1-1-2007

Evolvable Reconfigurable Hardware Framework for Edge Detection

Nader I. Rafla
Boise State University

Evolvable Reconfigurable Hardare Framework for Edge Detection

Nader I. Rafla
Electrical and Computer Engineering
Boise State University
Boise, Idaho. USA
nrafla@boisestate.edu

Abstract— Systems on Reconfigurable Chips contain rich resources of logic, memory, and processor cores on the same fabric. This platform is suitable for implementation of Evolvable Reconfigurable Hardware Architectures (ERHA). It is based on the idea of combining reconfigurable Field Programmable Gate Arrays (FPGA) along with genetic algorithms (GA) to perform the reconfiguration operation. This architecture is a suitable candidate for implementation of early-processing stage operators of image processing such as filtering and edge detection. However, there are still fundamental issues need to be solved regarding the on-chip reprogramming of the logic. This paper presents a framework for implementing an evolvable hardware architecture for edge detection on Xilinx Virtex–4 chip. Some preliminary results are discussed.

I. INTRODUCTION

Computer vision systems use image processing methods that are highly dependent on edge detection techniques [1]. The edge detection operators normally take significant amount of the computational time consumed by these systems. Several approaches had been developed in research in an effort to optimize the edge detection operation such as, the use of neural networks [2] or fuzzy-based approaches [3]. Other researchers focused on hardware implementation of edge detection algorithms [4, 5, 6]. Filed Programmable Gate Arrays (FPGAs) are the preferable hardware platform targeted for such implementations because of their rich resources of reconfigurable elements, general purpose processors, and the presence of memory on a single chip.

The framework represented here is based upon using the on-chip embedded processor to execute any Genetic Algorithm (GA) of choice and generate the best suitable configuration of the programmable elements of the same chip. This results in an Evolvable Reconfigurable Hardware Architectures (ERHA). This class of architectures dynamically changes its behavior to adapt to changes in the surrounding environment [7].

Such an environment is suitable for implementation of early-processing operators (e.g. edge detection) because of the complexity of the data set they operate on, the non-ideal image capture systems, and the execution time taken by the repetition nature of such algorithms.

II. EVOLVABLE RECONFIGURABLE HARDWARE

A. Filed Frogramable Gate Array Architecture

The architectural features of the Xilinx FPGA Virtex–4 device [8] are shown in Figure 1. The FPGA fabric contains the following various configurable elements:

- Configuration logic Blocks (CLBs) that provide combinatorial and sequential logic as well as distributed memory.
- Reconfigurable I/O blocks (IOBs) that provide the interface between the package pins and the CLBs.
- Block RAM modules that provide flexible dual-port RAM and optional FIFO logic.
- DSP slices with fast dedicated multipliers, adders, and accumulators.
- Digital Clock Manager (DCM) that facilitate clock calibration and distribution.

In addition to these modules, the FPGA provides two industry-standard embedded IBM PowerPC 405 RISC CPU cores [9]. Each of them possesses the following features:

- Standard 5-stage data-path pipeline.
- 32x32-bit general purpose registers.
- Two separate instruction and data caches.
- Memory management unit for virtual memory and RTOS implementation.

1-4244-1176-9/07/$25.00 ©2007 IEEE.

- Instruction and data on-chip memory (OCM) controllers interface directly to the embedded block RAM.
- Processor local bus (PLB) interface and support for the IBM CoreConnect™ bus architecture.

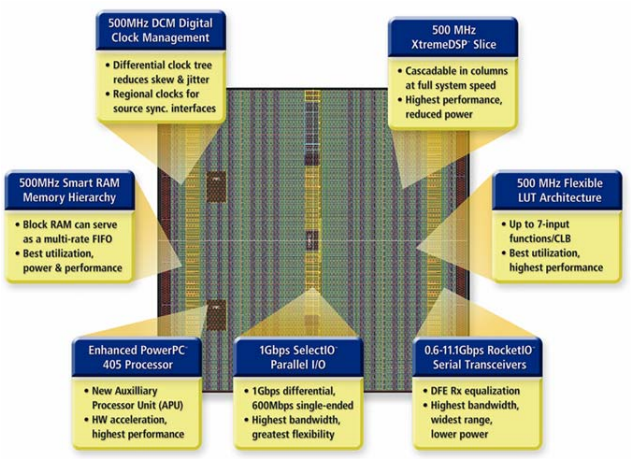

Figure 1. Key features of the Virtex–4 FPGA

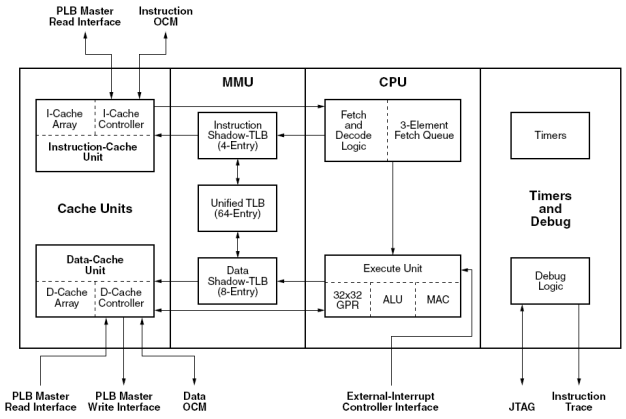

Figure 2. PowerPC 405 architecture

B. *Genetic Algorithms and Evolvable Hardware*

Genetic algorithms are one of the evolutionary computational techniques used for developing evolvable hardware [10]. This class of algorithms use techniques inspired by evolutionary biology such as mutation, selection, and crossover [11]. A set of candidate solutions, a population represented as a bit stream, must be present before executing the algorithm. Each candidate solution within the population is referred to as a chromosome. These candidate solutions undergo a process of evaluation, selection, and reproduction to select the best-fit chromosomes to specific constraints provided by the fitness function. These resultant chromosomes are used as a programming bit stream for the FPGA to generate the desired evolvable hardware architecture as depicted in figure 3.

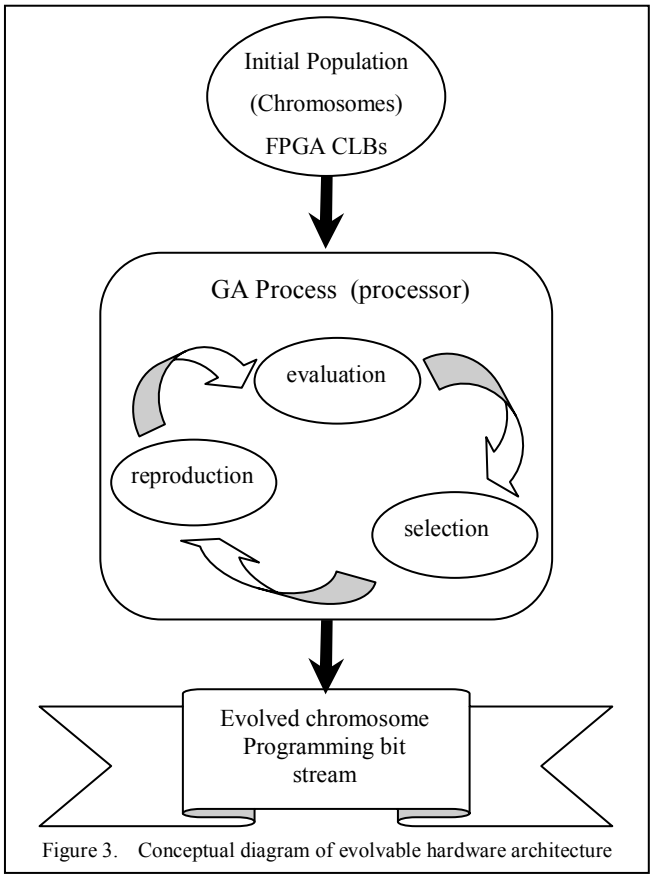

Figure 3. Conceptual diagram of evolvable hardware architecture

III. EDGE DETECTION ARCHITECTURE

A. *Edge Detection Operators*

Edge detection is an important step in all image processing systems. Several approaches had been developed in the literature for software implementation [1]. The Sobel operator [12] is a simple gradient-based operator that is commonly used and is immune to noise. When this algorithm is implemented in evolutionary hardware, the search process is done quickly and efficiently reducing the computation cost which is the major problem with its software-based counterpart.

The Sobel edge detection method separately process vertical and horizontal edges using two convolution gradients defined as:

$$g_1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad g_2 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

After these gradients are convoluted with the original image, the required edge information can then be calculated using the formula:

$$|g_1(m,n)| + |g_2(m,n)|$$

And the edge image is obtained after applying a threshold value.

B. *Image Operator Evolutionary Design*

The current evolution process shown in figure 4 is used to develop the best acceptable design configuration. In this process, an initial configuration represented by a set of chromosomes stored in an array of registers is selected. This configuration is evaluated using a fitness value calculated using the outputs of a pre-specified set of inputs; and the threshold. The process is repeated for all other configurations in the initial population and a new better population is generated. After a number of iterations, the best fit design configuration is obtained. The genetic algorithm described above uses mutation and crossover operations to generate the new population.

Algorithm: Evolution

Input: P: parameters of the initial edge detector including the threshold

N: number of configurations

Output: final configuration bitstream

- generate an initial configuration
- while (not N) and (P is not satisfied)
  - ❖ Evaluate each configuration *Ni*
  - ❖ Generate a configuration N_{i+1}
  - ❖ Sort configurations
- end while

Figure 4. The evolution algoithm

IV. EXPERIMENTAL RESULTS

An initial experiment is developed that include the implementation of the proposed algorithm on the on-chip processor to generate the edge detector design described above. The resulted configuration bit stream was extracted from the PowerPC and externally downloaded to the chip using the Xilinx ISE toolset since the current available configuration technology does not yet allow automatic reconfiguration of FPGAs. The proposed design performance in terms of function and speed is currently under investigation using a test benchmark image, Lena shown in figure 5, of size 128x128. Figure 6 shows the edge image.

Figure 5. Original image

Figure 6. Edge image

Preliminary results show a strong convergence to the sharpest edges in the image while almost eliminating the identification of weak edges. Modification to the mutation process is needed to maintain such weak edges. However, the identified edges are the most significant ones and as in the case of all edge detectors, not all edges are identified. To conserve time, while original image pixel values were initially being stored in memory, the preloaded algorithm starts generating and processing the chromosomes. Currently, an estimation of resource utilization from the synthesis report is being calculated

V. CONCLUSIONS AND FUTURE WORK

A framework for designing and implementation of an edge detection operator using ERHA had been presented. The framework is based upon using operators from genetic algorithms to develop the configuration bit stream for the FPGA. While the results had not been yet compared to other systems' outputs, preliminary experiment had been created and edges had been identified correctly.

Future work includes comparing the resulted edges with those of an optimized version of the proposed edge detection method simulated in a software tool, such as Matlab. Also different images need to be tested to further strengthen the conclusion about the proposed design. Although the current commercially available hardware chips do not allow a complete on-chip reconfiguration, partial configuration is feasible using the selected implementation platform.

REFERENCES

[1] Rafael C. Gonzalez, and Richard E. Woods, Digital Image Processing, 2nd ed., Prentice Hall, 2002.

[2] I. A. Hunter, and J. J. Soraghan, "Neural network edge detection-successes and failures," IEE Colloquium on Applications of Neural Networks to signal Processing, London, UK, December 1994.

[3] Madasu Hanmandlu, John See, and Shantaram Vasikarla, "Fuzzy Edge Detector Using Entropy Optimization," Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04), vol. 1, pp. 665- 670, April 2004.

[4] L. Basano, B. Caprile, E de Micheli, A. Geminiani, and P. Ottonello, "Edge-detection schemes highly suitable for hardware implementation," Journal of the Optical Society of America: Optics, Image Science, and Vision, vol. 5, Issue 7, July 1988, pp.1170-1175.

[5] Daggu Venkateshwar Rao, and Muthukumar Venkatesan "An efficient reconfigurable architecture and implementation of edge detection algorithm using Handle-C," International Conference on Information Technology: Coding and Computing vol. 2 p. 846-851.

[6] Biswajit Mishra , and Peter Wilson, "Color edge detection hardware based on geometric algebra" 3rd European Conference on Visual Media Production, 2006. (CVMP 2006), pp. 115-121.

[7] R. Tessier, "Reconfigurable computing in digital signal processing: A survey.," Journal of VLSI Signal Processing, vol. 28, 2001, pp. 7-27.

[8] Xilinx Inc. "Virtex–4 user guide," UG070, version 2.2, http://www.xilinx.com/bvdocs/userguides/ug070.pdf, April 2007.

[9] Xilinx Inc., "PowerPC Processor Block Reference Guide" UG018, version 2.1, http://www.xilinx.com/bvdocs/userguides/ug018.pdf, July 2005.

[10] T. Higuchi, T. Niwa, T. Tanaka, H. Iba, H. de Garis, and T. Furuya, "Evolvable hardware with genetic learning," Proc. of the IEEE International Symposium on Circuits and Systems, vol. 4, 1996, pp 29 – 32.

[11] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.

[12] Jan S. Lim, Two-Dimensional Signal and ImageProcessing", Prentice Hall, 1990, pp.476 – 479.