Embedded Systems Lab 3 Report

Hong-Bin Yang, Hsu-Li Huang, Tz-Ching Yu

December 11, 2022

1 Introduction

The final objective of the project is to construct an entire system that is capable of capturing frames with the camera module, TRDB-D5M, and displaying captured frames using the LCD module, LT24, by connecting these modules to the FPGA board where a softcore CPU, memory, and custom interfaces will be programmed. Therefore, the goal of the laboratory is to develop custom interfaces for the LCD and camera. As our group is responsible for the camera module, we will focus on the system design for it. In the following sections, we will discuss the details of the module, TRDB-D5M, to motivate the proposed design and we will then present our overall system diagram and detailed connections. Finally, we will elaborate on our proposed design of the custom interface for the module and we will also present additional components and configurations required for the whole system.

2 TRDB-D5M

The camera module is capable of capturing images up to 2592 × 1944 and can be set to a different resolution by setting the corresponding registers. The image organization of the camera is shown below.

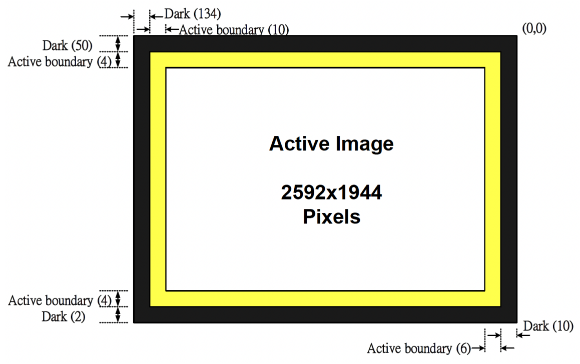

Figure 1: Captured image organization

Besides the active image region, there are also dark and active boundary regions. Valid data of the active image is read progressively scanning the region. The frame valid and line valid signals will be set to high in normal mode and can be used as timing signals.

Each pixel is a 12-bit data and a pixel of the image represent one of the color (R, G, B), in form of a classic Bayer Filter array. Pixels in the first row of the image are organized as a green pixel followed by a red pixel, and the second row of the image is sequenced as a blue pixel followed by a green pixel. The pixel structure is as illustrated in Figure 2.

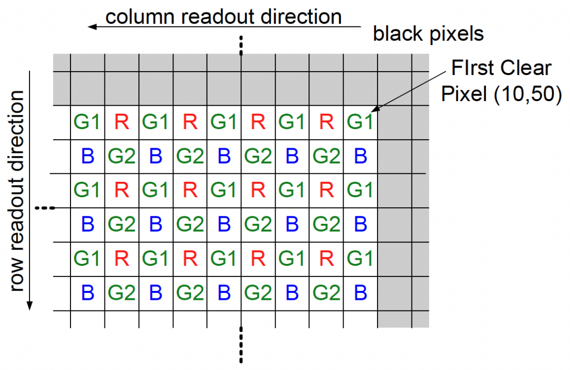

Figure 2: Pixel sequence of normal readout

2.1 Down Sample and Demosaicking

At this stage, the image is a single channel image with size 640×480 in the Bayer pattern. To reduce the size to 320×240 and demosaicking the Bayer filter array, four pixels are combined in one by taking R and B values directly and averaging G_1 and G_2, as shown in Figure 4.

2.2 Pixel Output

Although the camera uses 12 bits to store the pixel value for each color, since the LCD takes only 16 bits as input (5 bits red, 6 bits green, and 5 bits blue), we decided to discard the unused bits by keeping only the most significant bits. This process is performed in the camera interface module.

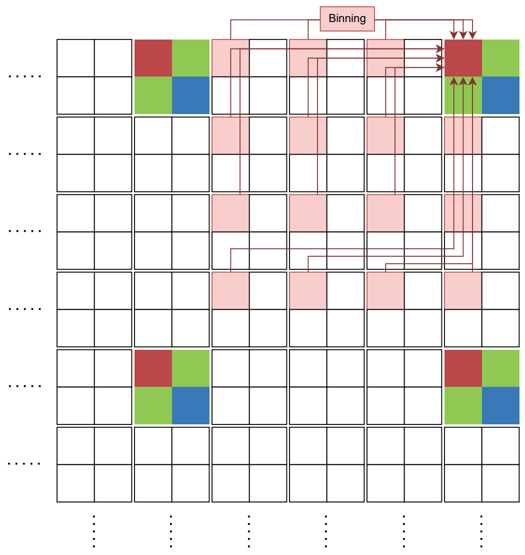

Figure 3: Pixel readout when $Row\_Bin = 3, Row\_Skip = 3, Column\_Bin = 3, Column\_Skip = 3$.

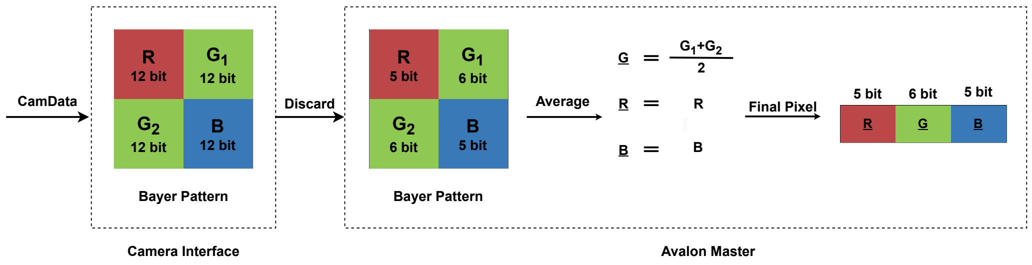

Figure 4: Image Subsampling and Pixel Data Demosaicking

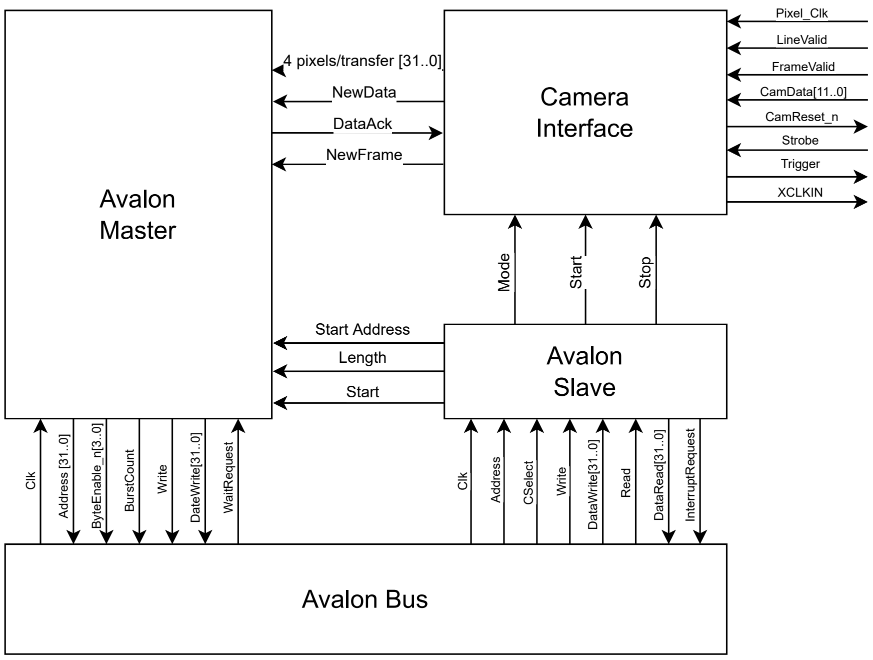

Figure 5: Camera Controller Architecture

3 Proposed Design

3.1 Camera Controller

The camera controller is composed of several components, including an Avalon master acting as a DMA module responsible for transferring pixel data to memory, an Avalon slave which should be programmed by the processor to configure the controller, and a camera interface handling the acquisition of camera pixels. The register map of the component is listed in Table 1. More specifically, the camera interface will directly receive pixels from the camera module, and the received pixels will be put in temporary storage in an ordered fashion. As we will assemble 4 pixels of R, G1, G2, and B as one composite pixel, there will be separate buffers for each color. The assembled pixel will then be transferred to the Avalon master where each pixel will then be sent to memory by the DMA. The function of the controller can be summarized as a finite state machine and the according diagram is shown in Figure

6

Address	Register	Reset Value	Description
0	Start	0	Set to start capturing
1	Stop	1	Set to stop capturing
2	Base Address	0	Specify the base memory address to write to
3	Length	0	Specify the length of memory region to write
4	Mode	0	0: Continuous mode / 1: Snapshot mode

Table 1: Camera Controller Register Map

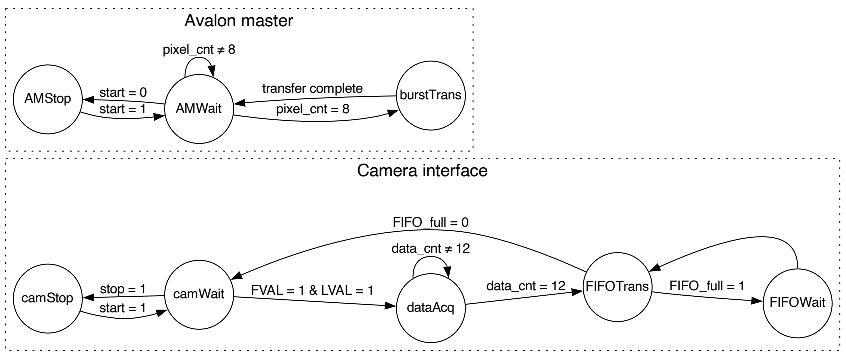

Figure 6: Finite State Machine for Camera Controller

3.2 Camera Configuration

The display resolution of the LCD is 320×240, so we set the resolution of the camera to be 640×480 by setting the column size, row size, binning, and skipping registers according to the following values.

Address	Register Name	Value
0x3	Row Size	1919
0x4	Column Size	2559
0x22[5:4]	Row_Bin	3
0x22[2:0]	Row_Skip	3
0x23[5:4]	Column_Bin	3
0x23[2:0]	Column_Skip	3

Table 2: Camera register setup

These registers will be set via I2C writes. As I2C design is already provided, we will not discuss it any further. By setting *Row_Skip* and *Column_Skip* to 3, the camera would skip three sets of pixels during readout, resulting in an output image with resolution equals to 640×480. However, this would cause an undesired aliasing effect. To obtain a smoother image, we can set *Row_Bin* and *Column_Bin* to 3, so the camera module would combine the neighbor pixel value by averaging, thus the aliasing effect would be eliminated. The pixel readout is shown in Figure 3.

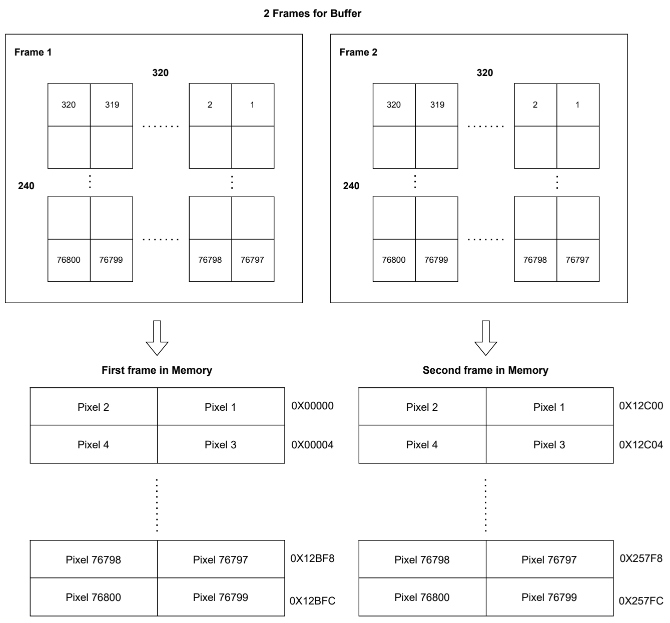

Figure 7: Frame Buffer and Memory Organization

3.3 Memory Organization

After discussing with the LCD group, we decided to reserve a continuous memory region of the size of two frames. The idea is to write to the memory space for two frames alternatively. The Avalon master of the camera controller will put pixels starting from the base address until it reaches the boundary of the region. It will then wrap around the address to the base address and start to write pixels from that address again. As for the LCD Avalon master, with the base address, it can read from the address until the size of a frame has been read, and the LCD controller will then know the next pixel will be the data of the next frame.

To be specific, since each pixel value is 16 bits, we will need $320 \times 240 \times 16 bits = 76800 \times 16 bits = 153600 bytes$ for each frame. To increase the efficiency, 2 pixels are stored in one address. For example, Pixel_1 is stored in Base_Address [15:0] and Pixel_2 is stored in Base_Address [31:16]. Thus, each frame needs $\frac{153600}{2} = 79800 bytes = 0x12C00 bytes$ of memory.

Therefore, the first frame will be stored in Base_Address + 0x00000 to Base_Address + 0x0x12BFC; while the second frame will be stored in Base_Address + 0x0x12C00 to Base_Address + 0x0x257FC. The design is shown in Figure 7.

3.4 System Design

In addition to usual components, including NIOS-II processor, on-chip memory, etc., the system consists of I2C and custom camera interface, and the system diagram is shown in Figure 8. As discussed earlier, I2C is used to configure the camera module and a custom camera controller is used for data acquisition and transfer.

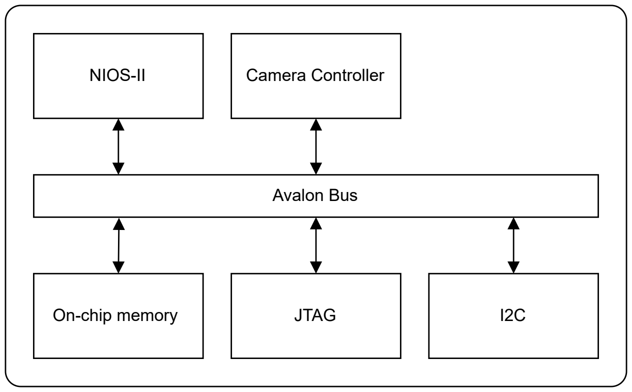

Figure 8: System Block Diagram with Camera Controller

3.4.1 Pin Connection

The camera module will be connected to the **GPIO1** 2×20 GPIO connection on the FPGA board. Hence, the data will be transferred through pin 16, 15, 14, 13, 10, 9, 8, 7, 6, 5, 4, and 2, and the I2C data and clock will be located at pin 26 and 27 respectively.

4 Conclusion

In this lab, we developed a conceptual design of the camera controller that will be implemented in the following weeks. There will surely be modifications to the design along the process, as there will be complications that we do not realize at the moment. However, this document will serve as a great starting point and a reference for the project.