論文2003-40SD-6-9

부분곱 압축단을 줄인 32×32 비트 곱셈기

(A 32×32-b Multiplier Using a New Method to Reduce a Compression Level of Partial Products)

洪 相 玟 *, 金 炳 民 **, 鄭 麟 虎 ***, 趙 泰 元 ****

(Sang-Min Hong, Byung-Min Kim, In-Ho Jeong, and Tae-Won Cho)

요 약

고속동작을 하는 곱셈기는 DSP의 기본 블록 설계에 있어서 필수적이다. 전형적으로 신호처리분야에 있어서 반복 알고리듬은 다량의 곱셈연산을 필요로 하고, 이 곱셈연산을 첨가하고 실행하는데 사용된다. 본 논문은 32×32-b RST를 적용한 병렬 구조 곱셈기의 매크로 블록을 제시한다. Tree part의 속도를 향상시키기 위해 변형된 부분곱 발생 방법이 구조레벨에서 고안되었다. 이것은 4 레벨을 압축된 3 레벨로 줄였고, 4-2 압축기를 사용한 월리스 트리 구조에서도 지연시간을 감소시켰다. 또한, tree part가 CSA tree를 생성하기 위한 4개의 모듈러 블록과 결합이 되게 하였다. 그러므로 곱셈기 구조는 부스 셀렉터, 압축기, 새로운 부분곱 발생기(MPPG : Modified Partial Product Generator)로 구성된 같은 모듈에 규칙적으로 레이아웃 될 수 있다. 회로레벨에서 적은 트랜지스터 수와 엔코더로 구성된 새로운 부스 셀렉터가 제안되었다. 부스 셀렉터에서의 트랜지스터 수의 감소는 전체 트랜지스터 수에 큰 영향을 끼친다. 설계된 셀렉터에는 9개의 PTL(Pass Transistor Logic)을 사용한다. 이것은 일반적인 트랜지스터 수의 감소와 비교했을 때 50% 줄인 것이다. 단일폴리, 5중금속, 2.5V, 0.25㎛ CMOS공정을 사용하여 설계하고, Hspice와 Epic으로 검증하였다. 지연시간은 4.2ns, 평균 전력소모는1.81㎽/㎒이다. 이 결과들은 발표된 성능이 우수한 일반적인 곱셈기보다도 성능이 우수하다.

Abstract

A high speed multiplier is essential basic building block for digital signal processors today. Typically iterative algorithms in Signal processing applications are realized which need a large number of multiply, add and accumulate operations. This paper describes a macro block of a parallel structured multiplier which has adopted a 32×32-b regularly structured tree (RST). To improve the speed of the tree part, modified partial product generation method has been devised at architecture level. This reduces the 4 levels of compression stage to 3 levels, and propagation delay in Wallace tree structure by utilizing 4-2 compressor as well. Furthermore, this enables tree part to be combined with four modular block to construct a CSA tree (carry save adder tree). Therefore,

* 正會員, 三星電子
(Samsung Electronics)
** 正會員, 라이온텍
(Lion Tech)
*** 正會員, 忠北大學校 電子工學科
(Department of Electronic Engineering Chungbuk National University)
**** 正會員, 忠北大學校 電子工學部 및 컴퓨터 情報通信硏究所
(Department of Electronic Engineering & Research Institute for Computer and Information Communication Chungbuk National University)

※ 본 논문은 IDEC센터의 부분적 지원에 의해서 이루어졌음.
接受日字:2002年8月5日, 수정완료일:2003年5月28日

combined with four modular block to construct a CSA tree (carry save adder tree). Therefore, multiplier architecture can be regularly laid out with same modules composed of Booth selectors, compressors and Modified Partial Product Generators (MPPG). At the circuit level new Booth selector with less transistors and encoder are proposed. The reduction in the number of transistors in Booth selector has a greater impact on the total transistor count. The transistor count of designed selector is 9 using PTL(Pass Transistor Logic). This reduces the transistor count by 50% as compared with that of the conventional one. The designed multiplier in 0.25㎛ technology, 2.5V, 1-poly and 5-metal CMOS process is simulated by Hspice and Epic. Delay is 4.2ns and average power consumes 1.81㎽/㎒. This result is far better than conventional multiplier with equal or better than the best one published.

Keyword : MPPG, Multiplier, PTL, High Speed, Booth Algorithm

Ⅰ. 서 론

최근 인터넷과 영상 정보 등의 일반화에 따라 고성능의 컴퓨터가 요구되고 있다. 이러한 성능향상의 요구는 반도체 설계 기술과 집적도의 향상에 의해 가능케 되어진다. 이에 따라 큰 규모의 회로를 하나의 칩 안에 내장되도록 하는 설계가 가능하게 되었고 RISC (Reduced Instruction Set Computer) 형태의 프로세서처럼 CPU 뿐만 아니라 캐시 메모리 및 FPU(Floating Point Unit)를 하나의 칩 안에 설계를 가능케 했다. 그러나 마이크로프로세서 성능의 향상은 곱셈기의 성능향상이 필수적이다. 또한 고속 동작의 곱셈기는 통신용 알고리듬이나 다차원 신호처리 등의 응용분야에 가장 기본적으로 사용된다. 많은 경우에 있어서 곱셈의 수행시간이 시스템의 성능을 결정하게 되므로 곱셈기의 성능향상은 시스템의 성능을 좌우하게 된다.

곱셈기의 종류에는 브라운(Braun) 곱셈기, 보우-울리(Baugh-Wooley) 곱셈기, 다다(Dadda) 곱셈기, 페라리-스테파넬리(Ferrari-Stefanelli) 곱셈기 등이 있다. 이러한 곱셈기 중에서 브라운 곱셈기는 음수 연산을 수행할 수 없고, 보우-울리 곱셈기는 입력 데이터의 비트수가 증가할수록 설계 면적이 커지고 고속동작에 적합하지 않아 적용된 예가 드물다. 일반적으로 곱셈기에는 부스 알고리듬과 전가산기의 배열, 또는 부스 알고리듬과 월리스 트리 구조를 이용한 방법이 많이 이용되고 있으며[1,2], 이러한 구조를 가지는 곱셈기의 데이터 경로는 크게 부스 리코더(recoder), 열압축 트리(column compression tree), 최종단 덧셈기의 세부분으로 구성된다[4].

설계한 곱셈기의 구조는 부스 알고리듬과 규칙적인 캐리 저장 덧셈기 트리[11]를 이용하여 설계하였으며 32×32 비트의 연산을 수행하도록 설계를 하였다. 본 곱셈기에서 적용한 방법은 다음과 같다. 첫째, 구조 레벨에서 새로운 부분곱 생성 기법을 사용하여 CSA tree에서 압축단을 한단 줄임으로써 지연시간 및 전력을 줄였으며, CSA tree의 규칙성을 확보하였다. 둘째, 회로 레벨에서는 CMOS에 비해 적은 전력을 소모하는 NMOS 패스-트랜지스터 로직으로 구성된 부스 셀렉터(selector) 회로를 제안하였다.

Ⅱ. 제안한 곱셈기의 구조

본 논문에서 설계한 곱셈기의 구조는 <그림 1>에 보인 바와 같이, 부분곱을 생성하는 부스 리코더 블록, 부분곱을 압축하는 캐리 저장 덧셈기 트리 및 최종단 덧셈기로 구성된다[2, 3]. 설계한 곱셈기에서는 기존구조의 동작속도 향상을 위하여, 부스 리코더 블록에서 발생되는 부분곱의 개수를 줄이는 새로운 부분곱 발생기를 제안하였다. 이렇게 함으로써, 기존 구조에 비하여 압축기단에서의 지연시간이 감소되어 전체 곱셈기의 지연시간을 감소시켰으며 부스 인코더 및 부스 셀렉터를 새롭게 설계를 하였다. 그 외에 최종단 덧셈기로는 현재까지 보고된 병렬 덧셈기 중에서 가장 성능이 우수한 ELM 덧셈기[5]를 사용하였다.

1. 부스 리코더 블록

본 논문에서는 인코더와 셀렉터를 새롭게 설계하였다. 부스 인코더는 트랜지스터 수는 증가했지만 논리 깊이는 논문상[2, 3]에 발표되는 것과 같은 깊이로 설계

를 하였다. 부스 인코더는 전체 곱셈기에서 16개만이 사용되므로 트랜지스터수 보다는 지연시간을 최소화시키는 것이 중요하다. 반면 부스 셀렉터는 사용되는 수가 많기 때문에 트랜지스터 수뿐만 아니라 지연시간이 중요한 문제가 된다. 그렇기 때문에 본 곱셈기에서는 회로 레벨에서 전력소모를 최소화하고 고속동작을 위해 NMOS 패스-트랜지스터 로직으로 구현하였다.

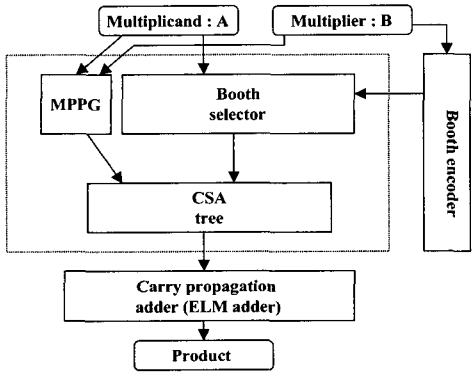


그림 1. 제안한 곱셈기의 구조
Fig. 1. Proposed multiplier architecture.

부스 인코더는 입력된 승수를 부스 알고리듬에 의하여 재코딩하는 기능을 가진다. 인코딩 방법은 설계자에 따라 다양하게 구현이 가능하며, 지금까지 다양한 종류의 인코더 회로가 제안되었다[2, 3]. 곱셈기의 고속동작을 위해서는 부스 리코더 블록에서 인코더는 부스 셀렉터와 함께 직렬로 연결되기 때문에 지연시간을 줄이는 것이 필수적이다. 설계된 곱셈기를 위한 부스 인코더의 진리표를 <표 1>에 정리하였다. 여기서 B_j, B_{j-1}, B_{j-2}는 입력되는 승수의 연속된 데이터이며, l1, l2, l3, l4, d0b는 인코딩 된 신호이다. 이 신호들은 부스 셀렉터 회로의 제어신호로 작용을 한다. 제안한 인코딩 방식은 다음과 같다. 먼저 승수의 3비트를 검색하여 승수가 부스 알고리듬에 의하여 '0'으로 재코딩 될 경우 d0b 신호가 0이 되어 부스 셀렉터의 PMOS를 구동하게 된다. 만약 승수가 +1 또는 -1로 재코딩 되면 l1 또는 l2의 신호가 1이 되어 승수 또는 승수의 보수가 부스 셀렉터로 인가되게 된다. 승수가 +2 또는 -2로 재코딩 될 경우 l3 또는 l4의 신호가 1이 되어 1 비트 이동(1-bit left shift)된 피승수 또는 1 비트 이동된 피승수의 1의 보수가 부스 셀렉터로 인가된다. 제안한 부스 인코더 회로는 <그림 2>와 같다. 이 방식은 Ohkubo의 논문[13]이나 Goto의 논문[3]에 제안된 방식에 비하여 트랜지스터의 수는 많지만, 논리 깊이를 같도록 설계를 하여 지연시간에는 차이가 없도록 하였다.

부스 셀렉터는 부분곱을 생성하여 압축기단으로 전송하므로 고속으로 데이터를 처리할 수 있는 기술이 요구될 뿐 아니라, 부스 리코더 블록의 대부분을 차지하므로 단위 셀에서 전력소모를 최소화하기 위한 설계기법이 적용되어야 한다. 회로 레벨에서의 저전력 실현을 위해, NMOS 패스-트랜지스터 로직으로 구현한 부스 셀렉터 회로를 제안하였으며, <그림 3>에 회로도를 도시하였다. <그림 3>의 회로에서 입력 신호 l1, l2, l3, l4, d0b는 부스 인코더의 출력에서 전달된 신호이며, $A_i, \overline{A_i}, A_{i-1}, \overline{A_{i-1}}$ 신호는 피승수의 값이다. 부스 셀렉터는 부스 인코더의 출력을 인가 받아 선택된 피승수의 신호를 출력한다. 이 출력값이 재코딩된 부분곱이며, 압축단으로 입력된다. 한편 기존에 발표된 정적 CMOS 등으로 설계된 부스 셀렉터 셀[2]에 비해, 제안한 회로는 입력 커패시턴스가 작고 논리문턱전압이 낮아 동작속도가 높고 전력소모가 적은 장점이 있다.

표 1. 제안한 부스 인코더의 진리표.
Table 1. Proposed truth table of Booth encoder.
Mj : negative partial product
2Xj : doubled
Xj : not doubled
l1 : not doubled & positive partial product
l2 : not doubled & negative partial product
l3 : doubled & positive partial product
l4 : doubled & negative partial product
d0b : zeroed

Inputs				Usual			Proposed				
B_j	B_{j+1}	B_{j+2}	Func	X_j	$2X_j$	M_j	l1	l2	l3	l4	d0b
0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	+A	1	0	0	1	0	0	0	1
0	1	0	+A	1	0	0	1	0	0	0	1
0	1	1	+2A	0	1	0	0	0	1	0	1
1	0	0	-2A	0	1	1	0	0	0	1	1
1	0	1	-A	1	0	1	0	1	0	0	1
1	1	0	-A	1	0	1	0	1	0	0	1
1	1	1	0	0	0	1	0	0	0	0	0

제안한 회로에서 d0b가 '0'일 경우를 제외한 모든 출력은 NMOS 2개와 인버터 1개만큼의 지연시간을 나타내므로 지연시간이 단축되는 반면에, NMOS 패스-트랜지스터 로직만으로 구현할 경우 NMOS 네트워크의 출력

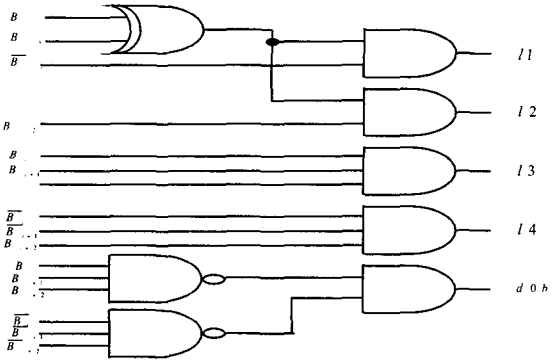


그림 2. 제안한 부스 인코더의 논리 회로도
Fig. 2. Proposed booth encoder circuit.

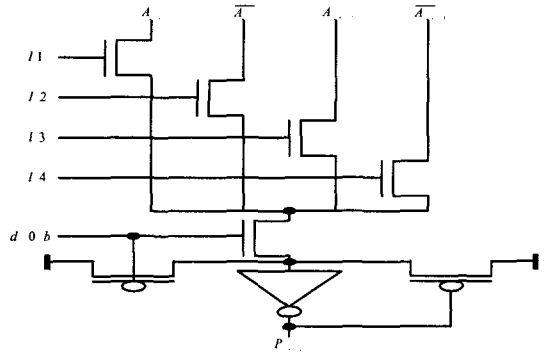


그림 3. 제안한 부스 셀렉터의 회로도
Fig. 3. Proposed booth selector circuit.

값 논리 '1'에 대한 전압레벨이 NMOS의 문턱전압만큼 낮아짐으로 인하여 잡음여유가 악화되는 단점이 있다[10].

이러한 경우를 위해 제안한 회로에서는 인버터의 출력에 weak PMOS를 이용한 궤환경로를 형성함으로써 출력이 풀 스윙을 할 수 있도록 설계하였다. 제안한 부스 셀렉터 회로는 9개의 트랜지스터로 구성되어 있고 정적 CMOS 로직 형태에서 24개의 트랜지스터로 구현된 것 보다 면적을 적게 차지하는 장점이 있다. 부스 셀렉터는 부스 리코더 블록에서 (N+2)M/2개가 사용되므로 면적의 중요성이 커진다.

2. 캐리 저장 덧셈기 트리(CSA tree)

일반적인 부스 알고리듬을 적용한 곱셈기 구조에서 N×M 비트 ($\log M = k$, k: 정수) 곱셈에서 부분곱단은 (M/2)+1단이 발생된다[21, 62]. <그림 4>는 N×32 비트의 곱셈기를 나타내는데, 그림에서 보는 것처럼 마지막 1단은 부호 확장 및 M/2번째 부분곱단의 2의 보수화로 생기는 캐리로 인해 발생된다. 이렇게 생긴 부분곱단들은 압축기단에서 캐리 저장 덧셈기 트리를 통해 압축이 되는데 일반적으로 사용하는 캐리 저장 덧셈기로는 4-2 압축기, 전가산기, 반가산기가 사용 가능하다. <그림 4>처럼 승수가 32비트이면 17단의 부분곱이 생기게 된다. 17단의 부분곱을 4-2 압축기를 사용하면, 위의 8단은 4-2 압축기를 사용해 압축하고, 마지막에 생기는 1단은 4-2 압축기로 압축을 하게 되면 하드웨어의 낭비가 심하므로, 전가산기 1단으로 압축을 하면 된다. 이러한 압축방법을 <그림 5>에 제시하였다. <그림 6>은 9-2 압축기를 사용한 구조를 나타낸다. 9-2 압축기를 사용할 경우 압축단이 2단으로 형성되는 것처럼 보이나 9-2 압축기는 <그림 7>처럼 구성이 되므로 4-2 압축기를 사용한 구조 <그림 5>와 같은 압축단을 같게 된다.

결국 기존의 부스 알고리듬을 사용한 부분곱들은 4-2 압축기와 전가산기를 이용할 경우 <표 2>에서 보는 것처럼 $\log_2 \frac{M}{2}$ 단의 압축기단이 생기게 되고, 32×32 비트 곱셈기는 4-2 압축기 3단과 마지막에 1단의 전가산기 단을 필요로 하게 된다. 이렇게 생긴 압축단 4단은 배정밀도를 위한 54×54 비트 곱셈기에서의 4단의 압축단과 같은 단수를 같게 됨으로써, 두 곱셈기간의 임계경로를 구성하는 캐리 저장 덧셈기 트리에서의 지연시간이 차이가 나질 않게 된다. 더욱이, 전가산기 단의 추가로 인한 캐리 저장 덧셈기 트리의 규칙성을 확보하기가 어려워진다.

위에서 언급한 문제점은 N×M 비트 곱셈기에서 항상 일어나는 문제이다. 이러한 일반적인 구조를 사용한 비트별 곱셈기의 캐리 저장 덧셈기 트리 그림이 2.8, 2.9에 도시되어있다. 그림에서 보면 압축기단의마지막에 전가산기 단이 삽입됨으로써, 4-2 압축기로만 구성되는 압축단에 전가산기단이 삽입되어 규칙성이 떨어지는 것을 볼 수 있다. 이러한 전가산기 단의 추가적인 삽입을 막기 위해 설계한 곱셈기에서는 전가산기 단에서 연산이 필요한 (M/2)+1번째 부분곱 $P_{0,(M/2)}$-1(M/2번째 부분 곱의 2의 보수화를 위한 캐리)비트와 2^N(부호 확장을 위한 1값)비트를 1번째 부분 곱($P_{1,0}$)에서 함께 연산하도록 함으로써 CSA tree에서 전가산기 단을 제거하였다.

본 논문에서는 $A_N \times B_M$ 비트를 갖는 곱셈기에서, 일반적인 부스 셀렉터를 사용할 경우, 압축기단에 인가되는 (M/2)+1 번째 부분곱의 단수를 제거하기 위하여 캐리와 부호 생성항을 첫 번째 부분곱과 함께 연산할 수

표 2. 비트별 요구되는 압축단의 개수
Table 2. Level of compression for N×M-b multiplier.

Bits	Partial product bits	Level of 4-2 compression	Level of 3-2 compression		Level of compression	
			conventional	proposed	conventional	proposed
8×8	5	1	1	0	2	1
16×16	9	2	1	0	3	2
24×24	13	3	0	0	3	3
32×32	17	3	1	0	4	3
54×54	28	4	0	0	4	4

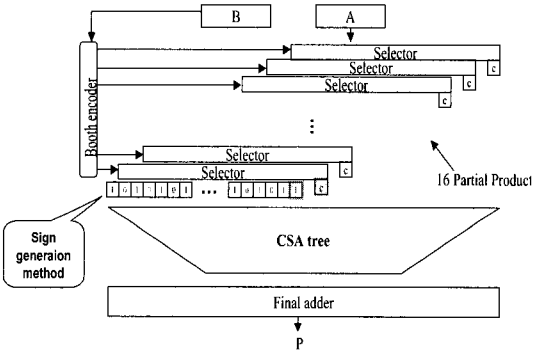


그림 4. 일반적인 N×32 비트 곱셈기의 구조
Fig. 4. General architecture of N×32-b multiplier.

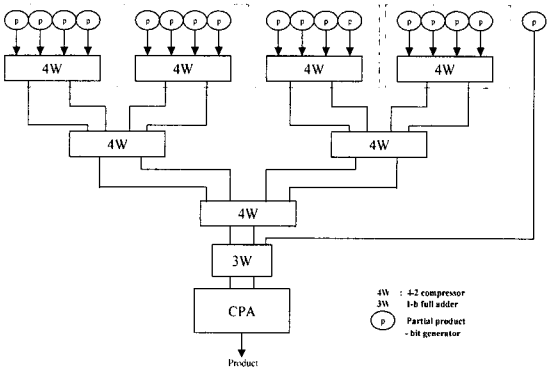


그림 5. 일반적인 32×32 비트 곱셈기의 압축 방법 1
Fig. 5. General compression method 1 of 32×32-b multiplier.

있도록 기존의 구조를 수정했다. 식 (1) ~ 식 (3)은 기존 첫 번째 부분곱의 N-2, N-1 및 N 비트 위치에서의 부분곱이 발생하는 식이다.

여기서 $B_{-1}=0$ 이며, 1의 보수로 표기하며 $P_{I,J}$ 는 I

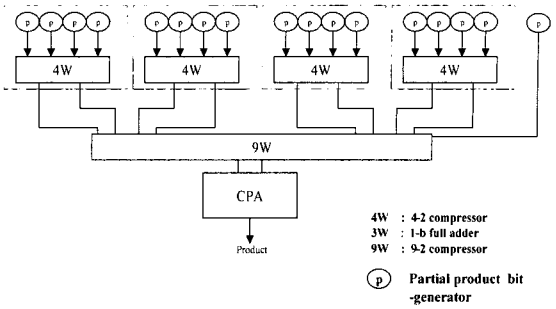


그림 6. 일반적인 32×32 비트 곱셈기의 압축 방법 2
Fig. 6. General compression method 2 of 32×32-b multiplier.

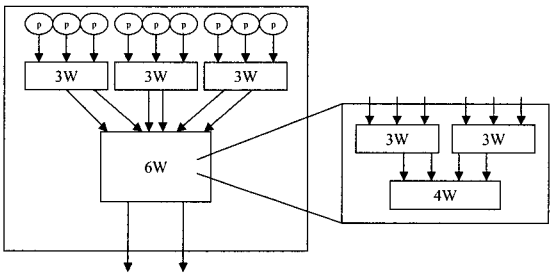


그림 7. 9-2 압축기의 구조.
Fig. 7. Architecture of 9-2 compressor.

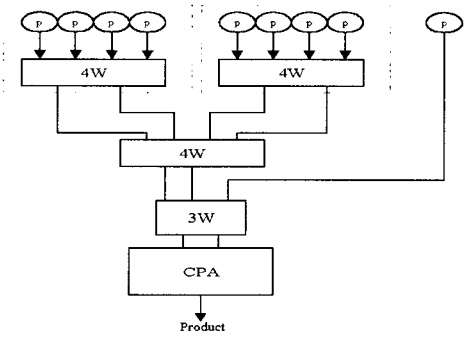


그림 8. 일반적인 16×16 비트 곱셈기의 압축 방법
Fig. 8. General compression method of 16×16-b multiplier.

번째 열 J번째 행을 나타낸다.

$$P_{N-2,0} = A_{N-2}(2B_1 + B_0 - B_{-1}) \quad (1)$$

$$P_{N-1,0} = A_{N-1}(2B_1 + B_0 - B_{-1}) \quad (2)$$

$$P_{N,0} = \overline{A_{N-1}}(2B_1 + B_0 - B_{-1}) \quad (3)$$

위의 식에 전가산기로 압축해야 하는(M/2)+1번째 부분곱인 $P_{C0,(M/2)}$-1와 2^N을 자릿수에 맞춰 더하면 식 (4) ~ 식 (8)과 같이 표현된다.

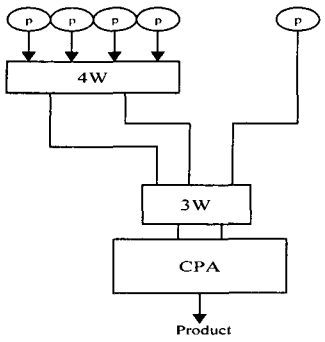


그림 9. 일반적인 8×8 비트 곱셈기의 압축 방법
Fig. 9. General compression method of 8×8-b multiplier.

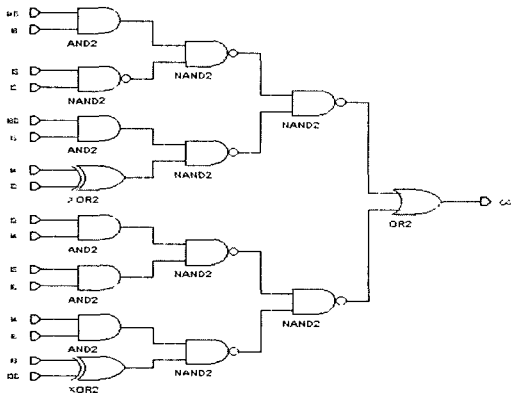


그림 10. (N-2)번째 비트부분곱 생성기 ($SP_{N-2,0}$)
Fig. 10. (N-2)th bit MPPG ($SP_{N-2,0}$).

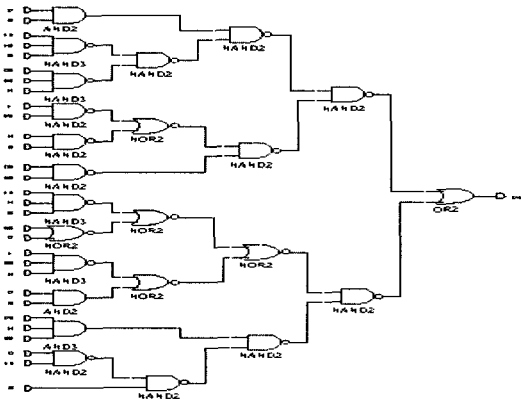


그림 11. (N-1)번째 비트부분곱 생성기 ($SP_{N-1,0}$)
Fig. 11. (N-1)th bit MPPG ($SP_{N-1,0}$).

$$P_{C0,(M/2)-1} = (\overline{B_{N-3}} \wedge \overline{B_{N-2}}) \wedge B_{N-1} \quad (4)$$

$$(P_{N-2,0} + P_{C0,(M/2)-1})2^{N-2} = (SP_{N-2,0} + CP_{N-2,0})2^{N-2} \quad (5)$$

$$(CP_{N-1,0} + P_{N-1,0})2^{N-1} = (SP_{N-1,0} + CP_{N-1,0})2^{N-1} \quad (6)$$

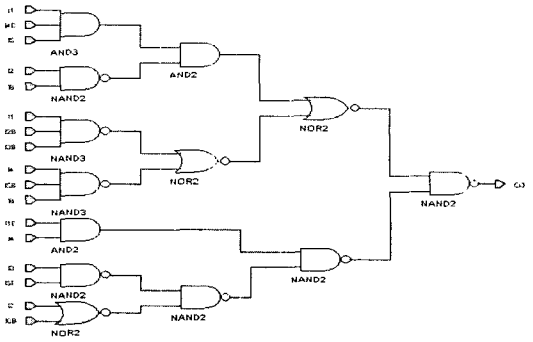


그림 12. (N)번째 비트부분곱 생성기($SP_{N,0}$)
Fig. 12. (N)th bit MPPG ($SP_{N,0}$).

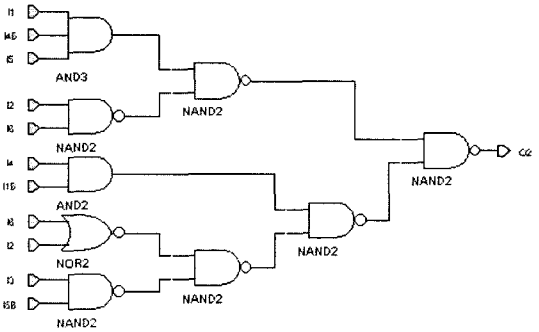


그림 13. (N+1)번째 비트부분곱 생성기 ($SP_{N+1,0}$)
Fig. 13. (N+1)th bit MPPG ($SP_{N+1,0}$).

$$(CP_{N,0} + \overline{P_{N,0+1}})2^{M} = (SP_{N,0} + CP_{N,0} + 1)2^{N} \quad (7)$$

$$(CP_{N,0} + 1)2^{N+1} = (SP_{N+1,0} + CP_{N+1,0})2^{N+1} \quad (8)$$

위의 식에서 , $2^{N}, 2^{N+1}$는 부호 생성 기법에 의해 계산된 항이다. 식에서 보듯이 기존의 N비트까지 부분곱을 발생시키던 것과는 달리 N+2 비트까지 연산결과가 나오는데, 이러한 이유는 부호 발생 기법에 의해 2^{N+1}값이 계산되어있고 그 값을 N 비트 자리의 캐리와 더해줘야 하기 때문이다. 그리고 N+1 비트 자리의 캐리는 N+2 비트 자리로 이동하므로 N+2 비트까지 연산되어야 한다. 하지만 위의 식으로부터 구현을 하면 캐리가 전파되어 지연시간이 오히려 기존 방법보다 커지게 된다. N-1 비트의 캐리가 계산되고 그 뒤에 N비트가 계산되고 차례로 캐리가 전파되어 N+2 비트까지 연산이 되기 때문이다. 이러한 캐리의 전파를 막기 위해서는 각 자리의 부분곱 생성기가 독립적으로 연산이 되어야 한다. 이를 위해서는, 위의 식을 그대로 구현하지 말고 진리표를 만든 뒤 그 진리표를 통해 새로운 부분곱 발생기를 설계하면 각 비트의 값들은 독립적으로 승수와

표 3. 제안한 부분곱 발생 방법에 의한 부스 인코더의 진리표

Table 3. Truth table of booth encoder for proposed partial product method. i1 : A_{N-1}, i2 : A_{N-2}, i3 : A_{N-3}, i4 : B_1, i5 : B_0, i6 : $P_{C0,(M/2)-1}$, o1 : $SPN_{-2,0}$, o2 : $SPN_{-1,0}$, o3 : $SP_{N,0}$, o4 : SP_{N+1}, o5 : $CP_{N+2,0}$

Input	Output	Input	Output
i1,i2,i3,i4,i5,i6	o1,o2,o3,o4,o5	i1,i2,i3,i4,i5,i6	o1,o2,o3,o4,o5
000000	10000	100000	10000
000001	10001	100001	10001
000010	10000	100010	01110
000011	10001	100011	01111
000100	01111	100100	10011
000101	10000	100101	10100
000110	01111	100110	10001
000111	10000	100111	10010
001000	10000	101000	10000
001001	10001	101001	10001
001010	10000	101010	01110
001011	10001	101011	01111
001100	01110	101100	10010
001101	01111	101101	10011
001110	01111	101110	10001
001111	10000	101111	10010
010000	10000	110000	10000
010001	10001	110001	10001
010010	10001	110010	10010
010011	10010	110011	10000
010100	01101	110100	10001
010101	01110	110101	10010
010110	01110	110110	10000
010111	01111	110111	10001
011000	10000	111000	10000
011001	10001	111001	10001
011010	10001	111010	01111
011011	10010	111011	10000
011100	01100	111100	10000
011101	01101	111101	10001
011110	01110	111110	10000
011111	01111	111111	10001

피승수에 따라 연산을 할 수 있게 된다. 위의 식으로부터 진리표를 만들면 <표 3>과 같게 된다. 진리표에서 i1, i2, i3은 피승수 N-1, N-2, N-3 비트를 의미하며, i4, i5는 피승수 1, 0 비트를 의미한다. i6은 M/2 번째 부분곱의 2의 보수화를 위한 캐리이다. o1, o2, o3, o4, o5는 N+2 ~ N-2 비트 자리의 부분곱 출력이다.

① 제안한 부분곱 발생기(MPPG)

<표 3> 진리표로부터 카노맵을 작성한 뒤 부울 함수로 변환하면 식 (9) ~ 식 (13)과 같이 된다.

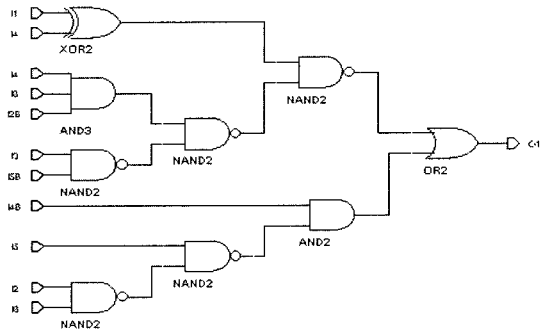


그림 14. (N+2)번째 비트 부분곱 생성기 ($CP_{N+2,0}$).
Fig. 14. (N+2)th bit MPPG ($CP_{N+2,0}$).

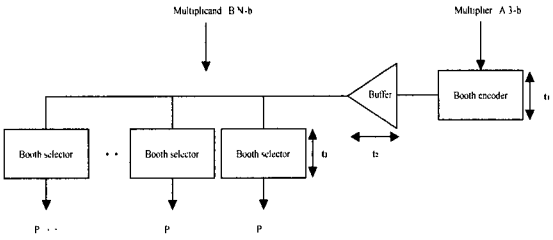


그림 15. 부스 리코더 블록의 지연시간
Fig. 15. Delay time of Booth recoder block.

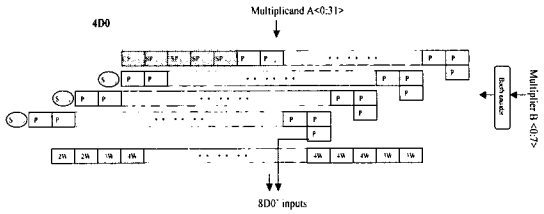


그림 16. 부분곱 압축단(4D0)
Fig. 16. Compression stage of Partial products(4D0).

$$o5 = \overline{i4}\,i6(\overline{i5}+\overline{i2}) + i5\,\overline{i6(i2 \oplus i4)} + i2\,i4\,i5\,i6 + i4\,\overline{i5}(\overline{i3}\,\overline{i6} + i3\,i6) \quad (9)$$

$$o4 = i2\,i6(\overline{i1}\,\overline{i4}\,i5 + \overline{i3}\,i4\,\overline{i5}) + \overline{i1}\,i4\,i5(\overline{i6}+i2) \quad (10)$$
$$+ \overline{i2}\,i4\,\overline{i5}(i1\,\overline{i3}+\overline{i6}) + i1\,\overline{i4}\,i5(\overline{i2}+\overline{i6}) + i1\,\overline{i2}\,i4\,i6(i5+i3)$$

$$o3 = i1\,\overline{i4}\,i5(\overline{i2}+\overline{i6}) + \overline{i1}\,i4(\overline{i6}+i3\,\overline{i5}+i2) + i1\,i2\,\overline{i3}\,i4\,\overline{i5}\,i6 \quad (11)$$

$$o2 = i1\,\overline{i4}\,i5(\overline{i2}+\overline{i6}) + \overline{i1}\,i4(\overline{i6}+i3\,\overline{i5}+i2) \quad (12)$$

$$o1 = \overline{i1 \oplus i4} + \overline{i2}\,i4\,i6(\overline{i3}+i5) + \overline{i4}(\overline{i5}+i2\,i6) \quad (13)$$

위의 식을 게이트 레벨 회로로 구현한 것이 <그림 10 ~ 14>이다.

한편, 위의 부분곱 생성 방법이 압축단에서 전가산기 단을 제거한 효과를 위해서는, MPPG가 식 (14)보다 지연시간이 작거나 같아야한다.

$$t_r = t_1 + t_2 + t_3 \quad (14)$$

그리고 최소한 제안한 방법이 압축기단에서 전가산기를 사용한 일반적인 구조보다 성능이 저하되지 않기 위해선 식 (15)의 지연시간과 작거나 같아야한다.

$$t_{rf} = t_r + t_{full\ adder} \quad (15)$$

즉, 제안한 방법이 식 (14)보다 작거나 같으면 전가산기 단을 제거한 효과가 나타나는 것이고, 식 (15)와 같게되면 제안한 방법이 일반적인 구조와 성능이 같다는 것을 보인다.

② MPPG를 이용한 캐리 저장 덧셈기 트리

MPPG를 사용하면(M/2)+1번째 부분곱인 $P_{C0,(M/2)-1}$와 2^N을 함께 연산함으로써 전가산기 단이 제거된다. <그림 16>은 제안한 방법을 적용한 압축단(4D0)의 구성을 보인다. <그림 16>에서 $SP_{N-1,0}$, $SP_{N,0}$, SP_{N+1}, SP_{N+2}, $CP_{N+2,0}$는 제안한 MPPG이다.

<그림 17>에서는 각 블록 4D0, 4D1을 4-2 압축기를 통과시킴으로써 8D0를 형성하고 4D2와 4D3을 같은 방법으로 압축하여 8D1을 형성한다. 마지막으로 8D0과 8D1을 4-2 압축기를 사용하면 각 비트 당 2비트의 출력이 형성되며, 이 출력들은 최종단 덧셈기로 인가된다.

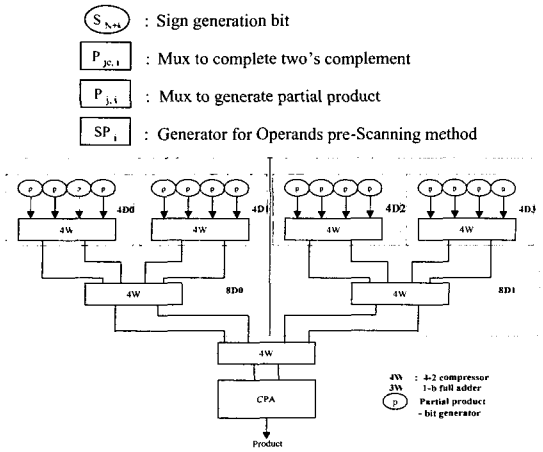


그림 17. 제안한 방법을 이용한 캐리 저장 덧셈기 트리
Fig. 17. CSA tree using the proposed method.

<그림 18, 19>는 제안한 MPPG $SP_{N-1,0}$, $SP_{N,0}$, SP_{N+1}, SP_{N+2}, $CP_{N+2,0}$을 이용한 8×8, 16×16 비트 곱셈기의 캐리 저장 덧셈기 트리를 나타낸다. <그림 8, 9>에 비교할 때 전가산기 단이 제거되어 있음을 알 수 있다. 즉 <그림 17 ~ 19>의 구조를 통해 제안한 부분곱 발생기를 사용함으로써 캐리 저장 덧셈기 트리가 기존의 구조보다 규칙성을 확보할 수 있음을 알 수 있다.

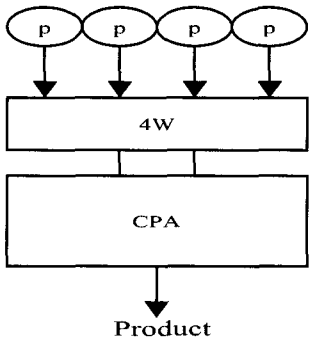


그림 18. 8×8 곱셈기의 캐리 저장 덧셈기 트리
Fig. 18. 8×8-b CSA tree using the propsed method.

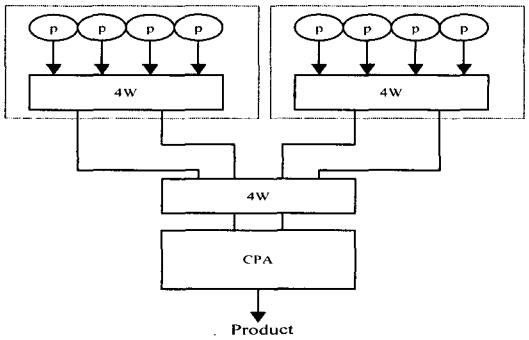


그림 19. 16×16 곱셈기의 캐리 저장 덧셈기 트리
Fig. 19. 16×16-b CSA tree using the propsed method.

3. 제안한 방법을 이용한 계산 예

기존의 계산 방법 및 제안한 방법을 이용해 계산하는 도식적인 방법을 <그림 20>과 <그림 21>에 나타내었다. 여기서는 8×8 비트 곱셈을 예로 들었으며 승수와 피승수 모두 0일 때를 예로 들었다.

<그림 20>은 8×8 비트 곱셈을 일반적인 수정된 부스 알고리듬을 사용해서 계산한 예이다. 승수가 8비트이므로 수정된 부스 알고리듬을 사용해서 부분곱이 4단으로 줄었으며, 5번째 부분곱은 부호 확장 및 4번째 부분곱의 2의 보수화를 위한 캐리로 구성이 된다. 그림에서 보듯이 총 5단의 부분곱이 생성되는 것을 알 수 있다.

<그림 21>을 보면 <그림 20>과 달리 부분곱이 5단이 아닌 4단으로 줄어 있음을 알 수 있다. 먼저 제안한 방법으로 계산을 하면 입력으로 피승수 최상위 3비트

를 i1, i2, i3으로 인가하고, 승수 최하위 2비트를 i4, i5로 입력 받는다. i6은 식 (4)로 계산이 된다.

이렇게 해서 생긴 값 "000000"은 <표 3>의 진리표에 대응되는 "10000"의 값을 얻는다. 이 값을 첫 번째 부분곱의 최상위 5비트에 값을 넣어 주면 계산이 된다. 기존의 계산 방법과 비교하면 부분곱이 1단이 줄고 동일한 결과를 얻음을 볼 수 있다.

설계한 곱셈기의 레이아웃을 <그림 22>에 보였다. 부스 리코더와 캐리 저장 덧셈기 트리를 함께 배치를 함으로써 연결선의 길이를 최소화하였다. 레이아웃 결과 곱셈기의 면적은 0.95×1.1㎟이고, 전체 트랜지스터의 수는 26,029개임을 확인하였다. 각 블록별 트랜지스

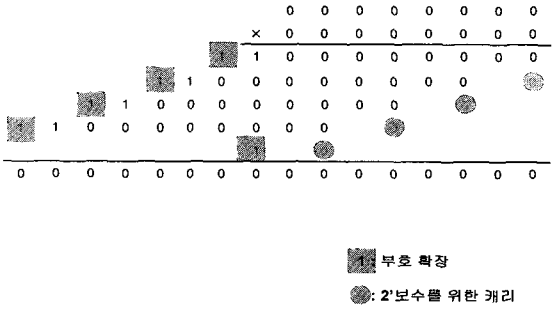


그림 20. 기존의 방법으로 계산한 예
Fig. 20. Example of conventional method.

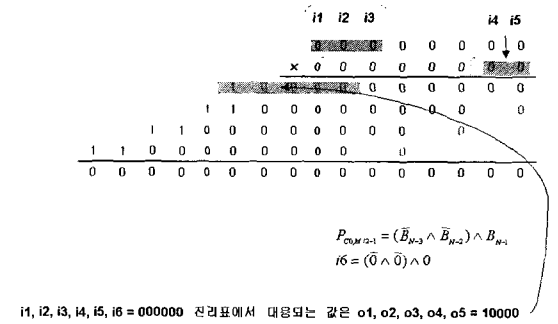


그림 21. 제안한 방법을 이용한 계산 예
Fig. 21. Example of proposed method.

표 4. 기능 블록별 트랜지스터의 수
Table 4. Numbers of transistor of functional blocks.

Functional block	Booth recoder	CSA tree	ELM adder	Total
Numbers of transistor	7,813	13,614	4,218	26,029

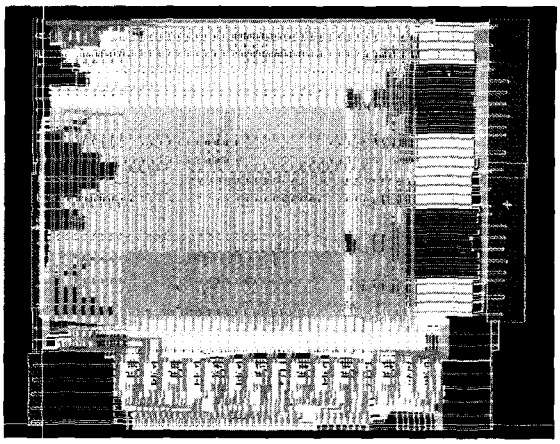


그림 22. 설계한 32×32 비트 곱셈기의 레이아웃 도면
Fig. 22. Physical Layout of 32×32-b multiplier.

터의 수를 <표 4>에 도시하였고, <그림 23>은 각 기능블록별 트랜지스터수의 비율을 나타낸다.

Ⅲ. 실험결과 및 분석

모의실험은 Hspice를 이용하여 지연시간을 측정하였으며, EPIC을 이용해 평균 전력 소모를 측정하였다. 공정변수는 ANAM 0.25㎛, 2.5V, CMOS 공정을 이용하였다. 온도는 25℃, 표준 공정, 전원 전압은 2.5V, 입력 신호의 상승 및 하강 시간은 각각 0.1ns로 설정하였다. 제안한 MPPG의 지연시간은 N×M 비트 (M은 2^k, k: 정수) 곱셈기에서 적용할 때의 지연시간을 비교하였다. 비교 대상으로는 8×8, 16×16, 32×32 곱셈기에서의 부스 리코더의 지연시간을 대상으로 하였다.

식 (14)에 나타나는 것처럼 비트 수가 증가할수록 부스 리코더의 지연시간 t_r은 버퍼의 지연시간 t_2가 증가하므로 커지게 된다. 제안한 방법이 기존방법[63]에 비해 성능이 우수하기 위한 조건은 식 (15)의 지연시간보다 작아야하며 식 (14)보다 작거나 같을 경우는 전가산기 단을 제거한 효과가 나타나는 경우이다. <표 5>은 제안한 MPPG의 지연시간, 비트별 곱셈기의 부스 리코더단의 지연시간 및 전가산기의 지연시간을 보여준다.

<표 6>는 제안한 방법과 일반적인 구조를 사용한 곱셈기에서 최종단 덧셈기를 제외한 나머지 블록에서의 성능을 나타낸다. 제안한 방법은 PDP 측면에서 기존의 구조보다 11.5 ~ 22.5 %의 감축률을 보이고 있다. 전가산기 한 단의 지연시간 만큼 차이가 나질 않는 이유는, 전가산기의 3개의 입력신호 a, b, cin이 동시에 들

어갈 경우는 전가산기 한 단의 지연시간을 보이지만, 곱셈기에서 전가산기에 동시에 입력이 들어갈 수는 없고 입력 벡터에 따라 3개의 신호 중 임계경로를 형성하는 입력과 sum 출력사이에 a, b 두 입력이 먼저 cin보다 인가가 되면 전가산기 한 단의 지연시간이 줄어들게 된다.

제안한 부스 셀렉터는 PTL(Pass Transistor Logic) 로직으로 구현되어졌다.

부스 리코더의 지연시간은 부스 인코더와 부스 셀렉터의 지연시간으로 이루어진다.

표 5. 제안한 부분곱 발생기, 전가산기 및 부스 리코더단의 지연시간

Table 5. Delay time of proposed partial product generator, full adder and booth recoder.

Proposed partial product generator(ns)	Full adder (ns)	Delay of booth recoder (ns)		
		8×8	16×16	32×32
0.44	0.2	0.33	0.40	0.45

표 6. 제안한 방법과 기존 방법의 PDP 비교

Table 6. Comparison of PDP between conventional and proposed method.

Multiplier (Bits)	Delay(ns)		Power(㎽)		PDP(pJ)	
	Conventional	Proposed	Conventional	Proposed	Conventional	Proposed
8×8	0.87	0.75	2.138	1.919	1.86	1.44
16×16	1.18	0.95	9.935	8.833	11.72	8.39
32×8	1.02	0.90	11.58	10.59	11.81	9.53
32×32	1.64	1.50	69.53	67.21	114.02	100.81

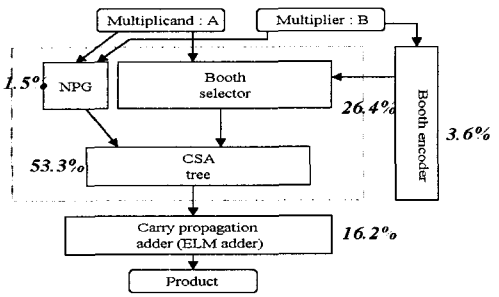


그림 23. 설계한 곱셈기의 트랜지스터 비율

Fig. 23. Block diagram of multiplier. The italic numbers shows the percentage of number of transistors used in each part with respect to the total transistor count in designed multiplier design.

설계한 곱셈기 성능을 비교하기 위해서는 비트 수, 공정 변수 등 여러 가지 조건이 같은 대상을 선정해야 한다. 그러나 32×32 비트 곱셈기의 경우 0.25㎛ 공정으로 설계한 논문은 보고된바 없으며, 0.8㎛ 공정에서 설계한 32×32 비트 곱셈기[63]와 0.25㎛ 공정으로 설계된 54×54 비트 곱셈기[2, 3]과 간접적인 비교를 하였다.

<표 7>의 참고 문헌[11]은 본 연구과제의 곱셈기와 같은 비트를 갖는 곱셈기이다. 직접적으로 비교할 수 있는 항목은 트랜지스터의 수로서 본 연구에서의 트랜지스터가 6.4% 적은 것으로 나타난다. 전력소모 및 지연시간은 최소 선폭과 전원전압 등이 일치하지 않고 최소 선폭에 대한 정확한 환산방법이 알려져 있지 않으므로, 더 이상의 비교는 불가능하다. 참고 논문 [2]와 [3]은 54×54 비트 곱셈기로 본 연구과제의 곱셈기보다 비트 수가 크지만 공정이 0.25㎛로 같기 때문에 간접적인 비교 수행하였다. 참고 문헌 [3]의 54×54 비트 곱셈기는 현재 발표된 가장 성능이 우수한 곱셈기이다. 지연시간 측면에서 문헌[3]과의 간접적인 비교를 위해서는, 리코더 블록, 캐리 저장 덧셈기 트리, 최종단 덧셈기를 구분해서 비교를 해야한다. 먼저 Goto의 곱셈기를 32×32 비트 곱셈기로 비트 수를 낮추고자 할 때, 버퍼링의 지연시간을 고려하지 않는다면 부스 리코더 블록의 지연시간은 54×54 비트의 곱셈기와 32×32 비트의 곱셈기가 같게 된다. 캐리 저장 덧셈기 트리는 일반적인 32×32 비트 곱셈기가 4-2 압축기 3단 전가산기 1단으로 구성되며, 4-2 압축기는 전가산기 2단으로 구성된다. 즉 32×32 비트 곱셈기의 캐리 저장 덧셈기 트리를 전가산기의 단수로 변환하면 7단으로 구성이 된다[4]. 54×54 비트 곱셈기는 4-2 압축기 4단으로 구성이 되어 있으며 전가산기로 변환시 8단으로 된다[4]. 최종단 덧셈기는 54×54 비트의 곱셈기의 덧셈기를 108 비트가 사용되며 32×32 비트의 곱셈기에는 64 비트가 사용된다.

일반적인 병렬 덧셈기의 지연시간은 $\log_2 N+1$로 나타낼 수 있으므로 108 비트 덧셈기는 8, 64 비트 덧셈기는 7의 지연시간 비율을 갖는다[5].

Goto의 논문에서 부스 리코더의 지연시간을 0.73ns, CSA tree의 지연시간은 1.99ns, 최종단 덧셈기의 지연시간을 1.38ns로 발표되어 있으므로, 트랜지스터의 증가로 인한 버퍼링을 무시하고 지연시간 비율에 따라 32×32 비트 곱셈기로 변환하면 CSA 트리의 지연시간은 1.75ns, 최종단 덧셈기의 지연시간은 1.21ns가 나온다.

부스 리코더의 지연시간을 0.73ns라 하면 32×32 비트의 곱셈기는 3.69ns 가 나오게 된다. 즉 본 연구에서 제안한 곱셈기는 지연시간 면에서 12.1%의 성능감소를 보인다. 하지만 위의 비교는 Goto의 곱셈기를 트랜지스터수의 감소로 인한 버퍼링의 지연시간 감소의 영향을 배제한 것이므로 직접적인 비교는 되지 않는다.

표 7. 다양한 곱셈기들의 성능 비교
Table 7. Performance comparison in various multipliers.

	Proposed	Reference[11]	Ohkubo[2]	Goto[3]
Process	0.25㎛ CMOS	0.8㎛ CMOS	0.25㎛ CMOS	0.25㎛ CMOS
Precision(bits)	32×32	32×32	54×54	54×54
Supply voltage(V)	2.5	5.0	2.5	2.5
Multiplication time(ns)	4.2	15	4.4	4.1
Power dissipation(㎽/㎒)	1.81	27		2.23
Layout area(㎜×㎜)	0.95×1.1	2.68×2.71	3.77×3.41	1.04×1.27
Transistor count	26,029	27,704	100,200	60,797

Ⅳ. 결 론

본 연구에서는 32×32 비트 곱셈기를 설계하였으며, 설계한 곱셈기에서는 알고리듬, 구조, 회로 측면에서 새로운 방법들의 제안 및 이미 발표된 회로들을 분석하여 적용하여 설계하였다.

고속동작과 규칙성을 확보하기 위해 MPPG를 제안하였으며, 그로 인해 PDP측면에서 일반적인 구조에 비해 부스 리코더와 캐리 저장 덧셈기 트리에서 11.5 ~ 22.5% 향상되었음을 모의실험 결과 확인을 하였다. 제안한 MPPG는 $N\times M$ 비트 (M은 2^k, k: 정수) 곱셈기에 모두 적용가능하며, $\log_2 \frac{M}{2}$의 압축단에서 전가산기단을 제거할 수 있어 곱셈기의 규칙성, 및 지연시간 감소 및 전력소모 면에서도 장점을 갖는다. 또한 곱셈기의 핵심 블록중의 하나인 부스 셀렉터 회로를 NMOS 패스-트랜지스터 로직으로 구현하여 저전력 구현 및 동작속도를 향상시켰다. 제안한 회로는 정적 CMOS로 구현한 회로에 비해 트랜지스터의 수가 적고 지연시간이 작은 장점을 가지고 있으나, 저전압 특성에서는 단점을 가진다.

0.25㎛ 파라메타를 가지고 모의실험 결과, 지연시간은 4.2ns, 전력소모는 1.81(㎽/㎒)이었다. 논문 수준의 비교 대상에서 공정 차이로 인한 정확한 비교는 할 수 없었지만 비교 대상 논문[11]에 비해 트랜지스터 수는 6.4%의 감소를 나타냈고 동일 비트의 곱셈기에 비해 성능의 우수성을 보였다.

참 고 문 헌

[1] Akilesh Parameswar, H. Hura and T. Sakurai, "A Swing Restored Pass-Transistor Logic-Based Multiply and Accumulate Circuit for Multimedia Applications," IEEE Journal of Solid-State Circuits, vol. 31, no. 6, pp. 804~809, Jun. 1996.

[2] Norio Ohkubo et al., "A 4.4ns CMOS 54×54-b Multiplier Using Pass-Transistor Multiplexer," IEEE Journal of Solid-State Circuits, vol. 30, no. 3, pp. 251~257, Mar. 1995.

[3] Gensuko Goto et al., "A 4.1-ns Compact 54×54-b Multiplier Utilizing Sign-Select Booth Encoders," IEEE Journal of Solid-State Circuits, vol. 32, no. 11, pp. 1676~1682, Nov. 1997.

[4] Vojin G. Oklobdzija and D. Villerger, "Improving Multiplier Design by Using Improved Column Compression Tree and Optimized Final Adder in CMOS Technology," IEEE Transactions on VLSI Systems, vol. 3, no. 2, pp. 292~301, Jun. 1995.

[5] T. P. Kelliber, R. M. Owens, M. J. Irwin and T. T. Hwang, "ELM-A Fast Addition Algorithm Discovered by a Program," IEEE Transactions on Computers, vol. 41, no. 9, Sep. 1992.

[6] 김문수, 유범선, 강성현, 이중석, 조태원, "하이브리드 로직 스타일을 이용한 저전력 ELM 덧셈기 설계," 전자공학회 논문지, 제 35권 C편, 제 6호, pp. 389~396, 1998

[7] 유범선, 조태원, "가변 크기 셀을 이용한 저전력 고속 16비트 ELM 가산기 설계," 전자공학회 논문지, 제 35권 C편, 제 8호, pp. 637~645, 1998

[8] 유범선, 이기영, 조태원, "글리치 감소를 통한 저전력 16비트 ELM 덧셈기 구현," 전자공학회 논문지, 제 36권 C편, 제 5호, pp. 346~355, 1999

[9] Beom Seon Ryu, Hyoung Sok Oh, Kihak Shim, Kie Young Lee and Tae Won Cho, "Multi-level Approaches to Low Power 16 bits ALU Design," IEEE Asia Pacific Conference on ASICs, pp. 158~161, 1999.

[10] Sung-Mo Kang, Yusuf Leblebici, CMOS Digital Integrated Circuits : Analysis and Design, McGraw Hill, pp. 322~340, 1996.

[11] Masato Nagamatsu, "A 15-ns 32×32-b CMOS Multiplier with an Improved Parallel Structure," IEEE Journal of Solid-State Circuits, vol. 25, no. 2, April 1990.

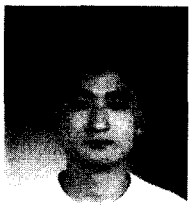
저 자 소 개

洪 相 玟(正會員)

1999년 2월 : 충북대학교 전자공학과 학사. 2001년 2월 : 충북대학교 전자공학과 석사. 2001년 2월~현재 : 삼성전자 DS 총괄 SYSTEM LSI. <주관심분야 : Image processing, Digital signal processing>

金 炳 民(正會員)

2000년 2월 : 충북대학교 전자공학과 학사. 2002년 2월 : 충북대학교 전자공학과 석사. 2002년 1월~현재 : (주) 라이온텍 기술연구소 전임 연구원. <주관심분야 : 저전력 회로설계>



鄭 麟 虎(正會員)

2001년 2월 : 청주대학교 전자.정보통신.반도체 공학부 학사. 2002년 2월~현재 : 충북대학교 전자공학과 석사과정. <주관심분야 : 저전력 회로설계, 마이크로프로세서 설계>

趙 泰 元(正會員)

1973년 2월 : 서울대학교 전기공학과 학사. 1973년~1984년 : 금성전선(주). 1986년 5월 : 미국 루이빌대학교 전자공학과 석사. 1992년 5월 : 미국 켄터키주립대 전자공학과 박사. 1992년 3월~현재 : 충북대학교 전기전자공학부 교수. <주관심분야 : 마이크로프로세서 설계, 집적회로 설계, Routing, 저전력 회로 설계>