Radix-8 Booth Encoded Modulo $2^n - 1$ Multipliers With Adaptive Delay for High Dynamic Range Residue Number System

Ramya Muralidharan, *Student Member, IEEE*, and Chip-Hong Chang, *Senior Member, IEEE*

***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 adaptive delay of the proposed family of multipliers is corroborated by CMOS implementations. In an RNS multiplier, when the critical modulo multiplier delay is significantly greater than the noncritical modulo $2^n - 1$ multiplier delay, $k = n$ and $k = n/3$ are recommended for n not divisible by three and divisible by three, respectively. Conversely, when this difference diminishes, k is better selected as $n/4$ and $n/6$ for n not divisible by three and divisible by three, respectively. Our synthesis results show that 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 bits are typical in RSA and ECC, respectively [4]–[7]. Hence, the long carry propagation of large integer multiplication is the bottleneck 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.,* $\prod_{i=1}^{N} L_i$ 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]. However, special moduli of forms 2^n or $2^n \pm 1$ are preferred over the generic moduli due to the ease of hardware implementation of modulo arithmetic functions as well as system-level inter-modulo operations, such as RNS-to-binary conversion and sign detection [20]–[26]. The most popular of these special moduli sets is the triple moduli set, $\{2^n - 1, 2^n, 2^n + 1\}$ which however has a DR of only $3n$ bits. It is obvious that the DR of an existing moduli set can be extended by appending many small word-length moduli or a few large word-length moduli. It has been shown that the speed of RNS processor is increasingly dominated by the residue arithmetic operation rather than the one-time forward or reverse conversion [27]–[29]. To facilitate design of high-speed full-adder based modulo arithmetic units, it is worthwhile to keep the moduli of a high-DR RNS in forms of 2^n or $2^n \pm 1$. The number of additional moduli that can be included to expand an existing moduli set is thus limited and high-DR moduli sets with imbalanced moduli word-lengths are unavoidable due to the conflicting requirements that the additional moduli be modulo-arithmetic friendly and at the same time, be relatively prime to the moduli in the existing set.

To transcend the $3n$-bit limit, moduli sets, $\{2^n - 1, 2^n, 2^n + 1, 2^{2n} + 1\}$, $\{2^{2n} - 1, 2^n, 2^{2n} + 1\}$ and $\{2^n - 1, 2^n, 2^n + 1, 2^{2n+1} - 1\}$ with DR of $5n$ bits and $\{2^n - 1, 2^n + 1, 2^{2n}, 2^{2n} + 1\}$ with DR of $6n$ bits, have been proposed recently [21], [30], [31]. Consequently, a RSA cryptosystem with a conservative key-length of 512 bits could be implemented in RNS using either the $5n$-bit DR moduli set with $n = 100$ or the $6n$-bit DR

Manuscript received April 23, 2010; revised September 10, 2010; accepted October 23, 2010. Date of publication December 20, 2010; date of current version April 27, 2011. This paper was recommended by Associate Editor C.-C. Wang.

The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798. (e-mail: ramy0003@ntu.edu.sg; echchang@ntu.edu.sg).

Digital Object Identifier 10.1109/TCSI.2010.2092133

moduli set with $n = 85$. For a ECC cryptosystem with a typical key-length of 256 bits, either the $5n$-bit DR moduli set with $n = 50$ or the $6n$-bit DR moduli set with $n = 42$ could be chosen.

The delay of an integer multiplication in RNS domain based on the $6n$-bit DR moduli set of [31] for example, is governed by the delay of the modulo $2^{2n} + 1$ multiplier. As the time complexity of partial product summation by a carry save adder (CSA) tree and a two-operand parallel-prefix adder is a logarithmic function of n, the critical path delay can be modeled as $O(\log 2n)$, but the delays of the modulo $2^n - 1$ and modulo $2^n + 1$ multipliers are only $O(\log n)$. This speedup of around $1/(1 + \log_2 n) \times 100\%$ by modulo $2^n - 1$ and modulo $2^n + 1$ multipliers over the critical path delay is of no consequence. As encryption and decryption in PKC involves repeated multiplications, the cumulative difference in the critical and noncritical modulo multiplier delays will increase with the number of multiplications involved. For lightweight cryptographic applications, such as smartcards and radio frequency identification (RFID) tags, the considerations of power, size and cost are of paramount importance [32]. The complexity of implementing reliable cryptographic hardware can be reduced by an ingenious exploitation of this timing headroom in the design of RNS multiplier.

The noncritical modulo multipliers can be made to operate at a slower speed that nearly matches the delay of the critical modulo multiplier. In doing so, the timing slack freed up from the modulo $2^n - 1$ and modulo $2^n + 1$ multipliers can be effectively explored for more area and power efficient architectures without compromising the overall system performance. This approach to reduce the overall area and power consumption of a RNS multiplier is based on architectural modification and can be implemented with any standard cell library. It does not require multiple supply voltages, multiple threshold voltages, or control circuitries for the generation and scaling of voltage and frequency [33]–[35] in order to exploit the timing surplus in the noncritical paths for power saving.

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 Montgomery modulo multiplication, while computing the modular product without trial division, is modulus-independent and incapable of exploiting number theoretic properties of modulo $2^n - 1$ arithmetic for combinational circuit simplification. The properties of modulo $2^n - 1$ arithmetic were most effectively exploited for the full adder based implementation of modulo $2^n - 1$ multiplier in [36]–[38]. In [36], the multiplier bits were not encoded, which lead to higher implementation area and longer partial product accumulation time. In [37] and [38], the radix-4 Booth encoding algorithm was employed to reduce the number of partial products to $\lfloor n/2 \rfloor + 1$ and $\lceil n/2 \rceil$, respectively. 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. With higher radix Booth encoding, the number of partial products is reduced by more than half and consequently, significant reduction in silicon area and power dissipation is feasible [39], [40]. 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. 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 and hence, the use of Booth encoding for modulo $2^n - 1$ multipliers have been restricted to only radix-4 in literature.

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

The paper is organized as follows. Section II describes the radix-8 Booth encoding algorithm for modulo $2^n - 1$ multiplication. A family of modulo $2^n - 1$ multipliers to adapt to different RNS delay is described in Section III. In Section IV, the criteria for selecting a suitable RCA word-length to achieve the desired performance are highlighted. The performance of the proposed family of modulo $2^n - 1$ multipliers is evaluated and compared against [38] in Section V. The paper is concluded in Section VI. The Appendix provides the derivation of the predetermined compensation constant for different valid combinations of the multiplier and RCA word-lengths.

II. Radix-8 Booth Encoded Modulo $2^n - 1$ Multiplication Algorithm

Let $X = \sum_{i=0}^{n-1} x_i \cdot 2^i$ and $Y = \sum_{i=0}^{n-1} y_i \cdot 2^i$ 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, $d_i, d_i \in [-4, 4]$, for $i = 0, 1, \ldots, \lfloor n/3 \rfloor$. The digit set conversion is formally expressed as

$$d_i = y_{3i-1} + y_{3i} + 2y_{3i+1} - 4y_{3i+2} \tag{1}$$

where $y_{-1} = y_n = y_{n+1} = y_{n+2} = 0$ [40].

TABLE I
MODULO-REDUCED MULTIPLES FOR THE RADIX-8 BOOTH 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	$\overline{X}$
+2	$CLS(X, 1)$	−2	$CLS(\overline{X},1)$
+3	$\lvert+3X\rvert_{2^n-1}$	−3	$\lvert-3X\rvert_{2^n-1}$
+4	$CLS(X, 2)$	−4	$CLS(\overline{X},2)$

Table I summarizes the modulo-reduced multiples of X for all possible values of the radix-8 Booth encoded multiplier digit, d_i, where $\mathrm{CLS}(X, j)$ 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:

$$\lvert -X\rvert_{2^n-1} = 2^n - 1 - X = \overline{X}. \tag{2}$$

2) Property 2: For any nonnegative integer, s, the periodicity of an integer power of two over modulus 2^n-1 can be stated as follows [41]:

$$\left\lvert 2^{n\cdot s+i}\right\rvert_{2^n-1} = \left\lvert \lvert 2^{n\cdot s}\rvert_{2^n-1} \cdot \lvert 2^i\rvert_{2^n-1}\right\rvert_{2^n-1} = \lvert 2^i\rvert_{2^n-1}. \tag{3}$$

Property 2 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 $\mathrm{CLS}(X, j)$ [14]. This property can be formally expressed as *Property 3*.

3) Property 3: For $j < n$

$$\left\lvert 2^j X\right\rvert_{2^n-1} = \sum_{i=0}^{n-j-1} x_i \cdot 2^{i+j} + \sum_{i=n-j}^{n-1} x_i \cdot 2^{i+j-n} = CLS(X, j). \tag{4}$$

In Table I, the modulo 2^n-1 reduction for $d_i \in \{\pm0, \pm1, \pm2, \pm4\}$ are replaced by simple bitwise inversion and bitwise circular-left-shift of X using *Properties 1* and *3*, respectively.

In contrast, the carry propagation addition of X and $2X$ is unavoidable in the computation of the hard multiple $\lvert+3X\rvert_{2^n-1}$. 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 [37], [42]–[45]. In [46], a high-speed application-specific modulo 2^n-1 adder that computes merely the hard multiple using the fewest number of prefix levels was proposed. The area complexity of these parallel-prefix modulo 2^n-1 adders is $O(\lceil\log_2 n\rceil \cdot n)$. 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 [33], [47]. Fig. 1 illustrates the computation of $\lvert+3X\rvert_{2^n-1}$ by

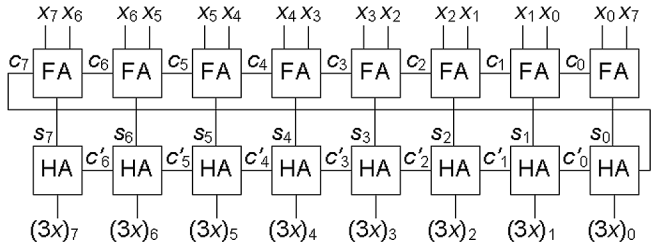

Fig. 1. Generation of $\lvert+3X\rvert_{2^n-1}$ using two n-bit RCAs.

an n-bit end-around-carry addition of $\lvert X\rvert_{2^n-1}$ and $\lvert 2X\rvert_{2^n-1}$ using RCAs for $n = 8$. The addends $\lvert X\rvert_{2^n-1}$ and $\lvert 2X\rvert_{2^n-1}$ 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 $\lvert+3X\rvert_{2^n-1}$ 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 $\lvert+3X\rvert_{2^n-1}$ 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 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 in 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 [48].

A. Generation of Partially-Redundant Hard Multiple

Let $\lvert X\rvert_{2^n-1}$ and $\lvert 2X\rvert_{2^n-1}$ be added by a group of $M(= n/k)k$-bit RCAs such that there is no carry propagation between the adders. Fig. 2 shows this addition for $n = 8$ and $k = 4$,

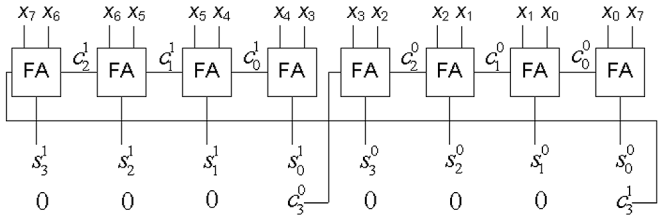

Fig. 2. Generation of partially-redundant $|+3X|_{2^n-1}$ using k-bit RCAs.

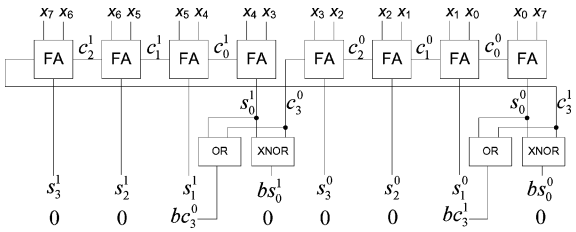

Fig. 3. Generation of partially-redundant $|B + 3X|_{2^n-1}$.

where the sum and carry-out bits from the RCA block j are represented as s_i^j and c_i^j for $i \in [0, k-1]$ and $j \in [0, M-1]$, 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 $2X$, *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

$$\begin{aligned} S &= s_{k-1}^{M-1} s_{k-2}^{M-1} \cdots s_0^{M-1} \cdots s_{k-1}^0 s_{k-2}^0 \cdots s_0^0 \\ C &= \underbrace{0 \cdots 0}_{k-1} c_{k-1}^{M-2} \cdots \underbrace{0 \cdots 0}_{k-1} c_{k-1}^0 \underbrace{0 \cdots 0}_{k-1} c_{k-1}^{M-1}. \end{aligned} \tag{5}$$

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:

$$\begin{aligned} \bar{S} &= \bar{s}_{k-1}^{M-1} \bar{s}_{k-2}^{M-1} \cdots \bar{s}_0^{M-1} \cdots \bar{s}_{k-1}^0 \bar{s}_{k-2}^0 \cdots \bar{s}_0^0 \\ \bar{C} &= \underbrace{1 \cdots 1}_{k-1} \bar{c}_{k-1}^{M-2} \cdots \underbrace{1 \cdots 1}_{k-1} \bar{c}_{k-1}^0 \underbrace{1 \cdots 1}_{k-1} \bar{c}_{k-1}^{M-1}. \end{aligned} \tag{6}$$

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

$$B = \sum_{j=0}^{M-1} 2^{k \cdot j} = \overbrace{\underbrace{0 \cdots 01}_{k} \cdots \underbrace{0 \cdots 01}_{k}}^{n}. \tag{7}$$

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 $|X \cdot Y|_{2^8-1}$.

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 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

$$\begin{aligned} BS &= s_{k-1}^{M-1} s_{k-2}^{M-1} \cdots bs_0^{M-1} \cdots s_{k-1}^0 s_{k-2}^0 \cdots bs_0^0 \\ BC &= \underbrace{0 \cdots 0}_{k-2} bc_{k-1}^{M-2} 0 \cdots \underbrace{0 \cdots 0}_{k-2} bc_{k-1}^0 0 \underbrace{0 \cdots 0}_{k-2} bc_{k-1}^{M-1} 0 \end{aligned} \tag{8}$$

where

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

and

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

for $j = 0, 1, \ldots, M-1$.

Let

$$\begin{aligned} \overline{BS} &= \overline{s_{k-1}^{M-1} s_{k-2}^{M-1}} \cdots \overline{bs_0^{M-1}} \cdots \overline{s_{k-1}^0 s_{k-2}^0} \cdots \overline{bs_0^0} \\ \overline{BC} &= \underbrace{0 \cdots 0}_{k-2} \overline{bc_{k-1}^{M-2}} 0 \cdots \underbrace{0 \cdots 0}_{k-2} \overline{bc_{k-1}^0} 0 \underbrace{0 \cdots 0}_{k-2} \overline{bc_{k-1}^{M-1}} 0. \end{aligned} \tag{11}$$

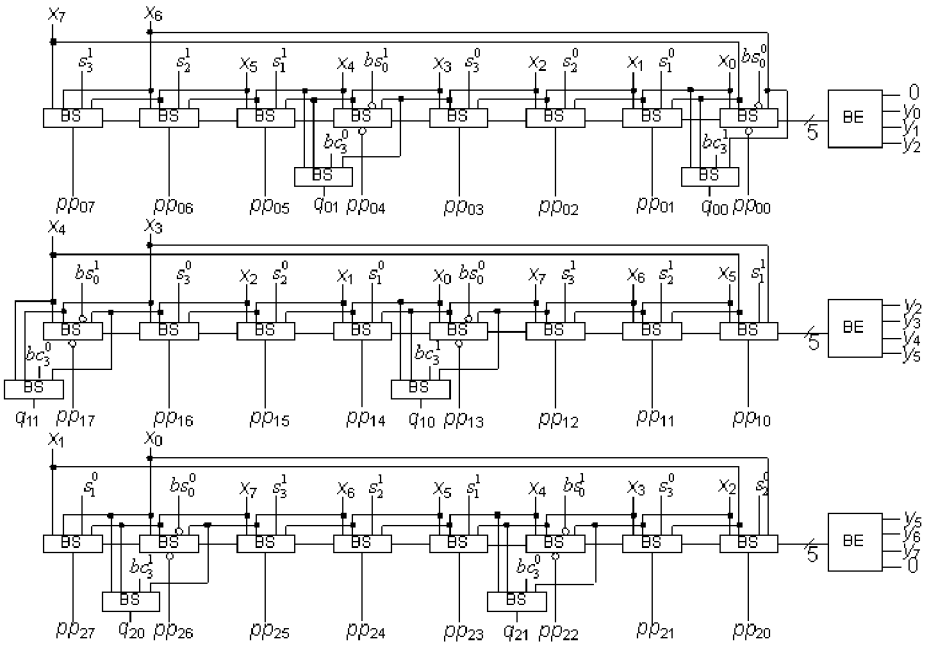

Fig. 6. Modulo-reduced partial product generation.

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

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

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

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

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, $(B + d_i \cdot X)$. For $n = 8$ and $k = 4$, Fig. 5 illustrates the partial product matrix of $|X \cdot Y|_{2^8-1}$ with

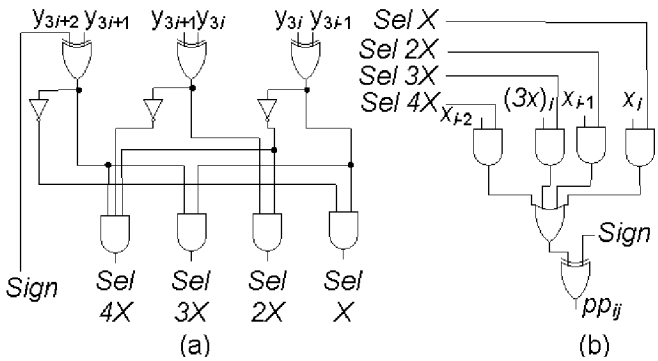

Fig. 7. (a) Bit-slice of Booth Encoder (BE). (b) Bit-slice of Booth Selector (BS).

$(\lfloor n/3 \rfloor + 1)$ partial products in partially-redundant representation. Each PP_i consists of an n-bit vector, $pp_{i7} \ldots pp_{i1} pp_{i0}$ 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, 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. 7(a). The signed one-hot encoded digit is then used to select the correct multiple to generate PP_i. A bit-slice

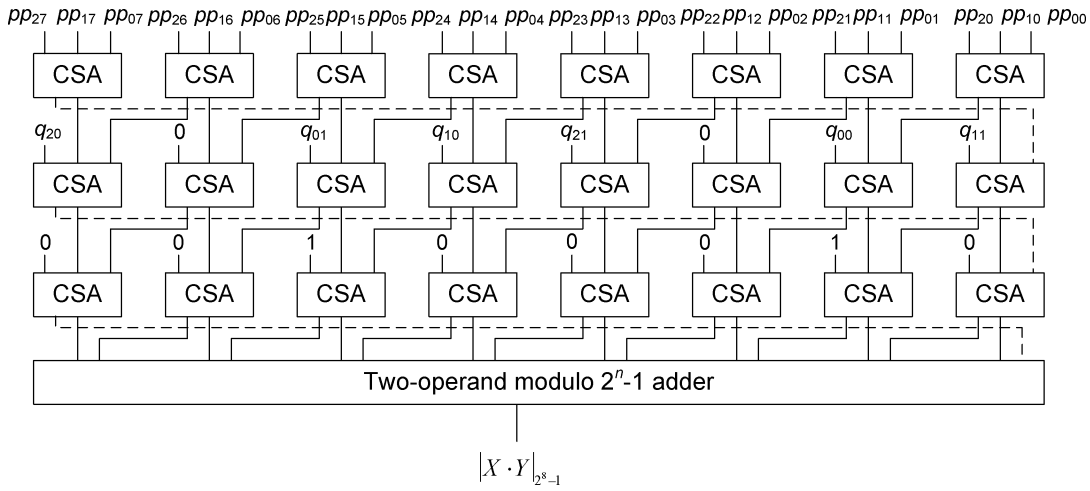

Fig. 8. Modulo-reduced partial product accumulation.

of the radix-8 BS for the partial product bit, pp_{ij} is shown in Fig. 7(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. 8.

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	$O(\log n)$

IV. 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 $2^n - 1$ are represented using only n bits, it is imperative that k divides $n \cdot k = 1$ 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 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, $3 \nmid k$.

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

$$CC = \left| - \sum_{i=0}^{\lfloor n/3 \rfloor} B \cdot 2^{3i} \right|_{2^n - 1} \tag{14}$$

where B is an n-bit binary word consisting of logic one at bit position 2^{kj}, $j \in [0, M-1]$ and logic zero at all other positions as defined in (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 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. In the following section, we show by means of synthesis results how the modulo multiplier delay can be matched to the RNS delay to save silicon area and reduce power dissipation.

V. PERFORMANCE COMPARISON

In this section, we evaluate the performance of the proposed family of partially-redundant modulo $2^n - 1$ multipliers with different suitably chosen RCA word-length, k. The proposed multipliers are also compared against the recent radix-4 Booth encoded modulo $2^n - 1$ multiplier [38].

For experimental analysis, k is selected as $n, n/2$ and $n/4$ to satisfy *Criteria 1* and *2* when n is not divisible by three. When

TABLE III
SYNTHESIS RESULTS WHEN n IS NOT DIVISIBLE BY THREE

n	k					
	n		$n/2$		$n/4$	
	Area (μm^2)	Delay (ns)	Area (μm^2)	Delay (ns)	Area (μm^2)	Delay (ns)
16	16026	6.99	16602	5.63	18208	5.18
20	23477	8.34	24136	6.20	26192	6.13
28	44703	10.50	45624	7.77	48555	6.72
32	56691	11.59	57909	8.38	60730	7.69
40	87730	13.62	89134	10.05	92656	8.23
44	103391	14.81	104851	10.45	109232	8.56
52	144422	16.76	146018	12.29	151148	9.13
56	168768	21.53	172051	13.43	178621	10.63
64	221082	20.15	224821	17.30	231550	10.99

TABLE IV
SYNTHESIS RESULTS WHEN n IS DIVISIBLE BY THREE

n	k			
	$n/3$		$n/6$	
	Area (μm^2)	Delay (ns)	Area (μm^2)	Delay (ns)
12	11459	4.90	12883	4.87
24	37302	6.45	40209	6.33
48	130800	9.78	137410	8.62
60	207194	12.57	212277	9.24

TABLE V
DYNAMIC AND LEAKAGE POWER DISSIPATIONS WHEN n IS NOT DIVISIBLE BY THREE

n	k					
	n		$n/2$		$n/4$	
	Dynamic (mW)	Leakage (μW)	Dynamic (mW)	Leakage (μW)	Dynamic (mW)	Leakage (μW)
16	0.560	0.153	0.569	0.158	0.642	0.176
20	0.832	0.223	0.849	0.229	0.935	0.249
28	1.598	0.415	1.622	0.423	1.744	0.452
32	2.024	0.522	2.053	0.530	2.194	0.560
40	3.160	0.796	3.185	0.808	3.332	0.841
44	3.720	0.929	3.761	0.943	3.939	0.983
52	5.220	1.290	5.255	1.300	5.482	1.350
56	5.732	1.460	5.796	1.480	6.083	1.554
64	6.717	1.950	6.765	1.970	7.026	2.042

TABLE VI
DYNAMIC AND LEAKAGE POWER DISSIPATIONS WHEN n IS DIVISIBLE BY THREE

n	k			
	$n/3$		$n/6$	
	Dynamic (mW)	Leakage (μW)	Dynamic (mW)	Leakage (μW)
12	0.392	0.109	0.443	0.124
24	1.329	0.345	1.450	0.377
48	4.732	1.172	4.998	1.228
60	6.647	1.809	6.899	1.884

n is divisible by three but not by higher powers of three , k is selected as $n/3$ and $n/6$. As M partial-carry bits (q_{ij}) are introduced by each partially-redundant partial product, the number of additional partial products resulting from merging the nonoverlapping q_{ij} bits is given by Q .

$$Q = \left\lceil \frac{\left(\lfloor \frac{n}{3} \rfloor + 1\right) \cdot M}{n} \right\rceil = \left\lceil \frac{\left(\lfloor \frac{n}{3} \rfloor + 1\right)}{k} \right\rceil. \qquad (15)$$

The $(\lfloor n/3 \rfloor + 1)$ partial products along with the merged q_{ij} bits and *CC* are accumulated in an $(\lfloor n/3 \rfloor + 2 + Q)$-operand n-bit CSA tree with end-around-carry addition at each CSA level. The final two-operand modulo $2^n - 1$ adder is implemented as a Sklansky parallel-prefix structure with an additional prefix level for the end-around-carry addition [37].

The proposed modulo $2^n - 1$ multiplier designs for various feasible combinations of n and k were specified in VHDL, synthesized using Synopsys Design Compiler (V2004.06-SP2) and mapped to TSMC 0.18 μm 1.8 V CMOS standard-cell library. The designs were synthesized under nominal synthesis design environment, *i.e.*, 25°C and 1.8 V initially before the timing constraint from the RNS is imposed. The area and delay synthesis results are shown in Table III for n that is not divisible by three and in Table IV for n that is divisible by three.

As the dynamic power dissipation of a combinational circuit is dependent on the input pattern, a Monte Carlo simulation method [49], [50] using a finite number of randomly generated test patterns is adopted to estimate the average power dissipation with 99.9% confidence that the error is bounded below 3% for a data rate of 20 Msamples/s. The average dynamic power and the leakage power are listed in Tables V and VI for n not divisible and divisible by three, respectively.

The synthesis results in Tables III–VI show that, without exerting specific timing optimization effort by the tool, k has an influence on the area and power dissipation of the proposed multiplier and its value can be adapted according to the timing surplus. Among the proposed family of modulo $2^n - 1$ multipliers, the multipliers with k equal to n and $n/3$ have the least area and power dissipation for n that is not divisible by three and for n that is divisible by three, respectively. This implies that for a high-DR RNS with a system delay in excess of the modulo $2^n - 1$ multiplier delay, the multiplier can be designed with the maximum feasible value of k to substantially reduce the area and power dissipation. On the other hand, the multiplier with k equal to $n/4$ and $n/6$ when n is not divisible and divisible by three, respectively has the least critical path delay. This choice of k is suitable for designing the modulo $2^n - 1$ multiplier of an RNS multiplier where the delays of all modulo channels are nearly balanced.

To show the merits of our proposed modulo $2^n - 1$ multiplier, we compare the proposed multiplier and [38] in three special moduli sets, $\{2^n - 1, 2^n, 2^n + 1, 2^{2n} + 1\}$, $\{2^n - 1, 2^n + 1, 2^{2n}, 2^{2n} + 1\}$ and $\{2^n - 1, 2^n, 2^n + 1, 2^{2n+1} - 1\}$, that possess $2^n - 1$ as one of its moduli. The system delays of these RNS are set by the critical modulo $2^{2n} + 1$ and modulo $2^{2n+1} - 1$ multipliers, which were implemented as described in [51] and [38], respectively. As the delay of the critical modulo channel of $\{2^n - 1, 2^n, 2^n + 1, 2^{2n} + 1\}$, $\{2^n - 1, 2^n + 1, 2^{2n}, 2^{2n} + 1\}$ and $\{2^n - 1, 2^n, 2^n + 1, 2^{2n+1} - 1\}$ is in large excess than the delay of $2^n - 1$ modulo channel, the proposed partially-redundant modulo $2^n - 1$ multiplier is designed with the maximum feasible value of $ki.e., n$, when n is not divisible by three and $n/3$ when n is divisible by three. Based on the above $5n$-bit and $6n$-bit DR moduli sets, n needs to be at least 26 and 32 for ECC and RSA with typical key-sizes of 160 and 512 bits, respectively. Hence, modulo $2^n - 1$ multipliers of $n < 26$ will not be considered.

TABLE VII
DELAY-CONSTRAINED AREA AND POWER RESULTS OF MODULO $2^n - 1$ MULTIPLIERS FOR $\{2^n - 1, 2^n, 2^n + 1, 2^{2n} + 1\}$ AND $\{2^n - 1, 2^n + 1, 2^{2n}, 2^{2n} + 1\}$

n	Delay	[38]			Proposed			% Savings in Area Power Product
		Area (μm^2)	Dynamic (mW)	Leakage (μW)	Area (μm^2)	Dynamic (mW)	Leakage (μW)	
28	3.71	59539	16.00	0.43	58953	15.28	0.36	5
32	3.96	78360	20.47	0.56	75585	18.59	0.47	12
40	4.06	122747	29.79	0.87	122019	29.33	0.74	2
44	4.49	143311	32.15	1.04	134150	28.90	0.82	16
48	4.56	174709	38.87	1.25	165681	36.71	1.06	10
52	4.43	200641	43.88	1.44	199507	43.15	1.21	2
56	4.56	232512	56.12	1.58	235199	48.19	1.40	13
60	4.76	267093	56.66	1.93	255061	54.09	1.61	8
64	5.13	304934	69.73	2.20	267099	51.71	1.72	35

TABLE VIII
DELAY-CONSTRAINED AREA AND POWER RESULTS OF MODULO $2^n - 1$ MULTIPLIERS FOR $\{2^n - 1, 2^n, 2^n + 1, 2^{2n+1} - 1\}$

n	Delay	[38]			Proposed			% Savings in Area Power Product
		Area (μm^2)	Dynamic (mW)	Leakage (μW)	Area (μm^2)	Dynamic (mW)	Leakage (μW)	
28	3.42	62885	18.42	0.45	66927	17.66	0.38	–2
32	3.9	78669	20.5	0.57	74015	20.44	0.49	6
40	4.11	120871	31.66	0.89	120019	31.74	0.77	1
44	4.33	143743	33.00	1.03	139000	30.97	0.85	9
48	4.34	174582	38.7	1.25	171465	36.41	1.06	7
52	4.35	202461	45.00	1.46	196340	42.96	1.19	7
56	4.39	232941	56.45	1.58	244533	48.22	1.41	10
60	4.61	267868	57.48	1.96	256628	53.54	1.60	11
64	4.83	302339	62.30	2.18	275881	58.21	1.76	15

The synthesis results for the modulo $2^n - 1$ multipliers constrained by the system delay are shown in Table VII for moduli sets, $\{2^n-1, 2^n, 2^n+1, 2^{2n}+1\}$ and $\{2^n-1, 2^n+1, 2^{2n}, 2^{2n}+1\}$ and in Table VIII for $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$. The percentage saving in area-power-product is also tabulated, where the power includes both dynamic and leakage power dissipations.

From Table VII, the proposed multiplier reduces the area as well as total power dissipation simultaneously. The percentage savings in area-power-product ranges from 2% to 35% for the range of n considered. From Table VIII, savings in area-power-product are observed for $n > 28$ and the savings increase to 15% for $n = 64$. Also, the percentage savings in leakage power dissipation is notably larger than the corresponding percentage savings in dynamic power dissipation. For $n > 28$ under both delay-constrained synthesis, the average savings in leakage power dissipation is around 16%.

To show that the reduction in logic complexity of the proposed design is not due to the mapping of specific cell library, the architectures of [38] and the proposed multipliers are compared theoretically using the normalized area model. In this model, the basic logic modules employed in both multipliers are identified and the area of each logic module is normalized to the area of the inverter for the target cell library. The normalized areas of the logic modules for TSMC 0.18 μm CMOS standard cell library are shown in Table IX [52]. The normalized area expressions of [38] and the proposed multiplier are expressed in Table X and compared in Table XI. The final parallel-prefix adder is excluded from the theoretical evaluation as it is common to both designs. From Table XI, the proposed design reduces the area of [38] for $n > 28$ and the trend is well correlated with the actual synthesis results.

TABLE IX
NORMALIZED AREA OF LOGIC MODULES

Logic module	*Notation*	*Normalized area*
Inverter	INV	1
2-input AND	AND2	2
3-input AND	AND3	2.5
2-input OR	OR2	2
4-input OR	OR4	3
2-input XOR	XOR2	4
2-input XNOR	XNOR2	4
Half Adder	HA	5.5
Full Adder	FA	10.5

TABLE X
NORMALIZED AREA EXPRESSIONS OF MULTIPLIERS

Multiplier	*Component*	*Normalized-area of component*	*Number of component*
[38]	BE	$A_{INV} + A_{AND2} + 2A_{XOR2} = 11$	$n/2$
	BS	$2A_{AND2} + A_{OR2} + A_{XOR2} = 10$	$n(n/2)$
	FA	$A_{FA} = 10.5$	$n(n/2–2)$
	Total	$10.25n^2 - 15.5n$	
Proposed	k-bit RCA	$k-1A_{FA} + A_{HA} + A_{OR2} + A_{XNOR2} = 10.5k + 1$	M
	BE	$3A_{INV} + 3A_{AND2} + A_{AND3} + 3A_{XOR2} = 23.5$	$\lfloor n/3 \rfloor + 1$
	BS	$4A_{AND2} + A_{OR4} + A_{XOR2} = 15$	$(\lfloor n/3 \rfloor + 1) \cdot (n + M)$
	FA	$A_{FA} = 10.5$	$n(\lfloor n/3 \rfloor + Q)$
	Total	$25.5n(\lfloor n/3 \rfloor + 1) + 10.5n + 38.5(\lfloor n/3 \rfloor + 1) + 1$ if $k = n$ $25.5n(\lfloor n/3 \rfloor + 1) + 21n + 68.5(\lfloor n/3 \rfloor + 1) + 3$ if $k = n/3$	

TABLE XI
COMPARISON OF NORMALIZED AREA

n	[38]	Proposed
28	7602	7820
32	10000	9736.5
40	15780	15240
44	19162	17871
48	22872	22984
52	26910	25108
56	31276	28453
60	35970	34832
64	40992	37424

The proposed idea can also be extended to modulo $2^n + 1$ multiplier. Greater overall savings in RNS multiplier area and power dissipation are envisaged without compromising the system speed, provided that modulus $2^n + 1$ is also a noncritical residue channel. However, due to the correction terms associated with the radix-8 Booth encoding, addition and negation in modulo $2^n + 1$ arithmetic [53], [54], there will be some intricate hardware modifications and tradeoff, which will be addressed in our future work.

VI. 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, *i.e.*, area, delay and power dissipation of the modulo $2^n - 1$ multiplier were analyzed by means of CMOS implementations. For maximal area and power savings, n when n is not divisible by three and $n/3$ when n is divisible by three, were recommended for the RCA word-length when the RNS multiplier delay exceeded the noncritical modulo $2^n - 1$ multiplier delay substantially. Conversely, when the RNS multiplier and the modulo $2^n - 1$ multiplier delays were nearly balanced, RCA word-lengths of $n/4$ and $n/6$ were recommended when n is not divisible and divisible by three, respectively. From synthesis results constrained by the critical channel delay of the RNS, it was shown that 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.

APPENDIX
DERIVATION OF COMPENSATION CONSTANT

The effect of the bias, B is counteracted by adding a compensation, *CC* in the partial product matrix. This appendix proves that *CC* of (14) is a constant binary word for different choices of k stipulated by *Criteria 1* and *2* in Section IV. Due to *Criterion 2*, the proof is constructed separately for $3 \nmid n$ and $3|n$.

A: n is Not Divisible by Three, I.e., $3 \nmid n$. Since n is not divisible by three, k can be chosen as any integer in the set $[n, n/2, n/4, n/5, \ldots, 2]$.

A.1: If $k = n$, then $M = n/k = 1$ and $B = 1$ from (7). By substituting $B = 1$ in (14), we have

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

Since $3 \nmid n$, $mod(n, 3) = 1$ or 2.

Case 1: $mod(n, 3) = 1$. The upper limit of the summation of (16) is replaced by $(n-1)/3$. We have

$$CC = \left| - \sum_{i=0}^{(n-1)/3} 2^{3i} \right|_{2^n - 1}. \tag{17}$$

By *Property 1*, (17) can be simplified to an n–bit one's complementation of $\sum_{i=0}^{(n-1)/3} 2^{3i}$. Therefore

$$CC = \sum_{i=0}^{(n-4)/3} \left(2^{3i+1} + 2^{3i+2}\right). \tag{18}$$

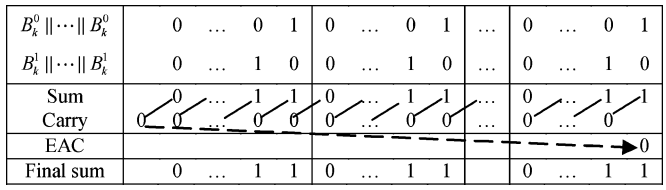

Fig. 9. Modulo $2^n - 1$ addition of $B_k^0 \| B_k^0 \| \cdots \| B_k^0$ and $B_k^1 \| B_k^1 \| \cdots \| B_k^1$.

Case 2: $mod(n, 3) = 2$. The upper limit of the summation of (16) is replaced by $(n-2)/3$. We have

$$CC = \left| - \sum_{i=0}^{(n-2)/3} 2^{3i} \right|_{2^n - 1} = \sum_{i=0}^{(n-2)/3} 2^{3i+1} + \sum_{i=0}^{(n-5)/3} 2^{3i+2}. \tag{19}$$

In both cases, *CC* is an n-bit binary word consisting of logic one at bit positions $3i+1$ and $3i+2$ and logic zero at other bit positions.

A.2: From (7), for any other legitimate value of k

$$B = \underbrace{\underbrace{0 \cdots 01}_{k} \| \underbrace{0 \cdots 01}_{k} \| \cdots \| \underbrace{0 \cdots 01}_{k}}_{n} = B_k^0 \| B_k^0 \| \cdots \| B_k^0 \tag{20}$$

where $\|$ is the concatenation operator and B_k^0 is a k-bit substring with only its lsb set to logic one. By substituting B into (14) and applying *Property 3*, we have

$$CC = \left| - \sum_{i=0}^{\lfloor n/3 \rfloor} CLS\left(B_k^0 \| B_k^0 \| \cdots \| B_k^0, 3i\right) \right|_{2^n - 1}. \tag{21}$$

The following circular left shift property holds for B.

Property 4:

$$\begin{aligned} &CLS\,(B, j) \\ &\quad = CLS\left(B_k^0 \| B_k^0 \| \cdots \| B_k^0, j\right) \\ &\quad = CLS\left(B_k^0, j\right) \| CLS\left(B_k^0, j\right) \| \cdots \| CLS\left(B_k^0, j\right) \\ &\quad = B_k^j \| B_k^j \| \cdots \| B_k^j \end{aligned} \tag{22}$$

where B_k^j is a k-bit substring with a logic one at the $(j \bmod k)$-th bit position.

Simplifying the CLS operation in (21) by *Property 4*

$$CC = \left| - \sum_{i=0}^{\lfloor n/3 \rfloor} B_k^{3i} \parallel B_k^{3i} \parallel \cdots \parallel B_k^{3i} \right|_{2^n - 1}. \tag{23}$$

Fig. 9 illustrates the two-operand modulo $2^n - 1$ addition of n-bit binary substrings, each composed of identical substrings of the form $B_k^j \| B_k^j \| \cdots \| B_k^j$. In Fig. 9, $B_k^0 \| \cdots \| B_k^0$ and $B_k^1 \| \cdots \| B_k^1$ are considered as the addends.

From Fig. 9, the substring at the output, $0 \ldots 11$ can be computed by a k-bit end-around-carry addition, *i.e.*, modulo $2^k - 1$ addition of B_k^0 and B_k^1. By extending the above observation to multi-operand addition, *Property 5* shows that modulo $2^n - 1$

summation of binary strings, each composed of identical substrings, is equivalent to modulo $2^k - 1$ summation of the substrings followed by concatenation.

Property 5:

$$\begin{aligned}&\left|\left(B_k^0 \parallel B_k^0 \parallel \cdots \parallel B_k^0\right) + \left(B_k^1 \parallel B_k^1 \parallel \cdots \parallel B_k^1\right) + \right.\\ &\left. \cdots + \left(B_k^j \parallel B_k^j \parallel \cdots \parallel B_k^j\right)\right|_{2^n-1}\\ &= \left|B_k^0 + B_k^1 + \cdots + B_k^j\right|_{2^k-1} \parallel \left|B_k^0 + B_k^1 + \cdots + B_k^j\right|_{2^k-1} \parallel\\ &\quad \cdots \parallel \left|B_k^0 + B_k^1 + \cdots + B_k^j\right|_{2^k-1}. \end{aligned} \tag{24}$$

Simplifying the modulo summation operation in (23) using *Property 5*, *CC* is given by

$$CC = \left|-\sum_{i=0}^{\lfloor n/3\rfloor} B_k^{3i}\right|_{2^k-1} \parallel \cdots \parallel \left|-\sum_{i=0}^{\lfloor n/3\rfloor} B_k^{3i}\right|_{2^k-1}. \tag{25}$$

For ease of exposition, *CC* is derived for $k = n/2$ and *Properties 4 and 5* can be applied to the remaining cases of k to show that *CC* is indeed a constant binary word. From (25), for $k = n/2$, *CC* is given by

$$CC = CC_{n/2} \parallel CC_{n/2} \tag{26}$$

where

$$\begin{aligned} CC_{n/2} &= \left|-\sum_{i=0}^{\lfloor n/3\rfloor} B_{n/2}^{3i}\right|_{2^{n/2}-1}\\ &= \left|-\sum_{i=0}^{\lfloor n/3\rfloor} CLS\left(\underbrace{0\ldots01}_{n/2}, 3i\right)\right|_{2^{n/2}-1}\\ &= \left|-\sum_{i=0}^{\lfloor n/3\rfloor} 2^{3i}\right|_{2^{n/2}-1}. \end{aligned} \tag{27}$$

The binary exponents greater than $n/2$ in (27) can be modulo reduced by *Property 2* to

$$CC_{n/2} = \left|-\sum_{i=0}^{\lfloor k/3\rfloor} 2^{3i} - \sum_{i=\lfloor k/3\rfloor+1}^{\lfloor n/3\rfloor} 2^{3i-k}\right|_{2^{n/2}-1} \tag{28}$$

where $k = n/2$ and $mod(k, 3) = 1$ or 2 since $3 \nmid k$.

Case 1: $mod(k, 3) = 1$. By substituting $\lfloor k/3\rfloor = (k-1)/3$ in the summation bounds of (28),

$$CC_{n/2} = \left|-\sum_{i=0}^{(k-1)/3} 2^{3i} - \sum_{i=(k+2)/3}^{(2k-2)/3} 2^{3i-k}\right|_{2^{n/2}-1}. \tag{29}$$

By adjusting the indices of the second summation

$$\begin{aligned} CC_{n/2} &= \left|-\sum_{i=0}^{(k-1)/3} 2^{3i} - \sum_{i=0}^{(k-1)/3-1} 2^{3i+2}\right|_{2^{n/2}-1}\\ &= \sum_{i=0}^{(k-1)/3-1} 2^{3i+1}. \end{aligned} \tag{30}$$

Case 2: $mod(k, 3) = 2$. By substituting $\lfloor k/3\rfloor = (k-2)/3$ in the summation bounds of (28),

$$\begin{aligned} CC_{n/2} &= \left|-\sum_{i=0}^{(k-2)/3} 2^{3i} - \sum_{i=(k+1)/3}^{(2k-1)/3} 2^{3i-k}\right|_{2^{n/2}-1}\\ &= \left|-\sum_{i=0}^{(k-2)/3} 2^{3i} - \sum_{i=0}^{(k-2)/3} 2^{3i+1}\right|_{2^{n/2}-1} = \sum_{i=0}^{(k-2)/3-1} 2^{3i+2}. \end{aligned} \tag{31}$$

From the final expressions of (30) and (31), $CC_{n/2}$ for *Cases 1* and *2* is an $n/2$-bit binary word consisting of logic one at bit positions $3i + 1$ and $3i + 2$, respectively, and logic zero elsewhere.

The binary value of *CC* for the various combinations of n and k can been precomputed using similar derivation when n is not a multiple of three. Table XII summarizes *CC* for $k = n$, $n/2$ and $n/4$. For $k = n/2$ and $n/4$, the substrings $CC_{n/2}$ and $CC_{n/4}$ are listed and *CC* can be obtained by $CC_{n/2}||CC_{n/2}$ and $CC_{n/4}||CC_{n/4}||CC_{n/4}||CC_{n/4}$, respectively. In Table XII, $\underbrace{\#\#\#}_{\times i}$ denote that the string ### is repeated i times.

B: n is Divisible by Three, I.e., $3|n$. When n is divisible by three, the choice of k is limited to the integers in the set $[n/3, n/6, n/9, \ldots, 2]$ that are not divisible by three according to *Criterion 2*. The derivation of *CC* for $k = n/3$ is illustrated here and *CC* for all other cases of k can be similarly derived.

From (25), for $k = n/3$, we have

$$CC = CC_{n/3} \parallel CC_{n/3} \parallel CC_{n/3} \tag{32}$$

where

$$\begin{aligned} CC_{n/3} &= \left|-\sum_{i=0}^{\lfloor n/3\rfloor} B_{n/3}^{3i}\right|_{2^{n/3}-1}\\ &= \left|-\sum_{i=0}^{\lfloor n/3\rfloor} CLS\left(\underbrace{0\ldots01}_{n/3}, 3i\right)\right|_{2^{n/3}-1}\\ &= \left|-\sum_{i=0}^{\lfloor n/3\rfloor} 2^{3i}\right|_{2^{n/3}-1}. \end{aligned} \tag{33}$$

TABLE XII
COMPENSATION CONSTANT WHEN n IS NOT DIVISIBLE BY THREE

n	CC ($k=n$)	$CC_{n/2}$ ($k=n/2$)	$CC_{n/4}$ ($k=n/4$)
8	$10\underbrace{110}_{\times 2}$	$0\underbrace{010}_{\times 1}$	10
16	$0\underbrace{110}_{\times 5}$	$00\underbrace{100}_{\times 2}$	$0\underbrace{110}_{\times 1}$
20	$10\underbrace{110}_{\times 6}$	$0\underbrace{010}_{\times 3}$	$10\underbrace{110}_{\times 1}$
28	$0\underbrace{110}_{\times 9}$	$00\underbrace{100}_{\times 4}$	$0\underbrace{110}_{\times 2}$
32	$10\underbrace{110}_{\times 10}$	$0\underbrace{010}_{\times 5}$	$10\underbrace{110}_{\times 2}$
40	$0\underbrace{110}_{\times 13}$	$00\underbrace{100}_{\times 6}$	$0\underbrace{110}_{\times 3}$
44	$10\underbrace{110}_{\times 14}$	$0\underbrace{010}_{\times 7}$	$10\underbrace{110}_{\times 3}$
52	$0\underbrace{110}_{\times 17}$	$00\underbrace{100}_{\times 8}$	$0\underbrace{110}_{\times 4}$
56	$10\underbrace{110}_{\times 18}$	$0\underbrace{010}_{\times 9}$	$10\underbrace{110}_{\times 4}$
64	$0\underbrace{110}_{\times 21}$	$00\underbrace{100}_{\times 10}$	$0\underbrace{110}_{\times 5}$

The binary exponents greater than $n/3$ of (33) are modulo reduced by *Property 2* to

$$CC_{n/3} = \Bigg| - \sum_{i=0}^{\lfloor k/3 \rfloor} 2^{3i} - \sum_{i=\lfloor k/3 \rfloor+1}^{\lfloor 2k/3 \rfloor} 2^{3i-k} - \sum_{i=\lfloor 2k/3 \rfloor+1}^{k-1} 2^{3i-2k} - \sum_{i=k}^{k} 2^{3i-3k} \Bigg|_{2^{n/3}-1} \tag{34}$$

where $k = n/3$ and $mod(k,3) = 1$ or 2 since $3 \nmid k$.

Case 1: $mod(k,3) = 1$. By substituting $\lfloor k/3 \rfloor = (k-1)/3$ in the summation bounds of (34)

$$\begin{aligned} CC_{n/3} &= \left| - \sum_{i=0}^{(k-1)/3} 2^{3i} - \sum_{i=(k+2)/3}^{(2k-2)/3} 2^{3i-k} - \sum_{i=(2k+1)/3}^{k-1} 2^{3i-2k} - 1 \right|_{2^{n/3}-1} \\ &= \left| - \sum_{i=0}^{(k-1)/3} 2^{3i} - \sum_{i=0}^{(k-1)/3-1} 2^{3i+2} - \sum_{i=0}^{(k-1)/3-1} 2^{3i+1} - 1 \right|_{2^{n/3}-1} \\ &= \sum_{i=1}^{(k-1)/3} 2^{3i} + \sum_{i=0}^{(k-1)/3-1} 2^{3i+1} + \sum_{i=0}^{(k-1)/3-1} 2^{3i+2}. \end{aligned} \tag{35}$$

Case 2: $mod(k,3) = 2$. By substituting $\lfloor k/3 \rfloor = (k-2)/3$ in the summation bounds of (34)

$$\begin{aligned} CC_{n/3} &= \left| - \sum_{i=0}^{(k-2)/3} 2^{3i} - \sum_{i=(k+1)/3}^{(2k-1)/3} 2^{3i-k} - \sum_{i=2(k+1)/3}^{k-1} 2^{3i-2k} - 1 \right|_{2^{n/3}-1} \\ &= \sum_{i=1}^{(k-2)/3} 2^{3i} + \sum_{i=0}^{(k-2)/3} 2^{3i+1} + \sum_{i=0}^{(k-2)/3-1} 2^{3i+2}. \end{aligned} \tag{36}$$

TABLE XIII
COMPENSATION CONSTANT WHEN n IS DIVISIBLE BY THREE

n	$CC_{n/3}$ ($k=n/3$)	$CC_{n/6}$ ($k=n/6$)
12	1110	10
24	$1\underbrace{1111110}_{\times 1}$	1110
48	$1\underbrace{111110}_{\times 4}$	$1\underbrace{1111110}_{\times 1}$
60	$1\underbrace{1111110}_{\times 5}$	$1\underbrace{111110}_{\times 2}$

In both cases, $CC_{n/3}$ is an $n/3$-bit binary word that is set to logic one in all but the lsb position.

The binary value of *CC* for combinations of n and k when n is a multiple of three are reported in Table XIII. For k chosen as $n/3$ and $n/6$, *CC* is given by $CC_{n/3}||CC_{n/3}||CC_{n/3}$ and $CC_{n/6}||CC_{n/6}||CC_{n/6}||CC_{n/6}||CC_{n/6}||CC_{n/6}$, respectively.

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," *J. Cryptol.*, vol. 14, no. 4, pp. 255–293, Aug. 2001.

[6] C. McIvor, M. McLoone, and J. V. McCanny, "Modified Montgomery modular multiplication and RSA exponentiation techniques," *IEE Proc. Comput. and Dig. Techniq.*, vol. 151, no. 6, pp. 402–408, Nov. 2004.

[7] C. McIvor, M. McLoone, and J. V. McCanny, "Hardware elliptic curve cryptographic processors over $GF(p)$," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 9, pp. 1946–1957, Sep. 2006.

[8] D. M. Schinianakis, A. P. Fournaris, H. E. Michail, A. P. Kakarountas, and T. Stouraitis, "An RNS implementation of an Fp Elliptic curve point multiplier," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 6, pp. 1202–1213, Jun. 2009.

[9] J. C. Bajard and L. Imbert, "A full RNS implementation of RSA," *IEEE Trans. Comput. – Brief Contributions*, vol. 53, no. 6, pp. 769–774, Jun. 2004.

[10] 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.

[11] 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.

[12] 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.

[13] I. Steiner *et al.*, "A fault-tolerant modulus replication complex FIR filter," in *Proc. 16th IEEE Int. Conf. Application-Specific Systems, Architecture and Processors*, Samos, Greece, Jul. 2005, pp. 387–392.

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

[15] 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.

[16] P. V. A. Mohan, *Residue Number Systems: Algorithms and Architectures*. Norwell, MA: Kluwer, 2002.

[17] 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.

[18] G. Dimitrakopoulos and V. Paliouras, "A novel architecture and a systematic graph-based optimization methodology for modulo multiplication," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 2, pp. 354–370, Feb. 2004.

[19] A. A. Hiasat, "New efficient structure for a modular multiplier for RNS," *IEEE Trans. Comput.*, vol. 49, no. 2, pp. 170–174, Feb. 2000.
[20] A. A. Hiasat and H. S. Abdel-Aty-Zohdy, "Residue-to-binary arithmetic converter for the moduli set $\{2^k, 2^k - 1, 2^{k-1} - 1, \}$," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 45, no. 2, pp. 204–209, Feb. 1998.
[21] B. Cao, C. H. Chang, and T. Srikanthan, "An efficient reverse converter for the 4-moduli set $\{2^n - 1, 2^n, 2^n + 1, 2^{2n} + 1\}$ based on the New Chinese Remainder Theorem," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 50, no. 10, pp. 1296–1303, Oct. 2003.
[22] B. Cao, T. Srikanthan, and C. H. Chang, "Efficient reverse converters for four-moduli sets $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$ and $\{2^n - 1, 2^n, 2^n + 1, 2^{n-1} - 1\}$," *IEE Proc. Comput., Dig. Techniq.*, vol. 152, no. 5, pp. 687–696, Sep. 2005.
[23] B. Cao, C. H. Chang, and T. Srikanthan, "A residue-to-binary converter for a new five-moduli set," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 5, pp. 1041–1049, May 2007.
[24] P. V. A. Mohan and A. B. Premkumar, "RNS-to-binary converters for two four-moduli sets $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$ and $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} + 1\}$," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 6, pp. 1245–1254, Jun. 2007.
[25] P. V. A. Mohan, "RNS-to-binary converter for a new three moduli set $\{2^{n+1} - 1, 2^n, 2^n - 1\}$," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 54, no. 9, pp. 775–779, Sep. 2007.
[26] T. Tomczak, "Fast sign detection for RNS $\{2^n - 1, 2^n, 2^n + 1\}$," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 6, pp. 1502–1511, Jul. 2008.
[27] 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.
[28] D. J. Soudris, V. Paliouras, T. Stouraitis, and C. E. Goutis, "A VLSI 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.
[29] B. Cao, Y. S. Low, C. H. Chang, and T. Srikanthan, "Performance analysis of different special moduli sets for RNS-based inner product step processor," in *Proc. 2010 Int. Conf. Green Circuits and Systems*, Shanghai, China, Jun. 2010, pp. 236–241.
[30] P. V. A. Mohan, "Reverse converters for a new moduli set $\{2^{2n} - 1, 2^n, 2^{2n} + 1\}$," *Circuits, Syst. Signal Process.*, vol. 26, no. 2, pp. 215–227, Apr. 2007.
[31] A. S. Molahosseini, K. Navi, C. Dadkhah, O. Kavehei, and S. Timarchi, "Efficient reverse converter designs for the new 4-moduli sets $\{2^n - 1, 2^n, 2^n + 1, 2^{2n+1} - 1\}$ and $\{2^n - 1, 2^n + 1, 2^{2n}, 2^{2n} + 1\}$ based on New CRTs," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 4, pp. 823–835, Apr. 2010.
[32] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, "Security in embedded systems: Design challenges," *ACM Trans. Embedded Comput. Syst.*, vol. 3, no. 3, pp. 461–491, Aug. 2004.
[33] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits*. Upper Saddle River, NJ: Prentice-Hall, 2003.
[34] M. Hamada, Y. Ootaguro, and T. Kuroda, "Utilizing surplus timing for power reduction," in *Proc. IEEE Conf. Custom Integrated Circuits*, San Diego, CA, May 2001, pp. 89–92.
[35] G. C. Cardarilli, A. D. Re, A. Nannarelli, and M. Re, "Low power and low leakage implementation of RNS FIR filters," in *Proc. 39th Asilomar Conf. Signals, Systems and Computers*, Pacific Grove, CA, Nov. 2005, pp. 1620–1624.
[36] Z. Wang, G. A. Jullien, and W. C. Miller, "An algorithm for multiplication modulo $(2^n - 1)$," in *Proc. 39th IEEE Midwest Symp. Circuits and Systems*, Ames, IA, Aug. 1996, pp. 1301–1304.
[37] R. Zimmermann, "Efficient VLSI implementation of modulo $(2^n \pm 1)$ addition and multiplication," in *Proc. 14th IEEE Symp. Computer Arithmetic*, Adelaide, Australia, Apr. 1999, pp. 158–167.
[38] 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.
[39] B. S. Cherkauer and E. G. Friedman, "A hybrid radix-4/radix-8 low power signed multiplier architecture," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 44, no. 8, pp. 656–659, Aug. 1997.
[40] M. J. Flynn and S. F. Oberman, *Advanced Computer Arithmetic Design*. New York: Wiley, 2001.
[41] S. J. Piestrak, "Design of squarers modulo A with low-level pipelining," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 49, no. 1, pp. 31–41, Jan. 2002.
[42] C. Efstathiou, D. Nikolos, and J. Kalamatianos, "Area-time efficient modulo $2^n - 1$ adder design," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 41, no. 7, pp. 463–467, Jul. 1994.
[43] 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.
[44] G. Dimitrakopoulos, D. G. Nikolos, H. T. Vergos, D. Nikolos, and C. Efstathiou, "New architectures for modulo $2^n - 1$ adders," in *Proc. 12th IEEE Int. Conf. Electronics, Circuits and Systems*, Gammarth, Tunisia, Dec. 2005, pp. 1–4.
[45] R. A. Patel, M. Benaissa, and S. Boussakta, "Fast Parallel-prefix architectures for modulo $2^n - 1$ addition with a single representation of zero," *IEEE Trans. Comput.*, vol. 56, no. 11, pp. 1484–1492, Nov. 2007.
[46] 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," in *Proc. 2010 IEEE Int. Symp. Circuits and Systems*, Paris, France, Jun. 2010, pp. 717–720.
[47] C. Nagendra, M. J. Irwin, and R. M. Owens, "Area-time-power tradeoffs in parallel adders," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 43, no. 10, pp. 689–702, Oct. 1996.
[48] G. W. Bewick, "Fast multiplication: Algorithms and implementation," Ph.D. dissertation, Stanford Univ., Stanford, CA, 1994.
[49] R. Burch, F. N. Najm, P. Yang, and T. N. Trick, "A Monte Carlo approach for power estimation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 1, no. 1, pp. 63–71, Mar. 1993.
[50] R. K. Satzoda, C. H. Chang, and T. Srikanthan, "Monte Carlo statistical analysis for dynamic power simulation of RTL designs using Synopsys Power Compiler," in *Synopsys Users' Group Conf.*, Singapore, Jun. 2006.
[51] L. Sousa and R. Chaves, "A universal architecture for designing efficient modulo $2^n + 1$ multipliers," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 6, pp. 1166–1178, Jun. 2005.
[52] TSMC 0.18 μm Process 1.8 Volt Sage-X™ Standard Cell Library Databook Artisan Components Inc., 2003.
[53] H. T. Vergos and C. Efstathiou, "A unifying approach for weighted and diminished-1 modulo $2^n + 1$ addition," *IEEE Trans. Circuits and Syst. II, Exp. Briefs*, vol. 55, no. 10, pp. 1041–1045, Oct. 2008.
[54] T.-B. Juang, C.-C. Chiu, and M.-Y. Tsai, "Improved area-efficient weighted modulo $2^n + 1$ adders design with simple correction schemes," *IEEE Trans. Circuits and Syst. II, Exp. Briefs*, vol. 57, no. 3, pp. 198–202, Mar. 2010.

Ramya Muralidharan (S'07) received the B.E degree in electrical and electronics engineering from Anna University, Chennai, India, in 2006. She is currently pursuing the Ph.D. degree at the Divison of Circuits and Systems, School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.

She is a member of the Research Staff at the Divison of Circuits and Systems, where her research interests include computer arithmetic circuits and digital IC design.

Chip-Hong Chang (S'92–M'98–SM'03) received the B.Eng. (Hons.) from the National University of Singapore in 1989 and the M.Eng. and Ph.D. degrees from Nanyang Technological University (NTU), Singapore, in 1993 and 1998, respectively.

He served as a Technical Consultant in industry prior to joining the School of Electrical and Electronic Engineering (EEE), NTU, in 1999, where he is now an Associate Professor. He holds joint appointments at the university as Assistant Chair of Alumni, School of EEE since June 2008, Deputy Director of the Centre for High Performance Embedded Systems (CHiPES) since 2000, and Program Director of the Centre for Integrated Circuits and Systems (CICS) from 2003 to 2009. His current research interests include low-power arithmetic circuits, digital filter design, application-specific digital signal processing, and digital watermarking for IP protection. He has published three book chapters and more than 150 research papers in refereed international journals and conferences.

Dr. Chang has served as the Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I since 2010, as an Editorial Advisory Board Member of the *Open Electrical and Electronic Engineering Journal* since 2007 and the *Journal of Electrical and Computer Engineering* since 2008, the Special Issue Guest Editor of the *Journal of Circuits, Systems and Computers* in 2010, and in several international conference advisory and technical program committees. He is a Fellow of the IET.