논문 98-23-10-24 한국통신학회논문지 '98-10 Vol.23 No.10

# 주파수 스펙트럼과 양자화를 이용한 효율적인 블록 FIR 필터 디자인

정회원 김 용 배*, 정 진 균*, 정 항 근**

# Efficient Block FIR Filter Design Using Frequency Spectrum and Quantization

Yong-Bae Kim*, Jin-Gyun Chung*, Hang-Geun Jeong** *Regular Members*

요 약

FIR 디지털 필터의 속도 증가와 전력 감소를 위하여 블록 FIR 필터 구조를 사용할 수 있다. 그러나 기존의 블록 필터 구현방법은 원하는 블록 사이즈에 따라 원래의 필터를 복제하는 방식이므로 블록 사이즈가 커짐에 따라 하드웨어 오버헤드가 선형적으로 증가하게 되는 단점이 있다. 최근에 블록 FIR 필터의 하드웨어 비용을 감소시키기 위한 연구들이 제안되었다. 본 논문에서는 블록 FIR 필터 구현시 주파수 스펙트럼 특성을 고려하고 새로운 블록 필터 양자화 알고리듬을 사용함으로써 하드웨어 비용을 약 5%-15% 정도 더 감소시킬 수 있음을 보인다.

ABSTRACT

Block FIR digital filters can be used either for high-speed or low-power (with reduced supply voltage) applications. Traditional block filter implementations casuse linear increase in the hardware cost with respect to the block size. Recently, an efficient block FIR filter implementation technique requiring a less-than linear increase in the hardware cost was proposed. In this paper, it is shown that the hardware cost can be reduced further (5%-15%) by exploiting the frequency spectrum characteristics. Also, an efficient block filter quantization algorithm is proposed.

## I. 서 론

디지털 필터는 비디오 및 이미지 프로세싱뿐만 아니라 이동통신 응용분야에 이르기까지 매우 다양한 DSP 시스템에 사용된다. 또한 디지털 필터는 많은 하드웨어를 필요로 하므로 디지털 필터의 효율적인 고속, 저전력 VLSI 구현은 전체 시스템의 성능 향상에 크게 기여한다고 할 수 있다. 디지털 필터는 FIR 필터와 IIR 필터로 나눌 수 있으며 FIR 필터는 정확한 선형 위상 특성을 얻을 수 있기 때문에 널리 쓰이나 동일한 주파수 특성을 위해 IIR 필터보다 훨씬 높은 필터 차수가 요구된다는 단점이 있다.

최근 높은 동작 주파수와 저전력이 요구되는 DSP 응용분야가 늘어나고 있는데 이러한 조건을 만족시키기 위한 VLSI 구현 방법으로 파이프라인과 블록 프로세싱이 널리 사용된다[1]. 블록 프로세싱은 파이프라인보다 큰 하드웨어 오버헤드를 필요로 하지만 파이프라인에 의해서는 더 이상 속도 증가가 불가능한 경우 사용된다. 기존의 블록 프로세싱은 원하는 블록 사이즈($L$)에 따라 원래의 필터 블록을 복제하는 방식이므로 원래의 필터 블록의 면적이 $A$라 하면 $L\times A$의 면적을 갖게 되어 하드웨어 오버헤드는 선형적으로 증가하게 된다. 따라서 효율적인 블록 FIR 디지털 필터의 구현을 위해서는 기존의 FIR 필터 블록 프로세싱보다 적은 하드웨어 오버헤드를 갖는 알고리듬의 개발이 필요하다.

* 전북대학교 정보통신공학과(jgchung@moak.chonbuk.ac.kr), ** 전북대학교 전자공학과(hgjeong@moak.chonbuk.ac.kr)
논문번호 : 98011-0106, 접수일자 : 1998년 1월 6일
※본 연구는 산업자원부와 정보통신부 및 과학기술부에서 시행하는 주문형반도체 개발사업의 지원을 받아 수행되었습니다.

지금까지 FIR 필터의 효율적인 구현에 대한 연구는 매우 많이 행해져 왔다. 그러나 연구의 분야가 필터 디자인[2], 및 필터 계수의 양자화[3], CAD 툴을 이용한 필터회로의 합성방법[4, 5]등에 한정되어 있었고, FIR 필터의 블록 프로세싱을 위한 하드웨어 복잡도 감소에 대한 연구는 매우 적었다. 최근에 FFA(Fast FIR Algorithm)를 이용하여 블록 FIR 필터 구현시 하드웨어 비용을 감소시키는 방법들이 제안되었다[6-9].

본 논문에서는 주파수 특성을 블록 프로세싱과 연관 시킴으로써 하드웨어 오버헤드를 기존의 알고리듬들에 비하여 훨씬 감소 시키는 새로운 FIR 필터의 블록 프로세싱 방법을 제안한다. 또한 하드웨어의 효율적인 구현을 위하여 블록 필터에 적합한 양자화 알고리듬을 제안한다. II절에서 FFA에 대해 간단하게 요약하고 주파수 스펙트럼에 따른 FFA의 효율적인 사용에 대해 논의한다. III절에서 블록 FIR 필터에 적합한 양자화 방법을 제안한다. IV절에서는 sub-structure sharing을 소개하며, V절에서는 블록 필터의 디자인 예를 보인다.

## II. FFA(Fast FIR Algorithms)

임의의 ($N$-1) 차 FIR 필터의 출력 $y_n$은 다음과 같이 표현될 수 있다,

$$y_n = \sum_{i=0}^{N-1} h_i x_{n-i}, \quad n = 0, 1, 2, \cdots, \infty. \tag{1}$$

여기서 $\{x_i\}$은 무한 길이의 입력이고, $\{h_i\}$은 길이가 $N$인 FIR 필터 계수이다. 이때 기존의 $L$-블록 FIR 필터의 폴리페이즈 표현은

$$\sum_{i=0}^{L-1} Y_i(z^L) z^{-i} = \sum_{j=0}^{L-1} H_j(z^L) z^{-j} \sum_{k=0}^{L-1} X_k(z^L) z^{-k} \tag{2}$$

와 같으며 여기서

$$Y_i(z) = \sum_{m=0}^{\infty} z^{-m} y_{mL+i},$$
$$H_i(z) = \sum_{m=0}^{N/L-1} z^{-m} h_{mL+i},$$
$$X_i(z) = \sum_{m=0}^{\infty} z^{-m} x_{mL+i}, \quad i = 0, 1, \cdots, L-1 \tag{3}$$

이다 [7]. 식 (3)을 행렬식으로 표현하면

$$\mathbf{Y} = \mathbf{HX},$$
$$\mathbf{Y} = [Y_0 \;\; Y_1 \;\; \cdots \;\; Y_{L-1}]^T,$$
$$\mathbf{X} = [X_0 \;\; X_1 \;\; \cdots \;\; X_{L-1}]^T,$$
$$\mathbf{H} = \begin{bmatrix} H_0 & z^{-L}H_{L-1} & z^{-L}H_{L-2} & \cdots & z^{-L}H_1 \\ H_1 & H_0 & z^{-L}H_{L-1} & \cdots & z^{-L}H_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ H_{L-1} & H_{L-2} & H_{L-3} & \cdots & H_0 \end{bmatrix} \tag{4}$$

이며, $\mathbf{H}$는 pseudo-circulant행렬이다. 식 (4)는 길이가 $N/L$인 $L^2$개의 FIR 필터를 이용하여 블록 FIR 필터를 구현할 수 있음을 보인다.

### 1. 2-by-2 ($L=2$) FFA

L=2인 경우 식 (2)로 부터

$$Y_0 + z^{-1} Y_1 = (H_0 + z^{-1} H_1)(X_0 + z^{-1} X_1),$$
$$= H_0 X_0 + z^{-1}(H_0 X_1 + H_1 X_0) + z^{-2} H_1 X_1 \tag{5}$$

을 얻으며 이것은

$$Y_0 = H_0 X_0 + z^{-2} H_1 X_1,$$
$$Y_1 = H_0 X_1 + H_1 X_0 \tag{6}$$

을 의미한다. 식 (6)의 직접적인 구현은 그림 1과 같다. 이 구조는 길이가 $N/2$인 4개의 FIR 필터와 2개의 후프로세싱 덧셈으로 구성되며 $2N$개의 곱셈기와 $2N$-2개의 덧셈기를 필요로 한다.

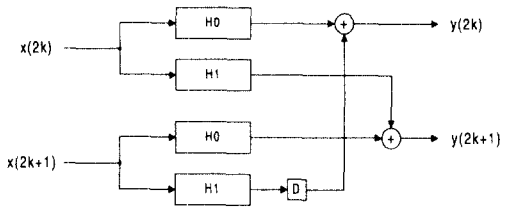



그림 1. 기존의 (2-by-2) 블록 FIR 필터.

식 (5)를 변환하여 다음과 같은 (2-by-2) FFA0 (FFA-type 0)를 얻을 수 있다 [7]:

$$Y_0 = H_0X_0 + z^{-2}H_1X_1,$$
$$Y_1 = H_{0+1}X_{0+1} - H_0X_0 - H_1X_1. \quad (7)$$

여기서, $H_{i+j} = H_i + H_j$이고, $X_{i+j} = X_i + X_j$이다. 식 (7)의 구현은 그림 2와 같다. 이 구조는 길이가 $N/2$인 3개의 FIR 필터와 4개의 전/후 프로세싱 덧셈기를 사용하며 $3N/2$개의 곱셈기와 $3(N/2-1)+4$개의 덧셈기를 필요로 한다. 그림 2의 FFA0는 그림 1의 기존 구조에 비해 25%의 하드웨어 이득을 얻을 수 있다.

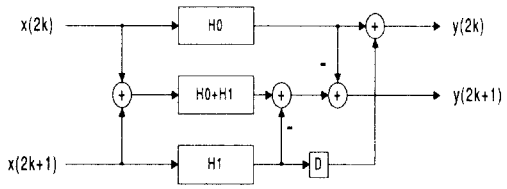



그림 2. FFA0을 이용한 (2-by-2) 블록 FIR 필터.

식 (7)을 변환하여 다음의 FFA1 (FFA-type 1)을 얻을 수 있다.

$$Y_0 = H_0X_0 + z^{-2}H_1X_1,$$
$$Y_1 = -H_{0-1}X_{0-1} + H_0X_0 + H_1X_1. \quad (8)$$

FFA1에 의해 얻어진 구조는 그림 3과 같다.

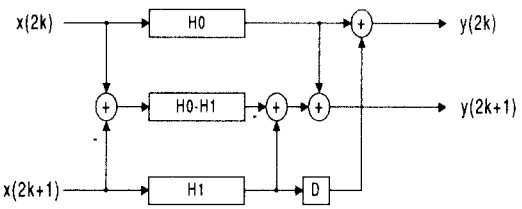



그림 3. FFA1을 이용한 (2-by-2) 블록 FIR 필터.

FFA0에서는 $H_{0+1}$이 사용되고 FFA1에서는 $H_{0-1}$이 사용된다는 점과 고정된 필터 계수를 갖는 FIR 필터를 구현할 때 하드웨어 복잡도는 필터 계수의 nonzero 비트수에 비례한다는 점을 이용하여 FFA0 또는 FFA1 중 적절한 선택을 함으로써 블록 필터의 하드웨어를 더욱 감소 시킬 수 있다. 주어진 임펄스 응답의 부호가 협대역 저역통과 필터의 경우처럼 자주 변하지 않을 때 $H_0+H_1$의 계수의 크기는 $H_0-H_1$의 계수의 크기보다 더 크게 되므로 $H_0+H_1$이 $H_0-H_1$보다 계수에서 더 많은 nonzero 비트수를 가진다. (V절의 예 참조) 주어진 임펄스 응답의 부호가 광대역 저역통과 필터의 경우처럼 자주 변하면 $H_0-H_1$이 $H_0+H_1$보다 계수에서 더 많은 nonzero 비트수를 가진다. 그러므로 구현하고자 하는 FIR 필터의 주파수 스펙트럼 특성을 고려하여 FFA0 또는 FFA1을 선택함으로써 블록 FIR 필터 구현시 하드웨어 면적과 전력소모를 줄일 수 있다.

### 2. 3-by-3 (L = 3) FFA

(3-by-3) FFA은 블록 사이즈가 3인 블록 필터 구조이다. L=3인 경우 식 (2)로부터,

$$Y_0 = H_0X_0 + z^{-3}(H_1X_2 - H_2X_1),$$
$$Y_1 = (H_0X_1 - H_1X_0) + z^{-3}H_2X_2,$$
$$Y_2 = H_0X_2 + H_1X_1 - H_2X_0 \quad (9)$$

이다. 식 (9)의 직접 구현에는 길이가 $N/3$인 9개의 필터와 6개의 후 프로세싱 덧셈기가 사용되며 $3N$개의 곱셈기와 $3N-3$개의 덧셈기를 필요로 한다. (2-by-2) FFA0와 같이 식 (9)를 변환하므로써 다음의 (3-by-3) FFA0을 얻을 수 있다:

$$Y_0 = H_0X_0 - z^{-3}H_2X_2 + z^{-3}[(H_{1+2}X_{1+2} - H_1X_1)],$$
$$Y_1 = [H_{0+1}X_{0+1} - H_1X_1)] - [H_0X_0 - z^{-3}H_2X_2],$$
$$Y_2 = H_{0+1+2}X_{0+1+2} - [H_{0+1}X_{0+1} - H_1X_1] - [H_{1+2}X_{1+2} - H_1X_1]. \quad (10)$$

그림 4는 (3-by-3) FFA0로 부터 얻어진 필터 구조이다. 이 구조는 길이가 $N/3$인 6개의 필터와 10개의 전/후 프로세싱 덧셈기를 사용하며 $6(N/3)$개의 곱셈기와 $6(N/3-1)+10$개의 덧셈기가 필요하다.

(3-by-3) FFA0구조는 기존의 구조에 비해 약 33%의 하드웨어 이득을 얻을 수 있다. (3-by-3) FFA1은 식 (9)를 다음과 같이 변환하여 얻을 수 있다:

$$Y_0 = H_0X_0 + z^{-3}H_2X_2 - z^{-3}[(H_{2-1}X_{2-1} - H_1X_1)],$$
$$Y_1 = -[H_{0-1}X_{0-1} - H_1X_1)] + [H_0X_0 + z^{-3}H_2X_2],$$
$$Y_2 = H_{0-1+2}X_{0-1+2} - [H_{0-1}X_{0-1} - H_1X_1] - [H_{2-1}X_{2-1} - H_1X_1]. \quad (11)$$

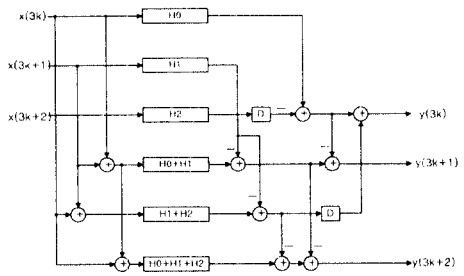



그림 4. FFA0을 이용한 (3-by-3) 블록 FIR 필터.

그림 5는 (3-by-3) FFA1으로 부터 얻어진 필터 구조이다.

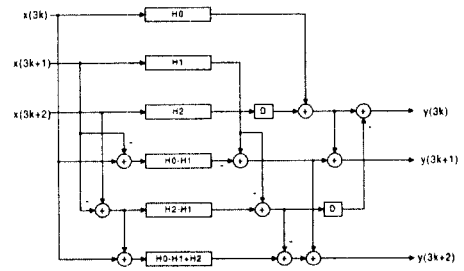



그림 5. FFA1을 이용한 (3-by-3) 블록 FIR 필터.

### 3. FFA의 직렬형

블록 레벨이 3보다 큰 블록 필터는 (2-by-2) 블록 필터와 (3-by-3) 블록 필터를 이용하여 직렬(cascade) 형으로 구현할 수 있다. 예를 들어 (4-by-4) 블록 필터는 (2-by-2) 블록 필터의 직렬형태로 구현할 수 있다. (4-by-4) 블록 필터를 구현하기 위한 폴리페이즈 표현식은 다음과 같다.

$$\begin{aligned} Y &= Y_0 + z^{-1} Y_1 + z^{-2} Y_2 + z^{-3} Y_3, \\ Y &= (X_0 + z^{-1} X_1 + z^{-2} X_2 + z^{-3} X_3) \\ &\quad (H_0 + z^{-1} H_1 + z^{-2} H_2 + z^{-3} H_3). \end{aligned} \tag{12}$$

식 (12)는 다음과 같은 변수를 정의하여

$$\begin{aligned} X_0' &= X_0 + z^{-2} X_2, \quad & X_1' &= X_1 + z^{-2} X_3, \\ H_0' &= H_0 + z^{-2} H_2, \quad & H_1' &= H_1 + z^{-2} H_3 \end{aligned} \tag{13}$$

다음 식과 같이 나타낼 수 있다.

$$Y = (X_0' + z^{-1} X_1')(H_0' + z^{-1} H_1'). \tag{14}$$

식 (14)로 부터의 (4-by-4) 블록 필터 디자인 과정은 다음과 같다.

단계 1 : 식 (14)에 (2-by-2) FFA 적용

$$\begin{aligned} Y = {} & X_0' H_0' + z^{-1}[(X_0' + X_1')(H_0' + H_1') \\ & - X_0' H_0' - X_1' H_1'] + z^{-2} X_1' H_1'. \end{aligned} \tag{15}$$

단계 2 : $X_0' H_0'$에 (2-by-2) FFA 적용

$$\begin{aligned} X_0' H_0' &= (X_0 + z^{-2} X_2)(H_0 + z^{-2} H_2), \\ X_0' H_0' &= X_0 H_0 + z^{-2}[(X_0 + X_2)(H_0 + H_2) \\ &\quad - X_0 H_0 - X_2 H_2] + z^{-4} X_2 H_2. \end{aligned} \tag{16}$$

단계 3 : $X_1' H_1'$에 (2-by-2) FFA 적용

$$\begin{aligned} X_1' H_1' &= (X_1 + z^{-2} X_3)(H_1 + z^{-2} H_3), \\ X_1' H_1' &= X_1 H_1 + z^{-2}[(X_1 + X_3)(H_1 + H_3) \\ &\quad - X_1 H_1 - X_3 H_3] + z^{-4} X_3 H_3. \end{aligned} \tag{17}$$

단계 4 : $(X_0' + X_1')(H_0' + H_1')$에 (2-by-2) FFA 적용

$$\begin{aligned} (X_0' + X_1')(H_0' + H_1') &= (X_0 + X_1)(H_0 + H_1) \\ &\quad + z^2[(X_0 + X_1 + X_2 + X_3)(H_0 + H_1 + H_2 + H_3)] \\ &\quad - (X_0 + X_1)(H_0 + H_1) - (X_2 + X_3)(H_2 + H_3)] \\ &\quad + z^{-4}(X_2 + X_3)(H_2 + H_3). \end{aligned} \tag{18}$$

그림 6의 (4-by-4) FFA0은 $9N/4$개의 곱셈기와 $9N/4+11$개의 덧셈기를 필요로 한다. 기존의 방법에 비해 약 44%의 하드웨어 이득을 얻을 수 있다.

(4-by-4) FFA1은 FFA0와 같은 방법으로 그림 7과 같이 구현될 수 있다. 블록 사이즈가 5이상인 경우에도 (4-by-4) 블록 필터에서와 같은 방법으로 (2-by-2) FFA와 (3-by-3) FFA를 사용하여 구현할 수 있다.

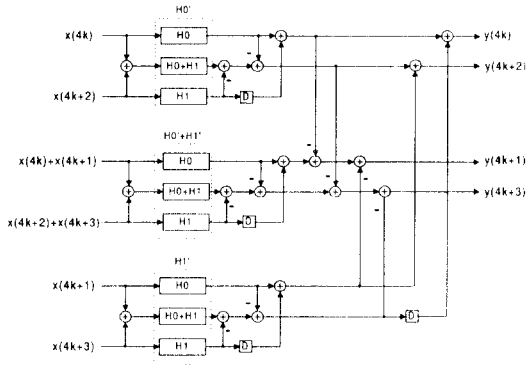



그림 6. FFA0을 이용한 (4-by-4) 블록 FIR 필터.

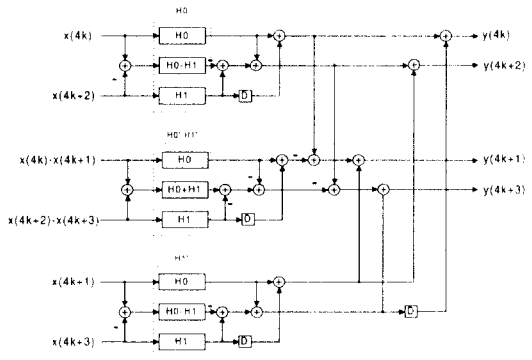



그림 7. FFA1을 이용한 (4-by-4) 블록 FIR 필터.

## III. Look-Ahead MAD 양자화

일반적으로 필터 계수가 양자화 과정을 거치기전에 먼저 스케일링 된다면 더 좋은 주파수 특성이 얻어짐이 알려져 있다[2, 10, 11]. NUS 알고리듬[11]에서는 이상적인 필터 계수중 가장 큰 계수의 절대값이 1이 되도록 정규화한 다음 정규화된 이상적인 필터 계수에 VSF (Variable Scale Factor)를 곱한후 양자화한다. VSF는 $2^{W}$(W = 워드길이)을 스텝 사이즈로 해서 0.4375에서 1.13까지 변화하는 스케일링 상수이다. 임의의 VSF가 곱해진 필터계수를 양자화하기 위해 먼저 양자화된 계수들을 모두 0으로 초기화한 후 이상적인 필터 계수에 VSF가 곱해진 값과 양자화된 계수와의 절대값의 차이가 가장 큰 계수에 SPT(Signed power-of-two) 항을 먼저 할당한다. NUS 알고리듬은 원하는 수의 SPT 항들이 할당될 때까지 또는 원하는 NPR(Normalized Peak Ripple) 이 만족될 때까지 SPT 항들을 반복적으로 할당한다. 0.4375에서 1.13까지의 모든 VSF에 대해 이러한 과정을 반복하고, 최소의 NPR을 제공하는 양자화된 필터 계수를 선택한다.

블록 FIR 필터에서는 FFA의 적용에 의해 얻어진 각 필터 블록에 대해 통과대역/저지대역의 NPR이 정의되지 않으므로 통과대역/저지대역 리플(ripple)은 가장 좋은 양자화된 필터를 선택하기 위한 기준으로 사용될 수 없다. [6]에서는 이상적인 필터와 양자화된 블록 필터에 대한 효율적인 선택 기준으로 두 필터의 주파수 응답의 최대 차이인 MAD(Maximum Absolute Difference)를 제시하였다.

양자화된 필터가 구현될 때 필터계수의 정규화 및 VSF에 의한 스케일링 효과를 보상하기 위해 PPSF (Post-Processing Scale Factor)를 필터의 출력단에 사용하며 다음과 같이 계산된다.

$$\text{PPSF} = \text{Max}[\text{Abs}(\text{Ideal Filter Coeffs.})] \;/\; \text{VSF}. \quad (19)$$

블록 필터 구현에서 블록 사이즈 $L=2^{a}\times 3^{b}$일 때 사용되는 PPSF 수는 $3^{a}\times 6^{b}$이므로 PPSF에 의해 큰 하드웨어 오버헤드를 가지게 된다. [6]에서는 이러한 하드웨어 오버헤드를 줄이기 위하여 PPSF를 다음과 같은 값으로 제한하였다: {0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875, 1}. 원래의 PPSF를 이 값들 중 가장 근접한 값을 갖는 새로운 PPSF로 대치시킨 후 다음 3단계를 통하여 최종의 양자화된 필터 계수를 얻는다.

단계 1 : PPSF를 포함한 계수 결정 :

$$\text{PPSF를 포함한 계수} = \text{양자화된 계수} \times \text{PPSF}. \quad (20)$$

단계 2 : 새로운 PPSF를 가진 변화된 계수 결정 :

$$\text{변화된 계수} = \text{PPSF를 포함한 계수} \;/\; \text{새로운 PPSF}. \quad (21)$$

단계 3 : 변화된 계수 양자화.

그러나, 위의 과정은 단계 3에서 양자화된 계수에 다시 양자화를 수행하므로써 새로운 PPSF에 대해 최적의 양자화된 계수를 보장할 수 없다. 이러한 문제를 해결하기 위해 본 논문에서는 Look-ahead MAD 양자화 알고리듬을 제안한다. 제안된 알고리듬에서는 양자화 과정을 수행하기 전에 주어진 VSF에 대한 PPSF

를 계산하고, 계산된 PPSF의 nonzero 비트수가 요구되는 비트수보다 작으면 정규화된 계수는 VSF에 의해 스케일되고 스케일된 계수는 양자화된다. 만일 주어진 VSF에 의한 PPSF의 비트수가 요구되는 비트수보다 많으면 그 다음 스텝의 VSF에 대해 이 과정을 반복한다.

Look-Ahead MAD Quantization

```
For each filter section in the block FIR filter
{
  Normalize the set of filter coefficients so that the
      magnitude of the largest coefficient is 1 :
  For VSF = Lower scale: Step size: Upper scale,
  {
    Compute PPSF by (19);
    Convert PPSF into Canonic Signed Digit form;
    IF (No. of nonzero bits in PPSF) < prespecified value,
    {
      Scale normalized coefficients with VSF;
      Quantize the scaled coefficients using SPT term
          allocation scheme in NUS algorithm;
      Calculate MAD between the frequency responses
          of the ideal and Quantized filters;
    }
  }
  Choose the quantized coefficients that leads to the
      minimum MAD;
}
```

**Example 1**: 다음 계수를 가진 이상적인 필터를 고려하자[6]: 이상적인 계수 = {−0.0648 0.1404 0.4328 −0.0818 0.0391}. 이 계수는 워드길이가 7 비트일 때 [6]의 Scalable MAD 양자화 알고리듬에 의해 다음과 같은 값으로 양자화된다: PPSF = 0.625, 양자화 계수 = {−0.109375 0.203125 0.6875 −0.140625 0.046875}. 계산된 MAD 값은 0.0360125이다. 동일한 PPSF 값에 대해 본 논문에서 제안한 알고리듬에 의해 양자화 되었을 때는 다음과 같다: PPSF = 0.625, 양자화 계수 = {−0.109375 0.21875 0.6875 −0.140625 0.0625}. 제안한 방법에 의한 MAD값은 0.01648125이고 [6]에 의한 MAD값의 약 45%에 불과하다. 그림 8에서는 제안한 방법에 의한 주파수 응답이 [6]의 방법에 비해 이상적인 값에 더 접근하고 있음을 보인다.

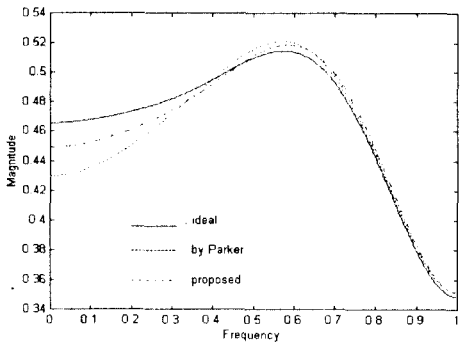



그림 8. Example 1의 주파수 응답.

## IV. Sub-Structure Sharing

Sub-structure sharing은 필터 계수의 하드웨어 구현 시 공통 부분의 활용을 극대화 하기 위한 방법이다. 본 논문에서는 FIR 필터 블록의 sub-structure sharing을 수행하기 위해 전치 직접형 구조를 사용하였으며 sub-expression sharing과 인접 계수 sharing의 두 가지 sub-structure sharing방법을 적용하였다 [6, 12]. [12]의 sub-expression sharing은 두 가지 과정으로 나눌 수 있다. 첫번째 과정에서는 CSD표현에서 가장 흔한 3개의 수[101, $\bar{1}01$, 1]에 대한 회로를 생성하고, 두번째 과정에서는 3개의 수의 적당한 합성에 의해 모든 필터 계수를 구현한다. 그림 9는 sub-expression sharing을 사용하여 계수가 [0.$\bar{1}0\bar{1}010\bar{1}0$, 0.10101010]인 2-탭 필터를 구현한 그림이다. Sharing을 사용하지 않을 때 이 필터의 구현에 7개의 덧셈기가 필요한 반면 sharing을 사용하여 5개의 덧셈기만으로 2-탭 필터를 구현할 수 있음을 알 수 있다.

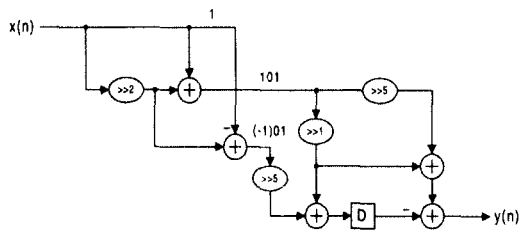



그림 9. Sub-expression sharing을 사용한 FIR 필터 구조.

인접 계수 sharing에서는 인터커넥션 비용을 줄이

기 위해 인접 필터 계수들만을 쌍으로 묶고, 그 인접 계수간에 sharing될 수 있는 하드웨어 요소를 결정한다. 그림 10은 계수가 $[0.00\bar{1}000\bar{1}0,\ 1.00\bar{1}000\bar{1}]$인 2-탭 필터에 대해 인접 계수 sharing을 적용한 예이다. 이 예에서는 sharing을 사용하지 않을 때와 비교하여 1개의 덧셈기를 줄일 수 있음을 보여준다. 본 논문에서는 블록 필터에 대한 하드웨어 비용을 줄이기 위해 위의 두 가지 sharing을 모두 사용하였다.

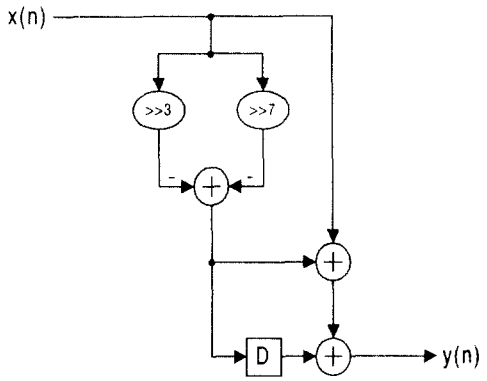



그림 10. 인접 계수 sharing을 사용한 FIR 필터 구조.

## V. 디자인 예

**Example 2** : 필터 차수 = 24, 통과대역 = $0\sim0.3\pi$, 저지대역 = $0.5\pi\sim\pi$인 저역통과 필터를 고려하자[6]. 그림 11은 이상적인 임펄스 응답이고, 그림 12에서는 이상적인 주파수 응답과 제안한 방법에 의한 주파수 응답을 비교한다. 표 1은 [6]과 본 논문에서 제안한 방법에 대해 블록 사이즈가 2, 3, 4, 6, 8 12인 경우의 하드웨어 비용을 비교한다. 본 논문에서 제안한 방법에 의해 각 블록 사이즈에 대해 약 5%~15%의 덧셈기 수를 절감할 수 있음을 알 수 있다.

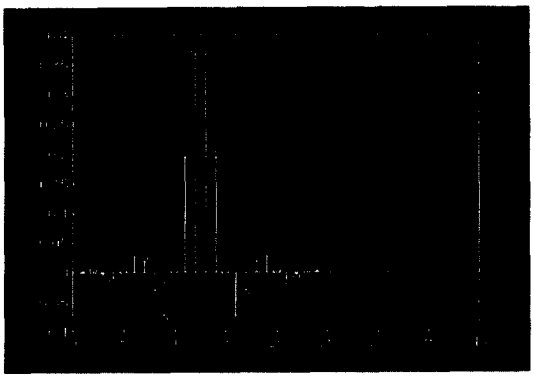


그림 11. Example 2의 이상적인 임펄스 응답.

그림 12. Example 2의 주파수 응답.

표 1. Example 2의 필터 특성

| | | Proposed method | Parker's method |
|---|---|---|---|
| L = 2 | # of adders for H | 65 | 73 |
| | # of SSS adders | 44 | 55 |
| | # of PPSF adders | 3 | 1 |
| | S. Ripple(dB) | 45.1507 | 45.1606 |
| | P. Ripple(dB) | 0.0455 | 0.0589 |
| L = 3 | # of adders for H | 95 | 114 |
| | # of SSS adders | 74 | 90 |
| | # of PPSF adders | 4 | 5 |
| | S. Ripple(dB) | 42.852 | 42.7698 |
| | P. Ripple(dB) | 0.0547 | 0.0554 |
| L = 4 | # of adders for H | 114 | 125 |
| | # of SSS adders | 88 | 99 |
| | # of PPSF adders | 9 | 8 |
| | S. Ripple(dB) | 40.6154 | 40.2009 |
| | P. Ripple(dB) | 0.0407 | 0.0502 |
| L = 6 | # of adders for H | 134 | 154 |
| | # of SSS adders | 116 | 130 |
| | # of PPSF adders | 16 | 8 |
| | S. Ripple(dB) | 41.3315 | 42.1684 |
| | P. Ripple(dB) | 0.0515 | 0.0558 |
| L = 8 | # of adders for H | 163 | 196 |
| | # of SSS adders | 141 | 169 |
| | # of PPSF adders | 25 | 12 |
| | S. Ripple(dB) | 42.5423 | 42.3029 |
| | P. Ripple(dB) | 0.0579 | 0.0575 |
| L = 12 | # of adders for H | 230 | 250 |
| | # of SSS adders | 196 | 231 |
| | # of PPSF adders | 48 | 20 |
| | S. Ripple(dB) | 42.9155 | 43.3736 |
| | P. Ripple(dB) | 0.0454 | 0.0584 |

#of SSS adders = sub-structure sharing의 적용 후 adder 수

**Example 3**: 필터 차수 = 72, 통과대역 = $0 \sim 0.4\pi$, 저지대역 = $0.45\pi \sim \pi$인 저역통과 필터를 고려하자[6]: 그림 13은 이상적인 임펄스 응답이고, 그림 14에서는 이상적인 주파수 응답과 제안한 방법에 의한 주파수 응답을 비교한다. 표 2에서는 [6]와 본 논문에서 제안한 방법에 대해 블록 사이즈가 2, 3, 4, 6, 8 12인 경우 하드웨어 비용을 비교한다. 본 논문에서 제안한 방법에 의해 약 5%~15%의 덧셈기를 절감할 수 있음을 보인다.

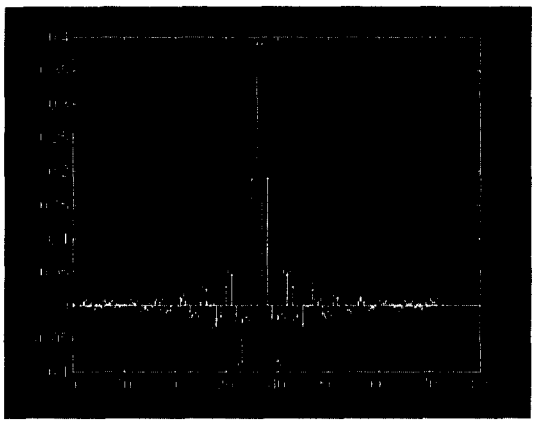


그림 13. Example 3의 이상적인 임펄스 응답.

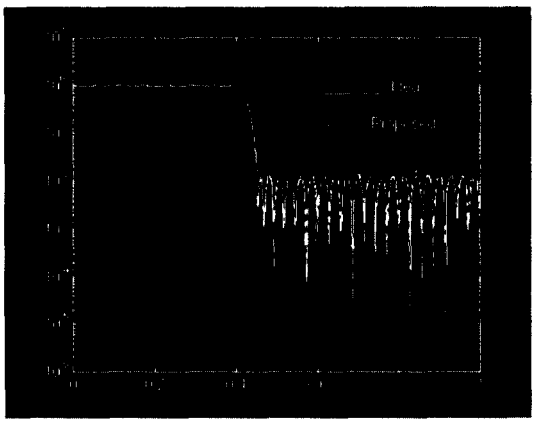


그림 14. Example 3의 주파수 응답.

표 2. Example 3의 필터 특성

| | | Proposed method | Parker's method |
|---|---|---|---|
| L = 2 | # of adders for H | 175 | 191 |
| | # of SSS adders | 127 | 141 |
| | # of PPSF adders | 3 | 1 |
| | S. Ripple(dB) | 32.4643 | 32.311 |
| | P. Ripple(dB) | 0.1306 | 0.1404 |
| L = 3 | # of adders for H | 266 | 290 |
| | # of SSS adders | 186 | 212 |
| | # of PPSF adders | 6 | 3 |
| | S. Ripple(dB) | 32.9597 | 32.8994 |
| | P. Ripple(dB) | 0.1357 | 0.1524 |
| L = 4 | # of adders for H | 298 | 309 |
| | # of SSS adders | 232 | 222 |
| | # of PPSF adders | 5 | 4 |
| | S. Ripple(dB) | 33.9957 | 33.1794 |
| | P. Ripple(dB) | 0.1331 | 0.1473 |
| L = 6 | # of adders for H | 366 | 438 |
| | # of SSS adders | 285 | 356 |
| | # of PPSF adders | 18 | 9 |
| | S. Ripple(dB) | 33.5189 | 0.1472 |
| | P. Ripple(dB) | 0.1455 | 31.8269 |
| L = 8 | # of adders for H | 446 | 505 |
| | # of SSS adders | 356 | 393 |
| | # of PPSF adders | 27 | 8 |
| | S. Ripple(dB) | 34.6538 | 33.6680 |
| | P. Ripple(dB) | 0.1789 | 0.1764 |
| L = 12 | # of adders for H | 655 | 696 |
| | # of SSS adders | 549 | 603 |
| | # of PPSF adders | 50 | 19 |
| | S. Ripple(dB) | 32.8167 | 33.1451 |
| | P. Ripple(dB) | 0.1237 | 0.1436 |

# of SSS adders = sub-structure sharing의 적용 후 adder

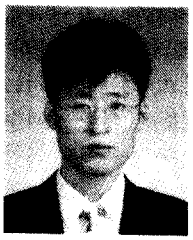
## VI. 결 론

본 논문에서는 높은 동작 주파수나 저전력 소모가 요구되는 FIR 필터의 VLSI 구현을 위한 효율적인 블록 프로세싱 방법을 제시하였다. 필터의 주파수 스펙트럼 특성을 이용하므로써 기존의 블록 필터 구현 방법에 비해 하드웨어 오버헤드를 더욱 감소시킬 수 있음을 보였다. 또한 기존의 블록 필터 양자화 방법보다 특성이 향상된 새로운 블록 필터 양자화 알고리듬을 제시하였다. 블록 필터 구현시 기존의 블록 필터 구현 방법보다 본 논문에서 제안한 방법을 사용함으로써 약 5%~15%의 하드웨어 이득을 얻을 수 있음을 보였다.

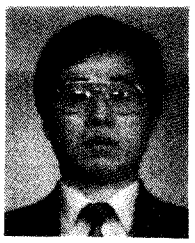
## 참 고 문 헌

1. A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, pp. 473-484, Apr. 1992.
2. Y. C. Lim, "Design of discrete-coefficient-value linear phase FIR filters with optimum normalized peak ripple magnitude", *IEEE Trans. on Circuits and Systems*, vol. 37, pp. 1480-1486, Dec. 1990.

3. H. Samueli, "An improved search algorithm for the design of multiplierless FIR filters with power-of-two coefficients," *IEEE Trans. on Circuits and Systems*, vol. 36, pp. 1044-1047, July 1989.
4. R. Jain, P.T. Yang, and T. Yoshino, "FIRGEN: A computer-aided design system for high performance FIR filter integrated circuits", *IEEE Trans. Signal Processing*, vol 39, pp. 1655-1667, July 1991.
5. R. A. Hawley, B.C. Wong, T. Lin, J. Laskowski, and H. Samueli, "Design techniques for silicon compiler implementations for high-speed FIR digital filters", *IEEE J. Solid-State Circuits*, pp. 656-667, May 1996.
6. D. A. Parker and K. K. Parhi, "Low-area/power parallel FIR digital filter implimentations", *Journal of VLSI Signal Processing*, vol. 17, pp. 75-92, Sept. 1997.
7. Z. J. Mou and P. Duhamel, "Short-length FIR filters and their use in fast nonrecursive filtering," *IEEE Trans. on Signal Processing*, vol. 39, pp. 1322-1332, June 1991.
8. D. A. Parker and K. K. Parhi, "Area-efficient parallel FIR digital filter implementations," in *International Conference on Applications-Specific Systems*, Architectures and Processors, (Chicago, IL), pp. 93-111, Aug. 1996.
9. J. G. Chung, Y. B. Kim, H. G. Jeong, K. K. Parhi and Z. Wang, "Efficient parallel FIR filter Implementations using frequency spectrum characteristics," in *Proceedings of 1998 IEEE ISCAS*, (Monterey, CA), pp. 483-486, May 1998.
10. C. L. Chen, K. Y. Khoo, and A. N. Willson, Jr., "An improved polynomial-time algorithm for designing digital filters with power-of-two coefficients," in *Proceedings of 1995 IEEE ISCAS*, (Seattle, WA), pp. 223-226, May 1995.
11. D. Li, J. Song, and Y. C. Lim, "A polynomial-time algorithm for designing digital filters with power-of-two coefficients," in *Proceedings of 1993 IEEE ISCAS*, (Chicago, IL), pp. 84-87, May 1993.
12. Richard I. Hartley and Keshab K. Parhi, *Digit-Serial Computation*. Norwell, MA: Kluwer Academic Publishers, 1995.



金 容 培(Yong-Bae Kim) 정회원
1996年 2月 : 순천대학교 전자공학과(공학사)
1998年 2月 : 전북대학교 정보통신공학과(공학석사)
<연구분야> VLSI 신호처리



鄭 鎭 均(Jin-Gyun Chung) 정회원
1985年 2月 : 전북대학교 전자공학과(공학사)
1991年 12月 : University of Minnesota 전기공학과(공학석사)
1994年 12月 : University of Minnesota 전기공학과(공학박사)
1995年 3月 ~ 현재 : 전북대학교 정보통신공학과 전임강사, 조교수
1996年 9月 ~ 현재 : 전북대학교 전기·전자 회로합성연구소 운영관리부장
<연구분야> VLSI 신호처리

丁 恒 根(Hang-Geun Jeong) 정회원
1955年 3月 17日生
1977年 2月 : 서울대학교 전자공학과 졸업(학사)
1979年 2月 : 한국과학원 전기 및 전자공학과 졸업(석사)
1989年 12月 : Univ. of Florida 전기공학과 졸업(박사)
1979年 3月 ~ 1982年 2月 : 한국전자통신연구소 연구원
1989年 8月 ~ 1991년 1月 : 모토롤라 연구소 엔지니어
1991年 3月 ~ 현재 : 전북대학교 전임강사, 조교수, 부교수, 전기전자 회로합성연구소 연구원
<연구분야> 아날로그 및 고주파 집적회로 설계