



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2008-0040978
(43) 공개일자 2008년05월09일

(51) Int. Cl.

G06F 17/14 (2006.01) G06F 17/10 (2006.01)

G06F 7/00 (2006.01)

(21) 출원번호 10-2006-0109020

(22) 출원일자 2006년11월06일

심사청구일자 2006년11월06일

(71) 출원인

인하대학교 산학협력단

인천 남구 용현동 253 인하대학교

(72) 발명자

이한호

인천 남구 용현동 253번지 인하대학교, 정보통신공학부

(74) 대리인

이원희

전체 청구항 수 : 총 6 항

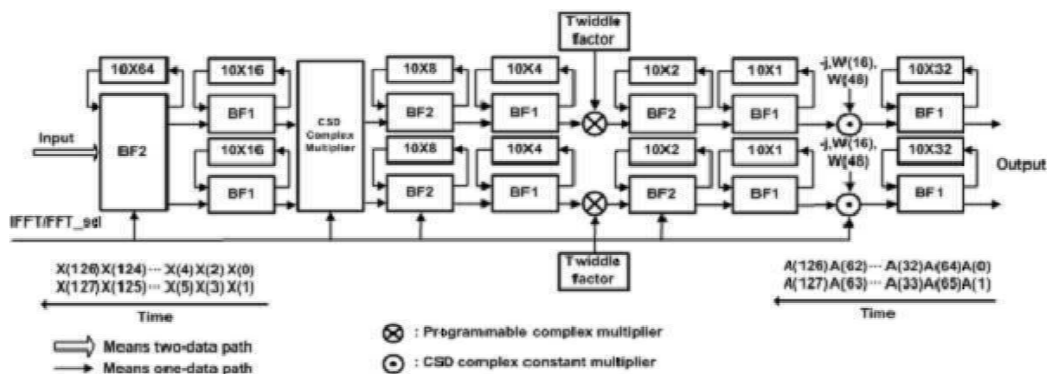
(54) 병렬 구조 및 파이프라인 방식을 이용한 Radix 2의4승 고속 푸리에 변환 프로세서

(57) 요약

본 발명은 병렬 구조 및 파이프라인 방식을 이용한 Radix 2의 4승 고속 푸리에 변환 프로세서에 관한 것으로서, 2개의 병렬 데이터 경로의 구조 및 단일 경로 딜레이 피드백(SDF: Single-path Delay Feedback) 구조를 이용함으로써, 동작 속도 및 데이터 처리율을 증가시킴과 동시에 구성을 용이하도록 하드웨어의 복잡도를 감소시켰다.

이를 위하여 본 발명은 Radix - 2^4 알고리즘을 고속 푸리에 변환에 적용하였고, 프로그래머블 복소 곱셈기의 절반에 해당하는 곱셈량을 CSD 방식을 이용한 특수 복소 상수 곱셈기로 설계하여, 하드웨어 비용 및 전력 소모를 감소시킬 수 있는 병렬 구조 및 파이프라인 방식을 이용한 Radix 2의 4승 고속 푸리에 변환 프로세서를 구현하였다. 그 기술적 구성은 고속 푸리에 변환 프로세서에 있어서, 입력신호가 인가되면 버터플라이 연산을 행하는 복수개의 Radix - 2^4 제1 버터플라이 연산부; 및 상기 Radix - 2^4 제1 버터플라이 연산부의 다음 스테이지에 연결되, 상기 제1 Radix- 2^4 버터플라이 연산부로부터 출력된 출력 데이터에 버터플라이 연산을 행하는 복수개의 Radix - 2^4 제2 버터플라이 연산부를 포함하여 이루어져, Radix - 2^4 FFT 알고리즘이 적용되는 것을 특징으로 한다.

대표도 - 도2



특허청구의 범위

청구항 1

고속 푸리에 변환 프로세서에 있어서,

입력신호가 인가되면 버터플라이 연산을 행하는 복수개의 Radix - 2^4 제1 버터플라이 연산부; 및

상기 Radix - 2^4 제1 버터플라이 연산부의 다음 스테이지에 연결하되, 상기 제1 Radix- 2^4 버터플라이 연산부로부터 출력된 출력 데이터에 버터플라이 연산을 행하는 복수개의 Radix - 2^4 제2 버터플라이 연산부를 포함하여 이루어져, Radix - 2^4 FFT 알고리즘이 적용되는 것을 특징으로 하는 병렬 구조 및 파이프라인 방식을 이용한 Radix 2의 4승 고속 푸리에 변환 프로세서.

청구항 2

제1항에 있어서,

상기 고속 푸리에 변환 프로세서는 상기 알고리즘을 적용한 2개의 병렬 데이터경로를 가지는 파이프라인 FFT프로세서인 것을 특징으로 하는 병렬 구조 및 파이프라인 방식을 이용한 Radix 2의 4승 고속 푸리에 변환 프로세서.

청구항 3

제1항에 있어서,

전체 곱셈기의 절반을 특수 복소 상수 곱셈기로 설계하는 것을 특징으로 하는 병렬 구조 및 파이프라인 방식을 이용한 Radix 2의 4승 고속 푸리에 변환 프로세서.

청구항 4

제3항에 있어서,

상기 특수 복소 상수 곱셈기는 CSD 복소 상수 곱셈기인 것을 특징으로 하는 병렬 구조 및 파이프라인 방식을 이용한 Radix 2의 4승 고속 푸리에 변환 프로세서.

청구항 5

제1항에 있어서,

상기 고속 푸리에 변환 프로세서는 상기 알고리즘을 적용한 SDF FFT 프로세서인 것을 특징으로 하는 병렬 구조 및 파이프라인 방식을 이용한 Radix 2의 4승 고속 푸리에 변환 프로세서.

청구항 6

제1항에 있어서,

상기 알고리즘은 인덱스 분해법에 의하여 유도된 것을 특징으로 하는 병렬 구조 및 파이프라인 방식을 이용한 Radix 2의 4승 고속 푸리에 변환 프로세서.

명세서

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

<9>

본 발명은 병렬 구조 및 파이프라인 방식을 이용한 Radix 2의 4승 고속 푸리에 변환 프로세서에 관한 것으로, 더욱 상세하게는 2개의 병렬 데이터 경로의 구조 및 단일 경로 딜레이 피드백 구조를 이용함으로써, 동작 속도 및 데이터 처리율을 증가시키고 동시에 구성을 용이하도록 하드웨어의 복잡도를 감소시켰다. 이를 위하여 본 발

명은 $\text{Radix} - 2^4$ 알고리즘을 고속 푸리에 변환에 적용시킴으로써, 프로그래머블 복소 곱셈기의 절반에 해당하는 곱셈량을 CSD 방식을 이용한 특수 복소 상수 곱셈기로 설계하여 하드웨어 비용 및 전력 소모를 감소시킬 수 있는 병렬 구조 및 파이프라인 방식을 이용한 $\text{Radix} 2$ 의 4승 고속 푸리에 변환 프로세서를 구현하였다.

<10> 일반적으로, 직교 주파수 분할 변조(Orthogonal Frequency Division Multiplexing: 이하, 'OFDM') 방식과 이산 다중 톤(Discrete Multi-Tone: 이하 'DMT') 방식은 다중 경로 채널을 통한 고속 데이터 전송에 유리하다.

<11> 이 방식들은 변조된 데이터를 부반송파의 수만큼 직/병렬 변환하고 각각에 대응하는 부반송파로 변조하는 방식이고, 이는 이산 푸리에 변환(Discrete Fourier Transform: 이하 'DFT')을 이용하여 구현하는데, 하드웨어 설계시에는 연산량을 줄이기 위하여 고속 푸리에 변환(Fast Fourier Transform: 이하 'FFT') 알고리즘을 이용한다.

<12> 여기서, FFT 알고리즘을 처리하기 위한 프로세서는 OFDM 시스템에 가장 큰 복잡도를 가지며, 고속 연산을 요구한다. 이러한, 고성능을 요구하는 모뎀응용분야의 FFT 설계는 파이프라인(Pipeline) 구조를 갖는 프로세서를 주로 사용한다. 파이프라인 구조는 각 스테이지(Stage) 마다 하나의 연산부를 두는 방식으로 하드웨어의 비용을 절감할 뿐 아니라 낮은 동작 주파수에서도 고속 연산의 효과를 얻을 수 있으며, 구조가 비교적 간단하여 설계가 용이한 장점을 가지고 있다.

<13> 그러나, 요구하는 FFT 포인트 수가 증가할수록 하드웨어의 비용은 로그 함수적으로 증가하며, 특히 복소 곱셈기는 전체 전력의 50%~80%의 전력을 소모하고, 하드웨어의 복잡성을 증가시키며, 데이터 처리율을 저하시키는 문제점이 있었다.

발명이 이루고자 하는 기술적 과제

<14> 본 발명은 상기한 문제점을 해결하기 위하여 안출한 것으로, 2개의 병렬 데이터 경로의 구조, 단일 경로 딜레이 피드백 구조 및 파이프라인 방식을 이용함으로써, 동작 속도 및 데이터 처리율을 증가시킴과 동시에 구성을 용이하도록 하드웨어의 복잡도를 감소시키도록 구현하였다. 그리고 $\text{Radix} - 2^4$ 알고리즘을 고속 푸리에 변환에 적용시킴으로써, 프로그래머블 복소 곱셈기의 절반에 해당하는 곱셈량을 CSD 방식을 이용한 특수 복소 상수 곱셈기로 설계하여 하드웨어 비용 및 전력 소모를 감소시킬 수 있는 병렬 구조 및 파이프라인 방식을 이용한 $\text{Radix} 2$ 의 4승 고속 푸리에 변환 프로세서를 제공하는 것을 목적으로 한다.

발명의 구성 및 작용

<15> 상기한 바와 같은 목적을 달성하기 위하여 본 발명은 고속 푸리에 변환 프로세서에 있어서, 입력신호가 인가되면 버터플라이 연산을 행하는 복수개의 $\text{Radix} - 2^4$ 제1 버터플라이 연산부; 및 상기 $\text{Radix} - 2^4$ 제1 버터플라이 연산부의 다음 스테이지에 연결하되, 상기 제1 $\text{Radix} - 2^4$ 버터플라이 연산부로부터 출력된 출력 데이터에 버터플라이 연산을 행하는 복수개의 $\text{Radix} - 2^4$ 제2 버터플라이 연산부를 포함하여 이루어져, $\text{Radix} - 2^4$ FFT 알고리즘이 적용되는 것을 특징으로 한다.

<16> 여기서, 전체 곱셈기의 절반을 특수 복소 상수 곱셈기로 설계한다.

<17> 그리고, 상기 특수 복소 상수 곱셈기는 CSD 복소 상수 곱셈기인 것을 특징으로 한다.

<18> 또한, 상기 고속 푸리에 변환 프로세서는 상기 알고리즘을 적용한 2개의 병렬 데이터경로를 가지는 파이프라인 FFT 프로세서인 것을 특징으로 한다.

<19> 더불어, 상기 고속 푸리에 변환 프로세서는 상기 알고리즘을 적용한 SDF FFT 프로세서인 것을 특징으로 한다.

<20> 한편, 상기 알고리즘은 인덱스 분해법에 의하여 유도된 것을 특징으로 한다.

<21> 이하, 본 발명에 따른 실시예를 첨부된 예시도면을 참고로 하여 상세하게 설명한다.

<22> 도 1은 본 발명에 따른 128 포인트 $\text{Radix}-2^4$ SDF FFT 알고리즘의 개략적인 신호 흐름도이다. 도면에서 도시하고 있는 바와 같이, $\text{Radix} - 2^4$ 의 전개과정을 간략하게 설명한다.

<23> 길이가 N인 DFT(Discrete Fourier Transform)은 수학적 식 1과 같이 정의한다.

수학식 1

$$X(k) = \sum_{n=0}^{N-1} x(n)W_n^k, \quad k=0,1,\dots,N-1$$

<24>

여기에서, W_n^k 는 회전 인자(Twiddle factor), k 는 주파수 인덱스, n 은 시간 인덱스를 나타낸다. 본 발명에 따른 Radix-2⁴ 알고리즘을 유도하기 위하여, 5차원 선형 인덱스 맵을 적용하면 수학식 2와 같다.

수학식 2

$$n = \left\langle \frac{N}{2}n_1 + \frac{N}{4}n_2 + \frac{N}{8}n_3 + \frac{N}{16}n_4 + n_5 \right\rangle_N$$

$$k = \langle k_1 + 2k_2 + 4k_3 + 8k_4 + 16k_5 \rangle_N$$

<26>

이때, 공통 인수 분해 알고리즘(CFA: Common Factor Algorithm)를 적용하면 수학식 3과 같다.

<27>

수학식 3

$$\begin{aligned} X(k_1 + 2k_2 + 4k_3 + 8k_4 + 16k_5) \\ = \sum_{n_5=0}^{\frac{N}{16}-1} \sum_{n_4=0}^1 \sum_{n_3=0}^1 \sum_{n_2=0}^1 \sum_{n_1=0}^1 x\left(\frac{N}{2}n_1 + \frac{N}{4}n_2 + \frac{N}{8}n_3 + \frac{N}{16}n_4 + n_5\right) W_N^{nk} \\ = \sum_{n_5=0}^{\frac{N}{16}-1} [G_{\frac{N}{16}}(n_5, k_1, k_2, k_3, k_4) W_N^{n_5(k_1 + 2k_2 + 4k_3 + 8k_4)}] W_{\frac{N}{16}}^{n_5 k_5} \end{aligned}$$

<28>

*수학식 3의 회전 인자를 계산하면, 수학식 4와 같다.

<29>

수학식 4

$$\begin{aligned} W_N^{nk} = & (-1)^{n_1 k_1} (-j)^{n_2 (k_1 + 2k_2)} W_{16}^{(2n_3 + n_4)(k_1 + 2k_2 + 4k_3)} \\ & \bullet (-1)^{n_4 k_4} W_N^{n_5 (k_1 + 2k_2 + 4k_3 + 8k_4)} W_{\frac{N}{16}}^{n_5 k_5} \end{aligned}$$

<30>

그리고, 네번째 버터플라이 연산부의 구조는 수학식 5와 같으며, 4개의 복소수를 가지는 회전인자를 나타낸다.

<31>

<32>

수학식 5

$$\begin{aligned} G_{\frac{N}{16}}(n_5, k_1, k_2, k_3, k_4) = & H(n_5) + (-1)^{k_4} W_{16}^{(k_1 + 2k_2 + 4k_3)} H(n_5 + \frac{N}{16}) \\ & + W_{16}^{2(k_1 + 2k_2 + 4k_3)} H(n_5 + \frac{N}{8}) + (-1)^{k_4} W_{16}^{3(k_1 + 2k_2 + 4k_3)} H(n_5 + \frac{3N}{16}) \end{aligned}$$

<33>

<34>

또한, 두번째 버터플라이 연산부의 구조는 H(n)으로 표현되며, 수학식 6과 같다.

<35>

수학식 6

$$H(n) = H(n, k_1, k_2) = B(n, k_1) + (-j)^{(k_1 + 2k_2)} B(n + \frac{N}{4}, k_1)$$

<36>

<37>

여기서, H(n) = H(n, k1, k2)이다. 수학식 5와 같이 네 번째 버터플라이 연산부는 두 번째 버터플라이 연산부

<38>

H(n)가 회전인자 Wn16 로 서로 연결된 형태를 가진다.

<39> 그리고, 첫번째 버터플라이 연산부는 수학적 식 7과 같다.

<40>

수학적 식 7

<41>
$$B(n, k_1) = x(n) + (-1)^{k_1} x(n + \frac{N}{2})$$

<42> 마지막으로, 본 발명에 따른 알고리즘은 프로그래머블 복소 곱셈기를 대신하여 복소 상수 곱셈기를 이용하고, CSD 상수 곱셈기는 1의 개수를 최소화하도록 연산과정을 간소화하여 하드웨어의 면적 및 전력 소비를 감소시킨다.

<43> 도 2은 본 발명에 따른 Radix-2⁴ 고속 푸리에 변환 구조를 개략적으로 나타내는 블록도이다. 도면에서 도시하고 있는 바와 같이, 2개의 병렬 데이터 경로를 가지는 128 포인트의 Radix-2⁴ SDF FFT 프로세서는 버터플라이 연산부와 복소Booth 곱셈기, CSD 복소상수 곱셈기, 레지스터와 메모리 블록으로 이루어진다.

<44> 여기서, CSD 복소 상수 곱셈기를 이용하여 복소 Booth 곱셈기를 감소시키며, 이에 따라 하드웨어 복잡도를 감소시킬 수 있다.

<45> 본 구조는 2개의 병렬 데이터 구조이므로 일반적인 데이터 흐름과는 다르게 2개의 복소수 값이 입력되므로 순차적으로 2개의 복소수를 저장 후 128 포인트의 반인 64번째의 데이터를 저장한 후 65번째의 데이터가 입력되면 이전에 저장되었던 첫 번째(홀수 번째)값을 새로운 입력 값과 연산하게 되고 함께 입력된 짝수 번째 값을 이전 홀수 번째 메모리에 저장되게 된다. 이러한 연산으로 홀수 값들을 모두 연산한 후에는 짝수 쌍에 대한 연산만을 수행하여 빈 메모리에는 입력되는 2개의 복소수들을 순차적으로 저장한다.

<46> 첫 번째 버터플라이 연산의 결과 값으로 홀수 번째 입력에 대한 연산이 먼저 수행되므로 버터플라이 연산에 필요한 N 포인트의 N/2의 램 면적은 첫 번째와 마지막 단을 제외하고 줄어들게 되지만 마지막 단의 버터플라이 연산은 홀수 번째 입력에 대한 연산을 끝낸 후 짝수 번째 연산이 시작될 동안의 값을 모두 저장하므로 일반적인 구조에 비해 램 면적인 늘어나게 된다. 하지만 2개의 병렬 데이터 구조에 의한 추가적인 램은 필요하지 않게 된다.

<47> 도 3은 본 발명에 따른 Radix-2⁴ 버터플라이 연산부의 개략적인 구성도이다. 도면에서 도시하는 바와 같이, 두 가지 연산 과정을 행하게 된다.

<48> 첫 번째 연산으로는 입력되는 실수와 허수를 램에 저장하고 이전에 램에 저장된 값은 출력 단으로 그대로 출력한다.

<49> 두 번째 연산으로는 버터플라이 연산부(BF1)에서 램에 저장된 실수와 허수와 새로운 입력된 실수와 허수는 각각 복소 감산기와 복소 가산기로 입력하고, 복소 감산기로부터 연산을 하고, 출력되는 데이터는 다시 램으로 입력되며, 복소 가산기에서 연산된 출력되는 데이터는 다음 연산을 위해 출력된다. 이러한 두 가지 연산 과정을 순차적으로 반복하게 된다.

<50> 한편, 버터플라이 연산부(BF2)는 (-j)의 연산을 수행하기 위하여 약간의 조합 회로를 포함한다.

<51> 여기서, 본 발명에 따른 2개의 병렬 데이터경로를 가지는 Radix-2⁴ FFT 를 구동하기 위한 알고리즘을 구현하기 위하여, 2개의 복소 Booth 곱셈기가 구비된다.

<52> 그리고, 여섯번째 스테이지의 버터플라이 연산부(BF1)로부터 생성된 결과데이터는 회전인자, -j, W(16), W(48)와 곱해지는데, ROM이 32 바이트의 상기 회전인자의 데이터를 저장한다.

<53> 또한, 복소 곱셈기의 절반을 CSD 복소 상수 곱셈기로 설계한다.

<54> 표 1은 회전인자를 십진수, 8-bit의 2진수, CSD 형식으로 나타낸다.

표 1

Coefficients	Decimal	2's Comp.	CSD
$\sin(\frac{\pi}{8})$	0.3827	00110000	0101̄0000
$\cos(\frac{\pi}{8})$	0.9239	01110110	10001̄01̄0
$\sin(\frac{\pi}{4}) = \cos(\frac{\pi}{4})$	0.7071	01011010	101̄01̄010

- <55>
- <56> 여기서, 회전인자에 해당하는 복소 곱셈계수는 삼각함수 $\sin(\pi/4) = \cos(\pi/4)$, $\cos(\pi/8)$, $\sin(\pi/8)$ 인데, 회전인자인 $W(8)$, $W(16)$, $W(24)$, $W(48)$ 로 나타낼 수 있다.
- <57> 그리고, 상수 곱셈기는 곱셈 계수 중에 '0' 이 아닌 비트의 위치에 따라 입력의 값 비트를 시프트하여 부분곱을 생성한 후, 서로 더하여 결과를 얻는다. 계수가 N 비트이면 곱셈기에서 발생하는 부분곱의 개수는 최대 N 개이고, 주로 면적을 차지하는 부분이 부분곱에 대한 가산이므로, 면적을 줄이기 위하여 부분곱을 감소시켜야 한다.
- <58> 그래서, 곱셈기 설계시, 면적을 감소시키는 방법은 2의 보수로 표현된 곱셈 계수를 CSD 방식으로 표현하는 것인데, CSD는 {-1, 0, 1}의 디지털 세트트를 가지며, 두개이상의 연속적인 '1'의 비트를 가질 수 없다. 이렇게 변환된 CSD 표현식은 2의 보수와 비교하여 '1'의 개수를 줄일 수 있다.
- <59> 이때, 1bar는 -1 을 나타낸다.
- <60> 도 4a는 본 발명에 따른 $\sin(\pi/8)$ 를 위한 CSD 복소 상수 곱셈기를 도시하는 블록도이고, 도 4b는 본 발명에 따른 $\cos(\pi/8)$ 를 위한 CSD 복소 상수 곱셈기를 도시하는 블록도이며, 도 4c는 본 발명에 따른 $\sin(\pi/4) = \cos(\pi/4)$ 를 위한 CSD 복소 상수 곱셈기를 도시하는 블록도이다. 도면에서 도시하고 있는 바와 같이, $\sin(\pi/4) = \cos(\pi/4)$, $\cos(\pi/8)$, $\sin(\pi/8)$ 을 위하여 구비되며, $\sin(\pi/8)$ 는 에러 보상 바이어스(C)와 반가산기와 벡터 합산 가산기를 포함하며, $\cos(\pi/8)$ 는 에러 보상 바이어스(C1,C2)와 전가산기와 반 가산기 및 벡터 합산 가산기를 포함하고, $\sin(\pi/4) = \cos(\pi/4)$ 는 에러 보상 바이어스(C)와 전가산기와 벡터 합산 가산기를 포함하여 이루어진다.
- <61> 여기서, 에러 보상 바이어스는 양자화 에러를 효율적으로 보상하기 위하여, 양자화 에러에 미치는 영향에 따라 버림수를 주 그룹과 부 그룹으로 구성하고, 주 그룹 버림 비트에 의하여 표현되며, 버림 비트의 영향을 확률을 이용하여 반영시키는데, 에러 보상 바이어스의 총 합은 C또는 C1, C2 로 정의한다.
- <62> 한편, 일반적인 N X N 비트의 2의 보수 곱셈기의 출력은 (2N - 1) 입력이며, 입력과 같은 N 비트의 출력만을 얻고자 한다면, (2N - 1) 비트의 결과 중에서 (N - 1) LSB 를 생략하여 N 비트로 양자화하는 방법을 사용하는데, 특히 FFT 프로세서는 디지털 연산의 반복으로 이러한 방법을 적용하면, 기존의 곱셈기를 그대로 사용할 수 있는 장점이 있으나, 양자화 에러가 증가할 뿐만 아니라 하드웨어의 낭비를 가져오며, 이에 따라 처음부터 출력 비트를 N 개로 고정시키고, CSD 곱셈기 설계시 가장 작은 에러를 갖도록 에러 보상 바이어스를 구현하여 결과를 얻는 방법을 사용한다.
- <63> 도 5은 본 발명에 따른 CSD 복소 상수 곱셈기를 개략적으로 나타내는 블록 구성도이다. 도면에서 도시하고 있는 바와 같이, CSD 복소 상수 곱셈기는 6개의 CSD 복소 상수 곱셈기와 2개의 비교기를 포함하여 이루어진다.
- <64> 여기서, 각각의 실수 데이터와 허수 데이터를 입력하여 6개의 CSD 복소 상수 곱셈기의 결과 데이터로 각각의 실수 데이터와 허수 데이터를 출력시킨다.

표 2

Time Sequence	0	1	2	3	4	5	6	7
Block1	0	0	0	0	0	W(16)	-j	W(48)
Block2	0	W(8)	W(16)	W(24)	0	W(24)	W(48)	W(8)
0.3827	X	2	X	2	X	2	X	2
0.9239	X	2	X	2	X	2	X	2
0.7071	X	X	2	X	X	2	2	2

<65>

<66>

표 2는 각각의 데이터 경로의 회전인자 연산 스케줄링을 도시한다. 도 5와 같이, 3가지의 종류를 가지는 6개의 CSD 복소 상수 곱셈기를 공유하여 본 발명에 따른 회전인자연산을 순차적으로 수행한다.

<67>

본 발명의 실시예에서는 본 발명에 따른 병렬 구조 및 파이프라인 방식을 이용하여 128포인트 Radix 2의4승 고속 푸리에 변환 프로세서를 하드웨어적으로 구현한다.

<68>

도 6은 본 발명에 따른 FFT 프로세서의 내부위드길이에 따른 잡음률을 도시한 그래프이다. 도면에서 도시하고 있는 바와 같이, 적절한 위드 길이를 결정하기 위하여 하드웨어 구현에 앞서 고정 소수점 시뮬레이션을 수행한다.

<69>

여기서, 입력 잡음률(IN SNR)은 FFT에 입력되는 신호에 따른 채널에 의한 AWGN(Additive White Gaussian Noise)의 비율이며, 출력 잡음률(OUT SNR)은 AWGN과 고정 소수점 연산에 따른 양자화 잡음에 의한 입력 신호비(Input Signal Ratio)이다.

<70>

이때, 도면에서 도시하고 있는 바와 같이 입력 잡음률을 고정한 상태에서 위드의 길이를 증가시키면, 양자화 잡음이 감소하여 출력 잡음률이 증가함을 보이며, 출력 잡음률이 입력 잡음률과 비슷해질수록 수렴하는데, 이는 양자화 잡음을 무시가능한 것이다.

<71>

이상에서는 본 발명의 바람직한 실시예를 예시적으로 설명하였으나, 본 발명의 범위는 이 같은 특정 실시예에만 한정되지 않으며 해당 분야에서 통상의 지식을 가진자라면 본 발명의 특허 청구 범위내에 기재된 범주 내에서 적절하게 변경이 가능 할 것이다.

발명의 효과

<72>

이상에서 설명한 바와 같이 상기와 같은 구성을 갖는 본 발명은 2개의 병렬 데이터 경로의 구조 및 단일 경로 딜레이 피드백 구조를 이용함으로써, 동작 속도 및 데이터 처리율을 증가시키고 동시에 구성을 용이하도록 하드웨어의 복잡도를 감소시키도록 구현하고, Radix - 2⁴ 알고리즘을 고속 푸리에 변환에 적용시킴으로써, 프로그램래블 복소 곱셈기의 절반에 해당하는 곱셈량을 CSD 방식을 이용한 특수 복소 상수 곱셈기로 설계하여 하드웨어 비용 및 전력 소모를 감소시킬 수 있는 등의 효과를 거둘 수 있다.

도면의 간단한 설명

<1>

도 1은 본 발명에 따른 128 포인트 Radix-2⁴ SDF FFT 알고리즘의 개략적인 신호 흐름도.

<2>

도 2은 본 발명에 따른 Radix-2⁴ 고속 푸리에 변환 구조를 개략적으로 나타내는 블록도.

<3>

도 3은 본 발명에 따른 Radix-2⁴ 버터플라이 연산부의 개략적인 구성도.

<4>

도 4a는 본 발명에 따른 sin($\pi/8$)를 위한 CSD 복소 상수 곱셈기를 도시하는 블록도.

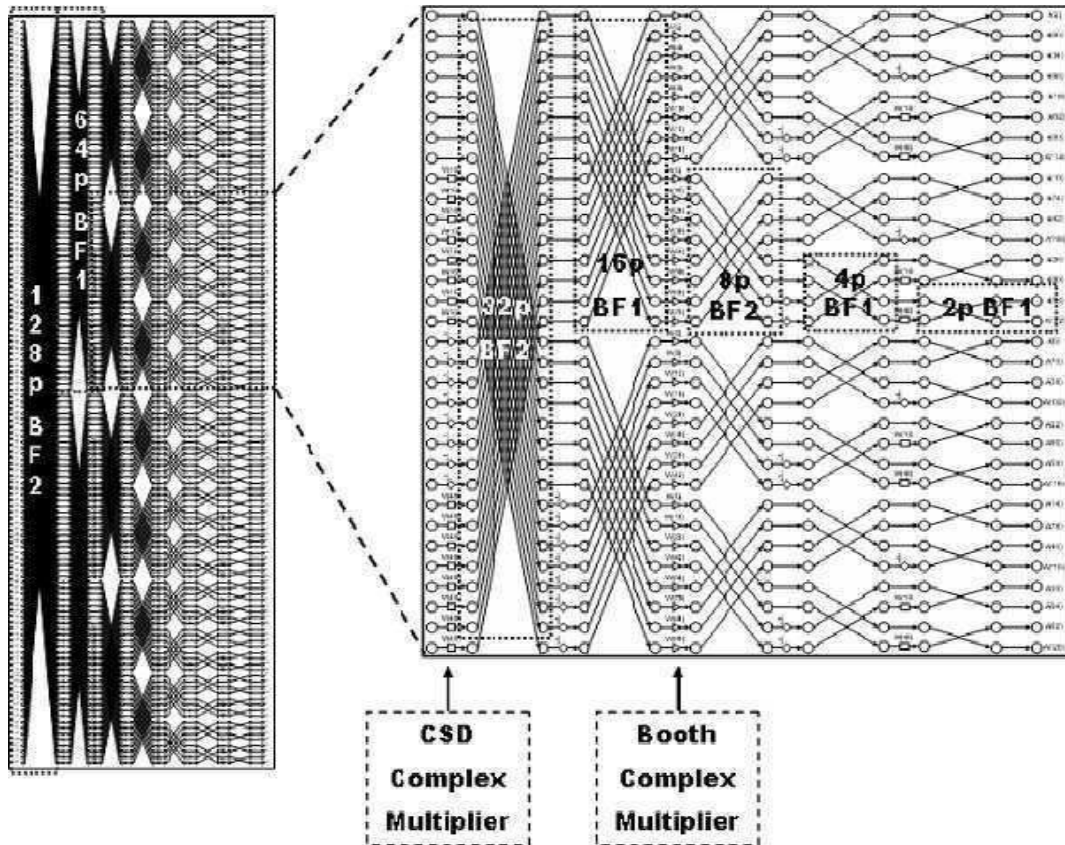
<5>

도 4b는 본 발명에 따른 cos($\pi/8$)를 위한 CSD 복소 상수 곱셈기를 도시하는 블록도.

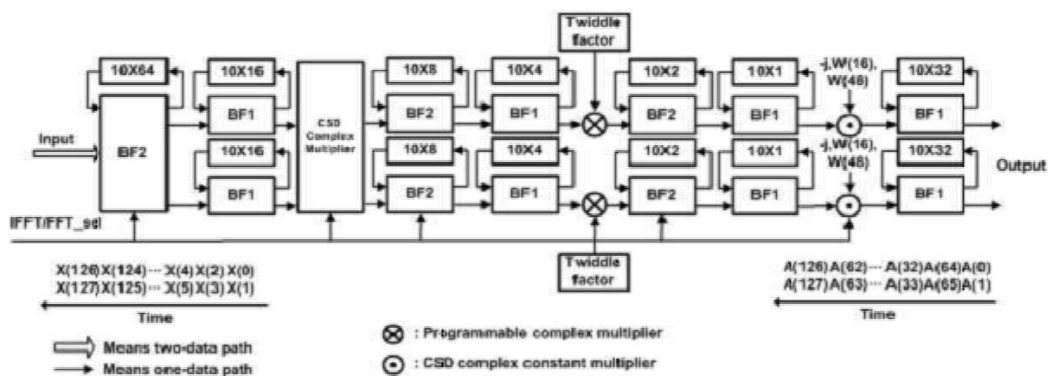
- <6> 도 4c는 본 발명에 따른 $\sin(\pi/4) = \cos(\pi/4)$ 를 위한 CSD 복소 상수 곱셈기를 도시하는 블록도.
- <7> 도 5은 본 발명에 따른 CSD 복소 상수 곱셈기를 개략적으로 나타내는 블록 구성도.
- <8> 도 6은 본 발명에 따른 FFT 프로세서의 내부위드길이에 따른 잡음률을 도시한 그래프.

도면

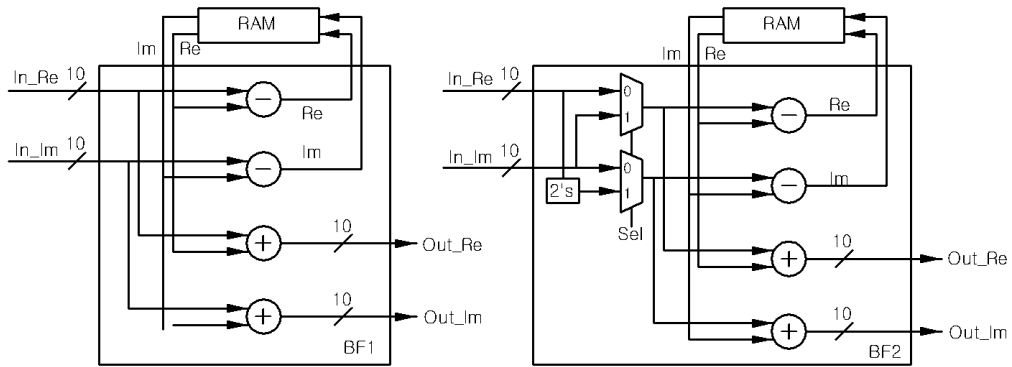
도면1



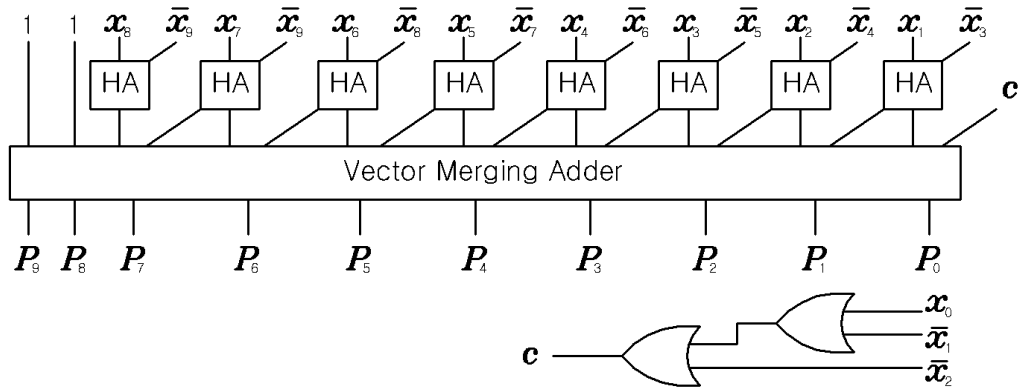
도면2



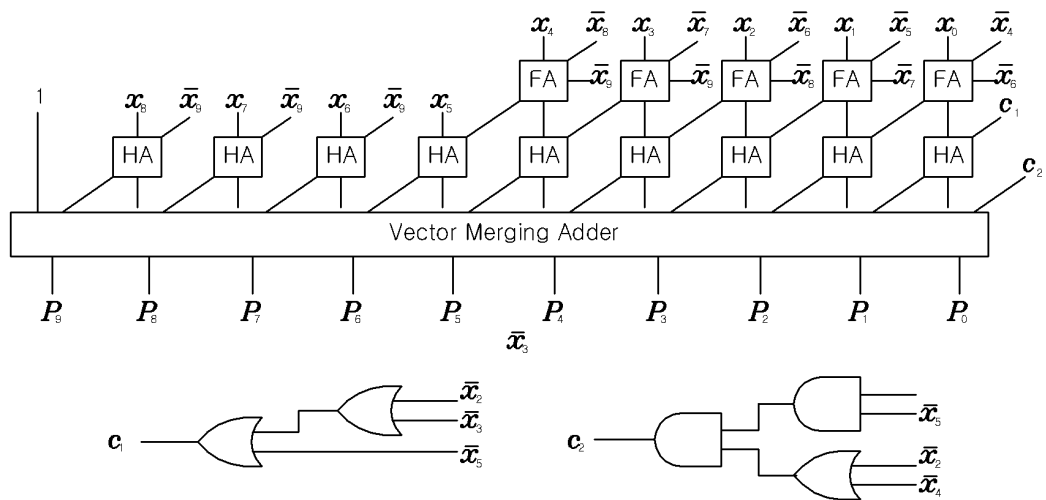
도면3



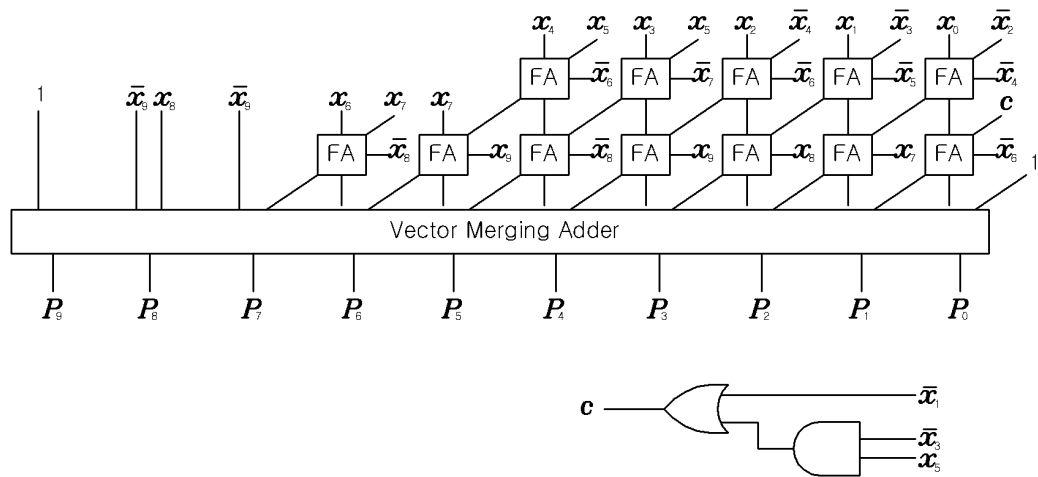
도면4a



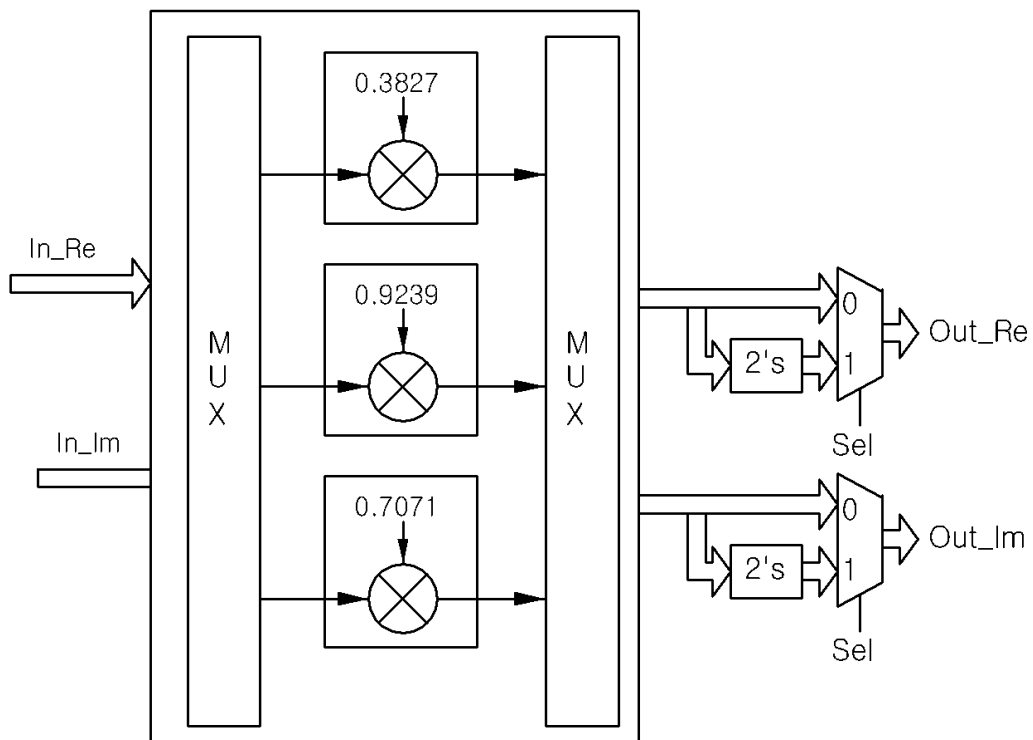
도면4b



도면4c



도면5



도면6

