|  |
| --- |
| Microelectronic Systems WS18/19 |
|  |
| Lab 3: HDMI Design on ZedBoard |
| Report submitted by :  **Bharath Ramachandraiah**  Matrikelnr : 762596  **Bharadwaj Krishnan**  Matrikelnr : 762559 |
|  |
| **MM/DD/YYYY** |

Content

[1 Introduction 3](#_Toc534793479)

[2 Implementation 3](#_Toc534793480)

[Part 1 : Study VHDL Design of HDMI project 3](#_Toc534793481)

[Part 2: Design of Camera Emulation 5](#_Toc534793482)

# Introduction

Data captured by a camera is getting displayed on the HDMI output of the ZedBoard. The camera needs to get connected to the Pmods, the screen to the HDMI transmitter of the ZedBoard. Both interfaces are connected to the PL side of Zynq device.

# Implementation

## Part 1 : Study VHDL Design of HDMI project

**Task 1:**

Task-Description:

Draw a block diagram of the top level RTL design. Name all input and output ports and also all internal signals which connect the two components HDMI\_V1 and IMG\_GENERATOR! (RTL Analysis -> Open Elaborated Design).

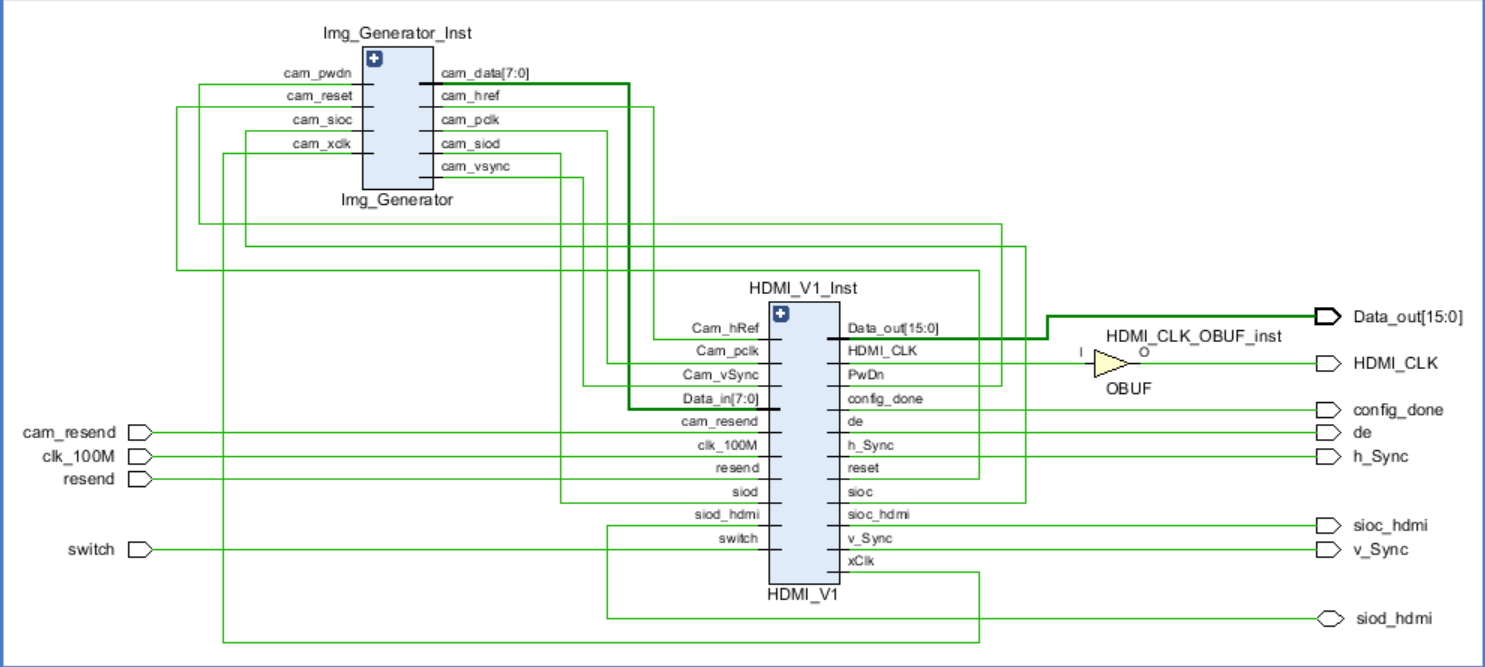


Figure 1: Top Level RTL Design

Analysis:

Figure 1 shows the Top-Level RTL design. The input and output pins are highlighted. The figure shows the input and output pins and the interconnection signals between the modules IMG\_GENERATOR and HDMI\_V1.

**Task 2:**

Task-Description:

Describe the meaning of all output signals of HDMI\_V1!! Use a table format with two columns: signal name and description! For the signals provided to the HDMI transmitter do research on ADV7511.

Elaboration:

|  |  |
| --- | --- |
| **Signal Name** | **Signal Description** |
| Data\_out [15:0] | Video Data Output |
| HDMI\_CLK | Video Clock Output |
| h\_Sync | Horizontal Sync Output |
| v\_Sync | Vertical Sync Output |
| config\_done | Configuration Done (to LED) |
| De | Data Enable Signal for Digital Video |
| sioc\_hdmi | I2C Interface (clock signal) |
| siod\_hdmi | I2C Interface (data) - bidirectional |

Analysis

The description of the output signals of HDMI\_V1 are provided in the table above.

**Task 3:**

Task-Description:

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 (task1)! Use a table format with two columns: signal name and description.

Elaboration:

|  |  |
| --- | --- |
| **Signal Name** | **Signal Description** |
| Cam\_data[7:0] | YUV/RGB video component output data bits from image generator |
| cam\_href | Horizontal camera sync signal |
| cam\_pclk | Camera pixel clock output |
| cam\_siod | SCCB (Serial Camera Control Bus) serial data I/O |
| cam\_vsync | Vertical camera sync signal |

Analysis:

The signal description from the IMG\_GENERATOR is provided in the table above.

## Part 2: Design of Camera Emulation

**Task 1:**

Task Description

When investigating the component Img\_Generator you will find two subcomponents:

**Camera\_Clock\_Generator**

**Pixel\_Generator**

Complete those VHDL models so that data is generated to display **blue pixels** on your screen!

Elaboration:

In order to obtain the color blue on the screen, the values for each component as well as the RGB values is provided in the table below.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Color** | **Y** | **Y\_hex** | **CB** | **CB\_hex** | **CR** | **CR\_hex** | **R** | **G** | **B** | **Display Color** |
| **Blue** | 00100111 | 27 | 11110000 | F0 | 01101101 | 6D | 0 | 0 | 255 |  |

Camera\_Clock\_Generator.vhd is necessary for generating control signals for displaying the image on the screen through the HDMI connection. This file is used to generate the horizontal and vertical synchronization signals.

Pixel\_Generator.vhd is used to generate the pixel with help of the Y, Cb and Cr components. The code for both the vhd files is provided below.

**Camera\_Clock\_Generator.vhd**

*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*

*-- your code*

*SIGNAL vsync\_copy : STD\_LOGIC;*

*SIGNAL href\_copy : STD\_LOGIC;*

*begin*

*PROCESS(pclk)*

*VARIABLE line\_time : integer range 0 to 2047 := 0; -- counts till tline= 784 \* 2 pclk's*

*VARIABLE t\_line : integer range 0 to 1023:=0; -- counts till 510 tline's*

*BEGIN*

*IF pclk'event and pclk = '0' THEN*

*IF (line\_time = 1568) THEN*

*line\_time := 0;*

*t\_line := t\_line + 1;*

*IF (t\_line = 510) THEN*

*t\_line := 0;*

*else*

*-- retain old values*

*END IF;*

*else*

*-- retain old values*

*END IF;*

*-- href generation according to the waveform*

*IF (t\_line >= 20) AND (t\_line <= 500) THEN*

*CASE line\_time IS*

*WHEN 0 => href\_copy <= '1';*

*WHEN 1280 => href\_copy <= '0';*

*WHEN OTHERS => NULL;*

*END CASE;*

*else*

*href\_copy <= '0';*

*END IF;*

*--vsync generation according to the waveform*

*CASE t\_line IS*

*WHEN 0 => vsync\_copy <= '1';*

*WHEN 3 => vsync\_copy <= '0';*

*WHEN OTHERS => NULL;*

*END CASE;*

*line\_time := line\_time +1;*

*else*

*-- retains old value*

*END IF;*

*END PROCESS;*

*vsync <= vsync\_copy;*

*href <= href\_copy;*

*end Behavioral;*

**Pixel\_Generator.vhd**

*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 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*

*begin*

*-- your code*

*pclk1: PROCESS(pclk)*

*---- As each pixel needs two bytes and each byte has Y and Cb/Cr components,*

*-----a count of 4 is needed*

*-- variable pdata\_count : integer range 0 to 3:=0; --during synthesis initialization will be ignored*

*BEGIN*

*IF pclk'event and pclk = '0' THEN*

*IF href = '1' and vsync = '0' THEN*

*-- Blue pixels*

*CASE pdata\_count IS*

*WHEN 0 => data <= "11110000"; -- Cb= 240*

*WHEN 1 => data <= "00101001"; -- y= 41*

*WHEN 2 => data <= "01101110"; -- Cr= 110*

*WHEN 3 => data <= "00101001"; -- y= 41*

*WHEN OTHERS => NULL;*

*END CASE;*

*pdata\_count := pdata\_count + 1;*

*IF (pdata\_count = 4)THEN*

*pdata\_count := 0;*

*else*

*--do nothing*

*END IF;*

*else*

*data <= "01011010";*

*pdata\_count := 0;*

*END IF; -- href*

*else*

*-- do nothing*

*END IF; --pclk*

*END PROCESS pclk1;*

*END Behavioral;*

Analysis

The code for both the vhd files is written and the blue color was visible in the screen. A screenshot has been attached below.



**Task 2:**

Task Description:

Implement your test design on the ZedBoard. 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.

Elaboration

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Pin** | **Signals** | **Output pin** | **Signals** |
| PIN T4 | cam\_test\_data[0] | Data\_out[0] | PIN Y13 |
| PIN U4 | cam\_test\_data[1] | Data\_out[1] | PIN AA13 |
| PIN R6 | cam\_test\_data[2] | Data\_out[2] | PIN AA14 |
| PIN T6 | cam\_test\_data[3] | Data\_out[3] | PIN Y14 |
| PIN Y4 | cam\_test\_data[4] | Data\_out[4] | PIN AB15 |
| PIN AA4 | cam\_test\_data[5] | Data\_out[5] | PIN AB16 |
| PIN AA7 | cam\_test\_data[6] | Data\_out[6] | PIN AA16 |
| PIN AB6 | cam\_test\_data[7] | Data\_out[7] | PIN AB17 |
| PIN AA11 | Data\_in[0] | Data\_out[8] | PIN AA17 |
| PIN AB10 | Data\_in[1] | Data\_out[9] | PIN Y15 |
| PIN Y10 | Data\_in[2] | Data\_out[10] | PIN W13 |
| PIN AB9 | Data\_in[3] | Data\_out[11] | PIN W15 |
| PIN AA9 | Data\_in[4] | Data\_out[12] | PIN V15 |
| PIN AA8 | Data\_in[5] | Data\_out[13] | PIN U17 |
| PIN W12 | Data\_in[6] | Data\_out[14] | PIN V14 |
| PIN V12 | Data\_in[7] | Data\_out[15] | PIN V13 |
| PIN V10 | Cam\_hRef | Config\_done | PIN T22 |
| PIN W10 | Cam\_pclk | de | PIN U16 |
| PIN T18 | cam\_resend | H\_Sync | PIN V17 |
| PIN V9 | Cam\_vSync | HDMI\_CLK | PIN W18 |
| PIN Y9 | clk\_100M | PwDn | PIN Y11 |
| PIN F22 | switch | Reset | PIN AB11 |
| PIN W8 | Siod | Sioc | PIN V8 |
| PIN Y16 | siod\_hdmi | Sioc\_hdmi | PIN AA18 |
|  |  | V\_sync | PIN W17 |
|  |  | xClk | PIN W11 |

Analysis

All the input and output pins and their respective signals are provided in the table above.

**Task 3:**

Task Description:

Change *Pixel\_Generator.vhd* in a way that you get colour strips on your screen.

Elaboration:

In order to obtain the colour strips on the screen, the values for each component as well as the equivalent RGB values are provided in the table below.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Color | Y | Y\_Hex | Cb | Cb\_Hex | Cr | Cr\_Hex | R | G | B |
| Red | 01010011 | 53 | 01011010 | 5A | 11110000 | F0 | 255 | 0 | 0 |
| Blue | 00100111 | 27 | 11110000 | F0 | 01101101 | 6D | 0 | 0 | 255 |
| Yellow | 11010010 | D2 | 00010000 | 10 | 10010010 | 92 | 255 | 255 | 0 |
| Green | 10010001 | 91 | 00110110 | 36 | 00100010 | 22 | 0 | 255 | 0 |

The modified code in the *pixel\_generator.vhd* file is provided below.

**Pixel\_Generator.vhd**

*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 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*

*begin*

*-- Colour Pixels code here*

*pclk1: PROCESS(pclk)*

*---- As each pixel needs two bytes and each byte has Y and Cb/Cr components,*

*-----a count of 4 is needed*

*variable pdata\_count : integer range 0 to 7:=0; --during synthesis initialization will be ignored*

*variable count : integer range 0 to 3:=0;*

*variable pixelpair\_count : integer:=0;*

*BEGIN*

*IF pclk'event and pclk = '0' THEN*

*IF href = '1' and vsync = '0' THEN*

*-----As href will be high for 1280 pclk at which horizontal output per line is processed*

*----- Pixels for each colour should be set during one fourth of the total time ie 1280pclk/2*

*------ RED pixels*

*IF (pixelpair\_count >= 0) AND (pixelpair\_count < 320 )THEN*

*CASE pdata\_count IS*

*WHEN 0 => data <= "11110000"; -- Cb= 240*

*WHEN 1 => data <= "00100111"; -- y= 39*

*WHEN 2 => data <= "01101101"; -- Cr= 109*

*WHEN 3 => data <= "00100111"; -- y= 39*

*WHEN OTHERS => NULL;*

*END CASE;*

*--Blue pixels*

*ELSIF (pixelpair\_count >= 320) AND (pixelpair\_count < 640) THEN*

*CASE pdata\_count IS*

*WHEN 0 => data <= "01011010"; -- Cb= 90*

*WHEN 1 => data <= "01010010"; -- y= 82*

*WHEN 2 => data <= "11110000"; -- Cr= 240*

*WHEN 3 => data <= "01010010"; -- y= 82*

*WHEN OTHERS => NULL;*

*END CASE;*

*---Green pixels*

*ELSIF (pixelpair\_count >= 640) AND (pixelpair\_count < 960) THEN*

*CASE pdata\_count IS*

*WHEN 0 => data <= "00110101"; -- Cb= 53*

*WHEN 1 => data <= "10010000"; -- y= 144*

*WHEN 2 => data <= "00100010"; -- Cr= 34*

*WHEN 3 => data <= "10010000"; -- y= 144*

*WHEN OTHERS => NULL;*

*END CASE;*

*--yellow pixels*

*ELSIF (pixelpair\_count >= 960) AND (pixelpair\_count < 1280) THEN*

*CASE pdata\_count IS*

*WHEN 0 => data <= "10100110"; -- Cb= 166*

*WHEN 1 => data <= "10101010"; -- y= 170*

*WHEN 2 => data <= "00010000"; -- Cr= 16*

*WHEN 3 => data <= "10101010"; -- y= 170*

*WHEN OTHERS => NULL;*

*END CASE;*

*else*

*END IF;*

*pixelpair\_count := pixelpair\_count + 1;*

*IF (pixelpair\_count = 1280) THEN -- 1280*

*pixelpair\_count := 0;*

*else*

*END IF;*

*pdata\_count := pdata\_count + 1;*

*IF (pdata\_count = 4)THEN*

*pdata\_count := 0;*

*else*

*--do nothing*

*END IF;*

*else*

*data <= "01011010";*

*count := 0;*

*pdata\_count := 0;*

*END IF; -- href*

*else*

*-- do nothing*

*END IF; --pclk*

*END PROCESS pclk1;*

*end Behavioral;*

Analysis

The color strip obtained on the screen is shown below.

