Eray Gündoğdu 28.11.2022

EEE102 LAB06:Greatest Common Divisor

Purpose:

Aim of this experiment is to design a 8 bit greatest common divisor finder by using substractors,comparators and registers.

Research Questions:

-What was your algorithm to calculate the GCD?

Simplified Eucledian Algorith is used to calculate GCD.

-Is your module a combinational circuit or an FSM? If the latter, is it a Moore machine or a Mealy machine? 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?

My module is FSM,Moore Machine.Even though combinational circuits are faster and cheaper than FSM.In this design memory elements are included.Keeping the memory or registering new values are much simpler with FSM however it can not be said that FSM is cheaper and faster tahn combinational circuit.

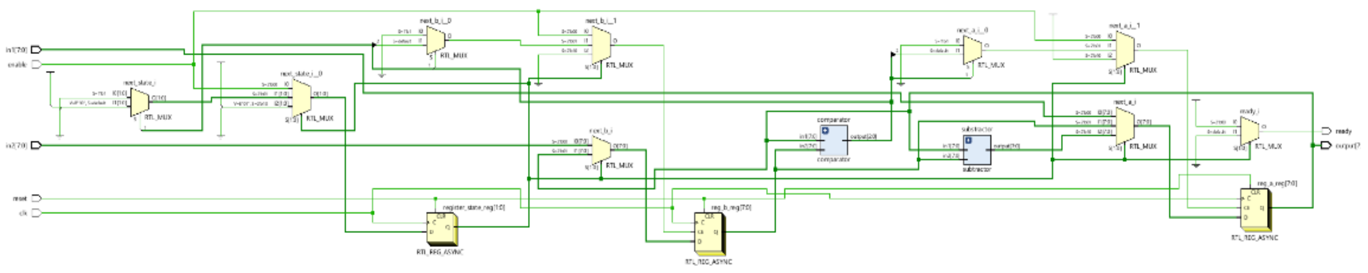
-How many clock cycles did it take in your simulation to calculate the GCD? Do you think this can be optimized? How so?

It took 296 clock cycles.By reducing operations with clock input,amount of clock cyles can be optimized.

Methadology:

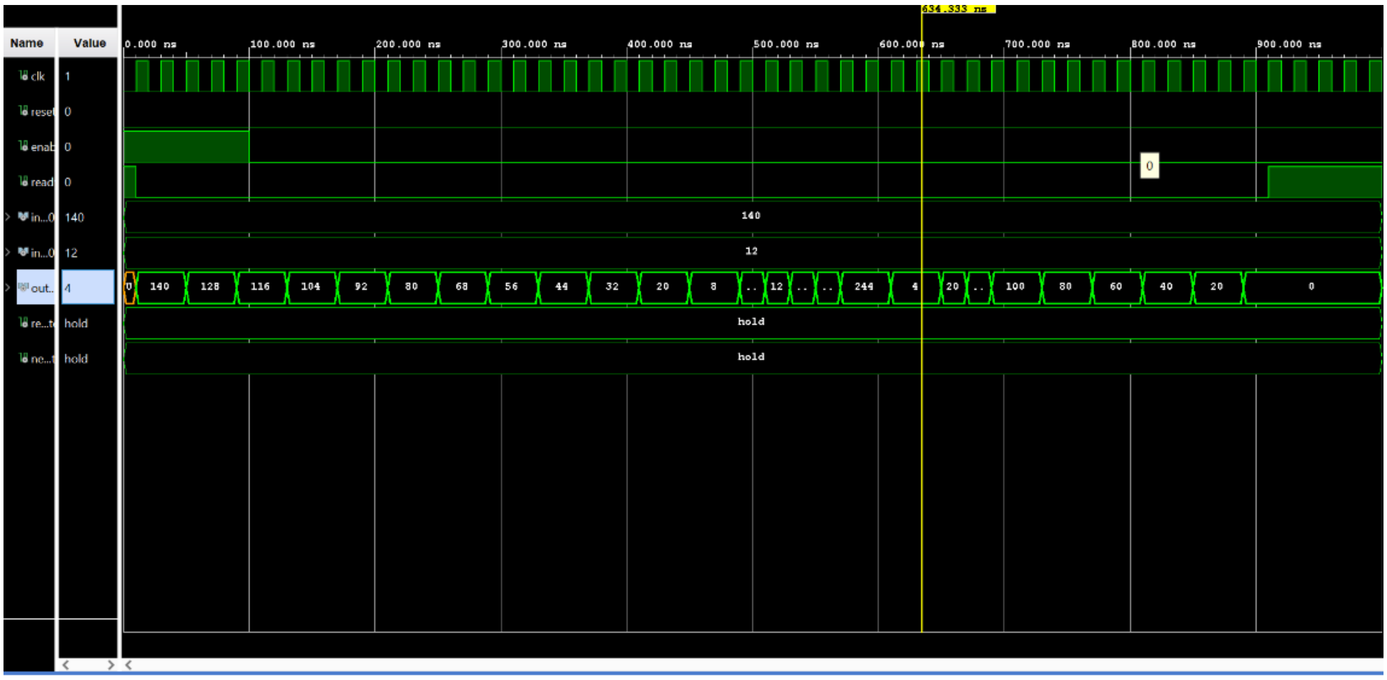
Greatest common divisor algorithm is designed by using a substractor and comparator.Initially,the design compares two 8 bit numbers by the 3 bit output(010-equal,001-greater,100 lesser). And if they are not equal smaller one is substracted from the greater one until both numbers are equal(Simplified Eucledian Algorithm).When they are equal the output is demonstrated on assigned LEDs as the output which is greatest common divisor.The process is initialized by U17 button which is asiggned to “start”.To find the greatest common divisor of another pair reset button is asigned to T18.Comparator is designed by obtainig every bit for the output in terms of inputs’ bits by Karnaugh Map.Then, with using logic gates output for the comparator is obtained.Substractor was obtained from LAB04.In gcd file,obtained values from substraction are registered as new input until both of the inputs are equal.Testbench is written in order to obtain simulation for a better understanding and design is implemented to BASYS3.

RTL Schematic:



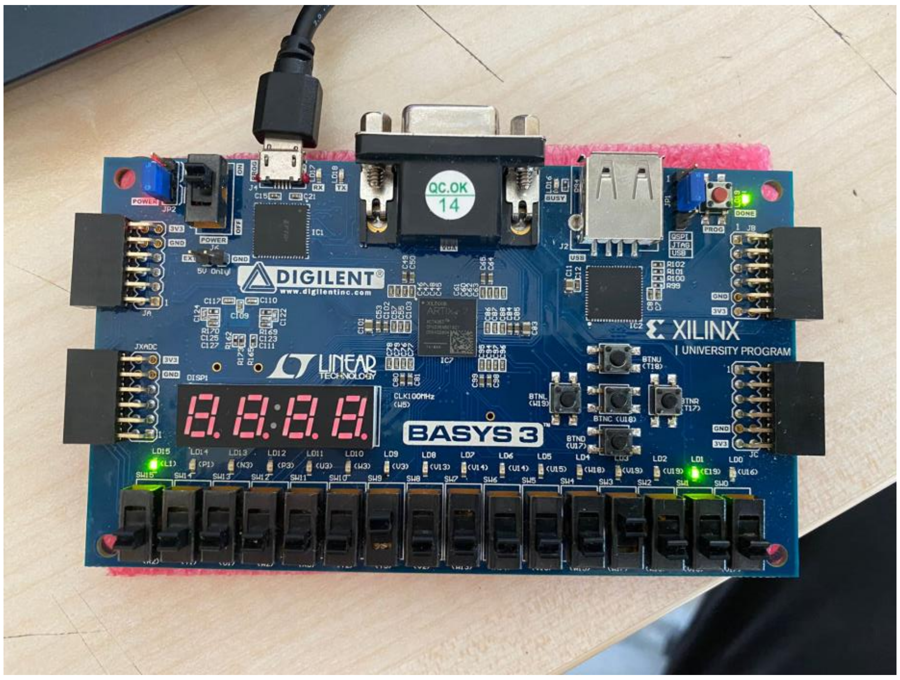
RTL schematic of greatest common diviser finder(gcd)

Simulation:

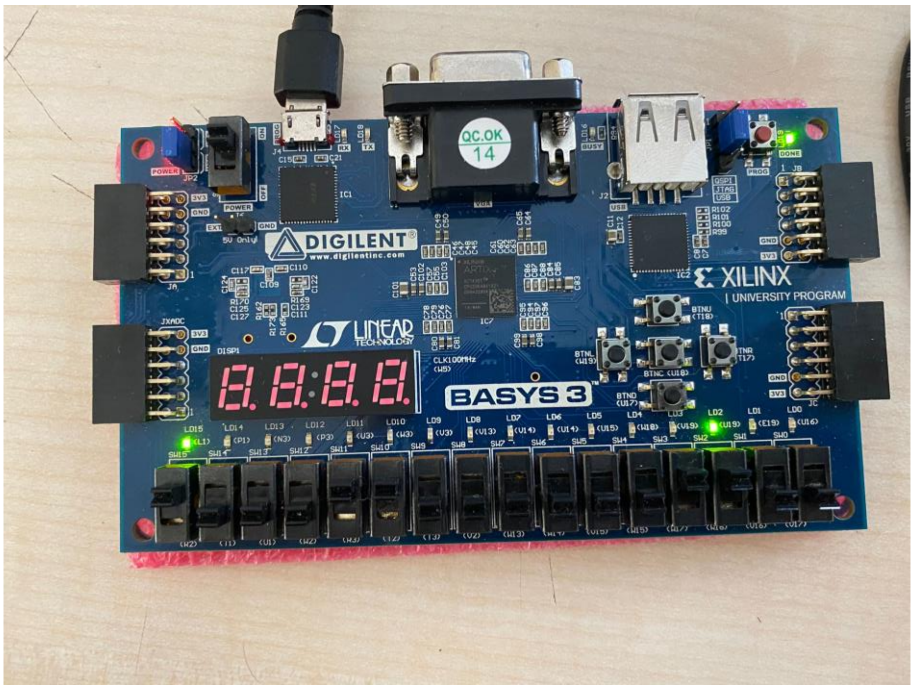


Simulation of in1=”10001100” and in2=”00001100

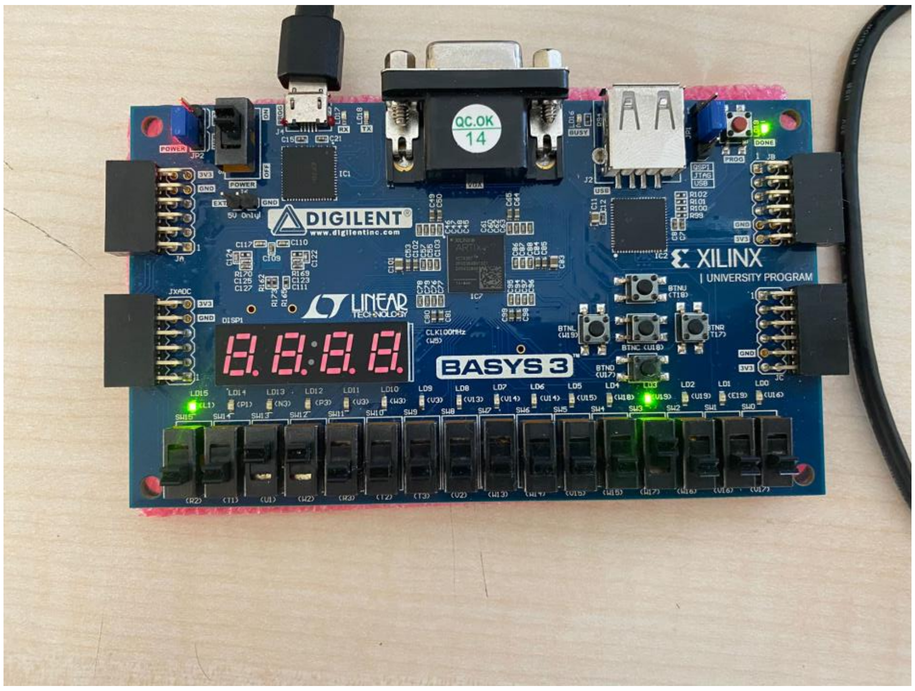
BASYS3 Demonstration:



in1=”00000010”,in2=”00000100”,output=”00000010”,ready=”1”



in1=”1001100”,in2=”00001100”,output=”00000100”,ready=”1”



in1="00110000",in2="00000100" ,output="00000100",ready="1"

Conclusion:

In this experiment,greatest common divisor is found by using Simplified Eucledian Algorithm.Substractor and comparator was necessary to use this alghorithm.Demonstration on BASYS3 was successful and ı feel more confident using registers in my designs.

Appendix:

Code of substractor

```
library ieee;
use ieee.std_logic_1164.all;
entity subtractor is
 port (in1, in2: IN STD_LOGIC_VECTOR ( 7 DOWNTO 0);
 output : OUT STD_LOGIC_VECTOR ( 7 DOWNTO 0);
 tst: out std_logic
 );
end subtractor;
architecture behavioral of subtractor is
 signal b1,b2,b3,b4,b5,b6,b7,b8: std_logic;
BEGIN
 tst <= in1(0) XOR in2(0) xor '0';
 output(0) <= in1(0) XOR in2(0) xor '0';
 b1 <= (not(in1(0)) and '0') or (not(in1(0)) and in2(0)) or (in2(0) and '0');
 output(1) <= in1(1) XOR in2(1) xor b1;
 b2 <= (not(in1(1)) and b1) or (not(in1(1)) and in2(1)) or (in2(1) and b1) ;
 output(2) <= in1(2) XOR in2(2) xor b2;
 b3 <= (not(in1(2)) and b2) or (not(in1(2)) and in2(2)) or (in2(2) and b2);
 output(3) <= in1(3) XOR in2(3) xor b3;
 b4 <= (not(in1(3)) and b3) or (not(in1(3)) and in2(3)) or (in2(3) and b3);
 output(4) <= in1(4) XOR in2(4) xor b4;
 b5 <= (not(in1(4)) and b4) or (not(in1(4)) and in2(4)) or (in2(4) and b4);
 output(5) <= in1(5) XOR in2(5) xor b5;
 b6 <= (not(in1(5)) and b5) or (not(in1(5)) and in1(5)) or (in2(5) and b5);
 output(6) <= in1(6) XOR in2(6) xor b6;
 b7 <= (not(in1(6)) and b6) or (not(in1(6)) and in2(6)) or (in2(6) and b6);
 output(7) <= in1(7) XOR in2(7) xor b7;
END behavioral;C
```

Code of comparator:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity comparator is
 port (
 in1, in2: in unsigned(7 downto 0);
 output: out std_logic_vector(2 downto 0)
 );
end entity;
architecture behavioral of comparator is
 signal equal1: std_logic_vector(7 downto 0);
 signal great1, great2: std_logic;
 signal result: std_logic_vector(2 downto 0);
 signal equal2,grtr, lssr: std_logic;
begin

 equal1(0) <= (not in1(0)) xnor (not in2(0));
 equal1(1) <= (not in1(1)) xnor (not in2(1));
 equal1(2) <= (not in1(2)) xnor (not in2(2));
 equal1(3) <= (not in1(3)) xnor (not in2(3));
 equal1(4) <= (not in1(4)) xnor (not in2(4));
 equal1(5) <= (not in1(5)) xnor (not in2(5));
 equal1(6) <= (not in1(6)) xnor (not in2(6));
 equal1(7) <= (not in1(7)) xnor (not in2(7));

 equal1ual2<= '1' when equal1 = x"FF" else '0';

 grtr <= (in1(7) and not(in2(7)))or
 (in1(6) and not(in2(6)) and equal1(7))or
 (in1(5) and not(in2(5)) and equal1(7) and equal1(6))or
 (in1(4) and not(in2(4)) and equal1(7) and equal1(6) and equal1(5))or
 (in1(3) and not(in2(3)) and equal1(7) and equal1(6) and equal1(5) and equal1(4))or
 (in1(2) and not(in2(2)) and equal1(7) and equal1(6) and equal1(5) and equal1(4) and equal1(3))or
 (in1(1) and not(in2(1)) and equal1(7) and equal1(6) and equal1(5) and equal1(4) and equal1(3) and
equal1(2))or
 (in1(0) and not(in2(0)) and equal1(7) and equal1(6) and equal1(5) and equal1(4) and equal1(3) and
equal1(2) and equal1(1));
lssr <= (in2(7) and not(in1(7)))or
 (in2(6) and not(in1(6)) and equal1(7))or
 (in2(5) and not(in1(5)) and equal1(7) and equal1(6))or
 (in2(4) and not(in1(4)) and equal1(7) and equal1(6) and equal1(5))or
 (in2(3) and not(in1(3)) and equal1(7) and equal1(6) and equal1(5) and equal1(4))or
 (in2(2) and not(in1(2)) and equal1(7) and equal1(6) and equal1(5) and equal1(4) and equal1(3))or
 (in2(1) and not(in1(1)) and equal1(7) and equal1(6) and equal1(5) and equal1(4) and equal1(3) and
equal1(2))or
 (in2(0) and not(in1(0)) and equal1(7) and equal1(6) and equal1(5) and equal1(4) and equal1(3) and
equal1(2) and equal1(1));
result(2) <= lssr;
 result(0) <= grtr;
 result(1) <= equal2;
 output <= result;
end behavioral;
```

Code of gcd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity gcd is
 port(
 clk, reset: in std_logic;
 enable: in std_logic;
 in1, in2: in std_logic_vector(7 downto 0);
 ready: out std_logic;
 output: out std_logic_vector(7 downto 0)
 );
end gcd ;

architecture behavioral of gcd is
 type state_type is (hold, replace, substract);
 signal register_state, next_state : state_type;

 signal reg1, reg2, next1, next2: unsigned(7 downto 0);
 signal comp1,comp2 : unsigned(7 downto 0);

 signal subs_2, subs_1, subs_out: std_logic_vector(7 downto 0);
 signal comp_out: std_logic_vector(2 downto 0);

begin
 ---------------------------------------------------------------
 comparator: entity work.comparator(behavioral)
 port map(in1 => comp1, in2=> comp2, output=> comp_out);
 substractor: entity work.substractor(behavioral)
```

```
port map(in1 => subs_1, in2 => subs_2, output => subs_out);

-----------------------------------------------------------------

process(clk,reset)

begin

if reset='1' then

register_state <= hold;

reg1 <= (others=>'0');

reg2 <= (others=>'0');

elsif (rising_edge(clk)) then

register_state <= next_state;

reg1 <= next1;

reg2 <= next2;

end if;

end process;

-----------------------------------------------------------------

process(register_state,reg1,reg2,enable,in1,in2)

begin

next1 <= reg1;

next2 <= reg2;

comp1 <= reg1;

comp2 <= reg2;

subs_1 <= std_logic_vector(reg1);

subs_2 <= std_logic_vector(reg2);


case register_state is

-----------------------------------------------------------------

when hold =>

if enable = '1' then

next1 <= unsigned(in1);

next2 <= unsigned(in2);

next_state <= replace;
```

```
 else
 next_state <= hold;
 end if;
 ----------------------------------------------------------------
 when replace =>
 if (comp_out(1) = '1') then
 next_state <= hold;
 else
 if(comp_out(2) = '1') then
 next1 <= reg2;
 next2 <= reg1;
 end if;
 next_state <= substract;
 end if;
 ----------------------------------------------------------------
 when substract =>
 next1 <= unsigned(subs_out);
 next_state <= replace;
 end case;
 end process;
 ----------------------------------------------------------------
 ready <= '1' when register_state = hold else '0';
 output <= std_logic_vector(reg1);
 ----------------------------------------------------------------
end behavioral;
```

Code of testbench

```
library ieee;
use ieee.std_logic_1164.all;
----------------------------------------------------------------------
entity testbench1 is
end testbench1;
----------------------------------------------------------------------
architecture behavioral of testbench1 is
 signal clk, reset,enable,ready: std_logic;
 signal in1, in2, output : std_logic_vector (7 downto 0);
 type state_type is (hold, replace, subtract);
 signal register_state, next_state : state_type;
begin
----------------------------------------------------------------------
 dut: entity work.gcd(behavioral)
 port map(clk,reset,enable,in1,in2,ready,output);
----------------------------------------------------------------------
 clock_process: process begin
 clk <= '0';
 wait for 10ns;
 clk <= '1';
 wait for 10ns;
 end process;
----------------------------------------------------------------------
 stim_process: process begin
 enable <= '1';
 in1 <= "10001100";
 in2 <= "00001100";
 reset <= '0';
 wait for 100ns;
 enable <= '0';
 wait;
 end process;
-------------------------------------------------------------------
end behavioral;
-------------------------------------------------------------------
configuration behavioral of testbench1 is
 for behavioral
 end for;
end behavioral;
```

Code of constraints

```
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 {in1[0]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {in1[0]}]
set_property PACKAGE_PIN V16 [get_ports {in1[1]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {in1[1]}]
set_property PACKAGE_PIN W16 [get_ports {in1[2]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {in1[2]}]
set_property PACKAGE_PIN W17 [get_ports {in1[3]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {in1[3]}]
set_property PACKAGE_PIN W15 [get_ports {in1[4]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {in1[4]}]
set_property PACKAGE_PIN V15 [get_ports {in1[5]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {in1[5]}]
set_property PACKAGE_PIN W14 [get_ports {in1[6]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {in1[6]}]
set_property PACKAGE_PIN W13 [get_ports {in1[7]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {in1[7]}]
set_property PACKAGE_PIN V2 [get_ports {in2[0]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {in2[0]}]
set_property PACKAGE_PIN T3 [get_ports {in2[1]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {in2[1]}]
set_property PACKAGE_PIN T2 [get_ports {in2[2]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {in2[2]}]
set_property PACKAGE_PIN R3 [get_ports {in2[3]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {in2[3]}]
set_property PACKAGE_PIN W2 [get_ports {in2[4]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {in2[4]}]
set_property PACKAGE_PIN U1 [get_ports {in2[5]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {in2[5]}]
set_property PACKAGE_PIN T1 [get_ports {in2[6]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {in2[6]}]
set_property PACKAGE_PIN R2 [get_ports {in2[7]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {in2[7]}]
# LEDs
set_property PACKAGE_PIN U16 [get_ports {output[0]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {output[0]}]
set_property PACKAGE_PIN E19 [get_ports {output[1]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {output[1]}]
set_property PACKAGE_PIN U19 [get_ports {output[2]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {output[2]}]
set_property PACKAGE_PIN V19 [get_ports {output[3]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {output[3]}]
set_property PACKAGE_PIN W18 [get_ports {output[4]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {output[4]}]
set_property PACKAGE_PIN U15 [get_ports {output[5]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {output[5]}]
set_property PACKAGE_PIN U14 [get_ports {output[6]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {output[6]}]
set_property PACKAGE_PIN V14 [get_ports {output[7]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {output[7]}]
set_property PACKAGE_PIN L1 [get_ports {ready}]
 set_property IOSTANDARD LVCMOS33 [get_ports {ready}]
#Buttons
set_property PACKAGE_PIN T18 [get_ports reset]
 set_property IOSTANDARD LVCMOS33 [get_ports reset]
set_property PACKAGE_PIN U17 [get_ports enable]
 set_property IOSTANDARD LVCMOS33 [get_ports enable]
```