Embedded Hardware Face Detection*

T.Theocharides, G. Link, N. Vijaykrishnan, M.J. Irwin
Dept. of Computer Science and Engineering, Pennsylvania State University {theochar, link, vijay, mji}@cse.psu.edu

W.Wolf
Electrical Engineering, Princeton University
wolf@princeton.edu

Abstract

Face detection is the first step towards face recognition and is a vital task in surveillance and security applications. Current software implementations of face detection algorithms lack the computational ability to support detection in real time video streams. Consequently, this work focuses on the design of special-purpose hardware for performing rotation invariant face detection. The synthesized design using 160nm technology is found to operate at 409.5 kHz providing a throughput of 424 frames per second and consumes 7 Watts of power. The synthesized design provided 75% accuracy in detecting faces from a set of 55 images that is competitive with existing software implementations that provide around 80-85% accuracy.

1. Introduction

Face detection is defined as the process of identifying all image regions that contain a face regardless of the position, the orientation and the environment conditions in the image. In essence, it is different than face recognition, because face recognition process already *knows* that an image contains a face, the problem now shifts into identifying the person whom the particular face belongs to [10, 13, 25]. As a result face detection has been a major research topic both in academia and industry, and the popularity of the topic appears in a wide range of applications and fields. From security to identification systems, face detection plays a primary role. It is the primary step towards face recognition [18] and serves as a fore step towards multiple applications such as identification, monitoring, tracking, etc. Face detection algorithms have been developed through the years, and have improved drastically both in terms of performance and speed. However, with today's design technology, we are given the chance to perform face detection at a higher level, which involves the real time domain, and independent of image and environment variations. Face detection so far has been extensively done in software, but with the technologies approaching the nanometer era, and the improvement of the algorithms, we are able to shift the detection stage in the hardware domain, to achieve several advantages.

Software face detection methods have reached a very high level of both effectiveness and detection rate, as well as a condition-invariant level, where detection can be performed under harsh environments. However, the state-of-the-art software face detection require around 3 seconds to detect a single image and are not quite suitable for real-time deployment. While there are some software implementations that operate in less than 1 second, their detection rates in the presence of environmental variations are poor. Hence, a fast hardware implementation that can be integrated either on a generic processor or as part of a larger system, directly attached to the video source, such as a security camera or a robot's camera is desirable.

There has been extensive research in the field, ranging mostly in the software domain [1, 2, 3, 4, 5, 8, 10, 12, 14, 15, and 17]. There have been a few attempts at hardware implementations that implement face detection on FPGAs as well as various microcontrollers and multiprocessor platforms using programmable hardware [7, 12, 13, and 19]. However, many of the proposed solutions are not compact. For example, the FPGA implementation utilizes nine boards [7]. Also, the attempted hardware implementations generally feature algorithms that are not as effective as traditional software approaches such as Competitive Feature Approach [7]. An exception is the implementation using neural networks in [19]. However, the implementation is not purely on hardware; rather than on a reconfigurable multi-processor platform integrated with embedded software. In contrast to previous approaches, our goal is to design a real time face detection system that provides real time detection while maintaining the detection accuracies achieved by state-of-the-art software implementations. Many software algorithms have been developed that can detect faces over a variety of environments and lighting conditions. A problem with almost all of these detection algorithms for real time support is the complexity of the preprocessing and filtering stages that the image goes through before the detection stage [1, 2, and 12]. Therefore, the focus of this paper is to implement an already established face detection algorithm in hardware, focusing on the speed, accuracy and area of the implementation. The next section describes face detection algorithms and relevant research issues. Section 3 provides the details of the proposed architecture. Section 4 shows the

* This work was supported in part by grants from NSF 0093085 and MARCO 98-DF-600 GSRC

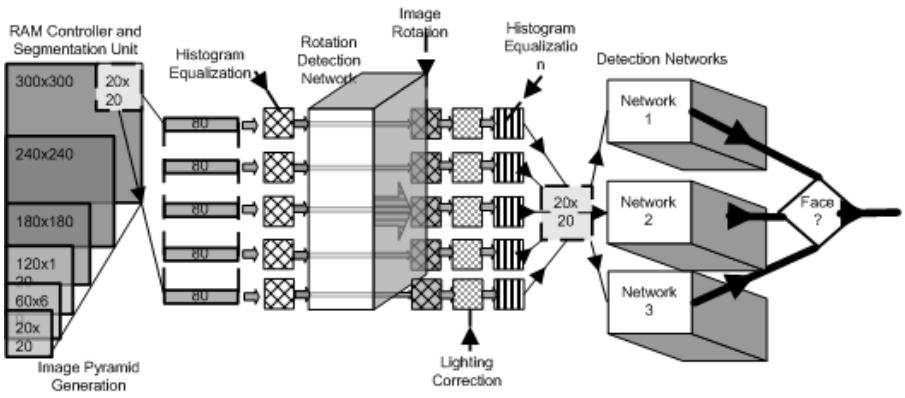

Figure 1: Algorithm Block Diagram

performance parameters of the synthesized hardware. Conclusions are provided in Section 5.

2. Face Detection – A General Background

This section gives a short background of the face detection process, and emphasizes the reasons behind our approach. Most of the face detection algorithms operate in three stages, two of which are common to all algorithms. The first stage is *image pyramid generation*. The purpose of this stage is to divide the input image into multiple segments of equal size, in order to allow parallel searches, as well as to perform search for a face in a greedy approach. This is enhanced by down scaling the original image by a constant scale factor, and by generating additional segments, thereby ensuring that if a face in the image happens to be larger than the generated segment size, it will still be a part of a down scaled segment.

The second stage is the *preprocessing stage*. This stage is also common to all algorithms, however it varies among algorithms in the way it performs its task. The task of this stage is to eliminate as much environment and lighting variations from an input image as possible, by performing various filtering functions over the image, reducing the variance. The filtering functions used during this stage depend on the robustness of the detection stage, which is the third and final stage.

The generalized task of the third stage is to take an image segment that has been filtered for adjustment to maintain environment and pose invariance, and to output whether or not the segment contains a face. This stage is initialized with a training set of data, that is a database of features that are used to match the presence of a face or not. The training set should contain both positive and negative examples that are both images with faces, and without faces [10, 13, 14].

Face detection algorithms can be classified into two major categories: Feature Based / Template Matching Approach, and Image Based / Pattern Classification Approach [10]. The first approach deals with face detection by searching for features that are unique to faces, or uses image features such as color and geometric models to search for a face. The second approach employs classification – it treats the image as data to be classified into containing a face or not. This second approach is more commonly employed, and includes neural networks, support vector machines and linear subspace methods such as PCA and ICA [10, 13]. As mentioned earlier, a generic face detection unit can be partitioned into the three major stages: image search, image processing and detection. The first two stages require most of the data manipulation, as well as large-scale mathematical operation such as down scaling and filtering. The third stage operates in a different manner however, and that makes it special – given that the detection unit classifies an image as a potential face or not, it has to have two different modes of operation – training and recognition. During the training mode, the detection unit is given data that contains faces and data that does not contain faces. During the detection stage, the unit uses the data from the training phase to classify the image as a face or not.

3. Architecture and System Overview

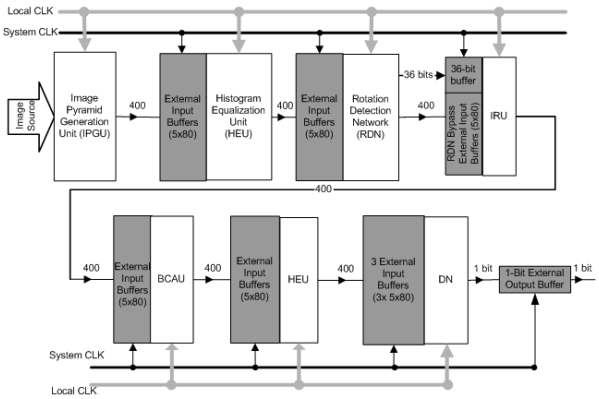

Figure 2: Hardware Block Diagram. Note that the external input buffers receive processed data from the internal output buffers of each unit at each system clock cycle, and serve as inputs to the next unit.

Our hardware implementation is based on the classification-based rotation invariant neural network algorithm proposed originally by Rowley, et al [1, 2]. The algorithm was selected because of the high parallelism capabilities as well as the high detection rate achieved by this algorithm. The algorithm and the corresponding hardware block diagram of the proposed implementation are shown in figures 1 and 2, respectively. The system consists of 7 stages. The first stage is the image pyramid generation [1], which generates 20x20 image segments (windows) and at the same time, down scales the original. Downscaling is necessary for faces that are larger than the processing window, to be included in the search. For example, if a face in the picture covers the entire image, it will be included in the final 20x20 window by successive downscaling of the image. The second stage is a histogram equalization unit, which performs discrete

histogram equalization over each window [4, 14]. The third stage is a rotation detection neural network [2]; this unit assumes that there is a face in the input window, and its task is to determine the angle of rotation of the presumed face. Upon determining the rotation angle, the fourth stage rotates the window on the opposite direction by the angle detected, thus bringing the presumed face in the upright position. The fifth stage performs brightness adjustment [4, 14] on the window in order to emphasize contrast between features. The sixth stage performs discrete histogram equalization on the window again, as rotation and brightness adjustment adjust the intensity values of the window. Finally, the seventh stage consists of three neural networks that are trained to detect upright images. Each network outputs a 0 (non face) or 1 (face) and the final output is selected by a majority vote of the 3 networks. In order to exploit parallel searches, each 20x20 window is processed in parallel as 5 groups of 80 pixels each.

Each unit interfaces to an output buffer of 400 pixels size, receiving each unit's outputs at the unit's internal clock speed, and outputs the entire 400 pixels to the external input buffer of the next stage at the system clock speed. Within a stage, all units perform internal operations using a faster local clock. Internal units communicate with each other via a sequence of 2 handshake signals, named *ready* and *valid*. Data is transferred if and only if the destination unit's *ready* signal and the sending unit's *valid* signal are high during a positive transition of the local clock. Each output buffer, once full, drops its *ready* signal and does not take any further inputs until the end of the system clock cycle. At that time, it transfers the data to the next stage and raises its ready signal to continue operation. This architectural choice permits the use of a slow clock for the entire design and a fast clock for each individual unit. Let us now take a closer look at each stage and explain the operation of each unit.

3.1. Image Pyramid Generation Unit (IPGU)

The IPGU is responsible for taking a 300x300 pixel image and generating 20x20 image windows. At the same time, the unit down scales the image subsequently to 240x240, 180x180, 120x120, 60x60 and finally, to 20x20 pixels. The IPGU interfaces with two on-chip RAM's, of sizes 300x300 and 240x240 bytes (pixels) respectively. Initially, the input image is stored into the 300x300 RAM (RAM 1). For each 2 local cycles, the IPGU reads 8 pixels from RAM 1 and stores them in an input buffer, with their corresponding x, y coordinates (i.e. RAM addresses). Then it shifts these 8 pixels to the output 80 pixel buffers, and in parallel, calculates the new pixel coordinates in order to down scale the image. Image scaling is performed using the matrix scaling equation

$$\begin{pmatrix} X' \\ Y' \end{pmatrix} = \begin{pmatrix} X \\ Y \end{pmatrix} \begin{pmatrix} Sx & 0 \\ 0 & Sy \end{pmatrix}$$

where X' and Y' are the new pixel coordinates, and Sx and Sy the scale factor in each direction [20]. In our implementation, the five scale factors required to generate the necessary images are stored in a lookup table. The execution unit consists of eight 8-bit multipliers, therefore for each pixel it takes two multiplication cycles. The resulting coordinates along with the corresponding pixels are stored in another buffer that communicates with the second RAM (240x240). Since different pixels map to the same coordinates when downscaling, each pixel's generated coordinates are compared and only the intensity of the pixel with the largest coordinates from the original image among the matching coordinates (in the down scaled image) are written into the 240x240 RAM (RAM 2). In addition, since the windows are overlapping, the pixels that have already been mapped will not need to be mapped again, hence they are simply propagated to the output. The process of downscaling is repeated for all the pixels in the 20x20 window. At this point, the five 80-pixel output buffers that feed to the histogram equalization stage are full. It must be noted that when all 20x20 windows have been scaled down, the roles of the input/output RAMs switch. Hence RAM 2 serves as the input for generating the 180x180 scaled down image, and so on. The IPGU needs 100 local cycles to complete its operation.

3.2. Histogram Equalization Unit (HEU)

The HEU transforms a set of input pixels from the 20x20 window into a set of output pixels, such that the intensity distribution of the output pixels is more uniform. In the proposed architecture, equalization is performed locally, among a 4x4 pixel window. The 16 pixels are fed into a CAM/shift register (CAM/SR) structure [11] sequentially. If the intensity of the pixel matches an already existing entry, a corresponding counter associated with that entry is incremented. If no match is found, a new entry is created for that intensity and the counter is set to one. Intensity values are sorted on the fly in the CAM structure, using a linear address sorting technique presented in [21] such that the minimum intensity is at the entry 0 of the CAM/SR. Note that the counters associated with the intensities move correspondingly during sorting. After the sorting is complete, the sum of the counters is evaluated from counter 0 to n (where n is the number of distinct intensities). Next, the pixels of the 4x4 window are used to address the corresponding cumulative sum in the counter associated with their intensity. The value of the counter and the intensity of the pixel, along with a scaling factor are used to obtain the pixel in the equalized image. The scaling factor is given as *(max intensity – min intensity +1) / 16*, which is implemented using a subtractor and a right shift. In our synthesized design the unit requires a total of 38 cycles to complete its operation. There are five units per 80 pixels (25 units total).

3.3. Rotation Detection Network (RDN)

The next stage is a neural network classifier (Fig. 5) that detects the angle of rotation of the presumed face in the image. The network is trained to distinguish between 36 classes, each class representing an angle of rotation from 0º to 360º, in increments of 10º. The network consists of three layers of neurons that are forward fully connected. The network consists of three different types of neurons. First we provide details of our hardware implementation of each type of neuron, which are the basic processing elements in our network. Each neuron (Fig. 3) takes a vector input of n components. Each vector component is multiplied by a fixed weight value, which is determined at training time. Weights are updated during the training process, but remain constant during the detection process. The user also sets a threshold value t, which is subtracted from the sum of the products. The result is then passed as an input to an activation unit. In our case, the activation function used is the hyperbolic tangent implemented using a ROM.

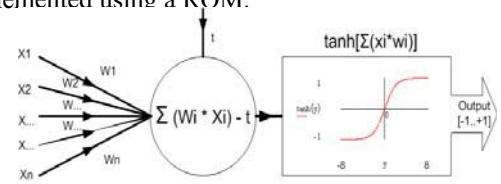

Figure 3: The Neuron

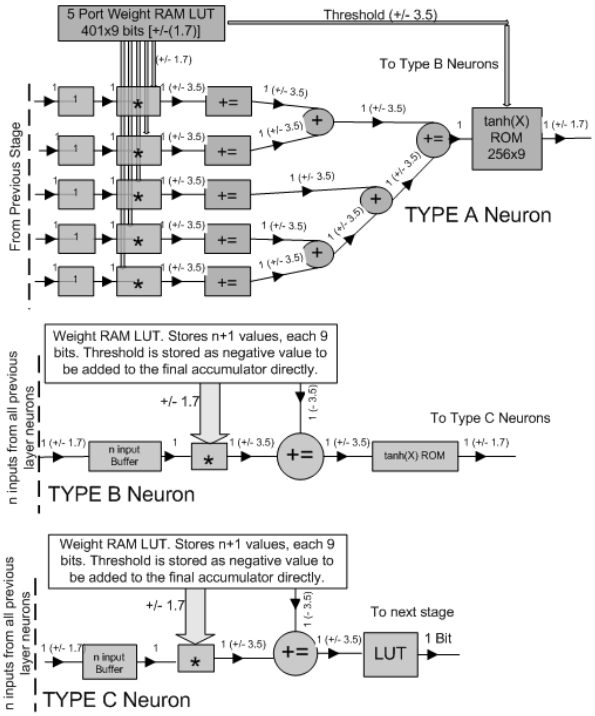

Figure 4: The three different types of neurons

We exploit the symmetric property of the hyperbolic function about the two axes in reducing the number of ROM entries. The ROM takes the rounded 8-bit result of the MAC operation and returns the value of the hyperbolic tangent in a +/- 3.5 (a sign bit, 3 bits for integer and 5 bits for fraction) format. Figure 4 shows the three different types of neurons; type A, B and C. Type A neurons are the input neurons, taking all 400 pixels from the incoming 20x20 window as inputs, and output a value between 1 and -1. Type B neurons are the ones that are in the hidden layer, taking the outputs of type A neurons as inputs, and similarly output a value between 1 and -1. Type C neurons take as inputs all the outputs of the type B neurons, and use a lookup table (LUT) to return a single digit, either 0 or 1. Note that the numeric format of the data between neurons is also shown in figure 4. In the RDN shown in figure 5, there are a total of 66 neurons. The first layer is made up of type A neurons and each neuron takes all the pixels in the 20x20 window as inputs. Due to the large fan-in for this operation, we allocate extra hardware for this stage as compared to the others. Hence, the type A neuron operates on five pixels in parallel as compared to sequential operation on pixels in the case of type B and type C neurons. The second layer is made up of type B neurons, and each neuron takes the outputs of all the first layer neurons. Finally, the output layer takes the outputs of the second layer as inputs, and outputs a single bit. Since there are 36 type C neurons in the output layer, an array of 36 bits makes the output of the RDN, and is sent to the rotation unit. In order to perform the rotation detection, type A neurons need a total of 104 cycles, type B neurons need a total of 17 cycles, and finally type C neurons need again 17 cycles.

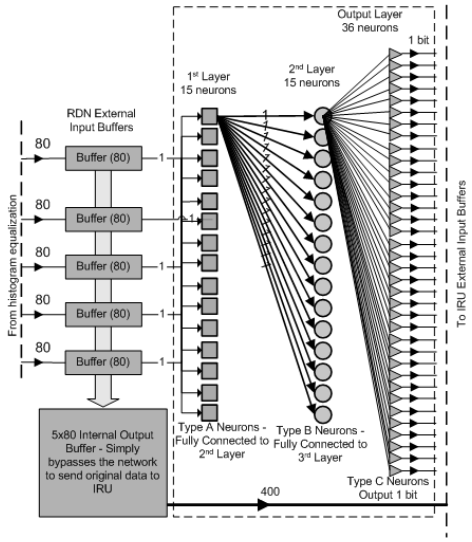

Figure 5: Rotation Detection Neural Network

3.4. Image Rotation Unit (IRU)

The rotation unit receives the 20x20 image from the HEU, and waits until the RDN generates the 36-bit array in order to rotate the image around its center to bring the face upright. There are 5 identical execution units (XUs), which operate on 80 pixels each, therefore a total of 25 XUs. Rotation is performed by determining the value of the new pixel coordinates, based on the old ones, using the equation

$$\begin{pmatrix} X' \\ Y' \end{pmatrix} = \begin{pmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix}$$

Each unit uses a lookup table which takes as inputs the 36 bits, and returns the values of sin Θ and cos Θ, and for each pixel coordinate, and generates the new coordinates for the 80 pixel window. Each control unit then writes its 80 pixels on a 20x20 register, which will hold the rotated window. To generate the rotated version of a 20x20 window the IRU requires 4 multiplications and 2 additions per pixel. There are 4 multipliers and 2 adders in each XU, which perform the requested operations in parallel per pixel for the x and y coordinates. The IRU XU operates on one pixel per local clock cycle, and takes 80 cycles.

3.5. Brightness and Contrast Adjustment Unit (BCAU)

BCAU performs brightness and contrast adjustment in order to emphasize the contrast between facial characteristics. Since, the algorithm outlined in [1] required excessive hardware resources to be considered, we implemented a variation of the algorithm from [18]. The idea is to look over a small region, and find the average intensity. Then, we find the deviation of each pixel from the average. Next, we add/subtract a constant value read from a look-up table to/from each intensity value, emphasizing the places where the intensities are higher than the average, and lowering the intensities which are smaller than the average, emphasizing the contrast.

There are five units, each operating on an 80-pixel group. The adjustment itself is done in five 4x4 pixel groups of each 80-pixel group. A total of 185 cycles are required for each 80 pixels.

3.6. Detection Neural Network (DNN)

The final unit of our detector consists of 3 parallel neural network units. A single unit is shown in figure 6. Each network is partitioned into 3 parallel stages. The first network stage segments the image in 4 regions, searching for large facial features such as nose, eyebrows, glasses, etc. The second stage segments the image in 16 5x5 regions, searching for small features, such as eyes, moles, etc. The third stage segments the image in six overlapping strips, each of size 5x20 pixels. This stage searches for features such as pairs of eyes and the mouth. Each DNN is made of three layers of types A, B and C neurons respectively, identical to the ones explained earlier. Each layer is fully connected to the next layer, and all three outputs from each DNN are then used to classify the input 20x20 window as a face or not. Each layer of neurons has to receive all inputs from the previous layer to compute its own outputs. As seen in figure 6, layer 1 consists of types A and B neurons. The third layer is made of a type C neuron, which takes 3 inputs, as shown in figure 6. The total number of cycles required to perform detection is 51.

4. Synthesis and Detection Results

4.1. Synthesis Results

The system was implemented in Verilog HDL, and synthesized using Synopsys Design Compiler® using 160nm technology. For the particular design, grayscale images of size 300x300 pixels were used, and the hardware was designed to accommodate images of this size. In our synthesized design, the local clock operates at 13.2ns. The system clock now can operate at the largest time each of the units requires to finish all 400 pixels. The BCAU, taking a total of 185 cycles, has the largest overall delay of 2442ns. That allows the system clock to operate at 409.5 kHz.

Chip Details	
System Clock Frequency(kHz)	409.5
Local Clock Frequency (MHz)	75.75
Area(mm^2)	30.4
Power(W)(@V_{DD} = 1.8V)	7.35
Detection Frame Rate (fps)	424
Frame Size (pixels)	300x300

Table 1: Summary of Synthesis Results

Table 1 shows the synthesis results for the entire detection system. The overall clock frequency is at 409.5 kHz. That translates into a total of approximately 409000 20x20 windows per second. A 300x300 image generates a total of 965 20x20 windows. The frame rate of the entire system is therefore timed at approximately ~424 (entire) image frames per second, a frame rate that not only meets real time requirements, but also provides sufficient time for other related operations to be completed once detection has been achieved. It also provides us a large slack to explore possible area reduction or addition of more accuracy and precision mechanisms. Table 1 also summarizes other relevant synthesis results, such as the area and the power consumption.

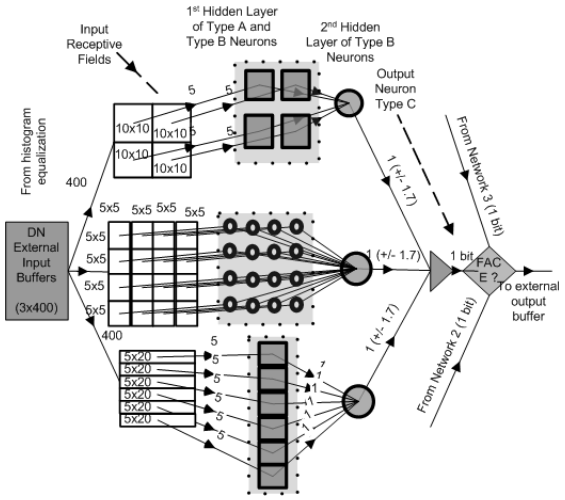

Figure 6: Detection Neural Network

4.2. Detection Results

The system was trained using 130 faces from the CMU face database [22, 23] as well as faces used initially by similar algorithms [1]. In order to test the system from the detection point of view, various images from existing face databases as well as the World Wide Web were collected, and a database of 55 images was constructed. 10 of the images belonged to the training set, in order to ensure that the system worked from a functional standpoint. In the 45 images not in the training set, 40 had faces in them and 5 did not have. Of the 40 images containing faces, 15 contained rotated images, at various possible angles. The images were then tested against the face detection software constructed by Rowley et al. and then tested against our implementation. We trained our system in the same procedure, and obtained the weight values. We converted the values into the hardware representation format, and stored them in the RAMs. Table 2 shows a summary of the results.

Hardware vs. Software Test Results					
Test Case		Total # of Faces	Detected Faces	False Positive	Detection Rate [%]
Upright Detection	H/W	186	148	39	79.57%
	S/W	186	161	28	86.56%
Rotated Detection	H/W	87	59	67	67.81%
	S/W	87	69	42	79.31%
Combined Results	H/W	273	207	106	**75.82%**
	S/W	273	230	70	**84.20%**

Table 2: H/W vs. S/W Detection Performance

The testing procedure involved some preprocessing as the images had to be scaled to the size that our hardware version supported. The entire image set was down scaled to 300x300 grayscale images, and the intensity values were exported in a text file. A script then was used to direct a Verilog testbench to read each text file and use the intensity values as inputs to the RAM module. The testbench collected intensity values upon completion of each image, and exported them again to text files, with coordinates of the faces detected in the text file. The weights as well as all the

other constant values (such as intensity and histogram necessary values) were rounded to the corresponding format, and a C program was used to generate text files containing the weight values as represented in hardware. The Verilog testbench read the values and placed them on the on-chip memory of each component.

As seen in Table 2, the performance of the hardware detection system is close to the performance of the software system, matching its detection rate at 90%. The software version detected 230 faces, where as the hardware version detected 207 faces. There are quite a few reasons why the results in software are relatively better. Hardware performance was affected from the HEU as well as the BCAU given that we operated in a smaller window for these units instead on the entire 20x20 window. However, it is to our belief that the most important reason is of course the mathematical operations during the detection stages, where lots of multiply and accumulate operations are forced into rounding. Given these results, it is very interesting to explore possible alternatives for performing the multiplication using a higher number of bit representation, or even going into the floating point domain, that involves tradeoffs in area and speed. In comparing the two detectors in terms of speed – the software version processes an image in an average time of 2 seconds [1,2,10], where as our proposed hardware model processes ~424 images per second.

5. Conclusions

This paper presents a hardware implementation of face detection and demonstrates that the proposed architecture achieves 75% detection accuracy. The proposed hardware makes it possible to detect faces in real time that is not possible using existing software presentations and the improved throughput is of significant importance for surveillance and security applications. There are lots of issues yet to be explored, the most important of which is to improve the accuracy of the detection rate of the hardware implementation.

6. References

[1] H. A. Rowley, S. Baluja, T. Kanade, "Neural Network-Based Face Detection", *IEEE Trans. On PAMI*, Vol.20, No. 1, Page(s).39-51, 1998.

[2] H. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network- based face detector. In *Proc. IEEE Conf on Computer Vision and Pattern Recognition*, Page(s) 38-44, Santa Barbara, CA, June 23-25, 1998

[3] B. Nagendra, "Pixel Statistics in Neural Networks for Domain Independent Object Detection", *MSc Thesis, RMIT University*, Melbourne, Australia, 2001.

[4] K. Sung, "Learning and Example Selection for Object and Pattern Recognition", *PhD Thesis, MIT*, MIT Press, 1995.

[5] S. Ben-Yacoub, B. Fasel, "Fast Multi-Scale Face Detection*" IDIAP*, Eurecom, Sofia-Antipolis, France, 1998.

[6] Oki Semiconductor, http://www.oki.com, June 27th, 2003.

[7] R. McCready, "Real-Time Face Detection on a Configurable Hardware System", *International Symposium on Field Programmable Gate Arrays*, 2000, Monterey, California, United States.

[8] R. Herpers, G. Verghese, et al, "Detection and Tracking of Faces in Real Environments", University of Applied Sciences, Sankt Augustin, Germany.

[9] Fraunhofer Institute of Integrated Circuits, IIS [http://www.iis.fraunhofer.de/].

[10] Erik Hjelmås, Boon Kee Low, "Face Detection: A Survey", *Computer Vision and Image Understanding*, Vol. 83, No. 3, September 2001.

[11] N.Ranganathan, *VLSI Algorithms and Architectures*, IEEE Computer Society Press, Los Alamitos, California, USA, 1993.

[12] Viola and Jones, "Robust Real-time Object Detection", *Statistical and Computational Theories of Vision Modeling, Learning, Computing and Sampling,* Vancouver, Canada, 2001.

[13] Yang Ming-Hsuan, DJ Kriegman, N Ahuja, "Detecting faces in images: a survey", *IEEE Trans. on PAMI*, Volume: 24 Issue: 1, Page(s): 34 -58, Jan. 2002;

[14] K. Sung, T Poggio, "Example-based learning for view-based human face detection", *IEEE Trans. on PAMI*, Volume: 20 Issue: 1, Page(s): 39 -51, Jan. 1998.

[15] R. Feraund, O.J. Bernier, J. Viallet, M Collobert, "A fast and accurate face detector based on neural networks", *IEEE Trans. On PAMI*, Volume: 23 Issue: 1 , Page(s): 42 - 53, Jan. 2001

[16] A. K. Jain, J. Mao, "Artificial Neural Networks: A Tutorial", *IEEE Computer*, Vol. 29, No. 3, Page(s) 56-63, March 1996.

[17] Zhang ZhenQiu, Zhu Long, S.Z. Li, Zhang HongJiang, "Real-time multi-view face detection" *Proc. of the 5th IEEE International Conference on Automatic Face and Gesture Recognition*, Page(s): 142 -147, 20-21 May 2002.

[18] F. Soulie, T. Huang, "Face Recognition, From Theory to Applications", *NATO ASI Series, Series F: Computer and Systems Sciences*, Vol.163, Springer, NY, 1997.

[19] B. Srijanto, "Implementing a Neural Network based face detection onto a reconfigurable computing system using Champion", *MS Thesis, University of Tennessee*, Knoxville, August 2002.

[20] E. Trucco, A. Veri, *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, New York, 1998

[21] M. Verleysen, B. Sirletti, A.M. Vandemeulebroecke, P. G. Jespers, "Neural Networks for High-Storage Content-Addressable Memory: VLSI Circuit and Learning Algorithm", *IEEE Journal of Solid State Circuits*, Vol. 24, No. 3, June 1999.

[22] Henry Schneiderman, Takeo Kanade, and Jay Pujara, "CMU Face Detection Algorithm Demonstration.," http://www.vasc.ri.cmu.edu/cgi-bin/demos/findface.cgi. 2001

[23] H. Schneiderman and T. Kanade, "Object Detection Using the Statistics of Parts", *International Journal of Computer Vision*, 2002.

[24] FERET Web Based Face Image Database, , http://www.itl.nist.gov/iad/humanid/feret/feret_master.html, June 2003.

[25] J.L. Ayala, "Design of a Pipeline Hardware Architecture for Real-Time neural network computations", Universidad Politechinca de Madrid, Spain, 2002.

[26] R. Frischholz, *The Face Detection Homepage*, http://home.t-online.de/home/Robert.Frischholz/face.htm, July 2003.