International Journal of Informatics and Communication Technology (IJ-ICT)
Vol. 14, No. 2, August 2025, pp. 382~392
ISSN: 2252-8776, DOI: 10.11591/ijict.v14i2.pp382-392

An improved approximate parallel prefix adder for high performance computing applications: a comparative analysis

Vamsidhar Anagani, Kasi Geethanjali, Anusha Gorantla, Annamreddy Devi
Department of Electronics and Communication Engineering, Raghu Engineering College, Visakhapatnam, India

Article Info

Article history:

Received Sep 11, 2024
Revised Nov 25, 2024
Accepted Dec 15, 2024

Keywords:

Approximate computing
AxPPA
Area-delay performance
Parallel prefix adder
Prefix operator

ABSTRACT

Binary adders are fundamental in digital circuit designs, including digital signal processors and microprocessor data path units. Consequently, significant research has focused on improving adders' power-delay efficiency. The carry tree adder (CTA) is alternatively referred to as the parallel prefix adder (PPA), is among the fastest adders, achieving superior performance in very large scale integrated (VLSI) implementations through efficient concurrent carry generation and propagation. This study introduces approximate PPAs (AxPPAs) by applying approximations in prefix operators (POs). Four types of AxPPAs-approximate kogge-stone, approximate brent-kung, approximate ladner-fischer, and approximate sparse kogge-stone-were designed and implemented on FPGA with bit widths up to 64-bit. Delay measurements from static timing analysis using Xilinx ISE design suite version 14.7 indicate that AxPPAs exhibit better latency performance than traditional PPAs. The AxPPA sparse kogge-stone, in particular, demonstrated superior area and speed performance, achieving a delay of 2.501ns for a 16-bit addition.

Corresponding Author:

Vamsidhar Anagani
Department of Electronics and Communication Engineering, Raghu Engineering College
Visakhapatnam, India
Email: vamsianagani@gmail.com

1. INTRODUCTION

Addition is a fundamental arithmetic operation, with carries rippling from one bit to the next. It can be performed rapidly, making it a crucial operation. The critical delay path of the adder determines overall speed. Half and full adders are essential for designing various adders and multipliers. Approximate computing, an emerging paradigm in integrated circuits, enhances performance without compromising acceptable quality by eliminating the need for exact computations [1]. Adder units are foundational and widely used in arithmetic hardware operations such as digital signal processing [2], image and video processing [3], computer vision, and machine learning [4]. Combining approximate adder (AxA) units with more complex modern approximate arithmetic units, such as squaring modules [5], [6], multipliers [7], [8], [9], square roots [10], and division [11], allows for interlayer approximations. Many approximate adder architectures make the logic from the least significant bit (LSB) to the most significant bit (MSB) accurate [12]-[15].

Parallel prefix adders (PPAs) are renowned for their speed and space efficiency in addition operations. These adders achieve their superior performance by implementing logarithmic reduction in the carry propagation channel, which significantly decreases the latency of the primary computational path. However, the main challenge in digital hardware design is the optimization of PPA circuit synthesis

[16]-[19]. An innovative technique combining speed and power efficiency in adder circuits has been engineered through the implementation of an approximate parallel prefix adder (AxPPA), that combines fast carry propagation and LSB-to-MSB logical approximation [20].

This paper examines four PPAs-brent-kung [21], kogge-stone [22], ladner-fischer [23], and sparse kogge-stone [24] to illustrate approximate prefix operators (AxPOs). The strategy aims to simulate carry propagation and generation for a prefix operator (PO). AxPPAs were tested on two hardware accelerators: sum of squared difference (SSD) pixel comparison and finite impulse response (FIR) filters in virtual/video processing applications [25]. Both FIR filters and SSD applications contain multiple adders, impacting area, delay, and power consumption [26], [27]. By incorporating additional strategies like Approximate Adder (AxA) combinations, one can optimize the size and energy consumption of these accelerators. Thus, our work focused on designing and implementing AxPPA-based brent-kung [21], kogge-stone [21], ladner-fischer [23], and sparse kogge-stone [24] architectures: AxPPA_brent-kung, AxPPA_kogge-stone, AxPPA_ladner-fischer, and AxPPA_sparse kogge-stone. The proposed AxPPAs aim to produce faster and more energy-efficient hardware accelerators for various applications. Effectiveness parameters for these AxPPA designs include area, measured in look-up tables (LUT), and delay, defined as the time from input application to output production. The proposed AxPPAs and additional strategies like adder combinations offer a novel approach to optimizing hardware design trade-offs.

The article's organization is outlined as follows: an extensive review of PPAs and their performance across multiple parameters is provided in section 2. In section 3 introduces an improved variant of PPA, known as AxPPA. The results of simulations, along with their corresponding analysis, are discussed in section 4. The concluding remarks of the study are presented in section 5.

2. PARALLEL PREFIX ADDERS

Substantial research is being conducted to achieve data path optimization in the design of PPAs [28]-[31]. As illustrated in Figure 1, three main stages are involved in PPA design: pre-processing, prefix computation, and post-processing [20].

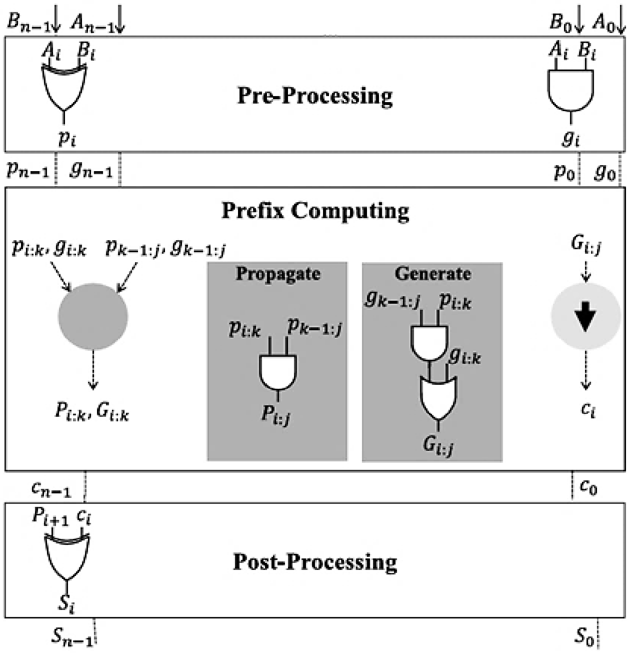


Figure 1. Stages involved in the computation of PPA

The first stage is pre-processing, which produces signals bit by bit for the subsequent stages of generating a carry and propagating a carry. The following Boolean equations illustrate how preprocessing encodes the operands' A and B input bits to generate g and propagate p [32].

$$\text{Propagate } (p_i) = A_i \oplus B_i \tag{1}$$

$$\text{Generate } (g_i) = A_i \cdot B_i \quad (2)$$

The carry-out of an adder is said to be true, when the g value is true, irrespective of the value of input carry. The signal p is true when the input carry of the ith bit order propagates to the output carry of ith bit order. Both the g and p functions are performed using logic gates, where the AND gate is used for g function and XOR gate is used for p function. These gates are evaluated simultaneously with a single-gate delay for all bits of the ith order. The size of the preprocessing circuit increases proportionally with the width of adder's input bits [19]. There are multiple approaches to implementing prefix computing, which involve adjusting various factors such as the number of links between generate and carry cells, the maximum number of outputs per gate, the overall quantity of logic gates, the depth of logic, and the area occupied by the circuit. The arrangement of the adders determines how carry and propagation are grouped in the prefix computation [33].

$$P = p_i \cdot p_{i+1} \quad (3)$$

$$G = (g_i \cdot p_{i+1}) + g_{i+1} \quad (4)$$

Where, g_i, p_{i+1} and g_{i+1} correspond to the preprocessing stage, explaining POs, the fundamental components of the prefix calculation phase respectively. The associative operator responsible for producing the carry-out and propagation (sum) bits must be contained within the PO blocks [29]. The PPA graph structure is constructed by integrating these PO blocks through prefix computations. The circuit size, energy, and delay of each PPA are dependent on the prefix computing step. Recombining the C produced by the prefix computation with the p from the pre-processing step, the final sum is formed in the post-processing step. As shown in (6), the post-processing function executes a bitwise XOR gate between the C and p signals in parallel for all bits of the ith order [34].

$$C_i = cin_i \cdot (P_i + G_i) \quad (5)$$

$$S_{i+1} = P_{i+1} \oplus C_i \quad (6)$$

2.1. Brent-kung adder

The brent-kung adder [21] exemplifies a PPA with standard architecture enabling efficient n-bit number addition in $O(\log_2 n)$ time, making it ideal for space-constrained, high-performance adders [35]. Its regular and symmetrical structure is suitable for pipeline systems, reducing production costs [25]. Prefix calculations for 8-bit groups use the brent-kung adder [21] method by first dividing 2-bit groups into 4-bit groups, continuing until the sum tree has the required bit count, with only two cells per logical level. This technique enhances traditional architecture's cost-effectiveness, crucial in very large scale integrated (VLSI) design [30]. Figure 2 shows the design process of conventional 8-bit PPAs; brent-kung adder [21], Kogge-stone adder [22] and ladner-fischer adder [23]. Figure 2(a) displays the 8-bit brent-kung adder [21] with propagate $p[1{:}8]$ and generate $g[1{:}8]$ bits. Parallel adders compute carries from LSB to MSB, establishing a critical route, with measures to ensure carry reaches MSB without delay [6].

2.2 Kogge-stone adder

The kogge-stone adder [22] is theoretically similar to the brent-kung adder [21], clustering adjoining bits based on cell size and reusing them by neighboring nodes. Consequently, the fan-out matches the cell size and has fewer levels than other architectures. For K inputs, the total cost or number of consumed cells is $K \log_2 K$. The efficiency increases and fan-out decrease in this adder [22]. Propagation across the tree and cells occurs simultaneously during generation. However, the systematic arrangement of the adder in a grid pattern results in an increase in circuit area due to scatter selections. This adder computes even digits separately while calculating the prefix for odd numbers [36]. Figure 2(b) shows the 8-bit kogge-stone adder [22] with propagate $p[1{:}8]$ and generate $g[1{:}8]$ bits.

2.3. Ladner-fischer adder

A high-performance addition operation is performed using a ladner-fischer adder [23]. To perform the addition operation, decrease the carry propagation latency that rises with ripple carry adders (RCA) [12], [31]. The data structure used to perform the calculation resembles a tree. Figure 2(c) shows the 8-bit ladner-fischer adder [23] with propagate $p[1{:}8]$ and generate $g[1{:}8]$ bits.

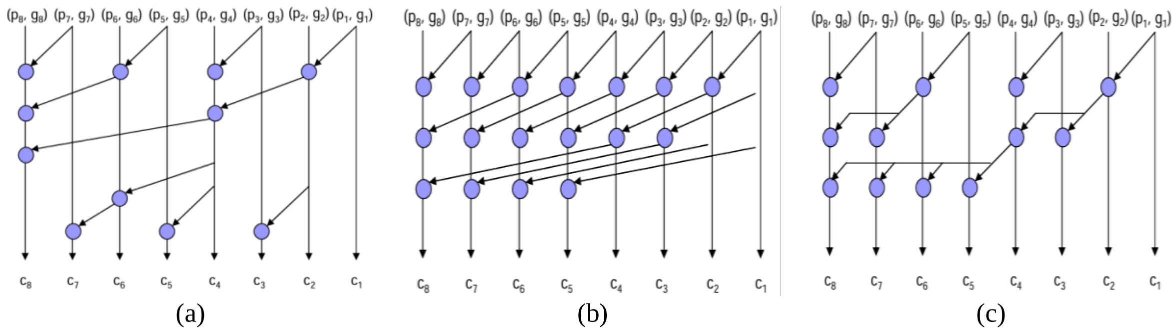


Figure 2. Design process of conventional 8-bit PPAs with propagate $p[1:8]$ and generate $g[1:8]$ bits; (a) brent-kung adder [21], (b) kogge-stone adder [22], and (c) ladner-fischer adder [23]

2.4. Sparse kogge-stone adder

One notable feature of tree-structured adders is the limiting path determined by the carry delay for an N-bit wide adder, which demonstrates an order of $\log 2N$. Multiple adder families were created using a prefix network configuration [8]. This research specifically examines the kogge-stone adder [22], distinguished by its minimal depth and limited fanout.

Figure 3 illustrates the architecture of the sparse kogge-stone adder [24]. The recurring patterns in the kogge stone prefix tree network have an impact on immune system function. The ordered pair is produced by the black cell (BC), whereas the gray cell (GC) merely provides the left signal. The connection area is well-known, although it is never as critical in an FPGA execution as it is in a VLSI one due to the high routing overhead included in every FPGA [31]. Kogge stone prefix tree networks are regular in a way that impacts defenses through repetition. This cross-sectional design streamlines the convey-prefix network by completing the summing operation with a 4-bit RCA [17]. Unique symbols are utilized in the parallel adder's schematic to distinguish between two node categories. A solid, dark-colored square represents the black-connected node, while a square featuring a central dot denotes the gray-connected node. It is fascinating to compare how this adder is implemented using RCA in FPGA as a quick carry chain, together with sparse kogge-stone and traditional kogge-stone adders [9].

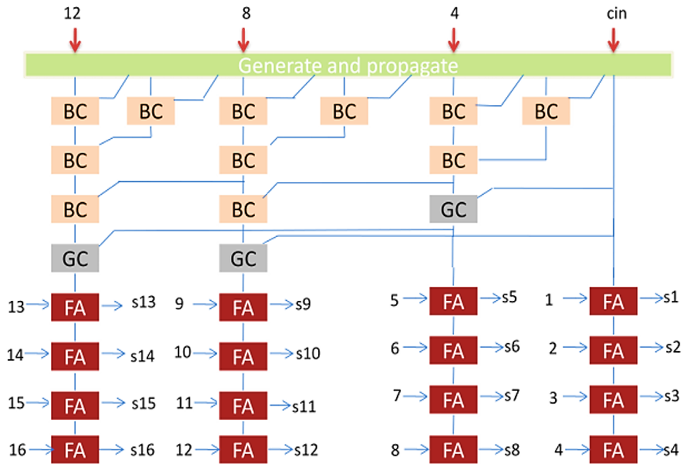


Figure 3. Sparse kogge-stone adder [24]

3. PROPOSED APPROXIMATE PARALLEL PREFIX ADDERS

In a PPA, groups of POs are utilized to calculate prefixes. Approximations were employed in the reasoning of the AxPPA concepts (see Figure 4). Adjusting the number of approximate POs during the outline timeframe allows for achieving the desired AxPPA precision level. As illustrated in Figure 4, our AxPO connects pretreatment and post-processing using only wires to develop prefix computing. In (3) and (4) describe how to compute POs [shown in Figure 4(f)], whereas (7) and (8) explore how to compute AxPOs [shown in Figure 4(h)].

$$P \approx p_{i+1} \quad (7)$$

$$G \approx g_{i+1} \quad (8)$$

In the prefix calculation stage, AxPPA removes logic gates. There are no PPA prefix computations in our technique because the PO is destroyed during the prefix computation stage. As a result, the type of PPA determines the order in which each PO appears in the computation of the PPA prefix [20].

Figure 4 shows a general 16-bit binary approximate addition for decimal numbers $33222_{(10)}$ and $116254_{(10)}$ with $K = 16$ bits. These $K = 16$ bits are split into two parts: an exact 8-bit component (see Figures 4(a)-4(c)) and an approximate 8-bit component (see Figures 4(b)-(d)). By summing $33222_{(10)}$ and $116254_{(10)}$, we found that the result is close to $217142_{(10)}$, as demonstrated in this example. For the same example as in Figure 4(b), but with $K = 8$ bits, the approximate sum is shown in Figure 4(d). Three sections are presented in Figures 4(c)-4(d): preprocessing, approximate prefix calculation, and post-processing. As shown in Figures 4(b) and 4(c), preprocessing relies on a single XOR logic gate for its critical path. It should be noted that in the approximation prefix computation, the only connections made are through wires, as shown in Figure 4(h), which links the generation and propagation of the preprocessing step to the post-processing step. One bit of carrying is generated by the approximate part in Figure 4(d) for the accurate part. The AxPPA calculation for carrying in the PPA is shown in Figure 4(c) by the dark yellow-colored POs. In Figure 4(g), the carry operator's critical path consists of two logic gates: an AND gate and an XOR gate. In this study, we constructed four different architectures based on AxPO suggestions in four different PPAs: AxPPA_brent-kung, AxPPA_ladner-fischer, AxPPA_kogge-stone, and AxPPA_sparse kogge-stone.

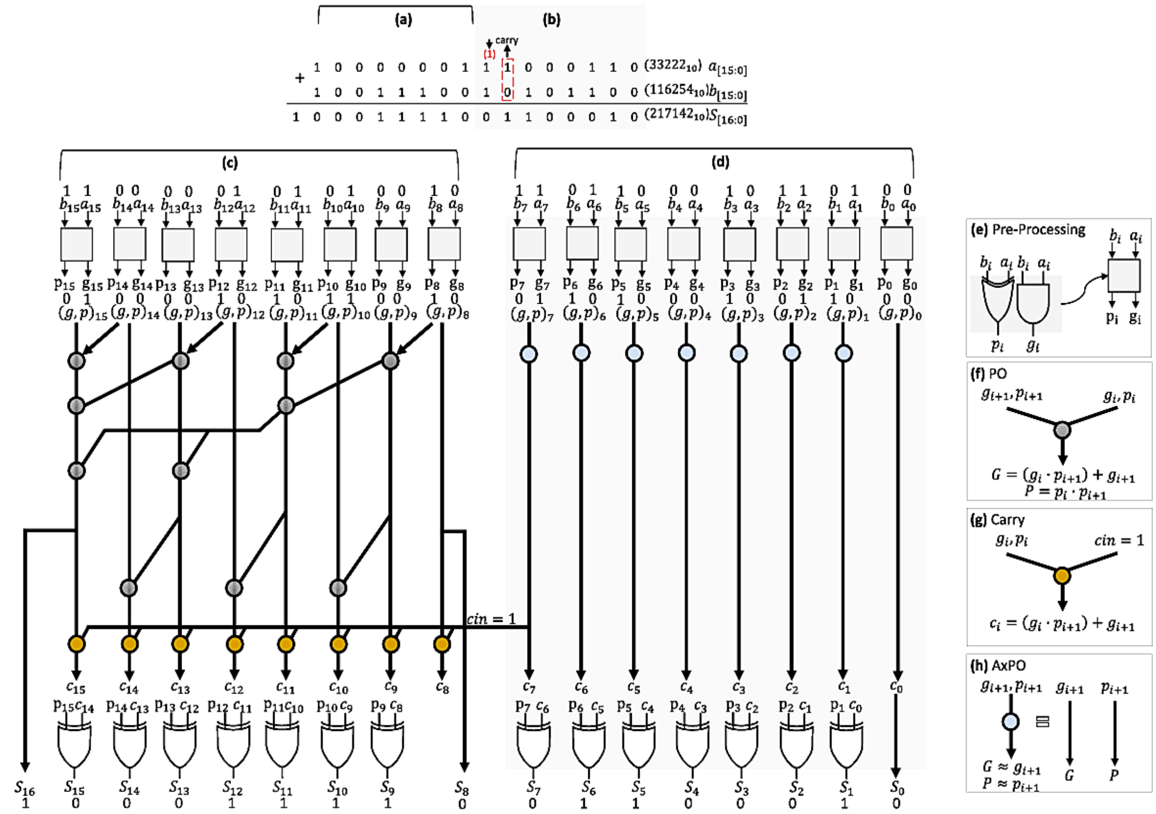


Figure 4. Example of AxPPA ladner-fischer for $K = 16$ bits; (a) PPA operation on MSB, $K = 8$ bits, (b) AxPPA operation on LSB, $K = 8$ bits, (c) accurate output generated by ladner-fischer adder on MSB, $K = 8$ bits, (d) approximate output generated by AxPPA_ladner-fischer adder on LSB, $K = 8$ bits, (e) pre-processing steps, (f) prefix operations (POs), and (g) Carry (h) AxPOs

4. RESULTS AND DISCUSSION

This segment describes the simulation of various PPAs and AxPPAs designs using Xilinx ISE design suite 14.7. The designs were implemented in verilog and synthesized through Xilinx Vivado. For all adder configurations, the inputs consist of 16-bit unsigned binary numbers (a and b) along with a carry input c_{in}. The corresponding outputs include the Sum and carry output c_{out}.

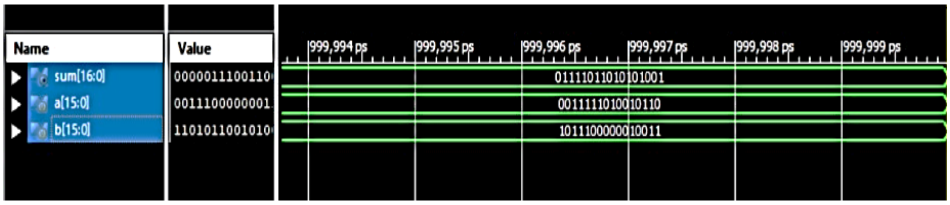
4.1. PPAs

The inputs and corresponding outputs for the PPA-based brent-kung adder [21], kogge-stone adder [22], ladner-fischer adder [23], and sparse kogge-stone adder [24] are listed in Table 1, and the simulated waveforms are shown in Figure 5. Figure 5(a) depicts inputs and ouputs of the brent-kung PPA [21] for 16-bit unsigned magnitude of inputs a and b generating 16-bit unsigned magnitude of output, sum. Similarly, Figures 5(b)-5(d) shows the input-output relations of kogge-stone [22], ladner-fischer [23], and sparse kogge-stone [24] PPAs respectively.

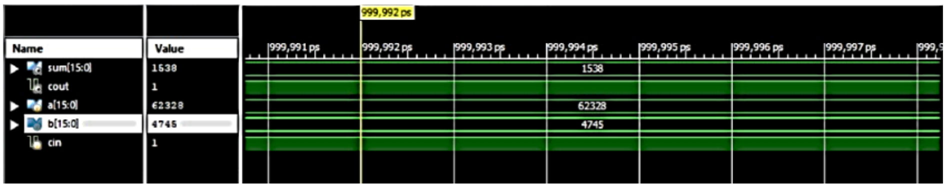
From Table 1, brent-kung [21] is applied with 16-bit unsigned magnitude of inputs a and b as $16022_{(10)}$ and $47123_{(10)}$ with $c_{in} = 0$, and the corresponding outputs sum obtained as $63145_{(10)}$ with $c_{out} = 0$. Similarly, when kogge-stone [22] is applied with a 16-bit unsigned magnitude of inputs a and b as $62328_{(10)}$ and $4745_{(10)}$ with $c_{in} = 1$, the corresponding output sum $67074_{(10)}$ with $c_{out} = 1$ is obtained.

Table 1. PPAs inputs and outputs

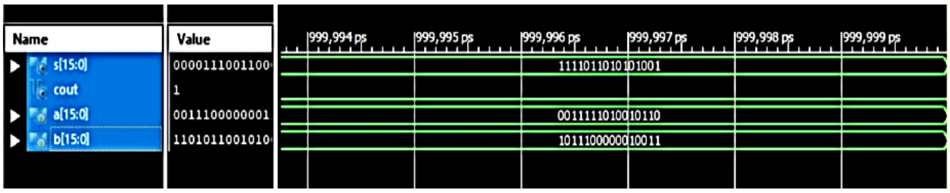
PPAs	Inputs			Outputs	
	a	b	c_{in}	Sum	c_{out}
Brent-kung [21]	$16022_{(10)}$	$47123_{(10)}$	0	$63145_{(10)}$	0
Kogge-stone [22]	$62328_{(10)}$	$4745_{(10)}$	1	$67074_{(10)}$	1
Ladner-fischer [23]	$16022_{(10)}$	$47123_{(10)}$	0	$63145_{(10)}$	1
Sparse kogge stone [24]	$10429_{(10)}$	$22573_{(10)}$	1	$33003_{(10)}$	0



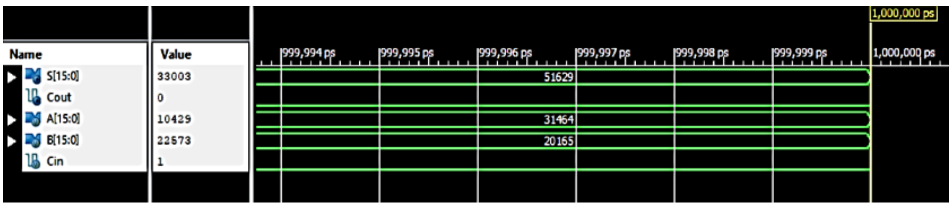
(a)



(b)



(c)



(d)

Figure 5. PPAs using 16-bit unsigned magnitude of inputs a and b generating 16-bit unsigned magnitude of output: sum (a) brent-kung [21], (b) kogge-stone [22], (c) ladner-fischer [23], and (d) sparse kogge-stone [24]

When ladner-fischer [23] is applied with a 16-bit unsigned magnitude of inputs a and b as $16022_{(10)}$ and $47123_{(10)}$ with $c_{in} = 0$, the corresponding output sum $63145_{(10)}$ with $c_{out} = 1$ is obtained. When sparse kogge stone [24] is applied with 16-bit unsigned magnitude of inputs a and b as $10429_{(10)}$ and $22573_{(10)}$ with $c_{in} = 1$, the corresponding output sum $67074_{(10)}$ with $c_{out} = 1$ are obtained. The PPAs outputs obtained are error-free and are differentiated with respect to area and propagation delay, which exhibit overall effeciency of the adder. Here, LUT are used to define the overall area of the respective adders, and a smaller delay indicates a faster addition. The PPAs performance metrics are presented in Table 2. The area occupied by brent-kung [21] is 24LUT with a delay of 4.255ns, kogge-stone [22] area is 48 LUT with a delay of 4.489ns, ladner-fischer [23] area is 24 LUT with a delay of 4.472ns, and sparse kogge-stone [24] area is 42LUT with a delay of 4.713ns.

From the obtained results, the area and delay are much less in the brent-kung adder [21], but in comparison with the three alternative adders, this one demonstrates notably inferior performance. Ladner-fischer [23] is smaller in area and more accurate than the others, while the fastest PPA is the kogge-stone [22]. Sparse kogge-stone [24] is a compromise in area compared to ladner-fischer [23] and kogge-stone [24], but reliable in terms of delay performance.

Table 2. Comparison of various PPAs

PPAs	Area	Delay (ns)
Brent-kung adder [21]	24 LUT	4.255
Kogge-stone adder [22]	48 LUT	4.489
Ladner-fischer adder [23]	24 LUT	4.472
Sparse kogge-stone adder [24]	42 LUT	4.713

4.2. Proposed AxPPAs

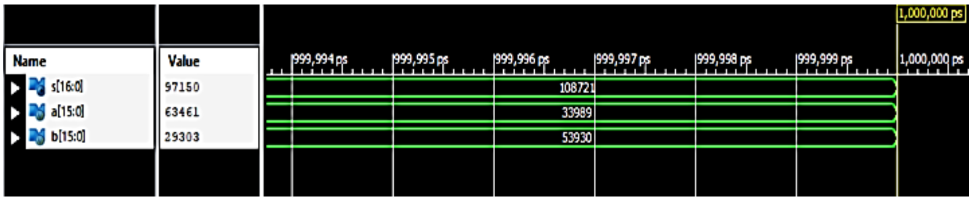
Table 3 presents the inputs and corresponding outputs for various adder types utilizing AxPPA, including brent-kung, kogge-stone, ladner-fischer, and sparse kogge-stone adders. The simulated waveforms for these adders are illustrated in Figure 6. Figure 6(a) depicts inputs and ouputs of the brent-kung AxPPA for 16-bit unsigned magnitude of inputs a and b generating 16-bit unsigned magnitude of output, sum. Similarly, Figures 6(b)-5(d) shows the input-output relations of kogge-stone, ladner-fischer, and sparse kogge-stone AxPPAs respectively.

From Table 3, AxPPA_brent-kung is applied with a 16-bit unsigned magnitude of inputs a and b as $63461_{(10)}$ and $29303_{(10)}$ respectively with $c_{in} = 0$, and the corresponding output sum obtained as $97150_{(10)}$ instead of $92764_{(10)}$ with an error count of 4. Similarly, when AxPPA_kogge-stone is applied with a 16-bit unsigned magnitude of inputs a and b as $64905_{(10)}$ and $30743_{(10)}$ with $c_{in} = 1$, the corresponding output sum is $94208_{(10)}$ instead of $95648_{(10)}$ producing an error count of 2; when AxPPA_ladner-fischer is applied with 16-bit unsigned magnitude of inputs a and b as $64905_{(10)}$ and $30743_{(10)}$ with $c_{in} = 1$, the corresponding output sum is $93521_{(10)}$ instead of $95648_{(10)}$ producing an error count of 3. When AxPPA sparse kogge-stone is applied with a 16-bit unsigned magnitude of inputs a and b as $31464_{(10)}$ and $20165_{(10)}$ respectively, with $c_{in} = 1$, the corresponding output sum is $51957_{(10)}$ instead of $51629_{(10)}$ with an error count of 1 is obtained.

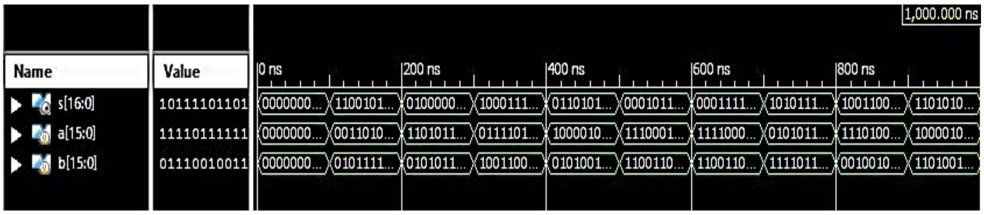
Table 4 shows the various AxPPAs and compares them with respect to area occupied, delay time, and performance. The area occupied by AxPPA_brent-kung is 23LUT with a delay of 2.220ns, AxPPA_kogge-stone area is 30 LUT with a delay of 3.097ns, AxPPA_ladner-fischer area is 22 LUT with a delay of 2.503ns, and the AxPPA_sparse kogge-stone area is 30 LUT with a delay of 2.501ns. Comparing all AxPPAs, the AxPPA_kogge-stone adder generates a high area and delay. The AxPPA_sparse kogge-stone adder achieves less delay even with an area of 30 LUT, thus exhibiting low power dissipation. Likewise, the AxPPA_kogge stone adder occupies a substantial area and exhibits high latency, yet it demonstrates superior speed in comparison to other AxPPA adders. It is also clear from Table 4 that the area and delay of the AxPPA_ladner-fisher adder are very small, and its performance is also very low when compared to the other AxPPA adders. By reducing the number of prefix stages, a new AxPPA_sparse kogge-stone adder is designed, which consumes less area with a faster addition performance.

Table 3. AxPPAs inputs and outputs

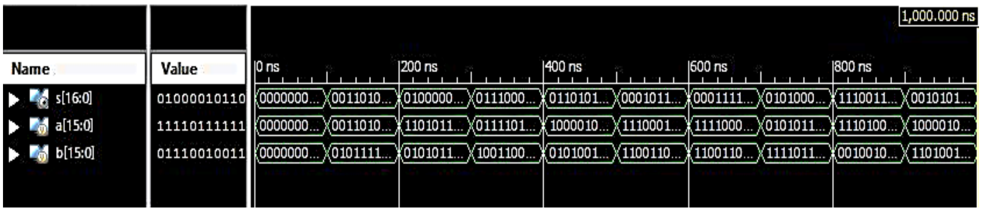
AxPPAs	Inputs			Output	Error count
	a	b	c_{in}	Sum	
AxPPA_brent-kung	$63461_{(10)}$	$29303_{(10)}$	0	$97150_{(10)}$	4
AxPPA_kogge-stone	$64905_{(10)}$	$30743_{(10)}$	1	$94208_{(10)}$	2
AxPPA_ladner-fischer	$64905_{(10)}$	$30743_{(10)}$	1	$93521_{(10)}$	3
AxPPA_sparse kogge-stone	$31464_{(10)}$	$20165_{(10)}$	0	$51957_{(10)}$	1



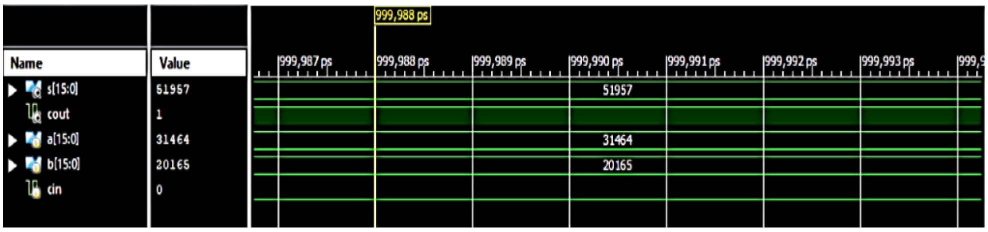
(a)



(b)



(c)



(d)

Figure 6. AxPPAs using 16-bit unsigned values of inputs a and b generating 16-bit unsigned magnitude of output, sum; (a) AxPPA brent-kung, (b) AxPPA kogge-stone, (c) AxPPA ladner-fischer, and (d) AxPPA_sparse kogge-stone

Table 4. Comparison of AxPPAs

AxPPAs	Area	Delay (ns)
AxPPA_brent-kung	23 LUT	2.220
AxPPA_kogge-stone	30 LUT	3.097
AxPPA_ladner-fischer	22 LUT	2.503
AxPPA_sparse kogge-stone	30 LUT	2.501

4.3. Discussion

Comparing PPA and AxPPAs reveals significant reductions in area and delay for AxPPAs. For instance, AxPPA_brent-kung has an area of 23LUT and a delay of 2.220ns, compared to PPA's 24LUT and 4.255ns [21]. However, AxPPAs incur errors. Similarly, AxPPA_kogge-stone shows an area of 30LUT and a delay of 3.092ns, much lower than PPA's 48LUT and 4.489ns [22]. For AxPPA_ladner-fischer versus PPA-based ladner-fischer [23], both area and delay are significantly reduced. These comparisons highlight those approximations in AxPPAs substantially decrease area and propagation delay. AxPPAs exhibit significant redundancy in delay but introduce errors due to rapid computation. Both minimal delay and error should be considered before asserting the superiority of an AxPPA. AxPPA_sparse kogge-stone outperforms other AxPPAs in delay, achieving a minimum of 2.501ns with 30 LUTs (see Table 3), and has the lowest error count of 1. This highlights the superior performance of AxPPAs, particularly AxPPA_sparse kogge-stone, compared to other variants.

5. CONCLUSION

This study proposes an improved PPA design using approximate architectures. An established approximation approach calculates the PO for the prefix contention phase. We evaluated the AxPPA concept for specific cases using bent-kung, kogge-stone, ladner-fischer, and sparse kogge-stone with application-specific evaluations. Our AxPPA technique outperformed integrated AxA regarding synthesis result savings. AxPPA meets high-quality standards and offers a higher approximation level. AxPPA_sparse kogge-stone demonstrated superior power-delay performance compared to other AxPPA designs. This is crucial for applications like high-precision arithmetic and cryptography, which often involve adding numbers on a 1,000-bit scale. The next generation of FPGA architectures must incorporate an improved carry path to enable tree-based adder implementations. This enhancement is vital for optimizing cycle time and reducing power consumption in applications such as digital signal processing and cryptography. Therefore, AxPPA are optimal for many time-sensitive applications.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Vamsidhar Anagani	✓	✓		✓	✓	✓			✓	✓	✓	✓		
Kasi Geethanjali	✓	✓	✓	✓		✓	✓		✓		✓			
Anusha Gorantla					✓	✓	✓			✓		✓		
Annamreddy Devi		✓	✓		✓			✓						

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.

REFERENCES

[1] A. Ahilan, A. A.Raj, A. Gorantla, R. Jothin, M. Shunmugathammal, and G. A. Safdar, "Design of energy-efficient approximate arithmetic circuits for error tolerant medical image processing applications," in *Lecture Notes in Electrical Engineering*, vol. 1116, 2024, pp. 679–692.

[2] P. T. L. Pereira *et al.*, "Energy-quality scalable design space exploration of approximate FFT hardware architectures," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 11, pp. 4524–4534, Nov. 2022, doi: 10.1109/TCSI.2022.3191180.

[3] G. Paim, H. Amrouch, E. A. C. da Costa, S. Bampi, and J. Henkel, "Bridging the gap between voltage over-scaling and joint hardware accelerator-algorithm closed-loop," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 1, pp. 398–410, Jan. 2022, doi: 10.1109/TCSVT.2021.3059229.

[4] Z. G. Tasoulas, G. Zervakis, I. Anagnostopoulos, H. Amrouch, and J. Henkel, "Weight-oriented approximation for energy-efficient neural network inference accelerators," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 12, pp. 4670–4683, Dec. 2020, doi: 10.1109/TCSI.2020.3019460.

[5] K. M. Reddy, M. H. Vasantha, Y. B. N. Kumar, and D. Dwivedi, "Design of approximate booth squarer for error-tolerant computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 5, pp. 1230–1241, May 2020, doi: 10.1109/TVLSI.2020.2976131.

[6] M. M. A. Da Rosa *et al.*,"AxRSU: approximate radix-4 squarer unit," in *Proceedings - IEEE International Symposium on Circuits and Systems*, May 2022, vol. 2022-May, pp. 1655–1659, doi: 10.1109/ISCAS48785.2022.9937770.

[7] W. Liu, J. Xu, D. Wang, C. Wang, P. Montuschi, and F. Lombardi, "Design and evaluation of approximate logarithmic multipliers for low power error-tolerant applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 9, pp. 2856–2868, Sep. 2018, doi: 10.1109/TCSI.2018.2792902.

[8] D. Esposito, A. G. M. Strollo, E. Napoli, D. De Caro, and N. Petra, "Approximate multipliers based on new approximate compressors," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 12, pp. 4169–4182, Dec. 2018, doi: 10.1109/TCSI.2018.2839266.

[9] A. G. M. Strollo, E. Napoli, D. De Caro, N. Petra, and G. DI Meo, "Comparison and extension of approximate 4-2 compressors for low-power approximate multipliers," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 9, pp. 3021–3034, Sep. 2020, doi: 10.1109/TCSI.2020.2988353.

[10] N. Arya, M. Pattanaik, and G. K. Sharma, "Energy-efficient logarithmic square rooter for error-resilient applications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 11, pp. 1994–1997, Nov. 2021, doi: 10.1109/TVLSI.2021.3114616.

[11] G. Paim, P. Marques, E. Costa, S. Almeida, and S. Bampi, "Improved goldschmidt algorithm for fast and energy-efficient fixed-point divider," in *ICECS 2017 - 24th IEEE International Conference on Electronics, Circuits and Systems*, Dec. 2017, vol. 2018-January, pp. 482–485, doi: 10.1109/ICECS.2017.8292070.

[12] C. H. P. Kumar and K. Sivani, "Implementation of efficient parallel prefix adders for residue number system," *International Journal of Computing and Digital Systems*, vol. 4, no. 4, pp. 295–300, Oct. 2015, doi: 10.12785/IJCDS/040409.

[13] J. Lee, H. Seo, H. Seok, and Y. Kim, "A novel approximate adder design using error reduced carry prediction and constant truncation," *IEEE Access*, vol. 9, pp. 119939–119953, 2021, doi: 10.1109/ACCESS.2021.3108443.

[14] K. L. Tsai, Y. J. Chang, C. H. Wang, and C. T. Chiang, "Accuracy-configurable radix-4 adder with a dynamic output modification scheme," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 8, pp. 3328–3336, Aug. 2021, doi: 10.1109/TCSI.2021.3085572.

[15] N. Zhu, W. L. Goh, G. Wang, and K. S. Yeo, "Enhanced low-power high-speed adder for error-tolerant application," in *2010 International SoC Design Conference, ISOCC*, Nov. 2010, pp. 323–327, doi: 10.1109/SOCDC.2010.5682905.

[16] V. Pudi, K. Sridharan, and F. Lombardi, "Majority logic formulations for parallel adder designs at reduced delay and circuit complexity," *IEEE Transactions on Computers*, vol. 66, no. 10, pp. 1824–1830, Oct. 2017, doi: 10.1109/TC.2017.2696524.

[17] Y. Ma, S. Roy, J. Miao, J. Chen, and B. Yu, "Cross-layer optimization for high speed adders: a pareto driven machine learning approach," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 12, pp. 2298–2311, Dec. 2019, doi: 10.1109/TCAD.2018.2878129.

[18] T. D. Ene and J. E. Stine, "A comprehensive exploration of the parallel prefix adder tree space," in *Proceedings - IEEE International Conference on Computer Design: VLSI in Computers and Processors*, Oct. 2021, vol. 2021-October, pp. 125–129, doi: 10.1109/ICCD53106.2021.00030.

[19] R. Roy *et al.*, "PrefixRL: optimization of parallel prefix circuits using deep reinforcement learning," in *Proceedings - Design Automation Conference*, Dec. 2021, vol. 2021-December, pp. 853–858, doi: 10.1109/DAC18074.2021.9586094.

[20] M. M. E. A. Da Rosa, G. Paim, P. U. L. Da Costa, E. A. C. Da Costa, R. I. Soares, and S. Bampi, "AxPPA: approximate parallel prefix adders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 1, pp. 17–28, Jan. 2023, doi: 10.1109/TVLSI.2022.3218021.

[21] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," *IEEE Transactions on Computers*, vol. C–31, no. 3, pp. 260–264, Mar. 1982, doi: 10.1109/TC.1982.1675982.

[22] H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Transactions on Computers*, vol. C–22, no. 8, pp. 786–793, Aug. 1973, doi: 10.1109/TC.1973.5009159.

[23] R. E. Ladner and M. J. Fischer, "Parallel prefix computation," *Journal of the ACM (JACM)*, vol. 27, no. 4, pp. 831–838, 1980, doi: 10.1145/322217.322232.

[24] S. Ghosh, P. Ndai, and K. Roy, "A novel low overhead fault tolerant kogge-stone adder using adaptive clocking," in *Proceedings of the conference on Design, automation and test in Europe*, Mar. 2008, pp. 366–371, doi: 10.1145/1403375.1403462.

[25] B. Silveira *et al.*, "Power-efficient sum of absolute differences hardware architecture using adder compressors for integer motion estimation design," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 12, pp. 3126–3137, Dec. 2017, doi: 10.1109/TCSI.2017.2728802.

[26] H. Jiang, L. Liu, P. P. Jonker, D. G. Elliott, F. Lombardi, and J. Han, "A high-performance and energy-efficient FIR adaptive filter using approximate distributed arithmetic circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 1, pp. 313–326, Jan. 2019, doi: 10.1109/TCSI.2018.2856513.

[27] P. Muthukumar, P. S. L. Kanthan, T. B. Immanuel, and K. Eswaramoorthy, "FPGA performance optimization plan for high power conversion," in *Communications in Computer and Information Science*, vol. 837, 2018, pp. 491–502.

[28] D. Esposito, D. De Caro, and A. G. M. Strollo, "Variable latency speculative parallel prefix adders for unsigned and signed operands," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 8, pp. 1200–1209, Aug. 2016, doi: 10.1109/TCSI.2016.2564699.

[29] S. Roy, M. Choudhury, R. Puri, and D. Z. Pan, "Towards optimal performance-area trade-off in adders by synthesis of parallel prefix structures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 10, pp. 1517–1530, 2014, doi: 10.1109/TCAD.2014.2341926.

[30] S. Daphni and K. S. V. Grace, "A review analysis of parallel prefix adders for better performnce in VLSI applications," in *IEEE International Conference on Circuits and Systems, ICCS 2017*, Dec. 2017, vol. 2018-January, pp. 103–106, doi: 10.1109/ICCS1.2017.8325971.

[31] K. Vitoroulis and A. J. Al-Khalili, "Performance of parallel prefix adders implemented with FPGA technology," in *2007 IEEE North-East Workshop on Circuits and Systems, NEWCAS 2007*, Aug. 2007, pp. 498–501, doi: 10.1109/NEWCAS.2007.4487969.

[32] D. H. K. Hoe, C. Martinez, and S. J. Vundavalli, "Design and characterization of parallel prefix adders using FPGAs," in *Proceedings of the Annual Southeastern Symposium on System Theory*, Mar. 2011, pp. 168–172, doi: 10.1109/SSST.2011.5753800.

[33] N. E. H. Weste and D. M. Harris, "CMOS VLSI design: a circuits and systems perspective," *Journal of Chemical Information and Modeling*, vol. 53, no. 9, pp. 1689–1699, 2013.

[34] T. Gupta, G. Verma, and S. Akhter, "FPGA implementation and performance analysis of parallel prefix structures for modular adders design," *Circuits, Systems, and Signal Processing*, vol. 44, no. 2, pp. 992–1016, Feb. 2024, doi: 10.1007/s00034-024-02857-1.

[35] S. H. Prakash and V. Balamurugan, "Design and implementation of fast radix-2 and radix-4 ACSU with different adders in viterbi decoder," in *Proceedings of 2016 Online International Conference on Green Engineering and Technologies, IC-GET 2016*, Nov. 2017, pp. 1–4, doi: 10.1109/GET.2016.7916801.

[36] A. Raju, R. Patnaik, R. K. Babu, and P. Mahato, "Parallel prefix adders-A comparative study for fastest response," in *Proceedings of the International Conference on Communication and Electronics Systems, ICCES 2016*, Oct. 2016, pp. 1–6, doi: 10.1109/CESYS.2016.7889974.

BIOGRAPHIES OF AUTHORS

Vamsidhar Anagani received his bachelor of Technology in Electronics and Communication Engineering (ECE) from Jawaharlal Nehru Technological University, Kakinada, India in 2000 and Master's in Digital Systems and Communications from National Institute of Technology, Calicut, India in 2003. He received his Ph.D. in Electronics and Communication Engineering from Andhra University, Visakhapatnam, India in 2018. He is currently a Professor of Electronics and Communication Engineering at Raghu Engineering College, Visakhapatnam, India. He has more than 20 years of teaching and research experience in various academic institutions. His areas of research include wireless communications, signal, image, and speech processing, VLSI, and machine learning. He can be contacted at email: vamsianagani@gmail.com.

Kasi Geethanjali received her Bachelor of Technology and Master's degrees from Jawaharlal Nehru Technological University, Kakinada, India. She is Pursuing Ph. D (Part-time). from KL University, Vijayawada, India. She is currently working as an Assistant Professor of Electronics and Communication Engineering at Raghu Engineering College, Visakhapatnam, India. Her current research interests include digital electronics and VLSI design. She can be contacted at email: geethanjalikasi01@gmail.com.

Anusha Gorantla received her Bachelor of Technology in Electronics and Communication Engineering (ECE) from JNTU, Hyderabad in 2006. She received Master's VLSI Design in 2008 and a Ph.D. in Information and Communication Engineering in 2018 from Anna University, Chennai, India. Now, she works as an Associate Professor, at the Department of Electronics and Communication Engineering, Raghu Engineering College, Visakhapatnam, Andhra Pradesh, India. Her research includes low-power VLSI design, approximate computing, and image processing. She can be contacted at email: anushagorantla3@gmail.com.

Annamreddy Devi completed her B.Tech. in the stream of Electronics and Communication Engineering (ECE) from Raghu Engineering College, Visakhapatnam, India in 2022. She had done various projects and mostly interested in digital electronics and VLSI design. She can be contacted at email: 19981A0407@raghuenggcollege.com.