Embedded Systems Spring 2019
Lab 4

Joseph Shenouda

April 11th 2019

Purpose

The purpose of this lab was to work with the Xilinix 7 series based Zybo board and use it understand how a VGA works with the corresponding VGA output on the Zybo. In this lab we first learned how VGA works and the timing involved in outputting pixels to the screen correctly. After this introduction of basic theory we read through the VGA reference from the Digilent site to see exactly how VGA works in the FPGA. We then took an image of choice and converted to a COE file from the MATLAB code given to us. Then we created a design in Vivado that would control the VGA timing and output our signals correctly. After some testing we moved on to making a memory block using the IP catalog and put in our COE file to create our Single Port ROM. Then we were tasked with creating an entity called "Pixel Pusher" which would control the RGB outputs according to the vga controller design made earlier and pulling the pixel data from the memory block that we had created from the Vivado tools. Finally we created a top level design utilizing the clock divider made earlier in the semester as well as the 3 entities that we created throughout this lab to output the static image of choice correctly on a monitor.

1 Prelab

Original Image

I used the following 480x480 image for this lab

Figure 1: Image used for VGA display

2 Timing is Everything

In this part of the lab we were tasked with creating the VGA controller that would essentially control the timing of our VGA output and control when we should be seeing a display as well as the values of our horizontal and vertical sync.

VHDL Design: vga_ctrl.vhd

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

entity vga_ctrl is
    Port ( signal clk : in std_logic;
           signal en : in std_logic;
           signal hcount, vcount : out std_logic_vector(9 downto 0);
           signal vid, hs, vs : out std_logic := '1'
           );
end vga_ctrl;

architecture Behavioral of vga_ctrl is

signal hcounter : std_logic_vector(9 downto 0) := (others => '0');
signal vcounter : std_logic_vector(9 downto 0) := (others => '0');


begin
--First process is to synchronously update hcounter and vcounter
process(clk)
begin
if(rising_edge(clk))then
    if en = '1' then

        if unsigned(hcounter) < 799 then
            hcounter <= std_logic_vector(unsigned(hcounter)+1);
        else
            hcounter <= (others => '0');
            --Only increments vcounter when hcounter has been reset to 0
                    if unsigned(vcounter) < 524 then
                        vcounter <= std_logic_vector(unsigned(vcounter)+1);
                    else
                        vcounter <= (others => '0');
                    end if;
        end if;

    end if;
end if;
end process;
--Second process to update vid, vs and hs whenever hcounter and vcounter change since we have to wait till we
process(hcounter ,vcounter)
begin
    if unsigned(hcounter) < 640 and unsigned(vcounter) < 480 then
        vid <= '1';
    else
        vid <= '0';
    end if;

    if unsigned(hcounter) >= 656 and unsigned(hcounter) <= 751 then
        hs <= '0';
    else
```

```
        hs <= '1';
    end if;

    if unsigned(vcounter) >= 490 and unsigned(vcounter) <= 491 then
        vs <= '0';
    else
        vs <= '1';
    end if;
end process;

hcount <=  hcounter;
vcount <= vcounter;

end Behavioral;
```

Testbench for vga_ctrl.vhd

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

entity vga_ctrl_tb is
end vga_ctrl_tb;

architecture Behavioral of vga_ctrl_tb is

component vga_ctrl is
port ( clk,en : in std_logic;
       hcount, vcount : out std_logic_vector(9 downto 0);
       vid, hs, vs : out std_logic
       );
end component;


--So our model only works off the clock input this is just for clarity in the tb
signal en : std_logic := '1';

signal clk : std_logic;
signal hcount, vcount : std_logic_vector(9 downto 0);
signal vid, hs, vs : std_logic;

begin

vga_control : vga_ctrl port map(
    clk => clk,
    en => en,
    hcount => hcount,
    vcount => vcount,
    vid  => vid,
    hs => hs,
    vs => vs
    );

process
begin
clk <= '1';
wait for 4ns;
clk <= '0';
wait for 4ns;
end process;
end Behavioral;
```

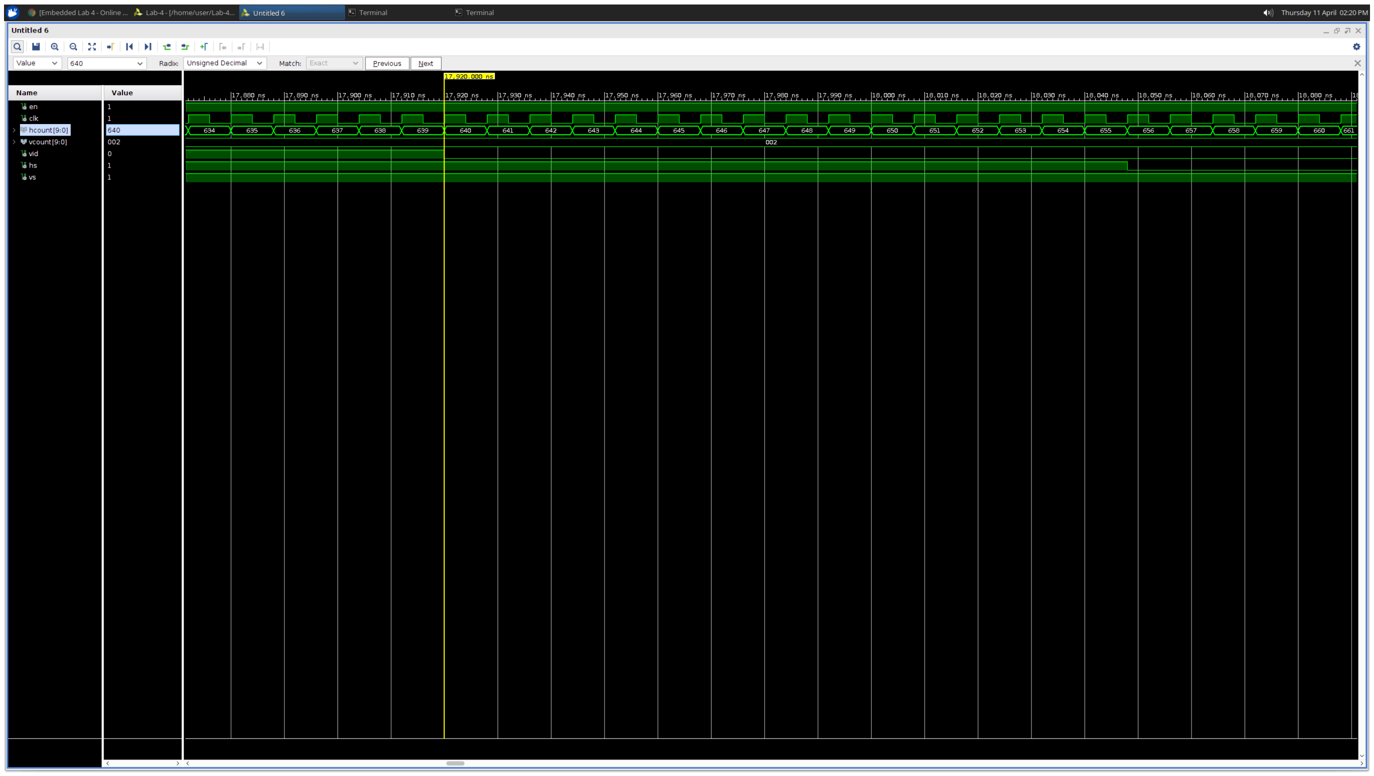


Figure 2: Waveform for vga_ctrl_tb.vhd

3 Pixel Pusher

In this section we created our memory block to store our pixel data into a single port ROM. We also created a VHDL design that would take our pixels from the memory block and output the corresponding pixels from the picture memory block. The pixel_pusher.vhd would operate according to the hcount that we had made in vga_ctrl.vhd. It also incremented an adder that would increment and get sent to addra in the picture memory block. In this section we also created the top level design that connected together the vga_ctrl.vhd pixel_pusher.vhd and clock_div.vhd with the clock divider set to output a clock at 25MHz. We also created the XDC file according to the pin out from the Digilent reference manual.

VHDL Design: pixel_pusher.vhd

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

entity pixel_pusher is
    port(
          signal clk, en, vs, vid : in std_logic;
          signal pixel : in std_logic_vector(7 downto 0);
          signal hcount : in std_logic_vector(9 downto 0);
          signal R :  out std_logic_vector(4 downto 0);
          signal B : out std_logic_vector(4 downto 0);
          signal G : out std_logic_vector(5 downto 0);
          signal addr : out std_logic_vector(17 downto 0)
         );
end pixel_pusher;

architecture Behavioral of pixel_pusher is

signal addrIn : std_logic_vector(17 downto 0) := (others => '0');


begin
```

```
adder : process(clk)
begin
if (rising_edge(clk)) then

    if en = '1' and 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";
        addrIn <= std_logic_vector(unsigned(addrIn)+1);
    else
        R <= (others => '0');
        G <= (others => '0');
        B <= (others => '0');
    end if;

    if vs = '0' then
        addrIn <= (others => '0');
    end if;
end if;
end process;

addr <= addrIn;

end Behavioral;
```

VHDL Design picture.vhd

```
-- (c) Copyright 1995-2019 Xilinx, Inc. All rights reserved.
--
-- This file contains confidential and proprietary information
-- of Xilinx, Inc. and is protected under U.S. and
-- international copyright and other intellectual property
-- laws.
--
-- DISCLAIMER
-- This disclaimer is not a license and does not grant any
-- rights to the materials distributed herewith. Except as
-- otherwise provided in a valid license issued to you by
-- Xilinx, and to the maximum extent permitted by applicable
-- law: (1) THESE MATERIALS ARE MADE AVAILABLE "AS IS" AND
-- WITH ALL FAULTS, AND XILINX HEREBY DISCLAIMS ALL WARRANTIES
-- AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING
-- BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-
-- INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and
-- (2) Xilinx shall not be liable (whether in contract or tort,
-- including negligence, or under any other theory of
-- liability) for any loss or damage of any kind or nature
-- related to, arising under or in connection with these
-- materials, including for any direct, or any indirect,
-- special, incidental, or consequential loss or damage
-- (including loss of data, profits, goodwill, or any type of
-- loss or damage suffered as a result of any action brought
-- by a third party) even if such damage or loss was
-- reasonably foreseeable or Xilinx had been advised of the
-- possibility of the same.
--
-- CRITICAL APPLICATIONS
-- Xilinx products are not designed or intended to be fail-
-- safe, or for use in any application requiring fail-safe
-- performance, such as life-support or safety devices or
-- systems, Class III medical devices, nuclear facilities,
-- applications related to the deployment of airbags, or any
-- other applications that could lead to death, personal
-- injury, or severe property or environmental damage
-- (individually and collectively, "Critical
-- Applications"). Customer assumes the sole risk and
-- liability of any use of Xilinx products in Critical
-- Applications, subject only to applicable laws and
-- regulations governing limitations on product liability.
--
-- THIS COPYRIGHT NOTICE AND DISCLAIMER MUST BE RETAINED AS
-- PART OF THIS FILE AT ALL TIMES.
--
-- DO NOT MODIFY THIS FILE.

-- IP VLNV: xilinx.com:ip:blk_mem_gen:8.4
-- IP Revision: 1

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

LIBRARY blk_mem_gen_v8_4_1;
USE blk_mem_gen_v8_4_1.blk_mem_gen_v8_4_1;

ENTITY picture IS
  PORT (
    clka : IN STD_LOGIC;
    addra : IN STD_LOGIC_VECTOR(17 DOWNTO 0);
    douta : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
  );
END picture;

ARCHITECTURE picture_arch OF picture IS
  ATTRIBUTE DowngradeIPIdentifiedWarnings : STRING;
  ATTRIBUTE DowngradeIPIdentifiedWarnings OF picture_arch: ARCHITECTURE IS "yes";
  COMPONENT blk_mem_gen_v8_4_1 IS
    GENERIC (
      C_FAMILY : STRING;
      C_XDEVICEFAMILY : STRING;
      C_ELABORATION_DIR : STRING;
      C_INTERFACE_TYPE : INTEGER;
      C_AXI_TYPE : INTEGER;
      C_AXI_SLAVE_TYPE : INTEGER;
      C_USE_BRAM_BLOCK : INTEGER;
      C_ENABLE_32BIT_ADDRESS : INTEGER;
      C_CTRL_ECC_ALGO : STRING;
      C_HAS_AXI_ID : INTEGER;
      C_AXI_ID_WIDTH : INTEGER;
      C_MEM_TYPE : INTEGER;
      C_BYTE_SIZE : INTEGER;
      C_ALGORITHM : INTEGER;
      C_PRIM_TYPE : INTEGER;
      C_LOAD_INIT_FILE : INTEGER;
      C_INIT_FILE_NAME : STRING;
      C_INIT_FILE : STRING;
      C_USE_DEFAULT_DATA : INTEGER;
      C_DEFAULT_DATA : STRING;
      C_HAS_RSTA : INTEGER;
      C_RST_PRIORITY_A : STRING;
      C_RSTRAM_A : INTEGER;
      C_INITA_VAL : STRING;
      C_HAS_ENA : INTEGER;
      C_HAS_REGCEA : INTEGER;
      C_USE_BYTE_WEA : INTEGER;
      C_WEA_WIDTH : INTEGER;
      C_WRITE_MODE_A : STRING;
      C_WRITE_WIDTH_A : INTEGER;
      C_READ_WIDTH_A : INTEGER;
      C_WRITE_DEPTH_A : INTEGER;
      C_READ_DEPTH_A : INTEGER;
      C_ADDRA_WIDTH : INTEGER;
      C_HAS_RSTB : INTEGER;
      C_RST_PRIORITY_B : STRING;
      C_RSTRAM_B : INTEGER;
      C_INITB_VAL : STRING;
      C_HAS_ENB : INTEGER;
      C_HAS_REGCEB : INTEGER;
      C_USE_BYTE_WEB : INTEGER;
      C_WEB_WIDTH : INTEGER;
      C_WRITE_MODE_B : STRING;
      C_WRITE_WIDTH_B : INTEGER;
      C_READ_WIDTH_B : INTEGER;
      C_WRITE_DEPTH_B : INTEGER;
      C_READ_DEPTH_B : INTEGER;
      C_ADDRB_WIDTH : INTEGER;
      C_HAS_MEM_OUTPUT_REGS_A : INTEGER;
      C_HAS_MEM_OUTPUT_REGS_B : INTEGER;
      C_HAS_MUX_OUTPUT_REGS_A : INTEGER;
      C_HAS_MUX_OUTPUT_REGS_B : INTEGER;
      C_MUX_PIPELINE_STAGES : INTEGER;
      C_HAS_SOFTECC_INPUT_REGS_A : INTEGER;
      C_HAS_SOFTECC_OUTPUT_REGS_B : INTEGER;
      C_USE_SOFTECC : INTEGER;
      C_USE_ECC : INTEGER;
      C_EN_ECC_PIPE : INTEGER;
      C_HAS_INJECTERR : INTEGER;
      C_SIM_COLLISION_CHECK : STRING;
      C_COMMON_CLK : INTEGER;
      C_DISABLE_WARN_BHV_COLL : INTEGER;
      C_EN_SLEEP_PIN : INTEGER;
      C_USE_URAM : INTEGER;
      C_EN_RDADDRA_CHG : INTEGER;
      C_EN_RDADDRB_CHG : INTEGER;
      C_EN_DEEPSLEEP_PIN : INTEGER;
      C_EN_SHUTDOWN_PIN : INTEGER;
      C_EN_SAFETY_CKT : INTEGER;
      C_DISABLE_WARN_BHV_RANGE : INTEGER;
      C_COUNT_36K_BRAM : STRING;
      C_COUNT_18K_BRAM : STRING;
      C_EST_POWER_SUMMARY : STRING
    );
    PORT (
      clka : IN STD_LOGIC;
      rsta : IN STD_LOGIC;
      ena : IN STD_LOGIC;
      regcea : IN STD_LOGIC;
      wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
      addra : IN STD_LOGIC_VECTOR(17 DOWNTO 0);
```

```
      dina : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
      douta : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
      clkb : IN STD_LOGIC;
      rstb : IN STD_LOGIC;
      enb : IN STD_LOGIC;
      regceb : IN STD_LOGIC;
      web : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
      addrb : IN STD_LOGIC_VECTOR(17 DOWNTO 0);
      dinb : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
      doutb : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
      injectsbiterr : IN STD_LOGIC;
      injectdbiterr : IN STD_LOGIC;
      eccpipece : IN STD_LOGIC;
      sbiterr : OUT STD_LOGIC;
      dbiterr : OUT STD_LOGIC;
      rdaddrecc : OUT STD_LOGIC_VECTOR(17 DOWNTO 0);
      sleep : IN STD_LOGIC;
      deepsleep : IN STD_LOGIC;
      shutdown : IN STD_LOGIC;
      rsta_busy : OUT STD_LOGIC;
      rstb_busy : OUT STD_LOGIC;
      s_aclk : IN STD_LOGIC;
      s_aresetn : IN STD_LOGIC;
      s_axi_awid : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
      s_axi_awaddr : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
      s_axi_awlen : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
      s_axi_awsize : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
      s_axi_awburst : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
      s_axi_awvalid : IN STD_LOGIC;
      s_axi_awready : OUT STD_LOGIC;
      s_axi_wdata : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
      s_axi_wstrb : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
      s_axi_wlast : IN STD_LOGIC;
      s_axi_wvalid : IN STD_LOGIC;
      s_axi_wready : OUT STD_LOGIC;
      s_axi_bid : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
      s_axi_bresp : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
      s_axi_bvalid : OUT STD_LOGIC;
      s_axi_bready : IN STD_LOGIC;
      s_axi_arid : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
      s_axi_araddr : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
      s_axi_arlen : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
      s_axi_arsize : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
      s_axi_arburst : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
      s_axi_arvalid : IN STD_LOGIC;
      s_axi_arready : OUT STD_LOGIC;
      s_axi_rid : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
      s_axi_rdata : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
      s_axi_rresp : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
      s_axi_rlast : OUT STD_LOGIC;
      s_axi_rvalid : OUT STD_LOGIC;
      s_axi_rready : IN STD_LOGIC;
      s_axi_injectsbiterr : IN STD_LOGIC;
      s_axi_injectdbiterr : IN STD_LOGIC;
      s_axi_sbiterr : OUT STD_LOGIC;
      s_axi_dbiterr : OUT STD_LOGIC;
      s_axi_rdaddrecc : OUT STD_LOGIC_VECTOR(17 DOWNTO 0)
    );
  END COMPONENT blk_mem_gen_v8_4_1;
  ATTRIBUTE X_CORE_INFO : STRING;
  ATTRIBUTE X_CORE_INFO OF picture_arch: ARCHITECTURE IS "blk_mem_gen_v8_4_1,Vivado 2018.2";
  ATTRIBUTE CHECK_LICENSE_TYPE : STRING;
  ATTRIBUTE CHECK_LICENSE_TYPE OF picture_arch : ARCHITECTURE IS "picture,blk_mem_gen_v8_4_1,{}";
  ATTRIBUTE CORE_GENERATION_INFO : STRING;
  ATTRIBUTE CORE_GENERATION_INFO OF picture_arch: ARCHITECTURE IS "picture,blk_mem_gen_v8_4_1,{x_ipProduct=Viv
"FILE=picture.mem,C_USE_DEFAULT_DATA=0,C_DEFAULT_DATA=0,C_HAS_RSTA=0,C_RST_PRIORITY_A=CE,C_RSTRAM_A=0,C_INITA_
"C_READ_DEPTH_B=230400,C_ADDRB_WIDTH=18,C_HAS_MEM_OUTPUT_REGS_A=0,C_HAS_MEM_OUTPUT_REGS_B=0,C_HAS_MUX_OUTPUT_R
"_RANGE=0,C_COUNT_36K_BRAM=56,C_COUNT_18K_BRAM=1,C_EST_POWER_SUMMARY=Estimated Power for IP     _     2.321479
  ATTRIBUTE X_INTERFACE_INFO : STRING;
  ATTRIBUTE X_INTERFACE_PARAMETER : STRING;
  ATTRIBUTE X_INTERFACE_INFO OF douta: SIGNAL IS "xilinx.com:interface:bram:1.0 BRAM_PORTA DOUT";
  ATTRIBUTE X_INTERFACE_INFO OF addra: SIGNAL IS "xilinx.com:interface:bram:1.0 BRAM_PORTA ADDR";
  ATTRIBUTE X_INTERFACE_PARAMETER OF clka: SIGNAL IS "XIL_INTERFACENAME BRAM_PORTA, MEM_SIZE 8192, MEM_WIDTH 3
  ATTRIBUTE X_INTERFACE_INFO OF clka: SIGNAL IS "xilinx.com:interface:bram:1.0 BRAM_PORTA CLK";
BEGIN
  U0 : blk_mem_gen_v8_4_1
    GENERIC MAP (
      C_FAMILY => "zynq",
      C_XDEVICEFAMILY => "zynq",
      C_ELABORATION_DIR => "./",
      C_INTERFACE_TYPE => 0,
      C_AXI_TYPE => 1,
      C_AXI_SLAVE_TYPE => 0,
      C_USE_BRAM_BLOCK => 0,
      C_ENABLE_32BIT_ADDRESS => 0,
      C_CTRL_ECC_ALGO => "NONE",
      C_HAS_AXI_ID => 0,
      C_AXI_ID_WIDTH => 4,
      C_MEM_TYPE => 3,
      C_BYTE_SIZE => 9,
      C_ALGORITHM => 1,
      C_PRIM_TYPE => 1,
      C_LOAD_INIT_FILE => 1,
      C_INIT_FILE_NAME => "picture.mif",
      C_INIT_FILE => "picture.mem",
      C_USE_DEFAULT_DATA => 0,
      C_DEFAULT_DATA => "0",
      C_HAS_RSTA => 0,
      C_RST_PRIORITY_A => "CE",
      C_RSTRAM_A => 0,
      C_INITA_VAL => "0",
      C_HAS_ENA => 0,
      C_HAS_REGCEA => 0,
      C_USE_BYTE_WEA => 0,
      C_WEA_WIDTH => 1,
      C_WRITE_MODE_A => "WRITE_FIRST",
      C_WRITE_WIDTH_A => 8,
      C_READ_WIDTH_A => 8,
      C_WRITE_DEPTH_A => 230400,
      C_READ_DEPTH_A => 230400,
      C_ADDRA_WIDTH => 18,
      C_HAS_RSTB => 0,
      C_RST_PRIORITY_B => "CE",
      C_RSTRAM_B => 0,
      C_INITB_VAL => "0",
      C_HAS_ENB => 0,
      C_HAS_REGCEB => 0,
      C_USE_BYTE_WEB => 0,
      C_WEB_WIDTH => 1,
      C_WRITE_MODE_B => "WRITE_FIRST",
    C_WRITE_WIDTH_B => 8,
    C_READ_WIDTH_B => 8,
    C_WRITE_DEPTH_B => 230400,
    C_READ_DEPTH_B => 230400,
    C_ADDRB_WIDTH => 18,
    C_HAS_MEM_OUTPUT_REGS_A => 0,
    C_HAS_MEM_OUTPUT_REGS_B => 0,
    C_HAS_MUX_OUTPUT_REGS_A => 0,
    C_HAS_MUX_OUTPUT_REGS_B => 0,
    C_MUX_PIPELINE_STAGES => 0,
    C_HAS_SOFTECC_INPUT_REGS_A => 0,
    C_HAS_SOFTECC_OUTPUT_REGS_B => 0,
    C_USE_SOFTECC => 0,
    C_USE_ECC => 0,
    C_EN_ECC_PIPE => 0,
    C_HAS_INJECTERR => 0,
    C_SIM_COLLISION_CHECK => "ALL",
    C_COMMON_CLK => 0,
    C_DISABLE_WARN_BHV_COLL => 0,
    C_EN_SLEEP_PIN => 0,
    C_USE_URAM => 0,
    C_EN_RDADDRA_CHG => 0,
    C_EN_RDADDRB_CHG => 0,
    C_EN_DEEPSLEEP_PIN => 0,
    C_EN_SHUTDOWN_PIN => 0,
    C_EN_SAFETY_CKT => 0,
    C_DISABLE_WARN_BHV_RANGE => 0,
    C_COUNT_36K_BRAM => "56",
    C_COUNT_18K_BRAM => "1",
    C_EST_POWER_SUMMARY => "Estimated Power for IP     :     2.321479 mW"
  )
  PORT MAP (
    clka => clka,
    rsta => '0',
    ena => '0',
    regcea => '0',
    wea => STD_LOGIC_VECTOR(TO_UNSIGNED(0, 1)),
    addra => addra,
    dina => STD_LOGIC_VECTOR(TO_UNSIGNED(0, 8)),
    douta => douta,
    clkb => '0',
    rstb => '0',
    enb => '0',
    regceb => '0',
    web => STD_LOGIC_VECTOR(TO_UNSIGNED(0, 1)),
    addrb => STD_LOGIC_VECTOR(TO_UNSIGNED(0, 18)),
    dinb => STD_LOGIC_VECTOR(TO_UNSIGNED(0, 8)),
    injectsbiterr => '0',
    injectdbiterr => '0',
    eccpipece => '0',
    sleep => '0',
    deepsleep => '0',
    shutdown => '0',
    s_aclk => '0',
    s_aresetn => '0',
    s_axi_awid => STD_LOGIC_VECTOR(TO_UNSIGNED(0, 4)),
    s_axi_awaddr => STD_LOGIC_VECTOR(TO_UNSIGNED(0, 32)),
    s_axi_awlen => STD_LOGIC_VECTOR(TO_UNSIGNED(0, 8)),
    s_axi_awsize => STD_LOGIC_VECTOR(TO_UNSIGNED(0, 3)),
    s_axi_awburst => STD_LOGIC_VECTOR(TO_UNSIGNED(0, 2)),
        s_axi_awvalid => '0',
        s_axi_wdata => STD_LOGIC_VECTOR(TO_UNSIGNED(0, 8)),
        s_axi_wstrb => STD_LOGIC_VECTOR(TO_UNSIGNED(0, 1)),
        s_axi_wlast => '0',
        s_axi_wvalid => '0',
        s_axi_bready => '0',
        s_axi_arid => STD_LOGIC_VECTOR(TO_UNSIGNED(0, 4)),
        s_axi_araddr => STD_LOGIC_VECTOR(TO_UNSIGNED(0, 32)),
        s_axi_arlen => STD_LOGIC_VECTOR(TO_UNSIGNED(0, 8)),
        s_axi_arsize => STD_LOGIC_VECTOR(TO_UNSIGNED(0, 3)),
        s_axi_arburst => STD_LOGIC_VECTOR(TO_UNSIGNED(0, 2)),
        s_axi_arvalid => '0',
        s_axi_rready => '0',
        s_axi_injectsbiterr => '0',
        s_axi_injectdbiterr => '0'
      );
END picture_arch;
```

Block Diagram for Top Level

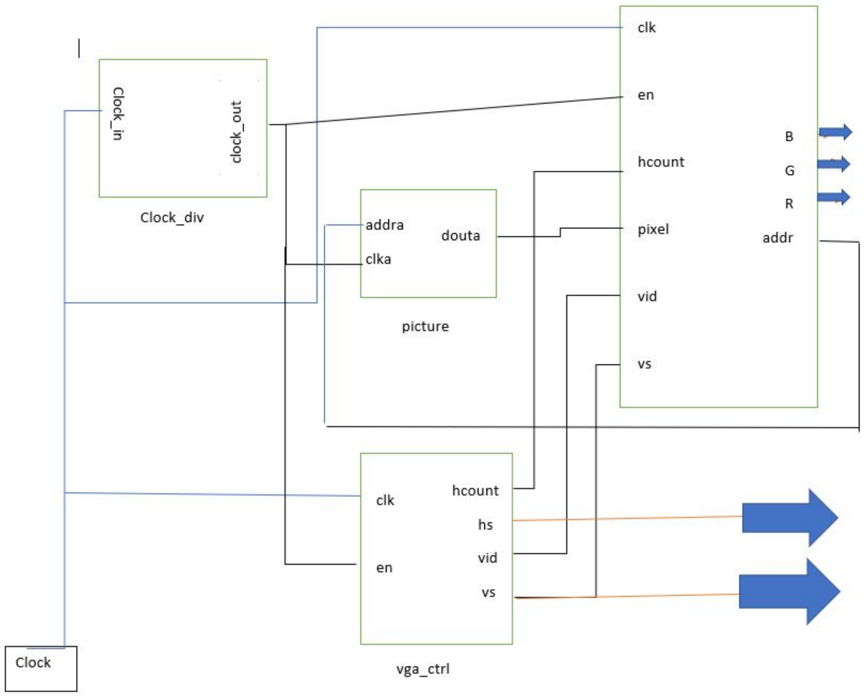


Figure 3: Block diagram for the top level design

VHDL Design: image_top.vhd

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


entity image_top is
    port( signal clk : in std_logic;
          signal vga_r , vga_b : out std_logic_vector(4 downto 0);
          signal vga_g : out std_logic_vector(5 downto 0);
          signal vga_hs, vga_vs : out std_logic
         );
end image_top;

architecture Behavioral of image_top is

signal vga_vs_temp : std_logic;
signal enable : std_logic;
signal vid : std_logic;
signal douta : std_logic_vector(7 downto 0);
signal hcount : std_logic_vector(9 downto 0);
signal addr : std_logic_vector(17 downto 0);

component picture is
port(addra : in std_logic_vector(17 downto 0);
     clka : in std_logic;
     douta : out std_logic_vector(7 downto 0)
     );
end component;

component pixel_pusher is
port( clk, en, vid, vs : in std_logic;
      pixel : in std_logic_vector(7 downto 0);
      hcount : in std_logic_vector(9 downto 0);
      R, B : out std_logic_vector(4 downto 0);
      G : out std_logic_vector(5 downto 0);
      addr : out std_logic_vector(17 downto 0)
      );
end component;

component vga_ctrl is
port( clk, en : in std_logic;
      hcount : out std_logic_vector(9 downto 0);
      vid : out std_logic;
      hs : out std_logic;
      vs : out std_logic
      );
end component;

component clock_div is
port( clock_in : in std_logic;
      clock_out : out std_logic
      );
end component;

begin

clk_div: clock_div port map(
    clock_in => clk,
    clock_out => enable
);

pixel_push : pixel_pusher port map(
    clk => clk,
    en => enable,
    vs => vga_vs_temp,
    vid => vid,
    pixel => douta,
    hcount => hcount,
    addr => addr,
    R => vga_r,
    B => vga_b,
    G => vga_g
);

pic : picture port map(
    clka => enable,
    addra => addr,
    douta => douta
);

vga_control : vga_ctrl port map(
    en => enable,
    clk => clk,
    hcount => hcount,
    vid => vid,
    hs => vga_hs,
    vs => vga_vs_temp
);

vga_vs <= vga_vs_temp;
end Behavioral;
```

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=sysc
create_clock -add -name sys_clk_pin -period 8.00 -waveform {0 4} [get_ports { clk }];


##VGA Connector
set_property -dict { PACKAGE_PIN M19   IOSTANDARD LVCMOS33 } [get_ports { vga_r[0] }]; #IO_L7P_T1_AD2P_35 Sch=
set_property -dict { PACKAGE_PIN L20   IOSTANDARD LVCMOS33 } [get_ports { vga_r[1] }]; #IO_L9N_T1_DQS_AD3N_35
set_property -dict { PACKAGE_PIN J20   IOSTANDARD LVCMOS33 } [get_ports { vga_r[2] }]; #IO_L17P_T2_AD5P_35 Sch
set_property -dict { PACKAGE_PIN G20   IOSTANDARD LVCMOS33 } [get_ports { vga_r[3] }]; #IO_L18N_T2_AD13N_35 Sc
set_property -dict { PACKAGE_PIN F19   IOSTANDARD LVCMOS33 } [get_ports { vga_r[4] }]; #IO_L15P_T2_DQS_AD12P_3
set_property -dict { PACKAGE_PIN H18   IOSTANDARD LVCMOS33 } [get_ports { vga_g[0] }]; #IO_L14N_T2_AD4N_SRCC_3
set_property -dict { PACKAGE_PIN N20   IOSTANDARD LVCMOS33 } [get_ports { vga_g[1] }]; #IO_L14P_T2_SRCC_34 Sch
set_property -dict { PACKAGE_PIN L19   IOSTANDARD LVCMOS33 } [get_ports { vga_g[2] }]; #IO_L9P_T1_DQS_AD3P_35
set_property -dict { PACKAGE_PIN J19   IOSTANDARD LVCMOS33 } [get_ports { vga_g[3] }]; #IO_L10N_T1_AD11N_35 Sc
set_property -dict { PACKAGE_PIN H20   IOSTANDARD LVCMOS33 } [get_ports { vga_g[4] }]; #IO_L17N_T2_AD5N_35 Sch
set_property -dict { PACKAGE_PIN F20   IOSTANDARD LVCMOS33 } [get_ports { vga_g[5] }]; #IO_L15N_T2_DQS_AD12N_3
set_property -dict { PACKAGE_PIN P20   IOSTANDARD LVCMOS33 } [get_ports { vga_b[0] }]; #IO_L14N_T2_SRCC_34 Sch
set_property -dict { PACKAGE_PIN M20   IOSTANDARD LVCMOS33 } [get_ports { vga_b[1] }]; #IO_L7N_T1_AD2N_35 Sch=
set_property -dict { PACKAGE_PIN K19   IOSTANDARD LVCMOS33 } [get_ports { vga_b[2] }]; #IO_L10P_T1_AD11P_35 Sc
set_property -dict { PACKAGE_PIN J18   IOSTANDARD LVCMOS33 } [get_ports { vga_b[3] }]; #IO_L14P_T2_AD4P_SRCC_3
set_property -dict { PACKAGE_PIN G19   IOSTANDARD LVCMOS33 } [get_ports { vga_b[4] }]; #IO_L18P_T2_AD13P_35 Sc
set_property -dict { PACKAGE_PIN P19   IOSTANDARD LVCMOS33 } [get_ports vga_hs]; #IO_L13N_T2_MRCC_34 Sch=VGA_H
set_property -dict { PACKAGE_PIN R19   IOSTANDARD LVCMOS33 } [get_ports vga_vs]; #IO_0_34 Sch=VGA_VS
```

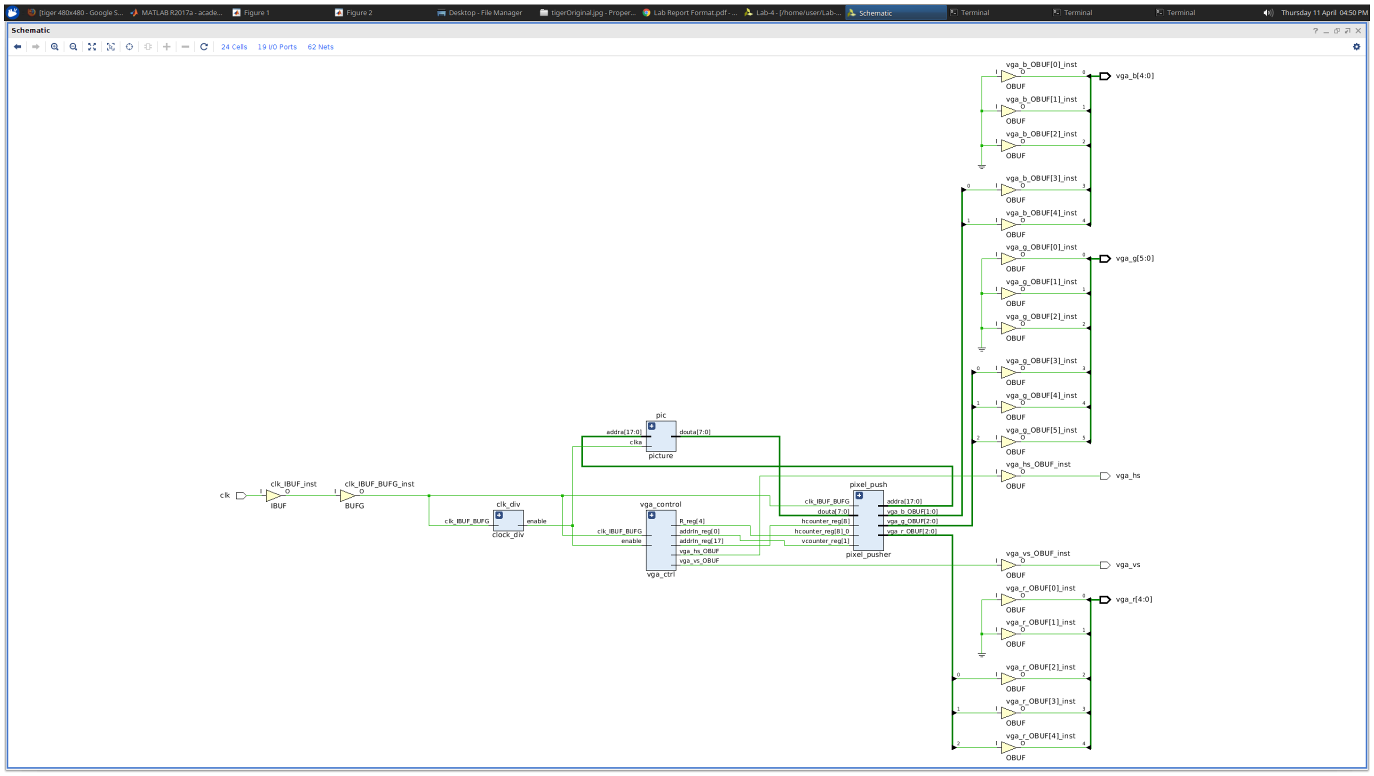


Figure 4: Synthesis Schematic of image_top.vhd

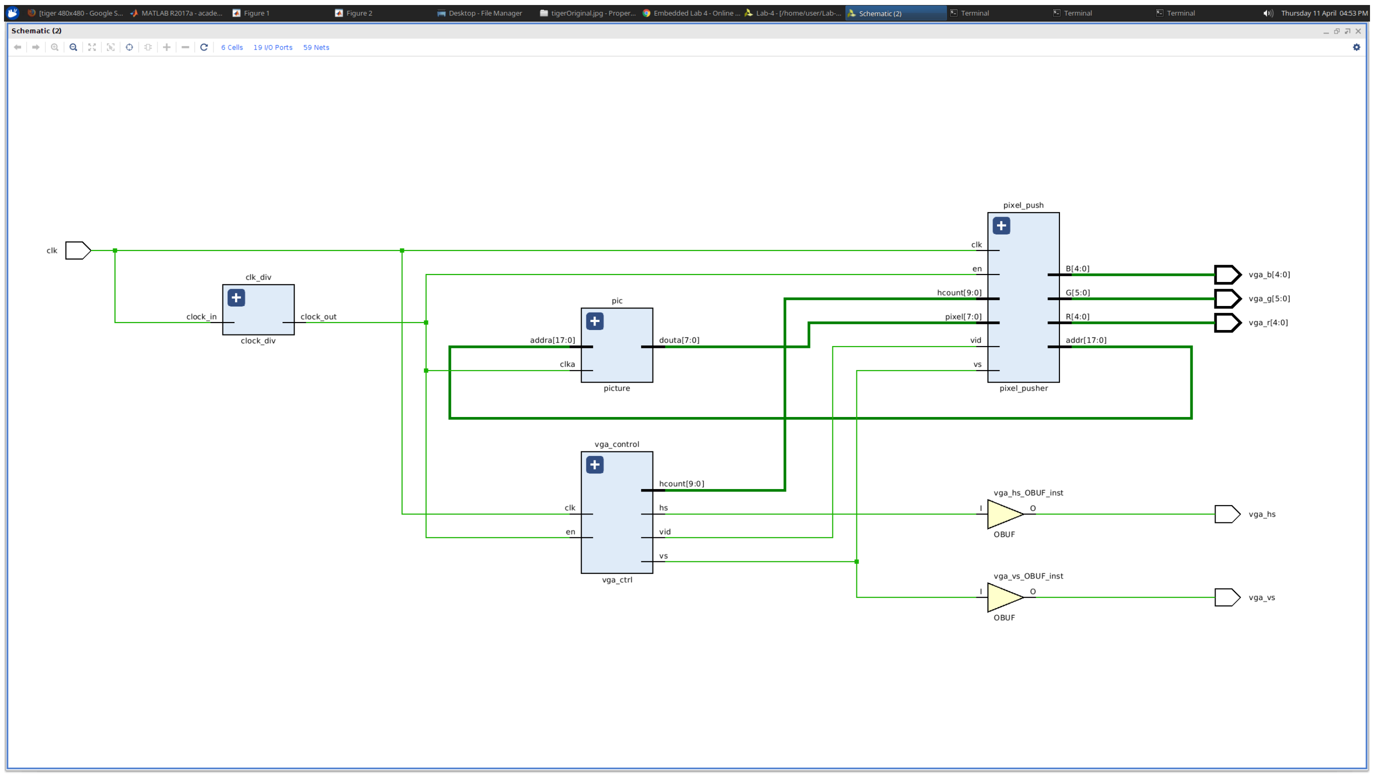
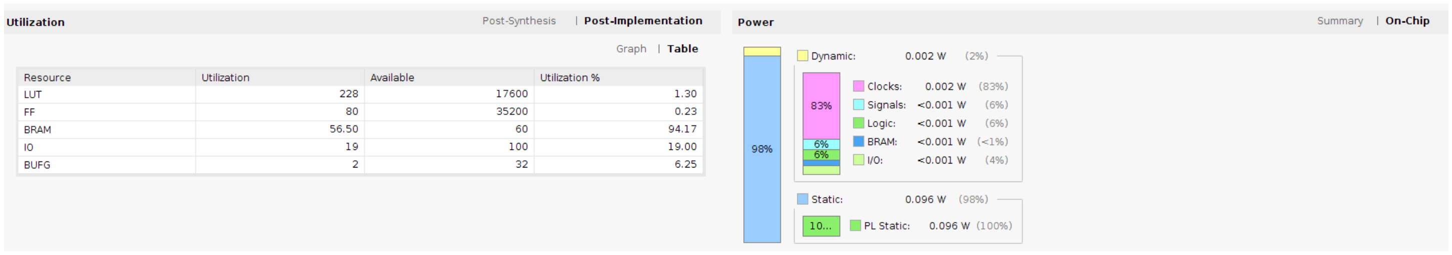


Figure 5: Elaboration Schematic



Resource	Utilization	Available	Utilization %
LUT	228	17600	1.30
FF	80	35200	0.23
BRAM	56.50	60	94.17
IO	19	100	19.00
BUFG	2	32	6.25

Dynamic:	0.002 W	(2%)
Clocks:	0.002 W	(83%)
Signals:	<0.001 W	(6%)
Logic:	<0.001 W	(6%)
BRAM:	<0.001 W	(<1%)
I/O:	<0.001 W	(4%)
Static:	0.096 W	(98%)
PL Static:	0.096 W	(100%)

Figure 6: Power Graphs and Utilization Table

Discussion

In this lab we worked with the VGA output and eventually worked our way up to displaying images from the Zybo board to a monitor. The major things I learned throughout this lab were how the board stores image in the COE files as well as how to use IP blocks for creating ROM and managing memory in the FPGA. I also learned more about how the process work in a VHDL design and the nuances involved when I was simulating and debugging the design. Finally I got more comfortable with the waveform tool in Vivado and the various features that it possesses for an easier time debugging.