International Journal of Advance Research and Innovation
Vol. 7(2), Apr-Jun 2019, pp. 50-56
Doi: 10.51976/ijari.721907
www.gla.ac.in/journals/ijari

Article Info
Received: 13 May 2019 | Revised Submission: 20 May 2019 | Accepted: 28 May 2019 | Available Online: 15 Jun 2019

Image Multiplier Based on Low Power Approximate Unsigned Multiplier

J. Loganayaki and M. Vasanthi***

ABSTRACT

Approximate circuits have been considered for applications that can tolerate some loss of accuracy with improved performance and/or energy efficiency. Multipliers are key arithmetic circuits in many of these applications including digital signal processing (DSP). This multiplier leverages a newly designed approximate adder that limits its carry propagation to the nearest neighbours for fast partial product accumulation. Different levels of accuracy can be achieved by using either OR gates or the proposed approximate adder in a configurable error recovery circuit. The approximate multipliers using these two error reduction strategies are referred to as AM1 and AM2, respectively. Both AM1 and AM2 have a low mean error distance, i.e., most of the errors are not significant in magnitude. Compared with a Wallace multiplier optimized for speed, an 8×8 AM1 using four most significant bits for error reduction shows a 60% reduction in delay (when optimized for delay) and a 42% reduction in power dissipation (when optimized for area). In a 16×16 design, half of the least significant partial products are truncated for AM1 and AM2, which are thus denoted as TAM1 and TAM2, respectively. Compared with the Wallace multiplier, TAM1 and TAM2 save from 50% to 66% in power, when optimized for area. Compared with existing approximate multipliers, AM1, AM2, TAM1, and TAM2 show significant advantages in accuracy with a low power-delay product. AM2 has a better accuracy compared with AM1 but with a longer delay and higher power consumption. Image processing applications, including image sharpening and smoothing, are considered to show the quality of the approximate multipliers in error-tolerant applications. By utilizing an appropriate error recovery scheme, the proposed approximate multipliers achieve similar processing accuracy as exact multipliers, but with significant improvements in power.

Keywords: *Multiplier; Digital Signal Processing; Optimization.*

1.0 Introduction

Approximate computing has emerged as a potential solution for the design of energy-efficient digital systems. Applications such as multimedia, recognition and data mining are inherently error-tolerant and do not require a perfect accuracy in computation.

For Digital Signal Processing (DSP) applications, the result is often left to interpretation by human perception. Therefore, strict exactness may not be required and an imprecise result may suffice due to the limitation of human perception.

For these applications, approximate circuits play an important role as a promising alternative for reducing area, power and delay, thereby achieving better performance in energy efficiency.

As one of the key components in arithmetic circuits, adders have been extensively studied for

approximate implementation. As the typical carry propagation chain is usually shorter than the width of an adder, the speculative adders use a reduced number of less significant input bits to calculate the sum bits. An error detection and recovery scheme has been proposed to extend the scheme for a reliable adder with variable latency. A reliable variable-latency adder based on carry select addition has been presented. As a number of approximate adders have been proposed, new methodologies to model, analyze and evaluate them have been discussed.

A multiplier usually consists of three stages: partial product generation, partial product accumulation and a Carry Propagation Adder (CPA)

**Corresponding Author: Department of Electronics and Communication Engineering, Gnanamani College of Technology. Tamil Nadu, India (E-mail: jloganayakiece@gmail.com)*
***Department of Electronics and Communication Engineering, Gnanamani College of Technology. Tamil Nadu, India (E-mail: vasanthiece@gmail.com)*

at the final stage. In the Under Designed Multiplier (UDM), approximate partial products are computed using inaccurate 2 × 2 multiplier blocks, while accurate adders are used in an adder tree to accumulate the approximate partial products, approximate 4 × 4 and 8 × 8bit Wallace multipliers are designed by using a carry-in prediction method. Then, they are used in the design of approximate 16 × 16 Wallace multipliers, referred to as AWTM. The AWTM is configured into four different modes by using a different number of approximate 4 × 4 and 8 × 8 multipliers. The use of approximate speculative adders has been discussed in [10] for the final stage addition in a multiplier. The Error Tolerant Multiplier (ETM) is based on the partition of a multiplier into an accurate multiplication part for Most Significant Bits (MSBs) and a non-multiplication part for Least Significant Bits (LSBs). The Static

Segment Multiplier (SSM) utilizes a similar partition scheme. In an n ×n SSM, an m ×m accurate multiplier (m-n/2) is used to multiply the m consecutive bits from the two input operands. Whether the (n-m) MSBs of each input operand are all zero determines the selection of the inputs for the accurate multiplier (m MSBs or m LSBs). These approximate multipliers are designed for unsigned operation. Signed multiplication is usually implemented by using a Booth algorithm. Approximate designs have been proposed for fixed width Booth multipliers.

2.0 Literature Review

Generally, a multiplier consists of stages of partial product generation, accumulation and final addition. The commonly used partial product accumulation structures include the Wallace, Dadda trees and a carry-save adder array. In a Wallace tree, log2 (n) layers are required for an n-bit multiplier. The adders in each layer operate in parallel without carry propagation, and the same operation repeats until two rows of partial products remain. Therefore, the delay of the partial product accumulation stage is O(log2 (n)). Moreover, the adders in a Wallace tree can be considered as a 3:2 compressor and can be replaced by other counters or compressors (e.g. a 4:2 compressor) to further reduce the delay. The Dadda tree has a similar structure as the Wallace tree, but it uses as few adders as possible. For a carry-save adder array, the carry and sum signals generated by the adders in a row are connected to the adders in the next row. Adders in a column operate in series. Hence the partial product accumulation delay of an n-bit multiplier is approximately O(n), longer than that of the Wallace tree. However, an array requires a smaller area and thus lowers power dissipation due to the simple and symmetric structure.

3.0 Classification of Approximation Multiplier

- Approximation in generating the partial products
- Approximation (including truncation) in the partial product tree
- Using approximate counters or compressors in partial product tree.

3.1 Approximation in generating partial products

The Under Designed Multiplier (UDM) utilizes an approximate 2 × 2 bit multiplier block obtained by altering a single entry in the Karnaugh Map (K-Map) of its function. In this approximation, the accurate result “1001” for the multiplication of “11” and “11” is simplified to “111” to save one output bit. Assuming the value of each input bit is equally likely, the error rate of the 2 × 2 bit multiplier block is (1 2) 4 = 1 16 . Larger multipliers can be designed based on the 2 × 2 bit multiplier. This multiplier introduces an error when generating partial products, however the adder tree remains accurate.

3.2 Approximation in the partial product tree

A bio-inspired imprecise multiplier referred to as a Broken Array Multiplier (BAM) is proposed. The BAM operates by omitting some carry-save adders in an array multiplier in both horizontal and vertical directions. TheError Tolerant Multiplier (ETM) is divided into a multiplication section for the MSBs and a non-multiplication section for the LSBs. A NOR gate based control block is used to deal with two cases:

i) if the product of the MSBs is zero, then the multiplication section is activated to multiply the LSBs without any approximation
ii) if the product of the MSBs is nonzero, the non-multiplication section is used as an approximate multiplier to process the LSBs, while the multiplication section is activated to multiply the MSBs.

The Static Segment Multiplier (SSM) was further proposed using a similar partition scheme. Different from ETM, no approximation is applied to the LSBs in the SSM. Either the MSBs or the LSBs

of each of the operands are accurately multiplied depending on whether its MSBs are all zeros. Shown that a small improvement in accuracy and hardware cost is achieved compared to the ETM, thus this design is not considered further in the comparison study. A power and area-efficient Approximate Wallace tree multiplier (AWTM) is based on a bit-width aware approximate multiplication and a carry-in prediction method. An n bit AWTM is implemented by four n/2-bit sub-multipliers, and the most significant n/2-bit sub-multiplier is further implemented by four n/4-bit sub-multipliers. The AWTM is configured into four different modes by the number of approximate n/4-bit sub-multipliers in the most significant n/2-bit sub-multiplier. The approximate partial products are then accumulated by a Wallace tree.

3.3 Using approximate counters or compressors in the partial product tree

In the Inaccurate Counter based Multiplier (ICM), an approximate (4:2) counter is proposed for an inaccurate 4-bit Wallace multiplier. The carry andsum of the counter are approximated as "10" (for "100") when all input signals are '1'. As the probability of obtaining a partial product of '1' is 1 4, the error rate of the approximate (4:2) counter is (1 4) 4 = 1 256. The inaccurate 4-bit multiplier is then used to construct larger multipliers with error detection and correction circuits. In the compressor based multiplier, accurate (3:2) and (4:2) compressors are improved to speed up the partial product accumulation stage. By using the improved compressors, better energy and delay characteristics are obtained for a multiplier. To further reduce delay and power, two approximate (4:2) compressor designs (AC1 and AC2); these compressors are used in a Dadda multiplier with four different schemes. Approximate counters in which the more significant output bits are ignored are presented and evaluated; several signed multipliers are also implemented using these approximate counters. As only unsigned multipliers are discussed in this paper, the more accurate schemes 3 and 4 of the approximate compressor based multiplier (referred to as ACM-3 and ACM-4) in are considered in the comparison. In the approximate multiplier with configurable error recovery, the partial products are accumulated by a novel approximate adder.

The approximate adder utilizes two adjacent inputs to generate a sum and an error bit. The adder processes data in parallel, thus no carry propagation is required. Two approximate error accumulation schemes are then proposed to alleviate the error of the approximate multiplier (due to the approximate adder). OR gates are used in the first error accumulation stage in scheme 1 (AM1), while in scheme 2 (AM2), both OR gates and approximate adders are used. The truncation of 16 LSBs in the partial products in AM1 and AM2 results in TAM1 and TAM2 respectively.

3.4 Proposed approximate multiplier

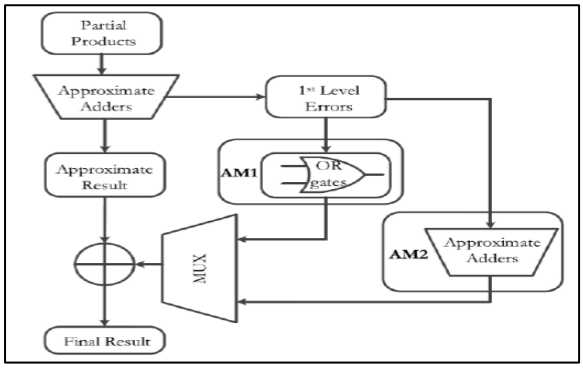
A distinguishing feature of the proposed approximate multiplier is the simplicity to use approximate adders in the partial product accumulation. It has been shown that this may lead to low accuracy, because errors may accumulate and it is difficult to correct errors using existing approximate adders. However, the use of the newly proposed approximate adder overcomes this problem by utilizing the error signal. The resulting design has a critical path delay that is shorter than a conventional one-bit full adder, because the new n-bit adder can process data in parallel. The approximate adder has a rather high error rate, but the feature of generating both the sum and error signals at the same time reduces errors in the final product. An adder tree is utilized for partial product accumulation; the error signals in the tree are then used to compensate the error in the output to generate a product with a better accuracy.

The architecture of the proposed approximate multiplier is shown in figure 1. In the proposed design, the simplification of the partial product accumulation stage is accomplished by using an adder tree, in which the number of partial products is reduced by a factor of 2 at each stage of the tree. This adder tree is usually not implemented using accurate multi-bit adders due to the long latency. However, the proposed approximate adder is suitable for implementing an adder tree, because it is less complex than a conventional adder and has a much shorter critical path delay.

Exact fast multipliers often include a Wallace or Dadda tree using full adders (FAs) and half adders (HAs); compressors are also utilized in the Wallace or Dadda tree to further reduce the critical path with an increase in circuit area. These designs require a proper selection of different circuit modules; for example, 4:2 compressors, FAs and HAs are commonly used in a Wallace tree and a judicious

connection of these modules must be considered in a tree design. This increases the design complexity, especially when multipliers of different sizes are considered; the proposed design is simple for various multiplier sizes.

Fig 1: An Approximate Multiplier with Partial Error Recovery



3.5 Error accumulation for approximate multiplier 1

As shown in Fig. 1, each approximate adder Ai generates a sum vector Si and an error vector Ei, where i = 1, 2,…. 7. If the error signals are added using accurate adders, the accumulated error can fully compensate the inaccurate product; however to reduce complexity, an approximate error accumulation is introduced. Consider the observation that the error vector of each approximate adder tends to have more 0's than 1's. Therefore, the probability that the error vectors have an error bit '1'at the same position, is quite small. Hence, an OR gate is used to approximately compute the sum of the errors for a single bit. If m error vectors (denoted by E1, E2, ...,Em) have to be accumulated, then the sum of these vectors is obtained as

Ei = E1i OR E2i OR ... OR Emi .

To reduce errors, an accumulated error vector is added to the adder tree output using a conventional CPA (e.g. a carry lookahead adder). However, only several (e.g. k) MSBs of the error signals are used to compensate the outputs to further reduce the overall complexity. The number of MSBs is selected according to the extent that errors must be compensated. Forexample in an 8 × 8 adder tree, there are a total of 7 error vectors, generated by the 7 approximate adders in the tree. However, not all the bits in the 7 vectors need to be added, because the MSBs of some vectors are less significant than the least significant bits of the k MSBs. In the example of Fig. 1, 5 MSBs (i.e. the (11 − 14)th bits, no error is generated at the 15th bit position) are considered for error recovery and therefore, 4 error vectors are considered (i.e., the error vectors E3, E4, E6 and E7). The error vectors of the other three adders are less significant than the 11th bit, so they are not considered. The accumulated error E is obtained using (8); then, the final result is found by adding E to S using a fast accurate CPA. The error accumulation scheme is shown in figure 2. As no error is generated at the least significant two bits of each approximate adder Ai (i = 1, 2…. 7), the least significant two bits of each error vector Ei are not accumulated.

Fig 2: Error Accumulation Tree for AM1

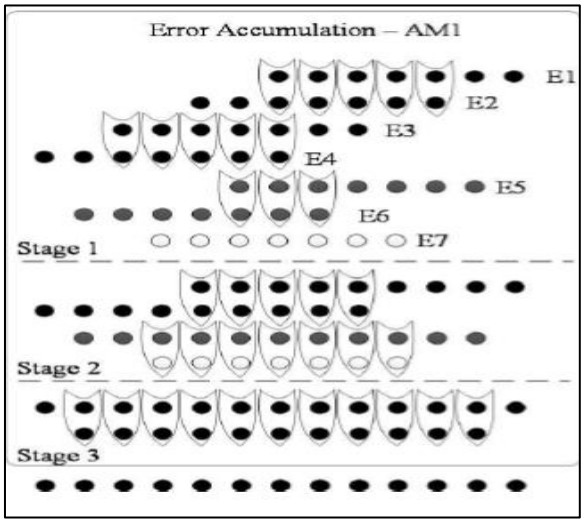


Fig 3: Approximate Multiplier Output

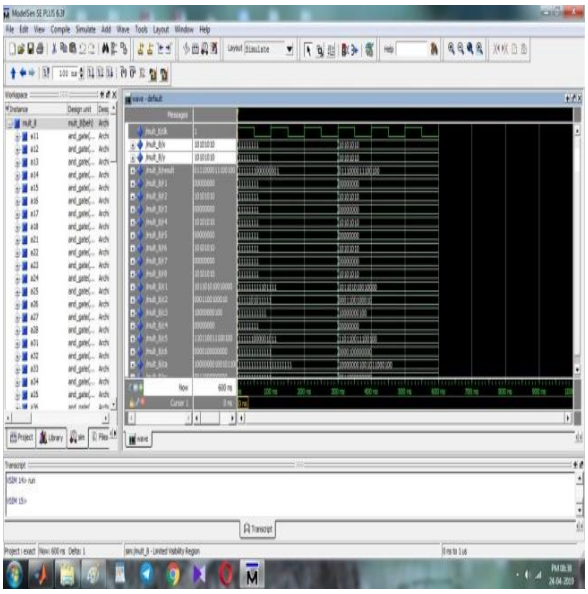
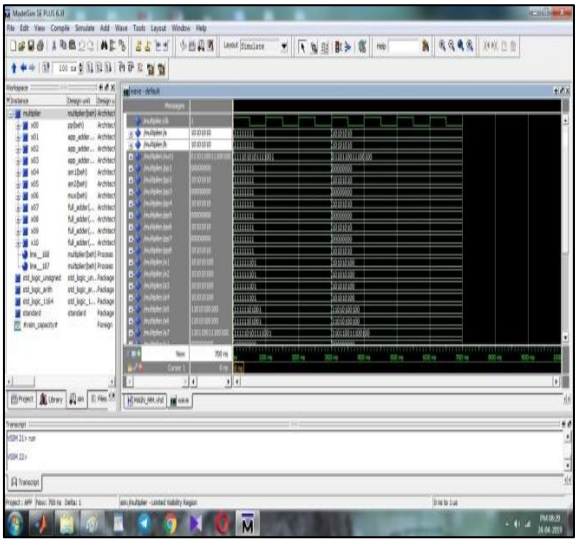


Fig 4: Proposed Approximate Multiplier Output



5.0 Results

Simulation results are shown by figure 3 and figure 4 respectively.

6.0 Conclusions

Approximate computing has recently emerged as a promising approach to energy-efficient design of digital systems. Approximate computing relies on the ability of many systems and applications to tolerate some loss of quality or optimality in the computed result. By relaxing the need for fully precise or completely deterministic operations, approximate computing techniques allow substantially improved energy efficiency.

References

[1] J Han, M Orshansky. Approximate computing: An emerging paradigm for energy-efficient design, in Proc. 18th IEEE Eur. Test Symp., 2013, 1–6.

[2] SL Lu. Speeding up processing with approximation circuits, Computer, 37(3), 2004, 67–73.

[3] AK Verma, P Brisk, P Ienne. Variable latency speculative addition: A new paradigm for arithmetic circuit design, in Proc. Design, Automat. Test Eur., 3, 2008, 1250–1255.

[4] N Zhu, WL Goh, KS Yeo. An enhanced low-power high-speed adder for error-tolerant application, in Proc. 12th Int. Symp. Integr. Circuits, 12, 2009, 69–72.

[5] HR Mahdiani, A Ahmadi, SM Fakhraie, C Lucas. Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications, IEEE Trans. Circuits Syst. I, Reg. Papers, 57(4), 2010, 850–862.

[6] V Gupta, D Mohapatra, SP Park, A Raghunathan, K Roy. Impact: IMPrecise adders for low-power approximate computing, in Proc. IEEE/ACM Int. Symp. Low Power Electron. Design, 8, 2011, 409–414.

[7] AB Kahng, S Kang. Accuracy-configurable adder for approximate arithmetic designs, in Proc. Design Automat. Conf., 6, 2012, 820–825.

[8] K Du, P Varman, K Mohanram. High performance reliable variable latency carry select addition, in Proc. Design, Automat. Test Eur. Conf. Exhib., 3, 2012, 1257–1262.

[9] J Liang, J Han, F Lombardi. New metrics for the reliability of approximate and probabilistic adders, IEEE Trans. Comput., 62(9), 2012, 1760–1771.

[10] J Huang, J Lach, G Robins. A methodology for energy-quality tradeoff using imprecise hardware, in Proc. Design Automat. Conf., 6, 2012, 504–509.

[11] J Miao, K He, A Gerstlauer, M Orshansky. Modeling and synthesis of quality-energy optimal approximate adders, in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design, 11, 2012, 728–735.

[12] R Venkatesan, A Agarwal, K Roy, A Raghunathan. Macaco: Modeling and analysis of circuits for approximate computing, Proc. IEEE/ACM Int. Conf. Comput.-Aided Design, 11, 2011, 667–673.

[13] H Jiang, C Liu, L Liu, F Lombardi, J Han. A review, classification, and comparative evaluation of approximate arithmetic circuits, ACM J Emerg. Technol. Comput. Syst., 13(4), 2017, Art. no. 60.

[14] P Kulkarni, P Gupta, MD Ercegovac. Trading accuracy for power in a multiplier architecture, J Low Power Electron., 7(4), 2011, 490–501.

[15] K Bhardwaj, PS Mane, J Henkel. Power- and area-efficient approximate wallace tree multiplier for error-resilient systems, in Proc. 15th Int. Symp. Qual. Electron. Design, 3, 2014, 263–269.

[16] KY Kyaw, WL Goh, KS Yeo. ow-power high-speed multiplier for error-tolerant application, Proc. IEEE Int. Conf. Electron Devices Solid-State Circuits, 12, 2010, 1–4.

[17] S Narayanamoorthy, HA Moghaddam, Z Liu, T Park, NS Kim. Energy-efficient approximate multiplication for digital signal processing and classification applications, IEEE Trans. Very Large Scale Integr. (VLSI) Syst., 23(6), 2015, 1180–1184.

[18] YH Chen, TY Chang. A high-accuracy adaptive conditionalprobability estimator for fixed-width booth multipliers, IEEE Trans. Circuits Syst. I, Reg. Papers, 59(3), 2012, 594–603,

[19] B Shao, P Li. Array-based approximate arithmetic computing: A general model and applications to multiplier and squarer design, IEEE Trans. Circuits Syst. I, Reg. Papers, 62(4), 2015, 1081–1090.

[20] H Jiang, J Han, F Qiao, F Lombardi. Approximate radix-8 booth multipliers for low-power and high-performance operation, IEEE Trans. Comput., 65(8), 2016, 2638–2644.

[21] K Nepal, Y Li, RI Bahar, S Reda. Abacus: A technique for automated behavioral synthesis of approximate computing circuits, in Proc. Design, Automat. Test Eur. Conf. Exhib., 3, 2014, 1–6.

[22] A Ranjan, A Raha, S Venkataramani, K Roy, A Raghunathan. Aslan: Synthesis of approximate sequential circuits, in Proc. Design, Automat. Test Eur. Conf. Exhib., 3, 2014, 1–6.

[23] C Liu, J Han, F Lombardi. A low-power, high-performance approximate multiplier with configurable partial error recovery, in Proc. Design, Automat. Test Eur. Conf. Exhib., 3, 2014, 1–4.

[24] B Parhami. Computer Arithmetic. London, U.K.: Oxford Univ. Press, 2000.

[25] MA Breuer. Intelligible test techniques to support error-tolerance, in Proc. 13th Asian Test Symp., 11, 2004, 386–393.

[26] N Weste, H David. CMOS VLSI Design: A Circuits and Systems Perspective, 3rd ed. London, U.K.: Pearson, 2005.

[27] VG Oklobdzija, D Villeger, SS Liu. A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach, IEEE Trans. Comput., 45(3), 1996, 294–306.

[28] KC Bickerstaff, EE Swartzlander, MJ Schulte. Analysis of column compression multipliers, in Proc. 15th IEEE Symp. Comput. Arithmetic, 6, 2001, 33–39.

[29] KC Bickerstaff, MJ Schulte, EE Swartzlander. Parallel reduced area multipliers, J. VLSI Signal Process. Syst. Signal, Image Video Technol., 9(3), 1995, 181–191.

[30] YK Cheng, CH Tsai, CC Teng, SM Kang. Electrothermal Analysis of VLSI Systems. New York, NY, USA: Springer, 2002.

[31] EJ King, EE Swartzlander. Data-dependent truncation scheme for parallel multipliers, Proc. 31st Conf. Rec. Asilomar Conf. Signals, Syst. Comput., 2(11), 1997, 1178–1182.

[32] MSK Lau, KV Ling, YC Chu. Energy-aware probabilistic multiplier: Design and analysis, in Proc. Int. Conf. Compil., Archit., Synthesis Embedded Syst., 2009, 281–290.

[33] HR Myler, AR Weeks. The Pocket Handbook of Image Processing Algorithms in C Englewood Cliffs, NJ, USA: Prentice-Hall, 1993.