A Design Approach for Compressor Based Approximate Multipliers

Naman Maheshwari
Electrical & Electronics Engineering,
Birla Institute of Technology & Science,
Pilani, Rajasthan - 333031, India
Email: naman.mah1993@gmail.com

Zhixi Yang and Jie Han
Department of Computer and Electrical
Engineering, University of Alberta,
Edmonton, Alberta T6G 2V4, Canada
Email: {zhixi, jhan8}@ualberta.ca

Fabrizio Lombardi
Department of Electrical and Computer
Engineering, Northeastern University,
Boston, Massachusetts, USA
Email: lombardi@ece.neu.edu

Abstract— **Approximate computing is best suited for error resilient applications, such as signal processing and multimedia. Approximate computing reduces accuracy, but it still provides meaningful and faster results with usually lower power consumption; this is particularly attractive for arithmetic circuits. In this paper, a new design approach is proposed to exploit the partitions of partial products using recursive multiplication for compressor-based approximate multipliers. Four multiplier designs are proposed using 4:2 approximate compressors. Extensive simulation results show that the proposed designs achieve significant accuracy improvement together with power and delay reductions compared to previous approximate designs. An image processing application is also presented to show the efficiency of the proposed designs.**

Keywords—Approximate computing, compressor, multiplier

I. INTRODUCTION

Many scientific and engineering problems are computed using accurate, precise and deterministic algorithms. However, in many applications involving signal/image processing and multimedia, exact and accurate computations are not always necessary, because these applications are error tolerant and produce results that are good enough for human perception [1]. In these error resilient applications, a reduction in circuit complexity, and thus, area, power and delay is very important for the operation of a circuit. Hence, approximate computing can be used in error tolerant applications by reducing accuracy, but still providing meaningful results faster and/or with lower power consumption [2].

Addition and multiplication are often used in these applications. For addition, full adders have been analyzed in detail and a number of approximate designs have been proposed [1]. In [3], several new metrics are proposed and a comparison is made among some of the adder designs. The error distance (ED) is defined as the arithmetic distance between an erroneous and the correct outputs for a given input. The mean error distance (MED) and normalized error distance (NED) are then proposed.

Recently, approximate multipliers have also gained significance because of their importance in arithmetic operations [4–10]; several approximate 4:2 compressors have been proposed in the reduction of the partial products of a Dadda tree. In this paper, the approximate compressors of [10] are utilized to design 8x8 bit multipliers by a novel partition of the partial products. The newly-designed approximate multipliers are more accurate than the ones proposed in [10] and require approximately the same power and delay; it is shown that the improvement in accuracy is significant, albeit at a slightly increase in area.

This paper is organized as follows. Section II reviews approximate multipliers and the compressors used in the proposed designs. Section III presents the proposed multipliers. Section IV provides the simulation results for the multipliers and compares the proposed design with [10]. Section V presents an image processing application using the approximate multipliers and Section VI gives the conclusion.

II. PRELIMINARIES AND REVIEW

A. Approximate multipliers

An error tolerant multiplier (ETM) uses accuracy as a design parameter and divides the operands into two parts – multiplication and non–multiplication, depending on the required accuracy [4]. It performs the multiplication only in the first part, thus saving power and delay at the cost of accuracy. A novel 2x2 bit underdesigned multiplier (UDM) is proposed and used to build a larger multiplier [5]. [6] presents a 6x6 bit broken array multiplier (BAM), that is faster than an accurate array multiplier. [7] proposes a 4x4 imprecise counter-based multiplier (ICM) that uses a 4:2 inaccurate counter to reduce the partial product stages of a Wallace tree multiplier. It leads to a power efficient design, which can then be used to implement multipliers of large sizes. Four different modes of an approximate Wallace tree multiplier (AWTM) are presented in [8]. This design uses a carry-in prediction method, resulting in hardware reduction and thus, less power, area and delay compared to the accurate Wallace tree multiplier. Also, AWTM uses the simple recursive multiplication technique that has also been used in this paper and explained in Section II.C. [9] proposes a fast and power-efficient multiplier based on an approximate adder that can process data in parallel by cutting the carry propagation chain. Two new approximate 4:2 compressors and four approximate multipliers are proposed in [10]. Similar compressors have been used in the partial product reduction stage in the multipliers proposed in this paper. Most of the approximate multipliers aim for a trade-off in accuracy, power, delay and area.

B. *Accuracy metrics*

- Error distance (ED) [3]: It is defined as the modulus of the difference between the accurate and approximate outputs. Suppose a is the accurate sequence of bits and b is the approximate sequence of bits, the error distance is given by

$$ED = |a - b| = |\sum_i a[i] * 2^i - \sum_j b[j] * 2^j| \quad (1)$$

- Normalized error distance (NED) [3]: It is the mean error distance divided by the maximum possible error and hence, this is the most comprehensive metric for comparing accuracy among different designs.

- Pass Rate: It is the number of correct outputs divided by the total number of outputs.

$$Pass\ rate = \frac{Number\ of\ correct\ outputs}{Number\ of\ total\ outputs} \quad (2)$$

- Error Rate: It is defined as the probability of incorrect outputs, when different inputs are provided.

$$Error\ rate = \frac{Number\ of\ incorrect\ outputs}{Number\ of\ total\ outputs} \quad (3)$$

C. *Recursive multiplication*

The technique used in this paper for designing 8×8 multipliers using 4×4 multipliers is known as recursive multiplication. Suppose there are 2 numbers A and B of $2a$ bits each. It is possible to break the two numbers into 2 halves, i.e. most significant a bits and least significant a bits. So A_H denotes the upper a bits of A, A_L denotes the lower a bits of A and similarly, B_H and B_L denote the upper and the lower a bits of B respectively. Then, instead of performing a $2a \times 2a$ multiplication, four $a \times a$ multiplications are performed ($A_H B_H$, $A_H B_L$, $A_L B_H$ & $A_L B_L$) and added to get the final output, as shown in Fig. 1.

D. *Accurate compressor*

Compressors are used to reduce the number of partial product stages. The basic structure of an accurate 4:2 compressor chain utilized in the partial product reduction is shown in Fig. 2. A 4:2 compressor produces a sum for the same order of the next stage, and a carry for one order higher in the next stage. Also, a carry out (C_{out}) is generated and becomes the carry in (C_{in}) of the next higher-order compressor. A 4:2 accurate compressor is implemented using two full adder circuits as shown in Fig. 3. There are many other designs for implementing the accurate compressor. [11] describes a design for a 4:2 accurate compressor using three XOR-XNOR gates, one XOR gate and two 2:1 multiplexers. The logic equations for the three outputs of the compressor are as follows:

$$Sum = A \oplus B \oplus C \oplus D \oplus C_{in} \quad (4)$$
$$C_{out} = (A \oplus B) C_{in} + AB \quad (5)$$
$$Carry = (A \oplus B \oplus C \oplus D) C_{in} + (A \oplus B \oplus C) D \quad (6)$$

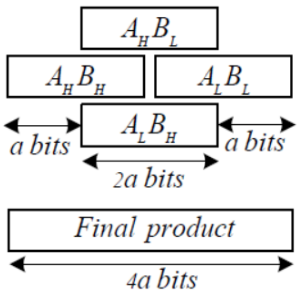


Fig. 1. Format of the inputs for recursive multiplication

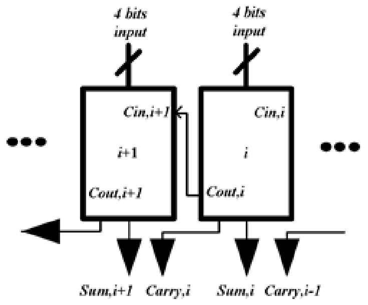


Fig. 2. Adjacent compressors within a chain in the partial product reduction stage

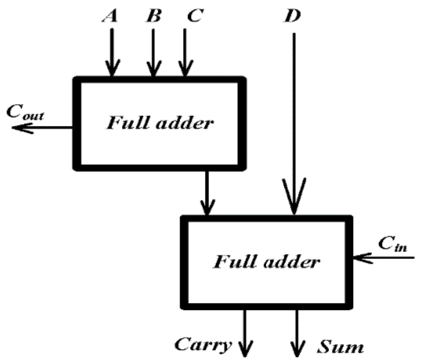


Fig. 3. An accurate compressor by using two full adders

E. *Approximate compressors utilized*

The two designs of inaccurate compressors as proposed in [10] have been used in this paper when designing multipliers. Both designs are based on the modification of the truth table of the accurate compressor to reduce the hardware. In design 1, the carry signal is directly connected to the C_{in} signal and the columns of the sum and C_{out} signals are modified to reduce the hardware, and hence reducing the delay. The logic functions for design 1 are given as:

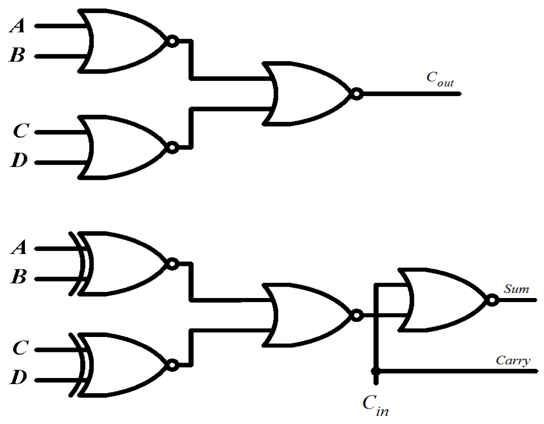
$$Sum = ((\overline{(A \oplus B)} + \overline{(C \oplus D)}))\overline{C_{in}} \quad (7)$$
$$C_{out} = \overline{((\overline{(A + B)} + \overline{(C + D)}))} \quad (8)$$
$$Carry = C_{in} \quad (9)$$

In design 2, C_{out} is completely removed from the circuit and hence, there is no need of C_{in} as well. Hence, this design further simplifies the circuit and gives better results in terms of accuracy. The logic functions for design 2 are given as:

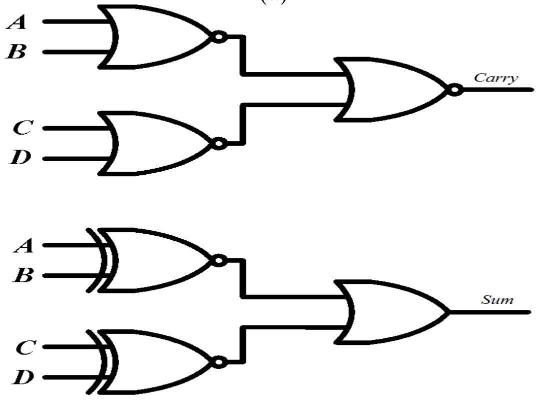
$$Sum = \overline{(A \oplus B)} + \overline{(C \oplus D)} \tag{10}$$

$$Carry = \overline{(\overline{A + B}) + (\overline{C + D})} \tag{11}$$

The circuit diagrams for the two inaccurate designs are shown in the Fig. 4.



(a)



(b)

Fig. 4. Two approximate compressors [10]

III. PROPOSED DESIGNS

In this section, the proposed 8×8 multiplier designs are presented. Since the technique of recursive multiplication is used, 4×4 multipliers are required for the implementation of the 8×8 product. Hence, the 4×4 multiplier designs are presented too. The method of recursive multiplication is shown using a partial product tree to illustrate its difference from a conventional design.

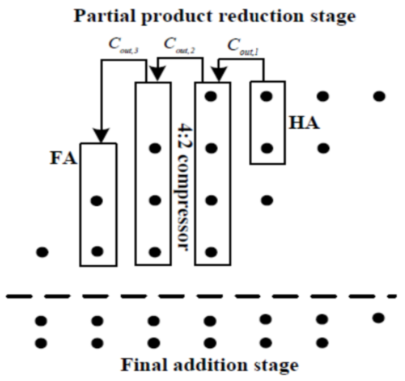
A. 4X4 bit designs

Three 4×4 bit multipliers have been implemented and further used in the 8×8 bit multiplication. All three designs are implemented using the Dadda tree technique by making use of different 4:2 compressors in the reduction stage. Using the compressors, the 4×4 bit product requires one reduction stage, making the product calculation faster.

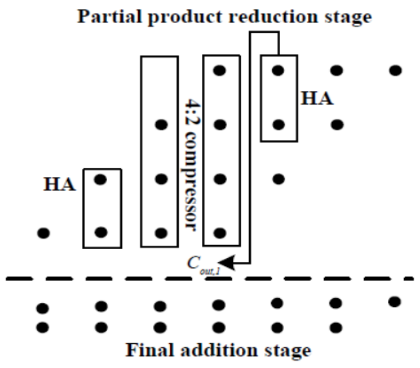
For the first design, Mul44_1, the design 1 compressor shown in Fig. 4 (a) is used in the partial product reduction stage. The Dadda tree implementation of Mul44_1 is shown in Fig. 5 (a); only two compressors are required in the partial product reduction stage.

Similarly, for the second design, Mul44_2, design 2 compressor shown in Fig. 4 (b) is used in the reduction stage and as design 2 does not have a carry to the next stage, the design is a bit different from Mul44_1. The Dadda tree implementation of Mul44_2 is shown in Fig. 5 (b). Only one compressor is required in the reduction stage, which significantly simplifies the design.

For the accurate 4x4 multiplier, Mul44_acc, the Dadda tree implementation is the same as Mul44_1, because the design 1 compressor and the accurate compressor use the same types of circuits. Hence, only accurate compressors need to be used in place of the design 1 compressors. In this multiplier, two accurate compressors are required in the reduction stage.



(a)



(b)

Fig. 5. Use of compressors for partial product reduction [10]

B. 8X8 bit designs

4×4 multipliers are used in the implementation of 8×8 multipliers. The partial product tree of the 8×8 multiplication are broken down to 4 products of 4×4 modules using the technique of recursive multiplication, as shown in Fig. 6. The advantage of breaking the products is to obtain smaller multiplication blocks that are performed in parallel and thus faster. Then, they merely need to be added, according to Fig.1 to obtain the final product.

There are three different choices available to be used for the required four 4×4 products. For high accuracy designs, Mul44_acc can be used for the three more significant products, i.e. $A_H B_H$, $A_H B_L$ and $A_L B_H$, and any of the other two approximate designs can be used for the least significant product, i.e., $A_L B_L$. The proposed multiplier Mul88_1 uses Mul44_1 for the computation of $A_L B_L$; the proposed multiplier Mul88_2 uses Mul44_2 for the same computation. Both multipliers use Mul44_acc for the computation of the other three products.

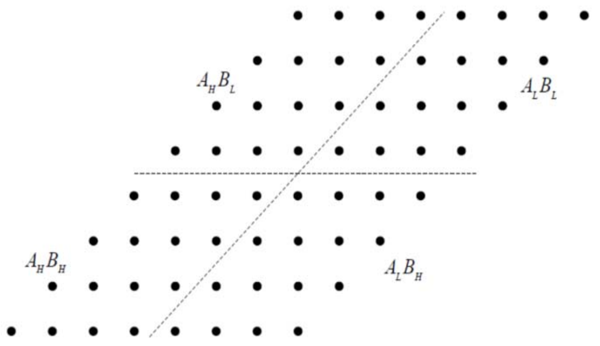


Fig. 6. 8 by 8 bit multiplication broken down into four parts of 4 by 4 bit multiplications (using recursive multiplication)

In many cases, accuracy is not a strict requirement and hence, it can be traded off with power, delay and area. When designing power efficient, low accuracy multipliers, Mul44_acc is used only for calculating the most significant product $A_H B_H$ and either Mul44_1 or Mul44_2 can be used for calculating the other three less significant products. The proposed multiplier Mul88_3 uses Mul44_1 for the other three products, while Mul88_4 uses Mul44_2 for the other three products. After the calculation of the four products, the two products $A_H B_L$ and $A_L B_H$ are added using a 9-bit adder (so not a 16-bit adder thus saving hardware); the result is added with the other two products $A_H B_H$ and $A_L B_L$ using a 16-bit adder.

IV. SIMULATION RESULTS

In this section, the designs of the proposed multipliers as explained in Section III are evaluated. Multiplier 4 of [10] as its best design is also simulated and compared. The error metrics (error rate and NED) and the design metrics (power, delay and area) have been considered and compared for all designs. For analyzing power, delay and area, the designs are implemented in VHDL and then synthesized by Synopsis Design Vision. STM 65nm process technology is utilized for the design in the analysis. For the error metrics, the designs are implemented on MATLAB and Monte-Carlo simulations are performed to establish the error rate and NED. In the first subsection, the measures of the three 4x4 bit multipliers are presented, while in the second subsection, the measures for the proposed 8x8 bit designs are presented.

A. 4X4 bit designs

The circuit measures of the 4×4 bit designs are shown in Table I.

Table I. Power, area and delay comparison for Mul44_1, Mul44_2 and accurate 4X4 bit multiplier

Design	Power (μW)	Delay (ns)	Area (μm^2)
Mul44_1	18.746	1.49	138.319
Mul44_2	20.8905	1.43	139.359
Mul44_acc	29.8582	1.66	166.39

The first two designs using the approximate compressors are more efficient in terms of power, delay and area than the accurate design; however, there is no significant difference between Mul44_1 and Mul44_2 and hence, both are equally good in terms of the design metrics.

The error characteristics of the 4×4 bit designs are shown in Table II.

Table II. Accuracy comparison for Mul44_1 and Mul44_2

Design	Average NED	Pass Rate (%)	Error Rate (%)
Mul44_1	0.0503	32.42	67.58
Mul44_2	0.0139	60.93	39.07
Mul44_acc	0	100	0

Mul44_2 is more accurate than Mul44_1 with a fairly small NED and nearly 100% better than Mul44_1 in pass rate. This is also an indication that the approximate compressor design 2 of [10] is better than its approximate compressor 1 design. Since the design metrics of both Mul44_1 and Mul44_2 are almost the same, it becomes an obvious choice to use Mul44_2 instead of Mul44_1 for the 4x4 bit approximate product.

8x8 bit multiplier designs are also simulated with Mul88_1 and Mul88_3 using Mul44_1 and Mul44_acc while Mul88_2 and Mul88_4 using Mul44_2 and Mul44_acc as in Section III. As expected the designs using Mul44_2 have a better accuracy than the similar designs using Mul44_1 (Table II) as confirmed by simulation.

B. 8X8 bit designs

The design metrics of the proposed designs and the multiplier 4 of [10] are shown in Table III.

Table III. Power, area and delay comparison of multipliers

Design	Power (μW)	Delay (ns)	Area (μm^2)
Mul88_1	168.7982	2.97	808.5999
Mul88_2	170.8133	2.91	809.6399
Mul88_3	140.7181	2.97	752.4399
Mul88_4	151.3424	2.91	755.5599
Multiplier 4 [10]	176.4869	3.13	727.999
Mul88_acc	179.7635	3.14	836.6799

Mul88_acc is the accurate 8×8 multiplier made using Mul44_acc for all four products and has been simulated as reference. In Table III, the proposed designs have a significantly better area, power and delay than the reference accurate design. Also, all proposed designs consume less power and have a smaller delay than the multiplier 4 of [10], even though the proposed designs requires slightly more area than multiplier 4 of [10]. The area consumption is higher because of the use of one additional 9-bit adder in the proposed designs; this was not required in the multiplier design of [10]. However, a significant reduction in power and delay has been obtained by the proposed designs due to the significant reduction in the number of compressors used.

The accuracy comparison of the multipliers is shown in Table IV.

Table IV. Accuracy comparison of multipliers

Design	Avg. NED ($x10^{-3}$)	Pass Rate (%)	Error Rate (%)
Mul88_1	0.17397	32.42	67.58
Mul88_2	0.048058	60.94	39.06
Mul88_3	5.00	6.59	93.41
Mul88_4	1.40	30.18	69.82
Multiplier 4 [10]	0.74581	13.28	86.72
Mul88_acc	0	100	0

As expected, the proposed multipliers Mul88_1 and Mul88_2 are very good in terms of accuracy. Compared to multiplier 4 of [10], Mul88_1 and Mul88_2 are very accurate and Mul88_2 has a NED that is almost one order of magnitude lower. Mul88_1 and Mul88_2 also have a high pass rate; Mul88_2 generates 39936 correct outputs as out of 65536, so a very high accuracy (and a pass rate of more than 60%).

Hence, Mul88_2 is better than Mul88_1 in almost every aspect and is undoubtedly, the best design proposed in this paper. It can be used in the applications requiring a very high accuracy with great savings in power and delay. This again proves the effectiveness of design 2 compressor over the design 1 compressor of [10]. The effectiveness of this design is also shown in the image processing application, as shown in the next section. Although Mul88_3 and Mul88_4 are not very good in terms of accuracy, they achieve a significant reduction in power dissipation. Therefore, they can be used in applications that require a significant saving in power and can tolerate loss in accuracy. Mul88_4 is better between these two designs.

V. IMAGE PROCESSING

This section presents an image processing application using the proposed 8x8 bit multipliers as well as design 4 of [10]. An image sharpening algorithm is considered and functionally implemented in Matlab, with an Intel Core i5 processor with 4GB RAM. The processed image quality is measured by the *peak signal noise ratio* (PSNR); the PSNR quantifies the maximum possible power of signal and the power of an image with loss of accuracy following an additional process, such as compression and/or approximate computation. The PSNR is usually used to measure the quality of a reconstructive process involving information loss and is defined by the mean square error (MSE). Given an accurate image I and an image K generated by an approximate process, the MSE is defined as:

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \quad (15)$$

The PSNR is defined as:

$$PSNR = 20log_{10}(\frac{MAX_I}{\sqrt{MSE}}) \quad (16)$$

The term MAX_I is the maximum possible pixel value in the image; for example, when a pixel is represented by 8 bits, then its maximum value is 255. If I is the original image and S is the processed image, the sharpening algorithm of [12] performs as

$$S(x,y) = 2I(x,y) - \frac{1}{273}\sum_{i=-2}^{2}\sum_{j=-2}^{2} G(i+3,j+3)I(x-i,y-j) \quad (17)$$

where G is a matrix given by:

$$G = \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix} \quad (18)$$

One example image is selected for the sharpening algorithm and is executed using different approximate multipliers; the PSNR and NED values are then measured. The original and processed images are shown in Figs. 7 and 8. When compared to the accurate results, all approximate designs (except design Mul88_3 and Mul88_4) produce images whose quality degradation is not perceived by the human eyes. Fig. 8 (e) and (f) are processed by Mul88_3 and Mul88_4 respectively, and some details are not very sharp, indicating that these multipliers are not suitable for approximate computing compared with other designs. Table V shows the PSNR and NED values for the example image. Mul88_2 obtains the best PSNR and the lowest NED, Mul88_1 achieves the second best performance. Image quality for both the designs is better than design 4 of [10].

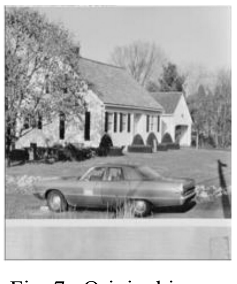


Fig. 7. Original image

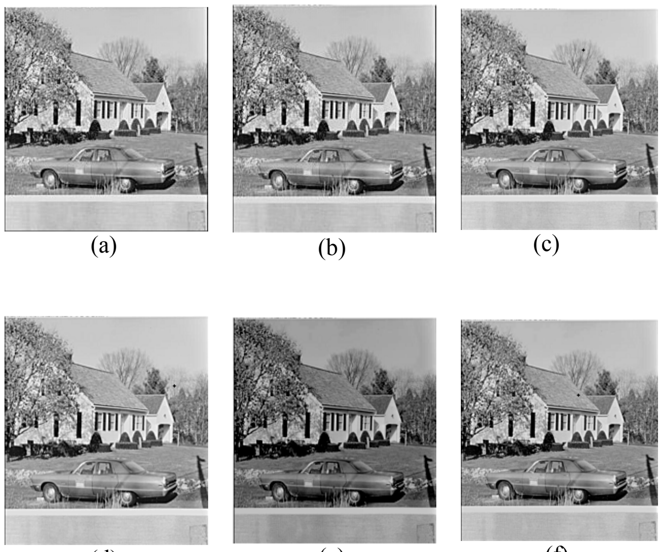


Fig. 8. Sharpened images with the following multipliers: (a) Accurate, (b) Multiplier 4 of [10], (c) Mul88_1, (d) Mul88_2, (e) Mul88_3, (f) Mul88_4.

Table V. PSNR and NED values for image sharpening

Design	PSNR (dB)	NED ($x10^{-5}$)
Mul88_1	48.11	1.17
Mul88_2	54.71	0.335
Mul88_3	16.32	48
Mul88_4	26.76	14.55
Multiplier 4 [10]	35.62	5.18

VI. CONCLUSION

In this paper, four designs of 8x8 bit approximate multipliers have been proposed. Simulation results have been reported for design and error metrics. Also, an image processing application has been presented in detail. The proposed designs show significant improvements in accuracy, power and latency at a cost of a slightly larger area.

For accuracy, the proposed Mul88_1 and Mul88_2 achieve improvements of 76.67% and 93.55% over design 4 of [10] (as the most accurate design in [10]). The four proposed designs are also efficient in terms of power, showing improvements of 4.36%, 3.21%, 20.27% and 14.24%, respectively, compared to design 4 of [10]. An improvement in delay is also obtained in the proposed multipliers, but the circuit areas of the proposed designs are higher than design 4 of [10] by 11.07%, 11.21%, 3.35% and 3.78%, respectively. This is due to the use of an additional 9-bit adder (in [10], only one 16-bit adder is required).

In summary, the first two proposed designs, Mul88_1 and Mul88_2, are suitable for error tolerant applications that require a high accuracy; Mul88_2 achieves the most accurate result. The other two designs, Mul88_3 and Mul88_4, are suited for low power applications in which a larger degradation in accuracy can be tolerated.

ACKNOWLEDGEMENT

The authors would like to acknowledge the support from the MITACS Globalink program for a summer internship carried out at the University of Alberta.

REFERENCES

[1] J. Han and M. Orshansky, “Approximate Computing: An Emerging Paradigm for Energy-Efficient Design,” in ETS’13, Avignon, France, May 27-31, 2013, pp. 1 – 6.

[2] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, “MACACO: Modeling and analysis of circuits for approximate computing,” in ICCAD 2011, pp. 667 – 673.

[3] J. Liang, J. Han, F. Lombardi, “New metrics for the reliability of approximate and probabilistic adders,” IEEE Trans. on Computers, vol. 63, no. 9, pp. 1760 - 1771, 2013.

[4] K.Y. Kyaw, W.L. Goh, K.S. Yeo, "Low-power high-speed multiplier for error-tolerant application," IEEE International Conference of Electron Devices and Solid-State Circuits (EDSSC), 2010.

[5] P. Kulkarni, P. Gupta, M. Ercegovac, "Trading accuracy for power with an Underdesigned Multiplier architecture," 24th International Conference on VLSI Design, 2011.

[6] H.R. Mahdiani, A. Ahmadi, S.M. Fakhraie, C. Lucas, "Bio-Inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," IEEE Transactions on Circuits and Systems, vol. 57 no. 4, 2010.

[7] C.-H. Lin, I.-C. Lin, "High accuracy approximate multiplier with error correction," IEEE 31st International Conference on Computer Design (ICCD), 2013.

[8] K. Bhardwaj, P.S. Mane, J. Henkel, "Power- and area-efficient Approximate Wallace Tree Multiplier for error-resilient systems," 15th International Symposium on Quality Electronic Design (ISQED), 2014.

[9] C. Liu, J. Han and F. Lombardi, “A Low-Power, High-Performance Approximate Multiplier with Configurable Partial Error Recovery,” DATE 2014, Dresten, Germany, 2014.

[10] A. Momeni, J. Han, P. Montuschi, F. Lombardi, "Design and analysis of approximate compressors for multiplication," IEEE Transactions on Computers, in press, 2014.

[11] C.H. Chang, J. Gu, M. Zhang, "Ultra low-voltage low-power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits," IEEE Transactions on Circuits and Systems, vol.51, no.10, pp.1985-1997, 2004.

[12] M.S.K Lau, K.V. Ling, Y.C. Chu. "Energy-aware probabilistic multiplier: design and analysis." In Proceedings of the 2009 international conference on Compilers, architecture, and synthesis for embedded systems, Grenoble, France, pp. 281-290, 2009.