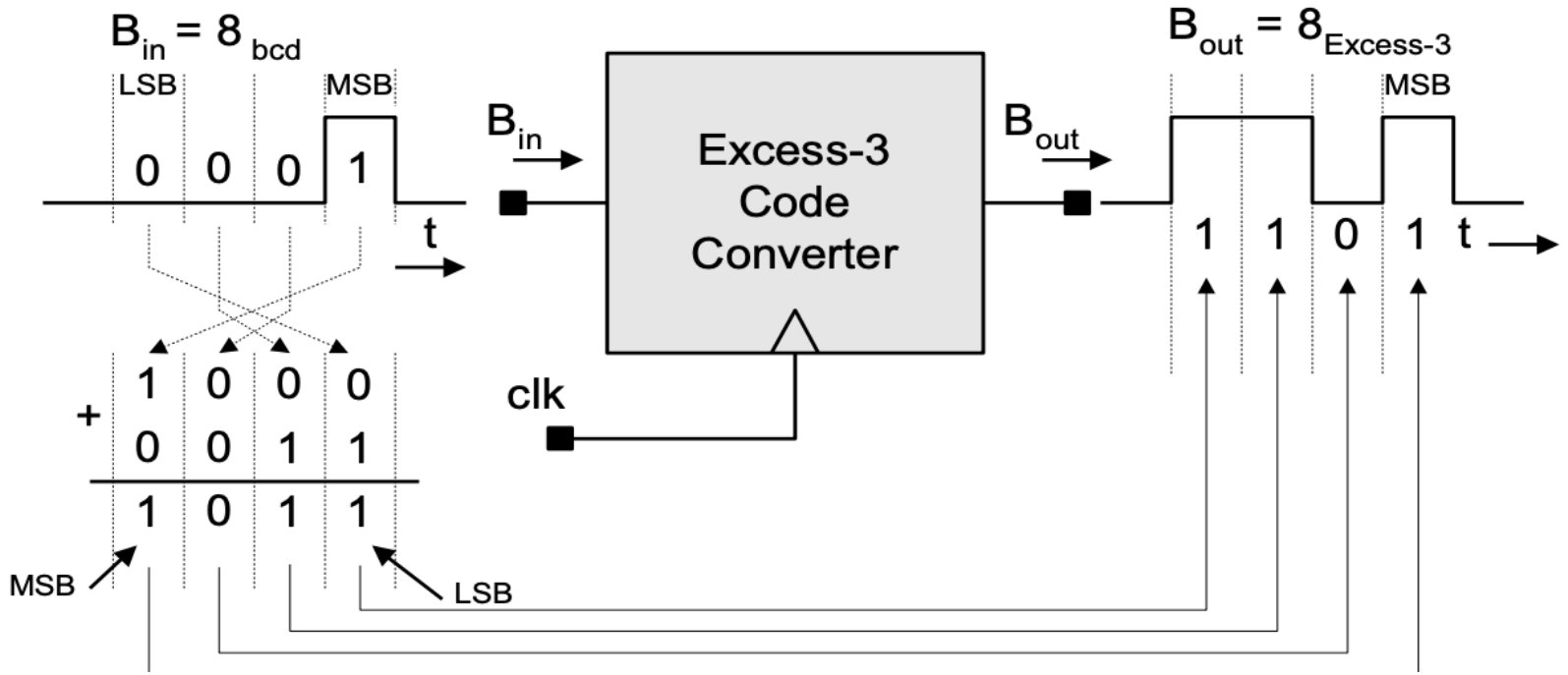


Página em branco.

Aula 21
Exercícios

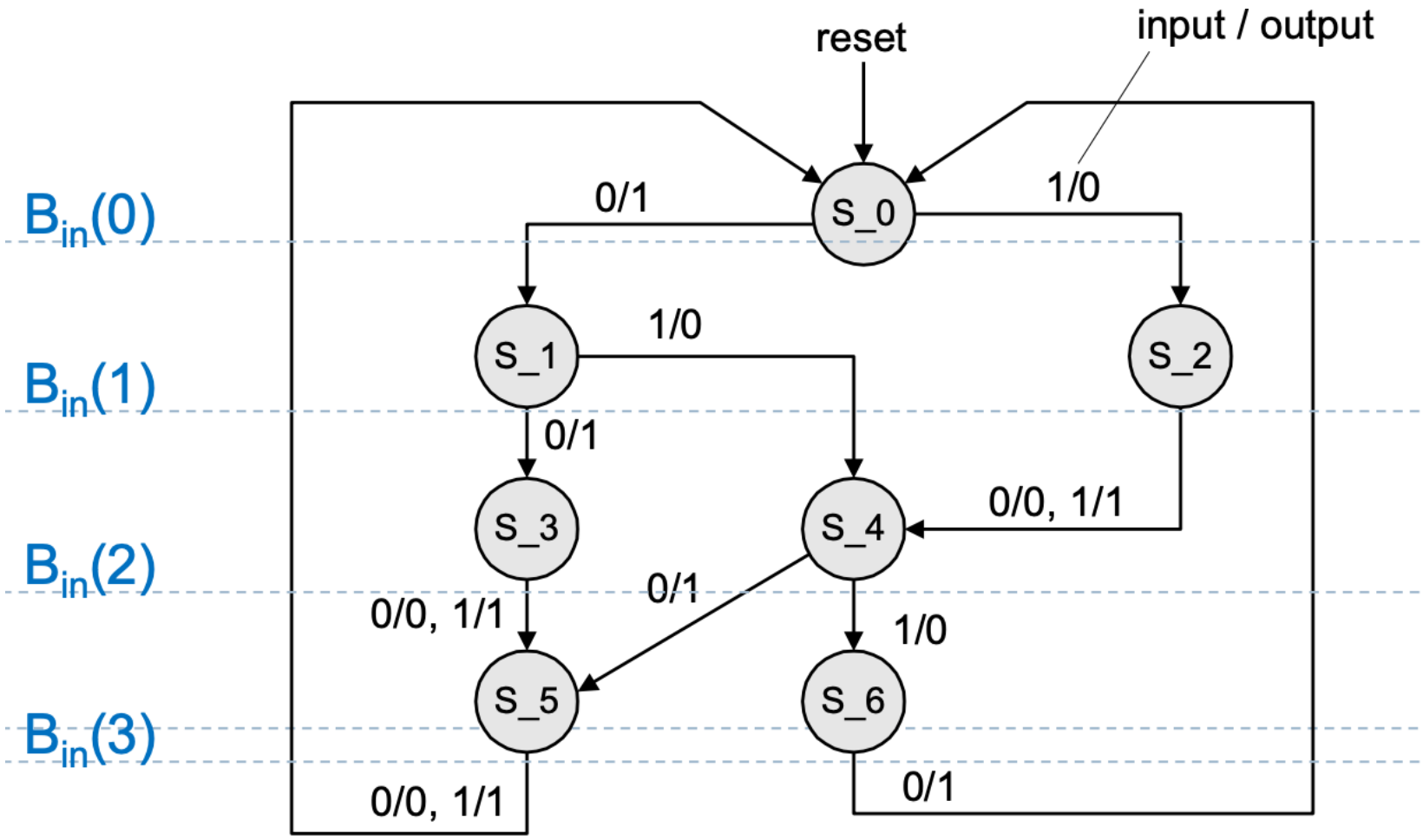
Exercício 1

- Implementar em HDL um conversor de código de BCD para Excesso-3. A entrada B_{in} é transmitida em sequência, iniciando com o LSB. A saída B_{out} deve mostrar o número em Excesso-3.



Exercício 1

- Diagrama de estados:



Exercício 2

- Alterar o exemplo do caracol robótico para que o cérebro do mesmo seja uma FSM de Mealy cuja saída y = 1 quando o caracol deslizar sobre o padrão 1101 ou 1110. Esboçar o diagrama de estados e implementar a FSM em HDL usando o menor número possível de estados.

Exercício 3

- Projetar e implementar um distribuidor de máquina de refrigerante na qual cada item custa apenas 25 centavos. A máquina aceita moedas de 5 centavos, 10 centavos e 25 centavos. Supor que uma moeda é inserida em cada ciclo. As saídas permitem entregar um refrigerante (quando as moedas inseridas chegam a 25 centavos) e devolver trocos de 5 centavos, 10 centavos e 20 centavos. Após fornecer um refrigerante e trocos (se necessário), a FSM deve retornar ao estado inicial.

Página em branco.