Asian Journal of Computer Science and Technology
ISSN: 2249-0701 (P) Vol.3 No.1, 2014, pp.29-35

DOI: https://doi.org/10.51983/ajcst-2014.3.1.1728

FFT/IFFT Processor for Ultra Wide Band Application

R.Radhika

Department of Electronics and Communication Engineering,
Saveetha Engineering College, Chennai - 602 105, Tamil Nadu, India
E-mail: radhika14490@gmail.com
(Received on 25 December 2013 and accepted on 05 March 2014)

Abstract – **In this paper, a novel 128-Point FFT/IFFT processor for OFDM-based UWB system has been proposed. In proposed MRMDF (Mixed Radix Multipath Delayed Feedback) architecture, high throughput rate can be achieved by using four data paths. Furthermore, the hardware costs of memory and complex multiplier can be saved by adopting delay feedback and data scheduling approaches. In addition, the number of complex multiplications is reduced effectively by using a higher radix algorithm. The measurement results show that the throughput rate of this test chip is up to 1Gsample/s while it dissipates 175mW. When the throughput rate of our processor meets UWB standard, it only consumes 77.6 mW and multiplexers. The features of the proposed MRMDF architecture are the following:**

- **Higher throughput rate can be provided by using four parallel data paths;**
- **The minimum memory is required by using the delay feedback approach to reorder the input data and the intermediate results of each module.**
- **The 128-point mixed-radix FFT/IFFT algorithm is implemented to power consumption.**
- **The number of complex multiplier is minimized by using the scheduling scheme and the specified constant multipliers.**

Keywords: **Fast Fourier transform (FFT), orthogonal frequency division multiplexing (OFDM), ultra wideband (UWB)**

I. Introduction

UWB standard in which the FFT throughput rate is 409.6 Msample/s. A mixed radix algorithm is a combination of different radix-r algorithms. That is, different stages in the FFT computation have different radices. For instance, a 128-point long FFT can be computed in two stages using one stage with radix-8 PEs, followed by a stage of radix-2 PEs. This adds a bit of complexity to the algorithm compared to radix-r, but in return it gives more options in choosing the transform length.

128 point FFT/IFFT processor for UWB system radix-2 feedback (MRMDF), can provide a higher throughput rate by using the multipath-path scheme. Furthermore, the hardware costs of memory and complex multipliers in MRMDF are only 38.9% and 44.8% of those in the known FFT processor by means of the delay feedback and the data scheduling approaches. The high-radix FFT algorithm is also realized in our processor to reduce the number of complex multiplications. Power dissipation is 77.6 mW when its throughput rate meets FFT. However, this algorithm does not offer the simple bit reversing for ordering the output sequences.

II. Mixed Radix Algorithms

In this section we describe a generalization of the basic FFT that can be applied to a sample set where the size is any composite (non-prime) number .We shall refrain from describing an algorithm based on this idea in detail. Instead, only the mathematical basis of the algorithm will be described. The only real complexity in this algorithm is the indexing scheme, which is far more difficult than the simple bit reversing.

Mixed radix algorithms are used for signal processing; the basic idea is similar to the prime factor approach except that there is no constraint on the factors. The penalty paid is that complex-multiplications should be used while combining the results of small-point blocks instead of just re-ordering. Thus if N can be factored then the transform can be implemented. The small-points can be further factored

along the same lines. The algorithm involves point FFT computations, data re-ordering and complex multiplications in between. For the two-factor approach the number of real additions and real multiplications.

There are circumstances where the use of this method can bring worthwhile performance benefits. Sometimes you don't have the flexibility to design the system for radix 2 FFT's. In the worst case, forcing the use of a radix 2 algorithm will double the FFT size required. This loss of performance is compounded in multi-dimensional FFT's. Two dimensional image processing is a good example. Images tend to come in a variety of sizes. The worst case of doubling the row and column transform sizes will result a factor of 4 increases in the total FFT size.

Fourier analysis of finite-domain discrete-time signals are widely employed in signal processing and related fields to analyze the frequencies contained in a sampled signal, to solve operations such as convolutions. The DFT can be computed efficiently in practice using a Fast Fourier transform (FFT) algorithm.

Difference between DFT & FFT though FFT algorithms are so commonly employed to compute the DFT, there is a difference :"DFT" refers to a mathematical transformation, regardless of how it is computed, while "FFT" refers to any one of several efficient algorithms for the DFT**.**

1. The normalization factor multiplying the DFT and IDFT and the signs of the exponents are merely conventions.

2. A normalization of for both the DFT and IDFT makes the transforms unitary, which has some theoretical advantages.

3. The convention of a negative sign in the exponent is often convenient because it means that X_K is the amplitude of a "positive frequency" $2\pi k/ N$.The DFT is often thought of as a matched filter: when looking for a frequency of +1, one correlates the incoming signal with a frequency of −1.

A Fast Fourier Transform (FFT) is an efficient algorithm to compute the discrete Fourier transform(DFT) and its inverse. Let$x_0, ...,x_{N-1}$,be complex numbers. The DFT is defined by the formula

$$X_{K=} \sum_{n=0}^{N-1} x_n \, e \qquad kn \qquad k=0\ldots\ldots,N-1$$

This paper presents an energy-efficient single-chip, 1024-point Fast Fourier Transform (FFT) processor. The transistor design has been fabricated in a standard 0.7 m (L poly= 0:6 m) CMOS process and is fully functional on first pass silicon. At a supply voltage of 1.1 V, it calculates a 1024-point complex FFT in 330s. While consuming 9.5 m W, resulting in adjusted energy efficiency more than 16 times greater than the previously most efficient known FFT processor. At 3.3 V, it operates at 173 MHz which is a clock rate 2.6 times greater than the previously fastest rate.

The Fourier transform (FFT) is one of the most widely used digital signal processing (DSP) algorithms .While advances in semiconductor processing technology have enabled the performance and integration of FFT processors to increase steadily unfortunately, led to an increase in power consumption. This has resulted in a situation where the number of potential FFT applications that are limited by power not performance (e.g., portable applications) is significant and growing.

III. FFT Processor Memory System Architectures

As with most DSP algorithms, FFT's make frequent accesses to data in memory. FFT's are calculated in stages, where is the length of the transform and is the radix the FFT decomposition. Each stage requires the reading and writing of all data words.

1) Single Memory: The simplest memory-system architecture is the single memory architecture, as in which a memory of at least words is connected to a process or by a bidirectional data bus. In general, data are read from and written back to the memory once for each of the Stages of the FFT.

2) Dual Memory: The dual-memory architecture places two memories of size on separate buses connected to a processor. Data begin in one memory and "ping- pong" from memory to memory times until the transform has been calculated.

3) Pipeline: For processors using pipelined architecture, a series of smaller memories replace the word memory .Either physically or logically, there are stages processors and buffer memories are interleaved, as well as the flow of data through the pipeline structure.

Typically, an word memory is on one end of the pipeline, and memory sizes increase by through subsequent stages, with the final memory of size.

IV. High-Performance Operation

At V_{dd}=3.3V, the processor is fully functional at 173 MHz calculating a 1024-point complex FFT in 30 s while consuming 845 m W. While clock speed is not the only factor, it is certainly an important factor in determining the performance and area efficiency of a design compares clock speeds of this cached-FFT design running with other FFT processors versus their CMOS technologies. Despite having a favourable maximum clock rate, the chip's circuits are not optimized for high-speed operation in fact, nearly all transistors in logic circuits are near minimum size.

The computational complexity is through directly performing the required computation. By using the FFT algorithm, the computational complexity can be reduced to, where means the radix-r FFT. The radix-r FFT algorithm can be easily derived from the DFT by decomposing the -point DFT into a set of recursively related -point.

FFT transform, if is power of r. In general, higher-radix FFT algorithm has less number of complex multiplications compared with radix-2 FFT algorithm which is the simplest form in all FFT algorithms. In an example, for the 128-point FFT, the number of nontrivial complex multiplications of the radix-8 FFT algorithm is 152, which is only 59.3% of that of the radix-2 FFT algorithm. Thus, in order to save the number of complex multiplications, we choose the radix-8 FFT algorithm. Because the 128-point FFT is not a power of 8, the mixed-radix FFT algorithm, including radix-2 and radix-8 FFT algorithms, is needed.

V. Algorithm for DFT Computation

Given a sequence, an N-point discrete Fourier transform (DFT) is defined as

$$X(K)=\sum_{n=0}^{N} x(n) w_N^{Kn} \qquad K=0,1\ldots127$$

$$W_N^{nk=e^{-j(2\pi nk/N)}}$$

N=128, $n=64n_1+n_2$ $\{n_1=0,1 \quad n_2=0,1\ldots.63$

$k=k_1+2k_2$ $\{k_1=0,1 \quad k_2=0,1\ldots.63$

$$x(k_1+2k_2)=x(64n_1+n_2)w_{16}^{(2k2+k1)}$$

$x(64n1+n2)$

$$=BU_2\,(k_1,n_2)\,W_{64}^{n2k2}$$

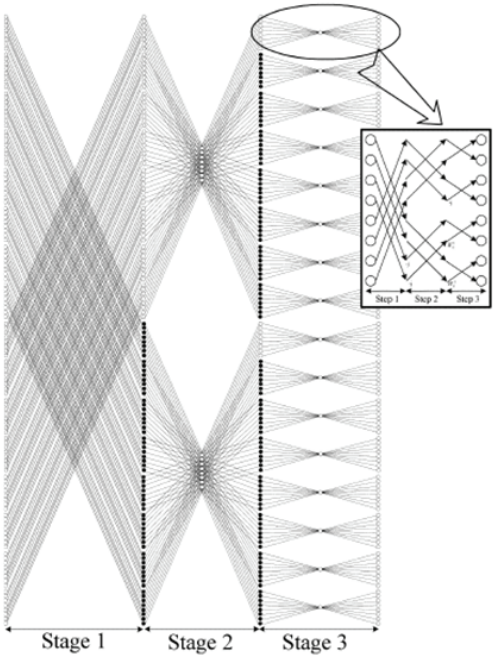

Fig.1 128-point mixed-radix FFT algorithm

The mixed-radix 4/2 butterfly unit uses both the radix-2^2 and the radix-2 algorithms can perform fast FFT computations and can process FFTs that are not power of four. The mixed-radix 4/2, which calculates four butterfly outputs based on X(0)~X(3). The proposed butterfly unit has three complex multipliers and eight complex adders. Four multiplexers represented by the solid box are used to select either the radix-4 calculation or the radix-2 calculation.

In order to verify the proposed scheme, 64-points FFT based on the proposed Mixed-Radix 4-2 butterfly with simple bit reversing for ordering the output sequences is exampled .For 64-points FFT is composed of total six-teen Mixed-Radix 4-2 Butterflies. In the first stage, the 64 point input sequences are divided by the 8 groups which correspond to n3=0, n3=1, n3=2, n3=3, n3=4, n3=5, n3=6, n3=7 respectively. Each group is input sequence for each Mixed-Radix 4-2 Butterfly. After the input sequences pass the first Mixed-Radix 4-2 Butterfly stage, the order of output value is expressed with small number below each butterfly output line. The proposed Mixed- Radix 4-2 is composed of two radix-4 butterflies and four radix-2 butterflies. In the first stage, the input data of two radix-4 butterflies which are expressed with the equation B4 (o, n3, k j) B4 (i, n3, k1), are grouped with the x(n3), x(N/4±n3), x(N/2±n3), x(3N/4±n3) and x(N/ 8±n3), x(3N/8±n3), x(5N/8±n3), x(7N/8±n3) respectively. After the each input group data passes the first radix-4 butterflies, the output data is multiplied by the

special twiddle factors. Then, these outputted sequences are inputted into the second stage which is composed of the radix-2 butterflies. After passing the second radix-2 butterflies, the outputted data are multiplied by the twiddle factors. These twiddle factors WQ (1+k) is the unique multiplier unit in the proposed Mixed- Radix 4-2 Butterfly with simple bit reversing the output sequences. Finally, we can also show order of the output sequences. The order of the output sequence is 0,4,2,6,1,5,3 and 7 which are exactly same at the simple binary bit reversing of the pure radix butterfly structure. Consequently, proposed mixed radix 4-2 butterfly with simple bit reversing output sequence include two radix 4 butterflies, four radix 2 butterflies, one multiplier unit and additional shift unit for special twiddle factors.

VI. Operand and Twiddle Factor Storage

The property of the FFT algorithms allows the input operands and intermediate and final results, to share the same memory locations. This, however, requires that the input sequence is stored into the RAM memory according to the input permutations. The in-place operand access during the actual computations can be realized with the aid of address rotation .In an N-point radix-2 FFT, there are N/2 different complex-valued twiddle factors. However, these factors can be computed from N/8+1 complex-valued numbers with the aid of an add/sub unit .On the other hand, there are N/4+1 different real-valued magnitudes present in the real and imaginary parts, thus the twiddle factors can be formed easily by fetching the magnitudes from a table and taking complement when needed. However, this arrangement requires that two memory accesses to the ROM table are needed. This is possible with the proposed pipelined butterfly units since the operand access takes two cycles. The previous method can be applied directly to butterfly units based on bit-parallel multipliers. The DA based butterfly units require sum and difference of the real and imaginary parts of the twiddle factors, thus an additional add/sub unit is needed. Butterfly units based on CORDIC need no ROMs since the angles can be generated easily from counters synchronizing the overall operation. In this paper, we have proposed a general organization for partial-column radix-2 and radix-2/4 FFT processors and described methods to improve the energy-efficiency of pipelined butterfly units. Several pipelined butterfly units have been synthesized onto a 0, 11 μm ASIC technology and the results show that butterfly units based obit-parallel multipliers are energy-efficient but cannot be used when high clock frequencies are used. The energy efficiency of these butterfly units can be further enhanced by utilizing radix-4 operation. The proposed radix-2/4butterfly unit to be the most energy-efficient choice, when the target clock frequency is high. Butterflies based on distributed arithmetic can support higher clock frequencies than the butterflies based on bit-parallel multipliers. When extremely long FFTs are needed, the pipelined CORDIC should be considered due to the fact that separate twiddle factor ROMs are not needed.

Mixed radix algorithm, another modified version of Cooley – Tuckey algorithm handle composite sizes. It behaves like FFT for any series that can be factored in factors 2, 3, 4, 5, 8 and 10. When there are other prime factors in the series, it will calculate the subseries with a complex Discrete Fourier Transform (DFT). The mixed-radix FFT is just as fast as normal FFT for power of 2 series, in all cases where the series can be composed of the factors above. In other cases it will be slightly slower. In various mixed radix architectures including mixed radix multipath delay commutation, mixed radix single path delay feedback and mixed radix combined delayed feedback commutation are to be dealt with.

The Mixed radix multipath delay commutation is a very crucial architecture for FFT computation, and it combines parallel pipelined concept and memory based architectures so as to obtain power efficiency. The processor, based on the multipath delay commutation architecture has high-radix arithmetic units with two main memories for input storage, inter mediate commutations, delays and output buffer for rescheduling purposes.

Here the 128 point FFT suited for UWB applications are calculated using four parallel 32 point FFT. The architecture shows the FFT computation for the 32 point FFT, which is decomposed into radix 4 FFT and radix2 FFT. The switches are more complex and it is implemented using multiplexers. All these operations are controlled by the control unit and the twiddle ROM is implemented using shift-add method of twiddle factor computation. The twiddle ROM computation is made by observing the unique twiddle factors and then mapping the values according to the mapping. The radix 4 is implemented using Carry Save Adder. It is found that Carry Save Adder, which is a fast adder, is having less delay compared to other adders and hence this adder is imported into the design to compute radix 4 algorithms. This design,

using mixed radix multipath delay commutation has four complex multipliers, so it consumes more multiplier power. To save the power consumption, a better implementation of the 128 point FFT is made by using the concept of single path delay feedback.

In this project mixed radix multipath delay feedback is used and to increase the throughput. Mixed radix algorithm is a combination of different radix –r algorithm, the input sequence and output sequence are in specified order. The order of the output sequence is one bit reversal of the order of the input signal. The 128 point mixed radix algorithm is implemented to save power consumption and the higher radix algorithm is used to reduce the number of complex multiplication.

In this section we describe a generalization of the basic FFT that can be applied to a sample set where the size is any composite (non-prime) number. We shall describe an algorithm based on this idea in detail. Instead, only the mathematical basis of the algorithm will be described. The only real complexity in this algorithm is the indexing scheme, which is far more difficult than the simple bit reversing.

Mixed radix algorithms are rarely used for signal processing. For most signal processing applications, your far better off designing your system with Radix 2 FFT's in mind (For example, chose your sampling frequency so that a Radix 2 FFT will give you the resolution you want from a spectrum analysis). The proposed MRMDF architecture combining the features of the SDF and MDC architectures consists of Module 1, Module 2, Module 3, conjugate blocks, a division block, and multiplexers.

In the MRMDF architecture, the input sequence and the output sequence are in the specified order. The order of the output sequence is the bit reversal of the order of the input sequence, and the operation of the FFT or IFFT is controlled by the control signal, FFT/IFFT when an IFFT is performed in our processor, the sign of the imaginary part of the input sequences will be changed and then they will be performed by the process in treating FFT. The sign of the imaginary part of output data from FFT will be changed again and then will be divided by 128. Because 128 is a power of two, the operation of the division is implemented by shifting the decimal point location. The function of Module 1 is to implement a radix-2 FFT algorithm, corresponding to the first stage of the SFG. Modules 2 and 3 are to realize the radix-8FFT algorithm, corresponding to the second and third stages of the SFG ,as displayed.in Modules 2 and 3 to implement the radix-8 FFT algorithm. In addition, the hardware complexity of the complex multiplier will be also considered in the proposed architecture.

1. Module 1: Module 1 consists of a register file which can store 64 pieces of complex data, one butterfly unit (BU),two complex multipliers, two ROMs, and some multiplexer. The function of ROM is used to store twiddle factors. Only period of cosine and sine waveforms are stored in ROM, and the other period waveforms can be reconstructed by these stored values. The BU consists of four BU_2s, which operate the complex addition and complex subtraction from two input data. Because the radix-2 FFT algorithm is adopted in this module, BU cannot start until both the input sequences and are available. This corresponds to the first stage of SFG.

The order of the four parallel input sequences in Module 1 is, and respectively, where is from. Therefore, these two available data of each data path are separated by 16 cycles if one input data of each path is available per clock cycle. At the first 16 cycles, the first 64 data are stored in the register file. At the next 16 cycles, the eight input data and of the BU are received from the register file and the input, respectively. Then the BU generates the outputs data according to the radix-2 FFT algorithm. Meanwhile, four output data, the BU, are fed to Module 2 directly, and the other four output data are stored into the register file. After 32 cycles, these data are read from the register file and are multiplied by the twiddle factors simultaneously before they are sent to Module 2. In general, four complex multipliers are needed in the four-parallel approach to implement the radix-2 FFT algorithm. Also, the utilization rate of the complex multiplier is only 50%. This paper proposes a new approach to increase the utilization rate and to reduce the number of complex multipliers. The detailed operation is described below. When these are generated by the BU, two are multiplied by the appropriate twiddle factors first before these are stored in the register file. After 32 clock cycles, other two and, are multiplied before the data's are fed to Module 2. By rescheduling the time of the complex multiplications, it is clear to find that only two complex multipliers are needed in our approach. The utilization of the complex multipliers can achieve100% by using our proposed approach.

2. Module 2: Module 2 consists of four BU_8 structures and

one modified complex multiplier. These four BU_8s operate in the same way. The architecture of BU_8 is direct mapped from the three-step radix-8 FFT algorithm. Also, the sizes of the three delay elements in the BU 8 are eight, four, and two points, respectively. The function of the delay element is to store the input data until the other available input data are received for the BU_2 operation. The output data generated by the BU_2 in the first step and second step are multiplied by a trivial twiddle factor, or before they are fed to the next step. These twiddle factors can be implemented efficiently. However, the four output data from the third step of the BU_8 need to be multiplied by the nontrivial twiddle factors simultaneously in the modified complex multiplier.

The scheduling of the twiddle factor in each data path after the twiddle factors are mapped to region A. It can be clearly seen that the twiddle factor of four paths in each time slot has different values, except for time slots 2 and 3. In time slots 2 and 3, the hardware conflict will happen if only one constant multiplier 4 is built. Therefore, an additional constant multiplier 4 is used in our design to avoid spending one more cycle. In the beginning, the four output sequences from the third step of the BU_8 are separated into real and imaginary parts. The data of each path are fed to appropriate constant multipliers according to the scheduling of the twiddle factor. Therefore, the entire constant multiplication calculation can be implemented by just using eight sets of constant values by swapping the real and imaginary parts appropriately and choosing the appropriate sign. The gate count of this approach can save about 38% compared to the four-complex-multiplier approach, and the performance of this approach is equivalent to that of the four complex multipliers.

3. Module 3: The radix-8 FFT algorithm is realized in Module 3. The structure of Module 3 is different from that of Module 2, because the two available data of the BU_2 in the second and third steps are in different data paths. Thus, a suitable structure is needed to ensure the correction of the FFT output data. Some output data, generated by the BU_2 in the first and second steps, are multiplied by the nontrivial twiddle factors before they are fed to the next step.

In general, the performance and hardware cost of the pipelined FFT architecture are increased by using the multiple data-path approach. Thus, the multipath-based architecture usually provides higher throughput rate with higher hardware cost if the parallel input data can be supported in this approach .The proposed MRMDF architecture hardware costs in terms of 128-point FFT are as follows:

- Registers number: 124.
- Complex multipliers: where the complexity of modified complex multiplier is only 62% of that of four complex multipliers
- Complex adders: 48.

compares the hardware requirement, FFT algorithm, and throughput rate with several classical and proposed approaches in the 128-point FFT. The known MDC architectures like R4MDC and the architecture proposed by Jung are not suitable for the 128-point FFT in UWB applications, because the FFT size used in their approaches is limited by a power of 4. In order that these two architectures are able to process the 128-point FFT, we modify both architectures by adding the proposed Module 1 to them. In addition, the throughput rate of the traditional MDC architecture is raised by increasing the utilization of butterfly units; this can be done by reordering the appropriate parallel input data in the input buffer before the data are loaded into the FFT processor. Consequently, the revised R4MDC and Jung's architectures. It should be emphasized that the input buffer, whose size is usually proportional to the number of data paths, is needed in all FFT processors listed in except for our proposed MRMDF architecture and R2 SDF architecture. By combining the features of the R2 SDF and the R4MDC approaches, the proposed FFT architecture not only can implement the radix-8 FFT algorithm in a 128-point FFT to reduce the number of complex multiplications but also can provide four times the throughput rate, compared with the R2 SDF scheme. In addition, the number so register excluding the input buffer and complex multiplier used in our scheme is only 38.9% and 44.8% of those in the SRMDC architecture. Although the number of complex adders in our design is greater than that in the others, the cost of complex adders is much less than that of registers and complex multipliers, respectively.

VII. Future Work

After the appropriate word length of the proposed FFT/IFFT processor is chosen, the architecture of the processor was modelled in Verilog and functionally verified using Verilog-XL simulator. The output data from the Verilog

coded architecture agreed with the output data of the FFT/ IFFT in UWB platform, which is coded by MATLAB. This test chip of the 128-point FFT/IFFT processor is fabricated in 0.18- m one-poly six-metal (1P6M) CMOS process. Input data are stored serially in the test module from the chip input pins before the operation of the processor. The test module provides four complex data in parallel to the FFT/ IFFT processor core when the processor begins to work. The highest throughput rate of our proposed architecture is up to 1 Gsample/s with power dissipation of 175 mW at 250 MHz. According to the specifications of UWB system, the throughput rate of the FFT/IFFT is 409.6 Msample/s. At the working frequency of 110 MHz, the power consumption of the FFT/IFFT processor is only 77.6 mW for 480 Mbs/s.

References

[1] A. Batraet *et. al.,* "Multi-Band OFDM Physical Layer Proposal for IEEE 802.15 Task Group 3a," IEEE P802.15-03/268r3, Mar. 2004.

[2] S. Magar, S. Shen, G. Luikuo, M. Fleming, and R. Aguilar, "An application specific DSP chip set for 100 MHz data rates," in Proc. Int.Conf. Acoustics, Speech, and Signal Processing, vol. 4, Apr. 1988, pp.1989–1992.

[3] S. He and M. Torkelson, "Designing pieline FFT processor for OFDM (de)modulation," in Proc. URSI Int. Symp. Signals, Systems, and Electronics, vol. 29, Oct. 1998, pp. 257–262.

[4] J. O'Brien, J. Mather, and B. Holland, "A 200 MIPS single-chip 1 k FFT processor," *in IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers,* vol.36, 1989, pp.

[5] B. M. Bass, "A low-power, high-performance, 1024-point FFT processor," *IEEE J. Solid-State Circuits,* vol. 34, no. 3, pp. 380–387, Mar.1999.

[6] L. R. Rabiner and B. Gold, Theory and Application of Digital SignalProcessing. Englewood Cliffs, NJ: Prentice-Hall, 1975.

[7] J. Garcia, J. A. Michel, and A. M. Burón, "VLSI configurable delay Commutation for a pipeline split radix FFT architecture," *IEEE Trans.SignalProcess.* vol. 47, no. 11, pp. 3098–3107, Nov. 1999.

[8] W.-C. Yeh and C.-W. Jen, "High-speed and low-power split-radix FFT," *IEEE Trans. Acoust., Speech, Signal Process,* vol. 51, no. 3, pp.864–874, Mar. 2003.

[9] L. Jia, Y. Gao, J. Isoaho, and H. Tenhunen, "A new VLSI-oriented FFT Algorithm and implement," in Proc. IEEE Int. ASIC Conf., Sep. 1998, pp. 337–341.

[10] K. Maharatna, E. Grass, and U. Jagdhold, "A 64-point fourier transform Chip for high-speed wireless lan application using OFDM," IEEE J. Solid-State Circuits, vol. 39, no. 3, pp. 484–493, Mar. 2004.

[11] Y. Jung, H. Yoon, and J. Kim, "New efficient FFT algorithm and Pipeline implementation results for OFDM/DM Tapplications," *IEEETrans.Consum. Electron.,* vol. 49, no. 1, pp. 14–20, Feb. 2003.