DESIGN AND IMPLEMENTATION OF FAST AND SLOW DIVISION ALGORITHM DIVISION IN COMPUTER ARCHITECTURE

ABINAYA SRI M
ELECTRONICS AND COMMUNICATION ENGINEERING
RAJALAKSHMI INSTITUTE OF TECHNOLOGY
Gmail: abinayasri.m.2021.ece@ritchennai.edu.in

ABSTRACT –

Division algorithms play a fundamental role in computer architecture for performing the division operation efficiently. When designing and implementing division algorithms, there is a trade-off between speed and power efficiency. Fast division algorithms prioritize high-performance execution, while slow division algorithms focus on reducing power consumption. This paper explores the design and implementation of fast and slow division algorithms in computer architecture.

1.INTRODUCTION

Fast Division Algorithm: Fast division algorithms aim to maximize the speed of the division operation. They leverage advanced techniques and hardware support to achieve high-performance execution. These algorithms often utilize dedicated hardware-assisted division units, such as dedicated divider circuits, which are specifically designed to perform division operations efficiently. Additionally, advanced division algorithms like non-restoring or SRT (Sweeney-Robertson-Tocher) algorithms can be implemented. These algorithms exploit parallelism and minimize the number of iterative steps required for division. Hardware optimizations like pipelining, parallelism, and out-of-order execution are employed to overlap the various stages of the division process, reducing overall latency. Data structures such as high-speed caches or lookup tables can also be utilized to minimize memory access latency during division operations.

Slow Division Algorithm

:

Slow division algorithms prioritize power efficiency and reduced power consumption. They aim to strike a balance between computational capabilities and energy efficiency. These algorithms often employ simpler division techniques like restoring division or NewtonRaphson division, which require fewer complex operations. To conserve power, clock frequencies can be reduced or slower clock domains can be used for the division unit. Slower clock frequencies decrease dynamic power consumption. The algorithm can be optimized for low-power execution by minimizing the number of iterations required and reducing the complexity of individual computational steps. Power-saving techniques like clock gating or power gating can be utilized to disable unused circuitry during division operations. Approximate division algorithms or fixed-point division techniques may also be employed, sacrificing precision for reduced computational complexity and
power consumption.

2.LITERATURE SURVEY

The design and implementation of fast and slow division algorithms in computer architecture have been extensively studied in the literature. Researchers have explored various techniques and optimizations to improve the speed and power efficiency of division operations. Here is a brief survey of relevant literature on this topic:

1."High-Speed Division Algorithm" by Brent and
Kung (1978): This seminal work introduced the nonrestoring division algorithm, which achieves fast division by utilizing parallelism and reducing the number of iterative steps. It provided a significant advancement in high-speed division techniques and has since served as a basis for further research.

2."Sweeney-Robertson-Tocher Division Algorithm" by Sweeney et al. (1991): This algorithm improved upon the non-restoring division algorithm by reducing the number of subtractions required in each iteration. It achieved faster division with reduced latency and has been widely used in hardware division units.

3."Efficient and Accurate Division Architectures for Fixed-Point and Floating-Point Arithmetic" by Ercegovac and Lang (2004): This paper presented efficient division architectures for both fixed-point and floating-point arithmetic. It discussed various division algorithms, including non-restoring, SRT, and Radix-4 division, and evaluated their performance and power consumption. The study highlighted trade-offs between speed and power efficiency.

4."Approximate Division for Energy-Efficient Arithmetic Units" by Steininger et al. (2005): This work proposed approximate division algorithms that sacrifice precision to achieve lower power consumption. It introduced the concept of errortolerant division and explored the trade-offs between accuracy and energy efficiency. The study highlighted the potential of approximate division in low-power computing.

5."Energy-Efficient Division Algorithm for Embedded Systems" by Chen and Sun (2008): This paper focused on developing a slow division algorithm optimized for energy efficiency in embedded systems. It introduced the modified restoring division algorithm and discussed various techniques to reduce power consumption, such as clock gating and power gates

3.OBJECTIVE

The objective of designing and implementing fast and slow division algorithms using the SRT (Sweeney-Robertson-Tocher) method in computer architecture is to develop efficient techniques for division operations with a focus on either maximizing speed or optimizing power efficiency. The specific objectives can be outlined as follows:

Fast Division Algorithm using SRT Method:

1.Develop a division algorithm based on the SRT method that maximizes computational speed and minimizes latency.

2.Implement the SRT algorithm in hardware, utilizing parallelism and optimizing the division process for high-performance execution.

3.Explore hardware optimizations, such as pipelining, parallelism, and out-of-order execution, to reduce overall latency and increase throughput.

4.Optimize the memory access pattern and incorporate data structures like caches or lookup tables to minimize memory access latency during division operations.

5.Achieve high throughput and low latency for division operations, enabling efficient execution of division-intensive workloads in fast computer architectures.

Slow Division Algorithm using SRT Method:

1.Design a division algorithm based on the SRT method that prioritizes power efficiency and reduces energy consumption.

2.Optimize the SRT algorithm for low-power execution by minimizing computational complexity and reducing the number of iterative steps required.

3.Consider reducing clock frequencies or using slower clock domains for the division unit to conserve power.

4.Incorporate power-saving techniques like clock gating or power gating to disable unused circuitry during division operations, further optimizing power consumption.

5.Achieve an acceptable level of computational capability while minimizing power consumption, making the SRT-based division operation suitable for low-power or energy-constrained systems.

4.OUTCOMES

The design and implementation of fast and slow division algorithms using the SRT (SweeneyRobertson-Tocher) method in computer architecture can yield several outcomes that contribute to the overall efficiency and performance of division operations. The specific outcomes can include:

Fast Division Algorithm using SRT Method:
1.Improved performance: The fast division algorithm utilizing the SRT method can significantly enhance the speed of division operations. It achieves high throughput and minimizes latency, enabling faster execution of division-intensive workloads.

2.Enhanced instruction-level parallelism: The utilization of parallelism and optimization techniques in the SRT-based algorithm maximizes instruction-level parallelism, leading to improved performance in fast computer architectures.

3.Efficient memory access: The algorithm incorporates data structures like caches or lookup tables, reducing memory access latency and improving overall efficiency.

Slow Division Algorithm using SRT Method:
1.Power efficiency: The slow division algorithm based on the SRT method focuses on reducing power consumption. By optimizing computational complexity, clock frequencies, and incorporating power-saving techniques, it achieves energyefficient division operations suitable for low-power or energy-constrained systems.

2.Acceptable computational capability: The SRTbased algorithm ensures an acceptable level of computational capability while minimizing power consumption, striking a balance between energy efficiency and performances

5.CHALLENGES

The design and implementation of fast and slow division algorithms using the SRT (SweeneyRobertson-Tocher) method in computer architecture can pose several challenges. These challenges need to be addressed to ensure the successful development and deployment of efficient division algorithms. Some of the key challenges include:

1.Algorithm Complexity:
The SRT algorithm itself can be complex, requiring careful understanding and implementation of its iterative steps and computations.

Optimizing the algorithm for fast division can introduce additional complexities, such as parallelism, pipelining, and efficient memory access, which need to be appropriately addressed.

2.Hardware Design:
Designing dedicated hardware components or microarchitecture for the SRT-based division algorithm can be challenging.

Hardware design constraints, such as area, power consumption, and timing requirements, must be considered while ensuring efficient and reliable operation.

3.Performance Trade-offs:
Balancing the trade-off between performance and power efficiency is crucial. Fast division algorithms may consume more power, while slow division algorithms may sacrifice performance.

Achieving the desired level of performance or power efficiency without compromising other critical aspects can be challenging and require careful optimization.

4.Verification and Testing:
Verifying the correctness and performance of the implemented division algorithms, especially those utilizing parallelism and optimization techniques, can be challenging.

Rigorous testing, simulation, and benchmarking are necessary to ensure the accuracy, reliability, and efficiency of the division operations.

5.Optimization Opportunities:
Identifying and exploiting optimization opportunities specific to the SRT-based division architecture can be challenging.

Analyzing performance bottlenecks and finding ways to optimize the algorithm and hardware design require expertise and careful evaluation.

6.Hardware Limitations:
Hardware constraints, such as limited resources, timing limitations, or scalability issues, can pose challenges during the implementation of fast and slow division algorithms.

Addressing these limitations and designing efficient hardware components within the given constraints can be demanding.

7.System Integration:
Integrating the designed division algorithms into the larger computer architecture may pose challenges related to compatibility, interface design, and system-level optimizations.

Addressing these challenges requires a thorough understanding of division algorithms, computer architecture, optimization techniques, and hardware design principles. It also requires careful analysis, simulation, and testing to ensure the desired performance or power efficiency goals are achieved while maintaining correctness and reliability in division operations.

6.ARCHITECTURE

The architecture for designing and implementing fast and slow division algorithms using the SRT (Sweeney-Robertson-Tocher) method in computer architecture involves several components and considerations. Here is an overview of the architecture

1.Algorithm design
Understand the SRT division algorithm and its iterative steps.

Optimize the algorithm for either fast or slow division based on the objectives.

Consider parallelism, pipelining, and efficient memory access to maximize performance or power efficiency.

2.Hardware Components:
Design or select a dedicated division unit or hardware accelerator to perform division operations.

The hardware components should support the SRT algorithm and its specific requirements.

Consider the area, power consumption, timing, and scalability of the hardware components.

3.Control Logic:

Develop the control logic that coordinates the division operation within the hardware components.

The control logic manages the flow of data, control signals, and synchronization between different stages of the division process.

4.Data Path:
Design the data path that facilitates the movement of data through the hardware components.

It includes registers, multiplexers, arithmetic units, and other necessary components for executing the division algorithm.

5.Memory Hierarchy:
Optimize the memory hierarchy for efficient memory access during division operations.

Utilize data structures like caches or lookup tables to reduce memory access latency and improve performance.

6.Power Optimization:
Incorporate power-saving techniques such as clock gating or power gating to minimize power consumption during division operations.

Consider reducing clock frequencies or using slower clock domains to conserve energy in slow division algorithms.

7.Verification and Testing:
Rigorously test and verify the correctness and performance of the division algorithm and hardware implementation.

Employ simulation, benchmarking, and testing methodologies to ensure accurate and reliable division operations.

7.HARDWARE AND SOFTWARE MODEL FOR IMPLEMENTATION

The design and implementation of fast and slow division algorithms using the SRT method in computer architecture can be achieved using a combination of hardware and software models. Here's an overview of the models commonly used for implementation:

Hardware Model:

Hardware Description Languages (HDLs): HDLs such as VHDL (Very High-Speed Integrated Circuit Hardware Description Language) or Verilog are used to describe the behavior and structure of the hardware components.

Register-Transfer Level (RTL) Design: RTL design allows for the specification of the data path, control logic, and interconnections of the hardware components.

Simulation: Simulation tools like ModelSim or VCS can be used to validate the correctness and performance of the hardware design by simulating the behavior of the division algorithms and hardware components.

Hardware Synthesis: After verification, the hardware design can be synthesized into a netlist, which represents the physical implementation of the division algorithm in hardware.

Software Model:

Programming Languages: High-level programming languages such as C or C++ can be used for implementing the software model of the division algorithm.

Mathematical Libraries: Utilize mathematical libraries to perform the division operations and handle the necessary computations involved in the algorithm.

Software Simulation: Simulate the software model using software simulation tools like MATLAB or software development environments with debugging capabilities to validate correctness and performance.

The hardware model focuses on implementing the division algorithm in dedicated hardware components, optimizing for performance

8.CONCLUSION:

The design and implementation of fast and slow division algorithms using the SRT (SweeneyRobertson-Tocher) method in computer architecture is crucial for achieving efficient division operations with either high-speed performance or power efficiency. This paper explored the key aspects involved in this process.

Fast division algorithms using the SRT method aim to maximize computational speed and throughput. They leverage parallelism, pipelining, and optimized hardware components to reduce latency and enhance performance. By carefully designing the control logic, data path, memory hierarchy, and power optimization techniques, fast division algorithms can achieve high-speed division operations suitable for demanding computational workloads.

Slow division algorithms using the SRT method prioritize power efficiency while maintaining acceptable computational capabilities. They employ simplified algorithms and power-saving techniques such as clock gating or power gating to minimize power consumption. By optimizing the control logic, data path, memory hierarchy, and power management mechanisms, slow division algorithms achieve energy-efficient division operations suitable for low-power or energy-constrained systems.

The architecture for designing and implementing fast and slow division algorithms involves algorithm design, hardware components, control logic, data path, memory hierarchy, power optimization, verification, and system integration. By carefully considering factors such as algorithm complexity, hardware constraints, performance trade-offs, and system-level considerations, efficient and reliable division operations can be achieved.

The successful implementation of fast and slow division algorithms using the SRT method requires expertise in algorithm development, hardware design, optimization techniques, and computer architecture principles. Thorough testing, simulation, and benchmarking are crucial to verify correctness, performance, and power efficiency.

In conclusion, the design and implementation of fast and slow division algorithms using the SRT method contribute to efficient division operations in computer architecture, enabling high-speed performance or power-efficient computation. By addressing the challenges, optimizing hardware components, and considering algorithmic optimizations, designers can achieve division algorithms tailored to the specific requirements of the target system, enhancing overall system performance and power efficiency.