2010 2nd International Conference on Education Technology and Computer (ICETC)

Design of Fixed-Point High-Performance FFT Processor

Li Wenqi, Wang Xuan, Sun Xiangran
Communication University of China
Beijing, China
liwenqi0724@yahoo.com.cn

***Abstract*—This paper, based on the complexity and hardware requirement of conventional FFT algorithms, analyses the architecture of radix-4 Single-Path Delay Feedback (SDF) in DIF and present an efficient FFT processor for real-time applications. A test bench is built up comparing Signal to Quantization Noise Ratio (SQNR) performances of the processors with a float-point model and fixed-point one. Several values of I/O word length and twiddle factor word length are implemented, to investigate the quantization effect of fixed-point arithmetic with limited precision derived from rounding or truncation errors, for enhancing the output performance. The simulation tests and an implementation of 1024-point FFT targeted on XC5VSX50T FPGA, show that the proposed FFT processor has a high computational frequency and hence suitable for usual OFDM wireless applications.**

Keywords-FFT; radix-4; twiddle factor; pipeline; fixed-point

I. INTRODUCTION

FFT, being a key component operation, is one of the most widely utilized algorithms for calculating the Discrete Fourier Transform (DFT) in digital signal processing (DSP) algorithms, owing to its efficiency in reducing computation time [1]. Recently, real-time processing FFT has played a significant role in many communication systems [10], especially, in which Orthogonal Frequency Division Multiplexing (OFDM) technology is adopted as modulation method, such as high-definition TV (HDTV), China Mobile Multimedia Broadcasting (CMMB), digital video broadcasting—terrestrial (DVB-T), and digital audio broadcasting (DAB). These applications require long length FFT for multiple carrier modulation, such as 8192/2048-point FFT of DVB-T and 4096/1024-point of CMMB.

Cooley and Tukey first revealed the idea of fast Fourier transform (FFT) to demonstrate a significant computational reduction from $O(N^2)$ to $O(N\log N)$ by taking full advantage of symmetry and periodicity properties of the twiddle factors [8]. Since that epochal work in 1965 [1], several algorithms have been proposed to further reduce the computational complexity, including radix-2^m Multipath Delay Commutator (radix-2^m MDC), radix-2^m Single-path Delay Feedback (radix-2^m SDF), radix-4 MDC, SDF, et al [3]. These algorithms decompose the N-length FFT operand into odd and even half parts recursively and reduce the number of complex multiplications by making efficient use of symmetric properties of FFT twiddle factor as possible.

Since pipelined FFT can be combined applicably with sequential nature of sampling, pipeline FFT implementations are highly appropriate for meeting real-time processing demand [10]. Several architectures for pipeline FFT processors listed in Table I, being proposed over the last 3 decades, have the distinctive merits and common requirements [7]. Since pipeline FFT architecture consumes large memory, reducing its memory requirement will save significant amount of area.

Above all, the accuracy of FFT/IFFT module is the most important design factor of system performance. In practice, fixed-point arithmetic is used to implement FFT algorithms in hardware because it is not possible to keep infinite resolution of coefficients and operations. All coefficients and input signals have to be represented with finite number of bits in binary format depending on the tradeoff between the hardware cost (memory usage) and the accuracy of output signals. Generally speaking, each multiplication may introduce errors by rounding operations or truncations, which is referred as arithmetic quantization error, as well as all the twiddle factors whose loss due to the inexact coefficients is called coefficient quantization error [8]. Errors inherited from the early stages can be viewed as added noisy input to the next stage. Error signals that arise at each stage accumulate at the output of processor. The word lengths of data and coefficients chiefly affect precision, quantization errors, and hardware complexity [10]. Increased word length means increased precision and reduced quantization error, meanwhile the increased size, both for memory and arithmetic operations at the cost of area and power. On the other hand, we can choose shorter word-lengths to maintain a lower hardware cost, at the sacrifice of precision. Thus, researches for optimized solutions of finite precision effect, is of great necessity.

This paper analyses the architecture of radix-4 Single-Path Delay Feedback (SDF) algorithm in DIF and present an efficient FFT processor for the real-time applications. A test bench is built up comparing Signal to Quantization Noise Ratio (SQNR) performance of the processors with a float-point model and fixed-point one. Several values of I/O word length and twiddle factor word length are implemented, to investigate the quantization effect of fixed-point arithmetic with limited precision derived from rounding or truncation errors, for enhancing the output performance. The simulation tests and an implementation of 1024-point FFT targeted on XC5VSX50T FPGA, show that the proposed FFT has a high clock frequency and hence suitable for usual OFDM wireless applications.

II. FFT Algorithm Design

TABLE I. COMPARISON OF THE PIPELINE FFT ARCHITECTURES

Algorithm	***Complex Multiplies***	***Complex Adders***	***Memory Size***	***Number of Switches***	***Control***
R2MDC	2 $\log_4 N$ -1	4 $\log_4 N$	3N/2-2	4 $\log_4 N$ -2	simple
R2SDF	2 $\log_4 N$ -1	4 $\log_4 N$	N-1	4 $\log_4 N$	simple
R4SDF	4 $\log_4 N$ -1	8 $\log_4 N$	N-1	4 $\log_4 N$	medium
R4MDC	3($\log_4 N$ -1)	8 $\log_4 N$	5N/2-4	12(4$\log_4 N$-1)	simple
R4SDC	$\log_4 N$ -1	3 $\log_4 N$	2N-2	5 $\log_4 N$	complex
R2^2SDF	$\log_4 N$ -1	4 $\log_4 N$	N-1	5 $\log_4 N$	simple

Among these architectures listed in Table I [4] [7], delay feedback (DF) approaches are always more efficient than the corresponding delay commutator (DC) approaches in terms of the required memory size. Radix-4 SDF requires fewer multipliers those by radix-2 SDF; Radix-2 SDF is simple and regular. Radix-2^2 SDF is a tradeoff between the R2SDF structure and the multiplicative complexity of R4SDF. Even if the complexity of multiplications was reduced in radix-2^2 algorithm, there is still an inevitable problem for the pipeline FFT implementation. Because 4 real multiplications and 2 real additions make up one multiplier, which consumes not only large area but also almost 50~80% of total power in the previous work [2], and date rate is a very critical criterion for real-time high processing. Owing to twice date rate during each clock period , half the number of stages, hence 20% [13] less calculation amount of radix-4 than those of radix-2, radix-4 pipeline architecture is more appropriate for systems that requires a high processing rate rather than a radix-2 one [9].

Due to the OFDM technology based on the application of multiple sub-carriers, DIF scheme, suitable for stream input, is adopted to avoid any transformation between time domain and frequency domain.

The N-point DFT of a set of N-point input signals{x[n]} is defined as follows:

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

Where $W_N^{nk}=e^{-j2\pi nk/N}=\cos(2\pi nk/N)-j\sin(2\pi nk/N)$ is the twiddle factor.

Thus, the function of radix-4 butterfly is represented by:

$$X(4r+l)=\sum_{n=0}^{N/4^P-1}\{x(n)+x(n+\frac{N}{4^P})W_4^l+ x(n+\frac{2N}{4^P})W_4^{2l}+x(n+\frac{3N}{4^P})W_4^{3l}\}\mathrm{W}_{N/4^{(P-1)}}^{nl}W_{N/4^P}^{nr} \tag{2}$$

Where $r=0\sim(N/4^P)-1$, $l=0,1,2,3$, $P=1\sim\log_4 N$, and $n=0\sim(N/4)-1$.

III. FFT Hardware Architecture

A. FFT Theory Structure

From Equation (2), When L=1 [13], we can see the corresponding radix-4 DIF architecture in Fig. 1 and the signal flow in Fig. 2.

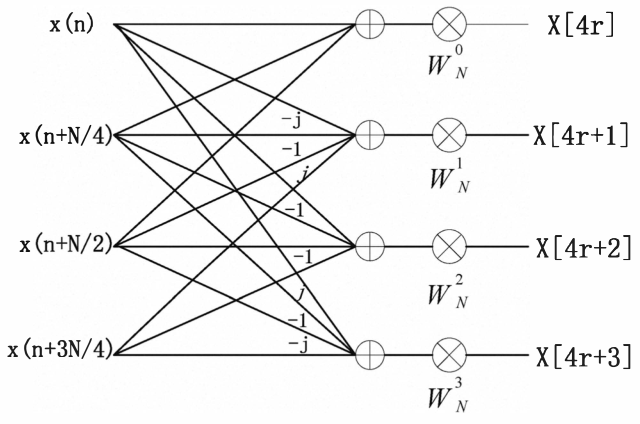

Figure 1. Radix-4 DIF butterfly unit

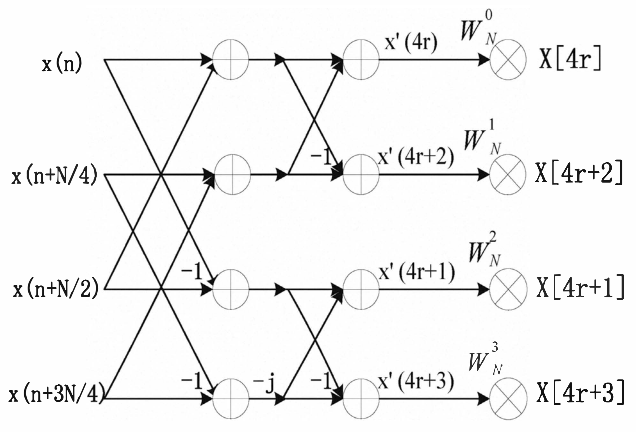

Figure 2. Signal flow in the radix-4 DIF

B. Operation of butterfly unit

The operation of the first stage starts immediately after the (3N/4+1) point is send in, whose memory cells can provide 3N/4 points input with storage at least. As seen from Fig. 3 is the radix-4 unit of the proposed FFT. SRAMs are chosen for temporary storage elements.

Four steps, as follows, consist of a single operation of the radix-4 butterfly unit.

1) During the preceding 3 clock of N/4 data points, data x(n), x(n+N/4), x(n+N/2), are sequently read and stored in 3 individual SRAM cells.

2) During the 4th clock of data points, the operation of the radix-4 butterfly starts immediately after data x(n+3N/4) is read in, to be computed with data stored in the 3 SRAMs. the 1st in 4 results of every butterfly operation will be directly delivered into the complex multiplier for multiplication with corresponding twiddle factor, the rest 3 will then be stored in SRAMs.

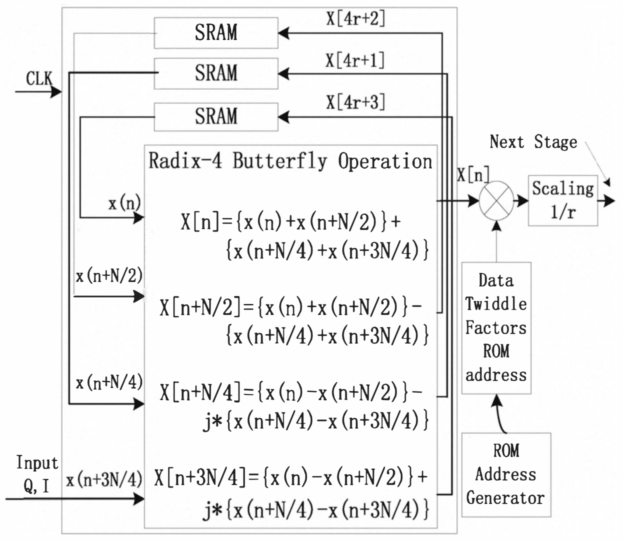
SRAM
X[4r+2]
SRAM
X[4r+1]
CLK
SRAM
X[4r+3]
Radix-4 Butterfly Operation
Next Stage
X[n]
Scaling 1/r
x(n)
X[n]={x(n)+x(n+N/2)}+{x(n+N/4)+x(n+3N/4)}
x(n+N/2)
X[n+N/2]={x(n)+x(n+N/2)}-{x(n+N/4)+x(n+3N/4)}
Data Twiddle Factors ROM address
x(n+N/4)
X[n+N/4]={x(n)-x(n+N/2)}-j*{x(n+N/4)-x(n+3N/4)}
Input Q, I
x(n+3N/4)
X[n+3N/4]={x(n)-x(n+N/2)}+j*{x(n+N/4)-x(n+3N/4)}
ROM Address Generator

Figure 3. Schematic of the proposed radix-4 butterfly

3) From the 5th clock to the 7th clock of N/4 data points, the rest 3 results x′(n+N/4) , x′(n+N/2) , x′(n+3N/4) from the last step, are delivered to the next pipline stage, after multiplication, with twiddle factors stored in the ROM, and scaling. The next N data points will be read in SRAMs.

4) From 8th clock of N/4 data points, next N data points begin to be calculated in butterfly operation.

C. Complex Multiplier

We assume 2 complex number: $A = a + jb$ and $B = c + jd$, and hence the multiplied result of A and B is represented as follows:

$$Y = A\times B = (ac-bd)+j(ad+bc) = y_r + y_i \tag{3}$$

From Equation (3), we infer one complex multiplication is composed of 4 real multiplications and 2 real additions., and then we can obtain:

$$y_r = ac-bd = ac-ad+ad-bd = (c-d)a+(a-b)d$$
$$y_i = ad+bc = ad-bd+bd+bc = (c+d)b+(a-b)d \tag{4}$$

We can define:

$$S_0 = (a-b)d, \quad S_1 = (c-a)a, \quad S_2 = (c+d)b \tag{5}$$

Then, we can express Y as:

$$Y = y_r + y_i = (S_0 + S_1) + (S_0 + S_2) \tag{6}$$

By equation (6), we can reduce the computation complexity of conventinal method in eaquation (3) to 3 real multiplications and 5 real additions. In integrated circuits implementataion, area requirement of one multiplier is distinctly much more than that of one adder.

Due to serial I/O in pipeline architecture, each buttlerfly only needs one complex multiplication, different from 3 complex multiplications in parallel architecture, and that no twiddle factor is needed for 1st result of each butterfly, makes multiplication utilization 75%[11].

D. Memory

1) ROM: In DIF scheme, decomposition of the first stage needs most of twiddle factors, and the ensuing one needs 1/r (r=radix) of twiddle factors of the preceding one., then each stage need 3N/4^i -2. At decomposition of first stage, N/4 butterfly units, 3N/4 twiddle factors W_N^n , W_N^{2n} *and* W_N^{3n} , respectively, are needed, where $n = 0,1,\ldots\ldots, N/4-1$. Take 1024-point FFT as an example, since 1/4 of twiddle factors are equal to $W_N^{0n} = 1$, and if 14-bit resolution represents the twiddle factors, 2 768 ×14 ROMs are sufficient to separately keep all of twiddle factors required at first stage.

2) SRAM: Dual-port SRAMs, are adopted as memory cells to constructed by 18kbits block RAMs, which only consume about as much 1/3 area as that of shift registers and no a large amount of dynamic power consumption, as DRAM storage elements, and save a large amount of operation time of control logics in addition. According to Fig. 3, every stage in pipeline needs 3 SRAM and 1 ROM. Take the first stage of 4096-point FFT as an example. A total of 3072 points for x[n], x[n+N/4], x[n+N/2], must be stored in 3 2048×14 SRAMs.

E. Control unit

Each butterfly has its own control unit, which mainly controls states of butterfly, such as whether it is in pipeline operation, selection of SRAM and complex multiplication, and generating addresses of SRAMs and ROMs. The units also generate enable signal of output and valid signal of input to next stage. The main control to RAM is for I/O data flows, because of the public use of RAM address lines, Write enable signal is needed to protect from writing in false addresses. Because only one QD quadrant twiddle factor is kept in one ROM, the plus quantity twiddle factor phase has to be converted [11]. In addition, to avoid saturation overflow and complicated dynamic scaling [6] processing, constant scaling is introduced by dividing butterfly outputs by r (r=radix) at the end of each stage.

IV. Simulation and Implementation Results

Here, we present a simulation-based method for FFT error analysis of the Signal-to-Quantization-Noise Ratio (SQNR), which evaluates accuracy of the transform in proposed architecture.

We present a block diagram of simulation-based analysis in Fig. 4 [10]. To perform the simulation, float-point of matlab model and fixed-point C++ model with the proposed FFT algorithm, were developed. According to configuration constraints, such as the word-length of each stage, the rounding or truncation of stages, and the I/O word-length, the C++ model can obtain proper operation results. Then, SQNR can be calculated by comparing the fixed-point output

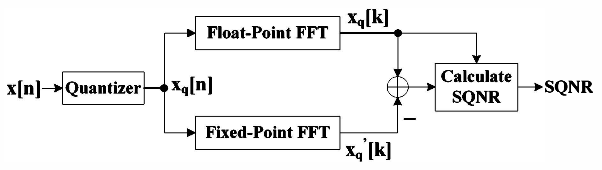

Figure 4. Block diagram of the simulation analysis

with the float-point output; the formula of the calculation is expressed as

$$SQNR_c = 10\log_{10}\frac{\sum_{k=0}^{N-1}|X_q(k)|^2}{\sum_{k=0}^{N-1}|X_q(k)-X'_q(k)|^2} \tag{5}$$

During simulation, random patterns are generated as inputs and, then the resulting $SQNR_c$s are averaged as an estimated average of the SQNR.

In the float-point model, the SQNR can exceed 306dB if the float-point outputs are compared to values of y=fft (x, N) from matlab. As for the fixed-point model, Each I/O data signal, intermediate value and all the twiddle factors are represented with limited and designated number of bits; and calculation results with 2 fixed-Point numbers will be rounded or truncated, especially for butterfly multiplication results, to avoid greater word-lengths than the operands.

The word-length of twiddle factors coefficients is also set to 14 bits. The I/O word-length is swept from 8 bits to 19 bits. Both 512-point and 1024-point FFT of radix-4 are simulated. As seen from Fig.5, the SQNR values of all 1024-point simulations are lower than that of 512-point simulations. In addition, a radix-2 1024-point FFT, comparing with that of radix-4, is simulated. From Fig.5 and data we obtain, we find that the SQNR values of radix-2 are lower than that of radix-4 in the same transform length, and the disparity becomes larger as I/O length increases.

Simulation tests above demonstrate that the internal arithmetic rounding errors typically accumulate successively over FFT stages, because the number of stages in pipelined architecture increases as the length of pipelined FFT increases.

We synthesized the proposed radix-4 1024-point FFT using Verilog HDL on ISE10.1 of Xilinx at 16bits for I/O word-length, targeted on XC5VSX50T FPGA that can run at a maximum clock frequency of 152.3MHz. Then, after implementation and P & R (place & route), we can still obtain a maximum clock frequency of 100.6MHz.

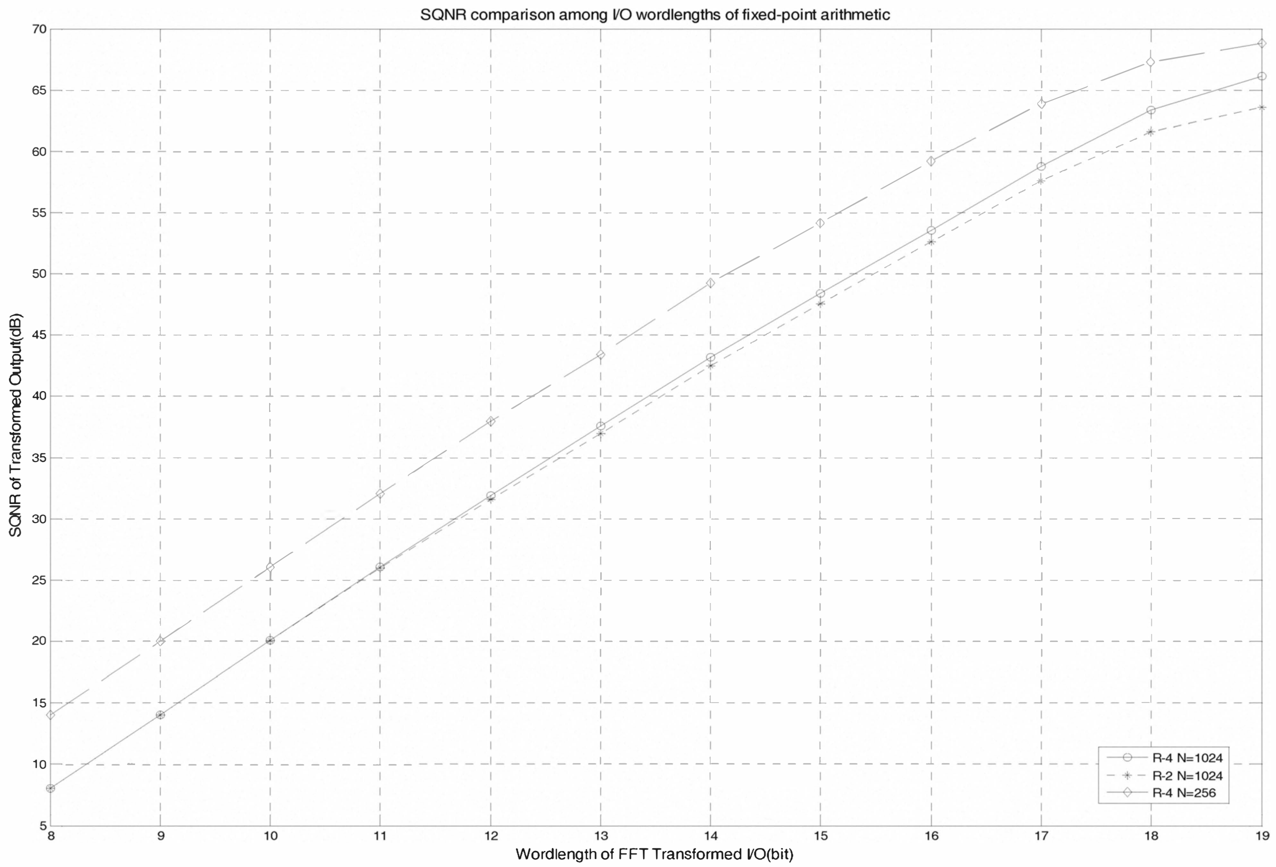

Figure 5. SQNR comparison among I/O word-lengths of fixed-point arithmetic

V. Conclusion

This work has presented a radix-4 algorithm pipeline FFT in fixed-point arithmetic and a test bench is built up comparing SQNR of the processor with a float-point model in matlab and fixed-point one in C++. Several values of I/O word length and twiddle factor word length are implemented, to investigate the quantization effect of fixed-point arithmetic with limited precision derived from rounding or truncation errors, for enhancing the output performance in terms of SQNR. Simulation tests show that our FFT achieves a SQNR of 53 dB at an I/O word-length of 16 bits. An implementation and P & R of 1024-point FFT targeted on XC5VSX50T FPGA that can run at a maximum clock frequency of 100.6MHz after P & R, show that the proposed has a high computational frequency, and hence suitable for usual OFDM wireless applications.

Acknowledgment

The authors would like to thank their colleagues of the DAB lab of Communication University of China for many substantial discussions. The authors would also like to thank the anonymous reviewers for their constructive comments.

References

[1] J.W. Cooley and J.W. Tukey, “An Algorithm for the Machine Calculation of Complex Fourier Series,” Math. Computation, vol. 19, pp. 297-301, 1965.

[2] J. Melander, Design of SIC FFT Architectures, Linköping Studies in Science and Technology, Thesis No. 618, Linköping University, Sweden, 1997

[3] S.-S. He and M. Torkelson, “Designing Pipeline FFT Processors for OFDM (De)Modulation,” Proc. URSI Int’l Symp. Signals, Systems,and Electronics (ISSSE ’98), pp. 256-262, 1998.

[4] L. Yang, K. Zhang, H. Liu, J. Huang, and S. Huang, “An Efficient Locally Pipelined FFT Processor,” IEEE Trans. Circuits and Systems II: Express Briefs, vol. 53, pp. 585-589, July 2006.

[5] S. Johansson, S. He, and P. Nilsson, “Wordlength Optimization of a Pipelined FFT Processor,” Proc. 42nd Midwest Symp. Circuits and Systems, pp. 501-503, 1999.

[6] Thomas Lenart and Viktor Öwall. “Architectures for Dynamic Data Scaling in 2/4/8K Pipeline FFT Cores.”

[7] J.-Y. Oh and M.-S Lim, “Area and Power Efficient Pipeline FFT Algorithm,” Proc. IEEE Workshop Signal Processing Systems Design and Implementation, pp. 520-525, 2005.

[8] Wei-Hsin Chang and Truong Q.Nguyen, Fellow, IEEE, “On the Fixed-Point Accuracy Analysis of FFT Algorithms.” IEEE Trans. Signal Processing, vol. 56, no. 10, pp 4673-4682 .Oct .2008.

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

[10] Cheng-Yeh Wang, Chin-Bin Kuo, and Jing-Yang Jou, Fellow, IEEE, “Hybrid Word-Length Optimization Methods.” IEEE Trans. Computers, Vol. 56, no. 8, pp.1105-1118, August .2007.

[11] HE Xing, ZHANG Tie-jun, HOU Chao-huan, “Design and Implementation of a Pipelined FFT/IFFT Processor”[A]. Microelectronics and Computer, vol. 24, no.2 (2007) 04- 0141- 03.

[12] Wang Chua - Chin , Huang Jian - Ming , Cheng Hsian -Chang. A 2K/ 8K mode small-area FFT processor for OFDM demodulation of DVB-T receivers. IEEE Trans. Consum. Electron., vol. 51, no.1, pp.28-32, February 2005

[13] HUANG Qiu-yuan, LEI Yan-min, and LI Wei-guo. “Design of 2k-8k FFT Processor and ROM Optimized in DVB-T Receiving System.”[A]. Microelectronics and Computer, vol. 26, no.2, February 2009, 1000-7180(2009)02-0016-05.