Embedded Systems
Lab Report #1
Timothy Langer RUID#: 189005424
2/28/2019

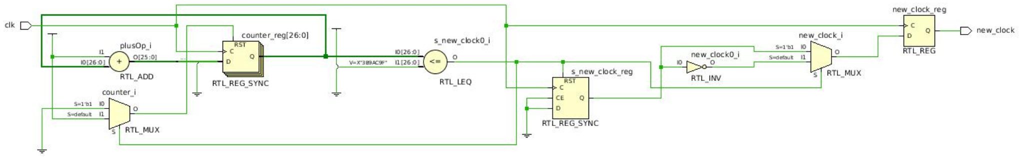
The purpose of this lab was to understand and be able to design a clock divider, inputting a high frequency and output a lower desired frequency. Also to understand the nature of a pressed button's input/output. Using this knowledge we can create a debounce system that can rectify the issues with buttons in a circuit. We than construct a bi-directional counter with conditions to its operation. The last purpose of this lab is the connection of these different components together to create a larger circuit. This helps understand how components are attached and create higher level designs.

1 My clock is Just Right

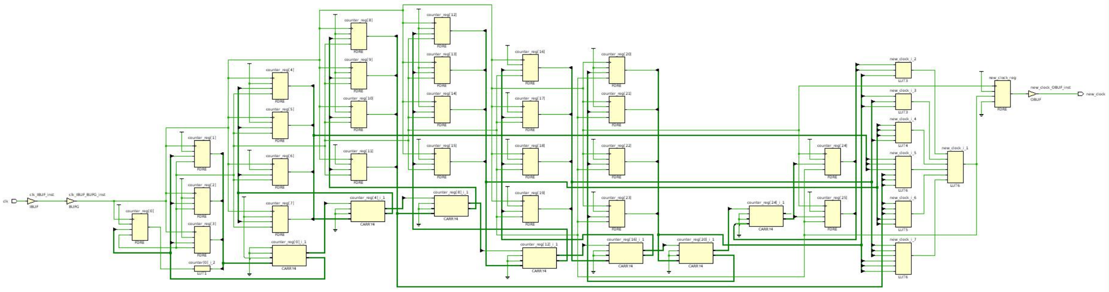
CLOCK_DIVIDER:

The purpose of this exercise is to be able to construct a clock divider that takes in a higher clock frequency, in our case 125Mhz, and output a lower desired frequency. By counting up to the divided value of 62.5 x10^6 we output a pulse at that time only.

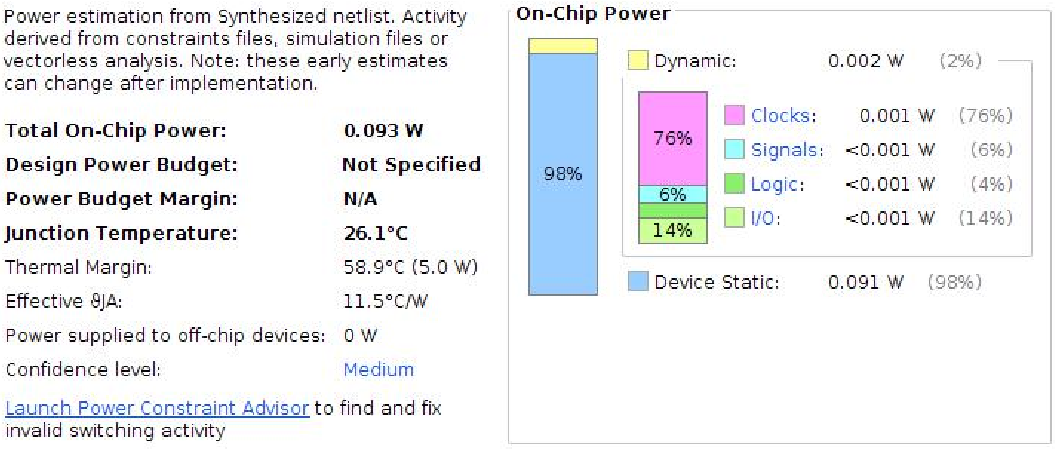
RTL SCHEMATIC:



SYNTHESIS SCHEMATIC:



POWER REPORT:



Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

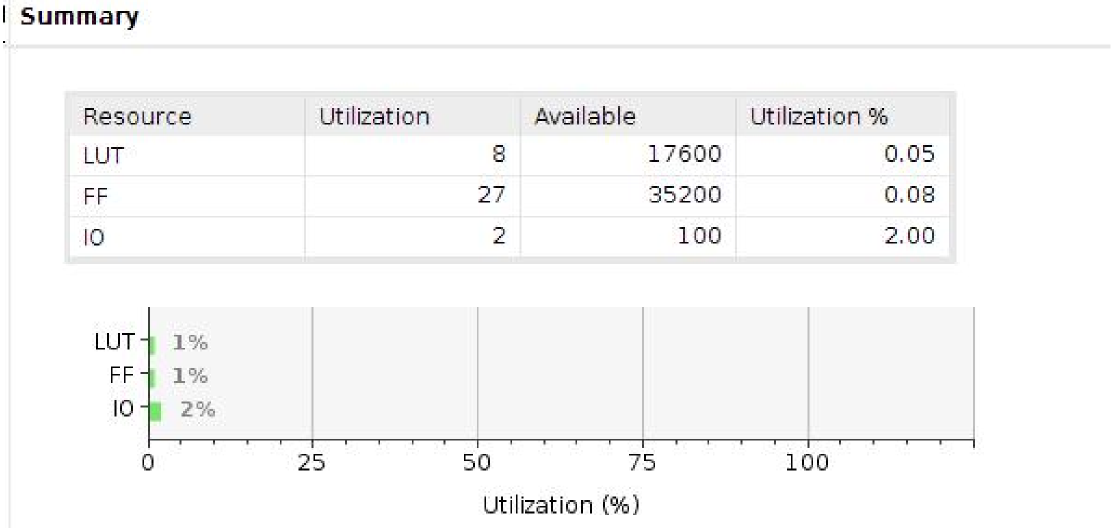
Total On-Chip Power:	**0.093 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**26.1°C**
Thermal Margin:	58.9°C (5.0 W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

🟨 Dynamic:	0.002 W	(2%)
🟪 Clocks:	0.001 W	(76%)
🟦 Signals:	<0.001 W	(6%)
🟩 Logic:	<0.001 W	(4%)
🟩 I/O:	<0.001 W	(14%)
🟦 Device Static:	0.091 W	(98%)

UTILIZATION:



Summary

Resource	Utilization	Available	Utilization %
LUT	8	17600	0.05
FF	27	35200	0.08
IO	2	100	2.00

SOURCE CODE:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;
entity clock_div is
port(
      clk   : in std_logic;
      new_clock : out std_logic
    );
 end clock_div;
architecture behavior of clock_div is
```

```
signal s_new_clock : std_logic:='0';
signal counter : std_logic_vector(26 downto 0) := (others => '0');
begin
 process(clk)
 begin
 new_clock <= s_new_clock;
 if rising_edge(clk) then
            if (unsigned(counter) <= 62499999) then
                s_new_clock <= '0';
                counter <= std_logic_vector(unsigned(counter) + 1);
         else
                new_clock <= (not s_new_clock);
                counter <= (others => '0');
         end if;
                 end if;

    end process;
end behavior;
```

TESTBENCH:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity clock_div_tb is
end clock_div_tb;
architecture Behavioral of clock_div_tb is
component clock_div
Port(
clk   : in std_logic;
new_clock : out std_logic
);
end component;

signal clk_in : std_logic :='0';
signal clk_out : std_logic :='0';

begin

uut: clock_div
Port map(
clk => clk_in,
new_clock => clk_out
);
```

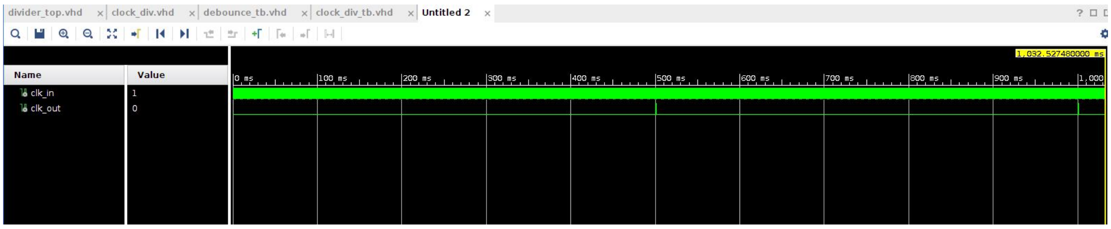
```
clk_process: process
begin
clk_in <= '1';
wait for 4ns;
clk_in <='0';
wait for 4ns;
end process;

end Behavioral;
```

CONSTRAINT: The clock is fed into a 125mHz clock on board. And led goes to the led pin specified below in the contraint file. PinL16 and M14 respectively

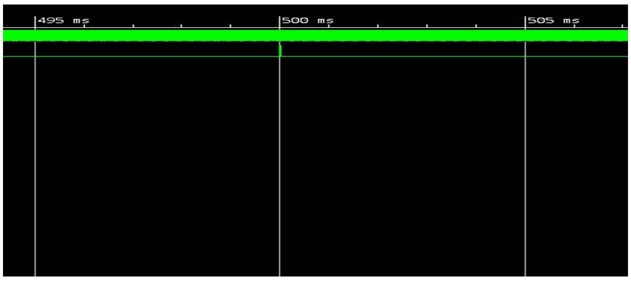
CONSTRAINT FILE:

```
set_property -dict { PACKAGE_PIN L16    IOSTANDARD LVCMOS33 } [get_ports { clk }]; #IO_L11P_T1_SRCC_35 Sch=sysclk
create_clock -add -name sys_clk_pin -period 8.00 -waveform {0 4} [get_ports { clk }];

##Switches
#set_property -dict { PACKAGE_PIN G15    IOSTANDARD LVCMOS33 } [get_ports { sw0 }]; #IO_L19N_T3_VREF_35 Sch=SW0
#set_property -dict { PACKAGE_PIN P15    IOSTANDARD LVCMOS33 } [get_ports { sw[1] }];  #IO_L24P_T3_34 Sch=SW1
#set_property -dict { PACKAGE_PIN W13    IOSTANDARD LVCMOS33 } [get_ports { sw[2] }]; #IO_L4N_T0_34 Sch=SW2
#set_property -dict { PACKAGE_PIN T16    IOSTANDARD LVCMOS33 } [get_ports { sw[3] }]; #IO_L9P_T1_DQS_34 Sch=SW3

##LEDs
set_property -dict { PACKAGE_PIN M14    IOSTANDARD LVCMOS33 } [get_ports { led }]; #IO_L23P_T3_35 Sch=LED0
#set_property -dict { PACKAGE_PIN M15    IOSTANDARD LVCMOS33 } [get_ports { led[1] }]; #IO_L23N_T3_35 Sch=LED1
#set_property -dict { PACKAGE_PIN G14    IOSTANDARD LVCMOS33 } [get_ports { led[2] }]; #IO_0_35=Sch=LED2
#set_property -dict { PACKAGE_PIN D18    IOSTANDARD LVCMOS33 } [get_ports { led[3] }]; #IO_L3N_T0_DQS_AD1N_35 Sch=LED3
```

SIMULATION:

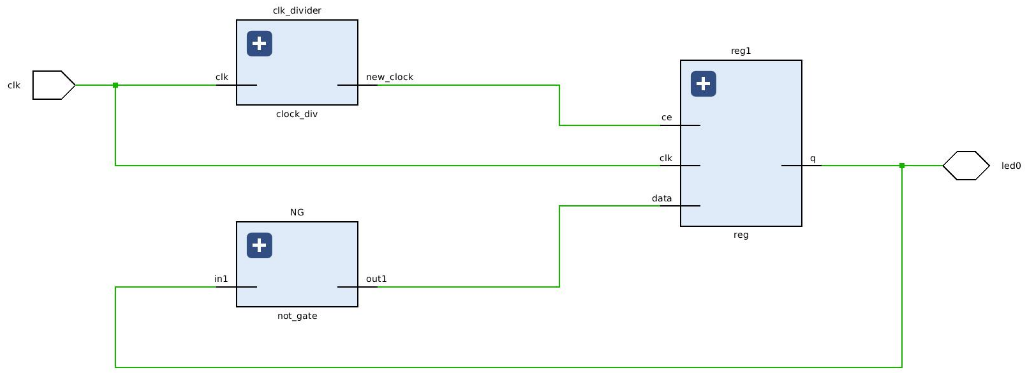


Zoomed in at 500ms you can see the impulse:

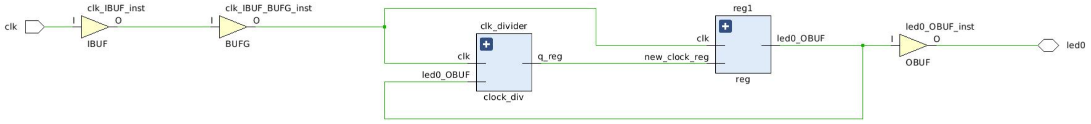


DIVIDER TOP DESIGN:

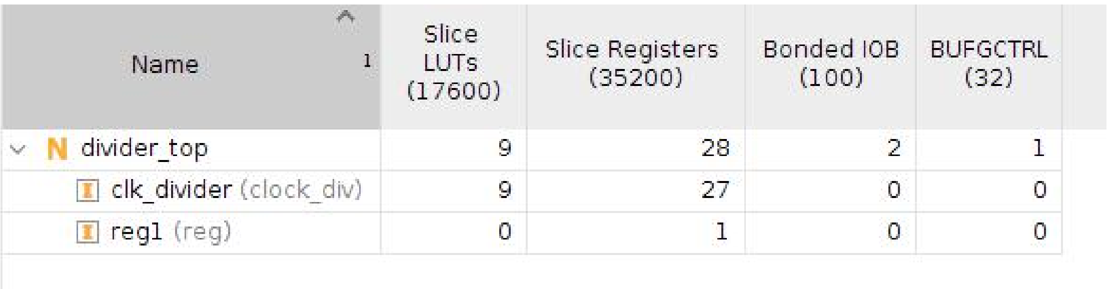
RTL/ELABORATE SCHEMATIC:



SYNTHESIS DESIGN:

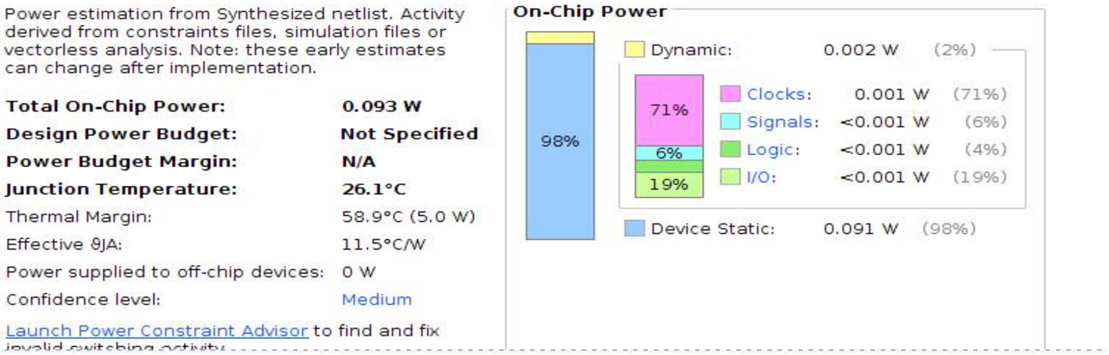


UTILIZATION:



Name	Slice LUTs (17600)	Slice Registers (35200)	Bonded IOB (100)	BUFGCTRL (32)
divider_top	9	28	2	1
clk_divider (clock_div)	9	27	0	0
reg1 (reg)	0	1	0	0

POWER:



Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.093 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:	Medium

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	0.002 W	(2%)
Clocks:	0.001 W	(71%)
Signals:	<0.001 W	(6%)
Logic:	<0.001 W	(4%)
I/O:	<0.001 W	(19%)
Device Static:	0.091 W	(98%)

SOURCE CODE:
TOP DESIGN SOURCE FILE:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity divider_top is
Port (
clk   : in std_logic;
led0 :inout std_logic
  );
end divider_top;


architecture Behavioral of divider_top is

component clock_div
port(
   clk   : in std_logic;
   new_clock : out std_logic
);
end component;
component reg
port(
      clk : in STD_LOGIC;
      ce : in STD_LOGIC;
      data : in STD_LOGIC;
      q : out STD_LOGIC
);
end component;
component not_gate
port(
in1 : in std_logic;
out1 : out std_logic
);
```

```
end component;

signal sig1,sig2 : std_logic;
begin

clk_divider: clock_div
port map(
clk =>clk,
new_clock => sig1
);

NG: not_gate
Port map(
in1 => led0,
out1=> sig2
);

reg1 : reg
port map (
clk => clk,
ce =>sig1,
data => sig2,
q => led0
);


end Behavioral;
```

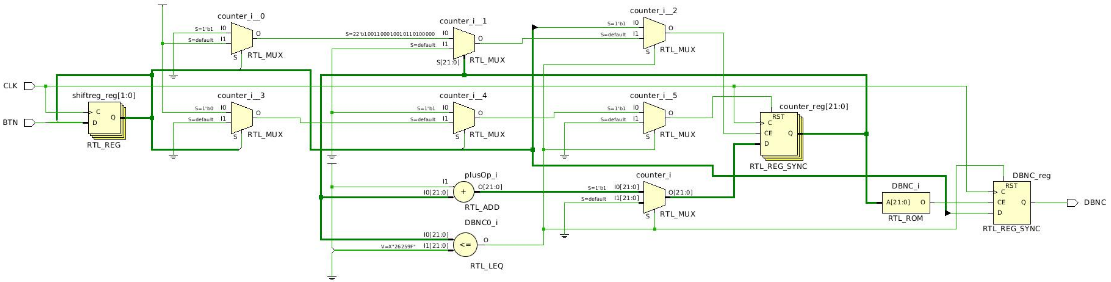
REGISTER SOURCE FILE:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity reg is
    Port ( clk : in STD_LOGIC;
           ce : in STD_LOGIC;
           data : in STD_LOGIC;
           q : out STD_LOGIC);
end reg;
architecture Behavioral of reg is
begin
process(clk)
begin
if (rising_edge(clk)) then
if ce = '1' then
q <= data;
else
q <= '0';
end if;
end if;
end process;
end Behavioral;
```

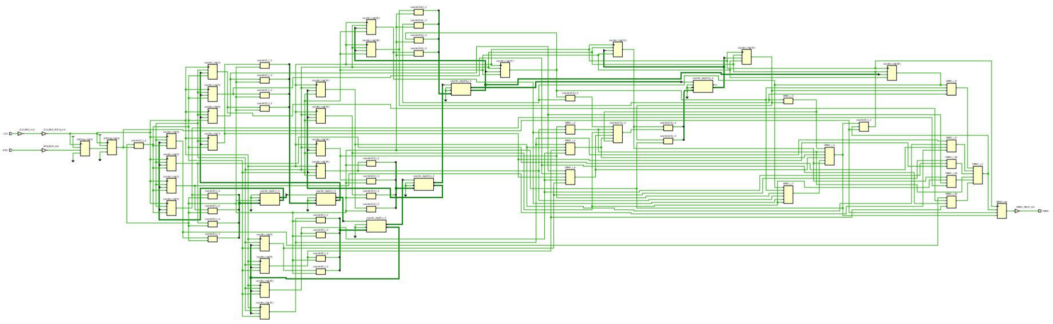
2 BOUNCY BUTTON

We constructed a de-bouncer. Due to the nature of push buttons, since they are mechanical components, the value when pushing the button does not immediately change. The output of a button pressed tends to stagger before it settles at its value. By creating a debouncer we can get rid of the staggering values before it settles. We designed it so that for 20ms of samples, if they are "1"s then that means that the button has settled. If a "0" enters the buttons shiftregister(1) either before the 20ms or even after, the button value goes back to "0". Only once the button has settled, meaning 20ms of time goes by where the value is "1", will the output be "1".

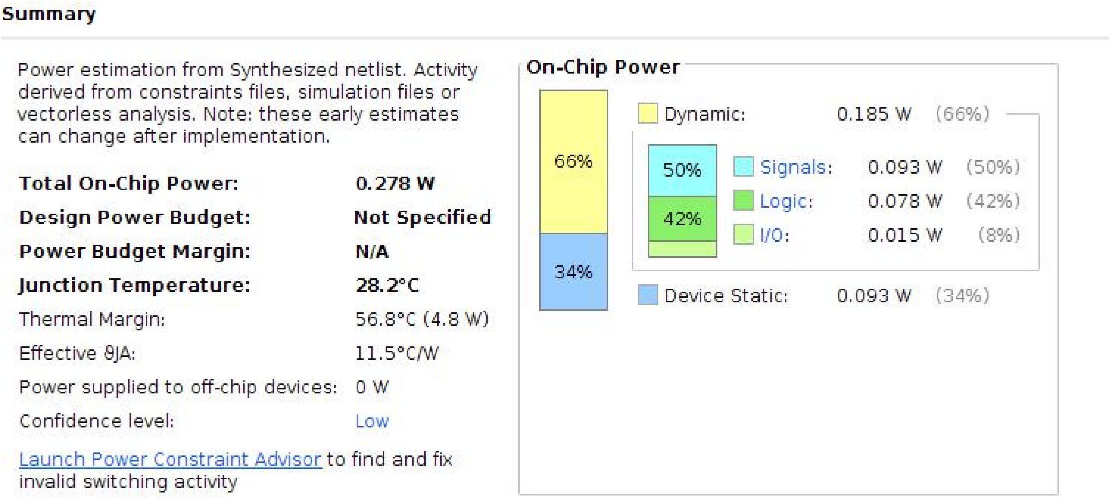
RTL/ELABORATE SCHEMATIC:



SYNTHESIS DEISGN:



POWER:



Summary

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

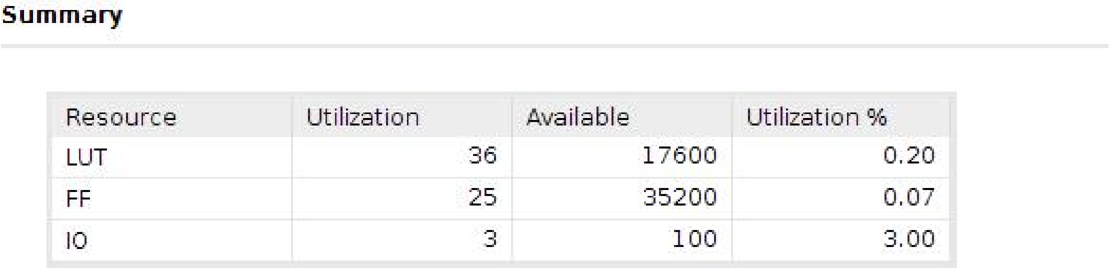
Total On-Chip Power:	**0.278 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**28.2°C**
Thermal Margin:	56.8°C (4.8 W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

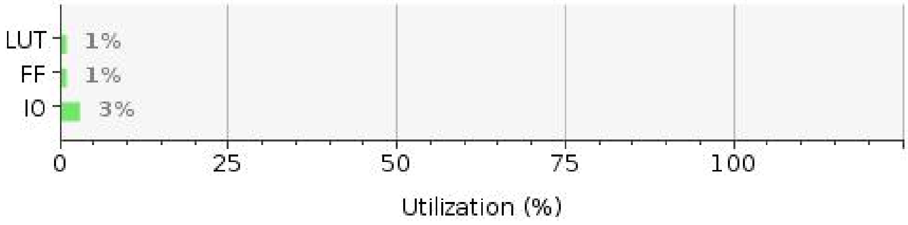
Dynamic:	0.185 W	(66%)
Signals:	0.093 W	(50%)
Logic:	0.078 W	(42%)
I/O:	0.015 W	(8%)
Device Static:	0.093 W	(34%)

UTILIZATION:



Summary

Resource	Utilization	Available	Utilization %
LUT	36	17600	0.20
FF	25	35200	0.07
IO	3	100	3.00



VHDL CODE:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;


entity debounce is
      Port ( BTN : in STD_LOGIC;
             CLK : in STD_LOGIC;
             DBNC : out STD_LOGIC);
end debounce;
```

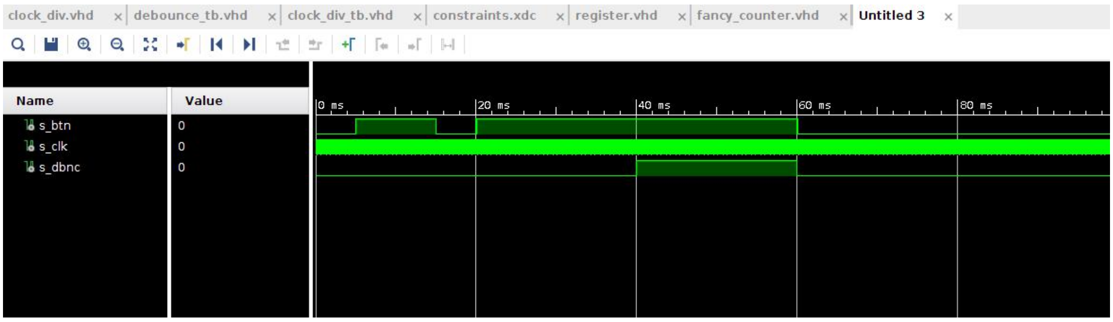
```
architecture Behavioral of debounce is
signal shiftreg : std_logic_vector (1 downto 0) := (others => '0');
signal counter : std_logic_vector(21 downto 0) := (others =>'0');
begin

process (clk)
begin

if (rising_edge(clk)) then
      shiftreg(1) <= shiftreg(0);
      shiftreg(0) <= BTN;

if (unsigned(counter)<= 2499999) then
      DBNC <= '0';
      if shiftreg(1) = '1' then
      counter <= std_logic_vector(unsigned(counter)+1);
      elsif (shiftreg(1)='0') then
      counter <= (others => '0');

      end if;

elsif (unsigned(counter)= 2500000) then
      if (shiftreg(1)='1') then
      DBNC <='1';
      else
      DBNC <='0';
      counter <= (others =>'0');
      end if;

end if;
end if;



end process;

end Behavioral;
```

TESTBENCH:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;



entity debounce_tb is

end debounce_tb;

architecture Behavioral of debounce_tb is
component debounce
port (
BTN : in STD_LOGIC;
CLK : in STD_LOGIC;
DBNC : out STD_LOGIC);
end component;
signal s_btn, s_clk : std_logic :='0';
signal s_dbnc : std_logic;
begin

uut: debounce
Port map(
BTN => s_btn,
CLK => s_clk,
DBNC => s_dbnc
);
 clk_process :process
  begin
 s_clk <= '1';
 wait for 4 ns;
 s_clk <= '0';
 wait for 4ns;
  end process;

stimulus: process
  begin

wait for 5 ms;

 s_btn <= '0';
 wait for 10 ns;
 s_btn <= '1';
 wait for 10 ms;
 s_btn <= '0';
 wait for 5 ms;
 s_btn <= '1';
 wait for 30 ns;
 s_btn <= '0';
 wait for 10 ns;
 s_btn <= '1';
 wait for 40 ns;
 s_btn <= '0';
 wait for 10 ns;
 s_btn <= '1';
 wait for 30 ns;
 s_btn <= '0';
 wait for 10 ns;
 s_btn <= '1';
 wait for 40 ms;
 s_btn <= '0';
 wait for 10 ns;
 s_btn <= '1';
 wait for 20 ns;
 s_btn <= '0';
 wait for 10 ns;
 s_btn <= '1';
 wait for 30 ns;
 s_btn <= '0';
     wait;
  end process;
end Behavioral;
```

SIMULATION:

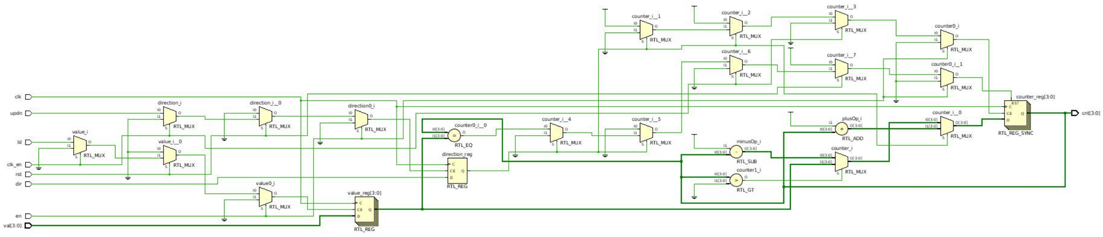
As you can see from the simulation below, At first only a few 1’s are registered but not enough to make the output high. But then it goes “1” for 20ms and then therefore turns on, until the next 0 is registered, and then it goes back off.



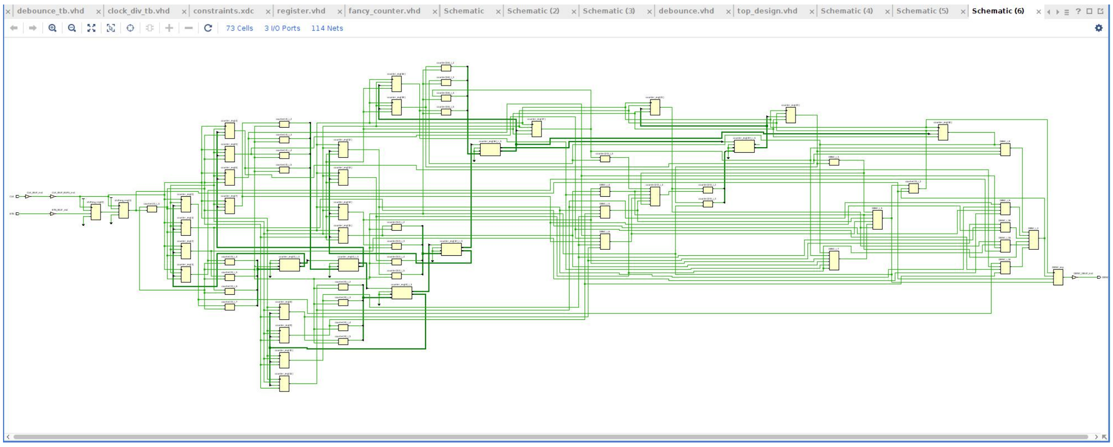
3 ACTUALLY USING A COUNTER TO COUNT

FANCY COUNTER:

RTL/ELABORATE SCHEMATIC:



SYNTHESIS SCHEMATIC:

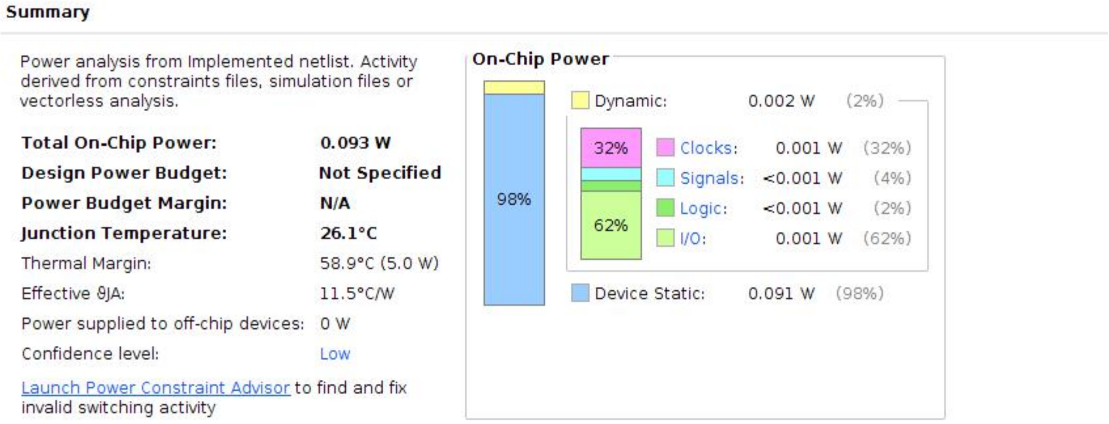


VHDL SOURCE CODE:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.



entity fancy_counter is
    Port ( clk : in STD_LOGIC;
           clk_en : in STD_LOGIC;
           dir : in STD_LOGIC;
           en : in STD_LOGIC;
           ld : in STD_LOGIC;
```

```
          rst : in STD_LOGIC;
          updn : in STD_LOGIC;
          val : in STD_LOGIC_VECTOR (3 downto 0);
          cnt : out STD_LOGIC_VECTOR (3 downto 0));
end fancy_counter;

architecture Behavioral of fancy_counter is

signal direction : std_logic:='0';
signal value : std_logic_vector(3 downto 0);
signal counter : std_logic_vector(3 downto 0):= "0000";
begin


process(clk)
begin

if en ='1' then

 if (rising_edge(clk)) then
       if (rst = '1') then counter <= (others => '0'); else
      if clk_en ='1' then

       if ld ='1' then value <=val; end if;

        if updn ='1' then
        direction <= dir;
        end if;
        if direction = '0' then
            if (counter = value) then counter<= (others => '0');
            else counter <= std_logic_vector((unsigned(counter)+1)); end if;
        end if;
        if direction ='1' then
            if(counter>"0000")thencounter<=
std_logic_vector((unsigned(counter)-1));
            else counter<= value; end if;
        end if;
                end if;
              end if;
         end if;
end if;
end process;
cnt <= counter;
end Behavioral;
```

POWER:



Summary

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

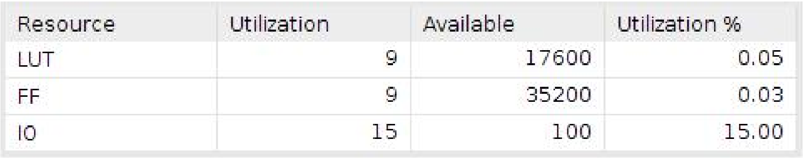
Total On-Chip Power:	**0.093 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**26.1°C**
Thermal Margin:	58.9°C (5.0 W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

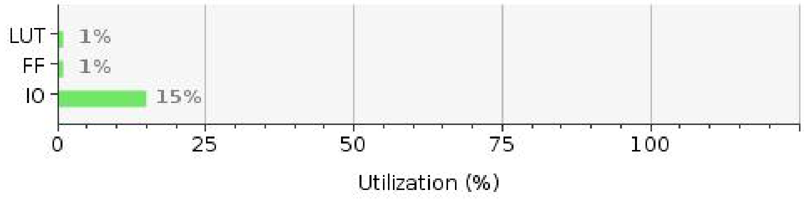
On-Chip Power

Dynamic:	0.002 W	(2%)
Clocks:	0.001 W	(32%)
Signals:	<0.001 W	(4%)
Logic:	<0.001 W	(2%)
I/O:	0.001 W	(62%)
Device Static:	0.091 W	(98%)

UTILIZATION:



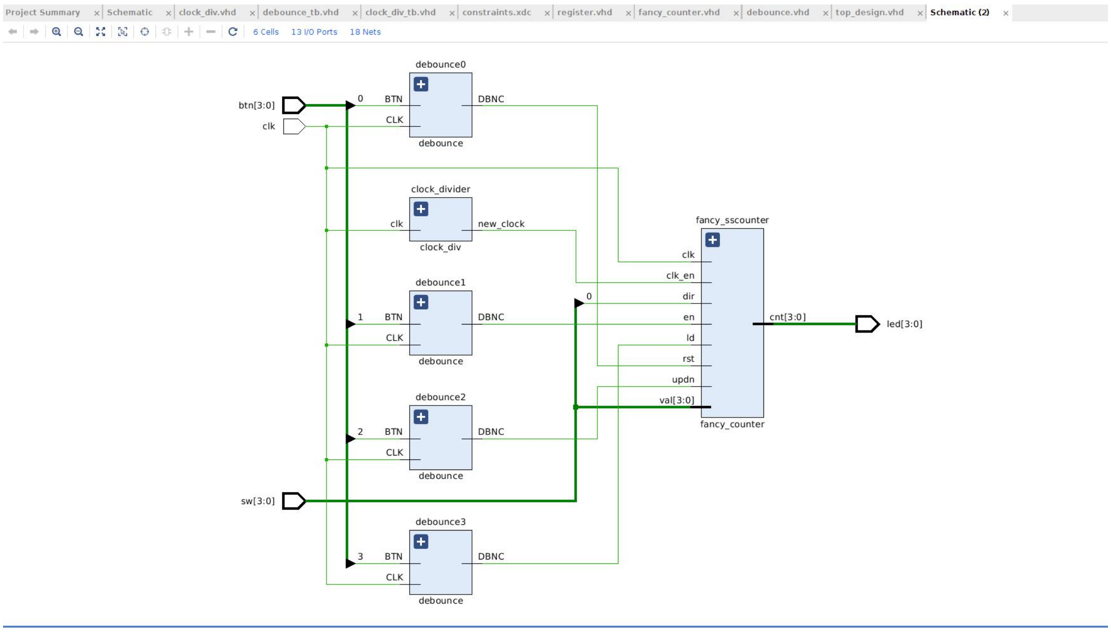
Resource	Utilization	Available	Utilization %
LUT	9	17600	0.05
FF	9	35200	0.03
IO	15	100	15.00



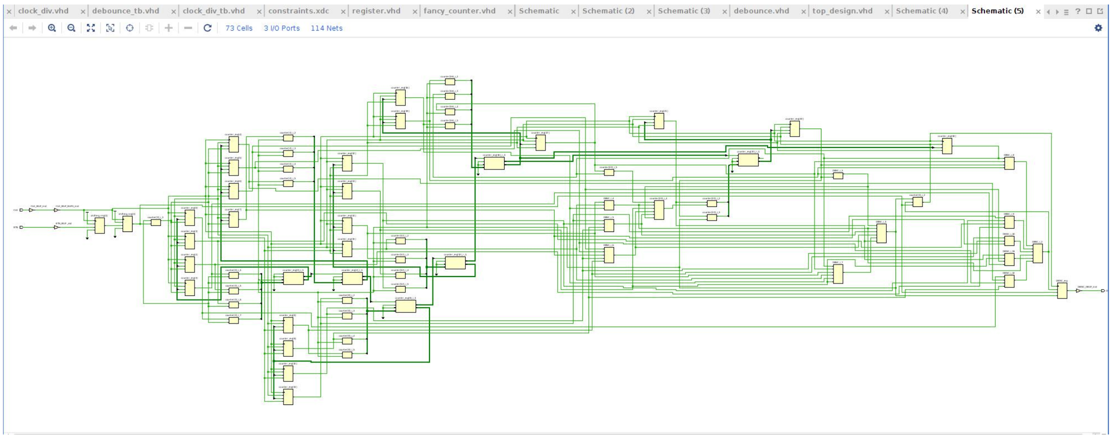
4 BRINGING IT ALL TOGETHER

Now we can create a top level design that incorporates all of the modules that we created. By creating a top design source file and port mapping the components, we can assemble the circuit shown below. The debouncer's are used between the button and the inputs to the fancy_counter to prevent erratic values at the inputs. The clock divider will control the frequency at which the counter actually counts.

RTL/ELABORATE SCHEMATIC:



SYNTHESIS_SCHEMATIC:



VHDL SOURCE CODE:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity top_design is
    Port ( btn : in STD_LOGIC_VECTOR (3 downto 0);
           clk : in STD_LOGIC;
           sw : in STD_LOGIC_vector(3 downto 0);
           led : out STD_LOGIC_VECTOR (3 downto 0));
end top_design;

architecture Behavioral of top_design is
component fancy_counter
port(
clk : in STD_LOGIC;
clk_en : in std_logic;
dir : in STD_LOGIC;
en : in STD_LOGIC;
ld : in STD_LOGIC;
rst : in STD_LOGIC;
updn : in STD_LOGIC;
val : in STD_LOGIC_VECTOR (3 downto 0);
cnt : out STD_LOGIC_VECTOR (3 downto 0)
);
end component;

component debounce
port(
BTN : in STD_LOGIC;
CLK : in STD_LOGIC;
DBNC : out STD_LOGIC
);
end component;
component clock_div
port(
clk   : in std_logic;
new_clock : out std_logic
);
end component;
signal s_rst, s_en, s_updn, s_ld, s_clk_en, s_dir : std_logic;

begin
```

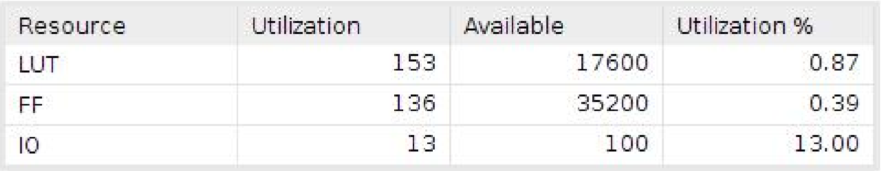
```
debounce0: debounce
port map(
BTN => btn(0),
clk => clk,
dbnc => s_rst
);
debounce1: debounce
port map(
BTN => btn(1),
clk => clk,
dbnc => s_en
);
debounce2: debounce
port map(
BTN => btn(2),
clk => clk,
dbnc => s_updn
);
debounce3: debounce
port map(
BTN => btn(3),
clk => clk,
dbnc => s_ld
);
clock_divider: clock_div
Port map(
clk => clk,
new_clock => s_clk_en
);
fancy_sscounter: fancy_counter
port map(
clk => clk,
clk_en => s_clk_en,
dir => sw(0),
en => s_en,
ld => s_ld,
rst => s_rst,
updn =>s_updn,
val => sw,
cnt => led

);

end Behavioral;
```

CONTRAINTS: The clk is connected to the L16 pin for 125mHz board clock. The output of counter is connected to the 4 leds, M14, M15, G14, D18. The buttons are connected to the 4 btn bits, R18, P16, V16, Y16. The switches are connected to the sw 4 bits to pins, G15, P15, W13, T16.

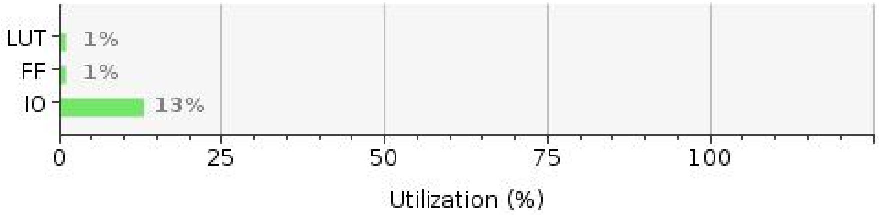
CONSTRAINTS FILE:

```
set_property -dict { PACKAGE_PIN L16     IOSTANDARD LVCMOS33 } [get_ports
{ clk }]; #IO_L11P_T1_SRCC_35 Sch=sysclk
create_clock -add -name sys_clk_pin -period 8.00 -waveform {0 4} [get_ports
{ clk }];


##Switches
set_property -dict { PACKAGE_PIN G15     IOSTANDARD LVCMOS33 } [get_ports
{ sw[0] }]; #IO_L19N_T3_VREF_35 Sch=SW0
set_property -dict { PACKAGE_PIN P15     IOSTANDARD LVCMOS33 } [get_ports
{ sw[1] }];   #IO_L24P_T3_34 Sch=SW1
set_property -dict { PACKAGE_PIN W13     IOSTANDARD LVCMOS33 } [get_ports
{ sw[2] }]; #IO_L4N_T0_34 Sch=SW2
set_property -dict { PACKAGE_PIN T16     IOSTANDARD LVCMOS33 } [get_ports
{ sw[3] }]; #IO_L9P_T1_DQS_34 Sch=SW3


##LEDs
set_property -dict { PACKAGE_PIN M14     IOSTANDARD LVCMOS33 } [get_ports
{ led[0] }]; #IO_L23P_T3_35 Sch=LED0
set_property -dict { PACKAGE_PIN M15     IOSTANDARD LVCMOS33 } [get_ports
{ led[1] }]; #IO_L23N_T3_35 Sch=LED1
set_property -dict { PACKAGE_PIN G14     IOSTANDARD LVCMOS33 } [get_ports
{ led[2] }]; #IO_0_35=Sch=LED2
set_property -dict { PACKAGE_PIN D18     IOSTANDARD LVCMOS33 } [get_ports
{ led[3] }]; #IO_L3N_T0_DQS_AD1N_35 Sch=LED3
set_property -dict { PACKAGE_PIN R18     IOSTANDARD LVCMOS33 } [get_ports
{ btn[0] }];
set_property -dict { PACKAGE_PIN p16     IOSTANDARD LVCMOS33 } [get_ports
{ btn[1] }];
set_property -dict { PACKAGE_PIN V16     IOSTANDARD LVCMOS33 } [get_ports
{ btn[2] }];
set_property -dict { PACKAGE_PIN Y16     IOSTANDARD LVCMOS33 } [get_ports
{ btn[3] }];
```

UTILIZATION:

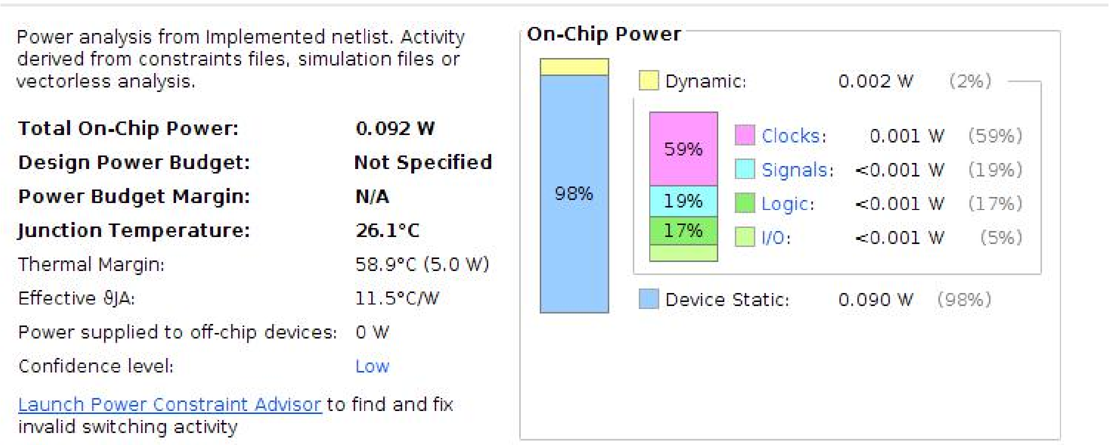


Resource	Utilization	Available	Utilization %
LUT	153	17600	0.87
FF	136	35200	0.39
IO	13	100	13.00



POWER:

Summary



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

Total On-Chip Power:	**0.092 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:	0.002 W	(2%)
🟪 Clocks:	0.001 W	(59%)
🟦 Signals:	<0.001 W	(19%)
🟩 Logic:	<0.001 W	(17%)
🟩 I/O:	<0.001 W	(5%)
🟦 Device Static:	0.090 W	(98%)

QUESTIONS:

Q1.1

To divide our input of 125Mhz to 2Hz we divide by 62.5x 10^6.

Q1.2

62.5x10^6 = 2^x +1

X = 25.89 or 26

The number of bits required is 26.

Q2.1

When the button is pressed on the zybo board, the value goes from 0 to 1, but not exactly. Since the push button is a mechanical device the value will stagger until it stabilizes. So if the current value unpressed is 0, then when the button is pressed it might do something like 0,1,0,1,1,0,1,1,1,1...... Once the button has settled it will stay “1”.

Q2.2 (Optional)

I believe you would not have to change the design of the debouncer. The only difference is once a zero is registered it immediately goes to zero. If you were to reverse the design, you could make it so it immediatly goes to 1 when a 1 is registered and only goes zero when a certain number of “0”s have been registered.

Q2.3

We would have to count up to do 125*10^6 * 20ms = **2.5 * 10^6 samples**

Q2.4

2.5 *10^6 = 2^x -1

X = 21.25

# of bits required = 22 bits

Observations/Discoveries:

The rtl schematic design is much more helpful than I originally thought. When the fancy_counter was being designed, there were issues with it running correctly. Using the schematic I was able to follow multiplexers,registers, comparators and other components and actually find the mistakes. It helped me fix the errors and provided greater insight into my design and how it is actually being made.

The power and utilization reports were also insightful. The utilization report shows the resources that are being used in the fpga and how many are available to use. This can be helpful when you need to reduce complexity or simplify design due to resource limits. In this lab this was not the case, we did not utilize much.

Concepts unsure of:

Would like to better understand the synthesized design that actually shows the resources being used on the chip.