FPGA Based Reconfigurable Platform for Complex Image Processing

Manish Kumar Birla
Siemens Corporate Technology – India

***Abstract*—Field Programmable Gate Arrays (FPGAs) are in use to build high performance DSP systems. FPGAs are uniquely suited to repetitive DSP tasks, such as multiply and accumulate (MAC) operations in parallel. As a result FPGAs can vastly outperform DSP chips, which perform operations in an essentially sequential fashion. The interfaces between FPGA and image sensor have been very slow, which inhibits possibility of exploiting parallel processing in the FPGA This paper discusses method to build an image-processing platform using FPGA. This involves interfacing of FPGA to CMOS image sensor and VGA monitor. We discuss techniques that helped attending speed of 50 FPS (Frames per Second) for an interface of CMOS image sensor to FPGA from earlier reported speed of 3 FPS. A benchmarking application has been executed on FPGA and DSP based system and comparative real time performance data is reported.**

***Index Terms*—CMOS Image Sensor, Field programmable gate arrays (FPGA), Image Processing, Parallel Processing.**

I. Introduction

Image processing is the application of signal processing techniques to the domain of images — two-dimensional signals such as photographs or video.

The vision systems that are often used in security, quality control and automatic handling systems use image processing to allow a "computer system" to extract information from the incoming images.

The ability of vision based systems to perceive a minute variation in color, shape and relationship has opened up applications for image processing in high-speed manufacturing quality control, criminal forensics, medicine, defense, entertainment, surveillance systems and the graphic arts.

Fig.1. Common Steps in Image Processing System

Manuscript received February 10, 2006.

Manish Kumar Birla is with the Siemens Corporate Technology, Bangalore, KA-560100 India (phone: +91-80-2511-3649; fax: +91-80-2511-3666; e-mail: Manish.Birla@siemens.com).

Thus image processing involves analysis and manipulation of images with a computer. Fig. 1 shows steps involved in a general image processing system.

Image acquisition can be simple or may include complex pre-computation and is usually accomplished by an image sensor. The image data is send serially to the processing unit. The type of processing unit ranges from a standard PC to an embedded system with a single or multi-processor architecture.

FPGAs have proven ability for parallel processing and therefore most sought real time performance can be obtained with them [1]. But the problem is in the grabbing the data at the same speed as given by the image sensor. Then processing can be done using parallel processes in FPGA. Therefore interfacing of CMOS image sensor and FPGA becomes focus of the problem.

The report is organized as following:

Section 1: Prior Art: This part gives reader an idea of the difference between existing systems and proposed system. This is important to establish our work as a unique initiative.

Section 2: Image Processing System- Main Components: This section deals with details of the components of image processing system.

Section 3: FPGA Based Image Processing Platform: We discuss the design of interfaces for CMOS image sensor & VGA monitor with FPGA. Detailed design along with complex issues and solutions with timing information is given

Section 4: Results & Comparison

II. Prior Art

FPGA(s) can be found in any digital camera, Firewire camera, smart camera etc. In these products, FPGAs are used as co-processor. There are some FPGA platforms also provided by Hunt Engineering [2], which provides a ready interface for RS422 digital camera with FPGAs. There have been efforts in doing an interfacing of digital (CMOS) camera but the results were only with 3 fps (frames per second) [3], [4]. While the system discussed in this paper gives a speed of 50 fps.

III. Image Processing System- Main Components

Any image processing system needs image acquisition, processing and output unit. Image acquisition is done by image sensor. Processing can be handled by DSP/CPU/FPGA/ASIC. Output unit depends on the way the output is to be used.

A. *Image Sensor (Camera)*:

Camera or an image sensor is the input device for any image processing system. Regardless of the technology of image acquisition (CCD or CMOS), electronic image sensors must capture incoming light, convert it to electric signal, measure that signal, and output it to supporting format. We have used a CMOS image sensor for the proposed system. CMOS image sensor offers digital output, needs single bias with low voltage and single clocking, this makes interfacing easier and compact. The proposed system uses OV7610 Single-chip CMOS VGA Color Digital Camera from Omnivision Technologies, refer [5] for detailed datasheet.

B. *Processing Unit:*

Processing unit is the brain of whole system. It has system controller that controls the interfacing with external components and memory, an image-processing module that does the required processing.

There are various options to make processing unit. Mainly DSP/CPU and FPGA are the chief contenders.

We have chosen Virtex-4 SX35 device for the proposed system as this FPGA device has special architecture for signal processing tasks. To know more about resources of this device refer [6].

C. *Output Unit*:

The output of an image processing application can be displayed or can be stored and transmitted to decision-making process. For the proposed system we have used a VGA monitor.

IV. FPGA Based Image Processing Platform

In this section we describe design beginning with top level. All external hardware devices that are connected to the FPGA are shown in Fig. 2.

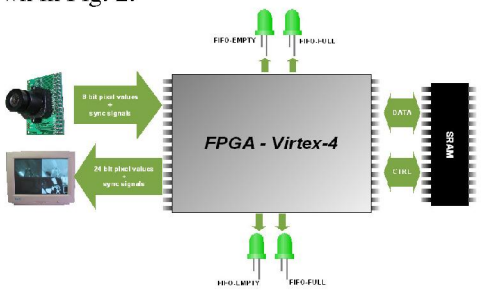


Fig. 2. System Architecture- Top Level

System Architecture:

The proposed setup uses Xilinx ML-402 development board with a Virtex-4SX35 chip. More information about this board can be found under [7] and [8].

Design of modules & parameterized interfaces:

Fig. 3 shows the dataflow from image sensor to the VGA port, involving the external memory.

When powered on, the CMOS image sensor continuously outputs pixel data along with synchronization signals and sends it to the FPGA. We performed some experiments with this camera. The experiments show that there are wide differences between datasheet and real timing information. For example number of pixels per row in QVGA image output are anywhere between 332-352. Based on the synchronization signals, the camera interface module (CAM IF) captures 8-bit monochrome pixel data and writes it an appropriate memory location in the BRAM. Multiple BRAMs are used so that parallel processing can be done. The addressing logic for each BRAM is made such that same module can be reused with different parameters. The images send by the sensor come at 50 fps. Once an image is written into the BRAM memory, the Processing modules starts continuously processing the image data by reading pixels back into registers, process them and writes the result into the external SRAM using intermediate buffering.

This SRAM (part number: CYPRESS CY7C1354B) is accessed for reading by VGA interface module and image is displayed on the monitor. The data width of SRAM is 32 bits; so 4 pixels data are packed in one address.

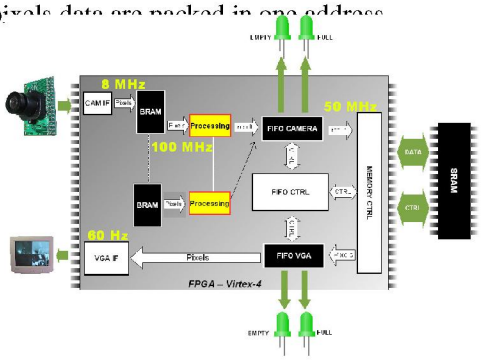
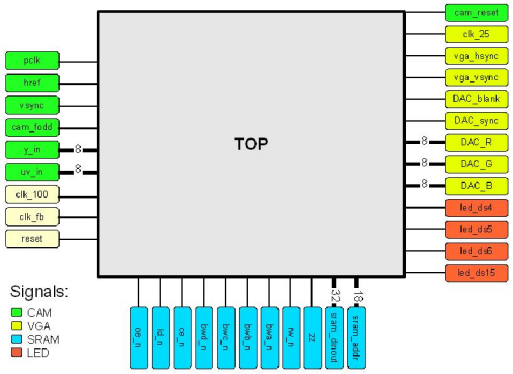


Fig. 3. Top-level architecture showing different modules

A. *Top Module*



The top module represents the interface to the external world. Here the signals from CMOS image sensor, external oscillator, external memory and VGA port are connected to further modules inside. Fig. 4 shows the external signals of top module. This figure shows one BRAM and one processing block as connection for parallel block will be identical.

Fig. 5 takes a peek inside the top module.

Note that only the important modules/signals are shown in a simplified way.

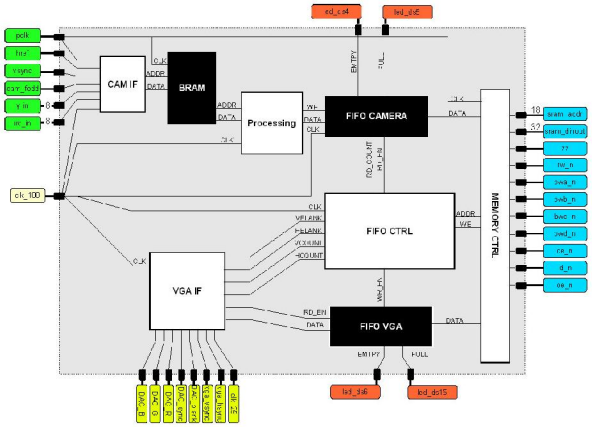


Fig. 5. Top module with internal signals in design

B. *Time critical Asynchronous FIFO (First-In-First-Out)*

FIFOs are made with distributed or block RAM [8]. There are 2 ports, which can operate at different clocks making it an asynchronous FIFO. This concept is used in our design to match different speeds of different processes. In our case, the FIFOs are used as input and output buffers between the memory controller and the VGA and the Processing module respectively.

The FIFO depth is 128, thus FIFO has to be flushed well

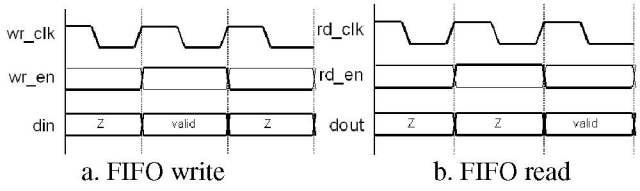


Fig. 6. FIFO Write & Read mode

timed and the flushed data is written into external memory. Thus timing for enabling read and write of FIFO are crucial. For example for VGA interface we store one row (80*4=320) data in the FIFO and then generate flag to start reading by VGA interface.

In write mode input data din will be registered on the positive edge of write clock only when write enable is set to HIGH. In read mode the data, which is read from the FIFO, has one clock cycle delay to read enable. To read a value you need to apply HIGH to read enable and after one clock cycle the data will be valid. Fig. 6 illustrates the waveform for both modes.

C. *Dual port BRAM*

The design uses internal memory of Virtex-4 FPGA. This memory is configured using COREGEN tool for the required specifications [10]. Since the memories are available in 18Kbits blocks, we consume 66 of 192 blocks of BRAM. Use of dual port memory (Fig. 7) enables us to read and write image continuously in real time. Read & write modes are

Fig. 7. BRAM module with external signals

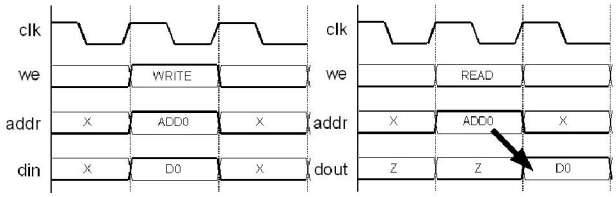


Fig 8. BRAM Write & Read Modes

shown in Fig. 8 with waveforms.

D. *High Speed Digital Camera Interface module*

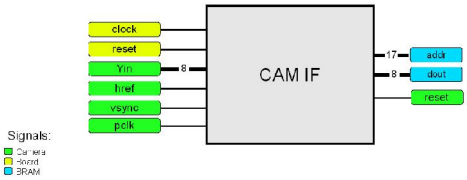


Fig. 9. Camera Interface module with external signals

All signals (Fig. 9) coming from the CMOS image sensor are input to this module. The module captures images, generates addresses and writes data into the BRAM. It also gives a reset for the CMOS sensor. Fig. 10 shows the periodic waveform produced by the sensor for each frame after power-up.

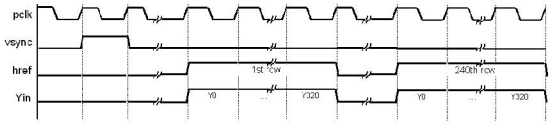


Fig. 10. Camera Signals

All signals are synchronous to the pixel clock (PCLK). In the beginning of each frame VSYNC goes HIGH for a few clocks. At that time the synchronizing signals are LOW and the output Yin is invalid. A few clock cycles after the VSYNC pulse, the sensor outputs pixel data column-by-column and row-by-row in a linear mode. In each forthcoming HREF pulse one row of data (from the left most to the right most pixel) is generated. On the oscilloscope we saw that the number of pixels send during this time varies between 332-352 pixels per row. In the current design we just capture 320 pixels and discard the rest. Between two VSYNC pulses there are 240 HREF (referring to 240 rows), thus image size is 320x240

(QVGA).

After studying behavior of synchronizing signals by experiments, we implemented two counters: XCOUNTER and YCOUNTER. During VSYNC = HIGH, the YCOUNTER will be set to zero. Similarly HREF = LOW, the XCOUNTER will be set to zero. Inside a HREF pulse the XCOUNTER will be increased by one on each positive edge of PCLK. When the counter exceeds 320, the YCOUNTER will be increase by one and the XCOUNTER stops increasing. Under each HREF pulse, we register Yin on each positive edge of PCLK to capture the pixel data (luminance data). At the same time we also generate an address for the BRAM, which is based on XCOUNTER and YCOUNTER. The equation is as follows:

$$ADDR = YCOUNTER \cdot 320 + XCOUNTER \quad (1)$$

This makes design fast as well as well controlled. Since address generation also happens in parallel, time is saved.

E. VGA Monitor Interface module

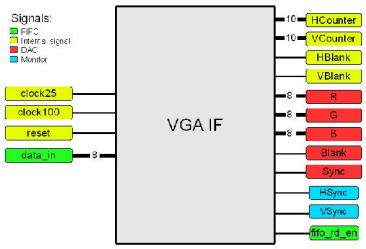


Fig. 11. VGA interface module and external signals

Fig. 11 shows VGA interface and signals. The task of this module is to generate accurate horizontal and vertical sync signals for the monitor and also to feed the DAC (Digital to Analog Converter) with the color information

(RGB, each 8-bit) required to generate appropriate analog R, G, and B signals. The timings for the standard 640x480 VGA sync signal are given in Table 1 and Fig. 12. All time specifications are for a 60Hz refresh rate and a 25MHz clock frequency.

For HSYNC and VSYNC, state machines based on simple counters are implemented. HCOUNTER and VCOUNTER are used for HSYNC and VSYNC respectively. Both state machines have the same structure, so state machine for HSYNC is explained here.

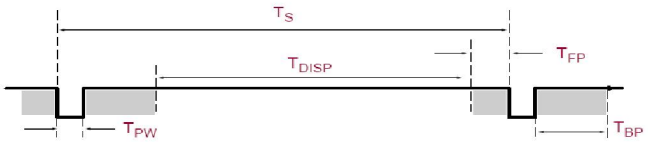
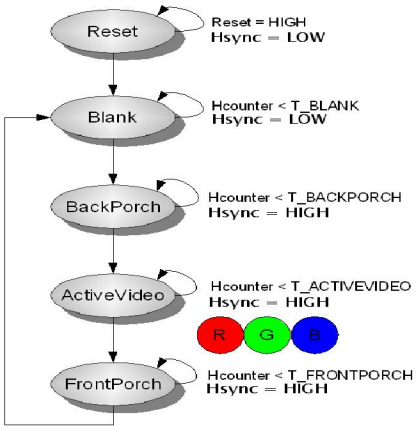


Fig. 13 shows the state machine for horizontal synchronization. In the RESET state all registers will be set to the default value. As long as Reset is HIGH the state does not change. Once Reset is LOW, the machine sets the HSync signal to LOW and remains in this state until the HCOUNTER exceeds the value defined by T BLANK. The following state sets HSYNC to HIGH and remains here for 1.92us. This is the time required to bring the ray of the monitor to the position where it can start displaying pixels. In the ACTIVEVIDEO state, 640 pixels will be displayed on the monitor screen. Here VGA IF module reads data for each pixel from the FIFO-VGA and applies the read value (luminous data) to all three color signals R, G, and B to produce a grayscale image. After the HCOUNTER reaches 640 the state is changed to FRONTPORCH, here it waits for 640ns. During this time the ray beam jumps back to the left side of the screen. At that time no pixel data should be applied to the *RGB* outputs. After this state the machine jumps back to the BLANK state.



F. Optimized FIFO Controller

This module as shown in Fig. 14, is a controller for VGA-FIFO and CAM-FIFO. There are two FSMs one each for VGA-FIFO and CAM-FIFO. For each FIFO there is one module that either reads data from FIFO and writes this data into external memory or the other way around. Both modules to control the MEMORY_CTRL generate the signals ADDR and RW_EN. Multiplexers (MUX) are used for these signals as shown in Fig. 15. MUX based logic makes circuit combinational and a single assign statement is used to generate optimized logic for MUX.

Since the VGA module needs to pixel data when the monitor beam is at the beginning of a new row, the process of

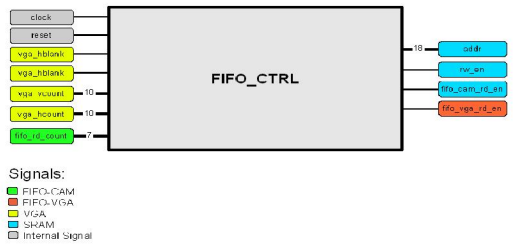


Fig. 14. FIFO Controller and external signals

filling the FIFO-VGA with one row of data is mandatory. Therefore this module has priority to FIFO-CAM FSM. The process of sharing the memory is synchronized with the VGA module. Initially the HREF signal is from LOW to

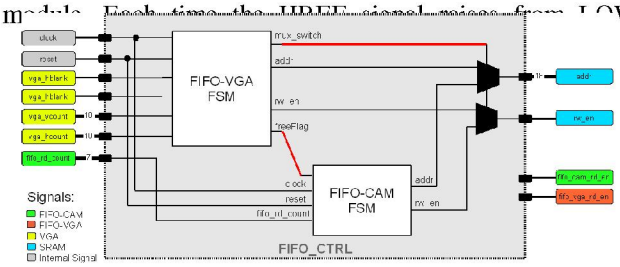


Fig. 15. Internal signals/modules of FIFO Controller

HIGH, that is in the BackPorch part, the MUX are switched to connect address and Read/Write to the output of the FIFO-VGA module and 320 pixel values will be read from external memory into the FIFO-VGA. At the same time the signal FREEFLAG indicates that now the FIFO-CAM-FSM has to wait until the other FIFO is filled. Fig. 16 shows generation of FREEFLAG depending on the HREF

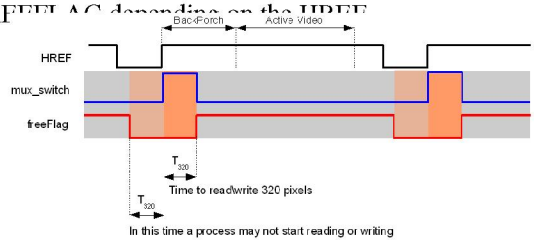


Fig. 16. FREEFLAG generation with timing details

During the MUX switch pulse the FIFO-VGA-FSM module controls the memory. The width T320 refers to time taken to read 320 pixels from the external memory. The FREEFLAG signal indicates whether the remaining time is sufficient to write 320 pixels into the external memory. The FIFO-CAM-FSM may not initiate a write process during this time; therefore the FREEFLAG remains LOW for 2-T320. The FIFO-CAM-FSM state machine sees this signal.

Getting correct timing is possible only by using precise clocks and counting the different pulse signals correctly.

CAM FIFO FSM: Fig. 17a shows the FSM.

RESET state: FSM enters in this state at system reset.

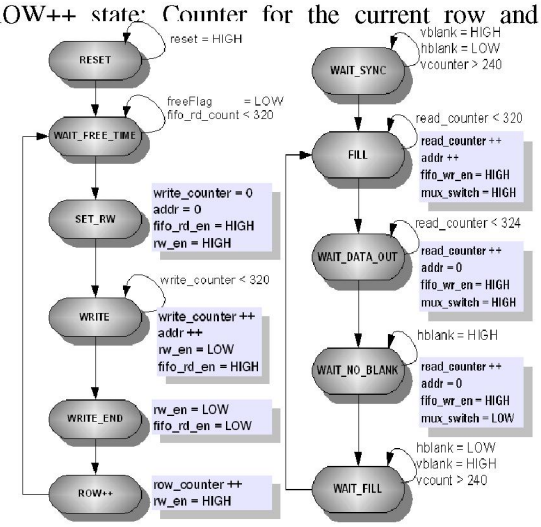
WAIT FREE TIME state: FSM waits here for two conditions. First condition: read count >= 80. Second condition: FREEFLAG = HIGH.

SET RW state: Address and counter registers are set to zero.

WRITE state: Write 320 pixels into the external memory by enabling read of FIFO-CAM and setting the memory controller to write operation. For each written pixel, the write counter increases by one. When this counter exceeds 320 go to the next state.

WRITE END state: Here reading from FIFO is disabled but memory is kept in write mode to compensate for one clock delay in reading from FIFO. Then we go to ROW++ state.

ROW++ state: Counter for the current row and set the



a. CAM-FIFO-FSM b. VGA-FIFO-FSM

Fig. 17. FSM in FIFO Controller module

memory controller in read mode (disabling write mode). Again go to the WAIT FREE TIME state to wait until 320 new pixels are accumulated in the FIFO.

VGA FIFO FSM: Fig. 17b shows the FSM.

As described in the previous section, this module initiates the filling of the FIFO-VGA by setting the memory controller in read mode (default mode) and generates addresses for the current row.

The data output of the memory controller is directly connected to the data-in of the FIFO. In that way this data will be written into the FIFO when write-enabled.

WAIT SYNC: In the beginning the state machine synchronizes with the VGA timing. Wait until the VGA state machine starts at the beginning of a new frame. The input VCOUNT shows the current row, so this counter has to be zero.

FILL state: 320 addresses are generated based on current row to read corresponding memory locations. For each data read, the read counter increases by one. Until this counter reaches 320, system remains in this state. When applying one address to the memory, the read data for this address will appear only four clock cycles later.

WAIT DATA OUT state: waiting till read count is 324 compensates the delay of four clocks.

WAIT NO BLANK: After one row of data has been read from the FIFO, the state machine waits again until the VGA

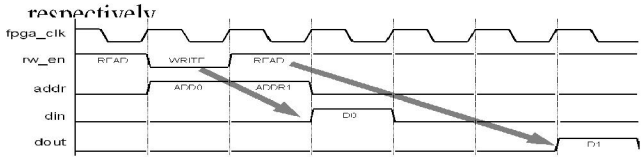
module has drawn one row of data and is at the end of a row (HBLANK=LOW).

WAIT FILL state: FSM is now in the blanking period, where no data is needed. In this system waits until HBLANK=HIGH showing start of horizontal sync pulse. At this time go to the FILL state.

G. SRAM Controller

The ML-401 development board provides one CYPRESS CY7C1354B SRAM memory [11]. It is used in the current design to store the resulting image. To interface with this memory SRAM controller, provide by Xilinx is used [12].

For read or write operations (Fig. 18) only a few signals are important to the user. These are ui addr, ui rw n, ui write data, and ui read data, referred as ADDR, RW, DIN, and DOUT respectively.



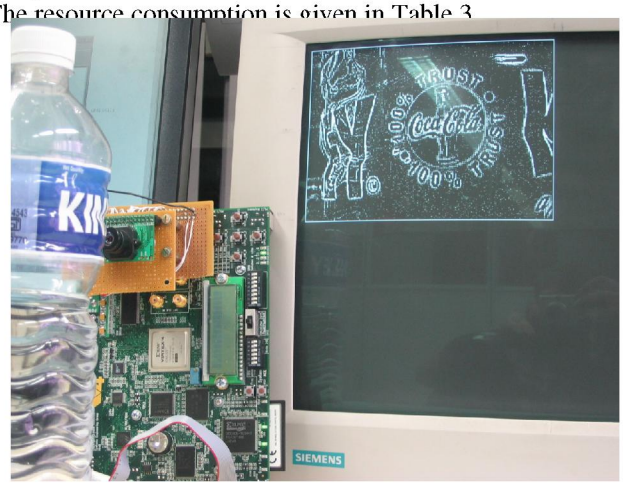
H. Plan for Processing module

In this module in the current system the image from camera is directly being put on the VGA interface via SRAM. This module can have the intended processing for different applications like MAC operation, masking operation etc. Next section shows results of using edge detection operation in processing module. Everything is done in parallel, and therefore we achieve tough timing targets.

V. RESULTS AND COMPARISON

The proposed system has been used to perform edge detection and output is shown in Fig. 19. From Table 2 it can be seen that FPGAs provide gain of the order of 100 times as compared to state-of-art DSPs.

The resource consumption is given in Table 3.



REFERENCES

[1] Hamamoto, T. et al., "Real-time image processing by using image compression sensor" IEEE Conference on Image Processing, Vol. 3-ICIP: 935-939,1999.

[2] Hunt Engg. (July, 2005). Reconfigurable High Performance DSP systems. [Online]. Available: http://www.hunteng.co.uk/info/reconfigurable-dsp.htm

[3] Daniel Crispell, "Implementation of a Streaming Camera using an FPGA and CMOS Image Sensor", Brown University. [Online] Available: http://www.lems.brown.edu/~dec/docs/fpga_camera.pdf

[4] Ryan Henderson, (May, 2002) "CMOS Digital Camera Controller and Frame Capture Device", Mentor Graphics Design Contest 2002

[5] OV7620 Product Specifications -Rev. 1.3, Omnivision Technologies, Inc., 2000

[6] Virtex-4 User Guide [Online]. Available: www.xilinx.com/virtex4

[7] ML-40x Evaluation Platform user guide

[8] C. Holst, M. Birla, "interfacing OV7620 camera with FPGA pins" Siemens Internal Report

[9] Asynchronous FIFO v6.1 product specification (DS232), Xilinx Logicore, 2004.

[10] Dual Port Block Ram v6.0 product specification (DS235 v0.1), Xilinx Logicore, 2003.

[11] CY7C1354B1 SRAM datasheet

[12] SRAM controller [Online]. Available: www.xilinx.com/bvdocs/appnotes/xapp136.pdf.

Manish Kumar Birla The author's birth place is Indore, India. Manish earned Master of Technology degree in Microelectronics from Indian Institute of Technology, Bombay, Mumbai, India in 2002. The author earned Bachelor of Engineering degree in Electronics & Instrumentation from DAVV, Indore, India in 2000. He had been with GE-Global Research Center for Feb'02-Sep'04. Currently he is working with Siemens Corporate Technology – India, Bangalore as Associate Member Technical Staff in Embedded System for Computer Vision group.

Mr. Birla has research interests in reconfigurable (FPGA) hardware usage in unique applications, ASIC design.