2001年度 第14回 信號處理合同學術大會 論文集 第14卷 1號

FFT 시스템을 위한 효율적인 인덱스 어드레싱기법 구현

홍선영, 신태철, *이광재, 이문호
전북대 전자정보공학부, *한려대
전주시 덕진구 덕진동 1가 664-14

An Efficient index Addressing Method Implementation for FFT system

Sunyoung Hong, Taechul Shin, *Kwangjae Lee, Moon Ho Lee,
*Department of Elec. & Info. Engineering Chonbuk National University
E-mail : moonho@moak.chonbuk.ac.kr

요 약

본 논문은 radix-2 FFT를 파이프라인 기법으로 구현할때의 성능 향상을 위한 메모리 어드레싱기법에 대한 새로운 구조를 제안하고자 한다. Fast Fourier Transform(FFT) 프로세서의 속도 및 성능은 파이프라인 싸이클과 클럭에 좌우되므로, 동시에 병렬로 처리하기 위한 입력 데이타에 access하기 위해 사용되어지는 기존의 메모리 어드레싱 기법은 지연문제로 인해 FFT 프로세서 성능 저하의 원인이 된다. 이 기법은 정확한 메모리 뱅크를 선택하기 위한 주소부 패러티 체크가 필요 없으므로 수행 속도를 빠르게 하고, ROM에 저장된 Coefficient의 실수부와 허수부의 상호교환특성을 이용하여 Coefficient ROM을 반으로 줄일 수 있다. 이 논문에서 제안된 구조는 VHDL을 사용하여 설계하였고, 설계된 회로를 시뮬레이션 및 합성시켰다.

1. 서론

OFDM (othogonal frequency division multiplexing) 시스템은 전송하려는 데이터를 여러개의 비트열로 쪼개고 각각을 부채널로 나누어 변조한 후 병렬로 전송하는 다중반송파 기술의 일종으로 기본 개념은 30년 전에 이미 제안 되었으나 구조의 복잡도로 인하여 널리 사용되지 못하였으나 최근 FFT (Fast Fourier Transform)를 포함한 각종 디지털 신호처리 기술이 발전함으로써 그 관심이 크게 고조되고 있는 실정이다. 이미 유럽의 디지털 방송의 표준으로 사용하고 있는 OFDM은 종래의 FDM (Frequency Division Multiplexing)과 비슷하나 무엇보다 부반송파간에 직교성을 유지함으로서 부반송파간 주파수 간격이 최소화됨으로써 단위 주파수당 전송효율이 FDM 과 달리 매우 높은 것이 특징이다.

병렬데이터 시스템에서 요구되는 신호 발생 장비와 동기 복조 세트들의 비용이 많이 드는데, 이러한 문제점들은 Fast Fourier Transform(FFT) 알고리즘의 사용으로 없앨 수 있다.

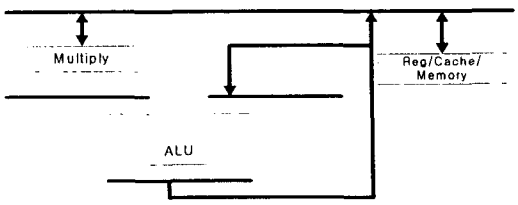

그림 1 FFT 연산의 DSP 구조

본 논문은 radix-2 FFT를 파이프라인 기법으로 구현할때의 성능 향상을 위한 메모리 어드레싱기법에 대한 새로운 구조를 제안하고자 한다.
FFT 프로세서의 속도 및 성능은 파이프라인 싸이클과 클럭에 좌우된다.

$$N\times K\frac{bits}{symbol}\times S\frac{symbols}{\sec ond}=NKS\frac{bits}{\sec ond} \quad (1)$$

N : FFT 길이 또는 캐리어수
K : 성상도 크기와 관련된 변수
S : 심볼율

동시에 병렬로 처리하기 위한 입력 데이타를 access하기 위해 사용되어지는 기존의 메모리 어드레싱 기법은 분리된 메모리 뱅크가 선택되어지기 전에 주소부 패러티가 결정되어져야 하고 Butterfly(BF)의 두 입출력 데이터 및 주소가 상호 교환되어져야 하므로 지연이 심하게는 CLA(Carry Look Ahead)에 가까운 경우도 발생하여 FFT 프로세서 성능 저하의 원인이 된다.
본 논문에서 제시하는 메모리 접근 방식은 메모리를 모듈화하여 H/W 복잡도를 감소시키고 필요한 메모리만 활성화시켜 전력 효율면에서도 성능 개선을 가져올 수 있다. FFT구조에서 2개의 BF 입력은 2개의 메모리 뱅크로 나누어지고 다음 단계로의 올바른 접근을 위해 BF 출력은 2개의 메모리 뱅크에 저장된다. 이때의 충돌을 피하기 위해 BF 카운터비트를 고려하여 메모리 뱅크를 선택한다. 이는 정확한 메모리 뱅크를 선택하기 위한 주소부 패러티 체크가 필요없으므로 수행 속도를 빠르게 할 수 있다. 또한, ROM에 저장된 Coefficient의 실수부와 허수부의 상호교환특성을 이용하여 Coefficient ROM을 반으로 줄일 수 있다. 본 논문에서 제안한 구조는 VHDL을 사용하여 설계하였고, 설계된 회로를 시뮬레이션 및 합성하였다. 본 논문은 다음과 같이 구성된다. 2장에서는 Radix-2 FFT 알고리즘을 소개하고, 3장에서는 병렬 데이터 접근을 위한 주소발생을 소개한다. 4장에서는 FFT 프로세서를 위한 인덱스의 메모리 어드레싱 과정과 시뮬레이션 및 합성 결과를 소개한다. 마지막으로 5장을 결론으로 본 논문을 마친다.

2. Radix-2 FFT 알고리즘

Discrete Fourier Transform(DFT)에서 총 실수 곱셈은 $4\times N^2$, 실수 덧셈은 2N(2N-1)이 된다. 이는 N값이 커짐에 따라 지수적으로 증가함으로 거의 구현이 불가능하다. 이에 계산량을 감소하기 위한 방안이 연구되었다. 1965년 James Cooley 와 John Turky는 DFT의 연산의 W_N^{kn}의 대칭성과 주기성을 이용하여 복잡도를 $O(N^2)$에서 $O(N\log N)$ 으로 줄였다. 스테이지 s에서 BF 연산 후 출력 시퀀스는 입력이 있던 메모리에 저장함으로써 부가적인 메모리를 요구하지 않게 되는 in-place 계산 특성을 가지므로 본 논문에서는 in-place radix-2 DIT(Discrete in-Time) FFT를 고려한다. Radix-2 in-place FFT 알고리즘은 각각의 BF에서 각각의 BF에서 동시에 입력을 읽어들여 출력을 내므로 메모리 어드레싱은 각 클럭 사이클마다 정확한 타이밍으로 4개의 메모리 연산이 동시에 수행되어야 한다. N 포인트 FFT의 일반적인 식은 다음과 같다.

$$X_1(k)=\sum_{n=0}^{N-1}x(n)W_N^{kn} \quad , \quad k=0,\cdots,N-1 \quad (2)$$
$$W_N=e^{-j2\pi/N}$$

$N=2^n$이라 하면, n은 스테이지 수가 된다. 각각의 스테이지에서의 BF계산은 그림2과 같고 이를 식(3),(4)과 같이 나타낼 수 있다.

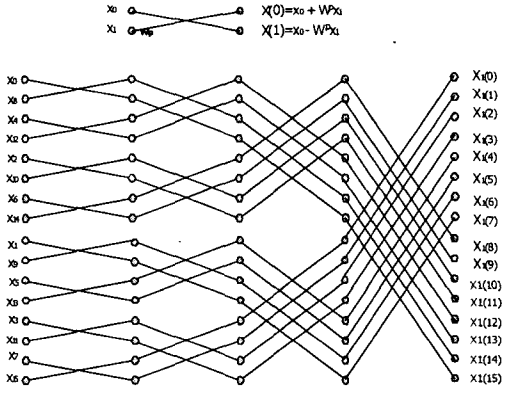

그림 2. 16포인트 FFT의 신호 흐름도

$$X_u(S) = X_u(S-1) + X_d(S-1)\ W_N^c \quad (3)$$

$$X_d(S) = X_u(S-1) - X_d(S-1)\ W_N^c \quad (4)$$

$$u = l + 2^{s+1}g \quad (5)$$

$$d = u + 2^s \quad (6)$$

$$l = 0, 1, \cdots, 2^p-1 \quad (7)$$

$$g = 0,1, \cdots, 2^{n-1-s}-1 \quad (8)$$

$$c = 2^{n-1-s} \cdot l \quad (9)$$

$$s = 0, 1, \cdots, n-1 \quad (10)$$

즉, u와 d는 BF연산의 두 입력 주소이고 s는 각 스테이지의 인덱스이다. 메모리에 접근하기 전에 주소부 패러티가 계산되어지는 일반적인 방식은 패러티 계산 지연이 크다. 본 논문의 목적은 각 스테이지의 인덱스 특성을 이용하여 매우 간단하며 지연이 데이터 주소 발생의 반정도로 주소 발생 지연을 감소시키는 것이다. 3장에서 이러한 주소 발생 기법을 소개하겠다.

3. 병렬 데이터 접근을 위한 주소 발생

식 (4)에서 (9)까지의 식을 구체적으로 보이기 위해 다음 표1과 같이 N=16일 경우를 예로 들어 보였다. 스테이지 s에서의 u = $p_{n-1}p_{n-2} \cdots p_{s+1}0p_{s-1} \cdots p_1p_0$,d=$p_{n-1}p_{n-2} \cdots p_{s+1}1p_{s-1} \cdots p_1p_0$ 이다. 두 입력 $X_u(p-1)$과 $X_d(p-1)$이 메모리 뱅크로 접근시 충돌을 피하기 위해 두 개의 메모리 뱅크 M0와 M1으로 메모리를 할당하고 그 때의 주소를 각각 U_r , D_r이라 하자. U_r = D_r =$p_{n-1}p_{n-2} \cdots p_{s+1}p_{s-1} \cdots p_1p_0$이다. 즉, 인덱스의 이진 표현시 각 스테이지 번째 비트만 다르고 나머지 비트는 u와 d가 동일하다.

데이타	연산	주소	메모리 뱅크
$X_u(p-1)$	읽기	U_r	M0
$X_d(p-1)$	읽기	D_r	M1
$X_u(p)$	쓰기	U_w	M_{b0}
$X_d(p)$	쓰기	D_w	M_{b0}

표 1 . Butterfly 연산을 위한 메모리 어드레싱

S	l	g	u	u3u2u1u0	d	d3d2d1d0
0	0	0	0	0000	1	0001
		1	2	0010	3	0011
		2	4	0100	5	0101
		3	6	0110	7	0111
		4	8	1000	9	1001
		5	10	1010	11	1011
		6	12	1100	13	1101
		7	14	1110	15	1111
1	0	0	0	0000	2	0010
		1	4	0010	6	0110
		2	8	0100	10	1010
		3	12	1100	14	1110
	1	0	1	0001	3	0011
		1	5	0101	7	0111
		2	9	1001	11	1011
		3	13	1101	15	1111
2	0	0	0	.0000	4	0100
		1	8	1000	12	1100
	1	0	1	0001	5	0101
		1	9	1001	13	1101
	2	0	2	0010	6	0110
		1	10	1010	14	1110
	3	0	3	0011	7	0111
		1	11	1011	15	1111
3	0	0	0	0000	8	1000
	1	0	1	0001	9	1001
	2	0	2	0010	10	1010
	3	0	3	0011	11	1011
	4	0	4	0100	12	1100
	5	0	5	0101	13	1101
	6	0	6	0110	14	1110
	7	0	7	0111	15	1111

표 2 각 스테이지에 따른 인덱스 분석(N=16)

4. FFT 프로세서를 위한 인덱스 메모리 어드레싱

3장에서 보인 바와 같이 U_r = D_r , U_w = D_w 이므로 필요한 시프트 수는 반으로 감소한다. U_r , D_r 은 읽기 연산을 위한 주소부이고, U_w , D_w은 쓰기 연산을 위한 주소부이다. 표1에서 u와 d는 각각 2b와 2b+1을 스테이지 s 만큼 반복 시프트 시킨 것과 같음을 보인다. 단, b=0,1,⋯,n-1. 연산과정이

계속 진행됨에 따라 BF에서 계산된 결과를 입력이 있던 메모리에 저장함으로써 부가적인 메모리 요구 없이 입출력은 인덱스에 따라 차례로 정렬된다.

1)입력을 위한 메모리 어드레싱

샘플링된 데이터($X_k(-1)$)가 FFT 프로세스의 0스테이지에 동시에 접근하면 인덱스의 최하위 비트에 따라 메모리 뱅크를 선택한다.

2)출력을 위한 메모리 어드레싱

출력($X_k(n-1)$)이 저장된 메모리 뱅크는 입력을 위해 할당했던 인덱스의 최하위 비트에 따라 결정됨에 따라 각 메모리 뱅크내의 주소가 변경되면서 저장된다. $k = k_{n-1}k_{n-2}\cdots k_1k_0$.

k_0	메모리 뱅크	주소	
0	M0	$X_k(-1)$을 위한 주소 할당	$k_{n-1}k_{n-2}\cdots k_1$
1	M1		$k_{n-1}k_{n-2}\cdots k_1$
0	M0	$X_k(n-1)$을 위한 주소 할당	$k_{n-2}\cdots k_1k_{n-1}$
1	M1		$k_{n-2}\cdots k_1k_{n-1}$

표 3. BF 입출력을 위한 메모리 어드레싱

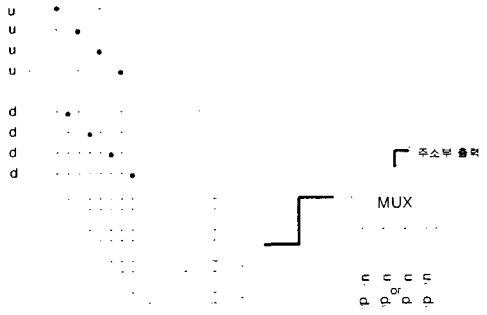

그림 3. 각 단의 주소부 생성기 블럭도(N=16)

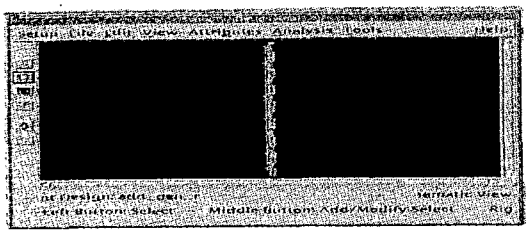

그림 4. 제안된 방법을 이용한
FFT 주소부 생성기

5. 결론

OFDM 시스템은 고속 데이터를 병렬처리하여 전송함으로써 고속 데이터 처리에 유리하다. 이러한 시스템에 사용되어지는 FFT는 OFDM의 성능을 좌우할 만큼 중요한 부분이다. 본 논문은 FFT 프로세서를 위한 효율적인 메모리 어드레싱 기법을 소개하고 실제 합성도를 보였다. 이 기법은 간단히 구현되어 하드웨어 복잡도를 줄이고 고속 시스템에 적합한 파이프 라인 FFT프로세서 구현에도 유용하다. 어드레스 발생을 간소화하고 coefficient 롬사용을 줄여 전력 소비면에서도 효율성을 보인다.

참고문헌

[1] D. Cohen, "Simplified control of FFT hardware," IEEE Trans. Acoust.,Speech Signal Processing, vol. ASSP-24, pp. 577-579, Dec. 1976.

[2] L.G. Johnson, "Conflict free memory addressing for dedicated FFT hardware," IEEE Trans. Circuit and Syst. II , vol. 39, pp. 312-316, May 1992.

[3] S.B. Weinsterin and Paul M. Ebert, "Data Transmission by Frequency-Division Multiplexing Using the Discrete Fourier Transform," IEEE Transaction on Communication Technology, Vol. COM-19, No.5, pp 623- 634, October 1971.

[4] A.V. Oppenheim, R.W. Schafer "Discrete-Time Signal Processing", Prentice hall, 1989.

[5] E.H.Wold, A.M. Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementation," IEEE Trans. on Computer, Vol. c-33, No. 5, pp.414-426, May, 1994.

[6] T.Widhe, J.Melander L.Wanhammar ,"Design of Efficient Radix-8 Butterfly PE for VLSI", 1997 IEEE Internatinal Symposium on circuits and systems Vol. 3, pp. 2084-2087.

[7] K. K. Parhi, "VLSI Digital Signal Processing", KLUWER ACADEMIC PUBLISHING, 1996.