A SINGLE-CHIP PIPELINED 2-D FIR FILTER USING RESIDUE ARITHMETIC

Naresh R. Shanbhag and Raymond E. Siferd
Department of Electrical Engineering
Wright State University, Dayton, OH-45435

I. ABSTRACT

Presented in this paper are novel circuits for residue arithmetic, which have been configured to form a 3x3 finite impulse response filter with programmable coefficients. The filter has a pipelined architecture and includes testability in the form of scan path. Area efficient circuits for residue adders, subtractors and binary-to-residue converters have been designed. An encoding scheme [7] has been employed to reduce the residue multiplier area. A tree architecture for residue-to-binary conversion, has been developed. The filter is timed with a two phase clock, which has an estimated frequency of 15 Mhz.

II. INTRODUCTION

High-speed digital signal processing (DSP) chips have found extensive applications in image and speech processing. The techniques, usually employed at the architectural level, for increasing the processing speed are parallel computation and pipelining. Residue arithmetic, described in detail by Tanaka and Szabo [1], is an option with considerable potential for high-speed DSP applications. The recent interest [2-4], in this mode of computation, stems from the presence of inherent parallelism in it. As far as actual hardware implementations are concerned, the design of 1-D digital filters, using residue arithmetic, has been done [2]. The only instance of 2-D filtering, using residue arithmetic, is a finite impulse response (FIR) filter using off-the-shelf ECL components [3].

As two-dimensional (2-D) filtering is computationally more intensive than its one-dimensional counterpart, it is of interest to develop a single-chip implementation of a 2-D FIR filter using residue arithmetic. Due to its distributed nature of computation, residue arithmetic systems seem to be area expensive. Therefore, a primary objective of this work was to develop area efficient residue arithmetic circuits, without trading off much speed. We present, in this paper, novel circuits for residue addition, subtraction, binary-to-residue and binary-to-residue conversion. A pipelined architecture has been adopted with the aim of keeping a high throughput. A systolic architecture for 2-D filtering is given in [5]. This architecture is not suitable for residue arithmetic implementation because, a systolic array for each modulus would have to be constructed. This requires a large number of latches which would occupy a large area. Testability in the form of scan path has been incorporated.

The paper is divided into five more sections begining 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 while we conclude with section VI.

II. PRELIMINARIES

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

$$X = (X_1, X_2, \ldots, X_n) \quad \text{where}$$
$$X_i = (X) \bmod m_i = |X|_{m_i}, \quad i = 1, \ldots, n$$

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.

The implemented 2-D FIR filter has a mask size of 3x3 with symmetric coefficients (fig.1). Apart from generating linear phase characteristics, symmetric coefficients offer a great deal of reduction in computational complexity. This is due to the fact that data which have to be multiplied with the same coefficients, can be added first. The output $y(i,j)$ is computed as

$$\begin{aligned} y(i,j) = {} & A[x(i-1,j) + x(i+1,j)] + \\ & B[x(i,j-1) + x(i,j+1)] + \\ & C[x(i-1,j-1) + x(i+1,j+1)] + \\ & D[x(i+1,j-1) + x(i-1,j+1)] + x(i,j) \end{aligned} \tag{2.3}$$

It may be noted that, if the data and the coefficients are represented in a residue number system with modulii set $m_1, m_2, \ldots, m_n$, then the residue digits of $y(i,j)$ are evaluated by substituting the residue digits of the data and coefficients into (2.3), 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 modulii set was (13,11,9,7,5,4) with a dynamic range of 180180 (18-bits).

III. CIRCUITS

In this section we present all the basic circuits developed for residue arithmetic. In the initial phase of the design, it was decided to implement residue adders and multipliers using PLA's. This approach was found to be expensive in terms of area and hence novel circuits were developed for the residue adder, subtractor, residue-to-binary and binary-to-residue converter, while an input encoding scheme [7] was employed to reduce the multiplier area.

A. Modulo m_i Adder

The Modulo m_i adder circuit (fig.2) consists of a conventional ripple-carry binary adder (RCBA), followed by a circuit which subtracts m_i (SUBm_i) from the output of RCBA ($RCBA$). A 2x1-multiplexor (MUX2x1) then selects between $RCBA$ and the output of SUBm_i ($SUBm_i$) to generate the final output SM_{m_i}. As both the inputs, X and Y, are residue digits, therefore their sum $RCBA$ wouldl lie between 0 and $2m_i - 2$. If $RCBA < m_i$ then MUX2x1 selects $RCBA$ as the final output otherwise it selects $SUBm_i$.

The circuit for SUBm_i was developed by modifying a ripple-carry binary adder with one of inputs premanently set equal to the 2'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 to find out whether $RCBA$ is greater than or equal to m_i. In fact it can be shown that $RCBA \geq m_i$, if either the final carry from $RCBA$ or the final borrow from $SUBm_i$ equals 1. As compared to a straightforward PLA realization, this circuit offers a 51% savings (for $m_i = 13$) in area. Delay comparisons, for the same load capacitance of 0.2pf, show that this circuit is just 2ns slower than the PLA realization.

B. Modulo m_i Subtractor

This circuit was needed in the residue-to-binary converter. In a fashion similar to the design of the Mod 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_mi). Though a PLA realization of the Mod m_i subtractor was not explicitly done, our experience with the Mod m_i adder would indicate that area savings of similar magnitude had resulted due to this circuit. For a $m_i = 13$, the subtractor occupied an area of $72420\mu m^2$ and had a delay of 20ns for a load of 0.2pf.

C. Modulo m_i Multiplier

An input encoding scheme [7] 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 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.

In the absence of a non-heuristic algorithm for determining the optimal input pairings, exhaustive search was done. The software package called ESPRESSO-MV [8] was used to minimize the multiplier truth table with input pairings. Later, it was observed that due to the modulo operation, the optimal input pairings for all n-bit moduli are the same except for $m_i = 2^n - 1$. Not only that but also for a particular modulii the both the modulo adder and the modulo multiplier have the same optimal input pairings.

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 0.2pf, the delays were nearly equal (47ns).

D. Binary to Residue Converter

A PLA realization of a binary to residue converter (BTOR) for $m_i = 13$ showed that an area of $619\mu m \times 774\mu m$ would be required. This was not acceptable due to the area limitations of the chip frame. Hence a new circuit was developed based on the following line of reasoning. Let an 8-bit number X be represented as follows

$$X = 2^4 B_M + B_L$$

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

$$\begin{aligned}(X)\text{mod}(m_i) &= (2^4 B_M + B_L)\text{mod}(m_i)\\ &= [(2^4 B_M)\text{mod}(m_i)] \oplus_{m_i} [(B_L)\text{mod}(m_i)]\\ &= [(2^4)\text{mod}(m_i) \otimes_{m_i} (B_M)\text{mod}(m_i)] \oplus_{m_i} [(B_L)\text{mod}(m_i)]\end{aligned}$$

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

As B_M and B_N are both 4-bit numbers, hence two PLA's with at most 16 product terms each and a Mod m_i adder would be required. The circuit block diagram (fig.4) shows that PLA1 computes the first term $((2^4) \mod m_i \otimes_{m_i} (B_M) \mod m_i)$ while PLA2 computes the second $((B_L) \mod m_i)$. Area and speed comparisons for $m_i = 13$ were made. The area reduction that was achieved with this technique was 71%. Speed comparisons show that the new circuit is about 8ns faster than the previous one, again for a load of 0.2pf.

E. Residue to Binary Converter

For residue to binary conversions we considered two methods before developing another. They were the popular mixed radix conversion method [1] and a parallel residue to binary converter [6]. The mixed radix conversion method is essentially a serial algorithm and hence is slow. Any attempt to pipeline it would be prohibitive in terms of area. On the other hand the parallel conversion algorithm [6], though fast, requires a large number of latches which was unacceptable in our situation.

Our method relies on combining two moduli at a time (fig.5(a)), to generate a tree structure. In this particular implementation, modulii pairs (13,4),(11,5) and (9,7) were combined, in the first stage, to generate a new modulii set (52,55,63). In the second stage modulii 52 and 55 were combined to generate representation in modulo 2860, which was then combined with modulus 63, in the last stage, to get the final answer. It can be seen that it is very convenient to pipeline this conversion method. Each layer of the tree can be made a stage in the pipeline.

The combination of two modulii was done using the conventional mixed radix algorithm [1](fig.5(b)). Hence it is important that the modulii 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.

IV. THE FILTER ARCHITECTURE

The FIR filter has a pipelined architecture (fig.6) with four stages of pipelining. The latches (L), demarcating the different stages of the pipeline, could be configured in a shift register configuration to form a scan path.

The first stage consisted of BTOR's with three such convertors being needed for each modulus. This is because we are moving the coefficient mask in a horizontal direction and hence at every clock cycle three new data (one from each row covered by the mask) are introduced into the filter. As 15 binary-to-residue converters were needed ($m_i = 4$ not requiring any) therefore the area reductions (mentioned in Section III(C)) played a crucial role.

In the second stage, the actual computations, required for filtering, were carried out in the different modulii (M13, M11, M9,M7, M5, M4). Again the new Mod m_i adder and the Mod m_i multiplier circuits played an important role in saving area. The third stage consisted of modulii convertors MC52, MC55 and MC63, which converted the modulii pairs (13,4), (11,5) and (9,7) into another residue number system with modulii set (52,55,63). In the third stage itself we combined modulii pair (55,52), using MC2860, into a modulo 2860 representation. In the last stage modulii pair (2860,63) were combined to generate the 18-bit final answer.

V. TESTABILITY AND TIMING FEATURES

Testability in the form of scan path has been incorporated. As the latches, before and after every stage of the pipeline, could form a shift register, in the testing mode, it possible to load a test vector into each stage of the pipeline. Then after one cycle we can clock out the results of every stage independent of each other. An attractive feature, which results due to the combination of residue computation and scan path testing, is the flexibility to exchange the precision of filter data with that of the coefficient. This is due to the fact that coefficient data can be loaded, through the scan path registers, in residue form. Hence, as long as the dynamic range of the output is not exceeded, we can increase the coefficient data precision right upto 18-bits.

Dynamic registers, with two-phase clocking, were used as they were area efficient. Clock drivers, generating a two-phase clock from a global clock line, were placed near every cluster of latches. The filter operation would start with loading the coefficient registers. This can be done either through the regular 8-bit data input ports or using the scan path facility mentioned earlier. In the former case, the coefficients would be input in decimal form, which would then be converted into residue by the binary-to-residue converters. The coefficient data precision would be limited to 8-bits, in this case. On the other hand, using the scan path facility we can load the coefficient data directly in residue form and thus increase the precision. Once the coefficients are loaded we start feeding three 8-bit inputs every clock cycle and get an 18-bit output. Simulations indicate that the filter should operate at a frequency of 15 Mhz

VI. CONCLUSIONS

Residue arithmetic, which has an inherent parallelism, has been employed to do 2-D filtering. Novel circuits for residue addition, subtraction, multiplication, residue-to-binary conversion and binary-to-residue conversion have resulted in a highly compact filter structure. We envisage an improvement in the speed of the residue-to-binary convertor, by employing adders not of the ripple-carry form. Testability in the form of scan path has been incorporated. This unique combination of residue arithmetic and scan path has resulted in the flexibility to exchange filter data and coefficient data precision. Pipelining has been employed to increase the throughput.

References.

[1] N.S. Szabo and R.I. Tanaka, *Residue Arithmetic and its Application to Computer Technology*, New York : McGraw-Hill, 1967.

[2] 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. on Circuits and Systems*, CAS-33, pp. 5-25, Jan.1986.

[3] 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. on Circuits and Systems*, vol. CAS-28, pp. 32-38, Jan.1981.

[4]W.K. Jenkins and B.J. Leon,"The use of residue number systems in the design of finite impulse response digital filters", *IEEE Trans. on Circuits and Systems*, April 1977.

[5] M.A. Sid-Ahmed,"A systolic realization for 2-D digital filters", *IEEE Trans. Acoust., Speech, Signal Processing*, vol.37, pp. 560-565, Apr. 1989.

[6] 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.

[7] T. Sasao, "Input variable assignment and output phase optimization of PLA's", *IEEE Trans. Comput.*, vol. C-33, pp. 879-894, Oct. 1984.

[8] R. Rudell and A. Sangiovanni-Vincentelli, "Multiple-valued minimization for PLA optimization", *IEEE Trans. Computer-Aided Des.*, vol. CAD-6, pp. 727-751, Sept. 1987.

C	A	D
B	1	B
D	A	C

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)

Fig.1. Filter coefficient mask and data.

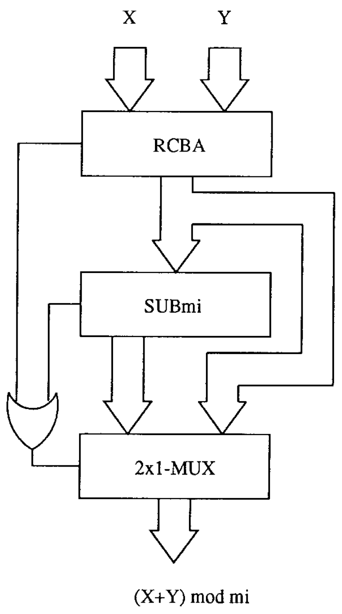

Fig.2. Modulo mi adder

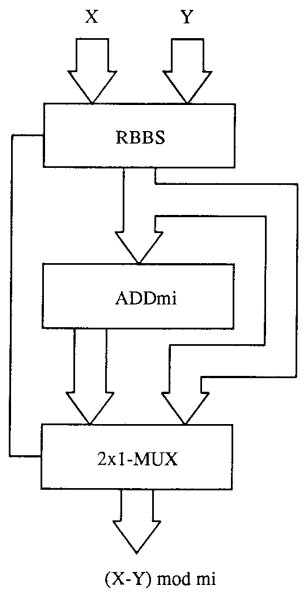

Fig.3. Modulo mi subtractor.

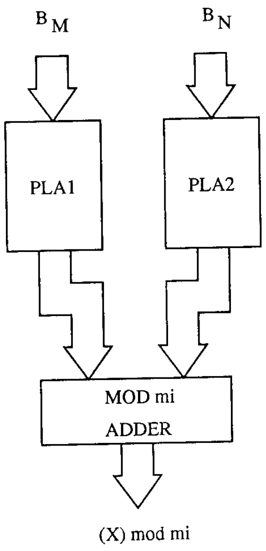

Fig.4. Binary to residue converter

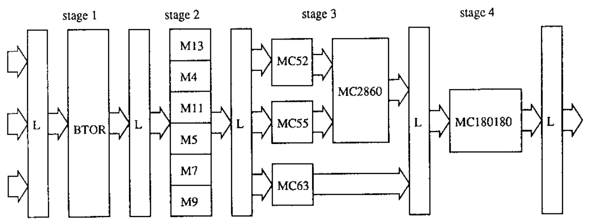

Fig.6. Filter architecture.

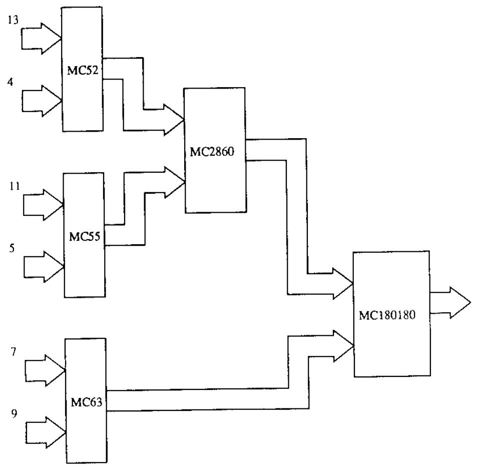

Fig.5. Residue to binary converter.