SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

December 2016, Vol. 59 122402:1–122402:13
doi: 10.1007/s11432-015-5312-z

A fast face detection architecture for auto-focus in smart-phones and digital cameras

Peng OUYANG, Shouyi YIN*, Chenchen DENG, Leibo LIU & Shaojun WEI

Institute of Microelectronic, Tsinghua University, Beijing 100084, *China*

Received October 9, 2015; accepted December 2, 2015; published online March 17, 2016

Abstract Auto-focus is very important for capturing sharp human face centered images in digital and smart phone cameras. With the development of image sensor technology, these cameras support more and more high-resolution images to be processed. Currently it is difficult to support fast auto-focus at low power consumption on high-resolution images. This work proposes an efficient architecture for an AdaBoost-based face-priority auto-focus. The architecture supports block-based integral image computation to improve the processing speed on high-resolution images; meanwhile, it is reconfigurable so that it enables the sub-window adaptive cascade classification, which greatly improves the processing speed and reduces power consumption. Experimental results show that 96% detection rate in average and 58 fps (frame per second) detection speed are achieved for the 1080p (1920×1080) images. Compared with the state-of-the-art work, the detection speed is greatly improved and power consumption is largely reduced.

Keywords auto-focus, AdaBoost, face-priority, architecture, reconfigurable

Citation Ouyang P, Yin S Y, Deng C C, et al. A fast face detection architecture for auto-focus in smart-phones and digital cameras. Sci China Inf Sci, 2016, 59(12): 122402, doi: 10.1007/s11432-015-5312-z

1 Introduction

Auto-focus is widely used in consumer electronics like digital and smart phone cameras. It adjusts the distance between the lens and the image sensors to get focused images. Auto-focus is convenient for users to capture sharp images [1–3]. Since many images captured by the cameras are focused on the faces of friends or family people, the face-priority auto-focus is especially important. The face-priority auto-focus involves face detection to position the faces without user intervention.

With the development of image sensor technology, smart phones and digital cameras support more and more high-resolution images [4, 5]. For these high-resolution images, quick processing of the face-priority auto-focus with low power consumption becomes difficult. Especially for the real time auto-focus in high-resolution video shooting, the problems of processing time and power cost are more serious. On the other hand, the energy capacity of batteries for these consumer electronics increases slowly because of limitations in the battery technology; meanwhile, the general-purpose processors (generally denote the application processors in the consumer electronics) need to process more than one application. Hence,

* Corresponding author (email: yinsy@tsinghua.edu.cn)

improving the processing speed and reducing the power consumption during auto-focus are the primary concerns for achieving good user experience in digital and smart phone cameras.

Face-priority auto-focus involves face detection to get the best face regions. Yet the real time deployments of face detection on consumer electronics are difficult due to the high power computation complexity [6]. The fast face detection algorithms are proposed in [3, 7–9]. Mohammad [3] proposed a fast detection algorithm for auto-focus by combining a skin color model with the scheme of shape processing. Huang [7] adopted learning-based face detection and Zhang [8] proposed the skin color model based on the modified generalized LHS space for face detection. However, for these studies, the power consumption and the processing time will increase quickly when the image resolution enlarges. Viola [9] introduced a fast and high precision face detection method based on the AdaBoost algorithm. This method can quickly filter out the false face regions based on the cascade classification. Because of its high detection rate and robustness on complex scenes, it has been widely used in the applications of consumer electronics including navigation, smart home, and man-machine interaction [10–13]. However, the pure software processing of this algorithm on the consumer electronics is time consuming and power inefficient. Designing the application specific hardware is a valuable way to solve these problems. As the co-processor or the IP (Intelligent Property) block for the general-purpose processors in the consumer electronics, application specific hardware can be easily deployed in consumer electronics.

Many studies have explored the efficient application specific hardware architectures for the AdaBoost-based face detection algorithm [14–16]. Hanai [14] implemented the algorithm in a specific processor for face detection in portable electronics. It achieved high processing speed on small size images, yet it will be inefficient for auto-focus on the high-resolution images. Kyrkou [15] proposed a systolic array based architecture for face detections in embedded applications. Yet the maximal supported image size is limited. It cannot achieve fast face detection with low power consumption for high-resolution images in consumer electronics. Hiromoto [16] achieved partial parallel computation for the face detection. Yet the parallel processing design focuses on the first few stages of the algorithm because of consideration for hardware cost, the processing efficiency is limited for face detection in auto-focus. This work proposes an efficient face detection architecture for the face-priority auto-focus based on the face detection algorithm [9]. We research efficient design methods involving the reduction of processing time and power consumption on high-resolution images.

In the AdaBoost-based face detection algorithm, the integral and squared integral image generation, cascade classification are involved. Integral image generation including the squared integral image generation is computation and memory access intensive. Besides, the cascade classification is sensitive to the detection scenes. In different scenes, the illumination conditions, the pose of target faces targets and the number of faces are different, which will cause redundant computations in the cascade classification. In order to improve the processing speed and reduce power consumption, this work proposes a configurable array based architecture for AdaBoost-based face detection. It improves the parallel processing capability on high-resolution images by supporting the block-based integral image computation in the architecture; meanwhile, this architecture supports sub-window adaptive cascade classification to reduce redundant computations, which reduces the processing time as well as power consumption.

The rest of the paper is organized as follows. Section 2 analyzes the face detection algorithm and explores the design considerations for hardware design. The proposed hardware architecture for the face detection algorithm is described in Section 3. Section 4 illustrates the computation of the algorithm in the proposed architecture. The evaluation results are then shown in Section 5. The last section concludes the paper.

2 Analysis of face detection algorithm

The original AdaBoost-based face detection algorithm [9] flow is shown in Figure 1. It mainly includes integral and squared integral image generation, cascade classification. In addition, the computations include the feature value calculation, light compensation, and threshold comparison in each stage of the cascade classification. This section will analyze these parts of the algorithm to derive the optimization

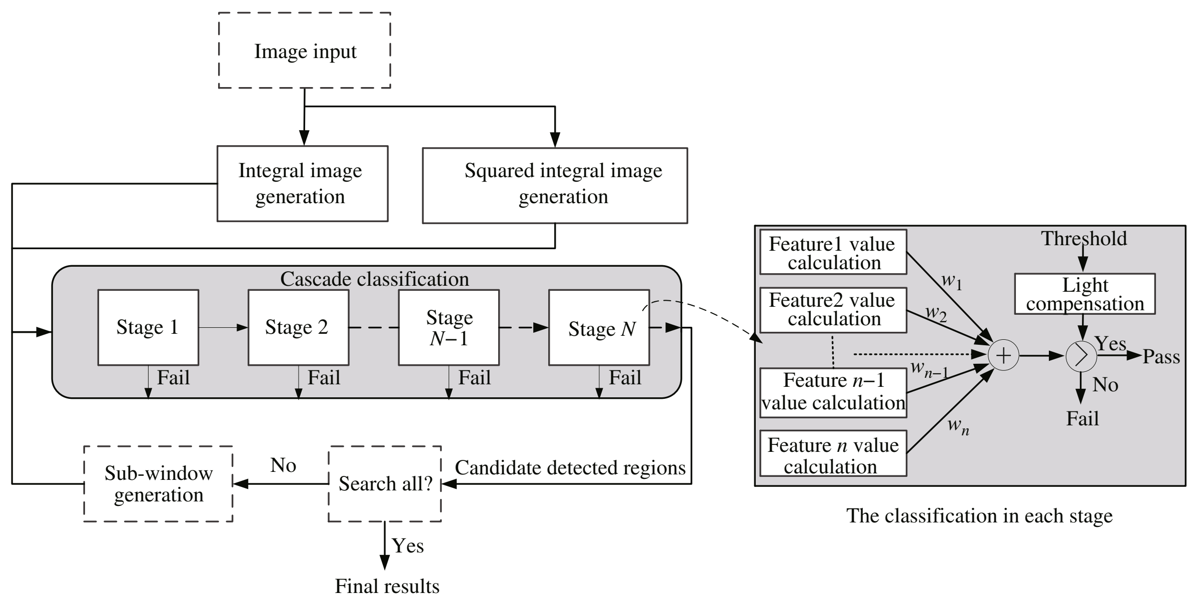


Image input
Integral image generation
Squared integral image generation
Cascade classification
Stage 1
Stage 2
Stage N−1
Stage N
Fail
Fail
Fail
Fail
Sub-window generation
No
Search all?
Candidate detected regions
Yes
Final results
Feature1 value calculation
Feature2 value calculation
Feature n−1 value calculation
Feature n value calculation
w1
w2
wn−1
wn
Threshold
Light compensation
Yes
Pass
No
Fail
The classification in each stage

Figure 1 The AdaBoost-based face detection algorithm.

considerations for the hardware design.

2.1 Integral image and feature value calculation

The algorithm adopts integral image to compute the Haar-like feature value. The Haar-like feature denotes the different regions in the face. It consists of the black rectangle and the white rectangle. The feature value equals the sum of gray value under the white rectangle area minus the sum of gray value under the black rectangle area. The feature value is computed based on the sub-window. The sub-windows are generated by scaling and shifting the search window on the image. The computation of feature value benefits from the integral image that denotes the sum of gray values located above and to the left of the location in the image. The main operations in integral image generation are the column and row accumulations. These operations can be processed in parallel on the blocks to improve the processing efficiency. In addition, the block-based integral image generation can adapt to different image sizes and reduce the stress on memory. This block method requires hardware with strong parallel processing capability and computation flexibility that can enhance the data transmission between blocks.

2.2 Cascade classification

As shown in Figure 1, the cascade classification is processed stage-by-stage; each stage achieves the identical operations. The cascade classification is suitable for pipeline processing. In each stage, the feature values are calculated and weighted to compare the light compensation threshold. This characteristic indicates that parallel computation for feature value is possible. The other benefit of the identical operations in different stages is that the hardware can be reused to process them to save the hardware resources.

In the original algorithm, the computation of feature value is based on a sub-window that is represented by the coordinates of its four corners. The sub-window generation includes the sub-window scaling and shifting (horizontal shifting and vertical shifting). The algorithm testing with 20×20 initial sub-window size on the processor (2.53 GHz) and 640×480 image shows that the step size of shifting and scaling determines the detection rate. As shown in Table 1, the sub-window parameter is denoted as the style (scaling factor, shifting step), which determines the size of the sub-window. The smaller the factor is, the higher the Detections Rate (DR) achieved; yet more sub-windows that cause more processing time are generated. The False Positive Rate (FPR) in the experiments is 0.002. DR denotes the number of correct faces against the number of true faces and FPR denotes the number of inaccurately detected sub-windows against all the detected sub-windows. According to Table 1, it can be concluded that the

Table 1 Window scaling evaluation (FPR = 0.002)

Sub-window parameter	DR (%)	Execution time (ms)	Sub-window number
(1.1,1)	97.3	2188	5898200
(1.2,1)	96.3	1591	3012400
(1.5,1)	92.4	946	1274700
(1.1,2)	95.6	1137	1474600
(1.2,2)	94.2	693	753100
(1.5,2)	89.8	367	318690

size of the sub-windows decides the classification to be coarse or fine. Hence, the sub-window adaptive consideration can adjust the computation amount and detection rate. In many detection scenes, since the image contents of the frame in the video stream are continuously changing, the original detection on each frame brings large amounts of redundant computation. The sub-window adaptive design that adjusts the size of the sub-window at runtime according to the contents of the frame stream plays an important role in balancing the detection rate and computation time.

Besides, as shown in Figure 1, light compensation is implemented for the classification threshold T_{origin} that is trained in advance by the AdaBoost-based face detection algorithm. The squared integral image supports light compensation. As shown in (1), the light compensated threshold T is equal to the product of the original threshold (T_{origin}) and the square root of the parameter C. C is implemented on the squared image and is equal to the sum of the squared integral image pixel I dividing the size of sub-window S_{size}. This process includes complex operations like division and multiplications.

$$C = \sum_{j=0}^{S_{\text{window}}} \frac{I^2}{S_{\text{size}}} \rightarrow T = T_{\text{origin}} \times \sqrt{C} \ \rightarrow T^2 = {T^2}_{\text{origin}} \times C. \tag{1}$$

3 Architecture design based on the analysis of the algorithm

3.1 Architecture overview

As shown in Figure 2(a), the algorithm can be divided into two parts. The first part includes the integral image generation and squared integral image generation. The second part is the cascade classification that needs the integral image data and the squared integral image data. These two parts are coupled using a strong loop, and their processing is characterized by the stream property. Hence, as shown in Figure 2(b), two identical RPA (Reconfigurable Processor Array) groups are designed to process these two parts in the pipeline way. The design of reconfigurable processor array has been proved very effective for the image processing [17, 18]. For these two RPA groups, each group consists of four homogenous reconfigurable processor arrays and each array contains 10×10 identical PUs (Processing Unit). The reconfigurable processor array has 20 GOPS (16-bit fixed-point Giga Operation per Second, 200 MHz) computation power. These two groups of reconfigurable processor arrays provide 160 GOPS computation power. The left group of the reconfigurable processor arrays is in charge of integral image and squared integral image generations; it supports block-based processing. The right group of the reconfigurable processors arrays is in charge of cascade classification, it supports the sub-window adaptive mechanism.

As shown in Figure 2(a), the generated data of these two parts is loose coupling, and the block-based integral image generation needs to flexibly access memory; hence, this work designs the two-level shared memory system that supports the multi-port to enhance the data bandwidth. The two-level shared memory system includes the shared memory_A, shared memory_B, shared memory_C and memory controller. The shared memory_A (1600×64 bit) supports eight 10×10 32-bit image blocks in each reconfigurable processor array to work in alternative mode for integral image computation. The shared memory_B (1 K $\times$ 64 bit) stores the detected sub-windows. The shared memory_C (5 K $\times$ 64 bit) works with the external memory to buffer the image blocks, integral image and detected windows. The shared memory is constructed by the sub-memory arrays to enhance data transmission. The memory controller

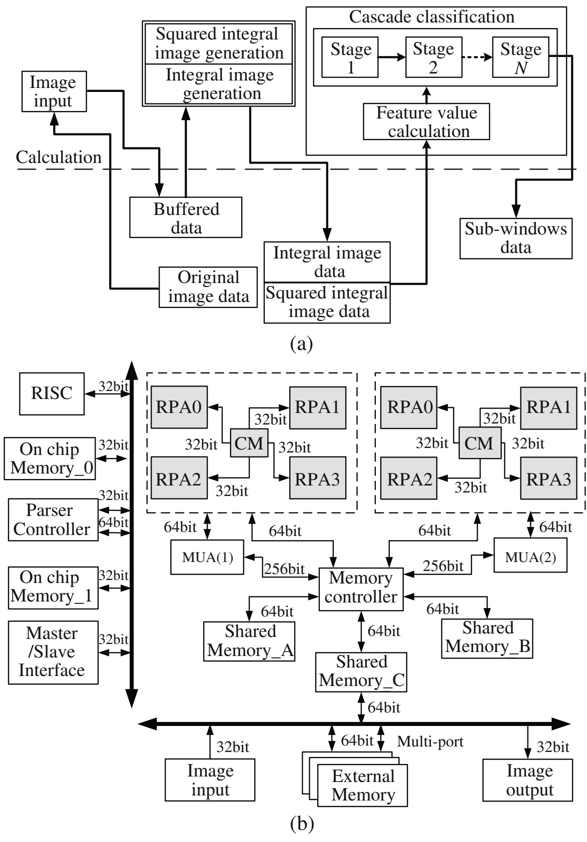
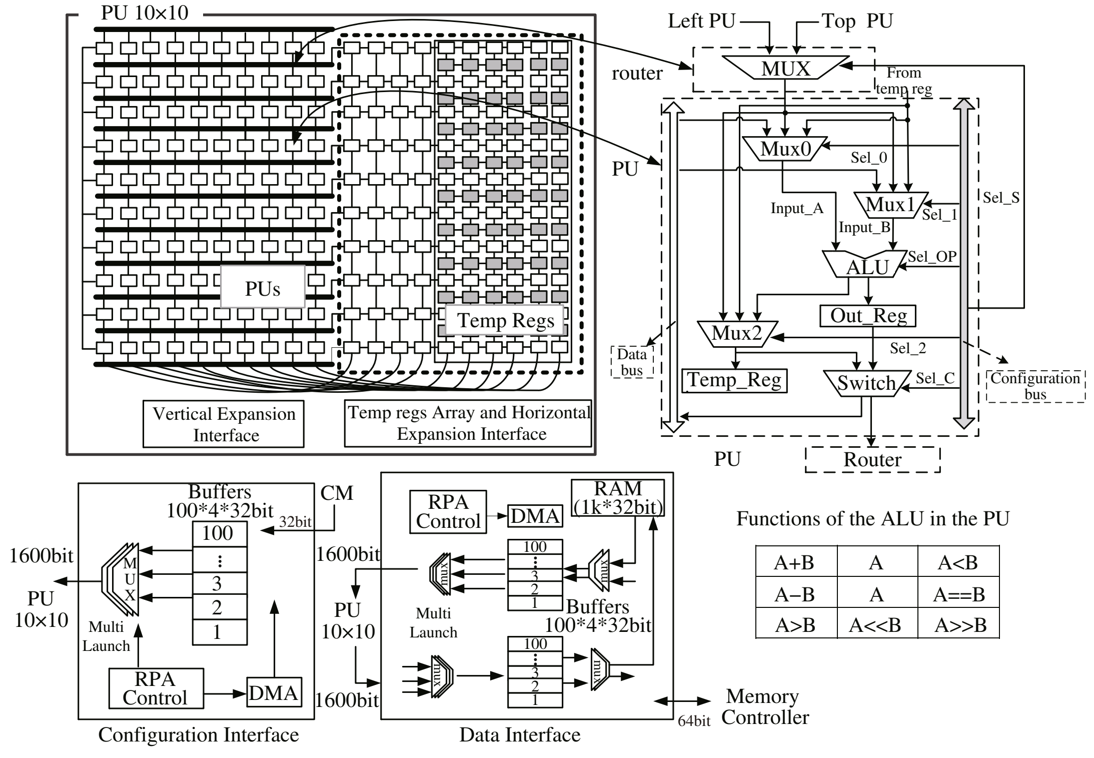


Figure 2 The architecture design. (a) The characteristics of algorithm processing; (b) the whole architecture.

establishes the data paths among the connected units, and arranges sub-memory arrays to construct hierarchical memory system. The parser controller generates the configurations and sends them to CM (Configuration Memory: 400×128 bit, supports four 32-bit configuration contexts to configure the PU in the pipeline way) to configure the operations of the PUs. Each configuration context is 32-bit length and includes the selected signals for the MUX, ALU, and data input/output address for the memory. In addition, since the squared operations in the squared integral image generation, and the multiplications and accumulations of the light compensation are characterized by high parallelism, two special arrays called Multiplication and Accumulation(MUA): 5×5 array are designed to achieve parallel processing. The MUA design saves the multipliers resources by 59% and 26% compared with the baseline design and the related work [15] separately. The baseline design is depicted in Section 5.

3.2 Reconfigurable processor array

As shown in Figure 3, the reconfigurable processor array includes the Configuration Interface (CI), Data Interface (DI), and 10×10 PUs. Each PU exchanges the data with the surrounding PUs through the input and output ports of the router. PU is the Arithmetic Logical Unit (ALU) based unit and can build several data paths. These data paths include the external data input and internal data output paths between the data bus and the PU, the temporary data path between the temporary registers and the PU, and the intermediate data transmission path among the PUs. While designing the PU, the experiments of the data width for ALU are implemented. In the experiments, for three kinds of data widths (32-bit,16-bit and 8-bit), the 32-bit design shows the best performance as it brings the highest data precision. The 16-bit design achieves comparable performance compared to the 32-bit design as the precisions of most data in the algorithm are met by the 16-bit design. The 8-bit design is the worst. Though the 32-bit design performs best, it costs an extra 20.5% hardware resource including the registers and control logics than the 16-bit design. Considering the hardware cost and the detection performance,



A+B	A	A<B
A−B	A	A==B
A>B	A<<B	A>>B

Figure 3 The architecture of the reconfigurable processor array (upper left: PU array (10 × 10); upper right: PU architecture; Left down: CI and DI; Right down: function table for the ALU).

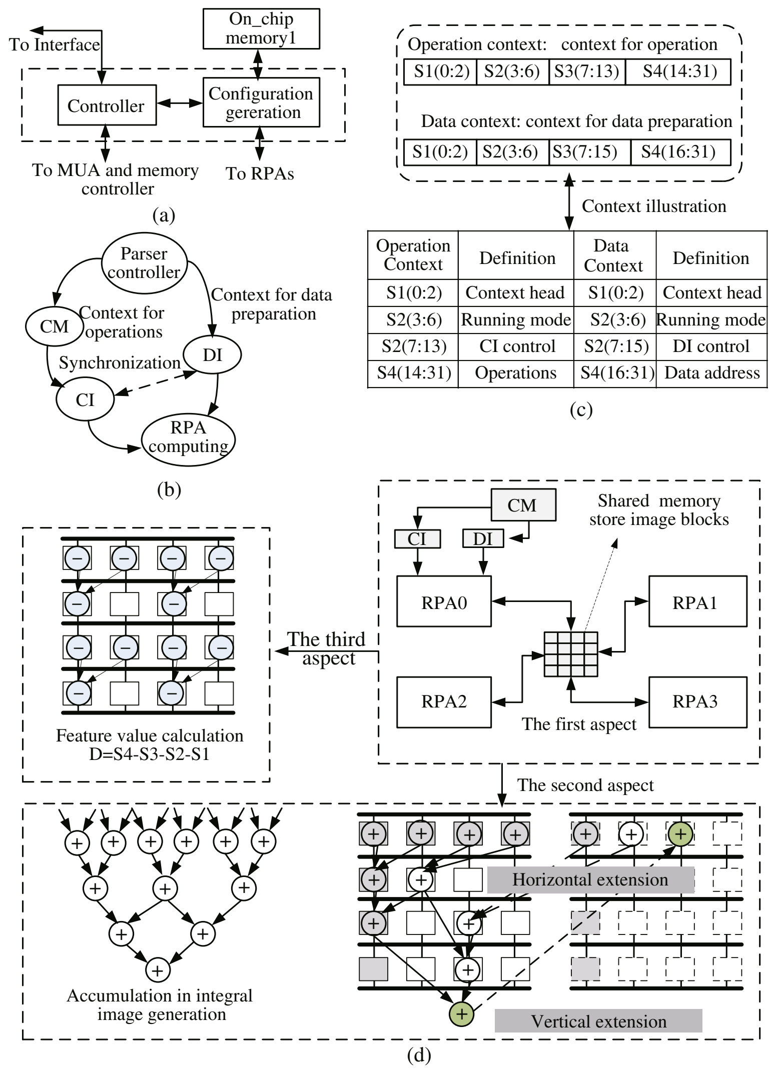
this work adopts the 16-bit as the data width for the ALU. The 32-bit data can be combined by two 16-bit data, and one 16-bit data can be divided into two 8-bit data.

The CI and DI are designed to reduce the processing time. A multi-launch unit is designed in the CI to reduce the configuration time; it dispatches the configurations to the whole array concurrently. In order to configure the array in the pipeline way to improve the processing time, one hundred buffers are designed in CI to store the configurations. Each buffer supports four sets of configurations. The RPA control module in CI receives configurations from the parser controller and implements the local control for the CI. In the DI, the input buffers and output buffers work in the alternate mode. Each group of input or output buffers consists of one hundred FIFOs, and the data are transmitted to the PUs through a multi-launch unit. In addition, an on-chip memory (RAM: 1 k × 32 bit) that stores constant data and temporary data is designed for the reconfigurable processor array; the DMA (Direct Memory Access) mechanism is used to improve the data transmission between the buffers and memories (internal memory and the shared memory). The RPA control module in the DI controls the DI according to configurations from the parser controller and adjusts the data bandwidth to adapt to the data transmission.

The reconfigurable processor array supports block computation as well as the PU array supports the array extension. As shown in Figure 3 (upper left part), the bottom switch network can be connected to the top switch network to implement the vertical extension, which is used for the mapping of the long data flow graph. The right columns of PUs can output the data to the column of PUs in the left to implement the horizontal extension; it is fit for the mapping of wide data flow graph. This array extensible design fully reuses the PUs. When multiple image blocks are processed in parallel, the extensible design enables the architecture to have enough capability to process the image in parallel.

3.3 Configuration memory

The parser controller configures the running mode of the reconfigurable processor arrays. As shown in Figure 4 (a) and (b), it receives the user setting, and delivers two types of contexts called operation context and data context to the CI and DI separately. As shown in Figure 4(c), the operation contexts


To Interface
On_chip memory1
Controller
Configuration gereration
To MUA and memory controller
To RPAs
(a)
Parser controller
CM
Context for operations
Context for data preparation
Synchronization
DI
CI
RPA computing
(b)
Operation context: context for operation
S1(0:2) S2(3:6) S3(7:13) S4(14:31)
Data context: context for data preparation
S1(0:2) S2(3:6) S3(7:15) S4(16:31)
Context illustration
(c)
CM
CI
DI
Shared memory store image blocks
RPA0
RPA1
RPA2
RPA3
The first aspect
The third aspect
The second aspect
Feature value calculation
D=S4-S3-S2-S1
Accumulation in integral image generation
Horizontal extension
Vertical extension
(d)

Operation Context	Definition	Data Context	Definition
S1(0:2)	Context head	S1(0:2)	Context head
S2(3:6)	Running mode	S2(3:6)	Running mode
S2(7:13)	CI control	S2(7:15)	DI control
S4(14:31)	Operations	S4(16:31)	Data address

Figure 4 (Color online) The parser mechanism to support the running mode configuration. (a) Parser controller; (b) configuration flow; (c) configuration definition; (d) running mode configuration.

are used to configure the operations of PUs, and the data context is used to transmit the data and control DI. As shown in Figure 4(d), the running mode configurations are summarized in three aspects. First, four reconfigurable processor arrays are combined as a group to implement full parallel running modes, single pipeline running modes or multi-pipeline running modes to support the image block-based computation. It provides great computation flexibility for mass data processing. The communications among the arrays are based on DI and the shared memory. Second, the reconfigurable processor array supports array extension to adapt to the processing of the wide or long data flow graph, which is used for the parallel computation of large image blocks. Third, the processing units in the reconfigurable processor arrays are grouped to support block computation, which is used for multiple feature value calculations.

3.4 The share memory scheme

This work adopts the two-level shared memory system to improve the memory accessing efficiency. As shown at right-top part of Figure 5, each shared memory consists of four sub-memories to construct the memory array; it improves the image block accessing efficiency. The shared memory contains multiple ports and constructs the hierarchical memory system to reduce the on-chip memory. The memory controller establishes the data transmission between the reconfigurable processor arrays and MUA, and generates the pixel address in block to load the block. This memory scheme achieves 63% improvement

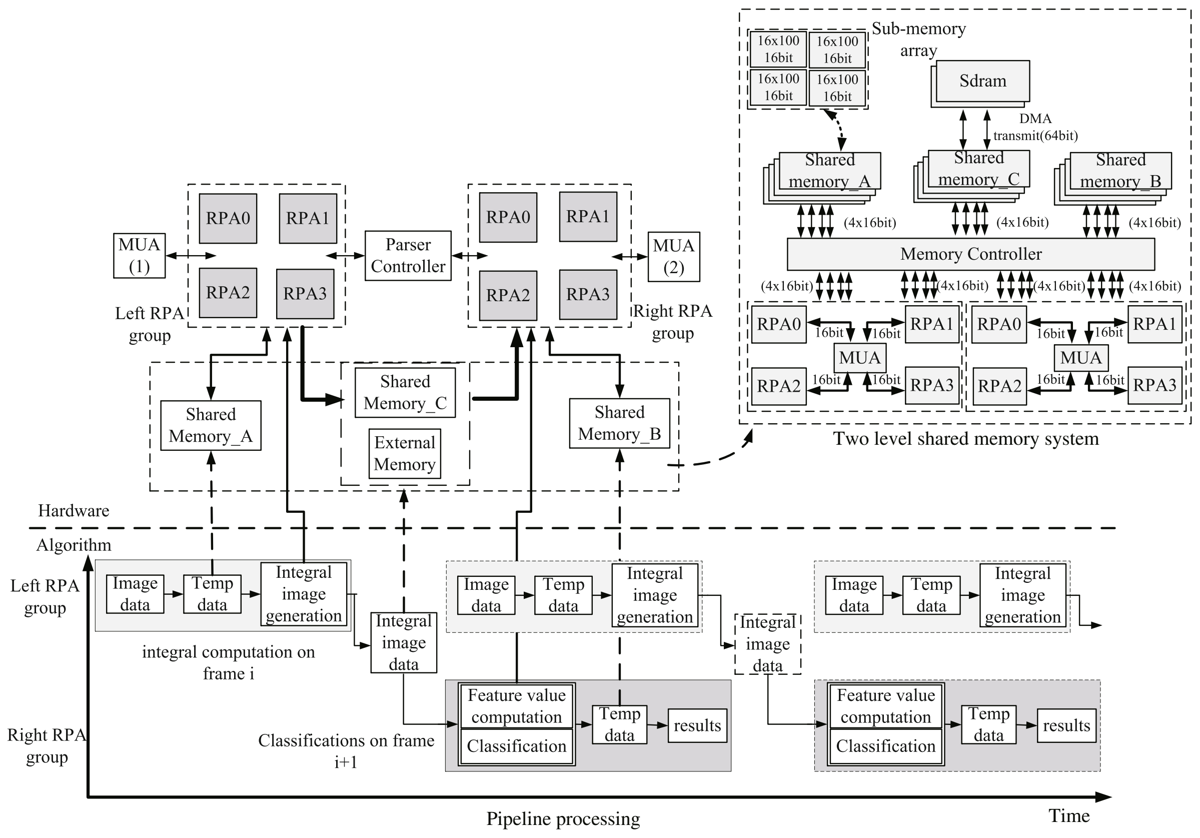


Figure 5 The computation flow of the algorithm in the architecture.

on memory accessing time over the baseline design. The baseline design is depicted in Section 5. Besides, considering the large span of data value in the integral image, a 16-bit is adopted as the memory data width. Multiple 16-bit data can be grouped to achieve high data precision. Besides, the memory controller is used as a digital controller to analyze the history detection results of the previous frames or receive the sub-window parameters configuration set by the users in the parser controller, and then determines the sub-window generation in the next frame. A detailed illustration is presented in Section 4.

4 Algorithm computation

Figure 5 shows the whole computation flow. Two groups of reconfigurable processor arrays work in the pipeline way. The left group processes block-based integral image generation. The right group processes the sub-window adaptive cascade classification. This section will explain the computations in detail. Besides, the shared memory_A and the shared memory_B are used to store the temporary data. The shared memory_C and external memory construct the hierarchical memory to buffer the data between these two reconfigurable processor array groups. It stores the integral image and original image.

4.1 Block-based integral image computation

These homogeneous configurable arrays support the block-based integral image computation. Figure 6 shows the data flow of integral image processing in the architecture. As shown in Figure 6, since the image data are flexibly inputted and transmitted through the routers in the array, this reconfigurable processor array supports the horizontal and vertical accumulation well. Consider a 4×4 image block as an example; the columns are horizontally delivered to achieve the column accumulation; then the rows are vertically delivered to achieve the row accumulation. After these steps, the integral operation of the image block is achieved. Besides, the integral image block is temporarily stored into the shared memory_A. The buffers

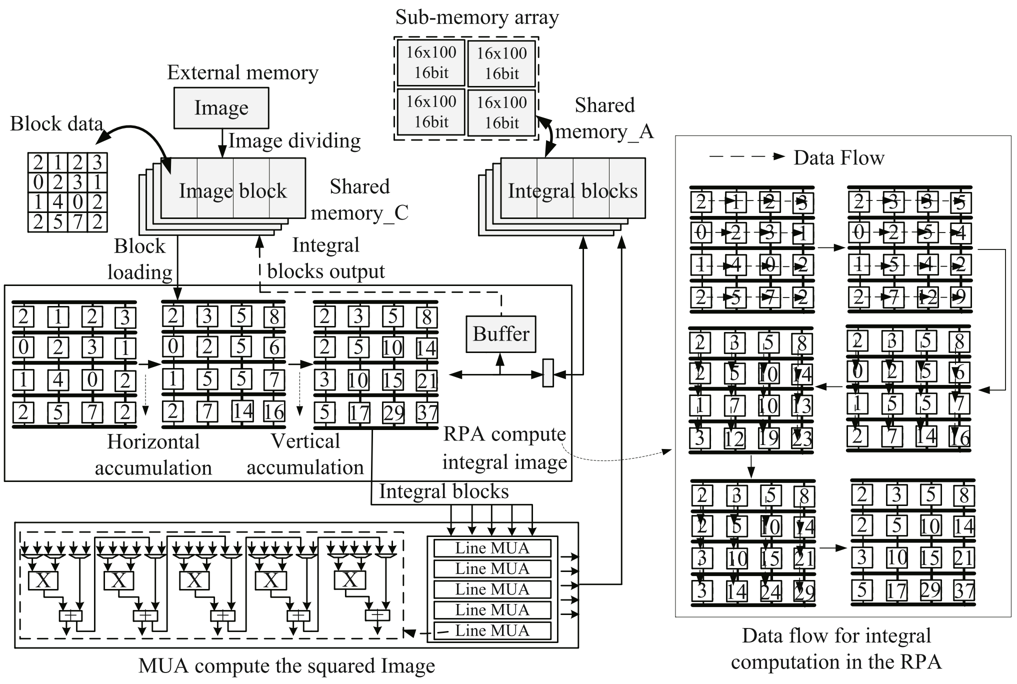


Figure 6 The integral image computation in the architecture.

in the reconfigurable processor arrays store the results of boundary row and column for the integral computation of the next block. The MUA (5×5) is used to achieve squared operation for the squared integral image. This work only needs six steps compared to the related work [15] that needs ten steps to achieve the 4×4 integral image block. Since the array size is 10×10 to provide a strong parallel processing capability, the main image block size is 10×10, and this architecture supports 8×8, 12×12, 16×16, and 20×20 well. The reconfigurable processor array fully processes these blocks. One 10×10 integral image block is divided into four 5×5 integral image block for the MUA computation.

4.2 Sub-window adaptive cascade classification

The number of sub-window affects the detection time and the detection rate. This architecture supports the sub-window adaptive cascade classification to adjust the data amount by changing the sub-window parameters. It strongly reduces redundant computation. As shown in Figure 7, the memory controller is a digital control module; it will check the number of the detected regions in the last four frames. If the number of detected regions tend to increase, which denotes that the contents in the frame tends to contain more face targets; the sub-windows are adjusted to implement fine detection. Otherwise, it denotes that the contents in the frame appear to be more redundant; the sub-windows are adjusted to implement coarse detection to reduce the redundant data computation. The initial sub-window is 20×20. The scaling factor can be adjusted from 1.1 to 2 (interval: 0.1) and the shifting step can be adjusted from 1 to 10 (interval: 1). Compared with the design that does not support the sub-windows adaptive mechanism, this sub-windows adaptive design improves the average computation speed by 2.48.

According to the coordinates of the sub-windows, the memory controller generates the memory address and loads the integral image data to the reconfigurable processor arrays. If the users reconfigure the sub-window parameters, the parser controller controls the memory controller to load the integral image data. The computation of feature value and threshold comparison in the reconfigurable processor array, weighting the threshold in MUA are processed in a pipeline way. The reconfigurable processor array supports eight pipelines working in parallel. If the classifications are true, the sub-windows coordinates are stored into shared memory_B for further classification in the next stage. Thirty stages of classification parameters have been trained and stored in the on chip memory_1.

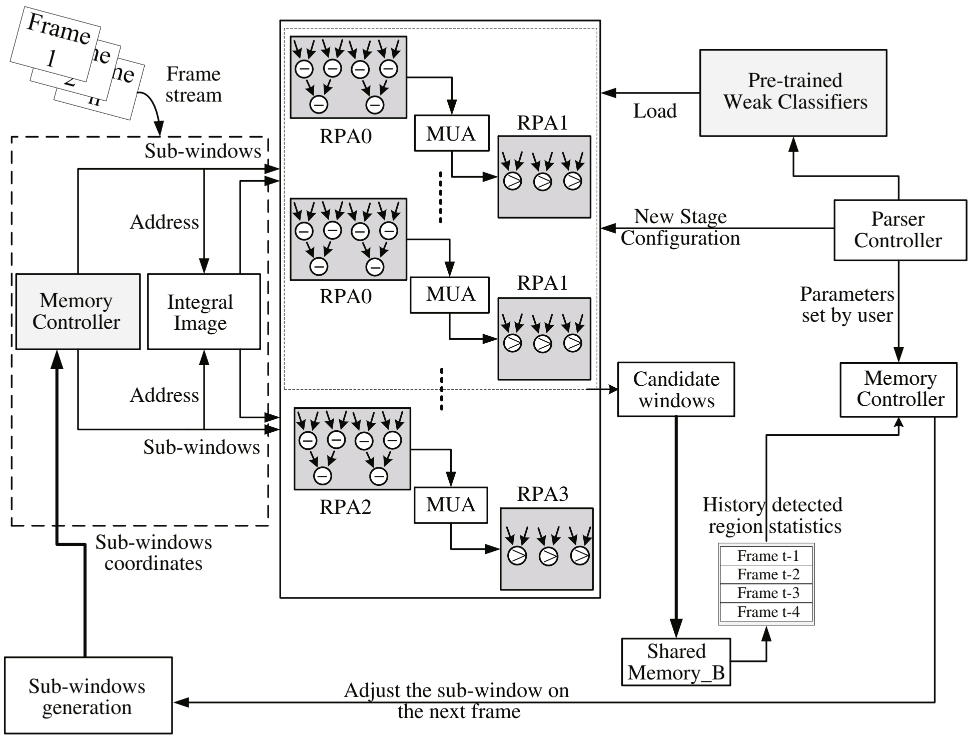


Figure 7 Sub-window adaptive cascade classification.

5 Evaluation

In this work, the face-priority auto-focus with fast processing speed and low power consumption for digital and smart phone cameras is the design goal. Hence, in this section, in order to show that the proposed hardware architecture can achieve fast processing speed and low power consumption for the auto-focus, the comparisons with the other face detection architectures and baseline designs that use the traditional application specific integrated circuit (ASIC) way are implemented. Meanwhile, the comparisons with related work on face detection based auto-focus are implemented to show the high performance of auto-focus in this work. Further, the evaluations on different image resolutions are implemented to show the high processing capacity on high-resolution images. In the evaluations, the experimental platform is a FPGA development board called DE2-115. This board provides a wide variety of interfaces, including VGA, ethernet, UART, and D5M camera interface. Besides, it has an Altera series FPGA chip called Cyclone IV (EP4CE115). We adopt this board to implement the architecture for comparison. In this FPGA chip, it contains 114480 logic elements (LEs), 3981312 on-chip memory bits and 532 embedded multiplier of 9-bit elements in total. These hardware resources in the chip can support our design well. Besides, TSMC 65 nm 1P9M GP+ technology is adopted to implement post-synthesis to achieve the area information and power information.

5.1 The cascade classifier training and baseline design

In this work, 3200 features are adopted to train the cascade classifier with 30 stages. The initial shifting and scaling parameters for the sub-window are 1 and 1.1, respectively. The number of features in each stage ranges from 20 to 550. The training images are from the Internet, MIT+CMU data set, Yale Face data set, and ORL data set. The number of the positive samples is 2500, and the number of the negative samples is 4300. The testing data are from the MIT+CMU image set and the Internet. Twenty testing groups are used for the experiments. Each group contains ten videos and ten image sequences. These testing groups contain the cases affected by the illumination and pose of the faces. The number of faces in each image or each frame ranges from one to fifteen. The results are averagely computed from the twenty testing groups. Besides, the baseline design of the algorithm is designed for comparison. In the baseline design, the algorithm is divided into integral image generation, squared integral image generation, sub-window generation for feature calculations and cascade classification. The whole architecture for the baseline design is heterogeneous; each part is designed as the specific hardware and has the private memory; hence, the memory scheme for the baseline design is not shared.

5.2 Comparison with the baseline design

In order to prove that the proposed architecture is the optimal design for the auto-focus, the comparisons with the baseline design are implemented. We set the condition of the same hardware resources including logic elements (25145LEs), multiplier (50) and memory (350 embedded memory block) for both schemes. Then we implement the testings to achieve fair comparison on the processing speed. The two groups of reconfigurable processor array (RPA)(consumes 149305 cycles) improve the performance by 45% than the baseline design (consumes 82513 cycles); the reason is that the reconfigurable processor array groups work in the pipeline way and the array based architecture exhibits strong parallel processing power. Design of the array of multiplication and accumulation (MUA) (consumes 13936 cycles) enhances the parallelism in computation; it improves the execution efficiency by 48% than baseline design (consumes 7219 cycles). Besides, this work adopts a two-level shared memory scheme that uses sub-memory arrays to enhance data transmission and constructs the hierarchal memory system to reduce on-chip memory (consumes 113160 cycles); it improves the memory accessing efficiency by 63% (baseline design consumes 41805 cycles).

Since the baseline design is implemented in ASIC way, it only needs the data memory (DM). Our architecture belongs to reconfigurable design, which needs the configuration memory (CM) and data memory (DM). Hence, the hardware resource of the configuration memory for the baseline design is zero as we do not need it in ASIC way. Besides, the CM and DM are used for different purposes. The CM is used to store the configurations which will configure the functions of processing units in the reconfigurable architecture. The DM is used to store the data including the image data, intermediate data and detection results. Hence, the size of CM and DM is different. In order to achieve the comparison between our work and baseline design, the detection performance is fixed. It means that the detection performance including the detection rate, false positive rate, and processing speed are set to be the same. Under the constraint of detection performance (Detection rate: 96.1%, False positive rate: 0.002, Processing speed: 122fps@640×480), we implement the baseline design, which will enable the hardware resource to achieve the detection performance goal. Compared to baseline design, the arrays of multiplication and accumulation process the squared integral image and classification in parallel (consumes 25145 LEs), it reduces the amounts of multipliers by 59%. Meanwhile, this work adopts the two-level shared memory scheme, though an extra configuration memory is designed for the array configurable functions, the total memory blocks (consumes 350 memory blocks in this work) are reduced by 52%. Further, since the reconfigurable processor arrays are homogenous and configurable, the configurations for the reconfigurable processor arrays are greatly compressed; the homogenous and configurable arrays design (consumes 50 multipliers) reduces the logic elements by 42%. Hence, this work achieves auto-focus at a lower cost, which brings certain advantages for deployment in digital and smart phone cameras.

5.3 Comparisons with face detection architectures

In this work, performance comparison is affected by the testing cases. We perform comparisons with other work in the same testing cases to achieve fair comparison on the detection metrics like detection rate, false positive rate, power consumption and process speed. Since different reference work adopts different testing cases, we test our work using the cases, which have been indicated in their manuscripts to achieve fair comparison. After the comparisons, we present the average performance in Table 2. Besides, for the comparisons in Table 2, we have new performance metrics like processing efficiency (pixels/s@1 MHz), power efficiency (pixels/s@1 mW, speed/power), area efficiency (speed/transistors), to evaluate the processing efficiency and power efficiency more fairly. Note that, since the overall power consumption depends on not only the architecture, but also the frequency, the image size, image contents,

Table 2 Comparisons with other works

	Hanai [14]	Hiromoto [16]	Kyrkou [15]	This work
Technology	90 nm	/	65 nm	65 nm
Image size	320×240	640×480	640×480	640×480
Frequency (MHz)	54	160.9	800	200
Detection speed (fps)	8	30	118	122
Power(unified to 65 nm on 320×240 image)	1.51 mW	/	2.4 mW	1.8 mW
Detection rate	81%	92.8%	95%	96.1%
False positive rate	< 0.05	0.00005	/	0.002
Transistors count (million)	2.1	/	88	19.2
Pixels/s@1 MHz	11.38 k	57.28 k	12.77 k	187.4 k
Pixels/s@1 mW	407.9 k	/	4256.0 k	10453.7 k
Speed/Transistor (fps/million)	3.81	/	1.51	6.35
Speed/power (fps/mW)	5.30	/	55.4	136.1

Table 3 Comparison with the auto-focus work

	Mohammad [3]	This work
Image size	640×480	640×480
Frequency	200 MHz	200 MHz
Detection rate	91%	96.3%
False positive rate	0.002	0.002
Processing time	average 35 ms	average 8.2 ms
Power	97 mW	3.52 mW

and the technology, independent power comparisons of these studies are not fair. Hence, for the reference work, we scale the technology (unified as the 65nm technology) to the same and test the performance on the 320×240 image to achieve fair comparison on power consumption. In Table 2, performance metrics including power consumption, speed/power, and pixels/s@ 1mW are based on the unified image size: 320×240. The other performance metrics, including the pixels/s@1 MHz and speed/transistor (fps/million) are based on the image size of 640×480.

As shown in Table 2, Hanai [14] implemented face detection on 320×240 images; Hiromoto [16] achieved the 30fps detection speed on the 640×480 images. Compared to the work of Hanai [14] and Hiromoto [16], this work obviously achieves higher detection speed, and improves the average processing efficiency by 15.5 times and 2.27 times respectively. Kyrkou [15] achieves the high detection speed on the 640×480 images using 800 MHz frequency; yet this work achieves higher detection speed on 640×480 images only using the 200 MHz frequency. We improve the processing efficiency by 13.68 times. Further, this work achieves the estimated area of 19.2 million transistors from the post-synthesis in 65nm technology; it reduces the number of transistors (unit:Million) compared to the work in Kyrkou [15] that costs 88 million transistors in 65nm technology. Hanai [14] achieves the 2.1 million transistors in 90nm technology, but they focused on the 320×240 image with small on-chip rams, we improve the area efficiency by 66.7%. In addition, as shown in Table 2, we improve the power efficiency by 24.63 times and 1.36 times respectively compared to that of Hanai [14] and Kyrkou [15].

5.4 Comparison with the auto-focus work of mohammad

Mohammad [3] proposes a face detection algorithm and combines the face detection algorithm and rule-based searching method [1] for face-priority auto-focus. They evaluate the face detection algorithm in an ARM processor (200 MHz). Table 3 shows the comparisons on the same testing images. In Table 3, the reference work is an implementation in the embedded processor rather than an architecture design; hence, we only compare the power and execution time under the same testing images. And the technology scaling and performance metrics like pixels/s@1 mW, speed/power are unnecessary to be implemented. According to Table 3, this work achieves higher processing speed and detection rate than the work of Mohammad [3]; meantime, power consumption is largely reduced. Besides, by improving a variety of training samples, the face-priority auto-focus can tackle complex scenes including the illumination changes, multiple faces, rotation, and partial occlusion.

6 Conclusion

This work proposes a fast face detection architecture with low power consumption for face-priority auto-focus in smart phones and digital cameras. It supports the methods of block-based integral image computation and sub-window adaptive cascade classification. Experimental results show that this work greatly improves the processing speed of auto-focus on high-resolution images and achieves low power consumption; meanwhile, it achieves high robustness for various face detection scenes.

Acknowledgements This work was supported in part by China Major Science and Technology (S&T) Project (Grant No. 2013ZX01033-001-001-003), National High-Tech R&D Program of China (863) (Grant Nos. 2012AA01-2701, 2012AA0109-04), National Natural Science Foundation of China (Grant No. 61274131), International S&T Cooperation Project of China (Grant No. 2012DFA11170), and Importation and Development of the High-Caliber Talents Project of Beijing Municipal Institutions (Grant No. YETP0163).

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Kehtarnavaz N, Oh H. Development and real-time implementation of a rule-based auto-focus algorithm. Real-Time Imag, 2003, 9: 197–203
2 Peddigari V, Gamadia M, Kehtarnavaz N. Real-time implementation issues in passive automatic focusing for digital still camera. J Imag Sci Technol, 2005, 49: 114–123
3 Rahman M, Kehtarnavaz N. Real-time face-priority auto focus for digital and cell-phone cameras. IEEE Trans Consum Electron, 2008, 54: 1506–1513
4 Xiong Y G, Pulli K. Color matching for high-quality panoramic images on mobile phones. IEEE Trans Consum Electron, 2010, 56: 2592–2600
5 Chandrasekaran V, Dantu R, Jonnada S, et al. Cuffless differential blood Pressure estimation using smart phones. IEEE Trans Biomed Eng, 2013, 60: 1080–1089
6 Yang M, Kriegman D, Ahuja N. Detecting faces in images: a survey. IEEE Trans Patt Anal Mach Intell, 2002, 24: 34–58
7 Huang D Y, Lin C J, Hu W C. Learning-based face detection by adaptive switching of skin color models and AdaBoost under varying illumination. J Inf Hid Multimed Signal Process, 2011, 2: 204–216
8 Zhang Z W, Wang M H, Lu Z M, et al. A skin color model based on modified GLHS space for face detection. J Inf Hid Multimed Signal Process, 2014, 5: 144–151
9 Viola P, Jones M. Robust real-time face detection. Int J Comput Vis, 2004, 57: 137–154
10 Isobe T, Fujiwara M, Kaneta H. Development and features of a TV navigation system. IEEE Trans Consum Electron, 2003, 50: 393–399
11 Zuo F, de With P H N. Real-time embedded face recognition for smart home. IEEE Trans Consum Electron, 2005, 51: 183–190
12 An K H, Chuang M J. Cognitive face analysis system for future interactive TV. IEEE Trans Consum Electron, 2009, 55: 2271–2279
13 Soowoong K, Jae-young S, Seungjoon Y. Vision-based cleaning area control for cleaning robots. IEEE Trans Consum Electron, 2002, 58: 685–690
14 Hanai Y, Hori Y, Nishimura J, et al. A versatile recognition processor employing Haar-like feature and cascade classifier. In: Proceedings of IEEE International Conference on Solid-State Circuits, San Francisco, 2009. 148–149
15 Kyrkou C, Theocharides T. A flexible parallel hardware architecture for AdaBoost-based real-time object detection. IEEE Trans Very Large Scale Integr Syst, 2011, 19: 1034–1047
16 Hiromoto M, Sugano H, Miyamoto R. Partially parallel architecture for AdaBoost-based detection with Haar-like features. IEEE Trans Circ Syst Vid Technol, 2009, 19: 41–52
17 Liu L B, Chen Y J, Wang D, et al. Implementation of multi-standard video decoder on a heterogeneous coarse-grained reconfigurable processor. Sci China Inf Sci, 2014, 57: 082406
18 Liu L B, Chen Y J, Yin S Y, et al. Implementation of AVS Jizhun decoder with HW/SW partitioning on a coarse-grained reconfigurable multimedia system. Sci China Inf Sci, 2014, 57: 082401