Fast AdaBoost-Based Face Detection System on a Dynamically Coarse Grain Reconfigurable Architecture

Jian XIAO, Member, Jinguo ZHANG, Min ZHU, Jun YANG, and Longxing SHI, Nonmembers

SUMMARY An AdaBoost-based face detection system is proposed, on a Coarse Grain Reconfigurable Architecture (CGRA) named “REMUS-II”. Our work is quite distinguished from previous ones in three aspects. First, a new hardware-software partition method is proposed and the whole face detection system is divided into several parallel tasks implemented on two Reconfigurable Processing Units (RPU) and one micro Processors Unit (μPU) according to their relationships. These tasks communicate with each other by a mailbox mechanism. Second, a strong classifier is treated as a smallest phase of the detection system, and every phase needs to be executed by these tasks in order. A phase of Haar classifier is dynamically mapped onto a Reconfigurable Cell Array (RCA) only when needed, and it’s quite different from traditional Field Programmable Gate Array (FPGA) methods in which all the classifiers are fabricated statically. Third, optimized data and configuration word pre-fetch mechanisms are employed to improve the whole system performance. Implementation results show that our approach under 200 MHz clock rate can process up-to 17 frames per second on VGA size images, and the detection rate is over 95%. Our system consumes 194 mW, and the die size of fabricated chip is 23 mm² using TSMC 65 nm standard cell based technology. To the best of our knowledge, this work is the first implementation of the cascade Haar classifier algorithm on a dynamically CGRA platform presented in the literature.

key words: face detection, CGRA, dynamically, REMUS-II, pre-fetch mechanism

1. Introduction

Face detection in images and video streams has been an important research field for several decades, which has many applications in security access control, image searching, personal authentication and human computer interaction (HCI). Many face detection algorithms have been proposed and proved to be effective in the past 20 years, summarized in [1], [2], including Neural Network (NN) based method [3], Support Vector Machine (SVM) [4] and so on. In 2001, Viola and Jones [5] introduced their famous AdaBoost algorithm for real-time face detection. Their approach attracted much interest of researchers due to its amazing speed, high detection rate and low complexity. The speed (about 15 frames per second on PC platform) of AdaBoost method is the result of the many-sided utilization with simple Haar-like features, integral images and a cascaded structure. AdaBoost algorithm constructs strong classifiers with weak classifiers, and arranges them into a coarse-to-fine structure which can ejects most of non-face sub-windows at the earliest several stages quickly. Many researchers followed Viola and Jones’s approach and made lots of improvements [6]–[12].

In recent years, hardware implementations of face detection have been another hot research topic because of its high performance and energy-efficiency characteristics in mobile phone, video indexing, digital camera and so on. Several face detection algorithms have been realized on Application Specific Integrated Circuit (ASIC) [13], Graphic Processing Unit (GPU) [14], Field Programmable Gate Array (FPGA) [15]–[17] and hybrid hardware environment [18]. Among all these application platforms, FPGA is the most common choice for its hardware flexibility, shorter time-to-market and zero Non-Recurring Engineering (NRE). Most of FPGAs have rich hardware resources and are suited for mapping a whole structure directly onto it. However, FPGAs contain huge routing area and consume much more power than ASICs. Moreover, the compilation and configuration of FPGAs take much longer time. So FPGAs are commonly used as verification platform for hardware prototype and algorithms, but might not be the best choice when power and area become principal concerns.

Viola and Jones’s algorithm is effective but very computationally intensive because the face classifier has to scan the whole target image at all location and all scale [5]. It is well known that the appearance of a face in a random picture is a small probability event. So how to abandon non-face sub-windows is the key to improve the performance of a face detection system. Almost all hardware implementations of AdaBoost cascade face detection resort to a coarse-to-fine structure in which initial classifiers are simple but can discard major portion of the input image, and more complex classifiers followed detect passed sub-widows more accurately. Our statistics of a trained AdaBoost classifier implemented on FPGA are shown in Fig. 1.

More than 85% of sub-windows are ejected by first 3 stages which are composed of simple classifiers. A small part of sub-windows continue to be detected by a string of classifiers which are more and more complex, and occupy much more resources. It means that most of resource of FPGAs are “waiting” to detect sub-windows, but only a few ones might contain faces can pass through all the classifiers.
This could cause a low power-efficient result in those traditional methods. If we could dynamically construct classifiers needed, partial power could be saved. This paper proposes a novel face detection framework on a dynamically Coarse Grain Reconfigurable Architecture (CGRA), which could achieve high power-efficient and maintain excellent detection rate compared with FPGA-based method.

This paper is organized as following. In Sect. 2, the face detection algorithm based on AdaBoost, several hardware implementations and typical CGRA platforms are reviewed. In Sect. 3, the overview of our hardware platform, which is called REconfigurable MUlti-media System-II (REMUS-II), is briefly introduced. In Sect. 4 the implementation details of AdaBoost face detection algorithm on REMUS-II are discussed. Experiment results and comparisons with other implementation methods are given in Sect. 5. Conclusion and future work are presented in Sect. 6.

2. Related works

2.1 Face Detection Algorithm

1) Haar Feature
In Fig. 2(a), there are 2 examples about how Haar features represent a potential candidate of a human face. Haar features are composed of some rectangles as illustrated in Fig. 2(b). The Haar feature value is defined as the difference between the sum of pixels within white rectangles and black ones. The training process of Viola and Jones face detection algorithm is looking for Haar features of a human face to form a cascade Haar classifier. How to training a robust and rapid AdaBoost classifier is not the focus of this work, and we adopt Forward Feature Selection (FFS) [9] algorithm to obtain such one.

2) Integral Image
One of the outstanding contributions of Viola and Jones is the notion of integral image [5]. The integral image is defined as the summation of pixel values of the original image. The value at location \( (x, y) \) of the integral image is the sum of these pixels above and to the left of location \( (x, y) \) in original image as formula (1).

\[
ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')
\]  

(1)

Figure 2(c) illustrates the fast method to sum up all pixels within A in the integral image, simply as \((4+1−2−3)\).

3) Classifier
A Haar classifier uses the integral image to calculate a Haar feature value. The Haar classifier multiplies the weight of a selected feature by its feature value and then compares the result with the threshold, which is produced during the training process. Several Haar classifiers compose a detection stage. A stage accumulator sums all the Haar classifier results, and a stage comparator compares the result with a stage threshold. The thresholds and weights are all constant obtained during the AdaBoost algorithm training process, and feature values can be easily calculated from integral image.

At \( i \)th stage, the classifier response \( h(w) \) is the sum of \( n_i \) feature responses \( h_j(w) \).

\[
h(w) = \sum_{j=1}^{n_i} h_j(w), h_j(w) = \begin{cases} 1 & f_j(w) < t_j \\ 0 & \text{otherwise} \end{cases}
\]  

(2)

Where \( f_j(w) \) is the feature response of the \( j \)th Haar feature, and 1, 0 are the feature weight coefficients. When one of selected features is found \( h(w) > 0 \), the classifier allows the face candidate to pass to the next stage. Generally these sub-windows have a fixed size [6], such as \( 24 \times 24 \) pixels.

4) Cascade
The Viola and Jones face detection algorithm uses a cascade of stages as Fig. 3. The cascade ejects most of non-face candidates with earliest stages, and later stages will be much more difficult for a candidate to pass. Candidates exit the cascade if fail at any stage. A face is detected if a candidate passes all stages.
2.2 Hardware Implementations of Face Detection

Many face detection algorithms have been implemented onto several platforms. Hori et al. [13] employed a steady state genetic algorithm for high-speed hardware implementation of template matching, and designed a low power consumption ASIC core. Hefenbrock et al. [14] presented a multi-GPU implementation of Viola-Jones face detection algorithm but the power-efficiency was quite low. Junguk et al. [17] proposed a parallelized architecture of multiple Haar classifiers on a Xilinx Virtex-5 FPGA, and gained 84 times speedup over an equivalence software solution. Their system can detect about 15 VGA-size images every second. Gao et al. [18] only mapped Haar classifier function onto Virtex-5 board as an accelerator working with a host PC by a PCIe channel which could reduce data transfer time dramatically. More than 16 pipeline stages of Haar classifier were implemented in their system to achieve a speed of 98 frames per second when detecting images of 256 × 192 pixels, but no detection rate were illuminated. Their method of treating FPGA device as an accelerator could not be applied in embedded system directly. Yang et al. [19], [20] introduced a complexity control scheme to meet real-time deadline on a low cost Cyclone II FPGA. They made some approximations of AdaBoost-based algorithm, including float factor, variance etc. As a result of low speed of the device and those approximations, the detection rate of their system is relatively low (about 80%). Theocharides et al. [15] explored a parallel data access architecture in ModelSim software. They synthesized the architecture using a commercial 90 nm library at 500 MHz clock, and obtained about 52 frames per second detection speed, but no detail about the resolutions of detected images was given.

2.3 CGRA

These reported methods based on FPGA platforms belong to statically reconfigurable computing, meaning that it is configured before working, and no reconfiguration takes place during the execution process at all. On the contrary, dynamically reconfigurable devices can be reconfigured when needed. The details of development of dynamically reconfiguration could be found in survey articles [21], [22]. The key feature of reconfigurable computing is the ability to perform computations in hardware to increase performance, while retaining much of the flexibility of a software solution [23].

Several CGRAs have been proposed in the last decade. Garp [24] resembles an FPGA, a MIPS-II-like host and a 32 by 24 RA of LUT-based 2 bit PEs. MorphoSys [25] combines both coarse grain and fine grain reconfiguration techniques to optimize hardware, based on the application domain. There is a general purpose 32-bit RISC processor named TinyRisc inside MorphoSys, which can controls the sequence of operations in MorphoSys as well as executing non-data parallel operations. The PACT XPP-III [26] belongs to reconfigurable processor, and is based on a hierarchical array of coarse grain, adaptive computing elements called processing array elements (PAEs) and a packet-oriented communication network. XPP-III also contains several concurrently executing Function-PAEs (FNC-PAEs). FNC-PAEs are sequential 16-bit processor kernels which are optimized for these algorithms contain a large amount of conditions and branches like bit-stream decoding or encryption. ADRES [27] is an architecture design template from which dynamically reconfigurable, statically scheduled CGRAs can be instantiated. In one ADRES instance for multimedia applications, there are a 4 by 4 Issue Slots (IS), a unified three-issue VLIW processor that executes non-loop code, and a shared 64-entry data Register File (RF). In SmartCell [28], a set of cell units is organized in a tiled structure. Each cell block consists of four processor elements along with the control and data memories.

3. Hardware Architecture of REMUS-II

REMUS-II is designed for multimedia processing and other computation-intensive applications, and REMUS-II is a modified version of REMUS [29], [30]. The top level architecture of REMUS-II is shown in Fig. 4. It consists of one host processor ARM7TDMI, 2 Reconfigurable Processor Units (RPU), and several assistant function modules, including an interrupt controller, a Direct Memory Access Controller (DMAC), an External Memory Interface (EMI) and a Micro Processor Unit (μPU) which is suited for control-intensive tasks and is the main improvement over REMUS. All modules connect with each other by an ARM AMBA bus. RPU is a powerful dynamic reconfigurable unit and each RPU consists of a 16 × 16 Processing Elements Array (PEA) and a 32 K × 32 bit SRAM.

In Fig. 5, to explain the data access manner and configuration of RPU, we draw 2 RPUs from different views with important sub-modules inside them. In fact, the 2 RPUs are the exactly same with each other. In one RPU, the 16×16 array is subdivided into four 8×8 Reconfigurable Cell Arrays (RCA). Context Interface (CI) receives configuration words.
and reconfigures RCAs in Fig. 5(a). In Fig. 5(b) RPU can exchange data with other modules by Block Buffer (BB) interface, and RCAs shared an inside RPU Internal Memory (RIM) which is used for internal data exchange.

RCA is the minimal reconfigurable block, which is an 8 × 8 Processing Element (PE) array. Each PE adopts the common ALU architecture.

The μPU is a group of RISC processors, and this module is quite different from the previous version of REMUS [29]. The main functions of μPU are: generating the configuration word dynamically, configuration context selecting, execution of sequential/ if-then-else instructions and other tasks which are not suited for RPU, such as float point operations. With the extended ability of μPU, the partition between hardware and software could have more choices. In our face detection system, μPU works as the key supervisor module. And μPs communicate with each other by a simple mailbox mechanism. When one mail arrives, an interrupt will inform corresponding μP to check the mail and handle it. The mailbox mechanism is very similar to mailbox communication in an Operation System (OS).

There are several kinds of hierarchical memory in REMUS-II. A fast on-chip internal scratch pad memory (SPM) SRAM can hold important data temporarily. There is an Exchange Memory (EM) for the data swapping between 2 RPUs. Each RCA can access input data from an input FIFO and store results to an output FIFO. These FIFOs are all 256-bit in width and 32 in depth, and provide a high throughput for RCAs. There is also a RIM in each RPU, where middle data can be stored by output FIFO and accessed by RCAs rapidly. The RIM can also access data in DDR SDRAM through EMI. In order to accelerate the data flow, the block buffer is designed to cache data in them. Configuration operations of input FIFO, output FIFO and RCA are also independent.

The CI of each RPU is responsible for receiving and buffering configuration words sent from the μPU. The 8 × 8 RCA only can be reconfigured as a whole by the CI. Configuration words, also called context here, can be dynamically pre-fetched and cached by a configuration storage hierarchy. The off-chip DDR provides a global shared storage of the configuration data needed. The on-chip configuration data cache is used to accelerate configuration operation by maintaining recently used context. The register files exist in each RCA where configuration data is set to be active. The configuration process is divided to three stages: the configuration word generating stage, the context group pre-fetching and sending stage, and the context kernel remapping stage, and these 3 stages can also be pipelined.

4. Implementation of AdaBoost Face Detection on REMUS-II

As mentioned in Sect. 2.1, a cascade face detection system is composed of several stages (strong classifiers), and each stages consists of several weak classifiers. Weak classifier is the basis of the cascade structure. Each weak classifier needs several parameters of a Haar feature inside a 24 × 24 pixels rectangle to calculate a feature value, and compares it with the weak classifier’s threshold, then give a result of pass or rejection finally. Mapping the cascade classifier onto a fine grain FPGA chip is not too hard because of its abundant computing resource. However, mapping the long chain of classifier onto a CGRA platform is a challenge work, because it involves reconfiguration arrangement, data access and exchange, task schedule etc. Although these jobs could be done by a reconfiguration compiler, but manual designation could achieve better performance because here are so many control-intensive flow. We treat the whole work as a multi-task object, and transform the algorithm in order to dig the parallelism of the execution and REMUS-II’s potentiality. Main factors are considered as following parts.

4.1 Sub-Window Integral Image

Integral image is a result of transformation of original image pixel’s value as explained in Sect. 2.1. But with the resolution of target image are getting higher and higher, figures in integral image and square integral image (used to calculate the variance of a sub-window) are getting bigger. Sooner or later, a 32-bit space could no longer hold a data of an integral image. Moreover, because the internal memory consumes much power and occupies large size, we can only have small on-chip memory hold the most important data to maintain high performance, where an integral image of a big picture could not be saved. So we only fetch a slice of image into internal memory of REMUS-II once for calculating the integral image of a sub-window.

When the detecting rectangle moves away (Fig. 6), the data of one new column will be fetched to replace the one moved out. In this way, one sub-window size memory can satisfy the need of selecting data from integral image, regardless of the resolution of input images. Of course this method would increase computation of integral image which need to be computed only once in original way, but pipelined execution and data pre-fetch mechanism could hide the processing time.
4.2 Sub-Window Grabber and Data Selecting

Because the internal memories of our system are disperse and small, only the most important data could been stored in them. Original image remains in DDR out of the chip, and part of them would be fetched into RPU’s internal memories when needed. EMI and RCA External Data Transfer (REDT) handle them in a special 2-D manner into RIM which can be read by RCAs. The 2-D reading manner is very useful for copying a special rectangle part from target image, which happens quite frequently in image processing. As in Fig. 7, original data of an image are stored in off-chip DDR memory continuously. While a block of data in a specified rectangle are needed to be grabbed, REDT module will access the DDR through EMI, and combine the disperse pieces of data into a continuous part and store to the internal memory. The REDT module is reconfigurable, and the row length, jump length, height, the target address etc. can be reconfigured rapidly.

In those implementations of face detection on fine grain reconfiguration platform, how to get data from an integral image is one of the biggest bottlenecks, which is limit by throughput and the selecting way.

In our architecture as shown in Fig. 7, there is a Data Selecting Module (DSM) in each RCA, whose main function is to pick up data for subsequent calculation. The RIM is fabricated by a group of registers, so DSM can pick up many data from different location and write to Exchange Memory (EM) in order. The data selecting operation can be done in 1 cycle. The location information of selecting process comes from the configuration context produced by

4.3 Update of Sub-Window Integral Image

If a whole integral image of detected photo could be stored in internal memory, there will be no necessary to update it. But to reduce the area of internal memories used, we only keep a sub-window integral image in RIM. So it is needed to update the content of the RIM according to the position of the detection rectangle, as illustrated in Fig. 6.

Except when the detection rectangle reach the bottom and will be restart from the top of the image, the differences of 2 adjacent sub-windows are only 2 lines. The new sub-window integral image can be easily obtained by 2 steps as in formula (3).

\[ii'(x, y) = ii(x, y) - ii(x, y_0), 0 \leq x \leq 23, 0 \leq y \leq 23\]

\[ii''(x, y) = ii'(x, y) + ii'(x, y_24), 0 \leq x \leq 23, 0 \leq y \leq 23\]

(3)

Here \(ii(x, y_0)\) denotes the line which will be moved out of the RIM, and \(ii(x, y_24)\) denotes the new line that will be added in. \(ii(x, y)\) denotes the old integral image, and \(ii'(x, y)\) denotes the new one.

By the simple 2 steps, a sub-window integral image can be updated rapidly. Addition and subtraction are the basis functions of RCA and these update steps can be either pipelined or parallelized.

4.4 Mapping of Classifier

Simple/weak classifiers are the basis of cascade face detection system and consume most of computing time. It is the critical part of accelerating a cascade classifier. A coarse-to-fine structure is commonly used to throw away most of non-face sub-windows in earliest stages. Each stage contains 7-200 simple classifiers in our trained cascade. We chose 5 basic features to form those classifiers showed in Fig. 2 (b). From the view of calculation, 5 basic features can be classified into 3 types of hardware structures as Fig. 8.

In Fig. 8 a-i denote input data from previous procedure of data selecting, and \(v\) denotes the feature value output which will be compared with a threshold to determine the
current sub-window to be a face candidate or not. There are also some constant numbers in formulas, such as 4, −2 etc., which can be read from global constant memory.

Mapping so many regular small structures onto a large FPGA device is not difficult with convenient Integrated Development Environment (IDE) and Hardware Description Language (HDL). But it is not so when facing a novel CGRA platform. Although a compiler [31] has been developed together with the REMUS-II hardware, but manual designation could achieve better performance relatively. Figure 9 gives a mapping example to achieve high utilization rate of an 8 × 8 RCA.

How to mapping many classifiers onto a limited hardware source is a knapsack problem. To make the best use of our hardware resource, we design the mapping templates by the cascade parameters. Because a sub-window within an image has no computing relationship with others, the system can detect as many as possible sub-windows concurrently. The other important thing should always be kept in our mind is the data of the data selecting process and the detecting process need to be synchronized.

4.5 Post-Process

After simple classifiers figure out the feature values, several comparison and addition operations will be done to determine whether the current sub-windows pass the stage or not. If the sub-window passed all stages, the coordinate will be saved as a face area.

In our approach, to detect all possible faces of all sizes, the target image will be resized by a factor 1.25 until it is the same size of training face sample (normally in 24 × 24 pixels). The operation to resize could be done with the detection process meanwhile by different μPs.

After all images of all sizes have been detected, those remained passed face sub-windows will be merged, because some of them are the reduplicate results of a same face. When all these have been done, a white rectangle will be added into the original image to show the detection results.

4.6 Schedule and Synchronization

Now we give the most important notion of our method. All the processes analyzed above could be done in any platform respectively, but merging them together into a practical system is not an easy work. We regard these processes as tasks, and one task is always under the control of a μP. There is a mailbox mechanism between μPs, which is briefly illustrated in Sect. 3. When a process has been done, for example, a group of data has been selected and stored in EM, a mail will be sent to the μP1 who are watching at the process of computing feature values. Then μP1 will handle the mail when hardware resources are still not all occupied. A brief explanation will be given in Fig. 10 in which some hardware modules are omitted for brief.

As Fig. 10 describes, the whole face detection system is divided into 4 tasks and communicate with others by means of mailbox. Here SPM is a fast internal memory where middle data and parameters of classifiers could be stored in. EM is another internal memory which is used for the data exchange between RPU0 and RPU1. DDR is a big external memory where original image and rescaled images are hold.

One task will be executed by a μP or a μP with a RPU. The resizing task can be done by μP3. When μP3 finished a job to resize one image, a mail contains the address of resized image in DDR will be sent to μP0. When all resizing jobs have been done, a final mail will be sent to μP0. μP0 is responsible for supervising the data selecting task. When a mail from resizing task comes, that means a scaled image is ready and the data selecting task could begin. The data selecting task is very significant for subsequent detecting task for their content must be synchronized precisely. Data selecting task should be executed before detecting task, but detecting information should be shared between the 2 tasks, because the types of classifiers, positions and sizes parameters of features need to inform the DSM (Fig. 7 (b)) to select the right data out. As the data from DDR are cached in corresponding buffer, the latency of reading a line of a image or 8 position parameters of a feature could as short as 1 cycle. 4 RCAs in RPU0 could work at the same time with different priorities, so we let them work on 4 sub-windows at different column. EM has been partitioned into several blocks to hold the output from different RCAs of RPU0. When
one RCA finishes a data selecting job, \( \mu P_0 \) will post a mail which contains the number of RCA and the number of stage to the detecting task.

The detecting task is the most important and intricate part of our work. The mails from data selecting task contain the number of RCA which indicates the address in EM where the selected data have been stored in, and number of stage which indicates the way of reconfiguration of a RCA. The structure of a strong classifier is fixed after the training process, and a relatively complicated stage contains over 150 features which could not be mapped into a RCA at the same time. We counted the proportions of 3 types separately in the all strong classifier. According to the result, we designed 30 mapping templates among which 17 strong classifiers can all find the most suitable one for their reconfiguration. So the mapping templates could be used to reconfigure corresponding RCA expediently. Here we regard a detection process of a strong classifier as a phase, no matter it takes how many reconfiguration operations.

As Fig. 11 illustrates, \( \mu PU \) takes an important role in detecting task. When a mail arrives, \( \mu P_1 \) parses the mail and get the number of stage, then configuration words of several templates, which could be cached in FIFO Write Channel or be stored in DDR out of the chip, will be sent to CI of RPU1, and a reconfiguration operation will be launched. With the pre-fetch mechanism of Context Group Control Unit (CGCU) and 3-stage pipeline of FIFO write channel, one configuration operation only consumes 4 cycles on average. The configuration of a RCA includes the configurations of the way data come in, RCA structure, and the way output data are stored into SPM. In the “worst” case, if a sub-window passed all the stages, the configuration operation will be executed 330 times in 17 phases totally.

However, in original cascade structure, there are a lot of judgments in each stage, which is an obvious drawback that could not be neglected. If we suspend the whole pipeline waiting for a phase’s result and then reconfigure a RCA to continue a next phase, the performance will be quite poor. Perhaps, it is the main reason that cascade face detection algorithm with so many if-then-else instructions is seemed to be unsuited for CGRA implementations. To alleviate this, we move judgments out of the detecting task and treat a phase as a smallest cell in our approach. Here is a case about the novel mechanism of phase handling. When all feature values of a sub-window \( x \) at stage \( y \) have been figured out and by detecting task, a mail will be sent to the final post-process task. At that time, the phase \((x, y)\) is over, and RCA can process the other phase regardless of whether sub-window \( x \) passed stage \( y \) or not. The final task will take the mail, calculate the sum of feature values, and a conclusion will be drawn. At the same time, the detecting task is not suspended, because the mailbox is comprised of a FIFO, 16 in depth, and there could be other mails from the data selecting task. If sub-window \( x \) passes the stage \( y \), a mail contains the sub-window number \( x \) and the next stage number \((y+1)\) will be sent to the data selecting task to make it to continue selecting data for the next stage \((y+1)\), that means the sub-window \( x \) will reenter the detecting task when its new phase
(x, y + 1) is scheduled some time. Since the sub-windows are all independence with each other, the order of execution of phases does not change the final result in a scaled image. When a sub-window passes a stage, the next phase might not start to work at once. Its real turn depends on the mail’s sequence of entering the detecting task.

In Fig. 12, different tasks of a cascade system are dispatched onto RPs and µPs. Clearly, the work of detecting task is driven by mails come from data selecting task. Although the mail fetching operation takes a long period time (about 40 cycles), but the fetching takes place together with detection processes, so this long period can be hidden. A smallest rectangle indicates a phase, and the numbers in them indicate the number of sub-window being detected and the stages to pass which a RCA could be reconfigured according to. The result of whether the sub-window can pass the stage will be known from the mail after a short time, in which post-process task will do judgment job and sent a mail contains passed number and new stage number to data selecting task. The selecting task selects the right data into EM, and sends mails to detecting task. All these work will be done at the same time and pipelined, so the latency could be all hidden. During these processes, the essential if-then-else instructions do not cause any extra delay. General speaking, the pipeline is driven by mails, labeled as 1⃝2⃝3⃝ in Fig. 12. Our method avoid the long period time of waiting for judgment statements which is harmful to performance.

It is certainly possible that detecting task is suspended when data selecting task is working on several complicated classifiers and no mail has be sent to detecting task for a short time, and the performance will decrease. For the reason we have elaborated in Sect. 1, such instances will not take place much.

5. Implementation Results and Comparison

Following the method of [9], we chose 5 basic Haar feature types and trained a cascade classifier with 17 stages and 2133 Haar features. We have realized the cascade classifier in ANSI C for validity simulation and mapped the cascade structure onto REMUS-II for performance and power evaluation. We use the MIT+CMU frontal face benchmark test set and 350 pictures from internet and our realistic scenarios.

Table 1

| Resolution    | 176*144 | 320*240 | 640*480 | 800*600 |
|---------------|---------|---------|---------|---------|
| Sub-windows   | 58,791  | 219,353 | 1,009,272 | 1,622,328 |
| Detection speed | 293.3 | 78.6 | 17.1 | 11 |

(fram/s)sec

As the number of sub-windows within a target image increases rapidly when the image’s resolution increases, as illustrated in Table 1, the detection speed of our system drops at the same time. Some detected results are showed in Fig. 13. It’s obvious that our face detection system is quite effective for frontal faces with a more than 95% detect rate statistically and a quite low false positive rate.

There are several instances of AdaBoost-base face detection hardware implementations in the Table 2, but no report about systems on CGRA platform has been published to the best of our knowledge. FPGA-based platforms are still the most common choices, and the main purposes are how to parallelize the algorithm, achieve better performance and maintain reasonable detection rate at the same time. In the column 2, structures of trained AdaBoost detection systems are illustrated, which are the target of hardware implementations. [32] simplified the cascade structure too much, which only contained 52 Haar classifiers, just to improve the detection speed. But the poor detection rate and false alarm rate make the method unpractical. [19] tried to realize a low-cost system and no detection speed and the resolution are reported, which are quite important. [17], [18], [33] chose the Xilinx Viertex-5 as the hardware platform and got real-time detection speed and quite good detection rate at different resolutions. Virtex-5 could afford abundant hardware resource but consume much power, which could be concluded by the coarse power estimations within the last
Table 2 Comparison with other AdaBoost-based face detection implementations.

| Reference | Structure of implementation system | Resolution (x*y) | Accuracy on average (%) | Detecting speed (frames/sec) | Main information of platform | Power (W) |
|-----------|------------------------------------|------------------|-------------------------|-----------------------------|-----------------------------|-----------|
| [17] | 22 stages, 2135 Haar classifiers | 320*240          | NA                      | 60                          | Xilinx Virtex-5 LX330, 21270 registers bit, 73741 LUTs, 53 BRAMs, 28 DSPs, no information about work frequency | 4.5       |
| [18] | 40 stages, 2192 Haar classifiers | 256*192          | 87.6                    | 98                          | Xilinx Virtex-5 LX330T, 95%LUT, 30%BRAM, 33%DSPs, at 125MHz | 3.8       |
| [19] | 11 stages, 140 Haar classifiers | NA               | 80                      | NA                          | Altera Cyclone II, 45K Gates, 22KByte RAM, at 50MHz | 0.3       |
| [32] | Only 52 Haar classifiers          | 640*480          | 86 (high false alarm rate) | 143                         | Xilinx Virtex-II Pro XC2VP30, 84% slices, 76% LUTs, 32% BRAM, at 126.8MHz | 0.9       |
| [33] | 25 stages, 2913 Haar classifiers | 640*480          | >90                     | 30                          | Xilinx Virtex-5 LX330T, 55515 registers bit, 63443 LUTs, at 169.9 MHz | 5.1       |
| Our     | 17 stages, 2133 Haar classifiers | 320*240          | 95                      | 78.6                        | REMUS-II at 200MHz          | 0.194     |

Although almost all papers did not provide the precise power of their hardware system, it is true that the FPGAs consume much more power. That could be a big drawback for FPGAs to be appropriate embedded application platforms. The manner of static reconfiguration FPGAs adopted could not save any power during processing, and the most parts of detection system are idle as the description by Fig. 1. CGRA platform is much more flexible than FPGA. Classifiers are reconstructed only when needed. On REMUS-II, our AdaBoost-based face detection system achieves a high detection rate, real time detection speed on 320 x 240 images and quite low power consumption at 200 MHz.

The REMUS-II was implemented into 23.7 mm² silicon by TSMC’s 65 nm logic process with a 400 MHz maximum working frequency.

6. Conclusion

We have proposed a framework of AdaBoost-based face detection system on a novel CGRA device, namely REMUS-II. To the best of our knowledge, it is the first report about AdaBoost-based face detection system implemented on a dynamically reconfigurable platform. Comparing with implementations on FPGA and ASIC, our design partitions a continuous cascade face detection system into discrete phases and reconstruct necessary classifiers only when needed to save partial power. We also move if-then-else instructions out of detecting process to keep the whole flow successive. Every phase needs to pass 3 tasks: data selecting task, detecting task and post-process task. Tasks are scheduled by µPs, and mails are used to communicate between tasks. Configuration cache and data pre-fetching mechanisms increase the performance enormously, and keep the three tasks to be well pipelined. Results show that our method achieves high detecting rate and low false positive rate on MIT+CMU frontal face data set. Our method on CGRA platform could be extended to applications of other algorithms. We will try a hybrid face algorithm and optimize the REMUS-II in memory hierarchy to achieve better performance in the future work.

Acknowledgments

This work was supported by the National High Technology Research and Development Program of China (863 Program) (grant no.2009AA011700).

The authors would like to thank to C.MEI, B.LIU, JJ.YANG, YC.LU, YQ.FAN, GG.GAO, H.LEI and CX.ZHANG for their helpful discussions and technical support.

References

[1] W. Zhao, R. Chellappa, P.J. Phillips, and A. Rosenfeld, “Face recognition: a literature survey,” ACM Computing Surveys, vol.35, no.4, pp.1–67, Sept. 2003.
[2] M.-H. Yang, D.J. Kriegman, and N. Ahuja, “Detecting faces in images: a survey,” IEEE Trans. Pattern Anal. Mach. Intell., vol.24, no.1, pp.34–58, Jan. 2002.
[3] H. a Rowley, S. Baluja, and T. Kanade, “Neural network-based face detection,” IEEE Trans. Pattern Anal. Mach. Intell., vol.20, no.1, pp.23–38, Jan. 1998.
[4] E. Osuna, R. Freund, and F. Girosi, “Training support vector machines: an application to face detection,” Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition, pp.130–136, Puerto Rico, June 1997.
[5] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition, vol.1, pp.511–518, Kauai, Hawaii, Nov. 2001.
[6] D.D. Le and S. SATOH, “A multi-stage approach to fast face detection,” IEICE Trans. Inf. & Syst., vol.E89-D, no.7, pp.2275–2285, July 2006.
[7] R. Lienhart and J. Maydt, “An extended set of Haar-like features for rapid object detection,” Proc. International Conf. on Image Processing, Rochester, New York, USA, vol.1, pp.900–903, Sept. 2002.
[8] P. Viola, M.J. Jones, and D. Snow, “Detecting pedestrians using patterns of motion and appearance,” J. Computer Vision, vol.63, no.2, pp.153–161, Feb. 2005.

[9] J. Wu, S.C. Brubaker, M.D. Mullin, and J.M. Rehg, “Fast asymmetric learning for cascade face detection,” IEEE Trans. Pattern Anal. Mach. Intell., vol.30, no.3, pp.369–382, March 2008.

[10] B. Wu, H. Ai, C. Huang, and S. Lao, “Fast rotation invariant multiview face detection based on real AdaBoost,” Sixth IEEE International Conf. Automatic Face and Gesture Recognition, pp.79–84, Seoul, Korea, May 2004.

[11] S.Z. Li and Z. Zhang, “FloatBoost learning and statistical face detection,” IEEE Trans. Pattern Anal. Mach. Intell., vol.26, no.9, pp.1112–1123, Sept. 2004.

[12] R. Fergus, P. Perona, and A. Zisserman, “Object class recognition by unsupervised scale-invariant learning,” IEEE Computer Society Conf. Computer Vision and Pattern Recognition, vol.2, p.264, Madison, Wisconsin, June 2003.

[13] Y. Hori, M. Kusaka, and T. Kuroda, “A 0.79-mm2 29-mW real-time face detection core,” IEEE J. Solid-State Circuits, vol.42, no.4, pp.790–797, 2007.

[14] D. Hefenbrock, J. Oberg, N.T.N. Thanh, R. Kastner, and S.B. Baden, “Accelerating Viola-Jones face detection to FPGA-level using GPUs,” 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, pp.11–18, Charlotte, USA, May 2010.

[15] T. Theocharides, N. Vijaykrishnan, and M.J. Irwin, “A parallel architecture for hardware face detection,” Proc. IEEE Comput. Soc. Annu. Symp. VLSI: Emerging VLSI Technologies Architectures, pp.452–453, Karlsruhe, Germany, March 2006.

[16] J. Cho, S. Mirzaei, J. Oberg, and R. Kastner, “Fpga-based face detection system using Haar classifiers,” Proc. ACM/SIGDA international symposium on Field programmable gate arrays, p.103, Monterey, California, USA, Feb. 2009.

[17] J. Cho, B. Benson, S. Mirzaei, and R. Kastner, “Parallelized architecture of multiple classifiers for face detection,” 20th IEEE International Conf. Application-specific Systems, Architectures and Processors, pp.75–82, Boston, MA, USA, July 2009.

[18] C. Gao and S.-ien Lu, “Novel FPGA based Haar classifier face detection algorithm acceleration,” 2008 International Conf. Field Programmable Logic and Applications, pp.373–378, Heidelberg, Germany, Sept. 2008.

[19] M. Yang, J. Crenshaw, B. Augustine, R. Mareachen, and Y. Wu, “Face detection for automatic exposure control in handheld camera,” Fourth IEEE International Conf. Computer Vision Systems, p.17, New York, USA, Jan. 2006.

[20] M. Yang, J. Crenshaw, B. Augustine, R. Mareachen, and Y. Wu, “AdaBoost-based face detection for embedded systems,” J. Computer Vision and Image Understanding, vol.114, no.11, pp.1116–1125, Nov. 2010.

[21] K. Compton and S. Hauck, “Reconfigurable computing: a survey of systems and software,” ACM Comput. Surv., vol.34, no.2, pp.171–210, June 2002.

[22] S. Vassiliadis and D. Soudris, Fine-and coarse-grain reconfigurable computing. Springer-Verlag, Berlin, 2007.

[23] R. Hartenstein, “The microprocessor is no longer general purpose: why future reconfigurable platforms will win,” IEEE International Conf. Innovative Systems in Silicon, pp.2–12, Austin, TX, USA, Oct. 1997.

[24] J.R. Hauser and J. Wawrzynek, “Garp: a MIPS processor with a reconfigurable coprocessor,” 5th IEEE Symposium on FPGA-Based Custom Computing Machines, p.12, Napa Valley, CA, 1997.

[25] H. Singh, F.J. Kurdahi, N. Bagherzadeh, and E.M. Chaves Filho, “MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications,” IEEE Trans. Comput., vol.49, no.5, pp.465–481, May 2000.

[26] P.X. Technologies, “XPP technologies XPP-III processor overview white paper,” PACT XPP Technologies White Paper, pp.1–3, July 2006.

[27] B. Mei, S. Vernalde, D. Verkest, H.D. Man, and R. Lauwereins, “ADRES: an architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix,” Field Programmable Logic and Application, vol.2778, pp.61–70, 2003.

[28] C. Liang and X. Huang, “SmartCell: a power-efficient reconfigurable architecture for data streaming applications,” IEEE Workshop on Signal Processing Systems, pp.257–262, Washington, DC, USA, Oct. 2008.

[29] M. Zhu, L. Liu, S. Yin, and Y. Wang, “A reconfigurable multi-processor SoC for media applications,” Proc. 2010 IEEE International Symposium on Circuits and Systems, pp.2011–2014, Paris, France, May 2010.

[30] T. Geng, L. Liu, S. Yin, M. Zhu, W. Jia, and S. Wei, “Parallelization of computing-intensive tasks of the H.264 high profile decoding algorithm on a reconfigurable multimedia system,” IEICE Trans. Inf. & Syst., vol.E93-D, no.12, pp.3223–3231, Dec. 2010.

[31] M. Zhu, L. Liu, S. Yin, and S. Wei, “A cycle-accurate simulator for a reconfigurable multi-media system,” IEICE Trans. Inf. & Syst., vol.E93-D, no.12, pp.3203–3210, Dec. 2010.

[32] H.-chih Lai, M. Savvides, and T. Chen, “Proposed FPGA hardware architecture for high frame rate (>100 fps) face detection using feature cascade classifiers,” IEEE International Conf. Biometrics: Theory, Applications, and Systems, pp.1–6, Crystal City, VA, Sept. 2007.

[33] M. Hiroimoto, S. Member, and H. Sugano, “Partially parallel architecture for AdaBoost-based detection with Haar-like features,” IEEE Trans. Circuits Syst. Video Technol., vol.19, no.1, pp.41–52, Jan. 2009.
Min Zhu was born in 1984. He received the B.S. degree from the Department of Micro & Nano Electronic, Tsinghua University, Beijing, China, in 2006, where he is currently working toward the Ph.D. degree in the Institute of Microelectronics. His research interests include reconfigurable computing and multimedia processing.

Jun Yang received the B.S., M.S., and Ph.D. degrees from Southeast University, Nanjing, China, in 1999, 2001, and 2004, respectively, all in electronic engineering. He is currently a research fellow and the chairman of SoC department of National ASIC system Engineering Research Center (CNASIC), Southeast University. His research interests include chip architecture design and VLSI design.

Longxing Shi received the B.S., M.S., and Ph.D. degrees from Southeast University, Nanjing, China, in 1984, 1987, and 1992, respectively, all in electronic engineering. He is currently a Professor and the Dean of Integrated Circuit (IC) College, Southeast University. His research interests include system-on-a-chip design, VLSI design, and power IC design.