ALGORITHM AND ARCHITECTURE OF VIDEO SEGMENTATION HARDWARE SYSTEM WITH A PROGRAMMABLE PE ARRAY

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

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

ABSTRACT

Video segmentation is a key unit in content-based video encoding systems, such as MPEG-4. Existing algorithms are too complex for real-time applications, and hardware implementation is infeasible because of the global and irregular operations. In this paper, a hardware system for video segmentation is proposed from algorithm level to hardware architecture level. A hardware-oriented algorithm is first proposed to generate accurate object masks with local pixel operations and morphological operations, which are suitable for hardware implementation. After that, the hardware architecture is designed based on partial-result-reuse architecture and programmable morphology PE array architecture, which can achieve both high flexibility and throughput. A prototype chip is implemented to achieve the processing speed of 30 QCIF frames per second and 7,680 morphological operations per second at 26 MHz. It also shows the hardware cost is small, and the proposed video segmentation hardware system is suitable to be integrated into any content-based video encoding systems.

1. INTRODUCTION

MPEG-4 has been taken as the most important standard for multimedia applications [1]. The key functionality of MPEG-4 visual part is content-based interactivity, that is, the video encoding system is object-based rather than frame-based as MPEG-2. To support this, the object masks of video objects should be generated in advance. Video segmentation is the technique to generate object masks from the input video sequences, and the object mask information is required for shape coding, texture coding, motion estimation, and motion compensation in MPEG-4 encoding systems, as shown in Fig. 1. For real-time applications, a real-time segmentation system is urgently required because without video segmentation, most of the new functionalities of MPEG-4 cannot be realized.

Many video segmentation algorithms have been proposed [2]. They can be classified into two types: algorithms based on temporal information and algorithms based on spatio-temporal information. Algorithms of the first type use the motion information derived in temporal domain. Motion estimation, optical flow, or change detection [3] is often employed. Then the object can be segmented based on the assumption that the motion of the foreground objects should be inconcurrent to the global motion. The segmentation results are usually not precise enough, and complicated post-processing processes are needed to refine the object masks. The other type of algorithms is based on both motion information and spatial information. The spatial information can be generated by image segmentation algorithms, such as watershed transform [4, 5, 6], which can separate a frame into many homogeneous regions. After that, motion information are used to select foreground regions. These algorithms can give precise segmentation results; however, the computation load are too high to be employed in real-time applications. Moreover, they are hard to be implemented in hardware since many global operations are included in these algorithms, and the irregular operations make the associated hardware hard to be accelerated with parallel and pipeline design techniques.

In this paper, an algorithm and hardware architecture for video segmentation hardware system are proposed. The algorithm is designed for hardware implementation. It is modified from our prior successful algorithm [7, 8]. All the operations in this algorithm are local pixel operations or morphological operations [9], which are very regular and suitable for hardware implementation. The hardware architecture is designed based on the hardware-oriented algorithm. A programmable morphology PE array architecture is proposed here to give both high flexibility and throughput for video segmentation systems.

*Thanks to SiS Education Foundation for funding.

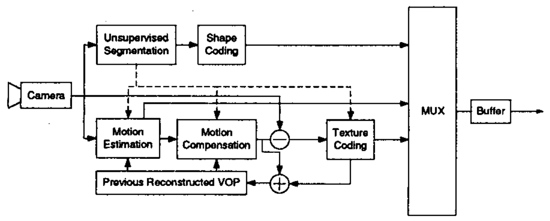

Fig. 1. Block diagram of a MPEG-4 content-based coding system with a segmentation unit.

The hardware-oriented video segmentation algorithm is first described in next section. Then the hardware architecture and simulation results are shown in Sec. 3 and Sec. 4, respectively. Finally, in Sec. 5, a short conclusion is given.

2. HARDWARE-ORIENTED VIDEO SEGMENTATION ALGORITHM

The block diagram of the proposed hardware-oriented video segmentation algorithm is shown in Fig. 2. It includes two modes: a baseline mode is designed for general situations, and a shadow cancellation mode is designed for video sequences influenced by shadow and light change.

Figure 2(a) shows the block diagram of baseline mode. It has five main parts: frame difference, background registration, background difference, object detection, and post-processing. First, the frame difference of current frame and previous frame are thresholded to form *Frame Difference Mask (FDM)*. The mask is then used in background registration to register background information into background buffer. The reliable background consists of pixels that are not parts of moving objects for consecutive *fth* frames. Next, the frame difference of current frame and background frame is also thresholded to form *Background Difference Mask (BDM)*. In object detection, if background exists, *BDM* is chosen as *initial object mask (OMi)*; otherwise, *FDM* is chosen. Finally, post-processing can refine *OMi* to *object mask (OM)*.

There are two parts in post-processing: noise region elimination and morphological close-open operation. The noise region elimination can filter out small black regions (holes) and small white regions (noise) of *OMi*. It is implemented with connected component operation in [8]. After that, the morphological close-open operation is applied to smooth the boundary of the object masks.

In shadow cancellation mode, which is shown in Fig. 2(b), morphological gradient filter is first applied to depress the influence of light change and shadow. An extra erosion operation is added in post-processing to eliminate the edge-thickening effect of gradient filter.

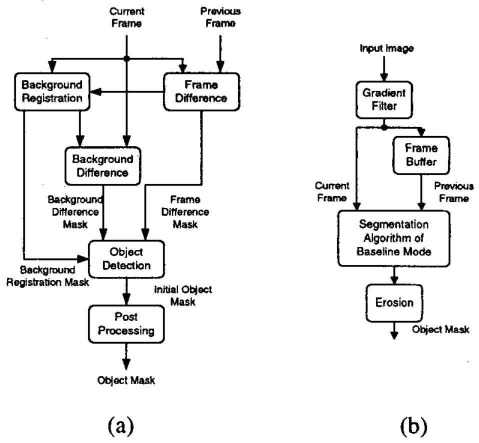

Fig. 2. Block diagram of the hardware-oriented video segmentation algorithm. (a) Baseline mode; (b) shadow cancellation mode.

Among all the operations of this algorithm, frame difference, background registration, background difference, and object detection are pixel operations, namely, each pixel is processed independently. They are regular and suitable for hardware implementation. The gradient filter, close-open operation, and erosion operation are morphological operations, which can also easily be mapped to hardware architecture. On the other hand, the connected component operation, which is a global operation, needs to be modified. Instead of connected component operation, the noise region elimination can be implemented with morphological dilation, conditional erosion, and opening operations, which can be shown in the following equation:

$$(\dots((I\oplus B_m)\underbrace{\ominus B_3;I)\dots\ominus B_3;I)}_{l}\circ B_3, \tag{1}$$

where $l > (m-1)/2$, $\oplus$ is dilation, which can be described by the following equation:

$$I\oplus B(x,y) = \max\{I(x-i,y-i)|(x,y)\in I,(i,j)\in B\}, \tag{2}$$

$\ominus$ is erosion, which can be described by the following equation:

$$I\ominus B(x,y) = \min\{I(x+i,y+i)|(x,y)\in I,(i,j)\in B\}, \tag{3}$$

B_n is an nxn structuring element, $\circ$ is opening, which can be described by the following equation:

$$I\circ B = (I\ominus B)\oplus B, \tag{4}$$

and the conditional erosion (geodesic erosion) is:

$$X\ominus B;Y = (X\ominus B)\cup Y. \tag{5}$$

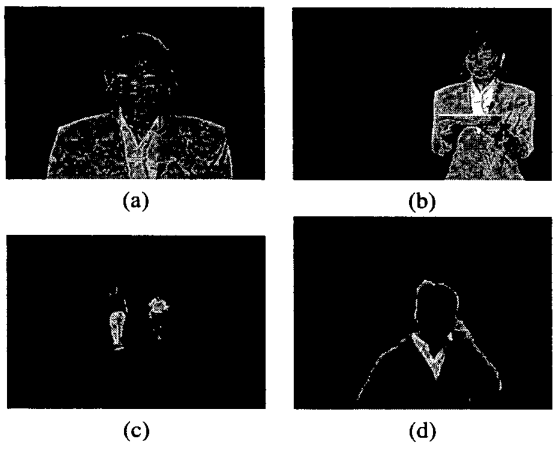

Fig. 3. Segmentation results of the proposed algorithm. (a) Akiyo #50; (b) Weather #50; (c) Hall Monitor #50;(d) Frank #50.

Although the modified noise region elimination operation may be more complicated for software implementation, it is local operation and has more regular dataflow; therefore, it is more suitable to be implemented in hardware.

Some segmentation results are shown in Fig. 3. Many standard sequences and other sequences captured in our lab with a general camera are tested. It is shown that the proposed algorithm performs well for various situations and can successfully segment out the object masks.

No global operations and complicated branch operations are included in this algorithm. All operations are either pixel based local operations or morphological operations, which can be mapped to efficient hardware architecture. Therefore, the proposed video segmentation algorithm is very suitable for hardware implementation.

3. HARDWARE ARCHITECTURE

The hardware architecture designed for the hardware-oriented algorithm is shown in Fig. 4. It has two parts: gray-scale part and binary part. The data width in gray-scale part is 8-bit, and the data width in binary part is 1-bit.

In gray-scale part, there are three units: control unit, gradient filter (*GRA*), and change detection mask and background generator (*CDMBG*). *GRA* applies morphological gradient filter on current frame data to eliminate shadow effect and light change effect. *CDMBG* can generate both *initial object mask (OMi)* and background information. The background information (*B*), previous frame (*P*), background indicator (*BI*), and stationary index (*SI*) are stored in an off-chip memory. Note that the background indicator can indicate where the background information exists, and the stationary index records the probability a pixel belongs to background. *OMi*, whose data width is 1-bit, is then sent to binary part.

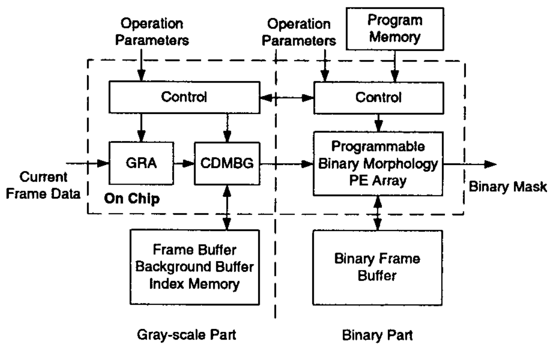

Fig. 4. Architecture of the video segmentation chip.

The main unit in binary part is a programmable binary morphology PE array. The PE array can be programmed by instructions stored in an off-chip program memory to perform different kinds of binary morphological operations in post-processing of our algorithm, such as noise region elimination and smooth operation. For complex operations, a folding technique [10] is applied, and an off-chip memory is used to store the partial results.

The detailed architectures of *GRA*, *CDMBG*, and *Programmable PE Array* are described in the following three subsections. Furthermore, the scheduling of this hardware architecture and how to combine several chips to deal with video sequences with large frame size are also shown.

3.1. Gradient Filter

The morphological gradient filter can be described by the following equation:

$$G = (I \oplus B) - (I \ominus B). \qquad (6)$$

The hardware architecture is shown in Fig. 5. In Fig. 5, w is the width of the frame, and *MAXMIN* can output the maximum and the minimum of its two input values. The gradient filter is implemented with a hardware-cost efficient architecture named partial-result-reuse architecture, in which the partial results during operations are kept and reused and is proved to be superior to other morphology architectures [11]. It needs only seven 8-bit comparators in the gradient filter. Note that, for size and power consideration, the delay lines are implemented with two on-chip two-port SRAMs rather than registers.

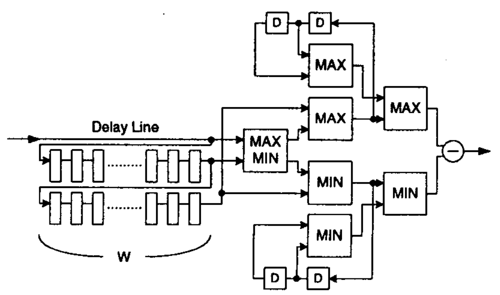

Fig. 5. Architecture of morphological gradient filter using partial-result-reuse architecture.

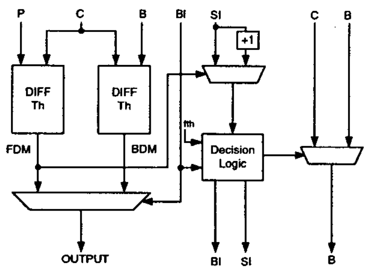

Fig. 6. Architecture of CDMBG (change detection mask and background generator).

3.2. CDMBG

Frame difference, background registration, background difference, and object detection in Fig. 2 are processed in *CDMBG* module. All these operations are pixel local operations, that is, the operation is applied pixel by pixel independently; therefore, it can be easily mapped to hardware architecture with high throughput. The hardware architecture of *CDMBG* is shown in Fig. 6, where *DIFF Th* is the frame difference and thresholding unit, and *Decision Logic* decides if the current input pixel is a part of reliable background. If it is, it will be written into background; otherwise, the background will be unchanged.

3.3. Programmable PE Array

The post-processing, which includes noise region elimination and close-open operation, is the most computationally intensive part in our algorithm. The post-processing includes three basic binary morphological operations: dilation, erosion, and conditional erosion.

To achieve real-time requirement, the throughput should be high, and a PE array architecture is suitable; however, the operations in post-processing are different for different situations, and a flexible architecture is required. To achieve both high throughput and flexibility, a programmable binary morphology PE array architecture is proposed as shown in Fig. 7.

In Fig. 7, each PE can be programmed to perform one of the four operations: 3x3 dilation, 3x3 erosion, 3x3 conditional erosion, and no operation. For larger structuring elements, the chain rule of morphological operations is applied as shown below:

$$A \oplus (B \oplus C) = (A \oplus B) \oplus C. \tag{7}$$

It means cascading two 3x3 dilation is equivalent to a 5x5 dilation. To achieve real-time requirement, eight PEs are required at 26MHz, that is, it has $4^8 = 65,536$ different configurations. For example, if the following operation is mapped into the programmable PE array:

$$I \oplus B_3 \oplus B_3 \oplus B_3 \ominus B_3 \ominus B_3, \tag{8}$$

the first three PEs are programmed as dilation, PE3 and PE4 are programmed as erosion, and PE5–PE7 are programmed as no operation. If a more complex operation is mapped into the array, a folding technique is applied to decompose the operation into many simple operations, and each simple operation is mapped into the PE array in different time slot. An off-chip memory is used to store the partial results of the PE array. For example, if the following morphological operation is considered:

$$I \oplus B_3 \oplus B_3 \oplus B_3 \oplus B_3 \oplus B_3 \oplus B_3 \oplus B_3 \oplus B_3 \ominus B_3 \ominus B_3, \tag{9}$$

it is equivalent to:

$$(I \oplus B_3 \oplus B_3 \oplus B_3 \oplus B_3 \oplus B_3 \oplus B_3 \oplus B_3 \oplus B_3) \ominus B_3 \ominus B_3. \tag{10}$$

To map this operation into the programmable PE array, the eight PEs are all first programmed as dilation, and the output data are stored into the off-chip binary frame buffer. After the whole frame is processed, the first two PEs are programmed as erosion, and the other PEs are programmed as no operation. The binary data are then sent into the PE array from the binary frame buffer, and the final results can be obtained.

The detailed architecture of a single PE is shown in Fig. 8. The max operation is replaced with OR operation in the binary morphology PE, and the hardware cost is also minimized with partial-result-reuse concept. The hardware cost of each PE can be further reduced by means of the duality property:

$$(A \ominus B)^c = A^c \oplus \check{B}. \tag{11}$$

Similar to *GRA*, the delay lines of each PE are implemented with an on-chip SRAM. The vertical boundary control and horizontal boundary control signal is given from the control unit to deal with the boundary condition.

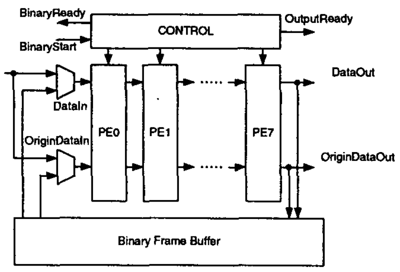

Fig. 7. Programmable binary morphology PE array.

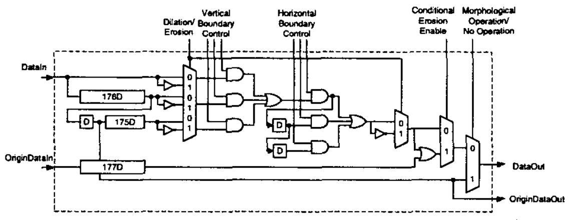
Vertical Boundary Control
Dilation/ Erosion
Horizontal Boundary Control
Conditional Erosion Enable
Morphological Operation/ No Operation
DataIn
OriginDataIn
176D
175D
177D
D
DataOut
OriginDataOut

Fig. 8. Detailed architecture of a single PE.

3.4. Scheduling

The scheduling of the whole system is shown in Fig. 9. For each frame, the frame data are first sent to the *GRA*. The output of the *GRA* is then sent to the *CDMBG*, the output of the *CDMBG* is sent to the PE array, The output of the PE array is stored in the binary frame buffer. During this period, the pipeline of the whole system is full. After all the frame data are processed, the *GRA* and the *CDMBG* are stopped. The control unit of the binary part reprograms the PE array with instructions loaded from the off-chip program memory, and then the PE array read data from the binary frame buffer and store the results into the binary frame buffer. After the program stored in the program memory is executed, the PE array sends out the object mask information, and the frame data of the next frame are sent to the *GRA*. The same procedure is repeated frame by frame.

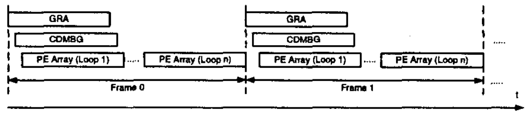

Fig. 9. Scheduling of the proposed segmentation chip.

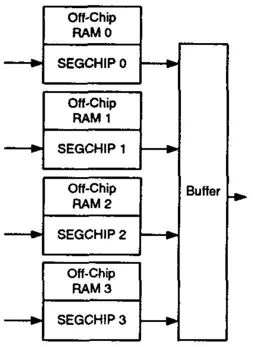

Fig. 10. Scalable architecture with multiple segmentation chips working in parallel.

3.5. Scalable architecture

After fabricated, the maximum frame size to be handled by a single chip is fixed; however, combining several chips can deal with video sequences with larger frame size. This scalable architecture with multiple segmentation chips is shown in Fig. 10. As shown in the figure, a frame is first segmented into four overlapped tiles, and each tile is sent to a video segmentation chip. Four chips work in parallel with four independent off-chip memory groups, and the output data are combined in a buffer. The results are the same as those of the original algorithm if the overlapped regions are large enough.

4. SIMULATION RESULTS

The segmentation chip is designed in cell-based design flow with Avant! standard cell library and Avant! RAM compiler. Two memory built-in-self-test modules are also integrated into the chip to test the 8-bit SRAM and 24-bit SRAM separately.

The video segmentation chip is currently under fabrication by TSMC. The chip layout of the proposed system is shown in Fig. 11. There are three on-chip two-port SRAM on the chip. Two of them are 176x8, which are used as delay lines in *GRA*. The other one, which is 176x24, is used as delay lines of the programmable PE array. The technology is TSMC 0.35μm 1P4M. The chip size is 2.77x2.77 mm^2 where the transistor count is 143,122. It shows the chip size and transistor count of this system is quite small and can be easily integrated into a single chip MPEG-4 encoding system. The detailed features of the chip is shown in Table 1.

Simulation results show that this chip can achieve real-time requirement for QCIF (176x144) video at 26MHz, and the programmable PE array can perform 7,680 binary morphological operations per second, which is sufficient for

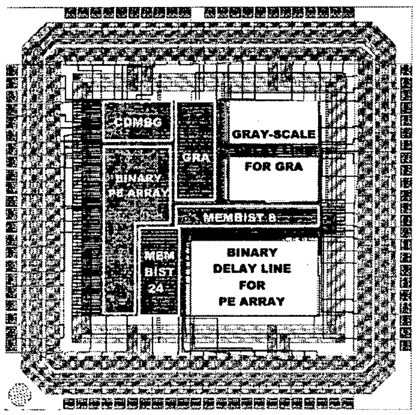

Fig. 11. Chip layout of the proposed system.

Table 1. Features of the Video Segmentation Chip.

Technology	TSMC 0.35μm 1P4M
Package	100 CQFP (91 Pads)
Transistor count	143,122
Chip size	2.77mm$\times$2.77mm
Power supply	3.3V
Power consumption	48.35mW@33MHz
Processing speed	30 QCIF frames/s
(at 26 MHz)	7,680 morphological operations/s
On-chip memory	2 176$\times$8 two-port RAM
	1 176$\times$24 two-port RAM

most situations. Note that although the target frame size of this prototyping chip is only QCIF, it is easy to scale the design into CIF format or even larger video format.

5. CONCLUSION

A hardware-oriented video segmentation algorithm and its associated hardware architecture is proposed in this paper. The algorithm is modified from our prior video segmentation algorithm. No global and irregular operations are included in this algorithm, which make it suitable for hardware implementation. The hardware architecture has two parts: gray-scale part and binary part. The gray-scale part is designed with partial-result-reuse design technique to have low hardware cost. The core unit of binary part is a programmable PE array. It can achieve high throughput as well as flexibility. A prototype chip is currently under fabrication with 0.35μm 1P4M technology by TSMC. It shows the chip size is small and can be easily integrated into any content-based coding systems.

6. REFERENCES

[1] *The MPEG-4 Video Standard Verification Model version 18.0*, ISO/IEC JTC 1/SC 29/WG11 N3908, 2001.

[2] *Annex F: preprocessing and postprocessing*, ISO/IEC JTC 1/SC 29/WG11 N4350, 2001.

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

[4] M. Kim, J. G. Choi, D. Kim, H. Lee, 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, December 1999.

[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, December 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, September 1998.

[7] 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*. SPIE, June 2000, pp. 1087–1098.

[8] S.-Y. Chien, S.-Y. Ma, and L.-G. Chen, "Efficient moving object segmentation algorithm using background registration technique," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 7, pp. 577–586, July 2002.

[9] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, London, 1982.

[10] K. K. Parhi and T. Nishitani, *Digital Signal Processing for Multimedia Systems*, Marcel Dekker, 1999.

[11] S.-Y. Chien, S.-Y. Ma, and L.-G. Chen, "A partial-result-reuse architec-ture and its design technique for morphological operations," in *Proc. of International Conference on Acoustics, Speech, and Signal Processing 2001*. IEEE, May 2001.