Embedded Systems

Lab Report 1

Matt Mann

2/28/19

Purpose

To make a counter

Part 1

Theory of Operation

This circuit will have a counter to keep track of the ticks from the clock. When a certain number of ticks has been reached, the output will change and imitate a clock.

Design

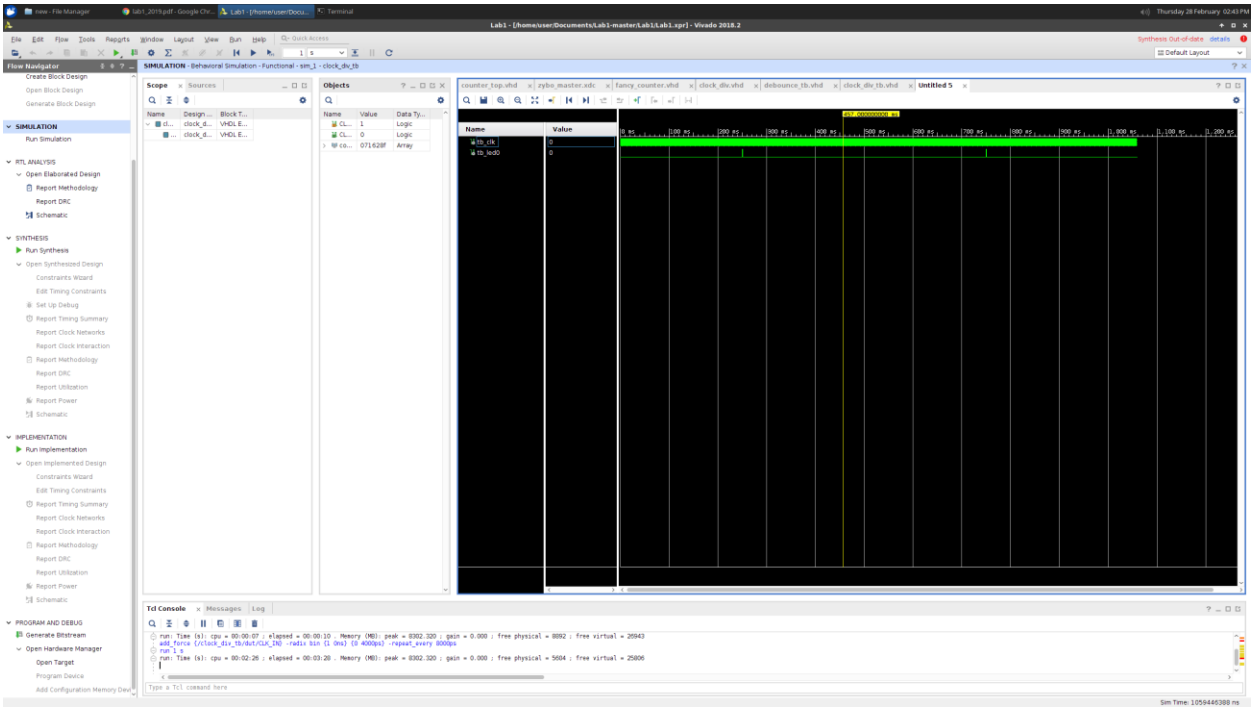
```
Library
IEEE;
          use IEEE.STD_LOGIC_1164.ALL;
          use IEEE.NUMERIC_STD.all;
entity clock_div is
    Port ( CLK_IN : in STD_LOGIC;
           CLK_OUT : out STD_LOGIC);
end clock_div;

architecture Behavioral of clock_div is
    signal counter : std_logic_vector(26 downto 0)
:= (others => '0');
begin
    process (CLK_IN)
    begin
        if(rising_edge(CLK_IN)) then
            if (unsigned(counter) < 62499999) then
               counter <=
std_logic_vector(unsigned(counter) + 1);
            else
               counter <= (others => '0');
            end if;
        end if;
```

```
            if (unsigned(counter) = 31250000) then
                CLK_OUT <= '1';
            else
                CLK_OUT <= '0';
            end if;
        end process;


    end Behavioral;
```

Test

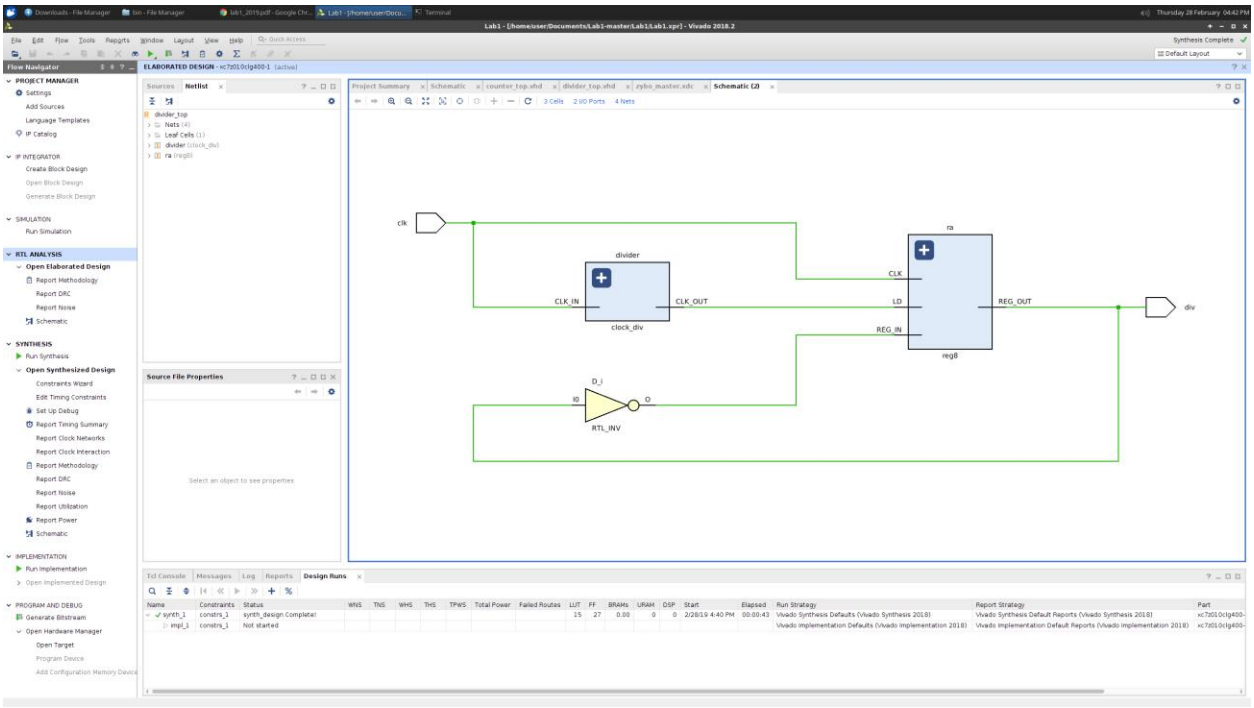
```
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;

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

architecture testbench of clock_div_tb is
    signal tb_clk : std_logic := '0';
    signal tb_led0 : std_logic;

    component clock_div is
        port(

            CLK_IN : in STD_LOGIC;
                CLK_OUT : out STD_LOGIC

        );
    end component;
begin
    clk_gen_proc: process
    begin

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

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

    end process clk_gen_proc;

    -- flip the switch high after 1ms

    dut : clock_div
```

```
        port map (

            CLK_IN  => tb_clk,
            CLK_OUT  => tb_led0

        );
end testbench;
```

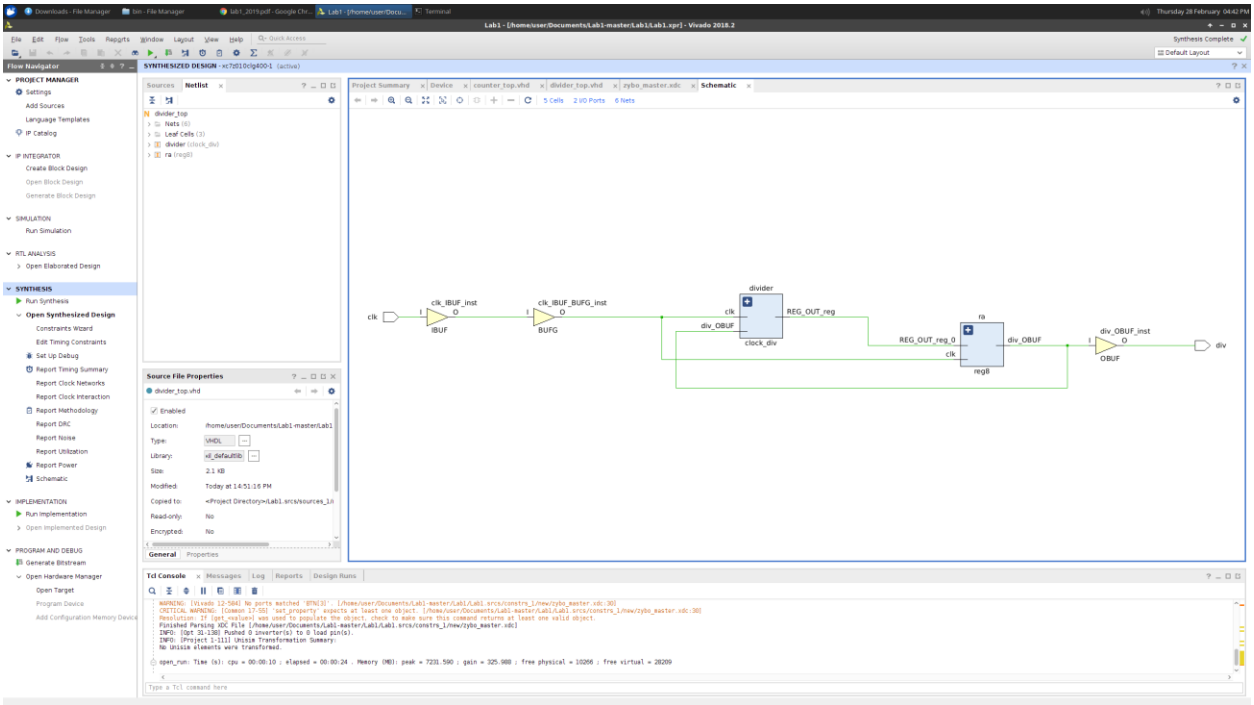


Implementation

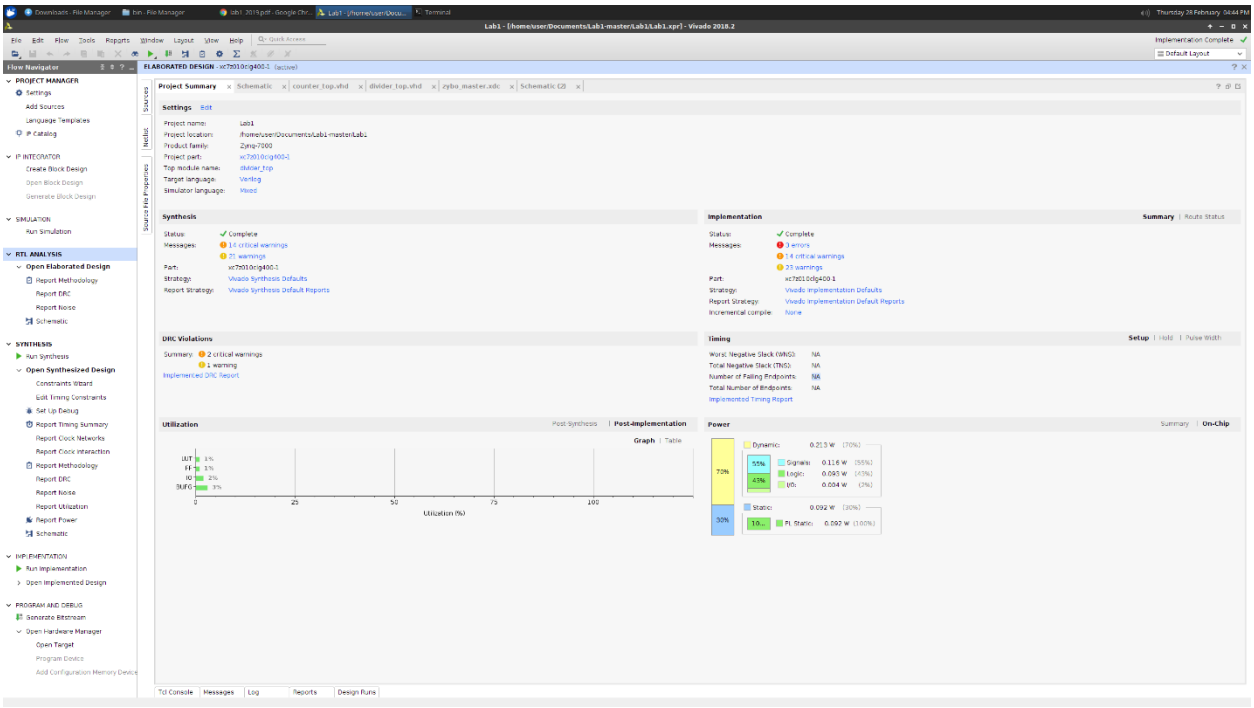
Elaborated Schematic



Synthesis Schematic



Project Summary



XDC

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


##LEDs
       set_property -dict { PACKAGE_PIN M14    IOSTANDARD LVCMOS33 } [get_ports { CLK_OUT
       }]; #IO_L23P_T3_35 Sch=LED0
```

I had to change the port names for everything except the clock, so it matched my code.

Discussion

Question 1.1

We need to divide by 4.

Question 1.2

26 bits are needed. 26 bits can store 67,554,432.

I have learned how to use the constraints file and change the ports to match what is in my code.

Part 2

Theory of Operation

This will have a counter keep track of the time and how many times we received the correct input, a ‘1’. Once the counter reaches a certain value, our circuit will pass that output.

Design

Clock_div

```
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 counter : std_logic_vector(22 downto 0)
:= (others => '0');
begin
    process(clk)
    begin
        if clk'event and clk='1' then
          if(btn='1') then
            if(unsigned(counter)>=2500000) then
                counter <= counter;
            else
                counter <=
std_logic_vector(unsigned(counter) + 1);
            end if;
          else
            counter <= "00000000000000000000000";
          end if;
          if(unsigned(counter)>=2500000) then
            dbnc <= '1';
          else
            dbnc <= '0';
```

```
                                        end if;
                                    end if;
                                end process;


                            end Behavioral;
```

Divider_top

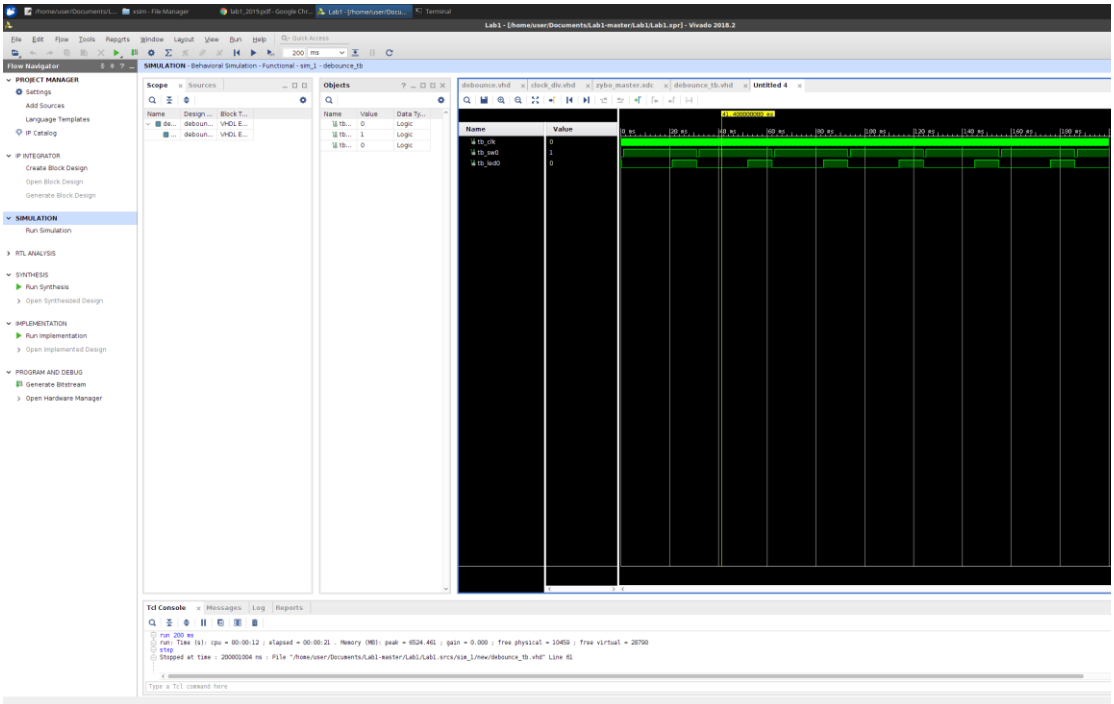
```
library
IEEE;
          use IEEE.STD_LOGIC_1164.ALL;


          entity reg8 is --- ENTITY
              Port ( REG_IN : in std_logic;
                  LD,CLK : in std_logic;
                  REG_OUT : out std_logic);
          end reg8;


          architecture reg8 of reg8 is --- ARCHITECTURE
              begin
                  reg: process(CLK)
                      begin
                          if (rising_edge(CLK)) then
                              if (LD = '1') then
                                  REG_OUT <= REG_IN;
                              end if;
                          end if;
                      end process;
                  end reg8;


          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;
           div : out STD_LOGIC);
end divider_top;


architecture rt1_structural of divider_top is

    component clock_div
        port(CLK_IN: in std_logic;
            CLK_OUT: out std_logic);
    end component;

    component reg8
        Port ( REG_IN : in std_logic;
            LD,CLK : in std_logic;
            REG_OUT : out std_logic);
    end component;

    signal clk_div_result : std_logic;
    signal D : std_logic;
    signal Q : std_logic;
begin
    D <= not Q;
    ra: reg8
    port map ( REG_IN => D,
            LD => clk_div_result,
            CLK => clk,
            REG_OUT => Q );
    div <= Q;
    divider:clock_div
    port map(CLK_IN => clk,
             CLK_OUT => clk_div_result);

end rt1_structural;
```

Test

```
librar
y
ieee;
            use ieee.std_logic_1164.all;
            use ieee.numeric_std.all;

        entity debounce_tb is
        --  Port ( );
        end debounce_tb;


        architecture testbench of debounce_tb is
            signal tb_clk : std_logic := '0';
            signal tb_sw0 : std_logic := '0';
            signal tb_led0 : std_logic;

            component debounce is
                port(

                    btn  : in std_logic;        -- 125 Mhz clock
                    clk  : in std_logic;        -- switch, '1' = on

                    dbnc : out std_logic        -- led, '1' = on

                );
            end component;
        begin
            clk_gen_proc: process
            begin

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

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

            end process clk_gen_proc;

            -- flip the switch high after 1ms
```

```
    switch_proc: process
    begin

        wait for 1 ms;
        tb_sw0 <= '1';

        wait for 30 ms;
        tb_sw0 <= '0';

    end process switch_proc;
    dut : debounce
        port map (

            clk  => tb_clk,
            btn  => tb_sw0,
            dbnc => tb_led0

        );
end testbench;
```



Discussion

Question 2.1

It is ‘1’.

Question 2.2

We would not have to alter the overall design, but we would have to change the value we check for. ‘0’ instead of ‘1’.

Question 2.3

We would need a counter to count 2,500,000 ticks.

Question 2.4

We would need 22 bits. 22 bits stores up to 4,194,304

I have learned how to make and use a testbench. I also learned how to make a debouncer. I am confident in my ability to make a debouncer now.

Part 3

Theory of Operation

This circuit will need two signals. One to keep track of the current value, and another to keep track of the direction. When told to be input signals, the circuit will change these values to what the circuit says. The circuit will then count up/down until 0 or the value is reached, then it will repeat. This stuff only happens when the two enables are asserted. If reset is asserted the counter outputs 0. Reset can only work when enable is on. It doesn’t care about clock enable though.

Design

```
library
IEEE;
          use IEEE.STD_LOGIC_1164.ALL;
          use IEEE.NUMERIC_STD.all;
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 reg : STD_LOGIC_VECTOR (3 downto 0) := (others => '0');
    signal direct : STD_LOGIC  := '0';
    signal counter : std_logic_vector(3 downto 0) := (others => '0');
begin
    process(clk)
    begin
        if(rising_edge(CLK)) then
            if(en = '1') then
                if(rst = '1') then
                    cnt <= "0000";
                    counter <= "0000";
                else
                    if(clk_en = '1') then
                            if(updn = '1') then
                           direct <= dir;
                            end if;
                      if(ld = '1') then
                        reg <= val;
                      end if;
                      if(direct = '1') then
                        if(counter = reg) then
```

```
                                    counter <= "0000";
                                  else
                                    counter <= std_logic_vector(unsigned(counter)
                + 1);
                                  end if;
                                else
                                  if(counter = "0000") then
                                    counter <= reg;
                                  else
                                    counter <= std_logic_vector(unsigned(counter)
                - 1);
                                  end if;
                                end if;


                              end if;
                            end if;
                          end if;
                cnt <= counter;
                        end if;
                    end process;



                end Behavioral;
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 reg : STD_LOGIC_VECTOR
(3 downto 0);
    signal direct : STD_LOGIC;
    signal counter :
std_logic_vector(3 downto 0) :=
(others => '0');
begin
    process(clk)
    begin
        if(rising_edge(CLK)) then
            if(en = '1') then
                if(rst = '1') then
                    cnt <= "0000";
                    counter <=
"0000";
                end if;
                if(clk_en = '1')
then
                  if(updn = '1')
then
                       direct <=
dir;
                  end if;
                   if(ld = '1')
then
                       reg <=
val;
                   end if;
                   if(direct =
'1') then
                       if(counter
= reg) then

counter <= "0000";
```

```
                        else
counter <=
std_logic_vector(unsigned(counter)
+ 1);
                        end if;
                    else
                        if(counter
= "0000") then
counter <= reg;
                        else
counter <=
std_logic_vector(unsigned(counter)
- 1);
                        end if;
                    end if;
                    cnt <=
counter;
                end if;
            end if;
        end if;
    end process;


end Behavioral;
```

Discussion

I didn't make any big discoveries in this part. This was good practice with writing custom code though. I have a complete understanding of how to make a file and create my own code.

Part 4

Theory of Operation

This circuit takes in all the previous components and uses port mapping to combine them. Switches determine what value to count to, and buttons determine everything else. All buttons are debounced before going into the counter.

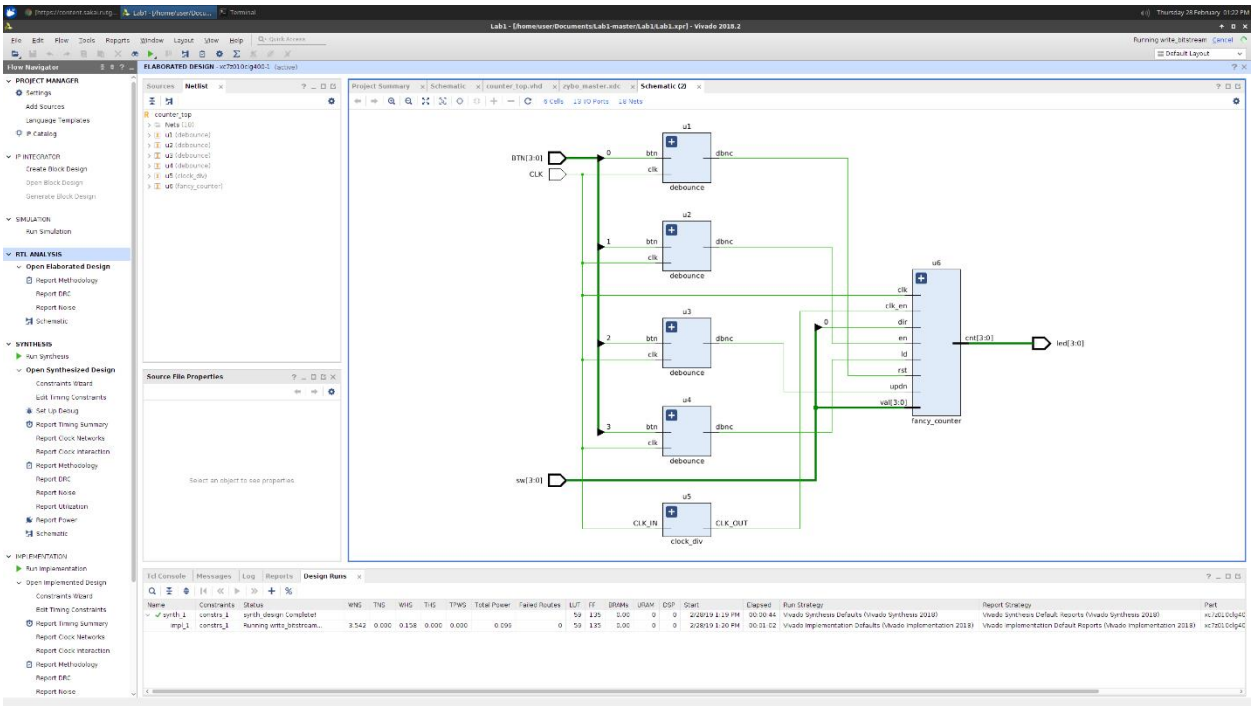
Design

```
library
IEEE;
          use IEEE.STD_LOGIC_1164.ALL;
entity counter_top is
                                                    Port ( CLK : in STD_LOGIC;
                                                           BTN : in STD_LOGIC_VECTOR (3 downto 0);
                                                           sw : in STD_LOGIC_VECTOR (3 downto 0);
                                                           led : out STD_LOGIC_VECTOR (3 downto
                                                0));
                                                end counter_top;


                                                architecture rt1_structural of counter_top is
                                                    component debounce
                                                        port(btn : in STD_LOGIC;
                                                               clk : in STD_LOGIC;
                                                               dbnc : out STD_LOGIC);
                                                    end component;

                                                    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 clock_div
                                                            port(CLK_IN: in std_logic;
                                                                CLK_OUT: out std_logic);
                                                        end component;

                                                    signal u1_out : std_logic;
                                                    signal u2_out : std_logic;
                                                    signal u3_out : std_logic;
                                                    signal u4_out : std_logic;
                                                    signal u5_out : std_logic;
```

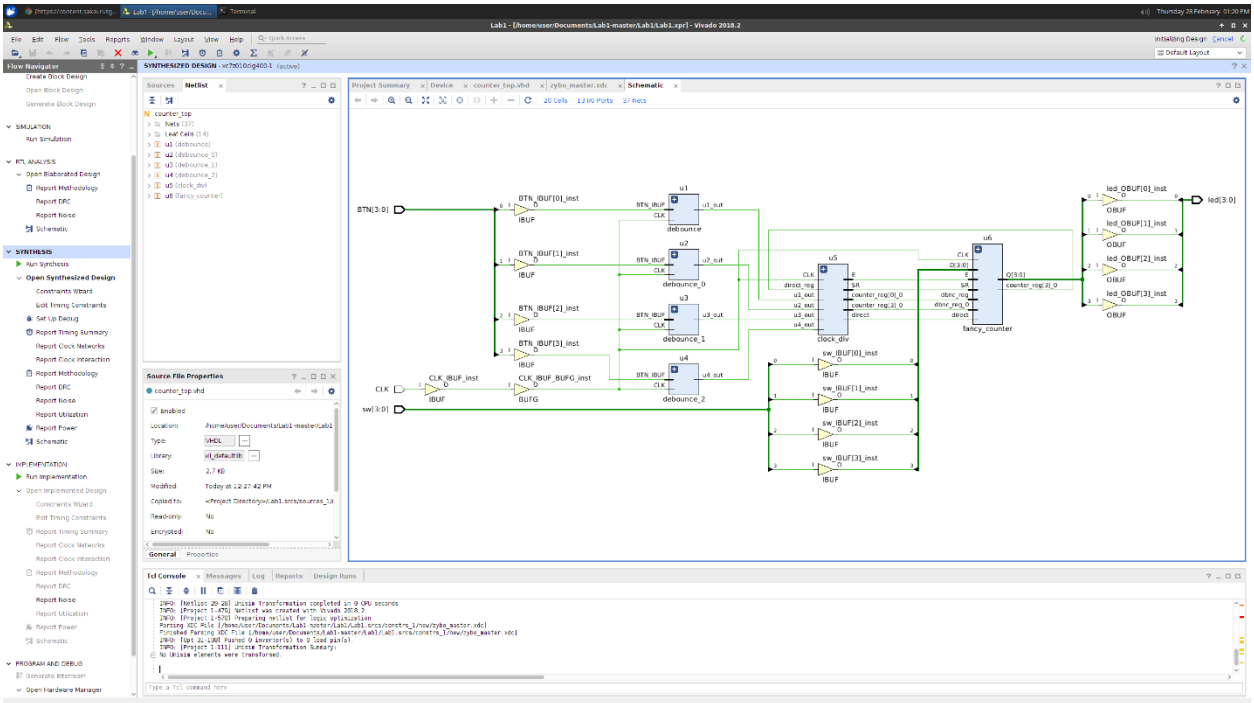
```
begin
    u1: debounce
    port map(btn => BTN(0),
             clk => CLK,
             dbnc => u1_out);

    u2:debounce
    port map(btn => BTN(1),
                clk => CLK,
                dbnc => u2_out);
    u3:debounce
    port map(btn => BTN(2),
                clk => CLK,
                dbnc => u3_out);

    u4:debounce
    port map(btn => BTN(3),
                clk => CLK,
                dbnc => u4_out);

    u5:clock_div
    port map(CLK_IN => CLK,
                CLK_OUT => u5_out);

    u6: fancy_counter
    port map(clk => CLK,
                clk_en => u5_out,
                dir => sw(0),
                en => u2_out,
                ld => u4_out,
                rst => u1_out,
                updn => u3_out,
                val => sw(3 downto 1),
                cnt => led);
end rt1_structural;
```

Implementation

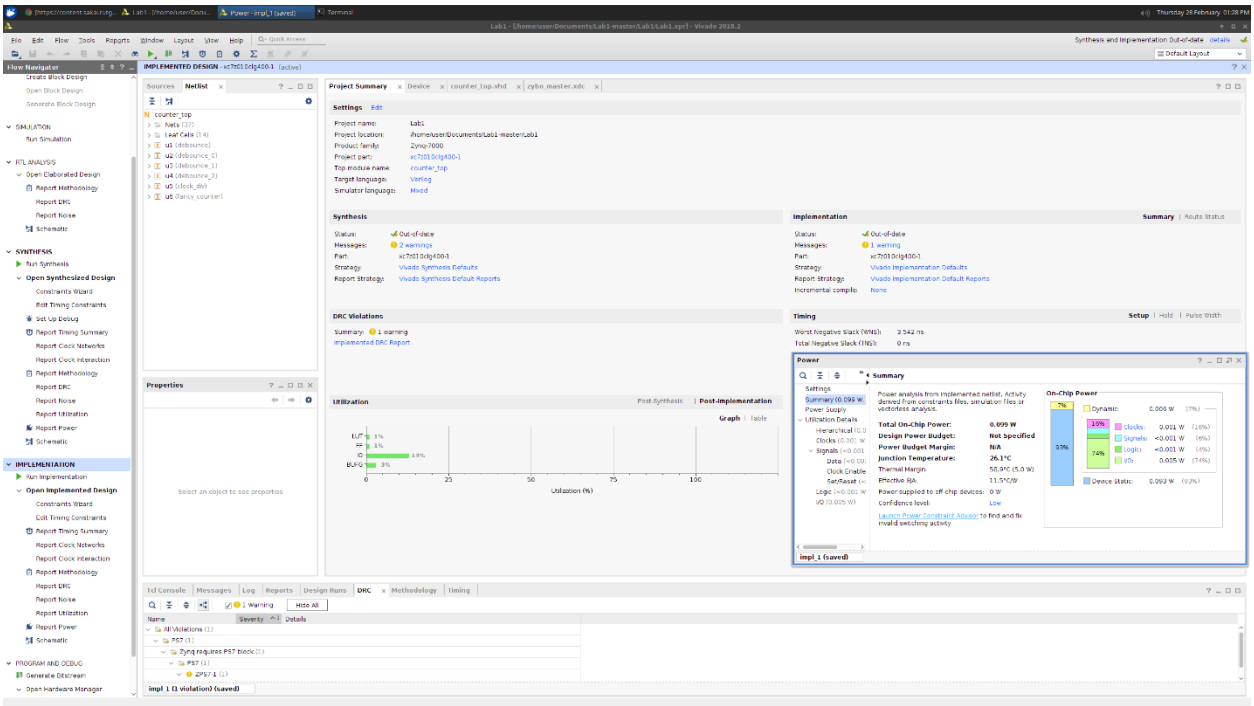
Elaboration Schematic



Synthesis Schematic



Project Summary



XDC

```
##Clock
signal
        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
```

```
##Buttons
set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports { BTN[0]
}]; #IO_L20N_T3_34 Sch=BTN0
set_property -dict { PACKAGE_PIN P16   IOSTANDARD LVCMOS33 } [get_ports { BTN[1]
}]; #IO_L24N_T3_34 Sch=BTN1
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports { BTN[2]
}]; #IO_L18P_T2_34 Sch=BTN2
set_property -dict { PACKAGE_PIN Y16   IOSTANDARD LVCMOS33 } [get_ports { BTN[3]
}]; #IO_L7P_T1_34 Sch=BTN3
```

I added buttons to the XDC file and changes the ports name for clock.

Discussion

I didn't learn many new things in this part either. I did however fully master the XDC file manipulation though.