Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2021) 29: 2012 – 2028
© TÜBİTAK
doi:10.3906/elk-2007-179

Area-delay efficient Radix-4 8×8 Booth multiplier for DSP applications

Subodh K. SINGHAL[1], **Sujit K. PATEL**[2,*], **Anurag MAHAJAN**[3], **Gaurav SAXENA**[1]
[1]Jaypee University of Engineering & Technology, Guna, Madhya Pradesh, India
[2]Thapar Institute of Engineering and Technology, Patiala, Punjab, India
[3]Symbiosis Institute of Technology, Symbiosis International (Deemed University), Pune, India

Received: 01.08.2020 • **Accepted/Published Online:** 29.03.2021 • **Final Version:** 26.07.2021

Abstract: Booth multiplier is the key component in portable very large-scale integration (VSLI) systems enabled with signal and image processing applications. The area, delay, and energy are the major constraints in these systems. Therefore, in this paper, a detailed analysis of the state-of-the-art Booth multiplier architecture and its various internal units are presented to find the scope of optimization. Based on the finding of analysis, optimized new binary to 2's complement (B2C), Booth encoder-cum-selector type-1 and type-2, and partial product addition units are proposed. Furthermore, using these optimized units, an efficient parallel radix-4 8×8 Booth multiplier architecture is proposed. The simulation is carried out to verify the functionality of the proposed design. The synthesis results show that the proposed structure offers a saving of 13.56% in delay and 34.87% in area compared to the recent similar Booth multiplier design. Comparison results also reveal that the proposed Booth multiplier design involves 43.7% less area-delay-product and 11.24% less energy compared to the recent Booth multiplier design. Therefore, the proposed Booth multiplier design could be helpful for efficient realization of digital signal processing systems.

Key words: Area, Booth encoding, delay, energy, multiplier, VLSI

1. Introduction

Modern electronic gadgets like laptops, mobiles, palmtops, and hearing-aids are battery-operated and portable in nature. The area, speed and energy are three major constraints to develop these portable devices with limited battery backup. Besides, these devices perform various signal and image processing tasks such as digital filtering [1–3], transformations [4–6], and audio/video encoding/decoding [7]. The multiplier is the most commonly used component in these applications to perform the multiplication operations. Therefore, multipliers with less area and less energy with high performance are extremely important. In multipliers, the multiplication operation is performed in two steps: (i) generation of partial product rows using AND operations and (ii) addition of partial product rows using adders to generate the final product. The multiplier has a more complex structure as compared to other arithmetic components. Therefore, various multiplier structures such as array, Wallace, Baugh Wooley, and Booth are suggested in the literature to achieve the area, and delay-efficient designs based on the optimization strategy adapted for the addition of partial product rows [8–11].

In recent years, Booth multiplier is most popular over others due to its area-, delay-, and energy-efficient structure. The key idea behind the Booth multiplication is to reduce the partial products using Booth encoding algorithm. In the conventional radix-4 8×8 Booth multiplier structure (figure 1a of [12]), the

*Correspondence: sujit.patel@thapar.edu

This work is licensed under a Creative Commons Attribution 4.0 International License.

multiplication operation is performed in four stages, where each stage contains 3-to-1 word-level multiplexer, 15-bit adder/subtractor, and encoding unit. The encoding unit generates select signal for word-level multiplexer and another signal for adder/subtractor. The word-level multiplexer selects any one of the encoded value based on the select signal received from the encoding unit; finally, adder/subtractor computes the addition/subtraction of the partial product and output of the previous adder/subtractor. The conventional Booth multiplier structure takes more delay due to serial addition of partial product and occupies more area due to involvement of large amount of logic resources. Therefore, several attempts have been made by various researchers in the literature for the development of area-delay-efficient Booth multipliers. The key developments of the last two decades in this area are as follows.

The modified Booth algorithm has been suggested in [13], where partial product rows are summed in parallel fashion for the fast implementation of multiplication operation. Subsequently, the author in [14] has given modified-Booth-algorithm-based multiplier for the implementation of multiply-accumulator. Kuang et al. [15] have given a regular partial product array generation scheme for Booth multiplier to reduce the area, delay, and power. Further in [16], researchers have suggested Booth recoding methodology for the low-power implementation of Booth multiplier for digital signal processing (DSP) processors. Reversible-logic-based radix-4 Booth multiplier has been suggested in [17] for implementation of DSP applications. Xue et al. [12] have suggested radix-4 8×8 Booth multiplier with an optimized architecture for various stages. In this work, a novel binary to 2's complementing circuit and a 2-to-1 multiplexers are used to minimize the delay and power. Few fixed width Booth multiplier architectures have been suggested in [18–22] for optimization of area, delay, and power but at the cost of error in the output.

Recently, the radix-4 8×8 Booth multiplier designs are proposed in [23, 24]. In [23], there are two designs suggested for the area and delay optimization. However, these designs use 15-bit adders for the addition of encoded partial products, which increases the complexity. The radix-4 8×8 Booth multiplier structure of [24] performs the parallel addition of encoded partial products for reducing the delay. In this structure, they have modified the Booth encoder and binary to 2's complement converter (B2C) designs. Besides, new carry-look-ahead-based carry select adder (CLA-CSLA) was also developed and utilized in the Booth multiplier design. However, it is observed that the Booth multiplier design of [24] involves redundant logics. Therefore, there is a scope for improvement in delay and area of the recent Booth multiplier structure [24]. A detailed analysis of the structure of [24] as well as its various internal units is presented in the next section to identify the redundant logics.

2. Analysis of Booth multiplier architecture

This section presents a detailed analysis of the recent Booth multiplier architecture of [24] for finding the possibilities of area-delay minimization. For the analysis, the radix-4 8×8 Booth multiplier architecture of [24] is considered as shown in Figure 1. It consists of B2C, three Booth encoders, 4-to-1 word-level multiplexer, and adder tree units. In this architecture, the multiplication of two binary numbers (A and B) is performed in three stages. In the first stage, the radix-4 Booth-encoded partial products are generated, whereas in the second and third stages, the addition of these partial products is performed. Therefore, for the optimization of Booth multiplier architecture, the separate analysis of each unit is presented in the following subsections.

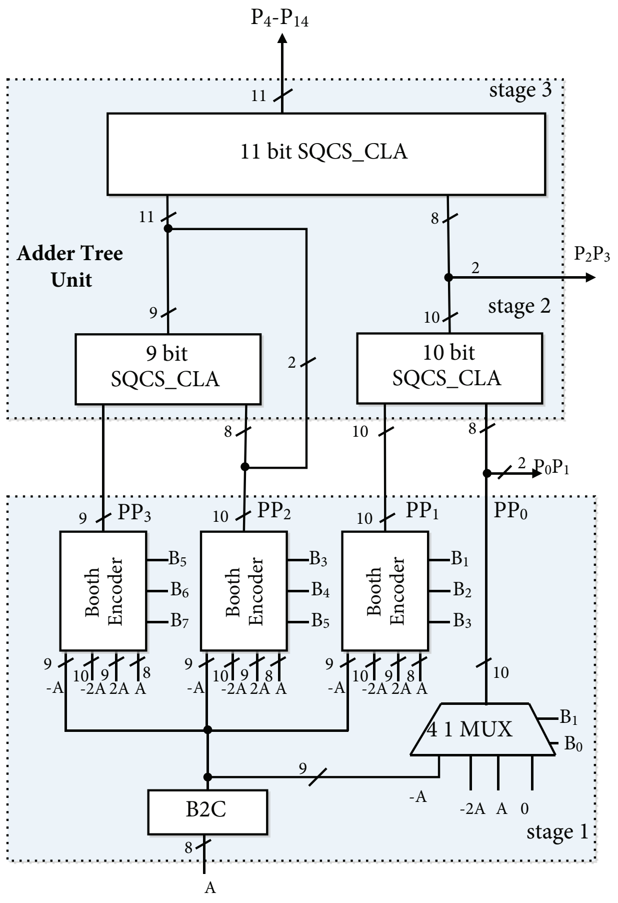


Figure 1. Booth multiplier architecture [24].

2.1. Binary to 2's complement unit

The binary to 2's complement (B2C) unit of [24] is shown in Figure 2. This unit computes the 2's complement of the applied input A. The path delay of the existing B2C unit (shown with red dotted line) can be expressed as

$$T_{B2C} = T_{XNOR2} + 2 \times T_{NOR4} + T_{NOR2} + 2 \times T_{NOT} \tag{1}$$

where $T_{B2C}, T_{XNOR2}, T_{NOR4}, T_{NOR2}$, and T_{NOT} represent the delay of existing B2C unit, 2-input XNOR, 4-input NOR, 2-input NOR, and NOT gates, respectively. From Eq. (1), it is observed that the existing B2C unit involves many gates in the critical path which increases its delay. Proper analysis of logic operation involved in the critical path can be helpful for the optimization of delay. Therefore, the logic expressions of the two most significant output bits (S_7 and S_8) are considered as given below

$$S_7 = \overline{A_7} \cdot \overline{X} + A_7 \cdot X \tag{2}$$

$$S_8 = \overline{A_8} \cdot \overline{Y} + A_8 \cdot Y; \quad Y = \overline{A7 + \overline{X}} \tag{3}$$

It is observed from Eqs. (2) and (3) that S_7 is the function of A_7 and X, whereas S_8 is the function of A_7, A_8, and X. However, A_8 is sign-extended bit of an input A, i.e. $A_8 = A_7$. This analysis provides the scope for

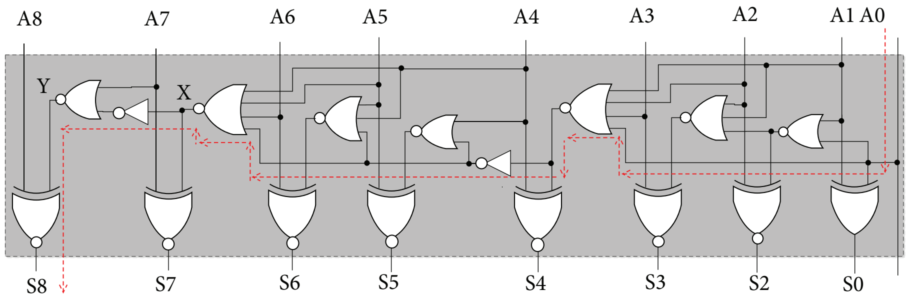


Figure 2. Logic circuit of B2C unit [24].

possible simplification in the logic circuit for the reduction in delay as well as in area.

2.2. Booth encoder and multiplexer unit

Booth encoder circuit basically generates the partial product. The existing [24] design of Booth encoder is shown in Figure 3a. It takes 3-bit of the multiplicand (B) as input and generates four select signals P_i, Q_i, R_i, S_i, where $1 \leq i \leq 3$. The Boolean expressions of the select signals are given below.

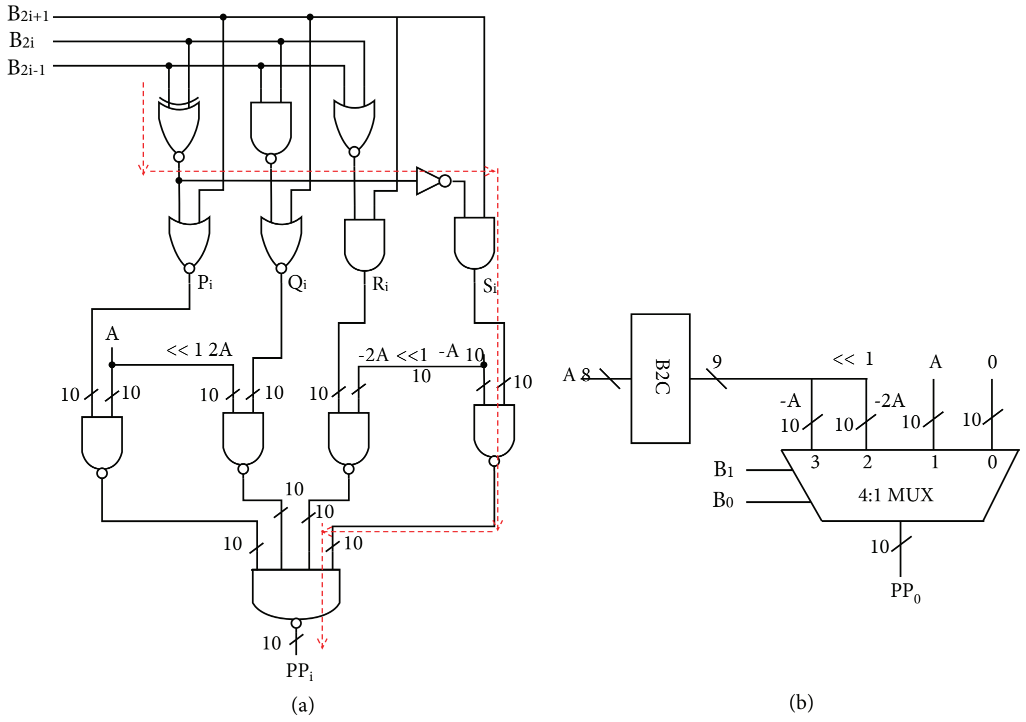


Figure 3. Partial product generation circuits for (a) PP_i, where $1 \leq i \leq 3$, (b) PP_0 [24].

$$P_i = \overline{(\overline{B_{2i-1} \oplus B_{2i}} + B_{2i+1})} \tag{4}$$

$$Q_i = \overline{\overline{B_{2i-1} \cdot B_{2i}} + B_{2i+1}} \tag{5}$$

$$R_i = (\overline{B_{2i-1} + B_{2i}}) \cdot B_{2i+1} \tag{6}$$

$$S_i = (B_{2i-1} \oplus B_{2i}) \cdot B_{2i+1} \tag{7}$$

The select signals given in Eqs. (4)–(7) are utilized for the selection of an appropriate partial product (PP_i) from the set of 10-bit partial products A, $2A$, $-A$, and $-2A$. The logic expression for the generation of partial product is given as:

$$PP_i = \overline{\overline{P_i \cdot A} + \overline{Q_i \cdot 2A} + \overline{R_i \cdot (-2A)} + \overline{S_i \cdot (-A)}} \tag{8}$$

The path (mentioned with dotted line in Figure 3a) delay of the existing Booth encoder circuit can be expressed as

$$T_{En} = T_{XNOR2} + T_{NOT} + T_{AND2} + T_{NAND2} + T_{NAND4} \tag{9}$$

where $T_{En}, T_{XNOR2}, T_{NOT}, T_{AND2}, T_{NAND2}$, and T_{NAND4} represent the delay of Booth encoder, 2-input XNOR, NOT, 2-input AND, 2-input NAND, and 4-input NAND gates, respectively. From Eq. (9), it is observed that the major contribution in the delay of encoder is due to delay of XNOR and AND gates. However, from the TCBN65GPLUS TSMC 65nm core library databook, it is observed that the delay of XNOR and AND gates is higher in comparison to other gates like NAND and NOR. Therefore, to reduce the critical path delay of an encoder circuit, the select signals given in Eqs. (4)–(7) need to be reformulated in such a manner that the XNOR and AND replaced with some other gates which having less delay. This reformulation could be helpful to optimize the delay of encoder circuit. Apart from this, the actual bit-width of possible partial products A, $2A$, $-A$, and $-2A$ are 8-, 9-, 9-, and 10-bit, respectively. In the existing Booth encoder design, these possible partial products bit-width are equalized to 10-bit and selected according to the values of P_i, Q_i, R_i, and S_i to get the desired partial product PP_i according to Eq. (8). This analysis reveals that the equalization of bit-width of the partial products before the first stage of NAND gates in the selection-logic increases some logic resources. This could be avoided if bit-width equalization is performed after the first stage of NAND gates.

Besides, in the existing Booth multiplier of [24], the 4-to-1 word level multiplexer (Figure 3b) is used for the generation of first partial product (PP_0) from the possible values 0, A, $-2A$, and $-A$. However, one of the input of multiplexer is fixed to 0 value which could be exploited to remove the redundant logic gates in the multiplexer design.

2.3. Adder tree unit

It is basically used to add the four partial products (PP_3, PP_2, PP_1, PP_0) generated through the three parallel Booth encoders and one 4-to-1 word-level multiplexer to get the final product. In the existing Booth multiplier of [24], the adder tree unit is constructed using three CLA-CSLA–based adders. For the analysis, 9-bit CLA-CSLA adder design is taken from Booth multiplier structure of [24] as shown in Figure 4. It can be observed from Figure 4 that each segment consists of two CLAs (one with carry-in '0' and other with carry-in '1') and 2-to-1 multiplexers. To explore the possible reduction in area and delay, the 3-bit CLA-CSLA segment (shown

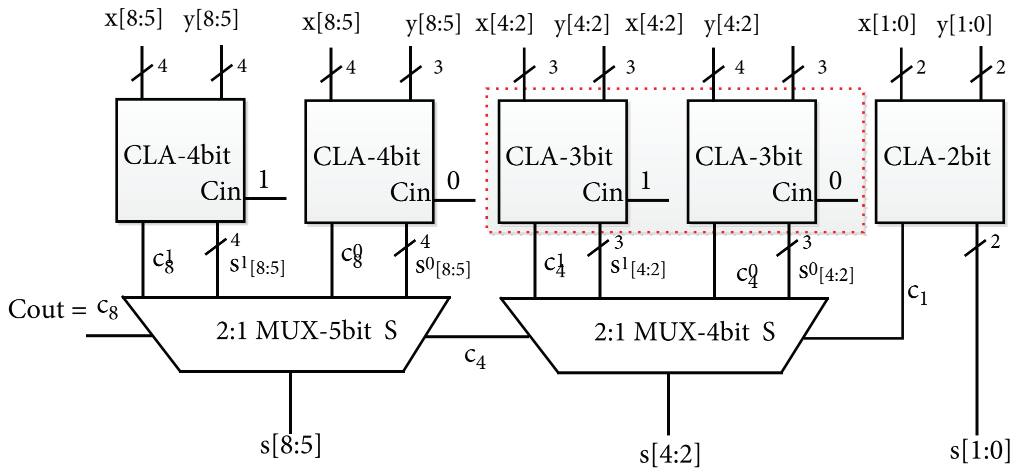


Figure 4. The 9-bit CLA-CSLA design [24].

in Figure 4 with red rectangular box) is considered. The operations performed in the 3-bit CLA-CSLA segment are expressed by the following Boolean expressions.

CLA with cin = 0 :

$$p_2 = x_2 \oplus y_2; \quad p_3 = x_3 \oplus y_3; \quad p_4 = x_4 \oplus y_4 \tag{10}$$

$$g_2 = x_2 \cdot y_2; \quad g_3 = x_3 \cdot y_3; \quad g_4 = x_4 \cdot y_4 \tag{11}$$

$$c_2^0 = g_2 \tag{12}$$

$$c_3^0 = g_3 + p_3 \cdot g_2 \tag{13}$$

$$c_4^0 = g_4 + p_4 \cdot g_3 + p_4 \cdot p_3 \cdot g_2 \tag{14}$$

$$s_2^0 = p_2; \quad s_3^0 = p_3 \oplus c_2^0; \quad s_4^0 = p_4 \oplus c_3^0 \tag{15}$$

CLA with cin = 1 :

$$p_2 = x_2 \oplus y_2; \quad p_3 = x_3 \oplus y_3; \quad p_4 = x_4 \oplus y_4 \tag{16}$$

$$g_2 = x_2 \cdot y_2; \quad g_3 = x_3 \cdot y_3; \quad g_4 = x_4 \cdot y_4 \tag{17}$$

$$c_2^1 = \underbrace{g_2}_{c_2^0} + p_2 \tag{18}$$

$$c_3^1 = \underbrace{g_3 + p_3 \cdot g_2}_{c_3^0} + p_3 \cdot p_2 \tag{19}$$

$$c_4^1 = \underbrace{g_4 + p_4 \cdot g_3 + p_4 \cdot p_3 \cdot g_2}_{c_4^0} + p_4 \cdot p_3 \cdot p_2 \tag{20}$$

$$s_2^1 = \overline{p_2}; \quad s_3^1 = p_3 \oplus c_2^1; \quad s_4^1 = p_4 \oplus c_3^1 \tag{21}$$

From Eqs. (16)–(21) given for CLA with carry-in ‘1’, it is observed that Eqs. (16) and (17) are identical to Eqs. (10) and (11) of CLA with carry-in '0', respectively. Similarly, in Eqs. (18)–(20) of CLA with carry-in '1', there are some terms which are also available in Eqs. (12)–(14) of CLA with carry-in '0'. From the above study, it seems that many terms are redundant in the CLA-CSLA segment. Removal of these redundant terms could be helpful to reduce the logic complexity of CLA-CSLA adder. Apart from this, the tree adder uses three CLA-CSLA adders for the addition of four partial products which increases hardware complexity of the Booth multiplier design. An alternate of tree adder may help to minimize the logic resources of the Booth multiplier design.

Based on the abovementioned analysis, it is found that the various units of existing Booth multiplier structure [24] contains many redundant/common logic operations. These redundancy could be avoided by simplifying the Boolean expressions corresponding to the operations of various units. The analysis also reveals that the adder tree unit used in the existing Booth multiplier [10] occupies most of the area of the design. An alternate approach of adder tree unit could be helpful to optimize the area complexity of the Booth multiplier design. By keeping all these facts in mind, the simplified Boolean expressions of various units of Booth multiplier structure are presented in this paper. Using these simplified expressions, the structure of Booth multiplier and its various units are also proposed. Besides, for area optimization, the existing adder tree unit is replaced by the proposed partial products addition unit. The key contributions of the paper are as follows:

- The critical path of B2C unit is analyzed for delay optimization and proposed new B2C unit with reduced delay.
- The design of Booth encoder and multiplexer is analyzed, and optimized Booth encoder-cum-selector type-1 and type-2 units to replace the existing booth encoder and word level 4-to-1 multiplexer is proposed.
- The logic redundancy in CLA-CSLA–based adder tree unit is analyzed, and partial products addition unit based on the compressor module is proposed, and CLA-CSLA adder is optimized.
- An efficient architecture for Booth multiplier based on the optimized proposed units is proposed.

The rest of the paper is organized as follows. Section 3 presents the proposed Booth multiplier architecture. In Section 4, functional verification and performance comparison are presented. Finally, Section 5 presents the conclusion.

3. Proposed Booth multiplier architecture

The proposed radix-4 8×8 Booth multiplier architecture is presented in Figure 5. It takes 8-bit inputs A and B, where A and B are multiplicand and multiplier terms, respectively, and generates 15-bit product (P). The proposed architecture consists of new B2C, three Booth encoder-cum-selector modules of type-1, one Booth encoder-cum-selector module of type-2, and partial product addition units. The new B2C unit performs 2's complement operation on A to generate the value $-A$. Three parallel Booth encoder-cum-selector of type-1 receive A, $-A$, and 3-bit of the B in parallel, whereas encoder-cum-selector of type-2 receives A, $-A$, and 2-bit of the B, and generate four partial products PP_3, PP_2, PP_1, and PP_0. These partial products are added using partial product addition unit to generate the final product (P). The detailed description of each unit of the proposed architecture is discussed in the following subsections.

3.1. New B2C unit

The analysis given in Subsection 2.1 reveals that the existing B2C unit contains some redundant logics in the critical path. These redundant logics could be avoided by simplification of Boolean expression of S_8 given in

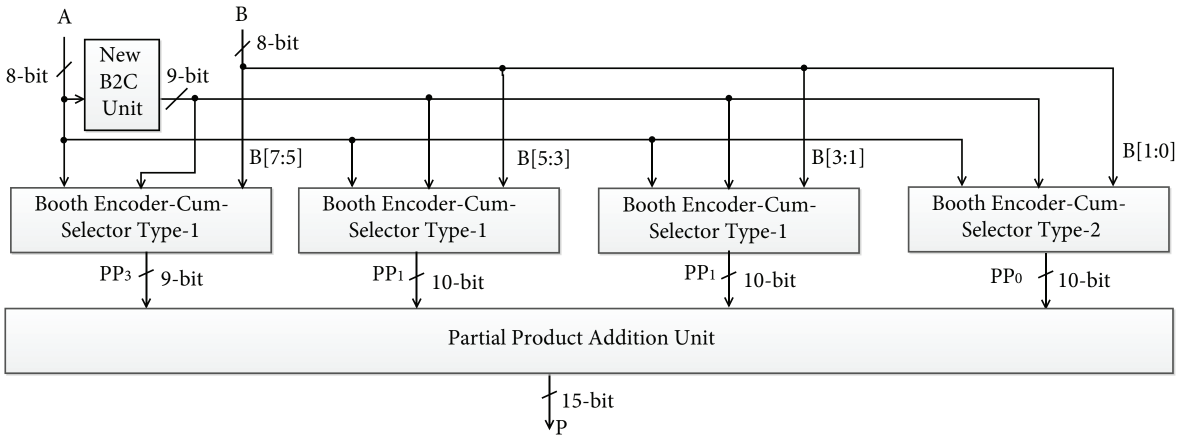

A
B
8-bit
8-bit
New B2C Unit
9-bit
B[7:5]
B[5:3]
B[3:1]
B[1:0]
Booth Encoder-Cum-Selector Type-1
Booth Encoder-Cum-Selector Type-1
Booth Encoder-Cum-Selector Type-1
Booth Encoder-Cum-Selector Type-2
PP3
9-bit
PP1
10-bit
PP1
10-bit
PP0
10-bit
Partial Product Addition Unit
15-bit
P

Figure 5. Proposed radix-4 8×8 Booth multiplier architecture.

(3) as well as restructuring of existing B2C logic circuit (Figure 2). Therefore, it can be simplified as

$$S_8 = \overline{A_8} \cdot \overline{Y} + A_8 \cdot Y; \quad Y = \overline{A_7 + \overline{X}}$$

$$S_8 = \overline{A_8} \cdot \overline{\overline{A_7 + \overline{X}}} + A_8 \cdot \overline{A_7 + \overline{X}}$$

$$S_8 = \overline{A_8} \cdot (A_7 + \overline{X}) + A_8 \cdot (\overline{A_7} \cdot X)$$

However, A_8 is sign-extended bit of A, i.e. $A_8 = A_7$. Therefore,

$$\begin{aligned} S_8 &= \overline{A_7} \cdot (A_7 + \overline{X}) + A_7 \cdot (\overline{A_7} \cdot X) \\ &= \overline{A_7} \cdot A_7 + \overline{A_7} \cdot \overline{X} + (A_7 \cdot \overline{A_7}) \cdot X \\ \because \quad & \overline{A_7} \cdot A_7 = 0 \\ \therefore \quad & S_8 = \overline{A_7} \cdot \overline{X} \end{aligned} \tag{22}$$

Based on the simplified expression of S_8 given in Eq. (22) and restructuring the other part of the existing B2C logic circuit, new logic circuit for B2C unit is proposed as shown in Figure 6. The path delay of the new B2C unit (shown with red dotted line) can be expressed as

$$T_{B2C,New} = T_{XOR2} + 2 \times T_{NOR2} + T_{NAND2} \tag{23}$$

where $T_{B2C,New}, T_{XOR2}, T_{NOR2}$, and T_{NAND2} represent the delay of new B2C unit, 2-input XOR, 2-input, NOR and 2-input NAND gates, respectively. To observe the effect of the proposed new B2C unit, the theoretical comparison of area and delay have been made with the existing B2C unit of [24]. For the area and delay calculation, TCBN65GPLUS TSMC 65nm core library data-book [25] is referred and the values related to the logic gates are listed in Table 1.

The delay of new and existing B2C units are calculated using the values of Table 1 and Eqs. (1) and (23). Similarly, area is calculated using values given Table 1 and logic circuits given in Figures 2 and 6. The calculated values are listed in Table 2 for comparison. From Table 2, it is observed that the new B2C unit involves 35.12% less delay compared to the B2C of [24] with nearly the same area. Therefore, new B2C unit could be helpful for the delay optimization of the Booth multiplier.

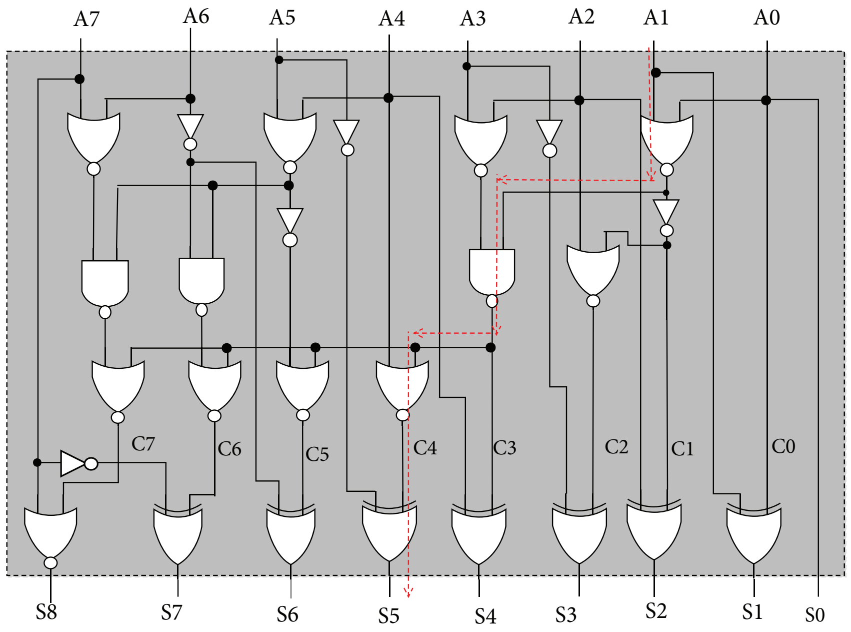


Figure 6. Proposed New B2C unit.

Table 1. Area and delay values of logic cells taken from TCBN65GPLUS TSMC 65nm core library databook.

Logic-Gates	Name of Cell	Delay (*ps*)	Area (# Transistors)
NOT	INVD0	9.4	2
NOR2	NR2D0	13.95	4
NOR3	NR3D0	20.0	6
NOR4	NR4D0	26.8	8
NAND2	ND2D0	12.75	4
NAND3	ND3D0	17.55	6
NAND4	ND4D0	22.95	8
OR2	OR2D0	28.5	6
AND2	AN2D0	26.15	6
XOR2	XOR2D0	42	12
XNOR2	XNR2D0	40.85	12
OAI	OAI21D0	17.55	6
AOI	AOI21D0	18.65	6

Table 2. Comparison for area and delay of new and existing B2C units.

Design	Dealy (ps)	Area (transistor counts)
Existing B2C [24]	127.4	140
New B2C	82.65	148

3.2. Proposed Booth encoder-cum-selector units

Based on the analysis given in Subsection 2.2, it is observed that the Booth encoder circuit could be optimized by reformulation of select signals P_i, Q_i, R_i, and S_i, where $1 \leq i \leq 3$. Therefore, the Boolean expressions of select signals given in Eqs. (4)–(7) are reformulated as:

$$\begin{aligned}
P_i &= \overline{(\overline{B_{2i-1} \oplus B_{2i}} + B_{2i+1})} \\
&= \overline{\overline{B_{2i-1}} \cdot \overline{B_{2i}} + B_{2i-1} \cdot B_{2i} + B_{2i+1}} \\
&= \overline{\overline{B_{2i-1} + B_{2i}} + B_{2i-1} \cdot B_{2i} + B_{2i+1}} \\
&= \overline{\overline{(B_{2i-1} + B_{2i}) \cdot (\overline{B_{2i-1} \cdot B_{2i}})} + B_{2i+1}} \qquad (24)
\end{aligned}$$

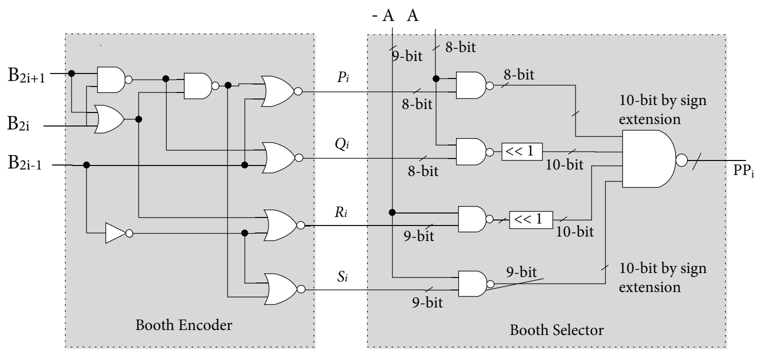
$$Q_i = \overline{\overline{B_{2i-1} \cdot B_{2i}} + B_{2i+1}} \qquad (25)$$

$$\begin{aligned}
R_i &= (\overline{B_{2i-1} + B_{2i}}) \cdot B_{2i+1} \\
&= \overline{(B_{2i-1} + B_{2i}) + \overline{B_{2i+1}}} \qquad (26)
\end{aligned}$$

$$\begin{aligned}
S_i &= (B_{2i-1} \oplus B_{2i}) \cdot B_{2i+1} \\
&= \overline{\overline{(B_{2i-1} \oplus B_{2i})} + \overline{B_{2i+1}}} \\
&= \overline{\overline{B_{2i-1}} \cdot \overline{B_{2i}} + B_{2i-1} \cdot B_{2i} + \overline{B_{2i+1}}} \\
&= \overline{\overline{B_{2i-1} + B_{2i}} + B_{2i-1} \cdot B_{2i} + \overline{B_{2i+1}}} \\
&= \overline{\overline{(B_{2i-1} + B_{2i}) \cdot (\overline{B_{2i-1} \cdot B_{2i}})} + \overline{B_{2i+1}}} \qquad (27)
\end{aligned}$$

From the reformulated expressions given in Eqs. (24)–(27), it can be observed that many terms are common in these expressions which can be shared to derive the efficient encoder circuit. Along with this, analysis given in Subsection 2.2 provides that the first stage NAND gates of the selection logic could be optimized if bit-width equalization is done after first stage of the NAND gates. Based on this observation and reformulated logic expressions of select signals given in Eqs. (24)–(27), Booth encoder-cum-selector type-1 circuit is proposed for the generation of partial product PP_i, $1 \leq i \leq 3$. The proposed logic circuit of this is shown in Figure 7a. In the proposed circuit, bit-width equalization is performed after the first stage (Booth selector) of NAND gates to save some logic gates but 1-bit left shift operation is performed by appending '1' at the least significant bit (LSB) position in place of '0'.

Besides, the analysis made in Subsection 2.2 for the 4-to-1 word level multiplexer, which is basically used to generate the first partial product, reveals that its one of the input is connected to 0 value. This observation has been used to optimize the multiplexer circuit and proposes encoder-cum-selector circuit type-2 for the generation of first partial product PP_0 as shown in Figure 7b. In the proposed circuit, 1-bit left shift operation is performed by appending '1' at the LSB position in place of '0'. To observe the effectiveness of the proposed Booth encoder-cum-selector circuits (type-1 and type-2), theoretical comparisons of delay and area complexities are made with the existing Booth encoder circuit and 4-to-1 word level multiplexer. The values given in Table 1 are used to calculate the delay and area for the proposed and existing circuits. Calculated values are listed in Table 3 for comparison. From Table 3, it is observed that the proposed Booth encoder-cum-selector type-1



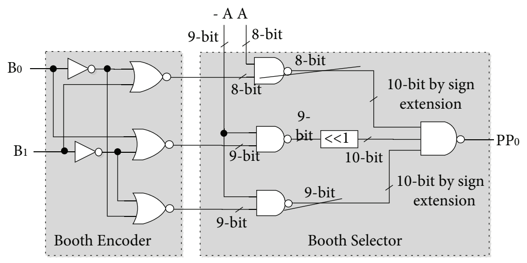


Figure 7. Proposed Booth encoder-cum-selector circuit (a) Type-1 for the generation of PP_i, where $1 \leq i \leq 3$, (b) Type-2 for the generation of PP_0.

involves 18.91% less delay and 12.05% less area over the existing encoder circuit of [24]. Similarly, the proposed encoder-cum-selector type-2 offers a saving of 23.13% in delay and 57.14% in area compared to the existing 4-to-1 word level multiplexer of [24]. Therefore, the proposed circuits could be helpful for the delay and area optimization of the Booth multiplier.

Table 3. Comparison for area and delay of the proposed and existing partial product generation units.

Design	Delay (ps)	Area (TC)
Existing booth encoder [24]	112.1	282
Proposed BECS type-1	90.9	248
Existing 4-to-1 word-MUX [24]	69.8	420
Proposed BECS type-2	53.65	180

BECS: Booth encoder-cum-selector, TC: transistor counts.

3.3. Proposed partial products addition unit

Based on the analysis given in Subsection 2.3, it is observed that CLA-CSLA–based adder tree unit involves large logic resources due to presence of redundant logics. An alternate approach for the addition of partial product could be helpful to reduce the logic resources. Therefore, in this subsection, the partial product addition unit which performs the addition of four partial products as shown in Figure 8 is proposed. It comprises a compressor module and an optimized CLA-CSLA adder. The four partial products (PP_3, PP_2, PP_1, PP_0) are compressed to two using compressor module and added by the optimized 10-bit CLA-CSLA adder to get the final sum. Detailed descriptions of the compressor module and the optimized adder are discussed below.

3.3.1. Compressor module

The compressor module as shown in Figure 8, comprises nine 4:2, two 3:2, and two 2-to-1 compressors.

The critical path delay of this module mainly depends on the delay of 4:2 compressors. The critical path of the existing 4:2 compressor of [26] (as shown in Figure 9a) consists of three 2-input XOR gates. It can be seen from Table 1 that the delay of XOR is higher than those of other gates.

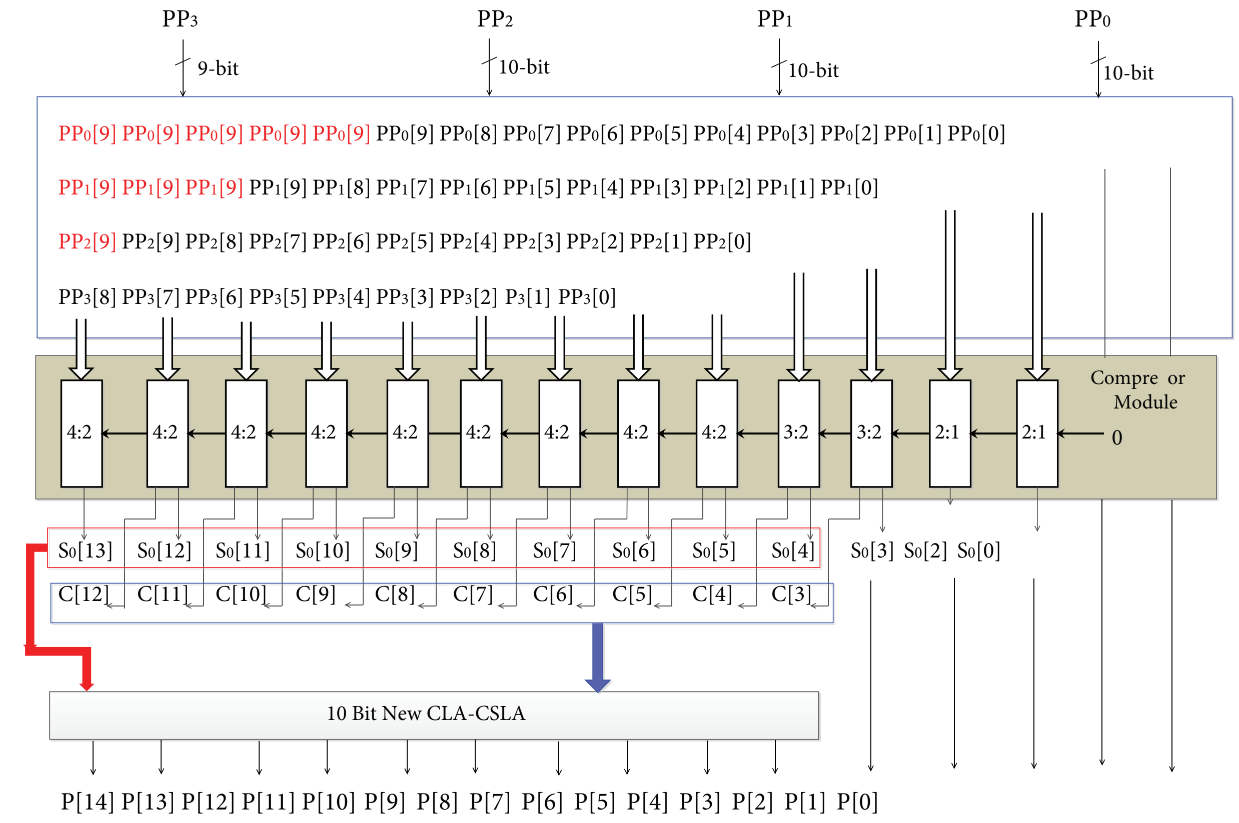

PP3
9-bit
PP2
10-bit
PP1
10-bit
PP0
10-bit
PP0[9] PP0[9] PP0[9] PP0[9] PP0[9] PP0[9] PP0[8] PP0[7] PP0[6] PP0[5] PP0[4] PP0[3] PP0[2] PP0[1] PP0[0]
PP1[9] PP1[9] PP1[9] PP1[9] PP1[8] PP1[7] PP1[6] PP1[5] PP1[4] PP1[3] PP1[2] PP1[1] PP1[0]
PP2[9] PP2[9] PP2[8] PP2[7] PP2[6] PP2[5] PP2[4] PP2[3] PP2[2] PP2[1] PP2[0]
PP3[8] PP3[7] PP3[6] PP3[5] PP3[4] PP3[3] PP3[2] P3[1] PP3[0]
4:2
3:2
2:1
Compre or Module
0
So[13] So[12] So[11] So[10] So[9] So[8] So[7] So[6] So[5] So[4] So[3] So[2] So[0]
C[12] C[11] C[10] C[9] C[8] C[7] C[6] C[5] C[4] C[3]
10 Bit New CLA-CSLA
P[14] P[13] P[12] P[11] P[10] P[9] P[8] P[7] P[6] P[5] P[4] P[3] P[2] P[1] P[0]

Figure 8. Proposed partial product addition unit.

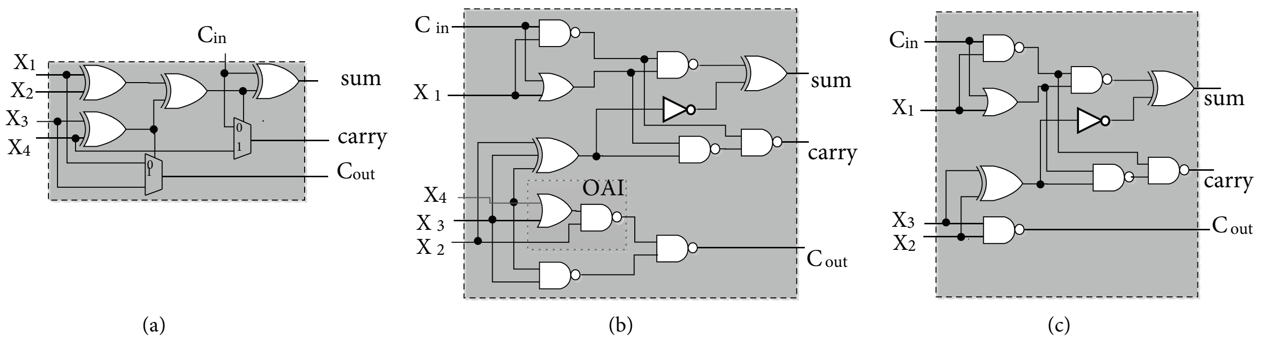


Figure 9. (a) Existing 4:2 compressor circuit [26], (b) proposed 4-to-2 compressor circuit, (c) proposed 3-to-2 compressor circuit with carry-in and carry-out.

Therefore, the delay optimization of 4:2 compressor circuit may reduce the overall delay of compressor module. Based on this observation, the delay-optimized 4:2 and 3:2 compressor circuits are proposed as shown in Figures 9b and 9c, respectively. In the compressor module, it seems that the carry is rippling through the compressor stages. To observe this, the cascaded structure of two 4:2 compressors is considered as shown in Figure 10. From Figure 10, it can be seen that the critical path (shown with red dotted line) involves one OAI, two NAND, one OR, and one XOR gates. The study of critical path reveals that the input carry is not rippling. To observe the delay efficiency of the proposed compressor module, theoretical comparison is made.

For theoretical comparison, delays of the proposed and existing 4:2 compressor-based compressor modules are calculated using delay values of gates given in Table 1. The calculated delay values of the proposed and existing 4:2 compressors are 113.55 ps and 126 ps, respectively. Besides, 2:1 compressor is designed using full-adder. From this delay calculation, it is found that the proposed 4:2 compressor-based compressor module involves 9.88% less delay over the existing 4:2 compressor-based [26] compressor module.

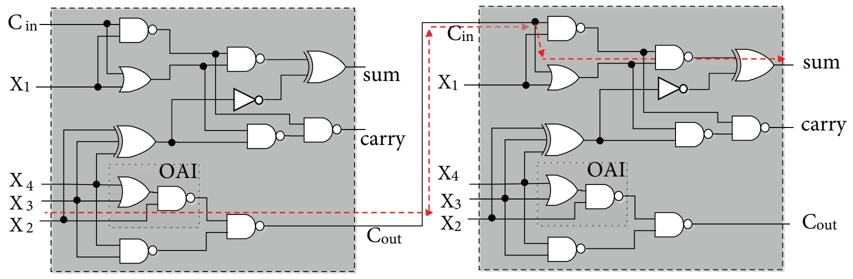


Figure 10. Cascaded 4:2 compressors for delay analysis.

3.3.2. Optimized CLA-CSLA adder

From the analysis given in Subsection 2.3, it is observed that existing CLA-CSLA [24] adder contains many redundant terms which are common between CLA with carry-in ‘0’ and CLA with carry-in ‘1’ logic expressions. The removal of these common terms from the logic expressions may reduce the logic resources significantly. The sharing of common logic resources between CLA with carry-in ‘0’ and CLA with carry-in ‘1’ is only possible when the logic expressions of CLA with carry-in ‘1’ are reformulated. Therefore, reformulated logic expressions of 3-bit CLA with carry-in ‘1’ are given as:

CLA with cin = 1 :

$$c_2^1 = c_2^0 + p_2 \tag{28}$$

$$c_3^1 = c_3^0 + p_3 \cdot p_2 \tag{29}$$

$$c_4^1 = c_4^0 + p_4 \cdot p_3 \cdot p_2 \tag{30}$$

$$s_2^1 = \overline{p_2}; \quad s_3^1 = p_3 \oplus c_2^1; \quad s_4^1 = p_4 \oplus c_3^1 \tag{31}$$

Using the reformulated expressions of CLA with carry-in ‘1’ (Eqs. (28)–(31)) and the logic expressions of CLA with carry-in ‘0’ (Eqs. (10)–(15)), an optimized structure of 10-bit CLA-CSLA is proposed as shown in Figure 11. In this structure, the PG block generates a set of propagate and generate signals (p, g), look-ahead carry generator (LACG)-0 block produced carries (c^0) with cin-‘0’ using p and g signals, LACG-1 block produced carries (c^1) with cin-‘1’ using p and c^0 signals. The SG-0 block computes a set of sum bits (s^0) using propagate bits (p) and carries c^0, whereas the SG-1 generates a set of sum bits (s^1) using propagate bits (p) and carries c^1. Finally, the word-level 2-to-1 multiplexer selects sum bits and carry-out bit for the generation of final sum bits and carry output. Since most of the redundant logics are removed in the proposed structure, it could be helpful to realize the area-efficient Booth multiplier structure.

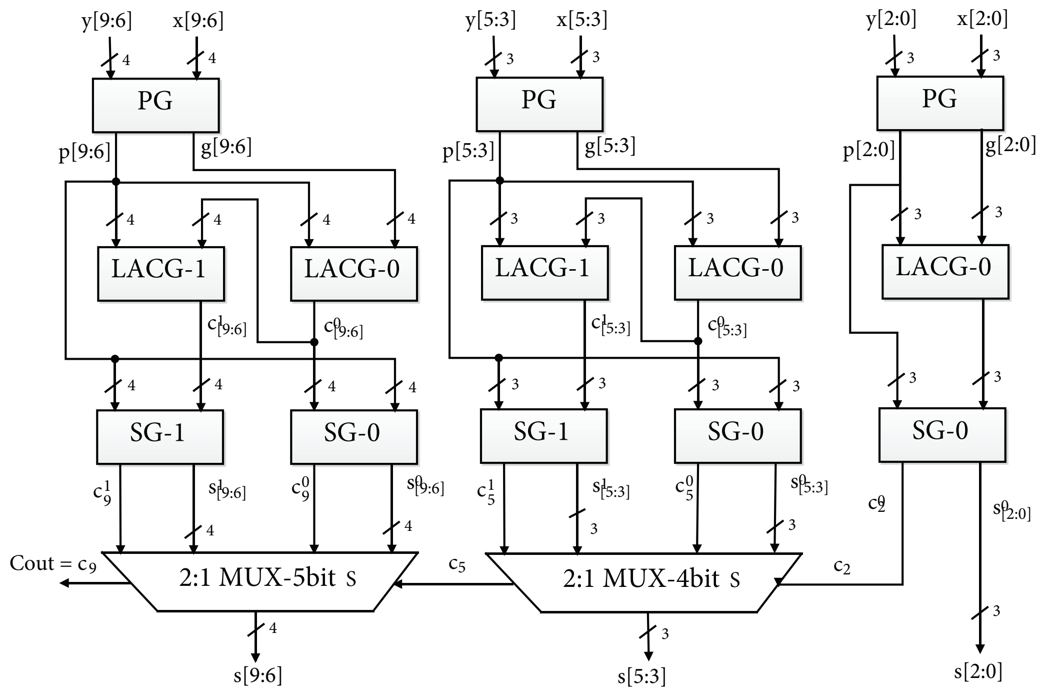
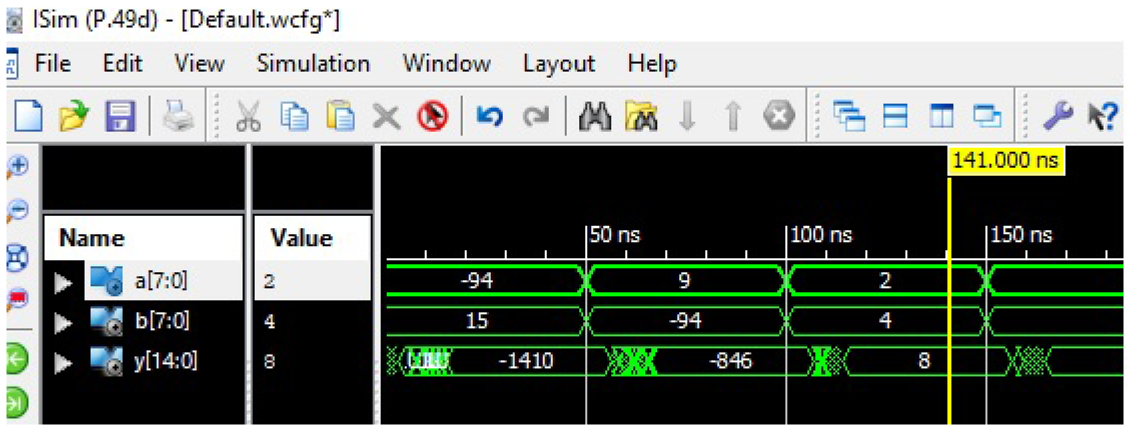


Figure 11. Proposed 10-bit CLA-CSLA adder.

4. Functional verification and performance comparison

For the functional verification, the proposed radix-4 8×8 Booth multiplier design is coded in VHDL and simulated on Xilinx ISim tool by applying about 1000 random inputs through test-bench at intervals of 50 ns. The simulated result (shown in Figure 12) validates the functional correctness of the proposed multiplier design.


ISim (P.49d) - [Default.wcfg*]
File Edit View Simulation Window Layout Help
141.000 ns
50 ns
100 ns
150 ns

Name	Value			
a[7:0]	2	-94	9	2
b[7:0]	4	15	-94	4
y[14:0]	8	-1410	-846	8

Figure 12. Simulation results for the proposed radix-4 8×8 Booth multiplier.

For the performance comparison, the proposed radix-4 8×8 Booth multiplier design, recent radix-4 8×8 Booth multiplier design of [24], and designs (version-I and version-II) of [23] are considered. All these are coded in VHDL and synthesized in Synopsys Design Compiler (SDC) using 65nm CMOS library. The delay, area, and power obtained from the SDC are presented in Table 4 for comparison. It can be seen from Table 4 that the proposed multiplier design offers a saving of 16.39% in delay and 23.41% in area compared to the design of [24]; 19.05% in delay and 9.44% in area compared to the design version-I of [23]; 13.56% in delay and 34.87% in area compared to the design version-II of [23]. To observe the overall effectiveness of the proposed multiplier design, the area-delay-product ($Area \times Delay$) and energy ($Power \times Delay$) parameters are calculated and listed in Table 4 for comparison. From Table 4, it can be seen that the proposed multiplier design offers a saving of 36.0% in area-delay-product (ADP) and 22.6% in energy compared to the design of [24]; 26.7% in ADP and 5.5% in energy compared to the design version-I of [23]; 43.7% in ADP and 11.24% in energy compared to the design version-II of [23]. Therefore, the proposed Booth multiplier design could be a good candidate for the area and delay efficient signal and image processing VLSI systems.

Table 4. Comparison of ASIC synthesis results for Booth multipliers.

Design	Delay (ns)	Area (μm^2)	Power (μW)	ADP($ns \times \mu m^2$)	Energy (fJ)
BM [24]	0.61	1816.12	639.64	1107.83	390.18
BM-I [23]	0.63	1536.04	507.15	967.71	319.50
BM-II [23]	0.59	2135.88	576.46	1260.17	340.11
Proposed	0.51	1391.04	591.93	709.43	301.89

BM: Booth multiplier, ADP = area×delay, energy = power×delay.

5. Conclusion

In this paper, a detailed analysis of the state-of-the-art Booth multiplier architecture is presented. Based on the finding of analysis, optimized new B2C, Booth encoder-cum-selector type-1 and type-2, and partial product addition units are proposed. Furthermore, using these optimized units, an efficient parallel radix-4 8×8 Booth multiplier architecture is proposed. The simulation is carried out to verify the functionality of the proposed design. The synthesis results show that the proposed structure offers a saving of 13.56% in delay and 34.87% in area compared to the recent similar Booth multiplier design. Comparison results also reveal that the proposed Booth multiplier design involves 43.7% less ADP and consumes 11.24% less energy compared to the recent Booth multiplier design. Therefore, the proposed Booth multiplier design could be helpful for the efficient realization of the high-performance DSP systems.

References

[1] Jiang H, Liu L, Jonker PP, Elliott DG, Lombardi F et al. A high-performance and energy-efficient FIR adaptive filter using approximate distributed arithmetic circuits. IEEE Transactions on Circuits and Systems I: Regular Papers 2018; 66 (1): 313-326. doi: 10.1109/TCSI.2018.2856513

[2] Durmuş B, Yavuz G, Aydın D. Adaptive iir filter design using self-adaptive search equation based artificial bee colony algorithm. Turkish Journal of Electrical Engineering & Computer Sciences 2019; 27 (6): pp. 4797-4817. doi: 10.3906/elk-1809-83

[3] Mittal A, Nandi A, Yadav D. Comparative study of 16-order FIR filter design using different multiplication techniques; IET Circuits, Devices & Systems 2017; 11 (3): 196-200. doi: 10.1049/iet-cds.2016.0146

[4] Chen KH, Chiueh TD. A low-power digit-based reconfigurable FIR filter. IEEE Transactions on Circuits and Systems II: Express Briefs 2006; 53 (8): 617-621. doi: 10.1109/TCSII.2006.875373

[5] Mohanty BK, Patel SK. Efficient very large-scale integration architecture for variable length block least mean square adaptive filter. IET Signal Processing 2015; 9 (8): 605-610. doi: 10.1049/iet-spr.2014.0424

[6] Meher PK. Seamless pipelining of DSP circuits. Circuits, Systems, and Signal Processing 2016; 35 (4): 1147-1162. doi: 10.1007/s00034-015-0089-2

[7] Gao W, Huang T, Reader C, Dou W, Chen X. IEEE standards for advanced audio and video coding in emerging applications. Computer 2014; 47 (7): 81-83. doi: 10.1109/MC.2014.122

[8] Wang G, Shield J. The efficient implementation of an array multiplier. In: 2005 IEEE International Conference on Electro Information Technology; Lincoln, NE, USA; 2005. 10.1109/EIT.2005.1626958

[9] Shanmuganathan R, Brindhadevi K. Comparative analysis of various types of multipliers for effective low power. Microelectronic Engineering 2019; 214: 28-37. doi: 10.1016/j.mee.2019.04.015

[10] Behrooz P. Computer Arithmetic: Algorithms and Hardware Designs. Oxford, England, UK: Oxford University Press, 2000.

[11] Sjalander M, Larsson-Edefors P. High-speed and low-power multipliers using the baugh-wooley algorithm and HPM reduction tree. In: IEEE International Conference on Electronics, Circuits and Systems; St. Julien's, Malta; 2008. pp. 33-36. doi: 10.1109/ICECS.2008.4674784

[12] Xue H, Patel R, Boppana N, Ren S. Low-power-delay-product radix-4 8×8 booth multiplier in CMOS. Electronics Letters 2018; 54 (6): pp. 344-346. doi: 10.1049/el.2017.3996

[13] Cooper A. Parallel architecture modified booth multiplier. In: Proceeding IEE G, Electronic Circuits and Systems; IET 1988; 135 (3): 125-128. doi: 10.1049/ip-g-1.1988.0019

[14] Elguibaly F. A fast parallel multiplier-accumulator using the modified booth algorithm. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing 2000; 47 (9): 902-908. doi: 10.1109/82.868458

[15] Kuang SR, Wang JP, Guo CY. Modified booth multipliers with a regular partial product array. IEEE Transactions on Circuits and Systems II: Express Briefs 2009; 56 (5): 404-408. doi: 10.1109/TCSII.2009.2019334

[16] Prabhu A, Elakya V. Design of modified low power booth multiplier. In: 2012 International Conference on Computing, Communication and Applications; Dindigul, Tamilnadu, India; 2012. doi: 10.1109/ICCCA.2012.6179166

[17] Nagamani A, Nikhil R, Nagaraj M, Agrawal VK. Reversible radix-4 booth multiplier for DSP applications. In: 2016 International Conference on Signal Processing and Communications (SPCOM); Bangalore, India: 2016. doi: 10.1109/SPCOM.2016.7746687

[18] Cho KJ, Lee KC, Chung JG, Parhi KK. Design of low-error fixed-width modified booth multiplier. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 2004; 12 (5): 522-531. 10.1109/TVLSI.2004.825853

[19] Shun Z, Pfander OA, Pfleiderer HJ, Bermak A. A VLSI architecture for a run-time multi-precision reconfigurable booth multiplier. In: 14th IEEE International Conference on Electronics, Circuits and Systems; Marrakech, Morocco; 2007, pp. 975-978. doi: 10.1109/ICECS.2007.4511155

[20] Kuang SR, Wang JP. Design of power-efficient configurable booth multiplier. IEEE Transactions on Circuits and Systems I: Regular Papers 2010; 57 (3): 568-580. doi: 10.1109/TCSI.2009.2023763

[21] Mohanty BK, Tiwari V. Modified PEB formulation for hardware-efficient fixed-width booth multiplier. Circuits, Systems, and Signal Processing 2014; 33 (12): 3981-3994. doi: 10.1007/s00034-014-9843-0

[22] Qian L, Wang C, Liu W, Lombardi F, Han J. Design and evaluation of an approximate Wallace-booth multiplier. In: 2016 IEEE international symposium on circuits and systems (ISCAS); Montreal, QC, Canada; 2016. pp. 1974-1977. doi: 10.1109/ISCAS.2016.7538962

[23] Pramod P, Shahana T. Efficient modular hybrid adders and radix-4 booth multipliers for DSP applications. Microelectronics Journal 2020. doi: 10.1016/j.mejo.2020.104701

[24] Boppana N, Kommareddy J, Ren S. Low-cost and high-performance 8×8 booth multiplier. Circuits, Systems, and Signal Processing 2019; 38 (9): 4357-4368.

[25] "dbtcbn65gplusbc0d88 TSMC 65nm CMOS library databook".

[26] Silveira B, Paim G, Abreu B, Grellert M, Diniz CM et al. Power-efficient sum of absolute differences hardware architecture using adder compressors for integer motion estimation design. IEEE Transactions on Circuits and Systems I: Regular Papers 2017; 64 (12): pp. 3126-3137. doi: 10.1109/TCSI.2017.2728802