Truncation Based Approximate Multiplier For Error Resilient Applications

Prashil Parekh, Samidh Mehta & Pravin Mane

INTERNATIONAL JOURNAL OF ELECTRONICS LETTERS
https://doi.org/10.1080/21681724.2021.1914183

ARTICLE

Truncation Based Approximate Multiplier For Error Resilient Applications

Prashil Parekh, Samidh Mehta and Pravin Mane

Department of Electrical Electronics Engg, BITS Pilani, K K Birla Goa Campus, Zuarinagar, India

ABSTRACT
Approximate computing is a promising approach for low power IC design and has recently received considerable research attention. To accommodate dynamic levels of approximation, a few accuracy configurable multiplier designs have been developed in the past. However, these designs consumed considerable area and power. Accuracy, as well as latency, power and area design metrics are used to evaluate our approximate multiplier designs of different bit widths, i.e. 16 x 16, 32 × 32 and 64 × 64. Simulation and synthesis results showed a considerable gain than previous designs since we can change the components required according to the error tolerance. Moreover, we have also proposed a technique, where the system takes charge of the design and makes a call depending on the magnitude of the numbers provided. When compared with the exact multiplier designs, for 16 bit, 32 bit and 64 bit, we achieved a reduction by 24.5%, 71.5% and 85.1% in area; reduction in power by 37.7%, 89.4% and 88.2% and with a mean relative error distance of 0.5393%, 0.5428% and 0.2878% respectively.

ARTICLE HISTORY
Received 28 June 2020
Accepted 29 March 2021

KEYWORDS
Approximate multiplier; dynamic approximation; error tolerant system; latency parameter; truncation

1. Introduction

ITRS- International Roadmap for Semiconductors had forecasted approximate computing to become state-of-the-art demand for applications which are error resilient and fault tolerant (Hoefflinger, 2011). This could lead to a reduction in power consumed, die area on the chip and also verification and testing costs in the industry. Due to limitations in the reaction time for the human visual and auditory systems, errors until a certain amount will not cause any degradation in the data processed by the brain. Some applications such as multimedia processing, wireless communication, data mining, and machine learning are already prone to background noise and thus performing accurate computation on erroneous data seems irrational. Thus, allowing for errors in that range will not have an impact on the output produced. Multipliers are an integral part for microprocessors, digital signal processors and embedded systems with applications ranging from filtering to convolution neural networks.

An approach was taken where approximation was included at various vital building blocks. Primarily they were included at the compressor level (Ansari et al., 2018; Esposito

CONTACT Prashil Parekh parekhprashil@gmail.com Department of Electrical Electronics Engg, BITS Pilani, K K Birla Goa Campus, Zuarinagar, India

et al., 2018; Maheshwari et al., 2015; Momeni et al., 2015) and as adders (Jiang et al., 2016; Reddy et al., 2015; Shafique et al., 2016). Compressors are units which count the number of ones in the input, a full adder can be considered as an example of a compressor. They are gaining attention due to their logic-rich circuitry which aids in partial product reduction. After they computed the partial products, different designs of variable length approximate adders are employed for the efficient summation- approximations can be made at both the boolean level or high level.

Truncation was another method to include approximations in the design (Gupta et al., 2013; Jiang et al., 2016; Kahng & Kang, 2012; Mazahir et al., 2016; Shafique et al., 2016; Swartzlander, 1999; Ye et al., 2013; Zhu et al., 2010). They achieved improvements in timing, area and power by skipping the implementation of less significant bits. This can be either introduced at the partial product reduction level or at the high level of the number itself. Our paper discusses two schemes based on truncation approximate multiplication. The first scheme is based on the previous research on approximate multipliers(Hashemi et al., 2015) and to alleviate some of its disadvantages, we propose a second scheme of approximate multiplier.

The type of the truncation-based multiplier proposed in this paper has applications where the scope of error is low. The proposed approximate multipliers can be used in some applications of image processing like sharpening the image by using a Gaussian filter(Garg & Sharma, 2016), edge detection algorithms like Marr and Hilderith (Smith et al., 1988), texture (Unser & Eden, 1989), image blurring(Popkin et al., 2010) and can also be used in MAC (Masadeh et al., 2019).

In this paper, comparisons of three error parameters - relative error, mean relative error distance and acceptance probability for 100,000 random numbers are presented for both the schemes of approximate multipliers in the following sections. Relative error is defined as percentage ratio of the difference between exact value and approximate value to the exact value. Mean relative error distance is the average of all the relative errors. Acceptance probability is the probability that the multiplication output produced by the approximate multiplier is within 1% of the relative error.

$$\mathit{Relative\ error}\ (\%) = \frac{(X_{\text{exactvalue}} - X_{\text{approximatevalue}})}{(X_{\text{exactvalue}})} \times 100$$

$$\mathit{Mean\ Relative\ error\ distance}(\%) = \frac{\sum_{n=1}^{n=n} \frac{(X_{\text{exact value(n)}} - X_{\text{approximatevalue(n)}})}{(X_{\text{exactvalue(n)}})}}{n} \times 100$$

Where $X_{\text{exact value}}$ is the value when the numbers are multiplied using accurate multiplier, $X_{\text{approximate value}}$ is the value when the numbers are multiplied using approximate multiplier and n is the total number of samples for which mean relative error distance is calculated.

Accurate multiplier is a multiplier which gives the value of multiplication with 0% relative error. It uses a binary type multiplication method(Gnanasekaran, 1985). The value of its area, power and timing with their respective lengths is given in Table 1, which will be compared with the approximate multiplier proposed in this paper. This design and all the designs presented in this paper have been synthesized by Cadence Genus tool, using GPDK045- slow_vdd1v0 1.0 library for 45 nm technology.

Table 1. Accurate multiplier.

Length	Area (nm^2)	Power (nW)	Timing (ps)
16 bit	1328	53,416	6759
32 bit	5424	357,094	7960
64 bit	21,076	784,255	7960

Table 2. 16 bit Scheme 1.

Latency	Area (nm^2)	Power (nW)	Timing (ps)
5	493	14,660	3664
6	603	20,154	4095
7	697	24,931	4486
8	818	32,466	4786
9	942	42,895	5435
10	1094	50,650	5492

Table 3. 32 bit Scheme 1.

Latency	Area (nm^2)	Power (nW)	Timing (ps)
6	853	23,740	4553
7	983	32,375	5208
8	1120	37,303	5455
9	1193	45,443	5683
10	1468	53,313	6310

Table 4. 64 bit Scheme 1.

Latency	Area (nm^2)	Power (nW)	Timing (ps)
7	2241	36,177	7356
8	2568	46,100	7533
9	2936	63,865	7744
10	3192	70,350	7600
11	3519	109,320	7958
12	3914	133,417	7958

2. Approximate multiplier: scheme 1

In Scheme 1, we use the principle that all the bits of a particular number are not required for approximate multiplication, and the operation can be performed by taking only a certain number of higher-order bits. The lower the number of bits used in the computation, lesser is the power and area consumed and higher is the error and frequency. The number of bits used in the multiplication of both the numbers is called the latency parameter. Here, multiplication is done of two 16,32 and 64 bit numbers respectively with a varying value of latency parameter to determine the effect on the area, power, frequency, and relative error on increasing the latency parameter. The latency parameter (K) is already chosen beforehand, and it is constant for a particular approximate multiplier. This latency parameter has been varied through several values in separate experiments in order to give a correlation between the latency and area, power and timing. Tables 2,3 and 4 contain these results.

It is important to strategically choose which bits from the two numbers need to be multiplied. The bits, which are the higher significant bits of the number, weigh more in multiplication. When represented in binary notation, we define the number of bits which are to be multiplied as N and the latency parameter as K. We first determine the leading one of the numbers since the zeros, which precedes the leading one are of no significance in multiplication. K bits following the leading one for both the numbers are taken for multiplication. Let the two numbers chosen for multiplication be K1 and K2, which are of length K bits each, and on multiplication of those two K bit numbers, we get a number S, which can be represented in 2 K bits. The location of the leading one in each of the two original numbers is stored such that that value can be used to determine the final shift required to obtain the correct output. If the leading one in the first number is at position L1 from the most significant bit and for the second number, it is at position L2 from the most significant bit, then the total number of unused bits in the multiplication are (N-L1-K + 1) + (N-L2-K + 1) = 2 N – 2 K – L1- L2 + 2. The answer S is shifted logically left by 2 N – 2 K – L1- L2 + 2 bits, and we obtain the final answer Y. A clear representation of this type of approximate multiplier can be seen in Figure 1.

The area, power and timing of scheme 1 for 16,32 and 64 bit approximate multiplier with varying latency parameter was obtained by synthesizing the verilog code of the circuit are shown in Tables 2, 3 and 4 respectively. Figure 2(a), 3(a) and 4(a) shows the graph of relative error with different latency parameters for 100,000 random numbers using the random module in python for 16,32 and 64 bit numbers respectively, Figure 2

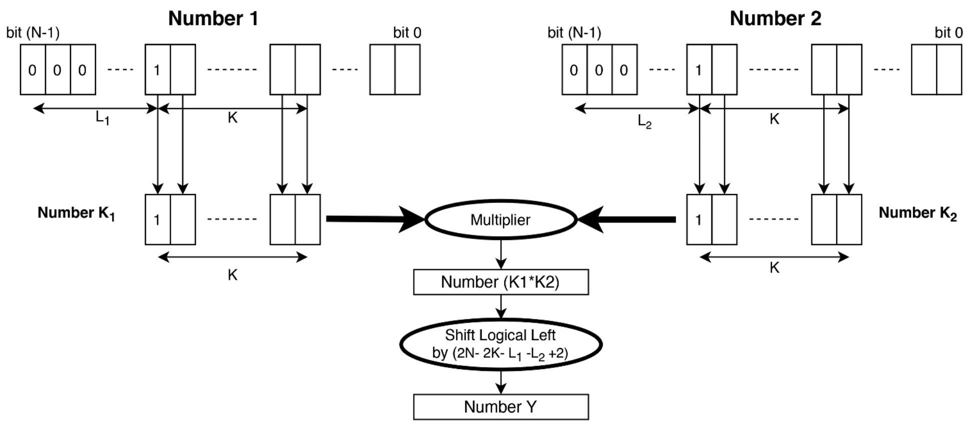
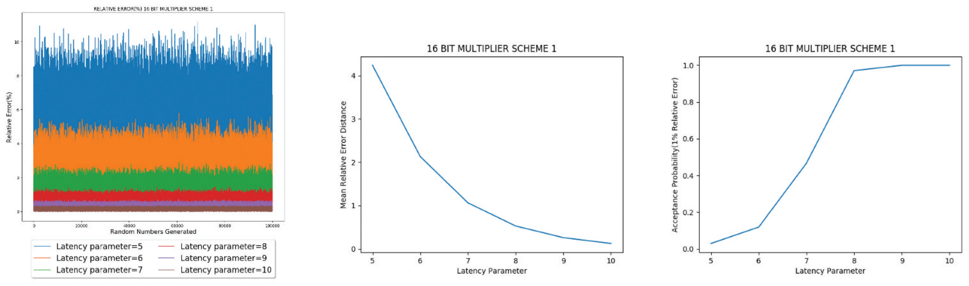
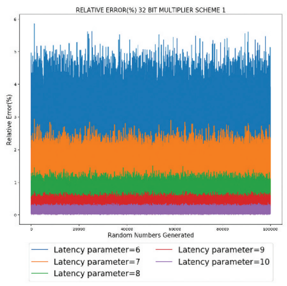


Figure 1. Schematic for approximate multiplier scheme 1.

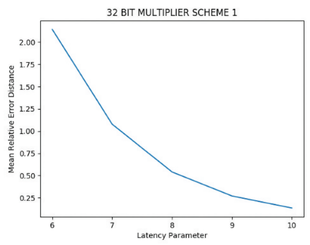


(a) Relative Error (b) Mean Relative Error Distance (c) Acceptance Probability

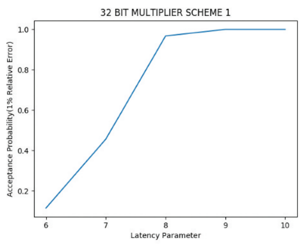
Figure 2. 16 bit Scheme 1.



(a) Relative Error

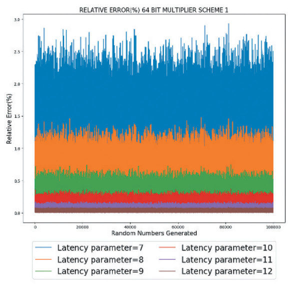


(b) Mean Relative Error Distance

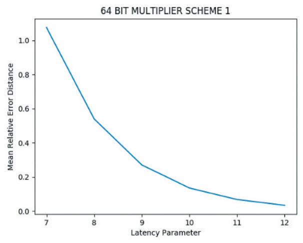


(c) Acceptance Probability

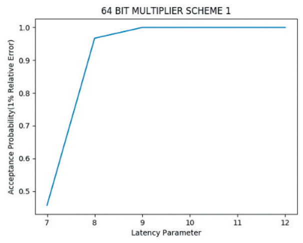
Figure 3. 32 bit scheme 1.



(a) Relative Error.



(b) Mean Relative Error Distance

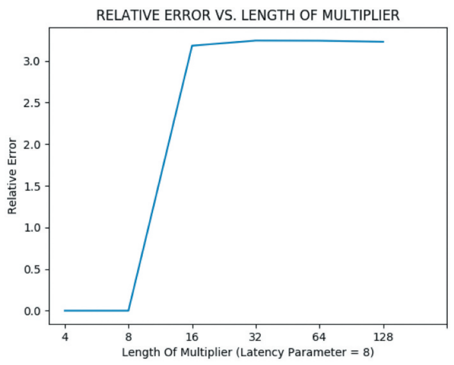


(c) Acceptance Probability

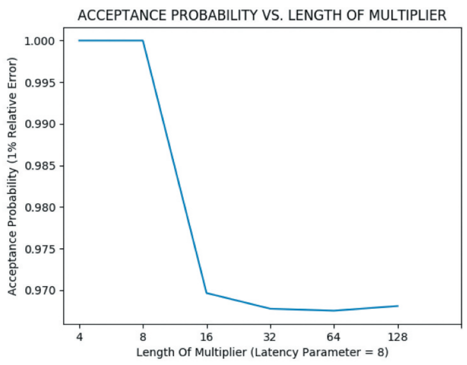
Figure 4. 64 bit Scheme 1.

(b), 3(b) and 4(b) shows the graph of mean relative error distance Vs. latency parameter for 16,32and 64 bit numbers respectively and figure 2(c), 3(c) and 4(c) shows the graph of acceptance probability for a relative error of 1% vs. latency parameter for 16,32 and 64 bit numbers respectively.

The relation between relative error, and acceptance probability individually with respect to bit length was determined for a fixed latency parameter and is shown in



(a)



(b)

Figure 5. Relation between bit length and relative error and acceptance probability for latency parameter 8.

Figure 5(a) and Figure 5(b) respectively. Figure 5(a) denotes that on increasing the bit length, the relative error for a fixed latency parameter will sharply increase for a particular bit length and then, will almost remain constant. A similar pattern is observed with acceptance probability and bit length in Figure 5(b). As the bit length is increased, the acceptance probability will sharply decrease till a certain bit length and after that, it will decrease at a negligible rate, remaining virtually constant. The experiments convey that the approximate multipliers which uses the method of leading 1 to compute the output will give lower error even if the latency parameter is kept constant and the initial bit length of the numbers to be multiplied is high.

It is also observed that the selection of a high latency parameter gives us favourable results when multiplying numbers of higher or lower magnitude, but the unwarranted use of a large latency parameter for lower magnitude of numbers leads to unnecessary power and area consumption. This fact is exploited in the design of Scheme 2.

3. Approximate multiplier: scheme 2

The second scheme is on lines with the first scheme, with a change made with respect to the selection of latency bits. The first scheme of approximate multiplier becomes very application-specific. If the approximate multiplier scheme 1 is designed for an application that has to multiply large numbers, a higher number of latency parameter bits would be required to have minimum mean relative error distance. Using this multiplier on smaller numbers would give a good error performance, but will consume the power and area needed for multiplication of larger numbers. This would be a wastage of power and area with not much benefit in terms of error performance. The second scheme has been designed in accordance with this predicament.

In this scheme of the approximate multiplier, the main advantage is that the latency parameter is not fixed and is flexible according to the numbers used for multiplication. If the two numbers to be multiplied are small in size, there is no need to use a large value of latency parameter and if the two numbers are large in size, there will not be much change in the error if a very large value of latency parameter is chosen. This is where the second scheme will save area and power compared to the first one with very little or nil degradation in relative error. The working of this approximate multiplier is the same as scheme 1 except for the fact that this scheme strategically chooses the latency parameter according to the two numbers that are to be multiplied.

- First of all, the bit position of leading one from the MSB (most significant bit) is determined for both the numbers.
- The bit positions of the leading one of both the numbers are fed to a logic level map which varies according to the size of the multiplier. The logic level map will give the select lines of 4:1 multiplexer as the output, as shown in .
- The input lines of the multiplexer will be the different values of the latency parameter to choose according to the numbers multiplied and the latency parameter will be selected on the basis of the maximum of the two numbers to be multiplied.

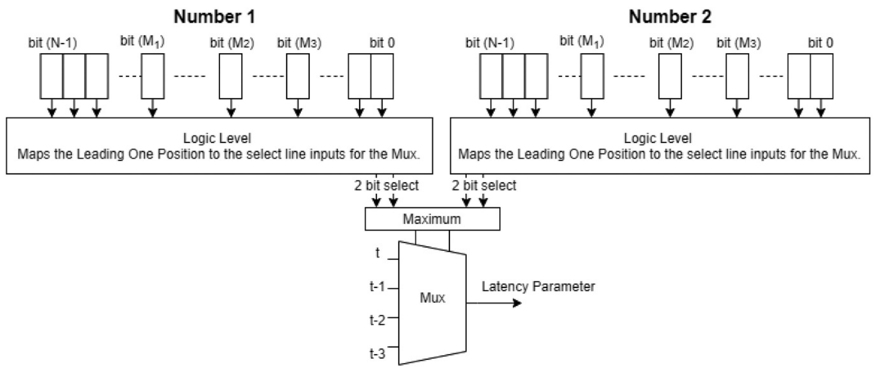


Figure 6. Selecting latency parameter for approximate multiplier scheme 2.

• The input line t will be chosen if the bigger number has the position of leading one between MSB and M1th bit.

• If the position of leading one from MSB is between (M1-1)th and M2th bit, then latency parameter t - 1 is selected for that numbers.

• If the position of leading one from MSB is between (M2-1)th and M3th bit, then latency parameter t - 2 is selected for that numbers.

• If the position of leading one from MSB of one of the numbers is between (M3-1)th bit and the least significant bit (LSB), then latency parameter t - 3 is selected for that numbers.

• M3 $>$ M2 $>$ M1 and the output of the multiplexer in Figure 6 will be the latency parameter to be selected.

• After selecting the latency parameter, all the remaining steps will remain the same in accordance with the scheme 1 of the approximate multiplier.

This was the method of selection of latency parameter in this scheme. Now, we shall see what is the logic map of t, M1, M2 and M3 and how are the values of t,M1,M2 and M3 selected.

• The logic map of t, M1, M2 and M3 will be different for different length of numbers and the values of t, M1,M2 and M3 are shortlisted in such a way that the mean relative error distance for the chosen values corresponding to their respective bit length is less than 1% and the acceptance probability (1% relative error) is more than 0.95 to ensure that the multiplier is error resilient. Various combinations of t,M1,M2 and M3 were tried for each 16,32 and 64 bit multipliers by writing python codes. Shortlisted values for 32 bit multiplier are given in Table 5.

• Verilog codes were written and synthesized for the shortlisted values of t,M1,M2 and M3 and the final values were be chosen in such a way that it had the least power and area among all the possibilities. The final values of t,M1,M2 and M3 for 16,32 and 64 bits are given in Table 6.

• For some cases, it is also possible that only M1 and M2 are required to be chosen for the best case, there might not be any need to choose M3, that means there will be only three values of latency parameters; t will be chosen if the maximum of the two numbers that are to be multiplied have the position of leading one between MSB and M1th bit, t-1 will be chosen if

Table 5. 32 bit approximate multiplier scheme 2.

Sr. No.	t	M1	M2	M3	Mean relative error distance(%)	Acceptance probability	Area (nm^2)	Power (nW)	Timing (ps)
1	9	28	23	20	0.2717	0.9996	1687	62,627	7219
2	9	27	22	19	0.2691	0.9999	1685	65,392	6978
3	9	31	26	23	0.3385	0.9920	1702	67,373	7176
4	8	29	23	-	0.5481	0.9577	1554	39,746	6462
5	8	28	25	-	0.5429	0.9681	1547	41,250	6542
6	8	29	24	-	0.5475	0.9584	1544	42,441	6336
7	8	28	24	-	0.5428	0.9659	1547	37,532	6384

Table 6. Approximate multiplier scheme 2.

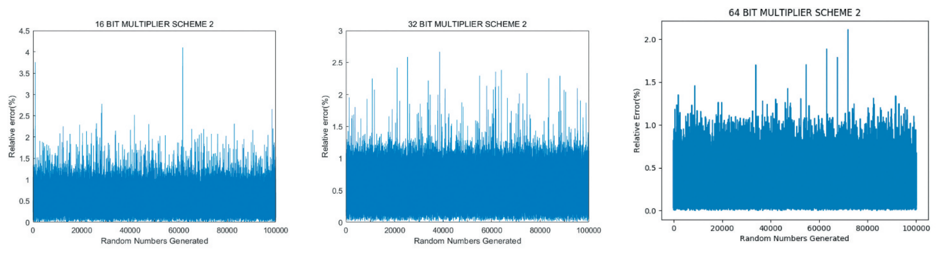
Bit length	t	M1	M2	M3	Area (nm^2)	Power (nW)	Timing (ps)	Mean relative error distance(%)	Acceptance probability
16 bit	8	13	10	9	1002	33,268	5208	0.5393	0.9621
32 bit	8	28	24	-	1547	37,532	6384	0.5428	0.9659
64 bit	10	62	58	55	3126	92,323	7960	0.2878	0.9980

the position of leading one is between (M1-1)th and M2th bit, t-2 will be chosen if the position of leading one is between (M2-1)th bit and LSB.

The area, power and timing of this type of approximate multiplier scheme 2 for 16, 32 and 64 bit and the probabilistic parameters for 100,000 random numbers of Scheme 2 are given in Table 6. The Relative error for 16,32 and 64 bit for this approximate multiplier is shown in Figure 7(a), 7(b) and 7(c), respectively.

4. Results

On comparison of area, power and timing of scheme 1 multiplier, all the hardware parameters, that are – area, power and timing for scheme 1 approximate multiplier and scheme 2 approximate multiplier are less than the accurate multiplier (Gnanasekaran, 1985) for all 16 bit, 32 bit and 64 bit, except for timing in 64 bit scheme 2 which is equal to that of accurate multiplier. To depict a comparison, Figure 8 shows the comparison between area, power and timing of 32 bit approximate multiplier scheme 1 with 32 bit accurate multiplier.



(a) Relative Error for 16 bit multiplier (b) Relative Error for 32 bit multiplier (c) Relative Error for 64 bit multiplier

Figure 7. Relative error scheme 2.

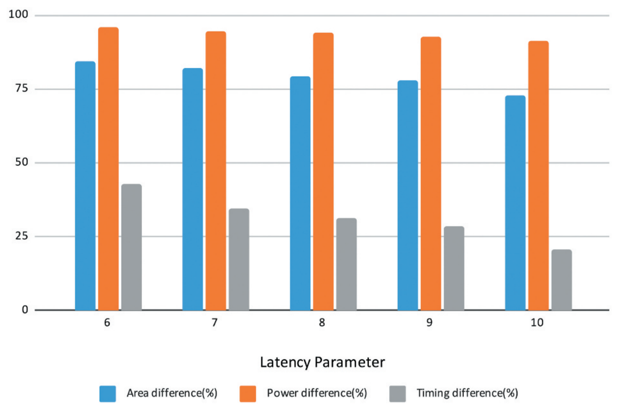


Figure 8. Comparison of 32 bit approximate multiplier scheme 1 with 32 bit accurate multiplier.

What is observed from scheme 1 is that as we keep on increasing latency parameter for a particular bit length, the hardware parameters (area, power, timing) increase, mean relative error distance decreases, relative error decreases and the acceptance probability increases. The generalized comparison between scheme 1 and scheme 2 is that as we keep on increasing latency parameter for scheme 1, there will come a stage when value of hardware parameters for scheme 2 will be smaller than that of scheme 1, with almost same accuracy as scheme 1. We can verify this fact by comparing 16 bit scheme 1 and scheme 2 multiplier. power consumed by scheme 2 is just slightly greater than that of scheme 1 latency parameter 8, area consumed for scheme 2 is between latency parameter 9 and 10 of scheme 1, timing of scheme 2 is between latency parameter 8 and 9 of scheme 1. The mean relative error distance of 16 bit scheme 2 is 0.5393 which is almost half of the mean relative error distance of 16 bit scheme 1 of latency parameter 7.

So, we observe a trade-off between hardware parameters and accuracy. If we choose a low latency parameter for scheme 1, then the area, power and timing consumed will be low but it will affect accuracy. If we choose a high value of latency parameter for scheme 1, then the accuracy will be really good but it will take a toll on hardware parameters. If scheme 2 is chosen, then the trade-off for scheme 1 is balanced, we get good enough accuracy and without much expense of hardware parameters. Also, as shown in Figure 9, when 16 bit scheme 2 is compared with 16 bit accurate multiplier, the area reduces by 24.5%, power reduces by 37.7% and the delay reduces by 22.9%. When a comparison is made between 32 bit scheme 2 and 32 bit accurate multiplier, the area and power consumed reduces by 71.5% and 89.4% respectively, and the delay of scheme 2 is decreased by 19.8%. When comparison is made between 64 bit scheme 2 and 64 bit accurate multiplier, the area and power consumed reduces by 85.1% and 88.2% respectively, and the delay is the same as that of the accurate multiplier.

Comparisons of mean relative error distance, power and area of both the schemes of 16 bit approximate multiplier with some other 16 bit approximate multipliers like Evolutionary approached approximate multiplier (EAM) (Vasicek & Sekanina, 2015) and approximate multiplier with formal error guarantee (FEG) (Ceska et al., 2017) are given in Table 7. Like both the multiplier schemes mentioned in this paper, EAM and FEG are also synthesized by

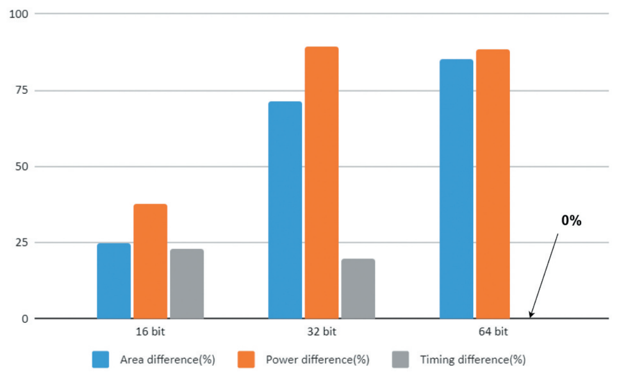


Figure 9. Comparison of approximate multiplier scheme 2 with accurate multiplier.

Table 7. Comparison of other 16 bit multipliers with scheme 1 and scheme 2 approximate multiplier.

Sr. No.	Multiplier type	Mean relative error distance(%)	Area (nm^2)	Power (W)	Timing (ps)
1	EAM	3.32	1670	58.4	6825
2	FEG	2.44	985	37.6	4537
3	Scheme1 (LP = 5)	4.24	493	14.6	3664
4	Scheme1 (LP = 6)	2.14	603	20.2	4095
5	Scheme1 (LP = 7)	1.08	697	25	4486
6	Scheme1 (LP = 8)	0.53	818	32.5	4786
7	Scheme1 (LP = 9)	0.26	942	42.9	5435
8	Scheme1 (LP = 10)	0.13	1094	50.7	5492
9	Scheme2	0.54	1101	33.2	5208
10	Accurate multiplier	0	1328	53.4	6759

using Cadence Genus tool, using the GPDK045 – slow vdd1v0 1.0 library for 45 nm technology.

From Table 7, it can be seen that scheme 2 approximate multiplier is better than EAM and FEG in the terms of mean relative error distance as both EAM and FEG have higher mean relative error distance than scheme 2. The power consumed by EAM and FEG is greater than scheme 2 by 76% and 13% respectively. The area and timing of scheme 2 are better than EAM but FEG has a better area and timing than scheme 2. Since the accuracy and power consumption of scheme 2 is better than both EAM and FEG, scheme 2 can be preferred over both EAM and FEG.

When scheme 1 is compared with EAM, we can observe that the mean relative error distance of scheme 1 is better than EAM for latency parameter greater than 5. Power, area and timing for any latency parameter of scheme 1 is less than EAM. So, scheme 1 can be preferred over EAM due to low power, area and timing and better accuracy. When scheme 1 is compared with FEG, the mean relative error distance of scheme 1 is better than FEG for latency parameter greater than 5. As far as hardware parameters are concerned, the area consumed by scheme 1 with latency parameters 5, 6, 7, 8 and 9 are lesser than FEG, the power consumed by scheme 1 with latency parameters 5,6,7 and 8 are lesser than FEG and the timing of scheme 1 with latency parameters 5,6 and 7 are better than FEG. When choosing between FEG and scheme 1, there might be a trade-off for hardware parameters depending on the latency parameter selected for scheme 1, but overall as scheme 1 has less mean relative error distance than FEG, scheme 1 can be preferred over FEG.

5. Conclusion

The approximate multiplier scheme 2 was developed such that the benefits of approximate multiplier scheme 1 like low relative error, high acceptance probability are maintained and at the same time, use less area and power than some of the latency parameters of approximate multiplier scheme 1. We can say that the scheme 1 multiplier is better than the accurate multiplier if the application is resilient to minimal error. To add on, scheme 2 multiplier is better than than the scheme 1 multiplier counterpart when all the factors like area, power, timing and probabilistic parameters are considered. On increasing the latency parameter in scheme 1, we observe a trade-off between hardware parameters like area, power, timing and accuracy, which is countered by scheme 2 multiplier as after a particular value of latency parameter, scheme 2 will have lesser power, area and timing than scheme 1 with the same amount of accuracy. Comparisons have also been made with the recent state-of-the-art multipliers and it was found that both scheme 1 and scheme 2 multiplier fare better in the terms of hardware parameters and accuracy.

Disclosure statement

No potential conflict of interest was reported by the author(s).

References

Ansari, M. S., Jiang, H., Cockburn, B. F., & Han, J. (2018). Low-power approximate multipliers using encoded partial products and approximate compressors. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, *8*(3), 404–416. https://doi.org/10.1109/jetcas.2018.2832204

Ceska, M., Matyas, J., Mrazek, V., Sekanina, L., Vasicek, Z., & Vojnar, T. (2017). Approximating complex arithmetic circuits with formal error guarantees: 32-bit multipliers accomplished. *2017 IEEE/ACM International Conference On Computer-Aided Design (ICCAD),* Irvine, California, USA. https://doi.org/10.1109/iccad.2017.8203807

Esposito, D., Strollo, A. G. M., Napoli, E., Caro, D. D., & Petra, N. (2018). Approximate multipliers based on new approximate compressors. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *65*(12), 4169–4182. https://doi.org/10.1109/tcsi.2018.2839266

Garg, B., & Sharma, G. (2016). A quality-aware energy-scalable gaussian smoothing filter for image processing applications. *Microprocessors And Microsystems*, *45*(Part A), 1–9. https://doi.org/10.1016/j.micpro.2016.02.012

Gnanasekaran, G. (1985). A Fast Serial-Parallel Binary Multiplier. *IEEE Transactions On Computers*, *C-34*(8), 741–744. https://doi.org/10.1109/TC.1985.1676620

Gupta, V., Mohapatra, D., Raghunathan, A., & Roy, K. (2013). Low-power digital signal processing using approximate adders. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *32*(1), 124–137. https://doi.org/10.1109/TCAD.2012.2217962

Hashemi, S., Bahar, R. I., & Reda, S. (2015). DRUM: A dynamic range unbiased multiplier for approximate applications. *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD),* Austin, Texas, USA. https://doi.org/10.1109/iccad.2015.7372600

Hoefflinger, B. (2011). ITRS: The International Technology Roadmap for Semiconductors. The Frontiers Collection, 161–174. Springer, Berlin. https://doi.org/10.1007/978-3-642-23096-7_7

Jiang, H., Han, J., Qiao, F., & Lombardi, F. (2016). Approximate radix-8 booth multipliers for low-power and high-performance operation. *IEEE Transactions on Computers*, *65*(8), 2638–2644. https://doi.org/10.1109/TC.2015.2493547

Kahng, A., & Kang, S. (2012). Accuracy-configurable adder for approximate arithmetic designs. *Proceedings Of The 49Th Annual Design Automation Conference On - DAC '12,* San Fransicso, California, USA. https://doi.org/10.1145/2228360.2228509

Maheshwari, N., Yang, Z., Han, J., & Lombardi, F. (2015). A design approach for compressor based approximate multipliers. *2015 28th International Conference on VLSI Design,* Bangalore, Karnataka, India. https://doi.org/10.1109/vlsid.2015.41

Masadeh, M., Hasan, O., & Tahar, S. (2019). Input-conscious approximate multiply-accumulate (MAC) Unit for Energy-Efficiency. *IEEE Access*, *7*,147129–147142. https://doi.org/10.1109/access.2019.2946513

Mazahir, S., Hasan, O., Hafiz, R., Shafique, M., & Henkel, J. (2016). An area-efficient consolidated configurable error correction for approximate hardware accelerators. *Proceedings of the 53rd Annual Design Automation Conference on - DAC '16,* Austion, Texas, USA. https://doi.org/10.1145/2897937.2897981

Momeni, A., Han, J., Montuschi, P., & Lombardi, F. (2015). Design and analysis of approximate compressors for multiplication. *IEEE Transactions on Computers*, *64*(4), 984–994. https://doi.org/10.1109/tc.2014.2308214

Popkin, T., Cavallaro, A., & Hands, D. (2010). Accurate and efficient method for smoothly space-variant gaussian blurring. *IEEE Transactions On Image Processing*, *19*(5), 1362–1370. https://doi.org/10.1109/tip.2010.2041400

Reddy, K., Nithin Kumar, Y., Sharma, D., & Vasantha, M. (2015). Low power, high speed error tolerant multiplier using approximate adders. *2015 19Th International Symposium On VLSI Design And Test,* Ahmedabad, Gujarat, India. https://doi.org/10.1109/isvdat.2015.7208150

Shafique, M., Hafiz, R., Rehman, S., El-Harouni, W., & Henkel, J. (2016). Invited - Cross-layer approximate computing: From logic to architectures. *Proceedings of the 53rd Annual Design Automation Conference on - DAC '16,* Austin, Texas, USA. https://doi.org/10.1145/2897937.2906199

Smith, T., Marks, W., Lange, G., Sheriff, W., & Neale, E. (1988). Edge Detection in Images Using Marr-Hildreth Filtering Techniques. *Journal of Neuroscience Methods*. *26*(1), 75–81. https://doi.org/10.1016/0165-0270(88)90130-6

Swartzlander, E. (1999). Truncated multiplication with approximate rounding. *Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers (Cat. No.CH37020),* Pacific Grove, California, USA. https://doi.org/10.1109/acssc.1999.831996

Unser, M., & Eden, M. (1989). Multiresolution feature extraction and selection for texture segmentation. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, *11*(7), 717–728. https://doi.org/10.1109/34.192466

Vasicek, Z., & Sekanina, L. (2015). Evolutionary approach to approximate digital circuits design. *IEEE Transactions On Evolutionary Computation*, *19*(3), 432–444. https://doi.org/10.1109/tevc.2014.2336175

Ye, R., Wang, T., Yuan, F., Kumar, R., & Xu, Q. (2013). On reconfiguration-oriented approximate adder design and its application. *2013 IEEE/ACM International Conference On Computer-Aided Design (ICCAD),* San Jose, California, USA. https://doi.org/10.1109/iccad.2013.6691096

Zhu, N., Goh, W. L., Wang, G., & Yeo, K. S. (2010). Enhanced low-power high-speed adder for error-tolerant application. *2010 International SoC Design Conference,* Incheon, South Korea. https://doi.org/10.1109/socdc.2010.5682905