Journal of Electronics and Informatics (ISSN: 2582-3825)
www.irojournals.com/iroei/

Performance Analysis of High-Speed Parallel Prefix Adders for Arithmetic Applications

Banumathi J.[1], Karthy G.[2]

[1]Research Scholar, [2]Associate Professor, ECE, School of Engineering and Technology, Dhanalakshmi Srinivasan University, Trichy, India.

E-mail: [1]jbanu84@gmail.com, [2]karthyg@gmail.com

Abstract

Research on affordable adder architectures has been spurred by the need for high-performance arithmetic in digital and embedded signal processors. Carry propagation affects speed, area, and power consumption by limiting two-operand addition, a fundamental operation in these systems. Because of their versatility in modifying particular performance metrics, Parallel Prefix Adders (PPAs) represent a viable substitute. The performance of various PPAs, including Kogge-Stone, Sparse Kogge-Stone, Spanning Tree, Brent-Kung, Han-Carlson, and Ladner-Fischer, on a Virtex-7 FPGA with Xilinx Vivado 2022.2 is investigated in this paper. The findings imply that each PPA design performs well in its own way and that no single design can offer the best balance of power, latency, and area. By taking performance parameter trade-offs into account, the research helps designers choose the best PPA for their design needs.

Keywords: Digital Signal Processors (DSP). Parallel Prefix Adders (PPAs), Xilinx Vivado. Virtex FPGA.

1. Introduction

Although decimal numbers are convenient for use in calculations, digital hardware such as digital signal processors (DSPs) and microprocessors utilize the binary numbering system. This is extremely significant in that it is an economical representation of a large range of values. The binary adders constitute critical components of such systems, and numerous different types of adders exist, each with certain performance advantages. The adder is chosen based on the

DOI: https://doi.org/10.36548/jei.2025.3.001
Received: 21.05.2025, received in revised form: 25.06.2025, accepted: 15.07.2025, published: 29.07.2025

specific application, and since it is a fundamental building block of digital systems, its performance has a significant contribution to the overall design.

Binary adders are elementary building blocks of the majority of digital circuits, including memory address units, dividers, and multipliers, according to [8]. Optimizing binary addition has the capability to improve overall system speed considerably. Thus, designers tend to utilize fast adders when optimizing designs since they normally assign the important path in calculations. However, the carry chain in binary addition is the principal bottleneck, and it becomes more difficult to manage with increasing input operands. The worst case for the carry is to propagate from the LSB to the MSB. Therefore, the speedup of the carry chain is a principal goal in high-speed adder design since it is difficult to eliminate it. Several topologies of adders have been studied over the past decade with the objective of improving performance.

For small bit sizes, traditional adders such as ripple carry adders and carry look ahead adders (CLAs) are usually adequate. In instances where bit sizes increase, however, these adders suffer significantly from delays due to longer carry chains. Parallel prefix adders are suitable for high-bit applications since they possess incredible speed with minimal space usage. The key benefit of parallel prefix adders is the parallel generation of the carry, which reduces logical steps and enhances speed. This research compares and contrasts several Parallel Prefix Adders (PPAs) in terms of space, power consumption, and latency for arithmetic applications. The PPAs are implemented on 7-series FPGAs, allowing for their comprehensive evaluation. The goal of the work is to find an optimal design for some arithmetic PPA through trade-offs among area, power, and latency.

2. Literature Survey

Parallel prefix adder (PPA) studies have increasingly evolved as a function of their central role in optimizing digital arithmetic operations, particularly within FPGA and ASIC domains. PPA performance implemented within an FPGA setting was investigated in [1], demonstrating enhanced speed and area efficiency compared to conventional adders. Based on this, Brzozowski [2] provided an inclusive comparison of dynamic power consumption in various PPA implementations in terms of the balance between performance and energy efficiency. Recent works like [3] and [4] compared the performance metrics of basic adder structures against parallel prefix architectures, showcasing improved performance metrics of

PPAs in FPGA and ASIC implementations. Thakur et al. [5] contributed to design enhancements by combining high-speed architectures with a goal of optimizing digital circuits.

Background material in digital circuits from texts such as Kime and Mano [6], Rabaey et al. [7], and Weste and Harris [8] underpins the majority of PPA innovation. Similarly, CMOS-focused analysis by Kang and Leblebici [9] is key background in understanding low-level circuit behavior in adder design. Earlier algorithmic studies, such as that by Wan and Wey [10] set the stage for binary arithmetic improvements, while Beaumont-Smith and Lim [11] proposed practical PPA layouts. Historical hardware implementations, as shown in Vitoroulis and Al-Khalili's [12] FPGA evaluation and Hoe et al.'s [13] detailed synthesis reports, offered empirical validation for theoretical models. Targeted deployments in speed- and power-critical applications have also been considered. Santhi and Deepika [14] indicated PPA models for targeted critical systems, and cost analysis studies like those by Marouf et al. [15] included variable architectures for cost-effective deployment. FPGA synthesis and verification continue to be pivotal, as evidenced in [16], with metrics such as delay, power, and area actively targeted in works such as [17] and [18]. Furthermore, module-level optimizations in Verilog-based designs in [19] focused on specific design-level improvements.

Comparative and application-oriented perspectives were presented by Rahila and Kumar [20], and more recent evaluations by Viraktimath et al. [21] included architecture-level trade-offs. High-end design innovations, such as quantum-dot cellular automata (QCA) integration presented by Touil et al. [22] characterize next-generation logic implementation. Lastly, the work by Athur et al. showed a novel high-speed design with promising simulations and served to further augmenting the design domain of high-performance PPAs.

3. Parallel Prefix Adders

PPAs are speedier and more effective than CLA in adding two operands in parallel [7,23]. This study analyzes six alternative PPA variants, each with a distinct tree structure design tailored to specific performance characteristics. PPAs improve efficiency for applications with high speeds by reducing logic complexity and latency, making them essential components in high-speed arithmetic circuits. By creating carries concurrently, PPAs avoid the limitations of traditional adders' carry propagation stages. Although this comes at the expense of additional power and space, the benefits are substantial. The primary benefit of PPAs is that they reduce the number of logic levels via parallel carry calculation. PPAs have a delay

complexity of $O(\log_2 N)$; hence, they are more reliable and quicker than other adders [13]. However, PPAs have limitations in terms of complexity, space overhead, wiring complexity, and power consumption. They can be difficult to scale to large bit widths and require extensive design space exploration. While PPAs have outstanding speeds, their complexity and space needs can be enormous. Optimizing PPA designs for specific applications and exploring alternative architectures may alleviate these restrictions.

3.1 Structure of PPAs

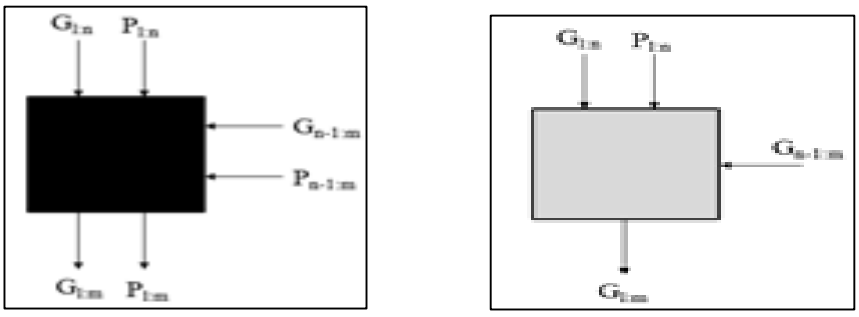
The process of binary addition using parallel prefix adders has three stages: preprocessing, carry generation, and postprocessing.

- **Pre-processing Stage:** This step calculates and produces bits for each pairing of input operands, such as ai and bi. The following equations determine the Generate (Gi) and Propagate (Pi) signals based on their prefix stage.

$$G_i = a_i \,\&\, b_i \qquad (1)$$

$$P^i = a_i \wedge b_i \qquad (2)$$

- **Carry-generation stage:** This stage necessitates the creation of carry signals for each bit position via carry-lookahead logic with black and gray cells, as illustrated in Fig. 1a and 1b. Using a series of concurrent algorithms, compute the carry signals from the generate and propagate signals. This is done in a tree-like fashion, merging pairs of bits in consecutive steps.



(a) (b)

Figure 1. (a) Black Cell (b) Grey Cell

$$G_{l:m} = (P_{l:n} \text{ and } G_{n-1:m}) \text{ Or } G_{l:n} \qquad (3)$$

$$P_{l:m} = P_{l:n} \text{ and } P_{n-1:m} \quad (4)$$

- **Post-Processing Stage:** This stage comprises the final calculation of the total. This stage applies to all types of adders. The sum is expressed as.

$$S_i = P_i \text{ and } C_{i+1} \quad (5)$$

- **Parallel Prefix Adders (PPAs):** Kogge-Stone Adder (KSA), Sparse Kogge-Stone Adder (SKSA), Spanning Tree Adder (STA), Brent-Kung Adder (BKA), Han-Carlson Adder (HCA), and Ladner-Fischer Adder (LFA) are the six types of PPAs that are the subject of this study. Each of these adders has a distinct prefix graph structure that is suited to particular performance metrics like area, latency, and power.

3.2 Kogge Stone Adder (KSA)

Similar to the carry-lookahead adder, the KSA is a straightforward but effective adder design that was created in 1973 by Peter M. Kogge and Harold S. Stone. As seen in Fig. 2, the KSA's tree structure has a significant number of processing units, which results in higher area consumption. Nonetheless, it is suitable for high-bit applications (32, 64, and 128 bits) due to its regular architecture and limited fanout of two, which significantly reduces the critical path [15]. The KSA was developed using the Xilinx Vivado 2022.2 tool, which comes with Virtex 7, and one of its main drawbacks is its high-power dissipation, which is caused by its robust parallel execution.

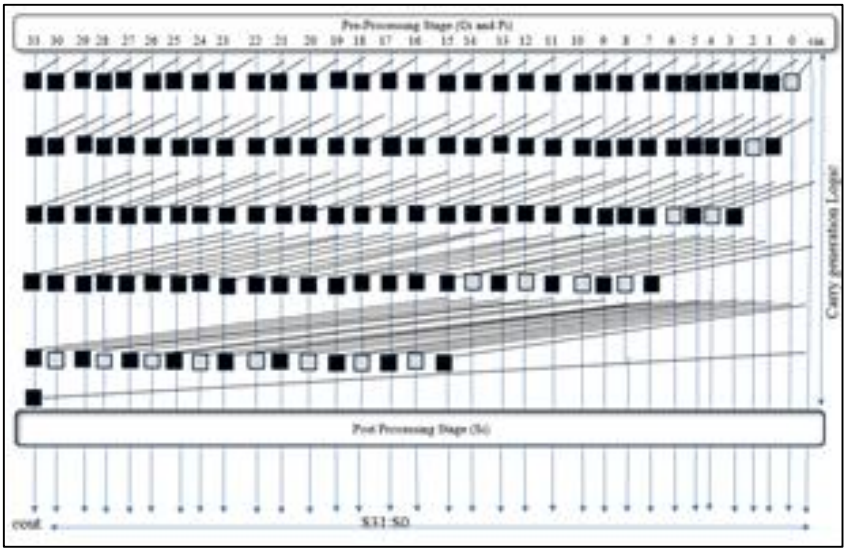


Figure 2. Structure of 32-bit KSA

3.3 Sparse Kogge-Stone Adder (SKSA)

A version of the KSA, the SKSA aims to lower the original architecture's power and space requirements. By integrating sparsity into the prefix graph, the SKSA conserves space by decreasing the number of processing blocks. Because of this innovation, the SKSA uses less power and is a more energy-efficient option. Even with these enhancements, the SKSA's low fanout and extremely regular structure allow for high-speed operation in high-bit applications. Applications where energy efficiency is highly valued can benefit from the SKSA's superior area, power consumption, and speed compared to the KSA. The SKSA's organizational structure is shown in Figure 3.

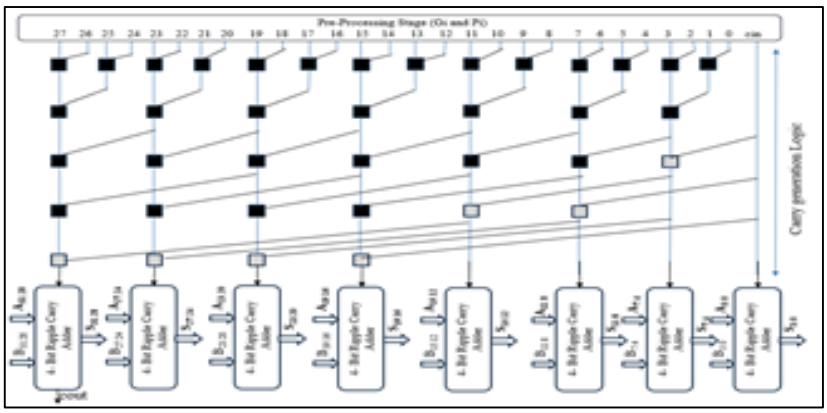


Figure 3. Structure of 32-bit SKSA

3.4 Spanning Tree Adder (STA)

The Spanning Tree Adder, or STA, is a parallel prefix adder that generates carries based on a spanning tree architecture. The STA aims to balance area, power, and speed through optimization.

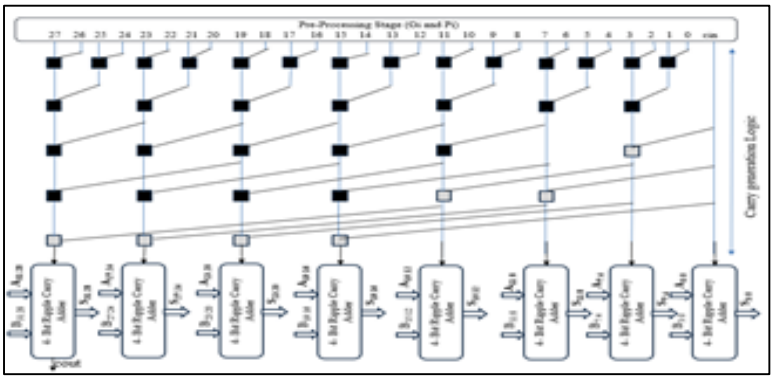


Figure 4. Structure of 32-bit STA

It generates carries in a tree architecture, reducing the number of processing blocks needed. The STA is also space-efficient and thus suitable for applications with minimal space. The STA combines high-speed performance with low power, making it perfect for high-bit applications requiring space efficiency and power preservation. The structure of the STA is illustrated in Fig. 4.

3.5 Brent-Kung Adder (BKA)

The LFA is a high-speed adder architecture that has lower latency because of its short logic depth. It was developed in 1980 by Richard Ladner and Michael Fischer. Nevertheless, as mentioned in [16], this quick performance is offset by a significant fan-out requirement of roughly n/2.

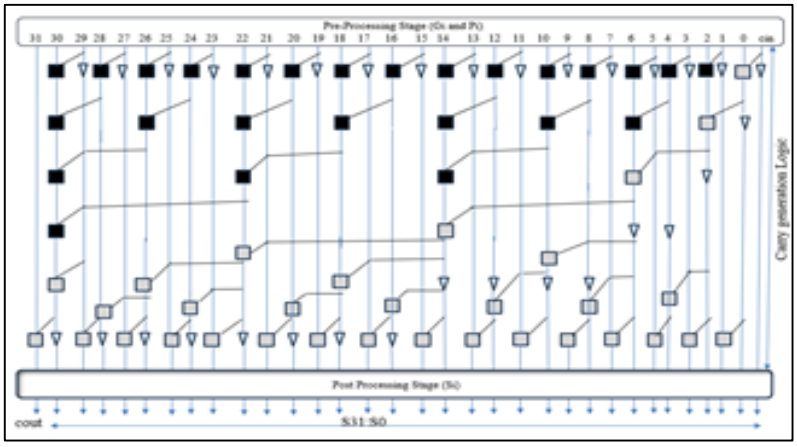


Figure 5. Structure of 32-bit BKA

3.6 Han Carlson Adder (HCA)

The BKA and KSA families in the network are connected by the HCA. In 1987, Han and Carlson made the proposal. This adder propagates/generates odd bits and carries/combines even ones, just like KSA. Lastly, actual carry bits are created by recombining odd and even bits. The first three of its five stages are comparable to those observed in Saudi Arabia. It is shorter and uses fewer cells than KSA. Although this reduces complexity, the carry-merge path must take an additional step. Just remove the final KSA row to construct Han Carlson's conjectural prefix processing step. Because HCA uses less space and power than RCA, it is the preferred adder. Compared to RCA, it saves more time. The HCA provides a suitable answer [3]. Fig. 6 shows the HCA structure.

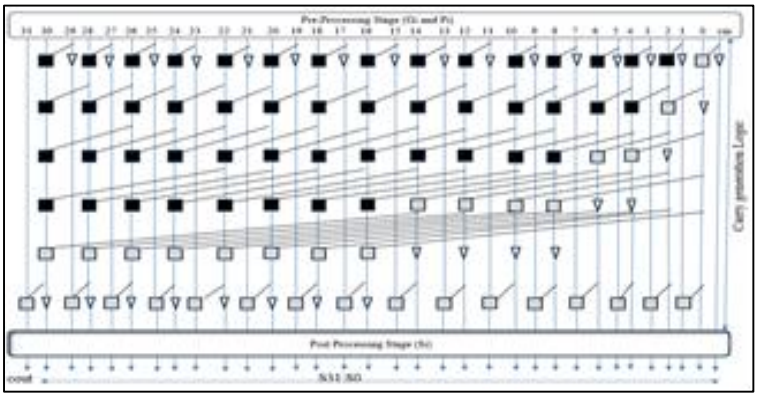


Figure 6. Structure of 32-bit HCA

3.7 Ladner-Fischer Adder (LFA)

Richard Ladner and Michael Fischer created the LFA in 1980, which is a high-speed adder architecture with reduced latency due to its short logic depth. However, this fast performance is offset by a substantial fan-out requirement of about n/2, as stated in [16].

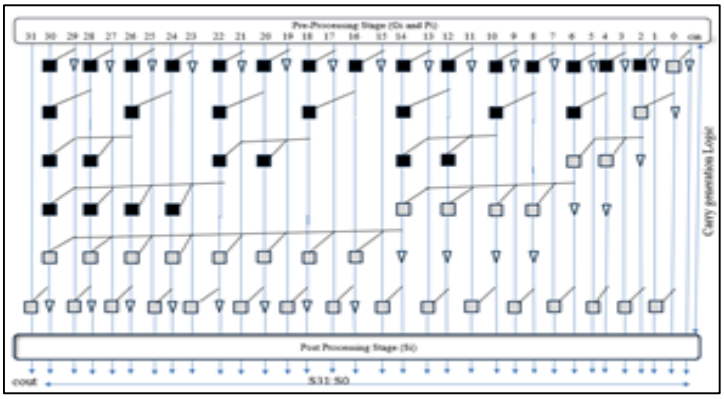


Figure 7. Structure of 32-bit LFA

Because it balances the ideal area of BKA with the ideal delay of KSA, LFA is the best adder for both area and delay [4]. LFA is a flexible design option because of its structure, which allows for a variable trade-off between area and latency (Fig. 7).

4. Results and Comparison

An FPGA-based method for evaluating and contrasting the performance of numerous PPAs using crucial metrics like latency, power consumption, and device usage is presented in this paper. The Xilinx Vivado 2022.2 Tool, which comes with Virtex 7, is used to simulate and create the PPAs. The functional accuracy of the design is demonstrated by the timing simulation results for the BKA at 8, 16, and 32 bits in Figures 8-10.

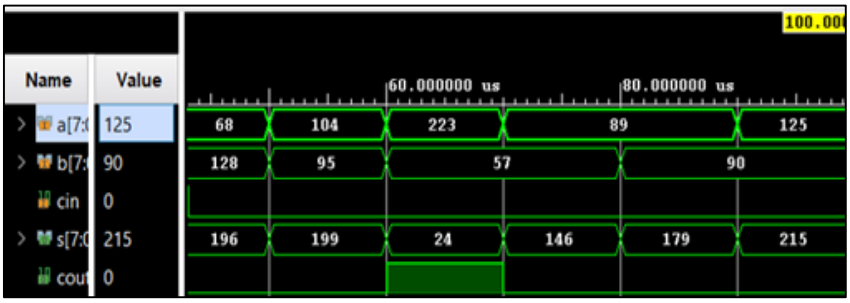


Figure 8. Simulation Output of 8-bit BKA

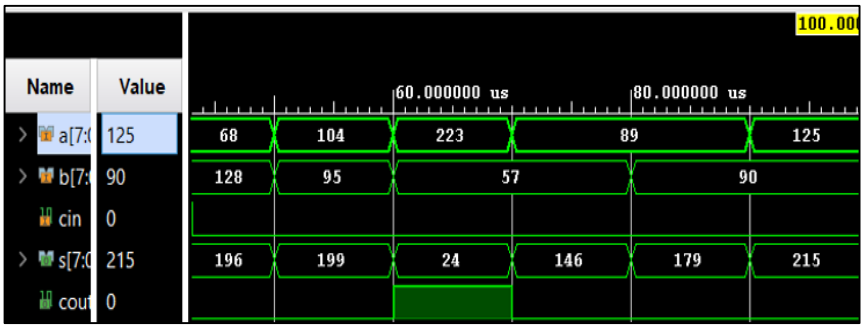


Figure 9. Simulation Output of 16-bit BKA

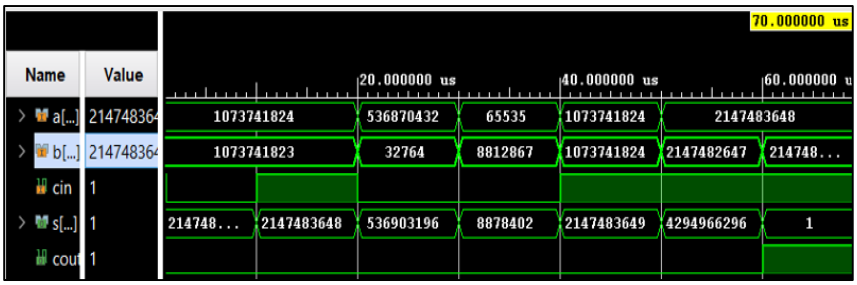
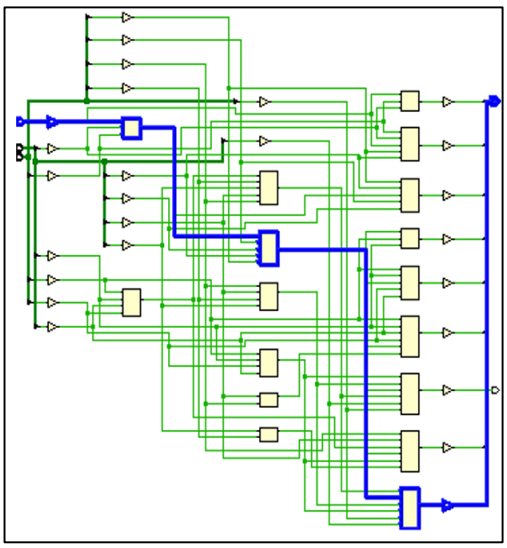
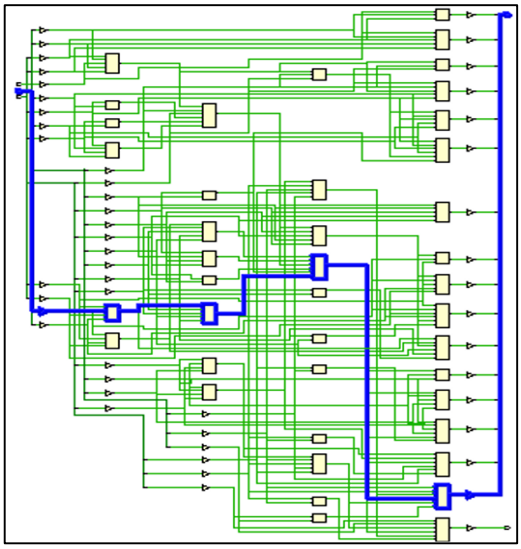


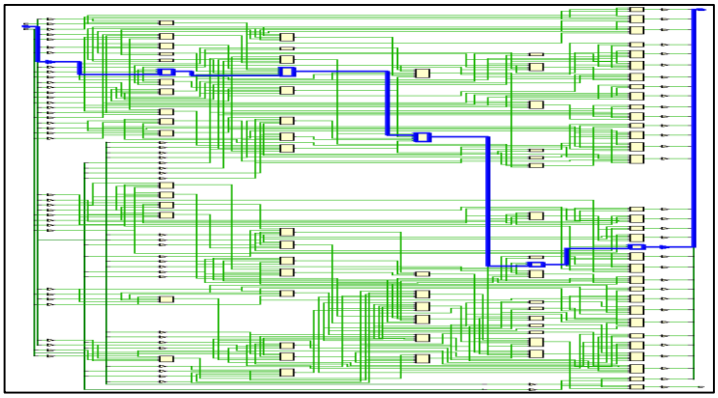
Figure 10. Simulation Output of 32-bit BKA



(a)



(b)



(C)

Figure 11. Implementation of 8-bit,16-bit, and 32-bit BKA

Figures 11a, 11b, and 11c show post-implementation schematics for the 8-bit, 16-bit, and 32-bit BKAs, demonstrating the automated placement and routing of various Verilog code-generated blocks. Similarly, other adder designs were simulated and constructed using the same process. Tables I, II, and III present the results for latency, power, and device utilization in 8-bit, 16-bit, and 32-bit architectures. Figures 12–15 show comparison charts for delay, power, LUTs, and slices.

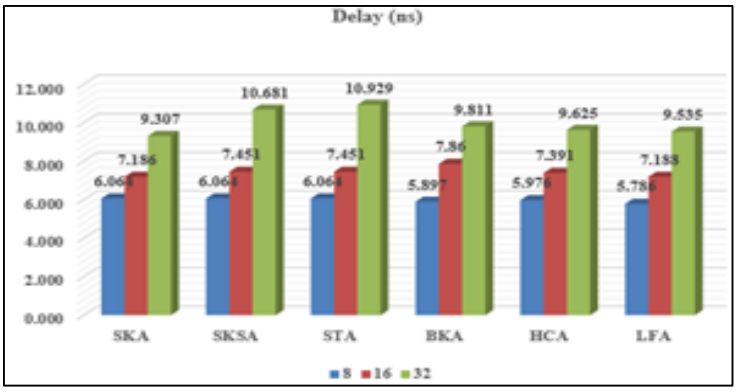


Figure 12. Delay Comparison of PPAs

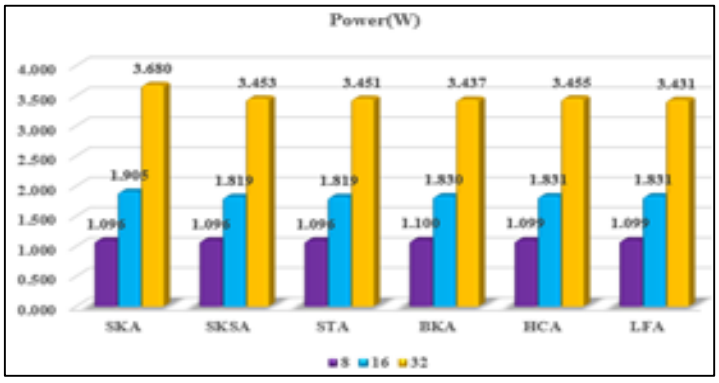


Figure 13. Power Comparison of PPAs

The important findings include: When compared to other adders, Figure 12 shows that the 8-bit LFA, 16-bit KSA, and 32-bit KSA have the lowest latency. The power consumption of all adders with different bit widths is almost the same, as shown in Figure 13.

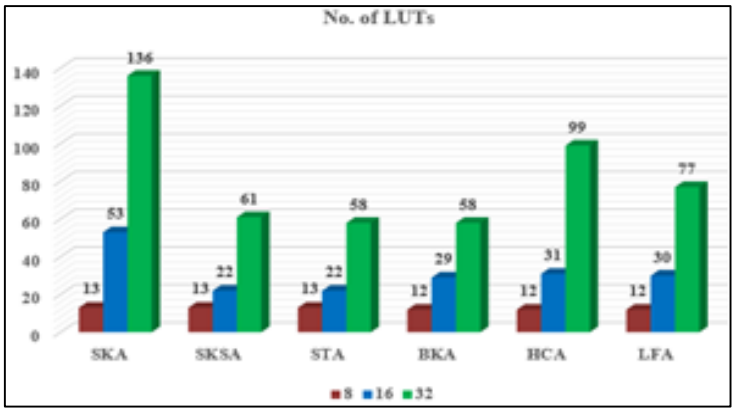


Figure 14. LUTs Comparison of PPAs

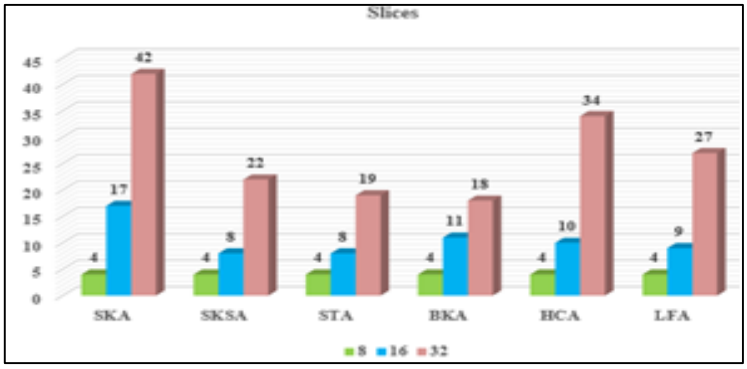


Figure 15. Slices Comparison of PPAs

Figure 14 shows that LUT consumption varies at 16-bit and 32-bit, with SKA and HCA exhibiting significant differences, while it is similar for all adders at 8-bit. At 8 bits, Figure 15 illustrates that slice consumption is constant for all adders. The SKSA needs fewer slices than the KSA, though it has 16 bits. In a similar vein, the KSA still uses the most slices in 32-bit, but the BKA uses fewer.

Table 1. Comparison of 8-bit PPAS

Name of the Adder	Delay (ns)	Power (w)	Device Utilization	
			No of LUTs	Slices
KSA	6.064	1.096	13	4
SKSA	6.064	1.096	13	4
STA	6.064	1.096	13	4
BKA	5.897	1.100	12	4

HCA	5.976	1.099	12	4
LFA	5.786	1.099	12	4

Table 2. Comparison of 16-bit PPAS

Name of the Adder	Delay (ns)	Power (w)	Device Utilization	
			No of LUTs	Slices
KSA	7.186	1.905	53	17
SKSA	7.451	1.819	22	8
STA	7.451	1.819	22	8
BKA	7.86	1.830	29	11
HCA	7.391	1.831	31	10
LFA	7.188	1.831	30	9

Table 3. Comparison of 32- bit PPAs

Name of the Adder	Delay (ns)	Power (w)	Device Utilization	
			No of LUTs	Slices
KSA	9.307	3.680	136	42
SKSA	10.681	3.453	61	22
STA	10.929	3.451	58	19
BKA	9.811	3.437	58	18
HCA	9.625	3.455	99	34
LFA	9.535	3.431	77	47

5. Conclusion and Future Work

Remarkable trade-offs between power consumption, resource utilization, and latency are revealed by the simulation of the six PPA designs on FPGA platforms in this work. Applications that demand speed and area efficiency seem to be a good fit for the LFA. BKA

has almost zero latency overhead and optimal area consumption, whereas KSA is best for high-speed operations. Importantly, all designs use the same amount of power. Based on these results, designers can select the optimal PPA architecture based on their specific needs, including those related to speed, area, and power.

Enhancing PPA designs for intended applications, investigating power-saving techniques, and evaluating PPA performance on future FPGA platforms or other hardware must be the goals on future research. This paper lays the groundwork for upcoming advancements in low-power arithmetic circuits. The work of future PPAs may involve enhancing scalability, investigating new prefix graph topologies, optimizing energy consumption, and refining designs for particular applications. To increase the effectiveness and performance of PPAs, researchers can examine hybrid PPA designs, design space exploration strategies, and emerging technologies.

References

[1] Performance analysis of parallel prefix adders developed with field programmable gate array technology. (2024). International Journal of Reconfigurable and Embedded Systems (IJRES), 14(1). https://doi.org/10.11591/ijres.v14.i1.

[2] Brzozowski, I. (2024). Comparative Analysis of dynamic power consumption of parallel Prefix Adder. ACM Transactions on Design Automation of Electronic Systems, 29(3), 1–22. https://doi.org/10.1145/3651984.

[3] S. Dubey and G. Verma, "Analysis of Basic Adder with Parallel Prefix Adder," 2020 First IEEE International Conference on Measurement, Instrumentation, Control and Automation (ICMICA), Kurukshetra, India, (2020): 1-6, doi: 10.1109/ICMICA48462.2020.9242842.

[4] S. Gauhar, A. Sharif and N. Alam, "Comparison of Parallel Prefix Adders Based on FPGA & ASIC Implementations," 2020 IEEE Students Conference on Engineering & Systems (SCES), Prayagraj, India, (2020): 1-6, doi: 10.1109/SCES50439.2020.9236737.

[5] G. Thakur, H. Sohal and S. Jain, "Design and Analysis of High-Speed Parallel Prefix Adder for Digital Circuit Design Applications," 2020 International Conference on Computational Performance Evaluation (ComPE), Shillong, India, (2020): 095-100, doi: 10.1109/ComPE49325.2020.9200064..

[6] Kime, Charles R., and M. Morris Mano. Logic and computer design fundamentals. Hoboken, NJ: Prentice Hall, 2003.

[7] Rabaey, Jan M., Anantha Chandrakasan, and Borivoje Nikolic. Digital integrated circuits. Vol. 2. Englewood Cliffs: Prentice hall, 2002.

[8] Weste, Neil HE, and David Harris. CMOS VLSI design: a circuits and systems perspective. Pearson Education India, 2015.

[9] Kang, Sung-Mo. "yusuf Leblebici: CMOS Digital Integrated Circuits." (2010): 218-230.

[10] Wan, Yi, and Chin-Long Wey. "Efficient algorithms for binary logarithmic conversion and addition." IEE Proceedings-Computers and Digital Techniques 146, no. 3 (1999): 168-172.

[11] Beaumont-Smith, Andrew, and C-C. Lim. "Parallel prefix adder design." In Proceedings 15th IEEE Symposium on Computer Arithmetic. ARITH-15 2001, IEEE, (2001): 218-225.

[12] Vitoroulis, Konstantinos, and Asim J. Al-Khalili. "Performance of parallel prefix adders implemented with FPGA technology." In 2007 IEEE Northeast Workshop on Circuits and Systems, IEEE, (2007): 498-501.

[13] Hoe, David HK, Chris Martinez, and Sri Jyothsna Vundavalli. "Design and characterization of parallel prefix adders using FPGAs." In 2011 IEEE 43rd Southeastern Symposium on System Theory, IEEE, (2011): 168-172.

[14] Santhi, G. L. A., and G. Deepika. "Realization of parallel prefix adders for power and speed critical applications." International Journal VLSI system and design communication systems (IJVDOS) 4, no. 02 (2016).

[15] Marouf, Ibrahim, Mohammed Mosab Asad, Ahmad Bakhuraibah, and Qasem Abu Al-Haija. "Cost analysis study of variable parallel prefix adders using altera cyclone IV FPGA kit." In 2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA), IEEE, (2017): 1-4.

[16] Al-Haija, Qasem Abu, Mohamad Musab Asad, Ibrahim Marouf, Ahmad Bakhuraibah, and Hesham Enshasy. "FPGA synthesis and validation of parallel prefix adders." Acta Electronica Malaysia 3, no. 2 (2019): 31-36.

[17] Yezerla, Sudheer Kumar, and B. Rajendra Naik. "Design and Estimation of delay, power and area for Parallel prefix adders." In 2014 Recent Advances in Engineering and Computational Sciences (RAECS), IEEE, (2014): 1-6.

[18] Cury, Ch, and M. Nisanth. "Design of Parallel Prefix Adders using FPGAs." IOSR Journal of VLSI and Signal Processing 4 (2014): 45-51.

[19] Nandini, M. E. D. A. P. A. T. I., and A. Jayavani. "High Speed and Power Optimized Parallel Prefix Modulo Adders using Verilog." International Journal of Advanced Technology and Innovative Research 7, no. 01 (2015): 0216-0133.

[20] Rahila, K. C., and Ulayil Sajesh Kumar. "A comprehensive comparative analysis of parallel prefix adders for asic implementation." In proceedings of the International Conference on Systems, Energy & Environment (ICSEE). 2019.

[21] Viraktimath, S. V., A. KS, P. Vaishnavi, and W. Wiranchi. "Analysis of Parallel Prefix Adders." International Journal for Research in Applied Science and Engineering Technology 11, no. 12 (2023): 1445-1450.

[22] Touil, Lamjed, Chteoui Henchir, and Abdellatif Mtibaa. "An efficient design of a parallel prefix adder based on qca technology." IETE Journal of Research (2023): 1-15.

[23] Athur, D. K., Narayanan, B., Gopalakrishnan, A., Palanisamy, S., & Augustine, A. A. (2022). Design of novel high speed parallel prefix adder. Indonesian Journal of Electrical Engineering and Computer Science, 29(3), 1345. https://doi.org/10.11591/ijeecs.v29.i3.pp1345-1354