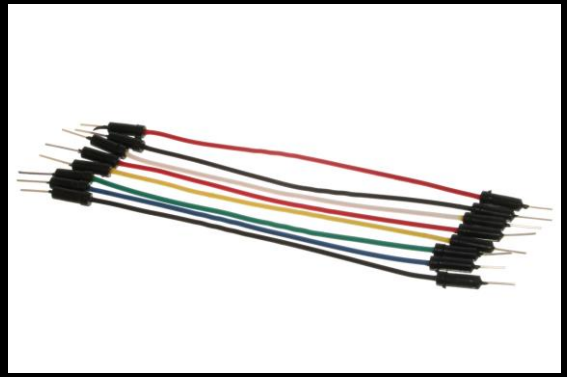
Traffic Light Controller

By: Austin Berger, Tyler Kavanagh, Remington Davids, Santiago Villarreal

Description

- 4-way intersection traffic light controller
- Controlled by a manual sensor
- FSM and Counter
- Used VHDL code and used a Nexys A7 Artix 100T fpga board
- 12 leds (Green, Red, and Yellow)
- 12 resistors
- Breadboards
- wires

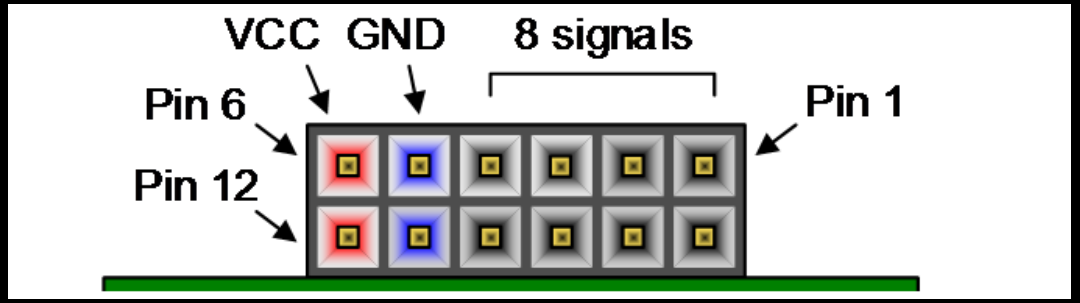


Introduction

- Traffic lights are everywhere and used to flow traffic efficiently
- We used a manual sensor on this project to make our traffic light more efficiently instead of constantly following the same cycle
- One side of the road will always be given a greenlight and then the other side would be a red light. When a car is approaching the lights will change. So to show this we made it a manual switch

Circuit implementation

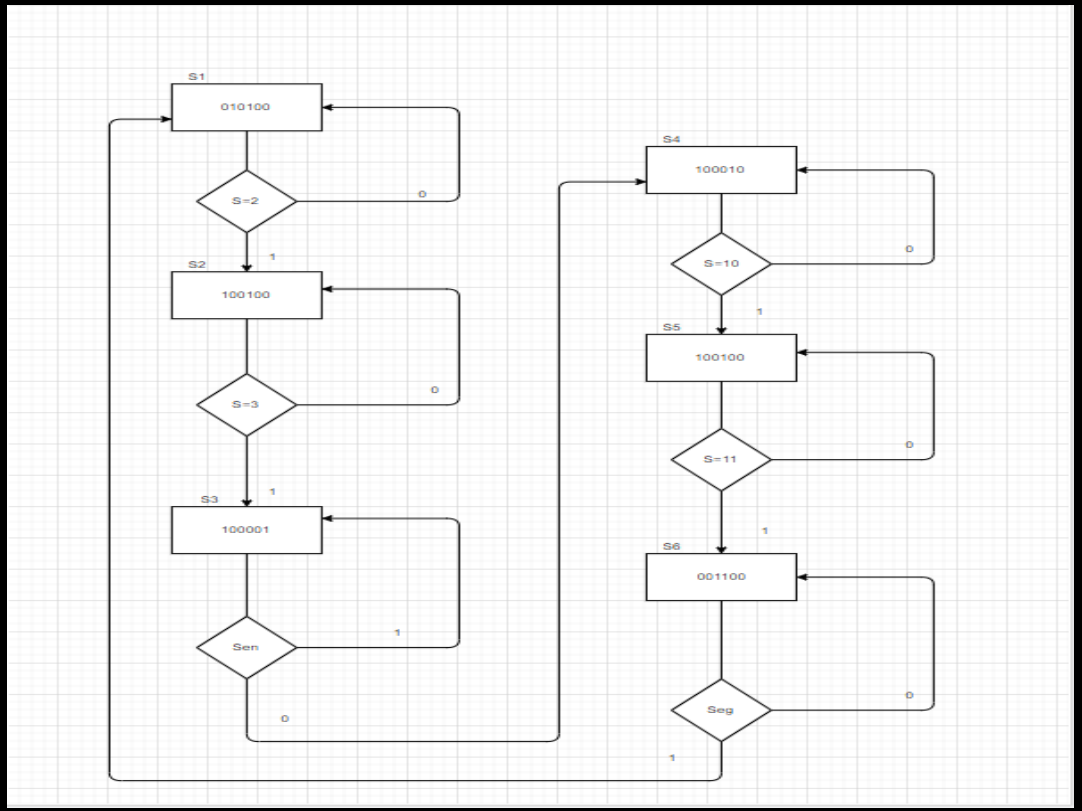
The Pmod port that was used on this project was JA which has 8 logic signals. We wired the leds to the JA pins 1,2,3,4 and 7,8, and they were used as logic signals to the leds. We also used GND Pin to ground our leds. On a pmod port they have two grounds and Pin 5 and Pin 11. Also on a pmod there 2 port which gives us 3.3V VCC signals and those pin are pin 6 and 12. Shown in the picture below



State Table

North/south			East/west			state
R	Y	G	R	Y	G	
0	0	1	1	0	0	S1
0	1	0	1	0	0	S2
1	0	0	1	0	0	S3
1	0	0	0	0	1	S4
1	0	0	0	1	0	S5
1	0	0	1	0	0	S6

State Diagram

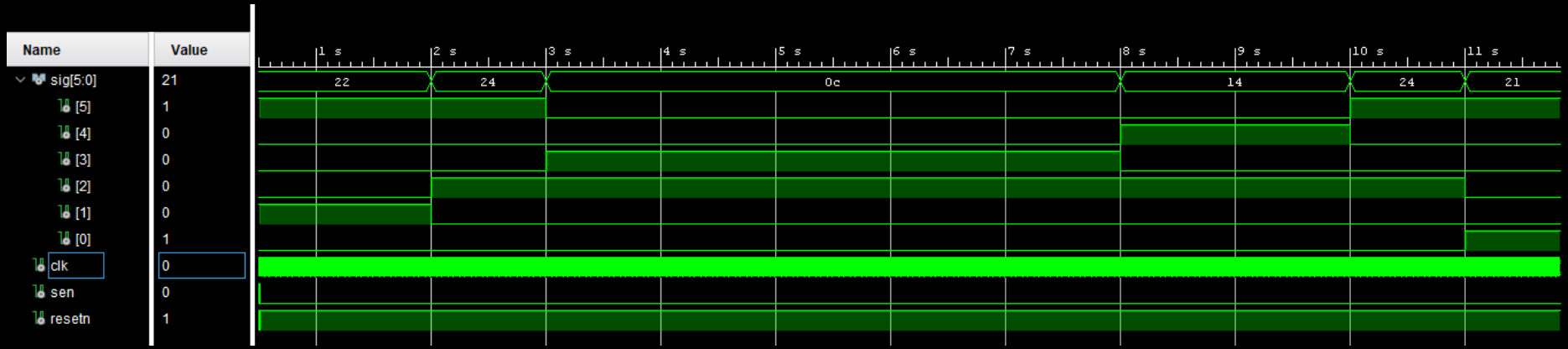


Design and Implementation

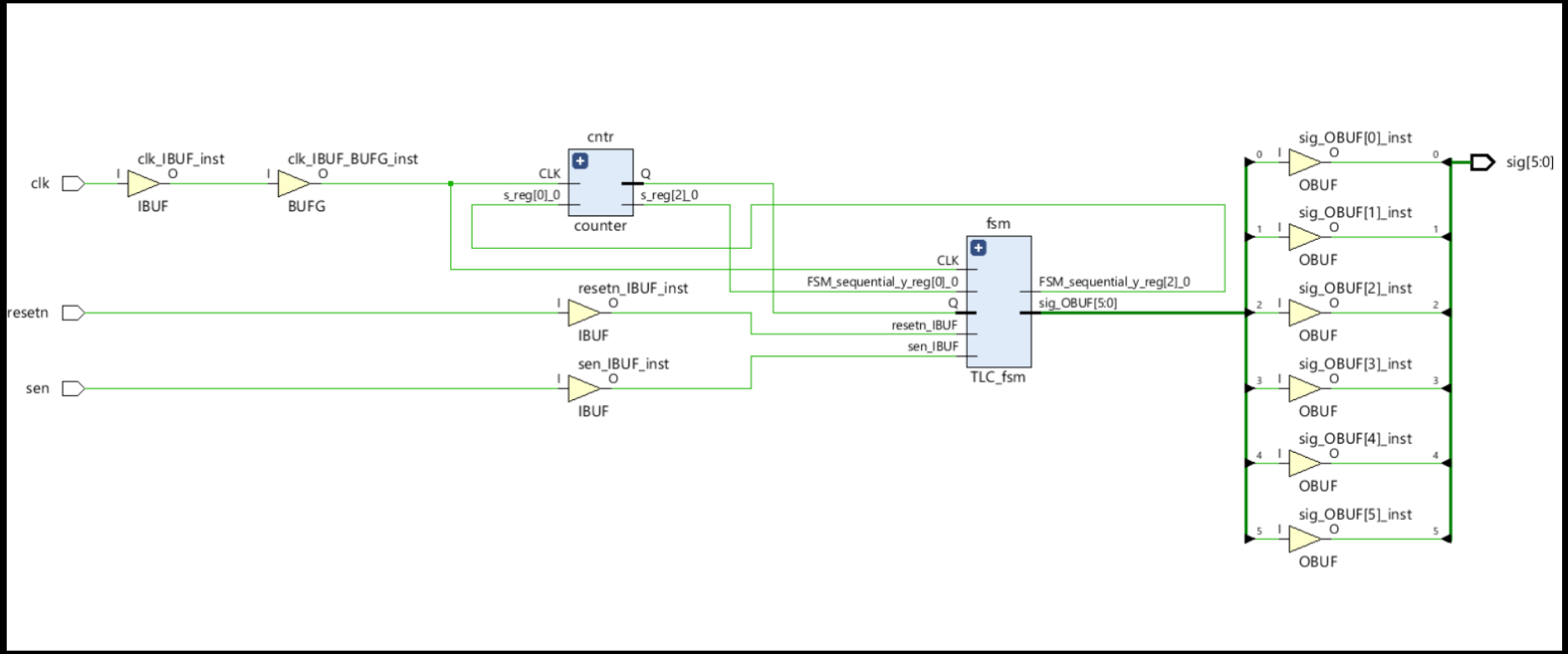
```
architecture behavioral of counter is


    signal ticks : std_logic_vector (26 downto 0):= "00000000000000000000000000";
    signal s : std_logic_vector (3 downto 0) := "0000";


begin

    seconds <= s;


    process(clk) is
    begin
        if (clk'event and clk = '1') then

            if resetn = '0' or resetcounter = '1' then
                ticks   <= "00000000000000000000000000";
                s <= "0000";

            end if;

            if ticks = "10111110101111000010000000" then

                ticks <= "00000000000000000000000000";
                s <= s + 1;


            else

                ticks <= ticks + 1;

            end if;

        end if;
    end process;
```

```
begin

    Transitions: process (resetn, clk, s)
    begin
        if resetn = '0' then
            y <= S1;
        elsif (clk'event and clk = '1') then
            case y is
                when S1 =>
                    if sen = '1' then y <= S2; else y <= S1; end if;
                when S2 =>
                    if s = "0101" then y <= S3; else y <= S2; end if;
                when S3 =>
                    if s = "0111" then y <= S4; else y <= S3; end if;
                when S4 =>
                    if sen = '0' then y <= S5; else y <= S4; end if;
                when S5 =>
                    if s = "0101" then y <= S6; else y <= S5; end if;
                when S6 =>
                    if s = "0111" then y <= S1; else y <= S6; end if;
            --when others => s1;
            end case;
        end if;
    end process;
```

Timing Simulation



Schematics


clk
clk_IBUF_inst
IBUF
clk_IBUF_BUFG_inst
BUFG
cntr
CLK
Q
s_reg[0]_0
s_reg[2]_0
counter
resetn
resetn_IBUF_inst
IBUF
sen
sen_IBUF_inst
IBUF
fsm
CLK
FSM_sequential_y_reg[0]_0
Q
resetn_IBUF
sen_IBUF
FSM_sequential_y_reg[2]_0
sig_OBUF[5:0]
TLC_fsm
sig_OBUF[0]_inst
OBUF
sig_OBUF[1]_inst
OBUF
sig_OBUF[2]_inst
OBUF
sig_OBUF[3]_inst
OBUF
sig_OBUF[4]_inst
OBUF
sig_OBUF[5]_inst
OBUF
sig[5:0]

Issues

- Timing the lights first time we coded it the lights were going through the Red,Yellow, Green cycle way to quick.

- Using the pmod ports and wiring them to the leds

Possible Improvement

- One of the improvements we could of made was to add an actual sensor to see when traffic is coming instead of having to do it manually.
- Emergency detection system for when emergency vehicles are approaching
- To make all the lights go to blinking red if there was like a power outage or a system issue

Demo

Questions?