Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Adaptive Parallel Prefix Adder with Dynamic Topology Selection for High-Performance Arithmetic Units

Priyanka C
Application Engineer,
Ramaiah Academy, Peenya, India

Darshan M R
Application Engineer,
Ramaiah Academy, Peenya, India

Adithya R Das
Department of Electronics and Communication Engineering,
SMVIT, Udupi, India

Sumathi R M
Department of Electronics and Communication Engineering,
AIEMS, Bidadi, India

Keerti N M
Department of Electronics and Communication,
SDMIT, Ujire, Dharmastala, India

***Abstract*— An Adaptive Parallel Prefix Adder with Dynamic Topology Selection is proposed for high-performance arithmetic units to address inefficiencies in conventional fixed-topology adders. Traditional parallel prefix adders such as Brent–Kung, Sklansky, and Kogge–Stone exhibit fixed architectural trade-offs, leading to suboptimal performance under varying input patterns. The proposed design employs an FSM-based adaptive mechanism to dynamically select the most suitable adder topology based on carry propagation characteristics. The propagate signal P is used to estimate carry length and classify operations into short, medium, and long propagation cases. Accordingly, Brent–Kung is selected for short carry chains, Sklansky for medium propagation, and Kogge–Stone for long carry chains. The architecture integrates multiple 4-bit prefix adders and selects outputs at runtime using a control FSM. Implemented in Verilog HDL without utilizing DSP blocks, the design ensures efficient FPGA resource usage. The proposed approach improves average delay and overall performance compared to conventional static adders, making it suitable for high-speed arithmetic applications.**

Keywords— Adaptive Parallel Prefix Adder (APPA), Parallel Prefix Adders (PPA), Brent–Kung Adder, Sklansky Adder, Kogge–Stone Adder, Finite State Machine (FSM), Carry Propagation, Dynamic Topology Selection, High-Speed Arithmetic, Verilog HDL, FPGA

I. INTRODUCTION

High-speed arithmetic operations are essential in modern digital systems such as processors, embedded systems, and real-time computing applications. Among these operations, binary addition plays a critical role in determining overall system performance. To reduce computation delay, Parallel Prefix Adders (PPAs) such as Brent–Kung, Sklansky, and Kogge–Stone are widely used due to their efficient parallel carry computation [1][3].

Each of these adders provides different performance trade-offs. The Brent–Kung adder offers low area and reduced wiring complexity but with increased delay [1]. The Sklansky adder provides a balanced trade-off between speed and hardware complexity [3], while the Kogge–Stone adder achieves the fastest carry propagation at the expense of higher area and routing complexity [2]. In conventional designs, a fixed adder topology is selected, which may not be optimal for varying input patterns.

A key observation is that carry propagation length depends on the input operands. For example, in the addition of 00001111 + 00000001, the carry propagates through multiple bits, resulting in a long carry chain. In contrast, in 10101010 + 01010101, the carry propagation terminates quickly due to alternating bit patterns. This variation indicates that a single adder structure cannot efficiently handle all cases.

To address this limitation, this paper proposes an Adaptive Parallel Prefix Adder with Dynamic Topology Selection. The design dynamically selects the most suitable adder based on the expected carry propagation length. The propagate signal $P = A \oplus B$ is used to estimate carry behavior and classify it into short, medium, and long propagation cases.

An FSM-based control mechanism is used to perform this classification and selection. For short carry propagation, the Brent–Kung adder is selected to minimize area [1]. For medium propagation, the Sklansky adder is used for balanced performance [3]. For long carry chains, the Kogge–Stone adder is selected to achieve high speed [2]. The system integrates these three 4-bit adders and dynamically selects the output based on runtime conditions.

The design is implemented in Verilog and verified using a testbench with different input patterns. It is realized using logic resources without using DSP blocks, ensuring efficient FPGA implementation.

By dynamically adapting the adder topology, the proposed system improves average performance and resource utilization compared to conventional static adders, making it suitable for high-performance arithmetic applications [4][5].

II. RELATED WORK

The design of high-speed arithmetic units has attracted significant research attention due to the need for fast and efficient binary addition in digital systems. Among various adder architectures, parallel prefix adders (PPAs) are widely adopted because they compute carry signals efficiently in logarithmic time. Several prefix structures have been proposed in literature, each offering unique trade-offs between delay, area, and wiring complexity [6][7][12].

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

A. Parallel Prefix Adder Architectures

Brent–Kung adders are area-efficient with low wiring complexity [1]. Kogge–Stone adders provide the fastest carry computation but require large area and complex routing [2]. Sklansky adders offer a balanced trade-off between speed and hardware complexity [3].

B. Optimization Techniques

Various improvements such as hybrid adders and spanning-tree structures have been proposed to optimize performance, power, and fan-out [9], [14]. Technology scaling has further enhanced speed and efficiency, but trade-offs between area, delay, and power still exist [15][16].

C. Limitations of Fixed Adders

Most existing adders use fixed topologies, which do not adapt to varying input patterns. Since carry propagation depends on input data, this leads to inefficient performance.

D. Motivation

To overcome this limitation, the proposed Adaptive Parallel Prefix Adder (APPA) dynamically selects the best adder (Brent–Kung, Sklansky, or Kogge–Stone) using FSM-based control, improving delay and resource utilization [10][13].

E. Comparative Summary

TABLE I. SUMMARY OF PREFIX ADDERS

Parameter	Brent–Kung Adder	Sklansky Adder	Kogge–Stone Adder
Logic Depth	Medium ($\approx 2\log_2 n - 1$)	Low ($\approx \log_2 n$)	Very Low ($\approx \log_2 n$)
Area	Low	Medium	High
Wiring Complexity	Low	Medium	Very High
Fan-out	Low	High	Low
Speed	Moderate	High	Very High (fastest)
Power Consumption	Low	Medium	High
Best Use Case	Area-efficient designs	Balanced performance	High-speed applications
Limitation	Higher delay	High fan-out issue	Large area & routing issues

Table 1 gives the comparative analysis of the three adders, which shows that the Sklansky adder gives a balanced performance compared to Brent-Kung and Kogge-Stone adders while it has a high fanout issues. Although Kogge-stone adder consumes more power, it is more suitable for high speed applications but complexity in terms of wiring and connectivity increases with the bit size as it would require more logic gates and interconnections.

III. PROPOSED DESIGN

The design of high-speed arithmetic units is a critical aspect of modern digital systems, where the efficiency of addition operations significantly affects the overall system performance. Among various adder architectures, parallel prefix adders (PPAs) are widely used due to their ability to compute carries in logarithmic time complexity. However, traditional PPAs such as Brent-Kung, Sklansky, and Kogge-Stone use fixed topologies, which leads to inefficient use of hardware resources when the input-dependent carry propagation varies.

To overcome this limitation, we introduce the proposed design Adaptive Parallel Prefix Adder (APPA), which dynamically selects the most appropriate prefix adder topology based on the expected carry propagation length. The design combines the theoretical foundation of prefix computation with a runtime decision-making mechanism implemented using a finite state machine (FSM).

A. Prefix Adder Fundamentals

Parallel Prefix Adders (PPAs) are widely used in high-speed arithmetic circuits due to their efficient carry computation using parallel structures. The operation of a prefix adder can be divided into three main phases: Pre-Processing, Prefix Computation, and Post-Processing.

1. Pre-Processing Phase

In this phase, the input operands A and B are used to compute the initial Generate (G) and Propagate (P) signals for each bit position.

$$G_i = A_i \cdot B_i$$
$$P_i = A_i \oplus B_i$$

- The Generate signal (G) indicates whether a carry is produced at that bit position, independent of input carry.
- The Propagate signal (P) indicates whether an incoming carry will propagate to the next stage.

These signals form the foundation for carry computation and are also used to estimate carry propagation length, which is essential in the proposed adaptive design.

The carry computation is governed by the recursive relation:

$$C_{i+1} = G_i + P_i \cdot C_i$$

2. Prefix Computation Phase

This phase computes the carry signals efficiently using a tree-based prefix structure. The computation is performed using the prefix operator:

$$(G_k, P_k) \circ (G_j, P_j) = (G_k + P_k \cdot G_j,\ P_k \cdot P_j)$$

- This operation combines adjacent generate and propagate pairs to form group generate and propagate signals.
- For an n-bit adder, the number of prefix levels required is:

$$\log_2(n)$$

➢ Prefix Structures Used in Proposed Design

The proposed adaptive adder integrates three different prefix architectures:

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

- *Brent–Kung Adder* → Optimized for low area and reduced wiring complexity
- *Sklansky Adder* → Provides a balanced trade-off between speed and hardware
- *Kogge–Stone Adder* → Optimized for minimum delay (fastest carry computation)

All three adders compute carry signals in parallel, allowing dynamic selection based on runtime conditions.

3. Post-Processing Phase

In the final stage, the sum bits are computed using the propagate signals and the calculated carry signals:

$$S_i = P_i \oplus C_i$$

- The final carry-out (Cout) is obtained from the most significant carry signal.
- This stage produces the final addition result.

B. Adaptive Control Mechanism

The proposed adaptive parallel prefix adder incorporates a control mechanism that dynamically selects the most suitable adder topology based on carry propagation behavior. This improves performance by avoiding the limitations of fixed adder structures.

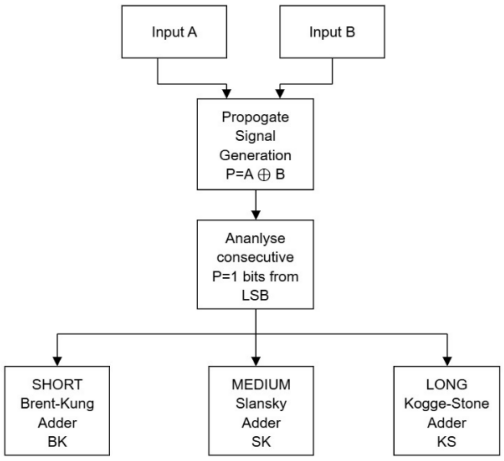

Fig 1: Block diagram of FSM based adaptive parallel prefix adder

1. Carry Propagation Concept

In binary addition, the carry propagation length depends on the input bit patterns.

- In some cases, the carry propagates through multiple stages
- In other cases, the carry terminates quickly

Examples:

- 00001111 + 00000001 → Long carry propagation
- 10101010 + 01010101 → Short carry propagation

This variation clearly shows that using a single fixed adder topology is not optimal for all input conditions.

2. Propagate Signal Generation

The first step involves computing the propagate signals for the given inputs:

$$P = A \oplus B$$

where $A$and $B$are 4-bit input operands.

The propagate signal indicates whether a carry will propagate through a bit position. This information is used to estimate the carry propagation length, which is critical for selecting the appropriate adder topology.

3. Carry Propagation Classification

Based on the propagate signals, the carry behavior is classified into three categories:

- *SHORT Propagation*
  Condition: $\sim P[0] | (P[0] \& \sim P[1])$
  Carry stops early

- *MEDIUM Propagation*
  Condition: $P[0] \& P[1] \& \sim P[2]$
  Moderate carry chain

- *LONG Propagation*
  Condition: $P[0] \& P[1] \& P[2]$
  Carry propagates through most bits

This classification is implemented using combinational logic in the FSM's CHECK state.

4. FSM-Based Control Unit

A Finite State Machine (FSM) is used to control the operation of the adaptive adder. The FSM consists of the following states:

- *IDLE* – Initializes outputs and waits for operation
- *CHECK* – Evaluates propagate conditions
- *SHORT/ MEDIUM / LONG* – Selects appropriate adder
- *OUTPUT* – Produces final result

The FSM transitions based on propagate conditions and ensures proper synchronization with the system clock.

5. Parallel Prefix Adder Modules

The design integrates three different 4-bit parallel prefix adders:

- *Brent–Kung Adder* - Selected for SHORT propagation, Optimized for low area and reduced wiring complexity.
- *Sklansky Adder* - Selected for MEDIUM propagation. Provides balanced performance.
- *Kogge–Stone Adder* - Selected for LONG propagation, Optimized for minimum delay.

Each adder operates in parallel, and their outputs are precomputed.

6. Dynamic Output Selection

Based on the FSM decision, the corresponding adder output is selected:

- *SHORT* → Brent–Kung output
- *MEDIUM* → Sklansky output
- *LONG* → Kogge–Stone output

Control signals:

- BK→ Brent–Kung active
- SK→ Sklansky active
- KS→ Kogge–Stone active

The selected sum and carry-out are assigned to the final output.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

IV. SIMULATION RESULTS

The proposed Adaptive Parallel Prefix Adder with Dynamic Topology Selection was verified through functional simulation using Xilinx Vivado. The design integrates multiple parallel prefix adder architectures and dynamically selects the most efficient topology based on carry propagation behavior.

A comprehensive testbench was used to apply different combinations of input operands A, B, and carry-in (C_{in}). The simulation evaluates correctness of output sum, carry-out, and control signals responsible for topology selection. The results demonstrate that:

- The adaptive control logic successfully classifies carry propagation into short, medium, and long categories.
- The FSM dynamically activates the appropriate adder using control signals (BA, SA, KA).
- The output sum and carry are computed correctly for all input conditions.
- The design efficiently operates using FPGA logic resources without relying on DSP blocks.
- These observations confirm that adaptive prefix architectures can improve performance compared to fixed-topology adders by selecting optimal structures at runtime.

A. Simulation Waveform

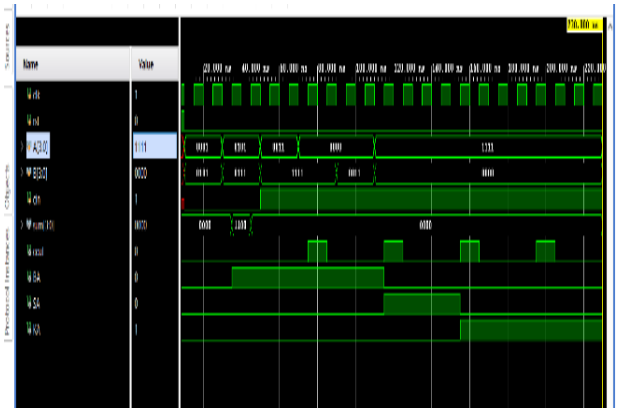

Fig 2: Simulation Waveform

The waveform represents the time-domain behavior of the adaptive adder under varying input conditions.

➢ *Signals*:

- Clock (clk): Controls sequential FSM operation.
- Reset (rst): Initializes the system.
- Inputs (A, B, cin): Multiple test vectors are applied to validate functionality.
- Outputs (sum, cout): Correct arithmetic results are observed for each case.

➢ *Control Signal Behavior:*

- BA (Brent–Kung Active): Enabled when carry propagation is short.
- SA (Sklansky Active): Enabled for medium propagation.
- KA (Kogge–Stone Active): Enabled for long propagation.

➢ Key Observations:

The FSM moves through states (CHECK → SHORT/MEDIUM/LONG → OUTPUT) to select the appropriate adder based on carry propagation. Only one control signal (BA, SA, or KA) is active at a time, ensuring correct and conflict-free selection.

The sum and carry outputs are generated accurately, confirming correct functionality for all input cases. Dynamic switching reduces delay by selecting the best topology, improving overall performance and validating the adaptive mechanism.

B. RTL Schematic

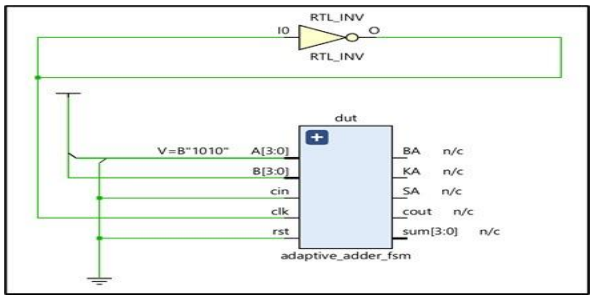

Fig 3: RTL Schematic

The RTL schematic shows the structural view of the synthesized adaptive adder design. The main module, adaptive_adder_fsm, represents the complete system.

The inputs include A[3:0] and B[3:0] as binary operands, along with cin as the carry input, and clk and rst as control signals. The outputs include sum[3:0] and cout for the final result, along with BA, SA, and KA as adder selection signals.

Internally, the design consists of multiple adder blocks (Brent–Kung, Sklansky, and Kogge–Stone). An FSM controls the selection process, and only one adder output is chosen at a time based on carry propagation classification.

The schematic confirms proper interconnection of combinational and sequential logic. No DSP blocks are used, indicating efficient implementation using FPGA logic resources. Some ports may appear as unconnected (n/c), which is normal in RTL representation. This validates the feasibility of the adaptive design.

V. CONCLUSION

The proposed Adaptive Parallel Prefix Adder (APPA) effectively addresses the limitations of conventional fixed-topology adders by dynamically selecting the most suitable prefix architecture based on carry propagation characteristics. By integrating Brent–Kung, Sklansky, and Kogge–Stone adders with an FSM-based control mechanism, the design achieves improved performance and efficient resource utilization.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

The propagation-based classification enables optimal selection for different input conditions, reducing unnecessary delay and enhancing overall speed. Simulation results confirm the correctness of the design and demonstrate better average delay compared to static adders.

Thus, the proposed approach provides a scalable and efficient solution for high-performance arithmetic units, making it suitable for modern digital system implementations.

REFERENCES

[1] R. P. Brent and H. T. Kung, "A Regular Layout for Parallel Adders," IEEE Trans. Computers, 1982.

[2] P. M. Kogge and H. S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," IEEE Trans. Computers, 1973.

[3] J. Sklansky, "Conditional-Sum Addition Logic," IRE Trans. Electronic Computers, 1960.

[4] S. Knowles, "A Family of Adders," Proc. IEEE Symposium on Computer Arithmetic, 2001.

[5] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs, 2010. this reference not again

[6] D. Harris, "A Taxonomy of Parallel Prefix Networks," Proc. IEEE Asilomar Conf., 2003.

[7] R. Zimmermann, "Binary Adder Architectures for Cell-Based VLSI," IEEE Trans. Computers, 1997.

[8] H. Ling, "High-Speed Binary Adder," IBM Journal of Research and Development, 1981.

[9] T. Lynch and E. E. Swartzlander, "A Spanning Tree Carry Lookahead Adder," IEEE Transactions on Computers, vol. 41, no. 8, pp. 931–939, 1992.

[10] J. Rabaey, A. Chandrakasan, and B. Nikolic, Digital Integrated Circuits: A Design Perspective, 2nd ed., Prentice Hall, 2003.

[11] S. Borkar, "Design Challenges of Technology Scaling," IEEE Micro, vol. 19, no. 4, pp. 23–29, 1999.

[12] V. G. Oklobdzija, "High-Speed VLSI Arithmetic Units: Adders and Multipliers," in Design of High-Performance Microprocessor Circuits, IEEE Press, 2000.

[13] A. Beaumont-Smith and C. C. Lim, "Parallel Prefix Adder Design," Proc. IEEE International Conference on Computer Design, 2001.

[14] Y. He and C. H. Chang, "A Power-Delay Efficient Hybrid Carry-Lookahead Adder," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 16, no. 10, pp. 1421–1425, 2008.

[15] R. Zlatanovici and B. Nikolic, "High-Performance Adders in 45 nm CMOS Technology," IEEE Journal of Solid-State Circuits, vol. 44, no. 2, pp. 569–583, 2009.

[16] C. Nagendra, M. J. Irwin, and R. M. Owens, "Area-Time-Power Tradeoffs in Parallel Adders," IEEE Transactions on Circuits and Systems II, vol. 43, no. 10, pp. 689–702, 1996