Analysis and Implementation of a Low Power/High Speed 64 point pipeline FFT/IFFT Processor

P. K. Srikanth
Department of Electronics and Communication Engineering,
K. S. Rangasamy College of Technology,
Namakkal Dist., Tamilnadu-637215, India.
pksrikanth@gmail.com

C. Saranya
Department of Electronics and Communication Engineering,
K. S. Rangasamy College of Technology,
Namakkal Dist., Tamilnadu-637215, India.
saranya1209@yahoo.com

Abstract

For hardware implementations, the various FFT processors available are mainly classified into memory-based and pipeline architecture. A pipelined FFT/IFFT processor is efficiently implemented in this paper. The pipelined FFT is viewed as the leading architecture for real time applications. The design adopts a single-path delay feedback style as the proposed hardware architecture, since the single delay feedback (SDF) pipeline FFT is good in its requiring less memory space (about N-1 delay elements) and its easy multiplication computation and the ease of design of control unit. Thus a pipelined FFT architecture accounts for both low power consumption and high speed of operation. To eliminate the read-only memories (ROM's) used to store the twiddle factors, the proposed architecture applies a reconfigurable complex multiplier and bit-parallel multipliers to achieve a ROM-less FFT/IFFT processor, thus consuming lower power than the existing works. This proposed architecture is suited for the power-of-2 radix style of FFT/IFFT processors.

Keywords: Transmission Pipeline FFT/IFFT, ROM-less FFT, Low Power FFT.

1. Introduction

Many digital signal processing (DSP) algorithms handle data on a frame basis. As the stream enters the DSP system, sets of data samples are taken together and an algorithm is computed on them. The Discrete Fourier Transform (DFT) and the Fast Fourier Transform (FFT) algorithms are usually the first step in this kind of processing. Both of them, transform frames of a signal into Fourier's domain and the resulting coefficients are then analyzed or processed depending on a particular application.

The better execution time of the FFT over the classical DFT algorithm places it as the standard selection for high performance signal processing applications. The relevance of this mathematical method impulsed the development of special purpose VLSI systems to compute it and many different approaches were followed looking forward to maximize the speed of operation, minimize area or minimize power consumption. Fast Fourier Transforms (FFT) is the fast implementation of the Discrete Fourier Transform (DFT) which relies on mathematical simplification and classification of the input sequence to achieve their performance gain. The fast Fourier transform (FFT) was proposed by Cooley and Turkey [1] to efficiently reduce the time complexity to O (Nlog 2N) operations in comparison to the straight forward DFT that requires O(N2) operations, where N denotes the FFT size.

The main problem domain for FFT processors includes achieving high throughput, low power operation and less area. For hardware implementation, various FFT processors have been proposed [2]-[15]. These implementations can be mainly classified into memory-based and pipeline architecture styles.

Memory-based architecture is widely adopted to design an FFT processor. This deign style is usually composed of a main PE and several memory units, thus the hardware cost and the power consumption are both lower than the other architecture style. However, this kind of architecture style has long latency, low throughput, and cannot be parallelized. On the other hand, the pipeline architecture style can get rid off the disadvantages of the foregoing style, at the cost of an acceptable hardware overhead

More over the FFT computation often needs to multiply input signals with different twiddle factors for an outcome, which results in higher hardware cost because a large size of ROM is needed to store the wanted twiddle factors.

The main research idea of this work is how to optimize the architecture of the available pipelined FFT processor. The chief objective is the reduction in power. By using new algorithms or architectures, it is even possible to achieve both power reduction and achieve high speed of operation, which is strength of high-level optimization. The organization of this paper is carried out in 4 sections. The next Section (II) deals

about the FFT algorithm. Section III describes the proposed design and FFT architecture used, the Section IV describes the simulation result for the various PE (Processing Element) used in the FFT architecture followed by the 64 point FFT processor simulation result.

The paper is structured as follows. In section 2 FFT algorithms are discussed. Section 3 Proposed systems. In section 4 simulation and result. Section 5 deals with the conclusions.

2. FFT Algorithm

The FFT algorithm is derived from the DFT (Discrete Fourier Transform) expression:

$$x(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \quad (1)$$

where k=0…(N-1) and $W_N = e^{-j2\pi/N}$

Dividing recursively the input or output sequence (time or frequency decimation) and taking advantage of the periodicity and symmetry of the complex exponential factors (“twiddle factors”) the number of complex multiplications is reduced. For this illustration, N is chosen to be a power of 2, that is, $N = 2^m$, where m is a positive integer. The length N is therefore an even number, and x(n) can be separated into two sequences of length N/2, where one consists of the even members of x and a second consists of the odd members. Splitting Eq. 1 into even-indexed and odd-indexed summations gives,

$$x(k) = \sum_{n_{evenm}=0}^{N-2} x(n) W_N^{nk} + \sum_{n_{odd}=1}^{N-1} x(n) W_N^{nk} \quad (2)$$

If 2m is substituted for n in the even-indexed summation and 2m + 1 is substituted for n in the odd-indexed summation (with m = 0, 1, . . . , N/2 − 1), the result is,

$$x(k) = \sum_{m=0}^{N/2-1} x(2m) W_N^{2mk} + \sum_{m=0}^{N/2-1} x(2m+1) W_N^{(2m+1)k}$$

$$= \sum_{m=0}^{N/2-1} x(2m) \left(W_N^2\right)^{mk} + \sum_{m=0}^{N/2-1} x(2m+1) \left(W_N^2\right)^{mk} W_N^k \quad (3)$$

$$X(k) = DFT_{N/2}\{x_{even}(m),k) + W_n^k DFT_{N/2}\{x_{odd}(m),k\} \quad (4)$$

FFT algorithms are based on the fundamental principle of decomposing the computation of discrete Fourier transform of a sequence of length N into successively smaller discrete Fourier transforms. There are basically two classes of FFT algorithms. They are decimation-in-time and decimation-in-frequency.

Table 1. Comparison of DFT and FFT Efficiencies

Transform Length (N)	DFT Operations	FFT Operations	DFT Ops/FFT Ops
16	256	64	4
128	16,400	896	18
1024	1.05*10^6	10,240	102
32,768	1.07*10^9	4.92*10^5	2185
1,048,576	1.10*10^12	2.10*10^7	52,429

The radix-2 decimation-in-frequency is obtained by the following steps:

1) Decomposing X(k) such that it is split into FFT of points 0 to N/2-1 and points N/2 to N-1

2) Then decimating X(k) into even and odd numbered Samples

3) Derivation performed in class

Radix-2 Decimation-in-frequency (DIF) algorithm:

- In radix-2, the "butterfly" element takes in 2 inputs and produces 2 outputs
- Butterfly implements 2-point FFT Computations:
- $(N/2)\log_2 N$ complex multiplications
- $N\log_2 N$ complex additions

In decimation-in-time, the sequence for which we need the DFT is successively divided into smaller sequences and the DFTs of these subsequences are combined in a certain pattern to obtain the required DFT of the entire sequence. In decimation-in-frequency approach, the frequency samples of the DFT are decomposed into smaller and smaller subsequences in a similar manner. Generally, FFT analyzes an input signal sequence by using decimation-in-frequency (DIF) or decimation-in-time (DIT) decomposition to construct an efficiently computational signal-flow graph (SFG).

3. Proposed Design

Generally, there are two popular hardware architectures to implement FFT algorithm for real-time applications. One is pipelined-based design and the other is memory-based design. The pipeline-based architecture is highly regular which can be easily scaled and parameterized in hardware design. The memory-based design be easily scaled and parameterized in hardware design.

Memory-based FFT processor architecture is single PE architecture. They are designed to increase the utilization rate of butterfly PE and reduce the redundancy of multiple butterfly PEs. The characteristic of this architecture is that it only uses one radix-r butterfly PE to compute all butterflies.

Pipelined FFT Hardware Architecture:

The Pipeline FFTs can be further classified in two groups:

1) Feed forward structures (Commutator)
2) Feedback structures

These 2 groups can also be categorized according to radix:

- Radix-2 structures
- Radix-4 structures

The key to pipeline FFTs is the utilization factor (i.e. want all computations elements to be active as much as possible). High utilization factor means that higher pipeline FFT efficiency is possible.The pipeline-based architecture is the best choice for high throughput applications. Due to its regular structure and relatively simple control, it is the best choice to implement high-speed long size FFT.

Several pipelined architecture have been developed, such as Multi-path Delay Commutator (MDC), Single Path Delay Feedback (SDF) and Single Path Delay Commutator (SDC). SDF architecture can implement various FFT algorithms like radix-2, radix-4, radix-8 and other split radix. In the R2SDF structure, $\log_2$ 2N-2multipliers, 2log2N adders and N-1 shift registers are needed [13]-[14] . The output data order form this structure is bit-reversed.

ROM-Less FFT/IFFT Processor:

The architecture described uses a reconfigurable complex constant multiplier and bit-parallel complex multipliers is used instead of using ROM's to store twiddle factors, which is suited for the power-of-2 radix style of FFT/IFFT processors. The architecture described below is composed of three different types of processing elements (PEs), a complex constant multiplier, delay-line (DL) buffers (as shown by a rectangle with a number inside), and some extra processing units for computing IFFT. Here, the conjugate for extra processing units is easy to implement, which only takes the 2's complement of the imaginary part of a complex value. The divided-by-64 module can be substituted with a barrel shifter. In addition, for a complex constant multiplier, a novel reconfigurable complex constant multiplier is used to eliminate the twiddle-factor ROM.

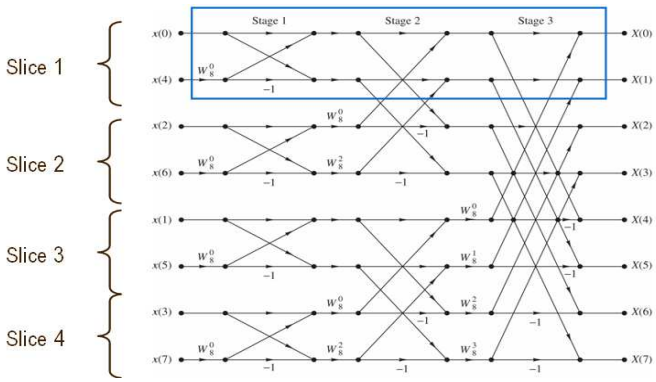

Fig. 1 Pipelined FFT

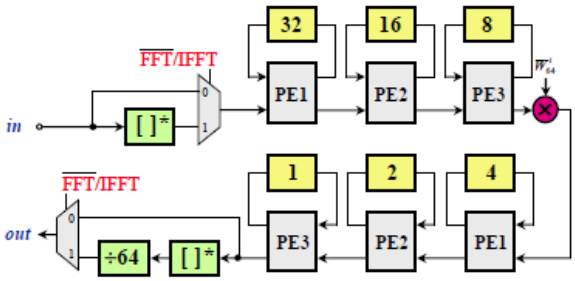

Fig. 2 Radix-2 64 Point FFT/IFFT Processor

There are 3 types of processing elements used in the design, the functions of which correspond to each of the butterfly stages. The PE3 stage is used to implement a simple radix-2 butterfly structure only, and serves as the sub modules of the PE2 and PE1 stages.

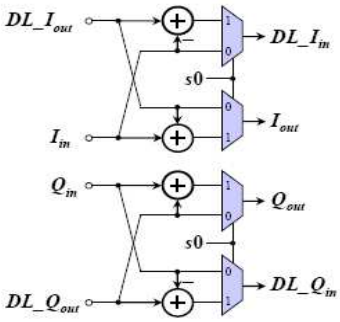

Fig. 3 Circuit Diagram of PE 3 Stage

The figures 3, 4 and 5 explains in detail about the structure of the processing elements used in figure 2. In the figure 3, Iin and Iout are the real parts of the input and output data, respectively. Qin and Qout denote the imaginary parts of the input and output data, respectively. The DL_Iin and DL_Iout stand for the real parts of input and output of the DL buffers, and DL_Qin and DL_Qout are for the imaginary parts, respectively.

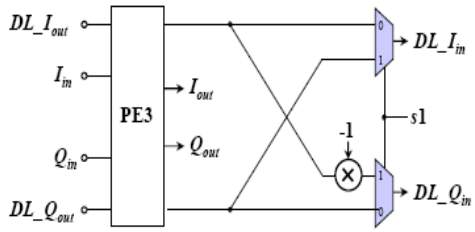

Fig. 4 Circuit Diagram of PE 2 Stage

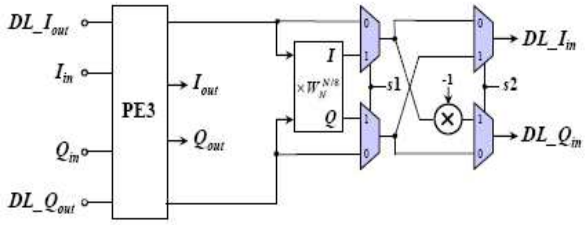

Fig. 5 Circuit Diagram of PE 1 Stage

Bit Parallel Mulripliers:

To replace the word length multiplier and square root evaluation for chip area reduction, the multiplication by $1/\sqrt{2}$ employs a bit parallel multiplier.

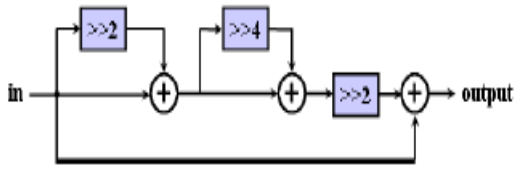

Fig. 6 Circuit Diagram of Bit-Parallel Multiplication

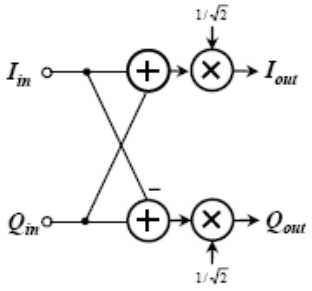

Fig.7 Circuit Diagram of multiplication by $W_N^{N/8}$

There are several bit level parallel multipliers available such as, carry ripple multiplier, carry save multiplier, Baugh Wooley multiplier and parallel multipliers with modified modified Booth recoding. The modification of the proposed design can be done by using any of these multipliers and achieving low power than the existing techniques.

Reconfigurable Complex Constant Multipliers:

The complex multiplier uses a cascaded scheme to achieve low-cost hardware. Here, the meaning of two input signals (Iin and Iout) and two output signals (Qin and Qout) are the same as the signals in the PE1 stage.

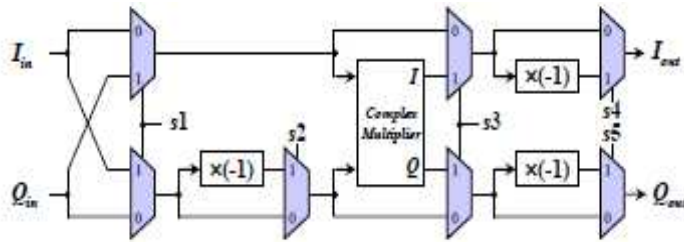

Figure 8. Reconfigurable Complex Constant Multiplier

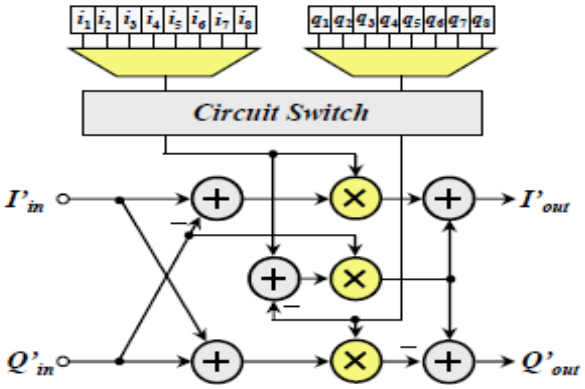

Figure 9. Complex Multiplier used in fig. 8

The circuit shown in figure 9 is responsible for the computation of multiplication by a twiddle factor W_{64}^{i}. The word-length multiplier used in Fig. 9 adopts a low-error fixed-width modified Booth multiplier for hardware cost reduction [16].

4. Simulation and Synthesis Results

The simulation of 64-point FFT was described in VHDL and the simulation was done in ModelSim and the code was functionally verified to be correct. .

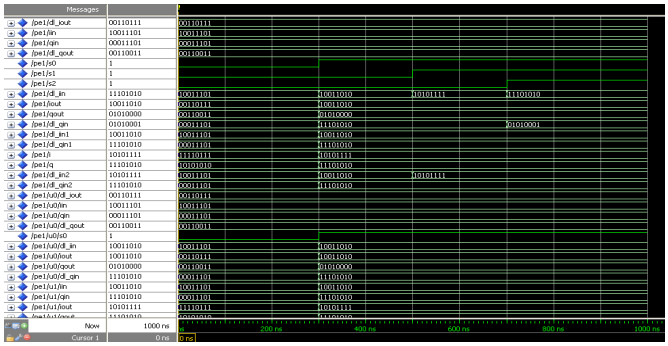

Figure 10: PE1 Simulation Result

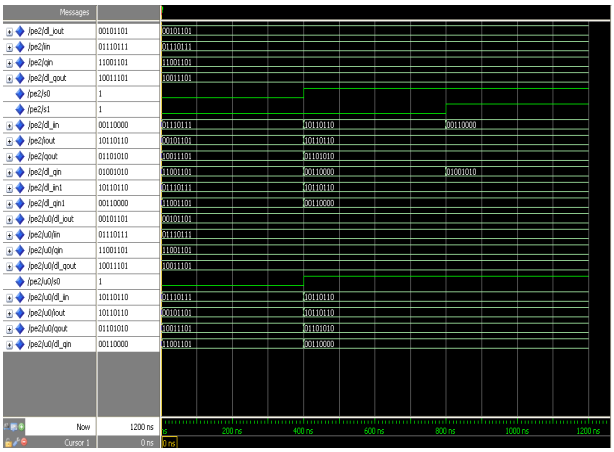

Figure 11. PE2 Stage Simulation Result

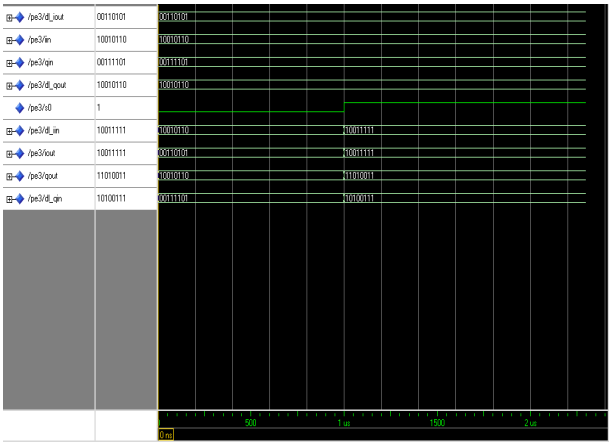

Figure 12. PE3 Stage Simulation Result

The simulation results of various processing elements are shown in the above figures 10, 11 and 12.

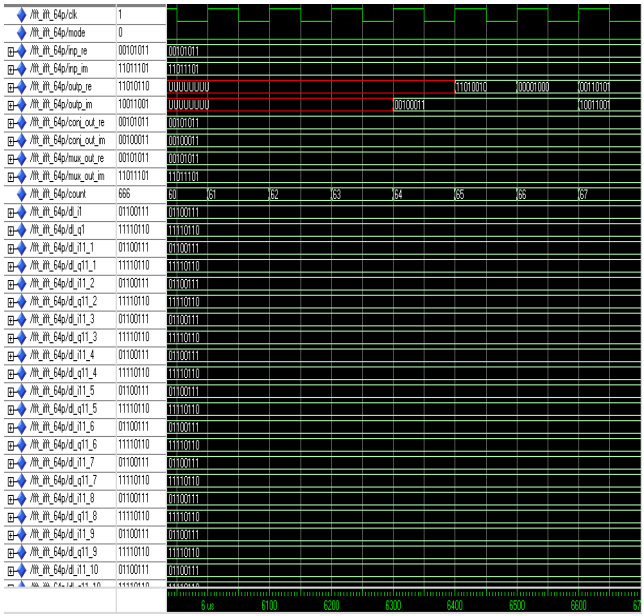

Figure 13. 64-Point FFT/IFFT Simulation Result

Synthesis Report Analysis:

The power and area analysis were done in Synopsys DC (Design Compiler) and was found that for a 64 point FFT, the total dynamic power and area specifications were met.

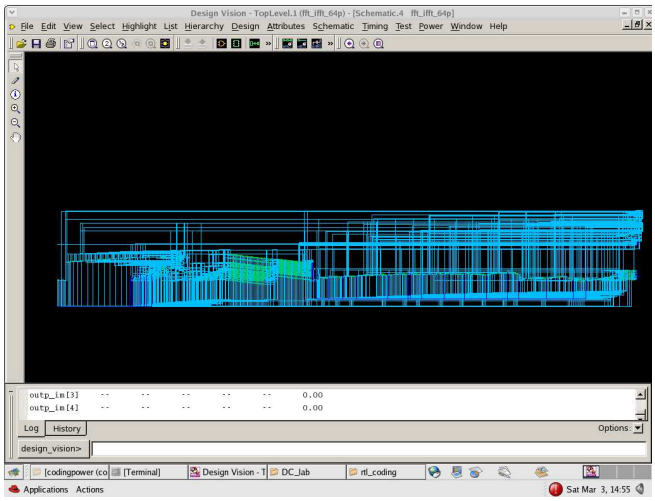

Figure 14. Schematic of 64-Point FFT/IFFT Processor

Synthesized Power Report:

The below shown table 2 describes the power analysis and the result was found that for a 64-point FFT processor. The total dynamic power was only 2.2876 mW and the cell leakage power was only 42.5143 uW. Compared with the work in [10], this design is about twice lower than its power consumption.

Table 2. Synthesized Power Report

S.No	Description	Power
1	Cell Internal Power	1.7046mW (75%)
2	Net Switching Power	582.9717uW (25%)
3	Total Dynamic Power	2.2876mW (100%)
4	Cell Leakage Power	42.5143uW

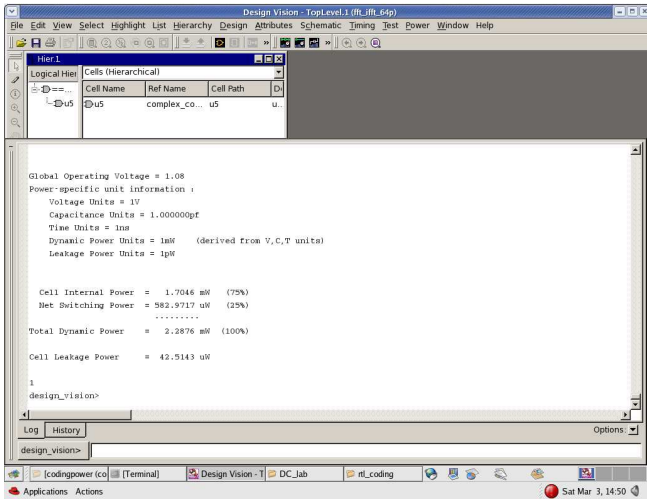

Figure 15. Synopsys DC Power Report

Table 3. Synthesized Area Report

S.No	Description	Area
1	Combinational area	1421
2	Non-Combinational area	5404.5
3	Net Interconnect area	769.82213
4	Total cell area	6825.5
5	Total area	7595.32213

5. Conclusion

A novel ROM-less and low-power pipeline 64-point FFT/IFFT processor is described in this work. Considering the symmetric property of twiddle factors in FFT, a reconfigurable complex constant multiplier is designed such that the size of twiddlefactor ROM is significantly shrunk, especially no ROM is needed. This result of 64-point FFT processor shows that our design meets the power, area and timing specifications. The future work can be extended to higher radix 64-point processor as proposed and the multipliers used can be varied to obtain a low power FFT processor.

References

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

[2] Koushik Maharatna, Eckhard Grass, and Ulrich Jagdhold, "A 64-Point fourier transform chip for high-speed wireless LAN application using OFDM," IEEE Journal of Solid-State Circuits, vol. 39, no. 3, pp. 484- 493, Mar. 2004.

[3] Chu Yu, Yi-Ting Liao, Mao-Hsu Yen, Pao-Ann Hsiung, and Sao-Jie Chen, "A Novel Low-Power 64-point Pipelined FFT/IFFT Processor for OFDM Applications," in Proc. IEEE Int'l Conference on Consumer Electronics. Jan. 2011, pp. 452-453.

[4] Chin-Teng Lin, Yuan-Chu Yu, and Lan-Da Van, "A low-power 64-point FFT-IFFT design for IEEE 802.11a WLAN application," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), May 2006, pp. 4523-4526

[5] Yuan Chen, Yu-Wei Lin, and Chen-Yi Lee, "A Block Scaling FFT/IFFT Processor for WiMAX Applications," in Proc. IEEE Asian Solid-state Circuits Conf., 2006, pp. 203-206.

[6] Sheng-Yeng Peng, Kai-Ting Shr, Chao-Ming Chen, Yuan-Hao Huang, "Energy-Efficient 128rv2048/1536-point FFT Processor with Resource Block Mapping for 3GPP-LTE system," in Proc. 2010 International Conference on Green Circuits and Systems (ICGCS), 2010, pp. 14-17.

[7] Minhyeok Shin and Hanho Lee, "A High-Speed Four-Parallel Radix-24 FFT/IFFT Processor for UWB Applications," in Proc. IEEE Int. Symp. Circuits and Systems, 2008, pp. 960-963.

[8] Jia Lihong, Gao Yonghong, Isoaho Jouni, and Tenhunen Hannu, "A new VLSI-oriented FFT algorithm and implementation," in Proc. IEEE International on ASIC Conference, Sept. 1998, pp. 337–341.

[9] Y.-W. Lin, H.-Y. Liu, and C.-Y. Lee, "A 1 GS/s FFT/IFFT processor for UWB applications," IEEE Journal of Solid-State Circuits, vol. 40, no. 8, pp. 1726-1735, Aug. 2005.

[10] Wen-Chang Yeh and Chein-Wei Jen, "High-speed and low-power splitradix FFT," IEEE Transactions on Signal Processing, vol. 51, no. 3, pp. 864- 874, Mar. 2003.

[11] Y. Jung, H. Yoon, and J. Kim, "New efficient FFT algorithm and pipeline implementation results for OFDM/DMT applications," IEEE Transactions on Consumer Electronics, vol. 49, no. 1, pp. 14-20, Feb. 2003.

[12] M. Hasan, T.Arslan, and J.S. Thompson, "A novel coefficient ordering based low power pipelined radix-4 FFT processor for wireless LAN applications," IEEE Transactions on Consumer Electronics, vol. 49, no. 1, pp. 128-134, Feb. 2003.

[13] S. He and M. Torkelson, "Designing Pipeline FFT Processor for OFDM (de)Modulation," in Proc. URSI Int. Sym. Signals, Systems, and Electronics, vol. 29, Oct.1998, pp. 257-262.

[14] H.L. Groginsky and G.A. Works, "A pipeline fast Fourier transform," IEEE Transactions on Computers, vol. C-19, no. 11, pp. 1015-1019, Nov. 1970.

[15] Y.T. Lin, P.Y. Tsai and T.D. Chiueh, "Low-power variable-length fast Fourier transform processor," IEE Proc. Comput. Digit. Tech., vol. 152, no. 4, pp. 499-506, July 2005.

[16] K. J. Cho, K. C. Lee, J. G. Chung, and K. K. Parhi, "Design of low-error fixed-width modified Booth multiplier," IEEE Trans. Very Large Scale Integration Systems, vol. 12, no. 5, pp. 522–531, May 2004.