

# A PROGRAMMABLE LOGIC-BASED IMPLEMENTATION OF ULTRA-FAST PARALLEL BINARY IMAGE MORPHOLOGICAL OPERATIONS

Songpol Ongwattanakul, Phaisit Chewputtanagul, David J. Jackson, Kenneth G. Ricks  
Electrical and Computer Engineering  
The University of Alabama  
Tuscaloosa, AL 35487-0286 USA

## Abstract

Binary morphological operations are a building block in many computer vision applications. Several iterative morphological operations are commonly performed for image analysis resulting in a significant computational load on the processing unit, especially in a real-time computer vision system. Custom designed hardware can exploit the natural parallelism exhibited in binary image morphological operations. In this paper we describe a parallel Binary Morphological Unit (pBMU) which can produce two 8-pixel outputs from one of fifteen morphological operations based on a 3x3 mask. Operations include Sobel edge detection, dilation, erosion, Laplacian, and edge thinning. Implementation on an Altera EPM7256SRC208-7 has shown a sustained performance of 1.066 billion output pixels per second at 66.67 MHz. Thus the pBMU can process 1,356 frame-operations on 1024x768 binary images every second. The pBMU architecture and details of implementation are presented.

Keywords: image processing, binary morphological operations, programmable logic.

## 1. INTRODUCTION

There are numerous applications in military, industrial and robotic vision systems that require ultra high speed image processing on the order of 1,000 frames per second. Binary image morphological operations are well suited to a large class of basic image processing applications. These operations include image analysis tasks such as shape recognition, image segmentation, noise reduction, and feature extraction [1]. Applying morphological operators across an entire image space in an iterative fashion usually implies a computationally expensive algorithm. However, custom hardware can be used for ultra-fast implementations of a number of basic binary morphological operations.

This paper is organized as follows. Section 2 presents the binary morphological unit (BMU). Section 3 details a parallel implementation using

multiple binary morphological units. Section 4 presents results and performance, and Section 5 presents conclusions and future research.

## 2. THE BINARY MORPHOLOGICAL UNIT

The binary morphological operations implemented are well-known image processing operations. Detailed information of these binary morphological operations can be found in [2, 3 and 4].

### 2.1 Edge detection and point detection operations

Both edge detection and point detection operations are convolution-based image processing functions [5]. In implementation, the difference between these operations lies in the construction of a 3x3 (typical) spatial mask. Sobel edge detection operation uses one of the 3x3 masks shown in Figure 1 depending on the direction of the interested edge: vertical or horizontal. On the other hand, the point detection or Laplacian operator uses the set of masks as shown in Figure 2.

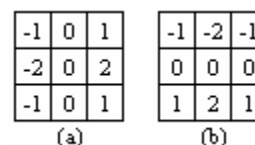


Figure 1. The 3x3 spatial masks for (a) vertical Sobel edge detection and (b) horizontal Sobel edge detection.

After applying the spatial mask to a given pixel of interest, the results are binarized according to a defined threshold. For a binary image, it is common to use a threshold at 0 which implies that the presence of the edge information coincides with a positive output. In some cases, a non-zero output can also be considered an edge. In our efforts, we implement the Sobel edge detection operations in both horizontal and vertical directions. In each direction, we consider two types of binarization on the output: positive-only, and both positive-and-negative.

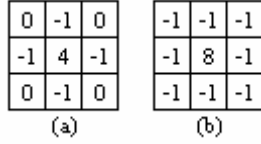


Figure 2. The 3x3 spatial masks for point detection based on (a) 4-connectivity and (b) 8-connectivity.

It is possible to evaluate simultaneously all possible combinations of the 3x3 spatial inputs. A 9-input truth table can be constructed for each edge and point detection operation as shown in Table 1. The location of the 9 inputs,  $I_0$ - $I_8$ , is shown in Figure 3.

Table 1. The 9-variable truth table of the Sobel edge and Laplacian point detection.

$I_8 I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0$	S0	S1	S2	S3	L0	L1
000000000	0	0	0	0	0	0
000000001	1	1	1	1	0	1
000000010	1	0	1	0	1	1
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
111111110	0	0	1	1	0	1
111111111	0	0	0	0	0	0

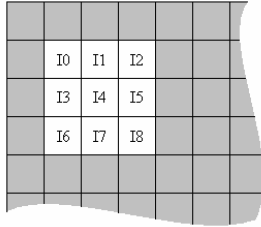


Figure 3. The mapping between the 3x3 sub-image and the 9-inputs,  $I_0$ - $I_8$ .

Table 1 shows a sample of the 9-pixel input and four Sobel edge detection outputs ( $S_0$ - $S_3$ ) and two Laplacian point detection outputs ( $L_0$  and  $L_1$ ). The edge detection outputs are horizontal positive-only, horizontal positive-and-negative, vertical positive-only, and vertical positive-and-negative respectively. The  $L_0$  and  $L_1$  outputs are the Laplacian point detection based on 4-connectivity and 8-connectivity respectively. From Table 1, minimized Boolean expressions for  $S_0$ - $S_3$  and  $L_0$ - $L_1$  as functions of  $I_0$ - $I_8$  can be obtained. The Boolean expressions represent the implementation of the variations of the Sobel and Laplacian operations that are a part of the BMU.

## 2.2 Dilation and erosion operations

Dilation and erosion operations [6] share similar 3x3 spatial masks, but use a different operator in calculation. In the dilation process, if any pixel in a 3x3-pixel sub-image coincides with the 1's pixel of

the mask, the result would be a 1. The erosion process, on the other hand, requires that all the 1's pixels of the mask must be covered by all the 1s in the 3x3-pixel sub-image in order to output a 1. From these realizations, the dilation operation becomes an OR operation over the inputs that are covered by 1s in the 3x3 mask. The erosion operation is a logical AND over the inputs covered by 1s in the 3x3 mask. The dilation and erosion masks are shown in Figure 4.

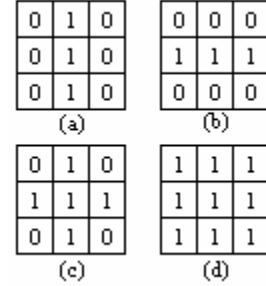


Figure 4. The four 3x3 spatial masks for dilation/erosion operation: (a) vertical, (b) horizontal, (c) 4-connectivity, and (d) 8-connectivity.

The Boolean equations for each mask can be derived as follows:

$$\begin{aligned}
 D_0 &= I_1 \vee I_4 \vee I_7 \\
 D_1 &= I_3 \vee I_4 \vee I_5 \\
 D_2 &= I_1 \vee I_3 \vee I_4 \vee I_5 \vee I_7 \\
 D_3 &= I_0 \vee I_1 \vee I_2 \vee I_3 \vee I_4 \vee I_5 \vee I_6 \vee I_7 \vee I_8 \\
 E_0 &= I_1 \wedge I_4 \wedge I_7 \\
 E_1 &= I_3 \wedge I_4 \wedge I_5 \\
 E_2 &= I_1 \wedge I_3 \wedge I_4 \wedge I_5 \wedge I_7 \\
 E_3 &= I_0 \wedge I_1 \wedge I_2 \wedge I_3 \wedge I_4 \wedge I_5 \wedge I_6 \wedge I_7 \wedge I_8
 \end{aligned}$$

The Boolean equations,  $D_0$ - $D_3$ , are for vertical, horizontal, 4-connectivity and 8-connectivity dilation operations, respectively. Equations for  $E_0$ - $E_3$  are defined similarly.

## 2.3 Edge thinning operations

The edge thinning operation [2] on a binary image is similar to template matching. If one of the eight masks in Figure 5 matches a 3x3-pixel sub-image, the result will be a 1. An \* in the mask represents a don't care pixel.

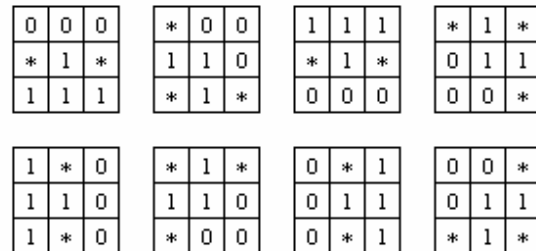


Figure 5. The eight masks of the edge thinning operation from [2].

The truth table for the edge thinning operation is shown in Table 2 where an  $x$  implies a don't care input. A minimized Boolean expression for  $T$  is readily derived from the truth table.

Table 2. The realized truth table for the edge thinning operation.

$I_0 I_1 I_2 I_3 I_4 I_5 I_6 I_7 I_8$	$T$
000x1x111	1
x00110x1x	1
1x01101x0	1
x1x110x00	1
111x1x000	1
x1x01100x	1
0x10110x1	1
00x011x1x	1
...	0

## 2.4 Synthesis of a Binary Morphological Unit

We have implemented a total of fifteen binary morphological operations on a single BMU based on the techniques discussed earlier. Other morphological operations can also be implemented. Figure 6 shows a 1-bit BMU where  $I_0$ - $I_8$  are the 3x3 pixel sub-image inputs arranged as shown in Figure 3. The opcode signal selects the operation function as defined in Table 3.

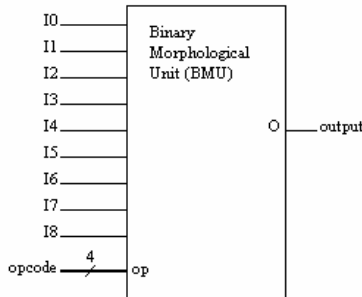


Figure 6. The Binary Morphological Unit (BMU).

Table 3. Opcodes of the BMU.

Opcode	Operation
0000	Horizontal Sobel (+)
0001	Vertical Sobel (+)
0010	Horizontal Sobel (+/-)
0011	Vertical Sobel (+/-)
0100	Horizontal dilation
0101	Vertical dilation
0110	4-connectivity dilation
0111	8-connectivity dilation
1000	Horizontal erosion
1001	Vertical erosion
1010	4-connectivity erosion
1011	8-connectivity erosion
1100	4-connectivity Laplacian
1101	8-connectivity Laplacian
111x	Edge thinning

## 3. THE PARALLEL IMPLEMENTATION

Each BMU requires a 3x3-pixel sub-image ( $I_0$ - $I_8$ ) as inputs to produce a one pixel output. Multi-pixel outputs can be implemented using multiple BMUs. This parallel implementation presents an additional challenge. An  $M \times N$ -pixel output requires an  $(M+2) \times (N+2)$ -pixel input as shown in Figure 7. The 2x8-pixel output requires a 4x10-pixel input. To maintain a constant stream of multiple pixels output per cycle, an efficient scheme for memory access must be addressed.

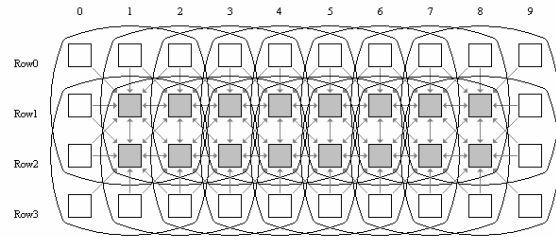


Figure 7. A 2x8-pixel output generated from a 4x10-pixel input.

It is obvious that multiple adjacent rows of an image must be presented at the input ports of the BMU. We design the memory architecture to include four banks. Each bank contains 4-interleaved image rows. These memories are  $2^n$ -bits wide where each bit represents a pixel. In our experiment,  $n$  ranges from 2 to 4. Increased parallelism can be easily accomplished by increasing  $n$ . The choice of  $n$  for common off-the-shelf memory is 3. This means each memory bank is 8 bits wide. We designed a 2x8-pixel output for the parallel BMU (pBMU) which requires a 4x10-pixel input. This is shown in Figure 8.

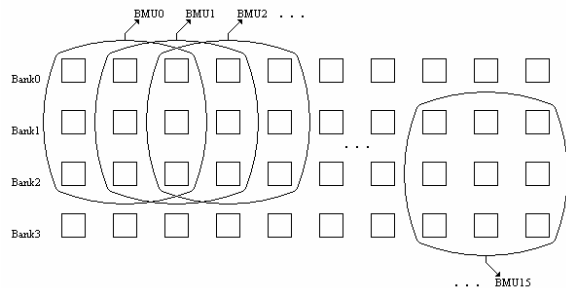


Figure 8. The 4x10-pixel input and its corresponding output.

The 4-bank memory architecture, shown in Figure 9, simplifies the memory addressing scheme. Since 2-row outputs are produced simultaneously, the row address for subsequent outputs is an increment of 2. The increment of the row address by 2 implies that the next image row will come from the next 2

adjacent banks. A larger number of banks will require a more complex row addressing scheme.

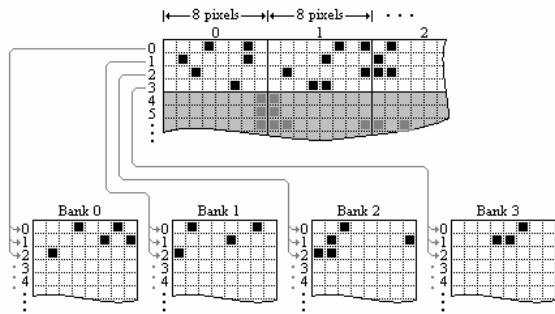


Figure 9. The 4-bank memory architecture and the corresponding pixel arrangement.

Each row of the 4x10-pixel input requires 10 pixels to be presented, but a memory bank only supplies an 8-pixel value. Our solution is to use a 2-pixel extender circuit to preserve the last 2 pixels of the previous input on each row as shown in Figure 10. There are two D Flip-flops inside the 2-pixel extender. The 8-pixel input,  $i_0-i_7$ , passes through to the output  $o_0-o_7$  and combines with the output of the 2-pixel extender,  $o_a$  and  $o_b$ , to form a 10-pixel output. The 2-pixel extender connects directly to the last 2 bits of the 8-pixel input,  $i_6$  and  $i_7$ , which are preserved to be used in the next cycle. The 2-pixel extender is reset at the beginning of the image row. This has the same effect as 2-column zero padding on the left side of the image as shown in Figure 11.

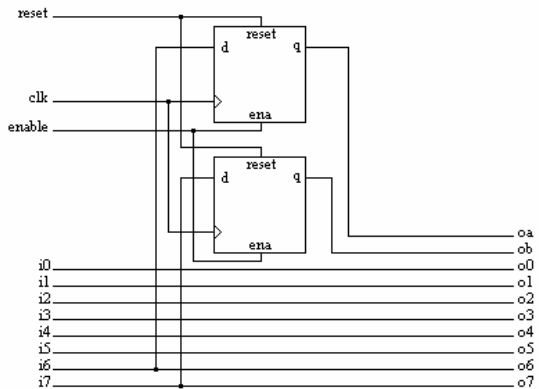


Figure 10. The 2-pixel extender preserving the last two pixels of the previous input.

The 2-column zero padding will cause the output image to be shifted to the right by one pixel as a side-effect. Iterative feedback operations require a 1-bit shifter to prevent further right shifting in the output. Other schemes such as using the mirror

image as an input on every other iteration can also be used to prevent iterative shifting.

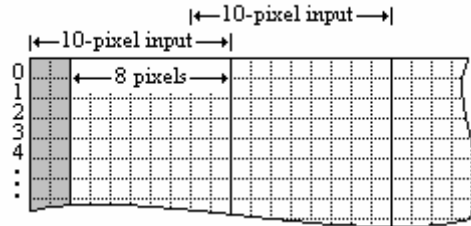


Figure 11. The 2-column zero padding on the left side of the image.

The 4-bank memory controller was designed to ensure a continuous stream of input data to the pBMU. The address controller receives a memory address and converts it to the logical address for each memory bank. The image pixels from the 4-bank memory are routed to the correct row through the multiplexers as shown in Figure 12. The pixel outputs of all rows,  $row_0-row_3$ , are connected to input ports of the pBMU through the 2-pixel extender circuit in Figure 10.

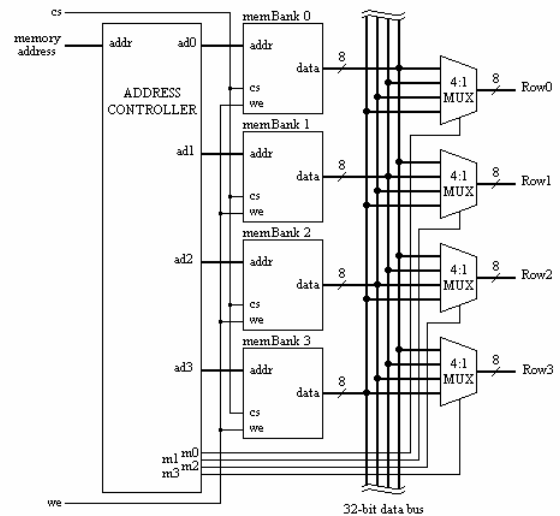


Figure 12. The address controller and the connection between the 4-bank memory and pBMU.

The assembly of all the components inside the pBMU is shown in Figure 13. The  $n$ -bit address selects a group of pixels to be processed according to the function defined in the *opcode*. The *reset* signal is asserted at the beginning of the image row. The two 8-pixel outputs,  $output_0$  and  $output_1$ , are the results of the morphological operation from the 4x10-pixel input stream.

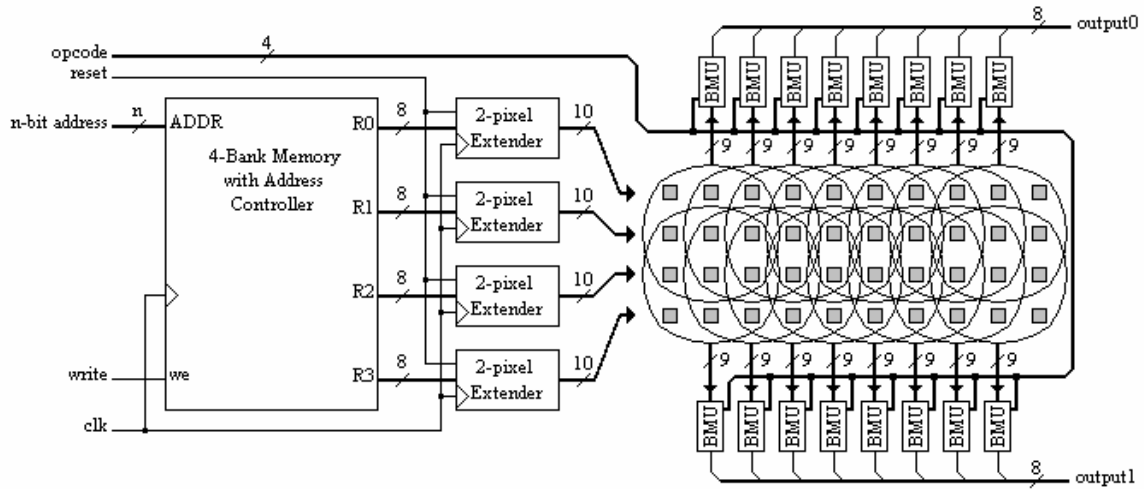


Figure 13. Implementation of the pBMU.



Figure 14. The outputs of 15 morphological operations on Lena image: (a) the original image, (b) – (p) the output images.

## 4. RESULTS

For this effort, we used a 256x256-pixel binary image of Lena as a test image. All the components are written in VHDL and synthesized on various Altera FPGA and CPLD devices using LeonardoSpectrum<sup>tm</sup> from Mentor Graphics. The results of the implemented morphological operations were collected from the timing simulation of the Altera MAX+plus II software. Those results were compared with the simulated results obtained from the same operations performed in MATLAB. The comparison shows an exact match for each operation. Consequently, the correctness of the binary morphological unit is confirmed. The outputs of the 15 implemented morphological operations are shown in Figure 14.

Devices chosen for implementation include Altera MAX7000 and FLEX10K devices. The Altera MAX7000 family of programmable logic devices (PLDs) [7] is based on Altera's second-generation MAX architecture. Fabricated with advanced CMOS technology, the EEPROM-based MAX 7000 family provides 600 to 5,000 usable gates. The EPM7128S device is ideal for large combinatorial and sequential logic functions with a capacity of 2,500 gates and a simple architecture. The Altera FLEX10K devices [8] are the industry's first embedded PLDs. Based on reconfigurable CMOS SRAM elements, the Flexible Logic Element MatriX (FLEX) architecture incorporates all features necessary to implement common gate array megafunctions with up to 250,000 gates. The EPF10K70 device is ideal for intermediate to advanced design including computer architecture, communications, and DSP applications. The EPF10K70 device has over 70,000 typical gates, 3,744 Logic Elements (LEs), and nine Embedded Array Blocks (EABs).

The implementation on an EPM7256SRC208-7 device shows a registered performance of 67.56 MHz which is equivalent to 1.08 billion pixels per second (16-pixel output/cycle  $\times$  67.56 MHz) on the output. On the other hand, the EPF10K70RC240-2 device shows a lower registered performance at 37.31 MHz, which is equivalent to processing 597 million pixels per second. The number of logic cells used on each device was 206 and 1,339 respectively. The frame rates attainable using the aforementioned devices for several basic image sizes are given in Table 4.

Table 4. Attainable frame rates for the pBMU at various image resolutions (WxH).

Device	WxH	Frames/sec
EPM7256SRC208-7 @ 66.67 MHz	320x240	13,890
	640x480	3,472
	1024x768	1,356
EPF10K70RC240-2 @ 36 MHz	320x240	7,500
	640x480	1,875
	1024x768	732

## 5. CONCLUSION

The design and implementation of a parallel binary morphological unit has been presented. Test results and simulation confirm the correctness of the design with the output rate beyond 1 Giga-Pixels/second. At this performance level, this low-cost pBMU outperforms many expensive DSP-based image processing platforms.

For further performance improvement, it is possible to create a custom VLSI implementation of the pBMU. This will allow a larger pBMU, in both the number of concurrent operation units and the number of morphological functions.

## REFERENCES

- [1] M. D. Levine, *Vision in Man and Machine*, McGraw-Hill Book Company, 1985.
- [2] R. C. Gonzalez, R. E. Woods, *Digital Image Processing*, Addison Wesley, Reading Massachusetts, 1992.
- [3] M. Sonka, V. Hlavac, R. Boyle, *Image Processing, Analysis, and Machine Vision*, 2<sup>nd</sup> Edition, PWS Publishing, 1998.
- [4] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, London, 1982.
- [5] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, Academic Press, Boston, 1982.
- [6] C. R. Giardina and E.R. Dougherty, *Morphological Methods in Image and Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1988.
- [7] *MAX 7000 Programmable Logic Device Family Data Sheet*, version 6.02, <http://www.altera.com>, November 2001.
- [8] *FLEX 10K Embedded Programmable Logic Family Data Sheet*, version 4.1, <http://www.altera.com>, March 2001.