International Journal of Computational and Applied Mathematics.
ISSN 1819-4966 Volume 12, Number 3 (2017), pp. 691-697
© Research India Publications
http://www.ripublication.com

A Modified Synthetic Division Algorithm

S. Subha
Department of Information Technology,
School of Information Technology and Engineering,
Vellore Institute of Technology Vellore-632014, India.

Abstract

Division using synthetic method is proposed in this paper. The algorithm takes eight bit number to be divided into four bit number in signed notation. The given inputs are converted into polynomial representation of powers of two. This is done from lookup table for the divisor. The dividend is given in the polynomial notation as input. The synthetic division is applied to this representation. Registers are used to store the coefficients and results of the synthetic division. The remainder is obtained from the division. If it is negative, it is converted to positive value by adding the divisor. The quotient is calculated by taking the polynomial representation of the result coefficient. The quotient is reduced by appropriate value if the remainder is negative and correction is applied to it. The proposed algorithm is simulated using Quartus2 toolkit. A performance improvement of 7% with area degradation of 40% and increase in power consumption of 8% is observed when compared with in-built division algorithm in Quartus2 Toolkit. The proposed algorithm performance is less when compared with restoring and non-restoring division algorithms.

Keywords: Division, Non-restoring division, Performance, Restoring division, Synthetic division, Unsigned numbers,

1. INTRODUCTION

Integer arithmetic operations in computer are addition, subtraction, multiplication and division. Division operations are defined by algorithms. The two common algorithms are restoring division and non-restoring division [1]. Many applications require division operations like FFT etc [3]. In division operation, 2n-bit dividend is divided into n-bit divisor giving n-bit quotient and n-bit remainder. The remainder is positive in restoring and non-restoring division algorithms. The authors in [5] suggest improvement for restoring division algorithm. The author in [7] propose methods to improve non-restoring division. Usually in restoring and non-restoring division after n steps the result is obtained. A method to perform division by constants is proposed in [4].

This paper proposes division using synthetic division method. The dividend is 2n-bits and divisor n-bits. The divisor is factorized into linear factors. The dividend is expressed as polynomial . The division is done taking the co-efficients of the dividend and performing synthetic division using the divisor. The number of additions is equal to the degree of the polynomial the dividend is expressed. The quotient is 2n bits and remaider is n bits.

The proposed model is simulated with Quartus2 Toolkit. The input is eight bit dividend and four bit divisor. A performance improvement of 7% with area degradation of 40% and increase in power consumption of 8% is observed when compared with in-built division algorithm in Quartus2 Toolkit. The proposed algorithm performance is less when compared with restoring and non-restoring division algorithms.

The rest of paper is organized as follows. Section 2 gives mathematical background, section 3 proposed algorithm, section 4 simulations, section 5 conclusions followed by references.

2. MATHEMATICAL BACKGROUND

Consider the synthetic division method [6]. In this method a polynomial is divided into binomial. The steps in this method for dividing polynomial with (x-c) where c is a constant is given below.

1. Write the coefficients of the numerator in descending order of exponent
2. Bring down leading coefficient to the bottom row.
3. Multiply c by value just written in bottom row.

4. Add the result of step 3 to next exponent in the input
5. Continue steps 3-5 till the last exponent i.e. exponent zero
6. The result is the polynomial of degree n-1 with first n-1 coefficients and remainder is the last term of the result in the bottom row.
7. Stop

Thus to divide $ax^3 + bx^2 + cx + d$ by x-k the following pattern is used

k	a	b	c	d
		ka		
	a	b+ka		
				Remainder

The first row consists of k and the coefficients of the dividend. In the vertical pattern terms are added. The diagonal terms are multiplied by k.The remainder can be negative in this division and greater than k. The following gives the example of synthetic division.

Dividend $x^4 - 10x^2 - 2x + 4$ Divisor x+3

-3	1	0	-10	-2	4
		-3	9	3	-3
	1	-3	-1	1	1
					Remainder

Answer is quotient = $x^3 - 3x^2 - x + 1$ Remainder : 1

3. PROPOSED MODEL

Consider the division of polynomial with binomial. Division is performed using synthetic division. Consider the division of eight bit number into four bit number. Both the numbers are signed. The following are the steps to perform the division.

Algorithm MODIFIED_SYNTHETIC_DIVISION: Given dividend of eight bits and divisor of four bits, this algorithm performs the division using synthetic division concept. The result is eight bit quotient and four bit remainder.

1. Start
2. Express the divisor as the sum of power of two and constant. This can be done from the look up table.
3. Express the dividend as combinations of sum , differences of powers of two and constant. This is assumed to be given as algorithm input.
4. Put the power of two in the denominator as baseval, a variable say y.
5. Express the numerator in terms of y

6. Let the resultant expression be c/(y-d), c is polynomial in y.
7. Perform synthetic division of c/(y-d). Keep track of number of times a non-zero value is present in input, say count, in this process to the various coefficients, k.
8. Express the result coefficients in terms of baseval. This is quotient q.
9. Let the remainder be r. Let z = baseval + d

   a. If the absolute value of r is less than baseval and r is positive, stop.

   b. if the absolute value of r is less than baseval and r is negative r=r+z. q=q-1. Stop

   c. If r is negative, t = absolute value of r. If r> z, r=k*z + r, q = q-k. Stop.

   d. If r is positive and greater than z, r = r-kz, q = q+k. Stop.

10. Quotient = q, Remainder =r
11. Stop

The algorithm makes the remainder positive unlike synthetic division algorithm. This is done in step 9 of the algorithm.

The time complexity of above algorithm is equal to the order of degree of polynomial which is seven in this particular case. For storing the polynomial coefficients, the results of the modified synthetic division as described in MODI_SYNTHETIC_DIVISION, separate registers are required. Separate registers to store baseval, z, k are required. The algorithm is scalable. The total time is O(degree of polynomial+constant). The correction performed in step 9 of the algorithm, leads to correct solution.

4. SIMULATIONS

The proposed model is simulated in Quartus 2 Toolkit. The details of the configurations are given in Table 1. The inputs are eight bit dividend, four bit divisor.

Table 1. Simulation Parameters

Parameter	**Value**
Processor Family	Cyclone 2
Package	FBGA
Pin Count	484
Speed grade	Fastest
Target Device	Auto select device by Fitter

There are sixteen combinations of the divisor. The Table 2 gives the resolution of the divisor into sum/difference of power of two. It is observed that difference expressions give quicker less complicated results than sum expressions.

Table 2. Divisor Resolution

Value	**Resolution**
1	Not applicable
2	Shift right once
3	2+1 or 4-1
4	Shift right twice
5	4+1 or 8-3
6	8-2
7	8-1
8	8, shift right three times
9	8+1
10	8+2
11	8+3
12	8+4
13	16-3
14	16-2
15	16-1

The dividend has eight bits and has 256 combinations. It is assumed that the user resolves the dividend into polynomial in 2 based on the choice of the divisor resolution. . The proposed model is compared with in-buit division algorithm in the Quartus 2 Toolkit. As the quotient is eight bits, the proposed algorithm is compared with division algorithm of input of 16-bit dividend and 8-bit divisor. The simulation results are shown in Table 3.

Table 3. Simulation Results

Parameter	**Traditional Algorithm**	**Proposed Algorithm**	**%improvement**	**Non-restoring**	**Restoring**
Area(#slices /total slices)	233/14448	326/14448	-39.91%	144/14448	136/14448
Power	71.43mW	77.74mW	-8.833%	71.43mW	71.53mW
Timing	54.095ns	50.099ns	7.387%	35.413ns	32.186ns

As seen from Table 3, a performance improvement of 7% with area degradation of 39% and increase in power consumption of 8% is observed when compared with in-built division algorithm in Quartus2 Toolkit. The proposed algorithm performance is less when compared with restoring and non-restoring division algorithms. For the restoring and non-restoring division algorithms, the dividend is 16 bits and divisor 8 bits. If the performance of proposed algorithm considers the fact that in the result register, bits 7 and 6 are always zero for the chosen problem size, the timing is 38.393ns which is about 29.026% improvement.

5. CONCLUSION

An algorithm to perform division using synthetic division concept is proposed in this algorithm. The input is 2n bit dividend, n bit divisor with n=4. The result is 2n bit quotient and n bit remainder. The algorithm expresses the dividend and divisor in polynomial of powers of two. The synthetic division is performed. The remainder is corrected to be positive with proper adjustment in the quotient. The proposed model is simulated with Quartus2 toolkit. The proposed model is compared with in-built division algorithm in the toolkit. A performance improvement of 7% with area degradation of 40% and increase in power consumption of 8% is observed when compared with in-built division algorithm in Quartus2 Toolkit. The proposed algorithm performance is less when compared with restoring and non-restoring division algorithms. If the performance of proposed algorithm considers the fact that in the result register, bits 7 and 6 are always zero for the chosen problem size, the timing improvement of 29.026% is observed.

REFERENCES

[1] Behrooz Parhami, Computer Arithmetic: Algorithms and Hardware Design, Oxford University Press, 2000

[2] Israel Koren, Computer Arithmetic Algorithms, Prentice Hall, NJ 1993

[3] Josue Saenz S, Juan J. Raygoza P., Edwin C, Susana Ortega Cisneros, Jorge Rivera Dominguez, FPGA Design and Implementation of Radix-2 Fast Fourier Transform Algorithm with 16 and 32 Points, Proceedings of ROPEC 2015 (FFT)

[4] Li, R, S,.Y, Fast Constant Division Routines, IEEE Transactions on Computers, Vol. 34, No. 9, pp. 866-869, 1985

[5] Nitish Aggarwal, Karthik Assoja, Saurabh Sekhar Verma, Sapna Negi, An Improvement in Restoring Division Algorithm (Needy Restoring Division Algorithm), Proceedings of 2nd International Conference on Computer Science and Information Technology, 2009, pp. 246-249

[6] Rajesh Pandey, A Textbook of Engineering Mathematics, Vol. 1, WordPress Publications, First Edition, 2010

[7] S Subha, An Improved Non-Restoring Algorithm, IJAER, Vol. 11, No.8, 2016, pp. 5452-5454