VHDL implementation of an optimized 8-point FFT/IFFT processor in pipeline architecture for OFDM systems

Mounir Arioua, Said Belkouch, Mohamed Agdad
Embedded Systems and Digital Controls Laboratory,
Dept. of Electrical Engineering, ENSAM
Cadi Ayyad University
Marrakech, Morocco
m.arioua@ieee.org, belkouch@ensa.ac.ma,
mohamed_agdad@hotmail.fr

Moha M'rabet Hassani
Electronic and Instrumentation Laboratory,
Dept. of Physics, Faculty of Science Semlalia
Cadi Ayyad University
Marrakech, Morocco
hassani@ucam.ac.ma

Abstract—**The Fast Fourier Transform (FFT) and its inverse transform (IFFT) processor are key components in many communication systems. An optimized implementation of the 8-point FFT processor with radix-2 algorithm in R2MDC architecture is presented in this paper. The butterfly- Processing Element (PE) used in the 8-FFT processor reduces the multiplicative complexity by using a real constant multiplication in one method and eliminates the multiplicative complexity by using add and shift operations in other proposed method. The pipeline architecture R2MDC has been implemented with the 8-point module and simulation results show that this module significantly achieves a better performance with lower resource usage.**

Keywords- Cooley-Tukey, FFT/IFFT, OFDM, R2MDC,VHDL.

I. INTRODUCTION

The FFT/IFFT is one of the most commonly used digital signal processing algorithm. Recently, FFT processor has been widely used in digital signal processing field applied for communication systems. FFT/IFFT processors are key components for an Orthogonal Frequency Division Multiplexing (OFDM) based wireless broadband communication system; it is one of the most complex and intensive computation module of various wireless standards PHY layer (OFDM-802.11a, MIMO-OFDM 802.11n…) [1]. However, the main constraints nowadays for FFT processors used in wireless communication systems are execution time and lower power consumption [2]. The main issue in FFT/IFFT processors is complex multiplication, which is the most prominent arithmetic operation used in FFT/IFFT blocks. It is an expensive operation and consumes a large chip area and power especially when it comes to a large FFT point [3]. To reduce the complexity of the multiplication, one of the two proposed methods in this article replaces the expensive complex multiplication with real and constant multiplications [4]. The other method optimizes further the processing, by wiping out the non-trivial complex multiplication with the twiddle factors and fulfills the processing with no complex multiplication.

We applied both methods to a simple 8-point FFT and we compared them to the conventional FFT and to the R2MDC processor in order to a comparative evaluation.

The paper is organized as follows: Section II discusses the FFT algorithm implementation (Cooley-Tukey) and complex multiplication used inside the butterfly-processing element. Section III devoted for an architectural description of the FFT used module. Section IV shows the resulting implementation and finally a conclusion is given in section V.

II. FFT ALGORITHM

In this section, a brief overview of IFFT and FFT algorithms is provided to be effectively used in OFDM applications.

The N-point discrete Fast Fourier Transform (DFT) is defined as:

$$X[k] = \sum_{n=0}^{N-1} x[n].W_N^{\ nk}$$

Where $$W_N^{\ nk} = e^{-j\frac{2\pi nk}{N}} \qquad 0 \le k \le N-1$$

X(k) is the k-th harmonic and x(n) is the n-th input sample. Direct DFT calculation requires a computational complexity of O(N^2). By using The Cooley–Tukey FFT algorithm, the complexity can be reduced to O (N.$\log_r$ N) [2] [5].

A. The Cooley-Tukey FFT Algorithm

The Cooley-Tukey FFT is the most universal of all FFT algorithms, because of any factorization of N is possible [5] [6]. The most popular Cooley-Tukey FFTs are those were the transform length is a power of a basis r, i.e., N = r^S. These algorithms are referred to as radix-r algorithms. The most commonly used are those of basis r = 2 and r = 4.

For r = 2 and S stages, for instance, the following index mapping of the The Cooley–Tukey algorithm gives:

$$n = 2^{S-1}n_1 + 2^{S-2}n_2 + \dots + 2n_{S-1} + n_S$$

$$k = 2^{S-1}k_S + 2^{S-2}k_{S-1} + \dots + 2k_2 + k_1$$

And: $\quad n_1, n_2, \dots, n_{S-1}, n_S = 0,1$

$$k_S, k_{S-1}, \dots, k_2, k_1 = 0,1$$

The Cooley–Tukey algorithm is based on a divide-and-conquer approach in the frequency domain and therefore is referred to as decimation-in-frequency (DIF) FFT. The DFT formula is split into two summations:

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(n)W_N^{nk} + \sum_{n=\frac{N}{2}}^{N-1} x(n)W_N^{nk} = \sum_{n=0}^{\frac{N}{2}-1} x(n)W_N^{nk} + \sum_{n=0}^{\frac{N}{2}-1} x(n+\frac{N}{2})W_N^{(n+\frac{N}{2})k}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x(n)W_N^{nk} + \sum_{n=0}^{\frac{N}{2}-1} x(n+\frac{N}{2})W_N^{nk}.W_N^{\frac{N}{2}.k} \quad \text{and} \quad W_N^{\frac{N}{2}.k} = (-1)^k$$

$$= \sum_{n=0}^{\frac{N}{2}-1} \left(x(n) + (-1)^k.x(n+\frac{N}{2}) \right).W_N^{nk} \tag{1}$$

X(k) can be decimated into even-and odd-indexed frequency samples:

$$X(2k) = \sum_{n=0}^{\frac{N}{2}-1} \left(x(n)+(n+\frac{N}{2}) \right).W_N^{2.n.k} = \sum_{n=0}^{\frac{N}{2}-1} \left(x(n)+(n+\frac{N}{2}) \right).W_{\frac{N}{2}}^{n.k} \tag{2}$$

$$X(2k+1) = \sum_{n=0}^{\frac{N}{2}-1} \left(x(n)-(n+\frac{N}{2}) \right).W_N^{2.n.k} = \sum_{n=0}^{\frac{N}{2}-1} \left(x(n)-(n+\frac{N}{2}) \right).W_{\frac{N}{2}}^{n.k} \tag{3}$$

The computational procedure can be repeated through decimation of the N/2-point DFTs X(2k) and DFTs X(2K+1). The entire algorithm involves $\log_2 N$ stages, where each stage involves N/2 operation units (butterflies). The computation of the N point DFT via the decimation-in-frequency FFT, as in the decimation-in-time algorithm requires $(N/2).\log_2 N$ complex multiplication and $N.\log_2 N$ complex addition [5].

B. 8-point FFT Module

The flow graph of complete DIF decomposition of 8-point DFT computation is represented in Fig. 1. The basic operation in the signal flow graph is the butterfly operation; it's a 2-point DFT computation as shown in Fig. 2.

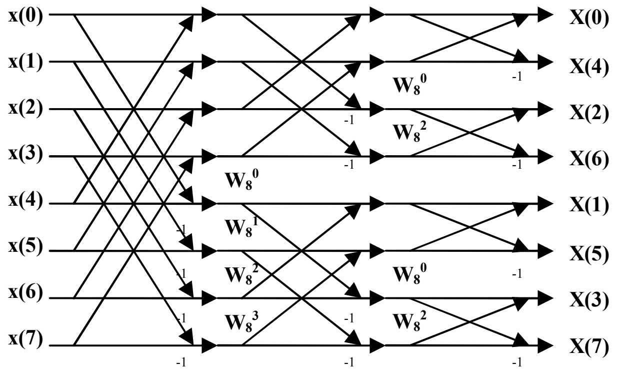

Figure 1. 8-point decimation-in-frequency FFT algorithm.

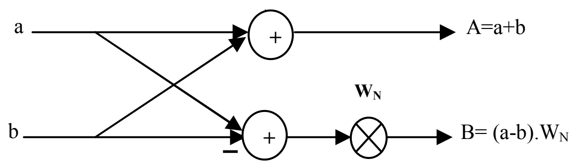

Figure 2. Basic Butterfly computation.

Radix-2 butterfly processor consists of a complex adder and complex subtraction. Beside that, an additional complex multiplier for the twiddle factors W_N is implemented. The complex multiplication with the twiddle factor requires four real multiplications and two add/subtract operations [1] [3] [7].

C. Complex Multiplication

Since complex multiplication is an expensive operation, we tend to reduce the multiplicative complexity of the twiddle factor inside the butterfly processor by calculating only three real multiplications and three add/subtract operations as in (5) and (6).

The twiddle factor multiplication:

$$R + jI = (X + jY).(C + jS) \tag{4}$$

However the complex multiplication can be simplified:

$$R = (C - S).Y + Z \tag{5}$$

$$I = (C + S).X - Z \tag{6}$$

With: $\quad Z = C.(X - Y) \qquad (7)$

C and S are pre-computed and stored in a memory table. Therefore it is necessary to store the following three coefficients C, C + S, and C - S.

The implemented algorithm of complex multiplication used in this work uses three multiplications, one addition and two subtractions as shown in Fig. 3. This is done at the cost of an additional third memory table which is given in (7). In the hardware description language (VHDL) program,□the twiddle factor multiplier was implemented using component instantiations of three lpm-mult and three lpm-add-sub modules from Altera library. Worth to note that lpm modules are supported by most of EDA vendors and LPM provides an architecture-independent library of logic functions or modules that are parameterized to achieve scalability and adaptability.

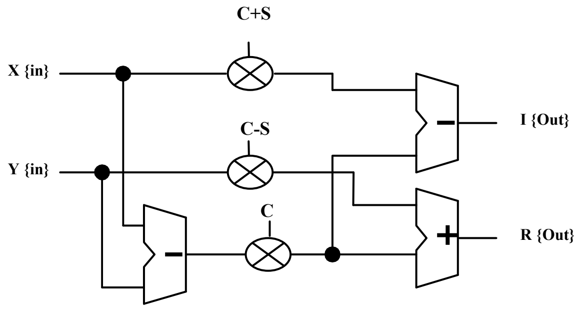

Figure 3. Implementation of complex multiplication.

In the 8-point FFT with radix-2 algorithm, the multiplication with W_8^2=-j and W_8^0 factors is trivial, multiplication with W_8^2 simply can be done by swapping from real to imaginary part and vice versa, followed by changing the sign [7] [8]. The number of complex multiplication in this scheme of FFT is two: W_8^1, W_8^3. This non-trivial complex multiplication was implemented with two multiplications for W_8^1 and three multiplications for W_8^3. Therefore, the number of real multiplications is 5. However, this solution was not suitable in practice; because the first stage has to process four different twiddle factors (trivial and non-trivial multiplications) in pipeline architecture with one complex multiplier. Therefore, it will require more elements in the structure to implement the two complex multiplications in addition to the two trivial multiplications in one block. Eventually, we used four complex multiplications with one complex multiplier in the first proposed architecture of the 8-point FFT processor as shown in TABLE I.

Another architecture solution was proposed in order to wipe out completely the complex multiplication inside the butterfly processor. The implementation complexity of non-trivial twiddle factors W_8^1, W_8^3 was replaced by basic operation of shift-and-add which is shown in Fig. 5.

III. PIPELINE 8-POINT FFT/IFFT PROCESSOR ARCHITECTURE

A. 8-point FFT Calculation Method

The application of the FFT algorithm for computation of the 8-point DFT required calculation of three of 2-point DFT (radix-2) [9].

n = 4n2 + 2n1 + n0	n2=0,1	n1=0,1	and	n0=0,1
k = 4k2 + 2k1 + n0	k2=0,1	k1=0,1	and	k0=0,1

The 8-point FFT can be expressed in the following sequence:

$$X(n2,n1,n0) = \sum_{k=0}^{7} x(k2,k1,k0) W^{(4n2+2n1+n0).(4k2+2k1+k0)} \quad (8)$$

Where:

$$W^{(4n2+2n1+n0).(4k2+2k1+k0)} =$$

$$W^{4n0k2}.W^{2n0k1}.W^{4n1k1}.W^{(2n1+n0)k0}.W^{4n2k0}$$

The process of computation of the 8-point FFT is done as follow:

First butterfly calculation:

$$x1(n0,k1,k0) = \sum_{k2=0}^{1} x(k2,k1,k0).W^{4n0k2} \quad (9)$$

Twiddle factor multiplication:

$$x'1(n0,k1,k0) = x1(n0,k1,k0)W^{2n0k1} \quad (10)$$

Second butterfly calculation:

$$x2(n0,n1,k0) = \sum_{k1=0}^{1} x'1(n0,k1,k0).W^{4n1k1} \quad (11)$$

Twiddle factor multiplication:

$$x'2(n0,n1,k0) = x2(n0,n1,k0)W^{(2n1+n0)k0} \quad (12)$$

Third butterfly calculation:

$$x3(n0,n1,n2) = \sum_{k1=0}^{1} x'2(n0,n1,k0).W^{4n2k0} \quad (13)$$

B. R2MDC Architecture

Simplicity, modularity, high throughput and real-time applications are required for FFT/IFFT processors in communication systems. The pipeline architecture is suitable for those ends [7] [9].

The sequential input stream in pipeline architecture unfortunately doesn't match the FFT/IFFT algorithm since the bloc FFT/IFFT requires temporal separation of data. In this case the data memory is required in the pipeline processor to be rearranged according to FFT/IFFT algorithm as shown in Fig.1.

One of the most straightforward approaches for pipeline implementation of radix-2 FFT algorithm is Radix-2 Multi-path Delay Commutator (R2MDC) architecture. It's the simplest way to rearrange data for the FFT/IFFT algorithm [2] [9]. The input data sequence are broken into two parallel data stream flowing forward, with correct distance between data elements entering the butterfly scheduled by proper delays. 8-point FFT in R2MDC architecture is shown in Fig. 4.

At each stage of this architecture half of the data flow is delayed via the memory (Reg) and processed with the second half data stream. The delay for each stage is 4, 2, and 1 respectively. The total number of delay elements is 4 + 2 + 2 + 1 +1 = 10.

In this R2MDC architecture, both Butterflies (BF) and multipliers are idle half the time waiting for the new inputs. The 8-point FFT/IFFT processor has one multiplier, 3 of radix-2 butterflies, 10 registers (R) (delay elements) and 2 switches (S).

IV. IMPLEMENTATION AND RESULTS

The 8-point FFT computation with radix-2 in R2MDC architecture was first coded in VHDL using Quartus software tool from Altera and then simulated and synthesized on Altera Cyclone 2 EP2C35F672C6 device. The purpose is to determine the resource usage of the two proposed designs. The first proposed structure combines two resource reductions, the complex multiplication reduction inside the butterfly, and the fact of using the pipeline architecture. The second structure attempts to eliminate the complex multiplication, hence avoid this expensive operation of multiplication and consumes less chip area.

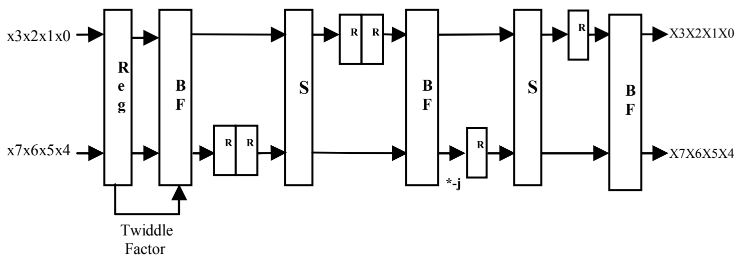

Figure 4. 8-point FFT implemented in R2MDC architecture.

TABLE I. COMPARISON OF THE FIRST PROPOSED ARCHITECTURE WITH COOLEY-TUKEY ALGORITHM AND CONVENTIONAL R2MDC ARCHITECTURE IN TERMS OF NUMBER OF ARITHMETIC OPERATIONS AND MULTIPLIERS.

Algorithm	**Complex Multiplications**	**Real Multiplications**	**Real add/ sub**	**Complex multipliers**	**Real multipliers**
Cooley-Tukey	12	48	72	12	48
R2MDC	12	48	72	3	12
1st Proposed architecture	4	12	60	1	3

In the first structure, the twiddle factors of first stage of the 8-point FFT are computed and stored in a register in order to synchronize the multiplication of those coefficients with the two radix-2 FFT points entered in the pipeline architecture as shown in Fig. 4.

From the algorithmic perspective, this proposed architecture demands less number of arithmetic operations compared to the Cooley-Tukey algorithm and the conventional R2MDC pipeline architecture. This comparison is shown in TABLE I.

The above comparison shows that the simple architecture of 8-point FFT proposed in this work requires 33% of non-trivial complex multiplication, and 25% of real multiplication, and 83% of number of addition/ subtraction when compared to the R2MDC approach. This shows an important reduction of resource usage, which leads to a high speed operation [10].

The number of Embedded 9-bit multiplier elements used in the proposed architecture was reduced from 12 elements to 3 by using the complex multiplication reduction in the FFT R2MDC pipelined architecture as shown in TABLE I. Hence, this solution is more efficient than the conventional R2MDC. However, the minimum size of data memory remains the same.

In order to further optimize the processor, another method proposed eliminates the non-trivial complex multiplication with the twiddle factors (W_8^1, W_8^3) and implements the processor without complex multiplication. The proposed butterfly processor performs the multiplication with the trivial factor W_8^2=-j by switching from real to imaginary part and imaginary to real part, with the factor W_8^0 by a simple cable. With the non-trivial factors $W_8^{\,1} = e^{-\frac{j\pi}{4}}$, $W_8^{\,3} = e^{-\frac{j3\pi}{4}}$, the processor realize the multiplication by the factor 1/√2 using hardwired shift-and-add operation as shown in Fig. 5.

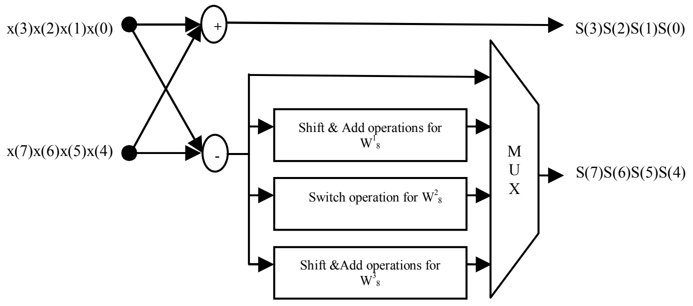
x(3)x(2)x(1)x(0)
x(7)x(6)x(5)x(4)
+
-
S(3)S(2)S(1)S(0)
Shift & Add operations for W_8^1
Switch operation for W_8^2
Shift &Add operations for W_8^3
MUX
S(7)S(6)S(5)S(4)

Figure 5. Butterfly processor with no complex multiplication.

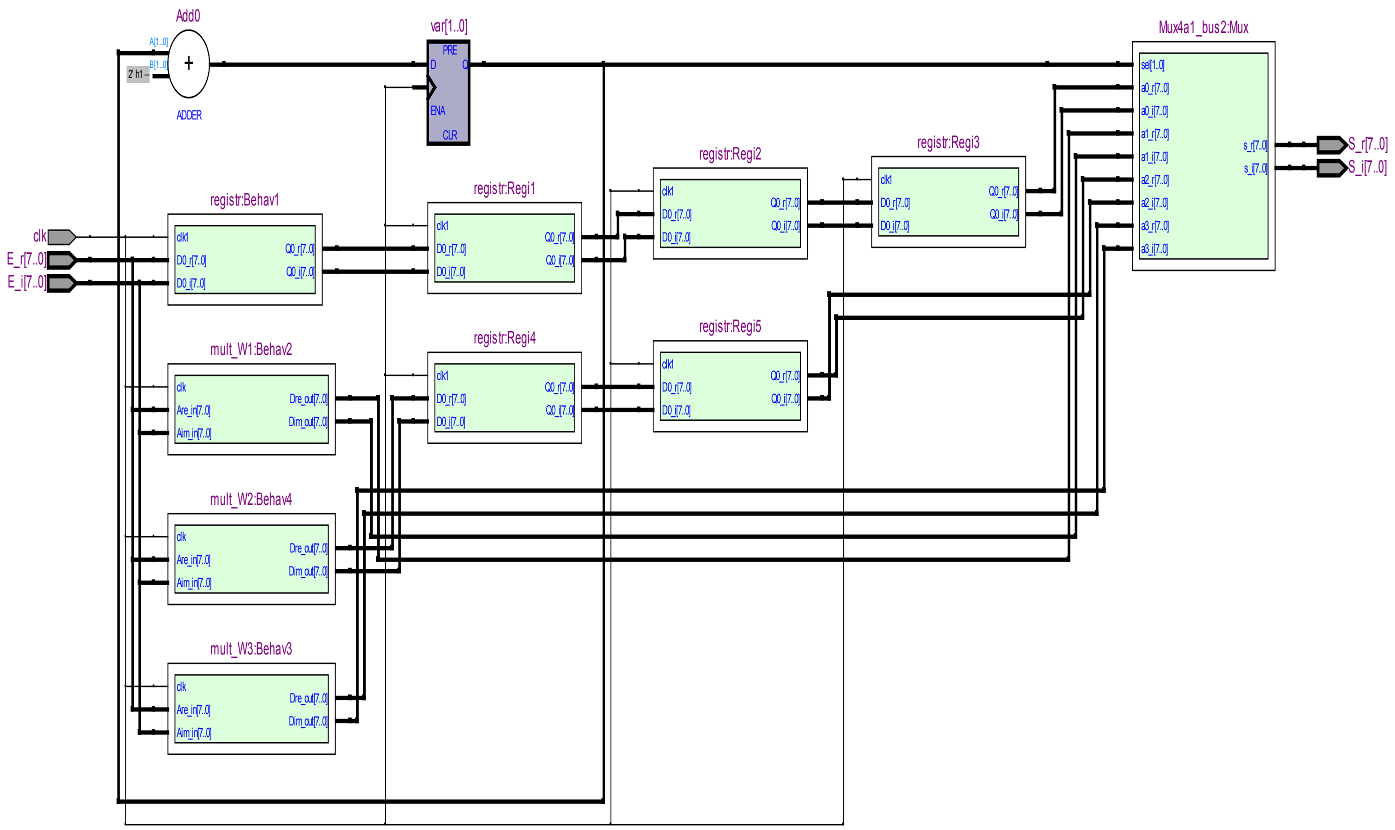

Figure 6. RTL viewer implementation of the butterfly processor architecture.

TABLE II. COMPARISON OF THE SECOND PROPOSED ARCHITECTURE WITH CONVETIONAL R2MDC ARCHITECTURE IN TERMS OF NUMBER OF ARITHMETIC OPERATIONS AND NUMBER OF MULTIPLIERS.

Algorithm	**Complex Multipli-cations**	**Real Multipli-cations**	**Real add/ sub**	**Complex multipliers**	**Real multipliers**
R2MDC	12	48	72	3	12
1st Proposed architecture	4	12	60	1	3
2nd Proposed architecture	0	0	64	0	0

The performance of the proposed processor is compared in Table II. The employed architecture within the proposed butterfly processor allows us to eliminate completely the multiplication operations inside the 8-point FFT/IFFT and cut down the number of complex multiplications for large points FFT processors (FFT 64, FFT 512).

In consequence, the optimized 8-point in this work can be used to build a 64-FFT points pipelined processor for OFDM systems (IEEE 802.11a, IEEE 802.11n MIMO-OFDM) [11] [12], due to the lower multiplier requirement compared to other proposed architectures [13] [14].

In order to verify the credibility of computation of the FFT core, we have simulated the calculation of 8-point FFT in timing and functional simulation mode. The output of the bloc FFT implemented completely matches the output of an FFT function written in Matlab representing a theoretical example of FFT calculation.

V. CONCLUSION

In this paper, a novel 8-point FFT processor based on pipeline architecture with no complex multiplication was described and compared to Cooley-Tukey and R2MDC processor. The proposed architecture gives an advantage in terms of area, resource usage using the complex multiplication reduction approach for large points FFT and pipelining method.

The simulation results show that the proposed architecture significantly reduces the number of operations inside the FFT processor compared to the Cooley-Tukey and R2MDC processors.

The proposed processor can be integrated with other components to be used as stand-alone processor applied for wireless communication systems.

REFERENCES

[1] Weidong Li, "Studies on implementation of lower power FFT processors," Linköping Studies in Science and Technology, Thesis No. 1030, ISBN 91-7373-692-9, Linköping, Sweden, Jun. 2003.

[2] S. He and M. Torkelson, "A new approach to pipeline FFT processor," In Proc. of the 10th Intern. Parallel Processing Symp. (IPPS), pp. 766–770, Honolulu, Hawaii, USA, April 1996.

[3] W.Li, L.Wanhammar, "Complex multiplication reduction in FFT processor," SSoCC'02, Falkenberg, Sweden, Mar. 2002.

[4] M.Arioua, S.Belkouch, M.M.Hassani, "Complex multiplication reduction in pipeline FFT architecture," In Proc. of 20th Intern. Conf. on Computer Theory and Applications (ICCTA), Alexandria, Egypt, Oct. 2010.

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

[6] U.M. Baese, Digital Signal Processing with Field Programmable Gate Arrays, 3rd ed. Springer, 2007.

[7] M.Petrov, M. Glesner, "Optimal FFT Architecture Selection for OFDM Receivers on FPGA," In Proc. of 2005 IEEE Intern. Conf. on Field Programmable Technology, pp. 313–314, PI. 0-7803-9407-0, Singapore.

[8] W. Li and L. Wanhammar, "An FFT processor based on 16-point module," In Proc. of NorChip Conf., pp. 125–130, Stockholm, Sweden, Nov. 2001.

[9] Y.Jung, H.Yoon, J.Kim, "New Efficient FFT Algorithm and Pipeline Implementation Results for OFDM/DMT Applications," IEEE Transactions on Consumer Electronics, vol.49, No.1, 2003, pp. 14-20.

[10] P.Verma, H. Kaur, M.Singh, M, B.Singh, " VHDL Implementation of FFT/IFFT Blocks for OFDM," In Proc. of Intern. Conf. on Advances in Recent Technologies in Communication and Computing, pp. 186-188, PI. 978-1-4244-5104-3, Kerala, 2009.

[11] B.Wang, Q. Zhang, T.Ao, M.Huang, "Design of Pipelined FFT Processor Based on FPGA," In Proc. of the 2nd Intern. Conf. on Computer Modeling and Simulation (ICCMS'10), pp. 432–435, ISBN 978-1-4244-5642-0, Jan. 2010, Sanya, Hainan.

[12] H.L.Lin, H.Lin, Y.C. Chen and R.C. Chang, "A Novel Pipelined Fast Fourier Transform Architecture for Double Rate OFDM Systems," IEEE workshop on signal processing systems design and implementation, pp. 7-11, ISBN 03-8504-7, Texas, USA, 2004.

[13] K. Maharatna, E. Grass, U. Jagdhold, "A Low-Power 64-point FFT/IFFT Architecture for Wireless Broadband Communication," 7th Intern. Conference on Mobile Multimedia Communication (MoMuC) 2000, Tokyo, Japan. 2A-2-1-2A-2-4.

[14] C. Eleanor Chu, G. Alan, Inside the FFT black box Serial and Parallel Fast Fourier Transform Algorithms, part II, CRC Press LLC, Boca Raton, 2000.