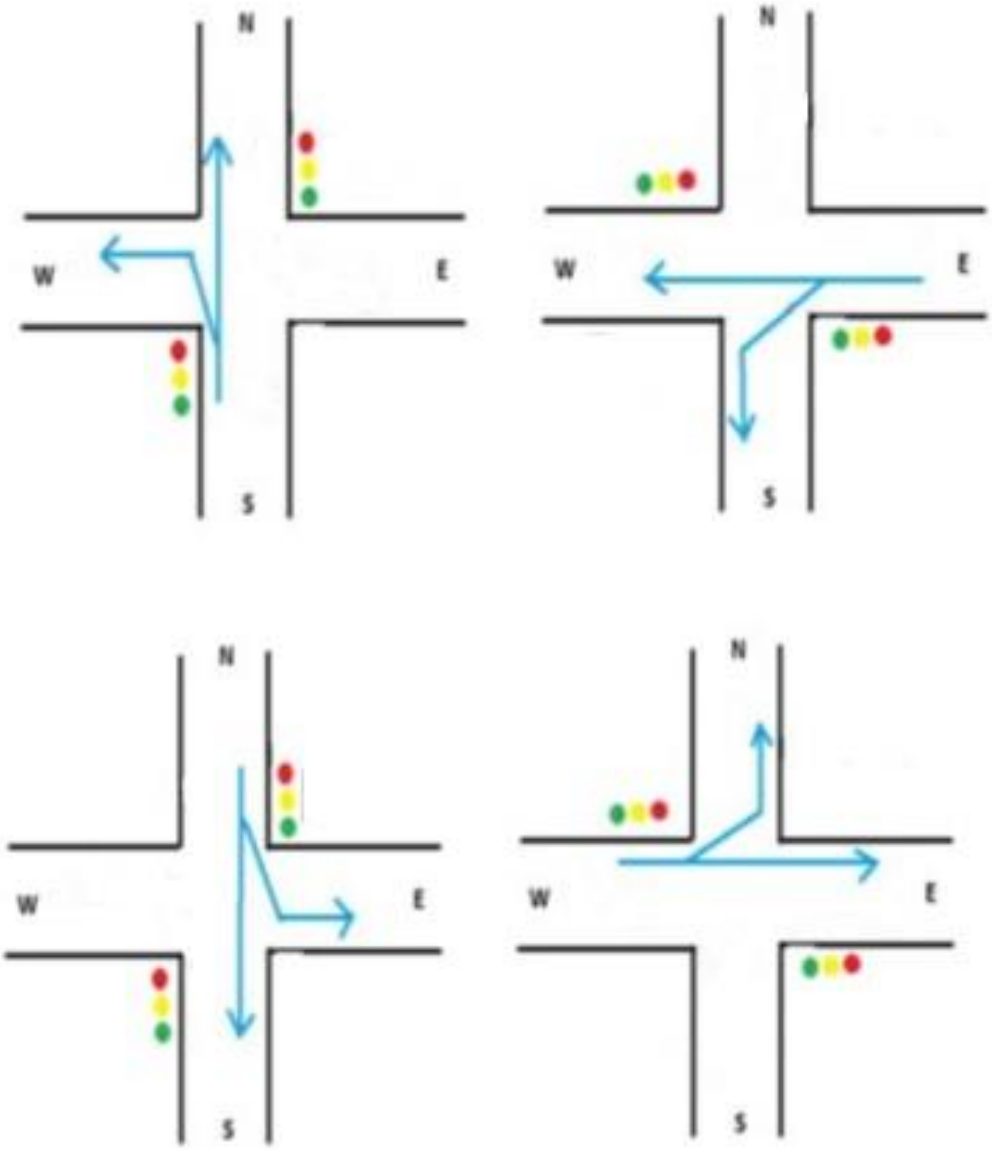
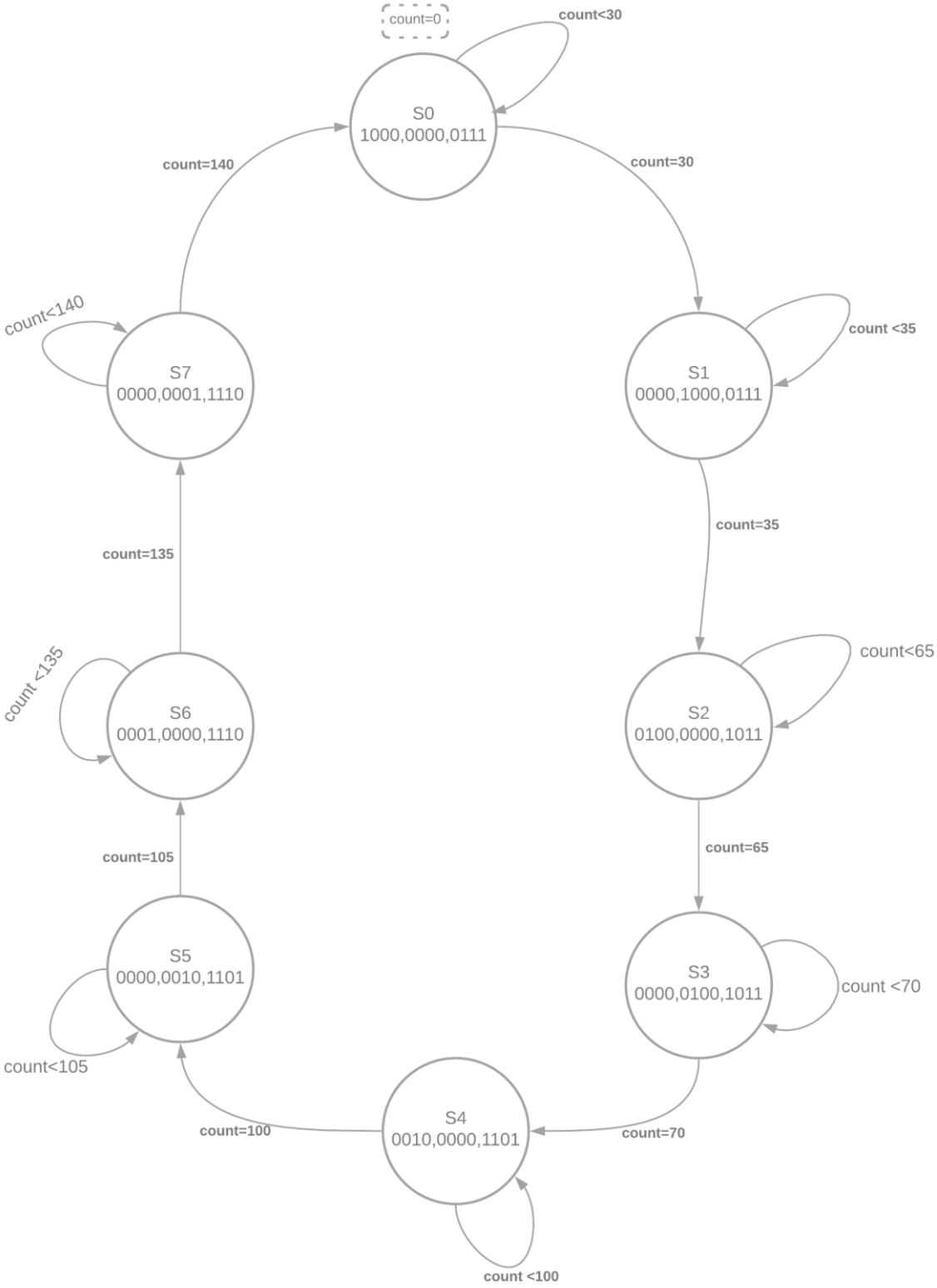
Specification:



Our design is for a 4-way road. Each Direction (North , South ,east and west) got a traffic 3 colours each, Green ,yellow and red. No traffic for pedestrians implemented.

4-Way Traffic Light Controller State Diagram:



VHDL CODE:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;




entity traffic_light is

port ( clk : in std_logic;
      rst : in std_logic;
      green : out std_logic_vector(3 downto 0); --NESW
      red : out std_logic_vector(3 downto 0);-- NESW
      yellow: out std_logic_vector(3 downto 0));  --NESW

end traffic_light;




architecture Behavioral of traffic_light is

type state is (S0,S1,S2,S3,S4,S5,S6,S7);

signal currentstate: state := S0;

begin
counting: Process ( clk,rst)
```

```
variable count: integer := 0; -- Intialize count by 0
  begin
     if rst = '1' then
       currentstate <= S0; -- remain at S0
       count := 0;

     else
       if clk'event AND clk = '1' then
         count := count +1;
         if count<30 then -- from 0 to 30 seconds green light for North
           currentstate <= S0;

       elsif  count>=30 AND count <35 then  --  from 30s to 35s  yellow light for North
            currentstate <= S1;
       elsif count >=35  AND count <65 then -- from 35s to 65s green light for East
            currentstate <= S2;
       elsif count >= 65 AND count <70 then -- from 65s to 70s yellow light for East
            currentstate <= S3;
       elsif count >=70 AND count <100 then -- from 70s to 100 green light for South
            currentstate <= S4;
       elsif count >= 100 AND count <105 then --from 100s to 105s yellow light for   South
            currentstate <= S5;
       elsif count >= 105 AND count <135 then --from 105s to 135s green light for West
            currentstate <= S6;
       elsif count >= 135 AND count <140 then --from 135s to 140s  yellow light for West
            currentstate <= S7;
       elsif count >= 140 then -- start again from state S0
            count := 0;
        END IF;
      END IF;
    END IF;
```

```
END PROCESS;


st : Process (currentstate)
begin
  case currentstate is
    when S0 => green <= "1000" ; yellow <= "0000" ; red <= "0111";
    when S1 => green <= "0000" ; yellow <= "0100" ; red <= "0111";
    when S2 => green <= "0100" ; yellow <= "0000" ; red <= "1011";
    when S3 => green <= "0000" ; yellow <= "0100" ; red <= "1011";
    when S4 => green <= "0010" ; yellow <= "0000" ; red <= "1101";
    when S5 => green <= "0000" ; yellow <= "0010" ; red <= "1101";
    when S6 => green <= "0001" ; yellow <= "0000" ; red <= "1110";
    when S7 => green <= "0000" ; yellow <= "0001" ; red <= "1110";
    when others => green <= "1000" ; yellow <= "0000" ; red <= "0111";
  end case;
end process;


end Behavioral;
```

CODE EXPLANATION:

A 4 way traffic light is controlled by 5 signals clk,rst,green,yellow and red. Each direction North,East,South and West Traffic lights is controlled by 1 bit from the bit vector of green ,yellow and red signal.

Count is incremented whenever there's a clock. Counts keep on incrementing and its values is checked in the if conditions to Change from one state to another.

S0 = green light for North direction and red for others

S1= yellow light for North direction and red for others

S2= green light for East direction and red for others

S3= yellow light for East direction and red for others

S4= green light for South direction and red for others

S5= yellow light for South direction and red for others

S6= green light for West direction and red for others

S7= yellow light for West direction and red for others

In the second process

We take the current state as the sensitivity list and whenever the current state gets changed the current state signal triggers the green,yellow and red signals to change values. If state is anything else than the stated states (rst=1) then it stays at the first state which green light for north and red for others as S0

State Table:

Current State	Next State	Green	Yellow	Red
S0	S2	1000	0000	0111
S1	S3	0000	1000	0111
S2	S4	0100	0000	1011
S3	S5	0000	0100	1011
S4	S6	0010	0000	1101
S5	S6	0000	0010	1101
S6	S7	0001	0000	1110
S7	S0	0000	0001	1110

Test Bench:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
entity Testbench is
end entity;
architecture tb of Testbench is
signal green: std_logic_vector (3 downto 0);
signal yellow : std_logic_vector (3 downto 0);
signal red : std_logic_vector (3 downto 0);
signal clk,rst: std_logic;
begin
DUT : ENTITY work.traffic_light
PORT MAP(
green => green,
yellow => yellow,
red => red,
clk => clk,
rst => rst );


Clock : process
begin
clk <= '0';
wait for 1 ns;
clk <= '1';
wait for 1 ns;
end process;
```

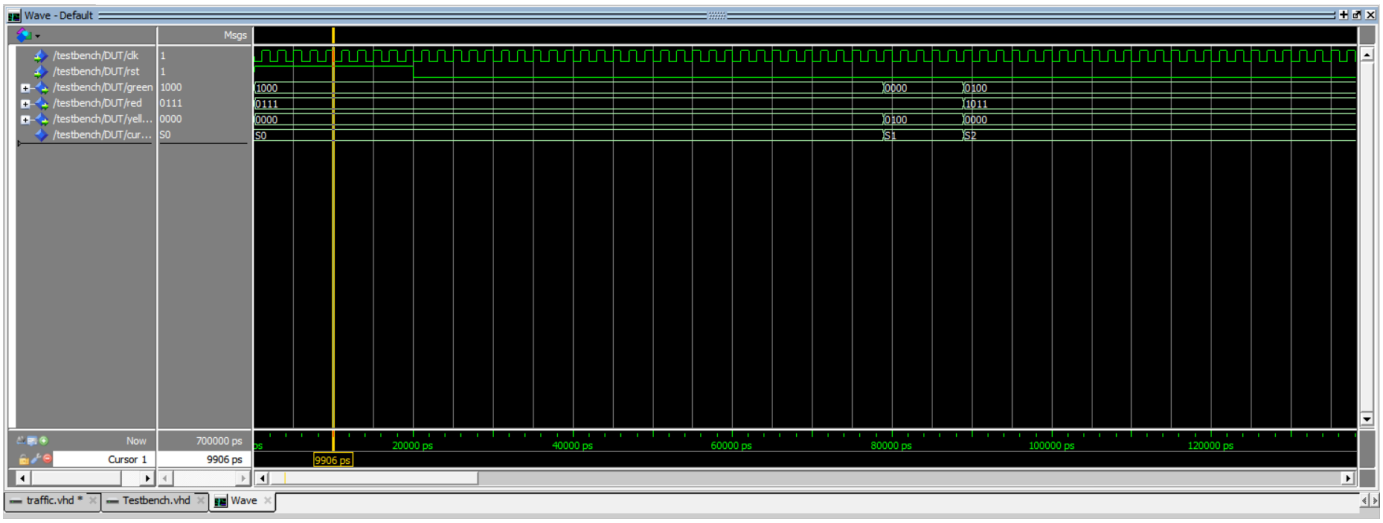
```
stimulis : process
begin
report("Starting simulation");
rst <= '1'; wait for 20 ns;
rst <= '0'; wait for 200000 ns;
report("End simulation");
end process;
end architecture;
```

Test Bench Explanation:

Clk is every 1 nanosecond and therefore count is incremented every 1 nanosecond

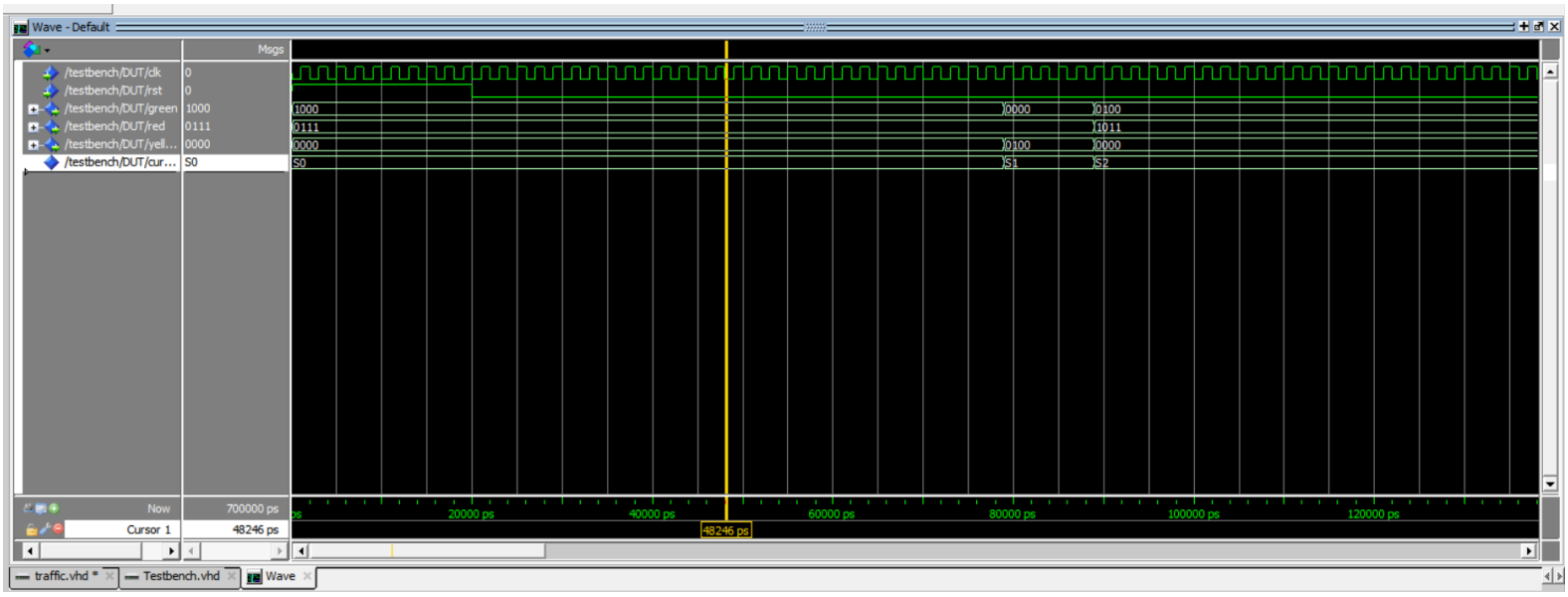
Also, the first 20 ns the rst signal is 1 and so we could see the output is S0 according to the code

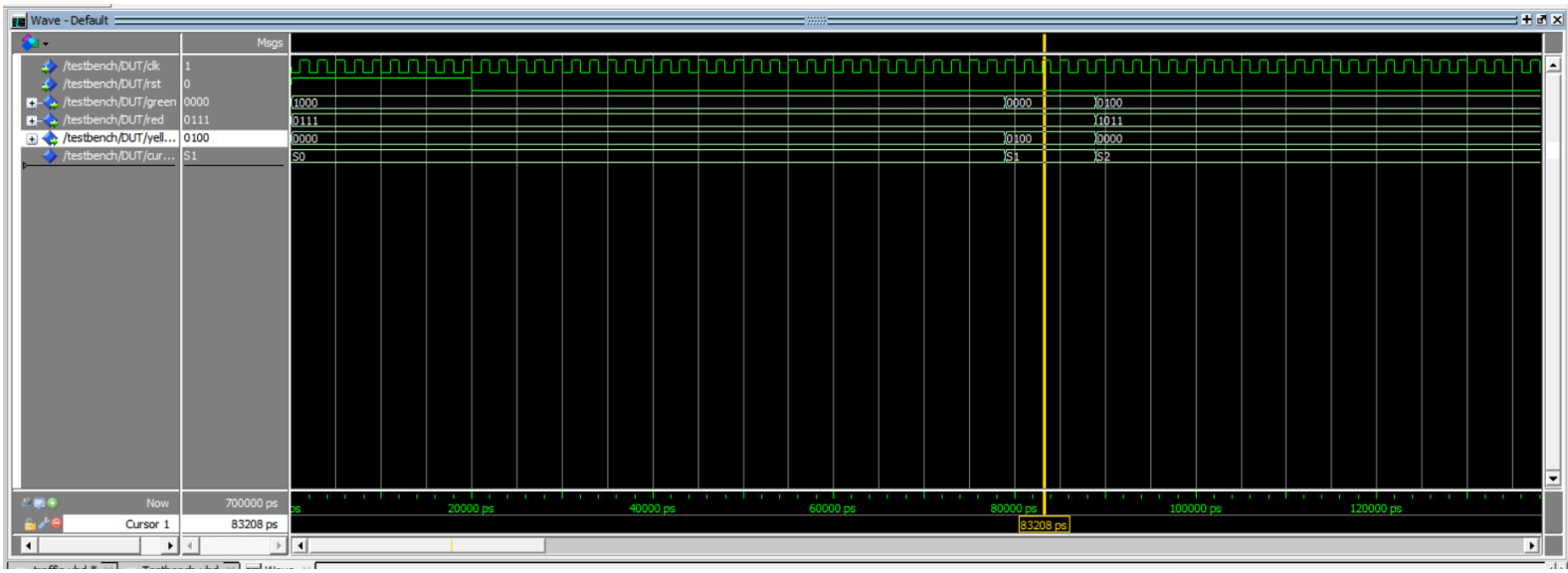
Output Waveform:

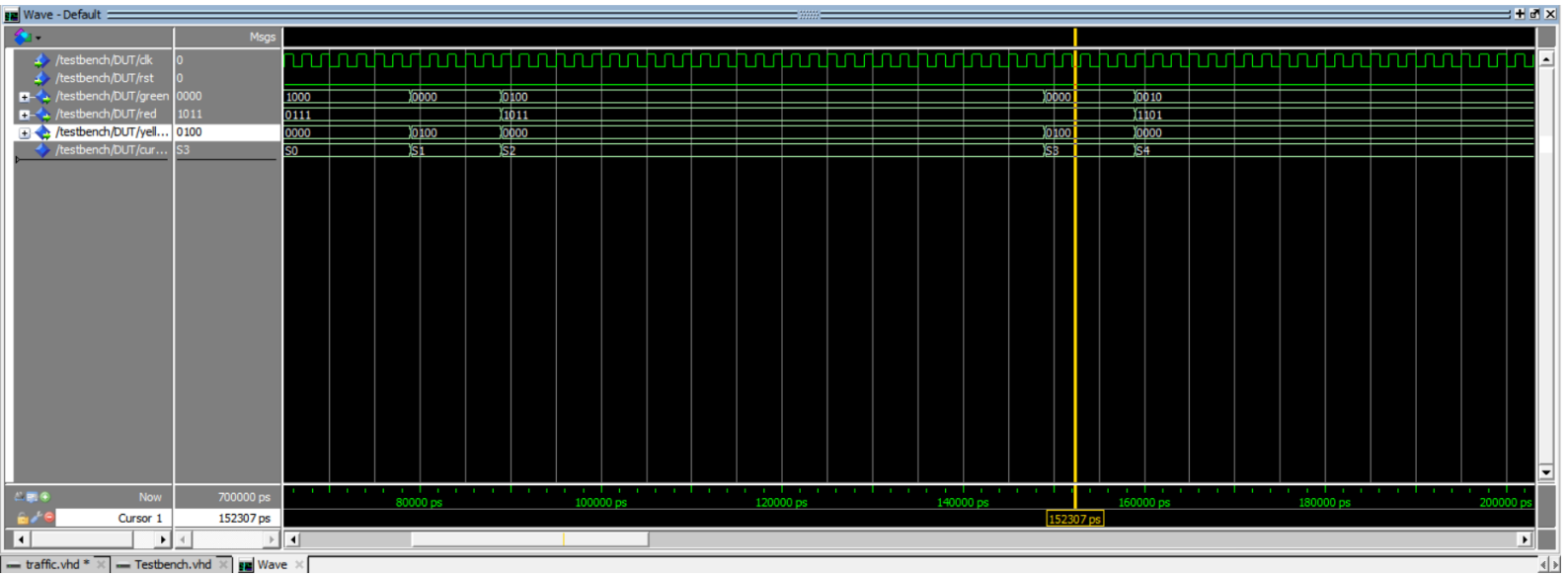


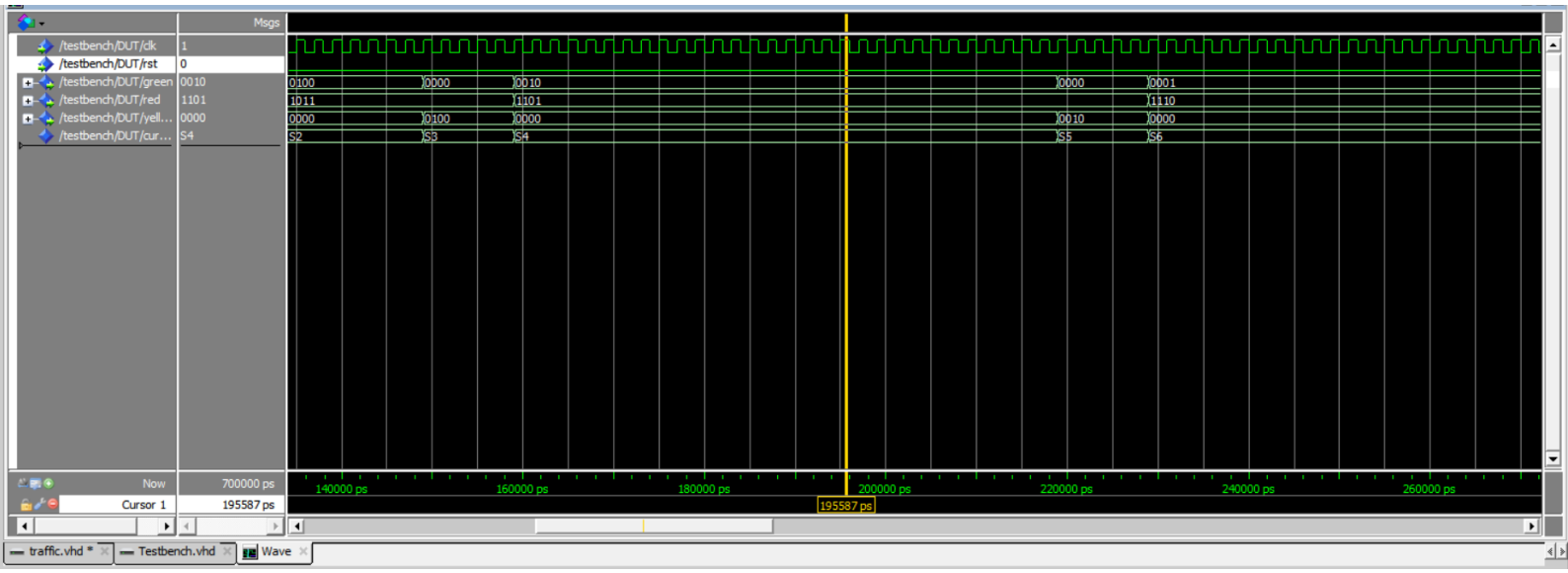
The first 20ns rst=1 and so state=S0

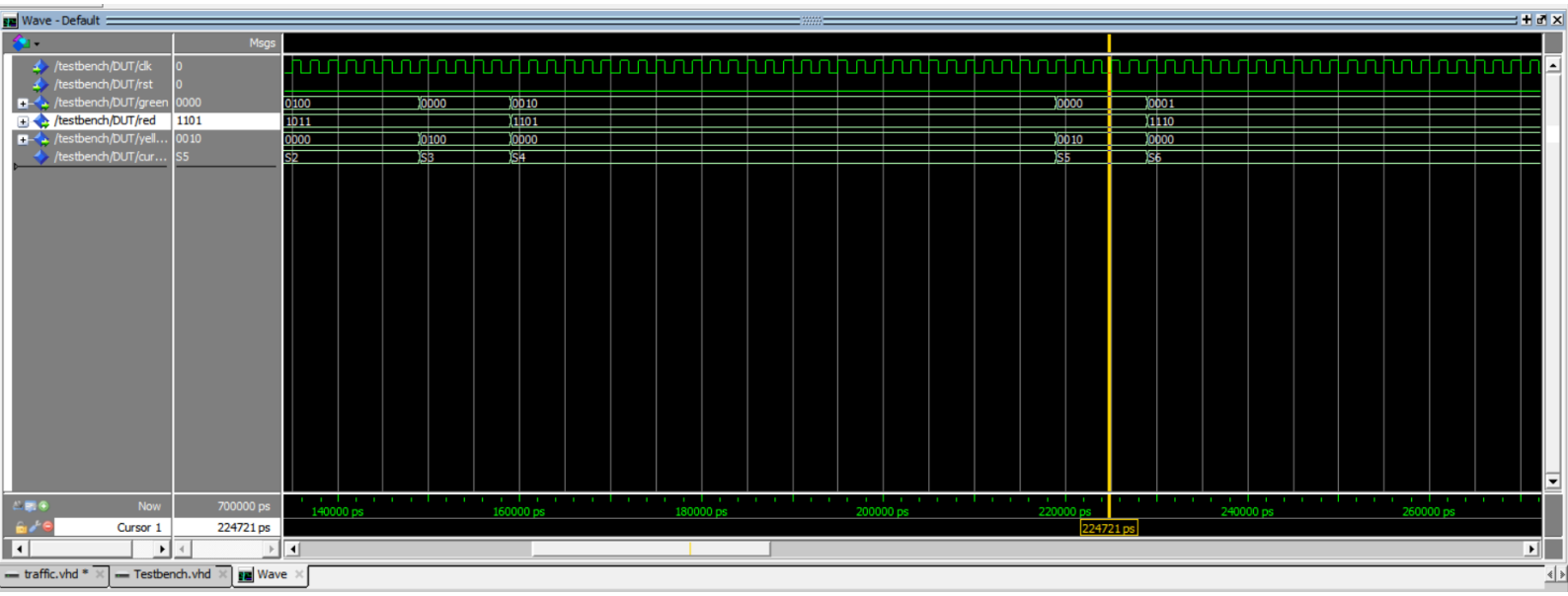
Then rst= 0 and states start changing

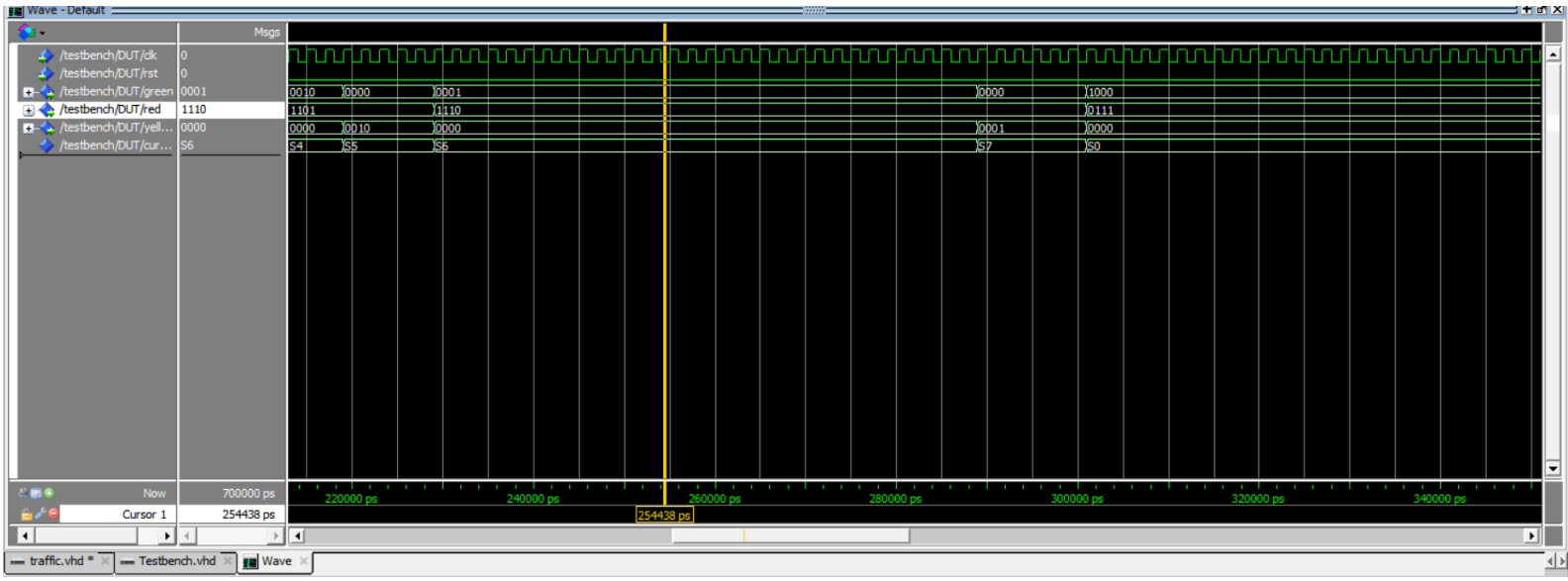


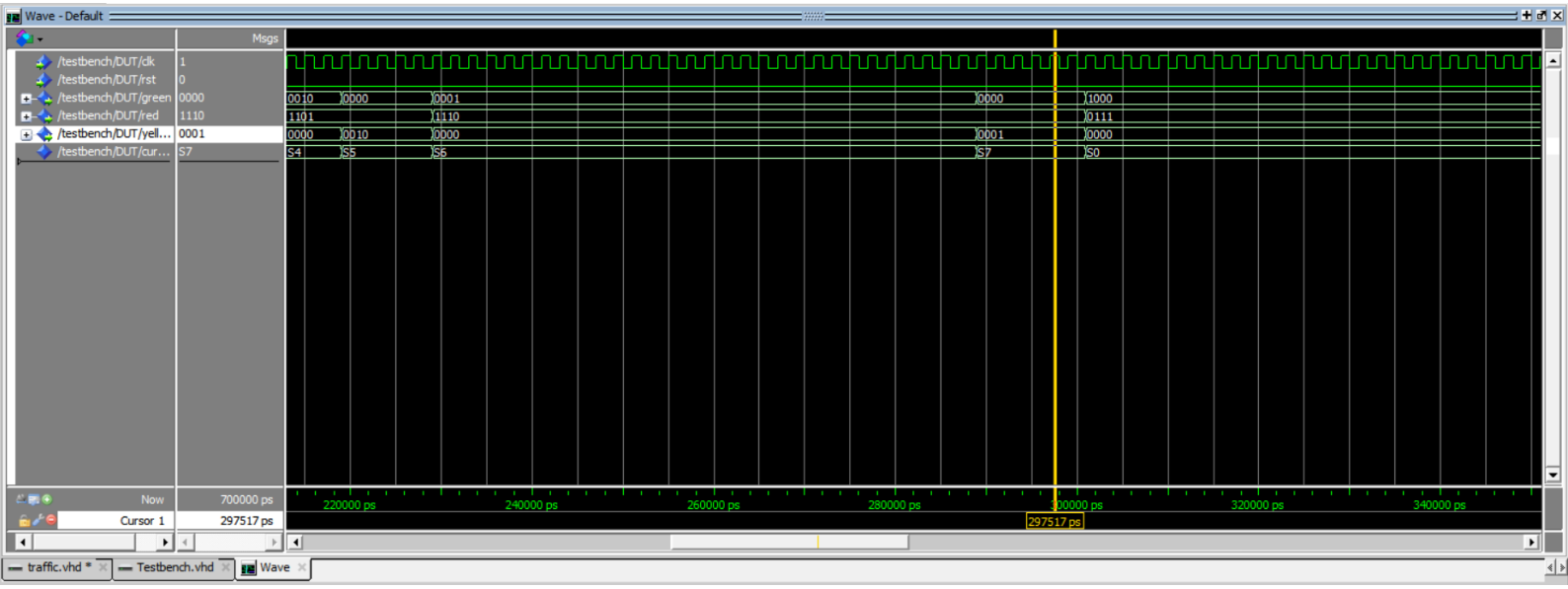

Wave - Default
Msgs
/testbench/DUT/clk
/testbench/DUT/rst
/testbench/DUT/green
/testbench/DUT/red
/testbench/DUT/yell...
/testbench/DUT/cur...
1
0
0000
0111
0100
S1
1000
0000
0100
0111
1011
0000
0100
0000
S0
S1
S2
Now
700000 ps
Cursor 1
83208 ps
20000 ps
40000 ps
60000 ps
80000 ps
100000 ps
120000 ps
83208 ps

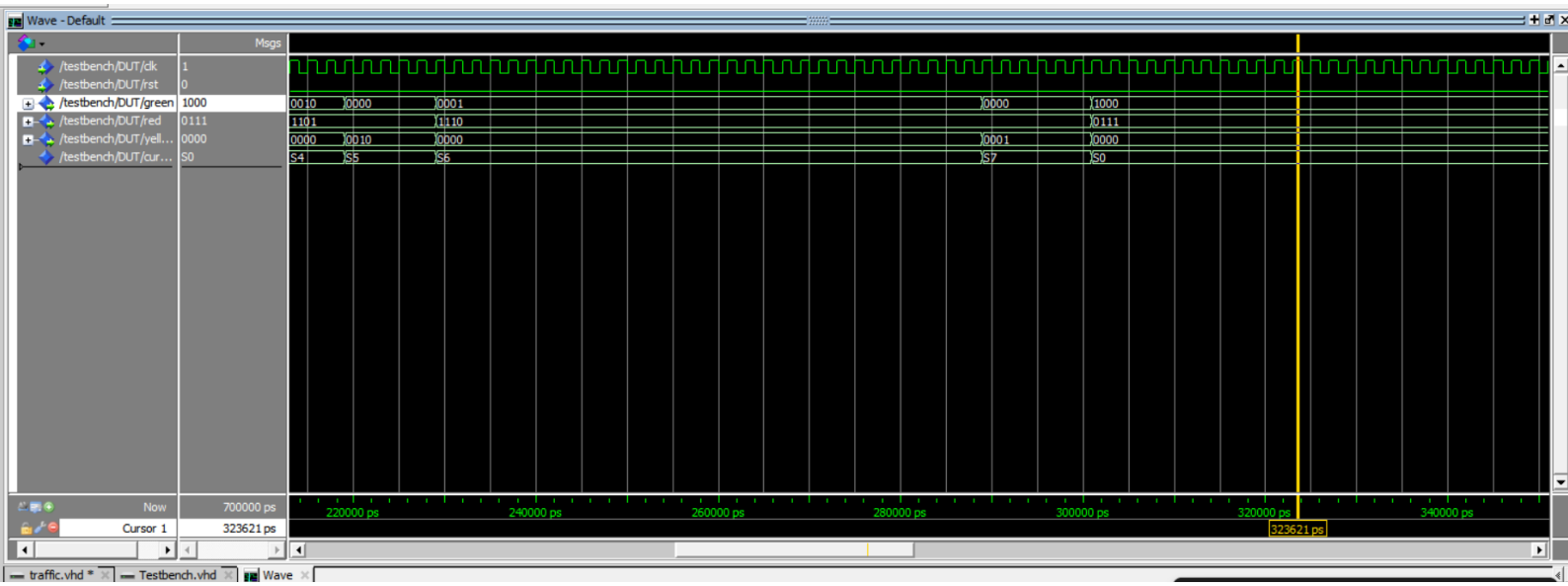

Wave - Default
Msgs
/testbench/DUT/clk
/testbench/DUT/rst
/testbench/DUT/green
/testbench/DUT/red
/testbench/DUT/yell...
/testbench/DUT/cur...
0
0
0000
1011
0100
S3
1000
0000
0100
0000
0010
0111
1011
1101
0000
0100
0000
0100
0000
S0
S1
S2
S3
S4
Now
700000 ps
Cursor 1
152307 ps
80000 ps
100000 ps
120000 ps
140000 ps
160000 ps
180000 ps
200000 ps
152307 ps
traffic.vhd *
Testbench.vhd
Wave


Msgs
/testbench/DUT/clk
/testbench/DUT/rst
/testbench/DUT/green
/testbench/DUT/red
/testbench/DUT/yell...
/testbench/DUT/cur...
1
0
0010
1101
0000
S4
0100
0000
0010
0000
0001
1011
1101
1110
0000
0100
0000
0010
0000
S2
S3
S4
S5
S6
Now
700000 ps
Cursor 1
195587 ps
140000 ps
160000 ps
180000 ps
200000 ps
220000 ps
240000 ps
260000 ps
195587 ps
traffic.vhd *
Testbench.vhd
Wave


Wave - Default
Msgs
/testbench/DUT/clk
/testbench/DUT/rst
/testbench/DUT/green
/testbench/DUT/red
/testbench/DUT/yell...
/testbench/DUT/cur...
0
0
0000
1101
0010
S5
0100
0000
0010
0000
0001
1011
1101
1110
0000
0100
0000
0010
0000
S2
S3
S4
S5
S6
Now
700000 ps
Cursor 1
224721 ps
140000 ps
160000 ps
180000 ps
200000 ps
220000 ps
240000 ps
260000 ps
224721 ps
traffic.vhd *
Testbench.vhd
Wave


Wave - Default
Msgs
/testbench/DUT/clk
/testbench/DUT/rst
/testbench/DUT/green
/testbench/DUT/red
/testbench/DUT/yell...
/testbench/DUT/cur...
0
0
0001
1110
0000
S6
0010
0000
0001
0000
1000
1101
1110
0111
0000
0010
0000
0001
0000
S4
S5
S6
S7
S0
Now
700000 ps
Cursor 1
254438 ps
220000 ps
240000 ps
260000 ps
280000 ps
300000 ps
320000 ps
340000 ps
254438 ps
traffic.vhd *
Testbench.vhd
Wave


Wave - Default
Msgs
/testbench/DUT/clk
/testbench/DUT/rst
/testbench/DUT/green
/testbench/DUT/red
/testbench/DUT/yell...
/testbench/DUT/cur...
1
0
0000
1110
0001
S7
0010
0000
0001
0000
1000
1101
1110
0111
0000
0010
0000
0001
0000
S4
S5
S6
S7
S0
Now
700000 ps
Cursor 1
297517 ps
220000 ps
240000 ps
260000 ps
280000 ps
300000 ps
320000 ps
340000 ps
297517 ps
traffic.vhd *
Testbench.vhd
Wave


Wave - Default
Msgs
/testbench/DUT/clk
/testbench/DUT/rst
/testbench/DUT/green
/testbench/DUT/red
/testbench/DUT/yell...
/testbench/DUT/cur...
1
0
1000
0111
0000
S0
0010
0000
0001
0000
1000
1101
1110
0111
0000
0010
0000
0001
0000
S4
S5
S6
S7
S0
Now
700000 ps
Cursor 1
323621 ps
220000 ps
240000 ps
260000 ps
280000 ps
300000 ps
320000 ps
340000 ps
323621 ps
traffic.vhd *
Testbench.vhd
Wave