A Novel Rounding and Truncation Based Approximate Multiplier for Reducing the Number of Partial Products

M.Sudheer Kumar Reddy
Electronics & Communication Engineering
JNTU Kakinada, India

E. Venkata Narayana
Assistant Professor
Dept. of Electronics & Communication Engineering
JNTU Kakinada, India

Abstract— **In this paper, we present an approximate multiplier in which operands are scaled with truncation and rounding parameters and multiplication is performed by shift, add, and small fixed-width multiplication operations ensuring large improvements in the energy consumption and area occupation related to those of the original multiplier approach. The scalability depends on the truncation and rounding parameters which can be varied as per the requirement. By implementing the truncation and rounding parameters, the levels in obtaining final result can be reduced compared to that of exact multiplier schemes. Enhancement of this work is done by improving the accuracy of the proposed multiplier. The proposed work is designed using VHDL, simulated with Modelsim and synthesized using Xilinx ISE design suite 14.2 tool**

Keywords— Truncation, rounding, approximate, VHDL, partial products, accuracy, absolute error, energy, scalable

I. INTRODUCTION

In considering the deep submicron technology, power has become a main concern due to increase in transistor count, higher speed of operation and greater device leakage currents. Scaling in nanometer technologies created problems in reliability and testing. So deep submicron technology has much leaned towards power. Commercial success of products depends on size, speed, weight, cost, computing power and battery life. Many DSP applications involve complex probabilistic mathematical models and are designed to process information that typically contains noise. So, for some computational error, they exhibit elegant degradation instead of catastrophic failure. Multipliers are one of the data path operators which occupy major share in a digital system. So, hence a multiplier is one of the deciding factor for system's performance. In addition, these multipliers consume most of the area. Hence, optimization of area and speed of a multiplier is a major design issue.

Approximate computing is a paradigm where we can reduce energy consumption or boost up the speed or at times both. By using the approximation techniques in multiplier techniques one can have the advantage of low energy consumption or high speed. As these computations result in approximate results which are very near with the exact computation, these are best choice for the applications which are error resilient. Most of the recent trending technologies like machine learning, computer vision and image processing are some of the areas where applications are designed to deal with noisy inputs and generate satisfactory results as shown in fig 1.

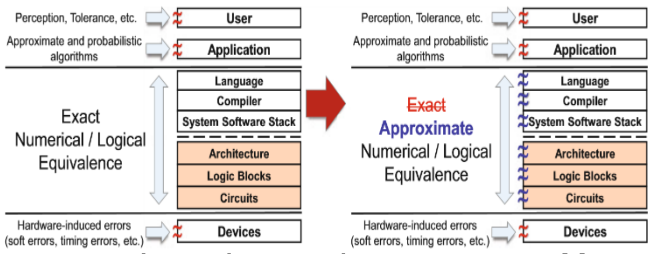
Perception, Tolerance, etc.
User
Approximate and probabilistic algorithms
Application
Exact Numerical / Logical Equivalence
Language
Compiler
System Software Stack
Architecture
Logic Blocks
Circuits
Hardware-induced errors (soft errors, timing errors, etc.)
Devices
Exact
Approximate
Numerical / Logical Equivalence

Fig 1. Hardware stack in exact and approximate computing [1]

In several cases the best useable algorithms cannot acquire exact results every time, occasionally it is besides expense to produce optimal results when dealing meta-heuristics and probabilistic algorithms. Insofar, the designers have largely adopted a notion of exact numerical or Boolean equivalence between the implementation and specification at different layers in the hardware and software stacks. Approximate computing focuses at multiple layers of the hardware/software stacks to trade-off computation accuracy with performance, power, and/or area. The table 1 presents the types of approximations where in the schemes can be applied to various levels of the stack to achieve desired outcomes. In this work we mainly rely on the data approximation as the input data will be truncated to small widths and the shift unit logic is altered with logic 1 to observe reduced absolute error.

The multiplier is used mainly in respective DSP applications and contributes in intensifying the performance of the practical application. This work includes the implementing both truncation and rounding schemes on input operands and produce the approximate results in minimum number of steps in which they are very near to the exact multiplication approach, additionally calculating the absolute error for 8-bit, 16-bit and 32-bit of the proposed multiplier scheme in accordance with exact result and enhance the absolute error by modifying the least significant part of the multiplication process.

Selective Approximation	Analysis of software code or instructions to suggest a certain accuracy mode for a part of code
Timing Relaxation	Relaxing of synchronization, timing and handshaking constraints to reduce control overhead
Functional Approximation	An approximate alternative of an algorithm that improves area/power performance
Domain Specific Approximation	Leveraging the domain specific knowledge for approximations in applications and their algorithms
Data Approximations	Use of unreliable memories, load value approximation, data truncation, data decimation, etc.

Table 1. Categorization for the types of approximation

In general, the process of multiplication involves partial products generation from the acquired operands and cumulate those partial products to a level till two rows are persisted. Now by employing an adder the remained rows adds up to generate final result. The process of approximation can be carried out in any levels of the multiplication process. One can use approximation in the beginning of the process i.e. on the operands to reduce the partial products number for a multiplier. The process of approximation can be applied at cumulation of partial product to reduce the power consumption. By employing an approximate adder at the ultimate step of adding we can achieve betterment in power consumed by a multiplier.

In this paper, we introduce an approximation method for reduced number of partial products by considering truncation and rounding parameters which are denoted by t and r respectively. Previous methods of approximation techniques involves employing either truncation or rounding parameters. In this work, by considering both the parameters we can achieve better outcomes than former methods. Let A and B be two 16 bit operands, multiplication of A and B results 256 partial products. Hence by processing all the 256 partial products we get the result and it occupies large area. So to reduce the partial products we employ scalable truncation and rounding parameters as per the requirement. Let us consider the truncation parameter t = 7 and rounding parameter r = 3 as illustrated in the figure 2**.**

The green square represents the "1" in the equation (5) and the orange circles denotes the partial products of approximation and the blue triangles denotes the truncated bits of input operands. The remaining grey triangles of the operands and the grey circles of the partial products can be neglected. With truncation and rounding parameters mentioned in fig.2 only 31 partial products are required to produce approximate result which is very close with exact. Around 87.8% of the partial products were reduced. This reduction of partial products will increase with increase in bit length of operands.

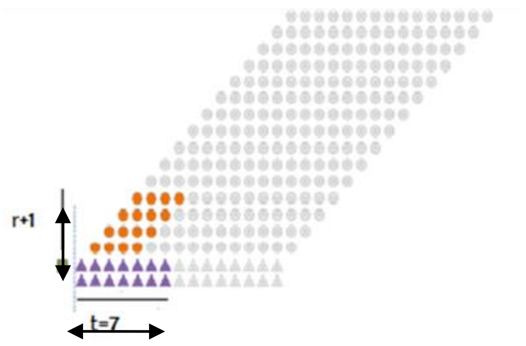

Fig 2. Dot diagram of multiplier

II. HARDWARE IMPLEMENTATION

Block diagram implementation of the proposed signed approximate multiplier is shown in fig.3 which comprises of following units:

Approximate Absolute Unit (AAU), which produces inverted results if the input is negative and is not changed if input is positive. For unsigned multipliers this AAU can be eliminated.

Leading One Detector Unit (LOD) will provide the leading 1-bit position in the most significant part of the operands. The position of the LOD can be obtained by using

$$K[i] = \left(\bigwedge_{j=i+1}^{n-2} \overline{I[i]}\right) \wedge I[i] \text{ for } 0 \leq i \leq n-2 \qquad (1)$$

Only one bit of the signal K is '1' which says about the position of the leading bit. For both operands A and B there exists leading bits k_A and k_B respectively.

Truncation Unit (TU) deals with truncating the approximation inputs based on leading one position and converts into fixed width operands.

Arithmetic Unit (AU) will implement the addition of truncated fixed width operand along with product of approximated input with unity which can be expressed as

$$TU = 1+(Y_A)_t + (Y_B)_t+ (Y_A)_{APX} \times (Y_B)_{APX} \qquad (2)$$

Shift unit (SU) shifts the resultant of arithmetic unit with k_a+k_b times leftwards to generate the result.

Sign And Zero Detector Unit (SZD) produces the sign of the output based on the input operands of the multiplier. SZD will produce zero if any of the input is zero. In case of unsigned multiplication this unit is replaced by zero detector unit.

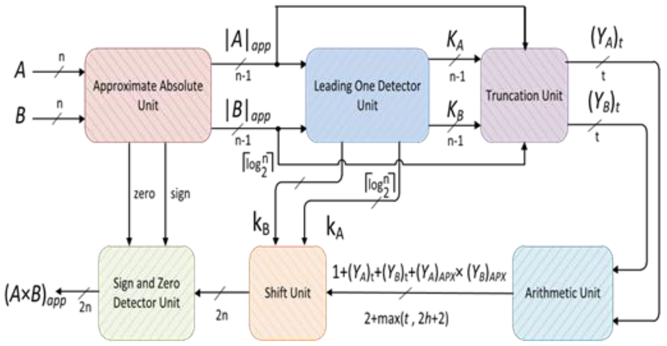

Fig. 3. Block diagram of proposed signed approximate multiplier

Numerical Example of Proposed Multiplier:

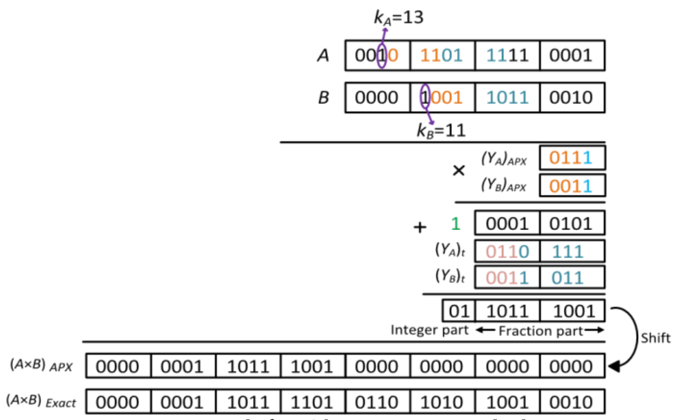

Fig 4. Example for 16-bit approximate multiplier

The approximate result $[(A \times B)_{APX}]$ = 28901376. Whereas the exact multiplication result $[(A \times B)_{Exact}]$ = 29190802. The approximate results are very close to the actual results.

III. ABSOLUTE ERROR CALCULATION

To measure the accuracy of the proposed approximate multiplier we use a parameter named absolute error which is specified as shown below

Absolute error (AE) = $(P_{app} - P)/p$ (3)

P_{app} denotes approximate value and P denotes exact value.

Percentage of absolute error can be calculated as

Percentage of AE = $[\,|(P_{app} - P)|/ P\,] *100$ (4)

In considering the calculation of 16-bit multiplier in above section the percentage of absolute error is calculated to be 0.99% of the actual result which is very low.

IV. RESULTS AND COMPARISONS

In the proposed work the shift unit of the absolute multiplier is modified with logic-1 instead of logic-0 and the observations are as follows:

A = 11761 ***B = 2482***

Exact Result = 11761×2482 = 29190802

Approximate result with logic-0 in shift unit = 28901376

Approximate result with logic-1 in shift unit = 28966911

Hence by intensifying the shift unit the absolute error has cut down from 289426 to 223891.

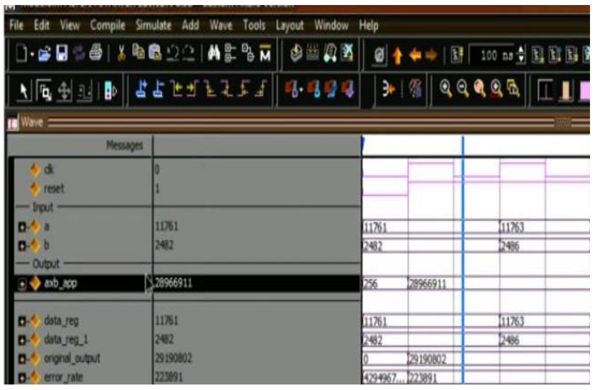

Fig. 6. Simulation of 16-bit absolute multiplier with intensified shift unit

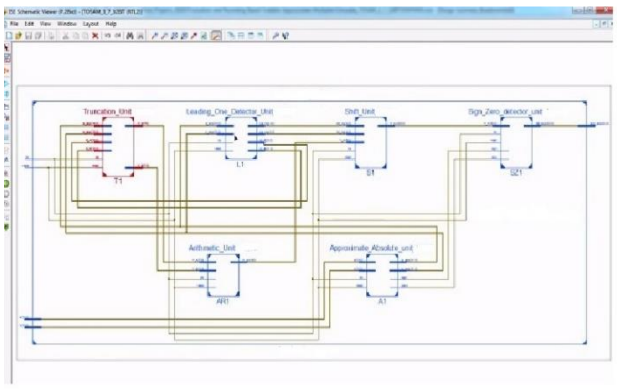

Fig. 5. RTL Schematic of proposed absolute multiplier

V. CONCLUSION

The proposed approximate multiplier in this paper is designed by considering the rounding parameter r = 3 and truncation parameter t = 7. By altering the logic of shift unit, the error rate is reduced from 0.99% to 0.76% of the actual result for 16-bit multiplier.

ACKNOWLEDGMENT

I wish to record my obligation and gratefulness to all those who helped me to prepare this paper. I want to convey my thanks to my guide Mr E.Venkata Narayana, Assistant Professor in Department of Electronics and Communication Engineering for his valuable advice and guidance towards this work. I received motivation, encouragement and hold up from him during the course of work.

REFERENCES

[1] Shafique M, Hafiz R, Rehman S, El-Harouni W, Henkel J (2016) Invited: cross-layer approximate computing: from logic to architectures. In: 2016 53nd ACM/EDAC/IEEE design automation conference (DAC), June 2016, pp 1–6

[2] 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

[3] S. Vahdat, M. Kamal, A. Afzali-Kusha, and M. Pedram, “LETAM: A low energy truncation-based approximate multiplier,” Comput. Elect. Eng., vol. 63, pp. 1–17, Oct. 2017

[4] R. Zendegani, M. Kamal, M. Bahadori, A. Afzali-Kusha, and M. Pedram, “RoBa multiplier: A rounding-based approximate multiplier for high-speed yet energy-efficient digital signal processing,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 2, pp. 393–401, Feb. 2017.

[5] S. Hashemi, R. I. Bahar, and S. Reda, “DRUM: A dynamic range unbiased multiplier for approximate applications,” in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), Austin,TX, USA, Nov. 2015, pp. 418–425

[6] A. Raghunathan and K. Roy, “Approximate computing: Energy efficient computing with good enough results,” in Proc. IEEE 19th Int. On-Line Test. Symp. (IOLTS), Chania, Greece, Jul. 2013, p. 258.