

Figure 8.30. Implementation of an FSM in a CPLD.

```

module control (Clock, Resetn, w, R1in, R1out, R2in, R2out, R3in, R3out, Done);
input Clock, Resetn, w;
output R1in, R1out, R2in, R2out, R3in, R3out, Done;
reg [2:1] y, Y;
parameter [2:1] A = 2'b00, B = 2'b01, C = 2'b10, D = 2'b11;

```

```

// Define the next state combinational circuit
always @(w or y)
case (y)
  A: if (w) Y = B;
     else Y = A;
  B: Y = C;
  C: Y = D;
  D: Y = A;
endcase

```

```

// Define the sequential block
always @(negedge Resetn or posedge Clock)
if (Resetn == 0) y <= A;
else y <= Y;

```

```

// Define outputs
assign R2out = (y == B);
assign R3in = (y == B);
assign R1out = (y == C);
assign R2in = (y == C);
assign R3out = (y == D);
assign R1in = (y == D);
assign Done = (y == D);

```

**endmodule**

Figure 8.35. Verilog code for the FSM in Figure 8.11.

```

module mealy (Clock, Resetn, w, z);
input Clock, Resetn, w;
output z;
reg y, Y, z;
parameter A = 0, B = 1;

// Define the next state and output combinational circuits
always @(w or y)
case (y)
  A: if (w)
     begin
       z = 0;
       Y = B;
     end
     else
     begin
       z = 0;
       Y = A;
     end
  B: if (w)
     begin
       z = 1;
       Y = B;
     end
     else
     begin
       z = 0;
       Y = A;
     end
endcase

// Define the sequential block
always @(negedge Resetn or posedge Clock)
if (Resetn == 0) y <= A;
else y <= Y;
endmodule

```

Figure 8.36. Verilog code for the Mealy machine of Figure 8.23.

```

module serial_adder (A, B, Reset, Clock, Sum);
input [7:0] A, B;
input Reset, Clock;
output [7:0] Sum;
reg [3:0] Count;
reg s, y, Y;
wire [7:0] QA, QB, Sum;
wire Run;
parameter G = 0, H = 1;

shiftrne shift_A (A, Reset, 1, 0, Clock, QA);
shiftrne shift_B (B, Reset, 1, 0, Clock, QB);
shiftrne shift_Sum (0, Reset, Run, s, Clock, Sum);

// Adder FSM
// Output and next state combinational circuit
always @(QA or QB or y)
case (y)
  G: begin
     s = QA[0] ^ QB[0];
     if (QA[0] & QB[0]) Y = H;
     else Y = G;
   end
  H: begin
     s = QA[0] ~^ QB[0];
     if (~QA[0] & ~QB[0]) Y = G;
     else Y = H;
   end
  default: Y = G;
endcase

// Sequential block
always @(posedge Clock)
if (Reset) y <= G;
else y <= Y;

// Control the shifting process
always @(posedge Clock)
if (Reset) Count = 8;
else if (Run) Count = Count - 1;
assign Run = |Count;
endmodule

```

Figure 8.49. Verilog code for the serial adder.

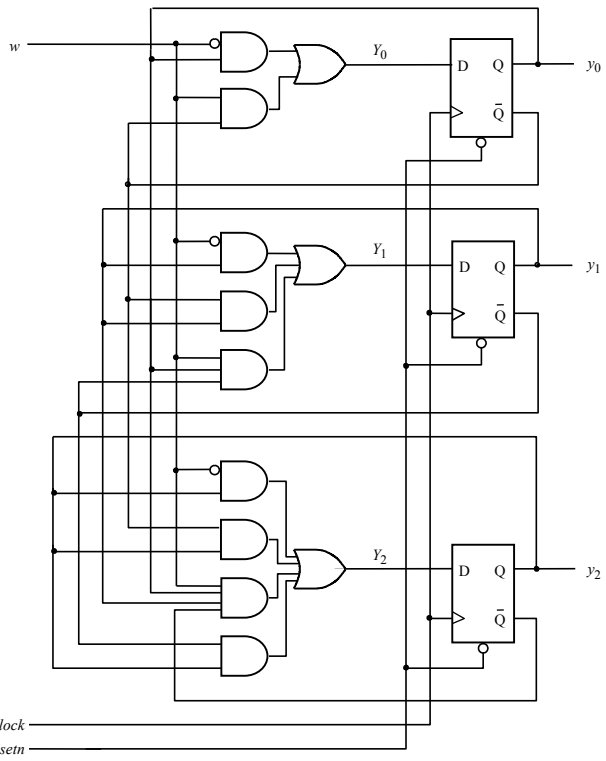


Figure 8.64. Circuit diagram for the counter.

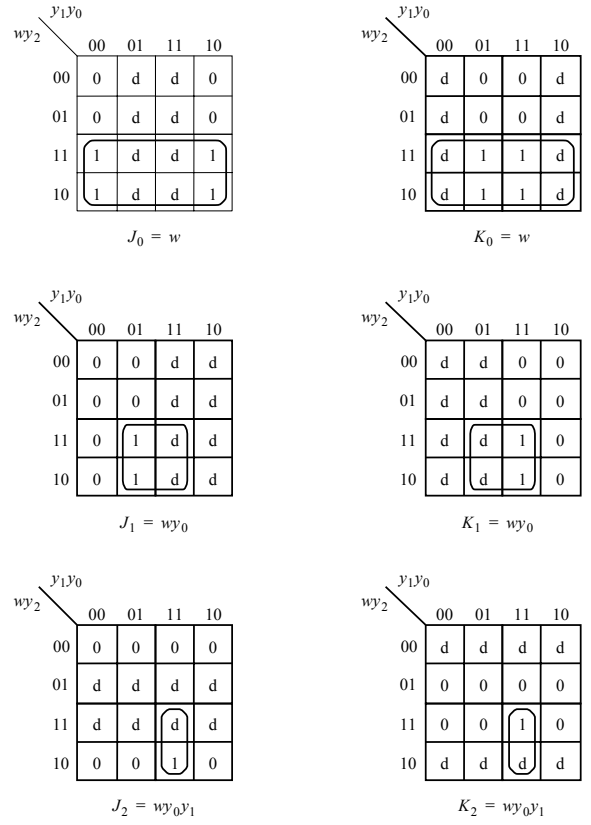


Figure 8.66. Karnaugh maps for JK flip-flops in the counter.

```

module arbiter (r, Resetn, Clock, g);
  input [1:3] r;
  input Resetn, Clock;
  output [1:3] g;
  wire [1:3] g;
  reg [2:1] y, Y;
  parameter Idle = 2'b00, gnt1 = 2'b01, gnt2 = 2'b10, gnt3 = 2'b11;

  // Next state combinational circuit
  always @(r or y)
    case (y)
      Idle: case (r)
        3'b000: Y = Idle;
        3'b1xx: Y = gnt1;
        3'b01x: Y = gnt2;
        3'b001: Y = gnt3;
        default: Y = Idle;
      endcase
      gnt1: if (r[1]) Y = gnt1;
             else Y = Idle;
      gnt2: if (r[2]) Y = gnt2;
             else Y = Idle;
      gnt3: if (r[3]) Y = gnt3;
             else Y = Idle;
      default: Y = Idle;
    endcase

  // Sequential block
  always @(posedge Clock)
    if (Resetn == 0) y <= Idle;
    else y <= Y;

  // Define output
  assign g[1] = (y == gnt1);
  assign g[2] = (y == gnt2);
  assign g[3] = (y == gnt3);

endmodule

```

Figure 8.74. Verilog code for the arbiter.

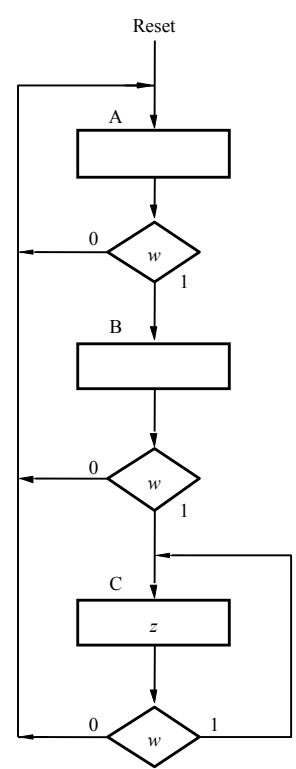


Figure 8.85. ASM chart for a simple FSM.