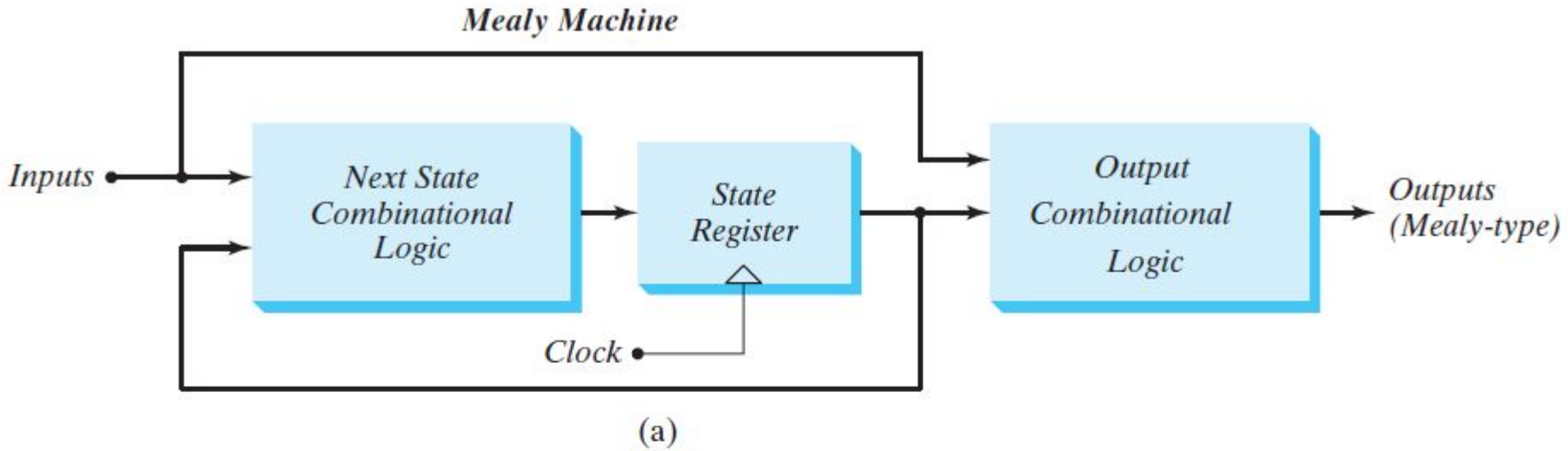
SISTEMAS SECUENCIALES

Técnicas Digitales I

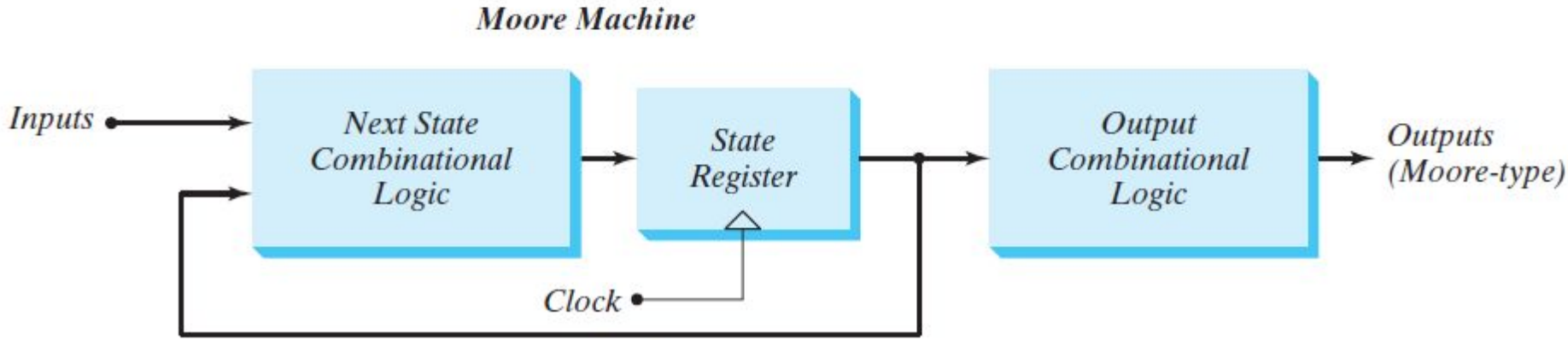
Luis Eduardo Toledo

UTN

MÁQUINA DE ESTADOS: MEALY Y MOORE

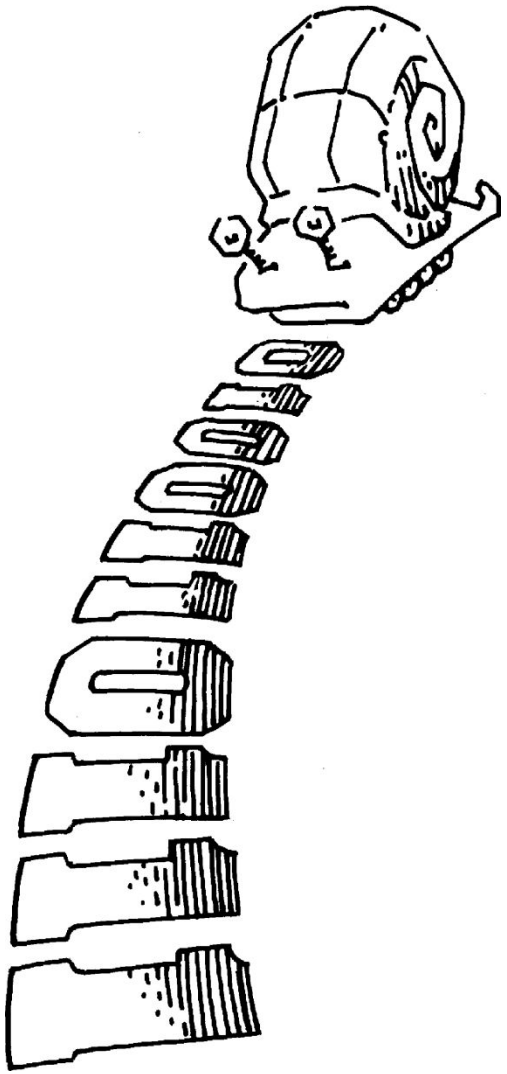


(a)



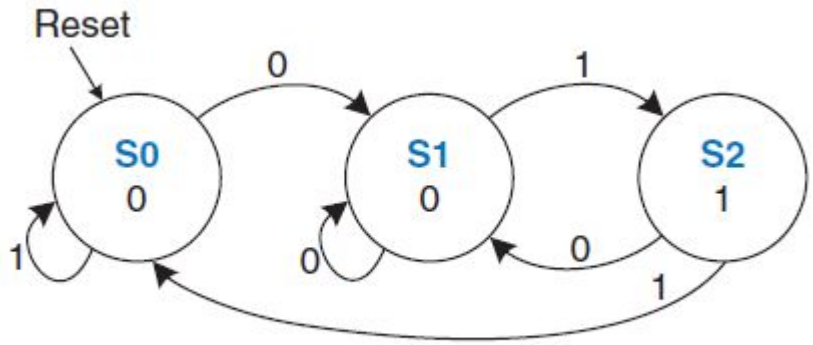
(b)

EJEMPLO COMPARATIVO MOORE-MEALY



Alyssa P. Hacker posee un caracol robotizado de mascota con un cerebro diseñado con una FSM. El caracol se arrastra de izquierda a derecha a lo largo de una cinta de papel que contiene una secuencia de 1s y 0s. En cada ciclo de reloj, el caracol se arrastra hasta el siguiente bit. El caracol sonríe cuando los dos últimos bits sobre los que se ha arrastrado son, de izquierda a derecha, 01. Diseñar la FSM para calcular cuando el caracol debe sonreír. La entrada A es el bit debajo de las antenas del caracol. La salida Y es VERDADERA cuando el caracol sonríe. Comparar los diseños de máquinas de estado de Moore y Mealy. Dibuje un diagrama de tiempo para cada máquina de estado que muestre la entrada, los estados y la salida cuando el caracol de Alyssa se arrastra a lo largo de la secuencia 0100110111.

MÁQUINA DE MOORE



(a)

Table 3.11 Moore state transition table

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

Table 3.12 Moore output table

Current State S	Output Y
S0	0
S1	0
S2	1

MÁQUINA DE MOORE

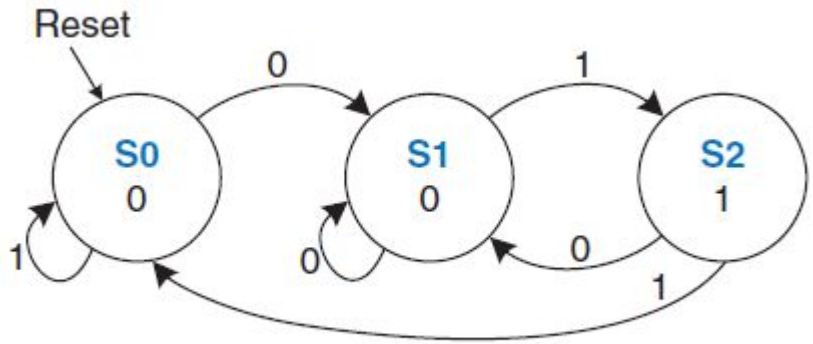


Table 3.13 Moore state transition table with state encodings

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

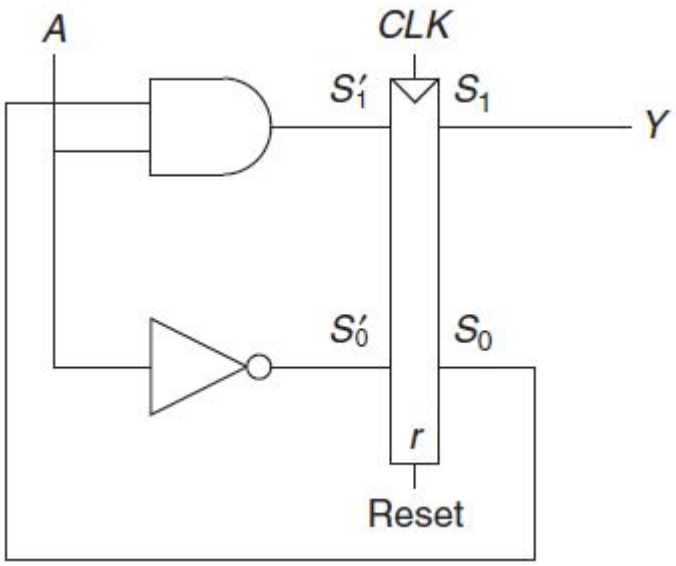
$$S_1' = S_0 A$$
$$S_0' = \overline{A}$$

Table 3.14 Moore output table with state encodings

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

$$Y = S_1$$

MÁQUINA DE MOORE: IMPLEMENTACIÓN



$$S_1' = S_0 A$$
$$S_0' = \overline{A}$$

$$Y = S_1$$

MÁQUINA DE MEALY

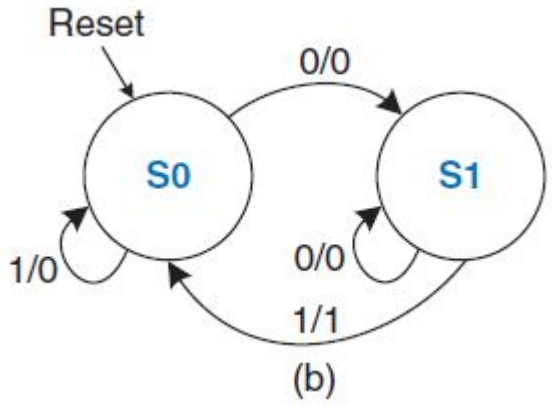


Table 3.15 Mealy state transition and output table

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

MÁQUINA DE MEALY

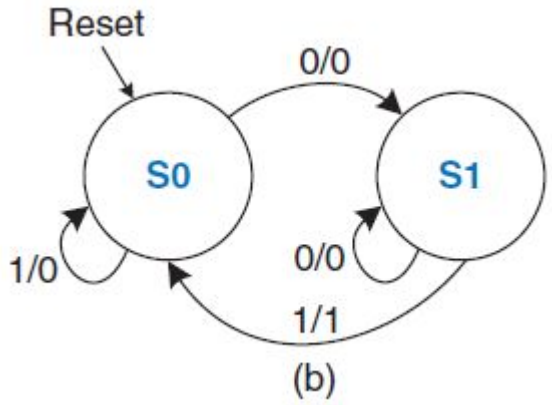


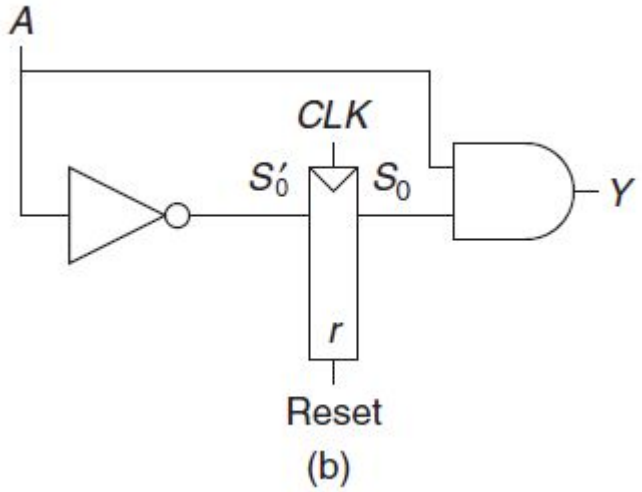
Table 3.16 Mealy state transition and output table with state encodings

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

$$S_0' = \overline{A}$$

$$Y = S_0 A$$

MÁQUINA DE MEALY: IMPLEMENTACIÓN

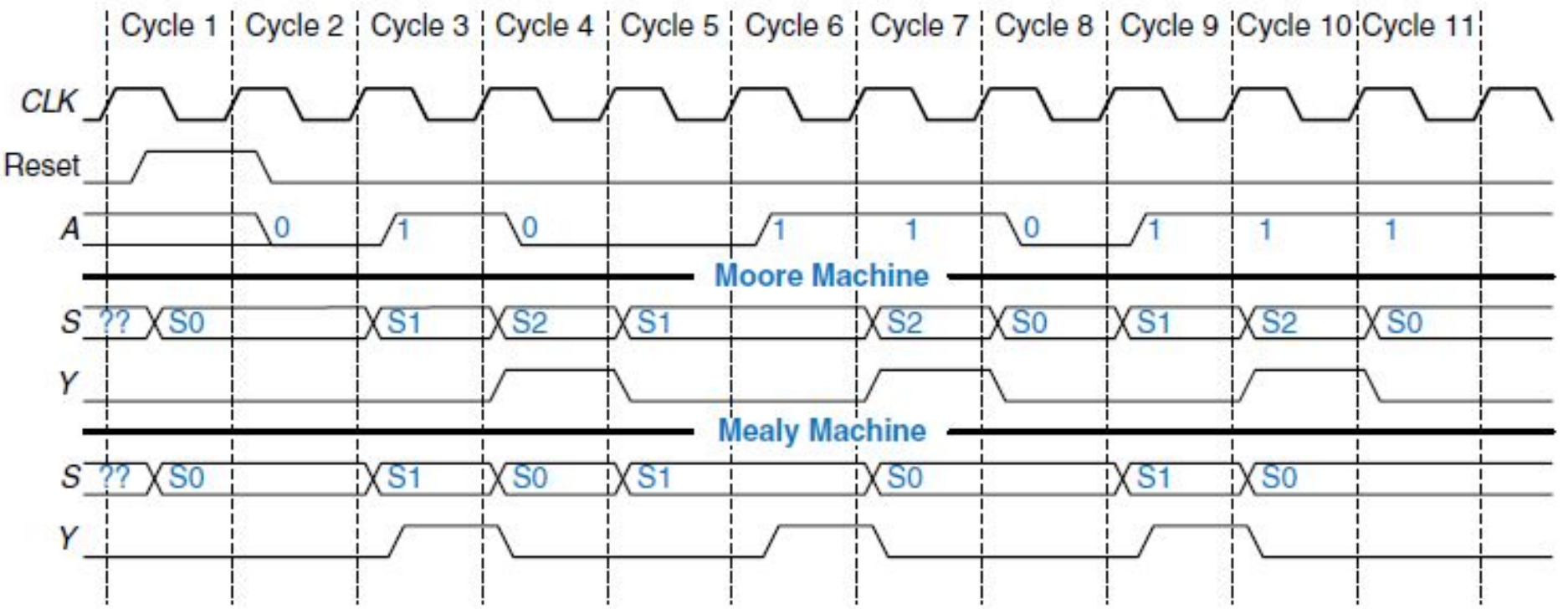


(b)

$$S_0' = \overline{A}$$

$$Y = S_0 A$$

DIAGRAMAS MÁQUINAS DE MEALY Y MOORE



HDL PARA MÁQUINAS DE MEALY Y MOORE

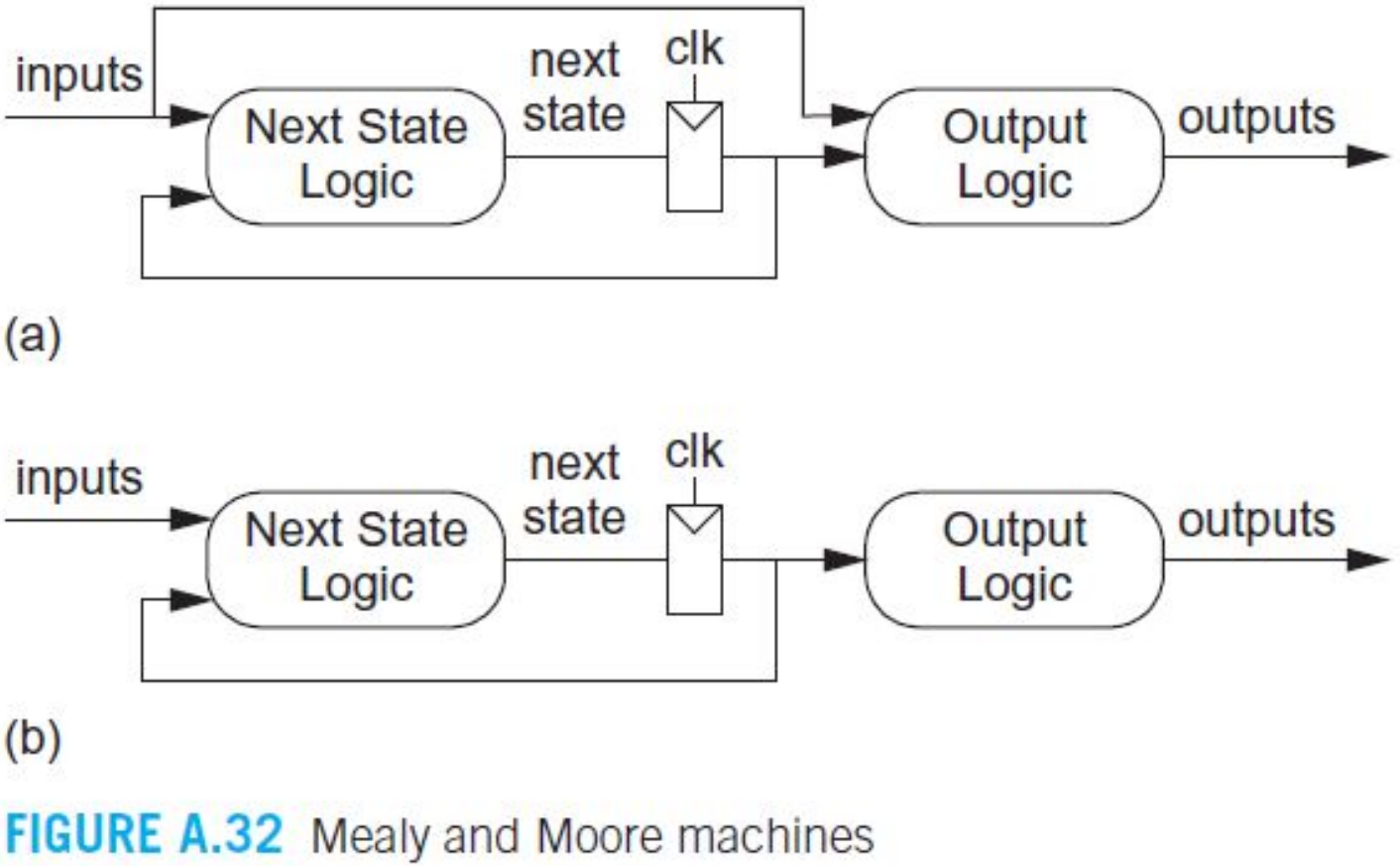
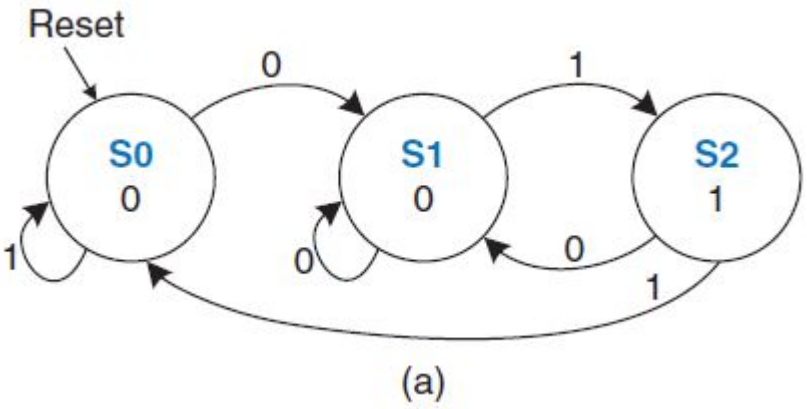


FIGURE A.32 Mealy and Moore machines

HDL PARA MÁQUINA DE MOORE



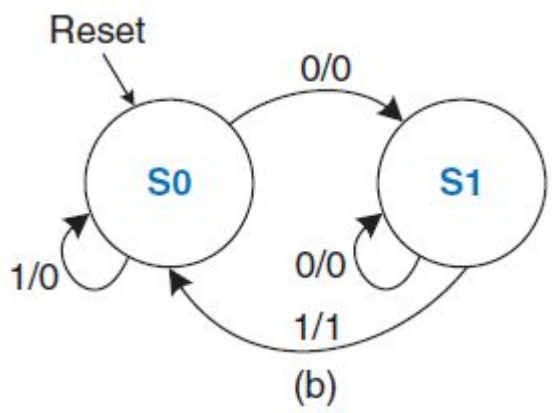

```
module patternMoore ( input clk,
                      input reset,
                      input a,
                      output y);

  reg [1:0] state,  nextstate;

  parameter S0 = 2'b00;
  parameter S1 = 2'b01;
  parameter S2 = 2'b10;
```

```
// state register
  always @ (posedge clk, posedge reset)
    if (reset) state <= S0;
    else       state <= nextstate;
  // next state logic
  always @ (*)
    case (state)
      S0: if (a) nextstate = S0;
          else   nextstate = S1;
      S1: if (a) nextstate = S2;
          else   nextstate = S1;
      S2: if (a) nextstate = S0;
          else   nextstate = S1;
      default:  nextstate = S0;
    endcase
 // output logic
  assign y = (state==S2);
endmodule
```

HDL PARA MÁQUINA DE MEALY




```
module patternMealy ( input clk,
                      input reset,
                      input a,
                      output y);
  reg  state, nextstate;
  parameter S0 = 1'b0;
  parameter S1 = 1'b1;
```

```
  // state register
  always @ (posedge clk, posedge reset)
    if (reset) state <= S0;
    else       state <= nextstate;

  // next state logic
  always @(*)
    case (state)
      S0: if (a) nextstate = S0;
          else   nextstate = S1;
      S1: if (a) nextstate = S0;
          else  nextstate = S1;
      default:  nextstate = S0;
    endcase

  // output logic
    assign y = (a & state==S1);
endmodule
```

HDL PARA MÁQUINA DE MOORE

```
module divideby3FSM(input  clk,
                    input  reset,
                    output y);

  reg [1:0] state, nextstate;

  parameter S0 = 2'b00;
  parameter S1 = 2'b01;
  parameter S2 = 2'b10;

  // state register
  always @ (posedge clk, posedge reset)
    if (reset) state <= S0;
    else       state <= nextstate;

  // next state logic
  always @ (*)
    case (state)
      S0: nextstate = S1;
      S1: nextstate = S2;
      S2: nextstate = S0;
      default: nextstate = S0;
    endcase

  // output logic
  assign y = (state == S0);
endmodule
```

FIGURE A.33 Divide-by-3 counter state transition diagram