(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2018년02월12일
(11) 등록번호 10-1828697
(24) 등록일자 2018년02월06일

(51) 국제특허분류(Int. Cl.)
G06F 17/14 (2006.01)
(52) CPC특허분류
G06F 17/141 (2013.01)
(21) 출원번호 10-2016-0110252
(22) 출원일자 2016년08월29일
심사청구일자 2016년08월29일
(56) 선행기술조사문헌
KR1020090079265 A

(73) 특허권자
세종대학교산학협력단
서울특별시 광진구 능동로 209 (군자동, 세종대학교)
(72) 발명자
박천수
서울특별시 동대문구 장한로12길 94, 103동402호 (장안동, 상록파라브아파트)
(74) 대리인
송인호, 윤형근, 최영중, 최관락

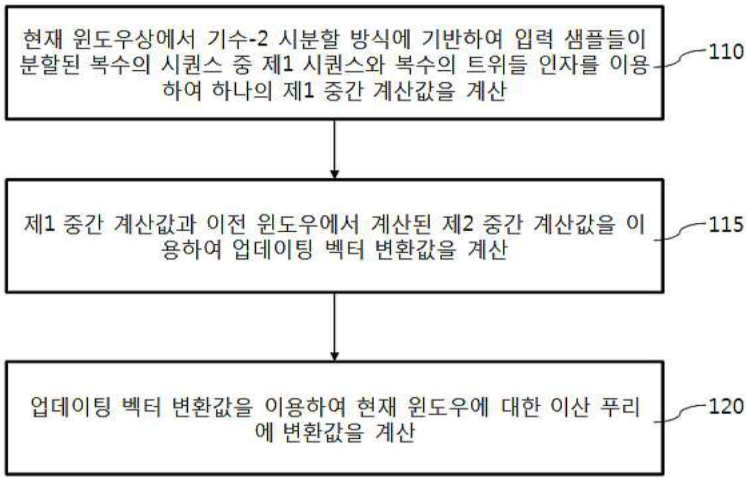
전체 청구항 수 : 총 16 항 심사관 : 이복현

(54) 발명의 명칭 **슬라이딩 이산 푸리에 변환 장치 및 그 방법**

(57) 요 약

슬라이딩 이산 푸리에 변환 장치 및 그 방법이 개시된다. 슬라이딩 이산 푸리에 변환 방법은, (a) 현재 윈도우상에서 기수-2 시분할 방식에 기반하여 입력 샘플들이 분할된 복수의 시퀀스 중 제1 시퀀스와 복수의 트위들 인자를 이용하여 하나의 제1 중간 계산값을 계산하는 단계; (b) 상기 제1 중간 계산값과 이전 윈도우에서 계산된 제2 중간 계산값을 이용하여 업데이팅 벡터 변환값을 계산하는 단계; 및 (c) 상기 업데이팅 벡터 변환값을 이용하여 상기 현재 윈도우에 대한 이산 푸리에 변환값을 계산하는 단계를 포함한다.

대 표 도 - 도1



이 발명을 지원한 국가연구개발사업
과제고유번호 C0349282
부처명 중소기업청
연구관리전문기관 한국산학연협회
연구사업명 산학연협력기술개발
연구과제명 시네마 게임을 위한 스마트폰 전용 실사영상 처리
기 여 율 1/1
주관기관 세종대학교산학협력단
연구기간 2015.11.12 ~ 2016.11.11

명 세 서

청구범위

청구항 1

슬라이딩 이산 푸리에 변환 방법에 있어서,

(a) 현재 윈도우상에서 기수-2 시분할 방식에 기반하여 입력 샘플들이 분할된 복수의 시퀀스 중 제1 시퀀스와 복수의 트위들 인자를 이용하여 하나의 제1 중간 계산값을 계산하는 단계;

(b) 상기 제1 중간 계산값과 이전 윈도우에서 계산된 제2 중간 계산값을 이용하여 업데이팅 벡터 변환값을 계산하는 단계; 및

(c) 상기 업데이팅 벡터 변환값을 이용하여 상기 현재 윈도우에 대한 이산 푸리에 변환값을 계산하는 단계를 포함하는 슬라이딩 이산 푸리에 변환 방법.

청구항 2

제1 항에 있어서,

상기 (a) 단계에서 분할된 시퀀스 중 제1 시퀀스는 상기 현재 윈도우에 신규 입력된 샘플을 포함하는 것을 특징으로 하는 슬라이딩 이산 푸리에 변환 방법.

청구항 3

제1 항에 있어서,

상기 기수-2 시분할 방식에 기반하여 상기 입력 샘플들은 원 포인트 시퀀스로 분할될 때까지 분할되되,

상기 (a) 단계 및 상기 (b) 단계는 상기 원 포인트 시퀀스에 상응하는 분할 계층부터 최상위 계층까지 재귀적으로 반복 수행되는 것을 특징으로 하는 슬라이딩 이산 푸리에 변환 방법.

청구항 4

제3 항에 있어서,

상기 계산된 업데이팅 벡터 변환값이 최상위 계층에서 계산된 업데이팅 벡터 변환값이 아닌 경우, 상기 계산된 업데이팅 벡터 변환값은 상기 제1 시퀀스의 상위 계층의 분할된 시퀀스 중 어느 하나로 입력되는 것을 특징으로 하는 슬라이딩 이산 푸리에 변환 방법.

청구항 5

제4 항에 있어서,

상기 계산된 업데이팅 벡터 변환값은 다음 윈도우로 복사되되,

상기 상위 계층의 분할된 시퀀스 중 어느 하나인 제2 시퀀스로 복사되는 것을 특징으로 하는 슬라이딩 이산 푸리에 변환 방법.

청구항 6

제5 항에 있어서,

상기 상위 계층의 분할된 시퀀스 중 하나는 기수-2 시분할 방식에 기반하여 입력 샘플이 분할된 시퀀스 중 홀수 파트 시퀀스이고, 상기 제2 시퀀스는 짝수 파트 시퀀스인 것을 특징으로 하는 슬라이딩 이산 푸리에 변환 방법.

청구항 7

제1 항에 있어서,

상기 현재 윈도우는 상기 이전 윈도우와 복수의 입력 샘플이 중첩되며, 하나의 신규 입력 샘플만 갱신되는 것을 특징으로 하는 슬라이딩 이산 푸리에 변환 방법.

청구항 8

제1 항에 있어서,

상기 (a) 단계에서, 상기 제1 시퀀스가 신규 입력된 샘플과 동일하면, 상기 신규 입력된 샘플과 상기 복수의 트위들 인자를 이용하여 4개의 제1 중간 계산값을 계산하되,

상기 4개의 제1 중간 계산값 중 하나만 상기 복수의 트위들 인자 중 하나와의 곱셈을 통해 계산되며, 나머지 제1 중간 계산값은 상기 신규 입력된 샘플과 상기 계산된 하나의 제1 중간 계산값을 이용하여 계산되는 것을 특징으로 하는 슬라이딩 이산 푸리에 변환 방법.

청구항 9

제1 항에 있어서,

상기 업데이팅 벡터 변환값은 하기 수학식을 이용하여 계산되는 것을 특징으로 하는 슬라이딩 이산 푸리에 변환 방법.

$$D_n^{s_l}(k) = D_{n-2^l}^{[s_l,o]}(k) + W_M^{-2^l k} D_n^{[s_l,o]}$$

여기서, n은 시간 인덱스를 나타내고, k는 $0 \leq k < M$ 범위를 가지는 주파수 도메인의 인덱스를 나타내며, 상기 $W_M = e^{j2\pi/M}$ 는 트위들 인자를 나타내고, $D_{n-2^l}^{[s_l,o]}(k)$ 는 이전 윈도우 포지션($n-2^l$)에서 계산된 짝수 파트의 업데이팅 벡터 변환값을 나타내며, 상기 s_l 는 길이가 l인 순차적인 분해 오퍼레이션을 나타내고, $[\cdot,\cdot]$ 는 접합(concatenation)을 나타내고, $D_n^{[s_l,o]}$ 는 s_l 에 대한 홀수 파트의 업데이팅 벡터 변환값을 나타내며, 상기 M은 윈도우 사이즈를 나타냄.

청구항 10

제1 항에 있어서,

상기 (c) 단계는, 상기 업데이팅 벡터 변환값이 최상위 계층에 상응하는 값인 경우 수행되는 것을 특징으로 하는 슬라이딩 이산 푸리에 변환 방법.

청구항 11

슬라이딩 이산 푸리에 변환 장치에 있어서,

현재 윈도우상에서 기수-2 시분할 방식에 기반하여 입력 샘플들이 분할된 복수의 시퀀스 중 제1 시퀀스와 복수의 트위들 인자를 이용하여 하나의 제1 중간 계산값을 계산하는 제1 계산부;

상기 제1 중간 계산값과 이전 윈도우에서 계산된 제2 중간 계산값을 이용하여 업데이팅 벡터 변환값을 계산하는 UVT 계산부; 및

상기 업데이팅 벡터 변환값을 이용하여 상기 현재 윈도우에 대한 이산 푸리에 변환값을 계산하는 DFT 계산부를 포함하는 슬라이딩 이산 푸리에 변환 장치.

청구항 12

제11 항에 있어서,

상기 제1 시퀀스는 상기 현재 윈도우에 신규 입력된 샘플을 포함하는 것을 특징으로 하는 슬라이딩 이산 푸리에 변환 장치.

청구항 13

제11 항에 있어서,

기수-2 시분할 방식에 기반하여 상기 입력 샘플들을 원 포인트 시퀀스로 분할될때까지 분할하는 분할부; 및

상기 원 포인트 시퀀스에 상응하는 분할 계층부터 최상위 계층까지 재귀적으로 반복 수행되도록 상기 제1 계산부와 상기 UVT 계산부를 제어하는 프로세서를 더 포함하는 슬라이딩 이산 푸리에 변환 장치.

청구항 14

제13 항에 있어서,

상기 프로세서는,

상기 계산된 업데이팅 벡터 변환값이 최상위 계층에서 계산된 업데이팅 벡터 변환값이 아닌 경우, 상기 계산된 업데이팅 벡터 변환값은 상기 제1 시퀀스의 상위 계층의 분할된 시퀀스 중 어느 하나로 입력되도록 제어하는 것을 특징으로 하는 슬라이딩 이산 푸리에 변환 장치.

청구항 15

제14 항에 있어서,

메모리를 더 포함하되,

상기 프로세서는, 상기 계산된 업데이팅 벡터 변환값을 다음 윈도우에서 이용될 수 있도록 상기 메모리에 복사하되,

상기 계산된 업데이팅 벡터 변환값은 상위 계층의 분할된 시퀀스 중 어느 하나인 제2 시퀀스로 복사되는 것을 특징으로 하는 슬라이딩 이산 푸리에 변환 장치.

청구항 16

제11 항에 있어서,

상기 제1 계산부는, 상기 제1 시퀀스가 신규 입력된 샘플과 동일하면, 상기 신규 입력된 샘플과 상기 복수의 트위들 인자를 이용하여 4개의 제1 중간 계산값을 계산하되,

상기 4개의 제1 중간 계산값 중 하나만 상기 복수의 트위들 인자 중 하나와의 곱셈을 통해 계산되며, 나머지 제1 중간 계산값은 상기 신규 입력된 샘플과 상기 계산된 하나의 제1 중간 계산값을 이용하여 계산되는 것을 특징으로 하는 슬라이딩 이산 푸리에 변환 장치.

발명의 설명

기 술 분 야

[0001] 본 발명은 슬라이딩 이산 푸리에 변환 장치 및 그 방법에 관한 것이다.

배 경 기 술

[0003] DFT(Discrete Fourier transform), DHT(Discrete Hartley Transform),

[0004] WHT(Walsh Hadamard Transform)와 같은 이산 직교 변환들은 디지털 신호 처리, 필터링 및 통신의 분야에서 중요한 역할을 수행한다. 최근, 슬라이딩 변환 프로세스에 관심이 증가하고 있다. 이 슬라이딩 변환 프로세스는 변환 윈도우가 한 번에 하나의 샘플만큼 시프트되고, 변환 프로세스가 반복되는 것이다.

[0005] Jacobsen와 Lyons가 제안한 슬라이딩 DFT(SDFT) 알고리즘([1] E. Jacobsen and R. Lyons, "An update to the sliding DFT," IEEE Signal Processing Mag., vol. 21, no. 1, pp. 110??111, Jan. 2004.)은 반복 스킴을 이용하여 DFT값을 계산하는 것이다. 이 SDFT알고리즘은 DFT의 연산량을 대폭 감소시킬 수 있으나, 잠재된 불안정성을 감수해야 한다. 그래서, SDFT의 안정성을 보장하기 위하여 다음과 같은 알고리즘들이 제안된 바 있다.

[0006] [2] E. Jacobsen and R. Lyons, "The sliding DFT," IEEE Signal Processing Mag., vol. 20, no. 2, pp. 74-80, Mar. 2003.

[0007] [3] K. Duda, "Accurate, guaranteed stable, sliding discrete Fourier transform," IEEE Signal Processing Mag., vol. 27, no. 6, pp. 124-127, Nov.2010.

[0008] [4] C. S. Park and S. J. Ko, "The hopping discrete Fourier transform," IEEE Signal Processing Mag., vol. 31, no. 2, pp. 135-139, Mar. 2014.

[0009] 이러한 알고리즘들은 계산 복잡도를 높이거나 계산 정확도를 낮추는 과정을 통해 슬라이딩 변환 프로세스의 안정성을 보장한다.

발명의 내용

해결하려는 과제

[0011] 본 발명은 연산량을 최소화한 슬라이딩 이산 푸리에 변환 장치 및 그 방법을 제공하기 위한 것이다.

[0012] 또한, 본 발명은 이전 윈도우에서 계산된 중간 계산값을 그대로 재이용하여 현재 윈도우의 이산 푸리에 변환 계산을 위해 이용함으로써 연산 복잡도를 줄일 수 있으며 연산 시간을 최소화할 수 있는 슬라이딩 이산 푸리에 변환 장치 및 그 방법을 제공하기 위한 것이다.

과제의 해결 수단

[0014] 본 발명의 일 측면에 따르면, 연산량을 최소화한 슬라이딩 이산 푸리에 변환 방법이 제공된다.

[0015] 본 발명의 일 실시예에 따르면, 슬라이딩 이산 푸리에 변환 방법에 있어서, (a) 현재 윈도우상에서 기수-2 시분할 방식에 기반하여 입력 샘플들이 분할된 복수의 시퀀스 중 제1 시퀀스와 복수의 트위들 인자를 이용하여 하나의 제1 중간 계산값을 계산하는 단계; (b) 상기 제1 중간 계산값과 이전 윈도우에서 계산된 제2 중간 계산값을 이용하여 업데이팅 벡터 변환값을 계산하는 단계; 및 (c) 상기 업데이팅 벡터 변환값을 이용하여 상기 현재 윈도우에 대한 이산 푸리에 변환값을 계산하는 단계를 포함하는 슬라이딩 이산 푸리에 변환 방법이 제공될 수 있다.

[0016] 상기 (a) 단계에서 분할된 시퀀스 중 제1 시퀀스는 상기 현재 윈도우에 신규 입력된 샘플을 포함할 수 있다.

[0017] 상기 기수-2 시분할 방식에 기반하여 상기 입력 샘플들은 원 포인트 시퀀스로 분할될 때까지 분할되되, 상기 (a) 단계 및 상기 (b) 단계는 상기 원 포인트 시퀀스에 상응하는 분할 계층부터 최상위 계층까지 재귀적으로 반복 수행될 수 있다.

[0018] 상기 계산된 업데이팅 벡터 변환값이 최상위 계층에서 계산된 업데이팅 벡터 변환값이 아닌 경우, 상기 제1 시퀀스의 상위 계층의 분할된 시퀀스 중 하나로 입력될 수 있다.

[0019] 상기 계산된 업데이팅 벡터 변환값은 다음 윈도우로 복사되고, 상기 상위 계층의 분할된 시퀀스 중 제2 시퀀스에 상응하여 복사된다.

[0020] 상기 제1 시퀀스는 기수-2 시분할 방식에 기반하여 상기 상위 계층에서 분할된 시퀀스 중 홀수 파트 시퀀스이고, 상기 제2 시퀀스는 짝수 파트 시퀀스이다.

[0021] 상기 현재 윈도우는 상기 이전 윈도우와 복수의 입력 샘플이 중첩되며, 하나의 신규 입력 샘플만 갱신될 수 있다.

[0022] 상기 (a) 단계에서, 상기 제1 시퀀스가 신규 입력된 샘플과 동일하면, 상기 신규 입력된 샘플과 상기 복수의 트위들 인자를 이용하여 4개의 제1 중간 계산값을 계산하되, 상기 4개의 제1 중간 계산값 중 하나만 상기 복수의 트위들 인자 중 하나와의 곱셈을 통해 계산되며, 나머지 제1 중간 계산값은 상기 신규 입력된 샘플과 상기 계산된 하나의 제1 중간 계산값을 이용하여 계산될 수 있다.

[0023] 상기 업데이팅 벡터 변환값은 하기 수학식을 이용하여 계산되되,

$$D_n^{s_l}(k) = D_{n-2^l}^{[s_l,o]}(k) + W_M^{-2^l k} D_n^{[s_l,o]}$$

[0024] 여기서, n은 시간 인덱스를 나타내고, k는 $0 \le k < M$ 범위를 가지는 주파수 도메인의 인덱스를 나타내며, 상기 $W_M = e^{j2\pi/M}$ 는 트위들 인자를 나타내고, $D_{n-2^l}^{[s_l,o]}(k)$ 는 이전 윈도우 포지션($n-2^l$)에서 계산된 짝수 파트의 업데이팅 벡터 변환값을 나타내며, 상기 s_l 는 길이가 1인 순차적인 분해 오퍼레이션을 나타내고, $[\cdot,\cdot]$ 는 접합(concatenation)을 나타내고, $D_n^{[s_l,o]}$ 는 s_l 에 대한 홀수 파트의 업데이팅 벡터 변환값을 나타내며, 상기 M은 윈도우 사이즈를 나타낸다.

[0025] 상기 (c) 단계는, 상기 업데이팅 벡터 변환값이 최상위 계층에 상응하는 값인 경우 수행될 수 있다.

[0027] 본 발명의 다른 측면에 따르면, 연산량을 최소화한 슬라이딩 이산 푸리에 변환 장치가 제공된다.

[0028] 본 발명의 일 실시예에 따르면, 슬라이딩 이산 푸리에 변환 장치에 있어서, 현재 윈도우상에서 기수-2 시분할 방식에 기반하여 입력 샘플들이 분할된 복수의 시퀀스 중 제1 시퀀스와 복수의 트위들 인자를 이용하여 하나의 제1 중간 계산값을 계산하는 제1 계산부; 상기 제1 중간 계산값과 이전 윈도우에서 계산된 제2 중간 계산값을 이용하여 업데이팅 벡터 변환값을 계산하는 UVT 계산부; 및 상기 업데이팅 벡터 변환값을 이용하여 상기 현재 윈도우에 대한 이산 푸리에 변환값을 계산하는 DFT 계산부를 포함하는 슬라이딩 이산 푸리에 변환 장치가 제공될 수 있다.

[0029] 상기 제1 시퀀스는 상기 현재 윈도우에 신규 입력된 샘플을 포함한다.

[0030] 기수-2 시분할 방식에 기반하여 상기 입력 샘플들을 원 포인트 시퀀스로 분할될때까지 분할하는 분할부; 상기 원 포인트 시퀀스에 상응하는 분할 계층부터 최상위 계층까지 재귀적으로 반복 수행되도록 상기 제1 계산부와 상기 UVT 계산부를 제어하는 프로세서를 더 포함할 수 있다.

[0031] 상기 프로세서는, 상기 계산된 업데이팅 벡터 변환값이 최상위 계층에서 계산된 업데이팅 벡터 변환값이 아닌 경우, 상기 제1 시퀀스의 상위 계층의 분할된 시퀀스 중 하나로 입력되도록 제어할 수 있다.

[0032] 메모리를 더 포함하되, 상기 프로세서는, 상기 계산된 업데이팅 벡터 변환값을 다음 윈도우에서 이용될 수 있도

록 상기 메모리에 저장하되, 상기 계산된 업데이팅 벡터 변환값은 상위 계층이 분할된 시퀀스 중 제2 시퀀스에 상응하도록 저장될 수 있다.

[0033] 상기 제1 계산부는, 상기 제1 시퀀스가 신규 입력된 샘플과 동일하면, 상기 신규 입력된 샘플과 상기 복수의 트위들 인자를 이용하여 4개의 제1 중간 계산값을 계산하되, 상기 4개의 제1 중간 계산값 중 하나만 상기 복수의 트위들 인자 중 하나와의 곱셈을 통해 계산되며, 나머지 제1 중간 계산값은 상기 신규 입력된 샘플과 상기 계산된 하나의 제1 중간 계산값을 이용하여 계산될 수 있다.

발명의 효과

[0035] 본 발명의 일 실시예에 따른 슬라이딩 이산 푸리에 변환 장치 및 그 방법을 제공함으로써, 이전 윈도우에서 계산된 중간 계산값을 그대로 재이용하여 현재 윈도우의 이산 푸리에 변환 계산을 위해 이용함으로써 연산 복잡도를 줄일 수 있으며 연산 시간을 최소화할 수 있는 이점이 있다.

도면의 간단한 설명

[0037] 도 1은 본 발명의 일 실시예에 따른 슬라이딩 이산 푸리에 변환 방법을 나타낸 순서도.

도 2는 본 발명의 일 실시예에 따른 분해 과정을 설명하기 위해 도시한 도면.

도 3은 본 발명의 일 실시예에 따른 슬라이딩 이산 푸리에 변환 방법을 설명하기 위해 도시한 버터플라이 구조를 나타낸 도면.

도 4는 본 발명의 일 실시예에 따른 슬라이딩 이산 푸리에 변환 장치의 구성을 개략적으로 도시한 블록도.

발명을 실시하기 위한 구체적인 내용

[0038] 본 명세서에서 사용되는 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 명세서에서, "구성된다" 또는 "포함한다" 등의 용어는 명세서상에 기재된 여러 구성 요소들, 또는 여러 단계들을 반드시 모두 포함하는 것으로 해석되지 않아야 하며, 그 중 일부 구성 요소들 또는 일부 단계들은 포함되지 않을 수도 있고, 또는 추가적인 구성 요소 또는 단계들을 더 포함할 수 있는 것으로 해석되어야 한다. 또한, 명세서에 기재된 "...부", "모듈" 등의 용어는 적어도 하나의 기능이나 동작을 처리하는 단위를 의미하며, 이는 하드웨어 또는 소프트웨어로 구현되거나 하드웨어와 소프트웨어의 결합으로 구현될 수 있다.

[0039] 이하, 첨부된 도면들을 참조하여 본 발명의 실시예를 상세히 설명한다.

[0041] 도 1은 본 발명의 일 실시예에 따른 슬라이딩 이산 푸리에 변환 방법을 나타낸 순서도이고, 도 2는 본 발명의 일 실시예에 따른 분해 과정을 설명하기 위해 도시한 도면이며, 도 3은 본 발명의 일 실시예에 따른 슬라이딩 이산 푸리에 변환 방법을 설명하기 위해 도시한 버터플라이 구조를 나타낸 도면이다.

[0042] 단계 110에서 슬라이딩 이산 푸리에 변환 장치(100)는 현재 윈도우상에서 기수-2 시분할 방식에 기반하여 입력 샘플들이 분할된 복수의 시퀀스 중 제1 시퀀스와 복수의 트위들 인자를 이용하여 하나의 제1 중간 계산값을 계산한다. 여기서, 제1 시퀀스는 현재 윈도우에 신규 입력된 샘플이 포함된 시퀀스일 수 있다.

[0043] 단계 115에서 슬라이딩 이산 푸리에 변환 장치(100)는 계산된 제1 중간 계산값과 이전 윈도우에서 계산된 제2 중간 계산값을 이용하여 업데이팅 벡터 변환값을 계산한다.

[0044] 여기서, 제2 중간 계산값은 기수-2 시분할 방식에 따라 입력 샘플들이 분할된 복수의 시퀀스 중 상기 제1 시퀀스와 페어(pair)를 형성하는 제2 시퀀스에 상응하는 중간 계산값일 수 있다. 이는 하기의 설명에 의해 보다 명확하게 이해될 것이다.

[0045] 하기에서 보다 명확하게 설명되겠지만, 단계 110 내지 단계 115는 재귀적으로 반복수행될 수 있다. 이때, 계산된 업데이팅 벡터 변환값은 단계 110의 제1 시퀀스의 상위 계층의 분할된 시퀀스 중 하나로 입력되어 중간 계산값 계산에 이용될 수 있다.

[0046] 결과적으로 단계 110 및 단계 120은 기수-2 시분할 방식에 기반하여 분할된 시퀀스의 크기가 입력 샘플들의 크기와 동일(즉, 최상위 계층)일 때까지 반복하여 수행될 수 있다.

[0047] 또한, 계산된 업데이팅 벡터 변환값과 제1 중간 계산값 중 적어도 하나는 다음 윈도우에서 그대로 이용되기 위해 복사될 수 있다. 계산된 업데이팅 벡터 변환값 및 제1 중간 계산값 중 적어도 하나가 다음 윈도우로 복사됨

에 있어, 상위 계층의 분할된 시퀀스 중 제2 시퀀스에 상응하도록 복사될 수 있다. 이때, 제2 시퀀스는 계산된 업데이팅 벡터 변환값에 상응하는 시퀀스의 짝수 파트 시퀀스일 수 있다.

[0048] 단계 120에서 슬라이딩 이산 푸리에 변환 장치(100)는 업데이팅 벡터 변환값을 이용하여 현재 윈도우에 대한 이산 푸리에 변환값을 계산한다. 이하, 본 발명의 일 실시예에 따른 슬라이딩 이산 푸리에 변환 방법을 빠르게 수행하기 위한 방법에 대해 상세히 설명하기로 한다.

[0049] 본 발명의 일 실시예에 따른 슬라이딩 이산 푸리에 변환 장치(100)는 고정된 사이즈의 윈도우상에서 반복적으로 슬라이딩 변환(sliding transform)이 수행된다. 이때, 윈도우에 신규 샘플들이 갱신됨에 따라 가장 오래전에 입력된 샘플들은 버려지게 된다. 고정된 사이즈의 윈도우를 중첩함에 따라 나뉘는 복소수 입력 신호를 $x(n)$라 나타내기로 한다. 여기서, $n = 0, 1, 2, \dots$ 이다.

[0050] 또한, 시간 인덱스 n에서 M-포인트(point) DFT(discrete furrier transform)의 k번째 계산값(bin)을 $X_n(k)$으로 표시하기로 한다. 여기서, $k = 0, 1, \dots, M-1$ 이다.

[0051] $X_n(k)$을 수학식으로 나타내면 수학식 1과 같다.

수학식 1

[0052]
$$X_n(k) = \sum_{m=0}^{M-1} x(\hat{n}+m) W_M^{-km}$$

[0053] 여기서, $\hat{n} = n - M + 1$ 이고, $W_M = e^{j2\pi/M}$ 이다.

[0054] 원형 시프트 속성에 따라, 수학식 1에서 연속적인 DFT 출력값간의 관계를 파생함에 따라 수학식 1은 수학식 2와 같이 나타낼 수 있다.

수학식 2

[0055]
$$\begin{aligned} X_n(k) &= \sum_{m=0}^{M-1} x(\hat{n}+m) W_M^{-km} \\ &= \sum_{m=0}^{M-1} x(\hat{n}+m-1) W_M^{-k(m-1)} \\ &\quad + x(\hat{n}+M-1) W_M^{-k(M-1)} - x(\hat{n}-1) W_M^{k} \\ &= W_M^k \sum_{m=0}^{M-1} x(\hat{n}+m-1) W_M^{-km} \\ &\quad + x(\hat{n}+M-1) W_M^{k} - x(\hat{n}-1) W_M^{k} \\ &= W_M^k (X_{n-1}(k) + x(n) - x(n-M)) \end{aligned}$$

[0056] 여기서, 복소 트위들 인자의 주기적 속성이 이용된다($W_M^{k+M} = W_M^k$).

[0057] 슬라이딩 이산 푸리에 변환(이하에서는 SDFT(sliding discrete furrier transform)이라 칭하기로 함)은 시프트된 윈도우의 DFT 계산 결과값을 획득하기 위해 이전 윈도우의 DFT 계산 결과에 복소 트위들 인자(W_M^k)를 곱해야 한다.

[0058] 결과적으로 SDFT는 시프트된 윈도우의 DFT를 계산하기 위해 한번의 복소수 곱셈과 두번의 복소수 가산이 요구된다.

[0059] SDFT는 Z-도메인내의 단위원에 놓이기 때문에 안정성이 보장되지 않는 전달 함수를 가진다.

[0060] 실제, 수학식 2의 복소 트위들 인자는 한정된 정밀도를 가지는 부동 소수점 형태로 표현된다. 복소 트위들 인자의 수치적 반올림(rounding)은 극을 단위원 밖으로 이동시키고 이로 인해 결과적으로 시스템 불안정을 야기시킨다.

[0061] 또한, 재귀적 공식내에 포함되는 복소 트위들 인자는 슬라이딩 트랜스폼 과정의 계산적 부하를 증가시킨다.

[0062] 따라서, 피드백 루프로부터 부정확한 복소 트위들 인자를 제외시킴으로써 계산의 복잡도를 줄인 안정성을 보장하는 슬라이딩 DFT 필터가 설계될 수 있다.

[0063] 연속적인 윈도우의 계산값들간의 관계를 확장시킴에 따라 L-홉(hop_ 거리를 가지는 DFT 계산값들간의 일반적인 공식이 파생될 수 있다.

[0064] $d(n) = x(n) - x(n-M)$ 이라 가정하기로 한다. 수학식 2에서 $X_n(k)$ 을 $X_{n-1}(k)$ 로 L번 대체시킴으로써 수학식 3이 획득될 수 있다.

수학식 3

[0065]
$$
\begin{aligned}
X_n(k) &= W_M^k (X_{n-1}(k) + d(n)) \\
&= W_M^{2k} (X_{n-2}(k) + d(n-1) + W_M^{-k} d(n)) \\
&\quad \vdots \\
&= W_M^{Lk} (X_{n-L}(k) + d(n-L+1) \\
&\quad + W_M^{-k} d(n-L+2) + \cdots + W_M^{-(L-1)k} d(n))
\end{aligned}
$$

[0066] 여기서, L-포인트 업데이트 벡터 변환(UVT)의 k번째 계산값을 $D_n(k)$ 로 나타낼 수 있다. 이를 수학식으로 나타내면 수학식 4와 같다.

수학식 4

[0067]
$$
D_n(k) = \sum_{m=0}^{L-1} d(\tilde{n}+m) W_M^{-mk}
$$

[0068] 여기서, $\tilde{n} = n - L + 1$ 이고, $0 \le k < M$ 이다.

[0069] 수학식 4를 이용하여, 수학식 3은 수학식 5와 같이 간단하게 나타낼 수 있다.

수학식 5

[0070] $$X_n(k) = W_M^{Lk}(X_{n-L}(k) + D_n(k))$$

[0071] 시간 인덱스 n에서 DFT 출력값은 $D_n(k)$ 을 이용함으로써 시간 인덱스(n-L)의 DFT 출력값으로부터 직접 계산될 수 있다.

[0072] 이하에서는 수학식 4의 UVT의 효율적인 구현 방법에 대해 보다 상세히 설명하기로 한다.

[0073] 트위들 인자(W_M^k)는 수학식 5의 반복되는 계산에서 지연된 출력값과 곱해지며, 트위들 인자의 값은 시간 홉(L)에 종속하여 가변된다.

[0074] $L = M/4$ 라고 가정하기로 한다.

[0075] Euler's 공식으로부터 트위들 인자를 수학식 6과 같이 나타낼 수 있다.

수학식 6

[0076] $$W_M^{Mk/4} = (\cos(\pi/2) + j\sin(\pi/2))^k = j^k$$

[0077] 수학식 6을 이용하여 수학식 5는 수학식 7과 같이 더 단순하게 정리될 수 있다.

수학식 7

[0078] $$X_n(k) = j^k(X_{n-M/4}(k) + D_n(k))$$

[0079] 그리고, j의 주기적 속성에 의해 수학식 8이 획득될 수 있다.

수학식 8

$$\begin{bmatrix} X_n(4i) \\ X_n(4i+1) \\ X_n(4i+2) \\ X_n(4i+3) \end{bmatrix} = \begin{bmatrix} X_{n-M/4}(4i) + D_n(4i) \\ j(X_{n-M/4}(4i+1) + D_n(4i+1)) \\ -(X_{n-M/4}(4i+2) + D_n(4i+2)) \\ -j(X_{n-M/4}(4i+3) + D_n(4i+3)) \end{bmatrix}$$

[0080]

[0081] 여기서, $i = 0, 1, \ldots, M/4-1$ 이다. $(n - M/4)$ 시간 인덱스에서, 이전 윈도우의 DFT 계산값이 주어지는 경우, n 시간 인덱스에서의 DFT 계산값들은 트위들 인자의 곱셈 연산 없이 직접 계산될 수 있다. 예를 들어, $k = 4i$ 일 때, $X_n(4i)$ 와 $X_{n-M/4}(4i)$ 사이의 관계는 수학식 9와 같이 나타낼 수 있다.

수학식 9

$$\begin{aligned} &\mathrm{Re}(X_n(4i)) + j\mathrm{Im}(X_n(4i)) \\ &\quad = \mathrm{Re}(X_{n-M/4}(4i)) + \mathrm{Re}(D_n(4i)) \\ &\qquad + j[\mathrm{Im}(X_{n-M/4}(4i)) + \mathrm{Im}(D_n(4i))] \end{aligned}$$

[0082]

[0083] 여기서, $\mathrm{Re}(\cdot)$ 과 $\mathrm{Im}(\cdot)$ 는 복소수의 실수 부분과 허수 부분을 각각 나타낸다.

[0084] $X_n(4i)$ 의 실수 부분과 허수 부분을 각각 정리하면 수학식 10과 같다.

수학식 10

$$\begin{cases} \mathrm{Re}(X_n(4i)) = \mathrm{Re}(X_{n-M/4}(4i)) + \mathrm{Re}(D_n(4i)) \\ \mathrm{Im}(X_n(4i)) = \mathrm{Im}(X_{n-M/4}(4i)) + \mathrm{Im}(D_n(4i)) \end{cases}$$

[0085]

[0086] $k = 4i+1, 4i+2, 4i+3$ 도 $k = 4i$ 와 마찬가지로 유사하게 파생될 수 있다.

[0087] 표 1은 $X_n(k)$ 과 $X_{n-M/4}(k)$ 의 실수 부분과 허수 부분 사이의 재귀적 관계를 정리한 표이다.

표 1

[0088]

q	$\mathrm{Re}(X_n(4i+q))$	$\mathrm{Im}(X_n(4i+q))$
0	$\mathrm{Re}(X_{n-M/4}(4i)) + \mathrm{Re}(D_n(4i))$	$\mathrm{Im}(X_{n-M/4}(4i)) + \mathrm{Im}(D_n(4i))$
1	$-[\mathrm{Im}(X_{n-M/4}(4i+1)) + \mathrm{Im}(D_n(4i+1))]$	$\mathrm{Re}(X_{n-M/4}(4i+1)) + \mathrm{Re}(D_n(4i+1))$
2	$-[\mathrm{Re}(X_{n-M/4}(4i+2)) + \mathrm{Re}(D_n(4i+2))]$	$-[\mathrm{Im}(X_{n-M/4}(4i+2)) + \mathrm{Im}(D_n(4i+2))]$
3	$\mathrm{Im}(X_{n-M/4}(4i+3)) + \mathrm{Im}(D_n(4i+3))$	$-[\mathrm{Re}(X_{n-M/4}(4i+3)) + \mathrm{Re}(D_n(4i+3))]$

[0089] $X_{n-M/4}(k)$ 와 $D_n(k)$ 을 사용하여 $X_n(k)$ 계산하는데, 단지 두 실수 가산만 필요한 것이 증명되었다.

[0090] 반복되는 계산으로부터 부정확한 트위들 인자에 의한 곱셈이 제외됨으로써, 수치적 에러가 누적되지 않아 안정적임을 알 수 있다.

[0091] 수학식 4에서 보여지는 바와 같이, UVT는 DFT와 유사하므로, FFT 알고리즘이 빠른 구현을 위해 사용될 수 있다.

[0092] 본 발명의 일 실시예에서는 기수-2 시분할(decimation-in-time(DIT) 알고리즘을 이용하여 각 분해 단계(decimation stage)에서 $L/2$ 의 두개의 인터리브된 UVT로 L 사이즈의 UVT를 분해할 수 있다.

[0093] DIT 접근에 기초하여 분해된 시퀀스(decimated sequeces)를 사용하여 $D_n(k)$ 를 수학식 11과 같이 나타낼 수 잇다.

수학식 11

[0094]
$$
\begin{aligned}
D_n(k) &= \sum_{m=0}^{L-1} d(\tilde{n}+m) W_M^{-mk} \\
&= \sum_{p=0}^{L/2-1} d(\tilde{n}+2p) W_M^{-(2p)k} \\
&\quad + \sum_{p=0}^{L/2-1} d(\tilde{n}+(2p+1)) W_M^{-(2p+1)k} \\
&= D_n^e(k) + W_M^{-k} D_n^o(k)
\end{aligned}
$$

[0095] 여기서, $D_n^e(k)$ 과 $D_n^o(k)$ 는 $D_n(k)$ 의 짝수 파트와 홀수 파트를 각각 나타내며, 이는 수학식 12 및 수학식 13과 같이 나타낼 수 있다.

수학식 12

[0096]
$$D_n^e(k) = \sum_{p=0}^{L/2-1} d(\tilde{n}+2p)W_M^{-2pk}$$

수학식 13

[0097]
$$D_n^o(k) = \sum_{p=0}^{L/2-1} d(\tilde{n}+(2p+1))W_M^{-2pk}$$

[0098] 수학식 11의 길이가 L인 UVT 변환값은 $\{d(\tilde{n}), d(\tilde{n}+2), \ldots, d(\tilde{n}+L-2)\}$와 $\{d(\tilde{n}+1), d(\tilde{n}+3), \ldots, d(\tilde{n}+L-1)\}$과 같이 분해된 시퀀스(decimated sequences)의 길이가 $L/2$인 두 개의 UVT 변환값을 이용하여 획득될 수 있다.

[0099] 분해 프로세스(decimation process)는 결과 시퀀스가 원-포인트 시퀀스로 감소할때까지 반복될 수 있다(도 2 참조).

[0100] UVT의 분해 단계(deciation stage)를 $l,\ l = 0, \ldots, \log_2 L - 1,$이라 칭하기로 한다.

[0101] 초기 시드로부터 l회 재귀적으로 수행되는 순차적인 분해 과정(decimation operation)의 집합을 Ψ_l로 나타내기로 한다.

[0102] 이를 수학적으로 정리하면 수학식 14와 같다.

수학식 14

[0103]
$$\begin{cases} \Psi_0 = \emptyset \\ \Psi_1 = \{e, o\} \\ \Psi_2 = \{ee, eo, oe, oo\} \\ \quad \vdots \\ \Psi_l = \{[s_{l-1}, e], [s_{l-1}, o] : \forall s_{l-1} \in \Psi_{l-1}\} \end{cases}$$

[0104] 여기서, s_l은 길이가 l인 순차적인 분해 오퍼레이션을 나타내고, $[\cdot,\cdot]$은 접합(concatenation)을 나타낸다. 이를 이용하여 수학식 11의 재귀적 관계는 수학식 15와 같이 정리될 수 있다.

수학식 15

[0105] $$D_n^{s_l}(k) = D_n^{[s_l,e]}(k) + W_M^{-2^l k} D_n^{[s_l,o]}(k)$$

[0106] 여기서, $D_n^{[s_l,e]}(k)$ 과 $D_n^{[s_l,o]}(k)$ 은 $D_n^{s_l}(k)$ 의 짝수 파트와 홀수 파트를 각각 나타낸다. 예를 들어, $D_n^e(k)$ 는 수학식 16과 같이 분해될 수 있다.

수학식 16

[0107] $$D_n^e(k) = D_n^{ee}(k) + W_M^{-2k} D_n^{eo}(k)$$

[0108] 슬라이딩 변환 과정 동안 UVT는 반복적으로 각 타임 인덱스마다 계산되어야 한다. 따라서, UVT의 계산 요구는 만일 현재 윈도우의 UVT 변환값이 이전 윈도우의 중간 계산 결과값을 이용함으로써 더 줄어들 수 있다.

[0109] $D_n^{s_l}(k)$ 계산을 위해 사용되는 분해된 시퀀스의 제1 엘리먼트의 시간 인덱스를 $\tilde{n}_{s_l},\ s_l \in \Psi_l,$ 라고 가정하기로 한다. 이 가정으로부터, 인접한 시간 인덱스들간의 UVT들의 관계는 수학식 17과 같이 나타낼 수 있다.

수학식 17

[0110] $$\begin{aligned} D_n^{[s_l,e]}(k) &= \sum_{p=0}^{L/2^i-1} d(\tilde{n}_{s_l} + 2^l \cdot 2p) W_M^{-2^{(l+1)}pk} \\ &= \sum_{p=0}^{L/2^i-1} d(\tilde{n}_{s_l} - 2^l + 2^l(2p+1)) W_M^{-2^{(l+1)}pk} \\ &= D_{n-2^l}^{[s_l,o]}(k). \end{aligned}$$

[0111] 수학식 17에서 보여지는 바와 같이, $D_n^{[s_l,e]}(k)$ 와 $D_{n-2^l}^{[s_l,o]}(k)$ 는 완벽하게 일치하는 것을 알 수 있다.

[0112] 수학식 15 및 수학식 17로부터 분해 과정(decimation process)는 수학식 18과 같이 다시 나타낼 수 있다.

수학식 18

[0113] $$D_n^{s_l}(k) = D_{n-2^l}^{[s_l,o]}(k) + W_M^{-2^l k} D_n^{[s_l,o]}$$

[0114] 즉, 결과적으로 이전 윈도우의 중간 계산 결과가 계산 손실 없이 현재 윈도우에서 재사용될 수 있다.

[0115] 따라서, 각 분해 단계(decimation stage)에서, 단지 재귀적 방법에서 단지 홀수 파트만 계산되면 되고, 홀수 파트는 간단하게 이전 윈도우에서 미리 계산된 중간 계산 결과값을 복사하여 이용할 수 있다. 이에 기반하여 본 발명의 일 실시예에 따른 슬라이딩 업데이팅 벡터 변환 계산이 수행될 수 있다.

[0116] 분해 단계(l)에서, 시분할 접근을 사용하여 SUVT는 우선 $D_n^{[s_l,o]}$ 를 계산한다.

[0117] 수학식 18에 따라 UVT 출력값($D_n^{s_l}(k)$)은 $D_n^{[s_l,o]}$ 과 $D_{n-2^l}^{[s_l,o]}(k)$ 을 사용하여 획득될 수 있다. 여기서, $D_{n-2^l}^{[s_l,o]}(k)$ 은 $(n-2^l)$ 윈도우 포지션에서 이미 획득될 수 있다.

[0118] 도 3에는 본 발명의 일 실시예에 따른 시프트된 윈도우의 모든 값들을 계산하기 위한 SUVT를 설명하기 적합한 버터플라이 구조가 예시되어 있다.

[0119] 본 발명에서는 슬라이딩 윈도우의 사이즈를 M=32로 가정하여 설명하기로 한다.

[0120] 이전 윈도우의 각 포지션에서 모든 계산은 이미 수행되어져 있고, 모든 포인트의 계산 결과가 이용될 수 있는 것을 가정하기로 한다.

[0121] 예를 들어, 신규 입력 샘플($d(\tilde{n}+L-1)$)이 입력되면, 가장 오래된 샘플($d(\tilde{n}-1)$)이 윈도우 갱신을 위해 제거된다.

[0122] 이미 전술한 바와 같이, 신규 입력 샘플의 입력에 따라 윈도우가 갱신됨에 따라 도 1의 버터플라이 구조에서 계산이 필요한 부분은 원형으로 하이라이트된 부분이다.

[0123] SUVT에서 사용되는 트위들 인자는 전통적인 버터플라이 구조 기반 FFT 알고리즘에서 사용되는 트위들 인자와 일치하기 때문에, 본 발명의 일 실시예에 따른 슬라이딩 이산 푸리에 변환 방법은 전통적인 FFT(fast furrier transform)와 동일한 정밀도를 가진다.

[0124] 도 1에서 보여지는 바와 같이, 윈도우 사이즈와 상관없이 가장 마지막 분해 단계(stage)에서 4개의 중간 계산 결과값(I_0, I_1, I_2, I_3)이 필요하다. 이러한 중간 계산 결과값은 신규 입력 샘플에 트위들 인자를 곱하여 획득될 수 있다.

[0125] 이를 수학식으로 나타내면 수학식 19와 같다.

수학식 19

[0126]
$$\begin{cases} I_0 = W_M^0 d(\tilde{n}+L-1) = d(\tilde{n}+L-1) \\ I_1 = W_M^{-M/8} d(\tilde{n}+L-1) \\ I_2 = W_M^{-M/4} d(\tilde{n}+L-1) = -jI_0 \\ I_3 = W_M^{-3M/8} d(\tilde{n}+L-1) = -jI_1 \end{cases}$$

[0127] 여기서, $W_M^0 = 1, W_M^{-M/4} = -j$, $W_M^{-3M/8} = -jW_M^{-M/8}$ 이다. 또한, I_0 는 신규 입력 샘플($d(\tilde{n}+L-1)$)과 동일한 것을 알 수 있다.

[0128] 또한, I_2 는 I_0 에 의해 획득될 수 있다. 이를 다시 수학식으로 정리하면 수학식 20과 같다.

수학식 20

[0129]
$$\begin{cases} \mathrm{Re}(I_2) = \mathrm{Im}(I_0) = \mathrm{Im}(d(\tilde{n}+L-1)) \\ \mathrm{Im}(I_2) = -\mathrm{Re}(I_0) = -\mathrm{Re}(d(\tilde{n}+L-1)) \end{cases}$$

[0130] 여기서, $I_2 = -jI_0$ 이다. 또한, I_3 는 I_1 을 이용하여 획득될 수 있다. 결과적으로, 마지막 분해 단계에서 단지 I_1 만 계산되면 되는 것을 알 수 있다.

[0131] I_1 의 계산을 위한 빠른 스킴에 대해 설명하기로 한다.

[0132] $W_M^{-M/8}$ 는 허수 부분과 실수 부분이 동일하기 때문, I_1 는 수학식 21과 같이 간단하게 나타낼 수 있다.

수학식 21

[0133]
$$\begin{cases} \mathrm{Re}(I_1) = \alpha[\mathrm{Re}(d(\hat{n}+L-1)) - \mathrm{Im}(d(\hat{n}+L-1))] \\ \mathrm{Im}(I_1) = \alpha[\mathrm{Re}(d(\hat{n}+L-1)) + \mathrm{Im}(d(\hat{n}+L-1))] \end{cases}$$

[0134] 여기서, $\alpha = \mathrm{Re}(W_M^{-M/8}) = \mathrm{Im}(W_M^{-M/8})$ 이다.

[0135] 결과적으로, 단지 마지막 분해 단계에서 두개의 실수 곱셈과 두개의 실수 덧셈에 의해 중간 계산 결과값(I_1)이 계산될 수 있음을 의미한다.

[0136] 즉, 본 발명의 일 실시예에 따른 슬라이딩 이산 푸리에 변환 방법은 각 회차에서 고정된 사이즈의 윈도우에 하나의 샘플을 갱신시킴에 따라 이산 푸리에 변환값을 반복적으로 출력할 수 있다.

[0137] 현재 시간 인덱스 n에서 윈도우 사이즈가 M인 경우, 슬라이딩 이산 푸리에 변환 방법을 정리하면 다음과 같다.

[0138] 입력 샘플을 이용하여 $d(n) = x(n) - x(n-M)$ 을 계산한다. 이어, 수학식 19, 20, 21의 빠른 실행 스킴을 이용하여 중간 계산값(I_0, I_1, I_2, I_3)을 계산한다. 이때, 두번의 실수 곱셈, 두번의 실수 덧셈 및 8번의 복소수 덧셈이 요구된다.

[0139] 수학식 18에 따라서, 재귀적 관계를 사용하여 업데이팅 벡터 변환값($D_n(k)$)를 계산한다. 각 분할 계층(decimation stage)에서 $M/2^{(l+1)}$ 복소수 곱셈과 $M/2^l$ 복소수 덧셈이 요구된다.

[0140] 전체(M-8) 복소수 곱셈과 (2M-16) 복소수 덧셈이 요구된다.

[0141] 마지막으로, $X_{n-M/4}(k)$과 $D_n(k)$를 이용하여 현재 윈도우에 대한 이산 푸리에 변환값($\mathrm{DFT}\ X_n(k)$)을 계산한다. 여기서, $X_{n-M/4}(k)$는 이전 윈도우 포지션($n-M/4$)에서 미리 계산되어져 있는 것을 가정하기로 한다. 이때, M 복소수 덧셈이 요구된다.

[0142] 본 발명의 일 실시예에 따른 슬라이딩 이산 푸리에 변환 방법은 이전 윈도우의 각 포지션의 이산 푸리에 변환값 및 중간 계산값을 각각 저장하기 위한 추가적인 메모리가 필요하다.

[0144] 도 4는 본 발명의 일 실시예에 따른 슬라이딩 이산 푸리에 변환 장치의 구성을 개략적으로 도시한 블록도이다.

[0145] 도 4를 참조하면, 본 발명의 일 실시예에 따른 슬라이딩 이산 푸리에 변환 장치(100)는 분할부(410), 제1 계산부(415), UVT 계산부(420), DFT 계산부(425), 메모리(430), 프로세서(435)를 포함하여 구성된다.

[0146] 이미 전술한 바와 같이, 본 발명의 일 실시예에 따른 슬라이딩 이산 푸리에 변환 장치(100)는 M 사이즈의 고정된 윈도우 상에 입력 샘플들이 하나씩 신규로 갱신되는 것을 가정하기로 한다.

[0147] 분할부(410)는 현재 윈도우상의 L개의 입력 샘플들을 기수-2 시분할 방식에 기반하여 복수의 계층으로 분할하기 위한 수단이다.

[0148] 도 1에서 설명한 바와 같이, 분할부(410)는 기수-2 시분할 방식에 기반하여 입력 샘플들이 원 포인트 시퀀스로 분할될때까지 반복하여 분해 과정(decimation operation)을 수행할 수 있다.

[0149] 제1 계산부(415)는 현재 윈도우상에서 기수-2 시분할 방식에 기반하여 입력 샘플들이 분할된 복수의 시퀀스 중 제1 시퀀스와 복수의 트위들 인자를 이용하여 하나의 제1 중간 계산값을 계산하기 위한 수단이다.

[0150] 도 1 내지 도 3을 참조하여 설명한 바와 같이, 본 발명의 일 실시예에 따른 슬라이딩 이산 푸리에 변환 장치(100)는 기수-2 시분할 방식에 기반한 각 분할 계층(stage)에서 분할된 시퀀스 중 하나의 시퀀스(제1 시퀀스)에 대해서만 제1 중간 계산값을 계산할 수 있다. 이외의 나머지 중간 계산값(즉, 제2 중간 계산값)은 이전 윈도우에서 계산된 중간 계산값을 그대로 이용한다.

[0151] 이를 위해, 하기에서 보다 상세히 설명하겠지만, 현재 윈도우에서 각 분할된 시퀀스에 상응하여 계산된 제1 중간 계산값은 다음 윈도우에서 이용되기 위해 저장될 수 있다.

[0152] 또한, 제1 시퀀스는 이미 전술한 바와 같이, 현재 윈도우에 신규 입력된 샘플을 포함하는 시퀀스일 수 있다.

[0153] 또한, 프로세서(435)의 제어에 따라 업데이팅 벡터 변환값은 상위 계층의 분할된 시퀀스 중 제1 시퀀스로 제1 계산부(415)에 입력될 수 있다. 이를 통해, 업데이팅 벡터 변환값은 상위 계층의 분할된 시퀀스 중 제1 시퀀스의 제1 중간 계산값 계산에 이용될 수 있다.

[0154] 도 2 및 도 3에서 보여지는 바와 같이, 각 분해 계층(l)에서 신규 입력된 샘플을 포함하는 시퀀스 이외의 나머지 시퀀스에 대한 중간 계산값(제2 중간 계산값) 및 업데이팅 벡터 변환값 중 적어도 하나는 이미 계산되어져 있는 것을 알 수 있다.

[0155] 이는 이미 도 1에서 상세하 설명한 바와 동일하므로 중복되는 설명은 생략하기로 한다.

[0156] UVT 계산부(420)는 제1 중간 계산값과 이전 윈도우에서 계산된 제2 중간 계산값을 이용하여 업데이팅 벡터 변환값을 계산하기 위한 수단이다.

[0157] DFT 계산부(425)는 업데이팅 벡터 변환값을 이용하여 현재 윈도우에 대한 이산 푸리에 변환값을 계산하기 위한 수단이다.

[0158] 이는 이미 도 1에서 설명한 바와 동일하므로 중복되는 설명은 생략하기로 한다.

[0159] 메모리(430)는 본 발명의 일 실시예에 따른 슬라이딩 이산 푸리에 변환 방법을 수행하기 위해 필요한 다양한 알고리즘, 이 과정에서 파생되는 다양한 데이터(예를 들어, 이전 윈도우의 각 포인트의 계산값들) 등을 저장한다.

[0160] 프로세서(435)는 본 발명의 일 실시예에 따른 슬라이딩 이산 푸리에 변환 장치(100)의 내부 구성 요소들(예를 들어, 분할부(410), 제1 계산부(415), UVT 계산부(420), DFT 계산부(425), 메모리(430)등)을 제어하기 위한 수단이다.

[0161] 또한, 프로세서(435)는 제1 계산부(410)과 UVT 계산부(420)가 재귀적으로 반복하여 동작될 수 있도록 제어하는 기능을 한다.

[0162] 또한, 프로세서(435)는 UVT 계산부(420)에서 계산된 업데이팅 벡터 변환값이 최상위 계층에서 계산된 업데이팅 벡터 변환값이 아닌 경우, 제1 시퀀스에 상응하는 상위 계층의 분할된 시퀀스 중 하나로 입력되도록 제어할 수 있다. 이는 이미 도 1에서 상세히 설명한 바와 동일하므로 중복되는 설명은 생략하기로 한다.

[0163] 또한, 프로세서(435)는 각 계층에서 분할된 시퀀스 중 제1 시퀀스에 상응하여 계산된 제1 중간 계산값 및 업데이팅 벡터 변환값 중 적어도 하나를 다음 윈도우에서 이용할 수 있도록 메모리(430)에 저장할 수 있다.

[0164] 이때, 제1 중간 계산값 및 업데이팅 벡터 변환값 중 적어도 하나는 분할된 시퀀스 중 제2 시퀀스에 대응하여 다음 윈도우에서 이용되도록 저장될 수 있다. 이 또한, 이미 도 1에서 상세히 설명한 바와 동일하므로 중복되는 설명은 생략하기로 한다.

[0165] 한편, 전술된 실시예의 구성 요소는 프로세스적인 관점에서 용이하게 파악될 수 있다. 즉, 각각의 구성 요소는 각각의 프로세스로 파악될 수 있다. 또한 전술된 실시예의 프로세스는 장치의 구성 요소 관점에서 용이하게 파악될 수 있다.

[0166] 또한 앞서 설명한 기술적 내용들은 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 실시예들을 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. 하드웨어 장치는 실시예들의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

산업상 이용가능성

[0168] 상기한 본 발명의 실시예는 예시의 목적을 위해 개시된 것이고, 본 발명에 대한 통상의 지식을 가지는 당업자라면 본 발명의 사상과 범위 안에서 다양한 수정, 변경, 부가가 가능할 것이며, 이러한 수정, 변경 및 부가는 하기의 특허청구범위에 속하는 것으로 보아야 할 것이다.

부호의 설명

[0170] 100: 슬라이딩 이산 푸리에 변환 장치

410: 분할부

415: 제1 계산부

420: UVT 계산부

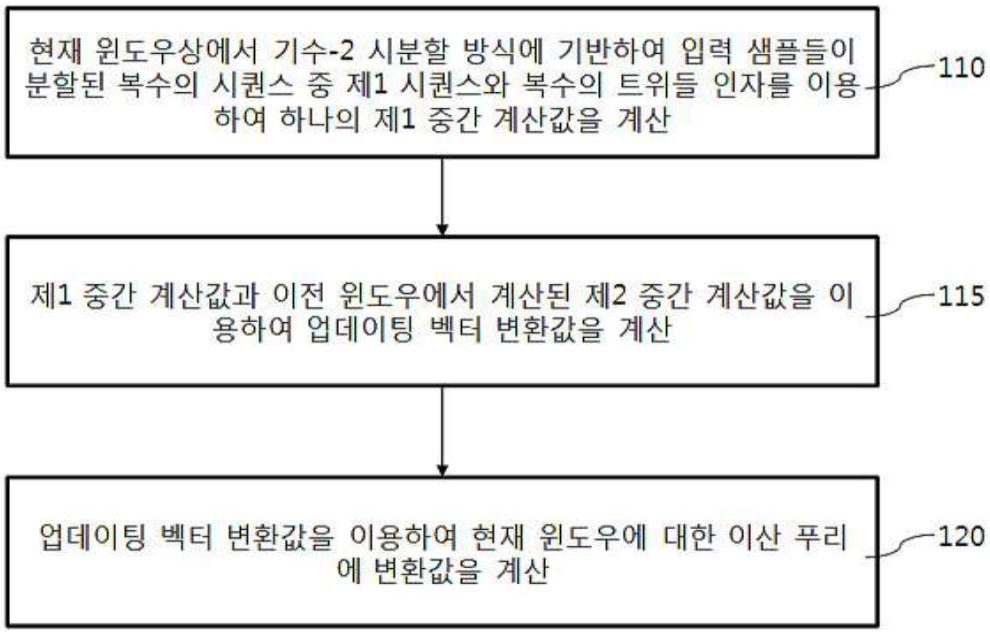
425: DFT 계산부

430: 메모리

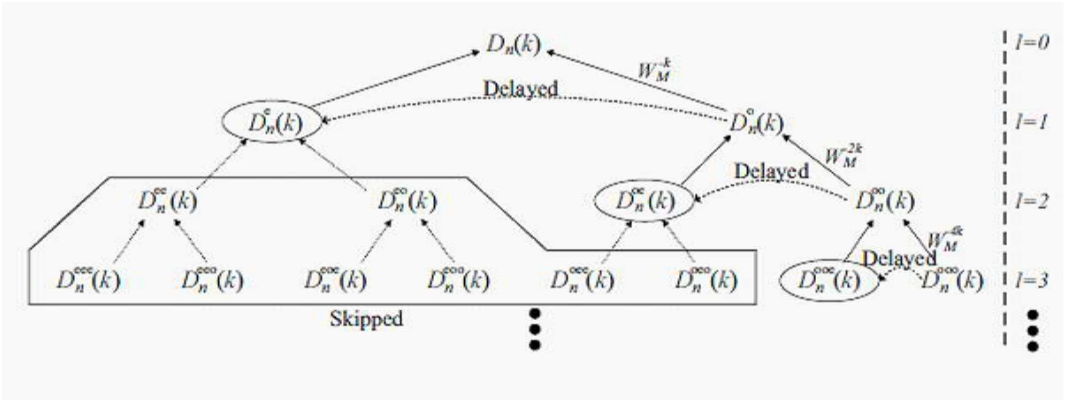
435: 프로세서

도면

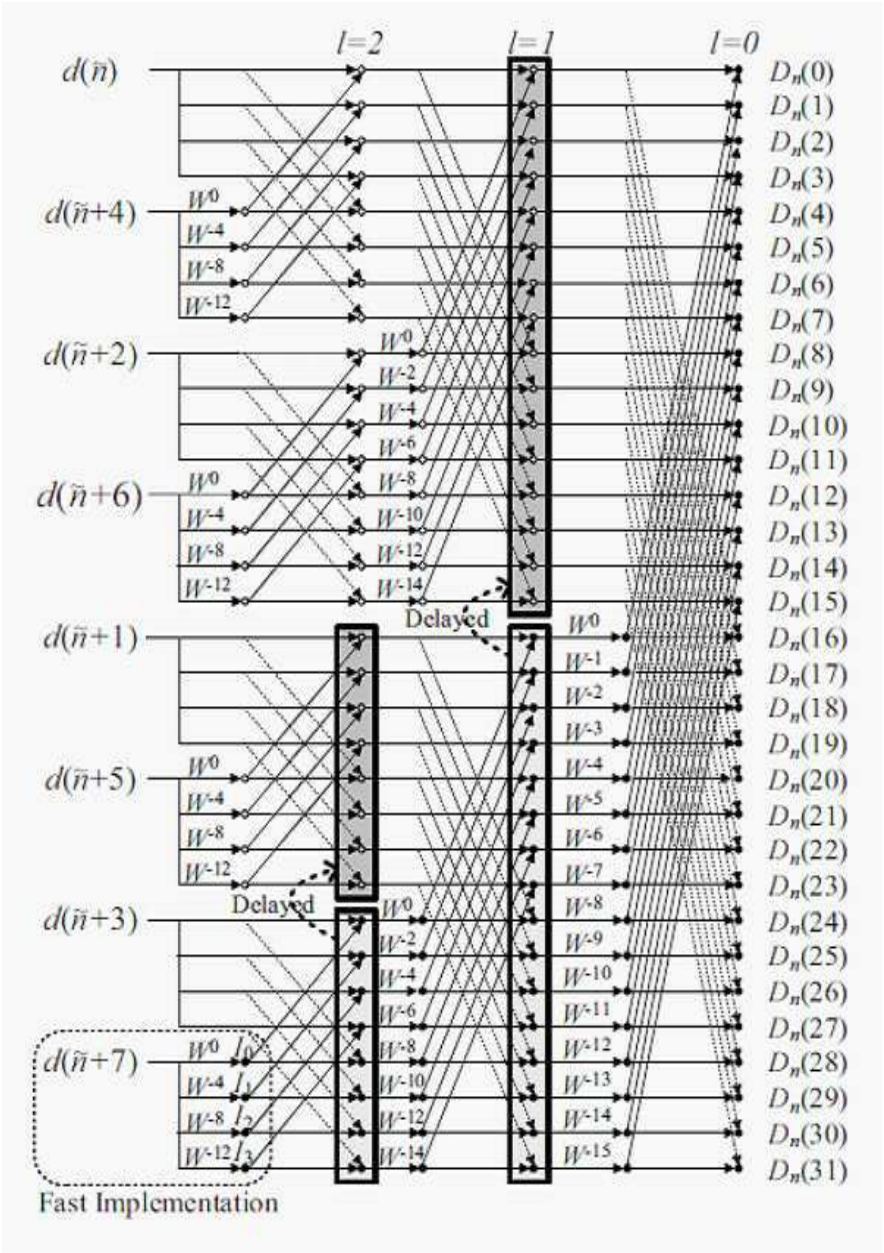
도면1


현재 윈도우상에서 기수-2 시분할 방식에 기반하여 입력 샘플들이 분할된 복수의 시퀀스 중 제1 시퀀스와 복수의 트위들 인자를 이용하여 하나의 제1 중간 계산값을 계산
110
제1 중간 계산값과 이전 윈도우에서 계산된 제2 중간 계산값을 이용하여 업데이팅 벡터 변환값을 계산
115
업데이팅 벡터 변환값을 이용하여 현재 윈도우에 대한 이산 푸리에 변환값을 계산
120

도면2


Delayed
Delayed
Delayed
Skipped
l=0
l=1
l=2
l=3

도면3


l=2
l=1
l=0
d(ñ)
d(ñ+4)
d(ñ+2)
d(ñ+6)
d(ñ+1)
d(ñ+5)
d(ñ+3)
d(ñ+7)
Delayed
Delayed
Fast Implementation
$D_n(0)$
$D_n(1)$
$D_n(2)$
$D_n(3)$
$D_n(4)$
$D_n(5)$
$D_n(6)$
$D_n(7)$
$D_n(8)$
$D_n(9)$
$D_n(10)$
$D_n(11)$
$D_n(12)$
$D_n(13)$
$D_n(14)$
$D_n(15)$
$D_n(16)$
$D_n(17)$
$D_n(18)$
$D_n(19)$
$D_n(20)$
$D_n(21)$
$D_n(22)$
$D_n(23)$
$D_n(24)$
$D_n(25)$
$D_n(26)$
$D_n(27)$
$D_n(28)$
$D_n(29)$
$D_n(30)$
$D_n(31)$

도면4

