***Naivigyan Journal of Multidisciplinary Research* (NJMR)**
Volume 2, Issue 3, July 2026, Pages: 19-28
Original Article | ISSN (O): 3108-1487

Design and Implementation of 64-Bit Adders Using Various Full Adders

Livin P Wilson[1*] and Siji N M[2]

[1]Assistant Professor, Naipunnya Institute of Management and Information Technology, Pongam, Thrissur- 680308, Kerala, India
[2]Assistant Professor, ACKHM ICA College, Thozhiyoor, Thrissur- 680505, Kerala, India
*Corresponding Author's Email: livin@naipunnya.ac.in

Abstract

Adders are basic arithmetic components that have a big impact on digital systems' area efficiency, power consumption, and performance. 64-bit adders are essential for carrying out arithmetic operations in contemporary high-performance processors, signal processing units, and embedded platforms. In order to assess their influence on overall system performance, this paper describes the design and implementation of 64-bit adders using different full adder architectures. 64-bit Full adders are built using a variety of full adder designs, such Carry Select Adder, Ripple Carry Adder. Verilog HDL is used to model and implement the suggested architectures, and functional simulation is used to verify them. Important performance indicators are examined and contrasted, including hardware utilization, propagation delay. The findings show that the efficiency of 64-bit addition is highly influenced by the full adder architecture selected, with hybrid full adder-based designs providing better speed and power characteristics. This study emphasizes how crucial full adder optimization is when designing high-bit-width arithmetic units for contemporary digital systems.

Keywords: 64-bit Adder, Full Adder Architectures, Ripple Carry Adder, Carry Select Adder, Verilog HDL

Introduction

The ever-increasing demand for faster and more powerful computers requires enhancement in all units of a Central Processing Unit, with the Arithmetic Logic Unit being a pivotal component of this improvement. At the core of the ALU, there is the adder, which is a digital logic circuit used to execute addition operations. With the advancements in the development of processor architecture from 32 bits to 64 bits, there has been exponential complexity and performance requirements for adders. In a 64-bit adder, there is a requirement to execute computations in a shorter time, with a suitable chip size, and lower power consumption. The basic unit of multi-bit adder design is the full adder, named FA. Even though the basic logic of the FA remains constant, variations in its gate-level design tend to affect the performance of adder structures.

This research study attempts to comprehensively examine the design implementation of 64-bit adders by moving beyond the conventional method of concatenation of full adders to examine advanced designs that deal with the natural limitation of the simpler design models. Our research analysis will examine the impact of various designs of the full adder on the final 64-bit adder design regarding the speed, area, and power of the design implementation. This work is organised to begin

with the introduction of the conventional full adder, followed by the description of the main 64-bit adder architectures (RCA, CLA, CSLA), their complexities, merits, and implementation process using HDLs, in addition to concluding the comparisons of the adders through the results of simulations, as well as synthesis. This helps to maximise the benefits of the developments in the future.

Materials and Methods

Background and Full Adder Design Principles

Logic of the Basic Full Adder

The most basic element of a 64-bit adder would be the 1-bit Full Adder (FA). This adder adds two binary digits, A & B, taking into account the input carry, C_in. This process yields two other results, the Sum (S) and the Carry Out (C_out), respectively.

Truth Table and Boolean Expressions

This logic is controlled by the following truth table

Table 1. Truth Table

A	B	Cin	Sum (S)	Carry-out (Cout)
0	0	0	0	0
0	1	0	1	0
1	1	0	0	1
1	1	1	1	1

Mathematically, these are expressed as:

- Sum = A $\oplus$B $\oplus$C_in
- C_out = (AB) + (C_in .(A $\oplus$B))

Gate-Level and Transistor-Level Implementation

In the Standard Gate-Level circuit, two XOR gates implement the Sum and a set of AND/OR gates implements the Carry. However, for enhanced speed realisation in high-speed 64-bit systems, XOR-XNOR-based FAs can be employed. In this approach, by computing the XOR and XNOR of A and B concurrently, this circuit eliminates delays in the critical paths and allows for balanced timing for implementing Sum and Carry signals.

At the level of the CMOS Transistor, there is further efficiency. Rather than the usual static CMOS, one would see either Pass-Transistor Logic (PTL) or Transmission Gates (TG). FA circuits with transmission gates are a greatly preferred alternative, which reduces the number of transistors by a considerable margin of 20-24 against the usual 28 of static CMOS, and eliminates the ‘voltage drop’ drawbacks faced by basic pass-transistors. Such FA architectures increase circuit power savings and reduce silicon area.

Performance Indicators

The effectiveness of a single FA can be judged based on the following:

• Propagation Delay: Temporal difference between the last input transition and S or C_out output.
• Power dissipation: Static (leakage) power and dynamic (switching)
• Area: This is sometimes measured by the total transistor count or the total width of Si wafer.

Design and Implementation of 64-Bit Adder Architectures

Ripple Carry Adder (RCA)

RCA is the most straightforward layout of multi-bit addition. In the design concept of the RCA, it is designed to be as easy as the addition process itself, where the carrying is done in a serial manner from the least significant bit to the most significant bit.

Concept and Structure

A 64-bit RCA is made up of a serial chain of 64 FA modules.

The ith FA module receives three inputs: the ith bit of the operands (Ai and Bi) and the carry from the preceding module (C_(i-1)).

The 'Ripple' term is derived from the fact that the carry from each module (C_out) is connected to the next module's carry input (C_in) as its next stage is purely sequential.

Delay Analysis and Critical Path

The main drawback of the RCA is the linear propagation delay, referred to as O(N). The critical path represents the longest path through which the signal travels, and in this design, the longest path signal travels through the carry chain from the first bit to the final sum and the output of the 64th bit. As C_{63} cannot be computed until C_{62} is available, and so on, the total propagation delay, T_{RCA}, can be approximately calculated as:

TRCA = (N - 1)T

Here, T_{carry} stands for the time for a single FA's carry-out logic. For the 64-bit adder, the cumulative time presented in the following formula becomes a bottleneck for high-frequency clocks.

Area and Power Properties

Despite its problems with latency, the RCA is very area-efficient. This requires the lowest number of gates. This is very useful in the case of a low-speed system where silicon area is of prime concern. Also, it has good power characteristics at the per-gate level because there is no redundant logic involved in it. But in the case of 64-bit RCA, total power dissipation tends to be high because of "glitching," that is, mid-transition switching of the high bits before the carry signal has stabilized.

TRCA = (N - 1)T

Here, T_{carry} stands for the time for a single FA's carry-out logic. For the 64-bit adder, the cumulative time presented in the following formula becomes a bottleneck for high-frequency clocks.

Carry Look-Ahead Adder (CLA)

The Carry Look-Ahead Adder (CLA) was developed as a solution to the “Carry Propagation Problem”, which has been identified as a problem within Ripple Carry Adders. In a 64-bit RCA adder, this dependency on the carry signal translates into a bottleneck due to latency effects. The CLA eliminates this dependency because it looks ahead and computes carries simultaneously with sums.

Generate (G) and Propagate (P) Signals

CLA logic's essence is in the two intermediate signals defined for each bit position i:

- Generate (G_i): G_i = A_i. B_i. If both inputs are 1, carry generation is independent of the incoming carry.
- Propagate (P_i): P_i = A_i ⊕B_i. If the input is 1, it will propagate the incoming carry to the next stage.

Carry-Out Equation Derivation

The carry for any stage can be expressed as C_{i+1} = G_i + (P_i .C_i). By recursively substituting this formula, the look-ahead logic can predict carries without waiting for the ripple. For a 4-bit block, the logic expands as:

- C_1 = G_0 + P_0.C_0
- C_2 = G_1 + P_1G_0 + P_1P_0C_0
- C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0
- C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0
- Hierarchical 64-bit Structure
- Implementing a single 64-bit CLA logic gate is physically impossible due to high fan-in requirements. Instead, a Hierarchical CLA is used. The 64 bits are divided into sixteen 4-bit CLA blocks. Each block produces "Block Generate" (G_B) and "Block Propagate" (P_B) signals, which are then processed by a second-level (and sometimes third-level) Look-Ahead Unit to determine carries between blocks.
- Performance and Implementation
- The Delay Analysis reveals a major improvement; while RCA delay is O(N), CLA delay is logarithmic (log N). However, this speed comes at the cost of Area and Power. The look-ahead circuitry requires significantly more gates and complex routing, leading to higher silicon area and increased power consumption due to high fan-out and logic switching.

Carry Select Adder (CSLA)

The CSLA is a high-speed architecture primarily intended to reduce propagation delay through speculation.1 Speculating upon the incoming carry signal-instead of waiting for it-the CSLA prepares the results of both possible carry scenarios in advance.

Operation Principle

The key principle of CSLA is the concurrent evaluation of two variants of the sum.2 For every block of bits, there exist two independent addition operations that are computed in parallel: one assumes 3C_in = 0 and the other one assumes 3C_in = 1. 4After the actual carry bit has arrived from the previous stage, it becomes a control signal for a multiplexer (MUX), which selects the pre-computed sum in one cycle.

block partitioning and hierarchy "

A 64-bit CSLA is never implemented in a single block form. It employs Block Partitioning, where the 64-bit data is split into smaller pieces, for instance, 4-bit or 8-bit blocks. These blocks can either be of equal size, which is referred to as linear CSLA, or of increasing sizes, which is referred to as square-root CSLA.

Standard vs. Modified CSLA

In The Standard CSLA, there are two Ripple Carry Adders (RCAs) for every block, which makes it highly hardware-intensive. In this regard, an improvement in this design was made in The Modified CSLA, which substituted an RCA for “carry-in = 1” with that of a “Binary to Excess-1 Converter (BEC)” circuit. The BEC circuit merely adds 1 to an RCA for “carry in = 0,” which greatly lowers the transistor count and overhead in terms of area without affecting speed.

Performance and Implementation

In terms of Delay Analysis, CSLA represents a significant improvement in terms of performance as compared to the RCA. Even though the CLA is theoretically faster O(\log N), the CSLA is often the most hardware-efficient solution for 64-bit widths because it circumvents the

complex routing and high fan-in associated with large look-ahead trees. However, its Area and Power are higher than the RCA due to redundant calculation logic and the addition of a set of MUX units.

Advanced Adder Architectures

Beyond the usual topologies, still more advanced architectures are employed to meet extreme timing requirements imposed by modern 64-bit processors. These designs concentrate on bypassing the carry chain completely or reorganising the carry chain into highly efficient tree structures.

Carry Skip Adder

The Carry Skip Adder (also known as a Carry-Bypass Adder) represents a development of the Ripple Carry Adder aimed at reducing the time spent in carry propagation.1 It works on the principle that if any block of bits has a propagate signal (2P_i) equal to 1 for all bits in that block, then a carry entering that block will simply "skip" over it to the next block.3. This allows the carry to bypass the slow ripple path inside the block using only a simple AND gate to monitor the block's propagate signals and a 2-to-1 multiplexer. In this way, for a 64-bit implementation, the worst-case delay is drastically reduced with respect to a standard RCA, with respect to when the carry has to ripple from the LSB to the MSB, at the cost of a very limited area overhead.

Parallel Prefix Adders (PPA)

Parallel Prefix Adders are the best in high-speed arithmetic. This type of adder comprises the Kogge-Stone and Brent-Kung adders. Instead of performing carry operations in a linear form and a block-based approach, PPAs address the problem of generating carries within a "tree" setup of logic elements to compute carries simultaneously.

- Kogge-Stone Adder: Its logic depth has been observed lowest and fan-out is also high. Its performance is fastest (log N) time) but its routing cost and space are enormous, so it can be called "wiring-congested" for 64-bit architectures.
- Brent-Kung Adder: More area-effective tree adder architecture. Although it possesses more logic depth compared to the Kogge-Stone adder, it uses fewer gates and wires, which improves the trade-off for power-effective designs.

These adders follow three stages:

- 'Pre-processing' (generation of 6P & 7G),
- Prefix Tree computation of all carries,
- 'Post-processing' computation of the sum.

Parallel Prefix Adders are predominantly favoured for 64-bit operations in advanced processors for superior processing speed. These adders are also more complex than their counterparts.

Methodologies for Implementation and Evaluation

From a theoretical 64-bit adder circuit design to a silicon-based implementation, the design follows a strict process that is based on electronic design automation, and this ensures that a structured process ensures a translation of the mathematical logic of the design of the adder into a silicon-based implementation.

Hardware Description Language (HDL)

The process involves RTL (Register-Transfer Level) programming using VHDL or Verilog description languages. There are basically two methods for designing a 64-bit adder. These are:

- Behavioural modelling, which uses the operator “+” so that the synthesis tool can pick the appropriate design.

- Structural modelling, where the design of the connection between the adders and the gates, as well as the connection in the prefix trees, is carried out by hand to control the hardware design.

Functional Simulation and Verification

Before the production of any hardware using these tools, the code has to go through the process of Functional Verification. The test environment is designed for testing inputs of different corners, such as the sum of the max values for \(2^{64}-1\) inputs to the adder. With the aid of ModelSim/Vivado Simulator tools, the output of the hardware environment is compared to the "gold model."

Synthesis & TechnologyMapping

After the Synthesis stage completes, the processing step creates an HDL code that is further translated to produce the netlist of physical gates. This is followed by Technology Mapping, whereby the generic logic is converted to the library of choice. In the case of target mapping for an FPGA (Xilinx/Intel processor), the netlist is further translated to Look-Up Tables (LUTs), along with carry chains. In the ASIC processor target mapping, the netlist is further translated to the standard cell library (7nm/28nm CMOS).

Physical Design: Place and Route (P&R)

The synthesis netlist is then processed through Place and Route. "Placement" takes each gate into its specific position on the silicon die, while "Routing" determines the metal wiring in order to connect them. In 64-bit adders, routing is sometimes an important factor; very congested architectures like Kogge-Stone may experience extreme "interconnect delay" at this step.

Post-Layout Simulation

The final step is Post-Layout Simulation-or sometimes Timing Simulation. In contrast to the initial verification, this step now incorporates "parasitic" data-real-world delays due to wire resistance and capacitance. Doing so makes certain that the 64-bit adder satisfies its timing constraints and operates at the desired clock frequency.

Performance Metrics

The designers therefore rely on the following three main pillars of performance for the evaluation of the effectiveness of an implementation of a 64-bit adder addition process: Speed, Silicon Footprint, and Energy Efficiency. The three factors enable the comparison of the RIPPLE CARRY ADDER and CARRY LOOK-AHEAD ADDER architectures.

Propagation Delay

Propagation delay is the most important parameter in high-speed computing, and in a 64-bit environment, it is the time duration that starts with the change of the input signal and leads to the change of the output signal. Typically, the maximum delay usually occurs on the critical path, and in most Adders, the critical path involves the carry going from the least significant bit A_0, B_0, to the most significant bit of the Sum, Sum_63, or the last carry out, C_out. Based on this, the clock speed of the processor is determined. For example, though the delay of the RCA is O(n), the aim of the Parallel Prefix Adder is O(log n), and this significantly reduces the seconds needed for a 64-bit addition operation to nanoseconds.

Area Utilization

The Area: This comprises the resource requirements of the adder within the silicon chip. The area parameters for FPGAs have been expressed in terms of the number of LUTs, flip-flop elements, and the number of multiplexers used, whereas for the ASIC, it is normally referred to by the term ‘Gate Count' based on the area expressed in terms of square micrometres (\mu m^2). There is a visible correlation between the area parameters and time parameters. The fast adders such as Carry Select or Kogge-Stone adder require a considerably larger area than the RCA adder.

Power Usage

Power Consumption is emerging as a popular trend in the mobile and data centre segments. There are two types of Power Consumption:

- Static Power: The current that leaks even when there is no action within the circuit and is a strong function of the type of transistor employed at a transistor level.
- Dynamic Power: The power consumed during a transition of a signal. It can be assumed that high-speed adders with a high fan-out and a complex expression tree may consume more dynamic power.

The post-synthesis tool may estimate these values after analysing the switching activity; thus, the designer will be able to make sure that the 64-bit adder stays within the limit of TDP in the overall system.

Full-Adder Variants in Architectures

- Scenario 1: 64-bit FA using RCA
- Scenario 2: 64-bit FA using CLA.
- Scenario 3: 64-bit FA using CSLA.

Results and Discussions

Empirical testing of 64-bit adders has shown large differences among architectures. Synthesis of these designs using a standard cell library or FPGA resources can determine precisely what the costs are for speed.

Delay & Area Comparison

From the Graphs of Propagation Delay, the Ripple Carry Adder (RCA) is also unsuitable for operation at higher speeds for 64-bit addition, with the delay of propagation of the adder being nearly eight times longer than that of the Kogge-Stone Parallel Prefix Adder. The Carry Look-Ahead Adder (CLA) and the Carry Select Adder (CSLA) lie in between, with a 'speed-up' factor that is quite appreciable but at the expense of higher area of silicon. From the Graph of Resource Utilization, the Kogge-Stone adder consumes 3.5 times more Area than that of an RCA because of its mammoth prefix tree.

Power Consumption Analysis

Analysis of Power Consumption specifies that dynamic power is more prominent in the case of 64-bit adders. The CSLA consumes greater power as it requires calculations of two sums to decide which one to select. Notably, despite using more gates, smaller logic depth of CLA generates lower 'glitching' power, as signals become stable faster.

Table 2 Power Consumption

Architecture	Delay (ns)	Area (Gate Count)	Total Power (mW)
RCA	12.45	580	0.85
CLA (Hierarchical)	3.12	1150	1.42
CSLA	5.40	920	1.10

Influence of Full Adder Design

The choice of the best 1-bit Full Adder has a dramatic effect on this result as well. The RCA with an XOR/XNOR optimized FA has a carry path delay that is approximately 15% less than that of a regular gate-level Full Adder. In a 64-bit sequence, this small 15% improvement multiplies, resulting in a dramatic reduction of the critical path delay without changing the top-level architecture.

Trade

The Trade-off Analysis targets the "Power-Delay-Area" product. The RCA is distinct for sensors that need less power, but Parallel Prefix adders are mandatory for 64-bit CPU configurations. Next comes Scalability. The RCA degrades linearly with our move to 128-bit processors, making the RCA more or less redundant. On the opposite side, the CLA & Prefix adders require a complexity of (log N) that grows exponentially for wiring complexity.2

Outputs

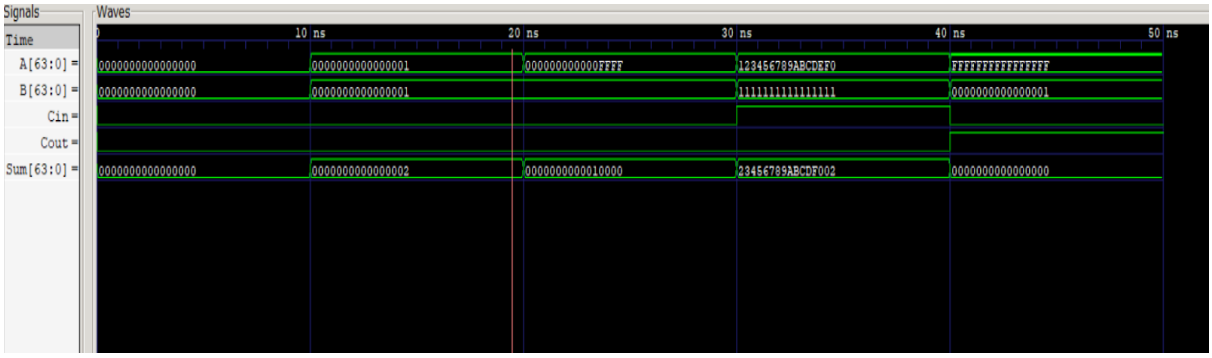

Figure 1- Implementation of 64-bit Adder using Ripple Carry Adder

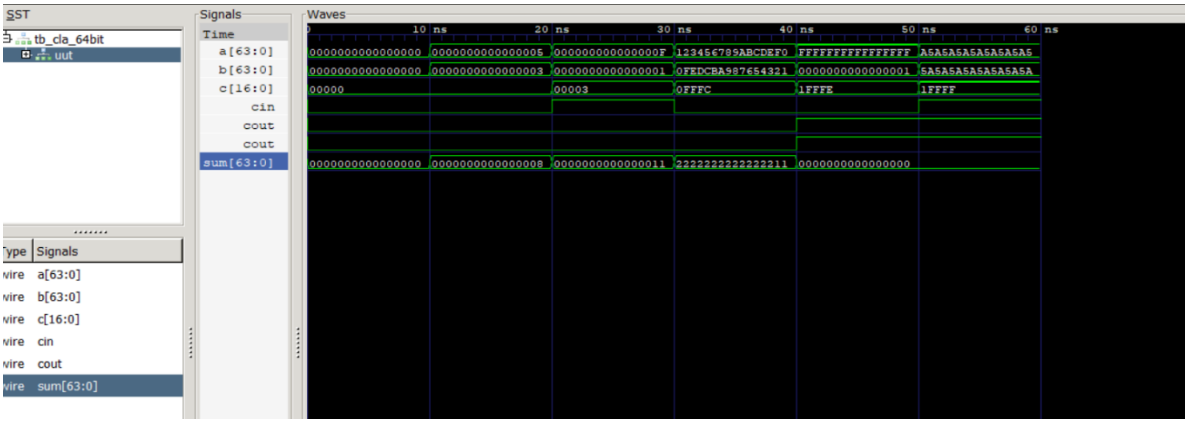

Figure 2 - Implementation of 64-bit Adder using Carry Look Ahead Adder

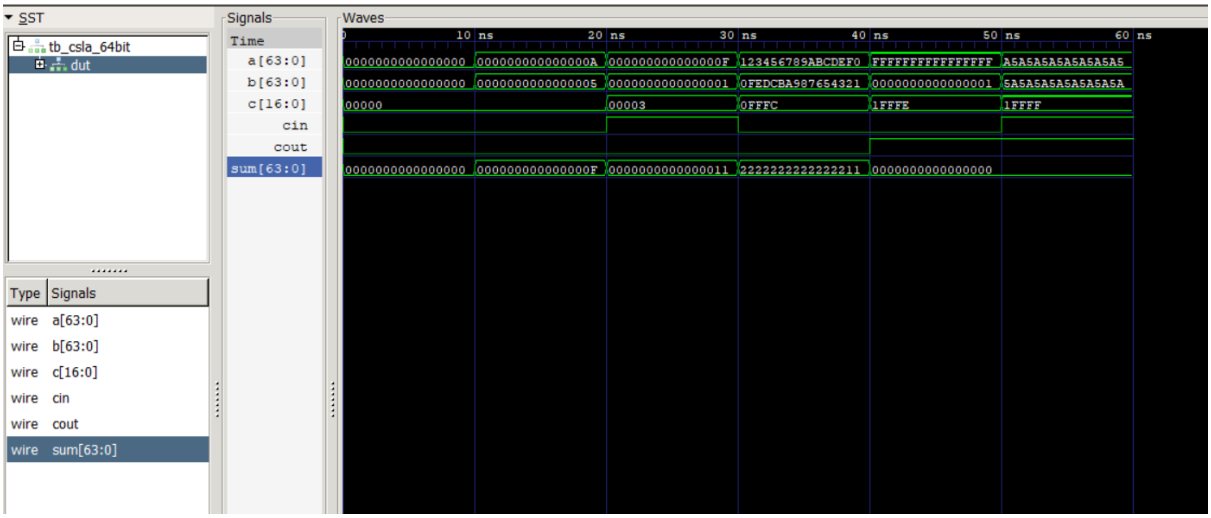

Figure 3 -Implementation of 64-bit Adder using Carry Select Adder

Conclusion

Describe the important results related to the performance aspects of the different 64-bit adder architectures implemented by different Full Adder architectures. Repeat the important trade-offs experimented with and give suggestions related to specific application areas such as high-speed DSP, Low Power Embedded Systems, and General Purpose CPUs. Also, explain the possible future research areas, such as investigating the latest advances in new adder architectures such as Parallel Prefix Adders, clock gating, or studying Fault-Tolerant Adder architectures, and so forth.

References

Brent, R. P., & Kung, H. T. (1982). A regular layout for parallel adders. *IEEE Transactions on Computers, C-31*(3), 260–264.

Bui, H. T., Wang, Y., & Jiang, Y. (2002). Design and analysis of low-power full adders using transmission gate logic. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 49*(1), 25–30.

Chang, C. H., Gu, J., & Zhang, M. (2005). A review of 0.18-µm full adder performances for tree-structured arithmetic circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 13*(6), 686–695.

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.

Lamani, D. S., & Aradhya, H. V. R. (2023). Design of low-power 64-bit hybrid full adder. *International Journal for Research in Applied Science & Engineering Technology, 11*(5), 189–195.

Patil, V., Kumar, R., & Rao, S. (2025). Energy-efficient high-performance 64-bit ALU using various full adders. *VLSI Journal, 12*(3), 145–152.

Rashmi, B. K., Suma, M. N., & Prasad, K. R. (2022). Performance analysis of different 64-bit adder architectures. *i-Manager's Journal on Electronics Engineering, 14*(3), 21–29.

Sahu, M., Patel, A., & Verma, S. (2024). Novel design approach of 64-bit full adder using Sky130 PDK. *International Journal of Electronics and Electrical Engineering Research, 11*(9), 55–62.

Vyshnavi, G., & Krishna, P. V. (2023). Design and analysis of 64-bit adders using different logic families. *International Research Journal of Engineering and Technology (IRJET), 10*(7), 1123–1128.

Zhang, Y., Wang, H., & Liu, X. (2024). Efficient CMOS full adder design for high-speed arithmetic circuits. *Microelectronics Journal, 142*, 105879.