IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

A New Modified Redundant Binary Multplier Using Redundant Binary Logic

Sowmya Vemula[1], Mahesh Jangam[2]

[1]M.Tech (DSCE), [2]Assistant Professor (ECE), Sri Visvesvaraya Institute of Technology & Science, Chowderpally, Devarkadra, Mahabubnagar

ABSTRACT

A redundant binary (RB) illustration would be utilized when devising elevated presentation multipliers due to its elevated modularity and carry-free summation. The usual RB multiplier needs other RB partial product (RBPP) row, since an error-correcting word (ECW) is produced by both the radix-4 Modified Booth encoding (MBE) and the RB encoding. This sustains in an extra RBPP accumulation part for the MBE multiplier. In this thesis, a novel RB modified partial product generator (RBMPPG) is projected; it removes the additional ECW and thus, it keeps one RBPP accumulation part. As a result, the projected RBMPPG produces less partial product rows than an usual RB MBE multiplier. Simulation outcomes illustrate that the projected RBMPPG supported structures notably progress the area and power utilization while the word length of every operand in the multiplier is as a minimum 32 bits; these reductions over earlier NB multiplier designs sustain in a modest delay increase (approximately 5 percent). The power-delay product would be decreased utilizing the proposed RB multipliers while evaluated with usual RB multipliers. The proposed modified partial product generator is synthesized exploiting XILINX ISE 14.5 simulation tool.

Keywords- Redundant binary, modified booth encoding, RB partial product generator

I. INTRODUCTION

Digital multipliers are extensively used in arithmetic units of microprocessors, multimedia and digital signal processors. Many algorithms and structures have been projected to design eminent-speed and low-power multipliers. A normal binary (NB) multiplication by digital systems comprises three stages. Partial products are prepared in the initial stage; all partial products are contained by a partial product reduction tree till two partial product rows reside in the subsequent stage. The two partial product rows are summed by a fast carry propagation adder in the next stage. There are two techniques have been used to complete the subsequent stage for the partial product reduction. A first technique uses four-two compressors, whereas a subsequent technique exploits redundant binary (RB) numbers. Both methods permit the partial product reduction tree to be decreased at a rate of 2:1.

The signed-digit arithmetic has been operated using proposed redundant binary number representation; the RB number has the facility to be characterized in unusual methods. Fast multipliers would be peformed utilizing redundant binary addition trees. The redundant binary illustration has too been concerned to a floating-point processor and employed in VLSI. High functionality RB multipliers have become popular due to the improvement characteristics, such as high modularity and carry-free addition.

A RB multiplier contains of a RB partial product (RBPP) generator, a RBPP reduction tree and a RB-NB converter. A Radix- Booth encoding or a modified Booth encoding (MBE) is usually utilized in the partial product generator of parallel multipliers to decrease the amount of partial product rows by half. A RBPP row would be achieved from two adjoining NB partial product rows by complementing one of the pair rows; an N-bit usual RB MBE (CRBBE-2) multiplier needs [N/4] RBPP rows. An additional error-correcting word (ECW) is also needed by together the RB and the Booth encoding; as a result, the amount of RBPP accumulation parts (NRBPPAS) needed by a power-of-two word-length (i.e., 2n-bit) multiplier is specified by:

$$\begin{aligned}\mathrm{NRBPPAS} &= \lceil \log_2(N/4+1) \rceil \\ &= n-1, \text{if } N = 2^n \end{aligned} \quad \text{....(1)}$$

If the additional ECW would be eliminated, an RBPP accumulation part is kept, thus effecting in developments of difficulty and critical path delay for a RB multiplier. For instance, an usual 32-bit RB multiplier has four RBPP accumulation parts; if the ECW is eliminated, then the amount of RBPP accumulation parts is diminished to 3, i.e., the step calculation is shrinked by 25 percent. Reminder that the difficulty of additional ECW does not survive in standard significand size (i.e., 24x24-bit and 54x54-bit) RB multipliers as utilized in floating point-arithmetic units.

This thesis centers on the RBPP generator for intending a 2^n-bit RB multiplier through less partial product rows by removing the superfluous ECW. A novel RB modified partial product generator supported on MBE (RBMPPG-2) is suggested. In the projected RBMPPG-2, the ECW of every row is stimulated to its subsequently neighbor row. In addition, the superfluous ECW produced by the preceding partial product row is joined through both the two most significant bits (MSBs) of the initial partial product row and the two least significant bits (LSBs) of the preceding partial product row by logic generalization. Therefore, the proposed technique decreases the amount of RBPP rows from N/4+1 to N/4, i.e., a RBPP accumulation part is accumulated. The proposed method is applied to 8x8-, 16x16-, 32x32-, and 64x64-bit RB multiplier designs; the designs are synthesized. The projected schemes reach considerable diminutions in area and power utilization evaluated through presented multipliers while the word length of each of the operands is as a minimum 32 bits. While a reserved raise in delay is run into (fairly accurately 5 percent), the power-delay product (PDP) at word lengths of as a minimum 32 bits proves that the projected schemes are the best also by this figure of merit.

This thesis is arranged as follows. Section 2 discusses radix-4 Booth encoding. The structure of the usual RBPP generator is also explained. Section 3 shows the proposed RBMPPG. This section also represents the acceptance of the projected RBMPPG into different word-length RB multipliers. The conclusion is given in Section 5.

II. REVIEW OF BOOTH ENCODING AND RB PARTIAL PRODUCT GENERATOR

2.1 Radix-4 Booth Encoding

Booth encoding has been proposed to assist the multiplication of two's complement binary numbers. It was changed as modified Booth encoding or radix-4 Booth encoding. The MBE scheme is reviewed in Table 1, where A = $a_{N-1}a_{N-2} \dots a_2\ a_1a_0$ situates for the multiplicand, and B = $b_{N-1}b_{N-2} \dots b_2\ b_1b_0$ situates for the multiplier. The multiplier bits are collected in sets of three adjacent bits. The two side bits are partly covered with neighboring collections excluding the initial multiplier bits collection in which it is {b1, b0, 0}. Each collection is decoded by choosing the partial product illustrated in Table 1, where 2A specifies double the multiplicand, which would be achieved by left shifting. Negation function is completed by complemening each bit of A and adding ‘1’ (described as correction bit) to the LSB. Techniques have been introduced to explain the trouble of correction bits for NB radix-4 Booth encoding (NBBE-2) multipliers. Alternatively, this problem has not been solved for RB MBE multipliers.

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

Table 1: MBE Scheme

2.2 RB Partial Product Generator

As two bits are utilized to represent one RB digit, then a RBPP is produced from two NB partial products. The summation of two N-bit NB partial products X and Y using two’s complement illustration can be stated as follows:

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

where $\overline{Y}$ is the inverse of Y, and the similar convention is used in the remaining of the paper. The composite number (X, $\overline{Y}$)can be took as a RB number. The RBPP is produced by reversing one of the two NB partial products and adding -1 to the LSB. Each RB digit X_i belongs to the set{ $\overline{1}$, 0, 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 2 represents one specific RB encoding , where the RB digit is reached by performing X_i^+- X_i^- .

X_i^+	X_i^-	RB digit (X_i)
0	0	0
0	1	$\overline{1}$
1	0	1
1	1	0

Table 2: RB Encoding

Both MBE and RB coding schemes establish errors and two correction terms are needed: 1) when the NB number is converted to a RB format, -1 must be appended 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

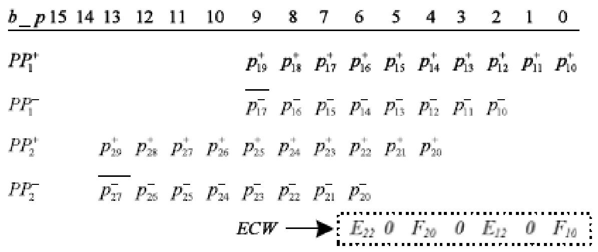


Fig. 1. Conventional RBPP architecture for an 8-bit MBE multiplier

must be appended to the LSB of the partial product. A solitary ECW would recompense faults from both the RB encoding and the radix-4 Booth recoding. The conventional partial product architecture of an 8-bit MBE multiplier is illustrated in Fig. 1, where b_p denotes the bit position, p^{+}_{ij} or p^{-}_{ij} is produced by using an encoder and decoder (Fig. 2) . An N-bit CRBBE-2 multiplier comprises N=4 RBPP rows and one ECW; the ECW takes the form as follows:

$$ECW = E_{(N/4)2}0\,F_{(N/4)0}\ldots 0E_{i2}0\,F_{i0}\ldots 0\,E_{12}0\,F_{10} \quad \ldots.(3)$$

where i denotes the ith row of the RBPPs, $E_{i2} \in \{0,1\}$ and $F_{i0} \in \{0, \overline{1}\}$. In F_{i0}, a-1 correction term is always needed by RB coding. If F_{i0} too 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:

$$E_{i2} = \begin{cases} 0, & \text{no negative encoding} \\ 1, & \text{negative encoding} \end{cases} \quad \ldots.(4)$$

As illustrated in Fig. 1 the first RBPP row, i.e. PP_1, contains of the initial partial product row PP^{+}_{1} and the second partial product row PP^{-}_{1} i.e., $PP^{+}_{1} = p^{+}_{19}p^{+}_{18} \ldots p^{+}_{10}$and $PP^{-}_{1}=p^{-}_{17}p^{-}_{16} \ldots p^{-}_{10}$, where, p^{+}_{19} and p^{+}_{18} are the sign extension bits, so

$$p^{+}_{19} = \overline{p^{+}_{18}} \quad \ldots.(5)$$

$$\begin{aligned} p^{+}_{18} &= \overline{b_1}\,\overline{b_0}\cdot 0 + \overline{b_1}b_0\cdot a_7 + b_1\overline{b_0}\cdot\overline{a_7} + b_1b_0\cdot\overline{a_7} \\ &= \overline{b_1}b_0\cdot a_7 + b_1\overline{a_7} \end{aligned} \quad \ldots.(6)$$

According to Eq. (2), the sign extension bit p^{+}_{29} is also the inverse of p^{+}_{28}. The p^{-}_{17} in PP^{-}_{1} and the p^{-}_{27} in PP^{-}_{2} are also reversed as $\overline{p^{-}_{17}}$ and $\overline{p^{-}_{27}}$. Eq. (5) and Eq. (6) are additional utilized in the subsequently division when presenting the proposed modified RBPP generator.

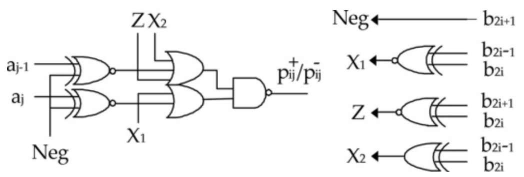


Fig. 2. An encoder and decoder of the MBE scheme

For a 2n-bit CRBBE-2 multiplier, one extra RBPP accumulation part is needed due to the ECW. For a 64-bit RB multiplier, there are five RBPP accumulation parts; as a result, the number of RBPP accumulation parts can be decreased by 20 percent when removing the ECW in a 64-bit RB multiplier, which develops both the complexity and the critical path delay.

III. PROPOSED RB PARTIAL PRODUCT GENERATOR

In this section, a novel RB modified partial product generator supported on MBE (RBMPPG-2) is explained; in this design, ECW is detached by including it into both the two MSBs of the primary partial product row (PP^{+}_{1}) and the two LSBs of the preceding partial product row ($PP^{-}_{(N/4)}$).

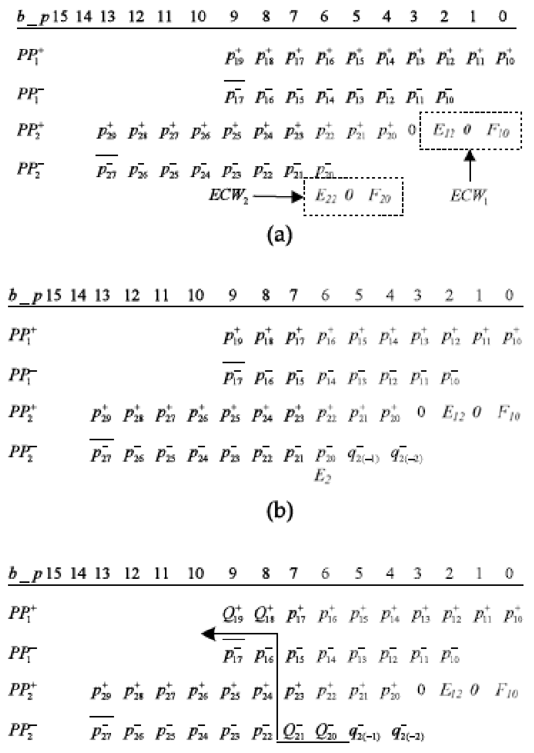


Fig. 3. (a) The first new RBMPPG-2 architecture for an 8-bit MBE multiplier; (b) the further revised RBMPPG-2 architecture by replacing E_{22} and F_{20} with E_2, $\bar{q}_{2(-2)}$, and; (c) the final proposed RBMPPG-2 architecture by totally eliminating ECW2 and further combing E2 into Q^{+}_{19}, Q^{+}_{18}, Q^{-}_{21}, and Q^{-}_{20}

3.1 Proposed RBMPPG_2

Fig. 3 shows the projected RBMPPG-2 scheme for an 8x8-bit multiplier. It is different from the scheme in Fig. 1, where all the error-correcting terms are in the last row. ECW_1 is produced by PP_1 and expressed as

$$ECW_1 = 0\ E_{12}\ 0\ F_{10} \quad \ldots.(7)$$

The ECW2 generated by PP2 (also defined as an extra ECW) is
left as the last row and it is expressed as:

$$ECW_2 = 0\ E_{22}\ 0\ F_{20} \quad \ldots.(8)$$

To remove a RBPP accumulation part, ECW_2 needs to be incorporated into PP_1 and PP_2. As explained in Section 2.2 for F_{i0} and as per Table 1, F_{20} is find out by b_5, b_4, b_3 as follows:

$$F_{20} = \begin{cases} -1, & b_5b_4b_3 = 000, 001, 010, 011,\ \text{or}\ 111 \\ 0, & b_5b_4b_3 = 100, 101, \text{or} 110 \end{cases} \quad \ldots.(9)$$

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

$b_7b_6b_5$	$E_{22}F_{20}$	$E_2q^-_{2(-2)}q^-_{2(-1)}$	p^-_{21}	p^-_{20}
0 0 0	0 $\bar{1}$	$\bar{1}$ 1 1	0	0
0 0 1	0 0	0 0 0	a_1	a_0
0 1 0	0 $\bar{1}$	$\bar{1}$ 1 1	a_1	a_0
0 1 1	0 0	0 0 0	a_0	0
1 0 0	1 $\bar{1}$	0 1 1	$\overline{a_0}$	1
1 0 1	1 0	1 0 0	$\overline{a_1}$	$\overline{a_0}$
1 1 0	1 $\bar{1}$	0 1 1	$\overline{a_1}$	$\overline{a_0}$
1 1 1	0 0	0 0 0	0	0

Table 3: Truth Table of E2, $q^-_{2(-2)}$, $q^-_{2(-1)}$ and p^-_{21}, p^-_{20}

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

$$F_{20} = \begin{cases} -1, & b_5b_4b_3 = 000, 001, 010, \text{or } 011 \\ 0, & b_5b_4b_3 = 100, 101, 110, \text{or } 111 \end{cases} \quad \text{....(10)}$$

By setting PP^+_2 to all ones and adding +1 to the LSB of the partial product, F_{20} can then be determined only by b_5 as follows:

$$F_{20} = \begin{cases} -1, & b_5 = 0 \\ 0, & b_5 = 1 \end{cases} \quad \text{....(11)}$$

A modified radix-4 Booth encoding and a decoding circuit for the partial product PP^+_2 are suggestd here (Fig. 4); an extra threeinput OR gate is then appended to the scheme of (Fig. 2). The three inputs of the additional OR gate are $\overline{b_5}$, $\overline{b_4}$, and $\overline{b_3}$. When b_5 b_4 b_3 = 111, it is clear that $\overline{b_5}\,\overline{b_4}\,\overline{b_3}$ = 000, p^+_{2i} = 1, and PP^+_2 is set to all ones. So, E_{22} and F_{20} in ECW_2 are now find out by $b_7b_6b_5$ without b_4, b_3. Although the complexity is slightly boosted evaluated with the previous design (Fig. 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 (i.e., replacing p^+_{19}; $p+_{18}$, p^-_{21} and p^-_{20}). $q^-_{2(-2)}$, and $q^-_{2(-1)}$ are used to represent the additional partial products that are find out by F20. As -1 can be coded as $\bar{1}$11 in RB format, E22 and F20 can be represented by E2, $q^-_{2(-2)}$, $q^-_{2(-1)}$, (Fig. 3b) as follows:

$$E_2 = \begin{cases} E_{22}, & F_{20} = 0 \\ E_{22} - 1, & F_{20} = -1 \end{cases} \quad \text{....(12)}$$

$$q^-_{2(-2)} = q^-_{2(-1)} = \begin{cases} 0, & F_{20} = 0 \\ 1, & F_{20} = -1 \end{cases} \quad \text{....(13)}$$

As per Eq. (11) and Eq. (13), $q^-_{2(-2)}$, and $q^-_{2(-1)}$ would too be stated as follows:

$$q^-_{2(-2)} = q^-_{2(-1)} = \overline{b_5}. \quad \text{....(14)}$$

This is further discussed by the truth table of E_{22}, F_{20} and E_2, $q^-_{2(-2)}$, $q^-_{2(-1)}$ (Table 3). Now ECW_2 only includes E_2 and $E_2 \in \{0,1,\bar{1}\}$; E_2 can be integrated into the modified partial products Q^+_{19}, Q^+_{18}, Q^+_{21} and Q^-_{20} by replacing p^+_{19}, p^+_{18} and p^-_{21} , p^-_{20} in the shortest path Fig. 3c. From the truth table, E_2 can be find out by $b_7b_6b_5$ as follows:

$$E_2 = \begin{cases} -1, & b_7b_6b_5 = 000, \text{or } 010 \\ 1, & b_7b_6b_5 = 101 \\ 0, & b_7b_6b_5 = 001, 011, 100, 110, \text{or } 111 \end{cases} \quad \text{....(15)}$$

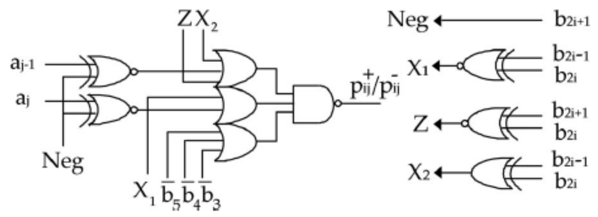


Fig. 4. The modified radix-4 Booth encoding and decoding scheme for PP^+_2

$p^+_{19}p^+_{18}p^-_{21}p^-_{20}$	$Q^+_{19}Q^+_{18}Q^-_{21}Q^-_{20}$ ($E_2 = 0$)	$Q^+_{19}Q^+_{18}Q^-_{21}Q^-_{20}$ ($E_2 = 1$)	$Q^+_{19}Q^+_{18}Q^-_{21}Q^-_{20}$ ($E_2 = -1$)
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

Table 4 The Truth Table of Q^+_{19}, Q^+_{18}, Q^-_{21}, Q^-_{20}

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: $Q^+_{19} = p^+_{19}$, $Q^+_{18} = p^+_{18}$, $Q^-_{21} = p^-_{21}$ and $Q^-_{20} = p^-_{20}$. 2) When $E_2 = 1$, a 1 is added to p^+_{19} ,p^+_{18} ,p^-_{21} ,p^-_{20}. 3) When $E_2 = -1$, a 1 is subtracted from $p^+_{19}p^+_{18}p^-_{21}p^-_{20}$.

The relationships between Q^+_{19}, Q^+_{18}, Q^-_{21}, Q^-_{20} and p^+_{19}, p^+_{18}, p^-_{21}, p^-_{20} are reviewed in Table 4. As the two MSBs of PP^+_1 i.e., p^+_{19} and p^+_{18} take complementary values as shown in Eq. (5), the operations of adding or sub-tracting a 1 will never sustain in an overflow. Therefore, as per Eq. (15) and Table 4, the logic functions of Q^+_{19}, Q^+_{18}, Q^-_{21}, and Q^-_{20} would be stated as follows:

$$\begin{aligned} Q^+_{19} =& (b_7 \oplus b_5 + b_7 b_6 b_5) \cdot p^+_{19} + \overline{b_7}\,\overline{b_5} \cdot (\overline{p^+_{18} + p^-_{21} + p^-_{20}} \oplus \\ & p^+_{19}) + b_7 \overline{b_6} b_5 \cdot (p^+_{18} p^-_{21} p^-_{20} \oplus p^+_{19}) \end{aligned} \quad \text{....(16)}$$

$$\begin{aligned} Q^+_{18} =& (b_7 \oplus b_5 + b_7 b_6 b_5) \cdot p^+_{18} + \overline{b_7}\ \overline{b_5} \cdot (\overline{p^-_{21} + p^-_{20}} \oplus p^+_{18}) \\ & + b_7 \overline{b_6} b_5 \cdot (p^-_{21} p^-_{20} \oplus p^+_{18}) \end{aligned} \quad \text{....(17)}$$

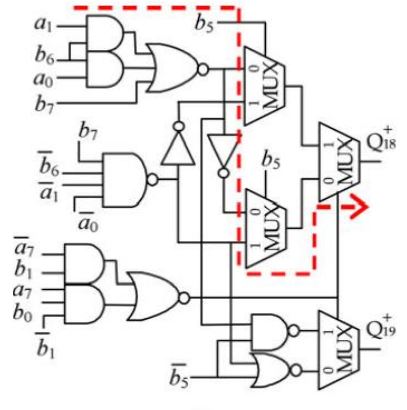
$$\begin{aligned} Q^-_{21} =& (b_7 \oplus b_5 + b_7 b_6 b_5) \cdot p^-_{21} + \overline{b_7}\ \overline{b_5} \cdot \overline{p^-_{21} \oplus p^-_{20}} \\ & + b_7 \overline{b_6} b_5 \cdot p^-_{21} \oplus p^-_{20} \end{aligned} \quad \text{....(18)}$$

$$\begin{aligned} Q^-_{20} =& (b_7 \oplus b_5 + b_7 b_6 b_5) \cdot p^-_{20} + \overline{b_7}\,\overline{b_5} \cdot \overline{p^-_{20}} \\ & + b_7 \overline{b_6} b_5 \cdot \overline{p^-_{20}} \end{aligned} \quad \text{....(19)}$$

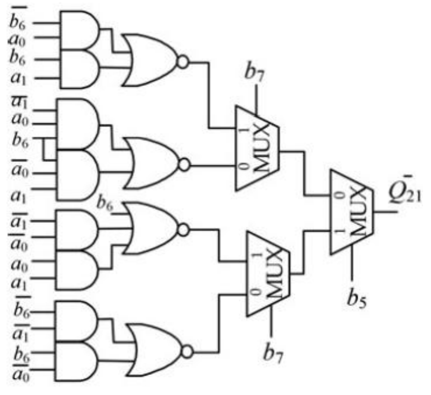
The delay of the RBMPPG-2 can be further decreased by generating Q^+_{19}, Q^+_{18}, Q^-_{21}, Q^-_{20} directly from the mul-tiplicand A and the multiplier B. The associations among p^+_{19}, p^+_{18} and A, B have been explained in Section 2.2 as Eq. (5) and Eq. (6). The relationships between p^-_{21}, p^-_{20} and A, B are also shown in Table 3 concurrencing to the MBE scheme. Therefore, Q^+_{19}, Q^+_{18}, Q^-_{21}, and Q^-_{20} would be stated as follows by replacing p^+_{19}, p^+_{18}, p^-_{21}, and p^-_{20} with the multiplicand bits (ai) and the multiplier bits (bi) after simplification:

$$\begin{aligned} Q^+_{19} =& \overline{\overline{b_1} b_0 a_7 + b_1 \overline{a_7}} \left(\overline{\overline{b_5} \cdot \overline{b_7 + b_6 a_0 + b_6 a_1}} \right) \\ & + \left(\overline{b_1} b_0 a_7 + b_1 \overline{a_7} \right) \cdot b_5 \cdot b_7 \overline{b_6}\, \overline{a_1}\ \overline{a_0} \end{aligned} \quad \text{....(20)}$$

$$\begin{aligned} Q^+_{18} =& \overline{\overline{b_1} b_0 a_7 + b_1 \overline{a_7}} \cdot \left(\overline{b_5} \cdot \overline{b_7 + b_6 a_0 + b_6 a_1} + b_5 b_7 \overline{b_6}\, \overline{a_1}\, \overline{a_0} \right) \\ & + \left(\overline{b_1} b_0 a_7 + b_1 \overline{a_7} \right) \cdot [\overline{b_5} \cdot (b_7 + b_6 a_0 + b_6 a_1) \\ & + b_5 \cdot \overline{b_7 \overline{b_6}\, \overline{a_1}\ \overline{a_0}}] \end{aligned} \quad \text{....(21)}$$



(a)



(b)

Fig. 5. The circuit diagram of the modified partial product va riables: (a) Q$^+_{18}$ and Q$^+_{19}$, (b) Q$^-_{21}$.

$$Q^-_{21} = \overline{b_5} \cdot (\overline{b_7} \cdot \overline{b_6 a_0 \overline{a_1} + b_6 \overline{a_0} a_1} + b_7 \cdot \overline{\overline{b_6} a_0 + b_6 a_1}) + b_5 \cdot (\overline{b_7} \cdot \overline{\overline{b_6}\,\overline{a_1} + b_6 \overline{a_0}} + b_7 \cdot \overline{b_6 + \overline{a_1}\;\overline{a_0} + a_1 a_0}) \quad \text{….(22)}$$

$$Q^-_{20} = \overline{b_6} a_0 + \overline{b_5}\,\overline{a_0} \quad \text{….(23)}$$

The circuit diagrams of the modified partial product variables Q$^+_{18}$, Q$^+_{19}$ and Q$^-_{21}$ are represented in Fig. 5. It is clear that Q$^+_{18}$ has the longest delay path. It is well known that the inverter, the 2-input NAND gate and the transmission gate (TG) are faster than other gates. So, this is popular to use TGs when designing the multiplexer. As illustrated in Fig. 5a, the critical path delay (the dash line) contains of a 1-stage AND-OR-Inverter gate, a one-stage inverter, and two-stage TGs. Therefore, RBMPPG-2 just boosts the TG delay by one-stage evaluated with the MBE partial product of Fig. 2.

The above explanation is only an example; the above method would be concerned to intend any 2^n-bit RB multipliers. It eliminates the superfluous ECW$_{N/4}$ and saves one RBPP accumulation stage, i.e., three XOR gate delays, whereas only a little amplifying the delay of the partial product generation stage. In common, an N-bit RB multiplier has N=4 RBPP rows utilizing the proposed RBMPPG-2. The partial product variables p$^+_{1(N+1)}$, p+$_{1N}$, p$^-_{(N/4)1}$ and p$^-_{(N/4)0}$ can be reinstated by Q$^+_{1\ (N+1)}$, Q$^+_{1\ N}$, Q$^-_{(N/4)1}$, and Q$^-_{(N/4)0}$. The radix-4 Booth decoding of a PPR (PP$^+_{N/4}$) needs additional three-input OR gates (Fig. 4). Therefore, the superfluous ECW$_{N/4}$ is removed by the transformation of four partial product variables Q$^+_{1(N+1)}$, Q$^+_{1N}$, Q$^-_{(N/4)1}$, Q$^-_{(N/4)0}$ and one partial product row is kept in RB multipliers through any power-of two word-length.

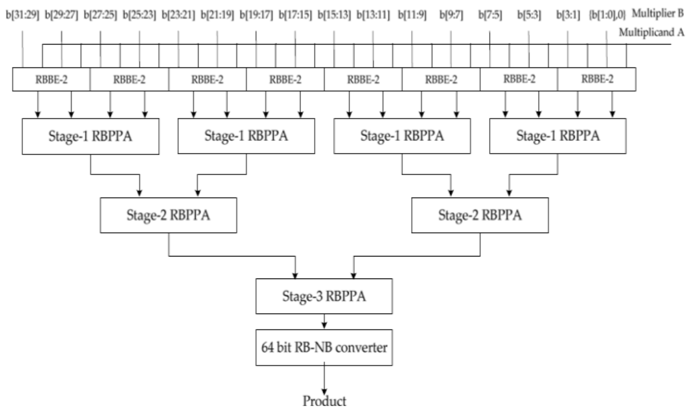


Fig. 6. The block diagram of a 32-bit RB multiplier using the proposed RBMPPG-2

3.2 Design of RBMPPG-2-Based High-Speed RB Multipliers

The projected RBMPPG-2 can be useful to any 2^n-bit RB multipliers through a diminution of a RBPP accumulation part evaluated with usual designs. Even though the delay of RMPPG-2 boosts by one-stage of TG delay, the delay of one RBPP accumulation part is mainly bigger than a one-stage TG delay. Therefore, the delay of the entire multiplier is decreased. The progressd complexity, delay and power utilization are very striking for the proposed design.

The multiplier includes of the projected RBMPPG-2, three RBPP accumulation stages, and one RB-NB converter. A 32-bit RB multiplier developing the projected RBPP originator is represented in Fig. 6. Eight RBBE-2 blocks make the RBPP (p^+_i, p^-_i); they are added up by the RBPP reduction tree that has three RBPP accumulation stages. Each RBPP accumulation obstruct includes RB full adders (RBFAs) and half adders (RBHAs). The 64-bit RB-NB converter translates the last accumulation outcomes into the NB illustration, which utilizes a hybrid parallel-prefix/carry select adder (as solitary of the mainly capable fast parallel adder designs).

There are four steps in an usual 32-bit RB MBE multiplier architecture; however, by utilizing the projected RBMPPG-2,the amount of RBPP accumulation parts is diminished from 4 to 3 (i.e., a 25 percent reduction). These are important savings in delay, area as well as power utilization. The developments in delay, area and power consumption are further demonstrated in the subsequently segment by simulation.

For a 64-bit multiplier, the projected scheme has four RBPP accumulation steps; it decreases the partial product accumulation delay time by 20 percent evaluated through CRBBE-2 multipliers. Although both the projected scheme and RBBE-4 have the same amount of RBPP accumulation stages, RBBE-4 is more complex,because it utilizes radix-16 Booth encoding.

IV. SYNTHESIS AND SIMULATION RESULTS

The projected RB multiplier employing the projected RBMPPG-2 is designed with the XILINX ISE 14.5 simulation tool and executed with Verilog HDL. The RTL diagram and simulation results are displayed below.

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

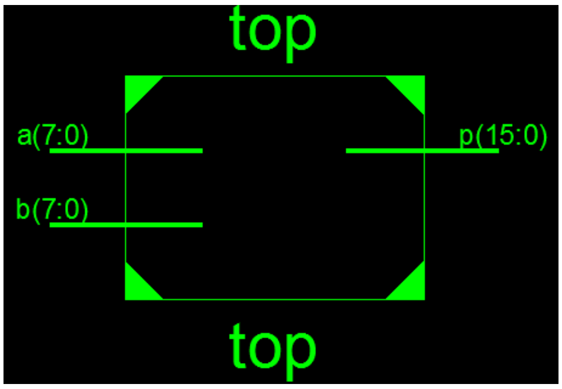


Fig. 7. Top level schematic diagram

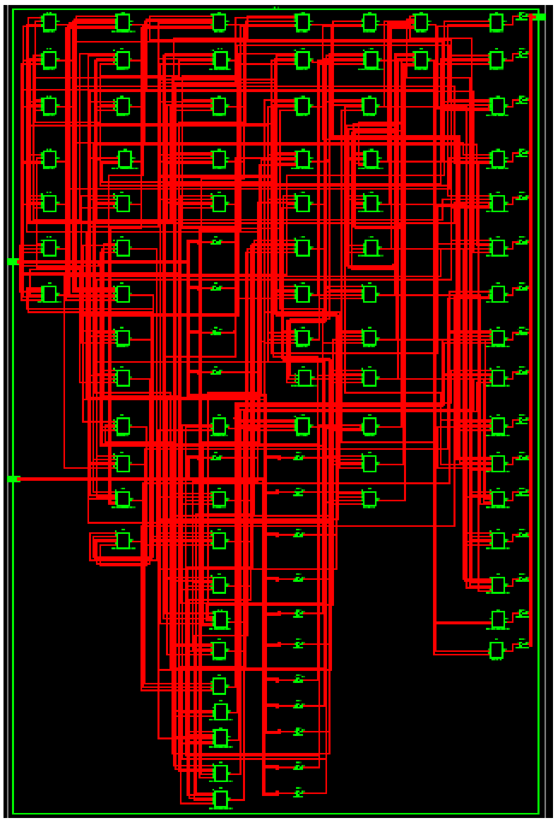
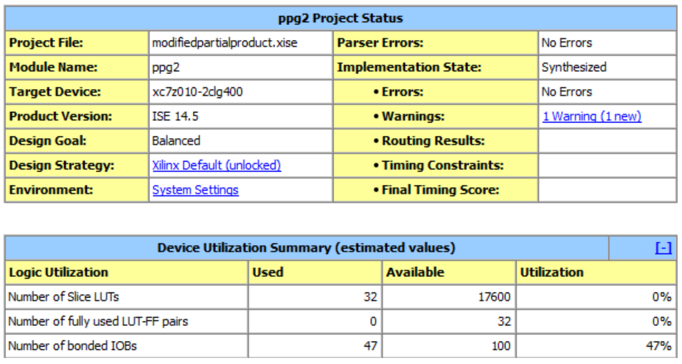


Fig. 8. Internal architectures of RTL diagram



ppg2 Project Status			
Project File:	modifiedpartialproduct.xise	**Parser Errors:**	No Errors
Module Name:	ppg2	**Implementation State:**	Synthesized
Target Device:	xc7z010-2clg400	• **Errors:**	No Errors
Product Version:	ISE 14.5	• **Warnings:**	1 Warning (1 new)
Design Goal:	Balanced	• **Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	
Environment:	System Settings	• **Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	32	17600	0%
Number of fully used LUT-FF pairs	0	32	0%
Number of bonded IOBs	47	100	47%

Fig. 9. Synthesis report

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

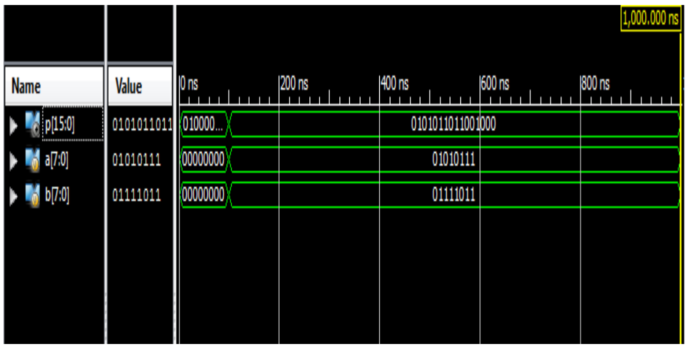


Fig. 10. Simulation result

V. CONCLUSION

In this thesis, a novel modified RBPP generator has been proposed; this design eliminates the extra ECW that is introduced by previous designs. Therefore, a RBPP accumulation part is kept due to the elimination of ECW. Simulation results have shown that the performance of RB MBE multipliers utilizing the projected RBMPPG-2 is improved significantly in terms of delay and area. Synthesis of proposed system done by the Verilog HDL synthesized in Xilinx ISE 14.5. The projected schemes complete main diminutions in area and power utilization when the word length is as a minimum 32 bits. The projected RBPP generation method is a very useful technique when designing area and PDP efficient power-of-two RB MBE multipliers.

VI. FUTURE SCOPE

In this proposed paper, we modify the multiplier by boosting the amount of input to the multiplier. In our modification these inputs are in 64 bits and the output in 128 bits.

REFERENCES

[1] A. Avizienis, “Signed-digit number representations for fast parallel arithmetic,” IRE Trans. Electron. Comput., 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. Comput., vol. C-34, no. 9, pp. 789–796, Sep. 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, no. 1, pp. 28–34, Feb. 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., 1988, pp. 152–153.

[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. Des., 1993, pp. 202–205.

[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, no. 6, pp. 773–783, Jun. 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, no. 10, pp. 1538–1545, Oct. 2001.

[8] Y. He and C. Chang, "A power-delay efficient hybrid carry-lookahead carry-select based redundant binary to two's complement converter," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 55, no. 1, pp. 336–346, Feb. 2008.

[9] G. Wang and M. Tull, "A new redundant binary number to 2's-complement number converter," in Proc. Region 5 Conf.: Annu. Tech. Leadership Workshop, 2004, pp. 141–143.

[10] W. Yeh and C. Jen, "High-speed booth encoded parallel multiplier design," IEEE Trans. Comput., vol. 49, no. 7, pp. 692–701, Jul. 2000.

AUTHOR DETAILS

	SOWMYA VEMULA, pursuing M.Tech (DSCE) from Sri Visvesvaraya Institute Of Technology & Science, Chowderpally (Vill), Devarkadra (Mdl), Mahabubnagar (Dist), TS, INDIA.
	MAHESH JANGAM, working as Assistant professor (ECE) from Sri Visvesvaraya Institute Of Technology & Science, Chowderpally (Vill), Devarkadra (Mdl), Mahabubnagar (Dist), TS, INDIA.