A Low-Power Mitchell based Error Tolerant Multiplier

Aly Sultan, *Member, IEEE,* Ali Elhussien, *Fellow,OSA,* and Dr. Hassan Mostafa, *Life Fellow, IEEE*

***Abstract*—Approximate multipliers have recently gained significance due to their reduced area and power consumption in applications that exhibit a high degree of error tolerance. This comes at the cost of accuracy which is a primary design parameter when selecting the appropriate approximate multiplier for a given application. In this paper, several approximate multiplier architectures are combined with the approximation method of the error tolerant multiplier (ETM) to produce hybrid designs. The proposed hybrid multipliers are evaluated using an image compression application - JPEG compression using Discrete Cosine Transform (DCT). The Mitchell based ETM was found to provid the greatest tradoff in terms of Area, Power, Delay and Accuracy in JPEG Compression, with an average PDP reduction of 47.5%, an area reduction of 62.8% and a corresponding 20% reduction in output image Peak Signal to Noise Ration (PSNR).**

***Index Terms*—Approximate Computing, adder, multiplier, low-energy design, imprecise computational blocks**

I. Introduction

WITH the increasing prevalence of mobile and embedded systems, demand for ultra lower power, small footprint, and high performance hardware is greater than ever. Approximate computing aims to satisfy this demand by trading computationally accuracy for superior performance and reduced power and area consumption. The effectiveness of approximate computing can be seen in applications like image processing where error tolerance is a product of human perceptual limitations [1]. Most approximate multipliers have been designed using the following methodologies:

1) Approximation in partial products generation [2]
2) Approximation in partial product tree [3], [4]
3) Approximation in partial products summation [5]
4) Mitchell's Approximate logarithmic multiplication [6]
5) Approximate hardware generation using genetic algorithms [7]

Of the previous methodologies ETM [4] allows the greatest flexibility when attempting to create hybrid approximate multipliers with significantly reduced hardware. In this paper several of the previously mentioned approximate multiplier designs are combined with ETM to create hybrid unsigned 16-bit approximate multipliers that are compared with each other while considering their error performance and circuit characteristics. The rest of the paper is organized as follows section II describes the ETM architecture and the proposed modification used to create more efficient hybrid approximate multipliers. Section III briefly describes the approximate multiplier architectures that are combined with ETM. Section IV details the error and circuit performance characteristics of the proposed hybrid multipliers. Section V details the evaluation of the different multipliers using JPEG compression, specifically in the Discrete Cosine Transform (DCT) encoding part of JPEG compression. A brief conclusion is given in section VI.

II. The Hybrid Error Tolerant Multiplier

A. Original Design

An illustration of the ETM algorithm is provided in Fig.1. Initially. both the multiplicand $(65295)_{10}$ = $(1111111100001111)_2$ and the multiplier $(36472)_{10}$= $(1000111001111000)_2$ are segmented into two equal 8 bit parts. When these inputs are fed to the multiplier, the upper parts of both operands are sent to an 8bit exact multiplier and the lower parts of both operands are sent to an approximation unit. The 8 bit Exact multiplier generates the upper 16 bits of the resulting 32 bit number and the approximation unit applies an approximation algorithm to generate the lower 16 bits. This algorithm begins at the point where the inputs are segmented into upper and lower parts then it progresses moves to rightmost bits of the input operands. An OR operation is applied to the lower bits of the operands. When the algorithm detects the presence of the first logic '1' output from the OR operation it begins to approximates the rest of the 16 bit output by setting all bits past the first '1' bit to a logic '1'. In the event where no logic '1' bit is detected when an OR operation is applied to lower bits of both operands, the lower 16 bits of the output are assumed to be logic '0'. If the upper parts of both operands are equal to zero, then the lower parts of both operands are sent to the exact multiplier. The starting point for the the multiplier is arbitrary. Moving the starting point left reduces the size of the exact multiplier which minimizes hardware but also decreases accuracy. Moving the starting point right increases accuracy at the cost of a larger multiplier.

B. Proposed Modification

The segmentation of a 16 bit multipliers into two components (Approximate, and Exact) greatly reduces power and area consumption of the multiplier by around 50% [4] due to it's removal of 50% of the exact multiplier's hardware. Additionally because of the flexibility provided by this segmentation, the exact part of the multiplier can also be replaced by an approximate multiplier to further reduce power and area consumption. This approximate multiplier must be able to evaluate a majority of the most significant bits of the input operands more precisely than the lower bits to limit output inaccuracy.

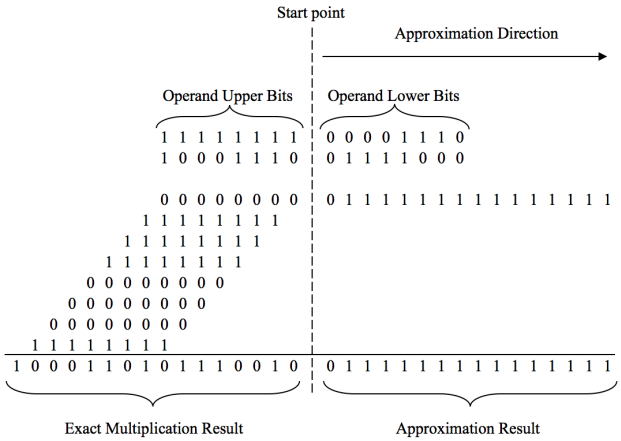

Fig. 1. ETM algorithm

III. Approximate Multipliers

A. Truncated Binary Multiplier

Petra et al. proposed a truncated binary nxn multiplier (TBM) that computes only the upper n bits of the expected 2n bit output [2]. They also introduce a suitable compensation function that minimizes the mean square error of the truncated output. Petra et al. obtain a closed form expression for the compensation function to be used in multipliers with arbitrary truncation points.

B. Broken Array Multiplier

Mahdini et al. proposed an approximate broken array multiplier (BAM) composed of carry select adder cells (CSA) and an nbit row of vector merging cells [3]. Depending on the desired reduction in hardware necessary Mahdini proposes the removal of several CSA blocks according to two design parameters, the vertical break line (VBL), and the horizontal break line (HBL). The VBL moves from the right-most CSA block to the left and depending on the position of the line any CSA blocks that fall to the right of it are omitted with their output assumed to be null. Similarly the HBL moves from top to bottom and all CSA blocks falling above the HBL are omitted with their output assumed to be null. BAM designs are defined by their VBL and HBL index number. A BAM with VBL equal to four and HBL equal to one refers to a BAM design with the first CSA cell row is omitted, and the first four columns are also omitted.

C. Approximated Partial Product Summation Multipliers

Masadeh et al. proposed several array based and tree based approximate multipliers composed of several approximate full adder cells [5]. The proposed designs are defined by the percentage and type of approximate full adders used. Designs considered in this paper are designs are both array based and tree based multipliers were only the full adders and compressors that contribute to the lowest 50% of the output bits are approximate. The designs considered in this paper are the array based EM4 and compressor based CEM5. The EM4 multiplier replaces 50% of the full adders in an array multiplier with an AMA4 full adder. The CEM5 multiplier uses approximate compressors to evaluate partial production summation for the lower 50% partial products. The compressors are based on the AMA5 full adder cell which is a set of two buffers connected to the inputs. The truth table for both the AMA cells is given in fig.2

Inputs			AMA4FA		AMA5A	
a	b	cin	sum	carry out	sum	carry out
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	1	0	1
1	0	1	0	1	0	1
1	1	0	0	1	1	1
1	1	1	1	1	1	1

Fig. 2. Truth Table for AMA4FA and AMA5FA

D. Mitchell Based Approximate Multipliers

Mitchell Logarithm multiplication converts input operands into approximate logarithms, adds the converted logarithms and then gets the anti-logarithm of the result. This reduces the multiplication of the two operands to a simple addition. McLaren proposed a modified Mitchell based logarithm multiplier that uses a look up table with 64 correction values to improve on Mitchell's original design [6]. McLaren managed to reduce the mean error in the result from 3% to 0.04% with the addition of the look up table in the design. While other implementations

E. Approximate Multiplier Generation Using Genetic Algorithm

Vojtech et al. developed a library of approximate multipliers generated by a multi-objective Cartesian genetic programming algorithm [7]. The library documents each multipliers error performance metrics in addition to to their circuit characteristics. A selection of nine 8bit evo multipliers with varied error performance metrics was chosen and used in this paper from the library provided in [8].

IV. Error Performance and Circuit Characteristics

A. Error Performance

Error performance was evaluated in MATLAB using a test-set of 100 million random multiplications. The operands in the test-set were all normally distributed 16bit numbers. The error rate (ER), mean relative error distance (MRED), and normalized mean error distance (NMED) are the primary metrics used to evaluate approximate multiplier performance [15]. ER is the rate at which a given design produces an incorrect output. The Error Distance (ED) is defined as the distance between the approximate result of a multiplication $M^{'}$ and the exact result of a multiplication M for each of the evaluated test cases as in (1). The Relative Error Distance (RED) is the scaling of ED by the exact result M as in 2. The Mean Error Distance (MED) is the mean of the evaluated ED's for all the test cases as in (4) where n_t is the number of test cases used. The Mean Relative Error Distance (MRED)

is the mean of the evaluated REDs for the given test set as defined in 4. The Normalized Mean Error Distance (NMED) is the normalization of MED by the maximum possible output of the design. It is a function of the number of output bits of the multiplier n_i as defined in 5. In all of the proposed designs in this paper $n_i = 32$. NMED's purpose is to compare multiplier error performance across different multiplier sizes.

$$ED^{(i)} = |M'^{(i)} - M^{(i)}| \tag{1}$$

$$RED^{(i)} = \frac{ED^{(i)}}{M^{(i)}} \tag{2}$$

$$MED = \frac{\sum_{i=1}^{n_t} ED^{(i)}}{n_t} \tag{3}$$

$$MRED = \frac{\sum_{i=1}^{n_t} RED^{(i)}}{n_t} \tag{4}$$

$$NMED = \frac{MED}{2^{n_i} - 1} \tag{5}$$

As seen in Fig.5, the ER for all multipliers was 100% since there are no 16bit input combinations that could result in an exact output being produced by any of the multiplers proposed. According to Fig.3, the Exact ETM had the lowest MRED and NMED of all the multipliers. BAM error performance progressively declined with the removal of more CSA blocks. The omission of CSA columns has a much greater effect on accuracy in comparison to the omission of CSA rows. BAMV6H3 has the worst error performance in terms of MRED and NMED which is understandable given that it ommits 57% of the total available CSA cells. Mitchell's approximate multiplier had the lowest MRED when compared to the Exact multiplier. However, according to Fig.4, NMED ranking follows MRED ranking for all multipliers except with TBM as it was analytically designed to produce the lowest mean square error through the use of a compensation function [2]. EVO MRED results were similair to those documented by [8] even with the addition of the ETM approximation, however, NMED for the EVO multipliers combined with ETM approximation was worse in comparison to EVO operating as a standalone 8 bit multipliers. CEM5 and EM4 are designed based on different approximate mirror full adders (AMA)[9]. CEM5 is based on had a higher MRED than EM4 because the AMA cells used in it's compressors had more erroneous test cases and thus a higher probability of error.

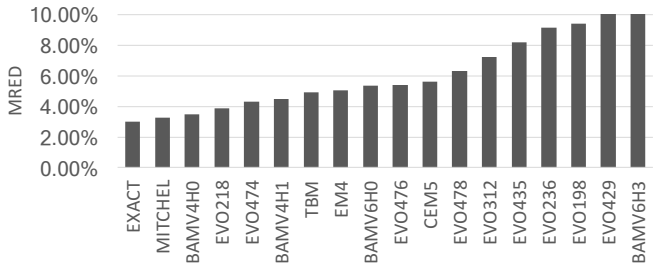

Fig. 3. MRED for different HETMs

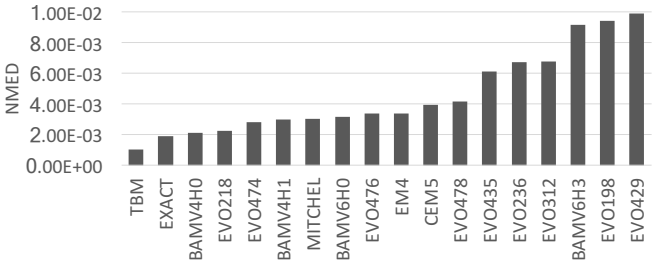

Fig. 4. NMED for different HETMs

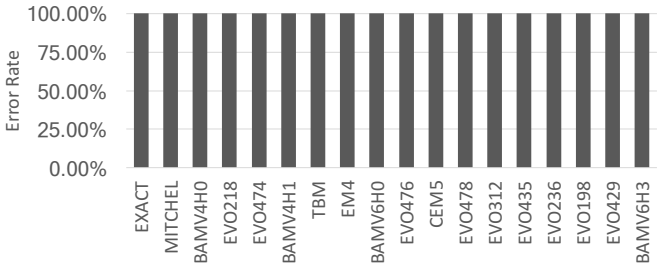

Fig. 5. ER for different HETMs

B. Circuit Characteristics

All hybrid multiplires where implemented on a Zynq xcz7020-1clg484 FPGA, with power and area estimates produced by Xilinx ISE post implemetation. Power estimation was performed in Xilinx XPower Analyzer using ten thousand randomly generated multiplications. Operand bits in each multiplication had an activity rate of 50% and input frequency was set to 10Mhz. PDP and Area Reduction percentages were evaluated relative to an array based 16 bit exact multiplier implemented on the same FPGA platform. Fig.6 shows the PDP and Area reduction for each of the different multipliers. Area reduction exceeded 50% for all characterised multipliers which is to be expected given the inclusion of the approximating unit. The highest reduction in both PDP and area is achieved by the evo198, BAMV6H3 and evo429. Significant gains in delay and power reduction can be observed in BAM based HETMs due to the omission of CSA cells. The movement of the VBL further left reduces delay in the critical path of the circuit by 1 CSA cell in addition to removing more CSA cells with each step left. EM4 performed well in terms of PDP given the simplicity of the AMA4 full adder cell it is based on. CEM5, while based on a simpler AMA5 full adder cell, performed worse in terms of power consumption due to it's internal node structure. The Exact HETM performed poorly in terms of PDP due to the 8x8 exact array mutliplier's high delay. All evo approximate multipliers where generated using a genetic algorithm, therefore no observations based on their structure can be made. However, there is a strong inverse relationship between superior error performance and PDP and Area reduction for all EVO multipliers.

V. Image Processing in JPEG Compression

Joint Photographics Experts Group (JPEG) is a lossy compression standard used for images. An integral part of JPEG is

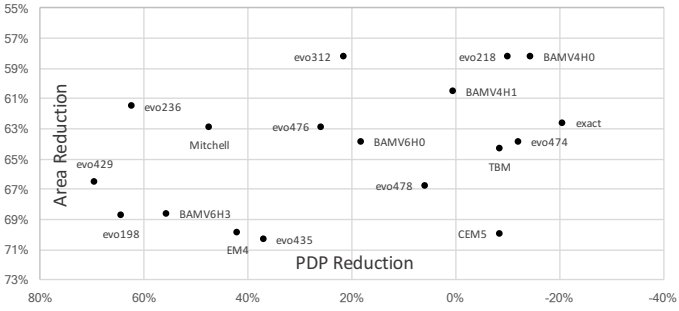

Fig. 6. PDP and Area Reduction Percentages for different multipliers

DCT [10], which is a mathametical transformation involving matrix multiplications of two $8*8$ matricies. To perform compression, DCT needs 512 multiplication operations per $8*8$ pixel block of any given grayscale image. If this operation is performed in realtime with approximate multipliers, resource use will be drastically reduced while maintaining satisfactory output image quality.

To evaluate real world error performance, the proposed hybrid multiplier are used in forward DCT to compress an image. A standard Lena image is compressed with various HETMs and the Peak Signal to Noise (PSNR) is evaluated for all compressed images using the uncompressed image as a reference. PSNR is appropriate for evaluating compressed image quality due to it's conistency with human visual quality analysis. PSNR is defined as follows

$$PSNR = 10\log_{10}\frac{255^2}{MSE} \tag{6}$$

Where 255 is the highest value a pixel can have in a grayscale image.

Fig.7 shows the original and compressed JPEG images using a selection of different approximate multipliers. The PSNR after compression using an exact 16bit multiplier, Mitchell ETM, EVO474 ETM, BAMv4H0 ETM, and EVO218 ETM are 32.08 dB, 25.75 dB, 22.89 dB, 19 dB and 18.07 dB respectively. In terms of PDP and Accuracy, Mitchell ETM provided superior performance and the lowest accuracy loss.

VI. Conclusion

In this paper we proposed several different hybrid Error tolerant multipliers based on the approximation method proposed in [4]. The proposed designs were evaluated based on PDP reduction, Area Reduction, MRED, NMED and accuracy loss in JPEG compression. JPEG compression was used to evaluate multiplier in terms of PDP and PSNR. The Mitchell based ETM was found to produce superior results in comparison to all other proposed multipliers in terms of PDP, and PSNR.

Fig. 7. (a) Original Image, Compressed JPEG images using (b) Exact 16bit Multiplier (c) Mitchell (d) EVO474 (e) BAMV4H0 (f) EVO312

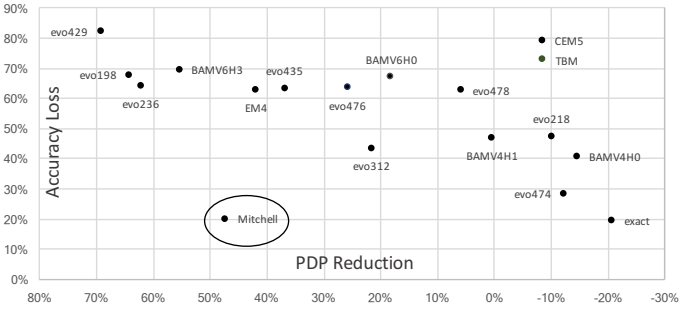

Fig. 8. PDP and Accuracy Loss in JPEG compression for different multipliers

References

[1] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *2013 18th IEEE European Test Symposium (ETS)*, May 2013, pp. 1–6.

[2] N. Petra, D. D. Caro, V. Garofalo, E. Napoli, and A. G. M. Strollo, "Truncated binary multipliers with variable correction and minimum mean square error," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 6, pp. 1312–1325, June 2010.

[3] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 4, pp. 850–862, April 2010.

[4] K. Y. Kyaw, W. L. Goh, and K. S. Yeo, "Low-power high-speed multiplier for error-tolerant application," in *2010 IEEE International Conference of Electron Devices and Solid-State Circuits (EDSSC)*, Dec 2010, pp. 1–4.

[5] M. Masadeh, O. Hasan, and S. Tahar, "Comparative study of approximate multipliers," in *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, ser. GLSVLSI '18. New York, NY, USA: ACM, 2018, pp. 415–418. [Online]. Available: http://doi.acm.org/10.1145/3194554.3194626

[6] D. J. Mclaren, "Improved mitchell-based logarithmic multiplier for low-power dsp applications," in *IEEE International [Systems-on-Chip] SOC Conference, 2003. Proceedings.*, Sept 2003, pp. 53–56.

[7] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, "Evoapproxsb: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, March 2017, pp. 258–261.

[8] "Evoapprox8b l approximate adders and multipliers library." [Online]. Available: http://www.fit.vutbr.cz/research/groups/ehw/approxlib

[9] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 124–137, Jan 2013.

[10] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 90–93, Jan 1974.