

Effat University Repository

High-performance, energy-efficient, and memory-efficient FIR filter architecture utilizing 8x8 approximate multipliers for wireless sensor network in the Internet of Things

Authors	J Charles, Rajesh Kumar;Majid, M A;Vinod, Kumar
DOI	https://doi.org/10.1016/j.memori.2022.100017
Download date	2026-04-12 12:44:21
Link to Item	http://hdl.handle.net/20.500.14131/627



High-performance, energy-efficient, and memory-efficient FIR filter architecture utilizing 8x8 approximate multipliers for wireless sensor network in the Internet of Things

Charles Rajesh Kumar J. ^{a,*}, D. Vinod Kumar ^b, M.A. Majid ^c

^a Department of Electronics and Communication Engineering, Vinayaka Mission's Kirupananda Variyar Engineering College, Vinayaka Mission's Research Foundation (Deemed to be University), Salem, Tamil Nadu, India

^b Department of Biomedical Engineering, Vinayaka Mission's Kirupananda Variyar Engineering College, Vinayaka Mission's Research Foundation (Deemed to be University), Salem, Tamil Nadu, India

^c Department of Electrical and Computer Engineering, College of Engineering, Effat University, PO Box 34689, Jeddah, 21478, Saudi Arabia

ARTICLE INFO

Keywords:

WSN
IoT
Approximate multiplier
Approximate adders
FIR filter
Wallace tree
Error reduction

ABSTRACT

IoT uses wireless sensor networks (WSN) to deploy many sensors to track environmental and physical parameters. The WSN measurements are frequently contaminated and altered by noise. The noise in the signal increases the sensor node's computation and energy utilization, resulting in less longevity of the sensor node. The Finite Impulse Response (FIR) filter is commonly employed in WSN to pre-process sensed signals to remove noise from the sensed signals using delay elements, multipliers, and adders. Traditional multiplier-based FIR filter designs result in hardware-intensive multipliers that consume a lot of energy, and area and have low computation speed. These drawbacks make them unsuitable for IoT-based WSN systems with stringent power efficiency necessities. Approximate computing enhances the energy efficiency of an FIR filter. Arithmetic circuits utilizing approximate computing improve the hardware performance, with some loss of accuracy to save energy utilization and boost speed. A novel approximate multiplier architecture employing a fast and straightforward approximation adder is proposed in this study. Approximate multiplier M1 using OR gate and approximate multiplier M2 using proposed approximate adders are compared. The proposed approximate adder is suited for building an adder tree to accumulate partial product (PP) because it is less complicated than traditional adders. Compared to a one-bit-full adder, the critical path delay (CPD) is reduced significantly in the proposed methods. The accuracy comparison of M1, M2 and Wallace tree using the normalized mean error distance (NMED), the mean relative error distance (MRED), the maximum error (ME), and the error rate (ER) with the number of bits utilized for reducing error. For the area (delay) optimized circuit, when the bit used is 4, the delay is 0.4 ns for M1, 0.43 ns for M2, and 1.08 ns for the Wallace tree multiplier. For the delay (area) optimized circuit, when the bit used is 4, the delay is 0.16 ns for M1, 0.16 ns for M2, and 0.40 ns for the Wallace tree multiplier. To more accurately evaluate performance at the circuit level, the PDP and ADP are computed. The NMED, MRED, ME, and ER versus PDP and ADP are computed. The proposed multipliers M1 and M2 are compared with existing approximate multipliers. When an equivalent MRED, NMED, or ER is taken into account, M1 has the smallest ADP and PDP among other multipliers. The very low likelihood of a significant ED occurring is indicated by the small values of NMED and MRED in M1 and M2. The proposed solutions effectively reduce delay, area, and power while maintaining increased accuracy and performance.

1. Introduction

Wireless Sensor Networks (WSN) are frequently employed in the Internet of Things (IoT) solutions [1]. IoT uses wireless sensor networks to deploy many sensors to track environmental and physical parameters. The contemporary digital era has undergone a paradigm shift with different IoT-enabled interconnected gadgets that link everything from devices and people to intelligent systems. On the other hand,

WSN, which has a variety of sensor nodes (SN), is a critical component of IoT since it can sense physical attributes. A WSN includes many sensor nodes organized in topographical patterns such as the mesh, ring, and star. These nodes are generally powered by batteries or solar panels and are designed to use minimal energy. The sensor nodes' radio transmission range is often only a few meters, and sensor nodes will have to forward data from sensors to the gateway, from which the data

* Corresponding author.

E-mail addresses: charlesece@yahoo.com (Charles Rajesh Kumar J.), vino.kd@gmail.com (D.V. Kumar), moabdulmajid@effatuniversity.edu.sa (M.A. Majid).

<https://doi.org/10.1016/j.memori.2022.100017>

Received 17 May 2022; Received in revised form 1 October 2022; Accepted 5 October 2022

is delivered to the cloud through a Wi-Fi network, cellular, or wired Ethernet. As a result, WSN data travels through multiple hops before reaching the base station or sink node. Building a wireless network is much less expensive than wiring all sensor nodes [2]. The sensor network can be created and changed into various topographies, such as star, mesh, and so on, using wireless technology very quickly. A wireless network also makes adding and removing sensor nodes easier than a wired network. A wireless sensor node consists of several components, one of which is the sensor [3]. A radio transceiver, a power supply, and a microcontroller are typically included. Sensor nodes are commonly used to monitor humidity, temperature, vibrations, movement, light, heat, wetness, and other variables [4]. All of the data from the WSN's sensors travels a multi-hop path to the gateway or base station, from whence it is directed to cloud IoT platforms like AWS-IoT. AWS-IoT-Core is a service that helps manage the connectivity between the AWS cloud and devices quickly and safely. AWS-IoT enables the gathering and processing device data, cleaning, and analysis to deliver actionable insights. During the data transmission, sensor signals are usually noisy [5]. White Gaussian noise and impact noise contaminates the sensor data. Changes in the sensor's functioning circumstances, immediate disruption, and undesirable high-frequency electrical noise make the signals noisy [6,7].

The contaminated noises have a detrimental effect on a WSN data fusion [8]. Noises should be filtered, and the actual properties of sensor signals must be preserved to the greatest extent possible to ensure the reliability and accuracy of sensing data. The Finite Impulse Response (FIR) filter is commonly employed as preprocessing step to remove noise from the sensed signals of WSN [9]. With more advancements in VLSI technology, FIR filters must be realized in real-time with minimal hardware requirements and delay. The FIR filter's output is finite for the input of finite length, such as impulse, and after a while, it returns to zero. The impulse response of the infinite impulse response (IIR) filter is infinite. Various digital signal processing applications (DSP) are digital image processing, audio signal processing, telecommunications, speech recognition, RADAR, SONAR, etc. DSP faces several obstacles, which include low energy needs, real-time limitations, and big data applications [10]. Novel strategies are used to develop an FIR filter that can process signals efficiently with low complex computations [11, 12]. Delay elements, multipliers, and adders are essential for creating FIR filters in hardware. The multiplier has the most significant effect on speed, energy, and area. The multiplier's speed governs the FIR filter speed and processor's execution time [13]. Traditional multiplier-based techniques result in hardware-intensive multipliers that consume a lot of energy and have low computation speed. These drawbacks make them unsuitable for IoT-based WSN systems with stringent power efficiency necessities [14]. Approximating computation in developing energy-efficient digital systems has become a viable option [15]. Approximate computing techniques are utilized in FIR filter design. Approximate computing finds application in signal processing, multimedia, data analytics, cryptography, pattern recognition, machine learning, scientific computing, etc. Approximate computing is a set of algorithms that return a potentially erroneous result instead of a promised accurate result which can be employed in situations where an approximate solution will suffice. It is used to increase their energy efficiency and performance [16]. Approximate circuitry may be a potential option for decreasing size, energy, and latency in digital designs that can accept the possible loss of precision, resulting in improved power efficiency and performance. In digital VLSI circuits, an adder is the most fundamental computing circuit. For approximate design, adders have been widely investigated. The new methodology for modeling, analyzing, and evaluating approximate adders has been developed, and new modeling methods for examining and assessing them have been discussed [17].

On the other hand, the development of approximate multipliers has received far less attention. The three parts in a multiplier are producing a partial product (PP), accumulation of PP, and carry propagation adder [18]. Approximate PP is calculated utilizing inaccurate

2×2 multiplier blocks, and the approximate PP is accumulated using accurate adders in an adder tree [19]. A carry-in-prediction approach creates approximate 4×4 , 8×8 bit, and 16×16 Wallace multipliers [20]. The speculative approximation adders can be employed in the final stage addition in a multiplier [21]. Approximate multipliers are designed for error-acceptant applications [22]. The static-segment multiplier [23] splits the multiplier into the multiplication part (MSB) and the non-multiplication part (LSB). These approximation multipliers are intended for use with unsigned numbers. A Booth algorithm is commonly used to create signed multiplication. Approximate techniques have been suggested for Booth multipliers whose width is fixed utilizing conditional probability techniques and can be extended to greater than 32-bit or large booth multipliers. The method achieves higher performance in accuracy and reduced area cost [24]. Approximation-array-multipliers are formed from accurate-array-multipliers by horizontal and vertical cuts of a carry-save adder [25]. Two signed 32-bit and 16-bit radix-8 Booth algorithms and approximate computation utilizing a 2-bit adder for designing an FIR adaptive filter to decrease PP and significantly reduce accumulation circuits [26]. Approximate computing has grown in importance to minimize memory usage, speed, and power consumption in embedded and high-performance systems. A primary function of WSN is data collecting. Due to communications bandwidth and power budget restrictions, approximate data collecting is a reasonable choice in many WSN applications. Due to the energy limitations of the sensor nodes, decreasing and balancing power usage are critical problems in data gathering to increase the lifespan of wireless sensor networks. The accuracy of the outcomes can be compromised in signal processing applications to save power and area. Approximation adders enhance FIR filters' size and power efficiency [27]. The development of the IoT creates the need for energy consumption efficiency. Due to the WSN's development with the IoT in recent years, the amount of data exchanged has increased. Power consumption is a significant issue due to the high density of devices frequently used in difficult-to-reach locations. IoT design faces considerable obstacles because of the devices' limited battery life and limited computing power. Additionally, the spectrum resources that are currently available are insufficient to handle the upcoming beyond 5G and IoT communication techniques. Future IoT and communication techniques will rely heavily on reliable, energy-efficient, rational, and data-informed AI models to automatically manage IoT networks and services. Energy efficiency and utilization are becoming crucial factors in assessing the performance and feasibility of various IoT devices as network and spatial device densities rise. Additionally, because of the increased communication volume caused by the large-scale deployment, there is a more significant energy requirement, which shortens the lifespan of IoT devices. Developing energy efficiency in such devices will reduce energy consumption and increase the lifespan of the systems.

A novel approximate multiplier architecture employing a fast and straightforward approximation adder is proposed in this study. The developed adder computes data in parallel by breaking the CPC by producing an error vector and approximate sum. Error mitigation techniques based on OR gates and approximation adders reduce the critical path delay more than a traditional one-bit full adder. An approximate adder is proposed for an energy-efficient and high-performance approximate PP accumulation tree for a multiplier. The suggested multipliers have minimal error distances, resulting in high accuracy. The suggested multipliers have substantially shorter critical paths than the typical Wallace tree. The proposed solutions effectively reduce delay and power while maintaining increased accuracy and performance. Below is a breakdown of the manuscript's content. The literature review and extant architectures are summarized in Section 2. Section 3 presents the proposed FIR adaptive filter design using an 8×8 approximate multiplier. Section 4 contains the results and comments, and Section 5 concludes the paper.

2. Related works

Approximation multiplier design primarily employs three approximate techniques: 1. approximation in the PP generation, 2. employing

truncation in the PP tree, and 3. Approximation adders and compressors are used for accumulating the PP, and the PP reducing phase of multiplication operations consumes more energy and occupies a significant silicon area. As a result, approximation computing has been created to save energy. Lavanya Maddiseti et al. [28] proposed an approximation method called probabilistic pruning is employed in this work. Compared to the energy usage of an identical multiplier, the simulation outcomes show that the energy utilization is reduced by 82.83 percent. Yufeng Xu et al. [29] proposed an area-efficient approximation multiplier with OR-based error compensation utilizing input re-ordered 4 to 2 compressors for low-energy design. Only two of the four inputs are concentrated by re-ordering the inputs, making the compressor more straightforward and with fewer gates. The proposed approach achieves 98.7% accuracy while consuming 44.72 percent less energy and 31.72 percent less space. Che-Wei Tung et al. [30] proposed 8×8 approximate multipliers utilizing high-order approximate compressors. Various weights use various compressors with varying accuracy to accumulate product terms and reduce energy consumption with minimal error. Higher-order approximation compressors such as 8 to 2 compressors are utilized for the middle significance weights to simplify the “carry chain’s” logic. E. Jagadeeswara Rao et al. [31] suggested the rounding method-based approximate multiplier in which rounding up the input operands to the next power of 2 is carried out to design an error-efficient system. An arithmetic module comprises subtraction, addition, and shifter units and processes the modified inputs. The sizes of the input operands vary from 8 to 32 bits. Simulation findings reveal that the latency and power usages are around 22% and 57%, better than equivalent approximate multipliers. Bharat Garg et al. [32] proposed an approximate multiplier based on a re-configurable rounding approach. The proposed multipliers decrease implementation complexity while also increasing power efficiency. The proposed method requires 59.8% reduced area, 54.7% reduced delay, and 32.5% reduced energy utilization over the filters with current multipliers. Seyed Amir Hossein Ejtahed et al. [33] proposed an approximate multiplier using an approximate compressor that utilizes only one gate. Compared to the present methods, the suggested method shows power usage and area are around 61% and 52% less, respectively. Mostafa Abbasmollaei et al. [34] proposed an approximate multiplier for unsigned numbers. Its goal is to lower all hardware metrics while maintaining excellent accuracy since it has a high level of configurability. It offers a broad spectrum of energy-saving choices that range from 35–85 percent, meeting the needs of most applications while staying within a reasonable power budget. Shaghayegh Vahdat et al. [35] proposed an approximate multiplier using the truncating method. The approximation multiplier functioned by calculating the outcome by truncating the intermediate results and employing scientific-binary representations of the operands. Compared to the identical multiplier, there is an average increase of 89.2 percent in power saving and a 74.9 percent reduction in area. This research proposes an energy-efficient, high-performance approximate PP accumulation tree for a multiplier utilizing a newly developed approximate adder. The suggested approximate adder ignores “carry propagation” by producing an error vector and an approximate sum. Two different approximate 8X8 multiplier designs, M1 and M2, are produced using OR gates and approximation adder-based error reduction strategies, respectively. It has been demonstrated that the proposed approximate multipliers dissipate less power than an exact Wallace multiplier that is speed-optimized. The suggested multipliers attain high accuracy due to minor error distances. Additionally, simulations have demonstrated that M2, at the expense of a longer delay and more power usage, has greater accuracy than M1. The proposed approximation multipliers are more accurate than earlier approximate models. The proposed solutions offer significant delay and power savings with a high degree of accuracy, in contrast to earlier designs that concentrate on decreasing delay and energy with sometimes unsatisfactory accuracy.

Table 1

The suggested adder-cell’s truth table.

S_j/E_j		$\bar{B}_j \bar{B}_{j+1}$			
		00	01	11	10
$\bar{A}_j \bar{A}_{j+1}$	00	0/0	X	X	X
	01	0/0	1/0	X	X
	11	1/0	1/1	1/0	0/0
	10	1/0	X	X	0/0

3. The proposed architecture

The two steps of a multiplication process are the production of PP and their addition. As a result, the multiplier’s speed primarily depends on how quickly we can add the partial products together after they are formed. If fewer PP is produced, the speed in generating PP is achieved. We require fast adder designs to accelerate the adding process among PP. The suggested approximation multiplier uses a simple tree of approximate adders for PP accumulation and employs error signals to compensate for errors to improve accuracy. The approximation design has evolved as a revolutionary model for obtaining energy-efficient DSP cores that exhibit satisfactory accuracy. The multiplier is the primary arithmetic unit in various signal processing designs, such as FIR filters utilized in the signal preprocessing stage (denoising) of IoT-based WSN. It has a significant effect on these cores’ effectiveness. As a result, this research proposes a novel energy-efficient approximation unsigned multiplier with error reduction strategies.

3.1. The proposed approximate -Adder

The suggested approximate-adder cuts the carry-propagation chain (CPC) because the data is processed parallel. Let the adder’s input operands in binary form be A and B. The sum is represented as S and the error vector as E. The j th LSB of S, E, A, and B are S_j , E_j , A_j , and B_j . When $\bar{B}_j = 1$, $\bar{A}_{j+1} = 1$, and $\bar{B}_{j+1} = 0$, CPC begins. Generally, the carry propagates to a higher bit in an accurate adder when $S_{j+1} = 0$. However, in the suggested approximate adder S_{j+1} is set to 1 and an error signal is produced as $E_{j+1} = 1$. As a result, the carry does not propagate to a higher bit. If $\bar{B}_j = 1$ then $C_j = 1$ and it only propagate to the next larger bit in $(j + 1)$ position. Table 1 contains the truth table for the proposed adder-cell. The letter “X” in the table denotes the absence of such a combination due to the input pre-processing. The input preprocessing functions are given in the equation as $\bar{A}_j = A_j + B_j$ and $\bar{B}_j = A_j B_j$. From the truth table, the logic function of S_j and E_j is given $S_j = \bar{B}_{j+1} + \bar{B}_j \bar{A}_j$ and $E_j = \bar{B}_j \bar{B}_{j+1} \bar{A}_j$. Substituting this value yields $S_j = (A_j \oplus B_j) + A_{j-1} B_{j-1}$ and $E_j = (A_j \oplus B_j) A_{j-1} B_{j-1}$.

Let $A = A_{k-1} A_{k-2}, \dots, A_1 A_0$, $B = B_{k-1} B_{k-2}, \dots, B_1 B_0$ then $\bar{S} = \bar{S}_{k-1} \bar{S}_{k-2} \dots \bar{S}_1 \bar{S}_0$. Now $\bar{S} = S_j + E_j$. The plus symbol indicate binary addition rather than OR operation. E is never a negative number and the approximate sum is less than accurate sum or equal to accurate sum. Fig. 1 shows the circuit diagram of full adder circuit and proposed accumulator adder cell.

3.2. The proposed approximate-multiplier

Fig. 2 illustrates the block diagram of the suggested approximate multipliers M1 and M2. The proposed multiplier includes the proposed approximate adders. Generally, any multiplier has generation of PP stage, accumulation of PP stage, and carry-propagation adder stage. Generally, an approximation multiplier is distinguished by the ease with which approximate adders can be used in PP accumulation. This has been demonstrated to generate poor performance since errors can accumulate, and correcting errors with present approximate adders is challenging. By using the error signal, the suggested approximate adder solves this issue. The data is processed in parallel by the n-bit

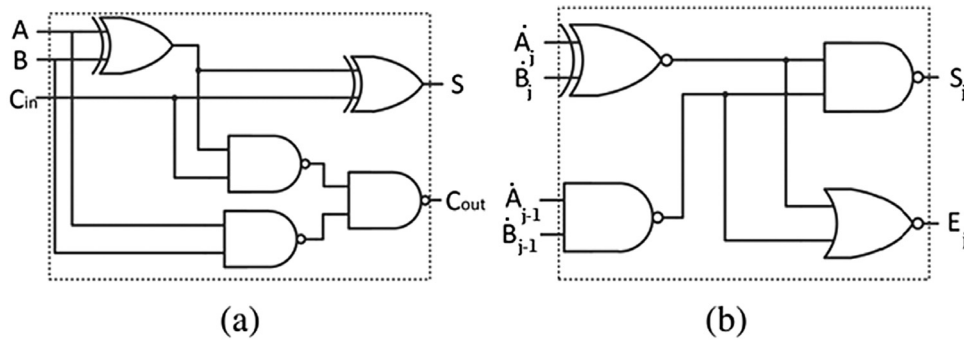


Fig. 1. (a) The traditional Full adder circuit. (b) The proposed Approximate-adder cell.

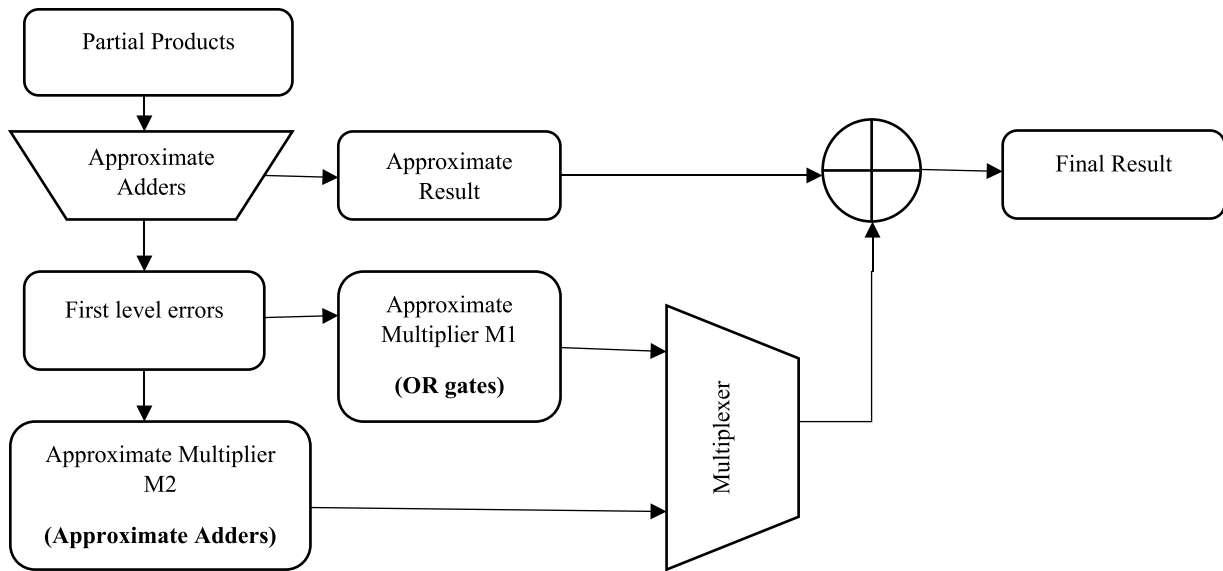


Fig. 2. The proposed 8 × 8 approximate multiplier.

adder resulting in a shorter critical path delay than a traditional one-bit full adder. Generally, the error rate of the approximation adder is considerably high, but the proposed approximate adder produces sum and error signals simultaneously, and errors in the final product are decreased.

The suggested approximate adder ignores “carry propagation” by producing an error vector and an approximate sum. For the accumulation of PP, an adder tree is used. Better accuracy in the product is generated by compensating for errors in the output using the tree’s error signal. At every level of the tree, the number of PP is decreased by a factor of two. It results in fewer memories and better energy-efficient memory schemes. Fig. 3 shows an approximate partial error recovery with the five most significant bits of the error vector. Power consumption in FIR filters can be minimized. The delay depends on the memory architecture and its internal structure, and the memory needed for storing the PP is lower; thus, speed is increased. It takes less time to access memory. The decreased memory requirement causes the delay to reduce. An approximate adder is proposed suited for building an adder tree to accumulate PP because it is less complicated than traditional adders the critical-path delay (CPD) is reduced significantly. The Dadda tree and Wallace tree design utilizing full adders, half adders, and compressors are complex because it occupies a large circuit area even though the critical path is reduced. The design complexity increases when different size multipliers are considered, but the suggested method is uncomplicated for different multiplier sizes.

3.3. The reduction of errors

The approximate adder A_j produces an error signal (E_j) and approximate sum (S_j). Here $j = 1, 2, 3, 4, 5, 6, 7$. The error signal minimizes the multiplier’s inaccuracy. An approximate multiplier with the recovery of partial error utilizing the five most significant bits of the error vector is shown in Fig. 3. The suggested approximate multiplier architecture can be set utilizing OR gates, referred to as approximate multiplier M1. Another error reduction approximate adder circuit is referred to as approximate multiplier M2. Different error recovery levels can also be accomplished by utilizing various MSBs to recover errors in both M1 and M2. Instead of applying an error-reducing circuit to every adder’s output, it is applied to the final multiplication output. Accumulation of error and recovery of error are the two steps involved in error-reducing steps. The accumulated error signal is a single error vector, and the adder adds the accumulated error signals with the PP accumulation adder tree output. The M1 and M2 are two approximate error accumulation strategies presented in this work. Error accumulation tree for approximate multiplier M1 is shown in Fig. 4 and M2 in Fig. 5.

3.3.1. Error accumulation tree for approximate multiplier M1

Each approximate adder A_j produce error vector E_j and sum vector S_j and $j = 1, 2, 3, 4, 5, 6, 7$. An approximation accumulation of error is employed to lower complexity. The sum of accumulated error vectors represented as $E_1, E_2, E_3, \dots, E_7. E_j = E_1 OR E_2 OR E_3 OR E_4 \dots OR E_7$.

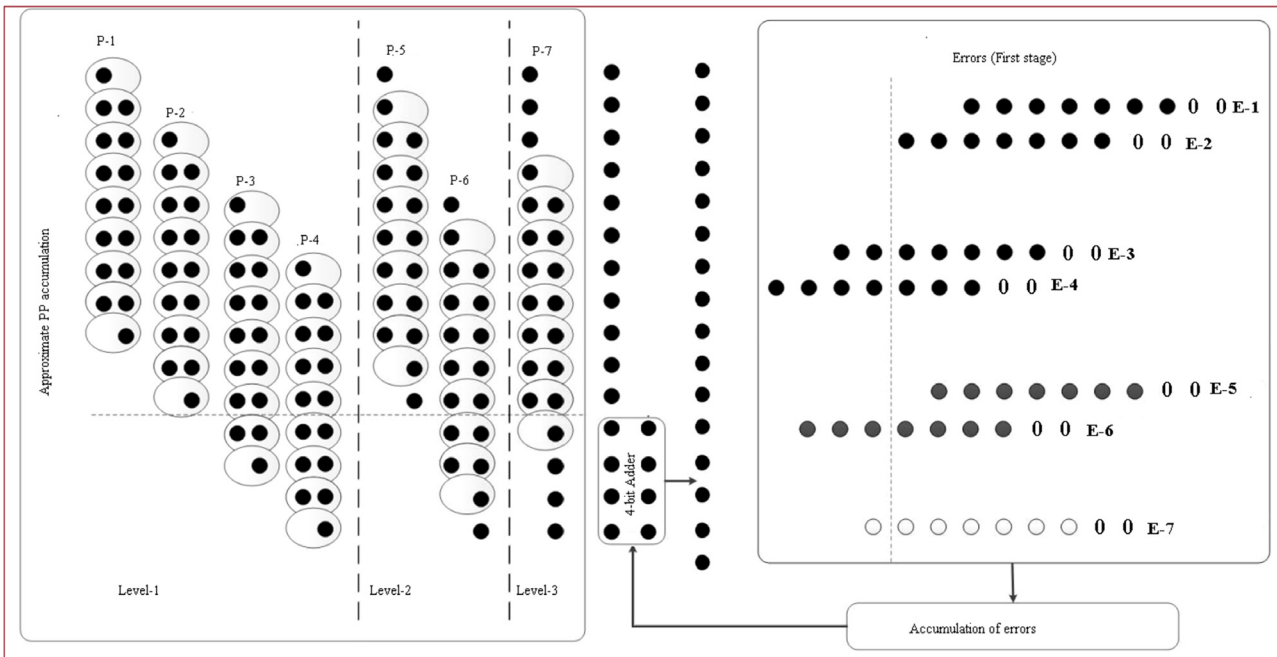


Fig. 3. An approximate multiplier with the recovery of partial error utilizing the five most significant bits of the error vector.

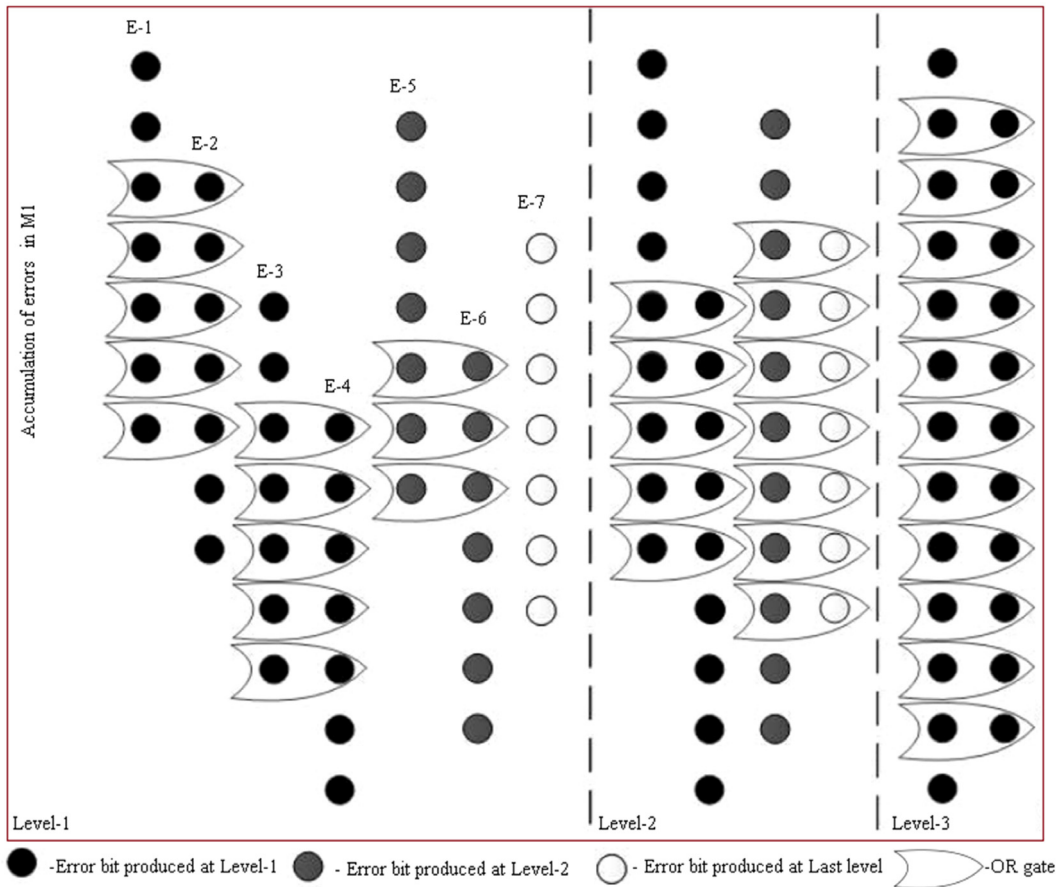


Fig. 4. Error accumulation tree for approximate multiplier M1.

A fast-accurate adder such as the CLA adder is utilized to add the output of the adder tree with the accumulated error vector. The overall complexity is reduced using only a few MSBs of the error signals needed for error compensation at the output. The number of MSBs is

determined by the amount of error compensation required. There are seven error vectors for an 8×8 adder tree, and not all 7 bits are utilized for error compensation, and only a few MBS are employed. Approximate error accumulation technique M1 is shown in Fig. 4. The

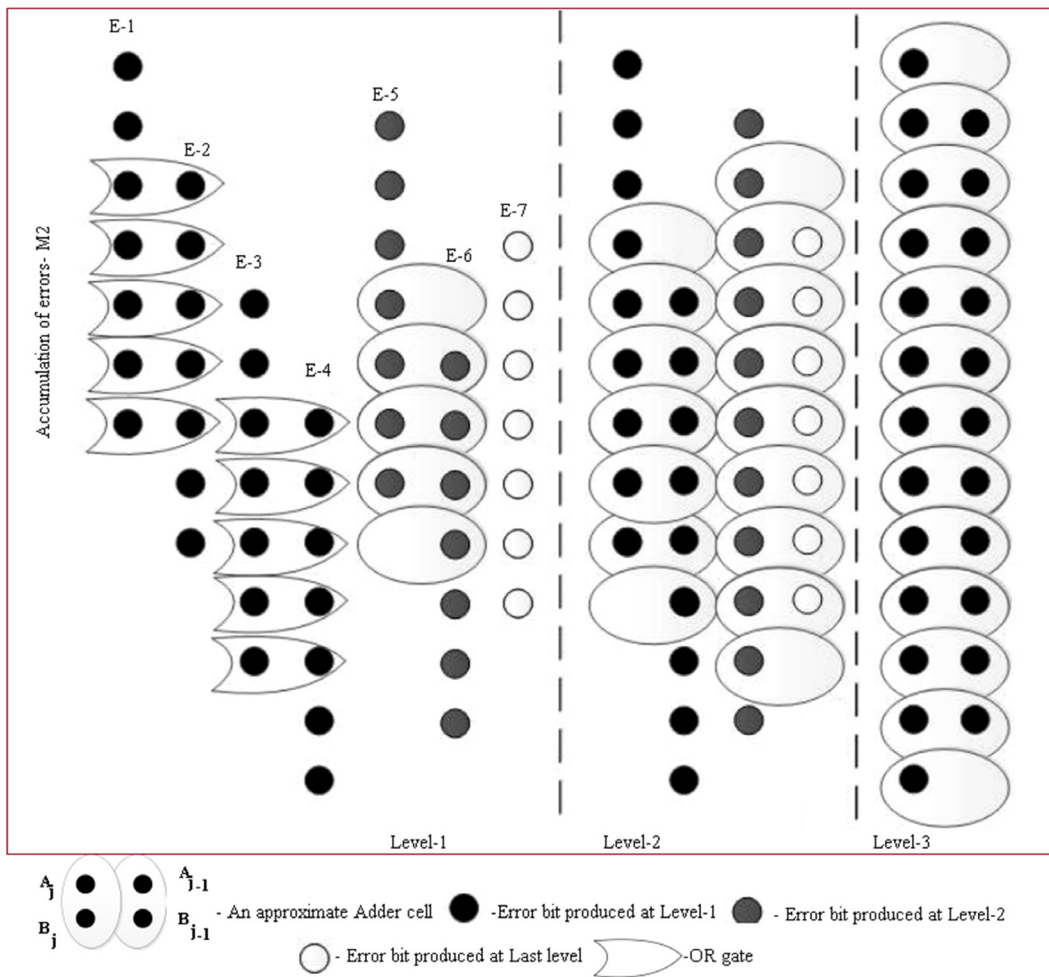


Fig. 5. Error accumulation tree for approximate multiplier M2.

LSB (2 bits) of every error vector (E_j) is not accumulated since no error is produced at the LSB (2 bits) of every approximate adder (A_j).

3.3.2. Error accumulation tree for approximate multiplier M2

Approximate error accumulation technique for M2 is shown in Fig. 5. Consider an 8×8 -multiplier having inputs A, B. Let the PP vectors accumulated by the first approximate adder be $A = r_0s_7, r_0s_6, r_0s_5, r_0s_4 \dots r_0s_0$ and $B = r_1s_7, r_1s_6, r_1s_5, r_1s_4 \dots r_1s_0$ which is shown as P1 in Fig. 2. If r_0 or r_1 is 0, there will be no error in this approximate adder because either B or A is 0. There will be an error only when $r_0 r_1 = 11$. When $r_0 r_1 = 11$ then $A_j = s_0$ and $B_j = s_{j-1}$. To compute E_j , $B_j = s_{j-1}$, $A_j = s_j$, $B_{j-1} = s_{j-2}$, $A_{j-1} = s_{j-1}$. If the value of $s_j s_{j-2} s_{j-1} = 011$ then $E_j = 1$ and this is based on the condition $A_{j-1} = B_{j-1} = 1$, $A_j \neq B_j$. Thus, an error only happens when the input has "011" as a bit sequence. There must be at least 3 bits among two errors in the proposed approximation multiplier. In level-1 of the PP tree, the two adjacent approximate adders will not have errors in the same column. The error vectors are represented as E1, E2, E3 and E4. An OR gate is employed for the two bits accumulation in the E1 and E2. An OR gate is employed for the two bits accumulation in the E3 and E4. The four vectors (E1, E2, E3, and E4) are compressed into two vectors. The error vectors E5, E6, and E7, are developed from the approximate summation of the PP instead of the PP. As a result, OR gates cannot appropriately accumulate them. If an error bit is one bit, then the adjacent bit is free of error, and in a row, there are no successive error bits. The CPC path is no longer than two bits for the accumulation of two error vectors. The suggested approximate adder correctly accumulates the error vectors E5 and E6 in level-1. Three error vectors are created after

the level-1 of accumulation of error, and these three error vectors are accumulated along with E7 from the previous stage using additional two approximate adders.

4. Results and discussion

Using Matlab, the functionalities of the suggested multipliers are implemented, and a thorough simulation is run for an approximate 8×8 multiplier. Table 2 compares the accuracy of M1 and M2 in terms of the mean relative error distance (MRED) and normalized mean error distance (NMED) when employing various numbers of MSBs to reduce errors. Table 3 compares the accuracy of M1 and M2 in terms of the error rate (ER), normalized mean error distance, and the maximum error (ME) when employing various numbers of MSBs to reduce errors. When the number of bits utilized for reducing error increases, the value of NMED and MRED significantly decreases. When the number of bits used for reducing error increases, the value of ER drop sharply. When the number of bits utilized for reducing error increases, the ME value does not decrease much. The value of NMED, ER, MRED, and ME drop to zero when the number of bits utilized for reducing error is 14. For the exact value of the number of bits used for reducing error, M2 performs better than M1 in terms of NMED, ER, MRED, and ME. When the number of bits used for reducing error is 8, the value of NMED is 0.17% for M2, and for M1, it is 0.30%. When the number of bits used for reducing error is 14, the error rate for M2 is 5.8%, and for M1, it is 17.6%. The accuracy comparison of M1, M2 and Wallace tree using the normalized mean error distance (NMED), the mean relative error

Table 2The accuracy of approximate multiplier M1 and M2 in terms of $\log_2(\text{NMED})$ and $\log_2(\text{MRED})$.

Approximate multipliers	$\log_2(\text{NMED})$ versus Number of bits utilized for reducing error						$\log_2(\text{MRED})$ versus Number of bits utilized for reducing error					
	Number of bits used for error reduction (4, 6, 8, 10, 12, 14)						Number of bits used for error reduction (4, 6, 8, 10, 12, 14)					
	4	6	8	10	12	14	4	6	8	10	12	14
Approximate multiplier M1	-6	-7	-8.37	-8.5	-8.5	-8.5	-3.5	-4.5	-5.5	-6.5	-7	-7
Approximate multiplier M2	-6.1	-7.1	-9.2	-9.3	-9.3	-9.3	-3.51	-4.51	-5.52	-7.6	-8	-8

Table 3The accuracy of approximate multiplier M1 and M2 in terms of $\log_2(\text{ER})$ and $\text{ME} \log_2(\text{ME})$.

Approximate multipliers	$\log_2(\text{ER})$ versus Number of bits utilized for reducing error						$\log_2(\text{ME})$ versus Number of bits utilized for reducing error					
	Number of bits used for error reduction (4, 6, 8, 10, 12, 14)						Number of bits used for error reduction (4, 6, 8, 10, 12, 14)					
	4	6	8	10	12	14	4	6	8	10	12	14
Approximate multiplier M1	-0.4	-0.45	-0.55	-1	-2	-2.5	-2.50	-2.51	-2.52	-2.53	-2.54	-2.55
Approximate multiplier M2	-0.41	-0.46	-0.56	-1.5	-2.9	-4.1	-2.52	-2.53	-2.54	-2.55	-2.56	-2.57

Table 4Comparison of performance of the proposed 8×8 approximate multiplier with accurate Wallace tree multiplier.

Multiplier		Optimized for area (μm^2)					Optimized for delay (ns)				
		Number of bits utilized for reducing errors (4, 5, 6, 7, 8)					Number of bits utilized for reducing errors (4, 5, 6, 7, 8)				
		4	5	6	7	8	4	5	6	7	8
Approximate multiplier M1	Delay (ns)	0.4	0.52	0.61	0.7	0.82	0.16	0.2	0.25	0.26	0.29
	Power (μW)	90	98	110	115	127	210	230	270	260	360
	Area (μm^2)	120	125	140	149	158	310	280	320	320	430
Approximate multiplier M2	Delay (ns)	0.43	0.68	0.71	0.8	0.89	0.16	0.2	0.25	0.29	0.29
	Power (μW)	90	100	110	125	130	210	200	260	310	360
	Area (μm^2)	120	132	141	163	180	310	260	310	370	450
Accurate Wallace tree Multiplier	Delay (ns)	1.06	1.07	1.08	1.09	1.10	0.41	0.42	0.43	0.44	0.45
	Power (μW)	160.01	160.02	160.03	160.04	160.05	470.01	470.02	470.03	470.04	470.05
	Area (μm^2)	186.01	186.02	186.03	186.04	186.05	500.01	500.02	500.04	500.05	500.06

distance (MRED), the maximum error (ME), and the error rate (ER) with number of bits utilized for reducing error.

VHDL and Synopsys design compiler (SDC) is utilized to implement the proposed 8×8 multiplier and synthesis respectively. Simulations are run using a voltage level of 1 V and a temperature of 25 °C. The 28 nm library is used to provide the modules for constructing the multiplier circuitry. SDC tool reports these multipliers' critical path delays. The Primetime-PX or Power compiler tool uses 10 million random input combinations with a clock duration of 2 ns to determine the power dissipation. The simulation results illustrate that the suggested multipliers have shorter critical paths than the typical Wallace tree [36]. Approximate multiplier M1 has a faster response time than approximate multiplier M2 since M1 utilizes OR gate for reducing error. Area optimization and delay optimization are performed. Table 4 compares the performance of the proposed 8×8 approximate multiplier M1 and M2 with an accurate Wallace-tree multiplier. Simulations have demonstrated that M2 has greater accuracy than M1 at the expense of a more extensive delay and higher power consumption. When the number of bits used for reducing error is increased the critical path delays of M1 and M2 increase. M1 shows shorter delay than M2 for the same number of MSB used for reducing error.

For the area (delay) optimized circuit, when the bit used is 4, the delay is 0.4 ns for M1, 0.43 ns for M2 and 1.08 ns for Wallace tree multiplier. For the area optimized circuit, when the bit used is 4, it is observed that M1 is 63 percent faster and M2 is 60 percent faster than the Wallace multiplier. For the area optimized circuit, when the bit used is 4, it is observed that M1 and M2 save 42 percent energy than the Wallace multiplier. For the area optimized circuit, when the bit used is 4, it is observed that M1 and M2 save 34 percent area than the Wallace multiplier. For the area optimized circuit, when the bit

used is 8, it is observed that M1 save 21 percent energy and M2 save 17 percent energy than the Wallace multiplier. For the area optimized circuit, when the bit used is 8, it is observed that M1 and M2 save approximately 20 percent energy than the Wallace multiplier. For the area optimized circuit, when the bit used is 4, it is observed that M1 uses smaller area of 23% than the accurate design. For the area optimized circuit, when the bit used is larger than 8, it is observed that M2 uses larger area than the accurate design. For the delay (area) optimized circuit, when the bit used is 4, the delay is 0.16 ns for M1, 0.16 ns for M2 and 0.40 ns for Wallace tree multiplier. For the delay optimized circuit, when the bit used is 4, it is observed that M1 and M2 are 60 percent faster than the Wallace multiplier [37]. The energy dissipation and multiplier area exhibit the same pattern as the delay. For the delay optimized circuit, when the bit used is 4, it is observed that M1 save 53 percent energy and M2 save 38 percent energy than the Wallace multiplier. The Power-Delay-Product (PDP) is a measurement of power and is defined by the product of the average power and the gate delay. When optimized for the delay, power is reduced. The Area-Delay-Product (ADP) estimates the trade-off between the average latency of data transmission and area overhead. When optimized for the delay, the area is reduced. The proposed methods are compared with the existing approximate multipliers. The power-delay product (PDP) is compared with existing methods and shown in Table 5. The area-delay-product (ADP) is compared with existing methods and shown in Table 6. The accuracy characteristics are obtained by Monte Carlo simulation with 10^8 random input combinations. The PDP and ADP are determined to evaluate performance at the circuit level more accurately. The mean relative error distance (MRED), the error rate (ER), normalized mean error distance and the maximum error (ME), normalized mean error distance (NMED) are computed. M2 has a better performance than M1 in terms of ER, MRED and NMED.

Table 5The PDP (fJ) versus MRED (\log_2) and PDP versus ER (\log_2) of the proposed method and the existing methods.

Multiplier	PDP versus MRED Area optimized					PDP versus MRED Delay optimized					PDP versus ER Area optimized					PDP versus ER Delay optimized				
	PDP (40,60,80,100,120)					PDP (50,75,100,125,150)					PDP (40,60,80,100,120)					PDP (50,75,100,125,150)				
	40	60	80	100	120	50	75	100	125	150	40	60	80	100	120	50	75	100	125	150
M1	-3.3	-4	-5	-5.5	-6.5	-3.7	-5	-5.6	-6.5	-7	-0.30	-0.38	-0.50	-0.7	-1	-0.32	-0.52	-0.62	-0.90	-1.01
M2	-3.2	-3.6	-4.4	-5	-6.2	-3.6	-4.4	-5	-5.7	-6.4	-0.29	-0.36	-0.40	-0.50	-0.8	-0.31	-0.39	-0.44	-0.61	-0.82
[22]	-2.9	-3.5	-4.2	-4.7	-5.9	-2.8	-3.7	-4.4	-5	-5.5	-0.19	-0.12	-0.14	-0.20	-0.3	-0.2	-0.12	-0.14	-0.18	-0.2
[23]	-2.7	-3.4	-4.1	-4.6	-5.2	-2.6	-3.2	-3.5	-4.2	-4.5	-0.02	-0.06	-0.1	-0.18	-0.2	-0.04	-0.06	-0.10	-0.12	-0.14

Table 6The ADP (μm^2 ns) versus NMED (\log_2) and ADP (μm^2 ns) versus ME (\log_2) of the proposed method and the existing methods.

Multiplier	ADP versus NMED Area optimized					ADP versus NMED Delay optimized					ADP versus ME Area optimized					ADP versus ME Delay optimized				
	ADP (40,60,80,100,120)					ADP (50,75,100,125,150)					ADP (40,60,80,100,120)					ADP (50,75,100,125,150)				
	40	60	80	100	120	50	75	100	125	150	40	60	80	100	120	50	75	100	125	150
M1	-5.5	-6.2	-7	-7.8	-8.4	-5.9	-7	-8	-8.5	-9	-2.6	-2.6	-2.7	-2.8	-2.8	-2.7	-2.7	-2.75	-2.8	-2.8
M2	-5.4	-5.8	-6.5	-7.5	-8	-5.8	-6.8	-7.5	-8.2	-8.8	-2.9	-2.9	-3	-3.1	-3.2	-2.8	-3	-3.2	-3.2	-3.2
[22]	-5.3	-5.7	-6.3	-7	-7.8	-5.3	-6	-6.7	-8	-8	-3.2	-3.5	-4.2	-5	-5.5	-3.2	-3.8	-4.7	-5.5	-6
[23]	-5	-5.4	-6.2	-6.6	-7.7	-5	-5.2	-5.6	-6	-6.5	-2.9	-3.1	-4	-4.4	-4.8	-2.8	-3	-3.2	-3.3	-3.3

5. Conclusion

A novel 8×8 approximate multiplier architecture employing a fast and straightforward approximation adder is proposed in this study. The developed adder computes data in parallel by breaking the CPC by producing an error vector and approximate sum. Error mitigation techniques based on OR gates and approximation adders reduce the critical path delay more than a traditional one-bit full adder. An approximate adder is proposed for an energy-efficient and high-performance approximate PP accumulation tree for a multiplier. Simulations have demonstrated that M2 has greater accuracy than M1 at the expense of a more extensive delay and higher power consumption.

M1 shows shorter delay than M2 for the same number of MSB used for reducing error. The energy dissipation and multiplier area exhibit the same pattern as the delay. The suggested multipliers have minimal error distances, resulting in high accuracy. The suggested multipliers have substantially shorter critical paths than the typical Wallace tree. The accuracy comparison of M1, M2 and Wallace tree using the normalized mean error distance (NMED), the mean relative error distance (MRED), the maximum error (ME), and the error rate (ER) with number of bits utilized for reducing error. For the area (delay) optimized circuit, when the bit used is 4, the delay is 0.4 ns for M1, 0.43 ns for M2 and 1.08 ns for Wallace tree multiplier. For the delay (area) optimized circuit, when the bit used is 4, the delay is 0.16 ns for M1, 0.16 ns for M2 and 0.40 ns for Wallace tree multiplier. To more accurately evaluate performance at the circuit level, the PDP and ADP are computed. The NMED, MRED, ME and ER versus PDP and ADP are computed. The proposed multipliers M1 and M2 are compared with existing approximate multipliers [22,23]. When an equivalent MRED, NMED, or ER is taken into account, M1 has the smallest ADP and PDP among M2, [22,23]. The very low likelihood of a significant ED occurring is indicated by the small values of NMED and MRED in M1 and M2. The proposed solutions effectively reduce delay and power while maintaining increased accuracy and performance. Noise and interferences frequently impacting WSNs are reduced using the FIR filter to decrease energy consumption and increase the lifetime of the WSN utilized in IoT. The proposed 8×8 approximate multiplier is utilized to design an FIR filter employed for signal denoising applications.

Data availability

No data was used for the research described in the article.

References

- [1] J. Charles Rajesh Kumar, Ahmed Almasarani, M.A. Majid, 5G-wireless sensor networks for smart grid-accelerating technology's progress and innovation in the Kingdom of Saudi Arabia, *Procedia Comput. Sci.* 182 (2021) 46–55.
- [2] J. Charles Rajesh Kumar, D. Vinod Kumar, B. Mary Arunsi, D. Baskar, M.A. Majid, Energy-efficient adaptive clustering and routing protocol for expanding the life cycle of the IoT-based wireless sensor network, in: 2022 6th International Conference on Computing Methodologies and Communication, ICCMC, 2022, pp. 328–336.
- [3] J. Charles Rajesh Kumar, D. Baskar, B. Mary Arunsi, D. Vinod Kumar, Energy-efficient and secure IoT architecture based on a wireless sensor network using machine learning to predict mortality risk of patients with CoVID-19, in: 2021 6th International Conference on Communication and Electronics Systems (ICES), India, 2021, pp. 1853–1861.
- [4] A. Muthukrishnan, A.J. Charles Rajesh kumar, D. Vinod Kumar, M. Kanagaraj, Internet of image things-discrete wavelet transform and gabor wavelet transform based image enhancement resolution technique for IoT satellite applications, *Cognit. Syst. Res.* 57 (2018) 46–53.
- [5] Q. Zhang, H. Liu, X. Zhou, Y. Wang, Wavelet soft-threshold denoising method of power quality signal, *High Volt. Eng.* 32 (1) (2006) 99–101.
- [6] Z.-C. Liu, X.-G. Chen, Y.-F. Li, Detection and identification of abrupt changes for on-line sensor output signal, *Trans. Beijing Inst. Technol.* 26 (12) (2006) 1104–1108.
- [7] Z. Liu, Analysis of noise at coal face by fully-mechanized coal winning technology, *J. China Univ. Min. Technol.: Engl. Ver.* 13 (1) (2003) 113–116.
- [8] A.M. Rao, D.L. Jones, A denoising approach to multisensor signal estimation, *IEEE Trans. Signal Process.* 48 (5) (2000) 1225–1234.
- [9] D. Bhat, A. Kaur, S. Singh, Wireless sensor network specific low power FIR filter design and implementation on FPGA, in: 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), 2015, pp. 1534–1536.
- [10] M.H. Nassralla, N. Akl, Z. Dawy, A clustering-based approach for designing low complexity FIR filters, *IEEE Signal Process. Lett.* 28 (2021) 299–303.
- [11] Yi Li, Jiayang Zhao, Wei Xu, Guiling Sun, A low computational complexity scheme for designing linear phase sparse FIR filters, *Circuits Systems Signal Process.* 41 (2022) 1550–1562.
- [12] Savita Srivastava, Atul Kumar Dwivedi, Deepak Nagaria, Low complexity multi-objective finite impulse response filter design using salp swarm algorithm and its improved version, *Int. J. Numer. Modelling, Electron. Netw. Devices Fields* 34 (6) (2021) e2914.
- [13] V. Dyana Christilda, A. Milton, Speed, power and area efficient 2D FIR digital filter using vedic multiplier with predictor and reusable logic, *Analog Integr. Circuits Signal Process.* 108 (2021) 323–333.
- [14] M. Alawad, M. Lin, Memory-efficient probabilistic 2-D finite impulse response (FIR) filter, *IEEE Trans. Multi-Scale Comput. Syst.*, 4 (1) 69–82.
- [15] V. Leon, K. Pekmestzi, D. Soudris, Exploiting the potential of approximate arithmetic in DSP & AI hardware accelerators, in: 2021 31st International Conference on Field-Programmable Logic and Applications, FPL, 2021, pp. 263–264.
- [16] A. Sokolova, et al., Maccelerator: Approximate arithmetic unit for computational acceleration, in: 2021 22nd International Symposium on Quality Electronic Design, ISQED, 2021, pp. 444–449.

- [17] G. Anusha, P. Deepa, Design of approximate adders and multipliers for error tolerant image processing, *Microprocess. Microsyst.* 72 (2020) 102940.
- [18] H. Jiang, C. Liu, L. Liu, F. Lombardi, J. Han, A review, Classification and comparative evaluation of approximate arithmetic circuits, *ACM J. Emerg. Technol. Comput. Syst.* 13 (4) (2017) 60.
- [19] P. Kulkarni, P. Gupta, M.D. Ercegovic, Trading accuracy for power in a multiplier architecture, *J. Low Power Electron.* 7 (4) (2011) 490–501.
- [20] K. Bhardwaj, P.S. Mane, J. Henkel, Power-and area-efficient approximate wallace tree multiplier for error-resilient systems, in: *Fifteenth International Symposium on Quality Electronic Design, IEEE*, 2014, pp. 263–269.
- [21] J. Huang, J. Lach, G. Robins, A methodology for energy-quality tradeoff using imprecise hardware, in: *Proceedings of the 49th Annual Design Automation Conference, ACM*, 2012, pp. 504–509.
- [22] Khaing Yin Kyaw, Wang Ling Goh, Kiat Seng Yeo, Low-power high-speed multiplier for error-tolerant application, in: *2010 IEEE International Conference of Electron Devices and Solid-State Circuits, EDSSC*, 2010, pp. 1–4.
- [23] S. Narayanamoorthy, H.A. Moghaddam, Z. Liu, T. Park, N.S. Kim, Energy-efficient approximate multiplication for digital signal processing and classification applications, *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* 23 (6) (2015) 1180–1184.
- [24] Y.-H. Chen, T.-Y. Chang, A high-accuracy adaptive conditional probability estimator for fixed-width booth multipliers, *IEEE Trans. Circuits Syst. I: Regular Pap.* 59 (3) (2012) 594–603.
- [25] Padmanabhan Balasubramanian, Raunaq Nayar, Douglas L. Maskell, Approximate array multipliers, *Electronics* 10 (20) (2021) 2460.
- [26] H. Jiang, J. Han, F. Lombardi, Approximate radix-8 booth multiplier for low-power and high-performance operation, *IEEE Trans. Comput.* 65 (8) (2016) 2638–2644.
- [27] Bin Wang, Xiaochun Yang, Guoren Wang, Ge Yu, Wanyu Zang, Meng Yu, Energy efficient approximate self-adaptive data collection in wireless sensor networks, *Front. Comput. Sci.* 10 (2016) 936–950.
- [28] Lavanya Maddiseti, Ranjan K. Senapati, J.V.R. Ravindra, Image multiplication with a power-efficient approximate multiplier using a 4:2 compressor, *Adv. Image Data Process. using VLSI Des. Smart Vis. Syst.* 1 (2021) 13–15.
- [29] Y. Xu, Y. Guo, S. Kimura, Approximate multiplier using reordered 4–2 compressor with OR-based error compensation, in: *2019 IEEE 13th International Conference on ASIC, ASICON*, 2019, pp. 1–4.
- [30] C. Tung, S. Huang, Low-power high-accuracy approximate multiplier using approximate high-order compressors, in: *2019 2nd International Conference on Communication Engineering and Technology, ICCT*, 2019, pp. 163–167.
- [31] E. Jagadeeswara Rao, P. Samundiswary, Error-efficient approximate multiplier design using rounding based approach for image smoothing application, *J. Electron. Test.* 37 (2021) 623–631.
- [32] Bharat Garg, Sujit Kumar Patel, Reconfigurable rounding based approximate multiplier for energy efficient multimedia applications, *Wirel. Pers. Commun.* 118 (2021) 919–931.
- [33] Seyed Amir Hossein Ejtahed, Somayeh Timarch, Efficient approximate multiplier based on a new 1-gate approximate compressor, *Circuits Systems Signal Process.* 41 (2022) 2699–2718.
- [34] Mostafa Abbasmollaei, Fahimeh Hajizadeh, Siamak Mohammadi, Mohammadreza Binesh Marvasti, Seyyed Amir Asghari, Behrouz Samieiyan, A power constrained approximate multiplier with a high level of configurability, *Microprocess. Microsyst.* 90 (2022) 104519.
- [35] Shaghayegh Vahdat, Mehdi Kamal, Ali Afzali-Kusha, Massoud Pedram, LETAM: A low energy truncation-based approximate multiplier, *Comput. Electr. Eng.* 63 (2017) 1–17.
- [36] V.G. Oklobdzija, D. Villeger, S.S. Liu, A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach, *IEEE Trans. Comput.* 45 (3) (1996) 294–306.
- [37] T. Jayasri, M. Hemalatha, Link quality estimation for adaptive data streaming in WSN, *Wirel. Pers. Commun.* 94 (2017) 1543–1562.