

論文2002-39SD-5-9

순차적 데이터 처리방식을 이용한 디지털 오디오 방송용 2048 Point FFT/IFFT의 VLSI 설계

(VLSI Design of a 2048 Point FFT/IFFT by Sequential Data Processing for Digital Audio Broadcasting System)

崔峻林 *

(Jun Rim Choi)

요약

본 논문에서는 순차적 입력 데이터 처리방식을 이용하여 2048 point FFT/IFFT를 단일 칩으로 구현하는 방법을 제안하고 검증하였다. 순차적으로 입력되는 2048개의 복소 데이터를 처리하기 위해서는 입력 데이터를 저장하는 버퍼가 필요하고 이 입력 버퍼로는 DRAM 회로를 이용한 지연 변환기 (delay commutator)를 사용하여 전체 칩 면적을 35% 이상 줄일 수 있었다. 전체 FFT/IFFT는 16 point FFT를 기본 블록으로 사용하여 radix-4 구조를 가지는 다섯 단계와 radix-2 구조를 가지는 하나의 단계로 이루어져 있다. 각 단계마다 연산을 수행하면서 증가되는 결과 S/N 비를 유지하면서 비트 라운딩을 하기 위해 convergent block floating point (CBFP) 알고리즘을 적용하여 digital audio broadcasting(DAB)을 위한 단일 칩 설계에 기여하였다.

Abstract

In this paper, we propose and verify an implementation method for a single-chip 2048 complex point FFT/IFFT in terms of sequential data processing. For the sequential processing of 2048 complex data, buffers to store the input data are necessary. Therefore, DRAM-like pipelined commutator architecture is used as a buffer. The proposed structure brings about the 60% chip size reduction compared with conventional approach by using this design method. The 16-point FFT is a basic building block of the entire FFT chip, and the 2048-point FFT consists of the cascaded blocks with five stages of radix-4 and one stage of radix-2. Since each stage requires rounding of the resulting bits while maintaining the proper S/N ratio, the convergent block floating point (CBFP) algorithm is used for the effective internal bit rounding and their method contributed to a single chip design of digital audio broadcasting system.

I. 서 론

기존의 FM 대역은 전파의 밀집화에 따른 다중경로

* 正會員, 慶北大學校 電子電氣工學部
(School of Electrical Engineering, Kyungpook National University)

※ 본 연구는 한국학술진흥재단 과학기술기초 반도체 연구사업 1998-016-E00015과 IDEC의 지원에 의한 결과임.

接受日字: 2001年11月6日, 수정완료일: 2002年4月8日

전파 환경과 반송파 주파수의 도플러효과 등의 영향으로 인해 음질 열화가 심화되고 있고, VHF 대역 내 방송국의 수가 늘어남에 따라 방송 채널의 추가할당도 거의 불가능한 실정이다. 이러한 FM 대역의 문제점을 해결할 뿐만 아니라 FM 대역의 낙후를 막기 위해 지상파 DAB를 이용하여 CD 수준의 고음질 오디오 서비스를 제공하기 위한 연구가 국내외적으로 활발히 진행되고 있다. 이중에서 현재 가장 활발하게 추진되고 있는 것은 그림 1의 유럽 EUREKA 147 프로젝트의 DAB 방식이다. 이 방식은 OFDM(Orthogonal Fre-

frequency Division Multiplexing) 변조 방식을 사용하고 있으며, 전송 대역폭은 1.536MHz이다.^[1,2] EUREKA-147에 따르면 246μs 동안 2048 point FFT 또는 62μs 동안 512 point FFT를 수행하여야 하므로 전용 FFT/IFFT의 ASIC 설계가 필수적이라고 할 수 있다. 이를 위하여 2048 point FFT/IFFT 칩을 순차적 데이터 처리방식으로 구현하는 방법을 제안한다. 먼저 본문 II장에서 2048 point FFT/IFFT 알고리즘에 대해 살펴보고 본문 III장에서는 칩 구현을 위한 각 기능별 하드웨어 구조에 대해 논의할 것이다. 끝으로 본문 IV장에서 제작된 칩에 대한 성능 분석을 통해 제안한 방법을 검증하였다.

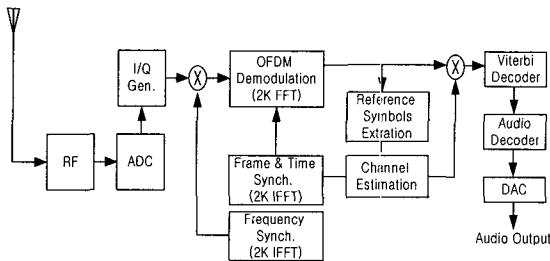


그림 1. DAB 시스템에서의 FFT/IFFT 블록다이어그램
Fig. 1. The FFT/IFFT block diagram in DAB system.

II. FFT/IFFT 알고리즘

1. 2048 point FFT 알고리즘

DFT를 계산하기 위한 효율적인 방법은 더 작은 DFT 계산으로 나누는 것이며 이는 FFT 알고리즘의 기본원리이다. 즉, N이 2048 point인 경우에는 2 point DFT를 11번 실행함으로써 계산을 할 수 있으며, 이를 radix-2 알고리즘이라고 한다. 그러나 11번의 2 point DFT로 2048 point FFT를 하드웨어로 구현할 경우 11 개의 radix-2 나비 연산 블록과 각 단계에 필요한 격자 계수를 저장하는 10개의 ROM이 필요하게 되어 단일 칩 구현뿐만 아니라 246μs 안에 FFT의 연산이 불가능하게 된다. 또한, radix-8 또는 그 이상의 radix 알고리즘을 사용하여 전체 FFT를 수행할 경우 전체 FFT의 실행시간은 단축되지만, radix-8 또는 그 이상의 radix 나비 연산을 수행하기 위한 블록의 복잡도가 급격히 늘어나게 된다. 따라서, 본 논문에서는 전체 2048 point FFT를 수행하기 위하여 radix-4 알고리즘을 기본으로

하였다. 즉, $N=4\times4\times4\times4\times2$ 와 같이 나누어 전체 연산을 수행하는 복합 radix 알고리즘을 사용하였다. 이 경우에는 radix-2 알고리즘의 장점을 유지하면서도 246μs 이내에 전체 FFT의 연산이 가능하게 된다. radix-4 알고리즘을 기본으로 한 2048 point FFT의 계수는 식 (1)에 나타나 있다.

$$n = 512 n_1 + 128 n_2 + 32 n_3 + 8 n_4 + 2 n_5 + n_6 \quad (1)$$

$$k = k_1 + 4 k_2 + 16 k_3 + 64 k_4 + 256 k_5 + 1024 k_6$$

$$\left| \begin{array}{ll} n_1 = 0, 1, 2, 3 & k_1 = 0, 1 \\ n_2 = 0, 1, 2, 3 & k_2 = 0, 1, 2, 3 \\ n_3 = 0, 1, 2, 3 & k_3 = 0, 1, 2, 3 \\ n_4 = 0, 1, 2, 3 & k_4 = 0, 1, 2, 3 \\ n_5 = 0, 1, 2, 3 & k_5 = 0, 1, 2, 3 \\ n_6 = 0, 1 & k_6 = 0, 1, 2, 3 \end{array} \right|$$

그리고, 2048 point DFT는 식 (2)와 같다.

$$X_k = \sum_{n=0}^{2047} x_n W_{2048}^{kn}, \quad 0 \leq k \leq 2047 \quad (2)$$

식 (1)의 n 과 k 를 위 식 (2)에 대입하면, 식 (3)과 같이 표시할 수 있다.

$$\begin{aligned} X_{k_1, k_2, k_3, k_4, k_5, k_6} &= \sum_{n_6=0}^1 \sum_{n_5=0}^3 \sum_{n_4=0}^3 \sum_{n_3=0}^3 \sum_{n_2=0}^3 \sum_{n_1=0}^3 \\ &x_{n_1, n_2, n_3, n_4, n_5, n_6} W_{2048}^{nk} \end{aligned} \quad (3)$$

식 (2)를 연산순으로 정리하면, 식 (4)와 같다.

$$\begin{aligned} X_{k_1, k_2, k_3, k_4, k_5, k_6} &= \sum_{n_6=0}^1 J_{k_1, k_2, k_3, k_4, k_5, n_6} \\ &W_{2048}^{n_6(1024 k_6 + 256 k_5 + 64 k_4 + 16 k_3 + 4 k_2 + k_1)} \end{aligned} \quad (4)$$

즉, 다섯 번의 radix-4 나비 연산과 한 번의 radix-2 나비 연산을 통하여 전체 2048 point FFT의 연산을 수행할 수 있다.

2. 2048 point IFFT 알고리즘

2048 point에 따른 IDFT는 식 (5)와 같다.

$$x_n = \frac{1}{2048} \sum_{k=0}^{2047-1} X_k W_{2048}^{-kn}, \quad 0 \leq n \leq 2047 \quad (5)$$

식 (2)의 DFT 수식과 비교해보면 식 (5)는 상수 $1/2048$ 와 W_N 만 서로 다른 뿐 나머지 계산과정은 동일

함을 알 수 있다. 따라서, 2048점 FFT와 동일한 계산 순서에 의해 IFFT의 값을 구할 수 있으며, 식 (5)에 계수 n 과 k 를 대입하면, 식 (6)과 같이 정리된다.

$$\begin{aligned} & X(n_1, n_2, n_3, n_4, n_5, n_6) \\ &= \sum_{k_6=0}^1 \sum_{k_5=0}^3 \sum_{k_4=0}^3 \sum_{k_3=0}^3 \sum_{k_2=0}^3 \left[\sum_{k_1=0}^3 F \right] G, \\ & F = X W^{-\frac{512}{2048} k_1 n_1}, \\ & G = W^{-\frac{(128 k_2 + 32 k_3 + 8 k_4 + 2 k_5 + k_6)n}{2048}} \quad (6) \end{aligned}$$

즉, IFFT는 2048 point FFT와 동일하게 다섯 번의 radix-4 나비 연산과 한 번의 radix-2 나비 연산을 수행한다. 단, 각각의 나비 연산에 사용되는 격자 계수의 허수부의 부호가 반대가 되어 radix-4 나비 연산이 약간 바뀌게 된다. 이에 대한 하드웨어 구현은 본문 III장에서 설명하기로 한다.

III. 블록별 하드웨어 구조

1. 전체 구조

전체적인 FFT 구조는 다섯 개의 radix-4 블록과 하나의 radix-2 블록으로 구성된다. 첫 번째 단계에서는 512 point 간격의 4개의 데이터를 입력받아 radix-4 나비 연산을 수행하고 그 값에 격자 계수를 곱하여 다음 단계로 전달한다. 두 번째 단계에서도 첫 번째 단계와 동일한 방법으로 연산을 수행하게 되는데, 첫 번째와는 달리 128 point 간격의 4개의 데이터에 대하여 radix-4 나비 연산을 수행한다. 세 번째 단계에서는 32 point 간격을 가지는 4개의 데이터를 연산하게 되며, 마지막 단계에서는 인접한 두 개의 데이터를 가지고 radix-2 나비 연산을 하게 된다. 그림 2는 2048 point FFT의 전체적인 데이터의 흐름을 나타낸 것이다.

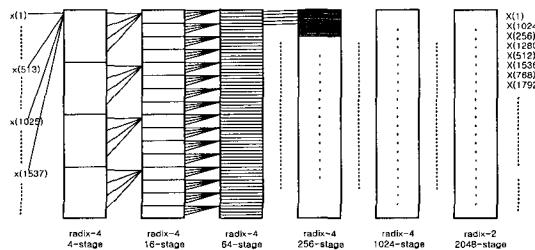


그림 2. 전체 FFT의 데이터 흐름
Fig. 2. The data flow of entire FFT.

그림 2에서 알 수 있듯이 데이터가 순차적으로 입력되는 경우에 radix-4 나비 연산을 수행하기 위해서는 첫 번째 계산에 앞서 512 point의 데이터 간격을 가지는 4개의 입력 데이터가 메모리에 저장되어 있어야 한다. 즉, 나비 연산을 하기 위해서는 최소 2048개의 복소 데이터를 저장하는 입력 버퍼를 필요로 하게 되며, 두 번째 단계에서는 512개의 복소 데이터를 저장하는 입력 버퍼가 필요로 하게 된다. 결과적으로, 전체 연산을 수행하기 위해서는 총 여섯 단의 입력 버퍼가 필요로 하게 되며, 이를 위해서 자연 변환기를 사용하였다.^[3,4] 자연 변환기는 RAM을 사용하는 것보다 면적을 줄일 수 있을 뿐만 아니라 데이터의 순차적 처리를 위해서도 바람직하다. 자연 변환기에서 정렬된 4개의 데이터는 나비 연산기에서 radix-4 또는 radix-2 나비 연산을 수행한다. ROM으로부터 FFT에 필요한 격자 계수를 읽어 나비 연산의 출력 데이터와 complex 곱셈을 수행한 후 다음 단계의 자연 변환기로 입력된다. 나비 연산기와 복소 곱셈기를 거친 데이터는 입력 비트수보다 2배 이상의 비트수를 가지게 되므로 이에 대한 대안이 필요하게 되는데, 이를 위하여 Convergent Block Floating Point(CBFP) 알고리즘을 사용하였다. CBFP는 각 단계의 전체 결과값 중에서 가장 큰 값을 기준으로 결과값의 크기를 조정하는 알고리즘으로써, 입력값의 크기에 영향을 받지 않고 정확한 FFT 연산을 가능하게 한다. 하드웨어 관점에서 본 전체 FFT의 블록 디자인은 그림 3과 같다.

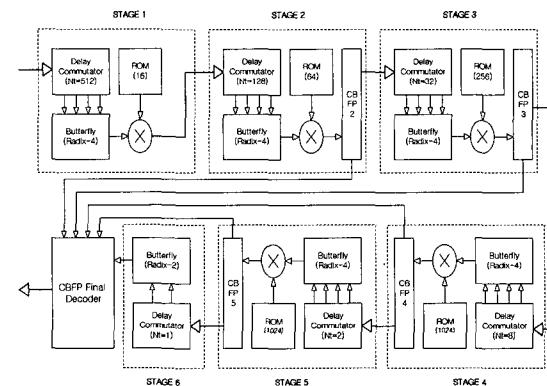


그림 3. 전체 FFT의 블록 디자인
Fig. 3. The block diagram of 2048-point FFT.

FFT전체 FFT/IFFT의 구조는 여섯 개의 단계로 구성되어 있으며, 각각의 단계는 자연 변환기 블록, 나비

연산기 블록, 복소 곱셈기와 CBFP 블록으로 구성되어 있다. 여섯 개의 단계가 직렬로 연결되어 있으므로, 전체 입력 데이터를 순차적으로 처리함을 알 수 있다.

2. 지연 변환기 (Delay Commutator)

순차적으로 입력되는 데이터의 FFT 연산을 위해서 각각의 단계마다 일정한 간격을 가지는 4개의 데이터가 radix-4 나비 연산에 필요로 하며, 이를 위한 지연 변환기 회로의 블록 다이어그램은 그림 4와 같다.

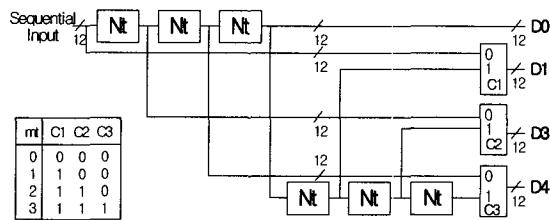


그림 4. Radix-4 연산용 지연 변환기

Fig. 4. Delay commutator for radix-4.

여기에서 Nt는 각 단계의 기본 지연 시간을 뜻하며, 첫 번째 단계에서는 512의 값을 가진다. 이 값은 하나의 단계를 지난 때마다 1/4씩 감소하게 되며, 각 단계에서 나비 연산에 필요로 하는 데이터의 간격을 나타낸다.

순차적으로 입력되는 각각의 데이터는 Nt의 시간간격을 두고 4번씩 사용되므로, 이를 위해서 단위 지연 시간(Nt)을 가지는 2개의 지연 블록이 추가되어, 지연 변환기는 총 여섯 개의 지연 블록으로 구성된다.^[5,6] 그리고, 순차적으로 입력되는 데이터를 서로 다른 경로로 나비 연산기 블록에 넘겨주기 주기 위해서 제어 신호 (C1, C2, C3)가 필요하다. 만약 지연 변환기를 메모리 커버일러로 설계한다면 지연 변환기의 게이트 수는 본문 IV장 표 3에서처럼 269,472개에 달하게 된다. 따라서, 본 논문에서는 한번의 사이클 동안에 1/M(column number)의 데이터만 이동할 수 있도록 DRAM-like 셀을 이용해 지연 변환기를 설계하였다. 결과적으로, 플립 플롭이나 메모리 커버일러를 사용하여 지연 셀을 대신 할 경우 하나의 클럭 사이클마다 전체 데이터의 이동이 불가피하므로 많은 전력소모가 발생할 뿐만 아니라 게이트 수가 증가하는데 비해서 DRAM-like 셀을 이용하여 지연 변환기를 설계할 경우 전력소모를 줄이고 게이트 수를 본문 IV장 표 3과 같이 153,984개로 줄일 수 있다. 이 DRAM-like 셀이 지연 변환기로 동작하

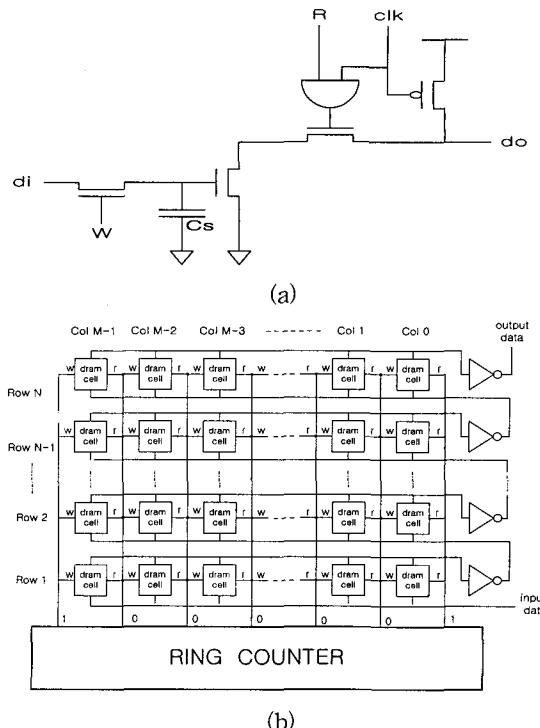


그림 5. (a) DRAM-like 셀의 구조, (b) DRAM-like 셀을 이용한 지연 변환기 구조

Fig. 5. (a) The structure of DRAM-like cell, (b) The Structure of commutator using DRAM-like cell.

기 위한 전체적인 구조는 그림 5와 같다. 전체 구조는 $M \times N$ 의 배열 형태로 구성된다. 열은 0~(M-1)까지 배타적 순환(exclusive rotation)을 하는 Johnson counter로 구성되어 있으며, 첫 번째 열의 read_word_line과 M번째 열의 write_word_line은 연결되어 있다. 전체 DRAM-like 셀 중에서 '1'의 값을 가지는 read-word-line과 접해 있는 셀에서 데이터를 읽은 다음, '1'의 값을 가지는 write-word-line과 접해 있는 셀에 데이터를 저장하게 되어 있다. 즉 열의 값이 '1'이 되는 좌우의 DRAM-like 셀에서만 데이터의 이동이 일어나며, 열이 값이 '1'이 되는 DRAM 셀의 데이터만이 열을 중심으로 데이터의 대각선 이동이 일어나게 되는 것이다.

3. 나비 연산기와 복소 곱셈기

지연 변환기에 의해 정렬된 데이터는 나비 연산기에 의해 곱셈과 뺄셈 연산을 하게 된다. Radix-4의 경우 나비 연산기에서 수행되는 연산은 식 (7)의 matrix 연산에 해당하는 부분이 된다.^[7]

$$X(p, q) = \sum_{l=0}^3 [W_N^{lq} F(l, q)] W_4^{lq} \quad p = 0, 1, 2, 3$$

$$\begin{bmatrix} X(0, q) \\ X(1, q) \\ X(2, q) \\ X(3, q) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} W_N^0 F(0, q) \\ W_N^1 F(1, q) \\ W_N^2 F(2, q) \\ W_N^3 F(3, q) \end{bmatrix} \quad (7)$$

지연 변환기를 통해 정렬된 데이터는 나비 연산기에서 연산을 수행한다. 이때 입력되는 계산 결과를 모두 하드웨어로 구현할 경우 계이트 수가 증가하게 되므로 입력 데이터가 순차적으로 들어오는 특성을 이용하여, 제어함으로써 하나의 하드웨어를 여러 번 사용할 수 있다. 이때 제어신호는 $m_t=0,1,2,3$ 에 대해서 C1, C2, C3의 신호를 가진다. 그럼 6은 나비 연산기와 곱셈기의 간단한 블록 다이어그램이다. 나비 연산기를 거친 데이터는 복소 곱셈을 수행해야 한다. 이 때 필요한 FFT 계수는 각 단계마다 같게 만들 수도 있고 다르게 만들 수도 있다. 제안된 구조는 전체 FFT가 여섯 단계로 구성되어 있으므로 다섯 개의 격자 계수가 필요하며, 각 단계마다 사용되는 격자 계수의 갯수는 다르게 되어 있다. 첫 번째 단계에서 다섯 번째 단계까지는 radix-4 연산을 기본바탕으로 하므로, stage. 첫 번째 단계에서는 16개의 격자 계수가 필요하다. 한 단계를 지날 때마다 격자 계수의 갯수는 4배씩 증가하여 두 번째 단계에서의 격자 계수의 갯수는 64가 되고, 네 번째 단계에서 필요한 격자 계수의 갯수는 1024가 된다. 단, 다섯 번째 단계는 radix-2 연산을 하므로 격자 계수의 갯수는 2배 증가하여 2048개가 필요하게 된다. 첫 번째 단계의

16개의 격자 계수는 첫 번째 단계의 연산을 마친 2048개의 데이터에 곱해지며, 하나의 격자 계수가 128번 연속적으로 반복 사용되며, 본 논문에서는 격자 계수의 연속적인 반복 사용을 통하여 칩의 면적과 전력소모의 최소화를 실현하였다.

4. Convergent Block Floating Point

여섯 단계의 FFT 연산을 하는 과정에서 일반적인 구조에서는 고정소수점 연산을 사용하기 때문에 내부 연산 비트가 증가하고 이를 해결하기 위해서 Convergent Block Floating Point(CBFP) 알고리즘을 사용하였다.^[8] 내부 비트의 수를 적절히 자를 경우 입력 비트의 크기에 의해서 출력 데이터의 S/N 비가 나빠지게 된다. 이를 보완하기 위한 방법은 각 단계를 거쳐 나온 전체 데이터를 버퍼에 저장한 다음, 저장된 데이터 중에서 가장 큰 값을 기준으로 데이터 크기를 조정하는 BFP (Block Floating Point) 방법이 있다. 그러나 이 방법은 이전 단계의 전체 결과 데이터를 모두 버퍼에 저장해야 하므로, 2048개의 복소 데이터를 저장하는 버퍼 6개가 추가적으로 필요해서 큰 면적을 차지하게 된다. 현 구조에서는 다음 단계의 입력이 그 이전 단계의 $N/4$ 개의 출력에만 관계되는 성질을 가지고 있으므로 이를 이용할 경우 작은 버퍼만으로 BFP의 변형된 구현이 가능하다. 즉, 그림 7(a)에서 128개 데이터 각각의 실수부와 허수부에서 부호 비트와 동일하면서 연속된 '0' 또는 '1'의 개수를 헤아리는데, 127번째 데이터의 허수부 값 "0000000111001100111100001"에서는 부호 비트와 동일한 '0'의 개수가 여섯 개다. 이 값은 128 데이터 중에서 가장 작은 값이므로 CBFP 계수가 되고 출

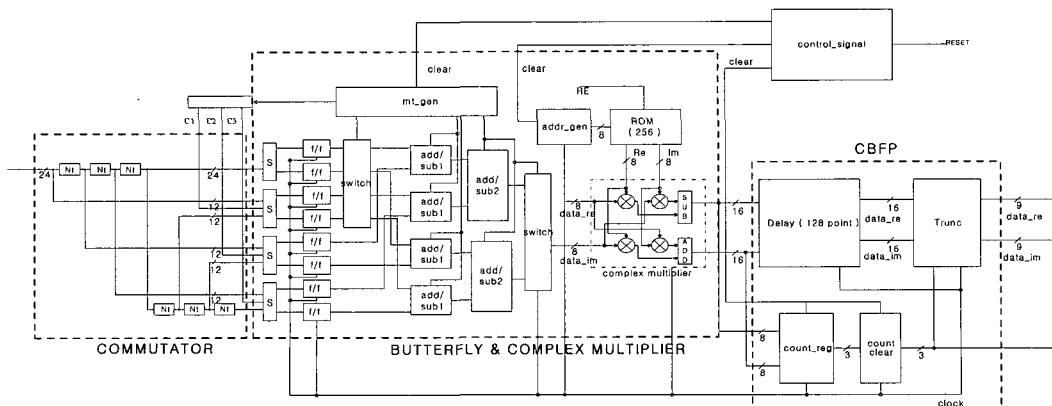


그림 6. 셋째 단의 블록 다이어그램
Fig. 6. The block diagram at the third.

력 데이터는 MSB 쪽은 여섯 비트 LSB 쪽은 일곱 비트 절단되어 다음 단계로 전달된다. 이 과정은 각 단계마다 반복적으로 수행되며, 하나의 단계를 지날수록 CBFP 계수의 수는 4배씩 증가하게 된다. 즉, 첫 번째 단계에서는 4개, 두 번째 단계에서는 128개의 데이터마다 얻을 수 있으므로 16개가 되며, 이렇게 얻어진 CBFP 계수는 그림 7(b)와 같이 데이터 연산 시간과 동일하게 지연되어, 다음 단계 CBFP 계수들과 합해져서 최종 출력단에서 라운딩된 데이터들이 보상된다.

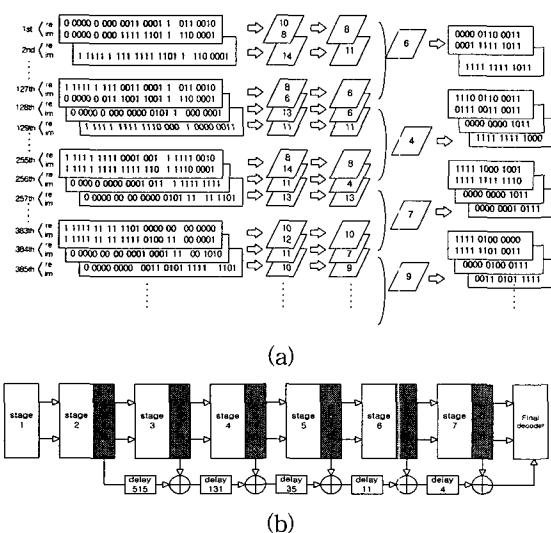


그림 7. (a) 두 번째 단계에서의 CBFP 동작 예, (b) 각 단계의 index를 보상하는 방법

Fig. 7. (a) An example of data flow in CBFP architecture at stage2, (b) The block diagram of CBFP.

5. IFFT의 구현을 위한 회로의 변형

2048개의 DFT 연산에서는 W_{2048}^{kn} 이 사용되고 IDFT에서는 W_{2048}^{-kn} 이 사용된다. 즉 IFFT 연산을 가능하게 하기 위해서는 FFT를 연산하는 하드웨어의 W_N 과 관련된 연산부의 허수부에 해당하는 블록의 출력부호만을 바꿈으로써 실현이 가능하다. 여기서 W_N 의 값은 격자 계수를 저장하는 ROM과 radix-4 나비 연산기에만 포함되어 있다. ROM으로부터 읽은 격자 계수 중에서 허수부에 해당하는 값의 부호를 바꿔주는 블록만을 추가함으로써 ROM의 데이터는 IFFT를 위한 격자 계수로 변하게 된다. 즉, IFFT radix-4 나비 연산은 식 (7)에서 식 (8)과 같은 행렬 연산으로 바뀌게 된다.

$$\begin{bmatrix} x(0, n_2) \\ x(1, n_2) \\ x(2, n_2) \\ x(3, n_2) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} \begin{bmatrix} G(0, n_2) \\ G(1, n_2) \\ G(2, n_2) \\ G(3, n_2) \end{bmatrix} \quad (8)$$

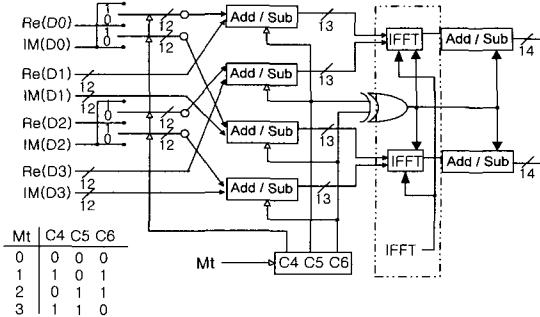


그림 8. IFFT 연산을 위한 radix-4 나비 연산부

Fig. 8. The radix-4 butterfly block for IFFT operation.

식 (8)에 따른 나비 연산부는 그림 8과 같이 되고 복소수 j 와 관련된 연산을 수행할 경우만이 FFT의 연산과 상이하므로, 이 경우에만 그림에서처럼 IFFT 제어 신호를 사용하여 나비 연산의 출력 데이터의 부호를 반전시키면 IFFT의 연산이 수행된다.

IV. 성능 분석 및 구현

1. 복소 입력 신호에 대한 모의 실험 결과

전체 FFT의 성능분석을 위하여 MATLAB을 이용해 컴퓨터로 모의 실험을 하였다. 그림 9와 그림 10은 각각 제안된 구조에 따른 FFT결과와 수식적인 계산에 의한 FFT 결과를 비교한 것이다. 그림 9 (a)는 FFT의 모의 실험을 위한 입력신호로써 QPSK 변조방식을 사용하였으며, 채널은 AWGN이다. 이 입력신호는 랜덤하게 발생된 신호를 QPSK 변조시킨 다음 IFFT를 취하여 만들었으며, 8 비트 양자화 하여 127에서 -128 사이의 값을 가진다. 그림 9 (b)는 그림 9 (a)의 입력 신호에 대한 이상적인 결과 값이고 그림 9 (c)는 제안된 구조에 의한 결과 값으로써 양자화에 의한 영향으로 성상도가 약간 흩어져 있음을 볼 수 있다. 그림 10은 실제 DAB 시스템에 수신되는 신호와 동일한 패턴의 신호를 사용하였으며, 제안된 구조에 의한 FFT 결과와 수식적 연산에 의한 FFT 결과는 같은 모양의 성상도를 나타냄을 볼 수 있다.

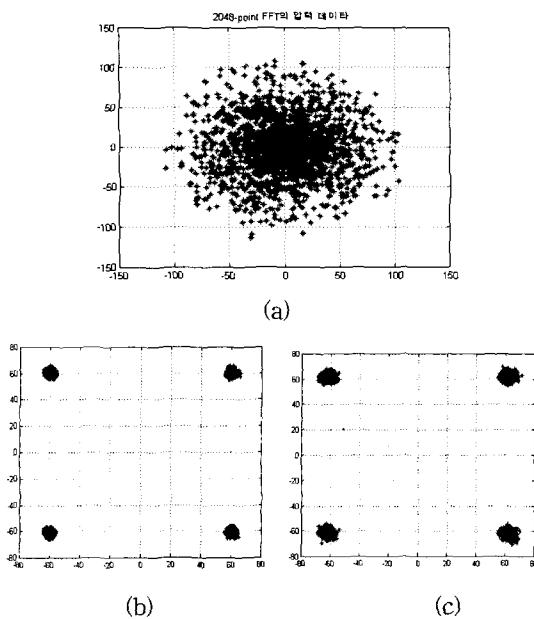


그림 9. (a) FFT 입력신호, (b) MATLAB 계산결과 (랜덤 신호), (c) 제안된 구조에 의한 계산결과 (랜덤신호)

Fig. 9. (a) Input signal for FFT simulation, (b) Calculated result by MATLAB, (c) Calculated result by proposed architecture for randomsignal.

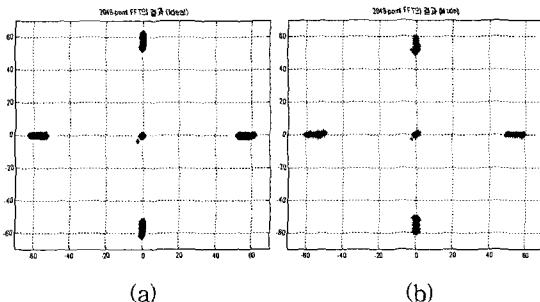


그림 10. (a) MATLAB 계산결과(실제 DAB 수신신호), (b) 제안된 구조에 의한 계산결과(실제 DAB 수신호)

Fig. 10. (a) Calculated result by MATLAB for real signal, (b) Calculated result by proposed architecture for real signal.

2. 비트 오류율 비교

제안된 구조에 의한 FFT의 비트 오류율을 구하기 위해서 MATLAB으로 모의 실험을 수행하였다. 입력 신호는 랜덤하게 발생한 신호에 백색 잡음을 추가하여 생성시켰으며, 각각의 잡음의 전력에 따른 비트 오류율은 표 1과 그림 11과 같다. 이상적인 연산 결과와 제안

된 구조에 의한 결과는 S/N 비가 8 dB 이상일 경우에는 거의 동일한 비트 오류율을 가짐을 알 수 있으며, 8 dB 이하에서만 약간의 차이가 나타나는데, 이 것은 제안된 구조가 8 비트의 출력력을 가지고 있음으로 인해서 발생하는 것이다.

표 1. 비트 오류율 비교
Table 1. The BER comparison.

구조 \ SNR	0 dB	4 dB	8 dB	12 dB
이상적인 경우	0.2949	0.1088	0.0118	6.8123×10^{-5}
제안된 구조	0.2931	0.1105	0.0121	7.8594×10^{-5}

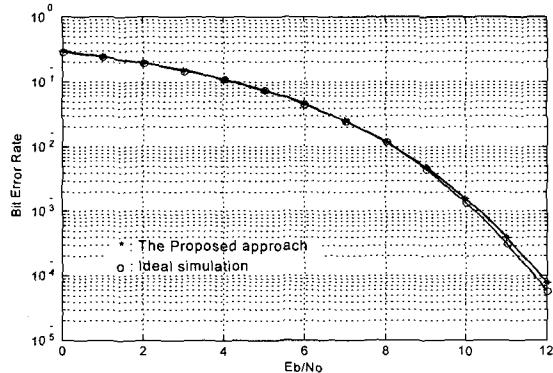


그림 11. 비트 오류율
Fig. 11. Bit error rate.

3. 게이트 수의 비교

표 2는 제안된 FFT 구조와 메모리 컴파일러를 사용한 일반적인 구조의 FFT를 비교한 것이다. 전체 게이트 수에서 35% 감소를 가져왔고 CBFP 알고리즘은 내부 비트 수를 12 비트 이하로 유지함으로써 비트 오류율을 효율적으로 유지할 수 있었다.^[8] 전체 침의 게이트 수를 좌우하는 블록은 지역 변환기와 CBFP 회로의

표 2. 제안된 구조와의 비교

Table 2. Architectural comparison.

	일반적인 구조	제안된 구조
전체 게이트 수	약, 400,000	260,890
데이터 버퍼	메모리 컴파일러	DRAM 회로를 이용한 지역 변환기
내부 비트수	16 비트 이상	12 비트 이하

버퍼이며 전체 면적의 80%를 차지하고 있으므로, 이 블록의 최적화가 필수적이다. 즉, 표 3에서처럼 이 블록을 DRAM-like 셀을 이용하여 설계함으로써 메모리 컴파일러를 사용한 경우에 비해서 42%의 게이트 감소를 가져왔다.

표 3. 지연 변환기와 CBFP 회로의 게이트 수 비교

Table 3. Gate counts comparison of delay commutator and CBFP.

단계 소자	예모리 컴파일러를 사용 할 경우		DRAM 셀 회로를 사용할 경우	
	지연 변환기	CBFP	지연 변환기	CBFP
단계 1	172,032	-	98,304	
단계 2	53,760	12,544	30,720	7,168
단계 3	18,816	3,584	10,752	2,048
단계 4	5,376	1,008	3,072	576
단계 5	1,512	280	854	160
단계 6	560		320	
합 계	269,472		153,984	

4. 전체 칩의 구현

전체 2048 point FFT는 VHDL을 이용하여 RTL(Register Transfer Level)레벨의 소스코드로 합성하였다. 합성 툴은 Synopsys를 사용하였으며 0.35 μ m의 Hynix standard cell 라이브러리를 이용하여 VHDL로 코딩하였다. 그림 12는 최종적으로 설계된 2048 point FFT/IFFT가 포함된 단일 칩 DAB 리시버의 사진이고 표 4는 설계된 FFT/IFFT에 대한 전체적인 특징을 보여주고 있다. 제어 신호에 의해 256, 512, 1024, 2048 point를 각각 연산 할 수 있으며 최대 동작 주파수는

표 4. 2048 point FFT/IFFT 칩의 특징
Table 4. Feature of 2048 point FFT/IFFT chip.

Technology	0.35 μ m CMOS
Processing samples	256, 512, 1024, 2048 points
Single supply operation	2.7 V~3.6 V
Operating frequency	40 MHz Max
Serial block data input	2 x 8 bit
Serial block data output	2 x 8 bit

40MHz이다. 입출력은 I성분과 Q성분이 복합된 16비트 신호로 이루어져 있다. 합성된 전체 회로에 대해서는 Verilog-HDL 포맷으로 netlist를 추출하여 Hynix 공정에서 제공하는 VELA2를 이용하여 디자인 룰 체크와 지연 정보를 계산하였다. Cadence사의 Verilog-XL을 사용하여 최종적인 pre-layout 시뮬레이션을 수행하였다.

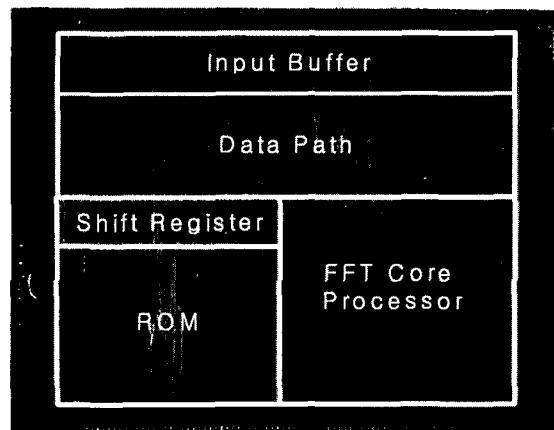


그림 12. DAB 리시버의 단일 칩 사진

Fig. 12. A Microphotograph of a single chip DAB receiver.

V. 결 론

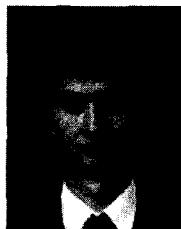
본 논문에서는 순차적 입력 데이터 처리방식을 이용하여, 2048 point FFT/IFFT를 단일 칩으로 구현하는 방법을 제안하였다. 전체 FFT는 radix-4 연산을 수행하는 다섯 단계와 radix-2 연산을 수행하는 하나의 단계로 구성되어 있다. 각 단계에서 radix-4 또는 radix-2 연산을 하기 위해서는 순차적으로 입력되는 2048개의 신호를 처리하기 위한 버퍼가 필요하다. 본 논문에서는 지연 변환기를 이용하여 순차적 입력 데이터를 지연시킴으로써 입력 버퍼를 대신하였고 전체 칩의 전력소모와 면적을 줄일 수 있었다. 특히, 지연 변환기를 DRAM-like 셀을 기본으로 구성할 경우, 메모리 컴파일러로 설계하였을 경우에 비해 35%의 전체 칩 사이즈의 감소를 가져왔다. 또한, 여섯 개의 단계에서 연산을 수행하는 동안 늘어나는 비트를 BER에 영향을 주지 않으면서 줄이기 위하여 CBFP 알고리즘이 사용되었다. 또한, 본 논문에서 제안된 FFT 칩은 제어신호(control signal)에 의해서 ROM의 격자 계수와 radix-4

연산블록을 변화시켜 IFFT 연산을 수행할 수 있게 설계되었다. 이는 DAB 시스템의 수신 블록에서 심벌 타이밍 복구를 위하여 IFFT를 수행하기 위한 추가적인 블록을 줄임으로써, 단일 칩 DAB 시스템에 기여하였다.

참 고 문 헌

- [1] Kenichi Taura, Masahiro T., Masayuki T., Hiroaki K., Masayuki I., and Yoshinobu I., "A digital audio broadcasting receiver," IEEE transactions on Consumer Electronics, Vol. 42, No. 3, pp. 323~327, August 1996.
- [2] Louis Thibault and Minh Thien Le, "Performance evaluation of COFDM for digital audio broadcasting," IEEE Transactions on broadcasting, Vol. 43, No. 1, pp. 64~75, March 1997.
- [3] Kevin J. McGee, "64-point Fourier transform chip for video motion compensation using phase correlation," IEEE J. Solid-state Circuits, Vol. 31, pp. 1751~1761, Nov. 1996.
- [4] Bevan M. Baas, "A low-power, high performance 1024-point FFT processor," IEEE Journal of Solid-state Circuits, Vol. 34, No. 3, pp. 380~387, Mar. 1999.
- [5] F. rothan, C. joanblanq, and P. senn, "A video delay line compiler," Proc. ISCAS. New Orleans, LA, pp. 65~68, May 1990.
- [6] John E. Whechel, John P. O'Malley, William J. Rinard, and James F. McArthur, "The systolic phase rotation FFT - a new algorithm and parallel processor architecture," IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 2, pp. 1021~1024, April 1990.
- [7] John G. Proakis and Dimitris G. Manolakis, Digital signal processing, Prentice Hall. 1996.
- [8] Teress M. Pytosh and Alberto M. Magnani, "A new parallel 2-D FFT architecture," IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 2, pp. 905~908. April 1990.

저 자 소 개



崔峻林(正會員)

1986년 연세대학교 전기공학과 학사.

1988년 미국 Cornell 대학교 전자전

기공학과 석사. 1991년 미국

Minnesota 대학교 전자전기 공학과

박사. 1991년 7월~1997년 2월 LG

전자기술원 책임연구원. 1997년 3

월~현재 경북대학교 전자전기공학부 조교수