FPGA Technology And Parallel Computing Towards Automatic Microarray Image Processing

Bogdan Belean, Monica Borda, Bertrand LeGal, and Raul Malutan

Abstract—**Automation, computational time and cost are open subjects in microarray image processing. The present paper proposes image processing techniques together with their implementations in order to eliminate the shortcomings of the existing software platforms for microarray image processing: user intervention, increased computational time and cost. Thus, for each step of microarray image processing, application-specific hardware architectures are designed aiming algorithms parallelization for fast processing. Computational time is estimated and compared with state of the art approaches. The proposed hardware architectures integrated inside microarray scanners deliver microarray image characteristics in an automated manner, excluding the need of an additional software platform. The FPGA technology was chosen for implementation, due to its parallel computation capabilities and ease of reconfiguration.**

Keywords—**Microarray, image processing, parallel computing**

I. INTRODUCTION IN DNA MICROARRAY TECHNOLOGY

MEASUREMENT of gene expression can provide clues about regulatory mechanism, biochemical pathways and broader cellular function. 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 mRNA. This molecule passes from nucleus to cytoplasm carrying the information to build up proteins [1].

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. Microarray technology is based on creating DNA microarrays which glass slide or microchip. Usually samples from two sources are labeled with two different fluorescent markers and hybridized on the same array (glass slide). The hybridization process represents the tendency of 2 single stranded DNA molecules to bind together. After 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 [2], which is further on processed to determine spots characteristics in order to estimate gene expression levels. The most common use for DNA microarrays is to measure, simultaneously, the level of gene expression for every gene in a genome [3]. In this way the microarray compares genes from normal cells with abnormal or treated cells, determining and understanding the genes involved in different diseases. The microarray technology is used also in toxicological research and monitoring environmental effects on different genomes.

The classical flow of processing a microarray image is generally separated in the following tasks: addressing, segmentation, intensity extraction and pre-processing to improve image quality and enhance weakly expressed spots [4]. The first step associates an address to each spot of the image. In the second one, pixels are classified either as foreground, representing the DNA spots, or as background. The last step calculates the intensities of each spot and also estimates background intensity values. The major tasks of microarray image processing are to identify the microarray image characteristics including the array layout, spot locations, size and shape, and to estimate spot intensity values. Existing software platforms (e.g. Agilen Feature Extraction, GenePix, ScanAlyze [5] etc.) for microarray image processing provide raw-data with microarray image characteristics organized in an *.xls* form (Table I), which are further on used in high order analyses like clustering and gene regulatory network estimation. As Table 1 shows, each microarray spot represents a specific gene, and it has a precise location.

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. Sophisticated computational tools and software platforms are available for microarray image processing. Their main disadvantages are the long runtime and the user intervention needed in processing. The well-defined

TABLE I.
RAW-DATA PARAMETERS FOR "MUS-MUSCULUS" EXPERIMENT DELIVERED BY AGILENT FEATURE EXTRACTION SOFTWARE

Row	Col	GeneName	PositionX	PositionY	PValLog Ratio
1	1	BrightCorner	395,618	100.5	7.68E+08
1	2	NegativeCtrl	415,962	995,462	9.03E+08
1	3	Psma5	437,833	100,891	8.90E+08
1	4	Mmp14	459,123	998,774	9.25E+08
…	…	…	…	…	…

Manuscript received May 20, 2011. This work was supported by the project "Development and support of multidisciplinary postdoctoral programmes in major technical areas of national strategy of Research - Development - Innovation" 4D-POSTDOC, contract no. POSDRU/89/1.5/S/52603, project co-funded by the European Social Fund through Sectoral Operational Programme Human Resources Development 2007-2013I.

B. Belean is with the Communication Department, Technical University of Cluj-Napoca, 26-28 George Baritiu St., 400027 Cluj-Napoca, Romania (phone/fax 004-264-401575, bogdan.belean@com.utcluj.ro).

M. Borda is with the Communications Department, Technical University of Cluj-Napoca (email: Monica.Borda@com.utcluj.ro).

B. LeGal is with the Laboratory IMS - Intégration des Matériaux au Systèmes, University of Bordeaux, Bordeaux, France (bertrand.legal@)ims-bordeaux.fr).

R. Malutan is with the Communications Department, Technical University of Cluj-Napoca (e-mail:raul.malutan@com.utcluj.ro).

structure of microarray images imposes automatic image processing in order to extract microarray spot characteristics. Based on the next image processing algorithms, our paper proposes an FPGA based system which can be integrated into the microarray scanner to automatically process microarray images.

II. Algorithms For Automated Microarray Image Processing

User intervention in microarray image processing brings up the need of a work station with a costly processing platform which will slow down the process of microarray analyses in case of large number of subjects is involved. In order to overcome the previous mentioned disadvantages, the following approaches are taken into account: image processing algorithms will be robust and independent of operator last time adjustments; microarray images are processed using FPGA in order to speed up computation.

A. Microarray image enhancement

Image pre-processing techniques are used in order to improve image quality and to enhance weakly expressed spots. The most common techniques used for microarray image enhancement is the spatial logarithm transformation. In Fig. 1, an original image together with results using logarithm transformations (1) are presented.

$$IL(x,y)=\frac{\ln(I(x,y)+1)}{n\ln 2}\cdot 2^{n} \qquad (1)$$

B. Microarray image addressing

For microarray image addressing an automatic estimation of spot distance is presented. After the pre-processing of the microarray images, the first step for spot localization is the computation of image projections as described in (2). It can be assumed that the profiles resulting from these projections contain a periodic signal which has been affected by noise.

$$HP(y)=\frac{1}{X}\sum_{x=0}^{X-1}I(x,y) \qquad (2)$$

To be able to find the periodicity, the signal is cross-correlated with itself, procedure called autocorrelation. In the discrete case autocorrelation is defined as in (3).

$$pv(i)=\sum_{j=0}^{X-1}HP(i)\cdot HP((i+j)\bmod X) \qquad (3)$$

$$VP(x)=\frac{1}{Y}\sum_{y=0}^{Y-1}I(x,y) \qquad (4)$$

with *I(x, y)* being the microarray image, *X* and *Y* image dimensions, *i = 0, 1,...,X-1*. The first derivative of the resulted array *pv(i)* crosses the *X* axis in points corresponding to the peaks and values of the spots. Taking the distance between zeros, the average dimension of the spots is estimated. Microarray spot localization using image pro-files can be seen in Fig. 2.

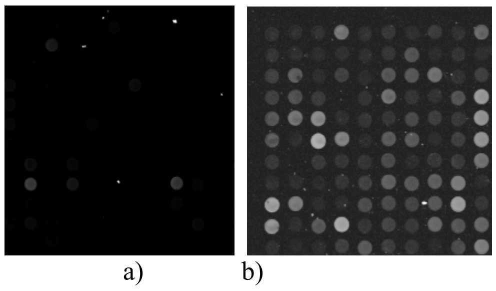

Figure 1. a) Original image, b) Logarithmic transformed image.

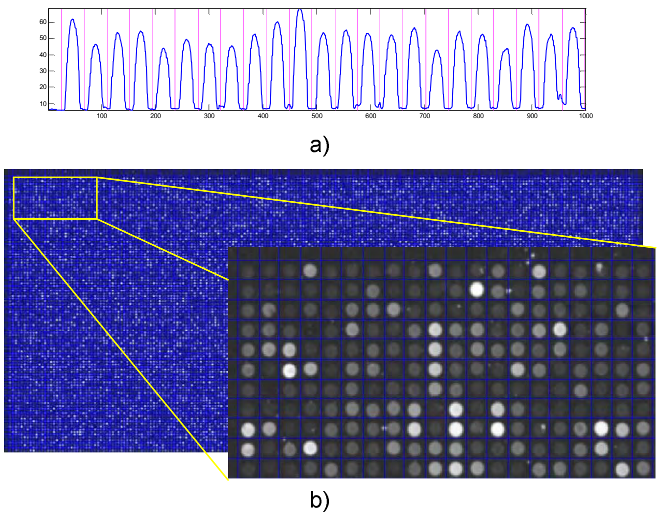

Figure 2. a) Section of the image vertical profile with vertical separators between microarray spots; b) grid alignement using autocorelation

C. Microarray image segmentation

In microarray image processing, edge detection is a fundamental tool used for 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 do-main is useful [7]. 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 filter [8], 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 Canny filter would be smoothing the image and differentiating the image in two orthogonal directions. Smoothing operation is done using convolution mask. After smoothing the image, gradient calculation (magnitude and phase) is performed in order to find the edge strength of the spot. To do so, the image is differentiated on two orthogonal directions using image convolution. 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 maxi-mum is eliminated, by comparing it with its neighbors along the direction of the gradient (non-maximum suppression). Up to this point, image processing algorithms were presented in order to realize a robust detection of microarray image features. A solution for implementing the previous image processing chain is presented next.

III. Hardware Implementations For Microarray Image Processing Algorithms

FPGA technology uses pre-built logic blocks and programmable routing resources for configuration and for implementing custom hardware functionality. Their main benefits are the low cost, the short time to market and the ease of reconfiguration. Microarray images are analyzed and processed using FPGA technology in order to speed up computation. The hardware implementations of micro-array image processing techniques make use of the FPGA features, which allow accessing at the same time hundreds of memory addresses. Indeed, FPGA technology offers the possibility to exploit spatial and temporal parallelism for microarray image processing in order to create a fast automated process which delivers raw-data information about microarray image characteristics [6]. Further on, an FPGA based system for microarray image processing is described where Xilinx board Virtex5 ML505 was used for architectures development. The system includes 3 processing units *PU_i*: *PU_1* realizes the microarray image enhancement, PU_2 computes image vertical and horizontal profiles and the last processing unit *PU_3* uses spatial parallelism for image segmentation. The proposed application specific hardware architectures are simulated and tested using the designed presented in Fig. 3.

A. Microarray image enhancement implementation

Spatial logarithm transformation is used for microarray image enhancement. The logic bloc LOG from Fig. 4 computes the logarithm transformation applied on the luminance information, Y, of the microarray image.

The hardware implementation of the logarithm transformation is based on linear approximation of the logarithm function. The logarithm function is calculated in a number of *$A_n(x,y)$* points stored in a memory named ROM_LOG. Also the slope *m* for each line described by two adjacent points is calculated and stored in a memory called *ROM_SLOPE*. In order to calculate the logarithm of the luminance, we are using (6) which represent the equation of a line which has the slope m and passes through the point *$A_i(x_i,y_i)$* from the initial *A_n* points.

$$y_{log}=m(y-x_i)+y_i \qquad (6)$$

The proposed architecture reduces the computational time for the logarithm unit to 1 pixel/clock cycle with an initial 3clock cycles delay. In order to evaluate the log function estimation, mean square error was calculated for y values between 1 and $Y_{MAX} = 256$ and the result is shown in (7).

$$MSE=\frac{1}{Y\max}\sum_{y}[\ln(y)-\ln\_est(y)]^2=1.807\cdot 10^{-5} \qquad (7)$$

B. Microarray image profiles computation

Computing the horizontal and vertical image profiles for spot localization involves logarithm computation of pixel intensity. In Fig. 5 is described the hardware architecture for evaluating image profiles. The spatial logarithm transformation is applied on the luminous image component *Y* for enhancement. The Σ_X and Σ_Y RAM memories and the two adders are used as accumulators for horizontal and vertical profiles while the whole image is scanned. Once the profiles are calculated, spot location are determined as shown in Fig. 2 using discrete autocorrelation. The spot locations are delivered as partial results for further processing. The next processing step is microarray image segmentation based on spatial convolution, which aims to extract specific microarray parameters, delivered as raw data parameters.

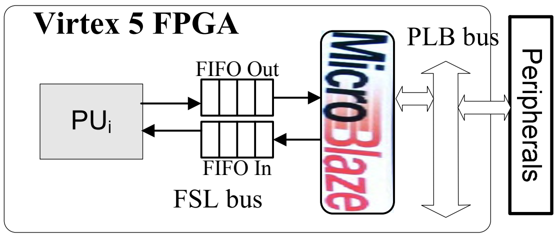

Figure 3. Hardware architecture for profile computation

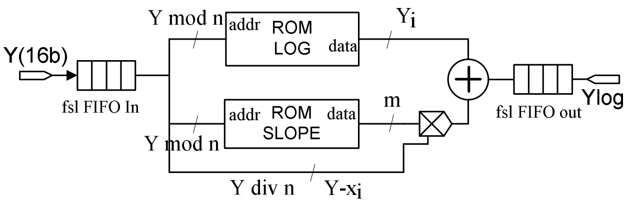

Figure 4. Hardware implementation for logarithm function

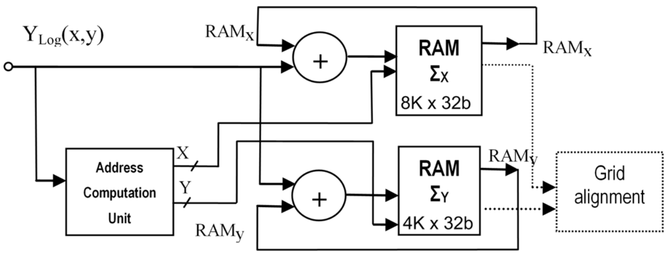

Figure 5. Hardware architecture for profile computation

C. Microarray image segmentation

This section presents a hardware implementation of Canny edge detection filter using FPGA technology, which provides the necessary performance for fast microarray image processing. The previously described algorithm is applied on a microarray spot. Due to its small dimension (approximately 30x30 pixels), the whole spot is copied into FPGA distributed RAM, so spatial parallelism is applied. Preliminary results for a 12x12 microarray spot are presented in Fig. 6.

The Xilinx Virtex5 xc5vlx110t FPGA was used for the hardware implementation of the previous segmentation method. The hardware resource usage is 26% of available FPGA slices. Summing up the computational time needed for each step of the border detection implementation we obtained a total processing time of 60 ns for a microarray spot, as

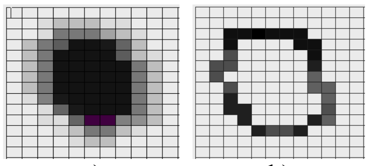

Figure 6. Preliminary results obtained on a microarray spot; a) smoothed image b) edge detection

TABLE II.
PARALLELIZATION LEVELS AND TIMING

Image processing algorithms	*Level of parallel.*	*Input data*	*Processing time*
1. Log. transformation	(MxN)/p	≈100 MB	527 ms
2. Image profiles	2	≈100 MB	264 ms
3. Autocorrelation	2	(M+N)x16kb	100 ms (estimated)
5. Canny filter	k	≈100 MB	306 ms

reported in [6]. Future work aims developing a customizable processing unit for a microarray spot in order to deliver fast segmentation results. Due to the independent processing for each spot, the processing unit can be cloned for computing more than one spot at a time.

IV. Timing consideration and Parallelization levels

Computational time was estimated considering the proposed hardware architectures applied on a 6100x2160 pixels Agilent image. The number of microarray spots k is 22575. The Microblaze design used for simulations works at 100MHz clock frequency. Results are depicted in table 2 under *processing time* column, where a total processing time of 1,197 sec. is obtained for a whole microarray image. It is to be mentioned that parallelization levels described next were not taken into account. Nevertheless, the state of the art processing time is reported to be 10 ms [9], which is nearly one order of magnitude larger than the time required by the proposed approach.

The levels of parallelization for the previously described image processing algorithms using the proposed hardware architectures are discussed next. In case of image enhancement, we consider *M, N* the image dimensions and p the number of logarithm computation units. Due to the independent computation of logarithm for each pixel, the maximum level of parallelization for image enhancement is *(MxN)/p*. For spot position estimation, the level of parallelization is *2*, one architecture being for each of the image profiles. Due to the recursive description of the autocorrelation and image profile computation, algorithms cannot be easily parallelized. Nevertheless, they are not applied over the full image; as a consequence, the parallelization is not mandatory. Once the spot locations are estimated, k being the number of spots, multiple instances for filters like Sobel or Canny for image segmentation can be used. The maximum parallelization level is k. In Table II parallelization levels are listed together with the computation time for the microarray image processing algorithms.

The computational time of 1.197 sec. for the whole image processing chain can easily be improved using the proposed parallelization levels.

V. Conclusions

The present paper proposes hardware implementations for microarray image processing algorithms taking advantage of the parallel computation capabilities offered by the FPGA technology. The levels of parallelization introduced by the proposed hardware architectures for microarray image processing bring up reduced computational time. Thus, an automated system can be developed and integrated at the microarray scanner level aiming fast microarray image processing. Due to its reduced computational time and cost, a larger number of microarray analyses can be performed compared with the existing software platforms which involve user intervention in processing. To conclude, the experimental results proved that FPGA technology is an efficient solution for automated microarray image processing.

References

[1] Mark Schena, "Micropuce Biochip Technology," Oxford University Press, 1999.

[2] Peter Bajcsy, "An Overview of DNA Microarray Image Requirements for Automated Processing," IEEE Transactions on Image Processing, vol. 13, no. 1, pp. 15-25, 2004.

[3] A. M. Campbell, W. T. Hatfield, L. J. Heyer, "Make mi-croarray data with known ratios," CBE – Life Sciences Edu-cation, vol. 6, 196-197, 2007.

[4] D. Bariamisa, D. Maroulisa, D. K. Iakovidisb, "Unsupervised SVM-based gridding for DNA microarray images", Computerized Medical Imaging and Graphics, vol. 34, pp. 418-425, 2010

[5] M.B. Eisen, "ScanAlyze User Manual," Stanford University, 1999.

[6] B. Belean, M. Borda, A. Fazakas, "Adaptive Microarray Image Acquisition System and Microarray Image Pro-cessing Using FPGA Technology," Lecture Notes in Computer Science 5179, pp. 327-334, 2008.

[7] Tanvir A. Abassi, Usaid Abassim, "A Proposed FPGA Based Architecture for Sobel Edge Detection Operator," *Journal of Active and Passive Electronic Devices*, pp. 271–277, 2007.

[8] J. F. Canny, "A computational approach to edge detec-tion," IEEE Trans Pattern Analysis and Machine Intelli-gence, vol. 8, no. 6, pp. 679-698, Nov. 1986.

[9] D. Bariamis, D. K. Iakovidis, D. Maroulis, "M3G: Maximum Margin Microarray Gridding", BMC Bioinformatics, vol. 11(49), 2010.