International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 12
April 2018

Design and Implementation of Area Efficient Approximate Multipliers

Medepalli Narasimha Rao & Keerti kumar korlapati
[1]Asistant professor, [2]Associate professor,
[1,2]Dept. of ECE,
[1,] Kodada Institute of Technology and Science for Women, Kodada, Telangana
[2.] Vaageswari College of Engineering, Karimnagar, Telangana.

Abstract: *Approximate computing can decrease the design complexity with an increase in performance and power efficiency for error resilient applications. This brief deals with a new design approach for approximation of multipliers. The partial products of the multiplier are altered to introduce varying probability terms. Logic complexity of approximation is varied for the accumulation of altered partial products based on their probability. The proposed approximation is utilized in two variants of 16-bit multipliers.*

Key words: Approximate computing, error analysis, low error, low power, multipliers.

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.

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.

A multiplier is an important part of digital signal processing systems, like frequency domain filtering (FIR and IIR), frequency-time transformations (FFT), Correlation, Digital Image processing etc. Multipliers have large area, long latency and consume considerable power. While many previous works focused on implementing high-speed multipliers, recently there have been many attempts to reduce power consumption. This is due to the increased demand for portable multimedia applications, which require low power consumption as well as high speed.

Direct hardware implementations of shift and add multipliers can increase performance over software synthesis, but are still quite slow. The reason is that as each additional partial-product is summed a carry must be propagated from the least significant bit **(LSB)** to the most significant bit **(MSB).** This carry propagation is time consuming, and must be repeated for each partial product to be summed.

To achieve even higher performance advanced hardware multiplier architectures search for faster and more efficient methods for summing the partial-products. Most increase performance by eliminating the time consuming carry propagate additions. To accomplish this, they sum the partial-products in a redundant number representation. The advantage of a redundant representation is that two numbers, or partial-products, can be added together without propagating a carry across the entire width of the number. Many redundant number representations are possible. One commonly used representation is known as carry-save form. In this redundant representation two bits, known as the carry and sum, are used to represent each bit position. When two numbers in carry-save form are added together any carries that result are never propagated more than one bit position. This makes adding two numbers in carry-save form much faster than adding two normal binary numbers where a carry may propagate. One common method that has been developed for summing rows of partial products using a carry-save representation is the array multiplier.

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 × m multiplication is performed instead of n × 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.

II. LITERATURE SURVEY

Low power is an imperative requirement for portable multimedia devices employing various signal processing algorithms and architectures. In most multimedia applications, human beings can gather useful information from slightly erroneous outputs. Therefore, we do not need to produce exactly correct numerical outputs. Previous research in this context exploits error resiliency primarily through voltage over scaling, utilizing algorithmic and architectural techniques to mitigate the resulting errors.

The conventional digital hardware computational blocks with different structures are designed to compute the precise results of the assigned calculations. The main contribution of our proposed Bio-inspired Imprecise Computational blocks (BICs) is that they are designed to provide an applicable estimation of the result instead of its precise value at a lower cost.

These novel structures are more efficient in terms of area, speed, and power consumption with respect to their precise rivals. Complete descriptions of sample BIC adder and multiplier structures as well as their error behaviors and synthesis results are introduced in this paper. It is then shown that these BIC structures can be exploited to efficiently implement a three-layer face recognition neural network and the hardware de fuzzification block of a fuzzy processor.

We propose logic complexity reduction at the transistor level as an alternative approach to take advantage of the relaxation of numerical accuracy. We demonstrate this concept by proposing various imprecise or approximate full adder cells with reduced complexity at the transistor level, and utilize them to design approximate multi-bit adders.

In addition to the inherent reduction in switched capacitance, our techniques result in significantly shorter critical paths, enabling voltage scaling. We design architectures for video and image compression algorithms using the proposed approximate arithmetic units and evaluate them to demonstrate the efficacy of our approach. We also derive simple mathematical models for error and power consumption of these approximate adders. Furthermore, we demonstrate the utility of these approximate adders in two digital signal processing architectures (discrete cosine transform and finite impulse response filter) with specific quality constraints. Simulation results indicate up to 69% power savings using the proposed approximate adders, when compared to existing implementations using accurate adders.

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.

Most of the authors prefer truncation of partial products or the truncation of inputs for multiplication for approximation multipliers. Because it gives maximum performance among the previous works done on the survey of approximation. This type of approximation is mainly used in the digital image processing applications where accuracy is not a key component.

III. PROPOSED SYSTEM

In general implementation of multiplier comprises three steps: generation of partial products, partial products reduction tree, 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.

A 8-bit unsigned1 multiplier is used for illustration to describe the proposed method in approximation of multipliers. Consider two 8-bit unsigned input operands $\alpha = \sum_{m=0}^{7} \alpha_m 2^m$ and $\beta = \sum_{n=0}^{7} \beta_n 2^n$. The partial product $a_{m,n} = \alpha_m \cdot \beta_n$ in Fig. 1 is the result of AND operation between the bits of α_m and β_n. 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 propagate and generate signals form altered partial products $p_{m,n}$ and $g_{m,n}$. From column 3 with weight 2^3 to column 11 with weight 2^{11}, the partial products $a_{m,n}$ and $a_{n,m}$ 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.

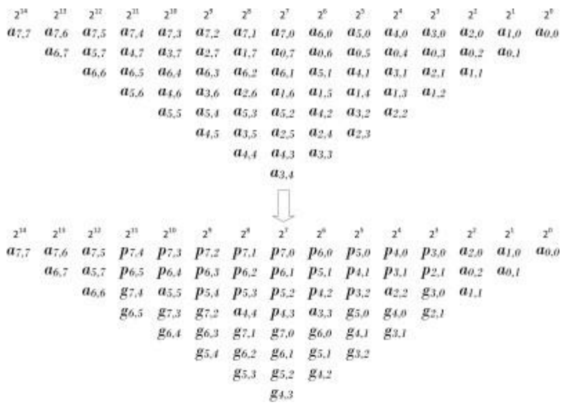


Fig 1 : Altered partial products generation from the normal partial products.

A. Approximation of Altered Partial Products gm,n

The accumulation of generate signals is done columnwise. As each element has a probability of 1/16 of being one, two elements being 1 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.

B. Approximation of Other Partial Products

The accumulation of other partial products with probability 1/4 for $a_{m,n}$ and 7/16 for $p_{m,n}$ uses approximate circuits. Approximate half-adder, full-adder, and 4-2 compressor are proposed for their accumulation. Carr y and Sum are two outputs of these approximate circuits. Since Carr y has higher weight of binary bit, error in Carr y 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 Carr y 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. This results in one error in the Sum computation as seen in the truth table of approximate half-adder in Table I. A tick mark denotes that approximate output matches with correct output and cross mark denotes mismatch.

$$\text{Sum} = x1+x2$$

$$\text{Carry} = x1.x2$$

TABLE I

TRUTH TABLE OF APPROXIMATE HALF ADDER

Inputs		Exact Outputs		Approximate Outputs		Absolute Difference
x1	x2	Carr y	Su m	carry	sum	
0	0	0	0	0	0	0
0	1	0	1	0	1	0
1	0	0	1	0	1	0
1	1	1	0	1	1	1

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. Carr y 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 II.

$$W = x1+x2$$

$$\text{Sum} = W \oplus x3$$

$$\text{Carry} = W . x3$$

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 12
April 2018

TABLE II

TRUTH TABLE OF APPROXIMATE FULL ADDER

Inputs			Exact Outputs		Approximate Outputs		Absolute Difference
x1	x2	x3	Carry	sum	carry	sum	
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	0	1	0	1	0
0	1	1	1	0	1	0	1
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 III

TRUTH TABLE OF APPROXIMATE 4-2 COMPRESSOR

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	0	1
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
1	1	1	1	1	1	1

$$W1 = x1.x2$$

$$W2 = x3.x4$$

$$Sum = (x1 \oplus x2)+(x3 \oplus x4)$$

$$Carry = W1 + W2$$

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 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. Carr y is simplified as in (4). The corresponding truth table is given in Table III.

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.

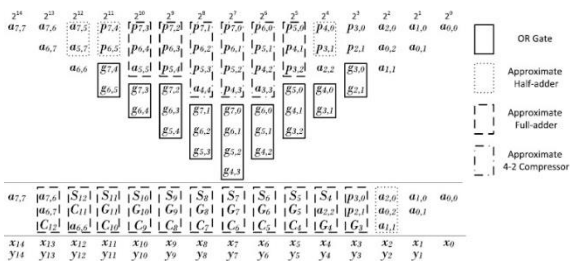


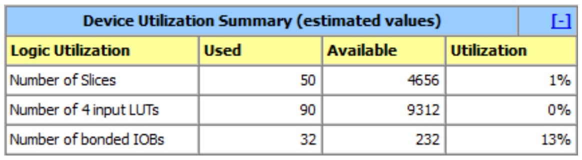
Fig 2 . Reduction of altered partial products.

The resultant signals of OR gates are labeled as Gi 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 Carr y 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.

IV. SIMULATION RESULTS

Area Report

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 12
April 2018



Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	50	4656	1%
Number of 4 input LUTs	90	9312	0%
Number of bonded IOBs	32	232	13%

Timing Report

Minimum period: No path found

Minimum input arrival time before clock:

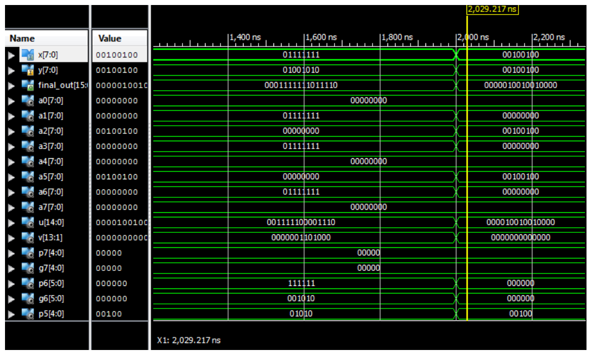
No path found

Maximum output required time after clock:

No path found

Maximum combinational path delay: 12.571ns

Simulation Results


2,029.217 ns
Name
Value
1,400 ns
1,600 ns
1,800 ns
2,000 ns
2,200 ns
x[7:0] 00100100 01111111 00100100
y[7:0] 00100100 01001010 00100100
final_out[15: 000001001 0001111111011110 0000010010010000
a0[7:0] 00000000 00000000
a1[7:0] 00000000 01111111 00000000
a2[7:0] 00100100 00000000 00100100
a3[7:0] 00000000 01111111 00000000
a4[7:0] 00000000 00000000
a5[7:0] 00100100 00000000 00100100
a6[7:0] 00000000 01111111 00000000
a7[7:0] 00000000 00000000
u[14:0] 000010010 001111100001110 000100100010000
v[13:1] 0000000000 0000001101000 0000000000000
p7[4:0] 00000 00000
g7[4:0] 00000 00000
p6[5:0] 000000 111111 000000
g6[5:0] 000000 001010 000000
p5[4:0] 00100 01010 00100
X1: 2,029.217 ns

V. CONCLUSION

In this we proposed approximation multiplier based on the partial product reduction tree. For that we have also proposed different approximte adders for partial product addition stage. In this brief, to propose efficient approximate multipliers, partial products of the multiplier are modified using generate and propagate signals. Approximation is applied using simple OR gate for altered generate partial products. Approximate half-adder, full-adder, and 4-2 compressor are proposed to reduce remaining partial products.

VI. REFERENCES

[1] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," IEEE Trans. Comput.- Aided Design Integr. Circuits Syst., vol. 32, no. 1, pp. 124–137, Jan. 2013.

[2] E. J. King and E. E. Swartzlander, Jr., "Data-dependent truncation scheme for parallel multipliers," in Proc. 31st Asilomar Conf. Signals, Circuits Syst., Nov. 1998, pp. 1178–1182.

[3] K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi, "Design of low-error fixed-width modified booth multiplier," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 12, no. 5, pp. 522–531, May 2004.

[4] 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 Trans. Circuits Syst. I,

Reg. Papers, vol. 57, no. 4, pp. 850–862, Apr. 2010.

[5] 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.

[6] 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.

[7] 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 Integr. (VLSI) Syst., vol. 24, no. 10, pp. 3105–3117, Oct. 2016.

[8] P. Kulkarni, P. Gupta, and M. D. Ercegovac, “Trading accuracy for power in a multiplier architecture,” J. Low Power Electron., vol. 7, no. 4, pp. 490–501, 2011.

[9] C.-H. Lin and C. Lin, “High accuracy approximate multiplier with error correction,” in Proc. IEEE 31st Int. Conf. Comput. Design, Sep. 2013, pp. 33–38.

[10] 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.

[11] 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.

[12] J. Liang, J. Han, and F. Lombardi, “New metrics for the reliability of approximate and probabilistic adders,” IEEE Trans. Comput., vol. 63, no. 9, pp. 1760–1771, Sep. 2013.

[13] S. Suman et al., “Image enhancement using geometric mean filter and gamma correction for WCE iamges,” in Proc. 21st Int. Conf., Neural Inf. Process. (ICONIP), 2014, pp. 276–283