Listing 8.7(cont.) doorlock2_top.vhd

```
begin
      btn012 <= btn(2) or btn(1) or btn(0);
      clr <= btn(3);
      bn(1) <= btn(2);
      bn(0) <= btn(1);

U1 : clkdiv
      port map(
            mclk => mclk, clr => clr, clk190 => clk190);

U2 : clock_pulse
      port map(
            inp => btn012, cclk => clk190,
            clr => clr, outp => clkp);

U3 : doorlock
      port map(
            clk => clkp, clr => clr, bn => bn, sw => sw,
            pass => ld(1), fail => ld(0));

end doorlock2_top;
```

Example 58: Traffic Lights

It is often useful to be able to sequence through an arbitrary number of states, staying in each state an arbitrary amount of time. For example, consider the set of traffic lights shown in Figure 8.13. The lights are assumed to be at a four-way intersection with one street going north-south and the other road going east-west.

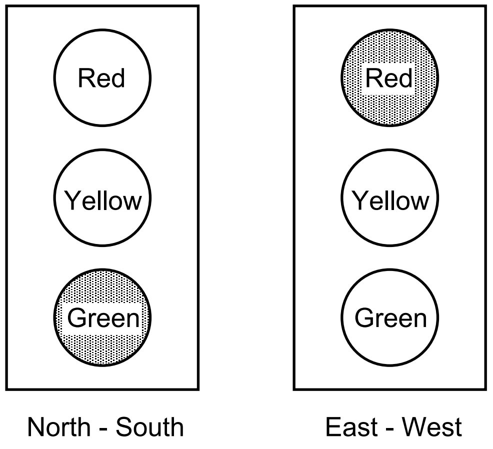

Figure 8.13 Six colored LEDs can represent a set of traffic lights

To simulate these traffic lights we will use the red, yellow, and green LEDs connected to *ld*(7:2) on the FPGA board and cycle through the six states shown in Table 8.2. A state diagram for controlling these traffic lights is shown in Fig. 8.14. If we use a 3 Hz clock to drive this state diagram then a delay of 1 second is achieved by staying in a state for three clock cycles. Similarly, a delay of 5 second is achieved by staying in a state for fifteen clock cycles. The *count* variable in Fig. 8.14 will be reset to zero when moving to the next state after a timeout.

Listing 8.8 is a VHDL program that implements the state diagram in Fig. 8.14 and its simulation is shown in Fig. 8.15. Because we need a counter for the delay count it is more convenient in this case to combine the state register and combinational modules C1 in the Moore machine in Fig. 8.3 into a single sequential *process* as shown in Listing 8.8. Note in this case we use only a single *state* variable.

To generate the 3 Hz signal we will use the version of *clkdiv* shown in Listing 8.9. The top-level VHDL program is given in Listing 8.10.

Table 8.2 Traffic Light States

State	North - South	East - West	Delay (sec.)
0	Green	Red	5
1	Yellow	Red	1
2	Red	Red	1
3	Red	Green	5
4	Red	Yellow	1
5	Red	Red	1

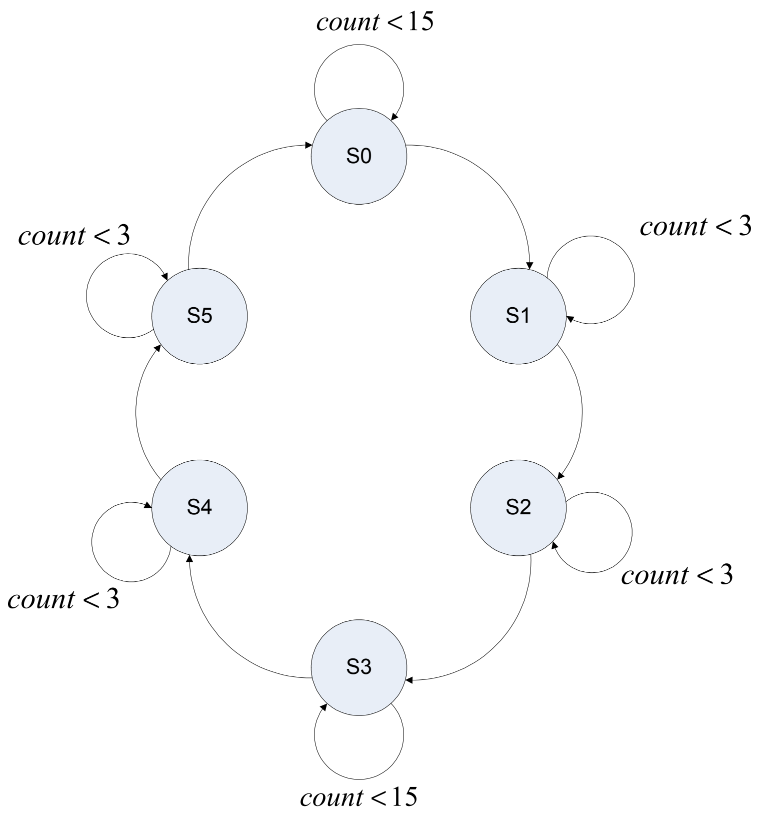

Figure 8.14 State diagram for controlling traffic lights

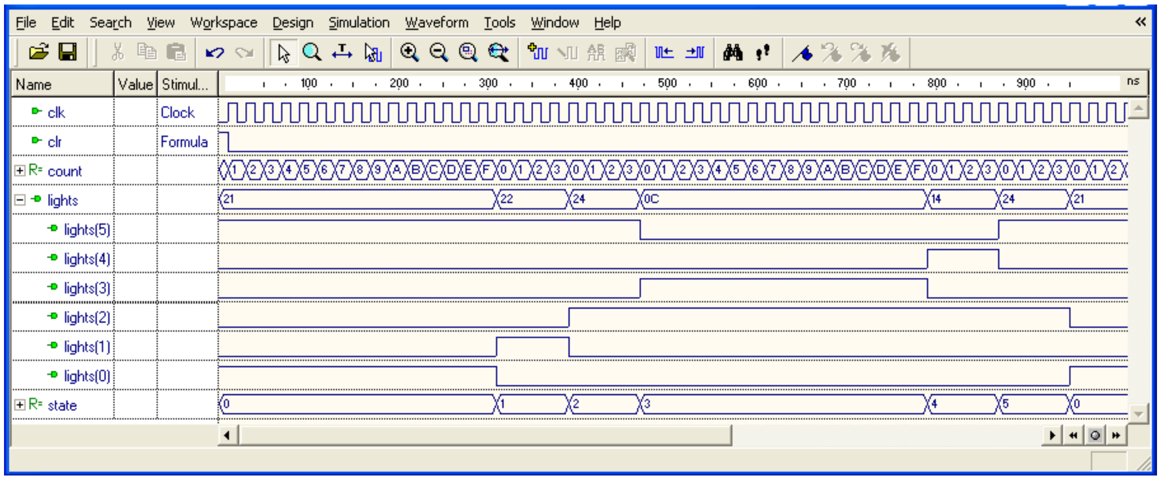

Figure 8.15 Simulation of the VHDL program in Listing 8.8

Listing 8.8 traffic.vhd

```
-- Example 62: traffic lights
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_unsigned.all;

entity traffic is
  port (clk: in STD_LOGIC;
         clr: in STD_LOGIC;
        lights: out STD_LOGIC_VECTOR(5 downto 0));
end traffic;

architecture traffic of traffic is
type state_type is (s0, s1, s2, s3, s4, s5);
signal state: state_type;
signal count: STD_LOGIC_VECTOR(3 downto 0);
constant SEC5: STD_LOGIC_VECTOR(3 downto 0)  := "1111";
constant SEC1: STD_LOGIC_VECTOR(3 downto 0)  := "0011";
begin
process(clk, clr)
begin
       if clr = '1' then
              state <= s0;
             count <= X"0";
       elsif clk'event and clk = '1' then
         case state is
              when s0 =>
                if count < SEC5 then
                  state <= s0;
                  count <= count + 1;
                else
                  state <= s1;
                  count <= X"0";
                end if;
```

Listing 8.8 (cont.) traffic.vhd

```
            when s1 =>
              if count < SEC1 then
               state <= s1;
                count <= count + 1;
              else
               state <= s2;
                count <= X"0";
              end if;
            when s2 =>
              if count < SEC1 then
                state <= s2;
                count <= count + 1;
              else
               state <= s3;
                count <= X"0";
              end if;
            when s3 =>
              if count < SEC5 then
                state <= s3;
                count <= count + 1;
              else
               state <= s4;
                count <= X"0";
              end if;
            when s4 =>
              if count < SEC1 then
                state <= s4;
                count <= count + 1;
              else
               state <= s5;
                count <= X"0";
              end if;
            when s5 =>
              if count < SEC1 then
                state <= s5;
                count <= count + 1;
              else
               state <= s0;
                count <= X"0";
              end if;
           when others =>
                state <= s0;
            end case;
      end if;
end process;

C2: process(state)
begin
      case state is
        when s0 => lights <= "100001";
        when s1 => lights <= "100010";
        when s2 => lights <= "100100";
        when s3 => lights <= "001100";
        when s4 => lights <= "010100";
        when s5 => lights <= "100100";
        when others => lights <= "100001";
      end case;
end process;
end traffic;
```

Listing 8.9 clkdiv.vhd

```
-- Example 52: clock divider
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_unsigned.all;

entity clkdiv is
       port(
             mclk : in STD_LOGIC;
             clr : in STD_LOGIC;
             clk3 : out STD_LOGIC
           );
end clkdiv;

architecture clkdiv of clkdiv is
signal q:STD_LOGIC_VECTOR(23 downto 0);
begin
  -- clock divider
  process(mclk, clr)
  begin
    if clr = '1' then
      q <= X"000000" & '0';
    elsif mclk'event and mclk='1' then
      q <= q + 1;
    end if;
  end process;
  clk3 <= q(23);  -- 3 Hz
end clkdiv;
```

Listing 8.10 traffic_lights_top.vhd

```
-- Example 62: traffic_lights_top
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use work.traffic_components.all;

entity traffic_lights_top is
       port(
             mclk : in STD_LOGIC;
             btn : in STD_LOGIC_VECTOR(3 downto 3);
             ld : out STD_LOGIC_VECTOR(7 downto 2)
           );
end traffic_lights_top;

architecture traffic_lights_top of traffic_lights_top is
signal clr, clk3: STD_LOGIC;
begin
  clr <= btn(3);

  U1: clkdiv
      port map (mclk=>mclk, clr=>clr, clk3=>clk3);

  U2: traffic
      port map (clk=>clk3, clr=>clr, lights=>ld);

end traffic_lights_top;
```