HIGH PERFORMANCE WITH REDUCED AREA 4096 POINT FEEDFORWARD FFT ARCHITECTURE FOR VDSL APPLICATIONS

A. ARUN C A B. PRAKASAM PERIYASAMY
A. Paavai College of Technology, Namakkal, India.
B. Department of ECE, SNS College of Engineering, Coimbatore, India.
Email – prakasamp@gmail.com

Abstract: *In this paper, a new representation for Fast Fourier Transform (FFT) algorithms based on feedforward FFT architecture is proposed. A 4096 point pipelined Feedforward FFT processor is designed to achieve high throughput for Very high-speed Digital Subscriber Line (VDSL) and IEEE 802.16e standard applications. The proposed hardware architecture has been designed based on reducing the number of rotators and their complexity by finding the efficient distribution of FFT rotations. The proposed 2 - parallel, 4 - parallel and 8 - parallel radix-2k Feedforward MDC architecture is compared with previous FFT architectures. From the analysis the proposed 4-parallel architecture savings the area of rotators increased from 17% to 23% and the proposed 8- parallel architecture save around 23% to 29% with respect to the existing architectures.*

Key words: *Fast Fourier Transform (FFT), Radix-2i algorithm,Pipelined Architecture, Coordinate Rotation Digital Computer(CORDIC) and Combined Coefficient Selection and Shift-and-add Implementation (CCSSI).*

1. Introduction

Fast Fourier Transform (FFT) is a crucial part of modern digital communication systems such as Digital Video Broadcasting - Terrestrial (DVB-T) [1], Ultra Wideband Systems (UWB) and Orthogonal Frequency Division Multiplexing (OFDM) based systems. OFDM technique is used in the applications, which need of a large point of FFT processor such as 4096 point FFT in Very high speed Digital Subscriber Line (VDSL). Fast Fourier Transform and Inverse Fast Fourier Transform (IFFT) are central part functions in such multi-carrier modulations based transmission systems [2]. For provided that high performance and meet up the real-time necessities of recent applications, hardware designers have constantly tried to put into practice efficient architectures for the working out of the FFT [3].

Gokhan [4] proposed full parallel a 256- point FFT processor for 100 GB/s applications, which uses the radix-4 scheme to reduce complexity as well as maintaining high-speed data flow. Hun-Sik Kang [5] reported a 256-point FFT/IFFT processor adopts 32-paths parallel architecture with mixed radix-2^3 and radix-2^5 algorithm to reduce a large number of complex multipliers. In this architecture [5], 256 full parallel data signals can be divided into eight sequential groups. Each group, composed of 32 parallel signals, is pipeline- processed. Chao Wang [6] proposed a structure to meet the throughput of 1.76 GS/s and reduce the complexity of twiddle factors; an eight-parallel radix-2^4-2^2-2^3 MRMDF structure is adopted. Furthermore, a novel single-RAM-group reorder buffer with simple control logic for parallel and pipelined FFT structures is proposed.

Jun-Feng Tang [7] reported 512-point FFT has been realized by a radix-2^3 and modified radix-2^5 algorithm. Hardware complexity is effectively reduced by replacing the complex multiplier with trivial multipliers. Chu Yu [8] proposed the data memory of the proposed FFT processor employs single-port register files instead of two port memories. Tram Thi Bao Nguyen [9] proposed a single shared Canonical Signed Digit code complex constant multiplier for two parallel data paths is used to reduce the hardware cost for parallel FFT processors. Seong-Weon Ko [10] presented the novel cooperative OFDM scheme with Signal Space Diversity (SSD), which transmits both of their signals in one-time slot. From the results, the throughput of the proposed scheme is increased by twice as compared with the Decode and Forward Cooperative OFDM scheme and it requires 3dB more power. R. Simon Sherratt [11] has presented a tested and implemented architecture to reduce the baseband processing clock rate by a factor of two by utilizing two parallel symbol processing paths which are then finally merged into a Double Data Rate (DDR) path for the transmission path, or demultiplexing the received DDR path into two processing paths.

From the past decade different FFT algorithms have been proposed, from the research, the previous algorithms focused mainly to trim down the hardware

complexity of multipliers and adders. The researchers do not give that much importance to reduce the phase vectors or twiddle factors required in the processor. The phase vectors generation, storage and multiplying with input signal requires a lot of hardware resources. The most complex process in FFT processor is phase vectors multiplication. Generally, ROM tables are used to store the phase vectors, when implementing a huge point FFT processor, the table required to store the phase vectors becomes large and it requires more area in the design. The complexity of the algorithm depends on the number phase vectors multiplication.

Different hardware architecture has been proposed to design a high-speed FFT/IFFT processor. High performance can be achieved by using pipelined processing with a reasonable hardware cost. Pipeline architecture requires completely scheduled operation sequences. The pipelined architecture is characterized by continuous processing of input data. In addition, the pipeline architecture is highly regular, making it straight forward to automatically generate FFT's of various lengths. It can be classified as Single path Delay Feedback (SDF), Multipath Delay Feedback (MDF), Single path Delay Commutator (SDC) and Multipath Delay Commutator (MDC) based on the quantity of data to be processed at a time.

Generally, a pipeline FFT processor is designed using one of two popular methods. The first is Single-path Delay Feedback (SDF) pipeline architecture and the second is Multipath Delay Commutator (MDC) pipeline architecture. In feedforward architecture there is no feedback loops and the processed data on each stage is forwarded to the next step. In Feedforward architecture the utilization ratio of butterflies, memory as well as efficient use of rotators is 100%. Due to that, the required number of adders is reduced by half in feedforward architectures [12]-[13].

In the proposed system eight parallel 4096 point feedforward radix-2^4 Multipath Delay Commutator (MDC) FFT / IFFT processor is designed to improve the performance of the OFDM system. In this paper, various approaches for twiddle factor multiplications are discussed; a feedforward radix-2^4 MDC Feedforward pipeline structure is adopted in our design. It improves the performance of FFT / IFFT processor in terms of reducing the multiplier complexity and also reduces a normalized area of the processor. The organization of this brief is as follows: Radix-2^i algorithm is discussed in Section II and Section III describes the rotators. Rotator allocation is explained in Section IV and the proposed 2-parallel, 4-parallel and 8-parallel 4096 point feedforward radix-2^4 MDC architecture is explained in Section V. Implementation of the proposed model is shown in Section VI and finally, conclusions are provided in Section VII.

2. Radix – 2^i Algorithm

In many signal processing algorithms, input signal is multiplied by a complex number which has magnitude is equal to one. For example, Fast Fourier Transform (FFT), fast Discrete Cosine Transforms (DCT), FIR filters and IIR filters follows such technique. Carry out the frequency analysis on discrete time signals x(n), the time-domain sequences are converted to an equivalent frequency domain representation. It leads to the Discrete Fourier Transform (DFT) computation; the functions X(k) and x(n) are represented by

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N}, k = 0,1,2,..N-1. \quad (1)$$

The Frequency domain data can be changed to time domain by using Inverse Discrete Fourier Transform (IDFT) in which the X(k) is transformed back to x(n) [14]-[15].

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j2\pi nk/N}, n = 0,1,2,...N-1. \quad (2)$$

The DFT and IDFT consume the same type of computational algorithm. For simplification, the variable W_N is frequently known as the "N^{th} root of unity with its exponent evaluated modulo N".

$$W_N^{\phi} = e^{\frac{-j2\pi\phi}{N}} = 1. \quad (3)$$

Here the difference between the (1) and (2) is the type of coefficients applied to it [14]-[15]. Most fast algorithms share the same general strategy, i.e mapping the one-dimensional transform form into a two or multi-dimensional representation, then exploiting the equivalence property of its coefficients to simplify the computation. Unlike conventional step-by-step decomposition of the twiddle factors, cascading the twiddle factor decomposition method is adopted here. Due to this new forms of FFT with high spatial regularity can be derived. For N-point FFT computation with Radix-2^i algorithm consists of i stages. The Radix-2^i algorithm has the same butterfly structure regardless of i value. However, the twiddle factor multiplication structure is different by factor i [16]-[17].

2.1 Radix-2^2 DIF FFT

After simplification [18]-[20], we have a set of DFTs of length N/4.

$$X(k_1+2k_2+4k_3)=\sum_{n_3=0}^{\left(\frac{N}{4}\right)-1}\left[BU_{\frac{N}{4}}^{k_1k_2}(n_3)W_N^{n_3(k_1+2k_2)}\right]W_{\frac{N}{4}}^{n_3k_3} \quad (4)$$

Where the first butterfly structure can be written as

$$\therefore BU_{\frac{N}{2}}^{k_1}\left(\frac{N}{4}n_2+n_3\right)=\begin{bmatrix} x\left(\frac{N}{4}n_2+n_3\right)+ \\ x\left(\frac{N}{2}+\frac{N}{4}n_2+n_3\right)(-1)^{k_1}\end{bmatrix} \quad (5)$$

Where the secondary butterfly structure can be expressed as

$$\therefore BU_{\frac{N}{4}}^{k_1k_2}(n_3)=BU_{\frac{N}{2}}^{k_1}(n_3)+(-j)^{k_1}(-1)^{k_2}BU_{\frac{N}{2}}^{k_1}\left(\frac{N}{4}+n_3\right) \quad (6)$$

Equation (5) and Eq. (6) represent the first two columns of butterflies with only trivial multiplications in the signal flow graph (SFG) of the Radix-2^2 algorithm. After first two columns, full multiplications are required to apply the decomposed twiddle factor in Eq. (4). The complete Radix-2^2 FFT Algorithm is obtained by applying this cascade decomposition recursively to the remaining DFT's of length N/4 in Eq. (4). We know that (–j) has the twiddle factor in between the first two columns. It can be implemented by real and imaginary part swamping and also a sign inversion.

2.2 Radix-2^3 DIF FFT

The common factor algorithm using 4-Dimensional Linear Index Map [18]-[20] takes the form of

$$X(k_1+2k_2+4k_3+8k_4)=\sum_{n_4=0}^{\left(\frac{N}{8}\right)-1}\begin{bmatrix} BU_{\frac{N}{8}}^{k_1k_2k_3}(n_4) \\ W_N^{n_4(k_1+2k_2+4k_3)}\end{bmatrix}W_{\frac{N}{8}}^{n_4k_4} \quad (7)$$

Where a third butterfly structure has the expression of

$$\therefore BU_{\frac{N}{8}}^{k_1k_2k_3}(n_4)=\begin{bmatrix} BU_{\frac{N}{4}}^{k_1k_2}(n_4)+BU_{\frac{N}{4}}^{k_1k_2}\left(\frac{N}{8}+n_4\right) \\ \left((0.7071(1-j))^{k_1}(-j)^{k_2}(-1)^{k_3}\right)\end{bmatrix} \quad (8)$$

The third butterfly contains a special twiddle factor

$$\therefore W_N^{\frac{N}{8}(k_1+2k_2+4k_3)}=\left((0.7071(1-j))^{k_1}(-j)^{k_2}(-1)^{k_3}\right) \quad (9)$$

Similarly, Radix-2^3 DIF-FFT can be represented in Eq. (7), corresponding butterfly structures represented in Eq. (8). Equation (9) represents the butterflies have twiddle factor $(0.7071(1-j))^{k_1}$ and $(-j)^{k_2}$ [18]-[20]. The Radix-2^3 algorithm has the same computational complexity as the Split Radix algorithm, yet with a much spatially regular SFG.

2.3 Radix-2^4 DIF FFT

The common factor algorithm using 5-Dimensional Linear Index Map takes the form of, We have a set of 16 DFTs of length N/16.

$$X(k_1+2k_2+4k_3+8k_4+16k_5)= \sum_{n_5=0}^{\left(\frac{N}{16}\right)-1}\left[BU_{\frac{N}{16}}^{k_1k_2k_3k_4}(n_5)W_N^{n_5(k_1+2k_2+4k_3+8k_4)}\right]W_{\frac{N}{16}}^{n_5k_5} \quad (10)$$

Where a fourth butterfly structure has the expression of

$$\therefore BU_{\frac{N}{16}}^{k_1k_2k_3k_4}(n_5)=\begin{bmatrix} BU_{\frac{N}{8}}^{k_1k_2k_3}(n_5)+ \\ BU_{\frac{N}{8}}^{k_1k_2k_3}\left(\frac{N}{16}+n_5\right)(-1)^{k_4}\end{bmatrix}W_{16}^{(k_1+2k_2+4k_3)} \quad (11)$$

The fourth butterfly contains a special twiddle factor

$$\therefore W_N^{\frac{N}{16}(k_1+2k_2+4k_3+8k_4)}=(-1)^{k_4}W_{16}^{(k_1+2k_2+4k_3)} \quad (12)$$

Similarly, Radix-2^4 DIF-FFT can be represented in equation (10) and corresponding butterfly structures represented in equation (11). The above twiddle factor can be divided $W_{16}^0, W_{16}^1, W_{16}^2$ and W_{16}^3 [18]-[20].

3. Rotators

Generally rotation is a circular movement with respect to the origin. The angle of rotations depends on the complex numbers. The rotation is calculated by, multiplying the input by the rotation coefficient. FFT has the symmetry property in their angles, by using the property for an N point FFT, only the M = (N/8) +1 angles in the range of (0, π/4). The remaining rotations of the FFT are obtained from the above angle by interchanging the real and imaginary parts and their signs.

The rotators are broadly classified as general rotators and constant rotators. General rotators are used to perform rotation by any angle, which is provided as input to the rotator. General rotators are implemented by a complex multiplier or by the Coordinate Rotation Digital Computer (CORDIC) algorithm. The complex multipliers can be implemented by four real multipliers and two adders. The sine and cosine components of complex multipliers are stored in a memory. While

storing in a memory, the number of bits used to store the complex numbers decides the quantization error and memory requirement. Increase in the coefficients word length it leads to reduction in error but it increases the memory requirement. In other case, for shorter word length both the memory size and accuracy will be reduced. The CORDIC algorithm works based on breaking down the rotation angle into series of micro rotations by specific angles. By using shift and addition operations with a minimum amount of hardware the micro rotations can be done [21]-[22].

Table 1 Number of rotations required for the FFT with different N values

Points in FFT	No. of bits	Rotation Angles					Rotation Error ε
		0	π/16	π/8	3π/16	π/4	
8	4	7	-	-	-	5+j5	0.00505
	5	14	-	-	-	10+j10	0.00505
	6	17	-	-	-	12+j12	0.000867
	7	41	-	-	-	29+j29	0.000149
	8	99	-	-	-	70+j70	0.0000255
16	4	7	-	7+j3	-	5+j5	0.0430
	5	13	-	12+j5	-	9+j9	0.0107
	6	31	-	29+j12	-	22+j22	0.00617
	7	55	-	51+j21	-	39+j39	0.00218
	8	120	-	111+j46	-	85+j85	0.000867
32	4	7	7+j1	7+j3	6+j4	5+j5	0.0574
	5	11	11+j2	10+j4	9+j6	8+j8	0.0261
	6	31	31+j6	29+j12	26+j17	22+j22	0.0116
	7	47	46+j9	43+j18	39+j26	33+j33	0.00496
	8	117	115+j23	108+j45	97+j65	83+j83	0.00279

Constant rotators are used to find out the rotations for specific angles like twiddle factors in FFT algorithms. Major elements in constant rotators are coefficient selection and the shift and add implementations. The coefficients are obtained by rounding the sine and cosine components of the angle. The following table.1 rotations for different number of points, different word lengths and the minimum rotation error value is indicated in the table. From fig.1, we clearly understood the relationship between rotation error, coefficients word length and number of points in FFT. Constant rotators are further divided into Single Constant Rotator (SCR) and Multiple Constant Rotator (MCR). Rotation by a single angle referred as single constant rotator. Similarly rotation by a multiple angles referred as multiple constant rotators [21]-[22].

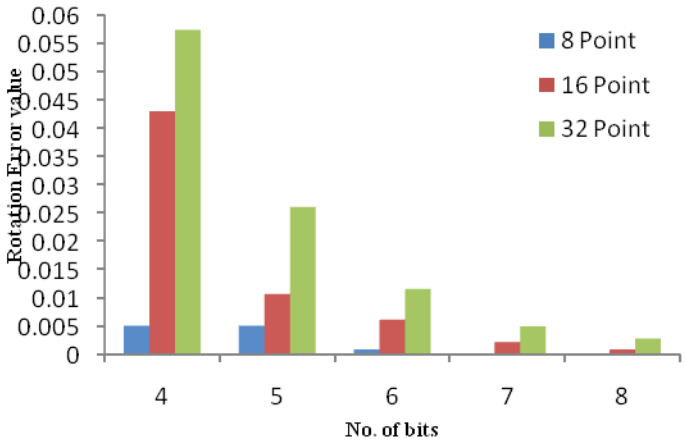

Fig.1: Rotation error as a function of the coefficient word lengths and number of points in FFT

4. Rotation Allocation

Generally M-rotators are a rotator that rotates any number of angles in M different symmetric angle sets. From the literature survey, based on the symmetric angle set the twiddle factor W_8 is called 2-rot, W_{16} is represented as 3-rot and W_{32} is represented as 5-rot. Hardware cost of rotators can be estimated in terms of adders. CCSSI [23] approach is used to obtain the equivalent adders required for M-rotators and CORDIC-II [24] algorithm is used to calculate the equivalent adders required for general rotators.

During the designing process of FFT architecture, the designer should follow the following properties. The properties are derived from the signal flow graph (SFG) of the radix-2^2 algorithm. The index of the data is represented as $I = b_{n-1}\ldots b_2, b_1, b_0$. In that the first property represents, 'n' is the number of FFT stages and's' is the stage which is going to be design. The

number of stages 'n' can be calculated by using the formula $n = \log_2 N$, where N is the number of points in the FFT. For any stage's' the index input to the butterfly differ by b_{n-s} position only. For example a 16 point FFT has 4 stages and the respective index of data is represented as $I = b_3b_2b_1b_0$. Consider the second stage of the FFT in the I = 1000 and I' = 1100 are processed by the butterfly. The indices are differ only in the position of b_2 ($b_{n-s} = b_{4-2} = b_2$). In the table 2.b the indices matrices show the stage 2 is differ only in the position b_2. The table 2.c shows the rotation matrix for each stage.

Table 2.a Index of the data

Stage - 1

b2	b3	b1	b0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Stage - 2

b3	b2	b1	b0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Stage - 3

b1	b2	b3	b0
0	0	0	0
0	0	0	1
1	0	0	0
1	0	0	1
0	1	0	0
0	1	0	1
1	1	0	0
1	1	0	1
0	0	1	0
0	0	1	1
1	0	1	0
1	0	1	1
0	1	1	0
0	1	1	1
1	1	1	0
1	1	1	1

Stage - 4

b1	b0	b3	b2
0	0	0	0
0	1	0	0
1	0	0	0
1	1	0	0
0	0	0	1
0	1	0	1
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	0
1	0	1	0
1	1	1	0
0	0	1	1
0	1	1	1
1	0	1	1
1	1	1	1

The next property related to the rotations at the FFT stages. For each stage the input is rotated by a factor according to the equation (3). In that the ϕ value depends on the stage index I and stage's' value. From table.3 for an odd stages the index value satisfied $b_{n-s}.b_{n-s-1} = 1$ the condition, it should be rotated by a trivial rotations. Similarly for even stages those index values satisfies the $b_{n-s+1} + b_{n-s} = 1$ condition, it should be rotated by non-trivial rotations. The angles which are multiples of $\pi/4$ considered as rotation by W_8 and the angles which are multiples of $\pi/8$ considered as rotation by W_{16}. Using the above properties mentioned in the table.2 we get different possibilities in rotator values with respect to change the data order at different FFT stages. The proposed hardware architecture works based on reducing the number of rotators and their complexity by finding the efficient distribution of FFT rotations [25]-[26]. The table.4 and table.5 represents the properties of the radix-2^3 and radix-2^4 decimation in frequency algorithm respectively.

Table 2.b Indices matrix for each stage

INDICES

Stage - 1

3	2	1	0
11	10	9	8
7	6	5	4
15	14	13	12

INDICES

Stage -2

3	2	1	0
7	6	5	4
11	10	9	8
15	14	13	12

INDICES

Stage 1

9	8	1	0
11	10	3	2
13	12	5	4
15	14	7	6

INDICES

Stage 2

12	8	4	0
13	9	5	1
14	10	6	2
15	11	7	3

Table 2.c Rotation matrix for each stage

ROTATIONS

0	0	0	0
0	0	0	0
0	0	0	0
4	4	4	4

ROTATIONS

0	0	0	0
0	2	4	6
0	1	2	3
0	3	6	9

ROTATIONS

0	0	0	0
0	0	0	0
0	0	0	0
4	4	4	4

Table 3 Properties of the Radix-2^2 Decimation In Frequency (DIF) algorithm

Properties	**Radix-2^2**
Butterfly Unit (BU)	b_{n-s}
Trivial Rotations (For odd stages)	$b_{n-s} . b_{n-s-1} = 1$
Non-Trivial Rotations (For even	$b_{n-s+1} + b_{n-s} =$

Table 4 Properties of the Radix-2^3 Decimation In Frequency (DIF) Algorithm

Properties		**Radix-2^3**	**M-rotator**
Butterfly Unit (BU)	$\forall s$	b_{n-s}	-
Trivial Rotations	s = 3i+2	$b_{n-s} . b_{n-s-1} = 1$	-
Non-Trivial Rotations (For even stages)	s = 3i+3	$b_{n-s+2} + b_{n-s+1} + b_{n-s} = 1$	-
	s = 3i+1	$b_{n-s}.(b_{n-s+1} + b_{n-s-2}) = 1$	2-rotator

Table 5 Properties of the Radix-2^4 Decimation In Frequency (DIF) algorithm

Properties		**Radix-2^4**	**M-rotator**
Butterfly Unit (BU)	$\forall s$	b_{n-s}	-
Trivial Rotations (For odd stages)	s = 4i+1	$b_{n-s} . b_{n-s-1} = 1$	-
	s = 4i+3	$b_{n-s} . b_{n-s-1} = 1$	-
Non-Trivial Rotations (For even stages)	s = 4i+4	$b_{n-s+3} + b_{n-s+2} + b_{n-s+1} + b_{n-s} = 1$	-
	s = 4i+2	$(b_{n-s+1} + b_{n-s}) .(b_{n-s-1} + b_{n-s-2}) = 1$	3-rotator

5. Proposed FeedForward Architectures

The total area of the processor is measured with the help of total number of twiddle factors, complex address and complex data memory requirement. Simultaneously, the performances of the proposed architectures are measured in terms of number of clock cycles required by the architecture to finish the process and number of samples processed by the architecture per clock cycle. The proposed model worked based on find an efficient method for distribution of FFT rotations. Rotation allocation approach is discussed in this paper. Due to that the number of rotations and their complexity is reduced.

Rotator allocation is done by rearranging the matrices of indexes and matrices of rotations. Complexities of rotators are reduced only the matrices of rotations have fewer rotations. When rearrange the matrices into more number of rows are equal to zero, the resultant structure has fewer rotator than other. The rotator allocation is distributing the bits $b_3b_2b_1b_0$ of the index *I* into serial and parallel bits. The combination of serial bits or parallel bits reflects on the complexity of the rotator. The table.2.a shows the 16 point FFT with 4-parallel line processor has 2 serial bits and 2 parallel bit dimensions. For example 16 point FFT has four stages in that each stage has six different possible alternatives. For example a N point FFT with P-parallel line processor (N=32, P=4 ;) has 3 serial bits and 2 parallel bits dimensions. The number of possible alternatives for each stage is ten [25]-[26].

For serial dimensions $n - p = \log_2 (N) - \log_2 (P)$; $n - p = 3$.

For parallel dimensions $p = \log_2 (P)$; =2.

5.1 Radix- 2^k 2-parallel pipelined architecture using feedforward MDC

In radix-2^k, all the approaches are almost have same twiddle factor values. But some of the rotators available in radix 2^3 can be adjusted to perform the entire operation with the help of W_8 and W_{16}. From the analysis, the Radix -2^4 has less number of address and reduced memory requirement, even other methods having the same twiddle factor values.

Generally the proposed architecture is derived from the base of radix-2 butterflies, non trivial rotator, trivial rotators and multiplexers. In 2-parallel architecture processes 2 samples in parallel and three distinct approaches are discussed like Radix-2^2, Radix -2^3 and Radix- 2^4. The table.6 compares the rotators required for 2- Parallel pipelined architectures different points FFT in Radix-2^2, Radix- 2^3 and Radix -2^4. The following Fig. 2, Fig. 3 and Fig. 4 show the 64 point pipelined 2 parallel Radix-2^2, Radix -2^3 and Radix- 2^4 Feedforward MDC architectures respectively. In Fig. 2 the rotations by W_8 is represented with the help of square box. From the analysis in 2-parallel, radix-2^2 and radix-2^3 has the same number of rotations, butterflies and total memory compared with radix-2 feedforward FFT [2]. Compare the Fig. 2, Fig. 3 and Fig. 4, we get both the 2 parallel Radix-2^2 and Radix -2^4architecture has the same structure except the rotator W_{16} is replaced for every four stages in radix-2^2 structures.

In feedforward architecture the number of complex adder is calculated by using the formula [27]-[28]

$$P.\log_2 N \tag{13}$$

General rotator can be calculated by using the formula,

$$P.\left(\frac{\log_2 N}{k}-1\right), \text{ if } \quad P < 2 \tag{14}$$

$$\frac{2^k-1}{2^k}.P.\left(\frac{\log_2 N}{k}-1\right), \text{ if } \quad P \geq 2k \tag{15}$$

Table.6 N-point radix-2^k 2- Parallel Pipelined Architecture Using Feedforward Multipath Delay Commutator

No. of Points	Rotators								
	Total			General			W_8 (or) W_{16}		
	2^2	2^3	2^4	2^2	2^3	2^4	2^2	2^3	2^4
64	4	4	4	4	2	1	0	2	3
128	5	5	5	5	3	1	0	2	4
256	6	6	6	6	3	2	0	3	4
512	7	7	7	7	4	3	0	3	5
1024	8	8	8	8	5	3	0	3	5
2048	9	9	9	9	5	4	0	4	5
4096	10	10	10	10	6	4	0	4	6

From Fig.2, Fig.3 and Fig.4 in the proposed architecture the required number of butterfly units depends on the number of samples in parallel. Generally the butterfly does not vary with respect to stages but the rotators depend on the stage. Rotators are classified into trivial, non-trivial and rotations by W_8 (or) W_{16}. Compared radix- 2^2 FFT architecture with the proposed radix- 2^4 Feedforward architecture requires the same number of rotators, area and memory. The Radix-2^3 and Radix -2^4 FFT architecture can be simplified into use only W_8 or W_{16} rotators. The proposed 2 parallel radix-2^4 Feedforward MDC architecture saves 50% of the adders and reduces the memory requirement compared with the existing structures [29]. But it keeps the same number of rotator values.

5.2 Radix- 2^k 4-parallel pipelined architecture using feedforward MDC

The 4- Parallel pipelined architectures using Feedforward Multi-path Delay Commutator structure for the computation of a N point FFT like Radix-2^2, Radix- 2^3 and Radix -2^4 is compared in table. 7. From the table.7, Radix-2^2 architecture requires only trivial multiplication. Simultaneously it has a very good architecture style compared to other methods. In 4 parallel architectures, Radix-2^3 and Radix-2^4 is compared with 4 parallel radix-4 feedforward FFT, it requires fewer rotators than other approach [30]-[31] but it has the equal amount of adders and memory. The Radix- 2^4 Feedforward architecture reduces 50% of address and 25% of twiddle factor W_{16} than existed Radix- 2^4 feedback architectures [32]-[33].

The following Fig. 5, Fig. 6 and Fig. 7 show the 4-parallel architecture, three distinct approaches are discussed like Radix-2^2, Radix -2^3 and Radix- 2^4. As we discussed earlier, due to Feedforward architecture the required number of rotators in proposed design is reduced when compared with existing feedback architectures. A 4 parallel Radix-2^2 architecture saves up to 25% of total number of rotators. Simultaneously, 4 parallel Radix-2^4 Feedforward architecture saves up to 50% of total number of adders and 25% of W_{16} rotators compared with existing Radix-2^4 feedback

architecture [32]-[33].

Table.7 N-point radix-2^k 4- Parallel Pipelined Architecture using Feedforward Multipath Delay Commutator

No. of Points	Rotators								
	Total			General			W_8 (or) W_{16}		
	2^2	2^3	2^4	2^2	2^3	2^4	2^2	2^3	2^4
64	6	8	7	6	4	2	0	4	5
128	8	10	8	8	5	3	0	5	5
256	9	12	10	9	6	4	0	6	6
512	11	14	12	11	8	5	0	6	7
1024	12	16	14	12	9	6	0	7	8
2048	14	18	15	14	10	7	0	8	8
4096	1	2	17	15	1	8	0	8	9

5.3 Radix- 2^k 8-parallel pipelined architecture using feedforward MDC

In 8 parallel architecture, Radix-2^2, Radix -2^3 and Radix- 2^4 has very good performance than any other proposed methods. Especially the Radix- 2^4 feed forward architectures has less number of W_{16} twiddle factors and 50% less in adders compared with the well known Radix- 2^4 feedback architectures [34]. From the existing research work and table.6, table.7 and table.8 the numbers of non-trivial multiplications are increased with respect to the number of points in the FFT processor.

Similarly, the required numbers of complex twiddle factors are reduced when the radix values rise from Radix-2, Radix-2^2, Radix -2^3 and Radix- 2^4. The following Fig. 8, Fig. 9 and Fig. 10 show the 8-parallel architecture, three distinct approaches are discussed like Radix-2^2, Radix -2^3 and Radix- 2^4. From the table.8, the required number of general multiplications are has the same value compared to the existing architectures [37] but the number of twiddle factor W_{16} is save up to 12% . Due to this the total area required by the processor is reduced. The Feedforward structure is more efficient than feedback structures in the applications like more samples to be processed in parallel manner, which has power of two. The required numbers of parallel samples are chosen based on the required throughout value depending on the applications. From the Fig.11 and Fig.12 we understand that for any FFT size, the proposed feed forward architecture has the smallest area.

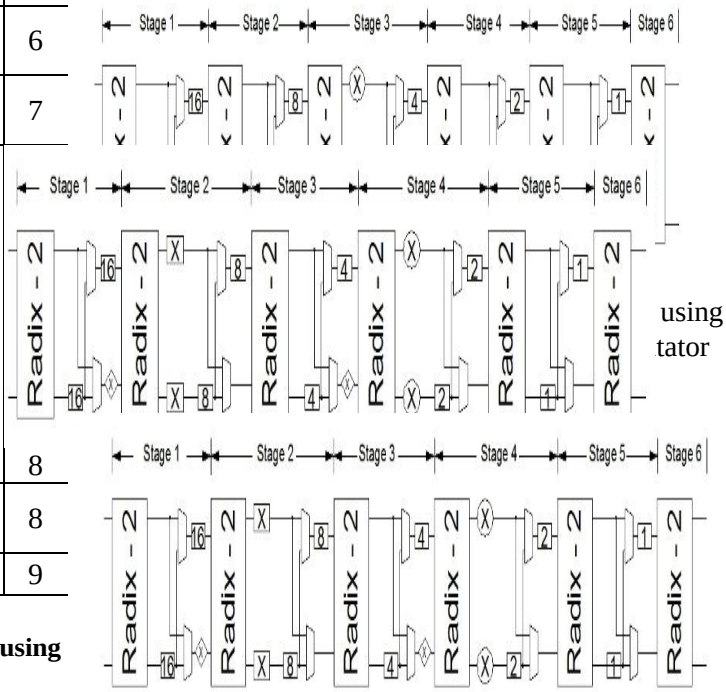

Fig.3 64-point 2- Parallel Pipelined Architecture using Radix-2^3 Feedforward Multipath Delay Commutator

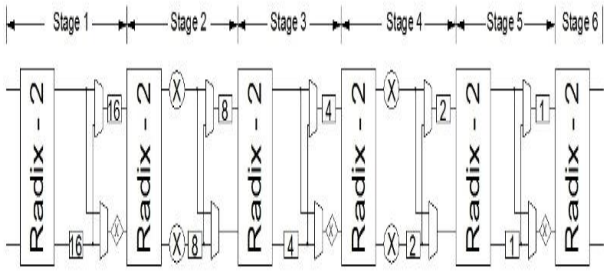

Fig.4 64-point 2- Parallel Pipelined Architecture using Radix-2^4 Feedforward Multipath Delay Commutator

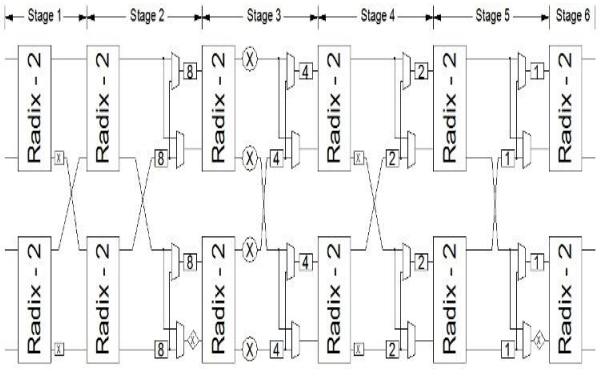

Fig.5 64-point 4- Parallel Pipelined Architecture using Radix-2^2 Feedforward Multipath Delay Commutator

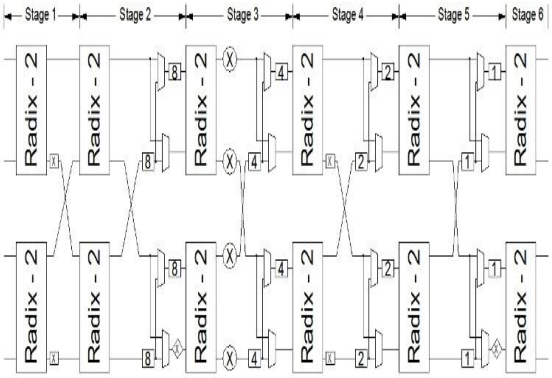

Fig.6 64-point 4- Parallel Pipelined Architecture using Radix-2^3 Feedforward Multipath Delay Commutator

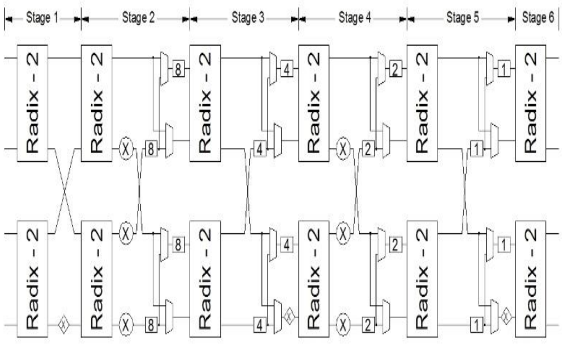

Fig.7 64-point 4- Parallel Pipelined Architecture using Radix-2^4 Feedforward Multipath Delay Commutator

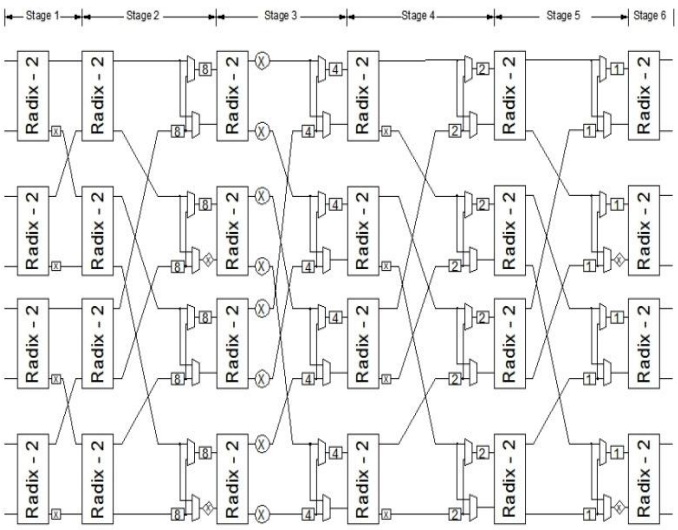

Fig.8 64-point 8- Parallel Pipelined Architecture using Radix-2^2 Feedforward Multipath Delay Commutator

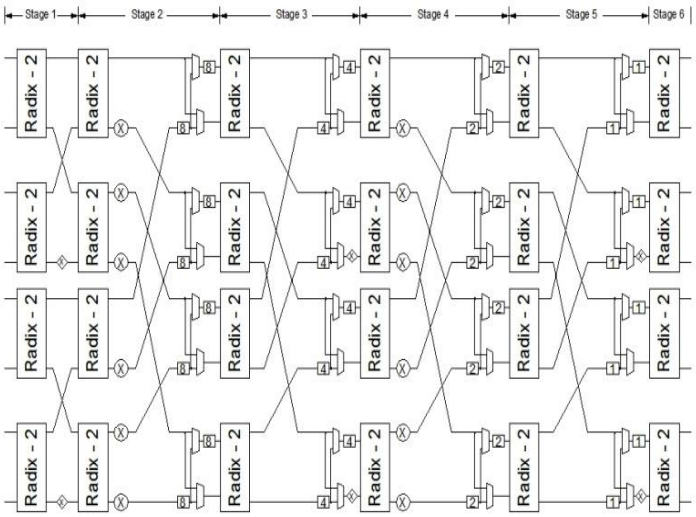

Fig.9 64-point 8- Parallel Pipelined Architecture using Radix-2^3 Feedforward Multipath Delay Commutator

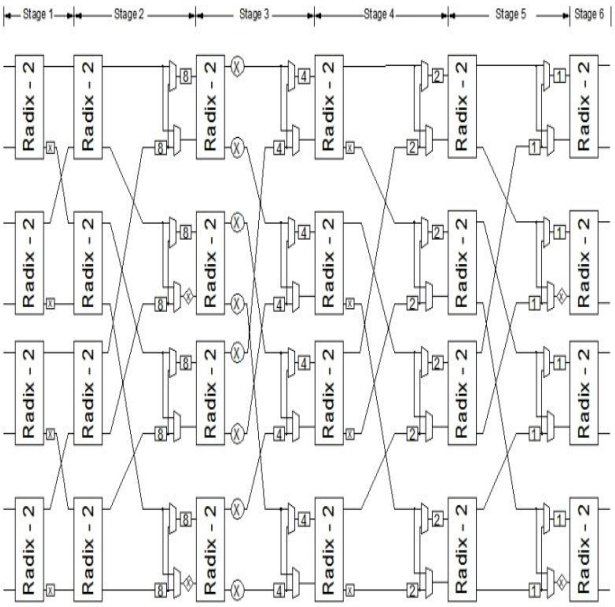

Fig.10 64-point 8- Parallel Pipelined Architecture using Radix-2^4 Feedforward Multipath Delay Commutator

Table.8 N-point radix-2^k 8- Parallel Pipelined Architecture using Feedforward Multipath Delay Commutator

No. of Points	Rotators								
	Total			General			W_8 (or) W_{16}		
	2^2	2^3	2^4	2^2	2^3	2^4	2^2	2^3	2^4
64	12	11	13	12	7	4	0	4	9
128	15	14	17	15	9	6	0	5	11
256	18	17	20	18	11	8	0	6	12
512	21	20	24	21	14	10	0	6	14
1024	24	23	27	24	16	12	0	7	15
2048	27	26	31	27	18	14	0	8	17
4096	30	29	34	30	21	16	0	8	18

6. Implementation of Proposed System

In proposed architecture the throughput is proportional to the number of samples in parallel. The latency is equal to the size of the FFT divided by the number of parallel samples. The suitable architecture has been selected based on the throughput and latency that depends upon the application requires. The memory size does not vary with respect to the number of parallel samples. In few applications input samples and output frequencies are required in normal order for that reordering circuits are required before and after the FFT. For input and output reordering, in our proposed design requires N-P amount of total memory is required for P-parallel N-point FFT. For radix-2^3 and radix 2^4 the non-trivial rotations are W_8 and W_{16} respectively. The number of non-trivial rotations $W_{L,}$ dependent on radix-2^k of the algorithm ($L=2^k$). If the k values are increased it also increases the number of angles of the kernel. Finally, it leads to implementations of non-trivial rotations are very difficult. In our proposed model we discussed only about the DIF structure. But from the research we know that both DIT and DIF requires architecture uses the same amount of hardware components.

From the table.9 for an 8-parallel MDC FFT architecture has only general rotators and 1-rotators. But remaining architectures have 3- rotators and 2-rotators in their design except radix-2^3. Compared to radix-2^3 our proposed architecture has fewer amounts of general rotators with the increases in 1-rotators.

Table.9 Twiddle factors in the 4- parallel and 8-parallel MDC-FFT Architecture for different Radix-2 algorithm

	Radix	Rotators				Equivalent adders
		General	3- Rotators	2- Rotators	1- Rotators	
4-Parallel MDC FFT Architecture	2	14	1	2	2	476
	2^3	12	0	0	4	442
	2^4	8	0	3	6	412
	2^6	8	2	4	4	438
8-Parallel MDC FFT Architecture	2	26	1	4	6	886
	2^3	24	0	0	6	876
	2^4	16	0	0	15	776
	2^6	12	2	8	12	782

Our proposed 8-parallel MDC FFT architecture can be divided in to upper part 4-parallel MDC FFT architecture and lower part 4-parallel MDC FFT architecture. Samples that available in odd clock cycles can be processed by lower part 4-parallel MDC FFT architecture. Similarly, samples that available in even

clock cycles can be processed by upper part 4-parallel MDC FFT architecture.

The equivalent adders can be calculated by using the following formula,

$$Equival.adders = 2P.\log_2 N + Adders.in.Rotators + 2\frac{P}{4}\log_2\left(\right. \quad (16)$$

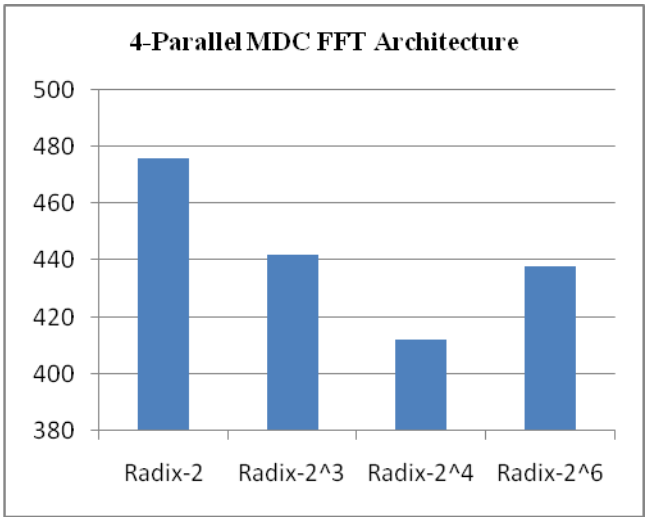

Fig.11 4096-point 4- Parallel Pipelined Architecture using Radix-2^k Feedforward Multipath Delay Commutator

The fig.11 and fig.12 shows the equivalent adders required for proposed 4096-point 4-parallel and 8-parallel pipelined architecture using Radix-2^k feedforward multipath delay commutator. From the figures we know that the proposed 4096-point 8-parallel and 4-parallel pipelined architecture using Radix-2^4 feedforward multipath delay commutator is better than other architectures for VDSL applications. In a 4 - parallel architecture the reduction in area is improved from 17% to 23% compared with [32]-[33]. For an 8 parallel architecture the reduction in area is improved from 23% to 29% compared with the existing architectures [32]-[33]. These results show our proposed design reduces the area of the processor significantly.

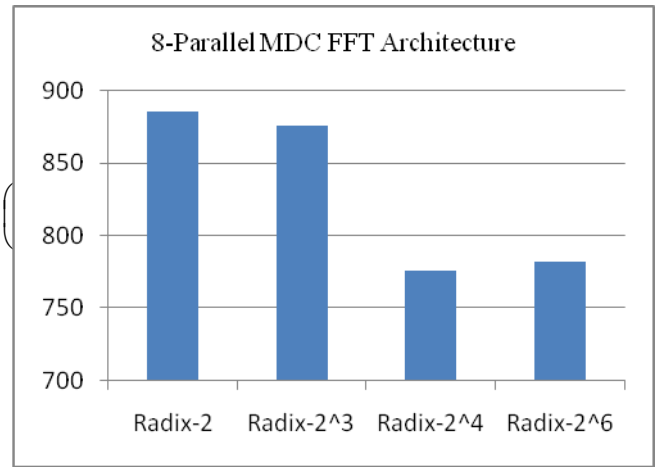

Fig.12 4096-point 8- Parallel Pipelined Architecture using Radix-2^k Feedforward Multipath Delay Commutator

7. Conclusion

From the literature review, none of the single FFT algorithm is suitable for different hardware platforms. For better result, the best algorithm should be selected based on the hardware and applications. The prime concerns are regularity in structure, low power consumption, less area and high performance. If the number of points in the FFT increased, the numbers of possible FFT computation algorithms are also increased. Finally, the different FFT algorithms are developed for achieving required goals in specific applications. Rotation allocation approach is discussed in this paper. The proposed model worked based on choosing an efficient method for distribution of FFT rotations. Due to that the number of rotations and their complexity is reduced. In the research work different radices are tried. From the observation the number of complex rotators decreases with increase in k value up to radix-2^4. From the analysis the proposed 4- parallel radix-2^4 MDC architecture savings the area of rotators increased from 17% to 23% and the proposed 8-parallel radix-2^4 MDC architecture save the area of rotators around 23% to 29% with respect to the existing architectures.

References

[1] Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for Digital Terrestrial Television, ETSI EN 300 744, 2004, ETSI, v.1.5.1.

[2] S.He and M.Torkelson, "Design and implementation of 1024-point pipeline FFT processor," in Proc. IEEE Custom Integr. Circuits

Syst.II, Exp. Briefs, vol.57, no.6, pp.451-455, jun 2010.

[3] Arun.C.A and P.Prakasam, "Design of high speed FFT algorithm for OFDM technique", in Proc.IEEE International Conference on Emerging Devices and Smart Systems (ICEDSS)., pp:66-71, March 2016.

[4] Gokan Polat, Sitki Ozturk, and Mehmet Yakut, "Design and Implementation of 256-Point Radix-4 100 Gbit/s FFT Algorithm into FPGA for High-Speed Applications," ETRI Journal, vol.37, no.4, Aug. 2015, pp. 667-676.

[5] Hun-Sik Kang, Sun Hyok Chang, In-Ki Hwang and Joon-Ki Lee, "A design and implementation of 32-paths parallel 256-point FFT/IFFT for optical OFDM systems," in Proc. IEEE International Conference on Advanced Communication Technology (ICACT).,pp: 417 - 421, Feb 2016.

[6] Chao Wang, Yuwei Yan and Xiaoyu Fu, "A High-Throughput Low-Complexity Radix- 24 - 22 - 23 FFT/IFFT Processor with Parallel and Normal Input / Output Order for IEEE 802.11ad Systems," IEEE Trans. on VLSI Systems. vol. 23, issue: 11. pp. 2728–2732, 2015.

[7] Jun-Feng Tang, Xiao-Jin Li, Gang Zhang and Zong-Sheng Lai, "Design of high-throughput mixed-radix MDF FFT processor for IEEE 802.11.3c," in Proc. IEEE. International Conference on Solid-State and Integrated Circuit Technology (ICSSICT), Pages: 1 - 3, 2012.

[8] Chu Yu, "A 128/512/1024/2048-point pipeline FFT/IFFT architecture for mobile WiMAX, "in Proc. IEEE. International Conference on Global Conference on Consumer Electronics (GCCE), pp: 243–244, 2013.

[9] Tram Thi Bao Nguyen and Hanho Lee," Shared CSD complex constant multiplier for parallel FFT processors, "in Proc. IEEE. International SoC Design Conference (ISOCC), pp: 27 – 28, 2015.

[10] Seong-Weon Ko, Jee-Hoon Kim, Jae-Seon Yoon and Hyoung-Kyu Song, "Cooperative OFDM system for high throughput in wireless personal area networks" , IEEE Trans. Consumer Electronics, vol. 56, issue: 2 pp. 458 - 462, June 2010.

[11] R. Simon Sherratt and Oswaldo Cadenas, "A double data rate architecture for OFDM based wireless consumer devices," IEEE Trans.Consumer Electronics, vol. 56, issue: 1 pp. 23 - 26, Jan.2010.

[12] S.He and M.Torkelson,"Designing pipelining FFT processor for OFDM (de)Modulation", Proc. IEEE URSI Int. Symp. Sig. Syst. Electron., pp.257-262, 1998.

[13] E. H. Wold and A. M. Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementations," IEEE Trans. Comput., no. 5, pp. 414–426, May 1984.

[14] J. G. Proakis and D. G. Manolakis, 1996. Digital Signal Processing Principles, Algorithm and Applications. Prentice Hall, pp: 449-495.

[15] C.S.Burrus, "Efficient Fourier transform and convolution Algorithms", in:J.S. Lim and A.V Oppenhiem, eds,. Advanced Topics in Digital Signal Processing, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[16] M.T.Heidmen, D.H.Johnson and C.S.Burrus,"Gauss and the history of the FFT", IEEE.Acoust.Speech Signal Process. Magazine, vol.1, No.4, October 1984, pp.14-21.

[17] Duhamel, P., and M.Vetterli, "Fast Fourier transforms: a tutorial review and a state of the art",Signal Processing, vol.19,pp.259–299,1990.

[18] Arun.C.A and P.Prakasam, "A mathematical approach on various radix-2i FFT algorithms", in Proc.IEEE International Conference on Electrical, Electronics and Optimization Techniques (ICEEOT)., pp:1040-1045, March 2016.

[19] C. S. Burrus. Index mapping for multidimensional formulation of the DIF and convolution. IEEE Trans. Acoust.,Speech, Signal Processing, ASSP- 25(3):239-242, June 1977.

[20] James C.Schatzman," Index Mapping for the Fast Fourier Transform", IEEE.trans.Signal Processing,Vol 44.No.3,pp.717-719.march 1996.

[21] M. Garrido and J. Grajal, "Efficient memoryless CORDIC for FFT computation," in Proc. IEEE Int. Conf. Acoust. Speech Signal Process.,Apr. 2007, vol. 2, pp. 113–116.

[22] J. E. Volder, "The CORDIC trigonometric computing technique," IRE Trans. Electron. Comput., vol. EC-8, pp. 330–334, Sep. 1959.

[23] M. Garrido, F. Qureshi, and O. Gustafsson, "Low-complexity multiplierless constant rotators based on combined coefficient selection and shift-and-add implementation (CCSSI)," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 61, no. 7, pp. 2002–2012, Jul. 2014.

[24] M. Garrido, P. Källström, M. Kumm, and O. Gustafsson, "CORDIC II: A new improved CORDIC algorithm," IEEE Trans. Circuits Syst.

II, Exp. Briefs, vol. 63, no. 2, pp. 186–190, Feb. 2016.

[25] M. Garrido, O. Gustafsson, and J. Grajal, “Accurate rotations based on coefficient scaling,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 58, no. 10, pp. 662–666, Oct. 2011.

[26] M.Garrido, J.Grajal, M.A. Sanchez, and O. Gustafsson, “Pipelined radix-2^k feedfoeward FFT architectures,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.21, no.1, pp.23-32, Jan.2013.

[27] M. Garrido, “A New representation of FFT algorithms using Triangular Matrices,” IEEE Trans. Circuits and Syst. I, vol. 63, no.10, pp.1737-1745, 2016.

[28] M. Garrido; S. J. Huang; S. G. Chen, "Feedforward FFT Hardware Architectures Based on Rotator Allocation," in IEEE Transactions on Circuits and Systems I: Regular Papers, 2017.

[29] J. Lee, H. Lee, S. in Cho, and S.-S. Choi, “A high-speed, low-complexity radix-24 FFT processor for MB-OFDM UWB systems,” in Proc. IEEE Int. Symp. Circuits Syst., 2006, pp. 210–213.

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

[31] J. H. McClellan and R. J. Purdy, Applications of Digital Signal Processing. Prentice-Hall, 1978, ch. 5, Applications of Digital Signal Processing to Radar.

[32] H. Liu and H. Lee, “A high performance four-parallel 128/64-point radix-2 4 FFT/IFFT processor for MIMO-OFDM systems,” in Proc. IEEE Asia Pacific Conf. Circuits Syst., 2008, pp. 834–837.

[33] S.-I. Cho, K.-M. Kang and S.-S. Choi, “Implemention of 128-point fast Fourier transform processor for UWB systems,” in Proc. Int. Wireless Comm. Mobile Comp. Conf., 2008, pp. 210–213.

[34] 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. I, vol. 57, no. 6, pp. 451–455, Jun. 2010.

[35] M. A. Sanchez, M. Garrido, M. L. L ´ opez, and J. Grajal, “Implementing ´ FFT-based digital channelized receivers on FPGA platforms,” IEEE Trans. Aerosp. Electron. Syst., vol. 44, no. 4, pp. 1567–1585, Oct. 2008.