IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 12, Issue 7, July 2025

DOI: 10.17148/IARJSET.2025.12756

Hybrid Power and Area Efficient Parallel Prefix Adder based on Approximate Computing

Ms. JASMINE PRIYADHARSHINI B, M.E[1], SAINATH. S[2], LOGESH. S P[3], CHIBI SIDDHARTH. R S[4]

Assistant Professor Senior Grade, Department of Electronics and Communication Engineering, Sri Ramakrishna Engineering College, Coimbatore, Tamilnadu, India[1]

UG Student, Department of Electronics and Communication Engineering, Sri Ramakrishna Engineering College, Coimbatore, Tamilnadu, India[2,3,4]

Abstract: The growing demand for energy-efficient and high-performance arithmetic units in digital systems has intensified research into low-power design techniques. Adders, being fundamental components in digital signal processing, multimedia, and machine learning applications, play a crucial role in overall system performance. This project presents a hybrid power and area efficient parallel prefix adder (PPA) design that utilizes approximate computing to reduce power and area consumption, while maintaining acceptable accuracy levels for error-tolerant applications. The proposed architecture divides the adder into two distinct sections: the Least Significant Part (LSP), which employs approximate logic to reduce complexity and resource usage, and the Most Significant Part (MSP), which retains exact computation to ensure result fidelity where it is most critical. A combination of Kogge-Stone and radix-4 adder structures is used to balance speed, architectural simplicity, and energy savings. The design introduces approximate prefix operators (AxPOs) that eliminate or simplify certain logic gates in the carry propagation paths, significantly optimizing power and delay. Four approximate PPA variants—AxPPA-BK (Brent-Kung), AxPPA-KS (Kogge-Stone), AxPPA-LF (Ladner-Fischer), and AxPPA-SK (Sklansky)—are developed and evaluated against the proposed hybrid adder. The implementation is carried out in Verilog HDL, simulated using ModelSim, and synthesized using Xilinx ISE. Comparative analysis shows that the hybrid adder achieves notable reductions in power, area, and delay compared to traditional PPAs and other approximate designs, making it a compelling solution for power-constrained systems in approximate computing domains.

Keywords: Approximate Adder, Kogge-Stone Adder, Ladner-Fischer Adder, Parallel Prefix Adder (PPA), Brent-Kung Adder, Sklansky Adder.

I. INTRODUCTION

In the modern era of digital electronics, the demand for high-speed, energy-efficient, and area-optimized arithmetic circuits has surged dramatically. Among these, the adder is one of the most fundamental components used in a variety of computational tasks. From arithmetic logic units (ALUs) and digital signal processors (DSPs) to multimedia systems and machine learning accelerators, adders are ubiquitous and play a vital role in determining the overall performance and efficiency of digital systems. As the complexity and scale of data-centric applications continue to grow, there is a pressing need for novel adder architectures that can meet stringent power and area constraints while maintaining acceptable accuracy and performance. One class of adders that has received considerable attention due to their high-speed capabilities is the PPA. PPAs such as Kogge-Stone, Brent-Kung, Sklansky, and Ladner-Fischer provide logarithmic time delay by computing the carry bits in parallel using a network of generate and propagate signals. These structures are well suited for high-performance applications where speed is a critical metric. However, the major drawback of PPAs is their increased area and power consumption due to the large number of prefix operators and interconnections involved in the carry generation process.

Simultaneously, a new design paradigm known as Approximate Computing (AxC) has emerged as a promising solution to the challenges posed by power and area constraints. Approximate computing leverages the fact that many modern applications—particularly those in the domains of image processing, audio/video streaming, machine learning, and sensor data analysis—do not always require perfectly accurate computations. These applications are inherently error-tolerant, meaning that small inaccuracies in the computation do not significantly impact the perceived quality of the output. By deliberately introducing controlled imprecision into the hardware, approximate computing allows for substantial savings in power, delay, and silicon area. The fusion of these two domains—parallel prefix adders and approximate computing—

forms the basis of this project. In this work, a Hybrid Approximate Parallel Prefix Adder architecture is proposed that intelligently partitions the adder into two regions: an approximate Least Significant Part (LSP) and an accurate Most Significant Part (MSP). This hybridization allows for aggressive approximation in the LSP, where inaccuracies are less impactful, while preserving accuracy in the MSP to ensure that the final computation remains within acceptable bounds. The proposed architecture combines the strengths of Kogge-Stone and radix-4 adder topologies to optimize the trade-off between power, delay, and accuracy.

II. RELATED WORKS

Padmanabhan Balasubramanian and Douglas L. Maskell (December 2024) introduced a novel fast bipartitioned hybrid adder (FBHA) that combines carry-select and carry-lookahead logic. In this design, the less significant part is implemented using carry-lookahead logic, while the significant part employs carry-select logic. The 32-bit FBHA demonstrated significant improvements over traditional adders, including a 19.8% reduction in delay compared to a carry-lookahead adder and a 46.5% reduction in area compared to the Kogge-Stone adder.

Hafsa and Aksa David (May 2022) explored the design of an area-efficient Wallace Tree multiplier incorporating approximate 4:2 compressors and Kogge-Stone adders. The study demonstrated that integrating approximate components in the multiplier's final addition stage could reduce area and improve speed without significantly compromising accuracy. Asma Iqbal et al. (December 2023) presented a performance-efficient and fault-tolerant approximate adder design. The proposed adder incorporated fault-tolerant techniques to enhance reliability while maintaining the benefits of approximate computing, making it suitable for applications where both efficiency and fault tolerance are critical.

R. Jothin et al. (June 2023) conducted a comprehensive comparison of high-performance adders, including Brent-Kung, Kogge-Stone, Ladner-Fischer, and Sklansky architectures, in the context of hybrid and error-tolerant applications. The study provided insights into the trade-offs between different adder designs concerning area, delay, and power consumption, aiding in the selection of appropriate architectures for specific application requirements.

Anum Khan and Subodh Wairya (2024) proposed a sparse Kogge-Stone adder architecture that integrates a hybrid carry prefix generator to enhance power efficiency and reduce area. This design is particularly beneficial for applications requiring intensive floating-point computations, offering improvements in both speed and power consumption over traditional Kogge-Stone adders.

Sandeep N. Uttarkar and K. B. Ramesh (2024) presented a high-frequency 16-bit full adder architecture optimized for low latency. Implemented using Verilog HDL and synthesized on Xilinx FPGA platforms, the design leverages techniques like carry-select and carry-lookahead to achieve enhanced performance, making it suitable for high-speed computing applications.

Hema Singaravelan and Dr. Kiran V (2024) investigated the realisation of a 32-bit Kogge-Stone hybrid adder utilising conventional cells from multiple logic families, including CMOS, Pseudo NMOS, and Modified Gate Diffusion Input (MGDI). Their results demonstrate that Pseudo NMOS logic realises a 14.03% reduction in area, but MGDI provides a 54.43% decrease in power consumption relative to conventional CMOS technology, underscoring the promise of alternative logic architectures in adder design.

III. SYSTEM METHODOLOGY

The design and development of a hybrid power and area efficient parallel prefix adder based on approximate computing follow a systematic methodology that integrates architectural innovation, hardware design, and performance optimization. The aim is to achieve a highly optimized arithmetic unit that meets the requirements of low power consumption and minimal area usage while maintaining acceptable computational accuracy. This methodology is divided into distinct phases starting from problem formulation to simulation, synthesis, and final analysis.

3.1 Parallel Prefix Adder

In Parallel Prefix wind case, twofold development routinely imparts in articulations of bring advancement sign, pass on spread sign, pass on sign, and the aggregate sign, at each piece position ($1 \leq I \leq n$) [6], by far most of these alerts can be obtained through regard to the condition underneath

A parallel prefix circuit computes N outputs YN , ..., Y1 from N inputs XN , ..., X1 using an arbitrary associative operator •, defined

$$Y_N = X_N \bullet Y_{N-1}$$

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁝⁝ Peer-reviewed & Refereed journal ⁝⁝ Vol. 12, Issue 7, July 2025

DOI: 10.17148/IARJSET.2025.12756

In majority of the prefix computations, the intermediate signals ZN:N , ..., Z1:1 are pre-computed. The prefix signals YN:1, ..., Y1:1 are produced from prefix networks by combining these intermediate signals. Final outputs are post-computed from the inputs and the prefixes. Parallel-prefix adders (PPA) use parallel-prefix scheme in order to realize fast computation of carry. Different parallel-prefix algorithms result in diverse adders characterizing diverse performances. However, in every architecture of PPA, the initial stage (computation of bit-level generate and propagate signals) and the final stage (computation of sum bits) remain the same. They differ only in the intermediate stage (arrangement of the intermediate carry computation levels). The performance of a parallel prefix adder is decided through many design choices such as recurrence equations used to implement the carry tree, valency of the carry graph, tree sparseness, carry tree topology, logic family, and so on. The choice of each of these factors greatly impacts the overall performance of the adder. The parallel prefix algorithm based adder accepts the inputs AN , ..., A1, BN , ..., B1 and Cin and produces the sum output SN , ..., S1 using the Generate (G) and Propagate (P) prefix signals.
The parallel prefix addition based on Weinberger's recurrence equations (Weinberger 1958) is executed in three stages as shown below.
i) Pre-computation: Computation of bit-level generate (gi) and propagate (pi) or (xi) is performed according to the following equations

$$g_i = A_i.B_i$$
$$p_i = A_i + B_i$$
$$x_i = A_i \oplus B_i$$

ii) Prefix processing: The prefix operator • is defined for producing group level generate and propagate signals, as given below

$$(G_{i:k+1}, P_{i:k+1}) \bullet (G_{k:j}, P_{k:j}) = (G_{i:k+1} + P_{i:k+1}.G_{k:j}, P_{i:k+1}.P_{k:j})$$

Here, $i \geq k \geq j$

iii) Post-computation: Generation of sum bits is done according to
$S_i = x_i \oplus G_{i-1:0}$
The block diagram of the afore stated PPA using Weinberger's recurrence equations is illustrated

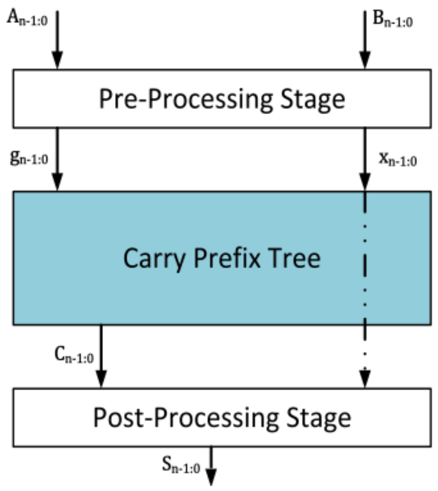


Figure 1 Block Diagram of Parallel Prefix Adder

In the prefix tree of PPAs, outputs at any significant position depends on all inputs of equal or lower significant bits. Further, each of the input bits influence all the outputs of equal or higher significance. As the operator • is associative, the individual operations can be carried out in any order. The input bits (xi, xi−1, ., xk) are grouped to produce the intermediate group variable Yi:k. At subsequent levels of carry tree, sequences of group variables can be combined using the associative operator, thus resulting in m levels of intermediate group variables Y l i:k . The group variable Y l i:k denotes the prefix result of the input bits (xi , xi−1, ., xk) at level l. The prefix variables at the last level m ought to cover all the inputs from the respective bit index to the bit index 0. The group variables in a prefix carry tree can therefore be represented as given below.

$$Y^0_{i:i} = x_i$$
$$Y^l_{i:k} = Y^{l-1}_{i:j+1} \bullet Y^{l-1}_{j:k} \qquad k \leq j \leq i; \quad l = 1, 2, \cdots, m$$
$$y_i = Y^m_{i:0} \qquad i = 0, 1, \cdots, n\text{-}1$$

In the above equations, the variable Y l i:k represent the group generate and group propagate signals, which depict the generation and propagation of carry signals. There exist many algorithms to produce the carries required for addition on the basis of the bit grouping properties as shown in above equation. As the binary addition operation using prefix operator describes a combinational input to output relationship, they can be realized using logic networks. The group variables produced at the level l are Y l i:k = Gl i:k , Pl i:k . The generate signals Gm i:0 obtained from the mth stage, represent the carry signals ci , which are then employed in the post-processing stage to compute the final sum bits Si.

There are many ways by which the prefix processing can be done. Based on the arrangement of the prefix nodes, numerous parallel prefix structures have evolved. Each of the classical prefix structures present different trade-offs in terms of the number of logic stages in the carry tree, lateral fan-out and number of horizontal wiring tracks. Some of the most commonly used prefix structures based on Weinberger's recursion are discussed in the following sub-section.

3.2 Hybrid Approximate Parallel Prefix Adders (HAxPPAs)

The proposed methodology focuses on the design and evaluation of Hybrid Approximate Parallel Prefix Adders (HAxPPAs) to achieve enhanced performance, power efficiency, and energy optimization in arithmetic circuits.
The hybrid adder is strategically partitioned into three operational zones: the Approximate Parallel Prefix Adder (AxPPA) block for the least significant bits (LSBs), the Radix-4 adder block for the middle bits, and the Kogge-Stone adder block for the most significant bits (MSBs). This modular design is intended to exploit the trade-offs between speed, accuracy, and silicon area, as each block contributes its specific advantages to different parts of the addition operation based on the sensitivity of bit positions to error and carry propagation.

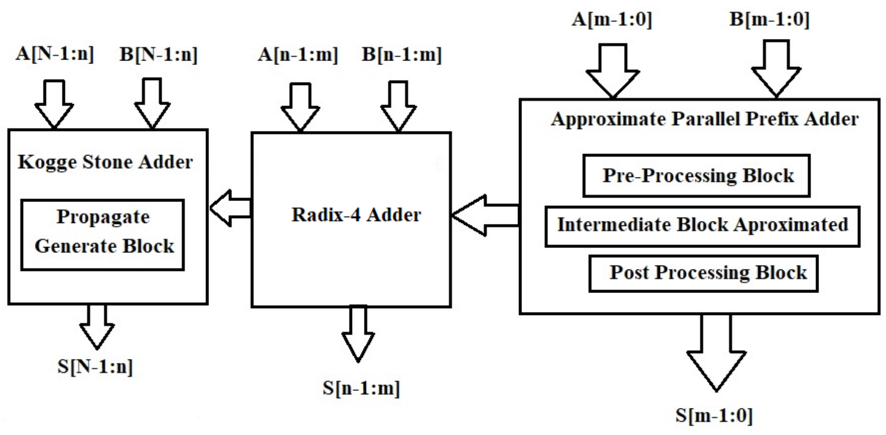


Figure 2 Proposed Block Diagram of Hybrid Adder

Approximate Parallel Prefix Adder (AxPPA) Block

The AxPPA block is employed for computing the LSBs, where approximation has a minimal effect on the overall accuracy of the result. This block reduces critical path delay and power consumption by simplifying the traditional prefix adder structure. Instead of performing full carry computation for every bit, it uses Approximate Prefix Operators (AxPOs) that eliminate certain logic stages. These AxPOs directly wire the generate and propagate signals to the post-processing step, bypassing the intermediate logic trees typical in standard adders like Brent-Kung or Han-Carlson.
This approximation avoids unnecessary computation, which greatly enhances speed and area efficiency. Since small errors in LSBs do not significantly affect the final numerical value, the architecture accepts this compromise. Additionally, this reduction in logic not only accelerates processing but also contributes to lower dynamic power dissipation.

Radix-4 Adder Block for Middle Bits

The central section of the hybrid adder uses a Radix-4 adder, which strikes a balance between speed and resource utilization. In contrast to traditional binary adders that compute one bit at a time, a Radix-4 adder processes two bits per stage, effectively halving the number of carry stages required. It uses a more complex carry lookahead logic that considers multiple bit groupings to anticipate and compute carry bits efficiently.This block serves as a bridge between the low-

DOI: 10.17148/IARJSET.2025.12756

accuracy AxPPA and the high-accuracy Kogge-Stone block. Since middle bits moderately influence the result's precision, the Radix-4 adder provides a trade-off—offering better accuracy than the AxPPA but faster and more compact than the Kogge-Stone adder. It ensures that carry propagation remains under control without significantly inflating delay or power overhead.

Kogge-Stone Adder Block for Most Significant Bits (MSBs)
The most significant bits of the operands are handled by the Kogge-Stone adder, a type of prefix adder known for its parallelism and minimum logic depth. These bits are crucial because errors here can drastically alter the output. The Kogge-Stone structure ensures complete accuracy and fast carries computation due to its logarithmic delay and wide fan-out capability.
In this block, generate and propagate signals are computed and combined across multiple stages to ensure that the carry information propagates as quickly and accurately as possible. Though it is area- and power-intensive compared to other adders, its placement at the MSB end is justified because the risk of error in these bits must be minimized.

Carry Propagation and Interfacing between Blocks
An essential aspect of this architecture is how the carry signals are passed between the blocks. The AxPPA block computes and passes the initial carry-out to the Radix-4 adder, which then computes the carry for its range and sends its carry-out to the Kogge-Stone block. This sequential carry chaining ensures continuity and correctness across the full adder range despite the heterogeneous nature of the blocks.
The interface design must ensure synchronization between these blocks to prevent glitches or timing mismatches. Additional logic may be used to align the timing of signals entering the Radix-4 and Kogge-Stone blocks, especially since the AxPPA may complete computation much earlier due to its simplified logic.

IV. EXPERIMENTAL RESULTS

The hybrid adder architecture in this work is implemented using Verilog HDL for hardware, simulated in ModelSim for functional verification, and synthesized using Xilinx tools (Vivado or ISE) for FPGA deployment. Verilog HDL serves as the primary language to design three interconnected adder blocks: the AxPPA block, which approximates the lower bits to reduce delay and logic usage; the Radix-4 adder block, which efficiently processes mid-level bits using group-based carry computation; and the Kogge-Stone adder, which handles higher-order bits with parallel prefix logic for high-speed addition. These modules are instantiated within a top-level hybrid module, connected via internal wires that pass computed carries between stages. ModelSim is used to simulate the behavior of the entire hybrid system using a testbench that applies various input vectors and monitors the output sum to ensure correctness across all possible input conditions. Finally, the design is synthesized using Xilinx tools, where it is mapped to the logic fabric of the target FPGA. During synthesis, the hybrid architecture demonstrates advantages in timing, area, and power by balancing approximate and accurate computation strategies across different bit segments, offering an optimal trade-off between speed, resource utilization, and computational accuracy.

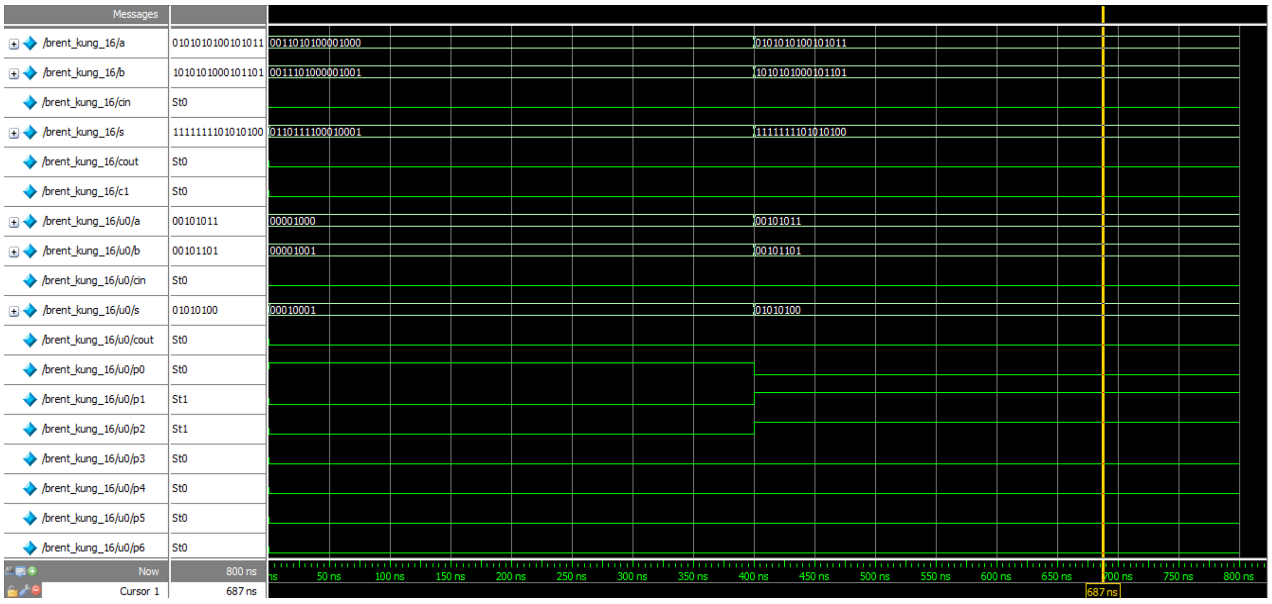

```
Messages
/brent_kung_16/a         0101010100101011   0011010100001000   0101010100101011
/brent_kung_16/b         1010101000101101   0011101000001001   1010101000101101
/brent_kung_16/cin       St0
/brent_kung_16/s         1111111101010100   0110111100010001   1111111101010100
/brent_kung_16/cout      St0
/brent_kung_16/c1        St0
/brent_kung_16/u0/a      00101011           00001000           00101011
/brent_kung_16/u0/b      00101101           00001001           00101101
/brent_kung_16/u0/cin    St0
/brent_kung_16/u0/s      01010100           00010001           01010100
/brent_kung_16/u0/cout   St0
/brent_kung_16/u0/p0     St0
/brent_kung_16/u0/p1     St1
/brent_kung_16/u0/p2     St1
/brent_kung_16/u0/p3     St0
/brent_kung_16/u0/p4     St0
/brent_kung_16/u0/p5     St0
/brent_kung_16/u0/p6     St0
Now                      800 ns
Cursor 1                 687 ns
```

Figure 3 Simulation result of 16-bit approximate Brent-Kung adder

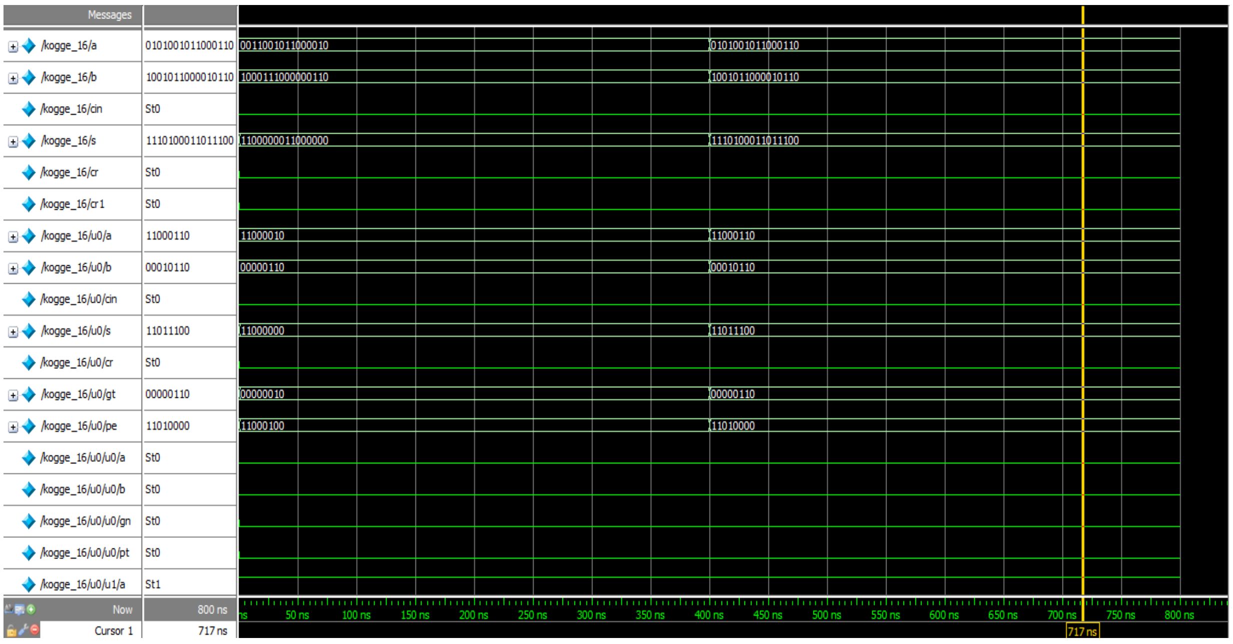
IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁓ Peer-reviewed & Refereed journal ⁓ Vol. 12, Issue 7, July 2025

DOI: 10.17148/IARJSET.2025.12756

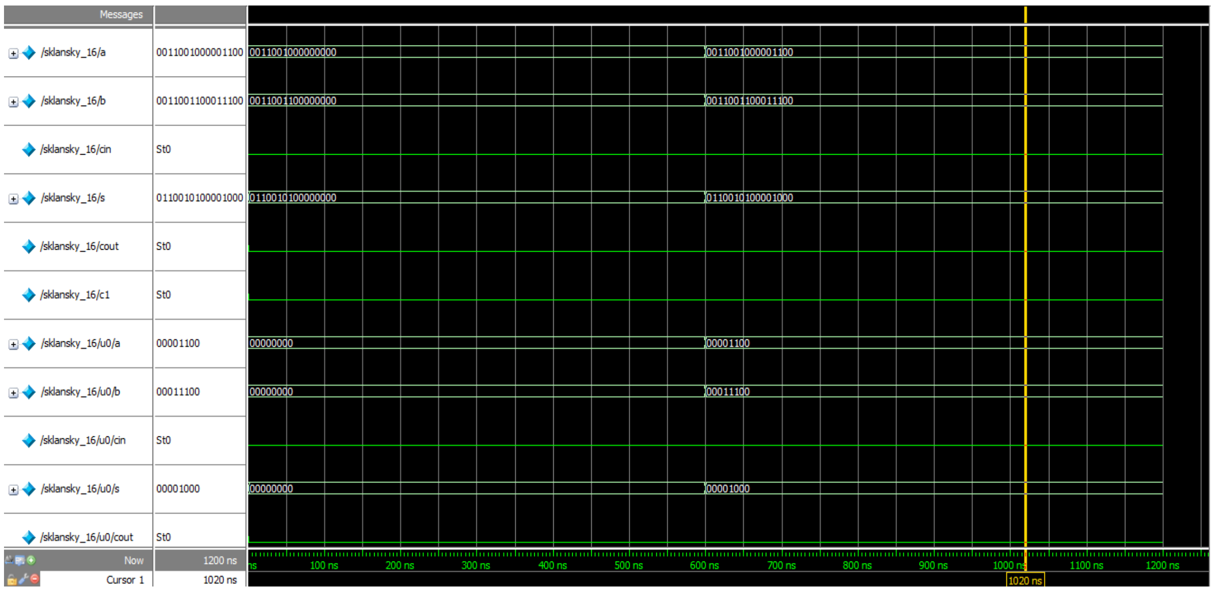
The figure 3 represents the simulation output of a 16-bit approximate Brent-Kung adder, where inputs A and B along with carry-in (CIN) are processed to produce the sum output (S) and carry-out (CR). In this architecture, the lower significant bits (LSBs) are computed approximately to reduce delay and power, while the higher significant bits (MSBs) are processed using the exact Brent-Kung parallel prefix logic. The Brent-Kung strategy structures the prefix computation in a hierarchical manner: 2-bit groups are combined to form 4-bit groups, which are further grouped to form 8-bit prefixes, ensuring minimal logic depth and a regular layout. This arrangement optimizes the design by reducing the fan-out and interconnect complexity, resulting in efficient implementation with reduced area and power consumption.



```
Messages
/kogge_16/a          0101001011000110   0011001011000010   0101001011000110
/kogge_16/b          1001011000010110   1000111000000110   1001011000010110
/kogge_16/cin        St0
/kogge_16/s          1110100011011100   1100000011000000   1110100011011100
/kogge_16/cr         St0
/kogge_16/cr1        St0
/kogge_16/u0/a       11000110           11000010           11000110
/kogge_16/u0/b       00010110           00000110           00010110
/kogge_16/u0/cin     St0
/kogge_16/u0/s       11011100           11000000           11011100
/kogge_16/u0/cr      St0
/kogge_16/u0/gt      00000110           00000010           00000110
/kogge_16/u0/pe      11010000           11000100           11010000
/kogge_16/u0/u0/a    St0
/kogge_16/u0/u0/b    St0
/kogge_16/u0/u0/gn   St0
/kogge_16/u0/u0/pt   St0
/kogge_16/u0/u1/a    St1
Now                  800 ns
Cursor 1             717 ns
```

Figure 4 Simulation result of 16-bit approximate Kogge-stone adder

This figure 4 illustrates the simulation outcome of a 16-bit approximate Kogge-Stone adder. The adder takes A, B, and CIN as inputs and generates the corresponding sum (S) and carry-out (CR). The LSB section employs approximation for faster computation, while the MSB portion uses an exact Kogge-Stone prefix logic. Kogge-Stone adders are known for their minimal logic depth and parallel structure, offering high-speed performance. The adder tree contains multiple parallel propagate and generate cells, which compute carry values efficiently. However, the complexity of interconnect routing in a regular layout increases the circuit area. Despite this, the Kogge-Stone adder remains favorable for high-speed arithmetic operations due to its consistent and balanced fan-out.

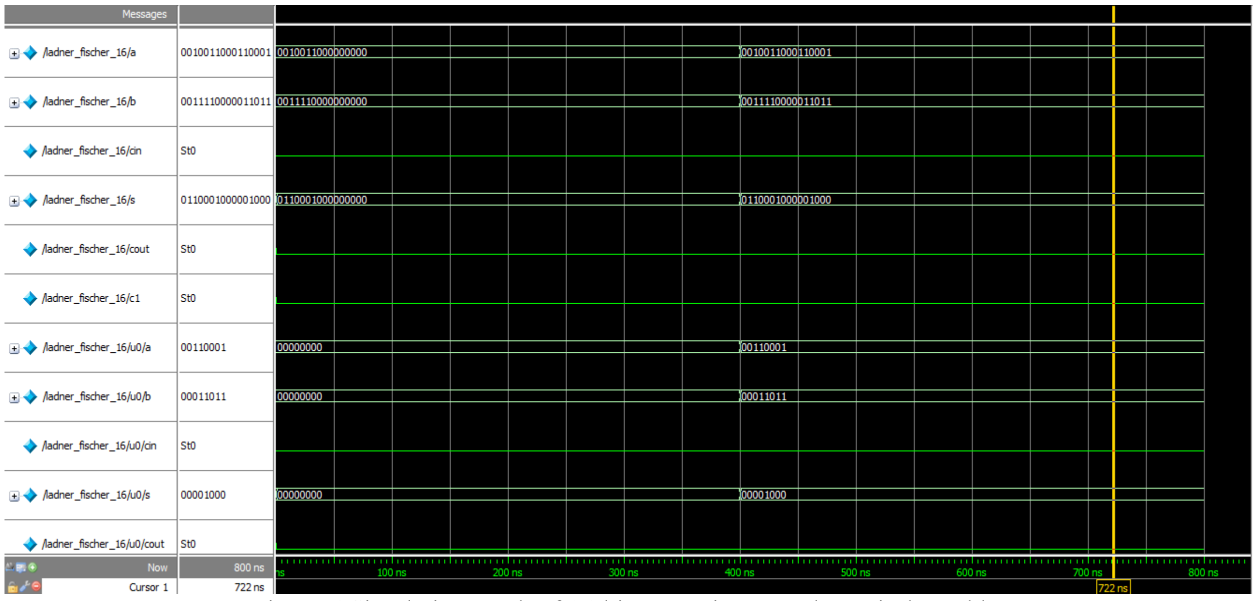


```
Messages
/sklansky_16/a         0011001000001100   0011001000000000   0011001000001100
/sklansky_16/b         0011001100011100   0011001100000000   0011001100011100
/sklansky_16/cin       St0
/sklansky_16/s         0110010100001000   0110010100000000   0110010100001000
/sklansky_16/cout      St0
/sklansky_16/c1        St0
/sklansky_16/u0/a      00001100           00000000           00001100
/sklansky_16/u0/b      00011100           00000000           00011100
/sklansky_16/u0/cin    St0
/sklansky_16/u0/s      00001000           00000000           00001000
/sklansky_16/u0/cout   St0
Now                    1200 ns
Cursor 1               1020 ns
```

Figure 5 Simulation result of 16-bit approximate Sklansky adder

DOI: 10.17148/IARJSET.2025.12756

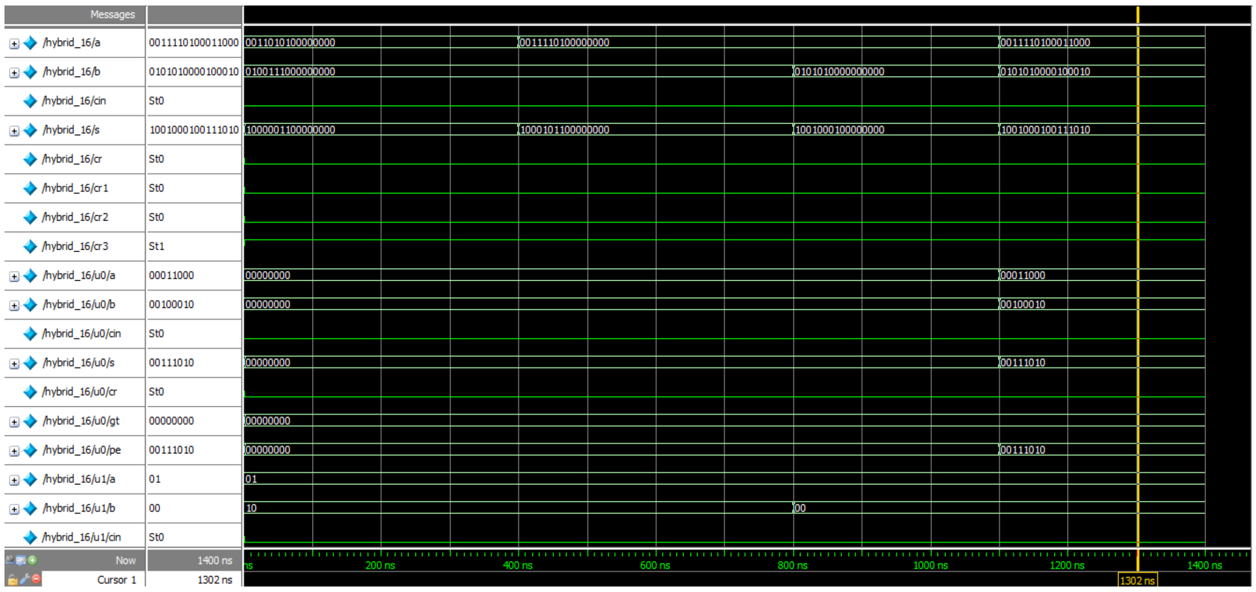
In this figure 5 simulation, the 16-bit approximate Sklansky adder demonstrates how inputs A and B with carry-in (CIN) are used to compute the sum (S) and carry-out (COUT). The lower bits are approximated to reduce complexity, and the upper bits are processed using the exact Sklansky prefix logic. The Sklansky adder is characterized by its divide-and-conquer approach, which significantly reduces the delay in computing intermediate prefixes. However, this comes at the cost of increased fan-out, which doubles at each stage of the adder, potentially causing timing issues and power inefficiencies in larger designs. Despite the high fan-out, Sklansky adders provide a scalable solution for fast carry propagation with reduced gate delay.



Messages	
/ladner_fischer_16/a	0010011000110001
/ladner_fischer_16/b	0011110000011011
/ladner_fischer_16/cin	St0
/ladner_fischer_16/s	0110001000001000
/ladner_fischer_16/cout	St0
/ladner_fischer_16/c1	St0
/ladner_fischer_16/u0/a	00110001
/ladner_fischer_16/u0/b	00011011
/ladner_fischer_16/u0/cin	St0
/ladner_fischer_16/u0/s	00001000
/ladner_fischer_16/u0/cout	St0
Now	800 ns
Cursor 1	722 ns

Figure 6 Simulation result of 16-bit approximate Ladner-Fischer adder

The simulation result figure 6 shown is for a 16-bit approximate Ladner-Fischer adder. Inputs A and B, along with CIN, produce the sum (S) and carry-out (COUT). The LSBs are handled using approximate logic to minimize resource usage, while the MSBs utilize the exact Ladner-Fischer prefix logic. This adder architecture strikes a balance between the regularity of Sklansky and the compactness of Brent-Kung structures. It computes prefix values for odd bits and applies an additional stage to integrate even bit computations. This structure enables moderate fan-out and logical depth, offering an efficient middle ground between complexity and performance, making it suitable for medium-performance and area-constrained applications.



Messages	
/hybrid_16/a	0011110100011000
/hybrid_16/b	0101010000100010
/hybrid_16/cin	St0
/hybrid_16/s	1001000100111010
/hybrid_16/cr	St0
/hybrid_16/cr1	St0
/hybrid_16/cr2	St0
/hybrid_16/cr3	St1
/hybrid_16/u0/a	00011000
/hybrid_16/u0/b	00100010
/hybrid_16/u0/cin	St0
/hybrid_16/u0/s	00111010
/hybrid_16/u0/cr	St0
/hybrid_16/u0/gt	00000000
/hybrid_16/u0/pe	00111010
/hybrid_16/u1/a	01
/hybrid_16/u1/b	00
/hybrid_16/u1/cin	St0
Now	1400 ns
Cursor 1	1302 ns

Figure 7 Simulation result of 16-bit approximate Hybrid Adder

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 12, Issue 7, July 2025

DOI: 10.17148/IARJSET.2025.12756

The figure 7 simulation shows the 16-bit approximate hybrid adder. This design accepts A, B, and CIN as inputs and outputs the sum (S) and carry-out (CR). The LSB section employs approximate logic for faster and simpler computation. The MSB section combines radix-4 and Kogge-Stone adder structures for high-speed operation. The MSB portion is split into two 4-bit blocks, where the carry output from the first block is propagated into the second block. Radix-4 logic provides fast partial sum generation, while Kogge-Stone ensures efficient and parallel carry computation. Although this hybrid approach enhances speed and reduces fan-out, it increases routing complexity and chip area slightly due to the intricate layout required for integration. This design is ideal for applications needing a trade-off between performance and area optimization.

Table 1Comparative Analysis

Types	Area (Gate count)	LUTs	Slices	Delay(ns)	Power(mW)
Brent-kung	156	26	15	16.239	204.39
Kogge-stone	186	31	18	16.059	219.32
Ladner-Fischer	216	36	22	16.059	231.56
Sklansky	168	28	17	17.665	203.76
Hybrid-KS-RD4A	**144**	**24**	**15**	**16.113**	**199.30**

The table 1 presents comparative analysis of different adders. The Brent-Kung adder presents a balanced design in terms of area, delay, and power. With a gate count of 156, 26 LUTs, and 15 slices, it maintains a compact area footprint while offering a moderate delay of 16.239 ns and power consumption of 204.39 mW. Its structured prefix logic with minimal fan-out enables an efficient layout and moderate performance, making it suitable for applications where a trade-off between speed and resource utilization is acceptable. The Kogge-Stone adder, known for its high-speed performance, achieves the lowest delay of 16.059 ns among all architectures due to its fully parallel carry propagation tree. However, this comes at the cost of higher area usage, requiring 186 gates, 31 LUTs, and 18 slices. It also consumes more power at 219.32 mW, reflecting the increased complexity and interconnection density in the layout. This design is ideal for applications prioritizing speed over area and power. In contrast, the Ladner-Fischer adder consumes the highest area and power among the evaluated adders, with 216 gates, 36 LUTs, and 22 slices, and a power consumption of 231.56 mW. Despite matching the delay of the Kogge-Stone adder (16.059 ns), its complex structure and intermediate prefixing stages contribute to higher hardware overhead. It is suitable for systems that demand performance with less concern for resource constraints. The Sklansky adder occupies a middle ground in terms of area and power, using 168 gates, 28 LUTs, and 17 slices. However, it exhibits the highest delay of 17.665 ns, largely due to its increasing fan-out at each stage, which can hinder speed in deeper logic levels. Its lower power consumption of 203.76 mW makes it an appealing option when energy efficiency is more critical than computation speed. Finally, the Hybrid-KS-RD4A adder demonstrates the most optimized trade-off across all parameters. It achieves the lowest area usage with only 144 gates, 24 LUTs, and 15 slices while maintaining a competitive delay of 16.113 ns. Its power consumption of 199.30 mW is also the lowest, showcasing its efficiency. The hybrid design leverages the speed advantages of Kogge-Stone and the area-efficient radix-4 adder, making it highly suitable for low-power, area-constrained, and moderately high-speed applications.

V. CONCLUSION

In this project, a hybrid power and area efficient parallel prefix adder (PPA) based on approximate computing has been successfully designed and evaluated, addressing the growing need for energy-efficient and high-performance arithmetic units in error-tolerant applications such as image processing, machine learning, and digital signal processing. The proposed architecture strategically divides the adder into two segments: an approximate Least Significant Part (LSP) that utilizes simplified logic to reduce power and area, and an accurate Most Significant Part (MSP) that ensures precision in the most impactful bits. By combining the Kogge-Stone and radix-4 adder topologies and introducing approximate prefix operators (AxPOs), the design achieves significant improvements in power efficiency, delay performance, and silicon utilization. Implemented in Verilog HDL, simulated using ModelSim, and synthesized using Xilinx tools, the hybrid adder demonstrates superior performance when compared to traditional and other approximate PPA variants such as AxPPA-BK, AxPPA-KS, AxPPA-LF, and AxPPA-SK. The analysis confirms that the proposed method effectively balances accuracy and efficiency, providing a practical and scalable solution for low-power digital systems. This work showcases the potential of approximate computing in conjunction with hybrid architectural strategies to create optimized arithmetic circuits for modern VLSI applications, paving the way for further innovations in adaptive and application-specific hardware design.

REFERENCES

[1]. Brent, R. P., & Kung, H. T. (1982). A regular layout for parallel adders. IEEE Transactions on Computers, C-31(3), 260–264. https://doi.org/10.1109/TC.1982.1675931

[2]. Khan, A., & Wairya, S. (2024). Efficient and power-aware design of a novel sparse Kogge-Stone adder using hybrid carry prefix generator adder. https://www.researchgate.net/publication/378674104

[3]. Kogge, P. M., & Stone, H. S. (1973). A parallel algorithm for the efficient solution of a general class of recurrence equations. IEEE Transactions on Computers, C-22(8), 786–793. https://doi.org/10.1109/T-C.1973.223892

[4]. Uttarkar, S. N., & Ramesh, K. B. (2024). Design and implementation of low latency 16-bit full adder. Journal of VLSI Design Tools & Technology. https://journals.stmjournals.com/jovdtt/article=2024/view=176362

[5]. Singaravelan, H., & Kiran, V. (2024). 32-bit Kogge-Stone based hybrid adder implemented using standard cells of different logic families. International Journal of Scientific Research and Engineering Trends. Retrieved from https://jusst.org/32-bit-kogge-stone-based-hybrid-adder-implemented-using-standard-cells-of-different-logic-families

[6]. Sklansky, J. (1960). Conditional-sum addition logic. IRE Transactions on Electronic Computers, EC-9(2), 226–231. https://doi.org/10.1109/TEC.1960.5221601

[7]. Anitha, G., et al. (2024). High-speed and power-efficient multiplier and adder designs for linear convolution. In S. S. Satapathy & V. Bhateja (Eds.), Smart Intelligent Computing and Applications (pp. 167–180). Springer. https://doi.org/10.1007/978-981-19-7753-4_16

[8]. Ladner, R. E., & Fischer, M. J. (1980). Parallel prefix computation. Journal of the ACM (JACM), 27(4), 831–838. https://doi.org/10.1145/322217.322232

[9]. Jain, A., et al. (2024). FPGA implementation of efficient 32-bit 3-operand addition using Kogge–Stone parallel prefix adder. In S. S. Satapathy & V. Bhateja (Eds.), Smart Intelligent Computing and Applications (pp. 221–233). Springer. https://doi.org/10.1007/978-981-19-7753-4_22

[10]. Vyas, P., & Bhargava, B. K. (2014). Performance analysis of parallel prefix adders using Verilog HDL. International Journal of Computer Applications, 94(3), 10–14. https://doi.org/10.5120/16306-5792

[11]. Zimmermann, R. (1997). Binary adder architectures for cell-based VLSI and their synthesis. ETH Zurich, Diss. ETH No. 12482. https://doi.org/10.3929/ethz-a-001848232

[12]. Mahalingam, R., & Kumar, R. (2013). A comparative analysis of different types of parallel prefix adders. International Journal of Engineering and Technology (IJET), 5(1), 179–184.

[13]. M. Pashaeifar, M. Kamal, A. Kusha, and M. Pedram, "A theoretical framework for quality estimation and optimization of DSP applications using low-power approximate adders," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 66, no. 1, pp. 327–340, Jan. 2019.

[14]. P. Stanley-Marbell et al., "Exploiting errors for efficiency: A survey from circuits to applications," ACM Comput. Surv., vol. 53, no. 3, pp. 1–39, Jun. 2020.

[15]. H. B. Seidel, M. M. A. da Rosa, G. Paim, E. A. C. da Costa, S. J. M. Almeida, and S. Bampi, "Approximate pruned and truncated Haar discrete wavelet transform VLSI hardware for energy-efficient ECG signal processing," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 68, no. 5, pp. 1814–1826, May 2021.

[16]. P. Pereira et al., "Energy-quality scalable design space exploration of approximate FFT hardware architectures," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 69, no. 11, pp. 4524–4534, Nov. 2022.

[17]. G. Paim, L. M. G. Rocha, G. M. Santana, L. B. Soares, E. A. C. da Costa, and S. Bampi, "Power-, area-, and compressionefficient eight-point approximate 2-D discrete tchebichef transform hardware design combining truncation pruning and efficient transposition buffers," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 66, no. 2, pp. 680–693, Feb. 2019.

[18]. L. Soares, J. Oliveira, E. Costa, and S. Bampi, "An energy-efficient and approximate accelerator design for real-time Canny edge detection," Circuits Syst. Signal Process., vol. 39, no. 12, pp. 6098–6120, 2020.

[19]. G. Paim, H. Amrouch, E. A. C. D. Costa, S. Bampi, and J. Henkel, "Bridging the gap between voltage over-scaling and joint hardware accelerator-algorithm closed-loop," IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 1, pp. 398–410, Jan. 2022.