ESKİŞEHİR TECHNICAL UNIVERSITY

EEM464

System-on-Chip (SoC) Design

A Trip from PMOD to VGA

Project Final Report

Bilgin SELİMOĞLU

34312459898

1. Abstract

The project is about video processing with using Software Development Kit (SDK) and Vivado. Hardware design of the project was done with Vivado and Software design was done in SDK. As camera, OV7670 is used. The OV7670 can output 12-bit RGB in 640x480 resolution at 30 fps. This camera cannot compete with against today's sophisticated commercial cameras, but it provides a platform to potentially try new ideas and enhancements, it is also cheap to take risk against break down.

The purpose of this project is to design a simple digital camera system to implement some of the main concept related to digital design with SoC, video formats, CMOS cameras, basic image processing algorithms embedded programming of microcontrollers.

2. Introduction

Cameras are used to control, monitor, or perform an ongoing work in many fields of work areas. Such a large area of use of cameras is due to their easy usability and the availability of models in different price ranges to suit the budget. The most important of these areas of use is undoubtedly the defense industry. It is very important to control an area, to reach unreachable places, and most importantly, to remove country territory from unnecessary human observations, to minimize the margin of error and to minimize the costs in this area. Another area of application is a highly preferred device in autonomous working bands in the manufacturing departments of factories. In the journey of the product on the autonomous working bands, it accelerates most operations, from where to go, to what to stick on it as label. The ease of use and the low cost have made the cameras very popular. These devices need to be controlled. The machine learning was further improved by taking the autonomous working principle in this field further. The ability to make decisions on an image using various image filters in the above fields is quite common today. This machine learning, known as DNN, has become easy to use, with high predictive capabilities. However, these estimates should be able to operate at high frequencies to operate at high speeds. The process slows down as calculations increase. Therefore, the costly part of the work is the necessity of the equipment where these processes can be done at high speeds. At this point, this project provides flexibility to users because of their operability on SoC platform. This flexibility is that hardware and software design can be run together, and the camera can be operated for purpose. This project includes Section 3, hardware and hardware design; Section 4, software design; Section 5 Conclusion.

3. Hardware and Hardware Design

3.1. OV7670 CMOS Camera Module

For the project, OV7670 camera module is chosen because of its low cost. The OV7670 is a CMOS image sensor + DSP that can operate at a maximum of 30 fps and 640x480 (VGA) resolutions, which is the equivalent of 0.3 Megapixels. The captured image can be pre-processed by the DSP before sending it out. This preprocessing can be configured via the Serial Camera Control Bus (SCCB) or through I2C interface. You can download its datasheet from [1] and implementation guide from [2].

Figure 1: OV7070 camera module

The camera module comes with a 9x2 header, the pin diagram is shown below:

3V3	**GND**
SIOC	**SIOD**
VSYNC	**HREF**
PCLK	**XCLK**
D7	**D6**
D5	**D4**
D3	**D2**
D1	**D0**
RESET	**PWDN**

The I/O signals of the camera module are listed in the following Table 1.

Pin	Type	Description
3V3	Supply	Power supply
GND	Supply	Ground level
SIOC	Input	Serial command bus clock
SIOD	Input/output	Serial command bus data
VSYNC	Output	Vertical synchronization (Active High)
HREF	Output	Horizontal synchronization (Active High)
PCLK	Output	Pixel clock
XCLK	Input	System clock
D0-D7	Output	Pixel data
RESET	Input	Device Reset (Active Low)
PWDN	Input	Device Power Down (Active High)

Table 1: I/O Signals

3.2. Pixel Format

The OV7670 camera module uses RGB color model. In the RGB color model, any color can be decomposed in Red, Green and Blue light at different intensities. Because a color can be made by mixing Red, Green and Blue, it is called the RGB color system or model. Using this model, each pixel must be stored as three intensities of these red, green and blue lights. The most common format is RGB888, in this format each pixel is stored in 24 bits, the red, green and blue channels are stored in 8 bits each: RRRRRRRRGGGGGGGGBBBBBBBB

This means that the intensity of each light can go from 0 to 255, where 0 is the absence of light, and 255 is the maximum intensity.

The formats used by the OV7670 are the RGB565, RGB555 and RGB444. The difference with the RGB888 format, is the number of bits assigned to each channel. For example, in the RGB565 format, the red channel is stored as 5 bits, the green channel as 6 bits and the blue channel as 5 bits. These formats take less memory when stored but in exchange sacrifice the number of colors available.

3.3. Signaling

The OV7670 sends the data in a parallel synchronous format. To get any data out of the OV7670, is necessary to supply a clock signal on the XCLK pin. According to the datasheet, this clock must have a frequency between 10 and 48 MHz. After a clock signal has been applied to the XCLK pin, the OV7670 will start driving its VSYNC, HREF and D0-D7 pins.

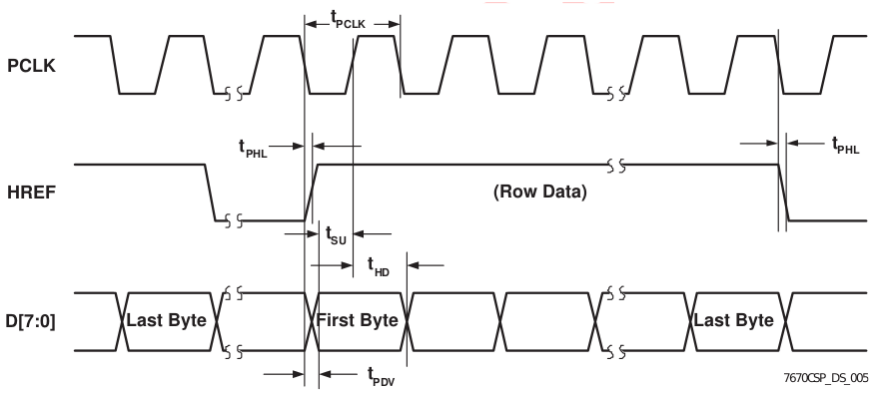

Figure 2: Horizontal Synchronization

The D0-D7 must be sampled at the rising edge of the PCLK signal. D0-D7 must be sampled only when HREF is high. Also, the rising edge of HREF signals the start of a line, and the falling edge of HREF signals the end of the line. All these bytes sampled when HREF was high, correspond to the pixels in one line. Note that one byte is not a pixel, it depends on the format chosen.

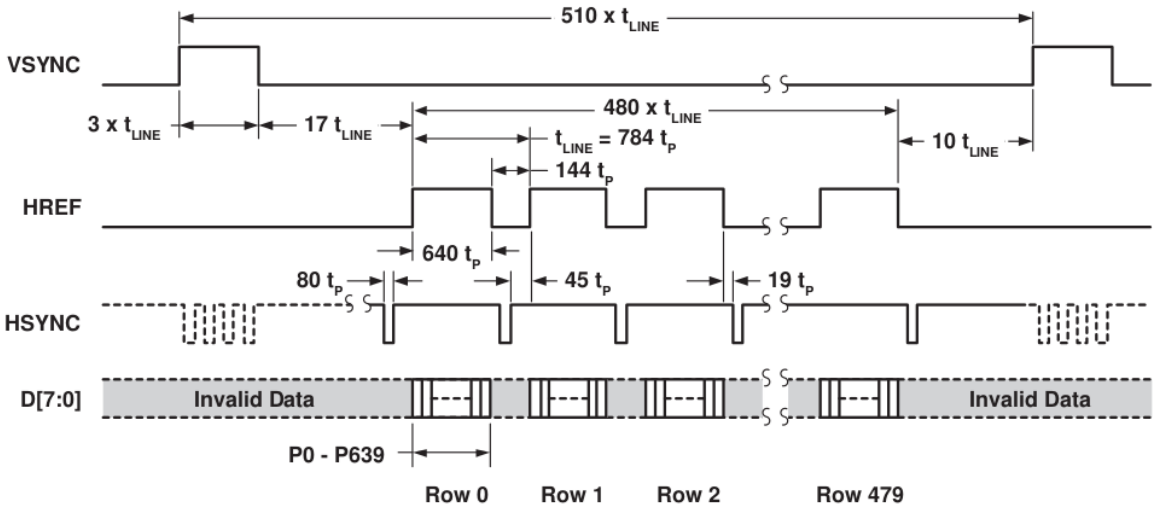

Figure 3: VGA Timing

The image above shows the signals for a "VGA" (640 x 480) frame. During HSYNC high state, we must capture 640 pixels, equivalent to a line. The 480 lines, equivalent to a frame, are captured during the low state of VSYNC. This means that the falling edge of VSYNC signals the start of a frame, and its rising edge signals the end of a frame.

3.4. ZedBoard Connection

In Figure 4 shows the connections between ZedBoard and OV7670, Monitor. Camera connections are made using PMOD connector. For camera pins two PMOD connectors used. They are JA1 and JB1.

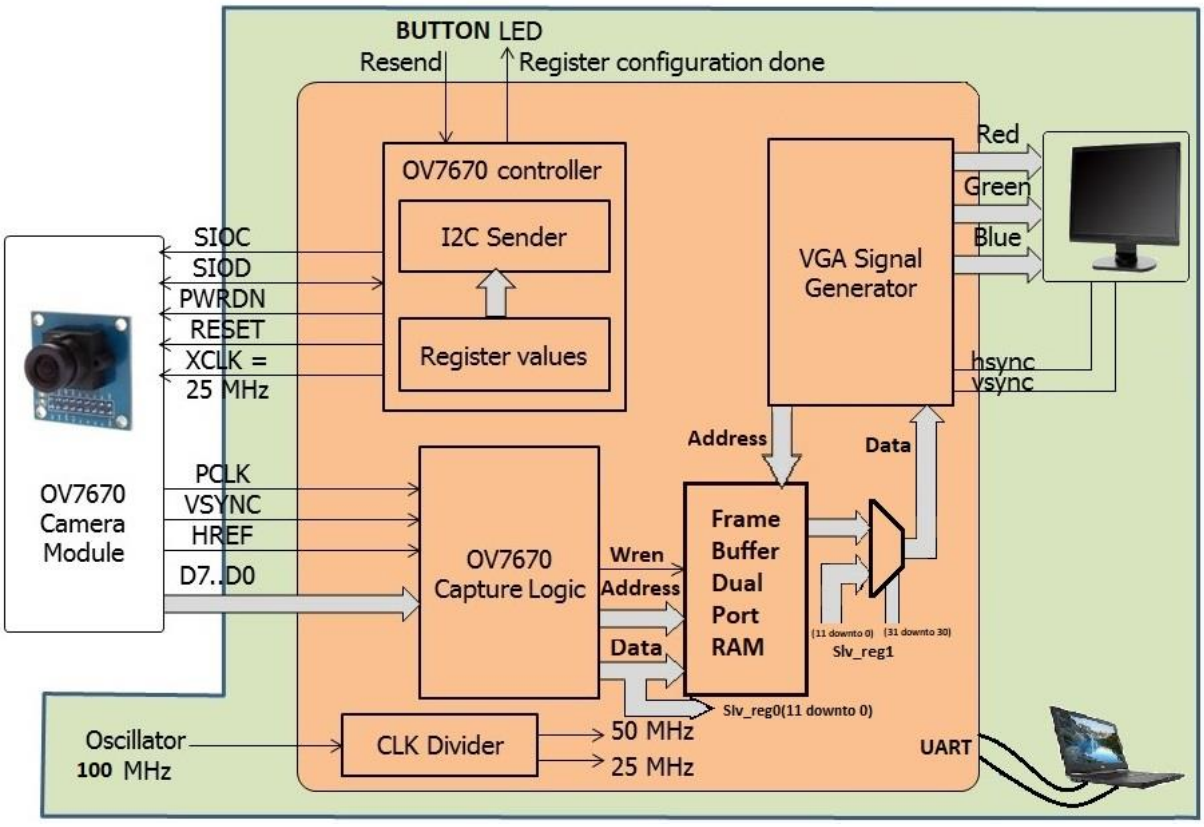

Figure 4: System Connections

Pmod is a simple interface type for adding small peripheral modules (hence the name **Pmod** = **P**eripheral **mod**ule), using either 6-pin or 12-pin interfaces. The name was standardized by, and is a trademark of, *Digilent Inc.*, but other companies such as *Maxim Integrated* also produce Pmods. Typical Pmods are sensors, actuators, data converters, and user I/O devices. There are also some communications transceivers available. Pmods can also facilitate direct wire connections [3].

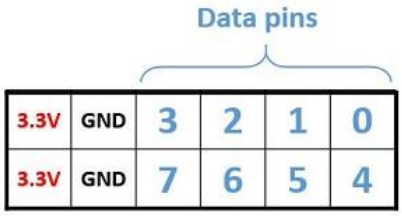

Figure 5: PMOD Connections

According to standard PMOD Connections (shown in Figure 5) related camera pins connected to Zedboard PMOD connectors. Before connecting the camera pins to Zedboard PMOD connector ZedBoard User Guide is researched and Zynq pins are connected with related OV7670 camera pins. In Figure 6, shows the connection to Zedboard's PMOD.

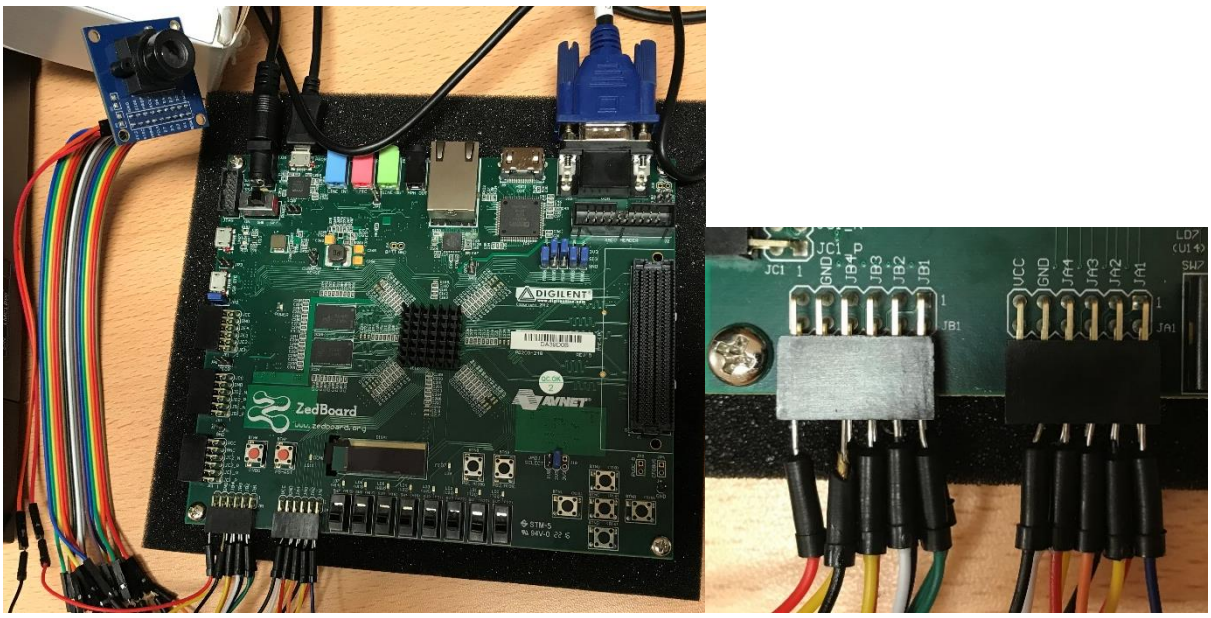

Figure 6: Camera Connection to PMOD

Up to this point, information was given about the system connections and the hardware used. Figure 4 represents the hardware design of the system. Figure 6 corresponds to Figure 7. In Figure 7 shows the Block Design in Vivado. Zynq Processing System Processor System Reset, AXI interconnect

and OV_7670 AXI lite IP can be seen easily. The project includes processing system and programmable logic part of the SoC.

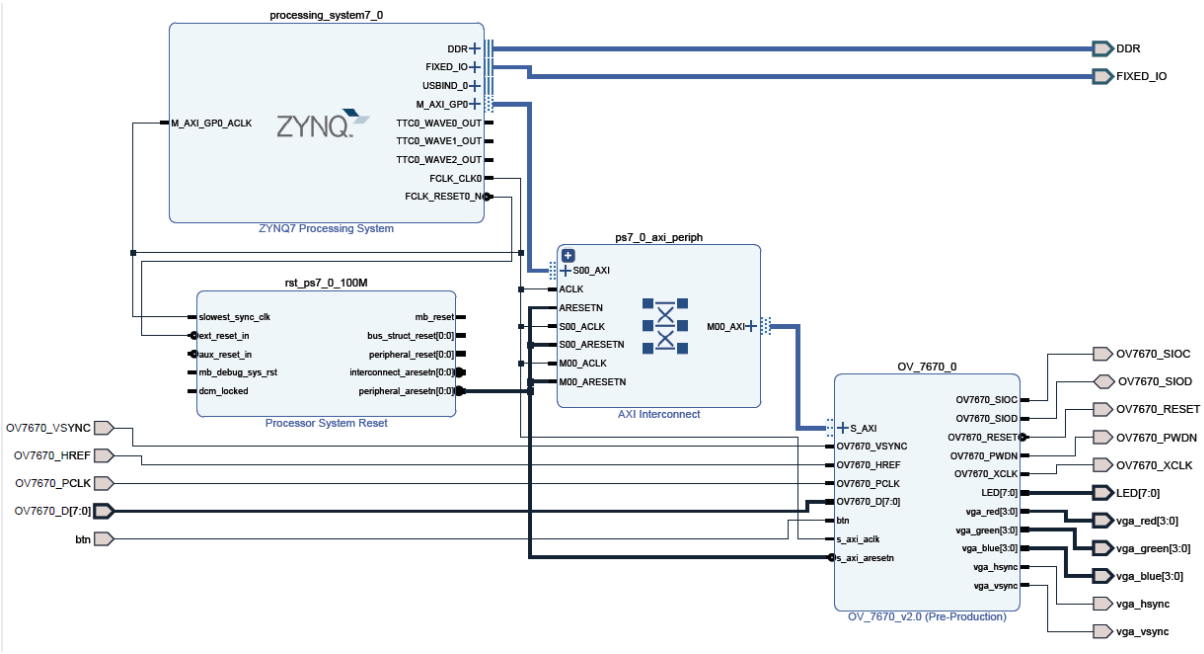

Figure 7: Block Design of Project

3.5. OV_7670 Custom IP

The purpose of the creating this IP, to access the pixels from SDK by using AXI interconnect. To create OV_7670 Manage IP toolbox is used. Provided AXI lite slave peripheral template and custom IP source code are used to create OV_7670 custom IP. After open the template VHDL code, it was updated for inputs and outputs of the project. As input VSYNC, HREF, PCLK, D[7:0] and button were added to entity of the IP. In the same way output signals of the project were added.

System VHDL codes were included to OV_7670_v2_0_S_AXI template VHDL code. OV_7670 custom IP has 4 registers to communicate from SDK. Each register has 32-bit data width. The OV_7670 custom IP has slave interface. Interface Type is Lite.

The components of the OV_7670 custom IP are clocking, debounce, i2c sender, OV_7670 capture, OV_7670 controller, OV_7670 registers, OV_7670 top and VGA as shown at Figure 4. The OV_7670 capture component takes the inputs from OV_7670 camera module and according to VSYNC and HREF signal it arrange the addr, data out and we signals and these signals goes through the block memory. Block memory (frame buffer) has Simple Dual Port 12-bit width. After takes a frame (640*480=307200 pixels) frame pixels goes VGA component. At this point, a multiplexer was added, and output of the multiplexer goes VGA component as input. Inputs of the multiplexer are shown in Figure 4. Registers of the OV_7670 custom IP were used in this point. Slv_reg1's 31 and 30 bit controls

the multiplexer output. One of the inputs of multiplexer is frame buffer output, other one is slv_reg1's 11 downto 0 bits. So, input of the multiplexer can be controlled from SDK. Other register slv_reg0 takes the frame buffer output to can be used in SDK. Controller component of the OV_7670 custom IP has two components inside itself. OV_7670 register and i2c sender are the components of the OV_7670 custom IP. OV_7670 register is setting for the OV_7670 camera module. It sets RGB pixel format, PCLK scaling, HREF start, HREF stop etc. I2c sender sends the commands to the over an i2c interface. The sequence for the SIOC signal. After the component settings are done OV_7670 custom IP was repackaged.

3.6. Zynq Processing System

Master AXI GP0 interface was used to communicate with custom IP from SDK. For clock Configuration of the PS part, FCLK_CLK0 is enabled and the frequency is 100 MHz.

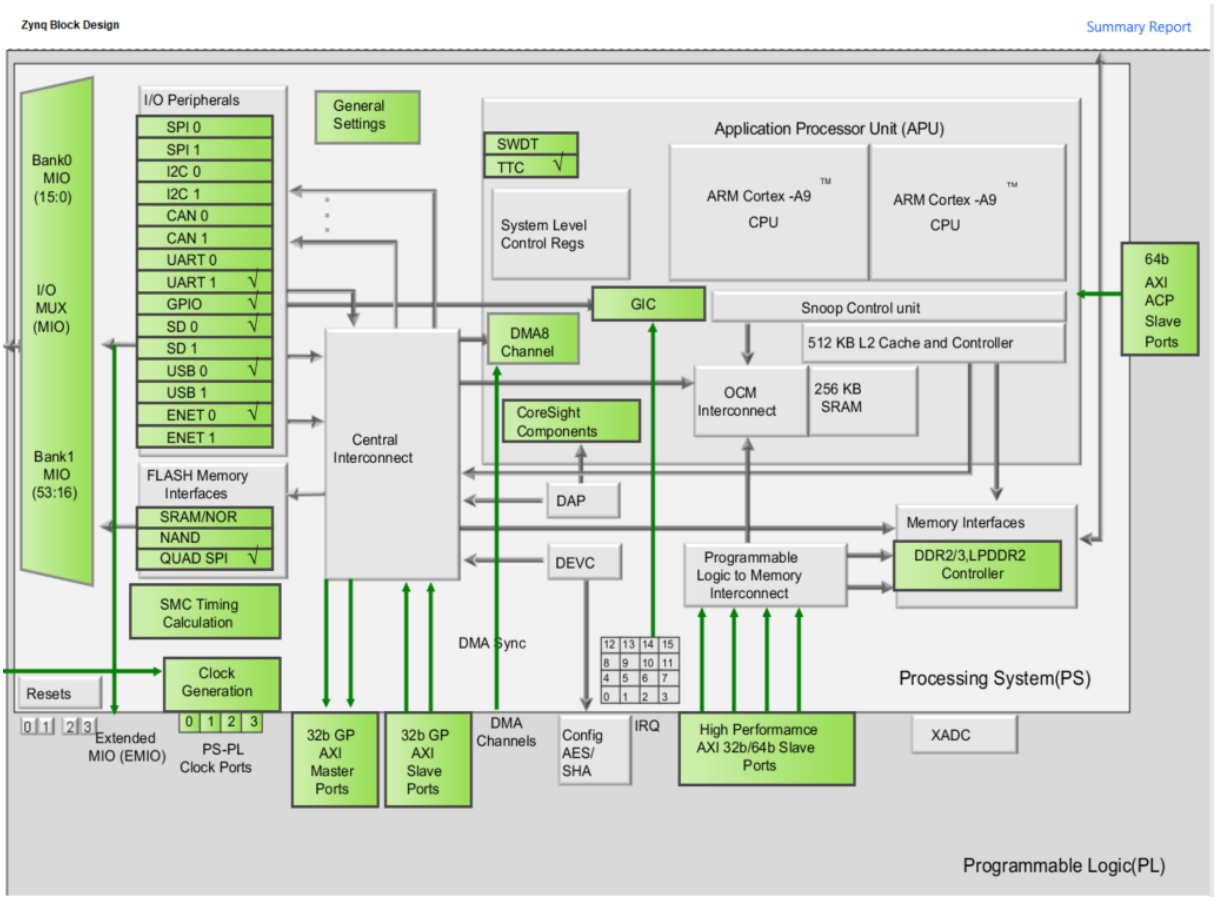

Figure 8: ZYNQ7 Processing System

3.7. AXI Interconnect

Interconnect is effectively a switch which manages and directs traffic between attached AXI interfaces. The interfaces are S_AXI port of the OV_7670 and M_AXI_GP0 of Zynq processing system.

Hardware design of the project was completed at this point. Software design of the project will start now.

4.Software Design

Software design of the project includes the controlling of the VGA inputs. Thanks to the register of the OV_7670 custom AXI lite IP, we can read or write pixel values of frames. User can apply what he wants to pixel values. Some of the operations are;

- Remove any color from pixel
- Controlling any pixel value
- Visualization video flow specific color

To apply what user wants to do is up to him/her. There is no limitation in this. But if the process is complicated then there is a problem because to do operation on pixels, takes time and then frame tingles. If the user wants to do complicated operation, then it has to be done in hardware not on chip.

5.Conclusion

This project is related to design a simple digital camera system to implement some of the main concept related to digital design with SoC. Hardware and Software design of the project are shows that end of the SoC Design course, designing an embedded system that includes hardware and software codesign is possible which learned from course. The project's contribution to me was to master ZedBoard's features and learn how to use them most effectively.

In content of the course, how the Hardware and Software co-design is done is the most important issue. Thinking about how the system will work before starting to build a system greatly affects the easy progress of the study. My project work plan continued with this thought. So, I did so many researches about the project. I saw very different way of solution to this project. Some of them better than my solution but because of the time limit and in terms of knowledge, I decided to solve in this way of solution. If I have more time and more knowledge, I would try to do this project with using VDMA and also, I would add this project some video processing filters. Apart from what we have seen in class, I learned how to make VGA connections, PMOD connections and how a simple camera used with the SoC.

6. References

[1] OV7670 Datasheet (of 2006). This is the simpler and cheaper version, without a FIFO memory on the camera's small board. http://www.cutedigi.com/pub/sensor/Imaging/OV7670-Datasheet.pdf

[2] OV7670 Implementation Guide and Schematic Diagram.
http://www.haoyuelectronics.com/Attachment/OV7670%20+%20AL422B%28FIFO%29%20Camera%20Module%28V2.0%29/OV7670%20Implementation%20Guide%20%28V1.0%29.pdf
http://www.haoyuelectronics.com/Attachment/OV7670%20+%20AL422B%28FIFO%29%20Camera%20Module%28V2.0%29/OV7670%20+%20AL422B%28FIFO%29%20Camera%20Module%28V2.0%29%20Schematic.pdf

[3] Masaryk University, "Zynq Book eBook", webpage.
Available: https://is.muni.cz/el/1433/jaro2015/PV191/um/The_Zynq_Book_ebook.pdf