

An efficient floating point adder for low-power devices

Manjula Narayanappa¹, Siva S. 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 23, 2022

Revised Nov 17, 2023

Accepted Dec 22, 2023

Keywords:

Approximate computing

Floating point adder

Low power

Mantissa adder

Single precision

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. 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. 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.

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 Electronic 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 silicon field effect transistors (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. 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. 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. 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.

In the current complementary metal-oxide semiconductor (CMOS) technology, the benefits of device scaling are impeded by the reliability issues due to the process variations, ageing effects and soft errors. Leakage current, static power are increasingly adding to the concerns towards 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. Design of approximate computation blocks is one such potential solution [1].

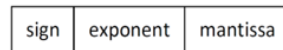
Most of the modern graphics processors for multimedia and other applications have dedicated digital signal processing blocks. These applications output an image, video or an audio signal and the limited perception of human senses allows for an approximation of the computations involved in the demanding digital signal processing (DSP) algorithms for these applications [2]. Even an analog computation that yields good enough results instead of accurate results is also acceptable [3]. Addition is the most fundamental and significant mathematical operations used in all signal/image processing applications [4], [5]. Deterministic approximate logic or probabilistic imprecise arithmetic are normally employed for soft adders [6].

Various low-power design approaches using approximate computing have been introduced, such as algorithmic noise tolerance [7], [8], non-uniform voltage over scaling [9], and significance-driven computation [10], [11]. Verma *et al.* [12] have presented an innovative adder design known as the almost correct adder (ACA), which offers exponentially faster performance compared to traditional adders. They also proposed the variable latency speculative adder (VLSA) with a slight area overhead. Additionally, some adder configurations meet real-time energy requirements by reducing complexity at the algorithmic level [13], [14]. The lower part OR adder [15] relies on approximate logic with a distinct truth table compared to a standard adder. The probabilistic full adder (PFA) [16]-[20] is based on probabilistic CMOS technology, which is a platform for modeling nano-scale designs and reducing power consumption [21]-[23].

2. BACKGROUND

2.1. IEEE-754 floating point format

Floating point representation offers a wider dynamic range in comparison to fixed-point representation for real numbers. However, floating point hardware is known for its complexity and substantial power consumption. The predominant standard for floating point formats is IEEE 754-2008 [24], which encompasses various basic and extended types. These formats include half precision (16-bits), single precision (32-bits), double precision (64-bits), extended precision (80-bits), and quad precision (128-bits). The typical IEEE floating point format, as depicted in Figure 1, features an exponent part with a bias of $2^{(E-1)}-1$, where E denotes the number of exponent bits. Single precision and double precision formats are the most commonly used in contemporary computer systems. You can find details regarding the exponent and mantissa bits for IEEE-754 basic and extended floating point types in Table 1.



$$FP\ No = (-1)^s \times 2^{exponent-bias} \times (1 + mantissa)$$

Figure 1. 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	62.5%
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%

2.2. Floating point adder architecture

A typical floating point adder architecture comprises distinct hardware components for tasks like exponent comparison, mantissa alignment, mantissa addition, normalization, and rounding of the mantissa (as depicted in Figure 2 and elaborated by Behrooz [25]). Initially, two operands are extracted from their floating point formats, and each mantissa has the hidden '1' bit added to it. The addition of floating point numbers entails a series of operations, starting with comparing the exponents and adding the mantissas. The exponents are first assessed to determine the larger of the two. Depending on the result of the exponent comparison, the mantissas are swapped and then aligned to have the same exponent value before undergoing addition in the mantissa adder. After the addition, normalization shifts are essential to bring the result back to the IEEE standard format. Normalization is achieved by left-shifting with a count of leading zeros, making the

detection of leading zeros a critical step in this process. Finally, rounding the normalized result is the last operation before storing the result back. Special cases such as overflow, underflow, and not-a-number are also detected and indicated by flags.

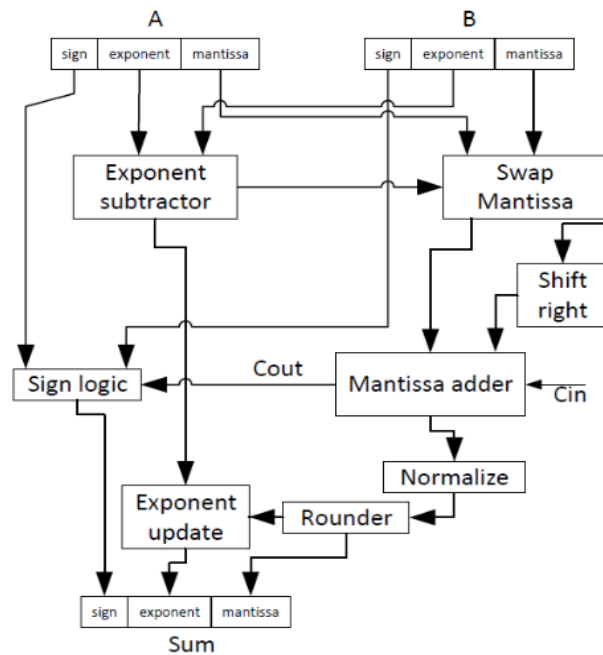


Figure 2. Floating point adder algorithm

3. APPROXIMATE FLOATING POINT ADDER

The approximate floating point 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 as shown in Figure 3. 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.

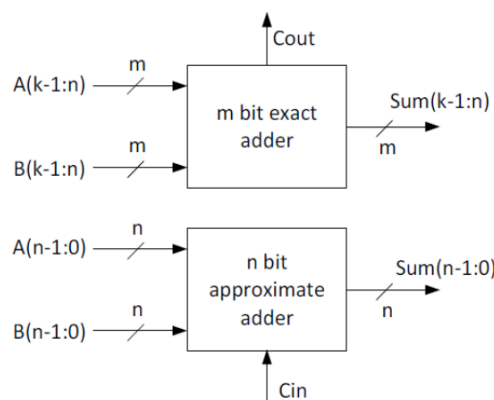


Figure 3. Approximate adder concept

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.

3.1. Basic building block: 8-bit approximate adder

The carry equation for a conventional carry look ahead adder is given by (1):

$$C_{i+1} = G_i + G_{i-1}P_i + \dots + G_0 \prod_{j=1}^i P_j + C_{in} \prod_{j=0}^i P_j \tag{1}$$

where, C_{in} is the input carry and P_i and G_i are propagate and generate signals of the i^{th} stage. If the carry equation is split up into two segments, as in (2):

$$C_{i+1} = \left(\sum_{j=i-W+1}^i G_j (\prod_{k=j+1}^{i-1} P_k) \right) + \left(\sum_{j=0}^{i-W} G_j (\prod_{k=j+1}^{i-1} P_k) + C_{in} \prod_{j=0}^i P_j \right) \tag{2}$$

where, W is the window size and, the first segment consists of W most significant (MS) bits and the second segment consists of $N-W$ least significant (LS) bits. The first part of the (2) is the approximate part, while the second part is called the augmenting part. For approximate carry generation with a window size of W , the output carry at the i^{th} stage is compute using the approximate part only. Computing an approximate C_{i+1} is faster and consumes less hardware resources and hence lesser power as compared to computing precise carry. An 8-bit adder is chosen as the basic building block for the floating point approximate adder in the proposed design. Figure 4 shows the structure of conventional full adder. 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. The output carry of this 8-bit adder block is computed approximately using the approximate part in (2), 4-bit carry generator block as shown in Figure 6 is used for generating the approximate carry for the 8-bit adder using the 4-MS bits.

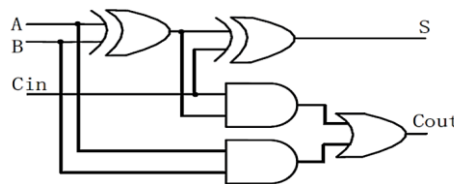


Figure 4. Conventional full adder

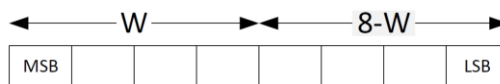


Figure 5. W-bit window for approximate computation

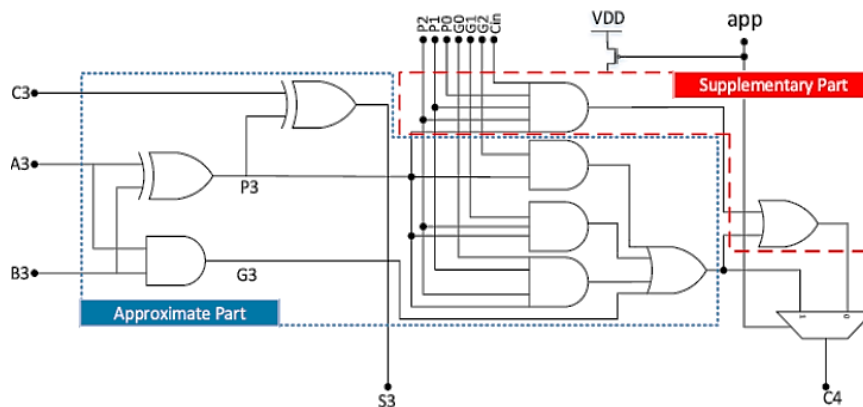


Figure 6. Carry generator for 4-bit block

In the proposed adder, for the LS, W -bit window sum and carry are computed as per (3). The schematic of 1-bit full adder in the proposed configuration is given by Figure 7. The sum and carry for most

significant (8-W=4) bits are computed as for an exact adder are given in (4). The truth table for the proposed sum and carry equations is given in Table 2.

$$\begin{aligned} C_{i+1} &= a_i b_i + c_{in} \\ S_i &= \underline{C}_{i+1} \end{aligned} \tag{3}$$

$$\begin{aligned} C_{i+1} &= a_i b_i + c_{in}(a_i \oplus b_i) \\ S_i &= (a_i \oplus b_i \oplus c_{in}) \end{aligned} \tag{4}$$

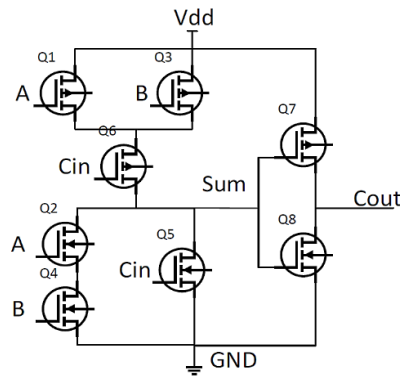


Figure 7. Schematic for carry and sum generator of proposed adder

Table 2. Truth table for proposed adder

A	B	C _{in}	Proposed		Exact	
			S	C _{i+1}	S	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 carry out at each stage to the sum computed for that stage reduces the hardware for 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 carry at each bit. A total of 8 transistors are used for 1-bit sum and carry generation.

3.2. 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.

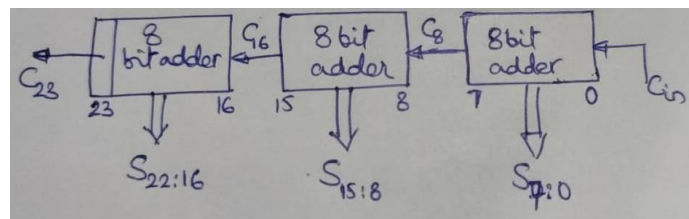


Figure 8. Proposed 23-bit mantissa adder

3.3. 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 exact 8-bit adder. Further, we discuss error metrics for the evaluation purpose.

4. ERROR METRICS

4.1. Error distance

The error distance (ED) between two binary numbers, a (erroneous) and b (correct), is defined as the arithmetic distance between these two numbers. Where, i and j are the indices for the bits in a and b , respectively. Suppose for an 8-bit adder, the correct sum for a given set of operands is “1110 0101” and the incorrect outputs are “11100100” and “11110101”. Then the two erroneous values “11100100” and “11110101” have an ED of 1 and 16 respectively.

$$ED(a, b) = |a - b| = \left| \sum_i a(i) \times 2^i - \sum_j b(j) \times 2^j \right| \quad (5)$$

For a non-deterministic implementation, the output is probabilistic and usually follows a distribution for a given input a_i . In this case, the ED of the output (denoted by d_i) is defined as the weighted average of EDs of all possible outputs to the nominal output. Assume that for a given input, the output has a nominal value b , but it can take any value given in a set of vectors b_j ($1 \leq j \leq r$). The ED of the output is then given by (6). Where p_j is the output probability of b_j ($1 \leq j \leq r$).

$$d_i = \sum_j ED(b_j, b) \times p_j \quad (6)$$

4.2. Mean error distance

Mean error distance (MED) d_m of a circuit for non-deterministic inputs with a certain probability of occurrence is defined as the mean value of all the EDs of all possible outputs for each input. Assuming that the inputs are defined by a_i . ($1 \leq i \leq s$) and probability of occurrence of each vector is q_i ($1 \leq i \leq s$), then MED is given by (7). Where, d_i is the ED of the outputs for input a_i . For a uniformly distributed system, all the inputs have an equal probability of occurrence and hence, q_i is same for all input vector.

$$d_m = \sum_i d_i \times q_i \quad (7)$$

5. SIMULATION AND RESULTS

The proposed adder circuit is simulated in Cadence environment for delay and power consumption and error analysis. The results are presented hereby. The maximum ED for the 8-bit adder is 3. 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, error distances computed between the approximate and accurate outputs. The maximum ED for the 8-bit adder is 3.

5.1. 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, error distances computed between the approximate and accurate outputs. The maximum ED for the 8-bit adder is 3.

5.2. 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 computation of 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 $4T$.

5.3. Power consumption tradeoff

The energy consumed by a probabilistic inverter experiences an exponential increase as the probability of obtaining the correct output rises. When it comes to approximate implementations, power consumption is generally viewed as being directly proportional to the number of gates involved. In the newly proposed adder circuits, the reduction in the number of transistors enables a lower operating voltage,

resulting in an overall reduction in power consumption. In the case of a traditional full adder, if the power consumed for a 1-bit operation is normalized to 1, then the power consumed for a k-bit conventional full adder amounts to k. However, in the context of the proposed 8-bit adder, the reduction in the number of transistors leads to a decrease in the operating voltage, from 1.13 V in an accurate implementation to 1.04 V. Consequently, this reduction in voltage contributes to an estimated decrease in power consumption.

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

Which is 7.5% lower than the conventional adder. 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 and 1.13 V in the exact mode, respectively. The different supply operating voltages are demonstrated in Figures 9 and 10 for different approximations and truncations considering varied bits. Table 3 demonstrates the percentage power savings considering varied approximations and truncation against the base case. Approximation 3 saves the maximum power.

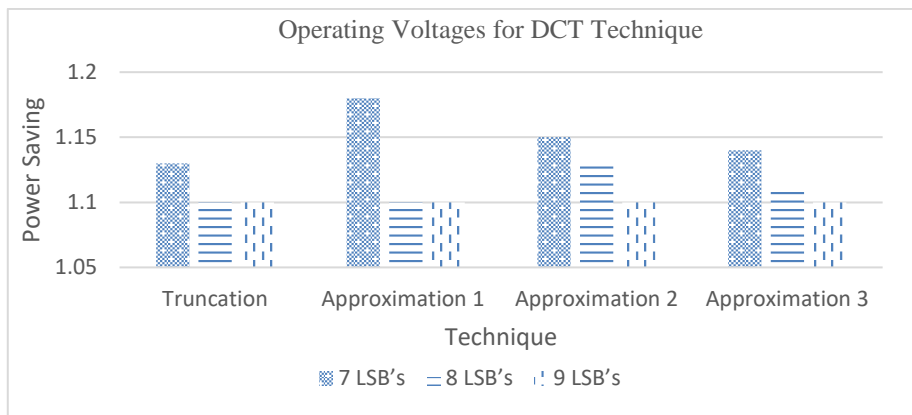


Figure 9. Operating voltages considering different bits for DCT technique

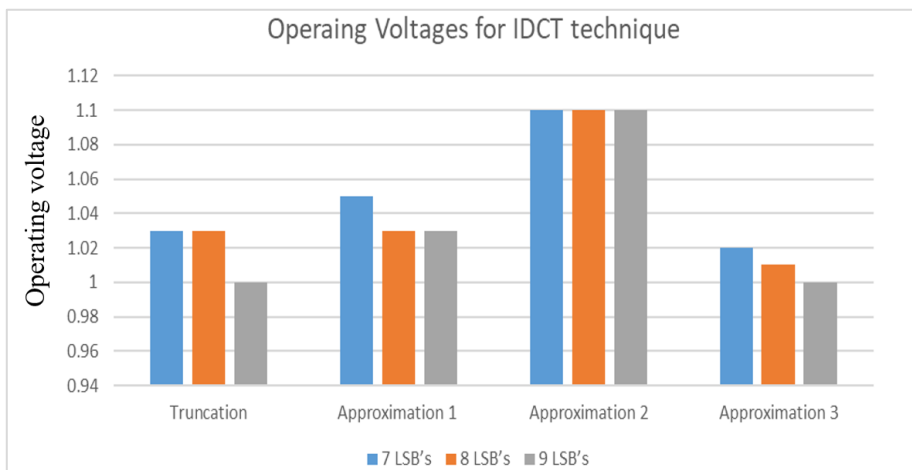


Figure 10. Operating voltages considering different bits for IDCT technique

Table 3. 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





6. 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 has 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 that of the approximate adder for the sum generation gives a distinct advantage in terms of the power consumption as compared to the conventional full adder.





REFERENCES

- [1] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *2013 18th IEEE European Test Symposium (ETS)*, May 2013, pp. 1–6, doi: 10.1109/ETS.2013.6569370.
- [2] R. Hegde and N. R. Shanbhag, "Soft digital signal processing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 6, pp. 813–823, Dec. 2001, doi: 10.1109/92.974895.
- [3] M. A. Breuer, "Let's think analog," in *IEEE Computer Society Annual Symposium on VLSI: New Frontiers in VLSI Design (ISVLSI'05)*, 2005, pp. 2–5, doi: 10.1109/ISVLSI.2005.48.
- [4] V. Beiu, S. Aunet, J. Nyathi, R. R. Rydberg, and W. Ibrahim, "Serial addition: locally connected architectures," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 11, pp. 2564–2579, Nov. 2007, doi: 10.1109/TCSI.2007.907885.
- [5] S. Cotofana, C. Lageweg, and S. Vassiliadis, "Addition related arithmetic operations via controlled transport of charge," *IEEE Transactions on Computers*, vol. 54, no. 3, pp. 243–256, Mar. 2005, doi: 10.1109/TC.2005.40.
- [6] J. Huang and J. Lach, "Exploring the fidelity-efficiency design space using imprecise arithmetic," in *16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011)*, Jan. 2011, pp. 579–584, doi: 10.1109/ASPDAC.2011.5722256.
- [7] S. Byonghyo, 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, May 2004, doi: 10.1109/TVLSI.2004.826201.
- [8] G. V. Varatkar and N. R. Shanbhag, "Energy-efficient motion estimation using error-tolerance," in *Proceedings of the 2006 international symposium on Low power electronics and design-ISLPED '06*, Oct. 2006, pp. 113–118, doi: 10.1145/1165573.1165599.
- [9] 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," 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.
- [10] D. Mohapatra, G. Karakonstantis, and K. Roy, "Significance driven computation," 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.
- [11] N. Banerjee, G. Karakonstantis, and K. Roy, "Process variation tolerant low power DCT architecture," in *2007 Design, Automation and Test in Europe Conference and Exhibition*, Apr. 2007, pp. 1–6, doi: 10.1109/DATE.2007.364664.
- [12] A. K. Verma, P. Brisk, and P. Jenne, "Variable latency speculative addition: a new paradigm for arithmetic circuit design," in *2008 Design, Automation and Test in Europe*, Mar. 2008, pp. 1250–1255, doi: 10.1109/DATE.2008.4484850.
- [13] 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.
- [14] 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 and Test in Europe Conference and Exhibition (DATE 2010)*, Mar. 2010, pp. 1725–1730, doi: 10.1109/DATE.2010.5457093.
- [15] 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.
- [16] S. H. Sreedhara, V. Kumar, and S. Salma, "Efficient big data clustering using adhoc fuzzy C means and auto-encoder CNN," in *Inventive Computation and Information Technologies*, 2023, pp. 353–368, doi: 10.1007/978-981-19-7402-1_25.
- [17] M. S. K. Lau, L. Keck-Voon, C. Yun-Chung, and A. Bhanu, "A general mathematical model of probabilistic ripple-carry adders," in *2010 Design, Automation and Test in Europe Conference and Exhibition (DATE 2010)*, Mar. 2010, pp. 1100–1105, doi: 10.1109/DATE.2010.5456973.
- [18] S. Borkar, T. Karnik, and V. De, "Design and reliability challenges in nanometer technologies," in *Proceedings of the 41st annual Design Automation Conference*, Jun. 2004, pp. 75–75, doi: 10.1145/996566.996588.
- [19] W. Huan-Sheng and R. M. Mersereau, "Fast algorithms for the estimation of motion vectors," *IEEE Transactions on Image Processing*, vol. 8, no. 3, pp. 435–438, Mar. 1999, doi: 10.1109/83.748899.
- [20] M. Shafique, L. Bauer, and J. Henkel, "3-tier dynamically adaptive power-aware motion estimator for h.264/AVC video encoding," in *Proceeding of the thirteenth international symposium on Low power electronics and design-ISLPED '08*, 2008, pp. 147–152, doi: 10.1145/1393921.1393962.
- [21] J. George, B. Marr, B. E. S. Akgul, and K. V. Palem, "Probabilistic arithmetic and energy efficient embedded signal processing," in *Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems*, Oct. 2006, pp. 158–168, doi: 10.1145/1176760.1176781.
- [22] G. V. Varatkar and N. R. Shanbhag, "Energy-efficient motion estimation using error-tolerance," in *Proceedings of the 2006 international symposium on Low power electronics and design-ISLPED '06*, 2006, pp. 113–118, doi: 10.1145/1165573.1165599.
- [23] D. Mohapatra, G. Karakonstantis, and K. Roy, "Low-power process-variation tolerant arithmetic units using input-based elastic clocking," in *Proceedings of the 2007 international symposium on Low power electronics and design*, Aug. 2007, pp. 74–79, doi: 10.1145/1283780.1283797.
- [24] "IEEE standard for floating-point arithmetic," IEEE Std 754-2008, Aug. 2008, doi: 10.1109/IEEESTD.2008.4610935.
- [25] P. Behrooz, *Computer arithmetic: algorithms and hardware designs*, Oxford University Press, 2000.

BIOGRAPHIES OF AUTHOR

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.



Siva S. 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 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 nanotubes. 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.