International Journal of Computer Applications (0975 – 8887)
Volume 97– No.5, July 2014

Enhanced Computational Algorithm of Binary Division by Comparison Method

Vandana
Assistant Professor
Computer Science Department, SRMSWCET
Bareilly, India

ABSTRACT

Binary division is the basic operation performed by arithmetic circuit. It is simpler than the decimal division because the result always produced in either 1 or 0. All the values of dividend, divisor, quotient and remainder are in 1's or o's form. There are number of binary division algorithms are available as restoring method, non restoring method, division by XOR logic operation and SRT division and comparison method. This paper presents a new concept of the comparison division method. The comparison division algorithm provides high speed computation work and increases' the system performance.

Key Terms

Binary division, Restoring concept, Comparison method and Non restoring method, Magnitude comparator

1. INTRODUCTION

Binary division is a procedure that shows how many times the divisor D divides the dividend A. In division operation the divisor is subtracted from the dividend, if dividend is greater than the divisor and the quotient bit is set to 1. Computer system performs the subtraction operation in 2's complement form. Division is equivalent to repeat subtraction of the divisor from the dividend until the quantity left is smaller in magnitude than the divisor. The division algorithm rules are [1]:

1-if the portion of the dividend above the divisor is greater than or equal to the divisor:

- then subtract the divisor from the dividend and
- put the result of the subtraction "1" to the right end of the quotient.
- If the result is zero, place a "0" to the right end of the quotient.

2- Shift the divisor one place right.

3-repeat until dividend is less than the divisor and quotient is correct. The dividend is the remainder.

This process, if done straight forwardly, is very time consuming. It is substantially speeded if the most significant digits of the divisor and dividend are aligned before the first subtraction, and the divisor then shifted to the right one position whenever the partial remainder becomes smaller than the divisor before shifting. One shift may be necessary before any subtraction

There are two kinds of division algorithms, digit recurrence division and division by convergence. Digit recurrence division is simple and has less complexity than convergence division algorithm. There are three methods available for division algorithms

1-Restoring method (additional restoration cycles for the restoration).

2-Comparison method

3-Non restoring method (restoration cycles are removed)

The non-restoring division algorithm is the fastest among the digit recurrence division methods because there is no need of restoration cycles.

2. RESTORATION METHOD

The division algorithm is very time consuming if it is done straight forwardly, because we need to compare the remainder with the divisor after every subtraction. The restoring division algorithm is the simplest of the three digit recurrence division methods. In restoring division, subtraction continues until the sign of the partial remainder changes; the change causes an immediate addition of the divisor and a corresponding decrement of the accumulating quotient, before the right shift. The restoring division performs two additions for each iteration when the temporary partial remainder is less than zero and this results in making the worst case delay longer.

2.1 Characteristics of Restoring Method:

1. Subtraction continues until the sign of the partial remainder changes.
2. Causes an immediate addition of the divisor
3. Additional restoration cycles for the restoration
4. In restoring method after operation value sign is change after its previous value.
5. Causes an immediate addition of the divisor

3. NON-RESTORING METHOD

This non-restoring method [2] provides high computational speed, because only one addition is performed in per iteration. In this a quotient set {+1,-1} is required. The quotient digit +1 is used for subtraction and -1 is used for addition [3].

3.1 Characteristics of non restoring method

1. The sign change causes a shift followed by one or more additions until the sign changes back.
2. The sign change causes a shift followed by one or more additions' until the sign changes
3. Negative radix, and would require a conversion routine to restore the quotient to normal form. [2]
4. Only one subtraction or addition is taken at each step, setting sign bit is 0 in the quotient if both the partial remainder and the divisor are of the same sign (opposite sign).
5. Sequence counter's value is proportional to the operand length (divisor).

4. COMPARISON METHOD

By using comparison method, a division algorithm can be performed in two ways:

1- Division with micro-operations
2- Division with magnitude comparator

4.1 Division with micro-operation

In the comparison method the divisor and dividend are compared prior to the subtraction operation. If divisor is greater than equal to dividend, the divisor is subtracted from dividend .if dividend is less than divisor nothing is done only the partial remainder is shifted left and the number s are compared again. The comparison operation can be determined prior to the subtraction by inspecting the end carry out of the addition operation through parallel adder. Consider that dividend is stored in register A and divisor is stored in register B. the comparison operation can be done with EA<-A+B+1. If E=0, it shows that the dividend is less than divisor and if E=1, It shows that the dividend is greater than the divisor. In the comparison method A and B is compared prior to the subtraction operation. If A>=B, b is subtracted from A. if A<B nothing is done. The hardware implementation consists of three parts: output carries , partial remainder and quotient bits.

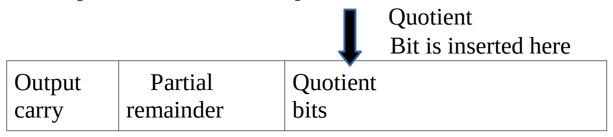


Fig (1) Program division

The end carry flip flop is used to store the carry bit after the addition operation. The comparison operation is done by inspecting the end carry. To compute the results firstly verify the register length because there may be exist an overflow condition. So it is assumed that the dividend part should be less than the divisor. Comparison method provides the fast speed for division operation because it saves the restoration time for partial remainder [1] [4].

4.1.1 Algorithm for comparison method:

1. First check the divide overflow condition by adding 2's complement of register B to register A. and the result is transferred to the E and A. and quotient bit is set to 0.

2. If the end carry flip flop E contains the 0 -value, only left shift operation is performed

3. If E contains 1- value the subtraction and left shift operation is performed. And the quotient bit is set to 1.

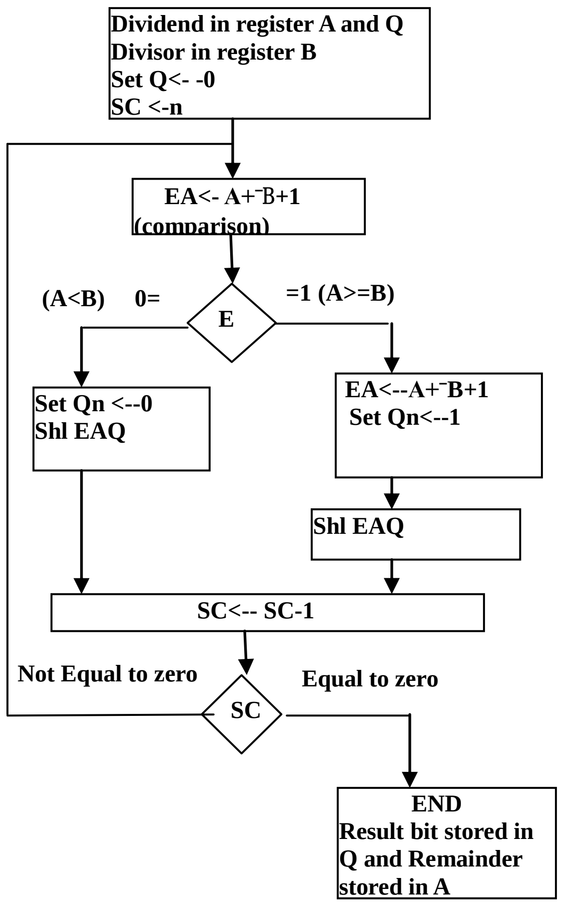


Fig (2) Flow chart for comparison algorithm

In this comparison method only two types of operation addition and left shift are required. Software and hardware aspects can be improved by replacing some divisions by shifts/adds/subs for optimizing compiler and hardware dividers may be replaced by simpler adders for VLSI circuits.

4.1.2 Characteristics of comparison algorithm:

1. Reduce execution time
2. No additional set is required for division operation that is no negative radix is used.
3. Only two micro-operations: add and left shift are required in each step.
4. Less circuitry is required.

This method gives better performance because it provides fast speed. But there may be insufficient partial remainder may produce because there is no restoration of the partial remainder in the case when dividend is less than divider.

5. PROPOSED WORK - COMPARISON ALGORITHM USING MAGNITUDE COMPARATOR

Comparing two binary numbers for equality is a commonly used operation in computer system and device interface. The hardware in the comparator can be reduced by implementing only two outputs, and the third output can be obtained using these two outputs. For example, if we have the LT and GT outputs, then the EQ output can be obtained by using only a NOR gate. Thus, when both the GT and LT outputs are zeros, then the 3rd one (i.e. EQ) is a ‘1. Another approach for division algorithm is that if we first compare the magnitudes of both the numbers by using magnitude comparator. This method saves the computational time in the case when A=B, and set the quotient bit 1, so there is no need to perform any micro-operation. In the case when A>B, first we make the dividend less than divisor with the help of normalization process so that the necessary condition can be obtained. The addition and subtraction micro-operation can be performed in the case when A>B and A<B. The magnitude comparator circuit saves the initial computational time for comparison operation to provide the fast speed. In the case of A>B and A<B it performs the necessary addition, subtraction and shift micro-operations as in the case of restoring and non restoring [5]. The advantage of this concept is that it does not perform the micro-operations for A=B to generate the quotient bit and there is no need to check the sequence counter value. But in the case of restoring and non-restoring method the initial equivalence condition is checked on the basis of micro-operation and we have to perform multiple micro-operations to produce the quotient bit even in the case of when A=B.

5.1 Characteristics of magnitude comparator comparison algorithm:

1-Initial comparison computational time is saved

2- The speed is fast as compare to other division methods.

3-The cost is high because of the addition of the addition of the combinational circuit.

5.2 Algorithm steps:

1- Check the sufficient condition for the division operation i.e dividend should be less than divisor for fixed and floating-point data representation.

2- In case of floating-point data representation the normalization process is used to make the dividend smaller than divisor.

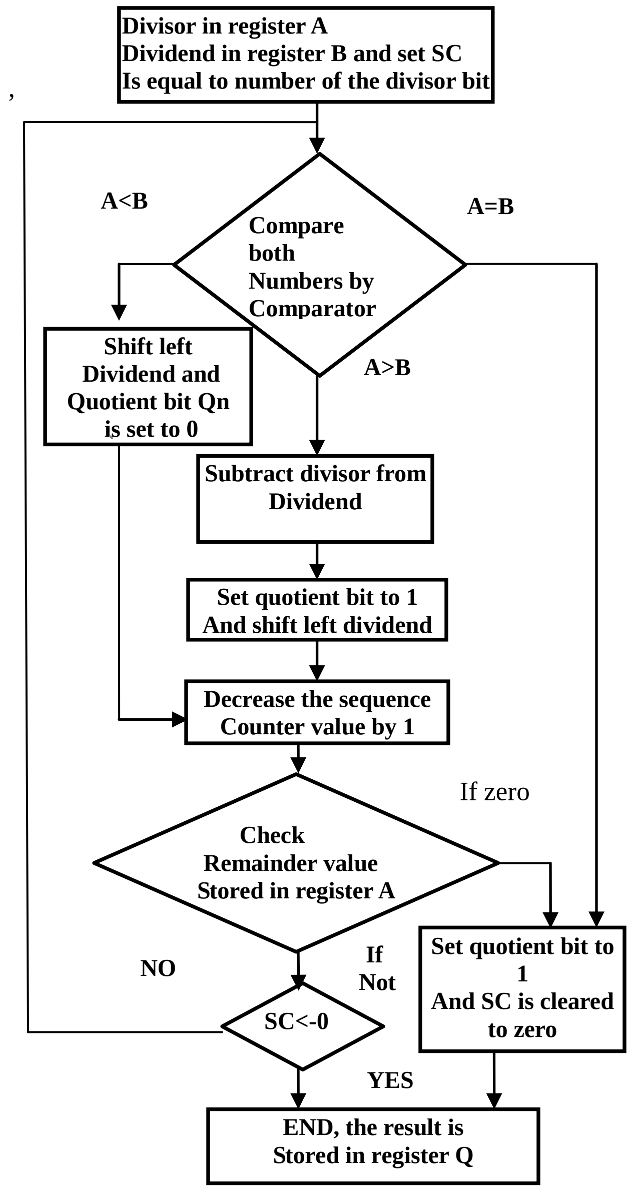
3- In this method bit-by-bit operations are performed. But in case of A=B, the quotient bit is set to 1.

4- If A>B, than the divisor is subtracted by dividend.

5- If A<B, than the dividend is shifted to the left and set quotient bit is equal to 0.

6- if A=B, the divisor is subtracted and after subtraction operation if remainder is zero than set quotient bit is equal to 1 and there will be no bit -wise operation is performed, and process will be ended.

So by collaborating the magnitude comparator with micro-operations the speed of the processing is increased thus it reduce the execution time because of the less number of computer instructions. This method gives the high-performance computing algorithms which is essential to meet the expanding demand for computation .The objective of the Division of Computational Algorithms is to achieve the levels of performance and reliability required for fundamental computational science applications [6]


Divisor in register A
Dividend in register B and set SC
Is equal to number of the divisor bit
A<B
A=B
Compare both Numbers by Comparator
Shift left Dividend and Quotient bit Qn is set to 0
A>B
Subtract divisor from Dividend
Set quotient bit to 1 And shift left dividend
Decrease the sequence Counter value by 1
If zero
Check Remainder value Stored in register A
NO
If Not
SC<-0
Set quotient bit to 1 And SC is cleared to zero
YES
END, the result is Stored in register Q

Fig(3) Flow chart for division algorithm by comparison algorithm using magnitude comparator

This flow chart (fig-3) gives the specification of binary division operation. This method is very simple because of the less number of instructions. In this algorithm Q is the quotient register. Initially dividend is stored in double register A and Q.

6. CONCLUSION

In this paper we propose a theoretical concept by giving an algorithm for binary division by using magnitude comparator and micro-operations. This method provides a better combination for both hardware and logical implementation to increase the speed of the binary division algorithm and generates the accurate quotient bits.

7. REFERENCES

[1] Arithmetic operations in a binary computer by Robert F.Shaw

[2] An algorithm for non-restoring algorithm by S. Sonycl, Tata Institute of Fundamental Research Bombay, India

[3] Fast 32-bit Division on the DSP56800E Minimized non restoring division algorithm by David Baca

[4] D. Banerji, T. Cheung, and V. Ganesan, "A High-speed Division Method in Residue Arithmetic, "Proceedings of 5th IEEE, Symposium on Computer Arithmetic, Michigan, USA, 1981, pp. 158-164

[5] A Protected Division Algorithm , Published in P. Honey man, Ed., Fifth Smart Card Research and Advanced Application Conference (CARDIS '02), pp. 69–74, Usenix Association, 2002. Marc Joye and Karine Villegas

[6] **J**. H. Yang, C. C. Chang, and C. Y. Chen, "A High- Speed Division Algorithm in Residue Number System Using Parity-Checking Technique,"

[7] Binary division and square-rooting using Gray code by CK Yuen

[8] Improved Algorithms for Non-restoring Division and Square Root by Kihwan Jun, B.S.E.E., M.S.E. M.S.E.E

[9] **A** Division Algorithm Using Bisection Method Residue Number System by Chin-Chen Chang and Jen-Ho Yang

[10] Binary division and square-rooting using Gray code by CK Yuen

[11] VHDL Implementation of Non Restoring Division Algorithm Using High Speed Adder/Sub tractor Sukhmeet Kaur1