2581-4575

International Journal For Recent Developments in Science & Technology

Crossref

A Peer Reviewed Research Journal

DESIGN OF AN EFFICIENT ROUNDING TECHNIQUE BASED APPROXIMATE MULTIPLIER FOR ENERGY EFFICIENT

D.INDRASENA REDDY, Y.BASAVARAJU

PG SCHOLAR.DEPT OF ECE, SRI KRISHNADEVARAYA ENGINEERING COLLEGE, GOOTY, AP, INDIA

ASSISTANT PROFESSOR, DEPT OF ECE, SRI KRISHNADEVARAYA ENGINEERING COLLEGE, GOOTY, AP,INDIA

Abstract

Approximate computing is one of best suited efficient data processing for error resilient applications, such as signal and image processing, computer vision, machine learning, data mining etc. Approximate computing reduces accuracy which is acceptable as a cost of increasing the circuit characteristics depends on the application. Desirable accuracy is the threshold point for controlling the trade off, between accuracy and circuit characteristics under the control of the circuit designer. In this work, the rounding technique is introduced as an efficient method for controlling this trade off. In this regard multiplier circuits as a critical building block for computing in most of the processors have been considered for the evaluation of the rounding technique efficiency. The impact of the rounding method is investigated by comparison of circuit characteristics for multipliers.

1.INTRODUCTION

ENERGY minimization is one of the main design requirements in almost any electronic systems, especially the portable ones such as smart phones, tablets, and different gadgets. It is highly desired to achieve this minimization with minimal performance (speed) penalty. Digital signal processing (DSP) blocks are key components of these portable devices for realizing various multimedia applications. The computational core of these blocks is the arithmetic logic unit where multiplications have the greatest share among all arithmetic operations performed in these DSP systems. Therefore, improving the speed and power/energy-efficiency characteristics of multipliers plays a key role in improving the efficiency of processors. Many of the DSP cores implement image and video processing algorithms where final outputs are either images or videos prepared for human consumptions. This fact enables us to use approximations for improving the speed/energy efficiency. This originates from the limited perceptual abilities of human beings in observing an image or a video. In addition to the image and video processing applications, there are other areas where the exactness of the arithmetic operations is not critical to the functionality of the system. Being able to use the approximate computing provides the designer with the ability of making tradeoffs between the accuracy and the speed as well as power/energy consumption.

Applying the approximation to the arithmetic units can be performed at different design abstraction levels including circuit, logic, and architecture levels, as well as algorithm and software layers. The approximation may be performed using different techniques such as allowing some timing violations (e.g., voltage over scaling or over clocking) and function approximation methods (e.g., modifying the Boolean function of a circuit) or a combination of them. In the category of function approximation methods, a number of approximating arithmetic building blocks, such as adders and multipliers, at different design levels have been suggested. We focus on proposing a high-speed low power/energy yet approximate multiplier appropriate for error resilient DSP applications. The proposed approximate multiplier, which is also area efficient, is constructed by modifying the conventional multiplication approach at the algorithm level assuming rounded input values.

2.Approximate multiplier

We are at the threshold of an explosion in new data, produced not only by large, powerful scientific and commercial computers, but also by the billions of low-power devices of various kinds. While traditional workloads including transactional and database processing continue to grow modestly, there is an explosion in the computational footprint of a range of applications that aim to extract deep insight from vast quantities of structured and unstructured data. There is an exactness implied by traditional computing that is not needed in the processing of most types of these data. Yet today, these cognitive applications continue to be executed on general purpose (and accelerator) platforms that are highly precise and designed with reliability from the ground up. Approximate computing aims to relax these constraints with the goal of obtaining significant gains in computational throughput - while still maintaining an acceptable quality of results.

A primary goal of research in approximate computing is to determine what degrees of approximations in the several layers of the system stack (from algorithms down to circuits and semi-conductor devices) are feasible so that the produced results are acceptable, albeit possibly different from those obtained using precise computation. Approximate computing techniques studied by various researchers have focused primarily on optimizing one layer of the system stack and have shown benefits in power or execution time. In this work we set out to investigate if combining multiple approximation techniques spanning more than one layer of the system stack compounded the benefits, and if these compounded benefits are widely applicable across different application domains.

In order to provide a concrete demonstration, we focused on three approximation categories: skipping computations, approximation of arithmetic computations themselves, and approximation of communication between computational elements. As representatives of each category we evaluated loop perforation, reduced arithmetic precision, and relaxation of synchronization. We selected applications that are computationally expensive but have the potential to significantly impact our lives if they became cheap and pervasive. Our applications spanned the domains of digital signal processing, robotics, and machine learning. Across the set of applications studied, our results show that we were able to perforate hot loops in the studied applications by an average of 50%, with proportional reduction in overall execution time, while still producing acceptable quality of results. In addition, we were able to reduce the width of the data used in the computation to 10-16 bits from the currently common 32 or even 64 bits, with potential for significant performance and energy benefits. In the parallel applications we studied, we were able to reduce execution time by 50% through partial elimination of synchronization overheads.

Finally, our results also demonstrate that the benefits from these techniques are compounded when applied concurrently. That is, combined judiciously, the multiple techniques do not significantly lessen the effectiveness of one another. As the benefits of approximate computing are not restricted to a small class of applications these results motivate a re-thinking of the general purpose processor architecture to natively support different kinds of approximation to better realize the potential to approximate computing.

Rounding technique is one of the most efficient methods for packing the input data before processing. This method has a potential to improve the circuit characteristics such as power and energy consumption, speed and area which is suitable method for the approximate computing. Approximate computing works very well to most of error resilient applications in the field of computer vision, image processing, pattern recognition, signal processing, scientific computing, and machine learning. Over past decade, research on these areas has given lots of opportunities in research. A multiplier is a fundamental block of computation and one of the most resource-consuming operation. We see innumerable research on this front with a significant tradeoff on accuracy and power-delay-energy. Fundamental building blocks of the multiplier are partial product generation, partial product reduction, and packing. This paper proposes rounding technique as a new method for input block prior to partial product generation. Accuracy curve as a criteria plays a critical role in controlling and minimizing the error range to be considerable depending on applications. Different algorithms are implemented on different levels of multiplier blocks. Input block has a rounding technique for both 16bit and 32bit based on accuracy levels. Generated partial products are divided into either active or inactive partial products. Inactive partial products are all zeros and hence are not needed to be considered in the reduction process with compressors.

3. PROPOSED DESIGN OF APPROXIMATE MULTIPLIER

The main idea behind the proposed approximate multiplier is to make use of rounded input for multiplication. Proposed algorithm applies a rounding technique before passing the data to the partial product generation. Fig. 1 shows the design chart for realization of the proposed method for the approximate multiplier design. Among the two inputs (Multiplicand and Multiplier), the Multiplier is rounded first by passing through rounding block. Before the multiplication operation starts, the sign bit of both inputs is stored, and the output sign of the multiplication result based on the inputs signs are determined. At the last stage, the proper sign is applied to the result. In an event of multiplying negative numbers, the respective input blocks are converted into their 2’s complement.

In conventional multipliers, with N-bit input, N × N partial products (partial products) are generated. But in the rounding technique, the partial products generated are the combination of active and inactive partial products. The active partial products are, that have "1" as the coefficient on the Multiplier. After rounding, it causes a complete row of Multiplicand as the result. Therefore, inactive partial products are the lines with whole 0’s. Therefore, has no necessity to cover them in the reduction process.

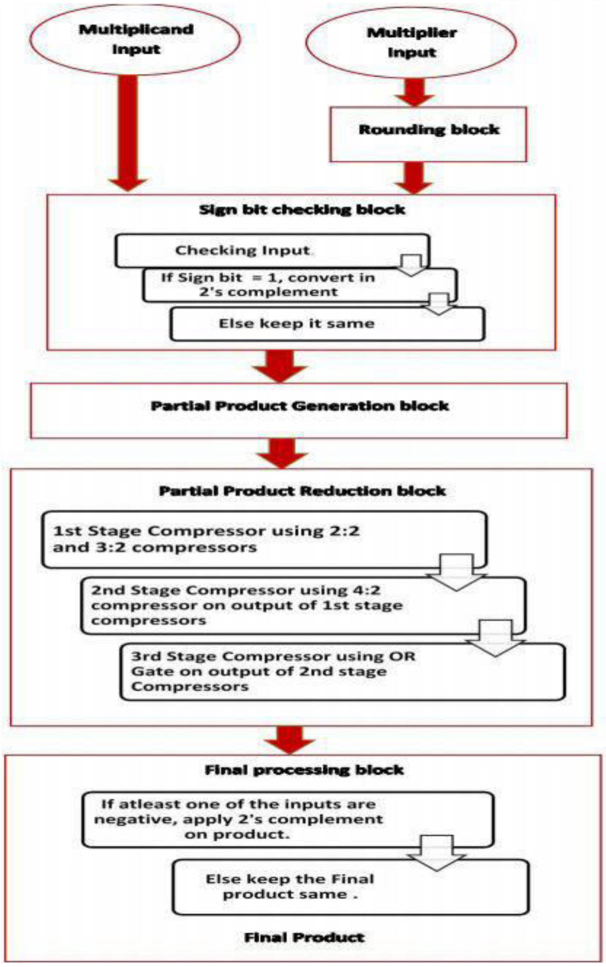
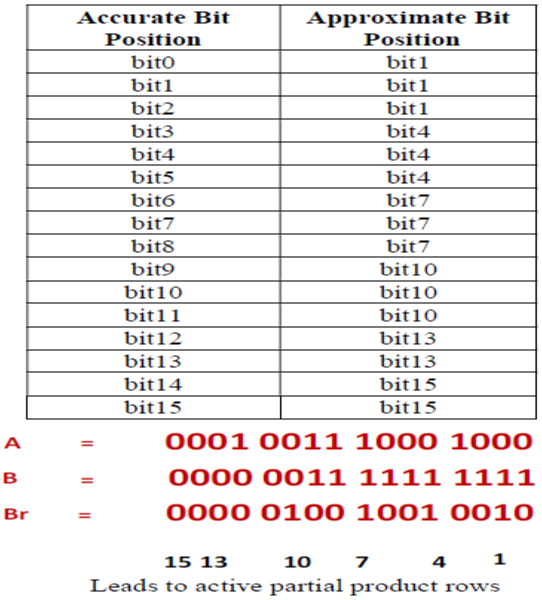

Multiplicand Input
Multiplier Input
Rounding block
Sign bit checking block
Checking Input
If Sign bit = 1, convert in 2's complement
Else keep it same
Partial Product Generation block
Partial Product Reduction block
1st Stage Compressor using 2:2 and 3:2 compressors
2nd Stage Compressor using 4:2 compressor on output of 1st stage compressors
3rd Stage Compressor using OR Gate on output of 2nd stage Compressors
Final processing block
If atleast one of the inputs are negative, apply 2's complement on product.
Else keep the Final product same .
Final Product

Fig. 1: 16-bit Block diagram of the algorithm

Rounding input data requires major responsibility in maintaining the accuracy. With a basic intuition, it can be stated that, rounding lower bits results in less error compared to rounding higher bits. Thus, the proposed algorithm has assigned rounding weights with respect to the bit position value. There is a small error gap between accurate bit position and rounded bit position. For every accurate bit, there is a corresponding rounded bit value assigned. Error gap reduces as the bit position value increases. Fig. 2 gives an example where ‘A’ and ‘B’ are inputs and input ‘B’ is rounded to get ‘Br’. "Rounding Technique" basically checks for a ‘1’ in the ‘X’ bit position and assigns ‘1’ to the respective ‘Y’ bit position with or without a small error.



Accurate Bit Position	Approximate Bit Position
bit0	bit1
bit1	bit1
bit2	bit1
bit3	bit4
bit4	bit4
bit5	bit4
bit6	bit7
bit7	bit7
bit8	bit7
bit9	bit10
bit10	bit10
bit11	bit10
bit12	bit13
bit13	bit13
bit14	bit15
bit15	bit15

A = 0001 0011 1000 1000
B = 0000 0011 1111 1111
Br = 0000 0100 1001 0010

15 13 10 7 4 1
Leads to active partial product rows

Fig. 2: An example after rounding ‘B’ (16-bit)

Partial products reduction is a stage where partial products are compressed using different kind of compressors. Proposed algorithm gives a flexibility on reducing number of partial

products rows. For an instance 16-bit design shown in Fig. 3 reduces partial products to 6 rows which is identified as active partial products. Like all traditional way, N-bit inputs are multiplied to generate N × N Partial products. In terms of computation complexity, as the number of bits increases, the length of Partial products increases with $O(N^2)$. Proposed algorithm provides computation complexity $\leq O(N\times6)$ for 16-bit and $\leq O(N\times13)$ for 32bit. For better understanding, along with design, an example is explained with input values A, B and Br (from fig. 6(right)). Multiplier input ‘B’ is first rounded to

‘Br’. Inputs are then multiplied to get N×N partial products. Due to rounding of multiplier input, N×N partial products is a combination of active and inactive partial products. Multiplier with ‘1’ as coefficient, after rounding, causes a complete row of Multiplicand as a result as illustrated in fig. 3. Therefore, inactive partial products are the whole zero values line which had "0" as the coefficient on the Multiplier. Thus, there is no need to cover them in the reduction process. In fact, inactive partial products could only increase hardware. This has led us to first eliminate, all inactive partial products before packing. This approach plays important role in reducing power, area and time usage and in turn increasing its efficiency. The active partial products are compressed and packed using three stages of compression. In the 1st stage, partial products are compressed using full adders and half adders. An output of 1st stage compression is further compressed using a 4:2 compressor when inputs are 16bit whereas for 32bit, 9:2 compressor. Fig. 3 (right) illustrates corresponding operations on an example. An output of the 2nd stage compressor is finally packed using OR gate to get a final product. Conventionally full adders are used instead of OR. The very idea of using OR gate instead of full adder is to reduce area and energy usage noticeably.

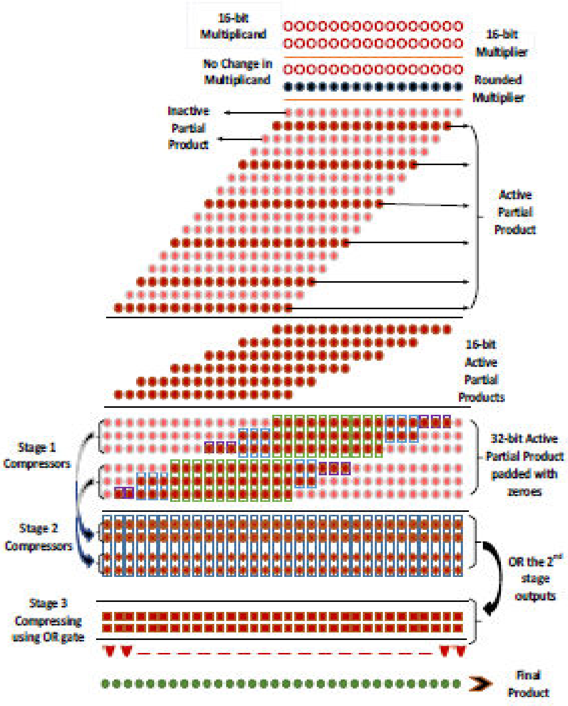


Fig. 3: Multiplier Design (left) with an example (right) (16-bit).

4. ROUNDING ERROR ANALYSIS

As seen in previous sections, only one Input (Multiplier) is rounded. As an instance let us consider 16-bit multiplier to analyze rounding error. From given inputs Multiplicand and

Multiplier, Multiplier input is rounded. For 16-bit values ranging from 0 to 65535, a set of rounded values are obtained using rounding algorithm. The step design obtained, shows how close the rounded values are with each other. Most of the places, step sizes are small which infers as lesser error and hence more accuracy. Only in one case, we see higher step size which may incur slightly higher error. This emphasizes us efficient data processing is done before the multiplication to mainly work on this area with a few additional hardware to obtain better accuracy. Thus, with this thorough analysis, a process. We then calculated the probability of occurrence of rounded values for numbers ranging from 0 to 65535. Red lines in fig. 4, shows very good analogy on how rounded values are distributed with its probabilities. Area until 9360 rounded values have lower probability which gives higher accuracy. Also, area from rounded value 40960 onwards also have lesser probability. Area at the center, with higher probability gives an opportunity to change rounding pattern to get better accuracy. Further studies and research will be mainly focusing on these areas to optimize algorithm with the expense of extra hardware.

5.Conclusion

Proposed algorithm proves to be best in terms of power-area delay and PDP efficiency when compared to other algorithms for both signed and unsigned data (16-bit and 32-bit). This is the primary investigation of rounding technique on approximate multiplier by having one method of rounding pattern which are fixed active partial product rows. With this rounding pattern, we see potential areas of less accuracy and areas with better accuracy corresponding to probability of rounding value. Based on accuracy required, rounding patterns are changed with a little extra expense of hardware. Rounding pattern can be modified to have fixed or dynamic partial product rows and yet have fewer active partial product rows for compression. The proposed algorithm can be used in the wide range of applications in image processing, machine learning and signal processing. Thus, different weights based on the bit position of ‘1’ plays an important role to keep accuracy relatively near to the conventional method. With flexible reduction of partial products, proposed algorithm produces great hardware characteristics, when compared to DRUM.

6.Future scope

This multipliers plays a very important role in our day to day life. In future the multipliers are going to play a major role. The speed of the multipliers are increased by using carry save adders, carry look ahead adder, and so on. Rounding patterns will be optimized based on required accuracy and different compression techniques. The area and delay can be reduced in future by using advanced technology.

7.BIBLIOGRAPHY

[1] M. Alioto, “Ultra-low power VLSI circuit design demystified and explained: A tutorial,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 59, no. 1, pp. 3–29, Jan. 2012.

[2] 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.

[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 Trans. Circuits Syst. I, Reg. Papers, vol. 57, no. 4, pp. 850–862, Apr. 2010.

[4] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, “MACACO: Modeling and analysis of circuits for approximate computing,” in Proc. Int. Conf. Comput.-Aided Design, Nov. 2011, pp. 667–673.

[5] F. Farshchi, M. S. Abrishami, and S. M. Fakhraie, “New approximate multiplier for low power digital signal processing,” in Proc. 17th Int. Symp. Comput. Archit. Digit. Syst. (CADS), Oct. 2013, pp. 25–30.

[6] P. Kulkarni, P. Gupta, and M. Ercegovac, “Trading accuracy for power with an underdesigned multiplier architecture,” in Proc. 24th Int. Conf. VLSI Design, Jan. 2011, pp. 346–351.

[7] D. R. Kelly, B. J. Phillips, and S. Al-Sarawi, “Approximate signed binary integer multipliers for arithmetic data value speculation,” in Proc. Conf. Design Archit. Signal Image Process., 2009, pp. 97–104.

[8] K. Y. Kyaw, W. L. Goh, and K. S. Yeo, “Low-power high-speed multiplier for error-tolerant application,” in Proc. IEEE Int. Conf. Electron Devices Solid-State Circuits (EDSSC), Dec. 2010, pp. 1–4.

[9] 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.

[10] K. Bhardwaj and P. S. Mane, “ACMA: Accuracy-configurable multiplier architecture for error-resilient system-on-chip,” in Proc. 8th Int. Workshop Reconfigurable Commun.-Centric Syst.-Chip, 2013, pp. 1–6.

[11] K. Bhardwaj, P. S. Mane, and J. Henkel, “Power- and area-efficient approximate wallace tree multiplier for error-resilient systems,” in Proc. 15th Int. Symp. Quality Electron. Design (ISQED), 2014, pp. 263–269.

[12] J. N. Mitchell, “Computer multiplication and division using binary logarithms,” IRE Trans. Electron. Comput., vol. EC-11, no. 4, pp. 512–517, Aug. 1962.

[13] V. Mahalingam and N. Ranganathan, “Improving accuracy in Mitchell’s logarithmic multiplication using operand decomposition,” IEEE Trans. Comput., vol. 55, no. 12, pp. 1523–1535, Dec. 2006.

[14] Nangate 45nm Open Cell Library, accessed on 2010. [Online]. Available: http://www.nangate.com/

[15] H. R. Myler and A. R. Weeks, The Pocket Handbook of Image Processing Algorithms in C. Englewood Cliffs, NJ, USA: Prentice-Hall, 2009.