A Single-Chip Pipelined 2-D FIR Filter Using Residue Arithmetic

Naresh R. Shanbhag, *Student Member, IEEE*, and Raymond E. Siferd, *Member, IEEE*

Abstract —**Presented in this paper are novel circuits and architecture for residue arithmetic. These circuits are aimed towards fast and area-efficient single-chip implementation of digital signal processors. This has been achieved by following an *algorithmic* approach as opposed to the conventional look-up table approach. As a result, substantial area savings have resulted. The circuits include the residue adder, residue multiplier, binary-to-residue converter, and residue-to-binary converter. Based on these circuits, a prototype single-chip, 3×3, finite impulse response (FIR), variable coefficient, linear-phase filter has been designed and fabricated in standard 2-μm CMOS technology. The filter has a pipelined architecture to increase the throughput. Testability in the form of scan-path registers has been incorporated. An interesting feature of this unique combination of residue arithmetic and scan-path testing is the possible trade-off available between the precision of the filter coefficients and the image data. The chip has a die size of 6.6×4.2 mm², dissipates 220 mW of power, and is synchronized with a 180-ns clock cycle.**

I. Introduction

DEMANDS for higher speeds from digital signal processing hardware are increasingly evident. Strategies for high-speed computation have evolved at the technological, layout, and architectural levels. Prominent among technological innovations are those that are gallium-arsenide (GaAs) based [1]. Even though GaAs technology looks promising in terms of attaining greater processing speeds than the well-established silicon technology, it is still maturing. Though some speed advantage is obtainable through modifications at the layout level [2], they are not substantial unless supplemented by a proper architecture. Therefore, most of the speed-enhancing techniques have been developed at the architectural level [3]–[5].

Pipelining [4] and parallel processing [5] are two major architectural options available for increasing the clock speed. Unless the computation is recursive in nature, pipelining can always be applied. Parallel processing, on the other hand, seems to depend more on the nature of the algorithm being implemented.

Yet another attractive option is the use of unconventional arithmetic like the residue number system (RNS) [6], which is gaining increasing usage due to its inherent parallelism. Most of the past work in this area has dealt with innovative circuits for residue applications [7]–[10] without any implementation. The cases where implementation has been successfully done [17], [25], [27], [28] have invariably resorted to multiple-chip implementation. For example, in [17], the design of a one-dimensional finite impulse response (FIR) filter, using residue arithmetic, is presented, which has been implemented using standard TTL IC packages. Another interesting implementation [25] is in multiple-valued logic for a one-dimensional filter, though it must be mentioned that 247 chips are needed. The reason for most implementations having multiple chips is that modular operations are realized through array structures like ROM's, RAM's, and PLA's, which are area expensive. We advocate a hybrid approach in this paper, where certain modules or their parts are realized by PLA's and the rest by an *algorithmic* (non-array based) circuit. This has enabled us to realize highly compact circuits with speed equal to or better than conventional array-based modular circuits.

In this paper, we present novel circuits for modular operations. Some of the circuits, which include the binary-to-residue converters, are based partially on PLA's. The others are completely algorithmic in nature. Using these circuits, the design and implementation of a single-chip two-dimensional FIR filter is presented.

The paper is divided into five more sections beginning with Section II, in which we give some preliminary results about residue arithmetic and 2-D filtering. In Section III, we present circuits for residue arithmetic, while Section IV contains the filter architecture. Timing and testability features of the chip is described in Section V, and finally we conclude with Section VI.

II. Preliminaries

Typically a residue number system has a predefined set of modulii $[m_1, m_2, \cdots, m_n]$, such that any decimal number X, which is less than the dynamic range M ($M = m_1 m_2, \cdots, m_n$), can be represented uniquely in this system as follows:

$$X = (X_1, X_2, \cdots, X_n)$$

Manuscript received June 4, 1990; revised October 23, 1990.

N. R. Shanbhag was with the Department of Electrical Engineering, Wright State University, Dayton, OH 45435. He is now with the Department of Electrical Engineering, University of Minnesota, Minneapolis, MN 55455.

R. E. Siferd is with the Department of Electrical Engineering, Wright State University, Dayton, OH 45435.

IEEE Log Number 9042382.

C	A	D
B	1	B
D	A	C

(a)

x(i-1,j-1)	x(i-1,j)	x(i-1,j+1)
x(i,j-1)	x(i,j)	x(i,j+1)
x(i+1,j-1)	x(i+1,j)	x(i+1,j+1)

(b)

Fig. 1. Filter (a) coefficient and (b) data window.

where

$$X_i = (X) \bmod m_i = |X|_{m_i}, \qquad i = 1,\cdots,n \tag{1}$$

are called the residue digits. The property which makes residue arithmetic attractive is that the addition and multiplication of two numbers can be carried out on their residue digits, independent of each other. In other words

$$X + Y = (X_1 \oplus_{m_1} Y_1, X_2 \oplus_{m_2} Y_2, \cdots, X_n \oplus_{m_n} Y_n) \tag{2}$$

$$X \times Y = (X_1 \otimes_{m_1} Y_1, X_2 \otimes_{m_2} Y_2, \cdots, X_n \otimes_{m_n} Y_n) \tag{3}$$

where $\oplus_{m_i}$ and $\otimes_{m_i}$ denote addition and multiplication modulo m_i.

The implemented 2-D FIR filter has a mask size of 3×3 with symmetric coefficients (Fig. 1(a)). Apart from generating linear-phase characteristics, symmetric coefficients offer a great deal of reduction in computational hardware. This is due to the fact that data (Fig. 1(b)), which have to be multiplied with the same coefficients, can be added first. The output $y(i,j)$ is computed as

$$\begin{aligned} y(i,j) = {} & A[x(i-1,j) + x(i+1,j)] \\ & + B[x(i,j-1) + x(i,j+1)] \\ & + C[x(i-1,j-1) + x(i+1,j+1)] \\ & + D[x(i+1,j-1) + x(i-1,j+1)] + x(i,j). \end{aligned} \tag{4}$$

It may be noted that if the data and the coefficients are represented in a residue number system with moduli set $[m_1, m_2, \cdots, m_n]$, then the residue digits of $y(i,j)$ are evaluated by substituting the residue digits of the data and coefficients into (4), with the addition and multiplication being modulo m_i. Thus n such computations can be carried out in parallel to generate all the residue digits of the output. For our filter the moduli set was [13, 11, 9, 7, 5, 4] with a dynamic range of 180180 (17.37 b).

III. Circuits

In this section, we present the design of all the modular arithmetic circuits, which include the binary-to-residue converter, residue multiplier, residue adder, and residue-to-binary converter. It will be seen that our approach is a hybrid between a totally array-based approach and an algorithmic approach. In fact, we have employed PLA's to realize logic functions with at most four inputs. For the case of residue multipliers we have used PLA's with input encoding. We also give an area and speed comparison between a complete PLA-based approach and our approach. As PLA's are known to be area efficient as compared to ROM's with their speeds being comparable, comparison with ROM-based structures has not been done. It must be mentioned that the speed estimations are based upon simulations on the event-driven logic-level simulator IRSIM, which provides conservative results as compared to an equivalent SPICE2 simulation. As we are interested in relative speeds, this conservatism is not crucial. The circuits are designed in standard 2-μm CMOS technology with typical MOSIS parameters. The supply voltage is 5 V.

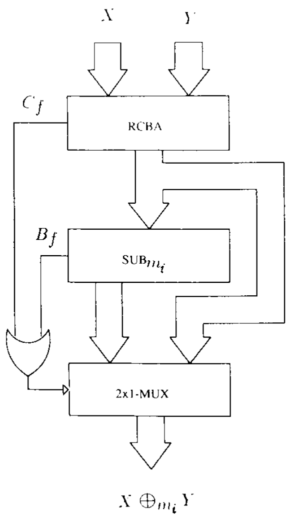


Fig. 2. Residue adder block diagram.

A. Modulo m_i Adder

The residue adder circuit proposed in [7] takes advantage of the fact that modular addition and subtraction form a finite cyclic group. Hence the adder [7] has an array of registers storing all possible residues and a complex rotate logic circuitry, controlled by the two operands, to select the correct residue as an output. It was found that not only is the adder area expensive, due to the presence of the registers and the rotate logic, but would also be slower. Our adder circuit, which is described next, is similar to the direct modulo adder described in [8]. It will be shown that our circuit offers a speed advantage as compared to the circuit in [8].

The residue adder circuit (Fig. 2) consists of a conventional n-bit ripple-carry binary adder (RCBA), which generates an $(n+1)$-bit output $(RCBA)$. Here n equals the number of bits required to represent the modulus m_i. Let

RCBA' represent the n-bit integer formed by ignoring the most significant bit (MSB) of *RCBA*, which is also the final carry (C_f) of RCBA. The next module (SUBm_i) subtracts m_i from *RCBA'*, by adding the two's complement of m_i to *RCBA'*. In this process an $(n+1)$-bit output ($SUBm_i$) is generated. Finally, a 2×1 multiplexor (MUX2×1) selects between *RCBA'* and $SUBm_i'$, where $SUBm_i'$ is the n-bit integer formed by ignoring the MSB (which is also the final borrow B_f) of $SUBm_i$, to generate the final output SMm_i. As both the inputs X and Y are residue digits, their sum *RCBA* would lie between 0 and $2m_i - 2$. If $RCBA < m_i$ then MUX2×1 selects *RCBA'* as the final output; otherwise it selects $SUBm_i'$. Hence, we see that there are two levels of logic (the RCBA and SUBm_i) through which the data have to be processed. This is also true for the adder circuit in [8]. The advantage of our circuit is that we have a very elegant logic for controlling the MUX2×1. In fact, it can be seen that

$$C = C_f \vee B_f \tag{5}$$

where C_f and B_f are the final carry and the final borrow of RCBA and SUBm_i, respectively. Here, $C=1$ results if *RCBA* is greater than or equal to m_i. A brief analysis below shows that both C_f and B_f are required for the correct generation of the final result. If $RCBA < m_i \leqslant 2^n - 1$ then $C_f = 0$, $B_f = 0$, $C = 0$ and $RCBA' = RCBA$ is selected as the final output. If $m_i \leqslant RCBA \leqslant 2^n - 1$ then $C_f = 0$, $B_f = 1$, $C = 1$ and hence $SUBm_i'$ is the final result. Finally, if $2^n - 1 < RCBA$ then $C_f = 1$, $B_f = 0$, $C = 1$ and therefore $SUBm_i'$ is again selected as the final output.

The speed advantage of our circuit is derived from the fact that the ith bit of the second logic level (SUBm_i) starts processing as soon as the corresponding bit of the first level (RCBA) is ready. In [8], on the other hand, the second level has to wait for the final carry from the first level in order to start processing.

The circuit for SUBm_i was developed by modifying a ripple-carry binary adder with one of inputs permanently set equal to the two's complement of m_i. This way designing SUBm_i for any m_i becomes trivial, and at the same time it gives us an elegant way (from (5)) to find out whether *RCBA* is greater than or equal to m_i.

Area comparisons between our circuit and a PLA-based circuit, for $m_i = 13$, shows a 51% savings in area. This reduction is due to the fact that the PLA realization requires the minimization of a 256-row truth table (eight-input, four-output PLA). Speed comparisons, for a 0.2-pF load, show that our circuit is negligibly slower (about 2 ns) than the PLA-based circuit. This enormous area savings was instrumental in achieving a single-chip implementation.

B. Modulo m_i Subtractor

The residue subtractor, in [7], is based on the same principles as the residue adder described in [7]. Therefore, our residue subtractor circuit, which is also similar to the residue adder described in Section III-A, has a similar degree of advantage over the residue subtractor in [7].

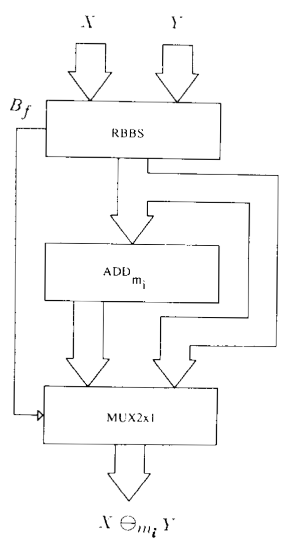


Fig. 3. Residue subtractor block diagram.

This circuit was needed in the residue-to-binary converter. In a fashion similar to the design of the modulo m_i adder, we first subtracted the two inputs (Fig. 3) in a conventional ripple-borrow binary subtractor (RBBS). The number m_i was then added to the output of RBBS (*RBBS*) in the circuit ADDm_i. The final borrow from RBBS was used to select between *RBBS* and the output of ADDm_i (ADD_{m_i}). Though a PLA realization of the modulo m_i subtractor was not explicitly done, our experience with the modulo m_i adder would indicate that area savings of similar magnitude had resulted due to this circuit. For $m_i = 13$, the subtractor occupied an area of 72420 μm^2 and had a delay of 20 ns for a load of 0.2 pF.

C. Modulo m_i Multiplier

Residue multipliers available in literature are the square-law multiplier [9], [10] (Fig. 4(a)) and the index calculus multiplier [17] (Fig. 4(b)). In the square-law multiplier (Fig. 4(a)), apart from two squaring ROM's, we also need three residue adders. This was found to be expensive in terms of area, for our application. The index calculus multiplier (Fig. 4(b)) requires three ROM's and a high precision at the output of the index ROM's. Other efficient multiplier circuits [11]–[13] require a specialized moduli set of the type $[2^{n-1}, 2^n, 2^{n+1}]$. Therefore, we opted for a PLA-based multiplier with input encoding. This circuit was found to be most area efficient.

An input encoding scheme [14] was incorporated for reducing the number of product terms in the PLA. In a conventional PLA, we feed the input and its complement to the AND plane. In an encoded PLA, we pair the inputs, say x_1 and x_2, and feed $x_1 \vee x_2$, $x_1 \vee \bar{x}_2$, $\bar{x}_1 \vee x_2$, and $\bar{x}_1 \vee \bar{x}_2$. In both cases the number of input lines finally

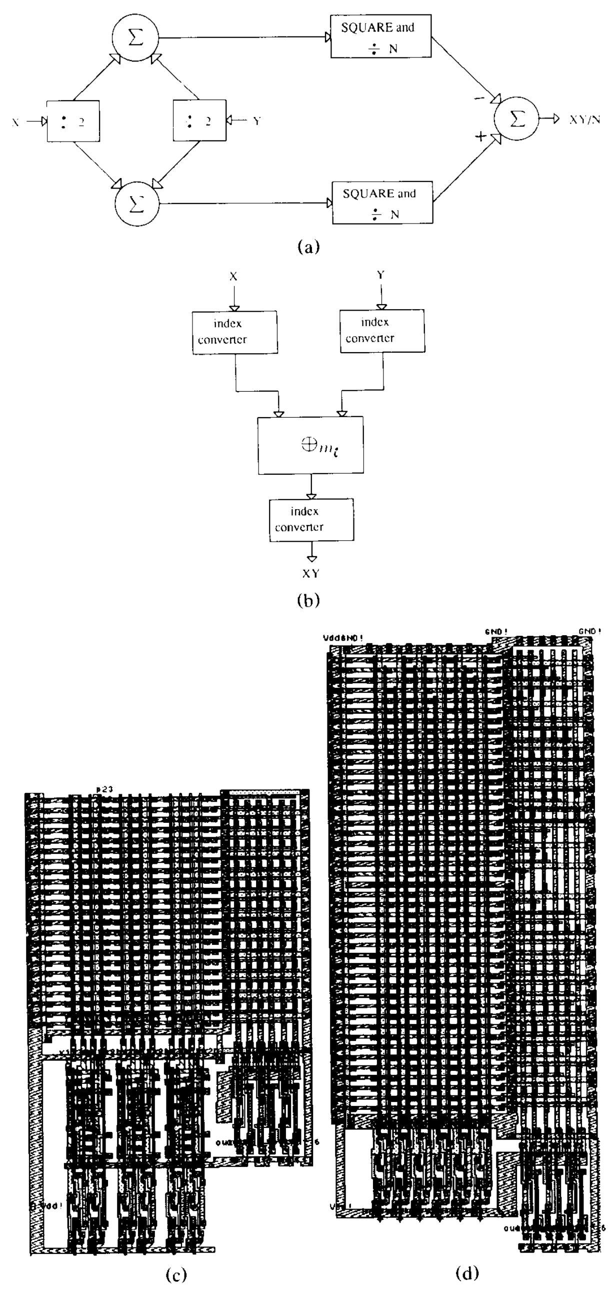


Fig. 4. Residue multipliers: (a) square-law, (b) index calculus, and (c) an encoded PLA, and (d) a conventional PLA realization of a modulo 7 multiplier.

passing through the AND plane is the same. Hence, the width of both PLA's is the same and if the input pairings are optimal then a substantial reduction in the number of product terms makes the encoded PLA smaller in area. For the sake of comparison, we show the layouts of the encoded and conventional PLA (Fig. 4(c)) accomplishing the same logic function (modulo 7 multiplier). It is to be noted that this reduction in area is partially offset by the presence of encoders and also depends on the logic function being realized.

In order to determine the optimal input pairings, there are quite a few heuristic algorithms [14]. In our application, and as would be the case in most residue number applications, as the bit width of the inputs was small (a maximum of four for each operand), an exhaustive search was resorted to for the determination of optimal input pairings. The software package called ESPRESSO-MV [15] was employed to minimize the multiplier truth table with input pairings. During the course of the simulations, an interesting feature was observed with regard to the optimal input pairings of multipliers of different moduli. It was observed that for a particular moduli, say m_i, both the modulo m_i adder and modulo m_i multiplier have the same optimal input pairings. In addition, the optimal input pairing, for a two n-bit operand residue multiplier or adder is generated by combining the ith bits of each operand. Though we are not providing any theoretical basis for this observation, it can be used as an heuristic guide.

Area and delay comparisons show that for $m_i = 13$ a net reduction in area (taking into account the area occupied by the encoders) of 21% was achieved, while for the same load capacitance of 0.2 pF, the delays were nearly equal (47 ns). The area reduction for a residue multiplier is highly dependent on the modulus. In our case, the multipliers for all other moduli had reductions in area of a similar magnitude.

D. Binary-to-Residue Converter

In [16], an algorithm for the binary-to-residue converter (BTOR) is formulated, for a general base. It requires two levels of modulo m_i adders, though for binary input the first level may be eliminated. In any case, a simple analysis showed that for 8-b input data, we need seven modulo m_i adders and eight MUX2×1's. Our circuit uses the same technique as the one suggested by the authors in [17].

We represent an 8-b number X as follows:

$$X = 2^4 B_M + B_L \tag{6}$$

where B_M and B_L are the four most and least significant bits of X, respectively. Therefore

$$\begin{aligned}(X)\bmod(m_i) &= (2^4 B_M + B_L)\bmod(m_i)\\ &= \left[(2^4 B_M)\bmod(m_i)\right] \oplus_{m_i} \left[(B_L)\bmod(m_i)\right]\\ &= \left[(2^4)\bmod(m_i) \otimes_{m_i} (B_M)\bmod(m_i)\right]\\ &\quad \oplus_{m_i}\left[(B_L)\bmod(m_i)\right].\end{aligned} \tag{7}$$

As B_M and B_L are both 4-b numbers, two PLA's with at most 16 product terms each and a modulo m_i adder, described in Section III-A, would be required. The circuit block diagram (Fig. 5) shows that PLA1 computes the first term $((2^4)\bmod(m_i)\otimes_{m_i}(B_M)\bmod(m_i))$ while PLA2 computes the second $((B_L)\bmod(m_i))$. This technique resulted in a very compact circuit.

A straightforward PLA realization of BTOR for $m_i = 13$ showed that an area of 619 μm×774 μm would be required. Area and speed comparisons for $m_i = 13$ were made. Thus a 71% area reduction was achieved with this technique. Speed comparisons show that the new circuit is about 8 ns faster than the previous one, again for a load of 0.2 pF. As three data streams had to be converted into residue, in each cycle, this area reduction was instrumental in achieving a single-chip implementation of the filter.

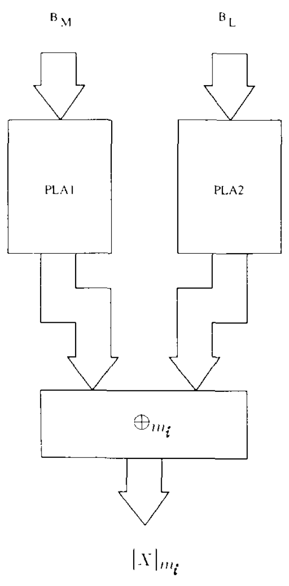


Fig. 5. Binary-to-residue converter.

E. Residue-to-Binary Converter

Many techniques for output translation exist in the literature [16]–[22]. All of them are based either on the mixed radix conversion (MRC) method [6] or the Chinese Remainder Theorem (CRT) [21]. The method described in [16] is similar to MRC and is sequential in nature. In [17], the authors present a method based on the CRT using the vector multiplication algorithm of Peled–Liu [23]. Similarly, in [18], a method based on MRC and the algorithm of Peled–Liu [23] is presented. Some methods [19] are restricted to a special moduli set of the type $[2^n+1, 2^n, 2^n-1]$. A unique technique, based on the CRT [21], is presented in [20]. Here modulo M addition, where M is the dynamic range, is replaced by a simpler FRAC operation. This operation deals with the fractional part of the operands of the modulo M addition. Yet another method involving parallelization of the MRC algorithm is given in [22].

The above-mentioned methods cannot be pipelined easily as they either require a large number of latches or are sequential in nature or both. Our technique, though, based on the MRC method, can be easily pipelined. This is because our method relies on combining two moduli at a time (Fig. 6(a)) to generate a tree structure. Pipelining latches can then be placed on appropriate branches. In this particular implementation, moduli pairs (13,4), (11,5), and (9,7) were combined in the first level to generate a new moduli set (52,55,63). In the second level, moduli 52 and 55 were combined to generate representation in modulo 2860, which was then combined with modulus 63,

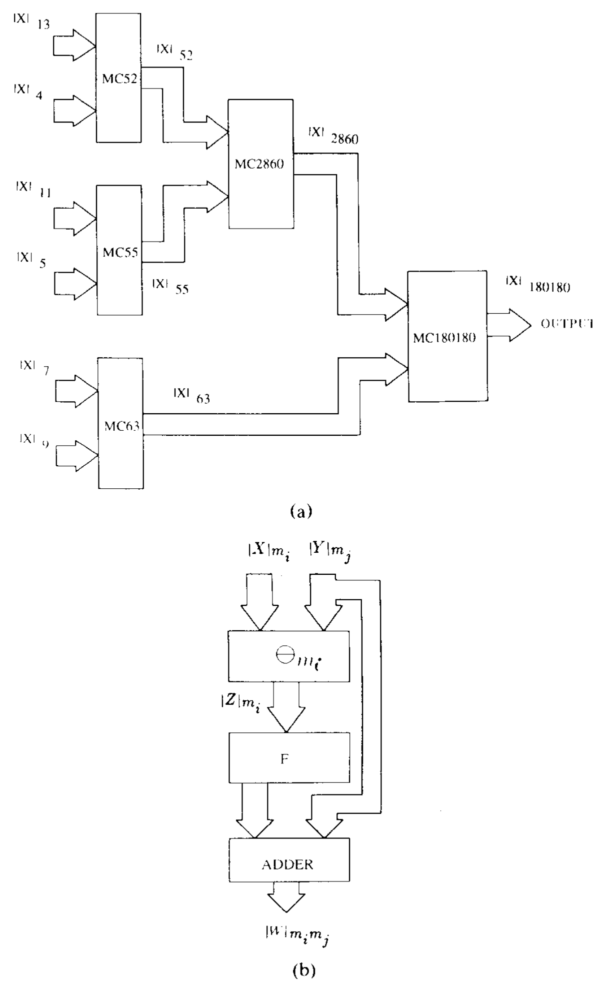


Fig. 6. Residue-to-binary converter: (a) system architecture and (b) block diagram for achieving the combination of two moduli.

in the last level, to obtain the final answer in modulo 180180.

The combination of two moduli was done using the conventional mixed radix algorithm [7] (Fig. 6(b)), where the function F is defined as

$$F = m_j \cdot \left[|Z|_{m_i} \otimes_{m_i} \left| \frac{1}{m_j} \right|_{m_i} \right] \tag{8}$$

where $|1/m_j|_{m_i}$ denotes the multiplicative inverse of m_j with respect to m_i. Hence, it is important that the moduli pair (m_i, m_j), which are to be combined, should be such that the smaller of the two should have a multiplicative inverse with respect to the other. This can be seen to be true in our case. A PLA was employed to realize the function F. For MC180180, it was found that this PLA would have 2857 terms after minimization, which was too expensive in terms of area. Therefore, we split this PLA into two, using Theorem 1, which is stated below.

Theorem 1: Any four positive integers A, B, C, and D always satisfy the following relation:

$$(A \oplus_D B) \cdot C = (A \cdot C) \oplus_{(D \cdot C)} (B \cdot C). \tag{9}$$

The proof of this Theorem is given in the Appendix.

Let X be the input to the PLA, which in this case has 12 inputs. Therefore, following an approach similar to (6) we write X as

$$X = 2^6 X_M + X_L \tag{10}$$

where X_M is the most significant and X_L is the least significant six bits of X. We wish to multiply X by the number 227 modulo 2860, where 227 is the multiplicative inverse of 63 with respect to 2860, and then multiply the result by 63, to generate the output of the PLA. Therefore, we carry out the residue multiplication first

$$\begin{aligned} X \otimes_{2860} 227 &= (2^6 X_M + X_L) \otimes_{2860} 227 \\ &= (2^6 X_M \otimes_{2860} 227) \oplus_{2860} (X_L \otimes_{2860} 227) \\ &= (X_M \otimes_{2860} 228) \oplus_{2860} (X_L \otimes_{2860} 227) \end{aligned} \tag{11}$$

then multiply (11) by 63

$$\begin{aligned} &(X \otimes_{2860} 227) \cdot 63 \\ &\quad = [(X_M \otimes_{2860} 228) \oplus_{2860} (X_L \otimes_{2860} 227)] \cdot 63. \end{aligned} \tag{12}$$

We see that the right-hand side of (12) is of the same form as the left-hand side of (9), where $A = X_M \otimes_{2860} 228$, $B = X_L \otimes_{2860} 227$, $C = 63$, and $D = 2860$. Hence, employing identity (9) we get

$$\begin{aligned} &(X \otimes_{2860} 227) \cdot 63 \\ &= [\{(X_M \otimes_{2860} 228) \cdot 63\} \oplus_{180180} \{(X_L \otimes_{2860} 227) \cdot 63\}]. \end{aligned} \tag{13}$$

We realized the expressions $(X_M \otimes_{2860} 228) \cdot 63$ and $(X_L \otimes_{2860} 227) \cdot 63$ by two PLA's. It may be noted that through this exercise we have split the original 12-input PLA into two six-input PLA's. The total number of product terms in the two resultant PLA's was 111, a reduction of 96%. Through this process we have enhanced the speed of the circuit too, though we pay the penalty of having a large modulo adder.

IV. The Filter Architecture

Filter architectures based on RNS are known to be efficient for variable coefficient filters, while the combinatorial architecture of [23] gives higher speed/cost ratios for fixed-coefficient filters. This is because the latter eliminates the need for multiplication. In [24], the author provides a hybrid architecture, which combines the advantages of both the architectures, though it is still applicable only for fixed coefficient filters. We chose a completely RNS-based approach as we were interested in a variable coefficient filter. Moreover, as we wished to avoid the use of array-based structures as much as possible, this approach was quite suitable.

Previous RNS-based filter designs [17], [25], [26] were mainly for one-dimensional filtering. The existing two-

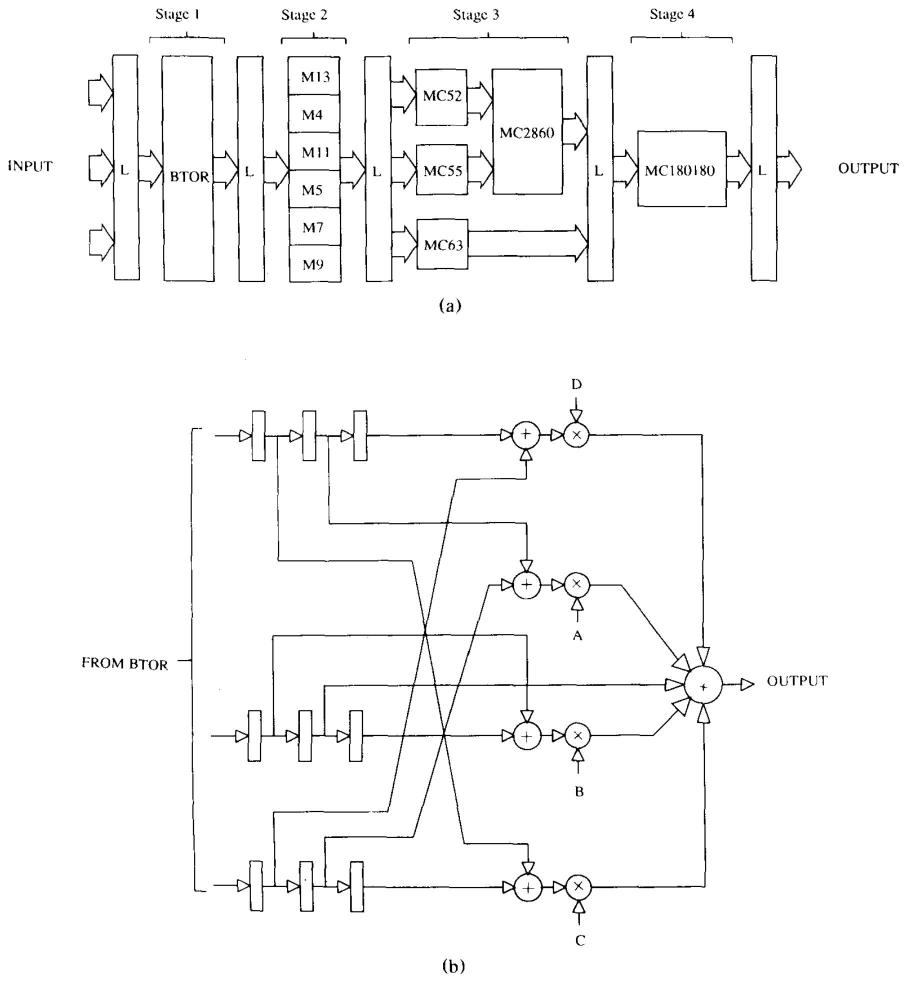


Fig. 7. Filter: (a) architecture and (b) stage 2 details.

dimensional RNS filters [27], [28] are not single-chip implementations. A simple analysis in [17] showed that for most practical purposes a dynamic range of 17 to 18 b is sufficient, for an 8-b data and coefficient representation. Therefore, we chose the moduli set [4, 5, 7, 9, 11, 13], which gives a dynamic range of 180180 (17.37 b).

The architecture of the filter (Fig. 7(a)) is similar to that in [25], though ours is a two-dimensional filter. It can be seen that the chip is pipelined into four stages. In order to conserve space, we have used dynamic latches. In the first stage (stage 1) binary data are converted into residue. As we use a 3×3 window (Fig. 1), in each clock cycle we input three new 8-b data. As each modulus requires three converters, the first stage contains 18 BTOR's in all.

The actual filtering operation takes place in the second stage (stage 2). The block diagram (Fig. 7(b)), of the second stage of one modulus shows that three sets of latches are required in order to form the 3×3 window, which are then processed according to (4). For most image processing applications it is desirable that the filter have linear-phase characteristics. This is the reason for having symmetric coefficient mask (Fig. 1). This symmetry also resulted in considerable area savings. As compared to other stages, this one was the slowest. Given the area restrictions due to the fabrication die size (6.6×4.2 mm^2), it was not possible to pipeline this section any further.

The third and fourth stages convert the result of the filtering operation back to binary. In the third stage (stage 3), moduli pairs (4, 13), (5, 11), and (7, 9) are combined in MC52, MC55, and MC63, respectively. In this stage itself, we combine the outputs of MC52 and MC55 in MC2860. In the last stage (stage 4), the outputs of MC63 and MC2860 are combined in MC180180, to generate the final 18-b answer.

It must be mentioned that the area reductions, realized by the unique circuits presented in Section III, played a crucial role in the successful realization of this architecture. The testability and timing features are described next.

V. Timing and Testability Features

The chip is synchronized with a two-phase nonoverlapping clock generated on chip, from a single clock input. All latches are therefore dynamic and two phase. In the first phase ϕ_1, the primary inputs and the output of each stage of the pipeline are loaded into the master section of the corresponding latch. Each stage of the pipeline then receives inputs for the current cycle, in the second phase (ϕ_2). Hence, each stage has ϕ_2 of the current cycle and ϕ_1 of the next cycle to carry out the computation for the current cycle. In the first three cycles, we input the coefficients through the normal data input and activate a LOAD signal to store them. Subsequently, three 8-b data are input and an output is obtained every cycle. The maximum clock rate was measured to be 5.6 MHz. Employing the scan-path facility, described later, the speed of each stage was measured independently. As predicted by the simulations, stage 2 had a maximum delay of 180 ns. Stages 1, 3, and 4 had delays of 130, 110, and 130 ns, respectively. As mentioned before, given slightly less area restrictions, we could have introduced an extra stage of pipeline in stage 2. Yet, given the slower CMOS technology, it compares well with the clock rate of twenty million operations per second reported in [27], which had utilized the faster emitter-coupled logic (ECL). Also, due to ours being a single-chip implementation, significantly less power is being consumed per operation. In fact, the power dissipation was measured to be 220 mW at the maximum clock rate.

Incorporation of testability features in VLSI chips is a recognized necessity. We have used the scan-path technique [29] to enhance the testability of our chip. After fabrication, extensive use of this facility was done to determine the correct operating conditions of the chip. In addition, it enabled us to determine the maximum speed of each stage of the pipeline. For implementing scan-path testing, special latches had to be introduced. These latches (Fig. 8) passed data from the BTOR's in their normal mode ($SCAN = 0$) of operation. In the scan mode ($SCAN = 1$), they were configured into a shift register with an external input (SCANIN) and an external output (SCANOUT). These scan latches were placed at the inputs of stages 2, 3, and 4. The control signals ϕ_{11} and ϕ_{12} were generated by the following logic expression:

$$\phi_{11} = \overline{SCAN \vee \overline{\phi 1} \vee \phi 2} \tag{14}$$

$$\phi_{12} = \overline{\overline{SCAN} \vee \overline{\phi 1} \vee \phi 2}. \tag{15}$$

An interesting feature introduced by this unique combination of scan-path testing and residue arithmetic is the possibility of loading filter coefficients with precision greater than 8 b. This can be done through the SCANIN input of stage 2, where we serially input the coefficient bits. As the coefficients have to be input in residue form, any degree of coefficient precision can be achieved. The precision of the data would then have to be reduced correspondingly, in order to generate the results within the dynamic range. This feature was actually confirmed during the testing phase, when coefficients with a value of 300 (greater than the maximum of 255 allowed by the 8-b data input channels) were input.

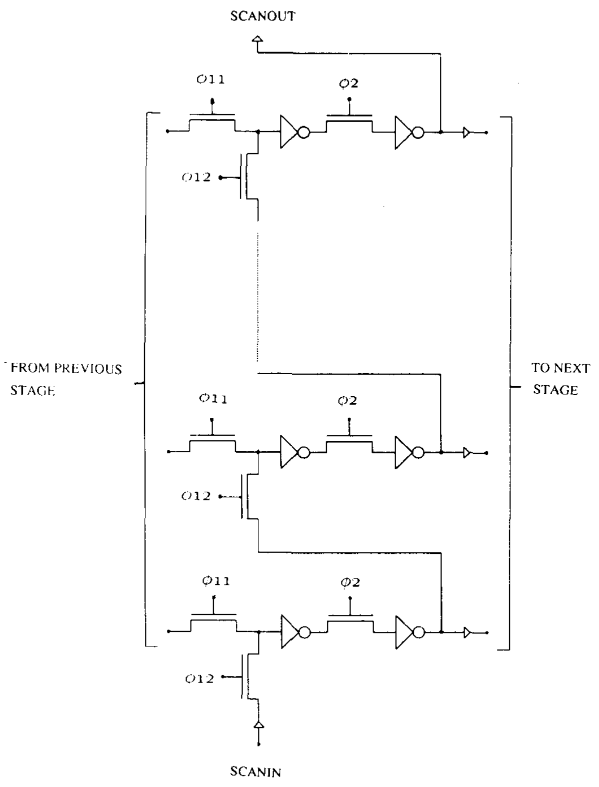


Fig. 8. Scan-path testing scheme.

VI. Conclusions

Design of certain novel residue arithmetic circuits is presented. Unlike most existing residue circuits, these have been built with the philosophy of having minimum array-based structures in them. Therefore, some circuits are completely algorithmic (nonarray based), others PLA-based, and the remaining are a hybrid of the two approaches. The residue adder, for example, is a completely algorithmic structure, while the residue multiplier is realized using an encoded PLA. For residue-to-binary converter, we employed the conventional MRC algorithm in a different way. The resultant tree-like architecture was found to be highly convenient for pipelining. Given the fact that it is the input/output conversion in RNS-based systems that form a major bottleneck when aiming for a high throughput, this feature of our technique is highly desirable. Theorem 1 was employed to split a large PLA in the module MC180180, which resulted in a 96% reduction in the number of product terms. Area and speed comparisons are presented for a completely array-based approach and ours. While the reductions in area

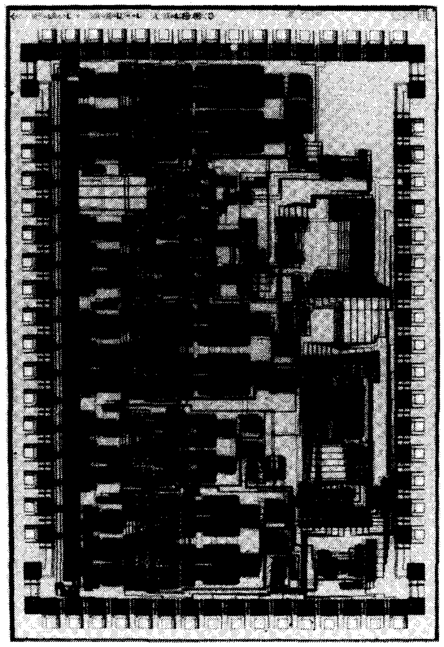


Fig. 9. Photomicrograph of the filter chip.

range from 21% to 71%, the speed is more or less the same.

These circuits were designed with the aim of realizing a single-chip implementation of a signal processor based on residue arithmetic. It can be seen that this is not the case with existing implementations [27], [28]. Employing these circuits, we have successfully fabricated and tested a single-chip two-dimensional FIR filter with variable coefficients. The filter has a 3×3 window with symmetric coefficients. The chip has been fabricated in a 64-pin dual-in-line package with a die size of 6.6×4.2 mm^2, using the 2-μm CMOS technology from MOSIS. It has a 5-V power supply. The chip operates correctly at a maximum clock frequency of 5.6 MHz and dissipates 220 mW of power. Given the difference in technology, the speed achieved compares well with that of [27], though our power consumption-per-logic operation is clearly superior due to the single-chip implementation. The chip has a pipelined architecture and incorporates scan-path testing. This combination of scan-path testing along with residue arithmetic has afforded a way of trading off coefficient precision with that of the data. Since the coefficients are variable it can be used in an adaptive environment. A photomicrograph of the chip is shown in Fig. 9.

We envisage substantial improvements in speed by incorporating carry-select adders instead of the ripple-carry adder in the circuit modules. Though we have implemented a 3×3 window, it is possible to realize higher order filters using the same architecture and with a larger die size or a 1-μm CMOS technology. With an extra off-the-chip adder unit, we can implement a higher, e.g., 6×6, order filter with four of the currently designed chips.

Appendix

To prove Theorem 1 we need the following Lemma.

Lemma 1: Any three positive integers A, B, and D, each of which is divisible by C, satisfy the following relation:

$$A \oplus_D B = \left[\frac{A}{C} \oplus_{D/C} \frac{B}{C}\right] \cdot C. \qquad \text{(A1)}$$

Proof: Let

$$\frac{A}{C} + \frac{B}{C} = E. \qquad \text{(A2)}$$

Given any E, the following always holds true:

$$\frac{nD}{C} \leqslant E < \frac{(n+1)D}{C} \qquad \text{(A3)}$$

for some $n = 0, 1, 2, \cdots$. Again, from (A2) we get

$$\left(\frac{A}{C} + \frac{B}{C}\right) \operatorname{mod} \left(\frac{D}{C}\right) = (E) \operatorname{mod} \left(\frac{D}{C}\right)$$

$$\left(\frac{A}{C}\right) \oplus_{D/C} \left(\frac{B}{C}\right) = E - \frac{nD}{C}. \qquad \text{(A4)}$$

Multiplying (A4) by C we get

$$\left[\left(\frac{A}{C}\right) \oplus_{D/C} \left(\frac{B}{C}\right)\right] \cdot C = EC - nD$$
$$= (A + B) - nD$$
$$= (A \oplus_D B). \qquad \text{Q.E.D.}$$

The proof of Theorem 1 is now trivial.

Proof of Theorem 1: Let $A' = A \cdot C$, $B' = B \cdot C$, and $D' = D \cdot C$. Therefore, A', B', and D' are divisible by C and hence by Lemma 1, we get

$$A' \oplus_{D'} B' = \left(\frac{A'}{C} \oplus_{D'/C} \frac{B'}{C}\right) \cdot C$$
$$= (A \oplus_D B) \cdot C. \qquad \text{Q.E.D.}$$

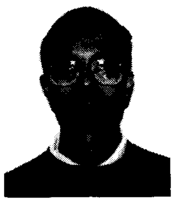
Acknowledgment

The authors wish to acknowledge the help given by L. Peterson and R. Hohne during the chip testing. The constructive suggestions of the reviewers are also appreciated.

References

[1] M. Ida, N. Kato, and T. Takada, "A 4-Gbit/s GaAs 16:1 multiplexer/1:16 demultiplexer LSI chip," *IEEE J. Solid-State Circuits*, vol. 24, pp. 928–932, Aug. 1989.
[2] C. Longway and R. Siferd, "A doughnut layout style for improved switching speed with CMOS VLSI gates," *IEEE J. Solid-State Circuits*, vol. 24, pp. 194–198, Feb. 1989.
[3] K. K. Parhi and D. G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters—Part I: Pipelining using scattered look-ahead and decomposition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 1099–1117, July 1989.

[4] M. R. Santoro and M. A. Horowitz, "SPIM: A pipelined 64×64-bit iterative multiplier," *IEEE J. Solid-State Circuits*, vol. 24, pp. 487–493, Apr. 1989.
[5] R. Sharma *et al.*, "A 6.75-ns 16×16-bit multiplier in single-level-metal CMOS technology," *IEEE J. Solid-State Circuits*, vol. 24, pp. 922–927, Aug. 1989.
[6] N. S. Szabo and R. I. Tanaka, *Residue Arithmetic and its Application to Computer Technology*. New York: McGraw-Hill, 1967.
[7] D. K. Banerji, "A novel implementation method for addition and subtraction in residue number systems," *IEEE Trans. Comput.*, vol. C-23, pp. 106–109, Jan. 1974.
[8] M. A. Soderstrand, "A new hardware implementation of modulo adders for residue number systems," in *Proc. 26th Midwest Symp. Circuits Syst.*, 1983, pp. 412–415.
[9] M. A. Soderstrand and E. L. Fields, "Multipliers for residue number arithmetic digital filters," *Electron. Lett.*, vol. 13, no. 6, pp. 164–166, Mar. 1977.
[10] M. A. Soderstrand and C. Vernia, "A high-speed low-cost modulo P_i multiplier with RNS arithmetic applications," *Proc. IEEE*, vol. 68, pp. 529–532, Apr. 1980.
[11] F. J. Taylor, "Large moduli multipliers for signal processing," *IEEE Trans. Circuits Syst.*, vol. CAS-28, pp. 731–736, July 1981.
[12] F. J. Taylor and C. H. Huang, "An autoscale residue multiplier," *IEEE Trans. Comput.*, vol. C-31, pp. 321–325, Apr. 1982.
[13] T. R. N. Rao and A. K. Trehan, "Binary logic for residue arithmetic using magnitude index," *IEEE Trans. Comput.*, vol. C-19, pp. 752–757, Aug. 1970.
[14] T. Sasao, "Input variable assignment and output phase optimization of PLA's," *IEEE Trans. Comput.*, vol. C-33, pp. 879–894, Oct. 1984.
[15] R. Rudell and A. Sangiovanni-Vincentelli, "Multiple-valued minimization for PLA optimization," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, pp. 727–751, Sept. 1987.
[16] D. K. Banerji and J. A. Brzozowski, "On translation algorithms in residue number systems," *IEEE Trans. Comput.*, vol. C-21, pp. 1281–1285, 1972.
[17] W. K. Jenkins and B. J. Leon, "The use of residue number systems in the design of finite impulse response digital filters," *IEEE Trans. Circuits Syst.*, vol. CAS-24, pp. 191–201, Apr. 1977.
[18] A. Baraniecka and G. A. Jullien, "On decoding techniques for residue number system realizations of digital signal processing hardware," *IEEE Trans. Circuits Syst.*, vol. CAS-25, pp. 935–936, Nov. 1978.
[19] F. J. Taylor and A. S. Ramanarayanan, "An efficient residue-to-decimal converter," *IEEE Trans. Circuits Syst.*, vol. CAS-28, pp. 1164–1169, Dec. 1981.
[20] M. A. Soderstrand, C. Vernia, and J-H Chang, "An improved residue number system digital-to-analog converter," *IEEE Trans. Circuits Syst.*, vol. CAS-30, pp. 903–907, Dec. 1983.
[21] D. E. Knuth, "Seminumerical algorithms," in *The Art of Computer Programming*, vol. 2. Reading, MA: Addison-Wesley, 1969, pp. 248–257.
[22] C. H. Huang, "A fully parallel mixed-radix conversion algorithm for residue number applications," *IEEE Trans. Comput.*, vol. C-32, pp. 398–402, Apr. 1983.
[23] A. Peled and B. Liu, "A new hardware realization of digital filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-22, pp. 462–465, Dec. 1974.
[24] W. K. Jenkins, "A highly efficient residue-combinatorial architecture for digital filters," *Proc. IEEE*, vol. 66, pp. 700–702, June 1978.
[25] M. A. Soderstrand and R. A. Escott, "VLSI implementation in multiple-valued logic of an FIR digital filter using residue number system arithmetic," *IEEE Trans. Circuits Syst.*, vol. CAS-31, pp. 5–25, Jan. 1986.
[26] M. A. Soderstrand, "A high-speed low-cost recursive digital filter using residue number arithmetic," *Proc. IEEE*, vol. 65, pp. 1065–1067, July 1977.
[27] C. H. Huang, D. G. Peterson, H. E. Rauch, J. W. Teague, and D. F. Fraser, "Implementation of a fast digital processor using the residue number system," *IEEE Trans. Circuits Syst.*, vol. CAS-28, pp. 32–38, Jan. 1981.
[28] H. K. Nagpal, G. A. Jullien, and W. C. Miller, "Processor architectures for two-dimensional convolvers using a single multiplexed computational element with finite field arithmetic," *IEEE Trans. Comput.*, vol. C-32, pp. 989–1000, Nov. 1983.
[29] R. G. Bennetts, *Design of Testable Logic Circuits*. Reading, MA: Addison-Wesley, 1984, pp. 46–82.



Naresh R. Shanbhag (S'86) received the B.Tech degree in electrical engineering from the Indian Institute of Technology, New Delhi, India, in 1988, and the M.S. degree in electrical engineering from Wright State University, Dayton, OH, in 1990. Currently he is working towards the Ph.D. degree in electrical engineering at the University of Minnesota at Minneapolis.

His research interests include VLSI architectures and algorithms for signal processing, multiple-valued logic, and robust control.

Raymond E. Siferd (S'73–M'76) received the B.S.E.E. degree from Ohio State University, Columbus, the M.S. degree in electrical engineering from the University of New Mexico, Albuquerque, and the Ph.D. degree in electrical engineering from the Air force Institute of Technology, Dayton, OH, in 1977.

He joined the Department of Electrical Engineering faculty at Wright State University, Dayton, OH, in 1981. His current research interests are in integrated circuits and signal processing.