# An FPGA-based Object Detector with Dynamic Workload Balancing

Chuan Cheng [#1], Christos-Savvas Bouganis [#2]

[#] *Department of Electrical and Electronic Engineering, Imperial College London*
*Exhibition Road, South Kensington, London, UK, SW7 2AZ*
[1] chuan.cheng09@imperial.ac.uk
[2] christos-savvas.bouganis@imperial.ac.uk

***Abstract*—In recent years, object detection has been more frequently integrated with other vision processing functions, acting for acquisition of region of interest and is widely adopted in portable devices such as digital camera capable for automatic focusing on faces. In applications targeting those devices, limitations in both hardware resources and power supply mean an efficient utilization of hardware resource is of significance. In this paper a novel hardware architecture for Viola and Jones object detectior is proposed. The novel feature of the architecture is that it features a mechanism of dynamic workload balancing, which adaptively re-distributes the workload among available processing units, thus achieving highly efficient utilization of hardware resource. The obtained results demonstrate that the proposed system can achieve high utilisation of the dedicated resources leading to high performance over resource ratio.**

## I. Introduction

Object detection, the task of detecting the position of a specific class of objects within an image, has received a significant attention in the last few years from the scientific community due to its wide applicability in security applications, tracking, face authentication just to name a few. Many applications use object detectior for acquisition of region of interest, acting as a pre-processing module in an integrated vision processing architecture; in addition, many of those applications target battery-powered devices. The limitations in both hardware resource and power supply result in that the ratio of performance per power/hardware resource needs to be maximized. In such case the Field Programmable Gate Arrays (FPGA) platform is usually the preferred choice.

Many algorithms have been proposed for general case of object detection and also have the capability to be further specialised for a specific target domain. Such approaches include neural networks[1], sub-space projection[2] and Support Vector Machines[3] just to name a few. Many researchers have focused on the acceleration of the above methods by proposing specialized hardware architectures and exploiting the parallel nature of the above algorithms[4][5][6].

The work in this paper is based on the algorithm proposed by Viola and Jones [7], one of the most widely used algorithms for object detection, which can be easily tuned for the task of human face detection. We proposes a novel hardware architecture for mapping Viola and Jones algorithm into an FPGA that optimises the recourse utilization by taking into account the internal characteristics of modern FPGA devices. Key contribution of the work is the proposed architecture's ability to perform dynamic workload balancing among the architecture's computational units, maximizing as such the utilization of the dedicated resources. The testing results showed an improvement in performance over resource ratio by 20% to nearly 50% after optimizing the resource utilization.

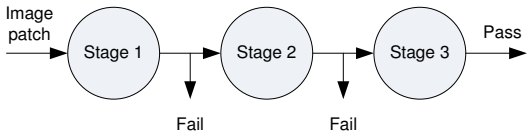


Fig. 1. Viola and Jones classifier chain

### A. Viola and Jones algorithm

In [7], Viola and Jones have proposed an algorithm for object detection, with a special application on face detection, which has been widely used by many researchers and practitioners from the image processing community. The key characteristic of the algorithm is that is based on a chain of classifiers with increasing complexity. As such, when an image needs to be classified as whether it contains an object of interest or not, only a subset of the classifiers needs to be applied. Thus, images that do not resemble a human face will be discarded early on by the system, where images that have a closer resemblance to a human face will need to be processed by more classifier stages. An image is classified to contain a human face, when it passes through all the classifiers in the chain. Fig. 1 shows a typical Viola Jones classifier chain consisting of three stages.

In each classification stage, a number of weak classifiers are applied in the input image and based on their responses the algorithm further considers the input image by advancing to the next classification stage or drops the image as it does

978-1-4577-1740-6/11/$26.00 

not contain a human face.

The training and the classification stages usually consider image patches with size $20 \times 20$ pixels. In order for the system to be able to detect faces with larger sizes, a pyramidal structure of the input image at various scales is usually considered.

## II. PROPOSED ARCHITECTURE

### A. Top-level Design

The top-level layout of the proposed architecture is shown in Fig. 2. Our proposed FPGA architecture does not cover entire process of face detection; rather it works with a host computer in the following way: The computer acquires video frame and down-scale it with scaling factor of 2 in order to detect objects with different size. The original image together with its down-scaled version will be transmitted onto off-chip memory buffer and wait for collection by the FPGA architecture; the data of workload distribution among computational units based on the previous video frame is collected and processed by host computer; the updated workload allocation will then be feedback to the FPGA architecture; in the end the coordinates of the objects detected will be collected and displayed by Host PC.

The FPGA architecture consists of a pre-processing component (which handles integral image generation (IIG) and lighting correction (LC)) , multiple processing elements (PE), a set of RAM blocks (RAM block B) together with a set of MUXs. The input frame (8 bits grey-scale image) stored in memory buffer are transmitted to pre-processing component first where a scanning window of $20 \times 20$ pixel resolution moves along the image with each movement of two pixels. Each image patch segmented by the scanning window is sent to integral image generator. The parameter of light correction is then evaluated by using the output from IIG. Each image patch has a particular integral image and corresponding parameter of light correction. These data are passed to the following PEs for further processing. Each PE is responsible for a number of stages from the classifier chain described in section I(A). Once passes the first PE, the data corresponding to that particular image patch is then transmitted to the next PE. Detection of a face is confirmed when all PEs are passed successfully. The classifier data are stored in RAM blocks B which are connected to PEs through controlled MUX.

### B. Processing Element

Each processing element (PE) comprises three components, a RAM block A, a calculating unit (CU) and a controlled MUX. The RAM block A is used to store several sets of data; each set corresponds to a particular image patch that is waiting in the queue or is currently under process by the CU, for this reason these data are called patch-related data in contrast to the classifier data which corresponds to a particular classifier chain. The content of each set includes the data of integral image (II), its corresponding parameter of light correction (LC) and the coordinates of the image patch. During the processing, the CU loads patch-related data from

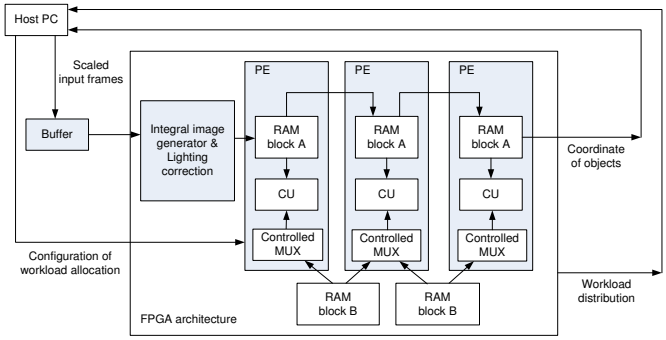


Fig. 2. Top-level architecture

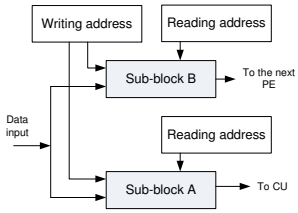


Fig. 3. RAM Block A

RAM block A as well as classifier data from RAM block B and performs the classification. The controlled MUX switches between the two RAM blocks B connected to it to ensure appropriate classifier data will be loaded.

### C. Store of Patch-related data

Detailed structure of RAM block A is illustrated in Fig. 3. The RAM block consists of two sub-RAM blocks and corresponding address generators. The write ports of both sub-RAM blocks are connected to the output of the previous PE while one of the read ports is connected to the CU and the other one is connected to the next PE, acting as an output. While receiving patch-related data from the previous PE, the data will be written to both sub-RAM blocks. The data in sub-block A will be collected by CU for classification; meanwhile the action for the data in sub-block B depends on the result of CU: if the candidate image patch passes the current PE, then the data will be sent to the next PE; if not, the data will be dropped off immediately. The reason for using two sub-RAM blocks to store the same data is that conventional FPGA devices are equipped with block RAMs with only two ports. For our design, one port is dedicated for the connection between RAM block A and CU; if using single RAM block, the remaining port must be shared by input and output transmission, in which case a bottleneck in data transmission could be generated, forcing CU to be idle while waiting for new patch-related data coming in. Using dual RAM blocks ensures that the import/export of patch-related data can happen at the same time.

### D. Store of Classifier Data

Each CU is connected with a pair of RAM blocks B through a controlled MUX. The architecture of controlled MUX is

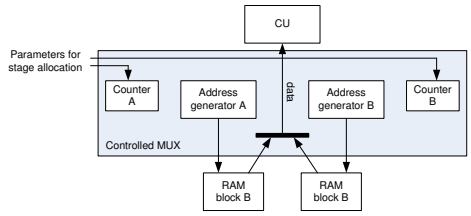


Fig. 4. RAM Block B

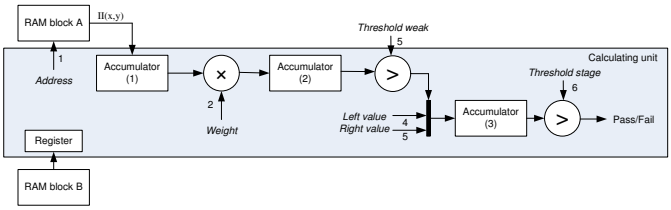


Fig. 5. Calculating unit

shown in Fig. 4. Each CU is configured to process certain number of classifier stages whose data are stored in left and right RAM blocks respectively. Because the system as a whole is a sequential system, a candidate image patch must pass the classifier stages one after another (also pass the PEs one after another); as a result a CU will start loading the classifier data in left RAM block before switching to the right one.

Two counters are used to record that how many stages belonging to the left and right RAM block respectively have been passed by the image patch that is being processed. Two registers are attached to these counters (not shown in Fig. 4) correspondingly, storing the total number of stages in the left and right RAM blocks respectively that are expected to be processed by the CU. This is how the stages allocation is configured. The parameters are determined so as to balance the workload among all processing elements. The CU starts processing an image patch by loading the classifier data in left RAM block, whenever the left counter reaches the prescribed value (a zero means starting from the right RAM block directly), the controlled MUX switches so that the CU now loads the classifier data from the right RAM block. When an image patch fails a stage, the counters and address generators will all be reset. This happens too when a patch passes all the stages allocated to the current CU.

### *E. Performing Classification*

The calculating unit collects patch-related data and classifier data from RAM block A and B respectively and performs the classification. The calculation is sequential due to the fact that the RAM block can only provide one data per clock cycle. The architecture of CU is shown in Fig. 5. The Italic notations represent the classifier data stored in RAM block B which are loaded in sequence according to the numbers shown in the figure. To perform the classification, CU first calculates the accumulation of the weighted area of rectangle features:

$$Acc = w_1 A_1 + w_2 A_2 + w_3 A_3 \tag{1}$$

where $w$ is the weight collected from RAM block B and $A$ is the area of the rectangle feature. For features containing only two rectangles, the third round of accumulation will not be performed.

The accumulated value is then compared with a weak classifier threshold $h_{weak}$ as:

$$output = V_{left}(Acc \leq h_{weak}) \tag{2a}$$
$$output = V_{right}(Acc > h_{weak}) \tag{2b}$$

where $h_{weak}$ and $V_{left/right}$ are all stored in RAM block B and are collected by CU. The output is then accumulated as:

$$Acc' = \sum_{i=1}^{n} Acc_i \tag{3}$$

where n is the number of weak classifiers contained in that particular stage. This accumulated value is finally compared with a stage classifier threshold $h_{stg}$ to decide whether the candidate image patch passes the current stage or not as:

$$result = fail(Acc \leq h_{stg}) \tag{4a}$$
$$result = pass(Acc > h_{stg}) \tag{4b}$$

## III. PERFORMANCE EVALUATION

The architectures equipped with single, dual, three and five PEs have been placed and synthesized in developing software respectively and are able to work at a maximum frequency of 200Mhz using Altera Stratix IV device. As input to the architecture, two synthetic images with $200 \times 200$ pixel resolution were used. For each testing image, eight rounds of testing were carried out to measure the latency of the system with different number of PEs before and after the workload balancing process. Given a hardware configuration (say dual PEs), the input image was passed through the detection process for the first time without deliberate stages allocation; the resulting workload distribution was then recorded, based on which the stages allocation was configured manually. This is followed by a second testing on the same input image to see whether any improvement could be achieved. As we assume that there is no significant change between adjacent video frames, it is reasonable to use the same static input image for the comparison between before and after workload balancing. As a reference point, the same images have been processed by a software implementation of the same algorithm with a computer equipped with 3.2Ghz CPU.

The testing results are illustrated in Fig.6 and Fig.7. Before the workload balancing, for the input image containing two faces, the latency based on various PE configurations are slightly faster than software version. With single PE, it is faster by around 5ms. A further improvement by adding extra PEs is not obvious. This is because with only two faces in the image plus simple solid colour background, most image patches from input image were dropped off in the early stages of the classifier chain. As a result the workloads gathered in the PEs

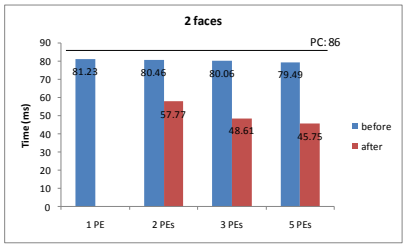


Fig. 6. Testing result for input containing 2 faces

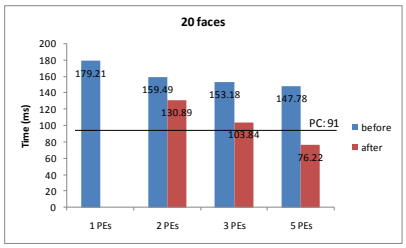


Fig. 7. Testing result for input containing 20 faces

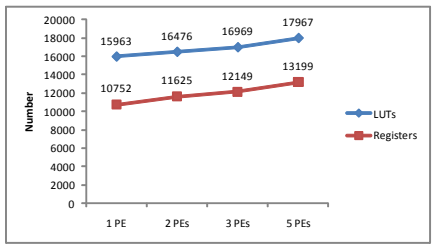


Fig. 8. Utilization of LUTs and Registers

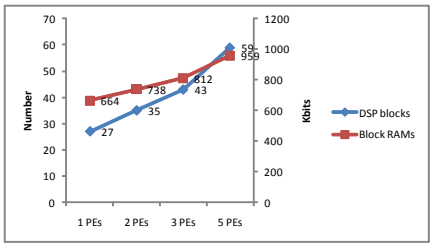


Fig. 9. Utilization of BlockRAMs and DSP blocks

handling these stages, leaving other PEs remain idle. Similarly in the case where the input image contains twenty faces, the improvement after adding extra PEs is still limited. However this time the software version outperforms the hardware versions. After introducing the workload balancing, the latencies in both cases are reduced as expected. For input image with twenty faces, hardware version with 5 PEs outperformed the software version by 14ms. The preliminary results showed that the workload balancing did work as expected.

## IV. Hardware Resource Utilization

The hardware utilization is illustrated in Fig. 8 and Fig. 9. It is difficult to compare the hardware utilization among various architectures not only because different FPGA devices have been used but also that the architectures were designed based on a unique specification such as resolution of input image, accuracy, frame rate etc. As for a reference, we chose a scalable architecture proposed by Cho et al. to compare with our architecture in terms of efficiency in hardware utilization. Cho's architecture, when equipped with single classifier and implemented on a Xilinx Vertax V FPGA, achieved a similar latency to our architecture equipped with dual PEs. As list in Table I, our design used less slices of LUTs, registers and block RAM bits. The resource consumption for a DSP-free version of our architecture is also estimated by replacing the DSP blocks with equivalent LUTs. A 20% increase in LUTs usage is expected for not using DSP blocks.

## V. Conclusion

In this paper a novel approach is introduced to design and implement a FPGA architecture for Viola and Jones object detectior that is capable for allocating the workload onto all available processing elements so as to achieve efficient utilization of hardware resource. The approach effectively relieves the problem of inefficient hardware resource utilization that commonly exists in previous designs on FPGA based Viola and Jones detector, allowing more hardware resource to be used for scalable components, leading to higher performance.

TABLE I
Comparison on Normalized Performance

| | Cho et. al. (1 classifier) | Our design (2 PEs) | Our design (without DSP) |
|---|---|---|---|
| Performance (ms) | 57.13 | 57.77 | 57.77 |
| LUTs | 62890 | 16476 | 19773 |
| Registers | 18122 | 11625 | 11625 |
| DSP blocks | 0 | 35 | 0 |
| BlockRAM (Kbits) | 756 | 738 | 738 |

## References

[1] S. Nazeer, N. Omar, K. Jumari, and M. Khalid, "Face detecting using artificial neural network approach," in *First Asia International conference on Modelling and Simulation*, Mar. 2007, p. 394.

[2] R. Gottumukkal and V. Asari, "Real-time face detection from colour video stream based on pca method," in *Proceedings on Applied Imagery Pattern Recognition Workshop*, Oct. 2003, pp. 146–150.

[3] C. Shavers, R. Li, and G. Lebby, "An svm-based approach to face detection," in *Eighth Southeastern Symposium on System Theory*, Mar. 2006, pp. 362–366.

[4] J. Cho, S. Mirzaei, J. Oberg, and R. Kastner, "Fpga-based face detection system using haar classifiers," in *FPGA '09 Proceeding of the ACM/SIGDA international symposium on Field programmable gate arrays*, 2009, pp. 103–112.

[5] M. Hiromoto, K. Nakahara, H. Sugano, Y. Nakamura, and R. Miyamoto, "A specialized processor suitable for adaboost-based detection with haar-like features," in *IEEE conference on computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[6] C. Gao and S. L. Lu, "Novel fpga based haar classifier face detection algorithm acceleration," in *Field programmable logic and applications*, 2008, pp. 373–378.

[7] P. viola and M. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, pp. 137–154, Dec. 2004.