EEE 102
Introduction to Digital Circuit Design

Final Term Project Report

Flight controller for a drone

By

Payam Sedighiani

Instructor: Ergin Atalar

https://youtu.be/-Ren-DlYsoM

Payam Sedighiani

Term project report

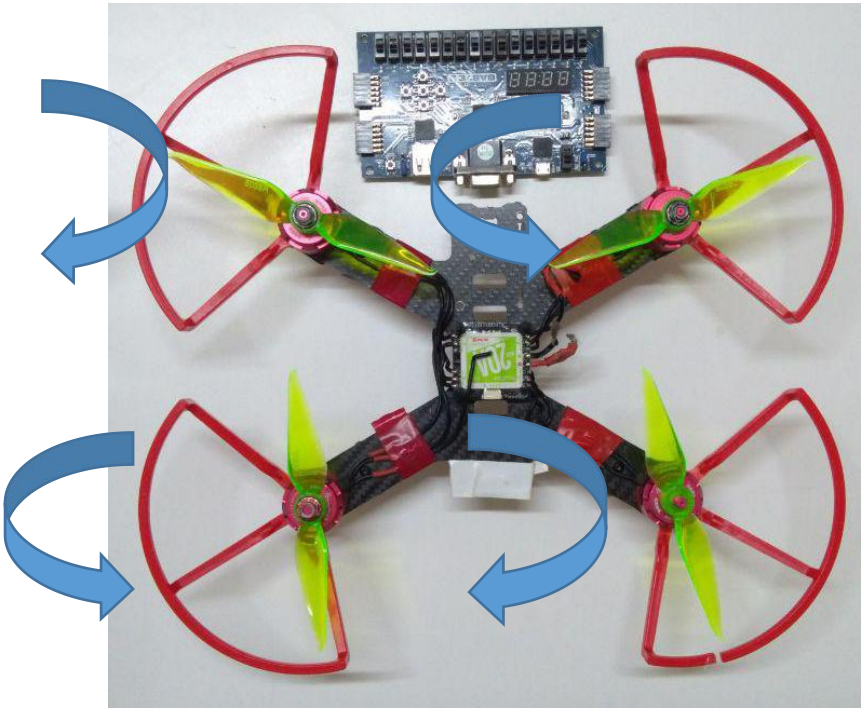
31st December 2019

Abstract/Objective

In the following project, a flight controller for a drone ,with BLDC motors, is designed and implemented, using FPGA board (BASYS 3) and VHDL language. The flight controller system is responsible for the directions of the drone's movement. In this design, there are left,right, frontward, backward and up directions are designed, using a FSM(finite state machine).

Design Specification Plan

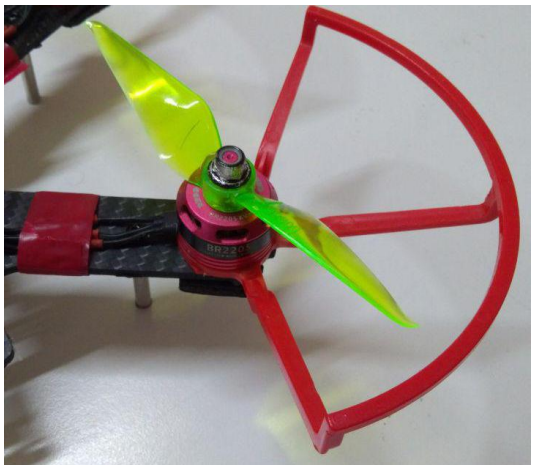
To test the flight controller, a drone is needed to be made with 4 BLDC motors, speed controller the motors direction of rotating must be designed in a specific way to lift the drone up. In this design the directions are set as in the following picture with the design of the drone and components are explained.:



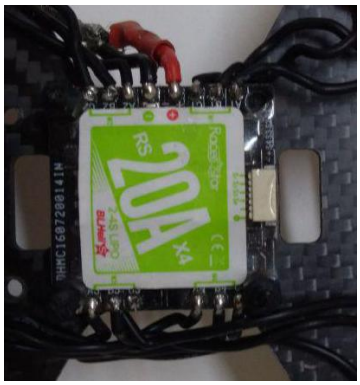
And to do this you can easily change the wires which goes to the ESC as shown in the following picture:

The main components:

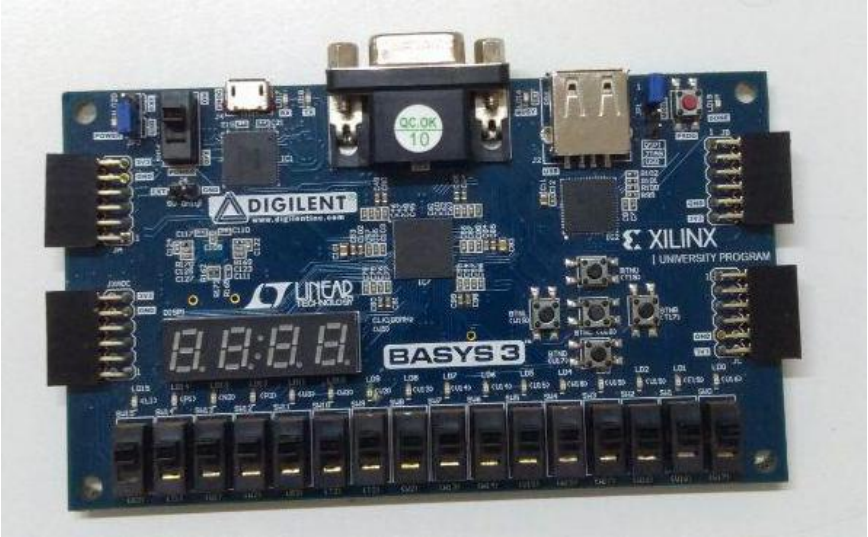
1. BLDC motor



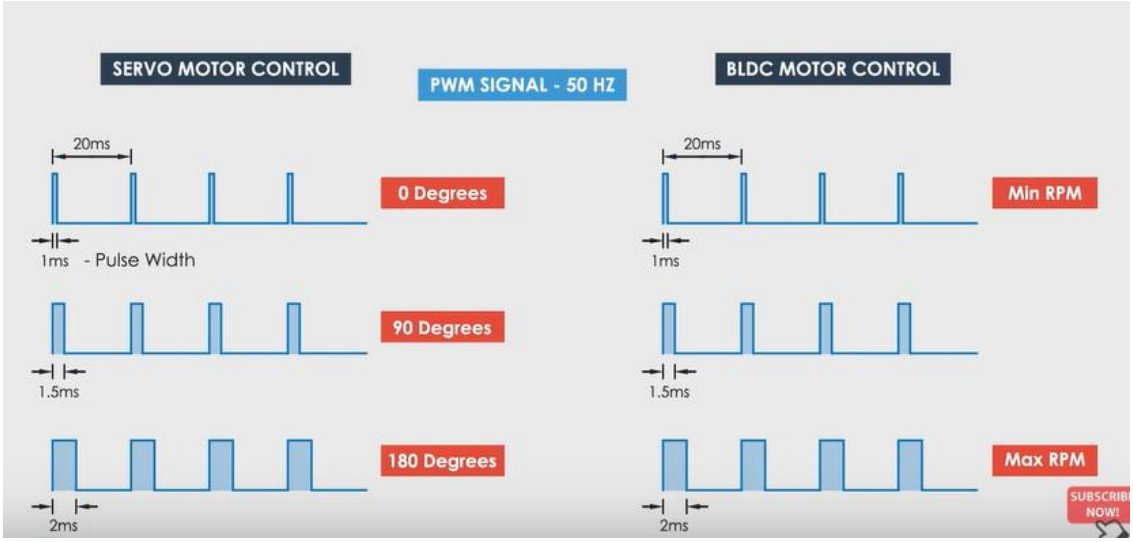
2. ESC(electronic speed controller)



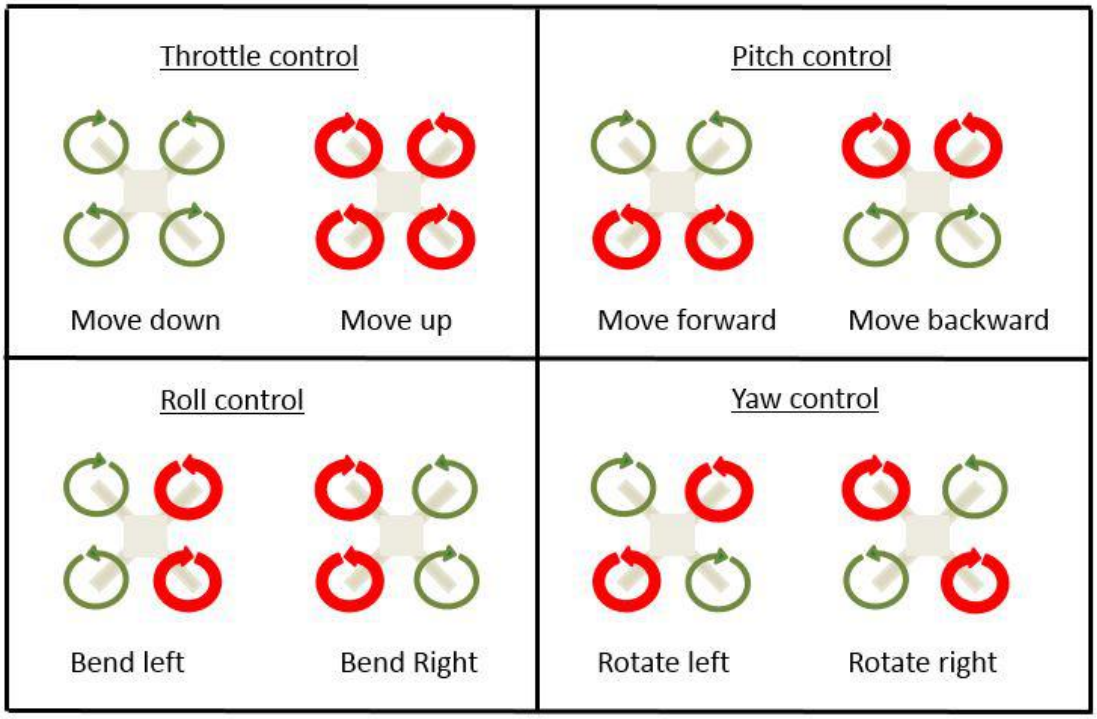
3. Basys 3 (FPGA board)



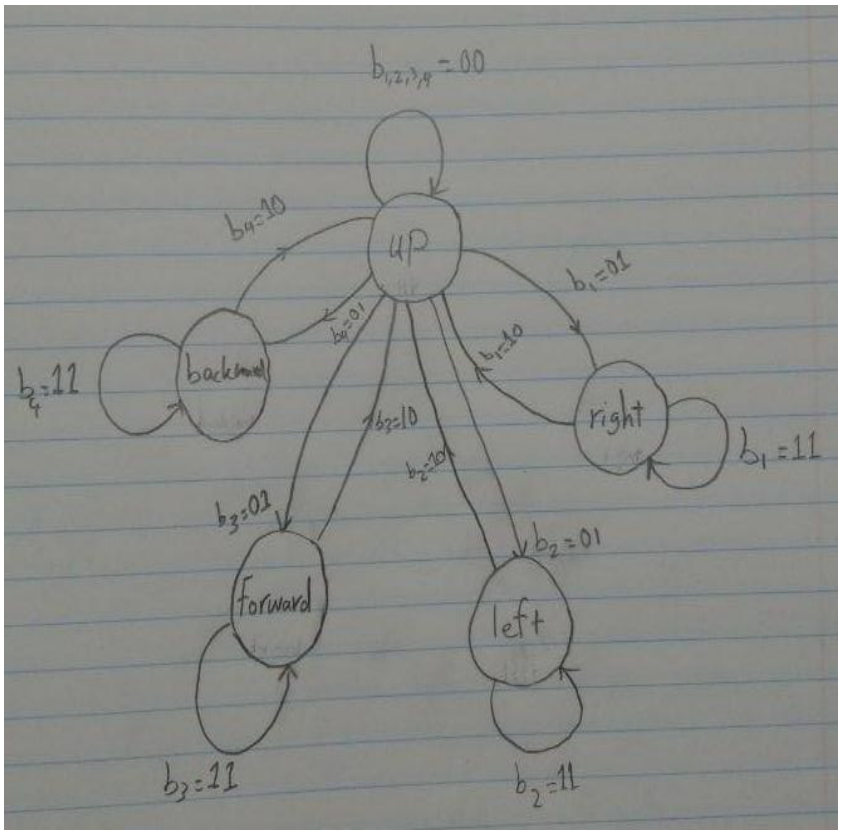
To control the speed of BLDC motors, PWM signals must be send to the ESC to initiate the motors. To initiate these type of motors, first their maximum and minimum must be set same as servo motors but this time instead of degrees, RPM is getting affected by changing the signal. For the minimum speed (the motor in this state wont move)the PWM signal must be high for 1 ms and 19 ms low(total 20 ms duty cycle because the frequency of the signal is 50 Hz) and for the maximum speed the signal must be 2 ms high and 18 ms low as shown in the following picture:



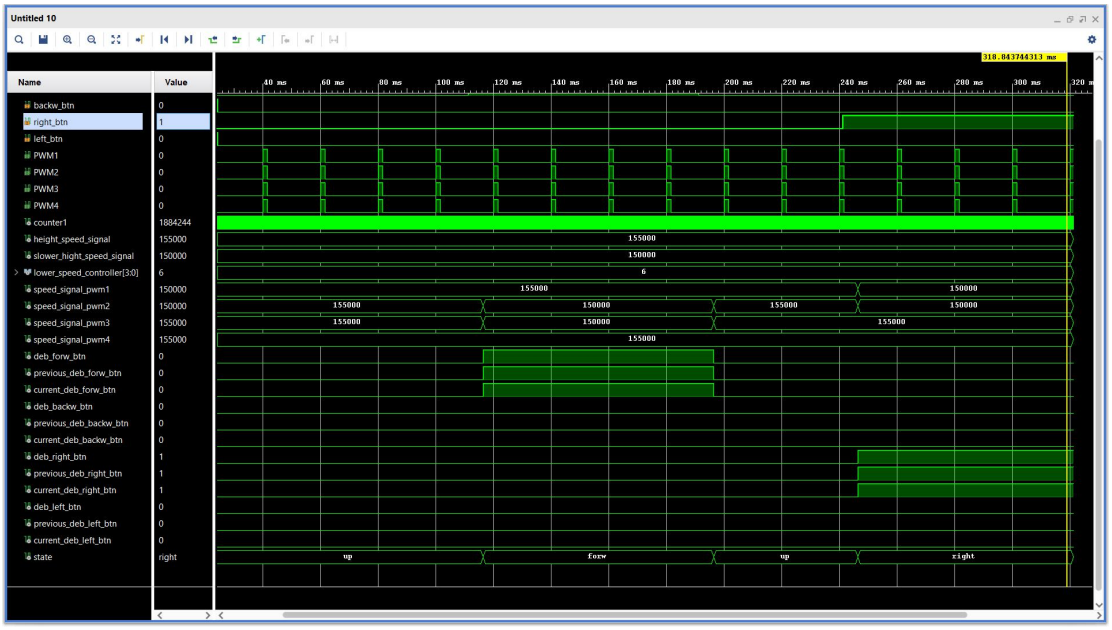
Using multiplexers in the code, different speeds are set by making the PWM signal high for numbers in between 1ms to 2ms(1.5 ms). so as we increase the speed the drone lifts up, therefore, we can adjust the height of the drone by setting up the switches on the Basys3 board, and since we have 16 to one multiplexer, four switches will be used to make a four digit binary number. Next stage is setting the directions of the drone. The logic behind the the directions is that for going forward the two front motors should decrease their speed with same amount and the two backward motors should keep going with their previous speed, similarly, we can define other directions for the drone. The following pictures are the logic behind all the directions.



To do this in FPGA, a Finite State Machine is designed to take care of all the directions. After the change in the speed of the motors, the drone should come back to its previous state. Similarly, the FSM should work for the other directions and the FSM will have 5 states: up, right, left, forward and backward . The following picture is the state diagram:

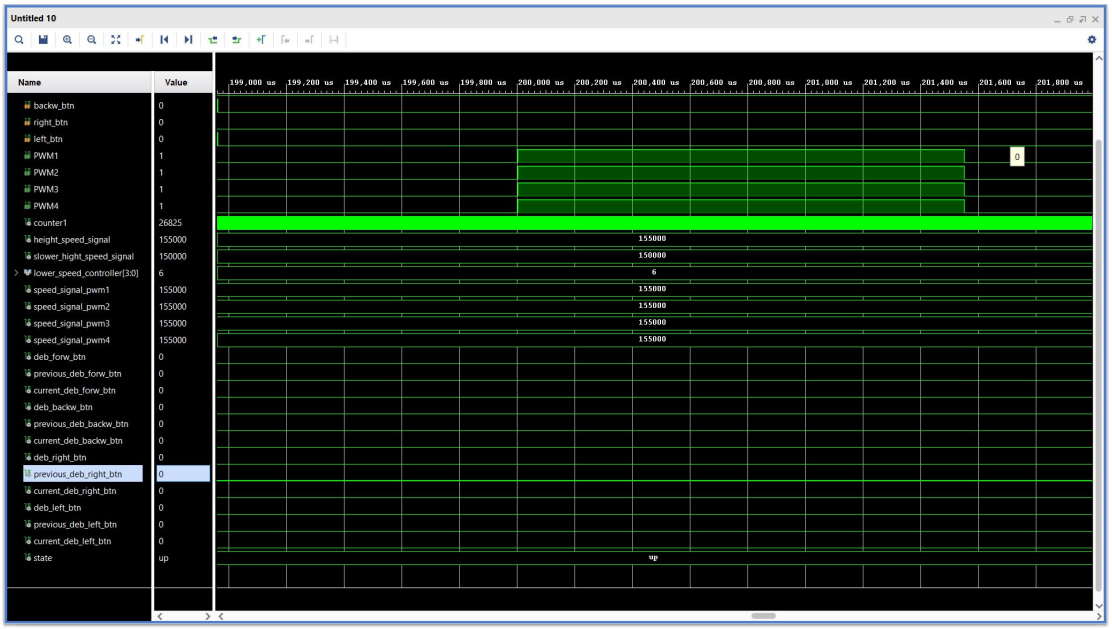


As shown in the above picture, when we leave the push button related to the direction, the drone goes back to its previous state which is up state. The FSM also has a reset button, p0, which also turns off the drone and this means every time that we turn off the drone, its first state will be zero and then the machine keeps storing the values. The timing diagram is the following pictures and we can see that the drone after leaving the button always goes back to its previous state which is up state and this is for the drones balance.



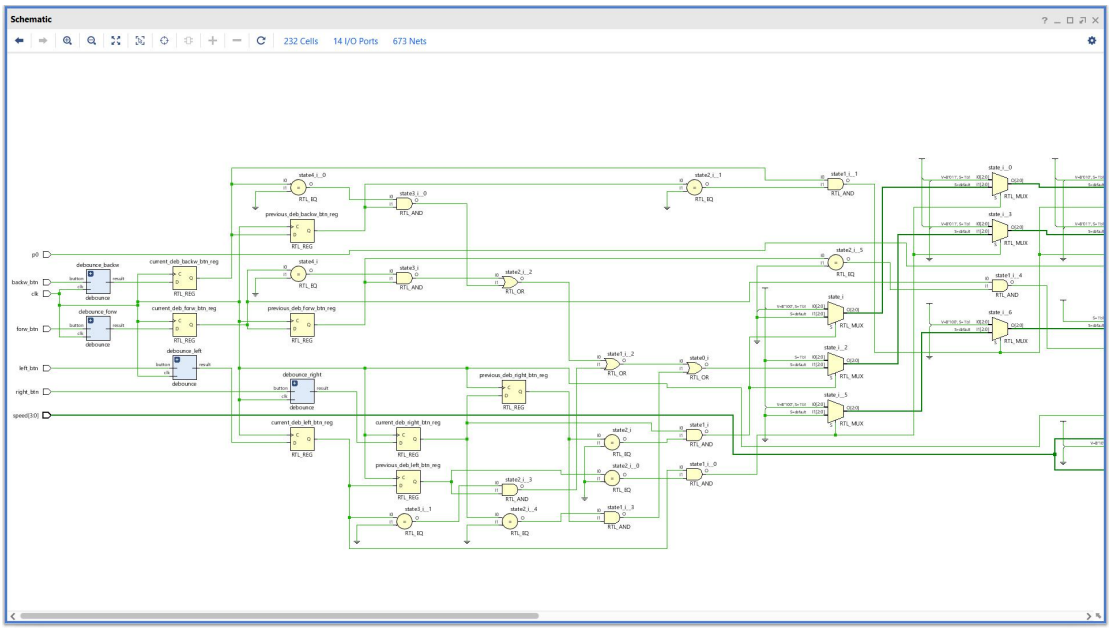
In the next picture, we can see that the speed of the two of the motors are decreased and the state is right. Note that in the code we can change how much the speed should be decreased.

And later after leaving the button, we can see that the state went back to up state as shown in the following picture:

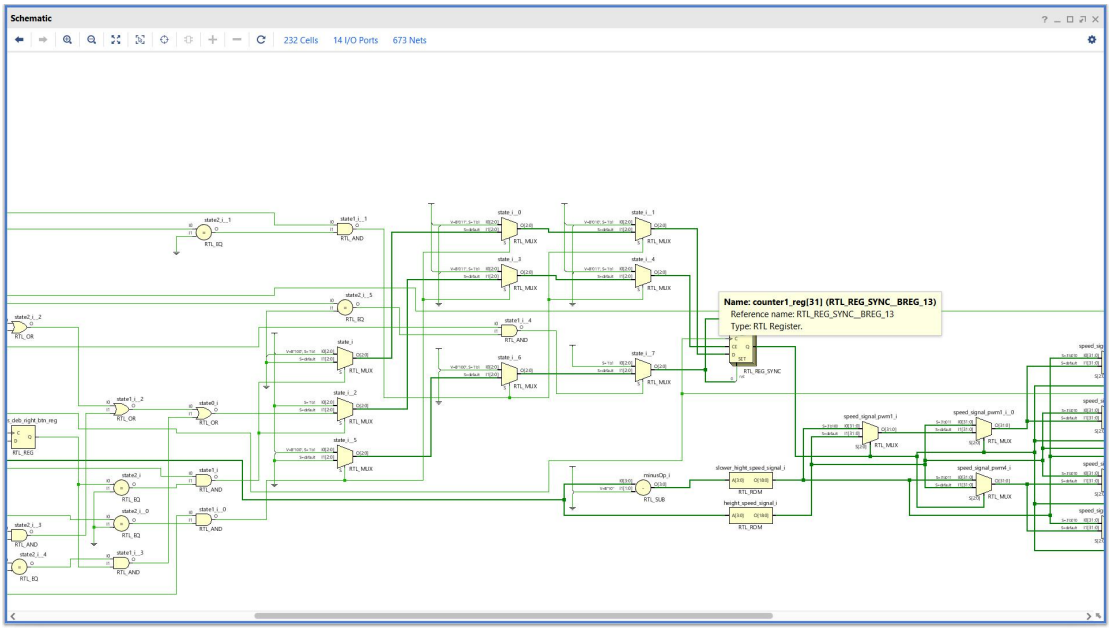


And here is the schematic of the complete logic designed(for better quality its in three pictures):

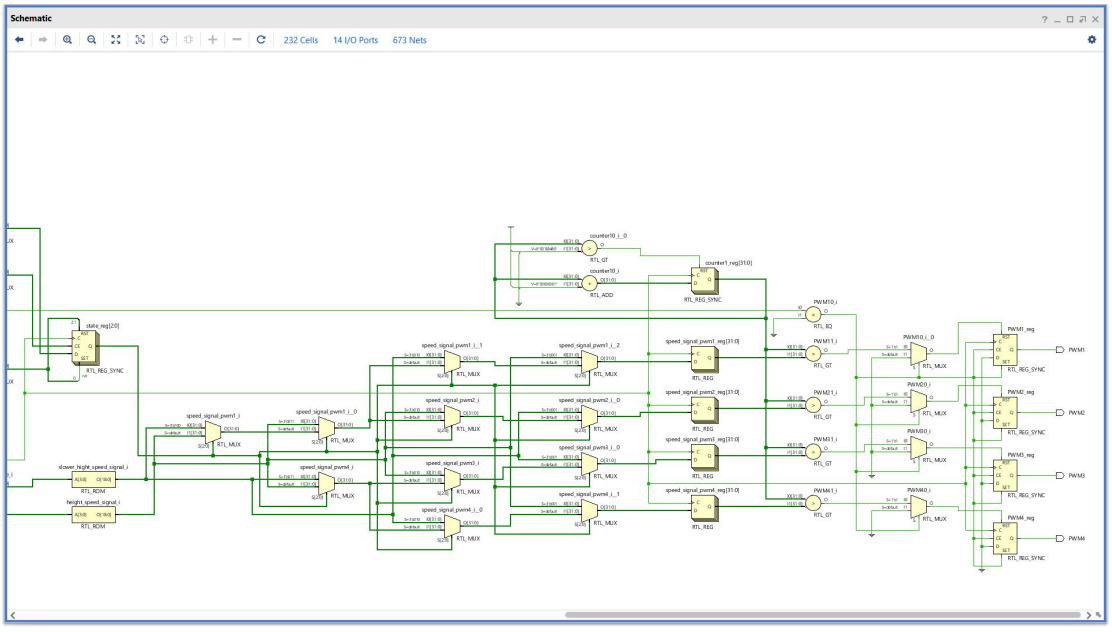
1)



2)


Schematic
232 Cells
14 I/O Ports
673 Nets
Name: counter1_reg[31] (RTL_REG_SYNC__BREG_13)
Reference name: RTL_REG_SYNC__BREG_13
Type: RTL Register.

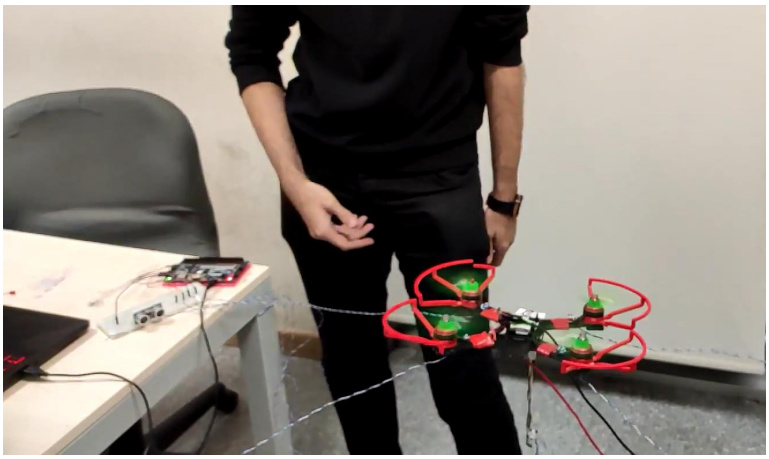
3)

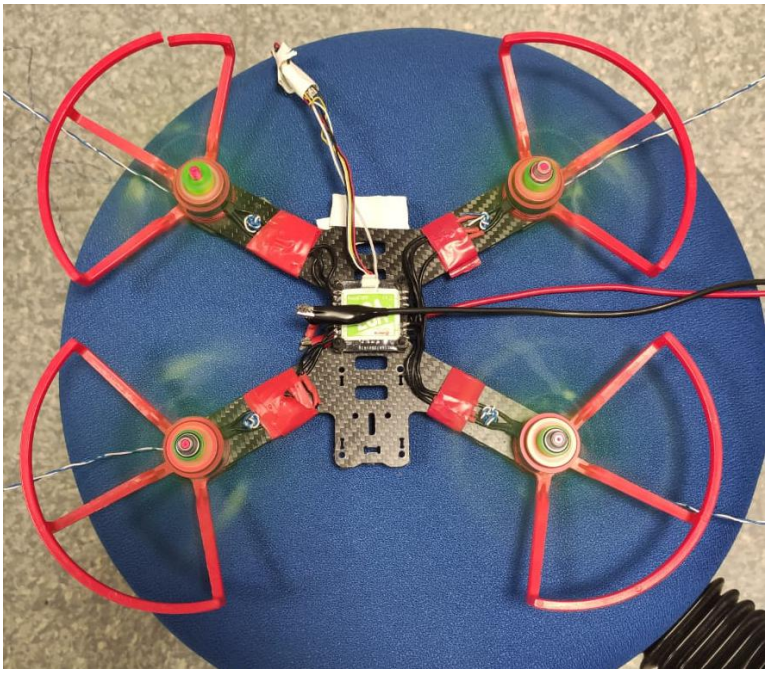

Schematic
232 Cells
14 I/O Ports
673 Nets

Results

The followings are the results after making the drone and coding it with VIVADO in VHDL:

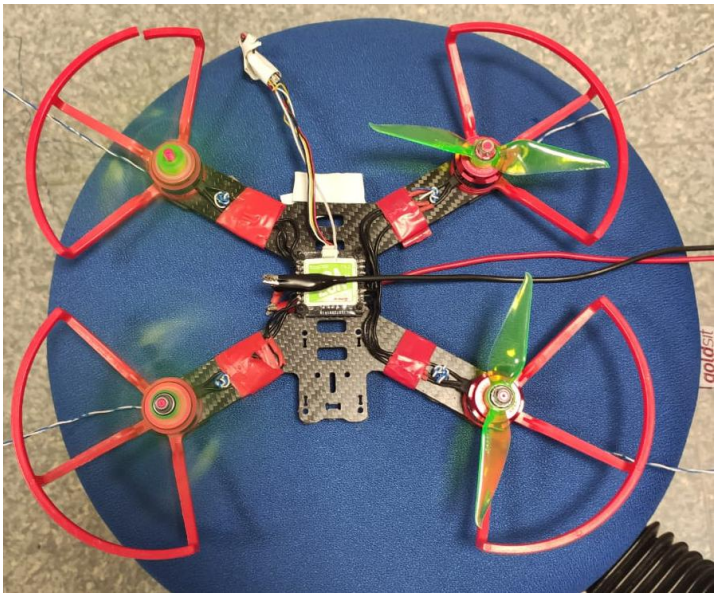
1) when the drone is at up state



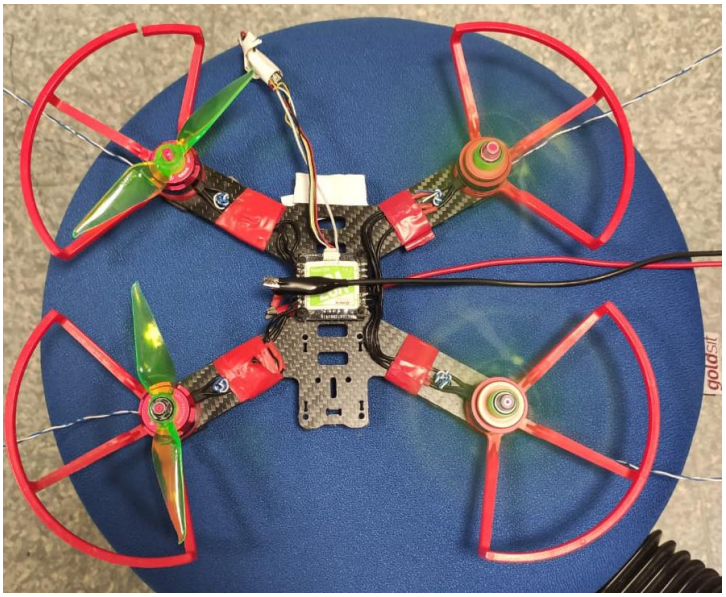


Note: the results are at the lowest speed to be shown in a better way, so when 1 speed the other two motors speed decrease the other two motors stops working because its in the lowest speed for education purposes.

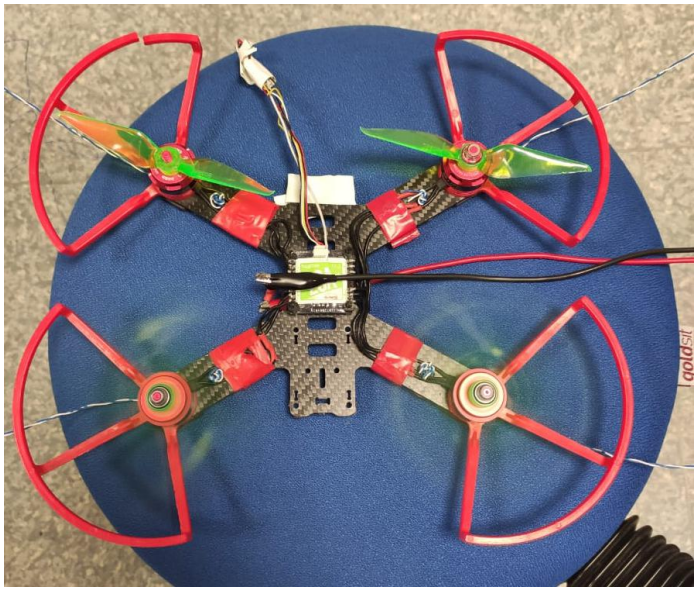
2) When the drone is at right state



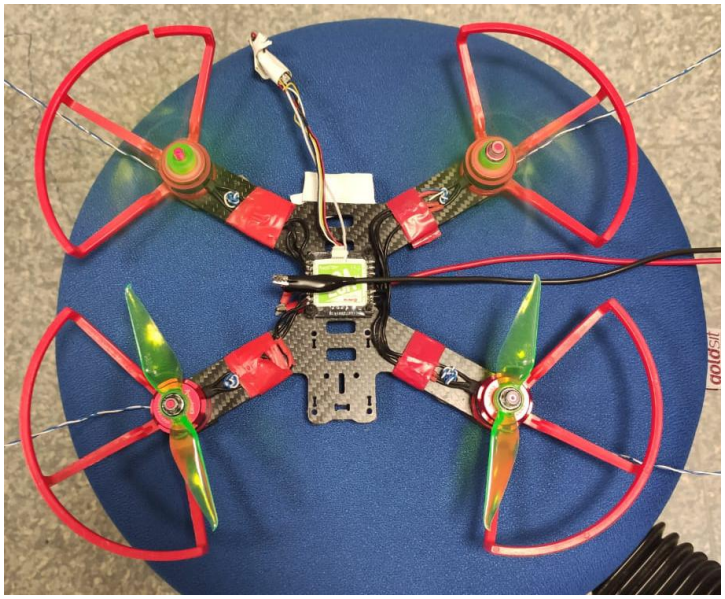
3) When the drone is at left state



4) When the drone is at forward state



5) When the drone is at backward state



conclusion

The drone worked properly as shown in the youtube video, however, there were some balancing issue in the project, which easily can be fixed by designing the drone more dynamic.

https://youtu.be/-Ren-DlYsoM

Appendixes

TOP MODULE:

```
library IEEE;
use IEEE.std_logic_1164.all;
--use IEEE.numeric_std.all;
use IEEE.std_logic_unsigned.all;

entity top is
    Port (
        clk         : in STD_LOGIC;
        -- speed switches
        speed      : in STD_LOGIC_VECTOR(3 downto 0); -- 00 => min
        p0          : in STD_LOGIC;                    --power
        forw_btn : in STD_LOGIC;                    --move to front
        backw_btn : in STD_LOGIC;                   --move to back
        right_btn : in STD_LOGIC;                   --move to right
        left_btn : in STD_LOGIC;                    --move to left

        -- pwm 1
        PWM1 : out STD_LOGIC;
        -- pwm 2
        PWM2 : out STD_LOGIC;
        -- pwm 3
        PWM3 : out STD_LOGIC;
        -- pwm 4
        PWM4 : out STD_LOGIC);
```

```
end top;

architecture Behavioral of top is
signal counter1: integer := 0;

signal height_speed_signal : integer := 100_000;
signal slower_hight_speed_signal : integer := 100_000;

-- this signal makes speed -1 for slower speed to be assigned to slower_hight_speed_signal
signal lower_speed_controller : STD_LOGIC_VECTOR(3 downto 0);

signal speed_signal_pwm1 : integer := 100_000;
signal speed_signal_pwm2 : integer := 100_000;
signal speed_signal_pwm3 : integer := 100_000;
signal speed_signal_pwm4 : integer := 100_000;



component debounce IS
   GENERIC(
      counter_size    :    INTEGER); --counter size (19 bits gives 10.5ms with 50MHz clock)
   PORT(
      clk         : IN    STD_LOGIC;    --input clock
      button    : IN    STD_LOGIC;    --input signal to be debounced
      result    : OUT STD_LOGIC); --debounced signal
END component;

-- Debounce signals
signal deb_forw_btn, previous_deb_forw_btn, current_deb_forw_btn : std_logic;
signal deb_backw_btn, previous_deb_backw_btn, current_deb_backw_btn : std_logic;
signal deb_right_btn, previous_deb_right_btn, current_deb_right_btn : std_logic;
```

```
signal deb_left_btn, previous_deb_left_btn, current_deb_left_btn : std_logic;

TYPE movement_state IS
	(up, forw, backw, left, right);

signal state : movement_state := up;

begin

lower_speed_controller <= speed - "0010";

	debounce_forw : debounce generic map
	(
		counter_size	=> 19
	)
	PORT map
	(
	clk => clk,
	button => forw_btn,
	result => deb_forw_btn
	);

	debounce_backw : debounce generic map
	(
		counter_size	=> 19
	)
	PORT map
	(
	clk => clk,
	button => backw_btn,
	result => deb_backw_btn
```

```
	);

	debounce_right : debounce generic map
	(
		counter_size	=> 19
	)
	PORT map
	(
	clk => clk,
	button => right_btn,
	result => deb_right_btn
	);

	debounce_left : debounce generic map
	(
		counter_size	=> 19
	)
	PORT map
	(
	clk => clk,
	button => left_btn,
	result => deb_left_btn
	);


-- assignment of button states
process(clk)
begin
	if rising_edge(clk) then
		previous_deb_forw_btn <= current_deb_forw_btn;
		current_deb_forw_btn <=	deb_forw_btn;
```

```
            previous_deb_backw_btn <= current_deb_backw_btn;
            current_deb_backw_btn <=    deb_backw_btn;

            previous_deb_left_btn <= current_deb_left_btn;
            current_deb_left_btn <=    deb_left_btn;

            previous_deb_right_btn <= current_deb_right_btn;
            current_deb_right_btn <=    deb_right_btn;
        end if;
end process;

-- motors speeds states
process(clk)
begin
        if rising_edge(clk) then
            if current_deb_forw_btn = '1' and previous_deb_forw_btn = '0' then
                state <= forw;

            elsif current_deb_backw_btn = '1' and previous_deb_backw_btn = '0' then
                state <= backw;

            elsif current_deb_left_btn = '1' and previous_deb_left_btn = '0' then
                state <= left;

            elsif current_deb_right_btn = '1' and previous_deb_right_btn = '0' then
                state <= right;

            elsif (current_deb_forw_btn = '0' and previous_deb_forw_btn = '1')
                or (current_deb_backw_btn = '0' and previous_deb_backw_btn = '1')
                or (current_deb_left_btn = '0' and previous_deb_left_btn = '1')
```

```
                or (current_deb_right_btn = '0' and previous_deb_right_btn = '1') then
                state <= up;
            end if;
        end if;
end process;

-- speed selection based specific button press
process(clk)
begin
        if rising_edge(clk) then
            if state = forw then
                -- high speed ones
                speed_signal_pwm1 <= height_speed_signal;
                speed_signal_pwm4 <= height_speed_signal;

                -- low speed ones
                speed_signal_pwm2 <= slower_hight_speed_signal;
                speed_signal_pwm3 <= slower_hight_speed_signal;

            elsif state = backw then
                -- high speed ones
                speed_signal_pwm2 <= height_speed_signal;
                speed_signal_pwm3 <= height_speed_signal;

                -- low speed ones
                speed_signal_pwm1 <= slower_hight_speed_signal;
                speed_signal_pwm4 <= slower_hight_speed_signal;

            elsif state = left then
                -- high speed ones
                speed_signal_pwm1 <= height_speed_signal;
```

```
                speed_signal_pwm2 <= height_speed_signal;

                -- low speed ones
                speed_signal_pwm4 <= slower_hight_speed_signal;
                speed_signal_pwm3 <= slower_hight_speed_signal;

            elsif state = right then
                -- high speed ones
                speed_signal_pwm3 <= height_speed_signal;
                speed_signal_pwm4 <= height_speed_signal;

                -- low speed ones
                speed_signal_pwm1 <= slower_hight_speed_signal;
                speed_signal_pwm2 <= slower_hight_speed_signal;
            else
                speed_signal_pwm1 <= height_speed_signal;
                speed_signal_pwm2 <= height_speed_signal;
                speed_signal_pwm3 <= height_speed_signal;
                speed_signal_pwm4 <= height_speed_signal;
            end if;

        end if;

end process;

with speed select height_speed_signal <=
        100_000 when "0000",
        110_000 when "0001",
        120_000 when "0010",
        130_000 when "0011",
        140_000 when "0100",
```

```
        145_000 when "0101",
        150_000 when "0110",
        155_000 when "0111",
        160_000 when "1000",
        165_000 when "1001",
        170_000 when "1010",
        175_000 when "1011",
        180_000 when "1100",
        200_000 when others;



with lower_speed_controller select slower_hight_speed_signal <=
        100_000 when "0000",
        110_000 when "0001",
        120_000 when "0010",
        130_000 when "0011",
        140_000 when "0100",
        145_000 when "0101",
        150_000 when "0110",
        155_000 when "0111",
        160_000 when "1000",
        165_000 when "1001",
        170_000 when "1010",
        175_000 when "1011",
        180_000 when "1100",
        200_000 when others;



-- setting up the counter
```

```
process(clk)
begin
	if rising_edge(clk) then
		counter1 <= counter1 + 1;
		if counter1 > 2_000_000 then
			counter1 <= 0;
		end if;
	  end if;
end process;



-- Motors
process(clk)
begin
if rising_edge(clk) and p0 = '0' then

	PWM1 <= '1';
	PWM2 <= '1';
	PWM3 <= '1';
	PWM4 <= '1';

	if counter1 > speed_signal_pwm1 then
		PWM1 <= '0';
	end if;

	if counter1 > speed_signal_pwm2 then
		PWM2 <= '0';
	end if;

	if counter1 > speed_signal_pwm3 then
		PWM3 <= '0';
```

```
        end if;

        if counter1 > speed_signal_pwm4 then
            PWM4 <= '0';
        end if;

end if;
end process;

-- speed control
process(clk)
begin
        if rising_edge(clk) then

        end if;

end process;

end Behavioral;
```

CONSTRAINTS:

```
## This file is a general .xdc for the Basys3 rev B board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports) according to the top level signal names in the project
```

```
## Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
	set_property IOSTANDARD LVCMOS33 [get_ports clk]
	#create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

## Switches
set_property PACKAGE_PIN V17 [get_ports {speed[0]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {speed[0]}]
set_property PACKAGE_PIN V16 [get_ports {speed[1]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {speed[1]}]
set_property PACKAGE_PIN W16 [get_ports {speed[2]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {speed[2]}]
set_property PACKAGE_PIN W17 [get_ports {speed[3]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {speed[3]}]
#set_property PACKAGE_PIN W15 [get_ports {speed[4]}]
#	set_property IOSTANDARD LVCMOS33 [get_ports {speed[4]}]
#set_property PACKAGE_PIN V15 [get_ports {sw[5]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[5]}]
#set_property PACKAGE_PIN W14 [get_ports {sw[6]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[6]}]
#set_property PACKAGE_PIN W13 [get_ports {sw[7]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[7]}]
#set_property PACKAGE_PIN V2 [get_ports {sw[8]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[8]}]
#set_property PACKAGE_PIN T3 [get_ports {sw[9]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[9]}]
#set_property PACKAGE_PIN T2 [get_ports {sw[10]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[10]}]
#set_property PACKAGE_PIN R3 [get_ports {sw[11]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[11]}]
#set_property PACKAGE_PIN W2 [get_ports {sw[12]}]
```

```
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[12]}]
#set_property PACKAGE_PIN U1 [get_ports {sw[13]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[13]}]
#set_property PACKAGE_PIN T1 [get_ports {forward}]
#   set_property IOSTANDARD LVCMOS33 [get_ports {forward}]
set_property PACKAGE_PIN R2 [get_ports {p0}]
    set_property IOSTANDARD LVCMOS33 [get_ports {p0}]


## LEDs
#set_property PACKAGE_PIN U16 [get_ports {led[0]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]
#set_property PACKAGE_PIN E19 [get_ports {led[1]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
#set_property PACKAGE_PIN U19 [get_ports {led[2]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
#set_property PACKAGE_PIN V19 [get_ports {led[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
#set_property PACKAGE_PIN W18 [get_ports {led[4]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
#set_property PACKAGE_PIN U15 [get_ports {led[5]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[5]}]
#set_property PACKAGE_PIN U14 [get_ports {led[6]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[6]}]
#set_property PACKAGE_PIN V14 [get_ports {led[7]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[7]}]
#set_property PACKAGE_PIN V13 [get_ports {led[8]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[8]}]
#set_property PACKAGE_PIN V3 [get_ports {led[9]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[9]}]
#set_property PACKAGE_PIN W3 [get_ports {led[10]}]
```

```
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[10]}]
#set_property PACKAGE_PIN U3 [get_ports {led[11]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[11]}]
#set_property PACKAGE_PIN P3 [get_ports {led[12]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[12]}]
#set_property PACKAGE_PIN N3 [get_ports {led[13]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[13]}]
#set_property PACKAGE_PIN P1 [get_ports {led[14]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[14]}]
#set_property PACKAGE_PIN L1 [get_ports {led[15]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[15]}]


##7 segment display
#set_property PACKAGE_PIN W7 [get_ports {seg[0]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]}]
#set_property PACKAGE_PIN W6 [get_ports {seg[1]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]
#set_property PACKAGE_PIN U8 [get_ports {seg[2]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
#set_property PACKAGE_PIN V8 [get_ports {seg[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
#set_property PACKAGE_PIN U5 [get_ports {seg[4]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
#set_property PACKAGE_PIN V5 [get_ports {seg[5]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]}]
#set_property PACKAGE_PIN U7 [get_ports {seg[6]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]}]

#set_property PACKAGE_PIN V7 [get_ports dp]
    #set_property IOSTANDARD LVCMOS33 [get_ports dp]
```

```
#set_property PACKAGE_PIN U2 [get_ports {an[0]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {an[0]}]
#set_property PACKAGE_PIN U4 [get_ports {an[1]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {an[1]}]
#set_property PACKAGE_PIN V4 [get_ports {an[2]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {an[2]}]
#set_property PACKAGE_PIN W4 [get_ports {an[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {an[3]}]


##Buttons
#set_property PACKAGE_PIN U18 [get_ports btnC]
    #set_property IOSTANDARD LVCMOS33 [get_ports btnC]
set_property PACKAGE_PIN T18 [get_ports forw_btn]
    set_property IOSTANDARD LVCMOS33 [get_ports forw_btn]
set_property PACKAGE_PIN W19 [get_ports left_btn]
    set_property IOSTANDARD LVCMOS33 [get_ports left_btn]
set_property PACKAGE_PIN T17 [get_ports right_btn]
    set_property IOSTANDARD LVCMOS33 [get_ports right_btn]
set_property PACKAGE_PIN U17 [get_ports backw_btn]
    set_property IOSTANDARD LVCMOS33 [get_ports backw_btn]



##Pmod Header JA
##Sch name = JA1
#set_property PACKAGE_PIN J1 [get_ports {JA[0]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JA[0]}]
##Sch name = JA2
#set_property PACKAGE_PIN L2 [get_ports {JA[1]}]
```

```
	#set_property IOSTANDARD LVCMOS33 [get_ports {JA[1]}]
##Sch name = JA3
#set_property PACKAGE_PIN J2 [get_ports {JA[2]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JA[2]}]
##Sch name = JA4
#set_property PACKAGE_PIN G2 [get_ports {JA[3]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JA[3]}]
##Sch name = JA7
#set_property PACKAGE_PIN H1 [get_ports {JA[4]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JA[4]}]
##Sch name = JA8
#set_property PACKAGE_PIN K2 [get_ports {JA[5]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JA[5]}]
##Sch name = JA9
#set_property PACKAGE_PIN H2 [get_ports {JA[6]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JA[6]}]
##Sch name = JA10
#set_property PACKAGE_PIN G3 [get_ports {JA[7]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JA[7]}]



##Pmod Header JB
##Sch name = JB1
#set_property PACKAGE_PIN A14 [get_ports {PWM1}]
#	set_property IOSTANDARD LVCMOS33 [get_ports {PWM1}]
#Sch name = JB2
set_property PACKAGE_PIN A16 [get_ports {PWM2}]
	set_property IOSTANDARD LVCMOS33 [get_ports {PWM2}]
#Sch name = JB3
set_property PACKAGE_PIN B15 [get_ports {PWM3}]
```

```
	set_property IOSTANDARD LVCMOS33 [get_ports {PWM3}]
#Sch name = JB4
set_property PACKAGE_PIN B16 [get_ports {PWM4}]
	set_property IOSTANDARD LVCMOS33 [get_ports {PWM4}]
#Sch name = JB7
set_property PACKAGE_PIN A15 [get_ports {PWM1}]
	set_property IOSTANDARD LVCMOS33 [get_ports {PWM1}]
##Sch name = JB8
#set_property PACKAGE_PIN A17 [get_ports {JB[5]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JB[5]}]
##Sch name = JB9
#set_property PACKAGE_PIN C15 [get_ports {JB[6]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JB[6]}]
##Sch name = JB10
#set_property PACKAGE_PIN C16 [get_ports {JB[7]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JB[7]}]



##Pmod Header JC
##Sch name = JC1
#set_property PACKAGE_PIN K17 [get_ports {JC[0]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[0]}]
##Sch name = JC2
#set_property PACKAGE_PIN M18 [get_ports {JC[1]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[1]}]
##Sch name = JC3
#set_property PACKAGE_PIN N17 [get_ports {JC[2]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[2]}]
##Sch name = JC4
#set_property PACKAGE_PIN P18 [get_ports {JC[3]}]
```

```
    #set_property IOSTANDARD LVCMOS33 [get_ports {JC[3]}]
##Sch name = JC7
#set_property PACKAGE_PIN L17 [get_ports {JC[4]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JC[4]}]
##Sch name = JC8
#set_property PACKAGE_PIN M19 [get_ports {JC[5]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JC[5]}]
##Sch name = JC9
#set_property PACKAGE_PIN P17 [get_ports {JC[6]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JC[6]}]
##Sch name = JC10
#set_property PACKAGE_PIN R18 [get_ports {JC[7]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JC[7]}]


##Pmod Header JXADC
##Sch name = XA1_P
#set_property PACKAGE_PIN J3 [get_ports {JXADC[0]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[0]}]
##Sch name = XA2_P
#set_property PACKAGE_PIN L3 [get_ports {JXADC[1]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[1]}]
##Sch name = XA3_P
#set_property PACKAGE_PIN M2 [get_ports {JXADC[2]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[2]}]
##Sch name = XA4_P
#set_property PACKAGE_PIN N2 [get_ports {JXADC[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[3]}]
##Sch name = XA1_N
#set_property PACKAGE_PIN K3 [get_ports {JXADC[4]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[4]}]
```

```
##Sch name = XA2_N
#set_property PACKAGE_PIN M3 [get_ports {JXADC[5]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[5]}]
##Sch name = XA3_N
#set_property PACKAGE_PIN M1 [get_ports {JXADC[6]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[6]}]
##Sch name = XA4_N
#set_property PACKAGE_PIN N1 [get_ports {JXADC[7]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[7]}]



##VGA Connector
#set_property PACKAGE_PIN G19 [get_ports {vgaRed[0]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[0]}]
#set_property PACKAGE_PIN H19 [get_ports {vgaRed[1]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[1]}]
#set_property PACKAGE_PIN J19 [get_ports {vgaRed[2]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[2]}]
#set_property PACKAGE_PIN N19 [get_ports {vgaRed[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[3]}]
#set_property PACKAGE_PIN N18 [get_ports {vgaBlue[0]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[0]}]
#set_property PACKAGE_PIN L18 [get_ports {vgaBlue[1]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[1]}]
#set_property PACKAGE_PIN K18 [get_ports {vgaBlue[2]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[2]}]
#set_property PACKAGE_PIN J18 [get_ports {vgaBlue[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[3]}]
#set_property PACKAGE_PIN J17 [get_ports {vgaGreen[0]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[0]}]
```

#set_property PACKAGE_PIN H17 [get_ports {vgaGreen[1]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[1]}]

#set_property PACKAGE_PIN G17 [get_ports {vgaGreen[2]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[2]}]

#set_property PACKAGE_PIN D17 [get_ports {vgaGreen[3]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[3]}]

#set_property PACKAGE_PIN P19 [get_ports Hsync]

#set_property IOSTANDARD LVCMOS33 [get_ports Hsync]

#set_property PACKAGE_PIN R19 [get_ports Vsync]

#set_property IOSTANDARD LVCMOS33 [get_ports Vsync]

##USB-RS232 Interface

#set_property PACKAGE_PIN B18 [get_ports RsRx]

#set_property IOSTANDARD LVCMOS33 [get_ports RsRx]

#set_property PACKAGE_PIN A18 [get_ports RsTx]

#set_property IOSTANDARD LVCMOS33 [get_ports RsTx]

##USB HID (PS/2)

#set_property PACKAGE_PIN C17 [get_ports PS2Clk]

#set_property IOSTANDARD LVCMOS33 [get_ports PS2Clk]

#set_property PULLUP true [get_ports PS2Clk]

#set_property PACKAGE_PIN B17 [get_ports PS2Data]

#set_property IOSTANDARD LVCMOS33 [get_ports PS2Data]

#set_property PULLUP true [get_ports PS2Data]

##Quad SPI Flash

##Note that CCLK_0 cannot be placed in 7 series devices. You can access it using the

##STARTUPE2 primitive.

```
#set_property PACKAGE_PIN D18 [get_ports {QspiDB[0]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[0]}]
#set_property PACKAGE_PIN D19 [get_ports {QspiDB[1]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[1]}]
#set_property PACKAGE_PIN G18 [get_ports {QspiDB[2]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[2]}]
#set_property PACKAGE_PIN F18 [get_ports {QspiDB[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[3]}]
#set_property PACKAGE_PIN K19 [get_ports QspiCSn]
    #set_property IOSTANDARD LVCMOS33 [get_ports QspiCSn]
```