OPEN ACCESS

International
Journal
Of Modern Engineering Research (IJMER)

Implementation of High Throughput Radix-16 FFT Processor

K Swetha sree[1], Mr. T. Lakshmi Narayana[2]
[1, 2] *Department of ECE, Andhra Loyola institute of engineering and technology, India*

Abstract: *The extension of radix-4 algorithm to radix-16 to achieve the high throughput of 2.59 giga-samples/s for WPAN's.We are also reformulating radix-16 algorithm to achieve low-complexity and low area cost and high performance. Radix -16 FFT is obtained by cascaded the radix -4 butterfly units. It facilitates low-complexity realization of radix-16 butterfly operation and high operation speed due to its optimized pipelined structure. Besides, a new three-stage multiplier for twiddle factor multiplication is also proposed, which has lower area and power consumption than conventional complex multipliers. Equipped with those new performance-boosting techniques, overall the proposed radix-16 FFT processor is area-efficient with high data processing rate and hardware utilization efficiency. The control circuit of the proposed simplified radix-24 FFT SDF architecture is simpler than that of the direct radix-16 FFT SDF structure. The multiplier cost of the proposed FFT architecture is less than that of the previous FFT structures in 256-point FFT applications. The throughput of the proposed FFT processor is one sample per clock. Although the radix-16 FFT algorithm has Less computational complexity, the control circuit of the direct radix-16 SDF architecture for implementing radix-16 FFT is very complex. Thus, the efficient simplified radix-24 SDF structure, which is described in the next section, will be applied to radix-16 FFT algorithm.*

Keywords: *Fast Fourier transform (FFT), non-conflict memory addressing scheme, OFDM, radix-16 FFT, WPANs.*

I. Introduction

SINCE the recent decade, the increasing demand for real time and high-rate multimedia services has been pushing the birth of high-rate wireless communication systems. Ultra wideband (UWB) communication system, for example, can deliver data rates up to 480 Mb/s at a short distance range. However, it is not enough to support high data rate applications of more than 1Gbps such as high-definition (HD) streaming content downloads HD video on demand, home theater, and etc. To meet the application demands, IEEE 802.15.3c-2009 standard for high-rate Wireless Personal Area Networks (WPANs) was ratified recently. In the standard, there are three PHY modes, i.e., Single Carrier mode (SC PHY), High Speed Interface mode (HSI PHY), and Audio/Visual mode (AV PHY). Except SC PHY, both HSI PHY and AV PHY are based on OFDM modulations. As is well known, Fast Fourier transform (FFT) operation is one of the key operations for OFDM-based communication systems. Besides, for SC PHY mode, FFT operations are widely employed for effective channel equalization. Low-cost 256-point FFT SDF design for OFDM systems. The rest of the paper is organized as follows. We begin with the derivation of the radix-16 FFT algorithm. By using the structure of radix-8 FFT algorithm, the radix-16 FFT algorithm is described to guide the depiction of hardware pipelined architectures. Next, the proposed simplified radix-24 SDF pipelined FFT architecture for realization of radix-16 FFT is also shown in this section. A low multiplier cost 256-point FFT architecture with simplified radix-24 SDF structure. The detailed function realizations of the simplified radix-24 SDF structure are also discussed, the proposed simplified radix-24 based FFT processor can reduce the hardware complexity and control circuit complexity.

Another benefit of higher-radix FFT operations over the lower-radix ones is that higher memory access bandwidth can be more conveniently provided in hardware implementations, as the bandwidth is proportional to. An improved radix-16 algorithm is proposed in which can reduce the numbers of twiddle factor operations and lookup-table accesses. A radix-16 algorithm suitable for multiply-and-add instruction is proposed in . However, those algorithms are mainly devised for the execution of general-purpose processors. Owing to the mentioned advantages of a high-radix memory-based FFT design, this work will design a high-performance radix-16 FFT processor which satisfies the mentioned throughput requirement of 802.15.3c applications. However, since generally a radix-16 butterfly unit is more complicated and less flexible than

lower-radix ones, this work reformulates conventional radix-16 FFT algorithm so as to facilitate efficient and optimized pipelined realization of a radix-16 PE with high computing power and speed. Further, several performance-enhancement techniques are applied to the whole FFT processor design, including an efficient multiplier structure for twiddle factor multiplication, schemes of conflict-free memory access and normal-order FFT output generations. The rest of this article is organized as follows. In Section II, design concerns for 802.15.3c FFT processor are examined. In a reformulated radix-16 algorithm and its butterfly structure are analyzed. In Section IV, a high-throughput and high-speed FFT processor architecture is introduced. Also, a three-stage multiplier for twiddle factor multiplication is presented. In a new conflict-free memory addressing scheme is presented. Block-floating point (BFP) operations are designed and implemented to achieve high signal-to-quantization-noise ratio (SQNR).

II. Radix-16 FFT Algorithm And The Simplified Pipelined SDF Architecture

Radix-16 FFT Algorithm

The butterfly structure for computation of the radix-8 FFT algorithm is proposed in [7]. Fig. 1 shows the butterfly structure of the radix-8 FFT algorithm, where replace the k index with the index k=2k' and the index k=2k'+1 in radix-8 FFT structure. Thus, the even parts and the odd parts of the radix-16 FFT algorithm can be described in Eq.(1). The radix-16 FFT algorithm listed in Eq.(1) can be summarized to the butterfly structure that is shown in Fig. 2. First, the radix-16 FFT algorithm can be realized with the direct radix-16 SDF architecture, which is shown in Fig. 3. The direct radix-16 SDF structure is constructed by an N/16 memory and the radix-16 butterfly processing element (PE). Fig. 4 shows the detailed function block of the radix-16 PE. Although the radix-16 FFT algorithm has less computational complexity, the control circuit of the direct radix-16 SDF architecture for implementing radix-16 FFT is very complex. Thus, the efficient simplified radix-24 SDF structure, which is described in the next section, will be applied to radix-16 FFT algorithm.

$$X[16k'+a+2b+4c+8d]=\sum_{n=0}^{N/16-1}\{\{\{[x(n)+(-1)^a\cdot x(n+N/2)]+W^{N(a+2b)/4}\cdot[x(n+N/4)+(-1)^{a+2b}\cdot x(n+3N/4)]\}$$

$$+W^{N(a+2b+4c)/8}\cdot\{[x(n+N/8)+(-1)^{a+2b+4c}\cdot x(n+5N/8)]+W^{N(a+2b)/4}\cdot[x(n+3N/8)+(-1)^{a+2b+4c}\cdot x(n+7N/8)]\}\}$$

$$+\{\{[x(n+N/16)+(-1)^{a+2b+4c+8d}\cdot x(n+9N/16)]+W^{N(a+2b)/4}\cdot[x(n+5N/16)+(-1)^{a+2b+4c+8d}\cdot x(n+7N/8)]\}$$

$$+\{[x(n+3N/16)+(-1)^{a+2b+4c+8d}\cdot x(n+11N/16)]+W^{N(a+2b)/4}\cdot[x(n+7N/16)+(-1)^{a+2b+4c+8d}\cdot x(n+15N/16)]\}$$

$$\cdot W^{N(a+2b+4c)/8}\}\cdot W^{N(a+2b+4c+8d)/16}\}\cdot W^{(a+2b+4c+8d)\cdot n}\cdot W_{N/16}^{nk'}, \qquad (1)$$

where a, b, c, and d is 0 or 1, and $k'=0,1,\ldots,N/16-1$

$$X(16k_2+k_1)$$

$$=\sum_{n_2=0}^{31}\sum_{n_1=0}^{15}x(32n_1+n_2)W_{512}^{(16k_2+k_1)(32n_1+n_2)}$$

$$=\sum_{n_2=0}^{31}\left\{\underbrace{\sum_{n_1=0}^{15}x(32n_1+n_2)W_{16}^{n_1k_1}}_{16\text{-point DFT of the 1st stage}}\underbrace{W_{512}^{n_2k_1}}_{\substack{\text{twiddle factor}\\ \text{of stage 1}}}\right\}W_{32}^{n_2k_2}$$

$$=\sum_{n_2=0}^{31}\left\{BO_{16}^{(1,n_2)}(k_1)\right\}W_{32}^{n_2k_2}$$

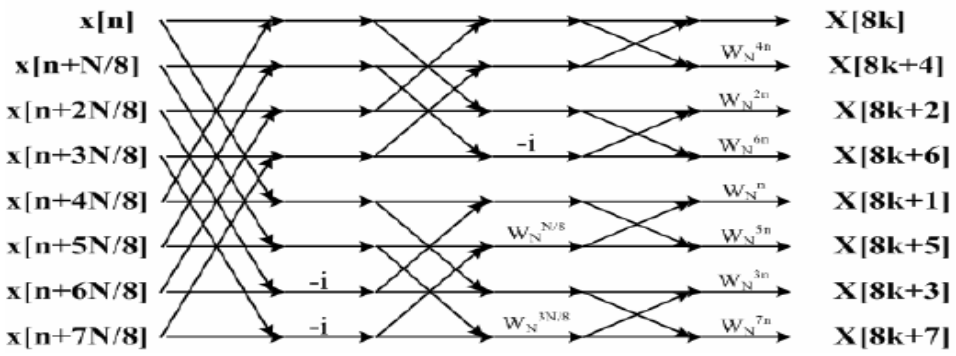


Fig. 1 Butterfly structure of the radix-8 FFT algorithm [7]

2.1 Simplified Radix-2^4 SDF Pipelined Architecture

In order to simplify the radix-16 control unit, we derive the radix-24 algorithm so that the hardware implementation is performed by cascading four simple radix-2 processing elements. Therefore, the radix-24 architecture is simplified and has the property of high spatial regularity. Subsequently, we use the common factor algorithm (CFA) decomposition method to develop the proposed FFT algorithm, and then the frequency domain indices k and the time domain indices n can be factorized as follows.

$$n =< \frac{N}{2}n_1 + \frac{N}{4}n_2 + \frac{N}{8}n_3 + \frac{N}{16}n_4 + n_5 >_N . \quad (2)$$

where N=256, $n_i = 0,1$ for i=1, 2 ,3 and 4, and $n_5 = 0,1,2,\ldots,N/16-1$. And

$$k =< k_1 + 2k_2 + 4k_3 + 8k_4 + 16k_5 >_N . \quad (3)$$

where $k_i = 0,1$ for i=1, 2, 3 and 4, and $k_5 = 0,1,2,\ldots,N/16-1$.

In Eq.(2) and Eq.(3), the indices n and k are mapped from one dimension to five dimensions linearly. By using Eq.(2) and Eq.(3), the DFT formula is rewritten as the multi-dimension form, which is shown as Eq. (4).

$$X[k_1 + 2k_2 + 4k_3 + 8k_4 + 16k_5] = \sum_{n_5=0}^{N/16-1}\sum_{n_4=0}^{1}\sum_{n_3=0}^{1}\sum_{n_2=0}^{1}\sum_{n_1=0}^{1} x\left(\frac{N}{2}n_1 + \frac{N}{4}n_2 + \frac{N}{8}n_3 + \frac{N}{16}n_4 + n_5\right) \cdot W_N^{\left(\frac{N}{2}n_1 + \frac{N}{4}n_2 + \frac{N}{8}n_3 + \frac{N}{16}n_4 + n_5\right)(k_1 + 2k_2 + 4k_3 + 8k_4 + 16k_5)} . \quad (4)$$

Subsequently, we use the period property of the twiddle factor in Eq.(4) and decompose the twiddle factor as follows.

$$W_N^{\left(\frac{N}{2}n_1+\frac{N}{4}n_2+\frac{N}{8}n_3+\frac{N}{16}n_4+n_5\right)(k_1+2k_2+4k_3+8k_4+16k_5)} = W_N^{\frac{N}{2}n_1k_1} \cdot W_N^{\frac{N}{4}n_2(k_1+2k_2)} \cdot W_N^{\frac{N}{8}n_3(k_1+2k_2+4k_3)} \cdot W_N^{\frac{N}{16}n_4(k_1+2k_2+4k_3+8k_4)} \cdot W_N^{n_5(k_1+2k_2+4k_3+8k_4)} \cdot W_{N/16}^{n_5k_5} . \quad (5)$$

We expand the dimensional variables n_1, n_2, n_3, and n_4

In Eq.(4) and Eq.(5), and then the butterfly expressions, which are ranged from the first stage to the fourth stage, are denoted as N

$$A_{N/2}(\frac{N}{4}n_2+\frac{N}{8}n_3+\frac{N}{16}n_4+n_5,k_1)=x(\frac{N}{4}n_2+\frac{N}{8}n_3+\frac{N}{16}n_4+n_5)+(-1)^{k_1}x(\frac{N}{4}n_2+\frac{N}{8}n_3+\frac{N}{16}n_4+n_5), \quad (6)$$

$$B_{N/4}(\frac{N}{8}n_3+\frac{N}{16}n_4+n_5,k_1,k_2)=A_{N/2}(\frac{N}{8}n_3+\frac{N}{16}n_4+n_5,k_1)+W_N^{\frac{N}{4}(k_1+2k_2)}A_{N/2}(\frac{N}{8}n_3+\frac{N}{16}n_4+n_5+\frac{N}{4},k_1), \quad (7)$$

$$C_{N/8}(\frac{N}{16}n_4+n_5,k_1,k_2,k_3)=B_{N/4}(\frac{N}{16}n_4+n_5,k_1,k_2)+W_N^{\frac{N}{8}(k_1+2k_2+4k_3)}B_{N/4}(\frac{N}{16}n_4+n_5+\frac{N}{8},k_1,k_2), \quad (8)$$

and

$$D_{N/16}(n_5,k_1,k_2,k_3,k_4)=C_{N/8}(n_5,k_1,k_2,k_3)+W_N^{\frac{N}{16}(k_1+2k_2+4k_3+8k_4)}C_{N/8}(n_5+\frac{N}{16},k_1,k_2,k_3). \quad (9)$$

/ 2 A , N / 4 B , N /8 C , and N /16 D , respectively

2.2 Pallar processing of radix 16 algorithm:

Stages of the proposed SR24SDF structure correspond to the relative 4-stage radix-16 FFT butterfly flow. The architecture of the proposed SR2^44SDF is. For the realization of N-point FFT, the SR24SDF architecture needs log16N-1 multipliers, 4log4N adders, and N-1 registers. The control circuit of SR24SDF is simpler in comparison with the direct radix-16 SDF architecture. The hardware requirements of different pallar SDF FFT architectures

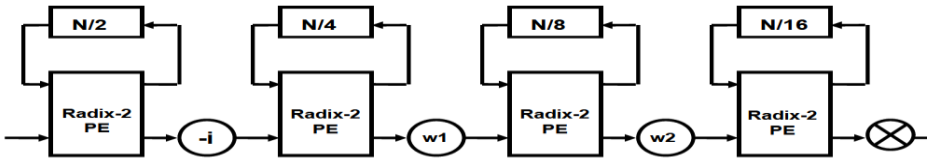
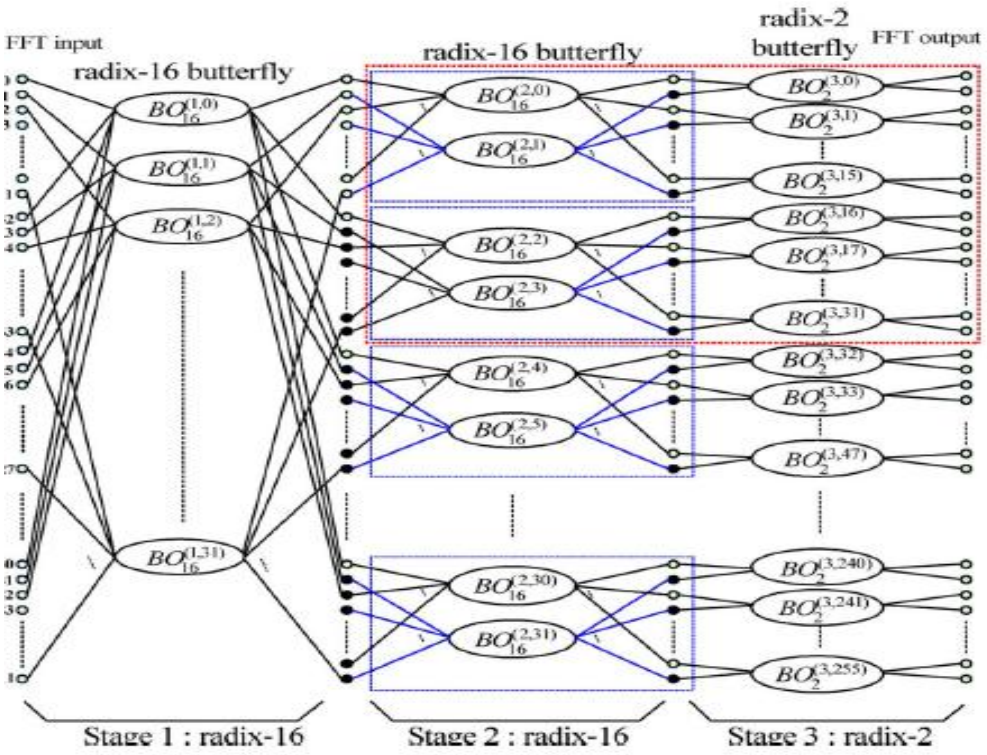


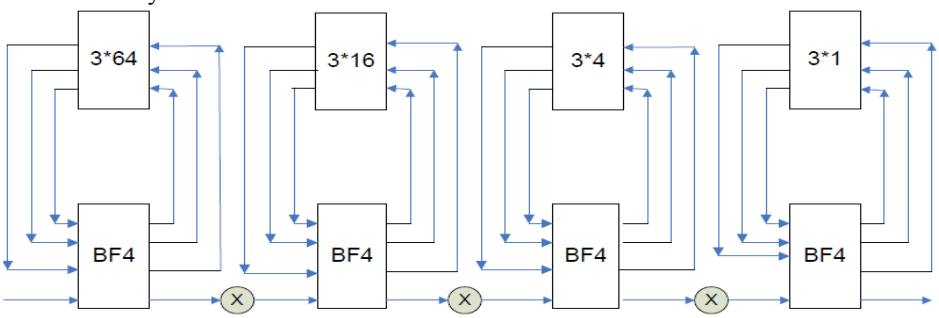
Fig. 5 SR2⁴SDF architecture for FFT



g. 2. SFG for radix-16 512-point FFT algorithm.

2.3 Pipelined Radix-16 Algorithem

To improve the performance of the sequential processor, parallelism can be introduced by using a separate arithmetic unit for each stage of the FFT. This increases the throughput by a factor of log2 N when the different units are pipelined. This architecture is also known as cascaded FFT architecture and will be used in our proposed design. Pipeline FFTs are a class of parallel algorithms that contain an amount of parallelism equal to logR N where N is the number of points for an FFT and R is the radix. A Pipeline implementation of the FFT was first proposed in which consisted of a series of computational blocks each composed of delay lines, coefficient storage, commutators, multipliers, and adders. Pipeline FFTs can be generally run at high-speeds and the amount of pipelining increased or decreased to meet timing. increased to 75% by storing 3 out of 4 butterfly outputs. However, the utilization of the radix-4 butterfly, which is fairly complicated and contains at least 8 complex adders, is dropped to only 25%. It requires logN − 1 multipliers, log, N full radix-4 butterflies and memory of size N − 1 words.



III. A Reformulated Radix-16 FFT Algorithm And Its Butterfly Structures

To fully exploit the advantage of a high-radix FFT algorithm particularly the radix-16 algorithm in this work, conventional FFT algorithm is reformulated in a form suitable for efficient realization of all the function components and overall FFT PE architecture, as will be detailed in the following

3.1 Given an -point discrete Fourier transform (DFT)

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

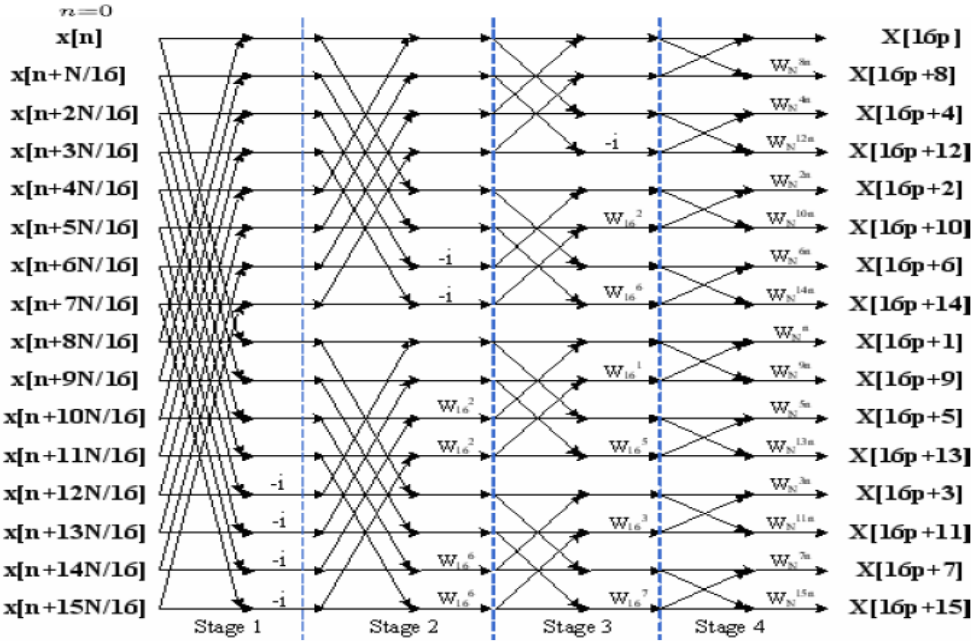


Fig. 2 Butterfly structure of the radix-16 FFT algorithm

where $x(n)$ and $X(k)$ denote the input and output of the D respectively, and W_N^{kn} is equal to $e^{-j2\pi kn/N}$. Let

$$N = 512,\ n = 32n_1 + n_2,\ k = 16k_2 + k_1,$$
$$n_1,\ k_1 = 0, 1, \ldots, 15; \quad n_2,\ k_2 = 0, 1, \ldots, 31.$$

A radix-16 decimation-in-frequency (DIF) FFT algorithm be derived based on the following first-stage decompositio substituting (3) into (2) as

$$X(16k_2 + k_1) = \sum_{n_2=0}^{31} \sum_{n_1=0}^{15} x(32n_1 + n_2) W_{512}^{(16k_2+k_1)(32n_1+n_2)}$$

$$= \sum_{n_2=0}^{31} \left\{ \underbrace{\sum_{n_1=0}^{15} x(32n_1 + n_2) W_{16}^{n_1 k_1}}_{16\text{-}point\ DFT\ of\ the\ 1st\ stage} \underbrace{W_{512}^{n_2 k_1}}_{\substack{\text{twiddle factor} \\ \text{of stage 1}}} \right\} W_{32}^{n_2}$$

$$= \sum_{n_2=0}^{31} \left\{ BO_{16}^{(1,n_2)}(k_1) \right\} W_{32}^{n_2 k_2}$$

3.2 Memory Based Archtecture

The combination of the pallar and pipeline process is the memory based architecture it mainly contains following three blocks those are

1. The inputs are given using pallar and pipelined process
2. The mux unit
3. The mixed radix unit

3.3 Implementation of Radix 16 Algorithem

The implementation of this algorithm is done in mat lad in mat lab the implementation is done in following steps

1. Radix 16 algorithm for 16 point is calculated
2. Then using the pallar and pipeline process we calculated the radix 16 for N point
3. Then memory based structure is implemented
4. Then radix 16 is extended to radix 32

Implementation radix 16 result:

```
f =
  Columns 1 through 3

   0.0000          -0.0000 + 0.0000i   0.0000 - 0.0000i

  Columns 4 through 6

   0.0000 + 0.0000i  -0.0000 +40.0000i   0.0000 - 0.0000i

  Columns 7 through 9

   0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000
```

```
  Columns 10 through 12

  -0.0000 + 0.0000i   0.0000 - 0.0000i   0.0000 + 0.0000i

  Columns 13 through 15

  -0.0000 -40.0000i   0.0000 - 0.0000i   0.0000 + 0.0000i

  Column 16

  -0.0000 - 0.0000i

twiddle =
     1
g = 0000000000000000
```

Implementation of radix 16 using pallor and pipeline for 256 point:

x =

```
  1.0e+002 *
  Columns 1 through 3
   1.9100           -0.0500 + 0.0400i  -0.0500
  Columns 4 through 6
  -0.0500 - 0.0400i   1.9100           -0.0500 + 0.0400i
  Columns 7 through 9
  -0.0500           -0.0500 - 0.0400i   1.9100
  Columns 10 through 12
  -0.0500 + 0.0400i  -0.0500           -0.0500 - 0.0400i
  Columns 13 through 15
   1.9100           -0.0500 + 0.0400i  -0.0500
  Columns 16 through 18
  -0.0500 - 0.0400i   0.0317 - 1.8129i   0.0239 + 0.0273i
  Columns 19 through 21
   0.0031 + 0.0311i  -0.0187 + 0.0346i   0.0317 - 1.8129i
  Columns 22 through 24
   0.0239 + 0.0273i   0.0031 + 0.0311i  -0.0187 + 0.0346i
  Columns 25 through 27
   0.0317 - 1.8129i   0.0239 + 0.0273i   0.0031 + 0.0311i
  Columns 28 through 30
  -0.0187 + 0.0346i   0.0317 - 1.8129i   0.0239 + 0.0273i
  Columns 31 through 33
   0.0031 + 0.0311i  -0.0187 + 0.0346i  -1.7028 + 0.1263i
  Columns 34 through 36
  -0.0341 - 0.0380i   0.0069 - 0.0420i   0.0499 - 0.0463i
  Columns 37 through 39
  -1.7028 + 0.1263i  -0.0341 - 0.0380i   0.0069 - 0.0420i
  Columns 40 through 42
   0.0499 - 0.0463i  -1.7028 + 0.1263i  -0.0341 - 0.0380i
  Columns 43 through 45
   0.0069 - 0.0420i   0.0499 - 0.0463i  -1.7028 + 0.1263i
  Columns 46 through 48
  -0.0341 - 0.0380i   0.0069 - 0.0420i   0.0499 - 0.0463i
  Columns 49 through 51
   0.3022 + 1.7777i  -0.0897 + 0.0683i  -0.0879 - 0.0165i
  Columns 52 through 54
  -0.0847 - 0.1096i   0.3022 + 1.7777i  -0.0897 + 0.0683i
  Columns 55 through 57
```

-0.0879 - 0.0165i -0.0847 - 0.1096i 0.3022 + 1.7777i
Columns 58 through 60
-0.0897 + 0.0683i -0.0879 - 0.0165i -0.0847 - 0.1096i
Columns 61 through 63
0.3022 + 1.7777i -0.0897 + 0.0683i -0.0879 - 0.0165i
Columns 64 through 66
-0.0847 - 0.1096i 1.8764 - 0.2897i 0.0614 + 0.1159i
Columns 67 through 69
-0.0352 + 0.0985i -0.1426 + 0.0753i 1.8764 - 0.2897i
Columns 70 through 72
0.0614 + 0.1159i -0.0352 + 0.0985i -0.1426 + 0.0753
Columns 73 through 75
1.8764 - 0.2897i 0.0614 + 0.1159i -0.0352 + 0.0985i
Columns 76 through 7
-0.1426 + 0.0753i 1.8764 - 0.2897i 0.0614 + 0.1159i
Columns 79 through 8
-0.0352 + 0.0985i -0.1426 + 0.0753i -0.2397 - 1.7855
Columns 82 through 84
0.0965 - 0.0731i 0.0944 + 0.0185i 0.0888 + 0.1201
Columns 85 through 87
-0.2397 - 1.7855i 0.0965 - 0.0731i 0.0944 + 0.0185i
Columns 88 through 90
0.0888 + 0.1201i -0.2397 - 1.7855i 0.0965 - 0.0731i
Columns 91 through 93
0.0944 + 0.0185i 0.0888 + 0.1201i -0.2397 - 1.7855i
Columns 94 through 96
0.0965 - 0.0731i 0.0944 + 0.0185i 0.0888 + 0.1201i
Columns 97 through 99
-1.6455 + 0.3718i -0.1293 - 0.0882i -0.0174 - 0.1221i
Columns 100 through 102
0.1122 - 0.1615i -1.645 + 0.3718i -0.1293 - 0.0882i
Columns 103 through 10
-0.0174 - 0.1221i 0.1122 - 0.1615i -1.6455 + 0.3718i
Columns 106 through 108
-0.1293 - 0.0882i -0.0174 - 0.1221i 0.1122 - 0.1615i
Columns 109 through 111
-1.6455 + 0.3718i -0.1293 - 0.0882i -0.0174 - 0.1221i
Columns 112 through 114
0.1122 - 0.1615i 0.5563 + 1.6822i -0.1225 + 0.1768
Columns 115 through 117
-0.1699 + 0.0238i -0.2239 - 0.1628i 0.5563 + 1.6822i
Columns 118 through 120
-0.1225 + 0.1768i -0.1699 + 0.0238i -0.2239 - 0.1628
Columns 121 through 123
0.5563 + 1.6822i -0.122 + 0.1768i -0.1699 + 0.0238i
Columns 124 through 126
-0.2239 - 0.1628i 0.5563 + 1.6822i -0.1225 + 0.1768i
Columns 127 through 129
-0.1699 + 0.0238i -0.2239 - 0.1628i 1.7777 - 0.5625i
Columns 130 through 132
0.1789 + 0.1506i 0.0077 + 0.1875i -0.2043 + 0.2244i
Columns 133 through 135
1.7777 - 0.5625i 0.1789 + 0.1506i 0.0077 + 0.1875i
Columns 136 through 138
-0.2043 + 0.2244i 1.7777 - 0.5625i 0.1789 + 0.1506i
Columns 139 through 141

0.0077 + 0.1875i -0.2043 + 0.2244i 1.7777 - 0.5625i
Columns 142 through 144
0.1789 + 0.1506i 0.0077 + 0.1875i -0.2043 + 0.2244i
Columns 145 through 147
-0.4963 - 1.6972i 0.131 - 0.1799i 0.1772 - 0.0201i
Columns 148 through 150
0.2279 + 0.1772i -0.4963 - 1.6972i 0.1312 - 0.1799i
Columns 151 through 153
0.1772 - 0.0201i 0.2279 + 0.1772i -0.4963 - 1.6972i
Columns 154 through 156
0.1312 - 0.1799i 0.1772 - 0.0201i 0.2279 + 0.1772i
Columns 157 through 159
-0.4963 - 1.6972i 0.1312 - 0.1799i 0.1772 - 0.0201i
Columns 160 through 162
0.2279 + 0.1772i -1.5349 + 0.5960i -0.2251 - 0.1042i
Columns 163 through 165
-0.0635 - 0.1906i 0.1435 - 0.3012i -1.5349 + 0.5960i
Columns 166 through 168
-0.2251 - 0.1042i -0.0635 - 0.1906i 0.1435 - 0.3012i
Columns 169 through 171
-1.5349 + 0.5960i -0.2251 - 0.1042i -0.0635 - 0.1906i
Columns 172 through 174
0.1435 - 0.3012i -1.5349 + 0.5960i -0.2251 - 0.1042i
Columns 175 through 177
-0.0635 - 0.1906i 0.1435 - 0.3012i 0.7795 + 1.5326i
Columns 178 through 180
-0.1190 + 0.2795i -0.2351 + 0.0857i -0.3855 - 0.1778i
Columns 181 through 183
0.7795 + 1.5326i -0.1190 + 0.2795i -0.2351 + 0.0857i
Columns 184 through 186
-0.3855 - 0.1778i 0.7795 + 1.5326i -0.1190 + 0.2795i
Columns 187 through 189
-0.2351 + 0.0857i -0.3855 - 0.1778i 0.7795 + 1.5326i
Columns 190 through 192
-0.1190 + 0.2795i -0.2351 + 0.0857i -0.3855 - 0.1778i
Columns 193 through 195
1.6207 - 0.8029i 0.2904 + 0.1460i 0.0741 + 0.2584i
Columns 196 through 198
-0.2252 + 0.3985i 1.6207 - 0.8029i 0.2904 + 0.1460i
Columns 199 through 201
0.0741 + 0.2584i -0.2252 + 0.3985i 1.6207 - 0.8029i
Columns 202 through 204
0.2904 + 0.1460i 0.0741 + 0.2584i -0.2252 + 0.3985i
Columns 205 through 207
1.6207 - 0.8029i 0.2904 + 0.1460i 0.0741 + 0.2584i
Columns 208 through 210
-0.2252 + 0.3985i -0.7233 - 1.5542i 0.1297 - 0.2817i
Columns 211 through 213
0.2435 - 0.0805i 0.3902 + 0.1964i -0.7233 - 1.5542i
Columns 214 through 216
0.1297 - 0.2817i 0.2435 - 0.0805i 0.3902 + 0.1964i
Columns 217 through 219
-0.7233 - 1.5542i 0.129 - 0.2817i 0.2435 - 0.0805i
Columns 220 through 222
0.3902 + 0.1964i -0.7233 - 1.5542i 0.1297 - 0.2817i
Columns 223 through 225

```
  0.2435 - 0.0805i   0.3902 + 0.1964i  -1.3783 + 0.7865i
 Columns 226 through 228
 -0.3116 - 0.0890i  -0.1265 - 0.2409i   0.1364 - 0.4566i
 Columns 229 through 231
-1.3783 + 0.7865i  -0.3116 - 0.0890i  -0.1265 - 0.2409i
 Columns 232 through 234
  0.1364 - 0.4566i  -1.3783 + 0.7865i  -0.3116 - 0.0890i
 Columns 235 through 237
 -0.1265 - 0.2409i   0.1364 - 0.4566i  -1.3783 + 0.7865i
 Columns 238 through 240
 -0.3116 - 0.0890i  -0.1265 - 0.2409i   0.1364 - 0.4566i
 Columns 241 through 243
  0.9596 + 1.3389i  -0.0841 + 0.3665i  -0.2773 + 0.1625i
 Columns 244 through 246
 -0.5582 - 0.1479i   0.9596 + 1.3389i  -0.0841 + 0.3665i
 Columns 247 through 249
 -0.2773 + 0.1625i  -0.5582 - 0.1479i   0.9596 + 1.3389i
 Columns 250 through 252
 -0.0841 + 0.3665i  -0.2773 + 0.1625i  -0.5582 - 0.1479i
 Columns 253 through 255
  0.9596 + 1.3389i  -0.0841 + 0.3665i  -0.2773 + 0.1625i
 Column 256
 -0.5582 - 0.1479i
```

Implementation of memory based : DEMO data are calculated as sum of the following components:

- real sinusoid at normalized frequency 0.137, magnitude 2;
- complex exponent at frequency 0.137+1/(4*64), magnitude 1;
- mean value (frequency 0), magnitude 1;
- rectangular pulse in frequency ranges [-0.137 0];
- noise generated by randn function (SNR~30 dB).

Input [1] to select uniformly sampled sequence X1
Input [2] to select nonuniformly sampled sequence X2
Input [3] to select sparse nonuniformly sampled sequence X3
Select input sequence for DEMO:2
Plots True Spectrum [red color], NEDFT [green colour] and DFT [blue colour]. Calculate [F,S]=nedft(X2,tk,fn,1) and ifft(F). Strike any key to continue.

IV. Conclusion

The proposed architecture achieves high throughput rate by employing several performance-enhancement techniques, including a reformulated radix-16 algorithm realized with multiple memory banks and PEs, a novel conflict-free memory addressing scheme, and a new twiddle factor multiplier structure. Also, high operation speed is obtained by devising an efficient pipelined PE structure, while the new twiddle factor multiplier has low hardware complexity and power consumption. Synthesis results showthat the proposed FFT processor can provide up to 2.59 GS/

V. Future scope

We can future extended this project by extending the radix or by implementing it in the another method rather than FFT and also we can processed for the radix 32 in the same way when we design this paper hardware using this process the control unit will be easy to design

REFERENCES

[1] IEEE 802.15.3c-2009: Part 15.3: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPANs).

[2] S. Johansson, S. He, and P. Nilsson, "Wordlength optimization of a pipelined FFT processor," in Proc. 42ndMidwest Symp. Circuits Syst.., 1999, pp. 501–503.

[3] S. He and M. Torkelson, "Designing pipeline FFT processor for OFDM (de)modulation," in Proc. URSI Int. Symp. Signals, Syst., Electron., 1998, pp. 257–262.

[4] Y. N. Chang and K. Parhi, "An efficient pipelined FFT architecture," IEEE Trans. Circuit Syst. II, Analog Digit. Signal Process., vol. 50, no. 6, pp. 322–325, Jun. 2003.
[5] D. Cohen, "Simplified control scheme of FFT hardware," IEEE Trans. Signal Process., vol. ASSP-24, no. 12, pp. 577–579, Dec. 1976.
[6] L. G. Johnson, "Conflict free memory addressing for dedicated FFT hardware," IEEE Trans. Circuit Syst. II, Analog Digit. Signal Process., vol. 39, no. 5, pp. 312–316, May 1992.
[7] Y. Ma, "An effective memory addressing scheme for FFT processors," IEEE Trans. Signal Process., vol. 47, no. 3, pp. 907–911, Mar. 1999.
[8] Y. Ma and L. Wanhammar, "A hardware efficient control of memory addressing for high-performance FFT processors," IEEE Trans. Signal Process., vol. 48, no. 3, pp. 917–921, Mar. 2000.
[9] B.M. Bass, "A low-power, high-performance, 1024-point FFT processor," IEEE J. Solid-State Circuits, vol. 34, no. 4, pp. 380–387, Mar. 1999.
[10] J. C. Kuo, C. H. Wen, and A. Y. Wu, "Implementation of a programmable 2048-point FFT/IFFT processor for OFDM-based communication systems," in Proc. 2003 Int. Symp. Circuit Syst., May 2003, pp. 121–124.
[11] Y. W. Lin, H. Y. Liu, and C. Y. Lee, "A dynamic scaling FFT processor for DVB-T applications," IEEE J. Solid-State Circuits, vol. 39, no. 11, pp. 2005–2013, Nov. 2004.