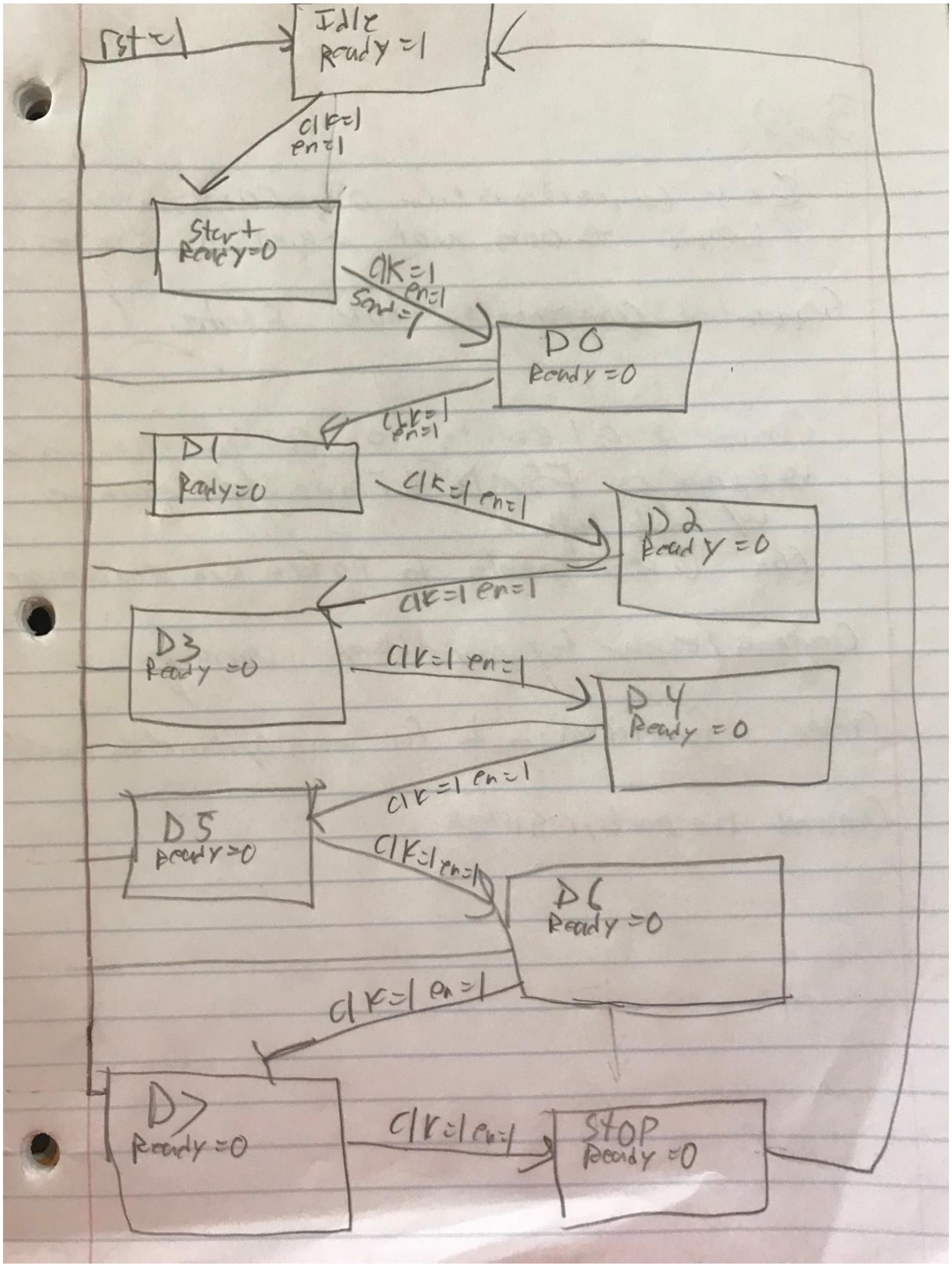
Kevin Honeker

Lab #3

3/29/19

Purpose: Ten purpose of this lab was to learn about UARTs and to be able to implement them on a Zybo using a PMOD.

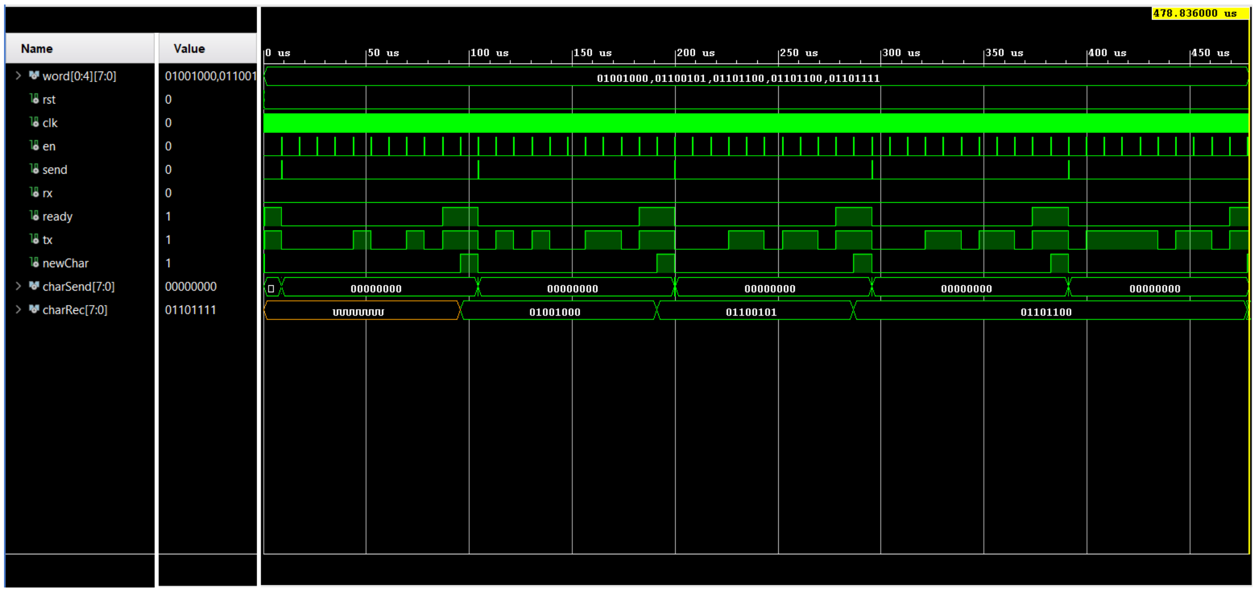
Pre lab


rst=1
Idle
Ready=1
clk=1
en=1
Start
Ready=0
clk=1
en=1
Send=1
D0
Ready=0
clk=1
en=1
D1
Ready=0
clk=1 en=1
D2
Ready=0
clk=1 en=1
D3
Ready=0
clk=1 en=1
D4
Ready=0
clk=1 en=1
D5
Ready=0
clk=1 en=1
D6
Ready=0
clk=1 en=1
D7
Ready=0
clk=1 en=1
STOP
Ready=0

Part 1

Theory of Operation: The simulated circuit should transmit a series of bits which are then outputted by the receiver.

Simulation results:



uart_tx code:

```
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 uart_tx is

   Port ( clk : in STD_LOGIC;

          en : in STD_LOGIC;
          send : in STD_LOGIC;
          rst : in STD_LOGIC;
          char : in STD_LOGIC_VECTOR (7 downto 0);
          ready : out STD_LOGIC;
          tx : out STD_LOGIC);
end uart_tx;

architecture uart_tx_arch of uart_tx is
signal char_reg : std_logic_vector(7 downto 0);
type state_type is (idle, start, d1, d2, d3, d4, d5, d6, d7);
signal ps : state_type := idle;
--signal ns : state_type;
begin

process(clk)
   begin
   if clk'event and clk = '1' then
      --ps <= ns;
       if rst = '1' then
          char_reg<= (others=> '0');
          ps <= idle;
          tx<='1';
          ready<='1';
          --char_reg<= (others=> '0');

      elsif en ='1' then
        if send = '1' then
          char_reg <= char;
        end if;
```

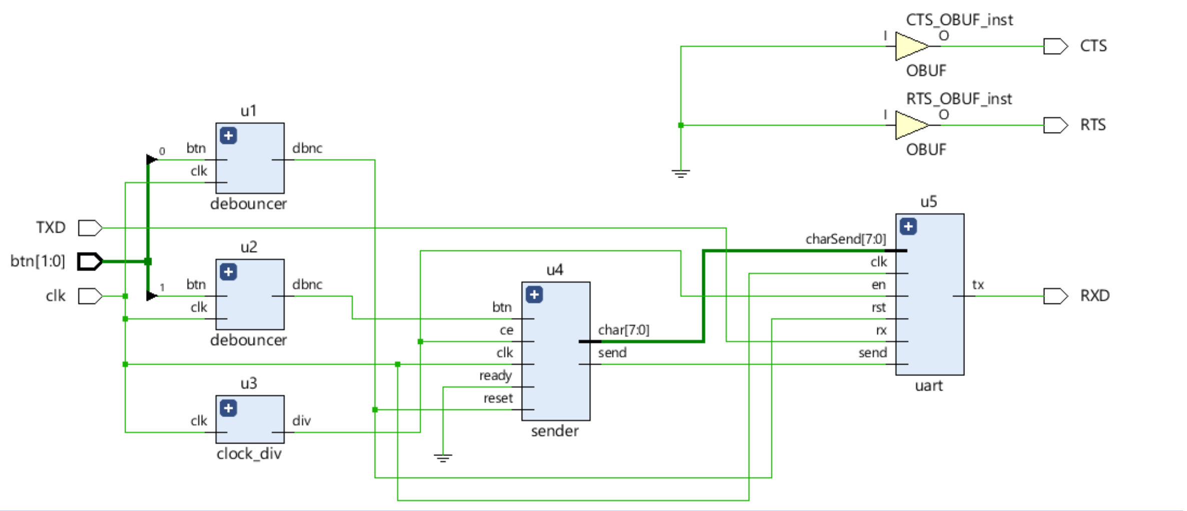
```
case ps is
    when idle =>


   -- ready <= '1';
   -- tx <= '1';
    if send = '1' then
        char_reg <= char;
        tx<= '0';
        ready <='0';
        ps <= start;


    else
        tx <= '1';
        ready <= '1';


   end if;


    when start =>


           tx <=char_reg(0);
           ready <= '0';
           ps <= d1;


    when d1 =>
  --   if send = '1' then
       tx <= char_reg(1);
       ready <= '0';
       --if en = '1' then
            ps <= d2 ;

    when d2 =>
   --  if send = '1' then
        tx <= char_reg(2);
        ready <= '0';
        --if en = '1' then
            ps <= d3;




    when d3 =>
     --if send = '1' then
        tx <= char_reg(3);
        ready <= '0';
        --if en = '1' then
            ps <=d4 ;

    when d4 =>
    -- if send = '1' then
        tx <= char_reg(4);
        ready <= '0';
        --if en = '1' then
            ps <=d5 ;

    when d5 =>
    -- if send = '1' then
```

```
            tx <= char_reg(5);

            ready <= '0';

            --if en = '1' then

                ps <=d6 ;


          when d6 =>

        --   if send = '1' then

            tx <= char_reg(6);

            ready <= '0';

           -- if en = '1' then

                ps <=d7 ;



         when d7 =>

        --  if send = '1' then

            tx <= char_reg(7);

            ready <= '0';

            --if en = '1' then

               ps <=idle;




    end case;

end if;

end if;

   --end if;

end process;
end uart_tx_arch;
```

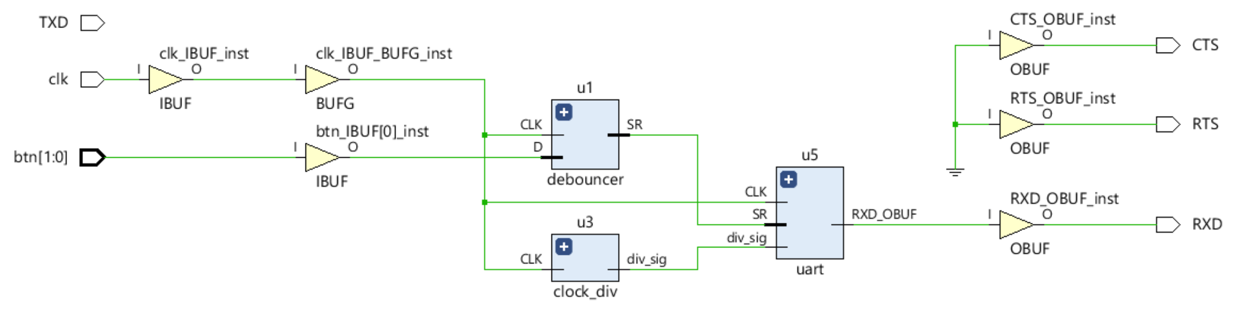
Part 2

Theory of operation: For this part we were to implement a sender file and a top level design to string all of the files together so we can use the zybo board to interact with a PMOD so every time btn(1) was pressed, my NETID would print to a termite console. Unfortunately, I could not get this to print anything to the termite console.

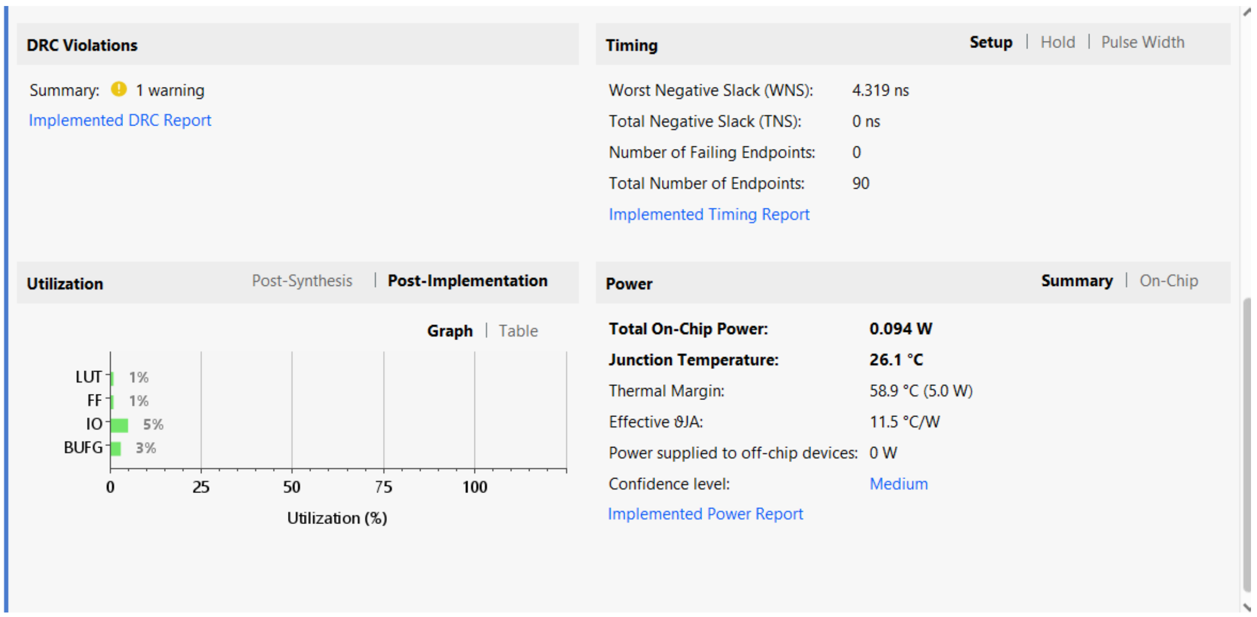
RTL Analysis



Synthesis schematic



Post synthesis utilization table



DRC Violations

Summary: 1 warning

Implemented DRC Report

Timing — **Setup** | Hold | Pulse Width

Worst Negative Slack (WNS):	4.319 ns
Total Negative Slack (TNS):	0 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	90

Implemented Timing Report

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

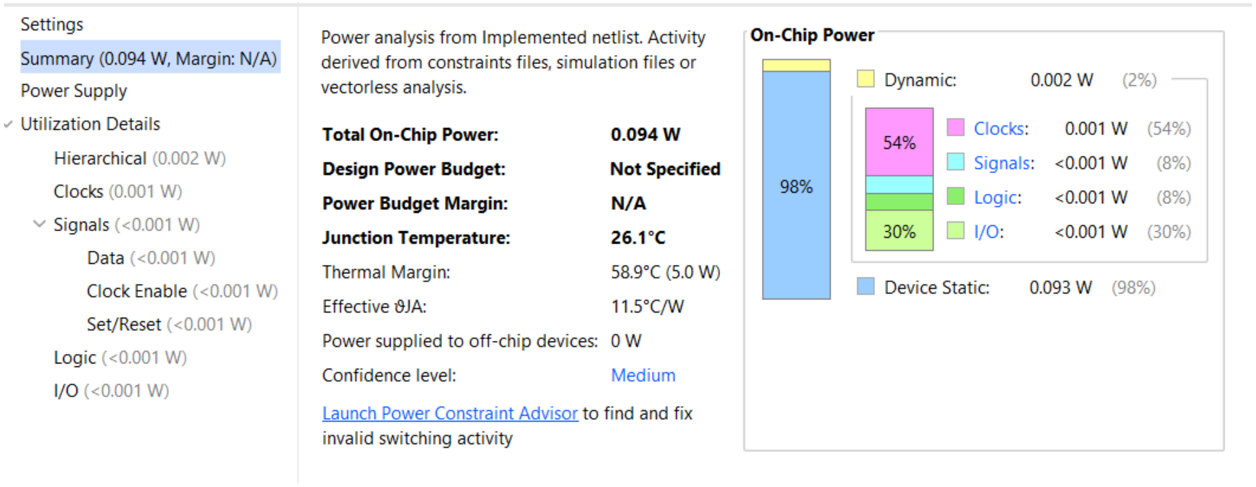
Graph | Table

Power — **Summary** | On-Chip

Total On-Chip Power:	**0.094 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:	Medium

Implemented Power Report

on chip pwer graphs



Settings

Summary (0.094 W, Margin: N/A)

Power Supply

Utilization Details
- Hierarchical (0.002 W)
- Clocks (0.001 W)
- Signals (<0.001 W)
  - Data (<0.001 W)
  - Clock Enable (<0.001 W)
  - Set/Reset (<0.001 W)
- Logic (<0.001 W)
- I/O (<0.001 W)

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

Total On-Chip Power:	**0.094 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	(54%)
🟦	Signals:	<0.001 W	(8%)
🟩	Logic:	<0.001 W	(8%)
🟩	I/O:	<0.001 W	(30%)
🟦	Device Static:	0.093 W	(98%)

Sender VHDL code:

```
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 sender is

   Port ( reset : in STD_LOGIC;

         clk : in STD_LOGIC;

         ce : in STD_LOGIC;

         send : out STD_LOGIC;

         ready : in std_logic;

         btn : in std_logic;

         char : out STD_LOGIC_VECTOR (7 downto 0));

end sender;


architecture sender_arch of sender is



type state_type is (IDLE,busyA,busyB,busyC);

signal ps: state_type :=idle;

signal bicounter : std_logic_vector(2 downto 0);

type netid_arr is array(0 to 5) of std_logic_vector(7 downto 0);

signal NETID : netid_arr  := (x"6b", x"6d", x"68", x"32", x"38", x"39");
```

```
begin


process(clk,ce)

begin

    if(clk'event and clk ='1' and ce='1') then

        if(reset='1') then

            ps<=idle;

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

            bicounter<="000";

            send<='0';

         end if;

        case ps is

           when idle=>

              if(  btn='1'and ready='1' and unsigned(bicounter)<6) then

                    send<='1';

                     bicounter<=std_logic_vector(unsigned(bicounter)+1);

                     ps<=busyA;


                    char<=NETID(to_integer(unsigned(bicounter)));

                 elsif(btn='1' and ready='1' and unsigned(bicounter)=6) then

                    bicounter<="000";

                    ps<=idle;

                  end if;

             When busyA=>

                 ps<=busyB;

             When busyB=>

                 send<='0';

                 ps<=busyC;

             When busyC=>
```

```
                  if(ready='1' and btn='0') then
                      ps<=idle;
                   end if;
       end case;
         end if;
end process;

end architecture;
```

Debouncer code:

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

entity debouncer is
    port ( btn, clk : in std_logic;
           dbnc     : out std_logic);
end debouncer;
architecture debounce of debouncer is
signal count : std_logic_vector(24 downto 0);
signal shift_reg : std_logic_vector(1 downto 0);
begin

process (clk)


begin
   if clk='1' and clk'event then
    shift_reg(0) <= btn;
    shift_reg(1) <= shift_reg(0);
    if shift_reg(1) = '1' then
```

```
        count <= std_logic_vector(unsigned(count) +1);
        if unsigned(count) < 2500000 then
            dbnc <= '0';
        else
            dbnc <= '1';
        end if;
    else
        count<=(others =>'0');
        dbnc <='0';
     -- if btn= not '1' then
       --  count := 0;          --old code
     -- elsif btn='1' then
       --  count := count + 1;
      end if;
  -- end if;



    end if;
    --end if;
end process;
end debounce;
```

Clock Divider code

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

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

architecture behavior of clock_div is

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

begin

    process(clk)
    begin

        if rising_edge(clk) then


                if (unsigned(counter) < 1085) then
                    counter <= std_logic_vector(unsigned(counter) + 1);

                else
                    counter <= (others => '0');

                end if;


                if (unsigned(counter) = 542) then
                    div <= '1';

                else
                    div <= '0';
```

```
            end if;


        end if;
    end process;


end behavior;
```

Top transmit code

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 03/28/2019 09:16:15 PM
-- Design Name:
-- Module Name: top_transmit - trans_arch
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
```

```
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 top_transmit is
Port (TXD, clk : in std_logic;
      btn : in std_logic_vector(1 downto 0);
      RXD, CTS, RTS : out std_logic);
end top_transmit;

architecture trans_arch of top_transmit is
signal dbnc1, dbnc2, div_sig, ready, send, outrxd  : std_logic;
signal char : std_logic_vector(7 downto 0);

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



   component clock_div
   port(
        clk  : in std_logic;
        div : out std_logic);
    end component;
   component sender
       Port ( reset : in STD_LOGIC;
           clk : in STD_LOGIC;
           ce : in STD_LOGIC;
           send : out STD_LOGIC;
           ready : in std_logic;
           btn : in std_logic;
           char : out STD_LOGIC_VECTOR (7 downto 0));
 end component;
      component  uart
    port (

    clk, en, send, rx, rst      : in std_logic;
    charSend                   : in std_logic_vector (7 downto 0);
    ready, tx, newChar          : out std_logic;
    charRec                    : out std_logic_vector (7 downto 0)
);
end component;



begin
RTS<='0';
CTS<='0';
    u1 : debouncer
        port map (btn => btn(0),
                clk => clk,
                dbnc => dbnc1);

    u2 : debouncer
       port map (btn => btn(1),
                clk => clk,
                dbnc => dbnc2);
    u3 : clock_div
       port map (clk => clk,
                div=> div_sig);

    u4 : sender
       port map ( clk => clk,
               ce=> div_sig,
               ready => ready,
               send=>send,
                 char=> char,
                 reset =>dbnc1,
                 btn=>dbnc2);
     u5 : uart
     port map(send => send,
              rst=>dbnc1,
              clk => clk,
              en=> div_sig,
              rx=>TXD,
              charsend=>char,
              tx=> outrxd);

RXD<=outrxd;
```

end trans_arch;

XDC File:

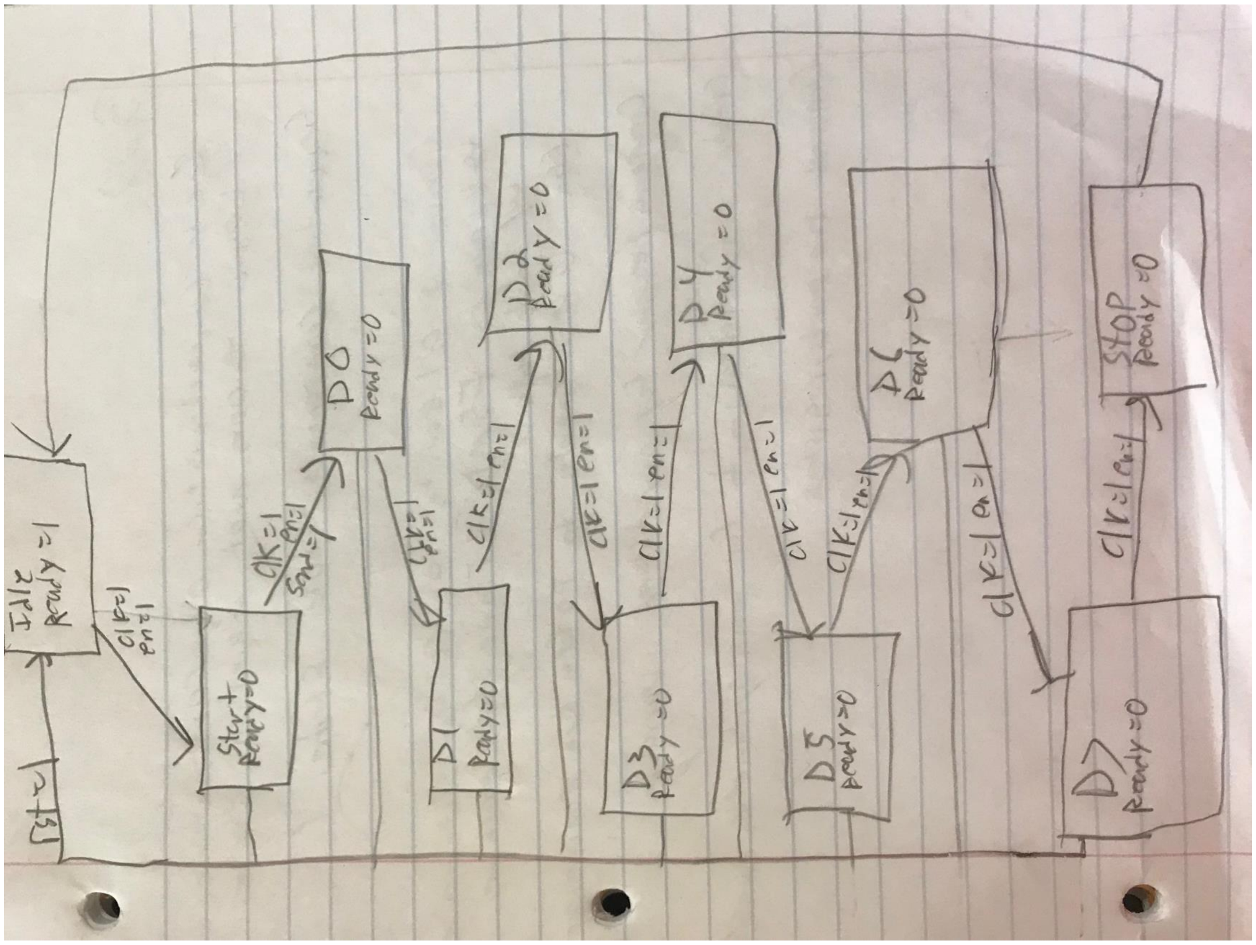
```
## This file is a general .xdc for the ZYBO Rev B board

## To use it in a project:

## - uncomment the lines corresponding to used pins

## - rename the used signals according to the project



#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 }];



##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
```

```
##Pmod Header JB

set_property -dict { PACKAGE_PIN T20   IOSTANDARD LVCMOS33 } [get_ports { RTS }];
#IO_L15P_T2_DQS_34 Sch=JB1_p

set_property -dict { PACKAGE_PIN U20   IOSTANDARD LVCMOS33 } [get_ports { RXD }];
#IO_L15N_T2_DQS_34 Sch=JB1_N

set_property -dict { PACKAGE_PIN V20   IOSTANDARD LVCMOS33 } [get_ports { TXD }]; #IO_L16P_T2_34
Sch=JB2_P

set_property -dict { PACKAGE_PIN W20   IOSTANDARD LVCMOS33 } [get_ports { CTS ] }];
#IO_L16N_T2_34 Sch=JB2_N

#set_property -dict { PACKAGE_PIN Y18   IOSTANDARD LVCMOS33 } [get_ports { jb_p[2] }];
#IO_L17P_T2_34 Sch=JB3_P

#set_property -dict { PACKAGE_PIN Y19   IOSTANDARD LVCMOS33 } [get_ports { jb_n[2] }];
#IO_L17N_T2_34 Sch=JB3_N

#set_property -dict { PACKAGE_PIN W18   IOSTANDARD LVCMOS33 } [get_ports { jb_p[3] }];
#IO_L22P_T3_34 Sch=JB4_P

#set_property -dict { PACKAGE_PIN W19   IOSTANDARD LVCMOS33 } [get_ports { jb_n[3] }];
#IO_L22N_T3_34 Sch=JB4_N
```

This lab required us to use PMOD header JB, two buttons and a clock signal.

*The files pre written for us in this lab (ie the uart.vhd, uart_rx, and uart_tb) were not changed and therefore not included in this lab report

Discussion: I still do not know why nothing would print to the termite console when I pressed btn(1). Feedback here would be greatly appreciated.


Idle
Ready = 1
clk=1
en=1
Start
Ready=0
clk=1
en=1
send=1
D0
Ready=0
clk=1
en=1
D1
Ready=0
clk=1 en=1
D2
Ready=0
clk=1 en=1
D3
Ready=0
clk=1 en=1
D4
Ready=0
clk=1 en=1
D5
Ready=0
clk=1 en=1
D6
Ready=0
clk=1 en=1
D7
Ready=0
clk=1 en=1
STOP
Ready=0