Low-Power Radix-8 Booth Encoded Modulo $2^n - 1$ Multiplier with Adaptive Delay

V. R. Prakash, Himanshu Shekhar, T. Shanmuganathan and P. Kumaraguru

Department of ECE, Faculty of Hindustan University, Padur, Chennai, India.
e-mail: vrprakash@hindustanuniv.ac.in

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 analysing 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 pre-computed 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.

Keywords: Booth algorithm, Design space exploration, Modulo arithmetic, Multiplier, Residue number system (RNS).

1. Introduction

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) is represented as an $N$-tuple $\{X_1, X_2, \ldots, X_N\}$, where $x_i$ is the residue of $X$ modulo $L_i$. RNS multipliers based on generic moduli have been reported in [3,4]. 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 [5,6]. The most popular of these special moduli sets is the triple moduli set, $\{2^n, 2^n + 1, 2^n - 1\}$ which however has a DR of only 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 [7,8]. 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 multipliers 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.

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 [9,10] 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, 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 [11–13]. In [11], the multiplier bits were not encoded, which lead to higher implementation area and longer partial product accumulation time. In [12] and [13], the radix-4 Booth encoding algorithm was employed to reduce the number of partial products to $\frac{1}{2}n/2^3 + 1$ and $\cdot n/2^3$, 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 [16,14]. The radix-8 Booth encoding reduces the number of partial products to $\frac{1}{2}n/3^3 + 1$, 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 $| + 3X |_{2n-1}$ 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 analysed 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 pre-computed to equal the delay of the critical modulo multiplier of the RNS. By means of modulo 2 arithmetic, 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 3. The performance of the proposed family of modulo $2^n$1 multipliers is evaluated and compared against [13] in Section 5. The paper is concluded in Section 6.

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

Let $X = \sum_{i=0}^{n-1} x_i 2^i$ and $Y = \sum_{i=0}^{n-1} y_i 2^i$ present 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, $y_{3i} + 2y_{3i} + y_{3i} + 1$ to a signed digit, $d_i, d_i \in \{-4, 4\},$ for $i = 0, 1, \ldots, \frac{n}{3^3}. The digit set conversion is formally expressed as

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

where $y_{n} = y_{n+1} = y_{n+2} = 0.$

Table 1 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:

$$| - X | = 2^n - 1 - X = X \quad (2)$$

2) Property 2: For any nonnegative integer, the periodicity of an integer power of two over modulus $2^n - 1$ can be stated as follows [17]:

$$| 2^n x + 1 |_{2^n-1} = | 2^n x |_{2^n-1} \cdot | 2^i |_{2^n-1} = | 2^i |_{2^n-1} \quad (3)$$
Table 1. Modulo-reduced multiples for the radix-8 Booth encoding.

| $d_i$ | $|d_i \cdot X|_{2^i-1}$ |
|---|---|
| +0 | $0\cdots0$ |
| +1 | $X$ |
| +2 | CLS($X$, 1) |
| +3 | $+3X|_{2^i-1}$ |
| +4 | CLS($X$, 2) |

| $d_i$ | $|d_i \cdot X|_{2^i-1}$ |
|---|---|
| -0 | $1\cdots1$ |
| -1 | $\bar{X}$ |
| -2 | CLS($\bar{X}$, 1) |
| -3 | $-3X|_{2^i-1}$ |
| -4 | CLS($\bar{X}$, 2) |

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 and an integer power of two, $2^j$, is equivalent to [1]. This property can be formally expressed as Property 3.

3) Property 3: For $j < n$

$$|2^j \cdot X|_{2^2-1} = \sum x_i \cdot 2^{i+j} + \sum x_i \cdot 2^{i+j} = \text{CLS} (X, j) \quad (4)$$

Figure 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 figure 1. The above technique for $| + 3X|_{2^n-1}$ computation involves two-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.

3. 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 figure 1 can be eliminated by making the end-around-carry bit $c7$ 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.
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 [15].

A. Generation of partially-redundant hard multiple

Let \(|X|_{2n-1}\) and \(|2X|_{2n-1}\) be added by a group of \(M(= n/k)\) \(k\)-bit RCAs such that there is no carry propagation between the adders. Figure 2 shows this addition for \(n = 8\) and \(k = 4\), where the sum and carry-out bits from the RCA block are represented as \(s_j^i\) and \(c_j^i\) for \(i \in [0, k-1]\) and \(j \in [0, M-1]\), respectively.

From figure 2, the partially-redundant form of \(|+3X|_{2n-1}\) given by the partial-sum and partial-carry pair \((S, C)\)

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

\[
C = 0 \cdots 0 c_{k-1}^{M-2} \cdots 0 c_{k-1}^0 \cdots 0 c_{k-1}^{M-1}
\]

(5)

Its complement will be.

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

\[
\bar{C} = 1 \cdots 1 \bar{c}_{k-1}^{M-2} \cdots 1 \bar{c}_{k-1}^0 \cdots 1 \bar{c}_{k-1}^{M-1}
\]

(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 its complement are sparse [15]. The value of is chosen as

\[
B = \sum_{j=0}^{M-1} 2^{k-j} = \underbrace{0\cdots 01}_{k}
\]

\[
BS = s_{k-1}^{M-1} s_{k-2}^{M-1} \cdots b_{k-1}^0 s_{k-2}^0 \cdots b_{k-1}^0
\]

\[
BC = 0 \cdots 0 b_{k-1}^{M-2} \cdots 0 b_{k-1}^0 \cdots 0 b_{k-1}^{M-1}
\]

(7)

\[\text{Figure 2. Generally of partially-redundant } |+3X|_{2n-1} \text{ using } k\text{-bit RCAs.}\]

\[\text{Figure 3. Generally of partially-redundant } |B + 3X|_{2n-1}.\]
where
\[ b_{x0}^j = \begin{cases} 
  s_0^j \oplus c_{k-1}^j & \text{when } j \neq 0 \\
  s_0^0 \oplus c_{k-1}^0 & \text{when } j = 0
\end{cases} \]
and
\[ bc_{k-1}^j = \begin{cases} 
  s_0^j + c_{k-1}^j & \text{when } j \neq M - 1 \\
  s_0^0 + c_{k-1}^0 & \text{when } j = M - 1
\end{cases} \]

3.1 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 1 must also be biased and generated in a partially-redundant form. Figure 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 figure 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.

![Figure 4. Generation of partially-redundant simple multiples.](image)

![Figure 5. Modulo-reduced partial products and CC for \(|X \cdot Y|2^{8-1}\).](image)

![Figure 6. Modulo-reduced partial product generation.](image)
V. R. Prakash, et al.

C. Radix-8 booth encoded modulo $2^n - 1$ multiplication with partially-redundant partial products

Figure 5 illustrates the partial product matrix of with $(n+3_i+1)$ partial products in partially-redundant representation. Each $PP_i$ consists of an $n$-bit vector, $pp_i7, pp_i6, \ldots, pp_i0$ and a vector of $n/k = 2$ redundant carry bits, $q_{i1}$ and $q_{i0}$. Since $q_{i1}$ and $q_{i0}$ are the carry-out bits for a given $PP_i$, 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 $q_{00}$ and $q_{01}$ by 3 bits, respectively. The last partial product in figure 5 is the Compensation Constant (CC) for the bias introduced in the partially-redundant representation.

The generation of the modulo-reduced partial products, $PP$, $PP_1$, and $PP_3$, in a partially-redundant representation using Booth Encoder (BE) and Booth Selector (BS) blocks are illustrated in figure 6. The BE block produces a signed one-hot encoded digit from adjacent overlapping multiplier bits as illustrated in figure 7(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 figure 7(b). As the bit positions of $q_{ij}$ do not overlap, as shown in figure 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 figure 8.

In the proposed modulo $2^n - 1$ multiplier, each partial product $PP_i$ is incremented by a bias $2^j_i \times B$.

To negate the effect of the bias, a constant CC is added and the value of $CC$ is given by

$$CC = \left| - \sum_{j=0}^{\lceil \frac{n}{3} \rceil} B \cdot 2^j \right|_{2n-1}$$

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 (8).

![Figure 7.](image-url) (a) Bit-slice of booth encoder (BE). (b) Bit-slice of booth selector (BS).

![Figure 8.](image-url) Modulo-reduced partial product accumulation.
4. 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-lengths, $k$. The proposed multipliers are also compared against the recent radix-4 Booth encoded modulo multiplier. For experimental analysis, $n$ is selected as $n$, $n/2$ and $n/4$ when $n$ is not divisible by three.

When $n$ is divisible by three but not by higher powers of three, $k$ is selected as $n/3$ and $n/6$.

5. 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 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 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.

References