International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-3 Issue-2, July 2013

Design and Modeling of Modulo Multipliers Using RNS

A. Sivannarayana, K. Harikishore

Abstract*—The special moduli set,Residue Number System is intended to implement the long and repeated multiplications of cryptographic and signal processing algorithms. In this paper, area and power trade-of of modulo $2^n - 1$ and modulo $2^n + 1$ multipliers based on RNS are proposed. The proposed modulo multipliers are based on the radix-8 Booth encoding technique. In the proposed modulo $2^n - 1$ multipliers, the number of partial products is lowered to $\lfloor n/3 \rfloor + 1$ for n = 32 to 64, which is around 33% reduction over radix-4 Booth encoded multiplier for n = 32 to 64.For modulo $2^n + 1$multiplier,the aggregate bias is composed of multiplier dependent dynamic bias and multiplier independent static bias due to hard multiple and modulo-reduced partial products generation. The total number of partial products is reduced to $\lfloor n/3 \rfloor + 6$ for modulo $2^n + 1$ multiplier. From synthesis results for modulo $2^n - 1$ and modulo $2^n + 1$ based RNS multipliers constructed from different modulo $2^n - 1$ and modulo $2^n + 1$ multipliers.

***Index terms*-Booth algorithm, computer arithmetic, multiplic-ation, residue number system(RNS).**

I. INTRODUCTION

High speed digital signal processors and cryptographic cores are strategically implemented in Residue Number System (RNS) [1]–[6].The RNS is defined by a vector of k moduli, (m1,m2,...,mk). The moduli set must be pair wise co-prime, which means that for any pair of moduli, the only common factor is 1.In RNS, each operand is represented by a list of its residues, one for each modulus. RNS implementation is faster than the TCS counterpart because the computations are performed in short word-length modulo channels without carry propagation between channels. The techniques such as multi-modulus and multi-function architectures to minimize the hardware redundancy as well as multi-threshold voltage and multi-supply voltage designs to lower the power dissipation have been suggested in [8]–[10]. Such control techniques are intended for algorithm level design space exploration and are applicable to the generic modulo arithmetic architectures. The early known full-adder based implementation of modulo $2^n - 1$ multiplication was presented in [11] and [12], where no encoding scheme was used on the multiplier bits and partial products were added in a regularly structured Carry Save adder (CSA) tree. In [12] and [13], radix-4 Booth encoding was employed and the number of partial products was reduced to $\lfloor n/2 \rfloor + 1$ and $\lceil n/2 \rceil$ respectively, leading to better overall area-delay-power performance, where $\lceil k \rceil$ and $\lfloor k \rfloor$ represent the smallest integer greater than or equal to k and the greatest integer smaller than or equal to k respectively.

Manuscript published on 30 July 2013.
*****Correspondence Author(s)
A.Sivannarayana, (Department of Electronics & Communication Engineering, KL University, Vijayawada. India.
K.Harikishore, (Department of Electronics & Communication Engineering, KL University, Vijayawada, India.

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

The residues for modulus $2^n + 1$ require the n + 1 bits to be represented in the traditional binary system. To limit the modulo $2^n + 1$ arithmetic to n bits the unconventional number representations have been used [12],[19]. The modulo $2^n + 1$ multiplier using the diminished-1 number representation have been proposed in [14]-[18].By increasing the radix of Booth encoding scheme beyond four, the number of partial products can be further reduced leading to even greater savings in silicon area and power dissipation [21]–[23]. Ideally, the radix-8 Booth encoding reduces the number of partial products to $\lfloor n/3 \rfloor + 1$, implying a maximal reduction of 35% over the radix-4 Booth encoding. But this reduction in the number of partial products comes at the expense of increased logic in generating them. In particular, the computation of three times the multiplicand, 3X is non-trivial and hence, 3X is termed the hard multiple of the radix-8 Booth encoded multiplication scheme. In [24], the radix-8 Booth encoding technique was explored for modulo $2^n - 1$ multiplication for the first time. The hard multiple, $|+3X|_{2n-1}$ was computed using -bit Ripple Carry Adders (RCAs) operating in parallel. The carry-out bits of the RCAs were not propagated but treated as partial product bits resulting in a partially-redundant representation of the hard multiple. As the word-length of the RCA can be manipulated to match the modulo $2^n - 1$ multiplier delay to the RNS delay, [24] is designed to exploit the timing slack of the non-critical modulo $2^n - 1$ channel in high dynamic range RNSs based on imbalanced word-length moduli sets. Our objective is to minimize the area and power consumption of VLWL multiplication in RNS based on moduli $2^n - 1$ and $2^n + 1$. Based on the preliminary customized adders we proposed in [25] and [26], a new logic cell that operates on the inverted-and-swapped generate and propagate signal pair is proposed to further simplify the prefix-computations in the generation of hard multiples. The proposed modulo $2^n - 1$ and the modulo $2^n + 1$ hard multiple generators (HMGs) employ only $\lceil \log_2 n \rceil - 1$ prefix levels. In the proposed modulo $2^n - 1$ multiplier, one partial product per radix-8 Booth encoded multiplier digit is generated. As the hard multiple is generated in an unbiased form, no further correction term is incurred. Thus, the number of modulo-reduced partial products in modulo $2^n - 1$ multiplier is lowered to $\lfloor n/3 \rfloor + 1$, which is the best achievable partial product count using radix-8 Booth encoding scheme. For proposed modulo $2^n + 1$ multiplier the aggregate sum of biasing constants is divided into multiplier dependent dynamic bias and multiplier independent static bias. The bias is generated by hardwiring the Booth encoder outputs and constant ones to appropriate the bit positions via at most two levels of the AND gates. The customized hard multiple generators, and the simple correction term for the modulus $2^n + 1$ and fewer modulo –reduced partial products will improve the area-power trade-off performance of the proposed modulo multipliers [7].

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

TABLE I. RADIX-8 BOOTH ENCODING

Multiplier bits $y_{3i+2}\ y_{3i+1}\ y_{3i}\ y_{3i-1}$	Signed multiplier digit d_i
0000	+0
0001	+1
0010	+1
0011	+2
0100	+2
0101	+3
0110	+3
0111	+4
1000	-4
1001	-3
1010	-3
1011	-2
1100	-2
1101	-1
1110	-1
1111	-0

II. MODULO-REDUCED PARTIAL PRODUCTS FOR RADIX-8 BOOTH ENCODED MULTIPLICATION

This section presents the preliminaries of radix-8 Booth encoded modulo multiplication using binary representation with dual zeros for modulo $2^n - 1$ arithmetic. For simplicity, all equations are assumed to be modulo-reduced by m, m ϵ $\{2^n - 1\}$. Mathematically, the modulo $2^n - 1$ multiplication can be expressed as follows.

$$|Z|_m = \begin{cases} X \cdot Y = \sum_{i=0}^{n-1} X.\, y_i \cdot 2^i & \text{if } m = 2^n - 1 \\ X \cdot Y + X + Y & \\ \sum_{i=0}^{n-1} X.\, y_i \cdot 2^i + X + Y & \text{if } m = 2^n + 1 \end{cases} \quad \text{__(1)}$$

Where X, Y and Z represent the n-bit multiplicand, multiplier and product, respectively [7].

By employing the radix-8 Booth encoding technique it can be used to reduce the multiplier bits, $\{y_i\}_{i=0}^{n-1}$ to signed digits, $\{d_i\}_{i=0}^{\lfloor n/3 \rfloor}$. By employing the radix-8 Booth encoding, the product can be computed from only $\lfloor n/3 \rfloor + 1$ modulo-reduced partial products, $|X \cdot d_i \cdot 2^{3i}|_m$ and (1) can be expressed succinctly as follows [7].

$$|Z|_m = \begin{cases} \sum_{i=0}^{\lfloor n/3 \rfloor} X.\, d_i \cdot 2^{3i} & \text{if } m = 2^n - 1 \\ \sum_{i=0}^{\lfloor n/3 \rfloor} X.\, d_i \cdot 2^{3i} + X + Y & \text{if } m = 2^n + 1 \end{cases} \quad \text{_____(2)}$$

TABLE II
Modulo $2^n - 1$ Reduced Partial Products for Radix-8 Booth Encoding:

d_i	$\lvert X \cdot d_i \cdot 2^{3i} \rvert_{2^n-1} = PP_i$
+0	CLS (0…0 , 3i)
+1	CLS (X , 3i)
+2	CLS (X , 3i + 1)
+3	CLS ($\lvert +3X \rvert_{2^n-1}$, 3i)
+4	CLS (X , 3i + 2)
-0	CLS (1…1 , 3i)
-1	CLS (X^- , $3i$)
-2	CLS (X^- , $3i + 1$)
-3	CLS ($\lvert +3X \rvert_{2^n-1}$, 3i)
-4	CLS (X^- , $3i + 2$)

The generation of $|X \cdot d_i \cdot 2^{3i}|_m$ can be simplified by virtue of the number theoretic properties of modulo $2^n - 1$ and modulo $2^n + 1$ arithmetic summarized below [17], [18], [24].

1) Property 1: Let X^- denote the one's complement of X. The modulo negation of X is given by,

$$|-X|_m = \begin{cases} X^- & \text{if } m = 2^n - 1 \\ X^- + 2 & \text{if } m = 2^n + 1 \end{cases} \quad \text{_____(3)}$$

2) Property 2: The modulo reduction of the binary weight of x in excess of the modulus is given by,

$$|x \cdot 2^{n+i}|_m = \begin{cases} x \cdot 2^i & \text{if } m = 2^n - 1 \\ |-x \cdot 2^i|_m = X^- \cdot 2^i - 2^i & \text{if } m = 2^n + 1 \end{cases} \quad \text{____(4)}$$

3) Property 3: Let the circular-left-shift, CLS(X , j) and complementary circular left shift,CCLS (X , j) operations denote the circular left shift of X by j bit positions where the j least significant bits (lsbs) are not inverted and inverted, respectively. As a corollary of Property 2, the modulo multiplication of X by a positive power-of-two term, 2^j is given by,

$$|2^j \cdot X|_m = \begin{cases} CLS(X , j) & \text{if } m = 2^n - 1 \\ CCLS(X , j) - 2^j + 1 & \text{if } m = 2^n + 1 \end{cases} \quad \text{____(5)}$$

4) Property 4: By Properties 1 and 3, the modulo multiplication of X by -2^j is given by,

$$|-2^j \cdot X|_m = \begin{cases} CLS(X^- , j) & \text{if } m = 2^n - 1 \\ CCLS(X^- , j) + 2^j + 1 & \text{if } m = 2^n + 1 \end{cases} \quad \text{___(6)}$$

Properties 3 and 4 imply that the modulo $2^n - 1$ multiplications of X by 2^j and -2^j can be efficiently implemented in hardware by CLS operations on X and X^-. Similarly,the modulo $2^n + 1$ multiplications by 2^j and -2^j can be realized by CCLS operations on X with a bias of $-2^j + 1$ and on X^- with a bias of $2^j + 1$ respectively.The generation of $|X \cdot d_i \cdot 2^{3i}|_m$ can be described by a CLS vector PP_i for m = $2^n - 1$ as shown in Table II and by the sum of a CCLS vector PP_i and a predefined bias K_i for m = $2^n + 1$ as shown in Table III [7]. Hence, (2) can be further simplified to,

$$|Z|_m = \begin{cases} \sum_{i=0}^{\lfloor n/3 \rfloor} PP_i & \text{if } m = 2^n - 1 \\ \sum_{i=0}^{\lfloor n/3 \rfloor} (PP_i + k_i) + X + Y & \text{if } m = 2^n + 1 \end{cases} \quad \text{_____(7)}$$

All modulo-reduced partial product expressions, except $|+3X|_m$ in Tables II can be implemented by selective bit shift and selective bit inversion. The bottleneck $|+3X|_m$ is commonly referred to as the hard multiple of radix-8 Booth encoded multiplier. To minimize its hardware overhead, new customized adders are proposed for generating the hard multiple of radix-8 Booth encoded modulo $2^n - 1$ and modulo $2^n + 1$ multipliers.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

III. PROPOSED MODULO $2^n - 1$ AND MODULO $2^n + 1$ HARD MULTIPLE GENERATORS (HMGS):

Modulus $2^n - 1$:

The modulo $2^n - 1$ addition of two operands, A and B is equivalent to an n-bit addition of A ,B and c_{out}, [7], i.e.,

$$|A + B|_{2^n - 1} = \begin{cases} |A + B|_{2^n} & \text{if } A + B < 2^n \\ |A + B + 1|_{2^n} & \text{if } A + B \geq 2^n \end{cases} = |A + B + c_{out}|_{2^n} \quad \text{_____(8)}$$

Where c_{out} is the carry output resulting from the addition of A and B. As c_{out} is added to the sum of A and B at the lsb position, modulo $2^n - 1$ addition is commonly referred to as end-around-carry (EAC) addition.

This modulo $2^n - 1$ adder can be implemented by a parallel-prefix structure with three operator stages, namely pre-processing,prefix-computation and post-processing [32], [33].

The pre-processing stage computes the generate (g_i), propagate (p_i) and half-sum (h_i) bits for to i = 0 to n – 1.

$$g_i = a_i \,.\, b_i$$
$$p_i = a_i + b_i$$
$$h_i = a_i \wedge b_i \quad \text{_____(9)}$$

The prefix-computation stage will use the prefix operators (•) on (g_i , p_i) to calculate the carry bits c_i for the EAC addition.

For i = 0 to n – 1

$c_i = (g_i , p_i) \bullet \ldots \bullet (g_0 , p_0) \bullet (g_{n-1} , p_{n-1}) \bullet \ldots \bullet (g_{i+1} , p_{i+1})$ __(10)
where $(g_i , p_i) \bullet (g_{i-1} , p_{i-1}) = (g_i + p_i \,.\, g_{i-1} , p_i \,.\, p_{i-1})$.

The post-processing stage computes the sum bit s_i for i = 0 to n - 1, i.e.,

$s_i = h_i \wedge c_{i-1}$. _____(11)

The hard multiple $|+3X|_{2^n - 1}$ is generated by adding $X = \sum_{i=0}^{n-1} x_i \,.\, 2^i$ and $|2X|_{2^n-1} = CLS(X , 1) = \sum_{i=0}^{n-2} x_i \,.\, 2^{i+1} + x_{n-1} \,.\, 2^0$

(5) Property 5: For the addition of X and CLS (X , 1),

$$p_i \,.\, g_{i-1} = \begin{cases} (x_0 .+ x_{n-1}) \,.\, (x_{n-1} \,.\, x_{n-2}) \\ = x_{n-1} \,.\, x_{n-2} = g_{n-1} & i = 0 \\ (x_1 + x_0) \,.\, (x_0 \,.\, x_{n-1}) \\ = x_0 \,.\, x_{n-1} = g_0 & i = 1 \\ (x_i + x_{i-1}) \,.\, (x_{i-1} \,.\, x_{i-2}) \\ = x_{i-1} \,.\, x_{i-2} = g_{i-1} & 2 \leq i \leq n-1 \end{cases} \quad \text{_____(12)}$$

By applying the property-5 to (10), only n/2 *modified generate- propagate bit pairs*,(g_i^* , p_i^*) are required in the prefix-computation stage of modulo $2^n - 1$ HMG as follows.

$c_i = (g_i^* , p_i^*) \bullet (g^*_{i-2} , p^*_{i-2}) \bullet \ldots \bullet (g^*_0 , p^*_0) \bullet (g^*_{n-2} , p^*_{n-2})$
$\bullet \ldots \bullet (g^*_{i+2} , p^*_{i-2})$ for even i
$= (g_i^* , p_i^*) \bullet (g^*_{i-2} , p^*_{i-2}) \bullet \ldots \bullet (g^*_1 , p^*_1) \bullet (g^*_{n-1} , p^*_{n-1})$
$\bullet \ldots \bullet (g^*_{i+2} , p^*_{i+2})$ for odd i _____(13)

where,

$$g_i^* = \begin{cases} g_1 + g_0 = x_0 \,.\, (x_1 + x_{n-1}) & i = 1 \\ g_i + g_{i-1} = x_{i-1} \,.\, (x_i + x_{i-2}) & 2 \leq i \leq n-1 \end{cases}$$

$$p_i^* = \begin{cases} p_0 \,.\, p_{n-1} = x_{n-1} + x_0 \,.\, x_{n-2} & i = 0 \\ p_1 \,.\, p_0 = x_0 + x_1 \,.\, x_{n-1} & i = 1 \\ p_i \,.p_{i-1} = x_{i-1} + x_i \,.\, x_{i-2} & 2 \leq i \leq n-1 \end{cases} \quad \text{_____(14)}$$

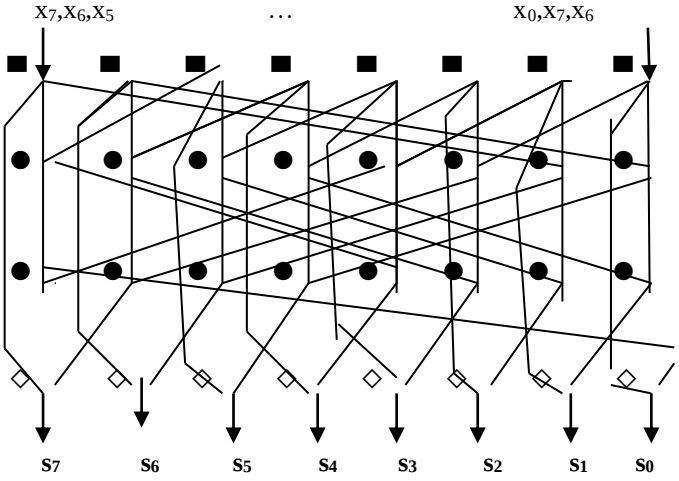


Figure -1: The Proposed Modulo $2^n - 1$ Hard Multiple Generator for n = 8.

Figure -1 shows the proposed modulo $2^n - 1$ hard multiple generator for n = 8 [7].The operator "■" represents the operations such as OR – AND , AND – OR , XOR gate for generating g_i^*, p_i^* , and h_i respectively in the pre-processing stage. There are $\lceil \log_2 n \rceil - 1 = 2$ levels of prefix operators in the prefix – computation stage. The operator "◊" which represents the sum bit s_i by the XOR of h_i and c_{i-1} in the post – processing stage.

Modulus $2^n + 1$:

The modulo **$2^n + 1$** addition of the two diminished-1 represented operands, A and B is equivalent to an n-bit addition of A and B along with c_{out}^-, i.e., as follows.

$$|S + 1|_{2^n + 1} = |A + B + 1|_{2^n + 1} = |A + B + c_{out}^-|_{2^n} \quad \text{----------(15)}$$

Where c_{out} is the carry output from the addition of A and B. As c_{out}^- is added to the sum of A and B at the lsb position, modulo $2^n + 1$ addition is called complementary end around carry (CEAC) addition. The pre-processing stage and post-processing stage of the parallel prefix modulo $2^n + 1$ adder are similar to those of the modulo $2^n - 1$ adder except the carry equation is implemented differently in the prefix-computation stage due to CEAC addition from [29], [30].The carry equation is computed as follows.

$c_i = (g_i , p_i) \bullet \ldots \bullet (g_0 , p_0) \bullet \overline{(g_{n-1} , p_{n-1}) \bullet \ldots \bullet (g_{i+1} , p_{i+1})}$
----------(16)

For the modulo $2^n + 1$ hard multiple generator, the addends are expressed as $X = \sum_{i=0}^{n-1} x_i \,.\, 2^i$ and $CCLS(X , 1) = \sum_{i=0}^{n-2} x_i \,.\, 2^{i+1} + x_{n-1}^- \,.\, 2^0$ to use the bit correlation between them. From the property-3,

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

$| 2 . X |_{2^n+1}$ = CCLS (X , 1) – 1 ----------(17)

From the definition of modulo $2^n + 1$ addition in equations (15) and (17) the sum of X and CCLS (X , 1) is computed by,

$| X + 2 . X + 1 + 1 |_{2^n+1} = | 3 . X + 2|_{2^n+1}$ ----------(18)

By the constant bias of 2, the hard multiple is generated.Instead of eliminating this constant bias from the addition, which will have to increase the critical path delay of the hard multiple generator (HMG), that is merged into the modified partial products, as described in the following section.

(6).Property 6: For the addition of X and CCLS (X , 1),

$$p_0 \cdot p_{n-1}^- = (x_0 + x_{n-1}^-)(x_{n-1}^- \cdot x_{n-2}^-) = p_{n-1}^- \quad \text{----------(19)}$$

$$p_i \cdot g_{i-1} = \begin{cases} (x_1 + x_0) \cdot (x_0 \cdot x_{n-1}^-) \\ \quad = x_0 \cdot x_{n-1}^- = g_0 & \text{for i = 1} \\ (x_i + x_{i-1}) \cdot (x_{i-1} \cdot x_{i-2}) \\ \quad = x_{i-1} \cdot x_{i-2} = g_{i-1} & 2 \le i \le \text{n-1} \end{cases} \quad \text{----------(20)}$$

$$g_i^- \cdot p_{i-1}^- = \begin{cases} (x_1^- + x_0^-) \cdot (x_0^- \cdot x_{n-1}) \\ \quad = x_0^- \cdot x_{n-1} = p_0^- & \text{for i = 1} \\ (x_i^- + x_{i-1}^-) \cdot (x_{i-1}^- \cdot x_{i-2}^-) \\ \quad = x_{i-1}^- \cdot x_{i-2}^- = p_{i-1}^- & 2 \le i \le \text{n-1} \end{cases} \quad \text{----------(21)}$$

By intending property-6,the carry equation,c_i is computed in the prefix computation and the number of modified generate (g_i*) bits and propagate (p_i*) bits are reduced to n/2 as shown below.

$$c_i = \bullet \begin{cases} (g_i^*, p_i^*) \bullet (g_{i-2}^*, p_{i-2}^*) \bullet \ldots \bullet (g_0^*, p_0^*) \bullet (p_{n-2}^{-*}, g_{n-2}^{-*}) \bullet \ldots \\ \quad \bullet (p_{i+2}^{-*}, g_{i+2}^{-*}) & \text{if i is even} \\ (g_i^*, p_i^*) \bullet (g_{i-2}^*, p_{i-2}^*) \bullet \ldots \bullet (g_1^*, p_1^*) \bullet (p_{n-1}^{-*}, g_{n-1}^{-*}) \\ \quad \ldots \bullet (p_{i+2}^{-*}, g_{i+2}^{-*}) & \text{if i is odd} \end{cases}$$

----------(22)

Where,

$$g_i^* = \begin{cases} g_0 + p_{n-1}^- = x_{n-1}^- \cdot (x_0 + x_{n-2}^-) & \text{if i = 0} \\ g_1 + g_0 = x_0 \cdot (x_1 + x_{n-1}^-) & \text{if i = 1} \\ g_i + g_{i-1} = x_{i-1} \cdot (x_i + x_{i-2}) & 2 \le i \le \text{n-1} \end{cases}$$

$$p_i^* = \begin{cases} p_0 \cdot g_{n-1}^- = x_{n-1}^- + x_0 \cdot x_{n-2}^- & \text{if i = 0} \\ p_1 \cdot p_0 = x_0 + x_1 \cdot x_{n-1}^- & \text{if i = 1} \\ p_i \cdot p_{i-1} = x_{i-1} + x_i \cdot x_{i-2} & 2 \le i \le \text{n-1} \end{cases} \quad \text{---------(23)}$$

In [26],equation-(22) is implemented by two identical prefix operators such that input to one of the operators,(g_i^*, p_i^*) is pair-wise swapped and inverted to form input (p_i^{-*}, g_i^{-*}) for the other operator.The number of dual prefix operators required in each prefix level l, $1 \le l \le \lceil \log_2 n \rceil - 1$, will be given by n - 2^{l+1}. To remove the implementation area and the interconn-

Ects, of dual prefix operators,a modified prefix operator which computes $(g_i^*, p_i^*) \bullet (g_j^*, p_j^*)$ and $(p_i^{-*}, g_i^{-*}) \bullet (p_j^{-*}, g_j^{-*})$ from the inputs (g_i^*, p_i^*) and (g_j^*, p_j^*),using the fewer number of logic gates is proposed.

From equation-(23),for (g_i^*, p_i^*) it is easily checked that,

$$g_i^* \cdot p_i^* = g_i^* \text{ and } p_i^{-*} \cdot g_i^{-*} = p_i^{-*} \quad \text{----------(24)}$$

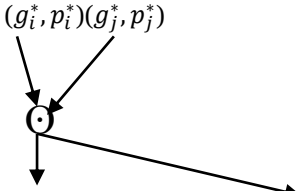
By intending equation-(24),the output of modified prefix oper-ator, can be computed as follows in the first prefix level,l = 1.

$$(g_i^*, p_i^*) \bullet (g_j^*, p_j^*) = (g_i^* + p_i^* \cdot g_j^*, p_i^* \cdot p_j^*)$$

$$= \overline{(g_i^{-*} \cdot g_j^{-*}} \cdot p_i^*, p_i^* \cdot p_j^*)$$

$$(p_i^{-*}, g_i^{-*}) \bullet (p_j^{-*}, g_j^{-*}) = (p_i^{-*} + g_i^{-*} \cdot p_j^{-*}, g_i^{-*} \cdot g_j^{-*})$$

$$= (\overline{p_i^* \cdot p_j^* \cdot} \; g_i^{-*}, g_i^{-*} \cdot g_j^{-*}) \quad \text{-(25)}$$



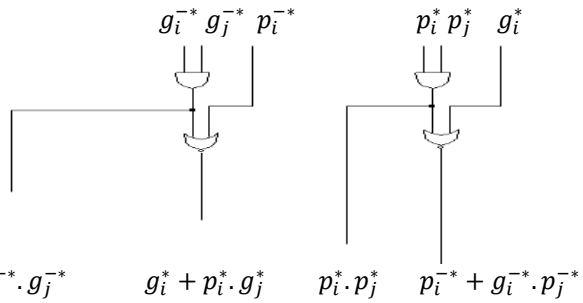


Fig-2: The modified prefix operator

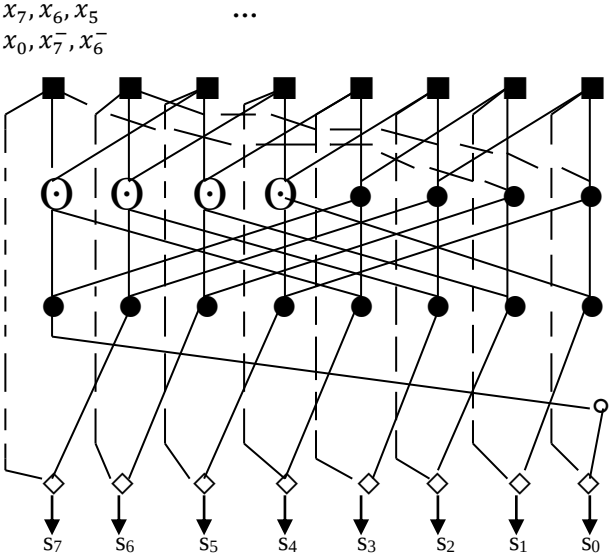


Fig-3: The Proposed Modulo $2^n + 1$ Hard Multiple Generator for n = 8

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

The Fig-3 shows the proposed modulo $2^n + 1$ hard multiple generator for n = 8. The computations are identical to those of Fig-1 in pre-processing stage and post-processing stage. We generate the carry signals c_i for i = 0 to n-1 in $\lceil \log_2 n \rceil - 1$=2 levels of prefix operators.The dotted line represents modified generate and propagate signals which have been pair wise swapped and complemented from the outputs of pre-processing st-ge, to produce (p_i^{-*}, g_i^{-*}), [7].

IV. PROPOSED MODULO $2^n - 1$ AND MODULO $2^n + 1$ MULTIPLIERS

The expressions of partial products of the modulo $2^n - 1$ and modulo $2^n + 1$ have been shown in tables II and III,[7].

TABLE III
MODULO $2^n + 1$ REDUCED PARTIAL PRODUCTS FOR RADIX – 8 BOOTH ENCODING

d_i	$\lvert X \cdot d_i \cdot 2^{3i} \rvert_{2^n+1} = PP_i + K_i$	
	PP_i	K_i
+0	CCLS (0…0, 3i)	$-2^{3i} + 1$
+1	CCLS (x , 3i)	$-2^{3i} + 1$
+2	CCLS (X , 3i + 1)	$-2^{3i+1} + 1$
+3	CCLS ($\lvert +3X \rvert_{2^n+1}$, 3i)	$-2^{3i} + 1$
+4	CCLS (X , 3i + 2)	$-2^{3i+2} + 1$
-0	CCLS (1…1, 3i)	$2^{3i} + 1$
-1	CCLS ($X^-, 3i$)	$2^{3i} + 1$
-2	CCLS ($X^-, 3i + 1$)	$2^{3i+1} + 1$
-3	CCLS ($\lvert +3X \rvert_{2^n+1}$, 3i)	$2^{3i} + 1$
-4	CCLS ($X^-, 3i + 2$)	$2^{3i+2} + 1$

This section will complete the architecture of proposed multiplIers with circuits for generating and accumulating the modulo –Reduced partial products.
Figure-4(a) shows the bit-slice implementation of the radix-8 Booth encoding of Table I.

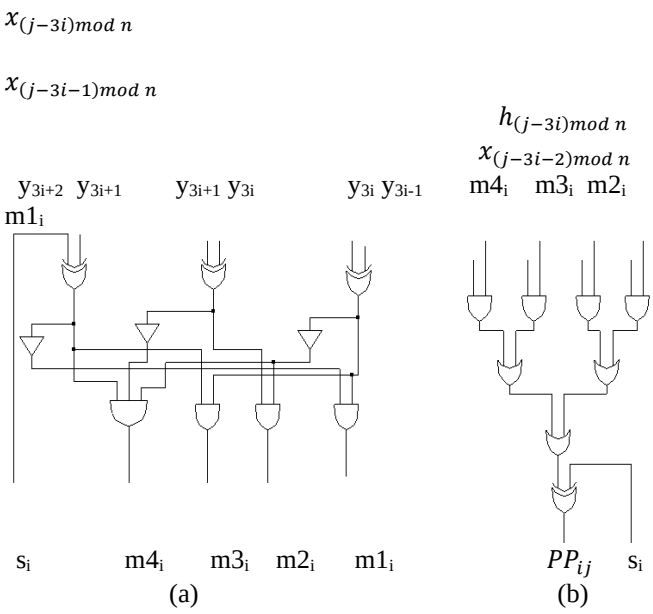


Figure-4 (a) Bit-slice of Booth Encoder (b) Bit-slice of Booth Selector

Booth Encoder (BE) will produce the signed digit represented by a sign bit s_i and the one-hot encoded magnitude bits denoted by $m4_i$, $m3_i$, $m2_i$, and $m1_i$ from the four consecutive multiplier bits y_{3i+2}, y_{3i+1}, y_{3i}, and y_{3i-1}.The left most bit of the BE bit-slicey3i-1 is shared with right most bit of the preceding BE bit-slice $y_{3(i-1)+2}$.To generate the PP_i vector given by Tables II and III, the one hot encoded bits, $m1_i$ to $m4_i$ are intended to select either the multiplicand,one-bit shift multiplicand,hard multiple or two-bit shift multiplicand.The sign bit determines whether the selected word needs to be complemented.The bit-slice of the radix-8 Booth Selector (BS) is shown in Figure-4(b),where h_j denotes the j-th bit of the hard multiple generated according to the Figure-(1) for modulo $2^n - 1$ and the Figure-(3) for modulo $2^n + 1$ multipliers,[7].

The generation of PP_i in BE, BS and HMG blocks for moduli $2^n - 1$ and $2^n + 1$ are shown in Figures (5) and (6),respectively for n = 8.A group of n identical BS blocks is needed to provide the single PP_i.Espicially for modulo $2^n + 1$ multiplier,outputs of the BS blocks at the least-significant 3i bit positions will be inverted to implement the CCLS operation.

The modulo reduced partial product accumulation expressed by (7) differs significantly for modulo $2^n - 1$ and modulo $2^n + 1$ multipliers.Since modulo $2^n - 1$ addition can be implemented by the EAC addition,The CSA for its MPPA is shown in Figure-7 for n = 8.Besides the $\lfloor n/3 \rfloor + 3PP_i$, the additional $\lfloor n/3 \rfloor + 1$ biasing constants K_i is needed to be accumulated as expressed in (7) for the radix-8 Booth encoded modulo $2^n + 1$ multiplier. Furthermore, $\lvert 3.X + 2 \rvert_{2^n+1}$ instead of $\lvert 3.X \rvert_{2^n+1}$ is generated by the hard multiple generator given by (22) and described in Figure-3.The additional bias of 2 will be accounted for by modifying the PP_i and K_i corresponding to Booth encoded digit $d_i = \pm 3$ as shown below [7].

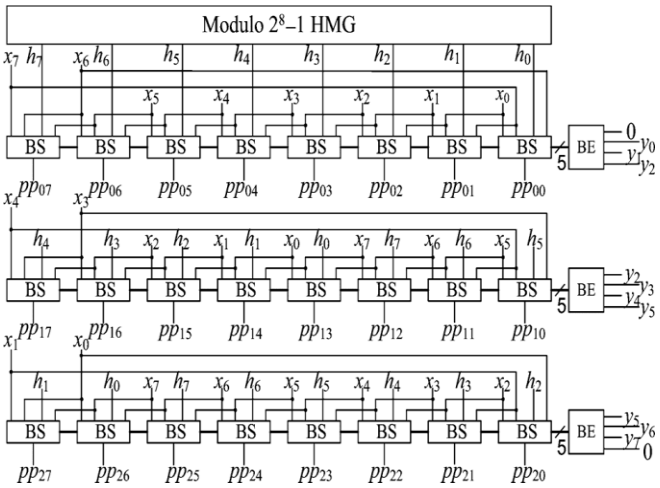


Fig.5. Modulo $2^n - 1$ PP_i generation for n = 8.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

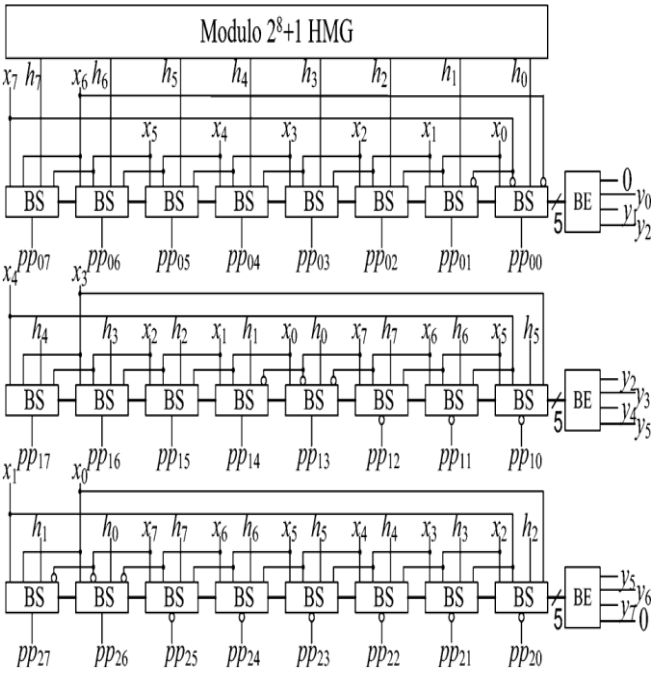


Fig.6. Modulo $2^n + 1$ PP_i generation for n = 8.

TABLE IV MODULO 2n + 1 REDUCED PARTIAL PRODUCTS FOR MULTIPLIER DIGITS ±3

di	$\lvert X \cdot d_i \cdot 2^{3i}\rvert_{2^n+1} = PP_i + K_i$	
	PP_i	K_i
+3	CCLS ($\lvert +3X + 2\rvert_{2^n+1}$, 3i)	$-2^{3i+1} - 2^{3i} + 1$
-3	$\overline{\text{CCLS}}$ ($\lvert +3X + 2\rvert_{2^n+1}$, 3i)	$2^{3i+1} + 2^{3i} + 1$

The modified PPi and Ki for di = ±3 are shown in the Table IV.

$$|+3 \cdot X \cdot 2^{3i}|_{2^n+1}$$
$$= |(3 \cdot X + 2) \cdot 2^{3i}|_{2^n+1} - |2 \cdot 2^{3i}|_{2^n+1}$$
$$= \text{CCLS}(|3X + 2|_{2^n+1}, 3i) - 2^{3i} + 1 - 2 \cdot 2^{3i}$$
$$= \text{CCLS}(|3X + 2|_{2^n+1}, 3i) - 2^{3i+1} - 2^{3i} + 1 \quad \text{_____(26)}$$

TABLE V
DYNAMIC BIAS FOR MULTIPLIER DIGITS

d_i	K_i	$KD_i = K_i + 2^{3i} - 1$
+0	$-2^{3i} + 1$	0
+1	$-2^{3i} + 1$	0
+2	$-2^{3i+1} + 1$	-2^{3i}
+3	$-2^{3i} + 1 - 2^{3i+1}$	-2^{3i+1}
+4	$-2^{3i+2} + 1$	$-2^{3i} - 2^{3i+1}$
-0	$2^{3i} + 1$	2^{3i+1}
-1	$2^{3i} + 1$	2^{3i+1}
-2	$2^{3i+1} + 1$	$2^{3i} + 2^{3i+1}$
-3	$2^{3i} + 1 + 2^{3i+1}$	2^{3i+2}
-4	$2^{3i+2} + 1$	$2^{3i+2} + 2^{3i}$

By using Property-3,modulo $2^n + 1$ multiplication by 2^{3i} introduces the bias of $-2^{3i} + 1$. K_i will be divided into two parts,$KS_i = -2^{3i} + 1$ and $KD_i = K_i - KS_i$.KS_i is the static bias which is specific to the radix of the Booth encoding algorithm and independent of d_i and KD_i is the dynamic bias which is dependent on d_i.K_i and KD_i for all d_i are tabulated in Table V from the Tables III and IV. The fixed KS_i will be easily aggregated into a single constant word.Let $KD_i = -2^{3i}a - 2^{3i+1}b + 2^{3i+1}c + 2^{3i+1}d + 2^{3i+2}e$,where a,b,c,d,e ϵ {0 , 1}.The truth table of a,b,c,d,e with Boolean inputs s_i, $m1_i$, $m2_i$, $m3_i$, $m4_i$ is given in the Table VI.

Those minterms which are not listed in Table VI will be the don't cares [7].

TABLE VI
TRUTH TABLE FOR BOOLEAN FUNCTIONS OF a,b,c,d,e

d_i	KD_i	s_i	$m4_i$	$m3_i$	$m2_i$	$m1_i$	a	b	c	d	e
+0	0	0	0	0	0	0	0	0	0	0	0
+1	0	0	0	0	0	1	0	0	0	0	0
+2	-2^{3i}	0	0	0	1	0	1	0	0	0	0
+3	-2^{3i+1}	0	0	1	0	0	0	1	0	0	0
+4	$-2^{3i} - 2^{3i+1}$	0	1	0	0	0	1	1	0	0	0
-0	2^{3i+1}	1	0	0	0	0	0	0	1	0	0
-1	2^{3i+1}	1	0	0	0	1	0	0	1	0	0
-2	$-2^{3i}+2^{3i+1}+2^{3i+1}$	1	0	0	1	0	1	0	1	1	0
-3	$-2^{3i+1}+2^{3i+1}+2^{3i+2}$	1	0	1	0	0	0	1	1	0	1
-4	$-2^{3i} - 2^{3i+1} + 2^{3i+1} + 2^{3i+1} + 2^{3i+2}$	1	1	0	0	0	1	1	1	1	1

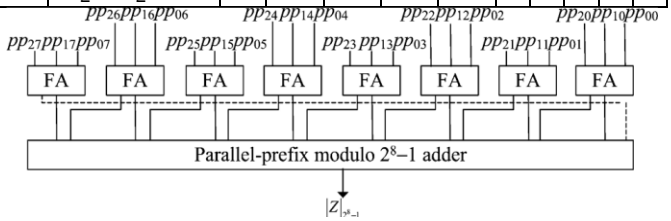


Fig.7. Modulo $2^n - 1$ reduced partial product accumulation for n = 8.

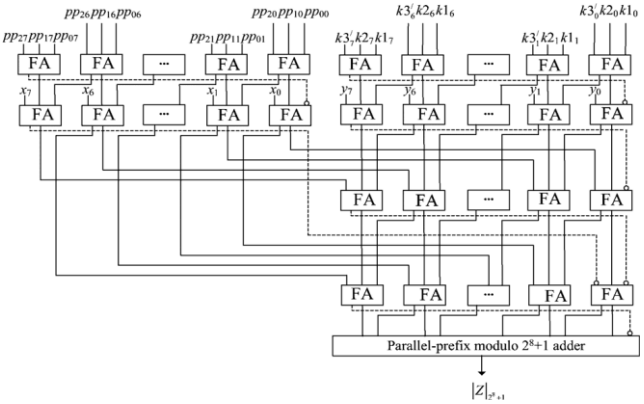


Fig.8. Modulo $2^n + 1$ reduced partial product accumulation for n = 8.

Minimization by a 5-variable K- map yields $a = m2_i \vee m4_i$, $b = m3_i \vee m4_i$,$c = s_i$,$d = (m2_i \vee m4_i) \wedge s_i$ and $e = (m3_i \vee m4_i) \wedge s_i$. So,

$$KD_i = -(m2_i \vee m4_i) \cdot 2^{3i} - (m3_i \vee m4_i) \cdot 2^{3i+1} + s_i \cdot 2^{3i+1} + ((m2_i \vee m4_i) \wedge s_i) \cdot 2^{3i+1} + ((m3_i \vee m4_i) \wedge s_i) \cdot 2^{3i+2}$$
_____(27)

By applying the Property-2 to the negatively weighted ($m2_i \vee m4_i$) and ($m3_i \vee m4_i$) terms,then we have,[7],

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

$KD_i = (\overline{m2_i \vee m4_i}) \cdot 2^{3i} - 2^{3i} + \overline{(m3_i \vee m4_i)} \cdot 2^{3i+1} - 2^{3i+1} + s_i \cdot 2^{3i+1} + ((m2_i \vee m4_i) \wedge s_i) \cdot 2^{3i+1} + ((m3_i \vee m4_i) \wedge s_i) \cdot 2^{3i+2}$.

_____(28)

For the case of mod(n , 3) = 2,the aggregate sum of KS_i and KD_i is given by,

$$\sum_{i=0}^{\lfloor n/3 \rfloor} K_i$$

$$= \sum_{i=0}^{\lfloor n/3 \rfloor} \frac{(\overline{m2_i \vee m4_i}) \cdot 2^{3i} + \overline{(m3_i \vee m4_i)} \cdot 2^{3i+1} + s_i \cdot 2^{3i+1}}{KD_i} + \sum_{i=0}^{\lfloor n/3 \rfloor} \frac{((m2_i \vee m4_i) \wedge s_i) \cdot 2^{3i+1} + ((m3_i \vee m4_i) \wedge s_i) \cdot 2^{3i+2}}{KD_i} + \sum_{i=0}^{\lfloor n/3 \rfloor} \frac{-2^{3i} - 2^{3i+1} - 2^{3i} + 1}{KDi \quad KSi}$$

$$= K1 + k2 + K3 + \sum_{i=0}^{\lfloor n/3 \rfloor} (-2^{3i+2}) + \lfloor n/3 \rfloor + 1 \quad \text{_____(29)}$$

Where

$$K1 = \sum_{i=0}^{\lfloor n/3 \rfloor} (\overline{m2_i \vee m4_i}) \cdot 2^{3i} + \overline{(m3_i \vee m4_i)} \cdot 2^{3i+1}$$

$$K2 = \sum_{i=0}^{\lfloor n/3 \rfloor} ((m2_i \vee m4_i) \wedge s_i) \cdot 2^{3i+1} + ((m3_i \vee m4_i) \wedge s_i) \cdot 2^{3i+2}$$

$$K3 = \sum_{i=0}^{\lfloor n/3 \rfloor} s_i \cdot 2^{3i+1}. \quad \text{_____(30)}$$

By substituting for ki in (7) by (29),

$$|Z|_{2^n+1} = \frac{\sum_{i=0}^{\lfloor n/3 \rfloor} PP_i + X + Y + K1 + K2 + K3}{\lfloor n/3 \rfloor + 6} + \sum_{i=0}^{\lfloor n/3 \rfloor} (-2^{3i+2}) + \lfloor n/3 \rfloor + 1 \quad \text{_____(31)}$$

Z is the sum of $\lfloor n/3 \rfloor + 6$ partial products which are dependent on the multiplicand and/or the multiplier and the constant term, $\sum_{i=0}^{\lfloor n/3 \rfloor} (-2^{3i+2}) + \lfloor n/3 \rfloor + 1$. The letter won't be considered as the partial product as it will depend only on n or equivalently the modulo $2^n + 1$.According to (15) a constant one is added to each modulo $2^n + 1$ addition of two diminished-1 operands.Therefore,the aggregate sum of $\lfloor n/3 \rfloor + 5$ is added in the modulo $2^n + 1$ summation of $\lfloor n/3 \rfloor + 6$ partial products.To negate this effect, $\lfloor n/3 \rfloor + 5$ is subtracted from (31),which gives,

$$|Z|_{2^n+1} = \frac{\sum_{i=0}^{\lfloor n/3 \rfloor} PP_i + X + Y + K1 + K2 + K3}{\lfloor n/3 \rfloor + 6} - \sum_{i=0}^{\lfloor n/3 \rfloor} 2^{3i+2} - 4$$

$$= \frac{\sum_{i=0}^{\lfloor n/3 \rfloor} PP_i + X + Y + K1 + K2 + K3}{\lfloor n/3 \rfloor + 6} + \sum_{i=0}^{\lfloor n/3 \rfloor} 2^{3i} + \sum_{i=0}^{\lfloor n/3 \rfloor} 2^{3i+1}. \quad \text{_____(32)}$$

The term $\sum_{i=0}^{\lfloor n/3 \rfloor} 2^{3i} + \sum_{i=0}^{\lfloor n/3 \rfloor} 2^{3i+1}$ will be merged with K3 and the final product is given by,

$$|Z|_{2^n+1} = \frac{\sum_{i=0}^{\lfloor n/3 \rfloor} PP_i + X + Y + K1 + K2 + K3'}{\lfloor n/3 \rfloor + 6} \quad \text{___(33)}$$

Where $K3' = \sum_{i=0}^{\lfloor n/3 \rfloor} 2^{3i} + \sum_{i=0}^{\lfloor n/3 \rfloor} 2^{3i+1} \cdot \bar{s_i} + \sum_{i=0}^{\lfloor n/3 \rfloor} s_i \cdot 2^{3i+2}$.

By suitably modifying the dynamic bias corresponding to the most significant signed multiplier digit,i.e.,$KD_{\lfloor n/3 \rfloor}$, the aggregate sum of $\lfloor n/3 \rfloor + 1$ bias K_i will be reduced to three n-bit words for the cases of mod(n , 3) = 0 and 1 as well. Thus,the total number of partial products to be summed is $\lfloor n/3 \rfloor + 6$.The CSA tree for MPPA of modulo $2^n + 1$ multiplier is shown in Figure-8 for n = 8.

V. SIMULATION RESULTS FOR N = 8, 16, 32 AND 64 BITS

Fig.9. Simulation for n = 8 for 2^n - 1and 2^n + 1 multipliers.

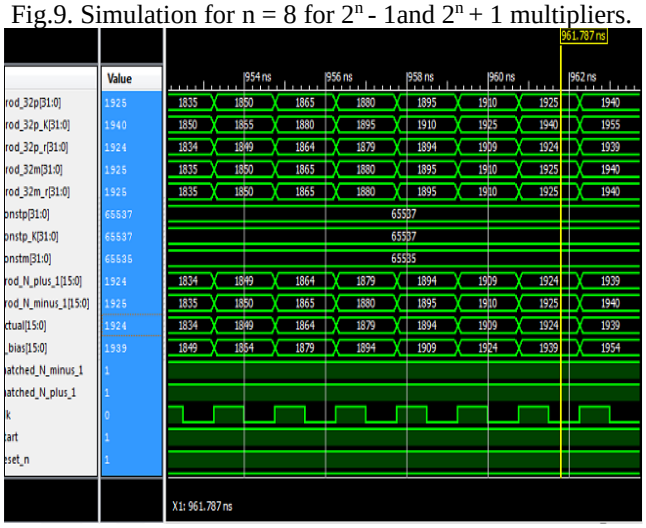


Fig.10. Simulation for n = 16 for 2^n - 1and 2^n + 1 multipliers.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

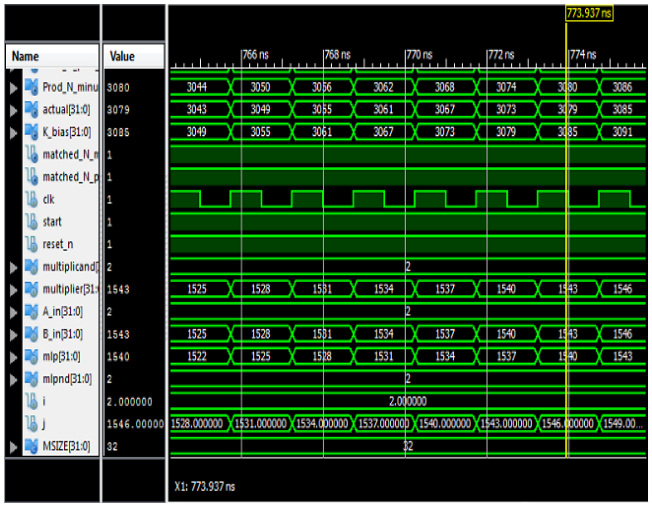


Fig.11. Simulation for n = 32 for 2^n - 1and 2^n + 1 multipliers.

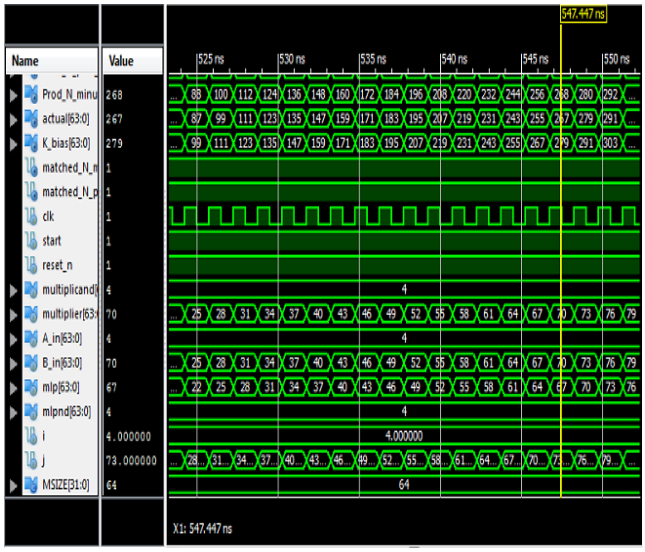


Fig.12. Simulation for n = 64 for 2^n - 1and 2^n + 1 multipliers.

VI. CONCLUSION

The modulo multipliers using RNS employing radix-8 Booth encoding scheme had been proposed. The modulo-reduced partial products were produced without having bias in the proposed modulo 2^n – 1 multiplier while the ineluctable bias in the proposed modulo 2^n + 1 multiplier was successfully expressed as three n-bit words by using the number theoretic properties of modulo arithmetic.

REFERENCES

1. 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.
2. R. Conway and J. Nelson, “Improved RNS FIR filter architectures,” IEEE Trans, Circuits and Syst. II, EXP. Briefs, vol. 51, no. 1, pp. 26-28,Jan 2004.
3. J. Ramirez, U. Meyer-Base, A. Garcia and A. Lloris, “Design and Implementation of RNS-based adaptive filters,” in Proc. 13th Int. Conf. Field Programmable Logic Applications, Lisbon, Spain, Sep. 2003, pp. 1135 – 1138.
4. H. Nozaki, M. Motoyama, A. Shimbo, and S. Kawamura, “Implementation of RSA algorithm based on RNS montgomery multiplication,”, in Proc. Workshop Cryptographic Hardware Embedded Syst., Paris, France, May 2001, pp. 364-376.
5. J.C. Bazard and L. Imbart, “A full RNS implementation of RSA, ” IEEE Trans. Comput., vol. 53, no. 6, pp. 769-774, Jun. 2004.
6. D.M. Schinianakis et al, “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.
7. Ramya Muralidharan, “Area-Power Efficient Modulo 2^n – 1 and Modulo 2^n + 1 Multipliers for {2^n – 1 , 2^n , 2^n + 1} Based RNS,”, IEEE Trans. Circuits Syst I., vol. 59, no. 10, Oct 2012.
8. V. Paliouras and T. Stouraitis, “Multiplication architectures for RNS processors,” IEEE Trans. Circuits Syst. II, Analog. Digit. Signal Process.,vol. 46, no. 8, pp. 1041-1054, Aug. 1999.
9. I. Kouretas and V. paliouras, “RNS multi-voltage low power multiply-and unit,” in Proc. 17th IEEE Int. Conf. Electronics, Circuits Systems, Athens, Greece, Dec. 2010, pp. 9-12.
10. G.C. Cardarilli, A.D. Re, A. Nannarelli, and M, Re, “Low power and low leakage implementation of Rns FIR filters,” in Proc. 39th Asilamar Conf. Signals, Syst. Comput., Pacific Grove, CA, Nov. 2005, pp. 1620-1624.
11. Z. Wang, G.A. Jullien, and W.C. Miller, “An algorithm for multiplication modulo (2n – 1),” in Proc. 39th IEEE Midwest Symp. Circuits Syst., Ames, 1A, Aug. 1996, pp. 1301-1304.
12. R. Zimmermann, “Efficient VLSI implementation of modulo (2n ± 1) addition and multiplication,” in Proc. 14th IEEE Symp. Computer Arithmetic, Adelaide, Australia, Apr. 1999, pp. 158 – 167.
13. C. Efstathiou, H. T. Vergos , and D. Nikolos, “Modified Booth modulo 2^n – 1 multipliers,” IEEE Trans. Comput., vol. 53, no. 3, pp. 370 – 374, Mar. 2004.
14. Z. Wang, G. A. Jullien, and W. C. Miller, “An efficient tree architecture for modulo 2n + 1 multiplication, ” J. VLSI Signal Process, vol. 14, no. 3, pp. 241 – 248, Dec. 1996.
15. C. Efstathiou, H. T. Vergos, G. Dimitrakopoulos, and D. Nikolos, “Efficient dimished – 1 modulo 2^n + 1 multipliers,” IEEE Trans. Comput., vol. 54, no. 4, pp. 491 – 496, Apr. 2005.
16. Y. Ma, “A simplified architecture for modulo (2^n + 1) multiplication,” IEEE Trans. Comput., vol. 47, no. 3, pp. 333 – 337, Mar. 1998.
17. 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.
18. J. W. Chen and R. H. Yao, “Efficient modulo 2^n + 1 multipliers for diminished-1 representation,” IET Circuits. Divices Syst., vol. 4, no. 4, pp. 291-300, Jun. 2010.
19. L. Leibowwitz, “A simplified binary arithmetic for the fermat number transform,” IEEE Trans. Acoustics, Speech Signal Process., vol. 24, no. 5, pp. 356-359, Oct. 1976.
20. H. T. Vergos and C. Efstathiou, “Design of efficient modulo 2n + 1 multipliers,” IET Comput. Digital Tech., vol. 1, no. 1, pp. 49-57, Jan. 2007.
21. G. W. Bewick, “Fast Multiplication Algorithms and Implimentation,” Ph.D. dissertation, Stanford Univ, Stanford, CA, 1994.
22. 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.
23. M. J. Flynn and S. F. Oberman, “Advanced Computer Arithmetic Design,” New York: Wiley, 2001.
24. R. Muralidharan and C. H. Chang, “Radix-8 Booth encoded modulo 2^n – 1 multipliers with adaptive delay for high dynamic range residue number system,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 58, no. 5, pp. 982-993, Jun. 2011.
25. R. Muralidharan and C. H. Chang, “Hard multiple generator for higher radix modulo 2^n – 1 multiplication,” in Proc.12th Int. Symp. Integr. Circuits, Singapore, Dec. 2009, pp. 546-549.
26. R. Muralidharan and C. H. Chang, “Fast hard multiple generators for radix-8 Booth encoded nodulo 2^n-1 and modulo 2^n + 1 multipliers,” in Proc. 2010 IEEE Int. Symp. Circuits Syst., Paris, France, May. 2010.
27. L. K. Kalampoukas et al., “High – Speed parallel – prefix modulo (2^n- 1) adders,” IEEE Trans. Comput., vol. 49, no. 7, pp. 673 – 680, Jul. 2000.
28. G. Dimitrakopoulos et al., “New architectures for modulo 2^n – 1 adders,” in Proc. 12th IEEE Int. Conf. Electronics. Circuits Syst., Gammarth, Tunisia, Dec. 2005, pp. 1 – 4.
29. H. T. Vergos, C. Efstathiou, and D. Nikolos, “Dimished – one modulo 2^n + 1 adder desigh,” IEEE Trans. Comput., vol. 51, no. 12, pp. 1389 – 1399, Dwc. 2002.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

30. H. T. Vergos and C. Efstathiou, "Efficient modulo 2^n+ adder architectures," VLSI J. Integr., vol. 42, no. 2, pp. 149 – 157, Feb. 2009.

Sivannarayana Amallapudi was born at Ravinuthala, Prakasam (Dist), AP, India. He attained his B -Tech from Electronics & Communication Engineering from Nalanda Institute of Engineering and Technology,affiliated to JNTU,Kakinada, Guntur ,AP, India. He is pursuing M-Tech from VLSI Engineering in Koneru Lakshmaiah University, Vijayawada, AP,India.His research areas are Digital Signal Processing and VLSI field.

Harikishore.Kakarla was born in Vijayawada, Krishna (Dist.), AP, India. He received B.Tech. in Electronics & Communication Engineering from St. Johns College of Engg.&Tech,Kurnool(Dist.,), AP, India, M.Tech from G. Pulla Reddy Engineering College, Kurnool, AP, India. He is pursuing Ph.D in the area of VLSI in KL University, Vijayawada, AP, India. He is working as Assistant Professor for Department of Electronics & Communication Engineering, KL University,Vijayawada, AP, India. He has published one National Conference.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.