논문 2011-06-31

IEEE 802.11n WLAN을 위한 FFT 프로세서의 하드웨어 복잡도 최적화에 대한 연구

(A Study on Optimization of Hardware Complexity of a FFT Processor for IEEE 802.11n WLAN)

최 락 훈*, 박 정 준, 임 태 민, 이 진 용, 김 영 록
(Rakhun Choi, Jungjun Park, Taemin Lim, Jinyong Lee, Younglok Kim)

Abstract : A FFT/IFFT processor is the key component for orthogonal frequency division multiplexing (OFDM) systems based IEEE 802.11n wireless local area network (WLAN). There exists many radix algorithms according to the structure of butterfly as FFT sub-module, each has the pros and cons on hardware complexity. Here, mixed radix algorithms for 64 and 128 FFT/IFFT processors are proposed, which reduce hardware complexity by using mixture of radix-23 and radix-4 algorithms. The proposed algorithm finish calculation within 3.2μs in order to meet IEEE 802.11n standard requirements and it has less hardware complexity compared with conventional algorithms.

Keywords : FFT, Mixed radix, OFDM, IEEE 802.11n, WLAN

1. 서 론

IEEE 802.11n 표준은 기존의 IEEE 802.11a/g와 달리 높은 데이터 처리율을 제공하기 위해 MIMO(multiple input multiple output) 기술과 40MHz 채널 대역폭을 가진 물리 계층 등을 적용한다. 또한 차세대 이동통신과 방송 시스템인 LTE(long term evolution), DVB-T(digital video broadcasting-terrestrial), WiMAX(world wide interoperability for microwave access), Wibro (wireless broadband)등의 시스템과 같이, 전송 대역폭과 주파수 효율의 극대화를 위해 OFDM 적용을 표준으로 하고 있다 [1-3].

OFDM 기반 시스템에서 다수 반송파의 변복조를 담당하는 블록인 FFT/IFFT(fast fourier transform/inverse fast fourier transform)는 DFT(discrete fourier transform)을 빠르게 처리할 수 있는 알고리즘으로, 디지털 신호처리 및 고속 유무선 디지털 통신 시스템에 폭넓게 사용된다. 하지만 FFT/IFFT 프로세서는 OFDM 기반 시스템의 서브 모듈 중 가장 많은 연산 수행이 요구됨에 따라 전체 하드웨어 복잡도의 매우 큰 부분을 차지한다. 따라서 이를 줄이기 위한 많은 연구 결과가 발표되고 있다 [4-6]. 기존에 제안된 FFT/IFFT의 구조에는 병렬구조, 파이프라인구조, 메모리기반구조 등이 있으며, 그 중 파이프라인구조는 빠른 실시간 처리가 가능하고 전력 소모가 적어 고속처리가 요구되는 응용 분야에 적합하다 [7].

FFT/IFFT 프로세서는 서브모듈인 버터플라이의 구조에 따라 많은 radix 알고리즘이 존재한다. 높은 radix 알고리즘은 낮은 radix 알고리즘에 비해 더 적은 수의 복소 곱셈 연산을 요구하므로 연산 복잡도를 줄일 수 있지만, 버터플라이의 하드웨어 복잡도가 커지게 되는 단점이 있다 [8]. FFT/IFFT 프로세서의 모든 스테이지에 같은 radix 값을 사용하는 고정 radix 알고리즘은 연산구조가 규칙적이므로 VLSI 설계에 적합하다. 하지만 radix 값의 크기

* 교신저자(Corresponding Author)
논문접수 : 2011. 01. 31., 수정일 : 2011. 02. 06.,
채택확정 : 2011. 04. 01.
최락훈, 박정준, 임태민, 이진용, 김영록 :
서강대학교 전자공학과
※ 본 연구는 2010년도 '3단계 BK21 사업'및 한국연구재단 기초연구사업(20100016662)의 지원을 받았으며, 설계 Tool은 IDEC의 지원을 받아 연구하였음.

에 따라 연산 복잡도와 하드웨어 복잡도의 상충관계가 발생한다. 이를 보안하기 위해 복합 radix 알고리즘 연구가 활발하다 [9].

본 논문에서는 IEEE 802.11n 사용되는 64/128 FFT/IFFT 프로세서를 하드웨어로 구현하고, 하드웨어 복잡도 측면에서 비교 분석한다. FFT/IFFT 프로세서 내부 곱셈기의 효율성 증대를 위해 radix-2^3와 radix-4 알고리즘을 이용한 복합 radix 알고리즘을 새롭게 제안한다.

논문의 구성은 다음과 같다. II장에서 기존의 radix-r 알고리즘에 대해 설명한다. III장에서는 복합 radix 알고리즘을 제안한다. IV장에서 기존 알고리즘과 제안한 복합 radix 알고리즘의 하드웨어 복잡도를 비교하고, V장에서 결론을 내린다.

II. 기존의 radix-r 알고리즘

N-point DFT(Discrete Fourier Transform)은 다음과 같이 표현된다.

$$X[k]=\sum_{n=0}^{N-1} x[n]\,W_N^{kn} \quad (k=0,1,\cdots N-1) \tag{1}$$

이 때 $W_N=e^{-j2\pi/N}$ 이고, N은 입력 시퀀스의 길이이다. 필요한 연산 복잡도는 $O(N^2)$이다. FFT는 DFT 연산을 고속으로 처리하기 위한 알고리즘이며, 일반적으로 $N=2^k$ 인 경우에 더 효율적이다. radix-r FFT를 사용함으로써 연산 복잡도는 $O(N\log_r N)$로 줄어든다. 표 1에 radix 값이 r일 경우의 DFT와 FFT의 연산 복잡도를 나타내었다.

표 1. DFT와 FFT의 연산 복잡도 비교
Table 1. Comparison of computational complexity of DFT and FFT

	복소 곱셈	복소 덧셈
DFT	N^2	N^2
FFT	$N\log_r N$	$N\log_r N$

FFT/IFFT 프로세서의 기본 구성단위인 버터플라이의 구조에 따라 radix 알고리즘이 결정된다. FFT는 r-point DFT의 반복연산으로 분해될 수 있고, 이 때 r이 radix 값으로 결정된다. 가장 기본적인 알고리즘으로 radix-2가 있으며, 이는 2개의 입력과 2개의 출력을 가진다. N이 2인 경우 radix-2 알고리즘을 이용한 DFT 수식은 다음과 같다.

$$\begin{aligned} X[0] &= x[0]\,W_2^0 + x[1]\,W_2^0 \\ X[1] &= x[0]\,W_2^0 + x[1]\,W_2^1 \end{aligned} \tag{2}$$

수식 (2)를 SFG(signal flow graph)로 나타내면 다음과 같다.

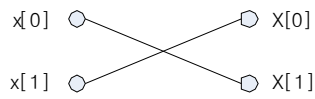

그림 1. Radix-2 SFG
Fig. 1. Radix-2 SFG

Radix-4는 한 번에 4개의 데이터를 DFT 할 수 있는 알고리즘으로 N이 4일 경우 radix-4 알고리즘의 DFT의 식을 행렬로 표현하면 다음과 같다.

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \end{bmatrix} = \begin{bmatrix} W_4^0 & W_4^0 & W_4^0 & W_4^0 \\ W_4^0 & W_4^1 & W_4^2 & W_4^3 \\ W_4^0 & W_4^2 & W_4^4 & W_4^6 \\ W_4^0 & W_4^3 & W_4^6 & W_4^9 \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \end{bmatrix} \tag{3}$$

이 때 사용되는 twiddle factor는 아래의 식으로 간단히 표현될 수 있다.

$$\begin{aligned} W_4^0 &= e^0 = 1 & W_4^4 &= e^{-j8\pi/4} = j \\ W_4^1 &= e^{-j2\pi/4} = -j & W_4^6 &= e^{-j12\pi/4} = -1 \\ W_4^2 &= e^{-j4\pi/4} = -1 & W_4^9 &= e^{-j18\pi/4} = -j \\ W_4^3 &= e^{-j6\pi/4} = 1 \end{aligned} \tag{4}$$

그림 2는 수식 (3), (4)를 이용하여 나타낸 radix-4 알고리즘의 SFG이다.

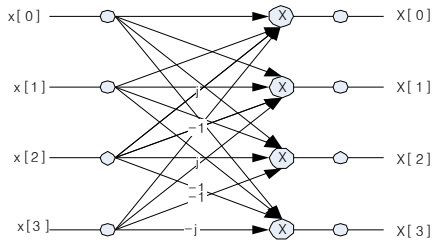

그림 2. Radix-4 SFG
Fig. 2. Radix-4 SFG

그림 1 또는 2의 SFG를 파이프라인 구조 기반의 SDF (single delay feedback) 방식을 이용하여 하드웨어를 구성할 수 있다.

Radix-2 기반 FFT/IFFT의 한 스테이지는 버터플라이와 딜레이 역할을 하는 한 개의 FIFO(first input first output)로 구성되며 구조는 다음과 같다.

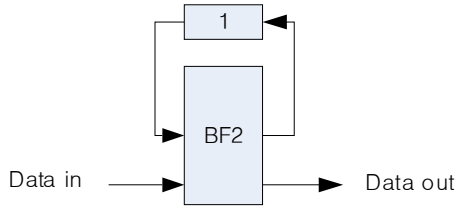

그림 3. Radix-2 SDF의 한 스테이지
Fig. 3. one stage of radix-2 SDF

Radix-4 역시 같은 방법으로 구현이 가능하며, 3개의 FIFO로 구성되어 있다.

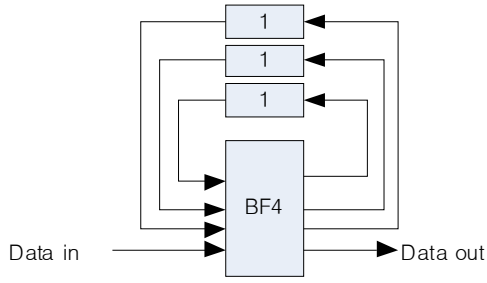

그림 4. Radix-4 SDF의 한 스테이지
Fig. 4. one stage of radix-4 SDF

그림 3와 4에서 알 수 있듯이 radix의 값 r이 증가할수록 구조가 복잡해지며, 이는 하드웨어 복잡도를 증가시키는 이유가 된다. 하지만 한 번에 처리할 수 있는 데이터의 양이 많아지게 되어 동작속도 측면에서 이득을 얻을 수 있다.

고정 radix-r 알고리즘을 이용한 FFT/IFFT 프로세서는 r의 크기에 따라 연산과 하드웨어 복잡도가 영향을 받는다. Radix-2을 이용한 FFT/IFFT 프로세서는 구조적으로 가장 간단하게 구현이 가능하다. 하지만 가장 많은 복소 곱셈 연산이 요구되고, 스테이지의 수도 가장 많아 동작 속도가 느리다. Radix-8 알고리즘은 64/128 FFT/IFFT를 구현하기 위해 2개 또는 3개의 스테이지가 요구된다. 때문에 radix-2, 4기반 알고리즘으로 구현된 FFT/IFFT 프로세서 보다 동작속도 측면에서 이득을 가진다. 하지만 8개의 입력을 한 번에 처리해야 하는 버터플라이와 이에 필요한 7개의 FIFO가 많은 하드웨어 복잡도를 요구한다 [4-6].

높은 radix 알고리즘이 가지는 하드웨어 복잡도를 극복하기 위해 radix-2^2, radix-8, radix-2^3등의 알고리즘이 제안되었다 [7]. Radix-2^2은 radix-2의 버터플라이 구조로 구현될 수 있고, radix-4 알고리즘과 같은 수의 복소 곱셈만을 필요로 한다. Radix-2^3 알고리즘은 대부분의 복소 곱셈 연산을 간소한 곱셈 연산으로 대체 할 수 있어, 가장 적은 수의 복소 곱셈 연산만을 필요로 한다. 때문에 최근 Radix-2^3 알고리즘이 많이 사용되는 추세이다.

Ⅲ. 제안된 복합 radix 알고리즘

복합 radix 알고리즘은 데이터의 수를 맞추기 위한 수단으로만 사용되었다. [8]에서는 128 FFT/IFFT를 구현하기 위해 1개의 radix-2 기반 스테이지를 2개의 radix-8 기반 스테이지 앞에 연결함으로써 구현하였다. 본 논문에서는 radix-2^3과 radix-4알고리즘의 혼합을 통해 연산과 하드웨어 복잡도를 최적화 하는 방법을 제안한다.

Twiddle factor의 경우 크게 복소 곱셈 연산과 간소한 곱셈 연산으로 구분된다. 복소 곱셈 연산은 하드웨어 구현 시 큰 복잡도를 필요로 한다. 간소한 곱셈은 복소 곱셈을 덧셈과 실수 곱셈으로 표현하는 방법으로 이를 이용하여 하드웨어 복잡도를 줄일 수 있다. 간소한 곱셈으로 표현할 수 없는 twiddle factor의 경우는 불가피하게 복소 곱셈을 사용한다.

Radix-2^3 알고리즘 기반 FFT/IFFT 프로세서의 2개의 스테이지를 1개의 radix-4 알고리즘으로 대체하는 과정에서 twiddle factor의 종류는 하드웨어 복잡도를 결정하는 중요한 척도가 된다. 제안하는 복합 radix 알고리즘은 다음과 같은 규칙을 적용하여 하드웨어 복잡도를 최적화한다.

Radix-2^3 알고리즘에서는 첫 번째 스테이지 후 twiddle factor W_4의 곱셈 연산이 요구되며, 2번째 스테이지 후에는 twiddle factor W_8의 곱셈 연산이 요구된다. 3번째 스테이지 후에는 간소한 복소 곱셈으로 대체될 수 없는 복소 곱셈 연산이 수행된다. W_4와 W_8는 실수부와 허수부의 스케일링을 통해 간

소한 복소 곱셈으로 구현될 수 있다.

$$W_4 = e^{-j2\pi/4} = 1-j \qquad (5)$$
$$W_8 = e^{-j2\pi/8} = \frac{(1-j)}{\sqrt{2}}$$

$-j$ 곱셈은 입력데이터의 실수 값의 보수와 허수 값을 바꾸는 것만으로 쉽게 구현 할 수 있다. 그림 5는 하드웨어 블록 구조를 나타낸 것이다.

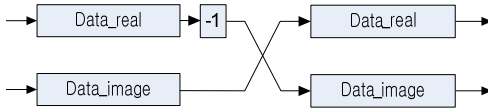

그림 5. $-j$의 스케일링 구현 블록
Fig. 5. Scaling implement block of $-j$

수식 (5)의 $1/\sqrt{2}$ 곱셈 연산은 입력 데이터를 각각 1bit, 3bit, 4bit, 6bit right shift를 수행한 후, 각각의 값을 덧셈 연산 하는 것으로 표현할 수 있다. 수식 (6)에 스케일링 과정을 2진법으로 표현하면 다음과 같다.

$$\frac{1}{\sqrt{2}} = 0.7071 = 2^{-1} + 2^{-3} + 2^{-4} + 2^{-6} \qquad (6)$$

$1/\sqrt{2}$의 스케일링 구현 블록 구조를 그림 6에 표현하였다.

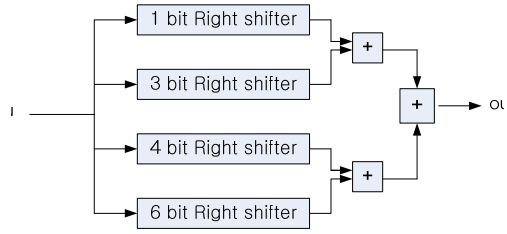

그림 6. $1/\sqrt{2}$의 스케일링 구현 블록
Fig. 6. Scaling implement block of $1/\sqrt{2}$

큰 하드웨어 복잡도가 요구되는 radix-4 버터플라이와 twiddle factor의 위치는 전체 하드웨어 복잡도의 크기에 많은 영향을 끼친다. Radix-4 알고리즘의 전·후에 복소 곱셈 연산이 있을 경우 제어 과정에 요구되는 하드웨어 복잡도가 증가하므로 간소한 곱셈이 사용되는 위치에 radix-4 알고리즘을 대체한다.

위의 규칙을 적용하여 최적화된 하드웨어 복잡도를 갖는 복합 radix 4×2×2×4 64 FFT/IFFT와 복합 radix 4×2×2×4×2 128 FFT/IFFT를 제안한다. 그림 7은 128 FFT 프로세서의 블록 다이어그램이며, 마지막 블록을 제거함으로써 64 FFT 프로세서를 구현할 수 있다.

IV. 실험 결과

기존의 고정 radix-r 알고리즘과 제안한 복합 radix 알고리즘 기반의 FFT 프로세서를 하드웨어

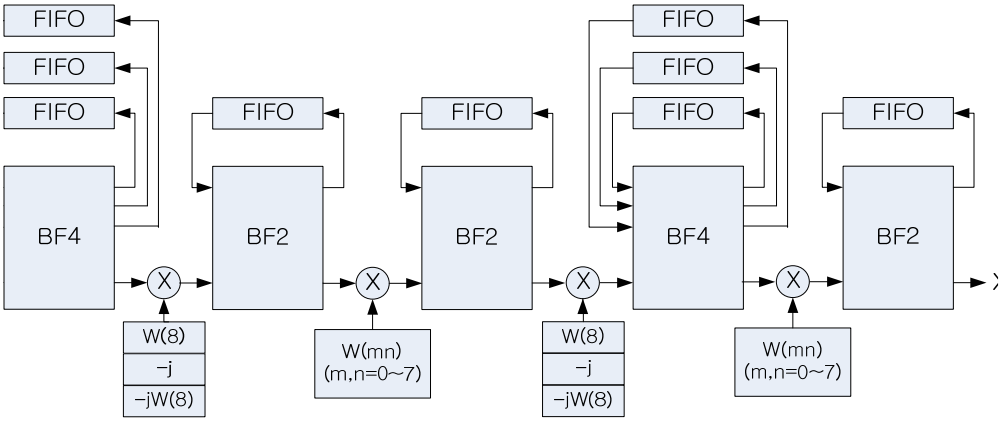

그림 7. 제안한 128 FFT 프로세서 블록 다이어그램
Fig. 7. Block diagram of the proposed 128 FFT processor

복잡도 측면에서 비교한다. Verilog HDL로 작성하였고, Xilinx ISE를 이용해 vertex4 계열의 XC4VSX35로 합성되었다.

표 2. 64/128 FFT/IFFT 프로세서 동작속도, 복잡도 비교
Table 2. Processing time and hardware complexity of the FFT/IFFT Processor

	동작속도 (MHz)	복잡도(# gates)	
		64 point	128 point
제안된 복합 radix	22.99	25,120	41,285
Radix-2^3	27.44	30,224 (120%)	44,597 (108%)
Radix-4	28.56	29,501 (117%)	

표 2에 radix 알고리즘에 따른 동작속도와 하드웨어 복잡도를 나타내었다. 기존의 알고리즘과 비교해 제안한 복합 radix 알고리즘은 하드웨어 구현 시 최대 20%의 하드웨어 복잡도를 감소시킬 수 있다. 제안한 FFT/IFFT 프로세서는 동작속도 측면에서 기존의 알고리즘에 비해 약간의 성능 저하를 보이지만, 시퀀스의 길이가 64 경우, 클락 주파수 20MHz에서 64사이클로 동작되고, 128일 경우, 클락 주파수 40MHz에서 128사이클로 동작된다. 이는 3.2μs만에 동작이 완료됨을 의미하고, IEEE 802.1n의 기준인 3.6μs과 4μs보다 적은 시간임을 알 수 있다.

V. 결 론

본 논문에서는 IEEE 802.11n 표준 OFDM 시스템의 핵심 블록인 FFT/IFFT 프로세서에 대해 소개하고, mixed radix 알고리즘을 제안한 후 하드웨어 복잡도를 비교 분석하였다. 제안한 알고리즘을 통해 더 적은 하드웨어 복잡도로 FFT/IFFT 프로세서를 구현할 수 있으며, 하드웨어 복잡도에 민감한 IEEE 802.11n WLAN 표준에 적합하게 사용될 수 있다.

참고문헌

[1] Bo Fu and Paul Ampadu, "An area efficient FFT/IFFT processor for MIMO-OFDM WLAN 802.11n.", Journal of Signal Processing Systems, Vol.56 pp. 59-68, 2009.

[2] 조용수, 무선 멀티미디어 통신을 위한 OFDM 기초, 대영사, 2001.

[3] 이홍원, 서정원, 정재학, 조상인, 최상성 "UWB-MIMO 시스템의 프리엠블 구조 설계", 대한임베디드공학회 논문집 Vol.2, No.1, pp. 66-75, 2007.

[4] M.P.Chitra and S.K.Srivatsa, "High Speed Mixed Radix FFT/IFFT Pipelined Architecture for MIMO OFDM WLAN 802.11n", European Journal of Scientific Research Vol.37 No.1, pp. 153-159, 2009.

[5] Chin-Teng Lin, Yuan_Chu Yu, and Lan-Da Van, "A low-power 64-point FFT/IFFT design for IEEE 802.11a WLAN application", Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE international Symposium on, pp. 4523-4526, 2006.

[6] Koushik Maharatna, Eckhard Grass, and Ulrich Jagdhold, "A 64-point fourier transform chip for high-speed wireless LAN application using OFDM", IEEE Journal of Solid-state Circuits. Vol.39 No.3 pp. 484-493, 2004.

[7] H. Shousheng and M. Torkelson, "Designing pipeline FFT processor for OFDM (de)Modulation", Signals, Systems, and Electronics, .IEEE URSI Intenational Symposium on, Vol.29, pp. 257-262. 1998.

[8] L. Yu-Wei and L. Chen-Yi, "Design of an FFT/IFFT processor for MIMO OFEM Systems", IEEE Transactions on circuits and systems, Vol.54, No.4, pp. 807-815, 2007.

[9] Anthony T. Jacobson, Dean N. Truong, and Bevan M. Baas, "The design of a reconfigurable continuous-flow mixed-radix FFT processor", IEEE International Symposium Circuits and Systems (ISCAS), pp 1133-1136, 2009.

저 자 소 개

최 락 훈 (Rakhun Choi)

2010년 : 서강대학교 전자공학과 학사.
현재, 서강대학교 전자공학과 석사과정.
관심분야 : 시스템 하드웨어, 무선통신 신호처리.
Email : rhchoi@sogang.ac.kr

박 정 준 (Jungjun Park)

2009년 : 서강대학교 전자공학과 학사.
현재, 서강대학교 전자공학과 석사과정.
관심분야 : 레이더 신호처리, 무선통신 신호처리.
Email : jungjun@sogang.ac.kr

임 태 민 (Taemin Lim)

2010년 : 한국항공대학교 전자공학과 학사.
현재, 서강대학교 전자공학과 석사과정.
관심분야 : 레이더 신호처리, 무선통신 신호처리.
Email : bbibbibbi@sogang.ac.kr

이 진 용 (Jinyong Lee)

1999년 : 서강대학교 물리학과 학사.
2007년 : 서강대학교 전자공학과 석사.
현재, 서강대학교 전자공학과 박사과정.
관심분야 : VLSI 설계를 위한 DSP 알고리즘, 무선통신 신호처리.
Email : jiny4509@sogang.ac.kr

김 영 록 (Younglok Kim)

1991년 : 서강대학교 전자공학과 학사.
1993년 : Polytechnic Univ. 석사.
1998년 : Polytechnic Univ. 박사.
1999~2003년 : Interdigital Comm. Corp. 연구원.
현재, 서강대학교 전자공학과 부교수.
관심분야 : VLSI 설계를 위한 DSP 알고리즘, 레이다 및 무선통신 신호처리.
Email : ylkim@sogang.ac.kr