An Efficient FFT Engine With Reduced Addressing Logic

Xin Xiao, *Student Member, IEEE*, Erdal Oruklu, *Member, IEEE*, and Jafar Saniie, *Senior Member, IEEE*

***Abstract*—In this study, an improved butterfly structure and an address generation method for fast Fourier transform (FFT) are presented. The proposed method uses reduced logic to generate the addresses, avoiding the parity check and barrel shifters commonly used in FFT implementations. A general methodology for radix-2 N-point transforms is derived and the signal flow graph for a 16-point FFT is presented. Furthermore, as a case study, a 16-point FFT with 32-bit complex numbers is synthesized using a CMOS 0.18 μm technology. The circuit gate count analysis indicates that significant logic reduction can be achieved with improved throughput compared to the conventional implementations.**

***Index Terms*—Digital signal processing chips, fast Fourier transform, parallel addressing, parallel processing.**

I. Introduction

FAST FOURIER transform (FFT) is one of the fundamental operations in digital signal processing. A key challenge in many applications such as OFDM [1] is the hardware implementation of FFTs with large transform lengths ($>$1024 points). A typical FFT processor is composed of butterfly calculation units, an address generator, and memories. Different architectures have been proposed for improving the performance and reducing the complexity of the FFT hardware. Pipelined architectures are widely used in FFT realization [2]–[5] due to their speed advantages. Higher radix [4], [5] and multibutterfly [6], [7] structures can also improve the performance of the FFT processor significantly, but these structures require more hardware resources at the same time.

Shared-memory-based schemes with a single radix-2 butterfly calculation unit [8]–[12] are used in many embedded FFT processors since they require least amount of hardware resources, and the "in-place" addressing strategy is a practical choice to minimize the amount of memory. With the "in-place" strategy, the two outputs of the butterfly unit can be written back to the same memory locations of the two inputs, and replace the old data. Therefore, for in-place FFT processing, two data read and two data write operations occur at every clock cycle, necessitating a four-port memory structure. However, four-port memory blocks are costly and inefficient. In most cases, only two-port high-speed memories are available in hardware. Instead of using one bank of four-port memory, two banks of two-port memories can be used to realize the four data accesses in one clock cycle. In this case, a special addressing scheme is needed to avoid the data address conflict.

Cohen [8] introduced a simplified control logic for FFT address generation, which is composed of parity checks, barrel shifters, and counters based on the fact that two data addresses of every butterfly operation differ in their parity. Ma [9]–[11] proposed a method to realize the radix-2 addressing logic that reduces the address generation delay by avoiding parity check (XOR operations), but barrel shifters are still needed. Furthermore, Ma's approach is not "in-place," so more registers and related control logic are needed to buffer the interim data to avoid the memory conflict. Yang [5] proposed a locally pipelined radix-16 FFT realized by two radix-2 deep feedback ($\mathrm{R2SD^2F}$) butterflies. This architecture can improve the throughput of the FFT processing and reduce the complex multipliers and adders compared to other pipelined FFT methods, but it needs extra memory as buffers and there is significantly more coefficient access due to radix-16 implementation. The address generator needs two accumulators, in addition to logic shifters and switches. Li [7] proposed a mixed radix FFT architecture that contains one radix-2 butterfly and one radix-4 butterfly. The two butterflies share the multipliers, which reduce the hardware consumption, but the address generation is based on XOR logic, and is similar to Cohen's design. Wang [12] proposed a new method to reduce the memory reference, but the addressing scheme of this method is optimized specifically for DSP processors.

In this study, we present a hardware-efficient FFT engine with reduced addressing logic by using a butterfly structure that modifies the conventional one by adding exchange circuits at the input and output of the butterfly [13]. With the proposed architecture, the two inputs and two outputs of any butterfly can be exchanged; hence, all data and addresses in FFT processing can be reordered. Using this flexible input and output ordering, a reduced addressing logic is designed that does not need a barrel shifter and it is "in-place."

In the following sections, conventional radix-2 FFT implementations with enhancements to the butterfly unit by Cohen and Ma are described. We present the modified butterfly architecture and the improved address generation logic, which is primarily based on inverter, counter, and multiplexors. Although a shifter is still needed in this design, it shifts only once for each pass instead of each clock. Implementation results are shown and compared to the existing architectures.

Manuscript received December 26, 2007; revised April 01, 2008 and June 09, 2008. Current version published December 10, 2008. This paper was recommended by Associate Editor P. K. Meher.

The authors are with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL 60616 USA (e-mail: xxiao2@iit.edu; erdal@ece.iit.edu; sansonic@ece.iit.edu).

Digital Object Identifier 10.1109/TCSII.2008.2004540

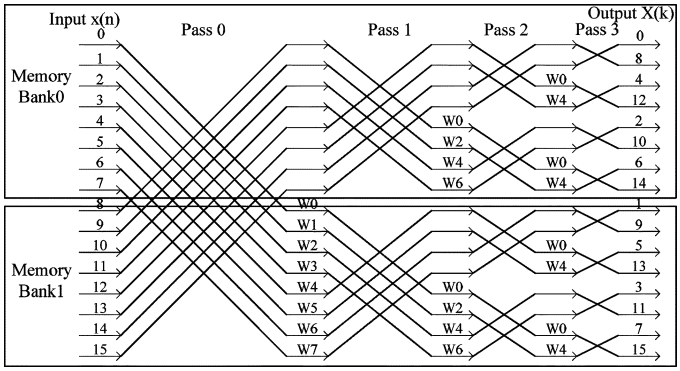

Fig. 1. Signal flow graph of a 16-point FFT.

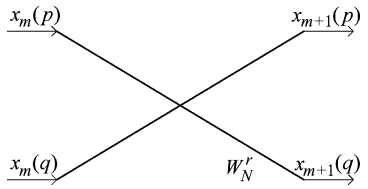

Fig. 2. Butterfly unit at pass m.

II. Radix-2 FFT and Address Generation Logic

The N-point discrete Fourier transform is defined by

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, \quad k = 0, 1, \ldots, N-1, \quad W_N^{nk} = e^{-j\frac{2\pi}{N}nk}. \tag{1}$$

Fig. 1 shows the signal flow graph of a 16-point decimation-in-frequency (DIF) radix-2 FFT.

The FFT algorithm is composed of butterfly calculation units

$$x_{m+1}(p) = x_m(p) + x_m(q) \tag{2}$$

$$x_{m+1}(q) = [x_m(p) - x_m(q)]\, W_N^r \tag{3}$$

Equations (2) and (3) describe the radix-2 butterfly calculation at pass m, as shown in Fig. 2. Parallel and "in-place" butterfly operation using two memory banks of two-port memory units requires that the two inputs of any butterfly are read from different banks of memory and the two outputs are written to the same address locations as the inputs.

As shown in Fig. 1, in the conventional FFT addressing scheme, only the butterflies in the first pass satisfy this requirement. Two inputs and two outputs of butterfly operations in all other passes are originating from and sinking to the same memory bank. Therefore, a special addressing scheme is required to prevent the conflicting addresses.

Cohen [8] uses parity check to separate the data into two memory banks. Fig. 3 is the signal flow graph of Cohen's approach and it shows that inputs and outputs of any butterfly pass utilize separate memory banks. The addresses of butterfly operations are "in-place" located. The drawback of Cohen's method is the address generation delay.

In order to reduce the delay of the address generation, Ma [9] proposed an alternative addressing scheme that avoids using parity check. The signal flow graph of Ma's scheme is shown in Fig. 4. In Ma's scheme, two inputs of a butterfly unit originate

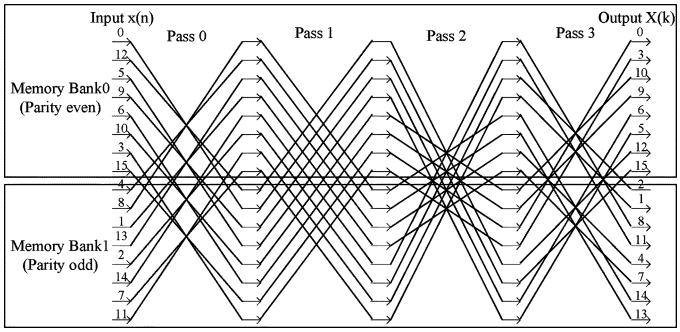

Fig. 3. Signal flow graph of a 16-point FFT using Cohen's method.

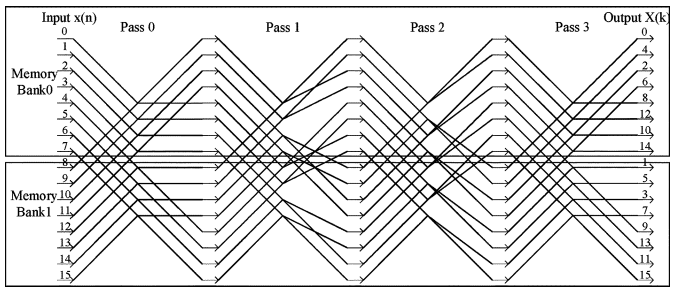

Fig. 4. Signal flow graph of a 16-point FFT using Ma's method.

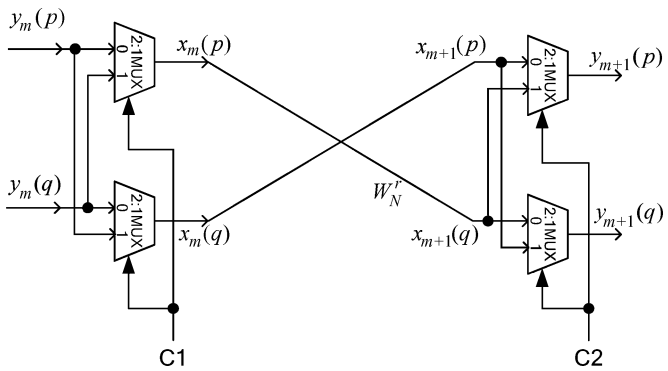

Fig. 5. Proposed butterfly structure.

from two separate memory banks but two outputs of the butterfly unit utilize the same memory bank, and the inputs and outputs of a butterfly unit are not "in-place" located. Therefore, extra registers and related control logic are needed to buffer the outputs of the butterfly until the next butterfly calculation is finished in order to realize the "in place" operation. Compared to Cohen's approach, which uses both parity check and barrel shifters, Ma's method needs only barrel shifters and avoids parity check resulting in a reduced address generation delay. However, Ma's approach consumes more hardware resources to realize the "in-place" operation.

III. Reduced Address Generation Logic for FFT

The goal of this study is to reduce both the address generation delay and the hardware complexity. The new addressing scheme is based on a modified butterfly structure, which is shown in Fig. 5. The main difference between the proposed butterfly structure and the conventional one is the two exchange circuits that are placed at both the input and the output of the

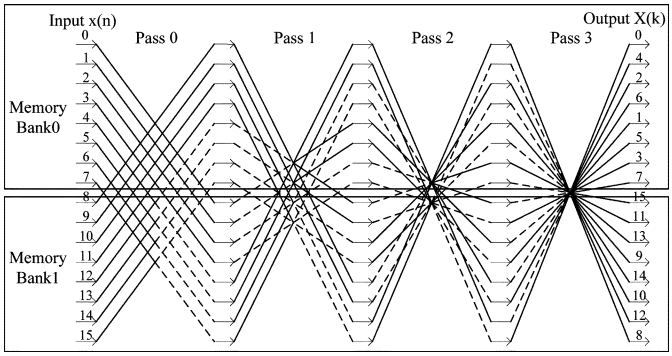

Fig. 6. Signal flow graph of a 16-point FFT using the proposed method.

butterfly unit. Each exchange circuit is composed of two (2:1) multiplexers; when the exchange control signal $C1$ or $C2$ is 1, the data will be exchanged; otherwise, they keep their locations.

The following equation shows the function

$$\begin{aligned}
&\text{If } C1 = 1: \\
&\qquad x_m(p) = y_m(q), \quad x_m(q) = y_m(p) \\
&\text{else}: \\
&\qquad x_m(p) = y_m(p), \quad x_m(q) = y_m(q). \\
&\text{If } C2 = 1: \\
&\qquad y_{m+1}(p) = x_{m+1}(q), \quad y_{m+1}(q) = x_{m+1}(p) \\
&\text{else}: \\
&\qquad y_{m+1}(p) = x_{m+1}(p), \quad y_{m+1}(q) = x_{m+1}(q). \qquad (4)
\end{aligned}$$

Based on this butterfly structure, all data within the FFT processing can be reordered by setting different values of the exchange control signals $C1$ and $C2$. The control signals are chosen such that the input data always originate from two separate memory banks and output data are written to the same memory location in order to achieve in-place operation.

A. 16-Point FFT Implementation

For a 16-point FFT, the signal flow graph of the proposed approach is shown in Fig. 6. In Fig. 6, the butterfly inputs or outputs indicated by broken lines denote that the data have been exchanged. Fig. 7 shows the complete address generation architecture for a 16-point FFT implementation. The address generation logic is composed of one 5-bit counter D, three inverters, one 3-bit shifter, three (2:1) multiplexers, two (4:1) multiplexers, four multibit (2:1) multiplexers and delay elements. $Pass\ Counter\ P$ indicates which pass of FFT is currently in progress and controls the two (4:1) multiplexers to generate the correct exchange control signal $C1$ and $C2$ for the butterfly operation. The 3-bit shifter shifts one bit at each pass and it controls three (2:1) multiplexers to generate the correct $M1$ address. Since the proposed technique is "in-place," the addresses for read and write are the same except for a delay introduced for compensating the butterfly computation time. Table I presents the counter values that are used to generate the addresses for $M0$ and $M1$ memory banks.

B. N-Point FFT Implementation

In order to generalize the addressing scheme for $N = 2^r$-point FFT, the necessary circuit components of the addressing and control logic can be listed as follows:

1) $(r-1)$-bit $Butterfly\ Counter\ B = b_{r-2}b_{r-3}\ldots b_1b_0$;
2) $(r-1)$ inverters that generate the complement of the $Butterfly\ Counter\ \overline{B} = \overline{b}_{r-2}\overline{b}_{r-3}\ldots\overline{b}_1\overline{b}_0$ from counter B;
3) $\lceil \log_2 r \rceil$-bit $Pass\ Counter\ P = (r-1), \ldots, 2, 1, 0$;
4) Two memory banks, $Bank\ 0\ (M0)$ and $Bank\ 1\ (M1)$.

In practice, $Pass\ Counter\ P$ and $Butterfly\ Counter\ B$ can be combined to a single $counter\ D$, where B is the least significant $(r-1)$ bits of $counter\ D$, and P is the most significant $\lceil \log_2 r \rceil$ bit of $counter\ D$. At any time, the read and write addresses of $M0$ is exactly the same as the value of $Butterfly\ Counter\ B$. For $M1$, the read and write address at Pass s is $\overline{b}_{r-2}\overline{b}_{r-3}\ldots\overline{b}_{r-s-1}b_{r-s-2}\ldots b_1b_0$, which is a combination of counters B and $\overline{B}$. The exchange control signal $C1$ is equal to b_{r-s-1} (assume $b_{r-1} \equiv 0$), and $C2$ is equal to b_{r-s-2} (assume $b_{-1} \equiv 0$). The address of twiddle factors at pass s is given by $b_{r-s-2}b_{r-s-3}\ldots b_0 0\ldots 0$ (s "0"s).

IV. Implementation and Results

The proposed FFT algorithm is synthesized using TSMC CMOS 0.18 μm technology. Synthesis is performed with Cadence build gates and encounter tools. The synthesis results for a 16-point FFT with 32-bit complex number inputs show a maximum clock frequency of 280 MHz with 0.665 mm^2 area and 0.645 mW total power consumption for the complete FFT operation, including butterfly unit, address generation unit, and memory circuits.

In order to compare different FFT addressing methods, the logic complexity is evaluated similar to [9] with the sizes of some basic circuits and gates listed in Table II. Estimated gate count comparison for a 1024-point FFT of 32-bit complex data (16-bit each for the real part and imaginary part) is shown in the Table III. In terms of area, the proposed scheme requires 24% fewer number of transistors. This reduction is mainly due to the difference in logic complexity of the multiplexers and barrel shifters. Based on the gate counts in Table II (and confirmed by synthesis results), r-input (r:1) multiplexer is approximately four times smaller than the $(r-1)$ barrel shifter in terms of area.

The delay of address generation for both read and write operations in the proposed scheme is determined by two stages of multiplexers, where the first stage uses an r-input (r:1) multiplexer and the second stage involves a 2-input (2:1) multiplexer for a 2^r-point FFT operation (see Fig. 7).

In [9], the worst case address generation delay is dominated by an $(r-1)$-bit barrel shifter and a (2:1)-multiplexer. An $(r-1)$-bit barrel shifter requires $\lceil \log_2(r-1) \rceil$ stages of (2:1) multiplexers in the critical path. Cohen's address generation method [8] uses an r-bit parity check unit, an $(r-1)$-bit barrel shifter, and two (2:1) multiplexers in the critical path. Standard cell synthesis results in Table IV show that the proposed address generation scheme is faster compared to that of Cohen[8] and Ma[9]

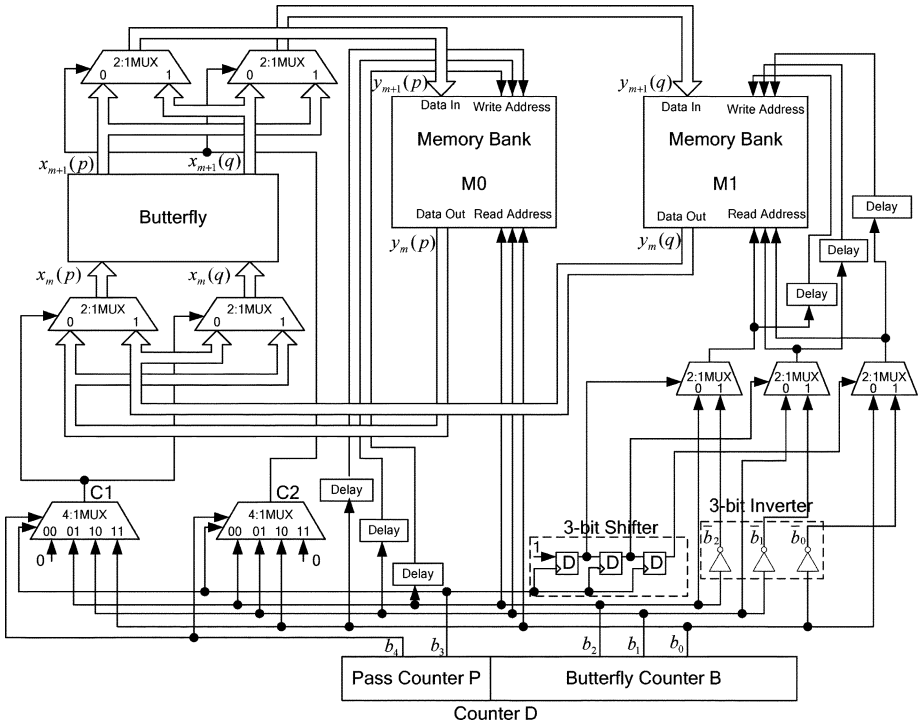

Fig. 7. Address generation circuits for a 16-point FFT.

TABLE I
ADDRESS GENERATION TABLE OF THE PROPOSED METHOD FOR A 16-POINT FFT

Counter $B(b_2b_1b_0)$	Counter $\overline{B}(\overline{b}_2\overline{b}_1\overline{b}_0)$	Pass 0 (exchange control signal: C1=0,C2=b_2)		Pass 1 (exchange control signal: C1=b_2,C2=b_1)		Pass 2 (exchange control signal: C1=b_1,C2=b_0)		Pass 3 (exchange control signal: C1=b_0,C2=0)	
		Bank0 address $b_2b_1b_0$	Bank1 address $b_2b_1b_0$	Bank0 address $b_2b_1b_0$	Bank1 address $\overline{b}_2b_1b_0$	Bank0 address $b_2b_1b_0$	Bank1 address $\overline{b}_2\overline{b}_1b_0$	Bank0 address $b_2b_1b_0$	Bank1 address $\overline{b}_2\overline{b}_1\overline{b}_0$
000	111	000	000	000	100	000	110	000	111
001	110	001	001	001	101	001	111	001	110
010	101	010	010	010	110	010	100	010	101
011	100	011	011	011	111	011	101	011	100
100	011	100	100	100	000	100	010	100	011
101	010	101	101	101	001	101	011	101	010
110	001	110	110	110	010	110	000	110	001
111	000	111	111	111	011	111	001	111	000

TABLE II
TRANSISTOR COUNTS FOR CMOS CELLS [9]

Types of Gates and Circuits	No. of. Transistors
2-Input XOR	10
2-1 Multiplexer	6
10-1 Multiplexer	42
1-bit Register/Latch	10
9-bit Counter	182
13-bit Counter	270
9-bit Barrel Shifter	152
10-bit Barrel Shifter	168

for large FFTs, due to the complex wiring and parasitic capacitances in barrel shifters and elimination of the parity-check operation.

Compared to a pipelined FFT architecture such as R2SD^{2}F given in [5], the proposed shared memory architecture offers significantly reduced hardware cost and power consumption at the expense of (slower) throughput. R2SD^{2}F requires $\log_4 N - 1$ multipliers, $2\log_4 N$ adders, and $10\log_4 N$ multiplexers for the butterfly operations in an N-point FFT. In contrast, only one multiplier, two adders, and four multiplexers are used in the proposed FFT architecture datapath. The latency (total clock cycles) of a pipelined FFT architecture is faster by a factor of $(1/2)\log_2 N$ but the maximum achievable clock frequency would be less than the proposed design due the increased complexity of the R2SD^{2}F datapath and address generation. For low-power embedded applications, the reduced logic shared memory FFT approach in this paper presents a more viable solution.

TABLE III
ADDRESS GENERATION LOGIC COMPARISON FOR A 1024-POINT FFT WITH 32-BIT COMPLEX DATA

Design Schemes	Components		Transistor Counts
	Quantity	Type	
Proposed Design	1	13-bit Counter	1562
	9	Inverters	
	1	9-bit Shifter	
	9	1-bit 2:1 Multiplexer	
	2	1-bit 10:1 Multiplexer	
	4	32-bit 2:1 Multiplexer	
	2	9-bit Latches	
Ma's Design	1	13-bit Counter	2066
	2	9-bit Barrel Shifters	
	4	9-bit Latches	
	2	32-bit Latches	
	2	9-bit 2:1 Multiplexers	
	2	32-bit 2:1 Multiplexers	
Cohen's Design	1	13-bit Counter	1924
	1	9-bit Counter	
	2	9-bit Latch	
	2	10-bit Barrel Shifter	
	2	9-bit 2:1 Multiplexer	
	4	32-bit 2:1Multiplexer	
	1	9-bit Address Parity Generator	

TABLE IV
DELAY COMPARISON OF ADDRESS GENERATION CIRCUITS

FFT size $=2^r$	Proposed Method	Ma's Method [9]	Cohen's Method [8]
r=4	1.28 ns	1.28 ns	1.82 ns
r=8	1.40 ns	1.53 ns	2.50 ns
r=10	1.47 ns	1.71 ns	2.61 ns
r=16	1.59 ns	1.85 ns	2.87 ns

V. CONCLUSION

FFT is a fundamental tool for many signal processing and communication applications. Therefore, several methods have been proposed to realize conflict-free memory addressing of FFT. These methods reorder the addresses of the butterfly inputs and outputs to realize the parallel accessing of the memory. In this paper, we introduce a reduced addressing structure by building a butterfly unit whose two outputs can be exchanged to allow in-place addressing and to minimize the addressing logic. The synthesis simulation results and analysis confirm a highly efficient FFT memory addressing scheme with significant logic reduction and delay improvements compared to existing shared-memory-based FFT methods.

REFERENCES

[1] R. M. Jiang, "An area-efficient FFT architecture for OFDM digital video broadcasting," *IEEE Trans. Consum. Electron.*, vol. 53, no. 4, pp. 1322–1326, Nov. 2007.
[2] W. D. Li and L. Wanhammar, "A pipeline FFT processor," in *Proc. IEEE Workshop Signal Process. Syst.*, Oct. 1999, pp. 654–662.
[3] S. S. He and M. Torkelson, "A new approach to pipeline FFT processor," in *Proc. 10th Int. Parallel Process. Symp.*, Apr. 1996, pp. 766–770.
[4] T. M. Hopkinson and G. M. Butler, "A pipelined, high-precision FFT architecture," in *Proc. 35th Midwest Symp. Circuits Syst.*, Aug. 1992, vol. 2, pp. 835–838.
[5] L. Yang, K. Zhang, H. Liu, J. Huang, and S. Huang, "An efficient locally pipelined FFT processor," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 53, no. 7, pp. 585–589, Jul. 2006.
[6] S. Bouguezel, M. O. Ahmad, and M. N. S. Swamy, "A new radix-2/8 FFT algorithm for length-$q \times 2^m$ DFTs," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 9, pp. 1723–1732, Sept. 2004.
[7] X. Li, Z. Lai, and J. Cui, "A low power and small area FFT processor for OFDM demodulator," *IEEE Trans. Consum. Electron.*, vol. 53, no. 2, pp. 274–277, May 2007.
[8] D. Cohen, "Simplified control of FFT hardware," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-24, no. 6, pp. 577–579, Dec. 1976.
[9] Y. Ma, "An effective memory addressing scheme for FFT processors," *IEEE Trans. Signal Process.*, vol. 47, no. 3, pp. 907–911, Mar. 1999.
[10] Y. Ma, "A fast address generation scheme for FFT processors," *Chin. J. Comput.*, vol. 17, no. 7, pp. 505–512, Jul. 1994.
[11] Y. Ma and L. Wanhammar, "A hardware efficient control of memory addressing for high-performance FFT processors," *IEEE Trans. Signal Process.*, vol. 48, no. 3, pp. 917–921, Mar. 2000.
[12] Y. Wang, Y. Tang, Y. Jiang, J. Chung, S. Song, and M. Lim, "Novel memory reference reduction methods for FFT implementations on DSP processors," *IEEE Trans. Signal Process.*, vol. 55, no. 5, pp. 2338–2349, May 2007.
[13] X. Xiao, E. Oruklu, and J. Saniie, "Efficient FFT engine with reduced addressing logic," in *Proc. IEEE Int. Conf. Electro/Inf. Technol.*, May 2007, pp. 390–395.