

Approximate single precision floating point adder for low power applications

Manjula Narayanappa¹, Siva Sankar Yellampalli²

¹Department of Electronics and Communication, Dr. Ambedkar Institute of Technology, Bengaluru, India

²Department of Electronics and Communication, SRM University, Amravati, India

Article Info

Article history:

Received Dec 2, 2022

Revised Jun 5, 2024

Accepted Jul 3, 2024

Keywords:

Approximate adders

Delay

Exact computing

Power consumption

Single precision floating point adder

ABSTRACT

With an increasing demand for power-hungry data-intensive computing, design methodologies with low power consumption are increasingly gaining prominence in the industry. Most of the systems operate on critical and non-critical data both. An attempt to generate a precision result results in excessive power consumption and results in a slower system. For non-critical data, approximate computing circuits significantly reduce the circuit complexity and hence power consumption. In this paper, a novel approximate single precision floating point adder is proposed with an approximate mantissa adder. The mantissa adder is designed with three 8-bit full adder blocks. In this paper, a detailed mathematical background, and proposed design approach in terms of the circuit configuration and truth tables are discussed. Additionally, a concept of switching between exact computing and approximate computing is analysed considering an approximate carry look-ahead adder. The delay and power consumption for the exact operating mode and approximate operation mode considering varied window sizes is observed. Performance of the approximate computation is compared against exact computation and varied approximate computing approaches.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Manjula Narayanappa

Department of Electronics and Communication, Dr. Ambedkar Institute of Technology

Bengaluru, Karnataka, India

Email: manjulan_12@rediffmail.com

1. INTRODUCTION

Battery-operated portable electronic devices have increasingly become an indispensable part of everyday life. The key behind this is the scaling ability of metal-oxide-semiconductor field-effect transistor (MOSFETS) seen in very large-scale integration (VLSI) due to which functionality per unit area has increased which has brought the price of the devices down leading to wide usage. Due to scaling and an increase in functionality per unit area, the power consumption has increased. Even though massive developments and innovations are observed in the last few years in digital integrated circuits, high power consumption remains the main issue. Due to the massive demand for performance-oriented circuits, power consumption becomes an important feature for appropriate circuit design. However, the increase in power consumption of the VLSI devices has not been matched by the improvement in the capacity of the battery. Therefore, operation time per charge has come down causing inconvenience to the users. For this reason, reducing the power consumption of portable devices has become a compelling design constraint. Furthermore, achieving desired performance trade-off between energy efficiency and reliability in portable digital processing systems and power circuits has become a massive challenge. To build an arithmetic and logical unit, adders are considered a key building block. Adders also provide massive support to varied

operations like multiplication, division, and subtraction and are considered one of the most power-hungry circuit components and remained in hot-spot locations [1].

Here, a large portion of energy consumption is dominated by two components: dynamic power and leakage power. To extend the battery life various technology-based, architecture-based, and circuit-based solutions that reduce the sum of the two power components without sacrificing the performance have to be developed. As a result, based on extensive research, varied design methods are presented over the years to meet power, and speed design requirements. Among them, one of the most emerging approaches is approximate computing to handle the market demands and maintain performance with high power efficiency. However, computational accuracy gets affected in the lowest manner [2]. This approach is built specifically for the application in which the set of approximate answers is acceptable.

At the technology level, feature size scaling has continuously brought lower power circuits by reducing the supply voltages. To retain performance, the threshold voltages of these circuits have also been reduced with technology scaling. However, in recent technologies, the benefits of constant-field scaling have been compromised by an exponential increase in the leakage current. On the architectural level, pipelining and parallelism have helped in lowering the power consumption of digital circuits. Besides, in the current complementary metal-oxide-semiconductor (CMOS) technology, the benefits of device scaling are impeded by reliability issues due to process variations, aging effects, and soft errors. Leakage current and static power are increasingly adding to the concerns about achieving low power consumption. Hence, the device scaling which once used to offer advantages for low-power applications is no more attractive and hence, new architectures need to be evolved to achieve low power consumption. Thus, a new paradigm, the design of approximate computation blocks is introduced to help in providing the simplification of the arithmetic unit circuits [3]. Many researchers have provided their efforts in designing approximate adders with varied structures [4]–[14]. However, most of the provided adders are approximate and massively advantageous for the applications like error resilient. On the other side, in these proposed adders a constant level of deviation is observed from the actual result. This shows that at the time of system processing, the accuracies are not tunable. Therefore, accuracy re-configurability can be an important and advantageous feature at the time of processing (Runtime) with varying levels of quality of service at the time of operations [4]–[6]. By compromising quality, the performance of computational time and power consumption can be minimized. As a result, energy efficiency can be improved.

Furthermore, most of the modern graphics processors for multimedia and other applications have dedicated digital signal processing (DSP) blocks. These applications output an image, video, or audio signal and the limited perception of human senses allows for an approximation of the computations involved in the demanding DSP algorithms for these applications [2]. Even an analog computation that yields good enough results instead of accurate results are also acceptable [4]. The addition is the most fundamental and significant mathematical operation used in all signal/ image processing applications [5], [6]. Deterministic approximate logic or probabilistic imprecise arithmetic are normally employed for soft adders [7]. In addition, Low-power designs through approximate computing have been proposed using Algorithmic Noise Tolerance [15], [16], non-uniform voltage over scaling [17], and significance driven computation [18], [19]. Verma *et al.* [7] proposed a novel adder design that is exponentially faster than traditional adders called an almost correct adder (ACA) and a variable latency speculative adder (VLSA) with an area overhead. Other adder configurations meet the real-time energy requirements by complexity reduction at the algorithm level [20], [21]. The lower part or adder [22] is based on an approximate logic with a different truth table than that of an original adder. Probabilistic full adder (PFA) [23], [24] is based on probabilistic CMOS, a technology platform for modeling the behavior of nano-metric designs as well as reducing power consumption.

Additionally, for both exact computing and approximate computing, general-purpose processors are also utilized in some digital systems. In these processors, mandatorily dynamic switching capabilities are induced to switch between exact and approximate computing. A correction unit can be added to the design circuit for incorporating this feature so that switching between approximate and exact computing can be performed easily. However, the delay, power, and design area overhead can be increased by the incorporation of a correction unit. On the other side, processing can be slowed down by the adoption of an error correction circuit, which requires more than one clock cycle [7], [8].

Therefore, this paper highlights the designing of an approximate adder and the performance comparison of an approximate adder with exact computing. Furthermore, a novel approximate single-precision floating-point adder is proposed with an approximate mantissa adder. The mantissa adder is designed with three 8-bit full adder blocks. Additionally, in this paper, an approximate carry look-ahead adder is presented which works based on the concept of switching between exact and approximate computing. Thus, the design circuit has an exact and approximate operating mode. The proposed adder structure does not require any kind of error correction unit and works on the principle of the traditional carry look-ahead adder for switching between exact and approximate computing. The delay and power consumption for the exact operating mode in the proposed adder methodology is kept similar to the

traditional carry look-ahead adder. However, in approximate computing, the delay and power consumption are relatively lesser than in exact computing. The power consumption is significantly minimized in the approximate operating mode by exploiting the power gating technique. The performance of the proposed 8-bit approximate adder is evaluated in two different parts in which first part concentrates on obtaining metrics results like delay, error metrics, and power consumption trade-off in terms of truth tables, circuits, performance values, and results are compared with exact computing outputs. In the second part, outputs are compared against varied previous approximate paradigms in terms of error percentage, mean error detection, delay, power consumption, and normalized error detection considering different window sizes.

This article is presented in the following manner. Section 2, describes related work and discusses the design challenges of the approximate adder, and section 3 describes the methodology to design approximate adders for low-power applications. Section 4 describes the experimental results and section 5 concludes the paper.

2. RELATED WORK

Despite the massive advancements in varied semiconductor technologies, low-power design applications, and power-optimization methodologies, several computing devices, and processing machines require high power to handle large-data processing and computations. This problem becomes more challenging in the case of mobile and internet of things (IoT) devices or battery-oriented systems. Thus, several researchers have shown great interest over the years to provide power-handling mechanisms in data processing layers, especially for these mentioned devices [22], [25]. Data is gathered from varied computing devices from various domains in huge numbers and analysis of those data is challenging. However, techniques like data mining, synthesis, and some recognition or analysis applications, error-resistant applications are used to understand meaningful data [2]. One of the approaches to handle this kind of issue is inexact solutions such as approximate computing. This approach has massively emerged in the last few years to minimize high power consumption and enhance system efficiency [26]. This approach can be utilized in many areas at varying levels such as in numerous devices, hardware systems, software, programming languages, algorithms, design architectures, and circuit designs. This approximate computing paradigm can be useful for the development of automated design and assessment. The base of approximate computing is its design rules like significance-guided design. This can be extended to a specific level of design based on a particular application.

Due to massive interest from the electronic and semiconductor industries, the approximate computing approach has come out as one of the most widely adopted power control paradigms in recent years. Thus, a massive number of researchers have provided detailed analyses and studies based on the generalized significance-guided design principles with concentrates on fewer resource utilization and lower computational complexity. Based on this principle, voltage scaling in CMOS design and logic circuit reduction is applied. Providing high supply voltages to the critical circuits can minimize power consumption in Probabilistic CMOS VLSI circuits and accuracy management of the most significant bits is also ensured. At the same time, the supply voltage is lowered for the least important bits to save power. In approximate computing, accuracy may be compromised compared to exact computing but in the least manner. For an instance, using a conventional adder, a probabilistic adder is constructed whose supply voltage depends upon the level of importance [9]. However, implementation cost can be a massive issue in this approach due to the complexity of supply voltage control. Thus, most of the approximate computing circuits are focused on logic reduction design and the pruning method can be used to control and implement.

Furthermore, floating-point adders have captured large attention in recent years. Liu *et al.* [27] have presented a floating-point adder design to improve critical path delay and enhance the efficiency of image processing applications. Camus *et al.* [28] have presented an approximate floating-point adder for image-processing applications by combining an inexact speculate adder approach and gate-level pruning methodology to build an approximate multiplier architecture. In this paper, three approximate units are presented. This unit utilizes tone mapping to enable high dynamic-range images. Yan *et al.* [29] have presented an approximate floating-point multiplier for the customization of the machine-learning framework based on the k-nearest neighbors (KNN) algorithm and this concept is used for the application of handwritten digit recognition. The computing processors where large data processing is required, high power is dedicated to those processing blocks. The main processing blocks for adder design are arithmetic and logic units (ALUs) [30]. Thus, overall power consumption may improve by minimizing the necessary power in an adder unit.

3. PROPOSED METHOD FOR APPROXIMATE DESIGN

3.1. Basic building block: 8-bit approximate adder

The carry equation for a conventional carry look-ahead adder is given by (1). Where M_{in} is the input carry and R_k and L_k propagate and generate signals of the k^{th} stage. If the carry equation is split up into two segments, as in (2).

$$M_{k+1} = L_k + L_{k-1}R_k + \dots + L_0 \prod_{d=1}^k R_d + M_{in} \prod_{d=0}^k R_d \quad (1)$$

$$M_{k+1} = \left(\sum_{d=k-N+1}^i L_d \left(\prod_{c=d+1}^{k-1} R_c \right) \right) + \left(\sum_{d=0}^{k-N} L_d \left(\prod_{c=d+1}^{k-1} R_c \right) + M_{in} \prod_{d=0}^k R_d \right) \quad (2)$$

Where, N is the window size, the first segment consists of N most significant (MS) bits and the second segment consists of $N-W$ least significant (LS) bits. The first part of (2) is the approximate part, while the second part is called the augmenting part. For approximate carry generation with a window size of N , the output carry at the k^{th} stage is computed using the approximate part only. Computing an approximate M_{k+1} is faster and consumes fewer hardware resources and hence lesser power as compared to computing precise carry.

It is explained in varied traditional methods that the M_{k+1} processing of approximate computing is relatively faster with minimum power consumption than the M_{k+1} processing of exact computing. In the proposed reconfigurable adder, two types of operating modes are discussed such as exact and approximate operating modes based on this M_{k+1} processing. However, only one multiplexer is utilized in the proposed computing methodology M_{k+1} than the traditional computing methods. Multiplex consists of input and output mechanisms in which input signals are approximate and output signals are considered as augmented carry signals as the selector. The carry output generation is computed using operating mode signals in either the exact computing or approximate computing. The gate-level computation is performed using the proposed methodology as demonstrated in Figure 1. The power consumption of augmenting region is handled using the power-gated p-channel metal-oxide semiconductor (p-MOS) headers when the carry computation block is in the approximate computing. All the computing signals of processing units are handled using only one signal when all the adder carry is generated using the proposed design structure and methodology. However, all the adder output bits may remain imprecise. The approximate adder accuracy is enhanced by utilizing the MS bit group of the exact carry generator as demonstrated in [10] and [20]. This approach can be used in designing the carry look ahead adder with multiple accuracy levels. Adders are subdivided into a few different segments for the approximate and exact operating modes to design adjustable accuracy composed structures. Thus, in this report, efficiency computation for approximate and exact computing is discussed.

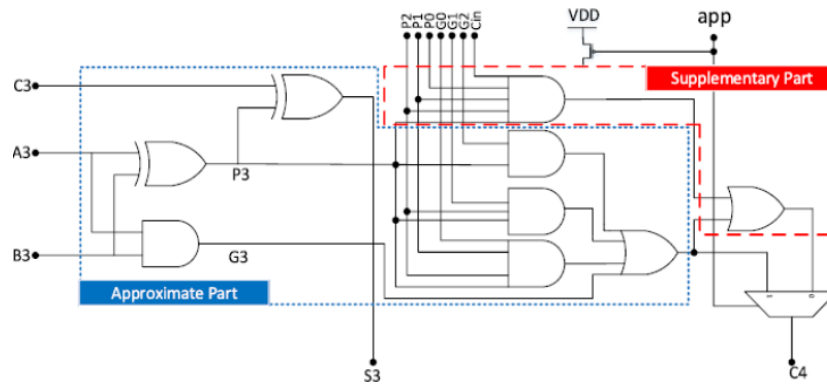


Figure 1. Carry generator for 4-bit block

3.2. Accuracy analysis of approximate adder

The approximate block accuracy is studied based on the error metrics and analytical expressions are analyzed using the proposed adder methodology. This section discusses the analysis of approximate adder accuracies compared with the traditional approximate adders. In the proposed ripple carry adder design, the generated approximate carry M_k decides adder accuracy for every bit position. The $\beta(M_k)$ function evaluates the exactness value of the approximate carry M_k for $k - \text{th}$ position based on the input signals. $\beta(M_k)$ function is evaluated using the following function given in (3). Where n is the window size and $\beta(M_k) \in$

{0,1}. Then, the error probability of the $k - th$ obtained approximate carry is evaluated by the (4). Then, the error rate for the proposed model considering the n -bit ripple carry adder is evaluated as given in (5).

$$\beta(M'_k) = \prod_{d=k-n}^{k-1} R_d \times [\sum_{m=-1}^{k-n-1} L_m \prod_{c=m+1}^{k-n-1} R_c - L_{k-n-1}] \quad (3)$$

$$R(M'_k) = \left(\frac{1}{2}\right)^{n+1} \sum_{d=0}^{k-n-2} \left(\frac{1}{4}\right)^{k-n-d-1} \quad (4)$$

$$R(\text{error}(n, s)) = \sum_{c=n+1}^s (-1)^{c+1} (\sum_{n+1 \leq k_1 < \dots < k_c \leq s} |R(M'_{k_1}) \cap \dots \cap R(M'_{k_c})|) \quad (5)$$

After error probability estimation, error detection is an important aspect of the accuracy analysis for the proposed adder mechanism. Using (3), the error detection metric for the proposed approximate adder mechanism is given by the (6).

$$ED(X, Y, n, s) = \sum_{k=n+1}^{s-1} 2^k (-1)^{R_k} \beta(M'_k) + 2^s \beta(M'_s) \quad (6)$$

Then, the normalized error detection metric is given by the (7),

$$NED = \frac{1}{2^{2s}} \sum_{k=1}^{2^{2n}} \frac{ED_k}{I} \quad (7)$$

where the maximum error value is given by I . Then, the average relative error detection metric for approximate adder is given by the (8).

$$ARED = \frac{1}{2^{2s}} \sum_{k=1}^{2^{2n}} \frac{|ED_k|}{G_k} \quad (8)$$

3.3. IEEE-754 floating point format

A floating-point representation provides a higher dynamic range than a fixed-point representation of real numbers. The floating-point hardware is both complex and consumes significant power. The most commonly used standard for the floating point (FP) format is the IEEE 754-2008 [31]. There are basic and extended types that are supported by this standard: half-precision (16 bits), single precision (32 bits), double precision (64 bits), extended precision (80 bits), and quad precision (128 bits). A general IEEE FP format is shown in Figure 2. The exponent part has a bias of $2^{E-1}-1$, where E is the number of exponent bits. The single-precision and double-precision formats are mostly used in today's computers. Table 1 demonstrates exponent and mantissa bits for IEEE-754 basic and extended floating-point types.

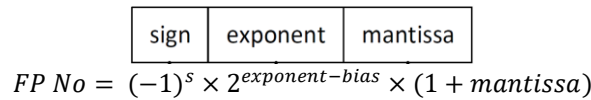


Figure 2. General IEEE-754 floating-point format

Table 1. Exponent and mantissa bits for IEEE-754 basic and extended floating point types

Type	Sign bit	Exp. bits	Mant. bits	Total	Mant. bits/total
Half	1	5	10	16	62.50%
Single	1	8	23	32	71.9%
Double	1	11	52	64	81.2%
Extended	1	15	64	80	80.0%
Quad	1	15	112	128	87.5%

3.4. Floating point adder architecture

A generic FP adder architecture includes hardware blocks for exponent comparison, mantissa alignment, mantissa addition, normalization, and rounding of the mantissa (shown in Figure 3 and detailed in [30], [32], [33]). Two operands are first unpacked from the FP format, and each mantissa is added to the hidden '1' bit. The addition of FP numbers involves comparing the two exponents, and adding the two mantissae; the exponents are first evaluated to find the larger number. The mantissa is then swapped according to the exponent comparison; they are then aligned to have an equal exponent prior to the addition

in the mantissa adder. Following the addition, normalization shifts are required to restore the result to the IEEE standard format. The normalization is completed by left shifting with a number of leading zeros; therefore, leading zero detection is a key step for normalization. Rounding the normalized result is the last step before storing back the result; special cases (such as overflow underflow, and not a number) are also detected and represented by flags.

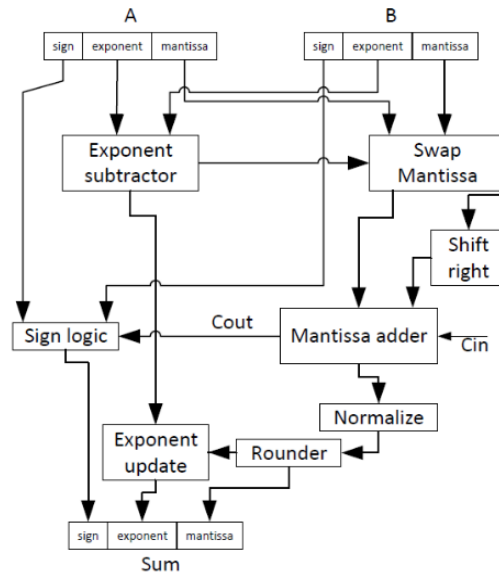


Figure 3. Floating point adder algorithm

3.5. Approximate floating-point adder

The approximate FP adder design originates at the architecture level with the exponent and mantissa adder/subtractor designed using approximate fixed-point adders. An *N-bit* adder consists of two parts, *i.e.*, an *m-bit* exact adder and an *n-bit* inexact adder (Figure 4). The exact adder part can have the exact implementation as a full adder circuit. The inexact adder will ignore the carry bits for computation thereby reducing the critical path as well as the hardware utilization. The modified approximate adder concept can also be used for the mantissa adder for approximate computation. The mantissa adder will provide a larger scope, as the number of bits in the mantissa are higher than the exponent and at the same time, the approximate design in the mantissa adder has a lower impact on the error, because the mantissa part is less significant than the exponent part. Therefore, an inexact design of a mantissa adder is more appropriate.

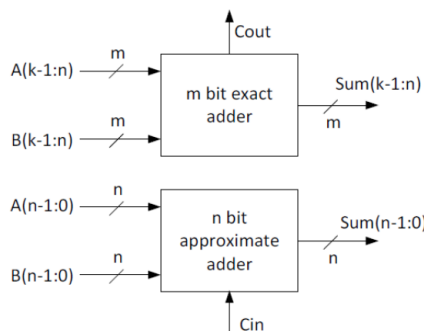


Figure 4. Approximate adder concept

An 8-bit adder is chosen as the basic building block for the FP approximate adder in the proposed design. As shown in Figure 5, the 8 bits are partitioned into two blocks, the MS block is of 4 bits, while the LS block is of 4 bits. Figure 6 shows the schematic diagram of the conventional full adder.

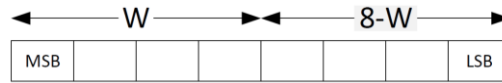


Figure 5. W-bit window for approximate computation

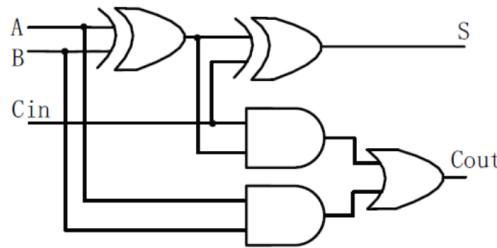


Figure 6. Conventional full adder

In the proposed adder, for the LS, W-bit window sum and carry are computed as per (9). The schematic of 1-bit full adder in the proposed configuration is given in Figure 7. The sum and carry for most significant (8-W=4) bits are computed as for an exact adder given in (10). The truth table for the proposed sum and carry equations is given in Table 2.

$$C_{i+1} = a_i b_i + c_{in}$$

$$S_i = \underline{C}_{i+1} \tag{9}$$

$$C_{i+1} = a_i b_i + c_{in}(a_i \oplus b_i)$$

$$S_i = (a_i \oplus b_i \oplus c_{in}) \tag{10}$$

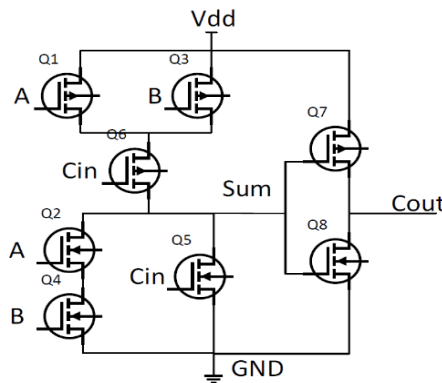


Figure 7. Schematic for carry and sum generator of proposed adder

Table 2. Truth table for the proposed sum and carry

A	B	C _{in}	Proposed S	Proposed C _{i+1}	Exact S	Exact C _{i+1}
0	0	0	1	0	0	0
0	0	1	0	1	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	0	1	0
1	0	1	0	1	0	0
1	1	0	0	1	0	0
1	1	1	0	1	1	1

Overall, 3 errors are introduced in sum computation and 1 error in carry computation. Assigning the inverted carryout at each stage to the sum computed for that stage reduces the hardware for the sum computation block. This is a significant reduction in hardware requirements as compared to a conventional adder. Utilizing the look-ahead carry generation logic from 4 MS bits improves the timing performance of the circuit by not depending on the sequential computation of carrying at each bit. A total of 8 transistors are used for 1-bit sum and carry generation.

3.6. Mantissa approximate adder

For realizing the 23-bit approximate adder, three 8-bit adders are used. The lower two 8-bit adders are the proposed 8-bit approximate adders, while the MS byte is implemented using an exact 8-bit adder. The proposed 23-bit mantissa adder is shown in Figure 8.

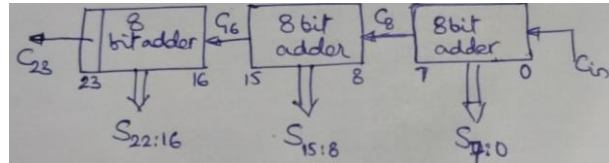


Figure 8. Proposed 23 bit mantissa adder

3.7. Exponent adder/subtractor

As the exponent for realizing the 23-bit approximate adder, is having the most impact on the accuracy of the result. The exponent adder is proposed to be implemented using an exact 8-bit adder. The maximum ED for the 8-bit adder is 3.

4. RESULT AND DISCUSSION

The proposed adder circuit is simulated in a Cadence environment for delay and power consumption and error analysis. The results are presented hereby. Several metrics are proposed to elaborate the performance discussion.

4.1. Performance discussion for approximate floating-point adder

4.1.1. Error metrics for proposed 8-bit adder

The proposed 8-bit adder with a window size of $W=4$ is simulated for all possible input combinations of a and b . For all the 256×256 combinations, the approximate and the exact sum and carries are computed, and error distances are computed between the approximate and accurate outputs. The maximum ED for the 8-bit adder is 3.

4.1.2. Delay

Considering a conventional 8-bit adder, the delay in 8-bit computation is due to the ripple carry effect, which takes 8 cycles. Assuming the delay in the computation of the 1-bit full adder result to be T , the delay in generating the 8-bit adder is $8T$. In the proposed adder, the total delay is equal to the delay for computation of carry out from MS 4-bits, which is equal to $4T$.

4.1.3. Power consumption tradeoff

The energy consumed by a probabilistic inverter increases exponentially with the probability of correct output. The power consumption is considered proportional to the number of gates in an approximate implementation. In the proposed adder circuits, the reduction in the number of transistors allows a lower operating voltage reducing the overall power consumption. For a conventional full adder, if the power consumed for 1-bit is normalized to 1, then the power consumed for a k -bit conventional full adder is k . For the proposed 8-bit adder, the operating voltage is reduced to 1.04 V from 1.13 V for accurate implementation due to a reduction in the number of transistors. Hence, the estimated power consumption is evaluated by (11),

$$E_{8-bit} = W \times \frac{1.04^2}{1.13^2} + (8 - W) = 4 \times \frac{1.04^2}{1.13^2} + 4 \quad (11)$$

which is 7.5% lower than the conventional adder.

A detailed analysis of full adder truth table is carried out and observed that $Sum = \underline{C}_{out}$ for 6 times out of all possible 8 combinations, except when the value of A, B, C_{in} is kept at 0 and 1 for all, respectively. Further, in the traditional approximate adder as shown in Figure 9, \underline{C}_{out} is evaluated in the first stage. Then, one way to simplify the traditional approximate adder is by discarding the Sum circuit. Thus, if $Sum = \underline{C}_{out}$ is set in the traditional approximate adder, then the amount of capacitance present in the Sum circuit will be a mixture of 4 source-drain diffusion and 2 gates capacitance. As a result, there is a massive increase in terms of total capacitance compared to the traditional approximate adder. However, it will cause a delay where two or more approximate adders are cascaded with each other. As mentioned, that $Sum = \underline{C}_{out}$ for 6 times out of all possible 8 combinations, thus the simplified approximate adder is cascaded for \underline{C}_{out} as demonstrated in Figure 10. Moreover, Figure 11 shows the simplified approximate adder circuit using the mentioned approach. As a result, 3 errors in Sum and 1 error in \underline{C}_{out} is generated as demonstrated in Table 3.

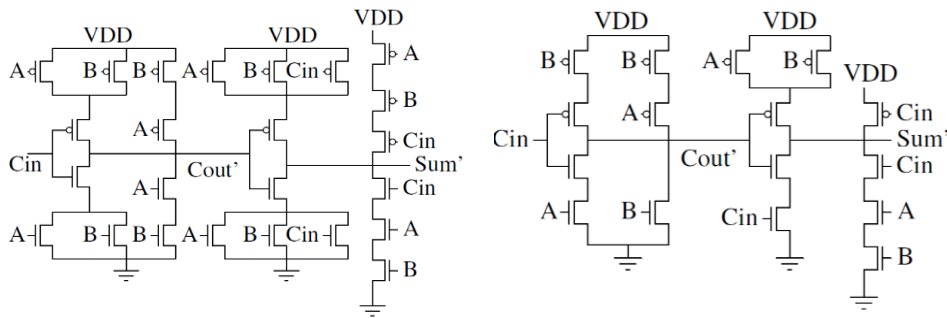


Figure 9. Traditional approximate adder

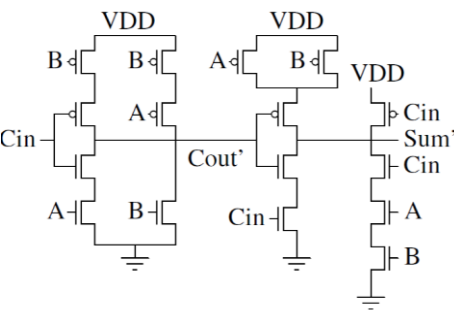


Figure 10. Simplified approximate adder

Table 3. Truth table for traditional full adder and approximations

Inputs			Exact outputs		Approximate outputs					
A	B	C _{in}	Sum	C _{out}	Sum ₁	C _{out1}	Sum ₂	C _{out2}	Sum ₃	C _{out3}
0	0	0	0	0	1	0	0	0	0	0
0	0	1	1	0	1	0	1	0	0	0
0	1	0	1	0	0	1	0	0	1	0
0	1	1	0	1	0	1	1	0	1	0
1	0	0	1	0	1	0	0	1	0	1
1	0	1	0	1	0	1	0	1	0	1
1	1	0	0	1	0	1	0	1	1	1
1	1	1	1	1	0	1	1	1	1	1

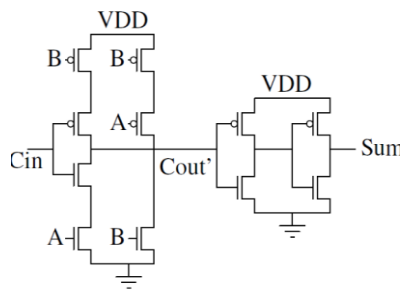


Figure 11. simplified approximate adder circuit

Both the discrete cosine transform (DCT) and inverse discrete cosine transform (IDCT) blocks operates at a lower supply voltage in case of approximate adders than the exact mode. Here, DCT and IDCT operates at a supply voltage of 1.28 V and 1.13 V in the exact mode, respectively. The different supply operating voltages are demonstrated in Figures 12 and 13 for different approximations and truncations considering varied bits. Table 4 demonstrates the percentage power savings considering varied approximations and truncation against the base case. Approximation 3 saves the maximum power.

Table 4. Percentage power savings for approximations over the base case

Technique	7 LSB's	8 LSB's	9 LSB's
Truncation	48.22	56.23	61.24
Approximation 1	37.86	50.85	55.26
Approximation 2	41.21	49.13	53.84
Approximation 3	42.46	52.64	59.23

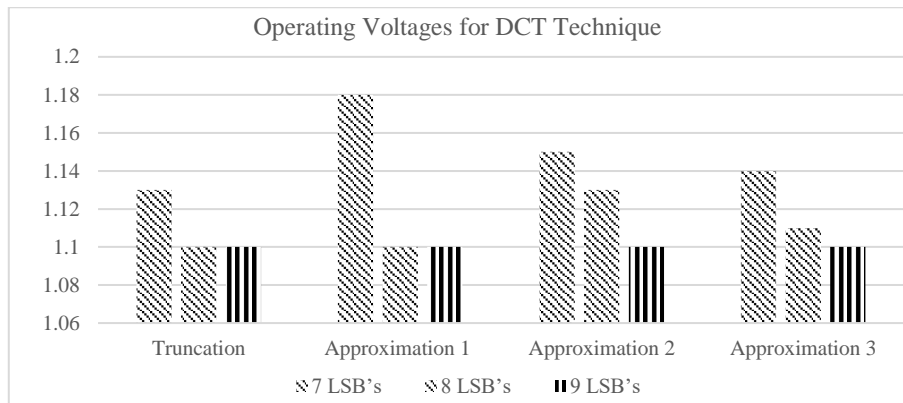


Figure 12. Operating voltages considering different bits for DCT technique

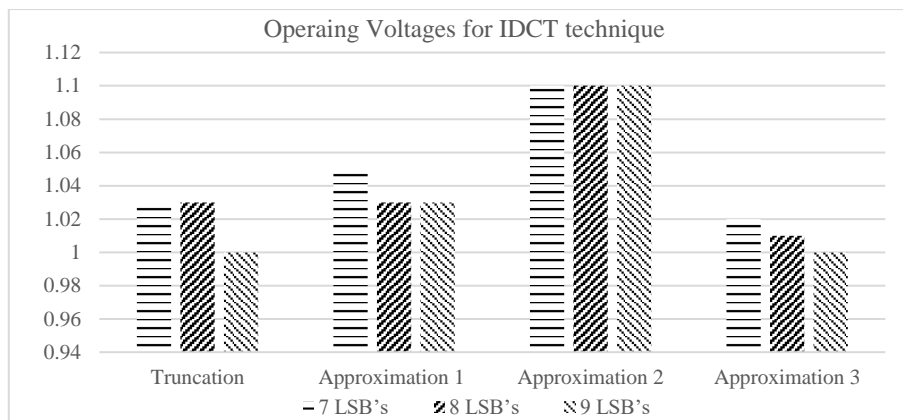


Figure 13. Operating voltages considering different bits for IDCT technique

4.2. Performance metrics comparison for approximate carry look-ahead adder

Here G_k is defined as the accurate result for the $k - th$ input set. The performance of the proposed adder design is compared with previous works such as [23], [24] in terms of performance metrics like percentage error rate, average error detection, and normalized error detection. The adder design structure presented in [24] is called generic accuracy configurable (GeAr), and in [23] is called ethylene recovery unit (ERU). However, the ERU design structure is segregated into two design structures i.e. with an error reduction unit and without an error reduction unit. Their detailed description is presented in [23]. The proposed adder design structure is studied considering 8-bit added for varied window sizes while in [23], the window size is fixed as $2k$. Figure 14 shows the proposed 8-bit approximate adder design comparison in terms of error rate in percentage against GeAr considering different window sizes. It is evident from Figure 14 results that the proposed accuracy of the proposed approximate adder design is higher than the [24]. Other possible combinations regarding error metrics for performance comparison are average error detection and normalized error detection.

Figure 15 shows the proposed 8-bit approximate adder design comparison in terms of average error detection against [23], [24] considering three different window sizes. It is evident from Figure 15 results that the proposed adder design has a slightly lower average error detection value than GeAR and similar values to the ERU design structure in case of without the use of an error reduction unit. However, the least average error detection values are observed when an error reduction unit is utilized. Figure 16 shows the proposed 8-

bit approximate adder design comparison in terms of normalized error detection against [23], [24] considering three different window sizes. It is evident from Figure 16 results that the proposed adder design has slightly lower normalized error detection values than GeAR and similar values to the ERU design structure in cases of without and with the use of an error reduction unit. Thus, in terms of normalized error detection values, the proposed adder design performs well.

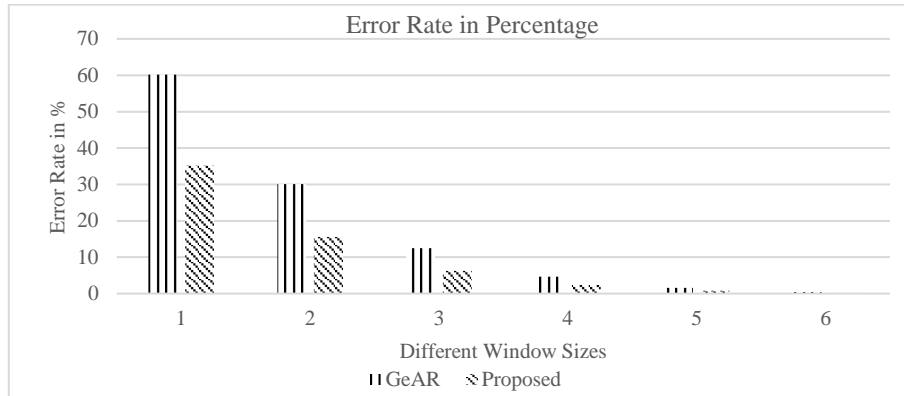


Figure 14. Error rate (%) comparison of proposed adder vs GeAR

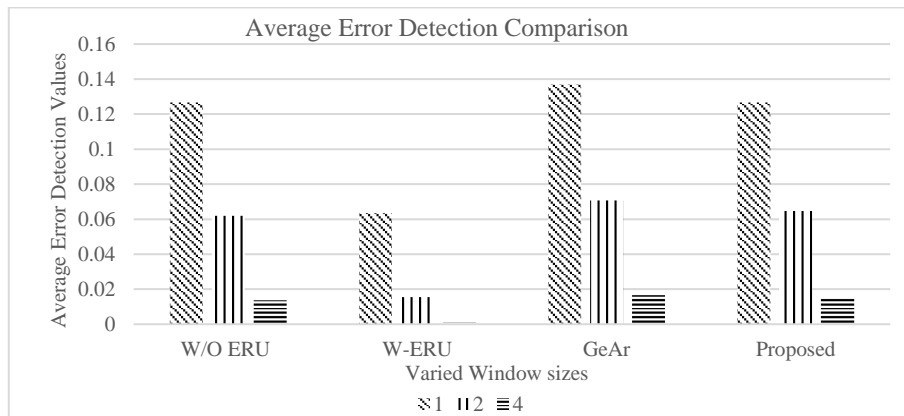


Figure 15. Average error detection comparison of proposed adder against varied adder designs

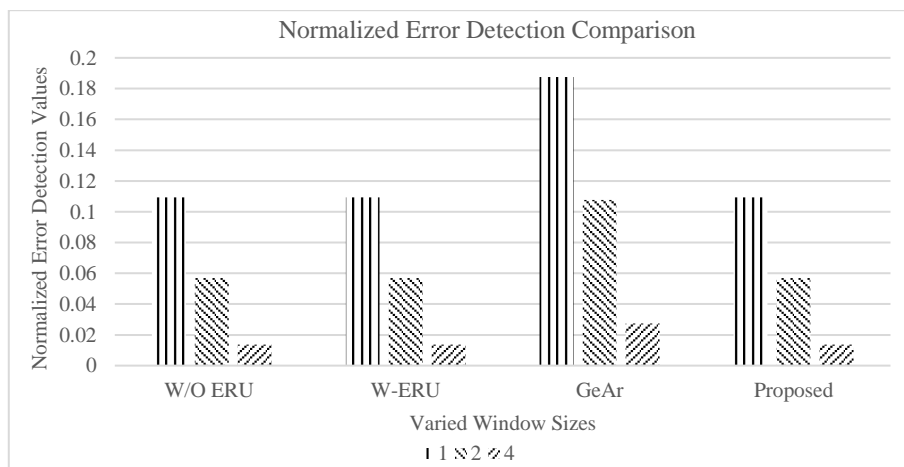


Figure 16. Average error detection comparison of proposed adder against varied adder designs

Figure 17 demonstrates the performance results for the proposed 8-bit approximate adder design against varied adder designs in terms of delay (ps). It is visible from Figure 17 that an increase in the window size will increase delay for all the adder designs. This shows parameter values are different for different window sizes. However, the proposed approximate adder design shows the least delay among all the approximate adder designs.

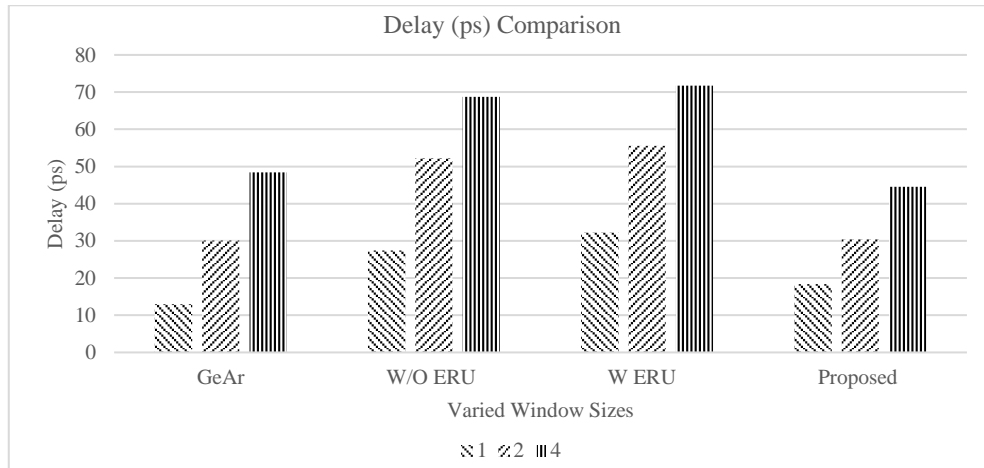


Figure 17. Delay comparison considering proposed adder design against varied approximate adder designs

Figure 18 demonstrates the performance results for the proposed 8-bit approximate adder design against varied adder designs in terms of area (μm^2). It is visible from Figure 18 that area is different for different window sizes. However, the proposed approximate adder design requires the least area among all the approximate adder designs.

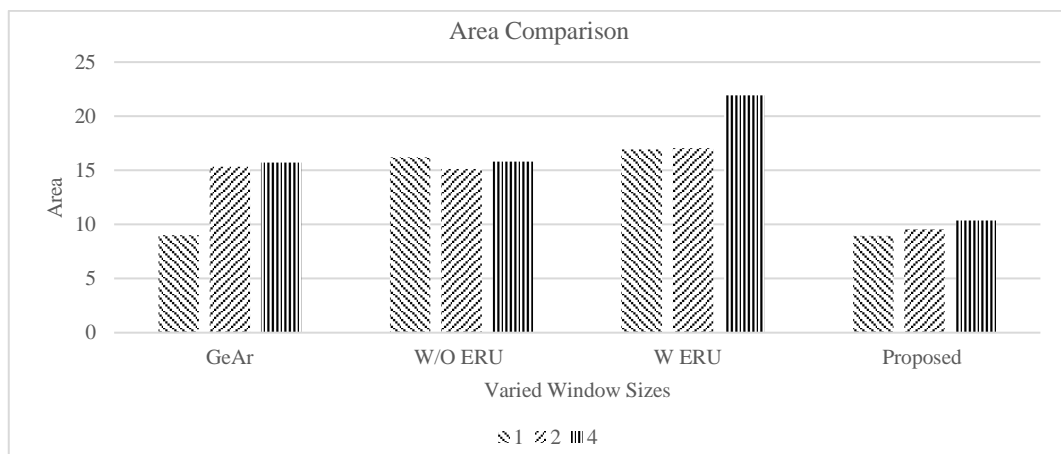


Figure 18. Area comparison considering proposed adder design against varied approximate adder designs

Figure 19 demonstrates the performance results for the proposed 8-bit approximate adder design against varied adder designs in terms of power (μW). It is visible from Figure 19 that the proposed approximate adder design requires the least power among all the approximate adder designs. Overall proposed approximate adder design shows superior results than the other approximate adder designs.

Figure 20 demonstrates the performance results for the proposed 8-bit approximate adder design against varied adder designs in terms of energy (aJ). It is visible from Figure 20 that the proposed approximate adder design requires the least energy among all the approximate adder designs. Overall proposed approximate adder design shows superior results than the other approximate adder designs.

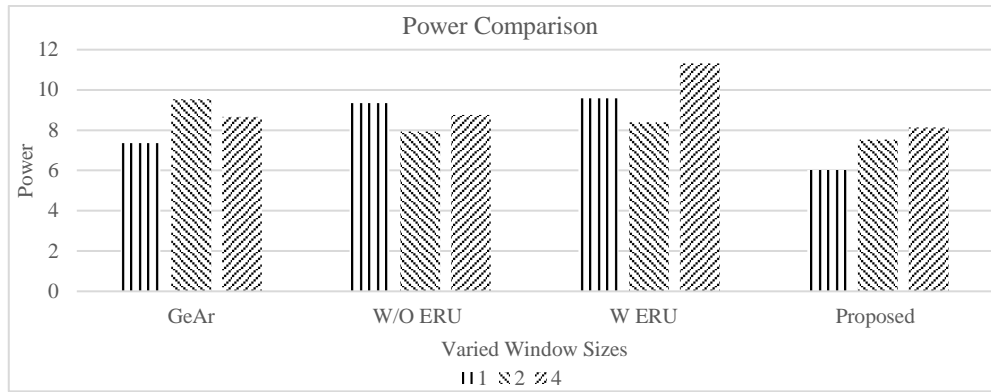


Figure 19. Power comparison considering proposed adder design against varied approximate adder designs

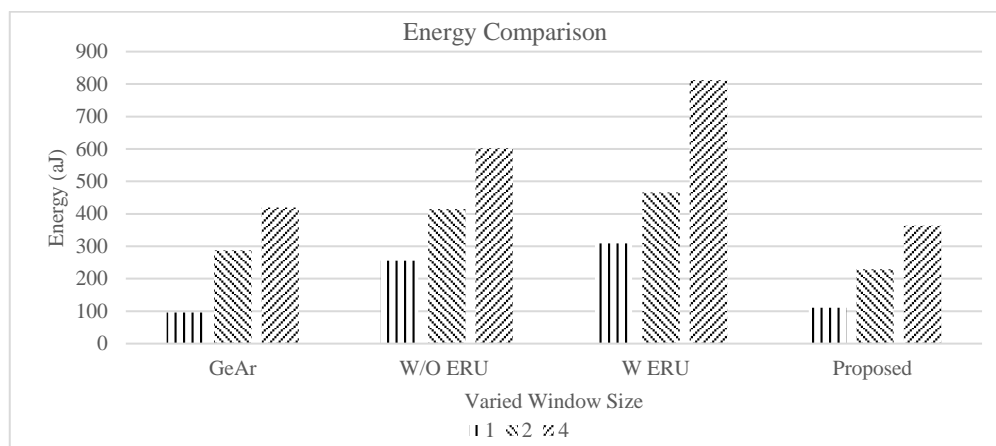


Figure 20. Energy comparison considering proposed adder design against varied approximate adder designs

5. CONCLUSION

A novel approximate adder topology for single point floating-point adder is presented in the paper. The proposed design takes advantage of the fact that the lower significant bit addition can be approximate and this will not be affecting the solution to a great extent, at the same time the power savings due to the approximate computation will be significant. The proposed configurations have a lower propagation delay and comparable error performance as compared to other architectures. With the proposed mantissa adder, which is a hybrid of look ahead, carry adder for the carry generation and the approximate adder for the sum, generation gives a distinct advantage in terms of power consumption as compared to the conventional full adder. In addition, the concept of switching between exact and approximate computing is also discussed and a performance comparison between exact and approximate computing is presented. It is evident from the performance results that in the case of an approximate adder considering the larger window size, the power savings due to the approximate computation will be significant with minimum delay. The proposed configurations have a lower propagation delay and comparable error performance as compared to other architectures. The accuracy performance difference between exact and approximate computing remains negligible. Therefore, approximate computing can be utilized instead of exact computing in future DSP applications. In future work, significant research will be performed to improve on-chip power efficiency in approximate computing as a practical mainstream computing paradigm.

REFERENCES





- [1] B. K. Mohanty and S. K. Patel, "Area-delay-power efficient carry-select adder," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 6, pp. 418–422, 2014, doi: 10.1109/TCSII.2014.2319695.
- [2] B. Shao and P. Li, "Array-based approximate arithmetic computing: a general model and applications to multiplier and squarer design," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 4, pp. 1081–1090, 2015, doi: 10.1109/TCSI.2015.2388839.

- [3] J. Han and M. Orshansky, "Approximate computing: an emerging paradigm for energy-efficient design," *Proceedings - 2013 18th IEEE European Test Symposium, ETS 2013*, pp. 1–6, 2013, doi: 10.1109/ETS.2013.6569370.
- [4] A. Raha, H. Jayakumar, and V. Raghunathan, "Input-based dynamic reconfiguration of approximate arithmetic units for video encoding," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 3, pp. 846–857, 2016, doi: 10.1109/TVLSI.2015.2424212.
- [5] M. S. Khairy, A. Khajeh, A. M. Eltawil, and F. J. Kurdahi, "Equi-noise: a statistical model that combines embedded memory failures and channel noise," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 2, pp. 407–419, 2014, doi: 10.1109/TCSI.2013.2268197.
- [6] R. Ye, T. Wang, F. Yuan, R. Kumar, and Q. Xu, "On reconfiguration-oriented approximate adder design and its application," *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD*, pp. 48–54, 2013, doi: 10.1109/ICCAD.2013.6691096.
- [7] A. K. Verma, P. Brisk, and P. Jenne, "Variable latency speculative addition: a new paradigm for arithmetic circuit design," *Proceedings - Design, Automation and Test in Europe, DATE*, pp. 1250–1255, 2008, doi: 10.1109/DATE.2008.4484850.
- [8] S. Beohar and S. Nemade, "VHDL implementation of self-timed 32-bit floating point multiplier with carry look ahead adder," *Proceedings of 2016 International Conference on Advanced Communication Control and Computing Technologies, ICACCCT 2016*, pp. 772–775, 2017, doi: 10.1109/ICACCCT.2016.7831743.
- [9] K. Palem and A. Lingamneni, "Ten years of building broken chips: the physics and engineering of inexact computing," *ACM Transactions on Embedded Computing Systems*, vol. 12, no. 2s, pp. 1–23, May 2013, doi: 10.1145/2465787.2465789.
- [10] A. Lingamneni, K. K. Muntimadugu, C. Enz, R. M. Karp, K. V. Palem, and C. Piguet, "Algorithmic methodologies for ultra-efficient inexact architectures for sustaining technology scaling," *CF '12 - Proceedings of the ACM Computing Frontiers Conference*, pp. 3–12, 2012, doi: 10.1145/2212908.2212912.
- [11] A. Jain, R. Jain, and J. Jain, "Design of reversible single precision and double precision floating point multipliers," *2018 International Conference on Advanced Computation and Telecommunication, ICACAT 2018*, 2018, doi: 10.1109/ICACAT.2018.8933712.
- [12] J. Y. F. Tong, D. Nagle, and R. A. Rutenbar, "Reducing power by optimizing the necessary precision/range of floating-point arithmetic," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 3, pp. 273–286, 2000, doi: 10.1109/92.845894.
- [13] A. Gupta *et al.*, "Low power probabilistic floating point multiplier design," *Proceedings - 2011 IEEE Computer Society Annual Symposium on VLSI, ISVLSI 2011*, pp. 182–187, 2011, doi: 10.1109/ISVLSI.2011.54.
- [14] F. Fang, T. Chen, and R. A. Rutenbar, "Floating-point bit-width optimization for low-power signal processing applications," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 3, 2002, doi: 10.1109/icassp.2002.5745332.
- [15] B. Shim, S. R. Sridhara, and N. R. Shanbhag, "Reliable low-power digital signal processing via reduced precision redundancy," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 5, pp. 497–510, 2004, doi: 10.1109/TVLSI.2004.826201.
- [16] G. V. Varatkar and N. R. Shanbhag, "Energy-efficient motion estimation using error-tolerance," in *ISLPED'06 Proceedings of the 2006 International Symposium on Low Power Electronics and Design*, Oct. 2006, pp. 113–118. doi: 10.1109/LPE.2006.4271817.
- [17] L. N. B. Chakrapani, K. K. Muntimadugu, A. Lingamneni, J. George, and K. V. Palem, "Highly energy and performance efficient embedded computing through approximately correct arithmetic: a mathematical foundation and preliminary experimental validation," in *Proceedings of the 2008 international conference on Compilers, architectures and synthesis for embedded systems*, Oct. 2008, pp. 187–196. doi: 10.1145/1450095.1450124.
- [18] D. Mohapatra, G. Karakonstantis, and K. Roy, "Significance driven computation: a voltage-scalable, variation-aware, quality-tuning motion estimator," in *Proceedings of the 2009 ACM/IEEE international symposium on Low power electronics and design*, Aug. 2009, pp. 195–200. doi: 10.1145/1594233.1594282.
- [19] N. Banerjee, G. Karakonstantis, and K. Roy, "Process variation tolerant low power DCT architecture," in *2007 Design, Automation & Test in Europe Conference & Exhibition*, Apr. 2007, pp. 1–6. doi: 10.1109/DATE.2007.364664.
- [20] Y. V. Ivanov and C. J. Bleakley, "Real-time H.264 video encoding in software with fast mode decision and dynamic complexity control," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 6, no. 1, pp. 1–21, Feb. 2010, doi: 10.1145/1671954.1671959.
- [21] M. Shafique, L. Bauer, and J. Henkel, "EnBudget: A run-time adaptive predictive energy-budgeting scheme for energy-aware motion estimation in H.264/MPEG-4 AVC video encoder," in *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*, Mar. 2010, pp. 1725–1730. doi: 10.1109/DATE.2010.5457093.
- [22] 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 Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 4, pp. 850–862, Apr. 2010, doi: 10.1109/TCSI.2009.2027626.
- [23] S. Cheemalavagu, P. Korkmaz, K. V. Palem, B. E. S. Akgul, and L. N. Chakrapani, "A probabilistic CMOS switch and its realization by exploiting noise," *IFIP Intl Conf VLSI*, vol. 5, pp. 535–541, 2005.
- [24] M. S. K. Lau, K - V Ling, Y - C Chu, and A. Bhanu, "A general mathematical model of probabilistic ripple-carry adders," in *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*, Mar. 2010, pp. 1100–1105. doi: 10.1109/DATE.2010.5456973.
- [25] Q. Xu, T. Mytkowicz, and N. S. Kim, "Approximate computing: a survey," *IEEE Design & Test*, vol. 33, no. 1, pp. 8–22, Feb. 2016, doi: 10.1109/MDAT.2015.2505723.
- [26] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "MACACO: modeling and analysis of circuits for approximate computing," in *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2011, pp. 667–673. doi: 10.1109/ICCAD.2011.6105401.
- [27] W. Liu, L. Chen, C. Wang, M. O'Neill, and F. Lombardi, "Design and analysis of inexact floating-point adders," *IEEE Transactions on Computers*, vol. 65, no. 1, pp. 308–314, Jan. 2016, doi: 10.1109/TC.2015.2417549.
- [28] V. Camus, J. Schlachter, C. Enz, M. Gautschi, and F. K. Gurkaynak, "Approximate 32-bit floating-point unit design with 53% power-area product reduction," in *ESSCIRC Conference 2016: 42nd European Solid-State Circuits Conference*, Sep. 2016, vol. 2016-Octob, pp. 465–468. doi: 10.1109/ESSCIRC.2016.7598342.
- [29] M. Yan, Y. Song, Y. Feng, G. Pasandi, M. Pedram, and S. Nazarian, "kNN-CAM: a k-nearest neighbors-based configurable approximate floating point multiplier," in *20th International Symposium on Quality Electronic Design (ISQED)*, Mar. 2019, vol. 2019-March, pp. 1–7. doi: 10.1109/ISQED.2019.8697584.
- [30] J. Hu and W. Qian, "A new approximate adder with low relative error and correct sign calculation," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2015, vol. 2015-April, pp. 1449–1454. doi: 10.7873/DATE.2015.0627.





- [31] "IEEE standard for floating-point arithmetic," *IEEE Std 754-2008*, pp. 1–70, doi: 10.1109/IEEESTD.2008.4610935.
- [32] P. Behrooz, *Computer arithmetic: Algorithms and hardware designs*, Oxford University Press, 2000.
- [33] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, "A low latency generic accuracy configurable adder," in *Proceedings of the 52nd Annual Design Automation Conference*, Jun. 2015, vol. 2015-July, pp. 1–6. doi: 10.1145/2744769.2744778.

BIOGRAPHIES OF AUTHORS



Manjula Narayanappa     working as assistant professor in electronics and communication domain at Dr. Ambedkar Institute of Technology. She is pursuing Ph.D. VTU Extension Centre, UTL Technologies Ltd., Bangalore. She has 13 years of working experience. Her areas of interest are VLSI and embedded systems. She can be contacted at this email: manjulan_12@rediffmail.com.



Dr. Siva Sankar Yellampalli     obtained his M.S. and Ph.D. from Louisiana State University. He is currently with VTU Extension Centre, UTL Technologies Ltd. He worked on a broad range of research topics including very large scale integration (VLSI), mixed signal circuits/systems development, micro-electromechanical systems (MEMS), and integrated carbon nanotube based sensors. He has published a book in the area of mixed signal design, and edited two books on carbon Nano tubes. He also published multiple journal papers and IEEE conference papers in these areas of research. He can be contacted at this email: siva.yellampalli@utltraining.com.