SRT DIVISION ALGORITHM

KUMAR SAI REDDY

1. Abstract:

The most frequent and popular version of the digit recurrence method is the SRT division algorithm, which Freiman called after the initials of Sweeney, Robert- son, and Tocher, who developed the algorithm independently at the same time.

2. Introduction:

Digit Recurrence Algorithm:

The division procedure begins with divisor d and dividend x, and then computes quotient q and its remainder, r. The fundamental division equation is recursive, as follows:

$$x = q \cdot d + r, \text{ where } r<d$$

One of the most intriguing aspects of the division process is the quotient digit. Quotients, in particular, are frequently implemented as a redundant digit set. The purpose for doing so is to make quotient digit selection easier. Unfortunately, as radix increase, so does the difficulty of quotient digit selection.

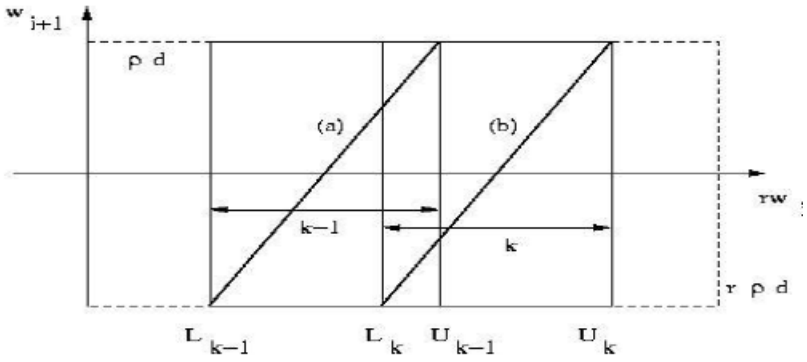
The division digit recurrence algorithm operates repeatedly using the following equation, where wi is the partial remainder for iteration i d is the divisor, r is the radix, and qi is the quotient digit for iteration i.

$$w_{i+1} = r \cdot w_i = q_{i+1} \cdot d$$

$$q_{i+1} = \mathrm{QST}(r \cdot w_i, d)$$

The comparison of the divisor and remainder to determine the quotient bit is the most difficult step in the division method. If this is accomplished by subtracting d from wi, be cautious if the result is negative. If this is the case, a restoration operation is performed to restore the remainder to the prior interaction; this procedure is known as restoring division. Non-restoring division is an alternative to sequential division in that it uses particular logic to avoid correcting the quotient; this is accomplished by include a correlation factor within the algorithm. Unfortunately, because non-restoring division necessitates a correction factor, additional post-processing may be necessary if the final remainder is negative.

i. As Radix increases, it reduces the number of iterations assuming r = 2k.
ii. Redundancy within the quotient digit set reduces and simplifies the QST.
iii. Partial remainder can be implemented using redundant notation, which simplifies the computation of the partial reminder using carry-free adder.



Robertson's PD Diagram

2.1 Quotient Selection Table:

Based on comparison between shifted partial remainder and the divisor. A symmetric SD digit set is utilized. The containment condition identifies selection interval for each quotient digit qi+1. On the other hand continuity condition details the range, which the quotient digit is selected.

Sign	Int	f0	Result	Quotient
0	0	0	<1/2	0
0	0	1	>=1/2	1
0	0	0	>=1/2	1
0	0	1	>=1/2	1
0	1	0	<-1/2	-1
0	1	1	<-1/2	-1
0	1	0	<-1/2	-1
0	1	1	>=1/2	0

2.2 Containment condition:

The containment condition sets up the selection intervals necessary for computing the subsequent quotient digit. For given quotient digit qi+1 to be chosen as k, there should be boundson an interval of allowable partial reminders, these regions are defined by the interval [Lk, Uk] such that Lk is the smallest value of partial remainder r.wj for which it is possible to choose qi+1 =k, whereas Uk is largest value of partial reminder r.wj for which it is possible to choose qi+1=k.

In other words selection intervals are for quotient digit qi+1 =k is given by: Uk = (k+P).d

Lk = (k.P).d

2.3 Continuity Condition:

Since the containment condition defines range of subsequent partial reminder, choosing the correct quotient digit from this region is job of continuity condi- tion. To satisfy the containment condition, the minimum value of x-axis of the Robertson's diagram is chosen such that qi+1=k is our quotient digit. This can be defined as the following inequality where sk is the minimum

value that user chooses before an implementation is devised.

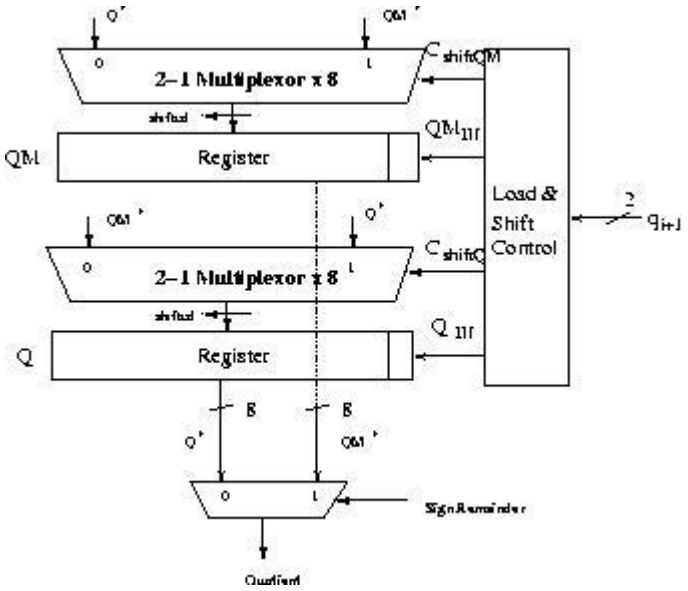
$$L_k <= s_k <= U_k$$

$$U_{k-1} - L_k = (k - 1 + \rho . d) - (k - \rho . d) = (2. \rho - 1).d$$

$$U_{k-1} \geq L_k$$

$$\rho \geq 2^{-1}$$

On the fly conversion:



3. Methodology:

Radix2 Division:

$$\frac{1}{2} \leq d \leq 1 \quad \frac{1}{2} \leq x \leq 1$$

$$\frac{1}{2} \leq q \leq 1$$

Radix2 Division: The Radix2 division SRT algorithm is most easy to implement. It produces one bit of quotient every iteration, requiring 25 or 54 clock cycles for single and double precision floating point respectively. This algorithm is an extension of non-restoring division with a quotient digit set of -1, 0, 1. The equation used for this division is as follows:

$$w_{i+1} = 2. w_i = q_{i+1} .d$$

$$q_{i+1} = 1 \quad \text{if } \frac{1}{2} < 2w[i]$$

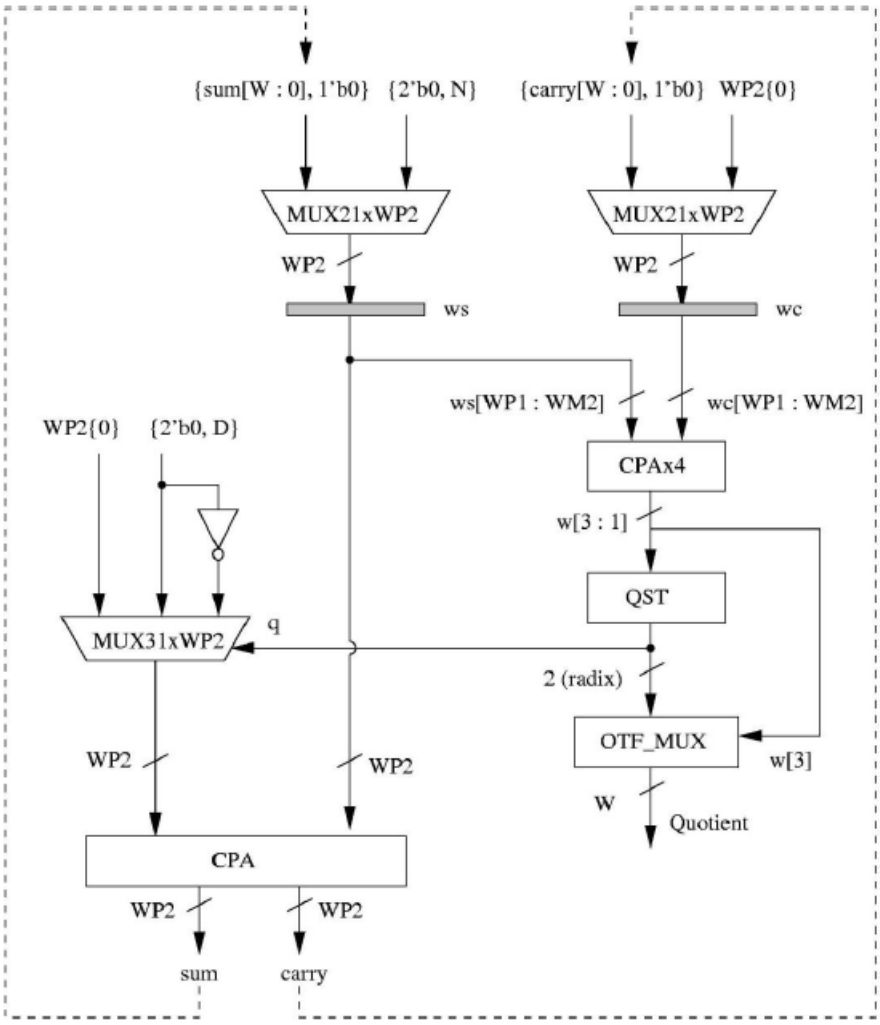
$$q_{i+1} = 0 \quad \text{if } -\frac{1}{2} < 2w[i] \leq \frac{1}{2}$$

$$q_{i+1} = -1 \quad \text{if } 2w[i] < -\frac{1}{2}$$

L1 = 0 U1 = 2 . d

L0 = -d U0 = d

L-1 = -2d U-1 = 0



Radix2 hardware block diagram

SRT division was named after Sweeney, Robertson and Touher. Main objective of this algorithm is to speed up the division by allowing 0 as a quotient digit. This eliminates need of subtraction or addition when the value of quotient selected is 0 Now we will derive the containment and continuity conditions for Quotient Selection table. Using the equation of containment and p=1.

Sign	Int	f0	Result	Quotient
0	0	0	<1/2	0
0	0	1	>=1/2	1
0	0	0	>=1/2	1
0	0	1	>=1/2	1
0	1	0	<-1/2	-1
0	1	1	<-1/2	-1
0	1	0	<-1/2	-1
0	1	1	>=1/2	0

z	.0 1 0 0 0 1 0 1	In [–1/2, 1/2), so OK
d	.1 0 1 0	In [1/2, 1), so OK
$-d$	1.0 1 1 0	
$s^{(0)}$	0.0 1 0 0 0 1 0 1	
$2s^{(0)}$	0.1 0 0 0 1 0 1	$\geq 1/2$, so set $q_{-1} = 1$
$+(-d)$	1.0 1 1 0	and subtract
$s^{(1)}$	1.1 1 1 0 1 0 1	
$2s^{(1)}$	1.1 1 0 1 0 1	In [–1/2, 1/2), so set $q_{-2} = 0$
$s^{(2)} = 2s^{(1)}$	1.1 1 0 1 0 1	
$2s^{(2)}$	1.1 0 1 0 1	In [–1/2, 1/2) so set $q_{-3} = 0$
$s^{(3)} = 2s^{(2)}$	0.1 0 1 0 1	
$2s^{(3)}$	1.0 1 0 1	$< -1/2$, so set $q_{-4} = -1$
$+d$	0.1 0 1 0	and add
$s^{(4)}$	1.1 1 1 1	Negative,
$+d$	0.1 0 1 0	so add to correct
$s^{(4)}$	0.1 0 0 1	
s	0.0 0 0 0 1 0 0 1	
q	0.1 0 0 $\bar{1}$	Uncorrected BSD quotient
q	0.0 1 1 0	Convert and subtract *ulp*

Steps involved in Radix2 SRT divide are as follows
a. On reset operand x is loaded in. 2 bits are added to it at the start which serves as sign and integer bit. It acts as first partial remainder w [0].

b. Partial reminder is shifted by 2 every iteration.

c. First 3 bits of partial reminder identifies correct quotient which is recoded into 2 bits as q+ and q-.

d. Selected quotient chooses appropriate value of d from {d, 0, -d}.

e. When -d is selected multiplexor chooses inverted version of d & additional 1 required to produce its 2's compliment is added as carry input to the CPA adder.

f. Redundant quotient bit is sent to OTF (On the Fly) converter.

g. When algorithm requires input operands to be of size greater than double precision, delay due to carry propagate adder becomes the critical path, CPA can be replaced by carry save format adder.

3.1 Radix4 Division:

In Radix4 divider algorithm two possibilities exist for redundant quotient digit set with digit set can be {-2, 1,0, 1, 2} or {-3, -2, -, 1, 0, 1, 2, 3}. The case with a=2 has an advantage that 21 multiple of d required are easy to generated whereas with a=3 multiple of 3d is required which can not be generated using shifting of operand d and it needs to be split into (d and 2d) which requires an extra adder in the critical path.

Steps involved in Radix4 SRT divide are as follows:

a. On reset operand x is loaded. 3 bits are added to it at the start which serves as sign and two integer bits. It acts as first partial remainder w[0].

b. Partial reminder is shifted by 4 (left shift by 2) every iteration & producing 2 bits of the quotient according to recurrence formula.

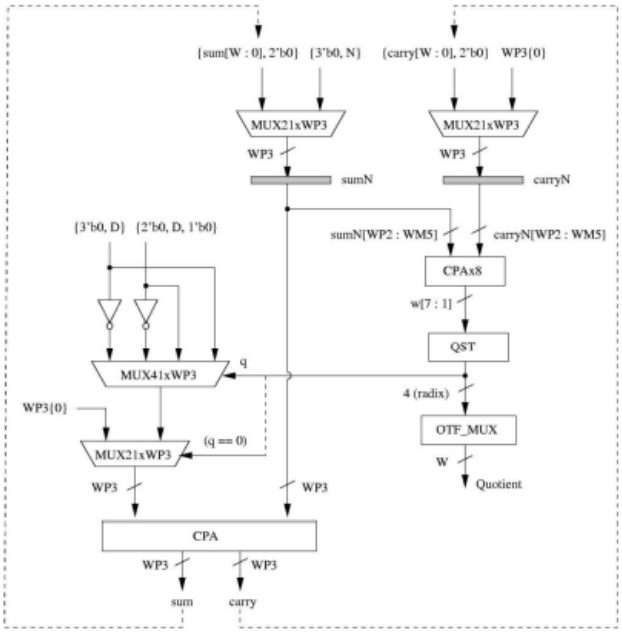
c. First 7 bits of partial reminder and 3 bits of divisor identifies correct quotient which is recoded into 4 bits as q2+, q+, q2- and q- as shown in table6.

d. Selected quotient chooses appropriate value of d from {2d, d, 0, -d, -2d}.

e. When -d or -2d is selected multiplexer chooses inverted version of d & additional 1 required to produce its 2's compliment is added as carry input to the CPA adder.

f. Redundant quotient bit is sent to OTF (On-The-Fly) converter.

g. When algorithm requires input operands to be of size greater than double precision, delay due to carry propagate adder becomes the critical path, CPA can be replaced by carry save format adder.



4. Conclusion:

Examined several modifications to the typical SRT division algorithms, and found, in particular, that simple pre-normalization will allow radix 8 division to be performed with approximately the same quotient selection complexity as radix 4 division, and also that some of the higher radix cases can be implemented with fewer bits in the quotient selection logic by using a borrow-save rather than a carry-save format for the remainder arithmetic.

5. References:

[1] DE. Atkins, “Higher-Radix Division Using Estimates of the Divisor and Partial Remainders,” IEEE Transactions on Computers, vol. C-17, pp. 925-934, October 1968.

[2] M.D. Ercegovac, “A Higher-Radix Division with Simple Selection of Quotient Digits,” Proceedings of the Sixth IEEE Symposium on Computer Arithmetic, pp. 94-98, May 1983.

[3] K. Hwang, Computer Arithmetic: Principles, Arhitecture, and Design, John Wiley & Sons, 1979.

[4] W.M. McAllister and D. Zuras, Hewlett-Packard, “An nMOS 64b Floating-Point Chip Set,” IEEE International Solid-State Circuits Conference, February 1986.

[5] J.E. Robertson, “A New Class of Digital Division Methods,” IRE Tmns. Electronic Computers, vol. EC-7, pp. 218-222, September 1958.

[6] A.L. Sangiovanni-Vincentelli, R.K. Brayton, et. al., Logic Minimization Algorithms for VLSI Synthesis, Kluwer Academic, 1984.

[7] G.S. Taylor, “Compatible Hardware for Division and Square Root,” Proceedings of the Fifth IEEE Symposium on Computer Arithmetic, pp. 127-134, May 1981.