A DIGIT-SERIAL ARCHITECTURE FOR GRAY-SCALE MORPHOLOGICAL FILTERING

Lori Lucke
Dept. of Electrical Engineering
University of Minnesota
Minneapolis, MN 55455, USA
lucke@ee.umn.edu

Chaitali Chakrabarti
Dept. of Electrical Engineering
Arizona State University
Tempe, AZ 85287-5706, USA
chaitali@enuxsa.eas.asu.edu

ABSTRACT

We present a digit-serial architecture for gray-scale morphological operations which operates on radix-2 redundant numbers. We present new implementations of a redundant number adder and maximum unit used in the morphological dilation unit. These new designs have areas comparable to 2's complement implementations, but have significantly smaller latencies.

1. INTRODUCTION

Morphological operations are widely used in image processing applications for feature extraction and machine vision applications [1]. The basic morphological operations are dilation, erosion, opening, and closing. Complex operations such as skeletonization and feature extraction utilize these basic operations [2]. Morphological operations can be defined in the binary domain as well as in the gray-scale domain. Morphological operations in the gray-scale domain are significantly more complex than those in the binary domain. The operations in the gray-scale domain require additions and subtractions followed by minimum or maximum operations, instead of the simple logic functions used in the binary domain.

Recently, several application specific VLSI architectures for morphological operations have been proposed. The architectures for binary morphological operations consist of large arrays of binary processors [3]. The architectures for gray-scale morphological operations, on the other hand, consist of smaller arrays of systolic processors which can add/subtract, and compute the maximum/minimum [4, 5, 6]. The systolic architectures directly map the morphological operations onto silicon, and as a result, have large area and large latency (time from input to output). Recently, gray-scale morphological operations which use threshold decomposition to map gray-scale operations onto binary operations, have been proposed in [7, 8]. Such implementations are very fast, but have an area complexity that is proportional to 2^k, where k is the precision or wordlength.

In this paper we present an efficient digit-serial VLSI architecture for gray-scale morphology, which has an area complexity that is proportional to N, the window size, and a latency of $O(1)$. The basic module is a dilation unit, which can be used to implement erosion, opening and closing. The dilation unit consists of adders followed by a maximum computation unit. We choose digit-serial radix-2 redundant arithmetic over bit-serial 2's complement arithmetic, in order to reduce the latency of the dilation unit from $O(k)$ to $O(1)$. Furthermore, the area requirements of the redundant number implementation are comparable to those of the 2's complement implementation. We present new designs for a most significant digit (MSD)-first adder and a MSD-first N-input maximum unit. The proposed architecture has a small area, supports variable wordlengths, and scales linearly with the input size.

2. MORPHOLOGICAL OPERATIONS

The gray-scale morphological operations can be defined as follows [2]. Let $f(x)$ and $g(x)$ be one-dimensional gray-scale functions of coordinate x, where $f(x)$ is the image and $g(x)$ is the structuring element. Let E represent Euclidean space. Then $f : F \rightarrow E$ and $g : G \rightarrow E$. The dilation, $\oplus$, and the erosion, $\ominus$, operations are defined as follows:

$$(f \oplus g)(x) = max\{f(x-z)+g(z)\} \; \forall \, z \in G \; and \; x-z \in F.$$

$$(f \ominus g)(x) = min\{f(x+z)-g(z)\} \; \forall \, z \in G \; and \; x+z \in F.$$

The erosion operation can also be defined in terms of dilation as

$$(f \ominus g)(x) = -((-f) \oplus \hat{g})(x)$$

where $\hat{g}(z) = g(-z)$ is the reflection of g. The opening operation is a dilation followed by an erosion, and the closing operation is an erosion followed by a dilation. Since erosion, opening and closing can all be implemented by the dilation operation, we choose the dilation operation to be our kernel operation. The aim of this paper is to develop an efficient architecture for the dilation unit. The architecture should have (i) small area, (ii) small latency, and (iii) support variable sized word-lengths. We choose a digit-serial mode of operation since it satisfies all these requirements.

This research was supported in part by a grant from NSF, no. MIP-9309504.

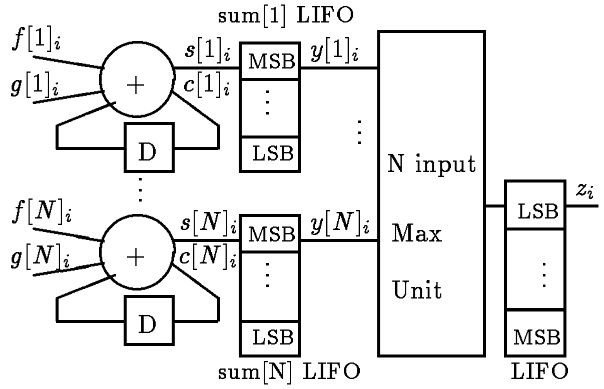


Figure 1: Bit-serial dilation unit for 2's complement binary numbers. The inputs are fed LSB-first. $f[j]_i$ represents the ith bit of $f[j]$.

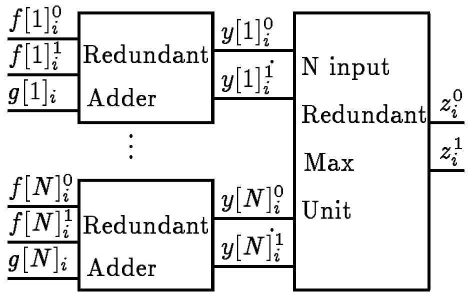


Figure 2: Digit-serial dilation unit for radix-2 redundant numbers. The inputs and outputs are fed MSD-first. $f[j]_i$, $y[j]_i$, z_i, are the ith digits of radix-2 redundant numbers. Each redundant number is encoded using two bits, for example, $f[j]_i^0$ and $f[j]_i^1$. $g[j]_i$ is the ith bit of a 2's complement binary number.

3. 2'S COMP. IMPLEMENTATIONS

The gray-scale dilation unit is described as follows. The N input samples represent the sample window of data or image elements, and are labeled $f[1]$ to $f[N]$, and the structuring elements are labeled $g[1]$ to $g[N]$. The sample window could be one or two-dimensional. In this paper we assume that the data is stored in a buffer, and is accessed appropriately.

There are two possible implementations of a dilation unit using 2's complement inputs. The first requires bit-serial adders and comparators to implement the maximum operation. This scheme has a latency of $O(klog_2N)$. In the second implementation, the input samples and the structuring elements are added using bit-serial adders, but now the outputs of the adders are fed to an N-input maximum computation (MAX) unit which operates on most significant bit (MSB)-first bit-serial data. Figure 1 describes such an implementation. Here the output from each bit-serial adder is placed in a LIFO (Last In First Out) register which holds the results of the add operation until the entire word has been processed. The N-input MAX unit recursively computes the output bits, starting from the MSB. All inputs are initially candidates for the maximum output. The current output bit is the bit with the maximum value among the inputs in the candidate set. In each cycle, the candidacy of an input is updated by comparing the current output bit with the current input bit. Since the output of the MAX unit is generated MSB-first, it requires an additional LIFO to reverse the bit order for input/output compatibility. The latency of this implementation is $O(k)$.

In an attempt to further reduce the latency, Lee, Chen and Jen [9] proposed a scheme consisting of candidate pair selection (which selects the correct $f[j], g[j]$ input pair that contributes to the output), followed by a single addition. Unfortunately, this scheme is erroneous, since both f and g are represented in binary 2's complement form, and thus a MSB-first candidate pair selection cannot be done on inputs prior to adding bits MSB-first.

The latency of the second implementation cannot be reduced to less than $O(k)$ when the inputs are represented in 2's complement form, because 2's complement addition is inherently a LSB-first operation, while computing the maximum is inherently a MSB-first operation. If, however, the numbers are represented in the redundant number system or in the carry save system, both the addition and the maximum can be done Most Significant Digit (MSD) first, and the latency of the dilation unit is no longer a function of k.

4. REDUNDANT NUMBER IMPLEMENTATION

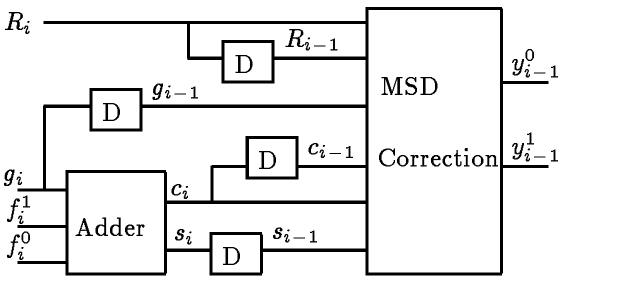
We present a digit-serial implementation where the numbers are represented in the radix-2 redundant system. We chose a radix-2 system over a higher order radix system, since the input formatting is straightforward, and the hardware components have area and latency comparable to binary systems. The proposed architecture consists of (i) N digit-serial adders which add radix-2 redundant numbers to 2's complement numbers, and (ii) a new digit-serial MAX unit which finds the maximum of a set of N radix-2 redundant numbers. The block diagram of the proposed dilation unit is given in Fig. 2. This dilation unit has a latency of $O(1)$ and does not require any LIFO registers.

It is also possible to achieve a similar reduction in the latency and in the number of LIFO registers by operating on carry save numbers. Carry save numbers also allow addition and MAX operations to be performed MSD first. We choose to work with the radix-2 redundant number system because it is easier to perform inversions [1] with the radix-2 redundant numbers.

4.1. REDUNDANT ADDER

The proposed redundant adder is a digit-serial adder which adds a radix-2 redundant number, f, to a bi-

[1] Inversions are required to build erosion, opening and closing units with dilation units (refer to Section 2).



Adder: $c_i = g_i \bar{f}_i^1 + \bar{f}_i^1 f_i^0 + g_i f_i^0$

$$s_i = g_i \bar{f}_i^1 \bar{f}_i^0 + g_i f_i^1 f_i^0 + \bar{g}_i \bar{f}_i^1 f_i^0 + \bar{g}_i f_i^1 \bar{f}_i^0$$

MSD Correction: $y^0_{i-1} = s_{i-1} \otimes (R_{i-1}(c_{i-1} \otimes g_{i-1}))$

$$y^1_{i-1} = (c_i \bar{R}_{i-1}) \otimes (R_{i-1}(c_{i-1} \otimes g_{i-1}))$$

Figure 3: The digit-serial adder. The inputs are f_i, g_i, and R_i. f_i is a radix-2 redundant number, g_i is a 2's complement number, and R_i is a control input. The output, y_i, is a radix-2 redundant number. Note, $\otimes$ is the exor operation.

nary 2's complement number, g, and outputs a radix-2 redundant number, y. We represent the image samples, f, in redundant form to allow us to use the result of one dilation as the input to the next dilation. The structuring element, g, is represented in 2's complement form, as it is never the result of a previous dilation. This simplifies the design of the redundant adder. The proposed adder is an extension of the digit-parallel adder of [10] which adds a k-bit 2's complement number to a k-digit redundant number, and outputs a (k+1)-digit redundant number. We have modified this adder to operate as a digit-serial adder, and also to correct the MSD so that the output can be represented by k digits (instead of k+1 digits as in [10]).

In the proposed adder, the redundant digits belong to the set $\{-1, 0, 1\}$. Each digit is coded with 2 bits. Digit 0 is coded with 00 or 11, digit 1 is coded with 01, and digit -1 is coded with 10. Since the structuring element, g, is in 2's complement form, its digits belong to the set $\{0, 1\}$. In our notation, f_i represents digit i of input f, where $i = 0$ corresponds to the MSD. f^c represents the cth bit that is used to encode a particular digit, $c = 0$ for the MSB, and 1 for the LSB.

Fig. 3 describes the proposed redundant adder. It consists of an adder followed by a MSD correction unit. The inputs are processed digit-serially MSD-first. In the ith cycle, the adder adds f_i and g_i and generates an interim sum, s_i, and carry, c_i. Since we are adding the digits MSD first, the output digit depends on both the current carry bit and the sum bit from the previous cycle. Thus the adder has a latency of 1 cycle. Let R be the control signal that is active high when it accompanies the MSD of the operands (R=1 for $i = 0$, and is 0 otherwise). Let $x_{i-1} = c_i - s_{i-1}$. When $R = 0$, $y_i = x_i$, and when $R = 1$, y_0 is a function of g_0, and c_0. Specifically, if $\bar{g_0} c_0 = 1$, $y_0 = 1$; else if $g_0 \bar{c_0} = 1$, $y_0 = -1$; else $y_0 = x_0$. Overflow occurs if $(g_0 \bar{c_0} + \bar{g_0} c_0)\bar{x_0} = 1$.

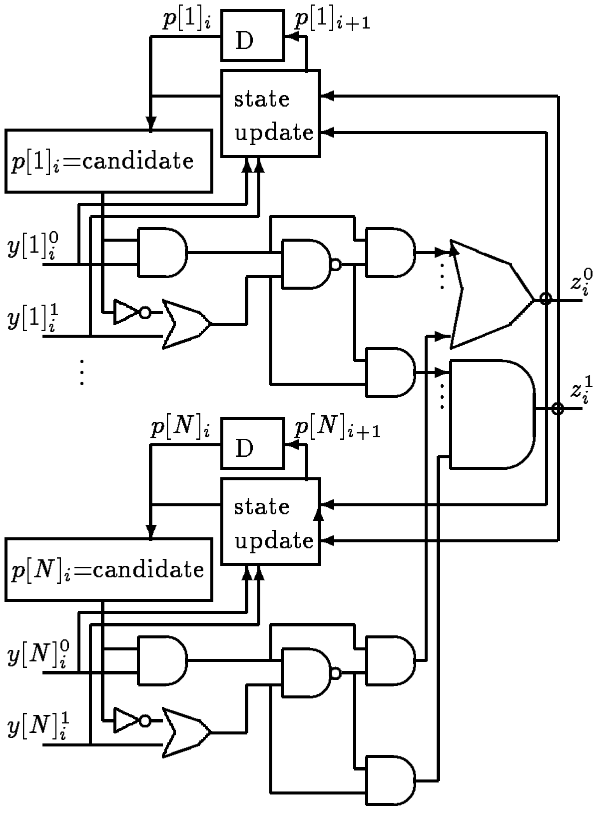


Figure 4: N input MAX unit for radix-2 redundant numbers

4.2. N INPUT REDUNDANT MAX UNIT

The redundant adders output k-digit radix-2 redundant numbers, and thus cannot be fed directly to the MAX unit for 2's complement numbers. One possible way of handling this is to convert the radix-2 redundant numbers into 2's complement numbers and then feed them to the MAX unit. However, such a method is inefficient since it requires an additional $2N$ registers and an additional delay of k for the conversion. Instead, we propose a new MAX unit which operates on a set of N radix-2 redundant numbers.

The N input redundant MAX unit is a digit-serial unit which finds the maximum of N redundant numbers entered MSD-first, and outputs a redundant number which is equivalent to the maximum number. Here each input $y[j]$ has a state variable $p[j]$ associated with it, where $p[j]$ can be one of three states: *candidate*, *undecided*, or *eliminated*. Initially, all the inputs belong to the candidate set. The current output digit, z, is the maximum digit among the inputs in the candidate set. In each cycle, the state of each input is updated by comparing the current input digit to the current output digit. If $p[j]$ = *candidate*, then $y[j]$ is a candidate for the maximum output. If $p[j]$ = *undecided*, then $y[j]$ may become a candidate for the output in the future. If $p[j]$ = *eliminated*, then $y[j]$

input[1]=5 0 1 0 1

input[2]=3 0 1 -1 1

input[3]=-4 -1 1 0 0

input[4]=6 0 1 1 0

p[1]=p[2]=p[3]=p[4]=candidate

i=0: z_0=0

p[3]=undecided; p[1]=p[2]=p[4]=candidate

i=1: z_1=1

p[3]=eliminated; p[1]=p[2]=p[4]=candidate

i=2: z_2=1

p[2]=p[3]=eliminated;p[1]=undecided;p[4]=candidate

i=3: z_3=0

p[2]=p[3]=eliminated;p[1]=undecided;p[4]=candidate

output: z=0110

Figure 5: Example illustrating the MAX operation of radix-2 redundant numbers when n=4 and k=4

can never be the maximum output. The operation of the MAX unit is described by the following algorithm.

Figure 4 describes a MAX unit. The input y, the output z, and the state p are each encoded using 2 bits. Let p^0 and p^1 be the two bits that are used to encode p. We choose the following state assignment: candidate = 01; undecided = 10; and eliminated = 00. Then the state update becomes $p^0_{i+1} = \bar{y_i}^0\bar{y_i}^1(p^0_i z^0_i + p^1_i z^1_i) + \bar{z_i}^0\bar{z_i}^1(p^0_i y^1_i + p^1_i y^0_i)$ and $p^1_{i+1} = y^1_i(p^0_i z^0_i + p^1_i) + p^1_i(z^0_i + \bar{y_i}^0\bar{z_i}^1)$. Figure 5 describes the the operation of the MAX unit with an example.

Algorithm:

Step 1: *for* $j = 1$ to N *do* in parallel $p[j]_0 = candidate$.

Step 2: Repeat steps 3,4, and 5 *for* each digit $i = 0$ to $k-1$. (It is assumed that steps 3,4, and 5 are applied in parallel to all the inputs $j = 1$ to N.)

Step 3: $z_i = max_j(y[j]_i$ and $p[j]_i = candidate)$.

Step 4: *if* $((p[j]_i = undecided)$ and $(y[j]_i \leq z_i))$ *then* $p[j] = eliminated$.

Step 5: Update remaining input states as follows:
if$((z_i = -1)$ and $(y[j]_i = 1)$ and $(p[j]_i = undecided))$ *then* $p[j]_{i+1} = candidate$.
else if $((z_i = -1)$ and $(y[j]_i = -1)$ and $(p[j]_i = candidate))$ *then* $p[j]_{i+1} = undecided$.
else if $((z_i = 1)$ and $(y[j]_i = 0)$ and $(p[j]_i = candidate))$ *then* $p[j]_{i+1} = undecided$.
else $p[j]_{i+1} = p[j]_i$.

If the dilation unit is used for erosion, then the inputs and outputs have to be replaced by their additive inverses. In other words, digit 1 has to be replaced by -1, and digit -1 has to be replaced by 1. Since digit 1 is represented by 01, and digit -1 is represented by 10, the bits corresponding to a digit have to be inverted. Since the output of the dilation unit is a radix-2 redundant number, it can be fed directly to the redundant adders of other erosion and dilation units.

5. IMPLEMENTATION

We have implemented a dilation/erosion unit for the case when $N = 9$ and $k = 8$. The unit functions like a dilation unit or an erosion unit depending on a control signal. We have used Berkeley CAD Tools to layout the chip. The technology was 2 micron n-well CMOS. The area of a single redundant adder was 0.3 sq.mm., and the delay was 10ns. The area of the 9-input MAX unit was 2.2 sq.mm., and the worst-case delay was 35 ns. The total die size (excluding I/O pads) was 5.3 sq.mm. The delay in the MAX unit can be reduced by applying look-ahead techniques and processing multiple digits at a time. This scheme is currently under investigation.

6. REFERENCES

[1] P. Maragos and R.W. Schafer. Morphological systems for multidimensional signal processing. *Proc. of the IEEE*, 78(4):690–709, April 1990.

[2] F. R. Giardina and E. R. Dougherty. *Morphological Methods in Image and Signal Processing.* New Jersey: Prentice Hall, 1986.

[3] W.C. Fang and et. al. Parallel morphological image processing with an opto-electronic VLSI array processor. In *Proc. of the IEEE Int'l Conf. on Acoustics, Speech, and Sig. Proc.*, pages 409–412, 1993.

[4] A.C.P. Loui and et. al. Non-linear pipeline architectures for morphological signal processing. In *Proc. of the IEEE Int'l Conf. on Circuits and Sys.*, pages 1438–1441, 1990.

[5] K.I. Diamantaras, K.H. Zimmerman, and S.Y. Kung. Integrated fast implementation of mathematical morphology operations in image processing. In *Proc. of the IEEE Int'l Conf. on Circuits and Sys.*, pages 1443–1445, 1990.

[6] M. Djunatan and T. Mengko. A programmable real-time systolic processor architecture for image morphological operations, binary template matching and min/max filtering. In *Proc. of the IEEE Int'l Conf. on Circuits and Sys.*, pages 65–68, 1991.

[7] F. Shih and O.R. Mitchell. Threshold decomposition of gray-scale morphology into binary morphology. *IEEE Trans. on Pattern Analysis and Machine Intell.*, 11(1):31–42, Jan. 1989.

[8] Y. Chang and N. Ansari. A practical VLSI realization of morphological operations. In *SPIE Visual Communications and Image Processing*, volume 1606, pages 839–850, 1991.

[9] C. L. Lee, J-S. Chen, and C-W. Jen. A VLSI architecture for gray-scale morphological filtering and its application. In *Proc. of the IEEE Int. Symp. on Circuits and Systems*, pages 2100–2103, 1991.

[10] H.R. Srinivas and K.K. Parhi. A fast VLSI adder architecture. *IEEE Journal of Solid-State Circuits*, 27(5):761–767, May 1992.