ISSN NO: 9726-001X

Volume 10 Issue 01 JAN 2022

JOURNAL OF CURRENT SCIENCE

The Development of an Efficient Approximate Multiplier Using Rounding Techniques

GAJULAPALLE SIVANJANEYA REDDY[1] , P RAJYALAKSHMI[2]
Assistant professor[1,2]
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
P.B.R.VISVODAYA INSTITUTE OF TECHNOLOGY & SCIENCE
S.P.S.R NELLORE DIST, A.P , INDIA , KAVALI-524201

Abstract:

Approximate computing is one of best suited efficient data processing for error resilient applications, such as Technologies including signal and image processing, computer vision, machine learning, data mining, etc. The degree to which a drop in precision caused by approximative computation is tolerated in exchange for a boost in circuit characteristics is application specific. The circuit designer has some leeway in determining the balance between accuracy and other, more malleable, circuit features. The rounding approach is provided as a practical tool for managing this trade-off in this study. In this context, multiplier circuits, a fundamental component of computing in the vast majority of processors, have been taken into account in order to assess the efficacy of the rounding approach. By comparing the performance of several multipliers' circuits, we may examine the effects of different rounding strategies.

INTRODUCTION

Minimizing power consumption is a major design goal for many modern electronic devices, particularly mobile ones like smartphones and tablets. This minimization should be accomplished with as little loss of performance (speed) as possible. Key components of these mobile devices for achieving a wide range of multimedia applications are digital signal processing (DSP) blocks. The arithmetic logic unit is the heart of these blocks' calculation, and in digital signal processing (DSP), multiplications make up the bulk of arithmetic operations. It follows that enhancing the multiplier's speed and power/energy efficiency qualities is crucial for enhancing the performance of CPUs. The end result of the algorithms implemented by many DSP cores is often a picture or video that is suitable for human consumption. As a result, we may employ approximations to boost speed/energy efficiency. That all stems from the fact that people have restricted visual processing capabilities. There are a variety of uses for precise mathematics outside the realm of image and video processing.

functioning is not essential to the operation of the system. With approximation computing, the designer is not limited to choose between speed and accuracy or power and energy efficiency. The approximation may be applied to the arithmetic units at any of the design abstraction levels, including the circuit, logic, architecture, algorithm, and software layers. Allowing certain timing violations (like voltage over scaling or over clocking) and function approximation approaches (like altering the Boolean function of a circuit) are

two examples of strategies that might be used to achieve the approximation. Approximating arithmetic building blocks like adders and multipliers have been proposed at a variety of design levels under the domain of function approximation approaches. We propose a fast, low-power, approximation multiplier well-suited to error-tolerant DSP applications. By adapting the standard multiplication strategy on the level of the algorithm and supposing rounded input values, the suggested approximation multiplier is built, making it both area efficient and accurate.

Nearest Multiple

Large, sophisticated scientific and commercial computers, as well as the billions of low power gadgets of many sorts, have brought us to the precipice of an explosion of fresh data. Yet, the computing footprint of a variety of applications that seek to extract deep insight from enormous volumes of structured and unstructured data has exploded, while conventional workloads such as transactional and database processing continue to rise moderately. Traditional computing assumes an accuracy that is unnecessary for processing most of these data kinds. And yet, even in the present day, these cognitive applications are still often run-on general purpose (and accelerator) platforms that are both extremely accurate and built with dependability in mind from the bottom up. By loosening these restrictions, approximate computing seeks to increase computational throughput while keeping the quality of its output within acceptable bounds. The fundamental focus of approximate computing studies is to learn how much of a reduction in precision may be made at each level of the system stack (from

algorithms all the way down to circuits and semi-conductor devices) while still yielding usable results. Researchers have researched approximate computing approaches, with much of their attention being directed on improving a single layer of the system stack, with positive results in terms of reduced power consumption or reduced execution times. In this study, we set out to determine whether or not the advantages of using several approximation approaches from across more than one tier of the system stack were cumulative, and whether or not this benefit was transferable to other contexts. In an example, we focused on three types of approximation: those that involve skipping calculations, those that approximate arithmetic computations, and those that approximate the communication between computational units. Perforating a loop, reducing numerical accuracy, and delaying synchronisation were selected as examples to examine. We chose computationally intensive applications that may have far-reaching consequences if they were made accessible and affordable. The scope of our implementations included DSP, robotics, and ML. Our findings indicate that across the board, we were able to perforate hot loops in the examined applications by an average of 50%, resulting in a corresponding decrease in total execution time and acceptable quality of output. There may be substantial speed and energy savings by reducing the breadth of the data utilised in the calculation from the standard 32 or even 64 bits to 10 or 16 bits.

We found that eliminating certain synchronisation overheads in parallel apps lowered their execution times in half. Lastly, our findings show that the advantages of using these methods together are amplified. That is, when used together wisely, the various approaches do not dramatically reduce the efficacy of each other. These findings encourage a rethinking of the general-purpose processor design to natively enable various forms of approximation in order to fully exploit the potential of approximate computing, since its advantages are not limited to a select group of applications. One of the most effective approaches to input data packing is rounding. This approach is a good fit for approximation computing and has the potential to enhance circuit parameters like power and energy consumption, speed, and area.

Most error-resistant applications in the fields of computer vision, image processing, pattern recognition, signal processing, scientific computing, and machine learning benefit greatly from approximate computing. Throughout the last decade, studies in these fields have opened up several avenues for further study. One of the most time- and energy-intensive computational building blocks is the multiplier. Many studies have been conducted in this area, all of which involve some compromise between precision and power-delay-energy. Multiplier components that are essential to its operation include: production of partial products, reduction of partial products, and packing. For generating a partial product from an input block, this study suggests using the rounding approach as an alternative way. The accuracy curve is a crucial criterion for managing and reducing the potentially large error range that might arise in various contexts. Each level of the multiplier uses a different algorithm. Depending on the required degree of precision, either a 16-bit or 32-bit input block may use a rounding method. The generated products are classified as either active or inactive. For the purpose of compression, it is not necessary to take into account inactive partial products, which all have values of zero.

APPROXIMATE MULTIPLIER PLANNED DESIGN

The suggested approximation multiplier relies on input that has been rounded off in order to do the multiplication. The information is rounded by the proposed method before being sent on to the production of the partial product. To see the implementation of the suggested strategy for the approximation multiplier design, see Fig. 1. The Multiplicand is not rounded until after the Multiplier has been rounded using a rounding block. The sign bit of each input is first recorded, and the sign of the multiplication result is calculated in advance according to the signs of the inputs. Last, the correct sign is assigned to the total. Each input block is changed into its 2's complement form if a negative integer is being multiplied. Regular multipliers take in data in bits, and produce N partial products for every N input bit. On the other hand, the rounding method results in a mixture of active and inactive partial products. The Multiplier partial products with a coefficient of "1" are the ones that are really being used. This causes a full row of Multiplicand to be produced after rounding. Hence, lines consisting entirely of zeros denote inert partial products. as a result, there is no need to include them in the reduction process.

ISSN NO: 9726-001X

Volume 10 Issue 01 JAN 2022

JOURNAL OF CURRENT SCIENCE

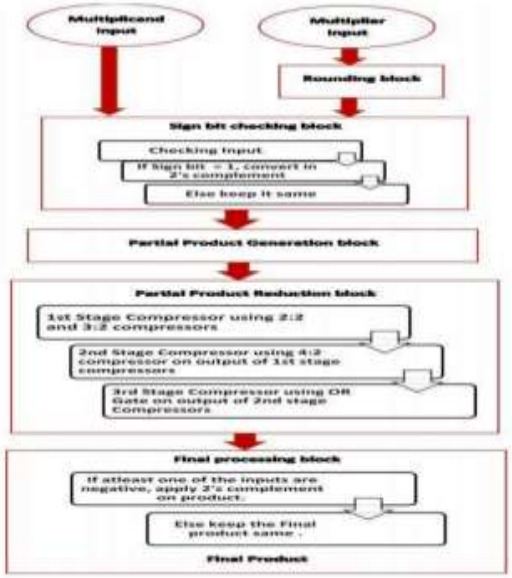

Fig. 1: 16-bit Block diagram of the

Algorithm

Input data rounding demands a great deal of caution to prevent inaccuracies. It's easy to see why rounding to fewer significant digits would produce less of an error than rounding to more significant digits. As a result, the suggested method incorporates rounding weights depending on the value of the corresponding bit position. The difference between a perfectly exact bit position and a rounded bit position is negligible. Each precise bit is paired with its rounded-off equivalent. When the value of the bit position grows, the error gap decreases. The inputs "A" and "B" in Figure 2 are combined and rounded to get the output "Br." To put it simply, the "Rounding Method" looks for a "1" in the "X" bit position and writes "1" into the "Y" bit position, with or without a little mistake.

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)

At the step known as "reduction of partial products," various compressors are used to further compress the partial products. The proposed technique provides leeway in the matter of minimising the rows containing incomplete product data. Example: the 16-bit architecture in Fig. 3 condenses partial products to only six rows,

ISSN NO: 9726-001X

Volume 10 Issue 01 JAN 2022

JOURNAL OF CURRENT SCIENCE

which are thus called active partial products. As with any conventional method, multiplying N-bit inputs yields N-by-N partial products. Complexity-wise, Partial products get longer with O(N2) as the number of bits grows. The proposed approach has a computational cost of at most O(N6) for 16-bit and O(N13) for 32-bit numbers. For clarity, we'll go through an example using the A, B, and Br values shown in fig. 6(right) as inputs to illustrate the concept and explain how it works. The "B" multiplier input is rounded up to the "Br" value first. To obtain NN partial products, we multiply our inputs by themselves. Since the input to the multiplier was rounded off, the NN partial products are a mix of active and inactive partial products. After rounding, a multiplier with a "1" coefficient produces a full row of Multiplicand, as seen in Fig. 3. As a result, the whole line of zero values where the Multiplier coefficient was "0" is considered to be a line of inactive partial products. Hence, protecting them throughout the reducing process is unnecessary. Just the opposite, hardware performance might be improved by using inert partial products. As a result, when we pack, we first ensure that there are no inert components left. This method significantly contributes to enhancing efficiency by decreasing power consumption, footprint, and operation time. There are three steps of compression used to pack the active partial products. Whole adders and half adders are used to compress the partial products in the first step. When the inputs are 16 bits, a 4:2 compressor is used to further compress the data, but when the inputs are 32 bits, a 9:2 compressor is used.

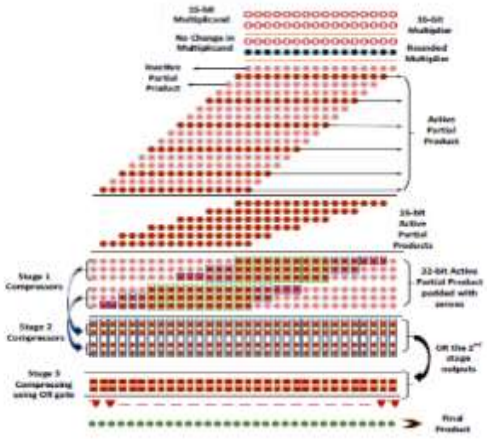

Fig. 3 (right) illustrates corresponding operations on an example.

The OR gate is used to compact the output of the second stage compressor. Usually, complete adders are used instead of OR. By replacing a complete adder with an OR gate, significant space and power savings may be achieved. Figure 3: Multiplier Example (right) and Design (left) (16-bit).

Adjusted Approximate Multiplier Structure

Using rounded input for multiplication is the core notion underlying the tweaked approximation multiplier. The updated procedure first rounds the data before sending it on to the computation of the partial product. The Multiplicand is not rounded until after the Multiplier has been rounded using a rounding block. The sign bit of each input is first recorded, and the sign of the multiplication result is calculated in advance according to the signs of the inputs. Last, the correct sign is assigned to the total. The updated multiplier's output is the absolute value of the partial products, so they may be added together using a single 2n-bit Parallel Prefix adder.

RESULT

Fig.4 Output wave form

Fig 4 show the simulation result of proposed multiplier a, b, calk, rest as the input and p as the output

COMPRESSION TABLE

	EXISTING ADDER	PROPOSED ADDER
DELAY	32.681ns	21.389ns
AREA	143	147
POWER	0.304w	0.186w
SPEED	30.591Mhz	46.753Mhz

CONCLUSION

Proposedmethods for both signed and unsigned data, and this one appears to be the most effective in terms of power-area latency and PDP efficiency (16-bit and 32-bit). This is the first study to focus just on a single rounding pattern, namely, fixed active partial product rows, while examining rounding techniques for an approximation multiplier. This rounding distribution shows us where the likelihood of rounding value is high and where it is low. With a little increase in hardware cost, rounding patterns may be adjusted to meet various degrees of precision. Compression may be improved by using a smaller set of active partial product rows, regardless of whether the rounding pattern is constant or dynamic. Many images processing, machine learning, and signal processing applications exist for the suggested approach. To maintain a level of precision comparable to that of the traditional approach, it is necessary to assign various weights depending on the bit location of "1." In comparison to DRUM, the suggested method yields superior hardware characteristics thanks to its flexible reduction of partial products. The improved multiplier, which achieved its high precision by the use of 2n-rounding on the inputs. As a result, time and energy efficiency were improved at the expense of a little inaccuracy by skipping the computation-intensive portion of the multiplication.

References

[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-powerdigital 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. AlSarawi, "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 errortolerant 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-onchip," 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.