논문 2004-41SD-7-8

고속 디지털 신호처리를 위한 MBA기반 병렬 MAC의 효율적인 구조

(A Efficient Architecture of MBA-based Parallel MAC for High-Speed Digital Signal Processing)

서 영 호*, 김 동 욱*

(Young-Ho Seo and Dong-Wook Kim)

요 약

본 논문에서는 고속의 곱셈-누적 연산을 수행할 수 있는 새로운 MAC(Multiplier-Accumulator)의 구조를 제안하였다. 부분곱의 생성을 위해서 1의 보수 기반의 고속 Booth 알고리즘(Modified Booth Algorithm, MBA)를 이용하였고 다수의 부분곱을 더하기 위해서 CSA(Carry Save Adder)를 이용하였다. 부분곱을 더하는 과정에서 Booth 인코딩 시 이용한 1의 보수 체계를 2의 보수 체계로 보상하고 이전 합과 캐리를 누적하는 연산을 수행하여 고속의 누적 연산이 가능한 구조를 제안한다. 또한 부분곱의 덧셈에서 하위 비트들을 2 비트 CLA(Carry Look-ahead Adder)를 이용하여 연산함으로써 최종 덧셈기의 입력 비트수를 줄임으로써 전체적인 임계경로를 감소시켰다. 제안된 MAC을 JPEG2000을 위한 DWT(Discrete Wavelet Transform) 필터링 연산에 적용하여 고속의 디지털 신호처리가 가능함을 보였고 기존의 연구와 비교하여 향상된 성능을 보이는 것을 확인하였다.

Abstract

In this paper, we proposed a new architecture of MAC(Multiplier-Accumulator) to operate high-speed multiplication-accumulation. We used the MBA(Modified radix-4 Booth Algorithm) which is based on the 1's complement number system, and CSA(Carry Save Adder) for addition of the partial products. During the addition of the partial product, the signed numbers with the 1's complement type after Booth encoding are converted in the 2's complement signed number in the CSA tree. Since 2-bit CLA(Carry Look-ahead Adder) was used in adding the lower bits of the partial product, the input bit width of the final adder and whole delay of the critical path were reduced. The proposed MAC was applied into the DWT(Discrete Wavelet Transform) filtering operation for JPEG2000, and it showed the possibility for the practical application. Finally we identified the improved performance according to the comparison with the previous architecture in the aspect of hardware resource and delay.

Keywords : Multiplier-Accumulator, Booth Multiplier, Digital Signal Processing, CSA Tree, Computer Arithmetic

I. 서 론

최근 컴퓨터와 통신시스템 분야가 급속도로 발전함에 따라서 디지털 신호처리, 영상처리, 그리고 3차원 처리 등과 같은 실시간 처리 및 대용량의 데이터 처리를 위하여 디지털 시스템의 고속처리 능력이 더욱 요구된다. 특히 곱셈기는 디지털 신호처리를 위한 필수적인 요소이고 이 곱셈기를 이용한 곱셈-누적 연산[1]은 디지털 신호처리 분야에서 필터링, 컨벌루션, 그리고 내적 등을 비롯한 모든 연산의 기본을 이룬다. 멀티미디어에 대한 관심이 높아지고 있고 수요가 늘어나면서 MPE-G[2] 혹은 JPEG2000[3]등과 같은 대규모 데이터들의 실시간 처리를 위해서는 DCT(Discrete Cosine Transfo-rm) 혹은 DWT과 같은 비선형 연산이 필요하게 된다.

* 종신회원, 광운대학교 Digital Design & Test Lab.
(Digital Design & Test Lab, Kwangwoon Univ.)
※ 이 논문은 2004학년도 광운대학교 교내 학술연구비 지원에 의해 연구되었음.
접수일자: 2004년2월2일, 수정완료일: 2004년6월7일

또한 이러한 비선형 함수의 계산은 곱셈과 덧셈을 반복적으로 수행함으로써 이루어지기 때문에 덧셈이나 곱셈의 단위 연산 시간이 전체의 연산 수행 속도를 좌우하게 된다. 일반적으로 곱셈기의 속도는 디지털 시스템을 구성하는 기본적인 연산 블록들 중에서 가장 긴 지연시간을 가지면서 전체 시스템의 임계경로(Critical path)를 결정하게 되는데 빠른 곱셈연산을 위해서 "Modified Radix-4 Booth Algorithm(MBA)[4]을 쓰는 것이 일반적이지만 그렇다 할지라도 곱셈 연산에 소요되는 긴 임계경로에 대한 문제를 완전히 해결할 수는 없다[5][6].

고속 곱셈을 위한 곱셈기 구조는 배열 곱셈기와 병렬 곱셈기로 나눌 수 있다[7]. 배열 곱셈기는 단위 셀의 규칙성 및 VLSI 구현이 쉽다는 장점을 가지고 있으나 계산 속도가 곱해지는 비트수에 비례한다는 단점을 가지고 있다. 병렬 곱셈기의 경우 곱셈 비트수가 증가함에 따라 곱셈 시간이 배열 곱셈기보다 적다는 장점을 가지고 있다[8]. 따라서 특정 목적에 따라 다양한 형태의 하드웨어 구조가 제안되고 있으며 응용 목적에 따라 적절한 선택이 요구된다. 즉, 파이프라인 방식의 병렬 곱셈기의 경우 특정 DSP 알고리듬 구현에 적합하고 트리구조의 병렬 곱셈기의 경우 마이크로프로세서나 범용 DSP 프로세서에 적합하다.

일반적으로 곱셈기에는 Booth 알고리듬[9]과 전가산기의 배열, 또는 Booth 알고리듬과 Wallace 트리[10]를 이용한 방법이 많이 이용되고 있으며 이러한 곱셈기는 크게 Booth 인코더, 부분곱 압축 트리, 최종 덧셈기의 세부분으로 구성된다[11][12]. Wallace 트리는 인코더로부터 나오는 부분곱들의 덧셈을 최대한 병렬로 수행하는데 (N-bit) 데이터의 곱셈을 위하여 $O(log_2N)$에 비례하는 수행시간을 가진다. 즉, CSA를 사용하여 N개의 입력 중에서 1의 개수를 카운트하여 출력으로 내보내면 출력 비트의 개수는 log_2N개로 줄어드는 원리를 이용한다. 실제로는 다수의 비트수를 입력으로 사용하는 CSA를 구현하는데 많은 면적이 필요하므로 (3:2) 혹은 (7:3) 카운터를 다수 사용하여 파이프라인 단계마다 이러한 카운터의 개수가 줄어들도록 하고 있다. 곱셈 과정은 일련의 부분곱 덧셈 과정을 반복적으로 수행하는 것이기 때문에 고속 승산을 위해 부분 곱의 수를 줄임으로서 계산 단계를 줄이는 MBA 알고리즘이 적용되고 있으며 또한 Wallace 트리를 적용하여 부분곱의 빠른 덧셈을 수행하고 있다. 최근 MBA의 속도를 높이기 위해 병렬 곱셈구조가 많이 연구되고 있다[13][14][15]. 일반적으로 "Baugh-Wooley Algorithm(BWA)"에 기초하여 MAC의 구조가 개발되어 왔고 제안된 구조들은 다양한 디지털 필터링 연산에 도입되었다[16][17][18].

범용적인 디지털 신호처리를 위한 가장 발달된 형태의 MAC 중에 하나가 [19]에서 제안되었는데 부분곱을 압축하는 CSA 트리에 누적 연산을 결합한 구조를 제시하고 있다. 본 논문에서는 [19]의 구조를 바탕으로 고속의 곱셈-누적 연산을 수행할 수 있는 새로운 MAC의 구조를 제안한다. 즉, 다양한 구조의 MAC이 연구되어 왔고, 각기 다른 구조적인 특징을 가지고 있는데, 본 논문은 그 중에서 병렬 곱셈기 방식을 따르고 있고, 그 병렬 곱셈기 방식 중에서도 부분곱을 더하는 CSA의 하이브리드 구조를 개선하고자 한다. [19]와 마찬가지로 부분곱의 생성을 위해서 1의 보수 기반의 MBA 알고리즘를 이용하였고 다수의 부분곱을 더하기 위해서 합과 캐리형태의 누적 연산을 수행하는 하이브리드 형태의 CSA 구조를 바탕으로 새로운 MAC의 구조를 제안한다.

Ⅱ. Booth 곱셈기와 MAC의 구조

곱셈기는 크게 세 부분으로 나눌 수 있는데, 첫 번째는 입력으로 들어오는 피승수와 승수로부터 부분곱들을 만들어내는 인코더 부분이고 두 번째는 생성된 부분곱들을 모두 더하여 합과 캐리형태의 값으로 만드는 덧셈기 배열 혹은 부분곱 압축 부분이다. 그리고 마지막은 합과 캐리를 더하고 최종적인 곱셈결과를 만드는 최종 덧셈기 부분이다. 여기에 곱한 결과를 누적하는 과정을 포함시키면 총 네 가지 단계로 나누어진다고 볼 수 있고 이러한 단계를 그림 1에 나타내었다. 그림 1은 앞서 설명한 각 단계를 나타내는 것으로 순서대로 Step1, Step2, Step3, 그리고 Step4에 해당한다.

그림 2에는 일반적인 MAC의 구조를 나타내었다. 입력되는 승수, X와 피승수, Y를 곱하여 곱셈결과, Z를 연산하고 이를 이전 곱셈결과인 Z'과 더하여 누적 연산을 수행한다. 여기서 N 비트의 크기를 가지고 2의 보수 체계를 사용하는 2진수, X를 식(1)과 표현할 수 있다.

$$X = -2^{N-1}x_{N-1} + \sum_{i=0}^{N-2} x_i 2^i, x_i \in \{0,1\} \quad (1)$$

식 (1)을 Booth 알고리즘을 적용하기 위하여 Base-4 형태의 Redundant Sign digit 방식으로 나타내면 식 (2) 및 (3)과 같이 나타낼 수 있고 앞서 설명한 곱셈-누적

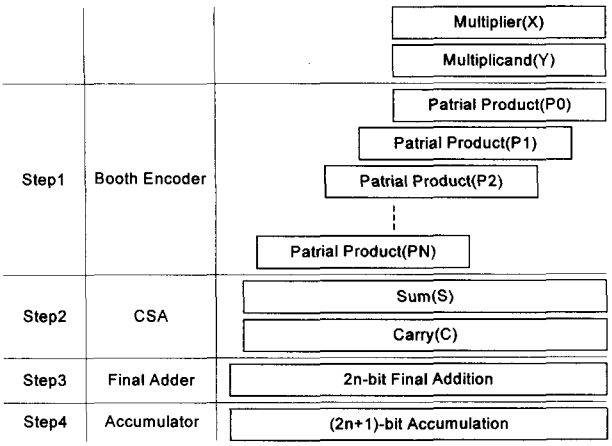


그림 1. 일반적인 곱셈과 누적 연산과정
Fig. 1. General arithmetic operation of multiplication and accumulation.

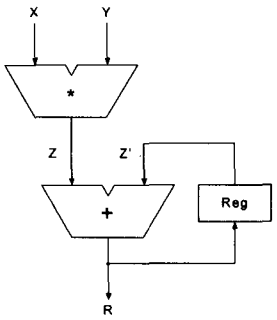


그림 2. 일반적인 MAC의 하드웨어 구조
Fig. 2. General hardware architecture of MAC.

연산의 결과는 식 (4)와 같이 표현할 수 있다.

$$X=\sum_{i=0}^{N/2-1} d_i 4^i \tag{2}$$

$$d_i=-2x_{2i+1}+x_{2i}+x_{2i-1},\ d_i\in\{0,\ \pm 1,\ \pm 2\} \tag{3}$$

$$P=X\times Y+Z=\sum_{i=0}^{N/2-1} d_i 2^{2i}Y+\sum_{j=0}^{2N-1} z_i 2^i \tag{4}$$

식 (4)를 구성하는 두 항들은 각각 독립적으로 연산되고 두 항의 결과가 결정되면 두 항이 덧셈연산, 즉 누적 연산을 수행하여 최종적인 연산을 마친다. 식 (4)와 같은 방식으로 구현된 구조를 표준 설계(standard design)라 한다.

기본적으로 N-bit 길이의 데이터끼리의 곱셈에서는 N에 비례하는 개수의 부분곱들이 만들어지고 이들을 순차적으로 더하기 위하여 역시 N에 비례하는 덧셈수행시간이 걸린다. 현재까지 고안된 곱셈기 중 가장 빠른 것으로 알려진 곱셈기의 형태는 부분곱들을 생성하는 Booth 인코딩[4][9]를 사용하고 부분곱들을 더하는 덧셈기 배열에는 CSA 기반의 Wallace 트리를 이용하는 방식[10]이다. Booth 인코딩 방법을 사용하면 이러한 Wallace 트리의 입력에 해당하는 부분곱의 수가 반으로 줄어들어 CSA 트리의 단계가 감소하고 2의 보수 기반의 부호를 가진 데이터들에 대한 곱셈도 가능하다. 이러한 이유로 인하여 현재의 거의 모든 곱셈기들에서는 Booth 인코딩 방식을 사용하고 있으며 이보다 더 좋은 성능을 가지는 인코딩 방식을 찾아내기는 쉽지 않을 것으로 보인다.

III. 제안된 MAC의 구조

제안된 MAC 구조는 2장에서 언급한 표준 설계라 부르는 일반적인 구조 및 누적 연산을 부분곱 연산에 포함시킨 [19]와 비교하였다. [19]에서 제안된 구조는 MBA를 이용하여 부분곱을 생성한 후 하이브리드 형태의 CSA 트리 구조를 이용하여 고속의 부분곱 덧셈과 최종 덧셈을 수행한다. 본 논문에서 제안된 구조 및 [19]에서 제안된 구조는 그림 2에서 나타낸 것과 같은 곱셈-누적 방식을 따르지만 최종 덧셈기(Final Adder)를 적용하는 방식에 있어서 차이를 가지고 있고 이로 인해 CSA 트리 구성 자체에서 차이를 가진다. 먼저, 2장에서 설명한 것과 같이 표준 설계에서는 곱셈과 누적 연산이 구조적으로 분리되어 있는 반면에 [19]와 본 논문에서 제안한 구조에서는 그림 3과 같이 통합된 구조를 가진다는 것이다. 식 (4)의 두 항에서 첫 번째항은 부분곱을 더하는 과정을 나타낸 것으로 N 비트 연산의 경우에 식 (5)와 같이 나열할 수 있다. 그리고 식 (4)의 두 번째항은 (2N-1)비트 크기의 이전 누적 결과를 나타내는데 이를 식 (6)과 같이 MSB부분과 LSB부분으로 분리할 수 있다. 마지막으로 식 (4)를 식 (5)와 식 (6)을 이용하여 식 (7)과 같이 표현할 수 있다. 식 (7)은 식 (4)와 동일한 연산을 나타내지만 부분곱을 더하는 부분과 누적연산을 독립적으로 수행하지 않고 부분곱의 연산과정 중에 누적 연산을 미리 포함시켜 연산하는 특성을 가진다.

$$X\times Y=\sum_{i=0}^{N/2-1} d_i 2^{2i}Y \tag{5}$$
$$=d_0 2Y+d_1 2^2 Y+d_2 2^4+\cdots+d_{N/2-1}2^{N-2}Y$$

$$Z=\sum_{i=0}^{2N-1} z_i 2^i=\sum_{i=0}^{N} z_i 2^i+\sum_{i=N+1}^{2N-1} z_i 2^i \tag{6}$$

$$P=X\times Y+Z$$
$$=(d_0 2Y+\sum_{j=0}^{N} z_i 2^i)+\sum_{i=1}^{N/2-2} d_i 2^{2i}Y+(d_{N/2-1}2^{N-2}Y+\sum_{i=N+1}^{2N-1} z_i 2^i) \tag{7}$$

식 (7)과 같은 접근은 MAC 연산(N×N)에서 곱셈 후에 큰 비트수로 이루어지는 누적연산(2N-1+2N-1)에 대한 병목현상을 해결하고자 하는 측면에서 전개한 것이다. 또한 [19]에서 제안된 구조는 CSA 트리에서 하위 비트들을 2-bit CLA를 이용하여 고속으로 캐리를 전파시켜 최종 덧셈기의 연산에 대한 부담을 덜어주어 속도를 높이고자 한 것이다. 하지만 이 경우에도 결국에는 CSA 트리의 하위비트들에 대한 연산이 끝나고 최종적인 최상위 비트에 대한 캐리전파가 이루어져야 최종 덧셈을 수행할 수 있기 때문에 표준 설계에 비해서는 효율성을 높였다고 하지만 많은 개선점을 가진다. 즉, 식 (6)을 살펴보면 누적값, Z를 두개의 항으로 독립적으로 분리하였으나 이는 Z의 값이 완전히 연산된 후에만 가능하다. 식 (7)의 결과가 다음 누적 연산을 위한 식 (6)의 Z 값에 해당되는데 식 (7)이 완전히 연산되어야만 식 (6)과 같이 분리가 가능하다. 수식적으로 식 (6)과 (7)의 관계는 아무런 문제가 없어 보이지만 하드웨어적인 측면에서 살펴보면 연산의 시간적인 순서에 따른 제약을 가지는 것이다. 이에 대해서 본 논문에서는 CSA 트리에서 하위비트들에 대한 캐리 전파가 끝난 후 최종 덧셈과정을 거치지 않고 합과 캐리 형태의 누적연산을 수행할 수 있는 CSA 트리 구조를 제안하여 고속 연산이 가능하게 한다. 즉, 앞서 설명한 것과 같이 식 (7)의 연산이 완전히 끝나지 않은 상황에서 다음 누적 연산을 위해서 식 (6)과 같이 Z를 분리할 수 있도록 하는 것이다. 식 (6)으로부터 Z를 합과 캐리의 연산형태로 재분리하면 식 (8)과 같다. 식 (5)와 식 (8)을 이용하여 식 (4)를 재구성하면 식 (9)와 같고 식 (5)와 (9)가 본 논문에서 제안하는 MAC의 연산 방식이다. 식 (9)에서 볼 수 있듯이 합과 캐리형태로 분리된 누적값, Z가 부분곱의 덧셈과정에 분해되어 누적 연산과정을 거친다. 또한 식 (7)로 인해 발생하는 식 (6)의 지연이 합과 캐리의 분리로 인해 제거되어 고속의 곱셈-누적 연산이 가능하다. 여기에서 $Y_{(A,B)}$는 2진수 Y의 A에서 B 비트까지를 뜻한다. 식 (9)에서 제안된 CSA 트리의 구조를 그림 4에 나타냈다. 그림 4에서 Sx는 부호 확장을 간략하게 하기 위한 보상값이고 Nx는 Booth 인코딩 시 사용했던 1의 보수를 2의 보수로 보상하는 값이다.

$$
\begin{aligned}
Z &= \sum_{i=0}^{2N-1} z_i 2^i = \sum_{i=0}^{N-1} z_i 2^i + \sum_{i=N}^{2N-1} z_i 2^i \\
&= \sum_{i=0}^{2N-1} z_i 2^i + \left(\sum_{i=0}^{2N-2} s_i 2^i + \sum_{i=0}^{2N-2} c_i 2^{i+1}\right) \\
&= \sum_{i=0}^{2N-1} z_i 2^i + \sum_{i=0}^{N-2} s_i 2^i 2^N + \sum_{i=0}^{N-2} c_i 2^{i+1} 2^N
\end{aligned} \tag{8}
$$

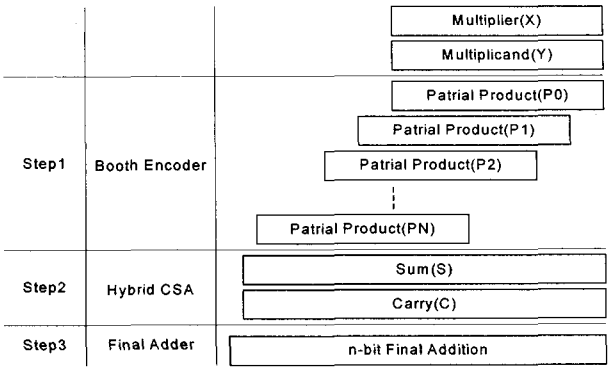


그림 3. 제안된 곱셈과 누적 연산과정
Fig. 3. The proposed arithmetic operation of multiplica-tion and accumulation.

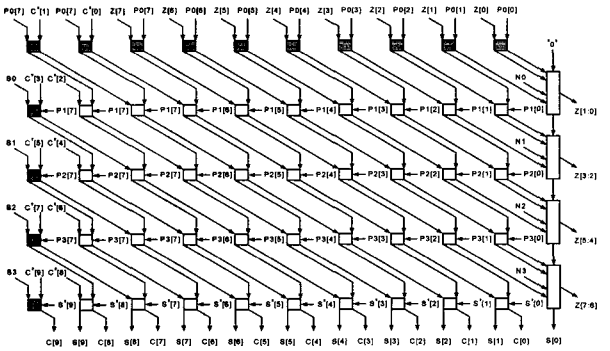


그림 4. 제안된 하이브리드 형태의 CSA 트리의 구조
Fig. 4. The proposed architecture of hybrid-type CSA tree.

$$
\begin{aligned}
P &= X \times Y + Z \\
&= \sum_{i=0}^{N/2-1} d_i 2^{2i} Y + \sum_{i=0}^{2N-1} z_i 2^i \\
&= \sum_{i=0}^{N/2-1} d_i 2^{2i} Y + \sum_{i=0}^{2N-1} z_i 2^i + \sum_{i=0}^{N-2} s_i 2^i 2^N + \sum_{i=0}^{N-2} c_i 2^{i+1} 2^N \\
&= (d_0 2 Y_{(0,N-1)} + \sum_{i=0}^{N-1} z_i 2^i \\
&+ \sum_{i=0}^{N/2-1} (d_i 2^{2i} Y_{(N,N+1)} + \sum_{j=0}^{1} c_{2i+j} 2^{2i+j} 2^N) + \sum_{i=0}^{N-2} s_i 2^i 2^N
\end{aligned} \tag{9}
$$

그림 3의 MAC 구조는 그 특성상 파이프라인형식의 병렬처리가 가능한데 그림 5에 나타낸 것과 같이 Booth 엔코더, CSA 트리, 그리고 최종 덧셈의 세 단계로 파이프라이닝을 수행하도록 설계하였다.

Ⅳ. 이산 웨이블릿 변환에의 적용

제안된 MAC의 구조를 디지털 신호처리의 핵심적인 사용분야인 필터링에 적용하였다. 디지털 신호처리를 위한 하드웨어를 구현할 경우에 알고리즘을 상위 수준의 프로그래밍 언어와 동일하게 표현할 수 없다. 따라서 시뮬레이션을 통하여 알고리즘을 간략화시킨 후에

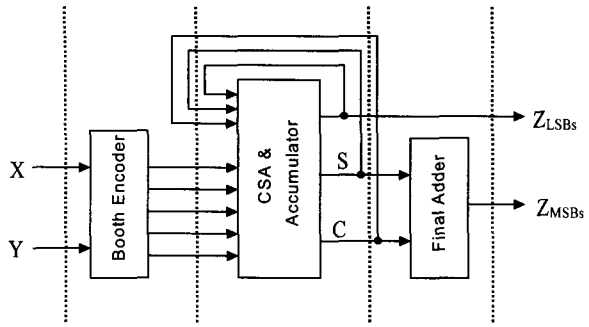
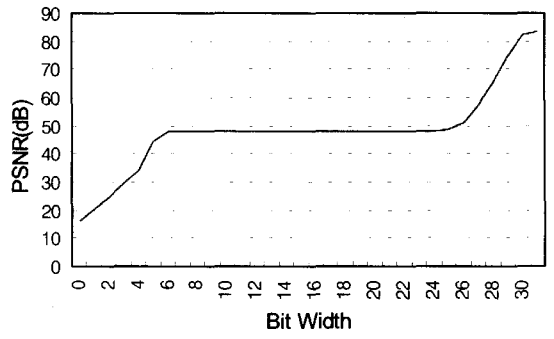
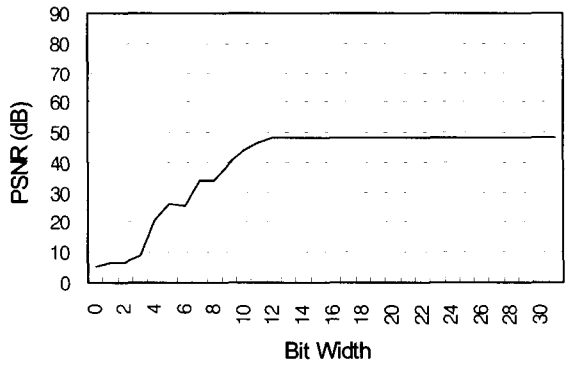


그림 5. 제안된 MAC의 파이프라인 단계
Fig. 5. Pipeline stage of the proposed MAC.



(a)



(b)

그림 6. 계수의 정밀도 분석 (a) 필터 (b) 웨이블릿 계수
Fig. 6. Precision analysis of filter coefficient (a) filter coefficient (b) wavelet coefficient.

허용할 수 있는 정밀도 범위 내에서 하드웨어로 구현한다.

JPEG2000에서 주파수 변환방법으로 채택된 이산 웨이블릿 변환은 컨벌루션 방식과 리프팅 방식으로 구현이 되는데 두 경우 모두 일반적인 필터링 동작으로 수행된다. 따라서 DWT에 제안된 알고리즘을 적용하는데 있어서 먼저 연산 결과에 영향을 미치는 필터 계수와 웨이블릿 계수를 고정소수점 시뮬레이션을 통해서 비트 제약이 성능에 미치는 영향을 분석했고 이를 바탕으로

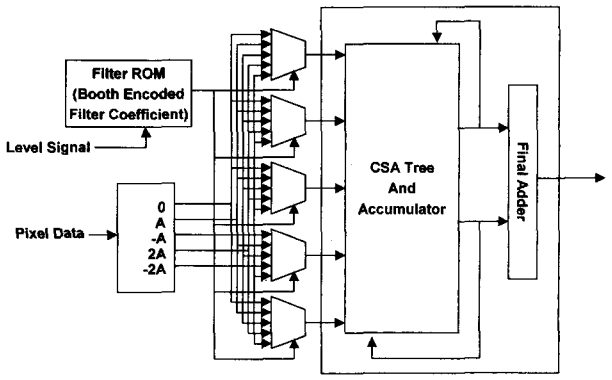


그림 7. DWT를 위한 제안된 MAC의 구조
Fig 7. The proposed MAC architecture for DWT.

하드웨어 구현을 위한 수 체계를 확립하였다. 그 결과를 그림 6에 나타내었는데 그림 6의 (a)에는 필터계수, 그리고 그림 6의 (b)에서는 웨이블릿 계수의 고정소수점 시뮬레이션을 통한 PSNR 결과를 나타냈다. 실험 환경은 C++ 언어를 이용하여 수행되었고 500개의 정지영상을 이용하였다. 또한 무손실 압축을 위해 사용되는 (5,3)필터는 실험에서 제외되었고 손실압축을 위한 (9,7) 필터만이 실험에 사용되었다. 그림 6에 보이는 것과 같이 필터 계수의 경우에 소수점 이하 비트가 8비트 이상이면 손실압축에 아무런 문제가 없을 정도의 성능을 보인다. 또한 웨이블릿 계수가 소수점 이하 13비트 이상으로 표현된다면 거의 화질에 영상을 주지 않고 일반적인 영상압축 위해서 30dB 이상의 화질을 요구할 경우에 7비트 이상으로 표현될 수 있다. 따라서 필터계수는 정수 2비트를 포함하여 10비트 그리고 웨이블릿 계수는 정수 9비트를 포함하여 16비트로 정하였다.

DWT에 최적화된 MAC의 구조를 그림 8에 나타내었는데, 이 MAC은 필터계수라는 일정한 값을 곱하기 때문에 일정부분 미리 Booth 인코딩을 수행한 후 필터 ROM에 저장한 뒤 병렬적으로 하나를 선택하여 사용한다. 또한 그림 7에 최적화된 CSA 트리구조를 나타내었는데, 그림에 나타낸 것과 같이 합(S[13:0])과 캐리(C[13:0])를 귀환시켜 다음의 부분곱 결과와 같이 덧셈을 수행하도록 하였다. 하위 10 비트(Z[9..0])는 CLA를 사용하여 미리 계산하도록 하여 최종 덧셈에 입력되는 데이터의 비트수를 현저히 감소시킨다. 일반적으로 최종 덧셈의 비트수는 임계경로를 형성하여 전체 회로의 동작주파수에 큰 영향을 끼치게 되는데, 본 구조로 임계경로의 지연시간이 상당히 감소할 수 있고 Motion JPEG2000등을 위한 효율적인 실시간 솔루션이 될 수 있다.

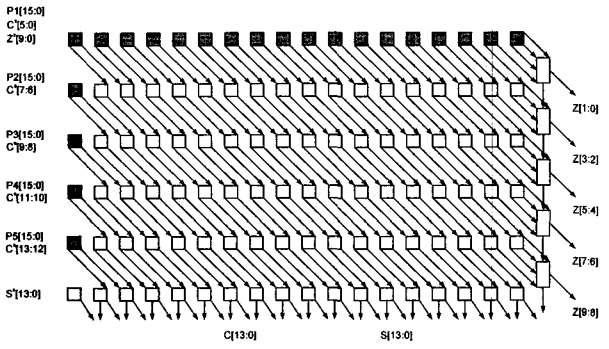


그림 8. DWT 필터링을 위한 최적화된 CSA의 구조
Fig. 8. Optimized CSA architecture of DWT filtering.

V. 하드웨어 자원 및 지연 모델

본 장에서는 제안된 MAC의 구조와 기존의 연구들과의 비교를 수행하여 기존의 연구들로부터 개선하고자 한 사항들이 구조적으로 반영되었는지 살펴본다. 기존의 연구들 중에서 가장 대표적인 병렬 MBA 구조의 표준 설계들[6]과 가장 발달된 형태인 [19]의 구조와 비교하고자 한다.

1. 하드웨어 자원

표 1에 앞서 언급된 4가지 하드웨어 구조에 대한 자원을 비교하였다. 하드웨어 자원은 각각의 구조를 구성하는 하드웨어 성분들을 모두 포함시킨 후에 일반적인 구조와 16비트 구조에 대한 결과를 나타냈다. 또한 16비트 구조의 경우에는 Hynix 0.35μm CMOS 라이브러리를 이용하여 최적화된 형태로 실제 합성한 후에 측정된 게이트 수를 바탕으로 나타냈다. 측정된 게이트 수에 대한 정보를 표 2에 나타냈는데 NAND 게이트를 게이트 측정 단위로 사용하고 있다. 표 2에 보이는 것과 같이 반가산기의 경우에 4개의 게이트 수를 보이고 있고 전가산기의 경우는 그 두 배인 8개의 게이트 수를 보인다. 또한 CLA의 경우에는 비트수가 증가함에 따라서 거의 선형적으로 게이트 수가 증가함을 볼 수 있다.

표 1를 보면 일반적으로 예측할 수 있는 것과 같이 표준설계가 가장 많은 하드웨어 자원을 점유하는 것을 알 수 있고 [19]의 구조와 제안된 구조가 거의 유사한 하드웨어 자원을 사용하는 것을 볼 수 있다. [19]의 구조는 CSA 트리 내에 (n+2)-bit의 최종 덧셈기가 존재하는 반면에 제안된 구조는 (3/2)n+2개의 HA가 존재하고 부분곱 연산과 누적 연산 후에 최종적인 합과 캐리를 더하는 최종 덧셈기가 존재하기 때문으로 그들의 하드웨어 자원수는 유사하다. 하지만 본 논문에서 제안된

표 1. MAC에 대한 하드웨어 자원 비교
Table 1. comparison of hardware amounts for MAC.

	Standard design		Design in [19]		Our design		Our design for DWT filtering	
	General	16-bit	General	16-bit	General	16-bit	General	16-bit
# of FA	n2/2+n	1152	n2/2+2n+3	1304	n2/2+n/2	1088	n2/4+(5/4)n+1	680
# of HA	0	0	0	0	(3/2)n+2	104	(5/4)n+2	88
# of 2-bit CLA	0	0	n/2-1	63	n/2	72	n/4+1	45
Size of Accumulator	(2n+1)-bit CLA	226	(n+2)-bit CLA	121	0	0	0	0
Size of final adder	(2n)-bit CLA	218	0	0	n-bit CLA	107	n-bit CLA	107
Total	-	1596		1488	-	1475	-	920

표 2. 각 하드웨어 구성요소에 대한 게이트 수
Table 2. Gate count of each hardware element.

Logic	Gate count
NAND	1
HA	4
FA	8
2-bit CLA	9
16-bit CLA	107
18-bit CLA	121
32-bit CLA	218
33-bit CLA	226

구조가 가장 작은 게이트수를 사용한다. 또한 DWT 필터링의 예와 같이 알고리즘 자체의 최적화를 이용한다면 더욱 효율적인 게이트 수를 보일 수 있다.

2. 지연 모델

본 논문에서는 지연 모델을 적용함에 있어서 구현환경을 고려하고 비교의 용이함을 위하여 [19]와 마찬가지로 "Sakurai alpha power law"를 사용한다[20]. 5.1장에서와 마찬가지로 CMOS 공정을 이용하고 논리 연산에 관계된 게이트 이외의 배선 등으로 인한 지연은 무시하기 때문에 $\alpha = 1$의 값을 사용한다. [19]에서 "Sakurai alpha power law"에 기초하고 일부 간략화를 통해 지연에 대한 모델링을 수행하였는데 사용되는 하드웨어 구성요소들에 대한 정규화된 입력 캐패시턴스(Cin)와 게이트 지연(Td)을 표 3에 나타냈다.

표 3에서 사용된 파라미터들에 대한 설명은 아래와 같다.

- η=2 :ratio of the saturation velocity
- c=C/Cg
- t=0.1×T/τ
- C :gate load capacitance
- Cg :gate capacitance of the minimum-area transistor

표 3. 정규화된 입력 캐패시턴스와 게이트 지연
Table 3. Normalized capacitance and gate delay.

Gate	Comment	C_{in}	T_d
Inverter		3	t+c
8×1 MUX	4-level logic	4	35.2+t+c
D-F/F	Slave delay	4	16.1+t+c
1-bit FA	input-to-sum	12	39.6+t+c
1-bit FA	input-to-carry	12	38.7+t+c
2-bit CLA	input-to-sum	12	64.9+t+c
2-bit CLA	input-to-carry	16	53.9+t+c
4-bit CLA	input-to-sum	12	96.8+t+c
4-bit CLA	input-to-carry	24	88+t+c

- T :duration
- τ :minimum-area inverter fall time by Cg

표 1에 나타난 하드웨어 구성요소와 표 3의 값들에 따라서 각 구조들에 대한 지연시간을 구한 결과가 표 4에 나타나있다. 전체적인 결과(표 4에서 Total)를 살펴보면 표준설계의 지연시간이 다른 것들에 비해서 상당히 크다는 것을 볼 수 있다. 제안된 구조의 Booth 인코딩은 [19]의 방식과 동일한 것으로 1의 보수 체계를 사용하여 2의 보수 체계의 사용으로 인한 부가적인 덧셈기의 발생을 막는 구조를 가지고 있다. 또한 앞서 언급한 것과 같이 가능한 모든 조합에 대해 병렬적으로 인코딩을 수행한 뒤 선택하는 방식을 사용하고 있다. 따라서 두 구조는 동일하게 (10.6)n+81.1의 지연을 가진다. 부분곱을 연산하는 CSA 트리의 경우에 지연시간에 대한 임계경로가 2-bit CLA에 달려있는데 표 1에 나타난 것과 같이 제안된 구조가 2-bit CLA를 하나 더 가지고 있기 때문에 (n/2)(67.1)의 지연을 가져서 [19]에 비해서 느린 결과를 보인다. 최종 덧셈기의 경우에 [19]는 부분곱에 대한 합과 캐리를 더하는 역할을 하는 반면에 제안된 구조는 최종 누적결과를 위한 합과 캐리를 더하는 역을 하기 때문에 서로 차이는 있지만 둘 사이에 2비트의 연차이가 있기 때문에 제안된 구조의 지연시간이 57.2만큼 짧은 것을 볼 수 있다.

표 4에서 살펴보면 총 지연시간이 [19]의 경우에 1235.2이고 제안된 구조의 경우에 1245.1로 [19]가 우수한 것과 같이 보일지 모르나 실제적으로 파이프라인을 적용한다면 본 논문에서 제안된 구조는 부분곱의 결과인 합과 캐리를 더하지 않고 2-bit CLA 연산 후에 곧바로 누적연산을 수행할 수 있는 반면에 [19]에서 제안

표 4. 임계경로에 따른 지연시간 비교
Table 4. Delay time comparison for the critical paths.

	Standard design		Design in [19]		Our design		Our design for DWT filtering	
	General	16-bit	General	16-bit	General	16-bit	General	16-bit
Step 1	Booth Encoding		Booth Encoding		Booth Encoding		Booth Encoding	
	(52.8)*n* +59.9	904.7	(10.6)*n* +81.1	250.7	(10.6)*n* +81.1	250.7	(10.6)*n* +81.1	250.7
Step 2	CSA		Hybrid CSA		Hybrid CSA		Hybrid CSA	
	(*n*/2-1) ×(51.9)	363.3	(*n*/2-1) ×(67.1)	469.7	(*n*/2)× (67.1)	536.8	(*n*/2-3) ×(67.1)	335.5
Step 3	Final Addition		Final Addition		Final Addition		Final Addition	
	57.2*n*	915.2	28.6*n*+ 57.2	514.8	28.6*n*	457.6	28.6*n*	457.6
Step 4	Accumulation		-		-		-	
	57.2*n*	915.2	-	-	-	-	-	-
Total	-	3098.4	-	1235.2	-	1245.1	-	792.1
Pipeline Critical Path Delay	Step2+ Step3	1278.5	Step2+ Step3	984.5	Step2	536.8	Step3	457.6

된 구조는 2-bit CLA 연산 후에 최종 캐리가 발생하면 이를 이용하여 최종 덧셈을 수행해야만 누적 연산을 수행할 수가 있다. 따라서 그림 3 혹은 표 4의 Step 단위로 파이프라인을 수행할 경우에 본 논문에서 제안된 MAC의 구조가 가장 우수한 것을 볼 수 있다. 또한 알고리즘 자체의 특성을 고려하여 MAC의 구조를 최적화시킨다면 DWT filtering에서 볼 수 있듯이 지연시간을 많이 줄일 수 있다. 하지만 표 2에서 보이는 것과 같이 자원적인 측면에서는 상당한 효율성을 보이지만 최종 덧셈기의 비트 너비는 줄지 않기 때문에 지연시간은 획기적으로 단축시킬 수 없다.

VI. 결 론

본 논문에서는 디지털 신호처리에서 핵심 동작인 필터링, 컨벌루션, 혹은 내적 등에 고속의 곱셈-누적 연산을 제공할 수 있는 새로운 MAC의 구조를 제안하였다. 기존의 연구에서 병렬 곱셈기 방식을 따르고 있는데 병렬 곱셈기 방식 중에서도 부분곱을 더하는 CSA의 하이브리드 구조를 개선하였다. MAC을 위한 연산 방식을 새롭게 재해석하여 하드웨어 구현을 고려할 경우에 최적화된 파이프라인을 이룰 수 있도록 하였다. 또한 제안된 MAC의 구조를 JPEG2000 주파수변환, MPEG-4의 택스쳐 영상의 압축, 그리고 워터마킹 등의 분야에서 핵심기술로 사용되고 있는 DWT 필터링에 적용하고 최적하는 과정을 수행함으로써 실제적인 적용으로의 가능성을 보였다. 뿐만 아니라 기존의 연구들과 하드웨어 자원 및 지연시간 비교 분석을 통해서 제안된 구조의

우수함으로 보임으로써 추후 다양한 분야에서 사용될 수 있을 것으로 사료된다.

참 고 문 헌

[1] J. J. F. Cavanagh, Digital Computer Arithmetic. New York: McGraw-Hill, 1984.

[2] ISO/IEC 13818-1, 2, 3, Information Technology -Coding of Moving Picture and Associated Autio, MPEG-2 Draft International Standard, 1994

[3] Martin Boliek, et al., JPEG 2000 Part I Final119l Draft International Standard, ISO/IEC JTC1/SC29 WG1, 24 Aug. 2000.

[4] O. L. MacSorley, "High Speed Arithmetic in Binary Computers", Proc. IRE, vol. 49, Jan. 1961.

[5] S.Waser and M. J. Flynn, Introduction to Arithmetic for Digital Systems Designers. New York: Holt, Rinehart and Winston, 1982.

[6] A. R. Omondi, Computer Arithmetic Systems. Englewood Cliffs, NJ:Prentice-Hall, 1994.

[7] Israel Koren, "Computer Arithmetic Algorithms", John wiley Inc., pp. 71-123, 1993.

[8] yoshita Harata, et al., "A High-Speed Multiplier Using a Redundant Binary Adder Tree," IEEE J. of Solide-State Circuits, Vol. sc-22, no. 1, pp.28-33, Feb 1987

[9] A. D. Booth, "A Signed Binary Multiplication Technique", Quart. J. Math., vol IV, pt. 2, 1952.

[10] C. S. Wallace, "A Suggestion for a Fast Multiplier", IEEE Trans. Electron Comp., vol. EC-13, pp. 14-17, Feb. 1964

[11] A. R. Cooper, "Parallel architecture modified Booth multiplier," IEE Proc.-G, vol. 135, pp. 125-128, 1988.

[12] N. R. Shanbag and P. Juneja, "Parallel implementation of a 4x4-bit multiplier using modified Booth's algorithm," IEEE J. Solid-State Circuits, vol. 23, pp. 1010-1013, 1988.

[13] G. Goto, T. Sato, M. Nakajima, and T. Sukemura, "A 54x54 regular structured tree multiplier," IEEE J. Solid-State Circuits, vol. 27, pp. 1229-1236, Sept. 1992.

[14] J. Fadavi-Ardekani, "M NBooth encoded multiplier generator using optimized Wallace trees," IEEE Trans. VLSI Syst., vol. 1, pp. 120-125, 1993.

[15] N. Ohkubo, M. Suzuki, T. Shinbo, T. Yamanaka, A. Shimizu, K. Sasaki, and Y. Nakagome, "A 4.4 ns CMOS 5454 multiplier using pass-transistor multiplexer," IEEE J. Solid-State Circuits, vol. 30, pp. 251-257, Mar. 1995.

[16] A. Tawfik, F. Elguibaly, and P. Agathoklis, "New realization and implementation of fixed -point IIR digital filters," J. Circuits, Syst., Comput., vol. 7, no. 3, pp. 191-209, 1997.

[17] A. Tawfik, F. Elguibaly, M. N. Fahmi, E. Abdel-Raheem, and P. Agathoklis, "High-speed area-efficient inner-product processor," Can. J. Elec. Comput. Eng., vol. 19, pp. 187-191, 1994.

[18] F. Elguibaly and A. Rayhan, "Overflow handling in inner-product processors," in Proc. IEEE Pacific Rim Conf. Communication, Computers, and Signal Processing, Victoria, B.C., Canada, Aug. 20-22, 1997, pp. 117-120.

[19] F. Elguibaly, "A Fast Parallel Multiplier-Accu mulator Using The Modified Booth Algorithm", IEEE, Trans. on circuits and Systems, vol. 27, pp. 902-908, Sep. 2000.

[20] T. Sakurai and A. R. Newton, "Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas," IEEE J. Solid-State Circuits, vol. 25, pp. 584-594, Feb. 1990.

저 자 소 개

서 영 호(종신회원)
1999년 2월 광운대학교 전자재료 공학과졸업(공학사)
2001년 2월 광운대학교 대학원졸업(공학석사)
2000년 3월~2001년 12월 인티스닷컴(주) 연구원
2001년 3월~현재 광운대학교 전자재료공학과 박사과정
2003년 6월~현재 한국전기연구원 연구원
<주관심분야: Image Processing/Compression, 워터마킹, 암호학, FPGA/ASIC 설계>

김 동 욱(종신회원)
1983년 2월 한양대학교 전자공학과 졸업(공학사)
1985년 2월 한양대학교 대학원 졸업(공학석사)
1991년 9월 Georgia 공과대학 전기공학과 졸업(공학박사)
1992년 3월~현재 광운대학교 전자재료공학과 정교수,
광운대학교 신기술연구소 연구원
1997년 12월~현재 광운대학교 IDEC 운영위원
2000년 3월~현재 인티스닷컴(주) 연구원
<주관심분야: 디지털 VLSI Testability, VLSI CAD, DSP 설계, Wireless Communication>