New Symbolic Substitution Algorithms for Optical Arithmetic using Signed-Digit Representation

Kai Hwang and Ahmed Louri

University of Southern California, Los Angeles California

ABSTRACT

In this paper, we present a new approach to designing optical digital arithmetic systems. We present new arithmetic algorithms based on a modified signed-digit number representation. New Signed-digit symbolic substitution rules are introduced to implement them. These signed-digit arithmetic algorithms are well suited for optical implementation because, of the confined carry propagation within adjacent digits. We present an optical architecture of such an arithmetic processor. The proposed optical arithmetic processor can potentially achieve $O(10^2)$ to $O(10^3)$ improvement in speed as compared with conventional electronic arithmetic processors.

1. INTRODUCTON

The inherent parallelism, high speed, noninterfering communication, and wide bandwidth of optics show hope for significantly improving the speed of arithmetic computations in future optical computers. In this paper, we proceed in a two-step approach: first, new algorithms are developed for digital arithmetic operations that are amenable to optical implementation; then, we present a new arithmetic architecture and discuss its implementation and performance based on the state-of-the-art optical technology.

Several number representations have been investigated in the past for optical arithmetic. Residue number system has been considered[1]. The advantage of usig residue numbers lies in performing carry-free addition, subtraction, and multiplication. However, this system suffers from the difficulties of performing division, an comparing two numbers. The residue number system is an integer field, however the results of division are not always integers and therefore do not always have a residue representation. Furthermore, it is not easy to determine the sign after a subtraction operation. A second number system known as DMAC has been introduced[2]. Its major drawback is the need for analog-to-digital conversion required to convert the mixed binary to pure binary numbers. This postprocessing step offsets the speed gained by optical processing.

In this paper, we use a *modified signed-digit* (SD) number representation originally proposed by Avizienis[3], and lately introduced to optics by Drake et al[4] to perform optical arithmetic computations. The SD number representation is a redundant system with a digit set of $\{\bar{1},0,1\}$, where $\bar{1}$ stands for -1. The redundancy overcomes the strong interdigit dependency that results in carry propagation manifested in conventional binary representation. Carry propagation is then limited to two adjacent digits in an SD system. This makes it possible to perform addition and subtraction of two SD numbers of any word length in parallel.

Holographic optical elements, prisms, and optical bistable devices were proposed for implementing the SD addition and subtraction optically[4]. Mirsalehi and Gaylord[5] proposed a direct table look-up method for implementing the SD addition. In this paper, we implement the SD addition, subtraction, multiplication, and division using *optical symbolic substitution* technique extended from the work of Huang[6]. We introduce a new set of bit-wise, *signed-digit, symbolic substitution* rules. The use of signed-digit number representation in conjunction with these new symbolic substitution rules for implementing optical operations will yield an optical arithmetic processor with a tremendous speed improvement over existing electronic counterparts.

2. SIGNED–DIGIT ARITHMETIC ALGORITHMS

The modified SD number system has a digit set of $\{\bar{1},0,1\}$. Given an SD number $Y = y_{n-1}y_{n-2}\cdots y_0.y_{-1}\cdots y_{-m}$, the algebraic value of Y is evaluated as :

$$Y_v = \sum_{i=-m}^{i=n-1} y_i \times 2^i, \quad y_i \in \{\bar{1},0,1\} \tag{1}$$

In this number system, there is no need for an explicit sign digit, in fact y_{n-1} determines the sign of Y. Hereafter, we concentrate on integer SD numbers for addition/subtraction and multiplication and on normalized fractions for division. We present below algorithms for various SD operations. The SD addition is generalized from Avizienis[3]. The SD multiplication and SD division algorithms are newly developed.

2.1 SD Addition and SD Subtraction

The addition of two n-digit SD integers, $X = x_{n-1}\cdots x_0$ and $Y = y_{n-1}\cdots y_0$ results in an $(n+1)$-digit SD integer, $S = s_n s_{n-1}\cdots s_0$. Three successive steps are needed to perform the totally parallel addition. At the first step, the following equation $x_i + y_i = 2t_{i+1} + w_i$, is performed at the i-th digit position, for $i = 0, \ldots, n-1$. where w_i and t_{i+1} are called the ***interim sum digit*** and the ***transfer digit*** respectively. These variables assume the following values:

$$w_i = \begin{cases} 1 & \text{if } x_i + y_i = \bar{1} \\ 0 & \text{if } |x_i + y_i| \neq 1 \\ \bar{1} & \text{if } x_i + y_i = 1 \end{cases} \qquad t_{i+1} = \begin{cases} 1 & \text{if } x_i + y_i \geq 1 \\ 0 & \text{if } x_i + y_i = 0 \\ \bar{1} & \text{if } x_i + y_i \leq \bar{1} \end{cases} \tag{2}$$

At the second step, the following equation $w_i + t_i = 2t'_{i+1} + w'_i$, is performed with :

$$w'_i \begin{cases} 1 & \text{if } w_i + t_i = 1 \\ 0 & \text{if } |w_i + t_i| \neq 1 \\ \bar{1} & \text{if } w_i + t_i = \bar{1} \end{cases} \qquad t'_{i+1} = \begin{cases} 1 & \text{if } w_i + t_i = 2 \\ 0 & \text{if } |w_i + t_i| \neq 2 \\ \bar{1} & \text{if } w_i + t_i = -2 \end{cases} \tag{3}$$

The third step provides the final sum digit:

$$s_i = w'_i + t'_i = \begin{cases} 1 & \text{if } w'_i + t'_i \geq 1 \\ 0 & \text{if } w'_i + t'_i = 0 \\ \bar{1} & \text{if } w'_i + t'_i \leq \bar{1} \end{cases} \tag{4}$$

Figure 1 shows a totally parallel adder constructed by three types of arithmetic Cells whose truth-tables are given in Table 1. There is no carry propagation beyond adjacent stages in the SD adder of Figure 1. Any sum digit is a function of only two adjacent operand digits. SD subtraction is performed by first negating the sign of the nonzero digits of the subtrahend, then perform the SD addition of the two operands. Example 1 illustrates the SD addition of $(-7)_{10} = (\bar{1}01\bar{1})$ and $(3)_{10} = (010\bar{1})$. The result is a 5-digit SD integer $S = (00\bar{1}00) = (-4)_{10}$.

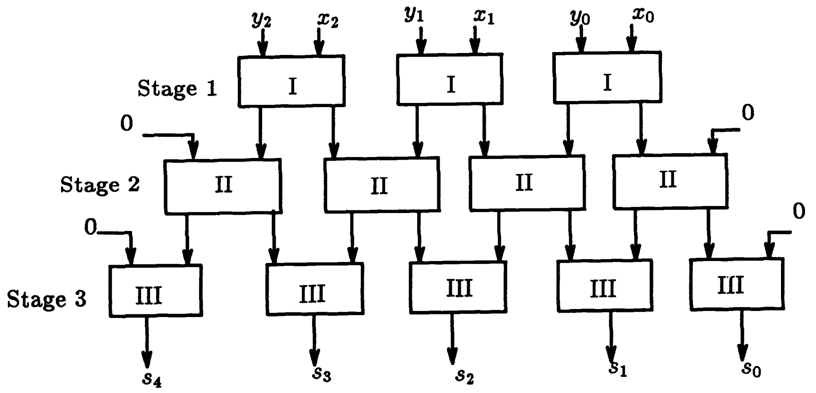


Figure 1. A parallel circuit for optical signed-digit addition (shown for operand length n = 3)

Example 1(SD Addition)

$X = (\bar{1}01\bar{1})_{sd}$	=		$\bar{1}$	0	1	$\bar{1}$	$= (-7)_{10}$
+							
$Y = (010\bar{1})_{sd}$	=		0	1	0	$\bar{1}$	$= (3)_{10}$
Stage 1		0	1	$\bar{1}$	$\bar{1}$	0	w_i
		$\bar{1}$	1	1	$\bar{1}$	0	t_{i+1}
Stage 2	0	$\bar{1}$	0	0	0	0	w'_i
	0	1	0	$\bar{1}$	0	0	t'_{i+1}
Stage 3	0	0	0	$\bar{1}$	0	0	s_i

$Z = (00\bar{1}00)_{sd} = (-4)_{10}$

2.2 SD Multiplication

The multiplication of two n-digit SD integers, X, Y produces a $2n$-digit SD integer $P = p_{2n-1}p_{2n-2}\cdots p_0$. This product can be expressed as:

$$P = y_{n-1} * X \times 2^{n-1} + \cdots + y_1 * X \times 2^1 + y_0 * X \times 2^0 \tag{5}$$

where y_i is the i-th multiplier digit, and * is the SD AND operation defined as follows for any $x, y \in \{\bar{1}, 0, 1\}$:

$$x * y = \begin{cases} 1 & \text{if } x = y = 1 \\ 0 & \text{if } (x = 0) \vee (y = 0) \\ \bar{1} & \text{if } (x = 1 \wedge y = \bar{1}) \vee (x = \bar{1} \wedge y = 1) \end{cases} \tag{6}$$

where $\vee, \wedge$ represent the logical or and the logical and. Using a nested notation, we can rewrite Eq. (5), in the following form:

$$P = 2^{n-1} \times (y_{n-1} * X + 2^{-1} \times (y_{n-2} * X + \cdots + 2^{-1} \times (y_1 * X + 2^{-1} \times (y_0 * X))\ldots) \tag{7}$$

From Eq. (7), we deduce a recursive formula to compute the i-th partial product P_i:

$$P_i = (P_{i-1} + y_{i-1} * X) \times 2^{-1}, \quad i = 1, \ldots, n, \quad P_0 = 0. \tag{8}$$

The final result is $P = P_n \times 2^{n-1}$ after n iterations. The product is obtained by a sequence of right shifts and additions only. At each iteration, we generate one digit of the product starting from the least significant digit. The SD multiplication algorithm has a time complexity $O(n)$, where n is the precision of the multiplier. Example 2 illustrates the SD multiplication process with two SD numbers $X = (1\bar{1}\bar{1}0)$ and $Y = (\bar{1}\bar{1}010)$ to yield a product $P = (00\bar{1}0\bar{1}11100)$.

Example 2 (SD multiplication)

Multiplicand $X = (1\bar{1}\bar{1}10)_{sd} = (6)_{10}$ Multiplier $Y = (\bar{1}\bar{1}010)_{sd} = (-22)_{10}$

Step	Digit	Quantity	SD representation
		P_0	00000
$i = 0$	$y_0 = 0$	y_0X	00000
	add	$P_0 + y_0X$	00000
	shift	P_1	000000
$i = 1$	$y_1 = 1$	y_1X	$1\bar{1}\bar{1}10$
	add	$P_1 + y_1X$	$0010\bar{1}00$
	shift	P_2	$0010\bar{1}00$
$i = 2$	$y_2 = 0$	y_2X	00000
	add	$P_2 + y_2X$	$001\bar{1}\bar{1}100$
	shift	P_3	$001\bar{1}\bar{1}100$
$i = 3$	$y_3 = \bar{1}$	y_3X	$\bar{1}11\bar{1}0$
	add	$P_3 + y_3X$	$000\bar{1}\bar{1}1100$
	shift	P_4	$000\bar{1}\bar{1}1100$
$i = 4$	$y_4 = \bar{1}$	y_4X	$\bar{1}11\bar{1}0$
	add	$P_4 + y_4X$	$00\bar{1}0\bar{1}11100$
Final product			$= 00\bar{1}0\bar{1}11100$
			$= (-132)_{10}$

2.3 SD Division

To take advantage of the parallel addition and the fast multiplication schemes described above, we developed an algorithm for SD division based on convergence division method[7] that uses only SD addition, SD multiplication, and shift operations.

The algorithm generates the quotient $Q = X/Y$ without a remainder. The operands X, Y are fractions in normalized form: $0.5 \leq |X| < |Y| < 1$, to avoid overflow and exception cases. The algorithm consists of finding a sequence of multiplicative factors $m_0, m_1, \ldots, m_n$ such that $Y \times (\prod_{i=0}^{i=n} m_i)$ converges to a definite limit k (within an acceptable error criterion). The procedure consists of implementing the following recursions, starting with $X_0 = X$ and $Y_0 = Y$:

$$X_{i+1} = X_i \times m_i, \qquad Y_{i+1} = Y_i \times m_i \tag{9}$$

such that for a small n : $Q = \frac{X}{Y} = \frac{X \times (\prod_{i=0}^{i=n} m_i)}{Y \times (\prod_{i=0}^{i=n} m_i)} = \frac{\alpha}{k}$. The final quotient Q is α and $k \to 1$. The recursive formula, $Y_{i+1} = Y_i \times m_i$, can be written as:

$$Y_{i+1} = \phi(Y_i) \tag{10}$$

We desire the function $\phi(Y)$ to converge to 1. There are several iterative methods that can be devised to enable a sequence $\phi(Y)$ to converge to 1. Krishnamurthy[8] has studied the quadratic convergence form and has found that, in order for $Y_i \times m_i$ to quadratically converge to 1, m_i should be :

$$m_i = (2 - Y_i) \tag{11}$$

This implies that the sequence of multipliers, m_i for $i = 0, 1, 2, \ldots, n$ can be easily obtained by two's complement operations. The final quotient becomes $Q = X_n$. The complete SD division algorithm is specified in Figure 2. The following example illustrates the steps involved in finding the quotient of $X = (0.\bar{1}1)$ and $Y = (0.11)$ using the SD division algorithm. Given the dividend $X = (0.\bar{1}0) = (-0.5)_{10}$ and the divisor $Y = (0.11) = (0.75)_{10}$, for a 16-bit precision, the algorithm finds the quotient $Q = X/Y = (\bar{1}.1\bar{1}10\bar{1}\bar{1}10\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}1) = (-0.66664)_{10}$ after 3 iterations.

Example 3 (SD division)

Dividend $X = (0.\bar{1}0)_{sd} = (-0.5)_{10}$ Divisor $Y = (0.11)_{sd} = (0.75)_{10}$

Iteration step	Multiplicative factor	Accumulated denominator	Accumulated numerator
$i = 0$	$m_0 = 2 - Y_0$ $(1.01)_{sd}$ $= (1.25)_{10}$	$Y_1 = Y_0 * m_0$ $(1.000\bar{1})_{sd}$ $= (0.9375)_{10}$	$X_1 = X_0 * m_0$ $(0.\bar{1}\bar{1}10)_{sd}$ $= (-0.625)_{10}$
$i = 1$	$m_1 = 2 - Y_0 * m_0$ $(1.001\bar{1})_{sd}$	$Y_2 = Y_0 * m_0 * m_1$ $(1\bar{1}.000000\bar{1})_{sd}$	$X_2 = X_0 * m_0 * m_1$ $(\bar{1}.1\bar{1}10\bar{1}\bar{1}10)_{sd}$
$i = 2$	$m_2 = 2 - Y_0 * m_0 * m_1$ $(1\bar{1}.0000001)_{sd}$ $= (1.00390625)_{10}$	$Y_3 = Y_0 * m_0 * m_1 * m_2$ $(1\bar{1}.000000000000000\bar{1})_{sd}$ $Y_3 \to 1$	$X_3 = X_0 * m_0 * m_1 * m_2$ $Q = (\bar{1}.1\bar{1}10\bar{1}\bar{1}10\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}1)_{sd}$ $= (-0.6666..)_{10}$

3. OPTICAL SYMBOLIC SUBSTITUTION RULES

In this section, we present optical symbolic substitution rules needed to carry out the SD arithmetic algorithms. Then we present a processor architecture for implementing optical The SD arithmetic optically. There are several prop‿rties of light that can be used to encode the digit set $\{1, 0, \bar{1}\}$. These include light intensity, polarization, and optical signal phase. Using light intensity, we need 2 binary pixels to encode 3 digits. A possible encoding scheme represents the digit value 1 by a bright pixel above a dark one, the digit value $\bar{1}$ by a reversed pixel pattern, and a zero by two dark pixels as shown in Figure 3.a. Note that in this scheme 1 and $\bar{1}$ negate each other.

3.1 Rules for SD Addition

In order to implement the SD addition optically, we derive the symbolic substitution rules from the truth-table specifications of Table 1. The left-hand-side of the rules are the input combinations and the right-hand-side represents the table entries as shown in Figure 3.(b-d). On the surface, it seems that we need $3^3 = 27$ rules to implement the SD addition, however, a closer look at the SD adder design in Figure 1, reveils that the logic of the first stage and of the second stage are very similar. Furthermore, if we pad the output of the last stage with zero, five of the nine rules become similar to stage 2 and 3. This results in only 17 rules for optical SD addition. The SD subtraction needs an extra stage to perform digitwise arithmetic negation, before the SD addition is performed. The negation stage requires two more substitution rules as shown in Figure 4.a, to invert 1 to $\bar{1}$ and vice versa.

Table 1 Truth-table descriptions of three arithmetic Cell Types used in the optical adder in Fig.1

x_i \ y_i		1	0	$\bar{1}$
1	t	1	1	0
	w	0	$\bar{1}$	0
0		1	0	$\bar{1}$
		$\bar{1}$	0	1
$\bar{1}$		0	$\bar{1}$	$\bar{1}$
		0	1	0

(a) For Type I Cell

t_i \ w_i		1	0	$\bar{1}$
1	t'	1	0	0
	w'	0	1	0
0		0	0	0
		1	0	$\bar{1}$
$\bar{1}$		0	0	$\bar{1}$
		0	$\bar{1}$	0

(b) For Type II Cell

t'_i \ w'_i		1	0	$\bar{1}$
1	s	1	1	0
0		1	0	$\bar{1}$
$\bar{1}$		0	$\bar{1}$	$\bar{1}$

(c) For Type III Cell

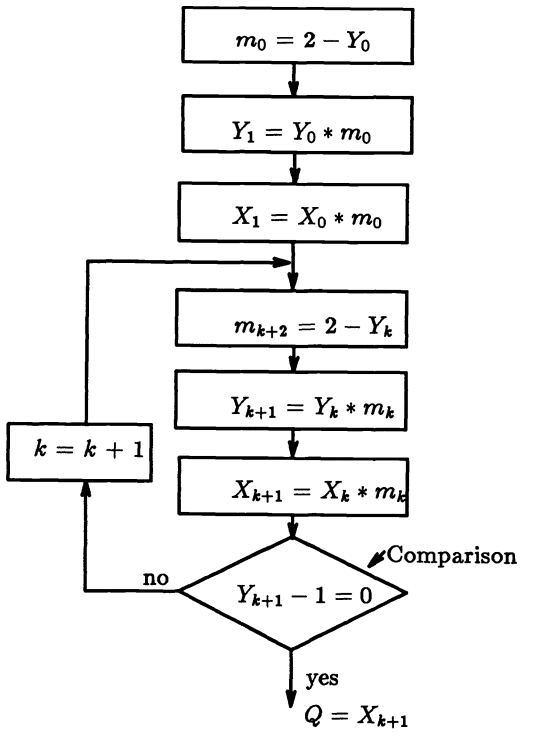
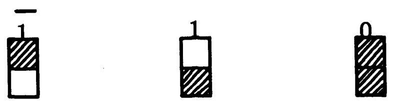
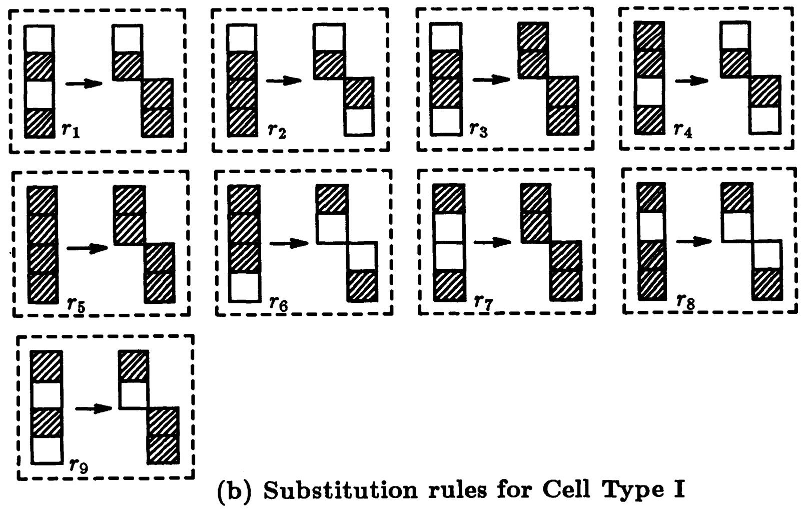


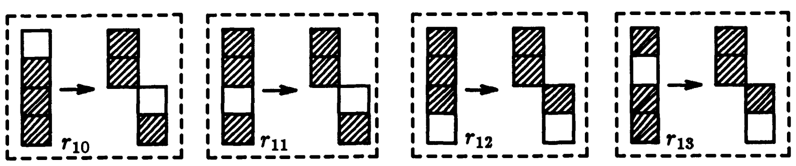
Figure 2 A signed-digit division algorithm that uses only signed-digit addition, AND and shifts



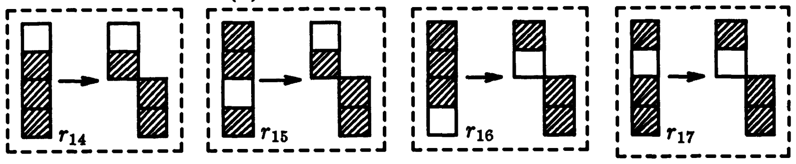
(a) Light intensity encoding of the digit set $\{\bar{1},0,1\}$



(b) Substitution rules for Cell Type I



(c) Additional rules for Cell Type II



(d) Additional rules for Cell Type III

Figure 3 The optical substitution rules for signed-digit addition

3.2 Rules for SD Multiplication

The SD multiplication is performed as a sequence of SD addition and SD shifts described previously. To produce the partial product P_i (Eq.8), we need to generate the multiplicative term $y_{i-1} * X$, add it to the previous partial product P_{i-1}, and shift the result one digit position to the right. We introduce symbolic substitution rules shown in Figures 4.b and 4.c to perform the SD AND operation and the SD right shift.

3.3 Rules for SD Division

The SD division outlined in Figure 2, requires SD addition, SD subtraction, SD multiplication, and SD comparison. The SD comparison operation can be implemented by subtracting 1 from Y_k (see Figure 2) and checking whether the subtraction result is equal to zero. The checking operation can be done using a photodetector array. The subtraction result $(Y_k - 1)$ is sent to the photodetector which generates an electrical signal to the controller in the presence of zero output. If $Y_k - 1 = 0$, then the controller stops the iterative process and outputs the last accumulated numerator, X_k as the quotient Q. Otherwise, the system keeps iterating until $Y_k - 1$ becomes zero.

4. OPTICAL ARITHMETIC ARCHITECTURE

Figure 5 depicts a block diagram of an optical arithmetic processor. It consists of *input registers*, an *input multiplexer*, an *SD arithmetic unit*, an *output router*, an *optical interconnect*, and a *control unit*. The heart of the processor is the SD arithmetic unit. This unit is divided into 3 functional modules: the adder/subtractor, the shifter, and the AND modules. Each module comprises the symbolic substitution rules associated with the SD operation to be performed, i.e. the adder/subtractor module consists of 19 rules, 17 for SD addition, and 2 for SD negation. The input registers represent temporary storage. The input multiplexer and the output router monitor the data flow. The control unit is responsible for the execution sequence and control flow. In what follows, we show how the SD operations can be implemented on such a processor.

SD addition: Adding two SD integers, X, and Y of length n using the processor of Figure 5 is done as follows: we start by loading the operands X and Y into register A and B respectively. The input multiplexer forms a plane of size $2 \times n$ by putting the contents of A and B on top of each other, and sends it to the adder module. The adder produces the SD sum S, after 3 successive stages. At each stage, it applies the corresponding symbolic substitution rules shown in Figure 3. The sum S is then sent to the router that outputs it.

SD multiplication: Multiplying two SD integers is performed as follows: initially the multiplicand X and the multiplier Y are stored in register A and B respectively, and register C is cleared to zero. The input multiplexer transfers the contents of A and B to the SD AND module, which in turn generates the multiplicative term $y_0 * X$. The router transfers this term to register B. The contents of register B and C are transferred to the adder, which produces $P_0 + y_0 * X$ and sends it to register C. A right shift of register C produces the first partial product $P_1 = (P_0 + y_0 * X) \times 2^{-1}$ that is put back into C. The whole process continues until all the multiplier's bits are exhausted, and the product P_n is generated.

SD division: knowing how SD addition, and multiplication is carried out, it is easy to figure out how the division algorithm of Figure 2 can be implemented on the processor. Unlike SD addition and SD multiplication, the SD division requires a variable number of steps depending on the convergence rate. The controller needs to interact with the data at each iteration for the algorithm to terminate. The algorithm should terminate and output the quotient when the denominator Y_i is equal 1. This interaction is implemented as a sequence of subtraction and detection operations.

5. OPTICAL IMPLEMENTATION CONSIDERATIONS

We can construct the processor from several modules: input multiplexer, arithmetic unit, output router and optical interconnects. We discuss below the implementation of each module separately, assuming that the formats of signals between modules(for example, data arrangement, carrier wavelength, etc.) are matched to each other.

The input multiplexer consists of a set of four optical registers, and a selector. Depending on the control signals, the selector selects one or two registers at a time to be latched to the arithmetic unit. This gives rise to 10 possibilities: transmitting A, B, C, D, AB, AC, AD, BC, BD, or CD (BA, CA, etc. are considered the same as AB, AC, etc.). A possible implementation is using combinatorial logic and optical gates to implement the 10 possibilities. A total of 4 control lines can generate $2^4 = 16$ combinations, we use 10 of these to control the combinatorial circuit.

The dynamic selection of a particular functional module (adder, AND, shifter) can be done in two ways. An obvious way is to replicate the selected register (or a combined image) into three copies, one for each functional module and only the active module will produce an output. Holograhic techniques can be used to replicate the input. This selection method, although fast, would be very light inefficient due to the replication of the input image at every iteration through the system. An alternative way, would be the use of an acousto-optic (or an electro-optic) deflector to deflect the input to the desired functional module. A stack of acousto-optic cells can be driven by electrical control signals whose frequencies are proportional to the desired spatial position of the desired module. Three angular positions are sufficient for selecting one of the 3 functional modules. This method is relatively fast, and more economical. Its drawback would be the number of electrical lines required to control the AO cells.

The arithmetic unit is composed of 3 functional modules: SD adder, SD AND, and SD shifter. These modules implement the substitution rules associated with each operation. Since their implementations are equivalent we only discuss the adder implementation. Figure 6 shows a schematic diagram of the optical SD adder. The input plane containing the two operands is replicated as many times as there are active rules; i.e. for the fist stage of the SD addition, we need to replicate the input 9 times and send it to the 9 substitution rules of the first stage (see section 2). Each active rule produces a copy where all the occurrences of the left side of that rule have been replaced with the right side. The outputs of active rules are then superimposed to form the final output. The SD addition requires 3 iterations, a fixed optical interconnects based on holograms or discrete passive optical devices can be used to feed back the output image between intermediate stages. Note that in case of the SD AND and shifter modules, we do not need the internal optical feedback. The final result is sent through a static beam steering element to the router. There are several methods that have been reported to implement the two processing steps of symbolic substitution technique[9,10,11,12,13]. Any one of these methods can be used to implement the individual symbolic substitution rules.

A simple optical setup for the router is shown in Figure 7. It consists of two polarizing image splitters (PIM1, PIM2), two halfwave plates (H1, H2) which are able to switch the polarization of light carrying information when activated by an electrooptical effect, and a photodetector array that detects the intensity of light and converts it to a proportional electrical signal. The combination of the on/off states of the polarizing beam splitters implement the 3 states of the router. Each PIM is preceded by a controllable halfwave plate. These halfwave plates allow the PIM's to direct the output in one direction or the other depending on their on and off states. For example, when H1 is on, the output is fed back to the input selector through PIM1; when H1 is off, the output is forwarded to PIM2 which is controlled via H2. When H2 is on, the output is sent to the detector array for detecting the zero value as discussed before; when H2 is off, the router outputs the data.

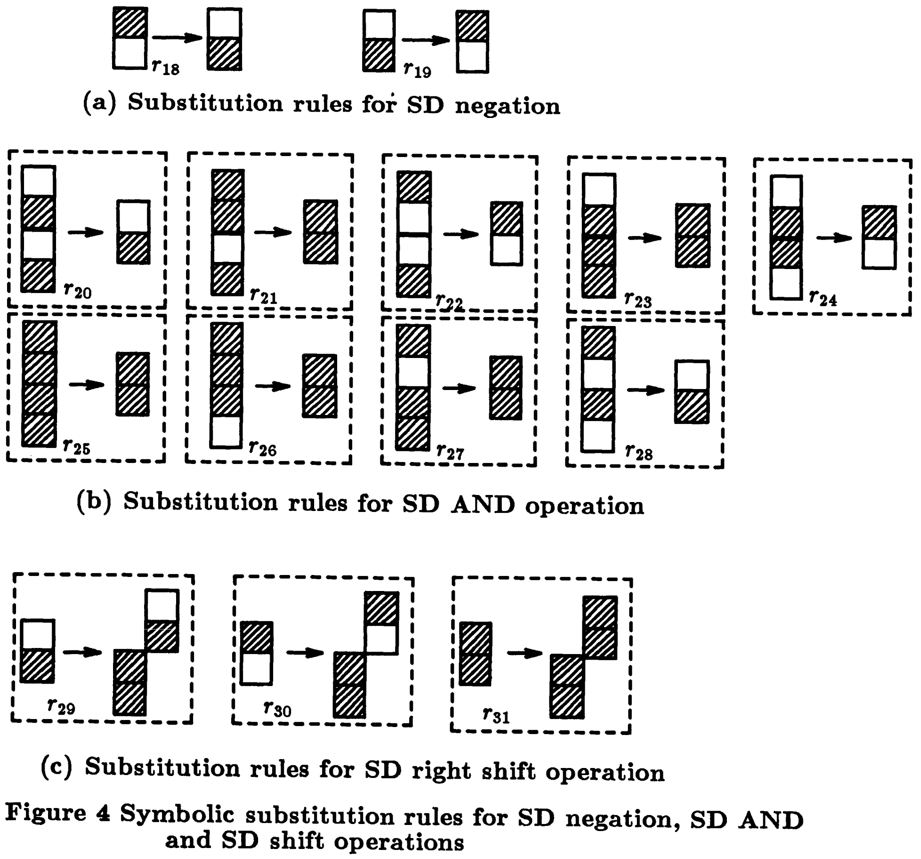

r_{18}
r_{19}
(a) Substitution rules for SD negation
r_{20}
r_{21}
r_{22}
r_{23}
r_{24}
r_{25}
r_{26}
r_{27}
r_{28}
(b) Substitution rules for SD AND operation
r_{29}
r_{30}
r_{31}
(c) Substitution rules for SD right shift operation

Figure 4 Symbolic substitution rules for SD negation, SD AND and SD shift operations

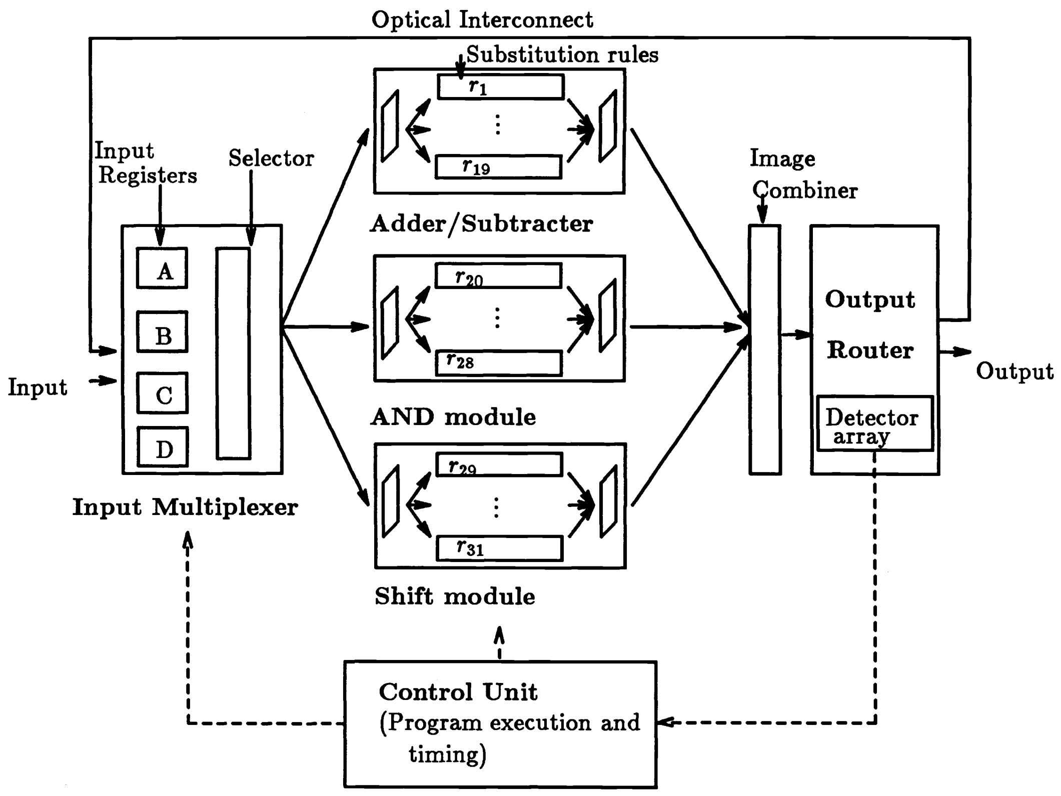
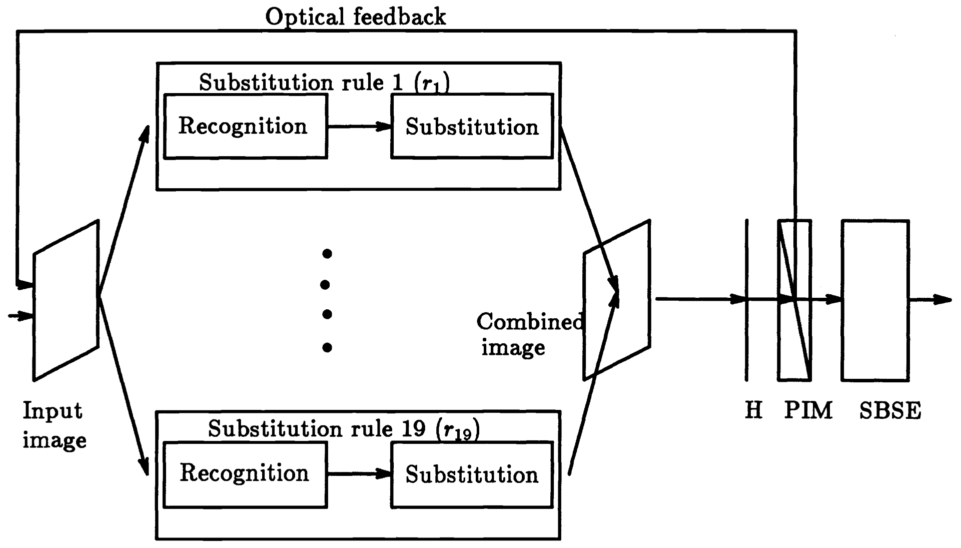


Figure 5 The block diagram of an optical arithmetic processor. (Solid arrows are optical paths and dash arrows are control signals.)



H : halfwave plate
PIM : Polarizing Image Splitter
SBSE : Static Beam Steering Element

Figure 6 Functional units used in the optical SD adder based on symbolic substitution

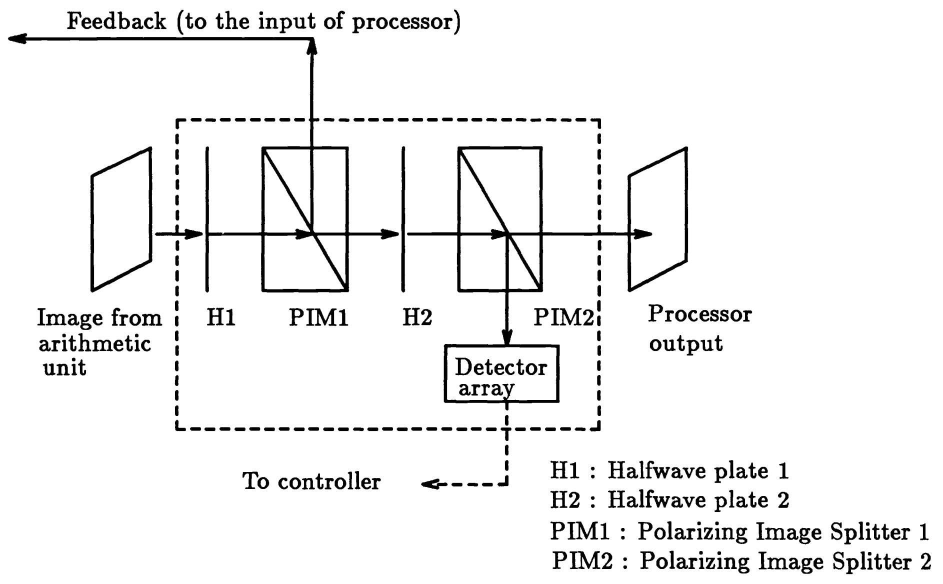

Feedback (to the input of processor)
Image from arithmetic unit
H1
PIM1
H2
PIM2
Detector array
Processor output
To controller
H1 : Halfwave plate 1
H2 : Halfwave plate 2
PIM1 : Polarizing Image Splitter 1
PIM2 : Polarizing Image Splitter 2

Figure 7 An optical setup for the output router

6. PERFORMANCE PREDICTION

We predict the performance of the system by analyzing the SD addition time. Each symbolic substitution rule is implemented by optical means[9,11,12]. Consider the design of Figure 5.

Let :

T_{select}	=	The input selector response time.
$T_{replicat}$	=	Image replication time.
$T_{recognition}$	=	The pattern recognition time.
$T_{substitute}$	=	The pattern substitution time.
T_{resp}	=	The response time of the optical gate array.
$T_{superpose}$	=	The superposition time.
$T_{feedback}$	=	The internal feedback time of the adder.
T_{rout}	=	The response time of the router.

Then the optical addition time is a weighted sum of these terms:

$$T_{add} = T_{select} + T_{replicat} + 3(T_{replicat} + T_{recognition} + T_{resp} + T_{substitute} + t_{superpose}) + 2T_{feedback} + T_{rout} \tag{12}$$

Eq. (12) can be rewritten as follows:

$$T_{add} = T_{propagate} + T_{switch} \tag{13}$$

where

$$T_{propagate} = 4T_{replicat} + 3T_{recognition} + +3T_{substitute} + 3t_{superpose} + 2T_{feedback} + T_{rout} \tag{14}$$

$$T_{switch} = T_{select} + 3T_{resp} \tag{15}$$

Each component of $T_{propagate}$ represents light propagation delay through associated passive optical devices such as lenses, prisms, beam splitters, etc. T_{switch} designates the response time of active optical devices such as the optical gates used to implement the combinatorial logic of the input selector, and the NOR-gate array needed in symbolic substitution[9]. The dominant limitation to the speed is the switching time of the active optical devices, because $T_{propagate}$ can be made in the order of $O(10^{-9})$ sec, however, state-of-the-art optical switching time is in the order of $O(10^{-6})$ sec for a reasonable power consumption.

7. CONCLUSIONS

We have proposed a new symbolic substitution algorithms for high-speed optical arithmetic operations. These algorithms are based on a modified signed-digit number representation which offers carry-free addition. We have sketched an optical arithmetic processor for implementing these algorithms, using state-of-the-art optical and electro-optical devices. The performance of the optical arithmetic processor is estimated to be $O(10^2)$to $O(10^3)$ times faster than any of the existing electronic processors.

ACKNOWLEDGEMENTS

This research was supported by an ONR Contract No. N14-86-k-559 and in part by a NOSC Contract No. 85-D-203

References

[1] A. Huang, Y. Tsunoda, J. W. Goodman, and S. Shihara, " Optical Computation Using Residue Arithmetic," *Appl. Opt.*, Vol. 18, no. 2, 15 Jan. 1979.

[2] D. Psaltis, D. Casasent and M. Carlotto, " Accurate Numerical Computation by Optical Convolution, " *Int'l Optical Computing Conf. (Book II)*, Ed. W.T. Rhodes, 1980.

[3] A. Avizienis, " Signed-Digit Number Representations for Fast Parallel Arithmetic , " *Trans. Elect. Computers*, EC-10 pp. 389-398, 1961.

[4] B.L. Drake, R.P. Bocker, M.E. Lasher, R.H. Patterson, and W.J. Miceli, " Photonic Computing using the Modified Signed-digit Number Representation, " *Opt. Eng.* 25 (1). pp. 038-043 Jan. 1986.

[5] M.M. Mirsalehi and T.K. Gaylord, "Logical Minimization of Multilevel Coded Functions," *Appl. Opt.*, Vol. 25, No. 18, Sept. 1986.

[6] A. Huang, " Parallel Algorithms for Optical Digital Computers, " *Proc. IEEE Tenth Int'l Optical Computing Conf.* pp. 13-17, 1983.

[7] E.V. Krishnamurthy, "On optimal Iterative Schemes for High-Speed Division," *IEEE Trans. on Computers,* Vol. C-19, No. 3, March 1970.

[8] R.Z. Goldschmidt, " Applications of Division by Convergence, " *Master Thesis,* M.I.T. Cambridge, Mass., June 1964.

[9] K.H. Brenner, A. Huang, and N. Streibl, " Digital Optical Computing with Symbolic Substitution, " *Appl. Opt.*, Vol. 25, No. 18, 15 Sept. 1986.

[10] K.H. Brenner, " New Implementation of Symbolic Substitution Logic, " *Appl. Opt.*, Vol. 25, No. 18, 15 Sept. 1986.

[11] Y. Li, G. Eichmann, R. Dorsinville and R.R. Alfano, "An AND Operation-Based Optical Symbolic Substitution Pattern Recognizer," *Opt. Comm.* Vol. 63, No. 6, Sept. 1987.

[12] A. Louri and K. Hwang, " Parallel Architectures for Optical Computing," *In Proc. of Third Int'l SIAM Conf. on Parallel Processing for Scientific Computing*, Los Angeles, Dec. 1-4, 1987.

[13] S. Toborg and K. Hwang, "Exploring Neural Network and Optical Computing Technologies," *in Parallel Processing for Supercomputing and Artificial Intelligence,* (K. Hwang and D. DeGroot, eds.) McGraw Hill, 1988.

[14] K. Hwang, *Computer Arithmetic : Principles, Architecture, and Design*, John Wiley & Sons, Inc., New York, 1979.