한국산학기술학회논문지
Vol. 12, No. 1 pp. 364-370, 2011

MAG 알고리즘에 의한 힐버트 변환기의 하드웨어 복잡도 감소에 관한 연구

김영웅[1], 이영석[2*]
[1]청운대학교 정보산업대학원 전자공학과, [2]청운대학교 디지털방송공학과

A Study on the Hardware Complexity Reduction of Hilbert transformer by MAG algorithm

Young-Woong Kim[1] and Young-Seock Lee[2*]
[1]**Dept. of Electronics Engineering, Graduate School of Information & Industry, Chungwoon University**
[2]**Dept. of Digital Broadcasting and Electronic Engineering, Chungwoon University**

요 약 힐버트 변환은 무선 디지털 통신으로부터 수신된 대역통과 신호를 저역통과 신호로 변환시켜 사용자에게 필요한 정보를 제공할 수 있는 역할을 수행한다. 힐버트 변환의 기본 연산과정은 승산과 가산 연산으로 구성되어 있으며, 힐버트 변환을 하드웨어로 구현한 힐버트 변환기는 승산기 설계에서 많은 양의 게이트들을 이용한 설계가 요구되고, 이에 따라 구현된 힐버트 변환기는 높은 소비전력과 넓은 면적을 차지하여 모바일 디지털 기기의 전체적인 성능에 영향을 미친다. 본 논문에서는 MAG(Minimum Adder Graph) 알고리즘을 이용하여 승산 연산의 복잡도를 감소시켜 디지털 통신기기 특히 모바일 시스템의 구성요소인 힐버트 변환기를 기존의 방법보다 더 적은 게이트를 이용하여 구현할 수 있는 방법을 제안하였다. 제안된 방법은 Xilinx사의 ISE환경에서 모의 실험하여 성능의 우수함을 보여주었다.

Abstract The Hilbert transform performs a role to transform band pass signals into low pass signals in wireless communication systems. The operation of Hilbert transform is based on a convolution process which is required adding and multiplying calculations. When the Hilbert transform is designed and hardware-implemented at gate level, the adding and multiplying operation requires a high power consumption and a occupation of wide area on a chip. So the results of adding and multiplying operation cause to degrade the performance of implemented system.
In this paper, the new Hilbert transformer is proposed, which has a low hardware complexity by application of MAG(Minimum Adder Graph) algorithm. The proposed Hilbert transformer was simulated in ISE environment of Xilinx and showed the reduction of hardware complexity comparing with the number of gate in the conventional Hilbert transformer.

Key Words : SOC, FPGA, Hilbert Transform, MAG algorithm

1. 서론

최근 디지털 멀티미디어기기는 휴대성과 편의성뿐만 아니라 시간과 장소에 구애받지 않는 유비쿼터스(Ubiquitous) 기술이 접목된 다양한 디지털 기기들이 개발되고 있다. 즉, 기본적인 통신기능 이외의 사용자의 다양한 요구를 충족시키기 위해 여러 부가 기능들이 결합되고도 작은 크기와 빠르고 안정적인 동작성능 및 낮은 소비전력을 갖추어야 한다. 이에 따라 하나의 칩 위에 여러 시스템을 설계하기 위한 SOC(system on chip)분야의

본 연구는 2010학년도 청운대학교 학술연구조성비에 의하여 수행되었음.
*교신저자 : 이영석(yslee@chungwoon.ac.kr)
접수일 10년 10월 27일 수정일 (1차 10년 12월 15일, 2차 11년 01월 12일) 게재확정일 11년 01월 13일

중요성이 증가하고 있다. 특히, 현대의 디지털 통신 및 멀티미디어 기기는 복합적인 아날로그 신호를 양자화한 디지털 신호를 다루기 위해 고성능의 디지털 신호처리 시스템을 필요로 한다. 그러나 정밀한 데이터 처리를 위한 높은 샘플링 비율은 연산량을 증가시키고 하드웨어 설계를 복잡하게 한다. 또한, 많은 게이트 소자 사용에 따른 설계로 전력 소비 및 개발 비용 증가의 문제를 발생시키기 때문에 하드웨어의 성능 향상과 저 전력 설계를 위해 다양한 연구가 수행되고 있다[1-4].

최근 모바일 디지털 기기들은 기본적으로 무선 네트워크 통신 시스템을 탑재하여 설계되고 있다. 무선 네트워크 통신 시스템의 설계에서 널리 사용되고 있는 하드웨어 기반 디지털 신호처리 알고리즘 가운데 힐버트 변환이 있다. 일반적으로 힐버트 변환은 통신 시스템의 디지털 신호처리를 위해 특정 데이터의 변환 및 추출을 많은 승산 연산을 통해 수행한다. 따라서 힐버트 변환기의 설계에서 승산기의 사용은 불가피하며, 승산기의 하드웨어 구현은 많은 수의 게이트 소자를 필요로 하고, 하드웨어의 많은 면적을 차지 할 뿐만 아니라 많은 양의 데이터 처리 시 연산 지연에 따른 동작 속도에 영향을 미친다.

본 논문에서는 이와 같은 문제를 보완하기 위해 기존 MAG(minimum adder graph) 알고리즘[5,6]을 이용해 곱셈 블록을 힐버트 변환기 설계에 적용하여 기존 승산기를 대체함으로써 빠른 동작 성능과 게이트 소자를 감소시킬 수 있는 하드웨어 설계 방법을 제안하였다. 제안된 하드웨어 설계 방법은 기존 승산기를 사용한 힐버트 변환기 설계에 필요한 gate 수에 따른 복잡도와 비교하여 제안한 하드웨어 설계 방법의 성능을 분석하였다.

2. 이산 힐버트 변환

2.1 이산 힐버트 변환

실함수(real function) $x(t)$의 힐버트 변환 쌍 $\tilde{x}(t)$는 다음과 같이 정의된다[7].

$$\tilde{x}(t) = -\frac{1}{\pi}\int_{-\infty}^{\infty}\frac{y(\tau)}{t-\tau}d\tau \tag{1}$$

$$H[x(t)] = \frac{1}{\pi t} * x(t)d\tau \tag{2}$$

위 식에서 *는 컨벌루션(convolution)을 나타내고, 이때 실함수 $x(t)$와 $\tilde{x}(t)$에 의해 구성된 복소함수는 해석함수(analytic function)가 된다. 식 (2)에 대한 이산 힐버트 변환은 식 (3)과 같이 정의된다.

$$y(k) = \sum_{s=0}^{N-1} x(s)h(k-s) \tag{3}$$

여기서, $h(k) = \frac{2}{\pi}\sum_{r=1}^{M}\sin\frac{2\pi rk}{N}$,

$$\begin{cases} M \triangleq N/2-1 & \text{for } N \ even \\ M \triangleq (N-1)/2 & \text{for } N \ odd \end{cases}$$

$$\begin{bmatrix} y(0) \\ y(1) \\ \cdot \\ \cdot \\ \cdot \\ y(N-1) \end{bmatrix} = \begin{bmatrix} \\ h_{ks} \\ \\ \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ \cdot \\ \cdot \\ \cdot \\ x(N-1) \end{bmatrix} \tag{4}$$

식 (3)의 이산 힐버트 변환은 입출력 샘플로 식 (4)와 같이 나타낼 수 있으며, 일반적인 힐버트 변환기의 설계의 수학적 모델로 사용하게 된다.

2.2 이산 힐버트 변환의 알고리즘의 구현

식 (4)는 h_{ks} 입력되는 N 샘플의 수에 따라 h_{ks}의 값은 달라지며 행렬 패턴 역시 달라진다. 이에 따라 각 N에 따른 h_{ks}와 입력 $x(k)$의 연산과정을 일반화시키기 위하여 $k=0,1,...,N-1$ 및 $s=0,1,...,N-1$에 대해 $H \triangleq [h_{ks}]$로 정의할 때 h_{ks} 다음과 같은 특징을 갖는다[7].

$$h_{ks}\begin{cases} -h_{sk} \\ h_{<k+1><s+1>}, & <k> \triangleq k \mod N \\ h_{ks}=0, & k=s \end{cases} \tag{5}$$

식 (3)의 $N=7,8$ 일 때를 가정하여 계산된 h_{ks}값은 각각 $H=[0, 0.6259, -0.0688, 0.1791, -0.1791, 0.0688, 0.6259]$이고 $H=[0, 0.6036, 0, 0.1036, 0, -0.1036, 0, -0.6036]$이며, 계산된 계수의 값이 서로 대칭을 이루는 특징을 나타낸다. 이 때 H는 식 (5)의 특징을 사용하여 다음과 같이 $N=7$ 일 때의 식 (6)과 $N=8$ 일 때의 식 (7)의 행렬로 표현할 수 있다.

$$H = \begin{bmatrix} 0 & h_0 & h_1 & h_2 & -h_2 & -h_1 & -h_0 \\ -h_0 & 0 & h_0 & h_1 & h_2 & -h_2 & -h_1 \\ -h_1 & -h_0 & 0 & h_0 & h_1 & h_2 & -h_2 \\ -h_2 & -h_1 & -h_0 & 0 & h_0 & h_1 & h_2 \\ h_2 & -h_2 & -h_1 & -h_0 & 0 & h_0 & h_1 \\ h_1 & h_2 & -h_2 & -h_1 & -h_0 & 0 & h_0 \\ h_0 & h_1 & h_2 & -h_2 & -h_1 & -h_0 & 0 \end{bmatrix} \tag{6}$$

$$H=\begin{bmatrix} 0 & h_0 & 0 & h_2 & 0 & -h_2 & 0 & -h_0 \\ -h_0 & 0 & h_0 & 0 & h_2 & 0 & -h_2 & 0 \\ 0 & -h_0 & 0 & h_0 & 0 & h_2 & 0 & -h_2 \\ -h_2 & 0 & -h_0 & 0 & h_0 & 0 & h_2 & 0 \\ 0 & -h_2 & 0 & -h_0 & 0 & h_0 & 0 & h_2 \\ h_2 & 0 & -h_2 & 0 & -h_0 & 0 & h_0 & 0 \\ 0 & h_2 & 0 & -h_2 & 0 & -h_0 & 0 & h_0 \\ h_0 & 0 & h_2 & 0 & -h_2 & 0 & -h_0 & 0 \end{bmatrix} \quad (7)$$

$$\begin{bmatrix} y(0) \\ y(1) \\ . \\ . \\ y(4) \\ y(5) \\ y(6) \end{bmatrix} = \begin{bmatrix} x(1)-x(6) & x(2)-x(5) & x(3)-x(4) \\ x(2)-x(0) & x(3)-x(6) & x(4)-x(5) \\ x(3)-x(1) & x(4)-x(0) & x(5)-x(6) \\ x(4)-x(2) & x(5)-x(1) & x(6)-x(0) \\ x(5)-x(3) & x(6)-x(2) & x(0)-x(1) \\ x(6)-x(4) & x(0)-x(3) & x(1)-x(2) \\ x(0)-x(5) & x(1)-x(4) & x(2)-x(3) \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \end{bmatrix} \quad (8)$$

$$\begin{bmatrix} y(0) \\ y(1) \\ . \\ . \\ y(4) \\ y(5) \\ y(6) \\ y(7) \end{bmatrix} = \begin{bmatrix} x(1)-x(7) & x(2)-x(6) & x(3)-x(5) \\ x(2)-x(0) & x(3)-x(7) & x(4)-x(6) \\ x(3)-x(1) & x(4)-x(0) & x(5)-x(7) \\ x(4)-x(2) & x(5)-x(1) & x(6)-x(0) \\ x(5)-x(3) & x(6)-x(2) & x(7)-x(1) \\ x(6)-x(4) & x(7)-x(3) & x(0)-x(2) \\ x(7)-x(5) & x(0)-x(4) & x(1)-x(3) \\ x(0)-x(6) & x(1)-x(5) & x(2)-x(4) \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \end{bmatrix} \quad (9)$$

여기에서 h_{ks}는 $h_{01} \triangleq h_0$, $h_{02} \triangleq h_1$, $h_{03} \triangleq h_2$으로 정의한다. 식 (6)과 식 (7)에서 $N=7$인 홀수 경우와 $N=8$인 짝수의 경우의 행렬식 패턴이 차이를 보이므로 식 (4)의 h_{ks}를 N에 대하여 홀수 경우와 짝수 경우를 구별하여 연산이 수행 된다.이에 따라 식 (6)과 식 (7)에서 일반화된 계수 h_{ks}는 h_0, h_1, h_2로 정의되며, 식 (6)과 식 (7)에 대한 입력 $x(k)$의 행렬은 식 (8)과 식 (9)와 같이 정리할 수 있다.

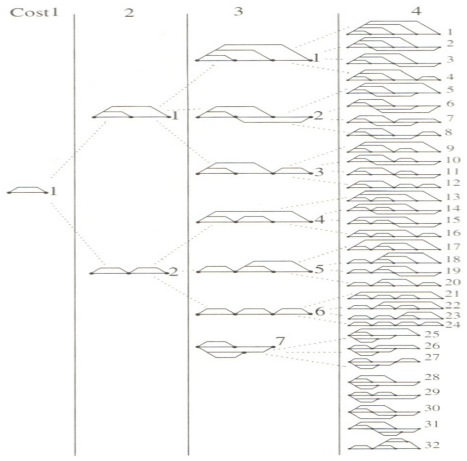
본 논문에서는 $N=7,8$ 일 때의 일반적인 힐버트 변환기의 각 설계를 위와 같은 과정을 통하여 일반화된 h_{ks}와 입력 $x(k)$의 승산 연산을 기존의 MAG 알고리즘과 add and shift의 승산 연산을 이용한 곱셈 블록으로 대체하여 구현하려 한다.

3. 제안된 힐버트 변환기의 설계

3.1 MAG 알고리즘

MAG 알고리즘은 최소한의 가산 연산을 위한 최적화된 과정을 2^n의 쉬프트 특징을 이용한 그래프 방식으로 식 (10)과 같이 각 단계의 계수를 만드는 가산 연산 과정에서 소비되는 불필요한 연산을 줄이는 알고리즘이다 [5,6].

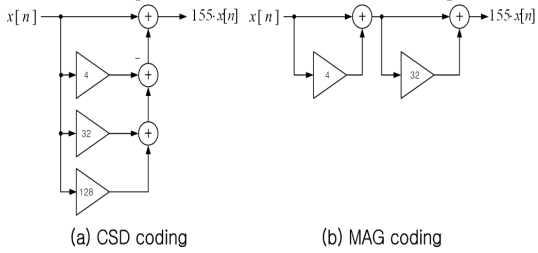
$$\begin{aligned} &cost1: 1)\ A=2^{k_0}(2^{k_1}\pm 2^{k_2}) \\ &cost2: 1)\ A=2^{k_0}(2^{k_1}\pm 2^{k_2}\pm 2^{k_3}) \\ &\qquad 2)\ A=2^{k_0}(2^{k_1}\pm 2^{k_2})(2^{k_3}\pm 2^{k_4}) \\ &cost3: 1)\ A=2^{k_0}(2^{k_1}\pm 2^{k_2}\pm 2^{k_3}\pm 2^{k_4}) \\ &\quad . \\ &\quad . \end{aligned} \quad (10)$$



[그림 1] 그래프 표현의 구현 가능한 각 단계의 계수

MAG 알고리즘은 하드웨어의 설계에서 가산 연산을 줄이기 위한 다른 알고리즘에 비해 효율적인 장점을 가지고 있다.

예를 들어 아래 그림 2는 본 논문에서 실제 적용된 계수 155를 CSD(canonical signed digit) 표현과 MAG 알고리즘을 적용한 각 곱셈 블록 설계에 대한 비교를 나타내었다.



(a) CSD coding (b) MAG coding

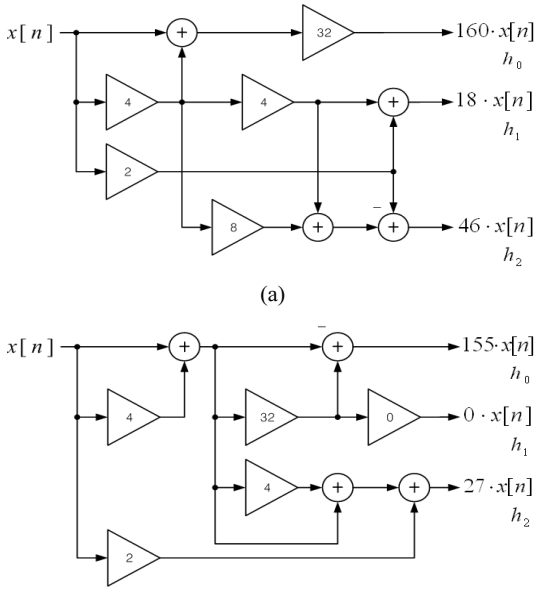
[그림 2] CSD표현과 MAG알고리즘을 적용한 각 설계의 비교

그림 2의 (a)에서 CSD 표현을 사용한 설계의 계수

$155_{10} = 10011011_2 = 10100\bar{1}0\bar{1}_{CSD}$로 표현되고 계수의 구성은 $155 = 128 + 32 + (-4) + (-1)$로 3번의 가산 연산과정과 4개(128, 32, -4, -1)의 구성요소가 필요하다. 그러나 그림 2의 (b)에서 MAG 알고리즘을 적용한 설계는 계수를 생성하는 과정이 계수의 구성요소와 무관하게 생성된다. 즉, $155_{10} = \{(4+1)\times 32\} + (-5)$로서 가산 연산 2번과 3개(5, -5, 32)의 요소로 구성되어 가산 연산을 줄일 수 있다. 이와 같은 특징은 기존 승산기를 설계할 때 가산기가 증가하는 단점을 보완하며, 본 논문에서 제안한 하드웨어 기반의 힐버트 변환기의 설계에서 유용하게 응용될 수 있다.

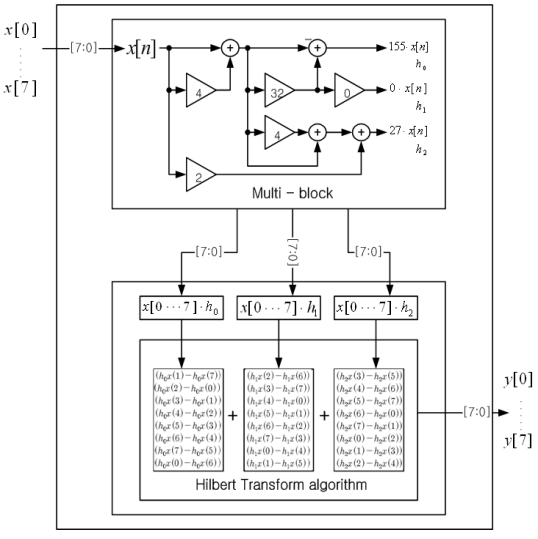
3.2 제안된 힐버트 변환기의 설계

식 (8)과 식 (9)를 이용한 일반적인 힐버트 변환기의 하드웨어 설계는 기존 승산기를 사용하여 구현되지만 제안된 힐버트 변환기의 하드웨어 설계는 기존 승산기를 MAG 알고리즘과 add and shift의 승산 연산을 적용한 곱셈 블록으로 대체하여 설계된다. 본 논문에서 $N = 7, 8$ 일 때 계수를 사용하여 설계한 힐버트 변환기의 각 곱셈 블록도는 다음에 나타낸 그림 3과 같이 설계된다.



[그림 3] $N = 7, 8$ 일 때, 각 곱셈 블록도
(a) $N = 7$ 일 때, 계수 $h_0 = 160, h_1 = 18, h_2 = 46$의 곱셈 블록
(b) N=8 일 때, $h_0 = 155, h_1 = 0, h_2 = 27$의 곱셈 블록

그림 3에서 사용된 계수는($N = 7$ 일 때, $h_0 = 160$, $h_1 = 18$, $h_2 = 46$ 및 $N = 8$ 일 때, $h_0 = 155$, $h_1 = 0$, $h_2 = 27$) 식 (3)의 $h(k)$값을 구하는 수식에 의해 계산된 계수 값으로서 정수연산을 위해 256을 승산 연산하여 얻어진 값이다. 이에 따라 식 (8)과 식 (9)에서 사용되는 승산연산은 MAG 알고리즘과 add and shift의 곱셈 블록으로 대체되어 하드웨어로 구현된다. 예를 들어 $N = 7$ 일 때, h_0, h_1, h_2는 입력 $x[7]$의 승산 연산과 같은 결과를 보인다. 즉, 입력$x(0)$은 곱셈 블록의 연산과정을 거치면 $x(0)\cdot h_0$(160), $x(0)\cdot h_1$(18), $x(0)\cdot h_2$(46)의 3번의 승산기를 사용한 결과와 동일한 출력을 1번의 곱셈 블록 과정을 통해 출력하게 된다. 또한, 일반적인 힐버트 변환기의 설계에 필요한 승산기의 수는 $N = 7, 8$ 일 때 각각 21, 24개의 승산기를 사용하는 반면에 MAG 알고리즘이 적용된 곱셈 블록을 사용한 힐버트 변환기는 각각 7, 8개의 곱셈 블록만으로 같은 출력결과를 얻을 수 있고, 기존 승산 연산의 1/2만이 필요하다. 본 논문에서 제안한 하드웨어 설계방법을 적용한 힐버트 변환기의 설계를 그림 4의 블록도로 나타내었다.



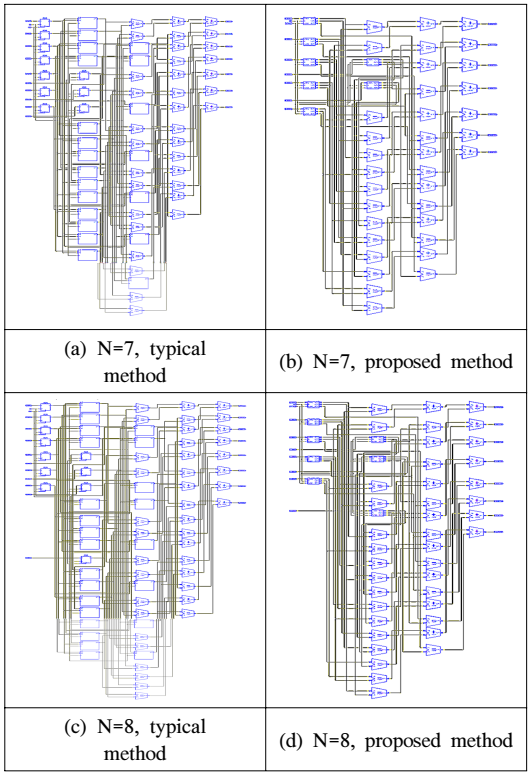
[그림 4] MAG 알고리즘에 의한 곱셈 블록을 적용하여 설계된 힐버트 변환기의 블록도

이와 같이 본 논문에서는 기존 승산기를 곱셈 블록으로 대체하여 구현된 힐버트 변환기는 MAG 알고리즘과 add and shift의 승산 연산에 의해 설계에 필요한 게이트 소자의 수가 감소되며, 하드웨어 설계면적과 복잡도를 줄이고, 높은 동작성능을 낼 수 있다.

4. 실험 및 결과 고찰

본 논문에서 일반적인 힐버트 변환기와 제안된 힐버트 변환기의 하드웨어 설계에 대한 실험 과정은 기존 승산기를 사용한 일반적인 설계와 MAG 알고리즘과 add and shift의 승산연산을 이용한 곱셈 블록이 적용된 설계방법을 $N=7,8$일 때를 VHDL로 각각 구현하여 비교 및 분석하였다.

설계한 이산 힐버트 변환기의 개발환경은 Xilinx FPGA Virtex-4(xc4vlx100-10)와 Xilinx ISE 9.2i에서 구현하였으며, Modelsim6.1f을 사용하여 시뮬레이션 하였고, 각각의 설계에 대해 게이트 소자 사용에 대한 비교와 하드웨어의 복잡도에 대한 비교 및 동작 주파수에 대한 비교, 분석을 수행하였다. 또한 각 설계의 비교 및 분석에 대한 결과를 그림 5와 그림 6 및 표 1 과 표 2에 나타내었다.



(a) N=7, typical method | (b) N=7, proposed method

(c) N=8, typical method | (d) N=8, proposed method

[그림 5] N= 7, 8 일 때, 각 설계의 복잡도 비교

그림 5의 (a)와 (c)는 $N=7,8$ 일 때, Xilinx ISE의 IPcore에서 제공하는 기존 승산기를 VHDL을 사용하여 구현한 일반적인 힐버트 변환기의 디지털 회로이고, (b)와 (d)는 MAG 알고리즘과 add and shift의 승산 연산을 이용한 곱셈 블록을 적용하여 VHDL로 구현한 제안된 힐버트 변환기의 디지털 회로이다. 그림 5에 나타난 바와 같이 일반적인 승산기를 사용한 힐버트 변환기의 설계는 입력 N이 증가함에 따라 힐버트 변환기 설계에 사용되는 승산기의 수는 증가 한다. 반면에 제안된 힐버트 변환기는 곱셈 블록을 사용함으로서 기존 승산기와 동일한 성능을 내지만 승산기 설계에 필요한 불필요한 가산기의 수를 줄여 설계하였기 때문에 N이 증가함에 따라 대체된 곱셈 블록의 수는 N에 비례하여 증가하지 않는다.

[표 1] N=7일 때, 각 힐버트 변환기의 설계에 사용된 게이트 소자 수 비교

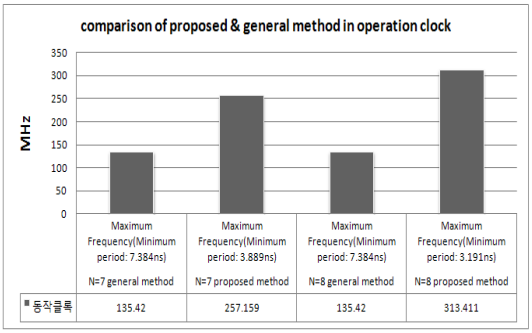
N	general method		proposed method	
N=7	Number of Slices	1454	Number of Slices	553
	Number of Slice Flip Flops	664	Number of Slice Flip Flops	483
	Number of 4 input LUTs	2590	Number of 4 input LUTs	855
	Number of bonded IOBs	218	Number of bonded IOBs	191

[표 2] N=8일 때, 각 힐버트 변환기의 설계에 사용된 게이트 소자 수 비교

N	general method		proposed method	
N=8	Number of Slices	1662	Number of Slices	521
	Number of Slice Flip Flops	785	Number of Slice Flip Flops	448
	Number of 4 input LUTs	2960	Number of 4 input LUTs	937
	Number of bonded IOBs	245	Number of bonded IOBs	218

따라서 $N=7,8$ 일 때, 표 1과 표 2의 결과에서 기존 승산기를 사용하여 설계된 힐버트 변환기는 하드웨어 설계에 사용되는 게이트 소자의 수가 입력 N에 따라 증가하며 기존 승산기의 불필요한 가산 연산은 하드웨어 설계 면적의 증가와 설계에 필요한 게이트 소자의 수가 증가시켜 하드웨어를 복잡하게 하고 성능을 저해한다. 반면에 제안된 힐버트 변환기의 하드웨어 설계는 MAG 알고리즘을 적용하여 불필요한 가산 연산을 줄이고 add and shift의 승산 연산을 이용한 곱셈 블록을 적용하여 설계에 사용되는 게이트 소자의 수의 감소와 동작성능의 향상을 확인 할 수 있으며 동일한 출력결과를 내는 제안된 힐버

트 변환기 설계에서 하드웨어의 복잡도 감소를 확인 할 수 있다. 그리고 그림 6은 최대 동작 주파수를 비교하여 나타내었다.



[그림 6] 제안된 설계와 일반적인 설계의 동작클록 속도 비교 그래프

그림 6의 동작 주파수에 대한 비교에서 알 수 있듯이 제안된 하드웨어 설계 방법이 적용된 힐버트 변환기 설계에서 게이트 소자의 수가 감소함에 따라 소자에서 발생되는 지연시간이 감소하여 동작 주파수가 증가하는 것으로 판단되며, 동작 주파수 증가에 따른 연산 속도 향상을 기대 할 수 있다.

5. 결론

본 논문에서는 MAG 알고리즘을 사용한 곱셈 블록을 힐버트 변환의 승산 연산에 적용하여 하드웨어 기반의 힐버트 변환기를 설계하여 하드웨어의 복잡도와 연산량을 줄여 게이트 소자 사용을 감소시킬 수 있는 설계방법을 제안하고, 구현하였다. 제안된 결과를 바탕으로 구현에 사용된 로직 셀을 1/2 이상 감소시킬 수 있었고, 동작 주파수의 증가로 연산속도 향상을 기대 할 수 있다. 특히, 모바일 디지털 기기의 SOC에서 필수요소인 디지털 신호처리 시스템의 하드웨어 설계에 응용된다면 효율적인 하드웨어 설계와 설계면적 감소 그리고 저 전력 설계를 구현할 수 있고 또한, 복잡도 감소에 따른 개발 기간 및 비용 감소를 기대할 수 있을 것으로 판단된다.

참고문헌

[1] D. Kodek and K. Steigliz, "Comparison of Optimal and Local Search Methods for Designing Finite Wordlength FIR Digital Filters," IEEE Transactions on Circuits and Systems, Vol. 28, pp.28-32, Jan. 1981.

[2] A. de la Serna and M. A. Soderstrand, "Tradeoff Between FPGA Resource Utilization and Roundoff Error in Optimized CSD FIR Digital Filters," IEEE Asilomar Conference, Vol. 1, pp. 187-191, 1994.

[3] I. Richard and Hartley, "Subexpression Sharing in Filters Using Canonic Signed Digit Multipliers," IEEE Transaction on Circuits and Systems, Vol. 43, No. 10, Oct. 1996.

[4] P.K. Dutta and P.B. Duttagupta, “Optimization Method for Broadband Modem FIR Filter Design Using Common Subexpression Elimination," IEEE Instrumentation and Measurement Technology Conference, Vol. 3, pp. 1321-1324, 1994.

[5] D. R. Bull and D. H. Horrocks, “Primitive operator digital filter,” IEEE Proc. G., vol. 138, no. 3, pp. 401–412, June 1991.

[6] A. G. Dempster and M. D. Macleod, “Use of minimum-adder multiplier blocks in FIR digital filters,” IEEE Trans. Circuits Syst., vol. 42, no. 9, pp. 569–577, Sept. 1995.

[7] N,K. Bose, Digital Filters Theory and Applications, North-Holland Ltd.,1985.

김 영 웅(Young-Woong Kim) **[준회원]**

- 2008년 7월 : 청운대학교 전자공학과 (공학사)
- 2009년 3월 ~ 현재 : 청운대학교 전산·전자공학과(공학석사 재학)

<관심분야>
임베디드 시스템, SOC

이 영 석(Young-Seock Lee) **[정회원]**

- 1993년 2월 : 서울시립대학교 전자공학과 (공학사)
- 1995년 2월 : 서울시립대학교 대학원 전자공학과 (공학석사)
- 1998년 2월 : 서울시립대학교 대학원 전자공학과 (공학박사)
- 1998년 3월 ~ 현재 : 청운대학교 디지털방송공학과 교수

<관심분야>
SOC, 임베디드 시스템, 의용생체시스템,VLSI 신호처리