



(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2017년07월10일
(11) 등록번호 10-1755987
(24) 등록일자 2017년07월03일

- (51) 국제특허분류(Int. Cl.)
G06F 17/14 (2006.01)
- (52) CPC특허분류
G06F 17/141 (2013.01)
G06F 17/14 (2013.01)
- (21) 출원번호 10-2016-0024639
(22) 출원일자 2016년02월29일
심사청구일자 2016년02월29일
- (56) 선행기술조사문헌
“The hopping discrete fourier transform”,
IEEE SIGNAL PROCESSING MAGAZINE(pp. 135-139),
2014.03.01.*
“The Sliding DFT”, IEEE SIGNAL PROCESSING
MAGAZINE Vol. 20 No. 2(pp. 74-80),
2003.03.31.*
“2D Discrete Fourier Transform on Sliding
Windows”, IEEE TRANSACTIONS ON IMAGE
PROCESSING Vol. 24 No. 3(pp. 901-907),
2015.03.31.
JP2013229050 A
*는 심사관에 의하여 인용된 문헌
- (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 10 항

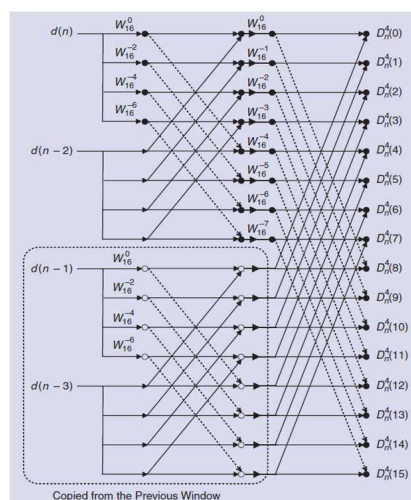
심사관 : 박승철

(54) 발명의 명칭 슬라이딩 이산 푸리에 변환 방법 및 장치

(57) 요약

슬라이딩 이산 푸리에 변환 방법 및 장치가 개시된다. 본 발명의 실시예에 따른 슬라이딩 이산 푸리에 변환 방법은 이전 윈도우에서 계산된 중간 결과를 이용하여 슬라이딩 업데이트 벡터 변환(SUVT: sliding updating vector transform)을 계산하는 단계 및 슬라이딩 업데이트 벡터 변환의 결과를 이용하여 SDFT 출력을 계산하는 단계를 포함한다.

대표도 - 도1



공지예외적용 : 있음

명세서

청구범위

청구항 1

안정성이 보장되는 슬라이딩 이산 푸리에 변환(SDFT: sliding Discrete Fourier Transform) 방법에 있어서,

이전 윈도우에서 계산된 중간 결과를 이용하여 슬라이딩 업데이트 벡터 변환(SUVT: sliding updating vector transform)을 계산하는 단계; 및

상기 슬라이딩 업데이트 벡터 변환의 결과를 이용하여 상기 SDFT 출력을 계산하는 단계를 포함하되,

상기 SUVT는,

DIT(decimation-in-time) 알고리즘을 이용하여, 각 반복 스테이지에서 L 사이즈의 UVT가 L/2 사이즈의 인터리브된(interleaved) 두 UVT로 나누어져 데시메이션(decimation)된 시퀀스의 두 UVT로 계산되며,

상기 두 UVT 중 하나는 상기 이전 윈도우에서 계산된 중간 결과이고, 상기 계산된SUVT는 다음 윈도우에서의 SUVT 계산의 입력값이 되는 것을 특징으로 하는 슬라이딩 이산 푸리에 변환 방법.

청구항 2

제1항에 있어서,

상기 SUVT는 하기 수학식으로 정의되는 것을 특징으로 하는 슬라이딩 이산 푸리에 변환 방법.

$$\begin{aligned} D_n^L(k) &= \sum_{m=0}^{L-1} d(n-m) W_M^{(m-L+1)k} \\ &= \sum_{p=0}^{L/2-1} d(n-2p) W_M^{(2p-L+1)k} \\ &\quad + \sum_{p=0}^{L/2-1} d(n-2p-1) W_M^{(2p-L+2)k} \\ &= W_M^{-k} \dot{D}_n^{L/2}(k) + \ddot{D}_n^{L/2}(k) \end{aligned}$$

여기서, $D_n^L(k)$ 는 L 포인트의 업데이트 벡터 변환(UVT: updating vector transform)의 k 번째 값이고, k 는 $0 \leq k < M$ 의 범위를 가지는 주파수 도메인의 인덱스이고, n은 시간 인덱스이고,

$d(n) = x(n) - x(n-M)$ 이고, $W_M = e^{j2\pi/M}$ 로서 복소 회전 인자(complex twiddle factor)임.

청구항 3

제2항에 있어서,

상기 $\ddot{D}_n^{L/2}(k)$ 는 $\dot{D}_{n-1}^{L/2}(k)$ 와 동일한 관계를 가지며, 상기 관계는 하기의 수학식으로 나타내지는 것을

특징으로 하는 슬라이딩 이산 푸리에 변환 방법.

$$\begin{aligned}\ddot{D}_n^{L/2}(k) &= \sum_{p=0}^{L/2-1} d((n-1)-2p) W_{M/2}^{(p-L/2+1)k} \\ &= \dot{D}_{n-1}^{L/2}(k)\end{aligned}$$

청구항 4

제2항에 있어서,

상기 SDFT는 하기 수학식으로 정의되는 것을 특징으로 하는 슬라이딩 이산 푸리에 변환 방법.

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

여기서, $X_n(k)$ 는 시간 인덱스 n 에서 M 포인트 DFT의 k 번째 값임.

청구항 5

제4항에 있어서,

상기 L 은 $M/4$, $M/2$ 및 $3M/4$ 중 어느 하나인 것을 특징으로 하는 슬라이딩 이산 푸리에 변환 방법.

청구항 6

제5항에 있어서,

상기 L 이 $M/4$ 인 경우, 상기 회전 인자 W_M^{Lk} 는 다음의 수학식으로 나타내지는 것을 특징으로 하는 슬라이딩 이산 푸리에 변환 방법.

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

청구항 7

제6항에 있어서,

상기 $X_n(k)$ 는 상기 j^k 적용되어 하기 수학식으로 간략화되는 것을 특징으로 하는 슬라이딩 이산 푸리에 변환 방법.

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

청구항 8

제7항에 있어서,

상기 L 이 $M/4$ 인 경우, 상기 $X_n(k)$ 는 상기 j^k 의 주기 특성에 의하여 하기의 수학식으로 일반화되는 것을 특징으로

하는 슬라이딩 이산 푸리에 변환 방법.

$$\begin{bmatrix} X_n(4i+0) \\ X_n(4i+1) \\ X_n(4i+2) \\ X_n(4i+3) \end{bmatrix} = \begin{bmatrix} X_{n-M/4}(4i+0) + D_n^{M/4}(4i+0) \\ j(X_{n-M/4}(4i+1) + D_n^{M/4}(4i+1)) \\ -(X_{n-M/4}(4i+2) + D_n^{M/4}(4i+2)) \\ -j(X_{n-M/4}(4i+3) + D_n^{M/4}(4i+3)) \end{bmatrix}$$

청구항 9

제8항에 있어서,

상기 $k=4i$ 일 때, $X_n(4i)$ 와 $X_{n-M/4}(4i)$ 사이의 관계는 다음의 수학식으로 나타내어지는 것을 특징으로 하는 슬라이딩 이산 푸리에 변환 방법.

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

여기서, $\text{Re}(\)$ 및 $\text{Im}(\)$ 의 표현은 각각 복소수의 실수 부분 및 허수 부분을 나타냄.

청구항 10

안정성이 보장되는 슬라이딩 이산 푸리에 변환(SDFT: sliding Discrete Fourier Transform) 방법을 수행하는 슬라이딩 이산 푸리에 변환 장치에 있어서,

명령어를 저장하는 메모리; 및

상기 명령어를 실행하는 프로세서를 포함하되,

상기 명령어는,

이전 윈도우에서 계산된 중간 결과를 이용하여 슬라이딩 업데이트 벡터 변환(SUVT: sliding updating vector transform)을 계산하는 단계; 및

상기 슬라이딩 업데이트 벡터 변환의 결과를 이용하여 상기 SDFT 출력을 계산하는 단계를 포함하는 슬라이딩 이산 푸리에 변환 방법을 수행하되,

상기 SUVT는,

DIT(decimation-in-time) 알고리즘을 이용하여, 각 반복 스테이지에서 L 사이즈의 UVT가 L/2 사이즈의 인터리브된(interleaved) 두 UVT로 나누어져 데시메이션(decimation)된 시퀀스의 두 UVT로 계산되며,

상기 두 UVT 중 하나는 상기 이전 윈도우에서 계산된 중간 결과이고, 상기 계산된SUVT는 다음 윈도우에서의 SUVT 계산의 입력값이 되는 것을 특징으로 하는 슬라이딩 이산 푸리에 변환 장치.

발명의 설명

기술 분야

[0001] 본 발명은 슬라이딩 이산 푸리에 변환 방법 및 장치에 관한 것이다.

배경 기술

[0002] DFT(Discrete Fourier transform), DHT(Discrete Hartley Transform), 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.

[0003] 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.

[0004] [2] E. Jacobsen and R. Lyons, "The sliding DFT," IEEE Signal Processing Mag., vol. 20, no. 2, pp. 74??80, Mar. 2003.

[0005] [3] K. Duda, "Accurate, guaranteed stable, sliding discrete Fourier transform," IEEE Signal Processing Mag., vol. 27, no. 6, pp. 124??127, Nov. 2010.

[0006] [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.

[0007] 이러한 알고리즘들은 계산 복잡도를 높이거나 계산 정확도를 낮추는 과정을 통해 슬라이딩 변환 프로세스의 안정성을 보장한다.

발명의 내용

해결하려는 과제

[0008] 본 발명은 빠르고 정확하며 안정성이 보장되는 새로운 슬라이딩 이산 푸리에 변환 방법 및 장치를 제공하기 위한 것이다.

과제의 해결 수단

[0009] 본 발명의 일 측면에 따르면, 안정성이 보장되는 슬라이딩 이산 푸리에 변환(SDFT: sliding Discrete Fourier Transform) 방법이 개시된다.

[0010] 본 발명의 실시예에 따른 슬라이딩 이산 푸리에 변환 방법은 이전 윈도우에서 계산된 중간 결과를 이용하여 슬라이딩 업데이팅 벡터 변환(SUVT: sliding updating vector transform)을 계산하는 단계 및 상기 슬라이딩 업데이팅 벡터 변환의 결과를 이용하여 상기 SDFT 출력을 계산하는 단계를 포함한다.

[0011] 상기 SUVT는 하기 수학식으로 정의된다.

$$\begin{aligned} D_n^L(k) &= \sum_{m=0}^{L-1} d(n-m) W_M^{(m-L+1)k} \\ &= \sum_{p=0}^{L/2-1} d(n-2p) W_M^{(2p-L+1)k} \\ &\quad + \sum_{p=0}^{L/2-1} d(n-2p-1) W_M^{(2p-L+2)k} \\ &= W_M^{-k} \dot{D}_n^{L/2}(k) + \ddot{D}_n^{L/2}(k) \end{aligned}$$

[0012]

[0013] 여기서, $D_n^L(k)$ 는 L 포인트의 업데이트 벡터 변환(UVT: updating vector transform)의 k 번째 값이고, k 는 $0 \leq k < M$ 의 범위를 가지는 주파수 도메인의 인덱스이고, n은 시간 인덱스이고, $d(n) = x(n) - x(n-M)$ 이고, $W_M = e^{j2\pi/M}$ 로서 복소 회전 인자(complex twiddle factor)임.

[0014]

상기 $\ddot{D}_n^{L/2}(k)$ 와 상기 $\dot{D}_{n-1}^{L/2}(k)$ 는 동일한 관계를 가지며, 상기 관계는 하기의 수학식으로 나타내진다.

$$\begin{aligned} \ddot{D}_n^{L/2}(k) &= \sum_{p=0}^{L/2-1} d((n-1)-2p) W_{M/2}^{(p-L/2+1)k} \\ &= \dot{D}_{n-1}^{L/2}(k) \end{aligned}$$

[0015]

[0016] 상기 SDFT는 하기 수학식으로 정의된다.

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

[0017]

[0018] 여기서, $X_n(k)$ 는 시간 인덱스 n에서 M 포인트 DFT의 k번째 값임.

[0019]

상기 L은 M/4, M/2 및 3M/4 중 어느 하나이다.

[0020]

상기 L이 M/4인 경우, 상기 회전 인자 W_M^{Lk} 는 다음의 수학식으로 나타내진다.

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

[0021]

[0022] 상기 $X_n(k)$ 는 상기 j^k 적용되어 하기 수학식으로 간략화된다.

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

[0023]

[0024] 상기 L이 M/4인 경우, 상기 $X_n(k)$ 는 상기 j^k 의 주기 특성에 의하여 하기의 수학식으로 일반화된다.

$$\begin{bmatrix} X_n(4i+0) \\ X_n(4i+1) \\ X_n(4i+2) \\ X_n(4i+3) \end{bmatrix} = \begin{bmatrix} X_{n-M/4}(4i+0) + D_n^{M/4}(4i+0) \\ j(X_{n-M/4}(4i+1) + D_n^{M/4}(4i+1)) \\ -(X_{n-M/4}(4i+2) + D_n^{M/4}(4i+2)) \\ -j(X_{n-M/4}(4i+3) + D_n^{M/4}(4i+3)) \end{bmatrix}$$

[0025]

[0026] 상기 $k=4i$ 일 때, $X_n(4i)$ 와 $X_{n-M/4}(4i)$ 사이의 관계는 다음의 수학식으로 나타내어진다.

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

[0027]

[0028] 여기서, $\text{Re}(\)$ 및 $\text{Im}(\)$ 의 표현은 각각 복소수의 실수 부분 및 허수 부분을 나타냄.

[0029] 본 발명의 다른 측면에 따르면, 안정성이 보장되는 슬라이딩 이산 푸리에 변환(SDFT: sliding Discrete Fourier Transform) 방법을 수행하는 슬라이딩 이산 푸리에 변환 장치가 개시된다.

[0030] 본 발명의 실시예에 따른 슬라이딩 이산 푸리에 변환 장치는 명령어를 저장하는 메모리 및 상기 명령어를 실행하는 프로세서를 포함하되, 상기 명령어는, 이전 윈도우에서 계산된 중간 결과를 이용하여 슬라이딩 업데이팅 벡터 변환(SUVT: sliding updating vector transform)을 계산하는 단계 및 상기 슬라이딩 업데이팅 벡터 변환의 결과를 이용하여 상기 SDFT 출력을 계산하는 단계를 포함하는 슬라이딩 이산 푸리에 변환 방법을 수행한다.

발명의 효과

[0031] 본 발명의 실시예에 따른 슬라이딩 이산 푸리에 변환 방법 및 장치는 빠르고 정확하며 안정성이 보장될 수 있으며, 기존에 안정성이 보장되는 SDFT 알고리즘들 중에서 가장 계산량이 적고, 윈도우 크기와 시간 인덱스에 상관없이 DFT와 수학적으로 동등한 출력값을 제공할 수 있다.

도면의 간단한 설명

[0032] 도 1은 본 발명의 실시예에 따른 슬라이딩 업데이팅 벡터 변환 방법의 예를 나타낸 도면.

도 2는 본 발명의 실시예에 따른 슬라이딩 이산 푸리에 변환 방법을 나타낸 흐름도.

도 3은 알고리즘별 M값에 따른 계산 복잡도를 나타낸 도면.

도 4는 본 발명의 실시예에 따른 슬라이딩 이산 푸리에 변환 방법의 시뮬레이션 결과를 나타낸 도면.

도 5는 본 발명의 실시예에 따른 슬라이딩 이산 푸리에 변환 장치의 구성을 개략적으로 예시한 도면.

발명을 실시하기 위한 구체적인 내용

[0033] 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.

[0034] 우선, 본 발명의 실시예에 따른 슬라이딩 이산 푸리에 변환 방법을 설명하기 앞서, 본 발명의 이해와 설명의 편의를 도모하고자, 기존에 안정성이 보장되는 SDFT 알고리즘에 대하여 설명하기로 한다.

[0035] SDFT 알고리즘에서는, 고정 길이의 윈도우에서 변환값이 계산된다. 복소 입력 신호($x(n)$, $n = 0, 1, 2, \dots$)는 M 사이즈의 중첩 윈도우로 나누어진다. 그리고, k 가 $0 \leq k < M$ 의 범위를 가지는 주파수 도메인의 인덱스라고 가정하면, 시간 인덱스 n 에서 M 포인트 DFT의 k 번째 값은 다음의 수학식으로 계산될 수 있다.

수학식 1

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

[0036] 여기서, $\hat{n} = n - M + 1$ 이고, $W_M = e^{j2\pi/M}$ 이다.

[0037] 따라서, 원형 시프트 특성에 의하여 SDFT[1]의 공식은 다음의 수학식으로 나타내질 수 있다.

수학식 2

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

[0038] 여기서, 복소 회전 인자(complex twiddle factor)의 주기 특성이 이용된다($W_M^{k+M} = W_M^k$).

[0041] SDFT는, 폴(pole)이 z 도메인의 단위원에 놓이기 때문에, 조금 안정한 전달 함수를 가질 수 있다. 실제로, 수학식 2에서의 복소 회전 인자는 한정된 정밀도를 가지는 이동 소수점 형식으로 나타내어 질 수 있다. 복소 회전 인자의 수치 라운딩(numerical rounding)은 단위원 밖으로 폴을 이동시킬 수 있으며, 시스템의 불안정을 야기할 수 있다. 이러한 문제를 해결하고자, 간단한 반복 업데이트 스킴을 이용한 안정성이 보장되는 SDFT 알고리즘이 제안된 바 있다[2]. 이 알고리즘은 주기적인 time-varying 시스템을 이용하여 실현됨으로써, 한정된 정밀도 연산에 의하여 발생하는 수치 에러들을 시간축에서 기하급수적으로 0으로 감소시킬 수 있다.

[0042] 다른 안정성이 보장되는 SDFT 알고리즘으로서, rSDFT라 불리우는 알고리즘[3]이 제안된 바 있다. 이 알고리즘은 댐핑 지수(damping factor) r 를 이용하여 단위원 내부의 r 반경에 폴이 위치하게 함으로써, 안정성을 보장할 수 있다. 댐핑 지수 r 를 이용한 DFT 근사치는 다음의 수학식으로 나타낼 수 있다.

수학식 3

$$\tilde{X}_n(k) = \sum_{m=0}^{M-1} x(\hat{n} + m) r^{M-m} W_M^{-km}$$

[0043]

[0044] 여기서, $0 < r < 1$ 이고, 물결 표시는 DFT의 근사치를 나타낸다. 비슷하게, 수학식 2의 SDFT 반복식은 다음의 수학식으로 나타낼 수 있다.

수학식 4

$$\begin{aligned} \tilde{X}_n(k) &= r W_M^k (\tilde{X}_{n-1}(k) \\ &\quad + x(n) - x(n-M) r^M) \end{aligned}$$

[0045]

[0046] rSDFT의 출력값은 수학식 1의 DFT와는 다르다. 더욱이, 출력값에 에러가 축적될 수 있다.

[0047] 최근, 변조된 SDFT(mSDFT: modulated SDFT) 알고리즘[4]이 소개된 바 있다. 이 mSDFT는 우선, 입력 신호에 변조 시퀀스(W_M^{-km})를 곱함으로써, 변조된 시퀀스를 산출한다. 다음으로, mSDFT는 변조된 시퀀스를 이용하여, $k=0$ 일 때의 DFT의 반복식을 다음과 같은 수학식으로 산출할 수 있다.

수학식 5

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

[0048]

[0049] 여기서, $W_M^{-k(m-M)} = W_M^{-km}$ 이다. 결국, $X_n(k)$ 는 다음의 수학식으로 산출될 수 있다.

수학식 6

$$X_n(k) = W_M^{k(m+1)} X_n(0)$$

[0050]

[0051] 즉, 공진기의 피드백으로부터 복소 회전 인자를 제외시킴으로써, mSDFT는 정확하게 단위원에 위치하는 폴을 가질 수 있고, 조건없이 안정성이 보장될 수 있다. 그러나, 이 mSDFT는 다른 SDFT 알고리즘보다 계산량이 두 배 이상 많다.

[0052] 이하, 본 발명의 실시예에 따른 슬라이딩 이산 푸리에 변환 방법에 대하여 설명하기로 한다.

[0053] 전술한 바와 같이, 안정성이 보장되는 SDFT 필터는 피드백 루프로부터 부정확한 회전 인자를 제외시킴으로써 실현될 수 있다. 그래서, 회전 인자를 이용하지 않고 표현될 수 있는 DFT 출력값 사이의 특이한 관계를 조사하는 것이 필요하다.

[0054] 기본적으로, SDFT는 연속적인 DFT값 사이의 반복적인 관계를 이용함으로써 DFT값을 계산할 수 있다. 즉, 연속적인 값 사이의 관계를 확장시킴으로써, L홉의 거리를 가지는 DFT값 사이의 일반적인 공식이 유도될 수 있다.

$d(n) = x(n) - x(n - M)$ 이라 가정하고, 수학식 2에서 $X_n(k)$ 를 $X_{n-1}(k)$ 을 이용하여 L번 반복적으로 대체함으로써, 다음과 같은 수학식이 획득될 수 있다.

수학식 7

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

[0055]

[0056] L 포인트의 업데이트 벡터 변환(UVT: updating vector transform)의 k 번째 값으로서 $D_n^L(k)$ 가 정의될 수 있으며, 다음의 수학식으로 나타낼 수 있다.

수학식 8

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

[0057]

[0058] 여기서, $0 \leq k < M$ 이다. 이 표기법을 이용하여 수학식 7은 간략화되어 다음의 수학식으로 나타낼 수 있다.

수학식 9

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

[0059]

[0060] 이것은, $D_n^L(k)$ 을 이용함으로써, 시간 인덱스 n 에서의 DFT 출력값이 시간 인덱스 $(n-L)$ 에서의 DFT 출력값으로부터 바로 계산되게 한다. 본 발명의 실시예에 따른 슬라이딩 업데이팅 벡터 변환 방법에 대해서는 이후 도 1을 참조하여 상세히 후술하기로 한다.

[0061] 수학식 9에서, 회전 인자 W_M^{Lk} 는 지연된 출력값과 곱해지며, 회전 인자의 값은 시간 홉 L 에 따라 변한다. 이하에서는 $L=M/4$ 인 경우로 가정하여 설명하나 이에 한정되는 것은 아니며, 예를 들어, $L=M/2$ 인 경우 및 $L=3M/4$ 인 경우도 가능하다.

[0062] 즉, L 이 $M/4$ 인 경우, 회전 인자 W_M^{Lk} 는 다음의 수학식과 같이 나타낼 수 있다.

수학식 10

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

[0063]

[0064] 따라서, 수학식 9가 보다 간략화되어 다음의 수학식으로 나타낼 수 있다.

수학식 11

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

[0065]

[0066] 그리고, j 의 주기 특성에 의하여 다음과 같은 수학식이 획득될 수 있다.

수학식 12

$$\begin{bmatrix} X_n(4i+0) \\ X_n(4i+1) \\ X_n(4i+2) \\ X_n(4i+3) \end{bmatrix} = \begin{bmatrix} X_{n-M/4}(4i+0) + D_n^{M/4}(4i+0) \\ j(X_{n-M/4}(4i+1) + D_n^{M/4}(4i+1)) \\ -(X_{n-M/4}(4i+2) + D_n^{M/4}(4i+2)) \\ -j(X_{n-M/4}(4i+3) + D_n^{M/4}(4i+3)) \end{bmatrix}$$

[0067]

[0068] 여기서, $i=0, 1, \dots, M/4-1$ 이다. 이것은 시프트된 윈도우의 DFT 출력값을 계산하기 위한 효과적인 스킴의 기초를 형성할 수 있다. 시간 인덱스 $(n-M/4)$ 에서 앞선 윈도우의 DFT 출력값이 주어지면, 시간 인덱스 n 에서 DFT 출

력값은 회전 인자와 곱해지지 않고 바로 계산될 수 있다. 예를 들어, $k=4i$ 일 때, $X_n(4i)$ 와 $X_{n-M/4}(4i)$ 사이의 관계는 다음의 수학식으로 나타낼 수 있다.

수학식 13

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

[0069]

[0070] 여기서, $\text{Re}(\)$ 및 $\text{Im}(\)$ 의 표현은 각각 복소수의 실수 부분 및 허수 부분을 나타낸다. 그리고, $X_n(4i)$ 의 각 부분은 다음의 수학식으로 획득될 수 있다.

수학식 14

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

[0071]

[0072] $k=4i+1, 4i+2, 4i+3$ 인 경우에도 비슷하게 계산될 수 있다. 이 결과는 $X_{n-M/4}(k)$ 과 $D_n^{M/4}(k)$ 을 이용하여 $X_n(k)$ 를 계산하기 위하여 오직 두 개의 실수 부분만이 필요함을 보여준다. 더욱이, 부정확한 회전 인자에 의한 곱셈이 반복 계산에서 제외되기 때문에, 수치 에러는 축적되지 않는다. 그러므로, 본 발명의 실시예에 따른 슬라이딩 이산 푸리에 변환 방법은 안정성이 보장될 수 있다(gSDFT: guaranteed-stable SDFT).

[0073] 도 1은 본 발명의 실시예에 따른 슬라이딩 업데이트 벡터 변환 방법의 예를 나타낸 도면이다.

[0074] 수학식 8에서의 UVT의 빠른 계산은, 본 발명의 실시예에 따른 슬라이딩 이산 푸리에 변환 방법의 계산 코스트를 결정하므로, 중요한 부분이다. UVT값의 계산은 DFT와 유사하기 때문에, 기존의 FFT(fast Fourier transform) 알고리즘이 사용될 수 있다. 본 발명의 실시예에 따른 슬라이딩 업데이트 벡터 변환(SUVT: sliding UVT) 방법에는, 기수2(radix-2) 시분할(DIT: decimation-in-time) 알고리즘이 사용된다. 이 알고리즘은, 각 반복 스테이지에서 L 사이즈의 UVT를 $L/2$ 사이즈의 인터리브된(interleaved) 두 UVT로 나눈다. 시분할 알고리즘에 기초하여, 데시메이션(decimation)된 시퀀스를 이용한 $D_n^L(k)$ 가 다음의 수학식으로 나타낼 수 있다.

수학식 15

$$\begin{aligned}
D_n^L(k) &= \sum_{m=0}^{L-1} d(n-m) W_M^{(m-L+1)k} \\
&= \sum_{p=0}^{L/2-1} d(n-2p) W_M^{(2p-L+1)k} \\
&\quad + \sum_{p=0}^{L/2-1} d(n-2p-1) W_M^{(2p-L+2)k} \\
&= W_M^{-k} \dot{D}_n^{L/2}(k) + \ddot{D}_n^{L/2}(k)
\end{aligned}$$

[0075]

[0076] 여기서, $\dot{D}_n^{L/2}(k)$ 및 $\ddot{D}_n^{L/2}(k)$ 는 다음의 수학식으로 나타낼 수 있다.

수학식 16

$$\begin{aligned}
\dot{D}_n^{L/2}(k) &= \sum_{p=0}^{L/2-1} d(n-2p) W_{M/2}^{(p-L/2+1)k} \\
\ddot{D}_n^{L/2}(k) &= \sum_{p=0}^{L/2-1} d(n-2p-1) W_{M/2}^{(p-L/2+1)k}
\end{aligned}$$

[0077]

[0078] 수학식 15는 데시메이션된 시퀀스의 두 UVT값을 이용하여 $D_n^L(k)$ 의 획득한 것을 나타낸다. 여기서, 데시메이션된 시퀀스는 $\{d(n), d(n-2), \dots, d(n-L+2)\}$ 및 $\{d(n-1), d(n-3), \dots, d(n-L+1)\}$ 이다. 데시메이션 프로세스는 결과 시퀀스가 원포인트 시퀀스로 감소할 때까지 반복된다.

[0079] SDFT 알고리즘에서는, UVT가 각 시간 인덱스에서 반복적으로 계산되는 것이 필요하다. 그래서, 현재 윈도우의 UVT 출력값이 앞선 윈도우의 중간 계산을 이용하여 획득될 수 있다면, UVT의 계산량은 보다 더 감소될 수 있다. 이로부터, 연속적인 시간 인덱스에서의 UVT 계산이 다음의 수학식으로 표현되는 관계를 가짐을 간단하게 도출할 수 있다.

수학식 17

$$\begin{aligned}
\ddot{D}_n^{L/2}(k) &= \sum_{p=0}^{L/2-1} d((n-1)-2p) W_{M/2}^{(p-L/2+1)k} \\
&= \dot{D}_{n-1}^{L/2}(k)
\end{aligned}$$

[0080]

- [0081] 수학식 17은 $\ddot{D}_n^{L/2}(k)$ 이 정확히 $\dot{D}_{n-1}^{L/2}(k)$ 과 동일함을 나타낸다. 이는, 계산 정확도의 결여없이 현재 윈도우에 대하여 앞선 윈도우의 중간 계산이 재사용될 수 있기 때문에 중요하다. 이 결과를 기초로, 본 발명의 실시예에 따른 슬라이딩 업데이트 벡터 변환 방법이 도출될 수 있다.
- [0082] 시간 인덱스 n 에서, 본 발명의 실시예에 따른 SUVT 알고리즘은 우선, 시분할(DIT) 알고리즘을 이용하여 $\dot{D}_n^{L/2}(k)$ 를 계산한다. 그런 다음, 수학식 15에 따라 $\dot{D}_n^{L/2}(k)$ 와 $\ddot{D}_n^{L/2}(k)$ 를 이용하여 UVT 출력값 $D_n^L(k)$ 이 획득된다. 여기서, $\ddot{D}_n^{L/2}(k)$ 는 이미 앞선 윈도우에서 획득되었다. 이를 위하여 필요한 것은 앞선 윈도우의 중간 계산 결과를 저장하는 추가적인 메모리일 뿐이다.
- [0083] 도 1은 본 발명의 실시예에 따른 SUVT 알고리즘의 예로서, $M=16$ 이고 $L=4$ 일 때 SUVT 계산 과정을 보여주고 있다. 도 1의 예에서, 일반선과 점선은 각각 더하기 및 빼기를 의미하며, 오직 흑색점으로 표시된 부분만 계산에 이용된다. 본 발명의 실시예에 따른 gSDFT는, SUVT에서 사용되는 회전 인자가 기존의 버터플라이 기반의 FFT 알고리즘에서 사용되는 것과 동일하기 때문에, 정확도가 기존의 FFT와 동일하다.
- [0084] 도 2는 본 발명의 실시예에 따른 슬라이딩 이산 푸리에 변환 방법을 나타낸 흐름도이다.
- [0085] 본 발명의 실시예에 따른 슬라이딩 이산 푸리에 변환 방법은 고정 길이의 윈도우를 한 번에 한 샘플 시프팅시켜 DFT 출력값을 반복적으로 산출한다. 이하에서는, 발명의 이해와 설명의 편의를 위하여, $L=M/4$ 인 경우에 대해서만 설명하기로 하며, 시간 인덱스는 n 이고 윈도우 사이즈는 M 이라 가정하기로 한다.
- [0086] S210 단계에서, 슬라이딩 이산 푸리에 변환 장치는 입력 샘플을 이용하여 $d(n)$ 을 계산한다. 이 단계에서는, 오직 하나의 복소 입력 샘플만이 필요하다.
- [0087] S220 단계에서, 슬라이딩 이산 푸리에 변환 장치는 $\{d(n), d(n-2), \dots, d(n-M/4)\}$ 를 이용하여 $\dot{D}_n^{M/8}(k)$ 를 계산한다. 여기서, $\{d(n), d(n-2), \dots, d(n-M/4)\}$ 은 앞선 윈도우에서의 계산 과정을 통해 이미 획득된 것이다. 테이메이션은 $(\log_2 M - 3)$ 번 수행되기 때문에, $\dot{D}_n^{M/8}(k)$ 의 계산에는, $(M/4)(\log_2 M - 3)$ 번의 복소 곱셈과 $(M/2)(\log_2 M - 3)$ 개의 복소 입력 샘플이 필요하다.
- [0088] S230 단계에서, 슬라이딩 이산 푸리에 변환 장치는 $\dot{D}_n^{M/8}(k)$ 및 $\ddot{D}_n^{M/8}(k)$ 를 이용하여 L 차수의 UVT $D_n^{M/4}(k)$ 를 계산한다. 여기서, $\ddot{D}_n^{M/8}(k)$ 는 앞선 윈도우의 계산 과정에서 계산된 것이다. 계산된 $D_n^{M/4}(k)$ 은 다음 단계에서 입력값이 된다. 이 단계에서는, 수학식 15에서 보여지듯이, $M/2$ 번의 복소 곱셈과 M 번의 복소 입력 샘플이 필요하다. 앞선 윈도우에서 계산된 $\dot{D}_{n-1}^{M/8}(k)$ 이 저장되기 위해서는 $M/2$ 사이즈의 추가 메모리가 필요할 수 있다.
- [0089] S240 단계에서, 슬라이딩 이산 푸리에 변환 장치는 수학식 12에 따라 $X_{n-M/4}(k)$ 와 $D_n^{M/4}(k)$ 을 이용하여 $X_n(k)$ 를 계산한다. 이 단계에서는, 오직 $2M$ 개의 실수 부분만이 필요하다.
- [0090] 도 3은 알고리즘별 M 값에 따른 계산 복잡도를 나타낸 도면이다. 도 3에서, R_u 및 R_A 는 각각 실수 곱셈의 횟수와 실수부의 개수를 나타낸다.
- [0091] 도 3을 참조하면, 특히, $M=16$ 일 때, 본 발명의 실시예에 따른 gSDFT 알고리즘의 곱셈 횟수는 rSDFT 및 mSDFT 알고리즘에 대하여 각각 50% 및 70%로 감소한다. $M < 2^{13}$ 일 때, gSDFT 알고리즘의 곱셈 횟수는 기존의 mSDFT 알고리즘보다 더 작다. 그러므로, 수치 에러가 축적되지 말아야 하고, 곱셈 횟수가 가장 중요한 고려사항이 되는 어플리케이션에 대하여, M 이 2^{13} 미만일 때 gSDFT 알고리즘이 사용될 수 있고, 다른 경우에는 mSDFT 알고리즘이 사

용될 수 있다.

[0092] 본 발명의 실시예에 따른 gSDFT 알고리즘의 주요 특징은 다음과 같다.

[0093] 1) M이 2^{13} 미만일 때, gSDFT 알고리즘의 곱셈 횟수는 기존 방법들보다 더 작다.

[0094] 2) 수학적 12의 반복식은 무조건 안정성이 보장되며, 수치 에러가 축적되지 않는다.

[0095] 3) gSDFT 알고리즘은 모든 시간 인덱스에서 수학적 1의 DFT 출력과 수학적으로 동등한 출력 신호를 산출한다.

[0096] 4) SUVT는 SDFT의 반복 계산과 독립적으로 계산된다. 이 분리 특징은 하드웨어 구현에 대해서 큰 장점이 된다.

[0097] 도 4는 본 발명의 실시예에 따른 슬라이딩 이산 푸리에 변환 방법의 시뮬레이션 결과를 나타낸 도면이다.

[0098] 본 발명의 실시예에 따른 슬라이딩 이산 푸리에 변환 방법은 표준 편차가 0인 제로 평균 가우시안 노이즈(zero-mean Gaussian noise)를 테스트 신호로 이용하여 효율이 조사되었다. 시뮬레이션은 64비트 고정밀도 연산(64-bit double-precision arithmetic)으로 수행되었으며, M은 16으로 설정되었다. 시뮬레이션에서, rSDFT, mSDFT 및 gSDFT의 수치 에러는 각각 수학적 4, 5 및 12를 반복적으로 계산하여 생성되었다. 그리고, 이 반복 계산은 수치 에러를 축적하기 위하여 10^6 번 반복되었으며, 각 시간 인덱스에서 수치 에러가 계산되었다. 모든 알고리즘은 고효율의 ANSI-C 코드를 이용하여 실행되었으며, 성능은 Intel i5 3.4 GHz CPU와 8 GB RAM에서 평가되었다.

[0099] 도 4는 모든 알고리즘의 수치 에러 측정 결과를 보여준다. 시간 인덱스 n에서 수치 에러 E_n 은 다음의 수학적 식으로 계산될 수 있다.

수학적 식 18

$$E_n = \sum_{k=0}^{M-1} |X_n^{\text{DFT}}(k) - X_n^{\text{Algorithm}}(k)|$$

[0100] 여기서, $X_n^{\text{DFT}}(k)$ 는 수학적 1의 표준 DFT의 k번째 값을 나타내고, $X_n^{\text{Algorithm}}(k)$ 는 각 알고리즘의 k번째 값을 나타낸다.

[0102] 도 4에 도시된 바와 같이, mSDFT 및 gSDFT 알고리즘은 rSDFT 알고리즘과 비교하여 수치 에러가 상당히 감소한다. 여기서, rSDFT 알고리즘의 댐핑 지수 r은 0.99999로 설정되었다. 그리고, 도 4에는, 에러 축적 과정 이후에 64 시간 인덱스 동안의 평균 수치 에러 $\bar{E}$ 가 도시되어 있다. 도 4에 도시된 바와 같이, rSDFT, mSDFT 및 gSDFT의 $\bar{E}$ 는 각각 9.72×10^{-3} , 7.41×10^{-11} , 및 7.17×10^{-11} 이다. 이를 통해, gSDFT의 수치 에러가 mSDFT보다 더 작음을 알 수 있다. 그러므로, 본 발명의 실시예에 따른 슬라이딩 이산 푸리에 변환 방법이 다른 알고리즘을 일관되게 능가한다.

[0103] 도 5는 본 발명의 실시예에 따른 슬라이딩 이산 푸리에 변환 장치의 구성을 개략적으로 예시한 도면이다.

[0104] 도 5를 참조하면, 본 발명의 실시예에 따른 슬라이딩 이산 푸리에 변환 장치는 프로세서(510), 메모리(520), 입력부(530) 및 디스플레이부(540)를 포함한다.

[0105] 프로세서(510)는 메모리(520)에 저장된 처리 명령어를 실행시키는 CPU 또는 반도체 소자일 수 있다.

[0106] 메모리(520)는 다양한 유형의 휘발성 또는 비휘발성 기억 매체를 포함할 수 있다. 예를 들어, 메모리(520)는 ROM, RAM 등을 포함할 수 있다.

[0107] 예를 들어, 메모리(520)는 본 발명의 실시예에 따른 슬라이딩 이산 푸리에 변환 방법을 수행하는 명령어들을 저장할 수 있다.

[0108] 입력부(530)는 슬라이딩 이산 푸리에 변환 장치를 제어하거나 슬라이딩 이산 푸리에 변환 장치에 설치된 어플리케이션을 실행하기 위한 다양한 명령, 정보 등을 사용자로부터 입력받기 위한 수단이다. 입력부(530)는 적어도

하나의 키버튼을 포함할 수 있으며, 디스플레이부(540)와 일체로 형성된 터치입력모듈을 포함할 수도 있다.

[0109] 디스플레이부(540)는 슬라이딩 이산 푸리에 변환 장치로 입력되거나 슬라이딩 이산 푸리에 변환 장치에 저장된 다양한 데이터를 시각 정보의 형태로 표출하기 위한 수단이다. 예를 들어, 디스플레이부(540)는 액정화면(LCD)일 수 있다.

[0110] 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.

[0111] 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.

[0112] 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

[0113] 510: 프로세서

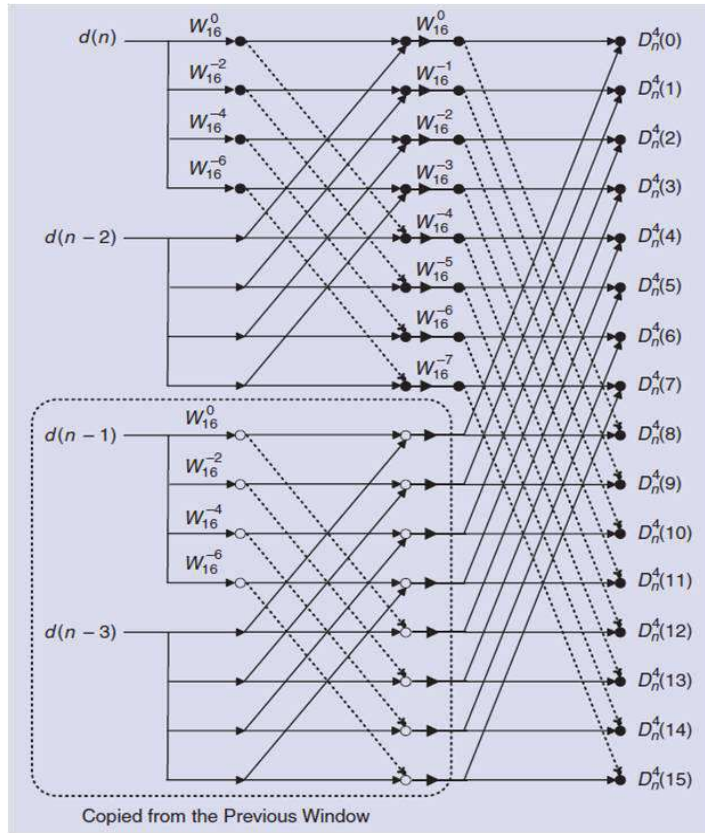
520: 메모리

530: 입력부

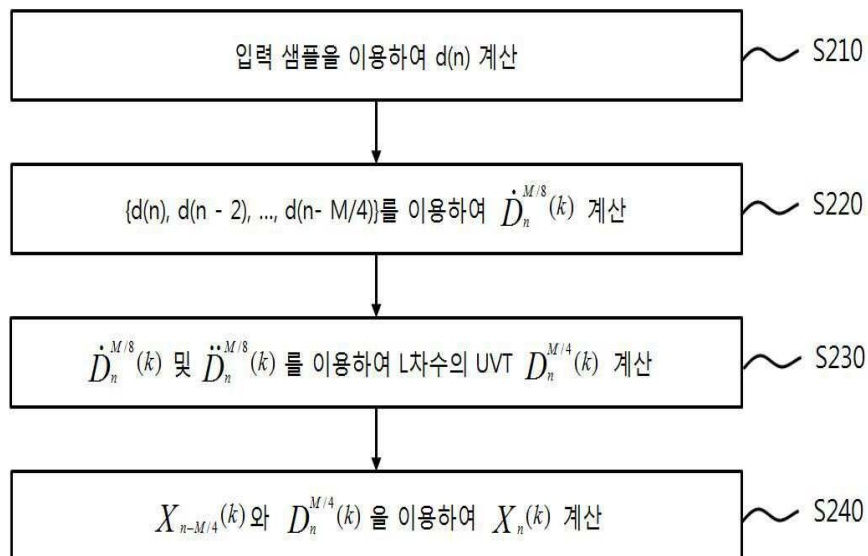
540: 디스플레이부

도면

도면1



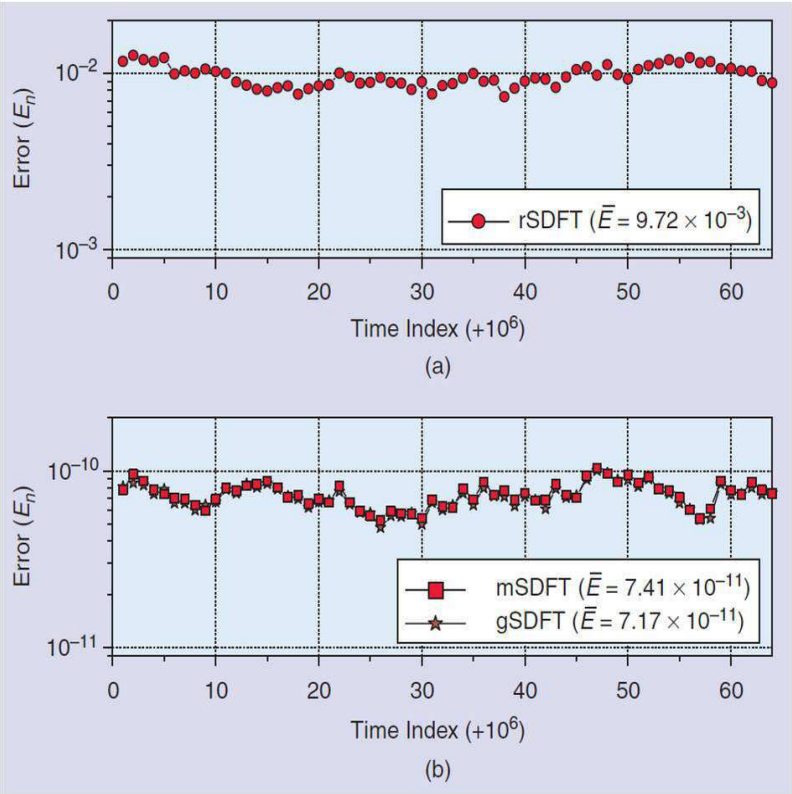
도면2



도면3

ALGORITHM	OPERATION		WINDOW SIZE				
	[6]						
FFT	R_M	4	16	64	256	1,024	4,096
	R_M	16	128	768	4,096	20,480	98,304
	R_A	24	192	1,152	6,144	30,720	147,456
ISDFT	R_M	24	96	384	1,536	6,144	24,576
	R_A	24	96	384	1,536	6,144	24,576
	R_M	48	192	768	3,072	12,288	49,152
mSDFT	R_A	40	160	640	2,560	10,240	40,960
	R_M	0	48	320	1,792	9,216	45,056
	R_A	10	42	610	3,202	15,874	75,778
							M
							$2M\log_2 M$
							$3M\log_2 M$
							$6M$
							$6M$
							$12M$
							$10M$
							$M(\log_2 M - 1)$
							$(M/2)(3\log_2 M + 1) + 2$

도면4



도면5

