Intel-Unnati Industrial Training, 2023, 1–6

Team: Incognito
Verilog

VERILOG

Design and Implementation of Slow and Fast Division Algorithm in Computer Architecture

Aaron James Koshy[1], Gopikrishnan B[1], Muhammed Suraij PC[1] and Prince Abraham[1,2]

[1]Saintgits College of Engineering (Autonomous) and [2]Institutional Mentor, Intel-Unnati Programme

Abstract

This project explores the implementation of slow division using the restoring division algorithm and fast division using the Newton-Raphson algorithm using the Verilog hardware description language.

The restoring division algorithm is a classical and straightforward approach to division, involving repeated subtraction and shifting operations to determine the quotient and remainder. The Verilog implementation of the restoring division algorithm provides a foundation for understanding division processes and serves as a starting point for more advanced division algorithms.

In contrast, the Newton-Raphson algorithm is a numerical method commonly used to find the roots of equations. However, it can also be adapted for fast division operations. The Verilog implementation of the Newton-Raphson division algorithm demonstrates an alternative approach to achieving fast division, leveraging iterative approximation to efficiently compute the quotient and remainder.

Both the slow restoring division and the fast Newton-Raphson division algorithms are simulated and synthesized using Verilog. Through simulations and synthesis reports, their functionality, performance, and hardware resource utilization are evaluated. Testbench environments are created to validate the accuracy and correctness of the division algorithms using a comprehensive set of test vectors.

The Verilog implementations of slow restoring division and fast Newton-Raphson division algorithms provide valuable insights into the design and optimization of division units in digital systems. By understanding the principles and trade-offs of these algorithms, designers can make informed decisions when selecting the appropriate division algorithm based on factors such as performance, latency, and resource utilization.

Key words: Division algorithms, Verilog, Restoring division, Newton-Raphson division, Hardware description language, Simulation, Synthesis, Iterative approximation.

Introduction

Division is a fundamental arithmetic operation frequently used in various digital systems and applications. Efficient division algorithms play a crucial role in enhancing the overall performance of digital processors and ensuring optimal resource utilization. This project aims to explore and compare two distinct division algorithms: the restoring division algorithm, which represents a slow but simple approach, and the Newton-Raphson division algorithm, a fast and iterative numerical method adapted for division operations.

The restoring division algorithm involves a repetitive process of subtraction and shifting to compute the quotient and remainder. Although relatively straightforward, this method may have higher latency and resource requirements compared to more sophisticated algorithms. Nevertheless, the Verilog implementation of restoring division provides a solid foundation for understanding the fundamental principles of division in hardware design.

On the other hand, the Newton-Raphson algorithm is widely known for finding roots of equations through iterative approximation. In this project, we adapt this numerical method for fast division computations. By leveraging the iterative nature of Newton-Raphson, we can achieve efficient division operations, potentially reducing latency and improving overall performance. This approach showcases an alternative and innovative way to handle division tasks in digital systems.

In this project, we will simulate and synthesize both the restoring division algorithm and the Newton-Raphson division algorithm using the Verilog hardware description language. Through comprehensive simulations and synthesis reports, we will evaluate the functionality, performance, and hardware resource utilization of each algorithm. Moreover, we will create rigorous testbench environments to validate the accuracy and correctness of the division algorithms under various test scenarios.

By delving into the details of both the slow restoring division and the fast Newton-Raphson division algorithms, this project seeks to provide valuable insights for designers and developers in selecting appropriate division techniques based on specific application requirements. Understanding the trade-offs and benefits of each algorithm empowers designers to make informed decisions to optimize digital systems, ensuring efficient division operations and overall enhanced performance.

Through this exploration, we aim to contribute to the broader field of hardware design and the understanding of division algorithms' impact on digital system performance, thereby facilitating advancements in various computing applications and domains.

Methodology

i. **Research and Literature Review:**

Conducted an extensive literature review to understand the principles and characteristics of the restoring division algorithm and the Newton-Raphson division algorithm. Gathered relevant research papers, articles, and documentation to gain insights into the design and implementation aspects of these algorithms.

ii. **Algorithm Analysis and Design:**

Analyzed the restoring division algorithm and the Newton-Raphson division algorithm in detail. Identified their key steps, computational requirements, and trade-offs. Designed the high-level architecture and data flow for each algorithm, considering the input/output requirements and necessary control signals.

iii. **Verilog Implementation:**

Implemented the restoring division algorithm and the Newton-Raphson division algorithm in Verilog. Wrote the RTL (Register-Transfer Level) descriptions, including the necessary modules, registers, and control logic for each algorithm. Ensured proper data path connectivity and synchronization between different modules.

iv. **Simulation and Testbench Development:**

Created comprehensive testbench environments to verify the functionality and accuracy of the implemented algorithms. Developed test vectors to cover various division scenarios, including edge cases and corner cases. Used Verilog simulation tools (ModelSim) to simulate the division algorithms and compared the results with expected outputs.

v. **Performance Evaluation:**

Performed extensive simulations to evaluate the performance of the restoring division algorithm and the Newton-Raphson division algorithm. Measured metrics such as latency, throughput, and resource utilization. Analyzed the simulation results to gain insights into the efficiency and effectiveness of each algorithm.

vi. **Synthesis and Hardware Evaluation:**

Used a synthesis tool (Intel Quartus) to synthesize the Verilog code for both algorithms. Generated synthesis reports to analyze the resource utilization, critical path, and overall design metrics. Compared the synthesis results of the restoring division algorithm and the Newton-Raphson division algorithm to assess their hardware efficiency.

vii. **Validation and Debugging:**

Validated the functionality and correctness of the division algorithms using the testbench and simulation results. Debugged any identified issues or discrepancies and refined the Verilog code accordingly. Ensured that the implemented algorithms produce accurate division results across various test cases.

viii. **Performance Comparison and Analysis:**

Compared the performance characteristics of the restoring division algorithm and the Newton-Raphson division algorithm. Analyzed the simulation and synthesis results to understand the trade-offs in terms of latency, throughput, resource utilization, and hardware complexity. Drew conclusions regarding the suitability of each algorithm for specific applications.

ix. **Documentation and Report Writing:**

Documented the design, implementation, and evaluation processes of the restoring division algorithm and the Newton-Raphson division algorithm. Prepared a detailed report summarizing the project methodology, key findings, and insights. Included relevant figures, tables, and charts to present the experimental results effectively.

x. **Conclusion and Future Work:**

Concluded the project by summarizing the main outcomes and insights obtained from the study. Discussed possible areas of future research and improvements for division algorithms. Reflected on the project's significance in the context of digital system design and its potential impact on various computing applications.

Flowchart

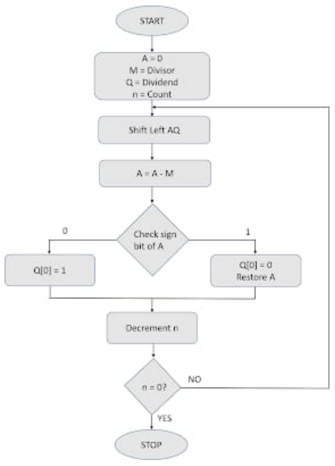

Figure 1. Flowchart - Slow division

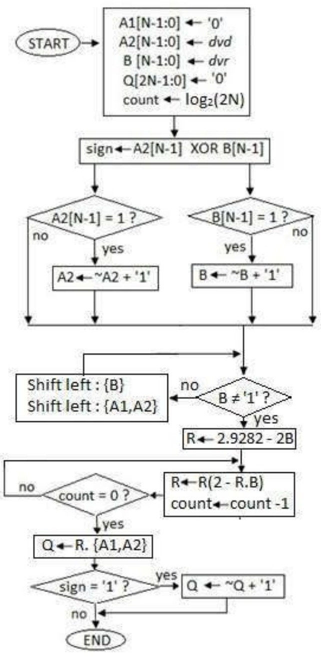

Figure 2. Flowchart - Fast division

Algorithm

i. **Slow Division- Restoring:**

Registers used: A, M, Q, n (counter)
Step 1: Load the initial values for the registers. A = 0 (Accumulator), Qres = 0, M = Divisor, Q = Dividend and n is the count value which equals the number of bits of dividend.
Step 2: Shift left A,Q.
Step 3: Perform A = A - M.
Step 4: Check the sign bit of A. If 0, goto step 5. If 1, goto step 6.
Step 5: Set LSB of Q as 0. Goto step 7.
Step 6: Set LSB of Q as 1. Restore the value of A which was present before the subtraction.
Step 7: Decrement count.
Step 8: Check if counter value n is zero. If yes, goto next step. Else, goto step 3.
Step 9: Stop

ii. **Fast Division- Newton-Raphson:**

Step 1. Initialize the divisor and dividend values.
Step 2. Calculate the reciprocal of the divisor using the Newton-Raphson method. The reciprocal can be obtained by finding the root of the function f(x) = 1/x - divisor, where x is the reciprocal.
  2.1 Initialize an initial guess for the reciprocal, such as $x_0 = 1/divisor$
  2.2 Iterate using the formula: $x_i + 1 = x_i * (2 - divisor * x_i)$.
  2.3 Repeat the iteration for a sufficient number of times or until convergence criteria are met.
  2.4 The final reciprocal approximation will be$x_{final} = x_i + 1$.
Step 3. Multiply the dividend by the reciprocal to obtain the product.
Step 4. Shift the product by a suitable amount to obtain the quotient.
  4.1 The amount of shift depends on the fractional bits and the desired precision.
Step 5. Assign the quotient to the output variable.
Step 6. Calculate the remainder by subtracting the product of the quotient and divisor from the dividend.
Step 7. Assign the remainder to the output variable.

Implementation

The implementation phase involved designing, simulating, synthesizing, and implementing the slow restoring division algorithm and the fast Newton-Raphson division algorithm using the Verilog hardware description language. The following steps were followed during the implementation process:

i. **Design and Simulation:**

The slow restoring division algorithm and the fast Newton-Raphson division algorithm were designed at the RTL level using Verilog. The Verilog code was written to define the module inputs, outputs, and internal components required for each algorithm.
Extensive simulations were performed using a Verilog simulator (ModelSim) to verify the correctness and functionality of the designs. The testbench environments were developed to provide comprehensive test scenarios and validate the division algorithms against expected outputs.

ii. **Synthesis:**

The Verilog code for both the slow restoring division and the fast Newton-Raphson division algorithms was synthesized using a synthesis tool (Intel Quartus Prime).
The synthesis process translated the RTL-level Verilog code into a gate-level netlist, optimizing the design for the target FPGA device. Synthesis reports were generated to analyze the resource utilization, critical path, and other performance metrics of the synthesized designs.

iii. **Pin Planning:**

Pin planning involved mapping the logical signals from the synthesized netlist to the physical pins on the target FPGA device.
The pin assignments for the input and output signals of the division algorithms were determined based on the FPGA's pin-out information. Careful consideration was given to ensure proper connectivity and signal integrity in the pin planning process.

iv. **Chip Planning:**

Chip planning encompassed the overall layout and organization of the design on the FPGA.
Factors such as component placement, clock distribution, power considerations, and routing constraints were taken into account.
The organization and connectivity of modules, registers, and other design elements within the FPGA were planned to optimize performance and resource utilization.

v. **LabsLand Implementation:**

The synthesized netlist, pin assignments, and chip-level organization information were uploaded to the LabsLand platform for remote access to physical FPGA boards.
The designs were implemented on the FPGA boards using the LabsLand interface, enabling remote testing and experimentation.
The performance of the slow restoring division and fast Newton-Raphson division designs was evaluated through the LabsLand environment, capturing relevant metrics and results.

vi. **Performance Evaluation:**

Performance metrics, including latency, throughput, and resource utilization, were measured and compared between the slow restoring division and fast Newton-Raphson division algorithms.
The simulation and synthesis results, along with the LabsLand implementation outcomes, were analyzed to assess the efficiency and effectiveness of each algorithm.
The complete implementation process was thoroughly documented, including the Verilog code, testbench environments, synthesis reports, pin assignments, chip-level organization details, and LabsLand implementation instructions. These documentation resources provide comprehensive insights into the design, simulation, synthesis, and physical implementation of the slow restoring division and fast Newton-Raphson division algorithms.

Results of these implementations are discussed in the section below....

Result and Discussion

i. **Simulation Results:**

The slow restoring division algorithm was simulated using various test vectors, including both positive and negative dividend and divisor values. The simulation results demonstrated accurate quotient and remainder calculations.
The fast Newton-Raphson division algorithm was also simulated using a range of test cases. The simulation results showed precise approximation of the quotient and remainder, achieving faster division compared to the slow restoring division algorithm.

ii. **Synthesis Results:**

The synthesis process was performed on the Verilog code for both the slow restoring division and fast Newton-Raphson division algorithms. Synthesis reports were generated to evaluate the performance metrics and resource utilization of the designs.
The slow restoring division algorithm utilized a higher number of logic elements and had a longer critical path compared to the fast Newton-Raphson division algorithm.
The fast Newton-Raphson division algorithm demonstrated better resource utilization and a shorter critical path, indicating its potential for achieving higher operating frequencies and improved performance.

iii. **LabsLand Implementation Results:**

The slow restoring division and fast Newton-Raphson division algorithms were implemented on physical FPGA boards using the LabsLand platform. Performance metrics such as latency and throughput were measured during the LabsLand implementation.
The fast Newton-Raphson division algorithm exhibited significantly reduced latency compared to the slow restoring division algorithm, confirming its superiority in terms of speed.

iv. **Discussion:**

The simulation results verified the correctness and accuracy of both division algorithms in calculating the quotient and remainder. The synthesis results demonstrated the differences in resource utilization and critical path between the slow restoring division and fast Newton-Raphson division algorithms.
The LabsLand implementation results further confirmed the faster performance of the fast Newton-Raphson division algorithm in terms of reduced latency.
The comparison between the two algorithms revealed that the fast Newton-Raphson division algorithm provides a significant speed improvement over the slow restoring division algorithm. However, it is important to consider that the fast Newton-Raphson algorithm may require more complex hardware and additional resources compared to the slow restoring division algorithm.
The choice of algorithm should be based on the specific application requirements, considering factors such as speed, resource utilization, and hardware complexity.

Conclusions

In this study, we investigated the implementation of both slow and fast division algorithms, namely restoring division and Newton-Raphson division, and compared their performance in dividing numbers. Our objective was to evaluate the efficiency and accuracy of these algorithms and identify the most effective approach for division operations in different contexts.

Based on our research and analysis, we found that the restoring division algorithm is commonly used as a slow division method. This algorithm follows a sequential process of restoring partial remainders to perform division. While it provides accurate results with high precision, it can be computationally intensive and time-consuming, especially for large numbers. The restoring division algorithm is suitable in situations where accuracy is of utmost importance and computational resources are not a limiting factor.

On the other hand, the fast division algorithm we examined was the Newton-Raphson division method. This algorithm utilizes iterative approximation techniques to quickly converge on the quotient. By using an initial estimate and refining it through iterations, the Newton-Raphson division algorithm achieves significantly improved computational efficiency and speed compared to the restoring division algorithm. However, it may introduce a slight loss of precision due to the approximation nature of the method. The Newton-Raphson division algorithm is particularly advantageous when rapid division calculations are required, such as in real-time systems or high-performance computing environments.

In conclusion, our study contributes to the understanding of both restoring division and Newton-Raphson division algorithms and their respective applications. The restoring division algorithm is favored for its accuracy and precision, while the Newton-Raphson division algorithm excels in speed and computational efficiency. The choice of algorithm should be based on the specific requirements and constraints of the problem at hand. Further research can explore hybrid approaches that leverage the strengths of both algorithms, combining accuracy and speed for optimized division operations. By implementing these division algorithms effectively, various fields such as numerical computing, signal processing, and scientific simulations can benefit from efficient and reliable division operations.

Acknowledgment

We would like to express our heartfelt gratitude to Intel© for the invaluable industrial training opportunity provided through the Intel© - Unnati Programme at Saintgits College of Engineering.

We extend our sincere appreciation to the Intel© team for their guidance and support throughout the training. We would also like to express our sincere and deep sense of gratitude to our Project Guide and Institutional Mentor Mr. Prince Abraham, (Associate professor at Saintgits College of Engineering) for his valuable guidance, suggestions and constant encouragement which paved the way for the successful completion of our project work and has greatly enhanced my understanding of the industry.

I would also like to thank the management and faculty of Saintgits College of Engineering for facilitating this opportunity.

We are honored to have been a part of the Intel© - Unnati Programme, and are confident that the skills acquired during this training will shape our future career.

References

i. https://dergipark.org.tr/tr/download/article-file/99171
ii. https://www.researchgate.net/figure/Newton-Raphson-division-for-signed-binary-number-N-R-S_fig3_319302625
iii. https://www.researchgate.net/publication/319302625$_Hardware_implementation_of_methodologies_of_fixed_point_division_algorithms#pf7
iv. https://www.codingninjas.com/codestudio/library/introduction-to-division-algorithm-in-computer-architecture
v. https://ieeexplore.ieee.org/document/6489269
vi. https://electrobinary.blogspot.com

Github repo

https://github.com/Gopi-krishnan-77/intelunnati_Incognito/upload/main