HARDWARE ARCHITECTURE FOR OBJECT DETECTION BASED ON ADABOOST ALGORITHM

Hui Xu, Feng Zhao and Ran Ju
School of Microelectronics, Shanghai Jiao Tong University, Dongchuan Road, Shanghai, China

Keywords: Hardware, Object detection, AdaBoost algorithm.

Abstract: This paper implements a hardware architecture for object detection based on AdaBoost learning algorithm and Haar-like features. To increase detection speed and reduce hardware consumption, an integral image calculation array with pipelined feature data flow are introduced. Input images are scanned by sub-windows and detected by cascade classifiers. Moreover, special design is made to enhance the parallelism of the architecture. In comparison with the original design, detection speed is improved by three, with only 5% increase in hardware consumption. The final hardware detection system, implemented on Xilinx V2pro FPGA platform, reaches the detection speed of 80fps and consumes 91% resources of the platform.

1 INTRODUCTION

Cascade AdaBoost object detection algorithm, first proposed by Viola and Jones (Viola and Jones, 2001; Viola and Jones, 2004), is widely used in object detection. The algorithm builds a strong classifier by taking in a set of training images and assigning weights to a series of weak classifiers based on Haar-like features. As long as the system is well-trained, detection can be made for all concerning objects afterward. Several software realizations of the algorithm already exist, including an open-source library, OpenCV(Intel, 2009), for general development. However, due to the prevailing trend of real-time object detection, software realization cannot catch up with the requirement on detection speed. Thus embedded system with fast performance becomes an alternative. To construct an efficient embedded system for object detection, two problems should be solved. First, a reasonable mapping from software algorithm to hardware architecture with high parallelism is needed. Second, the consumption of hardware resources should be affordable to the platform. Usually, trade-off needs to be made between these two aspects.

As one of the major fields of object detection, face detection based on hardware implementation is reported in serval literatures (T. Theocharides and Irwin, 2006; H.-C. Lai and Chen, 2007; M. Hiromoto and Miyamoto, 2009). A pipelined module of feature calculation was first introduced in (H.-C. Lai and Chen, 2007). Paper (M. Hiromoto and Miyamoto, 2009) discussed the parallelism of feature calculation. However, the hardware data flow in (M. Hiromoto and Miyamoto, 2009) was still software-like and the logic consumption was large. A CDTU (Collection and Data Transfer Unit) array was proposed in (T. Theocharides and Irwin, 2006) to speed up the integral image calculation, but the architecture is full-graph based (i.e., data of whole graph need to be stored and manipulated in the array), which is unrealistic to most hardware system due to the large resource consumption.

Recently, Shi et al. (Y. Shi and Zhang, 2008) proposed an architecture with fast integral image calculation and sub-window based feature detection. Electronic system level (ESL) simulation showed that the pipelined architecture was quite efficient and the sub-window based architecture avoided large hardware consumption. In this paper, based on this novel architecture, a hardware face detection system is implemented on Xilinx V2pro FPGA platform. Special design is made to enhance the parallelism of the architecture. The parameters of classifiers used in our system originate from Intel OpenCV Library (Intel, 2009). Testing on 16000 face pictures, same accuracy is achieved in comparison with OpenCV, meanwhile the hardware detection speed reaches 80 frames per second at 100MHz clock frequency, 4 times faster than OpenCV running on a PC with 2.0GHz CPU.

This paper is organized as follows: Section 2 will review the architecture in (Y. Shi and Zhang, 2008) and discuss the hardware implementation. Then fur-

Xu H., Zhao F. and Ju R. (2010).
HARDWARE ARCHITECTURE FOR OBJECT DETECTION BASED ON ADABOOST ALGORITHM.
In *Proceedings of the International Conference on Computer Vision Theory and Applications*, pages 420-424
DOI: 10.5220/0002841204200424

ther improvements are presented in section 3. Experimental result is illustrated in section 4. Finally, a conclusion is drawn in the last section.

2 SUB-WINDOW BASED HARDWARE ARCHITECTURE

In this section, a brief review on cascade AdaBoost algorithm (Viola and Jones, 2001) will be presented. Then the implementation of hardware architecture based on sub-window detection will be introduced.

2.1 Review on AdaBoost Algorithm

An cascade AdaBoost object detection system is composed by a series of strong classifier stages. Meanwhile, each strong classifier consists of several weak classifiers with different weights. Usually, the system can be depicted as the following equations.

$$C(x) = \begin{cases} \gamma_1 & \sum_{t=1}^{T} w_t \cdot h_t(x) \geq \sigma \cdot \alpha \\ \gamma_2 & \text{otherwise} \end{cases} \quad (1)$$

$$Result(x) = \begin{cases} 1 & \sum_{t=1}^{\#stage} C_t \geq \beta \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

,where w_t is the weight of every weak classifier. h_t denotes the feature function, which can be calculated from the integral image (Viola and Jones, 2001). σ is the deviation of the concerning sub-window image. α and β are thresholds of strong classifier and stage respectively.

Unlike in (Viola and Jones, 2001; Viola and Jones, 2004), we choose γ_1 and γ_2 instead of 1 and 0 as positive and negative factor respectively. (Intel, 2009) shows that with a reasonable value of γ_1 and γ_2, the detection rate of the system will be improved. If the summation of strong classifier values is larger than the stage threshold, the concerning image passes the stage and will be sent to the next stage. Otherwise, it is rejected by the stage and no further detection will be made on this sub-image.

2.2 Hardware Implementation

Figure 1 describes the sub-window ($24 \times 24 pixel^2$) based hardware architecture. The pipeline on the top is designed to propagate the feature information stored in the ROM. The central array aims to calculate the integral image of the sub-window according

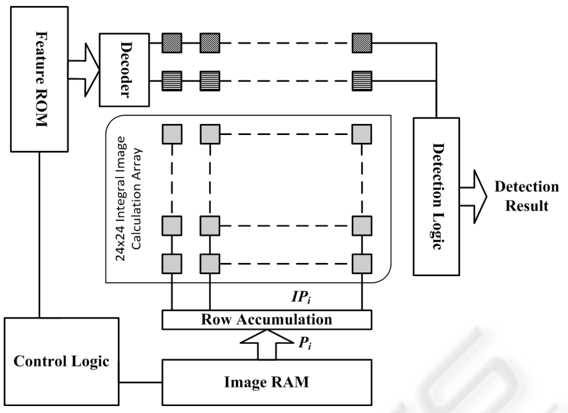
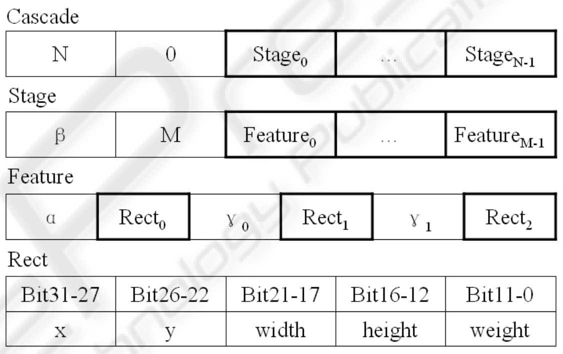


Figure 1: Pipelined hardware architecture for face detection.



Cascade

N	0	$Stage_0$	…	$Stage_{N-1}$

Stage

β	M	$Feature_0$	…	$Feature_{M-1}$

Feature

α	$Rect_0$	γ_0	$Rect_1$	γ_1	$Rect_2$

Rect

Bit31-27	Bit26-22	Bit21-17	Bit16-12	Bit11-0
x	y	width	height	weight

Figure 2: Hierarchical data structure of features.

to the data in the Image RAM. Detection logic determines whether the sub-window image passes the cascade stages.

As proposed in (Y. Shi and Zhang, 2008), the integral image calculation array, with a paralleled architecture, can obtain the integral image efficiently (only one clock cycle is needed to calculate integral image of next sub-window). The hierarchical data structure of features stored in the ROM is shown in Figure 2. The top structure *Cascade* is composed by N stages, each of which has a stage threshold β and M features. Meanwhile, each feature, described by a feature function (Eq. 1), contains three *Rects* which denote the rectangles in Haar-like features (Viola and Jones, 2001; Intel, 2009). When the data of *Rect* pass through the pipeline, each cell can switch to the integral image array according the data and calculate the summation of pixels of the *Rect* (Y. Shi and Zhang, 2008). Once filled with data, the pipeline can output sum of pixels of one *Rect* every clock cycle.

Next, as shown in Figure 3, $\sum_{Rect} P_i$ will be sent to the detection logic, which is composed by two stages, weak classifier stage and strong classifier stage. Notice that each feature contains three rectangles and

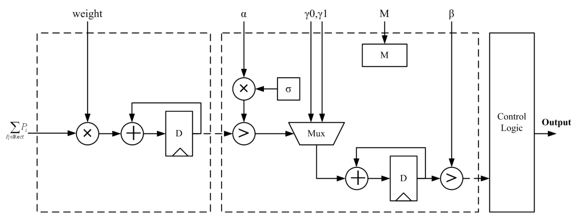


Figure 3: Object detection block.

three parameters (α, γ_1 and γ_2 which propagates with *Rect1*, *Rect2* and *Rect3* respectively through the pipeline), thus value of strong classifier (Eq. 1) needs three clock cycles to obtain. Similarly, the second stage of the detection logic is a direct mapping of Eq. 2. After checking all M features, the logic will determine whether the concerning sub-window passes this stage according to the stage threshold β. If the current sub-window does not pass, next one will be loaded to the array. Otherwise the current sub-window will be checked by the next stage of classifiers.

3 IMPROVEMENTS ON DETECTION SPEED

With the novel approach to calculate integral image and multi-stage pipeline, the architecture above can achieve fast performance compared to software realization. However, further improvements can be drawn thanks to the flexibility of the architecture.

In the above architecture, since every window will be checked to the same cascade classifiers, it is straightforward to think that if N windows can be checked at the same time, namely, multi-window detection, the performance will be improved proportionate to N. Shi et al, in (Y. Shi and Zhang, 2008), proposed that this goal can be achieved by adding $N-1$ rows to the integral image array. For instance, if one row ($N=2$) is added to the array, then row 1 to row 24 compose the first sub-window and row 2 to row 25 compose the other.

However due to the fact that total detection time of n windows is determined by the particular sub-window which passes the largest number of stages, the actual speedup rate is not proportionate to the number of additional rows. In the worst case, when two sub-window are detected simultaneously, the accelerate ratio will be only $(\#stages+1)/\#stages$ if one sub-window is rejected at the first stage while the other passes all stages. A software simulation result is shown in Figure 4. The accelerate ratio is 1.35 when $N=2$.

To address the problem in multi-window detection, multi-rectangle detection is implemented in this paper. Recall that the throughput of the pipeline is one $\sum P_i$ per clock cycle, which limits the detection speed since calculation of a strong classifier needs 3 clock cycles. As a result, if the width of pipeline is broadened to transmit n rectangles each clock cycle, the speedup ratio will be improved n times (Figure 4). Certainly, additional hardware consumption is inevitable due to the enlarged bit width of the pipeline, more complex control logic, and the new detection logic block. Thus balancing the cost and speed, this paper implements an improved architecture with $n=3$.

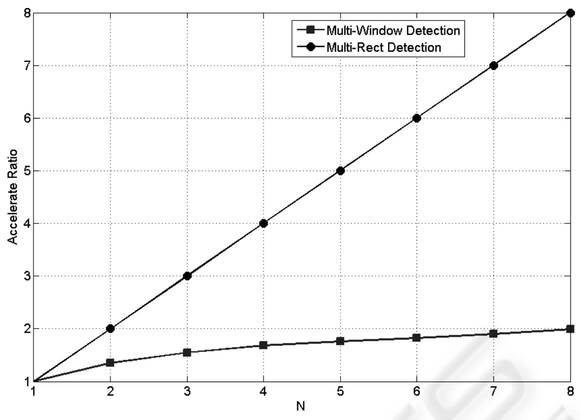


Figure 4: Accelerate ratio versus N.

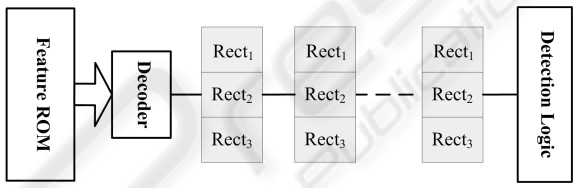


Figure 5: Improved pipeline structure.

Figure 5 depicts the new pipeline structure. The bit width is broadened three times, thus data of three *Rects* and weights can be transmitted simultaneously. Besides, a more complex control logic is designed to select integral image data from the central array to calculate the sum of pixels of each *Rect*. Control cells in the pipeline have an index, i ranging from 1 to 24, and a register, *Sum*, to store the sum of pixels within the rectangle. Meanwhile, each *Rect* has four coordinates, x_1, x_2, y_1, y_2. When one *Rect* propagates from the first cell to the last cell in the pipeline, $\sum_{Rect} P_i$ can be obtained according to the following equation:

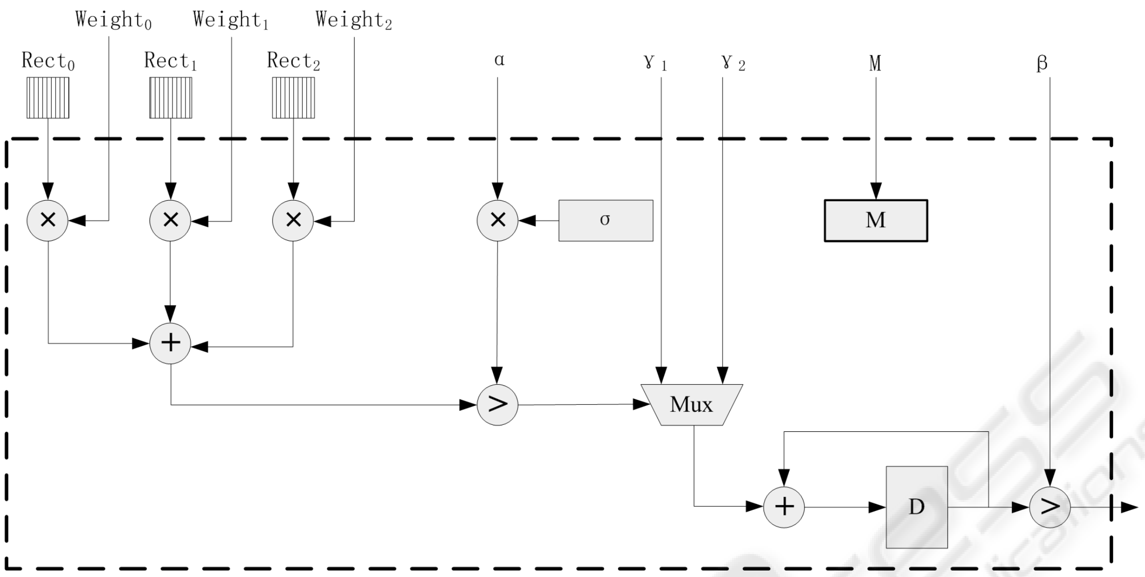


Figure 6: Improved detection logic.

Table 1: Summary of sub-window based hardware face detection system.

Method	Detection/False Positive Rate	Speed (fps)	Hardware Consumption (Xilinx V2pro)
Original detection	91.3%/3.2%	27	11765 slices/13696 slices
Multi-Rect detection	91.3%/3.2%	80	12470 slices/13696 slices

$$Sum = \begin{cases} Sum + SAT(x_1,y_1) - SAT(x_1,y_2) \\ \text{, when } i = X_1 \\ Sum + SAT(x_2,y_2) - SAT(x_2,y_1) \\ \text{, when } i = X_2 \\ Sum\text{, else.} \end{cases} \quad (3)$$

$SAT(x,y)$ means the integral image (Viola and Jones, 2001), which is stored in the central array. A multiplexer is used here to switch to the corresponding row of the array according to the y coordinates of *Rect*. Thus when three *Rects* transmitted together, the bus width of the multiplex should also be broadened.

Figure 6 shows the new detection logic. Now information of three *Rects* arrives at the same time. As a result, by adding two more multipliers, value of the strong classifier can be calculated every clock cycle. No register is needed to store the intermediate data and the speedup ratio, in comparison with the original design, reaches three.

4 EXPERIMENTAL RESULT

We have implemented both the original and the multi-rect detection system on Xilinx V2pro FPGA platform. The learning classifiers, including weights and thresholds, were constructed according to OpenCV (Intel, 2009). Same set of input images, including 5000 positive and 11000 negative sample images in CIF style, were tested on the two platform. Table 1 summarizes the comparison between the two approaches.

Since both the system is based on AdaBoost algorithm and classifiers in OpenCV (Intel, 2009), the detection and false positive rate are same to the result of OpenCV. Meanwhile the detection speed of the two approaches reaches $27fps$ and $80fps$ respectively, which is much faster than software realization. Moreover it is notable that the accelerate ratio between the improved multi-rect detection to the original one is approximately three ($80/27 \approx 3$), which is identical to the analysis in the pervious section. More importantly, the additional hardware burden of the improved architecture is only 5% of the original consumption. The total hardware consumption of the multi-rect detection system is $12470slices$, 91% of resources on Xilinx V2pro FPGA platform.

Besides, video camera and VGA display block were integrated to the whole system so that real-time face detection can be achieved. An output of our detection result is shown in Figure 7.

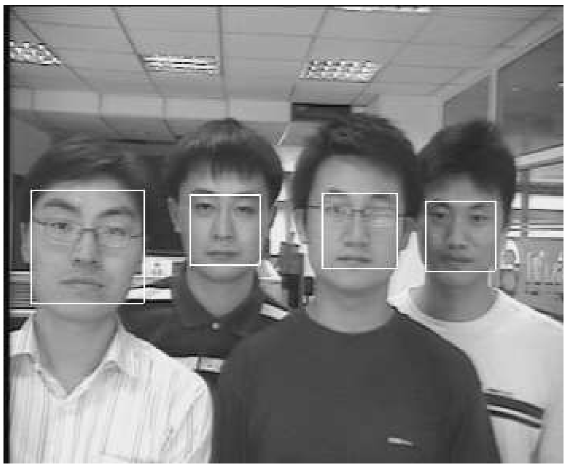


Figure 7: Result of real-time face detection system.

5 CONCLUSIONS

In this paper, an efficient object detection system is implemented. The sub-window based architecture avoids large hardware consumption. Meanwhile, the integral image calculation array and pipelined data flow improve the detection rate drastically. Due to the flexibility of the novel architecture, we further exploit the parallelism of the system. Reasonable trade-off is made to accelerate the detection with only a slight increase in hardware consumption. Implemented on Xilinx V2pro FPGA platform, the detection system runs at a speed of $80fps$ and consumes $12470slices$, 91% of the total resources on the platform.

REFERENCES

H.-C. Lai, M. S. and Chen, T. (2007). Proposed fpga hardware architecture for high frame rate ($\leq 100fps$) face detection using feature cascade classifiers. In *IEEE International Conference on in Biometrics: Theory, Applications, and Systems*, pages 1–6.

Intel (2009). Open computer vision library [online]. `http://sourceforge.net/projects/opencvlibrary/`.

M. Hiromoto, H. S. and Miyamoto, R. (2009). Partially parallel architecture for adaboost-based detection with haar-like features. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 19, pages 41–52.

T. Theocharides, N. V. and Irwin, M. (2006). A parallel architecture for hardware face detection. In *Proceedings of the IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*, volume 00, page 2.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518.

Viola, P. and Jones, M. (2004). Robust real-time face detection. In *Int. J. Comput. Vision*, volume 57, pages 137–154.

Y. Shi, F. Z. and Zhang, Z. (2008). Hardware implementation of adaboost algorithm and verification. In *22nd International Conference on Advanced Information Networking and Applications*, pages 343–346.