Area-Efficient Memory-Based Architecture for FFT Processing

Sang-Chul Moon and In-Cheol Park

Dept. of EECS, KAIST, Daejeon, Korea

ABSTRACT

In this paper, we propose a new area-efficient parallel architecture to calculate 2^n-point FFT. The proposed architecture is based on the radix-4 Cooley-Tukey algorithm, and consists of four complex multipliers, eight complex adders, and four RAMs each of which is partitioned into two banks. The implemented FFT processor can calculate 2K/4K/8K-point complex FFT in 28.2μs/62.0μs/135.2μs at 91Mhz, respectively.

1. INTRODUCTION

Orthogonal Frequency Division Multiplexing (OFDM) is one of the most suitable protocols for wideband communication because of its bandwidth efficiency and robustness[1][2]. In the implementation of wideband OFDM systems such as digital terrestrial video transmission (DVB-T)[3], a large-point FFT processor is one of the key components as it requires 2K/8K-point DFT in Europe standard and 4K-point DFT in Japan standard.

In the implementation of a long-point FFT processor, two architectures, pipelined and memory-based ones, are commonly used. The pipelined architecture consumes a relatively large chip area compared with the memory-based architecture because the former needs many complex adders and complex multipliers[4][5][6]. On the other hand, the radix-2 memory-based architecture cannot meet the throughput requirement because of its long latency, and the radix-4 architecture cannot deal with 2K/8K-point DFT as it calculates only 4^n-point DFT.

In this paper, we propose a parallel FFT architecture that can calculate 2K/4K/8K-point FFT with the latency of radix-4 computation. To achieve a parallelized FFT algorithm, one-dimensional N-point FFT is converted into N/4×4 two-dimensional FFT using Cooley-Tukey algorithm[7]. An area-efficient and high-performance FFT processor is designed based on the two-dimensional algorithm.

The rest of this paper is organized as follows. We discuss the proposed algorithm in Section 2. In Section 3, we describe the architecture of the processor. After showing performance and synthesis results in Section 4, conclusions are addressed in Section 5.

2. ALGORITHM

N-point discrete Fourier transform is defined as

$$X[k]=\sum_{n=0}^{N-1}x(n)W_N^{nk},\quad k=0,1,\ldots,N-1 \qquad (1)$$
$$with\ \ W_N^r=e^{-j(2\pi r/N)}$$

To achieve the same throughput as radix-4 FFT calculation and to reduce the resource requirement, a two-dimensional FFT algorithm is derived using Cooley-Tukey algorithm, as described below.

$$X[k]=\sum_{n_2=0}^{3}\left[\left(\sum_{n_1=0}^{N/4-1}x[4n_1+n_2]W_{N/4}^{k_1n_1}\right)W_N^{k_1n_2}\right]W_4^{k_2n_2}$$

$$n=3\cdot n_1+n_2,\qquad \begin{cases}0\le n_1\le N/4-1\\ 0\le n_2\le 3\end{cases} \qquad (2)$$

$$k=k_1+(N/4-1)\cdot k_2,\quad \begin{cases}0\le k_1\le N/4-1\\ 0\le k_2\le 3\end{cases}$$

This two-dimensional discrete Fourier transform consists of N/4-point DFT and 4-point DFT as shown in Fig. 1.

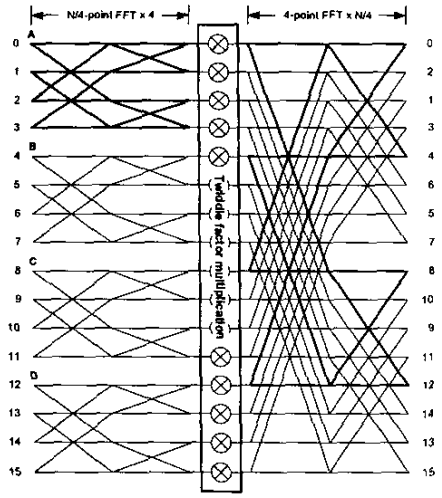

Figure 1 Proposed FFT algorithm for 16-point FFT

The proposed algorithm can compute the N-point FFT in three phases. At the first phase, four N/4-point FFTs are calculated, i.e., A, B, C and D in Fig. 1. Twiddle factors are multiplied to the results of N/4-point FFTs at the second phase. Finally, 4-point FFT is calculated N/4 times at the third phase.

3. ARCHITECTURE

In this Section, we describe the architecture of the proposed FFT processor and the memory addressing method.

0-7803-7761-3/03/$17.00 ©2003 IEEE

3.1 FFT Processor

The proposed FFT processor consists of a calculation unit and four 2K RAMs each of which is partitioned into two banks to provide two data simultaneously. The calculation unit has four complex multipliers and eight complex adders. The block diagram of the proposed FFT processor is shown in Fig. 2.

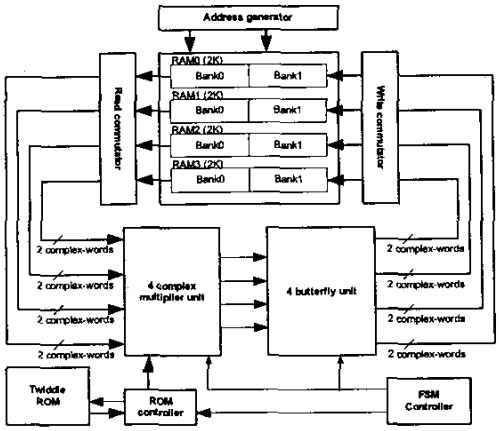

Figure 2 Block diagram of the proposed FFT processor

This FFT processor performs the proposed FFT algorithm in two steps.

At the first step, four N/4-point FFTs are calculated parallelly. At this moment, eight complex data needed for four butterfly operations are read from eight banks of four RAMs using two addresses.

At the second step, twiddle factor multiplication and 4-point FFT can be performed at once because the 4-point FFT does not need any multiplier. In this step, the calculation unit accesses four complex data using one read address. The data read from RAMs are multiplied by twiddle factors and fed into the butterfly unit. The second step is performed N/4 times.

Table I The performance comparison of radix-4 and proposed algorithm

	Radix-4	Proposed
Number of calculations	$\frac{N}{4}(\log_4 N)$	$\frac{N}{8}(\log_2 N-2)+\frac{N}{4}$

The proposed architecture can calculate the N-point FFT with latency of radix-4 FFT as shown in Table I. In addition, the proposed architecture can calculate 2^n-point FFT by adjusting the number of points performed in the first step, where as the radix-4 architecture can calculate only 4^n-point FFT.

The calculation unit of the proposed FFT processor consists of eight complex adders and four complex multipliers units, namely the butterfly unit and the multiply unit, respectively. To perform all the step of N/4×4 two-dimensional FFT with a calculation unit, the butterfly unit and the multiply unit are separated and connected to each other in an appropriate way in each step of the proposed algorithm. This separation enables the pipelining of the two units, and thus helps enhance the performance of the processor through the pipelining.

3.2 Step 1: Parallel Computation

The proposed FFT processor calculates four N/4-point FFTs in parallel at the first step of N-point FFT calculation. The calculation unit uses four radix-2 butterflies and four complex multipliers at this step as shown in Fig. 3.

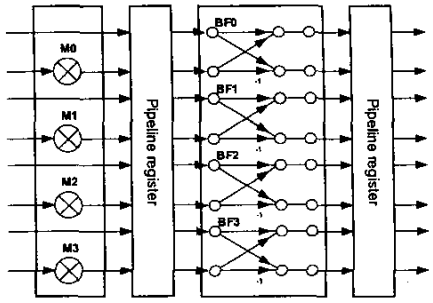

Figure 3 The calculation unit with the parallel computation

Each RAM stores the data of an N/4-point FFT and all the N/4-point FFTs are independent of one another. Therefore it is possible to concurrently process four N/4-point FFTs shown in equation (3).

$$\begin{aligned} G[0,k_1] &= \sum_{n_1=0}^{N/4-1} x[4n_1]W_{N/4}^{k_1 n_1} \\ G[1,k_1] &= \sum_{n_1=0}^{N/4-1} x[4n_1+1]W_{N/4}^{k_1 n_1} \\ G[2,k_1] &= \sum_{n_1=0}^{N/4-1} x[4n_1+2]W_{N/4}^{k_1 n_1} \\ G[3,k_1] &= \sum_{n_1=0}^{N/4-1} x[4n_1+3]W_{N/4}^{k_1 n_1} \end{aligned} \tag{3}$$

3.3 Step 2: 4-point FFT Computation

In the second step, the calculation unit performs twiddle factor multiplications and 4-point FFT computations shown in equation (4) and (5).

$$G'[n_2,k_1] = G[n_2,k_1]W_N^{k_1 n_2} \tag{4}$$

$$X[4k_2+k_1] = \sum_{n_2=0}^{3} G'[n_2,k_1]W_4^{k_2 n_2} \tag{5}$$

The data used in this calculation are accessed from four RAMs with one address. The twiddle factor multiplication is performed by the multiply unit firstly, and then the 4-point FFT is performed by the butterfly unit. The block diagram corresponding to the second step is shown in Fig. 4.

As the 4-point FFT is performed by the butterfly unit without any complex multiplier, the butterfly unit equipped with eight

complex adders and some glue logic for –j multiplication can perform the 4-point FFT. Hence, the calculation unit consisting of a multiply unit and a flexible butterfly unit is sufficient to perform the computation of step1 and step2 without invoking any timing overhead.

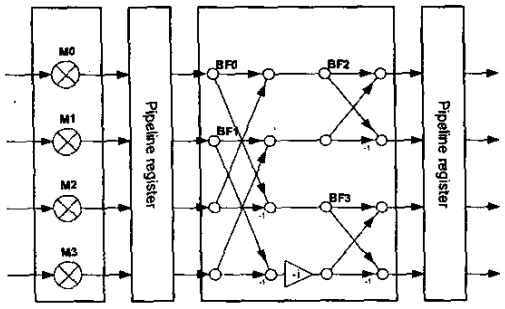

Figure 4 The calculation unit with 4-point FFT computation

3.4 Memory Addressing

To minimize the memory size required during the FFT computation, in-place memory addressing is used. For this, a RAM is partitioned into two banks that can be accessed at the same time.

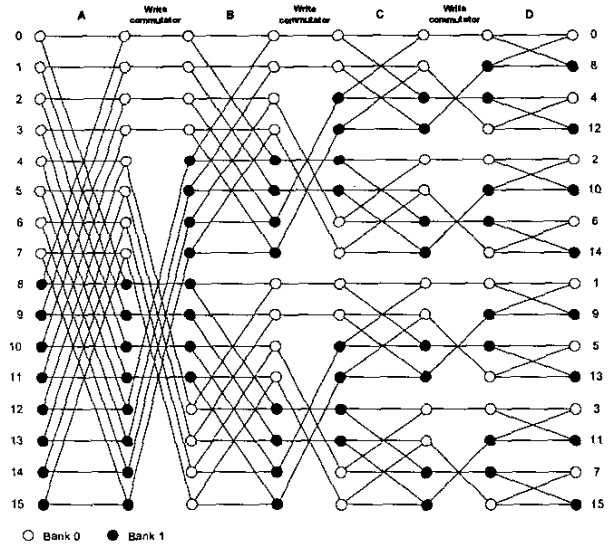

Figure 5 The addressing scheme of a 16-point FFT

Fig. 5 shows an example of the addressing scheme. The white circles mean bank0 accesses and the black circles mean bank1 accesses. To compute a radix-2 butterfly, two complex data are read and written concurrently. For concurrent reads and writes, the two complex data should be located in different banks of a RAM. The proposed addressing scheme is used for the N/4-point FFT computation.

The proposed addressing scheme operates with an address generator and a commutator. Two addresses generated by the address generator are accessed to read and store two complex data with two banks. The commutator swaps the two complex data to feed the data into 2-input butterfly appropriately as shown in Fig. 5. Table II shows an example of the address sequence and Fig. 6 shows the address generator. The unit consists of a counter and a shifter basically.

Table II The address sequence of a 16-point FFT

Seq.	A		B		C		D	
	Bank0	Bank1	Bank0	Bank1	Bank0	Bank1	Bank0	Bank1
0	000	000	000	100	000	110	000	111
1	001	001	001	101	001	111	001	110
2	010	010	010	110	010	100	010	101
3	011	011	011	111	011	101	011	100
4	100	100	100	000	100	010	100	011
5	101	101	101	001	101	011	101	010
6	110	110	110	010	110	000	110	001
7	111	111	111	011	111	001	111	000

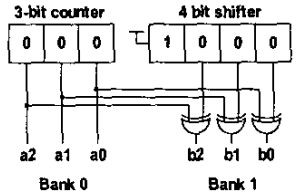

Figure 6 The address generator

This addressing scheme requires a memory size of N complex words for an N-point FFT computation, and can be applied even if the number of points increases, making it easy to implement the addressing controllers of a 2K/4K/8K-point FFT processor.

To compute 2K/4K/8K-point FFT, the processor needs the addressing controller for 2K-point FFT because 8K-point FFT is performed by parallel computing four 2K-point FFTs and a controller that is used for a long-point FFT can also be used for a short-point FFT. The controller of 2K-point FFT needs no architectural change except the bit width of the shifter and counter. For 2K-point FFT, the bit width of the shifter and counter is shortened to 11-bit. If it needs to change the number of FFT points, only the initial value of the shifter is changed as shown in Table III.

Table III Initial value of the shifter for various FFTs

# of FFT points	Initial value (11-bit)
2K (for 8K-point FFT)	10000000000
1K (for 4K-point FFT)	01000000000
512 (for 2K-point FFT)	. 00100000000

4. IMPLEMENTATION AND PERFORMANCE

To verify the proposed algorithm and control operation, a C model was developed and fully tested. After that, a RTL model of the proposed FFT processor is designed.

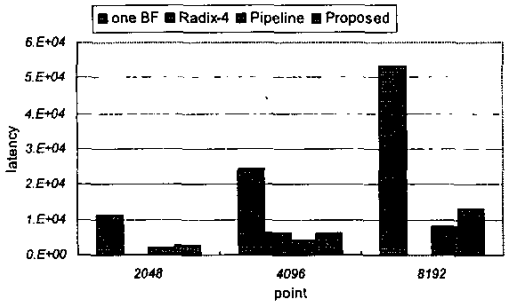

Figure 7 Performance comparison

Fig. 7 shows the performance of the proposed processor compared to the single-memory architecture with radix-2 and radix-4 butterfly and R2MDC architecture. The proposed processor can calculate 2K/4K/8K-point FFTs at the same latency as the radix-4 computation, and the latency is comparable to that of the pipelined architecture.

Table IV Hardware requirement

	# of multipliers	# of adders	memory size
R2MDC	$2(\log_4 N-1)$	$4\log_4 N$	$3N/2-2$
R2SDF	$2(\log_4 N-1)$	$4\log_4 N$	$N-1$
R4SDF	$\log_4 N-1$	$8\log_4 N$	$N-1$
R4MDC	$3(\log_4 N-1)$	$8\log_4 N$	$5N/2-4$
R4SDC	$\log_4 N-1$	$3\log_4 N$	$2N-2$
R2^{2}SDF	$\log_4 N-1$	$4\log_4 N$	$N-1$
Proposed	4	8	N

In Table IV, The hardware resources required in the proposed architecture is compared with various pipelined architectures[4]. It shows the proposed architecture has reached the minimum requirement at 2K/4K/8K-point FFT computation.

The proposed architecture is synthesized in TSMC 0.25μm standard CMOS technology by the design complier of Synopsys. The critical path delay is 10.40ns, meaning that maximum operation frequency is 91MHz. So it takes 28.2μs, 62.0μs and 135.2μs to compute 2K, 4K and 8K-point FFTs, respectively. This performance is enough to satisfy the DVB-T requirement, which is 224μs for 2K-point FFT and 896μs for 8K-point FFT.

The FFT processor consists of 70,600 gates except memory. The calculation unit is synthesized with 64,000 gates and the miscellaneous components including controllers have 6,600 gates.

5. CONCLUSIONS

In this paper, a parallel FFT architecture based on Cooley-Tukey algorithm has been proposed. Implementation details of a 2K/4K/8K-point FFT processor were described and an efficient memory addressing scheme needed to parallel computation was also proposed.

The proposed FFT processor consumes less chip area than the pipelined FFT processor, as it needs a small number of multipliers and adders. The proposed architecture can compute a 2^n-point FFT with the same latency of radix-4 computation and with similar performance to the pipelined processor. The synthesized FFT processor operates at 91Mhz and shows a performance sufficient for high-end applications.

6. ACKNOWLEDGEMENT

This work was supported by the Korea Science and Engineering Foundation through the MICROS center, by the Ministry of Science and Technology and the Ministry of Commerce, Industry, and Energy through the project System IC 2010, and by IC Design Education Center (IDEC).

7. REFERENCES

[1] S. B. Weinstein and P. M. Ebert. *Data transmission by frequency-division multiplexing using the discrete fourier transf.* IEEE Trans. Commun., COM-19(5):628-634 Oct. 1971.

[2] L. J. Cimini. *Analysis and simulation of a digital mobile channel using orthogonal frequency division multiplexing.* IEEE Trans. Commun., COM-33(7):665-675, Jul. 1985

[3] M.Alard and R.Lasalle, *Principles of modulation and channel coding for digital broadcasting for mobile receivers*, ITU WARC-ORB Conference., Sep. 1988.

[4] Shousheng He and Torkelson, M., *Design and implementation of a 1024-point pipeline FFT processor.* Proceedings of the IEEE Custom Integrated Circuits Conference, pp. 131 -134, 1998

[5] Bidet, E.; Castelain, D.; Joanblanq, C.; Senn, P. *A fast single-chip implementation of 8192 complex point FFT*, IEEE Journal of Solid-State Circuits, , Vol. 30 Issue: 3 , pp. 300 -305, March 1995

[6] Baas, B.M. *A low-power, high-performance, 1024-point FFT processor.* IEEE Journal of Solid-State Circuits, , Vol. 34 Issue: 3 , pp/ 380-387 March 1999

[7] J.W.Cooley and J.W.Tukey. *An algorithm for machine computation of complex Fourier Series*, Math. Comput., Vol. 19, 1965