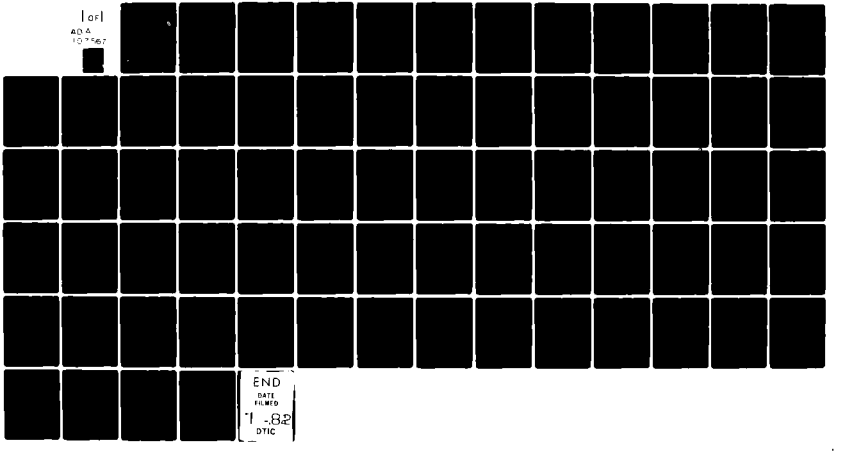
AD-A107 567 STANFORD UNIV CA STANFORD ELECTRONICS LABS F/G 9/3
OPTICAL COMPUTATION USING RESIDUE ARITHMETIC.(U)
MAR 78 A HUANG, Y TSUNODA, J W GOODMAN AFOSR-77-3219
UNCLASSIFIED SU-SEL-78-007 AFOSR-TR-81-0743 NL



LEVEL II

1

SEL-78-007

OPTICAL COMPUTATION

USING RESIDUE ARITHMETIC

Alan Huang
Yoshito Tsunoda
Joseph W. Goodman

March 1, 1978

Annual Technical Report No. 6422-1

Research sponsored by the Air Force Office of Scientific Research, Air Force Systems Command, USAF, under Grant No. AFOSR-77-3219. The United States Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation hereon.

Information Systems Laboratory
Stanford Electronics Laboratories
Stanford University
Stanford, California

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: AFOSR-TR-81-0743	2. GOVT ACCESSION NO.: AD-A107567	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Optical Computation Using Residue Arithmetic.	5. TYPE OF REPORT & PERIOD COVERED: Annual Report 1.1.77.-1.31.78.
	6. PERFORMING ORG. REPORT NUMBER: 1-SEL-78-007, TR-64
7. AUTHOR(s): Alan Huang, Yoshito Tsunoda, Joseph W. Goodman	8. CONTRACT OR GRANT NUMBER(s): AFOSR-77-3219
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Stanford Electronics Laboratories, Stanford University, Stanford, Ca. 94305	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61102F, 2305/B1
11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Office of Scientific Research, Bldg. 410, Bolling AFB, DC.20332	12. REPORT DATE: 1 Mar 1978
	13. NUMBER OF PAGES: 70
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): unclassified
	15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Residue arithmetic
Optical data processing
Integrated optics
Optical computing

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The possible use of residue arithmetic in an optical or opto-electronic computer is investigated. The fundamental properties of the residue number system are first considered. Various optical methods for implementing residue operations are discussed. A configuration for an opto-electronic residue matrix-vector multiplier is proposed. All-electronic implementation of residue operations is also considered, and the potential advantages of optics are identified.

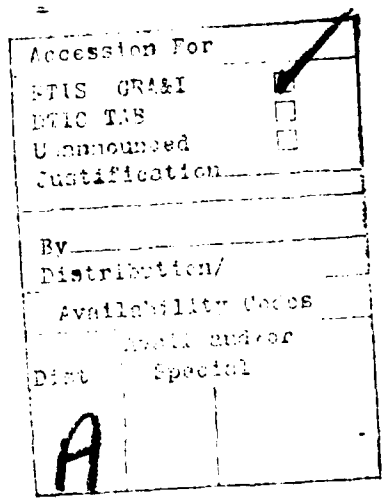
DD FORM 1 JAN 73 1473

Unclassified 332400

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ABSTRACT

The possible use of residue arithmetic in an optical or opto-electronic computer is investigated. The fundamental properties of the residue number system are first considered. Various optical methods for implementing residue operations are discussed. A configuration for an opto-electronic residue matrix-vector multiplier is proposed. All-electronic implementation of residue operations is also considered, and the potential advantages of optics are identified.



I. INTRODUCTION

A wide variety of coherent and incoherent optical systems have been developed for performing data processing operations, in either discrete or continuous form.[1,2,3] However, these systems have always suffered from the fact that they have limited output accuracy and dynamic range. Methods which retain the high degree of parallelism obtainable from optics, but which offer improved accuracy and dynamic range, are therefore of great interest.

The limiting accuracy of analog optical systems is often stated as being about 1 part in 1000 (0.1%, or 10 bits) under the best of conditions. This limitation arises from many sources and depends on the particular system considered; often the final detecting devices are limited to this kind of accuracy.

Closely connected with the issue of accuracy is the limited dynamic range of any physical system. Noise requires that the outputs be greater than some lower limit, while system nonlinearities require that the output not exceed a certain upper limit. We define the dynamic range of a given system, measured in bits, to be the base 2 logarithm of the ratio of the largest to the smallest allowable outputs.

To solve the problems of limited output accuracy and dynamic range, many possibilities for the construction of digital optical computing systems have been explored.[4,5,6,7] However, the difficulties in realizing optical logical switching devices, which play an essential role in a conventional optical-digital arithmetic unit, have so far prevented a practical solution to this problem.

In this report we consider some new types of high-speed opto-electronic systems capable of performing data processing operations with higher accuracy and dynamic range than are afforded by more conventional analog optical systems. In the systems of interest here, residue arithmetic is used, rather than the more conventional binary or decimal arithmetics. Electronic correlators based on the residue number system have been considered[8] and even constructed[9] in the past, and the use of residue arithmetic in an optical arithmetic unit has been proposed recently.[10] Such a system offers the interesting possibility of a direct trade-off between space-bandwidth product and accuracy, a property not shared by conventional optical processors.

There are two features of the residue number system that are extremely well matched to optical systems. Most obvious is the fact that there is no carry mechanism needed in residue arithmetic. This property allows all computations to run in parallel, with no necessity for interconnections between the results of sub-calculations until the final decoding step, which returns the calculation results to a more conventional number system.

A second important property arises from the fact that the residue number system decomposes a calculation into sub-calculations of smaller computational complexity. More precisely, each sub-calculation for a different modulus requires an accuracy commensurate with that modulus, yet a much higher accuracy is achieved after the results of these low-accuracy sub-calculations are recombined. The total range of the output is determined by the product M of all the moduli. As we increase M by increasing the number and/or the size of the moduli, the accuracy of the system improves. Once a calculation requiring a large dynamic range is decomposed into segments that can be handled directly by conventional analog

methods, the full advantage of the parallel processing of optical systems can be utilized to handle these segments.

In this report we first introduce the reader to the residue number system. Some optical methods for implementing residue arithmetic operations are then discussed. Next, the basic concept of an opto-electronic residue matrix-vector multiplier is described. Electronic implementations of a residue computer are considered. Finally, the potential advantages of optics are discussed.

II. RESIDUE ARITHMETIC

A. A Brief Introduction to Residue Arithmetic

There is evidence that residue arithmetic is theoretically the fastest way of performing addition, subtraction, multiplication, and polynomial transforms.[11,12,13,14,15] The main characteristic of the number system is that there are no carries and thus all columns of a calculation can be processed in parallel.

The residue number system is based upon N fixed and relatively prime (i.e., containing no common factors) integers $m_1, m_2, \ldots, m_N$ which are called moduli. The residue of any integer X with respect to a particular modulus m_i is denoted R_{m_i}, and is defined to be the least positive integer remainder of the division of X by m_i. The N-tuple of residues $(R_{m_i}, R_{m_2}, \ldots, R_{m_N})$ with respect to the N different moduli provides a unique representation of any integer X in the range 0 to $M-1$, where M is the product of the relatively prime moduli

$$M = \prod_{i=1}^{N} m_i. \qquad (1)$$

For integers outside this range the residue representation is ambiguous.

In order to help the reader understand what is probably an unfamiliar

number system, we now present some very specific examples of how arithmetic operations are carried out using residues. The moduli chosen for the various examples presented throughout this section are 5, 7, 9 and 4. Note that these numbers are relatively prime. The product of the moduli is in this case $M = 1260$, and hence the useful range before ambiguity is 0 to 1259. The order of the moduli is of no consequence, except that the last modulus is chosen to be an even number. As will be discussed later, this convention allows us to easily detect negative numbers, the unambiguous range being changed to $-630 \leq X \leq +629$.

B. Addition

An example of addition of two numbers in the residue system is presented in table 1. In this example, 19 is converted to (4,5,1,3) in the residue system. 4 is the residue of 19 for modulus 5; that is, 4 is the least positive integer remainder of the division of 19 by 5. Similarly, 5 is the residue for modulus 7, 1 for modulus 9, and 3 for modulus 4. 87 is translated into residues in a similar manner, yielding (2,3,6,3). Now each column in table 1 is added independently in the conventional fashion, but the sum is expressed as a residue with respect to the associated modulus. For example, in the first column, 4+2 = 6, but the residue of 6 for modulus 5 is 1. The other columns are treated similarly. The residue result (1,1,7,2) means that the conventional sum, when divided by 5 has remainder 1, when divided by 7 has remainder 1, etc. The four residues uniquely determine the answer within the range 0 to 1259. A quick check of the results can be obtained by converting the conventional result 106 into its residue representation. The answer is indeed (1,1,7,2). We defer to a later section a discussion of exactly how a residue representation can be decoded

		5	7	9	4
19	→	4	5	1	3
+87	→	+2	+3	+6	+3
106	←	1	1	7	2

TABLE I

to yield a decimal or other useable representation.

C. Subtraction

An example of subtraction in the residue system is presented in table II. To perform subtraction, the subtrahend is first converted to residues, and then each residue digit is complemented with respect to its particular modulus. Thus the residue 4 is complemented with respect to its modulus 5, resulting in 1. The other residues are complemented in a similar manner. The process then proceeds as with addition.
As alluded to previously, it is possible to represent both positive and negative numbers by residues. When one of the moduli is an even number, the range of 0 to M/2-1 can be used to represent positive numbers, while the range M-1 to M/2 represents complemented or negative numbers. If a number is found to be in the range M-1 to M/2 , M must be subtracted from it to give the correct answer. The sign of a number is thereby implicitly contained in its residue representation, just as in binary two's complement notation.

D. Multiplication

An example of multiplication in the residue system is presented in table III. The multiplier and multiplicand are converted into residues. The product of each column is expressed as a residue with respect to the corresponding modulus. Thus the result of this particular multiplication is the residue number (1,2,3,0) , which indeed is the residue representation of the decimal answer 156.

E. Division

The residue number system is an integer field. However, the results of division are not always integers, and hence do not always have a residue representation. For this reason, in the most general case, division is not possible using residues. However, in certain special cases, division can

					5	7	9	4
106	→				1	1	7	2
-99	→	4	1	0	3	→	+1	+6	+0	+1
7			←				2	0	7	3

TABLE II

		5	7	9	4
13	→	3	6	4	1
×12	→	×2	×5	×3	×0
26					
13					
156	←	1	2	3	0

TABLE III

be performed by means of multiplicative inverses.

An integer b is called the multiplicative inverse of an integer c for a particular modulus m_i if

$$\mathrm{MOD}(b \times c, m_i) = 1 \quad , \tag{2}$$

where $\times$ signifies a product and $\mathrm{MOD}(X, m_i)$ signifies the operation of taking the residue of X with respect to modulus m_i. We represent the fact that b is the multiplicative inverse of c for modulus m_i symbolically by the notation

$$b = \left| \frac{1}{c} \right|_{m_i} \quad . \tag{3}$$

In order to use multiplicative inverses for division, two conditions must be satisfied. First, the dividend must be exactly divisible by the divisor (i.e., the answer is an integer). Second, if the required multiplicative inverses are to exist for all moduli, the divisor must not contain any moduli as factors.

The use of multiplicative inverses to perform division is illustrated by the example in Table IV. The residue representation of the divisor 13 for moduli 5,7,9 and 4 is (3,6,4,1). The multiplicative inverse of 3 for modulus 5 is 2. Similarly, multiplicative inverses for 6,4 and 1 for their respective moduli are found to be 6,7 and 1. We now multiply the residue-representation of 156 (i.e., (1,2,3,0)) by the residue representation (2,6,7,1) for the multiplicative inverse. The result in residues is (2,5,3,0), which corresponds to the answer 12 in the decimal system.

F. Evaluation of Polynomials by Table Lookup

Tables can be constructed to perform integer polynomial transforms

		5	7	9	4
156	⟶	1	2	3	0
÷ 13	→ 3 6 4 1 →	×2	×6	×7	×1
12	⟵	2	5	3	0

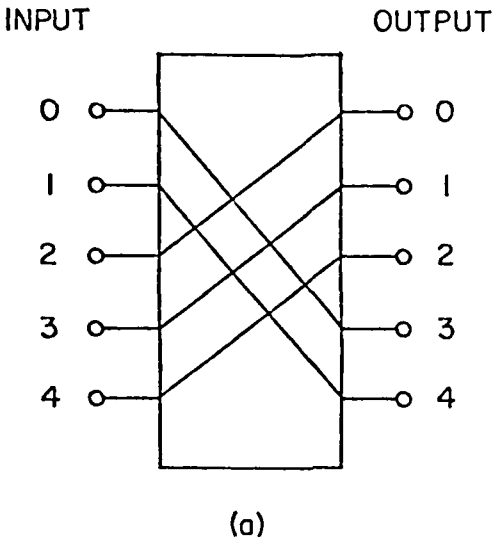
TABLE IV

(i.e., to evaluate polynomials with integer coefficients and arguments). The tables required to perform the polynomial transform $P(X) = X^2-X+1$ are shown in table V. To transform a number X , it is first encoded into residues, then each residue digit is used to index a table appropriate for its particular modulus. The decimal number 13 is (3,6,4,1) in residues. The first digit 3 is used to index the leftmost table, with the result 2. The remaining residue digits are translated in a similar manner. The result is (2,3,4,1), which is the residue representation of 157.

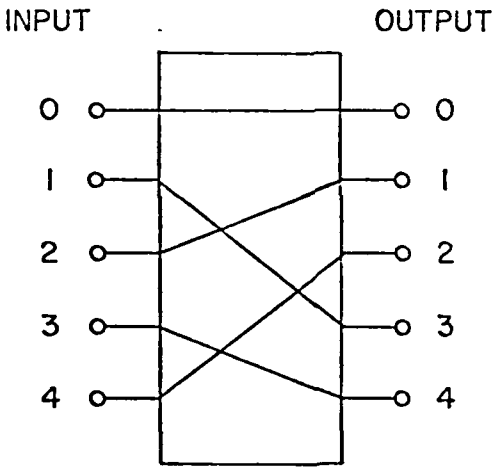
To construct a table for an integer polynomial P and for a particular modulus m_i , each possible residue digit R_{m_i} must be associated with a new digit $MOD((P(R_{m_i}),m_i))$. An example of how the table $P(X)= X^2-X+1$ is constructed for modulus 5 is shown in table VI. Tables can be constructed for integer polynomials of arbitrary degree. Tables can also be cascaded to perform polynomial transforms of polynomial transforms.

G. Residue Operations as Maps

A specific arithmetic operation, such as "addition by 3" , or "multiplication by 2", can be viewed as a mapping of the set of possible input residues (for a particular modulus) onto itself, but with a reassignment of values. We call such a reassignment of residue values a map, and illustrate the concept in Fig. 1. Suppose that the modulus of concern is 5, and that we wish to represent, for all possible input residues, the result of adding a number with residue 3. As shown in Fig. 1a, addition by 3 permutes the numbers assigned to different inputs by a cyclic shift of 3 units. Figure 1b shows the map that corresponds to multiplying any input residue by a number with residue 3 (again the modulus is 5). In this fashion



(a)



(b)

Fig. 1: Maps illustrating (a) residue addition by 3, and (b) residue multiplication by 3.

$$P(X) = X^2 - X + 1$$

$$X = 13 = (3,6,4,1)$$

Residue Digit	Moduli 5	7	9	4
0	1	1	1	1
1	1	1	1	1
2	3	3	3	3
3	→ 2	0	7	3
4	3	6	→ 4	
5		0	3	
6		→ 3	4	
7			7	
8			3	

$$P(X) = (2,3,4,1) = 157$$

TABLE V

$$P(X) = X^2 - X + 1$$

X	MOD(P(X),m_i))
0	MOD(0 - 0 + 1,5) = 1
1	MOD(1 - 1 + 1,5) = 1
2	MOD(4 - 2 + 1,5) = 3
3	MOD(9 - 3 + 1,5) = 2
4	MOD(16 - 4 + 1,5) = 3

TABLE VI

$$P(X) = X^2 - X + 1$$

X	MOD(P(X),m_i))
0	MOD(0 - 0 + 1,5) = 1
1	MOD(1 - 1 + 1,5) = 1
2	MOD(4 - 2 + 1,5) = 3
3	MOD(9 - 3 + 1,5) = 2
4	MOD(16 - 4 + 1,5) = 3

TABLE VI

various arithmetic operations can be viewed as maps.

Maps can be found which allow performance of any of the following operations: (1) conversion of a conventional number to residue form; (2) addition; (3) subtraction; (4) multiplication; (5) polynomial transformation; (6) conversion from residue form to a conventional number system. Polynomial transforms can be used as a basis for discussing all of these operations, as we shall now explain.

H. Polynomial Transforms for Encoding, Addition, Subtraction and Multiplication

Consider the conversion of a conventional number into residue form. We call this the encoding step. If the input is a decimal number, such as 19, it can be viewed as consisting of a sum of numbers, one for each digit (in this case, 10 + 9). Addition by 10 can be regarded as polynomial transform $P_{10}(X) = X + 10$, while addition by 9 requires a polynomial $P_9(X) = X + 9$. Each of these polynomials can be represented by a collection of maps, one for each modulus. For conversion of 19 to residue form, we input $X = 0$ to $P_{10}(X)$, yielding residues (0,3,1,2) for moduli 5, 7, 9 and 4. This number is then transformed by the maps for $P_9(X)$, yielding (4,5,1,3) for the output residues. If the decimal number has more than two digits, the higher digits can be handled similarly. In addition, the input need not be in decimal form. For example, binary inputs can be utilized if appropriate weighting factors are included. In general, to add the residue contribution of a particular digit to another residue number X requires a polynomial form

$$P(X) = X + \text{MOD}(d \times w, m_i) \quad , \tag{4}$$

where d represents the content of the digit in question, w is a weighting factor for that digit, (e.g., a particular power of 10, power of 2, etc.), and m_i is the i^{th} modulus.

Addition of numbers can be performed in a similar way. Suppose we wish to add the decimal numbers 19 and 87. Maps to perform conversion of 87 to residue form can be constructed and cascaded with the maps to convert 19 (discussed above). If these maps are all cascaded, the various contributions will cyclically accumulate, and for each modulus the output of the final map would be the residue of the desired sum.

As discussed above, subtraction can be viewed as addition by the complements of the residues of the number to be subtracted. To subtract the residue equivalent of a digit d from another residue X we use the polynomial

$$P(X) = m_i - \mathrm{MOD}(X + \mathrm{MOD}(d \times w, m_i), m_i) \qquad (5)$$

for each digit and for each modulus.

To perform multiplication, the multiplier must first be converted to residue form, for example by the procedure described above. The results of this conversion must be detected, and used to select residue multiplier maps. The signals representing the residues of the multiplicand must be routed through these maps to perform the desired multiplication.

i. Decoding of Residues

One of the most important operations in residue arithmetic is the conversion of a residue representation to a more conventional numeric form. Conversion is needed for sign detection, relative magnitude comparisons, and presentation of the results of calculations in forms easily interpreted by human users.

The most common method for decoding residues is by means of the Chinese Remainder Theorem.[16] This procedure is discussed in the reference, and since it is not the procedure used in what follows, we do not discuss it in any detail here. For the particular moduli 5, 7, 9 and 4 used in our

examples, this theorem yields a conversion or decoding formula

$$X = \text{MOD}(756 \times R_5 + 540 \times R_7 + 280 \times R_9 + 945 \times R_4 \ , \ 1260) \tag{6}$$

Thus, for example, the residue number $(1,1,7,2)$ becomes

$$\text{MOD}(756 \times 1 + 540 \times 1 + 280 \times 7 + 945 \times 2 \ , \ 1260)$$

$$= \text{MOD}(5146,1260) = 106. \tag{7}$$

A quick check is obtained by converting 106 to residue form. Unfortunately, this decoding procedure is extremely costly from a computational point of view, for it requires a number of conventional multiplications and additions, as well as a division. It can be seen that such a conversion process requires a large dynamic range. An alternative approach is possible in which numbers are converted back to a mixed radix system without the costly computations required above. This latter method will be explained in more detail.

Consider the following basic approach to converting from one number system to another. Suppose we wish to convert from a decimal system to a binary system. In other words, we wish to find the coefficients $a_n,a_{n-1},\ldots,a_1,a_0$ such that the decimal number X can be expressed as

$$X = a_n 2^n + a_{n-1} 2^{n-1} + \ldots + a_1 2^1 + a_0 \ . \tag{8}$$

If we divide X by 2, the remainder will clearly be a_0. If we subtract this a_0 from X, and divide by 2 again we obtain

$$X' = a_n 2^{n-1} + a_{n-1} 2^{n-2} + \ldots + a_2 2^2 + a_1 \ . \tag{9}$$

The remainder of X' after further division by 2 is a_1. We can repeat this process to find all the coefficients.

To convert from a residue number with moduli 5,7,9 and 4, we attempt to find four coefficients, a_3,a_2,a_1,a_0, such that the converted number X is given by

$$X = [a_3 \times (5\times7\times9)]+[a_2 \times (5\times7)]+[a_1 \times 5]+ a_0 \tag{10}$$

The number system that uses these coefficients is called a "mixed radix" system because the weighting factors are not powers of the same number as in the decimal or binary systems. The procedure for finding the coefficients is the same as that described above; i.e., we divide, find a remainder, subtract it, divide again, etc. If we initially know X in residue form, the remainder a_0 that would be produced by the first division (division by 5 in this example) is already available as a residue digit (R_5) and therefore need not be computed. The subtraction required is a residue subtraction, and the second division (by 5 in this case) can be accomplished by use of multiplicative inverses, as we now illustrate.

In table VII the conversion of the residue number (1,2,3,0) to mixed radix form is shown. The coefficient a_0 is the remainder of X after division by 5, and therefore is precisely equal to the residue R_5 for modulus 5. For our present example, we see that $a_0 = 1$. We now subtract a_0 from X in the residue number system, yielding a new residue number (0,1,2,3). Since the residue R_5 is 0, the new number must be exactly divisible by 5. To divide by 5 we multiply by the multiplicative inverses of 5 for the remaining moduli 7,9 and 4. Since 5 is relatively prime with respect to these numbers, the inverses exist and are given by 3,2 and 1. Multiplication of the new residues $(R_7',R_9',R_4') = (1,2,3)$ by (3,2,1) in the residue system yields (3,4,3), from which we can obtain

	moduli	5	7	9	4
$a_o \leftarrow$		[1]	2	3	0
Subtract $a_o=1$		1	1	1	1
		0	1	2	3
Multiply by $\left\|\frac{1}{5}\right\|$			3	2	1
$a_1 \leftarrow$			[3]	4	3
Subtract $a_1=3$			3	3	3
			0	1	0
Multiply by $\left\|\frac{1}{7}\right\|$				4	3
$a_2 \leftarrow$				[4]	0
Subtract $a_2=4$				4	4
				0	0
Multiply by $\left\|\frac{1}{9}\right\|$					1
$a_3 \leftarrow$					[0]

$$Z = 0(5\times7\times9) + 4(5\times7) + 3(5) + 1 = 156$$

TABLE VII

a_1 as the modulus 7 digit. The process is continued until all the coefficients have been found.

The decoding procedure described above can be carried out by means of maps. If R_{m_n} represents the residue for modulus m_n, the residue digits for the remaining moduli are transformed by the polynomial

$$P(X) = (X-R_{m_n}) \times Z \qquad (11)$$

where Z is the smallest number such that $MOD(Z \times m_n, m_i) = 1$. This operation corresponds to the subtraction and division steps of the conversion process. One of the remaining residues is then selected in place of R_{m_n}, and the process is repeated until all the residue digits have been used. The values of the residues obtained by this sequential process are the desired mixed radix coefficients. The weighting factor for each coefficient is the product of 1 and all the moduli that were used previously.

At first glance, the mixed radix conversion process appears just as complex and time consuming as the "carry" operations in conventional number systems. However, it should be noted that many calculations can be performed before this fixed overhead is incurred. Thus, the more operations there are to be performed, the more advantageous the residue system will be.

The conversion process appears to be sequential, but in fact it can be nicely pipelined to increase the throughput of the system. Suppose the processors for moduli 5, 7, 9 and 4 are completely independent units. Further, suppose that we wish to perform several different calculations, which we represent symbolically by A, B, C, etc. Calculation A is first given to the modulus 5 processor, the proper maps are selected, and the answer detected. Calculation A and the modulus 5 result are passed to the modulus 7 unit. Using both A and the modulus 5 result, maps are

selected in the modulus 7 unit. The modulus 5 result, the modulus 7 result and the calculation A are passed to the modulus 9 processor. The two results passed are two of the desired mixed radix coefficients. While all this is occurring, the modulus 5 unit, having finished calculation A , can be working on calculation B. This assembly-line approach can continue through all the modulus processing units, with little or no idle time. The calculations emerge with the desired mixed radix coefficients.

J. Example

As a demonstration of the use of maps for residue arithmetic operations, we illustrate with a specific example. Fig. 2 illustrates the addition of 19 to 87, subtraction of 99, and transformation by the polynomial $P(X) = X^2 - X+1$. The result should be 43, which in residues (again with bases 5, 7, 9 and 4) is (3,1,7,3).

The top row of Fig. 2 represents the modulus 5 operations, the second modulus 7, etc. The two leftmost columns encode 19 into its residue representation, the next two add 87 in residues, the fourth and fifth subtract 99, and the last column performs the polynomial transform. The paths are marked by arrows. The signals emerge at ports 3, 1, 7, and 3, which will be shown to translate to 43.

Decoding of this result is shown in Fig. 3. The coefficient a_0 is taken as the detected modulus 5 result and is thus 3. Maps for other signals corresponding to the polynomial $P(X) = (X-R_5) \times Z$ are selected. R_5 in this case is 3 and Z for modulus 7 is 3 since $MOD(3\times5,7) = 1$. Thus for modulus 7 a map corresponding to $P(X)= (X-3) \times 3$ must be selected. Similarly, maps for $P(X) = (X-3) \times 2$ and $P(X) = (X-3) \times 1$

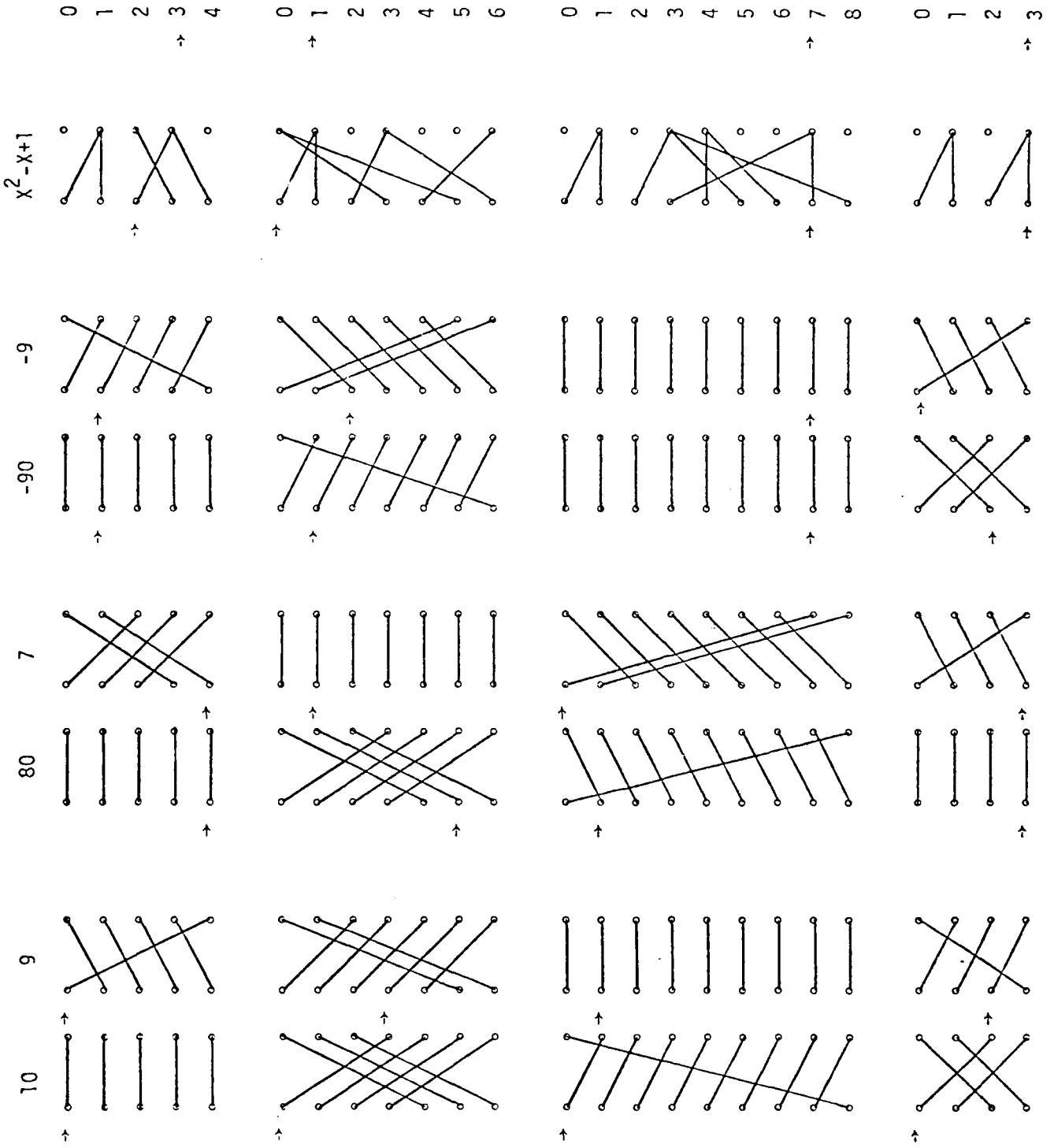


Fig. 2: Example of a sequence of arithmetic operations performed with maps.

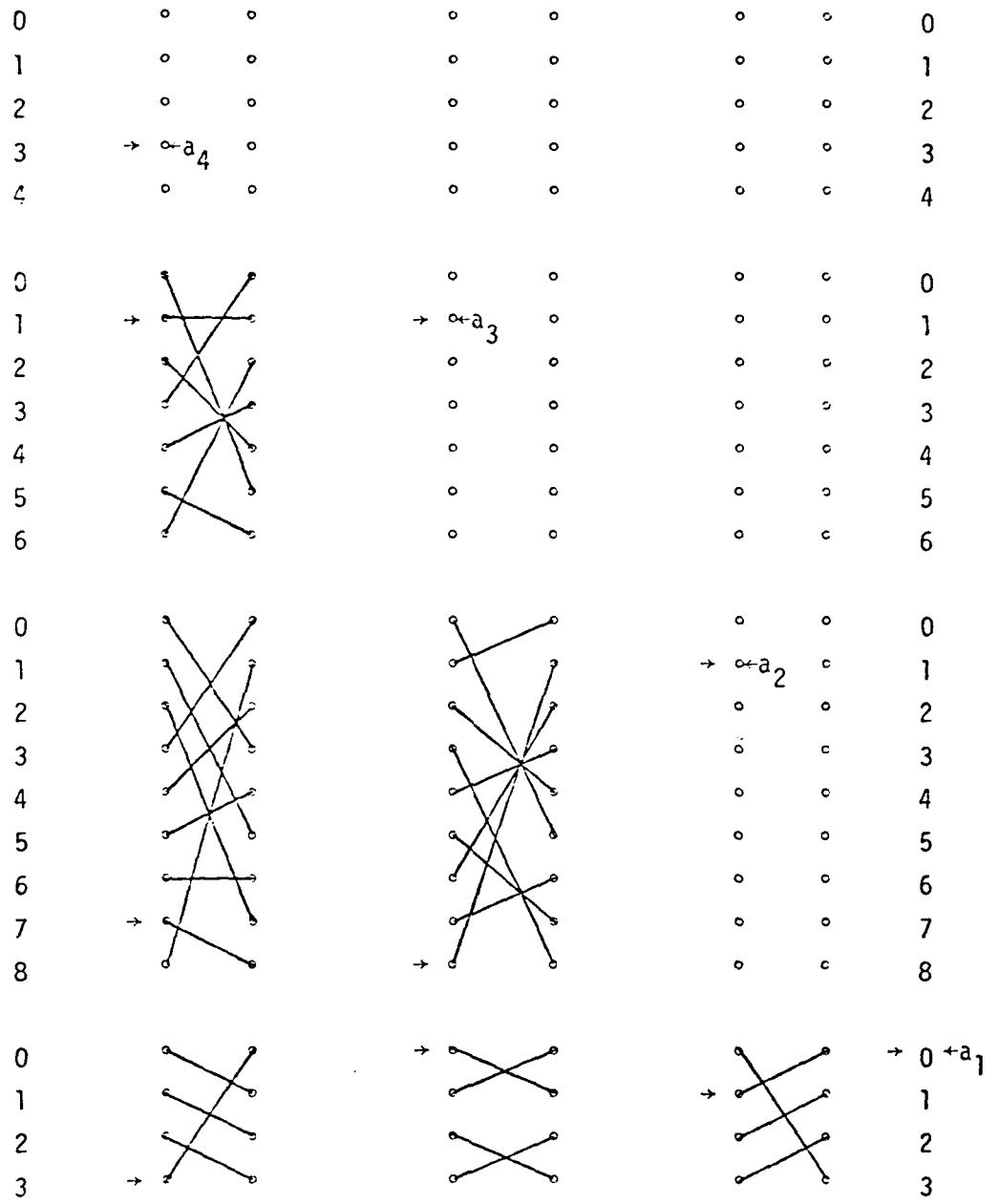


Fig. 3: Converting the residue result to a mixed radix representation

must be selected for moduli 9 and 4, respectively. The modulus 7 signal then passes through the appropriate map and is detected as a 1. The remaining signals are then routed through maps corresponding to $P(X) = (X-1) \times Z$, where Z for modulus 9 is 4, and for modulus 4 is 3. The modulus 9 signal is then detected as a 1. The remaining modulus 4 signal is then routed through a map corresponding to $P(X) = (X-1) \times 1$ and detected as 0. Thus the decoded mixed radix number is (a_3, a_2, a_1, a_o) where the weighting factors associated with the successive digits are 315, 35, 5 and 1 (i.e., $9\times7\times5\times1$, $7\times5\times1$, 5×1, and 1). Since a_3 is 0, the number is within the range 0 to $M/2-1$ and is thus positive; hence the answer is $1\times35 + 1\times5 + 3\times1 = 43$. If a_3 were either 2 or 3 then the answer would be negative and given by $a_3\times315 + a_2\times35 + a_1\times5 + a_0\times1 - 1260$.

III. OPTICAL IMPLEMENTATION OF RESIDUE ARITHMETIC OPERATIONS

A. *Physical Representation of Residues*

In searching for a useful match between optics and residue arithmetic, a fundamental decision must be made regarding what physical property of light will be used to represent residue numbers. The intensity, polarization or phase[17] of an optical signal might be chosen for this representation.

We have chosen to represent residue numbers by the spatial position of a spot of light. Thus to represent the residue of a number for modulus 5, five possible spatial positions would be allocated, and one of these positions would contain a spot of light. The position of this spot in the array of five possible locations indicates the residue number being represented.

Our choice of an "on-off" representation at each of several possible spatial locations is dictated by the serious consequences of making even small errors in the residue number system. An error of one unit in only one of the several residues can lead to a very large error in the final

decoded number. Hence for reasons of reliability it is desirable to make all decisions binary.

B. Spatial Maps for Performing Residue Operations

As discussed in Section II, the various operations required in a residue arithmetic unit (e.g., encoding, decoding, addition, etc.) can be implemented by means of maps which permute the "states" representing residue numbers. For our chosen representation of residues, the physical states are spatial positions of a light beam. Hence the maps of interest must be capable of performing spatial permutations of the possible positions of a light beam.

It is useful to distinguish between two types of processing operations of interest -- fixed operations and changeable operations. As an example of a fixed operation, consider a unit that always adds 3 to an incoming residue. Such an operation can be realized by a map that always introduces the same spatial permutation. On the other hand, a more general unit may be desired which adds an incoming residue to a number that can change with time. The latter unit must be implemented by means of a changeable map, which introduces any of a number of possible permutations, depending on instructions sent to it. We shall shortly consider various optical methods for realizing both fixed and changeable maps.

In considering methods for constructing changeable maps, it is important to realize that there are many different ways to achieve the desired mapping units, even after the physical means for implementation has been decided upon. For example, let the particular modulus of interest be m_i , and suppose that we wish to add any two-digit decimal number

to the incoming residue, obtaining the residue of the sum at the output. If the two-digit number is first converted to a residue with respect to modulus m_i, then that residue can be used to select one of the m_i distinct maps that are possible. Alternatively, it may be preferable to apply the unconverted two-digit number directly to the mapping device. In this case, with a single changeable map, 100 permutations of the input ports must be realized, one for each of the possible two digit numbers. However, only m_i of these permutations are different. Clearly, to minimize the number of different maps needed, it is preferable to use the residue of the two-digit number for map selection.

As an alternative approach, we could cascade two maps, one of which must realize the permutations required for addition by the "units" digit, and the second of which must realize the permutations required for addition by the "tens" digit.

Carrying this decomposition further, we could represent the incoming two digit decimal number by a seven digit binary number. The desired operation could then be achieved by cascading seven changeable maps, each of which is capable of realizing only the two permutations appropriate for that particular bit. In all cases, the mapping devices must have m_i input ports and m_i output ports. As will be demonstrated later, the binary decomposition is especially useful for realizing certain types of changeable maps.

C. Some Optical Methods for Realizing Fixed Maps

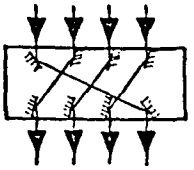
A wide variety of methods for realizing fixed spatial maps can be imagined. What follows is surely not an exhaustive discussion of the possibilities; the reader may conceive of other interesting possibilities.

Figure 4 illustrates a number of possible methods for making a fixed permutation device. In part (a) , small reflectors are used to re-direct the incident light beams to their new locations. The device shown in part (b) uses prisms to accomplish the same result. In part (c), the inter-connections are made by means of optical waveguides or fibers. Part (d) illustrates the use of thick gratings operating in the Bragg regime. The orientation of the gratings must be chosen to satisfy the Bragg condition in each case. All four approaches are amenable to integration in thin planar devices.

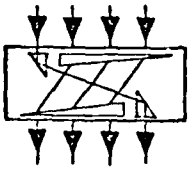
An alternative method for performing fixed permutations is shown in Fig. 5(a); we refer to this approach as the "permutation matrix" method. Light incident on any one of the input ports is spread vertically to fill the height of the matrix mask M. However, in the horizontal direction, lens combination L_2 images the input port to form a vertical column of light incident on the matrix mask. Lens combination L_3 images the matrix mask in the vertical direction onto the vertically stacked output ports, while integrating or adding the light transmitted across horizontal rows of the mask. Use of a properly chosen mask, in which each vertical column and horizontal row contain only one transparent cell, can yield any desired spatial permutation.

This method of performing permutations is very wasteful of light power, and therefore is not of direct practical interest. However, it is conceptually useful for it leads to other interesting approaches to performing arithmetic operations. For example, suppose we wish to construct a device which will multiply any incoming residue by a fixed number N , and output to a detector array a binary optical pattern representing the binary number representation of the residue of the product. The incoming residue number is represented by the spatial position of a light

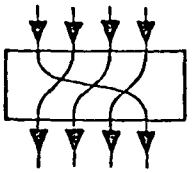
(a)



(b)



(c)



(d)

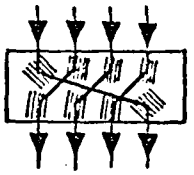


Fig. 4: Possible approaches to constructing fixed maps using (a) mirrors, (b) prisms, (c) optical waveguides or fibers, and (d) gratings.

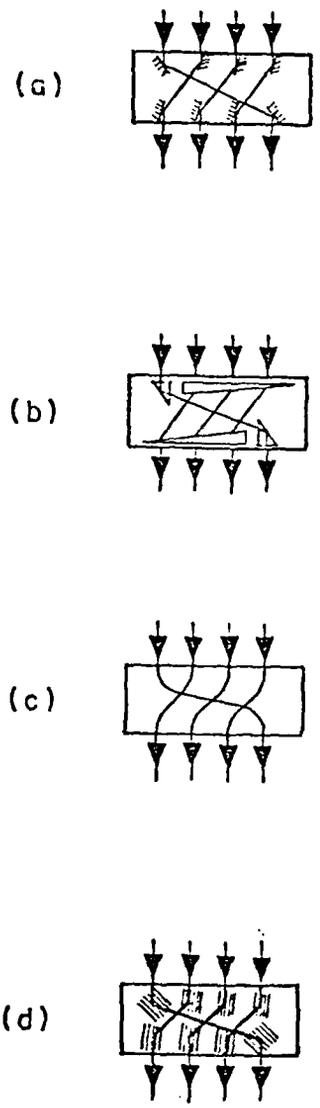


Fig. 4: Possible approaches to constructing fixed maps using (a) mirrors, (b) prisms, (c) optical waveguides or fibers, and (d) gratings.

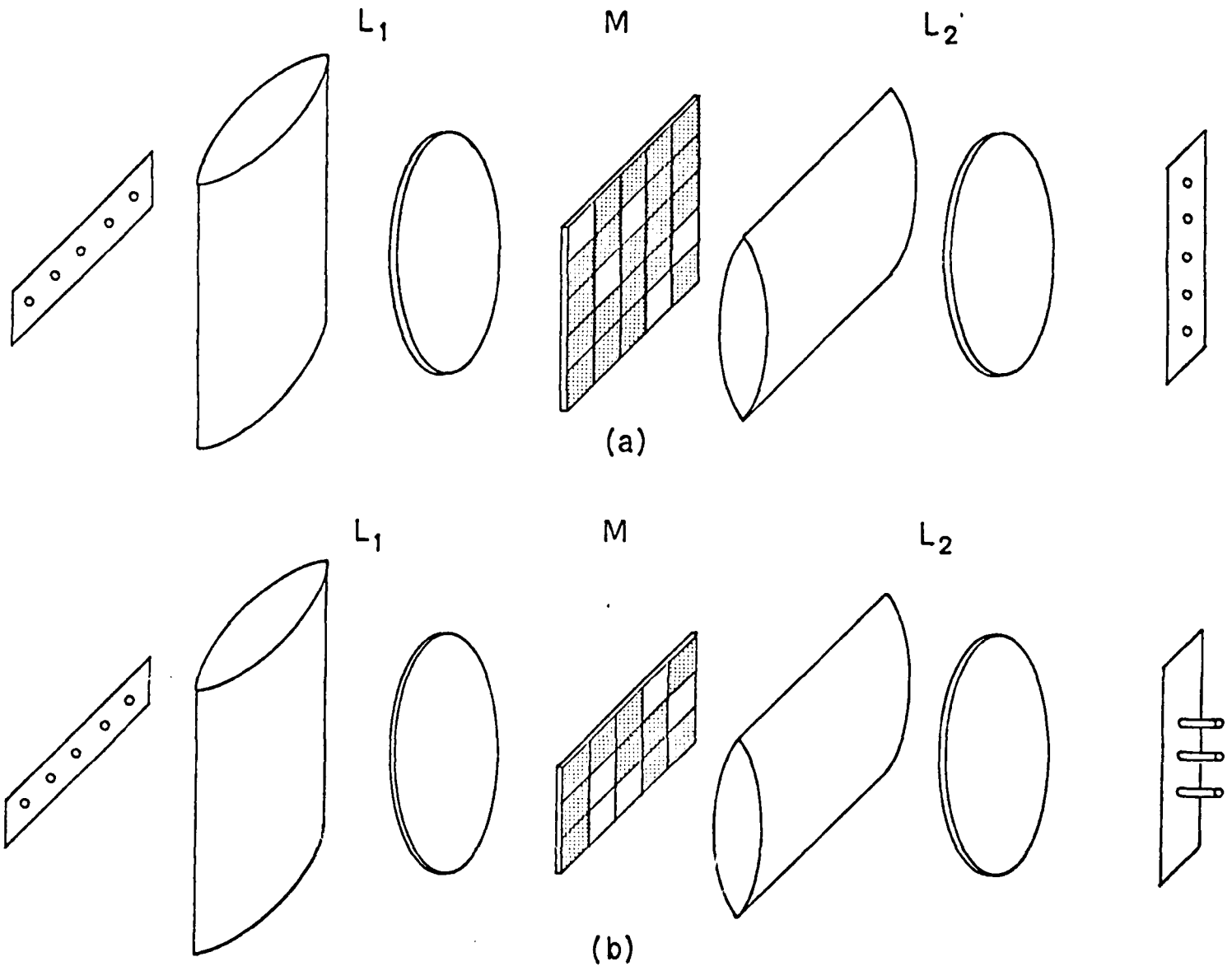


Fig. 5: (a) spatial permutations by means of a "permutation matrix"
(b) residue multiplication with a binary output

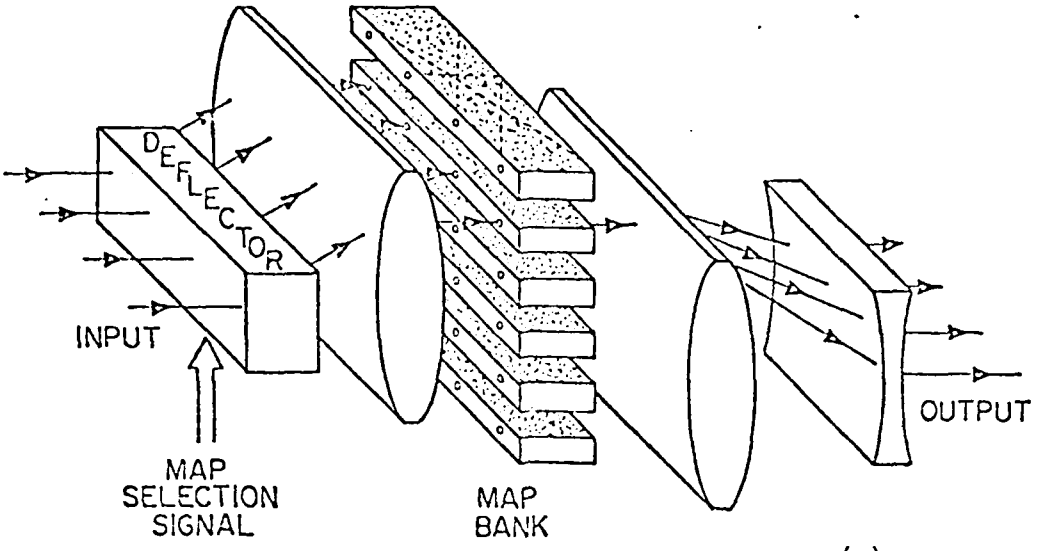
beam, as before. However, as shown in Fig. 5(b), each vertical column of the mask contains a binary code representing the residue of the product of the incoming number and the fixed number N. The output binary number is detected by a vertical detector array having as many elements as are required for the binary representation of the modulus in question. Thus for modulus 5, three output detectors are required, while for a modulus of 31, five output detectors are needed.

While methods for performing fixed operations are of interest, far more important is the problem of performing changeable operations. We therefore turn attention to the problem of constructing changeable maps.

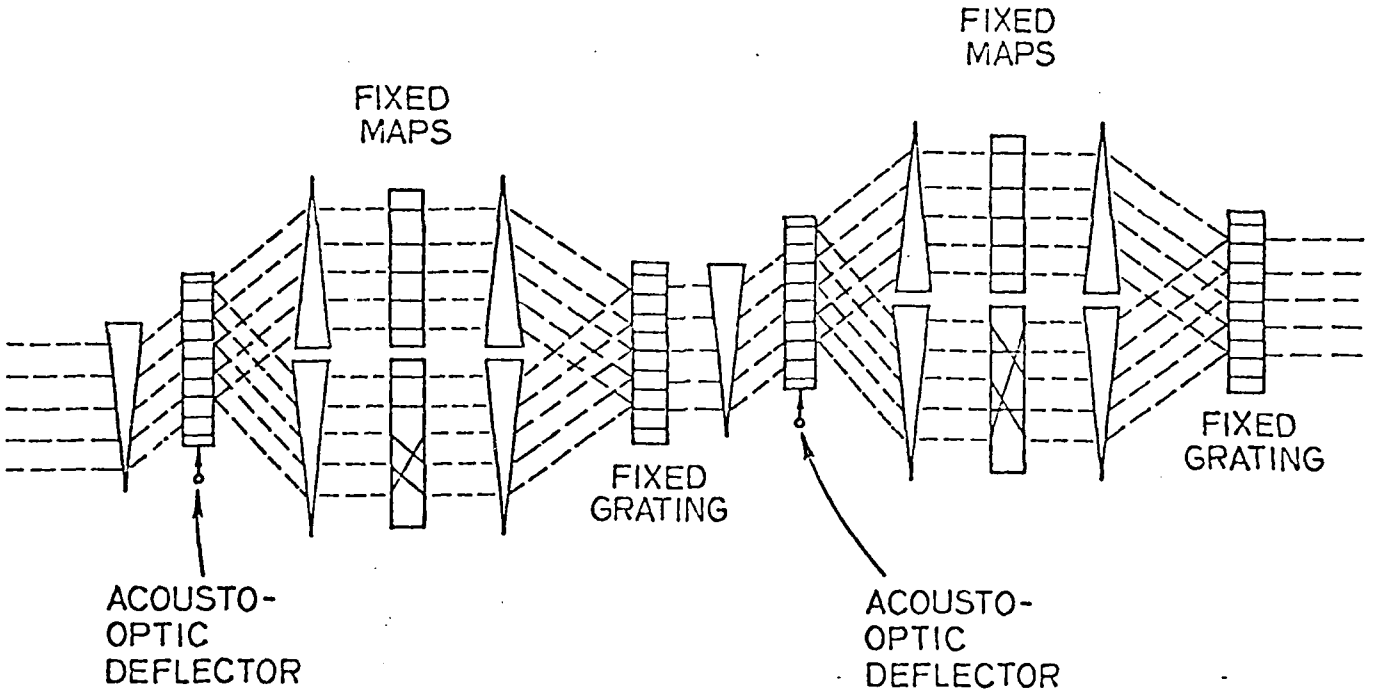
D. Realization of Changeable Maps by Means of Map-Banks

In order to construct an optical device or subsystem capable of performing variable or programmable arithmetic operations, some method for rapidly selecting or configuring maps must be used.

One approach, illustrated in Fig. 6, is to use a rapidly addressable bank of maps for each modulus. The map appropriate for a particular desired operation is selected by means of an optical deflector. As shown in part (a) of the figure, an acousto-optic or electro-optic deflector could be used for the map selection process. Suppose we wish to add an incoming residue number, represented by the spatial position of a spot of light, to a second residue number that drives the map selector. If the modulus is m_i, then m_i different maps must be used, each with m_i input and output ports. Alternatively, as shown in part (b) of the figure, the number to be added to the incoming optical residue could be represented in binary form, each bit controlling a selector



(a)



PLANAR BINARY ACOUSTO-OPTIC MAPS

(b)

Fig.6: Optical realization of a changeable map by means of (a) a single deflector addressing many maps, and (b) multiple deflectors addressing only two maps each.

which need address one of only two possible maps. A cascade of $m = \lceil \log_2 m_i \rceil$ maps pairs is required, where m is the smallest integer greater than or equal to $\log_2 m_i$ (in other words, the number of binary bits needed to represent m_i). Each map again has m_i input and output ports.

The use of a cascade of map-pairs, as shown in Fig. 6(b), is believed to be preferable to the use of a single large map bank. One reason for this preference comes from speed considerations. The speed-capacity product of an acousto-optic deflector (i.e., the number of different angular positions that can be randomly accessed per second) is known to be numerically equal to the bandwidth of the acoustic transducer. If we assume a bandwidth of 1 GHz and a modulus of 31 (i.e., a bank of 31 different maps), a time of about 30 nanoseconds will be required to select a chosen map in the bank. On the other hand, a sequence of 5 binary acousto-optic deflectors, each having 1 GHz bandwidth and all addressed in parallel, would require only 2 nanoseconds for selection of any desired map. In addition, it should be noted that a structure consisting of a sequence of pairs of maps can be realized in a planar geometry more easily than can a single larger map bank. Hence the possibilities for integration on a planar substrate are greater for a sequence of map pairs.

The adder described above can easily be changed to a subtractor. For example, if we wish to subtract the incoming number represented optically from the number represented electronically, we simply precede the adder by a fixed optical map that complements the incoming residue with respect to its modulus. The rest of the unit remains unchanged.

For multiplication, similar kinds of map-banks can be used.

However, residues 0 must be treated in a special way, since multiplication by zero always yields zero. It is straightforward to by-pass the map-bank when one of the inputs is zero, providing a proper zero output.

E. A Changeable Map for Cyclic Permutations

An entirely different kind of approach to constructing a changeable map is illustrated in Fig. 7. The modulus in this example is 5, so there are 5 input optical ports and 5 output optical ports. Suppose that a 10 bit number is to be added to the incoming optical residue. The map consists of 10 subunits, each subunit consisting of a planar waveguide with three "switching channels" represented by three lines on the figure. A light beam incident at any point on a switching channel is assumed to be either transmitted or reflected, depending on the electrical signal applied to the switch.

One of the three channels in each subunit is always along the diagonal of the subunit. If the binary number applied to that subunit is 0, then the diagonal channel (labeled "0") is activated as the reflector and no permutation of the ports occurs. However, if the binary number applied to the subunit is 1, the diagonal channel is transmissive, and the off-diagnonal channels (labeled "1") are reflective. As a consequence, when a binary 1 is applied, a cyclic permutation of the input ports occurs. The positioning of the off-diagonal channels is dependent on the different weights (powers of 2) that must be associated with each binary digit. For example, the tenth bit, for which a "1" represents 512, must introduce a net offset of 2 for modulus 5, since 2 is the residue of 512.

The first subunit corresponds to the most significant bit. After reflection from the diagonal or off diagonal channels, the reflected

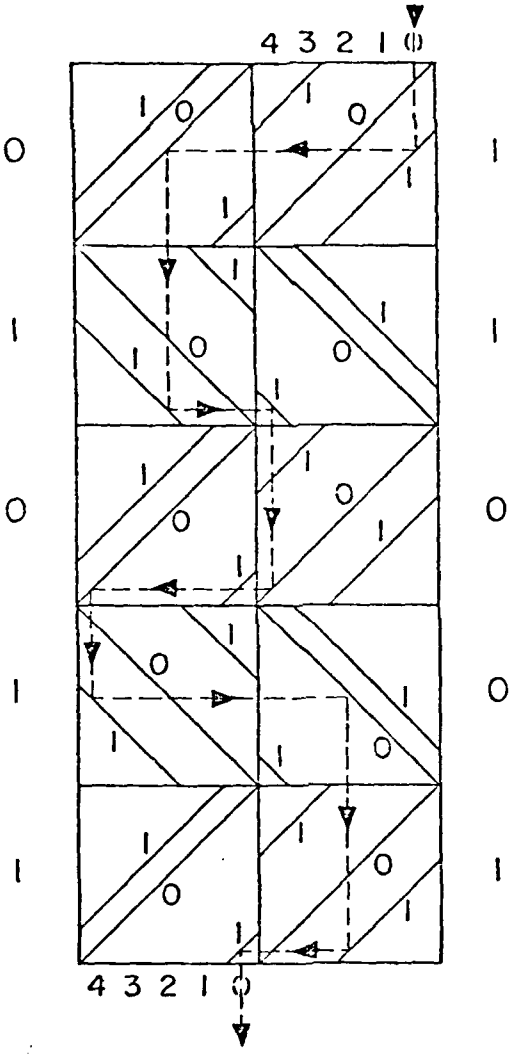


Fig. 7: Changeable map realized by a series of integrated optic switches

beam enters the second subunit, etc. The beam ultimately emerges from one of the ports of the tenth subunit, the number of this port representing the residue of the sum of the incoming residue and the input binary number, for a modulus of 5 in the case illustrated.

Residue addition is possible using the device described above because such addition always requires a cyclic map (c.f., Fig. 1a). However, other operations can be performed using such a device as well. Subtraction simply requires a fixed (non-cyclic) map to complement the incoming residue, followed by a cyclic map. Digital-to-residue conversion of the binary number applied to the device is realized by simply applying the optical input at port 0. Light emerges from the output port corresponding to the residue of the applied binary number. In the case illustrated in Fig. 7, the binary number applied to the device corresponds to decimal number 715. Light exits from the device at port 0, indicating that the residue of this number is zero for modulus 5.

Multiplication poses a more difficult challenge because it cannot be performed by a cyclic map. However, it has been pointed out to us[18] that a changeable cyclic map can be used as the heart of a multiplier provided the modulus is a prime number. As illustrated in Fig. 8 for a modulus of 5, if we cascade a fixed non-cyclic map with a changeable cyclic map and a second fixed non-cyclic map, we can perform residue multiplication by any input number other than zero. (The cases of residue input of zero and zero multiplier can be handled separately). The decomposition of multiplication into three maps is much analogous to first finding the logarithms of the two numbers to be multiplied,

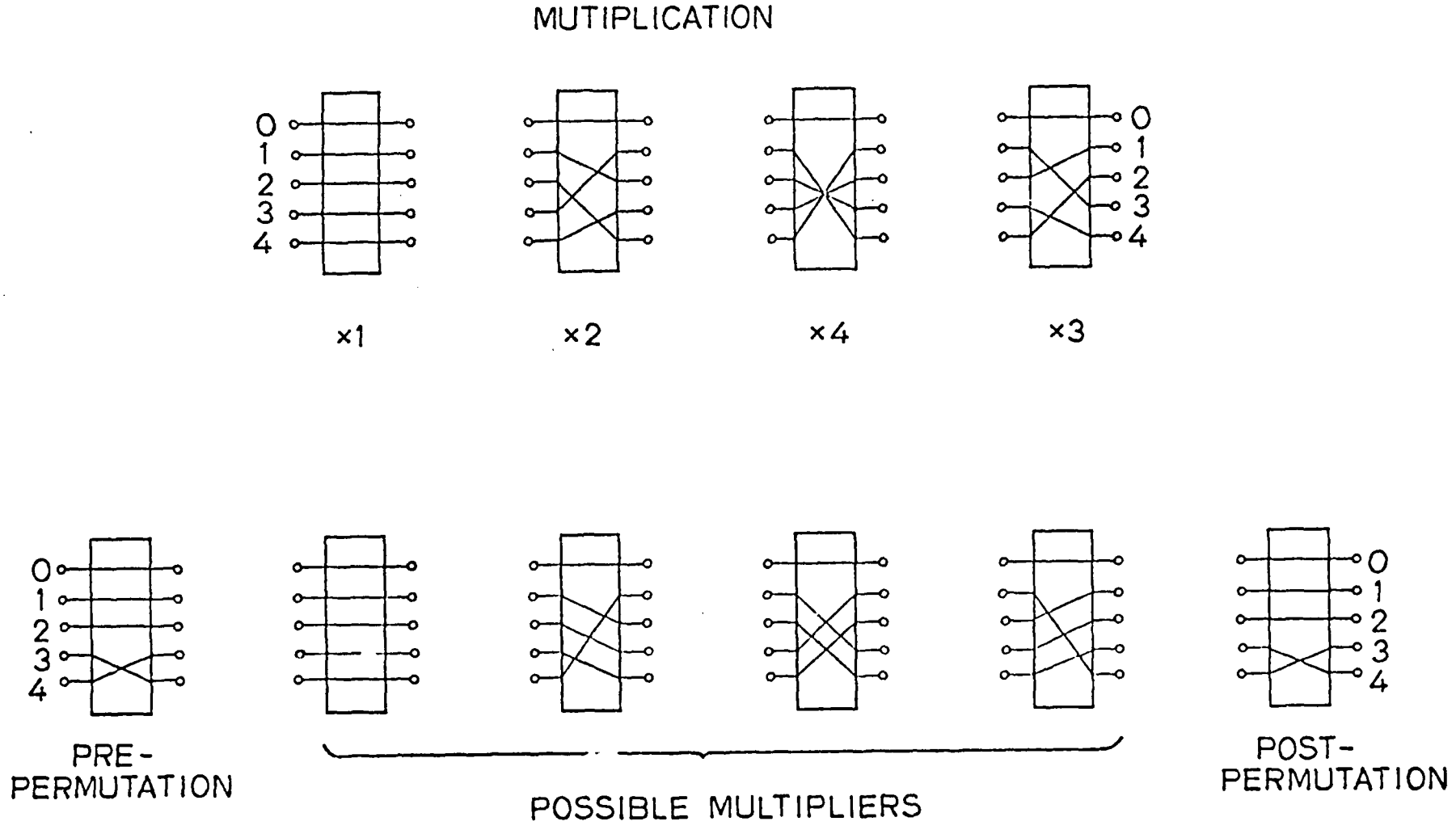


Fig. 8: Cascade of non-cyclic, cyclic and non-cyclic maps to perform residue multiplication

adding logarithms, and then finding the antilog of the result. The restriction to moduli that are absolute primes (rather than relative primes) does not appreciably increase the complexity of the arithmetic unit.

We conclude that electro-optic devices for performing cyclic permutations can serve as the basic building blocks of an arithmetic unit based on residue arithmetic. One promising approach to constructing the switches required in the device of Fig. 7 is described in reference 19 and is illustrated in Fig. 9. A thin dielectric channel is implanted in a y-cut $LiNbO_3$ planar waveguide. The refractive index of the channel is normally slightly smaller than that of $LiNbO_3$. Parallel metal electrodes are deposited along the edges of the channel. If no voltage is applied across the electrodes, light incident at an angle greater than the critical angle will suffer total internal reflection. When a voltage is applied across the electrodes, the refractive index of the channel is raised, the critical angle increased, and the angle of incidence now being smaller than the critical angle, the light beam is transmitted through the channel. Such switches have been predicted to have bandwidths of several GHz.[19]

The chief drawback to the above approach to construction of the desired switches is the small angle required between the switching channel and the incident light beam. Rather than having 45° incidence of light on the channel, as shown in Fig. 7, the angle of incidence (with respect to the channel) cannot exceed 10° in present practice. As a consequence the overall switching device must be elongated considerably in the vertical direction, as shown in Fig. 10.

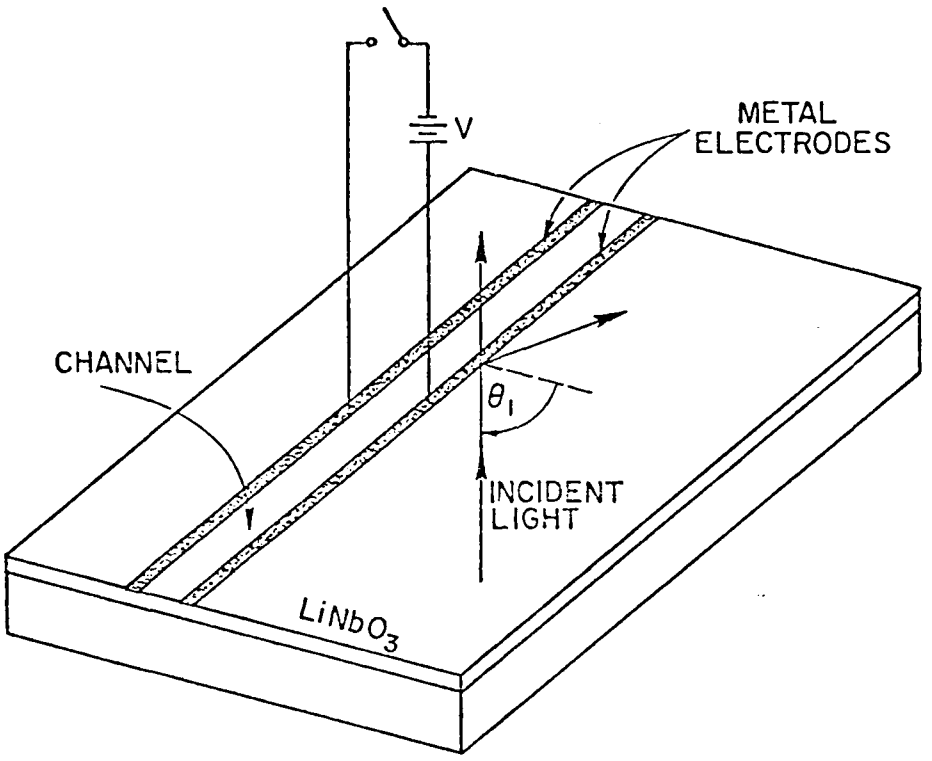


Fig. 9: Integrated optic switching device

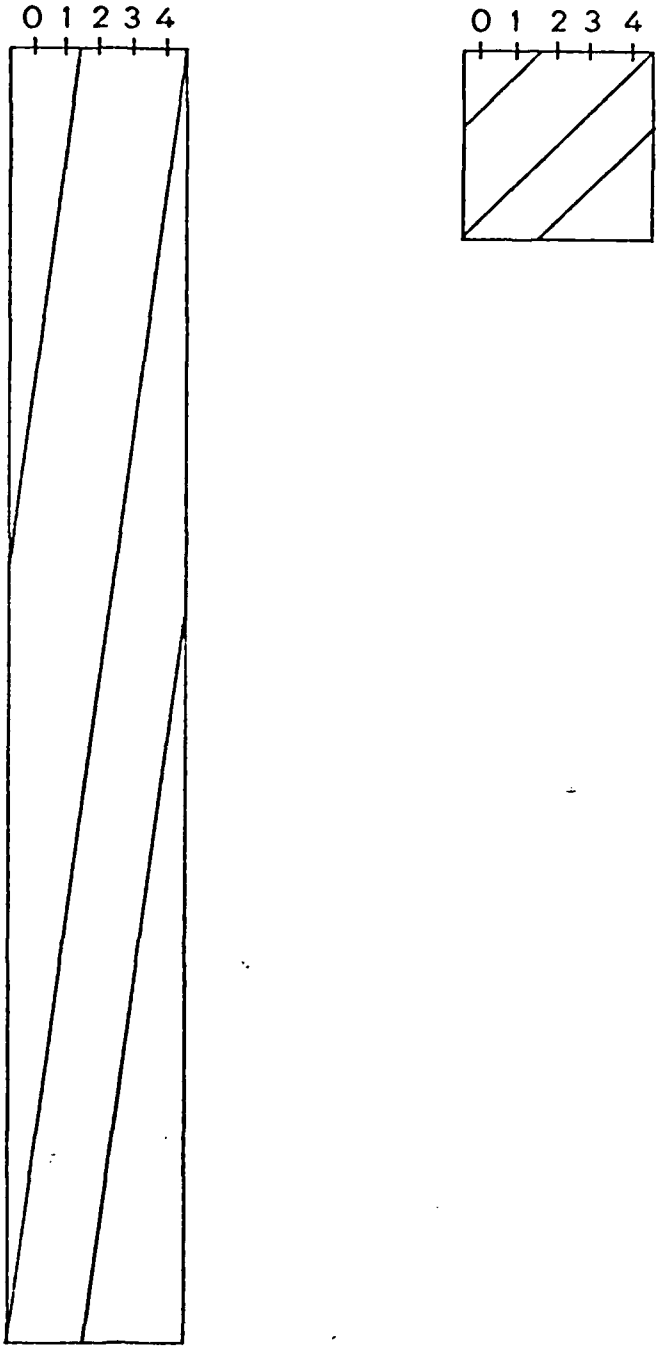


Fig. 10: Elongation of the switching unit due to the small angle of incidence required.

An alternative approach to making an electro-optic cyclic permutator is by means of integrated optic waveguide couplers, as shown in Fig. 11. The structure consists of an array of contiguous waveguides with couplers that can be activated electrically. When a coupler is switched on, the signals propagating in adjacent guides are interchanged. In other words, a simple permutation of adjacent ports is introduced. By constructing an appropriate geometrical array of couplers, cyclic permutations of all kinds can be synthesized. Non-cyclic permutations are also possible. Each horizontal line on the waveguide structure of Fig. 11 represents an electrically addressable coupler. Optical waveguide couplers reported on by Chen, Tangonan and Lee[23] of the Hughes Research Laboratories are attractive candidates for this kind of device. Switching times on the order of a nanosecond appear possible.

IV. A MATRIX-VECTOR MULTIPLIER

One of the most fundamental signal processing operations is the multiplication of an input vector and a stored matrix to produce an output vector. The input vector has components that generally represent samples of some data to be processed, while the output vector has components that are samples of the processed data. The matrix determines the form of the processing transformation that is applied. This type of operation includes discrete Fourier transforms, Hadamard transforms, correlation with multiple stored references, and general linear space-variant filtering operations.

A. Basic Concept of the Residue Matrix-Vector Multiplier

We restrict attention to data processing problems that require multiplication of an input vector $\vec{f}$ times a stored matrix $\underline{H}$ to produce an output vector $\vec{g}$. Here $\vec{f}$ is a P-element column vector

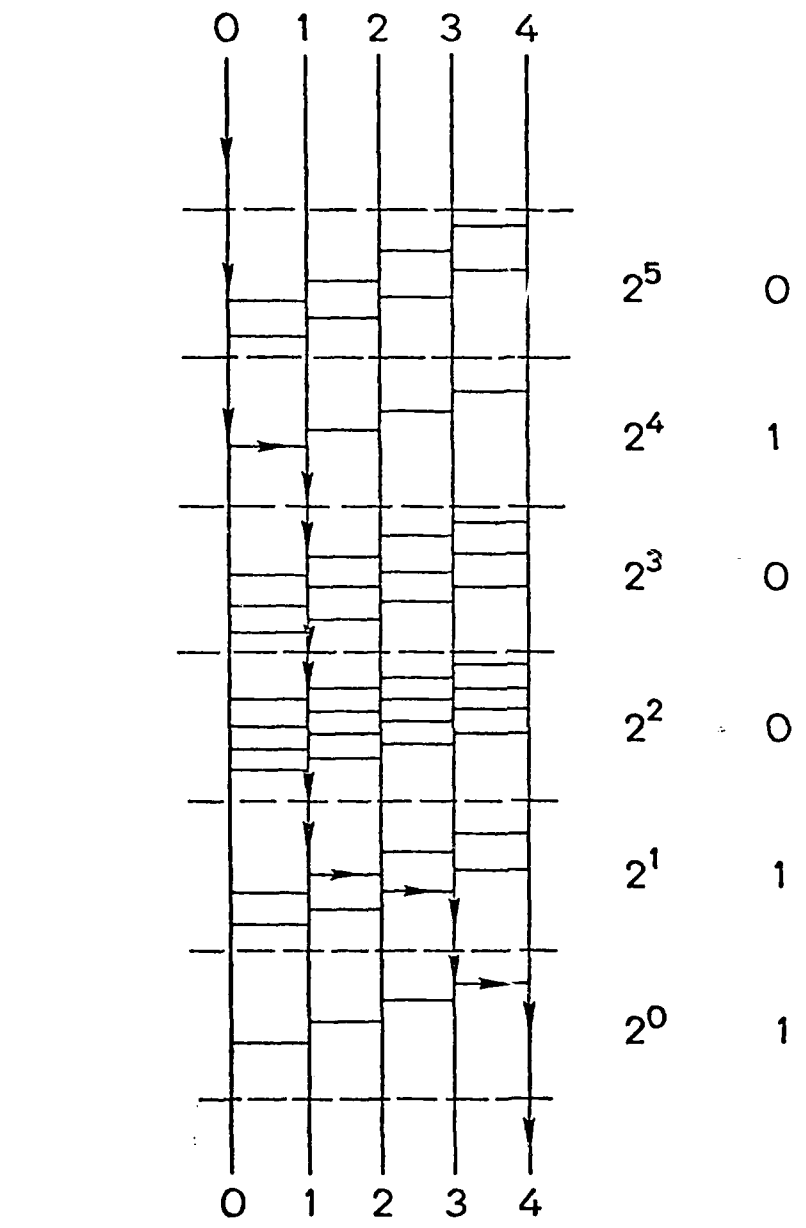


Fig. 11: Integrated optic waveguide couplers for performing permutations.

$$\vec{f} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_{P-1} \end{bmatrix} , \tag{12}$$

$\underline{H}$ is a P×Q matrix

$$\underline{H} = \begin{bmatrix} h_{0,0} & h_{0,1} & \cdots & h_{0,P-1} \\ \vdots & & & \\ h_{Q-1,0} & h_{Q-1,1} & & h_{Q-1,P-1} \end{bmatrix} \tag{13}$$

and $\vec{g}$ is a Q-element column vector given by

$$\vec{g} = \underline{H}\vec{f} \quad . \tag{14}$$

While this operation may at first appear somewhat specialized, it actually embraces a multitude of widely used operations. We do impose the restriction that the elements of $\vec{f}$ and $\underline{H}$ must be represented by integers. This restriction may be viewed as a kind of quantization of the data, which achieves an accuracy of representation limited only by how large a scaling factor is used. The size of the scaling factor is in turn limited by the dynamic range of the system.

To find a single component g_k of the output vector $\vec{g}$, it is necessary to multiply the P input numbers $f_0, f_1, \ldots, f_{P-1}$ by a set of P stored numbers $h_{k,0}, h_{k,1}, \ldots, h_{k,P-1}$, and to add the resulting P products,

$$g_k = \sum_{p=0}^{P-1} h_{k,p} f_p \quad . \tag{15}$$

The fact that this operation requires only sums of products suggests that

it may be well suited to a residue arithmetic approach. Figure 12 shows the block diagram of a residue matrix multiplier. In general, the input signals might be in either analog or digital form, and must first be converted to a residue representation. The multiplications and additions implicit in Eq.(8) must then be performed in the residue number system for each of the chosen moduli. Finally, the output residues must be combined to produce a total output, which presumably will be in digital form because of the high accuracy required.

It is highly desirable, for the purposes of processing speed, to enter the elements of the input vector $\vec{f}$ in parallel and to extract the elements of $\vec{g}$ in parallel. Our proposed solution is to have a separate opto-electronic unit for each modulus, to enter data in parallel to each such unit, and finally to extract all components of the output vector in parallel.

Because of the inherent parallelism afforded by optics, it should be a goal to perform all multiplications required for each modulus in parallel. Additions for any one component of the output vector (for any single modulus) can be performed serially at high speed, and parallel summing units should be used for all components of the output vector. The conversion from the residue answers to Q digital numbers representing the output components of $\vec{g}$ should also be performed at high speed with Q parallel channels.

B. An Opto-Electronic Implementation of the Matrix-Vector Multiplier

A wide variety of different opto-electronic versions of a residue matrix-vector multiplier can be envisioned. Here we discuss only one particular approach to the problem. The construction of such a system

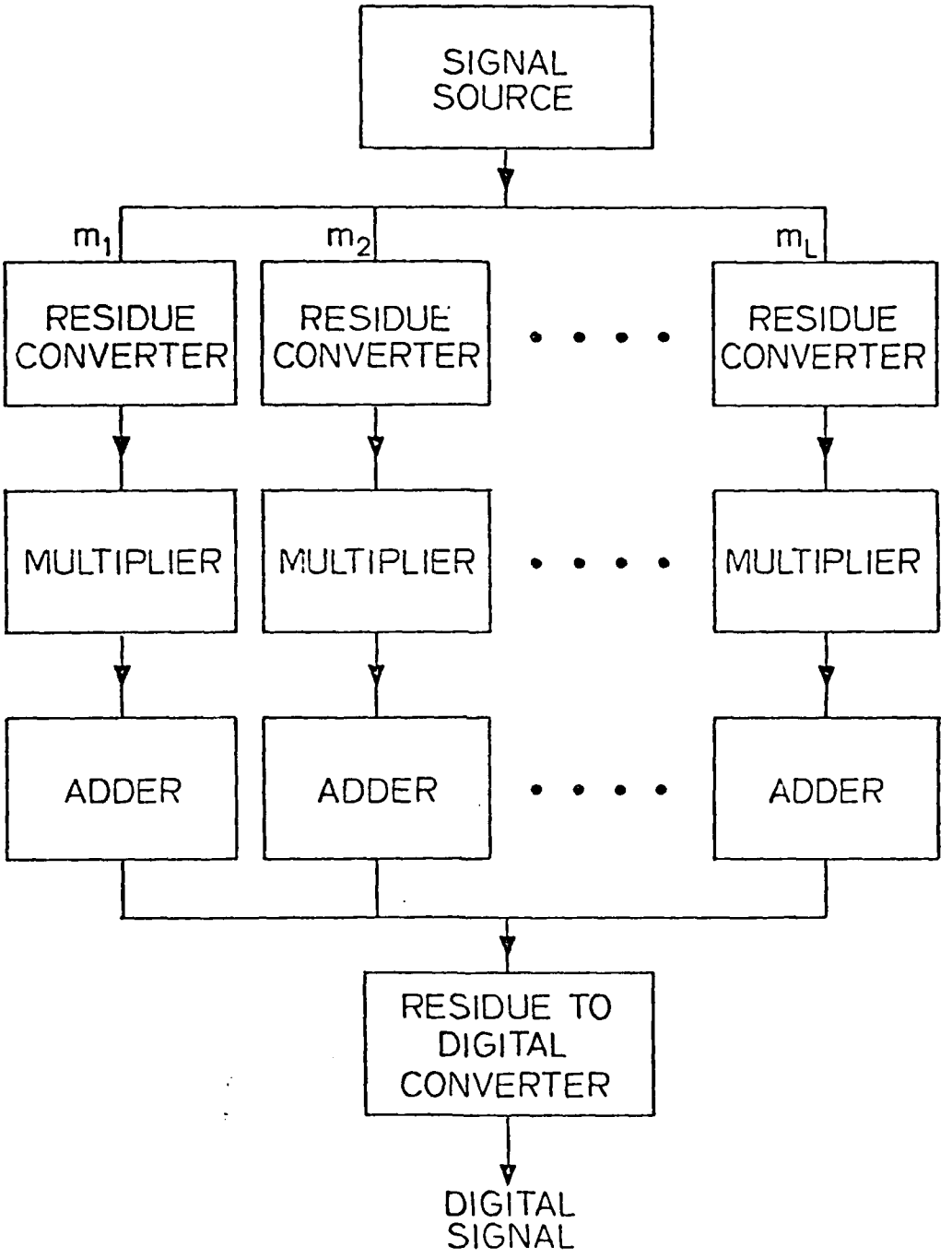


Fig. 12: Block diagram of residue matrix multiplier

requires a means for converting inputs to residue form, a means of performing many residue multiplications in parallel, a means for performing many sums of products, and finally a means for decoding the residue answers to a mixed radix number.

C. Conversion to Residues

Conversion of inputs to residue form can be accomplished by means of the integrated optic devices discussed in the previous section. All components of the vector $\vec{f}$ are applied in parallel to an array of converters. Laser diodes or LED's illuminate the "zero" input ports of all the converters, and light emerges from the output ports corresponding to the proper residues.

D. Multiplications

We assume that each separate modulus is handled by a separate unit, and we consider now only one of these units. The vector $\vec{f}$ is input in residue form as a series of light spots, each component of $\vec{f}$ being represented by one light spot in any of m_i positions, where m_i is the modulus. There are a total of P light spots, where P is the number of elements in $\vec{f}$. The next operation to be performed is the multiplication of each element f_p of $\vec{f}$ by the elements h_{pq} $(q = 0,1,\ldots,Q-1)$ of the matrix $\underline{H}$. A single residue product $h_{pq}f_p$ can be performed by the system shown in Fig. 5(b), which was drawn for a modulus of 5. A light beam enters at one of 5 possible spatial positions, and is mapped by the initial optics L_1 into a vertical column of light incident on a mask. The mask consists of a series of transparent and opaque squares, and each vertical column of such squares stores a binary code for the residue product of $h_{pq}f_p$. As different residue inputs f_p occur in time (i.e., as different vectors $\vec{f}$ are input), different vertical columns are

illuminated, and different products are read out in binary form.

Now suppose another mask representing, in binary form, all possible residue products of a different matrix element $h_{pq'}$ with the possible residue values of f_p, is stacked above the previous mask, as shown in Fig. 13a. It is clear that both products $h_{pq}f_p$ and $h_{pq'}f_p$ can be read out on respective detector arrays in parallel.

Progressing one step further, we imagine that an array of $P \times Q$ submasks is arranged in one large mask, as shown in Fig. 13b. Each vertical column of submasks produces binary encoded residue products $h_{pq}f_p$ for a particular value of p. Different columns yield results for different p's. A detector array consisting of

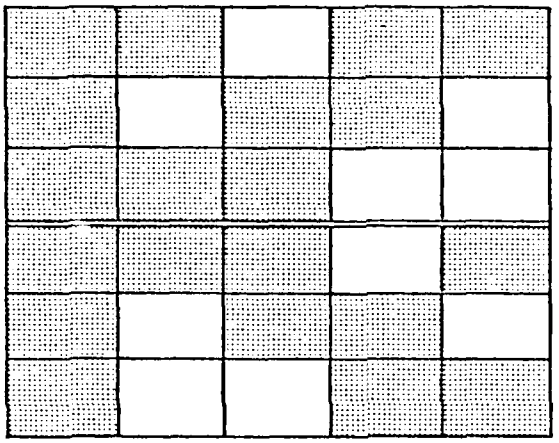
$$N = P \times Q \times m_i \tag{16}$$

elements is required, where m_i is the smallest integer greater than or equal to $\log_2 m_i$.

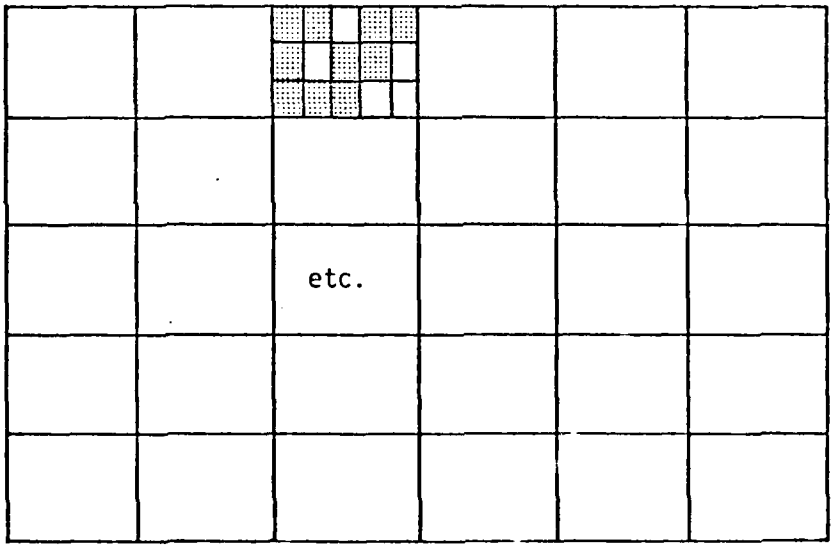
E. Additions

The next operation to be performed is the sum in Eq.(15). Actually, Q such sums must be formed in parallel, one for each element of the output vector $\vec{g}$. We discuss two different methods for performing these sums.

The integrated-optic cyclic permutators provide one means for generating the desired sums. The detected signals measured following the multiplier mask are used to set the switches in the integrated optic adders. Each row of submasks in Fig. 13b is followed by a row of integrated optic adders. All switches in a given row are set in parallel by the detected



(a)



(b)

Fig. 13: Masks for performing residue multiplication:
(a) a pair of products obtained in parallel;
(b) many products obtained in parallel

signals in that row. Light is input at the zero port of the first adder in each row, and the sums cyclically accumulate, the light beams emerging at output ports corresponding to the residues of the desired sums. This overall system is illustrated in Fig. 14.

The total time required to perform the matrix-vector multiply for one modulus consists of the time needed to: (1) set the $\vec{f}$ converters; (2) propagate light signals through the encoders and the multiplier mask; (3) activate the detectors; (4) set the summing encoders; and (5) propagate light through the summing encoders. The detectors are probably the slowest elements in this chain.

An alternative method for performing the additions is by means of a CCD detector and cyclic counters. A single row of a CCD detector array replaces a row of the discrete detectors used in the previous realization. A 2-D CCD detector replaces the entire array of discrete detectors, but it is assumed that all rows of the CCD array can be clocked out in parallel.

If the modulus of concern is 5, then each of the submasks must be followed by $m_5 = 3$ detectors. Hence each full row of submasks must read onto three rows of the CCD detector. Since a given row of detector elements sees only 1's or 0's, we can read out that row into a cyclic counter which tells us the sum across that detector row. To elaborate, let k_{pq0}, k_{pq1} and k_{pq2} be the least significant to most significant bits of the product $h_{pq}f_p$, mod 5. Then we wish to find the sum (for each q)

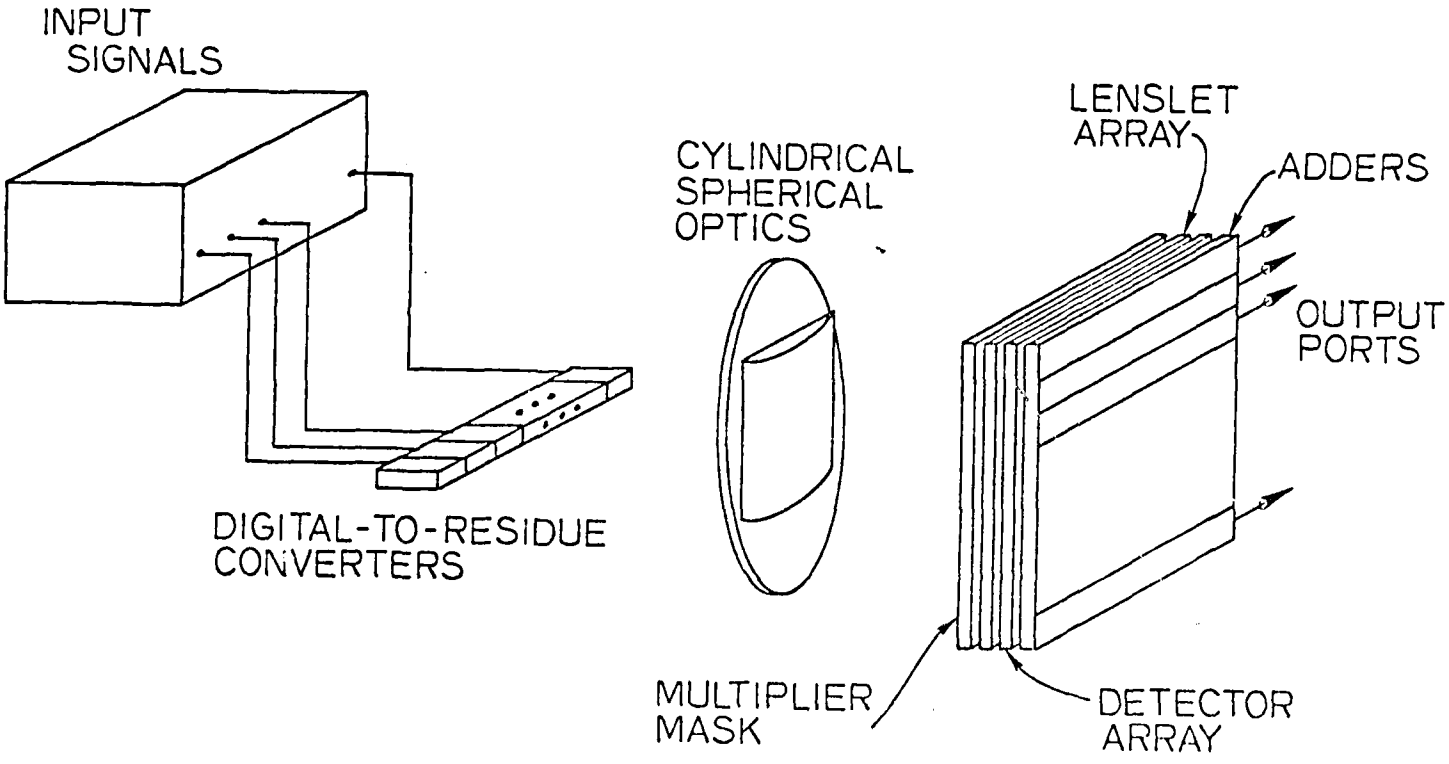


Fig. 14: Overall structure of proposed residue matrix-vector multiplier (for a single modulus)

$$S_q = \sum_{p=0}^{P-1} k_{pq2} \times 2^2 + k_{pq1} \times 2^1 + k_{pq0} \times 2^0$$

$$= 2^2 \times \sum_{p=0}^{P-1} k_{pq2} + 2^1 \times \sum_{p=0}^{P-1} k_{pq1} + 2^0 \times \sum_{p=0}^{P-1} k_{pq0} \qquad (17)$$

But we only want the residue of this sum, so we need only find

$$S_q' = 2^2 \times \text{mod}\left(\sum_{p=0}^{P-1} k_{pq2}, 5\right) + 2^1 \times \text{mod}\left(\sum_{p=0}^{P-1} k_{pq1}, 5\right)$$

$$+ 2^0 \times \text{mod}\left(\sum_{p=0}^{P-1} k_{pq0}, 5\right) \quad . \qquad (18)$$

Each of the mod operations above is accomplished by means of a cyclic counter. Since again we only desire the residue of the product, the multiplications by the weighting factors 2^2, 2^1 and 2^0 can be performed by fixed electronic maps. The addition of the three terms can then be accomplished by cascading integrated optic adders or their electronic equivalents.

F. Decoding to Mixed Radix Form

Additional processing of the modular outputs is required in order to express components of the output vector in mixed radix form. Such processing takes the form described previously in section II-I.

Considering a given modulus m_i, we assume the moduli $m_1, m_2, \ldots, m_{i-1}$ have already been processed to yield residues $\{R_1, R_2, \ldots, R_{i-1}\}$. The output of the residue-m_i processor is R_i. First we must subtract R_1 from R_i, then multiply this difference by $\left|\frac{1}{m_1}\right|_{m_i}$. R_2 would then be subtracted from this number, and the result multiplied by $\left|\frac{1}{m_2}\right|_{m_i}$.

This process must be repeated for all the previously obtained residues, requiring $(i-1)m_i$ encoder subunits for the subtractions, and $i-1$ fixed multiplicative mappings for each element of $\vec{g}$. These mappings can be incorporated into the final residue summations, with the only result being an increase of the propagation time needed for these summations.

G. Computational Complexity

Multiplication of a $P \times Q$ matrix times a length P vector requires $P \times Q$ multiplications and $P \times Q$ additions. Thus if $\underline{H}$ were a 16×16 matrix, $\vec{f}$ a 16-length vector, and all elements of $\underline{H}$ and $\vec{f}$ were a maximum of 12 bits long, the computations would require 256 12-bit multiplications and 256 24-bit additions, resulting in a maximum dynamic range of 32 bits (4,294,967,296) for $\vec{g}$.

This same computation could be performed by a residue matrix-vector multiplier. To achieve a 32 bit dynamic range, we could choose the 8 moduli (31,29,27,23,19,17,13,2) , the largest of which is expressible as a 5 bit number, and giving us a range from 0 to 4,688,427,041. Assuming the throughput rate is determined by the largest modulus, we can express this rate (in vectors/sec.) as

$$= \underbrace{(T_s + 12T_p}_{\text{encoding}} + T_m + \underbrace{T_d + T_s + 5PT_p}_{\text{sum products}} + \underbrace{7(T_p+5T_p) + T_d)^{-1}}_{\text{mixed radix conversion}} \qquad (19)$$

where

T_s = set time of optical encoders;

T_p = propagation delay through an encoder subunit or a single map;

T_m = propagation time through the multiplier mask and associated optics;

T_d = detection time;

P = number of elements in vector.

If $T_s \cong T_d \cong 10$ n sec, $T_p = T_m = 0.1$ n sec, and $P = 16$, then the matrix multiply is performed about 2×10^7 times per second with 32 bits of accuracy. This corresponds to 10.24×10^9 arithmetic operations per second.

A hybrid approach using a CCD array, electronic encoding, and cyclic counters would have a throughput rate

$$= \underbrace{(T_s + 12T_p}_{\text{encoding}} + T_m + \underbrace{P\times T_{CLOCK}}_{\text{CCD scan}} + \underbrace{T_s + 5T_p}_{\substack{\text{sum of binary} \\ \text{sub pro-} \\ \text{ducts}}} + \underbrace{T_s}_{\substack{\text{electronic} \\ \text{mapping}}}$$

$$+ \underbrace{7(T_p+5T_p) + T_d)^{-1}}_{\text{mixed radix conversion}} \qquad (20)$$

where T_{CLOCK} is the time required for one CCD shift (the clock period). For values $T_s = 2$ns, $T_p = 1$ n sec, $T_m = 0.1$ n sec, $T_{CLOCK} = 100$ n sec and $T_d = 10$ nsec, the throughput rate is about 6×10^5 vectors per second, or about 357.2×10^6 arithmetic operations per second. The dominant limitation to speed in this case is the CCD detector.

If such a unit were constructed as a digital electronic system using 12 bit parallel 175 n sec multiplier accumulators (TRW model TDC 1003J, $150 each, 2.5 watts dissipation each) and 32 bit parallel 10 n sec adders, then the fastest version would require 256 multipliers and 256 adders. The throughput rate would be 3×10^6 vectors per second, faster than the CCD implementation, but more than an order of magnitude less than the optical residue system, and achieved only at a very substantial cost in electronic hardware.

V. AN IMPLEMENTATION OF A RESIDUE ARITHMETIC UNIT WITH CONVENTIONAL LOGIC

Attention is now turned to the possibility of constructing a residue arithmetic unit with conventional logic. An obvious approach would be to use AND gates to gate signals off of a bus. The output of the gates would be connected to another bus in a prearranged manner, as shown in Figure 15. Thus, by selecting a particular set of gates the connections between the buses would be mapped in a certain manner. If this bank of maps were to function as a residue adder then m_i different maps would have to interconnect the buses, where m_i is the modulus. Thus m_i^2 gates would be needed. The outputs of each gate would be connected to m_i-1 other outputs and m_i inputs. The fan in and fan out constraints would be pressed by even a modest sized modulus. One solution would be to parallel each gate to increase the drive power.

The number of maps needed for a residue adder can be reduced with a binary decomposition technique. Thus, only 2 maps would be needed per bank. However, $m_i = \lceil LOG_2 m_i \rceil$ banks would be necessary. This would mean the signals would take approximately $LOG_2 m_i$ longer to propagate through such an adder than they would through a single map. This approach eases the fan in and fan out problem at the expense of propagation delay. The number of AND gates needed would be $2m_i\lceil LOG_2 m_i \rceil$.

The problem of implementing a residue multiplier would be similar. Either m_i maps would be needed in each bank or $2\lceil LOG_2(m_i-1) \rceil + 2$ selectable maps in $\lceil LOG_2(m_i-1) \rceil + 1$ banks could be used. Thus the

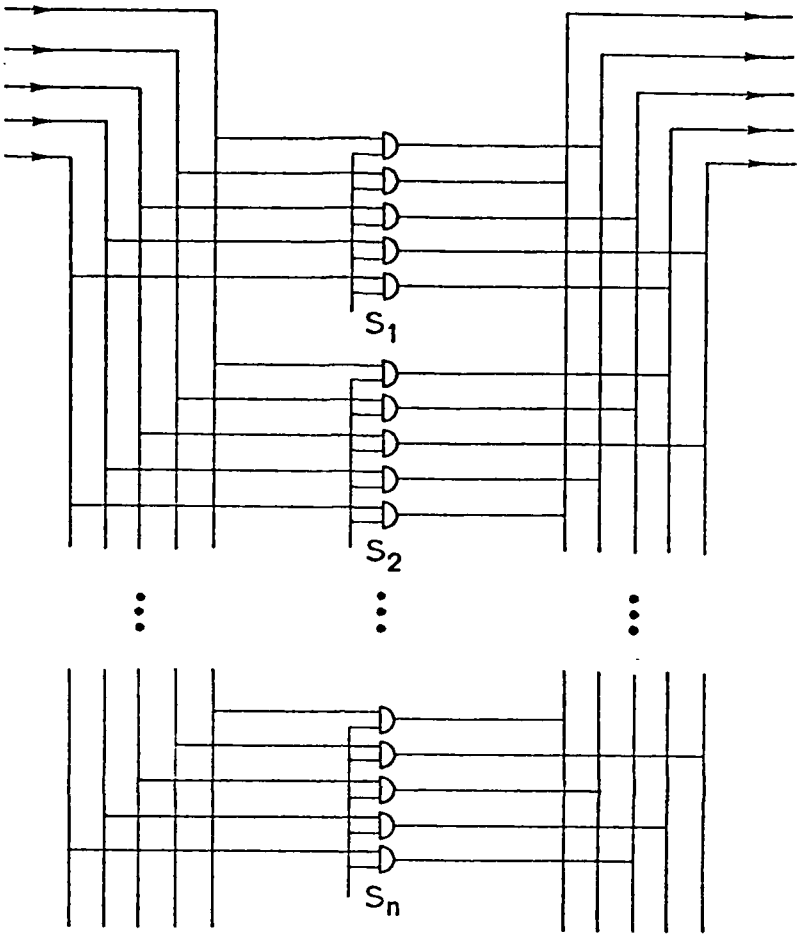


Fig. 15: A bus gating structure which allows selectable interconnections between two busses.

obvious approach would require m_i^2 gates while the decomposition approach would take $m_i(2\lceil LOG_2(m_i-1)\rceil + 2)$ gates. Other types of maps that are needed, such as fixed maps, would just be a rewiring and would not add to the gate count or delay. Polynomial transform maps are equivalent to any other gated maps.

The latency and component count of systems can be analyzed with this approach. The latency of such systems is proportional to the number of banks while the number of maps within a bank is limited by the fan out of the logic. The time to perform a calculation with a cascaded bank approach for any realistic computation becomes impractical quite quickly. This situation can be improved by pipelining such a unit by inserting latches between banks or groups of banks to act as temporary storage platforms. Thus the throughput rate of the system could be improved. However, this is accomplished at the expense of increased latency, which is the time delay associated with a particular calculation.

A circuit was designed which can either be considered as one stage of a pipelined system of the iteration primitive of a complete modular processor. The approach was to design a "super" bank. This is a bank that has the maps necessary to act as an adder, multiplier, and subtractor for a given modulus. The output of the bank was latched with some master-slave flip flops. The outputs of the flip flops serve as the inputs of the bank. (The circuit is playfully referred to as "a snake eating its tail". See Figure 16.) A version of the circuit was designed for modulus 5. It had 3 addition maps (reduced via binary decomposition), 5 multiplier maps, and a polynomial transform map $P(X) = 5-X$ to implement subtraction. Circuitry was also included for loading and resetting the

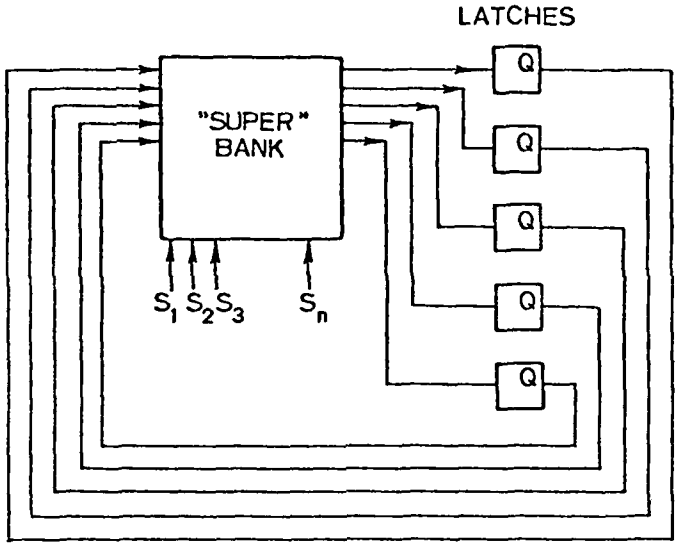


Fig. 16: One stage of a pipelined system or an iteration primitive of a complete modular processor.

flip flops. The circuit was designed on an automated digital design facility called SPRINT and located at the Stanford Linear Accelerator Center. The circuit required 11 integrated circuit chips. The board layout is shown in Figure 17. The horizontal connections are on one side of the board while the vertical ones are on the other side. 370 connections are needed with 147 vias (connections between the two sides of the board). About 150 inches of connections are needed. The design illustrates several problems with a digital approach. For high speed operation the path lengths, and thus the travel times through all the inputs of all the maps, must be about the same since the cycle time would be constrained by the worst case. This proved to be very difficult. The inputs and outputs of gates always seem to be on the wrong side of a chip.

At this level of circuit complexity the designer still has some degree of control. Larger moduli imply a greater complexity and would probably require design in a completely automated manner. The designer would thus lose control of the various path lengths. The path length problem is further complicated by the connection routing problem associated with printed circuit boards. The only way a connection can bridge another is by jumping to the other side of the board. As the circuit complexity increases the routing problem increases drastically. The problem of large differences in signal paths could probably be solved by integrating the whole circuit on a chip. This custom LSI approach would probably be quite costly since a different chip would be needed for each modulus. Another problem with such a chip would be the pin count needed. At least m_i pins for input, m_i pins for output,

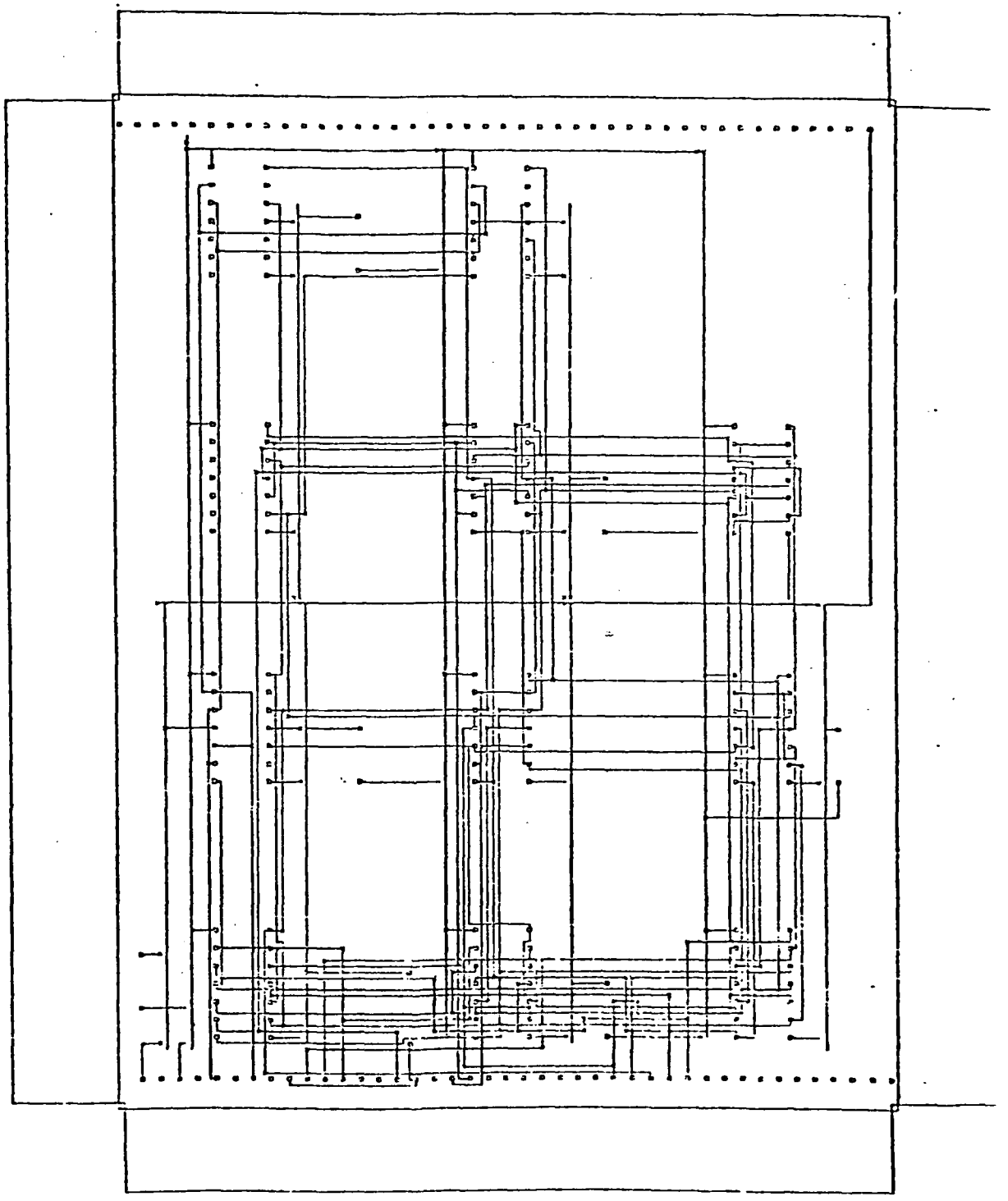


Fig. 17: Printed circuit board layout for a modulus 5 "super bank".

$\lceil LOG_2 m_i \rceil$ pins for addition map selection, $\lceil LOG_2(m_i-1) \rceil + 1$ pins for multiplicative map selection, a pin for reset, a pin for set, a pin for power, and a pin for ground. The number of pins for input or output could be reduced to $\lceil LOG_2 m_i \rceil$ each. However, binary to m_i decoders and the m_i to binary encoders would have to be included on the chip. This would further contribute to the delay.

The basic circuit was intended as more than just a testbed for implementation problems. It was designed as the iteration primitive of a modular processor of a residue processing system. It was designed to demonstrate the least number of components needed to implement a complete system. The circuit can mimic any number of cascaded banks by sequentially duplicating the effect of each bank. Thus a system of 100 banks could be simulated with 100 cycles of this circuit. An interesting aspect of this approach is that n banks can be implemented in less than n cycles depending on the calculation desired. Any cycles needed for addition of 0 or multiplication by 1 can be skipped. This savings is demonstrated in the process of encoding from binary to residue. If the binary digit is 0 nothing needs to be done. Thus on the average a D digit binary number can be encoded in $D/2$ cycles. Now suppose that there were n_A cascaded banks of addition and that the requests for each type of addition map were tallied in separate counters. Each additive map can be used repeatedly until it depleats its tally, since addition is commutative and the cascading of additive maps in any order is equivalent. Furthermore, the tally counters need only be modular m_i counters since the effect of cascading m_i+x of the same additive maps is equivalent to cascading x such maps. Thus if n_A additive requests were randomly

distributed to the $\lceil LOG_2 m_i \rceil$ types of additive maps each modular tally would, on the average, have $m_i/2$ requests. Thus the number of cycles needed for n_A requests could be reduced to $\lceil LOG_2 m_i \rceil (m_i/2)$ cycles. What is so unusual is that this is independent of the number of addititive requests n_A.

It must be noted that all requested additions or subtractions (addition of an additive inverse) must be honored before any multiplication since multiplication and addition are not commutative. If a multiplication by 0 is desired, all additive tallies can be immediately reset. Thus this method of tallies works best with large cascades of additions. While this scheme was originally developed to reduce the latency of an iterative electronic modular processor, it can be applied to any type of residue processor. It is a type of computational compression only available in residues. The technique might also be of interest in some special cascaded bank versions.

The electronic implementations of banks and their maps become quite complex very quickly. The problems associated with packaging could probably be minimized with an LSI approach; however, any approach using conventional logic will still face a fundamental difficulty. The entire residue arithmetic unit was designed to avoid detections and restimulations of signals because this would inhibit optical approaches. Conventional logic relies on detection and restimulation of a signal at each gate. Though this delay is very small, it is not required by the underlying architecture, and thus it becomes a burden. Even though the gates of n cascaded banks could be set in parallel, a signal would still take n set times to propagate through such a system. To try to avoid

such a problem, pipelined and iterative versions of the system, using conventional logic, were explored. The pipelined version contributed to a more efficient utilization of the banks, while the iterative version allowed computation compression via various shortcuts, but neither version could hide the implied set time of each level of logic. The fundamental fact is that this residue arithmetic unit was specifically designed to avoid the need of detections and restimulations at each level of logic. Thus, any such delay, no matter how small, would be a hinderance and would slow the system.

VI. POTENTIAL ADVANTAGES OF OPTICS

In the previous sections, both optical and electronic implementations of basic residue operations have been considered. A constantly reoccurring question throughout the past year has been: exactly what potential advantages does optics offer over an all-electronic implementation? Our perception of the answer or answers to this question has gone through many evolutions, generally progressing from the philosophical to the more concrete. The evolution is still in progress, but we offer the following thoughts on this matter.

A. Parallelism

The inherent parallelism of optical systems, and the parallel nature of the residue number system, have been constant driving forces behind our attempts to match the two. An extreme example of this parallelism is offered by a simple lens. By imaging a set of input ports onto a set of output ports, a reversal of the order of those ports can be accomplished. This reversal is achieved without the necessity of providing separate electrical connections between inputs and outputs, and without the

associated problems of connections crossing over connections.

The integrated optic cyclic permutation device of Fig. 7 provides a second good example. With only three separate switching channels per subcell, and only two electrical inputs to each subcell, a complicated cyclic permutation of many ports is accomplished. (Note that regardless of how large the modulus may be, only three switching channels are required per subcell.) The cyclic permutator achieved with waveguide couplers does not share this parallelism -- separate paths must be supplied for each possible connection.

B. Bandwidth

Unfortunately, optical devices (including detectors, switches, etc.) do not offer any significant speed advantages over their all-electronic counterparts, at least with respect to the time it takes to activate the device. Nonetheless, the possibility of accomplishing some form of wavelength multiplexing remains tantalizing. For the first time a concrete path towards this goal can be identified. The waveguide couplers composed of electrically activated Bragg gratings can be very wavelength sensitive. It is possible to envision one set of Bragg couplers which are effective only for a narrow band of optical frequencies, and are completely without effects on other frequency bands. It follows that multiple sets of couplers could be deposited, each set operating independently on its own optical frequency band. The throughput of the optical residue unit could thus potentially be increased substantially. One can even consider the possibility of realizing all of the different modular processors on one set of waveguides. While this possibility may today seem remote, due to coupler inefficiencies, low yield,and other problems,

nonetheless it provides a strong motivation for further work on perfecting the technology.

C. Set time vs. propagation time

While optical and electronic switching devices require comparable times for activation, the time required for propagation of a signal through the device may be much shorter in the optical case. Hence, if a long sequence of additions are required, as in the case of matrix-vector multiplication, for example, the speed with which the additions are performed may be much faster in the optical case, in spite of comparable set times for the optical and electronic devices. Effort is now being expended to quantify this idea further.

D. Light through light

In studying the electronic methods of implementing residue operations (section V), it became apparent that major problems in propagation delay arise due to the difficulties associated with crossing electrical interconnections. Such difficulties can be avoided in some optical realizations, due to the ability of light beams to pass through other light beams without interaction. Such an advantage is evident in the cyclic permutator of Fig. 7, in which many different optical paths cross one another in a purely planar geometry.

VII. CONCLUDING REMARKS

We conclude this report with miscellaneous information concerning the administration of this grant.

A. Publications. The following written publications were produced during the first year of this grant:

(1) Y. Tsunoda and J.W. Goodman, "Combined optical AD conversion

and page composition for holographic memory applications", Appl.Opt. 16, pp.2607-2609, October 1977.

(2) A. Huang, "An Optical Arithmetic Unit", Digest of Papers, International Optical Computing Conference, Budapest, Hungary, October 1977.

(3) A. Huang, "An Optical Arithmetic Unit", Proceedings of 1977 Electro-Optical Systems Design Conference, Anaheim, 1977.

B. Professional personnel

The following individuals contributed to the research effort supported by the grant:

(1) J.W. Goodman, Principal Investigator

(2) Alan Huang, Research Assistant

(3) Yoshito Tsunoda, Visiting Scholar

(4) Satoshi Ishihara, Visiting Scholar

(5) Amnon Aliphas, Research Assistant

C. Spoken Papers

(1) Alan Huang, "An Optical Arithmetic Unit", International Optical Computing Conference, Budapest, Hungary, October 1977.

(2) Alan Huang, "An Optical Arithmetic Unit", Annual Meeting of the Optical Society of America, Toronto, Canada, October 1977.

(3) Alan Huang, "An Optical Arithmetic Unit", Electro-Optical Systems Design Conference, Anaheim, Ca. November 1977.

D. Inventions

A patent disclosure on the cyclic permutator of Fig. 7 has been completed with Yoshito Tsunoda, Alan Huang and Joseph Goodman as coinventors.

A patent disclosure on an optical A/D converter, as described in publication (1) above, has been completed with Yoshito Tsunoda and Joseph Goodman as coinventors.

REFERENCES

1. M.A. Monahan, K. Bromley and R.P. Booker, Proc. IEEE, 65, p.121 (1977).
2. A. Vander Lugt, Proc. IEEE, 62, p. 1300 (1974).
3. J.W.Goodman, Proc. IEEE, 65, p.29 (1977).
4. V.N. Morozov, Digest of Technical Papers, CLEA '77, 16.10, p.81 (1977).
5. K. Preston, Jr., Coherent Optical Computers, p.232, McGraw-Hill, New York (1972).
6. D.H. Schaefer and J.P. Strong, III, Proc. IEEE, 65, p.129, (1977).
7. N.G. Basov, W.H. Culver and B. Shah, Laser Handbook (edited by F.T. Arecchi and E.O. Schulz-DuBois), p.1651, North Holland Publ. Co., (1972).
8. P.W. Cheney, "An Investigation of Residue Number Theory for Digital Systems", Ph.D. dissertation, Stanford University, September 1961.
9. J.W. Bond, Naval Undersea Center Report, AD-780-805-D1 (1974).
10. Alan Huang, "The Implementation of a Residue Arithmetic Unit via Optical and other Physical Phenomena", Proceedings of the International Optical Computing Conference, Washington, D.C. April 1975.
11. H.L. Garner, "The Residue Number System", IRE Trans.Electron. Comput., Vol. EC-8, pp.140-147, June 1959.
12. N.S. Szabó and R.I. Tanaka, Residue Arithmetic and Its Applications to Computer Technology, New York, McGraw-Hill, 1967.

13. A. Svoboda and M. Valach, Rational System of Residue Classes. In Stroje na Zpraccováni Informaci, Sbornik V., pp.9-37, Nakl. CSZV. Prha, 1957 (in English).

14. A. Svoboda and M. Valach, "Computer Progress in Czechoslovakia II, The Numerical Systems of Residue Classes", in Digital Information Processor, Walter Hoffman, ed. John Wiley & Sons, Inc., N.Y. 1962.

15. D.E. Knuth, Seminumerical Algorithms, vol. 2, pp.248-256, Addison-Wesley, Reading, Mass. 1969.

16. N.S. Szabó and R.I. Tanaka, (ibid.) pp.160-162.

17. S.A. Collins, Jr., "Numerical Optical Data Processor", Proc. of SPIE, Effective Utilization of Optics in Radar Systems, Vol. 128, September 1977, pp.313-319.

18. Private communication from William Stoner, Systems Applications, Inc., Bedford, Mass.

19. S.K. Sheem and C.S. Tsai, Digest of Technical Papers, CLEA meeting, 2.7, p. 7 (1977).

20. B. Chen, M.K. Barnoski, C.M. Beijer, "Thin Film Bragg Switch", Topical Meeting on Integrated and Fiber Optics, Salt Lake City, Utah, January 1978.