2002년 한국정보처리학회 춘계학술발표논문집 제9권 제1호

OFDM을 위한 새로운 구조의 FFT 프로세서 설계

이 종 민*, 정 용 진*
*광운대학교 전자통신공학과
e-mail: sonnet@explore.gwu.ac.kr*

Design of a New FFT processor for OFDM

Jong-Min Lee* Yong-Jin Jeong*
Dept. of Electronic Communication Engineering,
Kwangwoon University

요 약

OFDM은 제4세대 변조기술로 일컬어지는 방식이다. 이는 최근 유럽에서 디지털 오디오 방송(DAB)과 디지털 비디오 방송(DVB)에 표준이 되었으며, IEEE 802.11a 무선 LAN 에서도 이 방식을 채택했고, ADSL, VDSL 등에서도 사용되어지고 있다. 본 논문에서는 이러한 OFDM 방식의 핵심이라고 할 수 있는 64포인트 FFT(Fast Fourier Transform)하드웨어 프로세서의 여러 가지의 구현된 예를 비교 분석하고, 가장 효율적인 방법인 Radix-2 SDF(Singlepath Delay Feedback)[1] 방법을 개선하여 새로운 구조를 제안하였다. 동일한 속도 성능을 가지는 여러구조 중에서 적은 수의 지연소자를 활용하여 FFT 크기를 작게 한 것이 SDF 방식으로 가장 널리 사용되고 있다. 본 논문에서는 SDF 방식이 내부적으로 4개의 복소곱셈기를 필요로 하는데 비해 2개의 복소곱셈기만을 사용하는 구조로 변형하고 컨트롤을 조절하여 새로운 구조를 설계하였다. 구현한 결과, FFT에서 전체 구조의 약 80%를 차지하는 복소곱셈기의 수를 절반으로 줄여 FFT 하드웨어 크기를 SDF 방식의 60% 정도로 줄일 수 있게 되었고, 이러한 구현방식은 64포인트 FFT만이 아닌 더 큰 크기의 FFT를 구현함에 있어서도 동일하게 적용할 수 있으며 현재 국내외에 발표된 논문 중 성능 대 면적비가 가장 우수한 구조이다.

1. 서 론

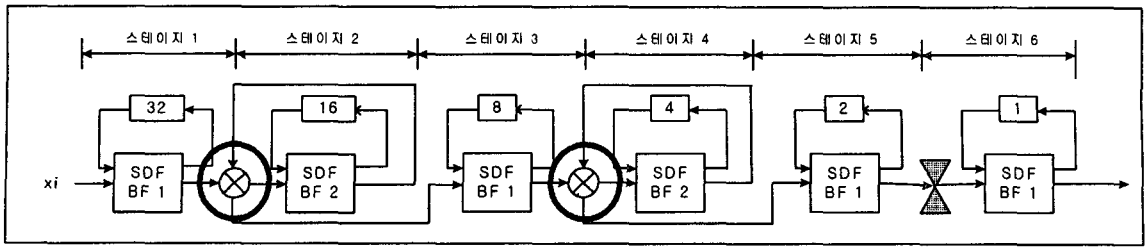
OFDM(Orthogonal Frequency Division Multiplexing)의 기본원리는 여러개의 반송파를 사용하여 고속 전송률을 갖는 입력 데이터를 낮은 전송률을 갖는 데이터로 반송파의 수만큼 병렬화 하여 각 반송파에 실어 전송하는 방식이다. 이때, 낮은 전송률을 갖는 부반송파의 심벌 duration이 증가하기 때문에 다중경로 지연확산에 의한 시간상에서의 상대적인 왜곡이 감소하고, 모든 OFDM 심벌 사이에 채널의 지연확산보다 긴 보호구간을 삽입하여 심벌간 간섭을 제거할 수 있다.[2]

OFDM 변복조는 다수의 부반송파를 사용하기 때문에 부반송파의 수가 많아지면 하드웨어 설계에 대한 어려움이 매우 크다. 또한 부반송파간의 직교성을 유지하기 위한 어려움이 발생하여 실제 시스템에 구현하기가 어려워진다. 이러한 문제점을 DFT(Discrete Fourier Transform)에 의해 구현할 수 있으며 DFT의 많은 연산량을 줄이기 위해 FFT 알고리즘을 이용하여 구현한다.[2]

FFT 알고리즘은 시간 데시메이션 또는 주파수 데시메이션에서 길이가 N인 DFT를 보다 작은 길이의 DFT로 연속해서 분해하여 계산하는 것이다. 이렇게 함으로써 DFT의 연산량에서 곱셈을 N^2번에서 $NlogN$번으로 줄일 수 있게 되는데,[2] 이러한 FFT의 효율적인 연산을 위해서 가장 많이 사용되는 알고리즘이 Radix-2 DIT(Decimation In Time) FFT와 DIF(Decimation In Frequency) FFT, Radix-4 DIT FFT와 DIF FFT이다.[3]

OFDM에서의 FFT는 실수부와 허수부를 갖는 복소수 연산을 수행한다. 따라서 하드웨어적으로는 실수부와 허수부가 분리되어 입력되어지는데, FFT 프로세서를 설계하여 입출력의 실수부와 허수부의 위치를 바꾸면 IFFT 역할을 수행할 수 있다.[4] FFT 알고리즘의 구현에는 크게 어레이 방식과 파이프라인 방식이 있는데, 어레이 FFT구조는 하드웨어적으로 매우 복잡하고 커져서 FFT의 연산점(N)이 큰 경우(256,2k,8k..etc)에 구현이 거의 불가능하다. 반면에 파이프라인 FFT 구조는 규칙적이고 비교적 제어가 간단하며 직렬 입력과 직렬 출력을 할 수 있기 때문에 높은 성능을 요구하는 응용분야에 가장 많이 사용하는 구조이다. 여러 가지 파이프라인 방식의 FFT 중에서 본 논문에서는 Radix-2 Singlepath Delay Feedback방식에서 복소곱셈기의 수를 1/2로 감소시키는 방법으로 64포인트 FFT를 설계하였다.

이 논문은 광운대학교 반도체 설계 교육 센터(IDEC)의 지원으로 연구되었음.



<그림-1> 제안한 구조의 FFT 전체 블록 다이어그램

2. Previous Work

파이프라인 방식으로 구현된 여러 가지 FFT의 구현 사례에서 각 스테이지간의 패스가 하나인지 여러개인지, 또한 Radix-2인지 Radix-4인지에 따라서 각각의 구조와 동작을 설명하였다.

• R2 MDC(Multipath Delay Commutator) [5]

각 스테이지 사이에 두 개의 패스가 있다. 입력 데이타 열을 두개의 병렬 데이터 열로 나누고 지연소자를 이용하여 두 데이타간의 거리를 조정하여 Butterfly(BF)[3]에 입력한다. 지연소자는 이전 스테이지의 1/2이며, BF는 1/2 주기(N/2 point)동안 동작하고 나머지 1/2은 쉬게 된다. 각 스테이지의 스위치는 앞 스테이지 스위치보다 두배 빠르게 스위칭한다.[5]

• R2 SDF(Singlepath Delay Feedback) [1]

각 스테이지 사이에 한 개의 패스가 존재한다. 입력 데이터열을 지연소자에 저장한 후 각 스테이지에 필요한 입력값이 들어올때에 지연소자에 있던 값과 BF 연산을 한다. 이때 BF출력의 일부를 피드백하여 비어있는 지연소자에 다시 저장하여 지연소자의 효율을 높임으로써 메모리 크기를 줄이는 방식이다. 이 구조는 BF, 복소곱셈기, 지연소자로 구성되어지며, BF는 N=16인 경우 첫 번째 스테이지에서 8개의 입력을 지연소자에 저장하고 그 다음 입력부터 식(1)을 연산한다.

$$B_i = A_i + A_{i+8}$$
$$B_{i+8} = A_i - A_{i+8} \qquad (1)$$

B_i는 두 번째 스테이지로 입력되고, B_{i+8}는 지연소자에 피드백 된다. B_i가 두 번째 스테이지에서 BF연산의 수행을 종료하면 B_{i+8}은 곱셈계수와 곱셈을 수행한다. 이때 두 번째 스테이지에서는 BF연산에 필요한 데이타들이 모두 입력되어 대기 시간 없이 BF 연산을 수행한다.[4]

• R4 MDC(Multipath Delay Commutator) [5]

Radix-2 MDC 방식의 Radix-4로써 각 스테이지 사이에는 4개의 패스가 있다. 입력 데이터 열을 4개의 병렬 데이터 열로 나누고 지연소자를 이용하여 네 개의 데이타의 거리를 조정하여 BF 에 입력한다. Radix-2 MDC와 마찬가지로 지연소자는 이전 스테이지의 1/2이며, BF는 1/2 주기(N/2 point)동안 동작하고 나머지 1/2는 쉬게 된다. 각 스테이지의 스위치는 이전 스테이지의 스위치보다 두배 빠르게 스위칭을 수행한다.[5]

• R4 SDF(Singlepath Delay Feedback) [6]

Radix-2 SDF방식의 Radix-4 방식으로써, BF 출력 3개의 값을 지연소자에 피드백 함으로써 메모리 크기를 줄인 방식이다.[6]

• R4 SDC(Singlepath Delay Commuatator) [7]

각 스테이지가 한 개의 패스로 이루어지고, 각 스테이지에서 BF 입력에 필요한 데이터를 얻도록 DC(Delay Commutator)에 지연소자를 두어 4개의 필요한 데이터를 BF 입력을 출력한다. 그리고 BF는 그 4개의 입력을 받아서 1개의 출력만 나온다. 다른 Radix-4 파이프라인 FFT 구조에 비해서 BF의 구조가 간단하지만 많은 수의 지연소자를 사용한다.[7]

기존의 64포인트 파이프라인 FFT 구조에서 Radix-2에서는 4개의 복소곱셈기를 사용하고, Radix-4에서는 많은 수의 지연소자를 사용하는 대신에 2개의 복소곱셈기를 사용하고 있다. 그러나 본 논문에서는 가장 효율적인 방법인 Radix-2 SDF FFT를 구현하되 50%의 이용율을 가지는 복소곱셈기를 100%로 늘리고 컨트롤을 조절하여 2개의 복소곱셈기만을 사용하는 새로운 구조를 구현하였다.

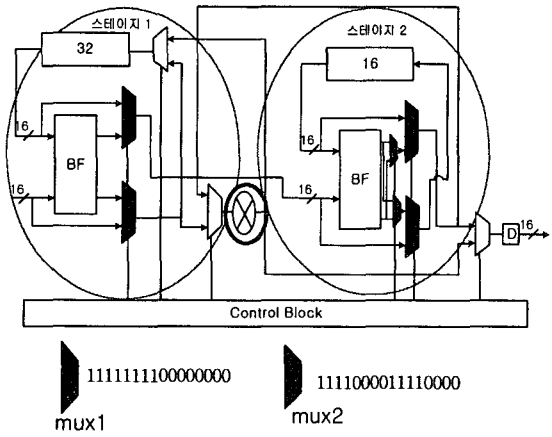
3. 제안된 구조의 FFT

제안된 구조는 SDF 방식을 변형하고 컨트롤을 조절함으로써 복소곱셈기 수를 절반으로 줄였다. 스테이지1은 32개의 지연소자를 가지고 있어서 32포인트 떨어진 두 복소수 입력 a와 b에 대해서 식(1)의 BF 연산을 수행한다. 두 입력 a와 b의 a+b는 스테이지2의 입력이 되고, a-b가 된 값은 피드백 되어 지연소자에 다시 저장 된다. 이렇게 하여 a+b가 모두 연산된 후 스테이지2로 입력되면 그때 지연소자에 저장되어 있던 a-b는 복소곱셈기에서 곱셈계수와 연산이 된 후 스테이지2로 입력된다. 이때 복소곱셈기는 a-b에서만 연산이 되기 때문에 그 효율이 50%지만, 본 논문에서는 스테이지2에서의 복소곱셈기에서 연산되는 시간을 조절하여 두 개의 50% 효율을 가지는 복소곱셈기를 한 개의 동일한 구조를 갖는 100% 효율의 복소곱셈기를 사용하는 구조로 설계하였다. 스테이지3과 스테이지4에서도 이와 같이 복소곱셈기 사용하는 시간을 조절하여 100% 효율의 하나의 복소곱셈기만을 사용하였다. 그림1은 두 스테이지에서 한 개의 복소곱셈기를 가지는 구조의 전체 블록 다이어그램이다.

스테이지1,2 에서 복소곱셈기 하나를 서로 공유한 구조를 그림2에 보였다. 스테이지1에서 a+b와 a-b가 동시에 연산되고, a-b는 곱셈계수와 곱해져서 지연소자로 피드백 된다. a-b가 피드백 될 때에 a+b는 스테이지2로 입력되어 연산을 하는데 스테이지2의 구조는 스테이지1의 구조와

다르다. 스테이지1의 연산에서 복소곱셈을 하는 시간에 스테이지2는 복소곱셈을 하지 않는 a+b가 먼저 연산되고, 스테이지1의 a+b가 모두 출력되는 시간이 스테이지1의 복소곱셈이 끝나는 시점이므로 그 다음부터는 스테이지2의 a-b가 피드백되고 복소곱셈기에 입력되어 연산을 한 후 스테이지3으로 입력된다.

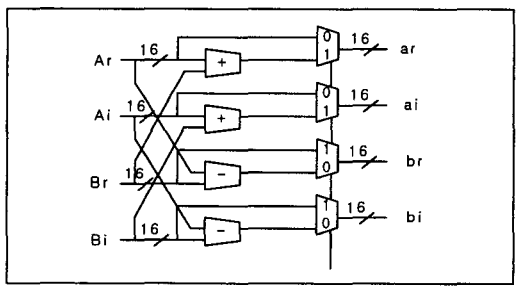
이렇게 함으로써 R2-SDF 방식보다는 2개의 스테이지에서 mux4개와 레지스터 1개가 더 소모되므로 전체에서 8개의 mux와 2개의 레지스터가 더 사용된다. 반면에 복소곱셈기의 수를 전체 구조에서 절반으로 줄이는 효과를 가지게 된다.



<그림-2> 두 스테이지의 내부 구조

3.1 BF(Butterfly)

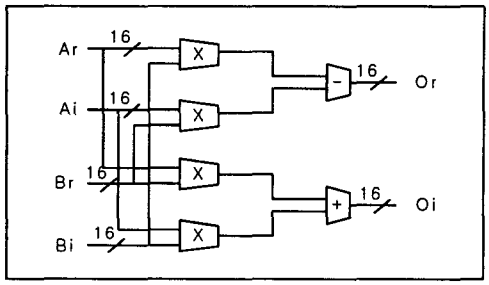
두 복소수 입력 데이터 $a = A_r + A_i$ 와 $b = B_r + B_i$를 각각 실수부와 허수부로 나누어서 연산을 한다. BF는 그림3과 같이 4개의 덧셈기와 4개의 mux로 이루어졌으며, 각각의 출력은 $a_r = (A+B)_r$, $a_i = (A+B)_i$, $b_r = (A-B)_r$, $b_i = (A-B)_i$ 이다.



<그림-3> Butterfly 구조

3.2 복소곱셈기

두개의 복소수 입력을 각각 실수부와 허수부로 나누어서 연산한다. 그림4와 같이 4개의 곱셈기와 2개의 덧셈기로 이루어졌기 때문에 지연소자, BF, 곱셈계수 테이블등과 함께 이루어진 전체 FFT 구조에서 복소곱셈기는 약 85%의 하드웨어 크기를 점유한다.

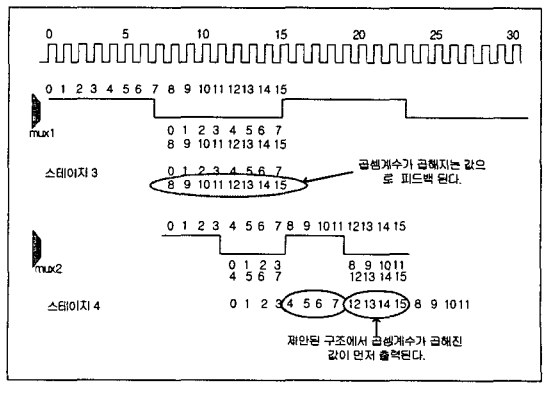


<그림-4> 복소곱셈기 구조

3.3 컨트롤 설계

두 개의 스테이지를 합쳐서 타이밍을 조절하고 복소곱셈기 수를 줄였으므로 두 개의 스테이지 중에서 스테이지2의 컨트롤이 달라진다. 즉 스테이지1에서 복소곱셈기를 사용하지 않는 시간에 스테이지2에서 복소곱셈기를 사용해야 하기 때문이다. R2-SDF에서는 두 복소수 $a = A_r + A_i$와 $b = B_r + B_i$가 BF에서 $a+b$ 와 $a-b$를 동시에 연산한 후에 $a+b$는 스테이지2로 입력되고, $a-b$는 피드백 되었다가 $a+b$이 모두 스테이지2로 입력된 후에 복소곱셈을 하고 스테이지2로 입력된다.

그러나 본 논문에서 제안한 방식은 BF에서 $a+b$은 다음 스테이지로 입력되고 동시에 $a-b$은 복소곱셈기에서 곱셈계수와 곱해진 후에 피드백 되어 지연소자에 저장된다. 그리고 $a+b$가 모두 스테이지2로 입력된 후에 $a-b$도 스테이지2로 입력되는데 이때는 복소곱셈을 하지 않고 스테이지2로 입력되기 때문에 그림2의 mux 컨트롤처럼 스테이지2에서는 곱셈계수와 곱해져야 하는 스테이지2의 $a'-b'$이 복소곱셈기에서 연산되도록 컨트롤을 해주었다. 이렇게 함으로써 단 한클럭의 지연도 없이 전체 FFT는 계속해서 연산이 이루어지며, 기존의 사례에서 Radix-2 MDC나 SDF 방식과 동일한 속도의 성능을 가진다. 자세한 컨트롤 내부 FSM(Finite State Machine) 함수는 [8]에 나타나 있으며, 16포인트에서 데이터의 입출력을 클럭에 맞추어 그림5에 나타내었다. 이것은 전체 FFT 구조에서 스테이지3과 스테이지4의 구조에서의 입출력 타이밍을 보



<그림-5> 제안된 구조의 FFT 타이밍도

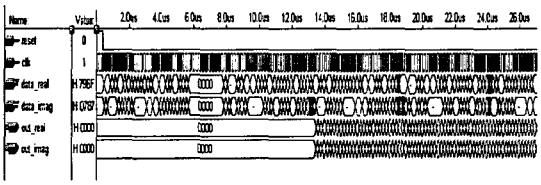
인 것이다. 스테이지3에서 곱셈계수가 곱해지는 부분(연산점8~15)과 스테이지4에서 곱셈계수가 곱해지는 부분(연산점 4,5,6,7,12,13,14,15)이 서로 교차되어 복소곱셈기는 100%의 이용율로 계속 연산한다. 이렇게 하여 64포인트 FFT 연산을 모두 마쳤을 때 출력되는 데이터 열을 표1에 나타내었다. 기존의 Bit Reverse 출력 열과 조금은 다른 순서의 출력이 나오는데, 그것은 복소곱셈기의 효율을 높이기 위해서 스테이지2와 스테이지4의 출력순서에 변화를 주었기 때문이다.

0	32	16	8	40	24	56	12	44	28	60	4	36	20	52
2	34	18	10	42	26	58	14	46	30	62	6	38	22	54
3	35	19	11	43	27	59	15	47	31	63	7	39	23	55
1	33	17	9	41	25	57	13	45	29	61	5	37	21	53

<표-1> 제안된 구조의 FFT 출력 데이터 열

4. 결 론

본 논문에서 제안한 하드웨어 구조에 대해 Matlab 시뮬레이션을 통해 FFT 연산된 값과 제안된 구조의 FFT의 결과값을 비교하여 검증하였다. 첫 번째 입력 값이 FFT에 입력되어 연산을 모두 마치고 출력될 때까지 총 클럭 지연은 66클럭이다. 그림6은 시뮬레이션에서 Matlab으로 뽑은 테스트 벡터를 넣었을 때 66번째 클럭에서부터 출력이 나오는 것을 보이고 있다.



<그림-6> 제안된 구조의 FFT 시뮬레이션 결과

제안된 구조로 64포인트 FFT를 설계하였을 때, 표2에서 보듯이 기존의 파이프라인 FFT 방식에서 Radix-2에서는 복소곱셈기를 $2(\log_4 N-1)$개, 즉 4개를 사용하였는데, 본 논문의 구조로 설계하여 복소곱셈기의 이용율을 100%로 높임으로 인하여 2개만을 사용하였다.

구 조		BF이용율	복소곱셈기수	복소곱셈기 이용율	지연소자수	덧셈기수
R2	MDC	50%	4	50%	94	12
	SDF	50%	4	50%	63	12
	제안된구조	**50%**	**2**	**100%**	**65**	**12**
R4	MDC	25%	6	25%	156	24
	SDF	25%	2	75%	63	24
	SDC	100%	2	75%	126	18

<표-2> 제안된구조와 여러가지 파이프라인 FFT 구조의 비교[9]

한 개의 복소곱셈기는 4개의 곱셈기와 2개의 덧셈기로 이루어지므로, 본 논문에서 구현한 구조는 복소곱셈기 수를 절반으로 줄이기 때문에 동일한 속도를 가지는 기존의 사례보다 성능 대 면적비가 가장 우수하다. 64포인트 R2-SDF와 비교하여 삼성 0.5um CMOS 스탠다드 셀 라이브러리[10]를 근거로 예측하여 표3에 나타내었다. 제안된 구조는 약 52000개의 게이트로서 R2-SDF의 약 60% 정도의 하드웨어 리소스를 가지게 됨을 볼 수 있다. 또한 더 큰 크기의 FFT에 사용함에 있어서 제안된 구조의 효과를 표4에 나타내었다. 1k,4k,8k와 같이 하드웨어 리소스를 많이 갖는 FFT에서 본 논문의 제안된 구조는 더욱 효과적으로 적용 될 것으로 보아 앞으로 DAB, DVB, 지문인식 등에 널리 활용이 가능할 것으로 기대된다.

	R2-SDF		제안된 구조	
	수량	게이트 수	수량	게이트 수
지연소자	63	4536	65	4680
Butterfly(BF)	6	5491	6	5491
복소곱셈기	4	67724	**2**	**33862**
기 타		7000		8000
합 계		84751		**52033**

<표-3> R2-SDF와 제안된 구조의 하드웨어 리소스 비교

	R2-SDF		제안된 구조	
	복속곱셈기 수	BF 수	복소곱셈기 수	BF 수
256	6	8	**3**	8
1k	8	10	**4**	10
4k	10	12	**5**	12
16k	11	13	**6**	13

<표-4> R2-SDF와 제안된 구조의 여러 가지 크기의 FFT 비교

5. 참고문헌

[1] E.H. Wold and A.M. Despain, "*Pipeline and parallel pipeline FFT processors for VLSI inplenentation,*", IEEE tran. Comput C-33(5), p414~426, 1984.

[2] R.Van Nee and R. Prasad, "*OFDM for Wireless Multimedia Communication*", Artech House, p33~50, 2000.

[3] Alan V. Oppenheim and Ronald W. Schafer, "*Discrete Time Signal Processing*", Prentice Hall, p581~605, 1989.

[4] 김재석 조용수 조중휘, "*이동통신용 모뎀의 VLSI 설계*", 대영사, p322~329, 2001.

[5] L.R. Raviner and B. Gold, "*Theory and Application of Digital Signal Processing*", Prentice-Hall, Inc, 1975.

[6] A.M. Despain, "*Fourier transform computer using CORDIC iterations,*", IEEE trans, Comput. C-23(10), p993~1001, 1974.

[7] G.Bi and E. V. Jones, "*A pipelined FFT processor for word-sequential data,*", IEEE Trans. Acoust. Speech, Signal Processing, 37(12), p1982~1985, 1989.

[8] 이종민 정용진, "*Design of FFT Processor in OFDM,*", RTA-TR-2001-02, p3~20, 2002. (under preparation).

[9] S. He and M. Torkelson, "*A new approach pipeline FFT processor,*", IEEE Proceedings of IPPS, p766~770, 1996.

[10] Samsung Electronics, "*ASIC STD85/STDM85 0.5um High Density CMOS Standard Cell Library*", September 1997.