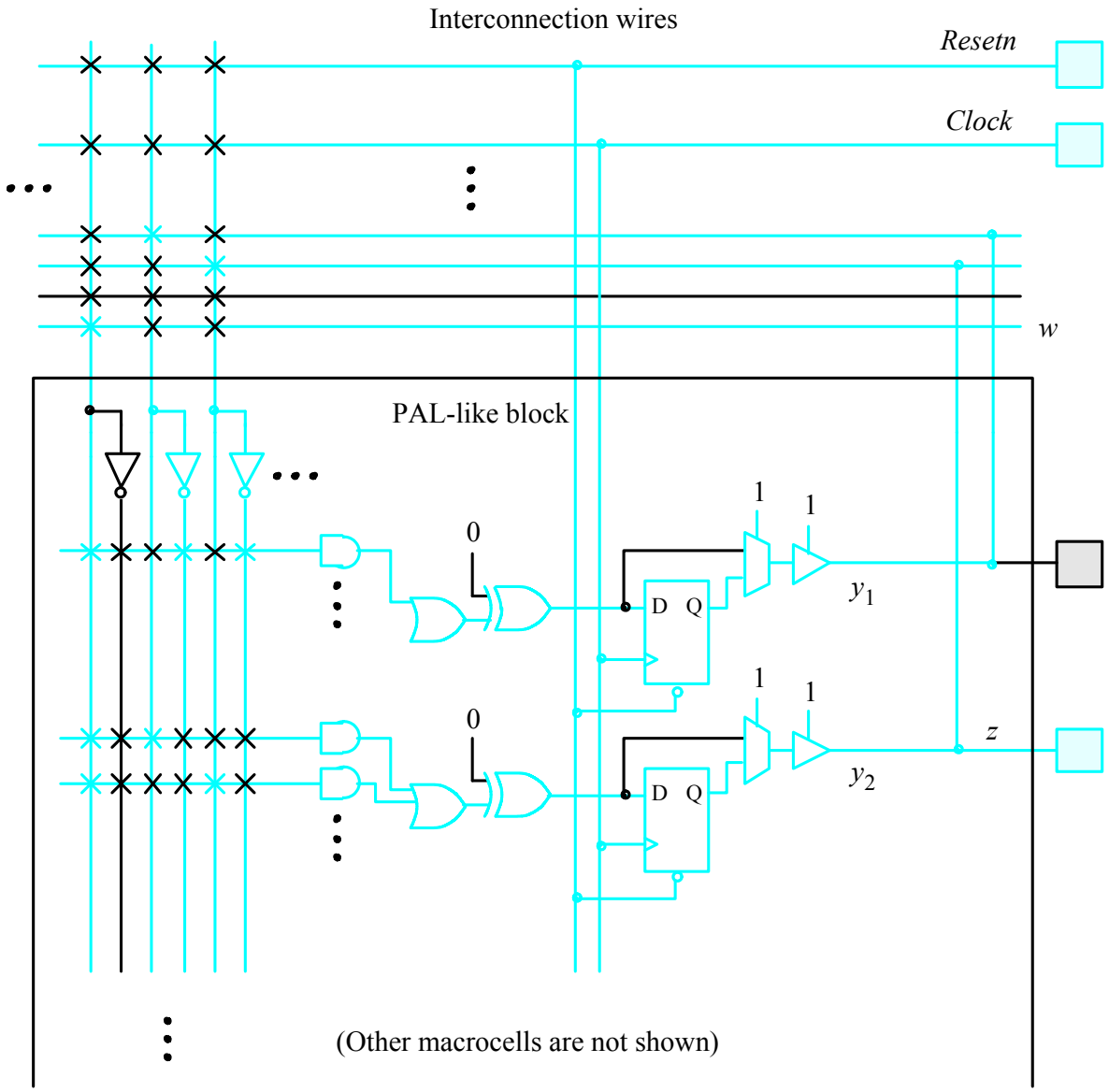


Figure 6.30. Implementation of the FSM of Figure 6.3 in a CPLD.

```
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, 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, 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
```

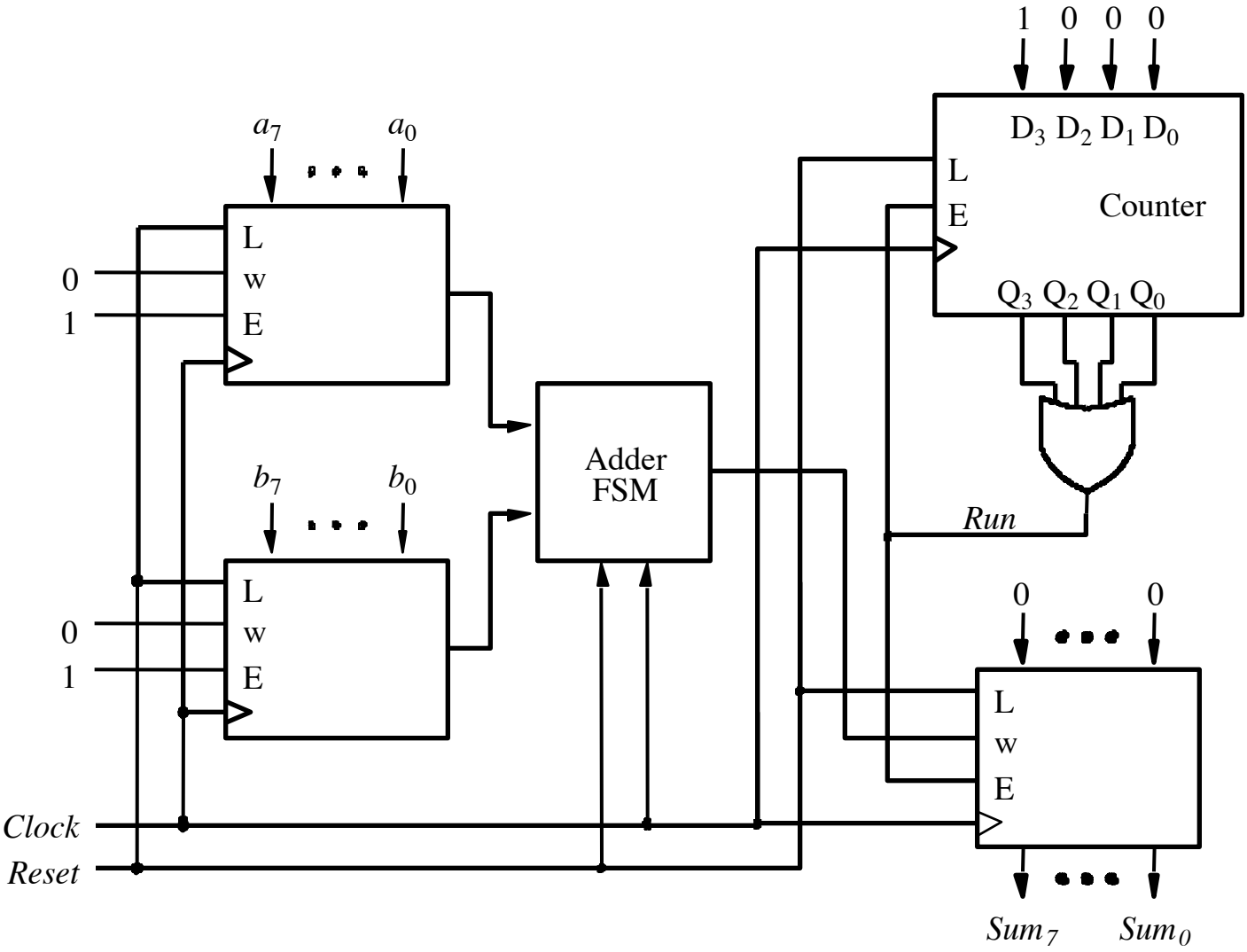
Figure 6.35. Verilog code for the FSM in Figure 6.11.

```
module mealy (Clock, Resetn, w, z);
    input Clock, Resetn, w;
    output reg z;
    reg y, Y;
    parameter A = 1'b0, B = 1'b1;

    // Define the next state and output combinational circuits
    always @(w, 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, posedge Clock)
        if (Resetn = = 0) y <= A;
        else y <= Y;

endmodule
```

Figure 6.36. Verilog code for the Mealy machine of Figure 6.23.

```
module serial_adder (A, B, Reset, Clock, Sum);
    input [7:0] A, B;
    input Reset, Clock;
    output wire [7:0] Sum;
    reg [3:0] Count;
    reg s, y, Y;
    wire [7:0] QA, QB;
    wire Run;
    parameter G = 1'b0, H = 1'b1;

    shiftrne shift_A (A, Reset, 1'b1, 1'b0, Clock, QA);
    shiftrne shift_B  (B, Reset, 1'b1, 1'b0, Clock, QB);
    shiftrne shift_Sum  (8'b0, Reset, Run, s, Clock, Sum);

    // Adder FSM
    // Output and next state combinational circuit
    always @(QA, QB, 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 6.49. Verilog code for the serial adder.



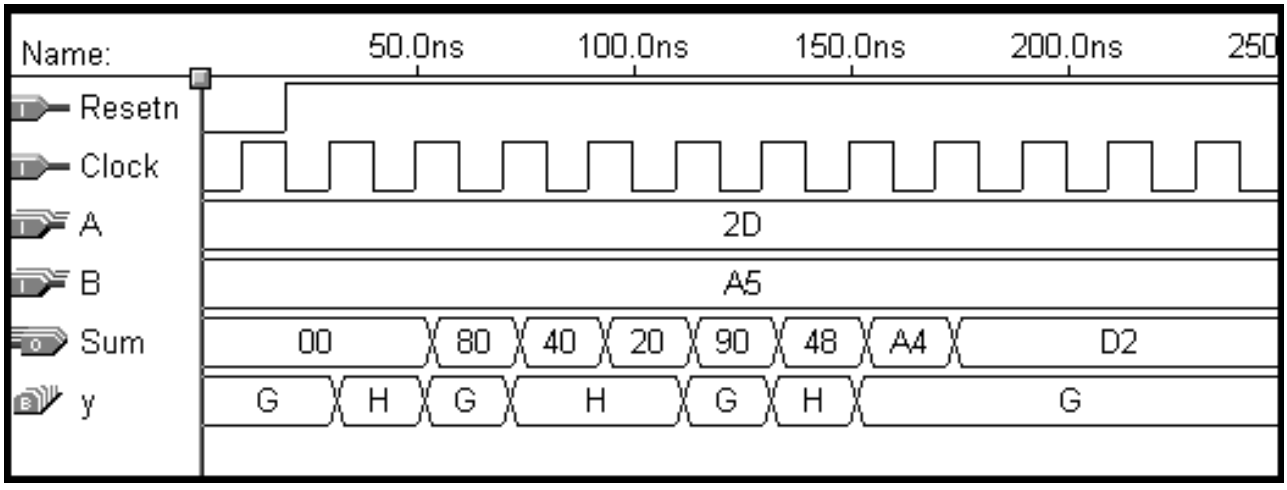


Figure 6.50. Synthesized serial adder.

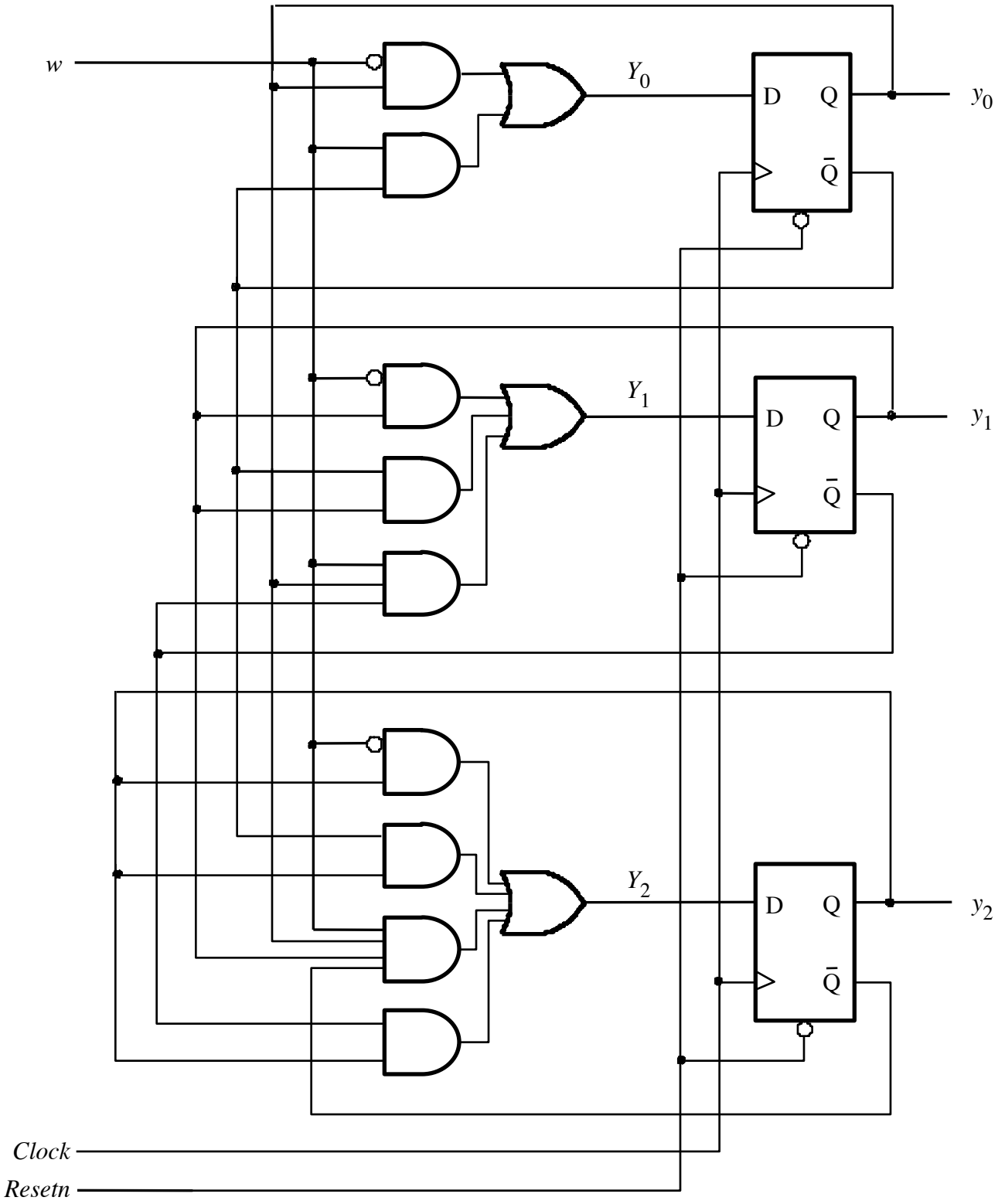
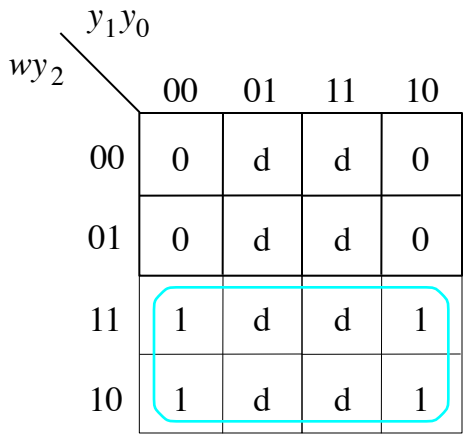
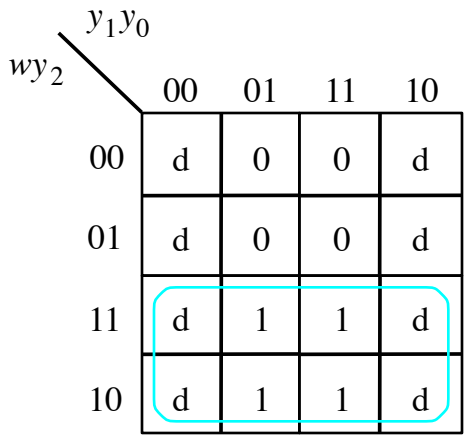


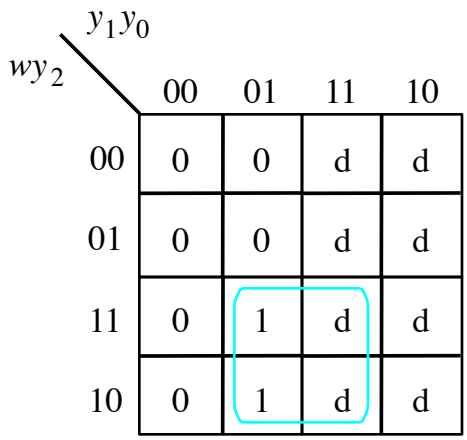
Figure 6.64. Circuit diagram for the counter implemented with D flip-flops.



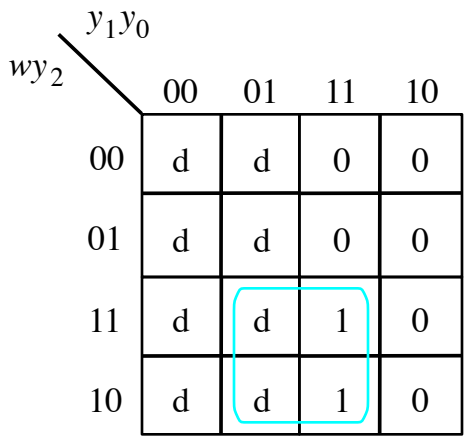
$J_0 = w$



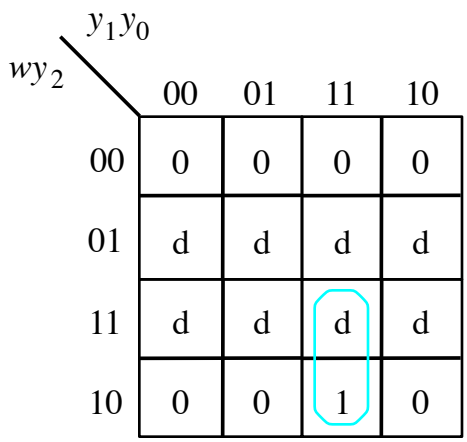
$K_0 = w$



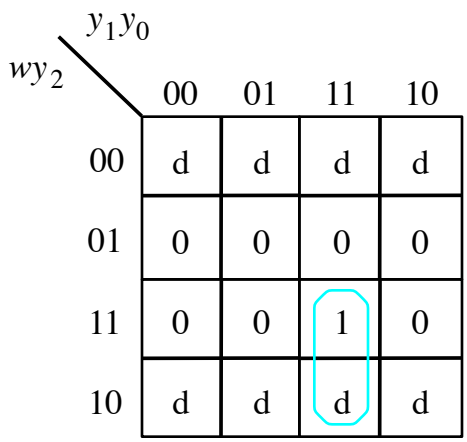
$J_1 = wy_0$



$K_1 = wy_0$



$J_2 = wy_0y_1$



$K_2 = wy_0y_1$

Figure 6.66. Karnaugh maps for JK flip-flops in the counter.

```
module arbiter (r, Resetn, Clock, g);
    input [1:3] r;
    input Resetn, Clock;
    output 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, y)
        case (y)
            Idle: casex (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 6.74. Verilog code for the arbiter.

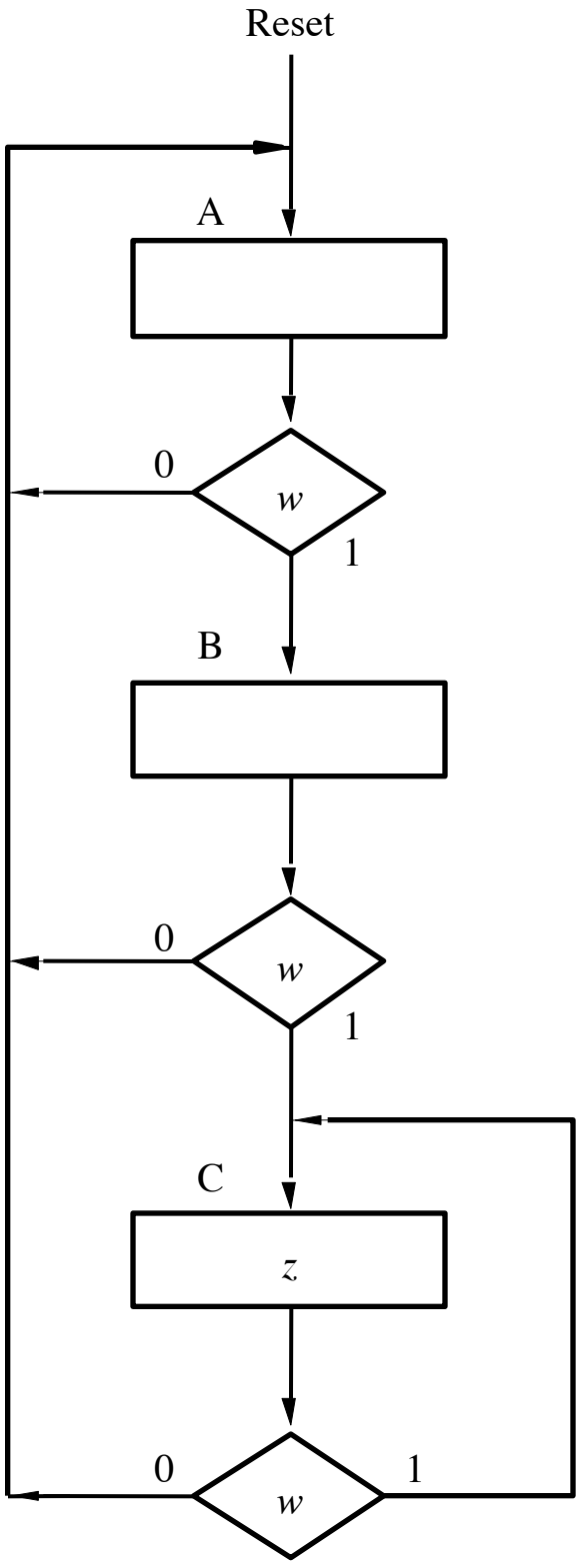


Figure 6.82. ASM chart for the FSM in Figure 6.3.

```
module sequence (Clock, Resetn, w, z);
  input Clock, Resetn, w;
  output z;
  reg [3:1] y, Y;
  parameter [3:1] A = 3'b000, B = 3'b001, C = 3'b010,
                  D = 3'b011, E = 3'b100;

  // Define the next state combinational circuit
  always @(w, y)
    case (y)
      A:  if (w)    Y = D;
          else      Y = B;
      B:  if (w)    Y = D;
          else      Y = C;
      C:  if (w)    Y = D;
          else      Y = C;
      D:  if (w)    Y = E;
          else      Y = B;
      E:  if (w)    Y = E;
          else      Y = B;
      default:      Y = 3'bxxx;
    endcase

  // Define the sequential block
  always @(negedge Resetn, posedge Clock)
    if (Resetn == 0) y <= A;
    else y <= Y;
  // Define output
  assign z = (y == C) | (y == E);
endmodule
```

Figure 6.95. Verilog code for the FSM in Figure 6.86.

```
module seqmealy (Clock, Resetn, w, z);
  input Clock, Resetn, w;
  output reg z;
  reg [2:1] y, Y;
  parameter [2:1] A = 2'b00, B = 2'b01, C = 2'b11;
  // Define the next state and output combinational circuits
  always @(w, y)
    case (y)
      A:  if (w) begin
                    z = 0; Y = C;
                  end
          else    begin
                    z = 0; Y = B;
                  end
      B:  if (w) begin
                    z = 0; Y = C;
                  end
          else    begin
                    z = 1; Y = B;
                  end
      C:  if (w) begin
                    z = 1; Y = C;
                  end
          else    begin
                    z = 0; Y = B;
                  end
       default: begin
                    z = 0; Y = 2'bxx;
                  end
    endcase
  // Define the sequential block
  always @(negedge Resetn, posedge Clock)
    if (Resetn == 0) y <= A;
    else y <= Y;
endmodule
```

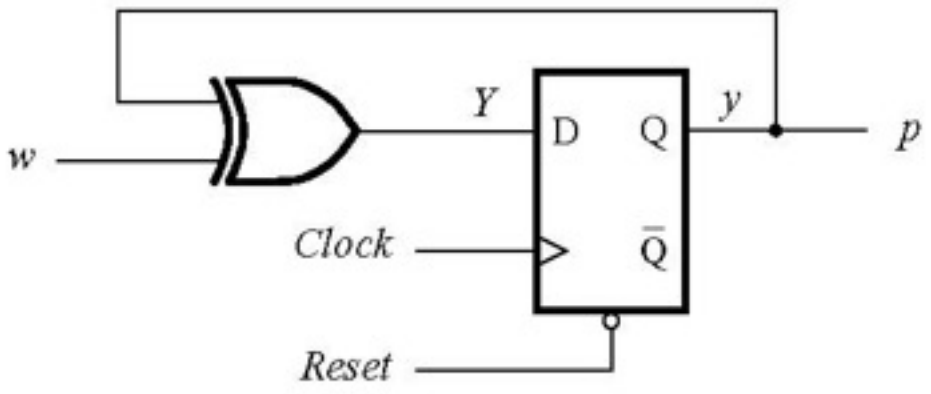
Figure 6.96. Verilog code for the FSM in Figure 6.91.

Present state	Next state		Output
	$w = 0$	$w = 1$	p
S_{even}	S_{even}	S_{odd}	0
S_{odd}	S_{odd}	S_{even}	1

(a) State table

Present state	Next state		Output
	$w = 0$	$w = 1$	p
y	Y	Y	
0	0	1	0
1	1	0	1

(b) State-assigned table



(c) Circuit

Figure 6.98. FSM for parity generation.