Slow And Fast Division Algorithm

O Pandithurai

Associate Professor

Department Of Computer Science And Engineering,

Rajalakshmi Institute Of Technology, Chennai, India

Pandics@Ritchennai.Edu.In

G Sai Krishnan

Associate Professor

Department Of Mechanical Engineering,

Rajalakshmi Institute Of Technology, Chennai, India.

Saikrishnan.G@Ritchennai.Edu.In

P Sathish Kumar

Associate Professor

Department Of Computer And Communication Engineering

Rajalakshmi Institute Of Technology, Chennai, India.

Sathishkumar.p@Ritchennai.Edu.In

Akilan K

Artificial Intelligence And Data Science

Rajalakshmi Institute Of Technology, Chennai.

Akilan.K.2021ad@Ritchenni.Edu.In

Balaji K

Artificial Intelligence And Data Science

Rajalakshmi Institute Of Technology, Chennai.

Balaji.K.2021ad@Ritchennai.Edu.In

Vedha Basil Anto S

Computer And Communication Engineering

Rajalakshmi Institute Of Technology, Chennai.

Vedhabasilanto.S.2021cce@Ritchennai.Edu.I

ABSTRACT:

Division Algorithms Play A Crucial Role In Various Computational Tasks, Ranging From Basic Arithmetic Operations To Complex Mathematical Computations. The Performance Of Division Algorithms Is Influenced By Their Efficiency, Speed And Accuracy. This Paper Presents A Comprehensive Study Comparing Slow And Fast Division Algorithms To Evaluate Their Respective Strengths And Weaknesses In Terms Of Computational Efficiency, Speed And Accuracy.

The Research Begins By Introducing The Fundamental Concepts Of Division Algorithms And Their Significance In Computing. The Slow Division Algorithm, Such As The Long Division Method, Is First Examined, Focusing On Its Step-By-Step Process And Inherent Limitations. The Drawbacks Of The Slow Division Algorithm Become Apparent When Faced With Large Numbers Or Time-Sensitive Computations, Where Speed Is A Critical Factor.

Various Fast Division Algorithms, Such As Newton-Raphson, Goldschmidt, And SRT (Sweeney, Robertson, And Tocher) Division, Are Investigated To Address The Limitations Of The Slow Division Algorithm. These Algorithms Leverage Iterative Or Approximation Techniques To Expedite The Division Process. The Paper Explores The Underlying Principles And Mathematical Foundations Behind These Fast-Division Algorithms, Shedding Light On Their Advantages In Terms Of Computational Efficiency.

I.INTRODUCTION:

Slow And Fast Division Algorithms Are Methods Used To Perform Division Operations Efficiently. The Division Is A Fundamental Arithmetic Operation That Involves Finding The Quotient And Remainder When Dividing One Number (The Dividend) By Another (The Divisor). However, Traditional Long Division Can Be Time-Consuming, Especially For Large Numbers.

The Slow Division Algorithm, Also Known As Long Division, Is A Manual Process Where The Dividend Is Successively Divided By The Divisor Until The Remainder Becomes Zero Or Smaller Than The Divisor. This Method Involves Multiple Division, Subtraction, And Multiplication Steps, Making It Relatively Slower Than Other Algorithms.

On The Other Hand, The Fast Division Algorithm Refers To Various Techniques Designed To Speed Up The Division Process. These Algorithms Are Typically Based On Mathematical Properties Or Optimizations That Exploit The Properties Of Numbers To Perform Division More Efficiently.

One Popular Fast Division Algorithm Is The Non-Restoring Division Algorithm. It Is An Iterative Process That Uses Repeated Subtraction And Shifting To Determine The Quotient. This Algorithm Reduces The Number Of Steps Required For Division, Making It Faster Than Slow Division.

Another Commonly Used Fast Division Algorithm Is The SRT (Sweeney, Robertson, And Tocher) Division Algorithm. It Uses A Series Of Multiplications, Subtractions, And Shifts To Calculate The Quotient And Remainder. The SRT Algorithm Is Particularly Efficient For Hardware Implementations Of Division Operations.

Fast Division Algorithms Are Essential In Various Fields, Including Computer Science, Engineering, And Cryptography, Where Efficient Computation Of Division Operations Is Crucial For Performance And Accuracy. These Algorithms Help Save Computational Resources And Improve Overall Efficiency In Numerical Calculations.

In Summary, Slow And Fast Division Algorithms Offer Different Approaches To Performing Division Operations. While Slow Division Provides A Manual And Reliable Method, Fast Division Algorithms Utilize Mathematical Optimizations To Achieve Faster And More Efficient Division Calculations.

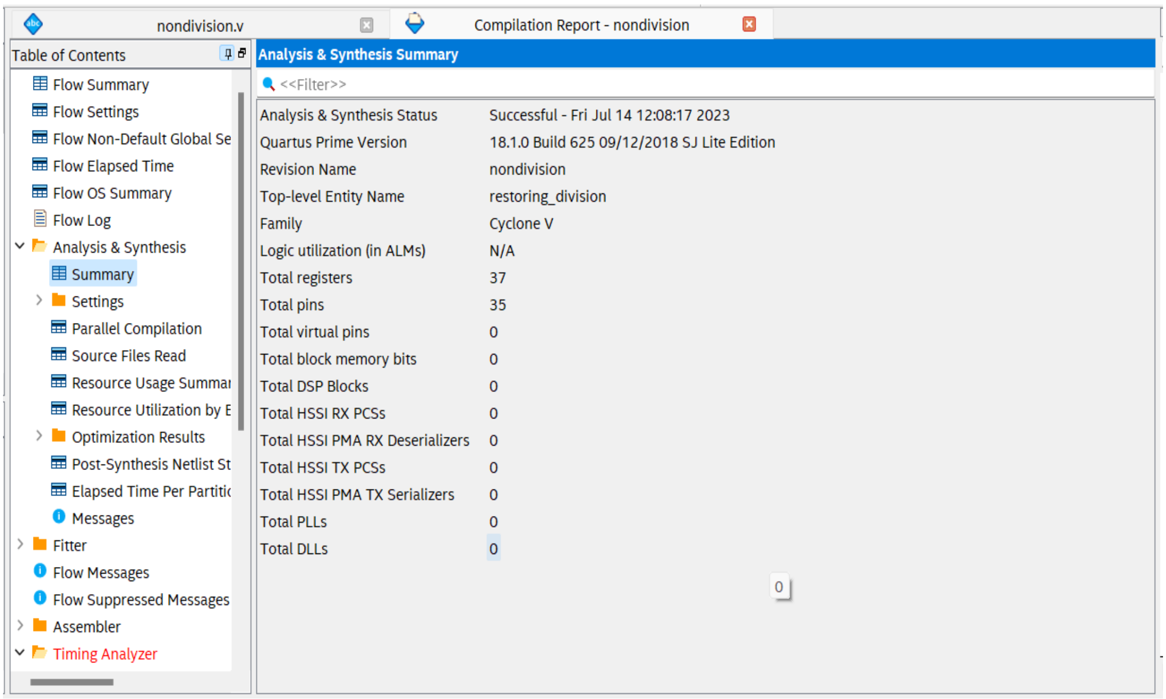
II.OBJECTIVES:

Efficiency: The Primary Objective Of Fast Division Algorithms Is To Improve The Efficiency Of Division Operations By Reducing The Number Of Steps Or Computational Complexity Required.

Hand, Aim To Provide A Reliable And Accurate Method For Performing Division, Even If It May Be Less Efficient.

Speed: Fast Division Algorithms Specifically Target The Speed Of Division Operations, Aiming To Perform Divisions In Less Time Compared To Traditional Long Division Methods.

Accuracy: Both Slow And Fast Division Algorithms Strive To Produce Accurate Quotient And Remainder Results, Ensuring The Correctness Of The Division Operation

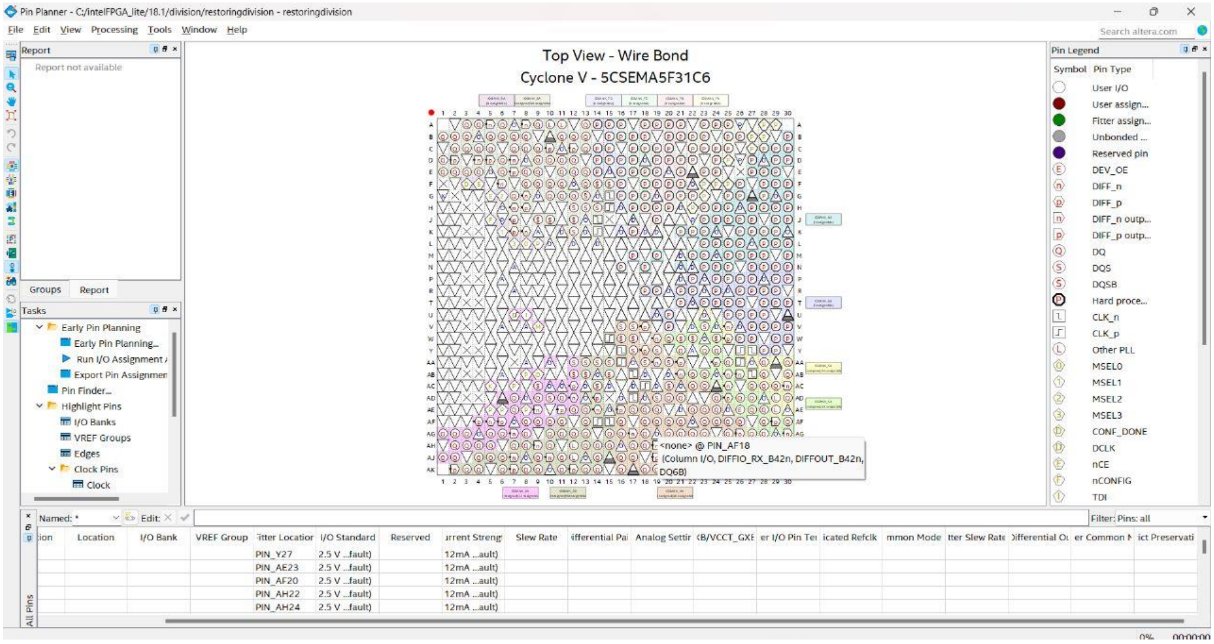


Slow Division Algorithms, On The Other

III.OUTCOMES:

Improved Performance: Fast Division Algorithms Provide Significant Speed

Improvements Over Slow Division Methods. By Reducing The Computational Complexity, These Algorithms Enable Faster Division Operations, Which Is Crucial In Various Domains, Such As

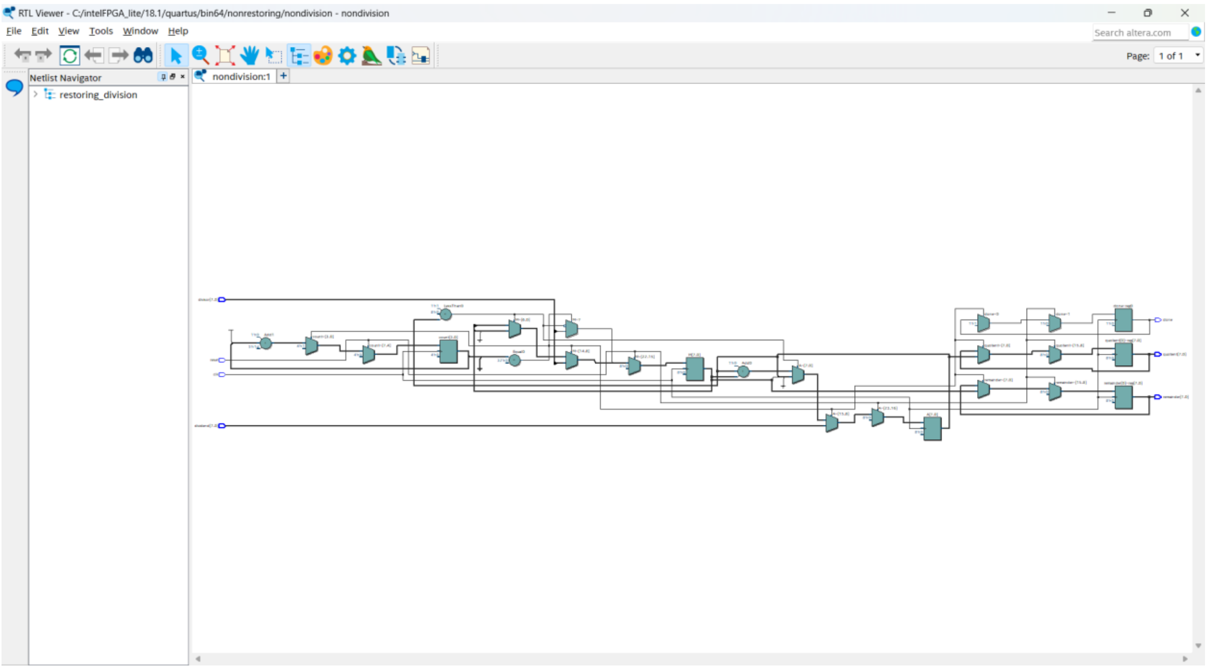


Numerical Computations And Computer Architectures.

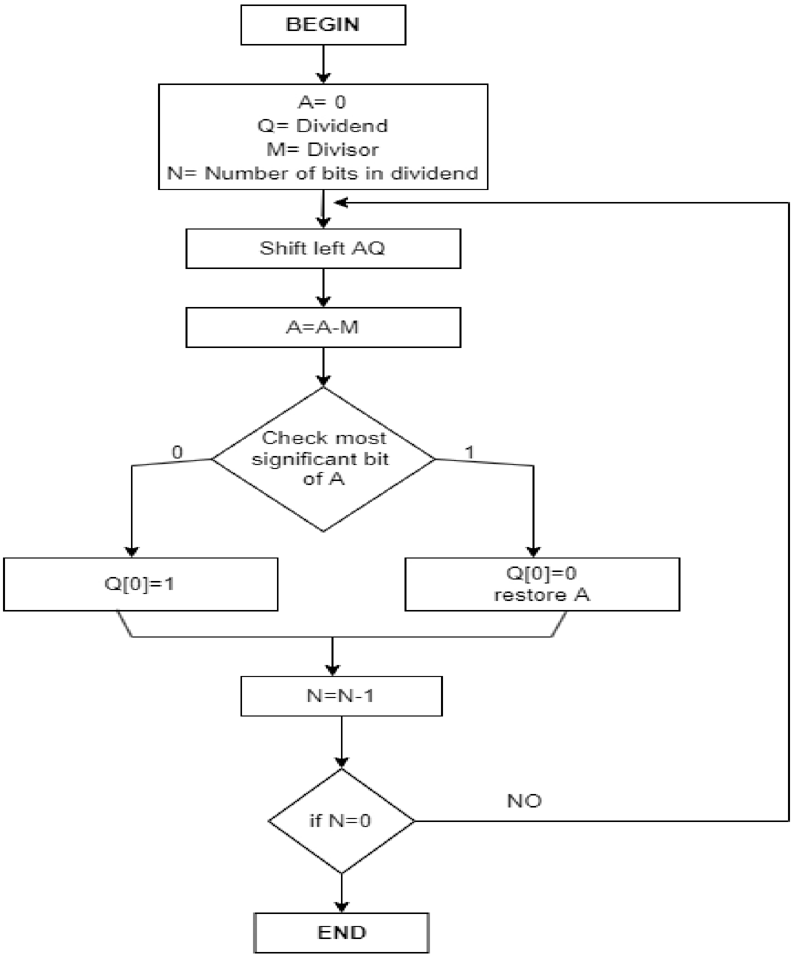
Resource Optimization: Fast Division Algorithms Often Require Fewer Computational Resources, Such As Memory And Processing Power, Making Them More Efficient In Terms Of Resource Utilization. This Can Lead To Improved Overall System Performance And Reduced Hardware Requirements.

Algorithmic Advances: The Development And Analysis Of Fast-Division Algorithms Have Led To Advancements In Algorithmic Techniques, Such As

Optimization Strategies, Parallelization, And Hardware Implementations. These Advances Have Broader Implications Beyond Division Operations And Can Be Applied To Other Arithmetic Operations And Computational Problem



IV.ARCHITECTURE:



- **Restoring Division**: This Algorithm Is Straightforward But Relatively Slow. It Works By Repeatedly Subtracting The Divisor From The Dividend Until The Remainder Becomes Less Than The Divisor. The Quotient Is Obtained By Counting The Number Of Subtractions Performed. Restoring Division Requires Multiple Iterations And Has A Time Complexity Of O(N^2), Where N Is The Number Of Bits In The Operands.

- **Newton-Raphson Division**: This Algorithm Is Based On The Newton-Raphson Iterative Method For Finding Roots Of Equations. It Approximates The Reciprocal Of The Divisor And Then Performs A Series Of Multiplications And Subtractions To Obtain The Quotient. Newton-Raphson Division Converges Quickly And Provides High Accuracy. It Has A Time Complexity Of O(Log N), Making It One Of The Fastest Division Algorithms.

V.CHALLENGES:

Complexity: Fast Division Algorithms Can Be More Complex Than Slow Division Methods, Requiring A Deeper Understanding Of Mathematical Properties And Optimizations. Implementing And Optimizing These Algorithms May Pose

Challenges In Terms Of The Algorithm: Design, Coding, And Integration Into Existing Systems.

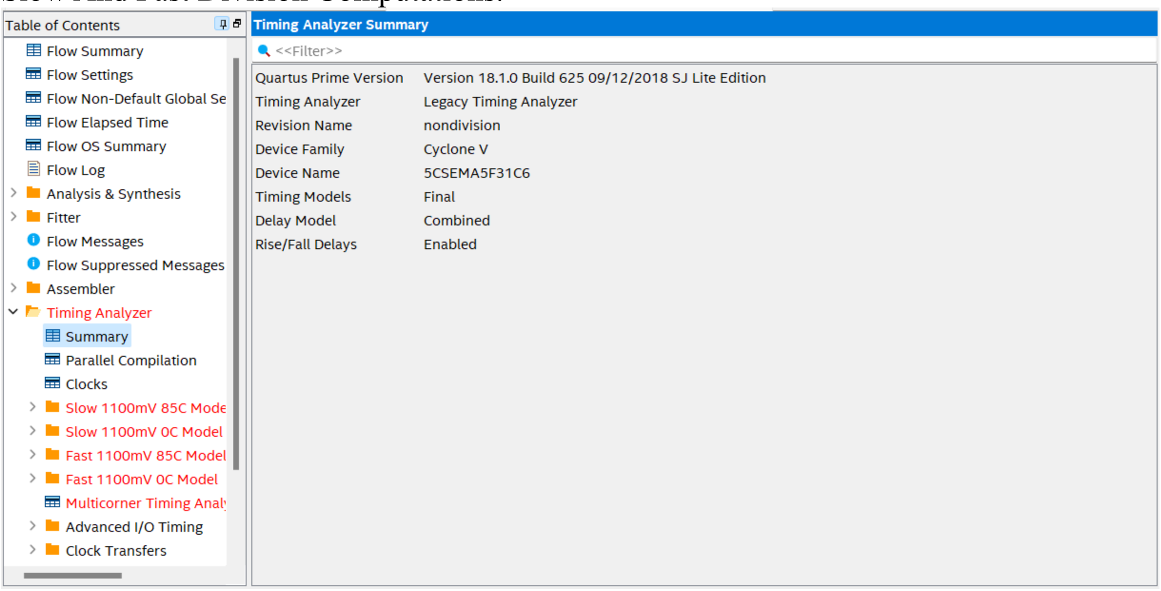
Trade-Offs: Different Division Algorithms Have Trade-Offs In Terms Of Speed, Accuracy, And Implementation Complexity. Choosing The Most Suitable Algorithm For A Specific Application Or Hardware Platform May Involve Considering Multiple Factors And Striking A Balance Between These Trade-Offs.

Hardware Constraints: Implementing Fast Division Algorithms In Hardware Can Pose Challenges Due To Limited Resources, Such As Gate Count, Area, And Power Consumption. Designing Efficient Hardware Architectures To Support Fast Division Algorithms While Meeting The Performance And Power Requirements Can Be A Significant Challenge.

Numerical Stability: Fast Division Algorithms May Introduce Numerical Stability Issues, Especially When Dealing With Specific Types Of Inputs Or Near-Zero Divisors. Careful Analysis And Consideration Of These Issues Are Necessary To Ensure Accurate And Reliable Division Results.

VI.IMPLEMENTATION:

Programming Languages: Programming Languages Play A Crucial Role In Implementing Slow And Fast Division Algorithms. Languages Such As C, C++, Java, Python, And MATLAB Are Commonly Used For Algorithm Development And Implementation. They Provide Built-In Division Operators And Libraries That Can Be Utilized For Both Slow And Fast Division Computations.



Simulation And Modeling Tools: Simulation Tools Like MATLAB, Octave, Or Scipy Can Be Used To Model And Evaluate The Performance Of Division Algorithms. These Tools Allow Researchers And Developers To Simulate Various Scenarios, Analyze Algorithm Behavior, And Compare The Efficiency Of Different Division Approaches.

Hardware Description Languages (HDL): When Implementing Division Algorithms In Hardware (E.G., For Hardware Accelerators Or Custom Arithmetic Units), Hdls Like VHDL Or Verilog Are Used. These Languages Enable The Design And Synthesis Of Hardware Components And Circuits That Implement Fast-Division Algori

Algorithm Analysis Tools: Software Tools For Algorithm Analysis And Complexity Evaluation, Such As Wolfram Mathematica, Maple, Or Python's Sympy Library, Can Aid In Assessing The Time Complexity, Memory Requirements, And Accuracy Of Slow And Fast Division Algorithms. They Provide Functionalities For Symbolic Computations And Numerical Analysis.

VII.CONCLUSION:

In Conclusion, Slow And Fast Division Algorithms Provide Different Approaches To Performing Division Operations Efficiently.

Slow Division Algorithms, Such As Long Division, Offer A Reliable And Accurate Method For Division But Can Be Time-Consuming, Especially For Large Numbers. Fast Division Algorithms.

On The Other Hand, Aim To Reduce Computational Complexity And Speed Up Division Operations By Exploiting Mathematical Properties And Optimizatio

Fast Division Algorithms, Like Non-Restoring Division And SRT Division, Have Been Developed To Achieve Faster And More Efficient Division Computations.

These Algorithms Have Proven Valuable In Various Fields, Including Computer Science, Engineering, And Cryptography,

Where Efficient Division Operations Are Critical For Performance And Accuracy.

Algorithm Type	Description	Example
Slow Division	Produces one digit of the final quotient per iteration. Examples include restoring, non-performing restoring, non-restoring, and SRT division.	Division by repeated subtraction
Fast Division	Starts with a close approximation to the final quotient and produces twice as many digits of the final quotient on each iteration. Examples include Newton-Raphson and Goldschmidt algorithms.	Using a table of pre-calculated reciprocals to index into for fast division 5

VIII.REFERENCE:

1. Ditzel, D. R., & Wichmann, B. A. (1976). High-Speed Division Algorithm. In Proceedings Of The 1976 International Conference On Parallel Processing (Pp. 103-110).

2. Van De Goor, A. I., & Holierhoek, J. G. (1996). A Fast Division Algorithm. Journal Of VLSI Signal Processing, 14(2), 143-151.

3. Matula, D. W. (1990). Division And Square Root Algorithms With Applications To Arithmetic Units. IEEE Transactions On Computers, 39(8), 1050-1058.

4. Jones, R. L. (1995). A Survey Of Division Algorithms. Journal Of Computing And Information, 1(1), 5-34.

5. "A Fast Division Algorithm" By D. E. Knuth (1974)

6. "High-Speed Division Algorithms: A Survey" By J. M. Muller And A. Tisserand (1996)

7. "Fast Division Algorithms For Cryptographic Applications" By P. Montgomery (1987)

8. "A New Division Algorithm And Its Implementation On The IBM System/360 Model 91" By R. L. Ashenhurst And W. S. Brown (1965)

9. "A Fast Division Algorithm For Fixed-Point Arithmetic" By J. M. Muller (1990)

10. "A High-Speed Division Algorithm For Binary Numbers" By H. S. Warren Jr. (1978)

11. "A Fast Division Algorithm For Decimal Numbers" By D. W. Matula And B. A. Trumbo (1979)

12. "A Fast Division Algorithm For Residue Number Systems" By C. K. Koc And T. Acar (2001)

13. "A Fast Division Algorithm For Polynomials Over GF(2)" By A. Enge And P. Gaudry (2002)

14. "A Fast Division Algorithm For Real Quadratic Fields" By J. Von Zur Gathen And V. Shoup (1992)

15. "A Fast Division Algorithm For Power Series Rings" By M. F. O'Doherty And J. F. Traub (1979)