# HARDWARE STRATEGIES FOR IMAGE PROCESSING IN AN FPGA BASED MICROARRAY IMAGE PROCESSING SOC

Bogdan BELEAN Monica BORDA Albert FAZAKAS
*Technical University of Cluj-Napoca*
*Str. Baritiu 26-28,Cluj-Napoca Bogdan.Belean@com.utcluj.ro*

**Abstract: The present paper describes a System on a Chip for microarray image processing together with the steps of a microarray experiment integrated in the proposed system. The system's reduced size and the hardware algorithms proved to overcome the disadvantages of the existing software for microarray image processing. FPGA technology was chosen for the implementation due to its parallel computation capabilities and to the ease of reconfiguration. Hardware strategies for implementing image processing techniques are presented, together with a hardware implementation for edge detection of microarray spots using distributed memory and spatial computation.**

***Key words:*** *cDNA Microarrays, FPGA, image processing, hardware algorithms.*

## I. INTRODUCTION IN MICROARRAY TECHNOLOGY

A cDNA (complementary DNA) microarray experiment is a common technique used in molecular biology and medicine to measure the gene expression levels for thousands of gene in parallel. By gene expression we understand the transformation of gene's information into proteins. The informational pathway in gene expression is as follows: DNA → mRNA → protein. The protein coding information is transmitted by an intermediate molecule called messenger ribonucleic acid. This molecule passes from nucleus to cytoplasm carrying the information to build up proteins. This mRNA acid is a single stranded molecule from the original DNA and is subject to degradation, so it is transformed into stable complementary DNA for further examination.

A cDNA microarray is a glass slide or microchip on which specific single stranded cDNA probes are arrayed. Usually, samples from two sources are labeled with two different fluorescent markers and hybridized on the same array (glass slide). By hybridization we understand the tendency of two single stranded DNA molecules to bind together. After the hybridization, the array is scanned using two light sources with different lengths (red and green) to determine the amount of labeled sample bound to each spot through hybridization process. The light sources induce fluorescence in the spots, which is captured by a scanner and a composite image is produced. The most common use of cDNA microarrays is for the determination of patterns of differential gene expression, comparing differences in mRNA expression levels between identical cells subjected to different stimuli or between different cellular phenotypes or development stages [1].

Further on, image processing techniques are used to quantify gene expression levels present in the captured microarray image, in order to identify the differential gene expression between normal and abnormal cells, labeled with the two different fluorescent markers.

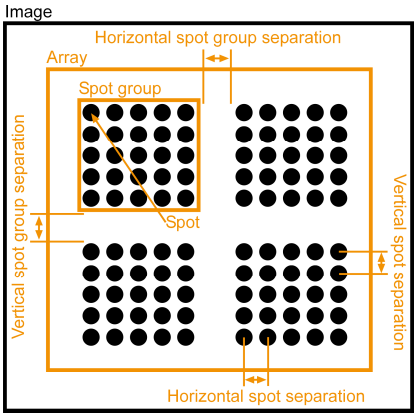


*Figure 1. Structure of cDNA microarray image*

### *I.1 EXISTING COMPUTATIONAL TOOLS*

According to the structure of the microarray image, the flow of processing a microarray image [2] is generally separated in the following tasks: addressing, segmentation, intensity extraction and pre-processing to improve image quality and enhance weakly expressed spots. The first step associates an address to each spot and spot group of the image. In the second one, pixels are classified either as foreground, representing the cDNA spots, or as background. The last step calculates spot intensities and estimates background intensity values.

The major tasks of microarray image processing, which contributes in fulfilling the last mentioned steps, are to identify the array format including the array layout, spot size and shape, spot intensities and distances between spots. The main parameters taken into consideration in microarray

*Manuscript received April 26, 2009; revised June 2, 2009*

image processing are accuracy and time. The accuracy is given by the quality of image processing techniques. The second parameter, time, is critical due to the large amount of data contained in a microarray image. A regular microarray image has up to hundreds of MB, and it can be divided in independent sub-images, which consists in a compact group of spots as it can be seen in figure 1. Sophisticated computational tools are available for microarray image processing but, their main disadvantages are the increased computational time and the user intervention needed in processing. An example of computational tool for microarray images is Feature Extraction Software [3] produced by Agilent. After applying image processing techniques over microarray images, Agilent software obtains raw data with the addresses of each spot, the spot intensity and local background intensity. The advantage of the last mentioned tool is the possibility of analyzing any kind of microarray image, but the price to be paid is the high cost. In [4], we proposed an FPGA based automated system on a chip for processing cDNA microarray images.

### *I.1 FPGA – A SOLUTION FOR MICROARRAY IMAGE PROCESSING*

Before describing the FPGA based Soc, we will describe the FPGA technology as being a solution for microarray image processing. In order to overcome the disadvantages mentioned in the previous paragraph, microarray images are analyzed and processed using FPGA technology. The hardware implementations for microarray image processing techniques make use of the features of FPGA, which allow accessing at the same time hundreds of memory addresses. In this way, calculations specific to microarray image processing are made in parallel, increasing the processing speed.

FPGA technology uses pre-built logic blocks and programmable routing resources to configure these chips and to implement custom hardware functionality. Their main benefits are the low cost, the short time to market and also the increased performance due to their structure which is able to exploit spatial and temporal parallelism. These advantages will be used to develop a system on a chip in order to process the microarray images in a manner that does not need user intervention. Also, time being critical in microarray image processing, using FPGA technology will decrease the computational time due to its parallel computation capabilities. The main goal of this approach for microarray image processing is to obtain a device which will be able to extract and quantify gene expression a lot more faster than existing computational tools, so that microarray analyses to be easily performed on a large number of subjects (thousands of patients and different diseases). This task can hardly be achieved with the existing tools which are time consuming and which also need user intervention.

This paper proposes hardware strategies for microarray image processing (paragraph 2), and also an implementation of spot border detection algorithm using distributed memory and spatial computation.

## II. FPGA BASED SOC FOR MICROARRAY IMAGE PROCESSING

Based on the advantages offered by FPGA technology, a system on a chip is built around a CCD sensor, in order to obtain a hand held device which swaps a double-laser over the microarray glass slide obtained after hybridizing. Due to the laser excitation fluorescence levels are produced. A highly sensitive CCD (charge coupled device) sensor is used to store the microarray image composed by the fluorescence levels found along the glass slide, after hybridization. Once the microarray image is captured, hardware strategies will be applied to fulfill the following steps, specific for microarray image processing: preprocessing, addressing, segmentation and intensity extraction. The architecture built around the CCD sensor to capture and store the microarray image is presented in [4]. The image processing techniques will have the same aim as the existing software for microarray image processing. For example, Agilent Feature Extraction Software produces raw data with the position of all spots, the spot's center, spot's intensity, and local background intensity for each spot. In order to obtain the same results, our FPGA based microarray image processing system will work as follows:

a) Image enhancement will be done using a logarithm transformation to enhance weekly expressed spots.

b) Addressing will determine addresses of each spot, by calculating on the enhanced image horizontal and vertical profiles. The horizontal (HP) and the vertical projections (VP) are calculated as follows:

$$HP(y) = \frac{1}{X}\sum_{x=0}^{X-1} I(x, y) \quad (1)$$

$$VP(x) = \frac{1}{Y}\sum_{y=0}^{Y-1} I(x, y) \quad (2)$$

where x = 1, 2,..X-1 and y = 1, 2,..Y-1 are the dimensions of the enhanced microarray image I(x,y).

c) Segmentation is done using edge detection methods and spatial and temporal parallelism. Based on the detected edge, Hough transform is applied to determine the center of the spots and the radius.

d) Intensity extraction and local background for each spot is made using the radius, the center of the spot and of course the address determined at point b).

This paper will focus on the hardware strategies for image processing, used to replace the conventional algorithms used by software like Agilent Feature Extraction Software for microarray image processing.

## III. HARDWARE STRATEGIES FOR MICROARRAY IMAGE PROCESSING

### *III.1 IMAGE CONVOLUTION*

Image processing operations like median filters and gradient calculation are based on the convolution, which is included in a class of algorithms called spatial filters. Convolution is used for implementing image operators, which have as output pixel value a linear combination between pixels of the original image. Conceptually, each pixel in the output image is produced by sliding an NxM window over the input image and computing an operation according to the input pixels under the window and the chosen window operator. The hardware approach for convolution is presented as follows: the entire input image is stored into a frame buffer; each time the window is moved, MxN pixels values are required to calculate the resulted pixel value. Memory bandwidth constraints make obtaining all these pixels each clock cycle impossible, so local caching is performed [6]. In this way, N-1 rows are cached using a shift register, which leads to the next block diagram:

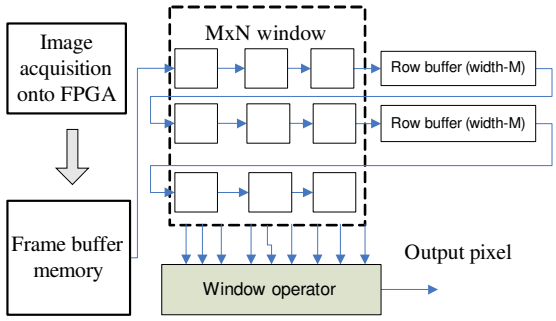


*Figure 2: Block diagram for hardware implementation of convolution operator.*

In this case, instead of sliding a window across the image, the implementation feeds the image through the window.

*III.2 TEMPORAL PARALLELISM*

Based on the possibility of cropping a microarray image into independent sub-images, a strategy for microarray image processing emerges. The idea is to design an architecture which processes in parallel two or more independent sub-images. Once the address and dimension of each independent sub-image is determined, two ore more microarray sub-images can be copied into the FPGA block RAM depending on the capacity of block RAM. In figure 3 a cropped microarray image which consists in a matrix of independent sub-images called A(i,j) is presented. The numbers from 1 to 3 and from 1' to 3' represent the operations applied on two independent microarray sub-images. This way, operation 1 stands for copying the sub-image from RAM to the FPGA block RAM. Operation 2 represents the processing of the sub-image using FPGA technology and spatial parallelism. Operation 3 copies the microarray processed sub-image back into the RAM, on the exact place from which it was transferred into FPGA. The operations 1, 2 and 3 are done for each microarray sub-image. In the end we will have in the RAM memory a processed microarray image.

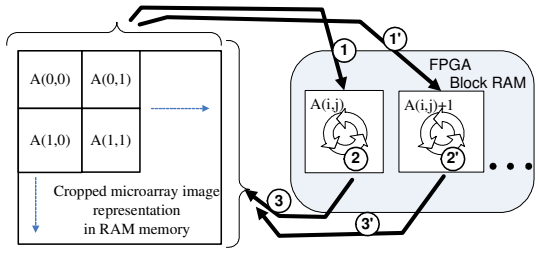


*Figure 3: Block diagram for pipeline architecture of microarray image processing*

Time being critical in microarray image processing, the previous approach reduces the processing time to half of its regular value. As an example we can mention that, while a sub-image is copied into the block RAM, image processing techniques are applied on another independent sub-image, in case that there is place for more than one microarray independent sub-image in RAM.

*III.3 SPATIAL PARALLELISM*

To start with, it is to be mentioned that processors divide computation across time, while dedicated logic divides it across space, which highly decreases computational time. Taking into consideration the large amount of data contained in a microarray image and also the repetitive nature of this type of data, FPGA dedicated logic was chosen for the implementation of microarray image processing techniques in order to reduce computational time. The use of hardware description language (HDL) allows a description of a design with parallel data paths and simultaneous computation.

Some of the major tasks in microarray image processing are to extract spot intensities and to determine spot contour and dimension. The dimension of a regular spot is around 20-40 pixels height and length. The reduced dimension of a microarray spot offers the possibility to exploit spatial parallelism capabilities of FPGAs. Each spot is copied into the distributed RAM of the FPGA, so spatial computation can be applied, thanks to the simultaneous access to each pixel.

## IV. EDGE DETECTION USING DISTRIBUTED MEMORY AND SPATIAL COMPUTATION

This paper presents a hardware implementation of an adaptive edge detection filter using FPGA, which provides the necessary performance for fast microarray image processing. In microarray image processing, edge detection is a fundamental tool used as a precursor step to intensity extraction and spot segmentation. Edges occur at images location with strong intensity contrast. For edge detection a high-pass filter in Fourier domain can be applied, or convolution with an appropriate kernel (Sobel, Prewitt etc.) in the spatial domain is useful. Convolution in the spatial domain has been chosen for implementation because it is computationally less expansive and offers better results.

The algorithm used for the hardware implementation is Canny algorithm, which is considered to be optimal, based on the following: it finds the most edges, marks the edge as close as possible to the actual edges, and provides sharp and thin edges. The filter that meets all the criteria mentioned above can be efficiently approximated using the first derivative of a Gaussian function. So the first two steps in applying the Canny algorithm would be smoothing the image and differentiating the image in two orthogonal directions. The next step, non-maximum suppression, computes the gradient direction and magnitude in order to eliminate the pixels that represent false edges.

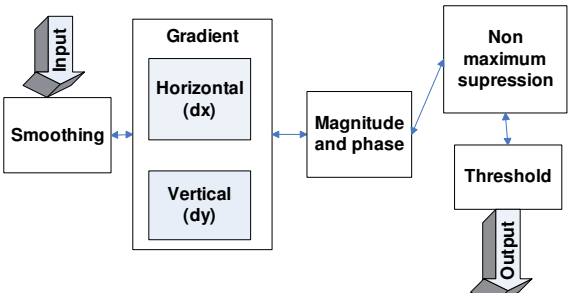


*Figure 5: Block diagram for Canny algorithm*

The previously described algorithm is applied on a microarray spot. Due to its small dimension (approximately 40x40 pixels), the whole spot is copied into the FPGA, so

hardware processing strategies (spatial parallelism) described in the previous paragraph can be applied. Further on, the hardware implementations for each of the previous steps are presented.

Smoothing operation is done using the following convolution mask, because it contains a division by 28 that is easily done with an 8 bits shift operation:

$$\frac{1}{256}\begin{matrix} 21 & 31 & 21 \\ 31 & 48 & 31 \\ 21 & 31 & 21 \end{matrix}$$

The effect of the previous Gaussian convolution is to blur the image, eliminating noise. As it can be seen in the next diagram, for processing 1 pixel only one clock cycle is needed, because all the neighborhood pixels used in computation are accessed at the same time.

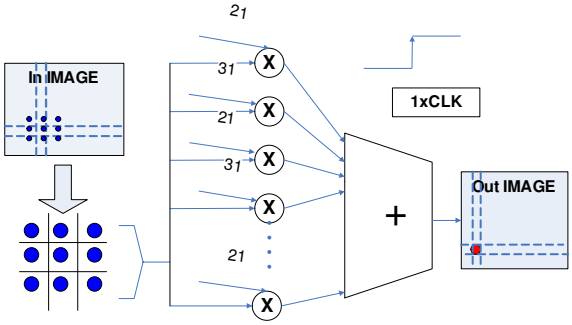


*Figure 5: Gaussian filter implementation*

After smoothing the image, the next step is gradient calculation in order to find the edge strength of the spot. To determine the orthogonal gradient at each pixel location, the following convolution masks are used:

$$\begin{matrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{matrix} \quad \text{and} \quad \begin{matrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{matrix}$$

In this case, only arithmetic operations of addition are used, so for the whole microarray spot written in FPGA distributed RAM, a single clock cycle and 2x40x40 adders are enough to determine the resulted image after applying the convolution masks.

The sign and value of the orthogonal components of the gradient determined before are used in estimating the magnitude and the direction of the gradient. Once the direction of the gradient is known, pixels values around the pixel being analyzed are interpolated. The pixel that does not represent a local maximum is eliminated, by comparing it with its neighbors along the direction of the gradient (non maximum suppression). Combinational logic and basic operations of addition and shifting are used in implementation, so each clock cycle one pixel it's processed.

EXPERIMENTAL RESULTS

The hardware platform used for implementation includes the XC3S5000 and a quartz oscillator with a frequency of 50 MHz used to generate the clock signal. Summing up the time needed for each step of the edge detection algorithm applied on a 20x20 pixels spot, a processing time of 3.2 µs is obtained. This results leads to a processing rate of approximately 250 Mbytes/second.

| Device Utilization Summary (estimated values) | | | |
|---|---|---|---|
| **Logic Utilization** | **Used** | **Available** | **Utilization** |
| Number of Slices | 22881 | 33280 | 68% |
| Number of Slice Flip Flops | 14965 | 66560 | 22% |
| Number of 4 input LUTs | 43360 | 66560 | 65% |
| Number of bonded IOBs | 11 | 633 | 1% |
| Number of MULT18X18s | 9 | 104 | 8% |
| Number of GCLKs | 8 | 8 | 100% |

*Table 1: Statistics of FPGA (XC3S5000) resource usage for edge detection of microarray spot*

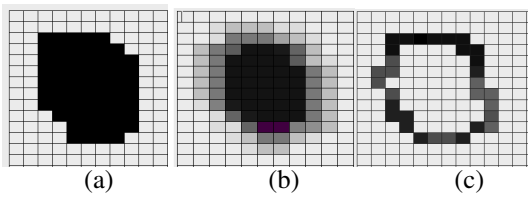


*Fig. 6: Preliminary results obtained on a microarray spot; a) original image b) smoothed image c) edge detection*

## V. CONCLUSIONS

The hardware implementation strategies for image processing presented in this paper represent an essential step for microarray image processing on a hardware platform. The results of the hardware implementation for edge detection in cDNA microarray image emphasize the importance of using hardware methods in cDNA microarray image processing. The technology chosen to implement a digital system for microarray image processing was FPGA, due to its parallel computation capabilities and to the possibility of reconfiguration.

To sum up, the experimental results made this hardware technology a solution for realizing a fast, cost efficient and accurate automated system for cDNA microarray image processing.

## ACKNOWLEDGEMENTS

This work was supported by PNII IDEI Nr. 332/2007 grant, code 909 and also by CNCSIS BD scholarship.

## REFERENCES

[1] P. Bajcsy, *An Overview of DNA Microarray Image Requirements for Automated Processing*, IEEE Transactions on Image Processing, Vol. 13, No. 1, pp. 15-25, 2004

[2] Y. Yang, M. Buckley, S. Dudoit, *Comparison of methods for image analysis on cDNA microarray data*, Department of statistics - University of California – Berkeley, Technical Report 584, 2001.

[3] www.agilent.com (Feature Extraction Software Reference Manual 10.1 - 2008)

[4] B. Belean, M. Borda, A. Fazakas, *Adaptive Microarray Image Acquisition System and Microarray Image Processing Using FPGA Technology,* ANSI Evolvable Hardware and Adaptive Systems, Springer, pp. 321-327, 2008.

[5] Q. Li, C. Fraley, R. Baumgarner, K. Yeung, and E. Raftery, *Donuts, scratches and blanks: Robust model-based segmentation of microarray images,* Department of statistics - University of Washington, Technical Report 473, 2006.

[6] C. Johnston, *Implementing image processing algorithms on FPGA*, Institute of Information Sciences and Technology, Palmerston, New Zeeland, 2004.