International Journal of Research and Review
Vol. 9; Issue: 11; November 2022
Website: www.ijrrjournal.com
E-ISSN: 2349-9788; P-ISSN: 2454-2237

Research Paper

Implementation and Comparison of Different Non-Restoring Division Algorithm

Vishwas B R[1], Dr. Kiran V[2]

[1]Student, Department of ECE, R V College of Engineering, Bengaluru, India
[2]Associate Professor, Department of ECE, R V College of Engineering, Bengaluru, India

Corresponding Author: Vishwas B R

DOI: https://doi.org/10.52403/ijrr.20221111

ABSTRACT

The non-restoring algorithm, which is derived from restoring division, determines the residual by repeatedly deducting the dividend from the shifted divisor until the remainder is within the desired range. Since just the shifting operation, arithmetic addition, and subtraction are used in the computation, non-restoring division requires less hardware to accomplish and provides the exact value of the quotient and remainder. In this paper, the Non-Restoring division algorithm is implemented in 2 ways for 64-bit input dividend and divisor and the method which dissipates less power compared to the other is shown.

Keywords: restoring division, non-restoring division, dividend, divisor, quotient, remainder

INTRODUCTION

Computer technology has advanced quickly ever since it was created. The fact that computers do math in order to operate programs and applications, however, has not altered. Addition, multiplication, and division, the 3 basic operations in arithmetic - are the basis on which computers handle a large number of numbers. In the field of digital signal processing, for instance in wireless signal processing, image processing, computer graphics, and other fields, division operation is used frequently. Due to anticipated space and power savings, fixed point arithmetic is typically used to design algorithms. Division takes substantially longer to calculate and is less frequently utilized in computer applications than multiplication, subtraction, or addition. Whether the method is implemented in hardware or software, division is the most challenging of all the fundamental operations. It has been demonstrated, though, that putting off its deployment can seriously harm the performance of many systems of the application.

Researchers have developed two different types of division techniques: digit recurrent division and division by convergence. However, digit recurrence division is the most used technique for division and square root in many floating-point units because it is straightforward and less difficult than division by convergence. Each method has its own merits. For digit recurrence division the representative algorithms include SRT, non-restoring, and restoring dividers. This research compares the two methods and focuses on the overall power savings of non-restoring division. There are many binary division algorithms and some of them include Multiplicative Algorithm, Continued Product Algorithm., Approximation Algorithms and CORDIC Algorithm.

METHODOLOGY

I. Restoring Division

Standard Long Division Method is another name for the restoring algorithm. The original partial remainder is rewritten to partial remainder if the partial remainder value following iterative subtraction is negative. Writing a quotient requires the usage of a non-redundant number system. To implement it is necessary to conduct comparison along the entire width inorder to determine the quotient. The main operations in this method are shift add and shift remove. Shifting may result in the loss of the most important bit, which could result in incorrect output and call for additional steps to check for overflow conditions. The implementation of this algorithm is very simple, which is an advantage. The drawbacks include its latency and space requirements.

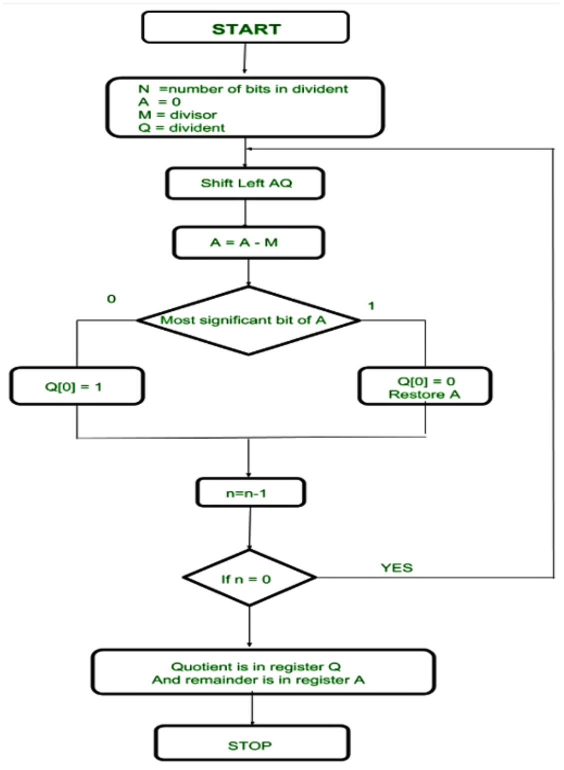

START
N =number of bits in divident
A = 0
M = divisor
Q = divident
Shift Left AQ
A = A - M
0
Most significant bit of A
1
Q[0] = 1
Q[0] = 0
Restore A
n=n-1
If n = 0
YES
Quotient is in register Q
And remainder is in register A
STOP

Figure 2.1 Restoring Division Algorithm

Restoring division is commonly performed on the fixed-point fractional numbers. The division algorithm returns the quotient and remainder when division operation is carried out on two numbers. In order to implement this algorithm, we assume that 0 < D < N. With the aid of digit set [0, 1], the restoring division algorithm can be used to formulate the quotient q. To store the quotient, a register Q is employed and register A is used to store or contain the remainder. The register M will be loaded with the divisor and n-bit divided will be loaded into the register Q. The starting value of a register is 0. This algorithm is named as Restoring division algorithm because at the time of iteration, the values of these registers are restored.

II. Non-Restoring Division

In this method, if the result of the subtraction is negative, the result is not restored. The partial remainder lies between the range of [–DR, +DR]. The first iteration is always subtraction when the dividend is positive. Therefore, the iteration may be set as R0 = 2DD - DR. The major disadvantage is that we need to maintain an extra sign bit and decide if we need to perform addition or subtraction which further leads to area and latency limitations during algorithm implementation. Another disadvantage is we need to maintain separate hardware to perform addition or subtraction. Further optimization is possible with 2’s complement, in which –DR is replaced by 2’s complement of DR.

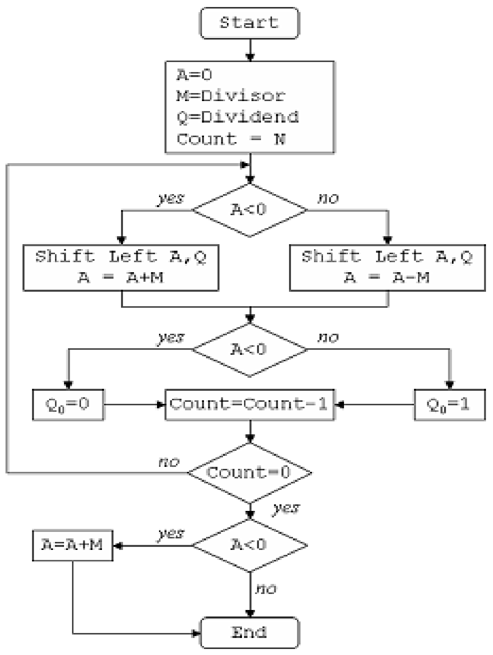
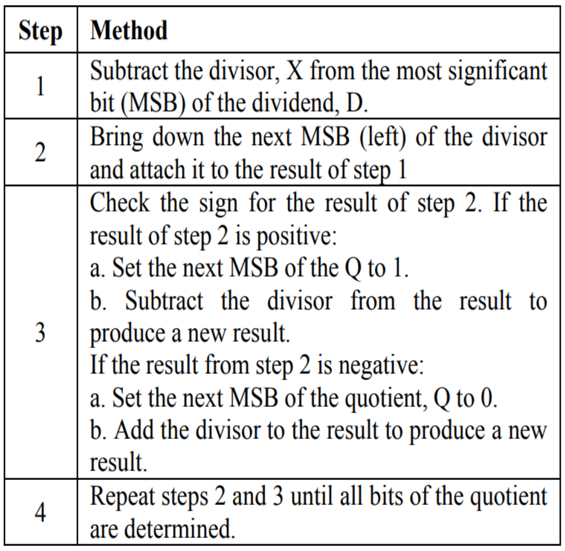


Figure 3.1 Non-Restoring Division Algorithm

There are 2 methods which are used to implement non-restoring division. A comparison is made between the 2 methods.

A. Method 1

The steps involved in this method is mentioned below:



Step	Method
1	Subtract the divisor, X from the most significant bit (MSB) of the dividend, D.
2	Bring down the next MSB (left) of the divisor and attach it to the result of step 1
3	Check the sign for the result of step 2. If the result of step 2 is positive: a. Set the next MSB of the Q to 1. b. Subtract the divisor from the result to produce a new result. If the result from step 2 is negative: a. Set the next MSB of the quotient, Q to 0. b. Add the divisor to the result to produce a new result.
4	Repeat steps 2 and 3 until all bits of the quotient are determined.

Figure 3.2 Non-Restoring Division Algorithm (Method 1)

B. Method 2

- Register A has a starting bit/value of 0
- The set [-1, 1] is used instead of the quotient digit set [0, 1], by the non-restoring division algorithm.
- Compared to restoring division algorithm, the non-restoring division algorithm is more complex. It contains only one decision and addition/subtraction per quotient bit when implemented in hardware which is advantageous.
- There will not be any restoration operations after the subtraction operation is completed. The numbers of operations which are performed are reduced to half and the execution will be faster in comparison.
- Basically, this algorithm executes elementary operations like addition and subtraction.
- The sign bit of register A will be utilised in this approach.

RESULT

The non-inverting division algorithms are written using Hardware Description Language such as Verilog and simulated using the Xilinx Vivado 2014.4. The testbench is written for different values of dividend and divisor which are of 64-bit size and the power consumed in each method is compared.

The simulation results of implementation of Non – Restoring division using Method 1 is shown along with the schematic.

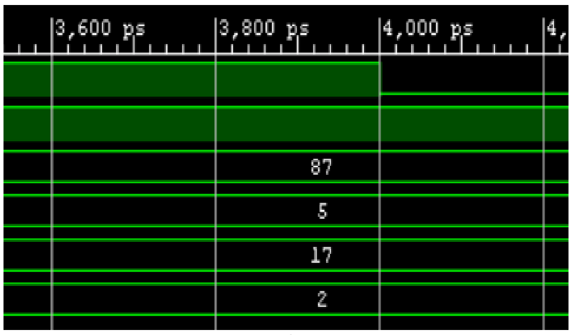


Figure 4.1 Simulation results for Dividend = 87, Divisor = 5 which yields Quotient = 17 and Remainder = 2

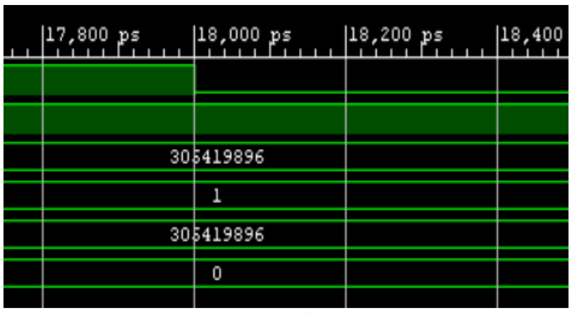


Figure 4.2 Simulation results for Dividend = 305419896, Divisor = 1 which yields Quotient = 305419896 and Remainder = 0

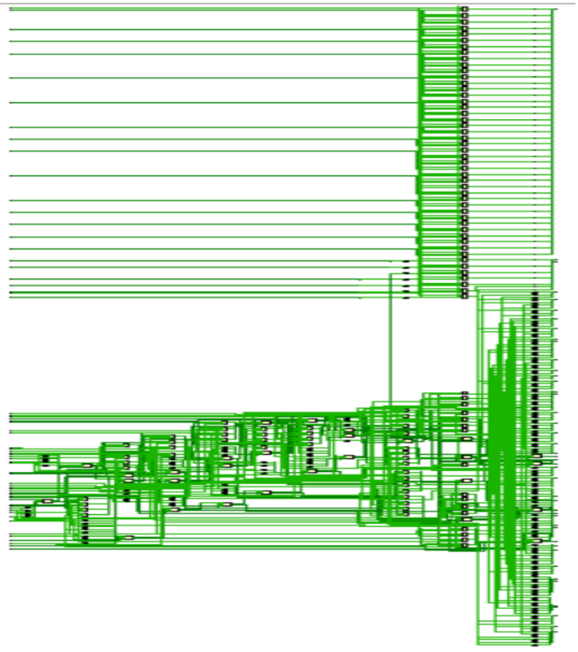


Figure 4.3 Schematic of Non- Inverting Division algorithm using Method 1

The simulation results of implementation of Non-restoring division using Method 2 is shown along with the schematic.

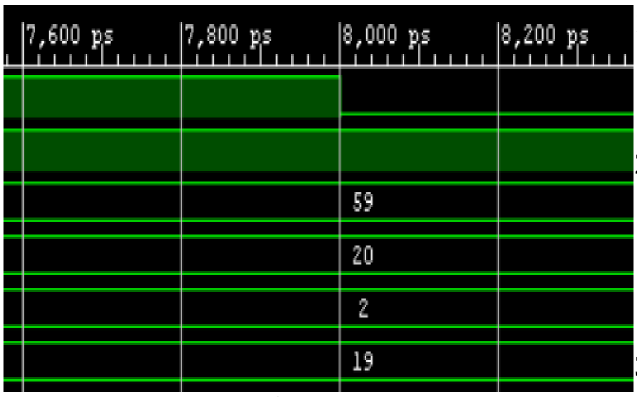


Figure 4.4 Simulation results for Dividend= 59, Divisor = 20 which yields Quotient = 2 and Remainder = 19

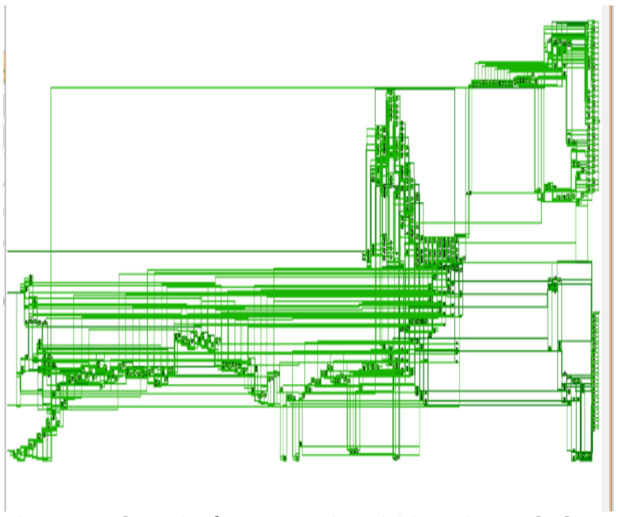


Figure 4.5 Schematic of Non- Inverting Division using Method 2

Table 4.1 Comparison of Power consumed by the 2 methods

Sl. No.	Method	Power (W)
1	Method 1	97.035
2	Method 2	25.795

CONCLUSION

Non-restoring division is implemented using 2 methods and the reduction in the power consumed is observed when implemented using the 2nd method. The schematic is also for the 2 methods.

Acknowledgement: None

Conflict of Interest: None

REFERENCES

1. S. F. Oberman and M. J. Fiynn, “Division algorithms and implementations”. IEEE Transaction on Computers. 46(8): 833-854, 1997
2. M. R. Patel, T.V. Shah and D. H. Shah, “Implementation and analysis of interval SRT radix-2 division algorithm”. International Journal of Electronics and Computer Science Engineering. 1(3): 971-976, 2012
3. Harris D. L., Oberman S. F. and M Horowitz. A., “SRT division architectures and implementations”. In: 13th IEEE Symposium on Computer Arithmetic. pp. 18-25, 1997
4. Tejas V. Shah, Deepali H. Shah and Dr. J.S. Shah, “Implementation of Floating Point Divider for Interval Arithmetic”
5. Jain S., Pancholi M., Garg H. and Saini S., “Binary division algorithm and high speed deconvolution algorithm”. In: 11th IEEE International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology. pp. 1-5, 2014
6. Gustavo Sutter, Jean-Pierre Deschamps, Gery Bioul and Eduardo Boemo, "Power Aware Dividers in FPGA," Power and Timing Modeling, Optimization and Simulation 2004, LNCS 3254, pp. 574-584,2004.
7. O. 1. Bedrij, "Carry-select adder," IRE Transactions on Electronic Computers, vol. EC-II, pp. 340 -346, 1962.
8. Behrooz Parhami, Computer Arithmetic - Algorithms and Hardware Designs, Oxford: Oxford University Press, 2000.

How to cite this article: Vishwas B R, Kiran V. Implementation and comparison of different non-restoring division algorithm. *International Journal of Research and Review*. 2022; 9(11): 70-73.
DOI: *https://doi.org/10.52403/ijrr.20221111*

******