High Speed Pipelined 64-Point FFT Processor based on Radix-2^2 for Wireless LAN

Manish Bansal
Department of EC Engineering
MANIT
Bhopal, India
manishbansal2008@gmail.com

Sangeeta Nakhate
Department of EC Engineering
MANIT
Bhopal, India
Sanmanit@gmail.com

***Abstract* — This Paper presents high Speed pipeline 64-point FFT processor based on Radix-2^2 for wireless LAN communication systems. This method uses Radix-2 butterfly structure and Radix-2^2 CFA algorithm. Radix-2 butterfly's complexity is very low and Radix-2^2 CFA algorithm reduces number of twiddle factors compared to Radix-4 and Radix-2. An efficient VHDL code has been written, synthesized successfully using XST of Xilinx ISE 14.1 and simulated using ModelSim PE Student Edition 10.4a. Also MATLAB code has been written and simulated with MATLAB R2012a tool. The computation speed of proposed design is observed to be 158.96 MHz after the synthesis process and SQNR 37.02dB for 64 point.**

***Keywords* — FFT, CFA, Radix-2^2, complex multiplier, SDF, WLAN.**

I. Introduction

Fast Fourier transform (FFT) class of algorithms is widely used in communication and digital signal processing. The FFT algorithm is considered one of the basic algorithms in many DSP projects. Nowadays, FFT is the key building block for the mobile communications especially for the orthogonal frequency division multiplexing (OFDM) transceiver systems. OFDM (Orthogonal Frequency Division Multiplexing) is an effective multicarrier technology for robust, reliable high-rate and high-speed data transmission in the wire/wireless communication systems such as LAN, wireless local area network. Most modern WLANs are based on IEEE 802.11 standards [16]. Implementation of FFT architectures for fast and efficient computational schemes has attracted many researchers [12]. Some of the approach to design FFT is memory based, pipeline based and general purpose DSP. Memory based is most area efficient but it needs many computation cycles. Pipeline based architecture possesses regularity, modularity, local connection and high throughput rate with a lower clock frequency.

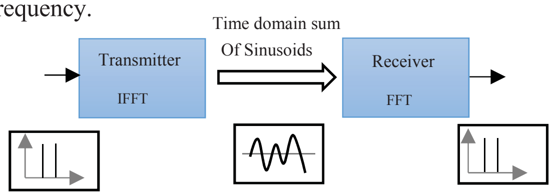

Fig. 1. Simplified WLAN - OFDM System Block Diagram.

All hardware implementations of FFT can be categorized into three kinds of pipelined architectures which include multiple delay commutator (MDC), single delay commutator (SDC) and single delay feedback (SDF) architectures. Among three pipelined architectures the SDF architecture is more suitable as-

(1) The SDF architecture is very convenient to implement the different length FFT.

(2) The number of the required registers in SDF architecture is smaller than that in MDC and SDC structures [13].

The controller of proposed pipelined SDF architecture is easier than the other structures as overall architecture and all components are controlled by count signal only.

This paper presents the implementation of Radix-2^2 single-path delay feedback pipelined FFT processor for 64 point which can be used for WLAN application. The Radix influences FFT architecture. A small Radix means simple butterfly structure, increases number of twiddle factors and higher Radix means complex butterfly structure, reduces number of twiddle factors. In the proposed FFT Radix-2^2 architecture, Radix-2 butterfly is used which is simplest butterfly structure and Radix-2^2 uses less number of twiddle factors as Radix-4.

This paper is structured as follows – Section II describes the proposed FFT processor design based on Radix–2^2 CFA. Section III describes novel process of proposed 64 point FFT processor. Section IV describes the comparison. At last conclusion in Section V.

II. Proposed FFT processor design based on Radix-2^2 CFA

Proposed processor is designed using Radix-2^2 common factor algorithm and Radix-2 butterfly structure which can be used for performing for 64-point FFT computation. Proposed processor is designed in SDF (Single delay feedback) pipeline architecture which is one the most efficient architecture and reduce complexity of FFT processor. Each stage consist of Radix-2 butterfly structure. In this section, Raidx-2^2 common factor algorithm, Proposed FFT processor and required component are described for 64-point in this paper.

The definition of Discrete Fourier Transform (DFT) of size N is defined as [13]:

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

$$0 \le k \le N-1$$

Where $W_N^{nk} = e^{\frac{-j2\pi nk}{N}}$ is a twiddle factor and represents the Nth root with its exponent evaluated modulo N. n is time index and k is frequency index. The Radix-2^2 algorithm is formulated using 3-dimensional liner index mapping and common factor algorithm. The Radix-2^2 algorithm for 64 point is expressed as follows –

Applying divide and conquer 3-D Liner Index Mapping –

$$n = 32n_1 + 16n_2 + n_3 \quad \ldots\ldots\ldots\ldots (2)$$

$$k = k_1 + 2k_2 + 4k_3 \quad \ldots\ldots\ldots\ldots (3)$$

Where

$$n_1, n_2 = 0,1 \quad and \quad n_3 = 0,1,\ldots\ldots,15$$

$$k_1, k_2 = 0,1 \quad and \quad k_3 = 0,1,\ldots\ldots,15$$

The common factor algorithm (CFA) form [13] –

$X\,(k_1 + 2\,k_2 + 4\,k_3)$

$$= \sum_{n_3=0}^{15}\sum_{n_2=0}^{1}\sum_{n_1=0}^{1} x(32n_1 + 16n_2 + n_3)\,.\,W_N^{nk}$$

$$= \sum_{n_3=0}^{15}\sum_{n_2=0}^{1}\sum_{n_1=0}^{1} x(32n_1 + 16n_2 + n_3)\,.\, W_N^{(32n_1+16\,n_2+n_3)(k_1+\,2k_2+\,4k_3)} \quad \ldots\ldots\ldots (4)$$

The twiddle factor can be expressed as –

$$W_N^{nk} = W_N^{(32n_1+16\,n_2+n_3)(k_1+\,2k_2+\,4k_3)}$$

$$= \underbrace{(-1)^{n_1k_1}}_{\text{BU}}\ \underbrace{(-j)^{n_2k_1}}_{\text{PE}}\ \underbrace{\overset{n_2k_2}{(-1)^{n_1k_2}}}_{\text{BU}}\ \underbrace{W_{64}^{n_3(k_1+2k_2)}}_{\text{PE}}\ W_{16}^{n_3k_3} \quad \ldots\ldots\ldots (5)$$

Where n_1, n_2, n_3 are the index terms of the input sample n and k_1, k_2, k_3 are the index terms of the output sample k. $(-1)^{n_1k_1}$ and $(-1)^{n_1k_2}$ are butterfly elements. $(-j)^{n_2k_1}$ and $W_{64}^{n_3(k_1+2k_2)}$ are processing elements.

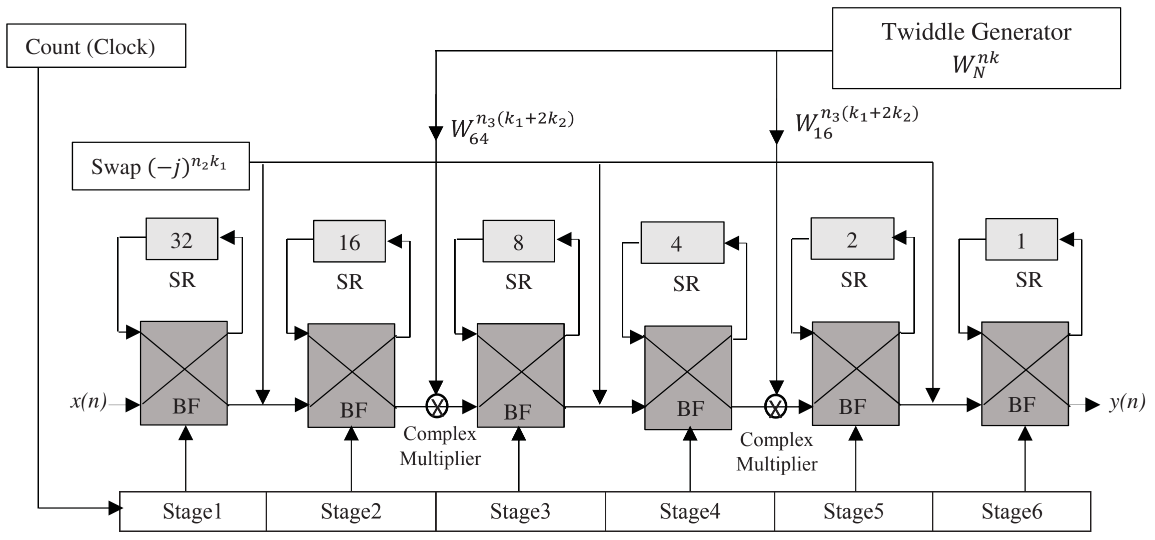

Fig. 2. Proposed Radix- 2^2 FFT Processor for 64-Point

The detailed structure of proposed Radix-2^2 FFT Processor for 64 Point is shown in figure 2 which retain 6 number of stages. SR indicates shift register which store values. In case of 64 point 1st shift register store 32 values, 2nd shift register store 16 values, 3rd shift register store 8 values, 4th shift register store 4 values, 5th shift register store 2 values and last 6th shift register store 1 value. Twiddle generator generates twiddle factor as per $W_{64}^{n_3(k_1+2k_2)}$ based on n_3, k_1 and k_2 values. Swap (-j) perform real sign inversion then swap signed inverted real value and imaginary value. Butterfly (BF) perform addition, subtraction and by-pass operations. Complex multiplier multiply input complex values with twiddle factors values.

Figure 3 show the butterfly (BU) structure which perform operation between nth value and $(n + N/2)$th value On first N/2 cycles, the Mux1, 2, 3, 4 in the butterfly module switch to position "0". The input data from left is directed to the shift registers until they are filled. On next N/2 cycles, the multiplexers turn to position '1' then butterfly start addition / subtraction operation to compute 2-point DFT with incoming data and the data stored in the shift registers [6]. The subtraction results are stored in same register, first half no. of addition results are stored in next stage register and second half no. of addition results start addition/subtraction with stored first half no. of addition in the same stage. Thus this process run from current stage to next stage till last stage and butterfly element is connected in pipelined structure in order to compute a result every clock cycle.

Figure 4. Trivial (-j) multiplier involves real imaginary swapping and real sign inversion. This real imaginary swapping is controlled by multiplexor control signal 'S' and real sign inversion is done by 2nd compliment. Wherever – j

multiplication is required as per signal flow graph, Mux1 & Mux2 multiplexors switches to position '1' then first sign of real value is inverted then inverted real value is swapped to imaginary and imaginary value swapped to real. In this way sign inversion and swapping are done instead of multiplication in case of –j multiplication which increase the performance and reduce the hardware logic.

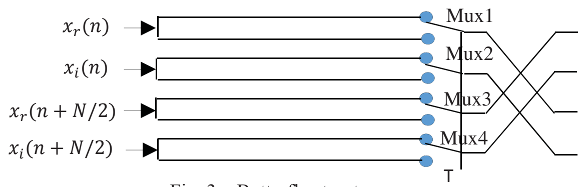

Fig. 3. Butterfly structure

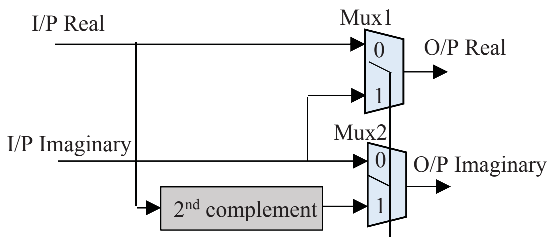

Fig. 4. Trivial (-j) multipler (swap)

III. NOVEL PROCESS OF PROPOSED 64 POINT FFT PROCESSOR

Proposed FFT processor is designed for 64-Point FFT. As shown in figure 2, all components - BFs, stages, SRs, twiddle generator and swap are configured by count control signal based on value of n_1, n_2, n_3, k_1, k_2, k_3 and all operations are controlled by count signal. Here proposed FFT processor is designed which takes 32% less twiddle factors than conventional FFT processor [1].

Proposed FFT generates 6 stages, 6 shift registers and 2 twiddle generator. First shift register store 32 values, second shift register store 16 values, third shift register store for 8 values, fourth shift register store 4 value, fifth shift register store 2 value and sixth shift register store 1. Twiddle $W_N^{n_3(k_1+2k_2)}$ generate twiddle factors $W_{64}^{n_3(k_1+2k_2)}$ and $W_{16}^{n_3(k_1+2k_2)}$. First stage swap$(-j)^{n_2k_1}$and second stage twiddle factors $W_{64}^{n_3(k_1+2k_2)}$ calculations are performed as per 64 point samples, third stage swap$(-j)^{n_2k_1}$ and fourth stage twiddle factor $W_{16}^{n_3(k_1+2k_2)}$calculation are performed as per 16 points samples, fifth stage $(-j)^{n_2k_1}$ and sixth stage calculation are performed as per 4 points samples as mentioned in Table I.

Figure 5 shows signal flow graph of proposed FFT 64-point Radix-2^2 FFT processor for 64 point. In the 64 point FFT computation, first 32 point is stored in stage 1 register using butterfly structure in each clock. It takes 32 clock and at 33th clock as x(32) point inputted into stage 1, butterfly start addition and subtraction between x(0) & x(32) point and this addition is stored in stage 2 register and subtraction is stored in stage 1 register. At 34th clock as x(34) point inputted into stage 1, butterfly again start addition and subtraction between x(1) & x(33) point and this addition is stored in stage 2 register and subtraction is stored in stage 1 register. This process run up to 48th clock. At 49th clock as x(48) point inputted into stage 1 butterfly again start addition and subtraction between x(16) & x(48) point and this addition is stored in stage 2 register and subtraction is stored in stage 1 at this clock stage 2 buttery also start addition and subtraction and store addition in next stage 3 register and subtraction in stage 2 register. This process continuously run up to 63th clock and at 64th clock as x(63) point inputted in to stage 1 butterfly again start addition and subtraction between x(31) & x(63) point and this addition is stored in stage 2 register and subtraction is stored in stage 1 and at this clock, stage 2, stage 3, stage 4, stage 5 and stage 6 buttery also start addition and subtraction and store addition in next stage 3, stage 4, stage 5, stage 6 registers and subtraction in stage 2, stage 3, stage 4, stage 5, stage 6 register. Stage 6 output is final FFT computation values.

Processing elements swapping and twiddle factors value find out based on n_1, n_2, n_3, k_1, k_2, k_3 value as per shown in table I and calculations of processing elements at each point happen as shown in figure 5. The value of processing elements, operation between butterfly and processing elements at each position are controlled by clock control signal

TABLE I. FIND PROCESSING ELEMENTS VALUE AT EACH POSITION FOR FFT PROCESSOR BASED ON RADIX-2^2 CFA

PE / N	**1st Stage** $(-j)^{n_2k_1}$	**2nd Stage** $W_N^{n_3(k_1+2k_2)}$	**3rd Stage** $(-j)^{n_2k_1}$	**4th Stage** $W_N^{n_3(k_1+2k_2)}$	**5th Stage** $(-j)^{n_2k_1}$	**6th Stage** $W_N^{n_3(k_1+2k_2)}$
64	$n_1, n_2, k_1, k_2 = 0, 1$ $n_3, k_3 = 0, 1, 2, .., 15$ & N=64		$n_1, n_2, k_1, k_2 = 0, 1$ $n_3, k_3 = 0, 1, 2, 3$ & N=16 and repeating for every next 16 points		$n_1, n_2, k_1, k_2 = 0, 1$ $n_3, k_3 = 0$ & N=4 and repeating for every next 4 points	

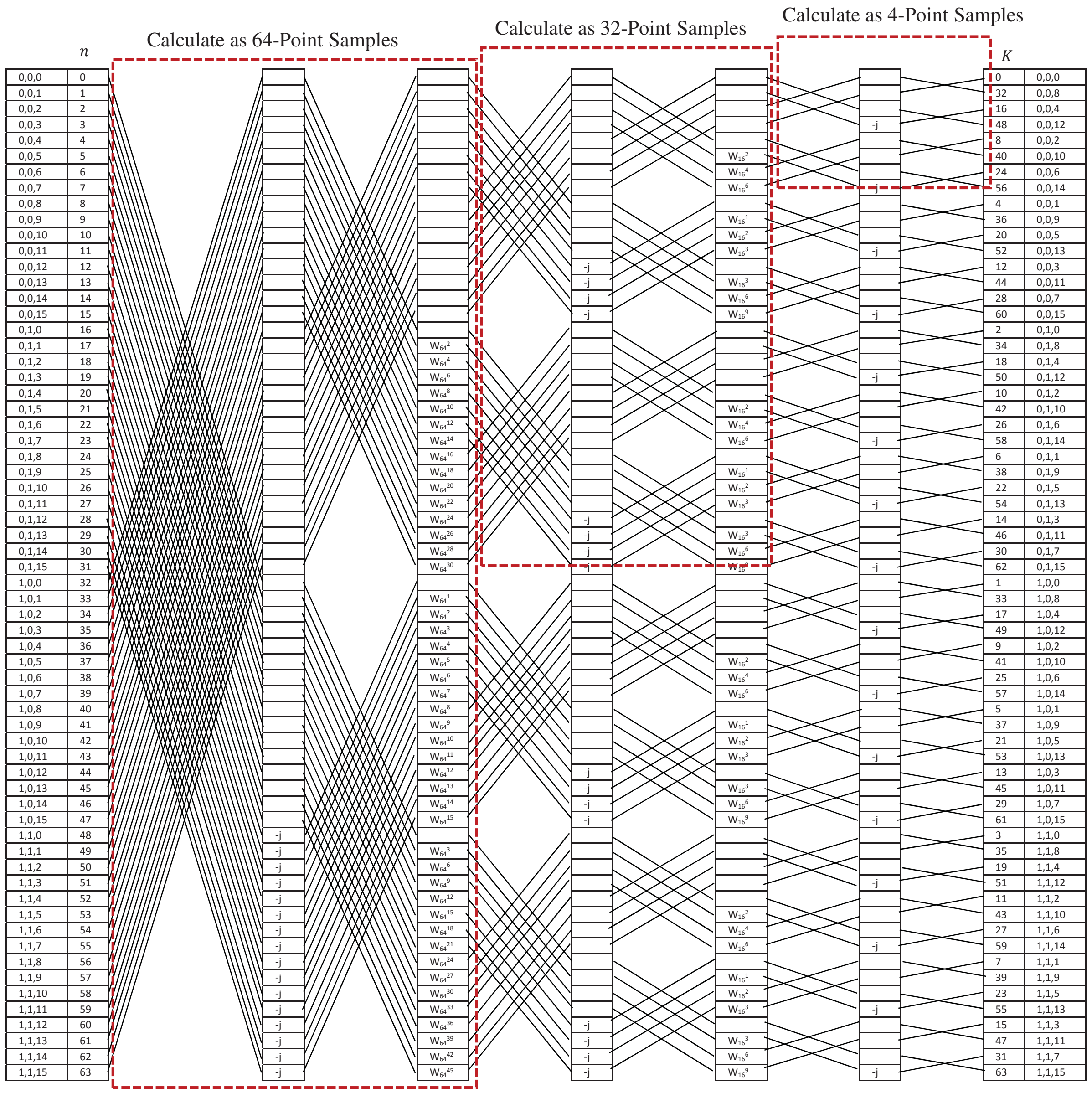
Calculate as 64-Point Samples
Calculate as 32-Point Samples
Calculate as 4-Point Samples
n
K
-j
W64
W16

Fig. 5. Single Flow Graph using for 64 point FFT processor using Radix- 2^2 CFA

IV. Comparison

The architecture of proposed Radix-2^2 FFT processor is designed in VHDL. It synthesized with xc7vx330t-3ffg1157 device and simulated in ModelSim & MATLAB to verify its results.

Table II and figure 6 show comparison of proposed FFT and conventional FFT processors [1]. The results shows that proposed architecture used 32% number of twiddle multiplier which causes less number of twiddle factor and use less area.

Table III shows implementation parameters of proposed architecture and conventional FFT processor [1] between slices used, max clock frequency, minimum period and combinational path delay. Figure 6 represents the bar graph between proposed and FPGA [1] architecture.

Figure 7 shows the RTL view of proposed FFT which is generated by view RTL schematic feature of Xilinx 14.1 synthesis-XST tool.

TABLE II. COMPARISON OF FFT ARCHIRECTURES

Architecture	No. of Point	Method	Radix	No. of stages	No. of twiddles	Memory
R2^2SDF[1]	64	Pipeline	Radix -2^2	6	128	63
Proposed FFT	64	Pipeline	Radix -2^2	6	90	63

TABLE III. IMPLEMENTATION PARAMETERS

Parameters	FPGA[1]	Proposed FFT
No. of Point (N)	64	64
No. of slices used	624	725
No. of slices Flip Flops used	-	538
No. of 4 input LUTs used	-	1345
No. of GCLK used	-	6
Max Frequency (in MHz)	95.2	158.96
SQNR	-	37.02
Minimum period in ns	6.670	6.291
Maximum combinational path delay in ns	2.655	0.971

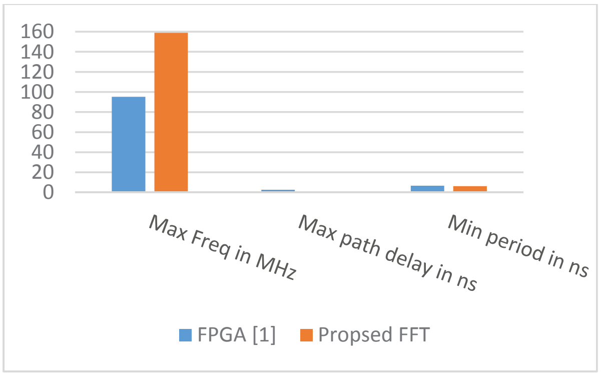

Fig. 6. Comparison of proposed and conventional FFT

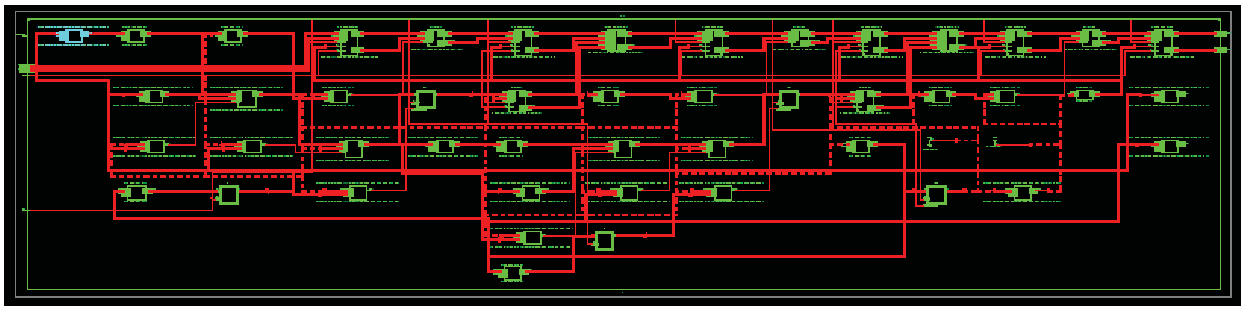

Fig. 7. RTL view of Propoesed FFT Radix-2^2 for 64 point

V. CONCLUSIONS

The aim of this research is to design and implement 64 point FFT processor based on Radix-2^2 for WLAN application and use 32 % less no. of twiddle factors then conventional FFT which reduce the memory & area and reduce the complexity using Radix-2 butterfly structure which overall results increases speed as decrease number of multiplications. Figure 6 bar graph shows that maximum frequency of proposed architecture is higher than conventional architecture.

REFERENCES

[1] Nisha John and Prof. Sadanandan G.K, “FPGA Implementation of a Novel Efficient Vedic FFT/IFFT Processor For OFDM,” International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 2014, pp 7965-7972.

[2] Taesang Cho and Hanho Lee, “A High-Speed Low-Complexity Modified FFT Radix - 2^5 Processor for High Rate WPAN Applications,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2013, pp 187-191.

[3] V.Sarada and T.Vigneswaran, “Reconfigurable FFT Processor - A Broader Perspective Survey,” International Journal of Engineering & Technology, April 2013, pp 949.

[4] K.Harikrishna, T. Rama Rao and Vladimir A. Labay, “FPGA Implementation of FFT Algorithm for IEEE 802.16e (Mobile WiMAX),” International Journal of Computer Theory and Engineering, April 2011, pp 197-203.

[5] T. Cho, H. Lee, J. Park and C. Park, “A high-speed low-complexity modified Radix - 2^5 FFT processor for gigabit WPAN applications,” IEEE International Symposium of Circuits and Systems (ISCAS), 2011, pp 1259–1262.

[6] Wei-Hsin Chang and Truong Nguyen, “An OFDM-specified lossless FFT architecture,” IEEE Transactions on Circuits and Systems I, 2006, pp. 1235-1243.

[7] Akshata. U and Gopika. D.K, “Hardware Implementation of Decimation in Time FFT,” MIT International Journal of Electronics and Communication Engineering, 2013, pp. 39-42.

[8] Wei-Hsin Chang and Truong Nguyen, “Design and simulation of 64 point FFT using Radix - 4 algorithm for FPGA Implementation,” International Journal of Engineering Trends and Technology, 2013, pp 109-113

[9] Jeesung Lee and Hanho LEE, “A High-Speed Parallel Radix-2^4 FFT/IFFT Processor for MB OFDM UWB System,” IEEE International Symposium on Circuits and Systems, 2008, pp 960 – 963.

[10] Minhyeok Shin and Hanho Lee, “A High-Speed Four-Parallel Radix-2^4 FFT/IFFT Processor for UWB Applications,” IEEE International Symposium on Circuits and Systems, 2008, pp 960-963.

[11] Mario Garrido, J. Grajal, M.A. S´anchez and Oscar Gustafsson, “Pipelined Radix-2^k Feedforward FFT Architectures,” IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS, 2013, pp 1-10.

[12] Zeke Wang, Xue Liu, Bingsheng He, and Feng Yu, “A Combined SDC-SDF Architecture for Normal I/O Pipelined Radix-2 FFT,” IEEE Transactions on very large scale integration (VLSI) systems, 2015, pp 973-977.

[13] Jiang Wang and Leif Arne Ronningen, “An Implementation of Pipelined Radix-4 FFT Architecture on FPGAs,” Journal of Clean Energy Technologies, Vol. 2, No. 1, January 2014, pp 101-103.

[14] Chih-Peng Fan1, Mau-Shih Lee and Guo-An Su, “Efficient low multiplier cost 256-point FFT design with Radix-2^4 SDF architecture,” Journal of Engineering, National Chung Hsing university, Vol. 19,No.2, 2008, pp 61-74

[15] Shen-Jui Huang and Sau-Gee Chen, “A green FFT processor with 2.5-GS/s for IEEE 802.15.3c (WPANs),” International Conference on Green Circuits and Systems (ICGCS), 2010, pp 9–13.

[16] Ahmed Saeed, M. Elbably, G. Abdelfadeel and M. I. Eladawy, “FPGA implementation of Radix-2^2 Pipelined FFT Processor,” Proceedings WAV'09 Proceedings of the 3rd WSEAS international symposium on Wavelets theory and applications in applied mathematics, signal processing & modern science, 2009, pp 109-114.

[17] Weidong Li, Yutai Ma and Lars Wanhammar, “Word Length Estimation for Memory Efficient Pipeline FFT/IFFT Processors,” Conference on Signal Processing Applications & Technology (ICSPAT), 1999

[18] Xiaoxin Cui and Dunshan Yu, “Digital OFDM transmitter architecture and FPGA design,” 2009 IEEE 8th International Conference on ASIC, 2009, pp 477-480.

[19] S. Salivahanan and A. Vallavaraj, “Digital Signal Processing” (Tata McGraw-Hill Education, 2001).