ECE493: Embedded Systems I
Lab Report 1
Timothy Petersen - trp87
2/27/2020

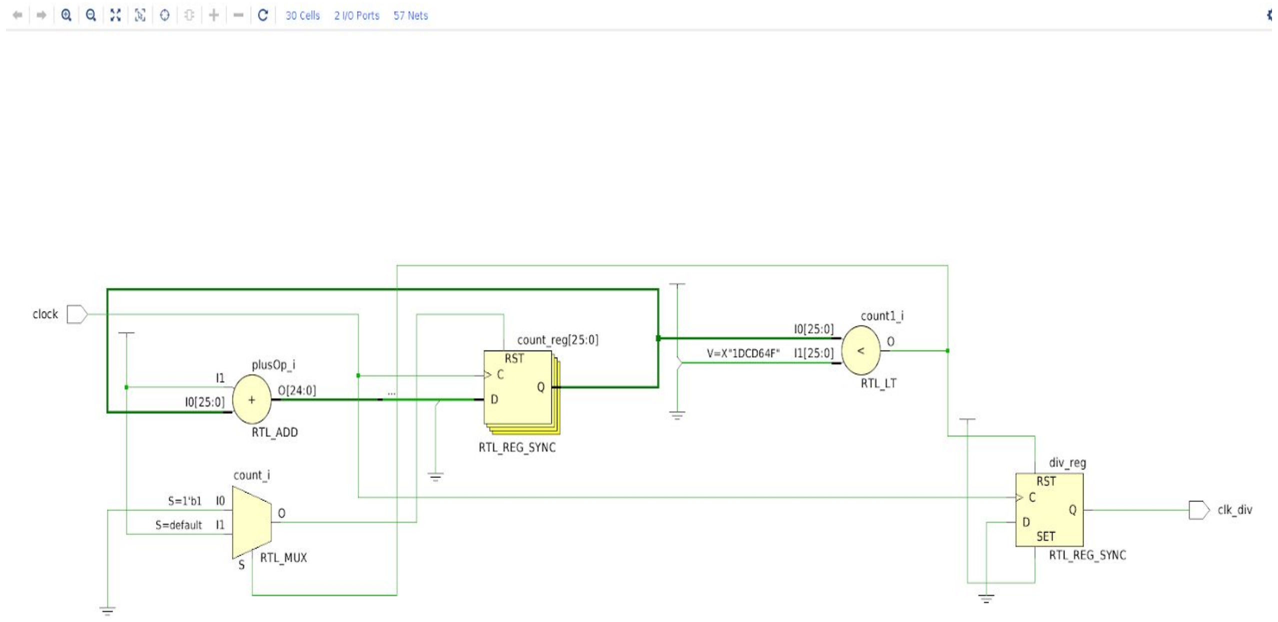
Purpose

In this experiment, we will formally introduce one of the core building blocks of digital circuits: the humble counter. By exploiting the versatility of the counter, a high frequency input clock will be slowed down by several orders of magnitude and then used to drive a bidirectional counter. Additionally, a counter will be used to solve one of the core problems involved with getting manual user input in digital circuits.

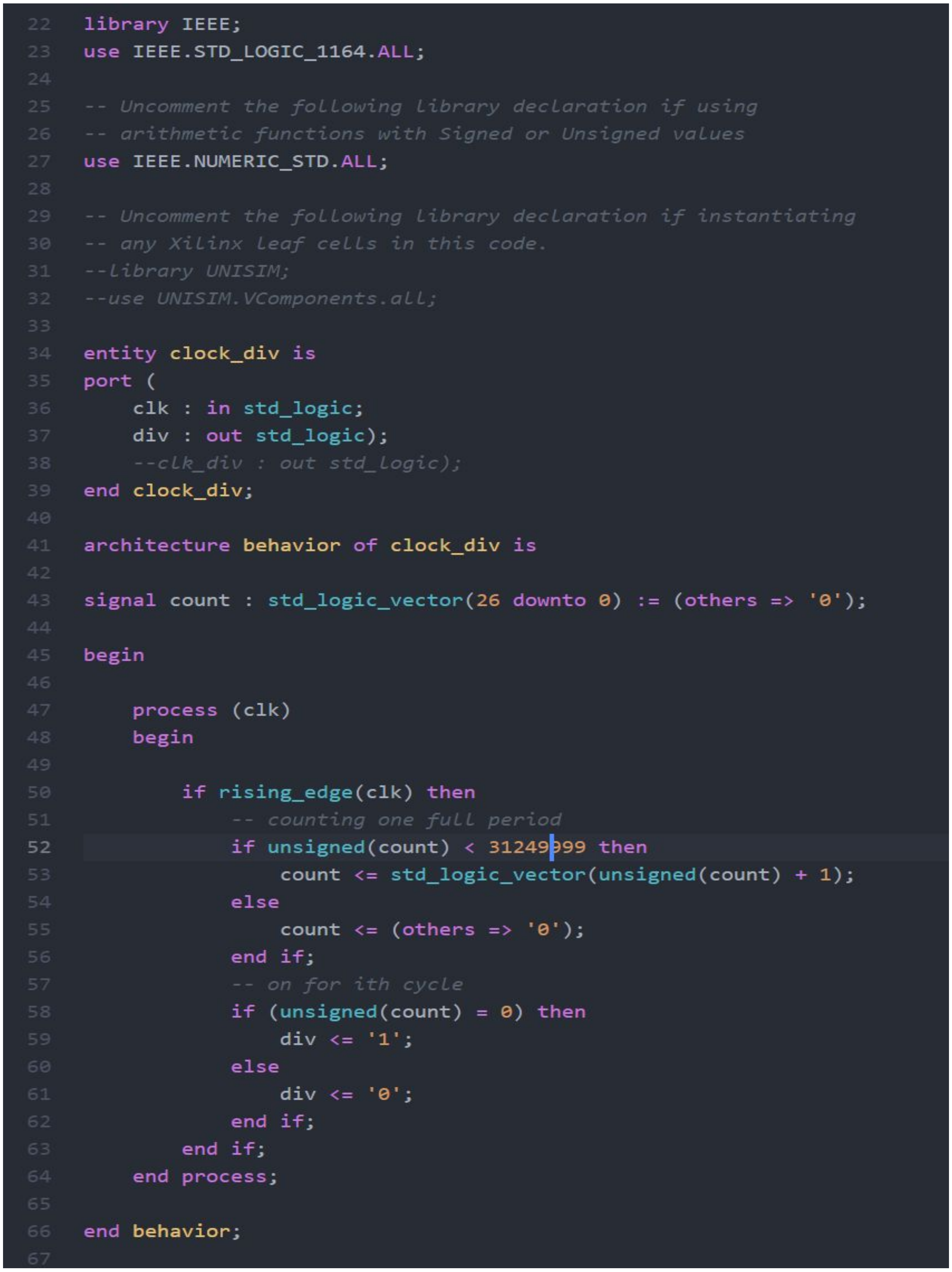
Clock_Div

Clock Div, operates as a clock divider, it takes a 125MHz Clock as an input and converts it to a pulse the width of one clock cycle at a rate of 2Hz. To do this, a counter was implemented. The counter counted up to ¼ of 125 million (31249999). When the counter is reached the output will turn to 1 until the next rising edge.

RTL Schematic:

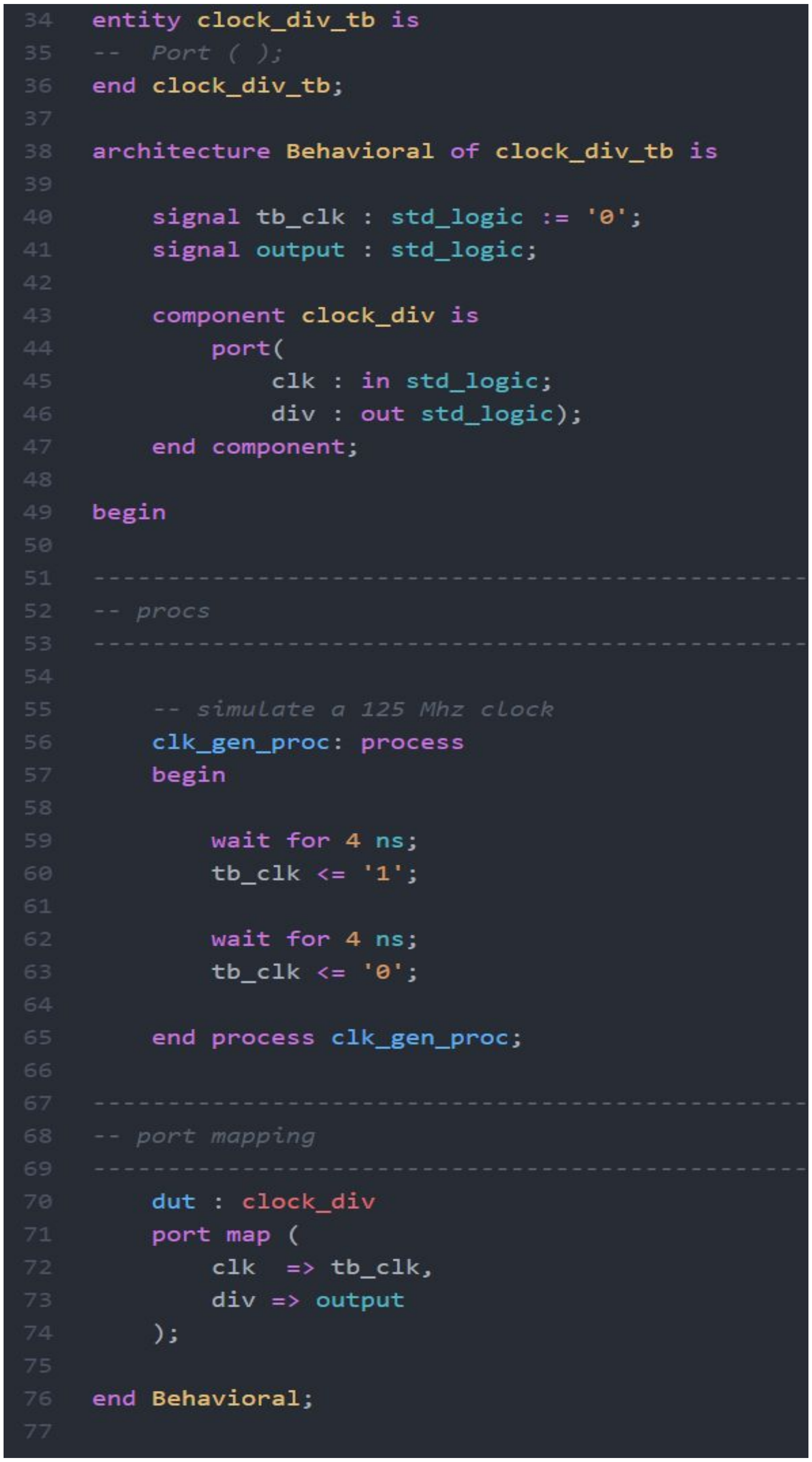


Code (on Github):



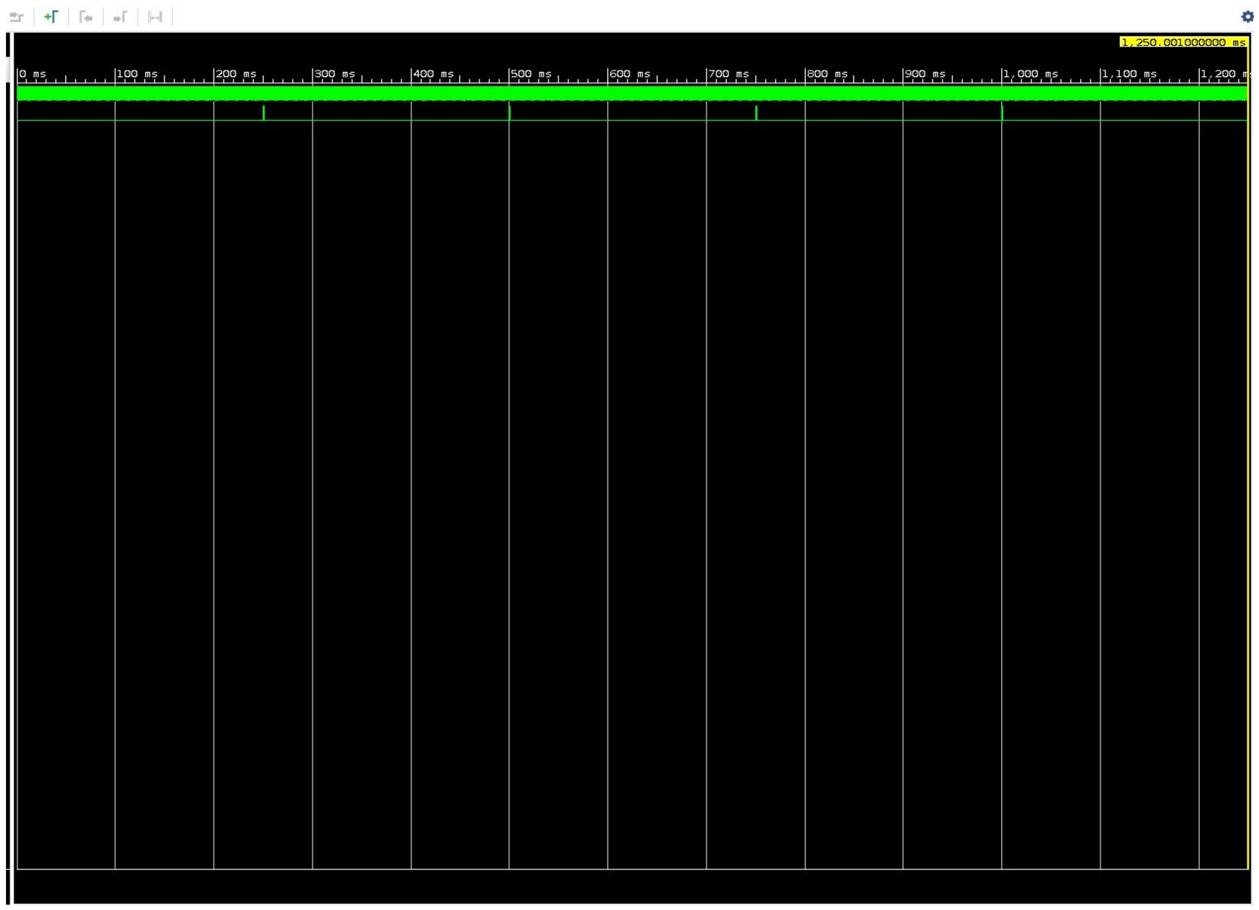
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

architecture behavior of clock_div is

signal count : std_logic_vector(26 downto 0) := (others => '0');

begin

    process (clk)
    begin

        if rising_edge(clk) then
            -- counting one full period
            if unsigned(count) < 31249999 then
                count <= std_logic_vector(unsigned(count) + 1);
            else
                count <= (others => '0');
            end if;
            -- on for ith cycle
            if (unsigned(count) = 0) then
                div <= '1';
            else
                div <= '0';
            end if;
        end if;
    end process;

end behavior;
```

Test bench code:

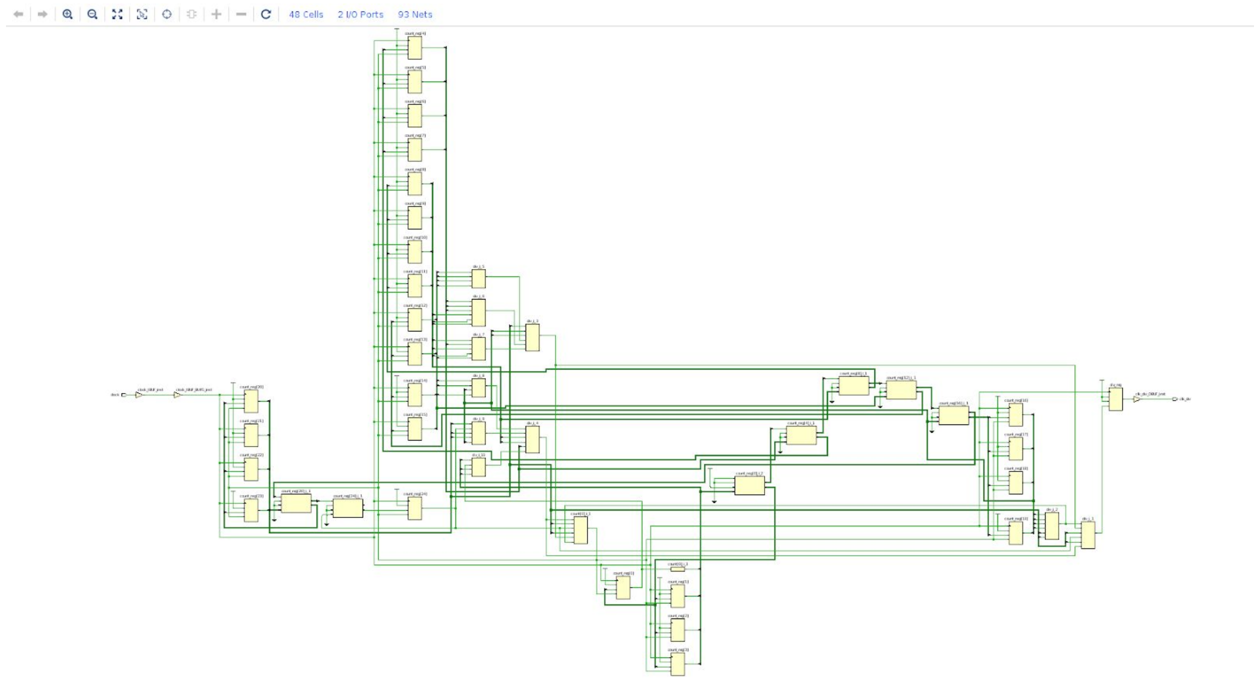


```vhdl
entity clock_div_tb is
--  Port ( );
end clock_div_tb;

architecture Behavioral of clock_div_tb is

    signal tb_clk : std_logic := '0';
    signal output : std_logic;

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

begin

------------------------------------------------
-- procs
------------------------------------------------

    -- simulate a 125 Mhz clock
    clk_gen_proc: process
    begin

        wait for 4 ns;
        tb_clk <= '1';

        wait for 4 ns;
        tb_clk <= '0';

    end process clk_gen_proc;

------------------------------------------------
-- port mapping
------------------------------------------------
    dut : clock_div
    port map (
        clk  => tb_clk,
        div => output
    );

end Behavioral;
```

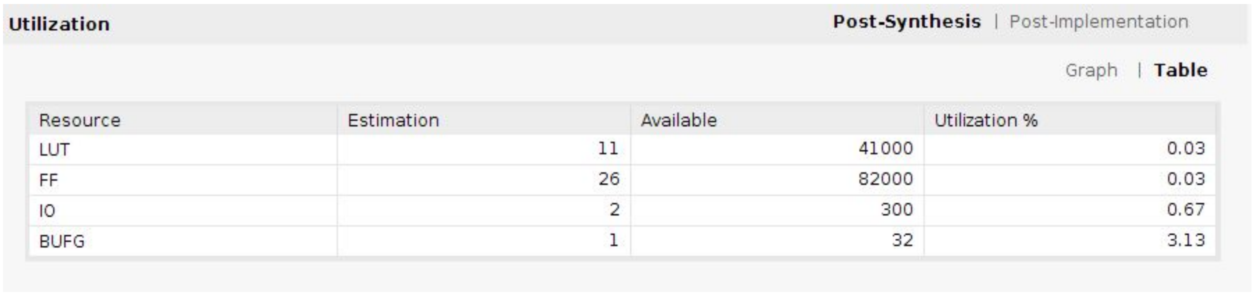
Test Bench Simulation Waves:



Synthesis Schematic:



Post Synthesis Utilization Table

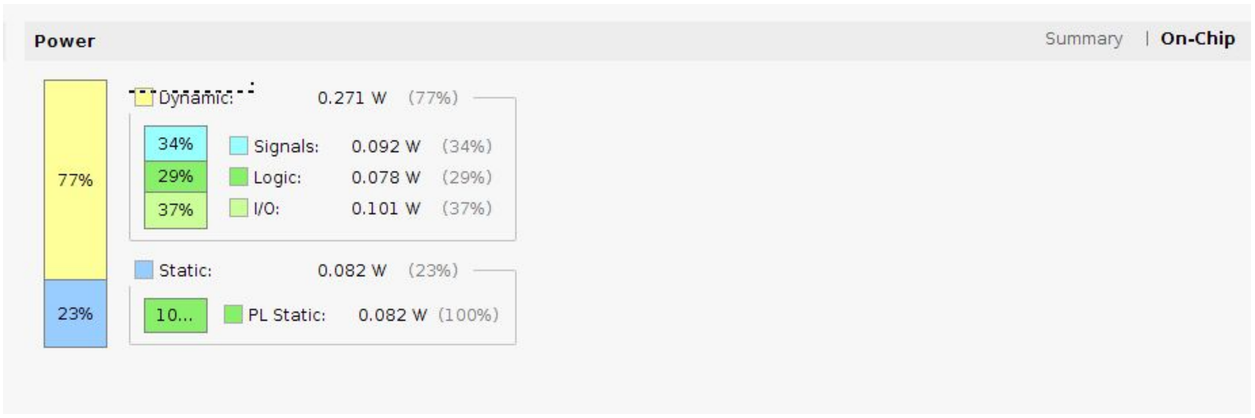


Utilization — **Post-Synthesis** | Post-Implementation

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	11	41000	0.03
FF	26	82000	0.03
IO	2	300	0.67
BUFG	1	32	3.13

On-Chip Power Graph



Power — Summary | **On-Chip**

Dynamic:		0.271 W	(77%)
34%	Signals:	0.092 W	(34%)
29%	Logic:	0.078 W	(29%)
37%	I/O:	0.101 W	(37%)
Static:		0.082 W	(23%)
10...	PL Static:	0.082 W	(100%)

XDC File Changes

Led0 was enabled by uncommenting it, along with the clock.

Questions

1. In order to divide the clock, the counter must count up to 31,249,999.
2. 25 bits are needed to count up to 31,249,999.

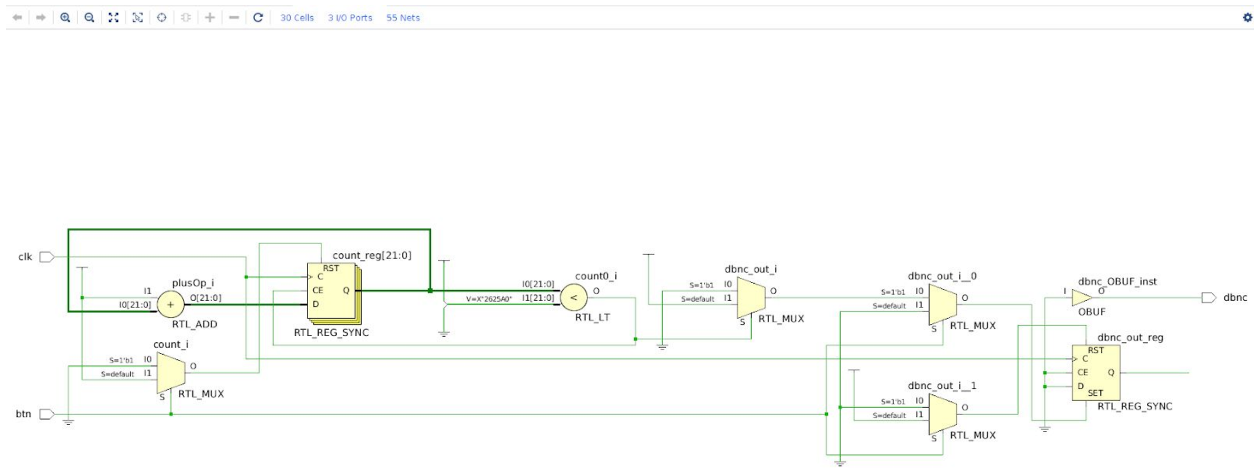
Reflection

During the design of this component I learned how to divide a clock, and change the duty cycle of the clock. Previously we had a 50% duty cycle in the previous lab, and this one had to be switched to be on for on pulse.

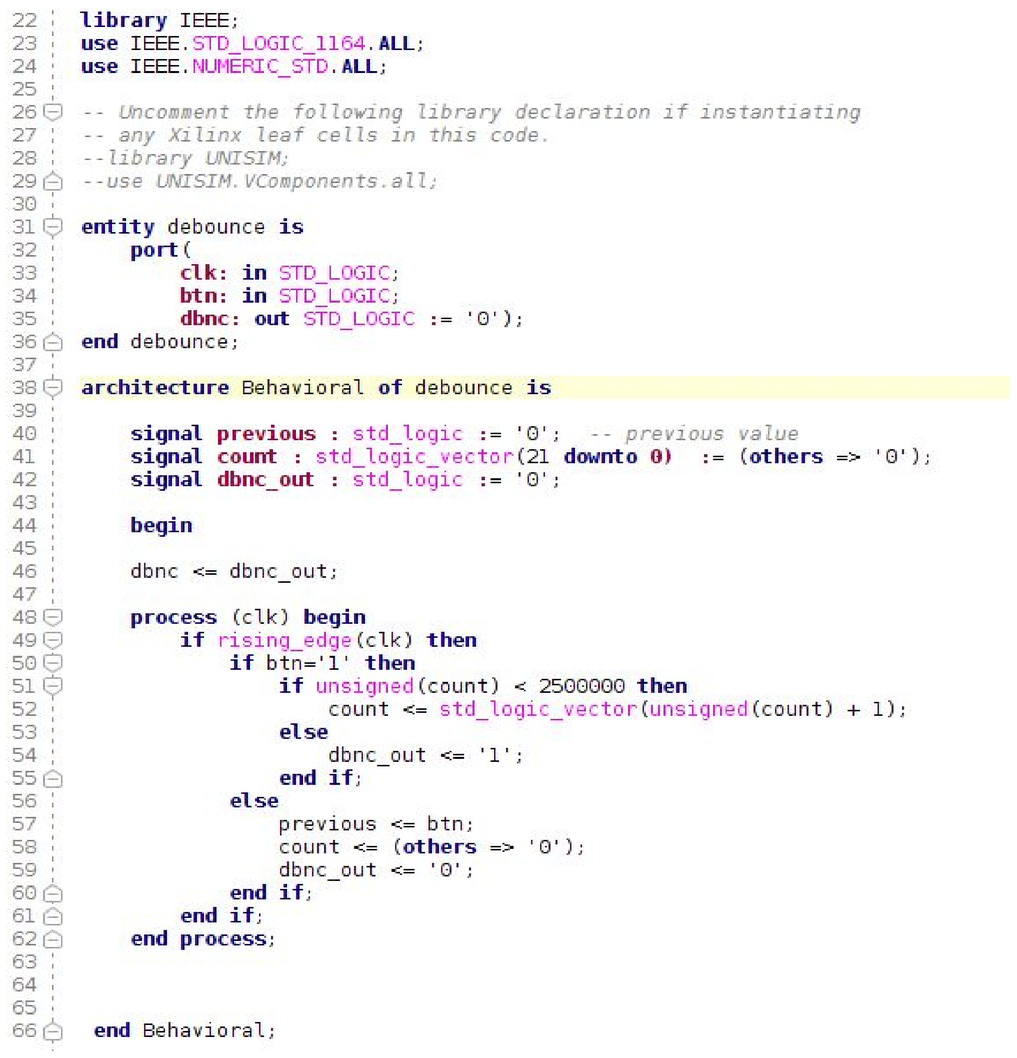
Bouncy Buttons

The purpose of this component is to debounce buttons. Mechanical buttons alter in value between 0 and 1 when pressed, so the purpose of this program is to delay the output of the button until it is sure that the button is 1 or 0 and not bouncing in between. To do this, a shift register is used

RTL Schematic:

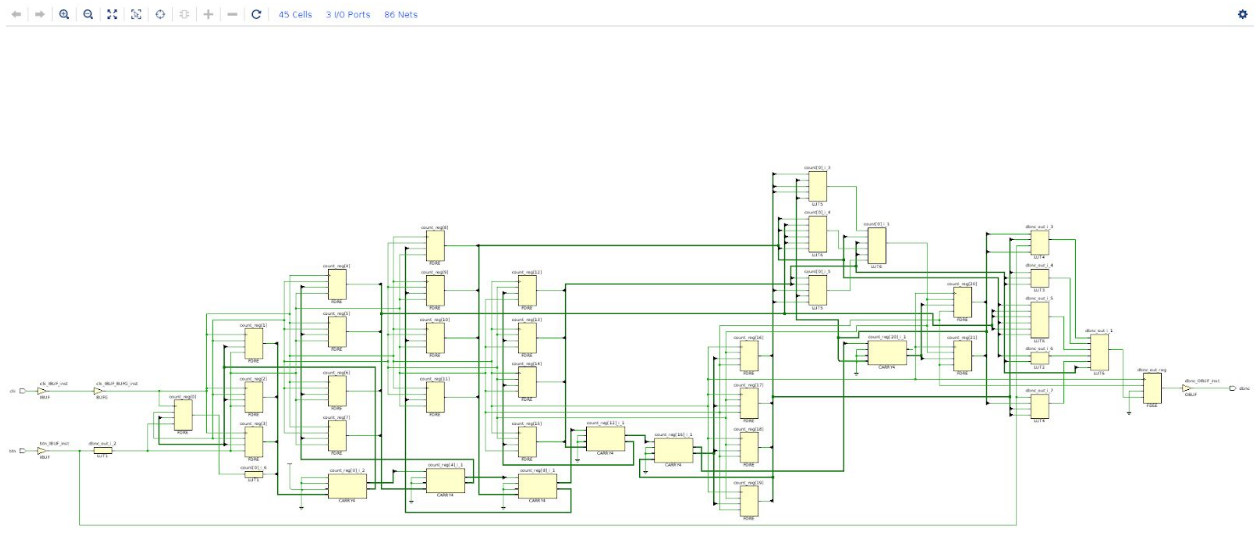


Code:

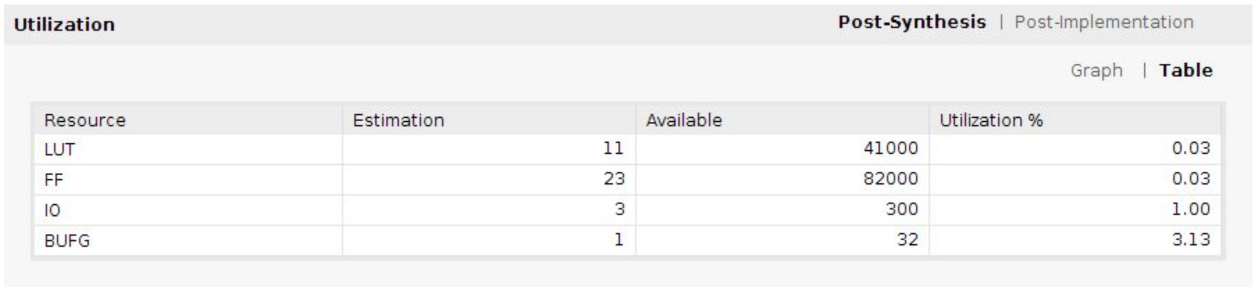


```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

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

entity debounce is
    port(
        clk: in STD_LOGIC;
        btn: in STD_LOGIC;
        dbnc: out STD_LOGIC := '0');
end debounce;

architecture Behavioral of debounce is

    signal previous : std_logic := '0';   -- previous value
    signal count : std_logic_vector(21 downto 0)   := (others => '0');
    signal dbnc_out : std_logic := '0';

    begin

    dbnc <= dbnc_out;

    process (clk) begin
        if rising_edge(clk) then
            if btn='1' then
                if unsigned(count) < 2500000 then
                    count <= std_logic_vector(unsigned(count) + 1);
                else
                    dbnc_out <= '1';
                end if;
            else
                previous <= btn;
                count <= (others => '0');
                dbnc_out <= '0';
            end if;
        end if;
    end process;



 end Behavioral;
```

Synthesis Schematic:



Post Synthesis Utilization Table:

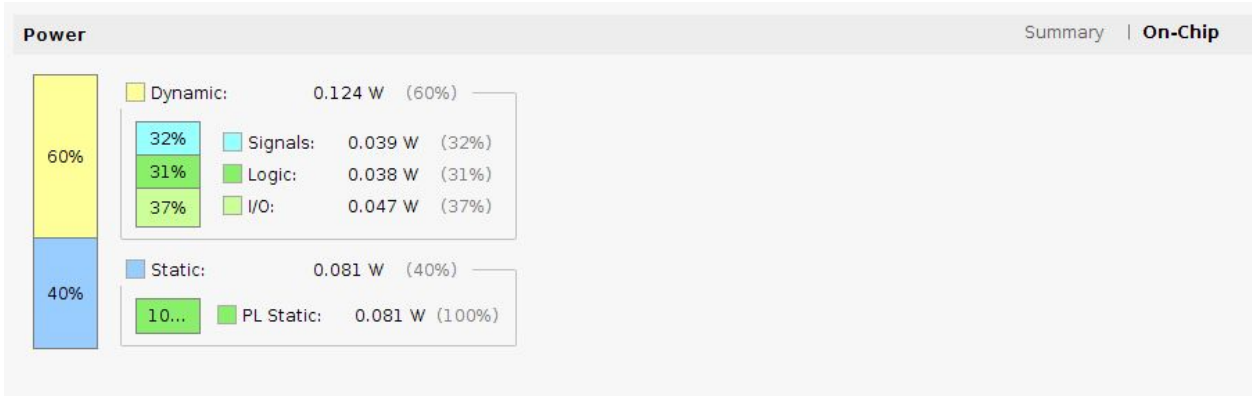


Utilization — **Post-Synthesis** | Post-Implementation

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	11	41000	0.03
FF	23	82000	0.03
IO	3	300	1.00
BUFG	1	32	3.13

On-Chip Power Graphs



Power — Summary | **On-Chip**

60%	Dynamic:	0.124 W	(60%)
32%	Signals:	0.039 W	(32%)
31%	Logic:	0.038 W	(31%)
37%	I/O:	0.047 W	(37%)
40%	Static:	0.081 W	(40%)
10...	PL Static:	0.081 W	(100%)

XDC Changes:

All buttons were uncommented.

Questions:

1. When the Zybo button is pressed it is the value of 1.

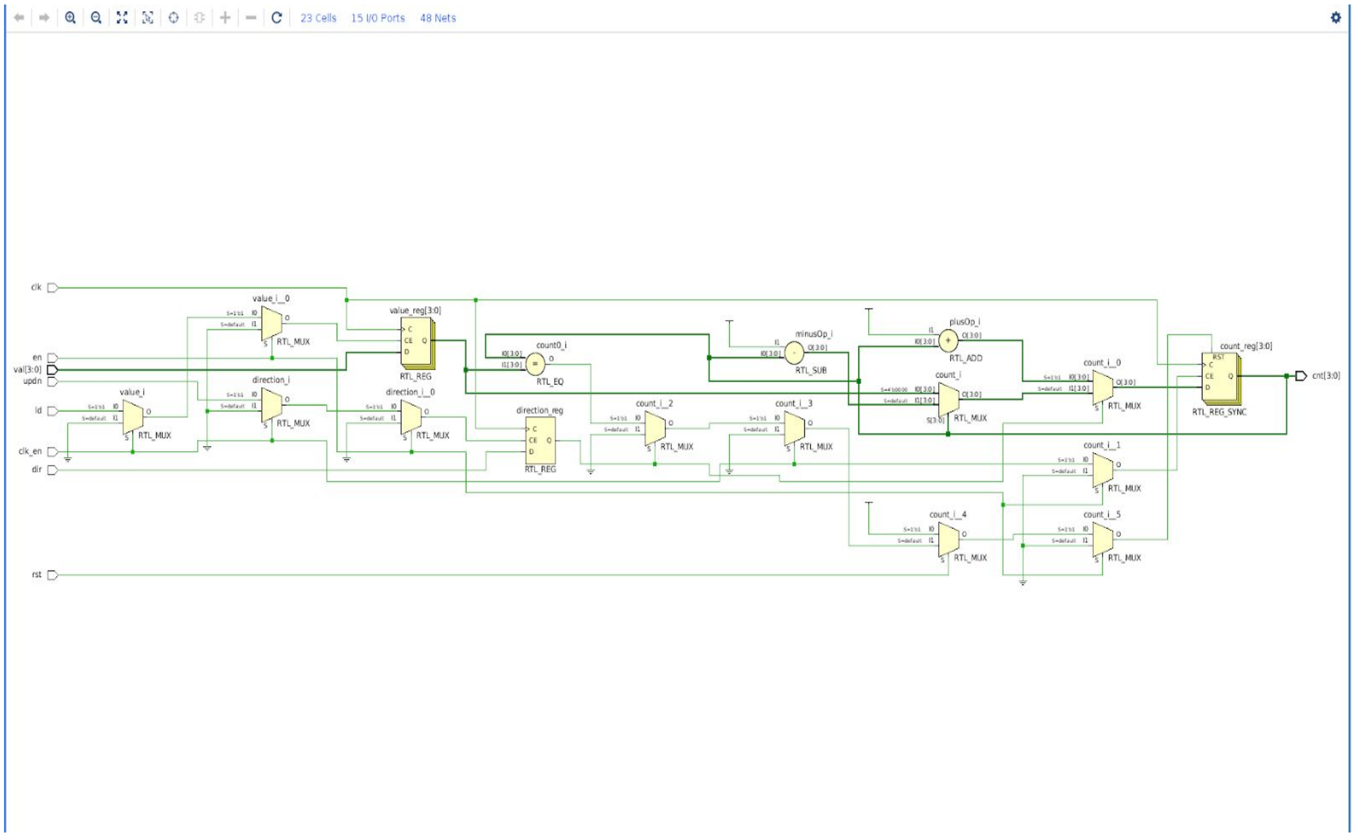
2. Optional
3. The button will need to be steady for 5 million ticks in order to be steady for 20ms based on a 125MHz clock.
4. The length of the counter was 23 bits, in order to count to 5 million.

Fancy Counter

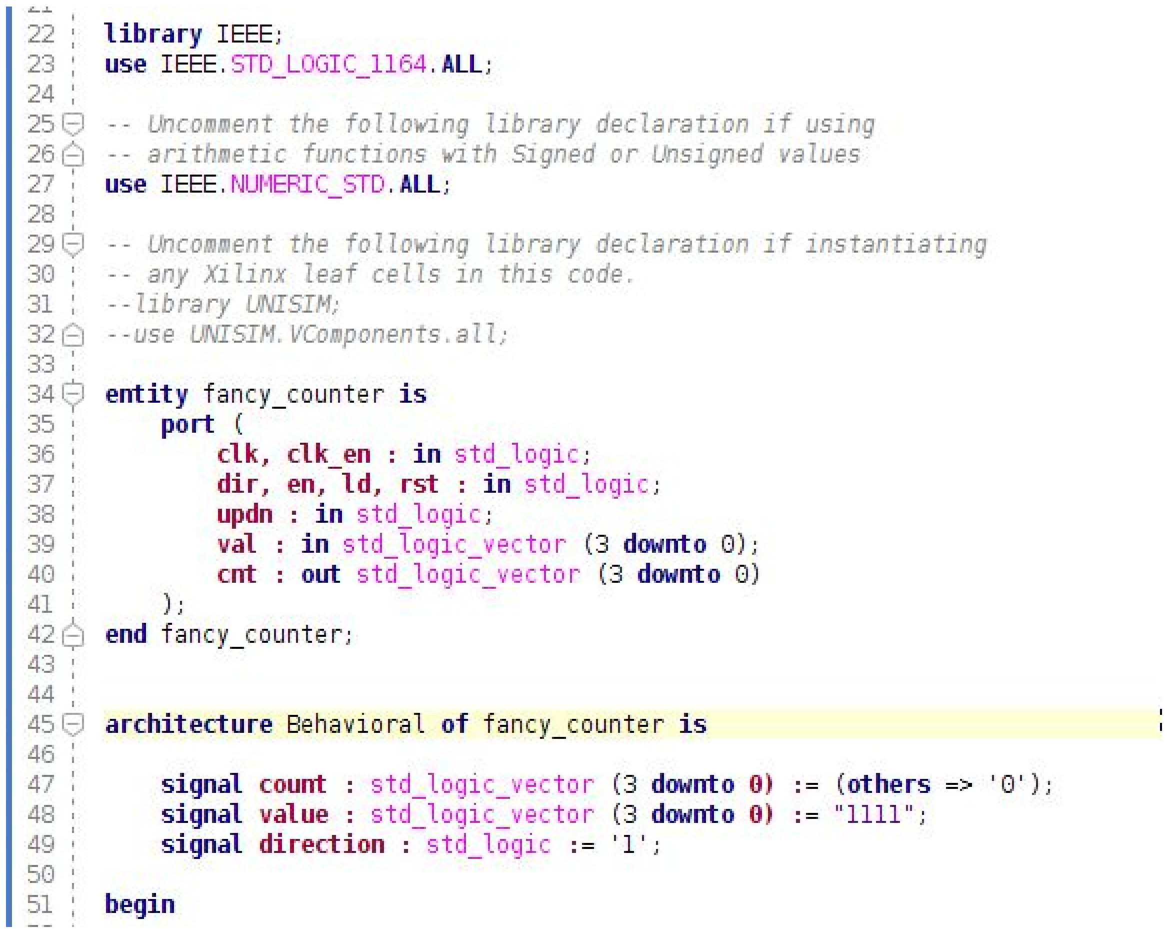
The task for this component was to create a bidirectional counter with a few extra signals. The following logic was implemented.

- Unless en is 1, nothing will change in the circuit.
- Even if en is 1, if clk en is 0 nothing can change the circuit except rst
- On the clock rising edge, when rst is asserted the cnt value will become 0.
- It can count either up or down depending on the value of a “direction” register, which is updated at the clock rising edge with the value present at dir when updn is 1.
- On the clock rising edge, if ld is 1, the value present at val will be loaded into the “value” register.
- If counting up, it will count until the number in a 4-bit “value” register has been reached, at which point it will roll over to 0000. If counting down, it will go from 0000 to value when it underflows.

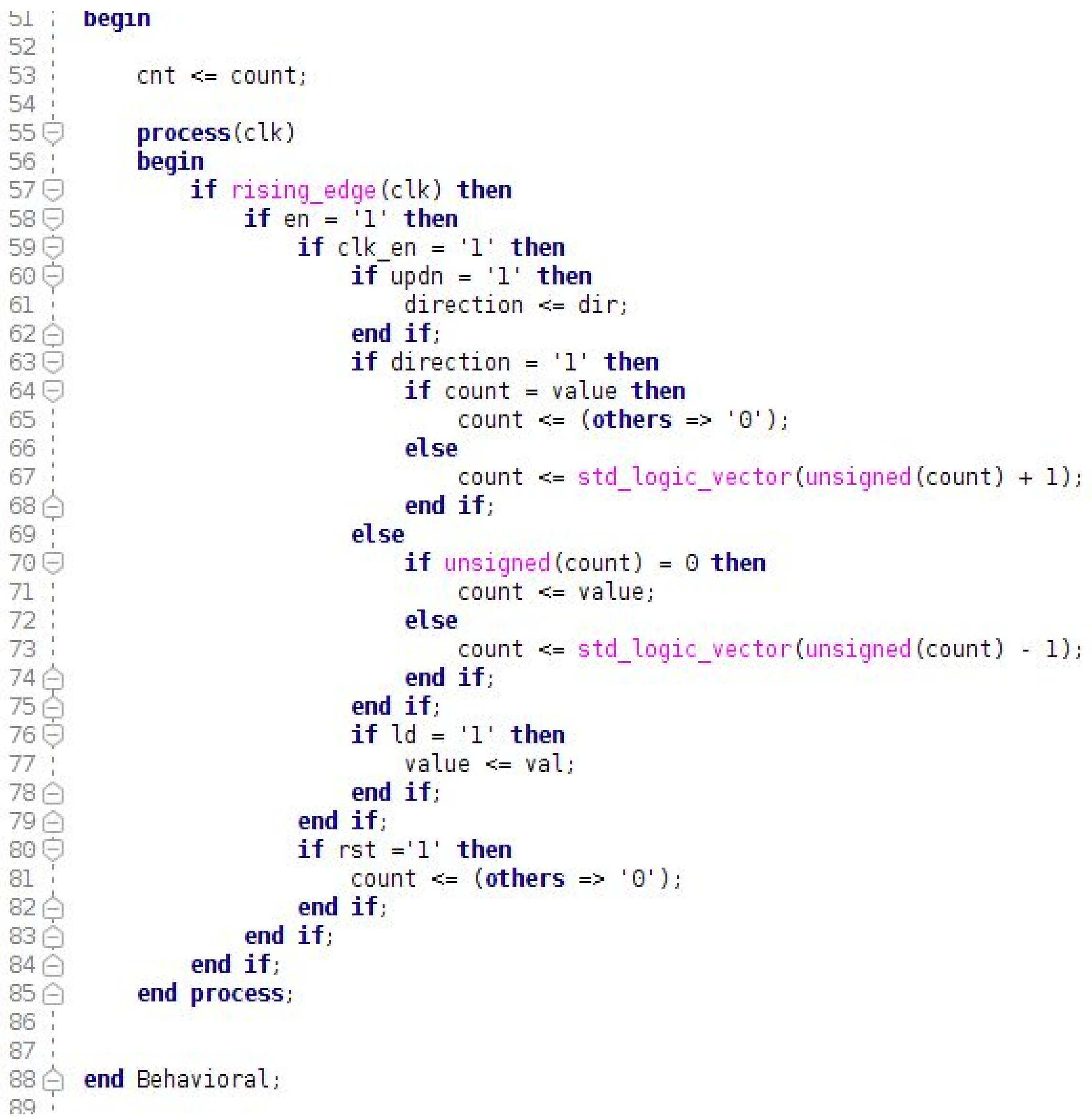
RTL Schematic:



Code:

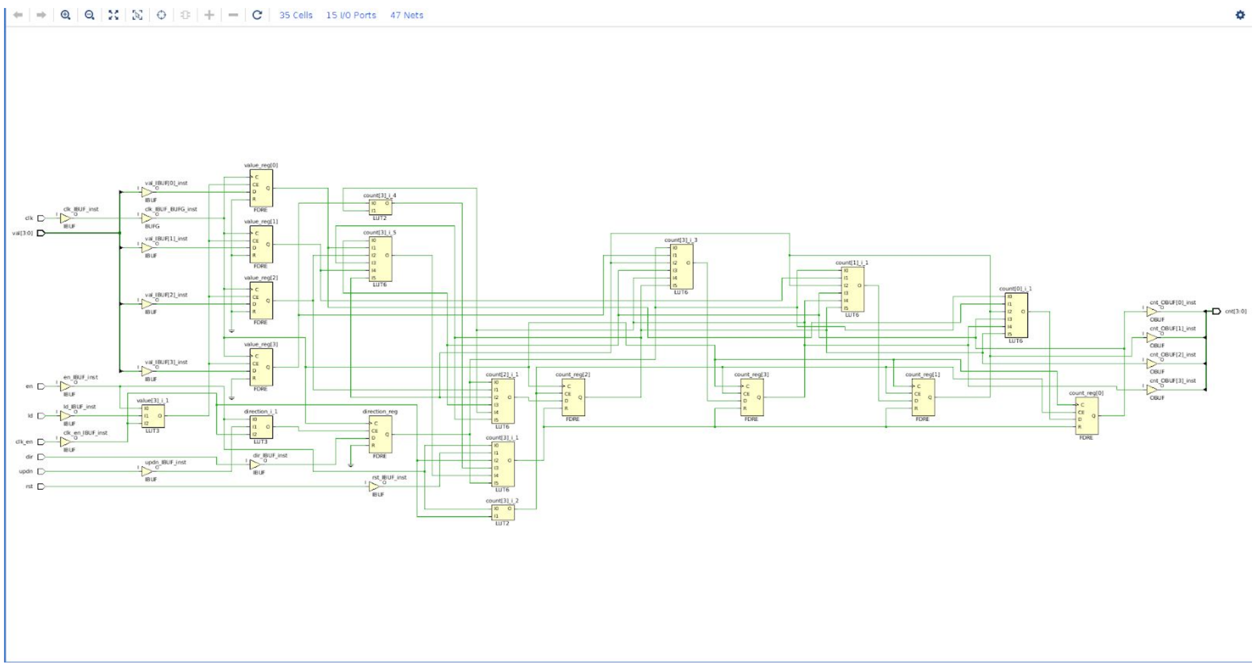


```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- 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 fancy_counter is
    port (
        clk, clk_en : in std_logic;
        dir, en, ld, 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 count : std_logic_vector (3 downto 0) := (others => '0');
    signal value : std_logic_vector (3 downto 0) := "1111";
    signal direction : std_logic := '1';

begin
```

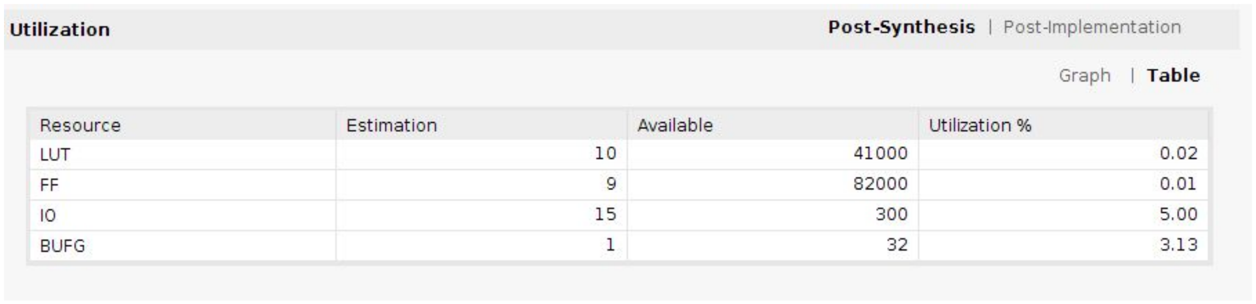


```vhdl
begin

    cnt <= count;

    process(clk)
    begin
        if rising_edge(clk) then
            if en = '1' then
                if clk_en = '1' then
                    if updn = '1' then
                        direction <= dir;
                    end if;
                    if direction = '1' then
                        if count = value then
                            count <= (others => '0');
                        else
                            count <= std_logic_vector(unsigned(count) + 1);
                        end if;
                    else
                        if unsigned(count) = 0 then
                            count <= value;
                        else
                            count <= std_logic_vector(unsigned(count) - 1);
                        end if;
                    end if;
                    if ld = '1' then
                        value <= val;
                    end if;
                end if;
                if rst ='1' then
                    count <= (others => '0');
                end if;
            end if;
        end if;
    end process;


end Behavioral;
```

Synthesis Schematic:



Post Synthesis Utilization Table:

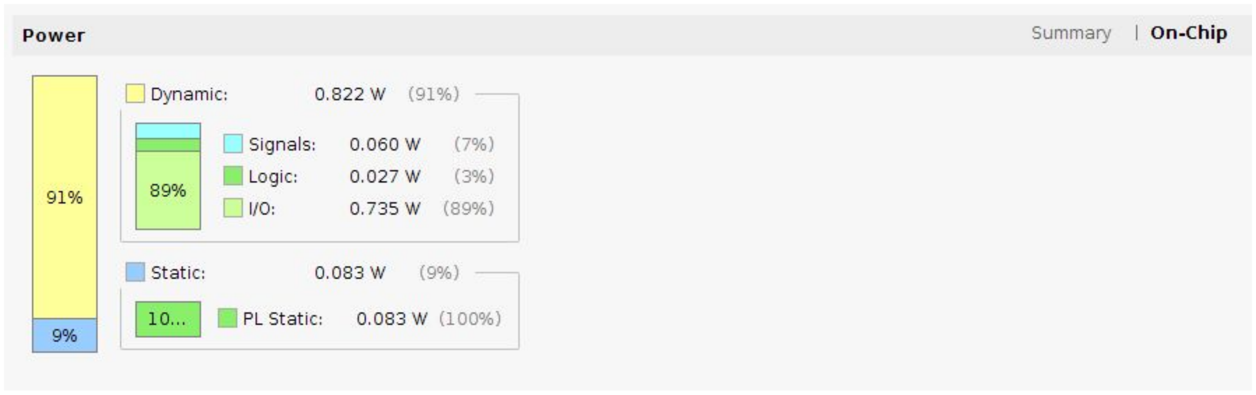


Utilization **Post-Synthesis** | Post-Implementation

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	10	41000	0.02
FF	9	82000	0.01
IO	15	300	5.00
BUFG	1	32	3.13

On-Chip Power Graphs



Power Summary | **On-Chip**

91%
9%

Dynamic: 0.822 W (91%)

89%

Signals:	0.060 W	(7%)
Logic:	0.027 W	(3%)
I/O:	0.735 W	(89%)

Static: 0.083 W (9%)

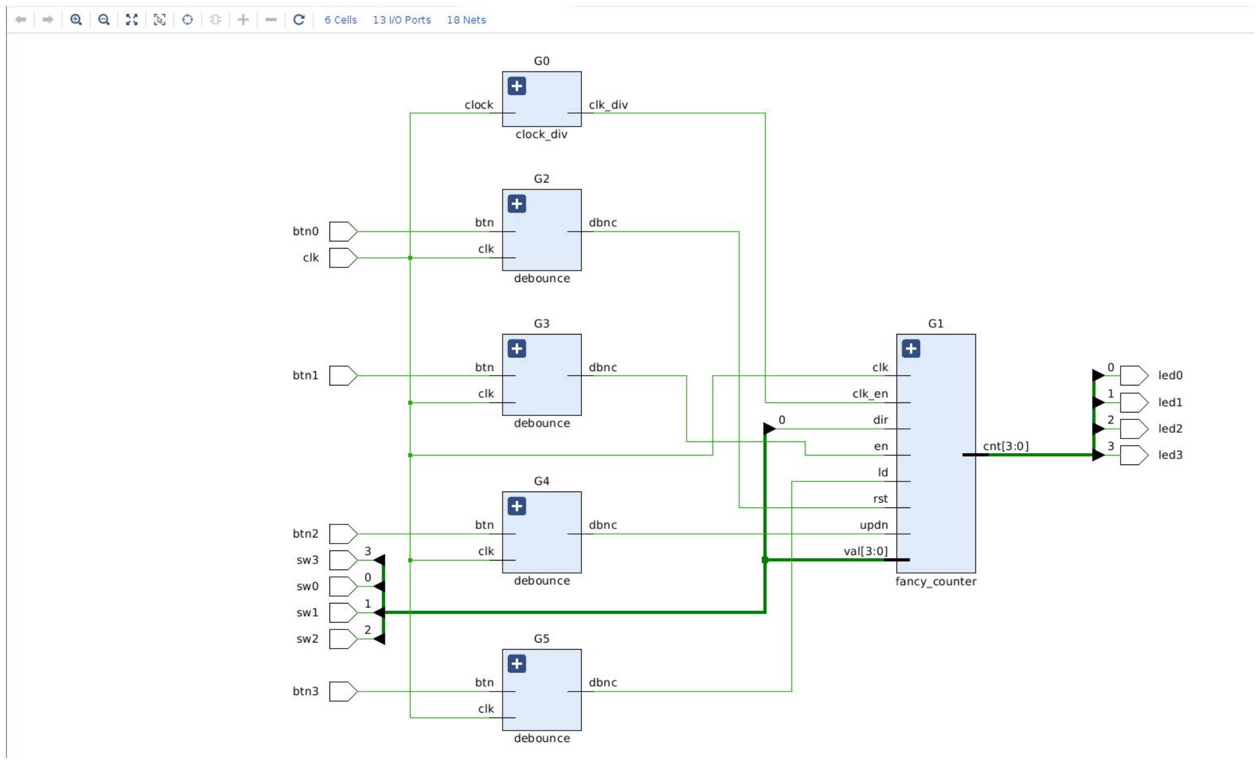
10...

PL Static: 0.083 W (100%)

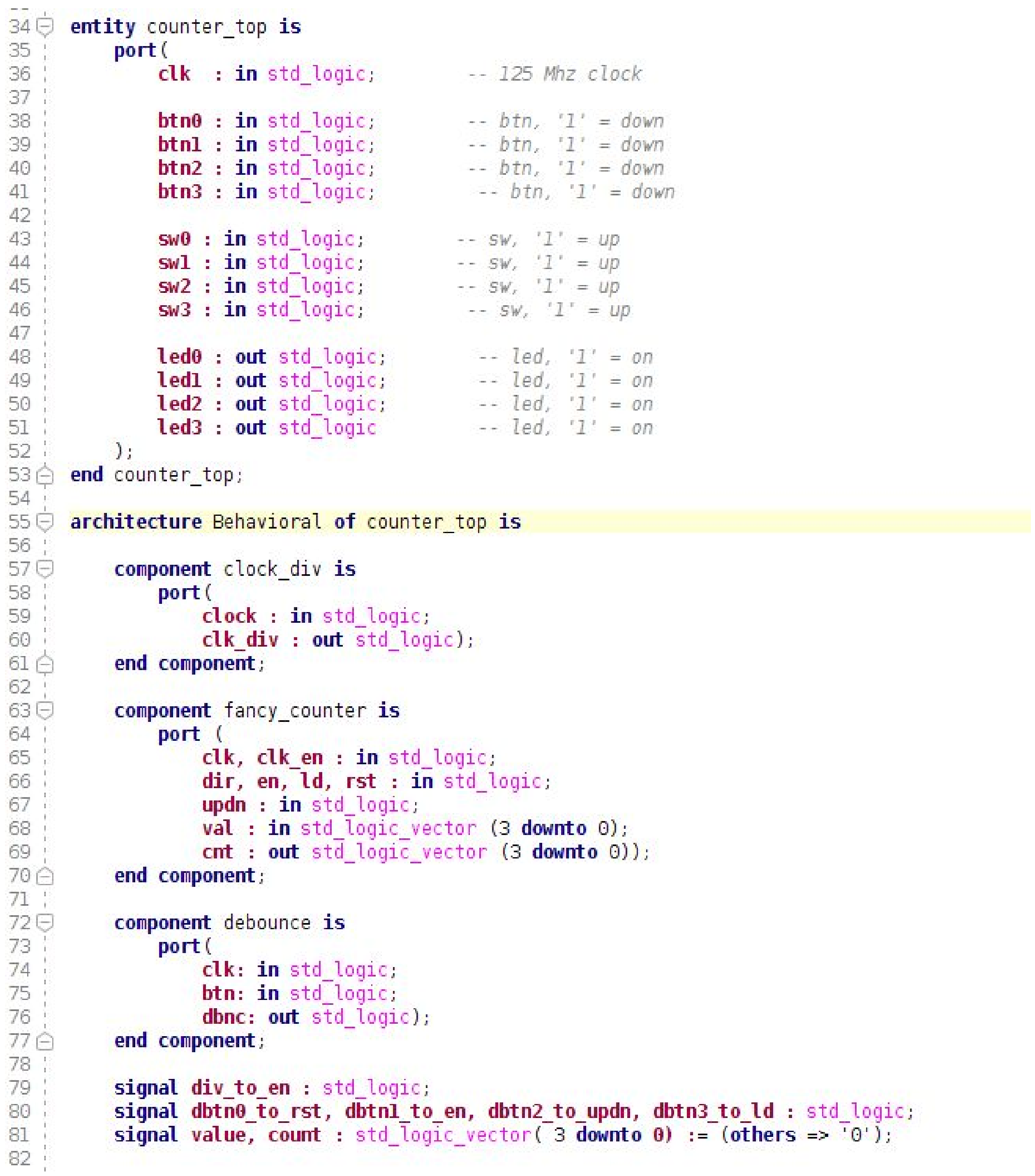
Counter_Top

In this lab, we will put together the previous modules of clock_div, fancy_counter and debounce into one circuit. All of the button inputs will be debounced which will then go into various inputs of the counter. The 125MHz clock will be divided into a 2Hz clock using clock_div, which will then be used for the clock of the counter.

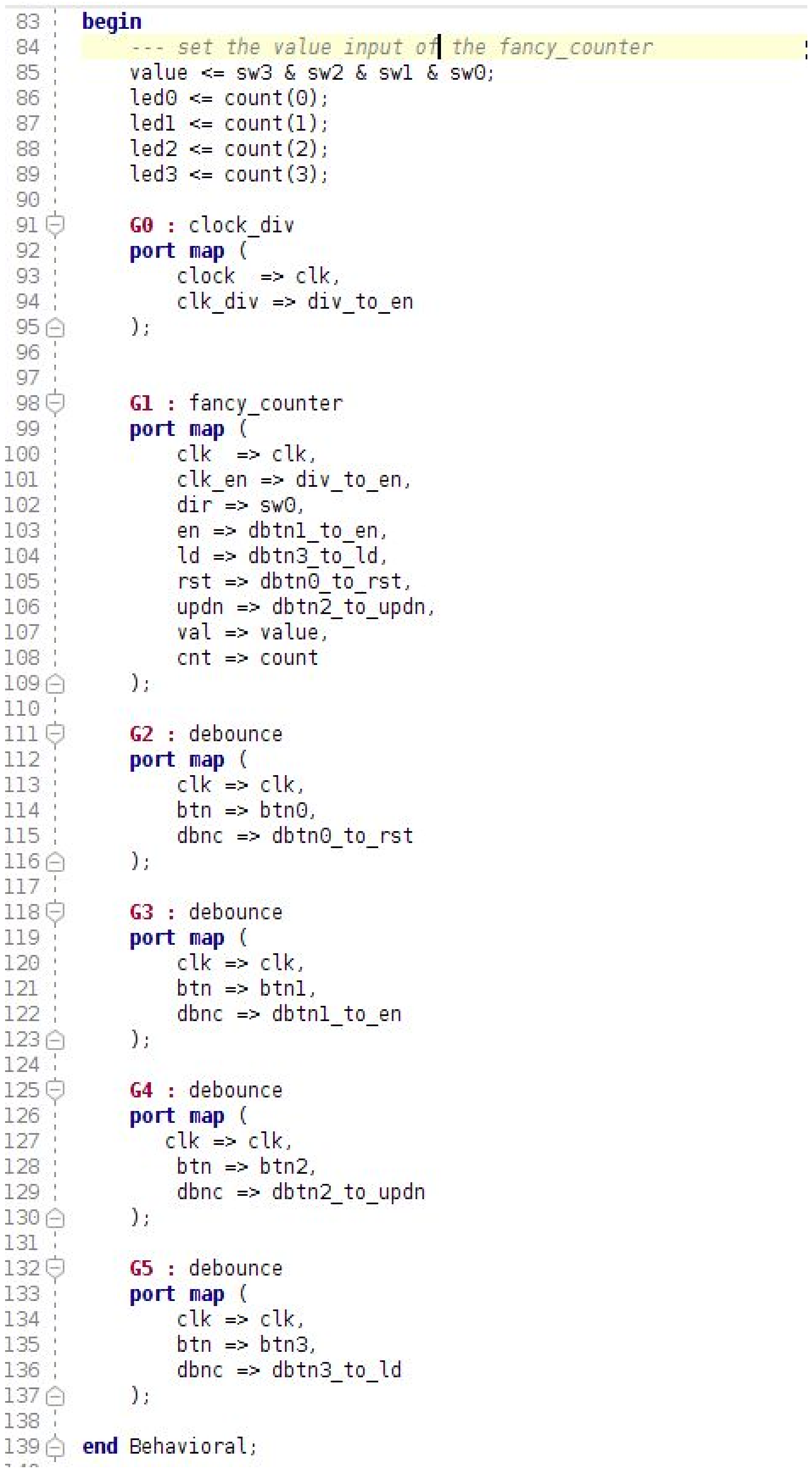
RTL Schematic:



Code:

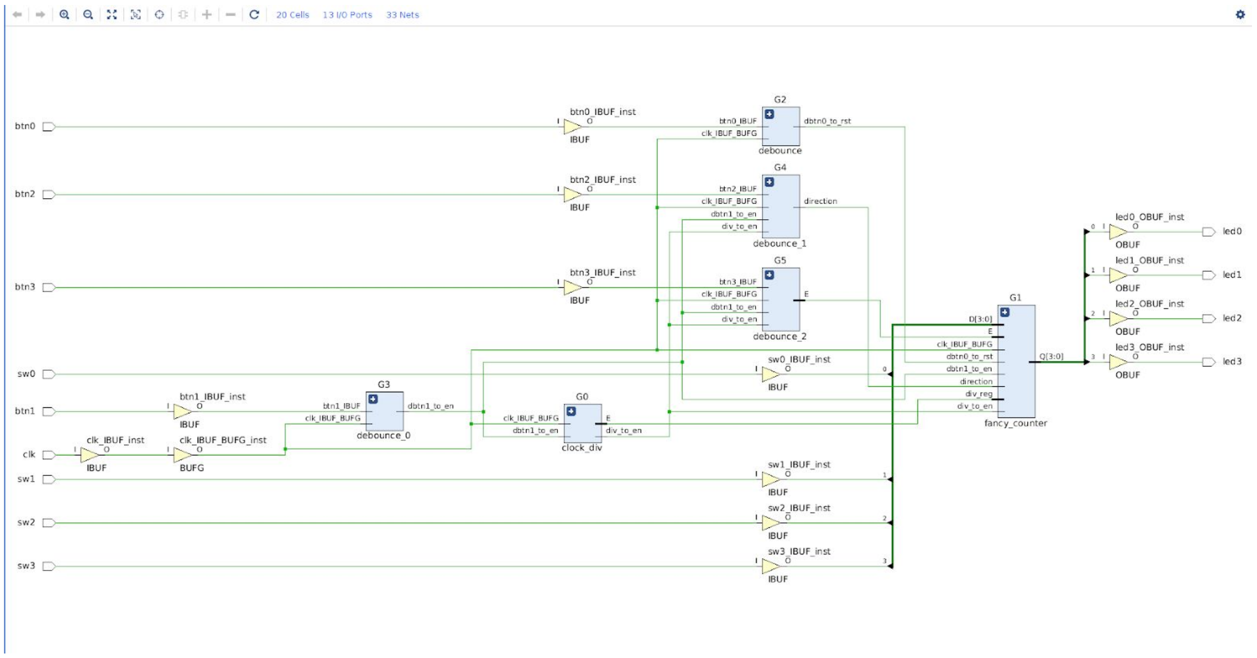


```vhdl
entity counter_top is
    port(
        clk  : in std_logic;        -- 125 Mhz clock

        btn0 : in std_logic;        -- btn, '1' = down
        btn1 : in std_logic;        -- btn, '1' = down
        btn2 : in std_logic;        -- btn, '1' = down
        btn3 : in std_logic;         -- btn, '1' = down

        sw0 : in std_logic;         -- sw, '1' = up
        sw1 : in std_logic;         -- sw, '1' = up
        sw2 : in std_logic;         -- sw, '1' = up
        sw3 : in std_logic;          -- sw, '1' = up

        led0 : out std_logic;        -- led, '1' = on
        led1 : out std_logic;        -- led, '1' = on
        led2 : out std_logic;        -- led, '1' = on
        led3 : out std_logic         -- led, '1' = on
    );
end counter_top;

architecture Behavioral of counter_top is

    component clock_div is
        port(
            clock : in std_logic;
            clk_div : out std_logic);
    end component;

    component fancy_counter is
        port (
            clk, clk_en : in std_logic;
            dir, en, ld, 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 is
        port(
            clk: in std_logic;
            btn: in std_logic;
            dbnc: out std_logic);
    end component;

    signal div_to_en : std_logic;
    signal dbtn0_to_rst, dbtn1_to_en, dbtn2_to_updn, dbtn3_to_ld : std_logic;
    signal value, count : std_logic_vector( 3 downto 0) := (others => '0');

```

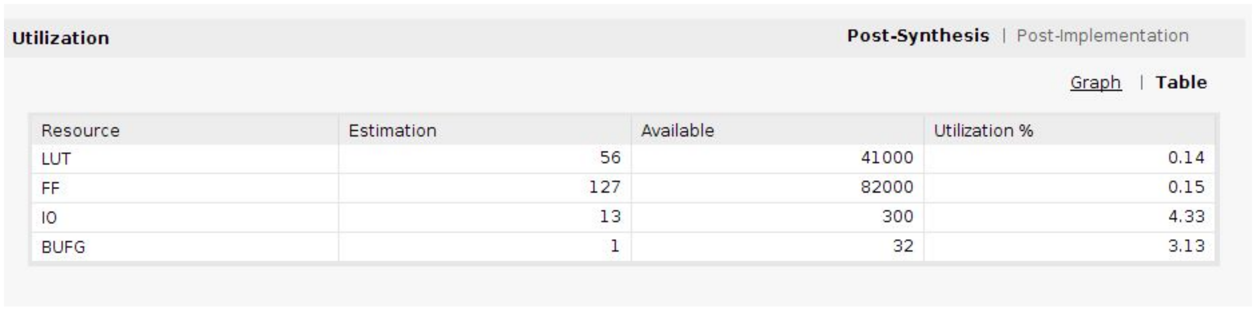


```vhdl
begin
    --- set the value input of the fancy_counter
    value <= sw3 & sw2 & sw1 & sw0;
    led0 <= count(0);
    led1 <= count(1);
    led2 <= count(2);
    led3 <= count(3);

    G0 : clock_div
    port map (
        clock   => clk,
        clk_div => div_to_en
    );


    G1 : fancy_counter
    port map (
        clk   => clk,
        clk_en => div_to_en,
        dir => sw0,
        en => dbtn1_to_en,
        ld => dbtn3_to_ld,
        rst => dbtn0_to_rst,
        updn => dbtn2_to_updn,
        val => value,
        cnt => count
    );

    G2 : debounce
    port map (
        clk => clk,
        btn => btn0,
        dbnc => dbtn0_to_rst
    );

    G3 : debounce
    port map (
        clk => clk,
        btn => btn1,
        dbnc => dbtn1_to_en
    );

    G4 : debounce
    port map (
       clk => clk,
        btn => btn2,
        dbnc => dbtn2_to_updn
    );

    G5 : debounce
    port map (
        clk => clk,
        btn => btn3,
        dbnc => dbtn3_to_ld
    );

end Behavioral;
```

Synthesis Schematic:



Post Synthesis Utilization Table:

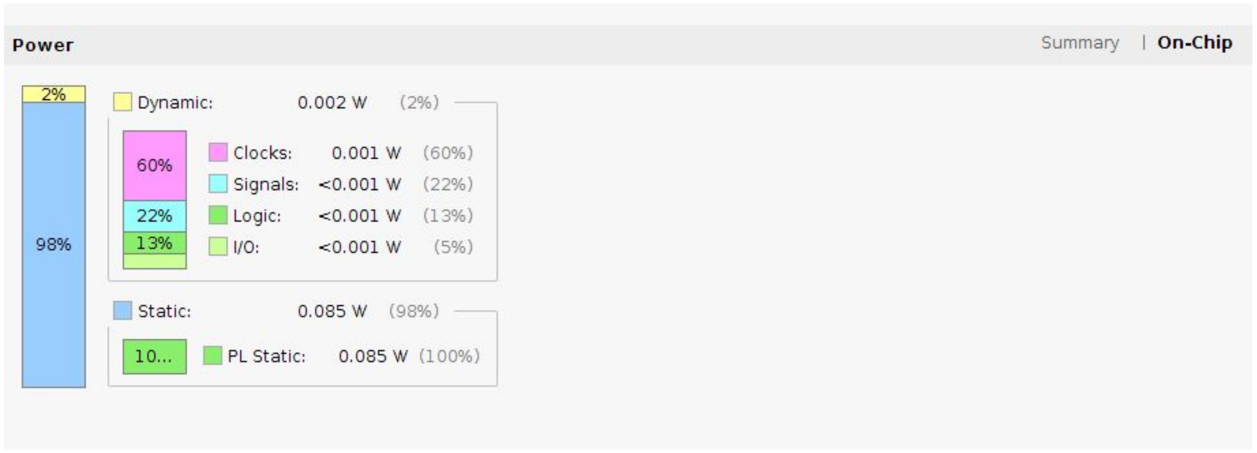


Utilization — **Post-Synthesis** | Post-Implementation

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	56	41000	0.14
FF	127	82000	0.15
IO	13	300	4.33
BUFG	1	32	3.13

On-Chip Power Graphs



Power — Summary | **On-Chip**

Dynamic:	0.002 W	(2%)	
Clocks:	0.001 W	(60%)	
Signals:	<0.001 W	(22%)	
Logic:	<0.001 W	(13%)	
I/O:	<0.001 W	(5%)	
Static:	0.085 W	(98%)	
PL Static:	0.085 W	(100%)	

XDC Changes (Github file)

The clock, switches, buttons, were all uncommented. This version is attached in the github folder.