DESIGN AND SIMULATION OF A HIGH SPEED DOUBLE PRECISION FLOATING POINT UNIT USING VERILOG

B.ANIL KUMAR1, K.SREENIVASA RAO2,
1Dept. of VLSI System design,2Associate professor,Dept of E,C,E,Annamacharya Institute of Techonology& Sciences, Rajampet, Andhra Pradesh, India

Abstract: Floating point formatrepresentsvery large or small values, large range is required as the integer representation is no longer appropriate. These values can be represented using the IEEE-754 standard based floating point representation.

In existing system floating point ALU with universal logic gate we can perform addition,subtraction,multiplication and logical operation with less delay and less area using single precision. Single precision floating point format is a computer number format that occupies 32-bits in a computer memory and represents a wide dynamic range of values by using a floating point.

The proposed system presents high speed ASIC implementation of a floating point arithmetic unit which can perform addition, subtraction, multiplication, division functions on 64-bit operands that use the IEEE 754-2008 standard. Pre-normalization unit and post normalization units are also discussed along with exceptional handling. All the functions are built by feasible efficient algorithms with several changes incorporated that can improve overall latency, and if pipelined then higher throughput. The algorithms are modeled in Verilog HDL and the RTL code for adder, subtractor, multiplier, divider, square root are synthesized using Xilinx ISE tool.

***Index Terms*—floating point number, normalization, exceptions, latency, overflow, underflow, etc.**

I. INTRODUCTION

An arithmetic circuit which performs digital arithmeticoperations has many applications in digital coprocessors,application specific circuits, etc. Because of the advancementsin the VLSI technology, many complex algorithms thatappeared impractical to put into practice, have become easilyrealizable today with desired performance parameters so thatnew designs can be incorporated [2]. The standardized methodsto represent floating point numbers have been instituted by theIEEE 754 standard through which the floating point operationscan be carried out efficiently with modest storage requirements. The three basic components in IEEE 754 standard floatingpoint numbers are the sign, the exponent, and the mantissa [3].The sign bit is of 1 bit where 0 refers to positive number and 1refers to negative number [3]. The mantissa, also calledsignificandwhich is of 23bits composes of the fraction and aleading digit which represents the precision bits of the number[3] [2]. The exponent with 8 bits represents both positive andnegative exponents. A bias of 127 is added to the exponent toget the stored exponent [2]. Table 1 show the bit ranges forsingle (32-bit) and double (64-bit) precision floating-pointvalues [2]. A floating point number representation is shown intable 2 The value of binary floating point representation is asfollows where S is sign bit, F is fraction bit and E is exponentfield.

Value of a floating point number= (-1)S x val (F) x 2val(E)

Table 1: Bit Range For Single (32-Bit) And Double (64-Bit) PrecisionFloating-Point Values

	Sign	Exponent	Fraction	Bias
Single precision	1[31]	8[30-23]	23[22-00]	127
Double precision	1[63]	11[62-52]	52[51-00]	1023

Table 2: Floating Point Number Representation

64 bits		
Sign	Exponent	mantissa
1 bit	11 bits	52 bits

There are four types of exceptions that arise during floatingpoint operations. The Overflow exception is raised whenever theresult cannot be represented as a finite value in the precisionformat of the destination [13].

The Underflow exception occurswhen an intermediate result is too small to be calculated accurately, or if the operation's result rounded to the destinationprecision is too small to be normalized [13] The Division byzero exception arises when a finite nonzero number is dividedby zero [13]. The Invalid operation exception is raised if thegiven operands are invalid for the operation to be performed[13].

In this paper, ASIC implementation of a high speed FPU hasbeen carried out using efficient addition, subtraction,multiplication, division algorithms. Section II depicts thearchitecture of the floating point unit and methodology, to carryout the arithmetic operations. Section III presents the arithmeticoperations that use efficient algorithms with some modificationsto improve latency. Section IV presents the simulation resultsthat have been simulated in Cadence RTL compiler using 180nm process. Section V presents the conclusion.

II. ARCHITECTURE AND METHODOLOGY

The FPU of a double precision floating point unit that performsadd, subtract, multiply, divide functions is shown in figure 1 [1].Two pre-normalization units for addition/subtraction andmultiplication/division operations has been given [1].

Post normalization unit also has been given that normalizes the mantissa part [2]. The final result can be obtained after postnormalization.To carry out the arithmetic operations, two IEEE-754 format single precision operands are considered. Prenormalizationof the operands is done. Then the selectedoperation is performed followed by post-normalizing the output obtained.Finally the exceptions occurred are detected and handled using exceptional handling. The executed operationdepends on a three bit control signal (z) which will determinethe arithmetic operation is shown in table 3.

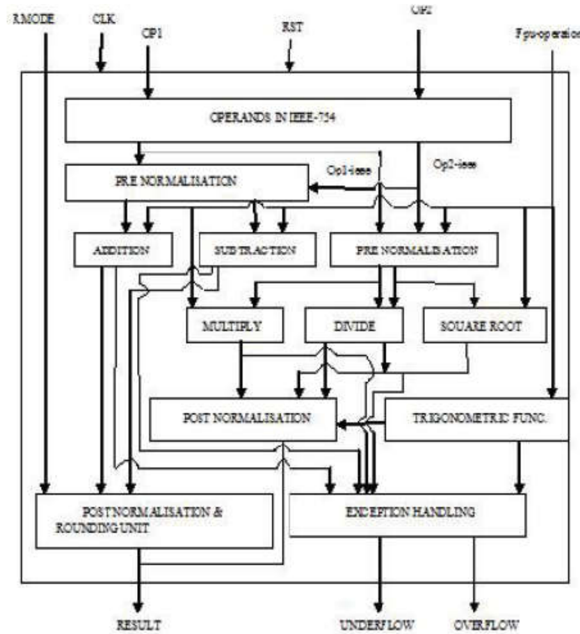

Fig.1: Block Diagram of floating point arithmetic unit [1]

Table 3: Floating Point Unit Operations

z(control signal)	Operation
2’b000	Addition
2’b001	Subtraction
2’b010	Multiplication
2’b011	Division
2’b100	Square root

III 64 BIT FLOATING POINT ARITHMETIC UNIT

A. Addition Unit:

One of the most complex operations in a floating point unitcomparing to other functions which provides major delay andalso considerable area. Many algorithms has been developedwhich focused to reduce the overall latency in order to improveperformance. The floating point addition operation is carried outby first checking the zeros, then aligning the significand,followed by adding the two significands using an efficientarchitecture.

The obtained result is normalized and is checkedfor exceptions. To add the mantissas, a high speed carry lookahead has been used to obtain high speed. Traditional carry look ahead adder is constructed using AND, XOR and NOT gates.

The implemented modified carry look ahead adder uses onlyNAND and NOT gates which decreases the cost of carry lookahead adder and also enhances its speed also [4].The 16 bit modified carry look ahead adder is shown in figure 2and the metamorphosis of partial full adder is shown in figure 3using which, a 24 bit carry look ahead adder has beenconstructed and performed the addition operation.

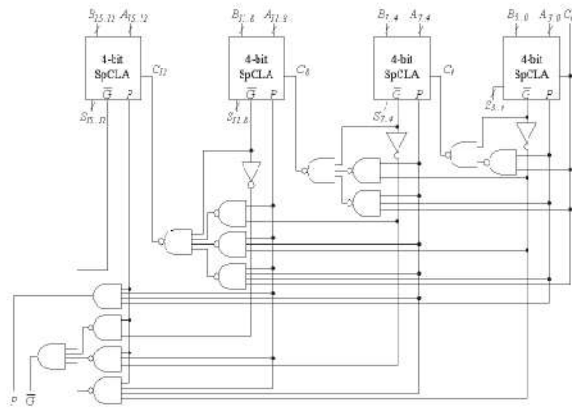

Fig.2: 16 bit modified carry look ahead adder [4]

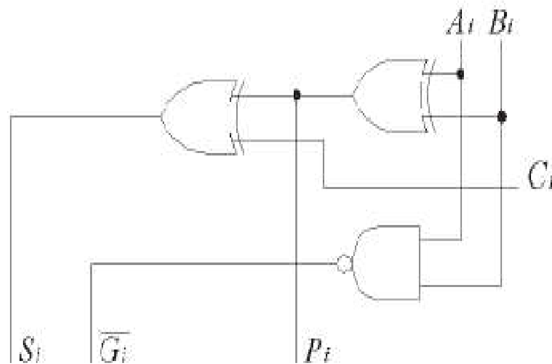

Fig.3: Metamorphosis of partial full adder [4]

B. Subtraction Unit:

Subtraction operation is implemented by taking 2'scomplement of second operand. Similar to addition operation,subtraction consists of three major tasks pre normalization,addition of mantissas, post normalization and exceptionalhandling. Addition of mantissas is carried out using the 24 bitMCLA shown in figure 2 and figure 3.

C. Multiplication Algorithm

Constructing an efficient multiplication module is a iterativeprocess and 2n-digit product is obtained from the product of twon-digit operands. In IEEE 754 floating-point multiplication, thetwo mantissas are multiplied, and the two exponents are added.Here first the exponents are added from which the exponent bias(1023) is removed. Then mantissas have been multiplied usingfeasible algorithm and the output sign bit is determined by exoring the two input sign bits. The obtained result has beennormalized and checked for exceptions.To multiply the mantissas Bit Pair Recoding (or ModifiedBooth Encoding) algorithmhas been used, because of which thenumber of partial products gets reduces by about a factor of two,with no requirement of pre-addition to produce the partialproducts. It recodes the bits by considering three bits at a time.

Bit Pair Recoding algorithm increases the efficiency ofmultiplication by pairing. To further increase the efficiency ofthe algorithm and decrease the time complexity, Karatsubaalgorithm can be paired with the bit pair recoding algorithm.One of the fastest multiplication algorithm is Karatsubaalgorithm which reduces the multiplication of two n-digitnumbers to 3nlog32 ~ 3n1.585 single-digit multiplications andtherefore faster than the classical algorithm, which requires n2single-digit products [11]. It allows to compute the product oftwo large numbers x and y using three multiplications of smallernumbers, each with about half as many digits as x or y, withsome additions and digit shifts instead of four multiplications[11]. The steps are carried out as follows

Let x and y be represented as n-digit numbers with base B and

$$m<n.$$

$$x = x1Bm + x0$$

$$y = y1Bm + y0$$

Where x0 and y0 are less than Bm [11]. The product is then

$$xy= (x1Bm + x0)(y1Bm + y0)= c1B2m + b1Bm + a1$$

$$\text{Where } c1 = x1y1$$

$$b1 = x1y0+ x0y1$$

$$a1 = x0y0.$$

$$b1 = p1- z2 - z0$$

$$p1 = (x1 + x0)(y1 + y0)$$

Here c1, a1, p1 has been calculated using bit pair recodingalgorithm. Radix-4 modified booth encoding has been usedwhich allows for the reduction of partial product array by half[n/2]. The bit pair recoding table is shown in table 3. In theimplemented algorithm for each group of three bits (y2iþ1, y2i,y2i_1) of multiplier, one partial product row is generatedaccording to the encoding in table 3.

Radix-4 modified booth encoding (MBE) signals and theirrespective partial products has been generated using the figures4 and 5. For each partial product row, figure 4 generates the one,two, and neg signals. These values are then given to the logic infigure 5 with the bits of the multiplicand, to produce the wholepartial product array. To prevent the sign extension the obtainedpartial products are extended as shown in figure 6 and the product has been calculated using carry save select adder.

Table 3: Bit-Pair Recoding [11]

BIT PATTERN		**OPERATION**
0 0 0	NO OPERATION	
0 0 1	1xa	prod=prod+a;
0 1 0	2xa-a	prod=prod+a;
0 1 1	2xa	prod=prod+2a;
1 0 0	-2xa	prod=prod-2a;
1 0 1	-2xa+a	prod=prod-a;
1 1 0	-1xa	prod=prod-a;
1 1 1	NO OPERATION	

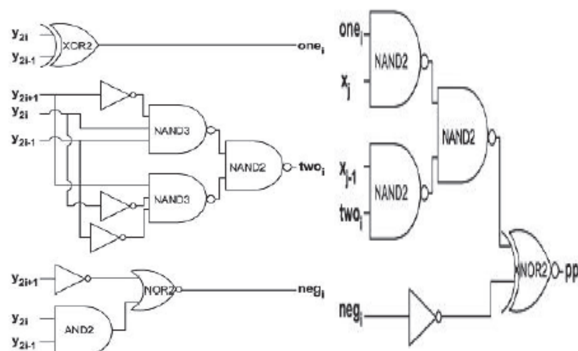

Fig.4: MBE signal generation [10] Fig.5: Partial Product Generation [10]

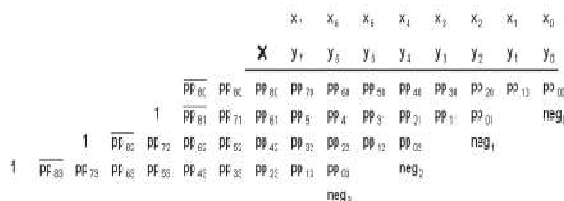
Fig.6: Sign prevention extension of partial products [10]

D. Division Algorithm

Division is the one of the complex and time-consumingoperation of the four basic arithmetic operations. Divisionoperation has two components as its result i.e. quotient and aremainder when two inputs, a dividend and a divisor are given.Here the exponent of result has been calculated by using theequation,

$e0 = eA – eB + bias (127) -zA + zB$

followed bydivision of fractional bits [5] [6]. Sign of result has beencalculated from exoring sign of two operands. Then the obtainedquotient has been normalized [5] [6].

Division of the fractional bits has been performed by usingnon restoring division algorithm which is modified to improvethe delay. The non-restoring division algorithm is the fastest among the digit recurrence division methods [5] [6]. Generallyrestoring division require two additions for each iteration if thetemporary partial remainder is less than zero and this results inmaking the worst case delay longer[5] [6]. To decrease the delayduring division, the non-restoring division algorithm wasintroduced which is shown in figure 7. Non-restoring divisionhas a different quotient set i.e it has one and negative one, whilerestoring division has zero and one as the quotient set[5] [6].

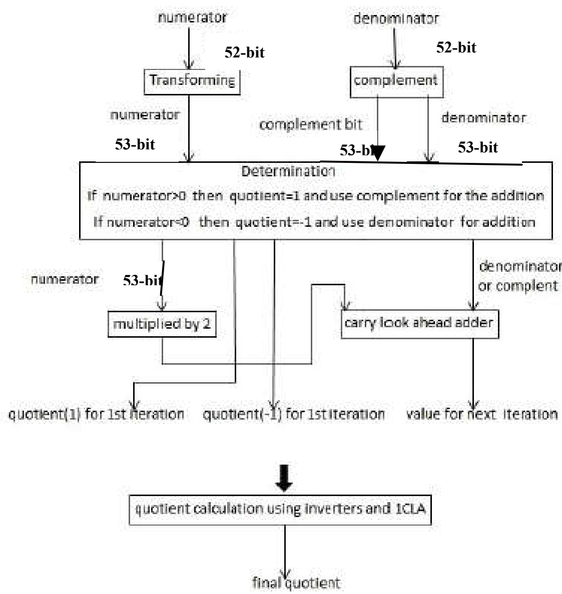

Fig.7: Non Restoring Division algorithm

Using the different quotient set, reduces the delay of non-restoringdivision compared to restoring division. It means,itonly performs one addition per iteration which improves itsarithmetic performance [6].

The delay of the multiplexer for selecting the quotient digitand determining the way to calculate the partial remainder canbe reduced through rearranging the order of the computations. Inthe implemented design the adder for calculating the partialremainder and the multiplexer has been performed at the sametime, so that the multiplexer delay can be ignored since theadder delay is generally longer than the multiplexer delay.

Second, one adder and one inverter are removed by using a newquotient digit converter. So, the delay from one adder and oneinverter connected in series will be eliminated.

E. Square Root Unit

Square root operation is difficult to implement because of thecomplexity of the algorithms. Here a low cost iterative singleprecision non-restoring square root algorithm has beenpresented that uses a traditional adder/subtractor whoseoperation latency is 25 clock cycles and the issue rate is 24clock cycles. If the biased exponent is even, the biased exponentis added to 126 and divided by two and mantissa is shifted to its left by 1 bit before computing its square root. Here beforeshifting the mantissa bits are stored in 52 bit register as 1.xx..xx.

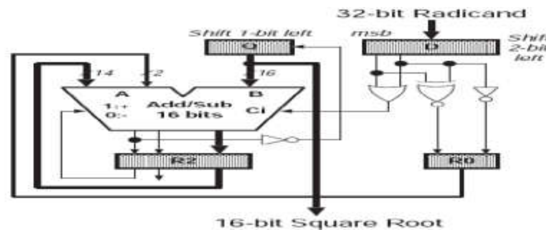

Fig.7: Non Restoring square root circuitry [15] [16]

After shifting it becomes 1x.xx…If the biased exponent is oddthe odd exponent is added to 127 and divided by two. Themantissa. The square root of floating point number has beencalculated by using non restoring square root circuitry which isshown in figure 8 [15] [16].

IV SIMULATION RESULTS

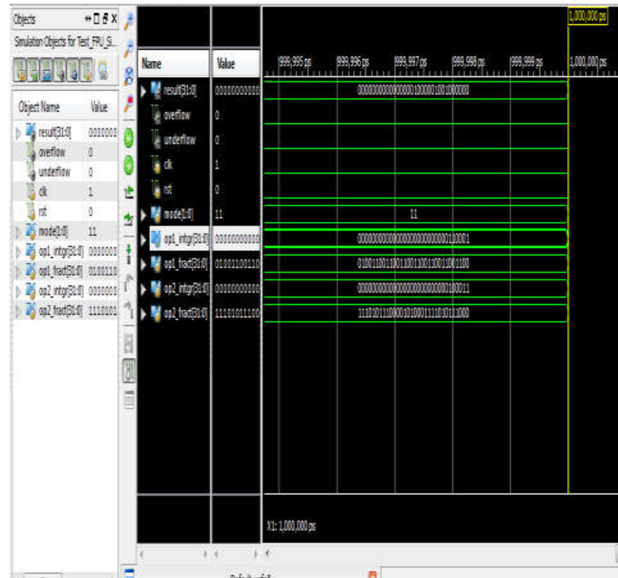

Figure : Implementation of 32 bit single precision operation

- In single precision floating point unit it executes only 32-bits.

Sotime consumption is more and speed alsoreduced. So for avoiding that problemwe propose a double precision floating point unit.

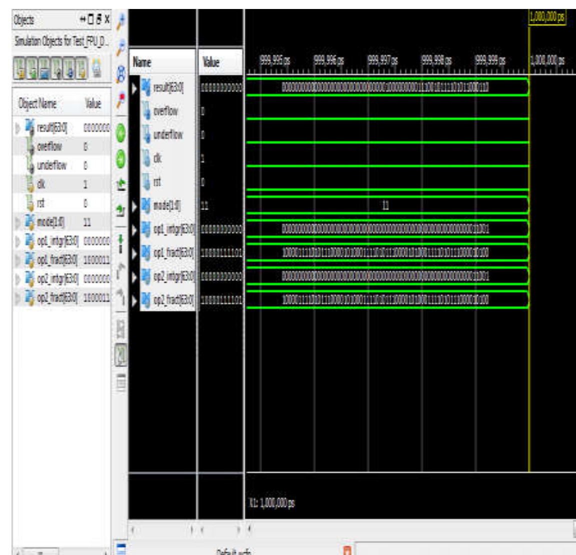

Figure : Implementation of 64 bit double precision

V CONCLUSION

The implementation of a high speed double precision FPU has beenpresented. . The design has been synthesized with Xilinx tool. Strategies havebeen employed to realize optimal hardware and power efficientarchitecture. The layout generation of the presented architecture usingthe backend flow is an ongoing process and is being done usingCadence RTL compiler with 180nM process technology. Hence it can be concluded that this FPU can be effectively used for ASICimplementations which can show comparable efficiency and speed andif pipelined then higher throughput may be obtained.

REFERENCES

[1] Rudolf Usselmann, "Open Floating Point Unit, The Free IP Cores Projects".
[2] EdvinCatovic, Revised by: Jan Andersson, "GRFPU – High PerformanceIEEE754 Floating Point Unit", Gaisler Research, FörstaLångatan 19, SE413 27 Göteborg, and Sweden.
[3] David Goldberg, "What Every Computer Scientist Should Know AboutFloating-Point Arithmetic", ACM Computing Surveys, Vol 23, No 1, March1991, Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto,California 94304.
[4] Yu-Ting Pai and Yu-Kumg Chen, "The Fastest Carry LookaheadAdder",Department of Electronic Engineering, Huafan University.
[5] Prof. Kris Gaj, Gaurav, Doshi, Hiren Shah, "Sine/Cosine using CORDICAlgorithm".
[6] S. F. Oberman and M. J. Flynn, "Division algorithms and implementations,"IEEE Transactions on Computers, vol. 46, pp. 833–854, 1997.
[7] Milos D. Ercegovac and Tomas Lang, Division and Square Root: Digit-Recurrence Algorithms and Implementations, Boston: Kluwer AcademicPublishers, 1994.
[8] ANSI/IEEE Standard 754-1985, IEEE Standard for Binary Floating-Point Arithmetic, 1985.
[9] BehroozParhami, Computer Arithmetic - Algorithms and Hardware Designs, Oxford: Oxford University Press, 2000.
[10] Steven Smith, (2003), Digital Signal Processing-A Practical guide forEngineers and Scientists, 3rd Edition, Elsevier Science, USA.