International Journal of Research in Engineering, Science and Management
Volume-1, Issue-10, October-2018
www.ijresm.com | ISSN (Online): 2581-5782


# Design of Pipelined IFFT Processor

Farjin J Tamboli[1], S. A. Shirsat[2]
[1]*Student, Department Electronics and Telecommunication, SCOE, Pune, India*
[2]*Professor, Department Electronics and Telecommunication, SCOE, Pune, India*

***Abstract*—OFDM is multicarrier technique, which is efficient due to its orthogonality. It uses FFT/IFFT processors for modulation and demodulation purpose. Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT) act as primary blocks in most of the digital signal processing applications. These two operations have undergone numerous advancements in terms of software implementation. However, there is a lack of hardware implementation of the same, due to the involvement of floating point complex numbers. Existing Fast Fourier Transform /Inverse Fast Fourier Transform implementations mostly deal with integer data only and are incompatible with floating point data. The need of the hour is a design that can operate on floating point numbers and also achieves maximum efficiency, minimum hardware utilization and high data precision. In this paper pipelined Inverse Fast Fourier Transform is designed using Hardware Description Language. The proposed architecture is simulated in Xlinx ISIM 14.7 software and results are compared and verified with MATLAB outputs.**

***Index Terms*—IFFT Processor; FFT Processor; Floating point**

## I. Introduction

Due to evolving electronics and telecommunication applications, dedicated FFT or IFFT architectures are necessary for baseband processing. Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT) act as primary blocks in most of the digital signal processing applications. These two operations have undergone numerous advancements in terms of software implementation. However, there is a dearth of hardware implementation of the same, due to the involvement of floating point complex numbers. Existing Fast Fourier Transform/Inverse Fast Fourier Transform implementations mostly deal with integer data only and are incompatible with floating point data. The need of the hour is a design that can operate on floating point numbers and also achieves maximum efficiency, minimum hardware utilization and high data precision [1].

The FFT is an adequate algorithm for computing DFT and require less number of computation than that required for direct evaluation of DFT. The DFT algorithm has O ($n^2$) complexity and therefore is not useful for direct hardware realizations. Instead of DFT, Fast Fourier Transform algorithm is used. The computational complexity of FFT is given by complex multiplies N/2*Log N and complex additions N*Log N. The Discrete Fourier Transform (DFT) X (k) of N-point is given by,

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

Where, k=0 to N-1;

n=0 to N-1;

$$W_N^{kn} = e^{-j2\pi kn/N} \quad (2)$$

Where X(k) and x(n) are frequency domain sequence and time domain sequence instead of direct implementation of above equation, the FFT algorithm factorizes a large -point DFT recursively into many small -point DFT in order to reduce overall operations.

IFFT is a speedy algorithm which perform Inverse Fourier Transform, which is processed under DFT.

$$x(n)=\frac{1}{N} * \sum_{k=0}^{N-1} X(k) * e^{j2\pi k\frac{n}{N}} \quad (3)$$

*X(k)*= Frequency Domain Samples
*x(n)*= Time Domain Samples
N= FFT size
k= 0,1,2….N-1.

In this paper, 8-bit IFFT Processor is designed for complex imputs.

## II. Implementation of IFFT Processor

This section gives implementation of IFFT processor using different algorithms such as radix 2, radix 4, etc. and IFFT architectures such as pipelined, parallel, etc.

### A. Radix 2 Algorithm

The established equations for radix-2 FFT are

$$X(2k) = \sum_{n=0}^{\frac{N}{2}-1} \left( \left[ x(n) + x\left(n+\frac{N}{2}\right) \right] W_{\frac{N}{2}}^{kn} \right) \quad (3)$$

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

The Fig. 1, shows the implementation of 8-point radix-2 DIF algorithm. Here inputs are in proper sequence and outputs are in bit reversed order.

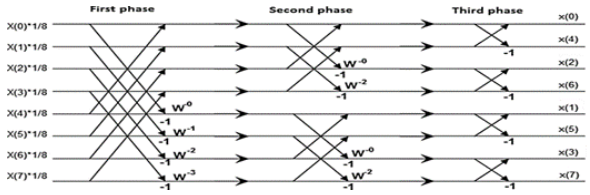


Fig. 1. 8 point radix 2 DIF algorithm [8]

Therotical calculations of 8 point IFFT:

X(k) = {0.5 , 2+j , 3+2j , j , 3 , -j , 3-2j , 2-j }

Let g(n) represent output of first stage and h(n) represent

output of second stage. The values of twiddle factor for IFFT are as follows:

$W_8^0 = 1$
$W_8^{-1} = 0.707 + j0.707$
$W_8^{-2} = j$
$W_8^{-3} = -0.707 + j0.707$

*Output of First Stage:*

g(0)=1/8(X(0)+X(4)) = 1/8(0.5+3) = 0.44
g(1)=1/8(X(1)+X(5)) = 1/8(2+j -j) = 0.25
g(2)=1/8(X(2)+X(6)) = 1/8(3+2j+3-2j) = 0.75
g(3)=1/8(X(3)+X(7)) = 1/8(j+2-j) = 0.25
g(4)=[1/8(X(0)-X(4))] $W_8^0$ = 1/8(0.5-3).1= -0.31
g(5)=1/8(X(1)-X(5)) $W_8^{-1}$ = 1/8(2+j+j)(0.707+j0.707) = 0.35j
g(6)=1/8(X(2)-X(6)) $W_8^{-2}$ = 1/8(3+2j-3+2j)j = -0.5
g(7)=1/8(X(3)-X(7)) $W_8^{-3}$ = 1/8(j-2+j)(-0.707+0.707j) = -0.35j

*Output of Second Stage*

h(0) = g(0)+g(2) = 1.19
h(1) = g(1)+g(3) = 0.5
h(2) = [g(0)-g(2)] $W_8^0$ = -0.31
h(3) = [g(1)-g(3)] $W_8^2$ = 0
h(4) = g(4)+g(6) = -0.81
h(5) = g(5)+g(7) = 0
h(6) = [g(4)-g(6)] $W_8^0$ = 0.19
h(7) = [g(5)-g(7)] $W_8^2$ = -0.7

*Final Output*

x(0) = h(0) + h(1) = 1.69
x(1) = h(4) + h(5) = -0.81
x(2) = h(2) + h(3) = -0.31
x(3) = [h(6) + h(7)] $W_8^0$ = -0.51
x(4) = [ h(0) - h(1) ] $W_8^0$= 0.69
x(5) = [ h(4) - h(5) ] $W_8^0$= -0.81
x(6) = [ h(2) - h(3) ] $W_8^0$= -0.31
x(7) = [ h(6) - h(7) ] $W_8^0$= 0.89
now sequence x(n) is
x(n) = {x(0), x(1),x(2),x(3),x(4),x(5),x(6),x(7)}
x(n) = {1.69, -0.81, -0.31, -0.51, 0.69, -0.81,-0.31,0.89}

### A. IFFT Pipeline Architecture

This architecture is also known as cascaded FFT architecture, and used in most of the designs. The basic structure of pipelined is as shown in Fig. 2, between each stage of radix-r pe's there is a commutator and last stage is unscrambling stage. The commutator records the output data from previous stage and feed to the next stage [8]. The unscramble rearranges data in natural sorted order in Fig. 2, denotes the stage number in the pipeline. The number in boxes gives the size of that fifo r in complex sampling. C2 is a switch and radix-4 butterfly element. Pipelined FFT architectures are fast and high throughput architectures with parallelism and pipelining. Even though the hardware complexity is high and less flexible compared to other architectures, they offer high throughput and energy efficient implementations.

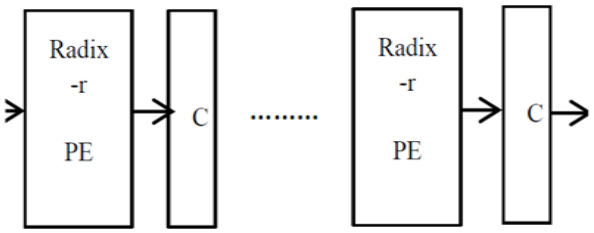

Fig. 2. General structure of a pipelined FFT architecture [8]

Performance of this architecture can be improved by Parallelism using separate arithmetic unit for each stage of FFT processer and through put can be increased by factor log2N using different units in pipelined. Pipelined FFT processers have features like high throughput, simplicity, fast, small area and energy efficient implementation. The most commonly used pipelined architectures such as Multipath Delay Commutator (MDC), Single Path Delay Commutator (SDC) and Single Path Delay Feedback (SDF).

## III. Results and Discussion

### A. Simulation Results for IFFT Processor

The architecture of IFFT Processor is simulated using Xilinx ISE 14.7 and compared with Matlab results. The Fig. 3, shows the simulated output of 8 point IFFT processor.

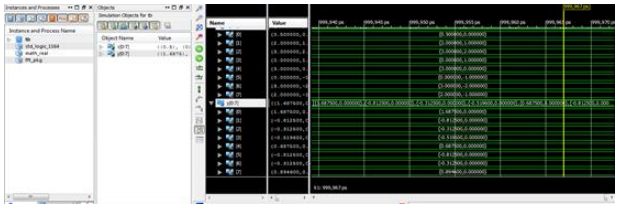
Fig. 3. Simulation results for IFFT processor

In this algorithm, X is an input array and Y is an output array. Each array consist of 8 elements hence it is called as 8- point IFFT algorithm. Each element is a complex number consist of real and imaginary part. Because of this, separate real and imaginary signals can be generated in RTL schematic. The processing time required for IFFT processor is 78ns.

#### 1. Matlab Output

VHDL results are validated by results obtained from 8 point IFFT using MATLAB R2010a. The Fig. 4, shows MATLAB output of 8-point IFFT.

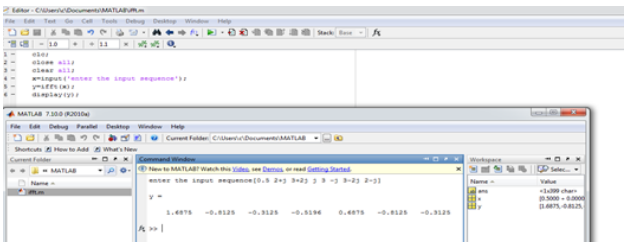
Fig. 4. MATLAB output of 8 point IFFT

### B. *Comparison of Results of IFFT Processor*

The Table-1, shows the comparison of the precision of 8 Point IFFT blocks as obtained in Matlab and on ISIM simulator.

TABLE I
COMPARISON OF RESULTS OF IFFT PROCESSOR OBTAINED IN MATLAB AND ISIM SIMULATOR

| Index | Inputs | IFFT Output on MATLAB | IFFT Output on ISIM simulator |
|---|---|---|---|
| 0 | 0+0j | 1.6875+0j | 1.6875+0j |
| 1 | 2+j | -0.8125+0j | -0.8125+0j |
| 2 | 3+2j | -0.3125+0j | -0.3125+0j |
| 3 | 0+j | -0.5196+0j | -0.5196+0j |
| 4 | 3+0j | 0.6875+0j | 0.6875+0j |
| 5 | 0-j | -0.8125+0j | -0.8125+0j |
| 6 | 3-2j | -0.3125+0j | -0.3125+0j |
| 7 | 2-j | 0.8946+0j | 0.8946+0j |

The Table-1, shows that results obtained on MATLAB and ISIM simulator are validated and verified.

## IV. CONCLUSION

The simulation of IFFT module is done in ISIM and the simulation results are compared with MATLAB results of pipelined architecture. This work can be extended for higher points of IFFT which can be used for communication applications.

## REFERENCES

[1] Uma B V, Harsha R Kamath, Mohith S, Sreekar V and 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)* Department of ECE, FET, MRIU, Faridabad, India, Oct 8-10, 2015

[2] kelly liew and lo hai hiung, "Performance comparison review of 32 bit multiplier designs", *International Conference On Intelligent and Advanced Systems, 2012.*

[3] Guoan bi, "A pipelined FFT processor for word sequential data", *IEEE Transactions on Acoustics, Speech, And Signal Processing*, Vol. 37. No. 12. December 1989.

[4] A. D. Booth, "A Signed Binary Multiplication Technique," Q.J. *Mechanics and Applied Mathematics*, Vol.4.1951.

[5] Anwar Bhasha Pattan and Dr. M. Madhavi Latha "Fast Fourier Transform Architectures: A Survey and State of the Art", *International Journal of Electronics & Communication Technology*, IJECT Vol. 5, Issue 4, Oct - Dec 2014.

[6] S.-N. Tang, J.-W. Tsai and T.-Y. Chang, "A 2.4-GS/s FFT processor for OFDM-based WPAN applications", *IEEE Trans. Circuits Syst. II, Exp. Briefs*, Vol. 57, No. 6, June 2010.

[7] Pierre Duhamel, "Implementation of Split-Radix FFT Algorithms for Complex, Real, and Real-Symmetric Data", *IEEE Transactions On Acoustics. Speech, And Signal Processing.* Vol. Assp-34. No. 2**,** April 1986 .

[8] B. Dinesh, V. Venkateshwaran, P. Kavinmalar and M. Kathirvelu, "Comparison of regular and tree based multiplier architectures with modified booth encoding for 4 bits on layout level using 45nm technology," *2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)*, Coimbatore, 2014, pp. 1-6.