DESIGN AND IMPLEMENTATION OF BOOTH MULTIPLIER IN COMPARISON WITH OTHER MULTIPLIERS

Shanmuga Priya.K[1], Ruth Anita Shirley.D[2],Prabu Venkateswaran.S[3]

[1]PG students / ECE, SNS College of Technology, Coimbatore, (India)

[2]PG student / VLSI Design, SNS College of Technology, Coimbatore, (India)

[3]Assistant Professor, Dept. of ECE, SNS College of Technology, Coimbatore,(India)

ABSTRACT

The multi-modulus design capable of performing the desired modulo operation for more than one modulus in Residue Number System. The Residue Number System and has been used for efficient carry free operation. It explores the efficient use of hardware resources by the use of Booth algorithm. This algorithm helps to reduce the partial products to half. Radix-4 and Radix-8 booth encoding algorithm has been employed. The Booth multiplier of both signed and unsigned numbers. The use of booth encoding optimizes area overhead and also increases the performance of multiplication process. This module has been implemented using Xilinx. The module has been programmed using Verilog.

Keywords: Booth Multiplier, RNS, Radix.

I INTRODUCTION

Residue Number Systems (RNS) allow the distribution of large dynamic range computations over small modular rings, which allows the speed up of computations. This feature is well known, and already used in both DSP and cryptography. Most of implementations use RNS bases of three elements to reduce the complexity of conversions, but if can increase the number of RNS modular computational channels, then we are able to compute over smaller rings and thus further increase the speed of computation.Residue Number System relay on Chinese Remainder Theorem. We consider a n-tuple of co-prime numbers (m_1,m_2, . . . ,m_n) . We note $= \prod_{i=1}^{n} m_i$, If we consider the n-tuple (x_1, x_2, . . . , x_n) of integer such that $x_i < m_i$. Then there exits an unique X which verifies:

$$0 \leq X < M$$

and

$$x_i = X \bmod m_i = (\bmod X) \, base \, m_i \quad pour \; 1 \leq i \leq n$$

The n-tuple (m1,m2, . . . ,mn) of co-primes is generally called RNS basis.

The main interest of the Residue Number Systems is to distribute integer operations on evaluations with the residues values. Thus an operation with large integers is made on the residues which are small numbers and where

computations can be executed independently for each modulo allowing a complete parallelization of the calculus.The multiplication operation is present in many parts of a digital system or digital computer, most notably in signal processing, graphics and scientific computation.

With the recent advances in technology, many researchers have worked on the design of increasingly more efficient multipliers. The common multiplication method is add and shift algorithm. Multiplication can be considered as a series of repeated additions. The number to be added is the multiplicand, the number of times that it is added is the multiplier, and the result is the product. Each step of addition generates a partial product. In most computers, the operand usually contains the same number of bits. When the operands are interpreted as integers, the product is generally twice the length of operands in order to preserve the information content.This repeated addition method that is suggested by the arithmetic definition is slow that it is almost always replaced by an algorithm that makes use of positional representation. It is possible to decompose multipliers into two parts.

The first part is dedicated to the generation of partial products, and the second one collects and adds them. The basic multiplication principle is twofold i.e. evaluation of partial products and accumulation of the shifted partial products. It is performed by the successive additions of the columns of the shifted partial product matrix. The 'multiplier' is successfully shifted and gates the appropriate bit of the 'multiplicand'. They are then added to form the product bit for the particular form. Multiplication is therefore a multi operand operation. To extend the multiplication to both signed and unsigned numbers, a convenient number system would be the representation of numbers in two's complement format. Booth's algorithms are meant for this.

II. COMPARISON OF MULTIPLIERS

An efficient multiplier should have following characteristics:-

Accuracy:- A good multiplier should give correct result.

Speed:- Multiplier should perform operation at high speed.

Area:- A multiplier should occupies less number of slices and LUTs.

Power:- Multiplier should consume less power.

Multiplication process has three main steps

1.Partial product generation.

2. Partial product reduction.

3. Final addition.

For the multiplication of an *n*-bit multiplicand with an *m* bit multiplier, *m* partial products are generated and product formed is $n + m$ bits long.

Here we discuss about three different types of multipliers which are

1. Array Multiplier
2. Wallace Tree Multiplier
3. Booth Multiplier

2.1 Array Multiplier

Array multiplier is well known due to its regular structure. Multiplier circuit is based on repeated addition and shifting procedure. Each partial product is generated by the multiplication of the multiplicand with one multiplier digit. The partial product are shifted according to their bit sequences and then added. The summation can be performed with normal carry propagation adder. N-1 adders are required where N is the no. of multiplier bits.

The Multiplication of two binary number can be obtained with one micro-operation by using a combinational circuit that forms the product bit all at once thus making it a fast way of multiplying two numbers since only delay is the time for the signals to propagate through the gates that forms the multiplication array. In array multiplier, consider two binary numbers A and B, of m and n bits. There are mn summands that are produced in parallel by a set of mn AND gates. n x n multiplier requires n(n-2) full adders, n half-adders and n2 AND gates. Also, in array multiplier worst case delaywould be (2n+1) td.

2.1.1 Disadvantage

- It requires larger number of gates because of which area is also increased.
- Delay for this multiplier is larger.

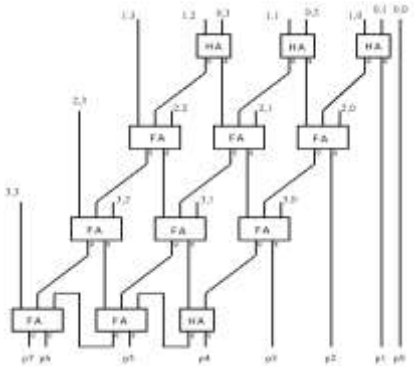

Fig2.1 Array Multiplier

2.2 Wallace Tree Multiplier

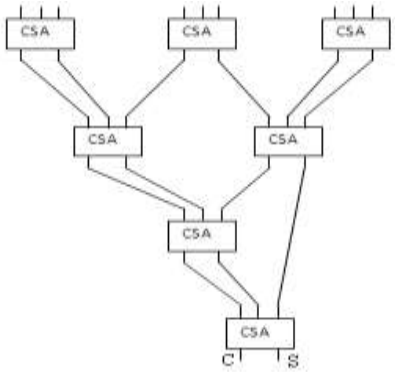

Fig2.2 Wallace Tree Multiplier

A Wallace tree multiplier is an efficient hardware implementation of a digital circuit that multiplies two integers devised by an Australian computer scientist Chris Wallace in 1964. Wallace tree reduces the no. of partial products and use carry select adder for the addition of partial products.

Wallace tree has three steps:-

1. Multiply each bit of multiplier with same bit position of multiplicand. Depending on the position of the multiplier bits generated partial products have different weights.

2. Reduce the number of partial products to two by using layers of full and half adders.

3. After second step we get two rows of sum and carry, add these rows with conventional adders.

Three bit signals are passed to a one bit full adder ("3W") which is called a three input Wallacetree circuit, and the output signal (sum signal) is supplied to the next stage full adder of thesame bit, and the carry output signal thereof is passed to the next stage full adder of the same noof bit, and the carry output signal thereof is supplied to the next stage of the full adder located ata one bit higher position.Wallace tree is a tree of carry-save adders arranged as shown in figure 2.2. A carry save adderconsists of full adders like the more familiar ripple adders, but the carry output from each bit isbrought out to form second result vector rather being than wired to the next most significant bit. The carry vector is 'saved' to be combined with the sum later. In the Wallace tree method,the circuit layout is not easy although the speed of the operation is high since the circuit is quiteirregular.

2.3 Booth Multiplier

Booth multiplication algorithm gives a procedure for multiplying binary integers in signed -2's complement representation. Following steps are used for implementing the booth algorithm:-Let X and Y are two binary numbers andhaving m and n numbers of bits(m and n are equal) respectively.

Step 1-Making booth table:

In booth table we will take four columns one column for multiplier second for previous first LSB of multiplier and other two (U and V) for partial product accumulator (P).

1. From two numbers, choose multiplier (X) and multiplicand (Y).
2. Take 2's complement of multiplicand (Y).
3. Load X value in the table.
4. Load 0 for X-1 value.
5. Load 0 in U and V which will have product of X & Y at the end of the operation.
6. Make n rows for each cycle because we are multiplying m and n bits numbers.

Step2-Booth algorithm:

Booth algorithm requires examination of the multiplier bits, and shifting of the partial product(P). Prior to the shifting, the multiplicand may be added to P, subtracted from the P, or left unchanged according to the following rules:

1. Xi Xi-1 operation

0 0 Shift only

1 1 Shift only

0 1 Add Y to U and shift

1 0 Minus Y from U and shift

2. Take U & V together and shift arithmetic right shiftwhich preserves the sign bit of 2's complement number.So, positive numbers and negative numbers remainspositive and negative respectively.

3. Circularly right shift X because this will prevent usfrom using two registers for the X value.

Repeat the same steps until n no. of cycles are completed.In the end we get the product of X and Y.

The Booth recording multiplier is one such multiplier; it scans the three bits at a time to reducethe number of partial products. These three bits are: the two bit from the present pair; and athird bit from the high order bit of an adjacent lower order pair.

After examining each triplet ofbits, the triplets are converted by Booth logic into a set of five control signals used by the addercells in the array to control the operations performed by the adder cells.To speed up the multiplication Booth encoding performs several steps of multiplicationat once. Booth's algorithm takes advantage of the fact that an adder subtractor is nearly as fastand small as a simple adder.From the basics of Booth Multiplication it can be proved that the addition/subtractionoperation can be skipped if the successive bits in the multiplicand are same. If 3 consecutive bitsare same then addition/subtraction operation can be skipped. Thus in most of the cases the delayassociated with Booth Multiplication are smaller than that with Array Multiplier. However theperformance of Booth Multiplier for delay is input data dependent. In the worst case the delaywith booth multiplier is on per with Array Multiplier.The method of Booth recording reduces the numbers of adders and hence the delayrequired to produce the partial sums by examining three bits at a time. The high performance ofbooth multiplier comes with the drawback of power consumption. The reason is large number ofadder cells required that consumes large power

2.3.1 Booth Multiplication Algorithm for radix 2

Booth algorithm gives a procedure for multiplying binary integers in signed –2's complementrepresentation.

I will illustrate the booth algorithm with the following example:

Example, $(2)_{10}$* $(-4)_{10}$

$(0010)_2$* $(1100)_2$

Step 1-Making the Booth table

I. From the two numbers, pick the number with the smallest difference between a series of consecutive numbers, and make it a multiplier.

i.e., 0010 -- From 0 to 0 no change, 0 to 1 one change, 1 to 0 another change ,so there are two changes on this one

1100 -- From 1 to 1 no change, 1 to 0 one change, 0 to 0 no change, so there is only one change on this one.

Therefore, multiplication of 2 x (– 4), where $(2)_{10}$i.e $(0010)_2$ is the multiplicand and $(-4)_{10}$i.e $(1100)_2$is the multiplier.

II. Let X = 1100 (multiplier)

Let Y = 0010 (multiplicand)

Take the 2's complement of Y and call it –Y–Y = 1110

III. Load the X value in the table.

IV. Load 0 for X-1 value it should be the previous first least significant bit of X

V. Load 0 in U and V rows which will have the product of X and Y at the end of operation.

VI. Make four rows for each cycle; this is because we are multiplying four bits numbers.

Step 2-Booth Algorithm

Booth algorithm requires examination of the multiplier bits, and shifting of the partial product. Prior to the shifting, the multiplicand may be added to partial product, subtractedfrom the partial product, or left unchanged according to the following rules:

- Look at the first least significant bits of the multiplier "X", and the previous leastsignificant bits of the multiplier "X - 1".
- Take U & V together and shift arithmetic right shift which preserves the sign bit of 2's complement number. Thus a positive number remains positive, and a negative number remains negative.
- Shift X circular right shift because this will prevent us from using two registers for the X value.

Table 2.1 Radix4 Modified Booth algorithm scheme for odd values of i .

X(i)	X(i-1)	X(i-2)	Y
0	0	0	+0
0	0	1	+y
0	1	0	+y
0	1	1	+2y
1	0	0	-2y
1	0	1	-y
1	1	0	-y
1	1	1	+0

2.3.2 Booth multiplication algorithm for radix 4

One of the solutions of realizing high speed multipliers is to enhance parallelism which helps to decrease the number of subsequent calculation stages. The original version of the Booth algorithm (Radix-2) had two drawbacks. They are:

(i) The number of add/subtract operations and the number of shift operations becomes variable and becomes inconvenient in designing parallel multipliers.

(ii) The algorithm becomes inefficient when there are isolated 1's. These problems are overcome by using modified Radix4 Booth algorithm which scan strings of three bits with the algorithm given above:

1) Extend the sign bit 1 position if necessary to ensure that n is even.

2) Append a 0 to the right of the LSB of the multiplier.

3) According to the value of each vector, each Partial Product will he 0, +y ,-y, +2y or -2y.

The negative values of y are made by taking the 2's complement and in this paper Carry-look-ahead (CLA) fast adders are used. The multiplication of y is done by shifting y by one bit to the left. Thus, in any case, in designing a n-bit parallel multipliers, only n/2 partial products are generated.

III. SIMULATION RESULT AND ANALYSIS

3.1 Simulated Output of Array Multiplier

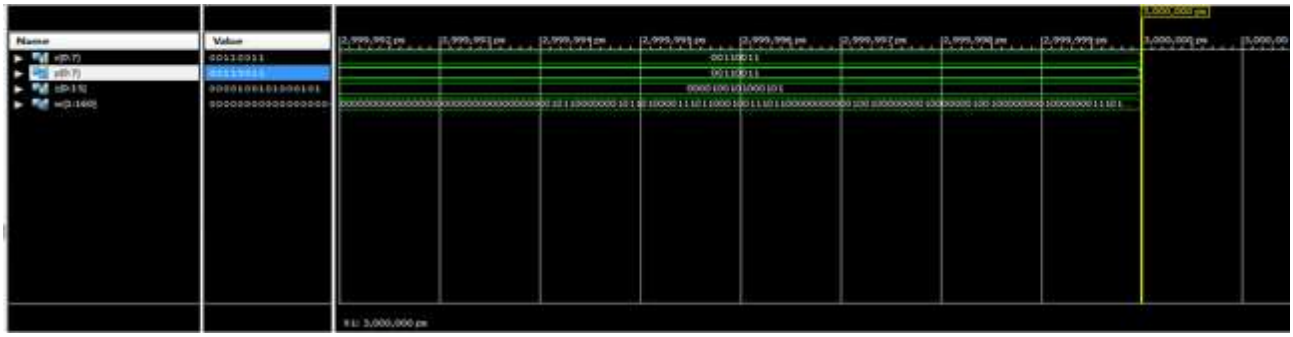

Fig 3.1 Simulated Input Output of Array Multiplier

3.2 Simulated Output of Wallace Tree Multiplier

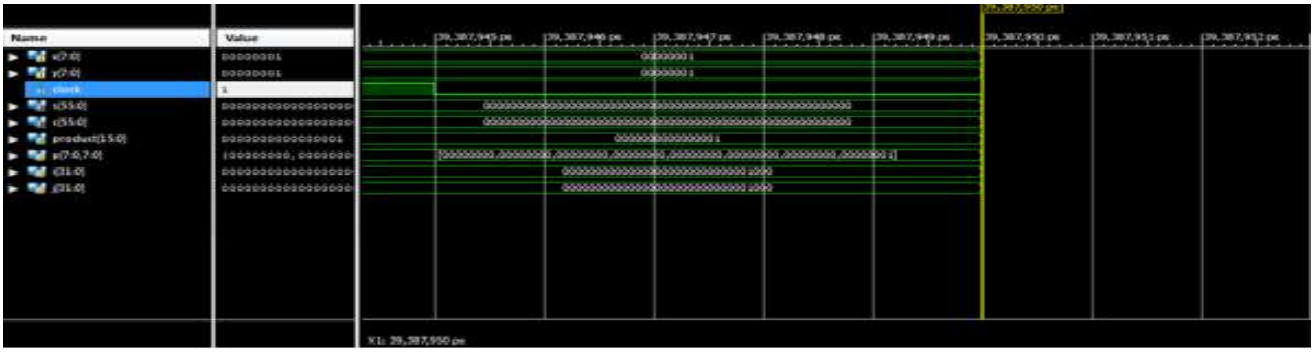

Fig 3.2Simulated Input Output of Wallace Tree Multiplier

3.3 Schematic Diagram of Booth Encoder

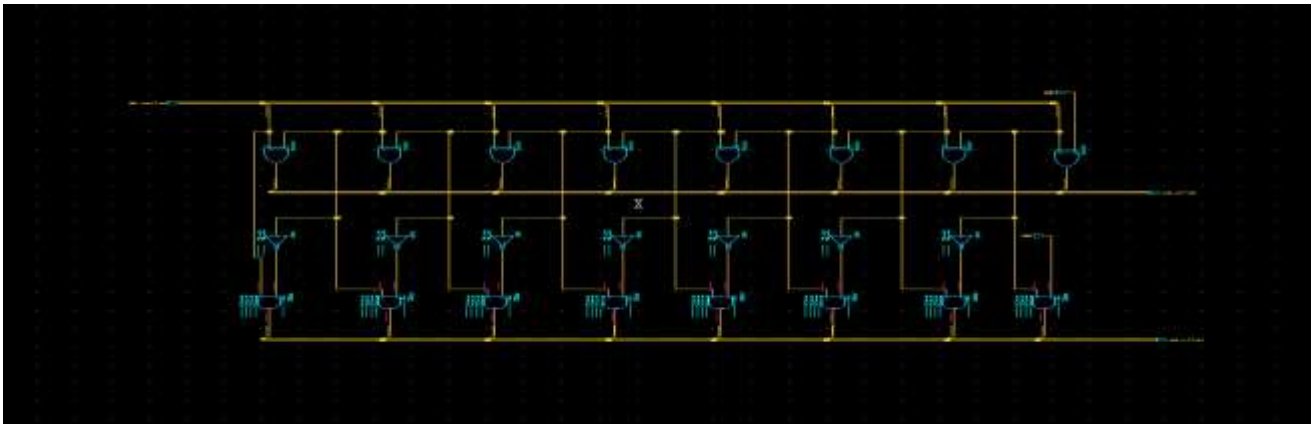

Fig 3.3 Schematic Diagram of Booth Encoder

3.4 Simulated Output of Booth Encoder

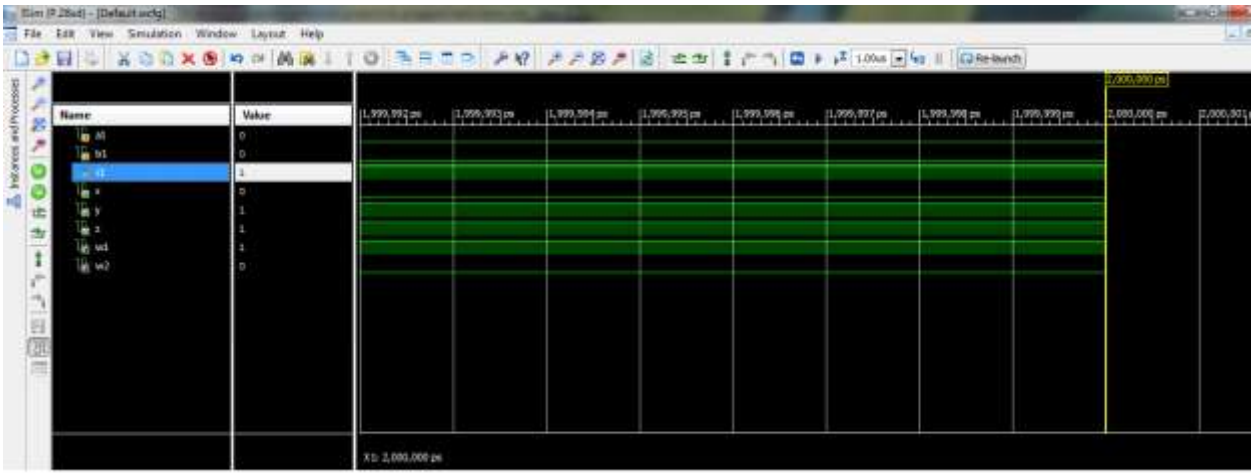

Fig 3.4 Simulated Input Output of Booth Encoder

3.5 Simulated Output of Booth Multiplier

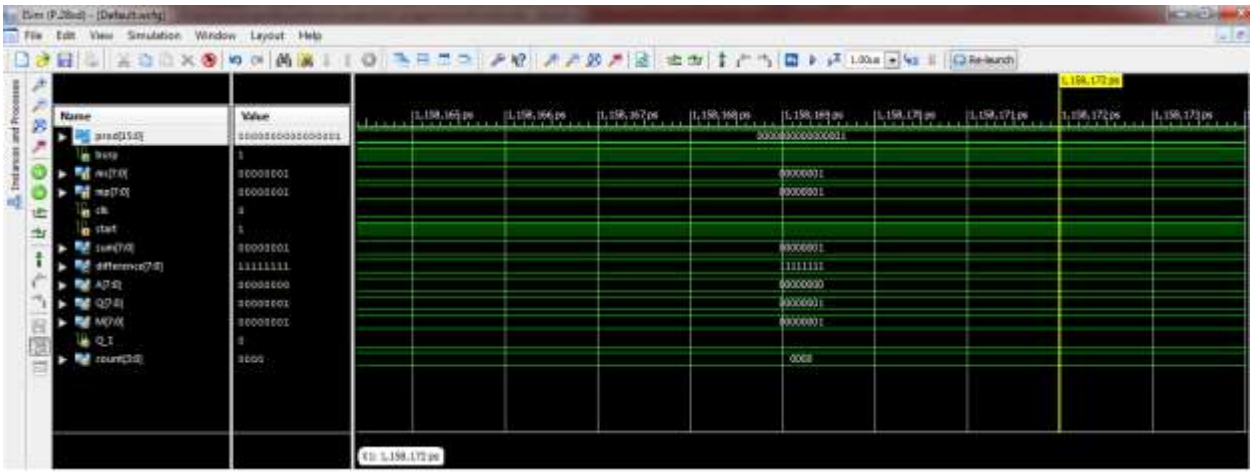

Fig 3.5 Simulated Input Output of Booth Multiplier

IV. CONCLUSION

Different types of 8-bit multiplier have been compared. As a result of comparison in terms of delay, Booth multiplier has been chosen to proceed the multiplication of both signed and unsigned numbers. It also becomes efficient for radix-4 and radix-8 modulo multiplication. It also reduces the number of partial products. Booth encoder saves nearly 40% of area. RNS, an integer system has been chosen since all the positions derive the same weight (base).

V. COMPARISON RESULT BETWEEN MULTIPLIERS

Table 5.1 Comparison of Multipliers

	ARRAY MULTIPLIER	WALLACE TREE	BOOTH MULTIPLIER
DELAY	30.999	25.095	5.584
SLICES	72	65	28
LUT'S	126	122	45

REFERENCES

[1] R. Muralidharan and C. H. Chang, "Area-power efficient modulo 2^n-1 and modulo 2^n+1 multipliers for {2^n-1,2^n,2^n+1} based RNS," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 59, no. 10, pp. 2263–2274, Oct. 2012.

[2] R. Muralidharan and C. H. Chang, "A simple radix-4 Booth encoded modulo 2^n+1 multiplier," in Proc. IEEE Int. Symp. Circuits Syst., Rio de Janeiro, Brazil, May 2011, pp. 1163–1166.

[3] C. Efstathiou, K. Pekmestzi, and N. Axelos, "On the design of modulo 2^n+1 multipliers," in Proc. 14th Euromicro Conf. Digit. Syst. Design, Oulu, Finland, Aug. 2011, pp. 453–459.

[4] J. W. Chen and R. H. Yao, "Efficient modulo 2^n+1 multipliers for diminishedrepresentation," IET Circuits, Devices, Syst., vol. 4, no. 4, pp. 291–300, Jul. 2010.

[5] H. T. Vergos, C. Efstathiou, and D. Nikolos, "Diminished-one modulo 2^n+1 adder design," IEEE Trans.Comput., vol. 51, no. 12, pp.1389–1399, Dec. 2002.

[6] V. Paliouras and T. Stouraitis, "Multifunction architectures for RNS processors," IEEE Trans. Circuits Syst. II,Analog Digit. Signal Process. vol. 46, no. 8, pp. 1041–1054, Aug. 1999.

[7] G. C. Cardarilli, A. Del Re, A. Nannarelli, and M. Re, "Residue Number System reconfigurable datapath," in Proc. IEEE Int. Symp. Circuits Syst., Scottsdale, AZ, USA, May 2002, vol. 2, pp. 756–759.

[8] C. Efstathiou, H. T. Vergos, and D. Nikolos, "Modified Booth modulo 2^n-1 multiplier," IEEE Trans. Comput., vol. 53, no. 3, pp. 370–374, Mar. 2004.

[9] W. K. Jenkins and B. A. Schnaufer, "Fault Tolerant architectures for efficient realization of common DSP kernels," in Proc. 35th Midwest Symp. Circuits Syst., Washington, DC, USA, Aug. 1992, vol. 2, pp. 1320–1323.

[10] H. T. Vergos and D. Bakalis, "Area-time efficient multi-modulus adders and their applications," Microprocessors Microsyst., vol. 36, no. 5, pp. 409–419, Jul. 2012.

[11] C. H. Chang, S. Menon, B. Cao, and T. Srikanthan, "A configurable dual-moduli multi-operand modulo adder," in Proc. IEEE Int. Symp. Circuits Syst., Kobe, Japan, May 2005, vol. 2, pp. 1630–1633.

[12] S. Menon and C. H. Chang, "A reconfigurable multi-modulus modulo multiplier," in Proc. IEEE Asia-Pacific Conf. Circuits Syst., Singapore, Dec. 2006, pp. 1168–1171.

[13] D. Bakalis and H. T. Vergos, "Area-efficient multi-moduli squarer for RNS," in Proc. 13th Euromicro Conf. Digit. Syst. Design, Lille, France, Sep. 2010, pp. 408–411.