Rounding Technique Analysis for Power-Area & Energy Efficient Approximate Multiplier Design

Priyanka Lohray
Electrical & Computer Eng. Dept.
Texas Tech University
Lubbock, Texas, USA
Priyanka.Lohray@ttu.edu

Satwik Gali
Electrical & Computer Eng. Dept.
Texas Tech University
Lubbock, Texas, USA
Satwik.Gali@ttu.edu

Srirathan Rangisetti
Electrical & Computer Eng. Dept.
Texas Tech University
Lubbock, Texas, USA
Srirathan.Rangisetti@ttu.edu

Tooraj Nikoubin
Electrical & Computer Eng. Dept.
Texas Tech University
Lubbock, Texas, USA
Tooraj.Nikoubin@ttu.edu

***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 three multipliers. These three multipliers are the conventional Wallace tree accurate multiplier, DRUM [4] the recently proposed approximate multiplier and the rounded based approximate multiplier proposed in this work. Simulation results for three selected technologies show significant improvement on the circuit characteristics in terms of power, area, speed, and energy for proposed multiplier in comparison with their counterparts. Input data rounding pattern and the probability of the repetition for rounded values has been introduced as two essential items to control the level of the accuracy for each range of the data with minimum cost on the hardware.**

Keywords—Data Processing, Digital Arithmetic, Approximate computing, Energy efficient, Hi-performance, Rounding Technique.

I. INTRODUCTION

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. This reduces the number of partial products for compression significantly. Compression of active partial products which takes most of the time, has been diminished and it has good impact on the circuit delay. Compressed output block contains both accurate compressor block and OR gates based approximate compressors to preserve its accuracy and minimize the hardware. The rest of the paper is organized as follows: section II represents related works, section III explains the proposed design of approximate multiplier, section IV Error analysis on rounding technique, simulation result analysis is on section V, then accuracy analysis is presented in section VI, and finally section VII contains the conclusion.

II. RELATED WORKS

Major research in approximate multipliers has mainly relied on reducing partial products or truncating it. There are different techniques for the implementation of approximate computation, which can be classified into two types: techniques at the hardware level with less accurate and highly efficient energy components; and software-level techniques that reduce calculations or accesses to memory to improve performance at the expense of precision in the results. Recently, a high-speed energy efficient multiplier based on rounding of the inputs in the form of 2n has been proposed in [1]. This approach dramatically improved the speed and the energy consumption (up to 65%)

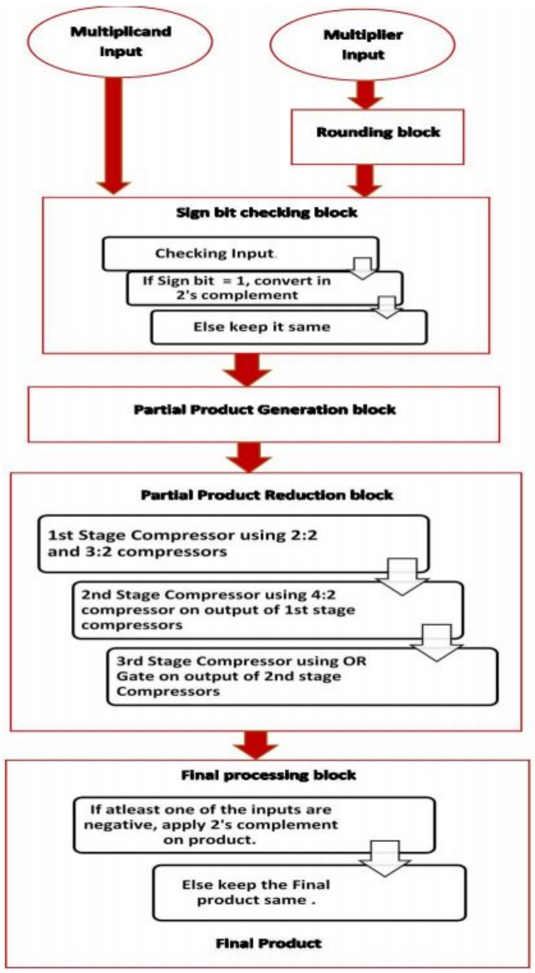

Multiplicand Input
Multiplier Input
Rounding block
Sign bit checking block
Checking Input
If Sign bit = 1, convert in 2's complement
Else keep it same
Partial Product Generation block
Partial Product Reduction block
1st Stage Compressor using 2:2 and 3:2 compressors
2nd Stage Compressor using 4:2 compressor 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

since the computationally intensive part of the multiplication was omitted. When looked at hardware level, [2] presents a hardware architecture with reconfigurable kernels and overflow-resilient limiter. There are several methods in the recent literature that relax a single part of a computing system (e.g., a functional unit [3]) for improved design. Authors in [4] proposed an approximate multiplier design with an error distribution reducing the propagation delay and improving the energy efficiency. None of the research focused on preprocessing data before computation. This paper presents a method of rounding input data before approximate multiplier design and shows a significant reduction in power, area, delay and energy.

III. Proposed Design of Approximate Multiplier

A. *Block diagram*

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

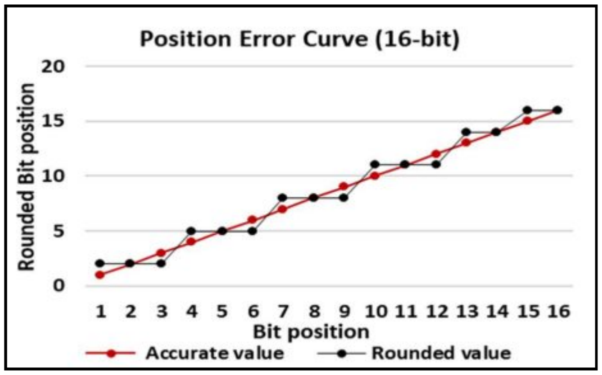


Fig. 2: Position error curve (16-bit)

TABLE 1: ROUNDING ALGORITHM (16-BIT)

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. 3: An example after rounding 'B' (16-bit)

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.

B. *Input Rounding*

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. From fig. 2, proposed method shows 16-bit position error curve that gives less weight to lower bits and more weight to higher bits. 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 as seen in Table 1. Error gap reduces as the bit position value increases. Fig. 3 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.

C. *Partial product Reduction*

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. 6 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. 6. 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. 6 (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.

IV. 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

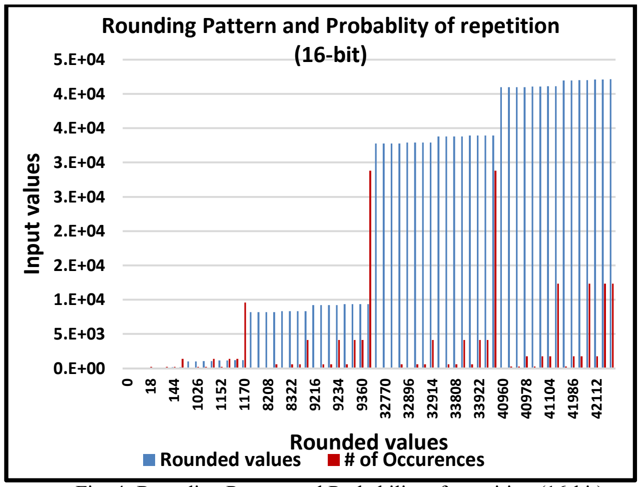
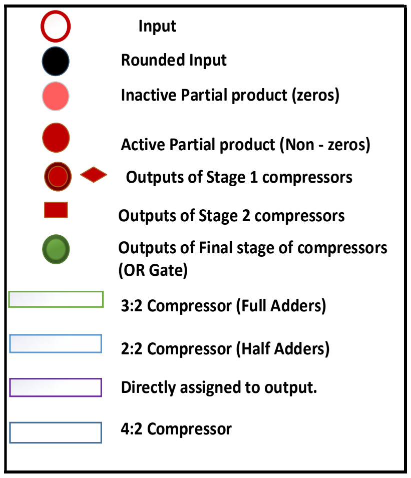


Fig. 4: Rounding Pattern and Probability of repetition (16-bit)

TABLE 2: ROUNDED VALUES AND THEIR OCCURRENCES RATE

Rounded value (DEC)	# of Occurrence of Rounded value	Rounded value (DEC)	# of Occurrence of Rounded value	Rounded value (DEC)	# of Occurrence of Rounded value
0	1	8338	1029	33936	1029
2	7	9216	21	33938	7203
16	7	9218	147	40960	9
18	49	9232	147	40962	63
128	7	9234	1029	40976	63
130	49	9344	147	40978	441
144	49	9346	1029	41088	63
146	343	9360	1029	41090	441
1024	7	9362	7203	41104	441
1026	49	32768	3	41106	3087
1040	49	32770	21	41984	63
1042	343	32784	21	41986	441
1152	49	32786	147	42000	441
1154	343	32896	21	42002	3087
1168	343	32898	147	42112	441
1170	2401	32912	147	42114	3087
8192	3	32914	1029	42128	3087
8194	21	33792	21	42130	21608
8208	21	33794	147		
8210	147	33808	147		
8320	21	33810	1029		
8322	147	33920	147		
8336	147	33922	1029		

Multiplier, Multiplier input is rounded. For 16-bit values ranging from 0 to 65535, a set of rounded values are obtained using rounding algorithm as discussed in fig. 2 from section III. Fig. 4 plots the relationship between rounded values and its occurrences. 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. Table 2 provides number of occurrences of rounded values against input values.

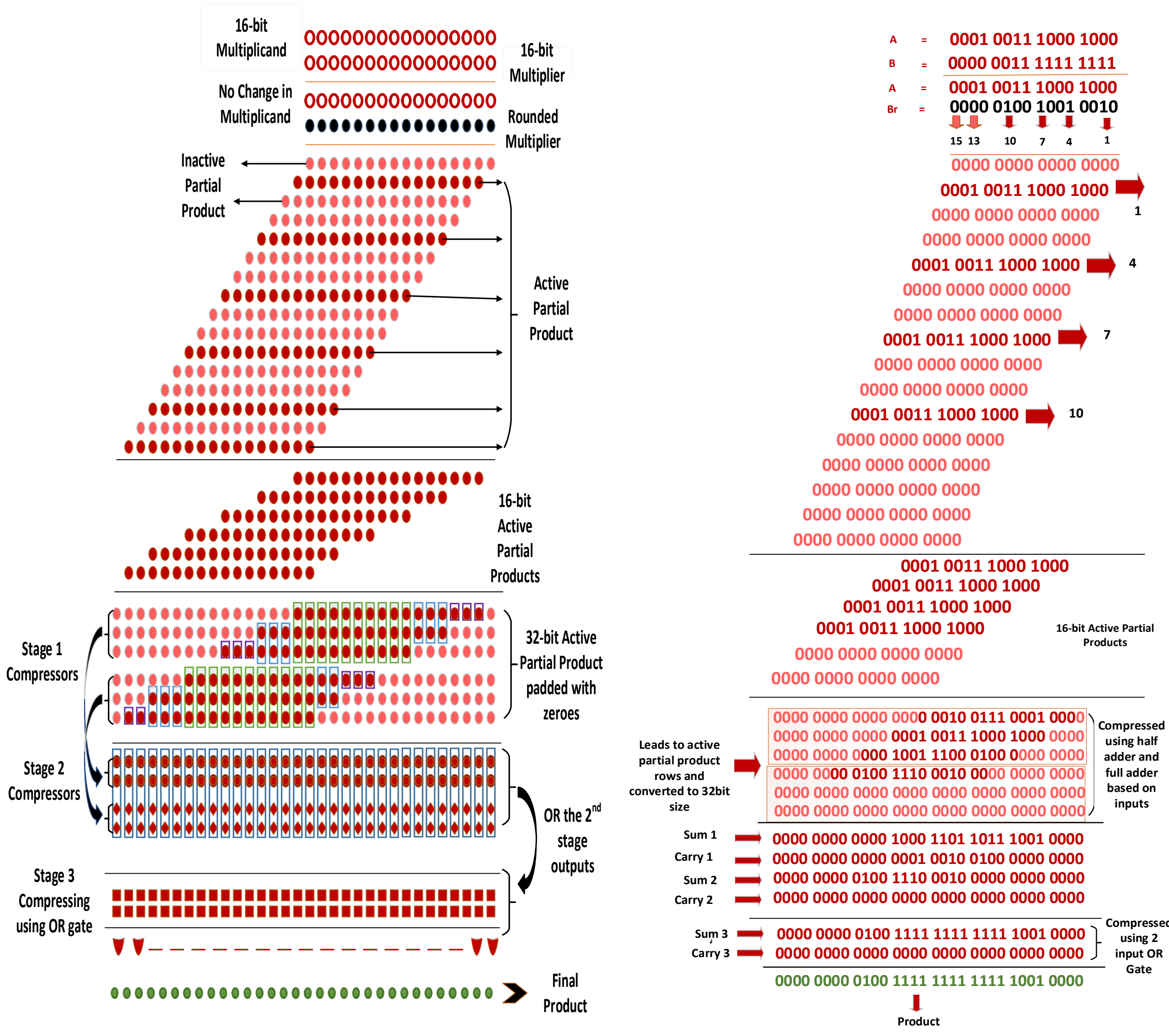


Input
Rounded Input
Inactive Partial product (zeros)
Active Partial product (Non - zeros)
Outputs of Stage 1 compressors
Outputs of Stage 2 compressors
Outputs of Final stage of compressors (OR Gate)
3:2 Compressor (Full Adders)
2:2 Compressor (Half Adders)
Directly assigned to output.
4:2 Compressor

Fig. 5: Labels for fig. 6.

V. Simulation Error Analysis

Power, Area, and Delay are calculated using design compiler on three different technologies (32nm, 45nm, and 90nm) for both 16-bit and 32-bit. Fig. 7 clearly shows that the proposed algorithm is better than other algorithms. It is compared with DRUM [4] and accurate multiplication using 32nm, 45nm, and 90nm. Table 3 shows the Area-Power-Delay comparable readings using design compiler. The result seen in table 3 reveals that, area usage is reduced by 45% (85%) for 16-bit(32-bit) inputs. Power usage has reduced significantly to 30% (5%) when compared to accurate 100% (100%) and DRUM [4] for 16- bit(32-bit).

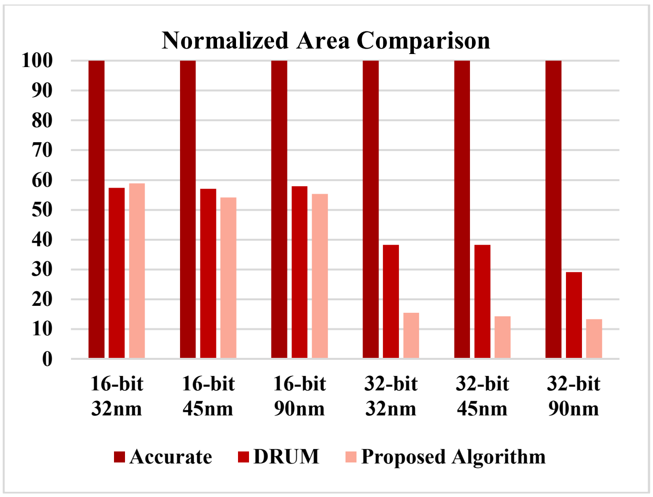


16-bit Multiplicand
16-bit Multiplier
No Change in Multiplicand
Rounded Multiplier
Inactive Partial Product
Active Partial Product
16-bit Active Partial Products
Stage 1 Compressors
32-bit Active Partial Product padded with zeroes
Stage 2 Compressors
OR the 2nd stage outputs
Stage 3 Compressing using OR gate
Final Product
A = 0001 0011 1000 1000
B = 0000 0011 1111 1111
A = 0001 0011 1000 1000
Br = 0000 0100 1001 0010
15 13 10 7 4 1
16-bit Active Partial Products
Leads to active partial product rows and converted to 32bit size
Compressed using half adder and full adder based on inputs
Sum 1 0000 0000 0000 1000 1101 1011 1001 0000
Carry 1 0000 0000 0000 0001 0010 0100 0000 0000
Sum 2 0000 0000 0100 1110 0010 0000 0000 0000
Carry 2 0000 0000 0000 0000 0000 0000 0000 0000
Sum 3 0000 0000 0100 1111 1111 1111 1001 0000
Carry 3 0000 0000 0000 0000 0000 0000 0000 0000
Compressed using 2 input OR Gate
0000 0000 0100 1111 1111 1111 1001 0000
Product

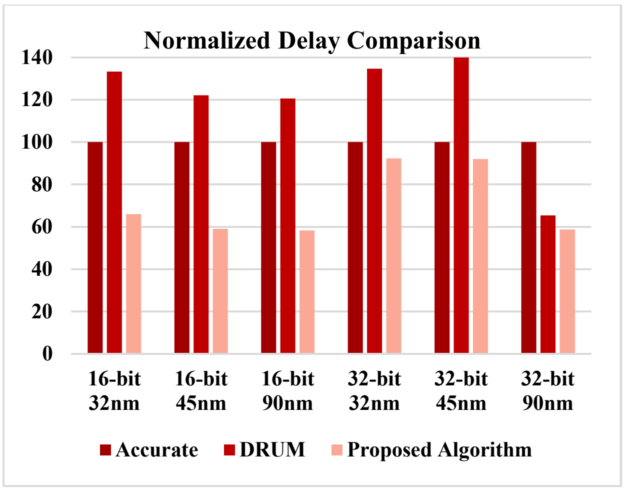
Fig. 6: Multiplier Design (left) with an example (right) (16-bit).

TABLE 3: AREA-POWER-DELAY COMPARISON FOR 16-BITS & 32-BITS (NORMALIZED)

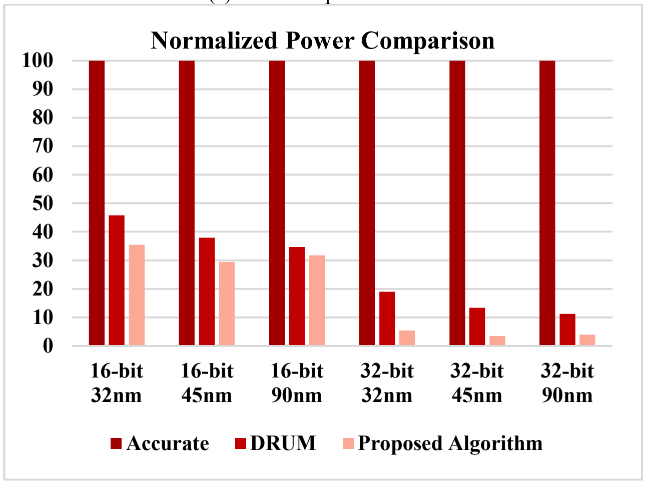
		16-bit						32-bit					
		32nm		45nm		90nm		32nm		45nm		90nm	
		µm²	Norm	µm²	Norm	µm²	Norm	µm²	Norm	µm²	Norm	µm²	Norm
Area	Accurate	3130.29	100.00	4830.97	100.00	11669.29	100.00	8725.01	100.00	13427.14	100.00	42684.82	100.00
	DRUM	1795.78	57.37	2756.19	57.05	6762.70	57.95	3342.50	38.31	5143.53	38.31	12451.74	29.17
	Proposed Al.	1844.83	58.93	2618.22	54.20	6453.96	55.31	1349.25	15.46	1924.60	14.33	5679.82	13.31
		mW	Norm	mW	Norm	mW	Norm	mW	Norm	mW	Norm	mW	Norm
Power	Accurate	0.34	100.00	3.53	100.00	6.70	100.00	1.23	100.00	13.09	100.00	26.28	100.00
	DRUM	0.15	45.77	1.34	37.92	2.32	34.63	2.32	18.93	1.74	13.34	2.93	11.18
	Proposed Al.	0.12	35.45	1.03	29.36	2.12	31.72	0.07	5.40	0.44	3.42	1.03	3.94
		ns	Norm	ns	Norm	ns	Norm	ns	Norm	ns	Norm	ns	Norm
Delay	Accurate	4.32	100.00	3.67	100.00	11.60	100.00	7.22	100.00	5.65	100.00	37.11	100.00
	DRUM	5.76	133.33	4.48	122.07	13.99	120.60	9.72	134.63	7.91	140.00	24.27	65.40
	Proposed Al.	2.85	65.97	2.17	59.13	6.77	58.36	6.66	92.24	5.20	92.04	21.79	58.72
		pJ	Norm	pJ	Norm	pJ	Norm	pJ	Norm	pJ	Norm	pJ	Norm
PDP	Accurate	1.46	100.00	12.96	100.00	77.72	100.00	8.88	100.00	73.96	100.00	975.25	100.00
	DRUM	0.86	61.03	6.00	46.29	32.46	41.77	22.55	25.49	13.76	18.68	71.11	7.31
	Proposed Al.	0.34	23.39	2.24	17.36	14.35	18.51	0.44	4.98	2.29	3.15	22.44	2.31



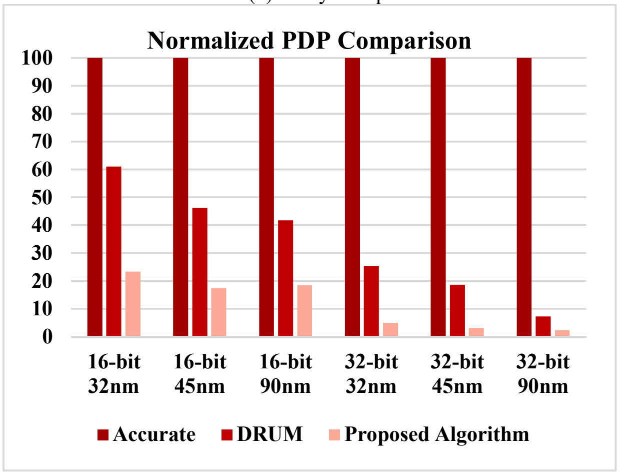
(a) Area Comparison



(b) Delay Comparison



(c) Power Comparison



(d) PDP Comparison

Fig. 7 (a) (b): Characteristic comparison of proposed algorithm and DRUM [2] in comparison with conventional accurate multiplier for 16 and 32-bit with three

TABLE 4: EXAMPLES OF MULTIPLIER WITH SMALL AND LARGE VALUES (16-BIT)

Samples	Small values					Large values				
	1	2	3	4	5	1	2	3	4	5
A	1000	153	2000	4000	213	13932	5000	20496	6500	35732
B	3	15	15	15	1023	127	1023	1023	8191	8191
Accurate	3000	2295	30000	60000	217899	1801944	5242768	23980320	57521864	268435160
DRUM	3024	2340	30240	60480	213696	1774080	5031936	21159936	52641792	243597312
Proposed Al.	2000	2482	32672	65344	229370	1769364	5115000	20967408	53241500	292680812

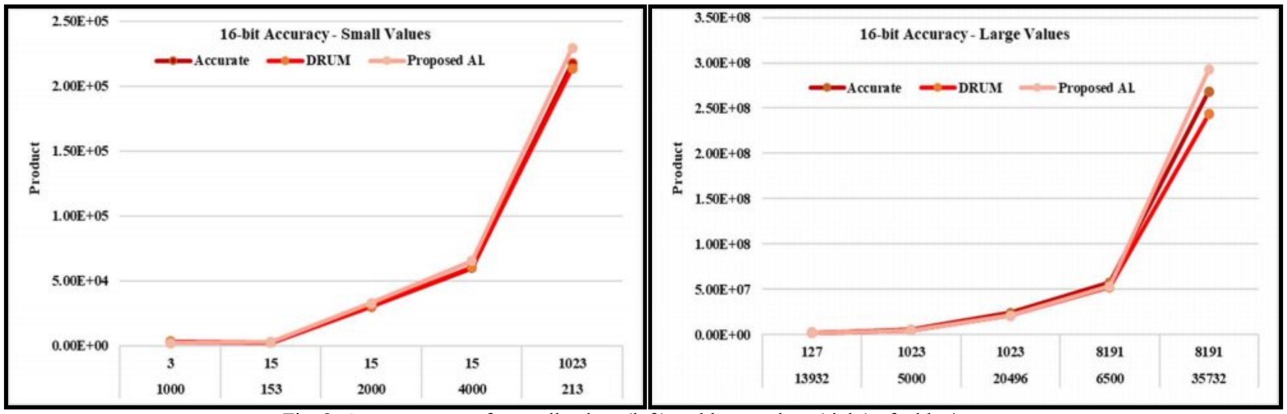


Fig. 8: Accuracy curve for small values (left) and large values (right) of table 4.

The results also indicate that, delay is as low as 60% (60%) when compared to accurate 100% (100% for 16-bit(32-bit).

VI. ACCURACY ANALYSIS

Table 4 shows, randomly chosen 5 sets of small and large values whose accuracy curves are as shown in fig. 8. 'A' is a Multiplicand and 'B' is a Multiplier. Multiplication result of proposed algorithm is compared with DRUM [4] and Accurate Multiplication. Results of randomly chosen sets shows comparable to other algorithms. But as seen in fig. 4 above, there are few areas where results of proposed algorithm may not be near to other algorithms. This increases an opportunity to modify rounding pattern based on required accuracy.

VII. 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 (as seen in fig. 2 and Table 1). With this rounding pattern, we see potential areas of less accuracy and areas with better accuracy corresponding to probability of rounding value (fig. 4 and Table. 2). 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 [4]. As future work, rounding patterns will be optimized based on required accuracy and different compression techniques.

REFERENCES

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

[2] M. Van Leussen, J. Huisken, L. Wang, H. Jiao, J. P. De Gyvez, "Reconfigurable Support Vector Machine Classifier with Approximate Computing", 2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). IEEE, pp. 13-18, jul 2017.

[3] Shaghayegh Vahdat, Mehdi Kamal, Ali Afzali-Kusha, Massoud Pedram, LETAM, Computers and Electrical Engineering, Science Direct, v.63 n.C, p.1-17, October 2017

[4] S. Hashemi, R. I. Bahar, and S. Reda, "DRUM: A Dynamic Range Unbiased Multiplier for approximate applications," IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 418-425, Nov. 2015.

[5] Bharat Garg, G, K Sharma: Low Power Signal Processing via Approximate Multiplier for Error Resilient Applications, 11th International Conference on Industrial and Information System (ICIIS), pp. 546-551, 2016.

[6] H.Bessalah, K.Messaoudi, M.Issad, N.Anane, M.Anane: Left to Right Serial Multiplier for Large Numbers on FPGA, Proceedings of the 2009 IEEE International Conference on Mechatronics Malaga, Spain, pp. 1-6, April 2009

[7] A. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE, pp. 820–825, june 2012.