A VLSI Chip Architecture Design For 2-D Gray-level Morphological Operations

Kun-Min Yang, Petros Maragos* and Lancelot Wu

Bellcore, 445 South Street, MRE 2A260, Morristown, NJ 07960-1910
*Division of Applied Sciences, Harvard University, Cambridge, MA 02138

ABSTRACT

The basic operations of mathematical morphology are quite useful for a broad area of image processing and analysis tasks. All morphological operations can be built from erosions and dilations. In this paper, we develop a single chip VLSI architecture of an erosion/dilation algorithm for real-time image processing. The new architecture allows sequential inputs and performs parallel processing with 100 percent efficiency. The erosions (min of differences) and dilations (max of additions) operate on 2-D gray-level image signals and use a 5×5-pixel gray-level structuring element. Two chips can be cascaded for a 7×7 structuring element. The overall chip design also provides the user with sufficient flexibility to optionally offset the output images by adding a constant level and/or scale their dynamic range.

1. INTRODUCTION

Mathematical morphology [11, 12] is a powerful set-theoretic approach to image processing/analysis. It complements and enriches classical image processing based on linear filtering and Fourier analysis. Some of the new operations and concepts that morphological image processing offers include nonlinear smoothing, skeletonization, shape-size distributions, and general capabilities for extracting information about geometrical structures in images. In addition, cellular logic image processing [9] can be formalized and further extended by using mathematical morphology. The current applications of morphological operations include image enhancement, coding, feature extraction, edge detection, shape analysis and object detection. Their definitions and detailed properties can be found in [11, 13, 3, 7].

Morphological operations are very modular. If the two basic operations of erosion and dilation are implemented, the rest can be formed as a series or parallel (using pointwise min/max) combination of individual erosion/dilations. A very large class of image processing systems can be represented only from erosions or/and dilations, as theoretically shown in [6]. Further, the erosions and dilations are among the prototype operations needed to construct image algebras (e.g., see [10, 1, 4]), which are large algebraic systems designed to express arbitrary image operations as a combination of a few elementary operations. So far the few hardware VLSI implementations of morphological erosions/dilations use very simple, i.e., flat (binary) structuring elements, in which case the implemented erosions/dilations are simply the moving local min/max operations investigated in [8]. Some of these VLSI designs are only for 1-D signals; e.g., see [2] for a VLSI design of 1-D rank-order filters, which contain as special cases the erosion and dilation by a binary structuring element. The only exception is the Cytocomputers [5] that use gray-level structuring elements, in which case the implemented erosions and dilations are, respectively, minima of differences and maxima of additions [13]. However, they restrict the basic neighborhood of the structuring element to be only 3×3 pixels. This choice is dictated by several reasons, one of which is its historic relation to the 3×3 neighborhoods of cellular automata (game "Life", etc.). Another reason is the ability to decompose a large composite structuring element B into a dilation of, say, N simple elements $B_1, \ldots, B_n$ which have either a 2-pixel or a 3×3-pixel support. For example the gray-level structuring element A which approximates the upper part of a discrete sphere of Figure 1 is the dilation of two others: B (a rhombododecahedron) and C. (See Section 2 for definition of dilation.) B and C in turn are the dilation of four doublets (two-point structuring elements), which are shown in Figure 1. Then a single erosion (dilation) of an image by A is equivalent to 8 consecutive erosions (dilations) of the image by $A_1, A_2, \ldots, A_8$. This structuring element decomposition into 3×3 elements is obviously advantageous for software implementations, but its usefulness for hardware implementations is doubtful for several reasons: 1) The cycle time of the morphological chip for a single erosion/dilation by A will be 8 times (or N times, in general, for decomposition into N simpler elements) the cycle time for one 3×3 operation. 2) Cytocomputers have a pipeline architecture and process pixels in a raster scan format, therefore need to buffer $2 \times N + 3$ intermediate results for a 3×3 structuring element, where N is the number of pixels in one raster scan line. The hardware for implementing the delay lines can occupy most of the chip area, thus it is costly. 3) The initial delay for producing the first result is very long, i.e. at least $2 \times N + 3$ cycles. 4) The number of pixels in a raster scan line depends on the image format, which needs to be adjusted for a variety of applications. Therefore the delay lines in the chip must be programmable for adjusting its length and large enough to accommodate the most demanding application, e.g. about 2000 pixels per line for HDTV. But it would be a huge waste for others which require much smaller delay lines. 5) There are many applications of morphological image processing such as background normalization through the rolling ball transform, noise suppression, feature detection, size distributions, etc., that require a gray-level structuring element with a neighborhood larger than 3×3 pixels. But it is very difficult for the architecture used by Cytpcomputers to implement a structuring element larger than 5×5-pixels, since the 4 delay lines required can be too large to fit into a single chip. 6) A larger neighborhood for the basic operations allows the design of a much larger variety of structuring elements. For all the above reasons, there is a need for a morphological chip with a larger, e.g. 5×5 neighborhood. This will provide a larger variety in the surfaces of gray-level structuring elements and will reduce the time required for executing a single erosion/dilation by a large structuring element.

In this paper, we develop an architecture design for a single VLSI chip that performs erosions (min of differences) or dilations (max of additions), operates on 2-D 8 bit/pixel gray-level images, and uses a 5×5-pixels 8 bit/pixel gray-level structuring element. By cascading two chips, a 7×7-pixel gray-level structuring element is allowed. The innovation of our architecture is the special data flow design which realizes the operations on an image in a 2-D manner directly without buffering the intermediate results and thus eliminates the need for delay lines in the chip. The special data flow design allows sequential inputs but performs parallel processing with 100 % efficiency. Sequential input is essential since the number of pins available to a single VLSI chip is limited. Multiple processing elements (PE's) performing parallel processing are required for real time application, but because of the data flow design the PE's can be busy all the time (performing useful operations every cycle). The inputs are efficiently utilized by multiple PE's simultaneously, thus the I/O bandwidth is relieved.

2. IMPLEMENTED OPERATIONS AND SPECIFICATION

Let $f(m,n)$ be an input image signal and let $g(m,n)$ be a structuring function-element. In our design the image function has a $N \times N$-pixel domain and a 8 bit/pixel amplitude range [0,255]. The structuring function g has a 5×5-pixel domain centered at the origin and an 8 bit/pixel amplitude range which is the interval [-128,127]. The two fundamental morphological operations implemented on this chip are erosion and dilation, defined as follows. The *dilation* of f by g is defined by

$$(f \oplus g)(m,n) = \max_{i=-2}^{2} \max_{j=-2}^{2} \{f(m-i, n-j) + g(i,j)\} \quad (1)$$

and the *erosion* of f by g is defined by

$$(f \ominus g)(m,n) = \min_{i=-2}^{2} \min_{j=-2}^{2} \{f(m+i, n+j) - g(i,j)\} \tag{2}$$

Thus, the dilation (a max of additions) has the same computational structure as a convolution, whereas erosion (a min of differences) has the same computational structure as a correlation; note that for dilation, g is reflected around the origin whereas for erosion it is not. Both operations, however, are simpler to implement than linear convolutions because they do not require multiplications.

Remarks:

- *Range:* Note that since g is gray-level, the result of a dilation or erosion of f by g may yield an output image whose output range exceeds the input range [0,255]. To cover these cases our architecture allows 10 bits per pixel, so that the output image may have a range which is any subinterval of [-512,511].

- *Cascadability:* There are 25 PEs in the chip for performing operations using a 5×5-pixel gray-level structuring element, and also there are 25 shift register, i.e. DFF_0-DFF_{24} in Figure 2, for holding the structuring element. The architecture design allows cascading two chips to have 49 PEs and DFFs for 7×7-pixel structuring element. Three signals represented by two bits are defined: Cas_0 denotes no cascade, Cas_1 and Cas_2 denote the first and second chip in a cascaded system. Therefore, Mux_2 is used to bypass the last DFF in the second chip in a cascaded chip system. The values of a structuring element are circulated within the 25 or 49 DFFs except for the loading period during which it is loaded from outside. One can also saves the structuring element in a buffer inside the chip and load it into the DFFs during loading period.

- *Offset/Scaling:* If the output image has a range that exceeds the input range [0,255] the user has the option to add to it a dc-offset and/or multiply or divide it by a power of 2, so that the range may be brought back to [0,255]. The offset and/or scaling constants can be preset and saved in registers shown in the right-bottom part of Figure 2. If the output levels are to be adjusted, Mux_5 is signaled to select the right-hand side inputs as results for which the outputs are added by the offset constant and multiplied by the scaling constant. $Latch_2$ can perform shift-right or shift-left according to the scaling constant in the register, thus results in a power of 2 scaling. Of course, if the output image is only an intermediate result in a sequence of morphological operations, such a rescaling is not needed. Meanwhile in a cascaded configuration, only the outputs in the last chip will be adjusted. Therefore the Mux_5 selection signal becomes $(Offset\_Scaling) \times (CAS\_0 + CAS\_2)$.

- *Erosion vs. Dilation:* Even though definitions of erosion and dilation are different, the new architecture implements them on the same hardware and thus increases the utilization potential of the chip. This is accomplished by having each PE performing max (respectively, min) of additions for dilation (respectively, erosion). In the case of erosion, an input flag, i.e. $Dil.\_Ero.$, set by the user instructs the chip to first *negate* the structuring element [i.e., convert $g(i,j)$ to $-g(i,j)$] and then perform a min of additions. Since this negating operations are done only in the first chip in a cascaded chip system and it is done only during the loading period of structuring element, therefore the signal $Load \times Cas\_1 \times (Dil.\_Ero.)$ is used to instruct the complementary circuitry to convert $g(i,j)$ into $-g(i,j)$. Other than that the structuring element will bypass the complementary circuitry and its original value remains. The PE has a pipeline architecture; there is an adder in the front stage which performs $f+g$ for dilation or $f+(-g)$ for erosion. The second stage of PE does comparison and replacement, where the current max (or min) is saved in $Latch_1$. Both Mux_3 and Mux_4 select upper signals for dilation and they select lower signals for erosion. For dilation, $max(t)-(f+g)(t)$ is performed. If $(f+g)(t) > max(t)$, then the overflow signals $Latch_1$ for replacement, i.e. $max(t+1)=(f+g)(t)$, otherwise $max(t+1)=max(t)$. For erosion, $(f+(-g))(t)-min(t)$ is performed. If $(f+(-g))(t) < min(t)$, the the overflow signals $Latch_1$ for replacement, i.e.$min(t+1)=(f+(-g))(t)$, otherwise $min(t+1)=min(t)$.

- *Minus Infinity:* To perform morphological operations by gray-level structuring elements g, we must set the pixels that do not belong to the domain of g as equal to $-\infty$. We accomplish this in our architecture by using a value outside the designed range [-128,127] for the pixels inside the 5×5 window at which g is equal to $-\infty$. When the PEs detect this value they do not operate on this pixel, i.e., the replacement operation is disabled.

Once the values of a 5×5 structuring element are buffered in DFFs, they are compared against a corresponding pixel in a 5×5 window of a $N \times N$ image data and the minimum or maximum of those results, depending on whether erosion or dilation is performed, replaces the center pixel of working window. The same sequence of operations goes on until all of the $N \times N$ image data are replaced by the values of the erosion or dilation.

Using a 5×5-pixel structuring element, at least 25 arithmetic operations are required for producing one output pixel value. For a real time processing of 30 or more image frames per second with a size of 512 by 512 pixels per frame, the conventional algorithm demands a large number of operations. The 10-bit adder/subtractor must perform an addition in less than 5 *ns* if this is done sequentially, and the same high speed is required for memory access.

Our architecture accesses data sequentially from frame storage memories by dual I/O port but performs the operations in parallel, hence not only relieving the I/O traffic and reducing the number of pins, but also resulting in an efficient data flow. For the case of 30 frames per second, a cycle time may last as long as 125 *ns*. Even a 60 *ns* cycle time will be sufficient up to a scanning rate of 60 frames per second.

3. DATA FLOW

As mentioned in the section I, our novel architecture is based on a special data flow design which realizes sequential inputs but performs parallel processing with 100 % efficiency. In this section, the data flow design will be explained in details. In our design, only one data, $g(i,j)$, of the structuring element, is buffered between PE's and it is piped through the delay elements DFF_1-DFF_{24} as shown in Figure 2. The timing diagram of the basic data flow is depicted in Figure 3. According to the data flow design, an address generator circuit is implemented to generate the addresses for fetching image data from a frame memory. The addresses are composed of a base addresses and an indexing sequence [14]. The decomposition of addresses makes it easy to implement the address generator and easy to generate the necessary control signals.

The data flow can be explained by examining Figure 2, 3 and 4. The first column in Figure 3 denotes the cycle instant t, the second column contains the inputs during each cycle, the third column represents the calculations carried out in PE_0 in Figure 2, and the fourth through the last column represent the calculations done in PE_1 through PE_{24}. Each PE is basically built with an adder and a comparator. At every cycle, except for the initial 25 cycles, one pixel is input from the structuring element, namely, $g(i,j)$ and two pixels are input from the current image data frame, namely, $f(l,k)$ & $f(l-1,k+5)$. During the valid period of these data, the associated operations listed in Figure 3 start and pipe through a series of sub-operations in each PE.

The detailed explanations of Figure 3 for the case of Dilation operation is as follows:

At cycle 0, $g(0,0)$ & $f(0,0)$ are input, $f(0,0)+g(0,0)$ is computed in PE_0 and $g(0,0)$ is latched at DFF_1. The result of $f(0,0)+g(0,0)$ is compared to the previous local maximum if there is any, and a new local maximum is then saved.

At cycle 1, $g(0,1)$ & $f(0,1)$ are input, $f(0,1)$ is broadcast to PE_0 & PE_1, $g(0,0)$ is piped to PE_1, then $f(0,1)+g(0,1)$ is computed in PE_0, and $f(0,1)+g(0,0)$ in PE_1. $g(0,0)$ is latched at DFF_2 and $g(0,1)$ at DFF_1. $f(0,1)+g(0,1)$ is compared to PE_0's previous maximum and $f(0,1)+g(0,0)$ to PE_1's previous maximum. Local maxima are saved for further comparison.

Similar operations as above are repeated for cycle 2, 3 and 4; $g(i,j)$'s are piped through the DFF's and available to PE's at the right timing, and $f(i,j)$'s are broadcast to PE's.

At cycle 5, however, $g(1,0)$, $f(1,0)$ & $f(0,5)$ are input, $f(1,0)$ is broadcast to PE_0 & PE_5, and $f(0,5)$ to PE_1-PE_4. $f(1,0)+g(1,0)$ is computed in PE_0, $f(0,5)+g(0,4)$ in PE_1,..., $f(0,5)+g(0,1)$ in PE_4, and $f(1,0)+g(0,0)$ in PE_5. Comparison with the previous maximum is performed in each PE and its local maximum is saved.

Similar operations repeat through inputting the 25 values of the structuring element. Right after that, the maximum value for the first 25 pixels is buffered into a common tri-state bus and ready for output. One new maximum value for a new working window is buffered every cycle after the initialization.

Since there are two inputs from image data frame, namely, $f(i,j)$ and $f(i-1,j+5)$, the select signals as shown in Figure 4 has to be provided for each multiplexor in PE's. By taking a close look at Figure 4, one can notice that the same pattern of signals, ie 1111 0111 0011 0001 0000, repeats itself every 5 cycles and that the select signal for PE_0, PE_5, PE_{10}, PE_{15}, and PE_{20} can be hardwired to the ground. Thus, as depicted in Figure 5, a 4-bit shifter which is triggered by a key signal from an address generator will generate the control signals easily.

The operation for Erosion is similar to the above except for subtraction instead of addition and the minimum instead of the maximum. If one examines the descriptions of erosion and dilation carefully, one may realize that for each structuring element, image data input for Erosion and Dilation are symmetric respect to the origin of working window. Thus, another temporary location providing structuring elements in an opposite order will have the same effect as an extra circuitry providing two different image data to the processing elements. Also, if only symmetric structuring elements are to be used, which is most of the cases, one temporary location for structuring elements is enough.

The data flow is realized by the implementation of an address generator in the chip. The address is generated by adding a regular sequence to a base address (I,J), the upper left corner of the working window. The regular sequence runs from 0 to 4 for j-coordinate and i-coordinate increases by one for every 5 increases in j. The mechanism is similar to that described in [14].

4. SUMMARY

Figure 2 shows the schematic diagram of the chip. The chip can be functionally divided into three parts; a temporary memory that holds structuring elements, an address generator that generates the address of image data and control signals, and processing elements that perform arithmetic operations. The essence of this architecture is that all 25 PE's are busy except for the initialization. This makes the chip very efficient and also, using the common buses for data transfer saves a substantial amount of silicon area and makes the routing easier. The chips can be cascaded to support 7×7-pixel structuring element. It is flexible enough for allowing user to offset the outputs by a constant level and/or scale up or down by a factor of power of two.

References

[1] E. R. Dougherty and C. R. Giardina, "Image Algebra-induced operators and induced subalgebras," in *Proc. SPIE 845: Visual Communications and Image Processing II*, 1987.

[2] R. G. Harber, S. C. Bass and G. W. Neudeck, "VLSI Implementation of a Fast Rank Order Filtering Algorithm," in *Proc. IEEE ICASSP-85*, Tampa, FL, Mar. 1985.

[3] R. M. Haralick, S. R. Sternberg, and X. Zhuang, "Image Analysis Using Mathematical Morphology", *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-9, pp.523-550, July 1987.

[4] K. S. Huang, B. K. Jenkins, and A. A. Sawchuk, "Binary Image Algebra and Optical Cellular Logic Processor Design", *Comp. Vision Graph. Image Process.*, 45, pp.295-345, 1989.

[5] R. M. Lougheed, D. L. McCubbrey, and S. R. Sternberg, "Cytocomputers: Architectures for Parallel Image Processing," in *Proc. Workshop Picture Data Descr. Manag.*, Pacific Grove, CA, 1980.

[6] P. Maragos, "A Representation Theory for Morphological Image and Signal Processing", *IEEE Trans. Pattern Anal. Mach. Intellig.*, PAMI-11, June 1989.

[7] P. Maragos and R. W. Schafer, "Morphological Filters - Part I: Their Set-Theoretic Analysis and Relations to Linear Shift-Invariant Filters," *IEEE Trans. Acoust. Speech, Signal Processing*, ASSP-, pp.1153-1169, Aug. 1987.

[8] Y. Nakagawa and A. Rosenfeld, "A Note on the Use of Local Min and Max Operations in Digital Picture Processing", *IEEE Trans. Syst., Man, and Cybern.*, SMC-8, Aug.1978.

[9] K. Preston, Jr., and M. J. B. Duff, *Modern Cellular Automata: Theory and Applications*, NY: Plenum Press, 1984.

[10] G. X. Ritter and J. N. Wilson, "Image Algebra in a Nutshell," in *Proc. 1st ICCV*, London, June 1987, pp.641-645.

[11] J. Serra, *Image Analysis and Mathematical Morphology*, NY: Acad. Press, 1982.

[12] J. Serra, Ed., *Image Analysis and Mathematical Morphology Vol.2: Theoretical Advances*, NY: Acad. Press, 1988.

[13] S. R. Sternberg, "Grayscale Morphology," *Comput. Vision, Graph., Image Proc.* **35**, pp.333-355, 1986.

[14] K.-M. Yang, L. Wu, and A. Fernandez, "A VLSI Architecture Design for Motion detection/Compensation Chip with Full Serach capability", in *Proc. 22nd Annual Conf. Inform. Sci. Syst.*, Princeton Univ., Mar. 1988

$$\mathbf{A} = \begin{array}{ccccccc} & & 0 & 1 & 0 & & \\ & 1 & 2 & 2 & 2 & 1 & \\ 0 & 2 & 3 & 3 & 3 & 2 & 0 \\ 1 & 2 & 3 & \boxed{4} & 3 & 2 & 1 \\ 0 & 2 & 3 & 3 & 3 & 2 & 0 \\ & 1 & 2 & 2 & 2 & 1 & \\ & & 0 & 1 & 0 & & \end{array} = \mathbf{B} \oplus \mathbf{C}$$

$$\mathbf{B} = \begin{array}{ccc} 0 & 1 & 0 \\ 1 & \boxed{2} & 1 \\ 0 & 1 & 0 \end{array} = \overset{\mathbf{A}_1}{\begin{pmatrix} \boxed{1} & 0 \end{pmatrix}} \oplus \overset{\mathbf{A}_2}{\begin{pmatrix} 0 & \boxed{1} \end{pmatrix}} \oplus \overset{\mathbf{A}_3}{\begin{pmatrix} \boxed{-1} \\ 0 \end{pmatrix}} \oplus \overset{\mathbf{A}_4}{\begin{pmatrix} 0 \\ \boxed{-1} \end{pmatrix}}$$

$$\mathbf{C} = \begin{array}{ccccc} & & 0 & & \\ & 1 & * & 1 & \\ 0 & * & \boxed{2} & * & 0 \\ & 1 & * & 1 & \\ & & 0 & & \end{array} = \overset{\mathbf{A}_5}{\begin{pmatrix} 0 & * \\ * & \boxed{1} \end{pmatrix}} \oplus \overset{\mathbf{A}_6}{\begin{pmatrix} * & 0 \\ \boxed{-1} & * \end{pmatrix}} \oplus \overset{\mathbf{A}_7}{\begin{pmatrix} * & \boxed{-1} \\ 0 & * \end{pmatrix}} \oplus \overset{\mathbf{A}_8}{\begin{pmatrix} \boxed{1} & * \\ * & 0 \end{pmatrix}}$$

Figure 1. Decomposition of a 2-D gray-level structuring element into a dilation of similar elements.

(* stands for $-\infty$. $\square$ shows the pixel at the origin (0,0) of the plane. The decompositions of A and B are from Serra [12, p. 196].)

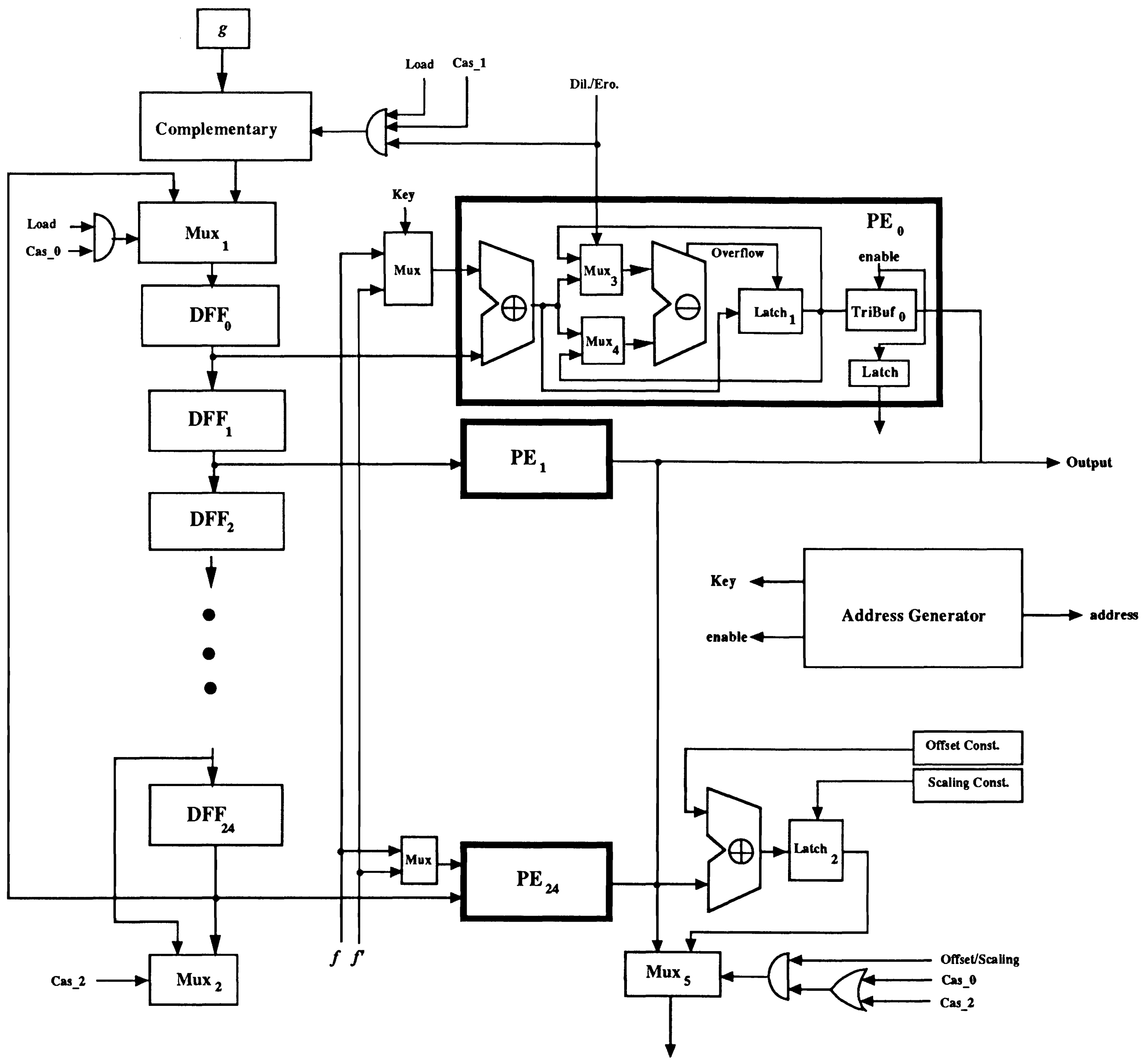
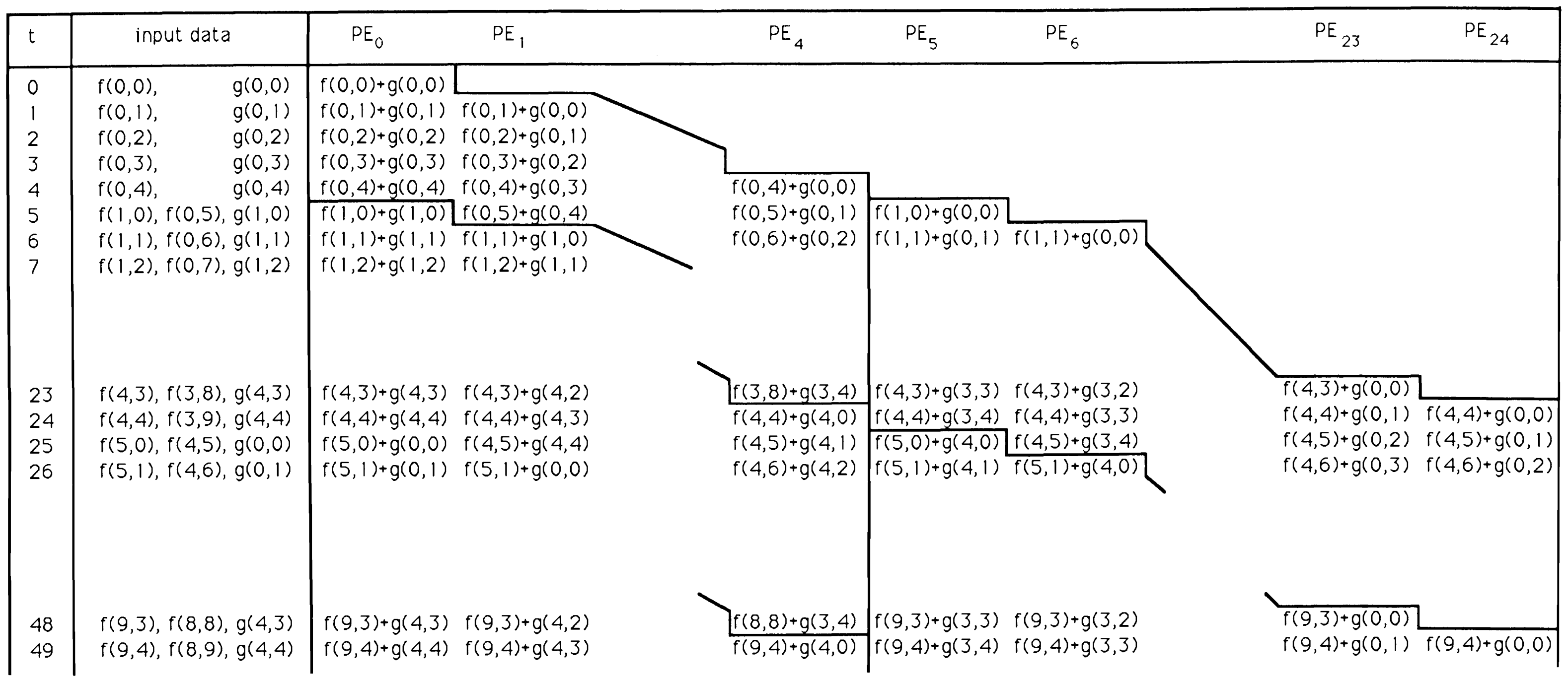


Figure 2. Schematic circuitry for morphological operations.



t	input data	PE_0	PE_1	PE_4	PE_5	PE_6	PE_{23}	PE_{24}
0	f(0,0), g(0,0)	f(0,0)+g(0,0)						
1	f(0,1), g(0,1)	f(0,1)+g(0,1)	f(0,1)+g(0,0)					
2	f(0,2), g(0,2)	f(0,2)+g(0,2)	f(0,2)+g(0,1)					
3	f(0,3), g(0,3)	f(0,3)+g(0,3)	f(0,3)+g(0,2)					
4	f(0,4), g(0,4)	f(0,4)+g(0,4)	f(0,4)+g(0,3)	f(0,4)+g(0,0)				
5	f(1,0), f(0,5), g(1,0)	f(1,0)+g(1,0)	f(0,5)+g(0,4)	f(0,5)+g(0,1)	f(1,0)+g(0,0)			
6	f(1,1), f(0,6), g(1,1)	f(1,1)+g(1,1)	f(1,1)+g(1,0)	f(0,6)+g(0,2)	f(1,1)+g(0,1)	f(1,1)+g(0,0)		
7	f(1,2), f(0,7), g(1,2)	f(1,2)+g(1,2)	f(1,2)+g(1,1)					
23	f(4,3), f(3,8), g(4,3)	f(4,3)+g(4,3)	f(4,3)+g(4,2)	f(3,8)+g(3,4)	f(4,3)+g(3,3)	f(4,3)+g(3,2)	f(4,3)+g(0,0)	
24	f(4,4), f(3,9), g(4,4)	f(4,4)+g(4,4)	f(4,4)+g(4,3)	f(4,4)+g(4,0)	f(4,4)+g(3,4)	f(4,4)+g(3,3)	f(4,4)+g(0,1)	f(4,4)+g(0,0)
25	f(5,0), f(4,5), g(0,0)	f(5,0)+g(0,0)	f(4,5)+g(4,4)	f(4,5)+g(4,1)	f(5,0)+g(4,0)	f(4,5)+g(3,4)	f(4,5)+g(0,2)	f(4,5)+g(0,1)
26	f(5,1), f(4,6), g(0,1)	f(5,1)+g(0,1)	f(5,1)+g(0,0)	f(4,6)+g(4,2)	f(5,1)+g(4,1)	f(5,1)+g(4,0)	f(4,6)+g(0,3)	f(4,6)+g(0,2)
48	f(9,3), f(8,8), g(4,3)	f(9,3)+g(4,3)	f(9,3)+g(4,2)	f(8,8)+g(3,4)	f(9,3)+g(3,3)	f(9,3)+g(3,2)	f(9,3)+g(0,0)	
49	f(9,4), f(8,9), g(4,4)	f(9,4)+g(4,4)	f(9,4)+g(4,3)	f(9,4)+g(4,0)	f(9,4)+g(3,4)	f(9,4)+g(3,3)	f(9,4)+g(0,1)	f(9,4)+g(0,0)

Figure 3. Data flow timing diagram.

	control signal for multiplexors (PE_0- PE_{24})																								
t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	0																								
1	0	0																							
2	0	0	0																						
3	0	0	0	0																					
4	0	0	0	0	0																				
5	0	1	1	1	1	0																			
6	0	0	1	1	1	0	0																		
7	0	0	0	1	1	0	0	0																	
8	0	0	0	0	1	0	0	0	0																
9	0	0	0	0	0	0	0	0	0	0															
10	0	1	1	1	1	0	1	1	1	1	0														
11	0	0	1	1	1	0	0	1	1	1	0	0													
12	0	0	0	1	1	0	0	0	1	1	0	0	0												
13	0	0	0	0	1	0	0	0	0	1	0	0	0	0											
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
15	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0									
16	0	0	1	1	1	0	0	1	1	1	0	0	1	1	1	0	0								
17	0	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0							
18	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0						
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
20	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0				
21	0	0	1	1	1	0	0	1	1	1	0	0	1	1	1	0	0	1	1	1	0	0			
22	0	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0		
23	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1

Figure 4. Control signal pattern for selecting an input.