社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－ARC－164（23）
2005／8／4

A New Interpretation of the Non-restoring Method and Some Applications

Hatsuhiko Kato †

A new interpretation will be made on the so-called non-restoring method for division and some applications will be introduced that are enabled by this interpretation. As opposed to the conventional interpretation of the non-restoring method which shifts the partial remainder left, this interpretation shifts the divisor right thus avoiding the overflow of the partial remainder. This interpretation is independent of the handling of the most significant bit of the partial remainder which differs from machine to machine. Moreover, the choice of addition or subtraction at each stage can be done based on the sign of the current partial remainder, not on the sign of the previous partial remainder. This simplifies the division process and eliminates mistakes when simulated on the desk. The sign-extended right shift of the divisor can be always done a priori without the knowledge of the current partial remainder. This enables the parallel operations of the addition/subtraction and divisor shift which has been unfeasible by the conventional interpretation in which addition or subtraction could be chosen only after the left shift of the partial remainder. The results help the better understanding of the algorithm and suggest a possible improvement in the speed of arithmetic operations.

1. Introduction

Conventionally, the non-restoring method for the division has been done in the sequence of the processes which are shown below [1], [2].

(1) The signs of the dividend C and the divisor D (hereinafter C and D) are compared. If they are of opposite signs, addition $C + D$ is performed, whereas subtraction $C - D$ is performed if they are of the same sign. The result of the addition or subtraction will be the initial value of the partial remainder R (hereinafter R).

(2) If the sign of R is opposite to that of C, division is feasible, otherwise the case is division overflow.

(3) If the division is feasible, the signs of R and D are compared. If the signs are the same, R is shifted left and D is subtracted from R. A 1 is assigned to the least significant bit of the quotient Q (hereinafter Q). If the signs are opposite, D is added to R instead of being subtracted after R is shifted left. A 0 is assigned to the least significant bit of Q. This process is repeated for all bits.

(4) The sign of R is compared with the sign of D. If they are of the same sign, adjustment of R is not necessary and 1 is assigned to the least significant bit of Q. If they are of opposite signs, adjustment is made adding D to R and

† Department of Information Science, Shonan Institute of Technology

0 is assigned to the least significant bit of Q.

(5) R is shifted right by the number of bits to compensate for the effect of left shift at each process.

This interpretation of the non-restoring method used to have significance when only a limited hardware was available. Each of the newly obtained bits of Q could be stuffed into the lower part of R which was evacuated by the left shift to save the number of flip flops [3]. However, it has the following shortcomings.

(1) There will be overflow of R when it is shifted left if the most significant two bits are 01 or 10, which means that the absolute value of R is too large to be doubled.

(2) In some CPUs, left shift is performed preserving the most significant bit of R to avoid the change of sign instead of shifting it with all the other bits. In such case, the information of the absolute value of R is lost and the further operations are meaningless.

(3) Subtraction or addition is chosen based on the comparison of the signs of R and D before shifting R left, but is performed after the left shift. To facilitate this, the sign of R before the left shift must be preserved. This can cause confusion at desk simulation and also complicates the hardware.

(4) Additional operation of shifting R right for remainder compensation causes increase in execution time.

2. New Interpretation

To avoid those shortcomings, we introduce another interpretation of the non-restoring method. Here the following assumptions are made.

(1) C and R consist of $2n + 1$ bits including the sign. The final values of D and Q will consist of n+1 bits. Each number, represented by N, is expressed in 2's complement.

(2) The decimal point of a number is placed at the right side of the most significant bit. This means that a number N satisfies $+1 > N \geqq -1$.

The whole processes of the new interpretation are executed as shown below.

(1) Q is expressed as $q_n.q_{n-1}q_{n-2}\ldots q_1q_0$. Its value can be expressed as

$$Q = -q_n + \sum_{i=0}^{n-1} 2^{-n+i} q_i = -q_n + \sum_{i=1}^{n-1} 2^{-n+i} q_i + 2^{-n} q_0 \qquad (1).$$

All digits in Q are not determined prior to division. In order that division is feasible with no overflow, $+1 > C/D \geqq -1$ must hold. Compare the signs of C and D. If they are of opposite signs, addition $C + D$ is performed, whereas subtraction $C - D$ is performed if they are of the same sign. The result of the addition or subtraction will be the initial value of R. If the sign of R is opposite to that of C, division is feasible. Otherwise the case is division overflow. Assuming a formal bit q_{n+1} above the most significant bit q_n. with the values $q_{n+1} = 0$ for addition and $q_{n+1} = 1$ for subtraction, these operations can be uniformly recognized as

$$R_n = C + D(1 - 2q_{n+1}) \qquad (2).$$

(2) If the division is feasible, D is sign-extended and shifted right and the signs of R and D are compared. If they are of the same sign, D is subtracted from R and 1 is assigned to the least significant bit of Q. If they

are of opposite signs, D is added to R instead of being subtracted from R and 0 is assigned to the least significant bit of Q. This process is repeated for all bits. That is

$$R_{i-1} = R_i + D(1-2q_i)\bullet 2^{-n+i-1} \qquad (3)$$

$$(\, i = n, n-1, \cdots, 1\,).$$

(3) The sign of R is compared with the sign of D. If they are of the same sign, adjustment is not necessary for R and then 1 is assigned to the least significant bit of Q. If they are of opposite signs, adjustment of R is necessary. D is added to R and then 0 is assigned to the least significant bit of Q.

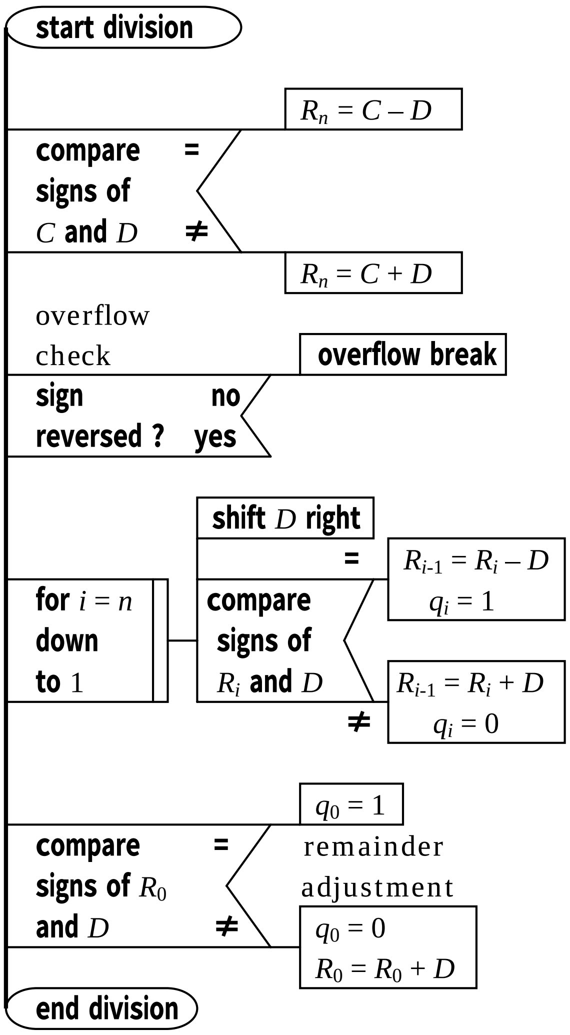


Figure 1. Division Sequence

The overall sequence of processes is shown on Figure 1.

The difference of this interpretation from the previous one is that the addition or subtraction is chosen and performed based on the signs of R and D after the shift operation at each stage. Another difference is that D is shifted right instead of shifting R left. This interpretation can be proven by summing up the formulae (2) and (3) for $i = n, n-1, \dots, 1$.

$$\cancel{R_n} = C + D(1-2q_{n+1})$$

$$\cancel{R_{n-1}} = \cancel{R_n} + D(1-2q_n)\bullet 2^{-1} \qquad i = n$$

$$\cancel{R_{n-2}} = \cancel{R_{n-1}} + D(1-2q_{n-1})\bullet 2^{-2} \qquad i = n-1$$

$$\vdots$$

$$R_0 = \cancel{R_1} + D(1-2q_1)\bullet 2^{-n} \qquad i = 1,$$

$$R_0 = C + 2D(1-2^{-n-1}) - 2D\sum_{i=1}^{n+1} q_i\, 2^{-n+i-1} \qquad (4)$$

is obtained. Moving terms from one side to the other, (4) can be expressed as

$$C = D\{\,2(q_{n+1}-1) + \sum_{i=1}^{n} 2^{-n+i} q_i + 2^{-n}\} + R_0$$

(5).

Regarding the terms in the braces on the right side as Q, (5) corresponds to the general expression of division, the dividend equals the product of the divisor and the quotient plus the remainder. In order that R_0 is qualified as the remainder, $|2^{-n}D| > |R_0|$, that is $(\,2^{-n}D\,)^2 > R_0^2$ must hold. This is a special case for

$$(\,2^{-n+i}D\,)^2 > R_i^{\,2} \qquad (6)$$

$$(\, i = n, n-1, \dots, 1, 0\,),$$

which can be proven by mathematical reduction. The case $i = n$ can be proven since the condition $+1 > C/D \geqq -1$ is already verified at the process (1) and R_n obtained by the process (2) satisfies $D^2 > R_n^2$. For $i = n-1, \dots, 1, 0$,

$$R_{i-1}^2 - (2^{-n+i-1}D)^2 = (R_{i-1} + 2^{-n+i-1}D)(R_{i-1} - 2^{-n+1-1}) \quad (7).$$

Applying the relation (3), the left side of (7) is equal to:

$$R_i(R_i - 2^{-n+i}D) < 0$$

for subtraction ($q_i = 1$)

and $R_i(R_i + 2^{-n+i}D) < 0$

for addition ($q_i = 0$).

Thus the relation (6) holds and accordingly the relations $(2^{-n}D)^2 > R_0^2$ and $|2^{-n}D| > |R_0|$ hold.

This is not a new algorithm for division but only a different interpretation of the well-known non-restoring method. The right-shift of D used to be commonly adopted at an early stage of computer arithmetic, when the comparison method or restoring method was widely used. In another word, this interpretation was inspired by the traditional method of division.

This interpretation has the following merits as compared with the conventional one.

(1) The left shift of R or any other operand is not involved during the processes. Accordingly there can be no liability of overflow.

(2) Avoidance of left shift guarantees the independence of the algorithm from the CPU architectures which differ in the treatment of the most significant bit at left shift.

(3) The choice and performing of addition or subtraction of R and D are done based upon the current status of the signs of R.

3. Numerical Examples

We verify this interpretation with two numerical examples. The value of n is assumed to be 4. Accordingly the dividend C and remainder R consist of 9 bits. The initial divisor D consists of 5 bits and will be sign-extended during the operation and the final number of bits will be 9. The quotient Q is not determined yet but 5 bits result is expected. All numbers are expressed in 2's complement. We verified these examples with a CASL II assembler program.

Example 1

dividend: C = 0.01101001 = 105/256

divisor: D = 1.0101 = –11/16

– divisor: – D = 0.1011 = 11/16

operations	processes
0.01101001 +)1.0101 1.10111001 = –71/256= R_4	opposite sign, addition, $R_4 = C + D$ sign reversed, division feasible
1.10111001 +)0.01011 0.00010001 = 17/256 = R_3	shift D right, same sign, $q_4 = 1$ subtraction $R_3 = R_4 - D$
0.00010001 +)1.110101 1.11100101 = –27/256 = R_2	shift D right, opposite sign, $q_3 = 0$ addition, $R_2 = R_3 + D$
1.11100101 +)0.0001011 1.11111011 = –5/256 = R_1	shift D right, same sign, $q_2 = 1$ subtraction, $R_1 = R_2 - D$
1.11111011 +)0.00001011 0.00000110 = 6/256 = R_0	shift D right, same sign, $q_1 = 1$ subtraction, $R_0 = R_1 - D$

The signs of R_0 and D oppose. Remainder adjustment needed. $q_0 = 0$

$R_0 = R_0 + D$ = 0.00000110 + 1.11110101 = 1.11111011 = –5/256 Q = 1.0110 = –10/16

This result is verified as

105/256 = (–10/16)(– 11/16) –5/256

Example 2
dividend: C = 1.10100010 = –94/256
divisor: D = 1.0011 = –13/16
– divisor: $– D$ = 0.1101 = 13/16

operations	processes
1.10100010 +)0.1101 0.01110010 = 114/256= R_4	same sign, subtraction, $R_4 = C – D$ sign reversed, division feasible
0.01110010 +)1.10011 0.00001010 = 10/256 = R_3	shift D right, opposite sign, $q_4 = 0$ addition $R_3 = R_4 + D$
0.00001010 +)1.110011 1.11010110 = –42/256 = R_2	shift D right, opposite sign, $q_3 = 0$ addition, $R_2 = R_3 + D$
1.11010110 +)0.0001101 1.11110000 = –16/256 = R_1	shift D right, same sign, $q_2 = 1$ subtraction, $R_1 = R_2 – D$
1.11110000 +)0.00001101 1.11111101 = –3/256 = R_0	shift D right, same sign, $q_1 = 1$ subtraction, $R_0 = R_1 – D$

The signs of R_0 and D coincide. Remainder adjustment not needed. $q_0 = 1$
R_0 = 1.11111101 = –3/256
Q = 0.0111 = 7/16
This result is verified as
–94/256 = (–13/16)(7/16) –3/256.

4. Extensions and Applications

4.1 Extended Binary Expression

Before discussing on the application to SRT method, we verify that this interpretation holds also for the extended binary expression, which permits $\bar{1} = -1$ as well as 1 and 0 for the bits q_i's of the quotient. In addition to that, we assume the most significant bit q_{n+1} , which is determined by the first step to check the division feasibility. The values for q_i's are $\bar{1}$ when C and D (or R_i) are of opposite signs and 1 when C and D (or R_i) are of the same sign. The quotient is obtained as $q_{n+1}.q_nq_{n-1}\ldots q_2q_1$ instead of $q_n.q_{n-1}q_{n-2}\ldots q_1q_0$. The examples in the chapter 3 can be recognized as:
Example 1:

$q_{n+1}.q_nq_{n-1}\ldots q_2q_1 = \bar{1}.1\bar{1}11 = \bar{1}.0111$ or $\bar{1}.0110$

Example 2:

$q_{n+1}.q_nq_{n-1}\ldots q_2q_1 = 1.\bar{1}\ \bar{1}11 = 0.0111$

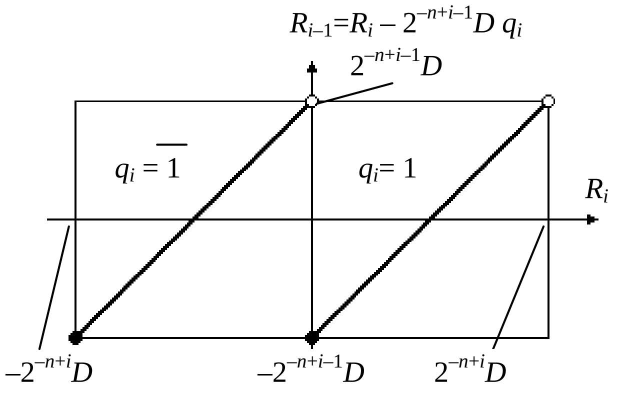
4.2 SRT Method

Applying SRT method to Example 2, we can obtain the result:

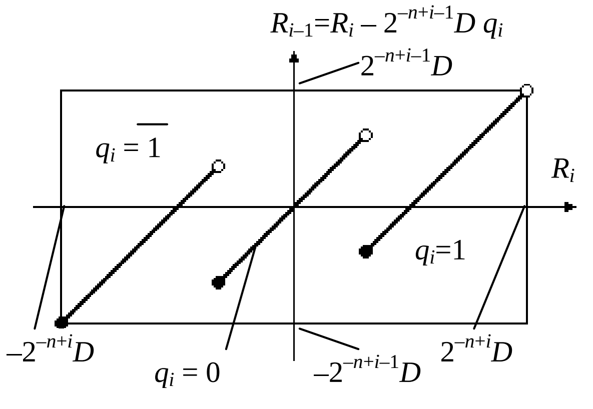
operations	processes
$\boxed{1.10}$100010 +)0.1101 0.01110010 = 114/256= R_4	$\boxed{110}$ same sign, subtraction, $R_4 = C – D$, $q_5 = 1$ sign reversed, division feasible
0.$\boxed{011}$10010 +)1.10011 0.00001010 = 10/256 = R_3	$\boxed{011}$ opposite sign, addition, $R_3 = R_4 + D$ $q_4 = \bar{1}$
0.0$\boxed{000}$1010 = 10/256 = R_2	$\boxed{000}$ no operation, $q_3 = 0$
0.00$\boxed{001}$010 +)1.1110011 1.11110000 = –16/256 = R_1	$\boxed{001}$ same sign, subtraction, $R_1 = R_2 – D$, $q_2 = \bar{1}$
1.111$\boxed{100}$00 +)0.00001101 1.11111101 = –3/256 = R_0	$\boxed{100}$ same sign, subtraction, $R_0 = R_1 – D$ $q_1 = 1$

$Q = 1.\bar{1}0\bar{1}1 ==$ 0.0111= 7/16

Here we examined the leading 3 bits of R_i at each process. The "leading 3 bits" means the most significant 3 bits excluding the extended sign bits designated by boxes in the tables. Accordingly, they consist of a single sign bit and two numerical bits. If they are $\boxed{111}$ or $\boxed{000}$, the absolute value of R_i is less than 2^{-n+i-1}. Addition or subtraction is not performed and only D is shifted right. The carry propagation to the extended sign bits is not necessary for the SRT method or non-restoring method since the carry needs only be copied from the sign bit to the higher bits. The Robertson's diagrams for the non-restoring method and SRT method are shown on Figure 2.



(a) Non-restoring method



(b) SRT method

Figure 2. Robertson's Diagrams

4.3 Application to Parallelism

This interpretation is well-suited for parallel high speed division through quotient bits prediction. The shifted results of D are obtained a priori and accordingly the next addition and subtraction can be initiated before the current operation is completed. Depending on the result of the current operation, either one of sum or difference is adopted as the result of the next operation. This overlapping of successive operations can be applied multiply.

4.4 Educational Aspects

The avoidance of left shift prevents overflow of R_i or loss of sign bits. Addition or subtraction (or even no operation) can be chosen based on the result of shift instead of the sign of R_i before the left shift. The right shift of R_0 is unnecessary. These points eliminate machine dependence and unnecessary confusions in the process of understanding the algorithmic principles.

5. Conclusions

We introduced and proposed a new interpretation of the non-restoring method for division. As the result, a new horizon of applications and education emerged. A similar aspect is adopted also in recent works [4]. Extension and application to higher-radix division or pipelining can be feasible.

References

[1] Department of Computer Science and Electrical Engineering, University of Maryland: Lecture 20, Multiply and divide, www.cs.umbc.edu/~squire/cs313_ l20.html (2004).

[2] T.C.Bartee: Computer Architecture and Logic Design, McGraw-Hill, Inc (1991).

[3] D.A.Patterson, J.L.Hennessy: Computer Organization & Design 2nd Edition, Morgan Kaufmann (1998).

[4] N.Takagi et al.: A Hardware Algorithm for Integer Division. Proc. 17th IEEE Symposium on Computer Arithmetic (2005).