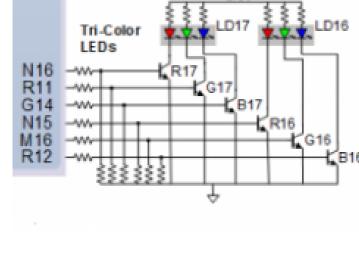
Assignment 8

1. Preparation tasks



•																
Input P	0	0	1	1	0	1	0	1	1	1	1	0	0	1	1	1
Clock	↑	†	↑	1												
State	Α	А	В	С	С	D	А	В	С	D	В	В	В	С	D	В
Output R	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0

Figure with connection of RGB LEDs on Nexys A7 board and completed table with color settings



State diagram

2. Traffic light controller

Listing of VHDL code of sequential process p_traffic_fsm with syntax highlighting

-- Clear all bits

-- Every 250 ms, CASE checks the value of the s_state

-- If the current state is STOP1, then wait 1 sec

-- variable and changes to the next state according

-- and move to the next GO_WAIT state.

-- Count up to c_DELAY_1SEC

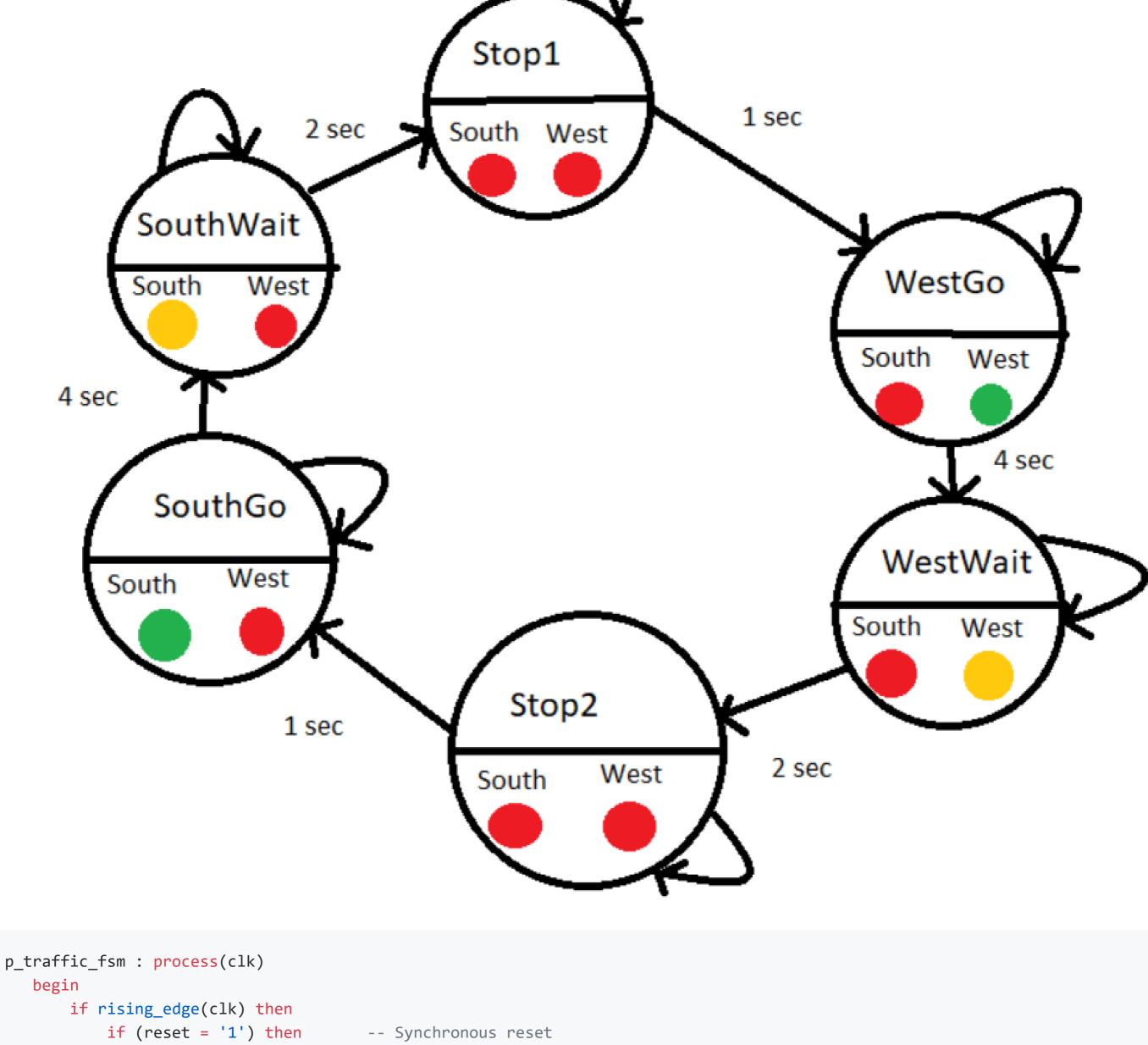
s_cnt <= c_ZERO;

-- to the delay value.

when STOP1 =>

elsif (s_en = '1') then

case s_state is



```
if (s_cnt < c_DELAY_1SEC) then</pre>
                              s_cnt <= s_cnt + 1;
                          else
                              -- Move to the next state
                              s_state <= WEST_GO;</pre>
                              -- Reset local counter value
                              s_cnt <= c_ZERO;
                          end if;
                      when WEST_GO =>
                          if (s_cnt < c_DELAY_4SEC) then</pre>
                              s_cnt <= s_cnt + 1;
                          else
                              s_state <= WEST_WAIT;</pre>
                              s_cnt <= c_ZERO;
                          end if;
                      when WEST_WAIT =>
                          if (s_cnt < c_DELAY_2SEC) then</pre>
                              s_cnt <= s_cnt + 1;
                          else
                              s_state <= STOP2;
                              s_cnt <= c_ZERO;
                          end if;
                      when STOP2 =>
                          if (s_cnt < c_DELAY_1SEC) then</pre>
                              s_cnt <= s_cnt + 1;
                          else
                              s_state <= SOUTH_GO;</pre>
                              s_cnt <= c_ZERO;
                          end if;
                      when SOUTH_GO =>
                          if (s_cnt < c_DELAY_4SEC) then</pre>
                              s_cnt <= s_cnt + 1;
                          else
                              s_state <= SOUTH_WAIT;</pre>
                              s_cnt <= c_ZERO;
                          end if;
                      when SOUTH_WAIT =>
                          if (s_cnt < c_DELAY_2SEC) then</pre>
                              s_cnt <= s_cnt + 1;
                          else
                              s_state <= STOP1;
                              s_cnt <= c_ZERO;
                          end if;
                      -- It is a good programming practice to use the
                      -- OTHERS clause, even if all CASE choices have
                      -- been made.
                      when others =>
                          s_state <= STOP1;</pre>
                  end case;
              end if; -- Synchronous reset
          end if; -- Rising edge
      end process p_traffic_fsm;
Listing of VHDL code of combinatorial process p_output_fsm with syntax highlighting
  p_output_fsm : process(s_state)
      begin
          case s_state is
              when STOP1 =>
                  south_o <= "100"; -- Red (RGB = 100)
                  west_o <= "100"; -- Red (RGB = 100)
              when WEST_GO =>
                  south_o <= "100"; -- Red (RGB = 100)
```

when STOP2 => south_o <= "100"; -- Red (RGB = 100)

west_o <= "010"; -- Green (RGB = 010)

south_o <= "100"; -- Red (RGB = 100)

west_o <= "110"; -- Yellow (RGB = 110)

when WEST_WAIT =>

100 / 010 / 110 /

010

110

yellow

red

red

red

S-W

0-0

0-1

1-0

SouthWait

2 sec

2 s

1 s

4 s

2 s

Stop1

West

South

South = 1, West = 0

S-W 0-0

0-1

WestGo

S-W

1 sec 1-0

red

red

green

yellow

100

₩ s_south[2:0] 110

100

X 110 X

6 [2] [1] **[**0] s_west[2:0]

```
west_o <= "100"; -- Red (RGB = 100)
               when SOUTH_GO =>
                    south_o <= "010"; -- Green (RGB = 010)
                    west_o <= "100"; -- Red (RGB = 100)
               when SOUTH_WAIT =>
                    south_o <= "110"; -- Yellow (RGB = 110)
                    west_o <= "100"; -- Red (RGB = 100)
               when others =>
                    south_o <= "100"; -- Red
                    west_o <= "100"; -- Red
           end case;
      end process p_output_fsm;
Screenshot(s) of the simulation, from which it is clear that controller works correctly
 Name
             Value
                                                      |1,000.000 ns |1,200.000 ns |1,400.000 ns |1,600.000 ns |1,800.000 ns |2,000.000 ns |2,200.000 ns |2,400.000 ns |2,600.000 ns |2,800.000
  1 s_clk_100MH; 0
  s_reset
  6 s_south[2:0] 110
    [2]
    14 [1]
    [0]
  s_west[2:0]
            100
    6 [2]
    [1] 🌡
    • [0]
  2,800.000 ns 3,000.000 ns 3,200.000 ns 3,400.000 ns 3,600.000 ns 3,800.000 ns 4,000.000 ns 4,200.000 ns 4,400.000 ns 4,600.000 ns 4,800.000 ns 5,000.000 ns 5,200.000 ns
 6 s_clk_100MH; 0
 s_reset
```

010

100

010

110 X

100

1 [2]	1											
16 [1]	0											
Ta [0]	0											
c_CLKRIOD	10000 ps				<u> </u>	10000 ps						
ame	Value	5,200.000 ns 5,400.000 n	5,600.000 ns 5,800.000 ns	6,000.000 ns	6,200.000 ns		600.000 ns	6,800.000 ns	7,000.000 ns	7,200.000 ns	7,400.000 ns	7,600.000 ns
s_clk_100MH	0											
s_reset	0					<u> </u>	V 212		<u> </u>			
s_south[2:0]	110	100 010	110 100	010	110 X	100	010	110	100	010	X 110 X	100
الله [1]	1											
Ta [0]	0											
s_west[2:0]	100	010 110 100	V 010 V 110 V	100		010 110	100	X_	010 110	100	X	010 1
1 [2]	1											
74 [1] 74 [0]	0		+	+		╀						
c_CLKRIOD						10000 ps						
ame	Value	7,500.000 ns	8,000.000 ns		8,500.000 ns		9,000	.000 ns		9,500.000 ns		10
l s_clk_100MF	łi 0											
s_reset	0											
s_south[2:0]	110	100	X 010 X 110 X	100	X 010 X	110	100	010	X 110 X	100	X 010	110
18 [2] 18 [1]	1											
[0]	0											
s_west[2:0]	100	100 010 110	100 0	10 110	100	010) X 110 X	100		010 110	100	0
[2]	1											
16 [1]	0											
¼ [0] ¼ c_CLKRIO[0					10000 ps						
6_CLKRIOL	7 10000 ps					10000 ps						
						· · · · · · · · · · · · · · · · · · ·			\ 			
. Smai	rt co	ntroller										
ate tak	ole											
C .		D: 1: C 1	D: 1: 14/ 1	Б. І		1						
Current s	state	Direction South	Direction West	Delay	Sensors	s combinati	on					
CTODA		ہے۔ ما	na al	1 -								
STOP1		red	red	1 s								
LIECT CO		rad	Q MO O M	1.0	Cauth	- O \\/oct	1					
WEST_GO		red	green	4 s	South	= 0, West =	I					

WEST_WAIT

SOUTH_GO

SOUTH_WAIT

State diagram

STOP2

South West South West S-W 0-0 S-W 4 sec 1-0 0-0 4 sec 1-1 0-1 1-1 WestWait SouthGo South West South West 1 sec 2 sec Stop2 S-W 0-0 0-1 0-1 1-0 South 1-0 1-1 1-1 Listing of VHDL code of sequential process p_smart_traffic_fsm with syntax highlighting p_smart_traffic_fsm : process(clk) begin if rising_edge(clk) then if (reset = '1') then -- Synchronous reset s_cnt <= c_ZERO; -- Clear all bits elsif (s_en = '1') then -- Every 250 ms, CASE checks the value of the s_state -- variable and changes to the next state according -- to the delay value. case s_state is

-- If the current state is STOP1, then wait 1 sec

-- and move to the next GO_WAIT state.

if (s_cnt < c_DELAY_1SEC) then</pre>

-- Move to the next state

-- Reset local counter value

-- Count up to c_DELAY_1SEC

s_cnt <= s_cnt + 1;

s_state <= WEST_GO;</pre>

s_cnt <= c_ZERO;

when STOP1 =>

else

end if;

s_state <= STOP1;</pre>

end case;

end if; -- Rising edge

end process p_smart_traffic_fsm;

end if; -- Synchronous reset

end if;
when WEST_GO =>
Count up to c_DELAY_4SEC
<pre>if (s_cnt < c_DELAY_4SEC) then</pre>
s_cnt <= s_cnt + 1;
<pre>elsif(south_sensor = '0' and west_sensor = '0') then</pre>
s_state <= WEST_WAIT;
<pre>s_cnt <= c_ZERO; elsif(south_sensor = '0' and west_sensor = '1') then</pre>
s_state <= WEST_GO;
s_cnt <= c_ZERO;
<pre>elsif(south_sensor = '1' and west_sensor = '0') then</pre>
s_state <= WEST_WAIT;
<pre>s_cnt <= c_ZERO; elsif(south_sensor = '1' and west_sensor = '1') then</pre>
s_state <= WEST_WAIT;
s_cnt <= c_ZERO;
end if;
when WEST_WAIT =>
Count up to c_DELAY_2SEC if (s_cnt < c_DELAY_2SEC) then
s_cnt <= s_cnt + 1;
else
Move to the next state
s_state <= STOP2;
Reset local counter value
<pre>s_cnt <= c_ZERO; end if;</pre>
when STOP2 =>
Count up to c_DELAY_1SEC
<pre>if (s_cnt < c_DELAY_1SEC) then</pre>
<pre>s_cnt <= s_cnt + 1; else</pre>
Move to the next state
s_state <= SOUTH_GO;
Reset local counter value
s_cnt <= c_ZERO;
end if;
when SOUTH_GO =>
Count up to c_DELAY_4SEC
<pre>if (s_cnt < c_DELAY_4SEC) then</pre>
s_cnt <= s_cnt + 1;
<pre>elsif(south_sensor = '0' and west_sensor = '0') then s_state <= SOUTH_WAIT;</pre>
s_cnt <= c_ZERO;
elsif(south_sensor = '0' and west_sensor = '1') then
s_state <= SOUTH_WAIT;
s_cnt <= c_ZERO;
<pre>elsif(south_sensor = '1' and west_sensor = '0') then s_state <= SOUTH_GO;</pre>
s_cnt <= c_ZERO;
elsif(south_sensor = '1' and west_sensor = '1') then
s_state <= SOUTH_WAIT;
s_cnt <= c_ZERO;
end if;
when SOUTH_WAIT =>
Count up to c_DELAY_2SEC
<pre>if (s_cnt < c_DELAY_2SEC) then</pre>
s_cnt <= s_cnt + 1;
else Move to the port state
Move to the next state s_state <= STOP1;
Reset local counter value
s_cnt <= c_ZERO;
end if;
It is a good programming practice to use the
OTHERS clause, even if all CASE choices have been made.
when others =>
s state <= STOP1: