TaaviSalum / **Digital-electronics-1**

main

Digital-electronics-1 / Labs / 08-traffic_lights /

TaaviSalum ...		2 minutes ago
..		
Pictures		21 minutes ago
traffic		2 minutes ago
README.md		2 minutes ago

Digital-electronics-1

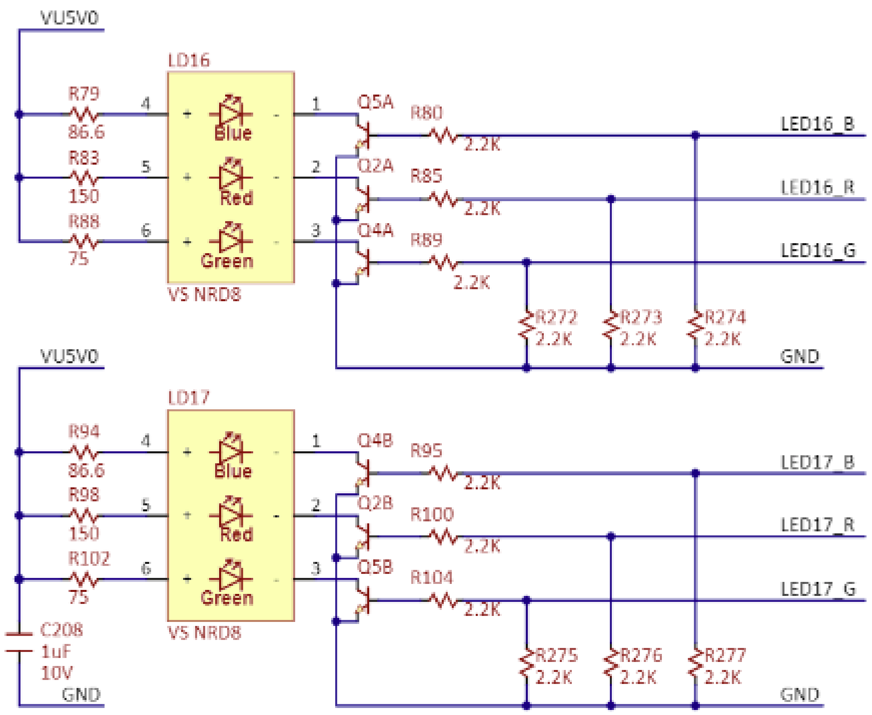
Laboratory #8

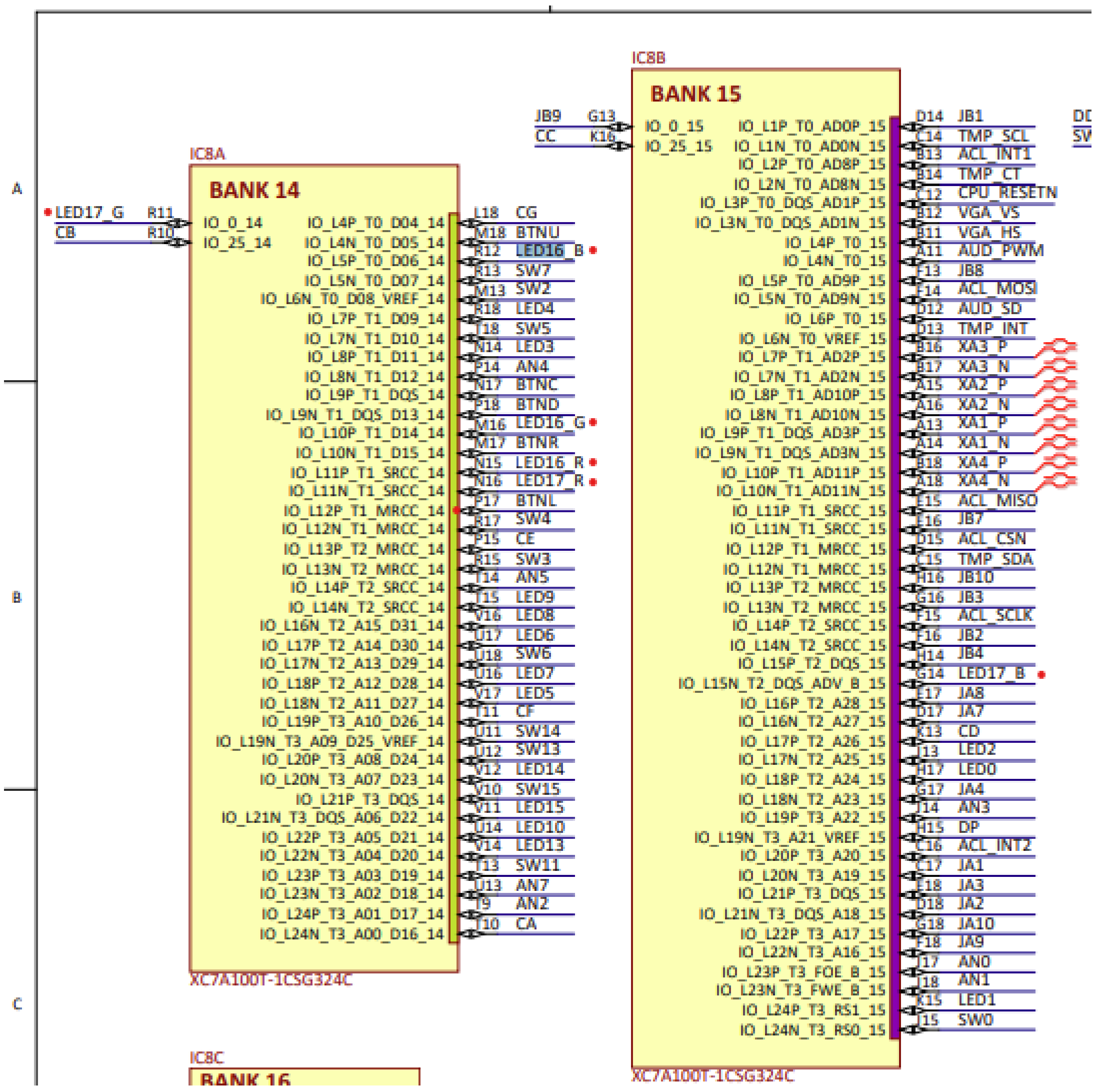
Exercise 1: Preparation tasks

Completed state table

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

Connection of RGB LEDs on Nexys A7 board and completed table with colour settings

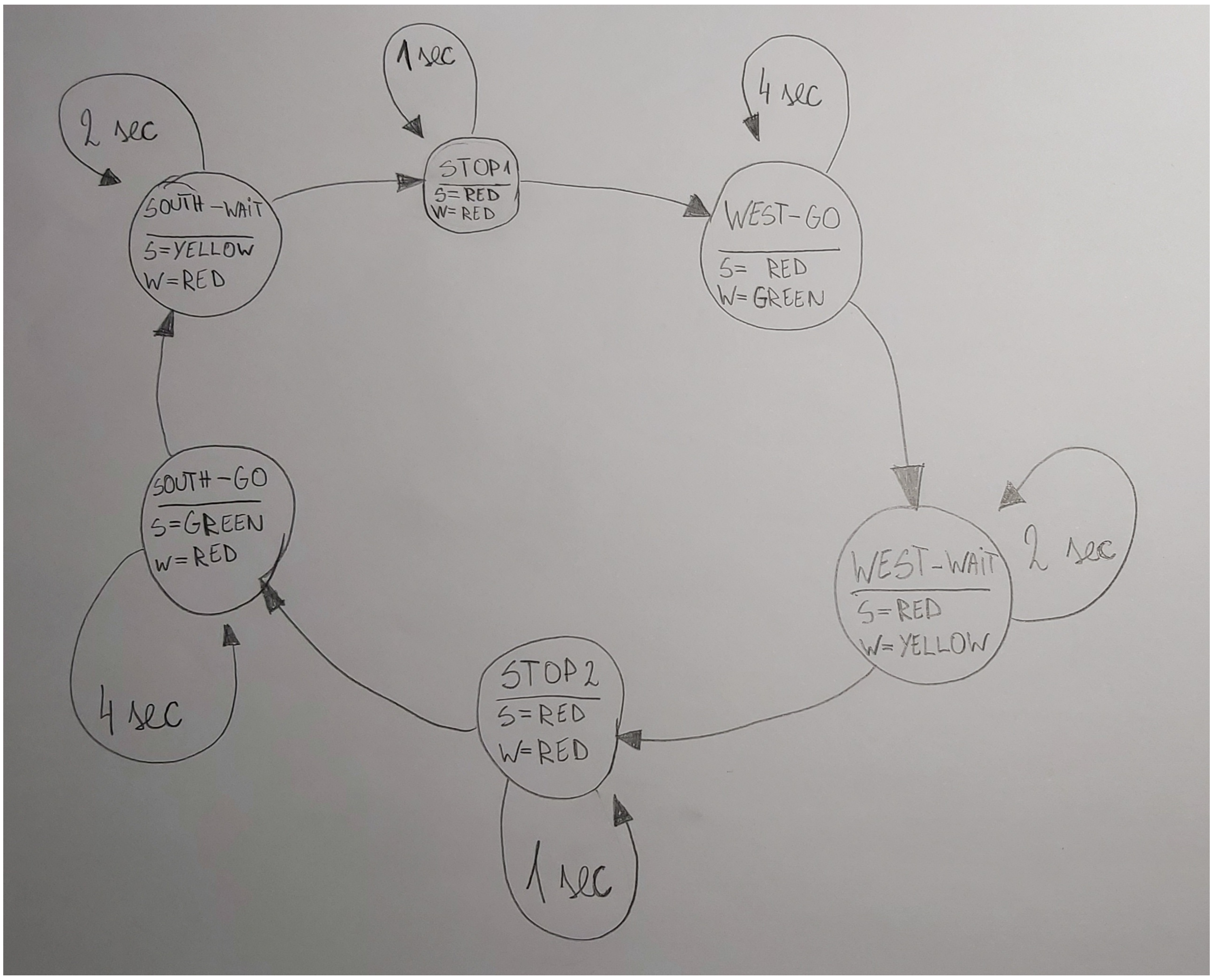




RGB LED	Artix-7 pin names	Red	Yellow	Green
LD16	N15, M16, R12	`1,0,0`	`1,1,0`	`0,1,0`
LD17	N16, R11, G14	`1,0,0`	`1,1,0`	`0,1,0`

Exercise 2: Traffic light controller

State diagram



VHDL code of the sequential process `p_traffic_fsm` :

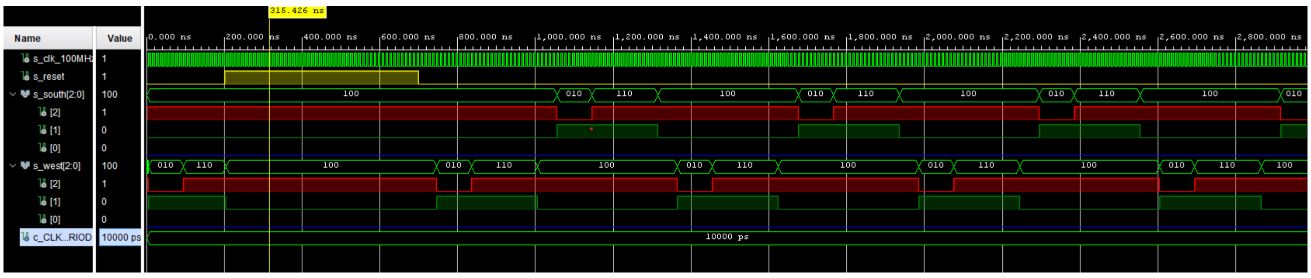
```
    p_traffic_fsm : process(clk)
    begin
        if rising_edge(clk) then
            if (reset = '1') then       -- Synchronous reset
                s_state <= STOP1 ;      -- Set initial state
                s_cnt   <= c_ZERO;      -- Clear all bits
            elsif (s_en = '1') then
                -- 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_2SEC) 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_4SEC) 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_2SEC) 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_4SEC) then
                            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;
```

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

README.md

```vhdl
p_output_fsm : process(s_state)
begin
    case s_state is
        when STOP1 =>
            south_o <= "100";   -- Red
            west_o  <= "100";   -- Red
        when WEST_GO =>
            south_o <= "100";   -- Red
            west_o  <= "010";   -- Green
        when WEST_WAIT =>
            south_o <= "100";   -- Red
            west_o  <= "110";   -- Yellow
        when STOP2 =>
            south_o <= "100";   -- Red
            west_o  <= "100";   -- Red
        when SOUTH_GO =>
            south_o <= "010";   -- Green
            west_o  <= "100";   -- Red
        when SOUTH_WAIT =>
            south_o <= "110";   -- Yellow
            west_o  <= "100";   -- Red
        when others =>
            south_o <= "100";   -- Red
            west_o  <= "100";   -- Red
    end case;
end process p_output_fsm;
```

Simulated time waveforms

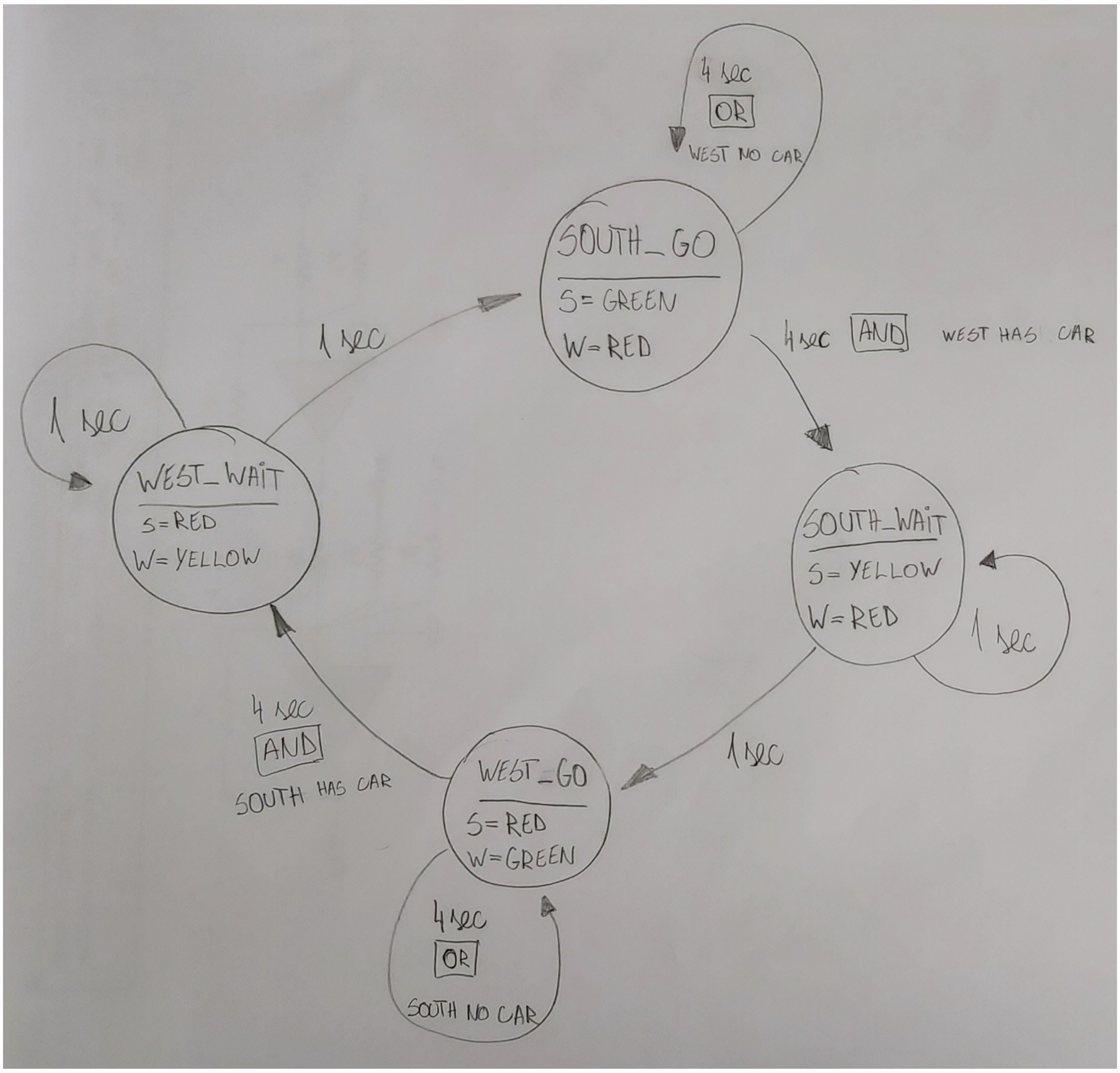


Exercise 3: Smart controller

State table

Input	No cars	`West_i`	`West_i`	`West_i`	No cars	`South_i`
Delay	4 seconds	1 second	4 seconds	4 seconds	4 seconds	1 second
State	`South_go`	`South_wait`	`West_go`	`West_go`	`West_go`	`West_wait`
Output	S: Green; W: Red	S: Yellow; W: Red	S: Red; W: Green	S: Red; W: Green	S: Red; W: Green	S: Red; W: Yellow

State diagram



VHDL code of the sequential process `p_smart_traffic_fsm` :

```vhdl
    p_smart_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
                    when South_go =>
                        -- Count up to c_DELAY_4SEC
                        if (s_cnt < c_DELAY_4SEC) then
                                s_cnt <= s_cnt + 1;
                        elsif (west_i = '1') then
                            -- Move to the next state
                            s_state <= South_wait;
                            -- Reset local counter value
                            s_cnt   <= c_ZERO;
                        end if;
                    when South_wait =>
                        -- 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 =>
                        -- Count up to c_DELAY_4SEC
                        if (s_cnt < c_DELAY_4SEC) then
                                s_cnt <= s_cnt + 1;
                        elsif (South_i = '1') then
                            -- Move to the next state
                            s_state <= West_wait;
                            -- Reset local counter value
                            s_cnt   <= c_ZERO;
                        end if;
                    when West_wait =>
                        -- 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 <= South_go;
                            -- 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 <= South_go;
                end case;
            end if; -- Synchronous reset
        end if; -- Rising edge
    end process p_smart_traffic_fsm;
```