3.2 (a)(10), (e)(20), 3.3(30), 3.6(20), 4.6(a)(20)

3.2  An older-model Thunderbird car has three left and three right tail lights, which flash in unique patterns to indicate left and right turns.

Left-turn pattern:  Right-turn pattern:

<table>
<thead>
<tr>
<th>LC</th>
<th>LB</th>
<th>LA</th>
<th>RA</th>
<th>RB</th>
<th>RC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Design a Moore sequential network to control these lights. The network has three inputs, LEFT, RIGHT, and HAZ. LEFT and RIGHT come from driver’s turn-signal switch and cannot be 1 at the same time. As indicated above, when LEFT = 1, the lights flash in a pattern LA on, LA and LB on, LA, LB, and LC on and all off; then the sequence repeats. When RIGHT = 1, the light sequence is similar. IF a switch from LEFT to RIGHT (or vice versa) occurs in the middle of a flashing sequence, the network should immediately go to the IDLE state (lights off) and then start the new sequence. HAZ comes from the hazard switch, and when HAZ = 1, all six lights flash on and off in unison. HAZ takes precedence if LEFT or RIGHT is also on. Assume that a clock signal is available with a frequency equal to the desired flashing rate.

(a)  Draw the state graph (8 states).

(e) Write a VHDL model

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity THUNDERBIRD is
  port (L, R, H, CLK : in std_logic;
        LC, LB, LA, RA, RB, RC : out std_logic);
end THUNDERBIRD;
```
architecture BEHAVE of THUNDERBIRD is
  type STATE_TYPE is (S0, S1, S2, S3, S4, S5, S6, S7);
  signal CURRENT_STATE, NEXT_STATE : STATE_TYPE;
begin
  process(CURRENT_STATE, L, R, H)
  variable INPUTS : std_logic_vector(2 downto 0);
  begin
    INPUTS := L&V&R&H;
    case CURRENT_STATE is
      when S0 => if (INPUTS = "000") then
        NEXT_STATE <= S0;
        elsif (INPUTS = "100") then
          NEXT_STATE <= S1;
          elsif (INPUTS = "010") then
            NEXT_STATE <= S4;
          else
            NEXT_STATE <= S7;
          end if;
        when S1 => if (INPUTS(0) = '1') then
          NEXT_STATE <= S7;
          elsif (INPUTS = "100") then
            NEXT_STATE <= S2;
          else
            NEXT_STATE <= S0;
          end if;
        when S2 => if (INPUTS(0) = '1') then
          NEXT_STATE <= S7;
          elsif (INPUTS = "100") then
            NEXT_STATE <= S3;
          else
            NEXT_STATE <= S0;
          end if;
        when S3|S6 => if (INPUTS(0) = '1') then
          NEXT_STATE <= S7;
          else
            NEXT_STATE <= S0;
          end if;
        when S4 => if (INPUTS(0) = '1') then
          NEXT_STATE <= S7;
          elsif (INPUTS = "010") then
            NEXT_STATE <= S5;
          else
            NEXT_STATE <= S0;
          end if;
        when S5 => if (INPUTS(0) = '1') then
          NEXT_STATE <= S7;
          elsif (INPUTS = "010") then
            NEXT_STATE <= S6;
          else
            NEXT_STATE <= S0;
          end if;
        when S7 => NEXT_STATE <= S0;
      end case;
    end process;


### 3.3 Find a minimum-row PLA table to implement the following set of functions.

(a) (15 points) \( f_1(A, B, C, D) = m(4, 5, 10, 11, 12) \),
\( f_2(A, B, C, D) = m(0, 1, 3, 4, 8, 11) \)
\( f_3(A, B, C, D) = m(0, 4, 10, 12, 14) \)

(b) \( f_1(A, B, C, D) = m(3, 4, 6, 9, 11) \),
\( f_2(A, B, C, D) = m(2, 4, 8, 10, 11, 12) \)
\( f_3(A, B, C, D) = m(3, 6, 7, 10, 11) \)
3.6 (modified) Write a VHDL model on a 6-bit binary up-down counter with reset. Simulate the model and verify that the counter works.

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity UP_DOWN_COUNTER is
  port (UP, DOWN, CLK, RESET : std_logic;
        QOUT : out UNSIGNED(5 downto 0));
end UP_DOWN_COUNTER;

architecture BEHAVE of UP_DOWN_COUNTER is
begin
  process(UP, DOWN, RESET, CLK)
  variable QTEMP : UNSIGNED(5 downto 0);
  begin
    if (RESET = '1') then
      QTEMP := "000000";
    elsif (CLK'event and CLK = '1') then
      if (UP = '1') then
        if (QTEMP = UNSIGNED("111111")) then
          QTEMP := "000000";
        else
          QTEMP := QTEMP + '1';
        end if;
      elsif (DOWN = '1') then
        if (QTEMP = UNSIGNED("000000")) then
          QTEMP := "111111";
        else
          QTEMP := QTEMP - '1';
        end if;
      end if;
    end if;
    QOUT <= QTEMP;
  end process;
end BEHAVE;
```

4.6 In Section 4.4 we developed an algorithm for multiplying signed binary fractions, with negative fractions represented in 2’s complement. (a) Illustrate this algorithm by multiplying 1.0111 by 1.101.

```
| 0 . 0 0 0 0 | | 1 . 0 1 1 1 | | add |
| 1 . 0 1 1 1 | | add |
| 1 . 1 1 0 1 1 | | shift w/sign extend |
| 1 . 1 1 0 1 1 1 | | shift w/sign extend |
| 1 . 0 1 1 1 | | |
| 1 . 0 1 0 0 1 1 | | add |
| 1 . 1 0 1 0 0 1 | | shift w/sign extend |
| 0 . 1 0 0 1 | | |
| 0 . 0 0 1 1 0 1 1 | | add 2’s complement |
```