Parallel 방식을 이용한 2K/4K/8K-point FFT processor

문상철, 박인철
한국과학기술원 전자전산학과 전기 및 전자공학 전공
대전광역시 유성구 구성동 373-1
전화 : 042-869-8061 Fax : 042-869-4040
Email : scmoon@ics.kaist.ac.kr , icpark@ee.kaist.ac.kr

요약

본 논문은 Cooley-Tukey 알고리즘을 이용하여 보다 적은 리소스로 8-point 이상의 2^n-point FFT를 Radix-4 방식의 성능으로 수행할 수 있는 parallel 구조를 제시하고, 이를 이용하여 2K/4K/8K-point FFT를 47Mhz에서 각각 59.9μs, 130.5μs, 282.8μs안에 수행할 수 있는 FFT 프로세서를 다룬다.

1. 서론

Orthogonal Frequency Division Multiplexing(OFDM)은 미래의 무선 멀티미디어 시스템에서 요구하는 높은 data rate와 잡음에 강한 전송을 만족시키는 차세대 통신 프로토콜이다[1][2]. OFDM의 VLSI 구현에 있어서 FFT 프로세서는 가장 중요한 요소 중 하나인데 특히, OFDM을 이용하는 디지털TV의 경우 DVB-T 규격에 의하여 유럽의 경우, 2K/8K-point DFT를 일본의 경우, 4K-point DFT를 요구한다[3].

FFT 프로세서의 구현에 있어 2K, 4K, 8K 같은 긴 point의 DFT를 처리하는 파이프라인 방식의 FFT 프로세서는 많은 수의 복소수 곱셈기와 복소수 가산기를 요구하여 많은 게이트 수를 갖게 된다.[4][5][6]. 이러한 단점을 한 개의 버터플라이를 메모리에 연결하여 FFT를 수행하는 방식으로 극복할 수 있으나, Rasix-2의 버터플라이 방식은 수행속도가 느리고, 수행속도가 빠른 Radix-4 방식은 4^n-point FFT에만 적용할 수 있어 2K, 8K point에 대해 적용할 수 없는 문제점을 갖고 있다.

이에 본 논문에서는 Cooley-Tukey 알고리즘[7]을 이용하여 N-point의 FFT를 N/4-point FFT와 4-point FFT로 구현함으로써, Radix-4 FFT의 성능을 2K, 8K에 적용할 수 있는 FFT 프로세서를 설계하였다.

본 논문의 2절은 FFT 프로세서에 사용된 Cooley-Tukey 알고리즘을 소개하고, 3절에서 FFT 프로세서의 구조와 메모리 어드레싱 알고리즘을 설명한다. 4절은 제안된 FFT 프로세서의 성능을 비교하고 합성결과를 제시하고 5절에서 결론을 맺는다.

2. Cooley-Tukey 알고리즘

N-point 이산(discrete) 퓨리에 변환은 다음과 같이 정의된다.

$$X[k]=\sum_{n=0}^{N-1}x(n)W_N^{nk},\quad k=0,1,...,N-1 \qquad (1)$$
$$with\quad W_N^r=e^{-j(2\pi r/N)}$$

이를 Cooley-Tukey 알고리즘을 이용하여 4×N/4의 2차원 퓨리에 변환으로 구성할 수 있다.

$$X[k]=\sum_{n_2=0}^{3}\left[\left(\sum_{n_1=0}^{N/4-1}x[4n_1+n_2]W_{N/4}^{k_1n_1}\right)W_N^{k_1n_2}\right]W_4^{k_2n_2}$$
$$n=N_2\cdot n_1+n_2,\quad \begin{cases}0\le n_1\le N_1-1\\ 0\le n_2\le N_2-1\end{cases} \qquad (2)$$
$$k=k_1+N_1\cdot k_2,\quad \begin{cases}0\le k_1\le N_1-1\\ 0\le k_2\le N_2-1\end{cases}$$

이 2차원 퓨리에 변환은 3단계로 구성된다. 첫 번째 단계에서 N/4 point의 DFT를 4번 수행하게 되고, 두 번째 단계에서 twiddle factor를 곱한다. 마지막으로 세 번째 단계에서 4-point DFT를 수행하여 N-point DFT를 수행하게 된다.

3. FFT 프로세서

이 절에서는 제안된 FFT프로세서의 구조와 어드레싱 방법에 대해서 설명한다.

3.1 FFT 프로세서의 개요

본 논문에서는 FFT 연산시 앞서 설명한 Cooley-Tukey 알고리즘을 적용하여 N-point의 큰 FFT를 이보다 작은 N/4-point FFT와 4-point FFT로 수행한다. 이때, N/4-point FFT가 4개 존재하게 되고, 4-point FFT가 N/4개 존재하게 되는데 이러한 구조는 각 N/4-point FFT간의 독립성으로 인해 병렬처리를 가능하게 함으로써 성능 향상을 꾀할 수 있게 한다.

이런 장점을 이용하여 크기가 N인 메모리와 한 개의 연산장치로 이루어진 구조의 FFT프로세서를 설계할 경우, 파이프라인 방식의 FFT 프로세서 구조와 비교할 때 보다 적은 리소스를 사용하면서 Radix-4 알고리즘을 사용한 구조와 동일한 성능으로 2K, 8K-point FFT를 수행할 수 있다.

제안된 FFT 프로세서는 동시에 접근할 수 있는 2개의 뱅크로 이루어진 4개의 2K 램(RAM)과 연산장치로 이루어져 있다. 그림 1은 제안된 FFT 프로세서의 구조를 나타낸다.

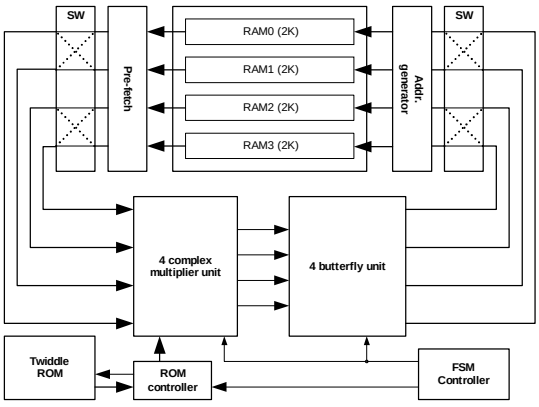

그림 1 제안된 FFT 프로세서의 구조

제안된 FFT 프로세서는 앞서 언급한 Cooley-Tukey 알고리즘을 두 단계로 수행한다

첫 번째 단계에서는 N/4-point FFT 4개가 동시에 수행되게 된다. 이때 메모리에서는 한 개의 읽기 주소로 각 RAM에 있는 8개의 뱅크로부터 동시에 8개의 데이터를 읽어 들인다.

두 번째 단계에서는 twiddle factor 곱셈 연산과 4-point FFT를 동시에 수행하게 된다. 이는 4-point FFT에서는 곱셈기가 필요하지 않기 때문에 가능하다. 이때 메모리에서는 한 개의 읽기 주소로 각 RAM에서 4개의 데이터를 읽어 들이고 twiddle factor 곱셈 연산을 수행한 후, 이 값을 버터플라이 연산장치에 입력하게 된다. 이러한 4-point FFT연산은 총 N/4번 일어난다.

이러한 방식은 표 1에서 보듯이 Radix-4방식의 버터플라이를 한 개 사용한 메모리 방식의 FFT 프로세서와 동일한 성능을 가질 수 있다. 표 1의 연산 수는 해당 프로세서의 연산장치에서 이루어지는 연산의 수이다. 그러나 Radix-4 방식의 FFT 프로세서가 4^n-point FFT에 대해서만 수행할 수 반면에 제안된 FFT 프로세서는 그림 2에서 볼 수 있는 것처럼 첫 번째 단계에서 수행하는 FFT의 point수를 조정함으로써, 8 point 이상의 2^n-point FFT를 수행할 수 있다.

표 1 N-point FFT에 대한 제안된 프로세서의 수행 성능

	Radix-4	Proposed
연산 수	$\frac{N}{4}(\log_4 N)$	$\frac{N}{8}(\log_2 N-2)+\frac{N}{4}$

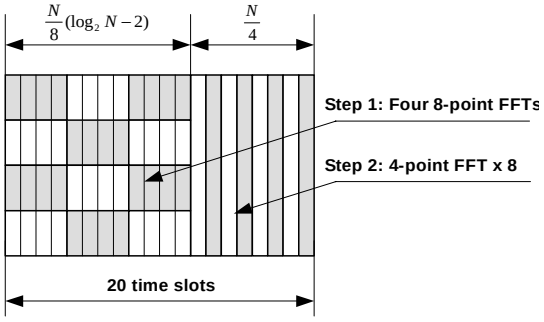

그림 2 32-point FFT에 대한 동작

제시된 알고리즘을 프로세서에서 구현하기 위해 4개의 Radix-2 버터플라이 연산장치를 사용하였다. 각 버터플라이 연산장치에는 1개의 복소수 곱셈기가 사용된다. 첫 번째 단계의 병렬 FFT연산과 두 번째 단계의 4-point FFT를 한 연산장치로 수행하기 위해서 곱셈기와 덧셈기를 분리하여 각 단계에 따라 입출력 관계를 정의하였다. 또한, 곱셈기와 덧셈기를 분리하여 파이프라인을 적용함으로써 성능을 향상 시킬 수 있다.

3.2 병렬 FFT 연산 단계

제안된 FFT 프로세서는 N-point FFT 연산의 첫 번째 단계에서 N/4-point FFT를 4개의 버터플라이 연산장치를 이용하여 동시에 수행한다. 이때 4개의 N/4-point FFT는 각각 다른 4개의 램에서 데이터를 읽어 들여 병렬처리가 가능하다. 또한, 각 램에서 읽히는 데이터는 4개의 램에서 모두 동일하므로 주소 생성회로를 간단히 할 수 있다.

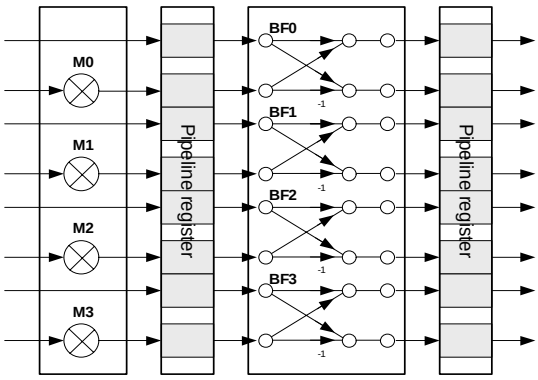

그림 3 병렬 연산 때의 연산장치

각 버터플라이 연산장치에서는 각 램의 N/4개의 데이터를 이용하여 다음과 같은 N/4-point FFT를 수행한다.

$$
\begin{aligned}
G[0,k_1] &= \sum_{n_1=0}^{N/4-1} x[4n_1]W_{N/4}^{k_1n_1} \\
G[1,k_1] &= \sum_{n_1=0}^{N/4-1} x[4n_1+1]W_{N/4}^{k_1n_1} \\
G[2,k_1] &= \sum_{n_1=0}^{N/4-1} x[4n_1+2]W_{N/4}^{k_1n_1} \\
G[3,k_1] &= \sum_{n_1=0}^{N/4-1} x[4n_1+3]W_{N/4}^{k_1n_1}
\end{aligned}
\qquad (3)
$$

3.4 4-point FFT 연산 단계

병렬 FFT 연산이 끝나고 난 뒤 Cooley-Tukey 알고리즘의 두 번째와 세 번째 단계인 twiddle factor 곱셈 연산과 4-point FFT연산이 동시에 이루어지게 된다.

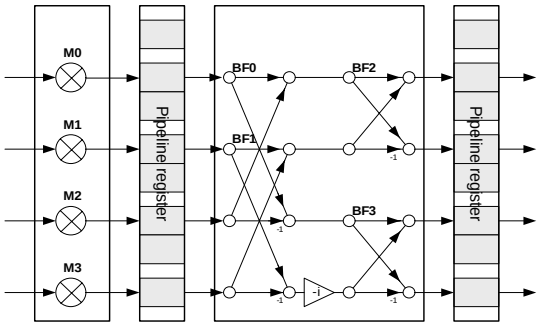

그림 4 4-point FFT 연산 때의 연산장치

연산장치의 입력은 각 램의 동일한 주소에 있는 데이터 4개가 된다. 입력된 데이터는 4-point FFT가 수행되기에 앞서 4개의 복소수 곱셈기를 갖고 있는 곱셈 연산장치에 의해 다음과 같은 twiddle factor 곱셈 연산이 이루어진다.

$$G'[n_2,k_1] = G[n_2,k_1]W_N^{k_1n_2} \quad (4)$$

곱셈기에 의해 twiddle factor 곱셈 연산이 행해진 이후 버터플라이 연산장치에서는 4-point FFT를 수행하게 된다.

$$X[4k_2+k_1] = \sum_{n_2=0}^{3} G'[n_2,k_1]W_4^{k_2n_2} \quad (5)$$

4-point FFT에서는 그림 4에서 볼 수 있듯이 곱셈장치가 필요하지 않으므로 실수와 허수를 바꾸기 위한 간단한 회로가 추가된 버터플라이 연산장치로 구현이 가능하여 병렬 연산 단계의 연산장치를 그대로 사용할 수 있다.

3.5 메모리 어드레싱

프로세서의 N-point FFT 연산의 첫 번째 단계에서 수행하는 N/4-point FFT 연산시

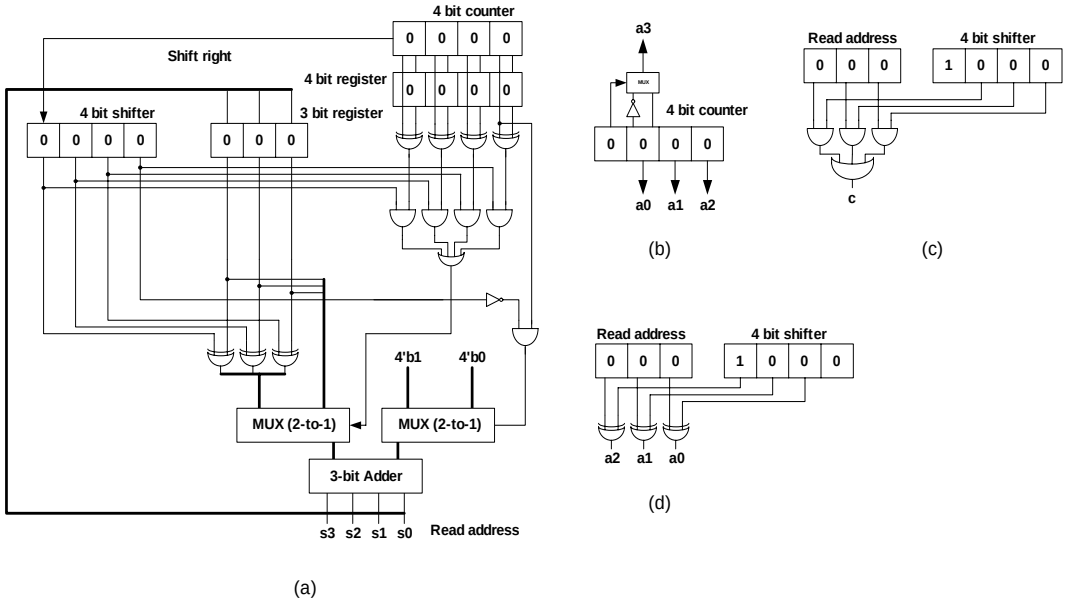

그림 5 16-point FFT에 대한 어드레싱 컨트롤러. (a) 읽기 주소 생성기. (b) 출력 주소 생성기. (c) 스위치 신호 생성기. (d) 쓰기 주소 생성기

사용되는 메모리를 최소화하기 위해 동시에 접근할 수 있는 2개의 뱅크를 갖는 메모리에 기반한 in-place 메모리 어드레싱 알고리즘을 제안하였다.

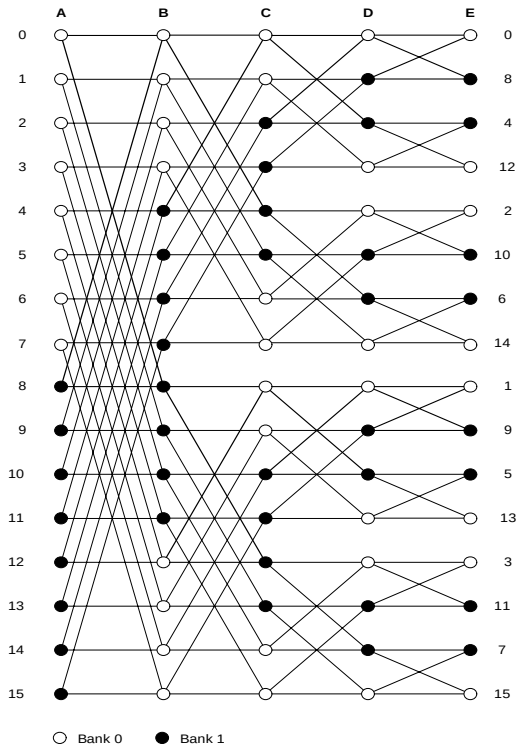

그림 6 16-point FFT에서의 어드레싱 방법

그림 6은 16-point FFT에 대해 어드레싱 알고리즘을 적용한 예이다. 하얀 원은 뱅크0를 검정색 원은 뱅크1을 의미하며 각 데이터가 저장되는 뱅크를 의미한다. 병렬 연산 단계에서 각 버터플라이 연산장치의 두 개의 입력은 같은 램에 있는 같은 주소의 서로 다른 뱅크에서 읽혀지므로 동시에 읽히는 두 개의 데이터가 늘 서로 다른 뱅크에 있게 주소를 할당하고 있다.

이러한 어드레싱 방식은 N-point FFT 연산을 위해 N 크기의 메모리만을 요구할 뿐만 아니라 point수가 증가하여도 기본 어드레싱 알고리즘을 그대로 확장하여 사용할 수 있어 2K/4K/8K-point FFT에 대응하는 주소 생성기를 쉽게 만들 수 있다. 또한, 메모리에서 데이터를 얻어낼 때 각 뱅크의 같은 주소에서 데이터를 얻어내므로 메모리의 설계를 보다 간단히 할 수 있다.

이 어드레싱 방법에서 가장 중요한 부분은 메모리에서 데이터를 얻어낼 주소를 생성하는 부분이다. 연산장치를 통과한 데이터는 다음 단계의 계산에서 읽혀지기 위해 원래 있던 주소가 아닌 다른 주소에 저장되는데 이때 데이터가 쓰여질 주소의 데이터가 미리 읽혀져야 하기 때문이다.

그림 5는 어드레싱 방법을 구현하기 위한

컨트롤러들이다. 기본적으로 카운터와 쉬프터를 이용하여 읽기 주소 생성기가 구성되고, 나머지 컨트롤러는 읽기 주소 생성기에서 얻은 주소 값과 카운터 혹은 쉬프터의 조합으로 매우 간단히 구현됨을 알 수 있다.

8K-point FFT연산은 2K-point FFT 연산의 병렬처리로 이루어지므로, 2K/4K/8K-point FFT 연산을 위해서는 2K-point FFT에 대한 컨트롤러가 필요하다. 이런 경우, 11bit 쉬프터와 11bit 카운터를 이용하여 동일한 구조의 컨트롤러를 구현하면 된다.

프로세서 연산의 첫 번째 단계인 N/4-point FFT 연산 단계에서는 읽기 주소 생성기에서 얻어낸 주소로 데이터를 4개의 각 램에서 읽어 들인 후 스위치 신호 생성기의 값에 따라 뱅크0와 뱅크1의 값을 그대로 혹은 교차하여 연산장치에 입력한다. 연산이 끝난 후엔 쓰기 주소 생성기에서 얻는 주소를 이용하여 각 4개의 램에 저장하게 된다. 램에 저장할 때 역시 스위치가 필요한데 이때에는 카운터의 LSB를 이용하여 제어하면 된다.

4-point FFT 연산 단계에서는 그림 5의 (b) 출력 주소 생성기에서 얻어낸 주소에 따라 각 램에서 한 개씩의 데이터를 읽어 들여 연산을 수행한다. 이 단계에서는 읽어 들인 주소로 쓰기를 수행하면 된다.

각 단계의 연산에서 곱셈기에 입력되는 계수값들의 생성 역시 쉬프터와 카운터에 의해 간단히 생성시킬 수 있다. 계수가 저장된 twiddle factor ROM의 읽기 주소를 연산의 첫 번째 단계에서는 쉬프터를 이용하여, 연산의 두 번째 단계에서는 카운터를 이용하여 얻어낼 수 있다.

모든 계산이 끝난 뒤 N-point FFT 결과를 최종 출력할 때에는 그림 5의 (b) 출력 주소 생성기에서 얻은 주소에 따라 한 램에 있는 데이터를 모두 출력한 뒤 다음 램에 대해 반복하고 이를 4개의 램에 대해 모두 수행하면 FFT 결과를 순서대로 출력할 수 있다.

프로세서의 각 단계에서 카운터는 0으로 리셋되어진다. 만약 프로세서의 FFT연산 point를 변경하고 싶을 때에는 쉬프터의 초기값을 변경해주면 된다.

4. 구현 및 성능 예측

제안된 FFT processor의 RTL 모델을 개발하기 전에 C언어를 사용하여 FFT 연산 알고리즘과 각 컨트롤러의 동작 알고리즘을 검증하였다.

그림 7은 한 개의 버터플라이 연산장치를 메모리와 함께 동작시킨 경우와, R2MDC 방식의 파이프라인 FFT 프로세서, Radix-4 버터플라이 연산장치를 메모리와 함께 동작시킨 경우를 제안된 프로세서의 수행 속도와 비교한 그래프이다. 제안된 프로세서는 Radix-4 방식의 FFT 프로세서와 동일한 속도를 2K/8K-point FFT에 적용하면서 파이프라인에 크게 뒤지지 않는 성능을 보이고 있다.

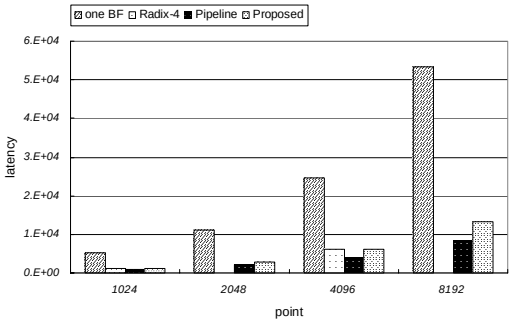

그림 7 성능 비교

제안된 프로세서의 필요 리소스 량을 표 2에서 여러 가지 파이프라인 방식의 프로세서와 비교해보았다. 제안된 프로세서는 2K/4K/8K-point FFT 수행에 있어서 파이프라인 방식의 프로세서보다 적은 량의 리소스를 요구함을 알 수 있다.

표 2 필요 리소스 비교

	multipliers	adders	mem. size
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	$\log_4 N-1$	$4\log_4 N$	$N-1$

Proposed	4	8	N

완성된 RTL 코드를 합성할 때에는 Synopsys의 Design Compiler와 TSMC 의 0.25㎛ 표준 CMOS 공정을 사용하였다. 합성결과 critical path delay는 곱셈기에서 10.62ns였고 이는 회로의 최대 동작 주파수가 47Mhz임을 의미한다. 따라서 제안된 FFT 프로세서로 2K, 4K, 8K-point FFT를 수행하는데 각각, 59.9㎲, 130.5㎲, 282.8㎲의 시간이 소요됨을 의미한다. 이러한 성능은 DVB-T 표준안에서 제한하는 2K, 8K에 대한 224㎲, 896㎲의 대기시간을 충분히 만족한다.

메모리 부분을 제외한 전체 게이트 수는 NAND 게이트 한 개의 크기로 환산한 결과 약 52,000 개였다. 이중 연산장치는 약 32,000 게이트를 차지하고, 나머지 컨트롤러가 약 20,000 게이트를 차지한다.

5. 결론

본 논문에서는 Cooley-Tukey 알고리즘을 메모리 방식의 FFT 프로세서에 적용하여 2K/4K/8K-point FFT 프로세서를 설계하였다.

파이프라인 방식의 구현에 비해 작은 수의 연산장치로 이루어져 보다 작은 게이트 수로 구현이 가능하면서, Radix-4 수행시간을 2K/8K-point FFT에 적용할 수 있도록 하여 수행 속도를 파이프라인 방식에 비해 크게 뒤떨어지지 않게 하였다.

합성결과 최대 주파수는 0.25㎛ 표준 CMOS공정에서 47Mhz였으며, 이러한 성능은 관련 어플리케이션에 적용할 때 충분히 빠른 성능이다.

참고문헌

[1] S. B. Weinstein and P. M. Ebert. “Data transmission by frequency-division multiplexing using the discrete fourier transf.” IEEE Trans. Commun., COM-19(5):628-634 Oct. 1971.

[2] L. J. Cimini. “Analysis and simulation of a digital mobile channel using orthogonal frequency division multiplexing.” IEEE Trans. Commun., COM-33(7):665-675, Jul. 1985

[3] ETSI ETS 300 744: Digital Video Broadcasting(DVB-T)

[4] Shousheng He and Torkelson, M., “Design and implementation of a 1024-point pipeline FFT processor.” Proceedings of the IEEE Custom Integrated Circuits Conference, pp. 131 -134, 1998

[5] Bidet, E.; Castelain, D.; Joanblanq, C.; Senn, P. “A fast single-chip implementation of 8192 complex point FFT,” IEEE Journal of Solid-State Circuits, , Vol. 30 Issue: 3 , pp. 300 -305, March 1995

[6] Baas, B.M. “A low-power, high-performance, 1024-point FFT processor.” IEEE Journal of Solid-State Circuits, , Vol. 34 Issue: 3 , pp/ 380-387 March 1999

[7] J.W.Cooley and J.W.Tukey. “An algorithm for machine computation of complex Fourier Series,” Math. Comput., Vol. 19, 1965