

# Comparisons of Different Binary Division Algorithms

ananth7121999

August 2020

# Introduction

This presentation discusses different serial multiplication algorithms and compares them on different parameters such as delay, area and power consumption. The algorithms discussed are:

- Digit Recurrence Algorithm
  - Restoring Algorithms
  - Non-Restoring Algorithms

# SRT Division

Digit recurrence algorithms use subtractive methods to calculate quotients one digit per iteration. SRT division or Non-Restoring division is the name of the most common form of digit recurrence division algorithm. The following recurrence is used in every iteration of the SRT algorithm:

$$rP_0 = \text{dividend} \quad (1) \quad q_{j+1} = \text{SEL}(rP_j, \text{divisor}) \quad (3)$$

$$P_{j+1} = rP_j - q_{j+1} \text{divisor} \quad (2) \quad q = \sum_{j=1}^k q_j r^{-j} \quad (4)$$

$$\text{remainder} = \begin{cases} P_n & P_n \geq 0 \\ P_n + \text{divisor} & P_n < 0 \end{cases} \quad (5)$$

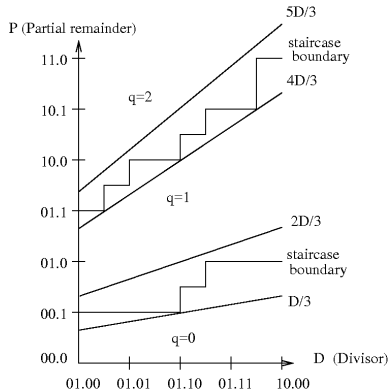
Implementing SRT divisor requires the following steps:

- **Choice of Radix** : Increasing the radix by a power of  $f$ ,  $r$  to  $r^f$ , reduces the number of iterations by a factor of  $f$ . Thus reducing latency, but increasing complexity.
- **Choice of Quotient Digit Set** : The simplest case is where, for radix  $r$ , there are exactly  $r$  allowed values of the quotient. However, to increase the performance of the algorithm, we use a redundant digit set. Such a digit set can be composed of symmetric signed-digit consecutive integers. But as the size of quotient digit set increases, the complexity and latency also increases.

## • Residual Representation :

- Conventional 2's complement representation or *nonredundant form*
- Carry-save 2's complement representation or a *redundant form*.

## • Quotient-Digit Selection Function The following diagram represents Radix-4 quotient digit selection function.



# Increasing Performance of SRT Divider

- **Simple Staging:** In order to retire more bits of quotient in every cycle, a simple low-radix divider can be replicated many times to form a higher radix divider. This helps to decrease latency, but increases area.
- **Overlapping Execution:** It is possible to overlap or pipeline the components of the division step in order to reduce the cycle time of the division step.
- **Overlapping Quotient Digit Selection:** To avoid the increase in cycle time that results from staging radix- $r$  segments together in forming higher radix dividers, some additional quotient computation can proceed in parallel.

# Functional Iteration

- **Newton-Raphson Method:** Division can be written as the product of the dividend and the reciprocal of the divisor. Here the Newton-Raphson Method is used to compute the reciprocal of the divisor and then the value is multiplied by the dividend using any binary multiplier.
- **Goldschmidt's algorithm:** Goldschmidt division (after Robert Elliott Goldschmidt) uses an iterative process of repeatedly multiplying both the dividend and divisor by a common factor  $F_i$ , chosen such that the divisor converges to 1. This causes the dividend to converge to the sought quotient  $Q$ .

$$\frac{N_{i+1}}{D_{i+1}} = \frac{N_i}{D_i} \cdot \frac{F_{i+1}}{F_{i+1}}, Q = N_k$$

# Disadvantages of Iterative Method

- The main disadvantage of using functional iteration for division is the difficulty in obtaining a correctly rounded result. With subtractive implementations, both a result and a remainder are generated, making rounding a straight forward procedure.
- The series expansion iteration, which converges directly to the quotient, only produces a result which is close to the correctly rounded quotient, and it doesnot produce a remainder.
- The Newton-Raphson algorithm has the additional disadvantage that it converges to the reciprocal, not the quotient. Even if the reciprocal can be correctly rounded, it does not guarantee the quotient to be correctly rounded.



# Comparison Table

Algorithm	Iteration Time	Latency (Cycles)	Approximate Area ( $\mu W$ )
SRT	Q-sel+ (multiple form+ subtraction)	$\lceil \frac{n}{r} \rceil + \text{scale}$	Qsel table + CSA
Newton-Raphson	2 serial multiplication	$2(\lceil \log_2 \frac{n}{i} \rceil + 1)t_{mul} + 1$	1 multiplier + table + control
Goldshmidt Method	1 multiplication	$(\lceil \log_2 \frac{n}{i} \rceil + 1)t_{mul} + 2, t_{mul} > 1$ $2\lceil \log_2 \frac{n}{i} \rceil + 3, t_{mul} = 1$	1 multiplier + table + control
Accurate quotient approx	1 multiplication	$(\lceil \frac{n}{i} \rceil + 1)t_{mul}$	3 multipliers + table + control

**Table:** Comparison of division algorithms






# Conclusion

Among all the division methods Radix-4 SRT or Non-restoring division is the best option for processors with area constraints and also provides reasonable amount of performance.

For fast computation where remainder value is not necessary the functional iteration methods like Newton-Raphson and Goldshmidt methods can be used.

Different High Radix algorithms also provide considerable boost in speed, but are of high complexity and requires larger area.

# References

-  M. D. Ercegovac and T. Lang. “Simple radix-4 division with operands scaling”. In: *IEEE Transactions on Computers* 39.9 (1990), pp. 1204–1208.
-  MiloTs Dragutin Ercegovac and Tomas Lang. *Division and Square Root: Digit-Recurrence Algorithms and Implementations*. USA: Kluwer Academic Publishers, 1994. ISBN: 0792394380.
-  J. Fandrianto. “Algorithm for high speed shared radix 8 division and radix 8 square root”. In: *Proceedings of 9th Symposium on Computer Arithmetic*. 1989, pp. 68–75.
-  Stuart F. Oberman and Michael J. Flynn. “AN ANALYSIS OF DIVISION ALGORITHMS AND IMPLEMENTATIONS”. In: (July 1995).
-  S. Waser and M. Flynn. *Introduction to Arithmetic for Digital Systems Designers*. 1982.