Partial-Result-Reuse Architecture and Its Design Technique for Morphological Operations With Flat Structuring Elements

Shao-Yi Chien, Shyh-Yih Ma, and Liang-Gee Chen, *Fellow, IEEE*

***Abstract*—Mathematical morphology operations are applied in many real-time applications, such as video segmentation. For real-time requirement, efficient hardware implementation is necessary. This paper proposes a new architecture named Partial-Result-Reuse (PRR) architecture for mathematical morphological operations with flat structuring elements. Partial results generated during calculation process are kept and reused in this architecture to reduce hardware cost. With PRR concept and self-affinity property of structuring elements, the proposed architecture is more cost-effective and more general than existing morphology architectures. Moreover, it can be combined with systolic array to give consideration to both flexibility and hardware cost. We also propose a methodology to generate PRR architecture. With graphic method, the PRR architecture can be easily generated, and it can deal with structuring elements of any shape. The very large scale integration implementation of the PRR architecture shows that the area of processing element is small and can be fully piplelined without large overhead. The maximum frequency of the chip is 200 MHz in simulation, while processing speed of 550 morphological operations/s on a 720 $\times$ 480 frame can be achieved.**

***Index Terms*—Mathematical morphology, morphology architecture, partial-result-reuse (PRR), video segmentation.**

I. Introduction

MATHEMATICAL morphology [1], which is derived from set theory, is very important in the field of digital image processing and computer vision [2]. It contains a lot of powerful tools for shape-based image processing and can be used for image analysis, image and video compression [3], error correction, noise suppresion, and video segmentation [4]. For image and video compression, connected operators, watershed transform, geodesic skeleton, and morphological interpolation technique are required [3]. Each of these operations are complicated operations which are combinations of a lot of basic morphological operations. Therefore, high speed morphological operations are needed for real-time coding applications. Another example is video segmentation. The most important functionality of MPEG-4 is content-based coding [5], which needs object shape information. Video segmentation, the methodology to give shape information, consequently becomes an essential part in MPEG-4 coding systems. Most of the existing video segmentation algorithms [4], [6]–[8] need mathematical morphology as basic operations; therefore, high speed morphological operations are also urgently required. For example, watershed transform with multiscale gradient has been proven to give very good segmentation quality [8], and at least 11 morphological operations with a 3×3 structuring element are needed for each frame, that is, the processing speed of 330 frames/s should be achieved to support real-time (30 frames/s) video segmentation. However, each morphological operation with a 3×3 structuring element on a CIF (352×288) frame needs the processing capability of 700 MOPS to achieve real-time requirement, meaning that 7.7 GOPS is needed to support real-time multiscale gradient transform. It cannot be achieved without hardware accelerators.

Many architectures for morphological operations have been proposed [9]–[13]. Loui *et al.* [9] developed a flexible architecture for morphological image processing. It is based on one-dimensional (1-D) pipeline architecture and can be extended to deal with two-dimensional (2-D) images with decomposition property of morphological operations. However, a lot of comparators are needed and a large comparator tree is required at the output of this architecture. From Loui's architecture, Malamas *et al.* [10] proposed a 1-D pipeline architecture to deal with binary morphological operations, but it is not suitable for 2-D morphological operations. Diamantaras and Kung [11] proposed a systolic array architecture. The architecture has high throughput, but the number of comparators required is in proportional to frame height, which is often larger than that of Loui's architecture. Ruetz and Brodersen [12] showed a delay line architecture with decomposition technique. Structuring elements are decomposed vertically and horizontally, and the hardware cost is dramatically reduced. Sheu *et al.* [13] presented a data-reuse architecture. With pipelined technique, the number of comparators is almost the same as that of Ruetz's architecture; however, the throughput is lower. All of them are valuable works, but the hardware cost still becomes enormous when encountering large structuring elements in these architectures. On the other hand, content addressable memory (CAM) based architecture is also proposed for mor-

Manuscript received May 5, 2002; revised January 16, 2004. This work was supported in part by National Science Council, R.O.C., under Grant NSC90-2213-E-002-095. The chip fabrication is supported by Natinal Chip Implementation Center (CIC). This paper was recommended by Associate Editor H. Watanabe.

S.-Y. Chien is with the Media IC and System Laboratory, Graduate Institute of Electronics Engineering, Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: sychien@cc.ee.ntu.edu.tw).

S.-Y. Ma is with Vivotek Inc. 6F, Taipei 235, Taiwan, R.O.C. (e-mail: syma@ieee.org).

L.-G. Chen is with the DSP/IC Design Laboratory, Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: lgchen@cc.ee.ntu.edu.tw).

Digital Object Identifier 10.1109/TCSVT.2005.852622

phology operations, which is proposed by Ikenaga and Ogura [14], [15]. By use of CAM, morphology operations with large structuring elements can be applied on the input image in a few microseconds, and different structuring elements can be supported in a hardware without re-designing the hardware architecture by changing the control sequences of the CAM. Although the processing speed of CAM-based architecture is very high, the data loading time for the CAM is long, and the hardware cost and power consumption become very large for large images. Consequently, a more cost-effective architecture should be necessary.

To avoid redundant computation, partial results generated during the calculation process should be kept and reused. Wang and He's algorithm [16] records the maximum (minimum) of current window when computing dilation (erosion), but it is designed only for 1-D structuring elements. Lam and Li's algorithm [17] is designed for 2-D (flat) structuring elements. It records the values and positions of points having maximum (minimum) of previous search windows, and a probability model is used to determine the optimal way to find the results of current search window. Gil and Werman's algorithm [18] and its improved version [19] can reduce the complexity to almost constant. These algorithms can reduce the computational complexity of morphological operations; however, applying these algorithms will lose the regularity of morphological operations so they are not suitable for hardware implementation. Some hardware architectures derived with the partial-result-reuse (PRR) concept are also proposed. Pitas's architecture [20], [21] and Ong's [22] are two of them, but their architectures are not optimized, and the hardware can be further reduced. Coltuc and Pitas [23] proposed an optimal solution for morphological operations with rectangular structuring elements; morphological operations, however, often use various kinds of structuring elements, such as circle, diagonal line, octagon, and disk. Therefore, a more general PRR architecture should be developed.

In this paper, we propose a new architecture named PRR architecture for morphological operations with flat structuring elements, which is the most common type of morphological operations. The PRR architecture has several features. First, with graphic method, the PRR architecture can be designed very easily. In addition, when dealing with morphological operations with rectangular structuring elements, the PRR architecture can achieve optimal hardware cost. The PRR architecture, moreover, can deal with arbitrary structure elements, such as disk, without large overhead. It can also be combined with systolic array architecture to give consideration to both flexibility and hardware cost. Finally, this architecture can be used not only for morphological operations but also for other running semigroup operations [18] (or T operations [23]).

In Section II, basic operations of mathematical morphology are introduced. Then the proposed PRR architecture is presented in Section III. Section IV compares the PRR architecture with other existing architectures, and Section V shows how to combine the PRR architecture with systolic array. The implementation of this architecture is shown in Section VI. Some applications of the PRR architecture are shown in Section VII. Finally, Section VIII gives a conclusion.

II. Morphological Operations

There are a lot of operations in mathematical morphology, such as dilation, erosion, opening, closing, hit-and-miss, thinning, and thickening. All of these operations are neighborhood operations, the operation result of each point depends only on the points in its neighborhood. There are two operands of morphological operations: input signal I, which is usually an image, and structuring element B, which records the range and shape of neighborhood region (or search window). All morphological operations are combinations of two basic operations: dilation and erosion.

Let $I(x,y)$ and $B(x,y)$ be 2-D gray-scale signals. $I(x,y)$ is input signal and $B(x,y)$ is structuring element. Let $I:\phi\to E$ and $B:\beta\to E$. The dilation operation, which is denoted by $\oplus$ can be expressed as the following equation:

$$I\oplus B(x,y)=\max\{I(x-i,y-j)+B(i,j)|(i,j)\in\beta,\quad \text{and } (x-i,y-j)\in\phi\}. \quad (1)$$

In common situations, flat structuring elements are used, the flat dilation operation is:

$$I\oplus B(x,y)=\max\{I(x-i,y-j)|(i,j)\in\beta,\quad \text{and } (x-i,y-j)\in\phi\}, \quad (2)$$

which is also called running maximum filter. The dilation operation takes max operation as key operation, hence it will enhance and grow the bright parts of an image.

On the other hand, flat erosion operation, which is used to enhance the dark regions of an image and denoted by $\ominus$ can be expressed as the following equation:

$$I\ominus B(x,y)=\min\{I(x+i,y+j)|(i,j)\in\beta,\quad \text{and } (x+i,y+j)\in\phi\}. \quad (3)$$

A flat erosion operation is also called running minimum filter.

The combinations of these two basic morphological operations can form other morphological operations. For example, the opening operation, which is denoted with $\circ$, and the closing operation, which is denoted with $\bullet$, can be expressed as the following equations respectively:

$$I\circ B=(I\ominus B)\oplus B \quad (4)$$
$$I\bullet B=(I\oplus B)\ominus B. \quad (5)$$

The opening and closing operations are often used to simplify image and eliminate noise in an image.

Note that the notations of mathematical morphology in [1] and [24] are slightly different . In this paper, the notation of mathematical morphology and illustration of structuring elements are taken from Haralick [2], [24].

III. Partial-Result-Reuse Architecture

The basic morphological operations are local operations, and the data flow is very regular; thus it is suitable for hardware implementation. Many of the existing architectures make use of data-reuse technique to reduce the amount of memory access in these operations. However, a lot of calculations in those architectures are redundant for flat structuring elements, which

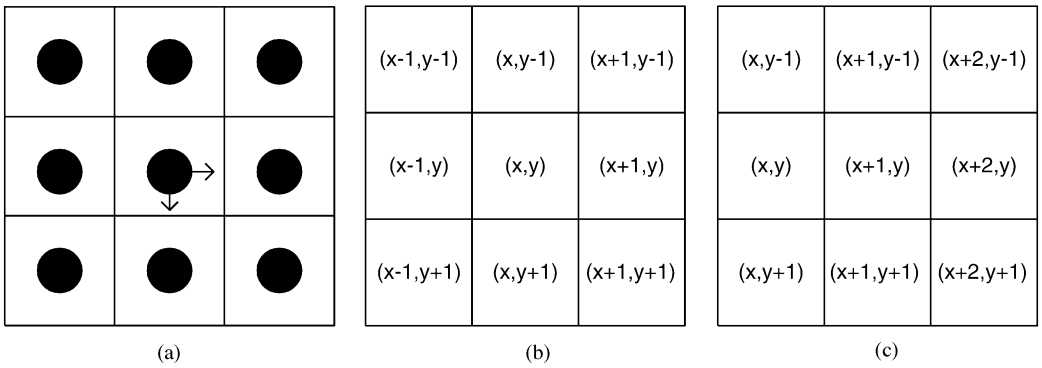

Fig. 1. (a) 3×3 structuring element. (b) Pixels needed when computing $I \oplus B(x,y)$. (c) Pixels needed when computing $I \oplus B(x+1,y)$.

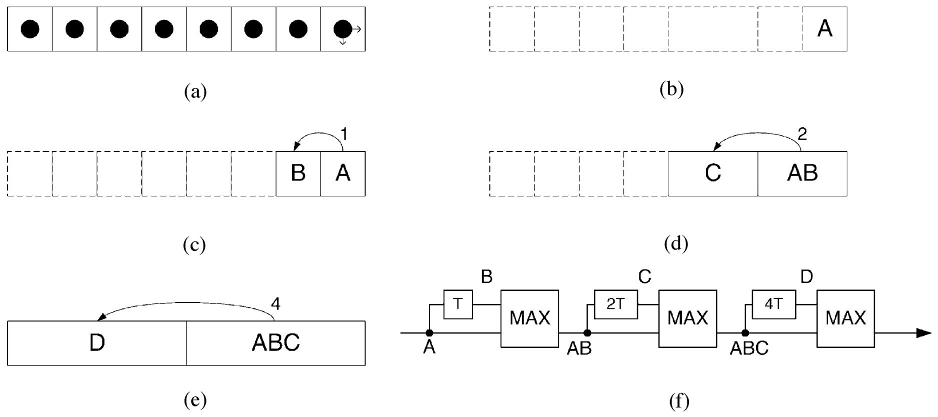

Fig. 2. PRR architecture for a 1×8 structuring element.

means the hardware cost can be further reduced. The concept is illustrated in the following example.

When dilating with a 3×3 structuring element, which is shown in Fig. 1(a), the dilation result of point (x,y) is the maximum of the nine points in Fig. 1(b), and the result of adjacent point $(x+1,y)$ is the maximum of the nine points in Fig. 1(c). It is obvious that many operations are duplicate and redundant. If the result of $\max\{I(x,y-1), I(x,y), I(x,y+1), I(x+1,y-1), I(x+1,y), I(x+1,y+1)\}$ can be propagated, only three comparators instead of eight comparators are needed.

The PRR concept can be used for running max operations and other running semigroup operations (or T operations) [23]. This kind of operations, such as max, min, $+$, and $\times$, has several important properties [23] as follows:

1) associativity: $(xTy)Tz = xT(yTz)$;
2) commutativity: $xTy = yTx$;
3) idempotence: $xTx = x$

where T denotes one semigroup operation.

On the basis of the PRR concept and the above three properties, the PRR architecture is proposed. First, the PRR architecture for structuring elements with self-affinity property is shown. After that, the PRR architecture for other kinds of structuring elements is proposed. Finally, the boundary condition of the PRR architecture is described.

A. Structuring Elements With Self-Affinity Property

Almost all general structuring elements have self-affinity property. In other word, a small basic element can be duplicated several times to generate a large structuring element, which can also be duplicated several times to generate other larger affined structuring elements. This property can be used to find the optimal ways for PRR and reduce the computation and hardware cost of morphological operations.

The following example can serve as an illustration. The procedure involves two steps: the self-affine step is to find the way to reuse partial results while the architecture design step is to use the results of the first step to design the hardware architecture. For simplification, only implementation of flat dilation is described in this paper, and flat erosion can be implemented with the same design technique. Flat dilation with a 1×8 structuring element is first considered, which is shown in Fig. 2(a). The basic element is a square. We start from the current point, which is denoted by A in Fig. 2(b). Duplicating square A and shifting it left one point to B, as shown in Fig. 2(c), give a larger rectangle AB consisting of A and B as shown in Fig. 2(d). With the same procedure, duplicating AB and shifting it two points to C generate a 1×4 rectangle ABC as shown in Fig. 2(e), and the whole structuring element can be generated by duplicating ABC and shifting it four points to D. After that, we can make use of this procedure to compute dilation with less operations as shown in the following equations:

$$I \oplus B(x) = \max\{I(x), I(x-1), I(x-2), I(x-3), I(x-4), I(x-5), I(x-6), I(x-7)\} \quad (6)$$
$$= \max\{A_x, B_x, C_x, D_x\} \quad (7)$$

where

$$A_k = \max\{I(k)\} = I(k) \quad (8)$$

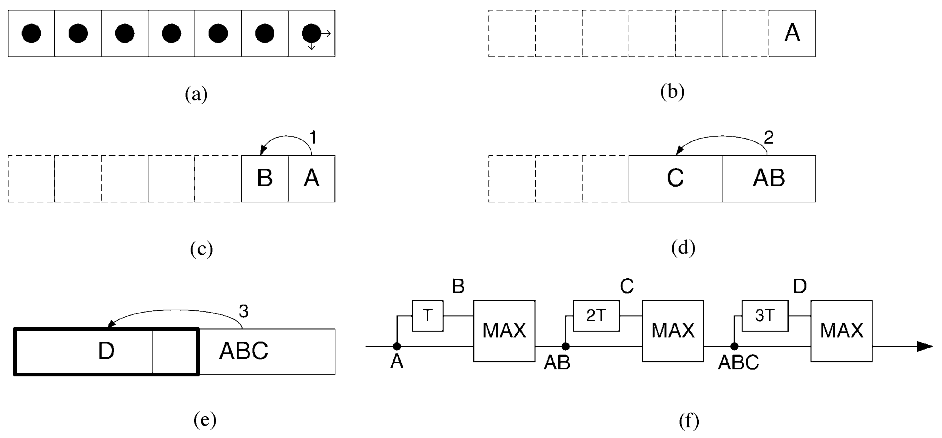

Fig. 3. PRR architecture for a 1×7 structuring element.

$$B_k = \max\{I(k-1)\} = I(k-1) = A_{k-1} \quad (9)$$
$$C_k = \max\{I(k-2), I(k-3)\} = AB_{k-2} \quad (10)$$
$$D_k = \max\{I(k-4), I(k-5), I(k-6), I(k-7)\} = ABC_{k-4} \quad (11)$$
$$AB_k = \max\{A_k, B_k\} \quad (12)$$
$$ABC_k = \max\{AB_k, C_k\}. \quad (13)$$

Finally, only three comparators are needed to compute (7), instead of seven comparators for calculating (6). The value of A_x is equal to that of current point $I(x)$, and the value of B_x is equal to that of A_k when $k = x-1$. Besides, the values of C_x and D_x are equal to those of AB_k and ABC_k when $k = x-2$ and $k = x-4$, respectively. Here, the value of B_x, C_x, and D_x are the partial results of former computation. The corresponding PRR architecture for a 1×8 structuring element is shown in Fig. 2(f). Only three comparators and seven delay elements (T) are needed. It is obvious that the same design technique can deal with operations with all power-of-2 length 1-D structuring elements.

If the length of the structuring element is not power-of-2, the same procedure can also be applied as illustrated in the following example:

In Fig. 3(b)–(e), the self-affine procedure for a 1×7 structuring element is shown. Note that in Fig. 3(e), the rectangle ABC, which is illustrated with normal block, and D, which is illustrated with bold block, overlap. It will not introduce any problem when computing the result as shown in the following equations:

$$\begin{aligned}\max\{A,B,C,D\} &= \max\{I(x), I(x-1), \max(I(x-2), I(x-3)), \\ &\qquad \max(I(x-3), I(x-4), I(x-5), I(x-6))\} \\ &= \max\{I(x), I(x-1), I(x-2), \max(I(x-3), \\ &\qquad I(x-3)), I(x-4), I(x-5), I(x-6)\} \\ &= \max\{I(x), I(x-1), I(x-2), I(x-3), I(x-4), \\ &\qquad I(x-5), I(x-6)\} \\ &= I \oplus B(x). \end{aligned} \quad (14)$$

The overlapping part introduces the operation $\max(I(x-3), I(x-3))$, which is equal to $I(x-3)$ because of the idempotence property of semigroup operations. The corresponding PRR architecture for a 1×7 structuring element is shown in Fig. 3(f). Only three comparators and six delay elements are needed. It is shown that this design technique can deal with operations with arbitrary 1-D structuring elements.

The same technique can be extended to deal with morphological operations with 2-D structuring elements with self-affinity property. For example, when the structuring element is 8×8 as shown in Fig. 4(a), the self-affine procedure is shown in Fig. 4(b)–(h). In Fig. 4(c), (e), and (g), W denotes the width of image. When the data of a point is propagated vertically to the next row, W delay elements are required since the data are inputted and manipulated in raster-scan manner. The direction of duplicate-and-shift procedure is both horizontal and vertical, which implies that the partial results are reused in two directions. Same as (7), the PRR computing procedure can be shown by the following equations:

$$\begin{aligned} I \oplus B(x,y) &= \max\{I(i,j) | (x-7) \le i \le x, (y-7) \le j \le y\} \\ &= \max\{A_{x,y}, B_{x,y}, C_{x,y}, D_{x,y}, E_{x,y}, F_{x,y}, G_{x,y}\} \end{aligned} \quad (15)$$

where the meanings of $A_{x,y}$, $B_{x,y}$, $C_{x,y}$, $D_{x,y}$, $E_{x,y}$, $F_{x,y}$, and $G_{x,y}$ are similar to those in (7) and can be found in Fig. 4.

The associate architecture for a 8×8 structuring element is presented in Fig. 4(i). Only six comparators and $7W + 7$ delay elements are needed. The number of comparators required for an $n \times n$ structuring element is

$$C(n) = 2\lceil \log_2 n \rceil \quad (16)$$

which has been proved to be the optimal solution in [23].

Besides rectangular structuring elements, structuring elements of other shapes, such as disk and diagonal line, can also be handled. Disk-shaped structuring elements are very useful; however, almost all existing architectures with PRR concept cannot deal with them. Fortunately, like rectangle, disk also has self-affinity property. It can be generated by duplicating a basic element, cross. For example, when the structuring element is a disk with diameter of 5, the self-affine procedure is shown in Fig. 5(b)–(d). First, duplicate A to form $ABCDE$, which is in the shape of a cross. Then duplicate $ABCDE$ and shift it upper-right to F to form $ABCDEF$. Finally, duplicate

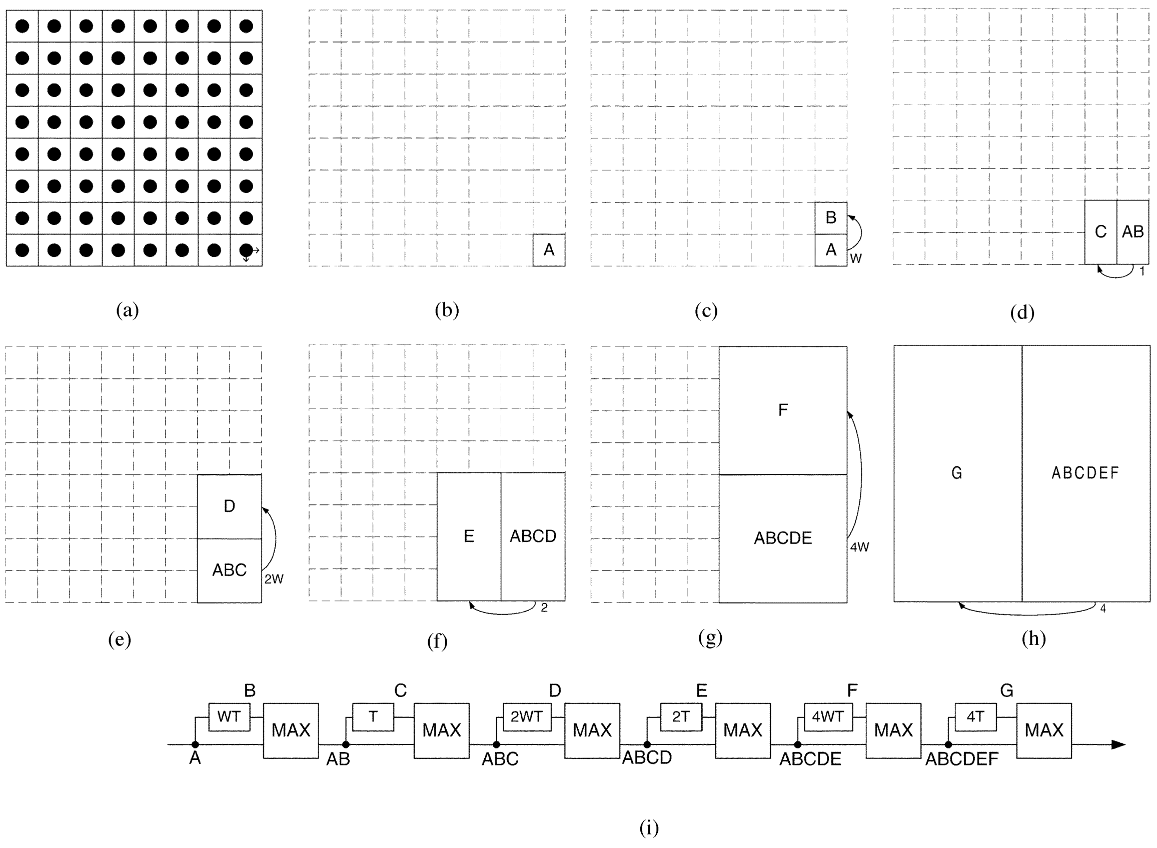

Fig. 4. PRR architecture for a 8 × 8 structuring element. W denotes the width of image. It requires W delay elements to propagate data vertically to the next row since the data are inputted and manipulated in raster-scan manner.

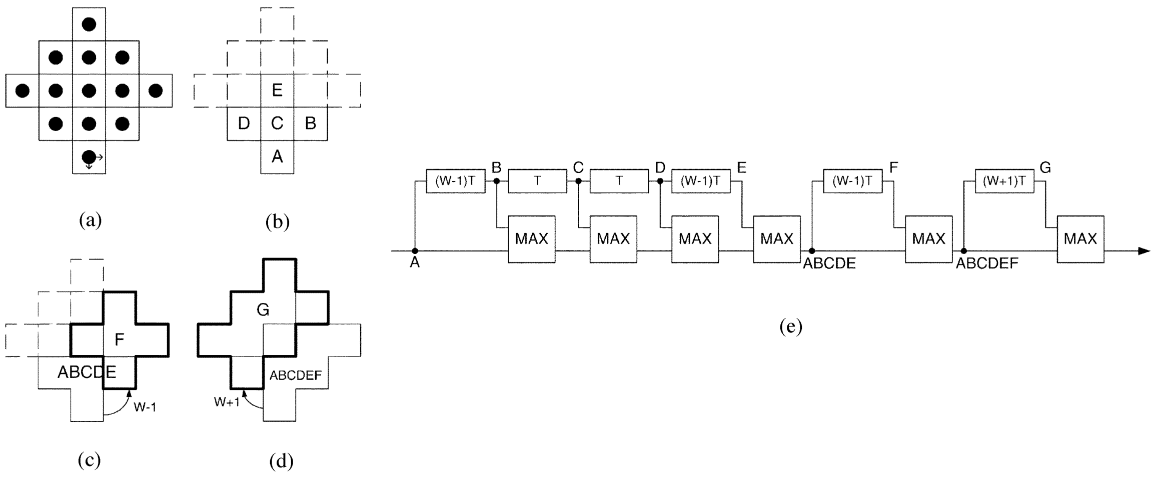

Fig. 5. PRR architecture for a disk structuring element with diameter of 5.

$ABCDEF$ and shift it upper-left to G to form the disk structuring element. With this procedure, the corresponding PRR architecture is shown as Fig. 5(e). Only six comparators and $4W$ delay elements are needed to implement this operation. The number of comparators required for a disk structuring element with diameter of n is

$$C(n) = \begin{cases} 2\lceil \log_2(n-1)\rceil + 2, & \text{if } n = \text{odd} \\ 2\lceil \log_2 n\rceil + 2, & \text{if } n = \text{even}. \end{cases} \quad (17)$$

Similar to the PRR procedure for a 1 × 8 structuring element, the PRR architecture for an 8-length diagonal line structuring element can be generated as shown in Fig. 6. Only three comparators and $7W + 7$ delay elements are needed to implement this operation.

B. Arbitrary Structuring Elements

Although most of the usual structuring elements have self-affinity property, structuring elements without this property are sometimes used for special purposes, such as hit-and-miss operations. For these structuring elements, another design technique can be employed. Only one-order PRR is considered in this technique, that is, all reused data are generated from the

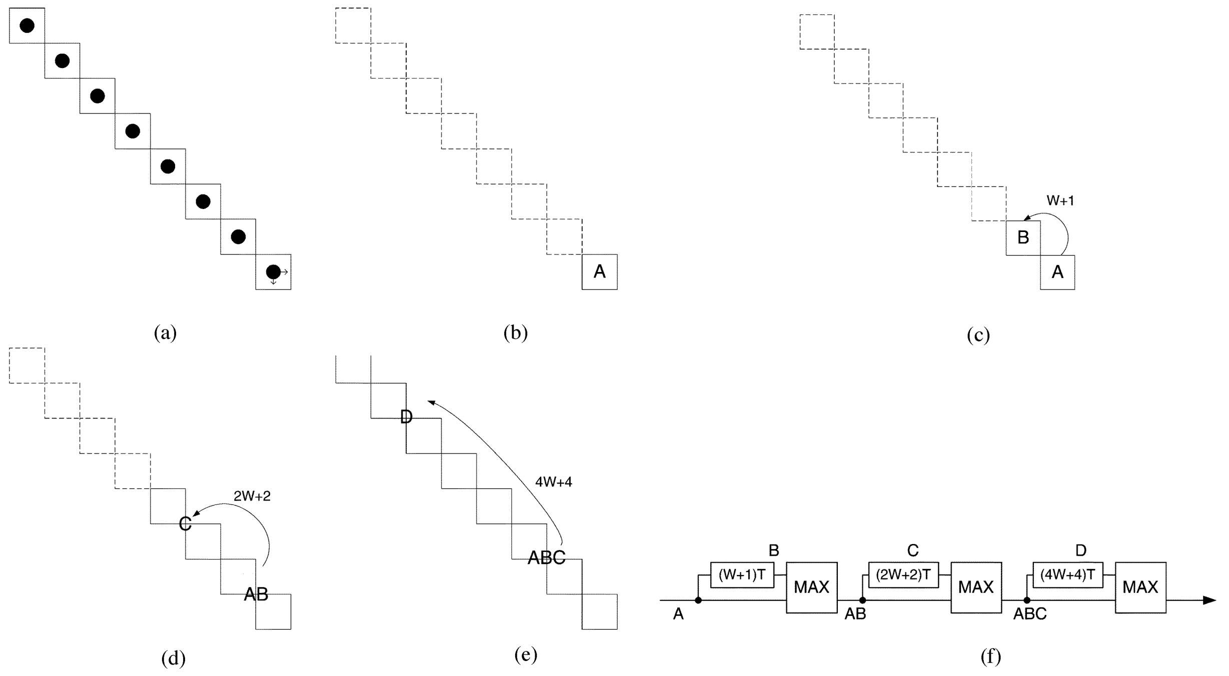

Fig. 6. PRR architecture for a diagonal line structuring element.

original data or only one reused data. For example, (12) and (13) are not equations of one-order PRR since two reused data are involved. The procedure involves three steps. First, the structuring element is divided into several exclusive segments according to different delay number considered, that is, if n delay number is considered, only partial results generated n points before can be reused. Then all the segmentation results are combined. Finally, according to the segmentation, the partial results in each search window and the optimal way to reuse them is determined, thus the PRR architecture can be designed.

The structuring elements are divided into several exclusive segments with different delay number consideration, and each segment stands for a partial result from the search window of the previous point and will be reused. The goal of segmenting is to fully reuse partial results, thus reducing the number of operations required. For example, if the structuring element is a 3×3 square as shown in Fig. 7(a) and one delay number is considered, the search window of point (x, y), which contains $I(x,y), I(x-1,y), I(x+1,y), I(x,y-1), I(x-1,y-1),$ $(x+1,y-1), I(x,y-2), I(x-1,y-2), (x+1,y-2)$, is denoted with set $S(x,y)$. The overlapping points of $S(x,y)$ and $S(x-1,y)$ are in the set $S(x,y) \cap S(x-1,y)$, and they are shown with solid blocks in Fig. 7(b), and the points appearing at first time, which are in the set $S(x,y) \setminus S(x-1,y)$, are shown with dashed blocks in Fig. 7(b). Only the overlapping points should be contained in the segments. If the segmentation is as shown in Fig. 7(b), the maximum of areas A and B are the reused partial results. The final result can be calculated by $\max\{A, B, I(x,y), I(x,y-1), I(x,y-2)\}$, and the values of $\max\{I(x,y), I(x,y-1), I(x,y-2)\}$ and the maximum of area B will be propagated to the next point. These segments should be as few as possible, which implies that the size of these segments should be as large as possible.

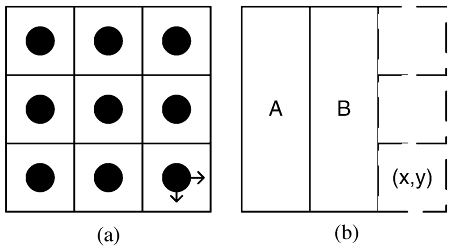

Fig. 7. Illustration of segmentation for PRR. (a) 3×3 structuring element. (b) the segmentation for PRR.

There are two propositions for deriving the optimal segmentation:

Proposition 1: Any segment should not across the overlap boundaries.

Exposition: When only one delay number is considered, for a segment $S_0(x,y)$, if $S_0(x,y) \cap S_0(x+1,y) \neq \emptyset$, the associated partial result to be propagated to the next point is

$$\max\{S_0(x+1,y)\} = \max\{\max\{S_0(x+1,y) \cap S_0(x,y)\} \\ \max\{S_0(x+1,y) \setminus S_0(x,y)\}\}. \quad (18)$$

From (18), it is obvious that not only the value of $\max\{S_0(x,y)\}$ but also the value of $\max\{S_0(x,y) \cap S_0(x+1,y)\}$ should be propagated from the previous point, meaning that an extra storage element is needed. If we divide $S_0(x,y)$ into two segments, $S_1(x,y) = S_0(x,y) \cap S_0(x-1,y)$ and $S_2(x,y) = S_0(x,y) \setminus S_0(x-1,y)$, the associated partial results to be propagated to the next point are

$$\max\{S_1(x+1,y)\} = \max\{S_0(x+1,y) \cap S_0(x,y)\} \quad (19)$$

$$\max\{S_2(x+1,y)\} = \max\{S_0(x+1,y) \setminus S_0(x,y)\}. \quad (20)$$

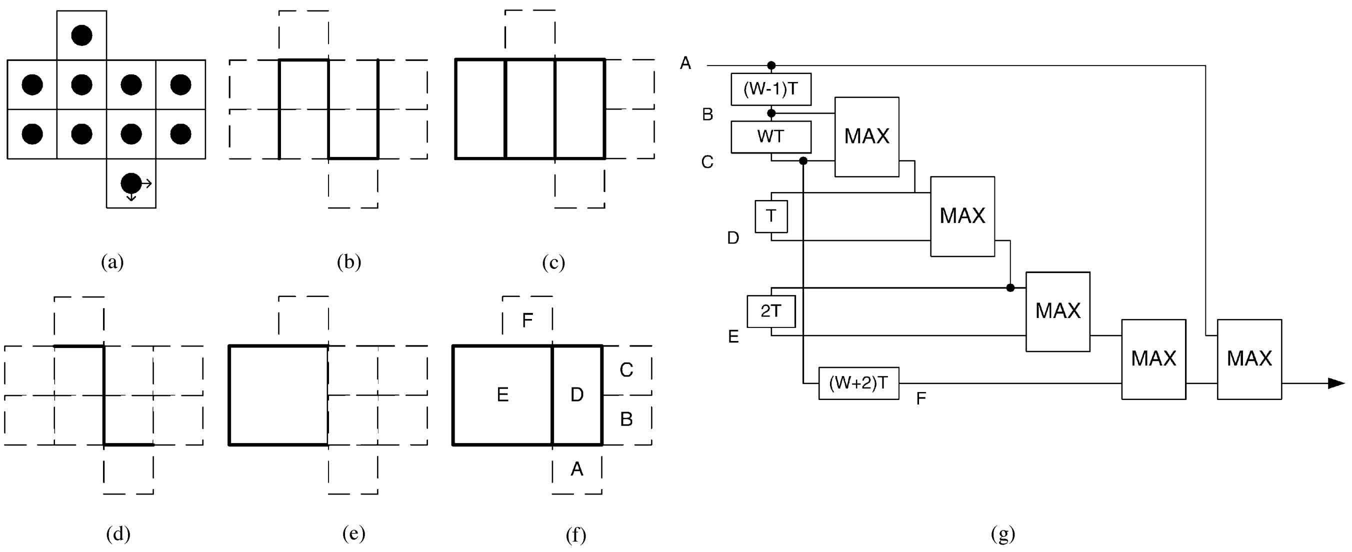

Fig. 8. PRR architecture for an example arbitrary structuring elements.

Although an extra comparator is needed to calculate the result of the current search window, $\max\{\max\{S_1(x,y)\}, \max\{S_2(x,y)\}\}$, but one less comparators are needed to compute (19) and (20) instead of (18). Moreover, the memory required is the same. Although the performance of these two kinds of segmentation are almost the same, the latter gives a systematic way to design the segmentation. As mentioned above, the larger the segment, the better it is. Assume that the whole overlapping area is contained in a single segment, $S_0(x,y) = S(x,y) \cap S(x-1,y)$, which is the largest possible of segment, the above procedure can formulate the following equations:

$$\begin{aligned}&\forall\, i,j \in Z,\ \text{if } S_i(x,y) \cap S_i(x+j,y) \neq \emptyset,\\ &\text{segment } S_i(x,y) \text{ into } S_i(x,y) \cap S_i(x+j,y)\\ &\text{and } S_i(x,y) \setminus S_i(x+j,y).\end{aligned}$$

The procedure can be simplified into a single rule: *the segments should not across the overlap boundary.* The overlap boundary when n delay elements are considered can be defined as the boundary of $[S(x,y) \cap S(x-n,y)] \cap [S(x+kn,y) \cap S(x+(k-1)n,y)] | k \in Z$ in $S(x,y)$. This procedure achieves optimal segmentation since the performance of this segmentation is the same as the one that yields the largest segment. ■

Proposition 2: The width of a segment should be maximized.

Exposition: The width of a segment is defined as the length of the border perpendicular to the direction of partial-result propagation. This rule is trivial and will not be further explained in this paper. Only one direction of PRR is considered in this technique because the width of segments often cannot be maximized if two directions are considered at the same time. ■

Next, all the segmentation results with different number of delay elements considered are combined. The rule of combination is to choose as large as possible a segment from as few delay elements as possible. The reason is that a larger segment means more efficient of PRR, and the fewer delay elements means less registers required.

This technique can be illustrated by the following example. In Fig. 8(a), the target structuring element is shown. In Fig. 8(b) and (c), the overlap boundary and segmentation by considering one delay element are shown, and those by considering two delay elements are shown in Fig. 8(d) and (e). Note that the solid blocks mean that the required data here can be reused from their previous points. The combination of Fig. 8(c) and (e) is shown in Fig. 8(f). Segment D comes from Fig. 8(c), meaning that it can be reused with only one delay element, while segment E comes from Fig. 8(e), meaning that it can be reused with two delay elements. The corresponding PRR architecture is shown in Fig. 8(g).

C. Boundary Condition

The operations at boundaries are different from those in the interior of a frame. Two solutions can be employed to deal with the boundary condition. The first one is to add extra control circuits. When dealing with boundary points, some nodes, which are originally connected to outputs of registers, should be fixed at 0 for dilation or 255 for erosion. However, adding extra multiplexers and control units will affect the cost-efficiency of the PRR architecture. The second one, which is also the recommended one, is to apply padding technique to input frames. The padding area can be shown as the following procedure:

For every points (x,y) in a frame
point $(k,l) = (x+i, y+j) | (i,j) \in B$
belongs to the padding area if point (k,l)
is not a point of the frame.

For a 5×5 structuring element, which is shown in Fig. 9(a), the padding area is shown in Fig. 9(b). It will be filled with 0 for dilation or 255 for erosion. The hardware then manipulates the padded frame instead of the original frame. The result of operation is correct since the original frame are in the interior of the inputted frame. This scheme required post-processing of cutting out the padding area to find the output data, which is simple and can be done by software on host computer.

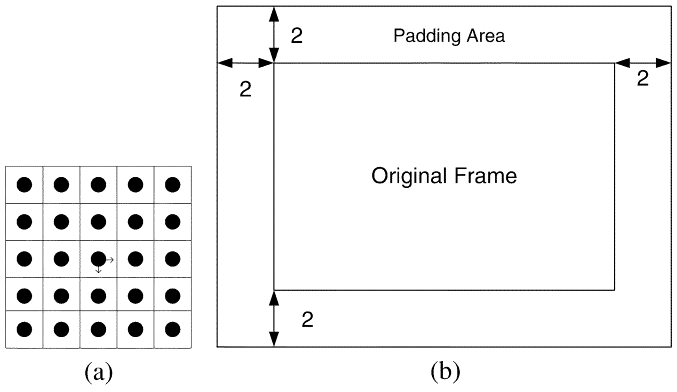

Fig. 9. Padding for boundary conditions. (a) 5×5 structuring element. (b) Padded frame.

TABLE I
COMPARISON OF HARDWARE EFFICIENCY BETWEEN PROPOSED MORPHOLOGY ARCHITECTURE AND OTHER ARCHITECTURES

Architecture	Comparator Count	Number of Delay Elements	Estimated Gate Count[a]	Cycles Required Per Frame
Pitas[20]	8	7W+7	448W+840	W(H+7)+6
Coltuc[23]	6	6W+6	384W+678	W(H+6)+5
Ong[22]	7	[6W+6]+1	[384W+384]+407	7WH
Diamantaras[11][b]	48	[6W]+42	[384w+2688]+2352	W(H+6)
Ruetz[12]	12	6W+6	384W+972	W(H+6)+5
Sheu[13]	13	[6W]+20	[384W]+1917	7WH+8
This work(PRR)	6	6W+6	384W+678	W(H+6)+5

[a]comparator: 49gates, 8-bits register: 64gates.
[b]With one PE

IV. Architectural Analysis

The comparison of the proposed architecture with other existing architectures is shown in this section. Morphological flat dilation operation with a 7×7 structuring element is considered, and W and H denote the width and height of input images, respectively. Comparator count, the number of delay elements, gate count, and required cycles per frame are taken into consideration. The smaller of these four values, the more efficient of the associated architecture. Gate count is estimated with SYNOPSYS Design Compiler. Note that, in order to make fair comparison, the input data of these architectures are all set in raster-scan manner. For those architectures whose input data are set in a special order, the extra delay number and gate count required for raster-scan input is estimated by ourselves, which are shown in square brackets. Table I shows the comparison results. It is shown that both Coltuc and Pitas's [23] and the PRR architecture are most efficient in gate count, number of delay elements, and clock cycles required. Table II shows the comparison of flexibility. Diamantaras and Kung's architecture is the most flexible one since it can be adapted for all kinds of structuring elements, including nonrectangle structuring elements and nonflat structuring elements. Ong and Sunwoo's architecture and the proposed one also have high flexibility. The other architectures are less flexible since they can only support limited structuring elements. Although the hardware cost of Coltuc and Pitas's architecture is the same as PRR architecture, it can only deal with rectangular structuring elements, while PRR architecture can also deal with arbitrary structuring elements.

TABLE II
COMPARISON OF FLEXIBILITY BETWEEN PROPOSED MORPHOLOGY ARCHITECTURE AND OTHER ARCHITECTURES

Architecture	Support Non-Flat Structuring Elements	Adaptability for Rectangle Structuring Elements	Support Non-Rectangle Structuring Elements
Pitas[20]	no	$2^m \times 2^n$, M×N[a]	no
Coltuc[23]	no	M×N	no
Ong[22]	yes	M×N	no
Diamantaras[11][b]	yes	M×N	yes
Ruetz[12]	no	M×N	no
Sheu[13]	yes	N×N	no
This work(PRR)	no	M×N	yes

[a]large effort is required for non-$2^m \times 2^n$ structuring elements
[b]With one PE

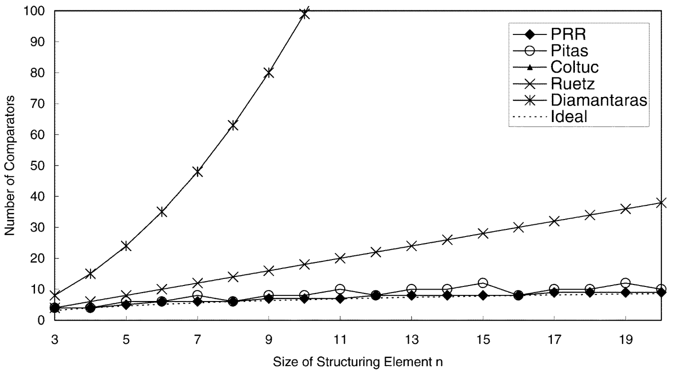

Fig. 10. Comparison between number of comparators required by difference architectures for $n \times n$ structuring elements.

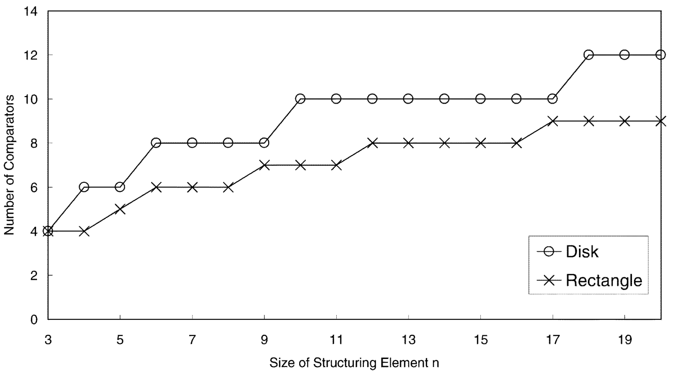

Fig. 11. Comparison between number of comparators required by n-diameter disk and $n \times n$ rectangular structuring elements.

A comparison between the number of comparators required for $n \times n$ structuring elements is shown in Fig. 10, where the ideal curve is $C(n) = 2\log_2(n)$. The number of comparators required by systolic array architecture [11] increases in square-law and will be very hardware-consuming when dealing with large structuring elements. Ruetz and Brodersen's architecture [12] needs comparators which increase linearly in number, and it will also become large when n is large. The PRR architecture is the optimal one, and the performance is very close to the ideal one, as shown in Fig. 10. The comparison between the number of comparators required for $n \times n$ rectangle and n diameter disk structuring elements is shown in Fig. 11. It shows that the

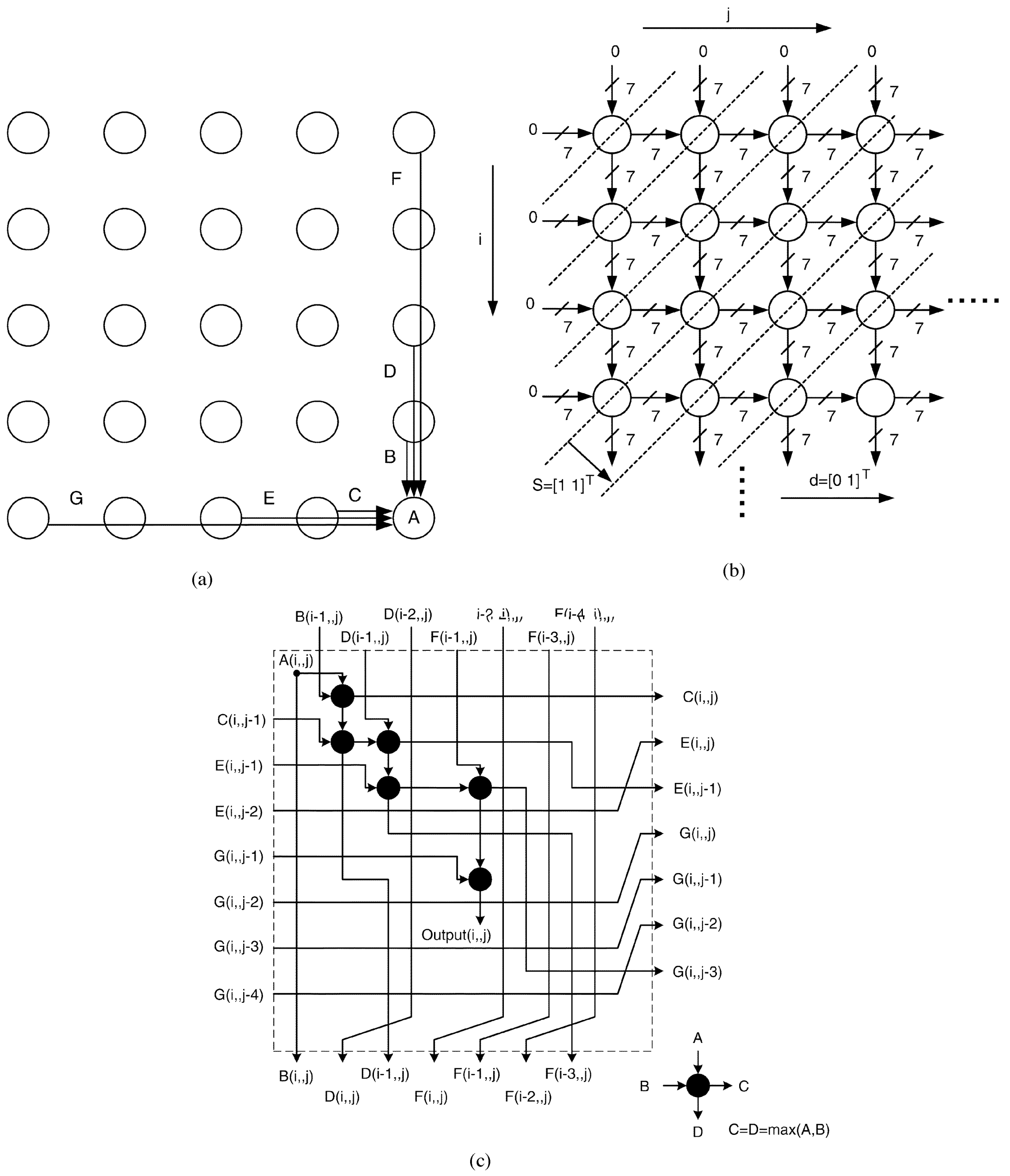

Fig. 12. (a) DG of dilation operation with 8×8 structuring element. (b) Localized DG. (c) Detailed node interior.

PRR architecture is also efficient with disk-shape structuring elements without the need for extra control circuits, which cannot be achieved by other architectures.

In summary, the PRR architecture has shown to be very cost-effective and efficient for flat morphological operations. It can deal with operations with structuring elements of arbitrary shape.

V. PRR Combined With Systolic Array

The PRR architecture can be further combined with systolic array architecture to enhance the flexibility of hardware design as well as maintain its cost-efficiency. The procedure for designing PRR architecture can be used to generate the dependence graph (DG) of the systolic array architecture. The following example shows the advantages of proposed technique.

Consider an 8×8 structuring element, which is shown in Fig. 4(a). With the self-affine procedure described in Section III, the DG can be generated as presented in Fig. 12(a). The operation performed by each node is $\max\{A, B, C, D, E, F, G\}$. Derived from Fig. 12(a), the localized DG is shown in Fig. 12(b). The detailed node interior is shown in Fig. 12(c). In each node, six comparators are needed, and the other parts of the node are connection lines. If we choose the projection vector $d = (0, 1)^T$, and the schedule vector $s = (1, 1)^T$ as shown in Fig. 12(b), the signal flow graph (SFG) can be derived with the systolic array architecture as shown in Fig. 13(a). In each processing element (PE), 14 registers and six comparators are needed. The number of PEs is equal to frame height, which is often infeasible for general cases. Folding technique [25] can be further employed to reduce the number of PEs. For example, if four PEs are sufficient to meet design specification, the folded PE array is shown in Fig. 13(b). Note that the architecture presented in Section III-A is a special case, which has only one PE.

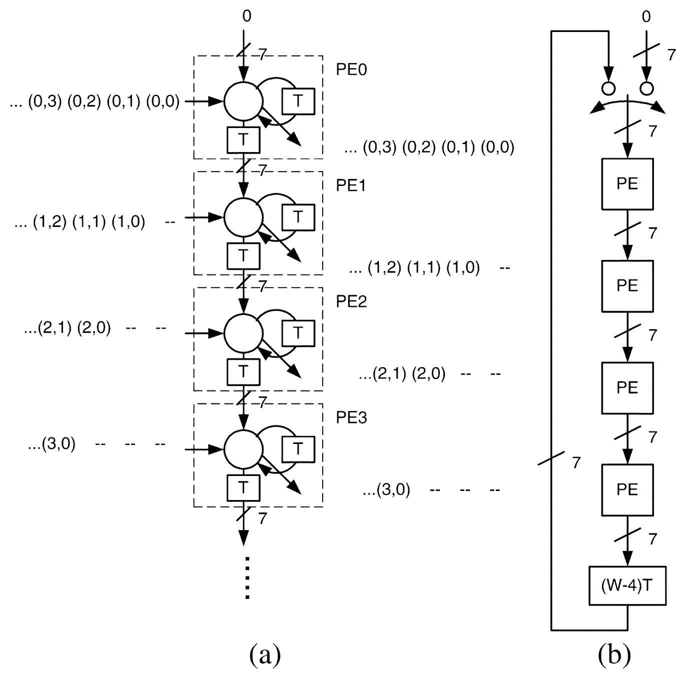

Fig. 13. (a) SFG and PE array of dilation operation with 8 × 8 structuring element. (b) Folded PE array.

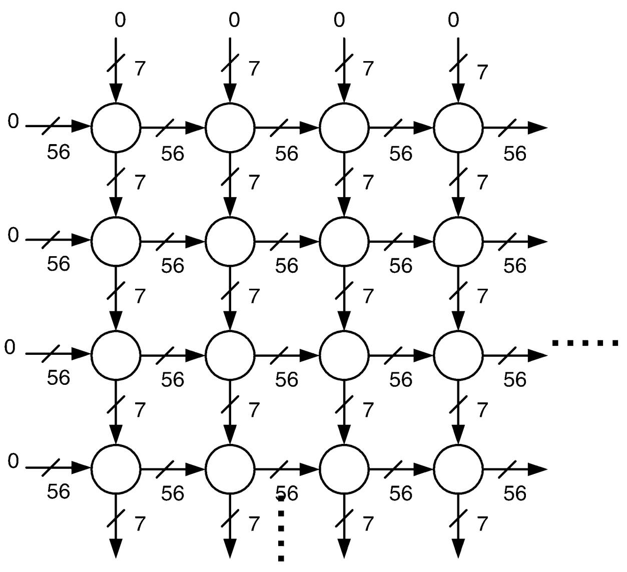

Fig. 14. DG of conventional systolic array.

Each node of the conventional systolic array [11], whose DG is shown in Fig. 14, has 56 inputs from left, 56 outputs to right, seven inputs from top, and seven outputs to bottom. The associated PE, which contains 63 registers and 63 comparators, is much larger than the PE derived with the PRR architecture. By generating DG with self-affine procedure instead of directly from single assignment code of algorithm to be mapped, the architecture becomes more efficient than the conventional systolic array.

VI. Implementation

VLSI implementation of PRR architecture is described in this section. Flat dilation with a five-diameter disk structuring element shown in Fig. 15(a) is implemented. Fig. 15(b) shows the associated fully pipelined PRR architecture. The number of comparators in the PRR architecture is minimized; therefore, delay elements dominate the chip area. The consideration of implementation delay lines is shown in the first subsection followed by the whole chip implementation.

A. Delay Line Consideration

A large number of delay elements are required to implement the delay lines in the PRR architecture. For such cases, RAM may be the most cost-efficient choice; however, it need an extra complicated control unit. For a prototyping chip, shift register is a better choice if power consumption is not considered.

In addition, a tiling technique can be used to shorten the length of delay lines. A frame is divided into several small frames (small tiles), which are then processed one by one. Finally, the results of all the small tiles are combined to give the final result. For general video format, such as QCIF (176 × 144), CIF (352 × 288), and CCIR601 (720 × 480), tiles which are 180 in width are good choices since 180 is the highest common factor of 180, 360, and 720. Further considering the chip area, 90 is finally chosen as the width of the tiles.

The same boundary condition occurs at the boundaries as described in Section III-C. The padding technique should also be applied to the small tiles, which can be shown as the following procudure:

For dilation, for every point (x, y) in each tile,
$(k, l) = (x + i, y + j) | (i, j) \in B$
If (k, l) is a point of the frame but not a
point of the tile, pad the point (k, l) to the tile.
If (k, l) is not a point of the frame and
also not a point of the tile, pad 0.

The tiling technique is illustrated in Fig. 16(a). Note that some parts of the tiles overlap, and the width of each tile becomes 94. The original frame in CIF format (360 × 243) shown in Fig. 16(b) can be segmented into four tiles as shown in Fig. 16(c). They are then inputted into the chip tile by tile, and the final result is assembled from the output of the chip.

B. Chip Implementation

The core parts of the chip consist of *MAX*s (comparators). The layout of *MAX* is shown in Fig. 17(a), which includes a 8-bits register. Fig. 17(b) shows the layout of the processing element, which contains six *MAX*s as shown in in Fig. 15(b).

When the morphological unit is integrated with other systems, such as DSP chips, the delay elements can be shared with the main memory of the systems. Only the processing element needs to be integrated. It will not introduce much overhead since the area is very small.

The die photo of this chip is shown in Fig. 18, and the chip specification is shown in Table III. The maximum frequency of this chip is 200 MHz. For video sequences in CCIR601 format, 550 frames can be processed per second, meaning that about 18 morphological operations can be applied to each frame in real-time (30 frames/s). To meet the requirement of the multiscale

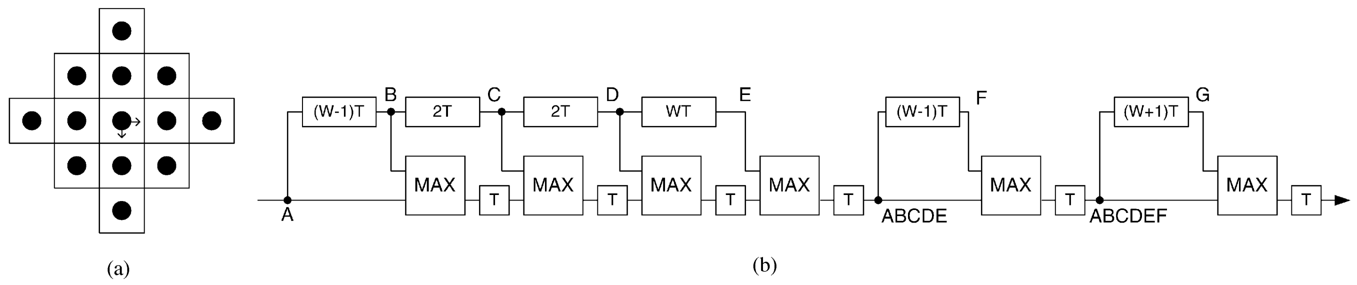

Fig. 15. Target of implementation: flat dilation with five-diameter disk structuring element. (a) Structuring element. (b) Fully pipelined architecture.

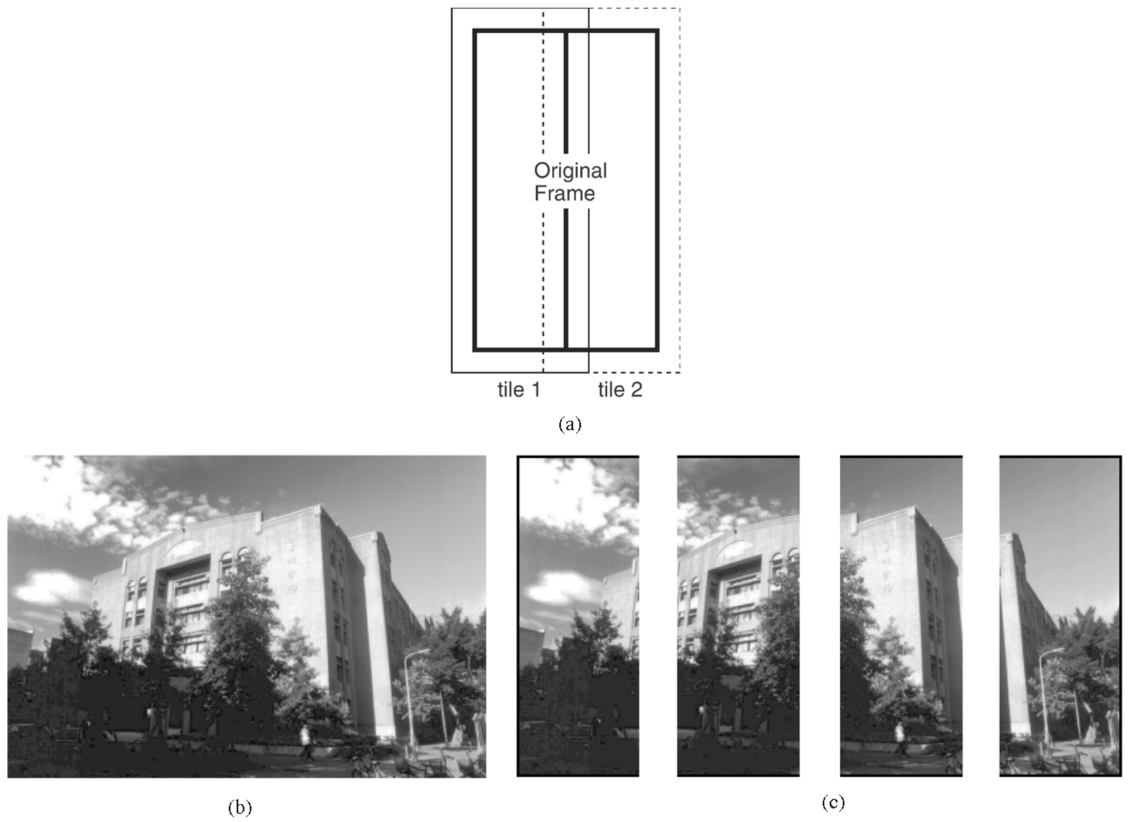

Fig. 16. Tiling technique. (a) Illustration. (b) Original frame. (c) Tiled frame.

gradient-based watershed described in Section I, only 11 morphological operations is needed; therefore, the frequency of 125 MHz is enough. Note that with software post-processing, the input frame format has no restriction. The output of this chip when the input frame is as shown in Fig. 16(b) can be found in Fig. 19.

The implementation of this chip reveals several features of the PRR architecture. With the PRR architecture, the area of the processing element can be dramatically reduced, which benefits the integration with other digital video processing systems. Moreover, although delay elements dominate the area in the chip, their area should not be taken into consideration since they are removable when the input data is not restricted to raster-scan manner. For example, if the morphological unit is integrated with other systems, many shared memory banks are available. The input data can be first loaded into those memory banks, and signal A, B, C, D, and E can be read out in parallel, where large amount of delay elements can be saved. Otherwise, the delay elements can be implemented with RAM, whose area is much smaller. Furthermore, the number of comparators required in the PRR architecture is minimized; therefore, the architecture can be fully pipelined without large overhead, thus resulting in the high clock frequency of this chip.

VII. Applications of PRR Architecture

As mentioned in Section I, PRR architecture can be applied to all running semigroup operations. Two applications of the PRR architecture are described in this section. The first one is the moving average filter, which is often employed in image processing systems for denoising and unsharp masking. The operation of a moving average filter is

$$\text{Output}\,(x,y) = \frac{1}{\#\beta}\sum_{(i,j)\in\beta} I(x-i, y-j) \tag{21}$$

where β is the search window of the moving average filter, and $\#\beta$ is the number of elements in the search window. The core operation of the moving average filter is addition, which is also

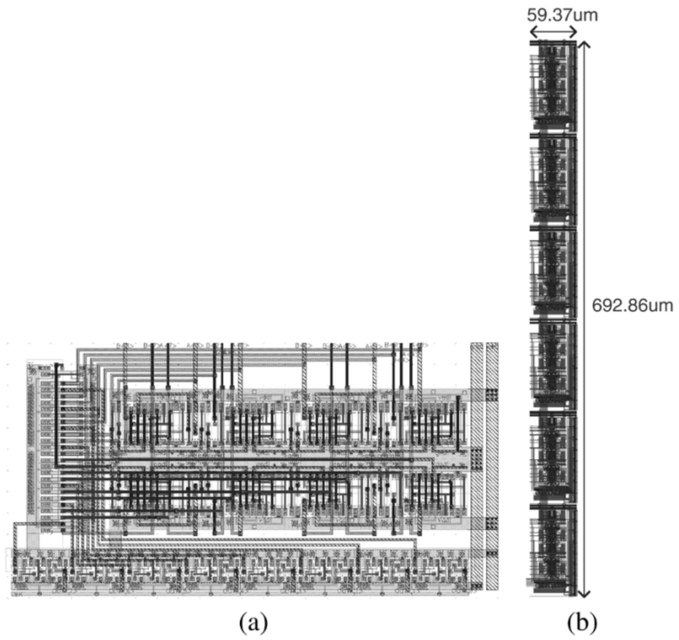

Fig. 17. Layout of (a) *MAX*. (b) Processing elements (6 *MAX*s).

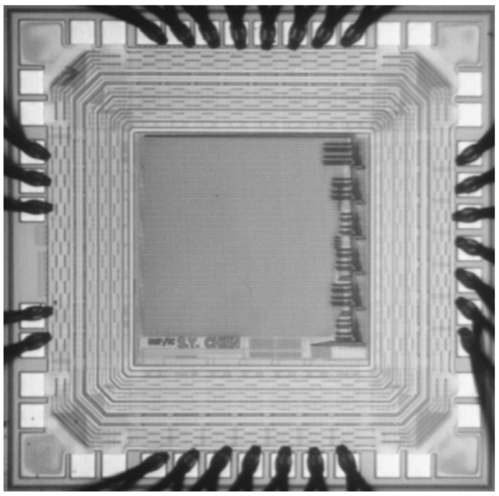

Fig. 18. Die photo of the chip.

TABLE III
CHIP SPECIFICATION

Process	TSMC 1P4M 0.35um
Area	1490umx1490um
Package	28 S/B
Transistor count	38488 (1740 in PE)
Maximum frequency	200MHz (simulated)
Power supply voltage	3.3 V
Power consumption	135.55 mW @ 125MHz

a semigroup operation; therefore, it can also be implemented with the PRR architecture. If the search window is the same as the five-diameter disk structuring element shown in Fig. 15(a), the associate hardware architecture can be seen in Fig. 20. Only six adders are needed in this architecture.

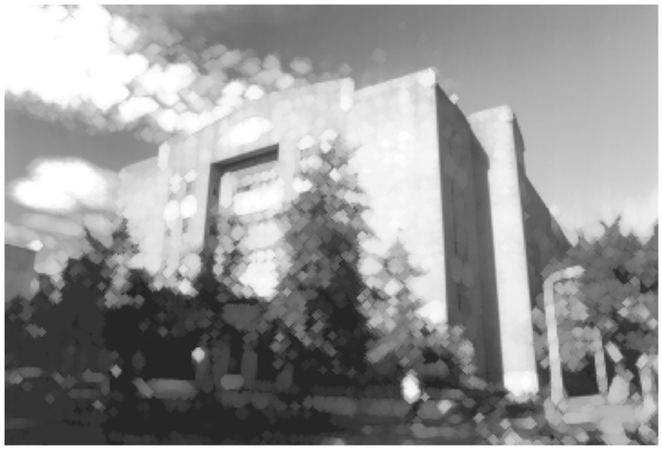

Fig. 19. Output of the chip which provides morphological operations with tiling. The image size is 360 × 243.

Median filter is also an important denoising filter in image processing. It is especially effective for shoot noise (or pepper-and-salt noise). The operation of the median filter is very computation-intensive and difficult to be implemented efficiently in hardware. Therefore, a pseudomedian filter, whose properties are similar to those of median filter, has been proposed [26]. If five points, a, b, c, d, and e, where c is the origin point, are in current search window, the output of pseudomedian filter is

$$\begin{aligned}\mathrm{PMED}(a,b,c,d,e) &\\ = \left(\frac{1}{2}\right) &\max[\min(a,b,c),\min(b,c,d),\min(c,d,e)]\\ + \left(\frac{1}{2}\right) &\min[\max(a,b,c),\max(b,c,d),\max(c,d,e)].\end{aligned} \tag{22}$$

It is obvious that the pseudomedian filter can be implemented with morphological operations as shown in the following equations:

$$\begin{aligned}\mathrm{PMED}(x,y) = \left(\frac{1}{2}\right)&(I\ominus B)\oplus B(x,y)\\ &+\left(\frac{1}{2}\right)(I\oplus B)\ominus B(x,y).\end{aligned} \tag{23}$$

If the search window is also the same as the five-diameter disk structuring element, the associate hardware architecture can be found shown in Fig. 21.

VIII. CONCLUSION

A novel PRR algorithm and architecture for mathematical morphology with flat structuring elements, which is very cost-effective, is proposed in this paper. Several examples demonstrate that the PRR architecture can be easily designed by graphic method. In addition, it can reduce hardware cost of morphological operations with arbitrary structuring elements using the PRR concept. For structuring elements with self-affinity property, the hardware can be further reduced with the proposed design technique. Some simulations have been made showing that this architecture is more efficient than the others. The PRR architecture, moreover, can be combined with

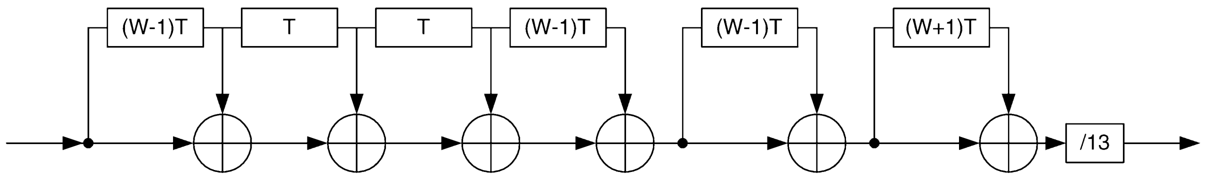

Fig. 20. PRR architecture of moving average filter.

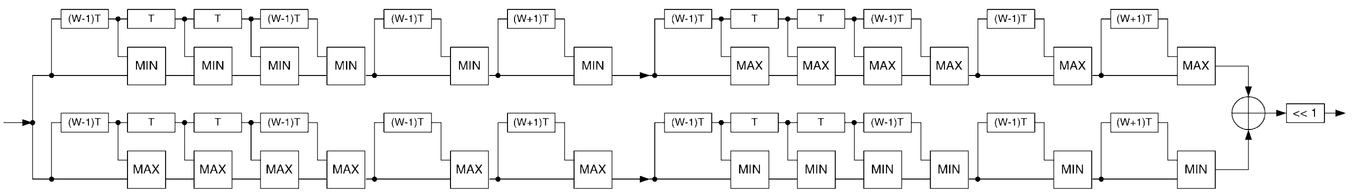

Fig. 21. PRR architecture of pseudomedian filter.

the systolic array architecture to design efficient hardware. VLSI implementation of the PRR architecture shows that the area of the processing element is quite small and suitable to be integrated with other DSP systems. The processing speed is high since the PRR architecture can be fully pipelined without large overhead. It is also shown that the PRR architecture can deal with other kinds of running semigroup operations.

Note that, the PRR architecture has other advantages. As shown in this paper, the PRR architecture can be designed with a systematic approach. Therefore, it is also suitable to be implemented as an electronic design automation (EDA) tool to accelerate the development process. With cell-based design flow as well, not only the fabrication cost but also the development cost of the PRR architecture is low. Furthermore, other than the application-specific integrated circuits (ASIC) example shown in this paper, the new algorithm, hardware architecture, and design techniques proposed in this paper can also be employed on other platforms, such as FPGA, DSP, and RISC, to further enhance the programmability for different structuring elements. It can also be used to further refine the control sequences of the CAM-based architecture [15] to shorten the processing time.

REFERENCES

[1] J. Serra, *Image Analysis and Mathematical Morphology*. London, U.K.: Academic, 1982.

[2] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*. Norwood, MA: Addison Wesley, 1992.

[3] P. Salembier, P. Brigger, J. R. Casas, and M. Pardàs, "Morphological operators for image and video compression," *IEEE Trans. Image Process.g*, vol. 5, no. 6, pp. 881–895, Jun. 1996.

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

[5] T. Sikora, "The MPEG-4 video standard verification model," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 1, pp. 19–31, Feb. 1997.

[6] 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," presented at the Workshop Consumer Electronics 2000, 2000.

[7] *Annex F: Preprocessing and Postprocessing*, ISO/IEC JTC 1/SC 29/WG11 N3056, 1999.

[8] D. Wang, "Unsupervised video segmentation based on watersheds and temporal tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 5, pp. 539–546, Sep. 1998.

[9] A. C. P. Loui, A. N. Venetsanopoulos, and K. C. Smith, "Flexible architectures for morphological image processing and analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, no. 1, pp. 72–83, Mar. 1992.

[10] E. N. Malamas, A. G. Malamos, and T. A. Varvarigou, "Fast implementation of binary morphological operations on hardware-efficient systolic architectures," *J. VLSI Signal Process.*, vol. 25, pp. 79–83, 2000.

[11] K. I. Diamantaras and S. Y. Kung, "A linear systolic array for real-time morphological image processing," *J. VLSI Signal Process.*, vol. 17, pp. 43–55, 1997.

[12] P. A. Ruetz and R. W. Brodersen, "Architectures and design techniques for real-time image-processing IC's," *IEEE J. Solid-State Circuit*, vol. SC-22, no. 2, pp. 233–250, Apr. 1987.

[13] 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 Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 39, no. 10, pp. 753–756, Oct. 1992.

[14] T. Ikenaga and T. Ogura, "A fully parallel 1-Mb CAM LSI for real-time pixel-parallel image processing," *IEEE J. Solid-State Circuits*, vol. 35, no. 4, pp. 536–544, Apr. 2000.

[15] ——, "Real-time morphology processing using highly parallel 2-D cellular automata CAM^2," *IEEE Trans. Image Process.*, vol. 9, no. 12, pp. 2018–2026, Dec. 2000.

[16] D. Wang and D.-C. He, "A fast implementation of 1-D grayscale morphological filters," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 41, no. 9, pp. 634–636, Sep. 1994.

[17] R. W.-K. Lam and C.-K. Li, "A fast algorithm to morphological operations with flat structuring element," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 45, no. 3, pp. 387–391, Mar. 1998.

[18] J. Gil and M. Werman, "Computing 2-D min, median, and max filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 5, pp. 504–507, May 1993.

[19] D. Z. Gevorkian, J. T. Astola, and S. M. Atourian, "Improving Gil-Werman algorithm for running min and max filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 5, pp. 526–529, May 1997.

[20] I. Pitas, "Fast glorithms for running ordering and max/min calculation," *IEEE Trans. Circuits Syst.*, vol. 36, no. 6, pp. 795–804, Jun. 1989.

[21] D. Coltuc and I. Pitas, "On fast running max-min filter," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 44, no. 8, pp. 660–663, Aug. 1997.

[22] S. Ong and M. H. Sunwoo, "A new cost-effective morphological filter chip," in *Proc. IEEE Workshop Design Signal Processing Systems*, 1997, pp. 421–430.

[23] D. Coltuc and I. Pitas, "Fast computation of a class of running filters," *IEEE Trans. Signal Process.*, vol. 46, no. 4, pp. 549–553, Mar. 1998.

[24] R. M. Haralick and S. R. Sternberg, "Image analysis using mathematical morphology," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 4, pp. 532–550, Jul. 1987.

[25] K. K. Parhi, *VLSI Digital Signal Processing Systems*. New York: Wiley, 1999.

[26] W. K. Pratt, T. J. Cooper, and I. Kabir, "Pseudomedian filter," presented at the SPIE Conf., 1984.

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, Taiwan, R.O.C., in 1999 and 2003, respectively.

During 2003–2004, he was a Research Staff Member 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.

Shyh-Yih Ma received the B.S.E.E, M.S.E.E, and Ph.D. degrees from National Taiwan University , Taipei, Taiwan, R.O.C., in 1992, 1994, and 2001, respectively.

He joined Vivotek, Inc., Taipei, 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.

Liang-Gee Chen (S'84–M'86–SM'94–F'01) received the B.S., M.S., and Ph.D. degrees in electrical engineering from National Cheng Kung University, Tainan, Taiwan, R.O.C., in 1979, 1981, and 1986, respectively.

In 1988, he joined the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan. From 1993 to 1994, he was a Visiting Consultant with the DSP Research Department, AT&T Bell Labs, Murray Hill, NJ. In 1997, he was a Visiting Scholar with the Department of Electrical Engineering, University of Washington, Seattle. Currently, he is a Professor at National Taiwan University. His current research interests are DSP architecture design, video processor design, and video coding systems.

Dr. Chen has served as an Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY since 1996, as Associate Editor of the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS since 1999, and as Associate Editor of IEEE TRANSACTIONS CIRCUITS AND SYSTEMS II: EXPRESS BRIEFS since 2000. He has been the Associate Editor of the *Journal of Circuits, Systems, and Signal Processing* since 1999, and a Guest Editor for the *Journal of Video Signal Processing Systems*. He is also the Associate Editor of the PROCEEDINGS OF THE IEEE. He was the General Chairman of the Seventh VLSI Design/CAD Symposium in 1995 and of the 1999 IEEE Workshop on Signal Processing Systems: Design and Implementation. He is the Past-Chair of Taipei Chapter of IEEE Circuits and Systems (CAS) Society and is a member of the IEEE CAS Technical Committee of VLSI Systems and Applications, the Technical Committee of Visual Signal Processing and Communications, and the IEEE Signal Processing Technical Committee of Design and Implementation of SP Systems. He is the Chair-Elect of the IEEE CAS Technical Committee on Multimedia Systems and Applications. From 2001 to 2002, he served as a Distinguished Lecturer of the IEEE CAS Society. He received the Best Paper Award from the R.O.C. Computer Society in 1990 and 1994. From 1991 to 1999, he annually received Long-Term (Acer) Paper Awards. In 1992, he received the Best Paper Award of the 1992 Asia-Pacific Conference on circuits and systems in the VLSI design track. In 1993, he received the Annual Paper Award of the Chinese Engineer Society. In 1996 and 2000, he received the Outstanding Research Award from the National Science Council and, in 2000, the Dragon Excellence Award from Acer. He is a member of Phi Tan Phi.