


 xnanko00 ↑ ...

20 seconds ago  History

..

 README.md

20 seconds ago

 README.md

1. Cvičenie

Preparation tasks

Completed state table

| | | | | | | | | | | | | | | | | |
|----------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Input P | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| Clock | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ |
| 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 |

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

Table with connection of RGB LEDs

| Button | Connection |
|------------|------------|
| LD16 (R16) | N15 |
| LD16 (G16) | M16 |
| LD16 (B16) | R12 |
| LD17 (R17) | N16 |
| LD17 (G17) | R11 |
| LD17 (B17) | G14 |

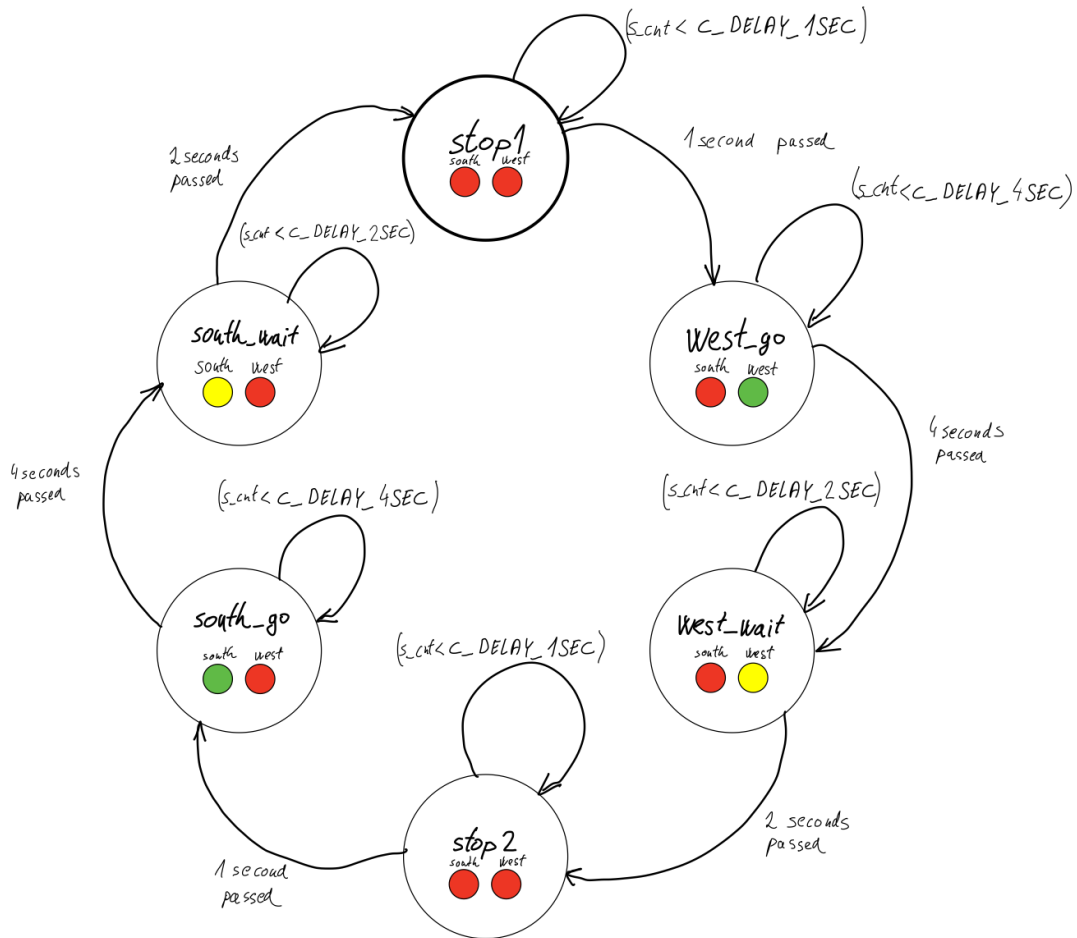
Completed table with color settings

| 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 |

2. Cvičenie

Traffic light controller

State diagram



VHDL code of process (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_4SEC) then
            s_cnt <= s_cnt + 1;
        else
            s_state <= SOUTH_WAIT;
            s_cnt    <= c_ZERO;
        end if;

    when SOUTH_WAIT =>
        if (unsigned(s_cnt) < c_DELAY_2SEC) 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;

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

```

VHDL code of process (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";    -- Red        (RGB = 100)
            west_o  <= "010";    -- Green      (RGB = 010)
        when WEST_WAIT =>
            south_o <= "100";    -- Red        (RGB = 100)
    end case;
end process;

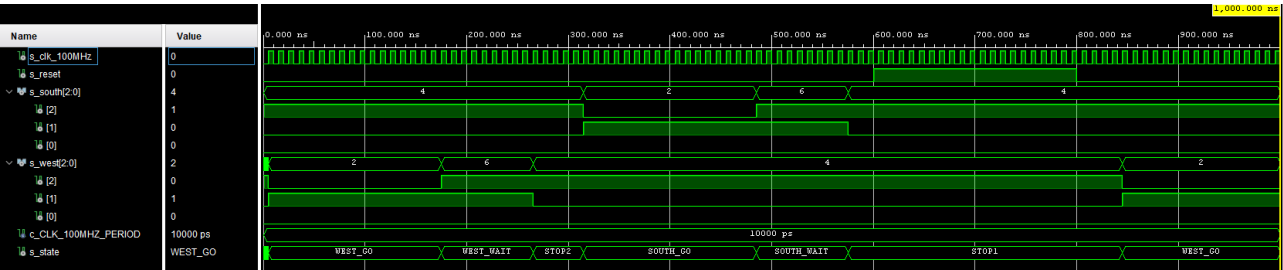
```

```

        west_o <= "110";    -- Yellow    (RGB = 110)
    when STOP2 =>
        south_o <= "100";    -- Red      (RGB = 100)
        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 with waveforms



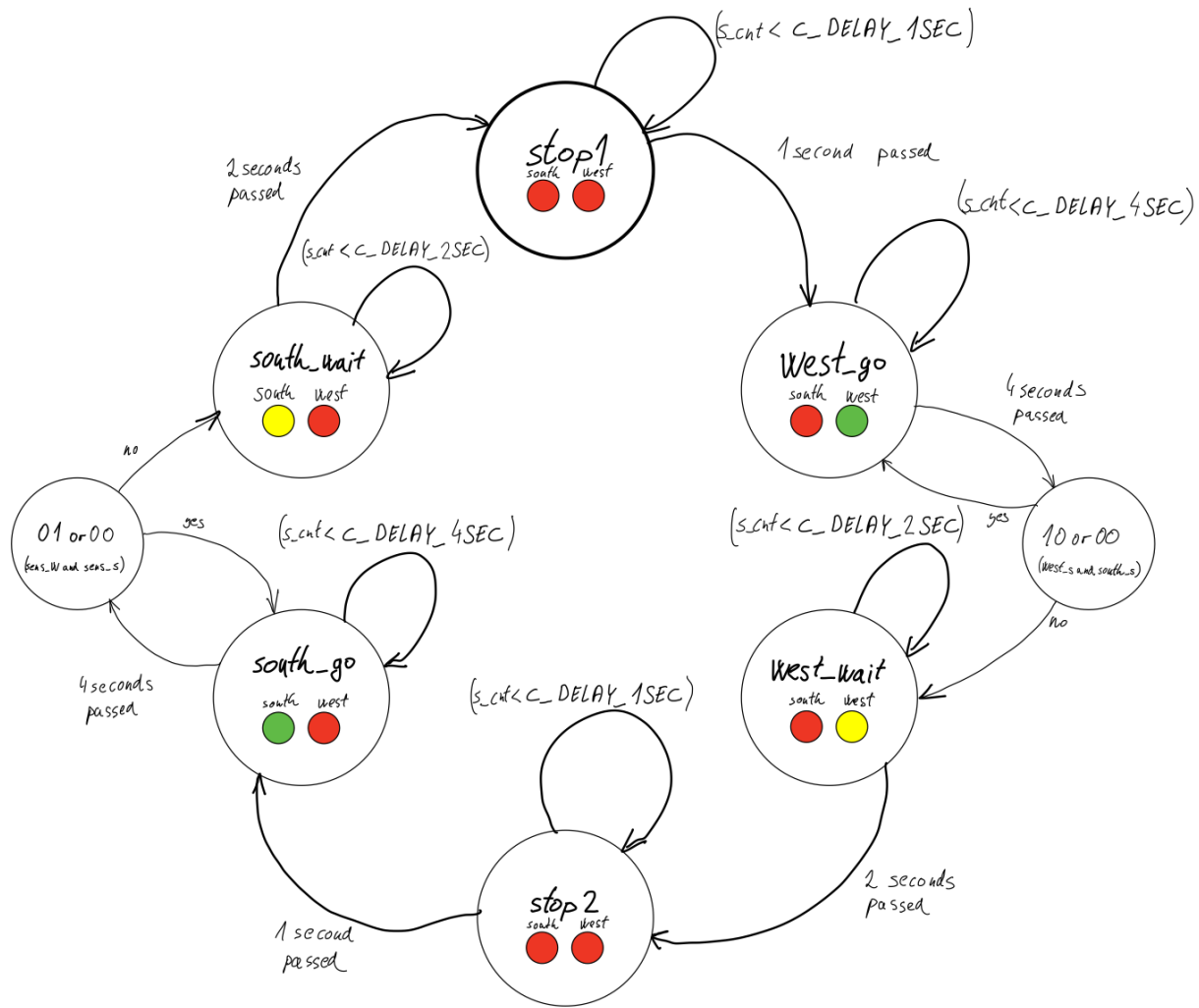
3. Cvičenie

Smart controller

State table

| Current state | Direction South | Direction West | Delay | Smart Controller |
|---------------|-----------------|----------------|-------|----------------------|
| STOP1 | red | red | 1 sec | n/a |
| WEST_GO | red | green | 4 sec | 10 or 11 => west go |
| WEST_WAIT | red | yellow | 2 sec | n/a |
| STOP2 | red | red | 1 sec | n/a |
| SOUTH_GO | green | red | 4 sec | 01 or 00 => south go |
| SOUTH_WAIT | yellow | red | 2 sec | n/a |

State diagram



VHDL code of (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
            -- 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;
                    
```

```

        elsif((sens_w = '1' and sens_s '0') or (sens_w = '0' and sens_s '0')) then
            s_state <= WEST_GO;
        else
            s_state <= WEST_WAIT;
        end if;
        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;
        elsif((sens_w = '0' and sens_s '1' ) or (sens_w = '0' and sens_s '0')) then
            s_state <= SOUTH_GO;
        else
            s_state <= SOUTH_WAIT;
        end if;
        s_cnt    <= c_ZERO;
    end if;

    when SOUTH_WAIT =>
        if (unsigned(s_cnt) < c_DELAY_2SEC) 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;

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

```