A HARDWARE ACCELERATOR FOR VIDEO SEGMENTATION USING PROGRAMMABLE MORPHOLOGY PE ARRAY

Shao-Yi Chien, Yu-Wen Huang, and Liang-Gee Chen*

DSP/IC Design Lab, Department of Electrical Engineering
National Taiwan University
1, Sec. 4, Roosevelt Rd., Taipei 106, Taiwan
{shoayi, yuwen, lgchen}@video.ee.ntu.edu.tw

ABSTRACT

Video segmentation is a key operation for MPEG-4 and MPEG-7. It is a computationally intensive task and cannot be afforded by general microprocessors. In this paper, we propose a hardware accelerator for video segmentation. Since most of the core operations of video segmentation algorithms can be mapped to morphological operations, the core of this accelerator is a programmable morphology PE array. With the programmable ability of each PE and the interconnection between them, this accelerator can achieve both high throughput and high flexibility. Simulation shows the proposed hardware accelerator can speed up most important video segmentation algorithms 20 times to achieve real-time performance and has high programmability with a little hardware cost overhead.

1. INTRODUCTION

Video segmentation is the technique which can generate object shape information from video sequences. It is a key operation for MPEG-4 since without video segmentation, the content-based coding functionalities cannot be realized. It is also a very important operation for feature extraction of video sequences, which can be used for MPEG-7 to represent and index video data or for other intelligent signal processing applications, such as recognition.

Video segmentation can be used for many real-time applications for MPEG-4 such as video phone, video conference, and content-based video camcorder; however, for SIF format or other larger frame size, even with the fast algorithm [1] and a powerful microprocessor, it is still very hard to achieve real-time requirement (30fps). Therefore, hardware implementation of video segmentation is necessary. On the other hand, 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, a hardware accelerator which can accelerate different segmentation algorithms with an unified architecture is urgently needed, and a video segmentation system should have both hardware parts and software parts.

In this paper, a hardware accelerator for video segmentation is proposed. The core of this accelerator is a programmable morphology PE array, which is extended from our prior work [2]. For different algorithms, the PE array can be re-programmed to perform different operations. The target algorithms to be accelerated are the most important algorithms and all the other algorithms developed with similar core operations.

*Thanks to SiS Education Foundation for funding.

Table 1. Core operations of each video segmentation algorithm.

Algorithm	Core operations
Ours[1]	gradient, change detection, background registration, post-processing
Kim[3]	change detection, watershed
Mech[4]	change detection, relaxation, optical flow, edge detection, edge fitting
Meier[5]	morphological motion filter, optical flow, Canny edge detection, Hausdorff distance
Wang[6]	multiscale gradient, watershed, motion tacking and projection

This paper is organized as follows. Existing algorithms are analyzed in Sec. 2. After that, in Sec. 3, core operations of video segmentation are mapped to morphological operations. Sec. 4 and Sec. 5 show the proposed architecture and the simulation results, respectively. Finally, Sec. 6 gives a conclusion.

2. ANALYSIS OF EXISTING ALGORITHMS

Five algorithms are analyzed in this section. These algorithms can represent most of video segmentation algorithms. The core operations of them are shown in Table 1.

These operations can be classified into five types according to their properties: morphological operations, region growing operations, pixel operations, motion estimation related operations, and other operations, as shown in Table 2. Note that the watershed transform includes both morphological and region growing operations. In addition, MAX-Tree (MIN-Tree) generation and distance transform of morphological motion filter and Hausdorff distance are region growing operations.

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. Canny edge detection, in addition, can be replaced with an edge detector based on morphological gradient operation, and the 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

Table 2. Analysis of 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, backgrond registration
Motion estimation related	optical flow, motion tracking and projection
Other	relaxation, Canny edge detection

to accelerate video segmentation, the hardware implementation of different types of morphological operations is a good choice.

3. MAPPING CORE OPERATIONS TO MORPHOLOGICAL OPERATIONS

In this section, the core operations are mapped to morphological operations. 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. The multiscale gradient operation [6] can be described as follows:

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

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

The post-processing of our algorithm [1] includes small region elimination and close-open operations. The small region elimination can be replaced with dilation and conditional erosion (geodesic erosion) operations:

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

where $l > (n-1)/2$, and the conditional erosion is

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

Moreover, close-open are originally morphological operations.

Watershed transform can separate an image into many homogeneous non-overlapped close 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 and can be afforded by software; 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. Fig. 1(a) shows the gray level value of an image. Each pixel is then given a unique label from low gray-level to high gray-level in raster scan order, as shown in Fig. 1(b). For example, there are six pixels valued 0 in Fig. 1(a), and each pixel is given a label from 0 to 5, from top to bottom and from left to right in Fig. 1(b). Note that, based on the address sort concept, the computational load of this procedure is

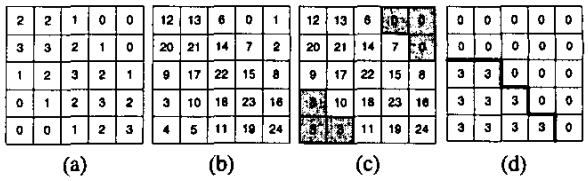

Fig. 1. Map flooding process of watershed transform to masked morphological erosion operation.

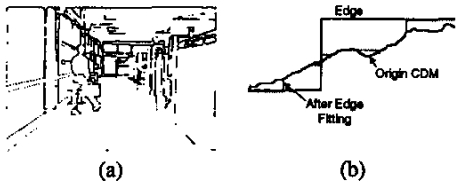

Fig. 2. Results of (a)morphological edge detection operation and (b)morphological edge fitting operation.

low. After that, masked erosion operations are applied, which only apply erosion operations on pixels with a specific gray-level value. After Fig. 1(b) is applied masked erosion operations at gray-level 0 until no change occurs, the result is shown as Fig. 1(c), where the operations are only applied on the pixels marked with gray color, whose gray-level are 0 in Fig. 1(a). If the same operations are applied gray-level by gray-level, the result is shown in Fig. 1(d), where the bold line shows the watershed. Therefore, the watershed transform can be mapped to masked erosion, and the result is the same as conventional watershed transform.

The MAX-Tree (MIN-Tree) generation of morphological motion filter and distance transform of Hausdorff distance are also region growing operations; therefore, they can be implemented with similar method as watershed transform.

Canny edge detector is optimized for the ability of edge detection and localization. Experiments show that the edge detector based on morphological gradient also has good ability of edge detection; however, the edge localization is poor. It can be further improved with erosion operations as the following equations:

$$Edge = Th(GRA) \ominus B - Th(GRA) \ominus B \ominus B, \tag{5}$$

where *TH(.)* is a threshold operation, and *GRA* is the gradient image. The performance of this edge detector is shown in Fig. 2(a), where a frame of sequence *Hall Monitor* is taken as an example. It is very similar to the results of Canny edge detector.

The edge fitting operation can be simply mapped to morphological operations as following equations:

$$\begin{aligned}&((((((CDM \oplus B_n)\underbrace{\ominus B_3; Edge)\ldots \ominus B_3; Edge)}_{l}\\ &\ominus B_n)\underbrace{\oplus B_3; \overline{Edge})\ldots \oplus B_3; \overline{Edge})}_{l} \oplus B_3 \ominus B_3,\end{aligned} \tag{6}$$

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

Finally, gradient, multiscale gradient, post-processing, watershed transform, morphological motion filter, Hausdorff distance,

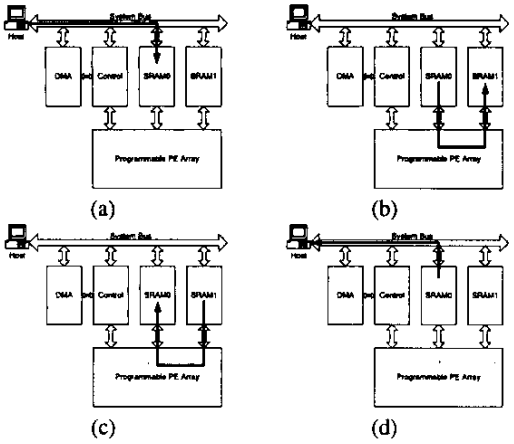

Fig. 3. Data flow of video segmentation hardware accelerator. (a)Load data; (b)process data from SRAM0 and store data to SRAM1; (c)process data from SRAM1 and store results to SRAM0; (d)transfer data back to the host computer.

edge detection, and edge fitting can be mapped to simple morphological operations: dilation, erosion, conditional erosion, conditional dilation, masked dilation, and masked erosion.

4. PROPOSED ARCHITECTURE

4.1. Overview of the proposed hardware accelerator

For the requirement of high throughput and flexibility, a hardware accelerator is proposed as shown in Fig. 3. A video segmentation system is divided into software parts and hardware parts, which are executed on the host computer and the hardware accelerator, respectively. The core of this accelerator is a programmble PE array, which can support various morphological operations and will be described in detail in next subsection.

A typical operation of the proposed accelerator can be divided into three steps: loading data, processing data, and writing back data. First, data is transferred from the host computer to a SRAM with DMA as shown in Fig. 3(a). Next, the data is processed with the PE array, and the results are stored in another SRAM, which is shown in Fig. 3(b). In Fig. 3(c), if the required operation is complex, the data may be processed again. After all the processes are finished, the results are transferred back to the host computer in Fig. 3(d). Note that the operation of the programmable PE array is controlled with instructions given from the control unit. Besides, the control unit can support zero-overhead loop, in-place operation, and self-control abilities, which can reduce the computational load of the host computer and the traffic load of the system bus.

4.2. Architecture of programmable PE array

The architecture of the programmable PE array is shown in Fig. 4. Pipelined architecture can increase processing speed with fixed input and output data rate. On the other hand, parallel architecture can increase processing speed with fixed internal memory size. In order to make good balance between internal memory size and input/output data rate, the proposed architecture is based on a pipelined-parallel array architecture.

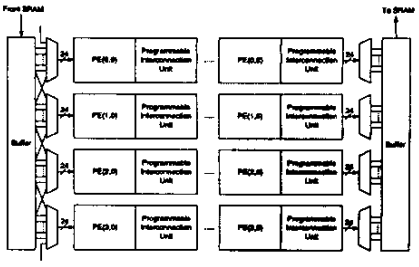

Fig. 4. Architecture of programmable PE array.

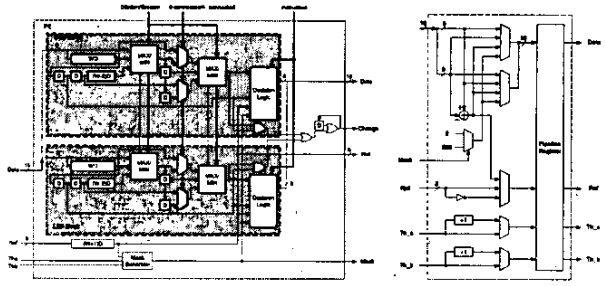

Fig. 5. (a)Architecture of a single PE, note that each PE has two sub-PEs. (b)Architecture of programmable interconnection unit.

If SIF format video sequences are considered, the architecture should be designed as Fig. 4, where four pipelined PE array operate in parallel. A frame is segmented into four overlapped tiles, and each pipelined PE array processes a tile. The boundary condition can be manipulated well because of the overlapped parts. The detail architecture of a single PE is shown in Fig. 5(a), which has sub word parallel ability. In each PE, it has two 8-bit sub-PEs. A sub-PE can perform 8-bit dilation, erosion, conditional dilation, conditional erosion, masked dilation, masked erosion, and no operation. Combining two sub-PEs, a PE can further perform 16-bit operations. Note that both the 3x3 structuring element (8-connected) and cross-shaped structuring element (4-connected) can be supported in this architecture. In Fig. 4, there are programmable interconnection units between PEs, that is, with some control signals, the interconnection between two PEs can be changed. That makes the proposed architecture more flexible. The detail architecture of a programmable interconnection unit is shown in Fig. 5(b). Note that the combination of a PE and a programmable interconnection unit is a MacroPE. In Fig. 4, there are 36 MacroPEs.

5. SIMULATION RESULTS

The result of hardware implementation is shown in Table 3, which is estimated with SYNOPSYS^{TM} Design Compiler. The operation frequency is 40MHz when the processing speed of 22140 16-bit morphological operations or 44280 8-bit morphological operations per second can be achieved. The gate count can be further reduce if the target frame size and the frame rate is reduced, or the operation frequency is increased. The internal memory size is only 20% of a frame memory. Note that the internal memory is used as delay lines in this architecture.

The estimated system performance is shown in Table 4. The host computer is a low-end computer with a Celeron 300MHz mi-

Table 3. Result of hardware implementation.

Unit	Gate count	Internal memory size
Single MacroPE		
PE	1189	2712b
Programmable interconnection unit	608	0b
Other	2	0b
Total(36 MacroPEs)	64764	97632b

Table 4. 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[1]	223.24	9.31	10.16
Watershed	453.24	23.82	28.47

croprocessor, and the system bus is a PCI bus. Assume no other peripheral shares the system bus with the accelerator. Our algorithm and the watershed transform are applied on SIF video sequences. It shows that without the hardware accelerator, the processing speed is far behind real-time requirement (33ms/frame). With the accelerator, the processing time of our algorithm is 10.16 ms, and the processing time of watershed transform is 28.47ms, which can achieve real-time requirement. Note that the processing time should be the maximum of the processing time of software part and hardware part since the host computer 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 real-time requirement with the proposed hardware accelerator.

Table 5 shows the comparison between proposed architecture and other morphology architectures. Morphological gradient operation is considered. It is shown that the proposed architecture has much more different configurations with very little hardware cost overhead. Note that the number of comparators is reduced with partial-result-reuse concept [2], and the memory size is not taken into consideration since the number of I/O pins of these architectures are quite different. It is unfair to compare the memory size at this situation.

Table 6 shows the comparison between the proposed architecture with other watershed architectures. Note that the simplification and gradient operations are not implemented in other two architectures. It is obvious that the proposed architecture is more feasible and more efficient.

6. CONCLUSION

A hardware accelerator for video segmentation is proposed in this paper. It is based on a programmable morphology PE array. Simulation shows this accelerator can accelerate most important video segmentation algorithms to achieve real-time requirement. Compared with existing architectures for mathematical morphology and watershed, the proposed architecture is efficient in hardware complexity and programmability.

7. REFERENCES

[1] S.-Y. Chien, S.-Y. Ma, and L.-G. Chen, "An efficient video segmentation algorithm for real-time MPEG-4 camera system," in *Proc. of Visual Communication and Image Processing 2000*, 2000, pp. 1087–1098.

Table 5. Comparison with other morphology architectures.

Architecture	Comparator count[a]	Register number	Estimated gate count[b]	Configuration
Diamantaras[7][c]	16	6	1168	1
Sheu[8][d]	10	16	1514	1
This work[e]	8	8	1189	175

[a] two-input comparator
[b] comparator:49gates, 8-bit register:64gates
[c] Parallel version, 1 PE
[d] Ignore the adders/subtractors
[e] Only one PE included

Table 6. Comparison with other watershed architecture.

Architecture	Software control loading	Gate count	Achieve real-time	Programmability
[9]	heavy	2965	hard	no
[10]	light	1792000	yes	no
This work	no	64764	yes	yes

[2] S.-Y. Chien, Y.-W. Huang, S.-Y. Ma, and L.-G. Chen, "A hybrid morphology processing units architecture for real-time video segmentation systems," in *Proc. of the 2001 IEEE International Symposium on Circuits and Systems*, 2001, vol. 5, pp. 275 –278.

[3] M. Kim, J. G. Choi, H. Lee D. Kim, M. H. Lee, C. Ahn, and Y.-S. Ho, "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*, vol. 9, no. 8, pp. 1216–1226, Dec. 1999.

[4] R. Mech and M. Wollborn, "A noise robust method for 2D shape estimation of moving objects in video sequences considering a moving camera," *Signal Processing*, vol. 66, 1998.

[5] T. Meier and K. N. Ngan, "Video segmentation for content-based coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 8, pp. 1190–1203, Dec. 1999.

[6] D. Wang, "Unsupervised video segmentation based on watersheds and temporal tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 5, pp. 539–546, Sept. 1998.

[7] K.I. Diamantaras and S.Y. Kung, "A linear systolic array for real-time morphological image processing," *Journal of VLSI Signal Processing*, vol. 17, 1997.

[8] M.-H. Sheu, J.-F. Wang, J.-S. Chen, A.-N. Suen, Y.-L. Jeang, and J.-Y. Lee, "A data-reuse architecture for gray-scale morphologic operations," *IEEE Transactions on Circuits and Systems–II Analog and Digital Signal Processing*, vol. 39, no. 10, pp. 753–756, Oct. 1992.

[9] C. J. Kuo, S. F. Odeh, and M. C. Huang, "Image segmentation with improved watershed algorithm and its FPGA implementation," in *Proc. of the 2001 IEEE International Symposium on Circuits and Systems*, 2001, vol. 2, pp. 753–756.

[10] D. Noguet, "A massively parallel implementation of watershed based on cellular automata," in *Proc. of the 1997 IEEE International Conference on Application-Specific Systems, Architecture and Processors*, 1997.