Application of Partial Reconfiguration of FPGAs in Image Processing

Tamás Raikovich, Béla Fehér
Department of Measurement and Information Systems
Budapest University of Technology and Economics
Budapest, Hungary
{rtamas, feher}@mit.bme.hu

Abstract—**FPGA based hardware accelerators have been more and more widely used in different kind of applications. As compared to other solutions and the direct hardware implementation, the advantage of the FPGA devices is their flexibility that arises from their programmable nature. In addition to this, some FPGA devices also support partial dynamic reconfiguration. When general purpose processors and reconfigurable computational structures are used together, the most advantageous properties of both components can be utilized. Algorithms (searching, sorting, signal and image processing, etc.) that are implemented such way have achieved 100 times or 1000 times better performance. This paper introduces a Xilinx Virtex-5 FPGA based reconfigurable hardware accelerator system, which has been created mainly for image processing purposes. By using the partial reconfiguration capability of the FPGA, the execution units in the system can be quickly reconfigured.**

Keywords-FPGA; partial reconfiguration; image processing; hardware accelerator

I. Introduction

Algorithms that require executing relatively simple operations on large amount of data usually can be efficiently accelerated with FPGAs. Many image processing algorithms belong to this group. Biological and biomedical experiments such as microarray experiments [1], high-content screening or cellular microscopy [2] often results in large amount of image data that have to be processed in order to evaluate the experiment.

The idea of the reconfigurable image processing solution is based on this demand for high performance batch like image processing of biological samples. The processing of the large input data set would require human intelligence to qualitatively analyze the content and needs automated steps, for quantitative analysis of the images. Our prototype tool is the open-source CellProfiler [3] image analysis software, which is a collection of Matlab based library of image processing modules, which can be used to built up different image processing pipelines. These image processing pipelines are created and tuned by the biologists, using an interactive process where the necessary modules, the optimal parameters and the classification criteria's are determined based on some representative sample images. The so called assays then applied on the thousand of images, already without direct human intervention. Our goal is to accelerate this second phase, implementing the basic CellProfiler modules in a reconfigurable FPGA fabric, and built up the processing pipeline from reconfigurable modules.

A Virtex-5 FPGA based reconfigurable test system has been created, which consists of pipelines and each pipeline consists of several reconfigurable execution units. By using the partial reconfiguration capability of the FPGA, the pipeline stages can be quickly reconfigured with the required algorithms. The test system is implemented on the Xilinx ML506 board, which utilizes an XC5VSX50T FPGA optimized for memory-intensive and DSP applications.

II. Partial Dynamic Reconfiguration

FPGA devices provide a number of serial and parallel configuration interfaces that can be used to download the configuration data to the device. During the normal configuration process, the operation of the FPGA is suspended. There are FPGA devices that support partial dynamic reconfiguration [4]. During the partial dynamic reconfiguration process, the device remains fully functional while a given part of the FPGA is reprogrammed without compromising the integrity of the running application. This capability allows using the FPGA hardware resources in a time-multiplexed way and smaller FPGAs can be used to implement the same design, which results in the reduction of costs and/or power consumption.

Partial reconfiguration is an important feature in case of reconfigurable computing. Unfortunately, very few devices on the market have the capability to be partially reconfigured. Currently, only the members of the Xilinx Virtex FPGA family support this feature. The Xilinx Virtex FPGAs have a special configuration interface called ICAP (Internal Configuration Access Port). Using this internal hardware module, the running application itself can reconfigure a part of the FPGA device.

III. Image processing Functions

The basic image processing modules of the CellProfiler are relatively simple, like Align, Apply Threshold, Average, etc. and some of the Object Processing and Measurement modules are also similar in complexity, so they can be good candidates to hardware implementation. In order to determine the resource requirements and the size of the reconfigurable regions, several image processing functions have to be examined as a reference. In the test system, the cell recognition algorithm has been chosen to implement. This algorithm determines the borders of the cells on the cellular microscopy images and executes the following image processing tasks on the input data:

- Noise removal (2D median filtering)
- Histogram computation
- Threshold value computation
- Image binarization
- Edge detection (2D FIR filtering)

The current implementation of these functions can process 8-bit grayscale images of 2048 x 2048 pixels maximum size.

A. 2D Filters

The 2D median and FIR filtering algorithms are called local image processing algorithms, because they depend on data from a relatively small neighborhood compared to the image size. These algorithms use a sliding window and execute different operations on the pixels found in the window. The sliding window image buffer can be easily implemented in hardware as shown in Fig. 1. Registers store the pixels in the 3 x 3 window, the remaining pixels (image width – 3) of the rows are stored in a dual-port Block-RAM. When a new data is written into the buffer, the window is shifted right one position.

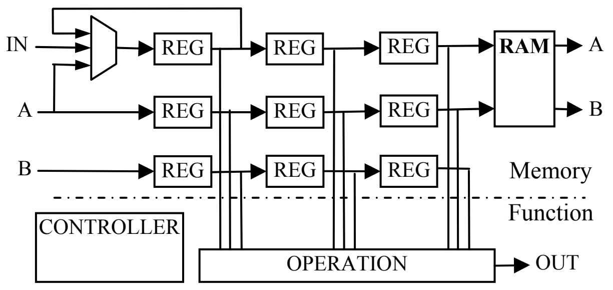


Figure 1. Block diagram of the filters.

As the window moves through the image, its center covers a smaller area than the original image, which results a smaller filtered image size. This problem can be solved in different ways. In this design, the filters process an expanded image, which contains the original image and its edges are the copy of the edges of the original image. This expanded image can be easily created in hardware by adding a multiplexer to the image buffer input, which can select different write operations.

1) Median filter

Median filtering can efficiently reduce or remove impulse like noise from the images. Its edge-preserving nature makes it useful in cases when edge blurring is undesirable. Median filters sort the pixels in the window by intensity and the center element of the window is selected as the output.

In this design, the median element is determined by the following way. At first, each row of the window is sorted (stage A). In the next step, each column of the window is sorted (stage B). At last, the elements in the secondary diagonal are sorted (stage C). The median element in the secondary diagonal will be the median element of the original inputs. The fully parallel and pipelined sorting network is shown in Fig. 2.

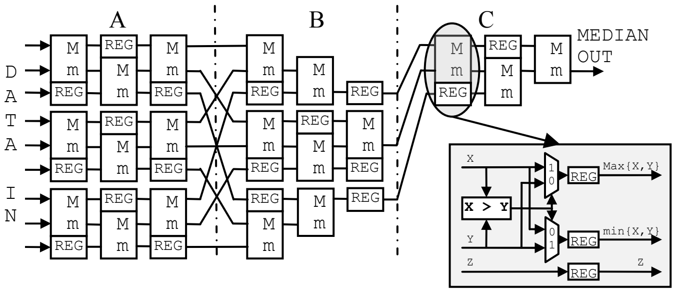


Figure 2. Network that determines the median element.

2) FIR filter

The operation executed by the FIR filters is convolution, which is defined in (1). The output of the filter is the weighted sum of the elements in the window.

$$f(x,y)=\sum_{i=-1}^{1}\sum_{j=-1}^{1}W_{i,j}\cdot image(x+i,y+j) \qquad (1)$$

By selecting the appropriate weights, convolution can implement low-pass, high-pass and band-pass frequency domain filters. Low-pass filters use positive weights and are used for image smoothing. High-pass filters use a positive center weight and negative outer weights and are used to enhance high frequency components in an image such as edges and fine detail. All of the filter coefficients ($W_{i,j}$ weights) are adjustable.

The parallel implementation of a 3 x 3 FIR filter requires nine multipliers and eight adders, both are registered at the outputs. The adders are organized in tree form to reduce the pipeline stages. The last pipeline stage is the saturation unit, which saturates the output when overflow occurs. Negative output values are possible when there are negative $W_{i,j}$ weights. These output values are meaningless and are replaced with zeroes.

B. Histogram Computation

This stage determines the distribution of the pixel intensity values of the median filtered image. In case of the specified maximum image size, the histogram computation requires a 256 x 23 bit RAM. At first, the memory is filled with zeroes, then the value addressed by the input pixel data is incremented in every step. After processing the last pixel, the RAM will contain the histogram.

C. Threshold Value Computation

The histogram-based Otsu method [5] is used to determine the threshold value for image binarization. This algorithm divides the pixels into two classes (foreground and background) then calculates the optimum threshold by separating the two classes so that their between-class variance (2) is maximal.

$$\sigma^2_{Between}(t)=P_{BG}(t)P_{FG}(t)\left[\mu_{BG}(t)-\mu_{FG}(t)\right]^2 \qquad (2)$$

In the beginning, every pixel belongs to the foreground class. After computing the initial values, the class probabilities (P_{BG}, P_{FG}) and the class means (μ_{BG}, μ_{FG}) can be updated recursively using (3), (4), (5) and (6) as each threshold value t is tested. The P(t) probability is the value in the histogram memory at address t.

$$P_{BG}(t+1) = P_{BG}(t) + P(t) \tag{3}$$

$$P_{FG}(t+1) = P_{FG}(t) - P(t) \tag{4}$$

$$\mu_{BG}(t+1) = \frac{\mu_{BG}(t)P_{BG}(t) + tP(t)}{P_{BG}(t+1)} \tag{5}$$

$$\mu_{FG}(t+1) = \frac{\mu_{FG}(t)P_{FG}(t) - tP(t)}{P_{FG}(t+1)} \tag{6}$$

The disadvantage of this algorithm is that it requires division, which cannot be as efficiently implemented in FPGAs as the multiplication. The length of the class means is 16 bits (8 integer bits and 8 fractional bits) so the standard restoring division algorithm, which is used in the current design, requires 16 clock cycles to complete.

D. Image Binarization

After computing the threshold value, each pixel of the median filtered image is tested. If the pixel intensity is lower than the threshold, it results a black binary image pixel (0). Otherwise the binary image pixel will be white (255).

IV. Reconfigurable Regions

A. Size

Table I shows the resource requirement of each image processing function. Because of the simplicity of the histogram computation and the image binarization, these are merged with the subsequent functions.

TABLE I. Resource Requirements and Utilization

Image processing function	**FPGA resource requirements and utilization**			
	LUT	***Flip-flop***	***RAMB36***	***DSP48E***
Median filter	664 / 960 (70 %)	600 / 960 (63 %)	1 / 4 (25 %)	0 / 16 (0 %)
Otsu threshold	542 / 960 (56 %)	530 / 960 (55 %)	1 / 4 (25 %)	12 / 16 (75 %)
Binarization, FIR filter	406 / 960 (43 %)	402 / 960 (42 %)	1 / 4 (25 %)	9 / 16 (57 %)

The height of the reconfigurable regions is determined by the smallest addressable configuration memory segment of Virtex-5 FPGAs, which is a 20-CLB-height frame [6]. The XC5VSX50T device has only 6 frames in a column therefore the height of each PR region should be 1 frame. PR regions with different sizes have been tried. The result of the experiments was that the place and route operation failed when more than the 75% of the logic resources were utilized within a PR region. As a consequence, an optimal reconfigurable region contains 120 CLBs (960 LUTs and FFs), 4 Block-RAMs and 16 DSP48E slices. The complexity of the XC5VSX50T device allows placing eight reconfigurable regions into a design. The remaining FPGA hardware resources are required for the static part of the system.

B. Interfaces and Connections

Fig. 3 shows the I/O interface of a reconfigurable region. Because the I/O signals routed into the reconfigurable modules through LUTs, their number should be minimized. Therefore, the PR modules have a simple data interface that consists of an 8-bit data bus and two control/status signals. The data ready signal is active when there is valid data on the data bus and the data acknowledge signal is active when the module is able to receive the next input. Every module requires a read (DATA_IN) and a write (DATA_OUT) data interface to communicate with the adjacent modules. Some functions (such as the image binarization) also require direct memory access, which is provided through the memory read (MEM_IN) and write (MEM_OUT) interfaces. The internal parameter registers can be written using the PRM_IN serial input line. The processing of the input data can be started by strobing the START input, the DONE output signals when the operation has been finished.

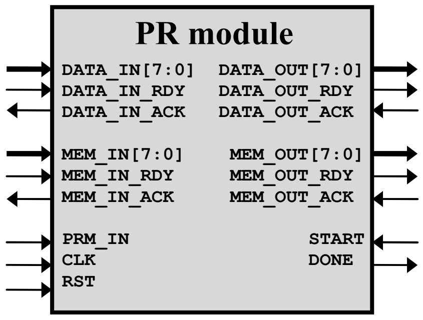


Figure 3. I/O interface of the reconfigurabl modules.

The connection of the reconfigurable regions can be seen in Fig. 4. In this design, the eight reconfigurable regions are arranged as two 4-stage pipelines. The adjacent pipeline stages are connected together through a FIFO, which helps to balance the variation of the execution speeds.

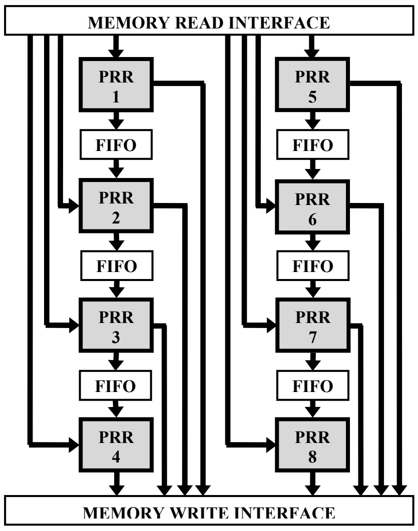


Figure 4. Connection of the reconfigurable regions.

V. Memory Interface

The memory interface greatly affects the performance of the system. Because of the limited number of the fast internal Block-RAMs, external memory is necessary for storing the large amount of image and configuration data. The ML506 board contains a 32M x 64 bit (256 MB) DDR2 SDRAM memory module, which is connected to the system through a multi-port memory controller. Assuming 125 MHz system clock frequency, the theoretical peak bandwidth of the DDR2 SDRAM is 2000 MB/s. The following peripherals of the system can directly access the external memory through the memory controller:

- Communication interface, which can be used to access the external memory and to control the system from the PC
- ICAP peripheral, which is required for the partial reconfiguration of the FPGA
- Memory read and write interfaces for the reconfigurable regions (Fig. 4)

The reconfigurable modules have simple 8-bit data interfaces, whose theoretical peak bandwidth is 125 MB/s. Comparing the bandwidth values, it can be seen that it is not worth using more than sixteen data interfaces at the same time, because memory interface would saturate.

In this design, the memory interface unit provides eight 8-bit read and eight 8-bit write ports for the reconfigurable modules. The transfers are scheduled using round-robin scheduling. Two read and write ports are connected to the first and the last stages of the pipelines. The remaining ports can be connected to the other stages in each pipeline, which allows dividing the 4-stage pipeline into smaller parts.

VI. Results

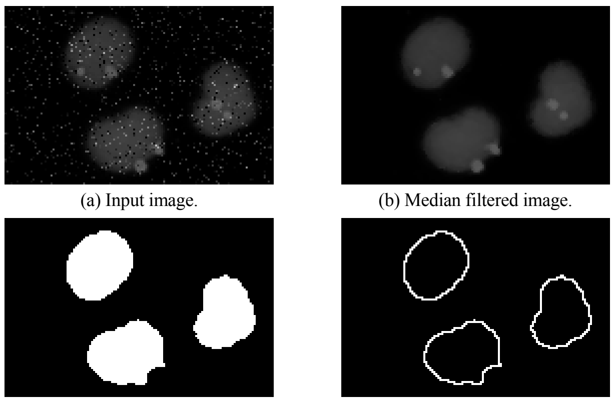
Table II shows the reconfiguration and the filtering times of the filter modules. As for the "theoretical" columns of the table, it is assumed that every input data can be processed in one clock cycle. The "theoretical" and the "actual" values are almost the same because the redesigned memory interface can provide the required bandwidth.

TABLE II. Reconfiguration and Processing Times

PR module	**Reconfiguration time @ 100 MHz**		**Processing time (512 x 512 image size)**		
			FPGA @ 125 MHz		***PC @ 2,8 GHz***
	Theoretical	***Actual***	***Theoretical***	***Actual***	
Median filter	0.178 ms	0.18 ms	2.097 ms	2.1 ms	~150 ms
Otsu threshold	0.178 ms	0.18 ms	2.138 ms	2.15 ms	~2 ms
Binarization, FIR filter	0.178 ms	0.18 ms	2.097 ms	2.1 ms	~60 ms

Comparing the hardware and software filtering times, the FPGA design performs better: the median filtering is 70 times faster and the FIR filtering is 30 times faster. In case of the threshold computation, there is no speed-up. Despite of this, it is more efficient when the threshold computation is implemented in hardware, because it is an integral part of the image processing flow. Comparing the reconfiguration and the filtering times, about 10 reconfigurations can be done while an image of size 512 x 512 pixels is processed.

Fig. 5 shows the results of the steps of the cell recognition algorithm. At first, the noisy input image (Fig. 5a) is median filtered (Fig. 5b). In the next step, the threshold value is calculated and applied to the median filtered image (Fig. 5c). In the last step, the binary image is FIR filtered (Fig. 5d) using the Laplace operator matrix as the filter mask.


(a) Input image. (b) Median filtered image.

(c) Binary image. (d) Final result.

Figure 5. The input image and the steps of the cell recognition algorithm.

VII. Conclusion

The high-performance FPGA devices offer a flexible and customizable solution for efficiently implementing wide range of applications. The partial reconfiguration capability of these devices allows utilizing them in a more efficient way. It is also an important feature in reconfigurable computing as it allows quickly swapping modules into and out of the devices without having to reset the complete device for a total reconfiguration.

References

[1] E. Wit, J. McClure, Statistics for Microarrays, Wiley, 2004.

[2] Q. Wu, F. A. Merchant, K. R. Castleman, Microscope Image Processing, Elsevier, 2008.

[3] CellProfiler cell image analysis software manual http://www.cellprofiler.org

[4] Xilinx Partial Reconfiguration User Guide Release 11.3

[5] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 9, No. 1, 1979, pp. 62-66.

[6] Xilinx UG191: Virtex-5 FPGA Configuration User Guide http://www.xilinx.com/support/documentation/user_guides/ug191.pdf