Survey on Implementation of IEEE754 Floating Point Number Division using Vedic Techniques

[1]Rajani M, [2]S Sridevi
[1]M.Tech,[2]Asst. Professor
[1]Electronics and Communication
[1]CMRIT, Bangalore, Karnataka

***Abstract* - Usually the processor arithmetic unit processes arithmetic operations, in that divider design is more critical as it consumes more area and power compared to other arithmetic operations.Efficient and compact Divider design has always been a challenging task. IEEE 754 floating point representations are considered as a computer storage format.This paper presents different Vedic division techniques which can be implemented in IEEE 754 floating point division format and also to study and compare the effect of various critical parameters in terms of power consumption and area utilization against conventional approaches.Vedic methods proven to be encouraging since this allows an easy and efficient way of computing various arithmetic computations.**

***Index Terms* - IEEE 754, Vedic division.**

I. INTRODUCTION

Vedic Mathematics uses the techniques for solving mathematical problems quickly and easily. Vedic mathematics transforms the difficult calculations into simpler. Vedic mathematics is having different methods for calculation of basic operations like division and multiplication. Nowadays CPUs are working at higher frequencies with the miniaturization of the size of a transistor. In CPU ALU is one of the most critical block. Hence it is essential to have fast and efficient ALU. Among the mathematical calculations the most time Consuming block is division block. Vedic mathematics provides algorithms to simplify the division operations. IEEE 754 standard supports two Types of floating point number representations as follows

1. Single precision floating point representation
2. Double precision floating point representation

Sign	Exponent=Real Exponent+bias	Fraction

Fig.1 IEEE 754 floating point standard [5]

Single precision floating point representation contains 32 bits. Among those bits, 32th bit represents sign (1bit wide), 31-23 bits represents Exponent bits (width is 8 bits), 22-00 bits represents fraction bits (width is 23 bits). Bias for the single precision floating point representation is 127.Double precision floating point representation contains 64 bits. Among those bits, 64th bit represents Sign (1bit wide), 63-52 bits represents Exponent bits (width is 11 bits), 51-00 bits represents fraction bits (width is 52 bits) .bias for double precision floating point representation is 1023 .When compared to the single precision, double precision the extra bits increases not only the precision but also the range of magnitudes that can be represented [5].

II. CONVERSION OF FLOATING POINT TO IEE 754FLOATING POINT STANDARDS

Initially the given floating point number is converted into normalized binary scientific representation. Then the given number is converted into (1. Fraction) $*2^{Exponent}$ form. Here (1. Fraction) represents the Mantissa. Consider 3. 5 as a floating point number, then normalized binary scientific representation of the given number is $1.11*2^1$.

1.11 represents mantissa, 1represents the exponent (power of 2), 11represent the fractional part (bits after the decimal point)

Single precision representation of the given number 3.5:

The given number is positive, hence sign bit is '0', exponent of single precision floating point standard=1 (exponent) +1111111 (base in binary) =10000000. Fraction is 11(width of 2 bits) but for a single precision fraction contains 23 bits and remaining bits are filled with zeros.

Single precision representation of 3.5 is 01000000011000000000000000000000.

For the double precision exponent is 10000000000.

Finally double precision representation of 3.5 represented as follows

0100000000001100000000000000000000000000000000000000000000000000.

III. IEEE 754 Floating point division

The sign of quotient can be found from the Fig. 2 shown below. If A (numerator) & B (denominator) are two numbers and "0" represents positive sign. "1" represents negative sign.

Sign of A	Sign of B	Sign of quotient (A/B)
0	0	0
0	1	1
1	0	1
1	1	0

Fig.2 Representation of the sign of the quotient

The sign of the quotient is XOR of the sign of the numerator and denominator. Exponent of the quotient numerator exponent is subtracted from denominator exponent. Fractional bits of quotient are obtained from a division operation which needs to be performed on mantissas of both numerator and denominator and finally normalization of a floating point quotient is done.

Example of single precision division

3.5÷ --1.5=--2.333333.....
Single precision floating point representation of 3.5 is 01000000011000000000000000000000
Single precision floating point representation of 1.5 is 10111111111000000000000000000000
Sign of quotient 0xor1=1
Exponent of quotient =Exponent of A-Exponent of B+bias $(127)_2$ =10000000
Fraction of quotient =mantissa (A) /mantissa (B) =1.fraction=1.11÷1.1=1.00101010101010101010101
Single precision floating point representation of 3.5/1.5 is 11000000000101010101010101010101

Example of double precision division

3.5÷--1.5=--2.333333.....
Double precision floating point representation of 3.5 is
0100000000001100000000000000000000000000000000000000000000000000
Double precision floating point representation of 1.5 is
1011111111111000000000000000000000000000000000000000000000000000
Sign of quotient 0xor1=1
Exponent of quotient =Exponent of A-Exponent of B+bias $(1023)_2$ =10000000000
Fraction of quotient =mantissa (A) /mantissa (B) =1.fraction=1.11÷1.1=1.00101010101010101010101
Double precision floating point representation of 3.5/1.5 is
1100000000010101010101010101010101010101010101010101010101010101.

Single precision floating point division operation performed in 32 bits. During division operation, binary division operation performed in 24 mantissa bits. Double precision floating point division operation performed in 64 bits. During that binary division operation performed in 53 mantissa bits. Because of more number of bits, the complexity also increases. Vedic mathematics provides algorithms to simplify the division. Three sutras can be directly applied for binary division. Nikhilam, Paravartya, Dwajanka (direct method or flag method) can be applied for binary division [5].

IV. DIFFERENT VEDIC DIVISIONS FOR BINARY DIVISION

There are 5 sutras can be applied for decimal division. Among those sutras three sutras can be applied for binary division these binary divisions can be applied to ieee754 floating point standards. Floating point division output obtained in terms of fractions so there is no need to calculate remainder. To obtain quotient in terms of fraction consider reminder part as after decimal point bits to get more accurate output pad more number of zeros to the right side of numerator bits[6].in this paper it shows how to perform binary division using Vedic sutras

1. Nikhilam

2. Dwajanka

3. Paravartya

The process of performing Nikhilam division

If A, B are the numerator and denominator. 'N' represents' number of bits of the denominator. In the process of Nikhilam division, first it is needed to find the Deficit bits. The deficit is 10^N subtracted from the denominator.
The total number of numerator bits are separated for number of bits of the quotient and number of bits of the remainder.
Remainder bits =denominator bits (1)
The number of quotient bits = number of numerator bits – number of denominator bits (2)

Need to keep the 1st digit of dividend below the last row under first digit (Q_1). Multiplication is performed as first digit Q_1 is multiplied by D_1, D_2.The products written below the 2nd digit and third digit P_1, P_2 and add only second digit column digits. Second digit's Colum digits sum obtained is Q_2.Multiplicaton is done for the second digits' column bits sum (Q_2) by D_1, D_2 thus P_3, P_4 are .The products under 3rd, 4th digits are written and only 3rd digit's Colum bits are added. This process continued till the last digit. Q_1, Q_2.... represents quotient bits, D1, D2..... represents deficit digits and p1, p2... represents partial products. Last quotient bits are considered for the remainder as R1, R2... RN [4].

Binary division by Nikhilam

For example 81÷9. The binary equivalent of 81 is 1010001.Binary equivalent of 9 is 1001.

Table 1 Binary division of Nikhilam method [3]

Denominator1001	Numerator 1010001						
Deficit =10000-1001=0111	Bits allotted for quotient			Bits allotted for remainder			
	1	0	1	0	0	0	1
		0	1	1	1		
			0	0	0	0	
				0	10	10	10
	1	0	10	11	0	1	1

From the above Table 1, quotient is 110; the remainder is 11011.Remainder is greater than the denominator, then a division operation to be performed again for the remainder.

TABLE 2 Continuation of binary division by Nikhilam method

Denominator 1001	Numerator = previous reminder=11011				
Deficit 0111	Bits allotted for quotient	Bits allotted for remainder			
	1	1	0	1	1
		0	1	1	1
	1	10	0	1	0

From the above Table 2, the quotient is 1, the remainder is 10010.This quotient is added to previous quotient which becomes 110+1=111, is the quotient and remainder is 10010. Again remainder obtained is greater than the denominator, hence again division should be performed on the remainder.

TABLE 3 Continuation of binary division by Nikhilam method

Denominator 1001	Numerator = previous reminder=10010				
Deficit 0111	Bits allotted for quotient	Bits allotted for remainder			
	1	0	0	1	0
		0	1	1	1
	1	1	0	0	1

From the above Table 3,the quotient obtained is 1and remainder is 1001.Remainder equals to denominator. Hence 1 is added to the quotient and remainder is '0'.Total quotient arrived is 111+1+1=1001 and the remainder is '0'.

This sutra can also be applicable for binary division, but it will take more number of steps than conventional methods in some cases, so it is not preferred for implementation [4].But according to the implementation of Nikhilam method of shifting operations will give better performance compared to conventional single precision floating point divider [3].

Process of Division by Dwajanka

In the process of division by Dwajanka, denominator is placed on the left most to numerator as shown in below Fig. 3. The numerator is separated into two parts. Right side part of numerator the number of digits equals to number of digits of the denominator. The denominator is represented by'd' numerator is represented by 'N' and the quotient is represented by Q. Remainder bits must be equals to flag bits.

	dn-1	dn-2	....	d1	d0	Nk	Nk-1	Nk-2	...	Nn	Nn-1	..	N1
dn													
						Qk-n	Qk-n-1	Q1	Q0				
			-										

Fig. 3 The process of division by Dwajanka [4]

For the first iteration, the leftmost (first) digit of a numerator (N_k) is considered and is divided with the leftmost digit of the denominator (d_n).The quotient is noted down as Q_{k-n}. And the remainder is to be considered for next iteration along with the next bit of the numerator. d_{n-1}, d_{n-2}..... d_0 bits are called as flag bits. Vertically and crosswise multiplication like Urdhvatiryakbhyam operation is done between flag and quotient bits. For the operation of Dwajaka, it needs series operations of multiplication subtraction and division. Vertical and Cross multiplication is done between flag bits and resultant quotient bits. And this product subtracted from the numerator prefixed with the previous remainder. The result of Subtraction is divided by the

denominator (d_n),subsequently quotient (Q_{k-n-1}) and the remainder for the next iteration are obtained. For calculation of the remainder of the series of operations no need to consider the division operation, but only multiplication and subtraction operations are needed [4].

Binary division example of Dwajanka

Dwajanka can also be applied for binary division.

TABLE 4 Binary Division by Dwajanka Method [4]

Denominator 1001	Numerator = previous reminder=10010						
	Bits allotted for quotient				Bits allotted for remainder		
	1	0	1	0	0	0	1
		0	0	0			
				1			
	1	0	1	-1			
			0	0			
	1	0	0	0			

To perform 81÷9 in binary, binary equivalent of 81 is 1010001.Binary equivalent of 9 is 1001.By performing the Dwajanka division operation,the quotient obtained is1000 and the remainder is 1001.Remainder obtained is equal to the denominator. To get the correct remainder,again division operation is performed on the remainder.Finally the quotient obtained is 1000+1=1001 and the remainder is 0.

Paravartya

Process of Paravartya division

The process of Paravartya division is similar to the Nikhilam method ,but here the Deficit bits are replaced with transposed bits of denominator bits.Paravartya means transpose and apply hence this division is named as Paravartya [1].

Paravartya division applied to binary numbers

Illustration: To perform 81÷9 in binary form. The binary equivalent of 81 is 1010001.And the binary equivalent of 5 is 101

Table 5 Binary number division by Paravartya method [1]

Denominator1001	Numerator 1010001						
Transposed Bits 0 0 -1	Bits allotted for quotient			Bits allotted for remainder			
	1	0	1	0	0	0	1
		0	0	-1			
			0	0	0		
				0	0	-1	
					0	0	1
Result	1	0	1	-1	0	-1	10

Paravartya sutra can be easily applied to binary numbers. Form the above Table 5, the quotient is 101-1 and the remainder is 0 -1 10. To obtain the correct quotient, consider quotient part 11-1-1 is equal to (101-1=1001) and the remainder is equal to (010-010=000. Total quotient is 1001 and the remainder is 0.

V. LITERATURE REVIEW

The main aim of this paper is to study the binary divisions using Vedic mathematics and introduce the Vedic algorithms for computing the division operations. Here the DE convolution algorithm is implemented. Vedic division algorithms are easy to learn and calculate the division operations. In this paper, implementation of non restoring division algorithm, Nikhilam and Paravartya binary dividers are done along with comparisons among non restoring division algorithm, Nikhilam and Paravartya binary dividers. From the simulated results obtained the Delay of the divider using Non Restoring Algorithm is 17.911 ns. Delay of the divider By Nikhilam Algorithm is 35.787 ns. Delay of the divider by using Paravartya Algorithm is 14.452 ns. From the simulated results, time delay of Vedic divider architecture is 19% less than the conventional method [1].

In the paper of high performance VLSI Vedic divider implementation done for high performance divider using a Parvartya Yojayet Vedic algorithm. Using 32nm standard cell libraries, static timing analysis is done on Vedic and conventional method of dividers of 32-bit dividend and 16-bit divisor. Vedic divider and conventional divider were synthesized by 32nm standard cell libraries. In this paper, it is proved that ~ 109 mW power is saved compared to conventional divider and Vedic divider is~7 times faster than conventional divider. The area occupied in Vedic divider is 13 times less than conventional divider [2].

In the paper Vedic implementation of single precision floating point divider, the division is performed by a Nikhilam method of Vedic division. Division algorithm consumes no of slice LUTs in the design are 159 whereas other algorithm uses minimum 176no of slice LUTs. No of slices consumed by this algorithm is 54 and other algorithms utilizes minimum of 103 to 447 slices.

Utilization of resources, improved to 12% when compared to conventional divider. The package density is also more when compared to the other division algorithm. This divider operated at operating frequency of 53.77MHZ. Power consumption is reduced to 2mw. Maximum delay occurred is 5.405ns. Conventional ALUs are consuming more power. Since, proposed design consumes less power, so bulky heat-sinks overhead can be eliminated. This makes the installation of processor simple and can be equipped with portable or mobile devices [3].

In this thesis binary division using Dwajanka is implemented using Verilog code and is compared with design ware block and shows advantages in power area and cycle time [4].

VI. CONCLUSION

In this paper three Vedic techniques applied for binary division are discussed.From existing papers, the Vedic binary divisions provides better performance in terms of area, power and delay when compared to the conventional binary divider structure. For implementation of ieee754 floating point divider, it is necessary to use a binary divider. By using Nikhilam ,Dwajanka and Paravartya Vedic techniques, IEEE 754 floating point divider structures yields better performance when compared to conventional methods.

REFERENCES

[1] Surabhi Jain, Mukul Pancholi, Harsh Garg, Sandeep Saini "Binary division algorithm and high speed deconvolution algorithm (Based on Ancient Indian Vedic Mathematics)"*international conference*, 14-17 May 2014,pp.1-5.

[2] bhanu tej "Vedic divider - A high performance computing algorithm for VLSI applications" *Controls and Communications (CCUBE),2013 international conference*, 27-28 Dec. 2013 pp.1 - 5 ISBN:978-1-4799-1599-6.

[3] Najib Ghatte, Shilpa Patil, Deepak Bhoir "Single Precision Floating Point Division" *proceedings of Fifth IRF International Conference*, 10th, 2014, Goa, India, ISBN: 978-93-84209-45-2.

[4] Bengali, Saurabh Sunil "Vedic Mathematics and Its Applications in Computer Arithmetic". (Under the direction of Dr. Paul Franzo) Raleigh, North Carolina

[5] Manisha Sangwan "Design and Implementation of Single Precision Pipelined Floating Point Co-Processor" *2013 International Conference on Advanced Electronic Systems (ICAES).*

[6] http://mathlearners.com/vedic-mathematics/division-in-vedic-mathematics