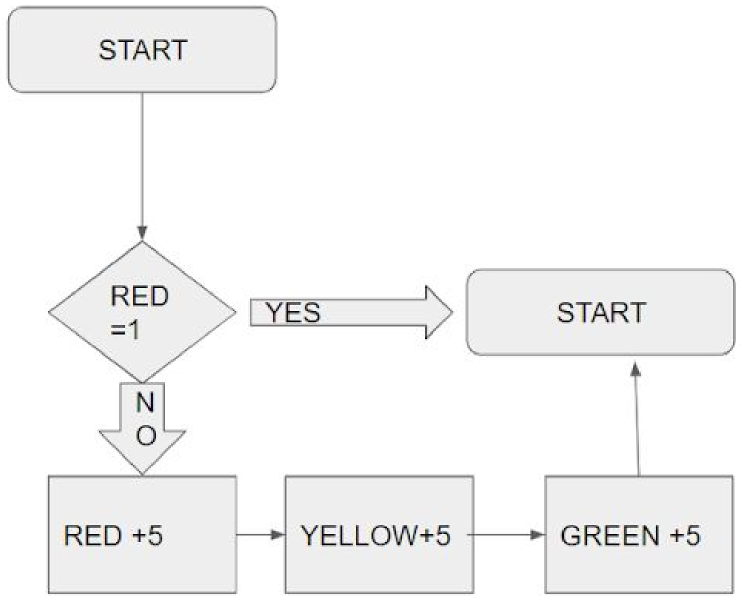
State machine

Moore machine:

Block diagram:



1) **Basic model**

CODE:

Main code:

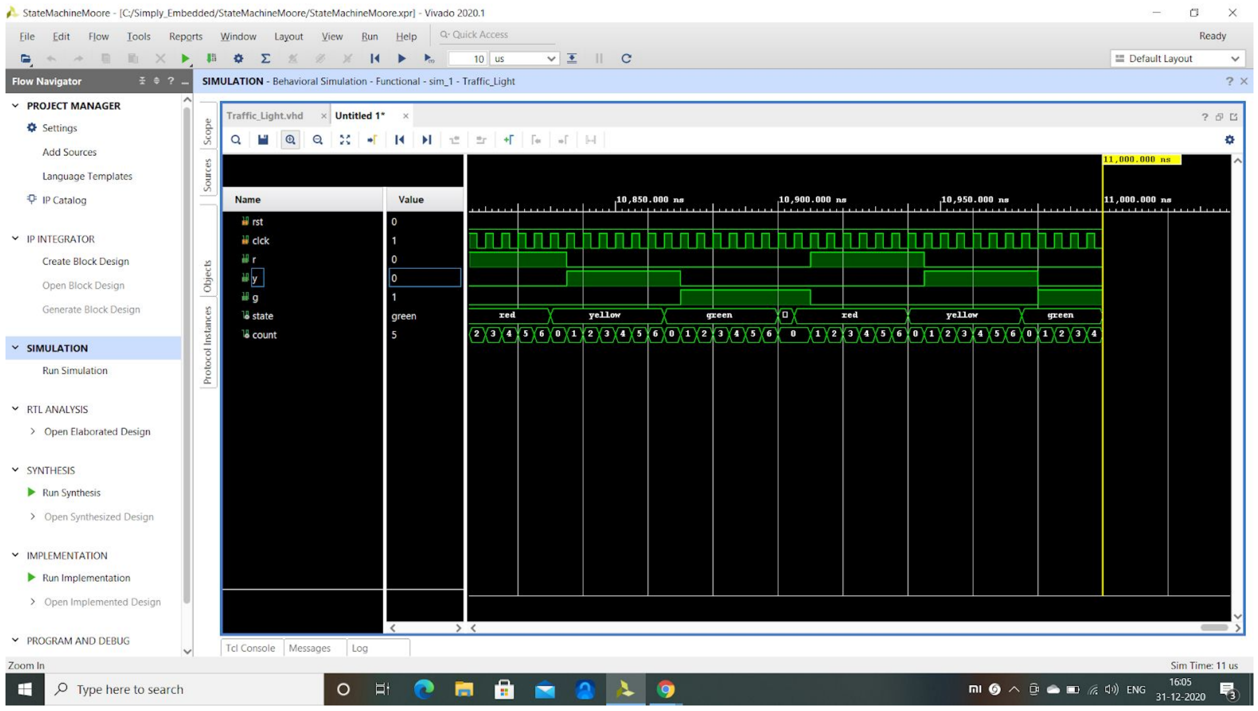
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Traffic_Light is
Port (
rst : in std_logic;
clck : in std_logic;
r,g,y : out std_logic
 );
end Traffic_Light;

architecture Behavioral of Traffic_Light is
type state_type is (init,red,green,yellow);
signal state : state_type;
signal count : integer range 0 to 6 := 0;
begin
process(clck) -- process is sensitive to clock
```

```
begin
if(rising_edge(clck)) then
if(rst = '1') then
   state <= init;
   r <= '0';
   g <= '0';
   y <= '0';
else
case(state) is
when init =>
   state <= red;
   count <= 0;
when red =>
  r <= '1';
  g <= '0';
  y <= '0';
  if(count <= 5) then
   state <= red;
   count <= count + 1;
  else
    state <= yellow;
    count <= 0;
  end if;
when yellow =>
  r <= '0';
  y <= '1';
  g <= '0';
  if(count <= 5) then
   state <= yellow;
   count <= count + 1;
  else
    state <= green;
    count <= 0;
  end if;
when green =>
  r <= '0';
  y <= '0';
  g <= '1';
  if(count <= 5) then
   state <= green;
   count <= count + 1;
  else
    state <= init;
    count <= 0;
  end if;
when others =>
     state <= red;
end case;
end if;
end if;
end process;
end Behavioral;
```

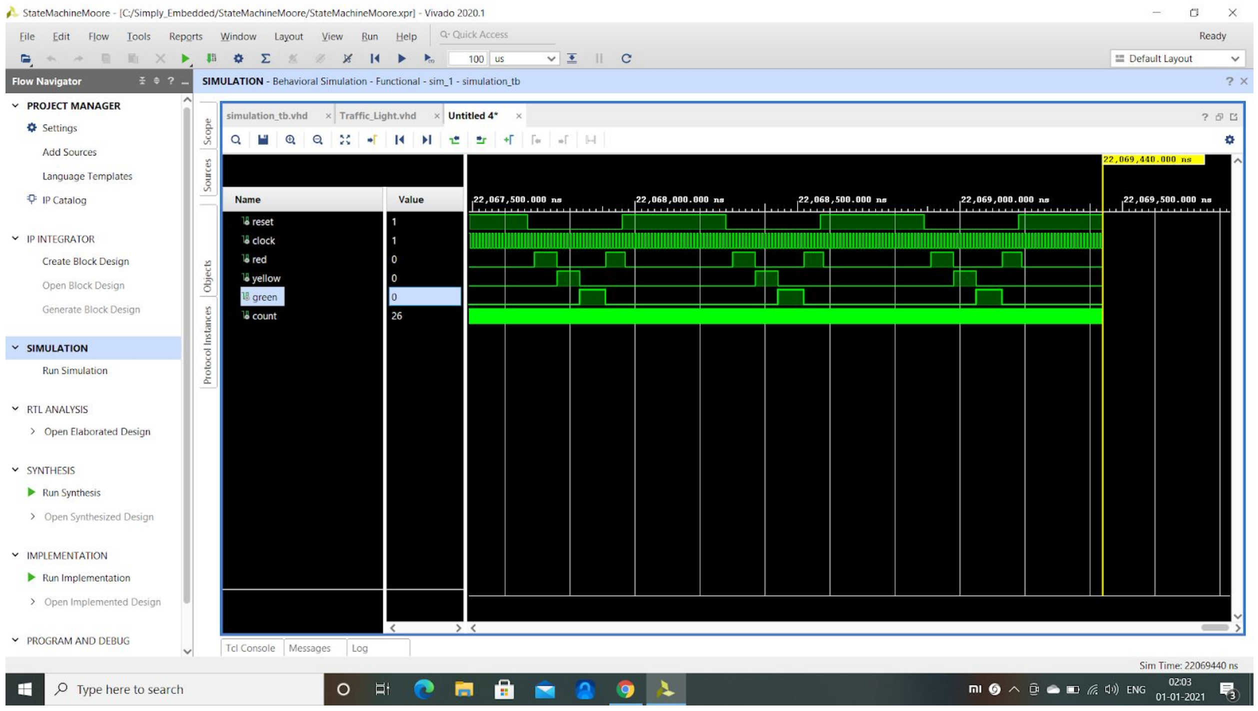
Test bench code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity simulation_tb is
--  Port ( );
end simulation_tb;

architecture Behavioral of simulation_tb is
component Traffic_Light is
Port (
rst : in std_logic;
clck : in std_logic;
r,g,y : out std_logic
 );
end component;
SIGNAL reset: std_logic := '0';
SIGNAL clock : std_logic := '0';
SIGNAL red,green,yellow: std_logic;
signal count : integer range 0 to 62 :=0;
begin
e1: Traffic_Light port map(rst =>reset,clck=>clock,r=>red,g=>green,y=>yellow);
p1: process
begin
clock <= '1';
wait for 5 ns;
clock <= '0';
wait for 5 ns;
end process p1;
p2: process(clock)
begin
if(rising_edge(clock)) then
if(count <= 30) then
 reset <= '1';
 count <= count +1;
elsif ((count > 30) and (count < 60)) then
 reset <= '0';
 count <= count + 1;
else
  reset <= '1';
  count <= 0;
 end if;
 end if;
end process p2;
end Behavioral;
```

WINDOW:



Name	Value
rst	0
clck	1
r	0
y	0
g	1
state	green
count	5



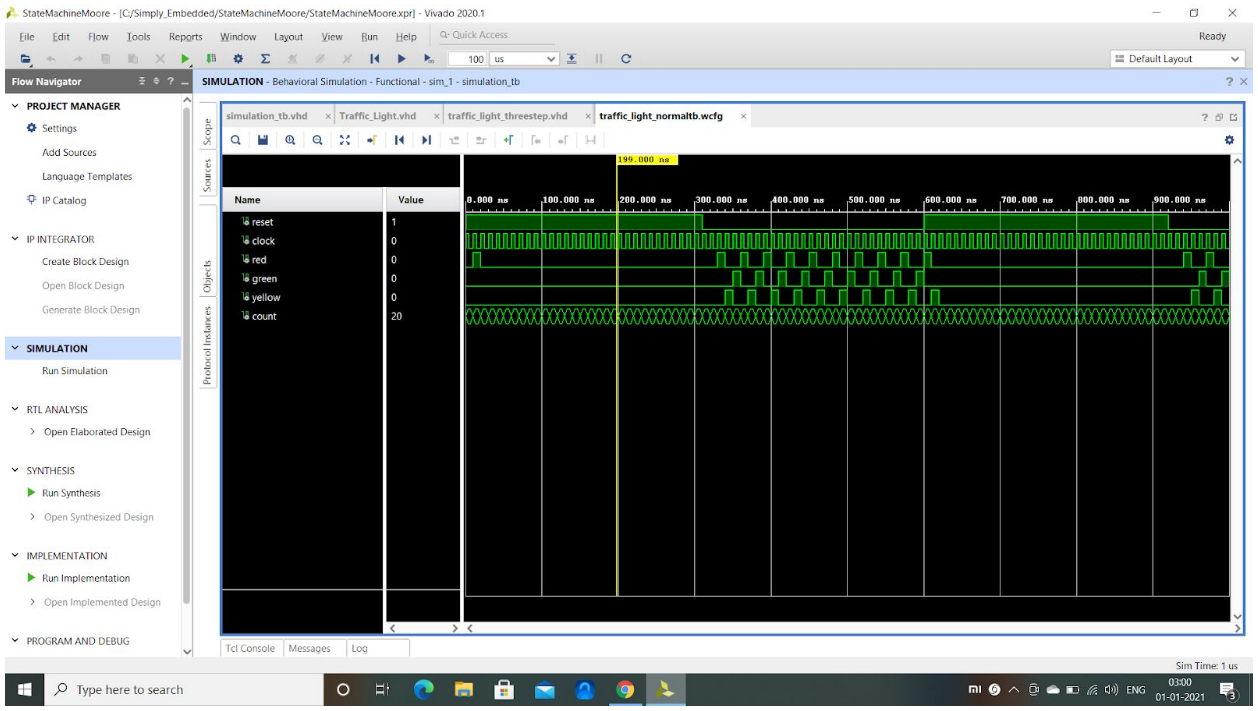
2) 3 step process:
Code:

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


entity traffic_light_threestep is
Port (
rst : in std_logic;
clck : in std_logic;
r,g,y : out std_logic
 );
 end traffic_light_threestep;

architecture Behavioral of traffic_light_threestep is
type state_type is (init,red,green,yellow);
signal state,next_state: state_type;
begin
reset_process: process(clck)
begin
if(rising_edge(clck)) then
if(rst = '1') then
  state <= init;
else
```

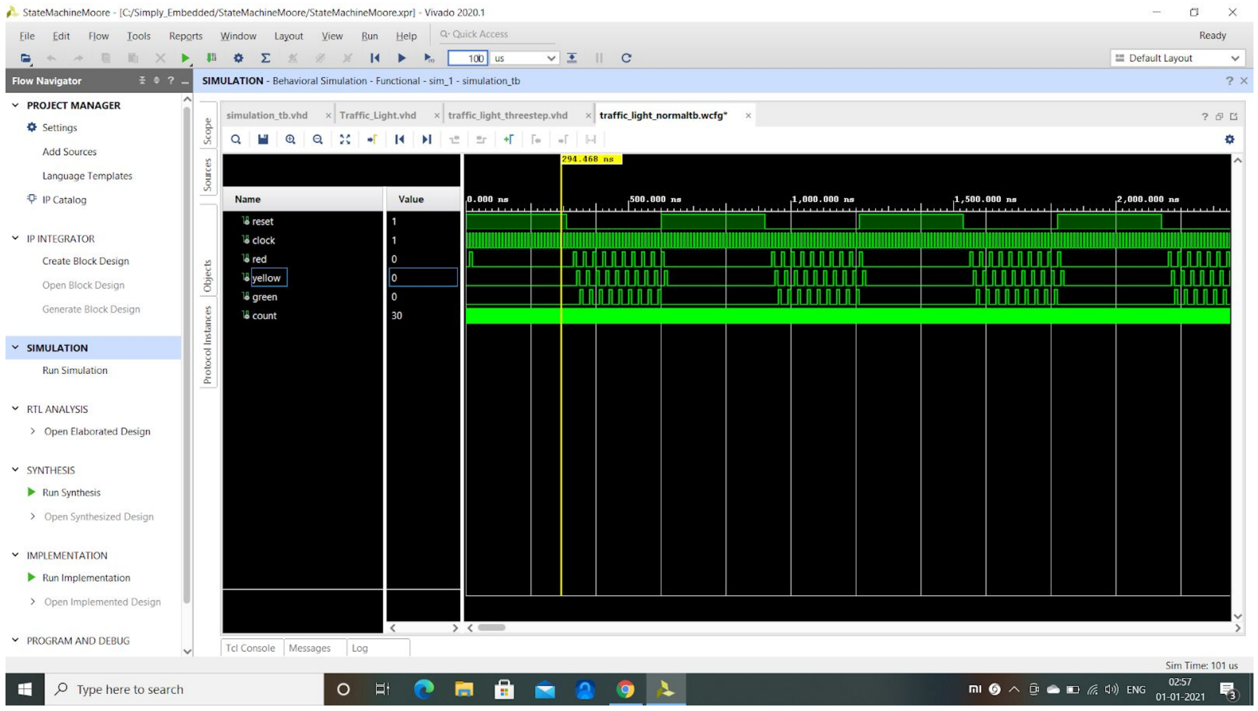
```
  state <= next_state;
end if;
end if;
end process reset_process;
process_output: process(clck)
begin
if(rising_edge(clck)) then
case(state) is
when init=>
 r <= '0';
 g <='0';
 y <= '0';
when red=>
 r <= '1';
 g <='0';
 y <= '0';
when yellow=>
r <= '0';
g <='0';
y <= '1';
when green =>
r <= '0';
g <='1';
y <= '0';
end case;
end if;
end process process_output;
process_nextstate: process(state)
begin
case(state) is
when init =>
    next_state <= red;
when red =>
   next_state <= yellow;
when yellow =>
   next_state <= green;
when green =>
   next_state <= red;
end case;

end process process_nextstate;
end Behavioral;
```

Test bench code : same as before;

Waveform:


StateMachineMoore - [C:/Simply_Embedded/StateMachineMoore/StateMachineMoore.xpr] - Vivado 2020.1
SIMULATION - Behavioral Simulation - Functional - sim_1 - simulation_tb
simulation_tb.vhd
Traffic_Light.vhd
traffic_light_threestep.vhd
traffic_light_normaltb.wcfg
199.000 ns
Sim Time: 1 us

Name	Value
reset	1
clock	0
red	0
green	0
yellow	0
count	20


StateMachineMoore - [C:/Simply_Embedded/StateMachineMoore/StateMachineMoore.xpr] - Vivado 2020.1
SIMULATION - Behavioral Simulation - Functional - sim_1 - simulation_tb
simulation_tb.vhd
Traffic_Light.vhd
traffic_light_threestep.vhd
traffic_light_normaltb.wcfg*
294.468 ns
Sim Time: 101 us

Name	Value
reset	1
clock	1
red	0
yellow	0
green	0
count	30

3) **2 step process**

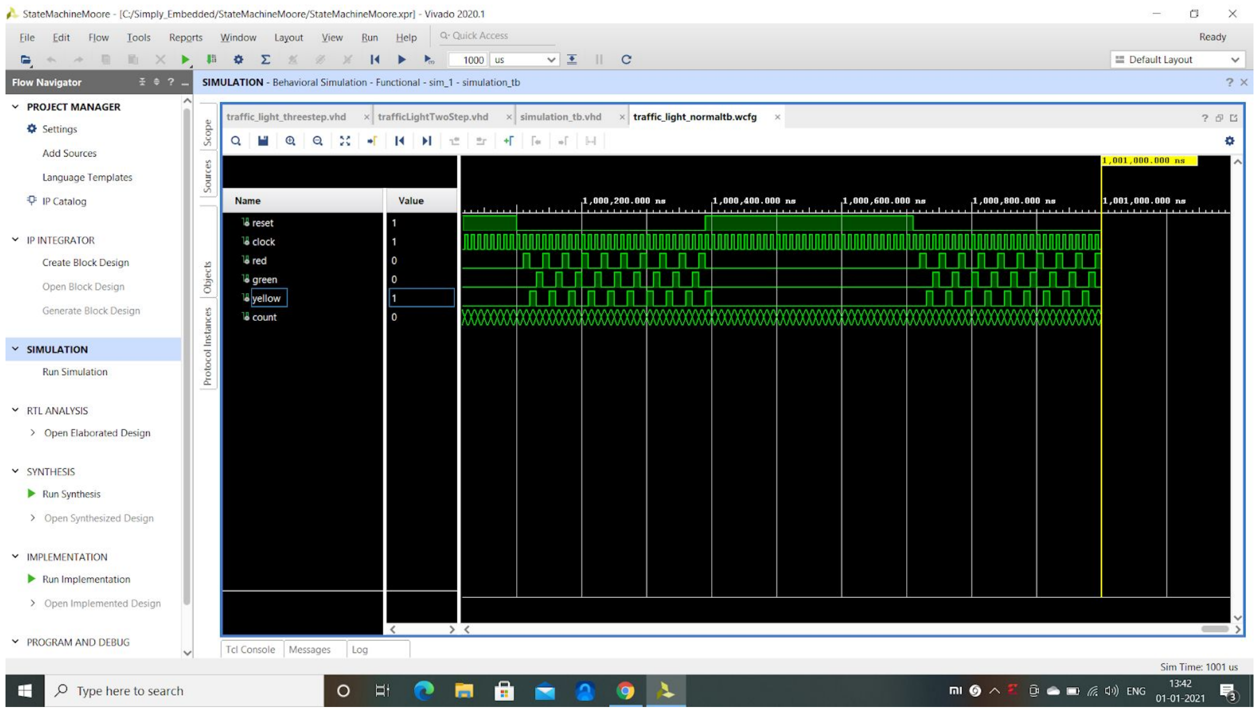
Code:

```
-------------------------------------------------------------------------------
-- Company:
-- Engineer:
```

```
--
-- Create Date: 01.01.2021 02:14:26
-- Design Name:
-- Module Name: traffic_light_threestep - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


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;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity traffic_light_twostep is
Port (
rst : in std_logic;
clck : in std_logic;
r,g,y : out std_logic
 );
 end traffic_light_twostep;

architecture Behavioral of traffic_light_twostep is
type state_type is (init,red,green,yellow);
signal state,next_state: state_type;
begin
reset_process: process(clck)
begin
```

```
if(rising_edge(clck)) then
if(rst = '1') then
  state <= init;
else
  state <= next_state;
end if;
end if;
end process reset_process;
--process_output: process(clck)
--begin
--if(rising_edge(clck)) then
--case(state) is
--when init=>
-- r <= '0';
-- g <='0';
-- y <= '0';
--when red=>
-- r <= '1';
-- g <='0';
-- y <= '0';
--when yellow=>
--r <= '0';
--g <='0';
--y <= '1';
--when green =>
--r <= '0';
--g <='1';
--y <= '0';
--end case;
--end if;
--end process process_output;
process_nextstate: process(state)
begin
--if(rising_edge(clck)) then

case(state) is
when init =>
    r <= '0';
    g <='0';
    y <= '0';
    next_state <= red;
when red =>
   r <= '1';
   g <='0';
```

```
    y <= '0';
    next_state <= yellow;
when yellow =>
    r <= '0';
    g <='0';
    y <= '1';
    next_state <= green;
when green =>
    r <= '0';
    g <='1';
    y <= '0';
    next_state <= red;
end case;
--end if;
end process process_nextstate;
end Behavioral;
```



Name	Value
reset	1
clock	1
red	0
green	0
yellow	1
count	0

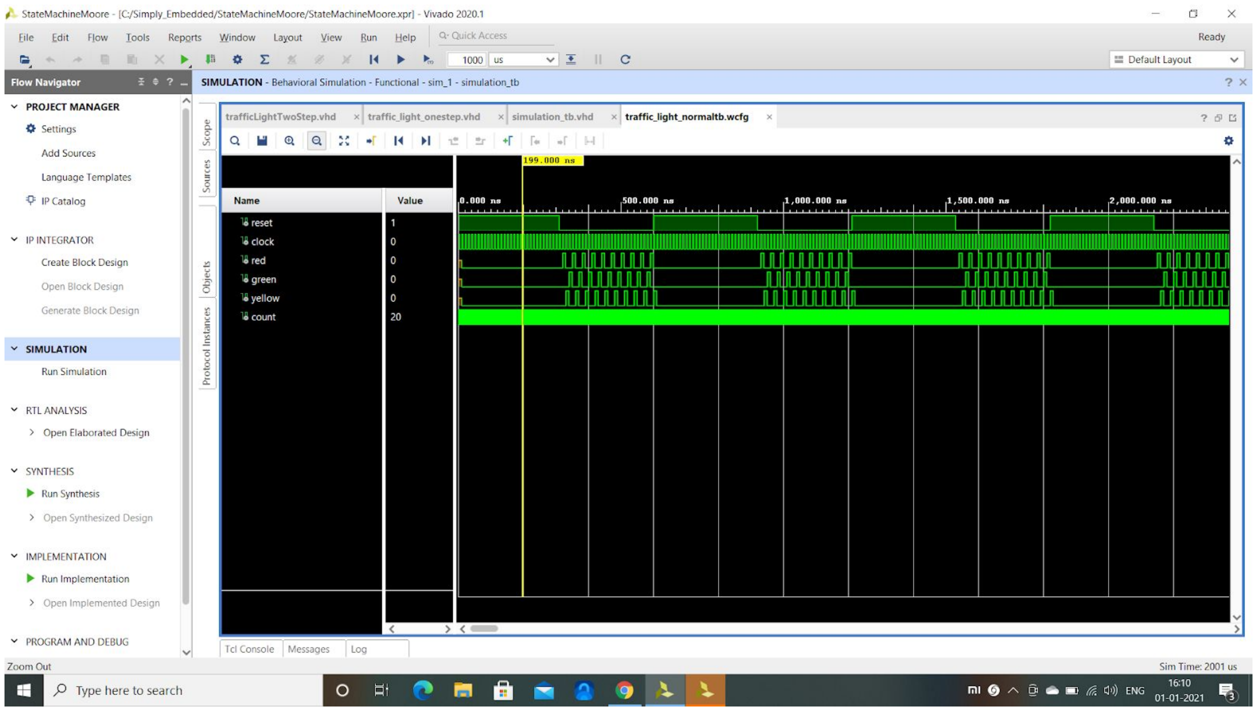
4) 1 step process:
Code:

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 01.01.2021 02:14:26
-- Design Name:
-- Module Name: traffic_light_threestep - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


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;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity traffic_light_onestep is
Port (
rst : in std_logic;
clck : in std_logic;
r,g,y : out std_logic
 );
 end traffic_light_onestep;

architecture Behavioral of traffic_light_onestep is
type state_type is (init,red,green,yellow);
signal state,next_state: state_type;
begin
--reset_process: process(clck)
--begin
--if(rising_edge(clck)) then
--if(rst = '1') then
```

```
--  state <= init;
--else
--  state <= next_state;
--end if;
--end if;
--end process reset_process;
--process_output: process(clck)
--begin
--if(rising_edge(clck)) then
--case(state) is
--when init=>
-- r <= '0';
-- g <='0';
-- y <= '0';
--when red=>
-- r <= '1';
-- g <='0';
-- y <= '0';
--when yellow=>
--r <= '0';
--g <='0';
--y <= '1';
--when green =>
--r <= '0';
--g <='1';
--y <= '0';
--end case;
--end if;
--end process process_output;
process_nextstate: process(clck)
begin
if(rising_edge(clck)) then
if(rst = '1') then
    r <= '0';
    g <='0';
    y <= '0';
    state <= red;
 else
case(state) is
when init =>

when red =>
   r <= '1';
   g <='0';
```

```
  y <= '0';
  state <= yellow;
when yellow =>
  r <= '0';
  g <='0';
  y <= '1';
  state <= green;
when green =>
  r <= '0';
  g <='1';
  y <= '0';
  state <= red;
end case;
end if;
end if;
end process process_nextstate;
end Behavioral;
```

Waveform:



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

SIMULATION - Behavioral Simulation - Functional - sim_1 - simulation_tb

Name	Value
reset	1
clock	0
red	0
green	0
yellow	0
count	20

Sim Time: 2001 us