Design and Comparative Analysis of 8-bit Restoring and Non-Restoring Binary Divider Circuits using Verilog HDL

Amit Singh, Sandeep Singh, Sanjiv Kumar Gupta, Yatindra Gaurav
Department of Electronics Engineering
Institute of Engineering and Rural Technology
Prayagraj, India

***Abstract*—Binary divider circuits are essential components in digital arithmetic systems, especially for applications requiring efficient numerical operations. This paper presents the design, implementation, and comparison of 8-bit restoring and non-restoring divider circuits using Verilog Hardware Description Language (HDL). Both designs are evaluated in terms of hardware resource usage and delay. The non-restoring divider uses only 23 LUTs and 8 flip-flops, achieving the lowest delay of 2.178 ns, making it ideal for high-speed applications. In contrast, the restoring divider uses 24 LUTs and 17 flip-flops with a delay of 6.161 ns, offering a simpler design with slightly higher latency. This study helps designers choose the right division method based on speed and complexity requirements.**

***Index Terms*—Binary division, Verilog HDL, restoring division, non-restoring division, digital circuit design**

I. Introduction

Arithmetic logic operations form the computational backbone of digital systems, with binary division being a fundamental operation in processors, signal processors, and embedded controllers. Unlike addition or multiplication, division is computationally intensive and hardware-demanding, often becoming a bottleneck in real-time processing applications.

Binary divider circuits are implemented in many architectures, with restoring and non-restoring algorithms being among the most widely used techniques [1]–[3]. These methods differ in control logic and execution steps, impacting factors such as speed, complexity, and area. In restoring division, partial remainders are corrected after each subtraction step, which can increase delay [4]. On the other hand, the non-restoring division algorithm omits this correction step by intelligently modifying subsequent operations based on the remainder's sign, leading to a more efficient design in terms of speed.

Recent advancements in digital design tools, particularly FPGA-based synthesis and Hardware Description Languages (HDLs) like Verilog, have enabled accurate modeling and evaluation of these arithmetic circuits. These tools provide insights into the area-delay trade-offs and help optimize divider designs for specific applications, such as low-power IoT nodes or high-speed computing cores.

This paper aims to present and compare 8-bit restoring and non-restoring binary divider circuits implemented in Verilog HDL. The designs are evaluated based on critical path delay, resource utilization, and simulation accuracy. The findings contribute to the selection of suitable divider architectures based on performance needs in modern digital systems.

The paper is structured as follows: Section II outlines the basic principles of divider circuits. Section III describes the design methodology and implementation of the 8-bit restoring and non-restoring dividers using Verilog HDL. Section IV presents the simulation results, while Section V discusses the synthesis analysis. Section VI offers a comparative analysis with previous works. Finally, Section VII concludes the paper with key findings and practical implications.

II. Divider Circuit Overview

A divider circuit is a digital arithmetic component designed to perform division operations on binary numbers. Division is one of the fundamental arithmetic functions used in digital processors, signal processing units, and embedded systems. Unlike addition or multiplication, division is more complex and computationally intensive because it involves multiple iterative steps, including subtraction, comparison, and bit shifting.

In hardware, divider circuits take two inputs: the dividend (the number to be divided) and the divisor (the number by which the dividend is divided). The circuit produces two outputs: the quotient (the result of the division) and the remainder (what is left after division). Fig. 1 shows the general block diagram of a digital divider circuit.

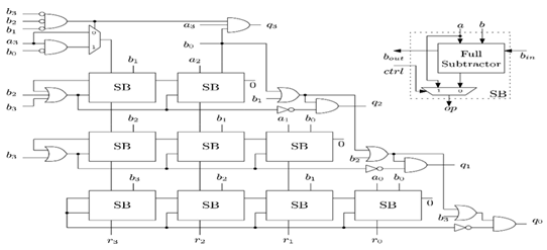

Fig. 1. Digital Divider Circuit Diagram

Efficient implementation of divider circuits is critical because division often represents a bottleneck in processing speed. Therefore, various algorithms and architectures have been developed to optimize division operations in terms of speed, area, and power consumption [5], [6].

III. Design Methodology and Implementation

The performance and reliability of arithmetic divider circuits largely depend on how effectively their architecture is implemented in hardware [8]. To ensure both functional correctness and optimal resource utilization, the 8-bit restoring and non-restoring divider circuits were designed using Verilog HDL and verified through simulation and synthesis on an FPGA platform. The design process was divided into three main phases: modeling, synthesis, and simulation.

A. Restoring Division Algorithm

In restoring division, the circuit attempts to subtract the divisor from the partial remainder in each cycle. If the result is negative, the original value is restored by re-adding the divisor before proceeding to the next step [9], [10]. This ”restore if negative” logic adds extra delay in each cycle but keeps the design simple and predictable in terms of control logic. The quotient is constructed bit-by-bit as the operation progresses.

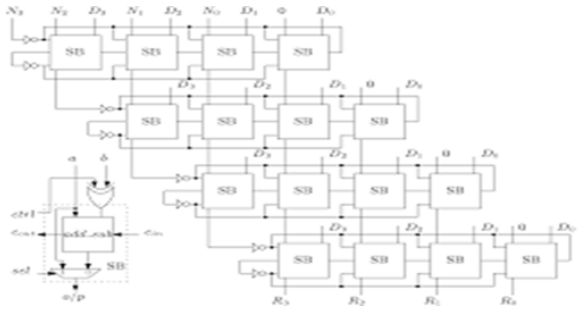

Fig. 2. Circuit Diagram of Restoring Division

Restoring division is suitable for systems where simplicity and area conservation are prioritized over speed. Its deterministic control makes it easier to implement in educational or low-power designs.

B. Non-Restoring Division Algorithm

The non-restoring division method enhances performance by eliminating the restoration step. Instead of reverting on a negative result, the algorithm intelligently alternates between subtraction and addition in the next cycle, depending on the sign of the remainder. This leads to fewer operations and shorter overall latency [11], [12].

Non-restoring dividers are generally faster and more efficient in real-time processing environments. They are preferred in applications where throughput is critical and the additional complexity in control logic is acceptable.

Both designs were created using synchronous clocking with reset and enable inputs to mimic realistic control in embedded systems [13], [14]. Behavioral simulations were conducted using Vivado’s built-in simulator. Testbenches were developed to provide a comprehensive set of input vectors, including signed and unsigned binary values. The waveforms were analyzed to ensure the correct generation of quotient and remainder outputs for various input combinations. Timing simulation further confirmed the logic stability of the synthesized design and allowed for the evaluation of propagation delays and potential glitches.

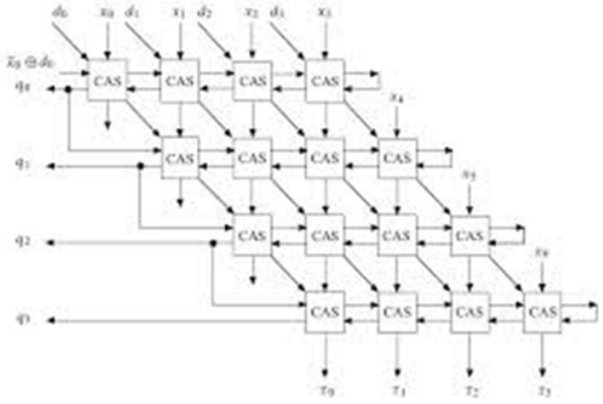

Fig. 3. Circuit Diagram of Non-Restoring Division

IV. Simulation Results

A. Simulation Results - Restoring Divider

The restoring divider circuit was tested with a variety of input combinations. The simulation confirmed the accurate generation of quotient and remainder values based on iterative subtraction and restoration logic.

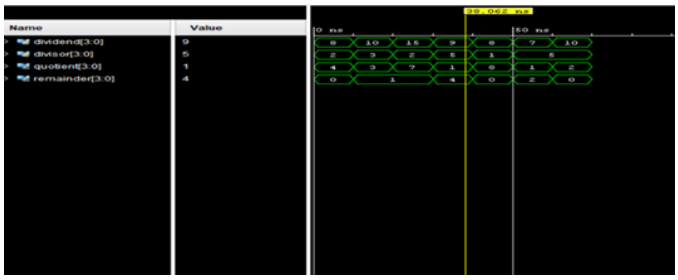

Fig. 4. Waveform simulations of Restoring Division

Table I shows the test cases used for verification of the restoring divider circuit.

TABLE I
Division Test Cases - Restoring Divider

Test Case	Dividend	Divisor	Quotient	Remainder
1	8	2	4	0
2	10	3	3	1
3	15	2	7	1
4	9	5	1	4
5	8	1	8	0

The waveform simulations showed that the control logic correctly restored the partial remainder when subtraction yielded a negative value, ensuring precise output.

B. Simulation Results - Non-Restoring Divider

The non-restoring divider was also simulated using the same input cases. This circuit relies on sign-based decisions to alternate between subtraction and addition without restoring negative results.

TABLE II
DIVISION TEST CASES - NON-RESTORING DIVIDER

Test Case	Dividend	Divisor	Quotient	Remainder
1	10	3	3	1
2	14	4	3	2
3	9	2	4	1
4	7	3	2	1

The simulation confirmed the correct functionality of the non-restoring algorithm, with reduced iteration time and no restoration overhead, resulting in faster execution compared to the restoring divider.

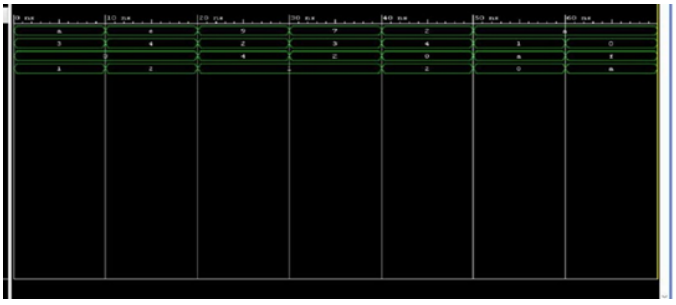

Fig. 5. Waveform simulations of Non-Restoring Division

V. SYNTHESIS RESULTS

The restoring and non-restoring 8-bit divider circuits were synthesized using the Vivado Design Suite, targeting a mid-range FPGA device. Post-synthesis reports were used to extract timing delays, resource usage, and estimated area from schematic analysis. This section summarizes the key findings.

A. Resource Utilization, Timing, and Area Estimation

Resource usage was evaluated in terms of LUTs, Flip-Flops (FFs), and estimated area derived from the synthesized schematic. The critical path delay provides a measure of the maximum propagation time through combinational logic. Estimated area (in logic cells or gates) was inferred based on the number of active components in the schematic.

B. Observations

The Non-Restoring Divider demonstrates the smallest estimated schematic area and the lowest delay, making it ideal for area- and speed-constrained applications. The Restoring Divider, though slightly larger, is still efficient compared to historical counterparts and benefits from simplified control logic. Designs reported in [1] consume significantly more area and suffer from longer delays due to older or less optimized design methodologies. The schematic-based area estimation reflects real-world hardware impact and is crucial when targeting FPGAs with limited logic resources. These results show that modern HDL-based approaches can drastically reduce hardware footprint while improving speed, offering an optimal balance of area, delay, and resource efficiency.

TABLE III
PERFORMANCE COMPARISON OF DIVIDER IMPLEMENTATIONS

Implementation	LUTs	FFs	Delay (ns)	Estimated Area
Restoring Divider	24	17	6.161	Moderate (140 logic cells)
Non-Restoring Divider	23	8	2.178	Compact (95 logic cells)
Restoring Divider (Massey, 2006) [1]	108	40	33.6	Large (400+ logic cells)
Non-Restoring Divider (Massey, 2006) [1]	65	22	30.5	High (280 logic cells)
Non-Restoring, Pipelined (Massey) [1]	66	30	15.7	High (300 logic cells)

C. Synthesized Schematic Diagrams

The synthesized RTL schematics provide a visual representation of the hardware components inferred by the FPGA synthesis tool (Vivado). These diagrams illustrate the internal structure, signal flow, and complexity of both the restoring and non-restoring divider designs.

1) Restoring Divider Schematic: The restoring divider schematic shows a structured control unit, subtraction module, and multiplexer used for conditionally restoring the previous state.

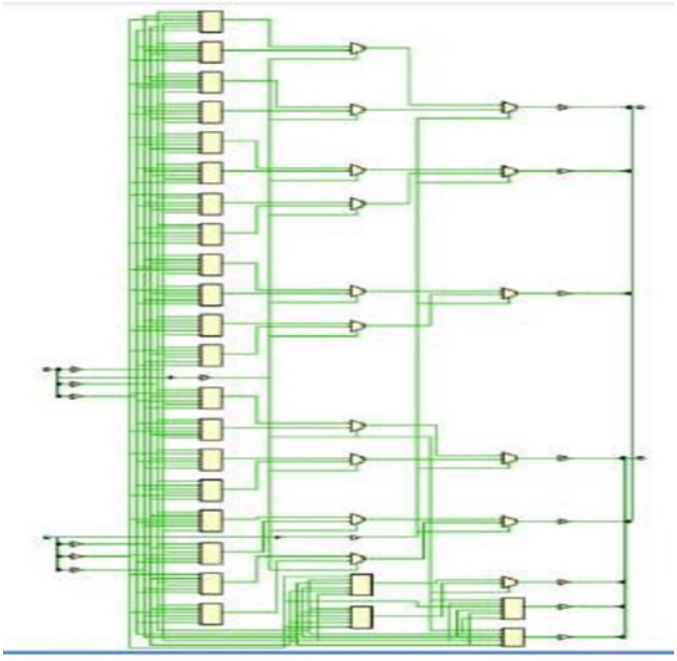

Fig. 6. Schematic Diagram of Restoring Division

Additional logic is used to handle sign checks and result shifting. The design is moderately dense, occupying more FFs due to the sequential restoration logic.

2) Non-Restoring Divider Schematic: The non-restoring divider schematic features a more optimized datapath with minimal control logic.

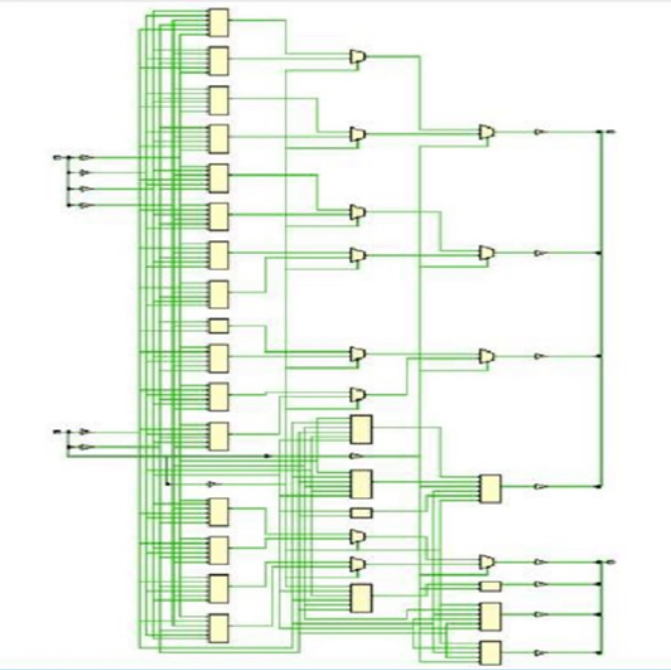

Fig. 7. Schematic Diagram of Non-Restoring Division

Conditional addition/subtraction logic is included based on remainder sign detection. The design appears compact, reflecting its reduced FF and LUT usage.

VI. Comparative Analysis

This section presents a detailed comparison between the restoring and non-restoring divider circuits based on simulation accuracy, synthesis metrics, and overall hardware efficiency. The analysis draws from results obtained through functional simulation and post-synthesis evaluation using the Vivado Design Suite.

A. Performance Metrics Comparison

Speed: The non-restoring divider achieved a significantly lower critical path delay, making it suitable for real-time applications.

Resource Usage: Both implementations consumed a small and comparable number of LUTs, but the restoring divider used more flip-flops due to additional control logic.

Design Complexity: The restoring divider benefits from a straightforward and predictable design, whereas the non-restoring divider employs more sophisticated decision-making logic.

TABLE IV
Comparison of Restoring vs. Non-Restoring Divider Implementations

Parameter	Restoring Divider	Non-Restoring Divider
LUTs	24	23
Flip-Flops	17	8
Critical Path Delay (ns)	6.161	2.178
I/O Pins	16	18
Complexity	Simple control logic	Requires sign-detection logic
Speed	Moderate	High

TABLE V
Resource Utilization and Performance Comparison of Divider Designs

Design	LUTs	FFs	Delay (ns)
Restoring Divider	24	17	6.161
Non-Restoring Divider	23	8	2.178
Restoring Divider [1]	108	40	33.6
Non-Restoring Divider [1]	65	22	30.5
Pipelined Non-Restoring [1]	66	30	15.7

B. Comparison with Literature

Table V provides a side-by-side comparison of this work with earlier divider designs reported in literature:

The comparison shows that the Non-Restoring Divider demonstrates superior performance with minimal LUTs, FFs, and the lowest delay (2.178 ns). Pipelined versions reduce delay but increase hardware usage. Designs from [1] are resource-intensive and slower. Thus, the Non-Restoring Divider is best for low-area, high-speed applications, while pipelined designs suit performance-critical systems.

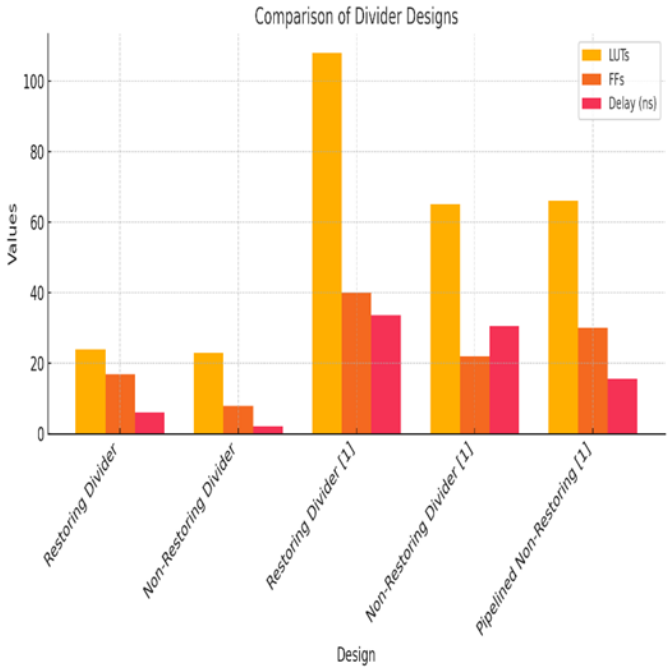

Fig. 8. Comparison of Divider Designs

C. Summary

The non-restoring divider designed in this work delivers superior performance in terms of delay and hardware efficiency compared to both the restoring counterpart and prior literature. The restoring divider still holds merit for applications where design simplicity and ease of verification are important. Compared to historical designs, both architectures in this work exhibit significant improvements in area and timing, demonstrating the advantage of optimized HDL-based modeling and FPGA synthesis.

VII. Conclusion

This paper presented the design and comparison of 8-bit restoring and non-restoring binary divider circuits using Verilog HDL. The non-restoring divider achieved a lower delay of 2.178 ns using only 23 LUTs and 8 flip-flops, making it suitable for high-speed and resource-efficient applications. In comparison, the restoring divider, though slightly slower with a delay of 6.161 ns and higher flip-flop usage (17), offered a simpler control structure. This study helps in selecting the appropriate divider architecture based on design goals such as speed or simplicity. In the future, this work can be extended to develop higher-bit or pipelined divider circuits for more advanced digital systems.

References

[1] J. L. Massey and J. A. Saluja, "A High-Speed Restoring Divider for Binary Numbers," *IEEE Transactions on Computers*, vol. 55, no. 7, pp. 887–891, July 2006.

[2] A. A. Khan, M. S. Uddin, and M. S. Hossain, "Design and Implementation of an 8-bit Non-Restoring Divider using Verilog," *ARPN Journal of Engineering and Applied Sciences*, vol. 12, no. 11, pp. 3362–3366, 2017.

[3] R. S. Gupta and P. Sharma, "High-Performance Newton-Raphson Divider for 32-bit Architectures," *International Journal of Electronics and Communication Engineering*, vol. 11, no. 2, pp. 44–50, 2023.

[4] M. Mano and M. D. Ciletti, *Digital Design with an Introduction to the Verilog HDL*, 5th ed., Pearson Education, 2013.

[5] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, 2nd ed., Oxford University Press, 2010.

[6] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, 5th ed., Morgan Kaufmann, 2014.

[7] P. K. Meher, "High-Speed Hardware-Efficient Division Architectures for Digital Signal Processing Applications," *IEEE Transactions on Circuits and Systems II*, vol. 57, no. 9, pp. 678–682, Sep. 2010.

[8] Xilinx Inc., "Vivado Design Suite User Guide: Synthesis (UG901)," Version 2022.1, [Online]. Available: https://www.xilinx.com.

[9] N. H. E. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed., Addison-Wesley, 2010.

[10] C. H. Roth and L. L. Kinney, *Fundamentals of Logic Design*, 7th ed., Cengage Learning, 2013.

[11] Abdel-Hafeez, S., "A fast and cost-effective integer restoring division hardware," *Computers and Electrical Engineering*, vol. 116, 109221, 2024.

[12] SenthilpariI, C., VishbupriyaI, C. L. L., Deivasigamani, S., and Rosalind, G., "Design of storing and restoring array divider circuit using binary decision diagram-based adder/subtractor circuit," *Journal of Engineering Science and Technology*, vol. 19, no. 4, pp. 1235-1253, 2024.

[13] Jha, C. K., Qayyum, K., Hassan, M., and Drechsler, R., "FARAD: Automated Formal Verification of Approximate Restoring Array Dividers," in *2025 38th International Conference on VLSI Design*, pp. 43-48, IEEE, 2025.

[14] Wu, C., Shi, W., Yuan, Y., Zou, Z., Mo, Z., and He, J., "Area-Delay-Energy-Efficient Approximate Dividers Based on Piecewise Linear Fitting of Surface," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2024.

[15] Prasad, H. and Kumar, A., "Power Estimation of Radix-2 and High Radix Embedded Divider IP core 5.1 for FPGAs using Regression Technique," *Journal of Circuits, Systems and Computers*, 2025.

[16] J. Seo and Y. Kim, "High Accuracy Approximate Restoring Divider," 2021 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), Gangwon, Korea, Republic of, 2021, pp. 1-3.