Research Article GU J Sci 33(1): 90-104 (2020) DOI: 10.35378/gujs.434806

Gazi University

Journal of Science

http://dergipark.gov.tr/gujs

Low-Error Reconfigurable Fixed-Width Multiplier for Image Processing Applications

Jean Jenifer Nesam JEYAKUMAR, Sivanantham SATHASIVAM*

School of Electronics Engineering, Vellore Institute of Technology, Vellore – 632 014, India.

Highlights

- Design of low-error fixed-width multiplier for image processing applications.
- New area-efficient algorithm for unsigned multiplication.
- Performed in three different modes.
- Reconfigurable from fixed-width to exact multiplier.
- Image multiplication evaluates the performance of the multiplier.

Article Info

Received: 21/06/2018
Accepted: 28/07/2019

Keywords

Fixed-Width multiplier
Reconfigurable multiplier
NBPR algorithm
Image processing applications

Abstract

Compensating the error using additional circuitry is mandatory in a low-error fixed-width multiplier. Instead of compensating the error, reconfiguring n-bit fixed-width multiplier to n/2-bit error-free full-width multiplier using decomposed multiplication is proposed in this paper. The decomposed block multiplication using an area-efficient New Bit Pair Recoding (NBPR) algorithm in fixed-width mode shows a relatively lesser truncation error than existing truncated multipliers. Reconfigurable 16×16 NBPR multiplier in three different modes $(8\times8, 16\times8, 16\times16)$ with a fixed 16-bit product is verified on the TSMC 65nm CMOS standard cell library. The experimental results show that the NBPR multiplier consumes a lesser area than standard Booth multipliers. Evaluating the proposed multiplier in imaging shows improved PSNR with minimal error compared to other fixed-width multipliers.

1. INTRODUCTION

Fixed width multiplication is a mandatory arithmetic computation in digital signal and image processing applications. The fixed-width multipliers get two n-bit inputs and produce the n-bit product. This can be achieved by truncating a few least significant parts of partial products (PPs) matrix [1,2]. The Post-Truncated (PT) fixed-width multiplier round off the product after the completion of multiplication. PT provides better accuracy but occupies a large area. The Direct-Truncated (DT) fixed-width multiplier directly truncates the n-least significant PPs without error compensation. DT occupies less area but provides less accuracy. For an area-efficient, low error fixed-width multiplier, the truncation error is compensated using various techniques. Kidambi et al. [3] introduced a simple constant bias technique that reduced the error by adding constant bias to the retained adder cells. But the error can still be reduced. Later, the Variable Correction Technique (VCT) [4] and data-dependent compensation to compensate for the error have been developed [5-8]. In VCT, the unused PP bits are used to define the compensation vector. The calculated compensation vector is added with the product to reduce the error. Juang et al. [9] developed efficient carry generating circuits that generate the carry and feed the carry as an input to the non-truncated adder cells to compensate for the error. Flexible error based multiplier and multiply-add design have been developed for area reduced error-resilient applications [10]. Self-tuned error compensation is an efficient method to reduce the truncation error [11-13]. The area reduction multiplication using an approximation for image processing application is developed in [14,15].

*Corresponding author, e-mail: ssivanantham@vit.ac.in

All error compensated multipliers provided in this literature require an additional complex circuit to find the amount of compensation vector. Albeit, it shows relatively more error for image processing applications. In this paper, the fixed-width multiplier performs without error compensation and provides the product accuracy better than error compensated fixed-width multipliers. The multiplier is reconfigurable to the exact multiplier using a control bit that provides distortion less application. In an inexact mode, the proposed multiplier eliminates the lower bits of the least decomposed block and the non-truncated higher bits are added to the higher blocks. This helps to reduce the truncation error with a slight increment in the area.

This area overhead is balanced by using an efficient New Bit Pair Recoding (NBPR) algorithm. NBPR reduces the PP row height without computing Negative Encoding (NE) and Sign Extension (SE). NE and SE are the common terms in existing bit pair recoding algorithms such as Booth [16-18] that incurred additional PP row which in turn occupies more area, power, and delay. NBPR generates the PP based on the number of 1's in 4-bit recoded multiplier groups. The performance benefits of NBPR is given in section 2. Finally, the developed multiplier is applied to image processing applications such as two different image multiplication and image color conversion. The behavioral simulation models are developed in Matlab to evaluate the performance of multiplier in imaging. The quality of the image is analyzed with its Peak Signal to Noise Ratio (PSNR) and compared with various truncated multipliers.

The rest of the paper is organized as follows; section 2 describes the performance benefits of the NBPR multiplier over existing algorithm based multiplication. Section 3 gives the detail about reconfigurable multiplication. The introduced error due to truncation is analyzed in section 4. Section 5 for results and comparison and section 6 explains the suitability of the proposed multiplier for different image processing applications. Section 7 concludes the paper.

2. NEW BIT PAIR RECODING (NBPR) ALGORITHM

2.1. The NBPR Recoding

Let us consider, 'Y' is the multiplier and NBPR recodes the multiplier Y as follows:

$$Y_j = \{y_{2i+3}, y_{2i+2}, y_{2i+1}, y_{2i}\} \tag{1}$$

where, $i = 2j - 2 \; and \; j = 1, 2, 3 \dots \frac{n}{4}$.

In Equation (1), the 'j' represents the number of recoded 4-bit groups. Based on the requirement of 'j', 'i' takes the value defined by 2j-2. For an example, 12-bit multiplier defines j=1,2,3 and i=0,2,4. For each j and i value the multiplier is recoded as follows in Equation (2)

$$Y_3 = \{y_{11}\, y_{10}\, y_9\, y_8\};\; Y_2 = \{y_7 y_6\, y_5\, y_4\};\; Y_1 = \{y_3\, y_2\, y_1 y_0\}; \tag{2}$$

Based on the recoded group the NBPR selects the PDVs for PP reduction.

2.2. PP Reduction Using NBPR Algorithm

NBPR recoded 4-bit group performs the PP reduction using the diverse modules namely, PDV (PDV) generation, select logic (recoding), Merged PP addition. The following sub-section details each phase individually:

2.2.1. PDV Generation

This module takes in two binary numbers for multiplication and recodes the multiplier operand as a non-overlapped 4-bit group. Prior to multiplication, this module pre-defines a few values. Considering 'X' as the multiplicand ($X = x_{n-1}x_{n-2}...x_2x_1x_0$) and the PDVs are assumed as follows:

1. The representation of '0' in Table 1 is computed $0 \times X$.
2. The other representations namely A, B, C, D are obtained by concatenating 0's in a different position with multiplicand as below Equation (3),

$$\begin{aligned} A &= 000x_{n-1}x_{n-2}...x_2x_1x_0; \\ B &= 00x_{n-1}x_{n-2}...x_2x_1x_0 0; \\ C &= 0x_{n-1}x_{n-2}...x_2x_1x_0 00; \\ D &= x_{n-1}x_{n-2}...x_2x_1x_0 000; \end{aligned} \tag{3}$$

3. Adding 'A' and 'B' to generate 'E' value
4. The summed up value is left-shifted twice ((A+B)<<2) and represented as 'F'

The values of A, B, C, D resemble the bit combinations of "0001", "0010", "0100" and "1000" respectively and the values are generated by replacing the 1's position with the multiplicand and without changing the 0's position. The "0011" bit combination is generated as the result of adding 'A' and 'B'. The bit combination "1100" is left-shifted twice to attain "0011". The computed PDVs are listed in Table 1 and the generated PPs using the PDVs are provided in Table 2.

Table 1. *PDVs of NBPR algorithm*

Bit combination	Operation	Representation
0000	$0 \times X$	0
0001	Replace the 1's position by the multiplicand	A
0010		B
0100		C
1000		D
0011	Add "0001" and "0010"	E
1100	2-bit left shift of "0011"	F

Table 2. *PDV selection using the NBPR Algorithm*

Recoded bits		Z1	Recoded bits		Z2
y_{i+1}	y_i		y_{i+3}	y_{i+2}	
0	0	0	0	0	0
0	1	A	0	1	C
1	0	B	1	0	D
1	1	E	1	1	F

2.2.2. Merged Addition

The two sets of 4-bit recoded group generates two (n+3)-bit size intermediate PPs. Thereby, merged addition on intermediate PPs reduces the number of PP rows from n to n/4. The pictorial representation of pp reduction using the NBPR algorithm is shown in Figure 1. The flowchart representation of the NBPR multiplier is provided in Figure 2. However, the 8×8 NBPR multiplier generates two PPs, it skips the reduction phase and directly fed to the final addition phase. NBPR uses adder tree in Carry Save Addition (CSA) format for n>8-bit. The final addition stage uses Carry Propagate Adder (CPA) to add the sum and carry.

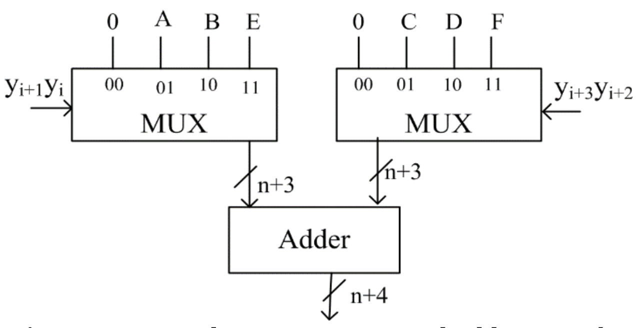

Figure 1. *Merged PP generation and addition in the NBPR algorithm*

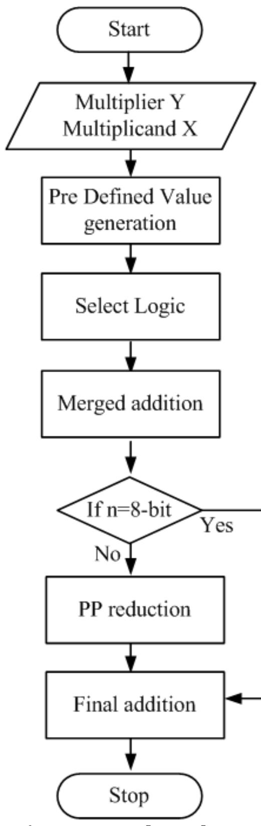

Figure 2. *Flowchart for NBPR multiplier*

2.3. Realization of NBPR Algorithm for n=8-Bit Multiplication

Consider two unsigned binary numbers X=10101100 and Y=11101001. Multiplication X is realized using the NBPR algorithm in the following steps. The diagrammatic representation of the NBPR algorithm for 8-bit multiplication is given in Figure 3.

Step 1: PDV generation

$$A = 00010101100;$$
$$B = 00101011000;$$
$$C = 01010110000;$$
$$D = 10101100000;$$
$$E = 10101100 + (\ 10101100\ ) \times 2^1$$
$$= 1000000100$$

$$F = E << 2 = 100000010000;$$

Step 2: Select Logic (Recoding)

$$Y_1 = y_3 y_2 y_1 y_0 = 1001;$$
$$Y_2 = y_7 y_6 y_5 y_4 = 1110;$$

MUX based selection of predefined value for the recoded bits

$$MUX1 = \begin{cases} Z1 = A = 00010101100 & y_1 y_0 = 01 \\ Z2 = D = 10101100000 & y_3 y_2 = 10 \end{cases}$$
$$MUX2 = \begin{cases} Z1 = B = 00101011000 & y_5 y_4 = 10 \\ Z2 = F = 100000010000 & y_7 y_6 = 11 \end{cases}$$

Step 3: Merged Addition/PP reduction (Z1 + Z2)

$$PP1 = 11000001100;$$
$$PP2 = 100101101000;$$

Step 4: final Addition

The second PP (PP2) is placed 4-bit left to the first PP (PP1) and Carry Propagate Adder (CPA) is applied for the final addition to producing a 16-bit product

$$\begin{aligned} Final\ product &= PP1 + 4-bit\ left\ shifted\ PP2 \\ &= 11000001100 + (100101101000) \times 2^4 \\ &= 1001110010001100 \end{aligned}$$

2.4. Decomposed 16x16 Multiplier Using 8x8 NBPR Multiplier

The decomposed 16×16 multiplication split the 16-bit operand into two 8-bit sub-blocks and requires four 8×8 NBPR multiplication as provided in Figure 4 to compute the 16×16 multiplier. After the sub-block multiplication, the appropriate blocks are added and the addition of overlapping bits is performed as shown in Figure 5.

2.5. Final Addition

The final addition is computed via Carry Propagating Addition (CPA). The CPA length of Wallace depends on the number of stages involved in the PP reduction phase. If S is the number of stages and *N* is the number of bits in the operands, the final CPA length of Wallace is defined as in Equation (4)

$$\begin{aligned} &for\ 3 \le \mathrm{N} \le 5, \\ &\qquad \text{CPA length}_{Wallace} = (2 \times N) - 2 - S \\ &and\ \text{if N} > 5 \\ &\qquad \text{CPA length}_{Wallace} = (2 \times N) - 1 - S \end{aligned} \tag{4}$$

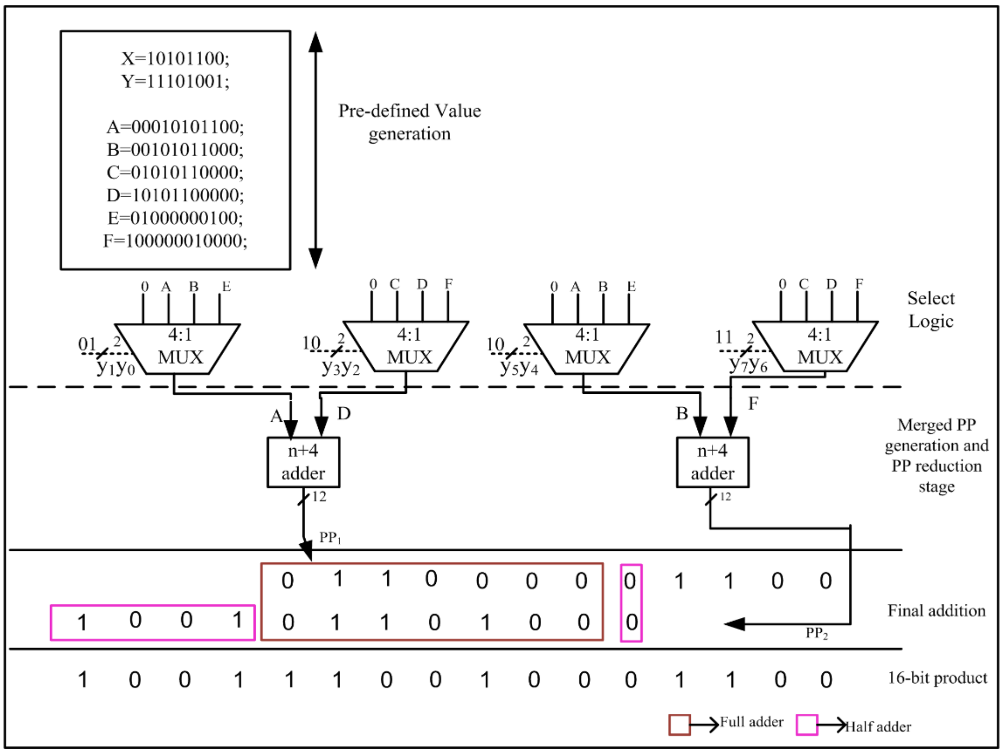

Figure 3. *8 × 8 multiplication using NBPR algorithm*

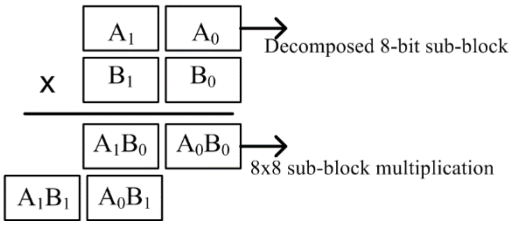

Figure 4. *Decomposed 16x16 NBPR multiplier*

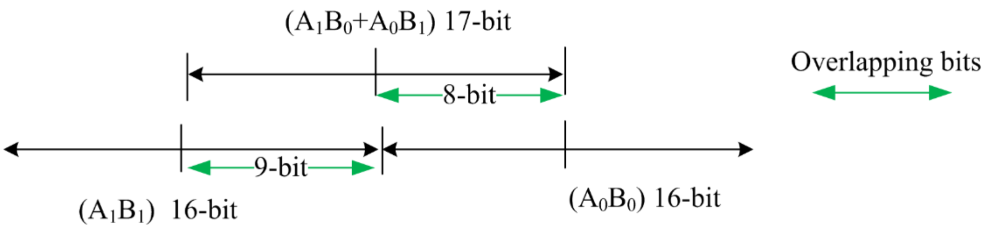

Figure 5. *Final addition for 16x16 NBPR multiplier*

A direct implementation of N-bit multiplier or Dadda N-bit multiplier’s final CPA length is defined as in Equation (5)

$$\text{CPA length}_{Dadda} = 2N - 2 \tag{5}$$

The proposed NBPR multiplier has 20% less CPA length and CPA size is determined by Equation (6)

$$\text{CPA length}_{\text{Proposed}} = 17i + 7 \tag{6}$$

where,

$$i = \begin{cases} \sim 0.23 & \text{for } 8\times 8 \text{ multiplier} \\ 1 & \text{for } 16\times 16 \text{ multiplier} \\ 2 & \text{for } 24\times 24 \text{ mutiplier} \\ 3 & \text{for } 32\times 32 \text{ multiplier} \end{cases}.$$

3. RECONFIGURABLE FIXED WIDTH MULTIPLIER

Small-sized and high accuracy applications need 8×8 exact multiplication, whilst, the large bit-sized error-resilient application uses the 16×8 or 16×16 inexact multiplication based on the requirement. A fixed width multiplier is the utmost common multiplier used in image processing applications. Many truncated methods available to maintain the 16×16 multiplier with the 16-bit product. Directly truncate the least part of the partial product matrix is the simple method that achieves more area reduction with large accuracy loss. This accuracy loss is compensated in various ways using variable or constant correction compensation vectors. These compensation methods consume more area.

This reconfigurable multiplier that can perform both exact and inexact multiplication without any area overhead. Distraction less image applications achieve in 8×8 exact mode albeit, the other two modes performing the application with less accuracy loss compared to existing truncated multiplier. The new reconfigurable multiplier is developed in this paper that can perform both exact multiplications without error and truncated multiplication with less error. In this proposed multiplier performs $n\times n, n\times \frac{n}{2}$ and $\frac{n}{2}\times \frac{n}{2}$ multiplication with the fixed n-bit product. This reconfigurability is achieved by using decomposed block multiplication and a 1-bit control bit. The decomposed multiplier plane for 16×16 multiplier using four 8×8 NBPR multiplier is shown in Figure 6.

In NBPR based fixed-width multiplier computes the multiplication in three different modes and the length of the data is modified based on the application requirement. Selecting differently sized sub-block multiplication as given in Figure 7, the reconFigured multiplication is performed. If A and B are two 16-bit operands, the control bits and the corresponding mode of multiplication are given in Table 3.

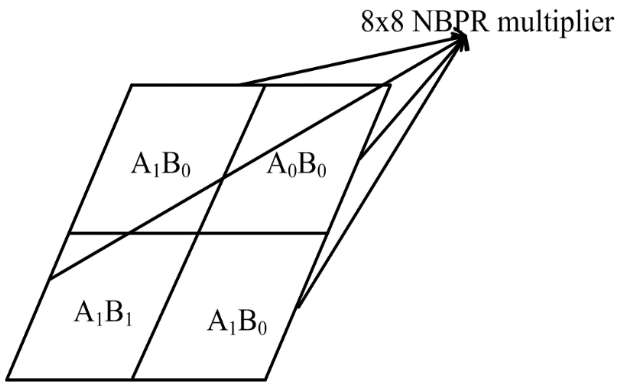

Figure 6. *Decomposed multiplier plane for 16 × 16 NBPR multiplier*

Table 3. *Control Bits and Corresponding Mode Selection*

Mode	Control bit	Input size		Output size
		A	B	
1	1	16-bit	16-bit	16-bit
2	0	8-bit	8-bit	
3	0	16-bit	8-bit	

In the proposed fixed-width multiplier, the output product size is fixed to 16-bit. The new NBPR multiplier can be performed as a 16×16 fixed-width multiplier ie). 16×16 multiplication with a 16-bit product in mode 1. In mode 2, it performs an exact 8×8 multiplication i.e). 8×8 multiplier with the 16-bit product, and truncated 16×8 multiplier with 16-bit product in mode 3.

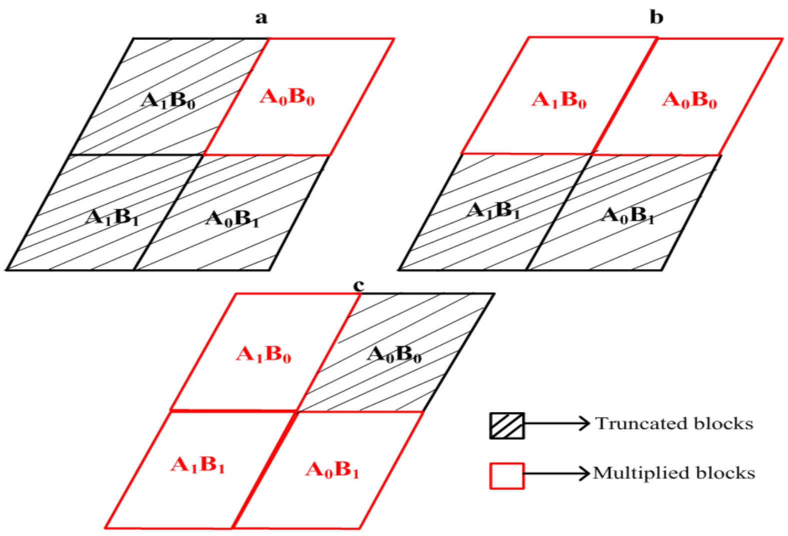

Figure 7. *Three modes of computation a). 8 × 8 full-width multiplier b). 16 × 8 truncated multiplier c). 16 × 16 truncated multiplier*

As shown in Figure 8, the 1-bit control bit selects the upper or lower half of the operand B values. If the control bit is '0', the 2:1 multiplexer selects the lower half of the B operand B_0 and (A_0B_0) NBPR 8×8 multiplication is performed. (A_0B_0) multiplication is considered as a 8×8 exact multiplier with 1the 6-bit product. Addition of (A_0B_0) and uthe pper half of the (A_1B_0) provides 16-bit truncated results for 16×8 multiplication. If the control bit is ' 1 ', the upper half of the B operand is B_1 selected and (A_1B_1), (A_0B_1) block multiplications are computed in mode 2. Direct (A_1B_0) multiplication, (A_1B_1) and (A_0B_1) blocks are added accordingly to provide fixed width 16×16 multiplication in mode 3.

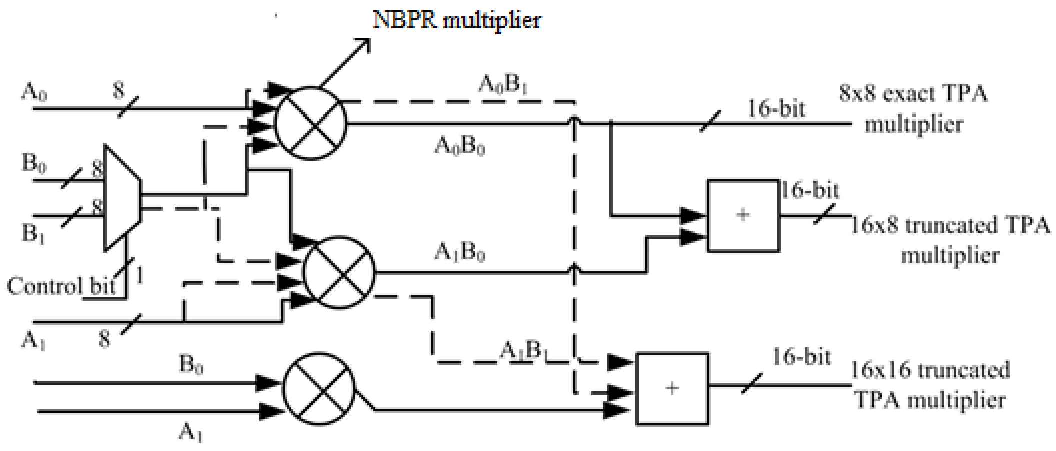

Figure 8. *Mode selection of NBPR multiplier*

4. ERROR ANALYSIS

The 16×16 multiplication is achieved by using four 8×8 sub-blocks. All sub-block multiplications are performed in parallel. The half of the product of each multiplied sub-blocks are overlapped to the next higher block and the truncation is performed only in the final addition stage. Keeping the overlapping bits and truncate the lower non-dependent bits help to reduce the truncation error. The generic product form of various modes of decomposed NBPR multiplier is given as in Equation (7).

$$\begin{aligned} P_{8\times 8} &= P_{A_0B_0};\\ P_{16\times 8} &= P_{A_1B_0} + P_{A_0B_0};\\ P_{16\times 16} &= P_{A_1B_1} + P_{(A_1B_0 + A_0B_1)} + P_{A_0B_0} \end{aligned} \tag{7}$$

Each $P_{A_xB_y}$ and $x, y \epsilon (0,1)$ in different modes refers one 8×8 NBPR multiplier. The error due to truncation in each mode is analysed in the following sub-section.

4.1. NBPR 8×8 Exact Multiplication

In 8×8 multiplication mode, it operates as an exact multiplier as given in Equation (8).

$$P_{8\times 8} = P_{A_0B_0} = \sum_{i=0}^{7} a_i \times \sum_{i=0}^{7} b_j \tag{8}$$

where (A_0B_0) is the 8×8 sub-block and $a_i, b_j \in (0,1)$ are individual bits inside the sub-block. The 8×8 multiplication provides $16-bit$ product without error. The maximum error is computed as the difference between the exact product and the truncated product. Since 8×8 is an exact multiplier, the truncation part is considered as equivalent to exact product and the maximum error is equal to zero and the value is computed as shown in Equation (9)

$$\varepsilon_{\max_{TPA8\times 8}} = \max_{A,B}(|P_{exact} - P_{truncated}|) = 0\ . \tag{9}$$

4.2. NBPR 16×8 Truncated Multiplier

In this mode, the multiplier gets two different sizes operand (16,8) for multiplication. The exact multiplication of 16×8 provides the 24-bit product. After the computation, truncating partial part of $P_{(A_0B_0)}$, the product size is fixed to 16-bit. The error limit in this mode is calculated as follows in Equation (10)

$$\begin{aligned} \mathrm{P}_{(16\times 8)} = \mathrm{P}_{\mathrm{A_1B_0}} + \mathrm{P}_{\mathrm{A_0B_0}} &= \sum_{\mathrm{i=8}}^{15} \mathrm{a_i} \times \sum_{j=0}^{7} b_j + \sum_{\mathrm{i=0}}^{7} \mathrm{a_i} \times \sum_{j=0}^{7} b_j \\ &= \sigma_{high} + \sigma_{low} \\ \mathrm{P}_{\mathrm{16x8(tr)}} &= \sigma_{high} + \sigma_{low(upper\ half)}\ . \end{aligned} \tag{10}$$

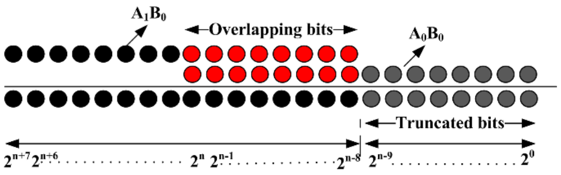
A₁B₀
Overlapping bits
A₀B₀
Truncated bits
2ⁿ⁺⁷ 2ⁿ⁺⁶ . . . . 2ⁿ 2ⁿ⁻¹ . . . . 2ⁿ⁻⁸ 2ⁿ⁻⁹ 2⁰

Figure 9. *The* $(n\ \times \frac{n}{2})$ *multiplication with* $n - bit$ *product* ($n = 16 - bit$)

The error limits can be computed from Figure 9, and the truncation error (ε_{tr}) of 16×8 the truncated multiplier is bounded as given in Equation (11),

$$2^0 \le \varepsilon_{tr} \le 2^{n-9} \tag{11}$$

where $n = 16 - bit$.

4.3. NBPR 16x16 Fixed-Width Multiplier

In this mode, the 16×16 multiplier discards the lower $(A_0 B_0)$ block without computation and truncate a few parts of the product $(A_0 B_1 + A_1 B_0)$ to fix the output product size to 16-bit.

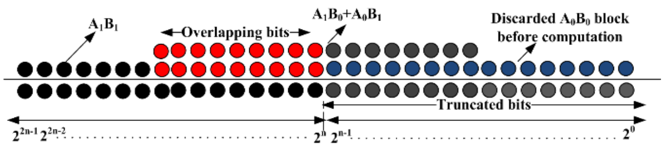

Figure 10. *The $(n \times n)$ multiplication with n- bit product ($n = 16 - bit$)*

The truncation performed as given in Figure 10 and the truncation error limit in this mode is computed as follows in Equation (12)

$$\begin{aligned} P_{16\times16_{fixed\ \mathrm{width}}} &= P_{A_1B_1} + P_{(A_1B_0+A_0B_1)} \\ &= \sum_{\mathrm{i}=8}^{15} \mathrm{a_i} \times \sum_{j=8}^{15} b_j + \Big(\sum_{\mathrm{i}=8}^{15} \mathrm{a_i} \times \sum_{j=0}^{7} b_j + \sum_{\mathrm{i}=0}^{7} \mathrm{a_i} \times \sum_{j=8}^{15} b_j\Big) \\ &= \sigma_{high} + \sigma_{mid(upper\ \mathrm{half})}. \end{aligned} \tag{12}$$

5. PERFORMANCE EVALUATION

The functionality of the presented multiplier is verified on Modelsim 6.5b. After the functional verification, the multiplier is synthesized using Cadence's RC RTL compiler in 180 and 65nm technology. The separate TCL and SDC script are written for each mode. The performance of the proposed design is individually evaluated in the following sections.

5.1. Analysis of NBPR Multiplier

Detailed area, power and delay analysis of NBPR multiplier with other bit pair recoding algorithm is provided in this section. The Modified Booth Encoding (MBE) is the common algorithm for area-efficient multiplication. MBE reduces the PP rows with the computation of NE and SE terms. Since NE and SE incurred more resources, many designs have been developed with the elimination of NE and SE. The negation of NE requires additional circuitry and its computations are highly complex. Even though the SE prevention techniques have been adopted, two SE bits till required in multiplication. From, the theoretical analysis of various Booth multipliers with NBPR claims the superiority of the proposed design.

The righteousness of theoretical analysis has been proven by its experimental results. Each module in the NBPR algorithm namely PDV generation ('E' value), recoder logic, PP reduction and final addition for 8, 16-bit multiplication is performed in 180 and 65nm technology and the results are provided in Table 4. The excel of the NBPR algorithm in terms of area, power, and speed is verified by computing the design for generating one bit of PP. The compared experimental results for generating one PP bit using various algorithms are provided in Table 5 and is the evidence that the NBPR uses only 72.5% less area and 18% less power with 34% increased speed when compared to the standard booth multipliers.

Table 4. *Synthesis report of various phases of NBPR multiplier (180 and 65nm)*

Mode of computation			n=8-bit				n=16-bit			
		nm	#cells	Area (μm^2)	Power (μW)	Delay (ns)	#cells	Area (μm^2)	Power (μW)	Delay (ns)
PDV generation		180	25	719	82.832	0.056	59	981	101.189	0.637
		65	24	61	8.187	0.044	31	132	1.316	0.081
Recoder	Select logic	180	113	2365	87.853	0.190	259	5409	870.269	2.173
		65	88	362	23.247	0.017	146	639	156.321	0.236
	merged pp	180	23	283	9.345	0.052	30	336	49.267	0.527
		65	22	44	2.128	0.056	31	49	9.844	0.056
PP reduction		180	-				253	4697	624.573	0.564
		65	-				94	558	169.290	0.067
Final adder		180	18	233	8.559	0.052	156	2262	217.123	8.772
		65	18	36	1.891	0.051	25	221	72.295	0.786

Table 5. *Area/power/delay report on generating one PP bit using MBE and NBPR in 65nm*

Design	#cells	Area (μm^2)	Power* (μW)	Delay (ps)
Standard Booth	4	14 (100%)	0.967(100%)	84 (100%)
Limberti [16]	6	16 (114.3%)	0.795 (84.7%)	74 (88%)
Antelo [17]	6	14 (100%)	0.735(83.4%)	61 (72.7%)
NBPR	3	10 (72.5%)	0.713(82%)	56 (66.7%)

**Power=Leakage power+Dynamic power*

5.2. Reconfigurable 16x16 NBPR Multiplier

The area, power and delay consumption of reconfigurable NBPR multiplier is verified in this section. The Verilog code has been written for reconfigurable design as in Figure 8 and its functionality has been verified in Modelsim 6.5b simulator. After the functional verification, the design has been implemented on Cadence TSMC 180 and 65nm with a standard cell library. The obtained results are provided in Table 6.

The area consumption of various truncated multipliers is compared with the NBPR multiplier in three different modes. The compared values on TSMC 180nm are provided in Table 7. The presented NBPR multiplier in 16×16 fixed-width multiplication mode occupies 33.15% less area when compared to standard Booth multiplier. In full-width mode 8×8 shows approximately 1% less area and in truncated 16×8 mode shows 54.65% less area compared to the standard multiplier. However, the NBPR multiplier occupies 15.5% more area compared to other direct truncated multipliers mentioned in the literature, due to the error reduced decomposed multiplication.

Table 6. *Implementation details of NBPR multiplier on 180 and 65 nm TSMC technology*

Technology	Area (μm^2)	Power (μW)	Delay (ns)	PDP (fJ)	APP ($\mu m^2.\mu W$)10^5
180nm	20654	1286.093	0.5	643.047	265.63
65nm	3614	220.176	0.5	110.088	7.96

Table 7. *Area consumption of various truncated multipliers on 180nm technology*

Design	Area (μm^2)	
	n=8	n=16
Standard multiplier	7092.164	30896.219
D-truncated	3337.495	15179.200
K-G-As' [6]	3299.660	15179.200
J-K-Cs' [1]	3640.836	16477.370
K-Ss'[4]	5604.888	20688.631
L.D. Vans' [3]	3640.83	16477.370
Proposed	7037 (8x8) exact	14011(16x8) tr
		20654(16x16) tr

Table 8. *Max Error (ε_{max}) for various* 16×16 *fixed width multiplier*

Design	Algorithm	ε_{max}	
		Bit width (n)	
		n=8	n=16
D-truncated	-	1793	983040
Post-truncated	-	128	32768
K-G-As' [6]	2's complement	1280	786432
J-K-Cs' [1]	Booth	514.94	219316.22
K-Ss' [4]	Sign-magnitude	263.03	111083.52
L.D Vans' [3]	Baugh-Wooley	441	220245
Yuan-Ho Chen [2]	2's complement	384.9	169181
Proposed	NBPR	0	128 (16 × 8)
			65537 (16 × 16)

5.3 Maximum Error $(\varepsilon_{\max})$ Computation

The maximum error $(\varepsilon_{\max})$ is theoretically computed as the difference between original results and actual results by applying its maximum value. The $\varepsilon_{\max}$ for 16×16 multiplier in different modes with various truncation depth (τ) value is given in Table 8. Compared to directly truncated (D-truncated) multiplier, the NBPR multiplier shows 93.33% reduction in error while, K-G-As', J-K-Cs', K-Ss', L.D Vans' and Yuan-Ho Chens' designs have error improvement of 20%, 77.69%, 88.7%, 79.63%, and 82.79% respectively. Hence the NBPR multiplier shows significant error improvement, compared to direct truncate and other truncating multipliers.

6. APPLICATIONS

6.1. Image Multiplication

The behavioral model of the multiplier is developed in Matlab and two 512×512 images are multiplied with different τ values. The τ value and the corresponding multiplied images are given in Figure 11. The quality of the resultant images is verified by using Peak Signal to Noise Ratio (PSNR). The PSNR is calculated by the Equation (13).

$$PeakSNR = PSNR\ (\ image_{tr}, image_{exact}\ )\ . \tag{13}$$

Table 9. *PSNR of a multiplied image using NBPR multiplier*

Images	PSNR(dB)		
	τ =0	τ =8	τ =16
image1	∞	54.5716	17.3624
image2	∞	54.42	15.3881
image3	∞	54.6148	14.2321

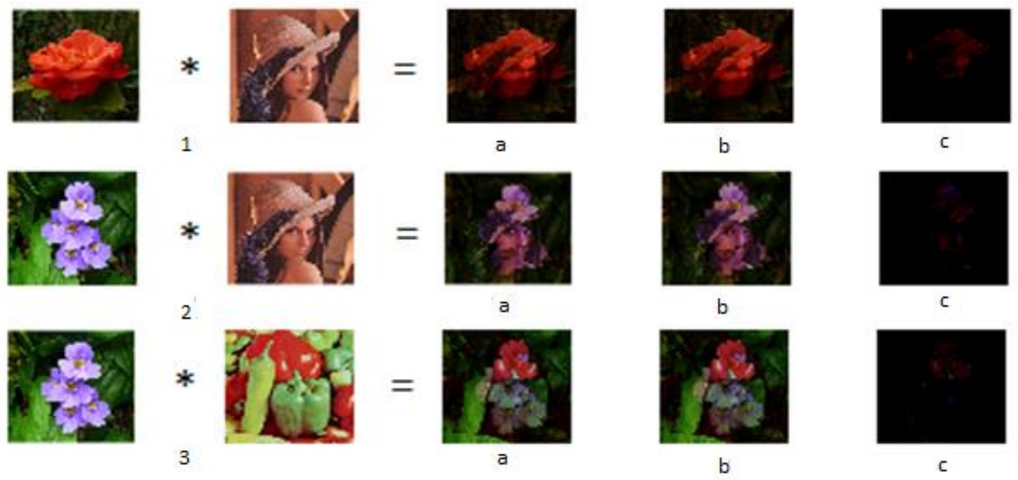

Figure 11. *Multiplied images 1). image1 2). image2 3). image3 with a). τ =0 b). τ =8 c). τ =16*

The PSNR is computed between the NBPR multiplier applied image and the exact multiplier applied image. The PSNR shows that the presented NBPR multiplier performs with better image quality. The PSNR values are given in Table 9. The various truncated multiplier based image multiplication in Table 10 is evident for the excellence of the proposed multiplier in imaging.

Table 10. *PSNR comparison*

Design	PSNR (dB)	
	τ =8	τ =16
DT	28.81	9.64
PT	33.45	11.34
K-G-As’ [6]	38.20	12.48
J-K-Cs’ [1]	39.46	13.65
K-Ss’ [4]	39.76	11.08
L.D Vans' [3]	34.01	10.30
Yuan-Ho Chens’ [2]	31.84	10.98
Proposed	54.61	14.23

6.2. Image Color Space Conversion (RGB to Gray)

The Red, Green and Blue values in the RGB color image has the integer range from 0 to 255. Multiplying R, G, and B value with different coefficient and adding the multiplied values together provides different color space. The NBPR multiplier in 8×8 exact multiplication mode is well suited for image color space conversion such as RGB color space to grayscale transformation. The RGB image is converted to a grayscale image using below Equation (14). The color-converted images from RGB to gray is given in Figure 12.

$$\text{Gray Image}=0.2989 \times R+0.587 \times G+0.114 \times B \tag{14}$$

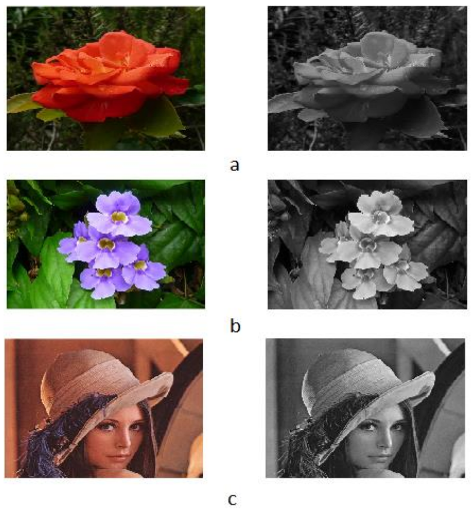

Figure 12. *RGB to Gray a. flower1 b. flower2 c. Lena*

7. CONCLUSION

Reconfigurable exact and inexact decomposed multipliers using a new algorithm for image processing applications have been presented in this paper. Based on the selection of different input sizes, the multiplier reconFigured from the exact 8×8 multiplier to inexact 16×8 and 16×16 multiplier. The error-free computation in 8×8 mode has provided the distortionless image quality, howbeit, the inexact other two modes $(16\times8, 16\times16)$ have provided 93.33% lesser error compared to directly truncated multipliers. Moreover, the exact 8×8 NBPR multiplier is consumed 33.15% lesser area compared to standard MBE multipliers. Finally, the detailed performance analysis of various color image applications showcased the outperformance of a new multiplier with minimal degradation of image quality when compared to existing fixed-width multipliers. This makes the proposed reconfigurable NBPR multiplier more amicable for image processing applications.

CONFLICTS OF INTEREST

No conflict of interest was declared by the authors.

REFERENCES

[1] Jou, S.J. and Wang, H.H., "Fixed-width multiplier for DSP application", Proc. IEEE International Symposium on Computer-Aided Control System Design , 318-332, (2000).

[2] Chen, Y.H., Lu, C.W., Chiang, H.C., Chang, T.Y., Hsia, C., "A low-error statistical fixed-width multiplier and its applications," International Symposium on Instrumentation & Measurement, Sensor Network and Automation (IMSNA), Sanya, 39-43, (2012).

[3] Van, L.D., Yang, C.C., "Generalized low-error area-efficient fixed width multipliers," IEEE Transactions on Circuits and Systems-I: Regular Papers, 52(8): 1608-1619, (2005).

[4] King, E.J., Swartzlander, E.E. Jr., “Data-dependent truncation scheme for parallel multipliers,” Proc. 31st Asilomar Conference Signals, Systems, and Computers, 2: 1178–1182, (1997).

[5] Chen, Y-H., Li, C.Y., Chang T.Y., "Area-Effective and Power-Efficient Fixed-Width Booth Multipliers Using Generalized Probabilistic Estimation Bias", IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 1: 277-288, (2011).

[6] Kidambi, S.S., El-Guibaly, F. Antoniou, A., “Area-efficient multipliers for digital signal processing applications,” IEEE Transactions on Circuits and Systems II Analog and Digital Signal Processing, 43(2): 90–94, (1996).

[7] Petra, N., Caro, D.D. Garofalo, V, "Truncated binary multipliers with variable correction and minimum mean square error", Transactions on Circuits and Systems I: Regular Papers, 57(6): 1312-1325, (2010).

[8] Wey, I.C., Wang, C.C, "Low-error and hardware-efficient fixed width multiplier by using the dual-group minor input correction vector to lower input correction vector compensation error", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 20(10): 1923-1928, (2012).

[9] Juang, T.B., Hsiao, S.F., "Low-error carry-free fixed-width multipliers with low-cost compensation circuits", IEEE Transactions Circuits Systems II, 52(6): 299-303, (2005).

[10] Caro, D.D., Petra, N., Strollo, A.G.M., "Fixed-width multipliers and multipliers-accumulators with min-max approximation error",Transactions on Circuits and Systems I: Regular Papers , 60(9): 2375-2388, (2013).

[11] Wey, I.C., Peng, C.C., Liao, F.Y., "Reliable low-power multiplier design using fixed width replica redundancy block", IEEE Transactions Very Large Scale Integr. (VLSI) Systems, 23(1): 78-87, (2015).

[12] Drane, T.A., Rose, T.M., Constantinides, G.A., "On the Systematic Creation of Faithfully Rounded Truncated Multipliers and Arrays," IEEE Transactions on Computers, 63(10): 2513-2525, (2014).

[13] Joshi, P.U., Deshmukh, R.B., Gudur, V., "Self-compensation scheme for truncation error in fixed width multipliers," IET Circuits, Devices & Systems, 12(1): 55-62, (2018).

[14] Venkatachalam, S., Ko, S.B., "Design of Power and Area Efficient Approximate Multipliers," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 25(5): 1782-1786, (2017).

[15] Sertbaş, A., El-Abdallah, H., Karabiber, F., “A Fast Multiplier Hardware Design for Interval Arithmetic”, Istanbul University-Journal of Electrical and Electronics Engineering, 6: 169-174, (2012).

[16] Lamberti, F., “Reducing the computation time in (short bit-width) twos complement multipliers,” ,IEEE Transactions on Computers, 60(2): 148–156, (2011).

[17] Antelo, E., Montuschi, P., Nannarelli, A., "Improved 64-bit Radix-16 Booth Multiplier Based on Partial Product Array Height Reduction," IEEE Transactions on Circuits and Systems I: Regular Papers, 64(2): 409-418, (2017).

[18] Ding, J., Li, S., "A Modular Multiplier Implemented With Truncated Multiplication," IEEE Transactions on Circuits and Systems II: Express Briefs, 65(11): 1713-1717, (2018).