A VLSI Implementation of Modulo $2^n - 1$ Multiplier By Using Radix-8 Modified Booth Algorithm

M. Ashokchakravarthi, K.V. Ramana Rao

Abstract: ***A special moduli set Residue Number System (RNS) of high dynamic range (DR) can speed up the execution of very large word-length repetitive multiplications found in applications like public key cryptography. The modulo 2^n - 1 multiplier is usually the noncritical datapath among all modulo multipliers in such high-DR RNS multiplier. This timing slack can be exploited to reduce the system area and power consumption without compromising the system performance. With this precept, a family of radix-8 Booth encoded modulo 2^n-1 multipliers, with delay adaptable to the RNS multiplier delay, is proposed. The modulo 2^n - 1 multiplier delay is made scalable by controlling the word-length of the ripple carry adder, k employed for radix-8 hard multiple generation. Formal criteria for the selection of the adder word-length are established by analyzing the effect of varying k on the timing of multiplier components. It is proven that for a given n, there exist a number of feasible values of k such that the total bias incurred from the partially-redundant partial products can be counteracted by only a single constant binary string. This compensation constant for different valid combinations of n and k can be precomputed at design time using number theoretic properties of modulo 2^n-1 arithmetic and hardwired as a partial product to be accumulated in the carry save adder tree. The proposed radix-8 Booth encoded modulo 2^n - 1 multiplier saves substantial area and power consumption over the radix-4 Booth encoded multiplier in medium to large word-length RNS multiplication.***

Index Terms**:** ***Booth algorithm, design space exploration, modulo arithmetic, multiplier, residue number system (RNS).***

I. INTRODUCTION

Rivest, Shamir, and Adleman (RSA) and elliptic curve cryptography (ECC) are two of the most well established and widely used public key cryptographic (PKC) algorithms. The encryption and decryption of these PKC algorithms are performed by repeated modulo multiplications [1]-[3]. These multiplications differ from those encountered in signal processing and general computing applications in their sheer operand size. Key sizes in the range of 512~1024 bits and 160~512 bit are typical in RSA and ECC, respectively [4]-[7]. Hence, the long carry propagation of large integer multiplication is the bottle-neck in hardware implementation of PKC. The residue number system (RNS) has emerged as a promising alternative number representation for the design of faster and low power multipliers owing to its merit to distribute a long integer multiplication into several shorter and independent modulo multiplications [8]-[11]. RNS has also been successfully employed to design fault tolerant digital circuits [12]-[13].

A RNS is defined by a set of N pair-wise co-prime moduli $\{L_1, L_2,\ldots,L_N\}$ such that any integer X within the dynamic range [DR]. i.e., $\Pi_{i=1}^{N} L_i$ is represented as an N-tuple $\{x_1,x_2,\ldots,x_N\}$, where x_i is the residue of X modulo L_i [14]-[16]. RNS multipliers based on generic moduli have been reported in [17]-[19].

The noncritical modulo multipliers can be made to operate at a slower speed that nearly matches the delay of the critical modulo multiplier. This paper focuses on the design space exploration of arithmetic operation in one of the two special moduli, i.e., the modulo 2^n -1 multiplier design.

The radix-8 Booth encoding reduces the number of partial products to $\lfloor n/3 \rfloor + 1$ which is more aggressive than the radix-4 Booth encoding. The shorthand notations $\lceil a \rceil$ and $\lfloor a \rfloor$ denote the smallest integer greater than or equal to a and the largest integer smaller than or equal to a, respectively. However, in the radix-8 Booth encoded modulo 2^n -1 multiplication, not all modulo-reduced partial products can be generated using the bitwise circular-left-shift operation and bitwise inversion. Particularly, the hard multiple $|3X|_{2^n-1}$ is to be generated by an n-bit end-around-carry addition of X and $2X$. . The performance overhead due to the end-around-carry addition is by no means trivial.

In this paper, we propose the first ever family of low-area and low-power radix-8 Booth encoded modulo 2^n-1 multipliers whose delay can be tuned to match the RNS delay closely. In the proposed multiplier, the hard multiple is generated using small word-length ripple carry adders (RCAs) operating in parallel. The carry-out bits from the adders are not propagated but treated as partial product bits to be accumulated in the CSA tree. The effect of the RCA word length, k on the time complexities of each constituent component of the multiplier is analyzed qualitatively and the multiplier de- lay is shown to be almost linearly dependent on the RCA word-length. Consequently, the delay of the modulo 2^n -1 multiplier can be directly controlled by the word-length of the RCAs to equal the delay of the critical modulo multiplier of the RNS.

Revised Manuscript Received on 30 October 2012
*****Correspondence Author(s)

M. Ashokchakravarthi*, M.Tech Department of ECE, JNTU Kakinada University/ Pydah College of Engineering and Technology, Visakhapatnam, India.

K.V. Ramana Rao, Associate Professor & Head of the Department, Department of ECE, Pydah College of Engineering and Technology, Visakhapatnam, India.

By means of modulo 2^n -1 arithmetic properties, we show that the compensation constant that negates the effect of the biased introduced in this process can be pre computed and implemented by direct hardwiring with no delay overhead for all feasible combinations of n and k. It

TABLE I
MODULO-REDUCED MULTIPLES FOR THE RADIX-8BOOTH ENCODING

d_i	$\lvert d_i \cdot X\rvert_{2^n-1}$	d_i	$\lvert d_i \cdot X\rvert_{2^n-1}$
+0	$\underbrace{0\cdots0}_{n}$	−0	$\underbrace{1\cdots1}_{n}$
+1	X	−1	$\bar{X}$
+2	$CLS(X, 1)$	−2	$CLS(\bar{X},1)$
+3	$\lvert +3X\rvert_{2^n-1}$	−3	$\lvert -3X\rvert_{2^n-1}$
+4	$CLS(X, 2)$	−4	$CLS(\bar{X},2)$

is shown that the proposed multiplier lowers the area and power dissipation of the radix-4 Booth encoded modulo 2^n-1 multiplier under the delay constraints derived from various high dynamic range RNS multipliers.

II. RADIX-8 BOOTH ENCODED MODULO 2^n-1 MULTIPLICATION ALGORITHM

Let $X = \sum_{i=0}^{n-1} x_i . 2^i$ and $Y = \sum_{i=0}^{n-1} y_i . 2^i$ represent the multiplicand and the multiplier of the modulo 2^n-1 multiplier, respectively. The radix-8 Booth encoding algorithm can be viewed as a digit set conversion of four consecutive overlapping multiplier bits, $y_{3i+2}y_{3i+1}y_{3i}y_{3i-1}$ to a signed digit, d_i, d_j Є [-4,4] for i = 0,1,…, $\lfloor n/3 \rfloor$.The digit set conversion is formally expressed as

$$d_i = y_{3i-1} + y_{3i} + 2y_{3i+1} - 4y_{3i+2} \quad (1)$$

where $y_{-1} = y_n = y_{n+1} = y_{n+2} = 0$ [31].

Table I summarizes the modulo-reduced multiples of X for all possible values of the radix-8 Booth encoded multiplier digit, d_i, where CLS(X, j) denotes a circular-left –shift of X by j bit positions.

Three unique properties of modulo 2^n-1 arithmetic that will be used for simplifying the combinatorial logic circuit of the proposed modulo multiplier design are reviewed here.

1) Property 1: The modulo 2^n -1 reduction of –X can be implemented as the n-bit one's complementation of the binary word X as follows.

$$|-X|_{2^n-1} = 2^n - 1 - X = \bar{X}. \qquad (2)$$

2) Property 2: For any nonnegative integers, s, the periodicity of an integer power of two over modulus 2^n -1can be stated as follows [32].

$$\left|2^{n.s+i}\right|_{2^n-1} = \left|\left|2^{n.s}\right|_{2^n-1} \cdot \left|2^i\right|_{2^n-1}\right|_{2^n-1} = \left|2^i\right|_{2^n-1}. \quad (3)$$

Property2 ensures that the modulo 2^n-1 reduction of binary exponents can be implemented with no logic cost. As a corollary, the modulo 2^n-1 reduction of the product of a binary word X and an integer power of two, 2^j , is equivalent to CLS(X, j) [14]. This property can be formally expressed as *Property 3*

3) Property 3: For $j<n$

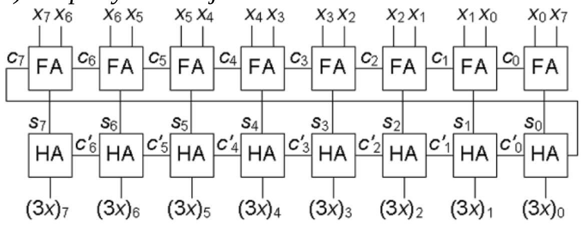

Fig. 1. Generation of $|3X|_{2^n-1}$ using two n - bit RCAs.

$$|2^j X|_{2^n-1} = \sum_{i=0}^{n-j-1} x_i \cdot 2^{i+j} + \sum_{i=n-j}^{n-1} x_i \cdot 2^{i+j-n} = CLS(X, j). \quad (4)$$

In Table I, the modulo 2^n-1 reduction for d_i Є {±0, ±1, ±2, ±4} are replaced by simple bitwise inversion and bitwise circular-left-shift of X using *Property 1* and *Property 3* , respectively.

In contrast, the carry propagation addition of X and $2X$ is unavoidable in the computation of the hard multiple $|3X|_{2^n-1}$. Numerous modulo 2^n-1 adders employing parallel-prefix structure with additional prefix operators for the end-around-carry addition have been proposed in literature [30], [33]–[36]. In [37], a high-speed application-specific modulo 2^n-1adder that computes merely the hard multiple using the fewest number of prefix levels was proposed. The use of such area intensive adders for the one-time computation of the hard multiple diminishes the area savings gained by higher radix Booth encoding.

Of all possible two operand adder implementations, the RCA has indubitably the least area and dynamic power dissipation [29], [38]. Fig. 1 illustrates the computation of $|3X|_{2^n-1}$ by an n-bit end-around-carry addition of $|X|_{2^n-1}$ and $|2X|_{2^n-1}$ using RCAs for n=8. The addends $|X|_{2^n-1}$ and $|2X|_{2^n-1}$ are added with carry propagation through full adders (FAs), and the end-around-carry addition is realized with carry propagation through half adders (HAs), as shown in Fig. 1. The above technique for $|3X|_{2^n-1}$ computation involves two n-bit carry-propagate additions in series such that the carry propagation length is twice the operand length, n. In the worst case, the late arrival of the $|3X|_{2^n-1}$ may considerably delay all subsequent stages of the modulo 2^n-1 multiplier. Hence, this approach for hard multiple generation can no longer categorically ensure that the multiplication in the modulo 2^n-1 channel still falls in the noncritical path of a RNS multiplier. In what follows, we propose a family of low-power and low- area modulo 2^n-1 multipliers based on the radix-8 Booth encoding, which allows for an adaptive control of the delay to match the delay of the critical modulo channel of a RNS multiplier.

III. PROPOSED RADIX-8 BOOTH ENCODED MODULO 2^n-1 MULTIPLIER DESIGN

To ensure that the radix-8 Booth encoded modulo 2^n-1 multiplier does not constitute the system critical path of a high DR moduli set based RNS multiplier, the carry propagation length in the hard multiple generation should not exceed *n*

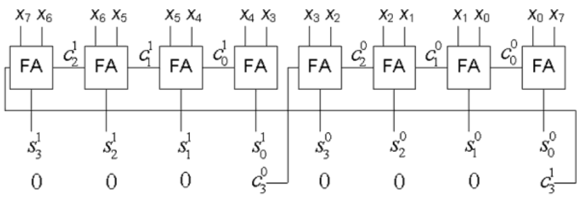
Fig.2. Generation of partially-redundant $|3X|_{2^n-1}$ using *k* bit RCAs.

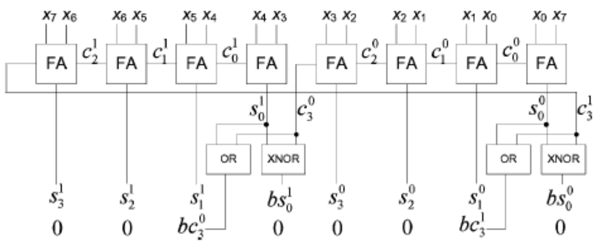
Fig.3. Generation of partially-redundant $|B+3X|_{2^n-1}$

bits. To this end, the carry propagation through the HAs in Fig. 1. Can be eliminated by making the end-around-carry bit c_7 a partial product bit to be accumulated in the CSA tree. This technique reduces the carry propagation length to *n* bits by representing the hard multiple as a sum and a redundant-end-around carry bit pair. The resultant $(\lfloor n/3\rfloor+1)$end-around-carry bits in the partial product matrix may lead to a marginal increase the CSA tree depth and consequently, may aggravate the delay of the CSA tree. In which case, it is not sufficient to reduce the carry propagation length to merely *n* bits using the above technique.

Since the absolute difference between the noncritical modulo 2^n-1 multiplier delay and the system critical path delay depends on the degree of imbalance in the moduli word-length of a RNS, the delays cannot be equalized by arbitrarily fixing the carry propagation length to *n*- bits. Instead, we propose to accomplish the adaptive delay equalization by representing the hard multiple in a partially-redundant form [39].

A. Generation of Partially-Redundant Hard Multiple

Let $|X|_{2^n-1}$ and $|2X|_{2^n-1}$ be added by group of *M* (=*n*/*k*) *k*-bit RCAs such that there is no carry propagation between the adders. Fig. 2. Shows this addition for *k*=8 and *k*=4, where the sum and carry-out bits from the RCA block *j* are represented as s_i^j and c_i^j for *j* Є [0, *k*-1] and *j* Є [0, *M*-1] respectively. In Fig. 2, the carry-out of RCA 0, c_3^0 is not propagated to the carry input of RCA 1 but preserved as one of the partial product bits to be accumulated in the CSA tree.

The binary weight of the carry-out c_3^1 of RCA 1 has however, exceeded the maximum range of the modulus and has to be modulo reduced before it can be accumulated by the CSA tree.

By *Property 2*, the binary weight of c_3^1 can be reduced from 2^8 to 2^0. Thus, c_3^1 is inserted at the least significant bit (lsb) position in Fig. 2. It should be stressed that the carry-out c_3^1 is a partial carry propagated through only *k* most significant FAs and hence, is different from the end-around-carry bit in the modulo 2^n-1 addition of *X* and *2X*,

0	0	0	1	0	0	0	1	B+0
		0				0		
x_7	x_6	x_5	$\bar{x}_4$	x_3	x_2	x_1	$\bar{x}_0$	B+X
		x_4				x_0		
x_6	x_5	x_4	$\bar{x}_3$	x_2	x_1	x_0	$\bar{x}_7$	B+2X
		x_3				x_7		
x_5	x_4	x_3	$\bar{x}_2$	x_1	x_0	x_7	$\bar{x}_6$	B+4X
		x_2				x_6		

Fig.4. Generation of partially-redundant simple multiples

	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0
X						d_2	d_1	d_0
	pp_{07}	pp_{06}	pp_{05}	pp_{04}	pp_{03}	pp_{02}	pp_{01}	pp_{00}
			q_{01}				q_{00}	
	pp_{17}	pp_{16}	pp_{15}	pp_{14}	pp_{13}	pp_{12}	pp_{11}	pp_{10}
				q_{10}				q_{11}
	pp_{27}	pp_{26}	pp_{25}	pp_{24}	pp_{23}	pp_{22}	pp_{21}	pp_{20}
	q_{20}				q_{21}			
	0	0	1	0	0	0	1	0

Fig.5. Modulo-reduced partial products and CC for $|X.Y|_{2^8-1}$

i.e., c_7 of Fig. 1.

From Fig. 2, the partially-redundant form of $|3X|_{2^n-1}$ is given by the partial-sum and partial-carry pair (*S*, *C*), where

$$S = s_{k-1}^{M-1} s_{k-2}^{M-1} \cdots s_0^{M-1} \cdots s_{k-1}^0 s_{k-2}^0 \cdots s_0^0$$

$$C = \underbrace{0\cdots0}_{k-1} c_{k-1}^{M-2} \cdots \underbrace{0\cdots0}_{k-1} c_{k-1}^0 \underbrace{0\cdots0}_{k-1} c_{k-1}^{M-1}. \quad (5)$$

Since modulo 2^n-1 negation is equivalent to bitwise complementation by *Property 1*, the negative hard multiple in a partially-redundant form, $|-3X|_{2^n-1} = (\bar{S}, \bar{C})$, is computed as follows:

$$\bar{S} = \bar{s}_{k-1}^{M-1} \bar{s}_{k-2}^{M-1} \cdots \bar{s}_0^{M-1} \cdots \bar{s}_{k-1}^0 \bar{s}_{k-2}^0 \cdots \bar{s}_0^0$$

$$\bar{C} = \underbrace{1\cdots1}_{k-1} \bar{c}_{k-1}^{M-2} \cdots \underbrace{1\cdots1}_{k-1} \bar{c}_{k-1}^0 \underbrace{1\cdots1}_{k-1} \bar{c}_{k-1}^{M-1}. \quad (6)$$

To avoid having many long strings of ones in $\bar{C}$, an appropriate bias. *B*, is added to the hard multiple such that both *C* and $\bar{C}$ are sparse [48]. The value of *B* is chosen as

$$B = \sum_{j=0}^{M-1} 2^{k\cdot j} = \overbrace{\underbrace{0\cdots01}_{k}\cdots\underbrace{0\cdots01}_{k}}^{n}. \qquad (7)$$

The addends for the computation of the biased hard multiple, $|B+3X|_{2^n-1}$ in a partially-redundant form are $|X|_{2^n-1}$, $|2X|_{2^n-1}$ and B or equivalently S, C and B. Since B is chosen to be a binary word that has logic ones at bit positions 2^{kj} and logic zeros at other bit positions, $|B+3X|_{2^n-1}$ can be generated by simple XNOR and OR operations on the bits of S and C at bit positions 2^{kj}. Fig. 3 illustrates how these bits in the sum and the carry

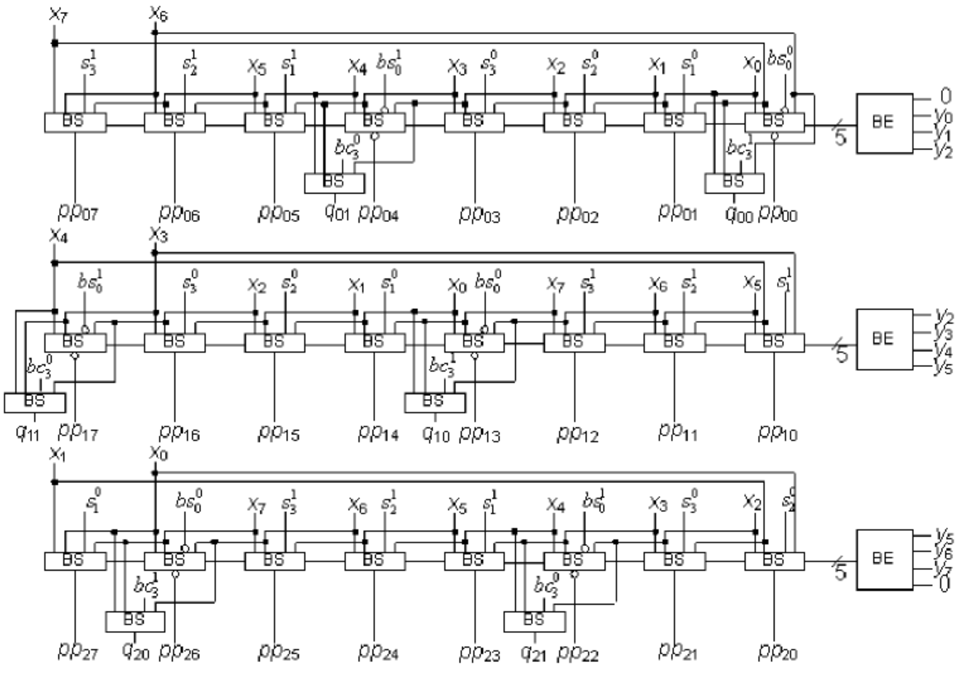

Fig.6. Modulo-reduced partial product generation.

outputs of RCA 0 and RCA 1 are modified.

In general, $|B+3X|_{2^n-1}$ is given by the partial-sum and partial-carry pair (BS, BC) such that

$$BS = s_{k-1}^{M-1}s_{k-2}^{M-1}\cdots bs_0^{M-1}\cdots s_{k-1}^{0}s_{k-2}^{0}\cdots bs_0^{0}$$

$$BC = \underbrace{0\cdots0}_{k-2}bc_{k-1}^{M-2}0\cdots\underbrace{0\cdots0}_{k-2}bc_{k-1}^{0}0\underbrace{0\cdots0}_{k-2}bc_{k-1}^{M-1}0 \qquad (8)$$

where

$$bs_0^j = \begin{cases} \overline{s_0^j \oplus c_{k-1}^{j-1}} & \text{when } j \neq 0 \\ \overline{s_0^0 \oplus c_{k-1}^{M-1}} & \text{when } j = 0 \end{cases} \qquad (9)$$

and

$$bc_{k-1}^j = \begin{cases} s_0^{j+1} + c_{k-1}^j & \text{when } j \neq M-1 \\ s_0^0 + c_{k-1}^{M-1} & \text{when } j = M-1 \end{cases} \qquad (10)$$

for j= 0,1,......, M-1

Let

$$\overline{BS} = \overline{s_{k-1}^{M-1}s_{k-2}^{M-1}}\cdots\overline{bs_0^{M-1}}\cdots\overline{s_{k-1}^{0}s_{k-2}^{0}}\cdots\overline{bs_0^{0}}$$

$$\overline{BC} = \underbrace{0\cdots0}_{k-2}\overline{bc_{k-1}^{M-2}}0\cdots\underbrace{0\cdots0}_{k-2}\overline{bc_{k-1}^{0}}0\underbrace{0\cdots0}_{k-2}\overline{bc_{k-1}^{M-1}}0. \qquad (11)$$

It can be easily verified that the sum of (BS, BC) and $(\overline{BS}, \overline{BC})$ modulo 2^n-1 is $|2B|_{2^n-1}$. Therefore, $(\overline{BS}, \overline{BC})$ represents the partially-redundant form of $|B-3X|_{2^n-1}$.

B. Generation of Partially-Redundant Simple Multiples

The proposed technique represents the hard multiple in a biased partially-redundant form. Since the occurrences of the hard multiple cannot be predicted at design time, all multiples must be uniformly represented. Similar to the hard multiple, all other Booth encoded multiples listed in Table I must also be biased and generated in a partially-redundant form. Fig.4 shows the biased simple multiples, $|B+0|_{2^n-1}$, $|B+X|_{2^n-1}$, $|B+2X|_{2^n-1}$ and $|B+4X|_{2^n-1}$ represented in a partially-redundant form for n=8. From Fig. 4, it can be seen that the generation of these biased multiples involves only shift and selective complementation of the multiplicand bits without additional hardware overhead.

C. Radix-8 Booth Encoded Modulo 2^n-1 Multiplication with Partially –Redundant Partial Products

The i-th partial product of a radix-8 Booth encoded modulo 2^n-1 multiplier is given by

$$PP_i = \left|2^{3i}\cdot d_i \cdot X\right|_{2^n-1}. \qquad (12)$$

To include the bias B necessary for partially-redundant representation of PP_i , (12) is modified to

$$PP_i = \left|2^{3i}\left(B + d_i \cdot X\right)\right|_{2^n-1}. \qquad (13)$$

Using *property* 3, the modulo 2^n-1 multiplication by 2^{3i} in (13) is efficiently implemented as bitwise circular-left-shift of the biased multiple, (B+ d_i. X).

For n=8 and k=4, Fig. 5 illustrates the partial product matrix of $|XY|_{2^8-1}$, with $(\lfloor n/3 \rfloor + 1)$ partial products in partially-redundant representation. Each PP_i consists of an n-bit vector $PP_{i7}....PP_{i1}PP_{i0}$ and a vector of n/k=2 redundant carry bits, q_{i1} and q_{i0} . Since q_{i0} and q_{i1} are the carry-out bits of the RCAs, they are displaced by k-bit positions for a given PP_i. The bits, q_{ij} is displaced circularly to the left of $q_{(i-1)j}$ by 3 bits, i.e., q_{20} and q_{21} are displaced circularly to the left of q_{10} and q_{11} by 3 bits respectively and q_{10} and q_{11} are in turn displaced to the left of q_{00} and q_{01} by 3 bits, respectively. The last partial product in Fig. 5 is the Compensation Constant (CC) for the bias introduced in the partially redundant representation. The derivation of this constant is detailed in Section IV and the Appendix. The generation of the modulo-reduced partial products,

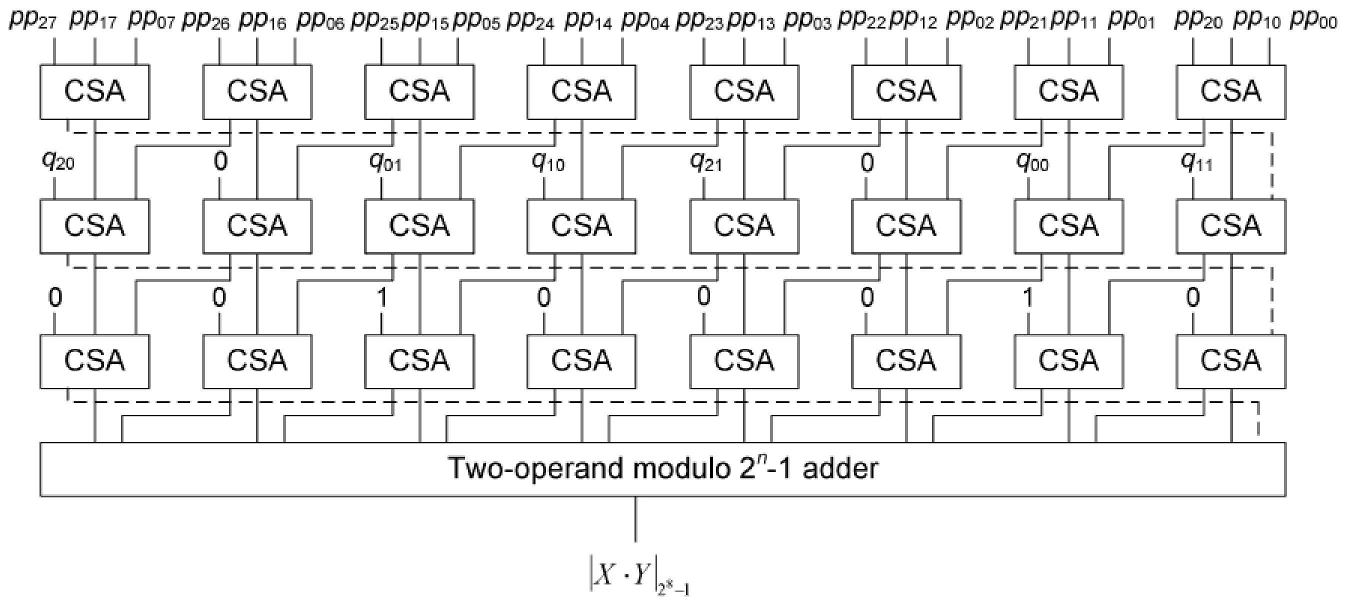
pp27 pp17 pp07 pp26 pp16 pp06 pp25 pp15 pp05 pp24 pp14 pp04 pp23 pp13 pp03 pp22 pp12 pp02 pp21 pp11 pp01 pp20 pp10 pp00
CSA
q20
0
q01
q10
q21
0
q00
q11
0
0
1
0
0
0
1
0
Two-operand modulo 2^n-1 adder
|X·Y|_{2^8-1}

Fig.7. Modulo-reduced partial product accumulation

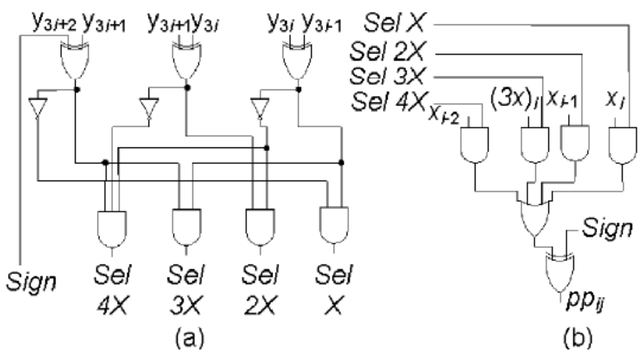

Fig.8 (a) Bit-slice of Booth Encoder (BE). (b) Bit-slice of Booth Selector (BS).

PP_0,PP_1 and PP_2 in a partially-redundant representation using Booth Encoder (BE) and Booth Selector (BS) blocks are illustrated in Fig. 6. The BE block produces a signed one-hot encoded digit from adjacent overlapping multiplier bits as illustrated in Fig. 8(a). The signed one-hot encoded digit is then used to select the correct multiple to generate PP_i . A bit-slice of the radix-8 BS for the partial product bit, PP_{ij} is shown in Fig. 8(b).

As the bit positions of q_{ij} do not overlap, as shown in Fig. 5, they can be merged into a single partial product for accumulation. The merged partial products, PP_i and the constant CC are accumulated using a CSA tree with end-around-carry addition at each CSA level and a final two-operand modulo 2^n-1 adder as shown in Fig. 7.

IV. PARALLEL PREFIX ADDER

A parallel prefix adder can be seen as a 3-stage process as shown in fig.9.

A. Pre-computation

In pre-computation stage, each bit computes its carry generate (g)/propagate (p) signals as below. These two signals are said to describe how the Carry-out signal will be handled.

$$g_i = a_i . b_i$$
$$p_i = a_i \oplus b_i \qquad (14)$$

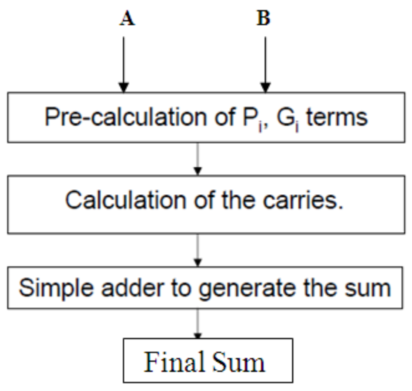
A
B
Pre-calculation of Pi, Gi terms
Calculation of the carries.
Simple adder to generate the sum
Final Sum

Fig.9 Parallel Prefix Addition Process Steps.

B. Prefix Stage

In the prefix stage, the group carry generate/propagate signals are computed to form the carry chain and provide the carry-in for the adder in the next stage. Various signal graphs/architectures can be used to calculate the carry-outs for the final sum.

Ladner-Fischer parallel prefix structure is one of the efficient architecture as shown in fig.10.

C. Post-computation

In the post-computation stage, the sum and carry-out are finally produced. The carry-out can be omitted if only a sum needs to be produced.

$$s_i = p_i \oplus c_i \qquad (15)$$

V. SELECTION OF *k*

The guidelines for choosing the RCA word-length, *k*, to achieve the desired performance are presented in this section.

Firstly, irrespective of the targeted delay, the choice of *k* must satisfy the following two criteria.

1) *Criterion 1:* As the residues of modulus 2^n-1are represented using only *n* bits, it is imperative that *k* divides *n*.*k*=1 is a trivial case and is excluded from this consideration. This criterion is expressed as k/n, $k \neq 1$.

2) *Criterion 2:* Since each partial product in radix-8 Booth

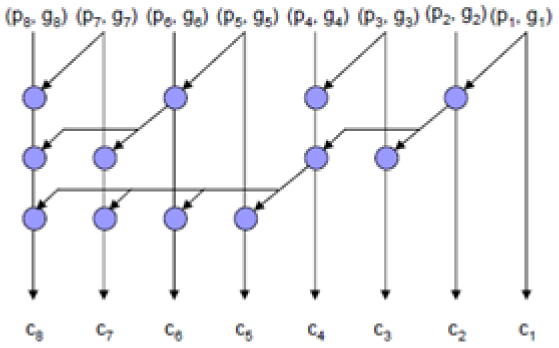

Fig.10 Ladner-Fischer adder

TABLE II
TIME COMPLEXITIES OF MODULO MULTIPLIER COMPONENTS

Component	*Time Complexity*
Hard multiple generation by *k*-bit RCAs	$O(k)$
CC generation by hardwiring	$O(1)$
Partial product generation by BE and BS blocks	$O(1)$
Partial product accumulation by CSA tree	$O(\log(n+n/k))$
Two-operand parallel-prefix modulo 2^n–1 adder	$O(\log n)$

encoding is shifted by three bits relative to the previous partial product, *k* must not be a multiple of three to ensure that the q_{ij} bits are nonoverlapping. Therefore, 3†*k*.

In the proposed modulo 2^n-1 multiplier, each partial product PP_i is incremented by a bias of $2^{3i} \times B$ as expressed in (13).To negate the effect of the bias, a constant *CC* is added and the value of *CC* is given by

$$CC = \left| -\sum_{i=0}^{\lfloor n/3 \rfloor} B.2^{3i} \right|_{2^n-1} \qquad (16)$$

where *B* is an *n*-bit binary word consisting of logic one at bit position 2^{kj} , j Є [0,*M*-1] and logic zero at all other positions as defined is (7).

It is evident that the value of *CC* depends only on *n* and *k* As *CC* is considered as one or more partial products to be summed in the CSA tree, the choice of *k* indirectly determines the regularity of the multiplier design and consequently its efficiency in VLSI implementation. A detailed analysis on the computation of *CC* for various combinations of *n* and *k* is presented in the Appendix. For any *k* that satisfies *Criteria 1* and *2*, it is shown that *CC* can be simplified by the properties of modulo 2^n -1 arithmetic and precomputed at design time. The resultant *CC* is shown to be a single binary word with a specific repetitive pattern of logic ones and zeros. As the generation of *CC* involves merely the assignment of logic constants to appropriate bit positions, it can be directly hardwired into the CSA tree as a constant partial product without any logic circuitry.

The effect of *k* on the delay of the constituent components of a radix-8 Booth encoded modulo 2^n -1 multiplier is analyzed qualitatively and summarized in Table II. As indicated in Table II, the partial products and *CC* can be generated in constant time. Similarly the delay of the final two-operand parallel-prefix modulo 2^n-1 adder is independent of *k* . From Table II, by reducing *k*, the delay of the RCA reduces linearly but the delay of the CSA tree stage increases only logarithmically. Hence, the delay of the

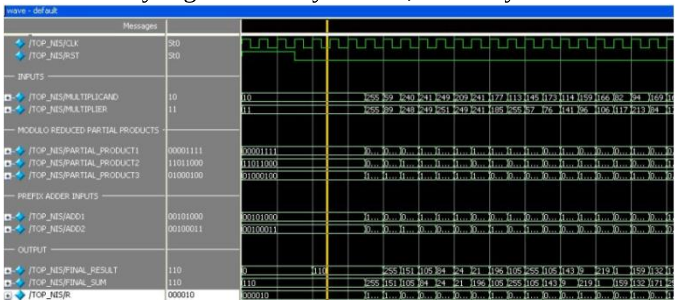

Fig.11. Output waveform of Radix-8 Booth Encoded Modulo 2^n-1Multiplier for various inputs.

modulo 2^n -1 multiplier is logarithmically dependent on *n* and almost linearly dependent on *k*. For a given *n*, the modulo 2^n-1 multiplier delay can be manipulated by varying the word-length of the RCA, *k*.

VI. SIMULATION RESULT

Fig.11 shows the output waveforms of Radix-8 Booth Encoded Modulo 2^n-1 multiplier for various inputs. If the number of bits for multiplier and multiplicand are 8 i.e, *n*=8 that means modulo 255 multiplier. The modulo result is the least positive remainder when the decimal multiplication result of the inputs is divided by the modulus 255. Hence if the decimal multiplication result is less than 255, the modulo result is the same as the decimal multiplication result of the inputs. If the decimal multiplication result of the inputs is 255, the modulo result is also same i.e, 255.

VII.CONCLUSION

A family of low-area and low-power modulo 2^n-1 multipliers with variable delay to achieve delay balance amongst individual modulo channels in a high-DR RNS multiplier was proposed. The delay of the proposed multiplier is controlled by the word-length of the small parallel RCAs that are used to compute the requisite hard multiple of the radix-8 Booth encoded multiplication in a partially-redundant form.

The trade-offs between the RCA word-length and the VLSI performance metrics are area, delay and power dissipation of the modulo 2^n-1 multiplier. From synthesis results constrained by the critical channel delay of the RNS, the proposed multiplier simultaneously reduces the area as well as the power dissipation of the radix-4 Booth encoded multiplier for $n \geq 28$, which is the useful dynamic range of RNS multiplication to meet the minimum key-size requirements of ECC and RSA algorithms.

REFERENCES

1. R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," Commun. ACM, vol. 21, no.2, pp. 120–126, Feb. 1978.
2. V.Miller, "Use of elliptic curves in cryptography," in Proc. Advances in Cryptology- CRYPTO'85, Lecture Notes in Computer Science, 1986, vol.218, pp. 417-426.
3. N. Koblitz, "Elliptic curve cryptosystems," Mathemat. of Comput., vol.48, no. 177, pp. 203–209, Jan. 1987.
4. National Institute of Standards and Technology [Online]. Available: http://csrc.nist.gov/publications/PubsSPs.html
5. A. K. Lenstra and E. R. Verheul, "Selecting cryptographic key sizes,"
6. J. Cryptol., vol. 14, no. 4, pp. 255–293, Aug. 2001.
7. C. McIvor, M. McLoone, and J. V. McCanny, "Modified Montgomery
8. modular multiplication and RSA exponentiation techniques,"IEE Proc. Comput. and Dig.Techniq., vol.151, no. 6, pp. 402–408, Nov.2004.
9. C. McIvor, M. McLoone, and J. V. McCanny, "Hardware elliptic curve
10. cryptographic processors over GF (p) ," IEEE Trans. Circuits Syst.I, Reg. Papers, vol. 53, no. 9, pp. 1946–1957, Sep. 2006.
11. J. C. Bajard and L. Imbert, "A full RNS implementation of RSA," IEEE Trans. Comput. – Brief Contributions, vol. 53, no. 6, pp. 769–774,
12. Jun. 2004.
13. J. C. Bajard and L. Imbert, "A full RNS implementation of RSA," IEEE Trans. Comput. – Brief Contributions, vol. 53, no. 6, pp. 769–774,
14. Jun. 2004.
15. H. Nozaki, M. Motoyama, A. Shimbo, and S. Kawamura, "Implementation of RSA algorithm based on RNS Montgomery multiplication." in Proc. Workshop on Cryptographic Hardware and Embedded Systems, Paris, France, May 2001, pp. 364–376.
16. T. Stouraitis and V. Paliouras, "Considering the alternatives in low-power design," IEEE Circuits Devices Mag., vol. 17, no. 4, pp. 22–29, Jul. 2001.
17. S. Pontarelli, G. C. Cardarilli, M. Re, and A. Salsano, "Totally fault tolerant RNS based FIR filters," in Proc. 14th IEEE Int. On-Line Testing Symp., Rhodes, Greece, Jul. 2008, pp. 192–194.
18. I. Steiner et al., "A fault-tolerant modulus replication complex FIR
19. filter," in Proc. 16th IEEE Int. Conf. Application-Specific Systems Architecture and Processors, Samos, Greece, Jul. 2005, pp.
20. 387–392.
21. N.S. Szabo and R. I. Tanaka, Residue Arithmetic and its Applications
22. to Computer Technology. New York: McGraw-Hill, 1967.
23. M. A. Soderstrand, W. K. Jenkins, G. A. jullien , and F. J. Taylor, Residue Number System Arithmetic: Modern Applications in Digital Signal processing. New York: IEEE Press, 1986.
24. P. V. A. Mohan, Residue Number Systems: Algorithms and Architectures. Norwell, MA: Kluwer, 2002.
25. I. Kouretas and V. Paliouras, "A low-complexity high-radix RNS multiplier," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 56, no. 11, pp.2449–2462, Nov. 2009.
26. G. Dimitrakopoulos and V. Paliouras, "A novel architecture and a sys-
27. tematic graph-based optimization methodology for modulo multiplication," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 51, no. 2, pp.354–370, Feb. 2004.
28. A. A. Hiasat, "New efficient structure for a modular multiplier for
29. RNS," IEEE Trans. Comput., vol. 49, no.2, pp. 170-174, Feb.2000.
30. A. A. Hiasat and H. S. Abdel-Aty-Zohdy, "Residue-to-binary arith-
31. metic converter for the moduli set $\{2^k,2^k\text{-}1,2^{k-1}\text{-}1\}$,"IEEE Trans. Circuits Syst. II, Analog Digit . Signal Process., vol. 45, no. 2, pp. 204-209, Feb.1998.
32. B. Cao, C. H. Chang, and T. Srikanthan, "An efficient reverse converter
33. for the 4-moduli set $\{2^n\text{-}1,2^n,2^n+1,2^{2n}+1\}$ based on the New Chinese Remainder Theorem," IEEE Trans. Circuits Syst. I, Fundam. Theory
34. Appl., vol. 50, no. 10, pp. 1296–1303, Oct. 2003.
35. B. Cao, T. Srikanthan, and C. H. Chang, "Efficient reverse converters
36. for four-moduli sets $\{2^n\text{-}1,2^n,2^n+1,2^{n+1}\text{-}1\}$ and $\{2^n\text{-}1,2^n,2^n+1,2^{n\text{-}1}\text{-}1\}$,"
37. IEE Proc. Comput., Dig.Techniq., vol.152, no. 5, pp. 687–696, Sep. 2005.
38. B. Cao, C. H. Chang, and T.Srikanthan, "A residue-to-binary converter
39. for a new five-moduli set," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 54, no. 5, pp. 1041–1049, May 2007.
40. P. V. A. Mohan and A. B. Premkumar, "RNS-to-binary converters for
41. two four-moduli sets $\{2^n\text{-}1,2^n,2^n+1,2^{n+1}\text{-}1\}$ and$\{2^n\text{-}1,2^n,2^n+1,2^{n+1}+1\}$
42. ,"IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 54, no. 6, pp. 1245–
43. 1254, Jun. 2007.
44. P. V. A. Mohan, "RNS-to-binary converter for a new three moduli set
45. $\{2^{n+1}\text{-}1, 2^n, 2^n\text{-}1\}$," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 54, no. 9, pp. 775–779, Sep. 2007.
46. T. Tomczak, "Fast sign detection for RNS $\{2^n\text{-}1, 2^n, 2^n+1\}$," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 55, no. 6, pp. 1502–1511, Jul.2008.
47. T. Stouraitis, S. W. Kim, and A. Skavantzos, "Full adder-based arithmetic units for finite integer rings," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 40, no. 11, pp. 740–745, Nov.1993.
48. D. J. Soudris, V. Paliouras, T. Stouraitis, and C. E. Goutis, "A VLSI
49. design methodology for RNS full adder-based inner product architectures," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol.44, no. 4, pp. 315–318, Apr. 1997.
50. J. M. Rabaey, A. Chandrakasan and B. Nikolic, Digital Integrated Cir-
51. cuits. Upper Saddle River, NJ: Prentice-Hall, 2003.
52. R. Zimmermann, "Efficient VLSI implementation of modulo $(2^n\pm1)$addition and multiplication," in Proc. 14th IEEE Computer Arithmetic, Adelaide, Australia, Apr. 1999, pp. 158–167.
53. M. J. Flynn and S. F. Oberman, Advanced Computer Arithmetic Design. New York: Wiley, 2001.
54. S. J. Piestrak, "Design of squares modulo A with low-level pipelining,"
55. IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 49, no.1 , pp. 31–41, Jan. 2002.
56. C. Efstathiou, D. Nikolos, and J. Kalamatianos, "Area-time efficient
57. modulo 2^n-1 adder design," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 41, no. 7, pp. 463–467, Jul. 1994.
58. L. Kalampoukas, D. Nikolos, C. Efstathiou, H. T. Vergos, and J. Kalamatianos, "High-speed parallel-prefix modulo 2^n-1 adders," IEEE Trans. Comput., vol. 49, no. 7, pp. 673–680, Jul. 2000.
59. G. Dimitrakopoulos, D. G. Nikolos, H. T. Vergos, D. Nikolos, and C
60. Efstathiou, "New architectures for modulo 2^n-1 adders," in Proc. 12th
61. IEEE Int. Conf. Electronics, Circuits and Systems, Gammarth, Tunisia, Dec. 2005, pp. 1–4.
62. R. A. Patel, M. Benaissa, and S. Boussakta, "Fast Parallel-prefix architectures with a single representation of zero," IEEE Trans. Comput., vol. 56, no. 11, pp. 1484–1492, Nov.2007.
63. R. Muralidharan and C. H. Chang, "Fast hard multiple generators for radix-8 Booth encoded modulo 2^n-1 and modulo 2^n+1 multipliers," Proc. 2010 IEEE Int. Symp. Circuits and Systems, Paris, France, Jun.2010, pp. 717–720.
64. C. Nagendra, M. J. Irwin, and R. M. Owens, "Area-time-power trade-offs in parallel adders," IEEE Trans. Circuits Syst. II, Analog Digit.
65. Signal Process., vol. 43, no. 10, pp. 689–702, Oct. 1996.
66. G. W. Bewick, "Fast multiplication: Algorithms and implementation,"
67. Ph.D. dissertation, Stanford Univ., Stanford, CA, 1994.

AUTHOR PROFILE

Mr M. Ashokchakravarthi is pursuing M.Tech (VLSI) in the Department of ECE at Pydah College of Engineering & Technology, Visakhapatnam, A.P., India. His research interest includes VLSI Design and computer aided design.

Mr K.V. Ramana Rao working as Associate Professor & Head of The Department, ECE, Pydah College of Engineering & Technology, Visakhapatnam, A.P., India. His research interest includes Digital Signal Processing and VLSI Design.