FPGA IMPLEMENTATION OF CMF FOR EMBEDDED PALM BIOMETRIC SYSTEM

Mihails Pudzs, Rihards Fuksis, Rinalds Ruskuls, Davis Barkans, Teodors Eglitis, Modris Greitans

Institute of Electronics and Computer Science
14 Dzerbenes Str., Riga, LV1006, Latvia

ABSTRACT

Most multimodal palm biometric systems are still PC-based, because of the complicated image processing algorithms necessary to process biometric data. In this paper we demonstrate how to implement Complex Matched Filter in FPGA based systems for real-time palmprint and palm vein image processing. CMF approach is based on matched filtering with rotated line extraction kernels, however, CMF requires less computational resources and it obtains additional angular information about the extracted biometric features. This information is valuable in feature description and recognition process. Topics that are covered in this paper include optimization of used multipliers and real-time data processing without use of external RAM resources.

***Index Terms*—** FPGA, biometric system, linear filtering, complex matched filtering, resource optimization.

1. INTRODUCTION

Development of embedded biometric systems that are able to process data in real-time (at the same speed as the data is acquired from the source) is a challenging task due to a complex image processing that is necessary to extract and compare biometric data. In this paper we focus on palm biometric image processing and our primary concern is the extraction of palmprint and palm vein patterns. However, introduced approach can be also used to process other images that contain line-like objects.

Biometric systems are supposed to perform tasks like image acquisition, processing, feature extraction and comparison in as less time as possible to minimize the authorization time. Each of the performed actions requires some fraction of total time, however the most costly is the image processing task that includes feature extraction. To extract palm biometric information from images, one of the techniques that might be used is *matched filtering* (MF) [1]. This method requires to convolve input image with kernel, rotated in different angles to extract features that appear in arbitrary orientations. However, each of the convolution operations slows down the overall performance of a system. To obtain speed or simplicity over the MF approach, the *complex matched filtering* (CMF) approach was introduced by Greitans et.al. [2]. CMF is able to extract line-like objects from the images by using one complex kernel, which can be implemented by performing two convolution operations. Instead of MF where the result is a matrix of scalar values, CMF obtains vectors in each pixel of the output image. These vectors represent the angle in which the corresponding line-like feature is found. This additional information is useful for further feature extraction and comparison.

In this paper we present how CMF can be implemented in FPGA for line-like object extraction in real-time. The proposed filter is implemented in Altera DE2-115 prototyping system without the need for CPU and OS.

Since CMF has complex kernel, it requires twice as many multipliers than a simple 2D linear filter with one mask of the same size. Therefore, multiplier reduction methods are presented in this work. However, the usage of such methods degrades the quality of filtered images. To evaluate the performance of implemented filter in terms of used FPGA resources (like registers and multipliers) and image processing quality, the experiments are conducted where images that are filtered using FPGA system are compared to images, filtered using MATLAB on PC.

2. RELATED WORK

This section briefly introduces CMF approach. Also convolution technique, which is based on the internal FPGA resources is discussed in more details [3].

2.1. Complex Matched Filtering

CMF is a filter from generalized complex matched filter (GCMF) bank that can be used for line-like object extraction [4].

CMF kernel can be described in polar coordinates using equation (1).

$$\underline{M}(\rho,\Theta) = e^{j2\Theta} r(\rho) \qquad (1)$$

The right part of (1) consists of two multipliers:

- $e^{j2\Theta}$ is the *angular component* $(0-360°)$ of the complex

This research is partially supported by ERSF project No.2010/0285/2DP/2.1.1.2.0/10/APIA/VIA/098

mask,
- $r(\rho)$ is the *radial component* of the complex mask.

Radial component varies only with distance from the center of the mask and defines magnitude of the angular component. The radial component is chosen depending on the scale of features to extract - r_0.

$$r(\rho) = \left[e(-\frac{(\rho - r_0)^2}{\sigma^2}) + e(-\frac{(\rho + r_0)^2}{\sigma^2}) \right] \quad (2)$$

As mask size depends on the radial component, optimal size for the mask is at least $2r_0$+6σ - where mask contains most of the energy (>99%).

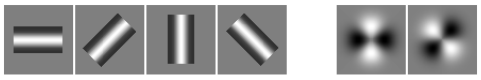

Fig. 1. MF (left) and CMF (right) masks comparison

CMF is angle invariant filter with one complex kernel, therefore, to detect line-like objects with arbitrary orientation only two convolution operations are needed. In contrast, MF requires at least four convolution operations for the same task, Fig. 1.

After the input image is convolved with CMF kernel complex values are obtained in each pixel. The angle of these values should be decreased by half to obtain matching intensity vectors. The absolute value of matching intensity vectors characterize the intensity of found objects and the angle - their direction. In this paper we evaluate the quality of processed images in terms of PSNR for absolute values of matching intensity vectors.

2.2. Linear 2D filtering with sliding window architecture

Real-time processing method allows to acquire output pixels with the same speed as the input pixel rate, introducing only slight delay between input and output events. This approach requires for some rows of the image to be stored and for part of the image rows to be accessible simultaneously, which can conveniently be implemented in FPGA without using any of the external RAM.

To save information the internal memory of FPGA such as RAM bits and registers is used. Size of the memory depends on resolution of a filter mask. It is assumed that the filter mask is square, so it has one parameter - horizontal/vertical size M_{size}. In our case, number of columns in one row is R_{size} = 640, therefore the number of pixels that need to be stored is:

$$M_{size}^2 + (M_{size} - 1) \times (R_{size} - M_{size}) \quad (3)$$

The memory consists of *active* and *passive* parts. Active and passive parts are united in pairs and each pair stores one row of a picture. Each active part consists of registers (DFF - **D**ata **F**lip-**F**lop) that are connected in series to form a serial in-parallel out shift register. Parallel outputs allow to access the stored data simultaneously and therefore are used to obtain image fragment and multiply it with the filter mask. Length of the active part is equal to length of the mask M_{size}. Each passive part is implemented using RAM bits and operates like serial in-serial out shift register. Passive parts are used just to store pixels. The length of each passive part can be calculated as follows:

$$P_{size} = R_{size} - M_{size} \quad (4)$$

Figure 2 shows an example of how the memory elements must be organized to carry out rapid filtration for 2D filter with kernel size of 3x3 and image width R_{size} = 6. A red dashed square indicates pixels that are stored in active parts where image and mask pixels are multiplied. A blue part indicates pixels that are stored in RAM bits. The active part of the filter forms a sliding window by carrying pixels from square image region of size 3x3, numbered as 0-2, 6-8, 12-14. Each new input pixel shifts the pixels stored in the filter memory, which is equivalent to movement of the sliding window (shown in Fig. 2 lower-right corner).

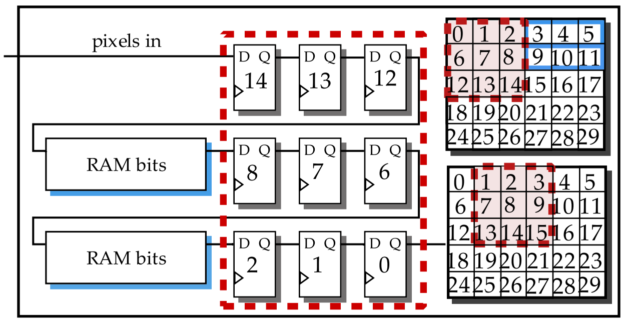

Fig. 2. Sliding window emulation using filter active part

3. PROPOSED APPROACH

The proposed filter (Fig. 3) consists of several blocks that include modified active parts for image pixel multiplication with mask coefficients and passive parts for storing pixels. We use tree adder to sum multiplied pixel values and acquire convolution values for real and imaginary parts of the filter kernel. Further steps include calculation of the absolute value by method from [5] and calculation of the matching intensity vectors (features). For last mentioned operation the angle of result must be decreased by half [2], which is accomplished by *CORDIC* [6] algorithm.

3.1. Active part of the proposed filter

The task of the active part includes storage of the input pixel values and mask coefficients. Since CMF has a complex

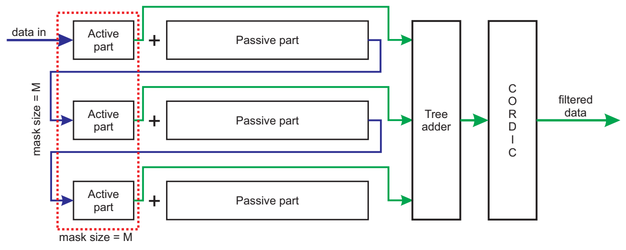

Fig. 3. 2D filter block diagram

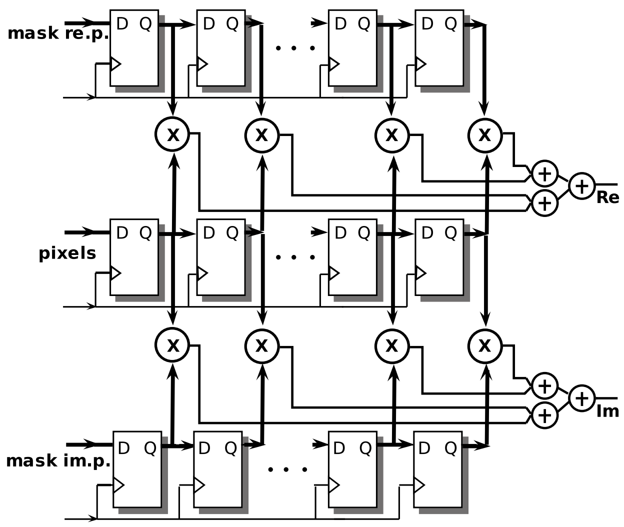

Fig. 4. Modified active part - coefficients of real and imaginary parts of the mask are used for filtering

mask, twice as many mask coefficients must be stored in the active part. To implement the imaginary part of a kernel one more row of DFFs along with multipliers and tree adder is added to each active part. The modified active part of the filter is shown in Fig. 4. This kind of structure uses twice as many multipliers than a typical linear 2D filter. Each of the filter parts (real and imaginary) works like a separate linear 2D filter and acquires its independent filtered value. To calculate absolute value of the filtered pixel square root computation circuit must be implemented. Quartus II FPGA software has inner square root function, but it is too slow for real time signal processing with chosen pixel clock frequency. Therefore, we have implemented simple, but fast method from [5]. To acquire matching intensity vector, CMF approach requires to reduce the angle of filtered pixels by a fraction of two, for this purpose two CORDIC units are implemented [6]. First transforms the data from Cartesian to polar coordinates and second - back to Cartesian coordinates. Angle reduction by half is performed in polar coordinates.

3.2. Optimizing the Multipliers

We use CMF with kernel size of 15×15 in our system, therefore, such filter will be analyzed and optimized in the subsequent sections. To implement CMF with mask of size 15×15 pixels 450 multipliers are needed.

Multipliers are essential to implement CMF, therefore this subsection presents the technique of multiplier usage optimization. CMF has two parts of the kernel - real and imaginary. Therefore, this kind of filter structure uses twice as many multipliers, but we can reduce that amount by analyzing coefficients of filter masks. Iff there are equal values at the same pixel position for both parts of the mask, then we can eliminate a multiplier from one part and use the multiplication result from the other part, Fig. 5 left. The similarity is further tested with a roundoff value of $\pm\Delta$, expressed for 8-bit signed mask with maximized dynamic range.

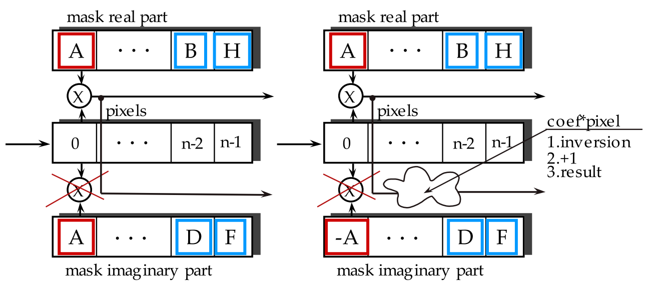

Fig. 5. Equal and inverse coefficient optimization

Second approach to reduce multipliers exploits inverse coefficients of the filter masks. This time we compare coefficients at the same pixel position for both parts of the filter mask and search for opposite values. Iff such values are found (again, with roundoff $\pm\Delta$ for absolute values), then one multiplier is replaced by inversion (Fig. 5 right).

When image pixel must be multiplied with coefficient 0 or 2^N (with roundoff of $\pm\Delta$), then multiplication might be either avoided, or replaced by a bit shift.

The automated analysis of mask coefficients is performed in order to apply these optimizations. Each time when mask parameters, like size or allowed round-off interval, are changed, the analysis of its coefficients must be repeated and the implemented filter structure recompiled.

For illustration purposes only real part of the filter kernel is shown in Fig. 6. Kernel size is 15×15. Red dots represent mask positions, where both mask pixels are zero and do not require a multiplier. Blue dots represent positions, where one of the mask pixels has a zero value and, therefore, does not need a multiplier. Blue circles represent equal valued pixels, where at least one multiplier might be ommited. Cyan circles represent pixels with opposite values. Green crosses represent pixels values that can be represented as 2^N. Multiplying with 2^N is a simple bit shift operation that does not require a

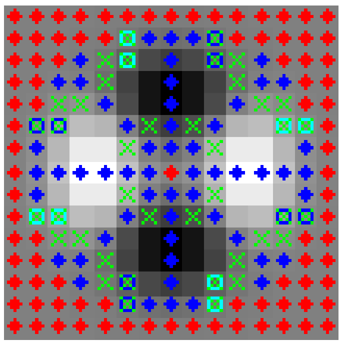

Fig. 6. Filter kernel with marks that represent places, where embedded multipliers can be saved

multiplier. For some pixels multiple optimizations are applied at the same time.

4. IMPLEMENTATION

As mentioned above, we use CMF in a multimodal palm biometric system to acquire feature vectors that describe person's palm print and palm veins. This system is implemented in *DE2-115* FPGA development board from *Terasic*. To perform tests and evaluate implemented filter several peripheral devices, such as image capturing device, memory unit (used only for storage of processed images), monitor, USB controller and a smart card interface (used for match-on-card comparison), are connected to our FPGA system. Fig. 7 shows architecture of this system. To ease the debugging process, system is controlled from a PC through USB interface. Following control commands are supported:
$\rightarrow$ acquire filtered and original images on PC,
$\rightarrow$ load mask coefficients from PC,
$\rightarrow$ control LEDs and shutter mechanism and reprogram image sensor to acquire images in different light spectrums.

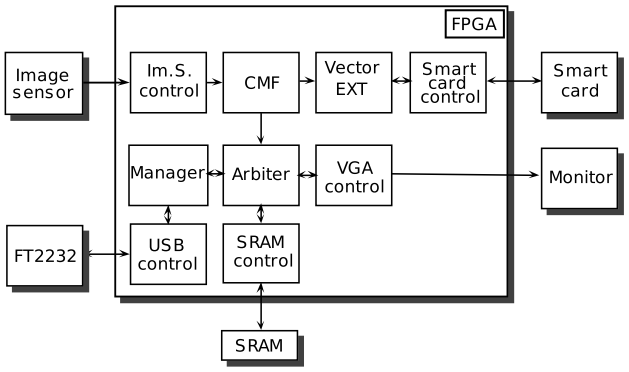

Fig. 7. Full system with external devices and FPGA part

To provide input data, we have created an image acquisition board, which consists of:
1) *APTINA MT9V024IA7XTM* image sensor,
2) infrared (850 nm) and white light spectrum LED's for palm illumination (to acquire palm vein images in infrared spectrum, and palmprint images in visible light spectrum).

Image sensor control module acquires data from the image sensor and streams pixels to CMF unit. Filtered image is written to SRAM, so, if needed, it can be displayed on the monitor using built-in VGA controller. To eliminate conflicts between reading and writing operations from/to SRAM, Arbiter module is used. Communication between PC and FPGA is provided by USB control module for FT2232 IC. Manager module controls the behavior of the system from a PC: it allows or prohibits to display the image, change mask coefficients and reads the images, stored in SRAM. Vector extraction module selects only certain amount of most intensive vectors from the filtered image. Smart card control module enables the communication between smart card and FPGA. It is used to send the biometric data to smart card for comparison and also to receive the authentication result. These modules are used in our biometric system, but are not relevant to the subject of this paper and will not be discussed further.

The main clock of the system is 125 MHz, and it is limited by the maximum operating frequency of a SRAM, which is used to store acquired (filtered and original) image data. The pixel clock of the image sensor is 25 MHz. Depending on the configuration, image sensor can acquire one image or a stream of images (30 fps). To synchronize all of the system modules the enable signal is used. CMF block works with the same clock frequency as the image sensor (25 MHz). There are no other memory between these blocks, filtering is performed in real-time.

5. EXPERIMENTAL SETUP

During the experiment various objects were shown to the image sensor to provide the graphical input. The experimental procedure consisted of several steps:

- choosing the roundoff interval $\pm\Delta$ (defines precision of the filter) and mask parameters (σ and r_0),
- generating the ideal mask on PC using Double precision numbers,
- applying the optimizations by changing mask's coefficients within the chosen roundoff interval (note: algorithm that finds mask coefficients minimizes the mask error) and programming FPGA with these coefficients,
- filtering 10 acquired images using CMF and comparing to the same images, filtered on PC using Matlab .

The roundoff interval Δ was sweeped between 0 (perfect reprezentation of CMF mask) and 28 (maximum distortion of the filter mask when no multipliers are used) to test its influence on the filter performance. Three different parameters were estimated for each of the Δ value:

1. mask error in terms of mask error L^2 norm divided to ideal mask L^2 norm;
2. amount of used multipliers (also equals to amount of used registers to store mask coefficients);
3. image filtration error in terms of peak signal-to-noise ratio (PSNR) for absolute values of matching intensity vectors.

6. RESULTS AND CONCLUSIONS

Experimental results are shown in Fig. 8.

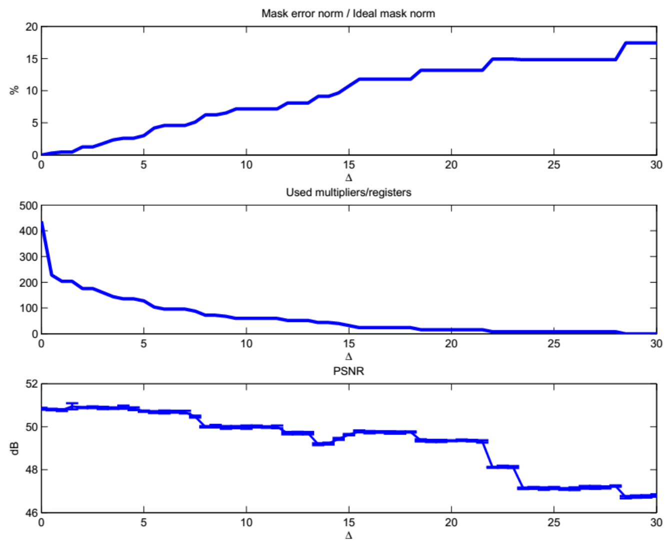

Fig. 8. Experimental results

As expected, there is a tradeoff between filter precision and the amount of used resources. By using automated mask coefficient roundoff procedure the amount of used multipliers might be decreased to zero by introducing only 17% of mask error. It must be noted that mask error is not linearly correlated with PSNR error because of the non-linear operations, applied during last stage of CMF (absolute value calculation, angle decrement by half).

This paper introduces the implementation of Complex Matched Filter in FPGA for an embedded multimodal palm biometric system. The implemented FPGA CMF is capable of real-time (30 fps) video/image stream processing and provides an acceptable palm vein and palmprint extraction quality, compared to the results of CMF, performed on PC using Matlab.

The implemented filter can also be used to extract line-like objects from non-biometric images.

7. REFERENCES

[1] Subhasis Chaudhuri, Shankar Chatterjee, Norman Katz, Mark Nelson, and Michael Goldbaum, "Detection of blood vessels in retinal images using two-dimensional matched filters," in *IEEE transactions on medical imaging, Vol.8, Issue:3*. 1989, pp. 263–269, IEEE.

[2] Modris Greitans, Mihails Pudzs, and Rihards Fuksis, "Object analysis in images using complex 2d matched filters," in *EUROCON 2009: Proceedings of IEEE Region 8 conference*. 2009, pp. 1392–1397, IEEE.

[3] M. Schaeferling and G. Kiefer, "Object recognition on a chip: A complete surf-based system on a single fpga," in *Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on*, 30 2011-Dec. 2, pp. 49–54.

[4] Mihails Pudzs, Modris Greitans, and Rihards Fuksis, "Generalized complex 2d matched filtering for local regular line-like feature detection," in *19th European Signal Processing Conference (EUSIPCO 2011)*. 2011, pp. 41–45, EURASIP.

[5] Pong P. Chu, *RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability*, Wiley-IEEE Press, 2006.

[6] Ray Andraka, "A survey of cordic algorithms for fpga based computers," in *Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays*, New York, NY, USA, 1998, FPGA '98, pp. 191–200, ACM.