A PARTIAL-RESULT-REUSE ARCHITECTURE AND ITS DESIGN TECHNIQUE FOR MORPHOLOGICAL OPERATIONS

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

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

ABSTRACT

This paper proposes a new cost-effective architecture for mathematical morphology named Partial-Result-Reuse (PRR) architecture. For a lot of real-time applications of mathematical morphology, the hardware implementation is necessary; however, the hardware cost of almost existing morphology architectures is too high when dealing with large structuring elements. With partial-result-reuse concept and self-affinity property of general structuring elements, the proposed architecture is more cost-effective and more general than existing morphology architectures. It can deal with morphological operations with arbitrary structuring elements and can be used for other semi-group operations, and only $2\lceil \log_2 n \rceil$ comparators are needed for nxn structuring elements. Simulation shows this architecture can dramatically reduce hardware cost of morphological operations with all kinds of structuring elements.

1. INTRODUCTION

Mathematical morphology[1], which is based on set theory, is very important in the field of digital image processing and computer vision. It contains a lot of useful tools for shape-based image processing and can be used for image analysis, image compression, error correction, and video segmentation[2], which is a key pre-processing of MPEG-4 coding systems. For real-time applications, high-speed morphological operations are urgently required and hardware implementation is necessary.

Many architectures for morphological operations have been proposed [3][4][5][6]. In these architectures, however, the hardware cost becomes enormous when encountering large structuring elements. Consequently, a more cost-effective architecture should be developed.

To avoid redundant computation, the partial results generated during the calculation process should be kept and reused. The Gil-Werman algorithm [7] and its improved version[8] can reduce the complexity to nearly constant; 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 concept are also proposed. Pitas's architecture [9] and Ong's[10] are two of them, but their architectures are not optimized, and the hardware can be further reduced. Coltuc and Pitas [11] show an optimal solution for morphological operations with rectangular structuring elements. Morphological operations, however, often use various kinds of structuring elements, such as circle and disk. Therefore, a more general partial-result-reuse architecture should be develped.

In this paper, we propose a new architecture named Partial-Result-Reuse (PRR) architecture. The PRR architecture has several features. First, with graphic method, the PRR architecture is very easy to design. 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 shape structure elements, such as disk, without large overhead. Finally, this architecture can be used for not only morphological operations but also other running semi-group operations[7] (or T operations[11]).

In Section 2, basic operations of mathematical morphology are introduced. Then the proposed PRR architecture is presented in Section 3. Section 4 compares the PRR architecture to other existing architectures. Finally, Section 5 gives a conclusion.

2. MORPHOLOGICAL OPERATIONS

There are a lot of operations in mathematical morphology, such as dilation, erosion, opening, closing, hit-and-miss, thinning, and thickening[1]. All of these operations are neighborhood operations, the operation result of each point only depends on the points in its neighborhood. The operands of morphological operations include two parts: input signal f, which is usually an image; structuring element B, which records the range and shape of neighborhood region. Almost all morphological operations are combinations of two basic operations: dilation and erosion.

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

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

Usually, the dilation operation is simplified to:

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

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

$$f \ominus B(x,y) = \min\{f(x+i,y+j) \mid (i,j) \in \beta, \text{and}(x-i,y-j) \in \phi\} \quad (3)$$

* This work is supported by SiS Education Foundation.

0-7803-7041-4/01/$10.00 ©2001 IEEE

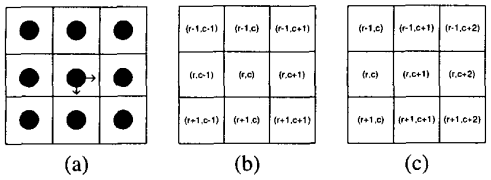

Fig. 1. (a) 3x3 structuring element; (b) pixels needed when computing $I \oplus B(r,c)$; (c) pixels needed when computing $I \oplus B(r,c+1)$.

3. PARTIAL-RESULT-REUSE ARCHITECTURE

The morphological operations are very regular and very suitable for hardware implementation. Many of existing architectures make use of data-reuse technique to reduce memory access amount of these operations. However, a lot of calculation in these architectures is redundant, which means the hardware can be further reduced. The concept can be shown as following example:

When dilating with 3x3 structuring element, which is shown in Fig. 1(a). The dilation result of point (r,c) is the maximum of the nine points in Fig. 1(b), and the result of adjacent point $(r,c+1)$ 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 $\max\{I(r-1,c), I(r,c), I(r+1,c), I(r-1,c+1), I(r,c+1), I(r+1,c+1)\}$ can be propagated, Only three comparators instead of eight comparators are needed.

The partial-result-reuse concept can be used for running max operations and other running semi-group operations (or T operations) [11]. This kind of operations, such as max, min, +, and ×, has several important properties[11]:

(1) Associativity: $(xTy)Tz = xT(yTz)$.

(2) Commutativity: $xTy = yTx$.

(3) Idempotence: $xTx = x$.

where the T denotes one semi-group operation.

Based on the partial-result-reuse concept and these three properties, the PRR architecture is proposed. First, a PRR architecture for structuring elements with self-affinity property is shown. After that, a PRR architecture for other kinds of structuring elements is proposed.

3.1. structuring element with self-affinity property

A lot of usual structuring elements have self-affinity property, which is, a structuring element can be generated by duplicating a small basic element several times, and a structuring element can also be duplicated several times to generate other larger affined structuring elements. This property can be used to find the optimal ways for partial-result-reuse and reduce the computation and hardware of morphological operations.

The procedure is easier to be described by an example. It contains two steps: the self-affine step is to find the way to reuse partial results; the architecture design step is to use the results of first step to design hardware architecture. For simplification, only implementation of dilation is described in this paper, and erosion can be implemented with the same design technique. Dilation with 1x8 structuring element is considered, which is shown as Fig. 2(a). The basic element is a square. We start from current point, which is denote by A in Fig. 2(b). Duplicate the square A and shift left one point to B, which is shown in Fig. 2(c), a larger rectangle AB consists of A and B is formed in Fig. 2(d). With the same procedure,

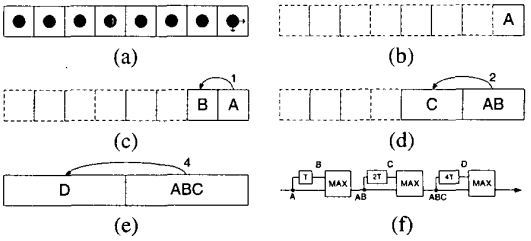

Fig. 2. PRR architecture for 1x8 structuring element.

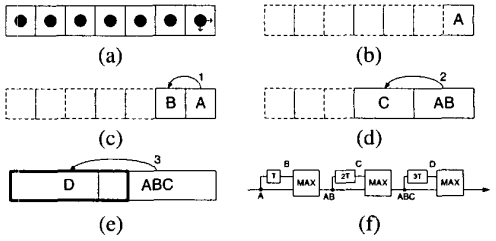

Fig. 3. PRR architecture for 1x7 structuring element.

duplicating AB and shift it two points to C, a 1x4 rectangle ABC is generated in Fig. 2(e), and the whole structuring element can be generated by duplicating ABC and shift it four points to D. After that, we can make use of this procedure to compute dilation with less operations as following equations:

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

$$= \max\{A_x, B_x, C_x, D_x\} \quad (5)$$

where

$$A_k = \max\{I(k)\} = I(k);$$
$$B_k = \max\{I(k-1)\} = I(k-1);$$
$$C_k = \max\{I(k-2), I(k-3)\};$$
$$D_k = \max\{I(k-4), I(k-5), I(k-6), I(k-7)\};$$
$$AB_k = \max\{A_k, B_k\};$$
$$ABC_k = \max\{AB_k, C_k\}.$$

Finally, only three comparators are needed to calculate (5), instead of seven comparators to calculate (4) directly. The value of A_x is equal to the value of current point $I(x)$, and the value of B_x is equal to the A_k value when $k = x-1$. Besides, the value of C_x and D_x is equal to the AB_k value and ABC_k value 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 1x8 structuring element is shown in Fig. 2(f). Only three comparators and seven delay elements are needed. It is obvious that the same design technique can deal with operations with all power-of-2 length structuring elements. If the length is not power-of-2, this procedure can be also used. An example with 1x7 structuring element is shown in Fig. 3. Note that the rectangle ABC, which is shown as normal block, and rectangle D, which is shown as bold block, are overlapped as shown in Fig. 3(e).

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 8x8 as shown in Fig. 4(a), the self-affine procedure is shown in Fig. 4(b)(c)(d)(e)(f)(g)(h). In Fig. 4(c)(e)(g), W denotes the width of image.

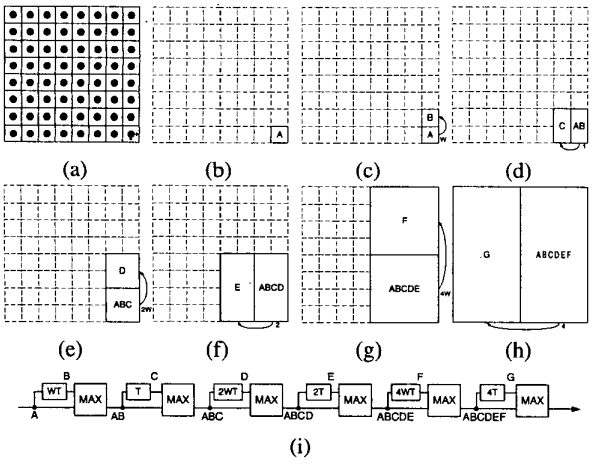

Fig. 4. PRR architecture for 8x8 structuring element.

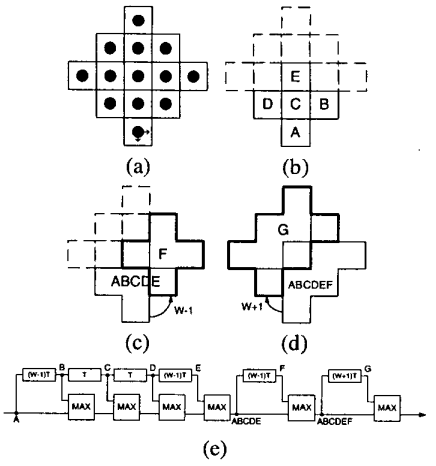

Fig. 5. PRR architecture for 5 diameter disk structuring element.

If the data of a point needs to be propagated vertically to next row, W delay elements is required since the data is 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. The associate architecture for 8x8 structuring element is presented in Fig. 4(i). Only six comparators and $7W + 7$ delay elements are needed. The number of required comparators for *nxn* structuring element is

$$C(n) = 2\lceil \log_2 n \rceil, \tag{6}$$

which is proved as optimal solution in [11].

Besides rectangular structuring elements, other shape of structuring elements, such as disk, can also be handled. Disk-shaped structuring elements are very useful; however, almost all existing architectures with partial-result-reuse concept cannot deal with them. Fortunately, disk has self-affinity property as rectangle. It can be generated by duplicating with a basic element, cross. For example, when the structuring element is a disk with 5 diameter, the self-affine procedure is shown in Fig. 5(b)(c)(d). First, duplicate A to form $ABCDE$, which is in a shape of cross. Then duplicate $ABCDE$ and shift it upper-right to F to form $ABCDEF$. Finally, duplicate $ABCDEF$ 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 comparator and $4W$ delay elements are needed to implement this operation. The number of required comparators for n diameter disk structuring element is

$$C(n) = \begin{cases} 2\lceil \log_2(n-1) \rceil + 2 & \text{if n=odd} \\ 2\lceil \log_2 n \rceil + 2 & \text{if n=even} \end{cases} \tag{7}$$

3.2. arbitrary structuring element

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 applied. Only the redundant operations between adjacent pixels are considered in this technique. The procedure contains three steps. First, the structuring element is divided into several exclusive segments with different delay number consideration. Then combine all the segmentation results. At last, according to the segmentation, the PRR architecture can be designed.

There are three rules for dividing the structuring element into

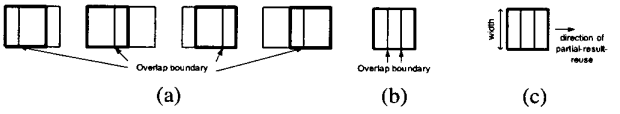

Fig. 6. The rules of PRR architecture for arbitrary structuring elements.

several segments: (1) any segment cannot cross the overlap boundary; (2) the overlap of segments should be avoided; (3) the width of a segment should be maximized until overlap occurs. Here, the overlap boundary is defined as the boundary of data needed by adjacent points. In Fig. 6, 3x3 structuring element is considered. In Fig. 6(a), the data needed for current point is presented by bold blocks, and that for its adjacent points are presented by normal blocks. If only *n-delay* situation and horizontal reusing is considered, which means only partial results generated n cycles before can be reused, each block is n points apart horizontally, and $n = 1$ in this example. The overlap boundary is shown as Fig. 6(b). In addition, the width of segment is defined as length of the border perpendicular to the direction of partial-result-reuse as shown in Fig. 6(c).

It is easier to show this technique by an example. In Fig. 7(a), a structuring element without self-affinity property is shown. In Fig. 7(b)(c), the overlap boundary and segmentation by considering one delay are shown, and those by considering two delay are shown in Fig. 7(d)(e). Note that the bold blocks mean that the required data here can be reused from its adjacent blocks. The combination of Fig. 7(c) and Fig. 7(e) is Fig. 7(f), where the segment D comes from Fig. 7(c), which means it can be reused with only one delay element, and the segment E comes from Fig. 7(e), which means it can be reused with two delay elements. The corresponding PRR architecture is shown in Fig. 7(g). The detail explanation and proof of this design technique will be presented in future publications.

4. COMPARE WITH OTHER ARCHITECTURES

The proposed architecture is compared with other existing architectures as shown in Table 1. Morphological dilation operation with 7x7 structuring element is considered. W and H respec-

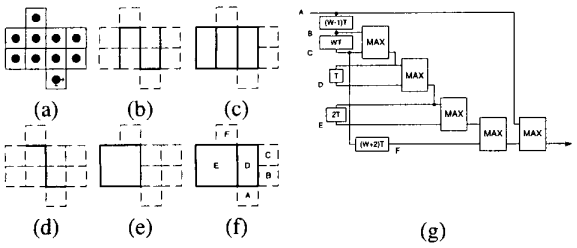

Fig. 7. PRR architecture for arbitrary structuring elements.

Table 1. Comparison between proposed morphology architecture and other's architectures.

Architecture	Comparator count	Delay number	Estimated gate count[a]	Required cycles per frame
Pitas[9]	8	7W+7	448W+840	W(H+7)+6
Coltuc[11]	6	6W+6	384W+678	W(H+6)+5
Ong[10]	7	6W+7	384W+791	7WH
Diamantaras[4][b]	48	6W+42	384W+5040	W(H+6)
Ruetz[5]	12	6W+6	384W+972	W(H+6)+5
Sheu[6]	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

tively denote the width and height of input image. Gate count is estimated with SYNOPSYS^{TM} Design Compiler. Note that, in order to make the comparison be fair, the input data of these architectures is all set in raster-scan manner. The results show that the Coltuc's architecture[11] and our architecture are most efficient in gate count, delay element number, and required clock cycles. However, Coltuc's architecture can only deal with rectangular structuring elements while PRR architecture can deal with arbitrary structuring elements.

Another comparison of comparator number for *nxn* structuring elements is shown in Fig. 8, where the ideal curve is $C(n) = 2\log_2(n)$. The number of comparators required by systolic array architecture[4] increases in square-law and will be very hardware-consuming when dealing with large structuring elements. Ruetz's architecture[5] needs comparators increasing 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, as shown in Fig. 8.

It shows that the PRR architecture is very cost-effective and efficient for morphological operations. It can deal with operations with arbitrary structuring elements.

5. CONCLUSION

A new Partial-Result-Reuse (PRR) architecture for mathematical morphology, which is very cost-effective, is proposed in this paper. Several examples show that the PRR architecture is very easy to design by graphic method. Besides, it can reduce hardware cost of morphological operations with arbitrary structuring elements via partial-result-reuse concept. For structuring elements with self-affinity property, the hardware can be further reduced with proposed design technique. Some simulation has been done to show this architecture is more efficient than others.

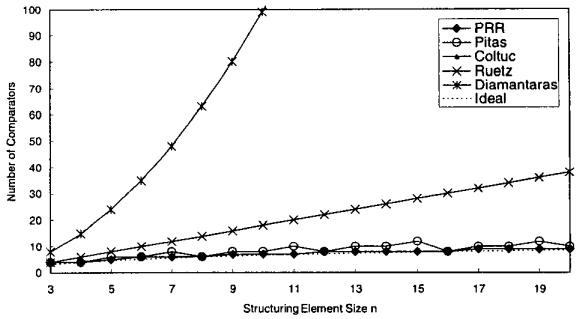

Fig. 8. Comparison of comparators number between difference architectures for *nxn* structuring elements.

6. REFERENCES

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

[2] 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*, pp.1087-1098, 2000.

[3] 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, pp.79-93, 2000.

[4] K.I. Diamantaras and S.Y. Kung, "A linear systolic array for real-time morphological image processing," *Journal of VLSI Signal Processing*, vol. 17, pp.43-57, 1997.

[5] P.A. Ruetz and R.W. Brodersen, "Architectures and Design Techniques for Real-Time Image-Processing IC's," *IEEE Journal of Solid-State Circuit*, vol. sc-22, no. 2, pp.233-250, April, 1987.

[6] 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. on Circuits and Systems–II Analog and Digital Signal Processing*, vol. 39, no. 10, pp.753-756, October, 1992.

[7] J. Cil and M. Werman, "Computing 2-D min, median, and max filters," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pp.504-507, vol. 15, no. 5, May, 1993.

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

[9] I. Pitas, "Fast glorithms for running ordering and max/min calculation," *IEEE Trans. on Cirsuits and Systems*, pp.795-804, vol. 36, no. 6, June, 1989.

[10] S. Ong and M.H. Sunwoo, "A new cost-effective morphological filter chip," *IEEE Worshop on Design Signal Processing Systems 1997 (SiPS 97)*, pp.421-430, 1997.

[11] D. Coltuc and I. Pitas, "Fast computation of a class of running filters," *IEEE Trans. on Signal Processing*, pp.549-553, vol. 46, no. 3, March, 1998.