|  |
| --- |
| Digital System Design SS2024 |
|  |
| Lab 1: Design and Test of VHDL IP |
| Report submitted by:  **Soumya Ranjan Sabat**  Matrikelnr: 1127993  **Azaz Hassan Khan**  Matrikelnr: 1128032 |
|  |
| **May/17/2024** |

Content

[1 Introduction 3](#_Toc167139637)

[1.1 Part-1: Study top level RTL design 3](#_Toc167139638)

[1.1.1 Task-1 3](#_Toc167139639)

[1.1.2 Task-2 4](#_Toc167139640)

[1.2 Part-2: Design of camera emulator 5](#_Toc167139641)

[1.2.1 Task-1 5](#_Toc167139642)

[1.2.2 Task-2 6](#_Toc167139643)

[1.3 Part-3: Simulation of camera emulator 7](#_Toc167139644)

[1.3.1 Task-1 7](#_Toc167139645)

[1.3.2 Task-2 8](#_Toc167139646)

[1.4 Part-4: Test of HDMI Display Controller on ZedBoard 8](#_Toc167139647)

[1.4.1 Task-1 8](#_Toc167139648)

[1.4.2 Task-2 9](#_Toc167139649)

[1.4.3 Task-3 10](#_Toc167139650)

[2 References 11](#_Toc167139651)

# Introduction

Test a HDMI display controller, designed to display images captured by a camera. Data of the HDMI display controller is finally sent to the on-board HDMI transmitter chip. The camera is getting emulated by another VHDL model.

Objectives:

* Study of the top-level RTL design
* VHDL design of the Camera Emulator
* Simulation of the Camera Emulator
* Test of HDMI Display Controller on ZedBoard

## Part-1: Study top level RTL design

### Task-1

Q1. Describe the meaning of all output signals of HDMI\_V1!! Consider that those output signals are inputs to the HDMI transmitter ADV7511. Do research on ADV7511 of ZedBoard. Use a table format with two columns: signal name and description!

Elaboration:

|  |  |
| --- | --- |
| **Signal Name** | **Description** |
| Data\_out [15:0] | * 16-bit Data output of frame buffer |
| HDMI\_CLK | * A clock of 25MHz from vga\_pll\_zedboard |
| config\_done | * CEC data signal. Supports CMOS logic levels from 1.8V to 5V * One of the outputs of ov7670\_controller * A LED to show when config is finished |
| PwDn | * Power saver mode. One of the outputs of ov7670\_controller (transfers registers to the camera over an I2C like bus) * A LED shows if it is at the power saver mode |
| de | * Data Enable signal input for Digital Video. Supports typical CMOS logic levels from 1.8V up to 3.3V * One of the outputs of vga\_controller and acts as input to Video data capture of the HDMI transmitter ADV7511 * Display enables where '1' is the display time and '0' is the blanking time |
| h\_sync | * Horizontal Sync Pulse input. Supports typical CMOS logic levels from 1.8V up to 3.3V * One of the outputs of vga\_controller and acts as input to Video data capture of the HDMI transmitter ADV7511 |
| v\_sync | * Vertical Sync Pulse input. Supports typical CMOS logic levels from 1.8V up to 3.3V * One of the outputs of vga\_controller and acts as input to Video data capture of the HDMI transmitter ADV7511 |
| siod\_hdmi | * Serial Port Data I/O. Supports CMOS logic levels from 1.8V to 3.3V * This pin serves as the serial port data I/O slave for register access |
| sioc\_hdmi | * Serial Port Data Clock input. Supports CMOS logic levels from 1.8V to 3.3V * Can write to the registers |
| xClk | * Clk Driver for OV7670 camera * One of the outputs of ov7670\_controller |
| Reset | * Always '1' for normal mode * One of the outputs of ov7670\_controller |

### Task-2

Q1. The component IMG\_GENERATOR emulates the camera. Do research on OV7670 camera (data sheet). Describe each signal provided by the camera (assume a resolution setting of 640x480 and an output format of Y/Cb/Cr 4:2:2) and compare with the output ports of IMG\_GENERATOR! Use a table format with two columns: signal name and description!

Elaboration:

|  |  |
| --- | --- |
| **Signal Name** | **Description** |
| cam\_pwdn | * Power Down Mode Selection * 0: Normal mode 1: Power down mode |
| cam\_reset | * Clears all registers and resets them to their default values. * 0: Normal mode 1: Reset mode |
| cam\_sioc | * SCCB serial interface clock input |
| cam\_xclk | * System clock input |
| cam\_data[7:0] | * 8-bit output of Pixel\_Generator * YUV video component output |
| cam\_href | * HREF output * It is responsible for synchronizing each line of the image frame |
| cam\_pclk | * Pixel clock output |
| cam\_siod | * SCCB serial interface data I/O |
| cam\_vsync | * Vertical sync output * It is responsible for synchronizing an entire image frame on the screen |

## Part-2: Design of camera emulator

### Task-1

Q1. In *Camera\_Clock\_Generator.vhd* we need to generate both, VSYNC and HREF in order to emulate the camera.

|  |
| --- |
| 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 Camera\_Clock\_Generator is  Port (  pclk : in STD\_LOGIC;  href : out STD\_LOGIC;  vsync : out STD\_LOGIC  );  end Camera\_Clock\_Generator;  architecture Behavioral of Camera\_Clock\_Generator is  signal temp\_vsync : std\_ulogic := '1';  signal href\_ena : std\_ulogic := '0';  signal cnt\_vsync\_pclk : integer range 0 to 799680 := 0;  signal cnt\_href\_pclk : integer range 0 to 784 := 0;  begin  HREF\_VSYNC\_gen: process(pclk)  begin  if falling\_edge(pclk) then  if (cnt\_vsync\_pclk = ((3\*1568)-1)) and (temp\_vsync = '1') then  temp\_vsync <= '0';  elsif (cnt\_vsync\_pclk = ((510\*1568)-1)) and (temp\_vsync = '0') then  temp\_vsync <= '1';  end if;  if (cnt\_vsync\_pclk = ((20\*1568)-1)) and (temp\_vsync = '0') then  href\_ena <= '1';  elsif (cnt\_vsync\_pclk = ((500\*1568)-1)) and (temp\_vsync = '0') then  href\_ena <= '0';  end if;  if (cnt\_href\_pclk = 0) and (href\_ena = '1') then  href <= '1';  elsif (cnt\_href\_pclk = 640) and (href\_ena = '1') then  href <= '0';  end if;  vsync <= temp\_vsync;  cnt\_vsync\_pclk <= cnt\_vsync\_pclk + 1;  cnt\_href\_pclk <= cnt\_href\_pclk + 1;  if cnt\_href\_pclk = 783 then  cnt\_href\_pclk <= 0;  end if;  if cnt\_vsync\_pclk = 799679 then  cnt\_vsync\_pclk <= 0;  end if;  end if;  end process;  end Behavioral; |

### Task-2

Q1. In Pixel\_Generator.vhd we need to specify Y, CB and CR. Use VHDL constant with data type STD\_LOGIC\_VECTOR (7 downto 0)! Generate blue pixels!

Elaboration:

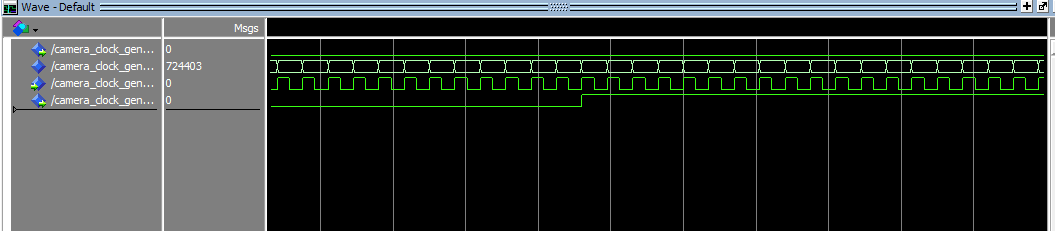
|  |
| --- |
| library IEEE;  use IEEE.STD\_LOGIC\_1164.ALL;  use IEEE.NUMERIC\_STD.ALL;  entity Pixel\_Generator is  Port (  pclk : in STD\_LOGIC;  href : in STD\_LOGIC;  vsync : in STD\_LOGIC;  data : out STD\_LOGIC\_VECTOR (7 downto 0)  );  end Pixel\_Generator;  architecture Behavioral of Pixel\_Generator is  -- Blue pixel: Y: 41 (29), Cb: 240 (F0), Cr: 110 (6E)  constant Y\_BLUE : STD\_LOGIC\_VECTOR (7 downto 0) := x"4C";  constant CB\_BLUE : STD\_LOGIC\_VECTOR (7 downto 0) := x"54";  constant CR\_BLUE : STD\_LOGIC\_VECTOR (7 downto 0) := x"FF";    signal cnt\_col\_pixel : integer range 0 to 1279 := 0; -- Column pixel counter  begin  -- Pixel generation process  pixel\_gen\_yCbCr : process(pclk)  begin  if rising\_edge(pclk) then  if (href = '1') and (vsync = '0') then -- Generate data only during active video line  if (cnt\_col\_pixel mod 4 = 0) then  data <= CB\_BLUE; -- Cb component  elsif (cnt\_col\_pixel mod 4 = 1) then  data <= Y\_BLUE; -- Y component  elsif (cnt\_col\_pixel mod 4 = 2) then  data <= CR\_BLUE; -- Cr component  else  data <= Y\_BLUE; -- Y component  end if;  end if;    -- Reset the column pixel counter at the end of the line  if (cnt\_col\_pixel = 1279) then  cnt\_col\_pixel <= 0;  else  cnt\_col\_pixel <= cnt\_col\_pixel + 1;  end if;  end if;  end process;  end Behavioral; |

|  |  |
| --- | --- |
| **Colors** | **Y-Cb-Cr Value** |
| Black | (16,128,128) |
| White | (235,128,128) |
| Red | (82,90,240) |
| Green | (145,54,34) |
| Blue | (41,240,110) |
| Yellow | (210,16,146) |
| Cyan | (170,166,16) |
| Magenta | (107,202,222) |

## Part-3: Simulation of camera emulator

### Task-1

Q1. Run simulation of *Camera\_Clock\_Generator.vhd* using ModelSim (Copy code to a ModelSim project). No test bench required, because one input signal only. Compare your output with fig.2 and fig.3!



### Task-2

Q1. Run simulation of *Pixel\_Generator.vhd* using ModelSim (Copy code to a ModelSim project). No test bench required, because three input signals only. Check 8-bit output data on expected CB,Y,CR,Y,CB,Y,CR sequence for blue pixel!

## Part-4: Test of HDMI Display Controller on ZedBoard

### Task-1

Q1. First check on file *video\_project\_constraints.xdc* if all the pins are set correctly. Provide a table for all I/O pins and their respective signal names!

|  |  |
| --- | --- |
| **Signal Name** | **Port** |
| HDMI\_CLK | W18 |
| Data\_out[15] | V14 |
| Data\_out[13] | U17 |
| Data\_out[12] | V15 |
| Data\_out[11] | W15 |
| Data\_out[10] | W13 |
| Data\_out[9] | Y15 |
| Data\_out[8] | AA17 |
| Data\_out[7] | AB17 |
| Data\_out[6] | AA16 |
| Data\_out[5] | AB16 |
| Data\_out[4] | AB15 |
| Data\_out[3] | Y14 |
| Data\_out[2] | AA14 |
| Data\_out[1] | AA12 |
| Data\_out[0] | Y13 |
| cam\_resend | T18 |
| Clk\_100M | Y9 |
| Config\_done | T22 |
| De | U16 |
| h\_sync | V17 |
| Resend | P16 |
| Sloc\_hdmi | AA18 |
| Slod\_hdmi | Y16 |
| Switch | F22 |
| V\_sync | W17 |

### Task-2

Q2. Synthesize your VHDL design and implement it on the ZedBoard. For this you **Run Synthesis**, then **Run Implementation**, finally click on **Generate Bitstream**.

|  |
| --- |
| library IEEE;  use IEEE.std\_logic\_1164.all;  entity Pixel\_Generator is  port(  pclk : in STD\_LOGIC;  href : in STD\_LOGIC;  vsync : in STD\_LOGIC;  data : out STD\_LOGIC\_VECTOR (7 downto 0)  );  end Pixel\_Generator;  architecture Behavioral of Pixel\_Generator is  -- Define constants for YCbCr values  constant y\_red : STD\_LOGIC\_VECTOR (7 downto 0) := x"52";  constant cb\_red : STD\_LOGIC\_VECTOR (7 downto 0) := x"5A";  constant cr\_red : STD\_LOGIC\_VECTOR (7 downto 0) := x"F0";  constant y\_blue : STD\_LOGIC\_VECTOR (7 downto 0) := x"29";  constant cb\_blue : STD\_LOGIC\_VECTOR (7 downto 0) := x"F0";  constant cr\_blue : STD\_LOGIC\_VECTOR (7 downto 0) := x"6E";  constant y\_green : STD\_LOGIC\_VECTOR (7 downto 0) := x"91";  constant cb\_green : STD\_LOGIC\_VECTOR (7 downto 0) := x"36";  constant cr\_green : STD\_LOGIC\_VECTOR (7 downto 0) := x"22";  constant y\_yellow : STD\_LOGIC\_VECTOR (7 downto 0) := x"D2";  constant cb\_yellow : STD\_LOGIC\_VECTOR (7 downto 0) := x"F0";  constant cr\_yellow : STD\_LOGIC\_VECTOR (7 downto 0) := x"6E";    -- Define a signal for counting pixels  signal cnt : integer range 0 to 639 := 0;  begin  pixel\_gen : process(pclk)  begin  if falling\_edge(pclk) then  if href = '1' and vsync = '0' then  if (cnt rem 4) = 0 then -- Every fourth pixel  if cnt > 0 and cnt < 160 then  data <= cb\_red;  elsif cnt >= 160 and cnt < 320 then  data <= cb\_blue;  elsif cnt >= 320 and cnt < 480 then  data <= cb\_green;  elsif cnt >= 480 and cnt < 640 then  data <= cb\_yellow;  end if;  elsif (cnt rem 4) = 1 then -- Every fourth pixel after the first  if cnt > 0 and cnt < 160 then  data <= y\_red;  elsif cnt >= 160 and cnt < 320 then  data <= y\_blue;  elsif cnt >= 320 and cnt < 480 then  data <= y\_green;  elsif cnt >= 480 and cnt < 640 then  data <= y\_yellow;  end if;  elsif (cnt rem 4) = 2 then -- Every fourth pixel after the second  if cnt > 0 and cnt < 160 then  data <= cr\_red;  elsif cnt >= 160 and cnt < 320 then  data <= cr\_blue;  elsif cnt >= 320 and cnt < 480 then  data <= cr\_green;  elsif cnt >= 480 and cnt < 640 then  data <= cr\_yellow;  end if;  elsif (cnt rem 4) = 3 then -- Every fourth pixel after the third  if cnt > 0 and cnt < 160 then  data <= y\_red;  elsif cnt >= 160 and cnt < 320 then  data <= y\_blue;  elsif cnt >= 320 and cnt < 480 then  data <= y\_green;  elsif cnt >= 480 and cnt < 640 then  data <= y\_yellow;  end if;  if cnt = 639 then -- Reset counter at the end of each line  cnt <= 0;  else  cnt <= cnt + 1;  end if;  end if;  end if;  end if;  end process;  end Behavioral; |

### Task-3

Q3. Change *Pixel\_Generator.vhd* in a way that you get color strips on your screen as shown in fig.4! Take photo of screen output

# References

|  |  |
| --- | --- |
| [1] | "VHDL Programming by Example" by Douglas L. Perry |
| [2] | "VHDL: Analysis and Modeling of Digital Systems" by Zainalabedin Navabi |
| [3] | "FPGA Prototyping by VHDL Examples: Xilinx MicroBlaze MCS SoC" by Pong P. Chu |
| [4] | "The Zynq Book" by Louise H. Crockett et al. |
| [5] |  |