(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0073890
(43) 공개일자 2025년05월27일

(51) 국제특허분류(Int. Cl.)
G06F 30/3308 (2020.01) *G06F 17/14* (2006.01)
(52) CPC특허분류
G06F 30/3308 (2020.01)
G06F 17/142 (2013.01)
(21) 출원번호 10-2023-0160360
(22) 출원일자 2023년11월20일
심사청구일자 2023년11월20일

(71) 출원인
인하대학교 산학협력단
인천광역시 미추홀구 인하로 100(용현동, 인하대학교)
(72) 발명자
서영교
경기도 부천시 길주로77번길 19-45, 920호(상동, 디아뜨갤러리4오피스텔)
김종혁
서울특별시 서대문구 수색로 100, 110동 801호(북가좌동, DMC 래미안 e편한세상)
조경선
경기도 화성시 동탄기흥로 393-20(오산동, 동탄역 파라곤)
(74) 대리인
양성보

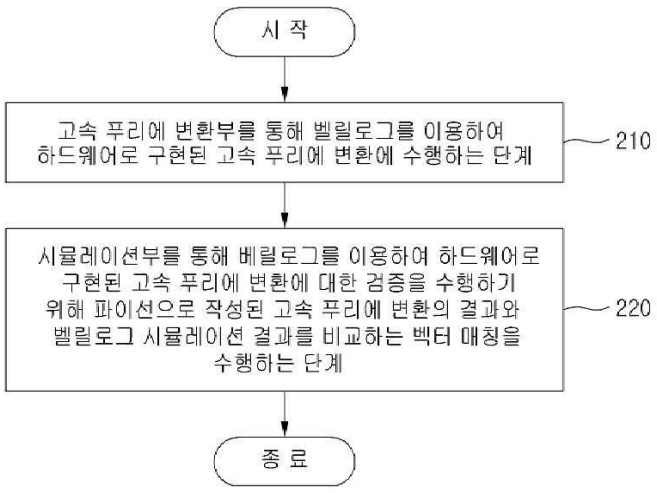
전체 청구항 수 : 총 8 항

(54) 발명의 명칭 **고속 푸리에 변환 하드웨어 가속기 설계 및 검증 방법 및 시스템**

(57) 요 약

고속 푸리에 변환 하드웨어 가속기 설계 및 검증 방법 및 시스템이 제시된다. 본 발명에서 제안하는 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 방법은 고속 푸리에 변환부를 통해 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환을 수행하는 단계 및 시뮬레이션부를 통해 상기 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환에 대한 검증을 수행하기 위해 파이선으로 작성된 고속 푸리에 변환의 결과와 베릴로그 시뮬레이션 결과를 비교하는 벡터 매칭을 수행하는 단계를 포함한다.

대 표 도 - 도2



이 발명을 지원한 국가연구개발사업
과제고유번호 1711194127
과제번호 2021M3F3A2A01037531
부처명 과학기술정보통신부
과제관리(전문)기관명 한국연구재단
연구사업명 신소자원천기술개발
연구과제명 [Ezbaro] Dynamic logic 기반 고성능 STMG 로직 회로 설계
기 여 율 3/10
과제수행기관명 인하대학교
연구기간 2023.01.01 ~ 2023.12.31
이 발명을 지원한 국가연구개발사업
과제고유번호 1415186528
과제번호 20009972
부처명 산업통상자원부
과제관리(전문)기관명 한국산업기술기획평가원
연구사업명 차세대지능형반도체기술개발
연구과제명 고효율 초저전력 경량 엣지 디바이스용 소자회로 및 SoC 개발(4차년도)
기 여 율 3/10
과제수행기관명 인하대학교
연구기간 2023.01.01 ~ 2023.12.31
이 발명을 지원한 국가연구개발사업
과제고유번호 1711193592
과제번호 2021-0-02052-003
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 대학ITRC센터육성지원사업(교육운영수익)
연구과제명 [Ezbaro][정부] 스마트 모빌리티를 위한 인공지능 시스템반도체 핵심 기술 개발 및 인력 양성(3차년도)
기 여 율 4/10
과제수행기관명 인하대학교
연구기간 2023.01.01 ~ 2023.12.31

명 세 서

청구범위

청구항 1

고속 푸리에 변환부를 통해 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환을 수행하는 단계; 및

시뮬레이션부를 통해 상기 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환에 대한 검증을 수행하기 위해 파이선으로 작성된 고속 푸리에 변환의 결과와 베릴로그 시뮬레이션 결과를 비교하는 벡터 매칭을 수행하는 단계

를 포함하는 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 방법.

청구항 2

제1항에 있어서,

상기 고속 푸리에 변환부를 통해 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환을 수행하는 단계는,

상기 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환의 연산속도를 감소시키기 위해 분할 정복 알고리즘인 쿨리-튜키 알고리즘(Cooley-Tukey algorithm)을 적용하여 고속 푸리에 변환을 수행하고, 복수의 점(point)에 대한 고속 푸리에 변환을 수행하기 위한 복소수 입력으로 상기 복수의 점에 대한 실수부와 허수부를 각각 입력 받고, 출력도 상기 복수의 점에 대한 실수부와 허수부로 각각 출력되는

고속 푸리에 변환 하드웨어 가속기 설계 및 검증 방법.

청구항 3

제1항에 있어서,

상기 시뮬레이션부를 통해 상기 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환에 대한 검증을 수행하기 위해 파이선으로 작성된 고속 푸리에 변환의 결과와 베릴로그 시뮬레이션 결과를 비교하는 벡터 매칭을 수행하는 단계는,

벡터 매칭을 수행하기 위해 시뮬레이션 할 베릴로그 파일 이름, 베릴로그 테스트벤치 파일 이름, 고속 푸리에 변환인지 또는 역 고속 푸리에 변환인지 여부, 시뮬레이션에 활용한 입력이 들어있는 파일 경로, 시뮬레이션 수행 횟수를 입력 받아 상기 입력된 정보에 따라 파이선으로 작성된 고속 푸리에 변환을 수행하여 파이선 시뮬레이션 결과를 저장하고, 상기 입력된 정보에 따라 베릴로그 파일과 베릴로그 테스트벤치 파일을 열어 베릴로그 시뮬레이션을 수행하고 결과를 저장하는

고속 푸리에 변환 하드웨어 가속기 설계 및 검증 방법.

청구항 4

제3항에 있어서,

상기 시뮬레이션부를 통해 상기 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환에 대한 검증을 수행하기 위해 파이선으로 작성된 고속 푸리에 변환의 결과와 베릴로그 시뮬레이션 결과를 비교하는 벡터 매칭을 수행하는 단계는,

상기 파이선 시뮬레이션 결과 및 베릴로그 시뮬레이션 결과를 비교하여, 두 결과값이 상이할 경우, 해당 시뮬레이션에 대한 파이선 시뮬레이션 결과 및 베릴로그 시뮬레이션 결과와 총 시뮬레이션 횟수 중 상기 두 결과값이 상이한 경우의 횟수를 확인하는

고속 푸리에 변환 하드웨어 가속기 설계 및 검증 방법.

청구항 5

베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환을 수행하는 고속 푸리에 변환부; 및

상기 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환에 대한 검증을 수행하기 위해 파이선으로 작성된 고속 푸리에 변환의 결과와 베릴로그 시뮬레이션 결과를 비교하는 벡터 매칭을 수행하는 시뮬레이션부

를 포함하는 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 시스템.

청구항 6

제5항에 있어서,

상기 고속 푸리에 변환부는,

상기 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환의 연산속도를 감소시키기 위해 분할 정복 알고리즘인 쿨리-튜키 알고리즘(Cooley-Tukey algorithm)을 적용하여 고속 푸리에 변환을 수행하고, 복수의 점(point)에 대한 고속 푸리에 변환을 수행하기 위한 복소수 입력으로 상기 복수의 점에 대한 실수부와 허수부를 각각 입력 받고, 출력도 상기 복수의 점에 대한 실수부와 허수부로 각각 출력되는

고속 푸리에 변환 하드웨어 가속기 설계 및 검증 시스템.

청구항 7

제5항에 있어서,

상기 시뮬레이션부는,

벡터 매칭을 수행하기 위해 시뮬레이션 할 베릴로그 파일 이름, 베릴로그 테스트벤치 파일 이름, 고속 푸리에 변환인지 또는 역 고속 푸리에 변환인지 여부, 시뮬레이션에 활용한 입력이 들어있는 파일 경로, 시뮬레이션 수행 횟수를 입력 받아 상기 입력된 정보에 따라 파이선으로 작성된 고속 푸리에 변환을 수행하여 파이선 시뮬레이션 결과를 저장하고, 상기 입력된 정보에 따라 베릴로그 파일과 베릴로그 테스트벤치 파일을 열어 베릴로그 시뮬레이션을 수행하고 결과를 저장하는

고속 푸리에 변환 하드웨어 가속기 설계 및 검증 시스템.

청구항 8

제7항에 있어서,

상기 시뮬레이션부는,

상기 파이선 시뮬레이션 결과 및 베릴로그 시뮬레이션 결과를 비교하여, 두 결과값이 상이할 경우, 해당 시뮬레이션에 대한 파이선 시뮬레이션 결과 및 베릴로그 시뮬레이션 결과와 총 시뮬레이션 횟수 중 상기 두 결과값이 상이한 경우의 횟수를 확인하는

고속 푸리에 변환 하드웨어 가속기 설계 및 검증 시스템.

발명의 설명

기 술 분 야

[0001] 본 발명은 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 방법 및 시스템에 관한 것이다.

배 경 기 술

[0002] 푸리에 변환(Fourier transform; FT)은 시간에 대한 함수 및 신호를 주파수에 대한 함수 및 신호로 변환하는 것이다. 반대로 주파수에 대한 함수 및 신호를 시간에 대한 함수로 변환하면 역 푸리에 변환(Inverse Fourier transform; IFT) 이라고 한다. 그 식은 다음과 같다.

[0003] $$Y(f) = \int_{-\infty}^{\infty} y(t) e^{-i2\pi ft} dt \quad (1)$$

[0004] $$y(t) = \int_{-\infty}^{\infty} Y(f) e^{i2\pi ft} dt \quad (2)$$

[0005] 상기 식에서 확인할 수 있듯이, FT 기준 신호를 어떤 복소수와 곱해서 모든 시간에 대해 적분하면 주파수에 대한 함수를 얻을 수 있다. 하지만 이 식을 컴퓨터를 이용해서 계산을 할 때 문제가 있다. 위 식은 연속 시간, 연속 주파수에 대한 계산이지만, 컴퓨터의 모든 신호는 이산 신호, 즉 디지털 신호이다. 따라서 이산 시간에 대한 신호에서 이산 주파수 신호로 변환을 하는 것이 이산 푸리에 변환 (Discrete Fourier Transform; DFT) 이다.

[0006] $$Y(k) = \sum_{n=0}^{N-1} y(n) e^{-i\frac{2\pi}{N}kn}, 0 \le k < N \quad (3)$$

[0007] 이와 같이 컴퓨터를 이용해 푸리에 변환 계산이 가능해졌지만, 여전히 연산량이 많다는 문제가 존재한다. 상기 식에서 확일할 수 있듯이, 입력이 N개일 때, N번의 곱셈 연산한 것을 N번의 덧셈 연산을 행하기 때문에 O(N^2)의 시간 복잡도를 갖는다.

선행기술문헌

비특허문헌

[0008] (비특허문헌 0001) [1] W. T. Cochran et al., "What is the fast Fourier transform?," in Proceedings of the IEEE, vol. 55, no. 10, pp. 1664-1674, Oct. 1967.

(비특허문헌 0002) [2] J. W. Cooley, P. A. W. Lewis and P. D. Welch, "The Fast Fourier Transform and Its Applications," in IEEE Transactions on Education, vol. 12, no. 1, pp. 27-34, March 1969.

(비특허문헌 0003) [3] Johnsson, S. Lennart, and Robert L. Krawitz. "Cooley-tukey FFT on the connection machine." Parallel Computing, pp 1201-1221, 1992.

발명의 내용

해결하려는 과제

[0009] 본 발명이 이루고자 하는 기술적 과제는 고속 푸리에 변환의 연산 속도를 감소시키기 위해 고속 푸리에 변환을 베릴로그를 이용하여 하드웨어로 구현하고, 구현한 것의 동작 검증을 위해 파이선으로 작성한 고속 푸리에 변환의 결과값과 베릴로그 시뮬레이션 결과를 많은 입력에 대해 비교하는 벡터 매칭을 수행함으로써, 설계된 고속 푸리에 변환 하드웨어 가속기를 검증하는 방법 및 시스템을 제공하는데 있다.

과제의 해결 수단

[0010] 일 측면에 있어서, 본 발명에서 제안하는 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 방법은 고속 푸리에 변환부를 통해 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환을 수행하는 단계 및 시뮬레이션부를 통해 상기 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환에 대한 검증을 수행하기 위해 파이선으로 작성된 고속 푸리에 변환의 결과와 베릴로그 시뮬레이션 결과를 비교하는 벡터 매칭을 수행하는 단계를 포함한다.

[0011] 상기 고속 푸리에 변환부를 통해 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환을 수행하는 단계는 상기 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환의 연산속도를 감소시키기 위해 분할 정복 알고리즘인 쿨리-튜키 알고리즘(Cooley-Tukey algorithm)을 적용하여 고속 푸리에 변환을 수행하고, 복수의 점(point)에 대한 고속 푸리에 변환을 수행하기 위한 복소수 입력으로 상기 복수의 점에 대한 실수부와 허수부를 각각 입력 받고, 출력도 상기 복수의 점에 대한 실수부와 허수부로 각각 출력된다.

[0012] 상기 시뮬레이션부를 통해 상기 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환에 대한 검증을 수행하기 위해 파이선으로 작성된 고속 푸리에 변환의 결과와 베릴로그 시뮬레이션 결과를 비교하는 벡터 매칭을 수행하는 단계는 벡터 매칭을 수행하기 위해 시뮬레이션 할 베릴로그 파일 이름, 베릴로그 테스트벤치 파일 이름, 고속 푸리에 변환인지 또는 역 고속 푸리에 변환인지 여부, 시뮬레이션에 활용한 입력이 들어있는 파일 경로, 시뮬레이션 수행 횟수를 입력 받아 상기 입력된 정보에 따라 파이선으로 작성된 고속 푸리에 변환을 수행하여 파이선 시뮬레이션 결과를 저장하고, 상기 입력된 정보에 따라 베릴로그 파일과 베릴로그 테스트벤치 파일을 열어 베릴로그 시뮬레이션을 수행하고 결과를 저장한다.

[0013] 상기 시뮬레이션부를 통해 상기 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환에 대한 검증을 수행하기 위해 파이선으로 작성된 고속 푸리에 변환의 결과와 베릴로그 시뮬레이션 결과를 비교하는 벡터 매칭을 수행하는 단계는 상기 파이선 시뮬레이션 결과 및 베릴로그 시뮬레이션 결과를 비교하여, 두 결과값이 상이할 경우, 해당 시뮬레이션에 대한 파이선 시뮬레이션 결과 및 베릴로그 시뮬레이션 결과와 총 시뮬레이션 횟수 중 상기 두 결과값이 상이한 경우의 횟수를 확인한다.

[0014] 또 다른 일 측면에 있어서, 본 발명에서 제안하는 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 시스템은 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환을 수행하는 고속 푸리에 변환부 및 상기 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환에 대한 검증을 수행하기 위해 파이선으로 작성된 고속 푸리에 변환의 결과와 베릴로그 시뮬레이션 결과를 비교하는 벡터 매칭을 수행하는 시뮬레이션부를 포함한다.

발명의 효과

[0015] 본 발명의 실시예들에 따르면 고속 푸리에 변환을 베릴로그를 이용하여 하드웨어로 구현하고, 구현한 것의 동작 검증을 위해 파이선으로 작성한 고속 푸리에 변환의 결과값과 베릴로그 시뮬레이션 결과를 많은 입력에 대해 비교하는 벡터 매칭을 수행함으로써, 고속 푸리에 변환의 연산 속도를 감소시고, 설계된 고속 푸리에 변환 하드웨어 가속기를 검증할 수 있다.

도면의 간단한 설명

[0016] 도 1은 본 발명의 일 실시예에 따른 버터플라이 고속 푸리에 변환 다이어그램을 나타내는 도면이다.

도 2는 본 발명의 일 실시예에 따른 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 방법을 설명하기 위한 흐름도이다.

도 3은 본 발명의 일 실시예에 따른 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 시스템의 구성을 나타내는 도면이다.

도 4는 본 발명의 일 실시예에 따른 베릴로그 구현 다이어그램을 나타내는 도면이다.

도 5는 본 발명의 일 실시예에 따른 버터플라이 유닛 모듈을 나타내는 도면이다.

도 6은 본 발명의 일 실시예에 따른 벡터 매칭 과정을 설명하기 위한 도면이다.

도 7은 본 발명의 일 실시예에 따른 벡터 매칭 결과를 나타내는 예시도이다.

발명을 실시하기 위한 구체적인 내용

[0017] 정보통신 및 전자 분야에서 자주 사용되는 푸리에 변환은 컴퓨터로 연산 시 그 계산속도가 매우 느리다. 분할 정복 알고리즘인 쿨리-튜키 알고리즘(Cooley-Tukey algorithm)을 적용한 고속 푸리에 변환은 푸리에 변환의 연산속도를 획기적으로 줄여준다. 본 발명에서는 고속 푸리에 변환을 베릴로그를 이용하여 하드웨어로 구현하고, 구현한 것의 동작 검증을 위해 파이선으로 작성한 고속 푸리에 변환의 결과값과 베릴로그 시뮬레이션 결과를 많은 입력에 대해 비교하는 벡터 매칭을 진행해 검증 한다. 이하, 본 발명의 실시 예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0019] 도 1은 본 발명의 일 실시예에 따른 버터플라이 고속 푸리에 변환 다이어그램을 나타내는 도면이다.

[0020] 본 발명의 실시예에 따르면, 고속 푸리에 변환(Fast Fourier Transform; FFT)이 O(N^2) 의 시간 복잡도를 가지는 DFT(Discrete Fourier Transform)를 O(NlogN) 시간 복잡도로 계산할 수 있도록 해준다. 이를 위해 분할 정복 알고리즘(예를 들어, 쿨리-튜키 알고리즘(Cooley-Tukey algorithm))을 활용한다. 분할 정복 알고리즘은 N개의 입력에 대한 계산을 N/2개씩 나누어 더 이상 나눌 수 없을 때까지 계산을 하고, 다시 합치는 방식이다.

[0021] 먼저 N개의 데이터 y(n) 을 DFT 한다고 하자. y(n)을 상기 DFT 식에 넣고, 짝수항과 홀수항으로 나눠 식을 전개하면 다음과 같다:

[0022] $$Y(k) = \sum_{n=0}^{N-1} y(n) e^{-i\frac{2\pi}{N}kn}$$

[0023] $$= \sum_{n=even}^{N-1} y(n)e^{-i\frac{2\pi}{N}kn} + \sum_{n=odd}^{N-1} y(n)e^{-i\frac{2\pi}{N}kn}$$

[0024] $$= \sum_{n=0}^{\frac{N}{2}-1} y(2n)e^{-i\frac{2\pi}{N}k(2n)} + \sum_{n=0}^{\frac{N}{2}-1} y(2n+1)e^{-i\frac{2\pi}{N}k(2n)} \quad (4)$$

[0025] 그리고 짝수항, 홀수항을 각각 p(n), q(n)이라 하고 식을 전개하면 다음과 같다:

[0026] $$\sum_{n=0}^{\frac{N}{2}-1} p(n)e^{-i\frac{2\pi}{\frac{N}{2}}kn} + \sum_{n=0}^{\frac{N}{2}-1} q(n)e^{-i\frac{2\pi}{N}k(2n)}e^{-i\frac{2\pi}{N}k}$$

[0027] $$= \sum_{n=0}^{\frac{N}{2}-1} p(n)e^{-i\frac{2\pi}{\frac{N}{2}}kn} + e^{-i\frac{2\pi}{N}k}\sum_{n=0}^{\frac{N}{2}-1} q(n)e^{-i\frac{2\pi}{\frac{N}{2}}k(2n)}, \quad 0 \le k \le n-1 \quad (5)$$

[0028] 상기 식의 마지막 항을 다시보면, 결국 첫 번째항은 짝수항의 DFT, 두 번째항은 어떤 상수에 홀수항의 DFT 가 곱해진 꼴임을 확인할 수 있다. 따라서 짝수항의 DFT 를 P(K), 홀수항의 DFT 를 Q(K) 라고 하면 식을 다음과 같이 나타낼 수 있다:

[0029] $$Y(k) = P(k) + e^{i\frac{2\pi}{N}k}Q(k), 0 \le k \le N-1 \quad (6)$$

[0030] Q(k) 앞의 상수는 다음과 같이 나타낼 수 있다:

[0031] $$W^k = e^{-i\frac{2\pi}{N}k} \quad (7)$$

[0032] 그리고 k=0 에서부터 하나씩 전개해보면 다음과 같이 나타낼 수 있다:

[0033] $$Y(0) = P(0) + W^0Q(0)$$
$$Y(1) = P(1) + W^1Q(1)$$
$$\cdots$$
$$Y(N-1) = P(N-1) + W^{N-1}Q(N-1) \quad (3)$$

[0034] 여기서 확인할 수 있는 것은 연산량이 N번의 덧셈 연산과 N번의 곱셈 연산. 총 2N 번 일어남을 확인할 수 있다. 시간 복잡도를 계산할 때는 앞의 상수는 무시할 수 있으므로 N번 연산이 일어났다고 볼 수 있다.

[0035] 이번엔 P(N)의 짝수항, 홀수항을 각각 A(n), B(n) 이라하고, Q(N)의 짝수항, 홀수항을 각각 C(n), D(n) 이라 하고 전개하면 다음과 같이 나타낼 수 있다:

[0036] $$Q(0) = C(0) + W^0D(0)$$
$$Q(1) = C(1) + W^1D(1)$$
$$\cdots$$
$$Q(\frac{N}{2}-1) = C(\frac{N}{2}-1) + W^{\frac{N}{2}-1}D(\frac{N}{2}-1) \quad (9)$$

[0037] $$P(0) = A(0) + W^0B(0)$$
$$P(1) = A(1) + W^1B(1)$$
$$\cdots$$
$$P(\frac{N}{2}-1) = A(\frac{N}{2}-1) + W^{\frac{N}{2}-1}Q(\frac{N}{2}-1) \quad (10)$$

[0038] 상기 식에서 확인할 수 있듯이, P(N)을 구하는데 N/2, Q(N) 을 구하는데 N/2, 총 N번의 연산이 필요하다. 이 과정을 원소가 2개가 남아 더 이상 나눌 수 없을 때까지 반복되고, 2개가 남았을 때도 마찬가지로 N번의 연산이 필요하다.

[0039] N 에서 시작하여 2개씩 나누어 최종적으로 2개 남을 때까지 수행되는 횟수는 Log2(N) 이다. 따라서 시간 복잡도는 O(NlogN) 이라 할 수 있고, DFT 에 비해 훨씬 개선된 속도를 보인다.

[0040] 여기서 DFT 의 입력이 N개 일 때, 주기가 N 이라는 점과, 복소지수함수의 성질을 이용하면, 식을 다음과 같이 나타낼 수 있다:

[0041] $$P(k) = P(k+\frac{N}{2}) \quad (11)$$

[0042] $$Q(k) = Q(k+\frac{N}{2}) \quad (12)$$

[0043] $$e^{i\frac{2\pi}{N}(k+\frac{N}{2})} = -e^{-i\frac{2\pi}{N}k} \quad (13)$$

[0044] $$Y(k) = P(k) + W^k Q(k), 0 \le k \le \frac{N}{2}-1$$
$$Y(k+\frac{N}{2}) = P(k) - W^k Q(k), 0 \le k \le \frac{N}{2}-1 \quad (14)$$

[0045] FFT의 식은 최종적으로 상기 식과 같이 표현된다. 두 개의 항을 곱하고, 나머지 항과 덧셈을 하거나 뺏셈을 하는 것의 반복이다. 이러한 연산을 버터플라이 연산(butterfly operation)이라하며, 그림으로 표현하면 도 1과 같다.

[0047] 도 2는 본 발명의 일 실시예에 따른 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 방법을 설명하기 위한 흐름도이다.

[0048] 제안하는 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 방법은 고속 푸리에 변환부를 통해 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환을 수행하는 단계(210) 및 시뮬레이션부를 통해 상기 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환에 대한 검증을 수행하기 위해 파이선으로 작성된 고속 푸리에 변환의 결과와 베릴로그 시뮬레이션 결과를 비교하는 벡터 매칭을 수행하는 단계(220)를 포함한다.

[0049] 단계(210)에서, 상기 고속 푸리에 변환부를 통해 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환을 수행한다.

[0050] 본 발명의 실시예에 따르면, 상기 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환의 연산속도를 감소시키기 위해 분할 정복 알고리즘인 쿨리-튜키 알고리즘(Cooley-Tukey algorithm)을 적용하여 고속 푸리에 변환을 수행하고, 복수의 점(point)에 대한 고속 푸리에 변환을 수행하기 위한 복소수 입력으로 상기 복수의 점에 대한 실수부와 허수부를 각각 입력 받고, 출력도 상기 복수의 점에 대한 실수부와 허수부로 각각 출력된다.

[0051] 단계(220)에서, 상기 시뮬레이션부를 통해 상기 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환에 대한 검증을 수행하기 위해 파이선으로 작성된 고속 푸리에 변환의 결과와 베릴로그 시뮬레이션 결과를 비교하는 벡터 매칭을 수행한다.

[0052] 본 발명의 실시예에 따르면, 벡터 매칭을 수행하기 위해 시뮬레이션 할 베릴로그 파일 이름, 베릴로그 테스트벤치 파일 이름, 고속 푸리에 변환인지 또는 역 고속 푸리에 변환인지 여부, 시뮬레이션에 활용한 입력이 들어있는 파일 경로, 시뮬레이션 수행 횟수를 입력 받아 상기 입력된 정보에 따라 파이선으로 작성된 고속 푸리에 변환을 수행하여 파이선 시뮬레이션 결과를 저장하고, 상기 입력된 정보에 따라 베릴로그 파일과 베릴로그 테스트벤치 파일을 열어 베릴로그 시뮬레이션을 수행하고 결과를 저장한다.

[0053] 본 발명의 실시예에 따르면, 상기 파이선 시뮬레이션 결과 및 베릴로그 시뮬레이션 결과를 비교하여, 두 결과값이 상이할 경우, 해당 시뮬레이션에 대한 파이선 시뮬레이션 결과 및 베릴로그 시뮬레이션 결과와 총 시뮬레이션 횟수 중 상기 두 결과값이 상이한 경우의 횟수를 확인한다.

[0055] 도 3은 본 발명의 일 실시예에 따른 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 시스템의 구성을 나타내는

도면이다.

[0056] 본 실시예에 따른 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 시스템(300)은 프로세서(310), 버스(320), 네트워크 인터페이스(330), 메모리(340) 및 데이터베이스(350)를 포함할 수 있다. 메모리(340)는 운영체제(341) 및 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 루틴(342)을 포함할 수 있다. 프로세서(310)는 고속 푸리에 변환부(311) 및 시뮬레이션부(312)를 포함할 수 있다. 다른 실시예들에서 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 시스템(300)은 도 3의 구성요소들보다 더 많은 구성요소들을 포함할 수도 있다. 그러나, 대부분의 종래기술적 구성요소들을 명확하게 도시할 필요성은 없다. 예를 들어, 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 시스템(300)은 디스플레이나 트랜시버(transceiver)와 같은 다른 구성요소들을 포함할 수도 있다.

[0057] 메모리(340)는 컴퓨터에서 판독 가능한 기록 매체로서, RAM(random access memory), ROM(read only memory) 및 디스크 드라이브와 같은 비소멸성 대용량 기록장치(permanent mass storage device)를 포함할 수 있다. 또한, 메모리(340)에는 운영체제(341)와 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 루틴(342)을 위한 프로그램 코드가 저장될 수 있다. 이러한 소프트웨어 구성요소들은 드라이브 메커니즘(drive mechanism, 미도시)을 이용하여 메모리(340)와는 별도의 컴퓨터에서 판독 가능한 기록 매체로부터 로딩될 수 있다. 이러한 별도의 컴퓨터에서 판독 가능한 기록 매체는 플로피 드라이브, 디스크, 테이프, DVD/CD-ROM 드라이브, 메모리 카드 등의 컴퓨터에서 판독 가능한 기록 매체(미도시)를 포함할 수 있다. 다른 실시예에서 소프트웨어 구성요소들은 컴퓨터에서 판독 가능한 기록 매체가 아닌 네트워크 인터페이스(330)를 통해 메모리(340)에 로딩될 수도 있다.

[0058] 버스(320)는 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 시스템(300)의 구성요소들간의 통신 및 데이터 전송을 가능하게 할 수 있다. 버스(320)는 고속 시리얼 버스(high-speed serial bus), 병렬 버스(parallel bus), SAN(Storage Area Network) 및/또는 다른 적절한 통신 기술을 이용하여 구성될 수 있다.

[0059] 네트워크 인터페이스(330)는 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 시스템(300)을 컴퓨터 네트워크에 연결하기 위한 컴퓨터 하드웨어 구성요소일 수 있다. 네트워크 인터페이스(330)는 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 시스템(300)을 무선 또는 유선 커넥션을 통해 컴퓨터 네트워크에 연결시킬 수 있다.

[0060] 데이터베이스(350)는 고속 푸리에 변환 하드웨어 가속기 설계 및 검증을 위해 필요한 모든 정보를 저장 및 유지하는 역할을 할 수 있다. 도 3에서는 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 시스템(300)의 내부에 데이터베이스(350)를 구축하여 포함하는 것으로 도시하고 있으나, 이에 한정되는 것은 아니며 시스템 구현 방식이나 환경 등에 따라 생략될 수 있고 혹은 전체 또는 일부의 데이터베이스가 별개의 다른 시스템 상에 구축된 외부 데이터베이스로서 존재하는 것 또한 가능하다.

[0061] 프로세서(310)는 기본적인 산술, 로직 및 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 시스템(300)의 입출력 연산을 수행함으로써, 컴퓨터 프로그램의 명령을 처리하도록 구성될 수 있다. 명령은 메모리(340) 또는 네트워크 인터페이스(330)에 의해, 그리고 버스(320)를 통해 프로세서(310)로 제공될 수 있다. 프로세서(310)는 고속 푸리에 변환부(311) 및 시뮬레이션부(312)를 위한 프로그램 코드를 실행하도록 구성될 수 있다. 이러한 프로그램 코드는 메모리(340)와 같은 기록 장치에 저장될 수 있다.

[0062] 고속 푸리에 변환부(311) 및 시뮬레이션부(312)는 도 2의 단계들(210~220)을 수행하기 위해 구성될 수 있다.

[0063] 고속 푸리에 변환 하드웨어 가속기 설계 및 검증 시스템(300)은 고속 푸리에 변환부(311) 및 시뮬레이션부(312)를 포함할 수 있다.

[0064] 본 발명의 실시예에 따른 고속 푸리에 변환부(311)는 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환을 수행한다.

[0065] 본 발명의 실시예에 따른 고속 푸리에 변환부(311)는 상기 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환의 연산속도를 감소시키기 위해 분할 정복 알고리즘인 쿨리-튜키 알고리즘(Cooley-Tukey algorithm)을 적용하여 고속 푸리에 변환을 수행하고, 복수의 점(point)에 대한 고속 푸리에 변환을 수행하기 위한 복소수 입력으로 상기 복수의 점에 대한 실수부와 허수부를 각각 입력 받고, 출력도 상기 복수의 점에 대한 실수부와 허수부로 각각 출력된다.

[0066] 본 발명의 실시예에 따른 시뮬레이션부(312)는 상기 베릴로그를 이용하여 하드웨어로 구현된 고속 푸리에 변환에 대한 검증을 수행하기 위해 파이선으로 작성된 고속 푸리에 변환의 결과와 베릴로그 시뮬레이션 결과를 비교하는 벡터 매칭을 수행한다.

[0067] 본 발명의 실시예에 따른 시뮬레이션부(312)는 벡터 매칭을 수행하기 위해 시뮬레이션 할 베릴로그 파일 이름, 베릴로그 테스트벤치 파일 이름, 고속 푸리에 변환인지 또는 역 고속 푸리에 변환인지 여부, 시뮬레이션에 활용한 입력이 들어있는 파일 경로, 시뮬레이션 수행 횟수를 입력 받아 상기 입력된 정보에 따라 파이선으로 작성된 고속 푸리에 변환을 수행하여 파이선 시뮬레이션 결과를 저장하고, 상기 입력된 정보에 따라 베릴로그 파일과 베릴로그 테스트벤치 파일을 열어 베릴로그 시뮬레이션을 수행하고 결과를 저장한다.

[0069] 도 4는 본 발명의 일 실시예에 따른 베릴로그 구현 다이어그램을 나타내는 도면이다.

[0070] 본 발명의 실시예에 따르면, 베릴로그를 이용하여 16 점(point) 고속 푸리에 변환을 구현하였다. 고속 푸리에 변환의 입력은 복소수 이므로, 입력으로 점의 실수부와 허수부를 따로 각각 16bit 로 받는다. 그리고 소수점 연산도 필요하므로, 입력은 Q8.8 포맷(format)으로 입력한다. 출력도 마찬가지로 점의 실수부, 허수부 각각 16bit 로 출력이 되고, 이 때는 Q12.4 포맷으로 출력된다.

[0071] 본 발명의 실시예에 따른 고속 푸리에 변환부는 제어부(control unit)(410), 카운터(counter)(420), Wk LUT(430), 버터플라이(butterfly)(440) 및 레지스터(register)(450)를 포함한다.

[0072] 본 발명의 실시예에 따른 제어부(410)는 16개의 점을 입력 받고, 입력 받은 점들을 버터플라이(440)를 통해 연산한 후에, 레지스터(450)에 저장하고, 연산이 끝난 16개의 점을 출력하는 방식으로 동작한다. 따라서 본 발명의 실시예에 따른 제어부(410)는 점을 입력 받을 때, 연산을 할 때, 출력을 할 때 고속 푸리에 변환부 안의 모듈들의 동작을 다르게 제어해주는 모듈이다.

[0073] 본 발명의 실시예에 따른 카운터(420) 내부에는 두 개의 카운터가 존재한다. 하나는 Wk LUT(430)의 주소를 가리키고, 다른 하나는 레지스터(450)의 주소를 가리킨다. 본 발명의 실시예에 따른 카운터(420)는 제어부(410)의 제어에 따라 동작한다. 본 발명의 실시예에 따른 카운터(420)는 버터플라이(440) 연산의 입력으로 점들을 순서대로 입력 해준다.

[0074] 본 발명의 실시예에 따른 Wk LUT(430)의 Wk 값은 k 값에 따라 정해져 있는 상수이므로 따로 연산하지 않고 LUT(Look-Up Table)를 사용한다. 본 발명의 실시예에 따른 16 점(point) 고속 푸리에 변환에서는 W0~W7까지의 값을 사용하므로 8개의 값이 저장되어 있다.

[0076] 도 5는 본 발명의 일 실시예에 따른 버터플라이 유닛 모듈을 나타내는 도면이다.

[0077] 도 5를 참조하면, 본 발명의 실시예에 따른 버터플라이(도 4의 (440))는 N=8 일 때의 도 5의 반복으로 볼 수 있다.

[0078] 따라서 버터플라이 모듈에서는 점 두 개의 실수부, 허수부와, Wk LUT 로부터 값을 입력 받아 도 5의 버터플라이 연산을 수행하고 연산된 점 두 개의 실수부, 허수부를 출력한다.

[0080] 다시 도 4를 참조하면, 본 발명의 실시예에 따른 레지스터(450)는 16*32 레지스터로 점 16개의 실수부, 허수부를 저장하고 있으며, 앞 16bit 는 실수부, 뒤 16bit는 허수부가 저장되어 있다.

[0082] 다시 도 3을 참조하면, 본 발명의 실시예에 따른 시뮬레이션부(312)는 상기 파이선 시뮬레이션 결과 및 베릴로그 시뮬레이션 결과를 비교하여, 두 결과값이 상이할 경우, 해당 시뮬레이션에 대한 파이선 시뮬레이션 결과 및 베릴로그 시뮬레이션 결과와 총 시뮬레이션 횟수 중 상기 두 결과값이 상이한 경우의 횟수를 확인한다.

[0084] 도 6은 본 발명의 일 실시예에 따른 벡터 매칭 과정을 설명하기 위한 도면이다.

[0085] 본 발명의 실시예에 따르면, 베릴로그로 구현된 고속 푸리에 변환의 검증을 위해 파이선을 이용하여 구현된 고속 푸리에 변환과 결과값을 비교하여 벡터 매칭을 수행한다.

[0086] 벡터 매칭은 먼저 시뮬레이션 할 베릴로그 파일 이름, 베릴로그 테스트벤치 파일 이름, 고속 푸리에 변환인지 또는 역 고속 푸리에 변환인지 여부, 시뮬레이션에 활용한 입력이 들어있는 파일 경로, 시뮬레이션 수행 횟수를

입력 받아 수행된다(610).

[0087] 입력 받은 정보들을 이용하여 먼저 입력 파일로부터 받은 입력으로 파이선으로 작성된 고속 푸리에 변환을 입력 받은 횟수만큼 수행하여 파이선 시뮬레이션 결과를 저장한다(622).

[0088] 그리고 같은 입력으로 베릴로그 파일과 베릴로그 테스트벤치 파일을 열어 베릴로그 시뮬레이션을 하고 시뮬레이션 결과 값을 저장한다(621).

[0089] 파이선 시뮬레이션 및 베릴로그 시뮬레이션이 끝나고 저장된 결과값들을 하나씩 비교하며(630), 두 결과값이 상이할 경우, 사용자에게 몇 번째 입력에서 두 결과값이 상이한지, 상이한 틀린 파이선 시뮬레이션 결과값 및 베릴로그 시뮬레이션 결과값을 제공하고, 마지막으로 총 시뮬레이션 횟수 중 상기 두 결과값이 상이한 경우의 횟수를 확인(640)하여 벡터 매칭 결과 파일을 사용자에게 제공한다(650).

[0091] 도 7은 본 발명의 일 실시예에 따른 벡터 매칭 결과를 나타내는 예시도이다.

[0092] 도 7(a)는 벡터 매칭 결과 파일의 예시를 나타내고, 도 7(b)는 벡터 매칭 결과를 디스플레이하는 예시를 나타내는 도면이다.

[0093] 본 발명의 실시예에 따라 구현된 고속 푸리에 변환 베릴로그 코드를 4096개의 점을 이용하여 벡터 매칭을 수행하였다. 도 7에 도시된 바와 같이, 4096개의 점에 대한 시뮬레이션이 오류없이 진행됐음을 벡터 매칭 결과 파일을 통해 확인할 수 있다.

[0095] 이상에서 설명된 장치는 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성요소는, 예를 들어, 프로세서, 콘트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPGA(field programmable gate array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0096] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 가상 장치(virtual equipment), 컴퓨터 저장 매체 또는 장치에 구체화(embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

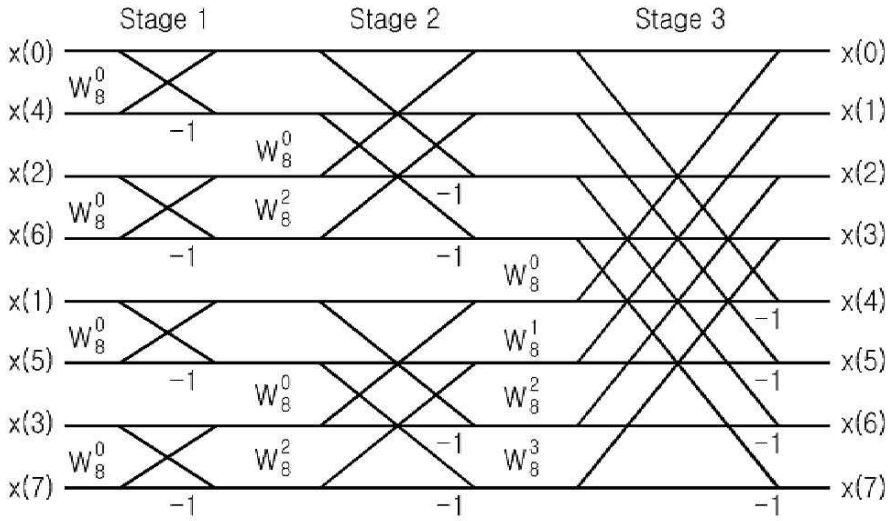
[0097] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다.

[0098] 이상과 같이 실시예들이 비록 한정된 실시예와 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

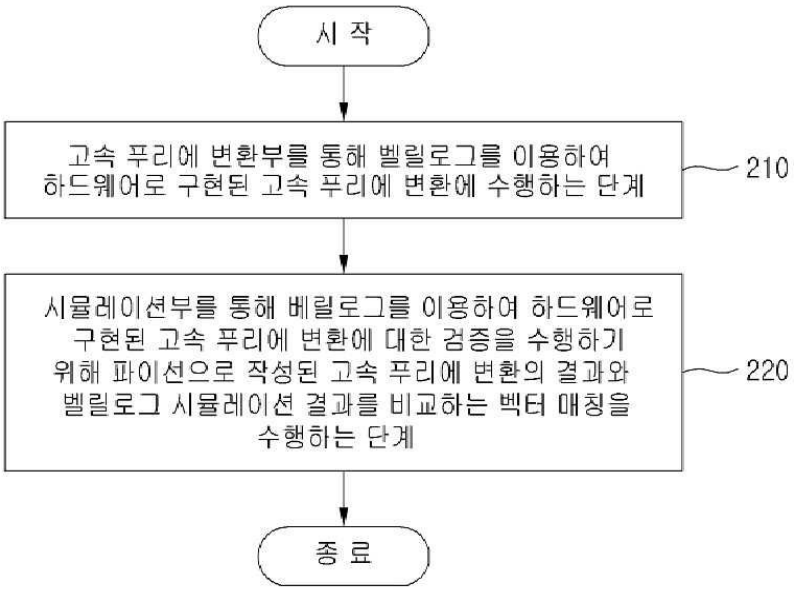
[0099] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

도면

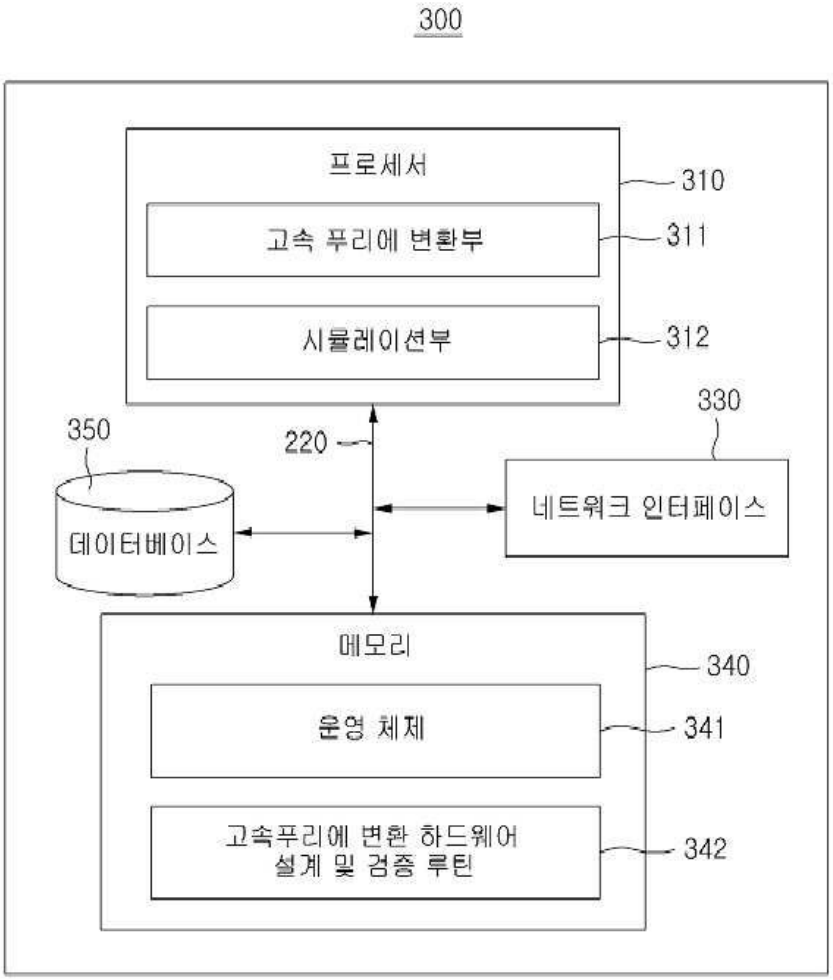
도면1



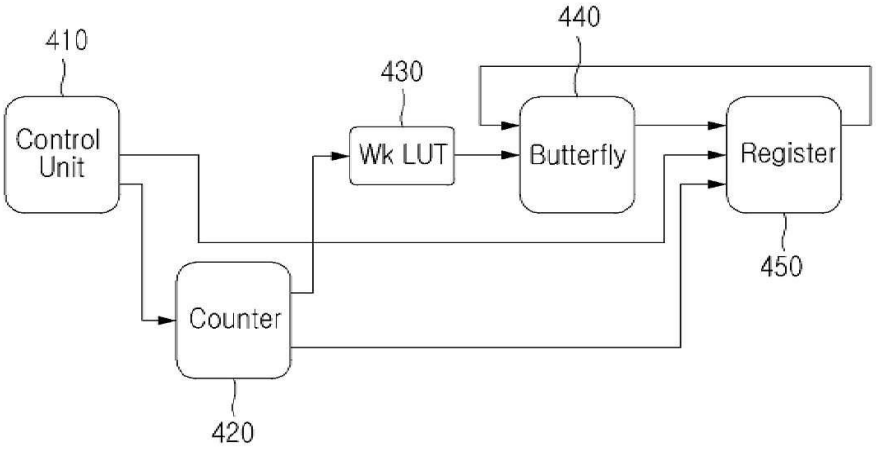
도면2



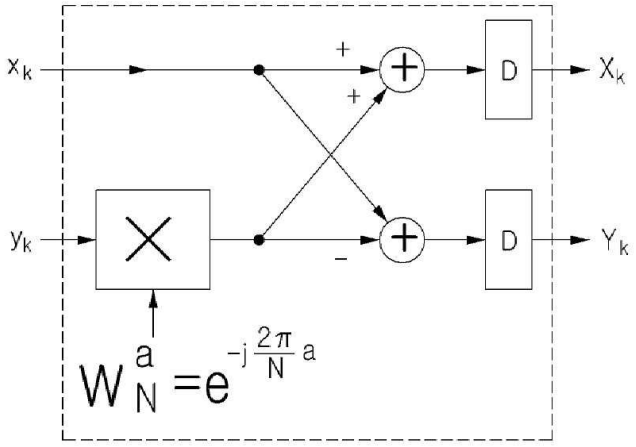
도면3


300
프로세서
310
고속 푸리에 변환부
311
시뮬레이션부
312
330
350
220
네트워크 인터페이스
데이터베이스
메모리
340
운영 체제
341
고속푸리에 변환 하드웨어
설계 및 검증 루틴
342

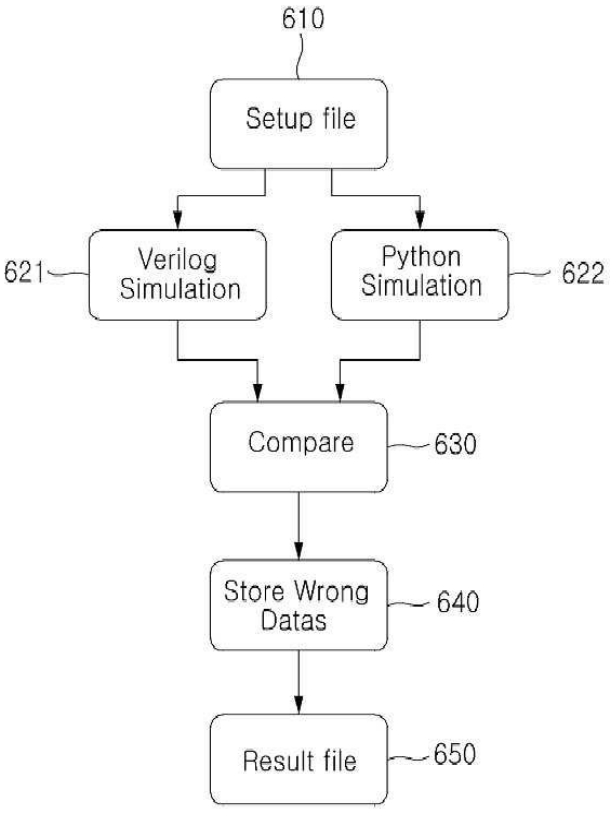
도면4


410
440
430
Control Unit
Wk LUT
Butterfly
Register
450
Counter
420

도면5


x_k
y_k
+
+
−
D
D
X_k
Y_k
$W_N^a = e^{-j\frac{2\pi}{N}a}$

도면6


610
Setup file
621
Verilog Simulation
Python Simulation
622
Compare
630
Store Wrong Datas
640
Result file
650

도면7a

```
----------------------------------------------------

 > Incorrect Input vec Number [4095]

Verilog input vec (REAL)
:[-7961, 8206, 15594, -6341, 1351, 17667, 149, -31450, -17043, 10758, -11274, -7829, 17979, 18215, -31240, -9265]

Verilog input vec (IMAG)
:[19981, -25306, 20533, -3870, -13435, -23304, 29281, -23199, -16239, -15666, 24653, 31662, -13347, 12889, -652, -8520]

Verilog output vec (REAL)
:[-2031, -2040, -2378, 2576, -1648, -420, -606, -6674, -2025, 5044, -1096, 1249, 4285, -339, -7007, 5138]

Python output vec (REAL)
:[-2030.25, -2037.771871621579, -2377.275768014725, 2576.982814699227, -1647.687500000001, -419.6112405756704, -605.2780
492319852742, 1250.4393228648646, 4284.812500000001, -338.1034575523273, -7006.096985824139, 5138.12694316471]

Verilog output vec (IMAG)
:[-285, 67, -4712, 1688, -12912, 8750, 7676, 2525, 6630, -1621, 4100, -1380, 804, 6013, 565, 2062]

Python output vec (IMAG)
:[-283.6875, 67.58431547763189, -4711.375659399553, 1688.0359584383389, -12911.625, 8751.06874679921, 7675.961036589192,
3, -1378.3875249182372, 804.75, 6014.059624617064, 565.9139634108078, 2063.5751211938914]
```

도면7b

