Design and Implementation of 16 Point Radix 4 Parallel Fast Fourier Transform Processor

Tin Moe Aye and Chaw Thet Zan
University of Computer Studies, Yangon
aye.tinmoe@gmail.com, chawthetzan@gmail.com

Abstract

The FFT processor is a critical block in all multi-carrier systems used primarily in the mobile systems for image and digital signal processing applications. It is therefore interesting to develop an FFT processor as a widely usable VLSI building block.

In order to be flexible so that the processor can be used in a variety of applications without major redesign , the performance in terms of computational throughput , and transform length should be: easily modifiable. This system implements the 16 point radix 4 parallel Fast Fourier Transform processor with sfixed format (signed fixed point representation) and focuses on the complex multiplier design with two different approaches.

Keywords: FFT, VLSI, Parallel, sfixed

1. Introduction

A major application of Fourier transforms is the analysis of a series of observations, x_l, $l = 0,\dots, N-1$: which allows us to decompose a signal in the time domain and analyze the signal in the frequency domain. The sources of such observations are many: ocean tidal records over many years, communication signals over many microseconds, sonar signals over a few minutes, and so on. The assumption is that there are repeating patterns in the data that form part of the x. However, usually there will be other phenomena which may not repeat, or repeat in a way that is not discernibly cyclic. This is called “noise.” The DFT (The Discrete Fourier Transform) helps to identify and quantify the cyclic phenomena. If a pattern repeats itself m times in the N observations, it is said to have *Fourier frequency m*. [1]

The Fast Fourier Transform (FFT), an efficient algorithm to compute the Discrete Fourier Transform (DFT), is one of the most important operations in modern digital signal processing and communication systems. [2]The parallel FFT is a special type of FFT which can compute the FFT algorithms by adding more processing elements to the processor in each sequential pipeline stage to improve the performance. But the drawback of parallel design is that the area of FFT becomes enlarge. [3][12]

2. A Brief Review on DFT and FFT

Basically, the computational problem for the DFT is to compute the sequence $\{X(k)\}$ of N complex-valued numbers given another sequence of data $\{x(n)\}$ of length N, according to the formula

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

Where $W_N = e^{-j2\pi/N}$

W_N's are also called "twiddle factors" which are complex values around the unit circle in the complex plane. [2][9]The complex ’rotator’ W_N rotates the other direction and the result is divided with the number of points N. So the computation is basically the same.

We can exploit shared twiddle factor properties (i.e. sub-expression sharing) to reduce the number of multiplications in DFT. These classes of algorithms are called Fast Fourier Transforms. An FFT is simply an efficient implementation of the DFT.

Mathematically FFT = DFT

FFT exploits two properties in the twiddle factors:

- Symmetry Property: $W_N^{k+N/2} = -W_N^k$
- Periodicity Property: $W_N^{k+N} = W_N^k$

Actually, direct computation of Discrete Fourier Transform (DFT) requires on the order of N^2 operations where N is the transform size. The FFT algorithm, first explained by Cooley and Turkey, open a new area in digital signal processing by reducing the order of complexity of DFT from N^2 to $N\log_2 N$. [4]

3. Radix 4 FFT Algorithm

Radix-4 split x (n) into four time sequences instead of two in radix-2.The algorithm splits x (n) into four decimated sample streams

$f_1(n) = x(4m)$

$f_2(n) = x(4m+1)$

$f_3(n) = x(4m+2)$

$f_4(n) = x(4m+3)$,n=0, 1, .. N/4-1

Therefore, the equations for breaking the he *N*-point DFT formula into four smaller DFTs becomes:

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{kn} \\ &= \sum_{n=0}^{N/4-1} x(n) W_N^{kn} + \sum_{n=N/4}^{N/2-1} x(n) W_N^{kn} + \sum_{n=N/2}^{3N/4-1} x(n) W_N^{kn} + \sum_{n=3N/4}^{N-1} x(n) W_N^{kn} \\ &= \sum_{n=0}^{N/4-1} x(n) W_N^{kn} + W_N^{Nk/4} \sum_{n=0}^{N/4-1} x\left(n+\frac{N}{4}\right) W_N^{kn} + \\ &\quad W_N^{Nk/2} \sum_{n=0}^{N/4-1} x\left(n+\frac{N}{2}\right) W_N^{kn} + W_N^{3Nk/4} \sum_{n=0}^{N/4-1} x\left(n+\frac{3N}{4}\right) W_N^{kn} \end{aligned}$$

From the definition of the twiddle factors, we have:

$$W_N^{kN/4} = (-j)^k, \qquad W_N^{kN/2} = (-1)^k, \qquad W_N^{3kN/4} = (j)^k$$

Thus, we get :

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

To convert it into an *N*/4-point DFT we subdivide the DFT sequence into four *N*/4-point subsequences, *X*(4*k*), *X*(4*k*+1), *X*(4*k*+2), and *X*(4*k*+3), *k* = 0, 1, ..., *N*/4. Then, we obtain the following equations;[2][9]

$$\begin{aligned} X(4k) &= \sum_{n=0}^{N/4-1} \left[x(n) + x\left(n+\frac{N}{4}\right) + x\left(n+\frac{N}{2}\right) + x\left(n+\frac{3N}{4}\right) \right] W_N^0 W_{N/4}^{kn} \\ X(4k+1) &= \sum_{n=0}^{N/4-1} \left[x(n) - jx\left(n+\frac{N}{4}\right) - x\left(n+\frac{N}{2}\right) + jx\left(n+\frac{3N}{4}\right) \right] W_N^n W_{N/4}^{kn} \\ X(4k+2) &= \sum_{n=0}^{N/4-1} \left[x(n) - x\left(n+\frac{N}{4}\right) + x\left(n+\frac{N}{2}\right) - x\left(n+\frac{3N}{4}\right) \right] W_N^{2n} W_{N/4}^{kn} \\ X(4k+3) &= \sum_{n=0}^{N/4-1} \left[x(n) + jx\left(n+\frac{N}{4}\right) - x\left(n+\frac{N}{2}\right) - jx\left(n+\frac{3N}{4}\right) \right] W_N^{3n} W_{N/4}^{kn} \end{aligned}$$

The signal flow graph of the 16 point radix-4 fast fourier transform algorithm is shown in the Figure 1.[5]

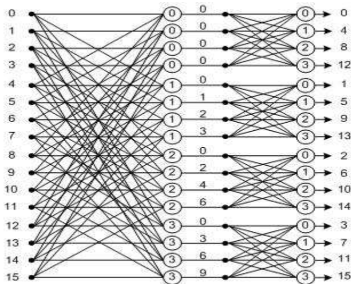


Figure 1. Signal flow diagram of 16 point radix-4 FFT

4. 16 Point Radix 4 Parallel FFT overview

In 16 Point Radix 4 Parallel FFT architecture, four butterfly units are used at the input sites that handle four inputs each.

After processing in each butterfly unit, some outputs are fed directly into the one of four output sites butterfly units and the others are fed directly to left 3 output site butterfly units via complex multiplier as shown in the Figure 2.

For all of the output comes out at the same time, we need to adjust and modify the complex multiplier design. The abstract view of the algorithm design is shown in the Figure 2.

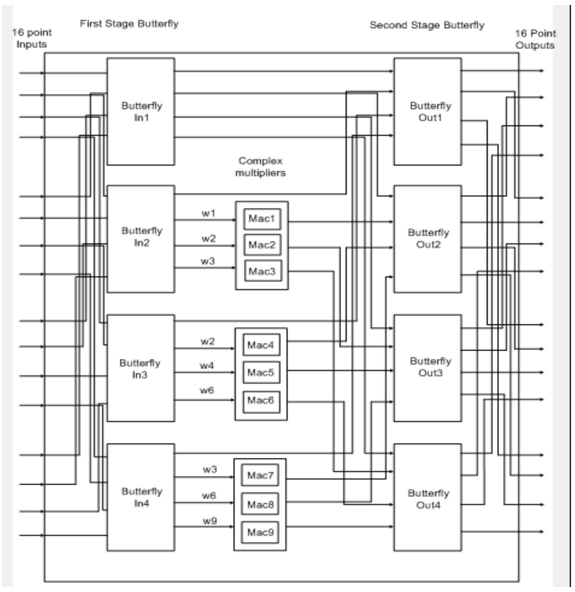


Figure 2. Abstract view of System Design

4.1. Butterfly Units

The butterfly takes four inputs and produces four outputs.

In radix-2 FFT, the DFT equation is expressed as the sum of two calculations; one calculation sum for the first half and one calculation sum for the second half.

In this design, the radix-4 FFT implements the DFT equations as four summations. Butterfly units compute the summations by using four equations, each of which computes every fourth sample.

The equations are as follows,

Let x(n) = xa+jya;

x(n+n/4) = xb+jyb;

x(n+n/2) = xc+jyc;

x(n+3n/4)= xd+jyd;

as the four inputs for the butterfly.

$$x(4r) = xa' + jya',$$
$$x(4r+1) = xb' + jyb',$$
$$x(4r+2) = xc' + jyc' and$$
$$x(4r+3) = xd' + jyd'$$

as the four outputs from the butterfly and

W_n=Wb =Cb+j(-Sb)
W_{2n}=Wc =Cc+j(-Sc)
W_{3n}=Wd =Cd+j(-Sd)

So that, the radix 4 equations become:

$$xa' = xa + xb + xc + xd$$
$$ya' = ya + yb + yc + yd$$
$$xb' = (xa + yb - xc - yd)Cb - (ya - xb - yc + xd)(-Sb)$$
$$yb' = (ya - xb - yc + xd)Cb + (xa + yb - xc - yd)(-Sb)$$
$$xc' = (xa - xb + xc - xd)Cc - (ya - yb + yc - yd)(-Sc)$$
$$yc' = (ya - yb + yc - yd)Cc + (xa - xb + xc - xd)(-Sc)$$
$$xd' = (xa - yb - xc + yd)Cd - (ya + xb - yc - xd)(-Sd)$$
$$yd' = (ya + xb - yc - xd)Cd + (xa - yb - xc + yb)(-Sd)$$

4.2. Complex Multiplier unit

In most of FFT design implementation scheme, the conventional complex multiplication is performed with four real multipliers, one adder and one subtractor. As the literature survey, the complexity of a multiplier is much more than that of adder and subtractor. Thus we implemented our system with two approaches by reducing the number of multipliers.

As the number of coefficients to be multiplied in 16-point FFT is 16 which are described in the follow table, Table 1. [5][6]

Table 1. The coefficients for 16 point Radix 4 FFT

Coefficient sequence m_1=0,1	Original quantized coefficient	Coefficient sequence m_1=2,3	Original quantized coefficient
W0	7fff, 0000	W0	7fff, 0000
W0	7fff, 0000	W2	5a82, a57d
W0	7fff, 0000	W4	0000, 8000
W0	7fff, 0000	W6	a57d, a57d
W0	7fff, 0000	W0	7fff, 0000
W1	7641, cf04	W3	30fb, 89be
W2	5a82, a57d	W6	a57d, a57d
W3	30fb, 89be	W9	89be, 30fb

A close observation reveals that the seven coefficients (7fff, 0000) are the trivial coefficients which are the quantized representation for (1, 0) in 16-bit two's complement format. The complex multiplication is not necessary for these coefficients. Data can directly pass through the multiplier unit without any multiplication, when data is multiplied with (7fff, 0000). [6]

And also from the theoretical point of view, the conventional butterfly unit involves three complex multiplications, since for the first twiddle factor to be multiplied W0 and is always 1. For the 16 point FFT, there are four butterfly units to complete the entire FFT and thus requiring 12 complex multiplications. [2]

By detaching the complex multiplier unit apart from the butterfly unit, we can reduce the complex multiplier units from 12 to 9 as the complex multiplication is required for only remaining nine nontrivial coefficients. Therefore, the system actually required 36 real multipliers and 18 adders/subtractors and this approach is considered as the **first approach** as shown in the Figure 3.

Let x and y be the complex numbers and the multiplications of the numbers can be implemented by the following equations:

Z-real=x-real*y-real – x-imag*y-imag
Z-imag= x-real*y-imag+ y-real*x-imag

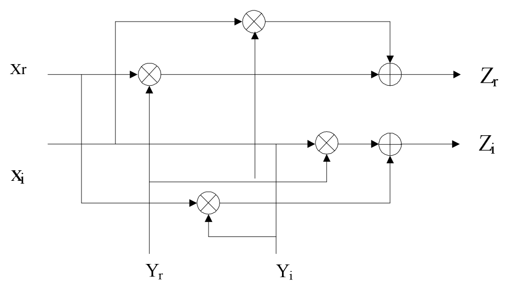


Figure 3.Implementation of First Approach

Then, we modify the complex multiplication equations in order to reduce the number of real multipliers using in each complex multiplier as shown in the Figure 4. The modified equations can be written as:

Z-real=x-real*(y-real+y-imag)–(x-real+x-mag)*y-imag
Z-imag= x-real(y-real+y-imag)+(x-real-x-imag)*x-real

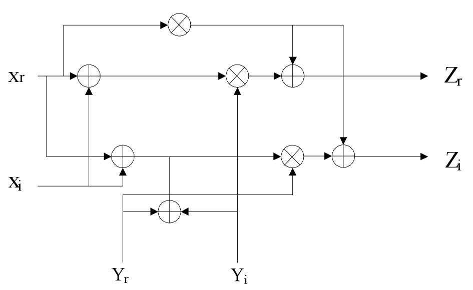


Figure 4.Implementation of Second Approach

In this **second approach**, the number of the real multipliers reduces to three, and the number of the real adders increases to five. But the complexity of adders is very lower than that of multipliers.

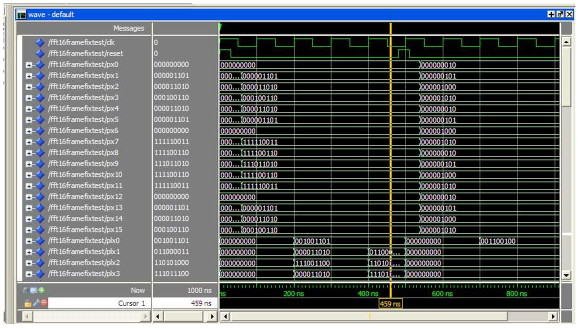
5. System Implementation and Synthesis Report

The 16 Point Radix 4 Parallel FFT processor is simulated on Modelsim SE 6.4 with VHDL.

The system works at the positive edge triggered clock together with the active low reset signal. And the timing simulation of the system design is depicted in the Figure 5 (a), 5 (b), 5 (c) and 5 (d).

5.1. Simulation Result

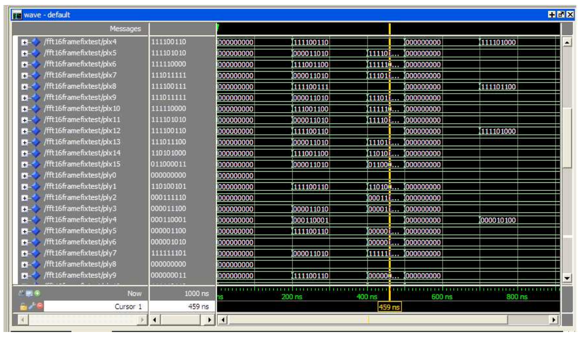
In the timing simulation diagram 5(a), we can see that clk and reset are the global control signals to the FFT.



```
wave - default
Messages
/fft16framefixtest/clk      0
/fft16framefixtest/reset    0
/fft16framefixtest/px0      000000000
/fft16framefixtest/px1      000001101
/fft16framefixtest/px2      000011010
/fft16framefixtest/px3      000100110
/fft16framefixtest/px4      000011010
/fft16framefixtest/px5      000001101
/fft16framefixtest/px6      000000000
/fft16framefixtest/px7      111110011
/fft16framefixtest/px8      111100110
/fft16framefixtest/px9      111011010
/fft16framefixtest/px10     111100110
/fft16framefixtest/px11     111110011
/fft16framefixtest/px12     000000000
/fft16framefixtest/px13     000001101
/fft16framefixtest/px14     000011010
/fft16framefixtest/px15     000100110
/fft16framefixtest/plx0     001001101
/fft16framefixtest/plx1     011000011
/fft16framefixtest/plx2     110101000
/fft16framefixtest/plx3     111011100
Now       1000 ns
Cursor 1  459 ns
```

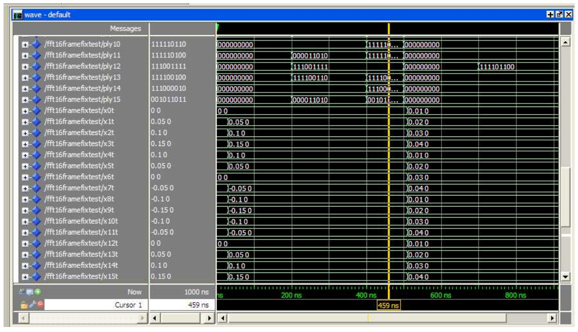
Figure 5(a). Modelsim simulation result

The Figure 5(b) and 5(c) show the output signals and their resultant values which can be represented in singed fixed point format.



```
wave - default
Messages
/fft16framefixtest/plx4     111100110
/fft16framefixtest/plx5     111101010
/fft16framefixtest/plx6     111110000
/fft16framefixtest/plx7     111011111
/fft16framefixtest/plx8     111100111
/fft16framefixtest/plx9     111011111
/fft16framefixtest/plx10    111110000
/fft16framefixtest/plx11    111101010
/fft16framefixtest/plx12    111100110
/fft16framefixtest/plx13    111011100
/fft16framefixtest/plx14    110101000
/fft16framefixtest/plx15    011000011
/fft16framefixtest/ply0     000000000
/fft16framefixtest/ply1     110100101
/fft16framefixtest/ply2     000111110
/fft16framefixtest/ply3     000011100
/fft16framefixtest/ply4     000110001
/fft16framefixtest/ply5     000001100
/fft16framefixtest/ply6     000001010
/fft16framefixtest/ply7     111111101
/fft16framefixtest/ply8     000000000
/fft16framefixtest/ply9     000000011
Now       1000 ns
Cursor 1  459 ns
```

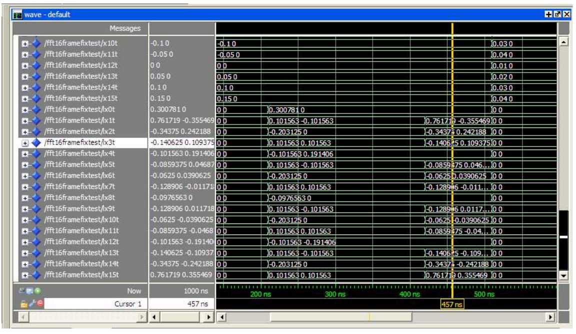
Figure 5(b). Modelsim simulation result



```
wave - default
Messages
/fft16framefixtest/ply10    111110110
/fft16framefixtest/ply11    111110100
/fft16framefixtest/ply12    111001111
/fft16framefixtest/ply13    111100100
/fft16framefixtest/ply14    111000010
/fft16framefixtest/ply15    001011011
/fft16framefixtest/x0t      0 0
/fft16framefixtest/x1t      0.05 0
/fft16framefixtest/x2t      0.1 0
/fft16framefixtest/x3t      0.15 0
/fft16framefixtest/x4t      0.1 0
/fft16framefixtest/x5t      0.05 0
/fft16framefixtest/x6t      0 0
/fft16framefixtest/x7t      -0.05 0
/fft16framefixtest/x8t      -0.1 0
/fft16framefixtest/x9t      -0.15 0
/fft16framefixtest/x10t     -0.1 0
/fft16framefixtest/x11t     -0.05 0
/fft16framefixtest/x12t     0 0
/fft16framefixtest/x13t     0.05 0
/fft16framefixtest/x14t     0.1 0
/fft16framefixtest/x15t     0.15 0
Now       1000 ns
Cursor 1  459 ns
```

Figure 5(c). Modelsim simulation result

The diagram 5(d) shows the output result in a real data type format.



```
wave - default
Messages
/fft16framefixtest/x10t     -0.1 0
/fft16framefixtest/x11t     -0.05 0
/fft16framefixtest/x12t     0 0
/fft16framefixtest/x13t     0.05 0
/fft16framefixtest/x14t     0.1 0
/fft16framefixtest/x15t     0.15 0
/fft16framefixtest/lx0t     0.300781 0
/fft16framefixtest/lx1t     0.761719 -0.355469
/fft16framefixtest/lx2t     -0.34375 0.242188
/fft16framefixtest/lx3t     -0.140625 0.109375
/fft16framefixtest/lx4t     -0.101563 0.191406
/fft16framefixtest/lx5t     -0.0859375 0.04687
/fft16framefixtest/lx6t     -0.0625 0.0390625
/fft16framefixtest/lx7t     -0.128906 -0.01171
/fft16framefixtest/lx8t     -0.0976563 0
/fft16framefixtest/lx9t     -0.128906 0.011718
/fft16framefixtest/lx10t    -0.0625 -0.0390625
/fft16framefixtest/lx11t    -0.0859375 -0.0468
/fft16framefixtest/lx12t    -0.101563 -0.19140
/fft16framefixtest/lx13t    -0.140625 -0.10937
/fft16framefixtest/lx14t    -0.34375 -0.242188
/fft16framefixtest/lx15t    0.761719 0.355469
Now       1000 ns
Cursor 1  457 ns
```

Figure 5(d). Modelsim simulation result

As the verification point of view, the system is synthesized with Virtex xc4vlx25-12ff668 prototype using Xilinx 12.2 se logic analyzer.

The timing summary for all approaches is described in Table 2.

Table.2 Timing Summary

Algorithms	Operating Frequency	Slices
Conventional FFT	91.416MHz	3471
First Approach	102.501MHz	2603
Second Approach	91.366MHz	2850

The HDL synthesis report of conventional and proposed two approaches are mention in the Table 3 and 4 receptively.

Table 3. HDL Synthesis Report of Conventional FFT Design

# Multipliers	48
17x17-bit multiplier	48
# Adders/Subtractors	240
18-bit adder	56
18-bit subtractor	32
19-bit adder	32
19-bit subtractor	32
20-bit adder	32
20-bit substractor	32
35-bit adder	12
35-bit substractor	12
#Registers	136
17-bit register	64
34-bit register	48
35-bit register	24
#Xors	24
1-bit xor2	24

Table 4. HDL Synthesis Report of Proposed Design with Two Approaches

	First Approach	Second Approach
# Multipliers	36	27
16x16-bit multiplier	36	27
# Adders/Subtractors	228	255
17-bit adder	50	41
17-bit subtractor	32	32
18-bit adder	32	32
18-bit subtractor	32	32
19-bit adder	32	32
19-bit substractor	32	32
33-bit adder	9	9
33-bit substractor	9	9
#Registers	118	109
16-bit register	64	64
32-bit register	36	27
33-bit register	18	18
#Xors	18	18
1-bit xor2	18	18

6. Conclusion

In this paper, the parallel FFT processor architecture with separate complex multiplier is presented. The parallel architecture is used in order to get high performance. Furthermore, the advantage of separating the butterfly units and complex multiplier units is that each component can be easily modifiable as the complex multiplier unit is embedded in the butterfly unit in most of the conventional FFTs.

In the first approach, two adders and four real multipliers are used for complex multiplier which has the greater speed of 102.501 MHz. The second approach which includes three real multipliers and five real adders, is used to reduce the number of the real multipliers as the real multiplier is larger and more complex than the adder .But the maximum frequency of latter approach is reduced to 91.366 MHz which is nearly the same speed as the conventional approach's maximum operating frequency (91.416 MHz), we can reduce the resource usage at all.

7. References

[1] Eleanor Chu and Alan George. *Inside the FFT Black Box Serial and Parallel Fast Fourier Transform Algorithms*, CRC press.

[2] John G.P. and D.G. Manolakis , 1988. *Introduction to Digital Signal Processing* .Mac Milan.

[3] Joseph McRae Palmer, *The Hybrid Architecture Parallel Fast Fourier Transforms (HAPFFT)* ,Master Thesis, Department of Electrical and Computer Engineering, Brigham Young University, August 2005

[4] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comp.*, vol. 19, pp. 297-301, 1965.

[5] Mahmud Benhamid and Masuri Bin Othman, "Hardware Implementation of Genetic Algorithm Based Canonical Signed Digit Multiplierless Fast Fourier Transform Processor for Multiband Orthogonal Frequency Division Multiplexing Ultra Wideband Applications ", *Journal of Mathematics and Statistics 5(4)*: 241-250, 2009.

[6] M. Kannan and S.K. Srivatsa , "Low Power Hardware Implementation of High Speed FFT Core ", *Journal of Computer Science 3 (6)*: 376-382, 2007.

[7] Nima Aghaee and Mohammad Eshghi. "Design of a pipelined R4SDF processor", *17th European Signal Processing conference (EUSIPCO 2009*), Glasgow, Scotland, August 24-28, 2009.

[8] S. He and M. Torkelson, "A new approach to pipeline FFT processor," *In Proc. of the 10th International Parallel Processing Symposium (IPPS)*, Honolulu, Hawaii, USA, 1996.

[9] Steven W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*, Second Edition.

[10] Weidong Li, *Studies On Implementation of Low Power FFT Processors*, Linköping Studies in Science and Technology , Thesis No. 1030

[11] http://fftguru.com/fftguru.com.tutorial.pdf

[12] http://mason.gmu.edu