STATE MACHINE
MEELAY MACHINE

SEQUENCE DETECTOR(overlapping)

CODE:

Main code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity SequencedDetector is
Port (
clck: std_logic;
rst : std_logic;
din : in std_logic;
out_1: out std_logic
 );
end SequencedDetector;

architecture Behavioral of SequencedDetector is
type state_type is (start,s0,s1,s2);
signal prev_state,next_state,state: state_type;
signal temp_out : std_logic;
begin
p1: process(clck)
begin
if(rising_edge(clck)) then
  if(rst = '1') then
     state <= start;
  else
    state <= next_state;
  end if;
end if;
end process p1;
p2: process(state,din)
begin
case(state) is
when s0=>
temp_out <= '0';
   if(din = '1') then
   next_state <= s1;
   else
    next_state <= s0;
    end if;
when s1=>
temp_out <= '0';
```

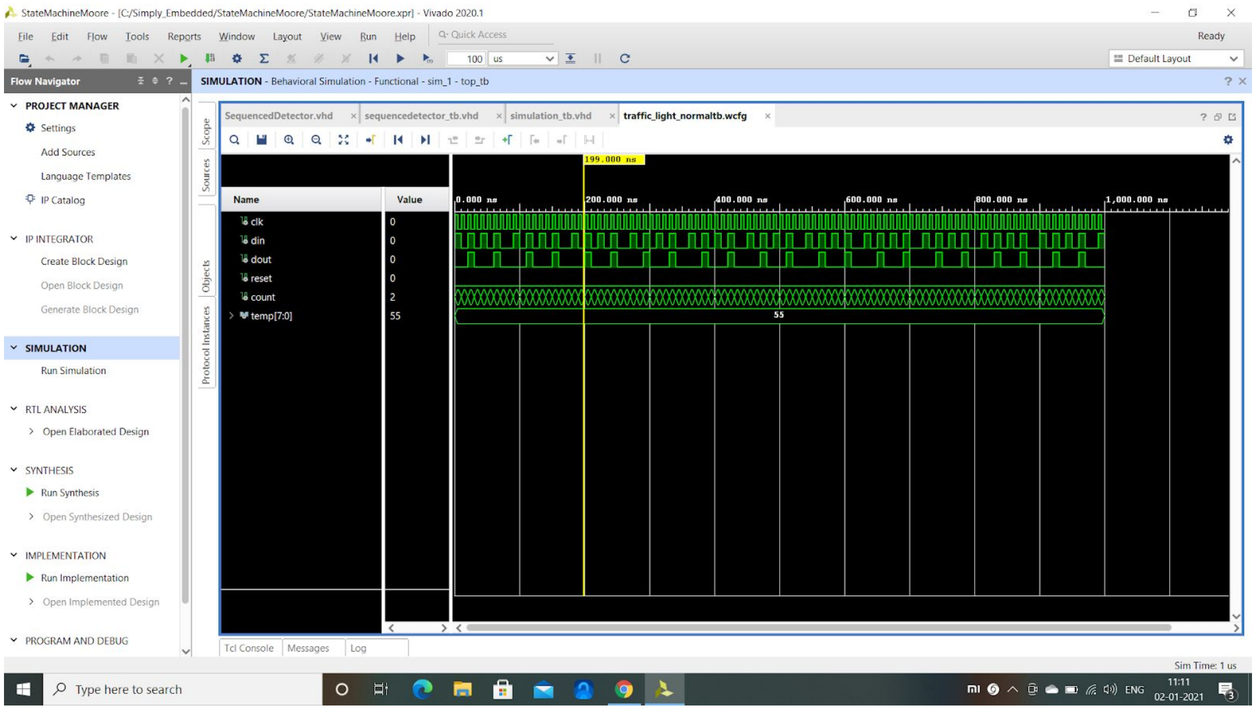
```
   if(din = '0') then
   next_state <= s2;
   else
    next_state <= s1;
    end if;
when s2=>
   if(din = '1') then
   next_state <= s0;
   temp_out <= '1';
   else
    next_state <= s0;
    temp_out <= '0';
    end if;
when others =>
   next_state <= s0;
end case;
end process p2;
out_1 <= temp_out;
end Behavioral;
```

Test bench:

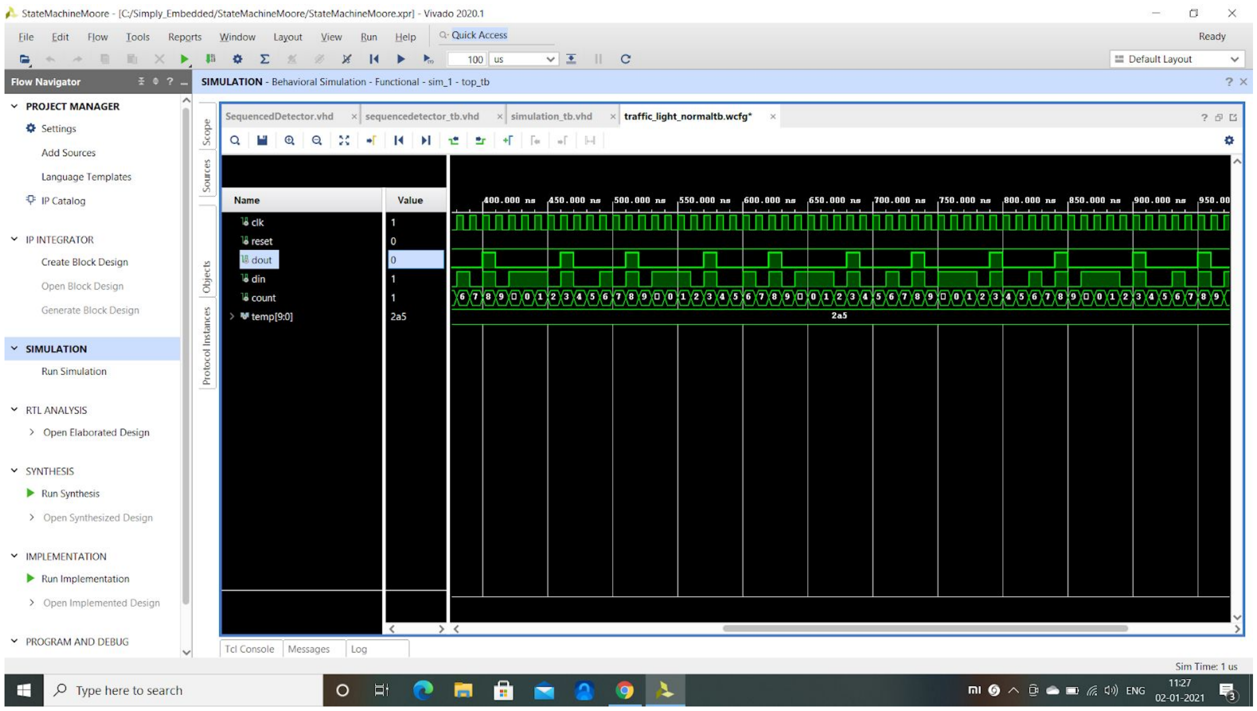
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use ieee.std_logic_unsigned.all;

entity top_tb is
end top_tb;

architecture Behavioral of top_tb is
component SequencedDetector is
Port (
clck: std_logic;
rst : std_logic;
din : in std_logic;
out_1: out std_logic
 );
end  component;

signal    clk : STD_LOGIC:= '0';
signal    din : std_logic :='0';
signal    dout : std_logic;
signal    reset : std_logic := '0';
signal temp : std_logic_vector(7 downto 0) := "01010101";
signal count : integer range 0 to 10 := 0;
begin

T1 : SequencedDetector port map(clck => clk,din => din,out_1 => dout,rst=> reset);

p1 : process
begin
clk <= '1';
wait for 5 ns;
clk <= '0';
wait for 5ns;
end process;

p2 : process(clk)
begin
if(rising_edge(clk)) then
   if(count < 8) then
     din <= temp(count);
     count <= count + 1;
   else
      count <= 0;
   end if;
end if;
end process;

end Behavioral;
```

WAVEFORM:

a) **Sequence : "01010101"**



b) **Sequence : 1010100101**



SEQUENCE DETECTOR (non-overlapping)

CODE:

Main code:

library IEEE;

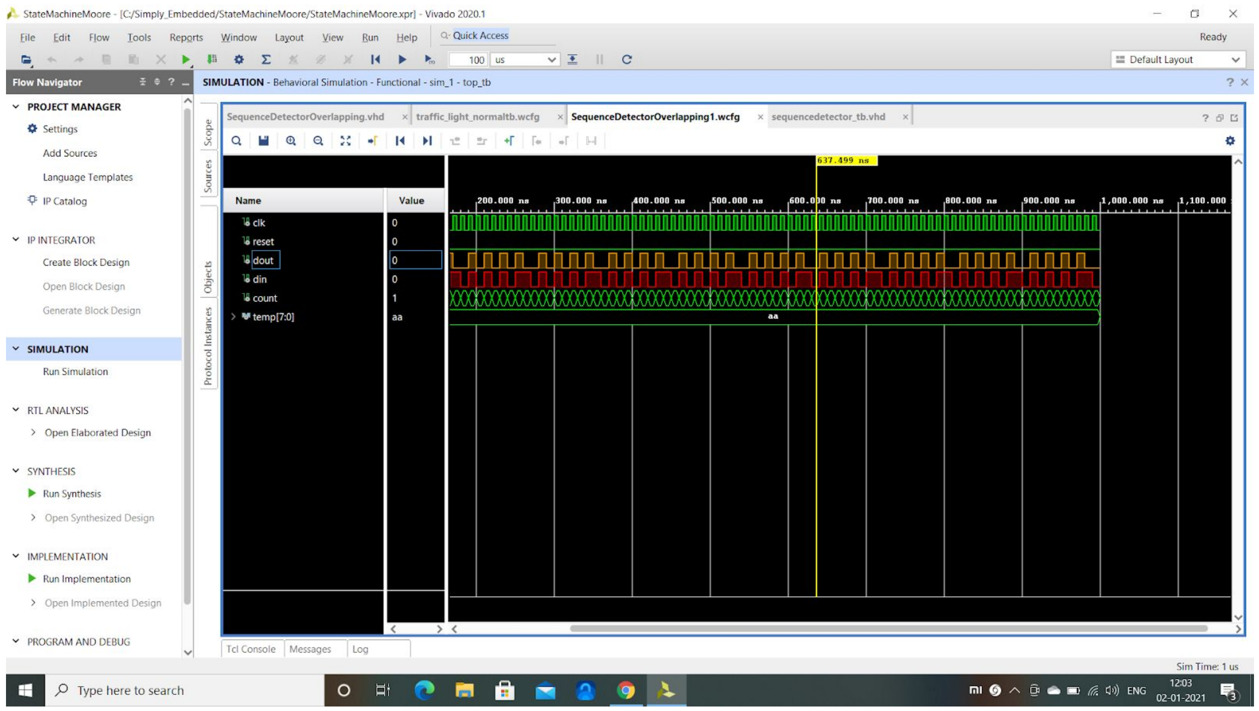
use IEEE.STD_LOGIC_1164.ALL;

```
entity SequencedDetectorOverlapping is
Port (
clck: std_logic;
rst : std_logic;
din : in std_logic;
out_1: out std_logic
 );
end SequencedDetectorOverlapping;

architecture Behavioral of SequencedDetectorOverlapping is
type state_type is (start,s0,s1,s2);
signal prev_state,next_state,state: state_type;
signal temp_out,prev_value : std_logic;
begin
p1: process(clck)
begin
if(rising_edge(clck)) then
  if(rst = '1') then
     state <= start;
  else
    state <= next_state;
  end if;
end if;
end process p1;
p2: process(state,din)
begin
case(state) is
when s0=>
temp_out <= '0';
   if((din = '1')) then
   next_state <= s1;
   else
    next_state <= s0;
    end if;
when s1=>
temp_out <= '0';
   if(din = '0') then
   next_state <= s2;
   else
    next_state <= s1;
    end if;
when s2=>
   if(din = '1') then
  next_state <= s1;
  temp_out <= '1';
  else
   next_state <= s0;
   temp_out <= '0';
   end if;
when others =>
  next_state <= s0;
end case;
end process p2;
out_1 <= temp_out;
end Behavioral;
```

Test bench code:

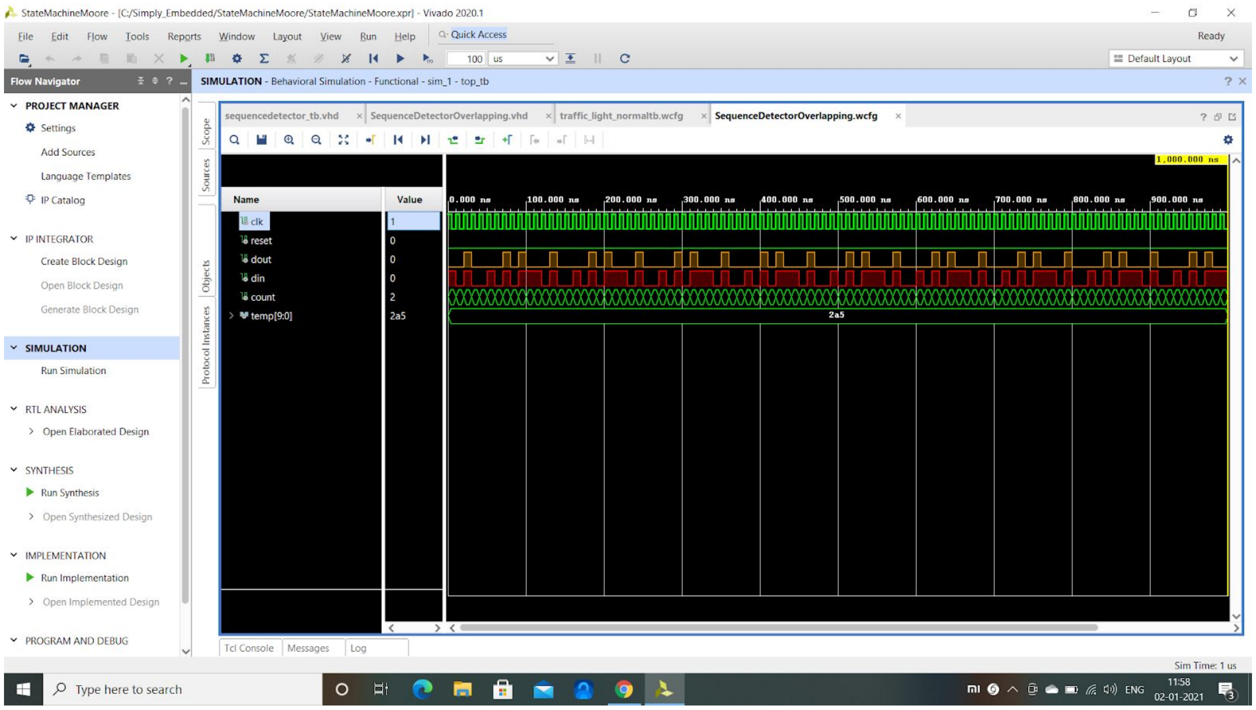
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use ieee.std_logic_unsigned.all;

entity top_tb is
end top_tb;

architecture Behavioral of top_tb is
component SequencedDetectorOverlapping is
Port (
clck: std_logic;
rst : std_logic;
din : in std_logic;
out_1: out std_logic
 );
end  component;

signal    clk : STD_LOGIC:= '0';
signal    din : std_logic :='0';
signal    dout : std_logic;
signal    reset : std_logic := '0';



signal temp : std_logic_vector(8 downto 0) := "101010010";
signal count : integer range 0 to 12 := 0;
begin
```

```
T1 : SequencedDetectorOverlapping port map(clck => clk,din => din,out_1 => dout,rst=> reset);

p1 : process
begin
clk <= '1';
wait for 5 ns;
clk <= '0';
wait for 5ns;
end process;

p2 : process(clk)
begin
if(rising_edge(clk)) then
   if(count < 9) then
     din <= temp(count);
     count <= count + 1;
   else
      count <= 0;
   end if;
end if;
end process;

end Behavioral;
```


StateMachineMoore - [C:/Simply_Embedded/StateMachineMoore/StateMachineMoore.xpr] - Vivado 2020.1

SIMULATION - Behavioral Simulation - Functional - sim_1 - top_tb

Name	Value
clk	0
reset	0
dout	0
din	0
count	1
temp[7:0]	aa



Name	Value
clk	1
reset	0
dout	0
din	0
count	2
temp[9:0]	2a5

WAVEFORM: