International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

High performance Radix-4 FFT using Parallel Architecture

Pooja Swamy[1], R. Pavan Kumar[2]

[1]M.Tech. Student, Lingaraj Appa Engineering College, Bidar, Karnataka

[2]Defence Electronics Research Laboratory, Hyderabad

Abstract: ***In this paper, we propose a design architecture of an efficient Radix-4 FFT algorithm using parallel architecture. This parallel architecture plays an important role in the FFT computation speed of data samples. The proposed algorithm has a better power and area consumption compared to the conventional Radix-4 FFT Algorithm. By the repeated use of twiddle factors, algorithm reduces the complexity and memory consumption in terms of hardware point of view.***

Keywords: DIF, DIT, FFT, DFT.

1. Introduction

The Fourier Transform is a widely used method in signal processing to estimate spectral content of any signal. The Fourier Transform when applied to an aperiodic discrete signal rather than a continuous signal is called Discrete Time Fourier Transform (DTFT). But DTFT of an aperiodic discrete signal is continuous in frequency domain. Hence to use DTFT in computers, we sample the DTFT. This sampled DTFT is called Discrete Fourier Transform (DFT). Fast Fourier Transform (FFT) is an efficient algorithm for computing DFT. The DFT is defined by equation (1)

$$X(K) = \sum_{n=0}^{N-1} x(n) e^{\left(\frac{-j2\Pi kn}{N}\right)} \quad (1)$$

Where, k = 0 to $N-1$ and N is the length of the DFT. Calculating the above equation requires N^2 Complex Multiplications and N*(N-1) additions. But by using FFT algorithms, the amount of computation involved is reduced significantly.

A Fast Fourier Transform (FFT) is a fast algorithm for computing the Discrete Fourier Transform (DFT). The development of FFT algorithm has a tremendous impact on computational aspects of signal processing and applied science. All the FFT algorithms use a basic computing core structure called as butterfly. Radix-2, Radix-4, Radix-8 are common butterfly structures. Radix-2 FFT algorithm is used for data vectors of lengths in powers of two. They proceed by dividing the DFT into two DFTs of lengths, which are half of original length and iterating. The number of stages required to obtain the DFT depends on the number of data points in the sequence. There are several types of radix- 2 FFT algorithms, the most common being the Decimation-In-Time (DIT) and the Decimation-In-Frequency (DIF).

To compute the DFT of an N data sample, where N is a power of four, one can always use the Radix-2 algorithm, but in such cases it is more computationally efficient to use a Radix-4 FFT algorithm. Radix-4 FFT algorithm is also implemented in the same manner as Radix-2 but instead of dividing the DFT into two DFTs, DFT is divided into four DFTs of lengths, which are quarter times the original length and iterating. This reduces the number of stages required to find the DFT of a given sequence. With this advantage of reduced number of computing stages, the number of complex multiplications and additions that are required to be performed the N point DFT also reduces. Radix-4 algorithm also requires less area and time compared to Radix-2 algorithm.

The rest of this paper is organized as follows: Section (2) describes existing traditional system. Next in Section (3) the Proposed Radix-4 DIF FFT algorithm is discussed. In Section (4) Parallel architecture is discussed and in next Section (5) Complex Multiplier is discussed. In Section (6) Simulation result is given and then we conclude the paper in Section (7).

2. Existing System

The existing traditional system computes DFT using Radix-2 FFT algorithms. In the Radix-2 FFT, base is equal to 2 and representation is 2^v where v represents the number of stages. It consist of two types of decimation domains, they are decimation in time and decimation in frequency. In Radix-2 FFT algorithms, the core butterfly structure takes two inputs and produces two outputs and outputs are arranged in bit reversal order. The number of stages are 2^{v-1} and each stages can be divided as $(N/2)^V$. Each stage includes a twiddle factor which is given below.

$$\text{Twiddle Factor} = \exp(-j2\pi kn/N)$$

In Radix-2 FFT, the total computational load is $N/2*\log_2 N$ complex multiplications and $N*\log_2 N$ complex additions.

3. Proposed Radix-4 DIF FFT Algorithm

The Radix-4 DIF FFT algorithm breaks N-point data samples into N/4 point data samples, then into N/16 point a so on. This iterative breaking down of data samples finally lands into a four data point computation which is termed as basic butterfly structure. Fig (1) shows the basic butterfly

architecture. In DIF operation principle, Radix-4 FFT inputs are in normal order and outputs are in digit reversed order. At the input side, samples are taken from time domain index which are processed with butterfly and twiddle factor and then produces output in frequency domain samples.

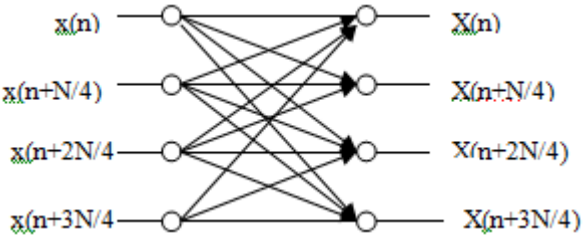


Figure 1: Basic Radix-4 FFT Butterfly

The following equation shows the basic computation of Radix-4 FFT algorithm.

$$X(k) = \sum_{n=0}^{N-1} x(n)W_N^{nk}$$

$$= \sum_{n=0}^{N/4-1} x(n)W_N^{nk} + \sum_{n=N/4}^{2N/4-1} x(n)W_N^{nk} + \sum_{n=2N/4}^{3N/4-1} x(n)W_N^{nk} + \sum_{n=3N/4}^{3N/4-1} x(n)W_N^{nk}$$

$$= \sum_{n=0}^{N/4-1} [x(n) + x\left(n+\frac{N}{4}\right)W_N^{(N/4)k} + x\left(n+\frac{2N}{4}\right)W_N^{(2N/4)k} + x\left(n+\frac{3N}{4}\right)W_N^{(3N/4)k}]W_N^{nk} \quad (2)$$

Then the above equation can be divided into N/4-point DFTs as shown below

$$X(4k) = \sum_{n=0}^{N/4-1} [+(n) + x(n+N/4) + x(n+3N/4) + x(n+3N/4)]W_{N/4}^{nk}$$

$$X(4k+1) = \sum_{n=0}^{N/4-1} [+(n) - jx(n+N/4) - x(n+3N/4) + jx(n+3N/4)]W_{N/4}^{nk}W_N^n$$

$$X(4k+2) = \sum_{n=0}^{N/4-1} [x(n) - x(n+N/4) + x(n+3N/4) + x(n+3N/4)]W_{N/4}^{nk}W_N^{2n}$$

$$X(4k+3) = \sum_{n=0}^{N/4-1} [x(n) + jx(n+N/4) - x(n+3N/4) - jx(n+3N/4)]W_{N/4}^{nk}W_N^{3n} \quad (3)$$

Each N/4-point is a sum of four input sample such as *x(n), x(n+N/4), x(n+2N/4)* and *x(n+3N/4)* are multiplied with by (+1, -1, j and -j). The sum is multiplied by a corresponding twiddle factor (W^0, W^1, W^2, and W^3).

The above expressions are the basic building blocks of Radix-4 FFT algorithm. From above fig (1), it can be shown that each radix-4 FFT butterfly requires 3 complex multiplications and 8 complex additions. This process is repeated again and again until the resulting sequence are reduced to one point sequence. Generally the number of stages required for the computation of N point DFT is $\log_4 N$. Thus the total number of complex multiplications are $3*(N/4)*\log_4 N$ and additions are $8*(N/4)*\log_4 N$. We can see that these computations are very less in comparison to Radix-2 FFT algorithm.

4. Parallel Architecture

The below fig (2) shows the overall parallel architecture

Figure 2: Parallel Architecture

It mainly consists of a Register1 (R1), Butterfly1 (B1), Register2 (R2) and Butterfly2 (B2). This approach would mean that all butterfly computations can be performed in parallel. All butterfly computation in a stage is performed in parallel and then at the end of the stage, the results are gathered. Now all nodes perform computation on the result of the first stage in parallel and output of the second stage are gathered again and so on.

Once all the computations of all the butterflies of stages are over, the memory locations can be freed. Then the same butterfly structures can be reused for the next computations of the next stage. This is an important structural advantage as the memory used for the first stage is getting reused for the next stage. Hence this uses very less resources in terms of hardware point of view.

As we can see in the simulation result that the original output of FFT and our proposed parallel architecture gives the same output for a sinusoidal signal input signal. But at the same time our proposed work takes fewer amounts of resources giving the same results for high speed real time signal processing applications.

5. Complex Multiplier

Complex Multipliers used in the hardware language such as Verilog is a high performance, optimized digital structures. All operands and the results are represented in signed two's complement format. The operand width and design widths are customizable. The complex multipliers have two basic architectures to implement complex multiplication.

Given the two complex operands $a = a_r + ja_i$ and $b = b_r + jb_i$, the output is $p = ab = p_r + jp_i$. Direct implementation requires four real multiplications:

$$p_r = a_r b_r - a_i b_i$$

$$p_i = a_r b_i - a_i b_r$$

By exploiting that

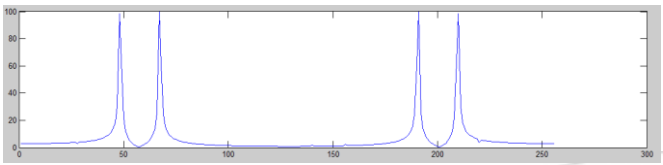
$$p_r = a_r b_r - a_i b_i = a_r(b_r + b_i) - (a_r + a_i)b_i$$

$$pi = a_r b_i - a_i b_i = a_r(b_r + b_i) + (a_i - a_r)b_r$$

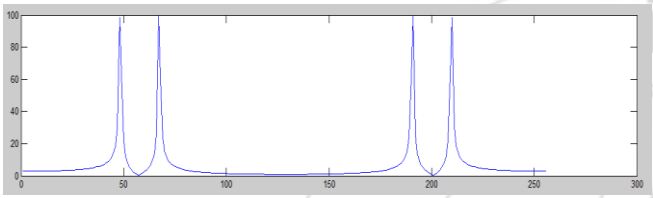
A three real multiplier solution is devised, which trades off one multiplier for three pre-combining adders and increased multiplier word length.

6. Simulation Result

For the simulation work, we have taken 256 point data samples from two sinusoidal signals. We have done simulation of parallel Radix-4 FFT architecture in Matlab as well as Verilog. Fig (3) describes the simulation results of both Matlab and Verilog. Clearly we can observe that the Matlab simulation result and Verilog simulation result is exactly same.



Radix-4 FFT in FPGA



Radix-4 FFT in Matlab

7. Conclusion

This proposed method describes a Radix-4 FFT algorithm using Parallel architecture. As seen from the simulation results both Matlab and Verilog achieves same result very accurately.Radix-4 FFT algorithm utilize less complex multipliers and additions than the Radix-2 FFT. And by using parallel architecture, the resources utilized are less than the conventional FFT architectures. This algorithm is very useful where there is a stringent requirement for the more number of Radix-4 FFT algorithm cores to be implemented in real time such as Radar Signal Analysis in defence applications.

References

[1] John G. Proakis, Dimitris G. Manolakis, “Digital Signal Processing: Principles, Algorithms, and Applications”, Prentice Hall, 1998.
[2] J. W. Cooley and J. W. Tukey, “An algorithm for the machine calculation of complex Fourier series.” Math. Coma. vol. 19. pp. 297-301, April 1965.
[3] G. D. Bergland, “The fast Fourier transform recursive equations for arbitrary length records,” Math. Comp., vol, 21, pp. 236238, April 1967.
[4] W. Li and L. Wanhammar, “A Pipeline FFT Processor,” accepted for publication at IEEE Workshop on Signal Processing Systems (SiPS), 1999.
[5] Y. Jiang, T. Zhou, Y. Tang and Y. Wang, “Twiddle factor-based FFT algorithm with reduced memory access,” in Proc. IEEE IPDPS’2002, April 2002, pp. 70-77.
[6] A. V. Oppenheim and R. W. Shafer, Discrete-Time Signal Processing, 2ndUpper Saddle River, NJ: Prentice Hall, 1998.

Author Profile

K.Pooja Swamy, is currently pursuing her M.Tech. In VLSI Design and Embedded Systems in ECE Department of LAEC,Bidar,Karnataka. She has completed B.E. in ECE in GNDEC, Bidar, Karnataka. She is presently working on an efficient architecture using Radix 4 FFT in FPGA. Presently she is a Project Trainee at DLRL,Hyderabad.

R. Pavan Kumar Graduated in Electronics and Communication Engineering (ECE) from VNR Vignana Jyothi Institute of Engineering and Technology, JNTU, Hyderabad, in 2006. He received his Post Graduation degree, M.Tech in Micro Electronics and VLSI design from Indian Institute of Technology Madras (IIT Madras), India in 2014. He joined Defence Electronics Research Laboratory (DLRL) in 2007.