Humble Counter

o Embedded Systems

o Lab Report #1

o Fareen Pourmoussavian

o 2/28/19

• Purpose

o The purpose of this lab is to understand the humble counter. It uses a high frequency input clock which is slowed down and used to drive a bidirectional counter.

• For Each Part

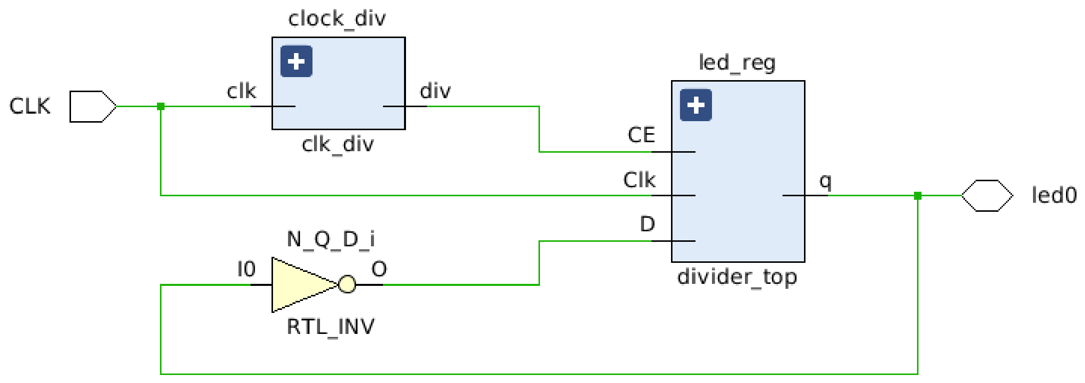
o Theory of Operation

♣ Given a value using the switches, the leds should light up in a way that shows the value getting incremented or decremented.

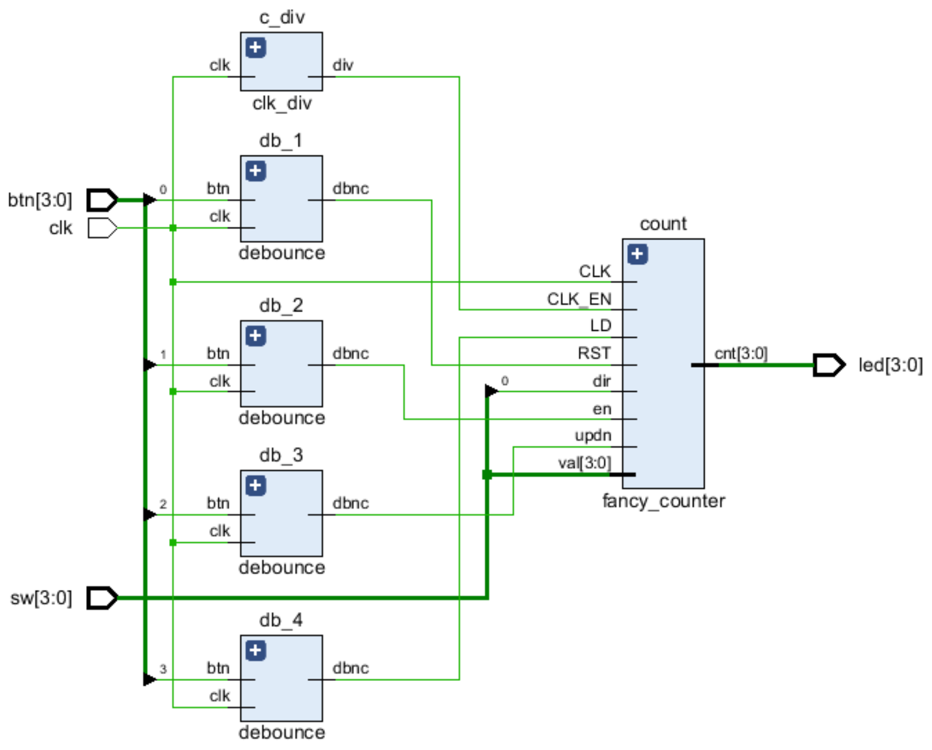
♣ Initially I'm expecting it not to work and obviously have to go back and debug but assuming all goes well, then I can change the switches and load in the value and have the counter increment or decrement the counter when I press the corresponding buttons.

o Schematic Diagram (as desired)

Divider_top



Counter_top



o Design

Library IEEE;

use IEEE.std_logic_1164.all,IEEE.numeric_std.all;

```
entity divider_top is
port(Clk,CE,D : in std_logic;
    q :out std_logic);
end divider_top;

architecture div_top of divider_top is
signal tmp : std_logic;
begin

   process(Clk)
   begin
       if(rising_edge(clk)) then
           if(CE = '1') then
               Q<=Not(D);
           else
               Q<= D;
           end if;
       end if;
       end process;
end div_top;

Library IEEE;
use IEEE.std_logic_1164.all,IEEE.numeric_std.all;

entity crt_1 is
port(CLK :in std_logic;
    led0 :inout std_logic);
end crt_1;
```

```
architecture crt_1_sim of crt_1 is

component divider_top

    port(Clk,CE,D : in std_logic;

         q :out std_logic);

 end component;


component clk_div

port(clk :in std_logic;

    div :out std_logic);

end component;


signal div_out, Q_D,N_Q_D : std_logic;


begin

N_Q_D <=NOT(led0);

clock_div : clk_Div

port map(CLK=>clk,

         div=>div_out);


led_reg : divider_top

port map(Clk=>clk,

         CE=>div_out,

         D=>N_Q_D,

         Q=>led0);

 end crt_1_sim;


library IEEE;
```

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

entity debounce is
port (clk : in std_logic;
      btn : in std_logic;
      dbnc : out std_logic);
end debounce;

architecture deb of debounce is
signal counter : std_logic_vector(22 downto 0);
signal btn_reg : std_logic_vector(1 downto 0);
begin

   process(clk)
   begin
   if(rising_edge(clk)) then
         btn_reg(0)<=btn; --bit 0 gets 1
         btn_reg(1)<=btn_reg(0);  --update bit 1 with bit 0 value
          if(btn_reg(1)='1') then --if bit 1 is a 1 increment counter
          counter<=std_logic_vector(unsigned(counter)+1);
             if(unsigned(counter)<2500000) then
                 dbnc<='0';
              else
                 dbnc<='1';
              end if;
          else--btn(1)==0
              counter<=(others=>'0');
              dbnc<='0';
```

```
            end if;
          end if; --end if button pressed or not
    end process;
end deb;



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

entity clk_div is
port(clk :in std_logic;
    div :out std_logic);
end clk_div;

architecture clk of clk_div is
signal counter : std_logic_vector (25 downto 0) := (others => '0');
begin
    process(clk)
    begin
        if(rising_edge(clk)) then
            if(unsigned(counter)<62499999) then
                counter <= std_logic_vector(unsigned(counter)+1);
                div <= '1';
            else
                counter <= (others => '0');
                div <= '0';
            end if;
              if (unsigned(counter) = 31250000) then
```

```
                  div <= '1';

               else
                  div <= '0';
               end if;
       end if;
     end process;
end clk;

library IEEE;
use ieee.std_logic_1164.all, ieee.numeric_std.all;

entity fancy_counter is
port(CLK,CLK_EN,LD,RST,dir,en,updn : in std_logic;
     val :in std_logic_vector(3 downto 0);
     cnt :out std_logic_vector(3 downto 0));
end fancy_counter;

architecture fc of fancy_counter is
signal count, count_d : std_logic_vector(3 downto 0) := (others=>'0');
signal direction : std_logic;
begin
    process(CLK)
    begin
    if( rising_edge(clk) and en='1')then
                if(rst='1')then
                    cnt<= (others=> '0');
                end if;
```

```
            if(clk_en='1') then
                if(ld='1') then
                    count<=val;
                    count_d<=count;
                    cnt<=count;
                end if;
                if(updn='1')then
                    direction<=dir;    --put dir into direction register
                    if(direction='1') then --if direction is 1 then it counts up
                         count<=std_logic_vector(unsigned(count)+1);
                         cnt<=count;
                         if(count=val) then --if you counted up to value reset
                           count<=(others=>'0');
                         end if;
                    elsif(direction='0') then --if direction is 0 then it counts down
                         count<=std_logic_vector(unsigned(count)-1);
                         cnt<=count;
                         if(count ="0000") then --if counting down reset back to original
                            count<=count_d;
                            cnt<=count;
                         end if;
                    end if;
                end if;
           end if;
 end if;
end process;

end fc;
```

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

 entity crkt_1 is
 port(btn, sw :in std_logic_vector(3 downto 0);
      clk :in std_logic;
      led :out std_logic_vector(3 downto 0));
end crkt_1;

architecture crkt_sim of crkt_1 is

component debounce
port (clk : in std_logic;
      btn : in std_logic;
      dbnc : out std_logic);
end component;

component clk_div
port(clk :in std_logic;
    div :out std_logic);
end component;

component fancy_counter
port(CLK,CLK_EN,LD,RST,dir,en,updn : in std_logic;
     val :in std_logic_vector(3 downto 0);
     cnt :out std_logic_vector(3 downto 0));
end component;
```

```
signal db1_rst, db2_en, db3_upd, db4_ld, cl_div_cl_en : std_logic;

begin
db_1 : debounce
port map(clk=> clk,
        btn=> btn(0),
        dbnc=>db1_rst);
db_2 :debounce
port map(clk=> clk,
        btn=> btn(1),
        dbnc=>db2_en);
db_3 :debounce
port map(clk=> clk,
        btn=> btn(2),
        dbnc=>db3_upd);
db_4 :debounce
port map(clk=> clk,
        btn=> btn(3),
        dbnc=>db4_ld);
c_div : clk_div
port map(clk=> clk,
        div=>cl_div_cl_en);

count : fancy_counter
port map(CLK=>clk,
        CLK_EN=>cl_div_cl_en,
        LD=>db4_ld,
        RST=>db1_rst,
        dir=>sw(0),
        en=>db2_en,
        updn=>db3_upd,
        val(3)=>sw(3),
        val(2)=>sw(2),
        val(1)=>sw(1),
        val(0)=>sw(0),
        cnt(3) =>led(3),
        cnt(2) =>led(2),
        cnt(1) =>led(1),
        cnt(0) =>led(0));
end crkt_sim;
```

o Test

♣ Test bench VHDL (not sure if required)

Clock_divider test bench

```
Library IEEE;
use IEEE.std_logic_1164.all,IEEE.numeric_std.all;


ENTITY Tb_clock_div IS
END Tb_clock_div;


architecture behave of Tb_clock_div is
Component clock_Divider
port(clk :in std_logic;
   led0 :out std_logic;
   clk_out :out std_logic_vector(24 downto 0));
end Component;
```

```
signal clk : std_logic := '0';

signal led0 : std_logic;
signal clk_out : std_logic_vector(24 downto 0);

constant clk_period : time := 10 ns;

begin
uut: clock_Divider PORT MAP (
clk => clk,
led0 => led0,
clk_out => clk_out);

clk_process :process
begin
clk <= '0';
wait for clk_period/2;
clk <= '1';
wait for clk_period/2;
end process;

stim_proc: process
begin
   wait for 100ns;
   wait for clk_period*10;
   wait;
   end process;
   end;
```

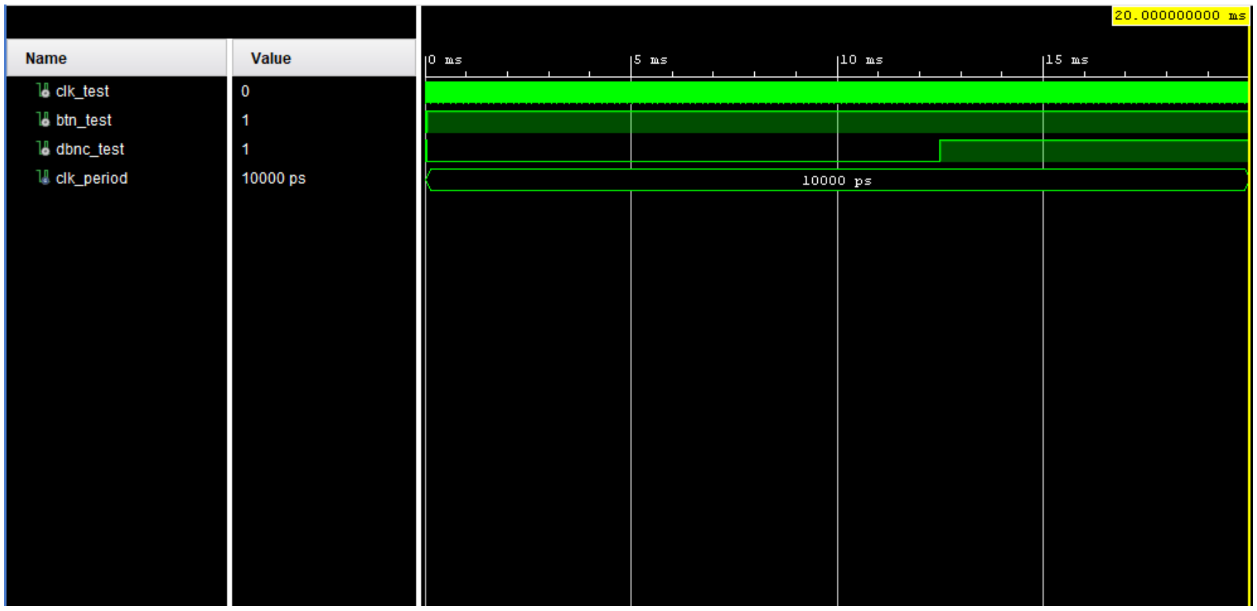
DebounceTest bench

```
Library IEEE;
use IEEE.std_logic_1164.all,IEEE.numeric_std.all;

ENTITY deb_t IS
END deb_t;

architecture behave of deb_t is
Component debounce
port(clk :in std_logic;
   btn :in std_logic;
   dbnc :out std_logic);
end Component;

signal clk_test : std_logic := '0';
signal btn_test : std_logic;
signal dbnc_test : std_logic;

constant clk_period : time := 10 ns;

begin
uut: debounce PORT MAP (
clk => clk_test,
btn => btn_test,
dbnc => dbnc_test);

clk_process :process
```

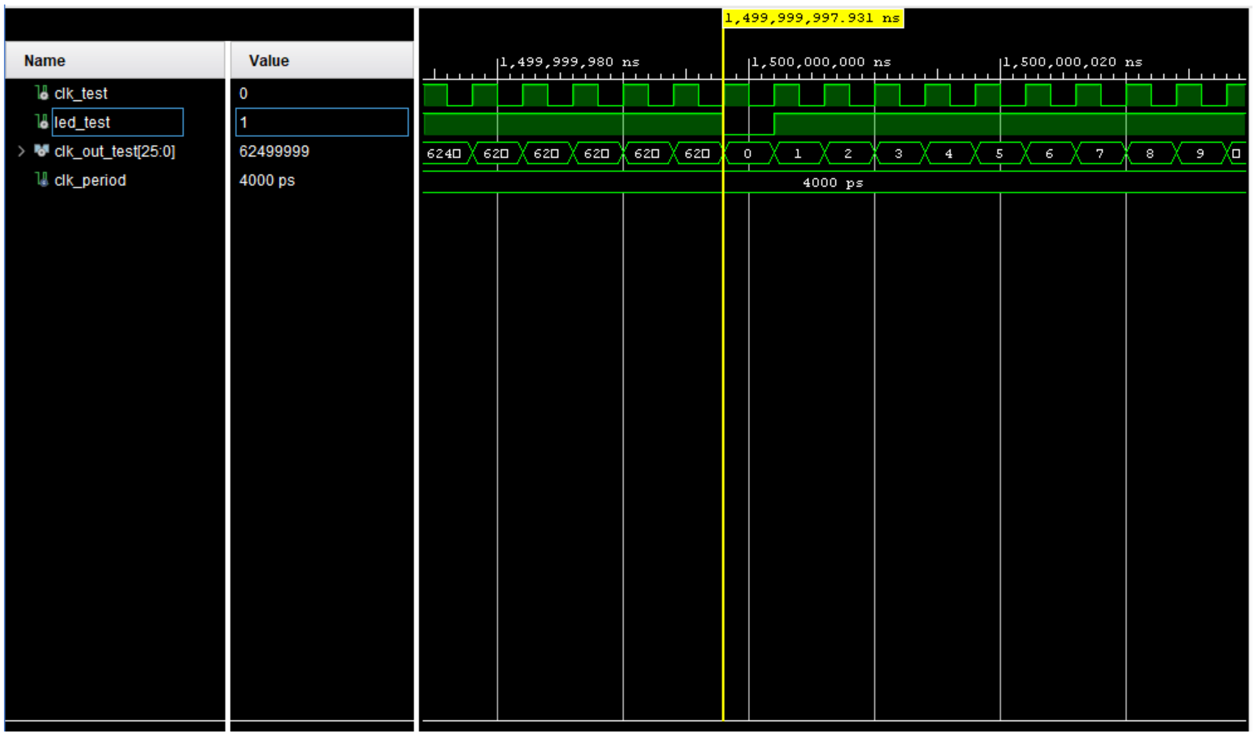
```
begin
clk_test <= '0';
wait for clk_period/2;
clk_test <= '1';
wait for clk_period/2;

end process;

stim_proc: process
begin
    wait for 100ns;
   btn_test<='1';
    wait;
    end process;
    end;
```

♣ simulation results (screen shots)

Debounce

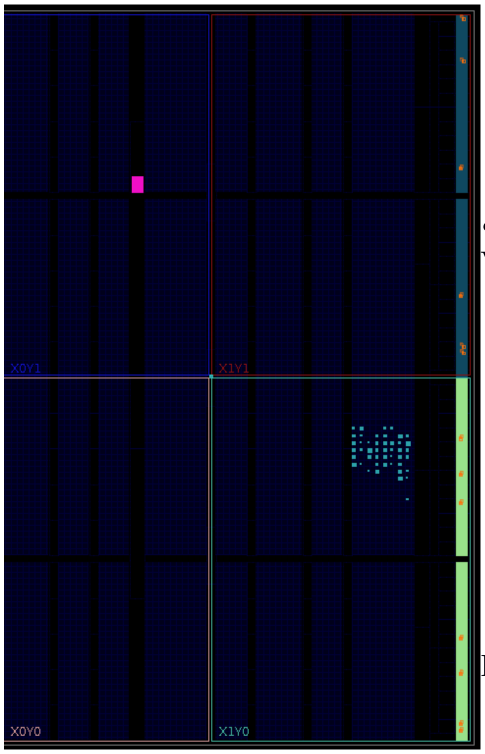


Clock divider test bench

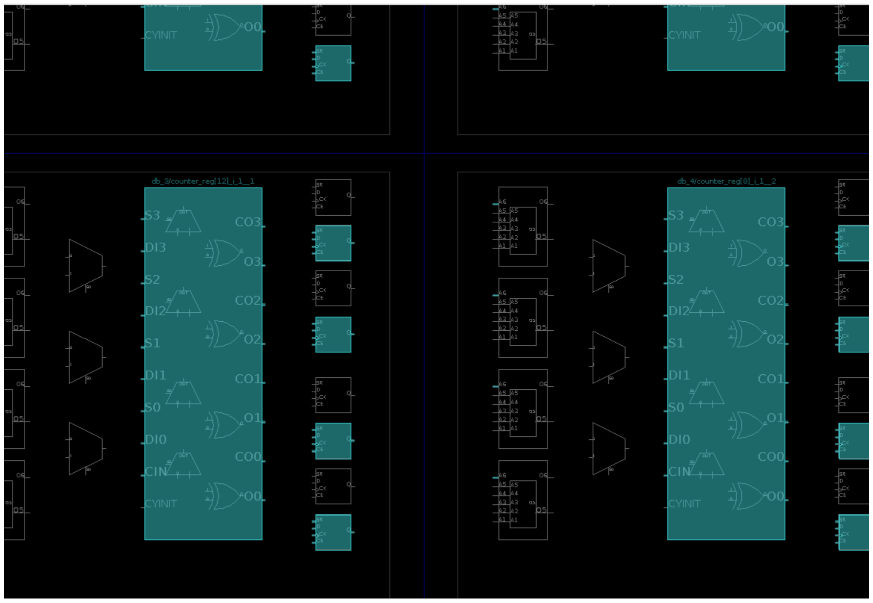


Name	Value
clk_test	0
led_test	1
clk_out_test[25:0]	62499999
clk_period	4000 ps

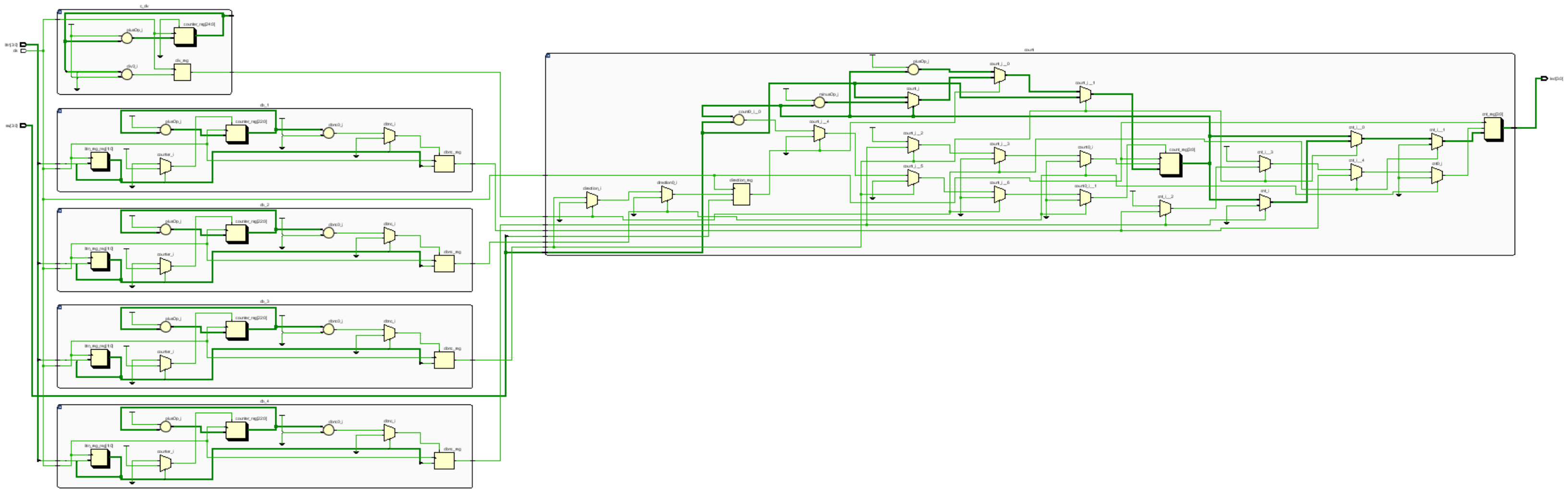
o Implementation (If required to demo on board)



♣ Vivado

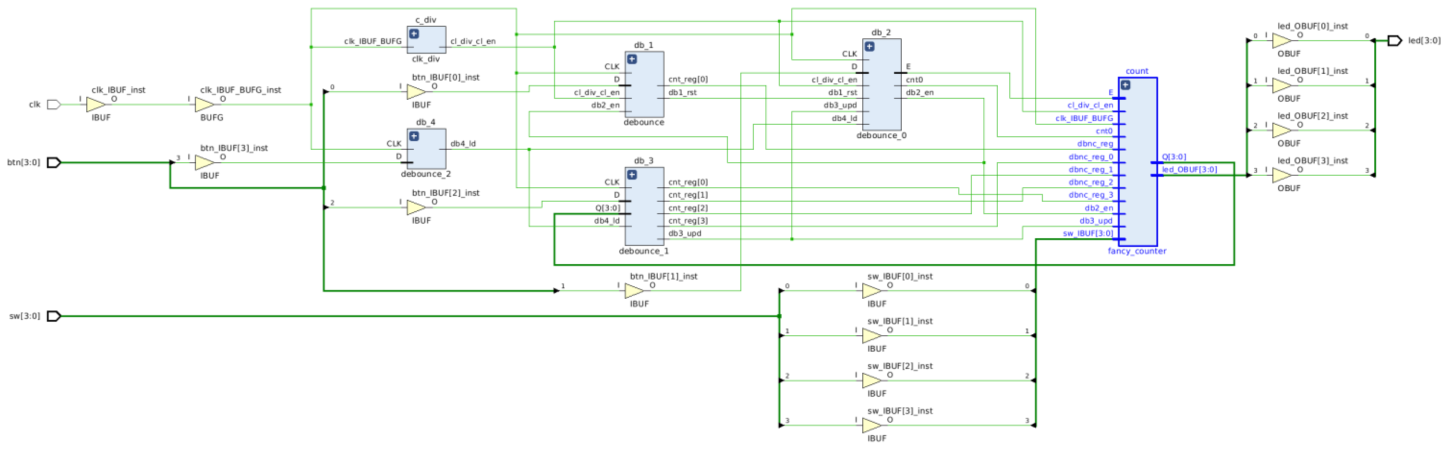


Elaboration Schematic

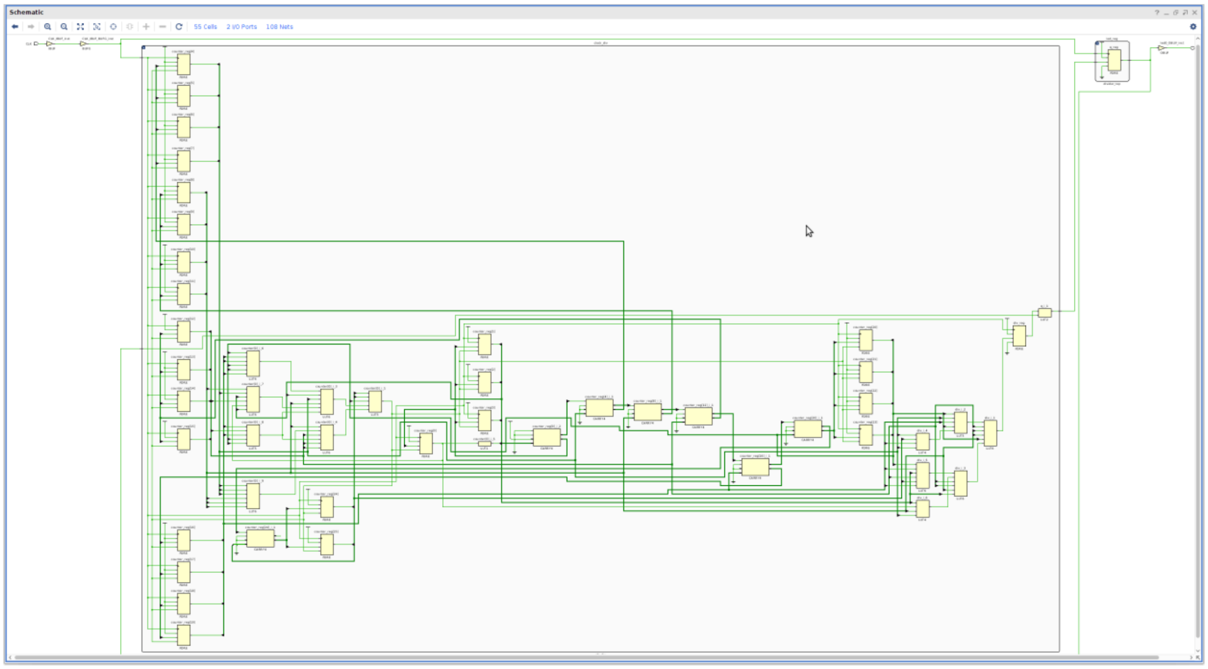


♣ Vivado Synthesis Schematic

Counter_Top

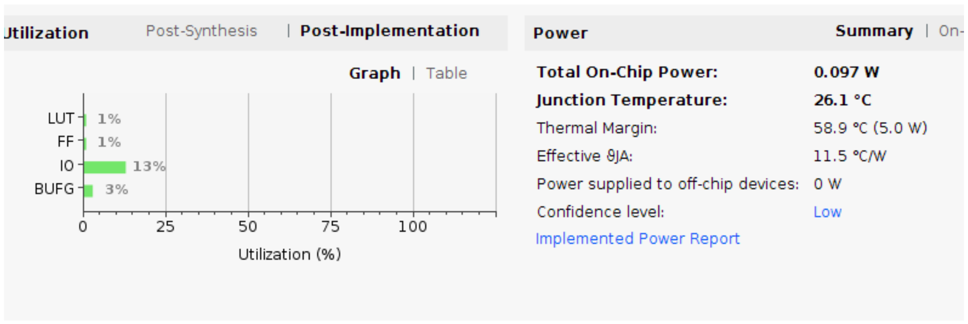


Divider_top(not sure if needed)



♣ Vivado Project Summary Images

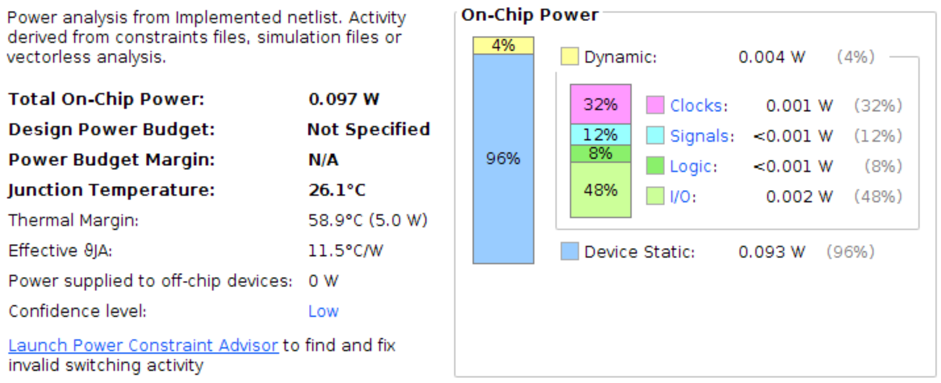
• Post-Synthesis Utilization Table



Total On-Chip Power:	**0.097 W**
Junction Temperature:	**26.1 °C**
Thermal Margin:	58.9 °C (5.0 W)
Effective ϑJA:	11.5 °C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Implemented Power Report

- On-chip Power Graphs


Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.097 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**26.1°C**
Thermal Margin:	58.9°C (5.0 W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	0.004 W	(4%)
Clocks:	0.001 W	(32%)
Signals:	<0.001 W	(12%)
Logic:	<0.001 W	(8%)
I/O:	0.002 W	(48%)
Device Static:	0.093 W	(96%)

♣ ZYBO_Master.xdc

- The only things that needed to be changed was the uncommenting of all the LED, Switches, buttons and clock signals so that those ports on the board would work when I try to implement my design on the board.

- Discussion

  o Q1.1: We need 62500000

o Q1.2: it requires 26 bits because if we do Log base 2 of 62500000 we get 25.8.. which rounds up to 26 bits

o Q2.1: The value of the button bounces between 1 and 0 until it stabilizes

o Q2.3: We need 2500000 ticks

o Q2.4: We would require 22 bits since the log base 2 of 2500000 is 21.2… we must round up and use 22 bits.

o Observations / Discoveries

♣ I actually learned how to code in behavioral and be able to combine those parts because I thought I had to code it all in structural so that I can connect everything together but actually I can make separate files and make components that refer to those entities and will do the same thing. I also learned that lab instructions are confusing and will need to re-read every task due to the convoluted directions.

o Questions / Follow Up

♣ I understand most of the things that were asked of me. The lab manual was a bit confusing to a degree where it took me around 20 minutes to understand that in part 1 the divider_top was asking for a D-Flip-Flop. The debouncer image in the manual was a bit confusing because what they were asking was the opposite of what the diagram was showing. I initially thought that we only needed to see that it was a logic ‘1’ for 4 consecutive times but after talking to the teaching assistants that was cleared up.

♣ The only thing I’m not 100% confident in is making test benches in general. That is the thing that took me a long time to figure out. Then I realized that once you make one test bench, other test benches use a similar format.