International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org, editorijettcs@gmail.com
Volume 11, Issue 4, July – August 2022 ISSN 2278-6856

Simulation of Floating-Point Division Algorithms

Shreehitha M. U.[1], Dr. P.N. Jayanthi [2]

[1]Department of Electronics and Communication,
RVCE, Bangalore, India

[2]Department of Electronics and Communication,
RVCE, Bangalore, India

Abstract: ***Multiple algorithms are available to implement division on hardware. The rate of quotient convergence, mathematical equations with respect to its formulae, and underlying hardware primitives all change between these algorithms. The complexity of these algorithms increases when it comes to floating-point division. The research proposes a comparison of three division algorithms based on their simulations and design implications. Sweeney, Robertson, and Tocher (SRT), Gold Schmidt, and Coordinate Rotation Digital Computer (CORDIC) division algorithms are simulated on the Vivado design suite and compared. It is found that CORDIC outperforms the other two by giving the least delay of 4.254ns and 235.048Mhz of maximum frequency. The results show a 61.19% and 51% decrease in delay for CORDIC with respect to Gold Schmidt and SRT algorithms respectively.***

Keywords: SRT, CORDIC, Gold Schmidt, Division

1. INTRODUCTION

Addition, subtraction, multiplication, and division are the unavoidable construction blocks for the implementation of modern applied mathematics. Division operation is different when compared to other arithmetic operations as it doesn't have commutative and associative characteristics. This makes the implementation of the division algorithm difficult. Fast rate of reaction and important calculations with the decreased area are the main requirements for the algorithms that are being used in the enhanced technologies. Adder, subtracter, and multiplication logic are easier to build and implement than division algorithms. The usual delay in performance for these arithmetic operations is between a duad of clock cycles to not more than decuplets o clock cycles. The latency of division algorithms varies in the range of tens of clock cycles. In comparison to other arithmetic operations, the divider module is larger and takes longer to complete. However, certain systems require the use of a divider module to accomplish the desired functionality.

Implementation of the division algorithm becomes a more tedious task when it is to be applied to floating-point numbers. Floating-point division operation characterizes longer delays and low frequency in operation. With the advancement in the commercial and financial sectors and due to rapid growth in Internet-based applications, computer manufacturers are now considering hardware support for floating-point division algorithms. Floating-point arithmetic (FPA) is frequently employed in a wide range of applications [1,2,3]. In terms of complexity, area needs, and performance speed, the FPA division operation is fairly intensive. Institute of Electrical and Electronics Engineers (IEEE) 754 is the technical standard for floating pointing arithmetic and specifications of decimal operations have been updated to the revised version of IEEE 754R. The IEEE-754 floating-point [4,5] standard specifies the number format as well as several rounding options that affect the accuracy of the result. Different algorithms of division have various logic for quotient conversion, contrasting design requirements, and discrete purposes [6,7]. The comparative examination of different classes of division is briefly discussed further in the following section.

2. DIVISION ALGORITHMS

There are five different types of division algorithms [8]. Most prominently division algorithms can be differentiated as serial, sequential, parallel, pipeline, slow, fast, iterative, and predictive based on the accessing techniques. Whereas depending on the hardware architecture they can be identified as digit recurrence, functional iteration, very high radix, look-up table, and variable latency division algorithm-based dividers. The major classifications are discussed further.

The main bifurcation of the division algorithm is based on performance [10]. Slow division and fast division are the classifications under this category. Each cycle of slow division algorithms produces one digit of the final quotient. Restoration, non-performing restoring, non-restoring, and SRT division are all examples of slow division. On each iteration, fast division methods start with a near estimate of the eventual quotient and produce twice as many digits. This includes the Newton–Raphson, and Goldschmidt algorithms. Dividers can be categorized into three types according to their hardware architecture [6,9]: serial or sequential, parallel, and pipelined. Division algorithms may be classified into the following categories based on the mechanism of conversion [8]: look-up table variable latency, digit recurrence, functional iteration, and very high radix. The Simple conversion logic of the digit recurring class makes it utilized more. The quotient is calculated using iterative type subtraction. Restoring, Non-restoring and SRT are the three types of dividers covered by this

class of division algorithm. A functional iteration divider works on multiplication rather than subtraction to achieve quotients. Based on execution [11], dividers can be classified as iterative subtraction types and predictive types. When no hardware multiplier is available, methods like pseudo-multiplication, pseudo-division, and factor combining are typically utilized. Additions, subtractions, bit shifts, and lookup tables are the only operations these multipliers require. As a result, they're all classified as shift-and-add algorithms. CORDIC is one such algorithm that is closely related to shift-and-add algorithms.

The literature review sheds light on the significance of floating-point dividers and compares existing techniques for their implementation. The CORDIC algorithm is found to be efficient in the implementation of a double-precision floating-point division. In this work, two major division algorithms such as SRT and Gold Schmidt are compared with CORDIC to prove the efficiency of CORDIC with latency as an important constraint. The existing works show how each algorithm is implemented and different application aspects of it are discussed. A comparison of the three major algorithms in terms of delay and frequency is discussed in this paper that can help architects decide on which algorithm to be used for delay-sensitive applications.

The rest of the paper is organized with theoretical aspects of the topics involved in the work followed by the actual work done. The next section gives a brief overview of the three major division algorithms that are compared in this paper. Section III details the simulations performed along with their individual results. The implementation results and comparisons are concluded under section IV.

2.1 SRT Division Algorithm

D. Sweeney, J. E. Robertson, and T. D. Tocher independently discovered a prominent method for division, called the SRT division algorithm. This algorithm is been used in several microprocessor implementations and the main goal is to speed up the non-restoring division. SRT division is a kind of non-restoring division, the only exception is that it determines each quotient digit using a lookup table based on the dividend and divisor. The methods for calculating the resultant exponent and sign are simple in this algorithm. Radix of the SRT division can be 2 to higher numbers. In the SRT analysis, the input operands are advised to be in a normalized floating-point format. Simple small radix implementation, final reminder and quotient availability at the end of the computation, and the non-necessity of pre-scaling operands are the few advantages of the SRT algorithm. The disadvantages involve the possibility to identify more than a single quotient value during digit selection, the necessity of attention for designing quotient selection logic for high radix implementations, etc.

2.2 Gold Schmidt Algorithm

Robert Elliott's Goldschmidt division is named after him. Goldschmidt uses an iterative method in which a prime element is multiplied on the dividend and divisor. The prime element is decided in such a way that when it multiplies the divider and divisor, the divisor converges to 1. This way the desired quotient is approached by the dividend. Taylor-Maclaurin series of 1/(x+1) gave a path to the origination of the Goldschmidt algorithm for division. The Gold-Schmidt division's main benefit is its low latency. This divider can be utilized in Very Large-Scale Integration (VLSI) implementations where a designer wants to improve design performance. Each loop employs two multipliers. One of the division's drawbacks is the high cost of pipeline deployment. As a result, a serial implementation is advantageous. Another drawback of this splitter is its precision. It is not suited for systems with high accuracy since input operands must be pre-scaled. Goldschmidt's algorithm requires two parallel independent multipliers, which turns out to be a disadvantage in terms of area overhead. The multipliers in turn require an immense number of logic areas when implemented for floating-point arithmetic operation. Best precision is achieved on a greater number of iterations, which increases the cost of implementation of the Goldschmidt algorithm. The gold Schmidt dividers were used in the IBM 360/91 [12] and TMS390C602A [13].

2.3 CORDIC Algorithm

Volder's algorithm is also known as CORDIC (COordinate Rotation DIgital Computer) algorithm. In the computation of trigonometric, logarithmic, exponential, and hyperbolic functions, and also in functions that converge to one bit or digit per iteration like square roots, divisions, multiplications, etc, simplistic and effective algorithms like linear CORDIC, digit-by-digit circular CORDIC, and hyperbolic CORDIC can be utilized. In boards like microcontrollers, and FPGAs, where there is an absence of hardware multiplier, the CORDIC division algorithm is most efficient as the algorithm only utilizes adders, subtracters, bit shifters, and look up tables. In special case scenarios where the platform lacks space or has cost-related concerns, this algorithm is often utilized to solve floating-point arithmetic problems. CORDIC division architecture works in a way such that the bits/digits are produced one at a time. It takes place by orienting the divisor under the dividend with continuous subtraction until the remains of the dividend are too small for further subtraction. When applied to binary numbers, this algorithm is essentially the same as the long division algorithm. Corner cases like the divisor being zero have to be handled via exceptions. In the scenario where the divisor is negative, the sign of the divisor and of dividend have to be changed. The limitation of this algorithm is that it is cost-effective due to its ASIC design.

3 SIMULATION

SRT, Gold Schmidt, and CORDIC division algorithms are simulated on Xilinx Vivado Integrated Design Suite. Vivado creates an integrated design environment that has built-in system-to-IC tools and a debug environment for the same. The Design and process specifications considered in the Integrated Synthesis Environment (ISE) are tabulated in table 1. The Register Transfer Level (RTL) Schematic of the SRT Division, Gold Schmidt, and CORDIC division implementation is delineated in Fig.1, Fig.2, and Fig.3 respectively.

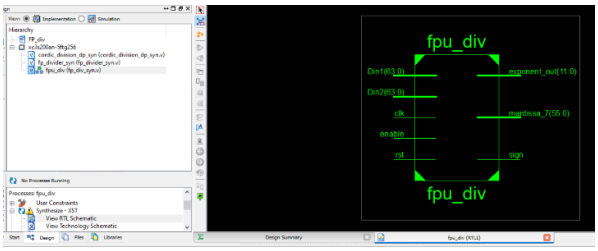

Figure 1 RTL Schematic of SRT Division

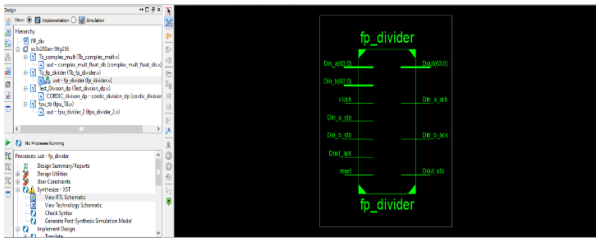

Figure 2 RTL Schematic of Gold Schmidt Division

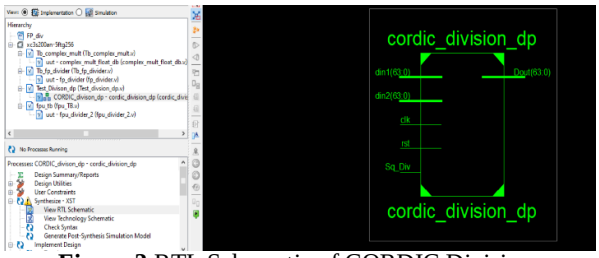

Figure 3 RTL Schematic of CORDIC Division

Table 1 Simulation Parameters

Specifications	Values
Top-Level Source Type	HDL
Family	Spartan3A and Spartan3AN
Device	XC3S400A
Package	FG400
Speed	-5
Synthesis Tool	XST(VHDL/Verilog)
Simulator	Isim (VHDL/Verilog)
VHDL Source Analysis Standard	VHDL-93
Simulation Run time	1000ns

4 RESULTS

The simulation results of the three division algorithms in comparison are as follows. The results are shown in hexadecimal 64bit representation of IEEE 754 floating-point standards. SRT divider results are as shown in Fig.4. Divider based on gold Schmidt algorithm and CORDIC algorithm simulation results are depicted in Fig.5. and Fig.6. respectively. Looking into the synthesis report of each algorithm simulation, obtained delay and maximum frequency achieved are summarized in Table 2. Area coverage for implementation is compared by analyzing the logic utilization. Table 3 summarizes the logic utilization for each algorithm. Both CORDIC and Gold Schmidt can be handy for architects who consider the area as the major constraint. CORDIC is the best option for hardware implementations where latency is the prime factor.

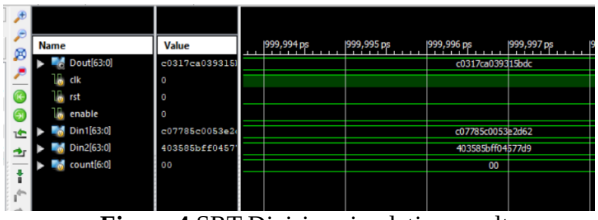

Figure 4 SRT Division simulation result

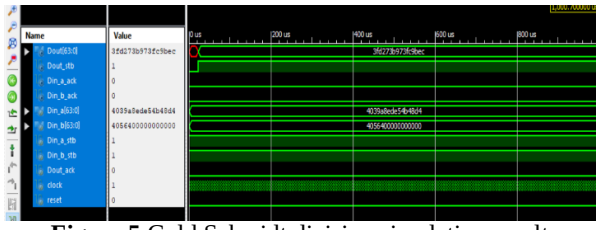

Figure 5 Gold Schmidt division simulation result

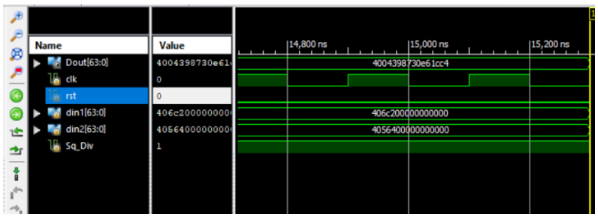

Figure 6 CORDIC division simulation result

Table 2 Comparison of delay and frequency for SRT, gold Schmidt, and CORDIC division algorithms

DIVISION ALGORITHM	SRT	GOLD SCHMIDT	CORDIC
DELAY	10.962ns	8.683ns	4.254ns
MAXIMUM FREQUENCY (MHz)	91.227	115.171	235.048

LOGIC UTILIZATION		DIVISION ALGORITHM		
		SRT	GOLD SCHMIDT	CORDIC
Total number of slice registers	Used	934	756	311
	Utilization	13%	10%	4%
Number of Occupied Slices	Used	1556	956	1489
	Utilization	43%	26%	41%
Number of 4 input LUTs	Used	2507	1383	2774
	Utilization	34%	19%	38%
Number of bonded IOBs	Used	193	200	194
	Utilization	62%	64%	62%
Average Fanout of Non-Clock Nets		3.9	3.34	2.69

Table 3 Area Comparison

5 *CONCLUSIONS*

Despite the fact that division appears to be a simple operation, it is extremely complex to perform due to rigorous conversion requirements, and an effective system requires an efficient divider. In this work, three-division algorithms are compared with respect to concerning features such as delay, maximum frequency, and area. SRT, Gold Schmidt, and CORDIC division algorithms are compared and summarized. CORDIC architecture for division outperforms the other two with the least delay of 4.254ns and a maximum frequency of 235.048MHz. With respect to area constraints of implementation, both Gold Schmidt and CORDIC can be used. The advantage of CORDIC architecture is that same can also be used to implement an algorithm to detect square roots.

References

[1] Guo, Zhi, Walid Najjar, Frank Vahid, and Kees Vissers. "A quantitative analysis of the speedup factors of FPGAs over processors." In Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays, pp. 162-170. 2004.

[2] Jaiswal, Manish Kumar, and Nitin Chandrachoodan. "FPGA-based high-performance and scalable block LU decomposition architecture." IEEE transactions on computers 61, no. 1,pp: 60-72,2011.

[3] Wang, Xiaojun, Sherman Braganza, and Miriam Leeser. "Advanced components in the variable precision floating-point library." In 2006 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, pp. 249-258. IEEE, 2006.

[4] IEEE Computer Society. Standards Committee. Working group of the Microprocessor Standards Subcommittee, and American National Standards Institute. IEEE standard for binary floating-point arithmetic. Vol. 754. IEEE, 1985.

[5] IEEE Computer Society. "IEEE standard for floating-point arithmetic." IEEE Std 754-2008,pp: 1-70,2008.

[6] S. F. Oberman and M. J. Flynn, "Division Algorithms and Implementations," IEEE Transactions on Computers, vol. 46, no. 8, pp. 833-854, Aug. 1997.

[7] G. Sutter, G. Biol, J-P Deschamps, "Comparative Study of SRT-Dividers in FPGA," in Field Programmable Logic and Application. FPL 2004, J. Becker, M. Platzner, S. Vernalde Eds. Lecture Notes in Computer Science, vol 3203. Springer, Berlin, Heidelberg, pp. 209-220, 2004.

[8] U. S. Patankar and A. Koel, "Review of Basic Classes of Dividers Based on Division Algorithm," in IEEE Access, vol. 9, pp. 23035-23069, 2021, doi: 10.1109/ACCESS.2021.3055735.

[9] K. Tatas, D. J. Soudris, D. Siomos, M. Dasygenis, and A. Thanailakis, "A Novel Division Algorithm For Parallel And Sequential Processing," 9th International Conference on Electronics, Circuits, and Systems, Dubrovnik, Croatia, Dec. 10, 2002, ISBN: 0-7803-7596-3,p.p–553-556.

[10] S. Dixit, and Mohd. Nadeem, "FPGA Accomplishment of a 16-Bit Divider," Imperial Journal of Interdisciplinary Research (IJIR), vol. 3, issue 2, ISSN: 2454-1362, pp. 140-143, 2017.

[11] Muhd. Kasim, T. Adiono, Muhd. Fahreza, "FPGA Implementation of Fixed-Point Divider Using Pre-Computed Values," 4th International Conference on Electrical Engineering and Informatics ICEEI 2013, Procedia Technology, vol. 11, pp. 206-211, 2013.

[12] S. F. Anderson, J. G. Earle, R. E. Goldschmidt and D. M. Powers, "The IBM System/360 Model 91: Floating-Point Execution Unit," in IBM Journal of Research and Development, vol. 11, no. 1, pp. 34-53, Jan. 1967, doi: 10.1147/rd.111.0034.

[13] M. Darley et al., "The TMS390C602A floating-point coprocessor for Sparc systems," in IEEE Micro, vol. 10, no. 3, pp. 36-47, June 1990, doi: 10.1109/40.56324.