08-traffic_lights

My repository

My git - Tomáš Křička, 223283

1. Preparation tasks

State table

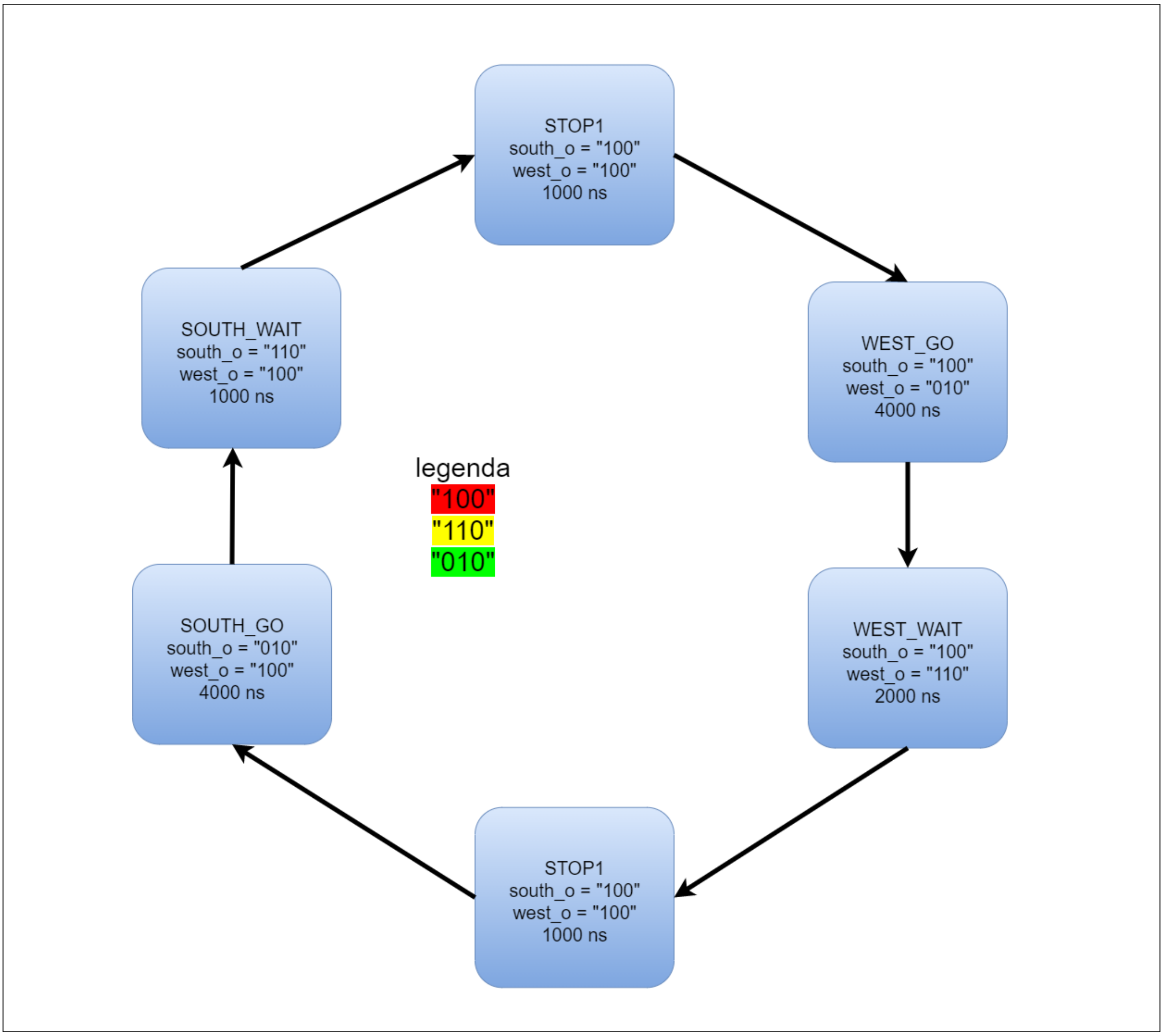
Input P	`0`	`0`	`1`	`1`	`0`	`1`	`0`	`1`	`1`	`1`	`1`	`0`	`0`	`1`	`1`	`1`
↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
State	A	A	B	C	C	D	A	B	C	D	B	B	B	C	D	B
Output R	`0`	`0`	`0`	`0`	`0`	`1`	`0`	`0`	`0`	`1`	`0`	`0`	`0`	`0`	`1`	`0`

connection of RGB

RGB LED	**Artix-7 pin names**	**Red**	**Yellow**	**Green**
LD16	N15, M16, R12	`1,0,0`	`1,1,0`	`0,1,0`
LD17	N16, R11, G14	`1,0,0`	`1,1,0`	`0,1,0`

2. Traffic light controller

State diagram



VHDL code `p_traffic_fsm`

```
p_traffic_fsm : process(clk)
   begin
       if rising_edge(clk) then
           if (reset = '1') then       -- Synchronous reset
               s_state <= STOP1 ;      -- Set initial state
               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.
                   when STOP1 =>
```

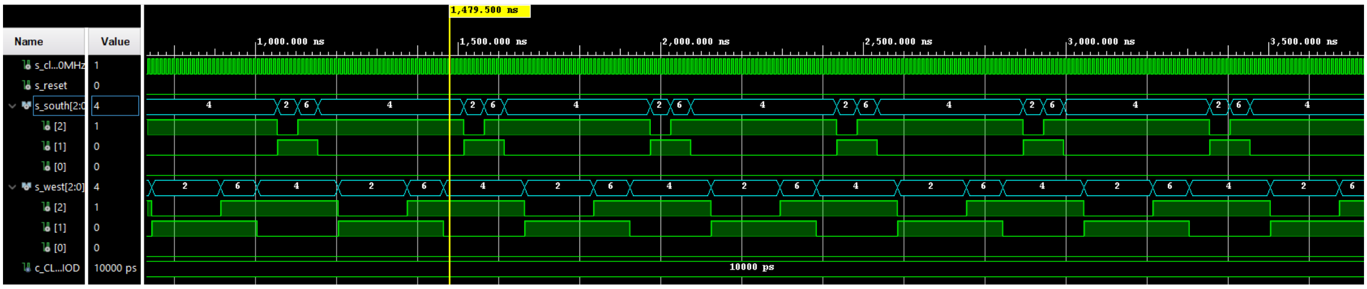
```
                    -- Count up to c_DELAY_1SEC
                    if (s_cnt < c_DELAY_1SEC) then
                        s_cnt <= s_cnt + 1;
                    else
                        -- Move to the next state
                        s_state <= WEST_GO;
                        -- Reset local counter value
                        s_cnt   <= c_ZERO;
                    end if;

                when WEST_GO =>
                    if (s_cnt < c_DELAY_4SEC) then
                        s_cnt <= s_cnt + 1;
                    else
                        s_state <= WEST_WAIT;
                        s_cnt <= c_ZERO;
                    end if;

                when WEST_WAIT =>
                    if (s_cnt < c_DELAY_2SEC) then
                        s_cnt <= s_cnt + 1;
                    else
                        s_state <= STOP2;
                        s_cnt <= c_ZERO;
                    end if;

                 when STOP2 =>
                    if (s_cnt < c_DELAY_1SEC) then
                        s_cnt <= s_cnt + 1;
                    else
                        s_state <= SOUTH_GO;
                        s_cnt <= c_ZERO;
                    end if;

                 when SOUTH_GO =>
                    if (s_cnt < c_DELAY_1SEC) then
                        s_cnt <= s_cnt + 1;
                    else
                        s_state <= SOUTH_WAIT;
                        s_cnt <= c_ZERO;
                    end if;

                 when SOUTH_WAIT =>
                    if (s_cnt < c_DELAY_1SEC) then
                        s_cnt <= s_cnt + 1;
                    else
                        s_state <= STOP1;
                        s_cnt <= c_ZERO;
                    end if;

                    -- WRITE YOUR CODE HERE


                -- It is a good programming practice to use the
```

```
                    -- OTHERS clause, even if all CASE choices have
                    -- been made.
                    when others =>
                        s_state <= STOP1;

                end case;
            end if; -- Synchronous reset
        end if; -- Rising edge
    end process p_traffic_fsm;
```

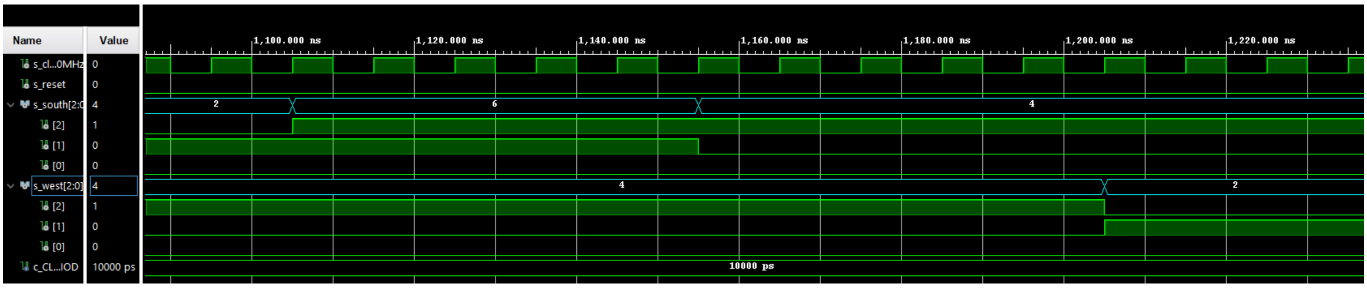
VHDL code `p_output_fsm`

```
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";
                west_o <= "010";
            when WEST_WAIT =>
                south_o <= "100";
                west_o <= "110";
            when STOP2 =>
                south_o <= "100";
                west_o <= "100";
            when SOUTH_GO =>
                south_o <= "010";
                west_o <= "100";
            when SOUTH_WAIT =>
                south_o <= "110";
                west_o <= "100";

            when others =>
                south_o <= "100";   -- Red
                west_o  <= "100";   -- Red
        end case;
    end process p_output_fsm;

end architecture Behavioral;
```

Waverorms



Waverorms



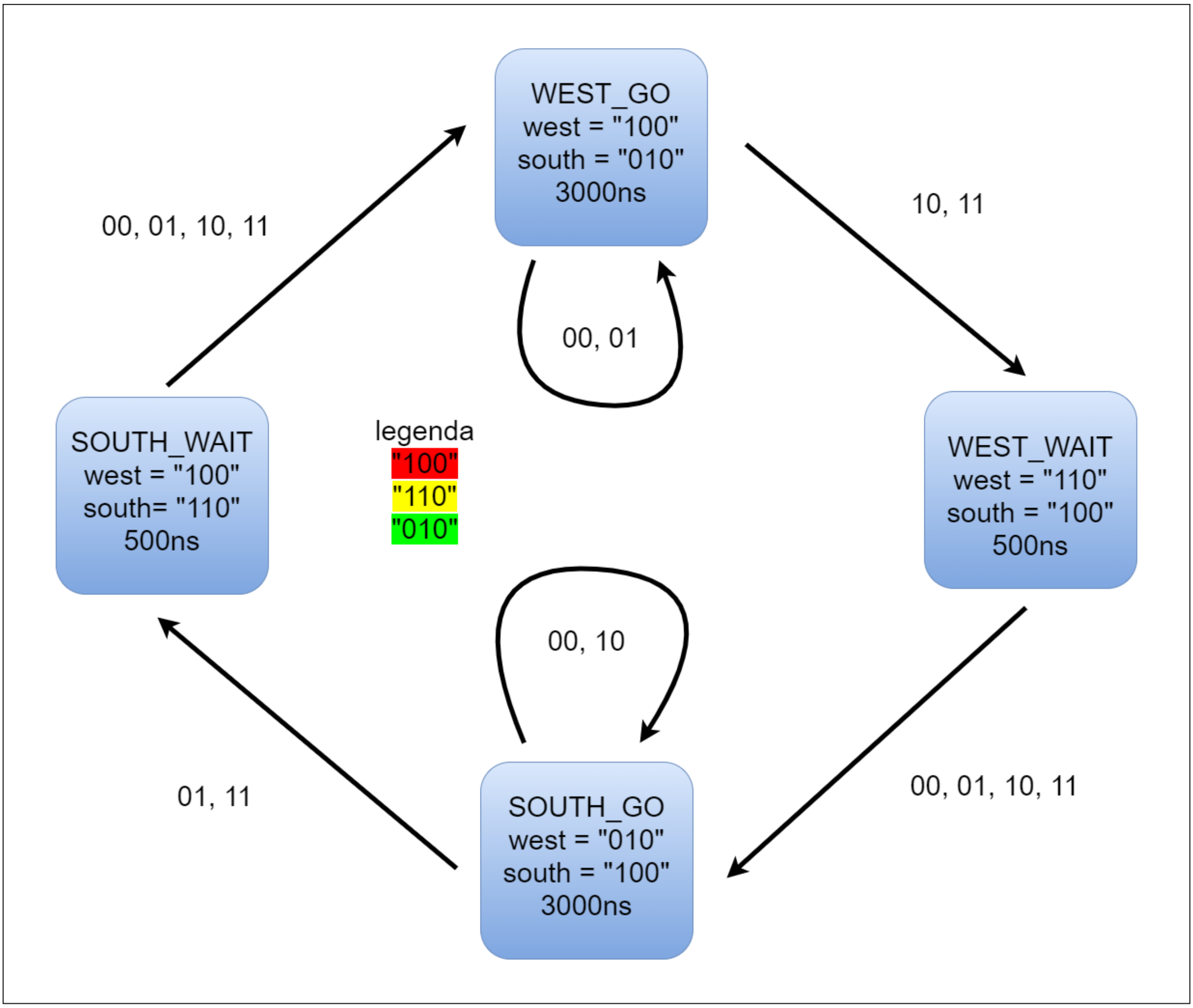
responds on rising edge

3.Smart controller

State table

Current state	Direction south	Direction west	No cars (00)	Cars to west (01)	Cars to south (10)	Cars both direction (11)
GO_SOUTH	green	red	GO_SOUTH	WAIT_SOUTH	GO_SOUTH	WAIT_SOUTH
WAIT_SOUTH	yellow	red	GO_WEST	GO_WEST	GO_WEST	GO_WEST
GO_WEST	red	green	GO_WEST	GO_WEST	WAIT_WEST	WAIT_WEST
WAIT_WEST	red	yellow	GO_SOUTH	GO_SOUTH	GO_SOUTH	GO_SOUTH

State diagram

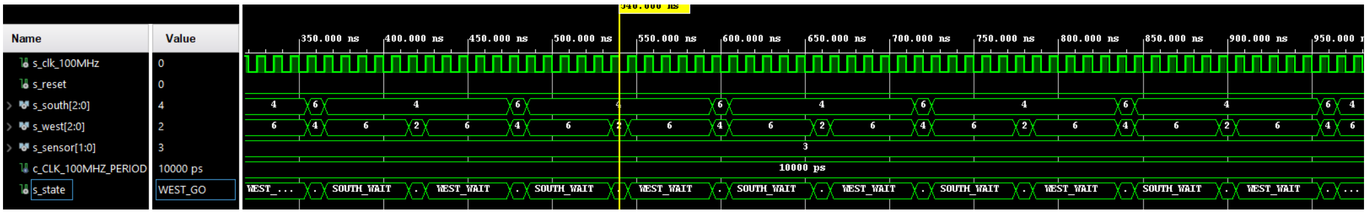


VHDL code `p_smart_traffic_fsm`

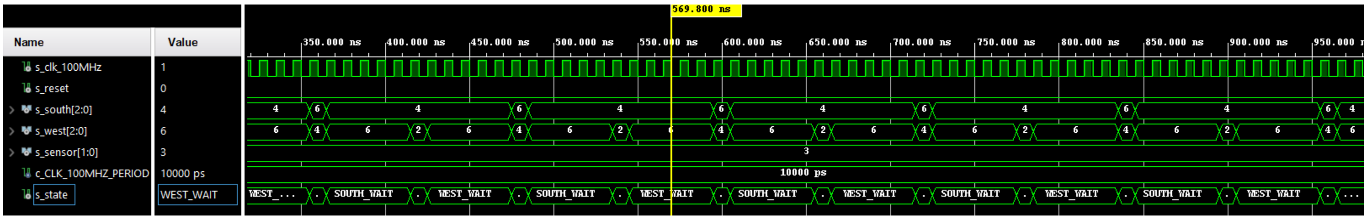
```
    p_smart_traffic_fsm : process(clk)
    begin
        if rising_edge(clk) then
            if (reset = '1') then       -- Synchronous reset
                s_state <= WEST_GO ;      -- Set initial state
                s_cnt   <= c_ZERO;      -- Clear all bits

            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.
                    ---------------------
                   when WEST_GO =>
                        if (s_cnt < c_DELAY_2SEC and (sensor_i = "00" or sensor_i = "01")) then
```

```
                            s_cnt <= s_cnt + 1;
                        else
                            s_state <= WEST_WAIT;
                            s_cnt <= c_ZERO;
                        end if;

                   when WEST_WAIT =>
                        if (s_cnt < c_DELAY_1SEC) then
                            s_cnt <= s_cnt + 1;
                        else
                            s_state <= SOUTH_GO;
                            s_cnt <= c_ZERO;
                        end if;



                   when SOUTH_GO =>
                        if (s_cnt < c_DELAY_2SEC and (sensor_i = "00" or sensor_i = "10")) then
                            s_cnt <= s_cnt + 1;
                        else
                            s_state <= SOUTH_WAIT;
                            s_cnt <= c_ZERO;
                        end if;

                   when SOUTH_WAIT =>
                        if (s_cnt < c_DELAY_1SEC) then
                            s_cnt <= s_cnt + 1;
                        else
                            s_state <= WEST_GO;
                            s_cnt <= c_ZERO;
                        end if;

                    when others =>
                        s_state <= WEST_GO;

                end case;
            end if; -- Synchronous reset
        end if; -- Rising edge
    end process p_smart_traffic_fsm;
```

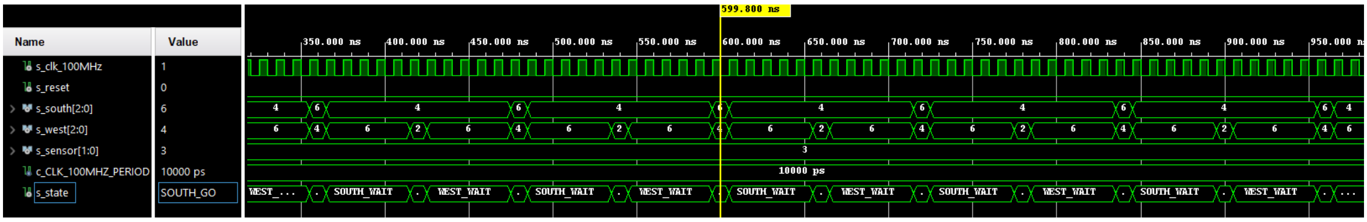
Waveforms



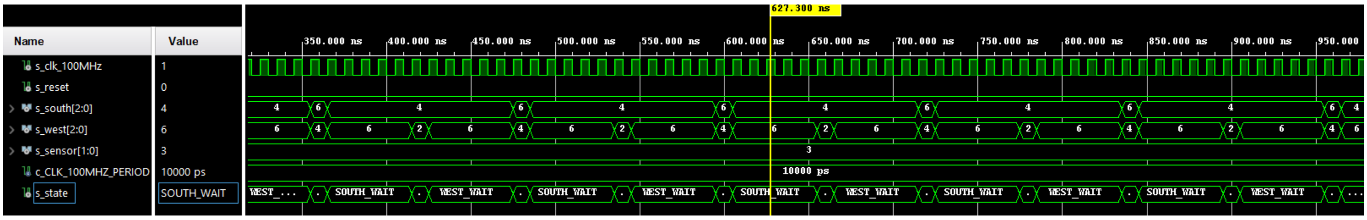
Name	Value
s_clk_100MHz	0
s_reset	0
s_south[2:0]	4
s_west[2:0]	2
s_sensor[1:0]	3
c_CLK_100MHZ_PERIOD	10000 ps
s_state	WEST_GO



Name	Value
s_clk_100MHz	1
s_reset	0
s_south[2:0]	4
s_west[2:0]	6
s_sensor[1:0]	3
c_CLK_100MHZ_PERIOD	10000 ps
s_state	WEST_WAIT



Name	Value
s_clk_100MHz	1
s_reset	0
s_south[2:0]	6
s_west[2:0]	4
s_sensor[1:0]	3
c_CLK_100MHZ_PERIOD	10000 ps
s_state	SOUTH_GO



Name	Value
s_clk_100MHz	1
s_reset	0
s_south[2:0]	4
s_west[2:0]	6
s_sensor[1:0]	3
c_CLK_100MHZ_PERIOD	10000 ps
s_state	SOUTH_WAIT