Hybrid Partial Product-based High-Performance Approximate Recursive Multipliers

Haroon Waris, Chenghua Wang, Weiqiang Liu, *Senior Member, IEEE,* Jie Han, *Senior Member, IEEE,* and Fabrizio Lombardi, *Fellow, IEEE*

Abstract—Approximate recursive multipliers exhibit low-power operation because they are designed using smaller power-efficient approximate multiplier blocks. These building blocks can be configured by varying the approximation levels for a wide range of larger multiplier sizes. However, most of the building blocks proposed for recursive multipliers are either slightly inaccurate or hardware-efficient with limited accuracy. In this brief, hybrid partial product-based building blocks are proposed by considering the probability distribution of the input operands. An efficient hardware implementation of approximate 4×4 multipliers is achieved, while maintaining the required accuracy. Moreover, high-performance approximate NOR-based half adder (NxHA) and full adder (NxFA) cells are proposed for use in a 4×4 multiplier. Three different strategies (Ax8-1/2/3) are further proposed and analyzed for utilizing the 4×4 multipliers when designing larger multipliers. Ax8-2 provides the best trade-off among the designs with a moderate MRED. A reduction of 30% and 17% in the MRED is achieved compared to previous best energy-optimized and MRED-optimized designs. Among the designs with higher MREDs, Ax8-3 exhibits the smallest MRED and PDP. Moreover, it shows an improvement of 7% to 28% in delay compared to existing approximate recursive designs. As a case study, image multiplication is evaluated; a high peak signal-to-noise ratio (PSNR) with a value close to 50dB is obtained for the proposed multiplier designs.

Index Terms—Approximate recursive multipliers, high-performance, half adder, full adder, NOR gate.

1 Introduction

Recent research has considered applications, such as image/video processing, big data analysis, RMS (recognition, mining and synthesis), and communications when error bounds can be relaxed at acceptable results. Approximate computing exploits this relaxed error requirement to achieve improvements in area, power and performance [1]. Multiplication is a basic arithmetic operation used commonly in many error-tolerant applications; therefore, designs of approximate multipliers (8/16/32 bit) [2], [3] have been proposed in the last few years. In [4] three approaches based on approximate rounding, have been proposed for fixed-width multipliers. [5] has proposed a rich library of approximate 8-bit adders and 8-bit multipliers using multiobjective genetic programming. Two 16-bit approximate multipliers have been proposed in [6] using the concept of altered partial products. However, [7] has emphasized the need of cross-layer approximate computing (e.g., at different levels, such as architecture, high level synthesis, and system). To introduce approximate computing at different levels of implementation, a systematic approach is presented; such approach relies on arithmetic modules as basic building blocks.

While several approximate adder blocks are available, very few approximate multiplier blocks have been reported for assembling larger designs. As cross-layer approximate computing requires configurability and flexibility, an extensive library of approximate multiplier blocks is needed. Therefore, in this brief, a probabilistic analysis of a conventional partial product (PP) array is performed; then, a hybrid partial product array based 4×4 multiplier is proposed. Note, [6] has also used the statistical analysis to propose 8bit and larger multipliers; however, the required performance gains for small multipliers cannot be achieved. Compared to [6], this approach leads to a reduction in the partial product size, thus, reducing the hardware complexity of a 4×4 building block. Moreover, this statistical analysis is used to design NOR based high-performance half-adder (NxHA) and full-adder cells (NxFA). An improvement of 41% in the delay is achieved compared to the approximate full adder proposed in [6].

The rest of the paper is organized as follows. A literature survey of recursive multipliers is presented in Section II. Section III presents the probabilistic analysis of a partial product array. Section IV discusses the proposed 4×4 multiplier including NOR-based HA and FA cells. The design of larger multipliers is considered in Section V. Hardware and error characteristics of the proposed and state-of-the-art approximate multipliers are presented in Section VI. The proposed designs are evaluated in Section VII for image processing. Section VIII concludes the paper.

2 Recursive Multipliers

An M-bit recursive multiplier [8], [9], [10], [11], [12] cascades N-bit multiplier blocks at the relevant bit positions,

- *Haroon Waris, Chenghua Wang and Weiqiang Liu are with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, 211106.*
  E-mail: (haroonwaris, chwang, liuweiqiang)@nuaa.edu.cn
- *Jie Han is with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada*
  E-mail: jhan8@ualberta.ca
- *Fabrizio Lombardi is with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA USA, 02115.*
  Email: lombardi@ece.neu.edu

Manuscript received XX XX, 2020; revised XX XX, 2020.

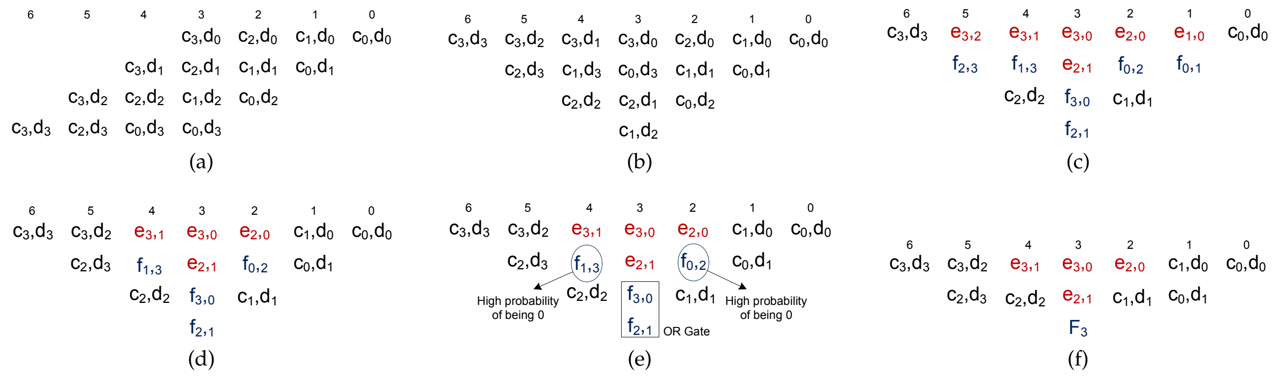


Fig. 1: 4×4 (a) conventional PP array (b) equivalent PP array (c) propagate-and-generate (PG) array (d) probabilistic analysis based PP array (e) approximation of generate elements (f) proposed hybrid PP array

where $M = 2^k N, k = 0, 1, 2, ..., log_2(M/N)$. Kulkurani *et al.* [8] have proposed an approximate 2×2 multiplier in which the output 1001 is approximated to 111. The use of 2×2 blocks to design M-bit multipliers reduces the addition stages. In [9], an inaccurate 4:2 counter has been proposed and used to design an approximate 4-bit multiplier. This multiplier uses only one approximate 4:2 counter in the partial product reduction stage; therefore, it exhibits a low error rate (ER). The 2×2 building blocks proposed by Rehman *et al.* [10] have a reduced maximum error magnitude compared to [8]. Ansari *et al.* [11] have proposed a 4×4 multiplier based on three approximate 4:2 compressors; subsequently larger approximate multipliers are designed. However, these 4:2 compressors have a large ER; for example, in the truth table for the stage-3 compressor, six sum values have been approximated out of sixteen, so an ER of 37.5% is introduced. Recently, [12] has proposed an approximate 4×4 multiplier using inexact half adder and full adder cells; 8×8 multipliers are then designed. Although the inexact adders reduce hardware complexity, they do not improve on the critical path delay compared to the other approximate adders.

3 Probabilistic Analysis of Partial Product Array

The probabilistic analysis performed in this brief assumes that every bit in the binary representation of the multiplier inputs A and B is 1 or 0 with equal probability. This assumption holds true only if the inputs follow a uniform distribution, as considered in this analysis. For a 4-bit multiplier, consider two unsigned input operands $c = \sum_{k=0}^{3} c_k.2^k$ and $d = \sum_{l=0}^{3} d_l.2^l$. A conventional PP array (Fig. 1a) is obtained by the AND operation of the bits in c and d. Fig. 1b is an equivalent representation of Fig. 1a by shifting up the elements in the second half (starting from column 4). Moreover, the elements in columns (2, 3 and 4) are adjusted so that propagate-and-generate functions can be applied to the PP array. The PP element $c_k.d_l$ is obtained by multiplying two binary inputs; therefore, the possible input combinations are {00,01,10,11}. The output is "1" only when the input combination is {11}. Therefore, the probability of any PP bit being 1 is 1/4 for statistically independent and uniformly distributed inputs. An approximation in the PP bit deteriorates the accuracy; however, the level of deterioration depends upon two factors: 1) The PP bit location and 2) The probability of occurrence of a particular PP bit. Therefore, propagate-and-generate (PG) is utilized to reduce the probability of occurrence of a particular PP bit. Specifically, the partial products $c_k.d_l$ and $c_l.d_k$ are used to establish a propagate (e) and generate (f) array (Fig.1c) from a conventional PP array such that the probability of occurrence is reduced for certain PP elements. Equations (1) and (2) are used to design the PG array as shown in Fig. 1c.

$$e_{k,l} = c_k.d_l + c_l.d_k \tag{1}$$

$$f_{k,l} = (c_k.d_l).(c_l.d_k) \tag{2}$$

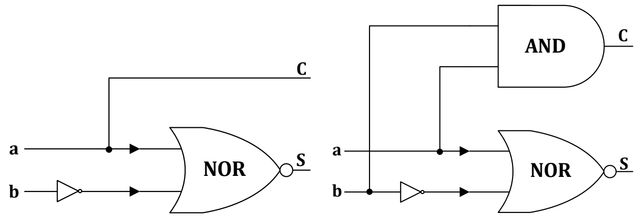
The probability of being one for any element in the new PG array is not the same as calculated earlier. The PG element $f_{k,l}$ has a probability of 1/16 of being 1, i.e., 6.25%, so substantially less than 25% for the original $c_k.d_l$. The probability of $e_{k,l}$ being one is 7/16, higher than $f_{k,l}$. Next, a probabilistic analysis is performed for both the conventional PP and the transformed PG arrays. In the PP array, the sum is performed column wise; thus, each column is analyzed individually for the probability of being one. Columns 1 and 5 in both the PP and PG array have a higher probability of being 0's (Table 1) compared to the probability of being 1. Therefore, for these two columns, the transformed PG representation has no obvious advantage. The probability of being all 1's (Table 1) decreases by 60% in the PG representation for columns 2 and 4; moreover, the probability of occurrence of all 0's and two 1's has also decreased. Similarly, the probability of all 1's and three 1's (Table 1) decreases by 94% and 47%, respectively, in the PG representation of column 3. Therefore, columns (2, 3 and 4) are represented using the PG array whereas in columns (1 and 5) a conventional PP array is utilized, because it saves hardware for the propagate-and-generate logic (Fig. 1d).

The probability of a generate element being one is low (1/16), two elements being 1 in the same column even decreases. The probabilistic analysis-based PP array (Fig. 1d) have generate elements in columns (2, 3 and 4). Columns 2 and 4 both have one generate element $\{f_{0,2}\}$ and $\{f_{1,3}\}$, respectively, whereas column 3 has two generate $\{f_{3,0}, f_{2,1}\}$

TABLE 1: Probabilistic Analysis of Conventional and Transformed Partial Product Arrays

Column No	Conventional PP Array					Transformed PG Array				
	All 1's	All 0's	single 1	two 1's	three 1's	All 1's	All 0's	single 1	two 1's	three 1's
(1 & 5)[a]	0.062	0.562	0.375	-	-	0.027	0.527	0.445	-	-
(2 & 4)[a]	0.015	0.421	0.421	0.140	-	**0.006**	0.395	0.465	0.131	-
3	0.012	0.316	0.421	0.210	0.046	**0.0007**	0.278	0.469	0.227	**0.024**

[a](1 & 5) , (2 & 4) have similar number of elements in a column.



(a) NxHA1 cell (b) NxHA2 cell

Fig. 2: Logic diagram of approximate HA cells.

TABLE 2: Truth Table of Exact and Approximate HA Cells

Inputs		Exact-HA		NxHA1		ED	NxHA2		ED
a	b	C	S	C	S		C	S	
0	0	0	0	0✓	0✓	0	0✓	0✓	0
0	1	0	1	0✓	1✓	0	0✓	1✓	0
1	0	0	1	1×	0×	1	0✓	0×	1
1	1	1	0	1✓	0✓	0	1✓	0✓	0

elements. Then, the probability of generate element being all 1's in columns (2 and 4) is 1/16, while the probability of the two generate elements being 1 in column 3 is significantly low, i.e., 1/256. Therefore, for column 3 one bit can be reduced by ORing these two bits and for columns (2 and 4), the generate element can be approximated to 0s with no significant impact on the accuracy (Fig. 1e). The proposed hybrid PP array with a reduced number of partial products is shown in Fig. 1f. Furthermore, the design of proposed approximate half and full adder cells is also based on the performed probabilistic analysis. The following key features are considered.

(a) Columns 1 and 5 are based on the conventional partial product elements (Fig. 1f) and they have a high probability of all 0's (Table 1), i.e., the input combination (00) has the highest probability of occurrence; therefore, an approximation should never be made for these inputs.
(b) Column 3 is based on the propagate-and-generate (PG) elements (Fig. 1f), they have a high probability of a single 1 (Table 1), i.e., input combinations {001,010,100} have a higher occurrence probability. Therefore, minimal approximation is to be introduced for these inputs.

4 4×4 Multiplier Design

In the proposed 4×4 multiplier, PPs are generated using AND gates. The design of the approximate HA and FA cells to be used in the partial product reduction step is pursued using the analysis presented in Section 3. In the proposed approximate HA, the input combination {00} is not approximated. Similarly, in the approximate FA, only the Sum (S) for one input combination {001} is approximated, so that a very small error is introduced. The normalized gate delay is usually used for CMOS gate level analysis; the NOR gate has the smallest delay (0.5) compared to other CMOS logic gates. Therefore, to achieve high-performance, NOR gates are used to design the approximate half adder (NxHA) and full adder (NxFA).

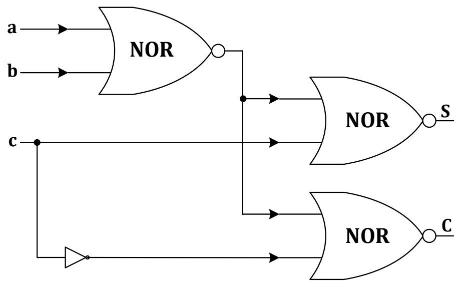


Fig. 3: Logic diagram of an approximate FA.

4.1 NOR-based Approximate Half-Adder (NxHA)

Two approximate HA designs of variable accuracy, denoted as NxHA1 and NxHA2, are proposed. Generally, XOR gates have a higher delay and area; therefore, in NxHA1, the sum is obtained by using NOR gates, instead of XOR gates. For both designs, the error difference between the exact and approximate outputs is kept as one and shown in Table 2. NxHA1 uses a larger approximation as the carry is also approximated. Fig. 2 shows the gate level diagrams. Equations (3) and (4) below describe the output of the NxHA1 and NxHA2 cells, respectively.

$$\left.\begin{aligned} Sum_{NxHA1} &= \overline{a+\bar{b}} \\ Carry_{NxHA2} &= a \end{aligned}\right\} \quad (3)$$

$$\left.\begin{aligned} Sum_{NxHA2} &= \overline{a+\bar{b}} \\ Carry_{NxHA1} &= ab \end{aligned}\right\} \quad (4)$$

4.2 NOR-based Approximate Full-Adder (NxFA)

The NxFA cell uses three NOR gates and one NOT gate as shown in Fig. 3. An error in the carry has a large impact, because it has twice the binary weight of the sum, so the approximation for carry is only considered for one input. Moreover, the difference between the exact and approximate outputs is kept to one as shown in Table 3. [7] has proposed a library of approximate 1-bit full-adders (five designs); truth tables are used to describe the gate-level logic circuits. In this paper, the same implementation approach is adopted

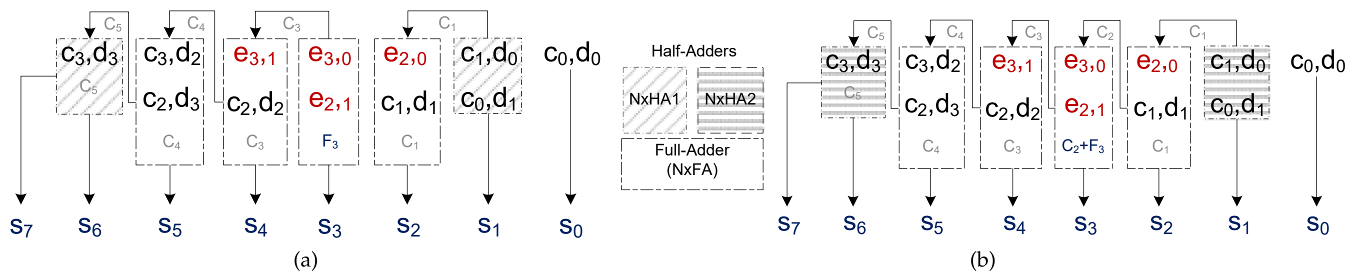


Fig. 4: Hybrid partial product array-based 4×4 multipliers designed using approximate half-adder and full-adder cells (a) MxA (NxHA1/NxFA) (b) LxA (NxHA2/NxFA)

TABLE 3: NxFA Truth Table

Inputs			Exact-FA		NxFA		ED
a	b	c	C	S	C	S	
0	0	0	0	0	0✓	0✓	0
0	0	1	0	1	0✓	0×	1
0	1	0	0	1	0✓	1✓	0
0	1	1	1	0	1✓	0✓	0
1	0	0	0	1	0✓	1✓	0
1	0	1	1	0	1✓	0✓	0
1	1	0	1	0	0×	1×	1
1	1	1	1	1	1✓	0×	1

TABLE 4: Critical Path Delay and Hardware Complexity

FA	Logic Gates				Delay
	XOR	AND/OR	NOR	NOT	
Exact	2	3	0	0	2.4
NxFA	0	0	3	1	**1.0**
AFA [6]	1	2	0	0	1.7
AA_1 [7]	0	6	0	2	2.1
AA_2 [7]	0	4	0	1	2.3
AA_3 [7]	0	2	0	4	1.8
IFA [12]	1	4	0	0	2.4

for comparative evaluation of the proposed NxFA. Three designs (AA_1, AA_2 and AA_3) from [7] are considered for comparison. AA_2 and AA_4 have very similar area, but AA2 offers a reduced power consumption and better accuracy; therefore, it is selected. AA5 offers the best area and power efficiency, but it has the lowest accuracy in terms of error cases; thus, it is not considered. The critical path delay of NxFA is reduced to 1.0 unit delay compared to 2.4 (for the exact design) and 1.7 (for the previous best high-performance based approximate FA design [6]) unit delays, respectively. Therefore, improvement of 58% and 41% are achieved (Table 4). The Boolean functions of NxFA are given in (5).

$$\left.\begin{aligned} Sum_{NxFA} &= \overline{\overline{a+b+c}} \\ Carry_{NxFA} &= \overline{a+b+\overline{c}} \end{aligned}\right\} \quad (5)$$

4.3 Basic-Building Block Variants

Two variants (i.e., the so-called more-approximated (MxA) and less-approximated (LxA) 4×4 elementary multipliers) are designed as shown in Fig. 4. They differ in the way sum (S_3) is generated and in the use of an approximate HA cell. NxHA1, NxHA2 and NxFA cells are used for partial product reduction. The utilization of two proposed HA cells in the 4×4 multipliers is systematically pursued. In MxA (Fig. 4a), the circuit latency is reduced by breaking the carry propagation path. Moreover, as it aims to achieve low-power, this scheme uses the less-accurate (NxHA1) cell. LxA achieves better accuracy than MxA by using the more-accurate (NxHA2) cell. In LxA, the computation of S_3 (Fig. 4b) requires the addition of three $pp_{(i,j)}$ terms and the carry (c_2) from previous stage. This is achieved by merging two (F_3+c_2) of these four signals and reducing them to three, as in (Fig. 4b). Table 5 shows the hardware savings of MxA and LxA compared to an exact 4×4 multiplier. MxA has less delay than LxA because the carry propagation path is truncated; an improvement of 22% in the critical path delay is achieved. Although LxA is less-approximated, the use of NOR-based HA and FA cells results in a reduced PDP of 35% compared to its exact counterpart.

TABLE 5: Hardware Resource Consumption of MxA/LxA

Multiplier	Area (um^2)	Delay (ns)	Power (uW)	PDP (fJ)
Exact	105.41	0.32	41.25	13.20
MxA	40.73	0.25	19.35	4.83
LxA	61.95	0.28	30.87	8.64

TABLE 6: MxA/LxA based 8×8 Multipliers

Architecture	**PP_4**	**PP_3**	**PP_2**	**PP_1**
Ax8-1	accurate	accurate	accurate	MxA
Ax8-2	accurate	accurate	LxA	MxA
Ax8-3	accurate	LxA	LxA	MxA

5 Design of Larger Multipliers

Recursive partitioning is used to achieve faster implementations for a larger multiplier. In the same cycle, multiple 4×4 basic modules (proposed previously in Section 4.3) are used to design 8×8 multipliers. Consider two 8-bit operands x and y, as a combination of two 4-bit operands (x_h, x_l) and (y_h, y_l), respectively. Note, x_h and y_h corresponds to the 4 MSBs whereas x_l and y_l indicate the 4 LSBs of the input operands. This partition of large input operands into 4-bit operands allows the use of 4×4 modules for multiplication. Four 4×4 multipliers are used in parallel to generate the partial products $\{pp_4, pp_3, pp_2, pp_1\}$. In [11], it has been shown that pp_4 has the greatest impact on the multiplier

TABLE 7: Comparative Performance Analysis of 8×8 Approximate Recursive Multipliers

Design	Area (um^2)	Delay (ns)	Power (uW)	Energy (uW.ns)	MRED (%)	NMED (10^{-3})	ER (%)
Ax8-1	419.23	0.84	181.65	152.58	0.10	0.46	30.47
Ax8-2	323.97	0.80	156.70	125.36	1.26	1.64	48.35
Ax8-3	301.65	0.77	143.54	110.64	2.83	6.12	69.73
mul8u_125K [5]	674.90	1.42	384.00	545.28	0.02	.015	17.19
mul8u_ZFB [5]	590.40	1.13	304.00	343.52	0.80	1.27	69.26
Multiplier1 [6]	460.91	0.87	233.54	203.17	3.20	8.50	98.43
Multiplier2 [6]	583.25	0.98	458.71	449.53	0.01	.003	97.12
UDM [8]	391.28	1.00	176.35	176.35	3.28	14.2	47.09
IWM [9]	426.50	0.93	183.70	170.84	0.06	0.29	5.45
ApproxMul$_2$ [10]	412.89	1.06	187.46	198.70	1.51	6.90	81.44
M8-1 [11]	313.75	0.82	151.20	123.98	6.49	19.0	73.17
M8-3 [11]	351.15	0.83	169.40	140.60	1.70	2.10	66.36
M8-5 [11]	458.30	0.91	204.35	185.95	0.13	0.06	36.22
LOAM [12]	279.81	0.93	112.78	104.88	1.81	2.00	75.91

output, thus, the least errors should be introduced for this computation. Therefore, in this brief, the accuracy-energy trade-off for large multipliers is assessed by considering the following three cases (Table 6).

- In the first case (Ax8-1), the MxA block is used for the generation of the least significant pp_1 while all other PPs are accurately computed. The worst-case error (WCE) for this design is given by:

$$WCE_{Ax8-1} = WCE_{MxA} \tag{6}$$

- In the second case (Ax8-2), the LxA block is used for the generation of pp_2 while the MxA block is used for pp_1. The WCE for this design is given by:

$$WCE_{Ax8-2} = 2^n\, WCE_{LxA} + WCE_{MxA} \tag{7}$$

- In the third case (Ax8-3), both pp_2 and pp_3 are generated using LxA blocks while the MxA is used for pp_1. The WCE for this design is given by:

$$WCE_{Ax8-3} = 2 \times 2^n\, WCE_{LxA} + WCE_{MxA} \tag{8}$$

The four partial products generated by each proposed multiplier are added using a Wallace tree at the relevant bit-positions. The objectives of Ax8-1 and Ax8-2 are to achieve a small error distance at a reduced power consumption. The improvement in delay for these designs is dependent on the exact PPs that are on the critical path. However, the proposed Ax8-3 has a large error distance because three PPs are approximated; therefore, reductions in delay and area are expected. While the first two designs have a lower error distance, the third design is expected to have a relatively higher error distance with better performance in terms of hardware complexity and delay.

6 Performance Evaluation

The proposed and existing approximate recursive multipliers are synthesized using Synopsys Design Compiler (DC) at 45nm. The Synopsys VCS is used for functional verification of the generated netlist. Power is found using the Primetime tool; SAIF (Switching Activity Interchange Format) and VCD (Value Change Dump) files are used as an input. The equivalent behavioral models are developed for error analysis. The error rate (ER), the normalized mean error distance (NMED) and the mean relative error distance (MRED) are calculated over the entire input space of 8×8 multipliers, i.e., 65,536.

Compared multipliers include EvoApprox8b designs [5], altered partial product based multipliers [6] and state-of-the-art approximate recursive multipliers [8], [9], [10], [11], [12]. Table 7 shows the delay, area, power, power-delay product (PDP) and error characteristics of the considered multipliers. The implementations of Ax8-1/2/3 designs include the propagate-and-generate logic used to achieve a hybrid partial product array. Ax8-3 has the least delay among all approximate designs; an improvement of 7% to 28% is achieved. This occurs because MxA and LxA in large multipliers are constructed from high-performance NOR gates. Moreover, the Ax8-1/2/3 designs have a reduced MRED of 23% to 56%, compared to the most recent state-of-the-art approximate recursive M8-5/3/1 [11] designs. All considered approximate multipliers are further compared with respect to the MRED and PDP as shown in Fig. 5. Based on the MRED metric the multipliers, are categorized in three groups. Among the designs with lower MREDs, IWM has the least MRED at the cost of a higher PDP, as only one approximate 4:2 counter is used in the design of 4×4 multiplier. However, the Ax8-1 has a comparable MRED with IWM and achieves a PDP saving of 18%. Ax8-2 provides the best trade-off among the designs with moderate MREDs. A reduction of 30% and 17% in the MRED is achieved compared to previous best energy-optimized (LOAM) and previous best MRED-optimized (ApproxMul$_2$) designs. Among the higher MRED-based designs, Ax8-3 has the smallest MRED as well as the PDP. Ax8-3 uses a combination of MxA and LxA; therefore, an improvement of 56% and 11% in the MRED and PDP is achieved compared to previous best PDP-optimized (Mul8-1) design. The considered EvoApprox8b (mul8u_125K and mul8u_ZFB) are from the Pareto-optimal subset of MRE vs power. Although they exhibit low MREDs, they also have a large delay, thus making them not favorable for applications that require a lower MRED and PDP. For an n-bit multiplier, multiplier 1 exhibits a high MRED with a reduced PDP, whereas Multiplier 2 has a high power consumption because it approximates only $n-1$ least significant columns. ApproxMul$_2$ has a reduced maximum error compared to a previous 2×2 based design (UDM); therefore, it lies in the set of multipliers with moderate MREDs. M8-

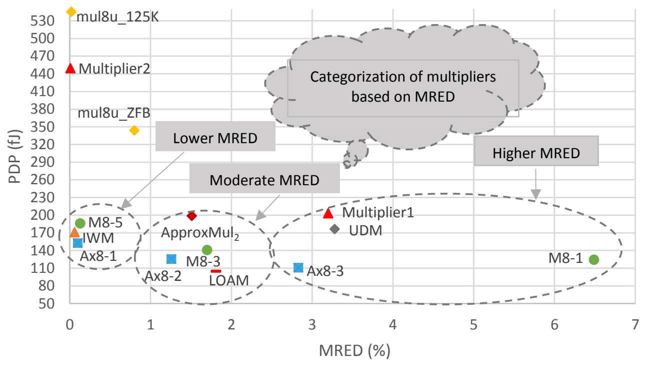


Fig. 5: PDP vs MRED analysis of approximate recursive multipliers.

TABLE 8: Performance metrics computed form image processing results

Multiplier	MED (10^{-2})	MSE (10^{-3})	PSNR (dB)	SSIM
Ax8-1	0.045	0.023	53.78	0.97
Ax8-2	0.069	0.042	49.31	0.91
Ax8-3	1.05	0.465	39.46	0.85

5/3/1 have a relatively moderate delay compared to other approximate designs; however, the use of three approximate 4:2 compressors in a single 4×4 building block results in a higher PDP.

7 Case Study: Image Processing

An application of the proposed approximate multipliers is illustrated using image multiplication. Pixels by pixel multiplication of two images result in a new output image. Fig. 6 shows the results obtained by multiplying two images using approximate (Ax8-1/2/3) multipliers. The mean error distance (MED), mean squared error (MSE), structural similarity index (SSIM) and PSNR are used as quality metrics to quantify the output image. The result shows that Ax8-1 and Ax8-2 result in a high PSNR (close to 50dB). Ax8-3 has the highest MED and MSE metrics (Table 8) as the three PP blocks are approximated, thus consistent with the design analysis presented in Section 5.

8 Conclusion

This paper has presented a hybrid partial product array based 4×4 (MxA and LxA) multipliers using a probabilistic analysis. High-performance NOR-based half-adder (NxHA) and full-adder (NxFA) cells have been proposed for use in the 4×4 multiplier. The proposed NxFA cell achieves an improvement of 41% in critical path delay. Three 8×8 multipliers (Ax8-1/2/3) are then designed using different configurations of MxA and LxA. Ax8-2 shows an improvement of 30% and 17% in the MRED compared to previous best energy-optimized (LOAM) and previous best MRED-optimized (ApproxMul$_2$) designs. Ax8-3 is the fastest design; an improvement of 7% to 28% in the delay is achieved compared to existing approximate recursive designs. Moreover, it also exhibits the smallest MRED and PDP among the multipliers with higher MREDs. The

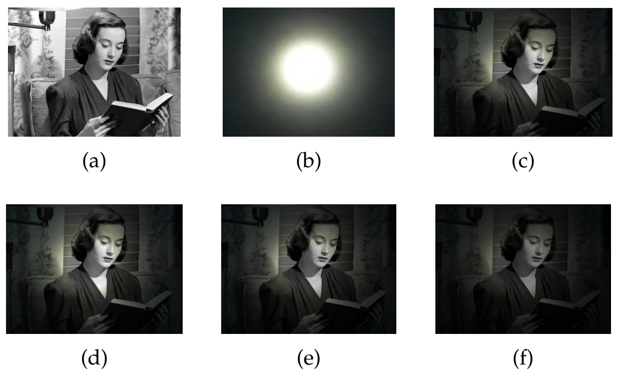


Fig. 6: Image multiplication (a) image1, (b) image2, (c) accurate multiplier, (d) Ax8-1, (e) Ax8-2, (f) Ax8-3

proposed approximate multipliers have been applied to image multiplication and two of them achieved a PSNR close to 50 dB, thus suitable for error-resilient applications with no significant loss of quality. The synthesizable Verilog files are provided as open-source libraries at https://sourceforge.net/projects/approxarithmeticlib/.

References

[1] W. Liu, F. Lombardi and M. Shulte, “A Retrospective and Prospective View of Approximate Computing [Point of View],” *Proceedings of the IEEE*, vol. 108, no. 3, pp. 394–399, 2020.

[2] S. Vahdat, M. Kamal, A. Afzali-Kusha and M. Pedram, “TOSAM: An Energy-Efficient Truncation-and Rounding-Based Scalable Approximate Multiplier,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 5, pp. 1161-1173, 2019.

[3] H. Waris, C. Wang and W. Liu, “Hybrid Low Radix Encoding based Approximate Booth Multipliers,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, doi: 10.1109/TCSII.2020.2975094.

[4] E. E. Swartzlander, “Truncated multiplication with approximate rounding,” *in Proc. Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers (Cat. No.CH37020)*, pp. 1480-1483, 1999.

[5] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, “EvoApprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods,” *in Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 258-261, 2017.

[6] S. Venkatachalam and S.-B. Ko, “Design of Power and Area Efficient Approximate Multipliers,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 5, pp. 1435-1441, 2017.

[7] M. Shafique, R. Hafiz, S. Rehman, W. El-Harouni, J. Henkel, “Cross-Layer Approximate Computing: From Logic to Architectures,” *in Proc. Design Automation Conference (DAC)*, pp. 1-6, 2016.

[8] P. Kulkarni, P. Gupta, and M. Ercegovac, “Trading accuracy for power with an underdesigned multiplier architecture,” *in Proc. 24th IEEE International Conference on VLSI Design (VLSID)*, pp. 346–351, 2011.

[9] C. Lin and I. Lin, “High accuracy approximate multiplier with error correction,” *in Proc. International Conference on Computer Design (ICCD)*, pp. 33–38, 2013.

[10] S. Rehman, W. El-Harouni, M. Shafique, A. Kumar, and J. Henkel, “Architectural-space exploration of approximate multipliers,” *in Proc. International Conference on Computer-Aided Design (ICCAD)*, pp. 1-8, 2016.

[11] M. S. Ansari, H. Jiang, B. F. Cockburn, and J. Han, “Low-power approximate multipliers using encoded partial products and approximate compressors,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 3, pp. 404-416, 2018.

[12] Y. Guo, H. Sun and S. Kimura, “Design of Power and Area Efficient Lower-Part-OR Approximate Multiplier,” *in Proc. IEEE Region 10 Conference (TENCON)*, pp. 2110-2115, 2018.