고속 SIMD형 곱셈 누산기

*조민석[0], **오형철
*고려대학교 대학원 전자정보공학과
**고려대학교(서창) 공학부
ohyeong@korea.ac.kr

A High-Speed SIMD MAC Unit

*Min-Suk Cho[0], **Hyeong-Cheol Oh
*Dept. of Elec. & Info. Eng., Graduate School, Korea University
**School of Engineering, Korea University at Seo-Chang

요 약

본 논문에서는 32x32비트 곱셈 연산의 하위 32비트 결과를 한 클록 주기에 얻기 위한, 130MHz 파이프라인용 SIMD형 2단 곱셈 누산기를 설계하였다. 이 과정에서, Booth 부호기의 부분곱의 생성에 소요되는 지연을 줄이면서 부호가 있는 수의 연산을 수행할 수 있는 Booth 부호기를 설계하였다. 생성된 부분곱을 SIMD 명령어에 따라 크기가 선택된 Wallace Tree로 합산하고, 32x32비트 곱셈 연산의 하위 32비트 결과를 제외한 모든 결과들은 두 번째 파이프라인 단에서 얻어지도록 하였다. 현재 설계된 SIMD형 곱셈 누산기는 삼성 0.18㎛ 표준 셀로 합성할 때, 1.65V, +125℃에서 약 7.61ns의 임계 경로 지연을 갖는다.

1. 서 론

곱셈 누산(Multiply-and-Accumulate)은 멀티미디어 데이터 처리에 있어 핵심이 되는 연산으로, 이 연산을 수행하기 위해서는 곱셈 연산과 그 결과를 누산하는 과정이 필요하다. 따라서 매 클록 주기마다 곱셈 누산 결과를 생성하기 위해서는 고속의 곱셈 누산 능력이 요구된다.

본 논문에서는 매 클록 주기마다 32x32비트 곱셈 연산의 하위 32비트 결과를 생성하고, 파이프라인 방식으로 매 클록 주기에 누산한 결과를 얻어낼 수 있는 고성능 2단 곱셈 누산기를 설계하였다. 이 과정에서 기존의 Booth 방식을 분석하여 빠른 방식을 채택하였으며, 설계된 곱셈 누산기는 두 개의 32비트 데이터나 4개의 16비트 데이터 혹은 8개의 8비트 데이터를 연산하여 결과를 누산할 수 있는 SIMD형태를 갖도록 하였다.

본 논문의 구성은 다음과 같다. 제2절에서는 Radix-4 수정 Booth 알고리즘을 적용한 SIMD형 곱셈 누산기의 구조 및 동작을 살펴보고, 제3절에서 본 논문에서 설계한 SIMD형 곱셈 누산기를 설명한다. 제4절에서는 설계한 곱셈 누산기를 검증하고 그 성능을 평가하며, 제5절에서 결론을 맺도록 한다.

2. 곱셈 누산기(MAC)의 개요

고속 곱셈 누산기에서 많이 사용되고 있는 수정 Booth 알고리즘은 여러 가지 구현 방법이 제안되어 있는데[1,2], 본 절에서는 본 논문에서 사용하는 방법을 설명한다.

Radix-4 수정 Booth 곱셈기의 부분곱 생성기는, 승수 Y를 3비트 단위(Y_{n+1}, Y_n, Y_{n-1})로 묶어 한 비트씩 겹치도록 구성하고, 표 1에 기술된 형태의 부분곱을 산출한다. N비트의 승수와 피승수에 대하여 산출된 (N+2)/2 개의 부분곱은, 2의 보수 연산을 완료하기 위한 N개의 보수 수정 신호와 함께, Wallace Tree로 전달된다[1,2]. 곱셈기의 마지막 단계에서는 Wallace Tree에서 얻어지는, 자리 올림을 제외한 합의 값과 자리 올림값의 합을 계산하게 되는데, 이 덧셈이 가장 긴 시간을 소요하는 과정으로서 CLA등의 고속 덧셈기를 사용하여 구현된다.

생성된 64비트 곱셈 결과는 80비트로, 32비트와 16비트 곱셈결과는 각각 40비트와 20비트로 부호 확장되어 이전 주기에서 생성된 결과와 누산된다.

3. 설계된 곱셈 누산기

본 논문에서 설계한 Radix-4 수정 Booth 곱셈기의 부분곱 생성기는, 입력되는 N비트의 길이를 가진 승수로부터 N/2개의 3비트 코드를 생성하는데, 각 3비트 코드는 표 1과 같이 5가지 종류의 연산 중 하나를 지정한다[1,2]. 이 부호기와 부호로부터 지정된 종류의 부분곱을 산출하는 복호기(Decoder)로 구성된, 부분곱 생성기는 곱셈기의 구현 비용과 동작 속도에 영향을 주므로 많은 구현 기법이 제안되어 있는데, 설계 비용을 중시한 [3]의 설계에서는 식(1)과 같이 4개의 신호로 부호화 하고 그림 1과 같이 6개의 게이트를 사용하여 부분곱을 생성한다.

$$\begin{aligned} Neg &= Y_{n+1}; \\ X1 &= \sim(Y_{n-1} \oplus Y_n); \\ Z &= \sim(Y_{n+1} \oplus Y_n); \\ X2 &= Y_{n-1} \oplus Y_n; \end{aligned} \quad (1)$$

한편, 동작 속도를 중시한 [4]의 설계에서는 식(2)와 같이 3개의 신호를 그림 2에 도시된 부호기(*Booth Encoder*)를 사용하여 생성한다.

$$\begin{aligned} \text{Direction } [D_n] &= Y_{n+1}; \\ \text{Addition } [A_n] &= Y_{n-1} \oplus Y_n; \\ \text{Shift } [S_n] &= Y_{n+1} \oplus Y_n; \end{aligned} \quad (2)$$

또한, [4]에서 제안된 부분곱 생성기에서는, 완전한 2의 보수화 처리가 추가로 필요하여 식(3)과 (4)에 기술된 신호를 정의하여 보수 수정에 사용한다.

$$\text{Negative\_2X } [N_n] = Y_{n+1} \cdot \sim Y_n \cdot \sim Y_{n-1}; \quad (3)$$

$$\text{Negative\_1X } [\mathbf{N1_n}] = Y_{n+1} \cdot (Y_n \oplus Y_{n-1}) ; \quad (4)$$

생성된 보수 수정 신호 $\mathbf{N_n}$은, SIMD 모드에 따라서 부분곱의 부호를 결정한다. 부호 확장은 일반적으로 부분곱의 최상위 비트에 '1'과 반전된 최상위 비트를 추가하여 구성하나, 기술된 부분곱 생성기에 적합하도록 보수 수정 신호를 이용하여 부호 비트를 생성한다. 식(5)는 신호 $\mathbf{N_n}$을 이용하여 부분곱 생성기에서 부분곱의 부호 연산을 위한 추가 비트 생성을 보인 것이다.

$$\sim P_{ext\_sign} = \mathbf{A_n} \cdot (X_{MSB} \oplus \mathbf{D_n}) + \sim \mathbf{A_n} \cdot \sim \mathbf{N_n} \cdot (X_{MSB-1} \oplus \mathbf{D_n}) \quad (5)$$

이 신호 $\sim P_{ext_sign}$는 그림 2에 도시한 바와 같이, 1개의 2:1 Mux를 추가하여, 추가 1게이트의 지연만으로 생성한다. 그림 2는 설계된 부분곱 생성기를 도시한 것이다. 표 2에 보인 바와 같이 설계된 부분곱 생성기는 식 (1)을 사용한 [3]의 구현에 비하여, 0.18㎛ 표준 셀 구현에서, 약 0.6ns 만큼 빠르게 동작한다.

4게이트의 지연으로 생성된 부분곱은 연산 시간과 자리올림 전파 지연 시간을 중첩하도록 구성된 Wallace Tree를 이용하여, 그림 3과 같이 SIMD 방식으로 연산된다.

그림 4는 설계된 SIMD형 곱셈 누산기의 블록도이다. 생성된 부분곱의 합과 자리올림 값은 본 논문에서 요구하는 하위 32비트의 곱셈 결과를 한 클록 주기내에 생성하기 위하여, 4비트의 CLA를 구성한 32비트 CLA를 사용하였다. 그림 5는 설계된 곱셈 누산기의 2단 파이프라인 동작을 도시한 것이다. 첫 번째 클록 주기에 연산된 부분곱의 합과 자리올림 값은 파이프라인 레지스터에 저장되어, 두 번째 클록 주기에서 64비트 곱셈 결과로 연산된다. 이 64비트 곱셈 결과를 누산을 하는 경우, 덧셈으로 생기는 자리올림을 연산하기 위하여 80비트로 부호를 확장하여 처리한다. SIMD 방식으로 구성된 곱셈 연산과 누산 연산은 4비트의 CLA를 자리올림 구조로 연결하여 구성한 80비트 CLA를 이용하여 연산한다.

4. 기능 검증 및 성능 평가

설계한 SIMD형 곱셈 누산기를 Verilog HDL로 모델화한 후, Random하게 생성한 60만개의 Test Vector를 사용하여 설계 로직의 기능을 검증하였다. 기능이 검증된 곱셈 누산기를 삼성 0.18㎛ 표준 셀을 사용하여 합성한 결과, 1.65V, +125 ℃의 환경에서 7.61 ns의 임계경로 지연을 갖는 것을 확인하였다. 표 3은 설계된 SIMD형 곱셈 누산기의 임계 경로 지연 시간을 블록별로 나누어 보인 것이다.

5. 결론

본 논문에서 설계한 SIMD형 2단 곱셈 누산기는 내장형 프로세서의 기능 블록으로의 사용을 목적으로 설계되었으며, 하위 32비트를 한 클록 주기에 생성하는 구조를 전제로 하고 있다. 또한, 3비트의 SIMD 신호에 따라 한 클록 주기에서 최대 4개의 8비트 연산과 2개의 16비트 연산 그리고 1개의 32비트 연산이 가능하다. 개선된 Booth 방식을 통하여 부분곱 생성의 속도 향상을 추구하였고, 1개의 2:1 Mux 지연만으로 부호 연산과 보수 수정을 구현하였으며, SIMD 모드에 따르는 Wallace Tree와 CLA 구조를 구현하였다.

현재, 설계된 CLA 덧셈기의 임계 경로 지연을 추가로 줄이기 위하여, 8비트나 16비트로 확장된 자리 올림 선택 구조로 설계를 변경하는 등의 방안을 연구 중이다.

감사의 글

본 연구는 반도체설계교육센터(IDEC)의 지원을 받아 수행되었습니다.

참고 문헌

[1] I. Koren, *Computer Arithmetic Algorithms*, 2nd ed., Prentice-Hall, 2002.

[2] M.J. Flynn, S.F. Oberman, *Advanced Computer Arithmetic Design*, John Willey & Sons, 2001.

[3] W.-C. Yeh, C.-W. Jen, "High-speed Booth Encoded Parallel Multiplier Design", *IEEE Trans. on Computers*, Vol. 49, No. 7, pp.692-701, July 2000.

[4] K.-S. Cho, et al., "54x54-bit Radix-4 Multiplier based on Modified Booth Algorithm", *Proc. ACM Great Lakes Symp. on VLSI*, pp.233-236, April. 2003.

Y_{n+1}, Y_n, Y_{n-1}	Booth op.	Direction	Shift	Add
0 0 0	0X	0	0	0
0 0 1	1X	0	-	1
0 1 0	1X	0	-	1
0 1 1	2X	0	1	0
1 0 0	-2X	1	1	0
1 0 1	-1X	1	-	1
1 1 0	-1X	1	-	1
1 1 1	0X	1	0	0

표 1. Radix-4 수정 Booth 부호기의 동작

	Yeh&Jen[3]	설계된 SIMD MAC
지연 시간	2.08 ns	1.48 ns

표 2. 부분곱 생성기의 비교

SIMD MAC의 블록	지연 시간
부분곱 생성기	1.48 ns
Wallace Tree	4.5 ns
32비트 CLA (하위 32비트 곱셈결과)	1.63 ns
최대 임계경로	7.61 ns

표 3. 설계된 SIMD형 곱셈 누산기의 블록별 시간 지연

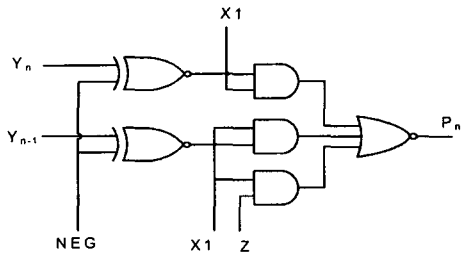

그림 1. 부분곱 생성 로직[3]

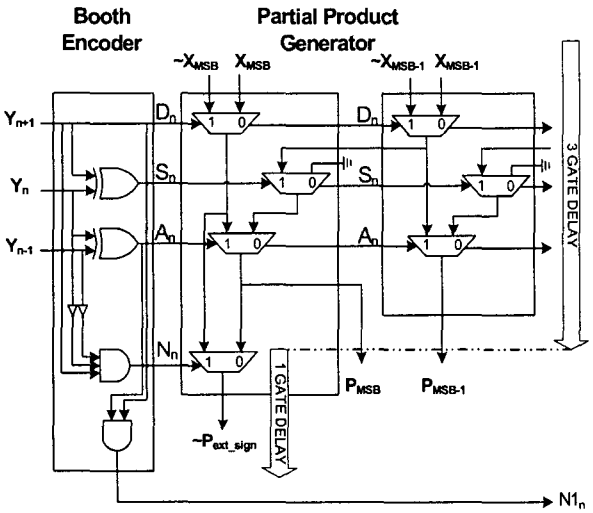

그림 2. 설계된 부분곱 생성기의 블록도

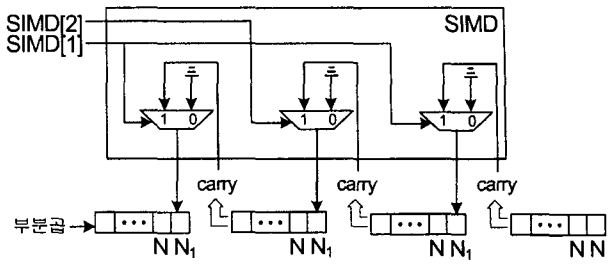

그림 3. Wallace Tree의 SIMD 블록도

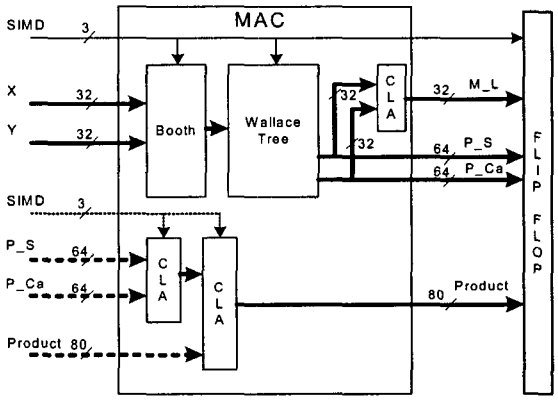

그림 4. 설계된 곱셈 누산기의 블록도

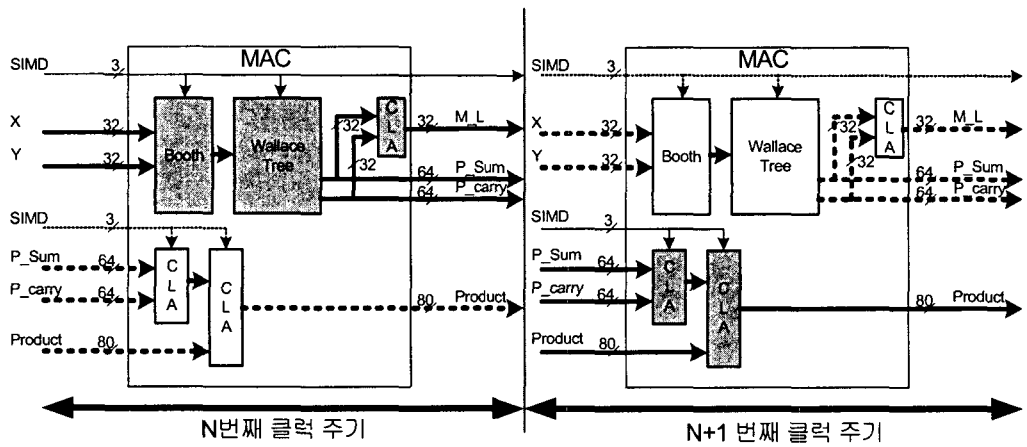

그림 5. 설계된 곱셈 누산기의 2단 파이프라인 동작