International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 6 (2018) pp. 4588-4592
© Research India Publications. http://www.ripublication.com

Design of Complex IEEE Floating Point Multiplier for FFT Applications

J.Abirami[1]
Student (final year), Department of Electronics and Communication Engineering, JEPPIAAR SRR engineering college, Chennai-603103 India.

Mrs. SharmilaHemanandh[2]
Assistant Professor M.E, Department of Electronics and Communication Engineering, JEPPIAAR SRR engineering college, Chennai-603103 India.

V.Divyashree[3]
Student (final year), Department of Electronics and communication engineering, JEPPIAAR SRR engineering college, Chennai-603103, India.

R.Krithika[4]
Student (final year), Department of Electronics and communication engineering, JEPPIAAR SRR engineering college, Chennai-603103, India.

ABSTRACT:

The DFT (Discrete Fourier Transform) is the family of Fourier transform and most important discrete transform, used to perform Fourier analysis in digital signal processing. DFT is purely discrete (i.e) discrete-time data sets are converted into a discrete-frequency representation. This is in contrast to the DTFT that uses discrete time, but converts to continuous frequency.DFT is implemented by FFT (Fast Fourier Transform) which accelerates and perform the DFT process in an efficient and optimal way. For this reason DFT is used in partial differential equations, convolutions, data compression, polynomial multiplication etc.

Multipliers play a major role in almost all DSP application. There is a requirement of efficient complex multiplier, since most of the digital signal processes involves operations on complex numbers. Hence a design of floating point multiplier (one of complex multiplier used to multiply floating numbers) for FFT processor using CSD (canonical signed digit) technique is being proposed in this paper. The design is analyzed and simulated by Quartus II v9.1. The proposed system improves speed, accuracy.

INTRODUCTION

The fundamental and the core of all the digital signal processors (DSPs) are its multipliers, and the speed of the DSPs is mainly determined by the speed of its multiplier. Multipliers are key components of many high performance systems such as microprocessors, FIR filters, digital signal processors, etc [1]. Performance of a system is generally determined by the performance of the multiplier because the multiplier is generally the slowest element in the system. Since multiplication dominates the execution time of most DSP application there is need of high speed multiplier which can increase the performance of the overall system or processor [1].

Operations involving floating point numbers have taken on greater significance in recent years which are performed by floating point multipliers [2,3]. Partial product generation and accumulation are the two major steps in multiplication. The speed of multiplication increases by reducing the number of partial products or accelerating the accumulation. This can be efficiently achieved by CSD recoding technique. CSD is a redundant representation that contains no adjacent nonzero digits [4]. Since it recodes a number with minimum number of non-zero digits it also leads to reduction in number of partial products since partial products depends on number of non-zero digits. Hence this technique reduces carry propagation & complexity and provides high level of accuracy and precision.

DISCRETE FOURIER TRANSFORM (DFT):

The discrete Fourier transform is used to transform continuous function to discrete function, so that the frequency analysis of discrete time signal can be performed on digital system. Fourier analysis consists of multiplying the waveform to be analyzed by a digital representation of sine and cosine waveforms of the test frequency. DFT allows the use of any integer number of samples in the analysis. The disadvantages of the DFT technique are, it requires each harmonic to be calculated separately, which requires much more processing power. Another drawback in the DFT is that the computation of each sample of DFT involves a large no of calculations and when large number of samples is required, the number of calculations will further increase.

FAST FOURIER TRANSFORM (FFT):

In order to overcome this drawback, a number of methods or algorithms have been developed to reduce calculations .The various methods developed to compute DFT with reduced number of calculations are collectively called Fast Fourier Transform. The FFT or Fast Fourier Transform is a method of calculating harmonics not one at a time, but as a group, using a special algorithm. The FFT requires much less processing power than a DFT for the same number of harmonic results. An FFT however, requires the number of samples being analyzed to be a binary number e.g. a power of two. The computational efficiency is achieved if divide and conquer approach is adopted .This approach is based on the decomposition of an N-point DFT into successively smaller DFTs .This basic approach leads to a family of an efficient computational algorithm known collectively as FFT algorithms. It is used to take advantage of the periodicity and symmetry property of the complex number Such decomposition ,if properly carried out, can result in a significant savings in the computational complexity given by the total number of multiplication and the total number of additions needed to compute all N DFT samples [5,6].

RADIX-2 FFT:

For radix-2FFT, the value of N should be such that, $N=2^m$ [8], so that the N-point sequence is decimated into 2-point sequences and the 2-point DFT for each decimated sequence is computed. From the results of 2-point DFTs, the 4-point DFTs can be computed. From the result of 4-point DFTs, the 8-point DFTs can be computed and so on until we get N-point DFT.

There are basically two types of FFT algorithms. They are,

1. Decimation in Time
2. Decimation in frequency

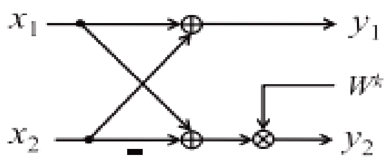

Figure 1. Butterfly diagram of DIT FFT

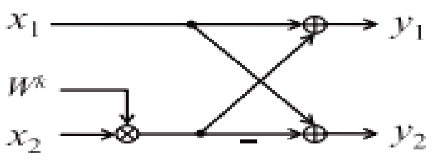

Figure 2. Butterfly diagram of DIF FFT

DECIMATION IN TIME (DIT) RADIX-2 FFT:

The N-point DFT of a sequence x(n) converts the time domain N-point sequence x(n) to a frequency domain N-point sequence X(k). In Decimation In Time (DIT) algorithm, the time domain sequence x(n) is decimated and the smaller point DFTs are performed [8]. The results of smaller point DFTs are combined to get the result of N-point DFT.

In DIT radix-2 FFT, the time domain sequence is decimated into 2-point sequences. For each two point sequence, the two point DFT is computed. The results of 2-point DFTs are used to compute one 4-point DFT. The results of 4-point DFTs are used to compute 8-point DFTs. A pair of 4-point DFT results is used to compute one 8-point DFT. The process is continued until we get N-point DFT.To decompose an N point time domain signal into N signals each containing a single point, each decomposing stage uses an interlace decomposition, separating the even and odd-indexed samples. In DIT the twiddle factor is first multiplied and then the addition and subtraction is performed.

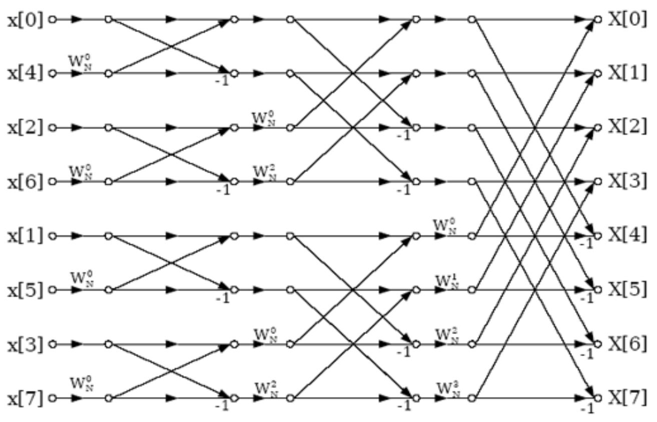

Figure 3. 8 point radix 2 DIT FFT

Here W_N is the twiddle factor . x(n) is the input sequence and X(k) is the output sequence. In the DFT computation scheme, the DFT samples X(k) appear at the output in a sequential order while the input samples x(n) appear in a different order: a bit-reversed order. Thus, a sequentially ordered input x(n) must be reordered appropriately before the fast algorithm can be implemented.(i.e)The input is bit- reversed order and the output is natural order

DECIMATION IN FREQUENCY (DIF) RADIX-2 FFT:

In decimation in frequency algorithm the frequency domain sequence X(k) is decimated ,(but in decimation in time algorithm, the time domain sequence x(n) is decimated) [8]. In this algorithm, the N-point time domain sequence is converted to two number of N/2 point sequences. Then each N/2 point sequence is converted to two numbers of N/2 point sequences. Thus we get 4 numbers of N/4 point sequences. This process is continued until we get N/2 numbers of 2-point sequences. Finally the 2-point DFTs of N/2 number of 2-point sequences will give N samples, which is the N-point DFT of the time domain sequence

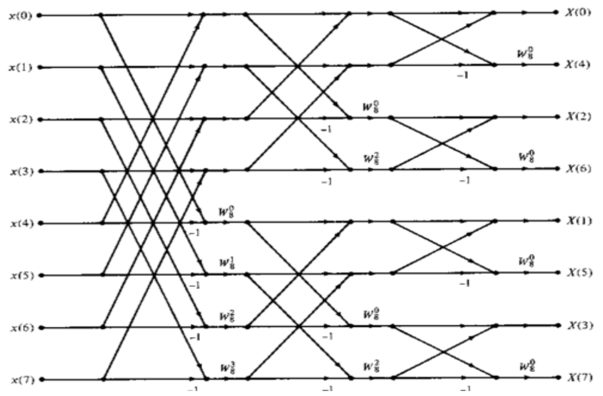

Figure 4. 8 point radix 2 DIF FFT

Decimation-In-Frequency is a popular form of FFT algorithm. Here the sequence X(k) is divided into smaller and smaller subsequences, hence the name decimation in frequency, Initially the input sequence x(n) is divided into two sequences x1(n) and x2(n) consisting of the first n/2 samples of x(n) and the last n/2 samples of x(n) respectively .

In DIF FFT computation scheme, the DFT samples X(k) appear at the output in a non -sequential order (bit reversed order) while the input samples x(n) appear in a sequential order. (i.e)The input is sequential order and the output is in bit reversed order. Here the twiddle factor is multiplied after addition or subtraction is performed. Both DIT-FFT and DIF-FFT have the identical computation complexity. (i.e) there are total L stages and each has N/2 butterfly computation. Each butterfly computation has 1 multiplication and 2 additions. Both DIT-FFT and DIF-FFT have the characteristic of in-place computation.

FLOATING POINT MULTIPLIER:

Floating point multiplier using conventional technique involves traditional multiplication of numbers. As mentioned earlier, multiplication is a major step in all fundamental digital signal processing applications hence it should be faster and simple and should have less complexity. Multiplication speed determines the overall performance of an FFT processor.

Apart from speed, power is an important parameter because higher power consumption reduces thermal stability and increases the cooling cost. Furthermore, many applications are battery powered and demand on low power consumption. Hence, low power design is important. Smaller area reduces the overall cost and complexity of the FFT system and reduced maximum frequency will increase the speed of the system. Hence speed, power and area are three most significant parameters.

Although conventional method is simple it is not faster since it gives increased number of non zero elements due to which partial products also increases. Conventional technique can go wrong in addition part, which sets another drawback. Synthesis of the conventional technique in floating point multiplication shows increased dynamic power of the multiplier and also has more number of logic elements. Hence conventional technique fails in fulfilling the three major aspects of a floating point multiplier for FFT applications

CANONICAL SIGNED DIGIT TECHNIQUE:

Canonical signed digit is a special technique to encode a value in a signed digit representation which itself is a unique representation and allows one number to be represented in many ways [10]. The representation uses a sequence of one or more of the symbols -1, 0,-1 with each position possibly representing the addition or subtraction of a power of 2.

The important characteristics of the CSD presentation are:

1. CSD presentation of a number consists of numbers 0, 1 and -1.
2. The CSD presentation of a number is unique.
3. The number of nonzero digits is minimal.
4. There cannot be two consecutive non-zero digits.
5. To implement a negative integer, just compute the CSD value of the positive integer and invert the sign of each digit.

CONVERSION OF BINARY NUMBER INTO CSD NUMBER:

Consider a number and find the binary presentation n of the number.

Example 1

Take for example a number 287, which is 1 0001 1111 in binary representation. (256 + 16 + 8 + 4 + 2 + 1 = 287)

Step 1;

10 001 1111

Starting from the right (LSB), if you find more than non-zero elements (1 or -1) in a row, take all of them, plus the next zero. (If there is no zero at the left side of the MSB, create one there) the first part of this number is

Step 2;

01 1111

Add 1 to the number (i.e. change the 0 to 1, and all the 1's to 0's), and force the rightmost digit to be -1.

Step 3;

01 1111->10 000-1

You can check that the number is still the same: 16 + 8 + 4 + 2 + 1 = 31 = 32 + (-1). Now the number looks like this

Step 4;

1 0010 000-1

Since there are no more consecutive non-zero digits, the conversion is complete. Thus, the CSD presentation for the number 287 is 1 0010 000-1, which is 256 + 31 - 1.

From the above example of conversion it can be seen that the binary form of the integer 287 has more number of non zero elements whereas the canonical signed digit form of 287 has reduced number of non zero elements.

CSD multiplication technique involves converting the multiplier into canonical form and then multiplying the multiplicand by canonical multiplier. Since the CSD form of any integer has reduced number of non-zero elements the multiplication results in reduced number of partial products which makes the addition easier and also the overall result of multiplication also has reduced number of non-zero elements.

Hence canonical signed digit technique of multiplication increases the speed of multiplication [10]. Further synthesis shows that due to reduced partial products, canonical signed digit multiplier has reduced dynamic power consumption and reduced area. In addition to fulfilling these three major aspects of floating point multiplier , CSD technique is proven to be useful in implementing multiplier with reduced complexity , because the cost of multiplication is a direct function of the number of non-zero bits in the multiplier [11].

The Floating point multiplier using both conventional technique and canonical signed digit is being implemented in FFT algorithm and the FFT processor is modeled in VHDL and synthesized using Quartus II v9.1. The following table shows the comparison of parameters for both conventional and proposed technique. The table proves that proposed system reduces the area by reducing the number of logic elements, reduces power consumption and increases the speed by reducing the maximum frequency.

Table 1: Performance Comparison of FFT with conventional multiplier and CSD multiplier

S.no	Parameters	FFT with conventional multiplier	FFT with CSD multiplier
1.	Number of logic elements	6,918	6,516
2.	Power	201.6mW	183.1mW
3.	Max frequency	72.92MHZ	72.58MHz

Bar Chart Analysis

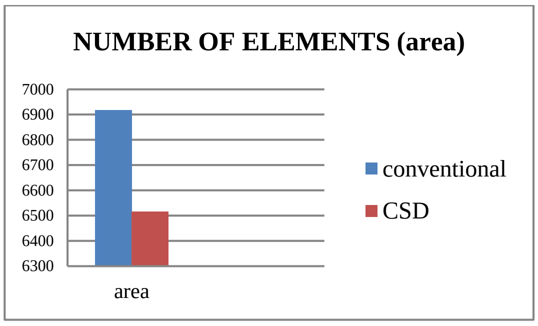

Figure 5. Area comparison of conventional with CSD

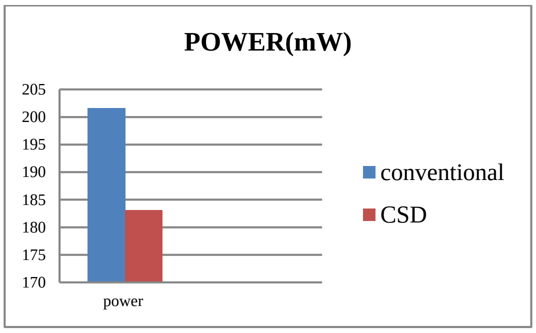

Figure 6. Power comparison of conventional with CSD

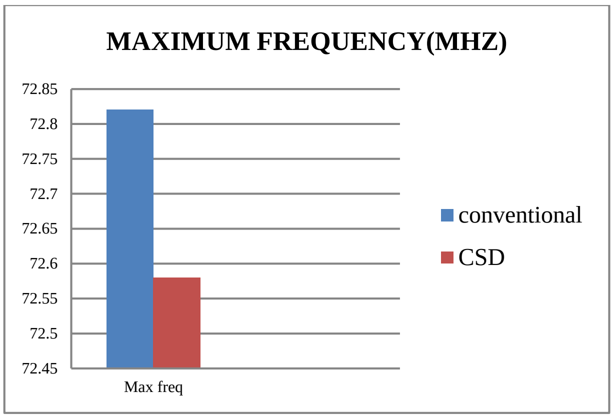

Figure 7. Speed comparison of conventional with CSD

Fig 5, 6 and 7 clearly illustrates the performance of both existing and proposed method. Fig 5.compares the number of logic elements to clearly show that conventional technique has more number of logic elements which eventually increases the area. Whereas the proposed FFT processor implemented using canonical signed digit multiplier exhibits reduced number of logic elements hence both area as well as cost decreases [14]. Fig 6 Compares the power consumption which is a major factor since power consumption determines thermal stability[10]. Increase in power consumption will lead to thermal instability which is undesired. FFT implemented using the conventional technique consumes more dynamic power, while FFT using CSD technique consumes less power consumption, hence resulting in good thermal stability. Fig7. Illustrates maximum frequency of both proposed and existing system. Maximum frequency is also a significant factor since it is a measure of speed of the system [14]. The maximum operating frequency of conventional technique is higher than the proposed canonical signed digit based FFT processor. Hence the speed of the proposed CSD based

FFT algorithm is higher than the conventional technique.

CONCLUSION:

In this paper complex floating multiplier using canonical signed digit technique is been proposed. The proposed multiplier is implemented in FFT and the results are compared with conventional method. The synthesis results show that the proposed multiplier saves area up to 5.8%. 9.1% reduction in power is achieved by using CSD multiplier. Hence the proposed multiplier proves to be a much effective one for floating point multiplication in FFT applications. It can also be used in differential equations, filters and in other digital signal processing applications.

REFERENCES:

[1] Sunesh n.v sathishkumar p, “ Design and implementation of fast floating point multiplier unit”, International conference on VLSI systems, architecture, technology and applications (VLSI-SATA) (2015)

[2] Ms. Anuja a. bhat1 , Prof. Mangesh n. Thakare, “Review on 32-bit IEEE 754 complex number multiplier based on FFT architecture using boothalgorithm”,(2017) International Journal Of Engineering And Computer Science Volume 6 Issue 2 eb.2017, Index Copernicus Value (2015)

[3] Vojin G. Oklobdzija, “An integrated multiplier for complex numbers”, Journal for VLSI systems,(2014)

[4] Ms. Pallavi ramteke dr. n. n. mhala prof. p. r. lakhe, “Single precision floating point multiplier using shift and add algorithm”, international journal of innovative research in advanced engineering (ijirae) volume 1 issue 5 (2014)

[5] k.baboji, sriadibhatla.sridevi,“FFT implementation using floating point fused multiplier with four term adder”, 978-1-5386-1716-v/17/$31.00 ©2017 IEEE

[6] Md.zakir hussain, kazinikhat parvin. zeba fatima mir ilyas ali “Performance efficient FFT processor design”,International conference on global trends in signal processing, information computing and communication (2016)

[7] J.m -ashrafy, a. salem, w. anis Saudi, “An efficient implementation of floating pointmultiplier” International Electronics, communications and photonics conference (SIECPC), (2011)

[8] Ranbeer rathore navneet kaur “Comparison study of DIT and dif radix-2 FFT algorithm”**,** international journal of computer applications (0975 – 8887) volume150no.7, (2016)

[9] Shengmei mouand xiaodong yang, “Design of a high-speed FPGA-based 32-bit floating- point FFT processor”,eighth ACIS International conference on software engineering, artificial intelligence, networking, and parallel/distributed computing (2007)

[10] Vishwanath b.r.1 Theerthesha.t.s, “Multiplier using canonical signed digit code”, International journal for research in applied science & engineering technology (IJRASET), (2015)

[11] Rajdeep kaur and Tarandip singh, “Design of 32-point mixed radix FFT processor using CSD multiplier”, fourth international conference on parallel distributed grid computing(2016)

[12] Michael a. Soderstrand, “CSD multipliers for FPGA DSP applications”, 0-7803-776 1-310310, (2003)

[13] Aiping hu, a.j.al-khalili, “Comparison of constant coefficient multipliers for CSD and booth recoding”, 0-7803-7573-4/02 ,(2002)IEEE

[14] Dr.Uma B V, Harsha R Kamath, Mohith S, Sreekar V, Shravan Bhagirath, “Area and time optimized realization of 16 point FFT and IFFT blocks by using IEEE 754 single precision complex floating point adder and multiplier”, International conference on soft computing techniques and implementations- (ICSCTI) (2015)