FPGA based system for automatic cDNA microarray image processing

Bogdan Belean\textsuperscript{a,b,*}, Monica Borda\textsuperscript{a}, Bertrand Le Gal\textsuperscript{c}, Romulus Terebes\textsuperscript{a}

\textsuperscript{a} Technical University of Cluj-Napoca, Department of Communication, 71-73 Dorobantilor, 400609 Cluj-Napoca, Romania  
\textsuperscript{b} National Institute for Research and Development of Isotopic and Molecular Technologies, 65 - 103 Donath, 400293 Cluj-Napoca, Romania  
\textsuperscript{c} University of Bordeaux 1, IMS Laboratory, 351 Cours de la Libération, Talence Cedex, France

ARTICLE INFO

Article history:
Received 5 July 2011
Received in revised form 23 November 2011
Accepted 26 January 2012

Keywords:
Microarray  
Gene expression  
Image processing  
Parallel computation  
FPGA technology

ABSTRACT

Automation is an open subject in DNA microarray image processing, aiming reliable gene expression estimation. The paper presents a novel shock filter based approach for automatic microarray grid alignment. The proposed method brings up significantly reduced computational complexity compared to state of the art approaches, while similar results in terms of accuracy are achieved. Based on this approach, we also propose an FPGA based system for microarray image analysis that eliminates the shortcomings of existing software platforms: user intervention, increased computational time and cost. Our system includes application-specific architectures which involve algorithm parallelization, aiming fast and automated cDNA microarray image processing. The proposed automated image processing chain is implemented both on a general purpose processor and using the developed hardware architectures as co-processors in a FPGA based system. The comparative results included in the last section show that an important gain in terms of computational time is obtained using hardware based implementations.

© 2012 Elsevier Ltd. All rights reserved.

1. Introduction in cDNA microarray technology

Measurement of gene expression can provide clues about regulatory mechanism, biochemical pathways and broader cellular function. Molecular biology and bioinformatics are using microarray technology in order to identify genes in biological sequences and to determine their functionality and their expression levels under different conditions. Genes are known as portions of DNA molecule that encode for a type of protein. By gene expression we understand the transformation of gene’s information into proteins. The informational pathway in gene expression is as follows: DNA $\rightarrow$ mRNA $\rightarrow$ 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 cDNA (complementary DNA) for further examination. Microarray technology is based on creating cDNA microarrays which represents gene specific probes arrayed on a matrix such as a glass slide or microchip [2]. Usually, samples from two sources (cDNA from target sample and cDNA from reference sample) are labelled with two different fluorescent markers (cyanine 3–Cy3 and cyanine 5–Cy5, respectively) and hybridized on the same array (glass slide). The hybridization process represents the tendency of two single stranded DNA molecules to bind together. After hybridization, the array is scanned using two light sources with different wavelengths for each marker (red and green) to determine the amount of labelled 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] (Fig. 1). In this way, microarrays compare genes from normal cells with abnormal or treated cells, determining and providing information for understanding the genes involved in different diseases [3]. The microarray technology is used also in toxicological research and monitoring environmental effects on different genomes.

Classical genomic microarray experiments involve complex steps including slide production and scanning. A brief description of a microarray experiment can be summarized as follows:

1. Generation of array ready cDNA (selecting specific cell material and using Polymeric Chain Reaction for DNA amplification);
2. cDNA selection and microarray slide printing;
3. Selection of specific cell material from target tissues to be tested and fluorescent labelling;
4. Hybridization of the target material on the microarray slide;
5. Microarray image scanning;
6. Image filtering and spot detection;
7. Intensity extraction in order to evaluate gene expression;
8. High order processing (Clustering and interpretation, gene regulatory network estimation).

Please cite this article in press as: Belean B, et al. FPGA based system for automatic cDNA microarray image processing. Comput Med Imaging Graph (2012), doi:10.1016/j.compmedimag.2012.01.002
used further on in high order analyses like clustering and gene regulatory network estimation. An example of raw-data information delivered by Agilent Feature Extraction software as a result of a microarray experiment is described in Fig. 1, where each information line corresponds to a microarray spot which has a precise location and represents a specific gene.

For each of the proposed microarray image processing techniques, using FPGA technology and taking advantage of its parallel computation capabilities, we designed application specific hardware architectures. All together they describe an FPGA based system for fast and automated microarray image processing and acquisition. The proposed system can be either integrated into the microarray scanner to automatically deliver results or into another device designed for remote microarray scanning and processing.

2. Automatic image processing techniques for cDNA microarray images

Spot detection and intensity extraction, included in a microarray experiment workflow, are fulfilled using image processing techniques. Recent research developed several microarray image processing methods specific to cDNA microarray analysis which provide grid alignment, spot segmentation and spot intensity extraction. This section details each step of cDNA microarray image processing by presenting the state of the art and also our proposed image processing techniques. The novel shock filter based approach for automatic image addressing is compared regarding the computational complexity to the state of the art approaches. Moreover, the accuracy of the proposed method in the presence of artefacts is illustrated compared to SVM and OMTG approaches reported in [8] and [9] respectively. In order to validate our results in terms of accuracy and reliability of spot detection, GEO (Gene Expression Omnibus), a MIAME compliant database was used to provide for comparison between different microarray images and the correspondent results delivered by existing software platforms for microarray image processing.

2.1. Microarray image enhancement

A well-known characteristic of microarray images delivered by existing scanners is the low level of expression for microarray spots, determined by their pixel intensity. Thus, the microarray image processing workflow commonly starts with a point-wise nonlinear transformations, used in order to improve image quality and to enhance weakly expressed spots [5,10]. One can use a logarithm transformation as shown in Eq. (1). The output, for a microarray image \( I(x,y) \) with \((x,y)\) denoting the coordinates of a pixel and \(n\) the number of bits for luminance/chrominance function representation, is described by:

\[
I_{l}(x, y) = \frac{\ln[I(x, y) + 1]}{n \ln 2}
\]

Alternatively, an arctangent hyperbolic based transformation can be used for image enhancement [11]. In case of such a nonlinear transformation only foreground (spot) information is selectively enhanced.

2.2. Automatic microarray image addressing (grid alignment)

The first operation performed on microarray image is known as addressing or grid alignment. This operation aims registering a set of horizontal and vertical lines which describe a two-dimensional array of spots. The existing software platforms for microarray image analysis together with late research impose two approaches for grid alignment, template-based and, respectively, data-driven methods [12]. Currently available software like GenePix Pro (Molecular

---

### Table 1: Raw data parameters for *Mus Musculus* experiment delivered by Agilent Feature Extraction software

<table>
<thead>
<tr>
<th>GeneName</th>
<th>PositionX</th>
<th>PositionY</th>
<th>Signal</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>AW219304</td>
<td>507.464</td>
<td>100.106</td>
<td>3.30E+03</td>
<td>4.39E+01</td>
</tr>
<tr>
<td>C5847437</td>
<td>528.834</td>
<td>101.483</td>
<td>3.65E+02</td>
<td>4.59E+01</td>
</tr>
<tr>
<td>TCS54731</td>
<td>550.349</td>
<td>99.906</td>
<td>3.08E+02</td>
<td>6.25E+01</td>
</tr>
<tr>
<td>Pro25G</td>
<td>571.669</td>
<td>100.822</td>
<td>3.20E+03</td>
<td>1.51E+03</td>
</tr>
<tr>
<td>TmRps2</td>
<td>592.255</td>
<td>99.8414</td>
<td>3.62E+02</td>
<td>6.45E+01</td>
</tr>
<tr>
<td>RGD1304622</td>
<td>613.64</td>
<td>100.673</td>
<td>4.99E+02</td>
<td>8.72E+01</td>
</tr>
</tbody>
</table>

Fig. 1. Composite cDNA microarray image together with its raw data parameters determined after performing step 6 and 7.
Devices, Inc.), Scanalyze, Dapple [13] or ImaGene (Biodiscovery, Inc.) use the procedure of manual template matching which involves defining the number of spots (along rows and columns), spot diameter and spot spacing. The defined template is overlaid on the microarray image and the previously mentioned parameters are adjusted in order to match the spots in the microarray image of interest. Automatic template adjustments are introduced by GenePix and QuantArray [14] but even so, if grid geometry deviation is increased, the method is not efficient. For each type of microarray technology different template definitions are necessary, thus the method is not fully automated.

In data-driven approach, image processing techniques are used to determine the grid alignment. First, horizontal and vertical projections vectors are computed by summing up the pixels intensities on microarray image rows and, respectively, columns [12]. Based on statistical analyses of 1D image projections, local extrema representing spots centres locations are than estimated [15]. Due to the profile irregularities, autocorrelation is used to determine spot spacing and spot dimensions [16]. This approach is efficient in spots centres locations computation, but on the other hand, due to profiles irregularities, additional powerful segmentation techniques are needed to detect spot dimensions and spot spacing.

For handling all the above mentioned issues, we propose an accurate low-complexity automated grid alignment method. It is to be mentioned that our novel shock filter based approach, besides addressing information also provides segmentation information, the next step in microarray image processing. The computational complexity of the proposed method is compared with the state of the art approaches. Results and advantages obtained by applying shock filters for microarray image addressing are pointed out in Section 2.4.

In image processing, shock filters generally serve as an edge enhancing algorithm. Aiming blurriness enhancement, Osher and Rudin proposed the first shock filter formulation in [17]. Based on a hyperbolic partial differential equation, the general one-dimensional (1D) shock filter model is described by Eq. (2), under the initial conditions $U(x, 0) = U(x)$ and with the operator $F$ fulfilling the following conditions: $F(0) = 0$ and $F(s) \times \text{sign}(s) \geq 0$.

$$\frac{\partial U}{\partial t} + F(U(x)) \frac{\partial U}{\partial x} = 0$$ (2)

In Eq. (2) $U(x)$ and $U(x,t)$ are denoting respectively, the first and the second order derivatives of the function $U$. By choosing $F(s) = \text{sign}(s)$, we obtain the classical shock filter equation:

$$U_t = - \text{sign}(U(x)) \frac{\partial U}{\partial x}$$ (3)

The following discrete scheme is used for 1D shock filter approximation [17]:

$$U^n_{i+1} = U^n_i - \Delta t \cdot [D^n U^n_i] \cdot \text{sign}(D^n U^n_i)$$ (4)

where:

$$D^n U^n_i = \frac{m(\Delta U^n_i, \Delta U^n_{i+1})}{h}$$

$$D^n U^n_{i+1} = \frac{(\Delta U^n_i - \Delta U^n_{i+1})}{h^2}$$ (5)

and $m(x,y)$ is the “minmod” function:

$$m(x, y) = [\text{sign}(x) + \text{sign}(y)] \cdot \min(|x|, |y|)$$ (6)

and

$$\Delta_{\pm} = \Delta(U_{i+1} \pm U_i)$$ (7)

are the forward and backward difference operators.

For microarray imaging addressing, we applied shock filters on the horizontal and the vertical image profiles, using the previously 1D discrete shock filter model, Eq. (4). The horizontal and vertical profile computation is described by (8) and (9), with $VP$ representing the vertical profile and $HP$ the horizontal one. We denote by $I(x, y)$ the grey scale microarray image, and, respectively with $X$ the height and with $Y$ the width of the image ($y = 0, 1, \ldots, Y - 1$ and $x = 0, 1, \ldots, X - 1$).

$$VP(x) = \frac{1}{Y} \sum_{y=0}^{Y-1} I(x, y)$$ (8)

$$HP(y) = \frac{1}{X} \sum_{x=0}^{X-1} I(x, y)$$ (9)

By applying shock filters, image profiles evolve as described by the Eq. (3), an example can be seen in Fig. 2a and b where a section of the original profile and its corresponding result after applying shock filters are described. A better view of the shock filter effect can be seen in Fig. 2c, where the thin continuous line represent the original image profile which evolves in the direction pointed by the arrows. The result is the thick continuous line, the dotted lines being intermediate steps in image profiles evolution. The main advantage is that shock filter creates strong discontinuities at the inflexion points of the profiles, thus, based on the resulted profiles described in Fig. 2c, grid alignment is performed as follows. Pairs of perpendicular lines can be drawn over the picture as shown in Fig. 2d and e, considering the inflexion points on both image profiles. Spot location is simply determined as the centre of the square defined by upper left corner $A(2l, 2j)$ and lower right corner $B(2l+1, 2j+1)$. Moreover, a region can be defined as being local background information around the spot (the area between the two squares in the single spot cropped image from Fig. 2d). Separating each spot from its local background is considered as information regarding segmentation.

2.3. Microarray image segmentation

The outcome of the previously described image processing technique is an automated approximation of spot locations, defined as a rectangular area enclosing one spot. The next step is to identify pixels that belong to the microarray spot and pixels that represent background. Thus, taking into account the grid alignment, a template which defines areas with foreground and background pixels is overlaid on the microarray spot. Different software platforms use different strategies to define the template Fig. 3a [18]; Scanalyze uses all the pixels that are not within the spot mask but within the dotted square centered as a region for background estimation, ArrayVision considers an area like the one between the two circles as background and GenePix estimates the background on the basis of the pixel values in the diamond areas. In case of our automated image processing chain, shock filter based approach for grid alignment introduces segmentation information. Thus, background area is delimited by two squares as shown in the single spot image from Fig. 3b.

For the foreground information edge detection is applied inside the smaller square to separate foreground from spot pixels. 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 suitable [19]. The algorithm added to our automated processing chain for image segmentation is Canny filter [20] due to its optimal results in terms of accuracy and computational complexity.

2.4. Results and discussion

Up to this point, image processing techniques were presented in order to realize a robust detection of microarray image spots (features). A novel shock filter based approach was introduced in microarray image processing chain, which provides automated grid
Fig. 2. Automatic microarray grid alignment using shock filters: (a) section of the microarray image horizontal profile, (b) resulted profile after applying shock filters, (c) profile evolution, (d) microarray spot addressing using shock filters on both horizontal and vertical profiles, and (e) automatic microarray image addressing on real size Agilent microarray image.
alignment and also delivers information regarding spot segmentation. In order to validate the proposed automated addressing method in terms of accuracy and efficiency, we compare our results with the ones drawn from GEO database and we estimate the computational complexity of our shock filter based approach compared with existing state of the art approaches.

### 2.4.1. Accuracy of spot location estimation

The data set used to evaluate our shock-filter based approach for localising microarray spots consists of 45 microarray images which contains over 1 million microarray spots. The 16-bit grey scale images of 22,575 microarray spots each are stored as TIFF files with a resolution of $6100 \times 2160$ pixels. The accuracy is estimated by comparing our obtained results with the ones available on Gene expression Omnibus. Thus, for each microarray image the determined locations for each microarray spot $i$ are defined by pairs $(X_i, Y_i)$ representing spot centre coordinates. The pairs $(X^C_i, Y^C_i)$ represent spot location expressed in pixels delivered by existing software platforms drawn from GEO data repository. Mean squared errors $MSE_X$ and $MSE_Y$ are computed between the two $(X_i, X^C_i)$ and $(Y_i, Y^C_i)$ spots coordinates. Moreover, the mean distance $d$ between spot centres drawn from GEO database and the ones obtained by our approach is estimated in case of each microarray image. In Table 1, results of the aforementioned comparison are listed considering a subset of 10 microarray images. Considering the whole dataset, each spot resides inside its determined grid cell, whereas the mean distance between the estimated spot position and the one drawn from the database is 1.08 pixels.

### 2.4.2. Computational complexity evaluation

Further on, current state of the art approaches for automatic grid alignment are summarised and their computational complexity is estimated (Table 2), considering $M$ and $N$ being the image dimensions. The most commonly used method for grid alignment is based on computing horizontal and vertical profiles and using autocorrelation to determine spot dimension and distance between spots [15,16,21]. Reduced complexity is achieved as shown in Table 2, but, on the other hand, major disadvantages are introduced. Spot locations are determined using profiles peaks and valleys, which in case of irregular pixel intensity distribution along a microarray spot are not accurate. Moreover, a fixed spot dimension and fixed spot spacing are estimated by autocorrelation, which fails in terms of accuracy in case of irregular image profiles and spots with different radii. Thus, more complex approaches for automatic grid alignment were proposed in the literature. In Refs. [22,23] a hill climbing algorithm is used for grid alignment with increased computational complexity. Morphological operators are used in [24] and for [25] automatic microarray image addressing, with a computational complexity of $O(2S,MN)$ where $S$ is the size in pixels $1k$ of the structural element for dilation and erosion.

### Table 1

Accuracy estimation for spot detection in case of our shock-filter based approach for automated image addressing applied on different microarray images (Image characteristics: pixel resolution $6100 \times 2160$, 16 bits/pixel, 22,575 spots).

<table>
<thead>
<tr>
<th>Experiment ID</th>
<th>Agilent scanner ID</th>
<th>MSE$_X$ position</th>
<th>MSE$_Y$ position</th>
<th>Mean Euclidean distance $d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>US45102867_1</td>
<td>11,472</td>
<td>0.148</td>
<td>0.602</td>
<td>1.081</td>
</tr>
<tr>
<td>US45102867_2</td>
<td>11,868</td>
<td>0.256</td>
<td>0.576</td>
<td>0.786</td>
</tr>
<tr>
<td>US45102867_3</td>
<td>11,978</td>
<td>0.450</td>
<td>0.442</td>
<td>0.832</td>
</tr>
<tr>
<td>GSM135598</td>
<td>11,978</td>
<td>0.295</td>
<td>0.998</td>
<td>1.086</td>
</tr>
<tr>
<td>GSM135599</td>
<td>11,978</td>
<td>0.129</td>
<td>0.982</td>
<td>0.822</td>
</tr>
<tr>
<td>GSM102718</td>
<td>11,978</td>
<td>0.342</td>
<td>0.517</td>
<td>0.822</td>
</tr>
<tr>
<td>GSM102721</td>
<td>11,978</td>
<td>0.411</td>
<td>0.730</td>
<td>0.927</td>
</tr>
<tr>
<td>GSM207313</td>
<td>11,521</td>
<td>0.253</td>
<td>0.902</td>
<td>0.935</td>
</tr>
<tr>
<td>GSM207316</td>
<td>11,521</td>
<td>0.372</td>
<td>0.979</td>
<td>1.013</td>
</tr>
<tr>
<td>GSM207320</td>
<td>11,521</td>
<td>0.595</td>
<td>0.906</td>
<td>1.073</td>
</tr>
</tbody>
</table>

Moreover, based on a selection of microarray spots, in [26] and [8] a SVM (support vector machine) approach is reported for automatic microarray grid division which overcomes the genetic algorithm presented in [27] with nearly one order of magnitude in terms of computational complexity. Parameter $k$ represents the number of selected microarray spots to train the SVM, in real case microarray

### Algorithm: Shock filter based grid alignment

**Input:** $M \times N$ size microarray image $I(i,j)$ point-wise pixel intensity

**Output:** microarray horizontal grid alignment $f_{HP}$ horizontal shock filtered profile

```
for j ← 1 to N do
    for i ← 1 to M do
        f_{HP}(i) = f_{HP}(i) + I(i,j)
    end
end

for iter ← 1 to α do
    for i ← 1 to M do
        HP_{\alpha}(i) = f_{HP}(i + 1) + f_{HP}(i - 1) - 2f_{HP}(i)
        Δ_{H}(i) = f_{HP}(i) + f_{HP}(i-1)
        Δ_{H}(i) = f_{HP}(i + 1) + f_{HP}(i)
        f_{HP}(i) = f_{HP}(i) - \text{sign}(HP_{\alpha}(i))
            \text{...} \cdot \text{min}(Δ_{H}(i), Δ_{H}(i)) \cdot \text{...}
            [\text{sign}(Δ_{H}(i)) + \text{sign}(Δ_{H}(i))]
    end
end
```

**Fig. 3.** Example of background and foreground separation methods.

**Fig. 4.** Computational steps performed by the shock filters for grid alignment.
images the order of $k$ being $10^2$. In [9] fully automatic microarray grid alignment is performed using an optimal multilevel threshold approach. The reported computational complexity is $O(t_\varepsilon N^2)$, where $t_\varepsilon$ denotes the threshold set size.

The number of computational steps performed by our proposed shock filter based approach for horizontal grid lines alignment is illustrated in Fig. 4. Considering the number of steps for vertical grid lines determination computed in a similar manner, the total computational cost is denoted by the upper bound function $f(M, N) = 2MN \cdot s + 6\alpha(M + N)s$ on the algorithm complexity, where $s$ represents one computational step. The computational complexity of the proposed grid alignment algorithm is $O(f(M, N)) = MN + \alpha(M + N)$, representing the order of growth for the computational cost. The $\alpha$ parameter represents the number of iteration necessary for profiles evolution to deliver accurate results. We empirically determined the optimal $\alpha$ value ($\alpha = 100$).

As presented in Table 2, the novel shock filter based approach brings up significantly reduced computational complexity compared to state of the art approaches, while similar results in terms of accuracy are achieved. Moreover, grid alignment is automatically performed while information regarding segmentation is delivered for next processing steps. The large size of microarray images is also taken into account while applying shock filters. Thus, in spite of their iterative nature, shock filters proved to be efficient since they are applied on image profiles.

### 2.4.3. Efficiency evaluation in the presence of artefacts

Fig. 5 illustrates the grid alignment performed by our approach in the presence of artefacts. More specifically, a set of microarray images (AT-20392-ch1, A201T-13036325-ch1, 1302-ch2-OD370, 1311-ch2-OD080) containing bright artefacts due to the microarray slide printing and scanning processes is analysed. In [9], failure to detect some spot regions due to the extremely contaminated

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(M + N)$</td>
<td>$O(M^2N^2)$</td>
<td>$O(25MN)$</td>
<td>$O(MN(M + k))$</td>
<td>$O(t_\varepsilon N^2)$</td>
<td>$O(MN + \alpha(M + N))$</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 5. Evaluation of our grid alignment approach in the presence of artefacts: (a) accurate gridding performed on AT-20392-ch1 compared with the approach presented in [OMTG]; (b)–(d) accurate gridding in the presence of bright artefacts.

Please cite this article in press as: Belean B, et al. FPGA based system for automatic cDNA microarray image processing. Comput Med Imaging Graph (2012), doi: 10.1016/j.compmedimag.2012.01.002
images is reported in case of the AT-20392-ch1 microarray image from the SMD dataset. For the same image, our approach delivers accurate results for the spots enclosed in the dotted squared area from Fig. 5a. Meanwhile, the aforementioned group of spots is not detected by the grid alignment method proposed in [9]. Despite the presence of artefacts and noise, for the images illustrated in Fig. 5b–d the proposed method achieves successful gridding even in the vicinity of the artefacts. It is to be mentioned that, the same type of artefacts were considered for accuracy evaluation as the ones referred in Refs. [8, 26].

To conclude, experimental results proved that our automated image processing techniques are accurately determining spots locations and characteristics, motivating the development of application specific architectures in order to eliminate user intervention and reduce computational costs.

3. FPGA based system for microarray image processing

FPGA technology uses pre-built logic blocks and programmable routing resources for configuration and for implementing custom hardware functionality. Its main benefits are the low cost, the short time to market and the ease of reconfiguration. Moreover, FPGA technology exploits spatial and temporal parallelism aiming algorithm parallelization for fast processing. All these advantages are used to implement application-specific architectures for our proposed automated image processing techniques described in Section 2.

Image scanning, spot detection and intensity extraction, steps 5–7 from a microarray experiment are susceptible to be integrated in a FPGA based system in order to reduce computational costs, to eliminate user intervention needed by the existing software platforms and to decrease the processing time. Fig. 6 describes the proposed FPGA based system for microarray image acquisition and processing. The proposed architecture is developed around an FPGA based Processing Unit and a Charged Coupled Device (CCD) image sensor. The CCD transforms the fluorescence levels (incoming photons) produced by the double-laser scanning device into electrons which are stored as electrical charge. The charge is amplified resulting in an analog output signal, which is digitized using an (Analog to Digital Converter) ADC and stored in the DDRAM memory as a microarray image.

The FPGA based Processing Unit implements the following functions: scanner control, CCD control, storage control through the Control Unit (CU), robust microarray image processing using FPGA based application specific architectures (described in the next section), and also the control for the two communication interfaces COM and COM 2. The first one, COM, uses a general purpose communication interface USB which allows the transfer of processed microarray images together with raw-data information describing microarray image characteristics for storage. The problems posed by acquisition and storage were discussed in [28, 29]; we refer to these publications for more details. The second one, COM 2, uses wireless communication specific for last generation hand-held devices in order to send the microarray image parameters, like the ones in Fig. 1, for high-order microarray image processing.

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.

3.1. FPGA based application specific architectures for microarray image processing

The hardware implementations of microarray image processing techniques make use of the FPGA features in order to evaluate the performances of our proposed system on target devices. 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. As a consequence, FPGA are efficient for processing microarray images as shown in [28].

Our FPGA based system on a chip including custom processing elements for microarray image processing techniques is described in Fig. 7. Each custom processing element from the proposed design is connected to the Fast Simple Link – FSL data bus as a co-processor for the soft-core MicroBlaze 100 MHz microprocessor. The FSL bus implements a point to point FIFO (First In First Out) based communication between processing units and microprocessor. A write operation to a FSL input FIFO is performed by MicroBlaze in a single clock cycle. A read operation transfers the content of a FSL bus to a general purpose register in 2 clock cycles. Our custom processing elements, using parallel computing, act as hardware accelerators.

![Fig. 5](image-url)
input data is sequentially written in the FSL input FIFO and the results are delivered through the FSL output FIFO. For example, in case of the logarithm based enhancement architecture, the FSL input FIFO is loaded with pixel intensities, while the computed results are sequentially delivered to the FSL output FIFO after a \( 3T_{clk} \) cycles delay. The MicroBlaze soft-core microprocessor is used as an interface between our processing elements and inputs/outputs. A Processor Local Bus IP block with cache system is used to store results in RAM memory.

The custom processing elements included in our system implement in a pipeline manner the independent processing steps specific to microarray image processing: image enhancement, grid alignment and segmentation. Thus, using spatial and temporal parallelism, the proposed hardware architectures realize a robust microarray image processing, aiming to speed up computation and to reduce power consumption compared to a general purpose microprocessor implementation. Custom processing elements corresponding to image enhancement, automatic grid alignment and segmentation are described in the following sections. Timing considerations and hardware resource usage are presented in Sections 3.2 and 3.3.

### 3.1. Image enhancement architecture

Spatial logarithm transformation is used to improve microarray image enhancement. The logic block from Fig. 8d computes the logarithm of the luminance function \( Y \) component for each pixel \( (x, y) \) of the microarray image. The hardware implementation of the logarithm transformation is based on linear approximation of the logarithm function and is efficiently designed according to algorithmic constraints (fixed point requirements). The logarithm function is calculated in an \( n \) number of points \( A_n(x, y) \) 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 use Eq. (10) which represents the equation of a line with slope \( m \) and passes trough the point \( A_n(x_0, y_0) \) from the initial \( A_0 \) points, where \( y_i \) and \( m \) represent the memory content at \( Y \) modulo \( n \) address.

\[
y_{log} = m(y_i - x_0) + y_0 (10)
\]

MSE = \[
\frac{1}{Y_{MAX}} \sum_y \left[ \ln(y) - \ln_{est}(y) \right]^2 = 1.807 \times 10^{-5} (11)
\]

In order to evaluate the log function estimation, mean square error was calculated for \( y \) values between 1 and \( Y_{MAX} = 2^{16} \) with a number of \( n = 2^{16} \) \( A_n \) points and the result is shown in Eq. (11). A full pipelined architecture was developed to maximize the processing throughput. This choice reduces the computational time for the logarithm unit to 1 pixel/clock cycle with an initial 3 clock cycles delay. Logarithm accuracy can be improved by increasing the number of \( A_n \) points for estimation which involves higher memory storage for the ROM memories.

### 3.1.2. Automatic grid alignment architecture

The custom processing element for grid alignment involves computing the horizontal and vertical image profiles of the logarithm transformed microarray image. Thus, the 16 bits \( Y_{log} \) intensity from the previous architecture is the current input. The \( \Sigma_X \) and \( \Sigma_Y \) RAM memories and the two adders (32 bits) are used as accumulators for computing horizontal and vertical profiles while the whole image is scanned only once. The architecture shown in Fig. 8a, is fully pipelined, each logarithm data received being summed with the corresponding raw and column data according to the address processing unit. Prefetch read and delayed write operations are realized on memories to support high throughput requirements. The stored data in the two memories RAM \( \Sigma_X \) and RAM \( \Sigma_Y \) is further on used in the next processing step, grid alignment, which aims to extract spot locations.

We consider the image profiles being stored in a RAM memory (Image profile RAM). The novel architecture proposed for shock filter computation is shown in Fig. 8c. The architectures divides the profiles in blocks having the dimension \( n \), the same as the dimension of Buffer_1 and Buffer_2 which work as shift registers. In order to fill up the Buffer_1, \( n \times CLK \) cycles are necessary since data is read from RAM. When the Buffer_1 is full, the Profile divider generates “load_ctrl” signal in order to perform a parallel load of Buffer_1 register into \( Out(i) \) register. While new \( n \) – values are loaded in Buffer_1, \( Out(i) \) and \( Out(i) \) registers implemented on to FPGA will use spatial parallelism to compute the following:

\[
\text{Loop } clk = \overline{T_1 \ldots \overline{n}}
\]

\[
\begin{align*}
\text{Out}(i) & = r_k^{(i-1)} + dt \cdot \text{sgn}(r_k^{(i-1)} - r_{k+1}^{(i-1)}) \cdot \text{min}(r_k^{(i-1)}, r_{k+1}^{(i-1)}) \quad (12) \\
& = r_k^{(i-1)} - r_{k+1}^{(i-1)} \cdot \text{sgn}(r_k^{(i-1)} - r_{k+1}^{(i-1)}) \quad (13)
\end{align*}
\]

End loopwhere \( n \) represents the number of clock-cycles for which the Out registers evolve as the Eqs. (12) and (13) and \( r_k = r_k - r_{k+1} \). Considering \( p \) and \( q \) the clock cycles needed for computing Eqs. (12) and (13) respectively, the number of iterations performed by the proposed application specific architecture is given by \( i = n/p + q \).

When the second series of \( n \) values from the image profile are loaded in Buffer_1, results of the shock filter computation are loaded in Buffer_2. Next step is loading the resulted profile in the Shock filter result RAM, which takes \( n \) clock cycles. It is to be mentioned the parameter \( n \) is chosen depending on the number of iterations \( i \) needed by shock filter. An empirical approach is used to compute the parameter \( i \). Accurate results are obtained with \( i = 100 \), which leads to \( n = 400 \), tacking into account that using spatial parallelism, \( p + q \) is less than 4 clock cycles.

### 3.1.3. Image segmentation architecture

This section presents a custom processing element which implements Canny edge detector aiming microarray image segmentation. Taking into account that spots location are provided by the grid alignment step, edge detection based segmentation is applied only on spot area. Applying Canny filter on each spot location determined by grid alignment involves the following steps: image smoothing, image gradient computation, non-maximum suppression to eliminate the pixels that represent false edges. The last mentioned image processing operations are based on image convolution, classified as a spatial filter. 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 \( N \times M \) 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, in our case a microarray spot (width \( \times \) height pixels), is stored into the FSL FIFO; \( M \times N \) pixels values are required to calculate one output pixel value. Memory bandwidth constraints make obtaining all these pixels each clock cycle impossible, so local caching is performed [30,31]. In this way, \( N - 1 \) rows are cached using a shift register which leads to the block diagram from Fig. 8b. Thus, instead of sliding a window across the image, the implementation feeds the image of width \( \times \) height dimension through the window. Corresponding pixels from the resulted image after convolution are delivered sequentially to the output after an initial delay of \( \Delta t = (M - 1) \text{width} + N \).

Using the previous approach for image convolution, logic blocks are implemented for smoothing the image, for computing the first order derivative and for non-maximum suppression. A
Table 3
Computational time for the proposed image processing techniques. Computational time was estimated for each microarray image processing techniques both on PC (Intel Dual core T2370 processor, 2GB RAM) and Virtex 5 platform (MicroBlaze processor, 256 MB RAM). Estimations were performed on Agilent US45102867,1 image (pixel resolution 6100 × 2160, 16 bits/pixel including 22,575 microarray spots).

<table>
<thead>
<tr>
<th>Gridding method</th>
<th>Processing platform</th>
<th>Processing time</th>
</tr>
</thead>
<tbody>
<tr>
<td>M²G</td>
<td>Athlon × 2–3.8 GHz, 3 GB RAM</td>
<td>10 s</td>
</tr>
<tr>
<td>Genetic algorithm</td>
<td>–</td>
<td>92 s</td>
</tr>
<tr>
<td>Proposed shock filter</td>
<td>Virtex 5, 100 MHz, 512 MB RAM</td>
<td>192 ms</td>
</tr>
<tr>
<td>Proposed shock filter</td>
<td>T2370-2 × 1.7 GHz, 2 GB RAM</td>
<td>313 ms</td>
</tr>
</tbody>
</table>

Table 4
HDL synthesis report.

<table>
<thead>
<tr>
<th>HDL synthesis report</th>
<th>RAM size</th>
<th>Processing time</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of multipliers</td>
<td>4 (16 × 16 bit)</td>
<td>2 (16 × 16 bit)</td>
</tr>
<tr>
<td>No. of adders</td>
<td>3 (16 bit)</td>
<td>2 (16 bit)</td>
</tr>
<tr>
<td>No. of single port RAMs</td>
<td>–</td>
<td>2 (4k × 32 bit)</td>
</tr>
<tr>
<td>No. of counters</td>
<td>2 (12 bit up)</td>
<td>1 (7 bit up)</td>
</tr>
<tr>
<td></td>
<td>2 (13 bit up)</td>
<td></td>
</tr>
</tbody>
</table>

3.2. Computation time considerations

The computational time needed for performing automatic grid alignment in case of the SVM approach presented in [26] is reported as 10 ms for a 450 × 450 pixel size microarray image block. The same type of microarray image was considered in order to evaluate the processing time of our proposed shock filter based approach for automatic grid alignment. The microarray image was processed using C code, while clock function was used for time measurement. Computational time is estimated also for each of the hardware architectures developed for automatic microarray image processing. In Table 3, results are presented for processing a 450 × 450 pixel microarray image block (16 bits/pixel representation), using both state of the art approaches and the proposed approach on different processing platforms (PC and Virtex 5 platform).

Fig. 9 illustrates the computational time for performing grid alignment in case of our proposed approach applied on different blocks of spots of the same microarray images US45102867,1, delivered by Agilent scanners. Computational time was measured on both PC (Intel Dual core T2370 processor, with a 1.73 GHz clock frequency, 2GB RAM) and Virtex5 platforms. Y axis represents the
processing time in milliseconds, while X axis refers to the dimension of each block of spots specified by the number of microarray spots enclosed. It can be seen that, in spite of higher performances provided by the PC, the application specific architectures added to a MicroBlaze system on Virtex5 FPGA bring up better results.

3.3. Hardware resource usage

The target device used for the implementations of the proposed hardware architecture is Xilinx xc5vlx110t FPGA found on Virtex5 platform. The architectures were completely described using VHDL and the VHDL code was behaviourally and post-place-and-route validated through simulations. HDL synthesis reports together with device utilization summary are detailed in Tables 4 and 5, respectively, for each microarray image processing step.

Besides spatial and temporal parallelism introduced by our implementations, future work aims designing a high throughput system which use multiple instances of the proposed application specific hardware architectures.

4. Conclusions

The paper proposes a novel shock filter based approach for automated grid alignment on microarray images, integrated in the classical flow of a microarray experiment. The obtained results, compared with the ones delivered by the existing software platforms for microarray image processing, proved to be reliable both in terms of automation and accuracy. Computational complexity proved to be significantly reduced compared to the state of the art approaches for automatic microarray image processing. Taking into account automation introduced by the proposed image processing techniques, an FPGA base system aiming to reduce both user intervention and computational costs is described. The system includes application specific hardware architectures for image enhancement, grid alignment and segmentation making use of spatial and temporal parallelism offered by FPGA technology. Measurements for the computational time were performed on the FPGA platform and on a personal computer. Experimental results bring up a lower processing time on the FPGA platform, in spite of higher resources available on PC. Moreover, the SVM approach proposed in [26] requires a processing time with more than one order of magnitude larger than our reduced complexity approach for automatic microarray image processing.

To conclude, the experimental results proved that our shock filter based approach for automatic grid alignment together with the FPGA based hardware architectures are an efficient solution for fast and automated microarray image processing, overcoming the disadvantages of existing software platforms in case of a large number of microarray analyses are needed.

Acknowledgements

The authors would like to thank the anonymous reviewers for their constructive and insightful comments. This work was funded by the Romanian Research Council of Higher Education through PNII IDEI program, contract no. 332/2007.

References


Bogdan Belean was born in Targu-Mures, Romania, on February 7, 1983. He received his B.E. and Ph.D degree in electronics and communication engineering from the Technical University of Cluj-Napoca, Romania in 2006 and 2010, respectively. He is currently a scientific researcher within the National Institute for Research and Development of Isotopic and Molecular Technologies, department of Mass Spectrometry, Cluj-Napoca. His research interests include signal and image processing, bioinformatics and application specific hardware architectures for parallel computing.

Monica Borda received the Ph.D degree in electronics and telecommunications from “Politehnica” University of Bucharest, Romania, in 1987. She has held faculty positions at the Technical University of Cluj-Napoca, Romania, where she is the Director of the Data Processing and Security Research Center. She has conducted research in coding theory, nonlinear signal and image processing, image watermarking, genomic signal processing and computer vision having authored and coauthored more than 150 research papers in referred national and international journals and conference proceedings. Her research interests are in the areas of information theory and coding, signal, image and genomic signal processing.

Bertrand Le Gal was born in 1979, in Lorient France. He received his Ph.D degree in information and engineering sciences and technologies from the Université de Bretagne Sud, Lorient, France, in 2005 and the DEA (MS Degree) in Electronics in 2002. He is currently an Associate Professor in the IMS Laboratory, ENSEIRB Engineering School, Talence, France. His research focuses on system design, application specific processors, high-level synthesis, SoCs design methodologies and security issues in embedded devices such as Virtual Component Protection (VCP).

Romulus Terebes was born in Livada, Romania, on December 13, 1969. He received the B.S. degree in electronics and telecommunications from the Technical University of Cluj-Napoca (TUC-N), Romania in 1994, and the Ph.D. degree from Universite Bordeaux 1, France and the TUC-N (co-advised Ph.D. thesis) in 2004. The subject of his Ph.D. thesis dealt with partial derivatives equations based image processing techniques. He is currently an Associate Professor in the Department of Communications at the Faculty of Electronics, Telecommunications and Information Technology, TUC-N. His research interests are in the area of image processing and computer vision.

Please cite this article in press as: Belean B, et al. FPGA based system for automatic cDNA microarray image processing. Comput Med Imaging Graph (2012), doi:10.1016/j.compmedimag.2012.01.002