Implementation and Comparative Analysis between Different Precision Interval Arithmetic based Multiplication using Modified Array Method

Krutika Ranjankumar Bhagwat[#1] , Dr. Tejas V. Shah[*2] , Prof. Deepali H. Shah[#3]
[#]*Instrumentation & Control Engineering Department*
L. D. College of Engineering
Ahmedabad-380015, Gujarat, India
[*]*S.S College of Engineering*
Bhavnagar - 364060, Gujarat, India

***Abstract*— This paper presents the design of different precision modified array multipliers, which performs interval multiplication. Modified array multiplier requires carry save adders instead of full adders that reduces the delay in respect of conventional array multiplier. The double precision multiplication , single precision multiplication, and half precision multiplication are require 53 x 53, 24 x 24 , 11 x 11 multiplication respectively, which are done by array multiplier. Multipliers are based on interval arithmetic which provides the better accuracy, by avoiding rounding off error over conventional floating point multiplier. There is performance improvement as increasing precision, but it requires slightly more area and delay.**

***Keywords*— Double Precision, Single Precision, Half Precision , Interval Multiplication , Significand Multiplier , Array Multiplier, Modified Array Multiplier.**

I. INTRODUCTION

IEEE 754 standard defines half, single and double precision they have 1 sign bit and 5,8,11 bits for exponent respectively and 11,24, 53 bits for mantissa .The typical precision of the basic binary formats is one bit more than the width of its mantissa. The extra bit of precision comes from an implied (hidden) 1 bit . [4]

Total Precisions are therefore 53 bits (approximately 3 decimal digits, 11 log10 (2) ≈ 3.31) , 11 bits (approximately 7 decimal digits, 24 log10 (2) ≈ 7.22) 24 bits (approximately 16 decimal digits, 53 log10 (2) ≈ 15.955) respectively . [4] The Bias and other detail of different precision has been depicted in table 1.

Table I. Different Precision Formats

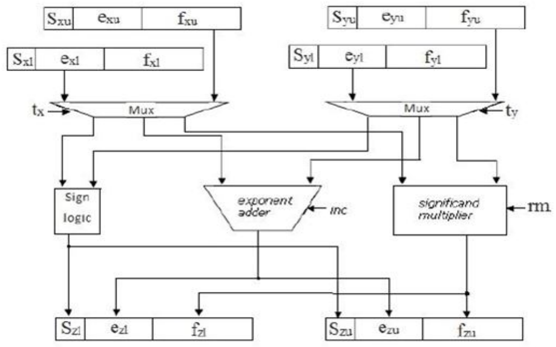
	Sign	Exponent	Mantissa	BIAS
Half Precision	1[15]	5[14-10]	10[9-00]	15
Single Precision	1 [31]	8 [30-23]	23 [22-00]	127
Double Precision	1 [63]	11 [62-52]	52 [51-00]	1023

II. INTERVAL MULTIPLICATION

Multiplication of the intervals $x = [x_l , x_u]$ and $y = [y_l ,y_u]$ is defined as:

$Z = x * y$
$= [\min(x_l y_l,\ x_l y_u ,x_u y_l ,\ x_u y_u),\max(x_l y_l,\ x_l y_u,\ x_u y_l,\ x_u y_u)]$

The interval multiplier shown in figure 1 has input and output registers, sign logic, an exponent adder and a significand multiplier with rounding and normalization logic. The input and output registers are each 16 bits,32 bits, 64 bits respectively and two multiplexer with control signal t_x ,t_y are used .[10]The sign logic computes the sign of the result by performing the exclusive-or of the sign bits of the input operands. For half , single and double precision the exponent adder performs an 5,8 and 11 bit addition of the two exponents and subtracts the exponent bias of 15, 127 and 1023 respectively. The significand multiplier performs a 11 bit by 11 bit , 24 bit by 24 bit and 53 bit by 53 bit array multiplication for half , single and double precision respectively. If the most signicant bit of the product is one, the normalization logic shifts the product right one bit and increments the exponent. The rounding logic rounds the product to 11,24 and 53 bits based on a rounding to nearest even mode for all the three precision formats.[10]



S_{xu} e_{xu} f_{xu}
S_{xl} e_{xl} f_{xl}
S_{yu} e_{yu} f_{yu}
S_{yl} e_{yl} f_{yl}
t_x
Mux
Mux
t_y
Sign logic
exponent adder
inc
significand multiplier
rm
S_{zl} e_{zl} f_{zl}
S_{zu} e_{zu} f_{zu}

Fig 1. Interval multiplier

A. *sign and toggle bits*

$$fp + l_e \cdot (\overline{S_{yl}} + \overline{S_{yu}} \cdot S_{xl}) + l_e \cdot (S_{yu} + S_{xl} \cdot S_{yl})$$

$$f_p + l_e \cdot (\overline{S_{xl}} + \overline{S_{xu}} \cdot S_{yl}) + \overline{l_e} \cdot (S_{xu} + S_{xl} \cdot S_{yl})$$

$$\overline{f_p} \cdot S_{xl} \cdot \overline{S_{xu}} \cdot S_{yl} \cdot \overline{S_{yu}}$$

Table 2 is given for Setting of sign and toggle bits for interval multiplication and Figure 2 is given for all 9 cases for interval multiplication for lower interval . In this figure 2 indicator indicates case 9 for that output z_c becomes 1. Where Z= {mn, mx} for condition $x_l < 0 < x_u$, $y_l < 0 < y_u$. That gives lower interval mx. Figure 3 is given for all 9 cases for interval multiplication for upper interval . In this figure 3 indicator indicates case 9 for that output z_c becomes 1. Where Z= {mn, mx}for condition $x_l < 0 < x_u$, $y_l < 0 < y_u$. That gives upper interval mn.[10]

B. *Rounding Mode*

$$r_m 0 = fp\_r_m 0 \cdot fp + le \cdot \overline{fp}$$
$$r_m 1 = fp\_r_m 1 + \overline{fp}$$

To specify the operation performed by the multiplier a control bit fp is used. This bit is set to one for floating multiplication and zero for interval multiplication. Figure 4 is given for Rounding Mode Waveforms for fp=0 and fp=1 . Table 3 gives value for Rounding mode bits.

TABLE II
Setting of *sign and toggle bits for interval multiplication*

							$l_e = 1$		$l_e = 0$		
Case	Condition	s_{xl}	s_{yl}	s_{xu}	s_{yu}	Z	t_x	t_y	t_x	t_y	zc
1	$x_l > 0, y_l > 0$	0	0	0	0	$\{x_l y_l , x_u y_u\}$	1	1	0	0	0
2	$x_l > 0, y_u < 0$	0	0	1	1	$\{x_u y_l , x_l y_u\}$	0	1	1	0	0
3	$x_u < 0, y_l > 0$	1	1	0	0	$\{x_l y_u , x_u y_l\}$	1	0	0	1	0
4	$x_u < 0, y_u < 0$	1	1	1	1	$\{x_u y_u , x_l y_l\}$	0	0	1	1	0
5	$x_l < 0 < x_u , y_l > 0$	1	0	0	0	$\{x_l y_u , x_u y_u\}$	1	0	0	0	0
6	$x_l < 0 < x_u , y_l < 0$	1	0	1	1	$\{x_u y_l , x_l y_l\}$	0	1	1	1	0
7	$x_l > 0, y_l < 0 < y_u$	0	0	1	0	$\{x_u y_l , x_u y_u\}$	0	1	0	0	0
8	$x_u < 0, y_l < 0 < y_u$	1	1	1	0	$\{x_l y_u , x_l y_l\}$	1	0	1	1	0
9	$x_l < 0 < x_u , y_l < 0 < y_u$	1	0	1	0	$\{x_u y_u , x_l y_u\}$	x	x	x	x	1

C. *l_e=1(lower interval)*

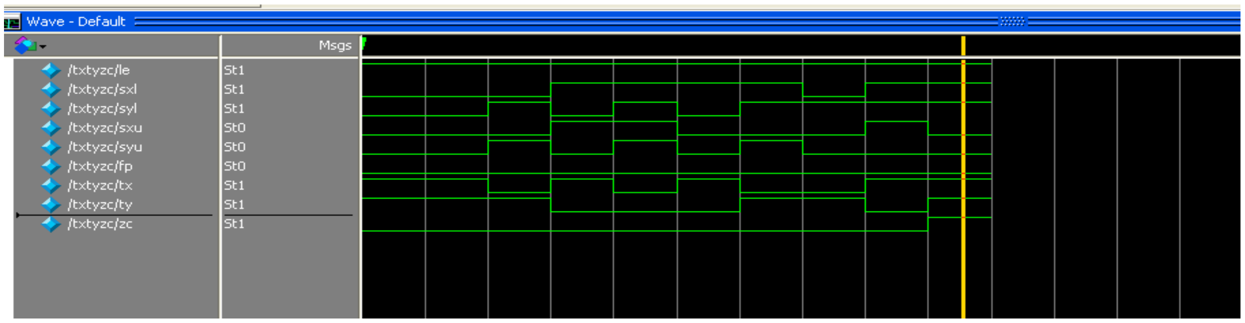


Fig 2 Sign and Toggle Bits Waveforms for le=1(lower interval)

D. *le=0(upper interval)*

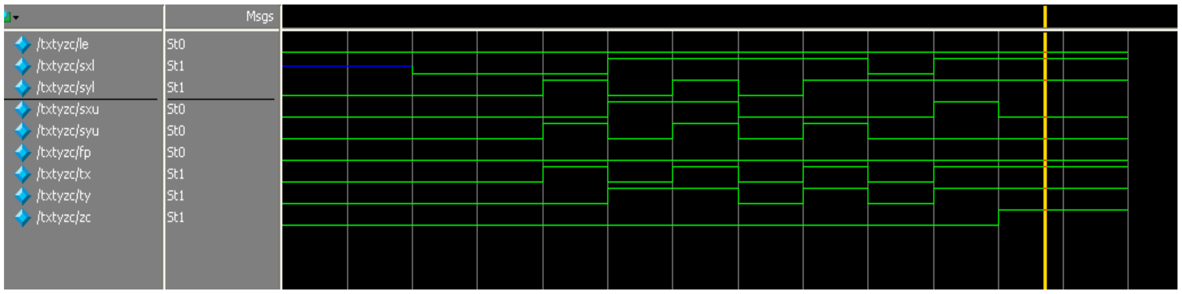


Fig 3:- Sign and Toggle Bits Waveforms for le=0(upper interval)

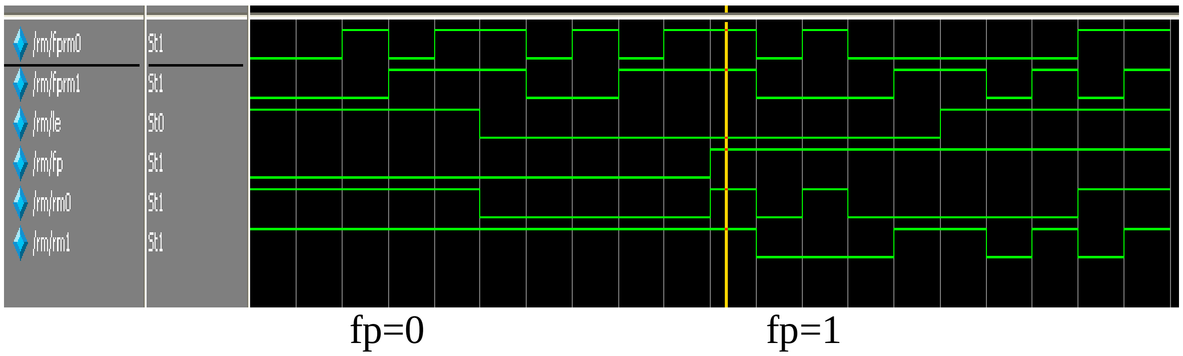


Fig4:- Rounding Mode Waveforms for fp=0 and fp=1

Table III. Rounding mode bits

Rounding mode	fp_rm1	fp_rm0
Round to nearest even	0	0
Round toward 0	0	1
Round toward + ∞	1	0
Round toward - ∞	1	1

III. SIGNIFICAND MULTIPLIER (ARRAY MULTIPLIER USING CSA)

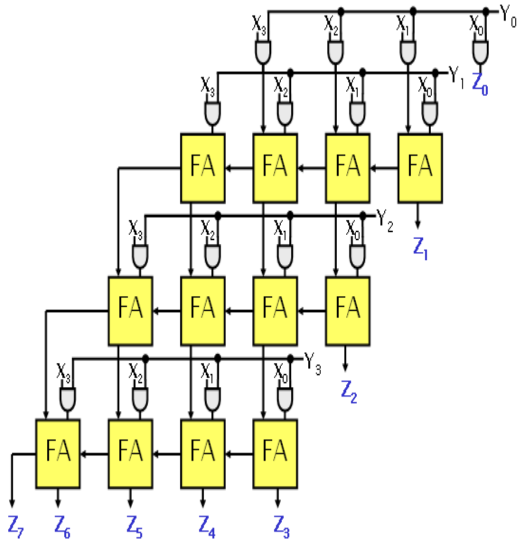


Fig 5. Array multiplier

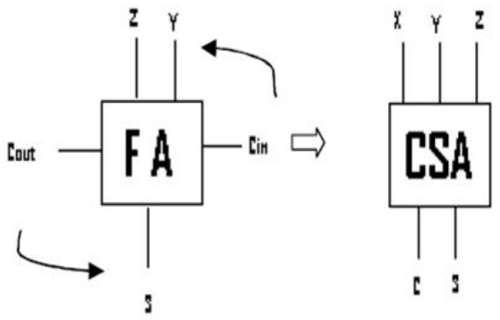


Fig 6. Modified FA to CSA

m x n bit multiplication can be viewed as forming n partial products of m bits each and then summing the appropriately shifted partial products to produce an m+n bit result p. Therefore, generating partial products consist of the logical ANDing of the appropriate bits of the multiplier and multiplicand. Then, each column of partial products must be added and if necessary, any carry values passed to the Next column. Simple array multiplication using full adder is shown in figure 5.[4] Partial products are added using carry save adder instead of full adder which reduces delay. Full adder is replaced with CSA given in figure 6.[13]

Multiplication requires n*(n-1) csas, where n=53, 24,11 respectively. So , n*(n-1) = 53 *52= 2756, n*(n-1) = 24 *23= 552, n*(n-1) = 11 *10= 110 csas are used respectively . Arrangement of 2756 , 552and 110 csas are used to add partial products of multiplier respectively

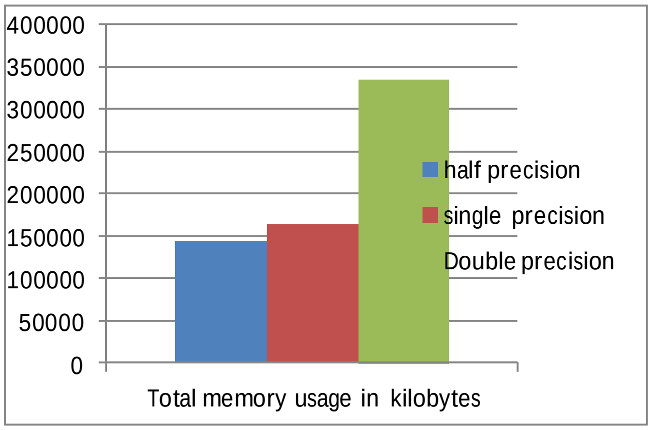


Fig 7 comparative area analysis

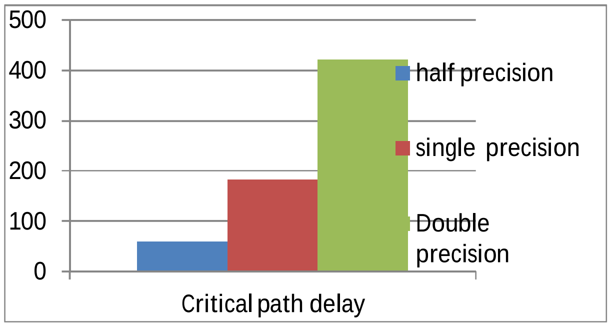


Fig 8. comparative critical path delay analysis

compared to half precision interval arithmetic array multiplier. So speed of interval arithmetic array multiplier decreases and area increases as precision increases.

TABLE IV. ANALYSIS REPORT

Device : XC351600E & Family: SPARTAN 3E			
Device utilization summary for interval arithmetic based modified array multiplication			
Area analysis	Half precision	Single precision	Double precision
Number of Slices	359	837	8603
Number of 4 input LUTs	637	1455	14967
Number of Ios	127	97	499
Number of bonded IOBs	127	97	499
Real time delay	18.00 ns	62.00 ns	358 ns
Number of MULT18X18SIOs	4	16	32
Critical path delay	58.57 ns	182.122 ns	421.563 ns
Total memory usage	145056 KB	164640 KB	334140 KB

IV COMPARATIVE ANALYSIS

From the analysis report , number of MULT 18X18S IOs are 4,16 and 32 for half , single and double precision respectively. The memory required is 145056, 164640 ,334140 kilobytes for half , single and double precision respectively . The critical path delay is 58.57,182.122, 421.563 ns for half , single and double precision interval arithmetic array multiplier respectively. the detail comparison has been provided in table 4.

Figure 7 gives comparative area analysis of half , single and double precision modified array multiplication using interval arithmetic. Figure 8 gives comparative critical path delay analysis of half , single and double precision modified array multiplication using interval arithmetic.

V. CONCLUSION

Interval arithmetic provides reliability and accuracy by computing a lower and upper bound in which result is guaranteed to reside. Concept of carry look ahead for exponent adder is used which reduces the delay.

Concept of carry save adder in array multiplication is used instead of half adders and full adders which reduces the number of gates and delay.

For interval arithmetic array multiplier double precision requires more real time delay compared to single precision. And single precision requires more real time delay

REFERENCES

[1] Josh Milthorpe and Alistair Rendell “Learning to live with errors: A fresh look at floating-point computation”, Australian National University, Computing Conference 2005

[2] Gupte, ruchir “Interval arithmetic logic unit for dsp and control applications”, Electrical and Computer Engineering, Raleigh 2006

[3] Samir Palniker, “ Verilog HDL: A Guide to Digital Design and Synthesis”, ISBN 81-297-0092-1, @2003 SUN MICROSYSTEMS

[4] “ IEEE Standard 754 for Binary Floating Point Arithmetic ” , ANSI/IEEE Standard No. 754, American National Standards Institute, Washington DC , 1985.

[5] Behrooz Parhami , “Computer Arithmetic, Algorithms and Hardware Designs” , 2nd Edn, OXFORD, Mar. 2011

[6] Alexandru Amaricai Mircea Vladuaiu Lucian Prodan Mihai Udrescu Oana Boncalo “Design of Addition and Multiplication Units for High Performance Interval Arithmetic Processor”, Computer Science and Engineering Department, ©2007 IEEE

[7] Michael J. Schulte and Earl E. Swartzlander Jr., “A Performance Comparison Study on Multiplier Designs” ,IEEE Transaction On Computers, May 2000

[8] Yong Dou S. Vassiliadis G. K. Kuzmanov G. N. Gaydadjiev , “64-bit Floating-Point FPGA Matrix Multiplication” , National Laboratory for Computer Engineering, FPGA’05, Monterey, California, USA, February , 2005

[9] Anane Nadjia, Anane Mohamed, Bessalah Hamid, Issad Mohamed & Messaoudi khadidja, “Hardware Algorithm for Variable Precision Multiplication on FPGA” © 2009 IEEE

[10] James E. Stine and Michael J. Schulte “A Combined Interval and Floating Point Multiplier”, Computer Architecture and Arithmetic Laboratory ,Electrical Engineering and Computer Science Department, Lehigh University, Bethlehem, PA 18015

[11] Sparc Architecture Manual

[12] Prof. LohCS3220- Processor Design “Carry-Save Addition” - Spring 2005, February , 2005

[13] “Carry Save Adder Trees in Multipliers” ecen 6 2 6 3 advanced vl sI design november 3, 1999

[14] .N. Marimuthu1, P. Thangaraj “Low Power High Performance Multiplier”, Anna University, Tamil nadu , India

[15] Steve Kilts, “Advanced FPGA Design Architecture, Implementation, and Optimization”, Wiley – Interscience, A John Wiley & Sons, ISBN 978-0-470-05437-6, @ 2007 IEEE.