일반논문-09-14-5-01

고속 컨벌루션을 위한 새로운 중첩보류기법

국 중 갑[a)‡], 조 남 익[a)]

A New Overlap Save Algorithm for Fast Convolution

Jung Gap Kuk[a)‡] and Nam Ik Cho[a)]

요 약

가장 많이 사용되는 변환영역 컨벌루션 알고리즘인 중첩보류기법의 경우 데이터를 M개 단위로 처리하고자 할 때 현재 M개의 데이터를 앞의 M개의 데이터와 연결시킨 $2M$개의 데이터에 대하여 길이 $2M$ FFT와 주파수 영역 곱셈을 수행하고 뒤의 M개의 데이터를 취함으로써 선형 컨벌루션 결과를 얻는다. 본 논문에서는 새로운 변환을 제시하고 이를 이용하여 M개의 데이터에 대하여 길이 M의 변환을 수행하면 되는 새로운 컨벌루션 알고리즘을 소개한다. 새로 제안된 변환은 M개의 곱셈과 고속 푸리에 변환의 계산으로 이루어지므로 기존의 FFT 라이브러리 또는 하드웨어를 그대로 사용할 수 있다. 또한 기존의 중첩보류기법에 비하여 약간의 계산량 감소가 있고 다루어야 하는 데이터의 길이가 반이므로 데이터 이동 및 기타 처리에서도 이득이 있어서 전체적인 CPU 시간이 줄어든다.

Abstract

The most widely used block convolution method is the overlap save algorithm (OSA), where a block of M data to be convolved with a filter is concatenated with the previous block and $2M$-point FFT and multiplications are performed for this overlapped block. By discarding half of the results, we obtain linear convolution results from the circular convolution. This paper proposes a new transform which reduces the block size to only M for the block convolution. The proposed transform can be implemented as the M multiplications followed by M-point FFT. Hence, existing efficient FFT libraries and hardware can be exploited for the implementation of proposed method. Since the required transform size is half that of the conventional method, the overall computational complexity is reduced. Also the reduced transform size results in the reduction of data access time and cash miss-hit ratio, and thus the overall CPU time is reduced. Experiments show that the proposed method requires less computation time than the conventional OSA.

Keywords: FFT, Overlap save algorithm, QDFT, DFT, Block convolution

a) 서울대학교 전기컴퓨터공학부, 뉴미디어 통신공동 연구소
Department of Electrical Engineering and Computer Sciences, Seoul National University, Institute of New Media & Communications
‡ 교신저자 : 국중갑(jg-kuk@ispl.snu.ac.kr)
※ 본 연구는 방위산업청과 국방과학연구소의 기초연구과제(UD080015FD)의 지원에 의하여 수행된 결과의 일부입니다. 지원에 감사드립니다.
접수일(2009년3월24일),수정일(1차:2009년8월12일,2차:9월14일),게재확정일(2009년9월16일)

Ⅰ. 서 론

디지털 필터링과 상관 값의 계산 등을 위한 컨벌루션을 수행하여 고속 이산 푸리에 변환(FFT)를 기반으로 한 중첩보류방법 (Overlap Save Algorithm) 또는 중첩합산방법 (Overlap Add Algorithm)을 이용함으로써 고속 연산이 가능하다는 것이 널리 알려져 있다[1]. 이와 같은 FFT이외에도

Number Theoretic Transform[2]과 Hartley transform[3] 등을 이용한 고속 컨벌루션이 가능하지만 여러 범용 CPU와 DSP 등에 특화되고 편하게 사용할 수 있는 FFT 라이브러리와 하드웨어들이 매우 많이 개발되어 다른 방법들은 거의 사용되지 않고 있다. 따라서 본 연구에서도 FFT를 기반으로 보다 고속으로 컨벌루션을 구현할 수 있는 방법을 개발하고 기존의 FFT 기반의 방법과 비교한다.

중첩보류기법의 핵심은 부분적인 블록 데이터들의 주파수 영역 계산으로 얻어지는 순환 컨벌루션의 결과들을 이용하여 선형컨벌루션 결과를 얻는 것이다. 여기서, 주파수 영역의 계산을 위해서 FFT가 사용되므로 FFT의 성능이 전체적인 계산시간에 큰 영향을 미친다. 본 논문에서는 보다 효율적인 중첩 컨벌루션의 구현을 위해서 FFT와 기타 연산에 필요한 덧셈, 곱셈 등의 계산량 감소 뿐만 아니라 시스템 측면도 고려한다. 기존의 중첩보류기법에서는 인접하는 블록들이 서로 겹치도록 데이터를 나누며, 중복된 부분들에 의해 생성된 결과는 최종적으로 버려지는데 이 과정에서 계산량의 낭비가 발생한다. 그러므로 본 논문에서는 계산량의 낭비를 없애기 위하여 데이터를 겹치지 않게 나누어도 선형 컨벌루션 결과를 얻을 수 있도록 해 주는 새로운 변환을 제안하고, 이 변환 영역에서의 곱셈이 선형 컨벌루션과 어떤 관련이 있는지 규명한다. 제안된 변환은 기존 방법 절반 길이의 FFT를 이용하여 구현이 가능하므로 블록 길이에 비례한 계산상의 이득을 얻을 수 있을 뿐 아니라 기존의 FFT 라이브러리 또는 하드웨어를 그대로 사용할 수 있어 고속 구현이 가능하다. 이와 더불어 처리해야 할 데이터의 길이가 짧으므로 메모리 접근 시간과 캐시 미스 (cash miss) 발생도 줄어서 전체적인 CPU 시간을 줄인다.

본 논문의 2장에서는 제안하는 방법을 설명하기 위해 필요한 여러 용어들을 정의하고 기존의 중첩보류기법을 요약한다. 정의된 용어를 바탕으로 3장에서는 절반 길이의 중첩보류기법을 가능케 하는 새 변환을 제안하고 변환 영역에서의 곱셈 계산과 선형 컨벌루션의 관계를 규명한다. 제안된 변환과 이에 기반한 중첩보류기법의 계산량은 4장에서 분석하고, 다양한 길이의 필터에 대한 블록 컨벌루션에 대하여 제안하는 방법과 기존의 방법을 PC 상에서 실험하여 그 성능을 검증한다.

II. 용어 정의 및 기존의 중첩 보류 기법 요약

1. 용어 정의

기존의 중첩 보류 기법을 요약하기 앞서 본 논문에서 사용하는 용어들을 먼저 정의한다.

L : 필터 길이

M : 중첩 보류 기법에서 새로이 입력되는 데이터의 길이, $M \geq L$

$N=2M$, 중첩 보류 기법에서 처리되는 블록의 길이

h_n : 필터 계수

s_n : 입력 데이터

$r_n = s_n \otimes h_n$, 선형 컨벌루션 결과

$f=[h_0,h_1,...,h_{L-1},0,...0]^T$, 길이 M이 되도록 0이 채워진 필터 벡터

0_M: 길이 M의 0으로 이루어진 벡터

$g=[f^T\ 0_M^T]^T$

x_p: 중첩 보류 기법에서 이전 블록과 겹치는 데이터, 길이는 M

x_c: 중첩 보류 기법에서 새로 들어온 데이터, 길이는 M

$x=[x_p^T\ x_c^T]^T$

$y=[y_0,y_1,...,y_{N-1}]^T$: 순환 컨벌루션 결과

D_K : $K\times K$ DFT 행렬 $(D_K(n,k)=W_K^{nk})$

Θ_K : $K\times K$ Odd DFT(ODFT) 행렬 $(\Theta_K(n,k)=W_K^{n(k+1/2)})$

Q_K: $K\times K$ Quarter DFT(QDFT) 행렬 $(Q_K(n,k)=W_K^{n(k+3/4)})$

$X^d=D_N x,\ X^\theta=\Theta_N x,\ X^q=Q_N x,$

$G^d=D_N g,\ G^\theta=\Theta_N g,\ G^q=Q_N g,$

$F^d=D_M f,\ F^\theta=\Theta_M f,\ F^q=Q_M f,$

$X_p^d=D_M x_p,\ X_p^\theta=\Theta_M x_p,\ X_p^q=Q_M x_p,$

$X_c^d=D_M x_c,\ X_c^\theta=\Theta_M x_c,\ X_c^q=Q_M x_c,$

2. 기존의 중첩 보류 기법 요약

정의된 용어를 바탕으로 기존의 중첩 보류 기법을 요약하면 다음과 같다

1) $G^d = D_N g$

2.1) $x_c \rightarrow x_p;\ [s_n,\ldots,s_{n+M-1}] \rightarrow x_c,\ x = [x_p^T\ x_c^T]^T$

2.2) $X^d = D_N x,\ Y = G^d X^d$

2.3) $y = D_N^{-1} Y,\ \{y_{M+1},\ldots,y_{N-1}\} \rightarrow \{r_n,\ldots,r_{n+M-1}\}$

먼저 과정 1)처럼 필터의 주파수 계수 G^d를 구한다. 그리고 새로 입력된 데이터를 이용하여 블록 x를 생성하고 주파수 계수 X^d를 구한 후 필터와 입력 블록의 주파수 계수를 각 원소끼리 곱한 후 역 푸리에 변환을 수행한다. 역 변환 후 순환 컨벌루션의 특성상 뒤의 절반의 결과를 취함으로써 선형 컨벌루션 결과를 얻는다. 그리고 더 이상 새로 입력되는 데이터가 없을 때까지 과정 2)를 반복한다.

III. 제안하는 알고리즘

제안하는 알고리즘은 기존의 중첩 보류 기법에서 통합된 프레임워크 안에서 계산되던 이전 블록 x_p와 현재 블록 x_c에 대한 계산을 따로 수행하는데 핵심이 있다. 기존의 중첩 보류 기법에서 함께 처리되던 블록 x_p와 블록 x_c에 대한 계산을 분리하기 위하여 먼저 x_p와 x_c로 이루어진 데이터 블록 x에 대해서 Decimation in frequency FFT 방법을 이용하여 푸리에 계수 X^d가 어떤 형태인지를 살펴본다. x의 Decimation in frequency FFT 결과는 X^d의 홀수 계수 X_o와 짝수 계수 X_e로 분리되어 구해진다.

$$\begin{bmatrix} X_e \\ X_o \end{bmatrix} = \begin{bmatrix} D_M & D_M \\ \Theta_M & -\Theta_M \end{bmatrix} \begin{bmatrix} x_p \\ x_c \end{bmatrix} \tag{1}$$

마찬가지 방법으로 필터 블록 g에 대해서 푸리에 계수 G^d를 구한다. 이 때 필터 블록의 앞 절반은 f이고 뒤 절반은 0_M이므로, 다음과 같이 필터 블록의 짝수 푸리에 계수는 F^d, 홀수 푸리에 계수는 F^θ와 같다.

$$\begin{bmatrix} G_e \\ G_o \end{bmatrix} = \begin{bmatrix} D_M & D_M \\ \Theta_M & -\Theta_M \end{bmatrix} \begin{bmatrix} f \\ 0_M \end{bmatrix} = \begin{bmatrix} F^d \\ F^\theta \end{bmatrix}$$

주어진 데이터와 필터의 주파수 계수를 이용하여 다음 식에 나타낸 바와 같이, 주파수 성분끼리 곱하고 역 푸리에 변환을 취하면 순환 컨벌루션의 결과 y를 얻는다.

$$y = \frac{1}{2} \begin{bmatrix} D_M^{-1} & \Theta_M^{-1} \\ D_M^{-1} & -\Theta_M^{-1} \end{bmatrix} \begin{bmatrix} F^d \otimes X_e \\ F^\theta \otimes X_o \end{bmatrix} \tag{2}$$

그리고 식 (1)의 X_e와 X_o를 식 (2)에 대입하고 이를 이전 데이터 블록 x_p와 현재 데이터 블록 x_c에 대한 계산으로 나누면 순환 컨벌루션은 최종적으로 식(3)과 같다.

$$\begin{aligned} y = &\frac{1}{2} \begin{bmatrix} D_M^{-1}(F^d \otimes X_p^d) + \Theta_M^{-1}(F^\theta \otimes X_p^\theta) \\ D_M^{-1}(F^d \otimes X_p^d) - \Theta_M^{-1}(F^\theta \otimes X_p^\theta) \end{bmatrix} + \\ &\frac{1}{2} \begin{bmatrix} D_M^{-1}(F^d \otimes X_c^d) - \Theta_M^{-1}(F^\theta \otimes X_c^\theta) \\ D_M^{-1}(F^d \otimes X_c^d) + \Theta_M^{-1}(F^\theta \otimes X_c^\theta) \end{bmatrix} \end{aligned} \tag{3}$$

그리고 위의 순환 컨벌루션 결과로부터 앞의 절반을 버리고 뒤의 절반을 취하면 식 (4)와 같이 원하는 선형 컨벌루션 결과를 얻는다.

$$\begin{aligned} &\frac{1}{2}\left[D_M^{-1}(F^d \otimes X_p^d) - \Theta_M^{-1}(F^\theta \otimes X_p^\theta)\right] + \\ &\frac{1}{2}\left[D_M^{-1}(F^d \otimes X_c^d) + \Theta_M^{-1}(F^\theta \otimes X_c^\theta)\right] \end{aligned} \tag{4}$$

식 (4)는 기본적으로 두 계산으로 이루어져있는데 일반화하여 표현하면 각각 $D_M^{-1}(F^d \otimes X^d)$와 $\Theta_M^{-1}(F^\theta \otimes X^\theta)$이다. 이 두 계산을 시간 영역에서 살펴보면 $D_M^{-1}(F^d \otimes X^d)$는 일반적인 DFT를 이용한 순환 컨벌루션의 결과로서

$$y_n = \sum_{k=0}^{n} x_k f_{n-k} + \sum_{k=n+1}^{M-1} x_k f_{n-k+M}$$

와 같고 $\Theta_M^{-1}(F^\theta \otimes X^\theta)$는 ODFT를 이용한 변형된 순환 컨벌루션 결과로서

$$y_n = \sum_{k=0}^{n} x_k f_{n-k} - \sum_{k=n+1}^{M-1} x_k f_{n-k+M}$$

와 같다. 그러므로 식 (4)에서 이전 블록에 대한 계산

$$\frac{1}{2}[D_M^{-1}(F^d \otimes X_p^d) - \Theta_M^{-1}(F^\theta \otimes X_p^\theta)]$$

는 시간 영역에서 $\sum_{k=n+1}^{M-1} x_k f_{n-k+M}$, 현재 블록에 대한 계산

$$\frac{1}{2}[D_M^{-1}(F^d \otimes X_c^d) + \Theta_M^{-1}(F^\theta \otimes X_c^\theta)]$$

은 $\sum_{k=0}^{n} x_k f_{n-k}$이다. 이 두 계산을 원초적인 방법으로 구하려 한다면 앞에서 언급한 두 계산을 두 번의 길이 M 블록 컨벌루션 $D_M^{-1}(F^d \otimes X_c^d)$와 $\Theta_M^{-1}(F^\theta \otimes X_c^\theta)$을 수행함으로써 구할 수 있다. 이전 블록에 대한 계산 $D_M^{-1}(F^d \otimes X_p^d)$와 $\Theta_M^{-1}(F^\theta \otimes X_p^\theta)$은 현재 블록과 한 블록만큼의 지연만 있으므로 추가 메모리를 사용하여 현재 블록에 대한 계산을 미리 저장하면 추가 계산량 없이 구할 수 있다는 것이다. 본 논문의 핵심은 이 계산들을 제안하는 사분 이산 푸리에 변환(Quarter-DFT)를 이용하여 한 번에 구하는 데 있다. 먼저 QDFT의 변환, 역변환은 다음과 같이 정의되고

$$\text{변환} \quad : X_k^q = \sum_{n=0}^{M-1} x_n W_M^{n(k+3/4)}$$

$$\text{역변환} : x_n = \frac{1}{M}\sum_{k=0}^{M-1} X_k^q W_M^{-n(k+3/4)}$$

QDFT를 이용한 블록 컨벌루션 $Q_M^{-1}(F^q \otimes X^q)$는 시간 영역에서

$$y_n = \sum_{k=0}^{n} x_k f_{n-k} + j\sum_{k=n+1}^{M-1} x_k f_{n-k+M}$$

와 같다. QDFT의 변환, 역변환관계와 컨벌루션 특성은 부록 1에서 증명된다. 그러므로 식 (4)를 QDFT를 이용하여 표현하면 다음과 같이 한 번의 블록 컨벌루션으로 표현할 수 있다.

$$y = [\Im(Q_M^{-1}(F^q \otimes X_p^q))] + [\Re(Q_M^{-1}(F^q \otimes X_c^q))] \tag{5}$$

식 (5)에서 보면 이전 블록에 대한 계산은 블록 컨벌루션의 허수 부분 $\Im(Q_M^{-1}(F^q \otimes X_p^q))$, 현재 블록에 대한 계산은 실수 부분 $\Re(Q_M^{-1}(F^q \otimes X_c^q))$으로 계산된다. 이때 이전 블록에 대한 계산인 허수부분은 현재 블록에 대한 계산 후 허수부분만 메모리에 저장함으로써 미리 계산될 수 있다. 그러므로 제안하는 길이 M의 QDFT에 기반한 중첩 보류 기법을 요약하면 다음과 같다.

1) $F^q = Q_M f$

2.1) $[s_n,\ldots,s_{n+M-1}] \rightarrow x_c$

2.2) $X_c^q = Q_M x_c$, $Y = F^q X_c^q$

2.3) $y = Q_M^{-1} Y$, $z_i + \Re(y) \rightarrow \{r_n,\ldots,r_{n+M-1}\}$, $\Im(y) \rightarrow z_i$

먼저 길이 M의 필터에 대해 QDFT 계수 F^q를 구한다. 그리고 새로 입력된 데이터에 대해 QDFT 계수 X_c^q를 구하여 필터의 QDFT 계수와 원소끼리 곱한 후 역 변환을 수행한다. 역 변환 후 z_i에 저장되어 있는 값과 역 변환 결과의 실수부분을 더하여 출력으로 내보내고, 허수 부분은 z_i에 저장한다. 그리고 더 이상 새로 입력되는 데이터가 없을 때까지 과정 2)를 되풀이 한다.

Ⅳ. 계산량 분석 및 PC기반 실험결과

1. 계산량 분석

중첩 보류 기법은 주파수 변환, 곱셈 그리고 역변환의 과정으로 이루어져 있으므로 먼저 FFT와 QDFT의 계산량을 분석하고 이를 바탕으로 중첩 보류 기법의 계산량을 분석한다. 그리고 편의상 본 논문에서는 복소수 곱셈과 복소수

덧셈을 단순히 곱셈과 덧셈으로 지칭한다.

주파수 변환을 위해서는 split-radix FFT[4]와 W 변환[5]을 이용한 FFT 등 여러 방법이 있으나 본 논문에서는 가장 기본적인 radix-2 방법과 실제 DSP와 PC 라이브러리에서 많이 사용되는 radix-4에 대하여 계산량을 분석한다. 먼저 FFT부터 살펴보면, 입력이 실수인 길이 N의 FFT는 일반적으로 길이 $N/2$의 복소수 입력의 FFT를 이용하여 보다 고속으로 구현할 수 있는데 그 과정은 부록 2에 설명한다. 부록 2에 따라 정리하면 실수 입력의 길이 N FFT의 계산량은 표 1과 같다.

표 1. FFT의 계산량
Table 1. Number of arithmetic operations of FFT

	곱셈	덧셈
radix-2	$\frac{N}{4}log_2\frac{N}{2}+N$	$\frac{N}{2}log_2\frac{N}{2}+\frac{N}{2}$
radix-4	$\frac{3N}{16}log_2\frac{N}{2}+N$	$\frac{N}{2}log_2\frac{N}{2}+\frac{N}{2}$

그리고 QDFT는 FFT와 달리 입력이 실수라 하더라도 고속으로 처리할 수 있는 알고리즘이 없다. 그러나 QDFT 또한 radix-2와 radix-4구조로 구현할 수 있으므로 계산량은 복소수 입력의 FFT와 같고 표 2와 같이 정리할 수 있다. QDFT의 radix 구조는 부록 3에 설명한다.

표 2. QDFT의 계산량
Table 2. Number of arithmetic operations of QDFT

	곱셈	덧셈
radix-2	$\frac{N}{2}log_2\frac{N}{2}$	$N\log_2 N$
radix-4	$\frac{3N}{8}log_2 N$	$N\log_2 N$

중첩 보류 기법은 주파수 변환, 곱셈 그리고 역변환의 과정으로 이루어져 있으므로 표 1과 표 2의 결과를 이용하여 기존의 중첩 보류 기법과 제안하는 방법의 계산량을 구하면 표 3과 같다. 이 때 기존의 방법의 경우 변환 길이는 N이지만 필터와 데이터의 주파수 계수가 앞 절반과 뒤 절반이 complex conjugate의 관계에 있으므로 $N/2+1$번의 곱셈만 수행하면 된다. 그리고 새로 제안하는 방법에서는 메모리에 저장된 값을 더해야 하므로 실수 덧셈 M번이 계산량에 추가된다.

표 3을 보면 radix-2와 radix-4의 두 경우 모두 제안하는 방법이 곱셈은 $2N+1$만큼 덧셈은 $2N$만큼 이득이 있음을 알 수 있다.

2. PC 기반 실험

본 절에서는 알고리즘의 성능을 검증하기 위하여 PC상에서 필터의 길이를 변환해 가며 실험을 수행한다. FFT를 위해 최적화되어 구현된 라이브러리들은 많이 있으나 QDFT가 최적화되어 구현된 라이브러리는 없으므로 공정한 비교를 위해 Ⅳ.1절에서처럼 QDFT의 radix-2나 radix-4 구조를 이용하여 구현하지 않고 기존의 라이브러리를 사용할 수 있도록 입력신호 x_n에 $W_M^{3n/4}$를 곱한 후 FFT를 수행한다. 이 경우 계산량이 Ⅳ.2 절과 비교하여 입력 신호를 가공하는데 필요한 곱셈 수만큼 즉 변환, 역 변환 시 필요한 $2M$의 곱셈 수만큼 증가한다.

표 3. 기존의 중첩 보류 기법과 제안하는 방법의 계산량
Table 3. Number of arithmetic operations of conventional and proposed OSAs

		곱셈	덧셈
기존방법	radix-2	$\frac{N}{2}log_2 N+2N+1$	$N\log_2 N+N$
	radix-4	$\frac{3}{8}N\log_2 N+2N+\frac{N}{8}+1$	$N\log_2 N+N$
제안하는방법	radix-2	$M\log_2 M+M(=\frac{N}{2}log_2 N)$	$2M\log_2 M+M'\ (\approx N\log_2 N-N)$
	radix-4	$\frac{3}{4}M\log_2 M+M(=\frac{3}{8}N\log_2 N+\frac{N}{8})$	$2M\log_2 M+M'\ (\approx N\log_2 N-N)$

$$X_k^q = \sum_{n=0}^{M-1} x_n W_M^{3n/4} W_M^{nk}$$

실험을 위해 Pentium 4, 3㎓의 PC에서 인텔의 IPP 라이브러리[7]에서 제공하는 FFT를 이용하여 필터 길이 M에 따라 기존의 중첩 보류 기법과 제안하는 방법을 비교하였다.

표 4. PC에서 구현 시 수행 시간(μs)
Table 4. elapsed time of conventional OSA and proposed OSA on PC(μs)

M	기존의 방법(c_1)	제안하는 방법(c_2)	계산량이득 $((c_1-c_2)/c_1\,\%)$
512	16.39	13.81	15.74
1024	34.74	31.11	10.45
2048	73.80	66.87	9.4
4096	152.84	136.34	10.79
8192	416.91	359.67	13.73
16384	1174.06	992.72	15.45
32768	2962.35	2303.87	22.23
65536	7515.18	6078.57	19.12

실제 PC에서 제안하는 방법과 기존의 방법의 수행시간을 비교한 결과, 표 4에서처럼 제안하는 방법이 기존의 방법보다 더 좋은 성능을 나타냄을 알 수 있었다.

V. 결 론

본 논문에서는 기존의 중첩 보류 기법에서 블록을 겹치게 나누지 않더라도 선형 컨벌루션 결과를 구할 수 있는 방법을 제안하였다. 제안하는 방법에서는 겹친 데이터와 새로 들어온 데이터에 대한 계산을 분리하여 분석하였고 사분이산 푸리에 변환을 이용한 블록 컨벌루션이 시간 영역에서 어떠한 특성을 갖는지 규명하여 이를 바탕으로 새로 들어온 데이터에 대한 계산만으로 선형 컨벌루션 결과를 얻을 수 있도록 하였다. 그리고 계산량을 분석함으로써 제안하는 방법이 이론적으로 갖는 이득을 구하였고, PC에서 최적화된 FFT 라이브러리를 공통적으로 사용하여 기존의 방법과 제안된 방법을 구현해봄으로써 실제 환경에서도 제안된 방법이 유리함을 확인하였다.

참 고 문 헌

[1] A. V. Oppenheim, R. W. Schafer and J. R. Buck, Discrete-Time Signal Processing, Upper Saddle River, New Jersey: Prentice Hall, 1998.

[2] R. C. Agarwal, C. S. Burrus, "Number theoretic transforms to implement fast digital convolution," Proceedings of IEEE, Vol. 63, No. 4, pp. 550 - 560, Apr. 1978.

[3] J. L. Vernet "Improved fourier and hartley transform algorithms: application to cyclic convolution of real data," IEEE trans. on Acoustics, Speech and Signal Processing, vol. ASSP-35, No. 6, June 1987.

[4] P. Duhamel, "Implementation of "Split-radix" FFT algorithms for complex, real, and real-symmetric data," IEEE Trans. on Acoust., Speech and Signal Processing, vol. 34, no.2, pp.285-295, Apr. 1986.

[5] Z. Wang, "Fast algorithms for the discrete W transform and for the discrete fourier transform," IEEE trans. on Acoustics, Speech and Signal Processing, vol. 32, pp. 803 - 816, Aug. 1984.

[6] Intel Performance Libraries. Intel Integrated Performance Primitives website. http://www3.intel.com/cd/software/products/asmona/eng/perflib/ 302910.htm

[7] R. Matusiak, "Implementing fast fourier transform algorithms of real-valued sequences with the TMS320 DSP family" Application Report of Texas Instruments, 1997

부록 1 QDFT 변환, 역변환 관계와 컨벌루션 특성 증명

1.1 QDFT 변환, 역변환 관계

QDFT의 변환, 역변환 관계는 다음과 같이 정의 된다.

$$\text{변환} \quad : X_k^q = \sum_{n=0}^{M-1} x_n W_M^{n(k+3/4)}$$

$$\text{역변환}: x_n = \frac{1}{M}\sum_{k=0}^{M-1} X_k^q W_M^{-n(k+3/4)}$$

변환, 역변환 관계를 증명하기 위하여 변환식을 역변환식에 대입하면

$$x_n = \frac{1}{M}\sum_{k=0}^{M-1}\sum_{m=0}^{M-1} x_m W_M^{m(k+3/4)} W_M^{-n(k+3/4)}$$

와 같고 이는 다음과 같이 증명된다.

$$\begin{aligned}
&\frac{1}{M}\sum_{k=0}^{M-1}\sum_{m=0}^{M-1} x_m W_M^{m(k+3/4)} W_M^{-n(k+3/4)} \\
&= \frac{1}{M}\sum_{k=0}^{M-1}\sum_{m=0}^{M-1} x_m W_M^{(m-n)(k+3/4)} \\
&= \sum_{m=0}^{M-1} x_m \frac{1}{M}\sum_{k=0}^{M-1} W_M^{(m-n)(k+3/4)} \\
&= \sum_{m=0}^{M-1} x_m W_M^{(m-n)(3/4)} \frac{1}{M}\sum_{k=0}^{M-1} W_M^{(m-n)k}
\end{aligned}$$

여기서 $\frac{1}{M}\sum_{k=0}^{M-1} W_M^{(m-n)k}$은 $m-n$이 M의 배수인 경우에 1이고 나머지 경우는 0이므로

$$\begin{aligned}&\sum_{m=0}^{M-1} x_m W_M^{(m-n)(3/4)} \frac{1}{M}\sum_{k=0}^{M-1} W_M^{(m-n)k} \\ &= \sum_{m=0}^{M-1} x_m W_M^{(m-n)(3/4)} \sum_{p=-\infty}^{\infty} \delta_{m,n+pM} \\ &= \sum_{p=-\infty}^{\infty} x_{n+pM} W_M^{pM(3/4)}\end{aligned}$$

와 같다. 이 때 $p=0$인 경우에만 x_{n+pM}이 0이 아니므로 결국 x_n과 같다.

1.2 QDFT의 컨벌루션 특성

QDFT 주파수 영역에서 X_k^q와 F_k^q의 곱셈은 시간 영역에서

$$y_n = \sum_{k=0}^{n} x_k f_{n-k} + j\sum_{k=n+1}^{M-1} x_k f_{n-k+M}$$

와 같다. 이러한 QDFT의 컨벌루션 특성은 다음과 같이 증명된다.

$$\begin{aligned}y_n &= \frac{1}{M}\sum_{k=0}^{M-1} X_k^q F_k^q W_N^{-n(k+3/4)} \\ &= \frac{1}{M}\sum_{k=0}^{M-1}\sum_{l=0}^{M-1} x_l W_M^{l(k+3/4)} \sum_{m=0}^{M-1} f_m W_M^{m(k+3/4)} W_M^{-n(k+3/4)} \\ &= \sum_{l=0}^{M-1} x_l \sum_{m=0}^{M-1} f_m \frac{1}{M}\sum_{k=0}^{M-1} W_M^{l(k+3/4)} W_M^{m(k+3/4)} W_M^{-n(k+3/4)} \\ &= \sum_{l=0}^{M-1} x_l \sum_{m=0}^{M-1} f_m \frac{1}{M}\sum_{k=0}^{M-1} W_M^{(l+m-n)(k+3/4)} \\ &= \sum_{l=0}^{M-1} x_l \sum_{m=0}^{M-1} f_m W_M^{(l+m-n)(3/4)} \frac{1}{M}\sum_{k=0}^{M-1} W_M^{(l+m-n)k}\end{aligned}$$

여기서 $\frac{1}{M}\sum_{k=0}^{M-1} W_M^{(l+m-n)k}$은 $l+m-n$이 M의 배수인 경우에 1이고 나머지 경우에는 0이므로

$$\begin{aligned}y_n &= \sum_{l=0}^{M-1} x_l \sum_{m=0}^{M-1} f_m W_M^{(l+m-n)(3/4)} \sum_{p=-\infty}^{\infty} \delta_{m,n-l+pM} \\ &= \sum_{l=0}^{M-1} x_l \sum_{p=-\infty}^{\infty} j^p f_{n-l+pM}\end{aligned}$$

이 되고 f_{n-l+pM}은 $p=0,1$인 경우에만 0이 아닌 값을 가지므로 최종적으로

$$y_n = \sum_{k=0}^{n} x_k f_{n-k} + j\sum_{k=n+1}^{M-1} x_k f_{n-k+M}$$

이 된다.

부록 2 실수 입력의 길이 N FFT의 고속 구현[6]

입력 데이터가 x라고 할 때 고속 구현 과정을 요약하면 다음과 같다.

1) $u_n = x_{2n} + jx_{2n+1},\ 0 \le n \le N/2-1,\ U = D_{N/2}u$
2) $X_k = U_k A_k + U_{N/2-k}^* B_k,\ 0 \le k \le N/2$
   , $U_{N/2} = U_0,\ A_k = (1-jW_N^k),\ B_k = (1+jW_N^k)$
3) $X_k = X_{N-k}^*,\ N/2+1 \le k < N-1$

먼저 과정 1)과 같이 입력 데이터 x의 짝수 부분과 홀수 부분을 이용하여 u를 만든 후 푸리에 계수를 구한다. 그리고 과정 2)와 같이 U를 이용하여 x의 푸리에 계수를 구한다. 이 때 범위가 $0 \le k \le N/2$ 으로 정의되어 있으므로 나머지 값들은 과정 3)과 같이 complex conjugate를 해서 구하면 된다.

그러므로 실수 입력의 길이 N FFT는 간단히 데이터를 조작하는 과정을 제외하면 허수입력의 길이 $N/2$ FFT에 N 번의 곱셈과 $N/2$ 번의 덧셈만 추가하여 계산될 수 있다. $N/2$이 4의 승수라는 가정 하에, 허수입력의 길이 $N/2$ FFT의 경우 radix-2는 곱셈이 $\frac{N}{4}\log_2\frac{N}{2}$, 덧셈이 $\frac{N}{2}\log_2\frac{N}{2}$ 그리고 radix-4는 곱셈이 $\frac{3}{16}N\log_2\frac{N}{2}$, 덧셈이 $\frac{N}{2}\log_2\frac{N}{2}$이므로 표 1과 같은 결과를 얻을 수 있다.

부록 3 QDFT의 radix 구조

먼저 QDFT를 다음과 같이 변환 T_s로 일반화한다.

$$\begin{aligned}&T_s : x \to X^s \\ &X^s = \sum_{n=0}^{N-1} x_n W_N^{n(k+s)}\end{aligned}$$

그리고 위의 T_s 변환의 radix-2 구조를 구하기 위하여 주파수 계수를 짝수와 홀수로 나누면 다음과 같이 변환 T_s가 $N/2$ 길이의 두 변환, $T_{s/2}$와 $T_{(s+1)/2}$로 나누어짐을 알 수 있다.

$$\begin{aligned}X_{2m}^s &= \sum_{n=0}^{N/2-1} (x_n + W_2^s x_{n+N/2}) W_{N/2}^{n(m+s/2)} \\ X_{2m+1}^s &= \sum_{n=0}^{N/2-1} (x_n - W_2^s x_{n+N/2}) W_{N/2}^{n(m+(s+1)/2)}\end{aligned}$$

이 식을 버터플라이 구조로 표현하면 다음과 같다. 그림 2에서 화살표가 그려진 선은 곱하기 -1을 의미한다.

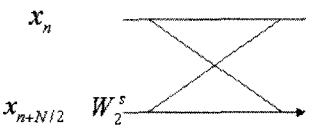

그림 2. QDFT의 radix-2 구조
Fig 2. radix-2 structure for QDFT

$$X_{4m}^{s} = \sum_{n=0}^{N/4-1} (x_n + W_4^s x_{n+N/4} + W_4^{2s} x_{n+2N/4} + W_4^{3s} x_{n+3N/4}) W_{N/4}^{n(m+s/4)}$$
$$X_{4m+1}^{s} = \sum_{n=0}^{N/4-1} (x_n - jW_4^s x_{n+N/4} - W_4^{2s} x_{n+2N/4} + jW_4^{3s} x_{n+3N/4}) W_{N/4}^{n(m+(s+1)/4)}$$
$$X_{4m+2}^{s} = \sum_{n=0}^{N/4-1} (x_n - W_4^s x_{n+N/4} + W_4^{2s} x_{n+2N/4} - W_4^{3s} x_{n+3N/4}) W_{N/4}^{n(m+(s+2)/4)}$$
$$X_{4m+3}^{s} = \sum_{n=0}^{N/4-1} (x_n + jW_4^s x_{n+N/4} - W_4^{2s} x_{n+2N/4} - jW_4^{3s} x_{n+3N/4}) W_{N/4}^{n(m+(s+3)/4)}$$

다음으로 T_s 변환의 radix-4 구조를 구하기 위하여 주파수 계수를 $4m+k\,(k=0,1,2,3)$ 에 따라 나누면 다음과 같이 변환 T_s가 $N/4$ 길이의 네 변환 $T_{s/4}$, $T_{(s+1)/4}$, $T_{(s+2)/4}$ 그리고 $T_{(s+3)/4}$으로 나누어짐을 알 수 있다.

이 식을 버터플라이 구조로 표현하면 그림 2와 같다. 그림 2에서 실선은 곱하기 1, 화살표가 그려진 선은 곱하기 -1 그리고 점선은 곱하기 j를 의미한다.

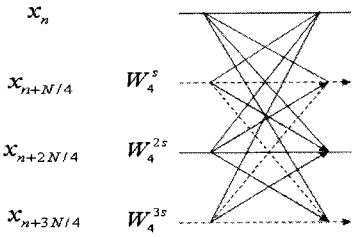

그림 3. QDFT의 radix-4 구조
Fig 3. radix-4 structure for QDFT

저 자 소 개

국 중 갑

- 2004년 : 서울대학교 전기공학부, 학사
- 2006년 : 서울대학교 전기공학부, 석사
- 2006년 ~ 현재 : 서울대학교 전기공학부, 박사과정
- 주관심분야 : 정지영상 및 동영상처리

조 남 익

- 1982년 ~ 1986년 : 서울대학교 제어계측공학과, 학사
- 1986년 ~ 1988년 : 서울대학교 제어계측공학과, 석사
- 1988년 ~ 1992년 : 서울대학교 제어계측공학과, 박사
- 1991년 ~ 1994년 : 서울대학교 제어계측신기술연구센터, 연구원
- 1994년 ~ 1998년 : 서울시립대학교 전자전기공학부, 조교수
- 1998년 ~ 현재 : 서울대학교 전기공학부 교수
- 주관심분야 : 적응신호처리, 정지영상 및 동영상처리, 음성신호처리