논 문
58-6-28

FFT를 위한 효율적인 Signal Reordering Unit 구현

Efficient Signal Reordering Unit Implementation for FFT

양 승 원* · 이 종 열†
(Seung-Won Yang · Jang-Yeol Lee)

Abstract - As FFT(Fast Fourier Transform) processor is used in OFDM(Orthogonal Frequency Division Multiplesing) system. According to increase requirement about mobility and broadband, Research about low power and low area FFT processor is needed. So research concern in reduction of memory size and complex multiplier is in progress. Increasing points of FFT increase memory area of FFT processor. Specially, SRU(Signal Reordering Unit) has the most memory in FFT processor. In this paper, we propose a reduced method of memory size of SRU in FFT processor. SRU of 64, 1024 point FFT processor performed implementation by VerilogHDL coding and it verified by simulation. We select the APEX20KE family EP20k1000EPC672-3 device of Altera Corps. SRU implementation is performed by synthesis of Quartus Tool. The bits of data size decide by 24bits that is 12bits from real, imaginary number respectively. It is shown that, the proposed SRU of 64point and 1024point achieve more than 28%, 24% area reduction respectively.

Key Words : SRU, Bit-reverse, FFT, OFDM

1. 서 론

FFT (Fast Fourier Transform) 프로세서는 OFDM (Orthogonal Frequency Division Multiplexing) 시스템에서 이용된다. OFDM 시스템은 WiBro, IEEE 802.11, IEEE 802.16, DAB, ADSL, VSDL, DVB-T 등의 표준기술로 채택하고 있다. 이와 같이 여러 응용분야의 OFDM 시스템에서 다양한 FFT 프로세서가 요구된다. 또한 근래에는 광대역과 이동성에 대한 요구가 높아짐에 따라 고성능, 큰 포인트의 FFT 프로세서가 요구되고 있고 있다. 그에 따라 저 전력, 저면적 FFT 프로세서의 연구가 필요하다. 간단하게 OFDM 시스템을 보면 그림 1과 같이 FFT 프로세서와 SRU (Signal Reordering Unit)로 구성된다. FFT 프로세서의 포인트 수가 높아질수록 메모리가 차지하는 면적은 증가한다. FFT 연산은 입력 시퀀스와 출력 시퀀스가 다름으로 인해 FFT 프로세서는 연산 시 반드시 신호를 재 정렬하는 블록이 필요하다. 신호를 재 정렬 하는 블록인 SRU는 FFT 프로세서에서 가장 큰 크기의 메모리를 가진다[1][2].

본 논문에서는 Radix-2 알고리즘을 기반으로 하는 파이프라인 FFT 프로세서에 사용되는 SRU의 구조를 변경하여 메모리의 크기를 줄이는 방법을 제안한다. 2장에서는 FFT에 간단히 알아보고 파이프라인 FFT 구조에 따르는 요구되는 구성요소의 크기에 대해 알아보고 Radix-2 알고리즘 FFT의 비트리버스에 대해 설명한다. 3장에서는 기존의 SRU에 알아보고 제안한 SRU의 구조에 대해 설명한다. 4장에서는 Altera 사의 Quartus 툴을 이용하여 기존의 SRU와 제안한 SRU를 구현하고 결과를 비교하고 5장에서 결론을 맺는다.

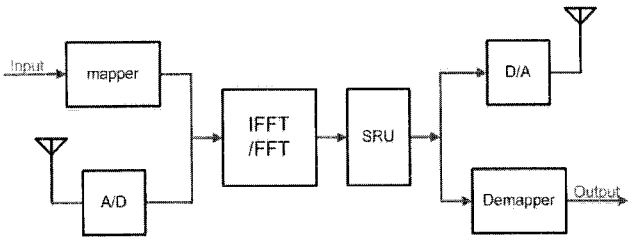

그림 1 OFDM 시스템의 FFT와 SRU 구성도
Fig. 1 FFT and SRU structure in OFDM system

† 교신저자, 정회원 : 전북대학교 전자 정보 공학부 교수
E-mail : yswon@chonbuk.ac.kr
* 준 회 원 : 전북대학교 전자 정보 공학부 석사과정
접수일자 : 2008년 11월 11일
최종완료 : 2009년 4월 21일

2. FFT

2.1 FFT 개요

N포인트의 DFT (Discrete Fourier Transform)는 N^2의 복소 곱셈과 $N(N-1)$의 복소 덧셈을 요구한다. 큰 크기의 N을 가질 경우 그 계산 량이 방대하여 구현의 어렵고 계산 시간이 비례적으로 커지게 된다. 그래서 계산 량을 줄여 계산 속도를 간소화하는 방법을 가리켜서 FFT라 한다. 다양한 FFT 기법이 있지만 그중에서 FFT 설계에 가장 효율을 가져다준 방식은 Cooley와 Tukey가 제안한 Cooley-Tukey 알고리즘이다. 이 알고리즘은 만일 N이 합성수일 경우 훨씬 적은 곱셈과 덧셈만을 갖는 형태로 바꿀 수 있음을 보였다. 이러한 기법에는 입력 시퀀스를 재배치하는 DIT (Decimation in Time) 구조와 출력 시퀀스를 재배치하는 DIF (Decimation in Frequency) 구조가 있다. Cooley-Tukey 알고리즘을 기반으로 효율을 높이기 위해 Radix-2, Radix-4 Radix-8, Split-radix, Radix-2^2, Radix-2^3, Radix-2^4, Radix-4/2, Radix-2/4/8 등의 다양한 알고리즘이 제안되었다 [3][4][5].

FFT 프로세서는 구조적인 측면으로 단일 메모리 구조 방식, 이중메모리 방식, 병렬처리 프로세스 방식과 파이프라인 방식 등이 제안되었다. 그 중 파이프라인 방식은 고속처리가 가능하고 면적대비 처리속도 관점에서 우수하며, 비교적 구조가 간단하여 큰 포인트의 FFT 프로세서 설계에 용이하여 큰 포인트를 요구하는 OFDM 시스템에서 사용된다. 파이프라인 방식의 종류에는 기본적으로 SDC (Single-Path Delay Commutator), MDC (Multipath Delay Commutator), SDF (Single-Path Delay Feedback) 가 있다. 표 1과 같이 높은 포인트를 갖는 FFT 프로세서를 설계할 때 복소 곱셈기와 메모리 관점에서 가장 효율적인 파이프라인 구조는 R22SDF임을 알 수 있고, 포인트가 낮을 경우는 R4SDC구조가 가장 효율적인 구조임을 알 수 있다 [5]. 표 1과 같이 N포인트의 FFT 프로세서의 경우 R2MDC FFT는 3N/2-2, R4MDC FFT는 5N/2-4, SDC FFT는 2N-2, SDF FFT는 N-1 크기의 메모리가 소요 된다. 하지만 FFT 프로세서에 신호를 재 정렬하기 위해 사용되는 SRU는 표 1의 FFT 메모리와 별도로 2N 크기의 메모리를 요구한다 [3][4][5].

표 1 파이프라인 FFT 구조에 따르는 구성요서 비교
Table 1 Comparison of the pipelined FFT architectures

구조	Multiplier	Adder	Memory
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	$\log_4 N-1$	$3\log_4 N$	$2N-2$
R2^2SDF	$\log_4 N-1$	$4\log_4 N$	$N-1$

2.2 비트 리버스

FFT 연산 시 입력시퀀스와 출력시퀀스가 다름으로 인해 반드시 신호를 재 정렬해야 한다. 그림 2는 Radix-2 알고리즘의 8포인트 DIF FFT 신호 흐름도이다. 입력시퀀스 k가 "0, 1, 2, 3, 4, 5, 6, 7" 일 때 출력시퀀스 n은 "0, 4, 2, 6, 1, 5, 3, 7" 로 나오는 것을 알 수 있다. 따라서 원하는 출력 시퀀스를 알기 위해선 신호를 재 정렬해야 한다. Radix-2 알고리즘의 FFT는 신호의 주소를 비트리버스 함으로 신호를 재 정렬 할 수 있다. 표 2의 첫 번째 열은 FFT의 출력 시퀀스 주소 값이고 두 번 번째 열은 그 주소 값을 이진수로 표시한 것이다. 세 번째 열은 그 주소 값을 비트 리버스한 것으로써 세 번째의 주소 값은 제 정렬된 출력 시퀀스의 주소임을 알 수 있다. 이와 같이 Radix-2 알고리즘의 경우 출력 시퀀스의 이진표기를 통해 그 주소 값을 비트 리버스하여 신호를 재 정렬된 시퀀스를 알 수 있다 [6].

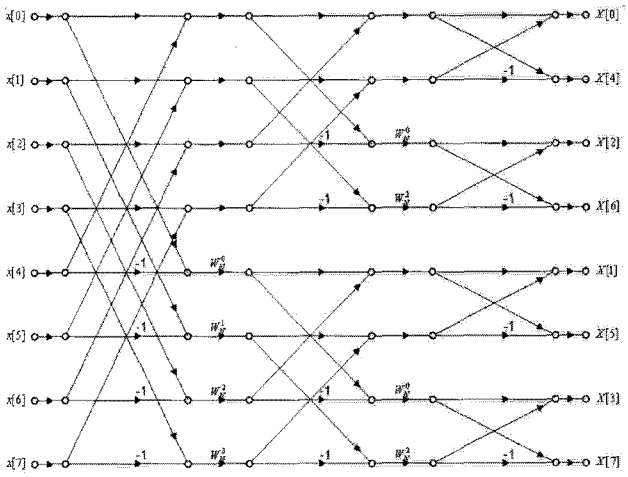

그림 2 8-point DIF FFT 신호 흐름도
Fig. 2 8-point DIF FFT signal flow graph

표 2 DIF (Decimation in Frequency) 비트리버스
Table 2 DIF (Decimation in Frequency) bit-reverse

FFT 출력 시퀀스		재 정렬된 데이터	
십진표기	이진표기	이진표기	십진표기
0	000	000	0
4	100	001	1
2	010	010	2
6	110	011	3
1	001	100	4
5	101	101	5
3	011	110	6
7	111	111	7

3. Signal Reordering Unit

3.1 기존의 SRU

데이터 재 정렬을 수행할 때 입력 시퀀스와 출력 시퀀스가 다름으로 인하여 ex1)과 같이 데이터를 읽기 전에 덮어쓰는 overwrite현상 발생으로 인해 기존의 SRU는 N 크기의 RAM을 2개를 사용하였다.

ex1) 8point FFT의 RAM 1개로 구성한 SRU 시퀀스

	write address	input data	read address	output data
step 0	0	0	0	x
step 1	4	4	1	x
step 2	2	2	2	x
step 3	6	6	3	x
step 4	1	1	4	x
step 5	5	5	5	x
step 6	3	3	6	x
step 7	7	7	7	x
SP → step 8	0	8	0	0
step 9	4	12	1	1
step 10	2	10	2	2
step 11	6	14	3	3
step 12	1	9	4	12
step 13	5	13	5	5
step 14	3	11	6	14
step 15	7	15	7	7

1) step0 ~ step7에서 bit-reverse된 write address로 데이터를 저장.
2) step8 ~ step15에서 write, read address로 데이터를 저장과 읽기를 수행.
step9, 11에서 어드레스 4, 6에서 데이터 12, 14를 덮어쓰게 된다.
step12, 14에서 덮어 쓰인 데이터 12, 14가 출력된다.

신호를 재 정렬을 하기위한 기존의 SRU는 그림 3과 같은 ping-pong 구조를 가지고 있다. 기존의 SRU의 동작은 RAM을 2개를 사용하여 N을 주기로 처음주기에 RAM0 쓰기 신호를 on하고 어드레스 신호를 비트리버스한 주소 값의 순서대로 데이터 저장한다. RAM1은 쓰기신호를 off하고 MUX에서 RAM1 의 데이터를 선택한다. 다음 주기에 RAM0의 쓰기 신호를 off 하고 데이터를 재 정렬된 순서로 출력할 때 MUX에서 RAM0의 데이터를 선택한다. 이 때 RAM1의 쓰기동작 on하여 비트리버스한 순서대로 데이터 저장한다. 이와 같은 시퀀스를 반복함으로서 재 정렬 작업이 수행된다. N포인트 FFT일 경우 N사이즈 2개의 RAM 블록이 요구되므로 N의 크기에 따라 큰 메모리가 소요된다.

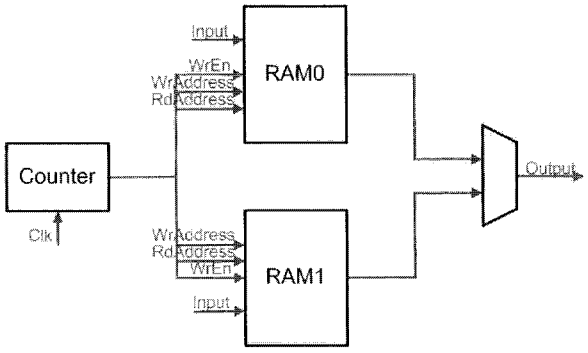

그림 3 기존의 Signal Reordering Unit (SRU)
Fig. 3 Conventional Signal reordering Unit (SRU)

3.2 제안한 SRU

제안한 SRU의 동작은 ex2)와 같이 FFT의 연 산후 나온 데이터를 비트리버스한 순서대로 RAM에 저장한다. 이때 overwrite되는 데이터는 FIFO에 저장한다. 다음 시퀀스에서 입력, 출력 시퀀스 데로 데이터를 입력, 출력한다. FIFO에서 출력되는 데이터는 RAM에 저장되고 overwrite가 발생되는 부분의 데이터가 FIFO에 저장 후 다음 시퀀스에서 RAM에 저장함으로써 overwrite를 방지하고 재 정렬 작업을 수행할 수 있다.

ex2) 8point FFT의 RAM 1개와 FIFO로 구성한 SRU 시퀀스

	FIFO in	FIFO out	write address	input data	read address	output data
step 0	-	-	0	0	0	x
step 1	4	-	4	x	1	x
step 2	-	-	2	2	2	x
step 3	6	-	6	x	3	x
step 4	-	-	1	1	4	x
step 5	-	-	5	5	5	x
step 6	-	-	3	3	6	x
step 7	-	-	7	7	7	x
SP → step 8	-	-	0	8	0	0
step 9	12	4	4	4	1	1
step 10	-	-	2	10	2	2
step 11	14	4	6	6	3	3
step 12	-	-	1	9	4	4
step 13	-	-	5	13	5	5
step 14	-	-	3	11	6	6
step 15	-	-	7	15	7	7

1) step0 ~ step7에서 bit-reverse된 write address로 데이터를 저장.
step1, 3에서 데이터 4, 6을 FIFO에 저장한다.
2) step8 ~ step15에서 write, read address로 데이터를 저장과 읽기를 수행.
step9, 11에서 FIFO에서 데이터를 불러와 어드레스 4, 6에서 데이터 4, 6을 저장한고. FIFO에 데이터 12, 14를 저장한다.
step12, 14에서 원하는 데이터 4, 6이 출력된다.

제안한 SRU의 구성은 그림 4와 같다, 제안한 SRU의 동작은 RAM을 1개와 FIFO를 사용하여 N을 주기로 비트리버스 된 주소 값의 순서대로 데이터를 RAM에 쓰고 올바른 순서대로 RAM에서 읽는다. 이때 overwrite현상이 발생하는 부분은 MUX에서 FIFO의 출력을 선택하여 RAM에 쓰고 입력은 FIFO에 저장한다. 이와 같은 시퀀스를 반복함으로서 재 정렬 작업이 수행된다. 제안한 SRU는 표 3과 같이 레지스터를 요구함으로 제안한 SRU의 구성은 N의 크기의 RAM 1개와 overwrite현상이 발생되는 부분이 n이라 하면 n크기를 갖는 FIFO, FIFO를 제어 하기위한 조합회로로 제안된 SRU를 구성할 수 있다.

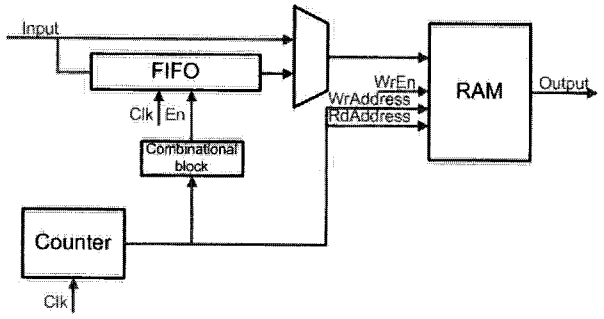

그림 4 제안한 Signal Reordering Unit (SRU)
Fig. 4 Proposed Signal Reordering Unit (SRU)

표 3 SRU 메모리 비교
Table 3 Comparison of the number of SRU memory

	64포인트	256포인트	1024포인트
기존	2N	2N	2N
제안	1.44N	1.46N	1.49N

FFT 프로세서의 포인트의 수가 증가하면 overwrite현상되는 저장하는 FIFO의 크기가 증가한다. 큰 포인트의 FFT 프로세서의 경우 FIFO를 레지스터로 구현하면 불필요한 데이터의 이동이 많아 소비전력이 커지게 된다. 이와 같은 경우 소비전력을 줄이기 위해 FIFO를 RAM으로 구성하였다. 그림 5는 FIFO를 RAM으로 구성한 제안된 SRU의 구성도이다 [4]. FIFO를 RAM으로 구성할 경우 N의 크기를 갖는 RAM 1개와 FIFO를 구성할 1/N크기를 갖는 RAM 1개 FIFO를 제어하기 위한 조합회로와 카운터로 구성할수 있다.

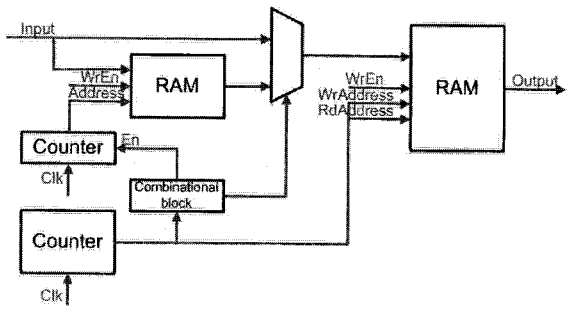

그림 5 제안한 Signal Reordering Unit (SRU)
Fig. 5 Proposed Signal Reordering Unit (SRU)

4. 구 현

64, 1024포인트 FFT 프로세서에 사용되는 기존의 SRU와 제안된 SRU를 Verilog 코딩하여 시뮬레이션 하여 동작을 확인 하였다. Altera사의 APEX20KE 계열의 EP20k1000EPC 672-3 디바이스를 선택하였고 Quartus툴을 사용하여 SRU를 합성하여 구현하였다. RAM의 크기는 데이터 비트를 실수부, 허수부 각 12bits인 24bits의 크기를 가지는 RAM과 FIFO를 MegaWizerd를 이용하여 생성하여 사용하였다. 합성한 결과 표 4와 같이 데이터의 크기가 24bit 일 때 면적의 관점에서 약간의 로직을 추가되지만 64, 1024 포인트에서 각 28, 24%의 이득을 얻을 수 있다.

표 4 합성 결과 비교
Table 4 Synthesis result comparison

		기존	제안	이득(%)
64 포인트	Logic elements	38	69	-81.6
	Memory bit	3072	2208	28.1
1024 포인트	Logic elements	48	91	-89.6
	Memory bit	49152	37528	24.2

4. 결 론

본 논문에서는 파이프라인 Radix-2 FFT 프로세서에 적용할 수 있는 SRU를 제안하였고 Wibro용 1024 포인트 R2^{4}SDF FFT 프로세서에 적용하였다. SRU는 FFT 프로세서에서 가장 큰 메모리를 가지고 있다. FFT의 출력 데이터 시퀀스의 overwrite되는 부분만을 먼저 처리함으로 SRU의 메모리의 크기를 줄일 수 있음을 알 수 있다.

본 논문에서 제안한 SRU는 큰 포인트를 가지고, 송수신이 되는 OFDM 시스템에 많이 응용될 것이다.

참 고 문 헌

[1] Xiaofei Dong, "Implementing OFDM modulation for wireless communications", www.embedded.com

[2] F. Kristensen, P. Nilsson, A. Olsson, "Flexible Baseband Transmitter for OFDM", Circuits, Signals, and Systems, 2003

[3] Jung-yeol Oh, Myoung-seob Lim, "Area and Power Efficient Pipeline FFT Algorithm", Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on, pp. 520 - 525, 2-4 Nov. 2005

[4] Shousheng He, Torkelson, M, "Designing pipeline FFT processor for OFDM (de)modulation", Signals, Systems, and Electronics, 1998. ISSSE 98. 1998 URSI International Symposium on, pp. 257 - 262, Oct. 1998

[5] Son, B.S. Jo, B.G. Sunwoo, M.H. Yong Serk Kim,

"A high-speed FFT processor for OFDM systems", Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on, pp. III-281 - III-284 vol.3, 26-29 May 2002.

[6] Paul A.Lyun, Wolfgang Fuerst, "Introductory Digital Signal Processing" WILEY 1998.0

저 자 소 개

양 승 원 (梁 勝 元)

2008년 군산대학교 전자정보 공학부 학사 졸업. 2008년~현재 전북대하교 전자정보 공학부 석사과정

이 종 열 (李 宗 烈)

1993년 한국과학기술원 전자전산학과 학사 졸업. 1996년 한국과학기술원 전자전산 학과 석사 졸업. 2002년 한국과학기술원 전자전산 학과 박사 졸업. 2002년 9월~2003년 9월 하이닉스 반도체 선임연구원. 2003년 10월~2004년 2월 한국과학기술원 BK21 초빙 교수. 2004년 3월~2006년 3월 전북대학교 전자정보공학부 전임강사. 2006년 4월~현재 전북대학교 전자정보공학부 조교수