# An Effective Design of 128 Point FFT/IFFT Processor UWB Application Utilizing Radix - (16+8) Calculation

N.Pritha[1], D.Kalaiyarasi[2]

[1]Assistant Professor, Electronics and Communication Department, Panimalar Engineering College, Chennai, India

[2]Associate Professor, Electronics and Communication Department, Panimalar Engineering College, Chennai, India

***Abstract***— *In this paper, we present a 128-point FFT/IFFT processor for ultrawideband (UWB) systems. The proposed FFT is developed based on the higher radix -$2^4$. It reduces computational complexity and hardware requirement compared to conventional radix -2 FFT. Since the proposed pipelined architecture for 128-point FFT is designed using mixed-radix (16+8) multipath delay feedback (MRMDF). The proposed multi data-path scheme improves the through put rate significantly. In addition to that, the hardware costs of memory and complex multipliers in MRMDF are further reduced by means of the delay feedback and the data scheduling approach. The implementation of mixed radix algorithm is more area efficient than the conventional radix-2 algorithm*

**Keywords—** *Fast Fourier transform (FFT), Orthogonal frequency division multiplexing (OFDM), Ultrawideband(UWB), Mixed-radix multipath delayfeedback (MRMDF).*

## I. Introduction

Increasing speeds and complexity of wireless communication systems have necessitated the progress and advancement of high performance signal processing elements. Fast fourier transform (FFT) is an important signal processing block being widely used in communication systems, especially in orthogonal frequency division multiplexer (OFDM) systems such as digital video broadcasting-terrestrial(DVB-T) , digital audio broadcasting (DAB), very high-speed digital subscriber line (VDSL) and IEEE 802.16e. As the FFT processor is the most computationally intensive component in OFDM communication, an improvement in the power efficiency of this component can have great impacts on the overall system. These impacts are significant considering the number of mobile and remote communication devices that rely on limited battery-powered operation.

Ultra-Wideband (UWB) Technology brings the comfort and versatility of remote correspondences to fast interconnect in gadgets all through the advanced home and office [1]. Rather than wired association, this innovation empowers remote association for transmitting video, sound, and other information with high information speed and low power utilization. Multiband-OFDM standard is one answer for UWB innovation. High Rate OFDM-based UWB not just has dependably high-information rate transmission in time-dispersive or recurrence particular channels without having complex time-space channel equalizers additionally can give high ghastly productivity. In any case, in light of the fact that the information inspecting rate from the simple to computerized converter to the physical layer is up to 528 M test/s or more, it is a test to understand the physical layer of the UWB system particularly the segments with high computational many-sided quality in VLSI usage. The FFT/IFFT processor is one of the module having high computational complexity in the physical layer of the UWB system, and the execution time of the 128-point FFT/IFFT in UWB system is just 312.5 ns. Therefore, if utilizing the conventional methodology, a lot of force utilization and high equipment expense of the FFT/IFFT processor will be expected to meet the strict determinations of the UWB system. Thus, this paper proposes a FFT/IFFT processor with a multipath pipelined architecture for high-throughput-rate applications. The power consumption and hardware cost can also be saved in our processor by using the higher radix FFT algorithm and less memory and complex multipliers.

Many FFT algorithms have been proposed, such as radix-$2^2$ [2], radix-$2^3$ [3], radix-(4+2) [4], split-radix [5], [6], vector-radix [6], and prime factor algorithms [7], to reduce the hardware complexity of the multiplier, butterfly (BF) and controller. Although the previous algorithms could reduce the computational hardware resources such as multipliers and adders, they did not seriously take into account the number of twiddle factors required in the FFT processing. To efficiently process the FFT, a number of hardware architectures have been proposed, including serial processing on a single processor [8], pipelined processing [11], [12], and parallel processing [9], [10]. Among them, the pipelined processing is preferred, as it can provide a high performance with a moderate hardware cost. This paper proposes the design of pipelined architecture of mixed radix FFT/IFFT for UWB-OFDM.

This paper is organized as follows. Section II identifies the problems for implementation of the FFT/IFFT processor in UWB system. Section III describes the proposed algorithm for 128-point mixed-radix FFT/IFFT. Section IV presents the details about module and comparison of hardware complexity and throughput rate of proposed method with existing FFT architectures. Then, conclusions are drawn in Section V.

## II. UWB- OFDM SYSTEMS

A block diagram of the proposed physical layer of OFDM-based UWB system is shown in Fig. 1. It contains a convolutional encoder, a Viterbi decoder, a pilot insertion, a QPSK-modulator/demodulator, a spreading/ de-spreading, a guard interval insertion/ removal, a 128-point FFT/IFFT, a serial-to-parallel (S/P) converter/ parallel-to-serial (P/S) converter, an analog-to-digital converter (ADC), a digital-to-analog converter (DAC), and a synchronization and channel estimation block. In order to implement the physical layer of the UWB system more efficiently, the four-data-path approach is adopted to reduce the data sampling rate from the ADC [18].

Various FFT architectures, such as single-memory architecture, dual-memory architecture [13], pipelined architecture [3], array architecture [15], and cached-memory architecture [16], have been proposed in the last three decades. In our view, the pipelined architecture should be the best choice for high-throughput-rate applications since it can provide high throughput rate with acceptable hardware cost.

The pipelined FFT architecture typically falls into one of the two following categories. One is multipath delay commutator (MDC) and the other is single-path delay feedback (SDF). In general, if appropriately reordered M parallel input data can be supported simultaneously in the MDC scheme. This scheme provides M times throughput rate of the SDF scheme. However, there are some limitations on the number of data path, the FFT size, and the radix-r FFT algorithm in the MDC architecture. Besides, the requirement of the memory and complex multiplier in the MDC scheme is more than that of the SDF scheme. In general, the MDC scheme can achieve a higher throughput rate, while the SDF scheme needs less memory and hardware cost.

To incorporate both, this paper utilize a design of four-data-path pipelined FFT architecture, which is called mixed-radix multipath delay feedback (MRMDF), by combining the features of the SDF and MDC architectures. The proposed FFT/IFFT processor not only suits the proposed UWB physical layer, as shown in Fig. 1, but also can provide an available throughput rate to meet the UWB specifications. The MRMDF architecture has lower hardware cost compared with the traditional MDC approach and adopts the high-radix FFT algorithm to save power dissipation.

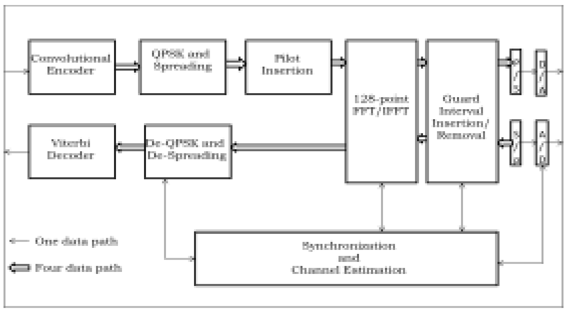


Fig.1- Block diagram of physical layer of OFDM-based UWB system

## III. PROPOSED ALGORITHM FOR 128- POINT MIXED - RADIX FFT/IFFT

Given a complex input sequence x(n), an N-point discrete Fourier transform (DFT) is defined as

$$X(k) = \sum_{n=0}^{N} x(n)\, W_N^{kn}, \quad K = 0,1,2 \ldots\ldots 15 \qquad (1)$$

where $W_N^{kn}$ is known as the twiddle factor. Due to the high computational complexity in (1), the radix-r FFT algorithms which have lower complexity have been widely adopted in DFT implementations.
In (1), the computational complexity is $O(N^2)$ through directly performing the required computation. By using the FFT algorithm, the computational complexity can be reduced to $O(N \log_r N)$, where r means the radix-r FFT. The radix-r FFT algorithm can be easily derived from the DFT by decomposing the N-point DFT into a set of recursively related r point FFT transform, if x(n) is power of r. In general, higher-radix FFT algorithm has less number of complex multiplications compared with radix-2 FFT algorithm which is the simplest form in all FFT algorithms. In an example, for the 128-point FFT, the number of nontrivial complex multiplications of the radix-$2^4$ FFT algorithm is 105, which is only 35% of that of the radix-$2^3$ FFT algorithm. Thus, in order to save the number of complex multiplications, we choose the radix-$2^4$ FFT algorithm. Because the 128-point FFT is not a power of 8, the mixed-radix FFT algorithm, including radix-$2^4$ and radix-$2^3$ FFT algorithms, is needed. This will be derived in detail below. First let

$n=\frac{N}{16}n_1 + n_2$; Where $n_1=0,1\ldots.15$
Where $n_2=0,1\ldots.7$
$k=k_1+16\,k_2$; Where $k_1=0,1\ldots.15$
Where $k_2=0,1\ldots.7$ (2)

$$X(\mathrm{k}_1 + 16\,\mathrm{k}_2) = \sum_{n_2}^{7}\sum_{n_1}^{15} x\left(\frac{N}{16}n_1 + n_2\right)\mathrm{W}_{\mathrm{N}}^{\left(\frac{N}{16}n_1+n_2\right)(K_1+16K_2)}$$

$$= \sum_{n_2}^{7}\left[\sum_{n_1}^{15} x\left(\frac{N}{16}n_1 n_2\right)\mathrm{W}_{\mathrm{N}}^{\left(\frac{N}{16}n_1K_1\right)}\right]\mathrm{W}_{\mathrm{N}}^{n_2(K_1+16K_2)}$$

$$= \sum_{n_2}^{7}\left[\sum_{n_1}^{15} x\left(\frac{N}{16}n_1 + n_2\right)W_{16}^{n_1k_1}W_{128}^{n_2k_1}\right]W_8^{n_2k_2}$$

$$X(\mathrm{k}_1 + 16\,\mathrm{k}_2) = \sum_{n_2=0}^{7}[BF_{16}(n_2,k_1)]\,W_8^{n_2k_2} \tag{3}$$

Equation (3) can be considered as processing a two-dimensional DFT where one dimension is built by a 16-point DFT and other by 8-point DFT. To implement the 16-point DFT more efficiently, we use the radix-2 FFT algorithm which can easily be derived from the radix-2 algorithm [3]. By further decomposing the radix-16 DFT into four steps by applying radix-2 index map, we first rewrite $n_1$ and $k_1$ as

$$n_1 = 8\alpha_1 + 4\alpha_2 + 2\alpha_3 + \alpha_4;\alpha_1 = \alpha_2 = \alpha_3 = 0,1$$
$$k_1 = \beta_1 + 2\beta_2 + 4\beta_3 + 8\beta_4\,;\,\beta_1 = \beta_2 = \beta_3 = 0,1 \tag{4}$$

$$\sum_{n_1=0}^{15} x(8n_1 + n_2)W_{16}^{n_1k_1}$$

$$= \sum_{\alpha_4=0}^{1}\sum_{\alpha_3=0}^{1}\sum_{\alpha_2=0}^{1}\sum_{\alpha_1=0}^{1} x\,(8(8\alpha_1 + 4\alpha_2 + 2\alpha_3 + \alpha_4)$$
$$+ n_2)W_{16}^{(8\alpha_1+4\alpha_2+2\alpha_3+\alpha_4)(\beta_1+2\beta_2+4\beta_3+8\beta_4)}$$

$$= \sum_{\alpha_4=0}^{1}\sum_{\alpha_3=0}^{1}\sum_{\alpha_2=0}^{1}\sum_{\alpha_1=0}^{1} x\,((64\alpha_1 + 32\alpha_2 + 16\alpha_3 + 8\alpha_4)$$
$$+ n_2)W_2^{\alpha_1\beta_1}W_4^{\alpha_2\beta_1}W_2^{\alpha_2\beta_2}W_8^{\alpha_3(\beta_1+2\beta_2)}W_2^{\alpha_3\beta_3}$$
$$W_{16}^{\alpha_4(\beta_1+2\beta_2+4\beta_3)}W_2^{\alpha_4\beta_4} \tag{5}$$

By means of (4), the first 16-point DFT in (3) is rewritten as (5), where the radix-16 DFT is completed by the four-step radix-2 algorithm. Similarly, by reformulating $n_2$ and $k_2$ with

$$n_2 = 4\gamma_1 + 2\gamma_2 + \gamma_3;\text{Where } \gamma_1 = \gamma_2 = \gamma_3 = 0,1$$
$$k_2 = \lambda_1 + 2\lambda_2 + 4\lambda_3;\text{Where } \lambda_1 = \lambda_2 = \lambda_3 = 0,1 \tag{6}$$

$$X(\mathrm{k}_1 + 16\,\mathrm{k}_2) = \sum_{n_2=0}^{7}[BF_{16}(n_2,k_1)]\,W_8^{n_2k_2}$$

$$= \sum_{\gamma_3=0}^{1}\sum_{\gamma_2=0}^{1}\sum_{\gamma_1=0}^{1}[BF_{16}(k_1,\gamma_1,\gamma_2,\gamma_3)]\,W_8^{(4\gamma_1+2\gamma_2+\gamma_3)(\lambda_1+2\lambda_2+4\lambda_3)}$$

$$= \sum_{\gamma_3=0}^{1}\sum_{\gamma_2=0}^{1}\sum_{\gamma_1=0}^{1}[BF_{16}(k_1,\gamma_1,\gamma_2,\gamma_3)]\,W_2^{\gamma_1\lambda_1}W_4^{\gamma_2\lambda_1}W_2^{\gamma_2\lambda_2}$$
$$W_8^{\gamma_3(\lambda_1+2\lambda_2)}W_2^{\gamma_3\lambda_3} \tag{7}$$

The second 8-point DFT in (3) can also be transformed to radix-2 FFT as (7).

Thus, by (3), (5), and (7), the 128-point radix-2 FFT is fully computed. The corresponding signal flow graph (SFG) of this 128-point FFT is shown in Fig.2. Note that the 128-point FFT is divided into two stages, where radix-16 and radix-8 FFT algorithm is employed in first and second stage respectively. The black point between these two stages means that the twiddle factor will be multiplied at that point. First butterfly of the radix-16 algorithm is further decomposed into four steps. The twiddle factors, $W_8^1$ and $W_8^3$ in the second step and $W_{16}^n$ in the third step, can be implemented by constant complex multiplications with simple shifters and adders to further reduce the hardware complexity

The IFFT of an N-point sequence X(K), k=0,1,….N-1 is defined as

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)W^{-nk} \tag{8}$$

If we take the complex conjugate of (8) and multiply both sides by N, we find

$$Nx^*(n) = \sum_{k=0}^{N-1} X^*(k)W^{nk} \tag{9}$$

By taking the complex conjugate of (9) and dividing both sides by N, the desired output sequence x(n) is given by

$$x(n) = \frac{1}{N}\left\{\sum_{k=0}^{N-1} X^*(k)W^{nk}\right\}^* \tag{10}$$

According to (10), the IFFT can be performed by taking the complex conjugate of the incoming data first and then the outgoing data without changing any coefficients in the original FFT algorithm so that the hardware implementation can be more efficient.

## IV. PROPOSED FFT ARCHITECTURE FOR UWB SYSTEM

The block diagram of the proposed 128-point FFT/IFFT processor derived in section III and the SFG (Fig.2) is depicted in Fig. 3. The proposed MRMDF architecture combining the features of the SDF and MDC architectures consists of Module 1, Module 2. The features of the proposed MRMDF architecture are the following:

- Four parallel data paths provide higher throughput rate.
- Delay feedback approach reduces memory size to reorder the input data and the intermediate results of each module.
- To save power consumption, the 128-point mixed-radix FFT/IFFT algorithm is used.

- Using the scheduling scheme and the specified constant multipliers, the number of complex multiplier is minimized.

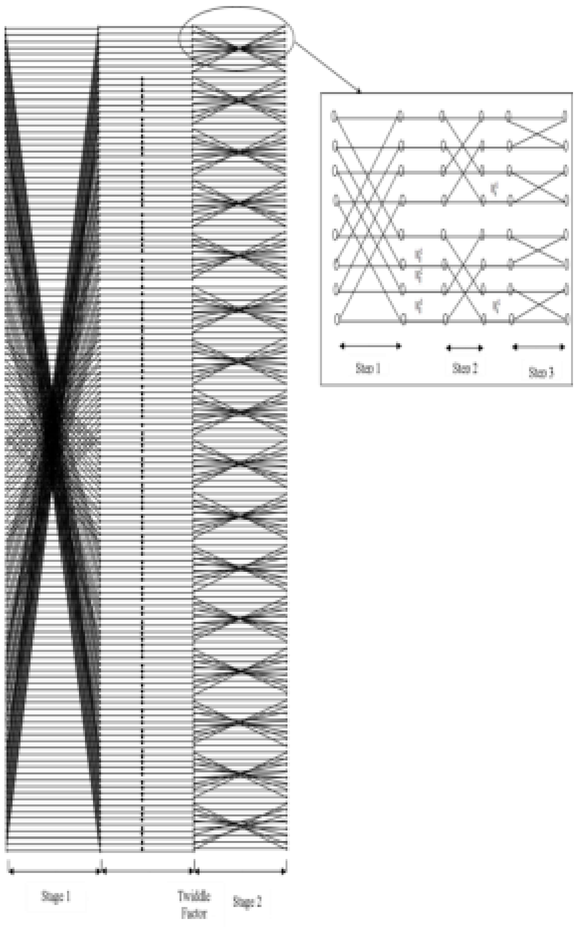


Fig.2. Signal flow graph for 128 point FFT

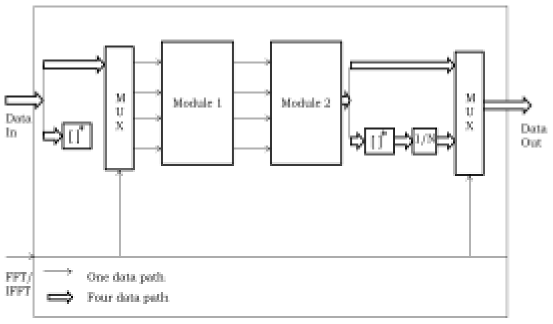


Fig.3. Block diagram of proposed 128 point FFT/IFFT

The function of Module 1 is to implement a radix-$2^4$ FFT algorithm, corresponding to the first stage of the SFG, as shown in Fig.2. Modules 2 is to realize the radix-$2^3$ FFT algorithm, corresponding to the second stage of the SFG, as displayed in Fig.2. In order to minimize the memory requirement and to ensure the correction of the FFT output data, different structure is built in Modules 2 to implement the radix-$2^3$ FFT algorithm. In addition, the hardware complexity of the complex multiplier will be also considered in the proposed architecture. In the next few subsections, each Module will be described in more detail.

## A. *Module 1*

Module 1 consists of four BU 16 structures and one modified complex multiplier, as shown in Fig.4. These four BU 2s operate in the same way. The architecture of BU 16 ($2^{4)}$ is directly mapped from the four-step radix-2 FFT algorithm, whose SFG is shown in Fig. 2. Also, the sizes of the three delay elements in the BU 16 are sixteen, eight, four and two points respectively, as shown in Fig.4. The function of the delay element is to store the input data until the other available input data are received for the BU 2 operation. The output data generated by the BU 2 in the first step and second step are multiplied by a trivial twiddle factor 1, -j, $W_8^1$ or $W_8^3$ before they are fed to the next step. These twiddle factors can be implemented efficiently, as mentioned in Section III. However, the four output data from the third step of the BU 2 need to be multiplied by the nontrivial twiddle factors simultaneously in the modified complex multiplier.

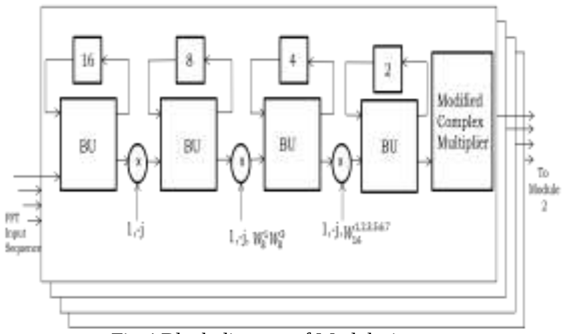


Fig.4.Block diagram of Module 1

It is inefficient to build four complex multipliers for multiplying different twiddle factors simultaneously. So modified approach [17] is used to simplify the complexity of the complex multipliers. The twiddle factors of the modified complex multiplier are , where $W_{128}^{p}$(e(-j2πp)/128) = Xp +j Yp, Xp =cos($2\pi p/128$) and Yp =cos($2\pi p/ 128$) are the real and imaginary parts of the twiddle factor and is from 0 to 105, as shown in Fig. 5. However, only nine sets of constant values, with 0–8 in region A are needed, because the twiddle factor in the other fifteen regions can be obtained by using the mapping table, as seen in Table I. In practice, we only need to implement eight sets of constant values in the A region, since the first set of constant values (1, 0) is trivial. In addition, these constant values can be realized more efficiently by using several adders and shifters [17].

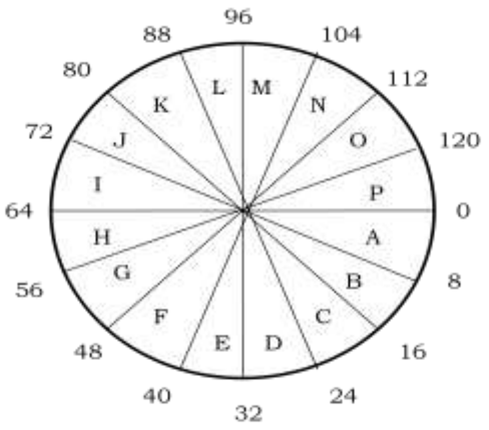


Fig.5. Sixteen Regions are divided for Twiddle Factors

TABLE I .MAPPING TABLE TO DETERMINE THE VALUE OF THE TWIDDLE FACTORS IN DIFFERENT REGIONS

| Region | Real part | Imaginary Part |
|---|---|---|
| A-B | Xp | Yp |
| C-D | - Yp | - Xp |
| E-F | Yp | - Xp |
| G-H | - Xp | Yp |
| I-J | - Xp | - Yp |
| K-L | Yp | Xp |
| M-N | - Yp | Xp |
| O-P | Xp | - Yp |

Table II shows the scheduling of the twiddle factor in each data path after the twiddle factors are mapped to region A. It can be clearly seen that the twiddle factor of four paths in each time slot has different values, except for time slots 2 and 3. In time slots 2 and 3, the hardware conflict will happen if only one constant multiplier 4 is built. Therefore, an additional constant multiplier 4 is used in our design to avoid spending one more cycle. The block diagram of the modified complex multiplier is shown in Fig. 6. In the beginning, the four output sequences from the third step of the BU 8($2^3$) are separated into real and imaginary parts. The data of each path are fed to appropriate constant multipliers according to the scheduling of the twiddle factor, as shown in Table II. Therefore, the entire constant multiplication calculation can be implemented by just using eight sets of constant values by swapping the real and imaginary parts appropriately and choosing the appropriate sign, following the mapping table (Table I). The gate count of this approach can save about 38% compared to the four-complex-multiplier approach, and the performance of this approach is equivalent to that of the four complex multipliers.

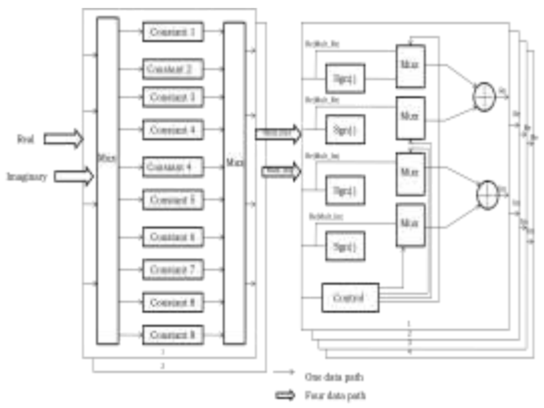


Fig.6.Block diagram of Modified complex multiplier

TABLE II SCHEDULING OF THE TWIDDLE FACTOR$W_{128}^{p}$, WHERE P IS FROM 1TO 8

| Data path | Time Slot | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0/1 | 2/3 | 4/6 | 5/7 | 8/10 | 9/11 | 12/14 | 13/15 |
| 1st | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 8 |
| 2nd | 0 | 8 | 4 | 4 | 2 | 2 | 6 | 6 |
| 3rd | 0 | 0 | 8 | 8 | 4 | 4 | 4 | 4 |
| 4th | 0 | 8 | 4 | 4 | 6 | 6 | 2 | 2 |
| Data path | Time Slot | | | | | | | |
| | 16/18 | 17/19 | 20/22 | 21/23 | 24/26 | 25/27 | 28/30 | 29/31 |
| 1st | 0 | 4 | 0 | 4 | 0 | 4 | 0 | 4 |
| 2nd | 1 | 5 | 5 | 1 | 3 | 7 | 7 | 3 |
| 3rd | 2 | 6 | 2 | 6 | 6 | 2 | 6 | 2 |
| 4th | 3 | 7 | 7 | 3 | 1 | 5 | 5 | 1 |

## *B. Module 2*

The radix-$2^3$ FFT algorithm is realized in Module 2. The detailed block diagram of Module 2 is shown in Fig. 7. The structure of Module 2 is different from that of Module 1, because the two available data of the BU 2 in the second and third steps are in different data paths. Thus, a suitable structure is needed to ensure the correction of the FFT output data. Some output data, generated by the BU 2 in the first and second steps, are multiplied by the nontrivial twiddle factors before they are fed to the next step

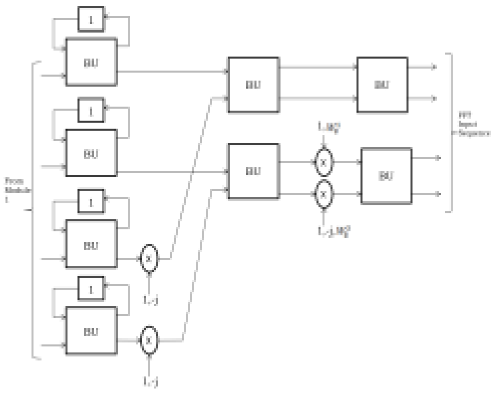
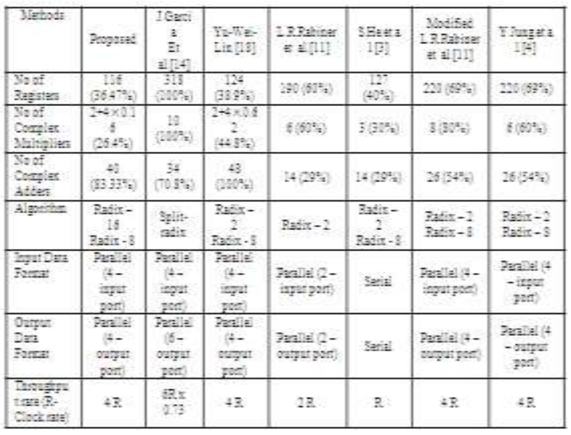


Fig.7.Block diagram of Module 2

## V. COMPARISON

In general, the performance and hardware cost of the pipelined FFT architecture are increased by using the multiple data-path approach. Thus, the multipath-based architecture usually provides higher throughput rate with higher hardware cost, if the parallel input data can be supported in this approach. Table III compares the hardware requirement, FFT algorithm and throughput rate with several existing methods and proposed approach in the 128 point FFT. The number of register excluding the input buffer and complex multiplier used in our approach are only 36.47% and 26.4% of those in existing architecture [14] and also enhance the throughput rate by four times using radix- (16+8) algorithm compared to existing [3].

TABLE III COMPARISON OF HARDWARE COMPLEXITY



| Methods | Proposed | J Garcia Et al [14] | Yu-Wei-Lin [18] | L.R.Rabiner et al [11] | S.He et al [3] | Modified L.R.Rabiner et al [11] | Y Jung et al [4] |
|---|---|---|---|---|---|---|---|
| No of Registers | 116 (36.47%) | 318 (100%) | 124 (38.9%) | 190 (60%) | 127 (40%) | 220 (69%) | 220 (69%) |
| No of Complex Multipliers | 2+4×0.16 (26.4%) | 10 (100%) | 2+4×0.62 (44.8%) | 6 (60%) | 3 (30%) | 3 (30%) | 6 (60%) |
| No of Complex Adders | 40 (83.33%) | 34 (70.8%) | 48 (100%) | 14 (29%) | 14 (29%) | 26 (54%) | 26 (54%) |
| Algorithm | Radix – 16 Radix - 8 | Split-radix | Radix – 2 Radix - 8 | Radix – 2 | Radix – 2 Radix - 8 | Radix – 2 Radix – 8 | Radix – 2 Radix – 8 |
| Input Data Format | Parallel (4 – input port) | Parallel (4 – input port) | Parallel (4 – input port) | Parallel (2 – input port) | Serial | Parallel (4 – input port) | Parallel (4 – input port) |
| Output Data Format | Parallel (4 – output port) | Parallel (6 – output port) | Parallel (4 – output port) | Parallel (2 – output port) | Serial | Parallel (4 – output port) | Parallel (4 – output port) |
| Throughput rate (R-Clock rate) | 4 R | 6R x 0.73 | 4 R | 2 R | R | 4 R | 4 R |

## VI. CONCLUSION

In this paper, a 128-point FFT/IFFT processor for OFDM-based UWB systems has been proposed. In our proposed MRMDF architecture, high throughput rate can be achieved by using four data paths. Furthermore, the hardware costs of memory and complex multiplier can be saved by adopting delay feedback and data scheduling approaches. In proposed approach, the number of complex multiplier and registers are effectively reduced by 26.4% and 36.47% respectively. In future, the same will be implemented using FPGA.

## REFERENCES

[1] 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.

[2] S. He and M. Torkelson, "Design and implementation of a 1024-point pipeline FFT processor," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC'98)*, May 1998, pp. 131–134.

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

[4] Y. Jung, H. Yoon, and J. Kim, "New efficient FFT algorithm and pipeline implementation results for OFDM/DMT applications," *IEEE Trans. Consum. Electron.*, vol. 49, no. 1, pp. 14–20, Feb. 2003.

[5] P. Duhamel and H. Hollomann, "Split radix FFT algorithm," *Electron. Lett.*, vol. 20, no. 1, pp. 14–16, Jan. 1984.

[6] D. Takahashi, "An extended split-radix FFT algorithm," *Electron. Lett.*, vol. 8, no. 5, pp. 145–147, May 2001.

[7] R. Wu and F. J. Paoloni, "The Structure of vector radix fast Fourier transforms," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 9, pp. 1415–1424, Sep. 1989.

[8] J. Ja'Ja' and R. M. Owens, "An architecture for a VLSI FFT processor,"*VLSI J. Integr.* , vol. 1, no. 4, pp. 305–316, 1983.

[9] H. S. Lee, "Systolic array architecture for VLSI FFT processor," *Int. J.Mini Microcomput.*, vol. 6, no. 3, pp. 49–54, 1984.

[10] H. Stone, "Parallel processing with the perfect shuffle," *IEEE Trans. Comput.*, vol. C-20, no. 2, pp. 153–161, Feb. 1971.

[11] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

[12] E. Swartzlander, V. K. W. Young, and S. J. Joseph, "A radix 4 delay commutator for fast Fourier transform processor implementation," *IEEE J. Solid-State Circuits*, vol. SC-19, no. 5, pp. 702–709, Oct.1984.

[13] W.-C. Yeh and C.-W. Jen, "High-speed and low-power split-radix FFT," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 51, no. 3, pp.864–874, Mar. 2003.

[14] J.Garcia, J.A.Michel and A.M.Buron,"Vlsi configurable delay commutator for a pipeline split radix FFT architecture," *IEEE Trans. Signal Process*, vol. 47, no.11 Nov. 1999, pp.3098–3107.

[15] J. O'Brien, J. Mather, and B. Holland, "A 200 MIPS single-chip 1 k FFT processor," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, vol.36, 1989, pp. 166–167.

[16] B. M. Bass, "A low-power, high-performance, 1024-point FFT processor," *IEEE J. Solid-State Circuits*, vol. 34, no. 3, pp. 380–387, Mar.1999.

[17] K. Maharatna, E. Grass, and U. Jagdhold, "A 64-point fourier transformchip for high-speed wireless lan application using OFDM," *IEEEJ. Solid-State Circuits*, vol. 39, no. 3, pp. 484–493, Mar. 2004.

[18] Y-W.Lin, H.Y.Liu, and C-Y.Lee,"A 1GS/s FFT/IFFT processor for UWB applications," *IEEE J. Solid-State Circuits*, vol. 40, no. 8, pp. 1726–1735, Aug.2005.