Journal of VLSI Signal Processing 42, 241–255, 2006

DOI: 10.1007/s11265-006-4185-1

Hybrid Morphology Processing Unit Architecture for Moving Object Segmentation Systems

SHAO-YI CHIEN
DSP/IC Design Lab, Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan

BING-YU HSIEH AND YU-WEN HUANG
Media Tek, Inc., Hsinchu 300, Taiwan

SHYH-YIH MA
Vivotek, Inc., Taipei 235, Taiwan

LIANG-GEE CHEN
DSP/IC Design Lab, Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan

Received April 21, 2004; Revised April 21, 2004; Accepted March 18, 2005

Published online: 13 February 2006

Abstract. Video segmentation is a key operation in MPEG-4 content-based coding systems. For real-time applications, hardware implementation of video segmentation is inevitable. In this paper, we propose a hybrid morphology processing unit architecture for real-time moving object segmentation systems, where a prior effective moving object segmentation algorithm is implemented. The algorithm is first mapped to pixel-based operations and morphological operations, which makes the hardware implementation feasible. Then the high computation load, which is more than 4.2 GOPS, can be overcome with a dedicated morphology engine and a programmable morphology PE array. In addition, the hardware cost, memory size, and memory bandwidth can be reduced with the partial-result-reuse concept. This chip is designed with TSMC 0.35 μm 1P4M technology, and can achieve the processing speed of 30 QCIF frames or 7,680 morphological operations per second at 26 MHz. Simulation shows that the proposed hardware architecture is efficient in both hardware complexity and memory organization. It can be integrated into any content-based video processing and encoding systems.

Keywords: video segmentation, moving object segmentation, hardware architecture, morphological operations

1. Introduction

MPEG-4 video standard [1] is one of the most important video coding standards for multimedia systems. It will be applied to many real-time applications, such as video phone, video conference, interactive program, and intelligent video surveillance systems. The key functionality of MPEG-4 is content-based coding, in which a video scene is divided into many video objects (VO), and these objects are then encoded separately. Video segmentation is the methodology for indicating where the VOs are and generating their

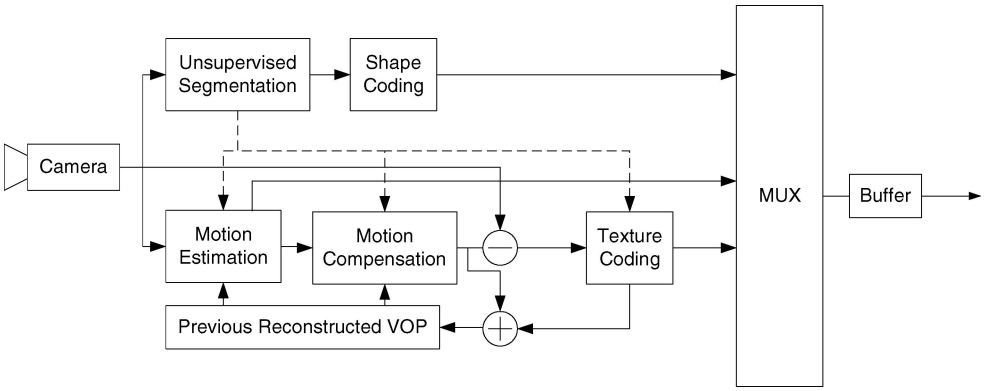

Figure 1. Block diagram of a MPEG-4 content-based coding system with a segmentation unit.

shape information. The shape information will be used for shape coding, texture coding, motion estimation, and motion compensation, as shown in Fig. 1 [2], where Video Object Plane (VOP) is a projection of VO on the camera plane. It is quite important since without video segmentation, MPEG-4 coding systems will lose most of their functionalities. In addition, since shape is a significant feature of a VO, video segmentation is also a key technique for MPEG-7 [3] for representing and indexing video content.

Many video segmentation algorithms have been proposed [4, 5]. They can be classified into two categories: temporal-spatial segmentation and spatial-temporal segmentation. The temporal-spatial segmentation algorithms first find the foreground objects with temporal information derived from optical flow, motion estimation, or change detection. Then the object mask can be further refined by spatial information, such as edge locations [6]. In view of its low computation load, change detection is the most popular temporal segmentation algorithm. However, it suffers from still objects and uncovered background problems, and change detection can be easily influenced by shadow and light changes. Therefore, these algorithms are usually not robust enough and need powerful post-processing algorithms, which may introduce complex computation. On the other hand, the spatial-temporal segmentation algorithms first analyze the spatial information of a single frame and then find where the objects are with temporal information. For example, Meier's algorithm [7, 8] takes edge locations as spatial information and then track the edge with morphological motion filter and Hausdorff transform. The most popular spatial segmentation algorithm is watershed transform [9], which can segment a frame into many homogeneous regions. It can be further combined with region tracking [10] or change detection [11] to generate the object mask. These algorithms can give precise video segmentation results; however, the computation load is too high to be employed in real-time applications, and they are too complex for hardware implementation. The efficient algorithm we have proposed employs change detection with background registration technique and shadow cancellation algorithm [12, 13], which can avoid still objects and uncovered background problems and give good segmentation results. It can also resist the influence of shadow and light changes. Although the limitation of this algorithm is that it can only separate moving objects from still background, it is proved that it is quite effective for segmentation in video conference, video surveillance, and other applications with still cameras. In addition, the computational complexity of this algorithm is lower than that of other algorithms. The proposed algorithm can be further optimized with bit-parallel and MMX techniques. It can process 25 QCIF frames per second with a Pentium-III 450 MHz processor when shadow cancellation mode is turned off [14]. However, if the frame size becomes large or the shadow cancellation mode is turned on, the required computation power will become too high to be afforded by general processors. Therefore, for real-time applications, hardware implementation of video segmentation is urgently needed.

Mathematical morphology [15, 16], which is based on set theory, is widely used for video segmentation algorithms, including the proposed algorithm. The morphological operations are regular and very suitable for hardware implementation. For that reason, the hardware implementation of video segmentation algorithms fully mapped to morphological operations are much easier. Many hardware architectures for morphological operations have been proposed [17–19], but

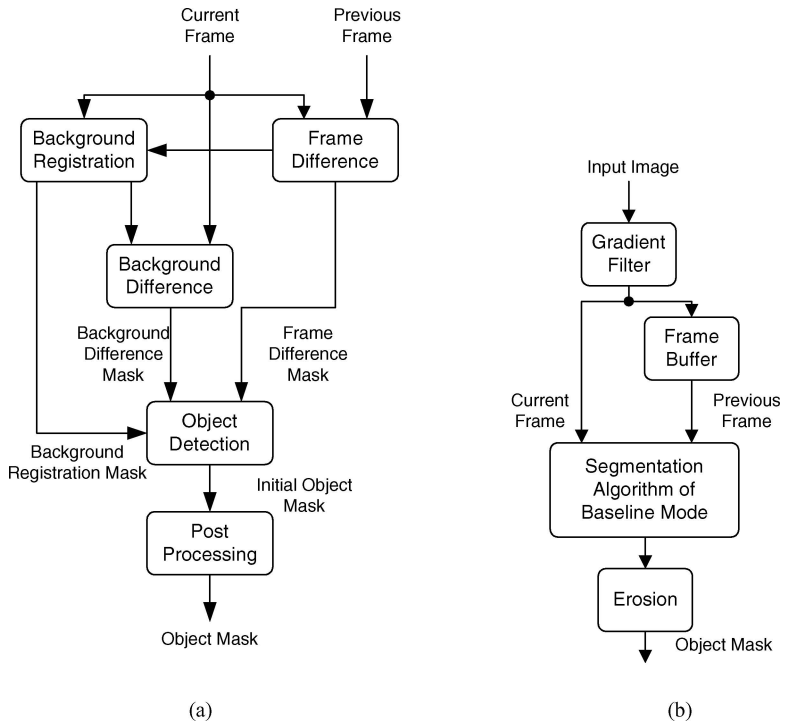

Figure 2. Block diagram of the proposed moving object segmentation algorithm. (a) Baseline mode; (b) shadow cancellation mode.

the hardware cost is large for complex morphological operations and should be further optimized. Moreover, these hardware architectures are not flexible enough to execute the segmentation algorithms with adaptive properties.

In this paper, a hybrid morphology processing units architecture for moving object segmentation systems is proposed. This architecture employs an effective moving object segmentation algorithm, which is our previous work. The high computation load and adaptability of the algorithm can be provided with a dedicated morphology engine and a programmable morphology PE array. In addition, the hardware cost, memory size, and memory bandwidth can be further reduced with the Partial-Result-Reuse architecture [20].

In Section 2, the moving object segmentation algorithm is first introduced. The proposed architecture is presented in Section 3. Section 4 describes the architectural analysis of this architecture, and Section 5 shows the VLSI implementation and simulation results. Finally, Section 6 gives the conclusion.

2. Moving Object Segmentation Algorithm

We have developed an effective moving object segmentation algorithm for still cameras [12, 13]. The algorithm has two modes: a baseline mode designed for general situations and a shadow cancellation mode designed for video sequences influenced by shadow and light changes. The block diagram of this algorithm is shown in Fig. 2.

In baseline mode, as shown in Fig. 2(a), the algorithm has five steps: frame difference, background registration, background difference, object detection, and post-processing. First, the frame difference of current frame and previous frame is thresholded to form *Frame Difference Mask (FDM)*. The mask is then sent to background registration unit to register background information into background buffer. The reliable background consists of pixels that are not in *FDM* for consecutive *FTH* frames, where *FTH* is the threshold value to separate the pixels belonging to the background. Next, the frame difference of current frame and background frame is also thresholded to form *Background Difference Mask (BDM)*. In the object detection unit, if background exists, *BDM* is chosen as the 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 boundary smoothing. Noise region elimination can filter out small black regions (holes) and small white regions (noise) of *OMi*. To

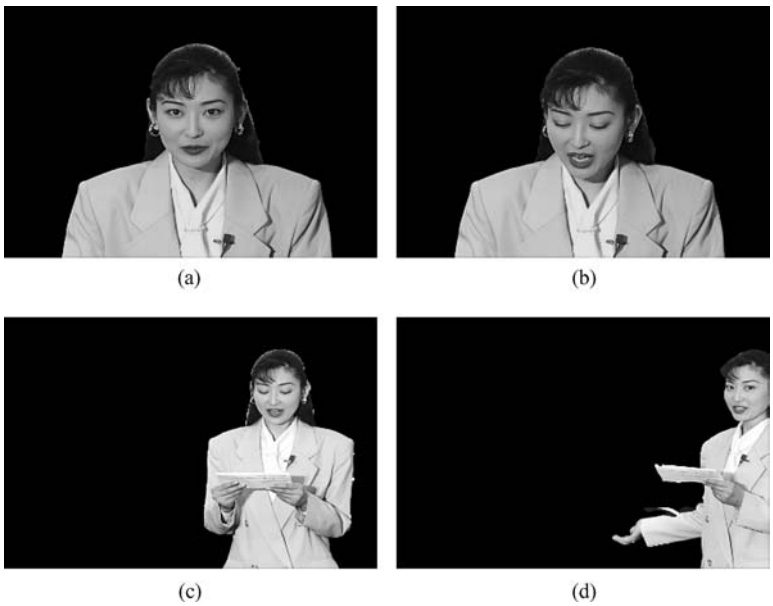

Figure 3. Segmentation results of the proposed algorithm for sequence (a) *Akiyo* #50; (b) *Akiyo* #100; (c) *Weather* #50; (d) *Weather* #100. The baseline mode is applied.

do this, the connected component operation [15] is employed to give each connected region a special label and record its area, and then regions with small area will be taken as noise and be eliminated. Finally, morphological closing and opening operation [16] is used to smooth the mask, and the *Object Mask (OM)* is generated.

Some segmentation results of baseline mode are shown in Fig. 3. In Fig. 3(a) and (b), the segmentation results of sequence *Akiyo* are shown, where the VO has only slight motion. On the other hand, the segmentation results of sequence *Weather* are also shown in Fig. 3(c) and (d). The VO in *Weather* has large motion. These results show that the proposed algorithm can deal with both situations and segment out the foreground object precisely.

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

The results of the shadow cancellation mode are shown in Fig. 4. Sequences *Hall Monitor*, *Frank*, and *Shaoyi* are used as examples. All sequences are influenced by both shadow and light changes. Figure 4(a) and (b) show the shadow of VOs in the sequence *Hall Monitor* can be eliminated by the shadow cancellation mode. It also shows that the proposed segmentation algorithm can deal with more than one VOs. Sequences *Frank* and *Shaoyi* are captured in our lab with a general camera. The segmentation results are also satisfied as shown in Fig. 4(c)–(f), implying that the proposed algorithm can also be applied with a general camera. For more segmentation results, please refer to [13].

Run-time analysis with bit-parallel optimization is shown in Fig. 5. The frame size is CIF (352×288) and the shadow cancellation mode is adopted. Figure 5 shows that the original algorithm needs 436 ms with a 450 MHz Pentium-III processor to process a frame, and the optimized one needs 263 ms where the real-time requirement is 33 ms per frame. Moreover, a simple morphological operation, such as dilation, needs computing power of about 700 MOPS to process 30 CIF frames per second, and six operations are needed in our algorithm for gradient filter and boundary smoothing, meaning that huge computing power of about 4.2 GOPS is required. Consequently, hardware implementation for video segmentation is inevitable.

3. Hardware Architecture

In Fig. 2(a), frame difference, background registration, background difference, and object detection are all pixel-based operations, that is, each pixel is manipulated independently. The boundary smoothing opera-

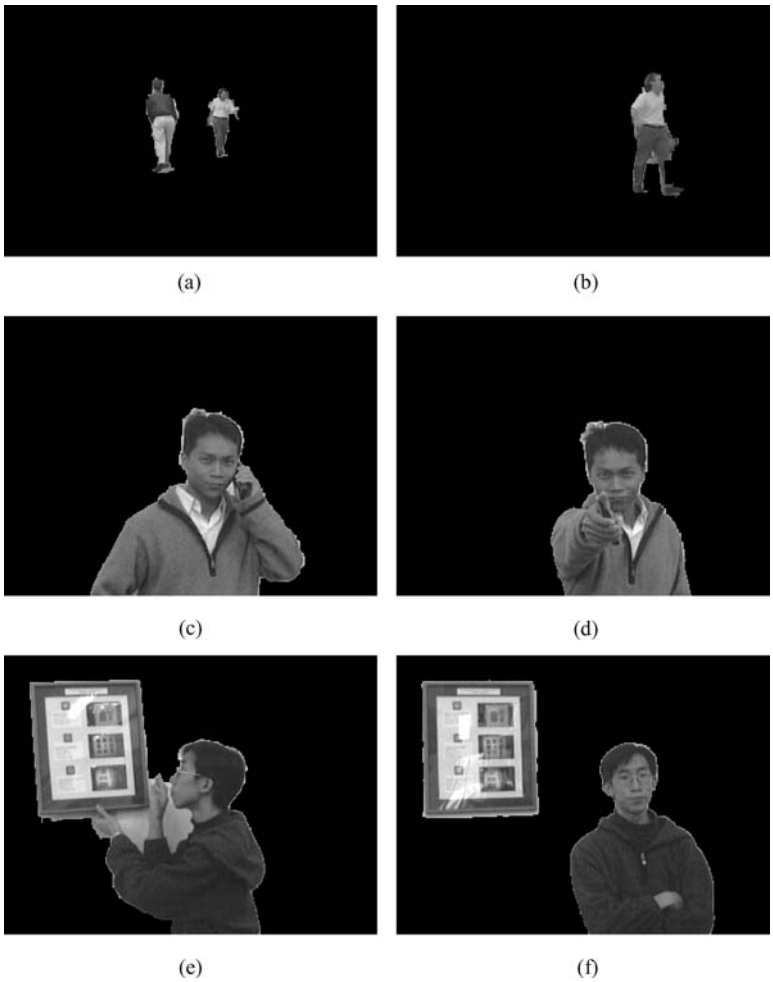

Figure 4. Segmentation results of the proposed algorithm for sequence (a) *Hall Monitor* #50; (b) *Hall Monitor* #100; (c) *Frank* #50; (d) *Frank* #100; (e) *Shaoyi* #50; (f) *Shaoyi* #100. The shadow cancellation mode is applied.

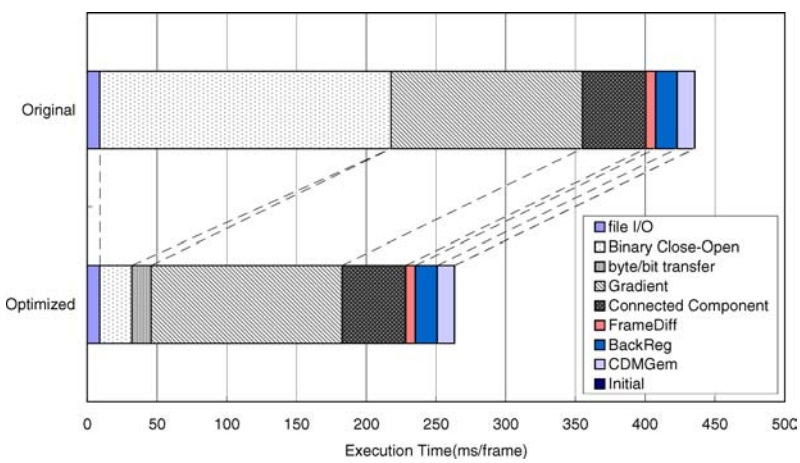

Figure 5. Runtime analysis of the proposed moving object segmentation algorithm.

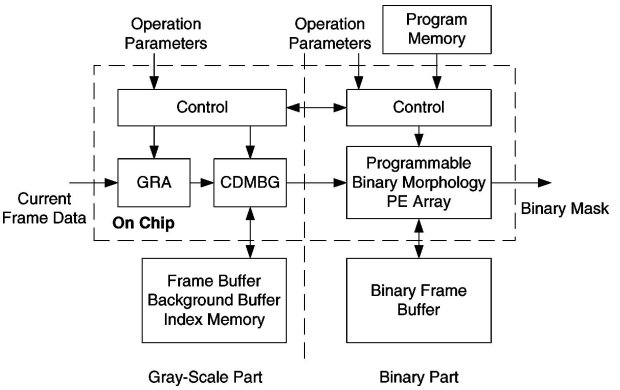

Figure 6. Block diagram of the proposed moving object segmentation hardware system.

tion in Fig. 2(a) and gradient filter in Fig. 2(b) are morphological operations, which are local neighborhood operations. Therefore, this algorithm is very suitable for hardware implementation because of its regularity and localization.

Both gray-scale and binary morphological operations are needed in our algorithm. They are dramatically different: the word-length of gray-scale morphological operations is 8-bit so byte processing units are needed; on the other hand, binary morphological operations need only bit-level operations and can be implemented with simple logic gates. Therefore, it is more efficient to separate the system into a gray-scale part and a binary part. Furthermore, the gray-scale part in the proposed algorithm is fixed for different video sequences, which is suitable to be implemented with dedicated logic; on the other hand, the parameters of the binary part in the proposed algorithm are adaptive for different video sequences, which is more suitable to be implemented with programmable logic. Therefore, the proposed hardware architecture is hybrid with a gray-scale morphology processing unit, a binary morphology processing unit, dedicated logic, and programmable logic.

The block diagram of this system is shown in Fig. 6. The gray-scale part contains four units. *GRA* is the morphological gradient filter, which corresponds to the gradient filter in Fig. 2(b). *CDMBG* is the change detection mask and background generator. All pixel-based operations, including frame difference, background registration, background difference, and object detection, are executed in this unit. *Frame Buffer*, *Background Buffer*, and *Index Memory* are used to store previous frame, background information, and the situation of each pixel, respectively. A control unit is also included in the gray-scale part. The binary part also includes four units: *Programmable Binary Morphology PE Array* takes charge of all kinds of binary morphological operations in post-processing in Fig. 2(a); *Control* is the control unit of the PE array; *Binary Frame Buffer* is used to store partial results of the PE array when dealing with complicated binary morphological operations; and *Program Memory* is used to store the program to be executed in the PE array. The detail of this system is presented in the following two subsections. Note that *Frame Buffer*, *Background Buffer*, *Index Memory*, *Binary Frame Buffer*,and *Program Memory* are off-chip memories. The other units of this system are all integrated in a single chip, as shown in Fig. 6.

3.1. Gray-Scale Part

There are two processing units in the gray-scale part: *GRA* and *CDMBG*. The operation of the morphological gradient filter *GRA* can be shown as follows.

Let I be the image, B be the structuring element (SE), and G be the output of gradient filter.

$$G = (I \oplus B) - (I \ominus B), \tag{1}$$

where

$$\begin{aligned} &I \oplus B(x, y) \\ &\quad = \max\{I(x - i, y - j) \,|\, (x, y) \in I, (i, j) \in B\}, \\ &I \ominus B(x, y) \\ &\quad = \min\{I(x + i, y + j) \,|\, (x, y) \in I, (i, j) \in B\}, \end{aligned}$$

where $\oplus$ and $\ominus$ are dilation and erosion, respectively. If the SE is 3×3 as shown in Fig. 7(a), the dilation result of point (r, c) is the maximum of the nine points in Fig. 7(b), which needs eight comparators to get

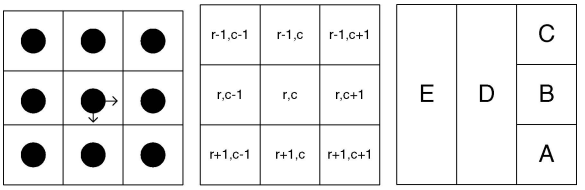

Figure 7. An example of Partial-Result-Reuse.

the result in one cycle. Note that the illustration of SE follows Haralick and Shapiro's notation [15]. We found that if the partial results during computation are kept and reused, the required comparators can be further reduced. The concept is derived below.

If the nine pixels are divided into five groups as shown in Fig. 7(c). Let

$$\begin{aligned} A_{i,j} &= I(i+1, j+1), \\ B_{i,j} &= I(i, j+1), \\ C_{i,j} &= I(i-1, j+1), \\ D_{i,j} &= \max\{I(i-1, j), I(i, j), I(i+1, j)\}, \\ E_{i,j} &= \max\{I(i-1, j-1), \\ &\quad \times I(i, j-1), I(i+1, j-1)\}, \\ F_{i,j} &= \max\{A_{i,j}, B_{i,j}, C_{i,j}\}. \end{aligned}$$

Then

$$\begin{aligned} I \oplus B(r, c) &= \max\{A_{r,c}, B_{r,c}, C_{r,c}, D_{r,c}, E_{r,c}\} \\ &= \max\{A_{r,c}, B_{r,c}, C_{r,c}, F_{r,c-1}, D_{r,c-1}\} \end{aligned} \tag{2}$$

Equation (2) implies that only $I(r-1, c+1)$, $I(r, c+1)$, and $I(r+1, c+1)$ are required for the computation, and the value of $D_{r,c}$ and $E_{r,c}$ can be got by reusing from the former partial results $F_{r,c-1}$ and $D_{r,c-1}$, respectively. In addition, if the input signal is in raster scan manner, extra delay lines are needed. The Partial-Result-Reuse architecture for morphological gradient filter is shown in Fig. 8. Two serial delay lines, whose length is equal to the frame width W, are required, and they can be implemented with registers or memories. On the right side of Fig. 8, the upper part is the dilation unit, and the lower part is the erosion unit. The nodes corresponding to A, B, C, D, E, and F in (2) are marked in the dilation unit of the figure. Two registers are needed to store the partial results for

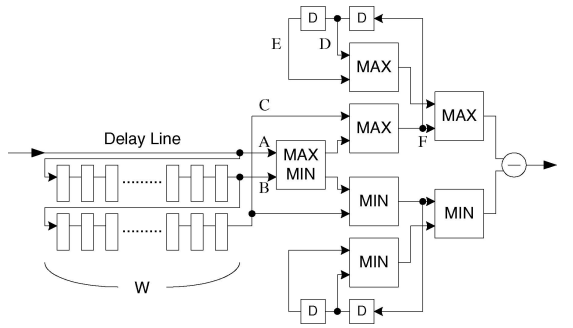

Figure 8. Architecture of morphology gradient filter.

each unit. *MAX* outputs the maximum of its two inputs, *MIN* outputs the minimum, and *MAXMIN* outputs the maximum and minimum simultaneously with only one comparator. Note that this architecture may introduce some errors at the boundaries of the image; however, it will not influence the final results since these errors will be eliminated in the binary part.

The *CDMBG* can be implemented directly as shown in Fig. 9. *P* is the previous frame, *C* is the current frame, *B* is the registered background, and *DIFF Th* is the frame difference and thresholding unit. *FDM* and *BDM* can be generated with two such units. *SI* is stationary index, which records how many consecutive frames each pixel is not in change detection mask up to now. If the current pixel is not in *FDM*, the corresponding *SI* will be increased by one; otherwise, *SI* will not change. *BI* is the background indicator, which indicates if the background exists. If it does, *BDM* is chosen as the output; otherwise, *FDM* is chosen. *Decision Logic* decides if the current input pixel is a part of the reliable background. If it is, it will be written

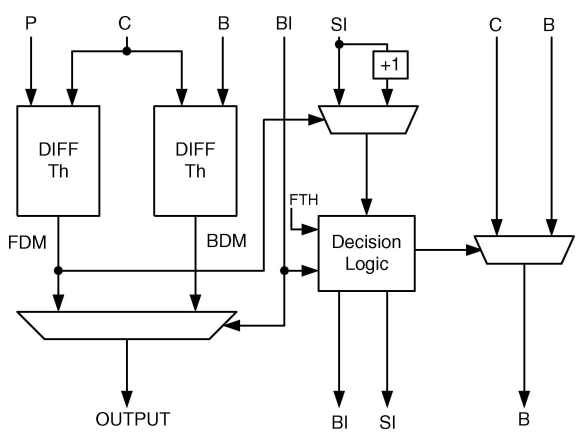

Figure 9. Change detection mask and background generation unit.

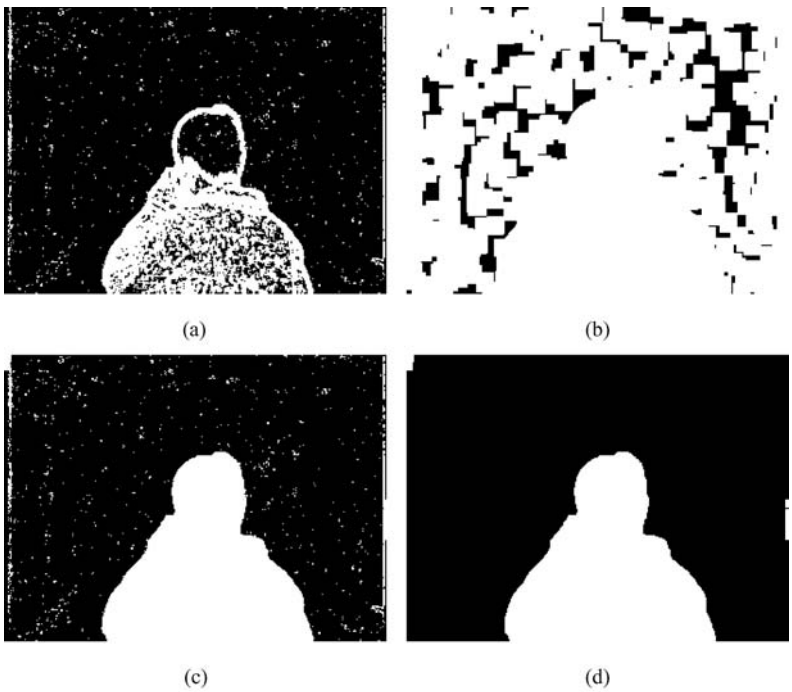

Figure 10. Noise region elimination implemented with morphological operations. (a) Initial object mask *OMi*; (b) after dilation; (c) after conditional erosion; (d) the final result.

into the background, or the background will be unchanged. *C* is then stored in *Frame Buffer*, *B* is stored in *Background Buffer*, and *SI* and *BI* are stored in *Index Memory*.

3.2. Binary Part

The output of *CDMBG* is the initial object mask, which is binary data and can be processed with binary processing units. The binary part of the segmentation system has two functions: noise region elimination and boundary smoothing. In the proposed algorithm, boundary smoothing is a set of morphological operations, and noise region elimination is based on connected component operation [15], which is not suitable for hardware implementation. Therefore, it should be mapped to morphological operations first.

We found that the white color noise (salt noise) in *OMi* is usually small in size, as shown in Fig. 10(a), and can be simply eliminated with morphological opening operation. However, the black color noise (hole) is usually too large in size to be eliminated with closing operation without degrading the precision of object boundaries. Conditional morphological operations [16] can preserve the shape information, and the combination of dilation and conditional erosion (geodesic erosion) can be used for black noise region elimination, which is a kind of reconstruction filter. The procedure can be shown as the following equation:

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

where B_m is an mxm structuring element, and B_3 is a 3×3 structuring element. The conditional erosion is

$$X \ominus B; Y = (X \ominus B) \cup Y. \qquad (4)$$

Note that the binary dilation and erosion are simply gray-scale dilation and erosion with *MAX* and *MIN* replaced by *OR* and *AND*, respectively. The frame size of the example sequence *Frank* is 320×240. If we choose $m = 15$ and $l = 50$, the result of dilation is shown in Fig. 10(b). After conditional erosion, the result is shown in Fig. 10(c). Finally, after opening operation, the result of (3) is shown in Fig. 10(d). It is very similar to that of connected component and region filtering. Note that m is proportional to object size and l is related to the shape of object and the frame width.

The boundary smoothing procedure contains morphological closing operation and opening operation. The closing operation is:

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

and the opening operation can be shown as:

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

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. For example, the selection of parameters m and l depends on the properties of input sequence. Hence, a flexible architecture is required. In order to achieve both high throughput and flexibility, a programmable PE array architecture is proposed, which is illustrated in Fig. 11.

In Fig. 11, each PE can perform one of the four operations: 3×3 dilation, 3×3 erosion, 3×3 conditional erosion, and no operation. For large size SE, the chain rule is applied as shown below:

$$A \oplus B_5 = A \oplus (B_3 \oplus B_3) = (A \oplus B_3) \oplus B_3. \tag{7}$$

It means cascading two 3×3 dilation is equivalent to one 5×5 dilation. Consequently, if the PE array contains n PEs, it has 4^n different configurations. For example, if the following operation is mapped into the programmable PE array:

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

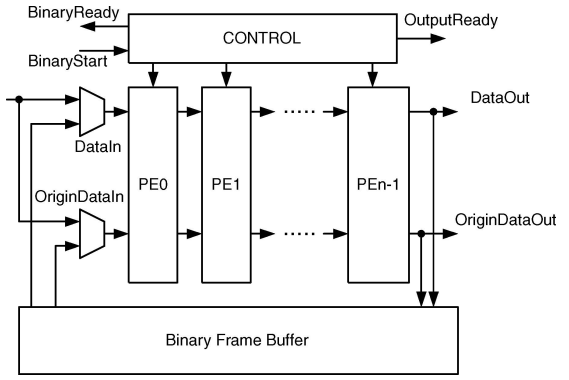

Figure 11. Programmable PE array architecture for binary morphological operations.

The first three PEs are programmed as dilation, and the next two PEs are programmed as erosion, and the others are programmed as no operation. If a more complex operation is mapped into the array, the fixed length of the array may not be enough. A folding technique is employed 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 $n = 8$, and 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 the following equation:

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

To map this operation into the programmable PE array, the eight PEs are all 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 illustrated in Fig. 12. It is also optimized with the partial-result-reuse concept. The hardware cost of each PE can be further reduced by means of the duality property [15]:

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

where $\check{B}$ is the transpose of B, and for the SE in Fig. 7(a), $\check{B} = B$.

The PE has two data input pins and two data output pins. *DataIn* and *DataOut* are binary input data and output data, respectively. *OriginDataIn* and *OriginDataOut* are the initial object mask generated from *CDMBG*, which is required if the PE is programmed to execute conditional erosion operation. *Dilation/Erosion (D/E)*, *Conditional Erosion Enable*, and *Morphological Operation/No Operation (OP/NOP)* are control signals to program the PE. The detailed instructions and associated control signals are shown in Fig. 13, where "-" denotes "don't care." Unlike those at the gradient filter, the errors at the boundaries will be propagated in this array. Hence, extra circuits are needed to avoid errors in boundary conditions. *Vertical Boundary Control* signals, including *VC0*, *VC1*,

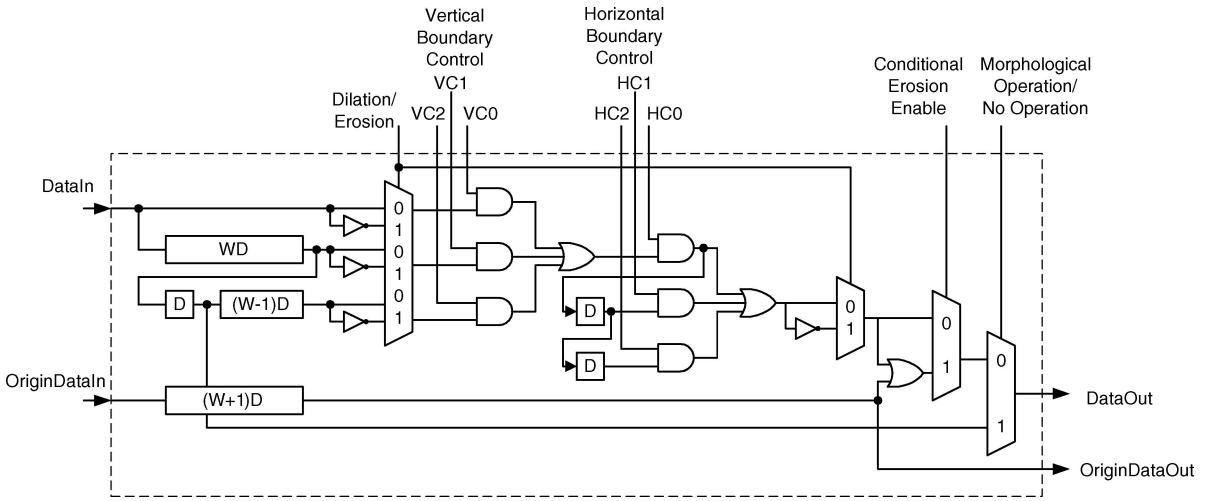

Figure 12. Detailed architecture of a single PE.

and *VC2*, and *Horizontal Boundary Control* signals, including *HC0*, *HC1*, and *HC2*, are boundary condition control signals. They can mask out the data outside the frame boundary to avoid error occurrence.

The architecture of the control unit of the binary morphology PE array is illustrated in Fig. 14. The control unit generates control signals and boundary control signals for each PE. The instructions for the PEs are loaded from the off-chip program memory to *Instruction Registers*. *Instruction Decoder* decodes the instructions for each PE to control signals as Fig. 13. Two counters, *x-counter* and *y-counter*, are used to record the coordinate of the input pixel data of PE0, and the values of the counters are propagated PE by PE. The two subtractors with saturation circuits are used to adjust the pixel coordinate of each PE. *VC Table* and *HC Table* can generate control signals for boundary conditions from the coordinate of the pixel being processed. Note that the architecture of the PE and the boundary condition control circuits are quite regular; hence, it can be easily pipelined as shown in Fig. 14. The detailed control signal for boundary conditions are shown in Fig. 15, where W is the frame width, H is the frame height, R denotes "row," which is the value of *y-counter*, and C denotes "column," which is the value of *x-counter*. Note that the x value and the y value of the first column and the first row are both "1" not "0," and redundant row data need to be sent to the array to deal with the boundary conditions. Moreover, *HC Table* refers to both x and y values. The control signals *HC0*, *HC1*, and *HC2* for pixel $(r, c) = (1, 1)$ and pixel $(r, c) = (2, 1)$ are different although they are in the same column, as shown in Fig. 15(b). A finite state machine *FSM* can finally control the PE array by controlling the two counters and the instruction registers. *BinaryReady* and *BinaryStart* are used to contact with the gray-scale part. If the process in the PE array is finished, *BinaryReady* signal is asserted to tell the gray-scale part to send another frame. *BinaryStart* signal is then asserted if the frame data are being sent. *OutputReady* is used to indicate when the output of the PE array is valid. Note that the control signals for *Binary Frame Buffer* are not illustrated in the figure for simplification.

Instruction	D/E	Conditional	OP/NOP
Dilation	0	0	0
Erosion	1	0	0
Conditional Erosion	1	1	0
NOP	-	-	1

Figure 13. Instructions and control signals of each PE.

3.3. *Scheduling Diagram of the System*

The scheduling of the whole system is shown in Fig. 16. For each frame, the frame data are first sent to the *GRA*. The output of the PE array is stored in the binary frame buffer. During this period, the pipeline of the whole system is full, and the hardware utilization can achieve 100%. After all the frame data are processed, the *GRA* and the *CDMBG* are stopped, and the PE array starts to read data from the binary frame buffer. The pipeline of the PE array is still full, and the hardware utilization of the PE array is still 100% while the *GRA* and *CDMBG* are idle. After the program stored in the

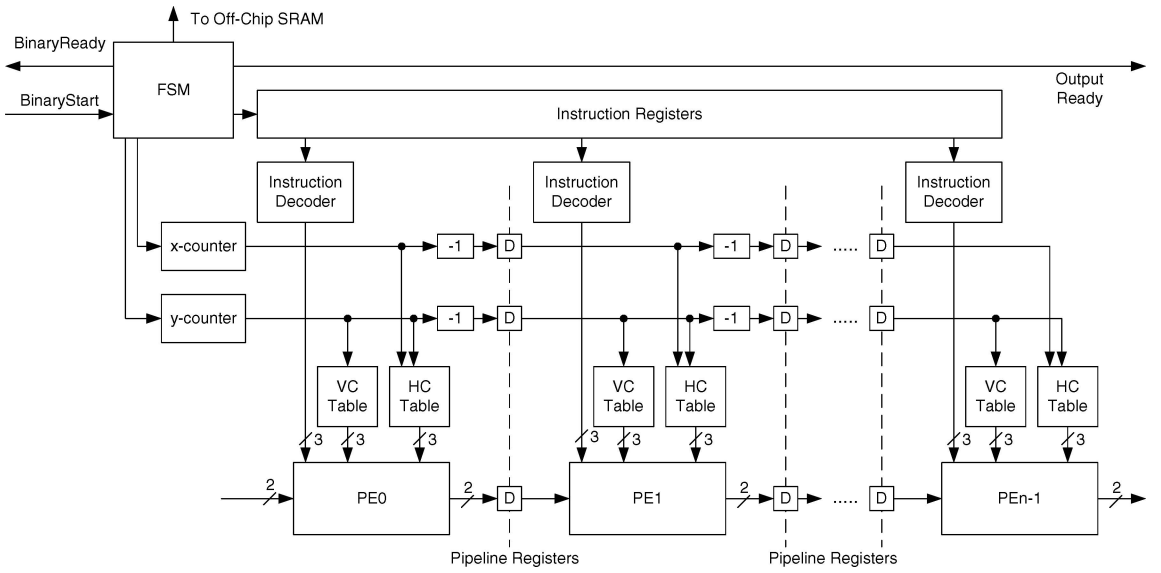

Figure 14. Pipelined architecture of the control unit of binary morphology PE array.

Value of y Counter	PE0			PE1			PE2			
	VC2	VC1	VC0	VC2	VC1	VC0	VC2	VC1	VC0	
R=1	0	0	1	0	0	0	0	0	0	
R=2	0	1	1	0	0	1	0	0	0	
R=3	1	1	1	0	1	1	0	0	1	
R=4	1	1	1	1	1	1	0	1	1	
⋮	⋮			⋮			⋮			
R=H	1	1	1	1	1	1	1	1	1	
R=(H+1)	1	1	0	1	1	1	1	1	1	...
R=1	0	0	1	1	1	0	1	1	1	
R=2	0	1	1	0	0	1	1	1	0	
⋮	⋮			⋮			⋮			

(a)

Value of x,y Counter	PE0			PE1			PE2			
	HC2	HC1	VC0	HC2	HC1	HC0	HC2	HC1	HC0	
C=1, R=1	0	0	1	0	0	0	0	0	0	
C=2, R=1	0	1	1	0	0	1	0	0	0	
C=3, R=1	1	1	1	0	1	1	0	0	1	
C=4, R=1	1	1	1	1	1	1	0	1	1	
⋮	⋮			⋮			⋮			
C=W, R=1	1	1	1	1	1	1	1	1	1	
C=1, R=2	1	1	0	1	1	1	1	1	1	...
C=2, R=2	0	1	1	1	1	0	1	1	1	
⋮	⋮			⋮			⋮			

(b)

Figure 15. Control signal for boundary conditions. (a) Vertical boundary control signals (*VC Table*); (b) horizontal boundary control signals (*HC Table*).

program memory is executed, the frame data of the next frame are sent to the *GRA*. The same procedure is repeated frame by frame. Although some pipeline bubbles are generated when data of a frame start to be sent into this system, the hardware utilization of the PE array is almost 100%. The pipeline bubbles can be avoided with more complex control circuits but are not implemented here for hardware cost consideration. On the other hand, the average utilization of the *GRA* and the *CDMBG* depends on the length of the program for the binary PE array. The longer the program, the lower the hardware utilization is.

4. Architectural Analysis

In this section, the proposed architecture is compared with other morphology architectures. Note that the architecture is verified with Verilog HDL, and the hardware cost is estimated with a logic synthesizer.

A Partial-Result-Reuse architecture is used to implement the gray-scale morphological gradient operation. The hardware cost of this architecture and other two architectures [17, 18] for gray-scale gradient filter is shown in Table 1. The proposed architecture has only half hardware cost of [17] and only about one third of [18]. Note that, for all the three architectures, the boundary conditions are not considered, and the memory cost is not included in this table. Since the input data pattern of the other two architectures [17, 18] are not set in raster scan manner, an extra large input buffer is needed in practical cases, and the required memory size should be larger than the proposed one.

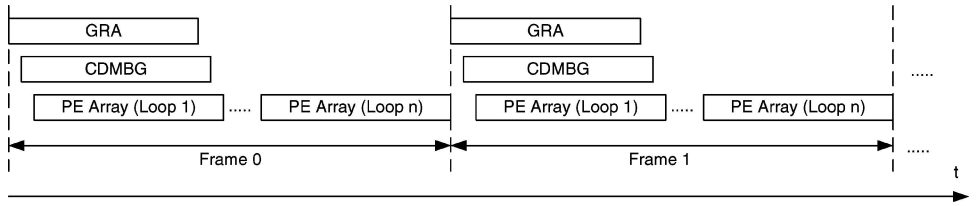

Figure 16. Scheduling of the proposed segmentation chip.

For CIF format video with 30 fps, the proposed architecture needs only 5.6 Kb on-chip memory, which is the minimum requirement when the data are input in raster scan manner, and it is far below the frame buffer size, 811 Kb. If the filter is implemented with an RISC, the amount of memory access is 438 Mb per second where it is only 24 Mb per second in this architecture. Therefore, the proposed architecture is also memory-cost-efficient.

In binary part, we take 7×7 dilation as an example, where only 3 PEs are required in the PE array.

Table 1. Comparison between the proposed gray-scale morphology architecture and other architectures.

Architecture	Comparator count[a]	Register number	Estimated gate count[b]
K.I. Diamantaras [17][c]	16	6	1,168
M.H. Sheu [18][d]	10	16	1,514
This work			
(Partial-Result-Reuse architecture)	7	4	599

[a]Two-input comparator.
[b]Comparator: 49 gates, 8-bits register: 64 gates.
[c]Parallel version, 1 PE
[d]Ignore the adders/subtractors.

Table 2. Comparison between the proposed binary morphology architecture and other architectures when 7×7 dilation is considered.

Architecture	Gate count[a]	Required cycles per frame[b]	Number of different configurations
K.I. Diamantaras[17]	384	101,376	1
E.N. Malamas[19]	5,075[c]	50,688	4
This work			
(Programmable PE array with 3 PEs)	1,541	101,376	64

[a]Ignore memory.
[b]Assume the pipeline is full.
[c]Ignore output networks of OR logic.

In Table 2, compared with the systolic array architecture [17], which is modified with replacing comparators with logic gates for fair comparison, and Erosion-Dilation Architecuture (EDA) [19], the proposed architecture is much more flexible although its hardware cost is higher than that of [17]. The proposed architecture has 64 different configurations; the EDA [19] has four different configurations: dilation, erosion, opening, and closing; and the systolic array architecture [17] has only one configuration because it cannot be programmed. It shows that the proposed hardware can achieve high flexibility without large hardware cost overhead. Note that the EDA can perform two basic morphological operations at the same time so only half of cycles per frame are needed; however, the hardware cost is enormous. Similarly, the memory parts are not included in this table. The memory requirement is the minimum in our architecture if the data are input in raster scan manner with one channel. In addition, the boundary conditions are considered in the proposed architecture, which are not in the other two architectures. More complex circuits are required for them to deal with this problem.

In this example, if target clock rate is 30 MHz, the cascaded 3 PEs can apply 7×7 dilation on about 300 CIF frames in one second. The gate count is 1,541, and 3 Kb on-chip memory is in demand, which is much lower than the frame buffer size, 101 Kb. Moreover, the off-chip memory access is 30 Mb per second. If an RISC is used in this speed, the amount of memory access will be 273 Mb per second, which shows that the proposed architecture has good memory organization.

5. Hardware Implementation

In order to show the feasibility of the proposed hardware architecture, a prototyping chip is implemented for QCIF video. For size and power consideration, the delay lines of GRA and the programmable PE array are implemented with two-port SRAMs. A delay line can be implemented with a two-port SRAMs combined with a register and an address generator (counter), as

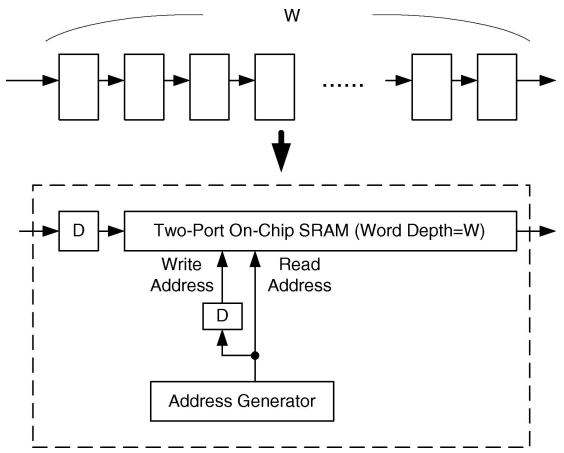

Figure 17. Use two-port SRAM to implement delay line.

shown in Fig. 17. It requires a memory whose word depth is W to implement a delay line with W registers. Note that the address generator can be shared in delay lines with the same length.

If the target frame size is QCIF and the target clock frequency is around 30 MHz, 8 PEs are needed to achieve real-time requirement. The gate count and memory size is shown in Table 3. The segmentation chip is designed with TSMC 0.35 μm 1P4M technology. The die photo of the proposed system is shown in Fig. 18. There are three on-chip two-ports SRAMs. Two of them are 176 $\times$ 8, which are taken as delay lines in GRA. The other one, which is 176 $\times$ 24, is taken as delay lines of the programmable PE array. The chip size is 2.77 $\times$ 2.77 mm^2 where the transistor count is 143,122, including memories and two memory built-in-self-test modules for the 8-bit SRAM and 24-bit SRAM, respectively. It shows that 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 moving object segmentation chip are shown in Table 4.

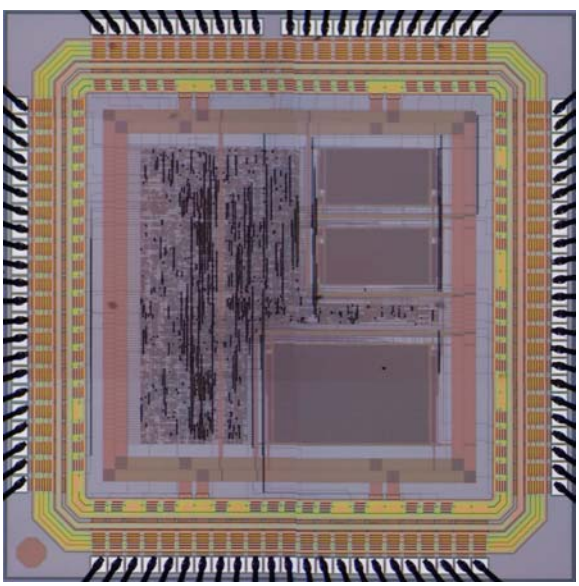

Figure 18. Die photo of the prototyping chip.

Table 3. Result of hardware implementation.

Unit	Gate count	On-chip memory size
Gray-scale part	3,408	2,816 b
Binary part (8 PEs)	3,652	4,224 b
Total	7,060	7,040 b

Table 4. Features of the prototyping chip.

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

Simulation results show that this chip can achieve real-time requirement for QCIF (176 $\times$ 144) video at 26 MHz, and the programmable PE array can perform 7,680 binary morphological operations per second, which is sufficient for 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.

6. Conclusion

A hybrid morphology processing unit architecture for real-time moving object segmentation systems is proposed in this paper. It is a gray-scale morphology, binary morphology, dedicated logic, and programmable logic hybrid architecture. Compared with the existing architectures for mathematical morphology, the proposed architecture is very efficient in both hardware

cost and memory organization. VLSI implementation of the proposed architecture shows that the chip size is small and can be easily integrated into MPEG-4 encoding systems or other content-based coding systems with still cameras.

There are still some limitations in this moving object segmentation system. First, this chip can only be employed for still camera systems. It is the limitation of the moving object segmentation algorithm used here. The other limitation is the restricted programmability of this chip. It is very hard to support other algorithms with the proposed hardware architecture. Therefore, although the proposed system is effective for the applications with still cameras, for general situations, it is still a challenged research topic to design the hardware system for video segmentation, which also belongs to our future work.

Acknowledgment

This work is supported in part by National Council, R.O.C., under Grant NSC93-2752-E-002-008-PAE and NSC90-2213-E-002-095. The chip fabrication is supported by National Chip Implementation Center (CIC).

References

1. T. Sikora, "The MPEG-4 Video Standard Verification Model," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 1, 1997, pp. 19–31.
2. MPEG Video Group, *The MPEG-4 Video Standard Verification Model version 18.0*, ISO/IEC JTC 1/SC 29/WG11 N3908, 2001.
3. S.-F. Chang, T. Sikora, and A. Puri, "Overview of the MPEG-7 Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 6, 2001, pp. 688–695.
4. MPEG Video Group, *Annex F: Preprocessing and Postprocessing*, ISO/IEC JTC 1/SC 29/WG11 N4350, 2001.
5. D. Zhang and G. Lu, "Segmentation of Moving Objects in Image Sequence: A Review," *Circuits Systems Signal Processing*, vol. 20, no. 3, 2001, pp. 143–183.
6. 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.
7. T. Meier and K.N. Ngan, "Automatic Segmentation of Moving Objects for Video Object Plane Generation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 5, 1998, pp. 525–538.
8. 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, 1999. pp. 1190–1203.
9. L. Vincent and P. Soille, "Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, 1991, pp. 583–598.
10. D. Wang, "Unsupervised Video Segmentation Based on Watersheds and Temporal Tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 5, 1998, pp. 539–546.
11. 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, 1999, pp. 1216–1226.
12. 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.
13. 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, 2002, pp. 577–586.
14. S.-Y. Chien, Y.-W. Huang, S.-Y. Ma, and L.-G. Chen, "Efficient Video Segmentation on SIMD Architecture for Real-time MPEG-4 Systems," in *Proc. of Workshop on Consumer Electronics 2000*, 2000.
15. R.M. Haralick and L.G. Shapiro, *Computer and Robot Vision*, Addison Wesley, 1992.
16. J. Serra, *Image Analysis and Mathematical Morphology*, London: Academic Press, 1982.
17. 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.
18. 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, 1992, pp. 753–756.
19. E.N. Malamas, A.G. Malamos, and T.A. Varvarigou, "Fast Implementation of Binary Morphological Operations on Hardware-efficient Systolic Architectures," *Journal of VLSI Signal Processing*, vol. 25, 2000.
20. S.-Y. Chien, S.-Y. Ma, and L.-G. Chen, "Partial-Result-Reuse Architecture and its Design Technique for Morphological Operations with Flat Structuring Elements," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 9, 2005, pp. 1156–1169.

Shao-Yi Chien was born in Taipei, Taiwan, R.O.C., in 1977. He received the B.S. and Ph.D. degrees from the Department of Electrical Engineering, National Taiwan University (NTU), Taipei, in 1999 and 2003, respectively.

During 2003 to 2004, he was a research staff in Quanta Research Institute, Tao Yuan Shien, Taiwan. In 2004, he joined the Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, as an Assistant Professor. His research interests include video segmentation algorithm, intelligent video coding technology, image processing, computer graphics, and associated VLSI architectures. shoayi@video.ee.ntu.edu.tw

Bing-Yu Hsieh was born in Taichung, Taiwan, in 1979. He received the B.S.E.E and M.S.E.E degrees from National Taiwan University (NTU), Taipei, in 2001 and 2003, respectively. He joined MediaTek, Inc., Hsinchu, Taiwan, in 2003, where he develops integrated circuits related to multimedia systems and optical storage devices. His research interests include object tracking, video coding, baseband signal processing, and VLSI design. bingyu@video.ee.ntu.edu.tw

Yu-Wen Huang was born in Kaohsiung, Taiwan, in 1978. He received the B.S. degree in electrical engineering and Ph. D. degree in the Graduate Institute of Electronics Engineering from National Taiwan University (NTU), Taipei, in 2000 and 2004, respectively. He joined MediaTek, Inc., Hsinchu, Taiwan, in 2004, where he develops integrated circuits related to video coding systems. His research interests include video segmentation, moving object detection and tracking, intelligent video coding technology, motion estimation, face detection and recognition, H.264/AVC video coding, and associated VLSI architectures. yuwen@video.ee.ntu.edu.tw

Shyh-Yih Ma received the B.S.E.E, M.S.E.E, and Ph.D. degrees from National Taiwan University in 1992, 1994, and 2001, respectively. He joined Vivotek, Inc., Taipei County, in 2000, where he developed multimedia communication systems on DSPs. His research interests include video processing algorithm design, algorithm optimization for DSP architecture, and embedded system design. syma@video.ee.ntu.edu.tw

Liang-Gee Chen was born in Yun-Lin, Taiwan, in 1956. He received the BS, MS, and Ph.D degrees in Electrical Engineering from National Cheng Kung University, in 1979, 1981, and 1986, respectively. He was an Instructor (1981–1986), and an Associate Professor (1986–1988) in the the Department of Electrical Engineering, National Cheng Kung University. In the military service during 1987 and 1988, he was an Associate Professor in the Institute of Resource Management, Defense Management College. From 1988, he joined the Department of Electrical Engineering, National Taiwan University. During 1993 to 1994 he was Visiting Consultant of DSP Research Department, AT&T Bell Lab, Murray Hill. At 1997, he was the visiting scholar of the Department of Electrical Engineering, University, of Washington, Seattle. Currently, he is Professor of National Taiwan University. From 2004, he is also the Executive Vice President and the General Director of Electronics Research and Service Organization (ERSO) in the Industrial Technology Research Institute (ITRI). His current research interests are DSP architecture design, video processor design, and video coding system.

Dr. Chen is a Fellow of IEEE. He is also a member of the honor society Phi Tan Phi. He was the general chairman of the 7th VLSI Design CAD Symposium. He is also the general chairman of the 1999 IEEE Workshop on Signal Processing Systems: Design and Implementation. He serves as Associate Editor of IEEE Trans. on Circuits and Systems for Video Technology from June 1996 until now and the Associate Editor of IEEE Trans. on VLSI Systems from January 1999 until now. He was the Associate Editor of the Journal of Circuits, Systems, and Signal Processing from 1999 until now. He served as the Guest Editor of The Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology, November 2001. He is also the Associate Editor of the IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing. From 2002, he is also the Associate Editor of Proceedings of the IEEE.

Dr. Chen received the Best Paper Award from ROC Computer Society in 1990 and 1994. From 1991 to 1999, he received Long-Term (Acer) Paper Awards annually. In 1992, he received the Best Paper Award of the 1992 Asia-Pacific Conference on Circuits and Systems in VLSI design track. In 1993, he received the Annual Paper Award of Chinese Engineer Society. In 1996, he received the Out-standing Research Award from NSC, and the Dragon Excellence Award for Acer. He is elected as the IEEE Circuits and Systems Distinguished Lecturer from 2001–2002.

lgchen@video.ee.ntu.edu.tw