論文2000-37SP-5-12

IF 디지털 다운 컨버터의 블록 FIR 필터링 아키텍처

(A Block FIR Filtering Architecture for IF Digital Down Converter)

張 永 釩 *

(Young-Beom Jang)

요 약

본 논문에서는, 고속의 필터링이 요구되는 IF 디지털 다운 컨버터를 위한 새로운 블록 FIR(Finite Impulse Response) 필터링 아키텍처를 제안한다. 디지털 다운 컨버터는 디지털 믹서, 데시메이션 필터, 그리고 다운 샘플러로 구성된다. 본 논문이 제안하는 아키텍처는 블록 필터링의 병렬처리 구조를 이용하여 데시메이션 필터를 구성함으로서 블록 필터링 아키텍처에서 구조적으로 생겨나는 업 샘플링이 직렬로 연결되는 다운 샘플러와 상쇄되어 구조가 간략하게 되어짐을 보인다. 이와 더불어 블록 FIR 구조를 이용하여 필터계수의 갯수가 블록의 크기의 역비례로 감소되어, 계산량이 그 만큼 감소되어짐을 보인다. 끝으로, 디지털 믹서의 0이 필터의 병렬입력을 0으로 만드는 것을 이용하여 아키텍처의 복잡도가 더욱 감소됨을 보이게된다.

Abstract

In this paper, a block FIR(Finite Impulse Response) filtering architecture is proposed for IF digital down converter. Digital down converter consists of digital mixers, decimation filters and down samplers. In this proposed structure, it is shown that a efficient parallel decimation filter architecture can be produced by cancellation of inherent up sampling of the block filter and following down sampler. Furthermore, it is shown that computational complexity of the proposed architecture is reduced by exploiting the block FIR structure and zero values of the digital mixers.

I. 서 론

최근에 여러 가지의 서로 다른 표준에 따라 휴대용 단말기를 교체하지 않고도 사용할 수 있는 개념의 소프트웨어 라디오가 널리 연구되고 있다. 아날로그 처리로는 여러 가지의 이동 통신 단말기 표준을 모두 지원하는 IF(Intermediate Frequency)처리를 적응적으로 하기 어렵기 때문에 IF의 디지털 처리가 필수적으로 요구된다. 그러나 휴대용 단말기와 같은 저전력이 요구되는 이동 통신기기에서는, 디지털 IF단의 고속 필터링에서 발생하는 전력소모가 반도체 구현의 걸림돌이 되어왔다. 따라서 디지털 IF는 적응성, 고속처리, 저전력의 구현을 모두 가능하게 하는 아키텍처를 요구한다. 디지털 IF가 구현되는 하드웨어로서는, 프로그래머블 DSP와 벡터 프로세서등이 있으며, 각각 적합한 아키텍처가 연구되어질 수 있다. 특히 벡터 프로세서로 디지털 IF를 구현할 경우에, 벡터화되어 처리되는 벡터의 크기에 제한이 있으므로 디지털 IF의 병렬처리에도 제한이 따르게 된다. 이와 같은 IF 단의 디지털 처리 방식을 디지털 다운 컨버터 (Digital Down Converter, 이하 DDC)라고 하며, 그림 1과 같이 나타내진다. IF신호 s(t)는 ADC(Analog-to-Digital Converter)를 통하여 디

* 正會員, 梨花女子大學校 情報通信學科
(Dept. of Information Electronics Engineering, Ewha Womans University)

※ 본 연구는 교육부 BK21 지원에 의해 수행되었음.
接受日字:2000年2月10日, 수정완료일:2000年8月9日

지털 신호로 변환된 후, 디지털 믹서를 통하여 I 방향의 신호와 Q방향의 신호로 분리된 후에 각각 필터에 입력된다.

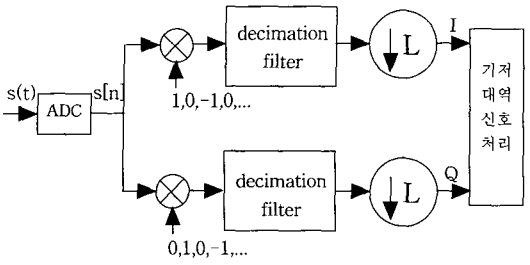

그림 1. 디지털 IF 신호처리 블록도
Fig. 1. Block diagram of digital IF signal processing.

그림 1에서, 디지털 믹서의 계수 값들은 ADC의 샘플링 주파수 f_s가 디지털 믹서의 주파수 f_{dm}의 4배가되는 다음의 관계식에 의하여 매우 효과적으로 정의된다.[1]

$$\begin{aligned} I\text{단 믹서값} &= \cos 2\pi f_{dm} k T_s = \cos 2\pi \frac{f_s}{4} k \frac{1}{f_s} \\ &= \cos \frac{k\pi}{2} = 1, 0, -1, 0, \cdots \\ Q\text{단 믹서값} &= \sin 2\pi f_{dm} k T_s = \sin 2\pi \frac{f_s}{4} k \frac{1}{f_s} \\ &= \sin \frac{k\pi}{2} = 0, 1, 0, -1, \cdots \end{aligned} \quad (1)$$

이 식에서 $T_s = 1/f_s$이다. 위의 식과 같이 샘플링 주파수를 정의하면 믹서 값들이 1, -1, 0으로 되어 믹서 값들을 만들어 내는 하드웨어가 필요 없으며, 믹서 값과 신호의 곱셈이 필요 없게 되는 효율적인 구조가 된다. 나아가서 데시메이션 필터를 본 논문이 제안하는 블록 필터의 구조로 구성할 때 아키텍처를 더욱 간략화 시킬 수 있게 된다. 믹서를 통과한 신호는 데시메이션 필터를 통하여 불필요한 신호대역이 제거된 기저대역 신호가 되며, 마지막으로 과잉의 정보를 제거하기 위하여 다운 샘플러에 의하여 정수 인수인 L 만큼 샘플링 속도가 낮춰진다. 다운 샘플러는 원안에 아랫방향의 화살표와 데시메이션 인수인 L로 표시된다. 그림 1과 같은 디지털 IF가 사용되는 이동 통신 단말기나 기지국 시스템이 가져야하는 적응성으로서는 필터의 차수와 다운 샘플러의 크기가 변수가 될 것이다. 여러 가지 표준의 DDC마다 필터의 차수와 다운 샘플러의 크기가 변화될 것이며, 이에 따른 적응적 아키텍처가 요구되어진다. 디지털 IF의 또 다른 요구조건인 고속처리를 위한 연구도 여러 가지로 진행되고 있다. 현재의 상업적으로 구현 가능한 반도체 집적기술로는 디지털 IF의 병렬처리 아키텍처가 불가피하다. 예를 들면 1차 버전의 CDMA의 IF 주파수는 4.9152MHz이므로 4배로 ADC의 샘플링 주파수로 정하면 샘플링은 19.6608MHz가 된다. 즉 샘플링 주기는 50.86nsec가 된다. 1000MIPS의 DSP로 구현한다고 가정해 보면, 한 개의 명령어 싸이클이 1nsec이므로 입력신호의 샘플링 주기에 겨우 51개 정도의 명령어만을 수행할 수 있다. 이러한 고속의 DSP로 필터링만 한다고 가정하더라도, 51탭의 FIR 필터링만이 가능하다. 따라서 DDC를 DSP로 상용화하기 위해서는 저속처리의 DSP로 구현 가능하도록 병렬처리 아키텍처의 개발이 이루어져야만 한다. 대표적인 병렬처리 방식으로는 Polyphase DDC 아키텍처가 있다.[2][3] 본 논문은 벡터 프로세서 등과 같은 병렬처리기로 DDC를 구현하는 경우에 적합한 블록 필터링 아키텍처를 제안한다. 제안된 아키텍처가 필터의 탭수 변화에 적응하기 용이하며 병렬처리와 저속처리가 가능함을 보이게 된다.

논문의 구성은, II장에서는 데시메이션 필터의 블록 필터링이 뒷단의 다운 샘플러와 어떻게 결합되어 단순화된 구조가 만들어지는지를 보이며, III장에서는 DDC에 적합한 블록 FIR 필터 아키텍처를 유도하는 과정을 살펴본다. 마지막으로, IV장에서는 아키텍처의 예제를, V장에서는 결론을 각각 기술한다.

II. DDC의 블록 필터링화

블록 필터는 필터의 처리속도를 높이기 위하여 만들어진 아키텍처이다.[4][5] 본 논문은 먼저 그림 2와 같이 블록 필터를 데시메이션 필터로 제안한다. 블록 필터는 순차 신호가 직렬-병렬 변환기를 통하여 벡터화되고 블록 필터링된 후에 다시 병렬-직렬 변환기를 통하여 순차신호로 만들어짐으로서 일반적인 Scalar 필터와 등가의 동작이 된다. 그림 2에서 블록 전달함수(Block Transfer Function) $\boldsymbol{H(z)}$는 각 엘리먼트가 다항식인 행렬로 나타내지며 블록 필터의 내부 아키텍처는 III장에서 자세히 다루기로 한다. 그림 2의 하드웨어 구조를 감소시키기 위하여 그림 3과 같은 병렬-직렬 변환기의 등가구조를 유도한다. 즉, 병렬-직렬 변환기는 벡터신호를 순차신호로 변환하는 장치이므로 그림 3과 같은

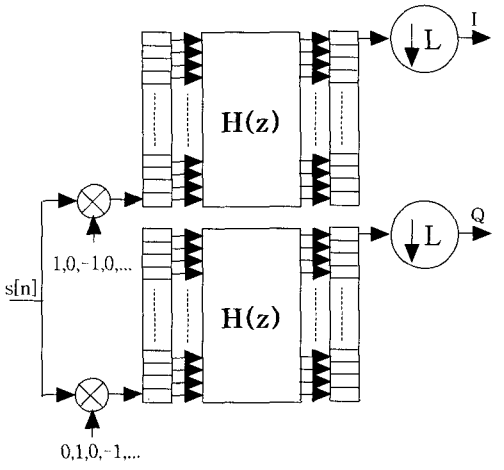

그림 2. 데시메이션 필터의 블록 필터링 구조
Fig. 2. Block filtering structure for decimation filter.

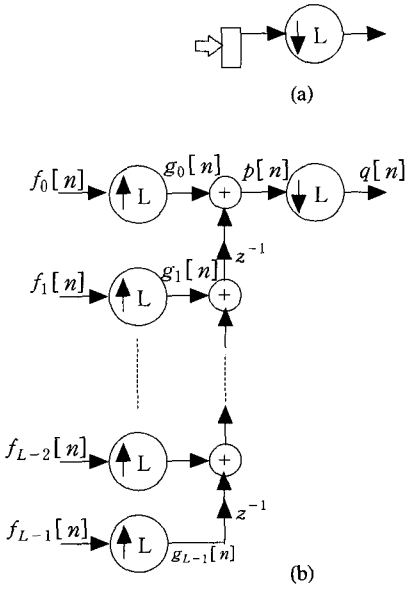

그림 3. (a) 병렬-직렬 변환기와 다운 샘플러
(b) (a)의 등가구조
Fig. 3. (a) Parallel-to-serial converter and down sampler, (b) equivalent structure of (a).

등가구조로 나타낼 수 있다. 즉 업 샘플러에 의해서 0이 삽입되고, 지연소자에 의해 순서대로 지연되어 더하면 순차의 신호가 만들어진다. 이 과정을 수식으로 표현하면 다음과 같다. 먼저 그림3 (b)의 업 샘플러의 입력단을 $f_k[n]$이라고 하고, 출력단을 $g_k[n]$이라고 하면 관계식은 다음과 같다.

$$\begin{aligned} g_0[n] &= \begin{cases} f_0[\frac{n}{L}], & n=0,L,2L,\cdots \\ 0, & o.w \end{cases} \\ g_1[n] &= \begin{cases} f_1[\frac{n}{L}], & n=0,L,2L,\cdots \\ 0, & o.w \end{cases} \\ &\vdots \\ g_{L-1}[n] &= \begin{cases} f_{L-1}[\frac{n}{L}], & n=0,L,2L,\cdots \\ 0, & o.w \end{cases} \end{aligned} \tag{2}$$

그림 3 (b)의 다운 샘플러의 입력단을 $p[n]$, 출력단을 $q[n]$이라고 정의하고, 먼저 $p[n]$을 $f_k[n]$의 조합으로 나타내보기로 하자. $p[n]$은 $g_k[n]$들의 지연된 합으로 표시되는데, $g_k[n]$들이 바로 앞의 업 샘플러들에 의해서 0이 삽입된 신호들이므로 실제로는 덧셈의 동작은 발생하지 않고, 한 개의 샘플씩 채워지는 동작이다. 따라서 다음의 식(3)와 같이 $f_k[n]$으로 나타낼 수 있다.

$$\begin{aligned} p[n] &= g_0[n]+g_1[n-1]+\cdots+g_{L-1}[n-L+1] \\ &= \begin{cases} f_0[\frac{n}{L}], & n=0,L,2L,\cdots \\ f_1[\frac{n-1}{L}], & n=1,L+1,2L+1,\cdots \\ \vdots & \\ f_{L-1}[\frac{n-(L-1)}{L}], & n=L-1,2L-1,3L-1,\cdots \end{cases} \end{aligned} \tag{3}$$

마지막으로, 다운 샘플러의 입출력신호 $q[n]$과 $p[n]$의 관계를 나타내보기로 하자. 다운 샘플러의 동작은 우선 L의 배수를 제외한 시간의 샘플 값들을 0으로 바꾸는 단계와, 시간 축을 압축하는 단계로 나누어 분석하는 것이 일반적인 방법이다. 즉, 첫 단계의 중간신호를 $p'[n]$이라고 하면 다음과 같이 표시된다.

$$p'[n] = \begin{cases} f_0[\frac{n}{L}], & n=0,L,2L,\cdots \\ 0, & o.w. \end{cases} \tag{4}$$

두 번째 단계는 시간 축에서 압축하여 0을 제거하는 단계이므로 다음의 결과 식이 유도된다.

$$q[n] = f_0[n], \quad n=0,1,2,\cdots \tag{5}$$

결과 식 (5)는 그림 2에서의 필터의 병렬출력, 병렬-직렬 변환기, 다운 샘플러의 3가지가 하나의 출력으로 간략화될 수 있음을 의미하며, 이를 그림으로 나타내면 그림 4와 같다. 구현 하드웨어의 감소와 더불어, 벡터 출력을 갖는 블록 필터의 L개의 출력 중에서 맨 위의 출력만을 계산하면 된다.

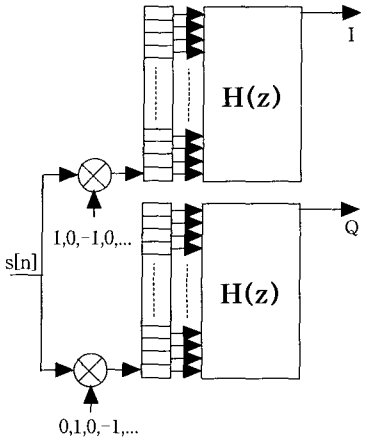

그림 4. 다운 샘플러의 상쇄에 의해 단순화된 블록 필터링 구조

Fig. 4. Simplified block filtering structure by down sampler cancellation.

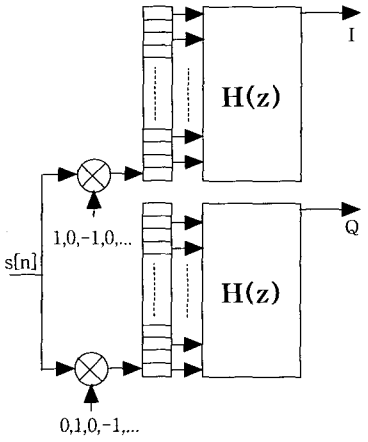

그림 5. 최종의 블록 필터링 DDC 아키텍처

Fig. 5. Final block filtering DDC architecture.

디지털 믹서의 곱셈 값들에 섞여있는 0들을 이용하여 구현 하드웨어와 계산량을 더욱 감소시키는 방법이 제안되었는데[2] 본 논문에서도 이를 그림 4에 다음과 같이 적용한다. 즉 그림 4의 윗쪽의 디지털 믹서의 짝수번째가 항상 0이기 때문에 직렬-병렬 변환기의 맨윗쪽의 출력단부터 짝수 번째는 항상 0이 나가게 되므로 그림 5와 같이 제거시킬 수 있다. 그림 5의 I단의 믹서의 의미는 s[n]의 짝수번째는 모두 버리고, 홀수번째의 신호는 하나는 그대로 다음은 극성을 반전시켜서 직렬-병렬 변환기로 보내는 것이다. 짝수번째의 신호를 모두 제거하였으므로 직렬-병렬 변환기의 크기도 실제로는 반으로 줄게 된다. Q단도 같은 방법으로 동작된다. 다만, 홀수번째의 신호를 버리는 것만이 다르다. 그림 5가 본 논문이 제안하는 최종의 아키텍처이다. 이제 다음 장에서 마지막으로 블록 필터의 효과적인 내부 구조를 제안하도록 한다.

Ⅲ. 블록 FIR 필터링 아키텍처

벡터 입력과 벡터의 출력을 갖는 블록 필터에도 여러 가지 FIR과 IIR(Infinite Impulse Response)의 구조가 연구되어져있다.[6] Polyphase와 같은 병렬처리 방식은 FIR 필터에만 적용이 가능하나, 본 논문이 제안하는 블록 필터 방식은 FIR과 IIR 모두 가능하다. 즉 그림 5의 블록 필터 내부 구조로서 FIR과 IIR이 모두 가능하다. 이 장에서는 블록 FIR Direct form을 내부 구조로 선택하였다. 그러면 일반적인 FIR 필터의 전달 함수가 어떤 방법으로 블록 필터의 전달 함수로 유도되는지를 먼저 알아본다. 일반적인 N차 필터의 시간 영역에서의 입출력 관계식은 다음과 같이 나타내어진다.

$$y_n = \sum_{k=0}^{N} h_k \, x_{n-k} \tag{6}$$

블록 전달함수(Block Transfer Function)를 유도하기 위하여 먼저 위의 시간영역에서의 관계식을 다음의 식 (7)과 같이 행렬식으로 나타내어야 한다.

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_N \\ y_{N+1} \\ y_{N+2} \\ \vdots \end{bmatrix} = \begin{bmatrix} h_0 & 0 & 0 & \cdots \\ h_1 & h_0 & 0 & \cdots \\ h_2 & h_1 & h_0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ h_N & h_{N-1} & h_{N-2} & \cdots \\ 0 & h_N & h_{N-1} & \cdots \\ 0 & 0 & h_N & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_N \\ x_{N+1} \\ x_{N+2} \\ \vdots \end{bmatrix} \tag{7}$$

첫 번째 단계로서, 위의 행렬식에서 다운 샘플러의 크기 L 과 같은 크기로 중첩되지 않도록 서브 행렬로 분할한다. 즉 벡터는 L 크기의 서브벡터로, 행렬은 $L \times L$ 크기의 서브행렬로 분할하면 다음 식 (8)과 같은 행렬식으로 표현된다.

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_M \\ y_{M+1} \\ \vdots \end{bmatrix} = \begin{bmatrix} \boldsymbol{H_0} & \boldsymbol{0} & \cdots \\ \boldsymbol{H_1} & \boldsymbol{H_0} & \cdots \\ \vdots & \vdots & \ddots \\ \boldsymbol{H_M} & \boldsymbol{H_{M-1}} & \cdots \\ \boldsymbol{0} & \boldsymbol{H_M} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_M \\ x_{M+1} \\ \vdots \end{bmatrix} \tag{8}$$

위의 행렬식에서, 벡터들의 엘리먼트들도 모두 서브

벡터들이며, 행렬의 엘리먼트들도 역시 모두 서브 행렬로 구성되며 다음의 식 (9)과 (10)와 같이 나타내진다.

$$\underline{y}_m = [\, y_{mL} \;\; y_{mL+1} \;\; y_{mL+2} \;\cdots\; y_{mL+L-1}]^t$$
$$\underline{x}_m = [\, x_{mL} \;\; x_{mL+1} \;\; x_{mL+2} \;\cdots\; x_{mL+L-1}]^t \qquad (9)$$
$$m = 0, 1, 2 \cdots$$

$$\boldsymbol{H}_0 = \begin{bmatrix} h_0 & 0 & \cdots & 0 \\ h_1 & h_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ h_{L-1} & h_{L-2} & \cdots & h_0 \end{bmatrix},$$

$$\boldsymbol{H}_1 = \begin{bmatrix} h_L & h_{L-1} & \cdots & h_1 \\ h_{L+1} & h_L & \cdots & h_2 \\ \vdots & \vdots & \ddots & \vdots \\ h_{2L-1} & h_{2L-2} & \cdots & h_L \end{bmatrix},$$

$$\cdots,\; \boldsymbol{H}_M = \begin{bmatrix} h_N & h_{N-1} & \cdots & h_{N-L+1} \\ 0 & h_N & \cdots & h_{N-L+2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & h_N \end{bmatrix}. \qquad (10)$$

블록의 크기 L은 다운 샘플러의 크기와 같도록 결정된다. 주어진 원래의 필터 차수 N과 L로부터 블록 필터의 차수 M이 결정되며, 이는 $M = \lceil N/L \rceil$로 나타내어진다. 이 괄호는 괄호 안의 값과 같거나 큰, 가장 작은 정수를 의미한다. 이와 같은 블록 행렬식으로부터 다음의 식 (11)과 같은 블록 차등방정식(Block Difference Equation)이 유도되어질 수 있다.

$$\underline{y}_k = \boldsymbol{H}_0\, \underline{x}_k + \boldsymbol{H}_1\, \underline{x}_{k-1} + \cdots + \boldsymbol{H}_M\, \underline{x}_{k-M} \qquad (11)$$

이식에 블록 z-Transform을 하면 다음의 블록 전달함수가 얻어진다.

$$\boldsymbol{H}(z) = \boldsymbol{H}_0 + \boldsymbol{H}_1 z^{-1} + \cdots + \boldsymbol{H}_M z^{-M} \qquad (12)$$

이렇게 유도된 블록 차등방정식이나 블록 전달함수로부터 다음 그림 6과 같은 Direct Form 구조의 블록 필터 구조가 만들어진다. 이 그림에서 굵게 표시된 화살표는 벡터 신호를 나타낸다. 일반적인 하나의 입력과 하나의 출력을 갖는 필터의 구성요소와 마찬가지로, 블록 필터의 구조도 곱셈기, 덧셈기, 그리고 지연소자로 구성된다. 이 구조에서는 곱셈기의 필터계수는 행렬로 표시되어 벡터와 곱해지며, 덧셈기는 벡터끼리의 덧셈이 되며, 지연소자는 벡터의 지연을 의미한다. 그림 6의 입력 벡터는 직렬-병렬 변환기를 통하여 L의 크기로 병렬화되었으므로, 시스템 클럭도 $1/L$로 감소된다. 예를 들어 입력신호의 주기가 16MHz이면, 일반 Scalar 데시메이션 필터를 사용하는 경우에 시스템 클럭도 16MHz가 된다. 반면에 블록 크기가 8인 블록 필터를 사용하면 시스템 클럭은 2MHz로 낮아진다. 따라서 그림 6의 지연소자, 곱셈기, 덧셈기들은 동작속도를 1/8로 낮출 수 있게된다. 위의 블록 필터의 곱셈의 양을 살펴보면, 하나의 필터 계수 행렬마다 $L \times L$ 의 곱셈이 필요하며, 이런 계수가 M+1개 있으므로 곱셈의 총 수는 $L^2(M+1)$이 된다. II장에서 본 논문이 제안한 그림 4의 아키텍처는 블록 필터의 출력 벡터 중에서 오직 한 개만을 필요로 하므로 이를 적용하면 그림 7과 같이 벡터 입력과 스칼라 출력을 갖는 구조가 얻어진다. 그림 6에서는 필터계수가 행렬이지만, 여기에서는 그 행렬의 첫 번째 행만을 취한 벡터가 된다. 왜냐하면 곱셈기의 출력벡터 중에서 하나만을 필요로 하기 때문에. 필터계수 행렬과 입력 벡터와의 곱이 필터계수 벡터와 입력 벡터와의 곱으로 단순화되기 때문이다. 따라서 식 (10)의 필터계수 행렬들이 다음과 같이 필터계수 벡터로 단순화된다.

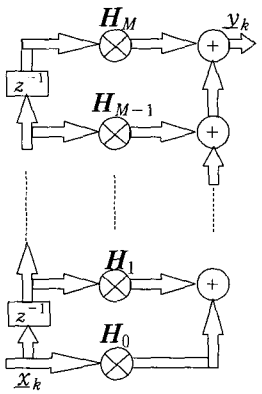

그림 6. 블록 FIR 필터링 아키텍처
Fig. 6. Block FIR filtering architecture.

$$\begin{aligned} \underline{H}_0 &= [h_0 \quad 0 \quad \cdots \quad 0 \;]^t \\ \underline{H}_1 &= [h_L \quad h_{L-1} \quad \cdots \quad h_1 \;]^t \\ &\vdots \\ \underline{H}_M &= [h_N \quad h_{N-1} \quad \cdots \quad h_{N-L+1}]^t \end{aligned} \qquad (13)$$

그림 7에서의 곱셈의 양을 살펴보면, 하나의 필터 계수 벡터마다 L개의 곱셈이 필요하며, 이런 계수가 M+1개 있으므로 곱셈의 총 수는 $L(M+1)$로 감소된다.

위의 블록 FIR 구조는 I단과 Q단에서 다음과 같이 구분되어진다. 즉, I단에서는 s[n]의 홀수번째 신호만

입력되므로 그림 7의 블록 크기가 반으로 감소되며 식 (13)의 필터계수 벡터도 다음 식과 같이 홀수번째 엘리먼트만이 사용되며 크기가 반으로 감소된다.

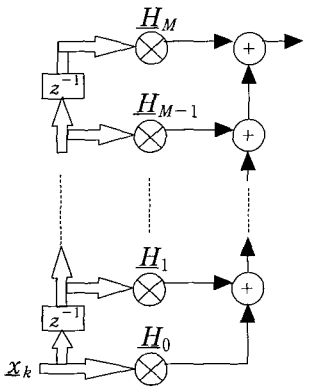

그림 7. 제안된 블록 FIR DDC 구조
Fig. 7. Proposed block FIR DDC structure.

$$\begin{aligned} \underline{H}^I{}_0 &= [h_0 \quad 0 \quad \cdots \quad 0 \quad]^t \\ \underline{H}^I{}_1 &= [h_L \quad h_{L-2} \quad \cdots \quad h_2 \quad]^t \\ &\vdots \\ \underline{H}^I{}_M &= [h_N \quad h_{N-2} \quad \cdots \quad h_{N-L+2}]^t \end{aligned} \tag{14}$$

Q단에서는 s[n]의 짝수번째 신호가 입력되므로 식 (13)을 마찬가지 방법으로 다음 식과 같이 짝수번째 엘리먼트만으로 변형시켜서 사용한다.

$$\begin{aligned} \underline{H}^Q{}_0 &= [\,0 \quad 0 \quad \cdots \quad 0 \quad]^t \\ \underline{H}^Q{}_1 &= [h_{L-1} \quad h_{L-3} \quad \cdots \quad h_1 \quad]^t \\ &\vdots \\ \underline{H}^Q{}_M &= [h_{N-1} \quad h_{N-3} \quad \cdots \quad h_{N-L+1}]^t \end{aligned} \tag{15}$$

그림 7에서 보여지듯이, 계산량을 현저히 감소시키기 위해서는 블록 필터의 구조가 갖추어야하는 조건이 입력벡터와 출력벡터 사이에 하나의 필터계수만이 존재해야한다. 여러 개의 필터계수가 존재하면 마지막의 계수만이 벡터와 벡터의 곱으로 단순화되므로 계산량의 큰 이득을 얻을 수 없다. 따라서 블록 State Space 필터구조나, 블록 Lattice 필터구조와 같은 여러 단의 직렬 연결로 되어있는 블록 필터구조는 DDC용으로 적합하지 않음을 알 수 있다.

Ⅳ. 아키텍처 비교 분석

이 장에서는 그림 1의 DDC를 본 논문이 제안한 아키텍처와, 기존의 병렬구조인 Polyphase DDC를 다음의 사양으로 구성하여 비교해본다. 데시메이션 필터의 차수는 32차로 하고 다운 샘플러의 크기는 8로 하였다. 그림으로 전체의 아키텍처를 나타낼 수 있도록 하기 위하여 차수를 32로 제한하였다. 블록의 크기는 다운 샘플러의 크기와 같이 8이 되며, 블록 필터의 차수는 주어진 필터의 차수 32를 블록의 크기 8로 나누어 4가 된다. 이렇게 하여 Ⅲ장에서의 단계를 거치면 다음의 블록 차등방정식이 얻어진다.

$$\begin{aligned} y_k = {} & \boldsymbol{H}_0\, x_k + \boldsymbol{H}_1\, x_{k-1} + \boldsymbol{H}_2\, x_{k-2} \\ & + \boldsymbol{H}_3\, x_{k-3} + \boldsymbol{H}_4\, x_{k-4} \end{aligned} \tag{16}$$

위의 식에서 필터계수 행렬들에서 실제로 필요한 첫째 행만으로 만들어진 필터계수 벡터는 다음과 같다.

$$\begin{aligned} \underline{H}_0 &= [h_0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^t \\ \underline{H}_1 &= [h_8 \quad h_7 \quad h_6 \quad h_5 \quad h_4 \quad h_3 \quad h_2 \quad h_1]^t \\ \underline{H}_2 &= [h_{16} \quad h_{15} \quad h_{14} \quad h_{13} \quad h_{12} \quad h_{11} \quad h_{10} \quad h_9]^t \\ \underline{H}_3 &= [h_{24} \quad h_{23} \quad h_{22} \quad h_{21} \quad h_{20} \quad h_{19} \quad h_{18} \quad h_{17}]^t \\ \underline{H}_4 &= [h_{32} \quad h_{31} \quad h_{30} \quad h_{29} \quad h_{28} \quad h_{27} \quad h_{26} \quad h_{25}]^t \end{aligned} \tag{17}$$

마지막으로, 위의 식은 I단과 Q단에 따라 식 (14)와 (15)를 이용하여 다음과 같이 구분하여 변형된다. 먼저 I단의필터계수 벡터는 다음 식과 같다.

$$\begin{aligned} \underline{H}^I{}_0 &= [h_0 \quad 0 \quad 0 \quad 0\,]^t \\ \underline{H}^I{}_1 &= [h_8 \quad h_6 \quad h_4 \quad h_2]^t \\ \underline{H}^I{}_2 &= [h_{16} \quad h_{14} \quad h_{12} \quad h_{10}]^t \\ \underline{H}^I{}_3 &= [h_{24} \quad h_{22} \quad h_{20} \quad h_{18}]^t \\ \underline{H}^I{}_4 &= [h_{32} \quad h_{30} \quad h_{28} \quad h_{26}]^t \end{aligned} \tag{18}$$

Q단에 사용되는 필터계수 벡터는 다음과 같다.

$$\begin{aligned} \underline{H}^Q{}_0 &= [0 \quad 0 \quad 0 \quad 0\,]^t \\ \underline{H}^Q{}_1 &= [h_7 \quad h_5 \quad h_3 \quad h_1\,]^t \\ \underline{H}^Q{}_2 &= [h_{15} \quad h_{13} \quad h_{11} \quad h_9\,]^t \\ \underline{H}^Q{}_3 &= [h_{23} \quad h_{21} \quad h_{19} \quad h_{17}]^t \\ \underline{H}^Q{}_4 &= [h_{31} \quad h_{29} \quad h_{27} \quad h_{25}]^t \end{aligned} \tag{19}$$

이와 같이 구한 필터계수 벡터의 식 (18)과 (19)를 사용하여 DDC를 구성하면 최종적으로 그림 8의 아키텍처가 만들어진다. 그림 8의 I단의 믹서 동작은 s[n]의 짝수번째 신호들은 버리고 홀수번째의 신호들만 1과 -1을 곱하여 통과시키는 동작이다. 마찬가지로 Q단 믹서의 동작은 s[n]의 홀수번째 신호들은 버리고 짝수번째의 신호들만 1과 -1을 곱하여 통과시키는 동작이다. 그림 8의 사선으로 채워진 네모는 신호의 방향이 위로 향하는 직렬-병렬 변환기로서 크기가 4이다. 본 논문이

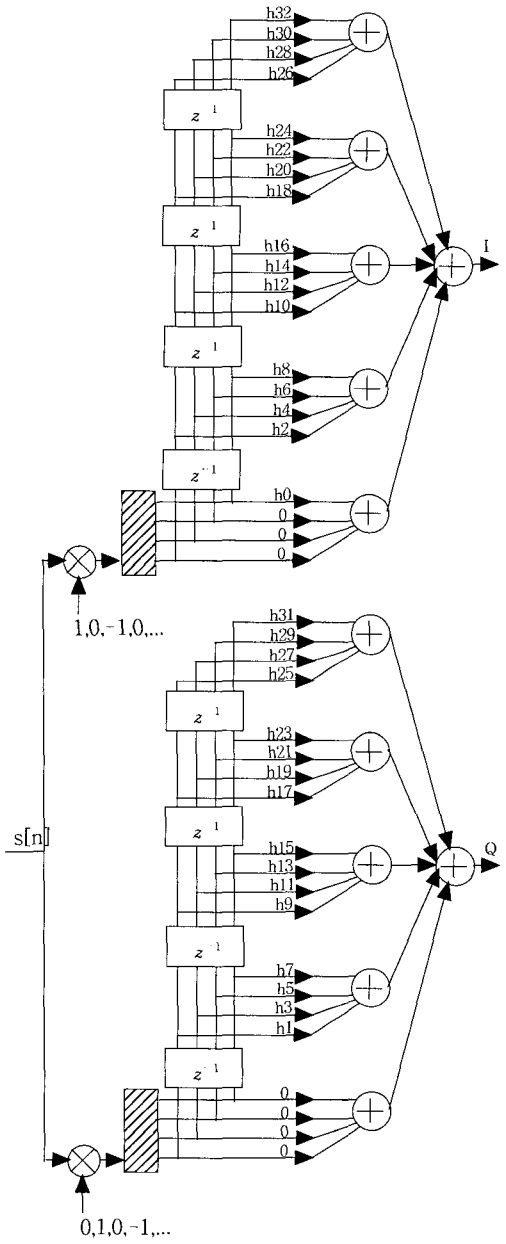

그림 8. 완성된 블록 FIR 필터링 DDC 아키텍처
(필터차수:32, 다운샘플러 크기:8)

Fig. 8. Final block FIR DDC architecture.
(filter order:32, down sampler size:8).

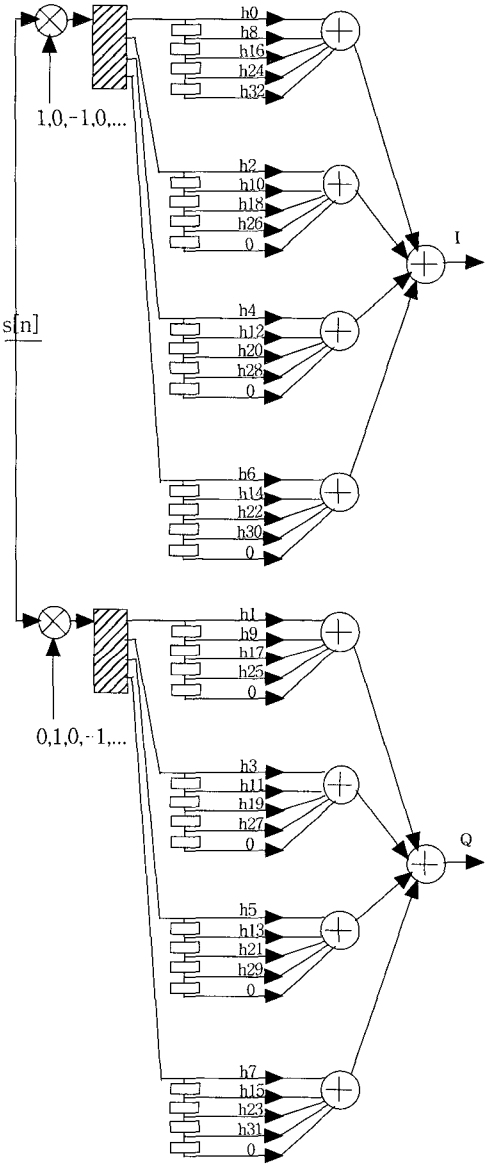

그림 9. Polyphase의 DDC 아키텍처
(필터차수:32, 다운샘플러 크기:8)

Fig. 8. Polyphase DDC architecture.
(filter order:32, down sampler size:8)

제안하는 그림 8의 아키텍처와 비교하기 위하여 필터의 차수와 다운 샘플러의 크기가 각각 32와 8인 Polyphase DDC를 그림 9와 같이 구성하였다. 그림 9에서의 사선으로 채워진 네모는 역시 크기가 4인 직렬-병렬 변환기이나, 신호의 방향이 아래로 향하는 점이 다르다. 그리고 그림 9에서 네모는 지연소자를 나타낸다. 그림 8과 그림 9에서 0인 필터계수들은 제거해도 되나 Modulo한 아키텍처를 나타내기 위하여 그대로 두었다. 각각의 구조에서 직렬-병렬 변환기를 통과한 후, 시스템 클럭은 모두 s[n]의 시스템 클럭에 대하여 1/8로 낮아지는 병렬처리 구조이다.

그림 8과 9는 소프트웨어 라디오와 같은 고속처리를

요하는 DDC에 모두 사용될 수 있는 병렬처리 구조이며, 곱셈과 덧셈의 수는 모두 같다. 그러나 소프트웨어 라디오와 같은 시스템은 여러 개의 표준을 실시간으로 모드 변환하며 지원할 수 있는 Reconfigurability가 중요한 고려사항이다. 여러 개의 상이한 표준마다 필터의 차수가 변화할 때 필터가 적응하기 위하여 그림 8과 같은 Modulo한 아키텍처가 바람직하다. 즉, 그림 8에서 알 수 있듯이 필터의 차수가 증가하여도 Modulo한 아키텍처는 그대로 유지되며 필터의 차수가 8이 증가할 때마다, I와 Q단의 맨 위에 한 블록씩만 추가하면 된다. 이와 비교하여 Polyphase DDC의 경우에는 각각의 모듈이 모두 변경되어야 한다. 그림 9에서의 모듈의 정의는 4개의 지연소자와 5개의 필터계수와 1개의 덧셈기로 구성된 부분을 의미한다. 그림 9에서는 필터차수가 8이 증가하면 각각의 모듈에 1개의 필터계수가 추가되는 구조로 변경된다.

필터 차수의 변화에 대한 계산시간의 차이를 블록 DDC와 Polyphase DDC에서 구해보기로 한다. 필터 차수는 24차에서 48차까지 8차씩 증가되는 4가지의 필터를 지원하는 DDC를 만들도록 한다. 즉 24, 32, 40, 48차의 차수 변화가 있는 경우를 비교하기로 한다. 각각의 아키텍처에서 한 개의 모듈 당 한 개의 곱셈기를 사용하도록 한다. 한 개의 곱셈, 즉 1 MAC (Multiplication and Accumulation)에 필요한 계산시간을 10nsec라고 가정한다. 먼저 블록 DDC의 경우에, 24차 필터이면 8개의 모듈이 필요하며 각 모듈 당 40nsec의 계산시간이 소요되며 총 소요시간도 역시 40nsec가 된다. 차수가 48차로 증가되면 모듈은 14개가 되며 역시 각 모듈 당 40nsec의 계산시간이 소요되므로 총 소요시간도 40nsec이다. 즉 모듈이 8개에서 14개까지 사용되며 계산시간은 항상 40nsec가 된다. Polyphase DDC의 경우를 살펴보자. 24차인 경우에는 8개의 모듈이 필요하며 각 모듈 당 40nsec의 계산시간이 소요되며 총 소요시간도 블록 DDC와 마찬가지로 40nsec가 된다. 차수가 48차로 증가되면 모듈은 8개이지만 각 모듈 당 7번의 곱셈이 필요하므로 70nsec의 계산시간이 소요되어 총 소요시간도 70nsec이다. 즉 모듈이 8개로 고정이나, 각 모듈의 계산시간이 40nsec에서 70nsec까지 변화한다. 이를 종합하면 표 1과 같다. 표 1에서의 필터차수 32인 경우가 그림 8과 9이다. 위의 표에서 모듈수가, 칩으로 구현시에는 면적이 된다. 따라서 24차에서 48차까지 지원되는 DDC를 설계할 경우에 블록 DDC는 면적이 14이고, Polyphase DDC는 8이 되므로 Polyphase DDC가 면적의 측면에서는 더 이득이 된다. 따라서 필터차수의 변화량과 계산시간의 사양에 따라 블록 DDC와 Polyphase DDC 중에서 적절한 아키텍처의 선택을 할 수 있을 것이다.

표 1. 필터차수 변화에 따른 DDC의 계산시간 비교

Table 1. Computation time for filter order variation.

필터 차수	블록 DDC			Polyphase DDC		
	모듈수	곱셈수/모듈	계산 시간 (nsec)	모듈수	곱셈수/모듈	계산 시간 (nsec)
24	8	4	40	8	4	40
32	10	4	40	8	5	50
40	12	4	40	8	6	60
48	14	4	40	8	7	70

V. 결 론

본 논문에서는 고속의 DDC에 적합한 블록 FIR 필터링 아키텍처를 제안하였다. 블록 필터링 구조가 다운 샘플러와 결합하는 경우에, 구조적으로 블록 필터링의 뒷단에서 만들어지는 업 샘플러와 따라오는 다운 샘플러가 상쇄되어, 블록 필터의 병렬-직렬 변환기와 다운 샘플러를 제거할 수 있음을 보였다. 더욱이 블록 FIR 구조를 채용하여 필터계수 행렬이 필터계수 벡터로 변환되어 계산량이 1/L로 감소됨을 보였다. 또한 디지털 믹서의 0의 계수로 인하여 입력벡터가 0이 되는 것을 이용하여 계산량이 다시 1/2로 감소되는 아키텍처가 됨을 보였다. 예제를 통하여, 필터의 차수가 증가할 때에 단위 블록만이 추가되는 modulo한 구조가 됨을 보였다.

참 고 문 헌

[1] 안승혁, 박인순, 최진규, 이용훈, "중간 주파수 디지털 신호처리," 전자공학회지 제27권 제4호 72-82쪽, 2000년 4월

[2] M. Bellanger, G. Bonnerot, and M. Coudreuse, "Digital filtering by polyphase network: Application to sample rate alteration and filter

banks," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-24, pp. 109-114, Apr 1976.

[3] S. J. Jou, S. Y. Wu, and C. K. Wang, "Low-power multirate architecture for IF digital frequency down converter," IEEE Trans. Circuits and Systems-II: Analog and Digital Signal Processing, vol. 45, No. 11, pp. 1487-1494, Nov. 1998.

[4] C. S. Burrus, "Block Implementation of Digital Filters," IEEE Trans. Circuit Theory, vol. CT-18, No. 6, pp. 697-701, Nov. 1971.

[5] C. W. Barnes and S. Shinnaka, "Block- Shift Invariance and Block Implementation of Discrete-Time Filters," IEEE Trans. Circuits and Systems, vol. CAS-27, No. 8, pp. 667-672, Aug. 1980.

[6] Y. Jang and S. P. Kim, "Block digital filter structures and their finite precision responses," IEEE Trans. Circuits and Systems-II: Analog and Digital Signal Processing, vol. 43, No. 7, pp. 495-506, July 1996.

저 자 소 개

張 永 釩(正會員)

1981년 2월 연세대학교 전기공학과 학사. 1990년 1월 Polytechnic University 전기공학과 석사. 1993년 12월 Polytechnic University 전기공학과 박사. 1983년 7월~1987년 12월 삼성전자 마이크로 사업부에서 공학용계산기 알고리듬 및 칩개발, 1993년 12월~1999년 12월 삼성전자 System LSI 사업부에서 오디오 및 음성 신호처리 알고리듬 및 칩 개발, 1999년 12월~현재 이화여자대학교 정보통신학과 조교수, 주관심분야는 통신 신호처리, 오디오 및 음성 신호처리