Radix- 4,2 SIC FFT 프로세서 설계

정기웅*, 한창용**, 김규철*
*단국대학교 전자컴퓨터공학과
**단국대학교 전자컴퓨터공학과
e-mail : ingo9@chollian.net

Design of Radix - 4,2 SIC FFT processor

Gi-Woung Jung*, Chang-yong Han**, Kyu-Cheol Kim*
*Dept. of Electronics-Computer Engineering, Dan-Kook University
** Dept. of Electronics-Computer Engineering, Dan-Kook University

요 약

OFDM(Orthogonal Frequency Division Multiplexing)은 제 4 세대 기술로 일컬어지는 변조 방식으로 최근 유럽의 디지털 오디오 방송(DAB)과 디지털 비디오 방송(DVB)에 표준으로 사용되고 있으며, IEEE 802.11a 무선 LAN 및 디지털 가입자 라인 xDSL 에서도 사용되고 있다. 본 논문에서는 OFDM 모뎀 구현의 핵심이라고 할 수 있는 64-포인트 FFT(Fast Fourier Transform) 프로세서의 여러 가지 구조를 분석하고, 이들과 비교하여 성능 대 면적 비를 획기적으로 향상시킨 새로운 FFT 프로세서인 Radix-4,2 SIC (Single Instruction Computer) 구조를 제안하였다. 본 논문에서 제안하는 SIC 구조는 버터플라이 연산의 재사용을 극대화하였으며 Radix-4,2 알고리즘을 사용함으로써 FFT 프로세서에서 면적의 80%를 차지하는 복소곱셈기의 수를 감소시켜 크기를 획기적으로 줄인 결과를 보여 준다.

1. 서론

OFDM(Orthogonal Frequency Division Multiplexing) 전송방식의 기본 개념은 직렬로 입력되는 데이터 열을 N 개의 병렬 데이터 열로 변환하여 각각 분리된 부반송파에 실어 전송함으로써 데이터 율을 높이는 것이다. 이때 부반송파는 직교성을 유지 할 수 있도록 설계 되어야 한다. 이러한 부반송파는 송-수신단에서 IFFT(Inverse Fast Fourier Transform)와 FFT processor 를 이용하여 구현가능하며 부 반송파의 수 즉, N 의 증가에 따라 많은 연산량을 효율적으로 처리 할 수 있으며 면적 또는 전력소모 면에서 보다 효율적은 N-pont FFT processor 계발에 대한 연구가 요구 되어지고 있다[1].

Algorithm 측면에서 60 년대 Cooley 와 Tukey 의 FFT algorithm[2]이 개발된 이래로, 고속 FFT 를 구현하기 위한 노력이 계속되어 Goertzel algorithm[3], Split-radix algorithm[4], Winograd algorithm[5], QFT(Quick Fourier Transform) algorithm[6], Radar-Brenner algorithm[7], radix-2/4/8 algorithm[8] ,radix- 2^2 ,radix - 2^3 algorithm[9] 등이 개발되었다.

구조적인 측면에서는 단일 버터플라이 연산자(single butterfly operator)구조[10], MDC(multi-path Delay commutator)[11], SDF(single-path Delay Feedback)[12], SDC(single-path Delay commutator)[13]등과 같은 파이프라인 구조, 완전 셔플(perfect shuffle), 시스토릭 어레이(systolic array)[14]와 같은 병렬 구조가 개발되었다. 이중 파이프라인 구조가 면적과 수율 면에서 유리하여 많이 사용되고 있다. 본 논문에서는 radix-2^3 algorithm 의 비단순 승산기의 개수를 줄이는 특성을 그대로 이용하면서 radix-4 버터플라이 연산자에 기반한 radix-4,2 algorithm 을 사용하여 radix- 2^3 알고리즘에 비해 수율과 면적면에서 이득을 보고자 하였으며 구조적으로 SIC(Single Instruction Computer) 구조를 사용함으로써 버터플라이 연산자의 재사용을 극대화하여 크기를 줄인 FFT processor 를 설계하고자 한다.

2. FFT 알고리즘

2.1 DFT 알고리즘

N-point 의 분석 방정식이라 불리는 DFT 와 합성방정식이라 불리는 IFFT 는 각각 식 (2-1)과 (2-2)로 나타 낼 수 있다.

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

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

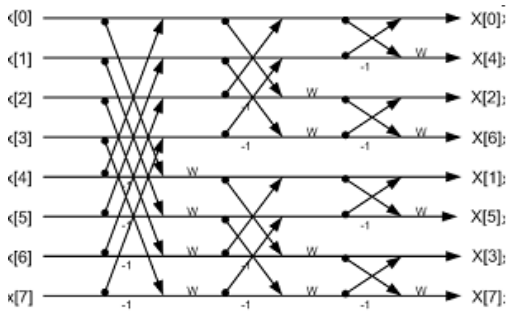
여기서 $W_N=e^{-\frac{j2\pi}{N}i}$ 이며, 트위들 팩터(twiddle factor)라 부른다. 트위들 팩터를 입력 데이터 x(n)에 승산하는 과정은 인덱스 i 에 따라 단순승산(trivial multiplication)과 비단순 승산(nontrivial multiplication)으로 나눌 수 있다. DFT 의 계산에는 많은 곱셈기와 덧셈기가 사용된다.

2.2 Radix-2 FFT 알고리즘

Cooley-Tukey 가 발표한 DIT(Decimation-In-Time) 알고리듬과 sande 가 발표한 DIF(Decimation-In-Frequency) 알고리듬은 길이 N 의 시퀀스를 이보다 작은 길이의 시퀀스로 분해함으로써 DFT 의 연산량을 줄이는 방식이다.

$$\begin{aligned} x(k) &= \sum_{n=0}^{N-1}x(n)W_N^{kn} \\ &= \sum_{n=0}^{N/2-1}x_{2n}W_N^{2kn}+\sum_{n=0}^{N/2-1}x_{2n+1}W_N^{(2n+1)k} \qquad (2) \\ &= \sum_{n=0}^{N/2-1}x_{2n}W_{N/2}^{kn}+W_N^k\sum_{n=0}^{N/2-1}x_{2n+1}W_{N/2})^{kn} \\ &= X_{11}(k)+W_N^kX_{12}(k) \end{aligned}$$

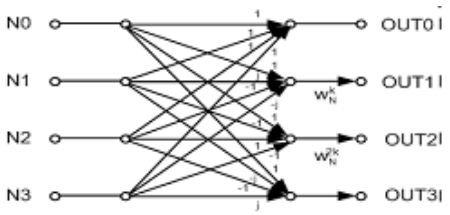
식(2)는 DIF 알고리즘의 간단한 수식이다. 이의 연산흐름도를 그림으로 표현하면 그림(1)과 같이 나타낼수 있다.



그림(1) 8 포인트 FFT 의 연산흐름도

2.3 Radix-4 FFT 알고리즘

radix-2 알고리듬은 길이 N 이 2 의 승수 형태 즉 N=2^x 일 때만 가능하다. N= 4^x 일때 역시 radix-R 알고리듬이 성립된다. radix-4 알고리듬은 $N\log_4 N$ 의 연산량을 가지게 되어 데이터 처리율 면에서 radix-2 알고리듬에 비해 효율적이 지만 butterfly 연산자의 구조가 복잡해지는 단점을 가지게 된다.



그림(2) radix-4 버터플라이 구조

2.3 제안된 알고리즘

본 논문에서 이용하고자 하는 Radix-4,2 알고리즘은 N 의 시퀀스를 3 차원 인덱스 맵에 의해 분해과정을 수행하는데 그 수행순서를 Radix-4, 2 순으로 진행하게 된다.

$$\begin{aligned} X(k) &= X(k_1+4k_2+8k_3) \\ &= \sum_{n_3=0}^{\frac{N}{8}-1}\sum_{n_2=0}^{3}[BF_4(\frac{N}{8}n_2+n_3,k_1)]W_8^{n_2(k_1+2k_2)}W_N^{n_1(k_1+2k_2)}W_{\frac{N}{8}}^{n_3k_1} \qquad (3) \\ &= \sum_{n_3=0}^{\frac{N}{8}-1}[H(k_1,k_2,n_3)W_N^{n_3(k_1+2k_3)}]W_{\frac{N}{8}}^{n_3k_3} \end{aligned}$$

$$\begin{aligned} H(k_1,k_2,n_3) &= \sum_{n_2=0}^{1}[BF_4(\frac{N}{8}n_2+n_3,k_1)]W_8^{n_2(k_1+4k_2)} \qquad (4) \\ &= BF_4(n_3,k_1)+BF_4(n_3+\frac{N}{8},k_1)W_8^{(k_1+4k_2)} \end{aligned}$$

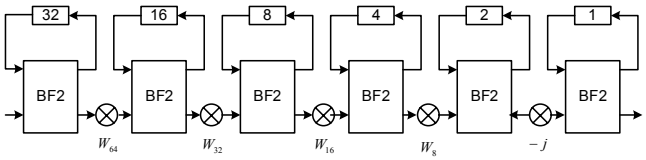
식(3),(4)는 Radix-4,2 알고리즘의 분해식과 버터플라이 연산식으로 이는 하나의 Radix-4 버터플라이 연산자와 하나의 Radix-2 버터플라이 연산자로 구현되어지며 Radix-2^3 알고리즘과 같이 최소의 비단순 승산기를 포함하면서 Radix-4 버터플라이 연산자를 사용하여 데이터 처리율면에서 두배의 이득을 얻게 됨을 보여준다.

3. 프로세서 구조

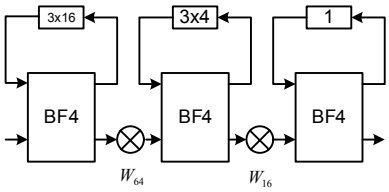
프로세서에 사용되는 hardware 구조는 네가지로 나눌 수가 있다. FFT 의 신호흐름을 그대로 구현한 Array FFT 구조, 신호 흐름의 행으로 열을 공유하는 pipe-line FFT 구조, pipe-line FFT 구조와 반대 개념인 column FFT 구조, 본 논문에서 적용하고자 하는 SIC FFT 구조이다.

3.1 pipe-line 구조.

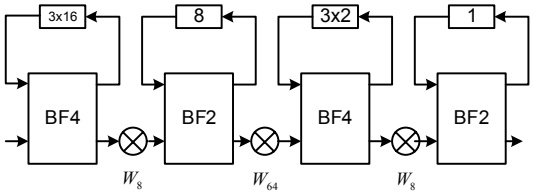
pipe-line FFT 구조(linear systolic array FFT)는 FFT 신호 흐름도에서 각각의 열을 $\log_r N$ 개의 처리요소를 가지는 하나의 행으로 나누어 표현하며 R2MDC, R2SDF, R4MDC, R4SDF, R4SDC 그리고 R2^2SDC 구조가 있다.



그림(2) R2SDF 64 포인트 FFT 프로세서



그림(3) R4SDF 64 포인트 FFT 프로세서



그림(4) R4,2SDF 64 포인트 FFT 프로세서

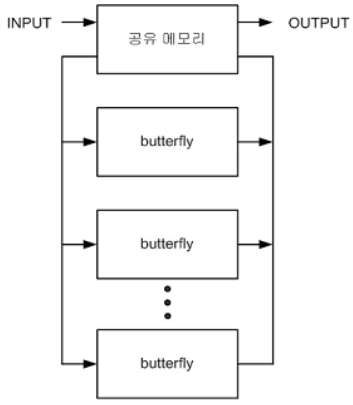
그림(2)(3)(4)는 SDF 방식의 FFT 프로세서이다. 살펴보면 제안된 radix-4,2 알고리즘의 비단순 복소승산기의 수가 하나로 제일 적음을 볼 수 있다. 표 1 은 SDF 방식의 알고리즘별 비단순복소승산기의 수를 비교한 것이다.

표 1. SDF 방식의 알고리즘별 비단순복소승산기수

알고리즘	R2SDF	R4SDF	R4,2SDF
복소승산기	3	2	1

3.2 SIC 구조 FFT 프로세서

파이프라인 구조가 면적과 처리율면에서 유리하여 많이 사용되고 있지만 파이프라인 구조는 포인트에 따라 많은 버터플라이 연산자와 지연소자가 필요하게 되어 포인트가 큰 프로세서에서는 그 면적이 커지게 되는 단점이 생기게 된다. 때문에 본 논문에서는 SIC 구조를 이용하여 프로세서를 설계 면적을 줄이고자 하였다.

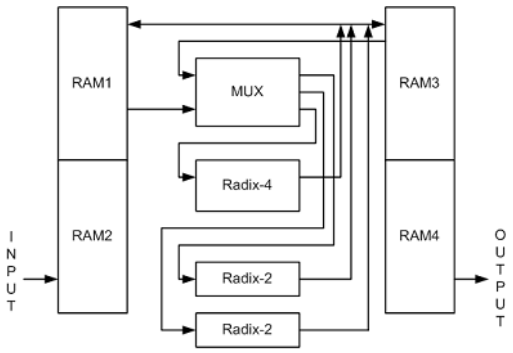


그림(5) SIC FFT 구조

SIC FFT 구조는 새로운 데이터가 FFT 프로세서에 입력되었을 때 그 데이터가 공유 메모리에 입력되고 butterfly 연산자는 공유 메모리에 저장되어 있는 입력데이터를 얻게 된다. 데이터가 공유 메모리로부터 butterfly 로 로딩되어 지면 연산을 수행하는 메모리의 위치가 butterfly 연산자로부터 계산된 결과를 저장하기 위해 다시 사용되어 진다.

3. 검증

제안된 FFT processor 의 검증은 기존의 SDF pipeline 방식의 radix-4,2 프로세서와 설계된 SIC 구조의 radix-4,2 FFT 프로세서를 논리 합성하여 비교하는 방식으로 진행하였다. 각 프로세서의 논리 합성은 synplify 로 진행하였다.



그림(6) SIC 구조 R4,2 FFT 프로세서 구조

그림(6)은 기본적인 SIC 구조 Radix-4,2 FFT 프로세서의 구조이다. 크게 한 개의 Radix-4 버터플라이 연산자와 2 개의 Radix-2 버터플라이연산자 그리고 공유메모리로 구성되었다. 2 개의 radix-2 연산자는 radix-4 버터플라이 연산자의 데이터 처리율에 맞추기 위하여 사용되었다.
논리합성 결과 표 2 와 같은 결과를 보였다.

표 2. Radix4,2 FFT 구조에 따른 게이트수 비교

알고리즘	복소곱셈기수	게이트수	연산속도
R4,2(MDC)	비단순: 1 개 단순 : 2 개	76,751	3.6us
R4,2(SIC)	비단순: 2 개 단순 : 1 개	57,932	3.9us

SIC 구조 방식의 FFT 프로세서가 비단순 복소곱셈기의 수가 하나 많으나 pipe-line 방식의 지연소자로 인해 면적면에서 이득이 있으며 연산속도 면에서는 컨트롤부분의 복잡성으로 인해 약간의 차이가 있다. 이는 컨트롤 부분에 대한 업그레이드를 통해 극복할수 있을 것이다.

4. 결론

본 논문에서는 기존에 많이 사용되던 파이프라인 구조의 FFT 프로세서를 대신해 공유메모리를 사용하는 SIC 구조의 FFT 프로세서를 설계함으로써 버터플라이 연산자의 수와 지연수자를 줄여 면적과 전력소모를 줄였다. 또한 Radix-4,2 알고리즘을 이용하여 Radix-2 기반의 알고리즘보다 데이터 처리율을 증가시키며 비단순복소곱셈기의 수를 줄임으로써 면적과 처리속도면에서 우위를 보였다. 하지만 SIC 구조의 특성상 포인트의 수가 증가함에 따라 그 컨트롤이 어려워지며 그로인해 처리속도가 지연되는 문제점이 나타났으며 앞으로 이 부분의 해결을 위해 더욱 연구해야 할 것이다.

참고문헌

[1] S. Bertaxxoni, G. C. Cardarilli, M. Iannuccelli, M. Salmeri, A Salsano, and O. Simonelli, "16-point high speed (I) FFT For OFDM Modulation", ISCAS98, Vol. 5, pp.210-212, 1998

[2] J. W Cooley and J. W Tukey, " An Algorithm for the machine calculation of Complex Fourier series", Math. Comp., Vol.19, pp.297-301, April 1965

[3] Blahut, Richard E, "Fast algorithm for Digital Signal Processing", Addison-Wesley Pub., pp.131-133, 1985

[4] P. Duhamel and H. Hollomann, "Split radix FFT algorithm", Electronic Letter, Vol. 20, pp.14-16, Jan 1984

[5] S. Winograd, "On Computing the Discrete Fourier Transform", Proc. Nat. Acad. Sci., U.S. Vol.73, pp.1005-1006, Apr. 1976

[6] H. Guo, G.A. Sitton, and C.S. Burrus, " The Quick Discrete Fourier Transform", Proc. of ICASSP, Vol.III, pp.445-447, Adelaide, Austrailia, April 1994

[7] C. M. Rader and N. M. Brenner, "A New Principle for Fast Fourier Transform", IEEE Trans. Acoust, Speech, Signal Processing, Vol.ASSP-24, pp.264-266, Apr.1969

[8] Robert D. Strum and Donald E. Kirk. "First Principles of Discrete Systems and Digital Signal Processing" - Addison Wesley publishing company, 1988, pp.493-528

[9] Glenn Zelniker and Fred J. Taylor, "Advanced Digital Signal Processing - Dekker, pp107-163

[10] Bevan M. Baas, "A 9.5 mW 330 ㎲ 1024-point FFT Processor", IEEE Custom Intergrated Circuits Conference, pp.127-130, 1998

[11] V. Szwarc, L. Desormeaux, W. Wong, C. Yeung, C.H. Chan and T. A. Kwasniewski, "A Chip Set for Pipeline and Parallel Pipeline FFT Architecture," Journal of VLSI Signal Processing, Vol. 8, pp. 253-265, 1994

[12] S. F. Gorman and J. M. Wills, "Partial Column FFT Pipelines," IEEE Transaction on Circuit and Systems II, Vol. 42, pp. 414-423, 1995

[13] I. Gertner and M. Shamash, " VLSI architectures for Multidimensional Fourier transform processing," IEEE Trans. Comput., Vol. C-36, pp.1265-1274, 1987.

[14] E. E. Sqartzlander, Jr., "Systoric FFT Processor,"The First International Workshop on Systoric Array, pp.133-140, 1987