Optical residue arithmetic computer with programmable computation modules

A. Tai, I. Cindrich, J. R. Fienup, and C. C. Aleksoff

The concept of a programmable multipurpose computation module for residue arithmetic is introduced. A possible design for the module is also presented. The application of the computation module in mathematical computation, coding, decoding, and scaling is demonstrated. Very high throughput rate is achieved by the use of parallel computation structures and pipelining of sequential operations.

Introduction

For some time, we have witnessed the competition between electronic digital computers and optical analog computers in several areas. The possibility of bringing the virtues of both types of computer together in the form of an optical numerical computer is intriguing. The potential advantages of an optical numerical computer have been discussed by Huang *et al.*[1] These include the inherent parallelism of optical systems, the possibility of wavelength multiplexing, and the short propagation time of optical signals. Many of these advantages cannot be fully realized at the present stage of hardware technology. Nevertheless, the potential of an optical numerical computer is significant, and it warrants a continuing design concept development that is not entirely constrained by the presently available hardware technology.

Residue arithmetic has been considered by computer engineers for many years as a means of achieving high-speed parallel processing.[2–5] It was not until recently, however, that the idea of implementing residue arithmetic optically was brought into vogue.[1,6–8] The basic concept of residue arithmetic and some of the possible implementation techniques have been discussed by Huang *et al.*[1] and others.[6–8] In this paper, we shall present the concept of a programmable multipurpose computation module and its possible design. The motivation is that generally it is more economical to mass produce a single type of device that can be programmed to perform various required functions than to produce different types of devices for different functions. The computation module will be used as the basic building block to implement various functions in an optical numerical computing system such as encoding, decoding, and scaling, along with all the basic arithmetic operations.

A residue number can be represented by spatial position, phase or polarization angle. Being cyclic phenomena, phase and polarization are natural choices for residue representations. However, using phase or polarization would entail the use of analog devices for the representation of discrete values. In order to avoid the high probability of error such representation can cause, a cyclic device with multistable states would be necessary. Collins *et al.*[7] have successfully demonstrated a feedback technique using liquid crystal modulators that produces two stable states. The concept can be extended to more than two states, but the problem of extending to many stable states for large moduli is severe. Another feedback technique was demonstrated recently by Okada and Takizawa[9] using $LiNbO_3$ as the nonlinear modulator. The technique can be utilized to produce several stable states, but the strong hysteresis characteristic may be a drawback in some applications. Furthermore, while cyclic devices can perform the addition operation very conveniently, implementing fixed transformations and multiplication is more complicated than with the use of spatial maps. On the other hand, the use of phase or polarization representation may allow greater packing density than the spatial representation. At least for the near future, the spatial representation using programmable spatial maps for computation will likely be the more readily achievable implementation approach.

The concept of residue arithmetic is described in various literature.[1–4] Here we will only review some basic concepts and the notations employed in this paper. A residue number system is based on N relatively prime integers $m_1, m_2, \ldots, m_N$ called moduli. A

The authors are with the Environmental Research Institute of Michigan, P.O. Box 8618, Ann Arbor, Michigan 48107.

Received 23 March 1979.
0003-6935/79/162812-12$00.50/0.

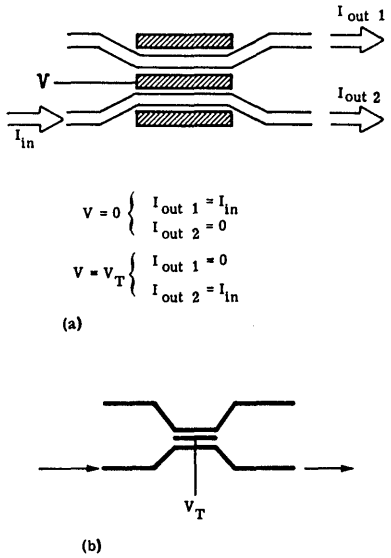


Fig. 1. (a) Directional coupler waveguide switch (not to scale); (b) schematic representation.

number X can be represented in this residue number system as $r_{m_1}, r_{m_2}, \ldots, r_{m_N}$. The residue r_{m_i} can also be written as $|X|_{m_i}$, and it is the remainder of the X/m_i division operation. For example, with $X = 24$ and $m_i = 5$, the residue for modulus 5 would be $r_5 = |24|_5 = 4$. The number of integers that can be represented uniquely by the residue number system is

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

The residue representation is cyclic over the range M. That is, the representations are the same for integers K, $K + M$, $K + 2M$, etc. Ordinarily, only positive numbers are represented by the residue number system. However, negative numbers can also be represented by letting numbers 0 to $M/2 - 1$ represent positive integers and $M/2$ to $M - 1$ represent negative integers such that $M/2 \equiv -M/2$ and $M - 1 \equiv -1$. We may add that a similar method of implicitly representing the sign of a number is used in the conventional two's complement binary system.

Implementation with Waveguide Switches

To demonstrate the design concept, we have chosen the use of directional coupler waveguide switches for the implementation of the computation module. The directional coupler is one of the better developed integrated optical devices, and it allows flexibility in optical circuit design.[10–12] We shall briefly describe the optical coupler waveguide switch and then proceed to formulate its use in a basic addition operator, which will be extended into a programmable multipurpose computation module. We should emphasize that there are other components that would be good candidates for the implementation of a numerical optical computer, and the waveguide coupler is but one of the more interesting possibilities at present.

A direction coupler is schematically shown in Fig. 1. Two waveguides are placed physically close to each other such that in the absence of an applied electric field, the waveguides are synchronous. That is, a light wave propagating in one waveguide will be coupled to the adjacent one producing a switch in light path.[10–12] When an appropriate voltage V_T is applied to the electrode, the synchronism between the waveguides is broken and the light propagation will remain in the waveguide originally excited as illustrated in Fig. 1(a). For simplicity, the coupler waveguide switch from here on will be represented as shown in Fig. 1(b).

Module Subunits

The heart of the programmable computation module is an adder. Addition in residue arithmetic is essentially a shifting operation. The input light beam is shifted by K positions for the operation $+K$ as illustrated in Fig. 2 for modulus 5. One possible implementation of a modulo 5 adder is shown in Fig. 3. With this design, the electrode voltages of all the coupler waveguide switches are initially set at V_T. The light wave injected into the input of the adder will therefore propagate inside the same waveguide through the adder. To program the device for the +2 operation, for example, the electrode voltage of the corresponding row of couplers is changed to 0. Thus, when the light propagation reaches that particular set of coupler waveguide switches, the light wave will be coupled into the adjacent waveguide, changing the optical path. The electrode

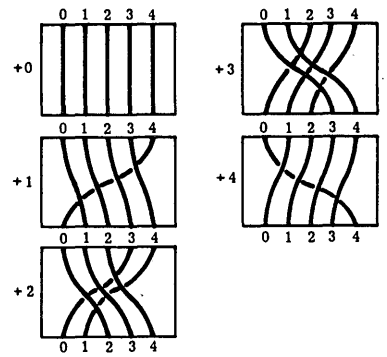


Fig. 2. Modulo 5 addition.

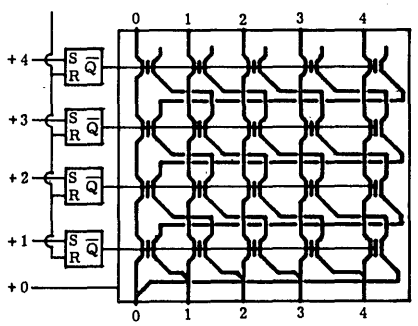


Fig. 3. Implementation of Modulo 5 adder (not to scale).

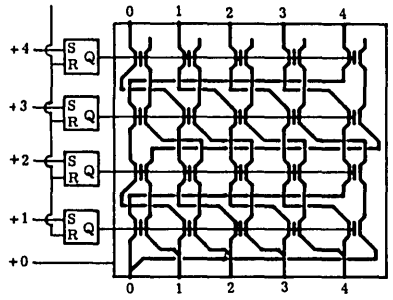


Fig. 4. Alternate design for Modulo 5 adder.

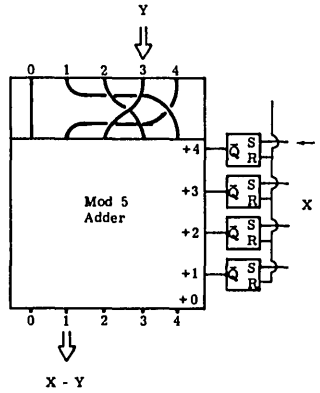


Fig. 5. Converting an adder for subtraction operation.

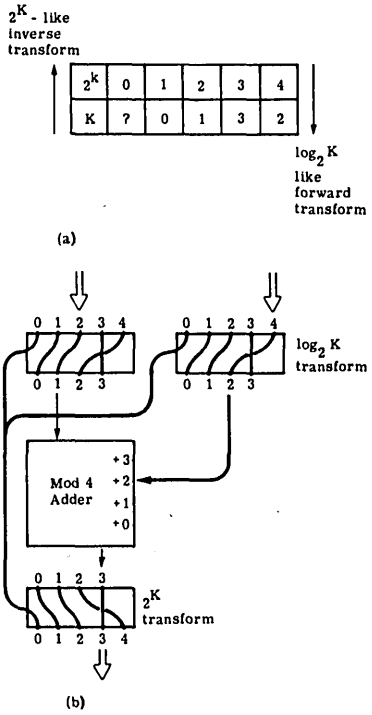


Fig. 6. (a) Transform table for Modulus 5 (b = 2); (b) Modulo 5 multiplication using the homomorphic approach.

voltages are maintained at constant levels of V_T or 0 by connecting the electrodes to a set of S–R flip-flops. The adder can be programmed by sending an electric pulse to the S input of the appropriate flip-flop, triggering it to change state. Alternatively, we could let the initial electrode voltage of all the couplers be 0 and program the adder by changing the electrode voltage of a particular row of coupler switches to V_T. The alternate design of a modulo 5 adder is shown in Fig. 4. However, we generally find that it is easier to trace the light path with the former design, and to make the devices easier to study, we shall make use of the former design in this paper. We shall also use the term on to describe the state where coupling occurs at the coupler switch and term off for the state where the light propagation will remain in the same waveguide.

Subtraction can be performed with the use of the additive inverse. The additive inverse $|-K|_{m_i}$ of a residue number K is defined such that

$$|K + |-K|_{m_i}|_{m_i} = 0.$$

There is a fixed one-to-one correspondence between a residue number and its additive inverse. The additive inverse transformation can therefore be implemented by a fixed map. And by adding this transformation map to an adder, one can convert it into a subtractor as shown in Fig. 5 for modulus 5.

Multiplication can be implemented directly by using m_i maps for the operations of $\times 0, \times 1, \times 2, \ldots, \times(m_i - 1)$. Alternatively, one can make use of a homomorphic approach where a modulo m_i multiplication is converted into a modulo $m_i - 1$ additive operation. A $\log_b K$-like forward transform is first performed on the operands. A modulo $m_i - 1$ addition is then performed, and the sum is inverse transformed by a b^K-like transform to obtain the product of the two original numbers. The transform table for modulus 5 is shown in Fig. 6(a), and the process is illustrated schematically in Fig. 6(b). Although the $\log_b K$-like transformation for the value 0 is not defined, it is known that if either the multiplier or the multiplicand is 0, the product is 0. A modulo 5 multiplier is shown in Fig. 7 using this homomorphic approach. We note that for a modulo 5 multiplication, a modulo 4 addition is performed. Thus, in order to convert a modulo 5 adder into a modulo 5 multiplier, the modulo 5 adder should be designed in such a way that it can be easily converted into a modulo 4 adder. This

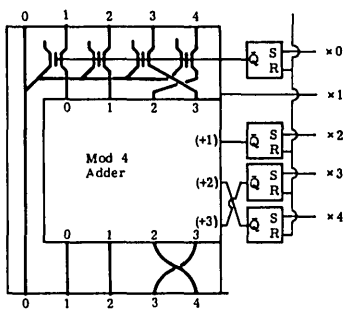


Fig. 7. Implementation of a Modulo 5 multiplier.

can be achieved with the design shown in Fig. 8. While the concept can be applied to an adder of any modulus, we should note that this homomorphic approach can be used only if the modulus is prime.

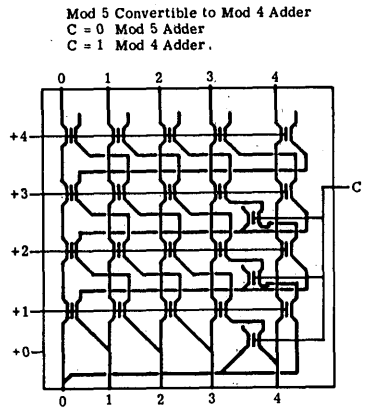


Fig. 8. Modulo 5 adder convertible to Modulo 4 adder.

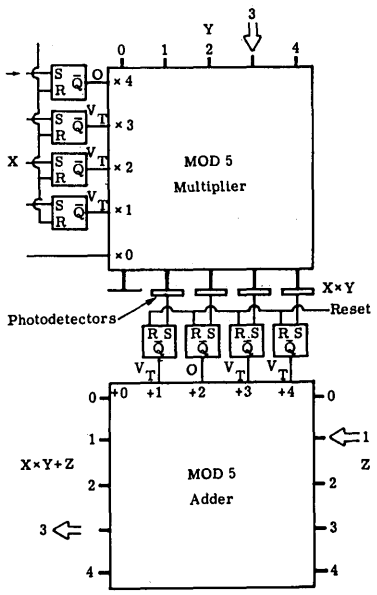


Fig. 9. Interconnection of Modulo 5 computation modules.

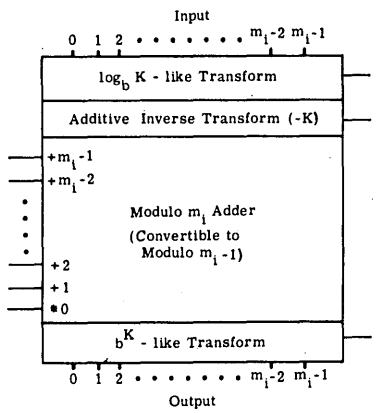


Fig. 10. Conceptual design of programmable multipurpose computation module.

One feature of this design is that the input, output, and programming controls are all represented spatially in the same way. This allows the interconnection of these devices for sequential operations. The outputs of one module can be connected directly to the inputs of the next module or it can be used to program the map of the next adder as illustrated in Fig. 9. An electrical pulse is sent to the first multiplier to program it to perform $\times|X|_{m_i}$, where m_i is the modulus. A light pulse is then injected into the adder at the spatial position corresponding to $|Y|_{m_i}$. The exit position of the light beam would correspond to $|X \times Y|_{m_i}$. A fast avalanche photodiode is connected to each output waveguide. The exiting light pulse will be detected by the photodiode, generating an electric pulse. The electric pulse in turn triggers the corresponding flip-flop of the next adder, setting it for the $+|X \times Y|_{m_i}$ operation. Another light pulse is then injected into the input of the second adder at the position corresponding to $|Z|_{m_i}$. The position where the light pulse exits will represent the sum of $|X \times Y + Z|_{m_i}$.

Programmable Multipurpose Computation Module

With the subunits described above, we can proceed to describe the multipurpose programmable computation module. The module will contain four distinct parts as shown in Fig. 10. Each of these subunits can be turned on and off individually, allowing the different combinations of the subunits to perform various computation operations. However, it is more complicated than simply stacking all the subunits together. Special attention must be paid to the case of $+0$ and $\times 0$ by noting that $X + 0 = X$, $X \cdot 0 = 0$, $0 \cdot Y = 0$, and $X \cdot 1 = X$. Furthermore, the modulus m_i adder must be modified to perform modulo $m_i - 1$ addition, and the $|-K|_{m_i}$ additive inverse transform must be converted into a $|-K|_{m_i-1}$ transform when the module is programmed to perform multiplication and division. A possible design of the programmable multipurpose computation module is shown in Fig. 11.

The multipurpose computation module can be programmed to perform $+$, $-$, $\times$, and $\div$ arithmetic opera-

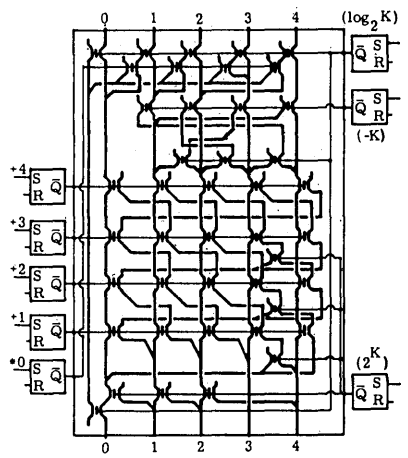


Fig. 11. Implementation of programmable multipurpose computation module.

tions with simple binary controls. For example, to perform modulo 5 addition, the subunits for $\log_2 K$-like transform, additive inverse transform, and 2^K-like transform are all turned off. That is, a light pulse injected into any of the m_i input ports will propagate undeviated along the same waveguide through these subunits. With these units off, the module would be essentially the simple adder shown earlier in Fig. 3. To perform subtraction, the additive inverse transform $|-K|_{m_i}$ unit is turned on, changing the light path according to the transform map shown in Fig. 6. We note that while operating in the addition and subtraction

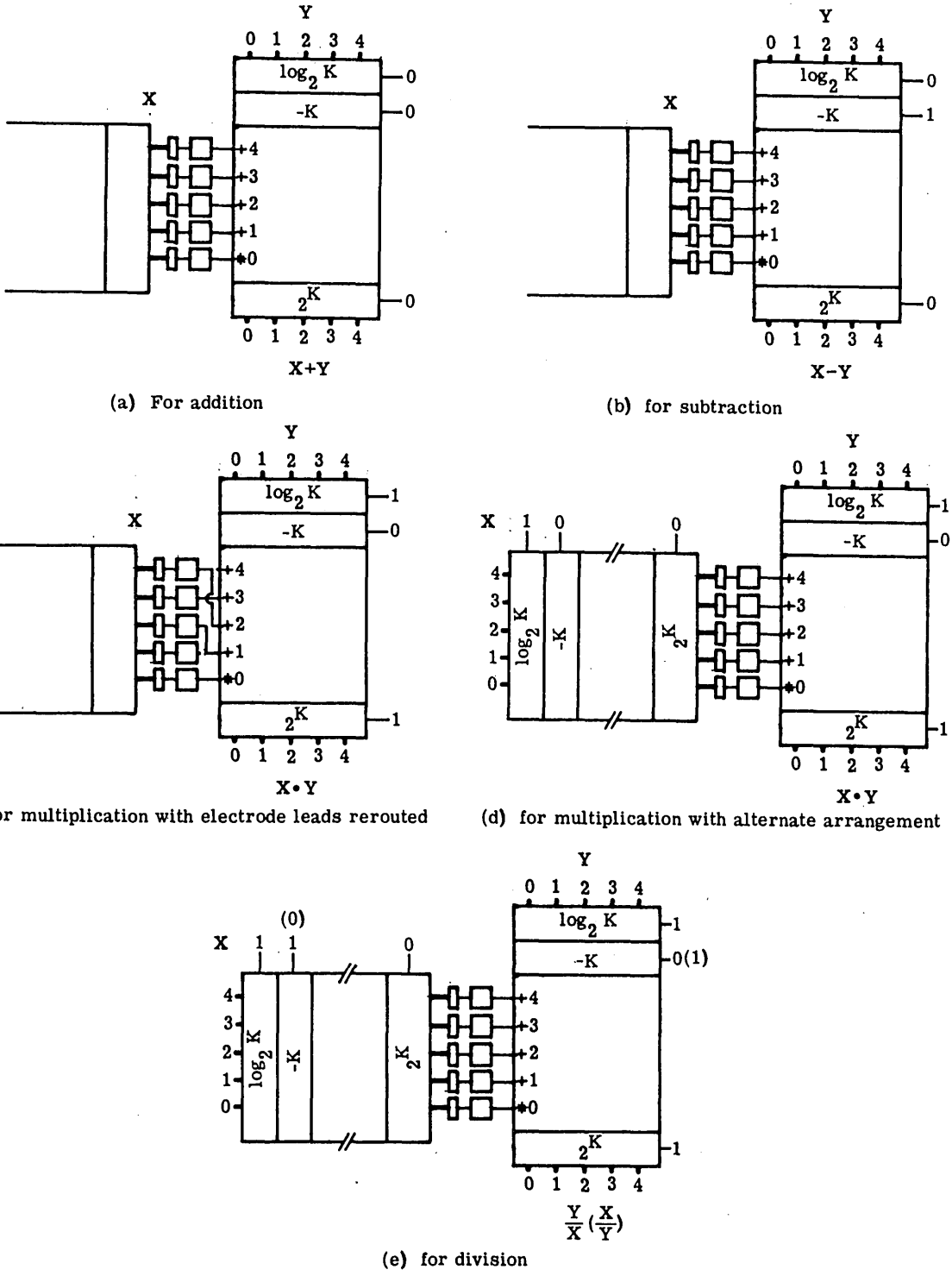


Fig. 12. Programming of computation module for: (a) addition; (b) subtraction; (c) multiplication with electrode leads rerouted; (d) multiplication with alternate arrangement; and (e) division.

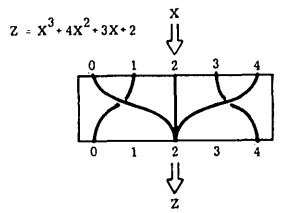


Fig. 13. Evaluation of polynomial with a single map.

modes with the $\log_2 K$-like transform unit off, an input to the *0 control has no effect on the light path. The position of the exit beam would therefore be the same as that of the input beam, performing in effect the +0 operation. The programming of the computation module for addition and subtraction operations is illustrated in Figs. 12(a) and 12(b).

In programming the computation module for multiplication, there are two possible approaches. The module can be connected as a multiplier by rerouting the electrode leads to perform the $\log_2 K$-like transform on the multiplier value (X). The 2^K-like transform unit is turned on to inverse transform the sum as illustrated in Fig. 12(c). With the second approach, both the multiplier (X) and the multiplicand (Y) values are transformed by computation modules, as illustrated in Fig. 12(d). This approach has two advantages. First, the connection of the electrode leads do not have to be changed, allowing the module to be switched back to addition mode when desired. Second, it provides more flexibility in performing division. Observe that the extra coupler switch at the left lower corner in Fig. 11 is necessary for the module to be programmed in this mode. The coupler is turned on together with the $\log_2 K$-like transform unit at the top. It keeps the transformed 0 output of the multiplier from setting the *0 control of the second module. If the value of the multiplier X is 0, the *0 control of the second module is turned on, and the ×0 operation is performed. If the multiplier is 1, its $\log_2 K$-like transform is 0. The extra coupler terminates the signal, and the second module performs a +0 operation by letting the light to exit at the same position as it entered. This bypass would be equivalent to performing an ×1 operation.

Before we illustrate how the module can be programmed for division, some comments on the division operation in residue arithmetic are in order. Division can be performed using the same homomorphic approach, converting a modulo m_i division into a modulo $m_i - 1$ subtraction. However, this can be done only if the quotient is an integer (i.e., no remainder). There are methods that can be employed for general division,[2,13] but they all require cumbersome sequential procedures. Furthermore, they provide only a round-off result since fractions cannot be represented in the residue number system. For this reason, residue arithmetic is generally applied to problems that do not require division operation such as those often encountered in signal processing. Nevertheless, it would be useful to be able to perform division even if it is limited to the remainder zero case. One operation that requires such a division operation is scaling. In order to keep the values within the range of the residue number system, it may be necessary to periodically scale the values down by a factor of K. We shall show later that scaling can be achieved by division if K is a value of one of the moduli or the product of two or more moduli. The programming of the computation module for the division operation is illustrated in Fig. 12(e). An

$$|-K|_{m_i-1}$$

additive inverse transform is required for the divisor after the $\log_2 K$-like transform. A

$$|-K|_{m_i}$$

transform can be changed into a

$$|-K|_{m_i-1}$$

transform by shifting down the values of the $|-K|_{m_i}$ transform by 1. Referring back to the module design shown in Fig. 11, the down shifting is performed by the set of three switches at the fourth row. They are turned on together with the $\log_2 K$-transform unit.

Mathematic Computation

To demonstrate how the computation modules can be interconnected to perform various mathematical calculations, we first apply them to the evaluation of polynomials. As discussed by Huang *et al.*,[1] a polynomial may be evaluated using a single map. For example, the modulo 5 map for the computation of $X^3 + 4X^2 + 3X + 2$ is shown in Fig. 13. However, to generate that map, one would require the help of some external intelligence. The routings of the m_i possible inputs have to be computed beforehand. This implementation is therefore not easily programmable. An alternative is to utilize a set of fixed maps for X^n, X^{n-1}, . . ., X^2 functions in conjunction with the computation modules as shown in Fig. 14. To program the modules for the computation of $X^3 + 4X^2 + 3X + 2$, for example, the coefficients 1, 4, and 3 are entered into the multipliers. Light pulses are injected into the inputs of the multipliers at the ports corresponding to the value of input X. The adders would be set by the output of the multipliers for $+(X^3)$, $+(4X^2)$, and $+(3X)$ operations. Another light pulse is then entered into the first adder at input port 2, and the position where the light pulse exits would correspond to the value of $X^3 + 4X^2 + 3X + 2$.

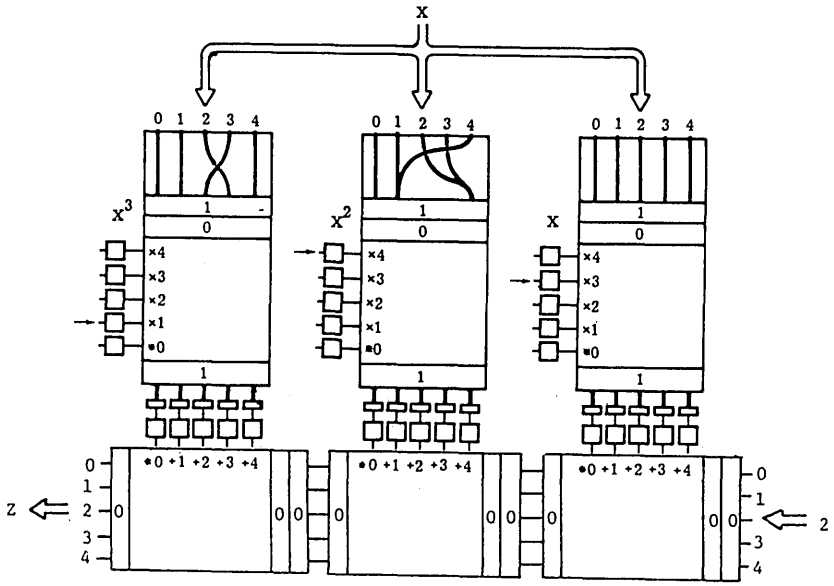


Fig. 14. Programmable arrangement for the evaluation of polynomials.

The computation time would be equal to the time needed to set the adder module plus the propagation time through four modules. The propagation time through a single module of 0.5-in. (1.27-cm) size would be about 40 psec. The set time of the module is the sum of the detection delay of the photodiode, the switching delay of the flip-flop, and the switching time of the waveguide coupler. It is possible to achieve a set time under 1.5 nsec for the computation module.[5–8,10] We may further assume that an additional 1.5 nsec is required for the light pulse to pass through the module and to reset the flip-flops. If the coefficients remain unchanged, the throughput rate would be about 1/(3.12 nsec) = 320 MHz. Due to parallelism of the arrangement, the computation time is approximately the same for polynomials of any order.

One important application of the numerical optical computer is the multiplication of matrices. It can be extended to a number of transform operations such as DFT, Hadamard transform, etc. We shall examine the general case of matrix multiplication, $[A]_{M\times N}[B]_{N\times P} = [C]_{M\times P}$. The coefficients of the fixed master matrix $[B]_{N\times P}$ are stored in the modules as multipliers as shown in Fig. 15. The values of the matrix $[A]_{M\times N}$ pass through the multipliers row by row setting the corresponding row of adders. Light pulses are entered into the first adder of each row, providing in parallel the values of the first row of $[C_{1j}]$ at the output. The flip-flops are then reset, ready for the entries of the next row of $[A]_{M\times N}$. The total computation time is equal to M set-reset times of the module, and the number of computation modules required is $2NP$. For example, to multiply two 10×10 matrices, the computation time would be about $3(10 + 0.04N)$ nsec if we assume a module set-reset time of 3 nsec and the use of 200 computation modules for each modulus. When N is small, the propagation delay will be negligible as compared with the set-reset time of the module. If N is large, however, the throughput rate would be significantly reduced by the propagation delay. Moreover, the optical loss through so many modules would be too high for the output to be detected unambiguously. Both of these problems can be alleviated with the use of pipelining. Instead of propagating a light pulse directly through all N modules, the summation can be partitioned into 2^k parts where k is an integer. The concept of pipelining will be discussed later in better details when the implementation of residue to mixed radix conversion is presented. We may also note that the number of modules can be reduced to $2N$ at the expense of computation speed by sequentially computing each value of $[C_{ij}]$.

Encoding

Before computation can be performed with the modules, the input must be encoded into its equivalent residue number in the appropriate spatial representation. The simplest approach may be to convert the analog input into an intermediate binary form with the use of a conventional A to D converter or the integrated optics implementation scheme introduced by Taylor.[14] The binary input can then be converted into residue numbers in the spatial form with the arrangement shown in Fig. 16. We note that for modulus 5,

$$2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 3,$$
$$2^4 = 1, 2^5 = 2, 2^6 = 4, 2^7 = 3.$$

For example,

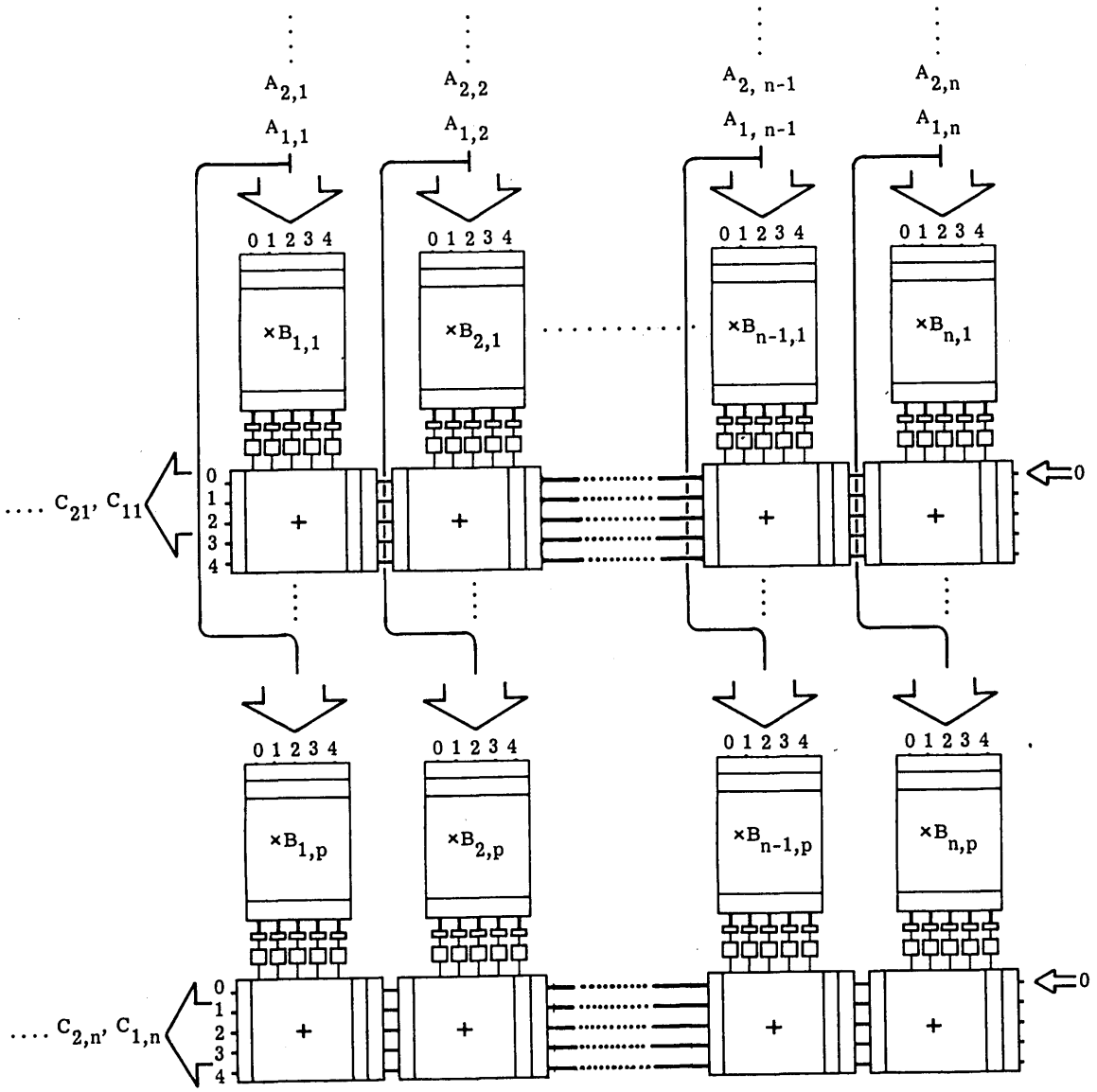


Fig. 15. Matrix multiplication.

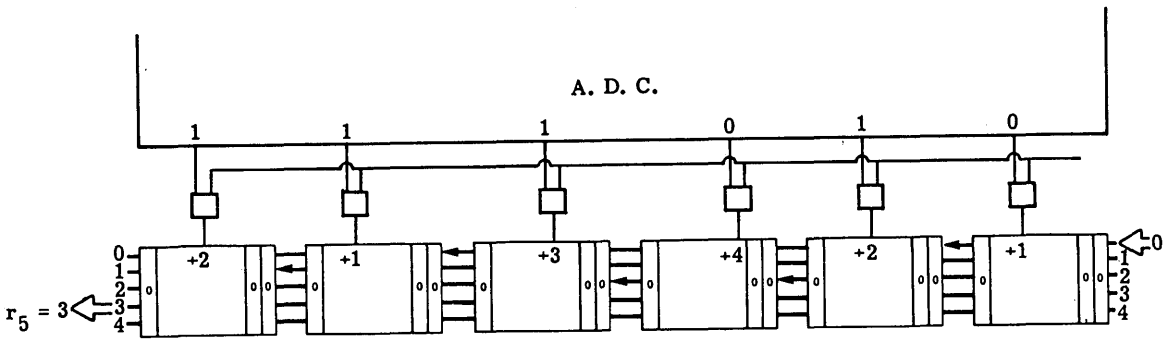


Fig. 16. Encoding into residue number system.

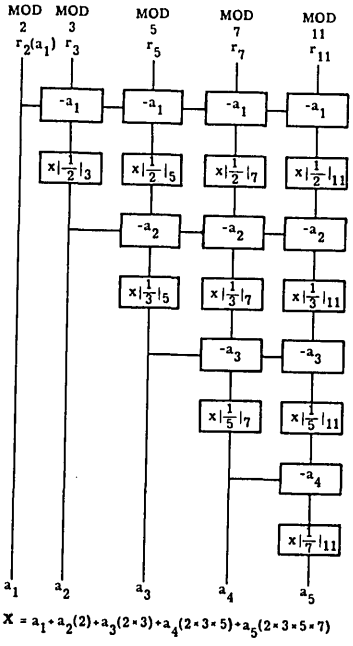
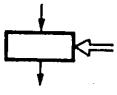


Fig. 17. Residue to mixed radix conversion.



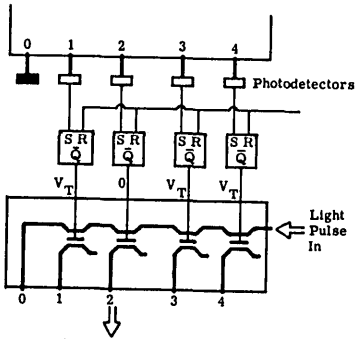


Fig. 18. Optical data register module.

$$58 = 1\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0,$$

$$|58|_5 = |1\times(2^5) + 1\times(2^4) + 1\times(2^3) + 0\times(2^2) + 1\times(2^1) + 0\times(2^0)|_5$$

$$= |1\times2 + 1\times1 + 1\times3 + 0\times4 + 1\times2 + 0\times1|_5$$

$$= |2 + 1 + 3 + 2|_5 = 3.$$

Since all the modules can be set in parallel, the time required for encoding is equal to one set time plus the propagation time through all the modules.

Decoding

Decoding a residue number is a more complicated operation than encoding. The most popular approach is to convert the residue number into the mixed radix system.[2] The reason is that the conversion procedure

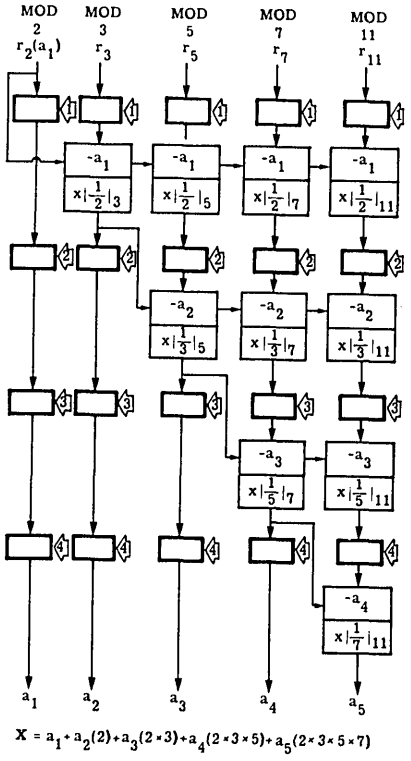


Fig. 19. Pipelined residue to mixed radix conversion.

can be performed with the same type of hardware used for the basic residue arithmetic computations. The algorithm is shown schematically in Fig. 17 for moduli 2, 3, 5, 7, 11. $|1/K|_{m_i}$ represents the multiplicative inverse where $|K \times |1/K|_{m_i}|_{m_i} = 1$. The drawback is that the procedure requires $N-1$ sequential steps where N is the number of moduli used. Since encoding and computation can be performed at a throughput rate of 1/(one set-reset time of a module), this sequential decoding procedure would seemingly be a bottleneck for the entire process. Fortunately, the conversion procedure can be pipelined. To pipeline the operation, it is necessary to delay synchronously the coefficients obtained earlier in the procedure such that all the coefficients would advance through the decoding procedure at the same rate. This necessary delay can be accomplished by setting an adder for $+K$, where K is the number to be stored temporarily. The number is recalled by sending a light pulse into the input port corresponding to 0. Alternatively, it can be achieved with the use of the simple data register module shown in Fig. 18. We also note that the multiplying factors $|1/m_i|_{m_j}$ are fixed, and they can be implemented by fixed maps. The design of a pipelined residue-to-mixed radix converter is illustrated schematically in Fig. 19. The input residue numbers are first stored in the data register modules (represented by boxes with bold lines). At the same time, the computation modules are set by r_2 for the $-a_1$ operation. Light pulses are then injected into the data registers to recall the residue numbers. The

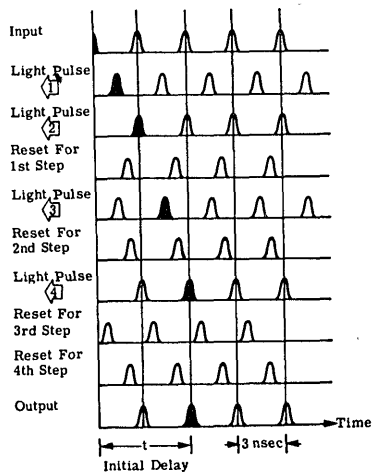


Fig. 20. Timing sequence for pipelined residue to mixed radix conversion.

light pulses propagate through the computation modules performing $-a_1$ operation and then through the fixed maps for $\times|1/2|_{m_i}$. The output is stored in the next set of data register modules, and the next computation modules are set for the $-a_2$ operation. Simultaneously, the second entry of the residue numbers are entered into the first set of data registers, ready for the first step of computation. The timing sequences of the input, the data recall light pulses, and the output are shown in Fig. 20. We see that no part of the converter sits idle at any time, and the conversion is performed at a constant throughput rate of 1/(one set-reset time) of a computation module. The pipelining concept can be applied to any sequential computation procedure. The encoding, computation, and decoding can therefore be performed at the same throughput rate. Assuming once again that the set-reset time of a computation module is 3 nsec, a numerical optical computer with a system throughput rate of over 300 MHz would be possible. We have also been able to apply this design concept for the computation of F.F.T., correlation and convolution, all with essentially the same throughput rate.

Residue to mixed radix conversion is a very important procedure in residue arithmetic. Besides decoding the output, the conversion is used for other important operations[2] such as sign detection, magnitude comparison, and overflow detection. Pipelining the procedure is therefore an important concept in an optical numerical computer using residue arithmetic. The original residue number may be stored in a cascade of data registers, while it is being converted into the mixed radix form for condition check. The residue number is moved down at the same rate as the conversion process, and the computation is continued after the checking is completed. Alternatively, after the residue numbers are converted into their mixed radix equivalent for sign detection or overflow detection, they can be converted back into the residue form for further computation. The inverse conversion (mixed radix to residue) can be achieved very easily and once again in one set time of the computation module.

Let us take the case where the moduli are 2, 3, 5, 7. The residue representation can be written as $x = (r_1,r_3,r_5,r_7)$ and the mixed radix representation as $x = [a_1,a_2,a_3,a_4] = a_1(1) + a_2(1 \times 2) + a_3(1 \times 2 \times 3) + a_4(1 \times 2 \times 3 \times 5)$. For example, to calculate the residue for modulo 3,

$$r_3 = |a_1|1|_3 + a_2|2|_3 + a_3|6|_3 + a_4|30|_3|_3.$$

Since $|30|_3 = |6|_3 = 0$, the expression can be simplified to $r_3 = |a_1|1|_3 + a_2|2|_3|_3$. The implementation is illustrated in Fig. 21.

Scaling

Overflow is a much more serious problem in residue arithmetic than in conventional arithmetic which utilizes a weighted number system. Not only is the residue for each individual modulus cyclic, the residue number system is also cyclic over its range M. The same representation is repeated for integers $K, K + M, K + 2M$, etc. Overflow detection is not automatic as with weighted number systems, and it is generally wise to avoid situations where overflow may occur.[16] For some computations, this would require a periodic downscaling of the residue numbers. To do this, division operations would be necessary. As pointed out earlier, general division cannot be carried out easily, and scaling by an arbitrary factor would not be practical. One can, however, scale a residue number by a factor equal to the value of one of the moduli or a product of two or more moduli. For example, for a system with moduli 2, 3, 5, 7, we want to scale down a number $X = 191 = [1, 2, 1, 2]$ by a factor of 7. Since the divisor 7 is also a modulus, the corresponding residue $|X|_7 = 2$ would be equal to the remainder when the number X is divided by 7.

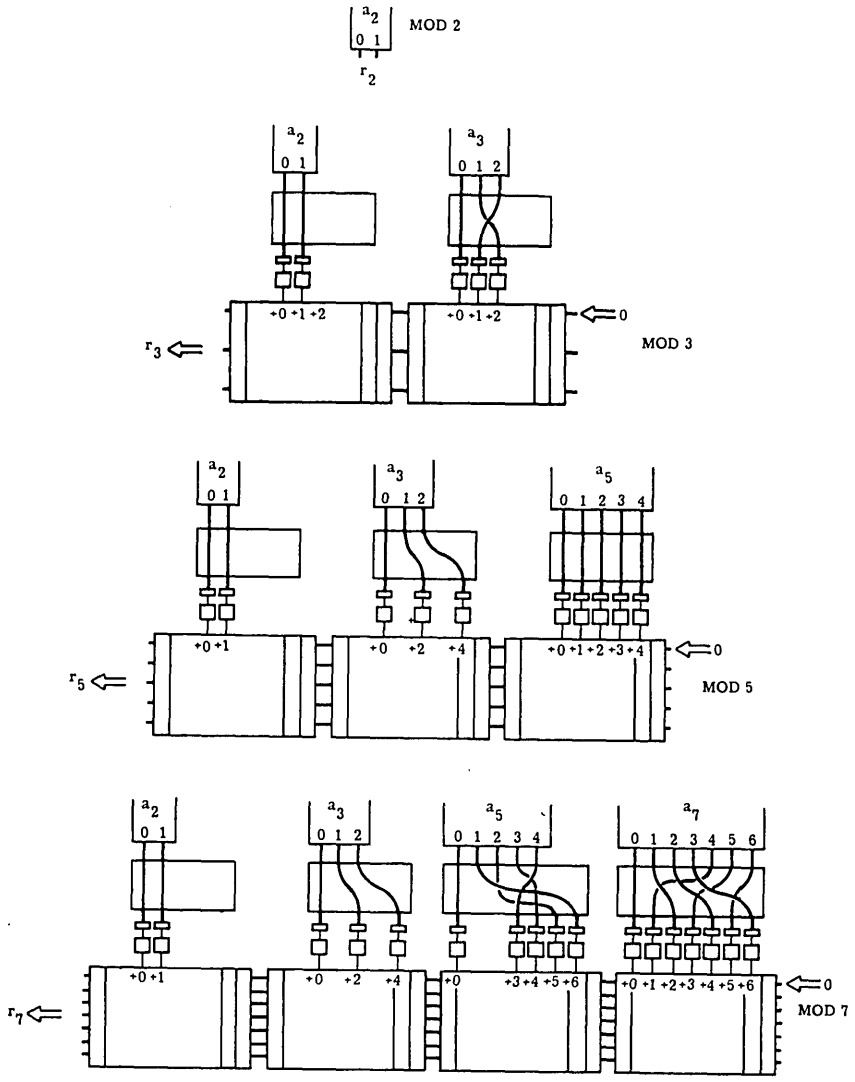


Fig. 21. Mixed radix to residue inverse conversion.

Therefore, $X - |X|_7$ is always exactly divisible by 7, and the homomorphic approach can be applied for the division operation. However, for modulus 7, the divisor is equal to 0, and division by 0 is not defined. The general approach is to proceed with the division while ignoring modulus 7. The residue of the quotient for modulus 7 is then obtained using the extension of base procedure.[2] It is essentially a residue to mixed radix conversion. The pipelined extension of base operation is shown schematically in Fig. 22. The entire scaling operation can be pipelined to maintain the throughput rate of 1/one set-reset time of a computation module.

Comments

The light sources to be used with the computation module are likely to be semiconductor pulse-modulated laser diodes, which are integrated with the module. This combined use of optical and electrical signals can be expected to continue as the numerical optical computer techniques evolve. It is therefore important to optimize the system designs with regard to the optical-electrical interfaces due to its impact on the computer speed and fabrication complexity.

There is a very useful feature in the use of computation modules for residue arithmetic that is shared by other implementations of the mapping approach. The operand and the operator are combined in a single representation. For example, to perform addition between an input value and a stored value with a conventional computer, the stored value has to be recalled from storage and entered with the input value into a fixed operator (adder). Implementing residue arithmetic with computation modules, the stored value is entered into the module as an operator (i.e., $+K$). The state of the module represents both the operand (K) and the operation (+). The module is therefore functioning simultaneously as the adder and the data storage device. This feature eliminates the need of a separate memory

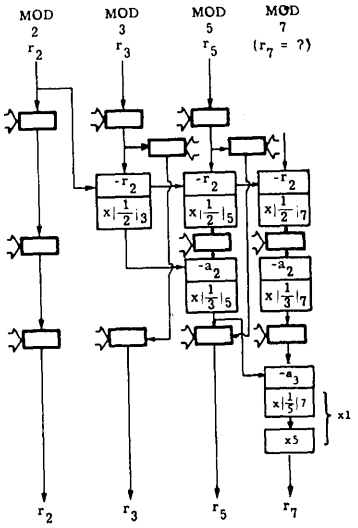


Fig. 22. Pipelined extension of base.

for values such as the coefficients of a reference function in correlation detection operations. Without the access time delay in reading out the stored values, the inputs can be processed at a very high rate, especially for computations that have to be performed repeatedly.

In this paper, we emphasized the use of the multipurpose computation module to demonstrate its versatility. We should note that some of the functions, such as binary to residue conversion, can be implemented with less complex devices. However, the use of a single mass-produced programmable module may in the end be more economical.

Conclusion

We have presented the concept of an optical programmable multipurpose computation module for residue arithmetic. A possible design for the module using directional coupler waveguide switches is also presented. The modules can be interconnected to perform various functions, from arithmetic computations to coding and decoding. The computation module design lends itself to parallel processing structure and pipelining. We have shown that through pipelining, the system throughput rate can be maintained at 1/one set-reset time of a computation module. Much of the hardware required for the construction of such a numerical optical computer has been demonstrated. However, the fabrication technology has not yet been developed to a stage where all the components can be put together in a small integrated package. Nevertheless, such a level of technology is attainable, and it raises the possibility of having a new generation of electrooptical computers, capable of performing, for example, a 1024-point FFT in microseconds instead of milliseconds.

We have presented material in this paper that is quite specific as to the number systems, architecture, and hardware realization. However, numerical optical computing is a broad field that encompasses many design concepts. Much future effort is needed in this area to develop new approaches, hardware components, system architecture, and applications.

The authors thank W. Holsztynski for his valuable discussions and comments. This research was supported by the Ballistic Missile Defense Advanced Technology Center under the direction of J. McKay.

References

1. A. Huang, Y. Tsunoda, J. W. Goodman, and S. Ishihara, Appl. Opt. **18**, 149 (1979).
2. N. S. Szabo and R. I. Tanaka, *Residue Arithmetic and Its Application to Computer Technology* (McGraw-Hill, New York, 1967).
3. P. W. Cheney, IRE Trans. Electron. Comput. **EC-11**, 63 (1961).
4. R. M. Guffin, IRE Trans. Electron. Comput. **EC-12**, 164 (1962).
5. D. K. Banerji, IEEE Trans. Comput. **23**, 1315 (1974).
6. D. Psaltis and D. Casasent, Appl. Opt. **18**, 163 (1979).
7. S. A. Collins, Jr., Proc. Soc. Photo. Opt. Instrum. Eng. **128**, 313 (1977).
8. A. Huang, Y. Tsunoda, and J. W. Goodman, Tech. Rept. 6422-1, Stanford Electronics Laboratories, Stanford, Calif. (1978).
9. M. Okada and K. Takizawa, IEEE J. Quantum Electron. **QE-15**, 82 (1979).
10. H. Taylor, Appl. Opt. **17**, 1493 (1978).
11. P. S. Cross, R. V. Schmidt, and R. L. Thornton, *Digest of Topical Meeting on Integrated and Guided Optics* (Optical Society of America, Washington, D.C., 1977), paper TUB1-1.
12. I. P. Kaminow, IEEE Trans. Microwave Theory Tech. **MTT-23**, 57 (1975).
13. H. L. Garner, IRE Trans. Electron. Comput. **EC-8**, 140 (1959).
14. H. Taylor, M. J. Taylor, and P. W. Bauer, in Ref. 11, paper TUC1-1.
15. F. Baisi and P. Maestrini, IEEE Trans. Comput. **C-27**, 1185 (1978).
16. Y. A. Keir, P. W. Chency, and M. Tannenbaum, IRE Trans. Electron. Comput. **EC-11**, 501 (1962).