National Conference on Innovative Paradigms in Engineering & Technology (NCIPET-2012)
Proceedings published by International Journal of Computer Applications® (IJCA)

Single Precision Floating Point Fft

Ujwal S. Ghate
SRKNEC Nagpur, India

ABSTRACT
In this paper the design and implementation of 32 bit IEEE 754 single precision floating point FFT architecture is proposed. Usually for FFT calculation the sequential circuits use for mantissa adjustment which is somewhat tedious job So, new approach is define for calculating FFT in pure combinational circuits form. The simulation result are compare with the quartus II and Active HDL software also it is cross verified with Matlab . The algorithm is implemented on FPGA.

Keywords
IEEE Floating-point, FPGA , FFT,HDL

1. INTRODUCTION
Fast Fourier Transform (FFT) plays an important role in many signal and image processing, data analyzing for vibration sensors, frequency measurement of earthquakes, and telecommunication systems such as WiMax technology which presents both wide bandwidth and wireless solutions. In real time applications, it is necessary to obtain and process the input data as fast as possible to be able to reach the result almost simultaneously. Although ASIC solutions always offer fastest and low power solutions for real time applications, they are unique designs for a specific application. Therefore redesign process of an ASIC for a new application requires much more money and time when comparing with field programmable chips. FPGA solutions also provide flexible design, low cost, and faster time-to-market features besides allowing parallel process implementations.
Floating point numbers have ability to represent a good approximation and dynamic range representations of the real numbers, so that floating point algorithms are frequently used in modern applications, which require millions of calculations per second, such as image processing and speech recognition. In this paper, firstly, the realized algorithms of the necessary arithmetic operations used in FFT implementation are presented. Next, these design blocks are used to realize the mathematical expression of the FFT and compared with the similar ones in the literature from structural and performance point of view.

2. FLOATING POINT ARITHMETIC ALGORITHMS
The single precision numbers in the binary IEEE standard are formed as shown in Fig.1. The most significant bit is the sign bit, which indicates a negative number if it is set to 1. The following field denotes the exponent with a constant bias added to it. As shown in Fig.1, the remaining part of the number is normalized to have one non-zero bit to the left of the floating point

31	23	0
S	e + bias	f
Sign	Biased exponent	Significant s=1.f (the 1 is hidden)

Fig 1 Format of IEEE single floating point number

Therefore, the value given by the standard format can be expressed using following expression.

$$m = (-1)^{Sign} \times 2^{e} \times 1.f$$

The range of single precision floating point number varies from -3.4028236 e+38 to -1.1754944 e-38 and from +1.1754944 e-38 to +3.4028236 e+38.

2.1 Floating point addition and subtraction
Fig 2 shows the design flow chart of the floating point addition and subtraction algorithm implemented. These algorithms are similar to the ones realized in many architecture. Let *A1* and *A2* represent two floating point numbers; *Aadd* represents the addition of both number; and *Aminus =A1-A2*. *Aminus* can be re-written as *Aminus= A1+(-A2)*. The subtraction process is converted to addition form by inversing the sign bit of F2. For this reason, only addition algorithm is elaborated here. Addition and subtraction algorithms are realized in three steps. Ai represents the number; Si is the sign, ei is exponent and fi is the fraction part of any number. Lets define the inputs as *A1=(s1,e1 ,f1)* and *A2=(s2,e2,f2)*. The result is represented as *Fans=(sans,eans ,fans)= A1+A2* or *A1 +(- A2)*
The algorithm steps are as follows:

1. Step:
If absolute value of *A1* is smaller than *A2*, *A1* and *A2* are interchanged. The right shift amount of *f2* is calculated by subtracting *e1* from *e2*. (Sign)_1_ (Mantissa) is added to the bits after the sign bit (*1.f1*) ve (*1.f2*).

2. Step*:*
(1.f1) is shifted to the right by the amount of (*e1- e2)*. If the sign bits are equal, then (*1. f1*) and (*1. f2*) are added, if not (*1.f2*) is subtracted from (*1.f1*). The sign of the resulting number *sans* is the sign of the bigger *f* number.

3. Step*:*
fans is shifted to the left until the first bit becomes _1_, and amount of the shift is calculated. *e*ans is obtained by subtracting the amount of shift from *e1*.

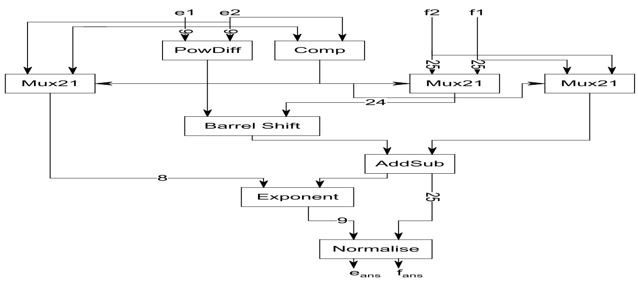

Fig2: Adder Substractor Unit

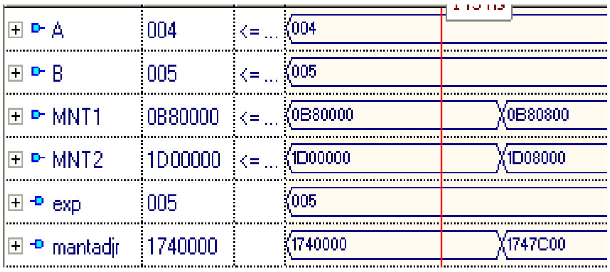

Fig2: Result of Adder Substractor Unit

Table 1

	Input		Actual Output(Before Normalize)	Result
Dec. No.	A(23)	B(-52)	-29	-29
Exp	4	5	5	5
Man	0B8	1D0	1E8	1E8

2.2 Floating point multiplication

Floating point multiplication shown in Fig 3 is similar to the integer multiplication. Therefore FP multiplication is easier than FP adding or subtracting algorithms here. It is realized in three steps as well.To make it easy, the algorithm never tests the illegal numbers or negative zero cases. The inputs are same as before, *A1=(s1, e1, f1)* and *A2=(s2, e2, f2)*.The result will be Aans = (sans, eans, fans)= A1* A2. The algorithm steps will
be as follows:

1. Step:

Exponent parts, *e1* and *e2* are added; the resulting number is appointed as *eans*. _1_ is added to the beginnings of *f1* and *f2*, yielding (*1.f1*) and (*1.f2*).

2. Step:

(*1.f1*) and (*1.f2*) are multiplied and the first 23 MSB bits out of the resulting 45 bits is appointed as the final result, fans . The sign bit of the final number, sans is obtained by EXOR_ing the two numbers.

3. Step:

fans is shifted to the left until the first bit becomes _1_, and amount of the shift is calculated. *eans* is obtained by adding the amount of shift from *e1*.

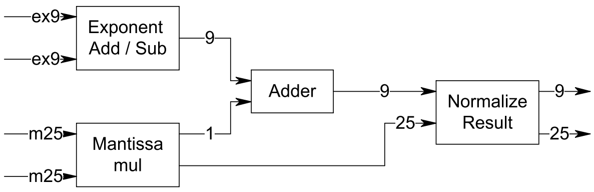

Fig3: Multiplication unit

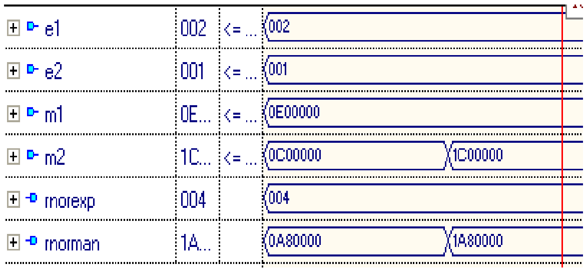

Fig4: Result Multiplication unit

Table 2

	Input		Actual Output	Result
Dec. No.	A(7)	B(-3)	-21	-21
Exp	2	1	4	4
Man	0E0	1C0	1A8	1A8

3. FLOATING POINT FFT DESIGN

The basic radix-2 butterfly unit is as shown in Fig 5 and corresponding floating point diagram is shown in Fig 6

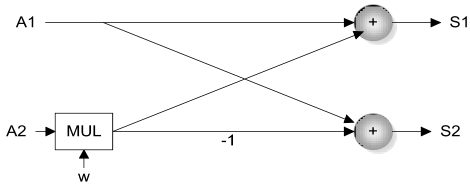

Fig 5: Radix-2 butterfly unit

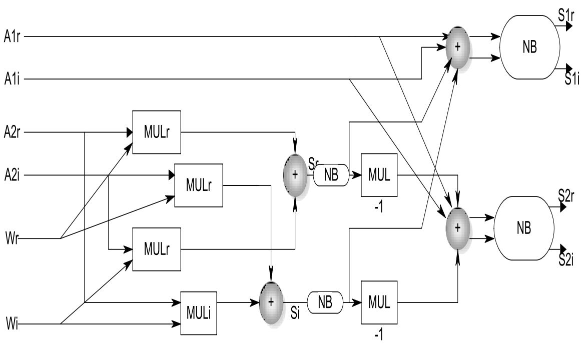

Fig6: Floating Point Butterfly Structure

The fig 8 shows the result of radix-2 single butterfly stage here if we use fsm for giving input then clk (clock) and rst (reset) is used.

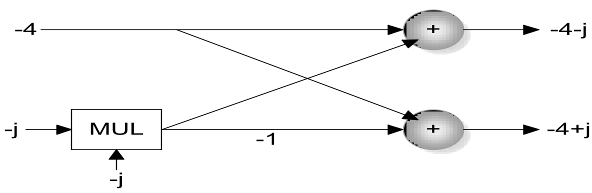

Fig7: Testing fft butterfly (one real and one img. number)

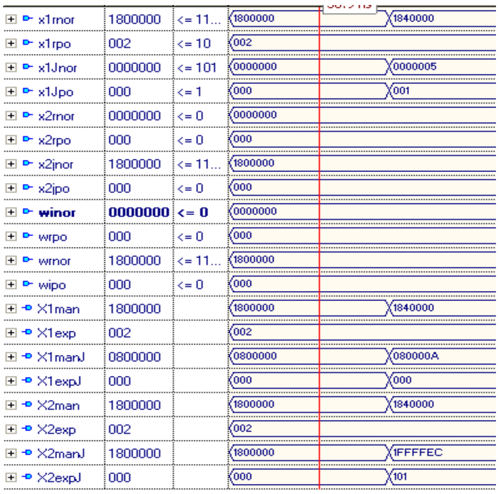

Fig8: Result of butterfly Fig 4

Table 3

	Input		Result			
			X1		X2	
Dec. No.	A(-4)	B(-j)	-4	-j	-4	+j
Exp	2	0	2	0	2	0
Man	180	180	180	180	180	080

4. CONCLUSION

In this paper the design and implement a 32-bit IEEE 754 single precision floating point FFT on FPGA. The Design is in fully combinational so there is no operating frequency limit i.e. output only depends on input only. For power calculation and layout we can use synopsis or cadence tool.

for optimization pipeline and parallel processing and block rearrangement can be used .

5. REFERENCES

[1] E. O. Brigham, *The fast Fourier transform and its applications*, Prentice Hall, 1988.

[2] J. G. Pmakis, *Digital signal processing: principles algorithms, and applications*., Prentice-Hall Intemational, 1996.

[3] H. Hu, T. Jin, X. Zhang, Z. Lu, Z. Qian, _ A Floating-point Coprocessor Configured by a FPGA in a Digital Platform Based on Fixed-point DSP for Power Electronics_, *IEEE IPEMC_2006*

[4]ShengmeiMou,XiaodongYang "Design of a high –speed FPGA-based 32-bit floating point FFT Processor" 2007 IEEE.

[5] Floating –point FFT Processor (IEEE 754 single presion Radix 2 core,White Paper from altera)

[6] Shiqun Zheng Dunshan Yu,"Design and implimention of a parrel real-time FFT Processor",7 th IEEE conference on Solid –State and Integated Circuits Technology,Vol.3,pp,1665-168,Oct2004.

[7] Bin Zhou ,David Hwang " Implementations and Optimizations of pipeline FFTs on Xilinx FPGAs.(2008 International Conference on Reconfigurable computing and FPGAs)

[8] M.Hasan & T. Arslan "Coefficient Memory Addressing scheme for VLSI Implementation of FFT Processor"[IEEE 2000 Scotland].