J Sign Process Syst (2011) 62:77–96
DOI 10.1007/s11265-008-0311-6

Reconfigurable Morphological Image Processing Accelerator for Video Object Segmentation

Shao-Yi Chien · Liang-Gee Chen

Received: 27 April 2008 / Revised: 14 September 2008 / Accepted: 25 October 2008 / Published online: 18 November 2008

Abstract Video object segmentation is an important pre-processing task for many video analysis systems. To achieve the requirement of real-time video analysis, hardware acceleration is required. In this paper, after analyzing existing video object segmentation algorithms, it is found that most of the core operations can be implemented with simple morphology operations. Therefore, with the concepts of morphological image processing element array and stream processing, a reconfigurable morphological image processing accelerator is proposed, where by the proposed instruction set, the operation of each processing element can be controlled, and the interconnection between processing elements can also be reconfigured. Simulation results show that most of the core operations of video object segmentation can be supported by the accelerator by only changing the instructions. A prototype chip is designed to support real-time change-detection-and-background-registration based video object segmentation algorithm. This chip incorporates eight macro processing elements and can support a processing capacity of 6,200 9-bit morphological operations per second on a SIF image. Furthermore, with the proposed tiling and pipelined-parallel techniques, a real-time watershed transform can be achieved using 32 macro processing elements.

Keywords Video object segmentation · Hardware accelerator · Morphological image processing element array · Reconfigurable · Stream processor

S.-Y. Chien (✉) · L.-G. Chen
National Taiwan University, Taipei, Taiwan
e-mail: sychien@cc.ee.ntu.edu.tw

1 Introduction

Video object segmentation is the technique which can generate object shape information from video sequences. It is the key operation for content-based video coding systems, such as MPEG-4, to realize content-based coding functionalities. It is also the key pre-processing for many video analysis systems. For example, for an intelligent video surveillance system, the object location and shape information are important for object behavior analysis, object recognition, and video indexing.

Several video segmentation algorithms have been proposed [4, 5, 7, 12, 13, 19, 20, 27, 29, 30, 32, 35]. Most of these algorithms can be covered by the video segmentation framework proposed in MPEG-4 standard [21], which is shown in Fig. 1. *Camera Motion Compensation* and *Scene Cut Detection* first compensate the effect of camera motion and detect when scene change occurs. *Temporal Segmentation* segments video sequences with temporal information, such as motion information and change detection. *Spatial Segmentation* detects spatial information of video sequences such as edge information and region information from image segmentation, and it combines this information with the segmentation results of *Temporal Segmentation*. The output of a video segmentation system can come from the combined results of both segmentation subsystems or only the results of *Temporal Segmentation*. In [36], it is shown that the most popular temporal segmentation algorithm is change detection where the most popular spatial segmentation algorithm is watershed transform.

Among these algorithms, however, no single algorithm is suitable for all kinds of situations. For example, our algorithm [7] can deal with still camera and

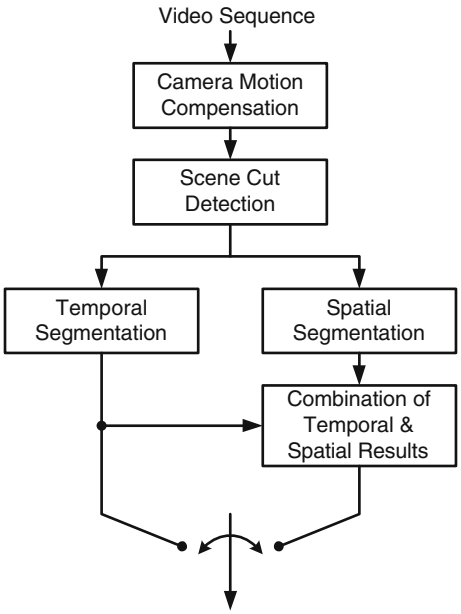


Figure 1 Video segmentation framework in Annex-F of MPEG-4 standard.

multiple objects situations with real-time performance on QCIF format video, and the modified one [5] can deal with slightly moving cameras; Kim's algorithm [20] and Mech's algorithm [19] can generate accurate segmentation results, but the computational complexity is very high; Meier's algorithm [20] can deal with moving camera situations, but the complexity is also high and the segmentation results are sometimes not good enough; Wang's algorithm [32] is very useful for semi-automatic video segmentation for off-line video editing, but it is hard to be applied in real-time applications.

On the other hand, video object segmentation can be used for many real-time applications, for example, real-time content-based coding applications such as video camcorders, video phones, and video conference systems. It is also very useful as a pre-processing for real-time video analysis applications such as video surveillance and intelligent car driving systems. However, for SIF format or other larger frame size, even with a fast algorithm [7] and a powerful microprocessor, it is still very hard to achieve the real-time requirement of 30 frames/s. Therefore, hardware implementation of video segmentation is necessary. Besides, the system should be flexible since there is still no general solution for video segmentation, and different algorithms should be adopted for different situations. Consequently, for a wider range of applications, a hardware accelerator which can accelerate different segmentation algorithms with an unified architecture is urgently needed.

In this paper, a reconfigurable hardware accelerator for both spatial and temporal segmentation of video object segmentation is proposed. The core of this accelerator is a reconfigurable morphological image processing element (PE) array. For different algorithms, the PE array can be re-programmed to perform different operations, and the interconnection between the PEs can also be reconfigured to meet the requirements. The target algorithms to be accelerated are the most important algorithms but also all other algorithms developed with similar core operations.

This paper is organized as follows. Existing algorithms are first analyzed in Section 2. After that, in Section 3, core operations of video segmentation are mapped to morphological operations [26, 28]. Section 4 and Section 5 show the proposed architecture and the optimization issues for this architecture, respectively. Then Section 6 shows the implementation results. Finally, Section 7 gives a conclusion of this paper.

2 Analysis of Existing Algorithms

Ten algorithms are analyzed in this section. These algorithms represent most of the existing video segmentation algorithms. Change detection is widely used in many algorithms [7, 13, 19]. Our prior video segmentation algorithm is designed for still-camera situations [7]. It is based on change detection and background registration techniques. A gradient filter is used to eliminate light changing and shadow effects, and a post-processing algorithm, which includes region size filtering and morphological close-open operations, is used to improve the segmentation results. Kim's algorithm [13] combines change detection and watershed [31] as a spatio-temporal segmentation algorithm. This algorithm can give better segmentation results, but with larger computation power. Mech's algorithm [19] is also based on change detection. A boundary relaxation algorithm is applied to improve the boundary of the change detection mask (CDM). The uncover background problem of change detection, that is, the uncover background regions are usually misclassified to foreground objects, is eliminated with motion information derived by optical flow, and edge detection and edge fitting are used to fit CDM to edges of a frame. In some algorithms, edge information is used as the input data instead of pixel values of input frames. Meier's algorithm [20] segments video objects on edge frames generated with a Canny edge detector [3]. A morphological motion filter [9, 25] is used to find the outlier parts of a frame with motion information, and a Hausdorff distance is applied to track the edges of a video objects. This algorithm can be used for moving camera situations; however, the accuracy may be not

as good as those of other algorithms. Kim's algorithm [12] employs the similar concepts to reduce the complexity of the algorithm by detecting changing parts of the edges. In addition, many algorithms employs watershed transform [31] to segment input frames into several non-overlapped regions, and then these regions are tracked and merged to form video objects. Wang's algorithm [32] applied a watershed transform on the gradient image derived with a multiscale gradient operation. After segmenting a frame into many regions, a motion tracking and projection algorithm is used to track each region between consecutive frames. Tsaig's algorithm [30] employs similar concepts with watershed and hierarchical region matching. An extension of the conventional watershed transform, 3-D watershed, is proposed by Tsai et al. [29], and a Bayesian approach is proposed to merge watershed volumes to form video objects. These algorithms can also be used in moving camera situations and give accurate segmentation results; nevertheless, it is quite complex and requires enormous computation power. Moreover, some algorithms employ motion segmentation, where motion vectors are clustered to derive regions with homogeneous motion features. Shamim's algorithm is an example of this kind of algorithm [27]; however, the computational complexity is very high. Finally, many algorithms use a hybrid approach with the above concepts, such as Xu's algorithm [35]. The core operations of these algorithm are shown in Table 1.

Many of the core operations in these algorithms are the same, and they can be classified into five types: morphological operations, region growing operations, pixel operations, motion estimation related operations, and others operations, as shown in Table 2. Mathematical morphological operations are based on two basic operations: dilation and erosion, and both binary and gray-scale morphological operations are included. Note that the watershed transform usually requires close-open operations to simplify a frame and a gradient operation to generate gradient images, so the watershed transform can be categorized as a morphological operation. Many core operations are region growing operations. The watershed transform is also one of them. Note that the MAX-Tree (MIN-Tree) generation and distance transform of morphological motion filter and Hausdorff distance are also region growing operations. Pixel operations are operations independent between pixels. The computational load of this kind of operations is low and can usually be accelerated with sub-word parallel instructions, such as Intel MMX instructions [23]. Motion estimation related operations generate motion field information with motion estimation algorithms. The computational load of these kind of operation is quite large.

Table 1 Core operations of each video segmentation algorithm.

Algorithm	Core operations
Ours [7]	Gradient, change detection, background registration, post-processing
Kim [13]	Change detection, watershed
Mech [19]	Change detection, relaxation, optical flow, edge detection, edge fitting
Meier [20]	Morphological motion filter, optical flow, Canny edge detection, Hausdorff distance
Kim [12]	Canny edge detection, morphological operation
Wang [32]	Multiscale gradient, watershed, motion tacking and projection
Tsaig [30]	Watershed, hierarchical region matching
Tsai [29]	3-D watershed, Bayesian volume merging
Shamim [27]	Global-to-local motion segmentation, morphology operations
Xu [35]	Motion segmentation, Canny edge detection, Hausdorff distance, watershed

Table 2 Classification of the core operations of video segmentation.

Operation type	Associate core operations
Morphological operation	Gradient, post-processing, watershed, multiscale gradient
Region growing operation	Watershed, edge fitting, morphological motion filter, Hausdorff distance
Pixel operation	Change detection, background registration
Motion estimation related	Optical flow, motion tracking and projection
Other	Relaxation, Canny edge detection

Since the computational load of pixel operations is usually low, and the hardware of motion estimation is an essential part in a video encoding system, these two kinds of operations are not taken into consideration in this paper. Moreover, we found that region growing operation can also be mapped to morphological operations [26, 28]. Canny edge detection, in addition, can be replaced with an edge detector based on the morphological gradient operation, and the morphology based post-processing of our algorithm can achieve spatial homogeneity as relaxation. Therefore, most of the core operations of video segmentation can be mapped to morphological operations. In order to accelerate video object segmentation, the accelerator or processor which can execute different types of morphological operations is a good choice. Detailed methods to map these core operations to morphological operations will be discussed in the next section.

Similar concepts have been proposed in other literatures. The PIMM1 (Processor Integre de Morphologie Mathematique) of Center de Morphologie

Mathematique [15] is one of them. It is a programmable morphological image processor which can execute eight binary morphological operations in parallel or pipeline, one gray-level morphological operation, or one recursive morphological operation. Also, twelve PIMM1 chips are employed in a real-time road segmentation system in the European PROMETHEUS project [24] with other processors. This chip is designed for image analysis applications; however, for video segmentation applications, many parts of this chip are redundant, and it is not cost-effective. Moreover, one chip can only execute one gray-level morphological operation. The computing power is too small without a multi-chip configuration for a video segmentation system. There is another processor for morphological operations with an FPGA structure [33, 34]. With a pipelined architecture, which is combined with several FPGAs, FIFOs, and triple-port memories, this system can be reconfigured to any operations including morphological operations, median filter, and convolution. However, this system has several drawbacks: the programming time of the FPGA takes tens to hundreds milliseconds, which means it is not suitable to be programmed on-the-fly in a real-time system; furthermore, the system is designed for real-time pre-processing, that is, only low-level and simple operations can be supported, and complicated morphological operations in video segmentation cannot be executed in this system.

Consequently, in this paper, a reconfigurable morphological image processing accelerator is developed for image/video segmentation. Stream processing concept is employed to increase the efficiency of the data flow. In addition, the programming time is very short because the reconfigurable circuits are not for general-purpose applications. Therefore, the hardware can be programmed on-the-fly, which can further reduce the hardware cost, since the resource can be shared not only spatially but also temporally.

3 Mapping Core Operations to Morphological Operations

In this section, the core operations of video segmentation are mapped to morphological operations.

3.1 Gradient

First of all, the gradient operation can be shown as the following equation:

$$GRA = I \oplus B - I \ominus B, \tag{1}$$

where $\oplus$ is dilation, $\ominus$ is erosion, I is input image, and B is structuring element. It is a combination of morphological operations [26].

On the other hand, the multiscale gradient operation [32], which can be used to enhance ramp edge information, can be described as follows:

$$MG = \frac{1}{3}\sum_{i=1}^{3}\left[(I \oplus B_{2i+1} - I \ominus B_{2i+1}) \ominus B_{2i-1}\right], \tag{2}$$

where B_n denotes a nxn structuring element. It is originally a morphological operation.

3.2 Post-Processing of Change Detection Based Algorithm

The post-processing of our algorithm [7] includes region size filtering and close-open operations. The region size filtering can be replaced with dilation and conditional erosion (geodesic erosion) operations [26]:

$$(((I \oplus B_n) \underbrace{\ominus B_3; I) \ldots \ominus B_3; I)}_{l}, \tag{3}$$

where $l > (n-1)/2$, n is proportion to the smallest allowed region size, and the conditional erosion is

$$(X \ominus B; Y) = (X \ominus B) \cup Y. \tag{4}$$

Moreover, the closing and opening operations are originally morphological operations. which can be shown as the following equations:

$$I \circ B = (I \ominus B) \oplus B, \tag{5}$$

$$I \bullet B = (I \oplus B) \ominus B, \tag{6}$$

where $\circ$ is opening, and $\bullet$ is closing. These operations can also be employed for other change detection based algorithms.

3.3 Watershed Transform

The watershed transform [31] can separate an image into many homogeneous non-overlapping closed regions. It has four main steps: simplification, gradient, sorting, and flooding. The simplification and gradient are morphological operations. The sorting can be efficiently implemented with address sort [31], whose computational complexity is low enough to be afforded by general-purpose processors; however, the computational load of flooding process is high. Fortunately, the flooding process, which is a region growing operation, can be mapped to masked morphological erosion operations, which is easy to describe with an example. Figure 2a shows the gray level values of a gradient image. Each pixel is then given an unique label from

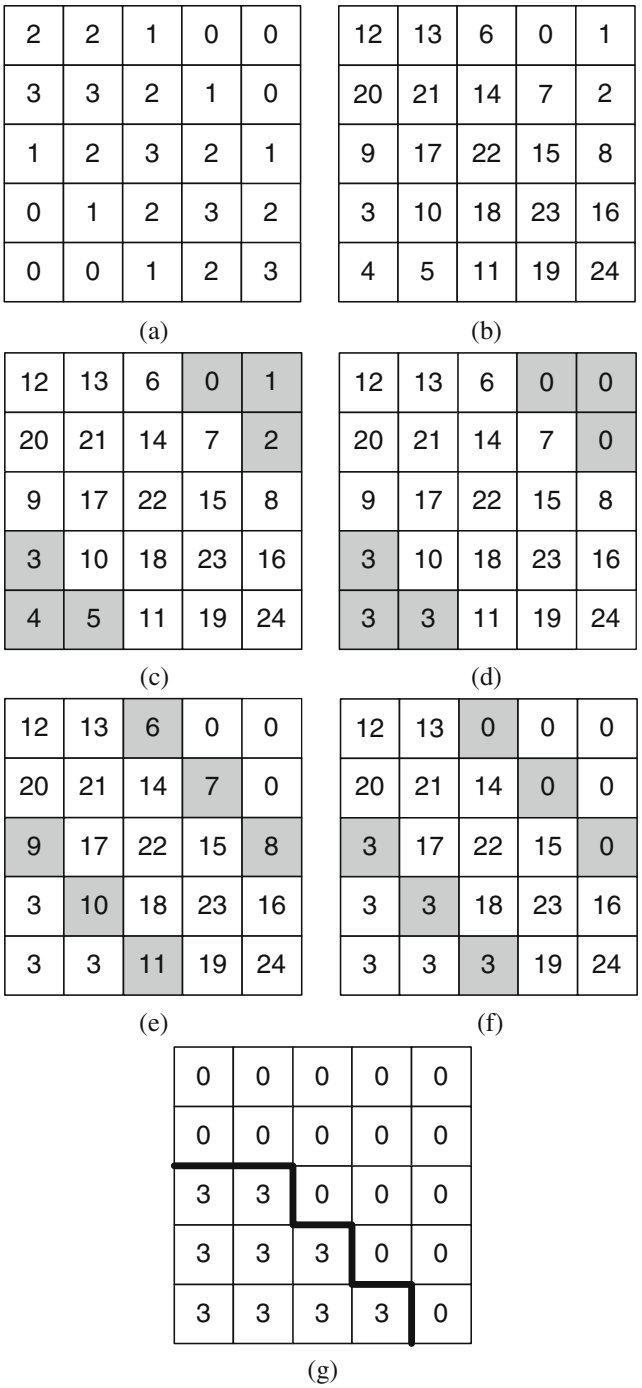


Figure 2 Map flooding process of watershed transform to masked morphological erosion operation (**a**–**g**).

low gray-level to high gray-level in raster scan order, as shown in Fig. 2b. For example, there are six pixels valued 0 in Fig. 2a, and each pixel is given a label from 0 to 5, from top to bottom and from left to right in Fig. 2b. Similarly, there are also six pixels valued 1 in Fig. 2a, and each pixel is given a label from 6 to 11, from top to bottom and left to right in Fig. 2b. Note that, based on the address sort concept, the computational load of this procedure is low. After that, masked erosion operations are applied, that is, erosion operations are applied only on pixels within a mask. For example, the mask of gray-level 0 is shown in Fig. 2c with gray color, where the pixel values are 0 in Fig. 2a. After Fig. 2c is processed by masked erosion operations only in the gray regions until no change occurs, the result is shown as Fig. 2d. For gray-level 1, the mask is shown in Fig. 2e, and the masked erosion result is shown in Fig. 2f. If the same operations are applied gray-level by gray-level, the result is shown in Fig. 2g, where the bold line shows the watershed. Therefore, the watershed transform can be mapped to masked erosion, and the result is the same as that of conventional watershed transform.

3.4 Hausdorff Distance

The distance transform is very important in the image analysis and computer vision algorithms, and is the essential operation to calculate Hausdorff distance. It can also be carried out with morphological operations:

$$Distance = X - \sum_{i=0}^{maxdistance-1} (BEdge \oplus D_i), \tag{7}$$

where *Distance* is the result of the distance transform, *maxdistance* is the maximum distance allowed in the distance transform, X is an image with all pixel values equal to *maxdistance*, $BEdge$ is a binary image, where edge pixels are valued 1, and other pixels are valued 0, and D_i is a disk-shaped structuring element with diameter i. An example, D_2, is shown in Fig. 3. Note that disk-shaped structuring element is used for the city block distance transform. An example of the distance transform implemented with morphological operations is demonstrated in Fig. 4, where Fig. 4a is $BEdge$, Fig. 4b is X, Fig. 4c is the result of the distance transform, *Distance*. Note that the allowed maximum distance in this example is 4.

Figure 3 Disk-shaped structuring element with diameter 2, D_2.

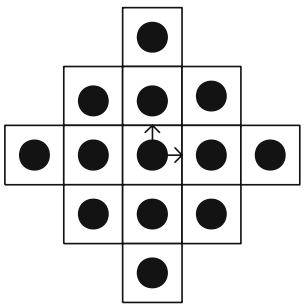


Figure 4 Distance transform with morphology operations (**a**–**c**).

0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

(a)

4	4	4	4	4
4	4	4	4	4
4	4	4	4	4
4	4	4	4	4
4	4	4	4	4

(b)

2	1	0	1	2
2	1	0	1	2
2	1	0	1	2
3	2	1	0	1
4	3	2	1	0

(c)

3.5 Edge Detector

The Canny edge detector [3] is optimized for the ability of edge detection and localization. Experiments show that edge detectors based on morphological gradient also have a good ability of edge detection; however, the edge localization is poor. It can be further improved with erosion operations using the following equations:

$$Edge = Th(GRA) \ominus D_1 - Th(GRA) \ominus D_1 \ominus D_1, \quad (8)$$

where *Th(.)* is a thresholding operation, and *GRA* is the gradient image derived with Eq. 1.

The performance comparison of this edge detector and the Canny edge detector is shown in Fig. 5, where the 50th frame of the sequence *Hall Monitor* is taken as an example. The implementation of the Canny edge detector in the OpenCV library [1] is used to generate Fig. 5a, where the first threshold is set as 133 and the second threshold is set as 399. The result of the proposed edge detector is shown in Fig. 5b with the threshold of 25. It is very similar to the result of the Canny edge detector. If we set the result of the Canny edge detector as the ground truth and assume the detected edges are true when the distance to the true edges are less than 1, the precision and recall are both 89%. Note that, since the proposed edge detector is still not equivalent to the Canny edge detector, if the Canny edge detector is still required, it is recommended to integrate another accelerator of the Canny edge detector in the proposed morphological image processing accelerator.

3.6 Edge Fitting

The edge fitting operation can be simply mapped to morphological operations with the following equation:

$$\begin{aligned}((((((CDM \oplus B_n) \underbrace{\ominus B_3; Edge) \ldots \ominus B_3; Edge)}_{l} \\ \ominus B_n) \underbrace{\oplus B_3; \overline{Edge}) \ldots \oplus B_3; \overline{Edge})}_{l} \oplus B_3 \ominus B_3, \end{aligned} \quad (9)$$

where *CDM* is change detection mask generated with change detection operation, $l = (n-1)/2$, and n is the distance range of edges to be fitted. It is a combination of dilation, conditional erosion, erosion, conditional dilation, and closing. The effect of the edge fitting operation is shown in Fig. 6.

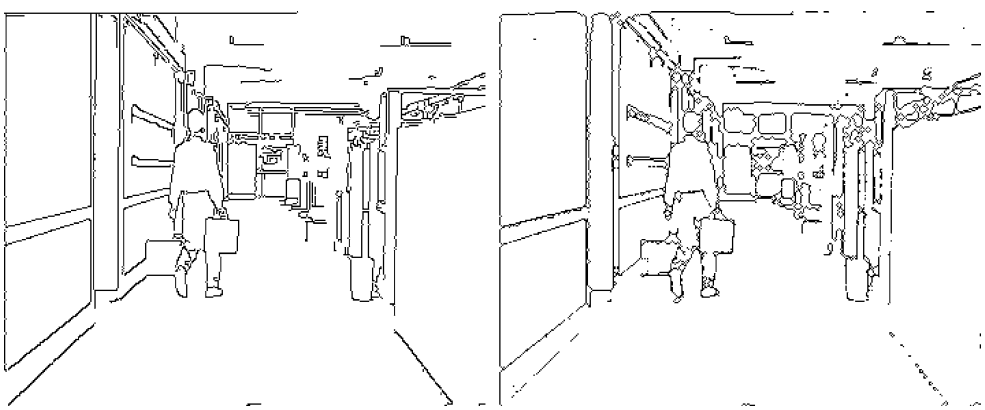


Figure 5 **a** Results of Canny edge detector. **b** Results of morphological edge detection operation.

(a) (b)

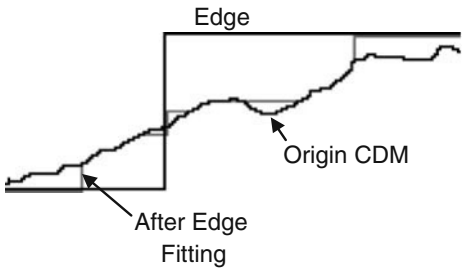


Figure 6 Results of morphological edge fitting operation.

In summary, the gradient, multiscale gradient, post-processing, watershed transform, morphological motion filter, Hausdorff distance, edge detection, and edge fitting can be mapped onto simple morphological operations: dilation, erosion, conditional erosion, conditional dilation, masked dilation, and masked erosion [26].

4 Proposed Architecture

4.1 Overview of the Proposed Hardware Accelerator

The video analysis system with the proposed reconfigurable morphological image processing accelerator is shown in Fig. 7. In order to support a variety of morphological operations, a programmable device is more suitable. Moreover, an array processor is a good choice to support high processing speed for real-time applications. Data access is usually the bottleneck for a good system performance for image/video processing systems. In order to improve the efficiency of this accelerator, the stream processing concept is employed [11], which is also employed in many stream processors. That is, the input and output data of the accelerator are modeled as streams, which consists of many stream elements in the same data type. The operations of the accelerator are separated into stream data accessing and stream data manipulation (kernel). In a stream processing model, the same kernel function is applied on the stream elements one-by-one [11], where the input stream and output stream are sequentially read-in and write-back from/to the off-chip stream buffers. The multiple *Off-Chip SDRAMs* in Fig. 7 are used as the stream buffers. In order to reduce the load on the system bus, only two 32-bit channels, one for the input stream, and the other one for the output stream, are required for the proposed accelerator. The *DMA* unit in Fig. 7 can help loading instructions from SDRAM to the *Instruction Memory* of the *Reconfigurable Morphological Image Processing Accelerator*. The *CPU* in this system is used as a controller, which is also responsible for several system tasks. Note that, for executing video analysis algorithms, the *Reconfigurable Morphological Image Processing Accelerator* only plays a role to accelerate the operations of video object segmentation. Other operations of video analysis are executed by the *Other Processors* in Fig. 7, which can be powerful DSPs, vision processors, and stream processors.

In the *Reconfigurable Morphological Image Processing Accelerator*, the modules can be classified into a stream data accessing part and a kernel part. The stream data accessing part is composed of a *Bus/DMA Interface*, *Input Stream Register File*, and *Output Stream Register File*. The *Bus/DMA Interface* can work as a master interface to access data stream from SDRAM via a *System Bus*. A part of the stream data is stored in the *Input Stream Register File*, and after the data is processed by the kernel part and stored into the *Output Stream Register File*, the output stream data is then stored back to the SDRAM. The kernel parts is composed of *Control*, *Instruction Memory*, and *Reconfigurable PE Array*. The *Control* unit decodes instructions from the *Instruction Memory* to reconfigure the *Reconfigurable PE Array* as the datapath to process the input stream data.

4.2 Instruction Set Architecture

Before designing the detailed hardware architecture, we first propose the instruction set architecture of the morphology accelerator. The instruction format is shown Fig. 8. Each instruction has 24 bits, including a 3-bit operation code (OP code), and a 21-bit operand. According to Section 3, there are five kind of operations to be executed as shown in Table 3: the "exit" operation is the end of the program, whose mnemonic is *EXT*; normal operation means normal morphological operations, such as dilation, conditional dilation, masked dilation, ... and so on, whose mnemonic is *NOR*; for watershed transform, a "loop until no change (idempotent)" operation is required, whose mnemonic is *LUN*; *STH* is the operation to set the threshold parameters for the PE array; "change PE array" operation, *CPE*, can program the rest PEs of the PE array with no operation, and the instruction behind *CPE* will be always executed by the first PE of the array, which is often used before the *STH* instruction.

The detailed instruction format of each operation is expressed in Fig. 9. For the *EXT* and *CPE* operations, the operand field is not required, as shown in Fig. 9a. For the *NOR* and *LUN* operations, the instruction format is more complicated as shown in Fig. 9b. It is very similar to very-long-instruction-word (VLIW) scheme of several modern DSPs, where several instructions

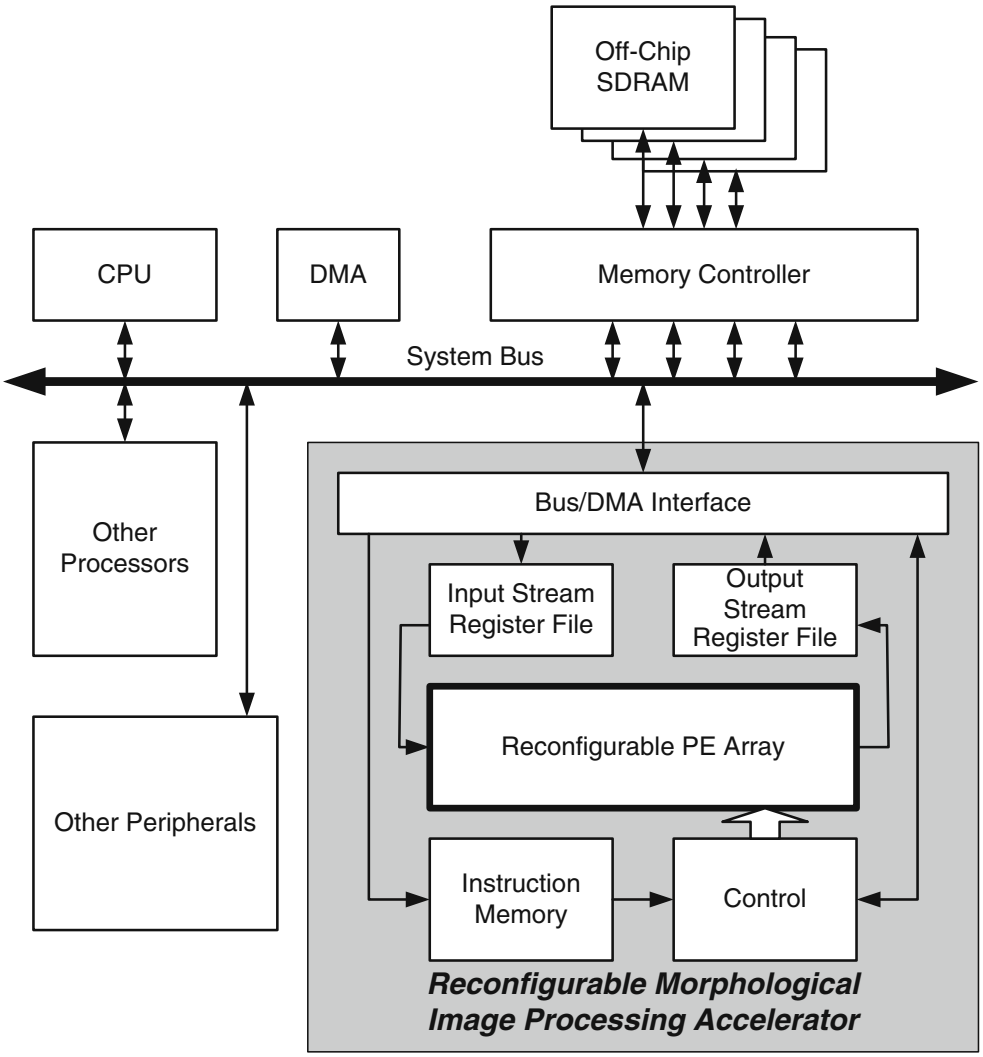


Figure 7 Overview of the proposed morphological image processing accelerator.

for different processing units are combined in a long instruction word which can be executed at the same time. Each PE can be programmed as byte mode or word mode using the 12th bit. When the 12th bit is 0, the PE can execute two 9-bit gray-level morphological operations at the same time with two sub-PEs. Bits 20–17 form the operation code for the MSB sub-PE, and bits 16–13 form the operation code for the LSB sub-PE. For each sub PE, one of the following 13 operations can be executed: no operation (*NOP*), normal eight-connected dilation (*N8D*), normal eight-connected erosion (*N8E*), normal four-connected dilation (*N4D*), normal four-connected erosion (*N4E*), masked eight-connected dilation (*M8D*), masked eight-connected erosion (*M8E*), masked four-connected dilation (*M4D*), masked four-connected erosion (*M4E*), conditional eight-connected dilation (*C8D*), conditional eight-connected erosion (*C8E*), conditional four-connected dilation (*C4D*), and conditional four-connected erosion (*C4E*). Note that eight-connected operation means the structuring element is a 3×3 square, and four-connect operation means the structuring element is a cross. When the 12th bit is 1, the PE can execute one 18-bit gray-level morphological operations, where the operation code of the two sub-PEs should be the same. Bits 11–6 are used to configure the inter-connection routing between PEs.

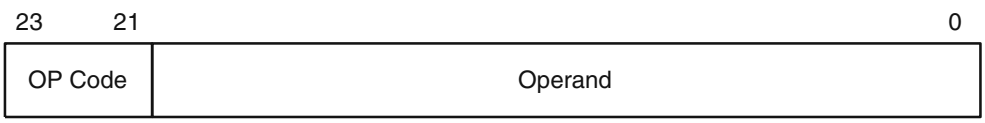


Figure 8 Instruction format of the proposed accelerator.

Table 3 Operation code of the accelerator.

OP Code	Mnemonic	Operation
000	EXT	Exit, end of the program
001	NOR	Normal mode
010	LUN	Loop until no change mode
011	STH	Set threshold
100	CPE	Change PE array, start at the first PE

For the routing switch of the MSB channel, LSB channel, and reference channel, two bits are used to represent the configurations. Details of the inter-connection unit will be described in Section 4.4. *Execution Times* is a 6-bit field which indicates how many times the operation should be executed, that it, how many PEs are required to be programmed with the instruction. The design of this field can reduce the length of the program and save the size of the instruction memory. Note that, for the *LUN* operation, this field is neglected. Finally, for the *STH* operation, two 8-bit operands are required to set the two threshold parameters in the PE array, as shown in Fig. 9c.

Based on the instructions defined in this subsection, the hardware architectures of the control unit and the PE array can be designed, which are described in the next two subsections.

4.3 Architecture of Control Unit

The control unit should be able to support zero-overhead loop, in-place operation, and self-control abilities to reduce the computational load of the host *CPU* and the traffic load of the system bus. It implies that the control unit should be able to execute instructions and control the PE array without the help of the host *CPU*. For the *LUN* operation described in Section 4.2, the control unit can watch the PE array and loop it until no change occurs.

The detailed signals to be controlled by the control unit are shown in Fig. 10. There are five main parts in the control unit: *program counter (PC)*, *instruction register (IR)*, *input address generator (Input AG)*, *output address generator (Output AG)*, and *finite state machine (FSM)*. The *PC* is used to record the pointer to the instruction to be executed in the *Instruction Memory*, and the instructions are fetched and decoded to generate control signals for each PE. The control signals, which are 15-bit for each PE, are stored in the *IR* to control the *Reconfigurable PE Array*. The *Input AG* and the *Output AG* are used to control the *Bus/DMA Interface* to read-in the input stream and write-back the output stream to the associated addresses of the off-chip stream buffers. The *FSM* is the core of the

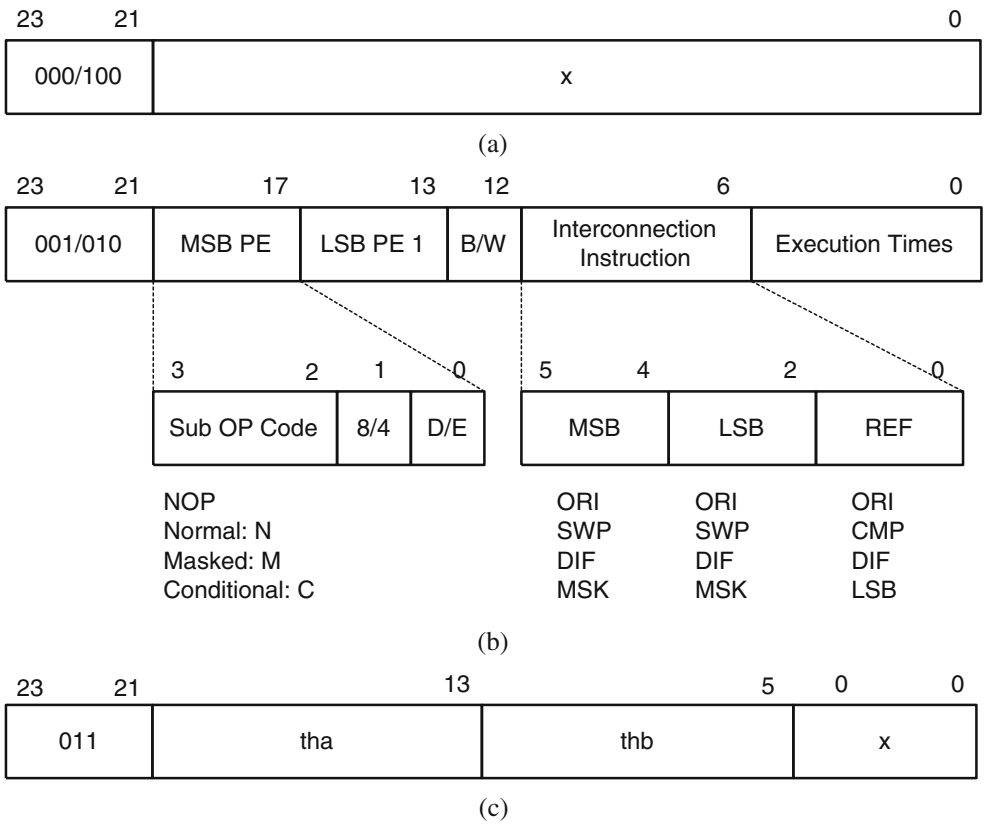


Figure 9 Instruction format of operation **a** EXT/CPE, **b** NOR/LUN, **c** STH.

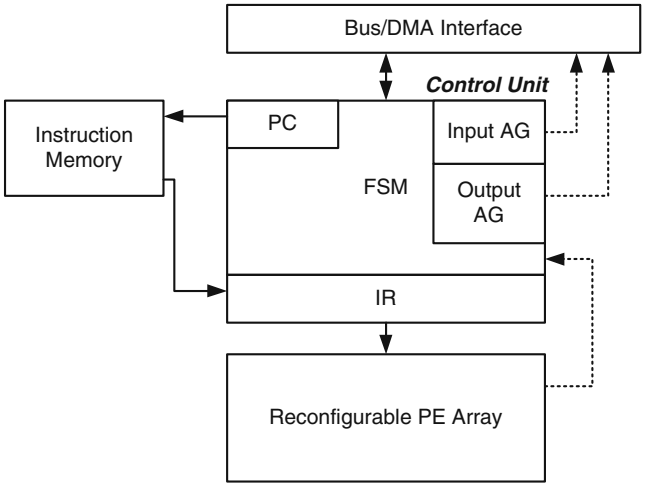


Figure 10 Detailed signals to be controlled by Control Unit.

control unit. It is responsible to control the other four parts, fetch and decode the instructions, watch if change occurs in the PE array, and communicate to the host *CPU* with the *Bus/DMA Interface*.

4.4 Architecture of Reconfigurable PE Array

The architecture of the *Reconfigurable PE Array* is shown in Fig. 11. It consists of several PEs and *Programmable Interconnection Units*. They can execute several instructions in a pipelined manner to increase the level of parallelism without increasing the requirements of the input/output data bandwidth. For each PE, a 9-bit control signal is required to decide the operation of the PE, and for each *Programmable Interconnection Unit*, a 6-bit control signal is required to configure its routing. Two global threshold parameters, *Tha* and *Thb*, can be used for thresholding and masked operation. The input and output data are 26-bit, which include a 9-bit MSB channel, a 9-bit LSB channel, and an 8-bit reference channel, which is used for thresholding, conditional operation, and masked operation, and the data of the reference channel can only be modified in the *Programmable Interconnection Unit*. If the array has eight PEs, as shown in Fig. 11, after the data flows through the array, eight morphological operations can be applied on one 18-bit image or two 9-bit images according to the control signals given from the control unit.

The detailed architecture of a single PE shown in Fig. 12a has a sub word parallel ability. In each PE, it has two 9-bit sub-PEs. A sub-PE can perform a 9-bit dilation, erosion, conditional dilation, conditional erosion, masked dilation, masked erosion, or no operation. Both the 3×3 structuring element (8-connected) and the cross-shaped structuring element (4-connected) can be supported in this architecture. Note that each sub-PE is designed with the Partial-Result-Reuse design technique [6, 14] to achieve lower hardware cost. Only four 9-bit two-input comparators (two comparators in one MAX/MIN module) are needed to implement the 3×3 morphological operations in each sub-PE. D is a delay element, or a register. W is the width of the input image. Therefore, there are two long 9-bit delay lines in each PE. *Decision Logic* can execute normal operations, conditional operations, masked operations, and no operations, with input data from the results of normal morphological operations, original data from the delay lines, reference data from the reference data channel, and mask data from *Mask Generator*. The mask data is "1" when the input of *Mask Generator* is less than or equal to *Thb* and larger than or equal to *Tha*; otherwise, the mask data is "0". Combining two sub-PEs, a PE can further perform 18-bit operations. It is obvious that this hardware architecture can execute all the operations described in Fig. 9b.

The *Programmable Interconnection Unit* between two PEs can change the inter-connection of these two PEs, which can make the proposed architecture more flexible and maintain the high throughput as well. The detailed architecture of *Programmable Interconnection Unit* is shown in Fig. 12b. There are three routing

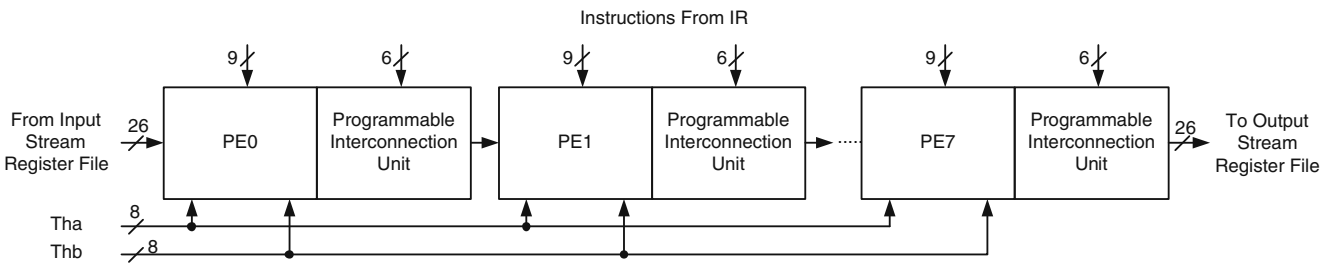


Figure 11 Architecture of programmable PE array.

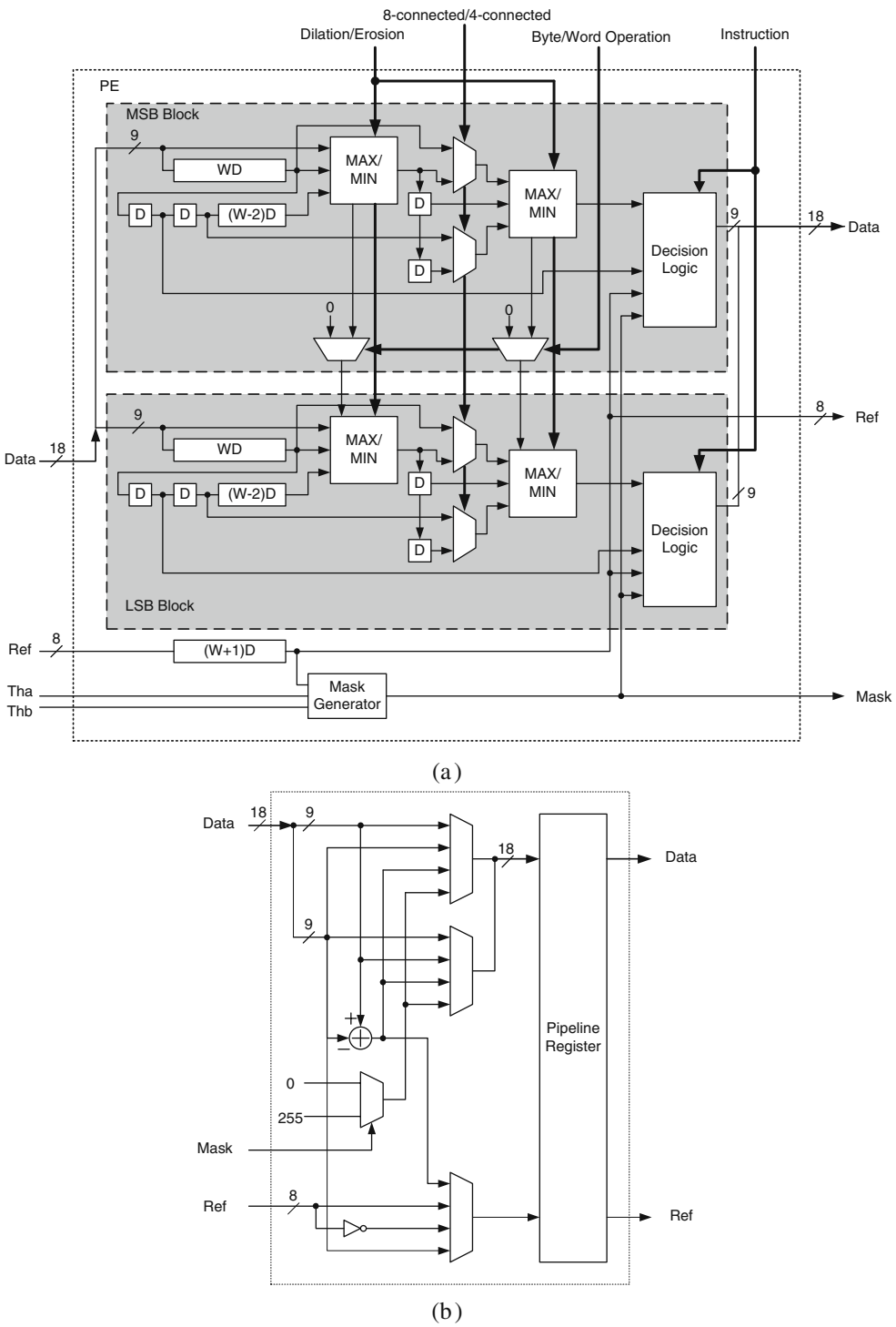


Figure 12 **a** Architecture of a single PE, note that each PE has two sub-PEs. **b** Architecture of the programmable interconnection unit.

switches (multiplexers) for a 9-bit MSB channel (MSB 9-bit of *Data*), a 9-bit LSB channel (LSB 9-bit of *Data*), and an 8-bit reference channel. Referring to the instructions described in Fig. 9b, for the MSB channel, the output of the routing switch could be the original MSB data (*ORI*), the LSB data (swapped data, *SWP*), the difference of MSB and LSB data (*DIF*), and mask data in gray-level (*MSK*). Similar behavior can be found in LSB channel. For the reference channel, the output could be the original reference data (*ORI*), the complement reference data (*CMP*), the difference of MSB and LSB data (*DIF*), and original LSB data (*LSB*). This

hardware architecture can achieve all the requirements of Section 3 and Section 4.2.

Note that the combination of a *PE* and *Programmable Interconnection Unit* is a MacroPE. In Fig. 11, there are eight MacroPEs.

4.5 Examples

Some examples are given in this subsection to make it more easier to understand how the core operations of image/video segmentation can be mapped to the proposed architecture, as shown Fig. 13 and Fig. 14. In Fig. 13a, the hardware accelerator for watershed transform is shown, where each PE is programmed to perform a 18-bit masked erosion. The associated program (firmware) is shown in Fig. 14a. Next, in Fig. 13b, the hardware for edge detection Eq. 8 is also shown. The associated firmware in Fig. 14b shows that the threshold value we use is 25. Finally, Fig. 13c demonstrates the hardware for edge fitting, and the firmware is shown in Fig. 14c, which is programmed according to Eq. 9. Note that the distance range of edges to be fitted is 5.

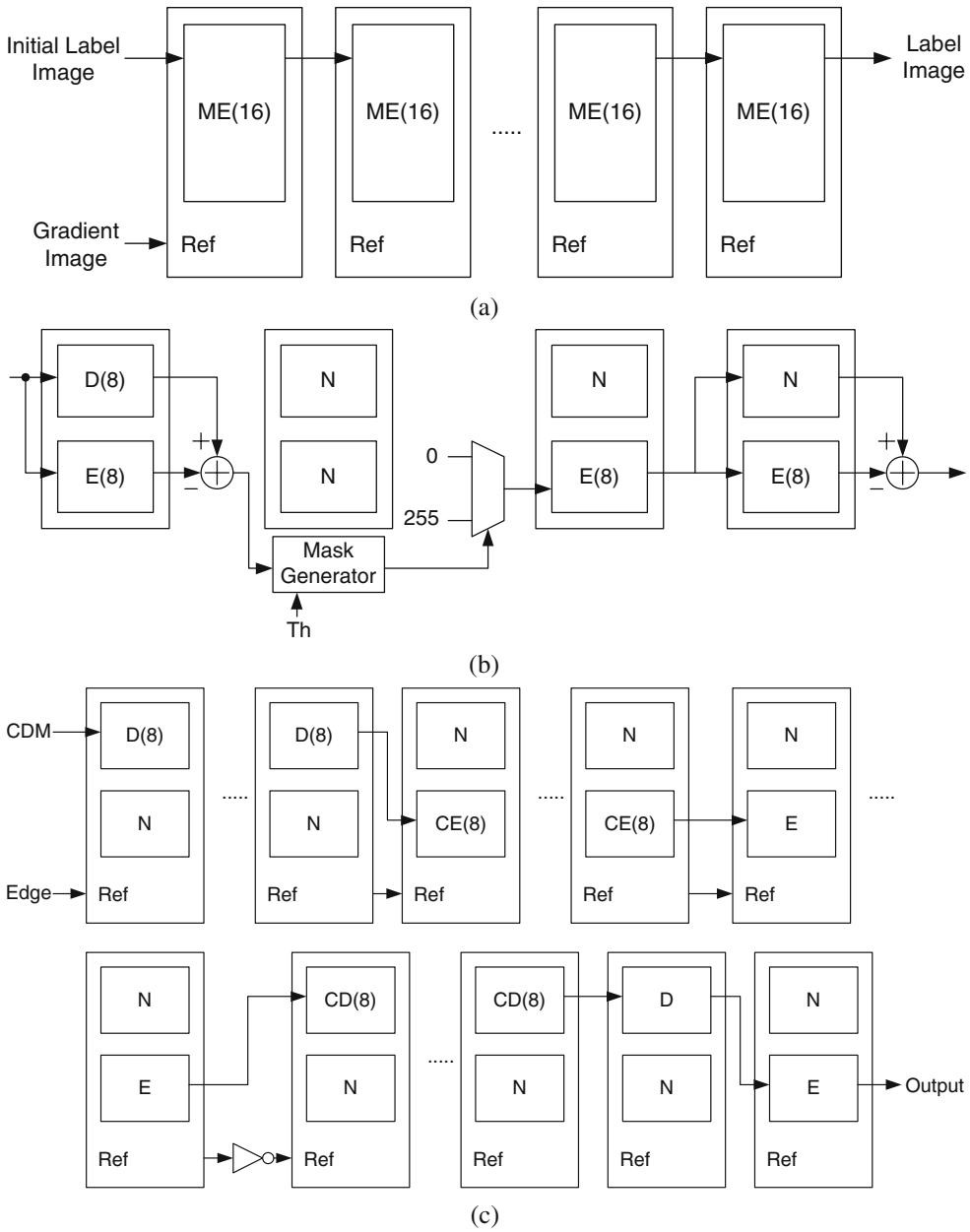


Figure 13 Example of different configuration for different operations: **a** watershed; **b** edge detection; **c** edge fitting.

```
STH 0 0
LUN M4E M4E W ORI ORI ORI 1
STH 1 1
LUN M4E M4E W ORI ORI ORI 1
STH 2 2
...
EXT
```

(a)

```
STH 25 255
NOR N8D N8E B ORI ORI DIF 1
NOR NOP NOP B MSK MSK ORI 1
NOR NOP N8E B LSB ORI ORI 1
NOR NOP N8E B ORI LSB ORI 1
EXT
```

(b)

```
NOR N8D NOP B ORI ORI ORI 4
NOR N8D NOP B ORI SWP ORI 1
NOR NOP C8E B ORI ORI ORI 5
NOR NOP N8E B ORI ORI ORI 4
NOR NOP N8E B SWP ORI CMP 1
NOR C8D NOP B ORI ORI ORI 5
NOR N8D NOP B ORI SWP ORI 1
NOR NOP N8E B ORI ORI ORI 1
EXT
```

(c)

Figure 14 Firmware for **a** watershed transform, **b** edge detection, and **c** edge fitting.

5 Architecture and Algorithm Optimization

In this section, some detailed architecture and algorithm optimization issues for implementation are discussed, including delay line issues, hardware cost for real-time watershed transform, and the pipelined-parallel architecture. With the proposed implementation methods, the hardware cost can be further saved, or the performance can be further enhanced.

5.1 Delay Line Issues

As shown in Fig. 12a, four 9-bit delay lines (the two WD lines and $(W-2)D$ lines) and one 8-bit delay line (the $(W+1)D$ line) are required for each PE. For a large frame size and a large number of PEs, the hardware cost of the delay lines will dominate the whole chip; therefore, the delay lines should be careful designed. The first way to reduce the hardware cost of a delay line is to implement the delay line with dual-port on-chip SRAM and a dedicated counter rather than shift registers.

Another technique to shorten the delay line is the tiling technique [14], which can be illustrated in Fig. 15. The input image is firstly divided into several tiles, for example, four tiles in Fig. 15a. For each tile, a padding technique is then employed to generate a padded region, as shown in Fig. 15b with the following rule.

For every point (x, y) in the padded region,

if $(x, y) \in I$, the padded value $P(x, y) = I(x, y)$,

otherwise, $P(x, y) = I(i, j)$,

where $(i, j) \in I$ and

$$distance[(x, y), (i, j)] < distance[(x, y), (m, n)] \forall (m, n) \in I \neq (i, j)$$

The purpose of the padded region is to provide a guard region, in which the error propagation from the boundary can occur. After the tile is processed, the padded region is removed before begin storing to memory. With the padding technique, the boundary conditions

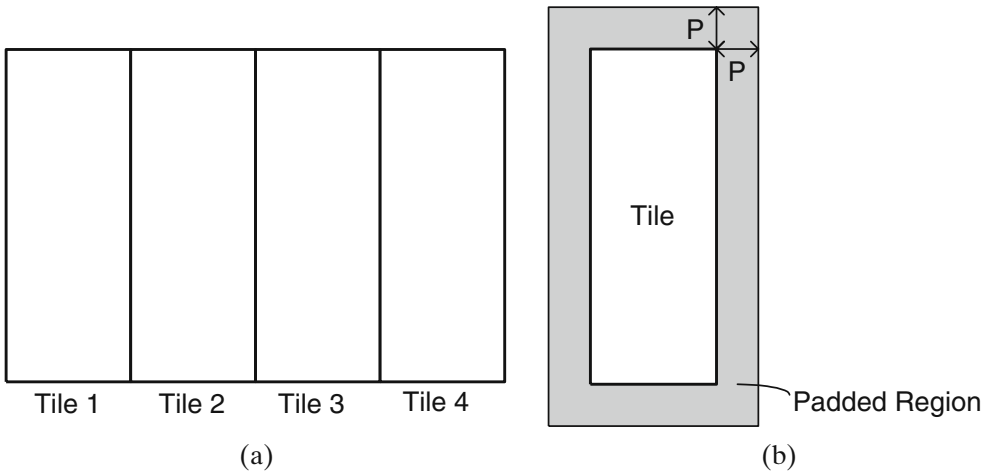


Figure 15 Tiling technique. **a** The input image can be divided into several tiles. **b** The padding region of each tile.

of morphological operations can be handled well without complicated control circuits. After all the tiles are processed, the result is the same as when the whole frame is processed at the same time; however, the length of the delay line becomes about a quarter of the original one, and three-fourths of the hardware cost can be saved. The overhead of the tiling technique is the increasing complexity of addressing and the longer latency because of the padded regions.

5.2 Simplified Watershed Transform

The watershed transform discussed in Section 3.3 requires a large amount of morphological operations for the "loop until no change" (LUN) operations, which implies the number of PEs should be very large for the real-time watershed transform. To reduce the hardware cost, some simplifying techniques for the watershed transform are proposed and discussed in this subsection. The first frame of the sequence *Table Tennis* is used as an example.

If the firmware shown in Fig. 14a is applied directly on the gradient image, severe over-segmentation will occur, as shown in Fig. 16a. 1,067 3×3 morphological operations are needed to be employed to the whole image to get the result. The clipping technique can solve this problem with clipping gradient values less than or equal to a threshold, $ClipTH$, to zero. An example with $ClipTH = 7$ is shown in Fig. 16b, where 1,344 operations are required. The associated firmware is shown in Fig. 17a. In order to achieve the real-time requirement, 40,320 morphological operations are needed to be applied on the whole frame in one second, which will require a large amount of PEs and an enormous hardware cost. A $\log_2$ anamorphosis [24] can reduce the 256 gray-levels to eight levels. Taking into account both the clipping technique and $\log_2$ anamorphosis technique, we can group the 256 gray-levels into six levels: 0–7, 8–15, 16–31, 32–63, 64–127, and 128–256. The result is demonstrated in Fig. 16c, which is very similar to Fig. 16b, with only 756 operations. Note that instead of adding a look-up-table (LUT) hardware in the system [24], the $\log_2$ anamorphosis technique can be easily employed by changing the firmware, as shown in Fig. 17b.

From the experiments we found that the first level of watershed transform often cost the largest number of morphology operations. For example, in a test image shown in Fig. 16, among the 756 operations of the watershed transform with anamorphosis technique, 653 operations are executed in the first level, since the area belongs to the first level is always the biggest. A local flooding scheme is proposed to reduce the computation in the first level. It can be illustrated by Fig. 18. It is a divide-and-conquer scheme, where the input image is divided into several slices as shown in Fig. 18. For

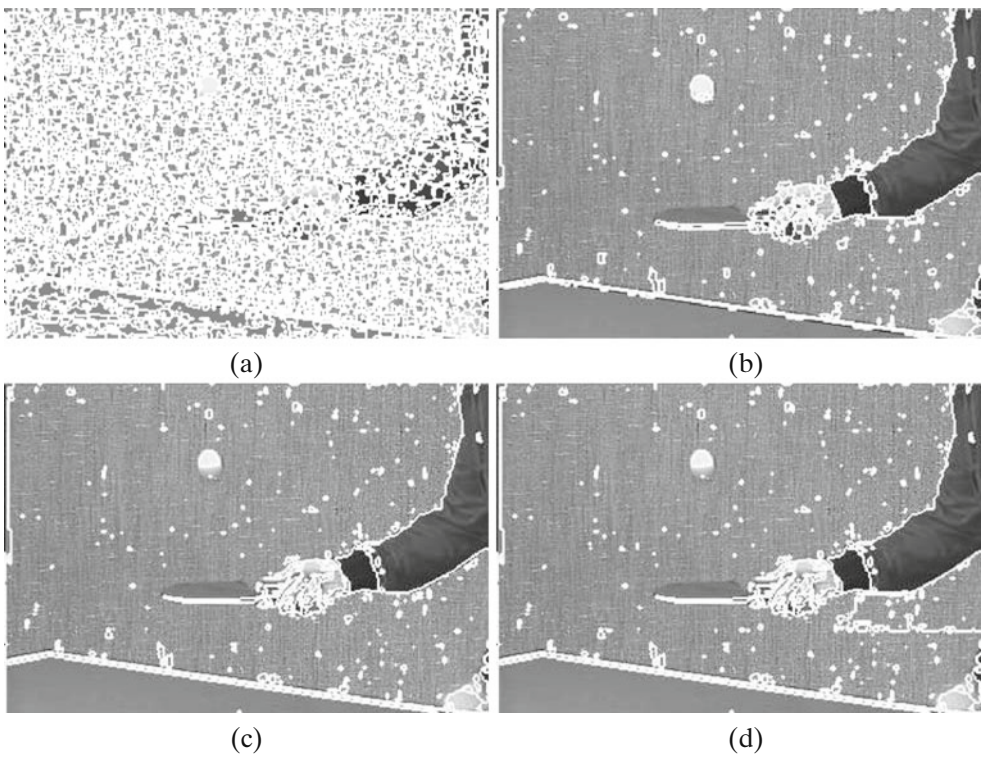


Figure 16 Simplified watershed transform. **a** The origin result of watershed transform, where over-segmentation is severe. 1,067 operations are required. **b** Watershed with gradient values lower or equal to seven clipped to zero. 1,344 operations are required. **c** Watershed with $\log_2$ anamorphosis gradient values. 756 operations are required. **d** Watershed with $\log_2$ anamorphosis gradient values and local flooding scheme. 402.5 operations are required.

```
STH 0 7
LUN M4E M4E W ORI ORI ORI 1
STH 8 8
LUN M4E M4E W ORI ORI ORI 1
STH 9 9
...
EXT
```

(a)

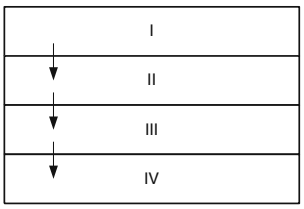
```
STH 0 7
LUN M4E M4E W ORI ORI ORI 1
STH 8 15
LUN M4E M4E W ORI ORI ORI 1
STH 16 31
LUN M4E M4E W ORI ORI ORI 1
STH 32 63
LUN M4E M4E W ORI ORI ORI 1
STH 64 127
LUN M4E M4E W ORI ORI ORI 1
STH 128 255
LUN M4E M4E W ORI ORI ORI 1
EXT
```

(b)

Figure 17 Firmware for **a** watershed transform with clipping gradient values lower or equal to seven to zero. **b** Watershed with $\log_2$ anamorphosis gradient values.

the first level, firstly, the masked erosion operation is applied only in region I until no change occurs, then region II, and then region III. Finally, after the masked erosion operation is applied in region IV, the next level is processed as the way before for the whole image. An example is shown in Fig. 16d, which is very similar to Fig. 16c, except the over-segmentation problem may occur near the boundary of the slices. The number of operations to be applied in a quarter of the input frame in the first level is 1,110, whose computation complexity is equal to $1{,}110/4 = 277.5$ 3×3 morphological operations for a whole frame. For other levels, only 125 operations are required with the $\log_2$ anamorphosis technique, namely, only $277.5 + 125 = 402.5$ operations are required. In summary, the local flooding scheme can dramatically reduce the computation complexity without degrading the result of watershed transform. Note that, to implement the local flooding scheme in the proposed morphological image processing accelerator, one only needs to change the parameters of the control registers in the control unit from the system bus.

Figure 18 Illustration of local flooding scheme.



It also shows that the proposed reconfigurable hardware morphological image processing accelerator is very flexible. By changing the firmware and the values in the control registers, new watershed algorithms can be employed without re-designing the hardware.

5.3 Pipelined-Parallel Architecture

The architecture of the programmable PE array shown in Fig. 11 is a pipelined architecture. A pipelined architecture can increase the processing speed with fixed input and output data rate; however, the internal memory size and the latency would become large as the number of PEs increases. On the other hand, a parallel architecture can increase the processing speed with fixed internal memory size; however, the input and output data rate would increase so the required bit-width of the system bus would also increase.

In order to make a good balance between internal memory size and input/output data rate, when a large number of PEs is required, we also propose a pipelined-parallel array architecture, which is demonstrated in Fig. 19. In this figure, there are four pipelined PE arrays, which are similar to the array in Fig. 11, working in parallel. For each PE, the length of the delay line is a quarter of the width of the input image; therefore, the on-chip memory requirement of the pipelined-parallel architecture is similar to that of the pipelined architecture shown in Fig. 11 when the tiling technique discussed in Section 5.1 is not employed. On the other hand, the processing speed of the pipelined-parallel architecture is four times faster since each pipelined PE array can manipulate a tile in Fig. 15, namely, four tiles are processed simultaneously, so only a quarter of the clock cycles are required for this architecture to process a frame. Note that, since the input and output data rate becomes four times that of the pipelined architecture, the frequency to accessing the system bus should be faster than the working frequency of the PE array to keep the balance between the memory bandwidth and the processing speed.

When the real-time watershed transform for SIF frames is taken into consideration, with the simplified watershed transform proposed in the previous subsection, 32 MacroPEs are required when the working frequency is 40 MHz, and the working frequency of the system bus is 120 MHz. The hardware architecture of the reconfigurable PE array is shown in Fig. 19.

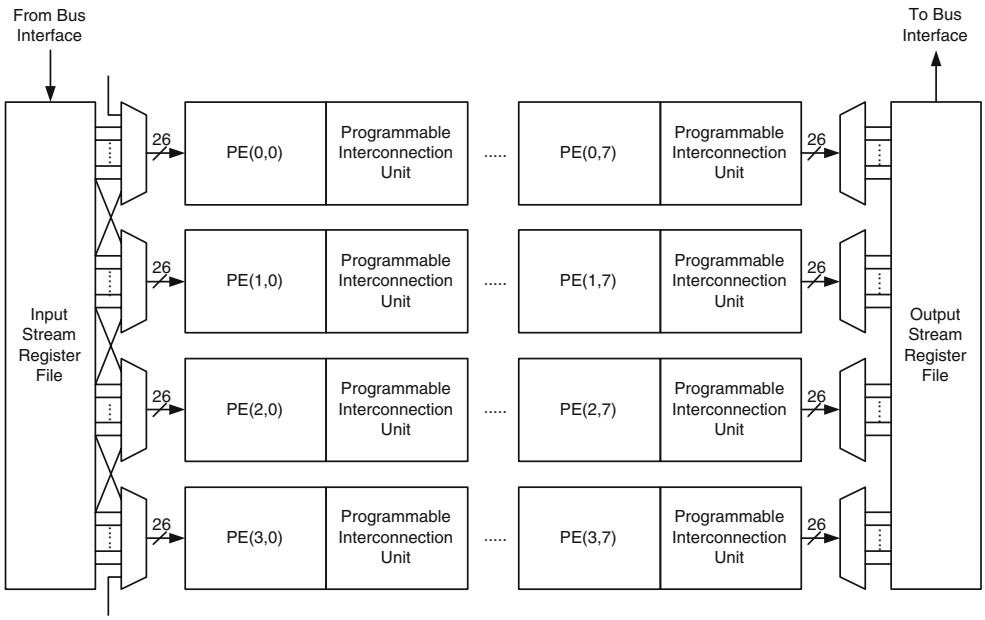


Figure 19 Pipelined-parallel architecture the reconfigurable PE array.

6 Implementation Results and Analysis

6.1 Synthesis Results

The result of hardware implementation is shown in Table 4, where the circuit is synthesized with the SYNOPSYS™ Design Compiler. The implementation results of an eight-MacroPE pipelined architecture with tiling technique and a 32-MacroPE pipelined-parallel architecture are listed in this table. The operation frequency is targeted to 40 MHz. With the eight-MacroPE pipelined architecture, a processing speed of 3,100 18-bit morphological operations per second for a SIF image or 6,200 9-bit morphological operations per second for a SIF image can be achieved, which is sufficient for real-time video segmentation [7]. With a 32-MacroPE pipelined-parallel architecture, for a SIF image, the processing speed of 12,400 18-bit morphological operations per second or 24,800 9-bit morphological operations per second can be achieved, which is sufficient for a real-time watershed transform with the proposed simplified techniques. The number of PEs can be further reduced if the target frame size and the frame rate is reduced, or when the target operation frequency is increased. The internal memory size is only 5% and 20% of a frame memory for the eight-MacroPE version and the 32-MacroPE version, respectively, that is, no internal frame buffer is required in this architecture to achieve high throughput.

Table 4 Result of hardware implementation.

Unit	Gate count	Internal memory size
Single MacroPE	4,441	4,576b
PE	3,901	4,576b
Programmable Interconnection unit	533	0b
Control unit	6,614	0b
Total (8 MacroPEs)	41,519	36,608b
Total (32 MacroPEs)	148,726	146,432b

6.2 Prototype Chip Implementation

The prototype chip layout of the proposed morphological image processing accelerator with *Control Unit* and *Reconfigurable PE Array* with eight MacroPEs is demonstrated in Fig. 20. Half of the chip area is occupied by the on-chip memory. The specification of the chip is shown in Table 5. The technology is TSMC 0.25 μm 1P5M. The chip size is 4.76×5.74 mm^2. Note that, this chip is not fabricated. All the numbers come from the post-layout simulation.

6.3 System Performance

The estimated system performance is shown in Table 6. The host computer is a low-end computer with a Celeron 300 MHz microprocessor, and the system bus is a PCI bus. Assume that no other peripheral shares the system bus with the accelerator. Our algorithm and the watershed transform are applied on a SIF sequence. It shows that without the hardware accelerator, the processing speed is far behind real-time requirement (33 ms/frame). With the accelerator, the processing

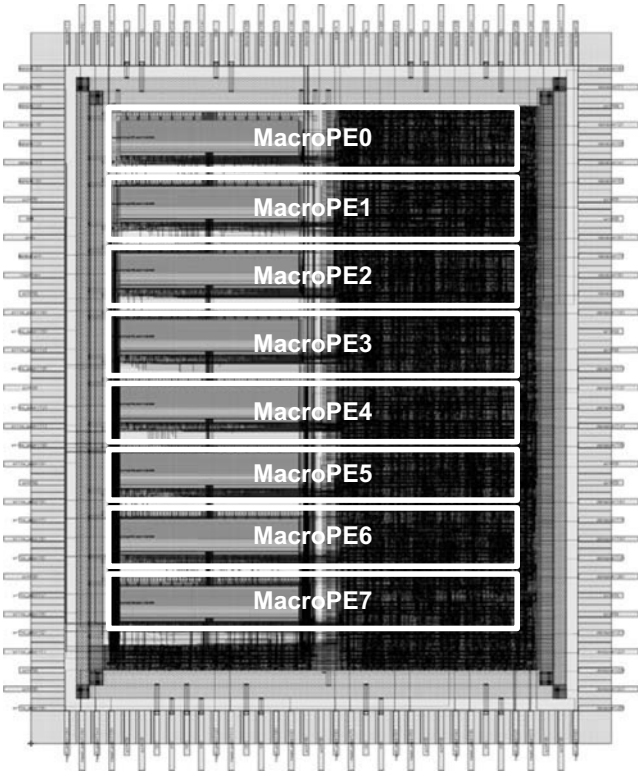


Figure 20 Layout of the prototyping chip.

time of our algorithm is 11.99 ms, and the processing time of watershed transform is 31.41 ms, which can achieve real-time requirement. Note that the processing time should be the maximum of the processing time of the software part and the hardware part since the host CPU and the accelerator can function at the same time. It is shown that even the host is a low-end computer, the system can still achieve the real-time requirement with the proposed hardware accelerator.

Table 5 Features of the prototyping chip.

Technology	TSMC 0.25 μm 1P5M
Package	144 CQFP (140 Pads)
Chip size	4.76 mm × 5.74 mm
Power supply	2.5 V
Power consumption	600 mW @ 40 MHz
Gate count (without memory)	41,519
Transistor count (with memory)	845,999
Processing speed@40 MHz	30 SIF frames/s (video segmentation with [7])
	3,100 18-bit 3 × 3 morphological operations second on a SIF image
	6,200 9-bit 3 × 3 morphological operations per second on a SIF image
On-chip memory	8 128×48 dual-port RAM

Table 6 System performance estimation.

Algorithm	Software only (ms/frame)	HW/SW co-work SW part (ms/frame)	HW/SW co-work HW part (ms/frame)
Ours [7]	223.24	11.99	9.98
Watershed	453.24	25.54	31.41

6.4 Discussion of the Reconfigurable Architecture

Table 7 shows the detailed hardware cost for each component of a MacroPE. The memory control unit is used to control the dual-port on-chip RAM as delay lines. The essential logic is the logic to implement two dedicated 9-bit parallel morphological operations, where only 9-bit comparators, 9-bit registers, and the memory control unit are included. On the other hand, the programmable logic is used to give the programmability to the PE, where the mask generation unit, programmable logic in PE, and the interconnection unit are included. It is shown that the overhead of programmability, only about half of the essential logic, is acceptable. With this overhead, 2^{15} different configurations can be achieved, where the dedicated hardware has only one configuration. Therefore, the programmable logic is hardware-cost-efficient since it can provide high programmability with acceptable hardware cost overhead to accelerate various morphological operations to support different kinds of segmentation algorithms. Besides, the programming time of this reconfigurable hardware is fast since it can be configured with firmware and the control unit can reconfigure the array without any help from the host CPU and the system bus.

Table 8 shows the comparison between the proposed architecture with other watershed architectures. In Kuo's architecture [16], only the core of the flooding operation is implemented, which means that the gradient, sorting, and the priority queue is not included

Table 7 Detailed hardware cost for each component of a MacroPE.

Component	Gate count
9-bit Comparator[a]	984
9-bit Register[b]	368
Memory control	1,641
Mask generation	95
Programmable logic in PE	820
Interconnection unit	533
Essential logic	2,993
Logic for programmability	1,448
Total	4,441

[a]Eight two-input comparator.
[b]Eight registers for partial-result-reuse.

Table 8 Comparison with other watershed architecture.

Architecture	Software control loading	Gate count	Achieve real-time	Programmability
[16]	Heavy	2,965	Hard	No
[22]	Light	3,548,160	Yes	No
This work	Light	148,726	Yes	Yes

in the hardware. It is implied that the software control loading is high, and it is hard to achieve real-time requirement. Noguet's work [22] use a large array processor for watershed transform, where each pixel is processed with a PE. It can achieve real-time; however, the hardware cost is enormous. Moreover, the gradient operation is not implemented in this architecture. Aid from software is still required. The proposed reconfigurable morphological image processing accelerator can also achieve real-time requirement. Since the gradient operation and flooding operation can be both implemented in this reconfigurable accelerator, and the accelerator supports self-control ability, the software control loading is lighter. In addition, the hardware cost is much smaller than Noguet's work [22]. Therefore, the proposed accelerator can support both gradient and flooding operations with less gate count. That is because different operations can share the same reconfigurable hardware resource in different time slots, and the hardware resource can also be shared spatially for every pixel with off-chip memories to store the partial results. It is obvious that the proposed reconfigurable architecture is more feasible and more efficient.

6.5 Comparison with Previous 1-D Array Processor Architectures

Several 1-D array processor architectures have been proposed. Here, the proposed reconfigurable morphological image processing accelerator is compared with these previous works. Warp processor is a systolic array computer [2], where processors are connected with a pre-defined topology, and the programs for all the processors are identical. Several concepts of stream processing is proposed with Warp processor. However, it is designed for general-purpose applications and is not optimized for image processing. CLIP7A is another 1-D array processor [8]. It is an SIMD array with autonomy, and the interconnection between processors can be flexibly changed for different modes. CLIP7A is also designed for general-purpose applications. IMAP-CE is another SIMD 1-D array processor [17]. It is designed for video recognition, and it is composed of 128 VLIW processors and can achieve a high processing capability of 51.2-GOPS. In order to achieve a high hardware utilization of the SIMD architectures, a high performance memory system is needed to feed-in the required data for each processing element. For example, large shift register and line buffer arrays are designed for IMAP-CE, which leads to a large hardware cost. That is, the high performance comes at much higher hardware cost and power consumption than the proposed morphological image processing accelerator.

Two previous works are highly related to the proposed architectures, where similar design concepts are also supported. Cytocomputer is a pipeline image processor [18], where delay lines are also included in each processor to improve the efficiency of image data accessing. It also analyzes the advantages of linear array than other 2-D mesh arrays: the I/O bandwidth is more reasonable, and the required processing time may be shortened when the time of data-loading is also considered. However, the design of processing elements are not mentioned in Cytocomputer, and the interconnections between the processors are fixed, which limits the target applications to only simple systolic image processing algorithms. On the other hand, PipeRench is a general-purpose reconfigurable array architecture [10], where the concept of reconfigurable interconnection is proposed, but it is not optimized for image processing.

Compared with these works, the proposed reconfigurable morphological image processing accelerator is also a 1-D array architecture. The architectures of the processing elements and interconnection units are optimized for morphological image processing. In order to lower the I/O bandwidth requirement, it is not an SIMD array processor [8, 17]. The high performance is achieved by many processors working in a pipeline with different assigned works, like Cytocomputer [18]. Besides, the flexibility of the dataflow is achieved by the reconfigurable interconnection [10].

7 Conclusion and Future Works

A hardware accelerator for video object segmentation is proposed in this paper. From the analysis of existing video segmentation algorithms, we find that most of the core operations can be implemented with different morphological operations. Therefore, the proposed accelerator is based on a reconfigurable morphological image processing PE array, and a stream processing concept is employed in the architecture design to increase the efficiency. Many examples are demonstrated

to show that this instruction set architecture is very suitable to be used to accelerate the core operations of video segmentation. Simulation shows that this accelerator can accelerate most important video segmentation algorithms to achieve real-time. It can be used to accelerate the change detection and background registration based video segmentation with eight MacroPE and can achieve a real-time watershed transform with 32 MacroPEs at 40 MHz. It is also shown that the hardware cost of the proposed architecture is low because the memory requirement can be reduced with the proposed tiling technique and pipelined-parallel architecture, and the hardware resource can be shared not only spatially but also temporally with the reconfigurability of this architecture.

There are several limitations for this reconfigurable morphological image processing accelerator. First of all, only flat rectangle or disk-shaped structuring elements are supported. It will limit the usage of other powerful morphological operations, especially for binary image analysis, such as the hit-or-miss operator [26, 28]. However, we believe that the supported morphological operations are enough for the applications of image/video segmentation. In addition, the threshold value is fixed in this accelerator and can still be changed for each frame by cooperating with the host CPU. The locally adaptive threshold is not supported in our design. Moreover, there are still some operations of video segmentation cannot be well implemented with morphological operations. For these operations, it is recommended to integrate other hardware accelerators into the system, or they can also be implemented with the CPU or other processors in the system.

In the current stage, the proposed reconfigurable PE array is designed to accelerate the morphology operations for video object segmentation. We believe it can be further extended to support more applications and other operations such as general region growing operations. In addition, the proposed PE architecture can be considered to be integrated into other vision processors to enhance the performance for executing morphology operations. Furthermore, the proposed accelerator can only be used to accelerate morphology operations. To design a complete image/video analysis system, other processors and accelerators are required to be integrated, which will be considered as our future works.

Acknowledgements The authors would like to thank chip implementation center (CIC) for EDA tool and design flow support.

References

1. sourceforge.net (2008). Open computer vision library (OpenCV). http://sourceforge.net/projects/opencvlibrary/.
2. Annaratone, M., Arnould, E., Gross, T., Kung, H. T., & Lam, M. S. (1986). Warp architecture and implementation. In *Proc. international symposium on computer architecture* (pp. 346–356).
3. Canny, J. (1996). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8*(6), 679–698.
4. Chien, S. Y., Huang, Y. W., & Chen, L. G. (2003). Predictive watershed: A fast watershed algorithm for video segmentation. *IEEE Transactions on Circuits and Systems for Video Technology, 13*(5), 453–461.
5. Chien, S. Y., Huang, Y. W., Hsieh, B. Y., Ma, S. Y., & Chen, L. G. (2004). Fast video segmentation algorithm with shadow cancellation, global motion compensation, and adaptive threshold techniques. *IEEE Transactions on Multimedia, 6*(5), 732–748.
6. Chien, S. Y., Ma, S. Y., & Chen, L. G. (2001). A partial-result-reuse architecture and its design technique for morphological operations. In *Proc. of 2001 IEEE international conference on acoustics, speech, and signal processing* vol. 2, (pp. 1185–1188).
7. Chien, S. Y., Ma, S. Y., & Chen, L. G. (2002). Efficient moving object segmentation algorithm using background registration technique. *IEEE Transactions on Circuits and Systems for Video Technology, 12*(7), 577–586.
8. Fountain, T. J., Matthews, K. N., & Duff, M. J. B. (1988). The CLIP7A image processor. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 10*(3), 310–319.
9. Garrido, L., Oliveras, A., & Salembier, P. (1997). Motion analysis of image sequences using connected operators. In *Proc. of visual communication and image processing* (pp. 546–557).
10. Goldstein, S., Schmit, H., Budiu, M., Cadambi, S., Moe, M., & Taylor, R. (2000). PipeRench: A reconfigurable architecture and compiler. *IEEE Computer, 33*(4), 70–77.
11. Kapasi, U., Rixner, S., Dally, W., Khailany, B., Ahn, J. H., Mattson, P., et al. (2003). Programmable stream processors. *Computer, 36*(8), 54–62.
12. Kim, C., & Hwang, J. N. (2002). Fast and automatic video object segmentation and tracking for content-based applications. *IEEE Transactions on Circuits and Systems for Video Technology, 12*(2), 122–129.
13. Kim, M., Choi, J. G. D., Kim, H. L., Lee, M. H., Ahn, C., & Ho, Y. S. (1999). A VOP generation tool: Automatic segmentation of moving objects in image sequences based on spatio-temporal information. *IEEE Transactions on Circuits and Systems for Video Technology, 9*(8), 1216–1226.
14. Kishore, A. D., & Srinivasan, S. (2003). A distributed memory architecture for morphological image processing. In *Proc. international conference on information technology: Coding and computing* (pp. 536–540).
15. Klein, J. C., & Peyrard, R. (1989). Pimm1, an image processing ASIC based on mathematical morphology. In *Proc. of second annual IEEE ASIC seminar and exhibit*.
16. Kuo, C. J., Odeh, S. F., & Huang, M. C. (2001). Image segmentation with improved watershed algorithm and its FPGA implementation. In *Proc. of the 2001 IEEE international symposium on circuits and systems*, vol. 2, (pp. 753–756).
17. Kyo, S., Koga, T., Okazaki, S., & Kuroda, I. (2003). A 51.2-GOPS scalable video recognition processor for intelligent

cruise control based on a linear array of 128 four-way VLIW processing elements. *IEEE Journal of Solid-State Circuits, 38*(11), 1992–2000.
18. Lougheed, R. M., & McCubbrey, D. L. (1980). The Cytocomputer: A practical pipelined image processor. In *Proc. international symposium on computer architecture* (pp. 217–277).
19. Mech, R., & Wollborn, M. (1998). A noise robust method for 2D shape estimation of moving objects in video sequences considering a moving camera. *Signal Processing, 66*(2), 203–217.
20. Meier, T., & Ngan, K. N. (1998). Video segmentation for content-based coding. *IEEE Transactions on Circuits and Systems for Video Technology, 9*(8), 1190–1203.
21. MPEG Video Group (2001). Annex F: Preprocessing and postprocessing. ISO/IEC JTC 1/SC 29/WG11 N4350.
22. Noguet, D. (1997). A massively parallel implementation of the watershed based oncellular automata. In *Proc. of the 1997 IEEE international conference on application-specific systems, architecture and processors* (pp. 42–52)
23. Peleg, A., & Weiser, U. (1996). MMX technology extension to the Intel architecture. *IEEE Micro, 16*(4), 42–50.
24. Peyrard, R., Gauthire, M., & Klein, J. C. (1994). Real-time road segmentation using a morphological multi-pipeline processor. In *Proc. of the intelligent vihecles symposium* (pp. 290–295).
25. Salembier, P., Oliveras, A., & Garrido, L. (1998). Antiextensive connected operators for image sequence processing. *IEEE Transactions on Image Processing, 7*(4), 555–570.
26. Serra, J. (1982). *Image analysis and mathematical morphology*. London: Academic.
27. Shamim, A., & Robinson, J. A. (2002). Object-based video coding by global-to-local motion segmentation. *IEEE Transactions on Circuits and Systems for Video Technology, 12*(12), 1106–1116.
28. Soille, P. (2007). *Morphological image analysis*. New York: Springer.
29. Tsai, Y. P., Lai, C. C., Hung, Y. P., & Shih, Z. C. (2005). A Bayesian approach to video object segmentation via merging 3-D watershed volumnes. *IEEE Transactions on Circuits and Systems for Video Technology, 15*(1), 175–180.
30. Tsaig, Y., & Averbuch, A. (2002). Automatic segmentation of moving objects in video sequences: A region labeling approach. *IEEE Transactions on Circuits and Systems for Video Technology, 12*(7), 597–612.
31. Vincent, L., & Soille, P. (1991). Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligece, 13*(6), 583–598.
32. Wang, D. (1998). Unsupervised video segmentation based on watersheds and temporal tracking. *IEEE Transactions on Circuits and Systems for Video Technology, 8*(5), 539–546.
33. Wiatr, K. (1998). Pipeline architecture of specialized reconfigurable processors in FPGA structures for real-time image pre-processing. In *Proc. of 24th Euromicro conference* (pp. 25–27).
34. Wiatr, K. (2002). Median and morphological specialized processors for a real-time image data processing. *EURASIP Journal on Applied Signal Processing, 2002*(1), 115–121.
35. Xu, H., Younis, A. A., & Kabuka, M. R. (2004). Automatic moving object extraction for content-based applications. *IEEE Transactions on Circuits and Systems for Video Technology, 14*(6), 796–812.
36. Zhang, D., & Lu, G. (2001). Segmentation of moving objects in image sequence: A review. *Circuits Systems Signal Processing, 20*(2), 143–183.

Shao-Yi Chien received the B.S. and Ph.D. degrees from the Department of Electrical Engineering, National Taiwan University (NTU), Taipei, Taiwan, R.O.C. in 1999 and 2003, respectively. During 2003 to 2004, he was a research staff in Quanta Research Institute, Tao Yuan Shien, Taiwan, R.O.C. In 2004, he joined the Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, where he is now an Associate Professor. His research interests include video segmentation algorithm, intelligent video coding technology, image processing, computer graphics, and associated VLSI architectures.

Liang-Gee Chen received the B.S., M.S., and Ph.D. degrees in Electrical Engineering from National Cheng Kung University, Tainan, Taiwan, R.O.C. in 1979, 1981, and 1986, respectively. In 1988, he joined the Department of Electrical Engineering, National Taiwan University. Currently, he is the Distinguished Professor of Department of Electrical Engineering and the Deputy Dean of office of Research and Development in National Taiwan University, Taipei, Taiwan, R.O.C. Since 2007, he also serves as a Co-Director General of National SoC Program. He is the IEEE Fellow from 2001. His research interests include DSP architecture design, video processor design, and video coding systems. He has published over 350 papers and 30 patents.

Dr. Chen has served as an Associate Editor of IEEE Transactions on Circuits and Systems for Video Technology and other international technical journals. He is also involved several IEEE technical committees, including the TPC Chair of 2009 IEEE ICASSP and the TPC chair of ISCAS 2012. He has received several outstanding research awards and outstanding industrial technology contribution awards from NSC. His group has won the DAC/ISSCC Student Design Contest for three times since 2004, and had the honor of Student Paper Contest at ICASSP 2006.