Contents lists available at SciVerse ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Face detection system for SVGA source with hecto-scale frame rate on FPGA board

Zheng Ding [a,*], Feng Zhao [b], Wei Shu [c], Min-You Wu [a]

[a] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
[b] Digilent Electronic Technology Co. Ltd., Shanghai, China
[c] Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, USA

ARTICLE INFO

Article history:
Available online 16 February 2012

Keywords:
Face detection system
Field programmable gate array
Haar feature

ABSTRACT

This paper proposes techniques for face detection using Haar-like features as weak classifiers and gives the implementation details for an FPGA development board. We analyze and discuss the relation between the system computation cost and selection of the image scaling factor. Based on the empirical results of our previous work, we give a new method to select the stop threshold for the image reduction process, which reduces the total computation by half. We present and implement an improved integral image pipeline calculation design. We also provide a color image output mode to let our system enjoy more human-oriented design. Test results show that the system achieves real-time face detection speed (100*fps*) and a high face detection rate (87.2%) for an SVGA (600×800) video source. The low power consumption (3.5 W) is another advantage over previous work.

1. Introduction

Progress in VLSI technology has led to the use of field programmable gate arrays (FPGAs) as a component for high performance computing (HPC). Scientists have been aiming at high performance gained through FPGAs in face detection applications.

Face detection [1] is concerned with finding whether there are any faces in a given image or not, and if there are, returning the image locations and contents of all faces. Usually grayscale images are used; if the input is not grayscale, a pre-transform will be performed. Since it is the first step of any fully automatic system that analyzes the information contained in faces, face detection is one of the most active research topics in the video surveillance, secure access control, face recognition, and face image database management field. Approaches to face detection can be divided into two categories [2]: feature-based approaches and image-based approaches. Systems based on Haar features generally include an image input/output component and a face detection component. The face detection component usually has three parts: the image scaling part, the integral image calculation part and the Haar feature calculation and detection part.

Systems using general purpose processors or graphics processing units (GPUs) have reached a very high detection rate for any image condition (such as low light intensity and profile). However, typical software implementations [3–7] give a detection speed lower than 15*fps* (frames per second), far less than realtime applications require. A realtime face detection system which can be directly attached to the video source (such as an industrial camera) is urgently needed.

There has been some work on implementing face detection using FPGAs. Cho et al. [8] implemented a full FPGA-based face detection system, but the best performance was only 28*fps* at 320×240 resolution and 7.51*fps* at 640×480, well short of the realtime requirement (>60*fps*). McCready's work [9] reaches a speed of 30*fps* with 320×240 input, but it uses a Transmogrifier-2 configurable hardware system which includes nine FPGA chips; this definitely cannot be used in a portable device. Paschalakis and Bober [10] presents an FPGA-based face detection and tracking system for audiovisual communications, with a particular focus on mobile video conferencing. However, the paper does not quantify the test cases and we cannot determine the exact performance of the design. Yu et al. [11] implements the AdaBoost algorithm on an FPGA, focusing on how to implement an efficient face classification stage. Based on synthesis results, Yu's design can operate at 91 MHz, equivalent to 15 video frames (120×120) per second. Changjian and Shih-Lien [12] gets a 37*fps* detection speed for 256×192 size input image by using Virtex 5 LX110T chip, but only the Haar feature classifiers (40 stages, 2192 Haar classifiers) are in the FPGA while the other parts of the system are still in a normal Intel CPU. The data transfer between the FPGA and the CPU through the PCIe interface.

Paper [13] gives a partial introduction to our face detection system. Here we give more details of the system: the algorithm, the system implementation and performance evaluation. For the algorithm, [13] only deals with the image reduction stop threshold and the factor selection problem. Here we compare the image scaling down algorithms, which strongly affect the final system

* Corresponding author. Address: SEIEE Building 3-121, SJTU, No. 800, Dongchuan Road, Shanghai 200240, China. Tel.: +86 21 34204045; fax: +86 21 34204728.
E-mail address: dingzheng@sjtu.edu.cn (Z. Ding).

performance. In system implementation, [13] only introduces the face detection circuit. We detail the image scaling component, the integral image pipeline and buffer, and the feature ROM. In this new integral image pipeline design, there is no draining of the pipeline between different sub-window integral image calculation processes. In performance evaluation, we give the system performance (resource needs, system frequency, detection speed, etc.) for different image reduction algorithms. Also, since the Shanghai IsVision Sample Database is not openly available we also use the Extended Yale Face Database B [14] to verify the effectiveness and commonality of our system.

Compared with previous work, our major contributions are:

1. Design and implement a feasible realtime (100*fps*) face detection system on a general-purpose FPGA development board (XUPV5LX110T board) to deal with a high resolution (600×800) video stream. The maximum supported resolution in previous work is 640×480, but the general output resolution of state-of-art industrial cameras is 600×800 or even higher. Our work can be quickly applied to industry.
2. Propose a new image reduction process stop threshold, reducing the amount of calculation by half on the average.
3. Give a theoretical derivation of the relation between image scaling factor and the computation cost, and a method of selecting the best scaling factor based on that.
4. Test different image reduction algorithms and show by experiment that the area average algorithm is the best for our application.
5. Design a new integral image calculation pipeline to speed up the calculation process. Most previous papers have not optimized the integral image calculation process. Our previous work [15] can pipeline the integral image calculation only within one sub-window. But in this new design, there is no draining of the pipeline between different sub-window integral image calculation processes.

The rest of this paper is organized as follows: Section 2 provides an overview of the face detection system, which includes the image scaling stop condition selection, the image reduction algorithm, the principles of Haar feature based face detection systems, and the input/output component of the system. Section 3 gives more implementation details about the key components, which include the image scaling logic, the integral image calculation pipeline and buffer, and the feature calculation and face detection circuit. Section 4 gives a performance analysis, experimental results and resource utilization, and Section 5 concludes the paper.

2. Face detection system design

In this section, we give an overview of our face detection system. The basics of integral image processing, Haar features and the face detection process are also given. Also, the image reduction stop threshold and scaling factor selection and image scaling algorithm are analyzed.

2.1. System overview

The basic operating mechanism of our face detection system is: Let a sub-window slide on the original image from the upper left corner to the lower right corner following column major order with one pixel steps. The system will calculate the integral image of the sub-window and use the Haar feature classifier to detect whether the sub-window contains a face or not. If the whole image has been examined and no face has been detected, the image will be reduced by a constant scaling factor (f) and the sub-window slide process redone until a face is detected or the image is smaller than a threshold.

Fig. 1 gives an overview of our system. The image source video is input to the system from a VGA interface and output to the monitor via a DVI interface. The chips for VGA and DVI are an Analog Devices AD9980 and a Chrontel CH7301C respectively. The system components include input/output control circuit, IO buffer, image buffer, image scaling circuit, integral image calculation (II Cal.) circuit, integral image (II) buffer, feature ROM and face detection circuit. These are all implemented with Verilog HDL on a Xilinx XC5VLX110T FPGA.

2.2. Image input control module

The functions of the input control module include sending control and synchronization messages to the AD 9980 chip, acquiring images from the video source, transforming the RGB image to a grayscale image and sending the output image to the IO buffer. We have two output options; one is to output the RGB image to the monitor and the other is to output the grayscale image. The image type which is sent to the IO buffer is decided by the output option selection.

The color space transformation is described by the following Eq. (1). Since the division can be replaced by a right shift, it can be done in one clock.

$$Gray = \frac{R \times 77 + G \times 151 + B \times 28}{2^8} \tag{1}$$

2.3. Image reduction stop threshold and factor selection

In our previous face detection system [15] using a Virtex-II Pro board, when the system was tested against the Shanghai IsVision Sample Database which includes 5000 positive images and 11,000 negative images, we got two observations. The first was that 89% of the faces had been detected before scaling down to smaller than 70% of the initial source image size. Table 1 gives the details and the scaling factor used is 1.2. The second was that if a face is detected, almost all other faces in the same image will be detected at the same scaling down level. The intuitive explanation is that all faces in the same image are about the same size so they are detected at about the same level. (Statistical data show that: $P(0.8 \leqslant \frac{ls}{ll} \leqslant 1) = 96.4\%$, $P(0.8 \leqslant \frac{ws}{wl} \leqslant 1) = 97.2\%$. The P means probability, and ls, ws (or ll, wl) are the length and width of the smallest (or largest) face in an image. The smallest and largest faces in all images are 42×52 and 513×547 respectively after the images are normalized to 600×800 size.) Based on these two observations, we can save time for the integral image and Haar feature calculation by optimizing the image reduction stop threshold and factor selection.

The basic detection process in previous work is similar to our design. The difference is that in those designs the image reduction process stops only when the reduced image is smaller than a preset threshold (usually the size of the sub-window). In our design, the process stops either when a face has been detected or the image is smaller than the sub-window. As described in 2.1, after every image scaling, the detection sub-window moves from the top left to the bottom right with a one column (the horizontal coordinate plus one) step. As each sub-window detection process attempts to detect the features, the amount of traveling the sub-window done on the image will determine the computation cost of the whole system. From the above description, we can infer that the computation load of the system is determined by the image reduction factor (f).

We assume the initial image is $P_h \times P_v$ (assume $P_h \leqslant P_v$) size, the sub-window size is $S \times S$ and the image scaling factor is

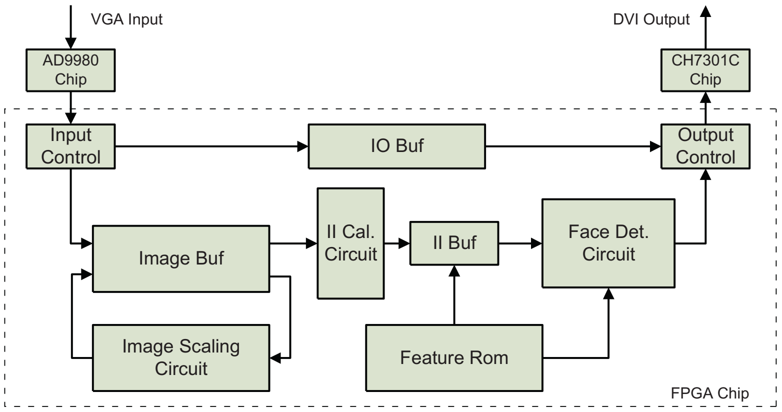


Fig. 1. Face detection system overview.

Table 1
Detection rate for different image scaling levels.

Scale times	Scaled size	D. num.	DR	Accum. DR
0	1	5982	0.374	0.374
1	0.833	6349	0.397	0.770
2	0.694	1987	0.124	0.894
More than 2	$\leqslant$0.579	303	0.019	0.913

$f\left(1 < f \leqslant \frac{P_h}{S}\right)$. If no face is detected in the initial image, the image is scaled down by f until a face is detected or the image becomes smaller than the sub-window size. So the scaling process is performed $\log_f \frac{P_h}{S}$ times. After the ith $\left(0 \leqslant i \leqslant \log_f \frac{P_h}{S}\right)$ iteration, the sub-window needs $\left(P_h \cdot \left(\frac{1}{f}\right)^i - S + 1\right) \cdot \left(P_v \cdot \left(\frac{1}{f}\right)^i - S + 1\right)$ steps to traverse the image, that is:

$$T_i = P_h \cdot P_v \cdot \left(\frac{1}{f^2}\right)^i - (S-1) \cdot (P_h + P_v) \cdot \left(\frac{1}{f}\right)^i + (S-1)^2 \tag{2}$$

We suppose that the face can be detected after the initial image has been scaled down to $\sigma_h \times \sigma_v$ size. The total steps which the sub-window needs to traverse are:

$$T = \sum_{0 \leqslant i \leqslant I} T_i \tag{3}$$

where $I = \log_f \frac{P_h}{\sigma_h}$. We let $w = \frac{\sigma_h}{P_H}(0 < w \leqslant 1)$ and can get T for our technique:

$$T = P_h \cdot P_v \cdot \frac{f^2 - w^2}{f^2 - 1} - (S-1) \cdot (P_h + P_v) \cdot \frac{f - w}{f - 1} - (S-1)^2 \cdot \log_f w \tag{4}$$

In the same way, the steps needed for previous work are:

$$T_p = P_h \cdot P_v \cdot \frac{f^2 - \left(\frac{S}{P_H}\right)^2}{f^2 - 1} - (S-1) \cdot (P_h + P_v) \cdot \frac{f - \frac{S}{P_H}}{f - 1} - (S-1)^2 \cdot \log_f \frac{S}{P_H} \tag{5}$$

We know that T_p is related to the scaling factor f. In this implementation, our input image size is 600×800 and sub-window size is 24×24 so $T_h = 600, T_v = 800$ and $S = 24$. In Fig. 2, the subgraph (a) gives the number of steps needed in our proposal, the subgraph (b) gives the number of steps needed in previous work and the subgraph (c) shows the speedup of our proposal relative to previous work. In these three subgraphs, the variables are w and f. Our design is 50 times faster in the best case. For clarity, we give the variation diagram of T_p and T under $w = 0.83$ condition in subgraph (d) and (e). We also give the normalized speedup in subgraph (f) for different scaling factors f, based on empirical values from our previous work. The normalization ratios of different w value are deduced from our previous experimental data shown in Table 1. We also found that the scaling factor should not be smaller than 1.1 since with small factors the amount of calculation increases dramatically. There is a trade off between the amount of calculation and the detection rate; in our design, we choose a scaling factor of 1.2.

2.4. Image reduction algorithm

The image scaling algorithm in our face detection system should meet three targets. First, it should maintain high quality in scaled images or the quality drop will reduce the face detection rate. Second, it should be fast enough since our image source could be realtime video; a low performance algorithm will cause display discontinuities. Third, it should not use too much logic or memory resources on the FPGA since we need to leave resources for the face detection feature matching units.

Though many image scaling algorithms such as nearest neighbor [16], bilinear [17], and bicube [18] have been implemented on FPGAs, we have selected the area average algorithm [19] for our system. The algorithm can be done on horizontal and vertical dimensions separately with minimal correlation which helps to exploit the parallel capabilities of the FPGA. Also, the scaling algorithm can be performed stepwise and without causing image quality loss. For example, if we need to reduce an image $f_1 \cdot f_2$ times, we can use the algorithm first to reduce the image f_1 times then reduce it f_2 times without quality loss.

The area average algorithm has two phases. If we need to reduce a image f (assume $f = \frac{\alpha}{\beta} > 1$) times, First we magnify each pixel in the original image β times and then shrink the enlarged image α times. Fig. 3 shows an example with $\alpha = 6$ and $\beta = 5$. In the upper part of Fig. 3, the pixel value of the point P can be calculated:

$$P = \frac{P_1 \cdot h_1 + P_2 \cdot h_2}{h_1 + h_2} \tag{6}$$

In the lower part of Fig. 3, the pixel value of the point P can be calculated:

$$P = \frac{P_1 \cdot h_1 v_1 + P_2 \cdot h_2 v_1 + P_3 \cdot h_1 v_2 + P_4 \cdot h_2 v_2}{(h_1 + h_2) \cdot (v_1 + v_2)} \tag{7}$$

or:

$$P = \frac{\frac{P_1 \cdot h_1 + P_2 \cdot h_2}{h_1 + h_2} \cdot v_1 + \frac{P_3 \cdot h_1 + P_4 \cdot h_2}{h_1 + h_2} \cdot v_2}{v_1 + v_2} \tag{8}$$

Comparing Eqs. (6) and (8), we find that the two-dimension scaling down process can be done separately or on the horizontal dimension first, and then on the vertical dimension.

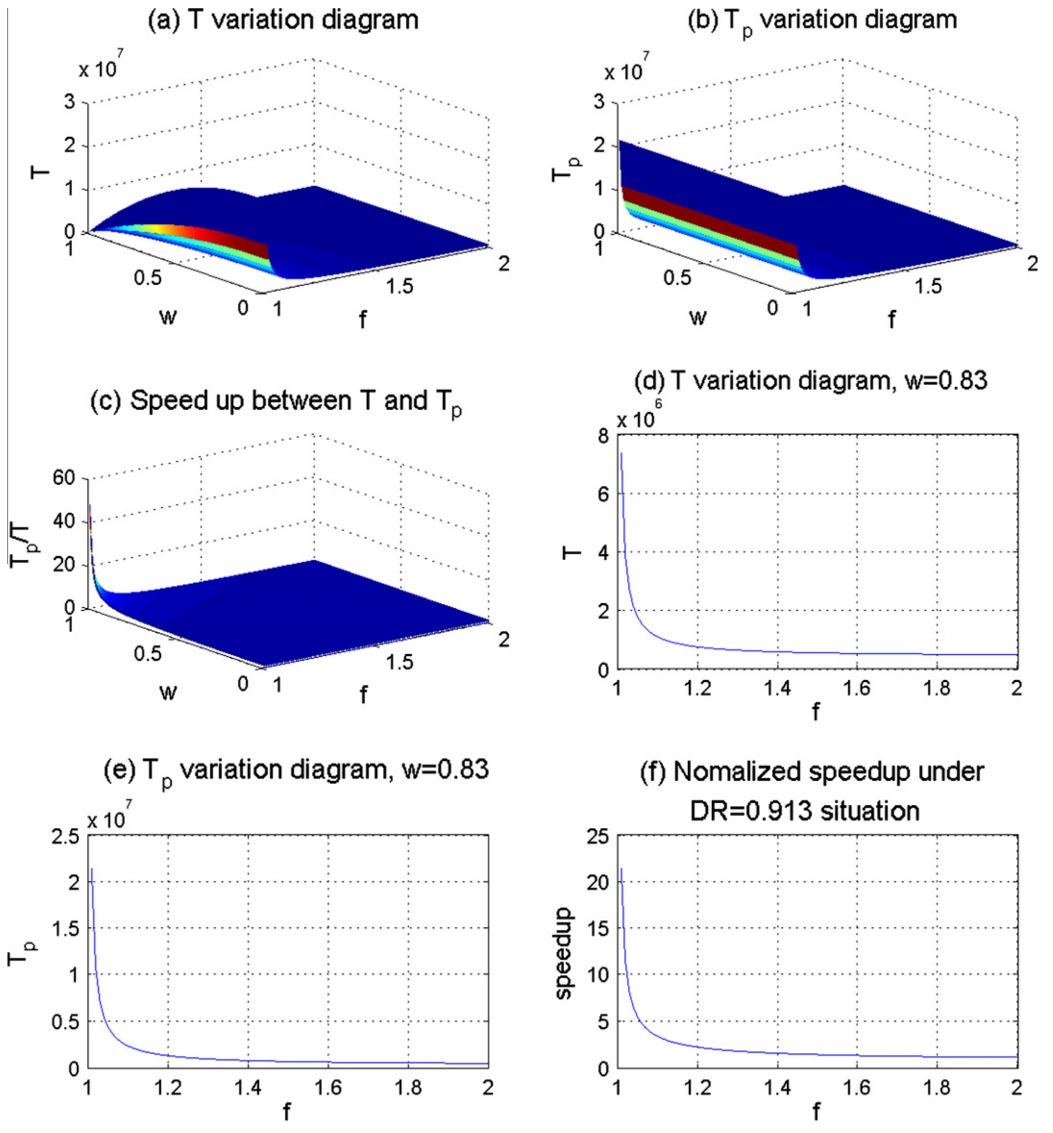


Fig. 2. The relation among T, T_p, w and f.

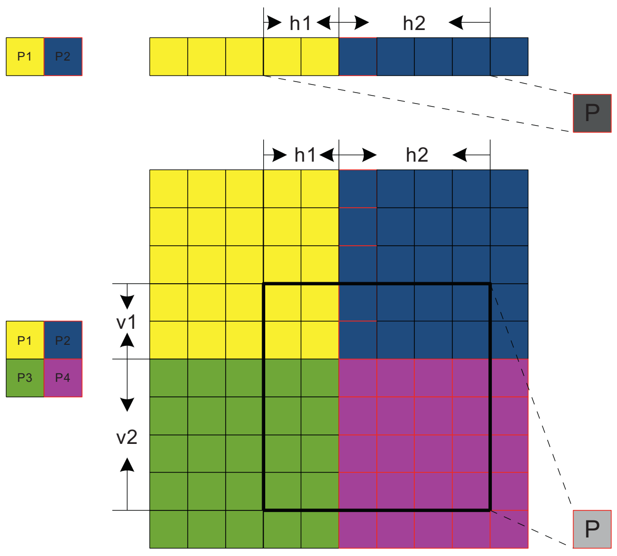


Fig. 3. Image scaling down algorithm.

2.5. Face detection algorithm

2.5.1. Integral Image

The integral image (called summed area tables in the graphics field) $II(x,y)$ can be used to compute rectangular two dimensional image features. It is the sum of the values above and the left of pixel (x,y). Formally, the integral image can be calculated by the following equation:

$$II(x,y) = \sum_{x' \leqslant x, y' \leqslant y} p(x',y') \tag{9}$$

or:

$$S(x,y) = S(x,y-1) + p(x,y) \tag{10}$$

$$II(x,y) = II(x-1,y) + S(x,y) \tag{11}$$

where $S(x,y)$ is the cumulative row sum and $S(x,-1) = 0$, $II(-1,y) = 0$.

2.5.2. Haar feature

Haar features are composed of two or three rectangles. The size, weight and the position of each feature are generated by a machine learning algorithm from AdaBoost. The value of a specific feature can be quickly obtained from the integral image. Fig. 4 shows some feature examples. In Fig. 4, the value of the middle rectangle is $II(P4) + II(P1) - (II(P2) + II(P3))$.

2.5.3. Haar feature classifier based face detection

A Haar feature classifier uses the rectangle integral (e.g. the black minus white rectangle integral value) to calculate the value of a feature. The face detection system organizes different Haar feature classifiers into cascaded stages. The cascade structure is guaranteed to reject non-face image windows as early as possible. Each feature has its weight and feature threshold. If the feature value multiplied by weight is larger than the feature threshold, the weighted value will be added into the stage value. When all weighted values have been accumulated, if the total is larger than the stage threshold then the image window has passed this detection stage and will enter the next stage. If the image window passes all stages, it is recognized as containing a face.

2.6. IO buffer

To improve the output from grayscale to color image, we use an IO buffer to cache the original RGB image. This has two significant advantages. First, we do not need to enlarge the reduced image for output; enlargement is a time-consuming process and also causes image quality degradation. Second, the input of the Haar feature based detection process is a grayscale image; restoring color would be both difficult and unreliable. The IO buffer is constructed using the Block RAM (BRAM) in our system.

2.7. Image output control module

The image output control circuit is used to combine the detected face frame with the original image and output the image and synchronization data to the CH7301C chip for display on a monitor through the DVI interface.

3. Face detection system implementation

In this section, we give more implementation details about the image scaling circuit, the integral image calculation pipeline and buffer, the feature ROM and the face detection circuit.

3.1. Image scaling algorithm implementation

We use an image reduction factor of 1.2 which means the reduced image is $\frac{5}{6}$ of the original image size in both horizontal and vertical dimensions after each scaling process. Following the algorithm introduced in Section 2.4, the method is to compress every 6 input pixels to 5 pixels, first in horizontal order and then in vertical order. Fig. 5 gives the details of the reduction function implementation. The input/output FIFO is used to adapt between the image read/write from BRAM memory speed and process speed. The selector is used to feed pixel values to the proper weighter. In the selector, the *Hnum* and *Vnum* register is used to record the number of pixels of the image in horizontal and vertical dimensions. The *indi* register indicates whether the current reduction is in horizontal or vertical order. The *hcon* and *vcon* registers count

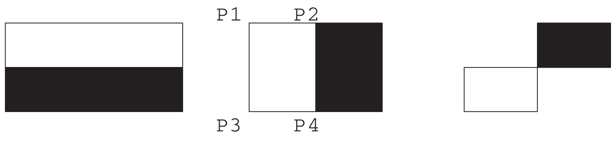


Fig. 4. Haar feature example.

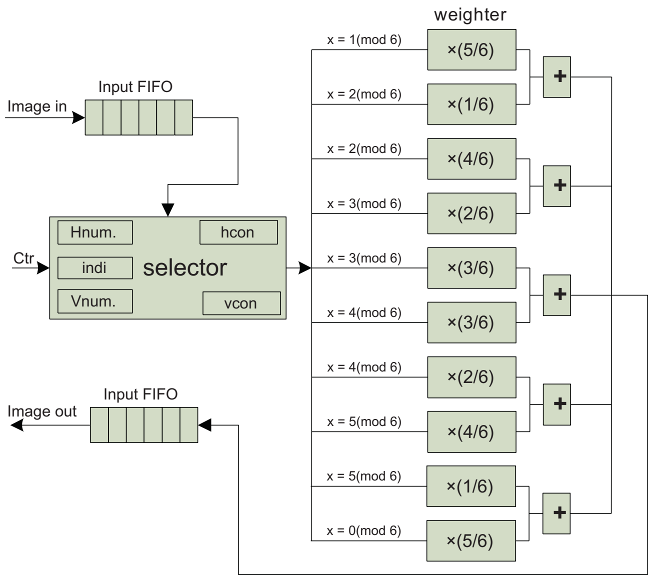


Fig. 5. Image scaling down algorithm implementation circuit.

the number of image pixels which have been processed in the horizontal and vertical dimension respectively.

Before each image reduction, the *Hnum* and *Vnum* registers are set to the proper values according to the image which needs to be reduced. For horizontal order processing, *hcon* and *vcon* are set to 1, and *hcon* is increased by 1 each time data is sent out to a weighter. The value in *hcon* indicates which weighter the data should be sent to. When *hcon* becomes larger than *Hnum* because a line of pixels has been processed, then *hcon* is set to 1 and *vcon* increased by 1 to start the next line. When *vcon* becomes larger than *Vnum*, the horizontal reduction process has finished. Next the vertical scaling is done; it is similar to the horizontal scaling. Note that before each scaling down process, the *Hnum* and *Vnum* will be updated to the proper value.

In each weighter, $\frac{5}{6} = \frac{54{,}613}{2^{16}}$, $\frac{1}{6} = \frac{10{,}923}{2^{16}}$, $\frac{4}{6} = \frac{43{,}690}{2^{16}}$, $\frac{2}{6} = \frac{21{,}845}{2^{16}}$ and $\frac{3}{6} = \frac{1}{2}$ so we can do the division with a right shift in one clock.

3.2. II calculation pipeline and II buffer

The II (integral image) calculation pipeline is used to speed up the generation of II for the sub-window. The II buffer is used to adjust for differences between the II production and consumption speed. As described above, we know that the image is stored in BRAM memory. For speed, the II pipeline and II buffer are both constructed with registers and LUTs.

The II pipeline has 24×24 18-bit content registers which are used to store the corresponding II values and these registers are labeled as $II(i,j)$ (where $0 \leqslant i \leqslant 23$, $0 \leqslant j \leqslant 23$) and organized into 24 columns. Each column has a buffer register, which is tagged as $Bufj(0 \leqslant j \leqslant 23)$. The structure of the II pipeline is given by Fig. 6. In Fig. 6, rectangles tagged as + are adders and − subtractors. As described above, the sub-windows slide on the image in horizontal order, the P_j means a line of pixel values in the image. Each clock, the pixel values will be sent into the II pipeline one by one.

The principle of the pipeline is Eqs. (10) and (11). But the equation only can pipeline the calculation within one sub-window. We have added a feedback mechanism (subtract the $II(0,j)$ by $II(i,j)(0 < i \leqslant 23)$ in the proper order) to pipeline the calculation between different sub-windows.

Before the pipeline starts to work, the registers $II(i,j)$ and $Bufj$ are initialized to 0. In the *1*st clock, the register $II(23,0)$ and $Buf0$

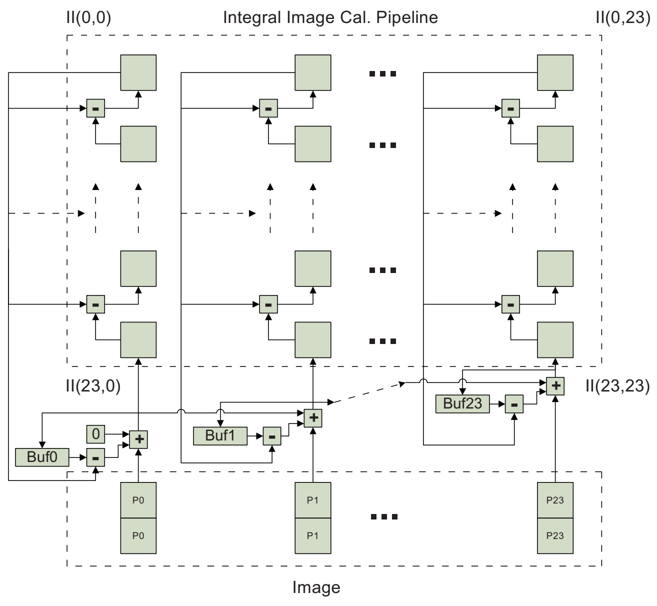


Fig. 6. II calculation pipeline circuit.

are set to value $P_0 + Buf0 - II(0,0)$. In the 2nd clock, first the $II(22,0)$ is set to value $II(23,0) - II(0,0)$,$II(23,1)$ and $Buf1$ are set to $P_1 + Buf0 + (Buf1 - II(0,1))$, then $Buf0$ and $II(23,0)$ are updated by value $P_0 + Buf0 - II(0,0)$. Generally, In each clock, first, the content registers value will move up one step after subtracting the value of the first register in each column $(II(i,j) = II(i+1,j) - II(0,j))$, then update the value of the last register in each column and buffer register of each column by the image pixel value, buffer register value of each column and left neighbor column $(II(23,j) = Buf(j-1) + (Bufj - II(0,j)) + P$ and $Bufj = Buf(j-1) + (Bufj - II(0,j)) + P_j)$. Just as this operation order, After the 24th clock, the $II(i,0)(0 \leqslant i \leqslant 23)$ values are ready for output to II buffer. After the 48th clock, the II value for a 24×24 sub-window is worked out. From the 49th clock, the pipeline can work out the II of a new sub-window in each clock.

3.3. Feature ROM

The feature ROM is used to store the Haar features which are used to detect the face. The organization of feature ROM is shown in Fig. 7. The ROM is organized into a linear structure. For each rectangle, the x and y stand for the coordinates of the top left point, the three other points are: $(x + width, y)$, $(x, y + height)$, $(x + width, y + height)$. The n, m and k stand for the stage number, the feature number in a stage and the number of rectangles of a feature. In this implementation, we use the classifier parameters from the Open Computer Vision Library (OpenCV), which trained a strong classifier with 25 stages and 2913 weak classifiers. In each stage, the number of Haar feature classifiers is: 3, 8, 17, 25, 33, 44, 50, 51, 56, 61, 75, 93, 101, 117, 125, 137, 150, 169, 177, 182, 201, 213, 229, 269, and 327.

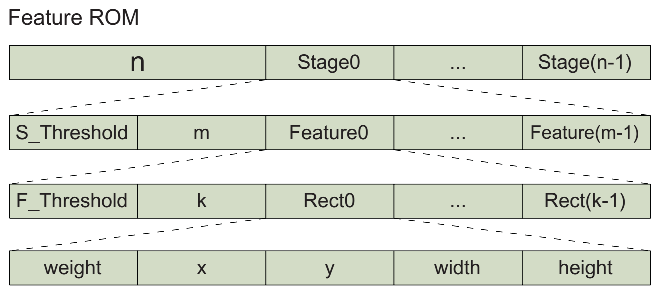


Fig. 7. Feature ROM organization.

3.4. Face detection circuit

The face detection circuit is the most important component in our system. The basic function of this component is to use the Haar feature based cascade classifier to detect whether the sub-window includes a face or not. Fig. 8 gives the details of our implementation. Since in our design, the integral image is generated in pipeline mode, we can carry out the detection work for more than one sub-window concurrently to accelerate the detection process. In Fig. 8, note that the two pieces of detection logic with dotted line frames are exactly the same. The reason we draw it into two parallel parts is to emphasize the concurrent pipeline detection process. Of course, if we have larger FPGA chips in the future, we can also implement it in two parts to get the parallelization speedup. When the first sub-window integral image data is finished the first stage detection and the second sub-window integral image data will feed to the detection pipeline and do the first stage detection work.

The data transferred between the feature ROM and detection logic include weight (W), feature threshold (FT) and stage threshold (ST) data. When the detection process begins, it will send the rectangular coordinate information of the first feature in the first stage to the integral image buffer. When the integral image buffer receives the point coordinate data, it will send the integral image value to the detection logic which will carry out the Haar feature calculation process. The Haar feature value will be multiplied by the feature weight and compared with the feature threshold. If it is larger than the threshold, the value will be added to the stage feature value buffer; otherwise the feature value is ignored. After all features are examined and the stage feature value accumulated, that value is compared with the stage threshold. If it is larger than the threshold, it means the sub-window may contain a face and will enable the second stage feature detection process. If the value is smaller, it means that there is no face in the sub-window and the Haar feature detection process for this sub-window is finished. If a sub-window passes all 25 detection stages, it will be tagged as having a face in it.

Our detection circuit implementation has advantages over the original cascade structure. First, different sub-windows can be operated on simultaneously so there is parallelism at the task level. Second, different Haar classifiers in the same stage can be checked concurrently so there is parallelism at the instruction level. Third, the feature data read from the feature ROM can be used by more than one process so there is data parallelism. Finally, the pipeline structure between different detection stages also accelerates the detection process.

4. Resource usage and performance evaluation

In this section, we first introduce the system resource usage. After that we justify the image reduction algorithm and the image reduction stop threshold by experiment result. Finally, we give the comprehensive performance result of our system.

4.1. Resource usage

The development tools used for this system implementation are ISE 12.1 by Xilinx. The Xilinx evaluation and development platform, XUPV5-LX110T, has been used.

The resource usage of different parts in our face detection system is shown in Table 2. The image buffer (Image Buf.) and integral image buffer (II Buf.) consume 10,338 and 22,168 slice registers

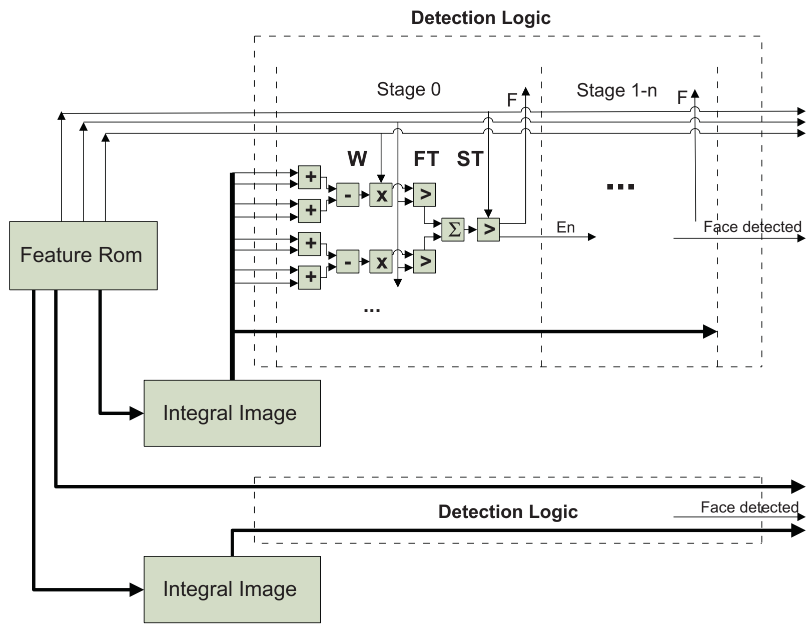


Fig. 8. Face detection circuit.

respectively. This register usage accelerates the data access time since the whole or partial Image Buf. or II Buf. data need to be accessible simultaneously. The slice LUTs usage of a component can show its corresponding logical complexity. Since the image reduction algorithm enjoys a high degree of parallelism, we have implemented 10 identical image reduction circuits to speed up the image scaling process; this uses 10,068 LUTs. The feature ROM including all the 2913 weak classifiers is constructed with BRAM. The capacity of each BRAM is 36*Kb*. In the output control module, the DSP48Es are used to quicken the detected face coordination calculation process by the coordination of the reduced image and the reduction count number. In summary, the system takes up 81.1%, 47.8%, 58.7% and 18.7% of on chip registers, LUTs, BRAM and DSP48Es respectively.

4.2. Image reduction algorithm contrast

It is hard to justify the advantage of the area average algorithm against other algorithms only by theory. We have made some experiments to justify our selection. Though nearest neighbor, bilinear and bicube algorithms are suitable for enlarging an image (generally, the bicube algorithm is better than bilinear and the bilinear is better than nearest neighbor in smoothness and sharpness), they are unsuitable for image reduction with an arbitrary scaling down factor due to the quick quality loss. We built the image reduction component with different algorithms and the resource needs are shown in Table 3. After integrating the reduction component to the face detection system, the final system frequency, detection rate and false positive/negative rate are given in Table 4. The test case is from Shanghai IsVision Sample Database. From Tables 3 and 4, though the area average algorithm is not the best resource optimization option (we know that resources are not the bottleneck of our system in the XC5VLX110T chip from Section 4.1), it supports the system to gain the best detection effect.

4.3. Image reduction stop threshold evaluation

Due to the facts that:

1. The Shanghai IsVision Sample Database is not openly available.
2. The novel image reduction process stop threshold which is proposed in this paper is based on the empirical or statistical data. If one face is detected, all or most other faces will be detected at the same image reduction level.

To make contrast and verify our assumption, we also implemented the image reduction component with the image reduction stop threshold set to the size of the subwindow (original threshold). That is to say the image scaling down and face detection process will not stop unless the reduced image is smaller than the subwindow. We use the Extended Yale Face Database B [14] (Test Case A) and Shanghai IsVision Sample Database (Test Case B) both as test cases. Tables 5 and 6 shows evaluation results for Test Cases A and B respectively. All the test cases shown our improved threshold gives a 2.2 times speedup with little detection rate or accuracy loss.

Table 2
Face detection system resource usage.

Component	Slices reg.	Slice LUTs	BRAM	DSP48Es
Input control	567	384	0	0
Output control	2063	1682	0	2
Image buf.	10,338	2086	16	0
Image reduce circuit	4903	10,068	0	0
II cal. circuit	10,917	8439	0	2
II buf.	22,168	2746	18	0
Feature rom	249	376	33	2
Face det. circuit	1964	4982	0	6
IO buf.	604	760	20	0
Whole system	56,107	33,053	87	12
Available	69,120	69,120	148	64

Table 3
Resource needs for different algorithm.

	Slice reg.	Slice LUTs	BRAM	DSP48Es
Area average	4903	10,068	0	0
Nearest neighbor	2816	5761	0	0
Bilinear	3982	7682	0	0
Bicube	5066	11,088	0	4

Table 4
System performance with different algorithm.

	Sys. freq. (MHz)	Det. speed (fps)	Det. rate (%)	False pos. rate (%)	False neg. rate (%)
Area average	150	100	90.10	0.40	0.20
Nearest neighbor	150	100	85.70	0.65	0.41
Bilinear	150	100	89.40	0.52	0.37
Bicube	120	80	89.90	0.34	0.21

Table 5
Evaluation result with different threshold for Test Case A.

	Improved threshold	Original threshold
Detection speed (fps)	100	45
Detection rate (%)	87.20	89.10
False positive rate (%)	1.70	1.66
False negative rate (%)	0.60	0.61

Table 6
Evaluation result with different threshold for Test Case B.

	Improved threshold	Original threshold
Detection speed (fps)	100	45
Detection rate (%)	90.10	91.30
False positive rate (%)	0.40	0.36
False negative rate (%)	0.20	0.20

4.4. Whole system performance

After justifying the area average image reduction algorithm and improved image reduction stop threshold, we give the system performance evaluation result below.

Since Hiromoto's work [20] has a lot of common ground with ours, we give the system performance summary and comparison with Hiromoto's in Table 7. The general detection rate between the two is roughly equal. We win in the input image resolution, output image mode, detection rate, resource usage and price aspects. The low cost with high performance character of our design gives us a good chance to enter the practical industrial market. The color image output also gives us a strong advantage compared to grayscale in real applications. One of the most important contributions in [20] is the relation between classifier bit and false positive rate, and in that area, his system outperforms ours though ours outperforms many others.

Due to the facts that the system detection accuracy (DR, false positive/negative rate given in Table 7) is from tests against the Shanghai IsVision Sample Database which is not openly available. To verify the effectiveness and commonality of our system and give more convincing evidence, we use the Extended Yale Face Database B [14] (Test Case A) as a test source and give the system detection accuracy result in Table 8. We also obtain excellent and stable detection accuracy with this source. The source of Test Case B is the Shanghai IsVision Sample Database.

Table 7
Face detection system performance summary.

	Our work	Hiromoto [20]
Platform	XUPV5LX110T	N.A.
FPGA chip	XC5VLX110T	XC5VLX330
FPGA chip price	\$1,613	\$9,313
Clock frequency	150 MHz	140–160 MHz
Input image resolution	600×800	640×480
Output image mode	Color image	Grayscale
Sub-window size	24×24	24×24
Detection rate	90.1%	≈90%
False positive rate	0.4%	≈0.002%
False negative rate	0.2%	N.A
Detection speed	100*fps*	30*fps*
Registers	56,107	63,443
LUTs	33,053	55,515

Table 8
Face detection system accuracy.

	Test Case A [14] (%)	Test Case B (%)
Detection rate	87.2	90.1
False positive rate	1.7	0.4
False negative rate	0.6	0.2

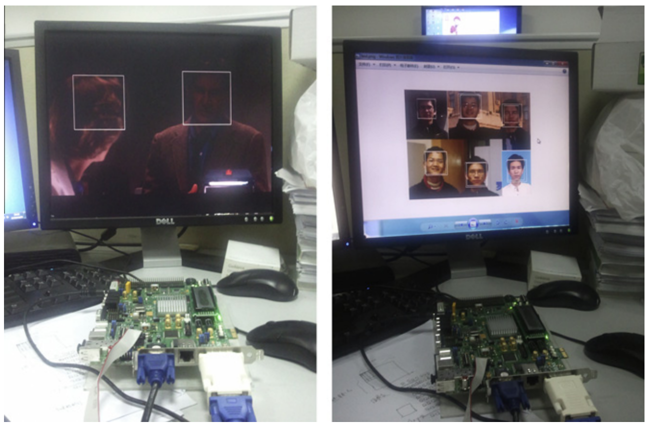


Fig. 9. Face detection result.

A state-of-the-art GPU-based (GPUs and FPGAs are widely considered to be the most promising solutions to the next generation HPC) face detection solution is [7], using four Geforce GT 220 GPUs. Our design gives better performance (100*fps* vs. 15.2*fps*) at lower cost (\$1613 vs. \$(500 × 4 = 2000)) using less power (3.5 W [21] vs. 200 W [22]). We cannot compare the detection rate and false positive rate since these are not given in [7].

The run results of our face detection system are shown in Fig. 9. The left picture is a screen shot when detecting the face in a movie video stream while the source of the right picture is some face photos which were collected from the Internet.

5. Conclusion

In this research, we have designed and implemented an FPGA-based face detection system. Through the design of a new image reduction stop threshold, the implementation of a new integral image calculation pipeline, and the optimization of the image reduction factor selection, our design achieved a significant performance improvement. Our system achieves real-time 100*fps* face detection on SVGA (600 × 800) video. On other performance indicators (resource usage, detection rate, and power saving) we also outperform previous work. In our implementation, we also introduce the color image output mode, which increases the competitiveness of our design when entering the industrial market. The Wuxi Municipal Government has purchased some copies of the system for monitoring. A new system based on the algorithms proposed in this paper but suitable for HDMI source will soon be available from Digilent Electronic Technology Co. Ltd.

Our future work in this area will mainly focus on two aspects. The first is to decrease the false positive rate. We have planned to retrain the Haar feature classifier. The second is that since we would like to transplant the face detection system into an FPGA-based portable device, we will pay more attention to the relation among the face detection rate, the detection speed and the power consumption.

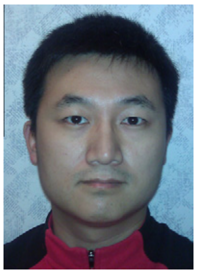
Acknowledgments

This work is supported by Shanghai Jiao Tong University Innovation Fund For Postgraduates. Special thanks to Sandy Harris for improving the English technical writing. A short version of this paper has been accepted by FCCM 2011.

References

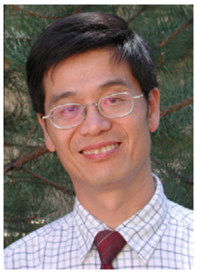
[1] Y. Ming-Hsuan, Detecting faces in images: a survey, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2002) 34–58.
[2] E. Hjelmas, B.K. Low, Face detection: a survey, Computer Vision and Image Understanding 83 (3) (2001) 236–274.
[3] P. Viola, M.J. Jones, Robust real-time face detection, International Journal of Computer Vision 57 (2) (2004) 137–154.
[4] S. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang, H. Shum, Statistical learning of multi-view face detection, in: Computer Vision ECCV 2002, vol. 2353, Springer, Berlin/Heidelberg, 2006, pp. 117–121.
[5] J. Niyoyita, Z. Tang, J. Liu, Multi-view face detection using six segmented rectangular features, in: The Sixth International Symposium on Neural Networks, vol. 56, Springer, Berlin/ Heidelberg, 2009, pp. 333–342.
[6] H. Zhu, L. Zhang, H. Sun, R. Xiao, Face detection using multi-feature, in: Advances in Cognitive Neurodynamics ICCN 2007, Springer, Netherlands, 2008, pp. 921–925.
[7] D. Hefenbrock, J. Oberg, T. Nhat, R. Kastner, S.B. Baden, Accelerating viola-jones face detection to FPGA-level using GPUS, in: 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, 2010, pp. 11–18.
[8] J. Cho, S. Mirzaei, J. Oberg, R. Kastner, Fpga-based face detection system using haar classifiers, in: Proceeding of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays, ACM, Monterey, California, USA, 2009, pp. 103–112.
[9] R. McCready, Real-time face detection on a configurable hardware system, in: Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing, vol. 1896, Springer, Berlin/ Heidelberg, 2000, pp. 157–162.
[10] S. Paschalakis, M. Bober, A low cost FPGA system for high speed face detection and tracking, in: Proceedings of 2003 IEEE International Conference on Field-Programmable Technology (FPT), 2003, pp. 214–221.
[11] W. Yu, B. Xiong, C. Chareonsak, FPGA implementation of AdaBoost algorithm for detection of face biometrics, in: IEEE International Workshop on Biomedical Circuits and Systems, 2004, pp. S1/6–17–20.
[12] G. Changjian, L. Shih-Lien, Novel FPGA based haar classifier face detection algorithm acceleration, in: International Conference on Field Programmable Logic and Applications (FPL), 2008, 2008, pp. 373–378.
[13] D. Zheng, Z. Feng, W. Tinghui, S. Wei, W. Min-You, Hecto-scale frame rate face detection system for SVGA source on FPGA board, in: 2011 IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2011, pp. 37–40.
[14] S.G. Athinodoros, From few to many: illumination cone models for face recognition under variable lighting and pose, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (2001) 643–660.
[15] T. Wang, F. Zhao, J. Wan, Y. Zhu, A novel hardware architecture for rapid object detection based on AdaBoost algorithm, in: Advances in Visual Computing, vol. 6455, Springer, Berlin/ Heidelberg, 2010, pp. 397–406.
[16] J. Allebach, W. Ping Wah, Edge-directed interpolation, in: Proceedings of International Conference on Image Processing, vol. 3, 1996, pp. 707–710.
[17] R.D. Hudson, D.I. Lehn, P.M. Athanas, A Run-time Reconfigurable Engine for Image Interpolation, 1998.
[18] M.A. Nuno-Maganda, M.O. Arias-Estrada, Real-time FPGA-based architecture for bicubic interpolation: an application for digital image scaling, in: International Conference on Reconfigurable Computing and FPGAs (ReConFig), 2005, pp. 1–8.
[19] Y. Tanida, T. Maruyama, An approach for downscaling images for real-time pattern detection, in: International Conference on ICECE Technology (FPT), 2008, pp. 265–268.
[20] M. Hiromoto, H. Sugano, R. Miyamoto, Partially parallel architecture for AdaBoost-based detection with Haar-like features, IEEE Transactions on Circuits and Systems for Video Technology 19 (1) (2009) 41–52.
[21] Xilinx, Xpower Estimator, 2010. <http://www.xilinx.com/ise/power_tools/license_virtex5.htm>.
[22] Nvidia, Nvidia Products, 2009. <http://www.nvidia.com/page/products.html>.

Zheng Ding (S'09) received the B.S. degree in Harbin Institute of Technology, Harbin, China, in 2007, and the M.S. degree in Computer Science from Shanghai Institute of Computing Technology, Shanghai, China, in 2009. Since 2009, he has been working towards the Ph.D. degree in Computer Science at Shanghai Jiao Tong University, Shanghai, China. His research lies in the fields of high performance computing, especially of FPGA based HPC.



Feng Zhao is the president of Digilent China corporation.

Wei Shu (M'90–SM'99) received the Ph.D. degree from the University of Illinois at UrbanaCChampaign. She was with the Yale University, New Haven, CT, the State University of New York at Buffalo, and the University of Central Florida, Orlando. She is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque. Her current research interests include resource management, multimedia networking, distributed systems, wireless networks, and sensor networks. She is a member of the Association for Computing Machinery.



Min-You Wu (S'84–M'85–SM'96) received the M.S. degree from the Graduate School of Academia Sinica, Beijing, China, and the Ph.D. degree from Santa Clara University, Santa Clara, CA. He is an IBM Chair Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. He serves as the Chief Scientist at the Grid Center, Shanghai Jiao Tong University. He is also a Research Professor with the University of New Mexico, Albuquerque. His research interests include grid computing, wireless networks, sensor networks, overlay networks, multimedia networking, parallel and distributed systems, and compilers for parallel computers. He has published over 150 journal and conference papers in the aforementioned areas.