International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-14 Issue-5, June 2025

# Machine Learning in Approximate Computing Applications: A Comprehensive Review of Recent Research and Accomplishments in Energy-Efficient Computing

**Sreelakshmi Vadlamudi, Syamala Yarlagadda, Madhu R**

***Abstract:*** ***Recently, approximate computing has become a well-known computer outlook. It is a broad field with new research paths emerging daily. Approximate computing systems enhance energy efficiency and computational speed at the expense of precision in output. From a computational standpoint, this paper offers a concise and thorough overview of recent research areas and accomplishments in energy-efficient computing. We classify and analyse the machine learning techniques used in approximate computing applications. Approximate computing is used at the software, circuit, and hardware levels. Machine learning (ML) methods are crucial in various approximate computing applications, enabling performance improvements at multiple levels. The scope of the systematic literature review encompasses an in-depth examination of the most prominent machine learning (ML) trending techniques in approximate computing applications. This paper also addresses recent breakthroughs in approximate computing hardware, software, and approximate data communication.***

***Keywords: Approximate Computing, Machine Learning, Energy Efficiency, Quality Control, Neural Networks***

***Abbreviations:***

VHDL: Very High-Speed Integrated Circuit Hardware Description Language
CPLDS: Complex Programmable Logic Devices
ICs: Integrated Circuits
CLBs: Configurable Logic Blocks
AC: Approximate Computing
ML: Machine Learning
EDA: Electronic Design Automation
MACACO: Modeling an Analysis of Circuits for Approximate Computing
NN: Neural Network
VOS: Voltage Over-Scaling
TOQ: Target Output Quality
FA: Full Adder
ABACUS: Automating the Creation of Approximate Computing Circuits
PDP: Power-Delay Product
MCUs: Microcontrollers
AxNN: Approximate Neural Network
KNN: K Nearest Neighbor
ACU: Analog Calculation Unit
SLR: Systematic Literature Review
SVM: Support Vector Machine
ACTs: Approximate Computing Techniques

## I. INTRODUCTION

The current level of computer microchip miniaturisation is VLSI (Very Large-Scale Integration), which typically applies to microchips containing thousands of transistors. System developers are reportedly centralising on the ways to establish their novel digital design concepts in configurable languages, namely very high-speed integrated circuit hardware description language (VHDL) and Verilog, and intend them for Complex programmable logic devices (CPLDS) and Field Programmable gate arrays due to extortionate progressions in VLSI and Field Programmable Gate Arrays (FPGAs) [1]. Since the 1970s, VLSI has been used to develop Integrated circuits (ICs) in computing. VLSI has experienced rapid growth in its industry usage since its inception. However, it needed suitable customisation with programmable logic components. The ideal design style of VLSI is a Field-Programmable Gate Array (FPGA) with Configurable Logic Blocks (CLBs), I/O pads, and routing channels. VLSI was also known for its energy-efficient design. Furthermore, FPGA design can be leveraged to incorporate approximate computing phenomena in embedded computing devices, resulting in improved computing efficiency and energy conservation, as well as reduced overhead. Unlike exact computing, approximate computing remarkably reduces computational complexity and exhibits acceptable error tolerance.

Approximate Computing (AC) is a technique of computation that may return a probabilistic (approximate) value rather than a very accurate result. Such computing is best utilised in applications that expect a value of approximately. Interestingly, AC is based on the observation that exact computation demands the consumption of more resources. Thus, a selective approximation can help achieve performance while satisfying the requirements of specific applications. AC is devised as an alternative to the computing phenomenon called exact computing [2]. To meet demands in terms of performance and constrained budgets, AC often trades off the effort needed with the quality of computing. AC can be used effectively in various processing units,

**Manuscript received on 21 June 2023 | First Revised Manuscript received on 20 December 2024 | Second Revised Manuscript received on 16 May 2025 | Manuscript Accepted on 15 June 2025 | Manuscript published on 30 June 2025.**

*Correspondence Author(s)

**Sreelakshmi Vadlamudi***, Research Scholar, Department of Electronics & Communication Engineering, JNTUK, Kakinada (Andhra Pradesh), India. Email ID: sreelakshmi.vadlamudi@gmail.com, ORCID ID: 0000-0002-5633-5112

**Dr. Y. Syamala**, Department of Electronics & Communication Engineering, Seshadri Rao Gudlavalleru Engineering College, Gudlavalleru (Andhra Pradesh), India. Email ID: coolsyamu@gmail.com

**Madhu Ramarakula**, Department of Electronics & Communication Engineering, JNTUK, Kakinada (Andhra Pradesh), India. Email ID: madhu_ramarkula@jntucek.ac.in

memory technologies, and components.

Approximate computing is a broad phrase that refers to strategies that take advantage of applications' inbuilt robustness to obtain productivity gains at every level of the computer system. Approximation techniques require a standard governance structure to govern the approximation standard and observe the output status. Compared to AC design, this vital domain has received little recognition from the scientific population. Setting approximate circuit design knobs can be accomplished in two ways: 1) forward problem design, which emphasises the influence of configuring design knobs on the outcome of an adjustable approximate layout, and 2) inverse problem design, which focuses on discovering the ideal design knobs setting for a specific output quality bound. The inverse technique is less researched than the forward problem, primarily due to the vast number of input-dependent design parameters that it involves. We address the inverse problem of determining the optimal approximate design knob settings that satisfy a given target operating quality (TOQ), input data characteristics, and user preferences. Existing strategies for controlling the approximate circuit (AC) status include Green [3], which employs a sampling-based analysis method. Similarly, SAGE utilises sampling to compare approximate outputs to precise outputs every NNN invocations [4]. In both approaches, calibration is performed if the difference exceeds a predefined threshold, such as the TOQ. On the other hand, the quality of uncontrolled invocations cannot be guaranteed, and prior quality violations cannot be reimbursed. Lightweight quality checkers with high efficiency are often employed to decide whether to utilise rollback approximation. However, the quality checker proposed is application-specific, limiting its general applicability [5]. Additionally, highlights that for large applications, where less than 20% of the inputs occupy the data capacity, significant approximation errors become unavoidable, resulting in low predictive accuracy [6]. To address these challenges, Wang et al [7] proposed integrating multiple lightweight predictors to achieve high-accuracy predictions. Xu et al [8]. introduced an optimal control framework that combines classifier and accelerator training with optimal training data selection, enabling a sequential learning algorithm. Nevertheless, this work focuses primarily on software proximity control, neglects input dependence, and does not consider selecting an appropriate layout from a set of design options.

An online administration of the exacting quality system for approximate computing, in which computation accuracy is compromised to gain higher production and/or lower energy usage. Approximate computing deals with the methods for achieving this trade-off. The methodical approximation has been applied in various disciplines, including machine learning, image processing, and video processing. Programmers have approximated several methods in these disciplines to improve performance. Because the consumer will not notice minor variations in output, video processing methodologies are suitable candidates for approximation. A consumer, for example, can tolerate incidental missed frames or a slight reduction in resolution while playing video if it enables smooth video playback. In particular, machine learning and data analysis applications offer opportunities to use approximation to increase efficiency, especially when dealing with large datasets. Various software approximation techniques have been developed. Machine Learning (ML) has benefited the VLSI industry thus far by utilising EDA (Electronic Design Automation) tools to their maximum potential, which helps reduce design time and production costs.

## II. RELATED WORK

R. Venkatesan et.al presented Modeling an Analysis of Circuits for Approximate Computing (MACACO), a systematic technique for modeling and analysing circuits for approximate computing which could be employed to analyse ways through which approximation circuit responds in comparison to a typical resolution professional by computing measures like worst- case, average-case error, error probability, and dispersion. Approximate computing approaches progressed from being impromptu and application-specific to being universally pertinent, due to the advancement of systematic design methodologies [9].

A. Rahaet. Al provided a unique design methodology for building high-quality, flexible Reduce-and-Rank kernel-based applications that utilise ideas from intermediate outcomes within the applications to inform the approximation approach. Approximate computing is a new layout prototype that applies applications' inborn capacity to generate satisfactory results even when their operations are handled approximately [10]. S. Venkataramani et.al revolutionised the concept of designing energy-efficient large-scale neural network (NN) hardware processors using approximation computation. The considerations that (i) NNs are applied in situations where fallible outcomes are permissible, if not unavoidable, and (ii) they are pretty resistant to erroneous in many of their constituent computations prompted this research [11].

V.K. Chippa et.al offered a structured methodology for hardware approximate computing. First, we proposed an application resilience analysis tool for determining approximation-prone operations in an application and quantifying the effect of estimating them on the application output quality [12]. V. Gupta et.al suggested logic complexity reduction as an intermediate methodology for obtaining the relaxation of arithmetic precision. Several inaccurate or estimated FA cells are established to be used to generate multi-bit arithmetic units along with trade-off power, domain, and quality for error-flexible DSP systems [13].

D. Mohapatra et.al proposed research methodologies for developing meta-functions with more gracious scaling performance for approximate computation under VOS. Approximate computing techniques, which apply processes such as voltage over-scaling (VOS) to leverage the inherent resilience of algorithms, have generated considerable excitement [14]. J. Han et al. discuss the rapid advances in the domain, particularly in the design of approximate arithmetic units, application errors, and quality metrics, as well as computation and approximate computing methodologies. Approximate computing has been established as a viable approach to developing energy-efficient digital systems [15].

Mahmoud Masadehet. Al proposed a compact and effective machine-learning-based strategy for designing an input-aware layout checkbox, i.e., status controller, that adapts an approximate layout to attain the target output quality (TOQ) [16]. Masadeh. M et.al introduced self-compensating approximation accelerators based on machine learning for energy-efficient systems. The suggested blunder recompense mechanism, which is built into the architecture of approximation hardware accelerators, effectively minimises the total error at the output. It uses minimally controlled machine learning technologies like decision trees to capture the error's input dependence [17].

Hasan. O et.al ML models were used to propose a unique pulverised input-formulated status control technique for AC [18]. The suggested method considers the input data for producing training data, creating ML-formulated layouts, and picking an approximate design that meets the TOQ with the least amount of money. This method is utilised in both hardware and software development. For hardware and software approximation designs, it is considered a detriment to construct a completely mechanised version of AC quality control utilising ML models formulated on the input data.

### A. Related Work Based on Approximate Computing at Different Levels

A range of approximate full adder (FA) implementations at the gate-netlist level, as well as several approximate multi-bit adders potentially used as DSP building blocks, are discussed in [13]. Schlachter et al [19], used probabilistic reduction to create approximate adders. This type of adder, on the other hand, cannot be modified in real-time. As a result, we proposed reconfigurable near-full-adder layouts that can adapt at runtime. Kahng and Kang developed their design by integrating control signals with multiplexers to manage the carry bit for each half-adder [20]. Similarly, the work introduced a runtime-reconfigurable adder layout focusing on improving adaptability and efficiency [21]. Liu et al [22], utilised adjustable partial error recovery for approximate multipliers to achieve a more efficient design. To address potential issues, designs often incorporate fragmentary result amplification, approximate full adders in the final stage, and the application of an OR gate. Farshchi et al [23], proposed a multiplier that combines truncation and approximation to handle various bit widths. Similarly, Hashemi et al [24], introduced an approach that pairs truncation with approximation for efficient multiplier design. Kulkarni et al [25], developed a 2 × 2 multiplier primitive and leveraged it as a fundamental unit to construct larger multipliers. In the meantime, the author adds a blunder identification module to increase the reliability of such a multiplier. Nepal et al [26], instituted mechanised behavioral approximation circuit synthesis to automate the creation of approximate computing circuits (ABACUS). Data type simplification, operation, loop transformation, and V2C exchanges were all conducted on behavioural HDL code, resulting in approximation circuits. Li et al [27], developed an approximate-aware HLS flow, with bit width evaluation in fixed-point computation. Lotfi et al [28], have recently applied approximated computing to the FPGA accelerator technology to achieve faster computational productivity by decreasing the fidelity of variables and using GPU to speed up profiling. This is done by lowering the fidelity of variables in OpenCL programs on a case-by-case basis. Approximate computing was used by Zhou et al to provide an energy-efficient SVM. A new approximate adder with a more straightforward carry compensation technique is developed. Additionally, a fixed-width approximate multiplier with an inexpensive compensatory unit is proposed. According to simulation data, approximation errors have a small impact on categorisation reliability. The synthesis outcomes show that the SVM classifier with approximate computing curtails power-delay product (PDP), area, and critical path delay by 32.4 percent, 18.7 percent, and 16.0 percent, sequentially, when weighed up to the same classifier with precise computation units, using the TSMC 90nm CMOS technology [29].

## III. MACHINE LEARNING ON THE TINIEST END DEVICES

While massive data centres were required to enforce ML tasks just a few years ago, prevailing ML layouts and the system requirements to operate them have shrunk to fit into tiny microcontrollers (MCUs), which govern and operate units in constrained IoT devices. This opens up new possibilities for ML applications that use restricted IoT devices to regulate and execute the program. This offers new opportunities for machine learning with limited IoT devices, such as always-on inference to recognise words and human pursuit identification.

Constrained devices have limited memory and computing capabilities and are therefore typically battery-powered or rely on energy harvesting. Limited hardware systems that run ML now have a 32-bit MCU, 500 KB of static RAM, scant megabytes of Flash memory, and cost negligible milliwatts of power.

Minimal powers of regulated mechanisms present several obstacles to implementing ML tasks. When it comes to ML training operations, for instance, specific ML systems built to operate effectively on powerful machines are frequently paired with dedicated hardware such as GPUs or TPUs, as well as significant math libraries. Due to the limited hardware and software capacities of these devices, models produced using these technologies cannot be effortlessly migrated to robust computing systems. As a result, the models must be physically optimised in a time-consuming, blundering, and frequently performance-degrading procedure. Complex operations that restricted devices cannot execute are removed, floating point potencies are quantised to 8-bit integers, and less significant characteristics are pruned.

Running ML (Machine Learning) programs on restricted devices has numerous benefits, including a wide range of conceivable use cases and minimal hardware expenditures, even when installing vast numbers of devices [30].

## IV. APPROXIMATE COMPUTING IN SOFTWARE

To facilitate improved parallel ascend ability on multi-cores, GPUs, and clusters, approximate computing reduces program run-time by skipping computations, easing synchronisation and minimising communication between threads. The selection of computations for approximation which has a reduced influence on the status of outcomes is a significant ultimatum in approximate computing. A way to address this problem is to provide developers with constructs that enable them to express resilient or less significant computations easily. On the other hand, a profiling-based approach may autonomously recognise robust computations and quantify how approximations affect the application outcome. To improve the efficacy of approximation approaches, specified domain knowledge is used [31].

### A. Approximating Neural Networks

Some researchers propose methods for approximating neural networks (NNs) based on the insight that they are often used in error-tolerant applications and are resilient to constituent computations. One such technique involves converting a given neural network (NN) into an approximate neural network (AxNN) for energy-efficient execution. The impact of approximating any neuron on overall performance is assessed using a backpropagation methodology commonly used for training NNs.

To develop an AxNN, neurons with minimal impact on network quality are replaced with their approximate counterparts. Different approximation equivalents are generated by adjusting input precision and neuron weights, allowing for a trade-off between energy consumption and precision. Since training is inherently an error-correction process, the AxNN is gradually refined after its creation, with approximations used to recover performance loss caused by approximation circuits (AC). This retraining process enables further approximation while preserving the desired output quality. Additionally, a neuromorphic computation engine is proposed to execute AxNNs with any weights, topologies, or degrees of approximation.

This configurable hardware framework executes approximate neural networks (AxNNs) and supports reliable energy trade-offs at runtime by integrating neural computing and activation function units. A technique for approximating neural networks involves classifying neurons based on their criticality. A neuron is considered critical (or robust) if a slight change in its computation results in a significant (or negligible) reduction in the final output quality. After determining weights during the training phase, a theoretical methodology is used to calculate neurons' criticality (or exigency) factor in both the output and hidden layers. Neurons with the lowest criticality are identified as candidates for approximation. However, criticality rankings change dynamically after each neuron is approximated due to strong interactions among neurons. An iterative approach is employed to address this, progressively approximating neurons based on a predefined quality budget.

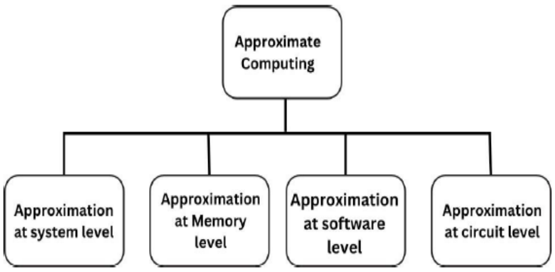
The quality budget decreases with each iteration as the process converges, reducing the number of neurons selected for approximation in each step. The method not only determines the number of neurons to approximate but also chooses the approximation techniques to apply—namely, reliability escalation, memory access skipping, and approximate multiplier circuits. These techniques are chosen based on their ability to save energy with minimal quality loss, leveraging that neural networks allow recovery mechanisms to mitigate the impact of neurons responsible for the most errors.

A design for an imprecise hardware neural network (HW NN) accelerator was proposed that utilises a two-layer feed-forward NN with circuitry to enhance synaptic weight and neuron output while boosting neuron inputs. The approach explores strategies for achieving effective approximate designs, focusing on synaptic weight multipliers rather than adders, as the latter offers only marginal benefits due to their low hardware cost.

Additionally, since a neural network's output layer typically contains a small number of neurons with no subsequent synaptic weights, retraining these neurons to minimise errors is challenging. Therefore, approximation is primarily introduced in the synaptic weight multipliers of the hidden layers. This approach demonstrates that using an imprecise NN accelerator in applications relying on HW NN accelerators can save significant space, time, and energy.

## V. APPROXIMATE COMPUTING STRATEGIES

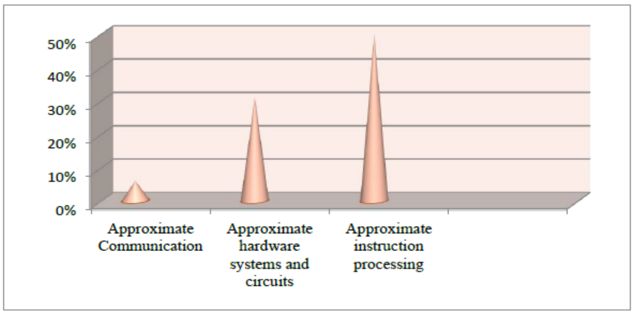
There are four distinct approaches to approximate computing, namely Approximate systems, software, storage, and arithmetic circuits. At the system level, approximate computing is implemented via customised structures with approximation accelerators and programmable processors. By bypassing the implementation of operations in a projected part of the code, software-level approximation minimises energy usage, eases the limitations on synchronisation timing and handshaking, bolsters field-specific knowledge to ease application methodologies, or modifies the execution of methods for a particular function. Supplying varying refresh rates to various memory blocks, easing the guard band of multi-level memory cells, and selective voltage escalating are all methodologies for approximate storage. To increase productivity and energy economy, circuit-level approximation approaches curtail calculation resolution or minimise the crucial delay channel [32]. Fig.1 illustrates the approximation computing methodologies.


Approximate Computing
Approximation at system level
Approximation at Memory level
Approximation at software level
Approximation at circuit level

**[Fig.1: Approximate Computing Strategies at Different Levels] [32]**

## VI. MAJOR AREAS OF APPROXIMATE COMPUTATION

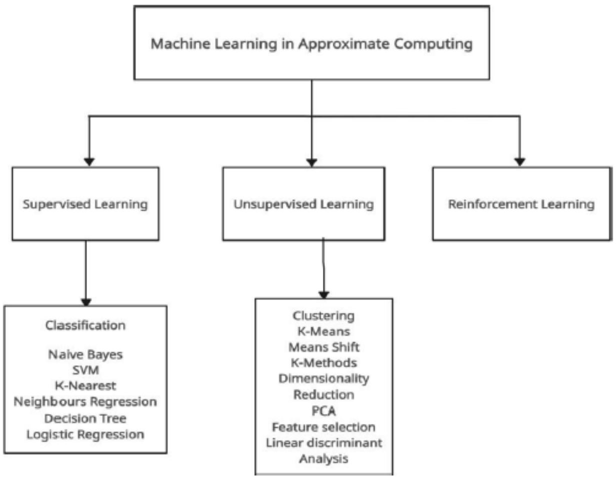
Image processing, computer vision, signal processing, data mining, scientific computing, and financial analysis have all effectively utilised and implemented approximate computing. Researchers find approximate computation in significant areas such as 6% in Approximate communication, 31% in hardware systems and circuits, and 50% in instruction processing. The main areas of Approximate computation are illustrated in Figure 2.



**[Fig.2: Major Areas of Approximate Computation]**

## VII. MACHINE LEARNING BASED ON APPROXIMATE COMPUTING

Figure 3 shows a machine learning system built on approximate computing. H. Younes and A. Ibrahim et al [33] provides a methodology for implementing algorithmic-level approximate computing techniques (ACTs) on dual-monitored machine learning procedures, K Nearest Neighbor (KNN) and Support Vector Machine (SVM).



**[Fig.3: Machine Learning Techniques for Approximate Computing]**

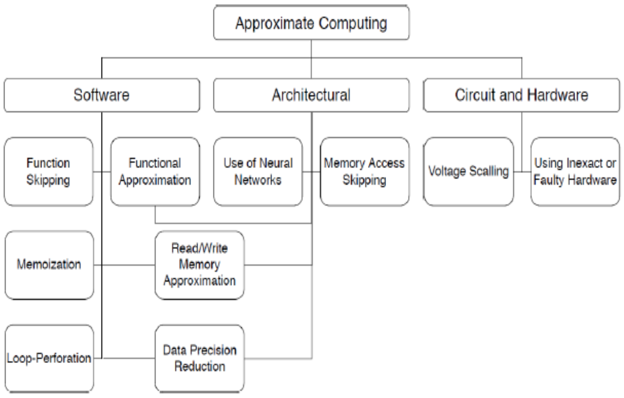
Efficiency, real-time operation, and low power utilisation are all essential considerations for embedded machine learning implementation. The created methodology was tested in two distinct applications: touch modality and picture categorisation. The proposed technology boosts productivity by up to 3.2 times with a precision loss of less than 4.7 per cent, according to the results. For touchmodality detection, H. Younes and A. Ibrahim et al [34], introduced an approximate machine learning classifier (KNN classifier). Approximation is used in ACTs on both the hardware and software levels. The approximation is used in adders, multipliers, memory, CPUs, GPUs, and FPGAs,among other hardware approaches. Software-level technologies are computing- and data-oriented strategies. By using software-level approximation methodologies, execution timing and memory usage are reduced by up to 38% and 55%, respectively. R. Zhang and N. Uetake et al. [35] created a programmable analogue calculation unit (ACU) that can approximatethe computation of arbitrary functions with dual operands. By adopting an effective process of Support Vector Regression, the intended capabilities are achieved by huge-scale integrated (VLSI) circuits in one clock cycle, utilising just 600 transistors (SVR). According to circuit simulation findings, the designed ACU successfully quantifies all of the intended functions with a mean deviation of less than 1.7 per cent. KNN – CAM, a K- Nearest Neighbors (KNN)-based adjustable Approximate floating-point Multiplier, was designed by A. M. Yan and Y. Song et al [36]. KNN-CAM saves significant space and energy by utilising approximate computingoptions. This technique achieves a 47.2% reduction in hardware cost with only a 0.3%reduction in classification accuracy. The approximate computing design paradigm was developedby Mario Barbareschi and Salvatore Barone et al [37] to address the hardware overhead in the classification reliability area. Precision-scaling techniques are used to train decision tree models. For both the least error and minimum area configurations, experimental outcomes reveal a considerable reduction in domain needs. Precision-scaling techniques are used to train decision tree models. Approximate Logic Synthesis: A Reinforcement Learning-Based Technology Mapping Approachwas presented by G. Pasandi, S. Nazarian et al [38] producing and mappinga given Boolean network to a library of logic cells so that inaccuracy amplitude among the approximate and initial (exact) Boolean netlist outcomes is controlled by a predefined total error criterion known as approximate logic synthesis. Q-ALS leverages the robust capabilities of the Q-learning algorithm to identify the maximum error rate. For academic measures, Q-ALS may deliver 70% area and 36% delay deduction, while industrial measures can provide52% area and 30% delay deduction. S. Hashemi and H. Tann et al [39] proposed a complete biometric security system with a rough SW/HW pipeline. Design space Exploration for the SW/HW pipeline flow and the identification of appropriate design knobs that minimise runtime, subject to reliability restrictions, is determined using a futuristic Recurrent Neural Network (RNN) technology formulated on augmentation learning. The efficacyof this procedure is 100 percent. L. Busoniu and D.Ernst et al [40] outlined methodologies for approximating RL with its potent programming origins and classified them into approximate value, policy iteration,and policy search. Agents can learn how to interact with intricate contexts more productively through reinforcement learning.K.V. Ravi Kanth and Divya Kant Agrawal et al [41], proposed effective methodologies for computing SVD-grounded dimensionality deduction in dynamic databases. Various databases are used to handle multimedia assets, including maps, photos, audio, and video. The shortcoming is that it is computationally costly and does not work with dynamic databases. The inaccuracy in SVD computation owing to the approximate computation is less than 10%.

**Table I: Comparison Between Approximate Computing Quality Control Approaches**

| Characteristics | Software-Based Design | Hardware-Based Design | ML-Based Adaptive Design |
|---|---|---|---|
| Hardware/ software | Primarily suited for software approximation, such as loop unrolling, at a granular level. | Applicable to hardware-based approximation but limited to specific configurations | Can be applied to both hardware and software, supporting multiple approximate components |
| Input data dependency | Does not account for variations in input data | Does not differentiate between workloads during training | Strongly dependent on input data for design adaptation and optimisation |
| Quality assurance | Ensures quality by rolling back execution or adjusting the program's output | Quality may not be guaranteed if the real workload differs from the one used for training. | Quality is ensured by selecting the most suitable components through machine learning models. |
| Overhead | Involves overhead for rollback recovery time and associated costs | Low runtime overhead; introduces area and delay overhead | Minimal overhead for generating training data and model building; energy and delay overheads are negligible |

## VIII. APPROXIMATE COMPUTING CLASSIFICATION

Approximation technologies are employed for all the computation task levels. Figure 2 illustratesthe approximate computing classification. Figure 4 classifies approximation techniques into threegroups. They are software, architecture, and hardware. Approximate computing classification based on machine learning techniques has been performed well over the previous years, as described by theResearchers. The software's approximate computing involves the Loop perforation methodology. This technique is an outstanding software approximation. For instance, task-skipping comprises skipping code blocks throughout the duration with the formerly stipulated ambience. Decreasing the bit-width employed for data rendition is another approximation technique for software applications. Data fidelity depletion influences the software's implementation time performance. Hardware-based approximation technologies use the implementations of arithmetic operators. Approximate compressors, a type of hardware approximation, can also be found in theimage processing sector.



**[Fig.4: Approximate Computing Classification]**

Goiri. Bianchini, R. et.al presented "ApproxHadoop: Bringing Approximations to Map Reduce Frameworks. The software's approximate computing involves Function Skipping. In Function skipping, some of the tasks are bypassed while preserving a rate of precision and defect tenacity are expounded by the user, in a system comprising functions that enhance each other in delivering an outcome [42].

Memorisation: Memorisation preserves the outcomes of functions for specific inputs to be utilised later. Some input data is regularly employed, and the computed outcomes are saved and repurposed without re-running the procedure.

Shafique.M and Ahmad. W et. al presented "A low latency Generic accuracy configurable Adder "The software approximate computing involves Loop-Perforation. It is used to reduce the execution time in loop-based algorithms. Numerical algorithms are considered a good example of this type of application. It is possible to apply it in both software and programmable hardware code. The distinction is that a loop can be realised as a series of parallel circuits on programmable hardware. Consequently, the effects of loop perforations on software and hardware enactment could be varied. It influences the application's processing timing in software, and it impacts the energy and area consumption in FPGA implementations [43].

Venkataramani. S and Chakradhar S.T et.al presented "Approximate computing and the quest for computing efficiency". The software's approximate computing involves Functional Approximation. This approximation can emerge as an alternate theoretical implementation of logical operators on an architectural level. This methodology is demonstrated using various approximation patterns on processors. Approximate computing, when used in software, consists of inexact computations that produce outputs that are less precise than typical [44].

Most approximate computing technologies for software involve modifying the algorithm to execute approximately, resulting in a faster outcome. Ranjan proposed "Approximate storage for energy-efficient spintronic memory." Venkataramani.S et al [45], approximation of Read/Write Memory: This involves estimating the data fetched from or written to memory, as well as the read and write operations that occur. This is primarily deployed in video and image applications, where precision and resolution are frequently reduced to reduce memory operations. Soft PCM: Enhancing energy efficiency and longevity of phase change memory in video applications via approximate write," presented by Fang. Y and Li et al [46].

"Texture cache approximation on GPUs" was presented by Sutherland, M, and San Miguel. In [47], the authors offer a methodology for generating approximation values that utilise GPU texture fetching units. This approximation results in an ultimate image output inaccuracy of less than 0.5 percent whilst lowering kernel completion time by 12 percent. "Approximation-aware multi-level cells STT-RAM cache architecture" was proposed by Sampaio. F and Shafique. M et al [48], the authors introduced an approximation methodology for multi-level cells.

STT-RAM memory techniques that reduced their dependability to a user-defined precision loss acceptability. This memory technique provides a high level of fidelity, which shall be deduced. They especially approximate the application's storage data and reduce error-protection hardware, reducing error utilisation.

Data Precision Reduction: It is a set of strategies that can be applied at both the software and architectural levels. Reduced memory usage lowers energy consumption without sacrificing precision. "Energy-aware hybrid precision selection framework for mobile GPUs," presented by Hsiao, C, Chu.C et al [49]. The authors demonstrated that lowering floating-point reliability on mobile GPUs reduces energy utilisation while degrading image quality. This degeneration is tolerable and even undetectable to the naked eye. Lower memory utilisation is appropriate for safety-critical systems, as it minimises the number of vital and critical bits, leaving them less vulnerable to errors.

Precimonius: Tuning Assistant for Floating-Point Precision, given by Rubio-Gonzalez and Nguyen.C et.al. A frequent approximation procedure is to deduce the bit-width used for data encoding [50].

Use of Neural Networks: Through machine learning, a neural network will explore the ways a typical function executes for various inputs. In complex systems, neural networks are applied to approximate functions through software-hardware co-design. Amant proffered "General-purpose code acceleration with limited-precision analogue computation." R.S and Yazdan Baksh et al [51]. Traditional approximate codes will be replaced with corresponding neural networks, which will have pruned outcome precision and faster processing timing.

Chippa proffered a scalable, hardware design. V.K. and Mohapatra. D et al [52]. It involves Voltage Scaling: The equipped voltage level could be escalated at the circuit level, influencing the computational time frame of subsequent blocks within the clock period. It has an impact on the final result's reliability as well as energy usage. The authors explained the voltage over the scaling of individual computation blocks in, asserting that the reliability of the outcomes will "gracefully" escalate with it. Voltage scaling is proactively incorporated in hardware.

## IX. CONCLUSION

Machine learning techniques play a significant role in numerous applications of approximate computation. This study provides a thorough evaluation of machine learning strategies forapproximate computing applications. Wherever plausible, approximate computing facilities with energy and time savings trade off the precision of outcomes. We examined software and hardware-level approximations, as well as data communication, using various methods. The literature on approximate computation in Machine learning over the past decade is examined. However,the systematic literature review (SLR) validates that machine learning (ML) approaches are promising methods forproviding efficiency and accuracy in various software and hardware implementations.

## DECLARATION STATEMENT

After aggregating input from all authors, I must verify the accuracy of the following information as the article's author.

- **Conflicts of Interest/ Competing Interests:** Based on my understanding, this article has no conflicts of interest.
- **Funding Support:** No organisation or agency has sponsored or funded this article. The independence of this research, as it has been conducted without any external sway, is crucial in affirming its impartiality.
- **Ethical Approval and Consent to Participate:** The data provided in this article is exempt from the requirement for ethical approval or participant consent.
- **Data Access Statement and Material Availability:** The adequate resources of this article are publicly accessible.
- **Author's Contributions:** The authorship of this article is contributed equally to all participating individuals.

## REFERENCES

1. O. K. Chinedu, E. C. Genevera, and O. O. Akinyele, "Hardware description language (HDL): An efficient approach to device independent designs for VLSI market segments," 3rd IEEE International Conference on Adaptive Science and Technology (ICAST 2011), Abuja, Nigeria, 2011, pp. 262-267, DOI: https://doi.org/10.1109/ICASTech.2011.6145181
2. Sinha, Sharad & Zhang, Wei. (2016). Low-Power FPGA Design Using Memorization-Based Approximate Computing. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 24. 1-14. DOI: https://doi.org/10.1109/TVLSI.2016.2520979
3. Woongki Baek and Trishul M. Chilimbi. 2010. Green: a framework for supporting energy-conscious programming using controlled approximation. SIGPLAN Not. 45, 6 (June 2010), 198–209. DOI: https://doi.org/10.1145/1809028.1806620
4. Mehrzad Samadi, Janghaeng Lee, D. Anoushe Jamshidi, Amir Hormati, and Scott Mahlke. 2013. SAGE: self-tuning approximation for graphics engines. In Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-46). Association for Computing Machinery, New York, NY, USA, 13–24. DOI: https://doi.org/10.1145/2540708.2540711
5. B. Grigorian, N. Farahpour and G. Reinman, "BRAINIAC: Bringing reliable accuracy into neurally-implemented approximate computing," *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, Burlingame, CA, USA, 2015, pp. 615-626, DOI: https://doi.org/10.1109/HPCA.2015.7056067
6. D. S. Khudia, B. Zamirai, M. Samadi and S. Mahlke, "Rumba: An online quality management system for approximate computing," *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*, Portland, OR, USA, 2015, pp. 554-566, DOI: https://doi.org/10.1145/2749469.2750371
7. Ting Wang, Qian Zhang, Nam Sung Kim, and Qiang Xu. 2016. On Effective and Efficient Quality Management for Approximate Computing. In Proceedings of the 2016 International Symposium on Low Power Electronics and Design (ISLPED '16). Association for Computing Machinery, New York, NY, USA, 156–161. DOI: https://doi.org/10.1145/2934583.2934608
8. Chengwen Xu *et al.*, "On quality trade-off control for approximate computing using iterative training," *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, Austin, TX, 2017, pp. 1-6, DOI: https://doi.org/10.1145/3061639.3062294
9. A. Agarwal, K. Roy, A. Raghunathan and R. Venkatesan, "MACACO: Modeling and analysis of circuits for approximate computing," in Computer-Aided Design, International Conference on, San Jose, CA, USA, 2011, pp. 667-673, DOI: https://doi.org/10.1109/ICCAD.2011.6105401
10. A. Raha, S. Venkataramani, V. Raghunathan and A. Raghunathan, "Quality configurable reduce-and-rank for energy efficient approximate computing," *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, France, 2015, pp. 665-670, DOI: https://doi.org/10.7873/DATE.2015.0569
11. S. Venkataramani, A. Ranjan, K. Roy and A. Raghunathan, "AxNN: Energy-efficient neuromorphic systems using approximate computing," *2014 IEEE/ACM International Symposium on*

*Low Power Electronics and Design (ISLPED)*, La Jolla, CA, USA, 2014, pp. 27-32, DOI: https://doi.org/10.1145/2627369.2627613

12. Chippa, Vinay & Venkataramani, Swagath & Chakradhar, Srimat & Roy, Kaushik & Raghunathan, Anand. (2013). Approximate computing: An integrated hardware approach. Conference Record - Asilomar Conference on Signals, Systems and Computers. 111-117. DOI: https://doi.org/10.1109/ACSSC.2013.6810241
13. Gupta, Vaibhav & Mohapatra, Debabrata & Park, Sang & Raghunathan, Anand & Roy, Kaushik. (2011). IMPACT: IMPrecise adders for low-power approximate computing. Proceedings of the International Symposium on Low Power Electronics and Design. 409-414. DOI: https://doi.org/10.1109/ISLPED.2011.5993675
14. D. Mohapatra, V. K. Chippa, A. Raghunathan and K. Roy, "Design of voltage-scalable meta-functions for approximate computing," *2011 Design, Automation & Test in Europe*, Grenoble, France, 2011, pp. 1-6, DOI: https://doi.org/10.1109/DATE.2011.5763154
15. J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," *2013 18th IEEE European Test Symposium (ETS)*, Avignon, France, 2013, pp. 1-6, DOI: https://doi.org/10.1109/ETS.2013.6569370
16. M. Masadeh, O. Hasan and S. Tahar, "Machine-Learning-Based Self-Tunable Design of Approximate Computing," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 4, pp. 800-813, April 2021, DOI: https://doi.org/10.1109/TVLSI.2021.3056243
17. Masadeh, Mahmoud & Hasan, Osman & Tahar, Sofiene. (2020). Machine Learning-Based Self-Compensating Approximate Computing. DOI: https://doi.org/10.1109/SysCon47679.2020.9275895
18. M. Masadeh, O. Hasan and S. Tahar, "Using Machine Learning for Quality Configurable Approximate Computing," *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Florence, Italy, 2019, pp. 1575-1578, DOI: https://doi.org/10.23919/DATE.2019.8714957
19. Schlachter, Jeremy & Camus, Vincent & Enz, Christian. (2016). Design of energy-efficient discrete cosine transform using pruned arithmetic circuits. 341-344. DOI: https://doi.org/10.1109/ISCAS.2016.7527240
20. Andrew B. Kahng and Seokhyeong Kang. 2012. Accuracy-configurable adder for approximate arithmetic designs. In Proceedings of the 49th Annual Design Automation Conference (DAC'12). Association for Computing Machinery, New York, NY, USA, 820–825. DOI: https://doi.org/10.1145/2228360.2228509
21. R. Ye, T. Wang, F. Yuan, R. Kumar and Q. Xu, "On reconfiguration-oriented approximate adder design and its application," *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, San Jose, CA, USA, 2013, pp. 48-54, DOI: https://doi.org/10.1109/ICCAD.2013.6691096
22. C. Liu, J. Han and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, Germany, 2014, pp. 1-4, DOI: https://doi.org/10.7873/DATE.2014.108
23. F. Farshchi, M. S. Abrishami and S. M. Fakhraie, "New approximate multiplier for low power digital signal processing," *The 17th CSI International Symposium on Computer Architecture & Digital Systems (CADS 2013)*, Tehran, Iran, 2013, pp. 25-30, DOI: https://doi.org/10.1109/CADS.2013.6714233
24. S. Hashemi, R. I. Bahar and S. Reda, "DRUM: A Dynamic Range Unbiased Multiplier for approximate applications," *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Austin, TX, USA, 2015, pp. 418-425, DOI: https://doi.org/10.1109/ICCAD.2015.7372600
25. P. Kulkarni, P. Gupta and M. Ercegovac, "Trading Accuracy for Power with an Underdesigned Multiplier Architecture," *2011 24th Internatioal Conference on VLSI Design*, Chennai, India, 2011, pp. 346-351, DOI: https://doi.org/10.1109/VLSID.2011.51
26. K. Nepal, Y. Li, R. I. Bahar and S. Reda, "ABACUS: A technique for automated behavioural synthesis of approximate computing circuits," *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, Germany, 2014, pp. 1-6, DOI: https://doi.org/10.7873/DATE.2014.374
27. Chaofan Li, Wei Luo, S. S. Sapatnekar and Jiang Hu, "Joint precision optimisation and high level synthesis for approximate computing," *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA, 2015, pp. 1-6, DOI: https://doi.org/10.1145/2744769.2744863
28. Rahimi, Abbas & Benini, Luca & Gupta, Rajesh. (2017). An Approximation Workflow for Exploiting Data-Level Parallelism in FPGA Acceleration. DOI: https://doi.org/10.1007/978-3-319-53768-9_10
29. Y. Zhou, J. Lin and Z. Wang, "Energy efficient SVM classifier using approximate computing," *2017 IEEE 12th International Conference on ASIC (ASICON)*, Guiyang, China, 2017, pp. 1045-1048, DOI: https://doi.org/10.1109/ASICON.2017.8252658
30. Doyu, Hiroshi & Morabito, Roberto & Brachmann, Martina. (2021). A TinyMLaaS Ecosystem for Machine Learning in IoT: Overview and Research Challenges. 1-5. DOI: https://doi.org/10.1109/VLSI-DAT52063.2021.9427352
31. Chippa, Vinay & Chakradhar, Srimat & Roy, Kaushik & Raghunathan, Anand. (2013). Analysis and characterisation of inherent application resilience for approximate computing. Proceedings - Design Automation Conference. 1-9. DOI: https://doi.org/10.1145/2463209.2488873
32. Pruthvy Yellu, Novak Boskov, Michel A. Kinsy, and Qiaoyan Yu. 2019. Security Threats in Approximate Computing Systems. In Proceedings of the 2019 Great Lakes Symposium on VLSI (GLSVLSI '19). Association for Computing Machinery, New York, NY, USA, 387–392. DOI: https://doi.org/10.1145/3299874.3319453
33. H. Younes, A. Ibrahim, M. Rizk and M. Valle, "Algorithmic Level Approximate Computing for Machine Learning Classifiers," *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Genoa, Italy, 2019, pp. 113-114, DOI: https://doi.org/10.1109/ICECS46596.2019.8964974
34. H. Younes, A. Ibrahim, M. Rizk and M. Valle, "Data Oriented Approximate K-Nearest Neighbor Classifier for Touch Modality Recognition," *2019 15th Conference on Ph.D Research in Microelectronics and Electronics (PRIME)*, Lausanne, Switzerland, 2019, pp. 241-244, DOI: https://doi.org/10.1109/PRIME.2019.8787753
35. R. Zhang, N. Uetake, T. Nakada and Y. Nakashima, "Design of Programmable Analog Calculation Unit by Implementing Support Vector Regression for Approximate Computing," in *IEEE Micro*, vol. 38, no. 6, pp. 73-82, 1 Nov.-Dec. 2018, DOI: https://doi.org/10.1109/MM.2018.2873953
36. 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," *20th International Symposium on Quality Electronic Design (ISQED)*, Santa Clara, CA, USA, 2019, pp. 1-7, DOI: https://doi.org/10.1109/ISQED.2019.8697584
37. Barbareschi, Mario & Barone, Salvatore & Mazzocca, Nicola. (2021). Advancing synthesis of decision tree-based multiple classifier systems: an approximate computing case study. Knowledge and Information Systems. 63. DOI: https://doi.org/10.1007/s10115-021-01565-5
38. Pasandi, Ghasem & Nazarian, Shahin & Pedram, Massoud. (2019). Approximate Logic Synthesis: A Reinforcement Learning-Based Technology Mapping Approach. DOI: https://doi.org/10.1109/ISQED.2019.8697679
39. S. Hashemi, H. Tann, F. Buttafuoco and S. Reda, "Approximate Computing for Biometric Security Systems: A Case Study on Iris Scanning," *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, Germany, 2018, pp. 319-324, DOI: https://doi.org/10.23919/DATE.2018.8342029
40. L. Buşoniu, D. Ernst, B. De Schutter and R. Babuška, "Approximate reinforcement learning: An overview," *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, Paris, France, 2011, pp. 1-8, DOI: https://doi.org/10.1109/ADPRL.2011.5967353
41. Ravi Kanth, K. V., et al. "Dimensionality Reduction for Similarity Searching in Dynamic Databases." Computer Vision and Image Understanding, vol. 75, no. 1-2, Jul. 1999, pp. 59-72. DOI: https://doi.org/10.1006/cviu.1999.0762
42. Goiri, Iñigo & Bianchini, Ricardo & Nagarakatte, Santosh & Nguyen, Thu. (2015). ApproxHadoop: Bringing Approximations to MapReduce Frameworks. ACM SIGPLAN Notices. 50. 383-397. DOI: https://doi.org/10.1145/2775054.2694351
43. M. Shafique, W. Ahmad, R. Hafiz and J. Henkel, "A low latency generic accuracy configurable adder," *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA, 2015, pp. 1-6, DOI: https://doi.org/10.1145/2744769.2744778
44. S. Venkataramani, S. T. Chakradhar, K. Roy and A. Raghunathan, "Approximate computing and the quest for computing efficiency," *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA, 2015, pp. 1-6, DOI: https://doi.org/10.1145/2744769.2744904
45. A. Ranjan, S. Venkataramani, X. Fong, K. Roy and A. Raghunathan, "Approximate storage for energy efficient spintronic memories," *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA, 2015, pp. 1-6,

DOI: https://doi.org/10.1145/2744769.2744799
46. Y. Fang, H. Li and X. Li, "SoftPCM: Enhancing Energy Efficiency and Lifetime of Phase Change Memory in Video Applications via Approximate Write," *2012 IEEE 21st Asian Test Symposium*, Niigata, Japan, 2012, pp. 131-136, DOI: https://doi.org/10.1109/ATS.2012.57
47. Sutherland, M.; San Miguel, J.; Enright Jerger, N. Texture Cache Approximation on GPUs. In Proceedings of the Workshop on Approximate Computing (WAPCO'15), Paderborn, Germany, 15–16 October 2015; pp. 1–3. https://www.researchgate.net/publication/281492019_Texture_Cache_Approximation_on_GPUs#fullTextFileContent
48. F. Sampaio, M. Shafique, B. Zatt, S. Bampi and J. Henkel, "Approximation-aware Multi-Level Cells STT-RAM cache architecture," *2015 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, Amsterdam, Netherlands, 2015, pp. 79-88, DOI: https://doi.org/10.1109/CASES.2015.7324548
49. Hsiao, Chih-Chieh & Chu, Slo-Li & Chen, Chen-Yu. (2013). Energy-aware hybrid precision selection framework for mobile GPUs. Computers & Graphics. 37. 431–444. DOI: https://doi.org/10.1016/j.cag.2013.03.003
50. C. Rubio-González *et al.*, "Precimonious: Tuning assistant for floating-point precision," *SC '13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, Denver, CO, USA, 2013, pp. 1-12, DOI: https://doi.org/10.1145/2503210.2503296
51. R. S. Amant *et al.*, "General-purpose code acceleration with limited-precision analogue computation," *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, Minneapolis, MN, USA, 2014, pp. 505-516, DOI: https://doi.org/10.1109/ISCA.2014.6853213
52. V. K. Chippa, D. Mohapatra, K. Roy, S. T. Chakradhar and A. Raghunathan, "Scalable Effort Hardware Design," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 9, pp. 2004-2016, Sept. 2014, DOI: https://doi.org/10.1109/TVLSI.2013.2276759

## AUTHOR'S PROFILE

**Sreelakshmi Vadlamudi** is a research scholar in the Electronics & Communication Engineering department under the JNTUK, Kakinada, Andhra Pradesh. She is currently working as an ASSISTANT PROFESSOR in the Electronics & Communication Engineering department at Seshadri Rao Gudlavalleru Engineering College. She completed a UG in Electronics and Communication Engineering in 2004 from JNTU, Hyderabad. She completed her PG in VLSI System Design in 2011 at JNTU, Hyderabad.

**Dr. Y. Syamala** received her B.E. and M.E. from Bharathiyar University and Anna University in 2001 and 2005, respectively. She received a Ph.D from JNTUH, Hyderabad in 2014. She has been working as a Professor at Seshdri Rao Gudlavalleru Engineering College, Gudlavalleru. Her research interests include low-power VLSI design, digital design, and testing.

**Madhu Ramarakula** received the Bachelor of Engineering (BE) degree in Electronics and Communication Engineering from Osmania University, Hyderabad, India, in 2003, and the Master of Technology (MTech) degree in Communication Systems from Jawaharlal Nehru Technological University, Hyderabad, India, in 2009. He received his PhD in Electronics and Communication Engineering from Andhra University, Visakhapatnam, India, in 2014. He is an assistant professor in the Department of Electronics and Communication Engineering, University College of Engineering Kakinada (A), JNTUK, Kakinada, India. He has 10 years of teaching experience. He has published over 40 research papers in various reputed national and international journals and conferences. His research interests include mobile and cellular communications, antennas, satellite communications, and Global Navigation Satellite Systems (GNSS). He is a Member of IEEE.

---