International Journal of Scientific Research in Computer Science, Engineering and Information Technology

© 2017 IJSRCSEIT | Volume 2 | Issue 6 | ISSN : 2456-3307

Realization of Redundant Binary Multiplier with Modified Partial Product Generator Using Verilog

V. Lakshma Reddy[1], H. Sudhakar[2], D. Ajay Kumar[2]

[1]M.Tech (VLSI Design), Department of ECE, SIR C.R.R. College of Engineering, Eluru,Andhra Pradesh, India

[2]Assistant Professor, Department of ECE, SIR C.R.R. College of Engineering, Eluru, Andhra Pradesh, India

ABSTRACT

Digital multipliers are widely used in arithmetic units of microprocessors, multimedia and digital signal processors. A redundant binary (RB) representation can be used when designing high performance multipliers due to its high modularity and carry-free addition. The conventional RB multiplier requires an additional RB partial product (RBPP) row, because an error-correcting word (ECW) is generated by both the radix-4 Modified Booth encoding (MBE) and the RB encoding. This incurs in an additional RBPP accumulation stage for the MBE multiplier. In this paper, a new RB modified partial product generator (RBMPPG) is proposed; it removes the extra ECW and hence, it saves one RBPP accumulation stage. Therefore, the proposed RBMPPG generates fewer partial product rows than a conventional RB MBE multiplier. Simulation results show that the proposed RBMPPG based designs significantly improve the area and power consumption when the word length of each operand in the multiplier is at least 32 bits.

Keywords : Binary Multiplier, Verilog, Partial Product Generator, ECW, RBMPPG, MBE, RBPP, RB MBE Multiplier, Normal Binary, MBE

I. INTRODUCTION

A normal binary (NB) multiplication by digital circuits includes three steps. In the first step, partial products are generated; in the second step, all partial products are added by a partial product reduction tree until two partial product rows remain. In the third step, the two partial product rows are added by a fast carry propagation adder. Two methods have been used to perform the second step for the partial product reduction. A first method uses 4-2 compressors, while a second method uses redundant binary (RB) numbers. Both methods allow the partial product reduction tree to be reduced at a rate of 2:1. The redundant binary number representation has been introduced by Avizienis to perform signed-digit arithmetic; the RB number has the capability to be represented in different ways. Fast multipliers can be designed using redundant binary addition trees. A RBPP row can be obtained from two adjacent NB partial product rows by inverting one of the pair rows.An additional error correcting word is also required by both the RB and booth encoding.Therefore,the number of RBPP accumulation stages required by power of two word length multiplier is given by

$$\text{NRBPPAS} = \log_2\left(\frac{N}{4}+1\right)$$
$$=n-1, \text{ if } N=2^n \qquad (1)$$

If the ECW can be removed, an RBPP accumulation stage is saved, so resulting in improvements of complexity and critical path delay for a RB multiplier. The number of accumulation stages in conventional 32-bit multiplier can be minimized by removing ECW.

Alternatively, a high-radix Booth encoding technique can reduce the number of partial products. However, the number of expensive hard multiples (i.e., a multiple that is not a power of two and the operation cannot be performed by simple shifting and/or complementation) increases too [14-16]. Besliet al. [16] noticed that some hard multiples can be obtained by the differences of two simple power-of-two multiplies. A new radix-16 Booth encoding (RBBE-4) technique without ECW has been proposed in [14]; it avoids the issue of hard multiples. A radix-16 RB Booth encoder can be used to overcome the hard multiple problem and avoid the extra ECW, but at the cost of doubling the number of

RBPP rows. Therefore, the number of radix-16 RBPP rows is the same as in the radix-4 MBE. However, the RBPP generator based on a radix-16 Booth encoding has a complex circuit structure and a lower speed compared with the MBE partial product generator [10] when requiring the same number of partial products.

II. REVIEW OF BOOTH ENCODING AND RB PARTIAL PRODUCT GENERATOR

2.1 Radix-4 Booth Encoding

Booth encoding has been proposed to facilitate the multiplication of two's complement binary numbers .It was revised as modified Booth encoding (MBE) or radix-4 Booth encoding [18]. The MBE scheme is summarized in Table I, where A $=a_{N-1}a_{N-2\ldots}a_2a_1a_0$ for the multiplicand, and B= $b_{N-1}b_{N-2}\ldots b_2b_1b_0$ for the multiplier. The multiplier bits are grouped in sets of three adjacent bits. The two side bits are overlapped with neighboring groups except the first multiplier bits group in which it is {b1, b0, 0}. Each group is decoded by selecting the partial product shown in Table I, where 2A indicates twice the multiplicand, which can be obtained by left shifting. Negation operation is achieved by inverting each bit of A and adding ‘1’ (defined as correction bit) to the LSB [10-13]. Methods have been proposed to solve the problem of correction bits for NB radix-4 Booth encoding (NBBE-2) multipliers. However, this problem has not been solved for RB MBE multipliers

Table 1. MBE SCHEME

$b_{2i+1}, b_{2i}, b_{2i-1}$	Operation
000	0
001	+A
010	+A
011	+2A
100	-2A
101	-A
110	-A
111	0

2.2 RB Partial Product Generator

As two bits are used to represent one RB digit, then a RBPP is generated from two NB partial products [1-6]. The addition of two N-bit NB partial products X and Y using two’s complement representation can be expressed as follows [6]

$$\begin{aligned} X+Y &= X-\overline{Y}-1 \\ &= \left(-x_N 2^N + \sum_{i=0}^{N-1} x_i 2^i\right) - \left(-\overline{y_N} 2^N + \sum_{i=0}^{N-1} \overline{y_i} 2^i\right) - 1 \\ &= -(x_N - \overline{y_N})2^N + \sum_{i=0}^{N-1} (x_i - \overline{y_i})\, 2^i - 1 \\ &= (X, \overline{Y}) - 1 \qquad (2) \end{aligned}$$

where $\overline{Y}$ is the inverse of Y, and the same convention is used in the rest of the paper. The composite number $(X,\overline{Y})$ can be interpreted as a RB number. The RBPP is generated by inverting one of the two NB partial products and adding -1 to the LSB. Each RB digit X_I belongs to the set $\{1, 0, \overline{1}\}$; this is coded by two bits as the pair (X_i^-, X_i^+)

Note that $\overline{1} = -1$. RB numbers can be coded in several ways. Table II shows one specific RB encoding [6], where the RB digit is obtained by performing $X_i^+ - X_i^-$.

Table 2. RB ENCODING

X_i^+	X_i^-	**RB DIGIT**(Xi)
0	0	0
0	1	$\overline{1}$
1	0	1
1	1	0

Both MBE and RB coding schemes introduce errors and two correction terms are required: 1) when the NB number is converted to a RB format, -1 must be added to the LSB of the RB number; 2) when the multiplicand is multiplied by -1 or -2 during the Booth encoding, the number is inverted and +1 must be added to the LSB of the partial product. A single ECW can compensate errors from both the RB encoding and the radix-4 Booth recoding. The conventional partial product architecture of an 8-bit MBE multiplier [5-6] is shown in Fig. 1, where b_p represents the bit position, p_{ij}^+ or p_{ij}^- is generated by using an encoder and decoder (Fig. 2) [10]. An N-bit CRBBE-2 multiplier includes N/4 RBPP rows and one ECW; the ECW takes the form as follows:

ECW = $E_{(N/4)}\,0\,F_{(N/4)}\,0\ldots\ldots 0E_{i2}0F_{i0}\ldots 0E_{12}0F_{10}$

where i represents the i th row of the RBPPs, $E_{i2}\in\{0,1\}$and $F_{I0}\in\{0, 1\}$. In F_{i0} , a -1 correction term is always required by RB coding. If F_{i0} also corrects the errors from the MBE recoding, then the correction term cancels out to 0. That is to say that if the multiplicand digit is inverted and added to 1, then F_{i0} is 0, otherwise F_{i0} is -1. The error-correcting digit E_{i2} is determined only by the Booth encoding:

$$E_{i2}=\begin{cases}0, & \textit{no negative encoding}\\ 1, & \textit{negative encoding}\end{cases}$$

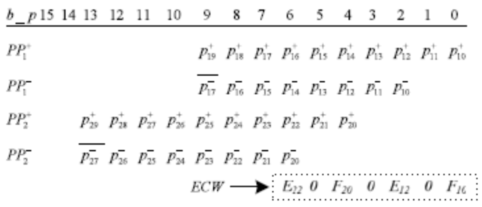


Figure 1. Conventional RBPP architecture for an 8-bit MBE multiplier

As shown in Fig. 1 the first RBPP row, i.e. PP_1 consists of the first partial product row PP_1^+ and the second partial product row PP_1^-

$PP_1^+ = p_{19}^+ p_{18}^+ \ldots .p_{10}^+$ and $PP_1^- = p_{19}^- p_{18}^- \ldots .p_{10}^-$ where p_{19}^+ and p_{19}^+ are the sign extension bits, so

$$p_{19}^{+} = \overline{p_{18}^{+}} \tag{5}$$

$$p_{18}^{+} = \overline{b_1}\,\overline{b_0}\cdot 0 + \overline{b_1}b_0\cdot a_7 + b_1\overline{b_0}\cdot\overline{a_7} + b_1b_0\cdot\overline{a_7}$$
$$= \overline{b_1}b_0\cdot a_7 + b_1\overline{a_7} \tag{6}$$

According to Eq. (2), the sign extension bit P^+_{29} is also the inverse of p_{28}^+.

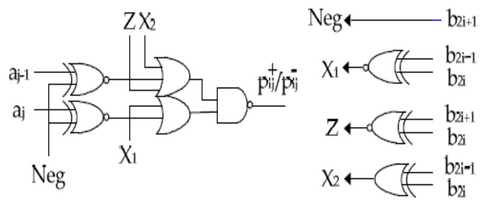


Figure 2. An encoder and decoder of MBE scheme

III. PROPOSED RB PARTIAL PRODUCT GENERATOR

A new RB modified partial product generator based on MBE (RBMPPG-2) is presented in this section; in this design, ECW is eliminated by incorporating it into both the two MSBs of the first partial product row (PP_1^+**)** and the two LSBs of the last partial product row($pp_{N/4}^-$**)**

3.1 Proposed RBMPPG-2

Figure 3 illustrates the proposed RBMPPG-2 scheme for an 8x8-bit multiplier. It is different from the scheme in Figure 1, where all the error-correcting terms are in the last row. ECW**1** is generated by PP_1 and expressed as

$$ECW_1 = 0\ E_{12}\ 0\ F_{12} \tag{7}$$

The ECW2 generated by PP_2 (also defined as an extra ECW) is left as the last row and it is expressed as:

$$ECW2=0\ \ E_{22}\ 0\ F_{20} \tag{8}$$

To eliminate a RBPP accumulation stage, ECW2 needs to be incorporated into PP1and PP2. As discussed in Section 2.2 for $F_{i0}$and as per Table I, F_{20} is determined by

$$F_{20}=\begin{cases}-1 & b5b4b3{=}000{,}001{,}010{,}011{,}111\\ 0 & b5b4b3{=}100{,}101{,}110\end{cases} \tag{9}$$

As per Table 1, when $b_5b_4b_3$ = 111,-0=0 can used. Therefore, F_{20} can be expressed as follows

$$F_{20}=\begin{cases}-1 & b5b4b3{=}000{,}001{,}010{,}011{,}\\ 0 & b5b4b3{=}100{,}101{,}110{,}111\end{cases} \tag{10}$$

By setting PP_2^+ to all ones and adding +1 to the LSB of the partial product, F20 can then be determined only by as follows:

$$F_{20}=\begin{cases}-1, & b5{=}0\\ 0, & b5{=}1\end{cases} \tag{11}$$

A modified radix-4 Booth encoding and a decoding circuit for the partial product PP_2^+ are proposed here ; an extra 3-input OR gate is then added to the design of [10] (Fig. 2). The three inputs of the additional OR gate are $\overline{b}_5$,$\overline{b}_4$ and $\overline{b}_3$ When $b_5b_4b_3$=111 it is clear that $\overline{b}_5\overline{b}_4\,\overline{b}_3$= 000, P_{2i}^+= 1, PP_2^+ is set to all ones. So, E_{22} and F_{20} in ECW_2 are now determined by $b_7b_6b_5$ without b_4,b_3. Although the complexity is slightly increased compared with the previous design (Figure 2), the delay stage remains the same.

In this work, $Q_{19}^+\,Q_{18}^+\,Q_{21}^-$ and Q_{20}^- are used to represent the modified partial products

$q_{2(-2)}^-$ and $q_{2(-1)}^-$ as follows

$$E_2=\begin{cases}E22, & F20{=}0\\ E22-1\,, & F20{=}-1\end{cases} \tag{12}$$

$$.\ q_{2(-2)}^- = q_{2(-1)}^- = \begin{cases}0, & F20{=}0\\ 1, & F20{=}-1\end{cases} \tag{13}$$

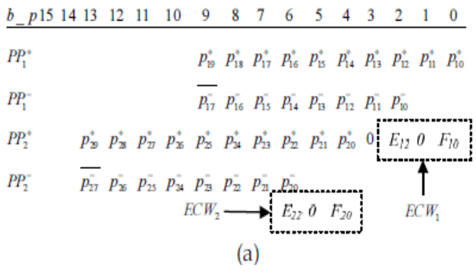


Figure 3. (a) the first new RBMPG-2 architecture

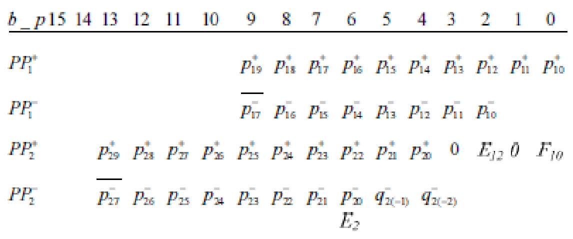
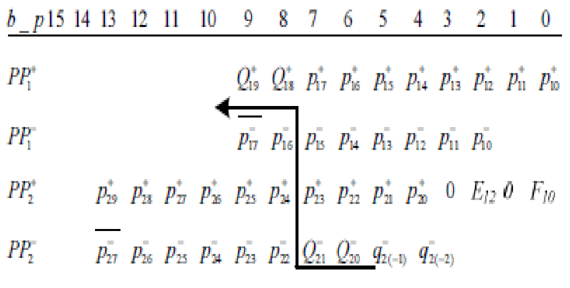


Figure 3. (b) further revised RBMPG-2 architecture



(c)

Figure 3. (c) the final proposed RBMPG-2 by totally eliminating ECW$_2$ and further combining E$_2$ into, Q_{19}^+ Q_{18}^+ Q_{21}^- and Q_{20}^-

Table 3.

$b_7b_6b_5$	$E_{22}F_{20}$	$E_2q_{2(-2)}^-q_{2(-1)}^-$	P_{21}^-	P_{20}^-
000	0 $\bar{1}$	$\bar{1}$11	0	0
001	0 0	000	a_1	a_0
010	0 $\bar{1}$	$\bar{1}$11	a_1	a_0
011	0 0	000	a_0	0
100	1 $\bar{1}$	011	$\overline{a0}$	1
101	1 0	100	$\overline{a1}$	$\overline{a0}$
110	1 $\bar{1}$	011	$\overline{a1}$	$\overline{a0}$
111	0 0	000	0	0

So the following three cases can be distinguished:1) when E$_2$=0,Q_{19}^+,Q_{18}^+,Q_{21}^- and Q_{20}^- remain unchanged as Q_{19}^+=P_{19}^+, Q_{18}^+ = P_{18}^+,Q_{21}^-=$P_{21}^-$$Q_{20}^-$=$P_{20}^-$ .2)when E$_2$=1 a 1 is added to P_{19}^+ $P_{18}^+$$P_{21}^-$ P_{20}^-.when E$_2$=-1 1 is substracted from $P_{19}^+$$P_{18}^+$$P_{21}^-$ P_{20}^-.The relationships between these partial product variables are summarized in Table 4.

Table 4.

$p_{19}^+p_{18}^+p_{21}^-p_{20}^-$	$Q_{19}^+Q_{18}^+Q_{21}^-Q_{20}^-$ E$_2$=0	$Q_{19}^+Q_{18}^+Q_{21}^-Q_{20}^-$ E$_2$=1	$Q_{19}^+Q_{18}^+Q_{21}^-Q_{20}^-$ E$_2$=-1
0100	0100	0101	0011
0101	0101	0110	0100
0110	0110	0111	0101
0111	0111	1000	0110
1000	1000	1001	0111
1001	1001	1010	1000
1010	1010	1011	1001
1011	1011	1100	1010

Therefore, as per table 4 the logic functions of Q_{19}^+ Q_{18}^+ Q_{21}^- and Q_{20}^- can be expressed and Boolean functions of Q_{19}^+ Q_{18}^+ Q_{21}^- and Q_{20}^- can be Further minimized as below

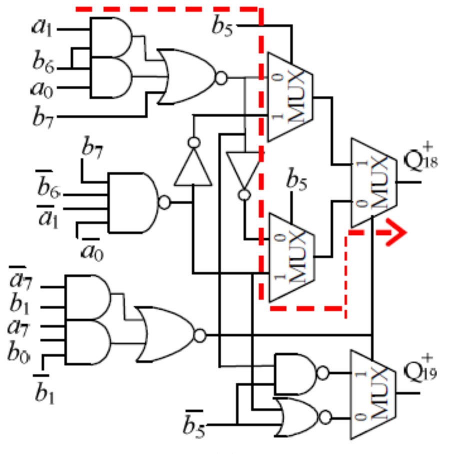
$$Q_{19}^+ = \overline{\overline{b_1}b_0a_7 + b_1\overline{a_7}}\left(\overline{\overline{b_5}\cdot\overline{b_7+b_6a_0+b_6a_1}}\right) + \left(\overline{b_1}b_0a_7+b_1\overline{a_7}\right)\cdot b_5\cdot b_7\overline{b_6}\,\overline{a_1}\;\overline{a_0}, \tag{14}$$

$$Q_{18}^+ = \overline{\overline{b_1}b_0a_7+b_1\overline{a_7}}\cdot\left(\overline{b_5}\cdot\overline{b_7+b_6a_0+b_6a_1}+b_5b_7\overline{b_6}\,\overline{a_1}\,\overline{a_0}\right) + \left(\overline{b_1}b_0a_7+b_1\overline{a_7}\right)\cdot\left[\overline{b_5}\cdot\left(b_7+b_6a_0+b_6a_1\right) + b_5\cdot\overline{b_7\overline{b_6}\,\overline{a_1}\;\overline{a_0}}\right], \tag{15}$$

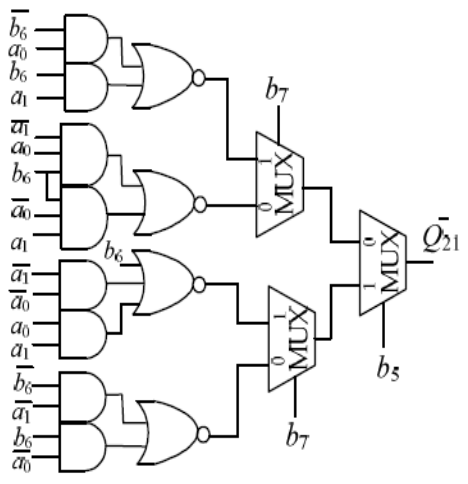
$$Q_{21}^- = \overline{b_5}\cdot\left(\overline{b_7}\cdot\overline{b_6a_0\overline{a_1}+b_6\overline{a_0}a_1} + b_7\cdot\overline{\overline{b_6}a_0+b_6a_1}\right) + b_5\cdot\left(\overline{b_7}\cdot\overline{\overline{b_6}\,\overline{a_1}+b_6\overline{a_0}} + b_7\cdot\overline{b_6+\overline{a_1}\;\overline{a_0}+a_1a_0}\right), \tag{16}$$

$$Q_{20}^- = \overline{b_6}a_0 + \overline{b_5}\,\overline{a_0}. \tag{17}$$

circuit diagram can be expressed in figure 5

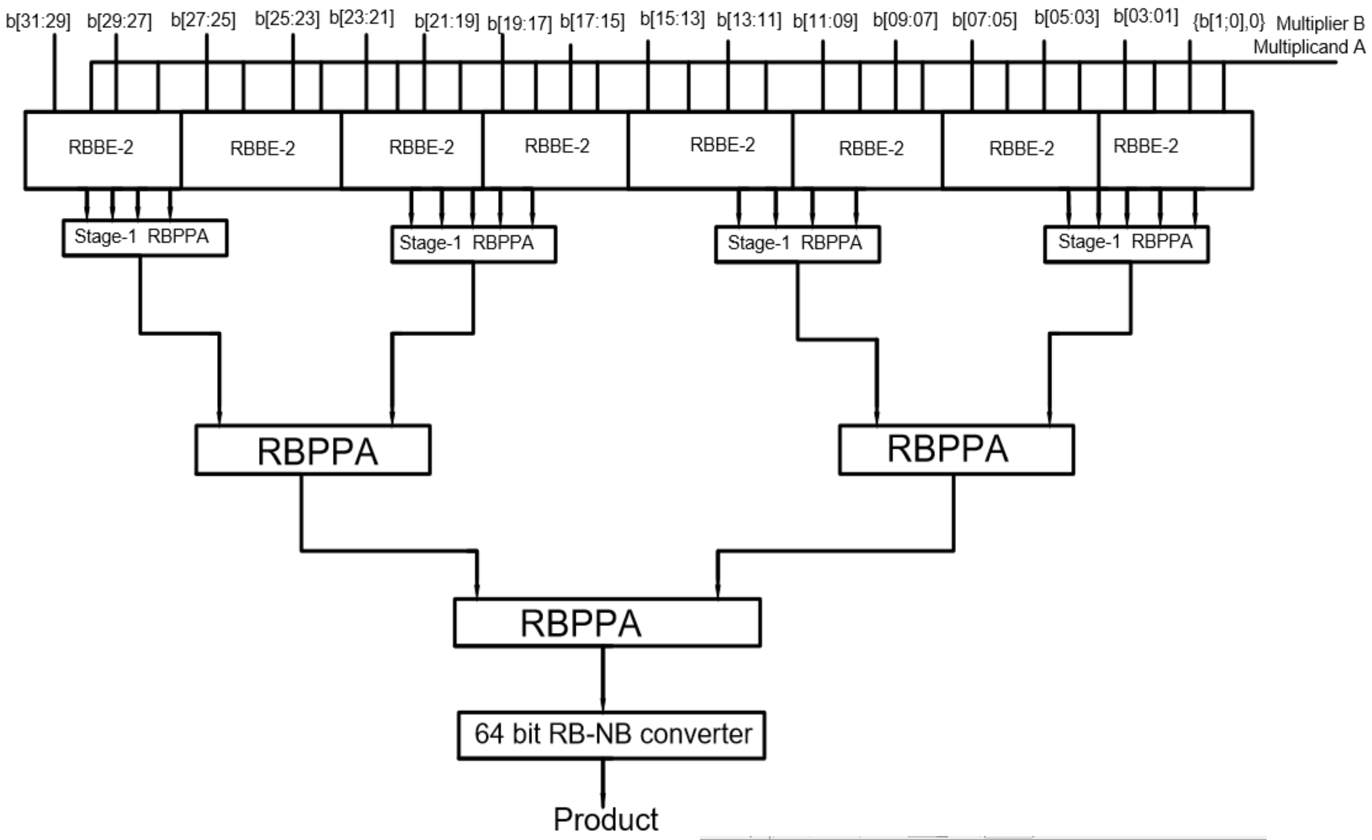


(a)



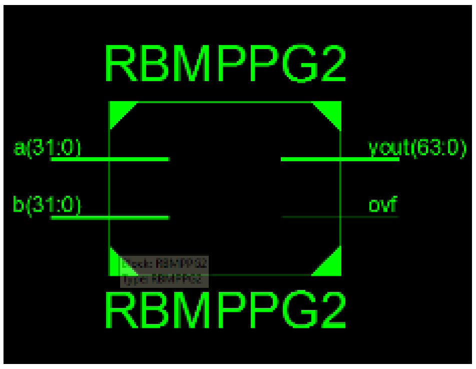
(b)

Figure 5. the circuit diagram of modified partial product variables (a) Q_{18}^+ and Q_{19}^+ (b) Q_{21}^-

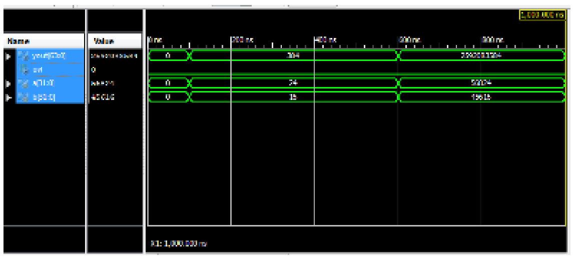


In the second stage, a 4-stage RBA summing tree is used to sum 16 RB partial products. Each RBA block contains 64 RB full adder (RBFA) cells and a varying number of RB half adder (RBHA) cells depending on where it is located**.** The proposed RBMPPG-2 can be applied to any bit RB multipliers with a reduction of a RBPP accumulation stage compared with conventional designs. The 64-bitRB-NB converter converts the final accumulation results into the NB representation, which uses a hybrid parallel prefix/carry select adder.

IV. RESULTS & DISCUSSIONS



(a) Top Level schematic of RBMPG-2



(b) Simulation results of RBMPG-2

Figure 6. (a) shows the top level schematic of RBMPG-2 and fig (b) shows the simulation results of propsed multiplier

In the above results a,b are the multiplicand and multiplier and both are of 32 bits.Where y is the result and it is of 64 bits.for convenience all are shown in signed decimal number.in the above fig we have taken multiplicand and multiplier as 24 and 16 and the output is 384.

By using proposed method we can reduce the the no.of accumulation stages as shown in table5.

Table 5.
Comparison of RBPP Accumulation Stages in RBPP Reduction Tree

Methods	64×64	32×32	16×16	8×8
CRBBE-2	5	4	3	2
RBBE-4 [14]	4	3	2	1
Proposed	4	3	2	1

V. CONCLUSION

A new modified RBPP generator has been proposed in this paper; this design eliminates the additional ECW that is introduced by previous designs. Therefore, a RBPP accumulation stage is saved due to elimination of ECW. The new partial product generation technique can be applied to any 2^n bit RB multipliers to reduce the number of RBPP rows from N/4 + 1 to N/4.

VI. REFERENCES

[1]. A. Avizienis, "Signed-digit number representations for fast parallel arithmetic," IRE Trans. Electron. Computers, vol. EC-10, pp. 389-400, 1961.

[2]. N. Takagi, H. Yasuura, and S. Yajima, "High-speed VLSI multiplication algorithm with a redundant binary addition tree," IEEE Trans. Computers, vol. C-34, pp. 789-796, 1985.

[3]. Y. Harata, Y. Nakamura, H. Nagase, M. Takigawa, and N. Takagi, "A high speed multiplier using a redundant binary adder tree," IEEE J. Solid-State Circuits, vol. SC-22, pp. 28-34,1987.

[4]. H. Edamatsu, T. Taniguchi, T. Nishiyama, and S. Kuninobu, "A 33 MFLOPS floating point processor using redundant binary representation," in Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC), pp. 152-153, 1988.

[5]. H. Makino, Y. Nakase, and H. Shinohara, "A 8.8-ns 54x54-bit multiplier using new redundant binary architecture," in Proc. Int. Conf. Comput. Design (ICCD), pp. 202-205, 1993.

[6]. H. Makino, Y. Nakase, H. Suzuki, H. Morinaka, H. Shinohara, and K. Makino, "An 8.8-ns 54×54-bit multiplier with high speed redundant binary architecture," IEEE J. Solid-State Circuits, vol. 31, pp. 773-783, 1996.

[7]. Y. Kim, B. Song, J. Grosspietsch, and S. Gillig, "A carry-free 54b×54b multiplier using equivalent bit conversion algorithm," IEEE J. Solid-State Circuits, vol. 36, pp. 1538-1545, 2001.

[8]. Y. He and C. Chang, "A power-delay efficient hybrid carry look ahead carry-select based redundant binary to two's complement converter," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 55, pp. 336-346, 2008.

[9]. G. Wang and M. Tull, "A new redundant binary number to 2'scomplement number converter," in Proc. Region 5 Conference: Annual Technical and Leadership Workshop, pp. 141-143, 2004.

[10]. W. Yeh and C. Jen, "High-speed Booth encoded parallel multiplier design," IEEE Trans. Computers, vol. 49, pp. 692-701,2000.

[11]. S. Kuang, J. Wang, and C. Guo, "Modified Booth multiplier with a regular partial product array," IEEE Trans. Circuits Syst. II, vol. 56, pp. .404-408, 2009.

[12]. J. Kang and J. Gaudiot,"A simple high-speed multiplier design," IEEE Trans. Computers, vol. 55, pp.1253-1258, 2006.

[13]. F. Lamberti, N. Andrikos, E. Antelo, and P. Montuschi, "Reducing the computation time in (short bit-width) two's complement multipliers," IEEE Trans. Computers, vol. 60, pp. 148- 156, 2011.

[14]. Y. He and C. Chang, "A new redundant binary Booth encoding for fast 2-bit multiplier design," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 56, pp. 1192-1199, 2009.

[15]. Y. He, C. Chang, J. Gu, and H. Fahmy, "A novel covalent redundant binary Booth encoder," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), vol. 1, pp. 69-72, 2005.

[16]. N. Besli and R. Deshmukh, "A novel redundant binary signed digit (RBSD) Booth's encoding," in Proc. IEEE Southeast Conf.,pp. 426-431, 2002.