The bubble diagram for the finite state machine is shown in Figure 14-10. Each arc in the FSM is labeled with a label <*input*>/<*output*> where input is 2-bit and output is 1-bit. For example, *x5/0* means transition to the state pointed to by the arc, when input is *x5 (2'b01)*, and set the output to 0.

State	*Money*
S0	*0 cents*
S5	*5 cents*
S10	*10 cents*
S15	*15 cents*

Input	*coin[1:0]*
x0	*2'b00*
x5	*2'b01*
x10	*2'b10*
-	*don't care*

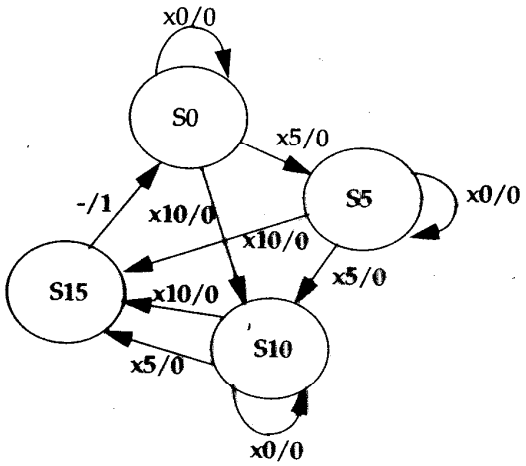

Figure 14-10 Finite State Machine for Newspaper Vending Machine

14.7.4 Verilog Description

The Verilog RTL description for the finite state machine is shown in Example 14-6.

Example 14-6 RTL Description for Newspaper Vending Machine FSM

```
//Design the newspaper vending machine coin acceptor
//using a FSM approach
module vend( coin, clock, reset, newspaper);

//Input output port declarations
input [1:0] coin;
input clock;
input reset;
output newspaper;
wire newspaper;

//internal FSM state declarations
wire [1:0] NEXT_STATE;
```

Example 14-6 RTL Description for Newspaper Vending Machine FSM (Continued)

```
reg [1:0] PRES_STATE;

//state encodings
parameter s0 = 2'b00;
parameter s5 = 2'b01;
parameter s10 = 2'b10;
parameter s15 = 2'b11;

//Combinational logic
function [2:0] fsm;
  input [1:0] fsm_coin;
  input [1:0] fsm_PRES_STATE;

  reg fsm_newspaper;
  reg [1:0] fsm_NEXT_STATE;

begin
  case (fsm_PRES_STATE)
  s0: //state = s0
  begin
    if (fsm_coin == 2'b10)
    begin
      fsm_newspaper = 1'b0;
      fsm_NEXT_STATE = s10;
    end
    else if (fsm_coin == 2'b01)
    begin
      fsm_newspaper = 1'b0;
      fsm_NEXT_STATE = s5;
    end
    else
    begin
      fsm_newspaper = 1'b0;
      fsm_NEXT_STATE = s0;
    end
  end

 s5: //state = s5
  begin
    if (fsm_coin == 2'b10)
    begin
      fsm_newspaper = 1'b0;
      fsm_NEXT_STATE = s15;
```

Example 14-6 RTL De

```
    end
    else if (fsm_coi
    begin
      fsm_newspaper
      fsm_NEXT_STATE
    end
    else
    begin
      fsm_newspaper
      fsm_NEXT_STATE
    end

 s10: //state = s10
  begin
    if (fsm_coin ==
    begin
      fsm_newspaper
      fsm_NEXT_STATE
    end
    else if (fsm_coi
    begin
      fsm_newspaper
      fsm_NEXT_STATE
    end
    else
    begin
      fsm_newspaper
      fsm_NEXT_STATE
    end
  end
  s15: //state = s1
  begin
    fsm_newspaper =
    fsm_NEXT_STATE
  end
  endcase
  fsm = {fsm_newspa
end
endfunction

//Reevaluate combin
//is put or the pre
assign {newspaper,
```

Example 14-6 RTL Description for Newspaper Vending Machine FSM (Continued)

```
    end
    else if (fsm_coin == 2'b01)
    begin
      fsm_newspaper = 1'b0;
      fsm_NEXT_STATE = s10;
    end
    else
    begin
      fsm_newspaper = 1'b0;
      fsm_NEXT_STATE = s5;
    end

 s10: //state = s10
  begin
    if (fsm_coin == 2'b10)
    begin
      fsm_newspaper = 1'b0;
      fsm_NEXT_STATE = s15;
    end
    else if (fsm_coin == 2'b01)
    begin
      fsm_newspaper = 1'b0;
      fsm_NEXT_STATE = s15;
    end
    else
    begin
      fsm_newspaper = 1'b0;
      fsm_NEXT_STATE = s10;
    end
  end
  s15: //state = s15
  begin
    fsm_newspaper = 1'b1;
    fsm_NEXT_STATE = s0;
  end
  endcase
  fsm = {fsm_newspaper, fsm_NEXT_STATE};
end
endfunction

//Reevaluate combinational logic each time a coin
//is put or the present state changes
assign {newspaper, NEXT_STATE} = fsm(coin, PRES_STATE);
```

Example 14-6 RTL Description for Newspaper Vending Machine FSM (Continued)

```
//clock the state flipflops.
//use synchronous reset
always @(posedge clock)
begin
  if (reset == 1'b1)
    PRES_STATE <=  s0;
  else
    PRES_STATE <=  NEXT_STATE;
end

endmodule
```

14.7.5 Technology Library

We defined abc_100 technology in Section 14.4.1, *RTL to Gates*. We will use abc_100 as the target technology library. abc_100 contains the following library cells:

```
//Library cells for abc_100 technology

VNAND//2-input nand gate
VAND//2-input and gate
VNOR//2-input nor gate
VOR//2-input or gate
VNOT//not gate
VBUF//buffer
NDFF//Negative edge triggered D flipflop
PDFF//Positive edge triggered D flipflop
```

14.7.6 Design Constraints

Timing critical is the only design constraint we used in this design. Typically, design constraints are more elaborate.

14.7.7 Logic Synthesis

We synthesize the RTL description by using the specified design constraints and technology library and obtain the optimized gate-level netlist.

14.7.8 Optimized G

We use logic synthesis to
optimized gate-level netli

Example 14-7 Optim

```
module vend ( coin,
input  [1:0] coin;
input  clock, reset;
output newspaper;
    wire \PRES_STATE
         n303, n304
         n295, n296
    PDFF \PRES_STATE

    PDFF \PRES_STATE

    VOR U119 ( .in0(
    VAND U118 ( .in0
                .c
    VNAND U117 ( .in
    VNOR U116 ( .in0
    VNOR U115 ( .in0
    VNOT U128 ( .in(
    VAND U114 ( .in0
    VNOT U127 ( .in(
    VAND U113 ( .in0
    VNOT U126 ( .in(
    VNAND U112 ( .in
    VNAND U125 ( .in
    VNOR U111 ( .in0
    VNAND U124 ( .in
    VAND U110 ( .in0
    VNAND U123 ( .in
    VNAND U122 ( .in
    VNAND U121 ( .in
    VOR U120 ( .in0
endmodule
```

The schematic diagram