Industrial Science
Vol.1,Issue.4/April. 2014 **ISSN** : 2347-5420

Research Papers

IMPLEMENTATION OF AN OPTIMIZED 8-POINT FFT IN PIPELINE ARCHITECTURE FOR FPGAS SYSTEMS

Kiran Prajapati[1], Amit Sharma[2] and Hardik Thakor[3]

[1]M.E Student Electronics and Communication, SSESGI, Rajpur, Gujarat,
[2]Asst. Professor Electronics and Communication, SSESGI, Rajpur,
[3]Asst. Professor Electronics and Communication, SSSRGI, Vadasma,

Abstract

The FFT processor is one of the key components in the implementation of wideband OFDM systems. The speciation and physical layer (PHY) standard of Multiband Orthogonal Frequency Division Multiplexing (MB-OFDM) Ultra Wideband (UWB) system was denied by ECMA. The data sampling rate for the analog-to-digital converter to the physical layer is up to 528 M sample/s. So, the challenge is to realize the physical layer of the UWB system-especially the components with high computational complexity in Very-Large-Scale Integration (VLSI) implementation. One component is the Fast Fourier Transform (FFT) block, which is the demodulation block of OFDM signals and it has more design complexity.The purpose of this paper is to design a FFT solution for this system. The speciation is denned from the system analysis and literature research. All the design choices and considerations are concluded and explained. Based on the algorithm and architecture analysis, a Radix22 Pipeline FFT architecture is proposed, which is a less design complexity, small-area and low-power-consumption solution for MB-OFDM UWB system.

KEY WORDS:

Implementation , pipeline architecture , physical layer , Division Multiplexing .

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.

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:

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

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

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(N2). By using The Cooley–Tukey FFT algorithm, the complexity can be reduced to O (N.logr 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 = rS. 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:

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

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

And:

$$n_1, n_2, \ldots, n_{S-1}, n_S = 0,1$$

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

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

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

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

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

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

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

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

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 log2N 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).log2N complex multiplication and N.log2N 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

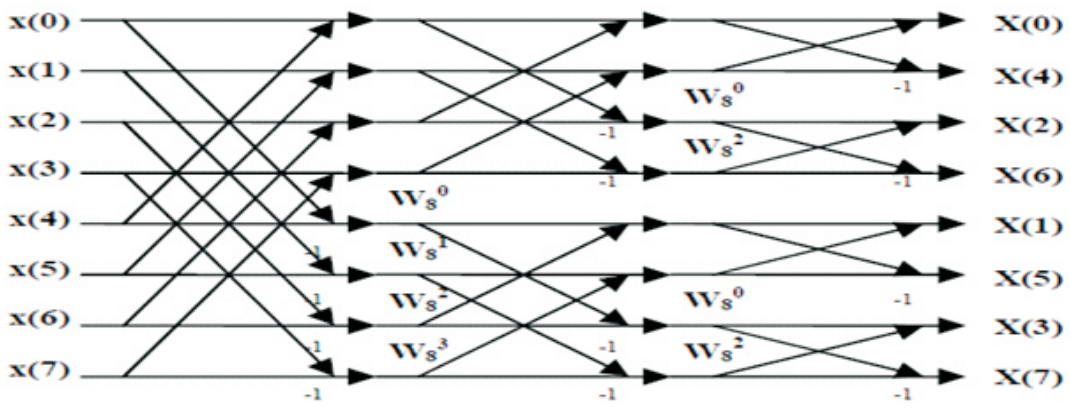

Figure 1. 8-point decimation-in-frequency FFT algorithm.

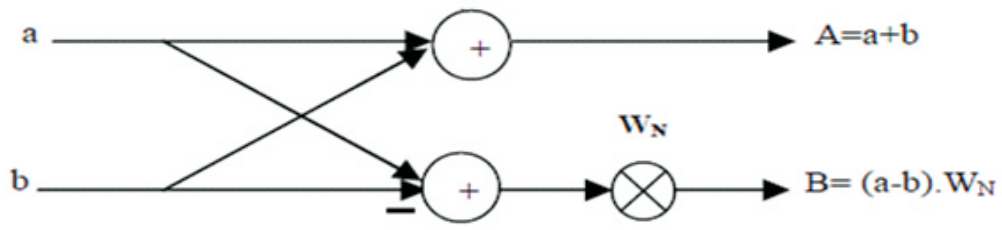

Figure 2. Basic Butterfly computation.

Radix-2 butterfly processor consists of a complex adder and complex subtraction. Beside that, an additional complex multiplier for the twiddle factors WN 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:

$$R + jI = (X + jY).(C + jS) \quad (4)$$

However the complex multiplication can be simplified:

$$R = (C - S).Y + Z \quad (5)$$

$$I = (C + S).X - Z \quad (6)$$

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

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.

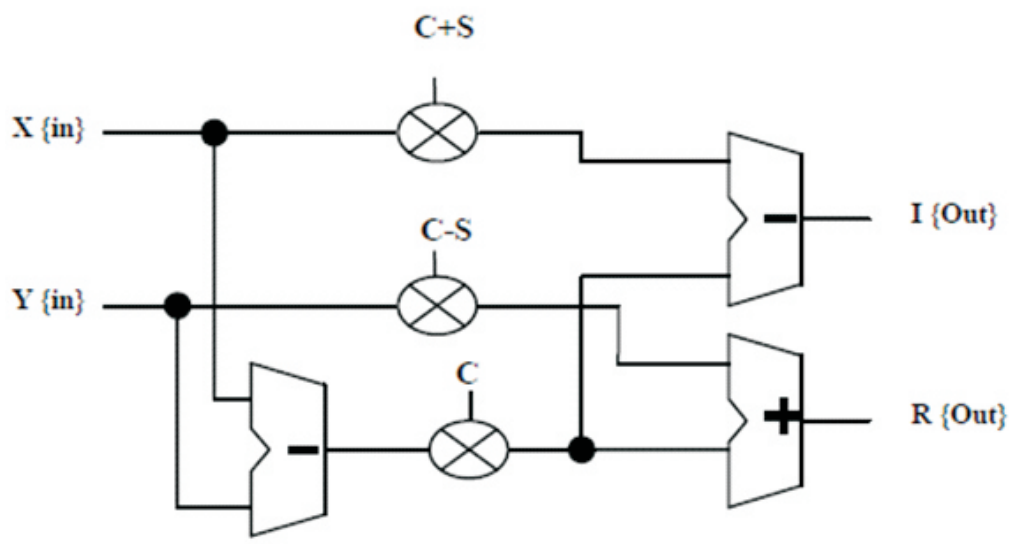

Figure 3. Implementation of complex multiplication.

In the 8-point FFT with radix-2 algorithm, the multiplication with W8 2=-j and W8 0 factors is trivial, multiplication with W82 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: W81, W83. This non-trivial complex multiplication was implemented with two multiplications for W81 and three multiplications for W8 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 wipeout completely the complex multiplication inside the butterfly processor. The implementation complexity of non-trivial twiddle factors W81, W8 3 was replaced by basic operation of shift-and-add which is shown in Fig.

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].

$$n = 4n2 + 2n1 + n0 \quad n2=0,1 \quad n1=0,1 \quad \text{and} \quad n0=0,1$$
$$k = 4k2 + 2k1 + n0 \quad k2=0,1 \quad k1=0,1 \quad \text{and} \quad k0=0,1$$

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 Commentator (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.

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).

VERIFICATION & RESULTS

The results which targets for FPGA(Virtex-4)are presented.

Synthesis:

Synthesis is the process of taking a design written in a hardware description language, such as VHDL, and compiling it into a netlist of interconnected gates which are selected from a user-provided library of various gates. FPGA targeted synthesis results are presented in following parts.

FPGA Targeted Results

The proposed design is synthesized and implemented by Xilinx ISE which is targeted for FPGA Xilinx Virtex4 implementation. The reason to use this FPGA is that the synthesis results can be compared with [3] to show the advantages of the proposed architecture. Because Virtex4 FPGA includes DSP48 components, the "+", "-", and "*" signs are used in VHDL coding, which are directly mapped to DSP48 components.

RTL of 128pt Pipeline FFT

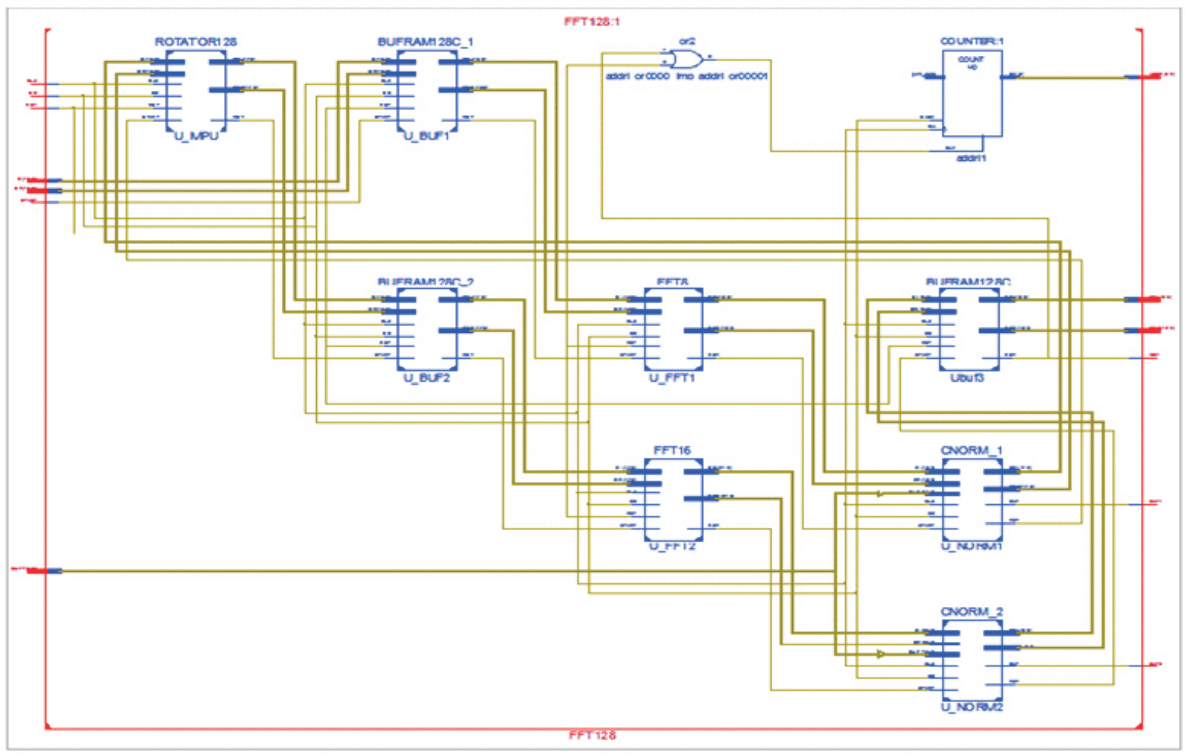

Figure 4: RTL

Devise Utilization Summary of 128pt Pipeline FFT

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	3919	6144	63%
Number of Slice Flip Flops	4263	12288	34%
Number of 4 input LUTs	6517	12288	53%
Number of bonded IOBs	90	240	37%
Number of FIFO16/RAMB16s	4	48	8%
Number of GCLKs	1	32	3%
Number of DSP48s	4	32	12%

Figure 5: Devise Utilization Summary

Simulation Result of 128pt Pipeline FFT
The resource usage of Virtex4 is shown in above figure.

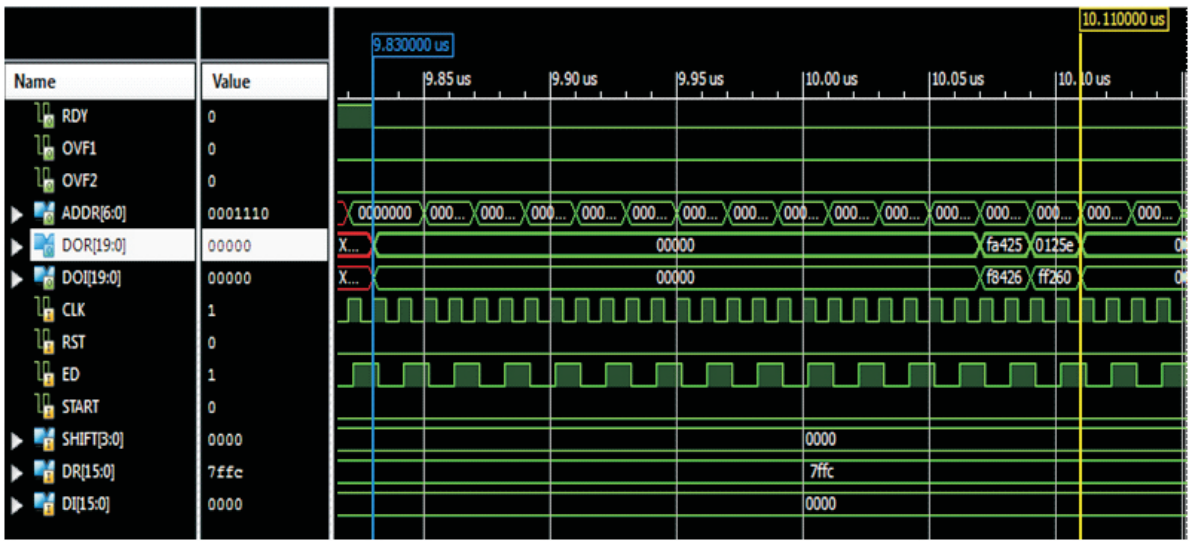

Figure 6 : Simulation Result

Table1 : The resource usage of Virtex4

Logic Utilization	Used
Total Number of Slice Register	3919
Number used as Flip Flop	4263
Number of 4 input LUTS	6517
Number of DSP48s	4

Summary:

This chapter shows the validation of proposed processor.FPGA targeted results are analyzed and comapared with other implementation.

CONCLUSIONS

Present work involves the design flow from dining the processor speciation's till to the synthesis results. It can be obtained from the results that the proposed architecture is suitable for FPGA implementation and dramatically reduces the area and power consumption and has less complexity. The

key contributions of this thesis work are summarized as follows:

a.Different FFT processor structures are compared on the architecture level. It is found that R22SDF structure is the optimum option for FFT implementation of Multiband UWB system with regard to the reduce memories and arithmetic blocks utilization. Important design choices and considerations are also ana- lyzed and concluded.
b.A novel pipeline structure is proposed, which is called Radix22Pipeline. It is a less complexity, small-area and low-power-consumption.
c.The proposed FFT solution is varied. The FPGA targeted synthesis results are presented.

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