Embedded Lab 4

Shannon L. Sabino

November 1, 2017

Contents

Introduction **2**

Part: Controls **2**

Part II: Pixel Pusher **2**

RTL Schematic 3

Discussion 3

Appendix **5**

Part 1 5

vga_ctrl.vhd 5

ctrl_tb.vhd 6

Simulation 7

Part 2 7

pixel_pusher.vhd 7

image_top.vhd 8

Synthesis Schematic 10

Implementation 11

Power and Utilization 11

Constraint File 11

Introduction

Now that Lab 3 has left students primed and ready to create more interfacing systems, we are ready to take the leap into the 1990s and use simple a Video Graphics Array (VGA). VGA is a standard so reliable that I, a luddite who will use my one phone until it causes me physical pain to call someone, rely upon it heavily in my personal life. VGA traditionally used cathode ray tubes (CRT), which fire a focused beam of electrons at a screen while utilizing physics magic to bend the electron beam around the screen. If we use multiple of these path-bending electron beams in different wavelengths (colors) together, we can rapidly scan an entire color image on screen.

Although for most monitors, the *how* of image display has moved away from actual cathode ray tubes, the VGA protocol can still be implemented on any monitor that understands it. We still act as if we are transmitting instructors to a scanning CRT - for example, we take into account the "down times" when the beam would need to sweep back to the other side of the screen without adding to the image.

One thing that we need to prepare for this lab to test our VGA protocol is a sample image to send to the screen. Our excellent TA provided us with a MATLAB script that down samples a 480×480 jpeg image into a file format that Xilinx Vivado can understand. This file format contains the bit information necessary to display the image on a screen via VGA. Vivado takes this file and stores this bits on a ROM on the FPGA for use as we please.

Part: Controls

We begin by writing the "controls" of for the VGA. This consists of nailing down all of the timing. As I alluded to earlier, we treat our VGA as if it is controlling an electron beam that needs time to reset back to the start, like a typewriter. Therefore, we have one counter for the *total* amount of time that a beam would take to complete a horizontal sweep, as well as a counter for vertical sweep completion. However, we only want to output to the screen for certain amount of this time. Therefore, when our two counters are in the "rectangle" that would represent the screen, we use a **vid** signal, which, when high, lets our VGA know it's okay to write to the screen. We additionally need to let our VGA know when our electron beam is resetting - the times when **vid** is low. When the beam is in the process of performing a horizontal newline, we set a signal **hs** to be low, and likewise we set a signal **vs** to be low during a vertical reset. Altogether, our controller will output the two counters, **hcount** and **vcount**, as well as the three earlier mentioned signals, **vid**, **hs**, and **vs**. This controller is driven only by the clock and a clock enable. The testbench and simulation used to test it is shown in the appendix.

Part II: Pixel Pusher

For the second part of this exciting lab, we take the image that I mentioned in the introduction as being stored on the FPGA's ROM, and the controller from above, and create a "pixel pusher" to be the final link in the chain that makes our image appear on the monitor. The pixel pusher uses the **vs** and **hcount** signals from above to find the right pixels by using them to drive an **addr** (address) counter. This counter runs through all of the pixels in the picture ROM's component as they are needed. The picture component outputs **pixel**, which we can break into the three RGB values we need for the monitor. These three RGB values are what we ultimately end up sending to the monitor. The top level circuit is set up as shown below.

RTL Schematic

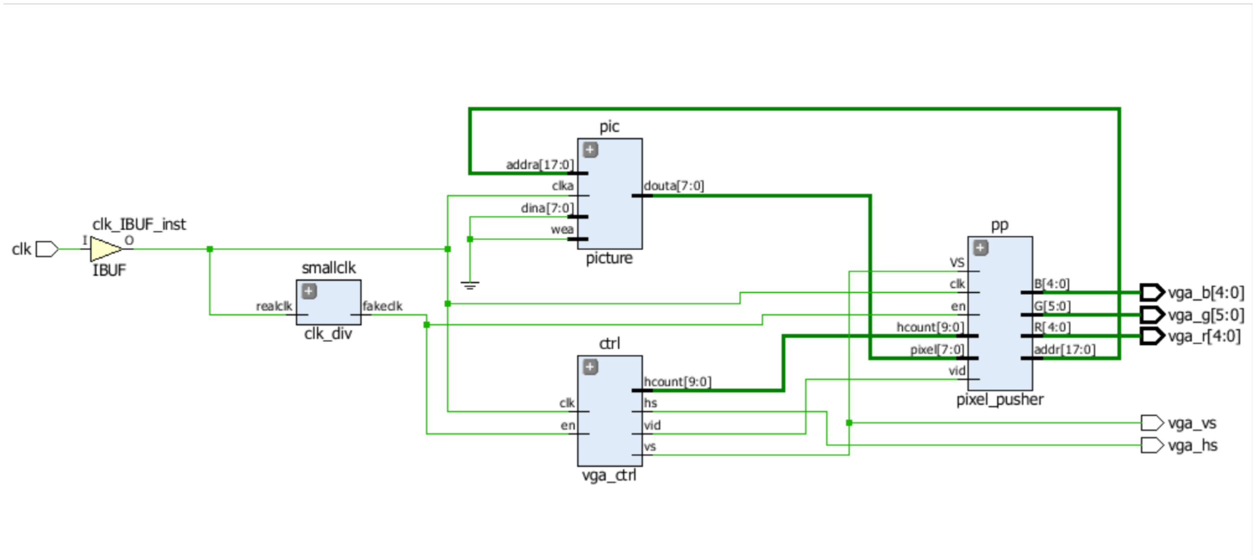


Figure 1: The top-level design.

Discussion

Once it worked, it worked great! I wish it didn't take so long to write the .coe files, I would have loved to have messed around with putting more pictures on the screen. It also must take a ton of memory to create animation. The picture that I put on the screen is below.

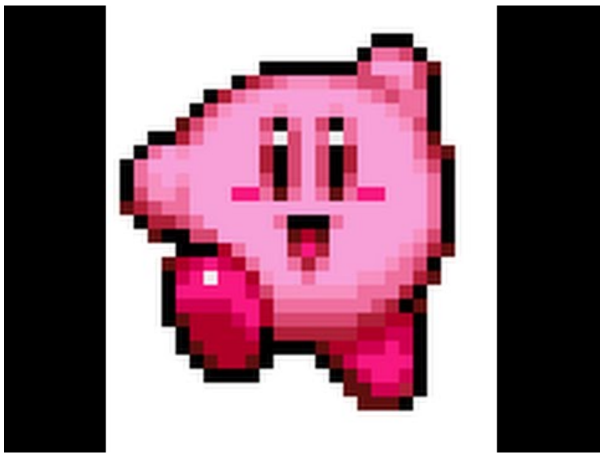


Figure 2: Image I displayed on the screen.

Appendix

Part 1

vga_ctrl.vhd

```vhdl

entity vga_ctrl is
    Port ( clk : in STD_LOGIC := '0';
           en : in STD_LOGIC := '0';
           hcount : out STD_LOGIC_VECTOR (9 downto 0) := (others => '0');
           vcount : out STD_LOGIC_VECTOR (9 downto 0) := (others => '0');
           vid : out STD_LOGIC := '0';
           hs : out STD_LOGIC := '0';
           vs : out STD_LOGIC := '0');
end vga_ctrl;

architecture Behavioral of vga_ctrl is
signal newline : std_logic := '0'; -- self commenting
--internal connections
signal horin, vorin : STD_LOGIC_VECTOR (9 downto 0) := (others => '0'); --
    hcount, vcount inside the component

begin

    process(clk) begin
        if rising_edge(clk) AND en='1' then
            if unsigned(horin)=799 then -- end of row
                newline <= '1';
                horin <= (others => '0'); -- reset
            else
                horin <= std_logic_vector(unsigned(horin)+1); --hcount
    increment
            end if;
            if newline='1' then -- start new row
                if unsigned(vorin) = 525 then
                    vorin <= (others => '0');
                else
                    vorin <= std_logic_vector(unsigned(vorin)+1); --vcount
    increment
                end if;
                newline <= '0';
            end if;
            if unsigned(horin)<640 then -- better way to implement this
    fucking distaster?
                hs <= '1'; -- hcount on screen
                if unsigned(vorin)<480 then
                    vid <= '1'; -- only time on the screen, output stuff
                    vs <= '1'; --vcount on screen
                else
                    vid <= '0';
                    vs <= '0';
                end if;
            else
                vid <= '0';
                hs <= '0';
                if unsigned(vorin)<480 then -- super fucking inelegant
                    vs <= '1';
                else
                    vs <= '0';
                end if;
            end if;

                    -- signal assignments
            hcount <= horin;
            vcount <= vorin;
        end if;


    end process;

end Behavioral;
```

ctrl_tb.vhd

```

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

architecture Behavioral of ctrl_tb is
component vga_ctrl -- instantiate
    Port ( clk : in STD_LOGIC := '0';
           en : in STD_LOGIC := '0';
           hcount : out STD_LOGIC_VECTOR (9 downto 0) := (others => '0');
           vcount : out STD_LOGIC_VECTOR (9 downto 0) := (others => '0');
           vid : out STD_LOGIC := '0';
           hs : out STD_LOGIC := '0';
           vs : out STD_LOGIC := '0');
end component;
--other signals
signal clk : std_logic := '0';
signal hcount, vcount : STD_LOGIC_VECTOR (9 downto 0) := (others => '0');
signal vid, hs, vs : std_logic := '0';
begin

    dut: vga_ctrl port map( -- I guess this is technically the instantiation
                clk => clk,
                en => clk,
                hcount => hcount,
                vcount => vcount,
                vid => vid,
                hs => hs,
                vs => vs);

         process begin
            clk <= '0';
            wait for 500 ps;
            clk <= '1';
            wait for 500 ps;
         end process;

end Behavioral;
```

Simulation

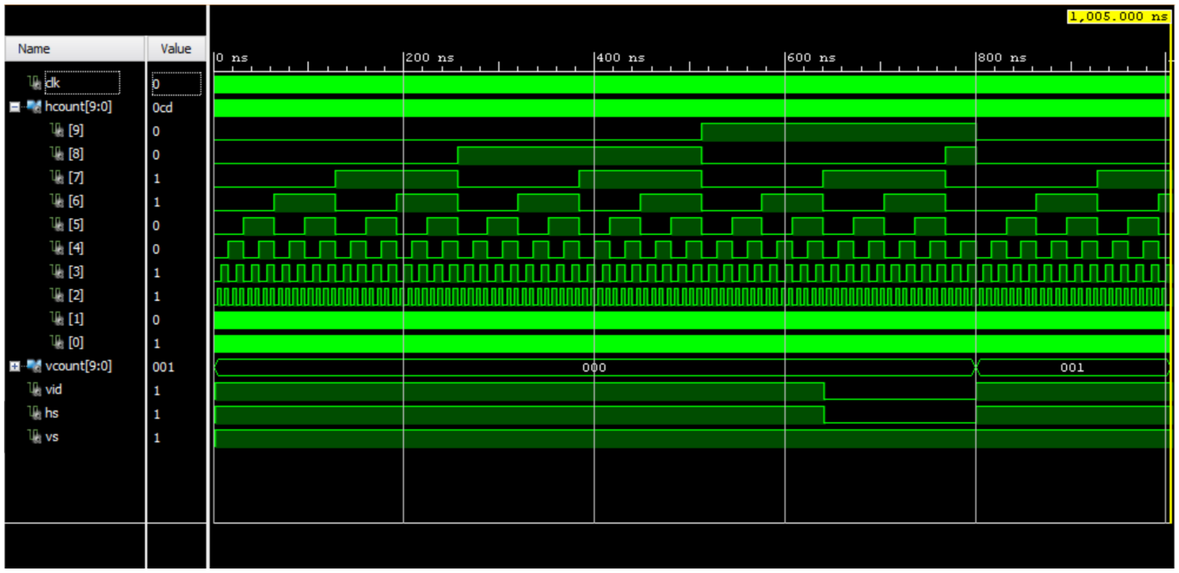


Figure 3: Testbench simulation of the VGA controller.

Part 2

pixel_pusher.vhd

```

entity pixel_pusher is
    Port ( clk : in STD_LOGIC;
           en : in STD_LOGIC;
           VS : in STD_LOGIC;
           pixel : in STD_LOGIC_VECTOR (7 downto 0);
           hcount : in STD_LOGIC_VECTOR (9 downto 0);
           vid : in STD_LOGIC;
           R : out STD_LOGIC_VECTOR (4 downto 0) := (others => '0');
           B : out STD_LOGIC_VECTOR (4 downto 0) := (others => '0');
           G : out STD_LOGIC_VECTOR (5 downto 0) := (others => '0');
           addr : out STD_LOGIC_VECTOR (17 downto 0));
end pixel_pusher;

architecture Behavioral of pixel_pusher is
signal countin : std_logic_vector(17 downto 0) := (others => '0'); --
    internal counter
begin
    process(clk) begin
        if rising_edge(clk) AND en='1' then
            if VS='0' then
            --reset
                R <= (others => '0');
                B <= (others => '0');
                G <= (others => '0');
                countin <= (others => '0');
            elsif vid='1' AND unsigned(hcount)<480 then
```

```
                R <= pixel(7 downto 5) & "00";
                G <= pixel(4 downto 2) & "000";
                B <= pixel(1 downto 0) & "000";
                countin <= std_logic_vector(unsigned(countin)+1);
            else
                R <= (others => '0');
                B <= (others => '0');
                G <= (others => '0');
            end if;

            --signal assignment
                    addr <= countin;
        end if;

    end process;
end Behavioral;
```

image_top.vhd

```

entity image_top is
    Port ( clk : in STD_LOGIC;
           vga_r : out STD_LOGIC_VECTOR (4 downto 0);
           vga_g : out STD_LOGIC_VECTOR (5 downto 0);
           vga_b : out STD_LOGIC_VECTOR (4 downto 0);
           vga_hs : out STD_LOGIC;
           vga_vs : out STD_LOGIC);
end image_top;

architecture Behavioral of image_top is
component picture
  PORT (
    clka : IN STD_LOGIC;
    addra : IN STD_LOGIC_VECTOR(17 DOWNTO 0);
    douta : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
  );
END component;
component pixel_pusher
    Port ( clk : in STD_LOGIC;
           en : in STD_LOGIC;
           VS : in STD_LOGIC;
           pixel : in STD_LOGIC_VECTOR (7 downto 0);
           hcount : in STD_LOGIC_VECTOR (9 downto 0);
           vid : in STD_LOGIC;
           R : out STD_LOGIC_VECTOR (4 downto 0) := (others => '0');
           B : out STD_LOGIC_VECTOR (4 downto 0) := (others => '0');
           G : out STD_LOGIC_VECTOR (5 downto 0) := (others => '0');
           addr : out STD_LOGIC_VECTOR (17 downto 0));
end component;
component clk_div
    Port ( realclk : in STD_LOGIC;
           fakeclk : out STD_LOGIC
           );
end component;
component vga_ctrl
    Port ( clk : in STD_LOGIC := '0';
           en : in STD_LOGIC := '0';
           hcount : out STD_LOGIC_VECTOR (9 downto 0) := (others => '0');
           vcount : out STD_LOGIC_VECTOR (9 downto 0) := (others => '0');
           vid : out STD_LOGIC := '0';
           hs : out STD_LOGIC := '0';
```

```
            vs : out STD_LOGIC := '0');
end component;
-- signals
signal en, vid, hs, vs : std_logic := '0';
signal hcount, vcount :  STD_LOGIC_VECTOR (9 downto 0) := (others => '0');
signal pixel : STD_LOGIC_VECTOR (7 downto 0) := (others => '0');
signal addr : STD_LOGIC_VECTOR (17 downto 0) := (others => '0');
begin

    -- no fucking clue how to connect the image
    smallclk : clk_div port map(
                    realclk => clk,
                    fakeclk => en);

    ctrl : vga_ctrl port map(
                    clk => clk,
                    en => en,
                    hcount => hcount, -- pp
                    vcount => vcount,
                    vid => vid, -- pp
                    hs => hs,
                    vs => vs); -- pp
    pp : pixel_pusher port map(
                    clk => clk,
                    en => en,
                    VS => vs, -- ctrl
                    pixel => pixel, -- pic ?
                    hcount => hcount, -- ctrl
                    vid => vid, --ctrl
                    R => vga_r, -- top?
                    B => vga_b, -- top
                    G => vga_g, --top
                    addr => addr); -- pic ?

    pic : picture port map(
                    clka => clk,
                    addra => addr, -- pp
                    douta => pixel); -- pp

                    vga_hs <= hs;
                    vga_vs <= vs;

end Behavioral;
```

Synthesis Schematic

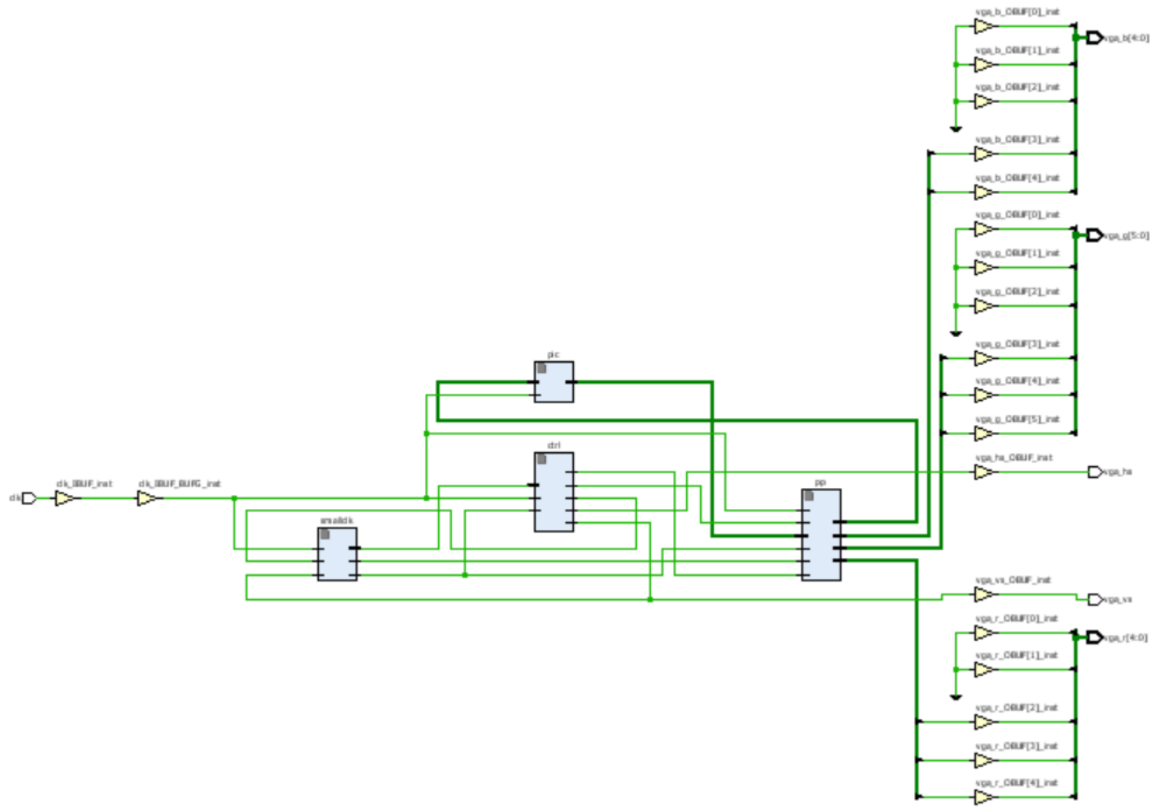


Figure 4: Synthesis of image_top.vhd.

Implementation

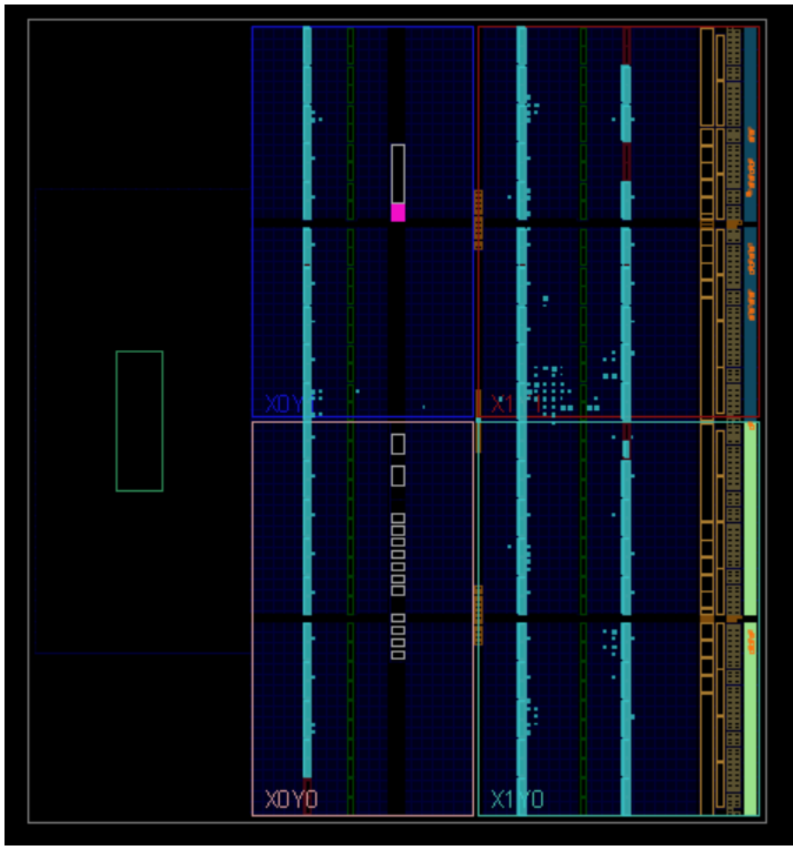
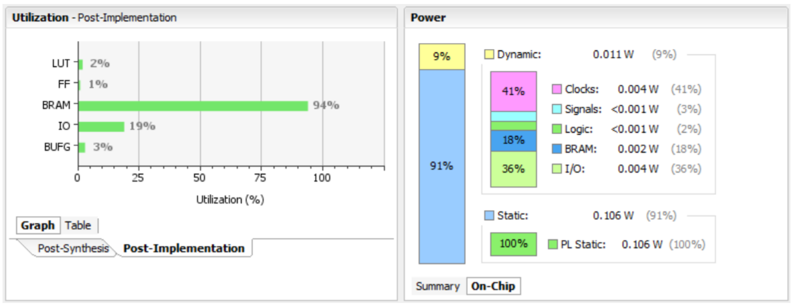


Figure 5: Implementation of image_top.vhd.

Power and Utilization



Power

```
9%    Dynamic:        0.011 W   (9%)
41%     Clocks:       0.004 W   (41%)
        Signals:     <0.001 W   (3%)
        Logic:       <0.001 W   (2%)
18%     BRAM:         0.002 W   (18%)
36%     I/O:          0.004 W   (36%)
91%   Static:         0.106 W   (91%)
100%    PL Static:    0.106 W   (100%)
```

Constraint 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 }];


##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 {
    jb_p[0] }]; #IO_L15P_T2_DQS_34 Sch=JB1_p
#set_property -dict { PACKAGE_PIN U20   IOSTANDARD LVCMOS33 } [get_ports {
    jb_n[0] }]; #IO_L15N_T2_DQS_34 Sch=JB1_N
#set_property -dict { PACKAGE_PIN V20   IOSTANDARD LVCMOS33 } [get_ports {
    jb_p[1] }]; #IO_L16P_T2_34 Sch=JB2_P
#set_property -dict { PACKAGE_PIN W20   IOSTANDARD LVCMOS33 } [get_ports {
    jb_n[1] }]; #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
```