VLSI DESIGN LAB. ASSIGNMENT -3

SAURAV GUPTA

ROLL No. – 153070046

Dept.- Electrical Engineering

Specialisation – Microelectronics (TA)

Batch - 2015-17

Teacher concerned – VIRENDRA SINGH

PROBLEM STATEMENT:

Design a lift controller, which controls a lift that services passengers in a 6 floor building.

Each floor has two hall call buttons (except the ground and top floors where they have only one), an up button to request transport to a higher floor and a down button to request transport to a lower floor. These buttons illuminate when pressed. The illumination is cancelled when lift visits the floor and is either moving in the desired direction or has no outstanding requests. In the latter case, if both floor buttons are pressed, only one should be cancelled.

Lift has a set of buttons (car call button), one for each floor. These illuminate when pressed and cause the lift to visit the corresponding floor. The illumination is cancelled when the corresponding floor is visited by the lift.

When lift has no requests to service, it should remain at its final destination with its doors closed and await further requests.

The lift control system has a set of sensors to detect the floor it is visiting which is communicating lift asynchronously.

The controller should satisfy the following conditions:

• A upward traveling lift should not change its direction at any floor when it has passengers wishing to go to higher floor, and vice---versa for downward traveling lift.

• Any request (hall call, and car call) should eventually be serviced Write a synthesizable behavioural description of the above circuit in VHDL.

Sol:

Controller FSM

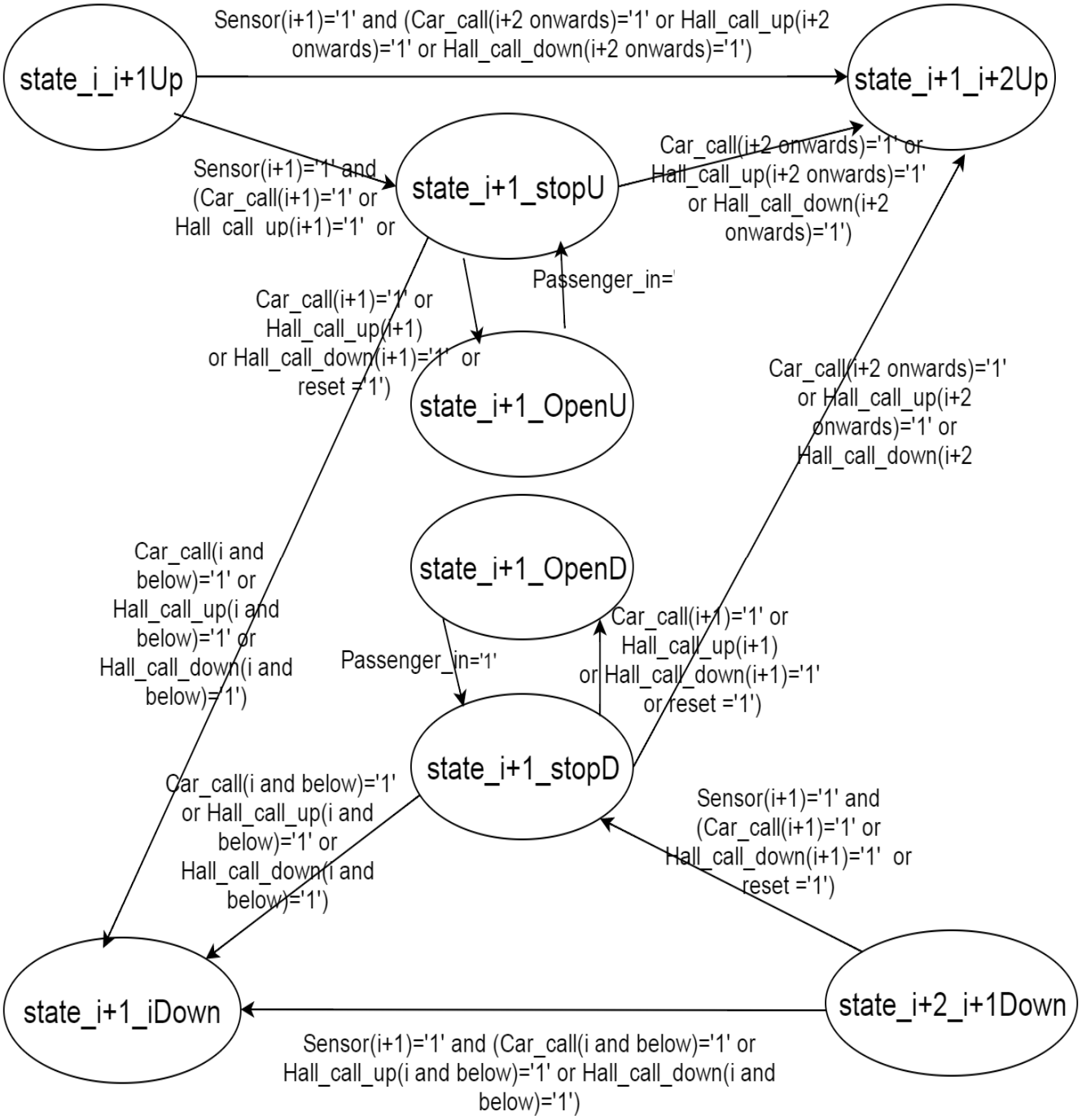


Fig. 1 Controller FSM for any state i

Circuit Diagram

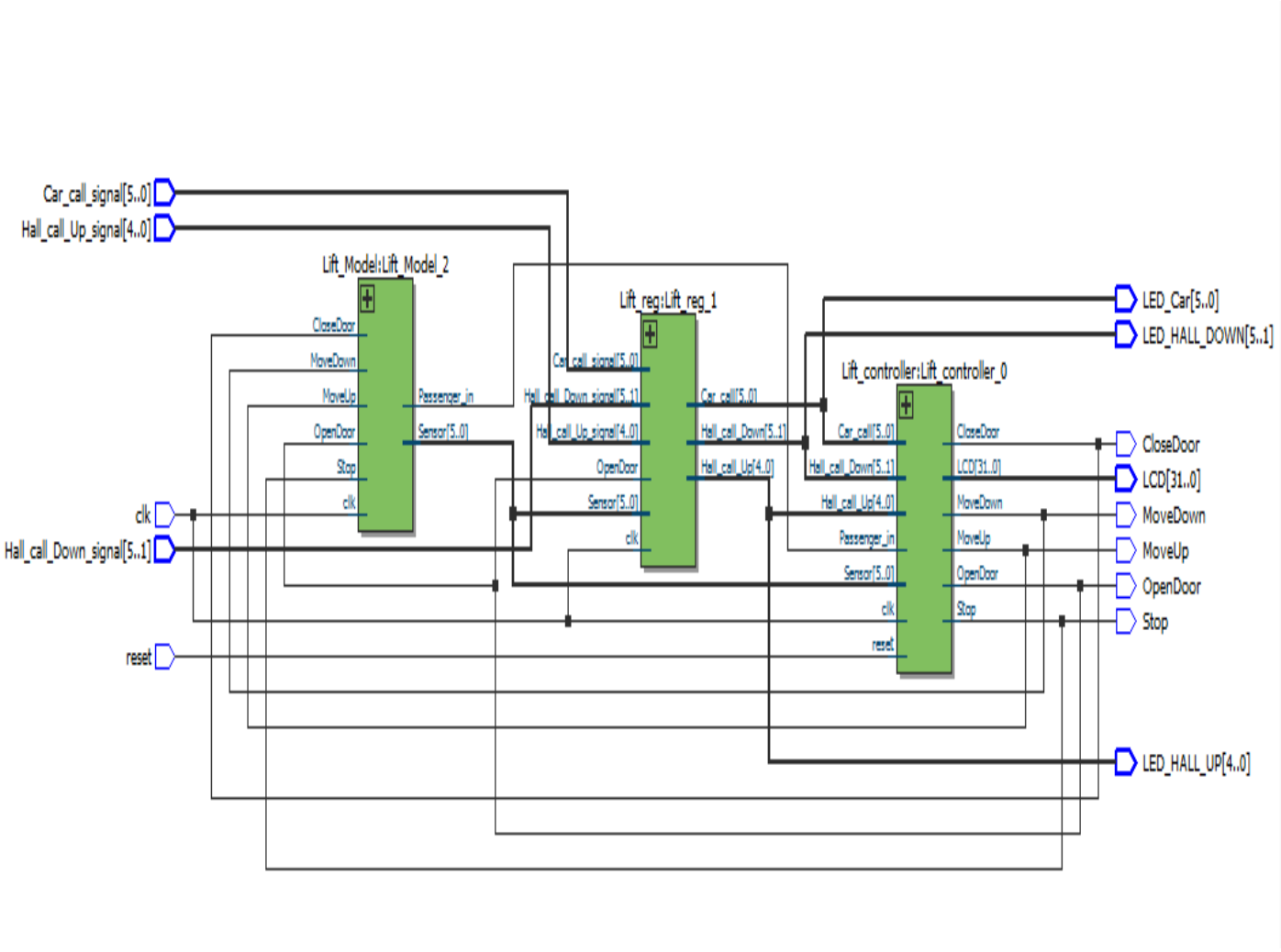


Fig. 2 Circuit Diagram

Individual Components

1. Lift Model

- **Lift**
- **counter**

2. Lift Register

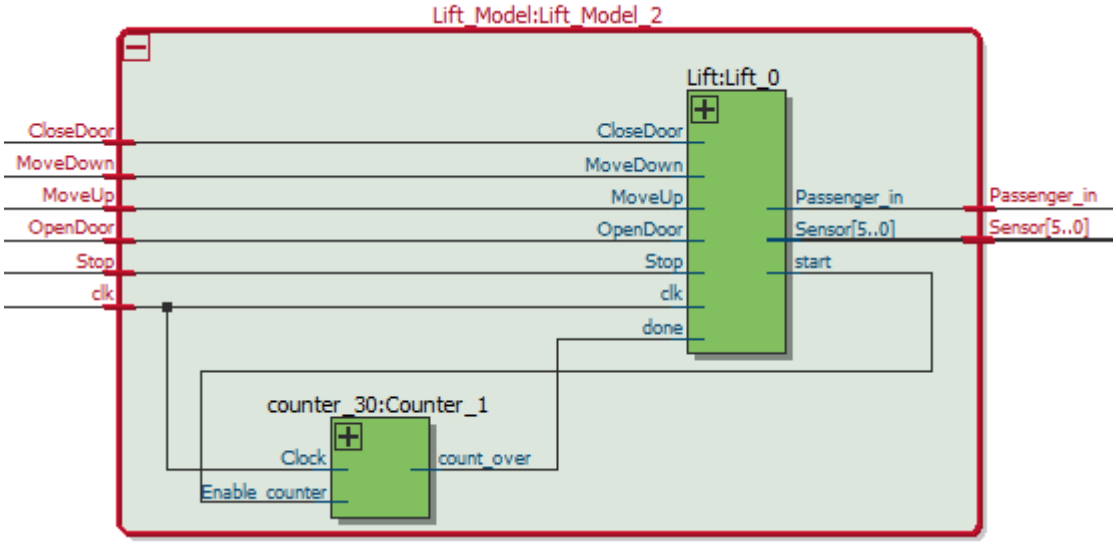
3. Lift Controller

1. Lift Model

--This Model acts as a Lift.

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

 entity  Lift_Model is
 port(clk: in std_logic;
        MoveUp: in std_logic;
        MoveDown: in std_logic;
        OpenDoor: in std_logic;
        CloseDoor: in std_logic;
        Stop: in std_logic;
        Sensor: out std_logic_vector(5 downto 0);
        Passenger_in: out std_logic
       );
end Lift_Model;

architecture behav of Lift_Model is

      component Lift
      port(clk: in std_logic;
        MoveUp: in std_logic;
        MoveDown: in std_logic;
        OpenDoor: in std_logic;
        CloseDoor: in std_logic;
        Stop: in std_logic;
        start: out std_logic;
        done: in std_logic;
        Sensor: out std_logic_vector(5 downto 0);
        Passenger_in: out std_logic
       );
      end component;

      component counter_30
         port(Clock: in std_logic;
                  Enable_counter: in std_logic;
                  count_over: out std_logic
                  );
    end component;

signal start: std_logic;
signal done : std_logic;
```

```
begin
      Lift_0 : Lift port
map(clk=>clk,MoveUp=>MoveUp,MoveDown=>MoveDown,OpenDoor=>OpenDoor,Clos
eDoor=>CloseDoor,Stop=>Stop,start=>start,done=>done,Sensor=>Sensor,Pas
senger_in=>Passenger_in);
      Counter_1: counter_30 port
map(Clock=>clk,Enable_counter=>start,count_over=>done);
end behav;
```



1.a Lift

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

 entity  Lift is
 port(clk: in std_logic;
        MoveUp: in std_logic;
        MoveDown: in std_logic;
        OpenDoor: in std_logic;
        CloseDoor: in std_logic;
        Stop: in std_logic;
        start: out std_logic;
        done: in std_logic;
        Sensor: out std_logic_vector(5 downto 0);
        Passenger_in: out std_logic
       );
end Lift;

architecture behav of Lift is

signal Sensor_temp: std_logic_vector(5 downto 0):="000001";
begin
```

```
process(clk,MoveDown,MoveUp,OpenDoor,CloseDoor,done,Stop)
begin
      if(clk='1' and clk'event) then
            if(MoveUp ='1') then
                  start<='1';
                  if(done='1') then
                        start<='0';
                        Sensor_temp<=
std_logic_vector(shift_left(unsigned(Sensor_temp),1));
                  end if;

            elsif(MoveDown ='1') then
                  start<='1';
                  if(done='1') then
                        start<='0';
                        Sensor_temp<=
std_logic_vector(shift_right(unsigned(Sensor_temp),1));
                  end if;

            elsif(OpenDoor='1' and stop='1') then
                  start<='1';Passenger_in<='0';
                  if(done='1') then
                        start<='0';
                        Passenger_in<='1';
                  end if;

            elsif(CloseDoor='1' and stop='1') then
                  start<='0';
            end if;
      end if;
end process;
      Sensor<= Sensor_temp;
end behav;
```

1. b Counter_30

--counts 30 clock cycle to provide delay for moving from one floor to other and also for time of closing door.

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

    entity counter_30 is
        port(Clock: in std_logic;
                  Enable_counter: in std_logic;
                  count_over: out std_logic
                  );
    end counter_30;
```

```
    architecture Behv of counter_30 is
     signal temp:integer:=0;
     begin
           process(Clock,Enable_counter)
           begin
           if Enable_counter='1' then
                 if (Clock ='1' and Clock'event) then
                       if temp=30 then
                             count_over <= '1';
                       else
                             temp <=temp+1;
                       end if;
                 end if;
           else
                 temp<=0;
                 count_over<='0';

           end if;
           end process;

    end Behv;
```

Lift Controller

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

entity  Lift_controller is
 port(clk: in std_logic;
       reset: in std_logic;
       Sensor: in std_logic_vector(5 downto 0);
       Car_call: in std_logic_vector(5 downto 0);
       Hall_call_Up: in std_logic_vector(4 downto 0);
       Hall_call_Down: in std_logic_vector(5 downto 1);
       Passenger_in: in std_logic;
       MoveUp: out std_logic;
       MoveDown: out std_logic;
       OpenDoor: out std_logic;
       CloseDoor: out std_logic;
       Stop: out std_logic;
       LCD: out integer
      );
end Lift_controller;

architecture behave  of Lift_controller is

type MyState is
(s_0_stop,s_0_open,s_01_Up,s_10_Down,s_1_stopU,s_1_stopD,s_1_openU,s_1
_openD,s_12_Up,s_21_Down,s_2_stopU,s_2_stopD,s_2_openU,s_2_openD,s_23_
```

```
Up,s_32_Down,s_3_stopU,s_3_stopD,s_3_openU,s_3_openD,s_34_Up,s_43_Down
,s_4_stopU,s_4_stopD,s_4_openU,s_4_openD,s_45_Up,s_54_Down,s_5_stop,s_
5_open);
   signal state_signal : MyState;

begin  -- behave

  next_state: process
(reset,clk,Sensor,Hall_call_Up,Hall_call_Down,Car_call,Passenger_in,st
ate_signal)
    variable next_state_var : MyState;
  begin  -- process next_state
    next_state_var := state_signal;

      case state_signal is
           when s_0_stop =>
                if(reset = '1' or Hall_call_Up(0)='1' or
Car_call(0)='1') then
                     next_state_var := s_0_open;
                elsif (Hall_call_Down(1)='1' or Hall_call_Up(1)='1' or
Car_call(1)='1' or Hall_call_Down(2)='1' or Hall_call_Up(2)='1' or
Car_call(2)='1' or Hall_call_Down(3)='1'or Hall_call_Up(3)='1' or
Car_call(3)='1' or Hall_call_Down(4)='1'or Hall_call_Up(4)='1' or
Car_call(4)='1' or Hall_call_Down(5)='1' or Car_call(5)='1') then
                          next_state_var := s_01_Up;
                end if;

           when s_0_open =>
                if(reset = '1') then
                     next_state_var := s_0_open;
                else
                     if(Passenger_in ='1') then
                          next_state_var := s_0_stop;
                     end if;
                end if;

           when s_01_Up =>
                if(Sensor(1)='1' and (Car_call(1)='1' or
Hall_call_Up(1)='1' or reset='1'))  then
                     next_state_var := s_1_stopU;
                elsif(Sensor(1)='1' and (Hall_call_Down(2)='1' or
Hall_call_Up(2)='1' or Car_call(2)='1' or Hall_call_Down(3)='1'or
Hall_call_Up(3)='1' or Car_call(3)='1' or Hall_call_Down(4)='1'or
Hall_call_Up(4)='1' or Car_call(4)='1' or Hall_call_Down(5)='1' or
Car_call(5)='1' )) then
                     next_state_var := s_12_Up;
                elsif(Sensor(1)='1') then
                     next_state_var := s_1_stopU;
                end if;
           when s_10_Down =>
                if(Sensor(0)='1') then
                     next_state_var := s_0_stop;
```

```
                end if;

           when s_1_stopU =>
                if(reset = '1' or Car_call(1)='1' or
Hall_call_Down(1)='1' or Hall_call_Up(1)='1') then
                     next_state_var := s_1_openU;
                elsif (Hall_call_Down(2)='1' or Hall_call_Up(2)='1' or
Car_call(2)='1' or Hall_call_Down(3)='1'or Hall_call_Up(3)='1' or
Car_call(3)='1' or Hall_call_Down(4)='1'or Hall_call_Up(4)='1' or
Car_call(4)='1'or Hall_call_Down(5)='1' or Car_call(5)='1') then
                     next_state_var := s_12_Up;
                elsif (Hall_call_Up(0)='1') then
                     next_state_var := s_10_Down;
                end if;

           when s_1_stopD =>
                if(reset = '1' or Car_call(1)='1' or
Hall_call_Down(1)='1' or Hall_call_Up(1)='1') then
                     next_state_var := s_1_openD;
                elsif (Hall_call_Up(0)='1') then
                     next_state_var := s_10_Down;
                elsif (Hall_call_Down(2)='1' or Hall_call_Up(2)='1' or
Car_call(2)='1' or Hall_call_Down(3)='1'or Hall_call_Up(3)='1' or
Car_call(3)='1' or Hall_call_Down(4)='1'or Hall_call_Up(4)='1' or
Car_call(4)='1'or Hall_call_Down(5)='1' or Car_call(5)='1') then
                     next_state_var := s_12_Up;
                end if;

           when s_1_openU =>
                if(reset = '1') then
                     next_state_var := s_1_openU;
                else
                     if(Passenger_in ='1') then
                          next_state_var := s_1_stopU;
                     end if;
                end if;

           when s_1_openD =>
                if(reset = '1') then
                     next_state_var := s_1_openD;
                else
                     if(Passenger_in ='1') then
                          next_state_var := s_1_stopD;
                     end if;
                end if;

           when s_12_Up =>
                if(Sensor(2)='1' and (Hall_call_Up(2)='1' or
Car_call(2)='1' or reset='1'))  then
                     next_state_var := s_2_stopU;
                elsif(Sensor(2)='1' and (Hall_call_Down(3)='1'or
Hall_call_Up(3)='1' or Car_call(3)='1' or Hall_call_Down(4)='1'or
```

```
Hall_call_Up(4)='1' or Car_call(4)='1' or Hall_call_Down(5)='1' or
Car_call(5)='1' )) then
                    next_state_var := s_23_Up;
               elsif(Sensor(2)='1') then
                    next_state_var := s_2_stopU;
               end if;

          when s_21_Down =>
               if(Sensor(1)='1' and (Hall_call_Down(1)='1' or
Car_call(1)='1' or reset='1'))  then
                    next_state_var := s_1_stopD;
               elsif(Sensor(1)='1' and (Hall_call_Up(0)='1' or
Car_call(0)='1')) then
                    next_state_var := s_10_Down;
               elsif(Sensor(1)='1') then
                    next_state_var := s_1_stopD;
               end if;

          when s_2_stopU =>
               if(reset = '1' or Hall_call_Down(2)='1' or
Hall_call_Up(2)='1' or Car_call(2)='1') then
                    next_state_var := s_2_openU;
               elsif (Hall_call_Down(3)='1'or Hall_call_Up(3)='1' or
Car_call(3)='1' or Hall_call_Down(4)='1'or Hall_call_Up(4)='1' or
Car_call(4)='1' or Hall_call_Down(5)='1' or Car_call(5)='1') then
                    next_state_var := s_23_Up;
               elsif (Hall_call_Up(0)='1' or Car_call(0)='1' or
Hall_call_Down(1)='1'or Hall_call_Up(1)='1' or Car_call(1)='1') then
                    next_state_var := s_21_Down;
               end if;

          when s_2_stopD =>
               if(reset = '1' or Hall_call_Down(2)='1' or
Hall_call_Up(2)='1' or Car_call(2)='1') then
                    next_state_var := s_2_openD;
               elsif (Hall_call_Up(0)='1' or Car_call(0)='1' or
Hall_call_Down(1)='1'or Hall_call_Up(1)='1' or Car_call(1)='1') then
                    next_state_var := s_21_Down;
               elsif (Hall_call_Down(3)='1'or Hall_call_Up(3)='1' or
Car_call(3)='1' or Hall_call_Down(4)='1'or Hall_call_Up(4)='1' or
Car_call(4)='1' or Hall_call_Down(5)='1' or Car_call(5)='1') then
                    next_state_var := s_23_Up;
               end if;

          when s_2_openU =>
               if(reset = '1') then
                    next_state_var := s_2_openU;
               else
                    if(Passenger_in ='1') then
                         next_state_var := s_2_stopU;
                    end if;
               end if;
```

```
            when s_2_openD =>
                  if(reset = '1') then
                        next_state_var := s_2_openD;
                  else
                        if(Passenger_in ='1') then
                              next_state_var := s_2_stopD;
                        end if;
                  end if;

            when s_23_Up =>
                  if(Sensor(3)='1' and (Hall_call_Up(3)='1' or
Car_call(3)='1' or reset='1'))  then
                        next_state_var := s_3_stopU;
                  elsif(Sensor(3)='1' and  (Hall_call_Down(4)='1'or
Hall_call_Up(4)='1' or Car_call(4)='1' or Hall_call_Down(5)='1' or
Car_call(5)='1' )) then
                        next_state_var := s_34_Up;
                  elsif(Sensor(3)='1') then
                        next_state_var := s_3_stopU;
                  end if;

            when s_32_Down =>
                  if(Sensor(2)='1' and (Hall_call_Down(2)='1' or
Car_call(2)='1' or reset='1'))  then
                        next_state_var := s_2_stopD;
                  elsif(Sensor(2)='1' and(Hall_call_Up(0)='1' or
Car_call(0)='1' or Hall_call_Down(1)='1'or Hall_call_Up(1)='1' or
Car_call(1)='1')) then
                        next_state_var := s_21_Down;
                  elsif(Sensor(2)='1') then
                        next_state_var := s_2_stopD;
                  end if;

            when s_3_stopU =>
                  if(reset = '1' or Hall_call_Down(3)='1' or
Hall_call_Up(3)='1' or Car_call(3)='1') then
                        next_state_var := s_3_openU;
                  elsif (Hall_call_Down(4)='1'or Hall_call_Up(4)='1' or
Car_call(4)='1' or Hall_call_Down(5)='1'  or Car_call(5)='1') then
                        next_state_var := s_34_Up;
                  elsif (Hall_call_Up(0)='1' or Car_call(1)='1' or
Hall_call_Down(1)='1' or Hall_call_Up(1)='1' or Car_call(1)='1' or
Hall_call_Down(2)='1' or Hall_call_Up(2)='1' or Car_call(2)='1') then
                        next_state_var := s_32_Down;
                  end if;

            when s_3_stopD =>
                  if(reset = '1' or Hall_call_Down(3)='1' or
Hall_call_Up(3)='1' or Car_call(3)='1') then
                        next_state_var := s_3_openU;
```

```
                elsif (Hall_call_Up(0)='1' or Car_call(1)='1' or
Hall_call_Down(1)='1' or Hall_call_Up(1)='1' or Car_call(1)='1' or
Hall_call_Down(2)='1' or Hall_call_Up(2)='1' or Car_call(2)='1') then
                      next_state_var := s_32_Down;
                elsif (Hall_call_Down(4)='1'or Hall_call_Up(4)='1' or
Car_call(4)='1' or Hall_call_Down(5)='1'  or Car_call(5)='1') then
                      next_state_var := s_34_Up;
                end if;

           when s_3_openU =>
                if(reset = '1') then
                      next_state_var := s_3_openU;
                else
                      if(Passenger_in ='1') then
                            next_state_var := s_3_stopU;
                      end if;
                end if;

           when s_3_openD =>
                if(reset = '1') then
                      next_state_var := s_3_openD;
                else
                      if(Passenger_in ='1') then
                            next_state_var := s_3_stopD;
                      end if;
                end if;

           when s_34_Up =>
                if(Sensor(4)='1' and (Hall_call_Up(4)='1' or
Car_call(4)='1' or reset='1'))  then
                      next_state_var := s_4_stopU;
                elsif(Sensor(4)='1' and (Hall_call_Down(5)='1' or
Car_call(5)='1')) then
                      next_state_var := s_45_Up;
                elsif(Sensor(4)='1') then
                      next_state_var := s_4_stopU;
                end if;

           when s_43_Down =>
                if(Sensor(3)='1' and (Hall_call_Down(3)='1' or
Car_call(3)='1' or reset='1'))  then
                      next_state_var := s_3_stopD;
                elsif(Sensor(3)='1' and (Hall_call_Up(0)='1' or
Car_call(0)='1' or Hall_call_Down(1)='1'or Hall_call_Up(1)='1' or
Car_call(1)='1' or Hall_call_Down(2)='1' or Hall_call_Up(2)='1' or
Car_call(2)='1')) then
                      next_state_var := s_32_Down;
                elsif(Sensor(3)='1') then
                      next_state_var := s_3_stopD;
                end if;

           when s_4_stopU =>
```

```
                if(reset = '1' or Hall_call_Down(4)='1' or
Hall_call_Up(4)='1' or Car_call(4)='1') then
                      next_state_var := s_4_openU;
                elsif (Hall_call_Down(5)='1' or Car_call(5)='1') then
                      next_state_var := s_45_Up;
                elsif (Hall_call_Up(0)='1' or Car_call(0)='1'or
Hall_call_Down(1)='1'or Hall_call_Up(1)='1' or Car_call(1)='1' or
Hall_call_Down(2)='1' or Hall_call_Up(2)='1' or Car_call(2)='1'or
Hall_call_Down(3)='1' or Hall_call_Up(3)='1' or Car_call(3)='1') then
                      next_state_var := s_43_Down;
                end if;

           when s_4_stopD =>
                if(reset = '1' or Hall_call_Down(4)='1' or
Hall_call_Up(4)='1' or Car_call(4)='1') then
                      next_state_var := s_4_openD;
                elsif (Hall_call_Up(0)='1' or Car_call(0)='1'or
Hall_call_Down(1)='1'or Hall_call_Up(1)='1' or Car_call(1)='1' or
Hall_call_Down(2)='1' or Hall_call_Up(2)='1' or Car_call(2)='1'or
Hall_call_Down(3)='1' or Hall_call_Up(3)='1' or Car_call(3)='1') then
                      next_state_var := s_43_Down;
                elsif (Hall_call_Down(5)='1' or Car_call(5)='1') then
                      next_state_var := s_45_Up;
                end if;

           when s_4_openU =>
                if(reset = '1') then
                      next_state_var := s_4_openU;
                else
                      if(Passenger_in ='1') then
                            next_state_var := s_4_stopU;
                      end if;
                end if;

           when s_4_openD =>
                if(reset = '1') then
                      next_state_var := s_4_openD;
                else
                      if(Passenger_in ='1') then
                            next_state_var := s_4_stopD;
                      end if;
                end if;

           when s_45_Up =>
                if(Sensor(5)='1') then
                      next_state_var := s_5_stop;
                end if;

           when s_54_Down =>
                if(Sensor(4)='1' and (Hall_call_Down(4)='1' or
Car_call(4)='1' or reset='1'))  then
                      next_state_var := s_4_stopD;
```

```
                elsif(Sensor(4)='1' and (Hall_call_Up(0)='1' or
Car_call(0)='1' or Hall_call_Down(1)='1'or Hall_call_Up(1)='1' or
Car_call(1)='1' or Hall_call_Down(2)='1' or Hall_call_Up(2)='1' or
Car_call(2)='1' or Hall_call_Down(3)='1' or Hall_call_Up(3)='1' or
Car_call(3)='1' )) then
                     next_state_var := s_43_Down;
                elsif(Sensor(4)='1') then
                     next_state_var := s_4_stopD;
                end if;

           when s_5_stop =>
                if(reset = '1' or Hall_call_Down(5)='1' or
Car_call(5)='1') then
                     next_state_var := s_5_open;
                elsif (Hall_call_Up(0)='1' or Car_call(0)='1' or
Hall_call_Down(1)='1'or Hall_call_Up(1)='1' or Car_call(1)='1' or
Hall_call_Down(2)='1' or Hall_call_Up(2)='1' or Car_call(2)='1' or
Hall_call_Down(3)='1' or Hall_call_Up(3)='1' or Car_call(3)='1' or
Hall_call_Down(4)='1' or Hall_call_Up(4)='1' or Car_call(4)='1') then
                     next_state_var := s_54_Down;
                end if;

           when s_5_open =>
                if(reset = '1') then
                     next_state_var := s_5_open;
                else
                     if(Passenger_in ='1') then
                          next_state_var := s_5_stop;
                     end if;
                end if;

        when others => null;
      end case;

    if(clk'event and clk = '1') then
      state_signal <= next_state_var;
    end if;

    end process next_state;


    -- purpose: output process
    -- type   : combinational
    -- inputs : the sensitivity list
    -- outputs: the outputs of the Control path.
    output_process: process(state_signal)
    begin  -- process output_process
      -- hot conditions indicated later

      case state_signal is
        when s_0_stop=>
                MoveUp <= '0';
```

```
        MoveDown <= '0';
        OpenDoor <= '0';
        CloseDoor <= '1';
        Stop <='1';
        LCD<= 0;

when s_0_open =>
        MoveUp <= '0';
        MoveDown <= '0';
        OpenDoor <= '1';
        CloseDoor <= '0';
        Stop <='1';
        LCD<= 0;

when s_01_Up =>
        MoveUp <= '1';
        MoveDown <= '0';
        OpenDoor <= '0';
        CloseDoor <= '1';
        Stop <='0';
        LCD<= 0;

   when s_10_Down=>
        MoveUp <= '0';
        MoveDown <= '1';
        OpenDoor <= '0';
        CloseDoor <= '1';
        Stop <='0';
        LCD<= 1;

   when s_1_stopU=>
        MoveUp <= '0';
        MoveDown <= '0';
        OpenDoor <= '0';
        CloseDoor <= '1';
        Stop <='1';
        LCD<= 1;

   when s_1_stopD=>
        MoveUp <= '0';
        MoveDown <= '0';
        OpenDoor <= '0';
        CloseDoor <= '1';
        Stop <='1';
        LCD<= 1;

when s_1_openU =>
        MoveUp <= '0';
        MoveDown <= '0';
        OpenDoor <= '1';
        CloseDoor <= '0';
        Stop <='1';
```

```
            LCD<= 1;

      when s_1_openD =>
            MoveUp <= '0';
            MoveDown <= '0';
            OpenDoor <= '1';
            CloseDoor <= '0';
            Stop <='1';
            LCD<= 1;

   when s_12_Up =>
            MoveUp <= '1';
            MoveDown <= '0';
            OpenDoor <= '0';
            CloseDoor <= '1';
            Stop <='0';
            LCD<= 1;

      when s_21_Down=>
            MoveUp <= '0';
            MoveDown <= '1';
            OpenDoor <= '0';
            CloseDoor <= '1';
            Stop <='0';
            LCD<= 2;

      when s_2_stopU=>
            MoveUp <= '0';
            MoveDown <= '0';
            OpenDoor <= '0';
            CloseDoor <= '1';
            Stop <='1';
            LCD<= 2;

      when s_2_stopD=>
            MoveUp <= '0';
            MoveDown <= '0';
            OpenDoor <= '0';
            CloseDoor <= '1';
            Stop <='1';
            LCD<= 2;

   when s_2_openU =>
            MoveUp <= '0';
            MoveDown <= '0';
            OpenDoor <= '1';
            CloseDoor <= '0';
            Stop <='1';
            LCD<= 2;

      when s_2_openD =>
            MoveUp <= '0';
```

```
            MoveDown <= '0';
            OpenDoor <= '1';
            CloseDoor <= '0';
            Stop <='1';
            LCD<= 2;

        when s_23_Up =>
            MoveUp <= '1';
            MoveDown <= '0';
            OpenDoor <= '0';
            CloseDoor <= '1';
            Stop <='0';
            LCD<= 2;

        when s_32_Down=>
            MoveUp <= '0';
            MoveDown <= '1';
            OpenDoor <= '0';
            CloseDoor <= '1';
            Stop <='0';
            LCD<= 3;

        when s_3_stopU=>
            MoveUp <= '0';
            MoveDown <= '0';
            OpenDoor <= '0';
            CloseDoor <= '1';
            Stop <='1';
            LCD<= 3;

        when s_3_stopD=>
            MoveUp <= '0';
            MoveDown <= '0';
            OpenDoor <= '0';
            CloseDoor <= '1';
            Stop <='1';
            LCD<= 3;

        when s_3_openU =>
            MoveUp <= '0';
            MoveDown <= '0';
            OpenDoor <= '1';
            CloseDoor <= '0';
            Stop <='1';
            LCD<= 3;

        when s_3_openD =>
            MoveUp <= '0';
            MoveDown <= '0';
            OpenDoor <= '1';
            CloseDoor <= '0';
            Stop <='1';
```

```
        LCD<= 3;

when s_34_Up =>
        MoveUp <= '1';
        MoveDown <= '0';
        OpenDoor <= '0';
        CloseDoor <= '1';
        Stop <='0';
        LCD<= 3;

   when s_43_Down=>
        MoveUp <= '0';
        MoveDown <= '1';
        OpenDoor <= '0';
        CloseDoor <= '1';
        Stop <='0';
        LCD<= 4;

   when s_4_stopU=>
        MoveUp <= '0';
        MoveDown <= '0';
        OpenDoor <= '0';
        CloseDoor <= '1';
        Stop <='1';
        LCD<= 4;

   when s_4_stopD=>
        MoveUp <= '0';
        MoveDown <= '0';
        OpenDoor <= '0';
        CloseDoor <= '1';
        Stop <='1';
        LCD<= 4;

when s_4_openU =>
        MoveUp <= '0';
        MoveDown <= '0';
        OpenDoor <= '1';
        CloseDoor <= '0';
        Stop <='1';
        LCD<= 4;

   when s_4_openD =>
        MoveUp <= '0';
        MoveDown <= '0';
        OpenDoor <= '1';
        CloseDoor <= '0';
        Stop <='1';
        LCD<= 4;

when s_45_Up =>
        MoveUp <= '1';
```

```
                MoveDown <= '0';
                OpenDoor <= '0';
                CloseDoor <= '1';
                Stop <='0';
                LCD<= 4;

           when s_54_Down=>
                MoveUp <= '0';
                MoveDown <= '1';
                OpenDoor <= '0';
                CloseDoor <= '1';
                Stop <='0';
                LCD<= 5;

           when s_5_stop=>
                MoveUp <= '0';
                MoveDown <= '0';
                OpenDoor <= '0';
                CloseDoor <= '1';
                Stop <='1';
                LCD<= 5;

       when s_5_open =>
                MoveUp <= '0';
                MoveDown <= '0';
                OpenDoor <= '1';
                CloseDoor <= '0';
                Stop <='1';
                LCD<= 5;

       when others =>null;
     end case;
   end process output_process;
 end behave ;
```

Lift Register

-- Register users request

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

 entity  Lift_reg is
 port(clk: in std_logic;
       Car_call_signal: in std_logic_vector(5 downto 0);
       Hall_call_Up_signal: in std_logic_vector(4 downto 0);
       Hall_call_Down_signal: in std_logic_vector(5 downto 1);
       Sensor: in std_logic_vector(5 downto 0);
       OpenDoor: in std_logic;
```

```
       Hall_call_Up: out std_logic_vector(4 downto 0);
       Hall_call_Down: out std_logic_vector(5 downto 1);
       Car_call: out std_logic_vector(5 downto 0)
      );
end Lift_reg;
     architecture behav of Lift_reg is
      signal Hall_call_Down_temp: std_logic_vector(5 downto
1):="00000";
      signal Hall_call_Up_temp: std_logic_vector(4 downto 0):="00000";
      signal Car_call_temp: std_logic_vector(5 downto 0):="000000";

      begin

     process(OpenDoor,clk,Hall_call_Down_signal,Hall_call_Up_signal,Se
nsor,Car_call_signal)
          begin
          if (clk = '1' and clk'event) then
               Hall_call_Up_temp <= Hall_call_Up_signal or
Hall_call_Up_temp;
               Hall_call_Down_temp<= Hall_call_Down_signal or
Hall_call_Down_temp;
               Car_call_temp<=Car_call_signal or Car_call_temp;
               if(OpenDoor = '1') then
                    if(Sensor(0) = '1') then
                         Hall_call_Up_temp(0)<='0';
                         Car_call_temp(0)<='0';
                    elsif(Sensor(1) = '1') then
                         Hall_call_Up_temp(1)<='0';
                         Hall_call_Down_temp(1)<='0';
                         Car_call_temp(1)<='0';
                    elsif(Sensor(2) = '1') then
                         Hall_call_Up_temp(2)<='0';
                         Hall_call_Down_temp(2)<='0';
                         Car_call_temp(2)<='0';
                    elsif(Sensor(3) = '1') then
                         Hall_call_Up_temp(3)<='0';
                         Hall_call_Down_temp(3)<='0';
                         Car_call_temp(3)<='0';
                    elsif(Sensor(4) = '1') then
                         Hall_call_Up_temp(4)<='0';
                         Hall_call_Down_temp(4)<='0';
                         Car_call_temp(4)<='0';
                    elsif(Sensor(5) = '1') then
                         Hall_call_Down_temp(5)<='0';
                         Car_call_temp(5)<='0';
                    end if;
               end if;
          end if;
          end process;
          Hall_call_Down<=Hall_call_Down_temp;
          Hall_call_Up<=Hall_call_Up_temp;
          Car_call<=Car_call_temp;
```

```
      end behav;
```

Overall Circuit

--final circuit

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

 entity  Lift_overall is
 port(clk: in std_logic;
       reset: in std_logic;
       Car_call_signal: in std_logic_vector(5 downto 0);
       Hall_call_Up_signal: in std_logic_vector(4 downto 0);
       Hall_call_Down_signal: in std_logic_vector(5 downto 1);
       MoveUp: out std_logic;
       MoveDown: out std_logic;
       OpenDoor: out std_logic;
       CloseDoor: out std_logic;
       Stop: out std_logic;
       LCD: out integer;
       LED_HALL_UP: out std_logic_vector(4 downto 0);
       LED_HALL_DOWN: out std_logic_vector(5 downto 1);
       LED_Car: out std_logic_vector(5 downto 0)
      );
end Lift_overall;

     architecture ST of Lift_overall is

       signal Car_call_temp: std_logic_vector(5 downto 0);
       signal Hall_call_Up_temp: std_logic_vector(4 downto 0);
       signal Hall_call_Down_temp: std_logic_vector(5 downto 1);
       signal OpenDoor_temp: std_logic;
       signal CloseDoor_temp: std_logic;
       signal Stop_temp: std_logic;
       signal Sensor: std_logic_vector(5 downto 0);
       signal MoveDown_temp: std_logic;
       signal MoveUp_temp: std_logic;
       signal Passenger_in_temp: std_logic;

       component  Lift_controller
        port(clk: in std_logic;
        reset: in std_logic;
        Sensor: in std_logic_vector(5 downto 0);
        Car_call: in std_logic_vector(5 downto 0);
        Hall_call_Up: in std_logic_vector(4 downto 0);
        Hall_call_Down: in std_logic_vector(5 downto 1);
        Passenger_in: in std_logic;
        MoveUp: out std_logic;
        MoveDown: out std_logic;
```

```
        OpenDoor: out std_logic;
        CloseDoor: out std_logic;
        Stop: out std_logic;
        LCD: out integer
       );
       end component;

        component  Lift_reg
         port(clk: in std_logic;
        Car_call_signal: in std_logic_vector(5 downto 0);
        Hall_call_Up_signal: in std_logic_vector(4 downto 0);
        Hall_call_Down_signal: in std_logic_vector(5 downto 1);
        Sensor: in std_logic_vector(5 downto 0);
        OpenDoor: in std_logic;
        Hall_call_Up: out std_logic_vector(4 downto 0);
        Hall_call_Down: out std_logic_vector(5 downto 1);
        Car_call: out std_logic_vector(5 downto 0)
       );
        end component;

        component  Lift_Model
        port(clk: in std_logic;
        MoveUp: in std_logic;
        MoveDown: in std_logic;
        OpenDoor: in std_logic;
        CloseDoor: in std_logic;
        Stop: in std_logic;
        Sensor: out std_logic_vector(5 downto 0);
        Passenger_in: out std_logic
       );
            end component;

       begin
            Lift_controller_0 : Lift_controller port map(clk=>clk,
reset=>reset,Sensor=>Sensor,Car_call=>Car_call_temp,Hall_call_Up=>Hall
_call_Up_temp,Hall_call_Down=>Hall_call_Down_temp,Passenger_in=>Passen
ger_in_temp,MoveUp=>MoveUp_temp,MoveDown=>MoveDown_temp,OpenDoor=>Open
Door_temp,CloseDoor=>CloseDoor_temp,Stop=>Stop_temp,LCD=>LCD);
            Lift_reg_1: Lift_reg port
map(clk=>clk,Car_call_signal=>Car_call_signal,Hall_call_Down_signal=>H
all_call_Down_signal,Hall_call_Up_signal=>Hall_call_Up_signal,Sensor=>
Sensor,OpenDoor=>OpenDoor_temp,Hall_call_Up=>Hall_call_Up_temp,Hall_ca
ll_Down=>Hall_call_Down_temp,Car_call=>Car_call_temp);
            Lift_Model_2: Lift_Model port
map(clk=>clk,MoveUp=>MoveUp_temp,MoveDown=>MoveDown_temp,OpenDoor=>Ope
nDoor_temp,CloseDoor=>CloseDoor_temp,Stop=>Stop_temp,Sensor=>Sensor,Pa
ssenger_in=>Passenger_in_temp);
            OpenDoor<=OpenDoor_temp;
            CloseDoor<=CloseDoor_temp;
            MoveUp<=MoveUp_temp;
            MoveDown<=MoveDown_temp;
            Stop<=Stop_temp;
```

```
           LED_HALL_UP<=Hall_call_Up_temp;
           LED_HALL_DOWN<=Hall_call_Down_temp;
           LED_Car<=Car_call_temp;
 end ST;
```

Overall circuit's test bench

```
library ieee;
use ieee.std_logic_1164.all;
 entity Lift_overall_tb is
 end Lift_overall_tb;

     architecture behav of Lift_overall_tb is

        --  Declaration of the component that will be instantiated.
       component Lift_overall
          port(clk: in std_logic;
         reset: in std_logic;
         Car_call_signal: in std_logic_vector(5 downto 0);
         Hall_call_Up_signal: in std_logic_vector(4 downto 0);
         Hall_call_Down_signal: in std_logic_vector(5 downto 1);
         MoveUp: out std_logic;
         MoveDown: out std_logic;
         OpenDoor: out std_logic;
         CloseDoor: out std_logic;
         Stop: out std_logic;
         LCD: out integer;
         LED_HALL_UP: out std_logic_vector(4 downto 0);
         LED_HALL_DOWN: out std_logic_vector(5 downto 1);
         LED_Car: out std_logic_vector(5 downto 0)
       );
       end component;

         --  Specifies which entity is bound with the component.
         for DUT: Lift_overall use entity work.Lift_overall;
         signal clk: std_logic:='0';
            signal reset: std_logic:='0';
            signal Car_call_signal: std_logic_vector(5 downto
0):="000000";
           signal Hall_call_Up_signal: std_logic_vector(4 downto
0):="00000";
           signal Hall_call_Down_signal: std_logic_vector(5 downto
1):="00000";
           signal MoveUp: std_logic;
           signal MoveDown: std_logic;
           signal OpenDoor: std_logic;
           signal CloseDoor: std_logic;
           signal Stop: std_logic;
           signal LCD: integer;
           signal LED_HALL_UP: std_logic_vector(4 downto 0);
           signal LED_HALL_DOWN: std_logic_vector(5 downto 1);
```

```
            signal LED_Car: std_logic_vector(5 downto 0);


            begin
                  --  Component instantiation.
                  DUT: Lift_overall port map (clk =>
clk,reset=>reset,Car_call_signal=>Car_call_signal,
Hall_call_Down_signal => Hall_call_Down_signal,Hall_call_Up_signal=>
Hall_call_Up_signal,OpenDoor=>OpenDoor,MoveDown=>MoveDown,MoveUp=>Move
Up,CloseDoor=>CloseDoor,Stop=>Stop,LCD=>LCD,LED_HALL_UP=>LED_HALL_UP,L
ED_HALL_DOWN=>LED_HALL_DOWN,LED_Car=>LED_Car);
                  --  This process does the real job.
                  process
                        begin
                        clk<='1';
                        wait for 10 ms;
                        clk<='0';
                        wait for 10 ms;
                  end process;

                  process
                        begin
                        reset<='0';
                        wait for 19000 ms;
                        --reset<='1';
                        --wait for 4000 ms;
                        wait;
                  end process;

                  process
                        begin
                        Hall_call_Up_signal(2)<='1';wait for 600 ms;
Hall_call_Up_signal(2)<='0';
                        wait for 2000 ms;
                        Car_call_signal(4)<='1';wait for 600
ms;Car_call_signal(4)<='0';
                        wait for 800 ms;
                        Hall_call_Down_signal(5)<='1';wait for 600
ms;Hall_call_Down_signal(5)<='0';
                        wait for 3000 ms;
                        Car_call_signal(0)<='1';wait for 600
ms;Car_call_signal(0)<='0'; --new third floor up request
                        wait for 400 ms;
                        Hall_call_Up_signal(3)<='1';wait for 600
ms;Hall_call_Up_signal(3)<='0'; --new second floor request
                        wait;
                  end process;
      end behav;
```