xmarek71 / Digital-electronics1

MIT License

0 stars 0 forks

Star Unwatch

Code Issues Pull requests Actions Projects Wiki Security

main Digital-electronics1 / Labs / 08-traffic_lights /

xmarek71 Update README.md ... 25 seconds ago History

..

README.md 25 seconds ago

README.md

Cvičenie 8

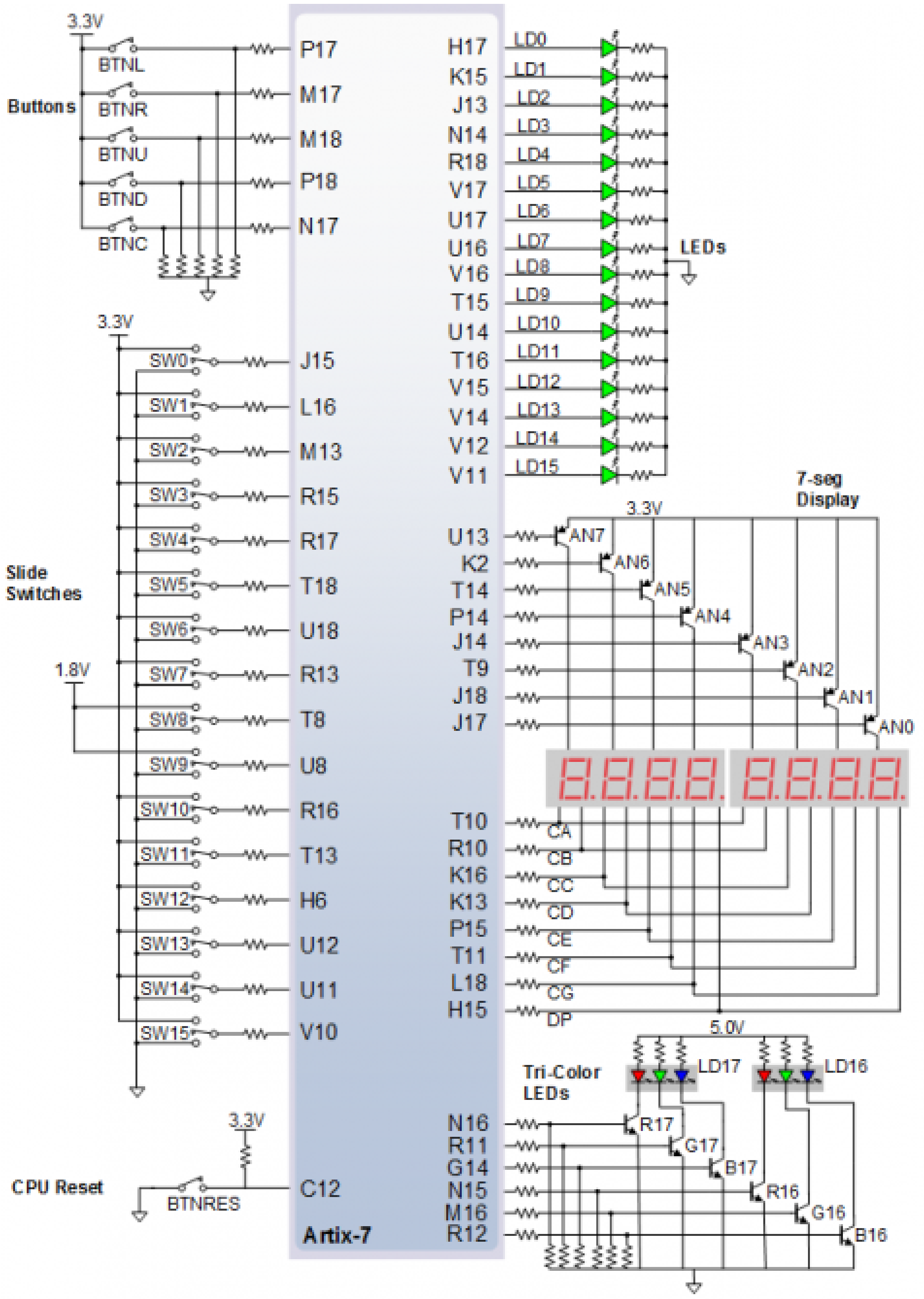
Preparation tasks

Table with the state names and output values accoding to the given inputs

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
Output R	`0`	`0`	`0`	`0`	`0`	`1`	`0`	`0`	`0`	`1`	`0`	`0`

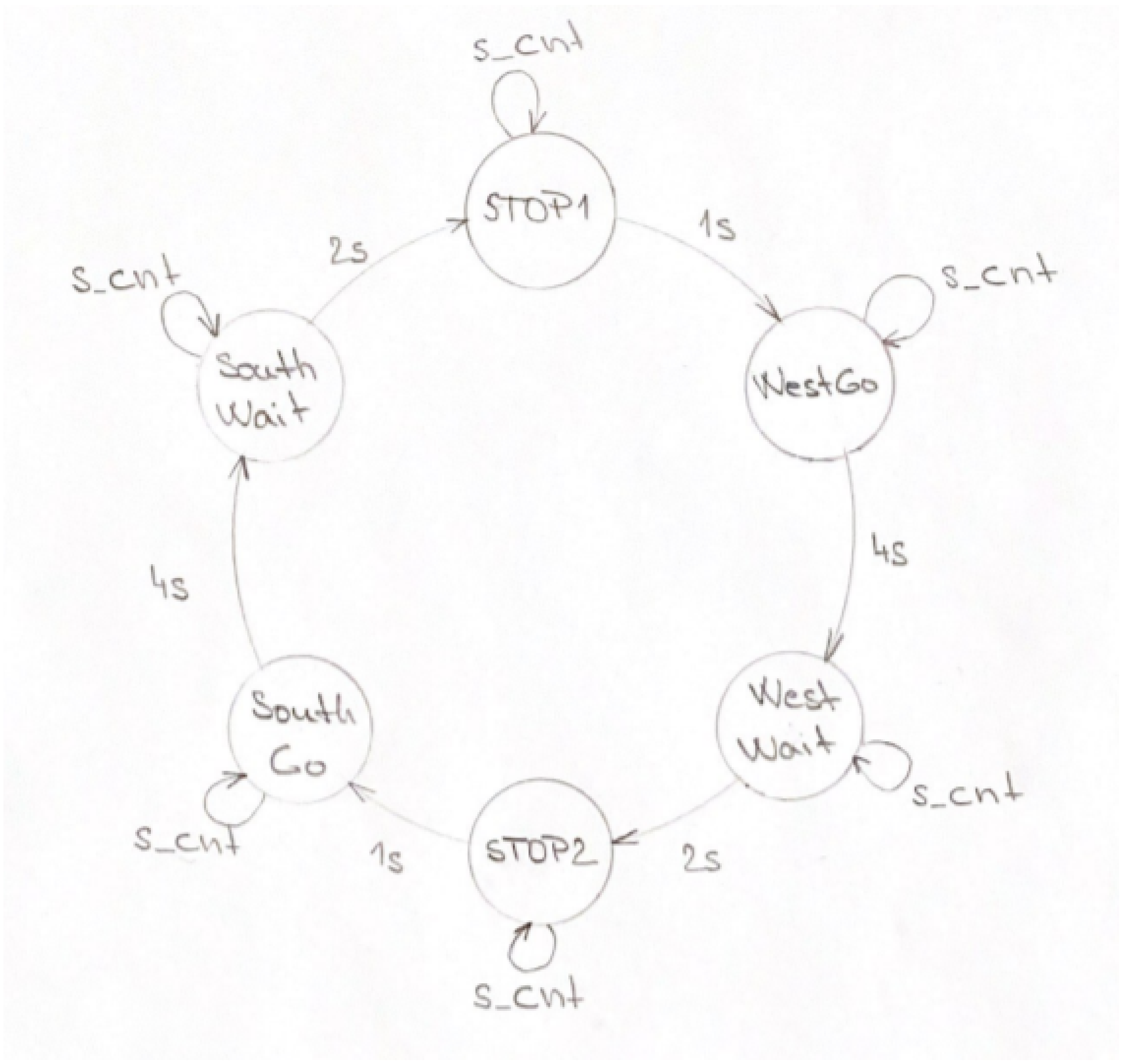
Figure with connection of RGB LEDs on Nexys A7 board

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`



Traffic light controller

State diagram



Listing of VHDL code of sequential process p_traffic_fsm

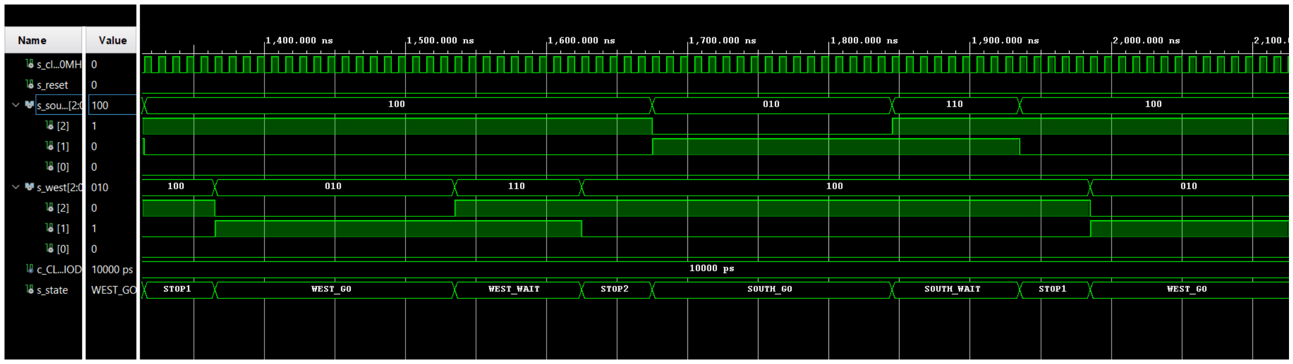
```
p_traffic_fsm : process(clk)
begin
    if rising_edge(clk) then
        if (reset = '1') then       -- Synchronous reset
            s_state <= STOP1 ;      -- Set initial state
            s_cnt   <= c_ZERO;      -- Clear all bits

        elsif (s_en = '1') then
            case s_state is

                when STOP1 =>
                    if (s_cnt < c_DELAY_1SEC) then
                        s_cnt <= s_cnt + 1;
                    else
                        s_state <= WEST_GO;
                        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_4SEC) 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_2SEC) then
                        s_cnt <= s_cnt + 1;
                    else
                        s_state <= STOP1;
                        s_cnt   <= c_ZERO;
                    end if;

                when others =>
                    s_state <= STOP1;

            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

```
    p_output_fsm : process(s_state)
    begin
        case s_state is
            when STOP1 =>
                south_o <= c_RED;
                west_o  <= c_RED;
            when WEST_GO =>
                south_o <= c_RED;
                west_o  <= c_GREEN;
            when WEST_WAIT =>
                south_o <= c_RED;
                west_o  <= c_YELLOW;
            when STOP2 =>
                south_o <= c_RED;
                west_o  <= c_RED;
            when SOUTH_GO =>
                south_o <= c_GREEN;
                west_o  <= c_RED;
            when SOUTH_WAIT =>
                south_o <= c_YELLOW;
                west_o  <= c_RED;
            when others =>
                south_o <= c_RED;
                west_o  <= c_RED;
        end case;
    end process p_output_fsm;
```

Screenshot(s) of the simulation

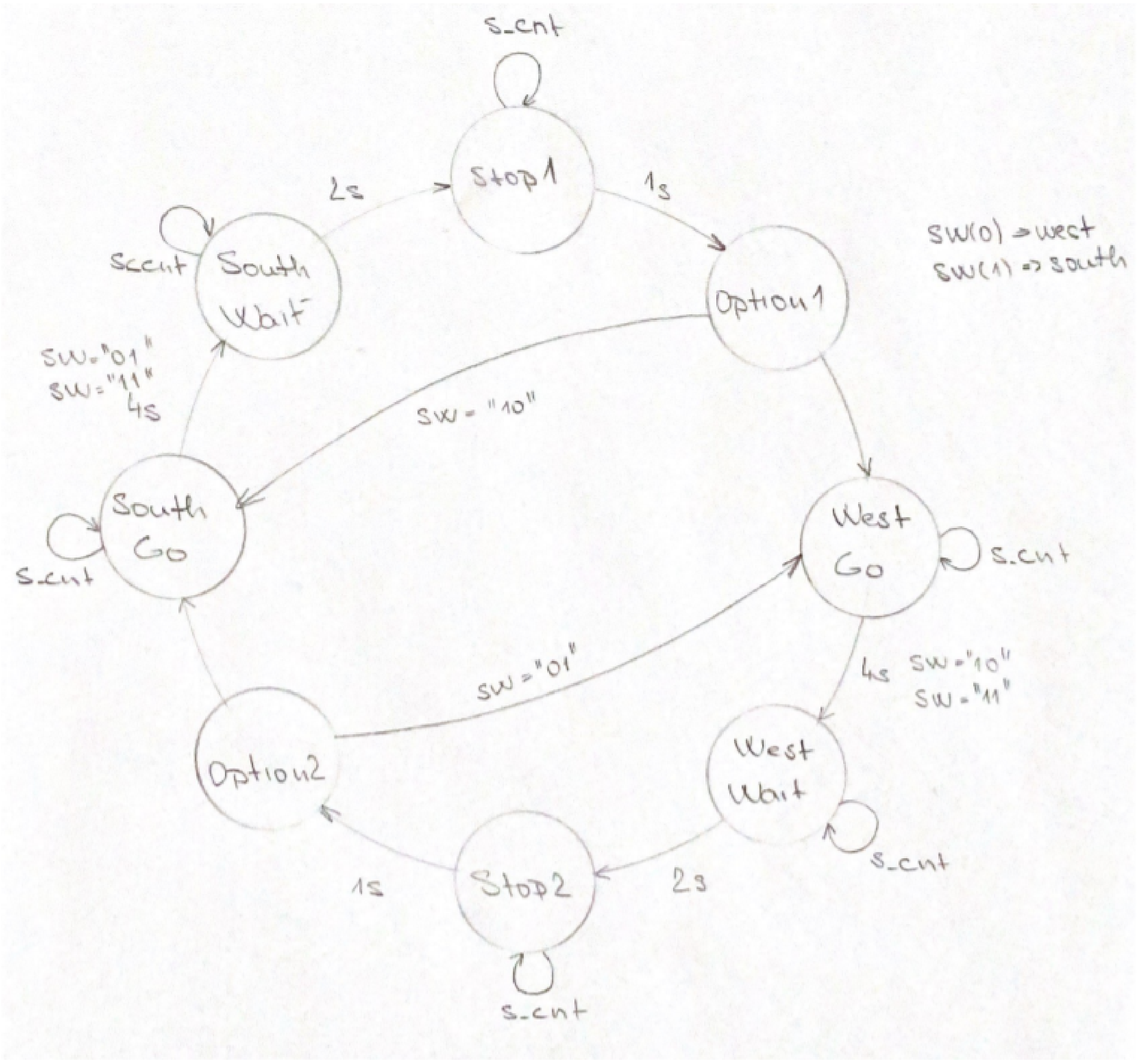


Smart controller

State table

Current state	Direction South	Direction West	Delay	No Cars	Cars to West	Cars Sou
`STOP1`	red	red	1 sec	`OPTION1`	`OPTION1`	`OPTIO`
`WEST_GO`	red	green	4 sec	`WEST_GO`	`WEST_GO`	`WEST_W`
`WEST_WAIT`	red	yellow	2 sec	`STOP2`	`STOP2`	`STOP`
`STOP2`	red	red	1 sec	`OPTION2`	`OPTION2`	`OPTIO`
`SOUTH_GO`	green	red	4 sec	`SOUTH_GO`	`SOUTH_WAIT`	`SOUTH_`
`SOUTH_WAIT`	yellow	red	2 sec	`STOP1`	`STOP1`	`STOP`
`OPTION1`	red	red	0 sec	`WEST_GO`	`WEST_GO`	`SOUTH_`
`OPTION2`	red	red	0 sec	`SOUTH_GO`	`WEST_GO`	`SOUTH_`

State diagram



Listing of VHDL code of sequential process p_smart_traffic_fsm

```
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

            case s_state is

                when STOP1 =>
                    if (s_cnt < c_DELAY_1SEC) then
                        s_cnt <= s_cnt + 1;
                    else
                        s_state <= OPTION1;
                        s_cnt   <= c_ZERO;
                    end if;

                 when OPTION1 =>

                    if(SW = "00" OR SW = "01" or SW = "11") then
                        s_state <= WEST_GO;
                    elsif(SW = "10") then
                        s_state <= SOUTH_GO;
                    end if;

                when WEST_GO =>

                    if (s_cnt < c_DELAY_4SEC) then
                        s_cnt <= s_cnt + 1;
                    elsif(SW = "00" or SW = "01") then
                        s_state <= WEST_GO;
                        s_cnt   <= c_ZERO;
                    elsif(SW = "10" or SW = "11") then
                        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 <= OPTION2;
                        s_cnt   <= c_ZERO;
                    end if;

                    when OPTION2 =>

                    if(SW = "00" OR SW = "10" or SW = "11") then
                        s_state <= SOUTH_GO;
                    elsif(SW = "01") then
                        s_state <= WEST_GO;
                    end if;

                    when SOUTH_GO =>

                    if (s_cnt < c_DELAY_4SEC) then
```

```
                        s_cnt <= s_cnt + 1;
                    elsif(SW = "00" or SW = "10") then
                        s_state <= SOUTH_GO;
                        s_cnt   <= c_ZERO;
                    elsif(SW = "01" or SW = "11") then
                        s_state <= SOUTH_WAIT;
                        s_cnt   <= c_ZERO;
                    end if;

                    when SOUTH_WAIT =>

                    if (s_cnt < c_DELAY_2SEC) then
                        s_cnt <= s_cnt + 1;
                    else
                        s_state <= STOP1;
                        s_cnt   <= c_ZERO;
                    end if;

                when others =>
                    s_state <= STOP1;

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