Performance Analysis of Floating Point MAC Unit

Sonali Mehta
Centre for Development of Advanced Computing,Mohali

Balwinder singh
Centre for Development of Advanced Computing, Mohali

Dilip Kumar, Ph.D
Centre for Development of Advanced Computing, Mohali

ABSTRACT
In order to meet the requirements in real time DSP applications MAC unit is required. The speed of the MAC unit determines the overall performance of the system. MAC unit basically consists of Multiplier, adder and an accumulator unit. In most of the cases floating point adder/subtractor and a multiplier are presented in IEEE-754 format for single precision format. In this research work MAC unit is proposed. Various floating point multipliers are designed with the help of adders such as Carry look ahead, Carry save, Carry skip and then their individual performance analysis is done on the basis of power, speed, and area. Finally MAC unit is made with the help of different floating point multiplier architectures to obtain the best results in terms of area, speed and power. Various floating point multiplier architectures used in MAC unit are pipelined floating point multiplier; carry save, carry look ahead and ripple carry multipliers. All the results are calculated in Xilinx ISE 12.4 design suite.

KEYWORDS
Area, Speed, Power, MAC, Floating point, Delay

1. INTRODUCTION
DSP applications involve many critical operations usually multiplications and accumulations. In real time signal processing applications high throughput MAC unit is always required to achieve a high performance in DSP applications .It is because in real time DSP applications speed and throughput are the major concerns. Floating point numbers represent an approximation to real numbers in a way that can support a wide range of values. The main goal of this paper is designing of floating point MAC unit. Representing real numbers in binary format requires floating point numbers .In this paper floating point numbers are represented according to IEEE 754 standard format. In single precision, a Floating Point number consists of a 32-bit word divided in 1 bit for sign, 8 bits for exponent that constitutes a 127 bias, and 23 bits for the significand. This standard supports two types of formats binary interchanges format and decimal interchange format. In many applications computation is done using floating point arithmetic. Earlier floating point operations were mainly implemented as software while for main stream general purpose processor hardware implementation was an option because cost of the hardware was not reasonable. Today every microprocessor is hardware specific for handling various floating point operations. In real time signal processing applications floating point MAC unit is required in order to achieve desired performance. But because of the advancements in reconfigurable logic now these Mac units can be implemented on FPGA. The goal of this project is FPGA implementation of floating point MAC unit for DSP applications. Floating point number can be given by equation (1):

$$z = (-1^{s}) * 2^{(exp-bias)} * (1*M) \quad (1)$$

Equation 1 represents IEEE 32 bit single precision floating point format. One very important requirement of the IEEE-754 representation is that the number should be represented with it closest equivalent for the precision chosen, which means that it is assumed that any operation is performed with infinite precision Any floating point number is first of all converted into this format (1) and further operations are performed. Floating-point (FP) addition is based on sequence of mantissa operations swap, shift, add, normalize, and round. A floating point adder first compares the exponents of the two input operands, swaps and shifts the mantissa of the smaller number to get them aligned. The number has to be adjusted if the incoming number is negative. Finally, the sum is renormalized, Exponents are adjusted accordingly, and resulting mantissas are truncated by an appropriate rounding scheme. If extra speed is required then FP adders use leading-zero anticipatory (LZA) logic to carryout pre-decoding for normalization shifts in parallel with the mantissa addition. Floating point multiplication basically involves xoring of the signs, multiplication of significands and adding exponents of both the numbers. After addition the exponent is called tentative result exponent. Then we have to subtract bias from added exponents. Result can be a normalised number if the MSB is 1.In this paper various multipliers such as carry save, carry look ahead, ripple carry are designed Then their performance analysis is done on various parameters such as area, speed and power. Then floating point MAC unit is designed.MAC basically consists of adder, multiplier and an accumulator certain values are used for special number representation, as follows. If the exponent is 0 and mantissa is 0 then the number is a zero number. If the exponent is 0 and mantissa is greater than 0 then the number is a subnormal number. If the exponent lies in between 0 and 255 and mantissa is greater than 0 then the number is a normal number. If the exponent is 255 and mantissa is 0 then the number is an infinite number. If the exponent is 255 and the mantissa is greater than 0 then the number is not a number.

Table 1. Special Numbers

S.No.	Exponent	Mantissa	output
1	=0	=0	Zero
2	=0	>0	subnormal
3	0<E<255	>0	normal
4	=255	=0	infinity
5	=255	=0	NAN

This paper has 4 sections: section II explains Related Work. Section III describes standard IEEE 754 format. Section IV explains organisation of work which has three subsections floating point adder, floating point multiplier, floating point MAC unit. Section V describes results.

2. RELATED WORK
Guillermo Marcus [2] presents a multiplier and an adder/subtractor for single precision floating point numbers in IEEE format. They have pipelined architecture which are implemented in VHDL. Mohamed Al-Ashrafy [1] presents a floating point multiplier In IEEE single precision floating

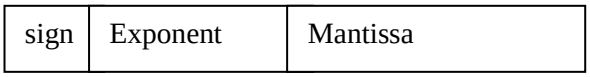
point format. The multiplier does not implement rounding and it just presents the significand multiplied result. Carlos Minchola [11] has presented FPGA implementation of a Decimal Floating Point (DFP) Adder/Subtractor. The design performs addition and subtraction on 64-bit operands that use the IEEE 754-2008 decimal encoding of DFP numbers and is based on a fully pipelined circuit. The design can operate at a frequency of 200 MHZ on a Virtex-5 with a latency of 8 cycles. Lamiaa S. A. Hamid [10] has presented a high speed generic Floating Point Unit (FPU) consisting of a multiplier and adder/Subtractor units is proposed. A novel multiplication algorithm is proposed and used in the multiplier implementation.

3. ORGANISATION OF WORK

In this section IEEE 754 single precision format, floating point adder, Floating point multiplier and floating point MAC unit are explained. MAC consists of a multiplier and an accumulator unit. Multiplier will multiply two numbers and result will be added to the number already stored in the accumulator.

3.1 STANDARD IEEE 754 FORMAT

The standard binary floating point format was issued by IEEE in 1985 [6]. It covers different types of floating-point formats (e.g. single, double), special coding representations (e.g. 0, $+\infty$, $-\infty$), rounding mechanisms, arithmetic operations, etc. The standard radix-2 binary floating-point representation can be written as in equation1 with s as the sign bit, M as the mantissa or fraction.



32 31 23 0

Figure 1. IEEE single precision floating point format

3.2 FLOATING POINT ADDER

Given two floating point numbers p1, p2 their sum can be denoted as p

P= p1+p2

$$=(-1)^{s1}.p1.2^{E1}+(-1)^{s2}.p2.2^{E2}$$
$$=(-1)^{s1}.p1.2^{E1}+(-1)^{s2}.p2.2^{E2-E1}2^{E1}$$
$$=(-1)^{s2}.(p1+p2\ 2^{E2-E1}).2^{E1} \quad (2)$$

The above equation describes addition of two floating point numbers. Figure 3 describes diagram of a floating point adder [2]. Adding two floating point numbers involves aligning both the numbers to bigger exponent and then adding aligned significands. The input exponents are subtracted and the result is used as tentative result exponent. The difference is used as amount of shift required to align both the mantissas. In significand processing preparation execution and normalization stages are basically involved. Preparation involves selecting the significand of the smaller exponent and aligning it to bigger one. In this stage if required one of the significands can be complemented this is actually done by inverting the selected significand and setting a carry-in in the adder. In execution stage actual addition of significands is realized. The normalization stage basically involves normalization, rounding and final normalization. The first normalisation is a single bit right shift or it can be n bit left shift that depends on the number of leading zeroes of the tentative result significand. The round and final normalisation are same as that in floating point multiplier. The resultant sign is obtained by xoring the sign bit of the two numbers. Finally output sign, mantissas and significand are obtained.

3.3 FLOATING POINT MULTIPLIER

Given two floating point numbers n1, n2 and after multiplication the result is n.

n=n1*n2

$$=(-1)^{s1}.n1.2^{E1}*(-1)^{s2}.n2.2^{E2}$$
$$=(-1)^{s1}+s2.p1.p2.2^{E1+E2} \quad (3)$$

In Figure 2 we present a general multiplier block diagram [2].The sign, exponent and mantissas are extracted from both the numbers respectively. Pipelining has been used for designing multiplier. The sign bits of both the numbers are xored. The 8 bit exponents are added and then bias is subtracted from it. Subtraction is easily achieved by adding carry in to the sum and then subtracting 128 from it by complementing most significant bit. For multiplying significands 48 bit multiplier is used. The 28 bits are considered as the most significand bits out of which 24 bits are the mantissa bits, 3 bits are for proper rounding, 1 bit is for range overflow. The result is then normalized for proper approximation to closest value. The approximation consists of a possible single bit right shift and corresponding exponent is incremented depending on b1 bit. The resultant sign, exponent and mantissas are obtained. The resultant sign, exponent and mantissas are then obtained. The figure shown below is simple floating point multiplier. In this paper three floating point multipliers have been designed using carry save, carry look ahead, ripple carry adder. Same flow is used for all of them .only for addition of exponents different adders are used.

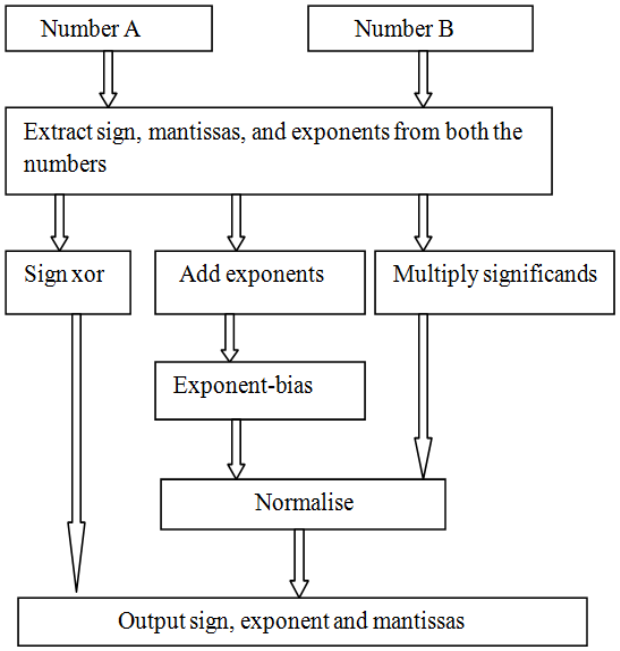


Figure 2. Block diagram of floating point multiplier

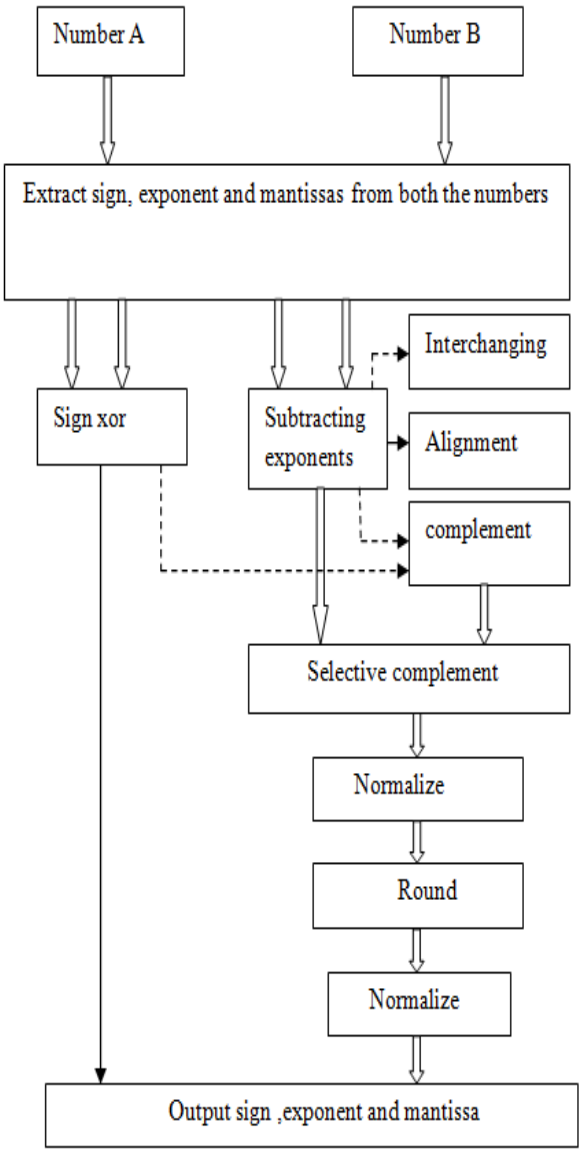


Figure 3. Block diagram of floating point adder

3.4 FLOATING POINT MAC UNIT

basically composed of adders, multiplier and an accumulator. The inputs which are given to the MAC are fetched from memory location and fed to the multiplier of block of MAC which performs multiplication and gives the result back to adder which will accumulate the result and store it in a memory location. Complete process is achieved in a single cycle. The design consists of 32 bit floating point adder and 1 register for memory location. A typical MAC unit consists of multiplier, adder and accumulator. The most important feature that differentiates general processor from digital signal processor is it's multiply and accumulate unit. Each DSP algorithm would require some form of Multiplication and accumulation system. This one is the most important block in DSP systems. Usually adders that are used are carry save, carry select, ripple carry adders because of their speed. The inputs of MAC are supposed to be fetched from memory location and then they are fed to the multiplier. Multiplier will multiply the inputs and it will give the results back to the adder and then the results of the multiplier are added to the previously accumulated results. Computation of most important formula i.e. b (n) x (n-k) is easily solved by this operation.

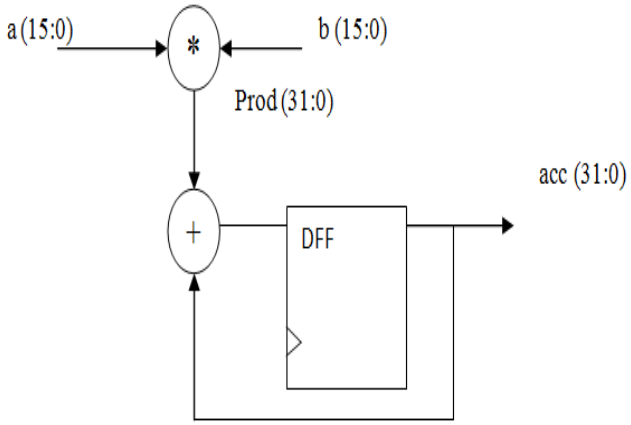


Figure 4. Block diagram of floating point MAC unit

4. RESULTS

The proposed Mac unit is implemented on Xilinx ISE design suite 12.4. Floating point adder is designed and then floating point multiplier is designed by using different adders such as carry save, ripple carry and carry look ahead and then performance analysis is done based on various parameters. In this paper pipelined multiplier, ripple carry multiplier, carry save multiplier, carry look ahead multiplier are designed. All the four multipliers are compared on the basis of power, thermal properties for commercial as well as Industrial applications. Then comparison is done on the basis of speed, area.

Table 1. Comparison of delay and area for floating point adders

Adders	Floating point adder(single cycle)	Floating point adder(pipelined)
No. of LUT FF	NA	292
Slice registers	501	292
No. of I/O	98	98
Speed	31.4 MHz	123.433MHz

Table describes the comparison for the following parameters such as LUT Flip flops, slice registers, No. of I/O, speed. Slice registers are 501 in single cycle and 292 in pipelined and I/O are same, speed of pipelined is more as compared to single cycle.

Table2. Power consumed by floating point adder

Commercial power	Floating point adder
Dynamic	5.47
Quiescent	555.46
Total	560.92

Table 3. Power consumed for different floating point multipliers

Floating point multipliers/power	quiescent (mW)	Dynamic (mW)	Total (mW)
Ripple carry	736.21	173.89	562.32
Carry save	460.87	163.90	296.91
Carry look ahead	612.26	170.72	441.54
Pipelined	582.67	21.99	560.60

Table describes the comparison of various floating point multipliers such as ripple carry, carry save, carry look ahead and pipelined.

Table 4.Comparison of speed/area for various floating point multipliers

Multipliers	Speed(MHz)	No. of slices	No. of bonded I/O B's
Ripple carry	23.2	858	96
Carry save	44.72	776	96
Carry look ahead	24.54	1201	96
pipelined	151.284	126	96

Table describes the comparison of various floating point multipliers such as ripple carry, carry save, carry look ahead and pipelined. Slices are highest for carry look ahead and speed is maximum for pipelined floating point

Table 5.Power consumed for simple MAC unit.

Power	Quiescent	Dynamic	Total
MAC	586.50	25.77	560.73

Table 6. Delay and Area for MAC unit.

Parameters	Delay	Area
MAC	31.4 MHz	501 out of 28800

Table 7. Speed and Area for MAC unit with different floating point multipliers.

Table describes Speed and area for various floating point

MAC	Speed	Area
Carry look ahead MAC	114.549MHz	313
Carry save MAC	115.955MHz	291
Ripple carry MAC	114.549MHz	291

MAC unit. Speed and area for various floating point MAC units is described. Speed is highest for carry save MAC

5. CONCLUSION

A FP adder and a FP multiplier are presented in this paper. Both are available in pipeline architectures and they are implemented in VHDL, are fully synthesizable with performance comparable to other available high speed implementations. Here in this paper three floating point multipliers are designed such as carry save, carry look ahead, ripple carry. Then pipelined floating point multiplier, carry save, carry look ahead, ripple carry are compared with each other and then results are analysed on the basis of area, delay, power. Then a complete MAC unit is designed based on above results and its FPGA implementation is done.

6. ACKNOWLEDGMENT

The author wants to express heartiest gratitude to Centre for Development of Advanced Computing (CDAC) Mohali faculty and staff members for their confidence in my efforts and the project. The author also wants to thank CDAC for providing equipments and laboratory to support this work.

7. REFERENCES

[1] Mohamed Al-Ashrafy, Ashraf Salem and Wagdi Anis, "An Efficient Implementation of Floating Point Multiplier, "proceeding of 2011 IEEE.

[2] Guillermo Marcus, Patricia Hinojosa, Alfonso Avila and Juan Nolazco-Flores, "A Fully Synthesizable Single-Precision, Floating-Point Adder/Subtractor and Multiplier in VHDL for General and Educational Use", Proceedings of the Fifth IEEE International Caracas Conference on Devices, Circuits and Systems, Dominican Republic.

[3] Xilinx Inc, ISE, at http://www.xilinx.com.

[4] Behrooz Parhami, Computer Arithmetic: Algorithms and Hardware Designs, 1st ed. Oxford: Oxford University Press, 2000

[5] John G. Proakis and Dimitris G. Manolakis (1996), "Digital Signal Processing: Principles. Algorithms and Applications", Third Edition.

[6] Patterson, D. & Hennessy, J. (2005), Computer Organization and Design: The Hardware/software Interface, Morgan Kaufmann.

[7] Mentor Graphics Inc, FPGA Advantage, at http://www.mentor.com/fpgaadvantage.

[8] IEEE Standards Board, IEEE-754, IEEE Standard for Binary Floating-Point Arithmetic, New York: IEEE, 1985.

[9] Lamiaa S.A.Hamid, Khaled A.Sheata, Hassan El-Ghitani, Mohamed Elsaid (2010)," Design of Generic Floating Point Multiplier and Adder/Subtractor Units", in proceedings of the 12th IEEE international Conference on computer modeling and Simulation.

[10] Carlos Minchola, Martin Vazquez, Gustavo sutter (2011),"A FPGA-754-2008 DECIMAL64 Floating point adder/subtractor" , "proceeding of 2011 IEEE.

[11] S.V.Siddamal, R.M.Banakar and B.C.Jinaga, "Design of high speed multiplier," 4th IEEE International symposium on Electronic Design, test and Application, pp.285-289, 2013.

[12] Michael Nachtigal, Himanshu Thapliyal and Nagarajan Ranganathan,, "Design of a Reversible Floating-Point Adder Architecture," proceeding of the 2011 11th IEEE International Conference on Nanotechnology Portland Marriott August 15-18, Portland, Oregon, USA, pp.451-456,2011.

[13] Rajit Ram Singh,Vinay Kumar Singh and Geetam S Tomar,(2011) "VHDL environment for Floating point Arithmetic Logic Unit – ALU design and Simulation," in proceeding of International Conference on Communication Systems and Network Technologies, pp 469-472,2011.