고성능 512-point FFT 프로세서의 설계

김선호*, 김정우*, 오길남**, 김기철*

A Design of High Throughput 512-point FFT Processor

Seonho Kim*, Jungwoo Kim*, Kil-Nam Oh**, Kichul Kim*

요 약

본 논문에서는 데이터 입출력을 고속으로 수행하며 작은 지연시간을 갖는 512-point FFT 프로세서의 구조 및 설계에 대하여 보인다. 설계된 512-point FFT 프로세서는 OFDM 방송에서 요구하는 심볼 레이트로 데이타를 처리할 수 있는 것을 목표로 하였다. 설계된 512-point FFT 프로세서는 셔플메모리를 이용하여 메모리의 요구 사항을 최소화하며, 새로운 strength reduction method 를 적용한 복소곱셈기를 이용하여 기존의 복소곱셈기에 비하여 하드웨어의 비용이 적은 특징을 갖는다.

ABSTRACT

This paper shows the design of a high throughput 512-point FFT processor. The performance target of the 512-point FFT processor is to achieve data symbol rate required for OFDM systems. The memory requirement of the 512-point FFT processor is minimized by adopting shuffle memory system. The hardware cost of the 512-point FFT processor is further reduced by using a complex multiplier with a new strength reduction method.

I. 서론

유럽의 디지털 오디오 방송과 TV 방송에는 이미 OFDM (Orthogonal Frequency Division Multiplexing)이 전송 방식으로 채택되어 있으며, 현재 유럽의 일부 국가에서는 정규 방송 또는 시험 방송이 실시되고 있다. 전송 방식으로서 OFDM 의 효율성은 잘 알려져 있으므로 OFDM 은 디지털 방송뿐만 아니라 통신 분야에서도 많은 관심을 끌고 있다. OFDM 시스템에서 중요한 부분 중의 하나가 OFDM 변복조기로, 이것은 FFT 프로세서를 이용하여 구현된다. 따라서 FFT 프로세서는 OFDM 기반 디지털 방송의 수신 단말을 구현하는 핵심 구성 블록이다. 특히 오디오 방송 수신기는 카 오디오 및 휴대형이 주종을 이룰 것이므로 소형화, 저가, 저전력 소모 등이 추구된다.

일반적으로 크기가 큰 FFT 프로세서의 구조는 보다 작은 FFT 로 분할된 형태를 갖게 되며, 이 경우

* 서울시립대학교 전자전기공학부

Dept. of Electrical Engineering, University of Seoul

** 한국전자통신연구원 DAB 시스템연구팀

Electronics and Telecommunications Research Institute

직교 연산을 연속적으로 수행하여 최종 결과를 얻을 수 있다. 2-D FFT(Fast Fourier Transform)와 같은 분리 가능변환(separable transform)을 하드웨어로 구현하는 가장 일반적인 방법은 두 개의 1-D FFT를 사용하여 각각 행 방향 변환과 열 방향 변환을 수행하게 하는 것이다[2,4,5]. 이 경우에 행 방향(열 방향) 변환의 결과는 임시로 메모리에 저장이 되고 메모리에 저장된 중간 결과에 열 방향(행 방향) 변환이 이루어져 원하는 최종 결과를 얻게 된다. 이러한 구현을 row-column separation method라고 하며 이 때 중간 값을 저장하는 메모리의 구조와 비용이 전체 2-D FFT의 구조와 비용에 큰 영향을 미치게 된다. MxN 2-D FFT의 구현에서 M과 N이 같은 경우에는 전치메모리, transposition stage 등의 효율적인 중간 메모리의 구조가 알려져 있으나 M과 N의 크기가 다른 경우에는 효율적인 방법이 알려져 있지 않다[1,3,6,7]. 본 논문에서는 셔플메모리라는 새로운 형태의 메모리를 M과 N의 크기가 다르며 병렬 구조를 갖는 512-point FFT 프로세서에 적용하여 메모리의 낭비 없이 효율적인 2-D FFT의 구현을 하였다.

트위들팩터의 곱셈을 수행하는 복소곱셈기는 FFT 프로세서의 연산기 중에서 가장 큰 면적을 차지하며, 복소곱셈기의 정밀도는 전체 FFT 연산의 정밀도에 크게 영향을 미치게 되므로 하드웨어의 요구량이 낮고 VLSI에 적합하도록 규칙적인 구조를 갖으며 높은 정밀도를 갖는 복소곱셈기가 반드시 필요하게 된다. 본 논문에서는 새로운 방식의 strength reduction method를 적용하여 복소곱셈기의 하드웨어 요구량을 줄이며, 우수한 정밀도 특성을 나타내는 복소곱셈기를 512-point FFT 프로세서에 적용하여 구현하였다.

본 논문의 나머지 부분은 다음과 같이 구성되어 있다. 본 절에 이어 제 2 절에서는 셔플메모리에 대하여 소개한다. 제 3 절에서는 새로운 구조의 복소곱셈기에 대해 설명한다. 제 4 절에서는 512-point FFT 프로세서의 구조 및 설계에 대해 소개한다. 마지막으로 제 5 절에서는 본 논문의 결론을 보인다.

II. 셔플메모리

본 절에서는 셔플메모리를 소개하고 셔플메모리의 여러 가지 특징을 보인다.

q-셔플 : qc 개의 객체에 대한 q-셔플은 S_{q*c} 로 표시된다. S_{q*c} 는 <0, 1, ..., (qc-1)>에 대한 순열로서 식 (1)과 같이 정의된다.

$$S_{q*c}(i) = \left(i*q + \left\lfloor \frac{i}{c} \right\rfloor \right) \bmod q*c \qquad (1)$$

그림 1에 12개의 객체에 대한 3-셔플(S_{3*4})에가 나타나 있다.

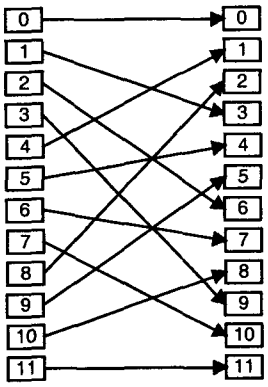

그림 1. 12 객체에 대한 3-셔플

크기가 MxN인 셔플메모리(SM)는 다음과 같은 특징을 가진다.

1. SM은 크기가 MxN인 어레이를 행 우선순서(열 우선순서)로 입력 받아서 열 우선순서(행 우선순서)로 출력한다.

2. "Read-then-write at the same address operation"을 제공한다. 즉 어레이 A의 요소를 열 우선순서(행 우선순서)로 출력하면서 동시에 어레이 B의 요소를 같은 주소에 행 우선순서(열 우선순서)로 입력한다.

셔플메모리의 동작을 크기가 3x4 인 셔플메모리를

예로 설명한다. 크기가 3x4 인 어레이 $A_0, A_1, \ldots, A_n$ 들이 차례로 입출력 될 때 셔플메모리는 어레이 $A_i, 1 \le i \le n,$ 를 행 우선순서로 입력하면서 어레이 A_{i-1} 을 열 우선순서로 출력하게 된다. 이 때 어레이 A_i 의 요소 $a_i(i,j)$는 어레이 A_{i-1}의 요소 $a_{i-1}(m,n)$, $4i+j=m+3n,\ \ 0 \le i,m \le 2,\ \ 0 \le j,n \le 3,$이 저장되어 있던 주소에 새로이 저장되게 된다. 이 때 셔플메모리에서는 식 (2)와 같은 순열을 제공하는 것이 된다.

$$S_{3*4}(4i+j) = \left((4i+j)*3 + \left\lfloor \frac{4i+j}{4} \right\rfloor \right) \bmod 12 \quad (2)$$
$$= i + 3j, \qquad 0 \le i \le 2,\ \ 0 \le j \le 3$$

즉 3x4 셔플메모리에 제공되는 순열은 3-셔플이 된다. 일반적으로 MxN 셔플메모리에서 제공되는 순열은 M-셔플이 된다.

3x4 2-D FFT의 일반적인 구조가 그림 2에 나타나 있다. 1 단계의 4-point FFT는 입력의 각 행을 처리하므로 행 우선순서로 데이터를 출력하며 2 단계의 3-point FFT는 각 열을 처리하므로 데이터를 열 우선순서로 공급 받아야 한다. 이 때 1 단계에서 생성된 중간 값을 저장하는 버퍼가 필요하게 된다. 즉 버퍼는 데이터를 행 우선순서로 입력 받아서 열 우선순서로 출력하여야 한다. 따라서 3x4 셔플메모리를 사용하면 효율적인 버퍼의 구현이 가능하다.

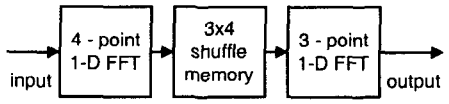

그림 2. 셔플메모리를 사용한 3x4 2-D FFT

그림 2에서 보인 바와 같이 하나의 1-D FFT와 연속하는 1-D FFT의 크기가 서로 다를 경우에 셔플메모리를 사용하는 방법은 크기가 매우 큰 FFT를 여러 개의 작은 FFT로 분할하여 구현하는 경우 작은 FFT 모듈들의 크기 차이 때문에 발생할 수 있는 비효율적인 메모리구조를 효과적으로 개선하여 준다.

III. 복소곱셈기의 구조

본 절에서는 새로운 방식의 strength reduction method를 적용하여 트위들팩터의 곱셈을 위한 복소곱셈기의 하드웨어 요구량을 줄이며, 우수한 정밀도 특성을 나타내는 복소곱셈기를 소개한다.

일반적인 복소곱셈은 4 개의 실수곱셈기를 이용해야 하므로 복소곱셈기의 하드웨어 비용이 증가하는 단점이 있다. 이러한 단점을 극복하기 위해 Golub의 알고리즘과 같이 3 개의 실수곱셈과 5 개의 실수 가감산을 이용하는 여러 가지 방법들이 이미 제안되어 있으나, 결선이 복잡하고 정밀도 측면에서도 우수한 특성을 갖지 못한다[8].

본 논문에서 제안하는 새로운 방법은 사인함수와 코사인 함수의 차를 이용하는 것으로서 식 (3)에 나타난 바와 같이 복소곱셈이 3 개의 실수 곱셈 만을 포함하는 것을 알 수 있다.

Real :
$$A_R \cos\theta + A_I \sin\theta = A_R(\sin\theta + d) + A_I \sin\theta$$
$$= (A_R + A_I)\sin\theta + A_R d \quad (3)$$
Imaginary :
$$A_I \cos\theta - A_R \sin\theta = A_I \cos\theta - A_R(\cos\theta - d)$$
$$= (A_I - A_R)\cos\theta + A_R d$$
$$d = \cos\theta - \sin\theta$$

새로운 방법은 입력샘플의 실수부와 허수부의 합과 차에 각각 사인함수와 코사인함수를 곱하고 이것을 미리 계산된 두 삼각함수의 차 d에 입력샘플의 실수부를 곱한 값과 각각 더함으로써 실수부와 허수부의 출력을 얻을 수 있는 방법이다. 새로운 방법에 대한 블록도가 그림 3에 나타나 있다.

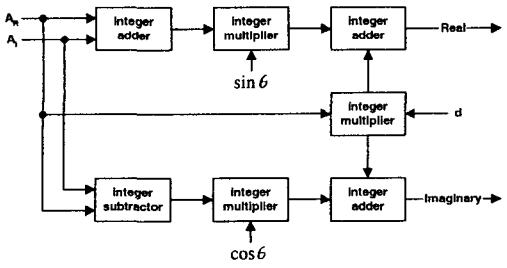

그림 3. 새로운 복소곱셈기의 구조

Golub 의 알고리즘은 실수부의 계산을 변형시킨 방법으로서 식(4)에 나타난 바와 같이 실수부의 계산을 위해 4 개의 실수 가감산기가 이용되므로 구조가 복잡하고 실수부와 허수부의 데이터 출력의 지연시간이 서로 다른 단점이 있다.

Real :
$$(A_R + A_I)\times(\cos\theta - \sin\theta) + A_R \times \sin\theta - A_I \times \cos\theta \quad (4)$$
Imaginary :
$$j(A_R \times \sin\theta + A_I \times \cos\theta)$$

그림 4 에 각 알고리즘의 정밀도 비교결과가 나타나 있다. 정밀도 평가를 위한 시뮬레이션은 C-code 를 이용하여 수행하였으며, 8 비트 데이터의 무작위 정수 샘플 256 개와 0.703125° 단위를 갖는 512 개의 트위들팩터의 조합 131,072 개에 대해 double float precision 을 이용해 얻어낸 기준 값에 대해 실제 하드웨어와 동일한 구조하에서 얻어진 세 가지 방법의 출력 값을 각각 비교하였다. 세가지 방법에 대한 시뮬레이션 결과 본 논문에서 제안한 새로운 방법이 가장 적은 오차율을 보였다.

Method \ item		RMS	mean error	abs mean error	abs max error
Direct algorithm	Real	0.390016	-0.001270	0.302178	1.145289
	Imaginary	0.388336	-0.000601	0.300374	1.148602
Golub's algorithm	Real	0.471116	-0.000969	0.362082	1.531930
	Imaginary	0.388336	-0.000601	0.300374	1.148602
proposed algorithm	Real	0.381820	-0.000660	0.295855	1.114070
	Imaginary	0.384273	-0.001049	0.297148	1.197582

그림 4. 복소곱셈기의 정밀도 비교

IV. 512-point FFT 의 구조 및 설계

본 절에서는 512-point FFT 프로세서의 구성요소와 전체구조를 소개하고 각 기능 블록 들의 동작을 설명한다.

그림 5 에 셔플메모리를 이용하며 64-point FFT 와 8-point FFT 로 분할된 512-point FFT 의 구조가 나타나 있다. 내부연산의 정밀도는 실수부 및 허수부 각각 16 비트를 사용하였으며, 입력 데이터는 실수부 및 허수부 각각 8 비트로서 SM(Shuffle Memory)1 에 저장될 경우 부호확장 방식에 의해 실수부 및 허수부 각각 16 비트로 변환된다.

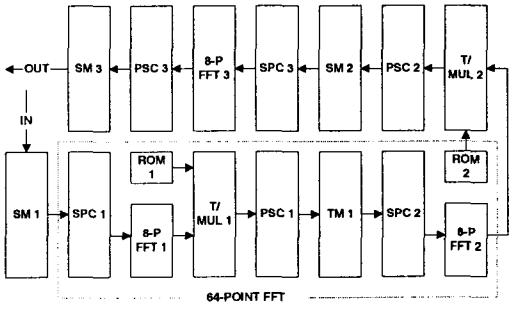

그림 5. 512-point FFT 프로세서의 구조

내부 정밀도가 16 비트이고 8-point FFT 와 T/MUL(Twiddle Factor Multiplier)는 8 개의 입출력 단자를 갖고 있는 병렬구조의 연산 장치이므로 데이터의 유입량과 유출량을 동일하게 조절하기 위해서 각 입출력 단자는 매 클럭 2 비트의 데이터 입출력을 수행하게 된다. SM(Shuffle Memory)3 으로부터의 최종출력은 실수부 및 허수부 각각 16 비트이다. 그림 5 에서 SPC(Serial to Parallel Converter)는 순차적으로 입력되는 데이터를 병렬형태로 공급하는 역할을 수행하며, PSC(Parallel to Serial Converter)는 병렬형태로 출력되는 데이터를 순차형태로 변환하여 공급하는 역할을 수행한다. 그림 5 의 512-point FFT 가 FFT 연산을 수행하는 과정은 다음과 같다.

1. 각각 8 비트의 정밀도를 갖는 실수부 및 허수부 데이터가 매 클럭 순차적으로 SM1 에 저장되며,

동시에 동일한 주소에서 읽혀지는 순차 데이터는 SPC1 으로 입력되어 매 클럭 마다 실수부 및 허수부 각각 2 비트의 크기를 갖는 8 개의 병렬 형태의 데이터로 변환되어 하나의 16 비트 데이터 당 8 클럭의 주기로 8-point FFT1 에 공급된다.

2. ROM1 은 64-point FFT 내부에서 2 단계로 분할되는 2 개의 8-point FFT1,2 사이에 발생하는 트위들팩터 곱셈을 위해 64 개의 트위들팩터를 저장하며, 8 개의 출력 단자를 통해 매 8 클럭의 주기로 T/MUL1 에 트위들팩터를 공급한다.

3. 8-point FFT1 은 SPC1 으로부터 매 클럭 출력되는 실수부 및 허수부 각각 2 비트의 크기를 갖는 8 개의 병렬형태의 데이터를 이용하여 8-point FFT 연산을 수행한다.

4. T/MUL(Twiddle Factor Multiplier)1 은 8-point FFT1 로부터 매 클럭 출력되는 실수부 및 허수부 각각 2 비트의 크기를 갖는 8 개의 병렬형태의 데이터와 ROM1 로부터 출력되는 8 개의 트위들팩터를 이용하여 2 단계로 분할되는 2 개의 8-point FFT1,2 사이에 발생하는 트위들팩터 곱셈을 수행한다.

5. TM(Transposition Memory)1 은 64-point FFT 내부에서 2 단계로 분할되는 2 개의 8-point FFT1,2 사이에서 제 1 단계의 8-point FFT1 의 행 단위 변환의 결과를 저장하고 재 2 단계의 8-point FFT2 의 열 단위 변환을 위해 데이터를 공급하는 전치메모리이며, PSC1 에서 순차 형태로 변환된 데이터를 입력 받아 저장하고 8-point FFT2 로의 병렬 형태의 데이터 공급을 위해 SPC2 로 순차 데이터를 출력한다.

6. ROM2 는 512-point FFT 내부에서 2 단계로 분할되는 하나의 64-point FFT 와 하나의 8-point FFT 사이에 발생하는 트위들팩터 곱셈을 위해 512 개의 트위들팩터를 저장하며, 8 개의 출력 단자를 통해 매 8 클럭의 주기로 T/MUL2 에 병렬형태의 트위들팩터를 공급한다.

7. T/MUL2 는 64-point FFT 로부터 매 클럭 출력되는 실수부 및 허수부 각각 2 비트의 크기를 갖는 8 개의 병렬형태의 데이터와 ROM2 으로부터 출력되는 8 개의 트위들팩터를 이용하여 2 단계로 분할되는 2 개의 64-point FFT 와 8-point FFT 사이에 발생하는 트위들팩터 곱셈을 수행한다.

8. 실수부 및 허수부 각각 16 비트의 내부 정밀도를 갖으며 매 클럭 2 비트씩 출력되는 8 개의 T/MUL2 의 연산 결과는 PSC2 에서 순차형태로 변환되어 매 클럭 순차적으로 SM2 에 저장되며, 동시에 동일한 주소에서 읽혀지는 순차 형태의 데이터는 SPC3 에서 실수부 및 허수부 각각 2 비트의 크기를 갖으며 배 클럭 출력되는 8 개의 병렬 데이터 형태로 변환되어 8-point FFT3 에 공급된다.

9. 8-point FFT3 의 연산결과는 PSC3 로 공급되며, PSC3 의 순차 형태의 출력 데이터는 SM3 에 저장되고, 동시에 최종 출력 단으로 실수부 및 허수부 각각 16 비트를 갖는 순차 데이터 형태로 512-point FFT 의 결과가 출력된다.

설계과정은 다음과 같다. 512-point FFT 프로세서의 설계에 사용된 CAD tool 은 SYNOPSYS, CADENCE, HSPICE 이다. 전체 블록을 VHDL 로 설계하여 SYNOPSYS 에서 기능 시뮬레이션을 수행한 후 0.6u SOG 라이브러리를 이용하여 합성하였다. 합성결과는 각 기능 블럭의 하드웨어 비용과 복잡도에 대한 예측에 이용되었다. CADENCE 를 이용하여 기본 셀들을 layout 하고 HSPICE 를 이용하여 회로 시뮬레이션을 수행하였다. 기본 셀들을 이용하여 상위 블록들을 구현하고 이러한 상위 블록간의 결선을 단계적으로 수행하며 DRC 및 LVS 로 검증하는 방법을 이용하였다.

그림 6에 512-point FFT 프로세서의 layout 도가 나타나 있다. 사용된 공정은 0.25micron CMOS, single poly, 5-metal 이다. 설계된 512-point FFT 프로세서의 주요특징이 그림 7에 나타나 있다.

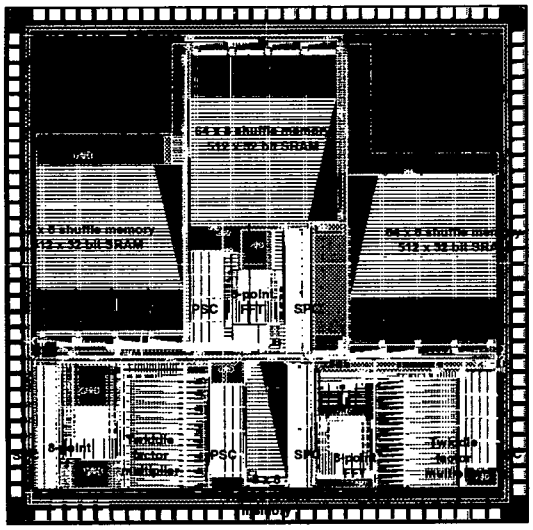

그림 6. 512-point FFT 프로세서의 layout

Internal precision	16-bit
Transistors	1,123,235
Die area	3.3mm x 3.3mm
Operation speed	~ 30MHz
Signal I/O pins	52
Power pins	32

그림 7. 512-point FFT 프로세서의 주요특징

V. 결론

본 논문에서는 OFDM을 이용하는 지상파 디지털 오디오 방송의 수신기에 사용되는 OFDM용 512-point FFT 프로세서의 구조 및 설계에 대하여 보였다. 설계된 512-point FFT 프로세서는 셔플메모리를 이용하여 크기가 서로 다른 2-D FFT의 경우와 같이 전치 메모리를 이용할 수 없는 구조에서 메모리의 요구사항을 최소화하며, 새로운 strength reduction method를 적용한 복소곱셈기를 이용하여 하드웨어의 요구량이 낮고 VLSI에 적합하도록 규칙적인 구조를 갖는 특징이 있다. 설계된 복소곱셈기는 정밀도 측면에서도 우수한 특성을 보이고 있다.

참고문헌

[1] A. Artieri, S. Kritter, F. Jutand, and N. Demassieux, "A One Chip VLSI for Real Time Two-Dimensional Discrete Cosine Transform," Proc. Intl. Symposium on Circuits and Systems, pp. 701-704, 1988.

[2] R. E. Blahut, Fast Algorithms for digital Signal Processing, Addison Wesly, 1987.

[3] Carlach, P. Penard, and JL. Sicre, "TCAD: a 27 MHZ 8x8 Discrete Cosine Transform Chip," Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing, pp. 2429-2432, 1989.

[4] R. C. Gonzalez and P. Wintz, Digital Image Processing, 2nd Ed., Addison-Wesley Publishing Company, 1987.

[5] A. V. Oppenheim, R. W. Schafer, Discrete-Time Signal Processing, Prentice-Hall, 1989.

[6] M-T. Sun, T-C. Chen, and A. M. Gottlieb, "VLSI Implementation of a 16x16 Discrete Cosine Transform," IEEE Trans. Circuits and Systems, vol. 36, no. 4, pp. 610-617, 1989.

[7] S. Uramoto, Y. Inoue, A. Takabatake, Y. Yamashita, H. Terane, and M. Yoshimoto, "A 100-MHz 2-D Discrete Cosine Transform Core Processor," IEEE J. Solid-State Circuits, vol. 27, no. 4, pp. 492-499, 1992.

[8] 디지털 연산 알고리즘 및 회로 설계, 광운대학교 반도체설계교육지역센터, 1998.