GRD Journals | GRD Journal for Engineering | National Conference on Emerging Trends in Electrical, Electronics and Computer Engineering (ETEEC-2018) | April 2018
e-ISSN: 2455-5703

Design of Energy Efficient Approximate Multiplier

[1]J. Gayathri [2]S. Sowmiya [3]S. K. Soundriya Leela [4]S. Bhavatharani
[1,2,3,4]Department of Electronics and Communication Engineering
[1,2,3,4]SCAD Institute of Technology, Palladam, India

Abstract

Multiplier is one of the arithmetic operations that are used in VLSI circuits. Approximate multiplier is designed by using half adder, full adder and 4-2 compressor. Approximate multiplier is used to reduce the logic gate count, power consumption, delay and it provides high speed output. Area and speed of approximate multiplier is efficient than the conventional multipliers. This adder is mainly used in DSP Application, Image Processing. The simulation result shows the low power consumption by using Xilinx ISE simulation tool.

Keyword- Approximate Computing, Gates, Error Analysis

I. INTRODUCTION

In applications like multimedia signal processing and data mining which can tolerate error, exact computing units are not always necessary. They can be replaced with their approximate counterparts. Research on approximate computing for error tolerant applications is on the rise. Adders and multipliers form the key components in these applications. In [1], approximate full adders are proposed at transistor level and they are utilized in digital signal processing applications. Their proposed full adders are used in accumulation of partial products in multipliers.

To reduce hardware complexity of multipliers, truncation is widely employed in fixed-width multiplier designs. Then a constant or variable correction term is added to compensate for the quantization error introduced by the truncated part [2], [3]. Approximation techniques in multipliers focus on accumulation of partial products, which is crucial in terms of power consumption. Broken array multiplier is implemented in [4], where the least significant bits of inputs are truncated, while forming partial products to reduce hardware complexity. The proposed multiplier in [4] saves few adder circuits in partial product accumulation.

In [5], two designs of approximate 4-2 compressors are presented and used in partial product reduction tree of four variants of 8×8 Dadda multiplier. The major drawback of the proposed compressors in [5] is that they give nonzero output for zero valued inputs, which largely affects the mean relative error (MRE) as discussed later. The approximate design proposed in this brief overcomes the existing drawback. This leads to better precision. In static segment multiplier (SSM) proposed in [6], m-bit segments are derived from n-bit operands based on leading 1 bit of the operands. Then, $m \times m$ multiplication is performed instead of $n \times n$ multiplication, where $m<n$. Partial product perforation (PPP) multiplier in [7] omits k successive partial products starting from jth position, where $j \in [0, n-1]$ and $k \in [1, \min(n-j, n-1)]$ of a n-bit multiplier. In [8], 2×2 approximate multipliers based on modifying an entry in the Karnaugh map is proposed and used as a building block to construct 4×4 and 8×8 multipliers. In [9], inaccurate counter design has been proposed for use in power efficient Wallace tree multiplier. A new approximate adder is presented in [10] which is utilized for partial product accumulation of the multiplier. For 16-bit approximate multiplier in [10], 26% of reduction in power is accomplished compared to exact multiplier. Approximation of 8-bit Wallace tree multiplier due to voltage over-scaling (VOS) is discussed in [11]. Lowering supply voltage creates paths failing to meet delay constraints leading to error.

Previous works on logic complexity reduction focus on straightforward application of approximate adders and compressors to the partial products. In this brief, the partial products are altered to introduce terms with different probabilities. Probability statistics of the altered partial products are analyzed, which is followed by systematic approximation. Simplified arithmetic units (half-adder, full- adder, and 4-2 compressor) are proposed for approximation. The arithmetic units are not only reduced in complexity, but care is also taken that error value is maintained low. While systemic approximation helps in achieving better accuracy, reduced logic complexity of approximate arithmetic units consumes less power and area. The proposed multipliers outperforms the existing multiplier designs in terms of area, power, and error, and achieves better peak signal to noise ratio (PSNR) values in image processing application. Error distance (ED) can be defined as the arithmetic distance between a correct output and approximate output for a given input. In [12], approximate adders are evaluated and normalized ED (NED) is proposed as nearly invariant metric independent of the size of the approximate circuit. Also, traditional error analysis, MRE is found for existing and proposed multiplier designs.

The rest of this brief is organized as follows. Section II details the proposed architecture. Section III provides extensive result analysis of design and error metrics of the proposed and existing approximate multipliers. The proposed multipliers are utilized in image processing application and results are provided in Section IV. Section V concludes this brief.

II. PROPOSED ARCHITECTURE

To Design the Approximate Multiplier. The approximate multiplier consists of seven logics. It reduces the area, delay and power consumption compared to array multiplier. The approximate multiplier is suitable for image processing and it result is similar to accurate multiplier. The carry signal only propagates one bit higher in any situation in this approximate adder. A simple tree of the approximate adders is used for partial product accumulation and the error signals are used to compensate the error for obtaining a better accuracy.

Implementation of multiplier comprises three steps: generation of partial products, partial products reduction tree, and finally, a vector merge addition to produce final product from the sum and carry rows generated from the reduction tree. Second step consumes more power. In this brief, approximation is applied in reduction tree stage.

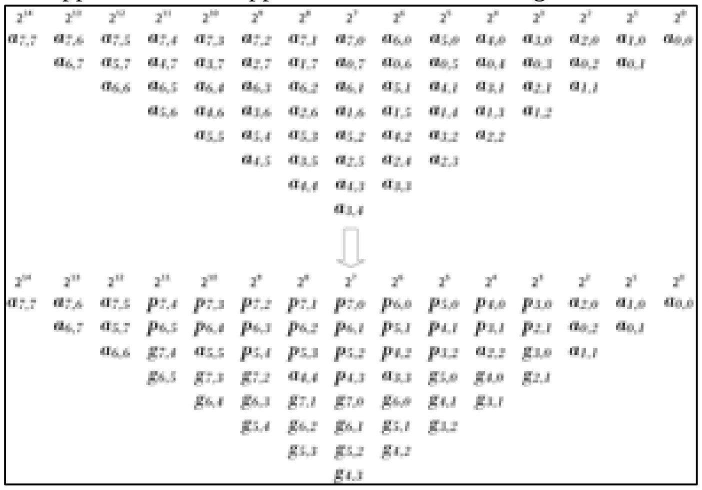
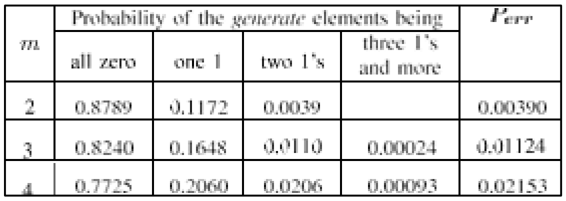


Fig. 1: Transformation of generated partial products into altered partial products



m	Probability of the *generate* elements being				P_{err}
	all zero	one 1	two 1's	three 1's and more	
2	0.8789	0.1172	0.0039		0.00390
3	0.8240	0.1648	0.0110	0.00024	0.01124
4	0.7725	0.2060	0.0206	0.00093	0.02153

Table 1: Probability Statistics of Generate Signals

From statistical point of view, the partial product $a_{m,n}$ has a probability of 1/4 of being 1. In the columns containing more than three partial products, the partial products $a_{m,n}$ and $a_{n,m}$ are combined to form propagate and generate signals as given in (1). The resulting propogate and generate signals form altered partial products $p_{m,n}$ and $g_{m,n}$. From column 3 with weight 23 to column 11 with weight 211, the partial products am,n and an,m are replaced by altered partial products $p_{m,n}$ and $g_{m,n}$. The original and transformed partial product matrices are shown in Fig. 1

$$p_{m,n} = a_{m,n} + a_{n,m}$$
$$g_{m,n} = a_{m,n} \cdot a_{n,m} \qquad (1)$$

The probability of the altered partial product $g_{m,n}$ being one is 1/16, which is significantly lower than 1/4 of $a_{m,n}$. The probability of altered partial product p_{m}, n being one is 1/16 + 3/16 + 3/16 = 7/16, which is higher than $g_{m,n}$. These factors are considered, while applying approximation to the altered partial product matrix.

A. Approximation of Altered Partial Products

The accumulation of generate signals is done column wise. As each element has a probability of 1/16 of being one, two elements being in the same column even decreases. For example, in a column with 4 generate signals, probability of all numbers being 0 is $(1 - pr)^4$, only one element being one is $4pr(1 - pr)^3$, the probability of two elements being one in the column is $6pr^2(1 - pr)^2$, three ones is $4pr^3(1-pr)$ and probability of all elements being 1 is pr^4, where pr is 1/16. The probability statistics for a number of generate elements m in each column are given in Table I. Based on Table I, using OR gate in the accumulation of column wise generate elements in the altered partial product matrix provides exact result in most of the cases. The probability of error (P_{err}) while using OR gate for reduction of generate signals in each column is also listed in Table I. As can be seen, the probability of misprediction is very low. As the number of generate signals increases, the error probability increases linearly. However, the value of error also rises. To prevent this, the maximum number of generate signals to be grouped by OR gate is kept at 4. For a column having m generate signals, m/4 OR gates are used.

Inputs		Exact Outputs		Approximate Outputs		Absolute Difference
$x1$	$x2$	$Carry$	Sum	$Carry$	Sum	
0	0	0	0	0✓	1✗	1
0	1	0	1	0✓	1✓	0
1	0	0	1	0✓	1✓	0
1	1	1	0	1✓	0✓	0

Table 2: Truth Table of Approximate Half Adder

Inputs			Exact Outputs		Approximate Outputs		**Absolute Difference**
$x1$	$x2$	$x3$	$Carry$	Sum	$Carry$	Sum	
0	0	0	0	0	0✓	1✗	1
0	0	1	0	1	0✓	1✓	0
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	1✓	0✓	0
1	1	1	1	1	1✓	0✗	1

Table 3: Truth Table of Approximate Full Adder

B. *Approximation of Other Partial Products*

The accumulation of other partial products with probability 1/4 for am,n and 7/16 for pm,n uses approximate circuits. Approximate half-adder, full-adder, and 4-2 compressor are proposed for their accumulation. Carry and Sum are two outputs of these approximate circuits. Since Carry has higher weight of binary bit, error in Carry bit will contribute more by producing error difference of two in the output. Approximation is handled in such a way that the absolute difference between actual output and approximate output is always maintained as one. Hence Carry outputs are approximated only for the cases, where Sum is approximated.

In adders and compressors, XOR gates tend to contribute to high area and delay. For approximating half-adder, XOR gate of Sum is replaced with OR gate as given in (2). This results in one error in the Sum computation as seen in the truth table of approximate half-adder in Table II. A tick mark denotes that approximate output matches with correct output and cross mark denotes mismatch.

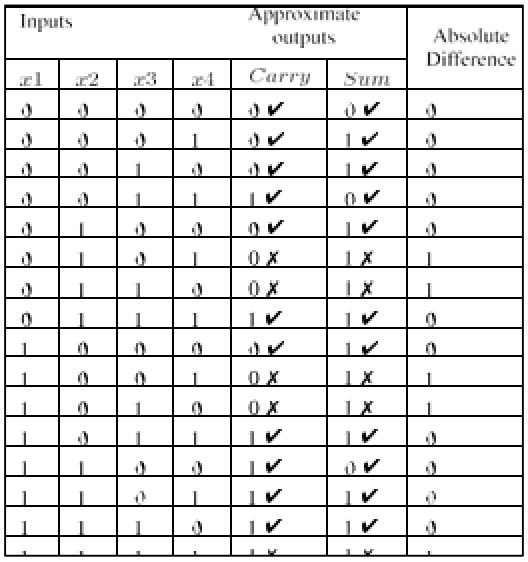
$$\text{Sum} = x1' + x2'$$
$$\text{Carry} = x1 \cdot x2 \qquad (2)$$

In the approximation of full-adder, one of the two XOR gates is replaced with OR gate in Sum calculation. This results in error in last two cases out of eight cases. Carry is modified as in (3) introducing one error. This provides more simplification, while maintaining the difference between original and approximate value as one. The truth table of approximate full-adder is given in Table III

$$\text{Sum} = (x2' \,.\, x3') + (x1' .\, x3') + (x1' .\, X2')$$
$$\text{Carry} = x1 + (x3 \cdot x2.)$$
$$\text{Carry} = W \cdot x3. \qquad (3)$$

Two approximate 4-2 compressors in [5] produce nonzero output even for the cases where all inputs are zero. This results in high ED and high degree of precision loss especially in cases of zeros in all bits or in most significant parts of the reduction tree. The proposed 4-2 compressor overcomes this drawback.

In 4-2 compressor, three bits are required for the output only when all the four inputs are 1, which happens only once out of 16 cases. This property is taken to eliminate one of the three output bits in



Inputs				Approximate outputs		Absolute Difference
$x1$	$x2$	$x3$	$x4$	$Carry$	Sum	
0	0	0	0	0 ✔	0 ✔	0
0	0	0	1	0 ✔	1 ✔	0
0	0	1	0	0 ✔	1 ✔	0
0	0	1	1	1 ✔	0 ✔	0
0	1	0	0	0 ✔	1 ✔	0
0	1	0	1	0 ✗	1 ✗	1
0	1	1	0	0 ✗	1 ✗	1
0	1	1	1	1 ✔	1 ✔	0
1	0	0	0	0 ✔	1 ✔	0
1	0	0	1	0 ✗	1 ✗	1
1	0	1	0	0 ✗	1 ✗	1
1	0	1	1	1 ✔	1 ✔	0
1	1	0	0	1 ✔	0 ✔	0
1	1	0	1	1 ✔	1 ✔	0
1	1	1	0	1 ✔	1 ✔	0
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Table 4: Truth Table of Approximate 4-2 Compressor

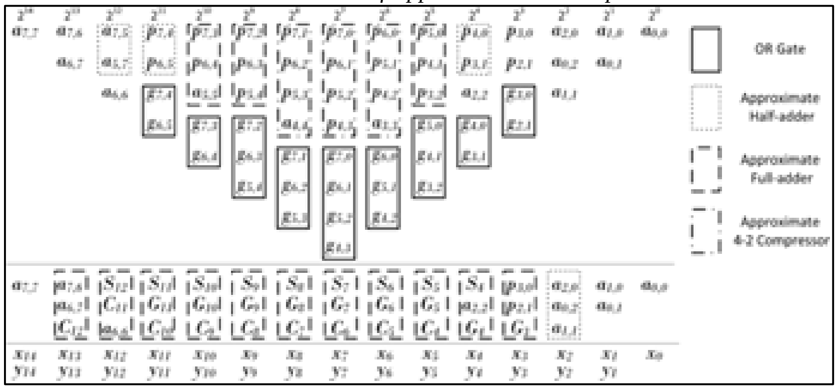


Fig. 2: Reduction of altered partial products

4-2 compressor. To maintain minimal error difference as one, the output "100" (the value of 4) for four inputs being one has to be replaced with outputs "11" (the value of 3). For Sum computation, one out of three XOR gates is replaced with OR gate. Also, to make the Sum corresponding to the case where all inputs are ones as one, an additional circuit x1· x2 · x3 · x4 is added to the Sum expression. This results in error in five out of 16 cases. Carry is simplified as in (4). The corresponding truth table is given in Table IV

$$W1 = x1 \cdot x2 \quad W2 = x3 \cdot x4$$
$$Sum = (x1 \oplus x2) + (x3 \oplus x4) + W1 \cdot W2 \quad Carry = W1 + W2 \qquad (4)$$

Fig. 2 shows the reduction of altered partial product matrix of 8 × 8 approximate multiplier. It requires two stages to produce sum and carry outputs for vector merge addition step. Four 2-input OR gates, four 3-input OR gates, and one 4- input OR gates are required for the reduction of generate signals from columns 3 to 11. The resultant signals of OR gates are labeled as G_i corresponding to the column i with weight 2^i. For reducing other partial products, 3 approximate half-adders, 3 approximate full-adders, and 3 approximate compressors are required in the first stage to produce Sum and Carry signals, S_i and C_i corresponding to column i. The elements in the second stage are reduced using 1 approximate half-adder and 11 approximate full-adders producing final two operands x_i and y_i to be fed to ripple carry adder for the final computation of the result.

III. RESULTS AND DISCUSSION

All approximate multipliers are designed for n = 16. The multipliers are implemented in Verilog and synthesized using Synopsys Design Compiler and a TSMC 65 nm standard cell library at the typical process corner, with temperature 25 °C and supply voltage 1 V. the proposed multipliers, the altered partial products are generated and compressed using half-adder, full-adder, and 4-2 compressor structures to form final two rows of partial products. The efficiency of the proposed multipliers is compared with existing approximate multipliers [5]–[8].

Exact 16-bit multiplier is designed using Dadda tree structure. Table shows compares all designs in terms of area, delay, power, power delay product (PDP), and area power product (APP). NED and MRE of the approximate multipliers are listed in Table VI. If high approximation can be tolerated for saving more power, Multiplier1 and ACM1 are the candidates to be considered. It can be seen that Multiplier1 has better APP, whereas ACM1 has better PDP. However, Multiplier1 has 64% lower NED and three orders of magnitude lower MRE, compared to ACM1. It should be noted that high values of MRE for ACMs are due to nonzero output for inputs with all zeros.

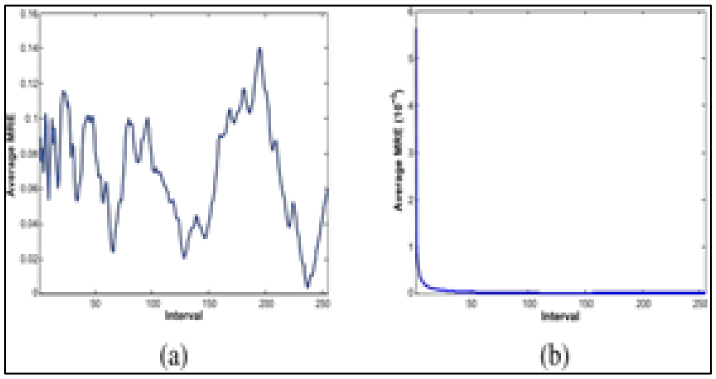


Fig. 3: .MRE distribution of (a) Multiplier1 and (b) Multiplier2

IV. Application - Image Processing

Geometric mean filter is widely used in image processing to reduce Gaussian noise [13]. The geometric mean filter is better at preserving edge features than the arithmetic mean filter. Two 16bits per pixel gray scale images. The algorithms are coded and implemented in MATLAB. Exact and approximate 16-bit multipliers are used to perform multiplication between 16-bit pixels. PSNR is used as figure

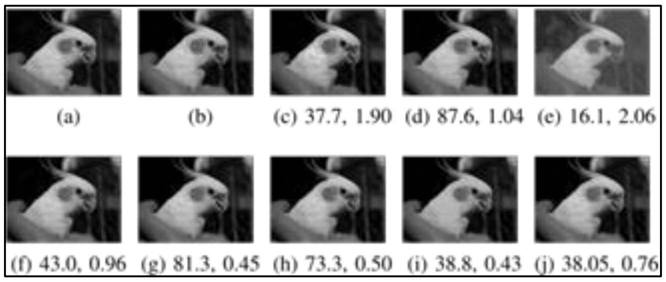


Fig. 4: (a) Input image-1 with Gaussian noise. Geometric mean filtered images and corresponding PSNR and energy savings in µJ using (b) exact multiplier, (c) Multiplier1, (d) Multiplier2, (e) ACM1, (f) ACM2, (g) SSM, (h) PPP, (i) UDM, and (j) VOS

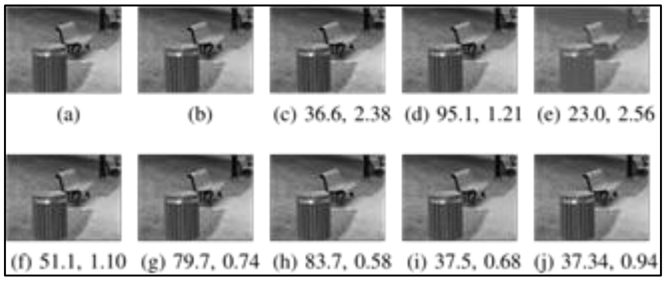


Fig. 5: (a) Input image-2 with Gaussian noise. Geometric mean filtered images and corresponding PSNR and energy savings in µJ using (b) exact multiplier, (c) Multiplier1, (d) Multiplier2, (e) ACM1, (f) ACM2, (g) SSM, (h) PPP, (i) UDM, and (j) VOS.

Of merit to assess the quality of approximate multipliers. PSNR is based on mean-square error found between resulting image of exact multiplier and the images generated from approximate multipliers.

Energy required by exact and approximate multiplication process while performing geometric mean filtering of the images is found using Synopsys Primetime. Further, exact multiplier is voltage scaled from 1 to 0.85 V (VOS), and its impact on energy consumption and image quality is computed.

The noisy input image and resultant image after denoising using exact and approximate multipliers, with their respective PSNRs and energy savings in µJ are shown in Figs. 4 and 5, respectively. Energy required for exact multiplication process for image-1 and image-2 is 3.24 and 2.62 µJ, respectively. Although ACM1 has better energy savings compared to Multiplier1, Multiplier1 has significantly higher PSNR than ACM1. Multiplier2 shows the best PSNR among all the approximate designs. Multiplier2 has better energy savings, compared to ACM2, PPP, SSM, UDM, and VOS. The intensity of image-1 being mostly on the lower end of the histogram causes poor performance of ACM multipliers. As the switching activity impacts most significant part of the design in VOS, PSNR values are affected.

V. CONCLUSION

In this paper, we proposed the design and simulation of a 16-bit approximate multiplier. Partial products of the multipliers are modified using generate and propagate signals. Approximate Half adder, Full adder and 4-2 compressor are proposed to reduce partial products. The total number of transistors used in approximate multiplier is reduced. It is easy to say the area and delay of approximate multiplier is reduced than the Existing multiplier. And, the total power consumption of approximate multiplier is 0.883W. And, it is used for signal and image processing.

REFERENCES

[1] Rong Ye, Ting Wang, Feng Yuan, Rakesh Kumar and Qiang Xu, "On Reconfiguration - Oriented Approximate Adder Design and Its Application", International journal of scientific and engineering research, vol-7, issue -5, May 16.
[2] Sumant Mukherjee and Dushyant Kumar Soni, "High Speed and Energy Efficient Approximate Adder for DSP Application", International Journal of Computer Science and Mobile Computing, Vol. 4, Issue 6, June 2015.
[3] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," IEEE Trans. Computer Aided Design Integer. Circuits Syst., vol. 32, no. 1, pp. 124–137, Jan. 2013.
[4] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," IEEE Trans. Comput., vol. 64, no. 4, pp. 984–994, Apr. 2015.
[5] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim, "Energy-efficient approximate multiplication for digital signal processing and classification applications," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 6, pp. 1180–1184, Jun. 2015.
[6] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi, "Design-efficient approximate multiplication circuits through partial product perforation," IEEE Trans. Very Large Scale Integer. (VLSI) Syst., vol. 24, no. 10, pp. 3105–3117, Oct. 2016.
[7] C. Liu, J. Han, and F. Lombardi, "A low- power, high-performance approximate multiplier with configurable partial error recovery," in Proc. Conf. Exhibit. (DATE), 2014, pp. 1–4.
[8] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "MACACO: Modeling and analysis of circuits for approximate computing," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), Oct. 2011, pp. 667–673.
[9] Jian Chen,Xingguo Xiaong and Mirza Rashid Hasan, "PSPICE Implementation of an 8-bit Low power Energy Recovery Full Adder", ASEE 2014 Zone 1st conference,2014.