논문 2008-45SD-4-14

Radix-2 MBA 기반 병렬 MAC의 VLSI 구조

(New VLSI Architecture of Parallel Multiplier-Accumulator Based on Radix-2 Modified Booth Algorithm)

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 MAC의 구조를 제안한다. 곱셈과 누적 덧셈 연산을 통합하고 하이브리드 형태의 CSA 구조를 고안하여 임계경로를 감소시키고 출력율을 개선하였다. 즉, 가장 큰 지연시간을 갖는 누적기 자체를 제거하고 누적기의 기능을 CSA에 포함시킴으로써 전체적인 성능을 향상시킨다. 제안된 CSA 트리는 1의 보수 기반의 MBA 알고리즘를 이용하고, 연산자의 밀도를 높이고자 부호비트를 위한 수정된 배열형태를 갖는다. 또한 최종 덧셈기의 비트수를 줄이기 위해서 CSA 트리 내에 2비트 CLA를 사용하여 하위 비트의 캐리를 전파하고 하위 비트들에 대한 출력을 미리 생성한다. 또한 파이프라인의 효율을 최적화시켜 출력율을 증가시키고자 최종 덧셈기의 출력이 아닌 합과 캐리 형태의 중간 연산결과들을 누적시킨다. 제안한 하드웨어를 설계한 후에 250μm, 180μm, 130μm, 그리고 90nm CMOS 라이브러리를 이용하여 합성하였다. 이론 및 실험적인 결과를 토대로 제안한 MAC의 하드웨어 자원, 지연시간, 그리고 파이프라인 등의 결과에 대해 분석하였다. 지연시간은 수정된 Sakurai의 alpha power low를 이용하였다. 결과를 살펴보면 제안한 MAC은 표준 설계에 대해서는 여러 측면에서 매우 우수한 특성을 보였고, 최근 연구와 비교할 때 클록속도는 거의 유사하면서 성능은 두 배로 우수하였다.

Abstract

In this paper, we propose a new architecture of multiplier-and-accumulator (MAC) for high speed multiplication and accumulation arithmetic. By combining multiplication with accumulation and devising a hybrid type of carry save adder (CSA), the performance was improved. Since the accumulator which has the largest delay in MAC was removed and its function was included into CSA, the overall performance becomes to be elevated. The proposed CSA tree uses 1's complement-based radix-2 modified booth algorithm (MBA) and has the modified array for the sign extension in order to increase the bit density of operands. The CSA propagates the carries by the least significant bits of the partial products and generates the least significant bits in advance for decreasing the number of the input bits of the final adder. Also, the proposed MAC accumulates the intermediate results in the type of sum and carry bits not the output of the final adder for improving the performance by optimizing the efficiency of pipeline scheme. The proposed architecture was synthesized with 250μm, 180μm, 130μm, and 90nm standard CMOS library after designing it. We analyzed the results such as hardware resource, delay, and pipeline which are based on the theoretical and experimental estimation. We used Sakurai's alpha power low for the delay modeling. The proposed MAC has the superior properties to the standard design in many ways and its performance is twice as much than the previous research in the similar clock frequency.

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

Ⅰ. 서 론

최근 멀티미디어 시스템 및 통신 분야가 급속도로 발전함에 따라서 디지털 신호처리, 영상처리, 그리고 3차원 신호처리 등과 같은 실시간 신호처리 및 대용량의 데이터 처리를 위한 디지털 시스템이 더욱 요구되고 있다. 특히 디지털 시스템의 연산기 중에서 곱셈기와 곱셈-누적기(multiplier-accumulator, MAC)는 디지털 신

* 평생회원, 광운대학교 교양학부
(Div. of General Edu., Kwangwoon University)
※ 본 논문은 교육인적자원부, 산업자원부, 노동부의 출연금 및 보조금으로 수행한 최우수실험실지원사업의 연구결과입니다.
접수일자: 2007년12월10일, 수정완료일: 2008년3월25일

호처리를 위한 필수적인 요소이다. 곱셈기를 이용한 곱셈-누적 연산[1]은 디지털 신호처리 분야에서 필터링(filtering), 컨벌루션(convolution), 그리고 내적(inner product) 등을 비롯한 모든 연산의 기본이다. 멀티미디어에 대한 관심이 높아지고 수요가 늘어나면서 MPEG[2] 혹은 JPEG2000[3]등과 같은 멀티미디어 신호처리 기술의 사용이 확대 되고 있다. 이러한 표준 기술들은 이산 코사인 변환 (discrete cosine transform, DCT) 혹은 이산 웨이블릿 변환(discrete wavelet transform, DWT)과 같은 비선형 연산을 사용한다. 비선형 함수의 계산은 기본적으로 곱셈과 덧셈을 반복적으로 수행하여 이루어지므로 덧셈과 곱셈의 연산시간이 전체 연산의 수행속도와 성능을 좌우한다. 디지털 시스템을 구성하는 기본적인 연산 블록들 중에서 곱셈기(multiplier)는 지연시간이 가장 길기 때문에 일반적으로 곱셈기에 의해 전체 시스템의 임계경로(Critical path)가 결정된다. 고속의 곱셈연산을 위해서 "Modified Radix-4 Booth Algorithm(MBA)[4]"을 쓰는 것이 일반적이다. 그러나 곱셈 연산에 소요되는 긴 임계경로에 대한 문제를 완전히 해결할 수는 없다[5~6].

고속 곱셈을 위한 곱셈기 구조는 배열 곱셈기(array multiplier)와 병렬 곱셈기(parallel multiplier)로 구분된다[7]. 배열 곱셈기는 단위 셀의 구조가 규칙적이고 VLSI 구현이 쉽다는 장점을 가지고 있으나 계산 속도가 곱해지는 비트수에 따라 비례적으로 증가한다는 단점을 갖는다. 병렬 곱셈기는 연산자의 비트수에 따른 곱셈 시간의 증가가 배열 곱셈기보다 작다는 장점을 갖는다[8]. 따라서 어떤 분야에서 사용되느냐에 따라 다양한 형태의 하드웨어 구조가 제안되어 왔고, 응용 목적에 따라 적절하게 선택하여 사용하여야 한다. 예를 들어 파이프라인 방식의 병렬 곱셈기의 경우 디지털 신호처리 기반의 알고리듬을 구현하는데 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 있다. 이러한 곱셈기는 크게 Booth 인코더, 부분곱 압축 트리, 최종 덧셈기의 세부분으로 구성된다[11~12]. Wallace 트리는 인코더로부터 나오는 부분곱들을 최대한 병렬형태로 더하기 위한 구조이므로 (N-bit) 데이터의 곱셈의 수행시간은 $O(log_2N)$에 비례한다. 즉, CSA를 사용하여 N개의 입력 중에서 1의 개수를 카운트하여 출력으로 내보내면 출력 비트의 개수는 log_2N개로 줄어드는 원리를 이용하는 것이다. 실제로는 다수의 비트수를 입력으로 사용하는 CSA를 구현하는데 많은 면적이 소요되므로 (3:2) 혹은 (7:3) 카운터를 다수 사용하여 파이프라인 단계마다 카운터의 개수를 감소시킨다. 곱셈 과정은 일련의 부분곱 덧셈 과정을 반복적으로 수행하는 것이기 때문에 고속의 곱셈을 위해서는 부분 곱의 수를 줄여서 계산 단계를 감소시켜야 한다. 부분곱의 계산 단계를 감소시키기 위해서 주로 MBA 알고리즘이 적용되고 있고, Wallace 트리를 적용하여 부분곱의 빠른 덧셈을 수행한다. MBA의 속도를 높이기 위해 병렬 곱셈구조가 많이 연구되어 왔다[13~15]. 일반적으로 "Baugh-Wooley Algorithm(BWA)"에 기초한 MAC의 구조가 개발되어 왔고, 제안된 구조들은 다양한 디지털 필터링 연산에 도입되었다[16~18].

범용적인 디지털 신호처리를 위해 가장 발달된 형태의 MAC 중에 하나가 Elguibaly[19]에서 제안되었는데 부분곱을 압축하는 CSA 트리에 누적 연산을 결합한 구조이다. Elguibaly가 제안한 구조는 누적연산을 위한 별도의 덧셈기를 없애고 최종 덧셈기의 비트수를 감소시킴으로써 임계경로를 감소시켰다. 임계경로의 감소로 인하여 이전에 제안된 MAC에 비해서는 높은 성능을 갖지만 누적연산에 최종덧셈기의 결과를 사용하기 때문에 출력율의 향상을 위한 여지를 남겼다.

본 논문에서는 고속의 곱셈-누적 연산을 수행할 수 있는 새로운 MAC의 구조를 제안한다. 제안하는 MAC은 곱셈과 누적 덧셈 연산을 통합하고, 하이브리드 형태의 CSA 구조를 제안하여 임계경로를 감소시키고 출력율을 개선하고자 한다. 제안된 CSA 트리의 기능을 살펴보면 부분곱들의 덧셈과 함께 누적연산을 포함한다. 세부적으로 살펴보면 1) 1의 보수 기반의 MBA 알고리즘를 이용하고 2) 연산자의 밀도를 높이고자 부호비트를 위한 수정된 배열형태를 가지며 3) 최종 덧셈기의 비트수를 줄이기 위해 CSA 트리 내에 CLA를 사용한다. 또한 파이프라인의 효율을 최적화시켜 출력율을 증가시키고자 최종 덧셈기의 출력이 아닌 합과 캐리 형태의 중간 연산결과들을 누적시킨다.

본 논문은 다음과 같이 구성된다. Ⅱ장에서는 일반적인 MAC에 대해서 간략히 소개하고, Ⅲ장에서 제안된 MAC의 구조를 설명한다. Ⅳ장에서는 구현 결과를 분석하여 제안된 MAC의 특성을 보인다. 마지막으로 Ⅴ장에서 결론을 맺는다.

Ⅱ. MAC의 개요

본 장에서는 일반적인 곱셈과 누적 연산에 대해서 소개한다. 곱셈기는 크게 세 가지 과정으로 나눌 수 있다. 첫 번째는 피승수(X)와 승수(Y)로부터 부분곱들을 만들어내는 Booth 인코딩이고, 두 번째는 생성된 부분곱들을 모두 더하여 합과 캐리형태의 값으로 만드는 덧셈기 배열 혹은 부분곱 압축 과정이다. 그리고 마지막은 합과 캐리를 더하고 최종적인 곱셈결과를 만드는 최종 덧셈이다. 여기에 곱한 결과를 누적하는 과정을 포함시킨다면 MAC은 총 네 가지 단계로 나누어진다. 이러한 연산 단계를 그림 1에 나타내었다. 그림 1은 앞서 설명한 곱셈-누적의 연산 단계를 나타내는 것으로 순서대로 Step1, Step2, Step3, 그리고 Step4에 해당한다.

그림 2는 일반적인 MAC의 H/W 구조이다. 입력되는

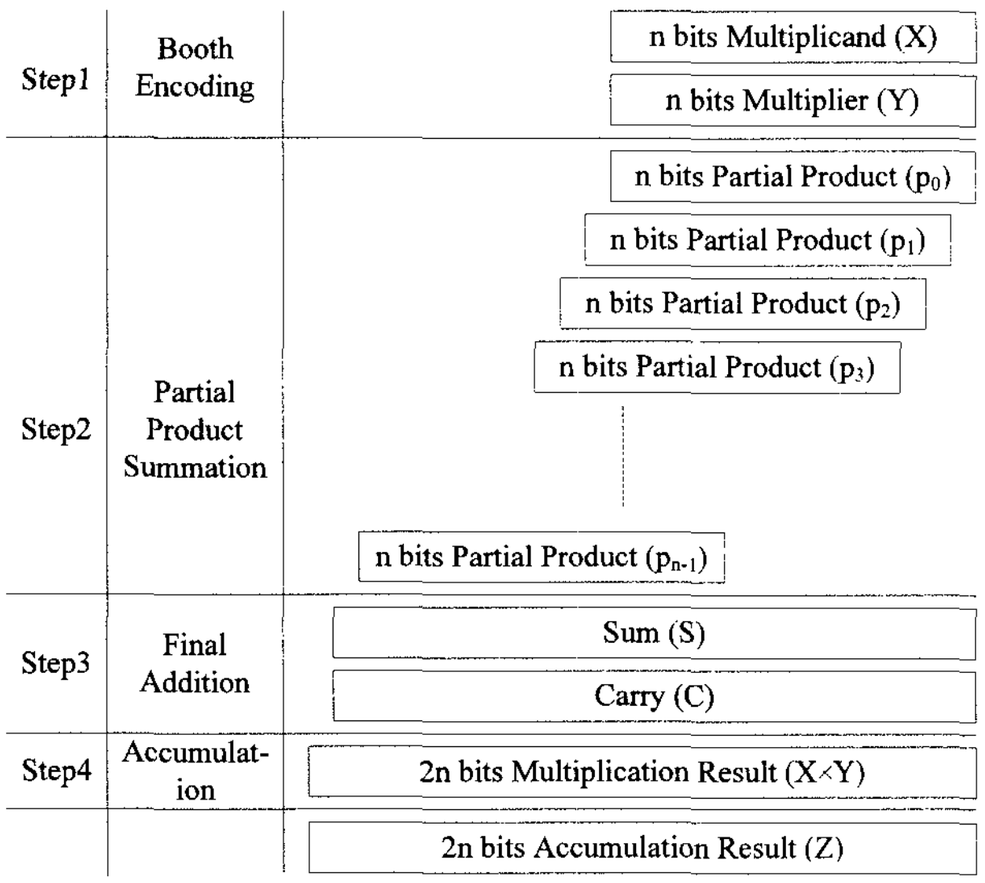

그림 1. 일반적인 곱셈과 누적 연산과정
Fig. 1. General arithmetic operation of multiplication and accumulation.

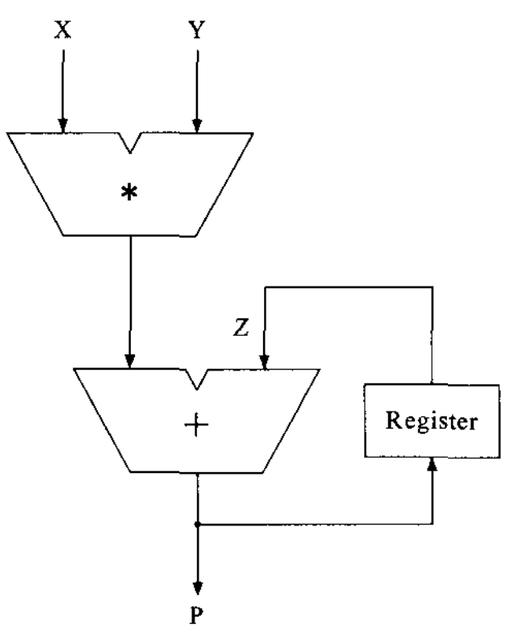

그림 2. 일반적인 MAC의 하드웨어 구조
Fig. 2. Hardware architecture of general MAC.

승수, X와 피승수, Y를 곱하여 곱셈 연산을 수행하고 이를 이전 곱셈결과인 Z와 더하여 누적 연산을 수행한다. 여기서 N 비트의 크기를 갖고 2의 보수 체계를 사용하는 2진수, X를 식 (1)로 표현할 수 있다.

$$X = -2^{N-1}x_{N-1} + \sum_{i=0}^{N-2} x_i 2^i, \qquad x_i \in \{0,1\} \tag{1}$$

식 (1)을 Booth 알고리즘을 적용하기 위하여 Base-4 형태의 redundant sign digit 방식으로 나타내면 식 (2) 및 (3)과 같이 나타낼 수 있다[7].

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

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

식 (2)를 이용하면 곱셈은 식 (4)와 같이 표현된다.

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

이러한 식들을 이용하면 앞서 설명한 곱셈-누적 연산의 결과는 식 (5)로 표현된다.

$$P = X \times Y + Z = \sum_{i=0}^{N/2-1} d_i 2^{2i} Y + \sum_{j=0}^{2N-1} z_i 2^i \tag{5}$$

식 (5)의 오른쪽에 있는 두 항들은 각각 독립적으로 연산된 후, 두 항의 결과에 대해 덧셈연산, 즉 누적 연산을 수행하여 최종적인 결과를 생성한다. 식 (5)로 구현된 MAC의 구조를 표준 설계(standard design)라 한다.

기본적으로 N-bit 길이의 데이터들을 곱하는 경우에는 N에 비례하는 수만큼의 부분곱들이 생성된다. 이들을 순차적으로 더하기 위하여 N에 비례하는 덧셈 수행시간이 소용된다. 현재까지 고안된 곱셈기 중 가장 빠른 것으로 알려진 곱셈기의 구조는 부분곱들을 생성하는 Booth 인코딩을 사용하고 부분곱들을 더하는 덧셈기 배열에는 CSA 기반의 Wallace 트리를 이용한다. Booth 인코딩 방법을 사용하면 Wallace 트리의 입력에 해당하는 부분곱의 수가 반으로 줄어들어 CSA 트리의 단계가 감소한다. 또한 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. 제안한 MAC의 구조

본 장에서는 표준 설계의 수식으로부터 새로운 연산 방식을 위한 수식을 유도하고, 이로부터 새로운 MAC의 VLSI 구조를 제안한다. 또한 제안한 MAC의 동작을 만족시킬 수 있는 하이브리드 형태의 CSA 구조를 제안한다.

1. MAC 연산의 유도

가. 개념

두 개의 N비트를 곱한 후에 2N 비트를 누적하는 연산을 고려한다면 임계경로는 2N 비트의 누적 연산에 의해서 결정된다. 그림 1의 표준 설계에서 각 Step 별로 파이프라인을 적용했다면 MAC의 성능을 향상하기 위해서는 마지막 누적기의 지연시간을 감소시켜야 한다. 본 논문에서 제안하고자 하는 MAC은 가장 큰 지연시간을 갖는 누적기 자체를 제거하고 누적기의 기능을 CSA에 포함시킴으로써 전체적인 성능을 향상시킨다. 누적기가 제거된 상황이라면 임계경로는 곱셈기의 최종 덧셈기에 의해서 결정된다. 최종 덧셈기의 성능을 향상시키는 근본적인 방법은 입력의 비트수를 축소시키는 것이다. 최종 덧셈기의 입력 비트수를 줄이기 위해서는 CSA가 다수의 부분곱들을 합과 캐리로 압축할 때 전체적인 성능을 감소시키지 않는 범위 내에서 합과 캐리의 하위 비트들을 미리 더하여 최종 덧셈기로 전달되는 합과 캐리의 비트수를 줄여야 한다. CSA 내에서 하위 비트들을 더하는데 2 비트 CLA를 사용한다. 또한 파이프라인을 적용할 경우에 출력율을 높이기 위해서 누적연산의 입력으로 최종 덧셈기의 출력이 아니라 최종 덧셈기의 입력에 해당하는 CSA의 합과 캐리를 사용한다. 이전 싸이클에서 출력된 CSA의 합과 캐리가 다음 싸이클에서 누적연산을 위한 CSA의 입력으로 사용된다. 합과 캐리가 모두 귀환되므로 표준 설계와 Elguibaly[19]의 구조에 비해서 CSA에 입력되는 데이터량이 증가한다. 데이터량의 증가를 효율적으로 해결하고자 부호 비트를 위해 수정된 CSA 구조를 제안한다.

나. 수식의 유도

앞서서 소개한 개념을 식 (5)에 적용하여 제안하고자 하는 MAC 연산 식을 유도한다. 곱셈연산을 제안한 방식에 적합한 구조로 변형하고, 누적연산을 위해 귀환되는 값을 새롭게 정의하여 새로운 MAC을 위한 연산식으로 전개한다.

먼저, 식 (4)의 곱셈 연산을 풀어서 정리하면 식 (6)과 같다.

$$X\times Y = d_0 2Y + d_1 2^2 Y + d_2 2^4 Y + \cdots + d_{N/2-1} 2^{N-2} Y \tag{6}$$

식 (6)을 첫 번째 부분곱, 중간 부분곱의 합, 그리고 최종 부분곱으로 나누면 식 (7)과 같다. 식 (7)과 같이 부분곱의 덧셈을 구분하는 것은 누적 연산을 위해 귀환되는 데이터가 세 가지이기 때문이다. 세 가지 데이터에는 CSA의 합, 캐리, 그리고 하위 비트들의 합과 캐리가 더해진 결과값이 포함된다.

$$X\times Y = d_0 2Y + \sum_{i=1}^{N/2-2} d_i 2^{2i} Y + d_{N/2-1} 2^{N-2} Y \tag{7}$$

다음으로 식 (5)의 Z에 대해 제안한 개념을 도입하자. Z를 먼저 상위 비트와 하위 비트로 구분하여 정리하면 식 (8)과 같다. 식 (8)의 첫 번째 항은 상위 비트에 해당하는데 합과 캐리 형태로 귀환되는 값이고, 두 번째 항은 하위 비트에 해당하는데 합과 캐리의 덧셈 결과 형태로 귀환되는 값이다.

$$Z = \sum_{i=0}^{N-1} z_i 2^i + \sum_{i=N}^{2N-1} z_i 2^i \tag{8}$$

식 (8)로부터 Z를 합과 캐리의 연산형태로 재분리하면 식 (9)와 같다.

$$Z = \sum_{i=0}^{N-1} z_i 2^i + \sum_{i=0}^{N-2} c_i 2^i 2^N + \sum_{i=0}^{N-2} s_i 2^i 2^N \tag{9}$$

식 (7)과 식 (9)를 이용하면 식 (5)의 MAC 연산은 식 (10)으로 표현할 수 있다.

$$\begin{aligned} P = {} & (d_0 2Y + \sum_{i=1}^{N/2-2} d_i 2^{2i} Y + d_{N/2-1} 2^{N-2} Y) \\ & + (\sum_{i=0}^{N-1} z_i 2^i + \sum_{i=0}^{N-2} c_i 2^i 2^N + \sum_{i=0}^{N-2} s_i 2^i 2^N) \end{aligned} \tag{10}$$

식 (10)을 비트 자리수에 맞추어 재정리하면 식 (11)과 같고, 이 식이 제안하고자 하는 MAC의 연산 형태에 해당한다. 오른쪽의 첫 번째 항은 첫 번째 부분곱이 합과 캐리의 덧셈결과와 누적되는 연산이고, 두 번째 항은 중간 부분곱들이 귀환된 CSA의 합과 누적되는 연산이다. 마지막으로 세 번째 항은 마지막 부분곱이 CSA의 캐리와 누적되는 연산을 나타낸다.

$$P=(d_0 2Y+\sum_{i=0}^{N-1} z_i 2^i) + (\sum_{i=1}^{N/2-2} d_i 2^{2i} Y+\sum_{i=0}^{N-2} c_i 2^i 2^N) + (d_{N/2-1} 2^{N-2} Y+ \sum_{i=0}^{N-2} s_i 2^i 2^N) \quad (11)$$

2. 제안한 MAC의 구조

앞 절에서 제안한 MAC의 연산 과정을 정리하면 그림 3과 같이 나타낼 수 있다. 그림 3을 살펴보면 MAC 연산은 세 가지 단계로 구성된다. 그림 1과 비교하면 누적 연산이 부분곱을 더하는 과정에 포함되었다는 차이점을 확인할 수 있다. 그림 3에서 직접으로 나타나있지는 않지만 또 한 가지 그림 1의 방식과 큰 차이점은 Step3의 최종 덧셈과정이 늘 필요하지 않다는 점이다. 누적 연산이 Step3의 결과가 아니라 Step2의 결과를 이용하여 수행되므로 누적 연산의 최종 결과가 필요한 시점이 아니라면 Step3을 동작시킬 필요가 없다.

그림 3의 과정을 만족하기 위한 MAC의 하드웨어 구조는 그림 4와 같다. MAC의 입력인 n 비트의 X와 Y는 Booth Encoder를 거치면서 (n+1) 비트 부분곱의 형태로 변환된다. CSA & Accumulator에서 부분곱의 덧셈과 함께 누적 연산이 수행되고, 결과로 n 비트의 S와 C 그리고 합과 캐리의 하위 비트 덧셈결과인 P[n-1:0]가 생성된다. 이 세 가지 값은 귀환되어 다음 누적연산에 이용된다. 만일 MAC의 최종 결과가 필요하다면 S와 C를 Final Adder에서 합하여 P[2n-1:n]을 생성하고 이미 생성된 P[n-1:0]와 함께 최종 결과를 구성한다.

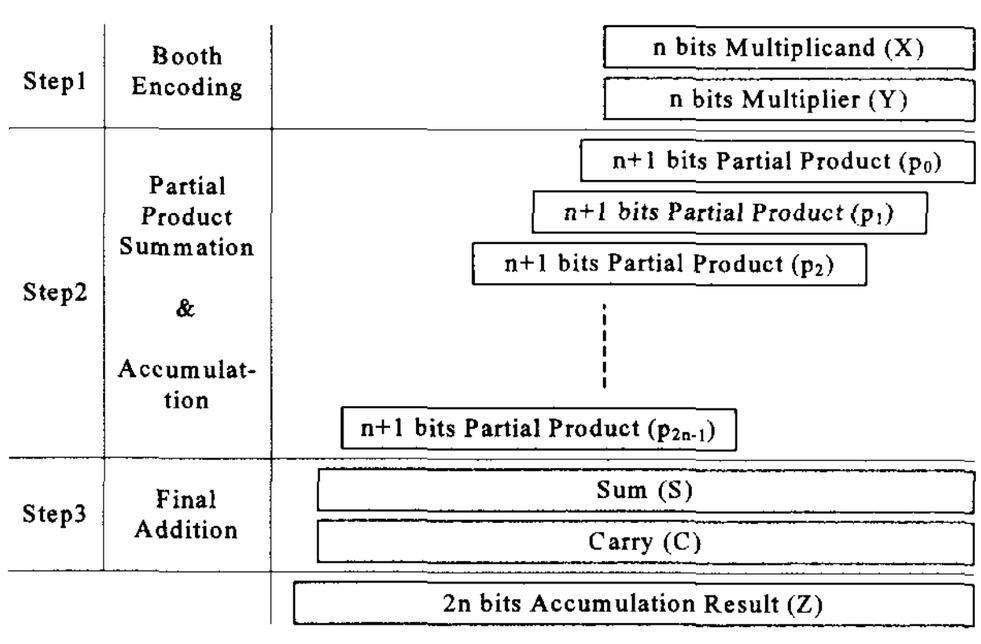

그림 3. 제안한 곱셈과 누적 연산과정
Fig. 3. Proposed arithmetic operation of multiplication and accumulation.

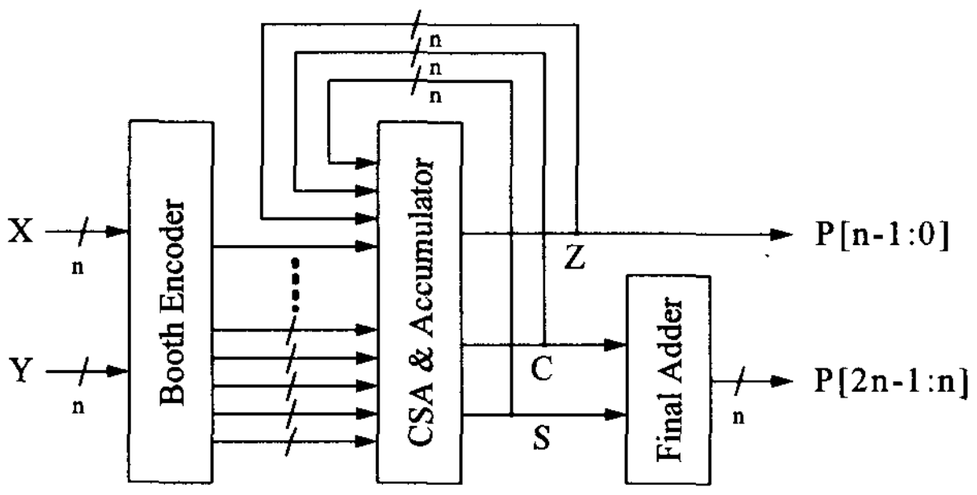

그림 4. 제안한 MAC의 하드웨어 구조
Fig. 4. Hardware architecture of the proposed MAC.

3. 제안한 CSA 구조

제안한 MAC의 동작을 수용할 수 있는 하이브리드 형태의 CSA의 구조를 그림 5에 나타내었다. 제안한 CSA는 식 (11)을 바탕으로 구성하였다. 그림 5에서 S_i는 부호 확장을 간략하게 하기 위한 보상값이고 N_i는 Booth 인코딩 시 사용했던 1의 보수를 2의 보수로 보상하는 값이다. $S[i]$와 $C[i]$는 각각 합과 캐리의 i번째 비트에 해당하고 $S'[i]$와 $C'[i]$는 누적연산을 위해 귀환된 합과 캐리의 i번째 비트이다. $Z[i]$는 각각 부분곱의 하위 비트들이 미리 더해진 합에 해당하고 $Z'[i]$는 이전 결과이다. 또한 $P_j[i]$는 j번째 부분곱의 i번째 비트에 해당한다. 그림 5는 8×8 비트 MAC 연산에 대한 CSA 트리의 구조이다. 승수가 8비트이므로 Booth Encoder로부터 총 4개의 부분곱($P_0[7:0]\sim P_3[7:0]$)이 생성된다. 식 (11)에서 $d_0 Y$와 $d_{N/2-1}2^{N-2}Y$는 각각 $P_0[7:0]$와 $P_3[7:0]$에 해당한다. CSA는 4개의 부분곱을 위해 최소 4개의 전가산기(full adder, FA) 열이 필요하다. 그리고 누적 연산을 위해 한 단계의 열을 더 사용하여 총 5단의 전가산기 열이 사용된다. n×n 비트의 MAC 연산이라면 CSA는 (n/2+1)의 크기이다. 그림에서 정사각형의 심볼은 FA를 나타내고 회색으로 칠해

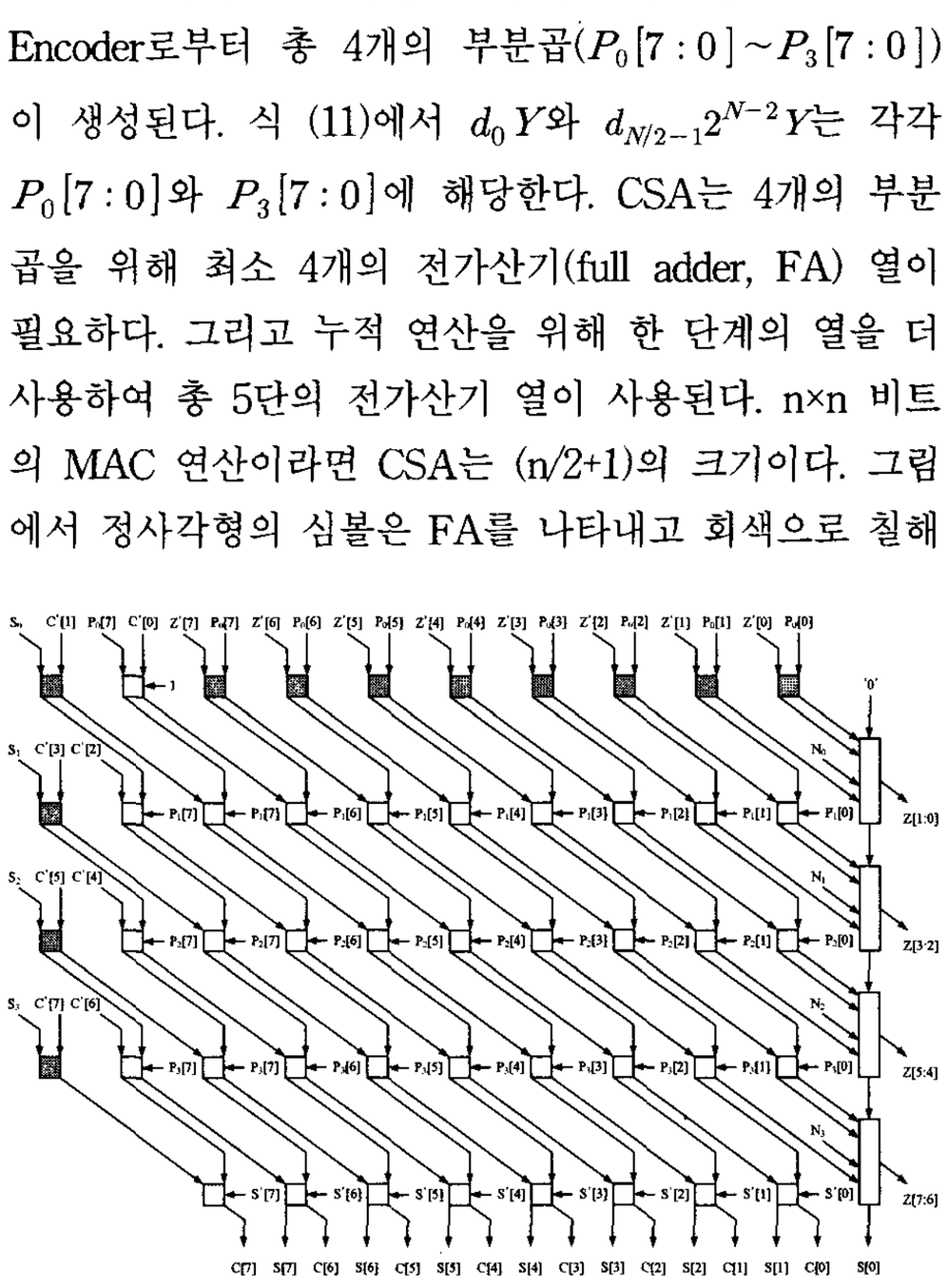

그림 5. 제안한 CSA 트리의 구조
Fig. 5. Architecture of the proposed CSA tree.

표 1. CSA의 특성표
Table 1. Characteristics of CSATable.

	Standard	Elguibaly's CSA	Seo's CSA
Number System	2's Complement	1's Complement	1's Complement
Sign Extension	Used	Used	Not Used
Accumulation	Result Data of Final Addition	Result Data of Final Addition	Sum and Carry of CSA
CSA Tree	FA, HA	FA, 2bits CLA	FA, HA, 2bit CLA
Final Adder	$2n$ bits	$(n+2)$ bits	n bits

진 정사각형 심볼은 HA(반가산기, half adder)이다. 또한 5비트를 입력으로 갖는 직사각형 심볼은 캐리 입력을 갖는 2비트 CLA이다.

CSA에서 임계경로는 2비트 CLA에 의해서 결정된다. 2비트 CLA를 사용하지 않고 FA를 사용하여 CSA를 구성할 수도 있다. 그러나 CLA를 사용하여 먼저 생성되는 부분곱의 하위비트들을 미리 처리하지 않으면 최종 덧셈기의 비트수가 증가하여 곱셈기 혹은 MAC의 성능이 저하된다.

표 1에 앞서 설명된 제안한 CSA 구조의 특성을 요약하고, 다른 구조들과 간략히 비교하였다. 수 체계는 1의 보수를 이용하였고, 부호 확장을 하지 않으면서 수정된 방식을 사용하였다. 가장 큰 차이점은 누적을 위해 귀환시키는 데이터의 종류이다. 그리고 최종 덧셈기의 입력이 n비트로 가장 작다.

IV. 구현 및 결과

본 장에서는 제안된 MAC을 구현하고 분석한 후에 제안한 MAC과 기존의 연구들을 비교하여 장점을 살펴본다. 먼저, 구현한 하드웨어가 사용하는 자원량을 이론 및 실험적으로 분석한다. 다음으로 Sakurai의 alpha power law를 간략화시켜 구현된 하드웨어의 지연시간을 모델링하고 분석한다. 마지막으로 파이프라인 단계를 제시하고 이를 바탕으로 성능을 분석한다. 각 절에서 구현 결과는 기존의 연구들 중에서 가장 대표적인 병렬 MBA 구조를 갖는 표준 설계[6] 및 Elguibaly[19]와 비교하여 차이점과 우수성을 보이고자 시도한다.

1. 하드웨어 자원 (Hardware resource)

가. 하드웨어 자원의 분석

표 2에 앞서 언급된 세 가지 구조를 분석하여 하드웨어 자원을 비교하였다. 하드웨어 자원을 계산하는데 있어서 Booth Encoder의 자원은 제외하였고, 그것은 모든 설계에 대해서 동일한 것을 사용한다고 가정하였다. 표 2의 하드웨어 자원은 요소 성분들을 모두 카운팅한 후에 일반적인 구조와 16비트 구조에 대한 결과를 나타냈다. 16비트에 대한 하드웨어 자원은 TSMC의 90nm CMOS HVT standard cell library를 이용하였다. 회로 요소 성분들을 최적화된 형태로 합성하여 각각의 게이트수를 구하고, 계산된 하드웨어 자원의 수에 곱하여 결과를 생성하였다. 합성을 통해 얻어진 회로 요소들의 게이트 수를 표 3에 나타내었다.

표 2. 하드웨어 자원의 계산
Table 2. Calculation of hardware resource.

	Standard design		Elguibaly		Seo's Design	
	General	16bits	General	16bits	General	16bits
FA	$\frac{n^2}{2}+n$	964.8	$\frac{n^2}{2}+2n+3$	1092.1	$\frac{n^2}{2}+\frac{n}{2}$	911.2
HA	0	0	0	0	$\frac{3n}{2}$	76.8
2bits CLA	0	0	$\frac{n}{2}-1$	49	$\frac{n}{2}$	56
ACC	$(2n+1)bits$ CLA	214	-		-	
Final adder	$(2n)bits$	197	$(n+2)bits$	109.5	$(n)bits$	97
Total		1375.8		1250.6		1141

표 3. 회로 요소의 게이트 수
Table 3. Gate size of logic circuit element.

Element	Gate Size
Inverter	0.8
2/3/4-NAND	1/1.5/2.5
2/3/4-NOR	1/2/2.2
2/3/4-XOR	2/4/6
2/3/4-AND	1.2/1.5/2
2/3/4-OR	1.2/1.5/2
Half Adder	3.2
Full Adder	6.7
D Flip-Flop	6.2
4×1 MUX	6
8×1 MUX	14.2
2bits CLA	7
4bits CLA	20.5

표 3에서 각 게이트수는 2입력 NAND 게이트를 기준으로 한다. 표 3에서 몇 가지 요소의 게이트수를 살펴보자. HA의 게이트수는 3.2이고 FA의 게이트수는 6.7이므로 FA는 HA의 약 두 배의 크기이다. 2비트 CLA의 게이트수는 7개이므로 FA의 게이트수보다 조금 크다. 즉, 제안한 CSA 구조에서 부분곱의 하위비트들을 미리 더하는데 2비트 CLA를 사용한다고 해도 하드웨어 자원이 크게 증가하는 것은 아니라는 것을 예측

할 수 있다.

표 2를 살펴보면 쉽게 예측할 수 있는 것과 같이 표준설계가 가장 많은 하드웨어 자원을 사용하고 제안한 구조가 가장 작은 하드웨어 자원을 사용하는 것을 볼 수 있다. 제안한 구조는 FA와 HA를 혼용하여 CSA의 자원을 최적화시켰고, 최종 덧셈기에 입력되는 비트수를 축소시킴으로써 Elguibaly에 비해 최종 덧셈기의 크기가 12개 감소하였다.

나. 합성에 의한 게이트 수

HDL(hardware description language)을 이용하여 Elguibaly와 제안한 MAC을 RTL(register transfer level)로 구현하였다. Synopsys의 design complier를 이용하여 설계된 회로를 합성하였고, 그 결과로 생성된 netlist의 게이트수를 측정하여 표 3에 요약하였다. 표 4의 회로는 16비트 MAC을 구현한 것이다. CMOS 공정에 따른 회로의 특성을 파악하기 위해 현재 디지털 반도체를 제작하는데 가장 많이 사용되는 네 가지 공정(0.25μm, 0.18μm, 0.13μm, 90nm)의 라이브러리를 이용하였다. 미세 공정을 적용할수록 게이트 수가 감소하는 것을 볼 수 있고, 표 2의 결과와 마찬가지로 제안한 구조의 게이트 수가 조금 더 작다는 것을 확인할 수 있다. 비록 동일한 constraint를 적용하여 합성하였지만, 다른 회로들과 함께 큰 회로의 일부분으로 사용될 경우에는 전체 회로의 타이밍 및 전기적인 상황에 따라서 게이트 수는 변할 수 있음을 염두에 두어야 한다. 표 4의 결과는 순차회로를 제외한 조합회로에 대한 결과이고, 전체 게이트 수는 Booth Encoder, CSA, 그리고 Final Adder의 합이다.

표 4. 합성에 의한 게이트 수 측정표
Table 4. Estimation of gate size by synthesis.

	CSA		Booth Encoder	Final Adder		Total (C/L)	
nm	Elguibaly	Seo		Elguibaly	Seo	Elguibaly	Seo
90	1,067	1,009	713	104	97	1,884	1,819
130	1,216	1,158	864	118	110	2,198	2,131
180	1,581	1,484	808	120	114	2,510	2,407
250	2,027	2,001	1,129	141	131	3,297	3,261

표 5. 게이트 수의 비교
Table 5. Comparison of gate size.

	Calculated		Experimental	
nm	Elguibaly	Seo	Elguibaly	Seo
90	1,250.6	1,141.0	1,171.0	1,106.1

표 5에는 표 2에서 보인 구조 분석에 의해 계산된 게이트수를 합성에 의해 생성된 실제 회로의 게이트수와 비교한 결과이다. 표 2와 동일한 조건에서 비교하기 위해 Booth Encoder를 제외한 CSA와 Final Adder만 포함시켰다. 계산에 의한 결과에서는 제안한 구조가 111개 작은 게이트수를 갖지만 실제 합성된 회로에서는 65개의 차이를 보인다. 합성 시 부여한 constraint의 조건에 따라서 합성 결과는 조금 달라질 수 있다.

2. 지연 모델 (Delay model)

가. 모델링 (Modeling)

본 논문에서는 지연시간을 모델링는데 Sakurai의 alpha power law를 사용한다[20]. CMOS 공정을 이용하고 논리 연산와 관계된 게이트 이외의 배선 등으로 인한 지연은 무시하기 때문에 α=1의 값을 사용한다. Elguibaly[19]는 alpha power law를 일부 간략화하여 지연시간을 모델링하였는데, 본 논문에서도 다른 구조들과 쉽게 비교하기 위해서 Elguibaly에 의해 추출된 모델링값을 동일하게 사용한다. 사용된 하드웨어 구성요소들에 대한 정규화된 입력 캐패시턴스(C_i)와 게이트 지연(T_d)을 표 6에 나타냈다. η는 포화 속도율이고, C 및 C_g는 각각 게이트 부하 캐패시턴스와 최소 트랜지스터의 게이트 캐패시턴스이다. T는 유지시간이고 γ는 C_g에 의한 최소 인버터의 하강시간이다.

지연 모델링과 그의 간략화 과정은 본 논문의 초점이 아니므로 자세하게 설명하지 않는다. 추가적인 설명은 참고문헌을 참조하기 바란다[19~20].

표 6. 정규화된 입력 캐패시턴스와 게이트 지연
Table 6. Normalized capacitance and gate delay. ($\eta=2,\ c=C/C_g,\ t=0.1\times T/\gamma$)

Gate	Comment	C_i	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
1bit FA	input-to-sum	12	39.6+t+c
1bit FA	input-to-carry	12	38.7+t+c
2bits CLA	input-to-sum	12	64.9+t+c
2bits CLA	input-to-carry	16	53.9+t+c
4bits CLA	input-to-sum	12	96.8+t+c
4bits CLA	input-to-carry	24	88+t+c

나. 지연시간의 분석

표 6과 참고문헌 [19~20]을 이용하여 booth encoder

(T_b), CSA (T_c), 그리고 최종 덧셈기 (T_f)의 지연시간을 모델링한 결과를 식 (12)에서 식 (15)에 나타내었다. 식 (13)에서 T_s, T_p, 그리고 T_m는 각각 선택 회로의 지연시간, 버퍼의 지연시간, 그리고 MUX의 지연시간을 나타낸다.

$$T_b = T_s + (n+2)\,T_p + T_m \tag{12}$$

$$T_b = 12.3 + (n+2) \times 10.6 + 47.6 = 10.6n + 81.1 \tag{13}$$

$$T_c = (\frac{n}{2})\,T_2(carry) = (\frac{n}{2})67.1 = 33.5n \tag{14}$$

$$T_f = (\frac{n}{4})\,T_4(carry) = 28.6n \tag{15}$$

표 2의 하드웨어 자원과 표 6의 게이트 지연시간을 이용하여 표 7에 지연시간을 구하였다. 결과를 살펴보면 표준설계의 지연시간이 다른 것들에 비해서 상당히 크다는 것을 확인할 수 있다. 제안된 구조는 Elguibaly와 동일한 Booth Encoder를 사용하기 때문에 지연시간도 $10.6n+81.1$로 동일하다. 부분곱을 더하는 CSA 트리는 2비트 CLA에 의해서 임계경로가 결정되므로 지연시간은 2비트 CLA에 비례한다. 표 2에 나타낸 것과 같이 제안된 구조는 Elguibaly에 비해 2비트 CLA를 하나 더 가지므로 지연시간이 67.1만큼 더 늦다. 최종 덧셈기의 입력 비트수는 제안한 구조가 하나 더 작기 때문에 지연시간도 57.2만큼 빠르다.

각 Step 별로 파이프라이닝을 적용할 경우에 제안한 구조의 임계경로는 $33.55n$이고 16비트의 MAC이라면 536.8의 값에 해당한다. 단순히 지연시간만을 고려하여 클록의 속도만 따진다면 제안한 구조가 Elguibaly의 구조에 비해서 특성이 좋지 않는 것처럼 보일 수 있다. 그러나 실제 출력율을 고려한 성능을 따진다면 제안한 구조가 더욱 우수하다는 것을 확인할 수 있다. 그 이유는 다음 절에서 파이프라인의 구조와 함께 자세히 설명하도록 한다. 일단 간략히 설명하면 제안한 구조는 성능을 높이기 위한 알고리즘이 적용되었기 때문에 Elguibaly의 구조에 비해서 CSA의 하드웨어 자원이 조금 더 추가되었다. 따라서 앞서 설명한 것과 같은 특성을 보이는 것이다.

표 7. 지연시간의 분석과 비교표
Table 7. Delay time analysis and comparison.

	Standard Design		Elguibaly's Design		Seo's Design	
	General	16bits	General	16bits	General	16bits
Step1	Booth Encoding		Booth Encoding		Booth Encoding	
	$52.8n+59.9$	904.7	$10.6n+81.1$	250.7	$10.6n+81.1$	250.7
Step2	CSA		Hybrid CSA		Hybrid CSA	
	$25.95n-51.9$	363.3	$33.55n-67.1$	469.7	$33.55n$	536.8
Step3	Final Addition		Final Addition		Final Addition	
	$57.2n$	915.2	$28.6n+57.2$	514.8	$28.6n$	457.6
Step4	Accumulation		-		-	
	$57.2n$	915.2	-	-	-	-
Critical Path	Accumulation		Final Addition		Hybrid CSA	
	$57.2n$	915.2	$28.6n+57.2$	514.8	$33.55n$	536.8

3. 파이프라인

가. 단계 분석

앞서서 얻어진 지연 모델링을 바탕으로 하여 파이프라인 단계를 결정한다. Booth encoding과 CSA 동작에 해당하는 Step1과 Step2를 Stage1으로 정하고, 최종 덧셈기에 해당하는 Step4를 Stage2로 정한다. 이러한 경우에 파이프라인 단계는 표 8과 같이 정리할 수 있고, 이 결과에 의해서 clock frequency가 결정된다.

표 8을 살펴보면 Elguibaly의 설계가 제안한 구조보다 더 빠른 클록 주파수에서 동작할 수 있다는 것을 볼 수 있다. 그러나 클록 주파수가 미세하게 높다고 해서 MAC의 전체적인 성능이 우수한 것은 아니다. 제안한 구조가 Elguibaly의 구조보다 지연시간이 조금 큰 것은 성능을 높이는데 초점을 맞추었기 때문이다. 다음 절에서 이러한 내용을 자세하게 살펴보도록 하자.

표 8. 파이프라인 단계
Table 8. Pipeline stage.

	Standard design	Elguibaly's Design	Seo's Design
Stage1	139.95n+87.1	44.15n+14	44.15n+81.1
Stage2	57.2n+28.5	28.6n+57.2	28.6n

나. 파이프라인 구조 및 동작

제안한 구조는 최종 덧셈기의 결과가 아니라 CSA의 최종 출력을 귀환시켜 누적연산을 수행하기 때문에 그림 6(a)와 같은 파이프라인 경로를 갖는다. 그리고 Elguibaly의 구조는 최종 덧셈기의 결과를 귀환시켜 누적연산을 수행하기 때문에 그림 6(b)와 같다. 그림 6의 구조들을 클록에 따라서 순차적으로 동작시킬 경우에 그림 7과 같다.

Elguibaly의 구조는 누적연산을 위한 구조적인 단점 때문에 매 클록마다 누적결과를 출력할 수 없지만 제안

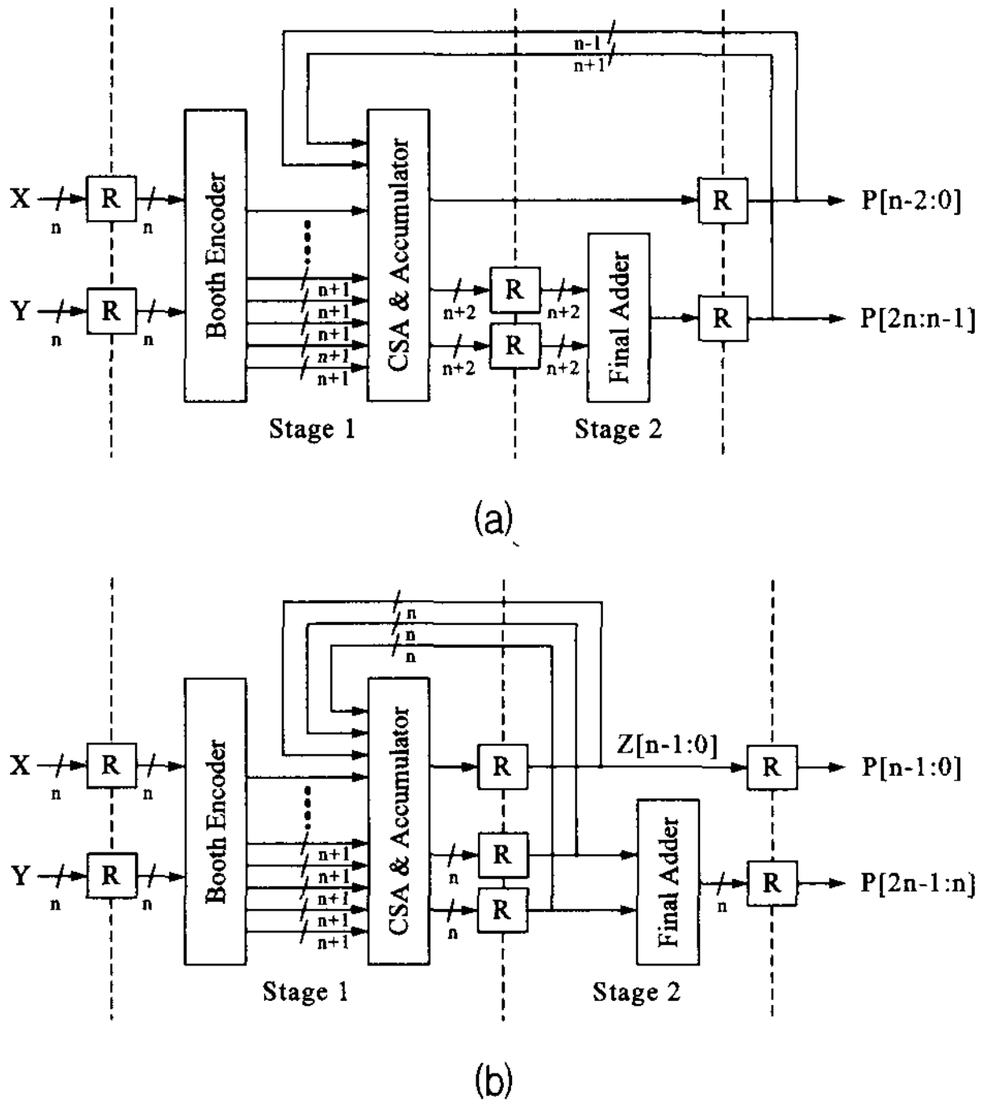

그림 6. 파이프라인된 하드웨어 구조
(a) 제안한 구조 (b) Elguibaly의 구조

Fig. 6. Pipelined hardware structure.
(a) the proposed (b) Elguibaly's structure

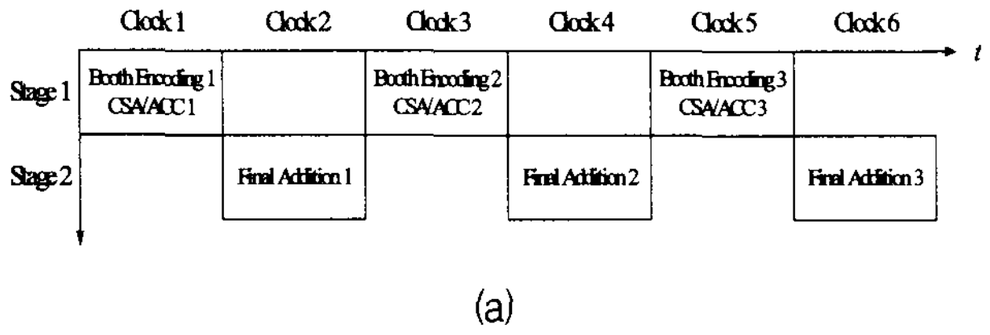

그림 7. 파이프라인 동작 순서
(a) Elguibaly의 동작 (b) 제안한 동작

Fig. 7. Pipelined operational sequence
(a) Elguibaly's (b) the proposed operation

표 9. 파이프라인과 성능 분석

Table 9. Pipeline and performance analysis.

	Standard design	Elguibaly's Design	Seo's Design
Output Rate	1 clock	2 clocks	1 clock
Pipeline Delay	139.95n+87.1	88.3n+28	44.15n+81.1

한 구조는 가능하다. 그림 6의 구조를 바탕으로 그림 7을 이용하면 두 설계의 출력율과 성능은 표 9와 같이 정리할 수 있다.

결론적으로 Elguibaly의 구조가 클록 속도는 조금 높지만 성능의 측면에서 살펴보면 제안한 구조가 약 2개 높은 성능을 갖는다.

다. 타이밍 분석

0.25, 0.18, 0.13, 및 0.90um 공정을 이용해서 합성한 후에 static timing analysis (STA) 결과를 추출하여 그림 8에 그래프로 나타내었다. 이 결과는 실제 칩을 만들 경우에 물리적인 합성(physical synthesis) 및 place&route (P&R) 과정을 위한 중요한 분석 자료이다.

미세공정에 대한 STA일수록 제안한 MAC이 Elguibaly에 비해서 더 많은 타이밍 마진(slack)을 확보하고 있다. 특히 90um공정에서는 표준 셀들의 driving force가 다양해졌기 때문에 설계된 회로의 데이터패스 및 구조가 규칙적으로 구조적일수록 design compiler는 더욱 세밀하고 정밀한 합성과 최적화를 수행할 수 있다. 왜냐하면 design compiler가 constraint의 요구 조건을 바탕으로 좋은 회로를 생성하기 위해 다양한 셀들을 mapping하고 STA를 진행하는 과정을 반복하기가 용이해지기 때문이다. 그림 8(d)에서는 slack의 차이가 20%이상이다. 일반적으로 back-end 과정에서 사용되는 EDA tool 간 correlation이 5%이하라고 가정할 때 +2ns 이상의 타이밍 마진을 확보한다면 이후의 물리적 합성 및 P&R 과정에서는 더 이상 최적화가 필요하지 않고 흔히 발생하는 routing congestion

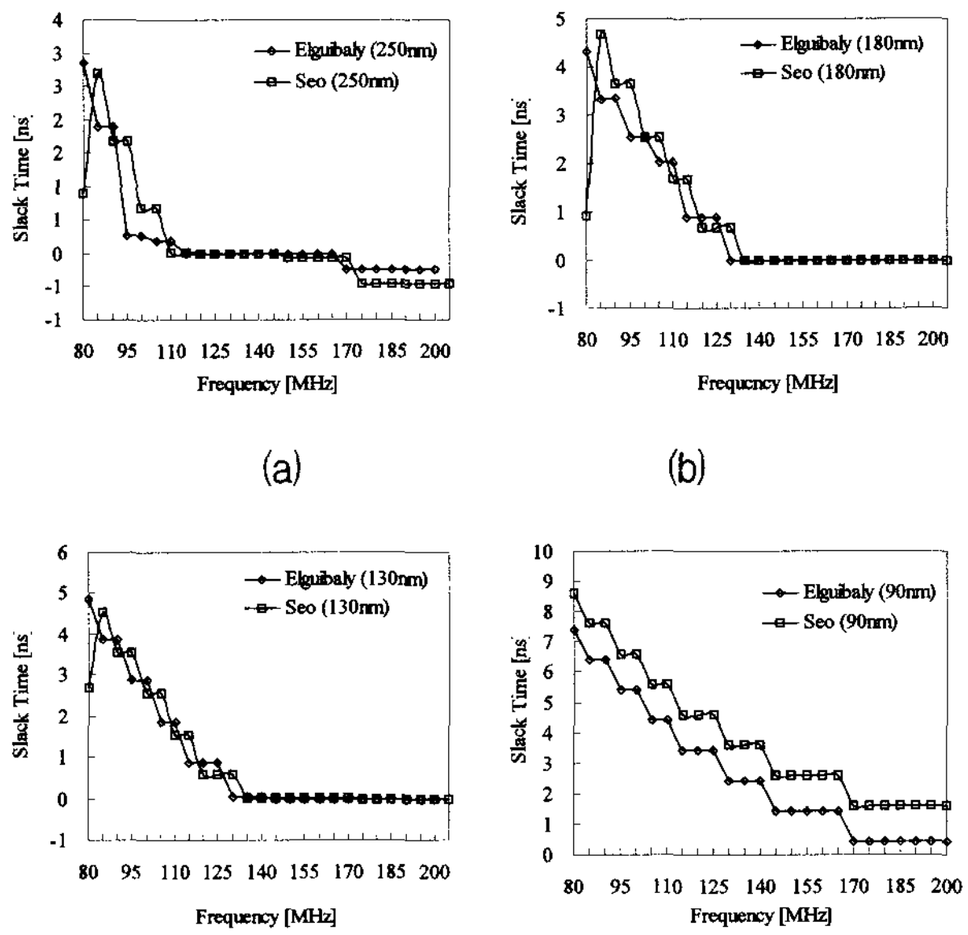

그림 8. 합성된 회로의 타이밍 분석

Fig. 8. Timing analysis of the synthesized circuits.

을 쉽게 극복할 수 있다.

합성이후에 STA 결과를 종합해보면 제안한 구조는 매우 구조적이고 규칙적이라는 결론을 내릴 수 있다.

V. 결 론

본 논문에서는 디지털 신호처리 및 멀티미디어 정보처리의 핵심 연산인 곱셈-누적 연산을 효율적으로 수행할 수 있는 새로운 MAC의 구조를 제안하였다. MAC은 가장 큰 지연시간을 갖는 독립적인 누적연산 과정을 제거하고 누적연산을 부분곱의 압축과정에 포함시킴으로써 이전 연구에 비해서 전체적인 성능을 거의 두 배 가량 향상시켰다. 제안한 하드웨어는 네 가지 CMOS 공정을 통해 구현 및 합성하였다.

이론 및 실험적인 결과를 바탕으로 볼 때 제안한 MAC은 이전 연구에 비해서 하드웨어 자원을 다소 감소시킬 수 있었다. Sakurai의 alpha power low를 이용하여 지연시간을 모델링하였고, 그 결과로 이전 연구에 비해 지연시간이 미세하게 증가하였지만 파이프라인을 고려한다면 실제 성능은 약 2배 가량 증가하였다. 전체적인 결과를 살펴보면 제안한 MAC은 표준 설계에 대해서는 여러 측면에서 매우 우수한 특성을 보였고, 이전 연구와 비교할 때 클록속도는 거의 유사하면서 성능은 두 배로 우수하였다.

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 Fina1119l 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.

[17] 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.

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

[19] 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.

[20] R. M. Rao and A. S. Bopardikar, Wavelet Transforms, Introduction to Theory and

Applications, Addison-Wesley Inc., Reading, MA, 1998.

저 자 소 개

서 영 호(평생회원)
1999년 2월 광운대학교 전자재료 공학과 졸업(공학사)
2001년 2월 광운대학교 대학원졸업(공학석사)
2000년 3월~2001년 12월 인티스닷컴(주) 연구원
2003년 6월~2004년 6월 한국전기연구원 연구원
2004년 8월 광운대학교 대학원졸업(공학박사)
2004년 11월~2005년 8월 유한대학 연구교수
2005년 9월~2008년 2월 한성대학교 교수
2008년 3월~현재 광운대학교 교수
<주관심분야 : 2D/3D 영상처리, 디지털 홀로그램, 컨텐츠 보안, SoC 설계, ASIC/FPGA>

김 동 욱(평생회원)
1983년 2월 한양대학교 전자공학과 졸업(공학사)
1985년 2월 한양대학교 대학원 졸업(공학석사)
1991년 9월 Georgia 공과대학 전기공학과 졸업 (공학박사)
1992년 3월~현재 광운대학교 전자재료공학과 교수
<주심분야 : 3D 영상처리, VLSI CAD 및 설계, 테스팅, 워터마킹 및 암호화>