Introduction:

Car Entry Exit barrier system is designed for the automatic entry and exit of car. RED Led is on when no car enters or exits the barrier. As soon the car reaches the barrier sensor detected the metallic object and RED LED is OFF and KEYPAD active is on indicating the user to enter password. After correct password entered car is allowed to move forward. After insertion of correct password barrier is high and GREEN LED is ON. Each time pushbutton (KEY) pressed led glows indicating the key is pressed as soon that LED I off user can press the other pushbutton. In case the wrong password is entered RED LED glows and barrier remains low.

---

---

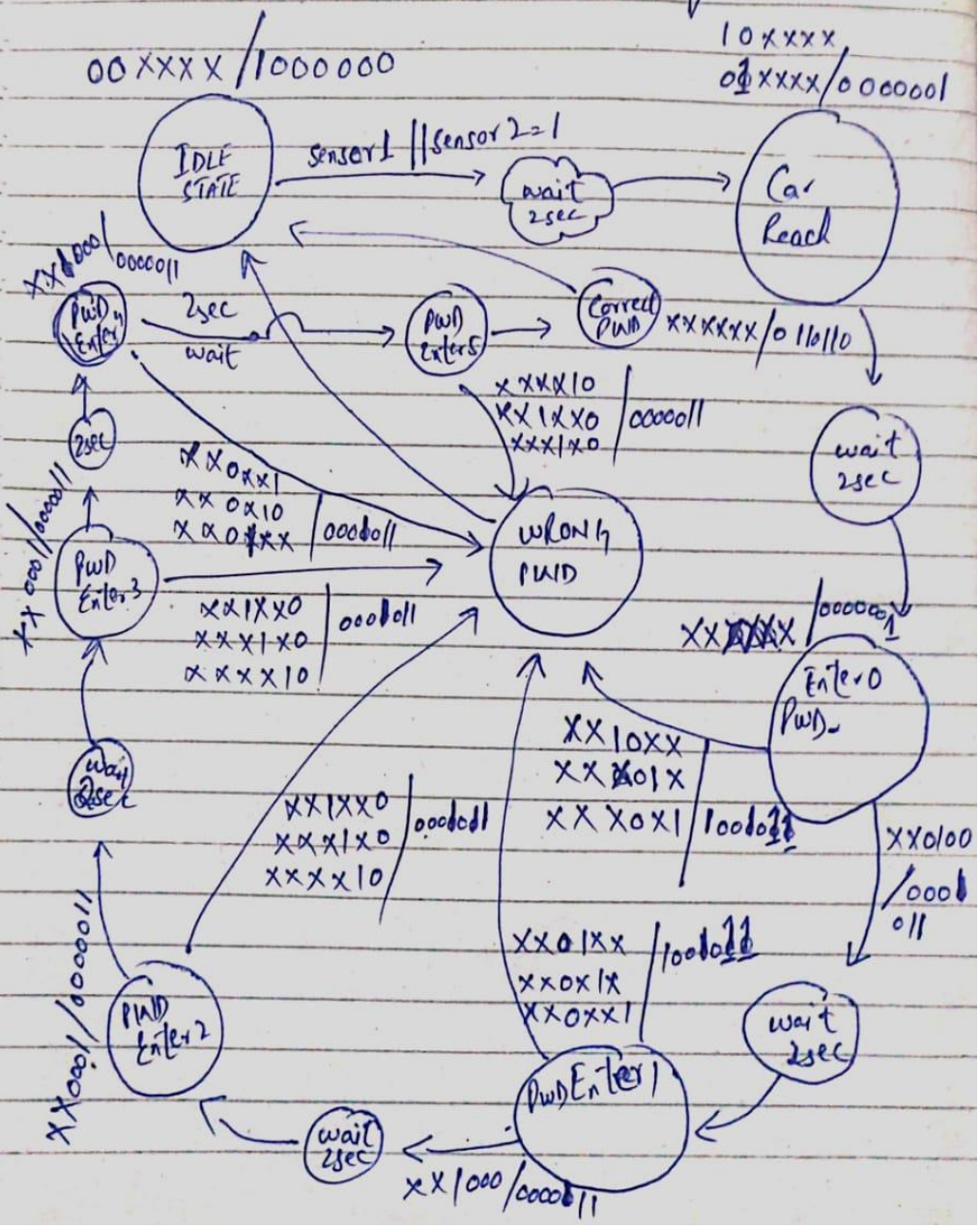
**STATE DIAGRAM:**

Data Representation =

## Inputs / Outputs

Sensor 1 - Sensor 2 - Key 3 - Key 2 - Key 1 - Key 0 / RED LED - GREEN LED

Entry pass - Entry fail
Barries Status - Key press
Active keypad



## VHDL CODE:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;


entity Car_Entry_Top is
port (
CLOCK: in std_logic; -- input CLK
KEY_0,KEY_1,KEY_2,KEY_3: in std_logic; -- input Push buttons
Sensor_1,Sensor_2: in std_logic; -- input Switches for sensors
RED_LED,GREEN_LED: out std_logic:='0'; -- LED for Stop and Go
Entry_pass, Entry_fail: out std_logic:='0'; -- Sequence indication for correct sequence and wrong seq
Barrier_status,Key_press,ACTIVE_KEYPAD: out std_logic:='0' -- Sequence indication for correct sequence and wrong seq
);
end Car_Entry_Top;

architecture Behavioral of Car_Entry_Top is
signal KEY0_PRESS,KEY1_PRESS,KEY2_PRESS,KEY3_PRESS: STD_LOGIC:='0';
signal KEY0_REG,KEY1_REG,KEY2_REG,KEY3_REG: STD_LOGIC_VECTOR(11 downto 0):=X"000";
signal Counter: STD_LOGIC_VECTOR(27 downto 0):=X"0000000";

-- State
type state_type is (IDLE_STATE,CAR_REACH,PWD_ENTER0,PWD_ENTER1,
                                                    PWD_ENTER2,PWD_ENTER3,PWD_ENTER4,PWD_ENTER5,
                                                    CORRECT_PWD,WRONG_PWD, BARRIER_UP,WAIT_STATE );
signal State,Next_State : state_type:=IDLE_STATE;

begin

process (CLOCK,KEY_0,KEY_1,KEY_2,KEY_3,Sensor_1,Sensor_2,State,Next_State,Counter)
begin


if (CLOCK'event and  CLOCK='1') then
KEY0_REG<=KEY0_REG(10 downto 0) & KEY_0;
KEY1_REG<=KEY1_REG(10 downto 0) & KEY_1;
KEY2_REG<=KEY2_REG(10 downto 0) & KEY_2;
KEY3_REG<=KEY3_REG(10 downto 0) & KEY_3;
end if;

if (CLOCK'event and  CLOCK='1') then
if (KEY0_REG=X"0FF")        then    KEY0_PRESS<='1'; else        KEY0_PRESS<='0'; end if;
if (KEY1_REG=X"0FF")        then    KEY1_PRESS<='1'; else        KEY1_PRESS<='0'; end if;
if (KEY2_REG=X"0FF")        then    KEY2_PRESS<='1'; else        KEY2_PRESS<='0'; end if;
if (KEY3_REG=X"0FF")        then    KEY3_PRESS<='1'; else        KEY3_PRESS<='0'; end if;
end if;

if (CLOCK'event and  CLOCK='1') then
case (State) is

when IDLE_STATE=>       --- starting the FSM
                        RED_LED<='1';
                        GREEN_LED<='0';
                        Entry_pass<='0';
                        Entry_fail<='0';
                        Barrier_status<='0';
                        ACTIVE_KEYPAD<='0';
                        Key_press<='0';
                        if (Sensor_1='1' and Sensor_2='1') then
                        Next_State<=IDLE_STATE;
                        elsif (Sensor_1='1') then
                        Next_State<=CAR_REACH;
                        State<=WAIT_STATE;
                        elsif (Sensor_2='1') then
                        Next_State<=CAR_REACH;
                        State<=WAIT_STATE;
```

```
                    else
                    Next_State<=IDLE_STATE;
                    end if;

when CAR_REACH=>
                    RED_LED<='0';
                    GREEN_LED<='0';
                    Entry_pass<='0';
                    Entry_fail<='0';
                    Barrier_status<='0';
                    ACTIVE_KEYPAD<='1';
                    Key_press<='0';
                    Next_State<=PWD_ENTER0;
                    State<=WAIT_STATE;

when PWD_ENTER0=>
                    RED_LED<='0';
                    GREEN_LED<='0';
                    Entry_pass<='0';
                    Entry_fail<='0';
                    Barrier_status<='0';
                    ACTIVE_KEYPAD<='1';
                    if (KEY2_PRESS='1') then
                    Next_State<=PWD_ENTER1;
                    State<=WAIT_STATE;
                    Key_press<='1';
                    elsif (KEY0_PRESS='1' or KEY1_PRESS='1' or KEY3_PRESS='1') then
                    Next_State<=WRONG_PWD;
                    State<=WAIT_STATE;
                    Key_press<='1';
                    else
                    Key_press<='0';
                    Next_State<=PWD_ENTER0;
                    end if;

when PWD_ENTER1=>

                    if (KEY3_PRESS='1') then
                    Next_State<=PWD_ENTER2;
                    State<=WAIT_STATE;
                    Key_press<='1';
                    elsif (KEY0_PRESS='1' or KEY1_PRESS='1' or KEY2_PRESS='1') then
                    Next_State<=WRONG_PWD;
                    State<=WAIT_STATE;
                    Key_press<='1';
                    else
                    Key_press<='0';
                    Next_State<=PWD_ENTER1;
                    end if;

when PWD_ENTER2=>

                    if (KEY0_PRESS='1') then
                    Next_State<=PWD_ENTER3;
                    State<=WAIT_STATE;
                    Key_press<='1';
                    elsif (KEY3_PRESS='1' or KEY1_PRESS='1' or KEY2_PRESS='1') then
                    Next_State<=WRONG_PWD;
                    State<=WAIT_STATE;
                    Key_press<='1';
                    else
                    Key_press<='0';
                    Next_State<=PWD_ENTER2;
                    end if;

when PWD_ENTER3=>

                    if (KEY0_PRESS='1') then
                    Next_State<=PWD_ENTER4;
                    State<=WAIT_STATE;
                    Key_press<='1';
                    elsif (KEY3_PRESS='1' or KEY1_PRESS='1' or KEY2_PRESS='1') then
                    Next_State<=WRONG_PWD;
```

```
                        State<=WAIT_STATE;
                        Key_press<='1';
                        else
                        Key_press<='0';
                        Next_State<=PWD_ENTER3;
                        end if;

when PWD_ENTER4=>

                        if (KEY3_PRESS='1') then
                        Next_State<=PWD_ENTER5;
                        State<=WAIT_STATE;
                        Key_press<='1';
                        elsif (KEY0_PRESS='1' or KEY1_PRESS='1' or KEY2_PRESS='1') then
                        Next_State<=WRONG_PWD;
                        State<=WAIT_STATE;
                        Key_press<='1';
                        else
                        Key_press<='0';
                        Next_State<=PWD_ENTER4;
                        end if;

when PWD_ENTER5=>

                        if (KEY1_PRESS='1') then
                        Next_State<=CORRECT_PWD;
                        State<=WAIT_STATE;
                        Key_press<='1';
                        elsif (KEY3_PRESS='1' or KEY0_PRESS='1' or KEY2_PRESS='1') then
                        Next_State<=WRONG_PWD;
                        State<=WAIT_STATE;
                        Key_press<='1';
                        else
                        Key_press<='0';
                        Next_State<=PWD_ENTER4;
                        end if;

when CORRECT_PWD=>
                        RED_LED<='0';
                        GREEN_LED<='0';
                        Entry_pass<='1';
                        Entry_fail<='0';
                        Barrier_status<='0';
                        ACTIVE_KEYPAD<='0';
                        Key_press<='0';
                        Next_State<=BARRIER_UP;
                        State<=WAIT_STATE;

when WRONG_PWD=>
                        RED_LED<='0';
                        GREEN_LED<='0';
                        Entry_pass<='0';
                        Entry_fail<='1';
                        Barrier_status<='0';
                        ACTIVE_KEYPAD<='0';
                        Key_press<='0';
                        Next_State<=IDLE_STATE;
                        State<=WAIT_STATE;

when BARRIER_UP=>
                        RED_LED<='0';
                        GREEN_LED<='1';
                        Entry_pass<='0';
                        Entry_fail<='0';
                        Barrier_status<='1';
                        ACTIVE_KEYPAD<='0';
                        Key_press<='0';
                        Next_State<=IDLE_STATE;
                        State<=WAIT_STATE;

when WAIT_STATE=>
                        if (Counter<X"5F5E0FF") then -- Hardware
                --      if (Counter<X"FFFF") then --Simulation
```

```
                        Counter<=Counter + '1';
                        else
                        Counter<=X"0000000";
                        State<=Next_State;
                        end if;


when others=>
                        State<=IDLE_STATE;
end case;
end if;
end process;

end Behavioral;
```

---

---

## TEST BENCH:

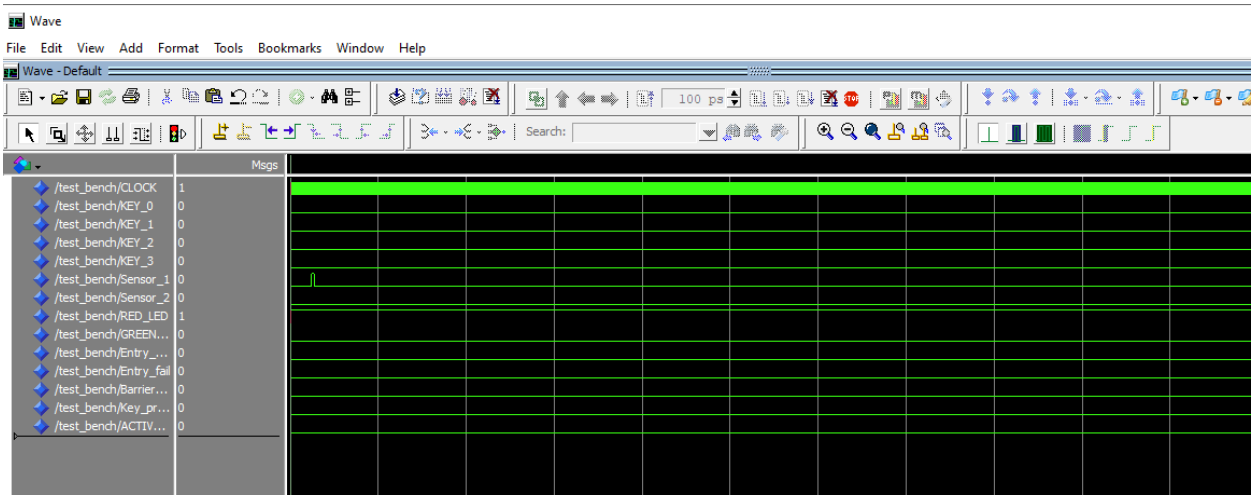
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY Test_bench IS
END Test_bench;

ARCHITECTURE behavior OF Test_bench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT Car_Entry_Top
   PORT(
      CLOCK : IN  std_logic;
      KEY_0 : IN  std_logic;
      KEY_1 : IN  std_logic;
      KEY_2 : IN  std_logic;
      KEY_3 : IN  std_logic;
      Sensor_1 : IN  std_logic;
      Sensor_2 : IN  std_logic;
      RED_LED : OUT  std_logic;
      GREEN_LED : OUT  std_logic;
      Entry_pass : OUT  std_logic;
      Entry_fail : OUT  std_logic;
      Barrier_status : OUT  std_logic;
      Key_press : OUT  std_logic;
      ACTIVE_KEYPAD : OUT  std_logic
     );
   END COMPONENT;


   --Inputs
   signal CLOCK : std_logic := '0';
   signal KEY_0 : std_logic := '0';
   signal KEY_1 : std_logic := '0';
   signal KEY_2 : std_logic := '0';
   signal KEY_3 : std_logic := '0';
   signal Sensor_1 : std_logic := '0';
   signal Sensor_2 : std_logic := '0';

               --Outputs
   signal RED_LED : std_logic;
   signal GREEN_LED : std_logic;
   signal Entry_pass : std_logic;
   signal Entry_fail : std_logic;
   signal Barrier_status : std_logic;
   signal Key_press : std_logic;
   signal ACTIVE_KEYPAD : std_logic;
```

```
  -- Clock period definitions
  constant CLOCK_period : time := 2 ns;

BEGIN

        -- Instantiate the Unit Under Test (UUT)
  uut: Car_Entry_Top PORT MAP (
       CLOCK => CLOCK,
       KEY_0 => KEY_0,
       KEY_1 => KEY_1,
       KEY_2 => KEY_2,
       KEY_3 => KEY_3,
       Sensor_1 => Sensor_1,
       Sensor_2 => Sensor_2,
       RED_LED => RED_LED,
       GREEN_LED => GREEN_LED,
       Entry_pass => Entry_pass,
       Entry_fail => Entry_fail,
       Barrier_status => Barrier_status,
       Key_press => Key_press,
       ACTIVE_KEYPAD => ACTIVE_KEYPAD
      );

  -- Clock process definitions
  CLOCK_process :process
  begin
                        CLOCK <= '0';
                        wait for CLOCK_period/2;
                        CLOCK <= '1';
                        wait for CLOCK_period/2;
  end process;


  -- Stimulus process
  stim_proc: process
  begin
    -- hold reset state for 1000 ns.
    wait for 1000 ns;

    wait for CLOCK_period*100;
                        Sensor_1<='1';
    wait for CLOCK_period*100;
                        Sensor_1<='0';
    wait for CLOCK_period*100000;
    wait for CLOCK_period*100000;
                        Key_1<='1';
    wait for CLOCK_period*100000;
                        Key_1<='0';

    wait for CLOCK_period*100000;
                        Key_2<='1';
    wait for CLOCK_period*100000;
                        Key_2<='0';

    wait for CLOCK_period*1000000;
                        Sensor_1<='1';
    wait for CLOCK_period*100;
                        Sensor_1<='0';
    wait for CLOCK_period*100000;
    wait for CLOCK_period*100000;
                        Key_2<='1';
    wait for CLOCK_period*100000;
                        Key_2<='0';
    wait for CLOCK_period*100000;
                        Key_3<='1';
    wait for CLOCK_period*100000;
                        Key_3<='0';
    wait for CLOCK_period*100000;
                        Key_0<='1';
    wait for CLOCK_period*100000;
                        Key_0<='0';
    wait for CLOCK_period*100000;
```

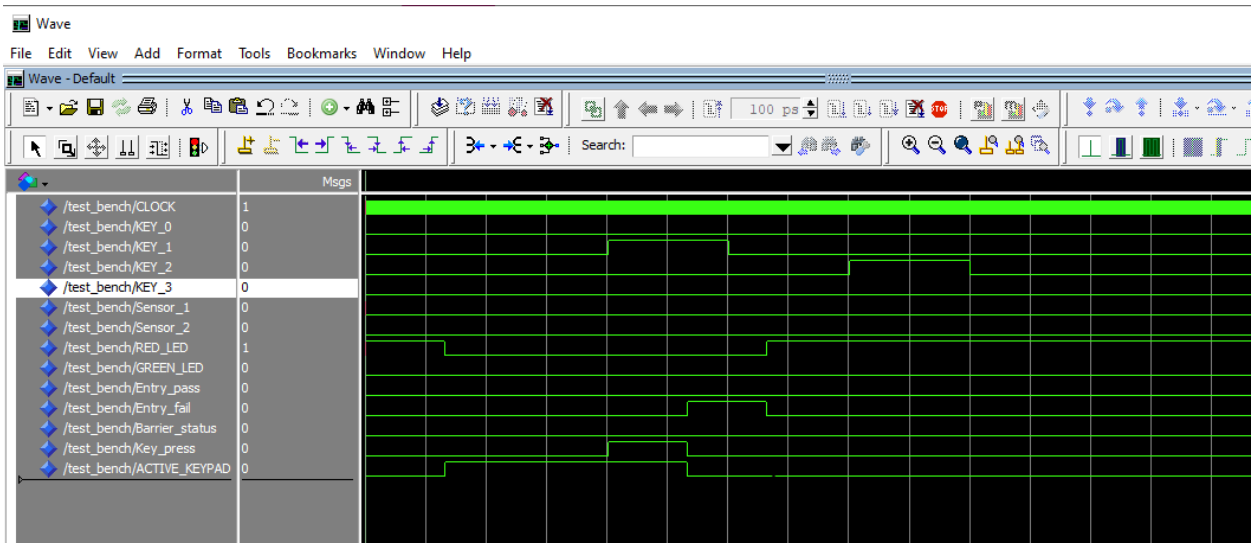
```
                        Key_0<='1';
    wait for CLOCK_period*100000;
                        Key_0<='0';

    wait for CLOCK_period*100000;
                        Key_3<='1';
    wait for CLOCK_period*100000;
                        Key_3<='0';
    wait for CLOCK_period*100000;
                        Key_1<='1';
    wait for CLOCK_period*100000;
                        Key_1<='0';

    wait for CLOCK_period*100;
                        Sensor_2<='1';
    wait for CLOCK_period*100;
                        Sensor_2<='0';
    wait for CLOCK_period*100000;
    wait for CLOCK_period*100000;
                        Key_1<='1';
    wait for CLOCK_period*100000;
                        Key_1<='0';

    wait for CLOCK_period*100000;
                        Key_2<='1';
    wait for CLOCK_period*100000;
                        Key_2<='0';

    wait for CLOCK_period*1000000;
                        Sensor_2<='1';
    wait for CLOCK_period*100;
                        Sensor_2<='0';
    wait for CLOCK_period*100000;
    wait for CLOCK_period*100000;
                        Key_2<='1';
    wait for CLOCK_period*100000;
                        Key_2<='0';
    wait for CLOCK_period*100000;
                        Key_3<='1';
    wait for CLOCK_period*100000;
                        Key_3<='0';
    wait for CLOCK_period*100000;
                        Key_0<='1';
    wait for CLOCK_period*100000;
                        Key_0<='0';
    wait for CLOCK_period*100000;
                        Key_0<='1';
    wait for CLOCK_period*100000;
                        Key_0<='0';

    wait for CLOCK_period*100000;
                        Key_3<='1';
    wait for CLOCK_period*100000;
                        Key_3<='0';
    wait for CLOCK_period*100000;
                        Key_1<='1';
    wait for CLOCK_period*100000;
                        Key_1<='0';

    -- insert stimulus here

    wait;
  end process;

END;
```

**Simulation Waveform:**

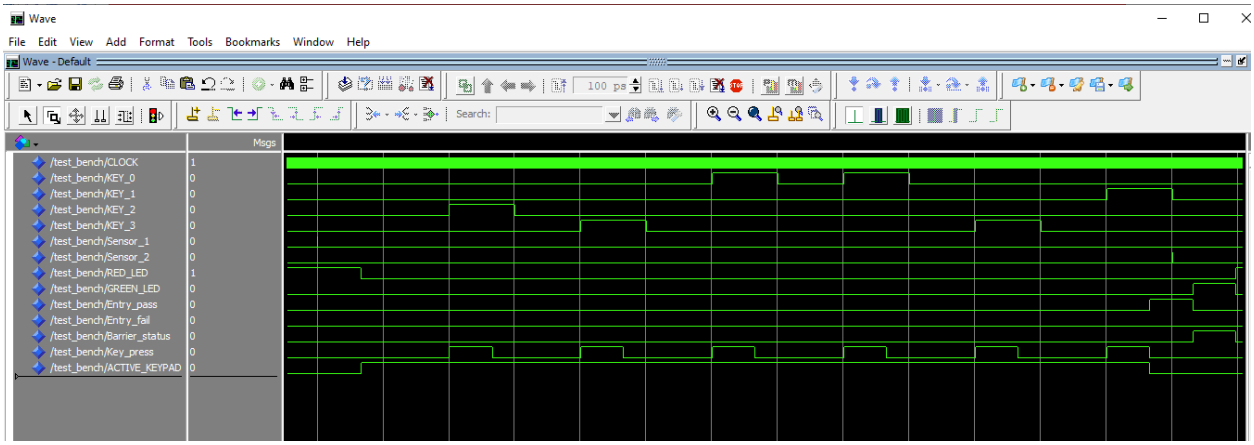
Sensor 1 is high indicating the Car reached the barrier for entrance as shown in figure below. RED LED is initially high after some delay it's OFF.




After RED LED is off and KEYPAD active is on. Driver can insert the password for car entrance. As the password sequence is Key2-Key3-Key0-Key0-Key3-Key1 but as shown in figure below wrong key is pressed hence fail LED glows and RED LED always glow indicating the car to stop.




Now again the sensor 1 got high means another car is ready to enter. Here Correct sequence is being inserted hence GREEN LED is ON indicating the car to GO and barrier is also high.




Complete Simulation waveform: