A High Performance RNS Multiply-Accumulate Unit

A.P. Preethy
Department of Computer Science
Georgia State University
30 Pryor St., SE 750 Atlanta
GA 30303
ppreet@cs.gsu.edu

Damu Radhakrishnan
Department of Electrical and Computer Engineering
State University of New York
75 S. Manheim Blvd., New Paltz
New York 12561
(845) 257-3772
damu@engr.newpaltz.edu

Amos Omondi
School of Computer Engineering,
Nanyang Technological University
Nanyang Ave., Singapore 639798
asamos@ntu.edu.sg

ABSTRACT
This paper discusses the VLSI implementation of a new architecture for a multiply-accumulate unit based on Residue Number System (RNS). The architecture and VLSI implementation of an arbitrary-moduli RNS MAC are given. The cost and performance are analyzed with respect to other designs, and the analysis indicates that the design is generally quite competitive.

1. INTRODUCTION
The demand for high-speed computing continues to increase for digital signal processing and multimedia applications, and appropriate research activity grows at a similar rate. In this regard multiplier designs and multiplier-accumulator units (MAC) have long been a topic of interest to the digital community due to their extensive use in almost every design for such applications. RNS features highly parallel carry-free addition, multiplication, and borrow-free subtraction. Hence RNS-based systems are preferable to conventional ones in view of the parallelism in such arithmetic operations. Many innovative techniques have been used in the past for the design of multipliers by making use of the properties of RNS and also the number-theoretical properties of finite fields. This gave birth to a class of processors known as RNS processors.

A residue number system is characterized by a base that is not a single radix but an n-tuple of integers $(m_1, m_2, \ldots, m_n)$ [1]. Each of these m_i, $i = 1, \ldots, n$ is called a modulus and is selected such that they are pair-wise relatively prime, *i.e.*, $(m_i, m_j) = 1$, for $i \neq j$. Any integer X is represented in the RNS by an n-tuple of integers $(x_1, x_2, \ldots, x_n)$, where each x_i is a non-negative integer satisfying the relationship $X = m_i \cdot q_i + x_i$. Here, the quotient q_i is the largest integer such that $0 \leq x_i \leq (m_i - 1)$. The remainder x_i is hence the residue of X modulo m_i, and the notations $X \bmod m_i$ and $|X|_{m_i}$ are commonly used.

It follows from the Chinese Remainder Theorem (CRT) that for any given n-tuple satisfying the above relationships, there exists one and only one integer X such that $0 \leq X < M$ where $M = \prod_{i=1}^{n} m_i$. The number X can be reconstructed from the n-tuple $(x_1, x_2, \ldots, x_n)$ using the CRT equation $X = (\sum_{i=1}^{n} \frac{M}{m_i} a_i x_i) \bmod M$, where a_i is defined by the relationship $\frac{M}{m_i} a_i \equiv 1 \bmod m_i$.

An alternate method, called mixed-radix-conversion (MRC), employs a mixed-radix-representation $(a_1, a_2, \ldots, a_n)$ for a number X, where $X = \sum_{k=1}^{n} a_k \prod_{i=1}^{k-1} m_i$, with the digits satisfying $0 \leq a_i < m_i$, and $i = 1, 2, \ldots n$. Here, $\prod_{i=1}^{0} m_i = 1$.

Arithmetic operations in RNS are defined by:
$(x_1, x_2, \ldots, x_n) \circledast (y_1, y_2, \ldots, y_n) = (z_1, z_2, \ldots, z_n)$ ……(1)
with $z_i = (x_i \circledast y_i) \bmod m_i$, where, $\circledast$ denotes any of the modulo operations of addition, subtraction or multiplication. The above definition has many implications. First, all arithmetic operations are performed on smaller residues instead of the large number. Also, these operations can be done simultaneously in all the modulo channels thereby speeding up the whole operation. Second, it gives fault isolation between different modulo channels. The net result is very high-speed parallel arithmetic processing.

2. THE MULTIPLY-ACCUMULATE UNIT
The basic operation involved in the Multiply-Accumulate Unit (MAC) is the summation of a number of products given by the general expression $Z = \sum_{i=1}^{N} Z_i = \sum_{i=1}^{N} X_i Y_i$. ……(2)
where X_i and Y_i are input samples and Z_i is the output sample. The output is calculated over N samples. If X_i and Y_i are represented in residue form using a moduli set $\{m_1, m_2, \ldots, m_n\}$ then, as $X_i = \{x_{i1}, x_{i2}, \ldots, x_{in}\}$ and $Y_i = \{y_{i1}, y_{i2}, \ldots, y_{in}\}$ in residue notation, the above summation thus becomes r independent summations:
$\sum_{i=1}^{N} x_{i1} y_{i1}, \sum_{i=1}^{N} x_{i2} y_{i2}, \ldots, \sum_{i=1}^{N} x_{ij} y_{ij}, \ldots, \sum_{i=1}^{N} x_{in} y_{in}$.
These summations correspond to the sum tuple $(z_1, z_2 \ldots, z_n)$.

Our new MAC architecture, which is described in detail in [2] uses index-transform based approach. The relatively prime moduli in any arbitrary moduli set take any of the three forms p, 2^m, and p^m, or a factorable modulus with any of these factors, where p is prime and m is any integer. It follows from Number Theory that the groups formed by p, 2^m, and p^m integer elements fall into the category of Galois Field $GF(p)$, and integer rings z_{2^m} and z_{p^m}.

For prime modulus p, the normal index mapping in $GF(p)$ is used. The multiplication is done by index addition and each nonzero integer is coded into an index $<\alpha>$ [1]. In the case of finite integers, the procedure for finding an index set for the elements of z_{2^m} is given in [3]. Using this, any integer $X \in \{1,2,\ldots,2^m-1\}$ can be coded using a triplet index code $<\alpha,\beta,\gamma>$ with the relationship $X = 2^{\alpha}\left|5^{\beta}(-1)^{\gamma}\right|_{2^m}$, where $\alpha \in \{0,1,\ldots,m-1\}$, $\beta \in \{0,1,\ldots,(2^{m-2}-1)\}$ and $\gamma \in \{0,1\}$. Multiplication of two integers can now be carried out as follows: let $X_1, X_2 \in z_{2^m}$, $X_1 \neq 0$, $X_2 \neq 0$, $X_1 = 2^{\alpha_1}\left|5^{\beta_1}(-1)^{\gamma_1}\right|_{2^m}$, and $X_2 = 2^{\alpha_2}\left|5^{\beta_2}(-1)^{\gamma_2}\right|_{2^m}$, then $\left|X_1X_2\right|_{2^m} = 2^{\alpha_1+\alpha_2}\left|5^{\beta_1+\beta_2}(-1)^{\gamma_1+\gamma_2}\right|_{2^m}$.

The indices are added subject to the following constraints: β_1 and β_2 are added mod 2^{m-2}, γ_1 and γ_2 are added mod 2, and α_1 and α_2 are added in normal binary mode. When the sum of αs equals m-1 the corresponding β and γ are made zero, and when it exceeds m-1, the final result is made zero. Thus, by storing the index and inverse index tables, multiplication of the nonzero elements can be replaced by index addition.

Similarly, in the case of z_{p^m}, where p is odd, an index pair coding (α,β) is used, where X is given by $X = (g^{\alpha}p^{\beta}) \bmod p^m$ [2]. The product of two numbers (X_1X_2) mod p^m can now be calculated as follows:

let $X_1, X_2 \in z_{p^m}$, $X_1 \neq 0$, $X_2 \neq 0$, $X_1 = (g^{\alpha_1}p^{\beta_1}) \bmod p^m$, and

$X_2 = (g^{\alpha_2}p^{\beta_2}) \bmod p^m$, then $(X_1X_2) \bmod p^m$ is given by:

$$\left|X_1X_2\right|_{p^m} = (g^{\alpha_1}p^{\beta_1}g^{\alpha_2}p^{\beta_2}) \bmod p^m$$

$$= (g^{\alpha_1+\alpha_2}p^{\beta_1+\beta_2}) \bmod p^m .$$

The indices are added subject to the following constraints: α_1 and α_2 are added mod $\phi(p^m)$, and β_1 and β_2 are added in normal binary mode. When the sum of βs exceeds m-1, the final result is made zero. From the above, it may be noted that index transform techniques are suitable for any arbitrary moduli set subject to the only proviso of relative primality. Thus all the MAC channels are designed using the index calculus techniques. The corresponding architecture consists of n separate channels each working independently and all in parallel.

2.1 Architecture

Figure 1 shows the MAC architecture suitable for any arbitrary moduli set. Block schematics of representative channels are shown in the Figures 2 (a), (b), and (c). Each prime field MAC channel has the design of Figure 2 (a). The i^{th} residues x_{ij}, and y_{ij}, of channel j at the time interval i address two index ROMs that are used to find the logarithms α_1 and α_2 corresponding to these operands. These indices are added using a modulo-m_i adder to get the index, α_i, of the product z_i. Subsequently, α_i addresses an inverse-index ROM to get back the product z_i, which then gets transferred into Register1. The modulo adder in the next stage adds the present sum to the previous sum fed back from Register2, which is initialized to zero, thus accumulating the summation of the products $x_i\,y_i$, over the interval $i=1,N$. The final sum is left in Register2.

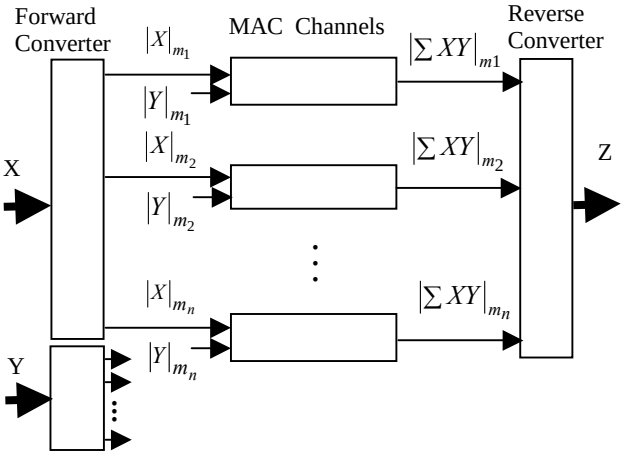

Figure 1. MAC architecture for arbitrary moduli sets

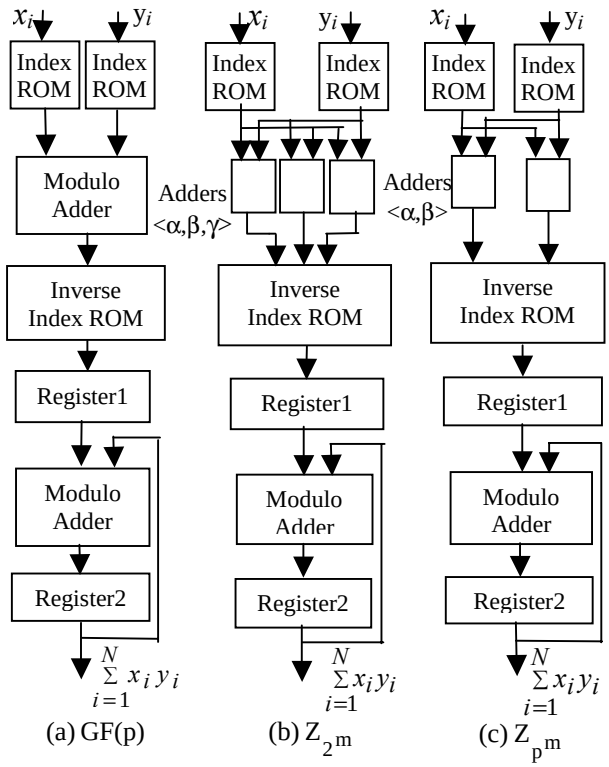

Figure 2. MAC channels

The integer-ring MAC channels are also designed on similar lines. The difference as apparent from Figures 2(b) and 2(c), is in the index adder section: since the integer mapping in z_{2^m} is done using an index triplet, three adders are needed in this case. For

adding the βs, a modulo 2^{m-2} adder is required. In the case of γ, only an XOR gate is needed. To add the αs, a conventional m-bit binary adder is sufficient. Similarly, in the case of z_{p^m}, for odd p, to carry out the addition of the αs and βs respectively, a $\phi(p^m)$ modulo adder and a conventional m-bit binary adder are required. Since modulo adders will take up a major proportion of the MAC hardware, the performance depends on the modulo adders employed in the architecture. Based on an evaluation of contemporary modulo adders, an area-time efficient modulo adder architecture is adopted for the implementation.

2.2 VLSI Implementation

For the design of a 36-bit multiplier, a perfectly balanced optimum moduli set {17,19,23,25,27,29,31,32} comprising of only 5-bit moduli was chosen. Since index is not defined when the residue is zero, extra logic is incorporated into the design to facilitate multiplication when the operands are zeros.

The 36-bit MAC architecture using the above moduli set was synthesized as a VHDL model using the Synopsys Synthesis tool. The process technology used is 0.35 micron CMOS from the Synopsys library (cb35os142d). In the implemented MAC, one full stage consists of 8 cells, one for each modulus, essentially consisting of memory and adder stages. Each cell takes up an average area of 0.02407 mm^2. The only significant difference between cells is in the number of index adders and in the programming of the ROMs. This results in a highly regular and VLSI-efficient structure. The delay of each MAC channel was found to vary according to the class of moduli. The lowest delay was found to be 7.44 ns (modulo17) and the highest was found to be 10.72 ns (modulo 25). Hence the delay of the MAC unit is taken as 10.72 ns, which is the delay through the critical delay channel.

2.3 Performance Analysis

In order to be able to compare our implementation with those of existing MACs (conventional or otherwise) and to be able to extrapolate into future VLSI processes, we scaled the figures obtained from the synthesis. The scaling for gate delay and gate area are done on a standard basis. Scaling for precision is generally more difficult, and we simply chose a linear model.

To compute the total area-time figures, the delays due to binary-to-residue and residue-to-binary converters also have to be taken into account. In our 18x18+36-bits precision MAC architecture, two 16-bit forward converters and one 32-bit reverse converter are required. It has been shown in [4] that a forward converter for the above moduli set has a latency of 4.5 ns when precision and process technology are scaled down to 18 bits and 0.35 micron respectively. A reverse converter for the same moduli set has a latency of 5.625 ns when precision and process technology are scaled down to 36 bits and 0.35 micron respectively [5]. So the worst case delay in the conversion modules of a 36-bits unit amounts to 10.125 ns. In our design, we have considered a 32nd order FIR filter realization as a possible application of our MAC unit; so only one conversion delay is needed for one summation as given in Equation 2, over the interval 1 to 32 (32 taps). Hence the conversion latency will amount to only 0.316 ns per tap. When this conversion delay is added to the MAC unit's critical delay 10.72 ns, the total delay of the 36-bit MAC becomes 11.03 ns.

The total area consumed by all the eight MAC channels amounts to 0.19256 mm^2. The forward converter and the reverse converter take up areas of 0.6075 mm^2 and 2.531 mm^2 respectively when scaled down to 0.35 micron process technology with precisions of 18 bits and 36 bits. Hence, the total area requirement for the 36-bit MAC unit comes to 3.33 mm^2. For the sake of a fair comparison with the existing 16x16+32-bits precision MAC designs, the performance metrics of the proposed MAC is scaled down to the same precision and thus the figures for delay and area become 9.81 ns and 2.96 mm^2 respectively. For comparisons with other designs, we also scaled their implementations to assume 0.35 micron CMOS technology and 16x16+32-bits precision.

Next, we compared the proposed MAC unit with two other RNS-based designs and two conventional designs. The two RNS processors used are the PRVP and the INMOS IMS A110 [6,7]. The PRVP is a 16x16+32-bit precision multiply-accumulator realized in 2-micron CMOS process and can perform one multiply-accumulate operation in 40 ns. The IMS A110 operates at 20 MHz, with a MAC that operates on 8-bit data/coefficients. The area used by the MAC unit is not reported in the literature.

The first of the conventional MAC units chosen for comparison is a 100 MHz DSP core proposed by a research group at NEC, Japan [8]. The DSP core includes a 0.25-micron multiplier-accumulator with a latency of 20 ns and a precision of 16x16+32-bits. The area occupied by the MAC unit is 0.77 x 0.72 mm^2 (0.55 mm^2). The other conventional MAC unit considered for comparison is also a 16x16+32-bit MAC and is from a most recent publication [9]. The unit operates at 66 MHz using transmission gate (TG) logic, and was implemented in 0.65-micron CMOS technology. The area occupied by the MAC unit is 2.25 mm^2.

The scaled area-time comparisons with the RNS-based MACs and the conventional MACs are given in Tables 1 and 2 respectively. Table 3 gives the normalized area, delay and cost:performance ratio. The INMOS processor has a latency of 14.58 ns whereas the proposed design has a latency of only 9.81 ns - a speed up of 150%. The PRVP processor on the other hand needs only 7 ns to complete one MAC operation, hence the proposed design is slower by 30%. When compared with the NEC group MAC unit, our proposed design's speed-up is a remarkable figure of 285%; but on the other hand, it is slower than the transmission-gate-based design by 11%. The chip area of the proposed design is also 4 to 10 times higher compared to three of the designs; the area of the INMOS processor is not reported in the literature. Because of the larger area, the normalized cost:performance ratio of the proposed MAC is about 1 to 14 times compared to the three cases.

Table 1. Area-time comparison with RNS-based MACs

MAC	Precision, Process	Scaled delay	Scaled area
Proposed	18x18, .35µ	9.81 ns	2.96 mm^2
INMOS	8x8, 1.2µ	14.58 ns	_
PRVP	16x16, 2µ	7 ns	0.298 mm^2

Table 2. Area-time comparison with conventional MACs

MAC	Precision, Process	Scaled delay	Scaled area
Proposed	18x18, .35μ	9.81 ns	2.96 mm^2
NEC group	16x16, .25μ	28 ns	1.078 mm^2
TG based	16x16, .6μ	8.75 ns	0.765 mm^2

The difference in delay and area can be partly attributed to the way in which the MAC was implemented. Only full-custom VLSI designs give the designer more flexibility in the hardware, which makes the realization of RNS processors more efficient. When implementation is done using synthesis tools, the delay and area will be determined by the built-in macros, and hence the figures given out will be higher compared with a full-custom design. For example, the area-delay figures given for a Synopsys full adder are 36 transistors and 0.6 ns respectively in 0.35 micron CMOS process, where as one of the most recent full adder designs incurs only a 0.42 ns delay in the same technology and needs only 14 transistors. Given that the other processors have been implemented using full-custom design, it is evident from the small delay differences that our design would be faster if implemented in the same finesse. Although transmission-gates have in the past been widely used in high-speed arithmetic, in deep-submicron realizations, transmission gates have prohibitively large capacitances. Consequently, it is extremely unlikely that they will be used at 0.18 micron and below. This immediately makes the transmission-gate design, one of the two faster ones, non-competitive.

Table 3. Normalized cost-performance figures

MAC	Normalized area	Normalized delay	Normalized cost:perf. ratio
Proposed	1	1	1
INMOS	_	1.48	_
PRVP	0.1	0.71	0.071
NEC group	0.36	2.85	1.026
TG based	0.26	0.89	0.2314

In an RNS-based system, the converters always form the bottleneck due to their lack of speed, and indeed, most of the cost of any RNS architecture will be largely concentrated in the converters. Another reason for the large area of our implementation is that it was optimized for high speed while compromising area.

3. CONCLUSIONS

An RNS-based MAC unit entirely based on index transform techniques is presented in this paper. It has been shown that index transform techniques can be applied to arbitrary moduli sets subject to the proviso of relative primality. Each modulus will fall into any one of the classes p, p^m, and 2^m , and each category supports index transform-based techniques. This makes the MAC architecture implementation multiplier-free, only a few binary and modulo adders and a few ROMs are needed. The architecture is simple, highly modular and parallel. A 36-bit precision MAC unit using a perfectly balanced 5-bit moduli set {17,19,23,25,27,29,31,32} was implemented as a VHDL model in 0.35 micron CMOS technology using Synopsys tools. The MAC unit incurs 11.03 ns delay and 3.33 mm^2 of area. The cost-performance was compared with two RNS-based designs and two other conventional designs. With two of the compared designs, the proposed MAC shows a speed up of 150% to 285% and with the other two it is slower by 11% to 30%. Nevertheless, the chip area of the proposed design is 4 to 10 times higher compared to three of the designs. The normalized cost:performance ratio of the proposed MAC unit is about 1 to 14 times compared to the three cases. Thus the proposed MAC design is generally quite competitive.

4. REFERENCES

[1] Soderstrand, M.A., Jenkins, W.K., Jullien, G.A., and Taylor, F.J., *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing.* New York: IEEE Press, 1986.

[2] Preethy, A.P., and Radhakrishnan, D., "A 36-bit Balanced Moduli MAC Architecture," *Proc. IEEE Midwest Symp. on Circuits and Systems*, New Mexico, pp. 380-383, Aug. 1999.

[3] Vinogradov, I.M., *Elements of Number Theory.* New York: Dover, 1954.

[4] Drolshagen, A., Henkelmann, H., and Anheier, W., "Processor Elements for the Standard Cell Implementation of Residue Number Systems," *Proc. IEEE Int'l Conf. on Application-specific Systems, Architectures and Processors*, pp. 116-123, 1997.

[5] Henkelmann, H., Drolshagen, A., Bagherinia, H., Ahrens, H., and Anheier, W., "Automated Implementation of RNS-to-Binary Converters," *Proc. IEEE Int'l Symp. on Circuits and Systems*, vol. 2, pp. 137-140, 1998.

[6] Hohne, R.A., and Siferd, R., "A Programmable High Performance Processor using the Residue Number System and CMOS VLSI Technology," *Proc .IEEE National Aerospace and Electronics Conf.*, vol. 1, pp. 41-43, 1989.

[7] Barraclough, S.R., Sotheran, M., Burgin, K., Wise, A.P., Vadher, A., Robbins, W.P., and Forsyth, R.M., "The Design and Implementation of the IMS A110 Image and Signal Processor," *Proc. IEEE Custom Integrated Circuits Conf.*, 1989.

[8] Izumikawa, M., Igura, H., Furuta, K., Ito, H., Wakabayashi, H., Nakajima, K., Mogami, T., Horiuchi, T., and Yamashina, M., "A 0.25-micron CMOS 0.9-v 100-MHz DSP Core," *IEEE Journal of Solid-state circuits*, vol. 32, no. 1, pp. 52-61, Jan. 1997.

[9] Lee, S., Chung, J., Yoon, H., and Lee, M.M., "High Speed and Ultra-Low Power 16x16 MAC Design using TG Techniques for Web-based Multimedia System," *Proc. IEEE Design Automation Conf.,* pp. 17-18, 2000.