Volume 3, Issue 1, January 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Use of Reconfigurable FPGA for Image Processing

Aniket Burkule*
**Department of Electrical Engineering,*
VJTI, Mumbai
India

P. B. Borole**
*** Department of Electrical Engineering,*
VJTI, Mumbai
India

Abstract -- Various image processing algorithms are used in today's mobile embedded system. Most of the image processing algorithms are time consuming due to large amount of calculations involved in them. Hence it is desirable to use high speed system. The most effective solution is to use FPGA. Single FPGA can perform single image processing algorithm at a time. Hence to make it multitasking the option of reconfigurability can be used. This paper deals with three image processing algorithms performed on different images. The performance of the system is evaluated using worst case and best case combination.

Keywords – Reconfigurable FPGA, Image processing, FPGA, Edge detection, multitasking using a single FPGA.

I. INTRODUCTION

Calculations involved in image processing algorithms are real challenge in front of system designers today. Traditionally three methods are followed for designing a system. 1.ASIC 2.Using microcontroller 3.FPGAs. ASIC systems are most efficient one, but once it is made for any particular purpose, its functionality cannot be changed. Microcontroller based system gives flexibility in the functionality change but its speed of operation is much lower than ASIC systems. The third option is FPGA. FPGA systems can work with speed comparable to ASIC and with much more flexibility.

Once a FPGA system is configured it acts as ASIC system. This limits to perform various types of applications with the speed of FPGA. This problem can be solved by configuring FPGA again and again. In this way we increase flexibility of FPGA. For applications in image processing, we considered three algorithms viz. image blurring, image edge detection and image compression. This paper does not give any new method for image compression, blurring or edge detection. It makes use of existing algorithms and tries to make system more flexible. Short introduction about these algorithms are given below.

A. Image Blurring:

Blurring an image usually makes the image unfocused. In signal processing, blurring is generally obtained by convolving the image with a low pass filter. In this Demonstration, the amount of blurring is increased by increasing the pixel radius. Blurred image is not desirable in many applications but this can be advantageous too. The areas in which secrecy is to be maintained the image can be blurred. This is the simplest way to hide the information in an image. Other way to blur the image is to multiply image pixel with some constant. As the pixel strength changes linearly with the constant, image won't look like original and many details can be hidden. While blurring the image some particular part of the image can be blurred so as to highlight important part of the image. Image can be un-blurred by dividing pixel strength by the same constant.

B. Image edge detection:

The objective of image edge detection is to find sudden changes in brightness, surface orientation, surface boundaries, and changes in material properties. Performing edge detection on image give a set of connected curves that show boundaries of object. By using edge detection discontinuities in surface can be found. There are many methods for edge detection, but most of them can be grouped into two categories, search-based and zero-crossing based. The search-based methods detect edges by first computing edge strength, usually a first-order derivative expression such as the gradient magnitude, and then searching for local directional maxima of the gradient magnitude using a computed estimate of the local orientation of the edge, usually the gradient direction. The zero-crossing based methods search for zero crossings in a second-order derivative expression computed from the image in order to find edges, usually the zero-crossings of the Laplacian or the zero-crossings of a non-linear differential expression. As a pre-processing step to edge detection, a smoothing stage, typically Gaussian smoothing, is almost always applied.

C. Image Compression:

The key to the compression process is a mathematical transformation known as the discrete cosine transforms (DCT)**.** The DCT belongs to a class of mathematical operations that includes the well-known fast Fourier transform **(**FFT**)** as well as many others. The DCT is closely related to the Fast Fourier Transform. The amplitude of the "signal" in this case is simply the value of a pixel at a particular point like the FFT. The DCT corresponds to representing a finite-length sequence with the same number of coefficients called DCT coefficients sequence can be exactly recovered from these coefficients. And like the FFT, there is an inverse discrete cosine transforms (IDCT) function that can convert the spectral representation of the signal back to a spatial one.

It takes a set of points from the spatial domain and transforms them into a representation in the frequency domain. In image compression, the DCT is most commonly applied to a sequence block of pixels taken from an image. This is a 2-D operation and results in a square block of DCT coefficients of the same size as the input. A considerably more efficient form of the DCT can be calculated using matrix operations. When multiplying two square matrices together, the arithmetic cost of each element of the output matrix will be N multiplication and N addition operations.

II. Related work

Reconfigurable custom computing machines, implemented as arrays of FPGAs, have been successfully used to accelerate applications executing on PCs or workstations [1, 2]. Image processing algorithms are particularly suitable for implementation on such machines, due to the parallelisms that may be exploited [3, 4]. Due to use of microcontroller or microprocessor instruction level parallelism is achieved. Research has been conducted to improve speed by designing system block by block. Hwang has suggested constructing *Universal Multiplication Networks* using small (4 bit) *Programmable Additive Multiply* (PAM) modules [5].

Simon D. Haynes suggested The multiplier which is constructed using an array of 4 bit Flexible Array Blocks (FABs), that can be embedded within a conventional FPGA structure [6]. Xuejun Liang suggested about use of multiple FPGAs to work simultaneously to achieve task parallelism [7]. The proposed system works with microcontroller. Microcontroller controls the configuration information of FPGA. This system can be implemented on any other FPGA architecture like Spartan-3, Virtex-II, Virtex -4, Virtex-6.

III. Reconfigurable Techniques

A. *Complete Reconfiguration*

In this type of reconfiguration, FPGA can be configured with a single configuration code at a time. In order to change functionality of system, configuration code needs to be changed. Configuration can be done by serial or parallel communication. Computers, microcontrollers can be used as master devices. In computer based configuration, communication can be carried out using RS232 or USB port for serial communication. In this paper, we configured FPGA with microcontroller. The use of microcontroller in design gives more flexibility and some task can be performed by microcontroller itself. Following fig shows interfacing FPGA with microcontroller

Another way to configure is using EPROM. This type of configuration is done particularly for preventing loss of configuration data. After power off, FPGA loses its data. To configure FPGA again the configuration code stored in EPROM can be used.

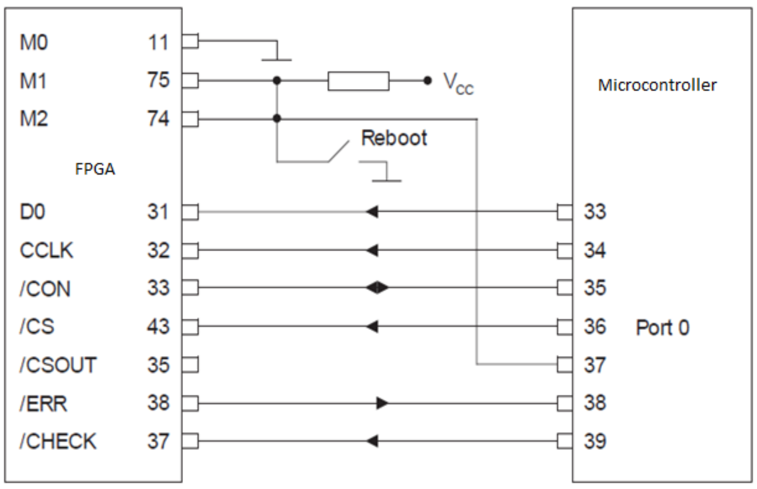

Figure 1. Interfacing between microcontroller and FPGA.

B. *Partial Reconfiguration:*

Run time partial reconfiguration is useful to designers who want to develop applications demanding adaptive and flexible hardware. Using partial reconfiguration can result in power and area savings. But an intelligent system is needed to manage reconfiguration in order to save power and meet timing constrains in real time systems.

IV. EXPERIMENTAL SETUP

As mentioned earlier we used microcontroller for reconfiguration. Three types of algorithms are performed on ten different images. Images used are of size 256x256. For this experiment we decided to have two different sequences. One of them is for best case, which means this type of sequence will give best result in terms of time. Other one is the worst case. Worst case sequence will take maximum time to perform operation on ten images. These two sequences are as shown in table1.

The time required for configuring FPGA depends on bitstream. The amount of information that makes up the configuration information for the FPGA is called a bitstream [8]. It is a file stored somewhere in the memory section. Figure 2 shows how this bitstream is structured. The bitstream begins with a token, the preamble, which indicates the beginning of the header section that contains global information concerning the whole configuration cycle. This is followed by the configuration information for the core cells and by the I/O configuration.

TABLE I.
Sequence of algorithms to be processed.

Best Case	**Worst case**
Image Blurring → Image 1	Image Blurring → Image 1
Image Blurring → Image 2	Edge Detection → Image 2
Image Blurring → Image 3	Image Compression → Image 3
Edge Detection → Image 4	Image Blurring → Image 4
Edge Detection → Image 5	Edge Detection → Image 5
Edge Detection → Image 6	Image Compression → Image 6
Image Compression → Image 7	Image Blurring → Image 7
Image Compression → Image 8	Edge Detection → Image 8
Image Compression → Image 9	Image Compression → Image 9
Image Compression → Image 10	Image Blurring → Image 10

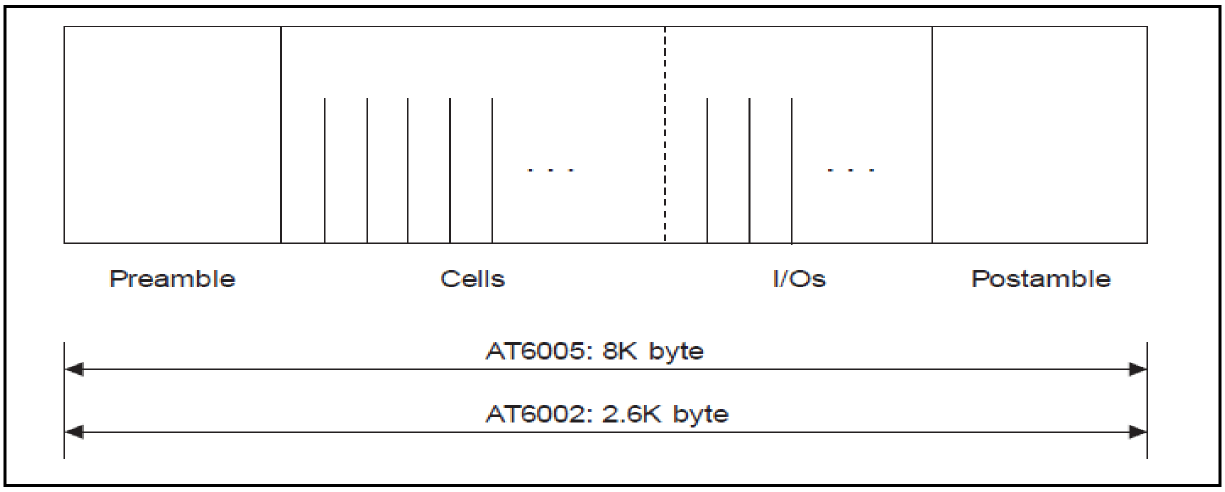

Figure 2 Bitstream Structure

The configuration cycle consists of three stages: reset, configuration and initialization. Configuration time is dominated by the time it takes to transfer data from the microcontroller to the FPGA device. The speed of transfer of data depends upon frequency of clock. The DCLK minimum frequency when using the 40-MHz oscillator is 20 MHz (50 ns). For example, the maximum configuration time estimate is (2.5 MBits of uncompressed data) = RBF Size x (maximum DCLK period / 1 bit per DCLK cycle) = 2.5 MBits x (50 ns / 1 bit) = 125 ms.

V. RESULTS

Time required for executing code also depends on the algorithm used. Here in our application 22 ms required for edge detection, 32 ms for image blurring and 35 ms for image compression. Considering these timings, operations are performed on images with sequence mentioned above.

Fig.3 shows result for sequence 1. Fig.4 shows result of sequence 2. With first sequence time required to complete processing is 677 ms and with second sequence time required is 1340ms.

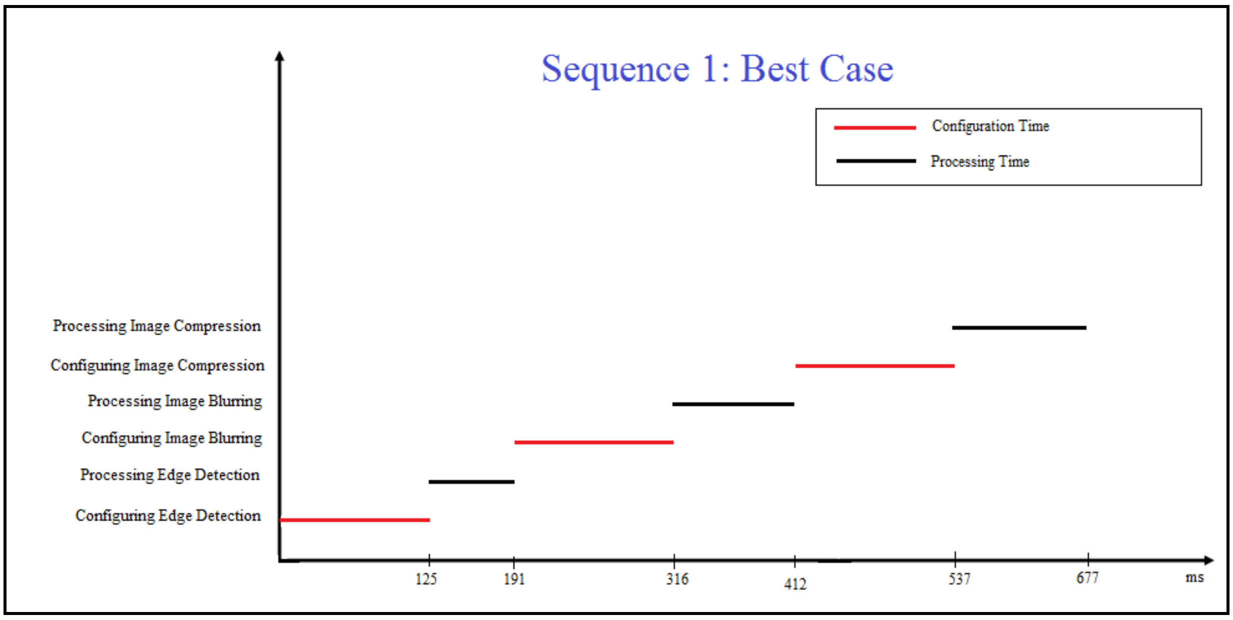

Figure 3 Result of sequence 1.

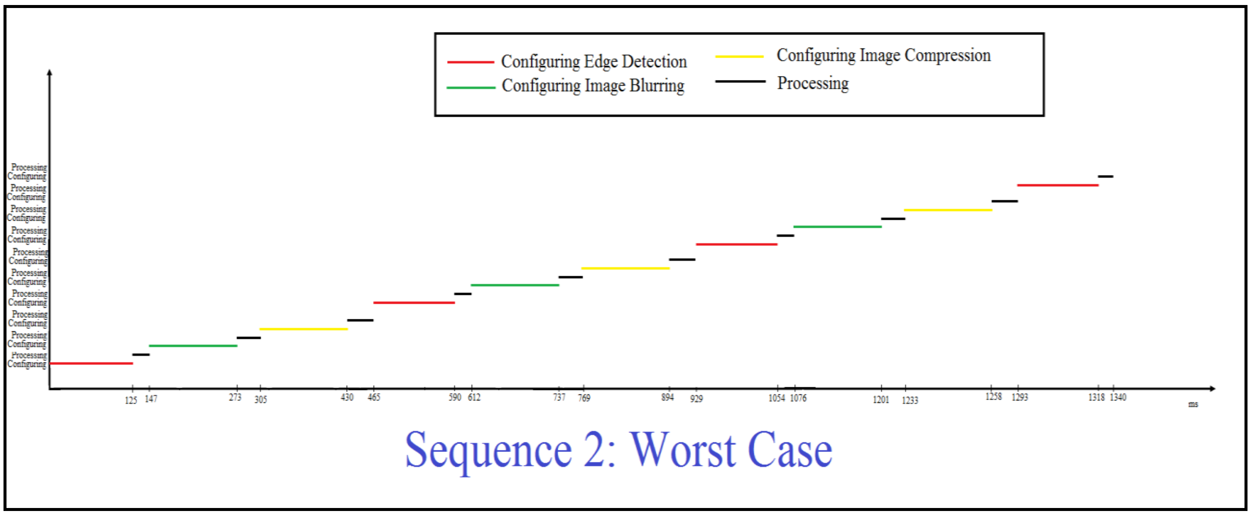

Figure 4 Result of Sequence 2.

VI. CONCLUSION

The FPGA based systems are mainly used for specialized purposes. Reconfiguring FPGA with different algorithms, it can be used for many other applications. This reduces time to process, time to market. This type of design is useful in application areas where processing should be real time but frequency of processing is less.

VII. FUTURE SCOPE

This paper presents the way of making a special purpose system to a generalized system. While making it to generalized system, power consumption is not considered. Further in this field FPGA can be self configured to save power consumed by microcontroller. Block by Block processing can be implemented so as to improve task parallelism.

REFERENCES

Journal Papers:

[1] Arnold, J.M., Buell, D.A., Hoang, D.T., Pryor, D.V., Shirazi, N., Thistle, M.R.: The Splash 2 processor and applications. In: IEEE International Conference on Computer Design: VLSI in Computers and Processors. (1993)

[2] Vuillemin, J.E., Bertin, P., Roncin, D., Shand, M., Touati, H.H., Boucard, P.: Programmable active memories: Reconfigurable systems come of age. IEEE Transactions on VLSI Systems **4** (1996) 56–69

[3] Haynes, S.D., Stone, J., Cheung, P.Y.K., Luk, W. "*Video image processing with the Sonic architecture.*" IEEE Computer **33** (2000) 50–57

[4] Athanas, P.M., Abbott, A.L.: "Real-time image processing on a custom computing platform". IEEE Computer **28** (1995) 16–24

[5] K. Hwang, "*Computer arithmetic Principles, Architecture and Design*", John Wiley & Sons,1st edn., pp. 194-197, 1979

[6] Simon D. Haynes, Peter Y. K. Cheung "*A Reconfigurable Multiplier Array For Video Image Processing Tasks, Suitable For Embedding In An FPGA Structure*" FPGAs for Custom Computing Machines, 1998. Proceedings. IEEE Symposium on.

[7] Xuejun Liang and Jack Shiann-Ning Jean, *"Mapping of Generalized Template Matching Onto Reconfigurable Computers"* IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 11, NO. 3, JUNE 2003

Website:

[8] www.atmel.com/Images/