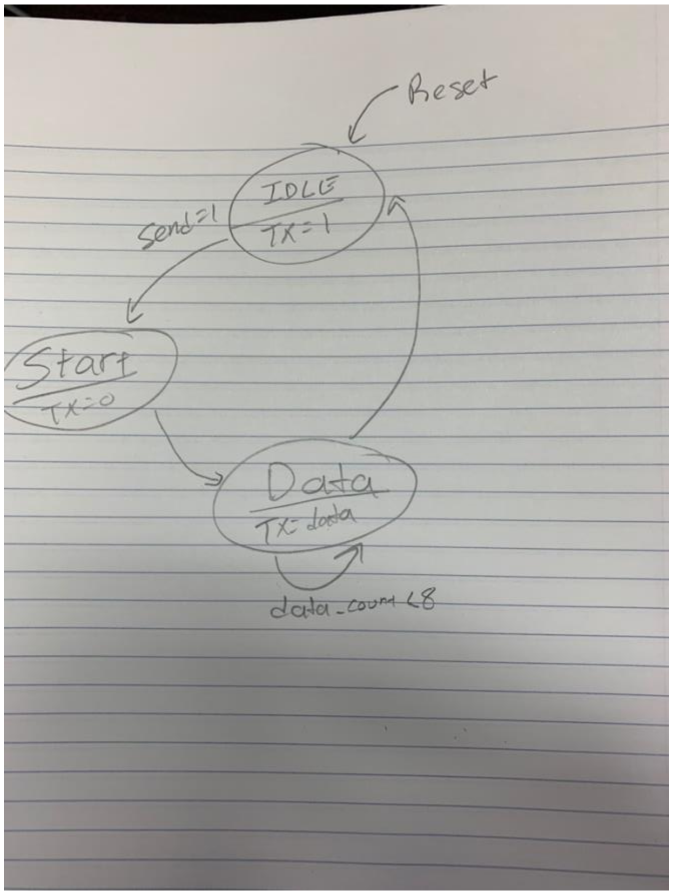
Lab 3 – Where no clock has gone before

Embedded Systems
Lab Report 2
Anthony Lau
March 7, 2019

Table of Contents

PreLab:



Purpose:

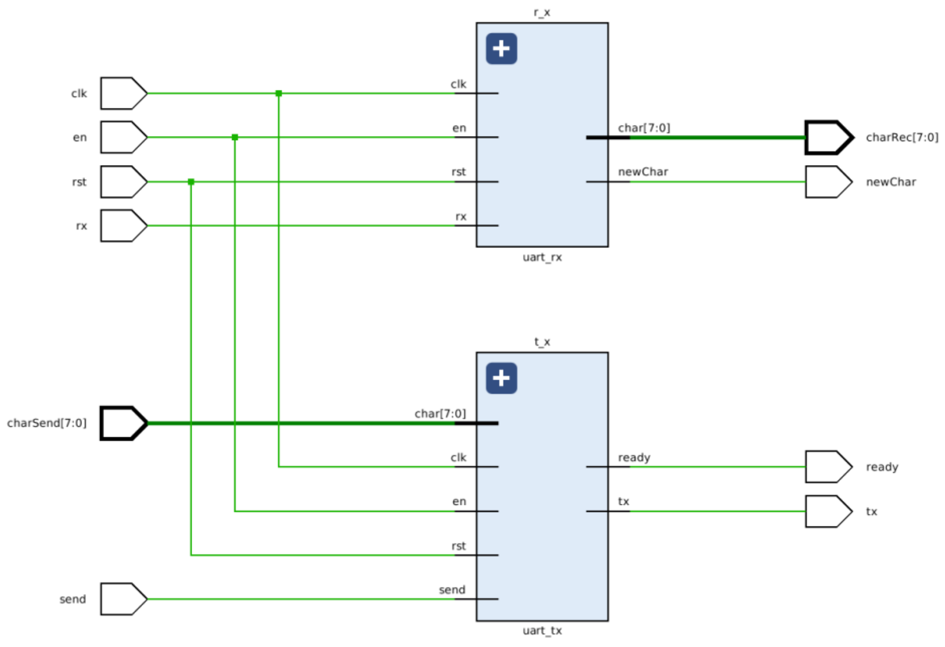
The purpose of this lab is to learn about serial communication which is very popular for data transmission. The lab calls for us to create an FSM design of the RS232 universal asynchronous receiver transmitter. It only uses two wires, one to send and one to receive. In order for both to work, the baud rate of both have to be the same

We can rebuild him

Theory:

RS232 is a serial communication protocol that defines how data is transmitted or received. When a signal is idle, the line is held high and begins to transmit data one the line goes low. The line of bits that are being transmitted are shifted down by LSB first. In this part we are to create the transmit FSM with 0 parity bit and 1 stop bit in order to send 8 bits of data. The tx and given rx files are combines to a whole uart to test data transmission

Schematic:



Design:

Transmitter:

```
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 , en , send , rst : in std_logic ;
char : in std_logic_vector (7 downto 0) ;
ready , tx : out std_logic
) ;
end uart_tx ;
architecture Behavioral of uart_tx is

type state is (idle, start, data);
signal currentState: state := idle;
signal data_input: std_logic_vector(7 downto 0);
signal counter: integer := 0;

begin
process(clk)
    begin


    if(rising_edge(clk)) then
        if(rst = '1') then
            currentstate <= idle;
            data_input <= (others => '0');
            counter <= 0;
            ready <= '1';
            tx <= '1';
        elsif(en = '1') then
            case currentState is
                when idle =>
                        if(send = '1') then
                            data_input <= char;
                            tx <= '0';
                            ready <= '0';
                            currentstate <= start;
                        else
                        ready <= '1';
                        tx <= '1';
                        end if;
                when start =>
                    tx <= data_input(0);
                    data_input <= '0' & data_input(7 downto 1);
                    counter <= 0;
                    currentstate <= data;
                when data =>
                    if(counter < 7) then
                        tx <= data_input(0);
                        counter <= counter + 1;
                        currentstate <= data;
                        data_input <= '0' & data_input(7 downto 1);
                    else
                        tx <= '1';
                        currentState <= idle;
                    end if;
                when others =>
                    currentstate <= idle;
                end case;
        end if;
    end if;
end process;
end Behavioral;
```

Receiver:

```vhdl
--
-- written by Gregory Leonberg
-- fall 2017
--
----------------------------------------------------------------------------

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


entity uart_rx is
    port (
    clk, en, rx, rst    : in std_logic;
    newChar             : out std_logic;
    char                : out std_logic_vector (7 downto 0)
);
end uart_rx;

architecture fsm of uart_rx is

    -- state type enumeration and state variable
    type state is (idle, start, data);
    signal curr : state := idle;

    -- shift register to read data in
    signal d : std_logic_vector (7 downto 0) := (others => '0');

    -- counter for data state
    signal count : std_logic_vector(2 downto 0) := (others => '0');

    -- double flop rx plus 2 extra samples to take majority vote of 3
    -- majority vote of samples helps mitigate noise on line
    signal inshift : std_logic_vector(3 downto 0) := (others => '0');
    signal maj : std_logic := '0';

begin

    -- double flop input to fix potential metastability
    -- plus 2 extra samples to take majority vote of 3 inputs (oversampling)
    -- majority vote of samples helps mitigate noise on line
    process(clk) begin
        if rising_edge(clk) then
            inshift <= inshift(2 downto 0) & rx;
        end if;
    end process;

    -- majority vote of 3 samples (oversampling)
    -- majority vote of samples helps mitigate noise on line
    process(inshift)
    begin
        if (inshift(3) = '1' and inshift(2) = '1' and inshift(1) = '1') or
           (inshift(3) = '1' and inshift(2) = '1') or
           (inshift(2) = '1' and inshift(1) = '1') or
           (inshift(3) = '1' and inshift(1) = '1') then
                maj <= '1';
        else
            maj <= '0';
        end if;
    end process;

    -- FSM process (single process implementation)
    process(clk) begin
    if rising_edge(clk) then

        -- resets the state machine and its outputs
        if rst = '1' then

            curr <= idle;
            d <= (others => '0');
            count <= (others => '0');
            newChar <= '0';

        -- usual operation
        elsif en = '1' then
            case curr is

                when idle =>
                    newChar <= '0';
                    if maj = '0' then
                        curr <= start;
                    end if;

                when start =>
                    d <= maj & d(7 downto 1);
                    count <= (others => '0');
                    curr <= data;

                when data =>
                    if unsigned(count) < 7 then
                        d <= maj & d(7 downto 1);
                        count <= std_logic_vector(unsigned(count) + 1);
                    elsif maj <= '1' then
                        curr <= idle;
                        newChar <= '1';
                        char <= d;
                    else
                        curr <= idle;
                    end if;

                when others =>
                    curr <= idle;

            end case;
```

```
            end if;

        end if;
        end process;

    end fsm;
```

Uart:

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

entity uart is
    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 uart;

architecture structural of uart is
    component uart_tx port
    (
        clk, en, send, rst   : in std_logic;
        char                 : in std_logic_vector (7 downto 0);
        ready, tx            : out std_logic
    );
    end component;

    component uart_rx port
    (
        clk, en, rx, rst     : in std_logic;
        newChar              : out std_logic;
        char                 : out std_logic_vector (7 downto 0)
    );
    end component;

begin

    r_x: uart_rx port map(
        clk => clk,
        en => en,
        rx => rx,
        rst => rst,
        newChar => newChar,
        char => charRec);

    t_x: uart_tx port map(
        clk => clk,
```

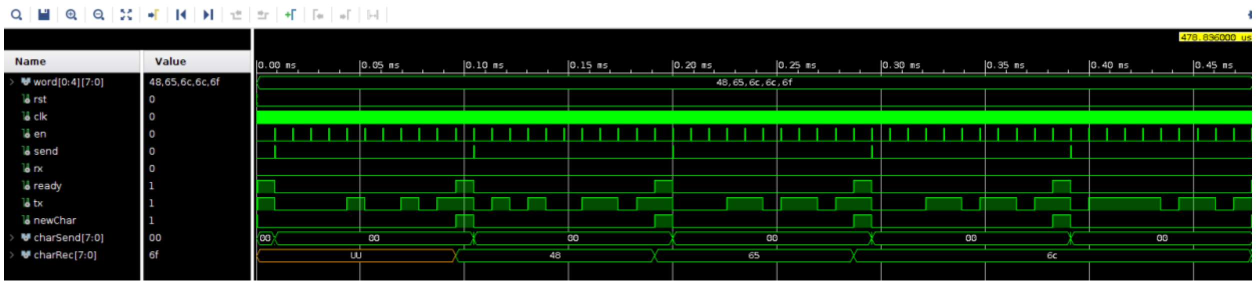
```vhdl
        en => en,
        send => send,
        rst => rst,
        char => charSend,
        ready => ready,
        tx => tx);

end structural;

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use IEEE.std_logic_arith.all;

entity sender is
port (clk, clk_en, rst, ready, btn: in std_logic;
      char: out std_logic_vector(7 downto 0);
      send: out std_logic);

end sender;

architecture behavioral of sender is

type str is array(0 to 4) of std_logic_vector(7 downto 0);
signal word: str := (x"61", x"6C", x"37", x"35", x"31");
signal i: integer := 0;
signal final : std_logic_vector(7 downto 0);

type state is (idle, BusyA, BusyB, BusyC);
signal send_state : state := idle;

begin
    process(clk)
    begin
        if(clk_en = '1') then
            if(rising_edge(clk)) then
                if(rst = '1') then
                    i <= 0;
                    char <= (others => '0');
                    send_state <= idle;
                else
                    case send_state is
                    when idle =>
                        if (ready = '1' and btn = '1') then
                            if(i < 6) then
                                send <= '1';
                                char <= word(i);
                                send_state <= BusyA;
                                i <= i + 1;
                            else
                                i <= 0;
                                send_state <= idle;
                            end if;
                        end if;
                    when BusyA =>
                        send_state <= BusyB;

                    when BusyB =>
                        send <= '0';
                        send_state <= BusyC;

                    when busyC =>
                        if(ready = '1' and btn = '0') then
                            send_state <= idle;
                        else
                            send_state <= BusyC;
                        end if;
                    end case;
                end if;
            end if;
        end if;
    end process;
end behavioral;
```

Test:

```
--
-- written by Gregory Leonberg
-- fall 2017
--
---------------------------------------------------------------------------

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

entity uart_tb is
end uart_tb;

architecture tb of uart_tb is

    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;

    type str is array (0 to 4) of std_logic_vector(7 downto 0);
    signal word : str := (x"48", x"65", x"6C", x"6C", x"6F");

    signal rst : std_logic := '0';
    signal clk, en, send, rx, ready, tx, newChar : std_logic := '0';
```

```vhdl
    signal charSend, charRec : std_logic_vector(7 downto 0) := (others =>
'0');

begin

    -- the sender UART
    dut: uart port map(
        clk => clk,
        en => en,
        send => send,
        rx => tx,
        rst => rst,
        charSend => charSend,
        ready => ready,
        tx => tx,
        newChar => newChar,
        charRec => charRec);


    -- clock process @125 MHz
    process begin
        clk <= '0';
        wait for 4 ns;
        clk <= '1';
        wait for 4 ns;
    end process;

    -- en process @ 125 MHz / 1085 = ~115200 Hz
    process begin
        en <= '0';
        wait for 8680 ns;
        en <= '1';
        wait for 8 ns;
    end process;

    -- signal stimulation process
    process begin

        rst <= '1';
        wait for 100 ns;
        rst <= '0';
        wait for 100 ns;

        for index in 0 to 4 loop
            wait until ready = '1' and en = '1';
            charSend <= word(index);
            send <= '1';
            wait for 200 ns;
            charSend <= (others => '0');
            send <= '0';
            wait until ready = '1' and en = '1' and newChar = '1';

            if charRec /= word(index) then
```

```
                report "Send/Receive MISMATCH at time: " & time'image(now) &
                lf & "expected: " &
                integer'image(to_integer(unsigned(word(index)))) &
                lf & "received: " &
integer'image(to_integer(unsigned(charRec)))
                severity ERROR;
            end if;

        end loop;

        wait for 1000 ns;
        report "End of testbench" severity FAILURE;

    end process;

end tb;
```



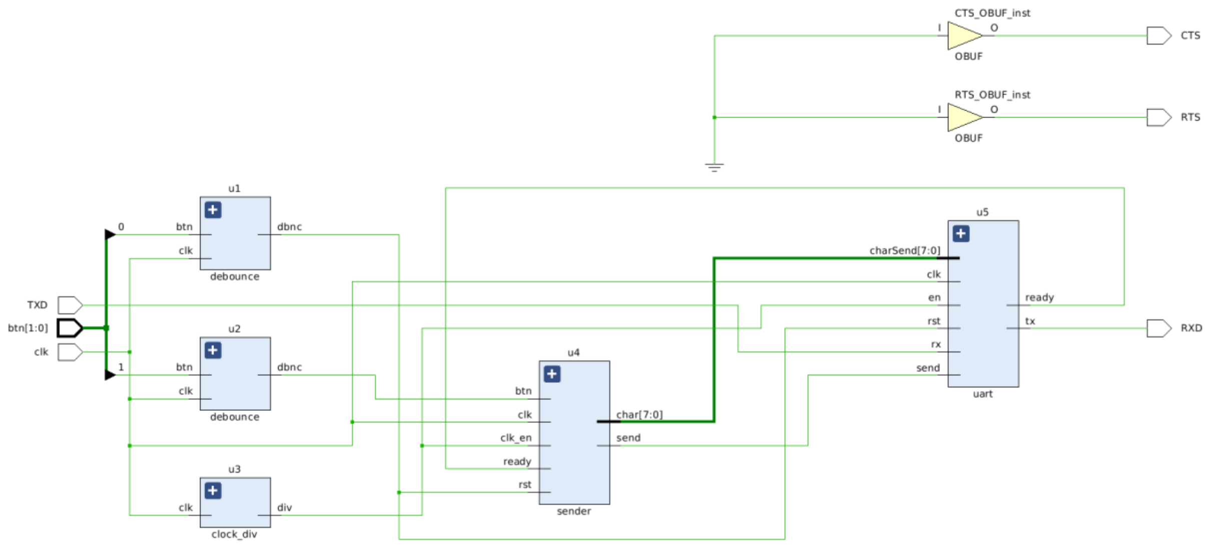
Name	Value
word[0:4][7:0]	48,65,6c,6c,6f
rst	0
clk	0
en	0
send	0
rx	0
ready	1
tx	1
newChar	1
charSend[7:0]	00
charRec[7:0]	6f

We have the technology

Theory:

The aim of this part of the lab was to actually test out transmitting data to a computer. We sent out netids in ascii through a sender FSM each character of the netid is represented by 8 bits.

Schematic:



Design:

Sender:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity sender is
port (clk, clk_en, rst, ready, btn: in std_logic;
      char: out std_logic_vector(7 downto 0);
      send: out std_logic);

end sender;

architecture behavioral of sender is

type str is array(0 to 4) of std_logic_vector(7 downto 0);
signal word: str := (x”61”, x”6C”, x”37”, x”35”, x”31”);
signal i: std_logic_vector(2 downto 0);
signal final : std_logic_vector(7 downto 0);

type state is (idle, BusyA, BusyB, BusyC);
signal send_state : state := idle;

begin
    process(clk)
    begin
        if(rising_edge(clk)) then
            if(rst = ‘1’) then
                I <= (others => ‘0’);
                char <= (others => ‘0’);
                send_state <= idle;
            if(clk_en = '1') then
                    case send_state is
                    when idle =>
                        if (ready = '1' and btn = '1') then
                            if(unsigned(i) < 5) then
                                send <= '1';
                                char <= word(to_integer(unsigned(i)));
                                I <= std_logic_vector(unsigned(i) + 1);
                                send_state <= BusyA;
                            else
                                I <= (others => '0');
                                send_state <= idle;
                            end if;
                        end if;
                    when BusyA =>
                        send_state <= BusyB;

                    when BusyB =>
                        send <= '0';
                        send_state <= BusyC;

                    when busyC =>
                        if(ready = '1' and btn = '0') then
                            send_state <= idle;
                        else
                            send_state <= BusyC;
                        end if;
                    end case;
                end if;
            end if;
        end if;
    end process;
end behavioral;
```

Clock Divider:

```
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 cnt1 of clock_div is
 signal count : std_logic_vector (15 downto 0) := (others => '0');

 begin
 --div <= enable;

 process(clk)
 begin
    if rising_edge(clk) then
        if (unsigned(count) < 1085) then
            count <= std_logic_vector( unsigned(count) + 1 );
             div <= '1';
        else
            count <= (others => '0');
            div <= '0';

        end if;
            if(unsigned(count) = 543) then
                div <= '1';
            else
                div <= '0';
            end if;
    end if;
end process;
end cnt1;
```

Debounce:

```
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 debounce is
Port (clk, btn: in std_logic;
      dbnc: out std_logic);
end debounce;

architecture Behavioral of debounce is

signal counter: std_logic_vector(21 downto 0);
signal count_set: std_logic;
signal shift_register: std_logic_vector(1 downto 0);

begin
    process(clk)
    begin
    count_set <= shift_register(1) xor shift_register(0);
```

```
        if(rising_edge(clk)) then
            shift_register(0) <= btn;
            shift_register(1) <= shift_register(0);
                if(count_set = '1') then
                    counter <= (others => '0');
                elsif(counter(21) = '0') then
                    counter <= std_logic_vector(unsigned(counter) + 1);
                else
                    dbnc <= shift_register(1);
                end if;
        end if;
     end process;
end Behavioral;
```

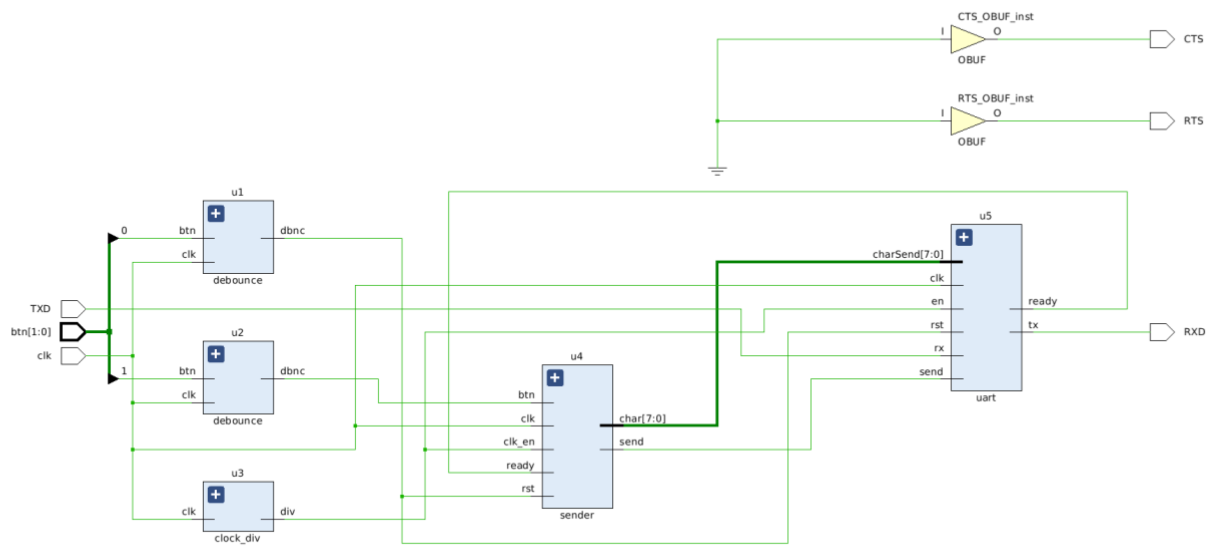
Final Top Level Design

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

architecture Behavioral of finalTopLevel is
signal output : std_logic_vector(2 downto 0);
signal new_character: std_logic_vector(7 downto 0);
signal new_send: std_logic;
signal new_ready: std_logic;
signal new_tx: std_logic;
---------------UART Component----------------------
component uart is
    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;
```

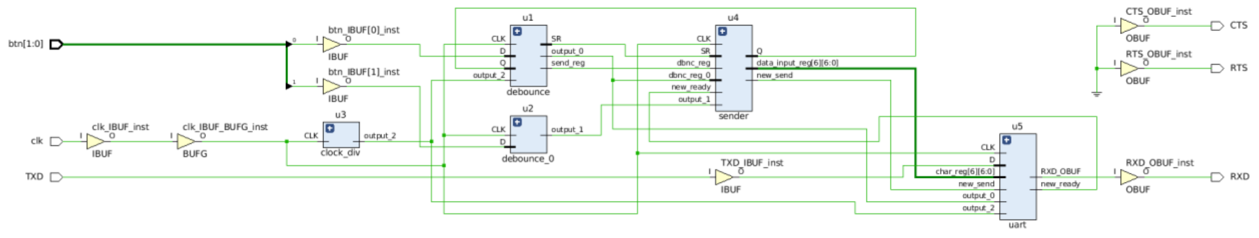
```vhdl
-----------------Debounce component--------------------
component debounce is
Port (clk, btn: in std_logic;
      dbnc: out std_logic);
end component;

--------------------Clock Divider Component-------------------
component clock_div is
Port (clk: in std_logic;
      div: out std_logic);
end component;

---------------------Sender Component-------------------------
component sender is
port (clk, clk_en, rst, ready, btn: in std_logic;
      char: out std_logic_vector(7 downto 0);
      send: out std_logic);
end component;

begin
CTS <= '0';
RTS <= '0';

    u1: debounce
    port map(clk => clk,
             btn => btn(0),
             dbnc => output(0));

    u2: debounce
    port map(clk => clk,
             btn => btn(1),
             dbnc => output(1));

    u3: clock_div
    port map(clk => clk,
             div => output(2));

    u4: sender
    port map(btn => output(1),
             clk => clk,
             clk_en => output(2),
             ready => new_ready,
             rst => output(0),
             char(7 downto 0) => new_character(7 downto 0),
             send => new_send);

    u5: uart
    port map(charSend => new_character,
             clk => clk,
             en => output(2),
             rst => output(0),
             rx => TXD,
             send => new_send,
             ready => new_ready,
             tx => RXD);

end Behavioral;
```

Implementation:

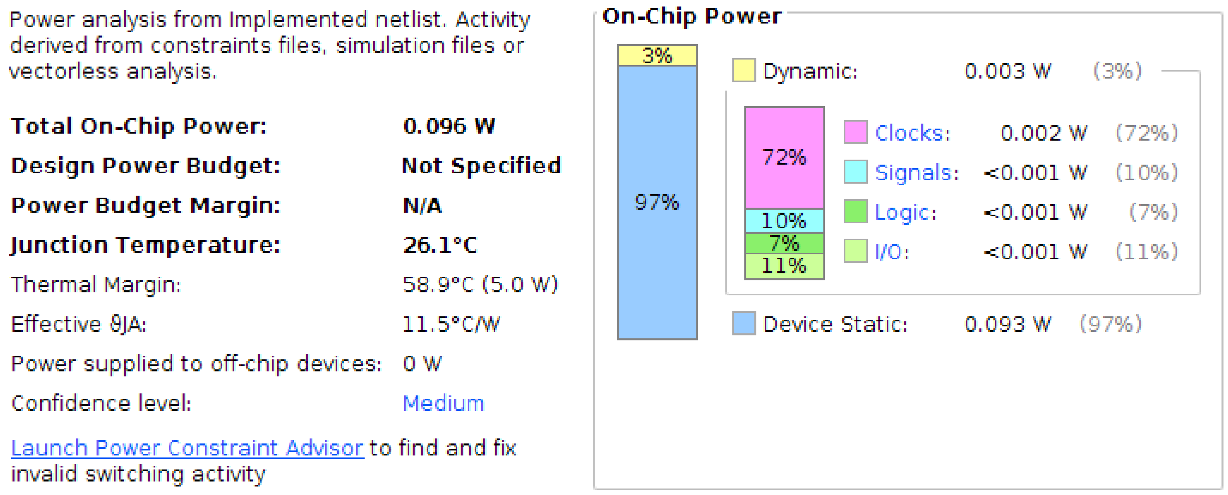
Elaboration Schematic



Synthesis Schematic



Project Summary



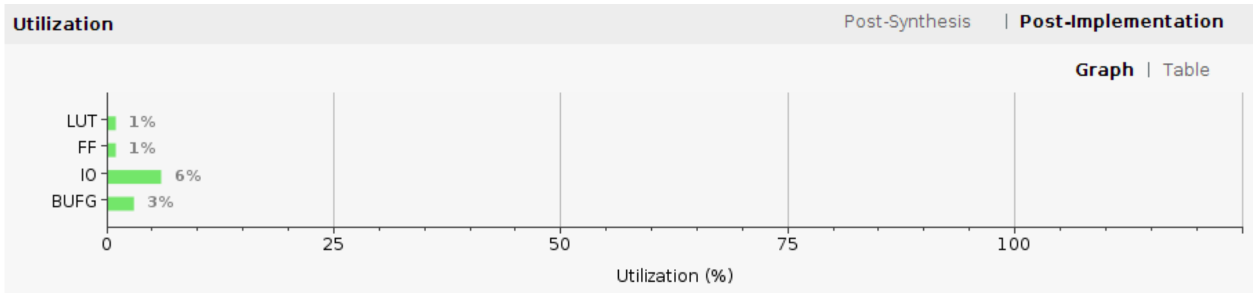
Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.096 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:	0.003 W	(3%)	
	Clocks:	0.002 W	(72%)
	Signals:	<0.001 W	(10%)
	Logic:	<0.001 W	(7%)
	I/O:	<0.001 W	(11%)
Device Static:	0.093 W	(97%)	



XDC

```
## 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 }];


##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
```

```
##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


##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


##I2S Audio Codec
#set_property -dict { PACKAGE_PIN K18   IOSTANDARD LVCMOS33 } [get_ports
ac_bclk]; #IO_L12N_T1_MRCC_35 Sch=AC_BCLK
#set_property -dict { PACKAGE_PIN T19   IOSTANDARD LVCMOS33 } [get_ports
ac_mclk]; #IO_25_34 Sch=AC_MCLK
#set_property -dict { PACKAGE_PIN P18   IOSTANDARD LVCMOS33 } [get_ports
ac_muten]; #IO_L23N_T3_34 Sch=AC_MUTEN
#set_property -dict { PACKAGE_PIN M17   IOSTANDARD LVCMOS33 } [get_ports
ac_pbdat]; #IO_L8P_T1_AD10P_35 Sch=AC_PBDAT
#set_property -dict { PACKAGE_PIN L17   IOSTANDARD LVCMOS33 } [get_ports
ac_pblrc]; #IO_L11N_T1_SRCC_35 Sch=AC_PBLRC
#set_property -dict { PACKAGE_PIN K17   IOSTANDARD LVCMOS33 } [get_ports
ac_recdat]; #IO_L12P_T1_MRCC_35 Sch=AC_RECDAT
#set_property -dict { PACKAGE_PIN M18   IOSTANDARD LVCMOS33 } [get_ports
ac_reclrc]; #IO_L8N_T1_AD10N_35 Sch=AC_RECLRC


##Audio Codec/external EEPROM IIC bus
#set_property -dict { PACKAGE_PIN N18   IOSTANDARD LVCMOS33 } [get_ports
ac_scl]; #IO_L13P_T2_MRCC_34 Sch=AC_SCL
#set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 } [get_ports
ac_sda]; #IO_L23P_T3_34 Sch=AC_SDA
```

```
##Additional Ethernet signals
#set_property -dict { PACKAGE_PIN F16   IOSTANDARD LVCMOS33 } [get_ports
eth_int_b]; #IO_L6P_T0_35 Sch=ETH_INT_B
#set_property -dict { PACKAGE_PIN E17   IOSTANDARD LVCMOS33 } [get_ports
eth_rst_b]; #IO_L3P_T0_DQS_AD1P_35 Sch=ETH_RST_B


##HDMI Signals
#set_property -dict { PACKAGE_PIN H17   IOSTANDARD TMDS_33 } [get_ports
hdmi_clk_n]; #IO_L13N_T2_MRCC_35 Sch=HDMI_CLK_N
#set_property -dict { PACKAGE_PIN H16   IOSTANDARD TMDS_33 } [get_ports
hdmi_clk_p]; #IO_L13P_T2_MRCC_35 Sch=HDMI_CLK_P
#set_property -dict { PACKAGE_PIN D20   IOSTANDARD TMDS_33 } [get_ports {
hdmi_d_n[0] }]; #IO_L4N_T0_35 Sch=HDMI_D0_N
#set_property -dict { PACKAGE_PIN D19   IOSTANDARD TMDS_33 } [get_ports {
hdmi_d_p[0] }]; #IO_L4P_T0_35 Sch=HDMI_D0_P
#set_property -dict { PACKAGE_PIN B20   IOSTANDARD TMDS_33 } [get_ports {
hdmi_d_n[1] }]; #IO_L1N_T0_AD0N_35 Sch=HDMI_D1_N
#set_property -dict { PACKAGE_PIN C20   IOSTANDARD TMDS_33 } [get_ports {
hdmi_d_p[1] }]; #IO_L1P_T0_AD0P_35 Sch=HDMI_D1_P
#set_property -dict { PACKAGE_PIN A20   IOSTANDARD TMDS_33 } [get_ports {
hdmi_d_n[2] }]; #IO_L2N_T0_AD8N_35 Sch=HDMI_D2_N
#set_property -dict { PACKAGE_PIN B19   IOSTANDARD TMDS_33 } [get_ports {
hdmi_d_p[2] }]; #IO_L2P_T0_AD8P_35 Sch=HDMI_D2_P
#set_property -dict { PACKAGE_PIN E19   IOSTANDARD LVCMOS33 } [get_ports
hdmi_cec]; #IO_L5N_T0_AD9N_35 Sch=HDMI_CEC
#set_property -dict { PACKAGE_PIN E18   IOSTANDARD LVCMOS33 } [get_ports
hdmi_hpd]; #IO_L5P_T0_AD9P_35 Sch=HDMI_HPD
#set_property -dict { PACKAGE_PIN F17   IOSTANDARD LVCMOS33 } [get_ports
hdmi_out_en]; #IO_L6N_T0_VREF_35 Sch=HDMI_OUT_EN
#set_property -dict { PACKAGE_PIN G17   IOSTANDARD LVCMOS33 } [get_ports
hdmi_scl]; #IO_L16P_T2_35 Sch=HDMI_SCL
#set_property -dict { PACKAGE_PIN G18   IOSTANDARD LVCMOS33 } [get_ports
hdmi_sda]; #IO_L16N_T2_35 Sch=HDMI_SDA


##Pmod Header JA (XADC)
#set_property -dict { PACKAGE_PIN N15   IOSTANDARD LVCMOS33 } [get_ports {
ja_p[0] }]; #IO_L21P_T3_DQS_AD14P_35 Sch=JA1_R_p
#set_property -dict { PACKAGE_PIN L14   IOSTANDARD LVCMOS33 } [get_ports {
ja_p[1] }]; #IO_L22P_T3_AD7P_35 Sch=JA2_R_P
#set_property -dict { PACKAGE_PIN K16   IOSTANDARD LVCMOS33 } [get_ports {
ja_p[2] }]; #IO_L24P_T3_AD15P_35 Sch=JA3_R_P
```

#set_property -dict { PACKAGE_PIN K14   IOSTANDARD LVCMOS33 } [get_ports { ja_p[3] }]; #IO_L20P_T3_AD6P_35 Sch=JA4_R_P
#set_property -dict { PACKAGE_PIN N16   IOSTANDARD LVCMOS33 } [get_ports { ja_n[0] }]; #IO_L21N_T3_DQS_AD14N_35 Sch=JA1_R_N
#set_property -dict { PACKAGE_PIN L15   IOSTANDARD LVCMOS33 } [get_ports { ja_n[1] }]; #IO_L22N_T3_AD7N_35 Sch=JA2_R_N
#set_property -dict { PACKAGE_PIN J16   IOSTANDARD LVCMOS33 } [get_ports { ja_n[2] }]; #IO_L24N_T3_AD15N_35 Sch=JA3_R_N
#set_property -dict { PACKAGE_PIN J14   IOSTANDARD LVCMOS33 } [get_ports { ja_n[3] }]; #IO_L20N_T3_AD6N_35 Sch=JA4_R_N

##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

##Pmod Header JC
#set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports { jc_p[0] }]; #IO_L10P_T1_34 Sch=JC1_P
#set_property -dict { PACKAGE_PIN W15   IOSTANDARD LVCMOS33 } [get_ports { jc_n[0] }]; #IO_L10N_T1_34 Sch=JC1_N
#set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33 } [get_ports { jc_p[1] }]; #IO_L1P_T0_34 Sch=JC2_P
#set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33 } [get_ports { jc_n[1] }]; #IO_L1N_T0_34 Sch=JC2_N
#set_property -dict { PACKAGE_PIN W14   IOSTANDARD LVCMOS33 } [get_ports { jc_p[2] }]; #IO_L8P_T1_34 Sch=JC3_P
#set_property -dict { PACKAGE_PIN Y14   IOSTANDARD LVCMOS33 } [get_ports { jc_n[2] }]; #IO_L8N_T1_34 Sch=JC3_N
#set_property -dict { PACKAGE_PIN T12   IOSTANDARD LVCMOS33 } [get_ports { jc_p[3] }]; #IO_L2P_T0_34 Sch=JC4_P

```
#set_property -dict { PACKAGE_PIN U12   IOSTANDARD LVCMOS33 } [get_ports {
jc_n[3] }]; #IO_L2N_T0_34 Sch=JC4_N


##Pmod Header JD
#set_property -dict { PACKAGE_PIN T14   IOSTANDARD LVCMOS33 } [get_ports {
jd_p[0] }]; #IO_L5P_T0_34 Sch=JD1_P
#set_property -dict { PACKAGE_PIN T15   IOSTANDARD LVCMOS33 } [get_ports {
jd_n[0] }]; #IO_L5N_T0_34 Sch=JD1_N
#set_property -dict { PACKAGE_PIN P14   IOSTANDARD LVCMOS33 } [get_ports {
jd_p[1] }]; #IO_L6P_T0_34 Sch=JD2_P
#set_property -dict { PACKAGE_PIN R14   IOSTANDARD LVCMOS33 } [get_ports {
jd_n[1] }]; #IO_L6N_T0_VREF_34 Sch=JD2_N
#set_property -dict { PACKAGE_PIN U14   IOSTANDARD LVCMOS33 } [get_ports {
jd_p[2] }]; #IO_L11P_T1_SRCC_34 Sch=JD3_P
#set_property -dict { PACKAGE_PIN U15   IOSTANDARD LVCMOS33 } [get_ports {
jd_n[2] }]; #IO_L11N_T1_SRCC_34 Sch=JD3_N
#set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports {
jd_p[3] }]; #IO_L21P_T3_DQS_34 Sch=JD4_P
#set_property -dict { PACKAGE_PIN V18   IOSTANDARD LVCMOS33 } [get_ports {
jd_n[3] }]; #IO_L21N_T3_DQS_34 Sch=JD4_N


##Pmod Header JE
#set_property -dict { PACKAGE_PIN V12   IOSTANDARD LVCMOS33 } [get_ports {
je[0] }]; #IO_L4P_T0_34 Sch=JE1
#set_property -dict { PACKAGE_PIN W16   IOSTANDARD LVCMOS33 } [get_ports {
je[1] }]; #IO_L18N_T2_34 Sch=JE2
#set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports {
je[2] }]; #IO_25_35 Sch=JE3
#set_property -dict { PACKAGE_PIN H15   IOSTANDARD LVCMOS33 } [get_ports {
je[3] }]; #IO_L19P_T3_35 Sch=JE4
#set_property -dict { PACKAGE_PIN V13   IOSTANDARD LVCMOS33 } [get_ports {
je[4] }]; #IO_L3N_T0_DQS_34 Sch=JE7
#set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports {
je[5] }]; #IO_L9N_T1_DQS_34 Sch=JE8
#set_property -dict { PACKAGE_PIN T17   IOSTANDARD LVCMOS33 } [get_ports {
je[6] }]; #IO_L20P_T3_34 Sch=JE9
#set_property -dict { PACKAGE_PIN Y17   IOSTANDARD LVCMOS33 } [get_ports {
je[7] }]; #IO_L7N_T1_34 Sch=JE10


##USB-OTG overcurrent detect pin
#set_property -dict { PACKAGE_PIN U13   IOSTANDARD LVCMOS33 } [get_ports
otg_oc]; #IO_L3P_T0_DQS_PUDC_B_34 Sch=OTG_OC
```

```
##VGA Connector
#set_property -dict { PACKAGE_PIN M19   IOSTANDARD LVCMOS33 } [get_ports {
vga_r[0] }]; #IO_L7P_T1_AD2P_35 Sch=VGA_R1
#set_property -dict { PACKAGE_PIN L20   IOSTANDARD LVCMOS33 } [get_ports {
vga_r[1] }]; #IO_L9N_T1_DQS_AD3N_35 Sch=VGA_R2
#set_property -dict { PACKAGE_PIN J20   IOSTANDARD LVCMOS33 } [get_ports {
vga_r[2] }]; #IO_L17P_T2_AD5P_35 Sch=VGA_R3
#set_property -dict { PACKAGE_PIN G20   IOSTANDARD LVCMOS33 } [get_ports {
vga_r[3] }]; #IO_L18N_T2_AD13N_35 Sch=VGA_R4
#set_property -dict { PACKAGE_PIN F19   IOSTANDARD LVCMOS33 } [get_ports {
vga_r[4] }]; #IO_L15P_T2_DQS_AD12P_35 Sch=VGA_R5
#set_property -dict { PACKAGE_PIN H18   IOSTANDARD LVCMOS33 } [get_ports {
vga_g[0] }]; #IO_L14N_T2_AD4N_SRCC_35 Sch=VGA_G0
#set_property -dict { PACKAGE_PIN N20   IOSTANDARD LVCMOS33 } [get_ports {
vga_g[1] }]; #IO_L14P_T2_SRCC_34 Sch=VGA_G1
#set_property -dict { PACKAGE_PIN L19   IOSTANDARD LVCMOS33 } [get_ports {
vga_g[2] }]; #IO_L9P_T1_DQS_AD3P_35 Sch=VGA_G2
#set_property -dict { PACKAGE_PIN J19   IOSTANDARD LVCMOS33 } [get_ports {
vga_g[3] }]; #IO_L10N_T1_AD11N_35 Sch=VGA_G3
#set_property -dict { PACKAGE_PIN H20   IOSTANDARD LVCMOS33 } [get_ports {
vga_g[4] }]; #IO_L17N_T2_AD5N_35 Sch=VGA_G4
#set_property -dict { PACKAGE_PIN F20   IOSTANDARD LVCMOS33 } [get_ports {
vga_g[5] }]; #IO_L15N_T2_DQS_AD12N_35 Sch=VGA=G5
#set_property -dict { PACKAGE_PIN P20   IOSTANDARD LVCMOS33 } [get_ports {
vga_b[0] }]; #IO_L14N_T2_SRCC_34 Sch=VGA_B1
#set_property -dict { PACKAGE_PIN M20   IOSTANDARD LVCMOS33 } [get_ports {
vga_b[1] }]; #IO_L7N_T1_AD2N_35 Sch=VGA_B2
#set_property -dict { PACKAGE_PIN K19   IOSTANDARD LVCMOS33 } [get_ports {
vga_b[2] }]; #IO_L10P_T1_AD11P_35 Sch=VGA_B3
#set_property -dict { PACKAGE_PIN J18   IOSTANDARD LVCMOS33 } [get_ports {
vga_b[3] }]; #IO_L14P_T2_AD4P_SRCC_35 Sch=VGA_B4
#set_property -dict { PACKAGE_PIN G19   IOSTANDARD LVCMOS33 } [get_ports {
vga_b[4] }]; #IO_L18P_T2_AD13P_35 Sch=VGA_B5
#set_property -dict { PACKAGE_PIN P19   IOSTANDARD LVCMOS33 } [get_ports
vga_hs]; #IO_L13N_T2_MRCC_34 Sch=VGA_HS
#set_property -dict { PACKAGE_PIN R19   IOSTANDARD LVCMOS33 } [get_ports
vga_vs]; #IO_0_34 Sch=VGA_VS
```

I had to uncomment the clock, 2 buttons and the first 4 ports of the JB port.

Dicussion:

In this lab I learned about RS232 serial communication and how it works. Creating the FSM was the main purpose of this lab in order to transmit data. We put everything together in order to send our netid to the computer screen using a uart pmod. Overall, it was a little challenging putting it all together because at first I started getting unknown values displayed on the screen.