IMECO: A RECONFIGURABLE FPGA-BASED IMAGE ENHANCEMENT CO-PROCESSOR FRAMEWORK

Z.Salcic, J.Sivaswamy
University of Auckland, Department of Electrical and Electronic Engineering
Private Bag 92019, Auckland, New Zealand
{z.salcic, j.sivaswamy}@auckland.ac.nz

ABSTRACT

This paper presents a way to improve the computational speed of image contrast enhancement using low-cost FPGA-based hardware primarily targeted to X-ray images. The enhancement method considered here consists of filtering via the high boost filter (HBF), followed by histogram modification using histogram equalisation (HE). An image enhancement co-processor (IMECO) concept is proposed that enables efficient hardware implementation of enhancement procedures and hardware/software co-design to achieve high-performance, low-cost solutions. The co-processor runs on an FPGA prototyping ISA-bus board. It consists of two hardware functional units that implement HBF and HE and can be downloaded onto the board sequentially or reside on the board at the same time. These units represent an embryo of virtual hardware units that form a library of image enhancement algorithms. In trials with chest X-ray images performance improvement over software-only implementations was more than two orders of magnitude, thus providing real-time or near real-time enhancement.

1. INTRODUCTION

Contrast enhancement is a digital image processing methodology which has a wide range of applications in medical and non-medical areas. Various techniques have been developed for contrast enhancement to suit different types of images. These techniques use a transformation operation applied either globally, or adaptively, to enhance the contrast of given images [1] [7]. Global techniques are computationally simple and are suited to images with poor global contrast while adaptive techniques are more computationally expensive and are better suited to natural images with varying local contrast. In application areas such as radiology, the computational cost is an important factor affecting the efficiency of medical diagnosis.

In this paper, we present a way to improve the computational speed of image contrast enhancement primarily targeted to X-ray images. In particular, we consider an enhancement algorithm that consists of filtering followed by histogram modification. Filtering is done via the HBF which is based on unsharp masking, and the histogram modification is based HE. Existing approaches to speeding up computations have primarily focussed on the adaptive version of histogram equalisation (AHE). These use linear interpolation to reduce computations [4], or employ several precomputed mapping curves which have to be manually selected [2]. Hardware solutions have also been proposed for the speed up problem in AHE. However, they are expensive, calling for MIMD parallel machines [3] or other specially designed parallel machines [5].

We have investigated the use of a low cost FPGA-based hardware that is simple to design for implementing both HBF and HE. (We present the global version of HE for a start, since extensions to AHE are straightforward.) FPGAs have become one of the prevailing technologies for fast prototyping and implementation of digital systems [6]. Being dynamically reconfigurable, FPGAs provide additional interesting features to hardware implementation of complex algorithms with performance exceeding both general-purpose and digital signal processor implementations. Using FPGAs we propose an image enhancement co-processor, IMECO, that enables efficient hardware implementation of enhancement procedures and hardware/software co-design. We present the IMECO framework in Section 2, and the HBF and HE implementation using IMECO, in the following sections.

2. FPGA-IMPLEMENTED IMECO FRAMEWORK

The IMECO framework consists of a standard PC and a general purpose FPGA prototyping ISA-bus board. The PC provides storage resources, programming facilities, flexibility, user-friendly interface (under the Windows environment), and controls the operation of IMECO. The FPGA board is used as a rapid prototyping system (RAPROS) for implementing and executing hardware implemented enhancement algorithms. IMECO is intended to contain a library of hardware implemented algorithms (configurations) which can be combined using software, into complex algorithms to suit specific needs. These hardware implemented algorithms can be viewed as functional units which appear as co-processors to the PC. At present, there are two units that implement HBF and (global) HE which can be downloaded onto the RAPROS sequentially or reside on the board at the same time. FPGAs on the RAPROS are used on a time-multiplexed basis to implement different functional units. Thus IMECO offers two key features, namely, custom-configurability and dynamic reconfigurability. The global organisation of the FPGA board used in our approach is illustrated in Figure 1.

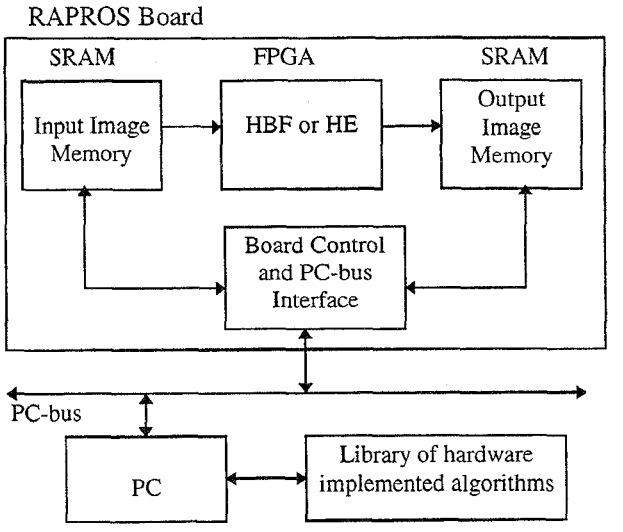


Figure 1. IMECO Framework

Prototyping resources in the RAPROS consist of:

1. Four FPGA (Altera's FLEX8282) prototyping devices.
2. Four 32-kB static (S)RAM chips.
3. A number of interconnects that can be used to form user bus structures, or other type of interconnections between devices.
4. Board control unit with PC ISA-bus interface.

With IMECO, complex image enhancement algorithms adapted to specific goals of target application can be implemented by combining basic algorithms such as HBF, HE, etc. These functions can be called from any programming language. Developed software support enables downloading of the desired hardware version of the algorithm to the RAPROS and control of its operation from a program. Different versions of the algorithm represent different hardware designs. Given an application, the user can select the implementation of computational units that suit best its requirement. A selected configuration, represented by a file on the PC host, is downloaded to the board. A program is used to store the input image to source SRAM locations. Then, it activates the functional units used in the enhancement algorithm (HBF & HE in our case), which in turn produces the output image in destination SRAM chips. The output image is available either to other hardware units or the program for further processing.

Program control is only needed to control DMA transfers of the original and the final images to/from SRAM chips on the board and for the change of hardware configuration of the FPGAs. A user-friendly interface provides easy selection of the configurations that will be used in the algorithm, and subsequently loaded by a configuration loader.

3. HIGH-BOOST FILTER IMPLEMENTATION

The task of the HBF is to calculate for each pixel in the filtered image $P_{HB}(x, y)$, $(x, y = 0, 1,\ldots, N-1)$

$$P_{HB}(x,y) = \alpha P_{IN}(x,y) - P_{LP}(x,y) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

where $P_{IN}(x, y)$ is the input image of size NxN, $P_{LP}(x, y)$ is the low-pass filtered (LPF) image, and α is a constant that can take different values (>1). The LPF is defined as

$$P_{LP}(x,y) = \frac{1}{n^2} \sum_{(x,y)\in M} P_{IN}(x,y)$$

where $(x,y) \in M$ denotes pixels within a square (filter) mask M of size n x n. Therefore, the LPF operation is an averaging operation over a local neighbourhood M. This operation, depends on the size of the mask and requires n^2 additions and one division operation. Finding the HBF version of an input image is thus a process in which each pixel of the input image is processed in the same way: it becomes the centre pixel of a square window of size equal to the mask, which moves (or slides) along the whole input image in an ordered way. A straightforward solution is to slide it along either rows or columns of the image, from top to down or left to right. The number of operations that have to be performed on an image is proportional to the size of image (N^2) and the size of the window (n^2).

In our implementation, we have reduced the number of memory read operations by recognising and exploiting the fact that the only change when the sliding window moves to a new row is the addition of a new row. Therefore, only the sum of pixels of the new (leading) row is calculated. All these pixels occupy adjacent memory locations. The new mean window value, which is actually the LPF value of the centre pixel, is calculated by subtracting the value of the row dropped from the sliding window (last trailing row) and adding the value of the new leading row. This is illustrated below for the case of a 5x5 ($n^2 = 25$) sliding window:

$$P_{LP}(x,y) = P_{LP}(x,y-1) + S_P$$

$$S_P = \frac{1}{25}[\text{Lead}(x,y+2) - \text{Trail}(x,y-3)]$$

Here Lead and Trail represent the partial sums of pixels of the new leading and previous trailing row of the sliding window. The data path in the HBF unit is shown in Figure 2. The sums of sliding window rows are calculated and stored in a FIFO structure. The FIFO always contains sums of five current rows of the sliding window. When a new leading row is encountered, its sum is calculated in the accumulator. Next, the sum of the trailing row is subtracted and the sum of the new leading row is added, to the existing sum of all rows belonging to the sliding window. This sum is divided by n^2 to obtain the LPF pixel value. Finally, the high boost-filtered pixel value is calculated using equation (1).

The total number of operations in our approach is proportional to the size of the image (N^2) and the width (n) of the applied mask. The whole algorithm is implemented in

FPGA hardware and employs parallelism. At the beginning of each new column, contents of the FIFO is cleared and partial sums of the rows are calculated to be stored in the FIFO as the window slides downwards. The details of the algorithm, including the processing of boundary pixels, are not discussed in this paper.

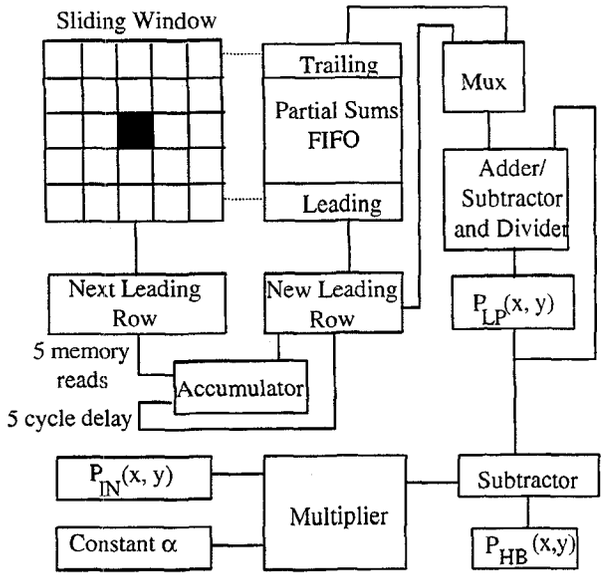


Figure 2. Data path for the HBF functional unit

4. HISTOGRAM EQUALISATION FUNCTIONAL UNIT

The HE functional unit performs global histogram equalisation of the entire image. The global HE algorithm can be described as follows:

1. Compute histogram $H_A(k)$ of given image A of size NxN, with A being the result of HBF operation, as

$$H_A(k) = n_k$$

where, n_k is the total number of pixels in the image at the kth gray level, with k = 0,1,...L-1.

2. Compute the equalisation value s_k for each gray level k as

$$s_k = \text{Int}[\frac{L}{N^2} \sum_{j=0}^{L-1} H_A(j)]$$

where 'Int' is an integer part of the calculated number.

3. Equalise and compute new image pixels $P_{HE}(x,y)$ as

If A(x,y) = k, then $P_{HE}(x,y) = s_k$ for every x and y.

The data path for the HE functional unit is presented in Figure 3. The unit finds the input image in the memory block to which it was stored by HBF functional unit. Then, it reads all pixel values and accordingly increments the histogram value of the corresponding gray level. The histogram values are stored in a separate memory called Histogram Memory which was added to the existing FPGA board. The gray level of the pixel fetched from image memory represents the memory address of the histogram value that has to be incremented. An incrementer is used to do this operation and return the new histogram value to the corresponding histogram

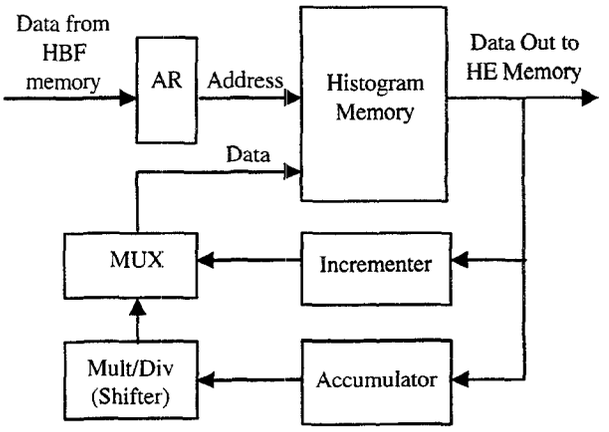


Figure 3. Data path for the HE functional unit

memory location. The second step of the algorithm takes place when all pixels from the input image have been processed. In this step, equalisation values for each gray level are calculated by reading the values from histogram memory and accumulating them to the previously found values. Only one pass of the histogram memory read operations and memory write operations is needed to calculate equalisation values which are now stored in histogram memory. This part of the algorithm is performed by an accumulator and an arithmetic block that carries out multiplication of the accumulated value by $\frac{L}{N^2}$ and rounding to the integer part. If we consider the expression $\frac{L}{N^2}$, we can see that usually we have $L = 2^b$, where b is the number of bits which represent gray levels, and $N^2 = 2^{2w}$, where $2w > b$. Therefore, the $\frac{L}{N^2}$ will produce result in the form 2^{-q}, where q is an integer and $q > 0$. This further means that the resulting accumulated sum must be divided by 2^q reducing the multiplication and division operation to a simple shift right (by q bits) operation. The final result is stored in HE memory block which is essentially the block in which the image was stored (input and output memory for each algorithm swap their roles). The number of operations involved in the HE computation is now proportional to N^2.

4. PERFORMANCE ANALYSIS

A full implementation of the HBF and HE functional units have been performed using Altera's FLEX8282 FPGAs. Several versions of each of the designs are stored in a library of configuration files and enable the end user to structure the algorithm by selecting appropriate files. Because of the small capacity of SRAM chips available on the board, the maximum size of the image which can be processed is

256x256. Larger images need to be partitioned into blocks of 256x256, and processed block by block under global software control. The maximum frequency at which functional units can operate is limited by the system clock on the board, to 10 MHz of the PC ISA-bus clock. The circuit simulation has shown that the maximum frequency can be increased to 30 MHz using FLEX8000 devices, effectively enabling tripled performance without any architectural or design change. The execution times for processing 256x256 image is presented in Table 1. Equivalent software implementations are in the order of several minutes, using much faster processors.

Table 1. Performance figures for a 256 x 256 X-ray image

Operation	**Time (sec) 10 MHz clock**	**Time (sec) 30 MHz clock**
HBF Straightforward hardware implementation	1.18	0.39
HBF (simplified arithmetic)	0.36	0.12
HBF (full arithmetic)	0.42	0.14
HE	0.15	0.05

The HBF design versions require between two and four FLEX8282 devices, depending on the complexity of arithmetic circuits used. The result of processing a chest X-ray image using our HBF hardware is shown in Figure 4.

5. CONCLUSIONS

Our main goal was to find a low cost solution to increasing the computational speed of image enhancement algorithms. We have done that by substituting the software implementations with FPGA-based application-specific hardware. The HBF-HE combination has been chosen as an example for implementation in FPGAs. The algorithm can be customised to specific enhancement requirements, downloaded into FPGAs, and executed by a single host PC instruction. We have demonstrated the feasibility of the whole concept of a flexible algorithm execution using virtual hardware units, executed on the FPGA prototyping board. Our further research is directed towards other algorithms for image enhancement and their implementation using FPGAs to form a library of hardware-implemented algorithms. They are to be combined in any desired order or with different implementation variations on a PC/FPGA flexible hardware/software platform.

6. REFERENCES

[1] Hummel,R. (1977) Image enhancement by histogram transformation, *Computer Graphics and Image Processing*, vol.6, 184-195.

[2] Kobayashi,N., Saito,H & Nakajima,M. (1994) Fast adaptive contrast enhancement method for the display of gray-tone images, *Systems & Computers in Japan*, vol.25 No.13, 87-94.

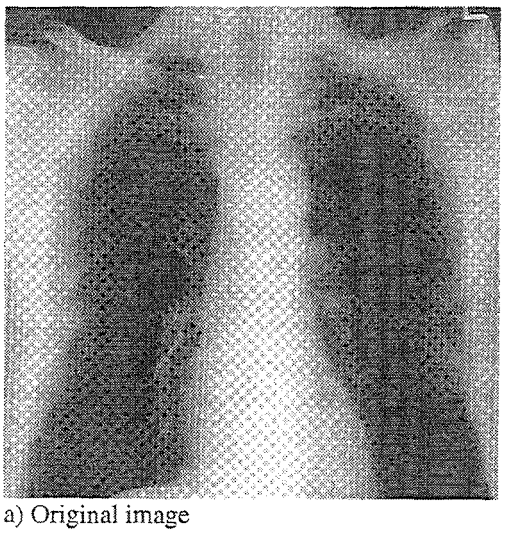
[3] Kurak,C.W. (1991) Adaptive histogram equalisation: a parallel implementation, *Proceedings of Fourth Annual Symposium on Computer Based Medical Systems*, 192-199.

[4] Pizer,S.M., Amburn,E.P., Austin,J.D., Cromartie,R., Geselowitz,A., Greer,T., ter Haar Romeny,B., Zimmerman,J.B. & Zuiderveld,K. (1987) Adaptive histogram equalisation and its variations, *CVGIP*, vol.39 No.3, 355-368.

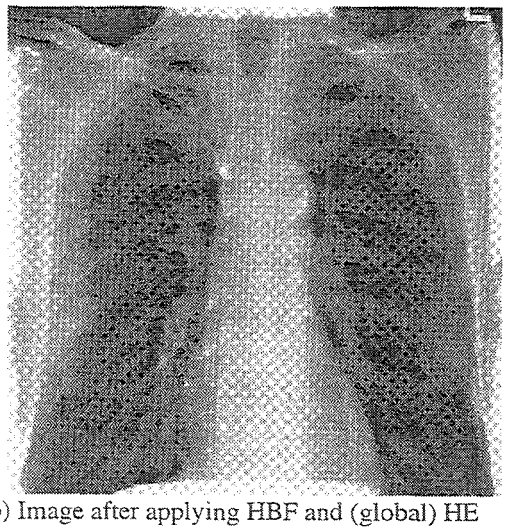
[5] Pizer,S.M., Johnston,R.E., Ericksen,J.P., Yankaskas,B.C. & Muller,K.E. (1990) Contrast-limited adaptive histogram equalisation: Speed and effectiveness, *Proceedings of First Conference on Visualisation in Biomedical Computing*, 337-345.

[6] Salcic,Z. & Cheng,M.S.(1997) RAPROS - A Rapid Prototyping System for PC-compatible Hardware/Software Solutions", to be published in *Proceedings of International Conference on Manufacturing Technology*, Auckland, New Zealand, 1997

[7] Sherrier,R.H. & Johnson,G.A. (1987) Regionally adaptive histogram equalisation of the chest, *IEEE Transactions on Medical Imaging*, No.1., 1-7.



a) Original image



b) Image after applying HBF and (global) HE

Figure 4. Contrast enhancement of a chest X-ray image