K.V.GANESH ,T.SUDHA RANI, P.N.VENKATESWARA RAO, K.VENKATESH /
International Journal of Engineering Research and Applications (IJERA)
ISSN: 2248-9622 www.ijera.com Vol. 2, Issue 3, May-Jun 2012, pp.2734-2740

Constructing a low power multiplier using Modified Booth Encoding Algorithm in redundant binary number system

K.V.GANESH[*] ,T.SUDHA RANI[**], P.N.VENKATESWARA RAO[***], K.VENKATESH[***]
*(Department of ECE ,JNTUK,Kakinada)
**(Department of ECE,JNTUA,Anantapur)
***(Department of ECE,JNTUK, Kakinada)
****(Department of ECE ,JNTUK,Kakinada)

Abstract-- This paper introducing a novel technique called as redundant binary booth algorithm. The redundant binary in the design of high-speed digital multipliers is beneficial due to its high modularity and carry-free addition. Generally, in a high radix modified Booth encoding algorithm the partial products are reduced in multiplication process. But it yields complexity in producing in generation of hard multiples. Therefore booth encoding scheme along with redundant binary scheme solve this problem by using Booth encoder, RB partial product generator, RB partial product accumulator, RB to NB converter stages. In this process after booth encoding the two booths encoded digits are polarized in to differential pair to restore the effective RB partial product reduction rate without the NB to RB conversion over head. A new Booth encoding algorithm is presented in this project to simplify the generation of hard multiples and reduce the number of RB partial products without introducing any form of correction vector. The proposed algorithm binds two adjacent Booth encoders to compose an RB partial product by exploiting the RB coding. The common bit of the two adjacent Booth encoders is used as an enabler for the polarization of two equally weighted partial product bits. As the formation of an RB partial product digit is analogous to the charge sharing of two oppositely charged atoms in a covalent bond, we name the algorithm the covalent redundant binary Booth encoding.

IndexTerms—Arithmetic circuit, Booth encoding algorithm, digital multiplier, energy-delay product, redundant binary adder (RBA).

I. INTRODUCTION

The digital multiplier is a ubiquitous arithmetic unit in microprocessors, digital signal processors, and emerging media processors [1]–[4]. It is also a kernel operator in application- specific data path of ideo and audio codecs, digital filters, computer graphics, and embedded systems [5]–[8]. Compared with many other arithmetic operations, multiplication is time-consuming and power hungry. The critical paths dominated by digital multipliers often impose a speed limit on the entire design. Hence, VLSI design of high-speed multipliers, with low energy dissipation, is still a popular research subject. Redundant binary (RB) representation is one of the signed digit representations first introduced by Avizienis [9] in 1961 for fast parallel arithmetic .This new arithmetic was applied for fast multiplication by Takagi *et al.* [10] and implemented in VLSI by Edamatsu *et al.* [11]. The RB addition is carry-free, making it a promising substitute for two's complement multi-operand addition in a tree-structured multiplier [12]. Similar to a normal binary (NB) multiplier, an RB multiplier is anatomized into three stages and consists of four modules: the Booth encoder, RB partial product generator (also known as decoder), RB partial product accumulator, and RB-to-NB converter [11], [13]–[17]. The latter is required mainly for communicating the result to the peripheral devices which are largely designed based on the NB number system. The communications among RB adders across different stages of RB partial product summing tree are simpler than those of the full adders in a carry-save adder tree. In addition, the reduction rate of the redundant binary adder (RBA) summing tree is binary logarithmic to the number of RB partial products, which is particular beneficial to the generic power-of-two word size in computing. Booth encoder and partial product generator affect the efficiency of the partial product generation. The number of partial products that can be saved by this stage impacts the cost, performance, and power consumption of the RB summing tree and the multiplier as a whole. Although the number of partial products can be reduced with a high-radix Booth encoder, the number of hard multiples that are expensive to generate also increases simultaneously [12]. In conventional RB multiplier design, a modified Booth encoding algorithm in NB regime is employed to reduce the number of partial products, and then pairs of NB partial products are encoded to form RB partial products. In this process, an additional constant binary vector is introduced to compensate for the aggregate errors resulting from both the RB and Booth encodings [13], [14], [16]. This correction vector incurs hardware overhead in the RB summing tree and, to a certain extent, offsets the regularity of the layout and increases switching activities. As 8-, 16-, 32-, and 64-b operands are pervasively used in application-specific data paths and multimedia and very long instruction word(VLIW) processors which focuses on power-of-two word-length RB multipliers to exploit the binary logarithmic partial product reduction rate of the RBA summing tree. In order to overcome the overheads of existing Booth encoding algorithms, covalent redundant binary Booth encoding was used. The covalent redundant binary Booth multiplier circuits have been enhanced with a different RB coding and more efficient converter. The proposed method overcomes the hard multiple generation problem of NB Booth encoders without incurring any correction vector. Compared with the

RB Booth encoder, CRBBE generates the RB partial products more efficiently by consuming two RB digits for every RB partial product it generated. Consequently, encoder and decoder are less complex for the same radix the algorithm used for Booth encoder and decoder .

II. ISSUES OF BOOTH ENCODING ALGORITHMS FOR RB MULTIPLICATION

In fast digital multiplier design, modified Booth encoding algorithm is an efficient way to reduce the number of partial products by grouping consecutive bits in one of the two operands to form the signed multiples [21]. The operand that is Booth encoded is called the multiplier and the other operand is called the multiplicand. In this section, two major issues on using the modified Booth encoding algorithm for RB multiplication and some existing solutions are discussed.

A. Hard Multiples Problem

When modified Booth encoding [21] is applied to two's complement number, it is known as normal binary Booth Encoding (NBBE). In radix-r Booth-k encoding($r=2^k$), a signed digit c_i is generated from adjacent binary bits $Y_{k(i+1)-1}$ $Y_{k(i+1)-2}$……$y_{ki+1}y_{ki}$ and a borrow bit y_{ki-1} as follows:

$$c_i = -2^{k-1} \times y_{k(i+1)-1} + \sum_{j=2}^{k} 1 \times 2^{k-j} \times y_{k(i+1)-j} + y_{ki-1},$$

for i = 0, 1, 2, ….. $\left\lceil \frac{N}{k} \right\rceil - 1$

where k is an integer, $\lceil \alpha \rceil$ denotes the smallest integer value larger than or equal to α ,N is the word length of the multiplier Y and $y_{-1}=0$.

As the radix number r of Booth-k encoder increases, the number of Booth encoded digits and hence the number of partial products decreases to approximately 1/k of the original number. However, as the number of multiples increases with the radix to 2^k+1, the number of hard multiples also increases simultaneously. A hard multiple refers to a multiple that is not a power of two and thus cannot be obtained easily by simple shifting and/or complementation.

The multiplier is partitioned into 4-b groups with an overlapping borrow bit between two adjacent groups. Each group is encoded in parallel to generate a select signal from the set $\{\pm 4M, \pm 3M, \pm 2M, \pm M, 0\}$. c_iM refers to the select signal for the partial product c_iX, where X is the multiplicand. The partial product 3X is a hard multiple, which can only be obtained by adding X and 2X and by a carry propagation adder (CPA). The existence of hard multiple increases the latency of the multiplier as a whole because the generation of the partial products will not be accomplished until all the hard multiples are produced. Therefore, the advantage of using Booth encoding of radix-8 and above has been greatly offset because of the criticality of generating the hard multiples and the complexity of the decoding logic.

Multiplier Bits $y_{3i+2}y_{3i+1}y_{3i}(y_{3i-1})$	c_iM	Multiplier Bits $y_{3i+2}y_{3i+1}y_{3i}(y_{3i-1})$	c_iM
000(0)	+0	100(0)	$-4M$
000(1)	$+M$	100(1)	$-3M$
001(0)	$+M$	101(0)	$-3M$
001(1)	$+2M$	101(1)	$-2M$
010(0)	$+2M$	110(0)	$-2M$
010(1)	$+3M$	110(1)	$-M$
011(0)	$+3M$	111(0)	$-M$
011(1)	$+4M$	111(1)	−0

Table : Radix-8 normal binary booth encoding (NBBE-3)

To speed up the generation of hard multiples in high-radix Booth encoding, a partially redundant biased Booth encoding (PRBBE) algorithm was used. The following figure depicts the generation and negation of 3X hard multiple. It is generated in a partially redundant form by using a series of small length adders (4-b). The carry bit of each small length adder is not propagated but brought forward to the partial product summing tree. However, when the 3X multiple is negated, both the sum and the carry vectors need to be complemented and a "1" is added at the least significant bit (LSB) position. Therefore, the long strings of zeros between carries become strings of ones in the negative multiple. A properly selected biasing constant is introduced to revert the strings of one's into strings of zeros. The "1"s can be combined with the carry and sum bits to form a single compensation vector. The biasing constant of each such partial product introduces an extra compensation vector to the partial product summing tree.[4]

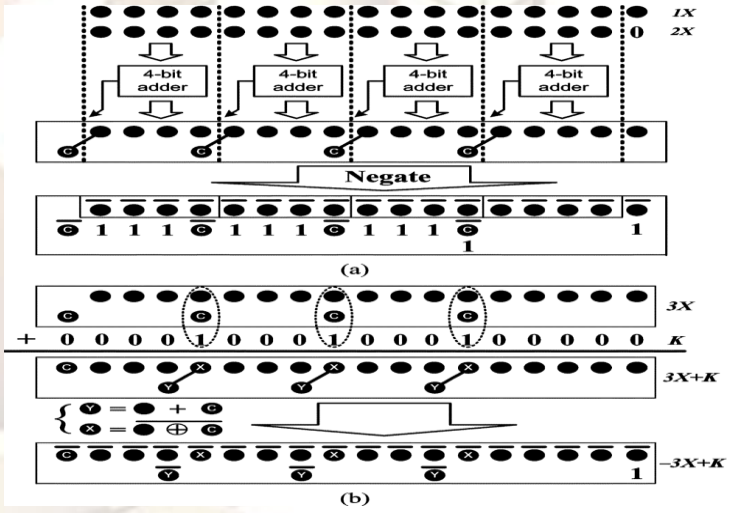


Figure: 3X hard multiple generation and negation in partially redundant form

B. Negative Multiples and NB-to-RB Partial Products Conversion Problem:

Negation in two's complement arithmetic requires carry propagation addition, negative partial product is more efficiently generated by the bit inversion of the multiplicand followed by the insertion of a "1" at its LSB position in the partial product summing tree. Therefore, one additional partial product row is generated to complete the two's complement negation of partial products for the negative multiples. Furthermore, to accumulate the partial products in an RBA summing tree, the NB partial products generated by

NBBE and PRBBE need to be converted to RB partial products. An NB number can be encoded into RB representation using either sign magnitude, positive-negative, or positive-negative-complement codings. Positive-negative-complement coding is adopted here, but it is also valid for other binary coding of signed digit set since the architecture designed for one code converter can be easily adapted to the other. In RB multiplication, the summation of two n-bit NB partial products A= $(a_{n-1}\ a_{n-2}\ \ldots\ldots\ldots\ a_0)_2$ and B= $(b_{n-1}\ b_{n-2}\ \ldots\ldots\ldots\ b_0)_2$ can be combined into a single -digit RB number R by

R=A+B=A-(-B)

$$\text{Since } -B = \overline{B} + 1,$$
$$R = A - (\overline{B} + 1) = A - \overline{B} - 1$$
$$= \left(-2^{n-1}a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i\right) - \left(-2^{n-1}\overline{b_{n-1}} + \sum_{i=0}^{n-2} 2^i \overline{b_i}\right) - 1$$
$$= -2^{n-1}(a_{n-1} - \overline{b_{n-1}}) + \sum_{i=0}^{n-2} 2^i (a_i - \overline{b_i}) - 1.$$

Coding (r^+, r^-)	RB Digit r
(0, 0)	$\bar{1}$
(0, 1)	0
(1, 0)	0
(1, 1)	1

Table : Positive-Negative-complement coding

As shown in Table, an RB digit **r** can be encoded with two binary bits $\mathbf{r}^+$ and $\mathbf{r}^-$ by

$$r = (r^+, r^-) = r^+ - \overline{r^-}$$

$$\text{where } r^+, r^- \in \{0,1\}, \text{ and } r \in \{0,1,\bar{1}\}.$$

Therefore, according to above equation, the terms $(a_i - \overline{b_i})$ can be encoded as $r_i = (a_i, b_i)$. To eliminate the hardware required for sign extension, the most significant digit term can be simply negated as - (a_{n-1}, b_{n-1}).

$$-(r^+, r^-) = -r^+ + \overline{r^-} = (\overline{r^-}, r^+).$$

Since the positive-negative-complement coding is symmetric, r^+ and r^- is commutative and

$$(\overline{r^-}, r^+) \equiv (\overline{r^+}, r^-).$$

Therefore, R can be coded as follows:

$$R = (\overline{a_{n-1}}, \overline{b_{n-1}})(a_{n-2}, b_{n-2}) \ldots (a_1, b_1)(a_0, b_0) + (0,0)$$

From the above equation, it is clear that every RB partial product row thus composed requires one correction constant $(0,0) \equiv \bar{1}$ to be added by an RBA at its LSB position. All of the correction constants generated from the RB partial products, together with those constants from the negative multiples, can be accumulated to form a new RB partial product, collectively called the RB correction vector.

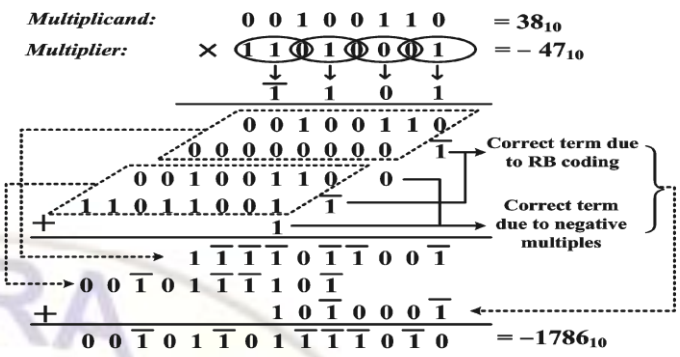


Figure : Illustration of the correction vector generation on an 8*8-b multiplication with NBBE-2.

From the above figure, it can be seen that NBBE-2 (radix-4 NB Booth encoding) generates three instead of two RB partial products for an 8*8-b multiplication. Owing to the absence of hard multiples, NBBE-2 is attractive especially for the short operand length multiplication. The additional delay required to add an extra partial product row critically slows down the short operand length multiplier due to the relatively lower number of adder stages in its partial product summing tree.

The RB correction vector incurs additional hardware for its accumulation. It can even increase the number of stages of the summing tree, if the word length of the multiplier is exactly 2^n, such as the 8-b and 16-b multipliers in application-specific data paths of multimedia and wireless applications and the multipliers for single extended and double extended floating point numbers, whose effective mantissa are 32 and 64 b, respectively. Consequently, the power dissipation and worst case delay are also degraded by the inclusion of this correction vector.[5]

C. Two's Complementation Method (TCM):

TCM is used to resolve the extra correction vector problem associated with the NBBE-2 algorithm. The TCM algorithm uses a divide-and-conquer approach to perform the two's complement conversion so that five signed partial products $\{0, \pm 1X, \pm 2X\}$ are originated for selection. In this way, the correction vector due to the negative multiples in two's complement arithmetic can be eliminated.

If TCM is used for the design of RB multiplier, the RB coding induced compensation constants can also be similarly circumvented. With TCM algorithm, the RB multiplier achieves exactly $\lceil N/4 \rceil$ RB partial products as opposed to $\lceil N/4 \rceil + 1$ in NBBE-2 multiplier. Besides, the multiplier is modular and more regularly structured. However, the worst case delay of the TCM algorithm is logarithmically proportional to the operand lengths (OlogN).Comparing with the constant delay time of conventional Booth encoding

algorithms, the dependency of speed on word length of TCM algorithm is a limiting factor for large integer multiplication.[6]

D. RB Booth Encoding (RBBE):

Hard multiples could be obtained from the differences of two simple power-of-two multiples. In radix-16 RB Booth encoding, the multiplier bits are $y_{4i+3}y_{4i+2}y_{4i+1}\ y_{4i}y_{4i-1}$ and each of the original hard multiples selected by $\pm 3M, \pm 6M \text{ and } \pm 7M$ are replaced by $\pm(4X - 1X), \pm(8X - 2X) \text{ and } \pm(8X - 1X)$ respect ively. The partial products generated in this way conform to the format of the RB coding. The only exception is that,

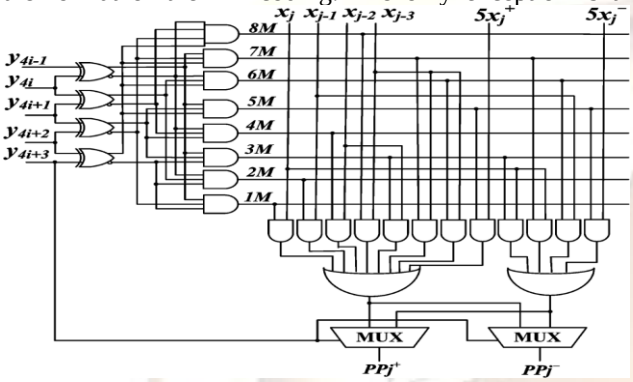


Figure:Radix-16 RBBE and its partial productgenerator.

hard multiples selected by $\pm 5M$ Cannot be readily generated in this manner, a simple carry-free RB adder is used to add 4X and X. The advantage of this method is the correction vector due to the two's complement arithmetic and the RB coding has been completely eliminated. Comparing with NBBE, the ease of generating the hard multiples by RBBE, to a certain extent has been offset by its complex circuitry.

High-radix RBBE requires high fan-in gates in the partial product generator circuit. Since the circuit for each digit of the RB partial product will be duplicated in a large number, the overhead of high fan-in gates is more prominent in long operand length multipliers. Besides, as only one Booth encoded digit is consumed for one RB partial product, half of the binary bits representing an RB partial product generated from a simple power-of-two multiple in the RBBE are filled with "0"s, which is rather inefficient.

RADIX-16 REDUNDANT BINARY BOOTH ENCODING (RBBE-4)

Multiplier Bits	Multiple		Multiplier Bits	Multiple	
	$+M$	$-M$		$+M$	$-M$
0 0 0 0 (0)	0	0	1 0 0 0 (0)	0	$8M$
0 0 0 0 (1)	M	0	1 0 0 0 (1)	M	$8M$
0 0 0 1 (0)	M	0	1 0 0 1 (0)	M	$8M$
0 0 0 1 (1)	$2M$	0	1 0 0 1 (1)	$2M$	$8M$
0 0 1 0 (0)	$2M$	0	1 0 1 0 (0)	$2M$	$8M$
0 0 1 0 (1)	$4M$	M	1 0 1 0 (1)	0	$5M$ *
0 0 1 1 (0)	$4M$	M	1 0 1 1 (0)	0	$5M$ *
0 0 1 1 (1)	$4M$	0	1 0 1 1 (1)	0	$4M$
0 1 0 0 (0)	$4M$	0	1 1 0 0 (0)	0	$4M$
0 1 0 0 (1)	$5M$ *	0	1 1 0 0 (1)	M	$4M$
0 1 0 1 (0)	$5M$ *	0	1 1 0 1 (0)	M	$4M$
0 1 0 1 (1)	$8M$	$2M$	1 1 0 1 (1)	0	$2M$
0 1 1 0 (0)	$8M$	$2M$	1 1 1 0 (0)	0	$2M$
0 1 1 0 (1)	$8M$	M	1 1 1 0 (1)	0	M
0 1 1 1 (0)	$8M$	M	1 1 1 1 (0)	0	M
0 1 1 1 (1)	$8M$	0	1 1 1 1 (1)	0	0

* Hard multiples.

Table: Radix-16 redundant binary booth encoding (RBBE4)

To overcome the problem of generating hard multiples in high-radix Booth encoding, N. Besli et al. noticed that some hard multiples can be obtained by the differences of two simple (power-of-two) multiples. The partial products so generated conform to the format of positive -negative RB coding. This distinguishing Booth encoding logic is RB Booth encoding. The table illustrates the RB Booth-4 encoding, where the original hard multiples of ±3*M*, ±6*M* and ±7*M* are replaced by ±(4*M*−*M*), ±(8*M*−2*M*) and ±(8*M*−*M*), respectively. The only exception is the hard multiple 5*M*, which cannot be readily produced in this manner. Therefore, additional hardware is necessary to generate this 5*M* multiple. simple RB adder to add 4*M* and 1*M*. It turns out that this RB adder is carry free and does not lie in the critical path of the RB Booth-4 encoder and PPG circuit. Compared to NBBE, the ease of generating the hard multiples in RBBE has been offset to certain extend, by its complex circuitry involving the use of high fan-in gates. In addition, the cost of high fan-in gates and their associated detriments are aggravated by the duplication of each digit in the RB partial products.

III.COVALENT REDUNDANT BINARY BOOTH ENCODING (CRBBE) ALGORITHM

Covalent redundant binary booth encoding (CRBBE) algorithm is used to simplify the generation of hard multiples and reduce the number of RB partial products without introducing any form of correction vector. This algorithm binds two adjacent Booth encoders to compose an RB partial product by exploiting the RB coding. The common bit of the two adjacent Booth encoders is used as an enabler for the polarization of two equally weighted partial product bits. As the formation of an RB partial product digit is analogous to the charge sharing of two oppositely charged atoms in a covalent bond, we name the algorithm the covalent redundant binary Booth encoding.

CRBBE-4 Algorithm:

CRBBE-4 is composed of two adjacent radix-4 Booth encoders. Its gate-level implementation is represented, where the sign and magnitude of the radix-4 Booth encoded digit d_i are represented with three binary bits, $\mathbf{sgn_i}$, $\mathbf{m_i^{(2)}}$, $\mathbf{m_i^{(1)}}$, and as follows:

$d_i = (-1)^{sgn(i)}(m_i^{(2)} + m_i^{(1)})$

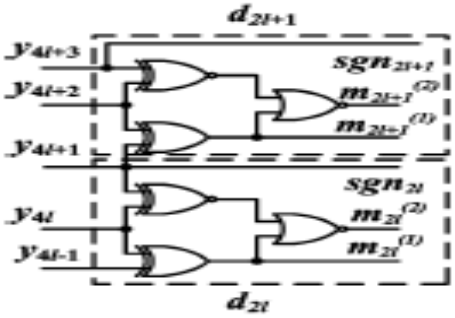


Figure : Two adjacent radix-4 Booth encoder

The above figure shows the '**l**' th slice of a radix-16 CRBBE-4 circuit for the generation of the control signals c_lM_l. The indexes '**i**' and '**l**' are related by i=2l. The lower encoder takes three consecutive bits $y_{2i+1}y_{2i}y_{2i-1} = y_{4l+1}y_{4l}y_{4l-1}$ from the multiplier to generate the magnitude bits $m_{2l}^{(2)}$ and $m_{2l}^{(1)}$ of d_i Its sign bit $sgn_i = y_{4l+1}$. The upper encoder takes the binary bits $y_{2i+3}y_{2i+2}y_{2i+1} = y_{4l+3}y_{4l+2}y_{4l+1}$ and generates the magnitude bit $m_{2l+1}^{(2)}$ and $m_{2l+1}^{(1)}$ of d_{i+1}. Its sign bit $sgn_{i+1} = y_{4l+3}$. All of these output signals are mapped to the polarization circuit. The control signals c_lM_l, generated are used to select the RB partial products correspond to the multiples c_lX.

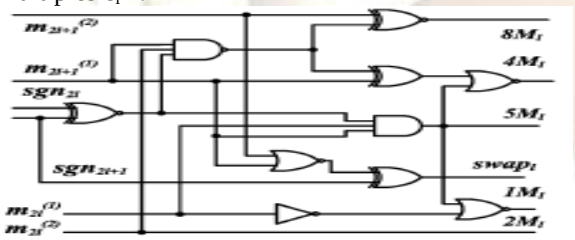


Figure : Polarization circuit

The polarization circuit performs the mapping (d_{i+1}, d_i) (p_l^+, p_l^-) .

The control signals $1M_l$, $2M_l$, $4M_l$ and $8M_l$ are computed as follows:

$$1M_l = m_{2l}^{(1)} \cdot \overline{5M_l}$$

$$2M_l = m_{2l}^{(2)}$$

$$4M_l = \left(m_{2l+1}^{(1)} \cdot m_{2l}^{(2)} \cdot (sgn_{2l+1} \odot sgn_{2l})} \odot m_{2l+1}^{(1)}\right) \cdot \overline{5M_l}$$

$$8M_l = \overline{m_{2l+1}^{(1)} \cdot m_{2l}^{(2)} \cdot (sgn_{2l+1} \odot sgn_{2l}) \odot m_{2l+1}^{(2)}}.$$

The 5M multiple is generated as

$$5M_l = (sgn_{2l+1} \odot sgn_{2l}) \cdot m_{2l+1}^{(1)} \cdot m_{2l}^{(1)}.$$

The control flag, swap is used to exchange p_l^+ and p_l^- in the partial product generator to negate the selected RB partial product. When d_{i+1} is 0, the sign bit of d_{i+1} is complemented before it is used as an active high swap flag to the RBPPG. Otherwise, the original sign of d_{i+1} is used as the swap flag. Therefore, the *swap* signal can be generated by:

$$swap_l = \overline{\left(m_{2l+1}^{(1)} + m_{2l+1}^{(2)}\right)} \oplus sgn_{2l+1}$$

IV. DESIGN OF CRBBE-4-BASED RB MULTIPLIER

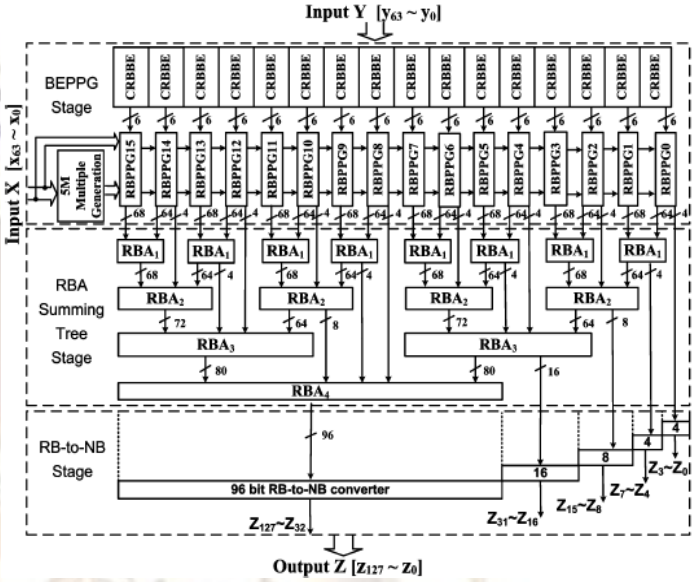


Figure: Block diagram of 64*64 RB multiplier

The block diagram of 64*64 consists of 3 stages:

(1)Booth encoder and partial product generator stage (BEPPG stage).

(2)Redundant binary adder summing tree stage (RBA summing stage).

(3)Redundant binary to NB conversion stage (RB-to-NB stage).

A. Booth encoder and partial product generator stage (BEPPG stage):

Booth encoder and partial product generator affect the efficiency of the partial product generation. The number of partial products that can be saved by this stage impacts the cost, performance, and power consumption of the RB summing tree and the multiplier as a whole. In the first stage, 16 CRBBE-4 slices are used to generate the control signals from the multiplier. The hard multiple 5X is generated. The multiplicand bits are shifted and selected into 16 rows of RB partial products in 16 slices of RBPPG.

Design architecture also eliminates the error-correction block used in the partial product accumulator. This will enable us to expand the number of bits in the multiplier to 64 bits, while keeping the number of adder stages to four. Similar methods can be followed in the design of any multiplier having the number of partial products as perfect Powers of two (64x64, 32x32, 16x16, etc). This design will accommodate RB encoding in such multipliers while enjoying the benefits of both lesser number of partial products with optimum number of adder stages.

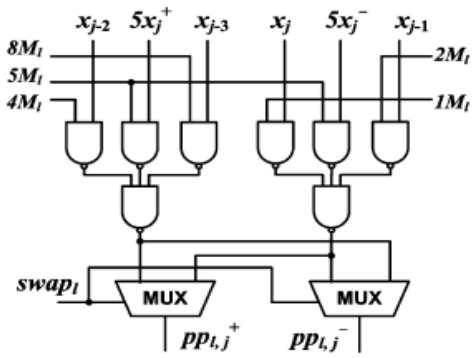


Figure:RB partial product generator (RBPPG) of CRBBE-4.

B. Redundant binary adder summing tree stage (RBA summing stage):

In the second stage, a 4-stage RBA summing tree is used to sum 16 RB partial products. Each RBA block contains 64 RB full adder (RBFA) cells and a varying number of RB half adder (RBHA) cells depending on where it is located. The RBA block in the **i**-th level, designated RBA_i (i=1 to 4) contains $\mathbf{2^{i+1}}$ RBHA cells in its most significant digit positions. Due to the positive-negative-complement coding, the second binary bit $\mathbf{pp_{l,j}^-}$ of the RB partial product generated from CRBBE-4 and RBPPG circuit should be inverted before it is input to the RBA. A preprocessing circuit is needed for each RB digit to avoid the inconsistent representations of "0" prior to the RBA summing tree stage. An important benefit of the coding format adopted in this design is that these preprocessing circuits can be completely eliminated due to its symmetry.

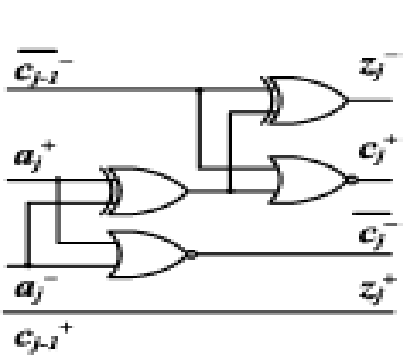


Figure : RB Half adder

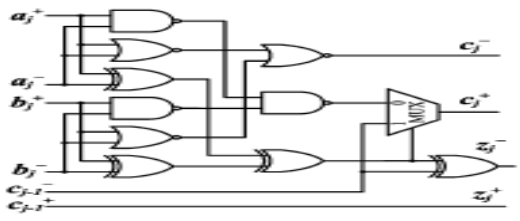


Figure : RB Full adder

C. Redundant binary to NB conversion stage (RB-to-NB stage):

An RB-to-NB converter converts the final accumulation result to NB representation. Due to the unequal delay profile of the final RB result bits, the conversion can be carried out in uneven groups of consecutive digits according to their arrival time. Groups of 4, 4, 8, 16 and 96 digits from the least significant digit position are evaluated concurrently. The first three groups of 4, 4, and 8 digits can be independently converted with ripple-carry adders to reduce the circuit complexity. The carry generation of the next group of 16 digits can be evaluated with a carry-look ahead adder as they do not depend on the final summation results in the RBA tree stage.

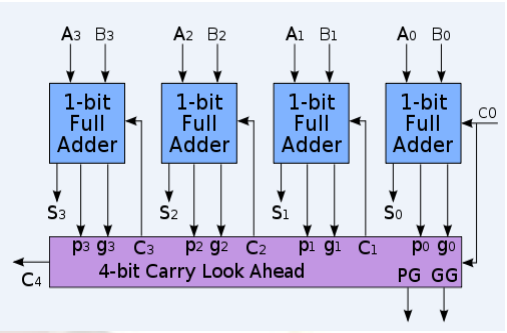
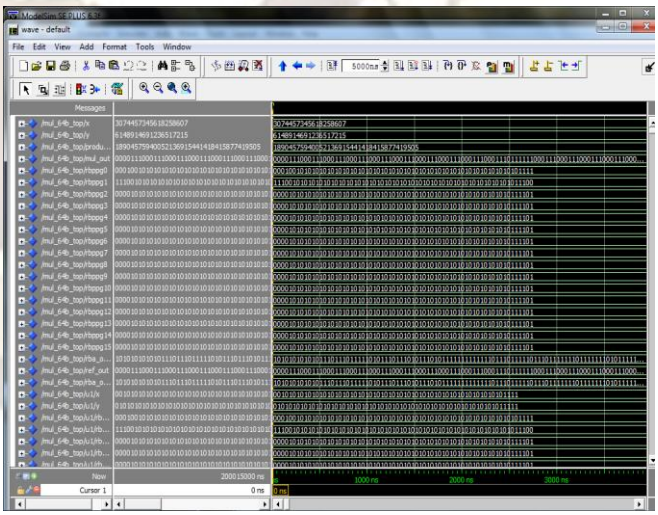


Figure 5.6: 4-Bit carrylook-ahead adder

The conversion speed of the RB-to-NB stage depends solely on the conversion time of the most significant 96-digit group. This group is converted with a hybrid carry-look ahead or carry-select adder since it is widely known as one of the most efficient structures for fast parallel adder design.

V. Simulation result

64 x 64 bit Multiplier final module:



VI .CONCLUSION

Hence, a high-speed and energy-efficient RB multiplier is designed based on new covalent RB Booth encoding algorithm. The idea is to polarize two adjacent Booth-encoded digits into a differential pair to restore the effective RB partial product reduction rate without the NB-to-RB conversion overhead. This method fully exploits the

characteristics of the positive–negative complement coding of RB number to directly generate an RB partial product from two adjacent Booth-encoded digits. Consequently, it shares the same advantages of RB Booth encoder for the ease of generating hard multiples and avoidance of error compensation vector, the two problems that are confronted by RB multiplier with normal binary Booth encoding.

VII.References

[1] C. Shi, W.Wang, L. Zhou, L. Gao, P. Liu, and Q. Yao, "32B RISC/DSP media processor: MediaDSP3201," SPIE Embedded Processors for Multimedia and Communications II, vol. 5683, pp. 43–52, Mar. 2005.

[2] B. Parhami, Computer Arithmetic Algorithms and Hardware Designs. New York: Oxford Univ. Press, 2000.

[3] G. W. Bewick, "Fast multiplication: Algorithms and implementation," Ph.D. dissertation, Stanford Univ., Stanford, CA, 1994.

[4] O. L. MacSorley, "High-speed arithmetic in binary computers," IRE Proc., vol. 49, pp. 67–91, Jan. 1961.

[5] P.Kornerup, "Digit-set conversions: Generalizations and applications," IEEE Trans. Comput., vol. 43, no. 5, pp. 622–629, May 1994.

[6] J.-Y. Kang and J.-L. Gaudiot, "A simple high-speed multiplier design," IEEE Trans. Comput., vol. 55, no. 10, pp. 1253–1258, Oct. 2006.

[7] Y. He, C. H. Chang, J. Gu, and H. A. H. Fahmy, "A novel covalent redundant binary Booth encoder," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'2005), Kobe, Japan, May 2005, vol. 1, pp. 69– V. Kantabutra, "A recursive carry-lookahead/carry-select hybrid adder," IEEE Trans. Comput., vol. 42, no. 12, pp. 1495–1499, Dec. 1993.