Embedded systems Laboratoy 3
Design of an Avalon camera controller

Snoeijs, Jan
EPFL
`jan.snoeijs@epfl.ch`

Spieler, Michael
EPFL
`spieler.micheal@epfl.ch`

November 2017

1 Introduction

For this lab work we chose to design the component for the Terasic THDB-D5M camera. In this document we detail the complete design of the component and explain the synchronization between our component and an Avalon interfaced LCD controller. We will also present how the NIOS II processor should be programmed to control the hardware component and handle the synchronization between camera and LCD.

2 Camera controller overview

In this section we present the system architecture and how it is divided in the different sub-components and their interaction.

2.1 High-level architecture

Figure 1 illustrates the full system architecture view. A NIOS-II processor runs with data and code form the internal SRAM memory. The Camera is connected to our custom camera controller. The camera controller is configured through an Avalon slave interface. Once configured it acquires image data from the camera, processes it into a more usable format and forwards it over an Avalon master interface to an external SDRAM. The SDRAM is connected through an Avalon Slave SDRAM controller. Finally the image data can be read from SDRAM by a LCD controller to display it on a LCD display.

This document describes the design of the camera controller as well as the interface with the NIOS-II processor and LCD controller.

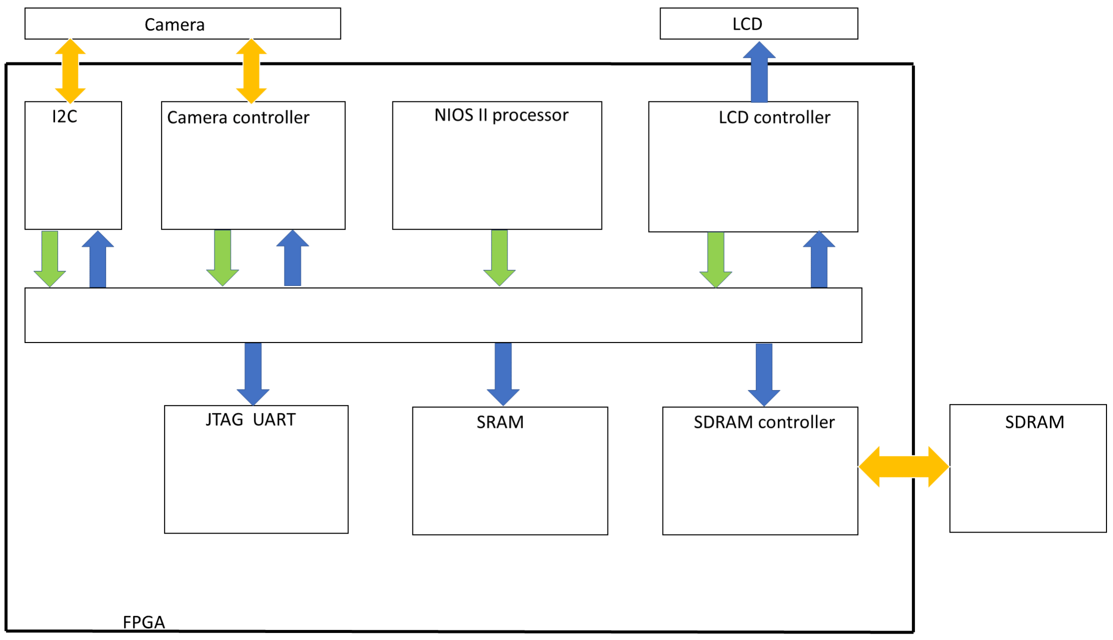


Figure 1: Full system architecture

2.2 Camera controller description

Our camera component is divided into three main subcomponents. The first one is an Avalon slave component, containing the readable/writable registers, which are accessible to the processor. These registers store configuration data for the other subcomponents. The second subcomponent is the camera interface, which receives the camera sensor data, processes it to obtain desired RGB pixel values and finally forwards them into a FIFO buffer. A master Avalon component is the third block of our design. This block is responsible for reading pixel data from the FIFO and transferring it in a convenient format to the DRAM memory by using Avalon bus burst transfers. To these three main parts of our system we include a PLL module to generate a clock signal `XCLKIN` for the camera. The FIFO between camera interface and Avalon master component is dual-clock and allows to synchronize between the camera and the main clock domains. Finally there is an I2C Avalon Slave which allows camera configuration by the NIOS-II processor.

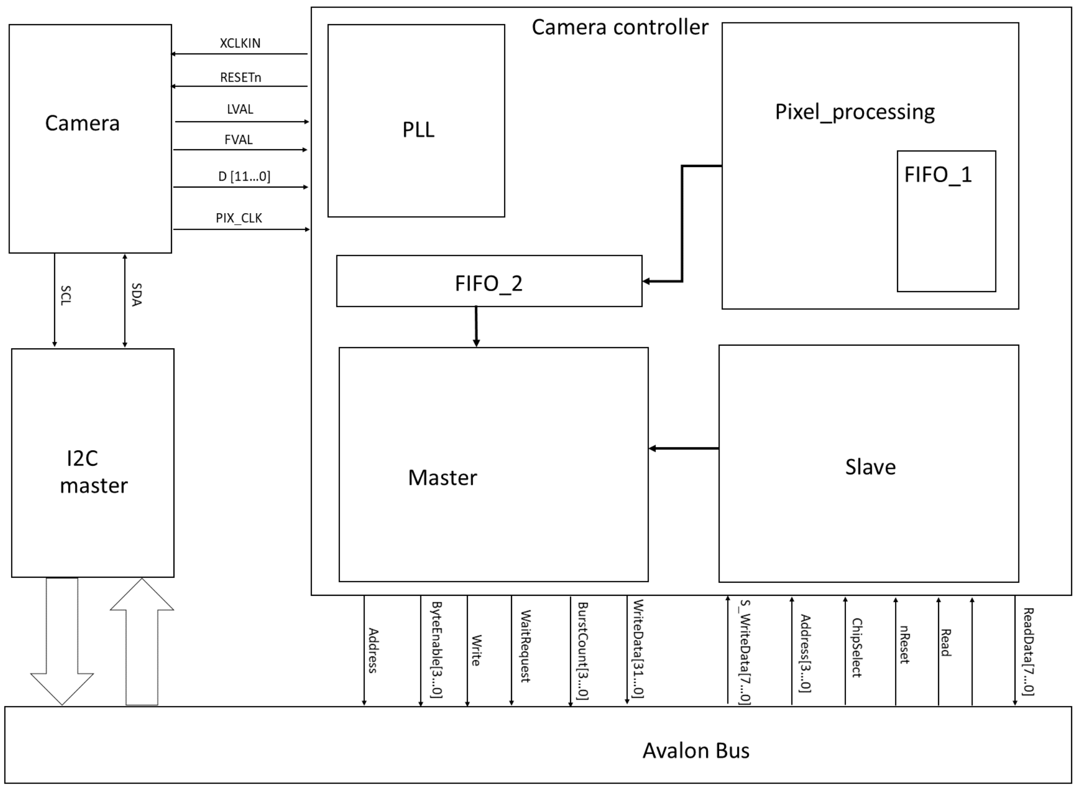


Figure 2: Custom IP block diagram of the camera controller

3 Slave component

The Avalon slave component consists of a set of configuration and control register accessible through an Avalon slave interface. It also provides an interrupt line to notify the NIOS-II CPU of start and end of image reception. Furthermore it provides the image destination address to the master component and controls the internal enable signals and the camera reset line. Figure 3 shows the block diagram with the in- and out-going signals and the internal registers.

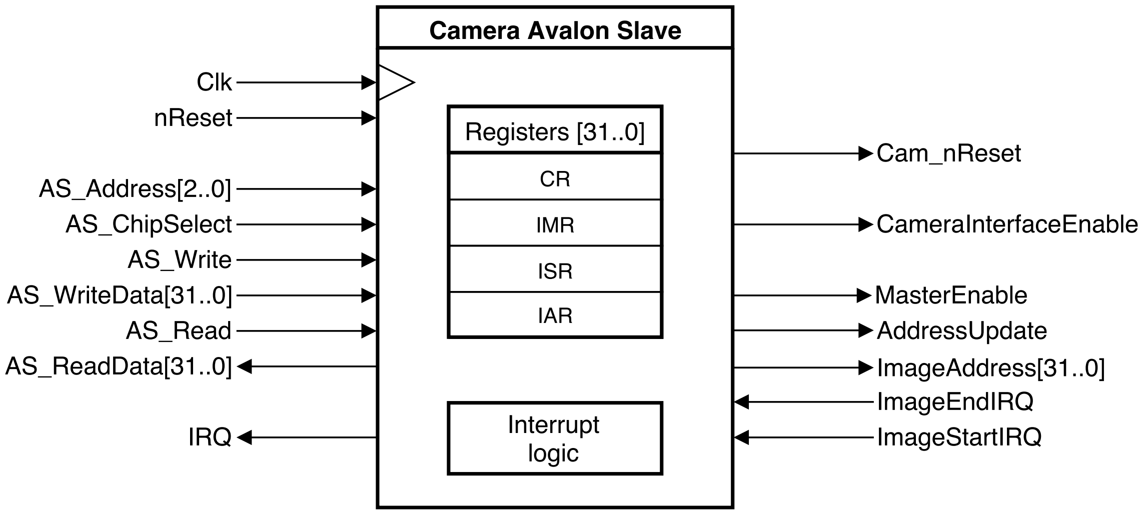


Figure 3: Camera Avalon slave block diagram

3.1 Interrupts

The camera controller provides an interrupt line with two configurable interrupt sources for start and end of image reception. The interrupts can be enabled in the Interrupt Mask Register (IMR).

The camera interrupts can be used by the NIOS-II software to handle the image buffers.

3.2 Register map

3.2.1 Control Register (CR)

Address offset: 0x00
Reset Value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CE	EN

Reserved	Bits 31:2, read only Reserved, read as 0
CE	Bit 1, read/write Camera Enable. Controls the camera reset line. 0: Camera is held in reset. 1: Camera is enabled.
EN	Bit 0, read/write Enable the Camera controller. 0: Camera controller is disabled. 1: Camera controller is enabled.

3.2.2 Interrupt Mask Register (IMR)

Address offset: 0x01
Reset Value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														SIE	EIE

Reserved	Bits 31:2, read only Reserved, read as 0
SIE	Bit 1, read/write image reception Start Interrupt Enable. 0: Interrupt is inhibited. 1: Interrupt is generated when the bit SIF=1 in ISR register.
EIE	Bit 0, read/write image reception End Interrupt Enable. 0: Interrupt is inhibited. 1: Interrupt is generated when the bit EIF=1 in ISR register.

3.2.3 Interrupt Status Register (ISR)

Address offset: 0x02
Reset Value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														SIF	EIF

Reserved	Bits 31:2, read only Reserved, read as 0

SIF Bit 1, read/write
Start Interrupt Flag. Set by hardware when the reception of a new image frame starts. This bit must be cleared by software by writing a 1.

EIF Bit 0, read/write
End Interrupt Flag. Set by hardware when the image data is completely written to memory. This bit must be cleared by software by writing a 1.

3.2.4 Image Address Register (IAR)

Address offset: 0x03
Reset Value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IA[31:16]															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IA[15:0]															

IA Bits 31:0, read/write
Image Address. Destination address for the 320x240x2 = 153600 byte long image buffer. Note: This register should be written only between image receptions to guarantee a correct address update. Ideally this is done in the "image reception end interrupt" ISR.

4 Camera interface component

The camera interface component acquires the 12-bit data from the camera sensors and transforms it to a 16-bit data for each pixel which are then forwarded into a FIFO buffer.

4.1 Data reception

The THDB-D5M camera provides a 12 bit data signal, a clock PIXCLK and control signals LVAL and FVAL for marking the valid image data window. The image is transmitted line by line from top to bottom as shown in figure 4.

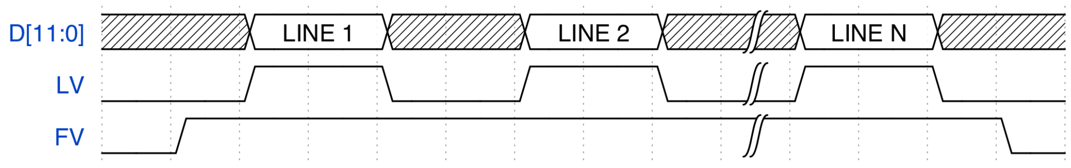


Figure 4: Image transmission with LineValid and FrameValid signals.

Each line contains the camera sensor data for the individual colors red, green and blue. Figure figure 6 shows the bayer pattern composition for each pixel consisting of 4 color sensors: Red (R), green1 (G1), green2 (G2) and blue (B). Pixels are separated over two lines: R and G1 on the first and G2 and B on the second. Thus, to receive a line of pixels we have to receive 2 lines of color sensors as shown in figure 5.

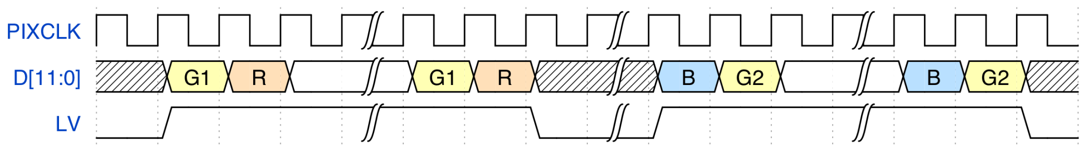


Figure 5: Color sensor line transmission with bayer pattern ordering

4.2 Pixel data transformation

To store the pixel values we decided for a compact 16bit format for one RGB pixel value consisting of 5bits red, 6bits green and 5bits blue, as illustrated in figure 6. Thus we only consider the 5

MSbits and simply discard the remaining LSbits from the sensor data. By combining the two 5bit green values G1 and G2 we can simply add them together and obtain a 6bit value[1].

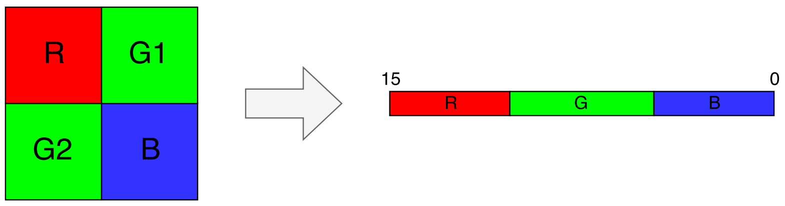


Figure 6: Bayer pattern to 16bit RGB (5bit,6bit,5bit) transformation.

4.3 Camera interface overview

The camera interface component implements the process described above. It receives the camera data and control signals and outputs the transformed 16bit RGB pixel values to the pixel FIFO buffer. The component consists of 3 main blocks: A FIFO to store a line of color sensor values, a processing unit to convert the bayer pattern to the RGB value and finally a finite state machine (FSM) that controls the two components and the output signals. A diagram of the camera interface composition is shown in figure 7.

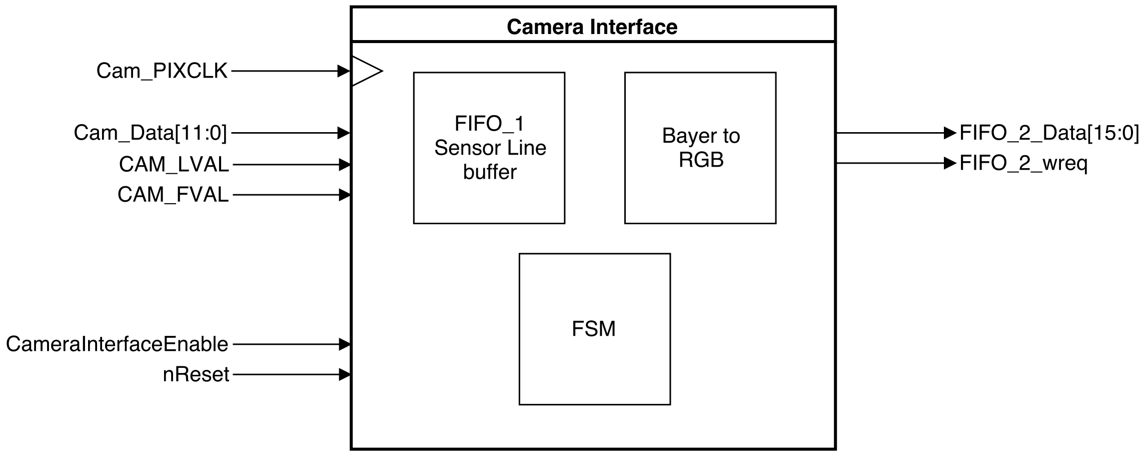


Figure 7: Camera interface block diagram

4.3.1 Bayer to RGB subcomponent

Figure 8 shows the order of the camera sensor data coming from the line FIFO and directly from the camera. For each pixel we identify two states: the state BLUE where colors blue and green1 are received and the state RED where colors red and green2 are receive.

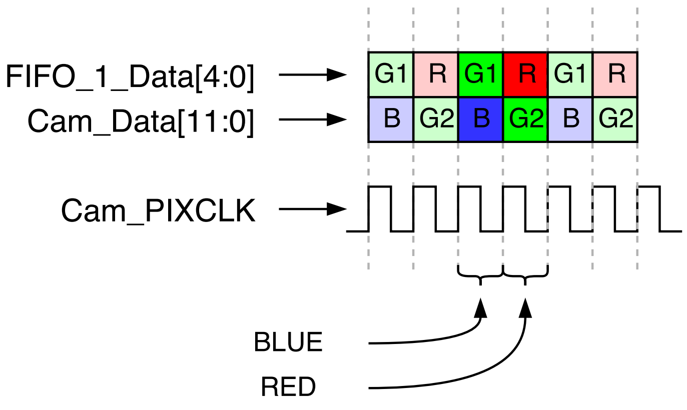


Figure 8: Bayer input signal order with indicated internal signals RED and BLUE

[1]Having a higher resolution for green makes sense, since the human eye is more sensitive in the green spectrum

Figure 9 shows the process for transforming the bayer pattern to a RGB value. The states `BLUE` and `RED` are external signals provided by the camera interface FSM. During the first phase state `BLUE` is active while state `RED` is inactive. The colors green1 (G1) and blue (B) are buffered in 5bit registers. Next, the state `BLUE` becomes inactive and state `RED` becomes active. The buffered G1 and the incoming G2 are added to a 6bit green value. The result is combined with the buffered blue value and the incoming red value to a 16bit RGB signal which is provided as output of the "Bayer to RGB" subcomponent.

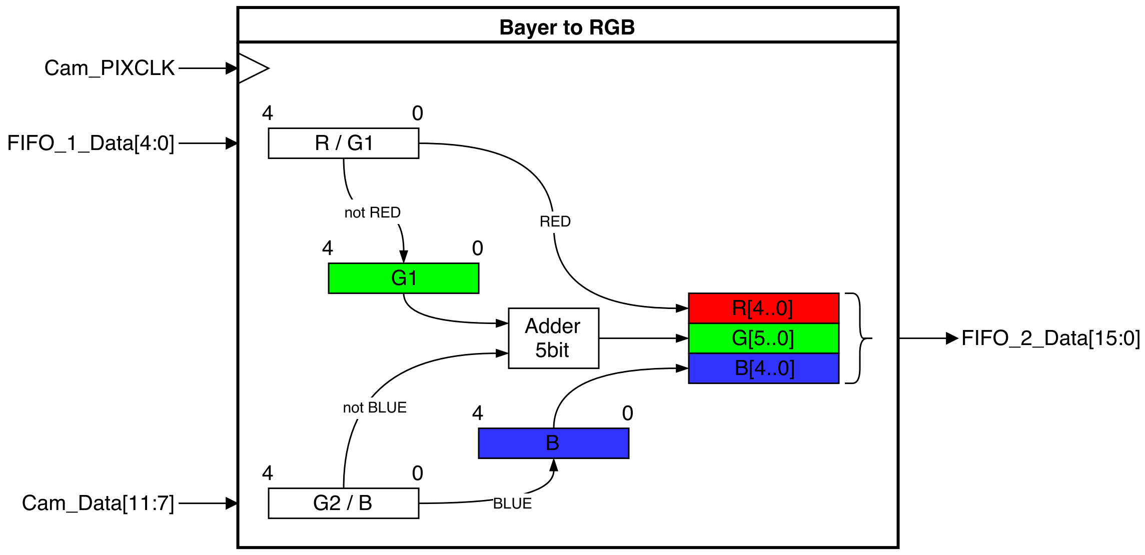


Figure 9: Bayer to RGB transformation block diagram.

4.3.2 Sensor line FIFO

The second subcomponent of the camera interface is a 2048x5-bit FIFO [2] buffer which stores an entire line of color sensor data for later processing.

4.3.3 Camera interface FSM

We have three different FSM's working together in a hierarchical way, which are illustrated in figures 10.

The lowest-level FSM has 3 states: `IDLE`, `BLUE`, `RED`. It defines which color is being read from the camera data lines and FIFO respectively. During state `BLUE` the camera outputs the blue sensor value while the FIFO outputs green (G1) value. During state `RED` the FIFO outputs the buffered red sensor value while the Camera outputs green (G2) value. This process is illustrated in figure 8.

The second FSM has 4 states: `IDLE`, `PIXEL_BUFFER`, `PIXEL_OUTPUT` and `PIXEL_SKIP`. In the state `PIXEL_OUTPUT` the RGB pixel data is written into FIFO2. The state `PIXEL_SKIP` results in the down sampling from 640 to 320 columns.

The last FSM at the highest hierarchical level has the following states: `IDLE`, `LINE_BUFFER`, `LINE_PROCESS`, `LINE_SKIP1` and `LINE_SKIP2`. In the `LINE_BUFFER` state a first line containing the red (R) and green (G1) sensor values is stored in FIFO1. Then, during the next `LINE_PROCESS` state, the data from FIFO1 is read simultaneously with the incoming is data from the camera which is then transformed to a RGB pixel value by the "Bayer to RGB" component. The following two states `LINE_SKIP1` and `LINE_SKIP2` result in the down sampling from 480 to 240 lines.

[2]Quartus only allows for a 2048xNbit size. Actually only a 1280x5bit FIFO is needed.

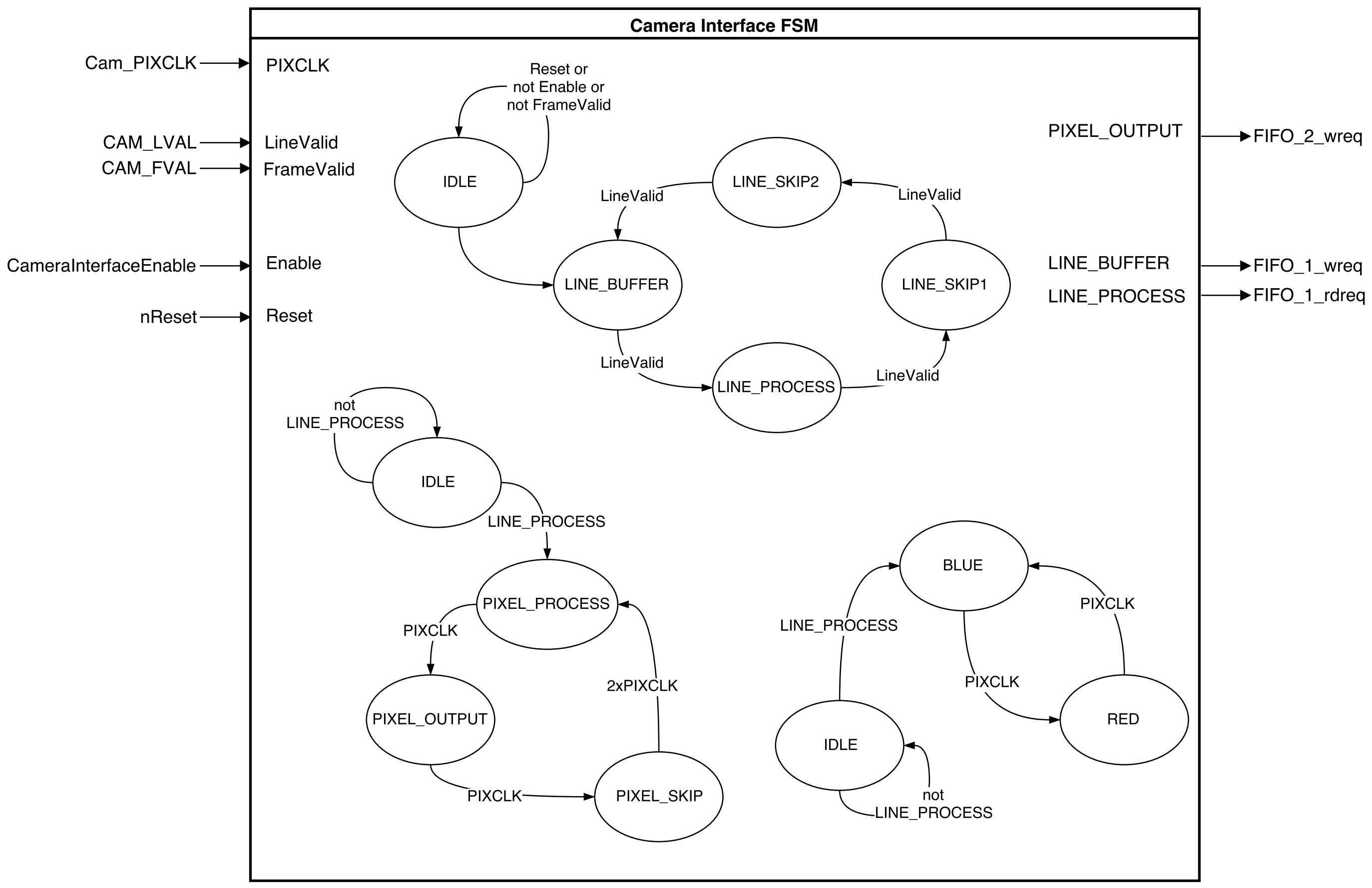


Figure 10: Finite state machine controlling the Camera interface

Figure 11 shows the desired waveform of the input and output signals. Only every second PIXCLK clock pulse a RGB pixel is calculated and only every second pixel is written in FIFO2 resulting in the horizontal down sampling from 640 to 320 pixel line width.

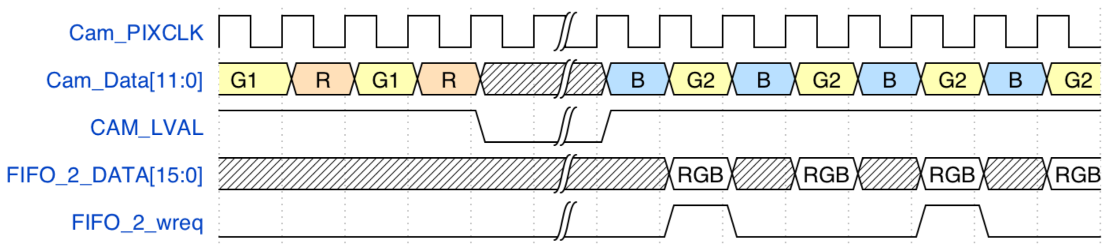


Figure 11: Camera interface input and output signals wave diagram

5 Master component

5.1 Description

The master component is fully synchronous to the main clock domain. It reads data out of the 16x16-bit FIFO containing the desired format of the data to be stored in memory. Its main functionality is to transfer data through bursts to the SDRAM via an Avalon interface. We have defined the burst-length as 8 and on each burst count, a 32-bit signal is transferred to the off-chip memory. 16 pixels are thus sent to the memory in one burst. The FIFO is designed such that its total size matches the length of one burst to avoid inferring a counter in the control part of this block.

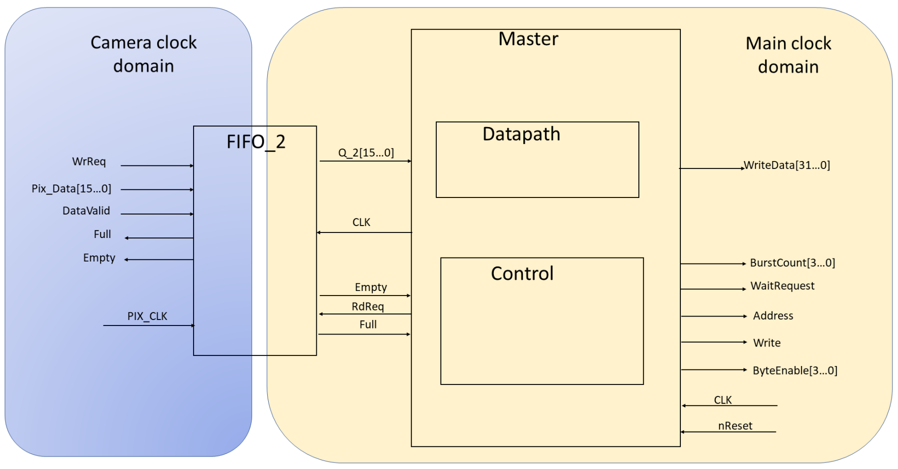


Figure 12: Architecture of Master component

The input data being 16-bit sized (FIFO) and the output 32-bit sized (SDRAM), the master component needs to concatenate 2 consecutive inputs to build the output signal. This is done by the proposed datapath in Figure 13.

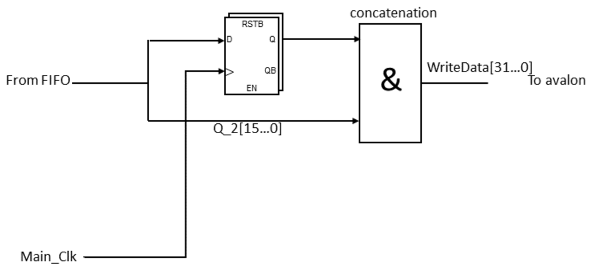


Figure 13: Proposed datapath architecture for the master component

A simple 2-state FSM (Figure 14 is controlling the data flow and the interfacing between the master component, the FIFO and the Avalon bus. The timing of the interface signals of the FIFO with the master component are shown in Figure 15 and Figure 16 shows the timings of the Avalon interface signals.

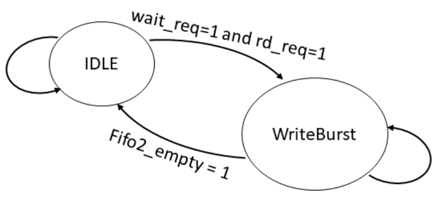


Figure 14: Proposed FSM for the master component control

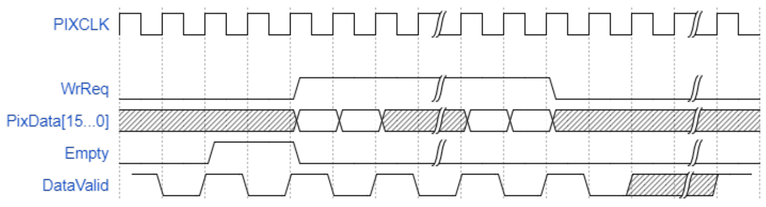


Figure 15: Timing diagram Master component front-end signals

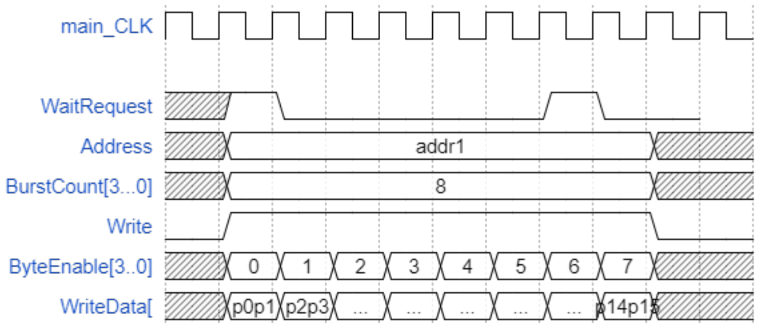
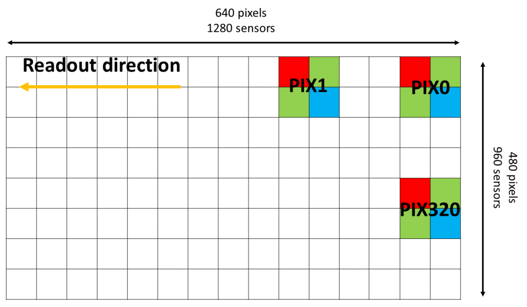


Figure 16: Timing diagram of Master Component/Avalon interface. WriteData is of size [31...0]

6 Data storage in memory

The data of the camera is stored in the DRAM in 32 bit format, so 2 pixels are stored per memory 32-bit address. The readout direction of the camera is from right to left and from top to bottom, and the storage will have the same order, and the LCD has to be configured such that it writes the data back in this order which should be possible via software control in LCD slave registers. It should be done this way otherwise the data of the full image has to be stored fully in our custom camera component because the LCD default writing order is totally different (lines and columns are swapped and the order is from left to right). We cannot afford to store the full image because of on-chip memory limitations.



Address tag	Memory 32-bit data	Organization
0x-----------------0	RED[5…0] | GREEN [6…0] |BLUE [5…0]	PIX 1 | PIX 0
0x-----------------1	…	PIX 3 | PIX 2
0x-----------------2	…	PIX 5 | PIX 4
0x-----------------3	…	PIX 7 | PIX 6
…	…	…
…	…	…

Figure 17: Camera readout direction and SDRAM memory mapping

7 Camera configuration

7.1 Clock

Through a PLL component the FPGA system provides a clock to the camera throuth the XCLKIN line. The camera supports up to 96MHz clock frequency. We plan to configure the PLL to 50MHz and use this clock undivided as PIXCLK. This can be achieved by setting `Divide_Pixel_Clock`=0 in `Pixel Clock Control` register (bits 6:0 in R0x00A).

If a different clock is needed, the camera provides a configurable PLL.

7.2 I2C commands

The camera is configured the usual way through I2C having read and write access to the 8bit address space with 2byte register size.

Register write A write access is done by addressing the camera through the I2C write address (0xBA) and then writing the 8bit register address followed by the two byte register data to be written. The register data is sent MSByte first.

Register read A read access happens in similar fashion by first addressing the camera through the I2C write address (0xBA) followed by sending the 8bit register address. Then the camera is addressed again using the I2C read address (0xBB) followed by reading the two byte register data.

Multiple read/write For both read and write access also multiple registers can be written at once using the address auto-increment. To do this, after the first register data transfer, the master can just continue reading or respectively writing data which will be read/written form the following registers.

7.3 Register configuration

We want to configure the camera to output an image of resolution 640x480 using binning sub sampling. According to "Table 1.7 Standard Resolutions" in the TerasIC THDB-D5M Hardware specification we have to configure following registers:

- `Row Size` = 1919 (R0x03)
- `Column Size` = 2559 (R0x04)
- `Shutter Width Lower` = 3 (R0x09)
- `Row_Bin` = 3 (R0x22 [5:4])
- `Row_Skip` = 3 (R0x22 [2:0])
- `Column_Bin` = 3 (R0x23 [5:4])
- `Column_Skip` = 3 (R0x23 [2:0])

After the register configuration we have to ensure that `Chip Enable`=1 in `Output Control` register (bit 2 in R0x07).

If for displaying reasons the image has to be mirrored, it is easiest to change the camera configuration to mirror row or column of the image. This can be achieved by setting `Mirror Row` or `Mirror Column` in `Read Mode 2` register (bits 15 or 14 in R0x20).

7.4 Operation modes

7.4.1 Continuous

For continuous mode we must clear the bit `Snapshot` in register `Read Mode 1` (bit 8 in R0x1E)

7.4.2 Snapshot

The camera allows also a snapshot mode where image capture can be triggered through the TRIGGER line or by setting `Trigger`=1 in `Reset` register (bit 2 in R0x0B). For this the camera needs to be put into snapshot mode by setting `Snapshot`=1 in register `Read Mode 1` (bit 8 in R0x1E)

7.4.3 Test pattern mode

For testing our custom digital logic, the camera provides a `Test_Pattern_Mode` which outputs a known color pattern, which facilitates debugging.