Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 7
31.05.2021

Design Methodology:

In this lab work, we are needed to design circuit that calculates two 8-bit inputs greatest common divisor. I used Euclid's GCD algorithm to do this design. In this algorithm, I am subtracting lesser number from higher number until they are equal. For example:

-	GCD of 20 and 6.	(a = 20, b = 6)
a > b , then a - b	20 – 6 = 14	(a = 14, b = 6)
a > b , then a - b	14 – 6 = 8	(a = 8, b = 6)
b > a , then b - a	8 – 6 = 2	(a = 2, b = 6)
b > a , then b - a	6 – 2 = 4	(a = 2, b = 4)
b > a , then b - a	4 – 2 = 2 **a = b, GCD=2**	(a = 2, b = 2)

Table 1: Euclid's GCD algorithm

Then I wrote this algorithm in module GCD. I used process with asynchronous output: button. When button is pressed, it calculates again. To use button, I wrote finite state machine with 0.2 second time. If I press button 0.2 second, it changes state to ON.

Results:

After generating bitstream, codes are working as expected. There are the schematics and demo results of the lab work.

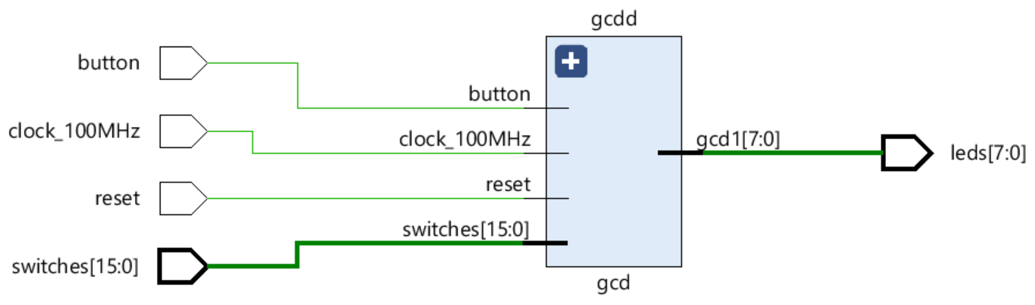

Figure 1: RTL Schematic

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 7
31.05.2021

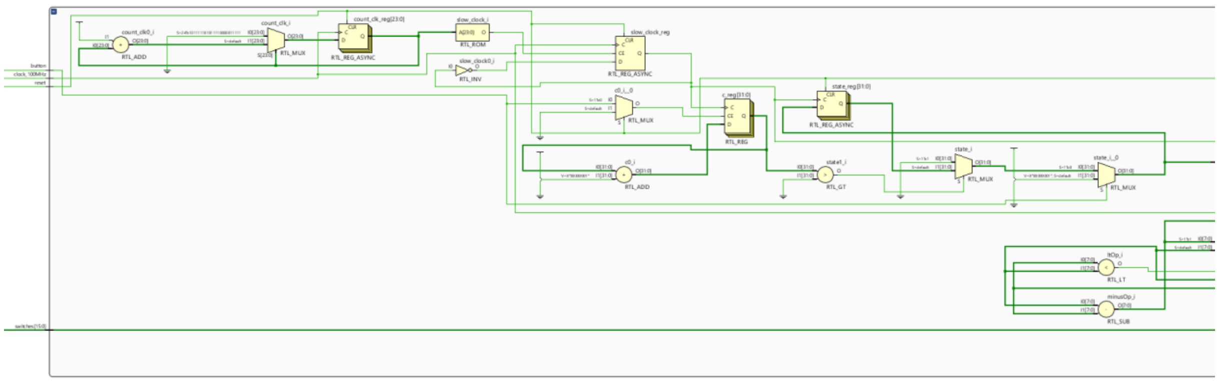

Figure 2: Gcd Schematic Part 1

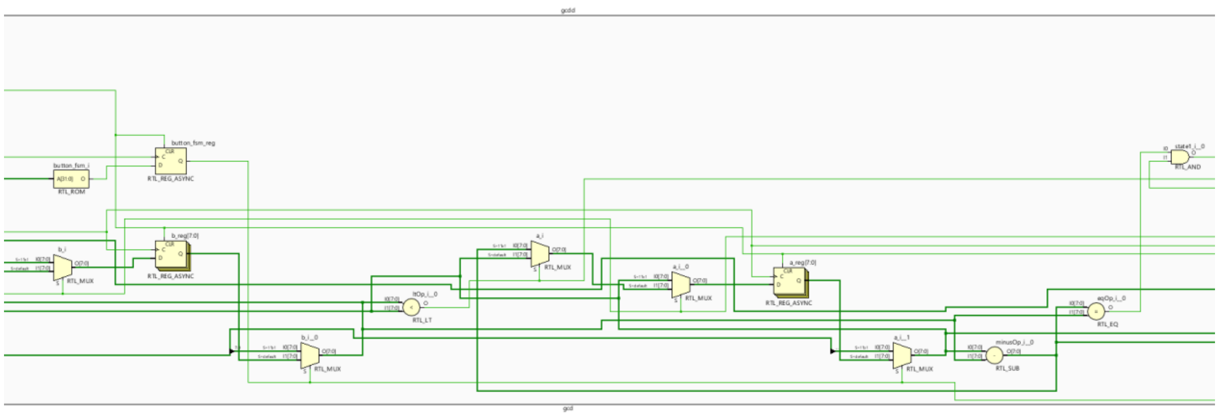

Figure 3: Gcd Schematic Part 2

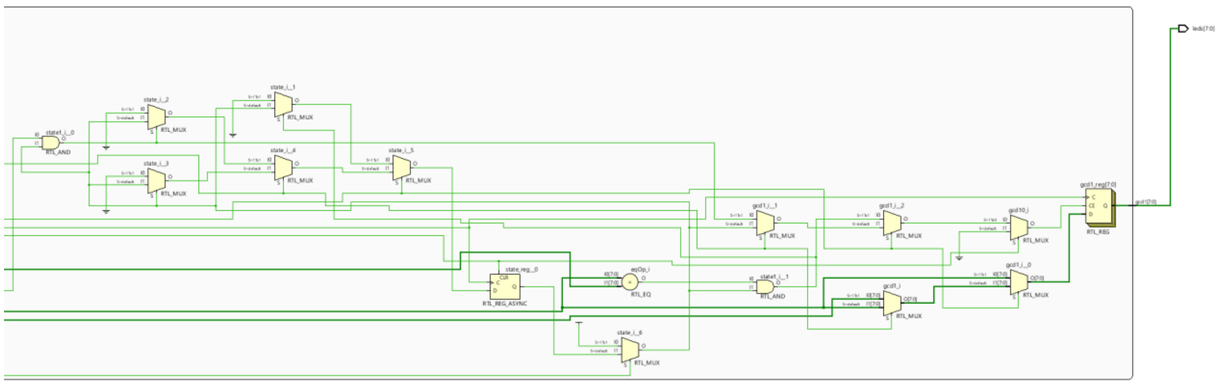

Figure 4: Gcd Schematic Part 3

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 7
31.05.2021

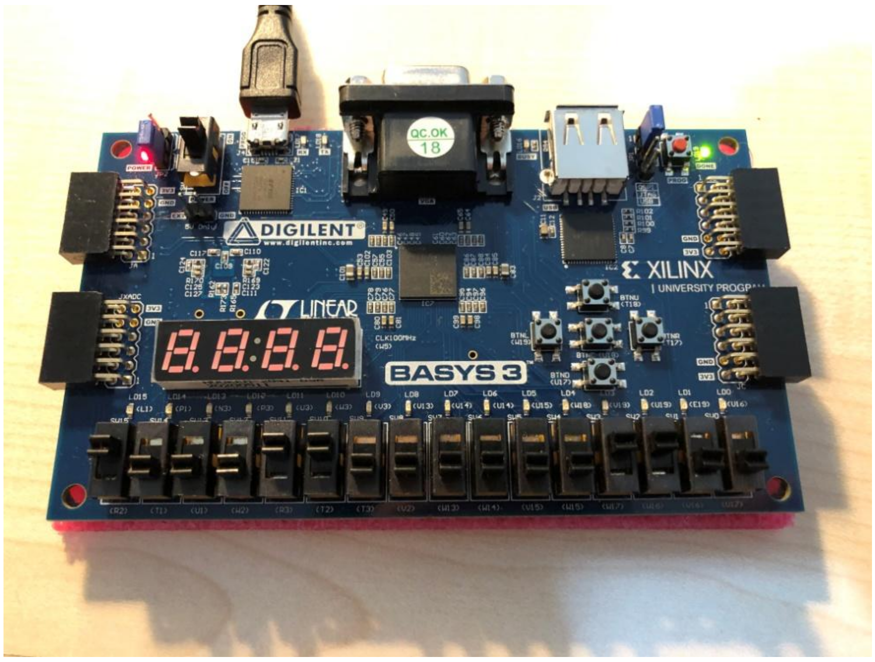

Figure 2: Demo 1 (140 – 12, no button signal)

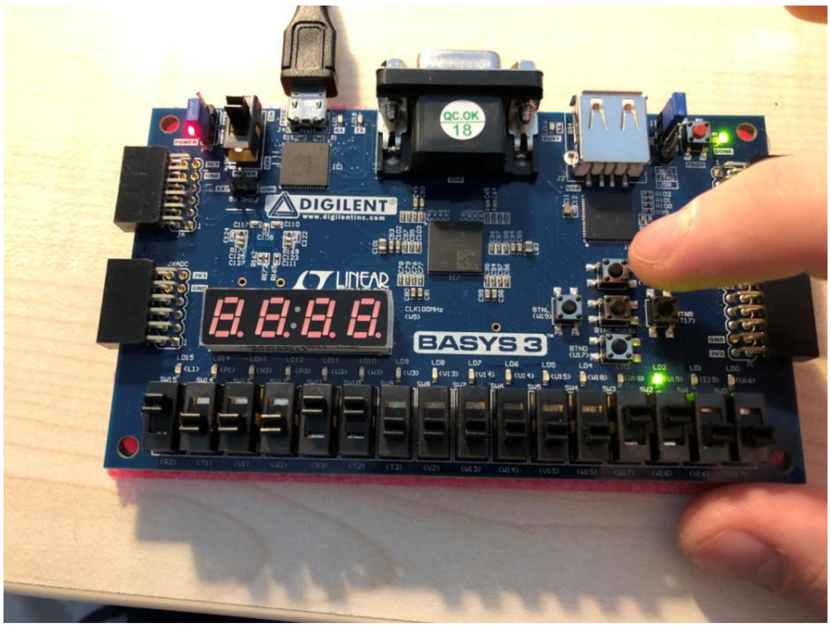

Figure 3: Demo 2 (140 – 12, button signal, output = 4)

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 7
31.05.2021

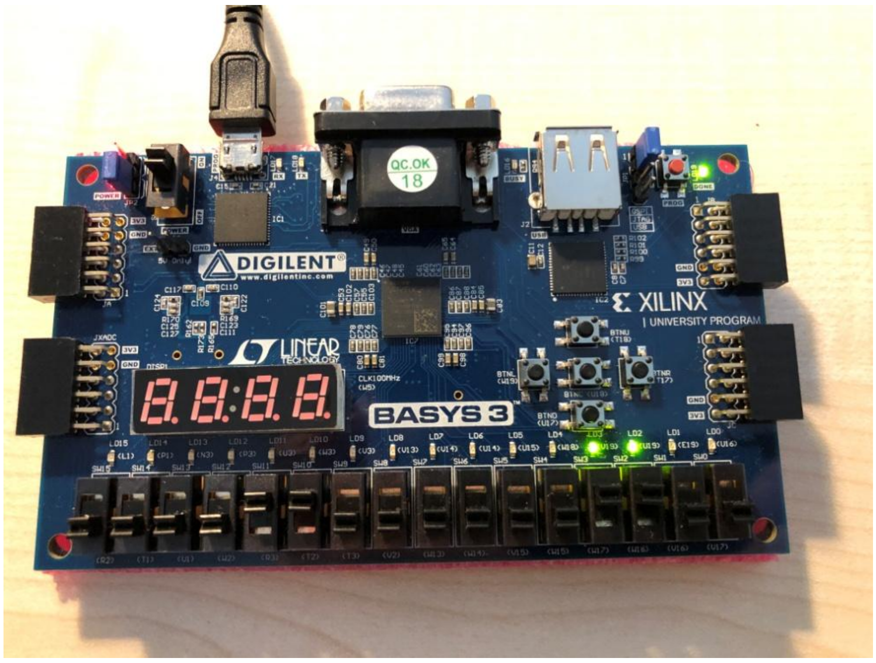

Figure 4: Demo 3 (12 – 12, button signal, output = 12)

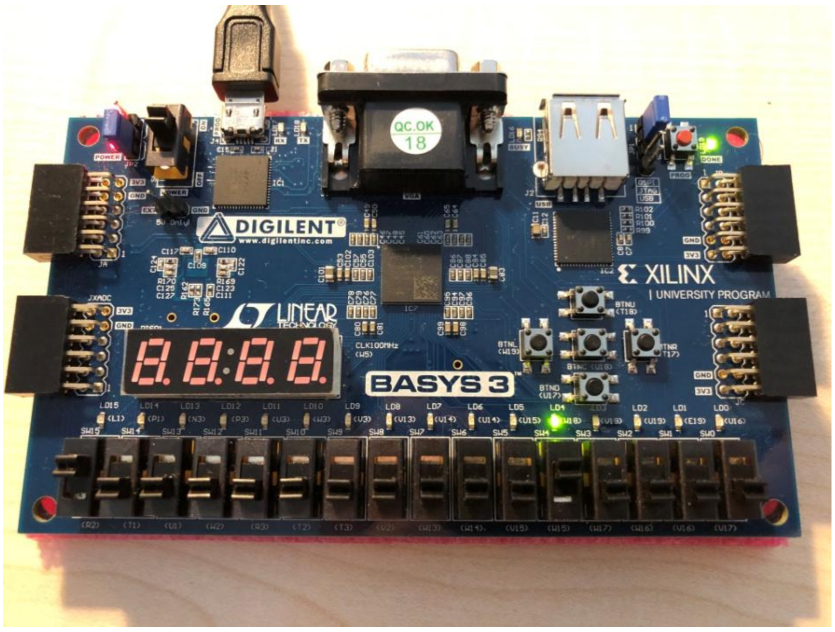

Figure 5: Demo 4 (128 – 16, button signal, output = 16)

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 7
31.05.2021

Conclusion:

In this lab work, I learned how to calculate mathematical things with using special algorithms. It is hard to calculate integer division and modulus from divisions in VHDL but still we can subtract one from another. It is simple integer division actually. Writing testbench was challenging and I took U values in first tries. It takes more time than writing GCD code. Since 100 MHz is very high speed clock which Basys3 has, calculations are very quick.

Questions:

1) What was your algorithm to calculate the GCD?
   I used Euclid's algorithm. It is explained in design methodology part.

2) Is your module a combinational circuit or an FSM?
   My design is combinational circuit. I did not use any states while writing GCD process.

3) Would it be cheaper to implement GCD as a combinational circuit or as an FSM? Would it be faster? What are the drawbacks in each case?
   I think implementing as combinational circuit is faster because it is not using states. I think combinational circuit is faster than FSM.

4) How many clock cycles did it take in your simulation to calculate the GCD? Do you think this can be optimized? How so?
   My design takes 13 clock cycles after pressing button to calculate GCD of 140 and 12. It changes due to input numbers. I do not think it can be optimized because of Euclid's algorithm. If there are operators like integer division "//" and modulus "%" in VHDL like Python, it can be shorter.

Appendices:

Top module:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

entity main is
```

```
 Port (

    clock_100MHz: in std_logic;

    reset: in std_logic;

    switches: in std_logic_vector(15 downto 0);

    button: in std_logic;

    leds: out std_logic_vector(7 downto 0)

    );

end main;


architecture Behavioral of main is


component gcd is

  Port (

    clock_100MHz: in std_logic;

    reset: in std_logic;

    switches: in std_logic_vector(15 downto 0);

    gcd1: out std_logic_vector(7 downto 0);

    button: in std_logic

    );

end component;

signal gcd1: std_logic_vector(7 downto 0);


begin

leds <= gcd1;
```

```
gcdd: gcd port map ( clock_100MHz => clock_100MHz, reset => reset, switches => switches,
gcd1 => gcd1, button => button);

end Behavioral;
```

GCD Module:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.std_logic_unsigned.all;


entity gcd is

 Port (

       clock_100MHz: in std_logic;

       reset: in std_logic;

       switches: in std_logic_vector(15 downto 0);

       gcd1: out std_logic_vector(7 downto 0);

       button: in std_logic


       );

end gcd;


architecture Behavioral of gcd is

signal slow_clock: std_logic;

signal button_fsm: std_logic;

begin
```

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 7
31.05.2021

```
clk_div : process (reset,clock_100MHz) -- 100MHz to 4Hz, 50% duty cycle, clock divider

variable count_clk : integer range 0 to 12499999 := 0;  -- (100.000.000/4)/2 - 1

begin

   if reset = '1' then

      count_clk := 0;

      slow_clock <= '0';

   elsif (rising_edge(clock_100MHz)) then

            if (count_clk = 12499999) then

               slow_clock <= not slow_clock; -- to obtain a 50% duty cycle

               count_clk := 0;

            else

               count_clk := count_clk + 1;

                   end if;

          end if;

end process clk_div;


process(reset,slow_clock) -- pushbutton 3 fsm to use as switch

   variable c: integer :=0;

   variable state: integer :=0;

   begin

   if reset = '1' then

      state:= 0;

      button_fsm <= '0';

   elsif rising_edge(slow_clock) then

      if button = '0' then
```

```
            if c > 0 then
               state:= 0;
            end if;
         else
            state := 1;
            c := c+1;
         end if;
         if state = 0 then
            button_fsm <= '0';
         else
            button_fsm <= '1';
         end if;
      end if;
      end process;


process(reset,clock_100MHz)
variable a: std_logic_vector(7 downto 0) := "00000000";
variable b: std_logic_vector(7 downto 0) := "00000000";
variable state: std_logic:= '0';
begin
   if reset = '1' then
      a := "00000000";
      b := "00000000";
      state:= '0';
   elsif rising_edge(clock_100MHz) then
```

```
    if button_fsm = '1' then

        a := switches(15 downto 8);

        b := switches(7 downto 0);

        state:= '1';

    end if;



    if a < b then

        b := b - a;

        if a = b and state = '1' then

            gcd1 <= a;

            state:= '0';

        end if;

    elsif b < a then

        a := a - b;

        if a = b and state = '1' then

            gcd1 <= a;

            state:= '0';

        end if;

    else

        if state = '1' then

            gcd1 <= a;

            state:= '0';

        end if;

    end if;

end if;
end process;

end Behavioral;
```

Testbench:

```
library ieee;

use ieee.std_logic_1164.all;


entity tb_gcd is

end tb_gcd;


architecture tb of tb_gcd is


  component gcd

     port (clock_100MHz : in std_logic;

         reset       : in std_logic;

         switches    : in std_logic_vector (15 downto 0);

         gcd1        : out std_logic_vector (7 downto 0);

         button      : in std_logic);

  end component;


  signal clock_100MHz : std_logic;

  signal reset       : std_logic;

  signal switches    : std_logic_vector (15 downto 0);

  signal gcd1        : std_logic_vector (7 downto 0);

  signal button      : std_logic;
```

```
    constant TbPeriod : time := 1000 ns; -- EDIT Put right period here

    signal TbClock : std_logic := '0';

    signal TbSimEnded : std_logic := '0';


begin


    dut : gcd

    port map (clock_100MHz => clock_100MHz,

              reset        => reset,

              switches     => switches,

              gcd1         => gcd1,

              button       => button);

    TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else '0';

    clock_100MHz <= TbClock;

    stimuli : process

    begin

        switches <= (others => '0');

        button <= '0';

        reset <= '1';

        wait for 100 ns;

        reset <= '0';

        wait for 100 ns;

        wait for 100 * TbPeriod;

        switches <= "1000110000001100";

            wait for 100 * TbPeriod;
            button <= '1'

            wait for 100 * TbPeriod;

            button <= '0'


        TbSimEnded <= '1';

        wait;

    end process;

end tb;

configuration cfg_tb_gcd of tb_gcd is

    for tb

    end for;

end cfg_tb_gcd;
```

Constraints:

```
#clock

set_property PACKAGE_PIN W5 [get_ports clock_100MHz]

    set_property IOSTANDARD LVCMOS33 [get_ports clock_100MHz]

set_property PACKAGE_PIN T18 [get_ports reset]

    set_property IOSTANDARD LVCMOS33 [get_ports reset]

# Switches

set_property PACKAGE_PIN V17 [get_ports switches[0]]

    set_property IOSTANDARD LVCMOS33 [get_ports switches[0]]

set_property PACKAGE_PIN V16 [get_ports switches[1]]

    set_property IOSTANDARD LVCMOS33 [get_ports switches[1]]

set_property PACKAGE_PIN W16 [get_ports switches[2]]

    set_property IOSTANDARD LVCMOS33 [get_ports switches[2]]
```

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 7
31.05.2021

```
set_property PACKAGE_PIN W17 [get_ports switches[3]]
    set_property IOSTANDARD LVCMOS33 [get_ports switches[3]]
set_property PACKAGE_PIN W15 [get_ports switches[4]]
    set_property IOSTANDARD LVCMOS33 [get_ports switches[4]]
set_property PACKAGE_PIN V15 [get_ports switches[5]]
    set_property IOSTANDARD LVCMOS33 [get_ports switches[5]]
set_property PACKAGE_PIN W14 [get_ports switches[6]]
    set_property IOSTANDARD LVCMOS33 [get_ports switches[6]]
set_property PACKAGE_PIN W13 [get_ports switches[7]]
    set_property IOSTANDARD LVCMOS33 [get_ports switches[7]]
set_property PACKAGE_PIN V2 [get_ports switches[8]]
    set_property IOSTANDARD LVCMOS33 [get_ports switches[8]]
set_property PACKAGE_PIN T3 [get_ports switches[9]]
    set_property IOSTANDARD LVCMOS33 [get_ports switches[9]]
set_property PACKAGE_PIN T2 [get_ports switches[10]]
    set_property IOSTANDARD LVCMOS33 [get_ports switches[10]]
set_property PACKAGE_PIN R3 [get_ports switches[11]]
    set_property IOSTANDARD LVCMOS33 [get_ports switches[11]]
set_property PACKAGE_PIN W2 [get_ports switches[12]]
    set_property IOSTANDARD LVCMOS33 [get_ports switches[12]]
set_property PACKAGE_PIN U1 [get_ports switches[13]]
    set_property IOSTANDARD LVCMOS33 [get_ports switches[13]]
set_property PACKAGE_PIN T1 [get_ports switches[14]]
    set_property IOSTANDARD LVCMOS33 [get_ports switches[14]]
set_property PACKAGE_PIN R2 [get_ports switches[15]]
```

Mehmet Bayık
21802166
EE-102 Section 3
Lab Work 7
31.05.2021

```
    set_property IOSTANDARD LVCMOS33 [get_ports switches[15]]


#leds

set_property PACKAGE_PIN U16 [get_ports leds[0]]

    set_property IOSTANDARD LVCMOS33 [get_ports leds[0]]

set_property PACKAGE_PIN E19 [get_ports leds[1]]

    set_property IOSTANDARD LVCMOS33 [get_ports leds[1]]

set_property PACKAGE_PIN U19 [get_ports leds[2]]

    set_property IOSTANDARD LVCMOS33 [get_ports leds[2]]

set_property PACKAGE_PIN V19 [get_ports leds[3]]

    set_property IOSTANDARD LVCMOS33 [get_ports leds[3]]

set_property PACKAGE_PIN W18 [get_ports leds[4]]

    set_property IOSTANDARD LVCMOS33 [get_ports leds[4]]

set_property PACKAGE_PIN U15 [get_ports leds[5]]

    set_property IOSTANDARD LVCMOS33 [get_ports leds[5]]

set_property PACKAGE_PIN U14 [get_ports leds[6]]

    set_property IOSTANDARD LVCMOS33 [get_ports leds[6]]

set_property PACKAGE_PIN V14 [get_ports leds[7]]

    set_property IOSTANDARD LVCMOS33 [get_ports leds[7]]


#center button

set_property PACKAGE_PIN U18 [get_ports button]

    set_property IOSTANDARD LVCMOS33 [get_ports button]
```