論文2002-39SD-5-11

병렬배열구조를 사용한 비동기 곱셈기

(Asynchronous Multiplier with Parallel Array Structure)

朴 贊 鎬 *, 崔 炳 秀 **, 李 東 翊 **

(Chan-Ho Park, Byung-Soo Choi, and Dong-Ik Lee)

요 약

본 논문에서는 기존의 배열구조의 문제점인 전력낭비와 느린 연산속도를 보완하기 위하여 병렬배열구조를 채택하고 비동기 시스템에 적합하도록 평균 연산속도를 최소화한 곱셈기를 제안한다. 실험 결과 제안된 비대칭 병렬배열구조는 기존의 배열구조와 비교하였을 때, 평균 55% 정도의 연산시간 단축이 가능하며, 이 구조를 이용한 Booth 인코딩 비동기 곱셈기는 기존의 Booth 인코딩 배열 곱셈기에 비해 40% 정도의 시간 단축 효과가 있음을 확인하였다.

Abstract

In this paper an asynchronous array multiplier with a parallel array structure is introduced. This parallel array structure is used to make the computation time faster with a lower power consumption. Asymmetric parallel array structure is used to minimize the average computation time in an asynchronous multiplier. Simulation shows that this structure reduces the time needed for computation by 55% as compared to conventional booth encoding array structures and that the multiplier with the proposed array structure shows a reduction of 40% in the computational time with a relatively lower power consumption.

Ⅰ. 서 론

곱셈기는 DSP나 마이크로 프로세서의 데이터 처리부에서 연산시간이 길고, 많은 전력을 소모하는 부분이다. 따라서 동기식 시스템에서는 파이프라인 형태나, 여러 클럭주기동안 연산을 하는 경우가 많으며, 성능에 있어서 병목으로 작용하기도 한다.

곱셈기에는 여러가지가 있으나, 대표적으로 다음과 같이 세가지로 나누어 볼 수 있다.

1) Wallace 트리 형태[1]

2) Iterative 형태

3) 배열구조 형태

Wallace 트리 형태의 곱셈기는 연산속도가 빠르나 불규칙적인 구조로 인하여 배열구조와 비슷한 개수의 CSA(Carry Save Adder)를 사용하지만 면적이 넓어지고 구현이 어려워지는 단점이 있다. 또한 비동기 시스템의 장점인 평균연산시간을 갖는 곱셈기를 구현하기가 힘들다. Iterative 형태의 곱셈기는 면적이 작은 장점이 있으나 래치의 시간지연으로 인한 연산속도의 저하라는 문제점을 가지고 있다. 배열구조의 곱셈기는 규칙적인 구조로 설계가 간단하여 많이 사용되고 있으나, 트리 형태에 비해 속도가 느리고 전력소모가 큰 단점을 가지고 있다.

비동기 프로세서는 전역클럭을 사용하지 않고 각각의 블록에서 req와 ack신호에 의하여 컨트롤이 이루어지므로 블록에서 연산이 종료되는 시간을 알수 있다면

* 正會員, 韓國電子通信研究院
(Electronics & Telecommunication Research Institute)
** 正會員, 光州科學技術院 情報通信工學科
(Dept. of Info. & Comm., Kwang-Ju Institute of Science and Technology)

接受日字:2000年6月14日, 수정완료일:2002年2月7日

항상 critical path의 연산시간을 소요하는 동기식 프로세서와는 달리 이상적인 경우 평균연산시간을 가지는 빠른 프로세서를 구성할 수 있다. 그러나, 마이크로 파이프라인 구조[2]의 비동기 프로세서에서 데이터 의존적인 연산시간을 가진 블록이 존재하여도 그 단계에서의 가변적인 지연시간이 다른 블록의 고정연산시간보다 적으면 효과를 볼 수 없다. 파이프라인 구조에서는 전체의 속도가 가장 느린 블록에 의하여 좌우되기 때문이다. 예를 들어 1~5ns 정도의 가변적인 지연시간을 가진 블록이 그 효과를 보려면 5ns이상의 고정적인 지연시간을 가진 블록이 없어야 한다. 따라서 데이터 의존적인 회로는 곱셈기처럼 전체 시스템에서 비교적 큰 지연시간을 차지하는 부분일수록 효과가 크다.

이 논문에서는 기존의 배열구조의 단점을 보완하여 빠른 연산시간과 적은 전력소모를 가지는 비동기 곱셈기를 제안하였다. 기존 배열구조의 전력낭비를 줄이기 위하여 LR(Left to Right) 구조를 일부 사용하였으며, 배열을 둘로 나누어 병렬적인 연산을 수행함으로써 전체적인 연산속도를 높였다. 또한 비동기 시스템에 적합하도록 데이터 패턴의 특성을 이용하여 비대칭적으로 배열을 나누어 평균연산속도를 높였다.

표 1. 곱셈명령의 데이터 패턴
Table 1. Data pattern of multiplication.

Architecture	benchmark program	Average bit length	
		multiplicand	multiplier
SPARC	lisp	5.323	8.000
	gcc	5.805	3.668
	ijpeg	7.689	11.236
Simplescalar	lisp	1.535	0.431
	gcc	6.692	9.353
	perl	0.006	4.005
	go	3.703	4.404
	ksim	11.430	12.876

이 논문의 구성은 다음과 같다. 2장에서는 벤치마크 프로그램의 데이터 패턴을 분석하고, 기존의 배열구조가 가지는 문제점에 대하여 설명한다. 3장에서는 이를 해결하기 위한 병렬배열구조를 설명하고 연산속도를 높이기 위하여 Booth 인코딩을 사용한 경우에 이 구조의 성능에 대하여 설명한다. 비대칭 병렬배열구조를 이용한 비동기 곱셈기에 대하여 4장에서 설명하고 있으며, 그 실험결과는 5장에 나타나 있다. 마지막으로 결론은 6장에서 기술한다.

Ⅱ. 기존의 배열 구조

1. 입력 데이터 패턴

표 1은 SPEC INT95[3] 프로그램을 Simple- scalar architecture[4]와 SUN SPARC[5]의 기반에서 추출한 입력 데이터의 평균 길이(leading 0 를 제거한 bit 수)를 보여주고 있다. 음수연산의 경우는 operand들을 양수로 변환하였다. 이 표를 살펴보면 32×32bit 곱셈이라 할지라도 실제 입력 데이터의 평균 길이는 약 7~8bit 정도로 길지 않음을 알 수 있다. C 컴파일러의 경우는 32bit 곱셈의 연산 결과중 하위 32bit만을 결과로 사용하지만 이와같은 곱셈 명령어의 입력 패턴 때문에 결과를 얻는데는 큰 문제가 없으며, 이를 위하여 상위 32bit의 결과는 연산하지 않고 overflow만을 체크하는 곱셈기가 설계되기도 하였다.[6] 그림 1은 비교적 곱셈 명령어가 많은 ijpeg과 go 그리고 ksim의 데이터 패턴을 나타내었다.

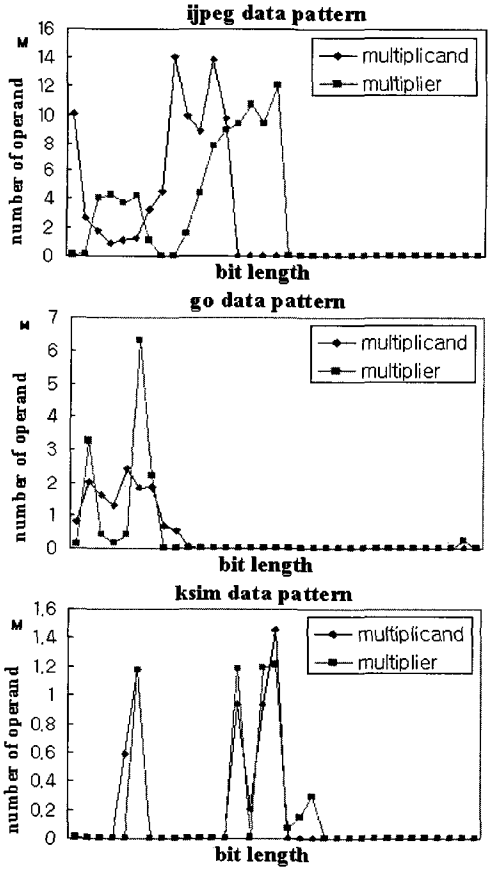

그림 1. 입력 데이터의 평균 비트수
Fig. 1. Bit length of input operands.

2. 기존 배열구조의 문제점

기존의 배열구조는 Wallace 트리구조와 비교할 때, 연산속도가 느린 단점을 가지고 있다. 32×32bit Booth 인코딩 곱셈기의 경우 Wallace 트리구조를 사용하면 마지막 덧셈기에 입력될 두 개의 partial result를 얻는데 6단계의 CSA가 필요한 반면 배열구조는 15단계의 CSA가 필요하다.

또 다른 단점은 위와 같은 데이터 양상에서 기존의 배열구조를 사용하면 낭비되는 전력이 많다는 점이다. 그림 2(a)는 16bit 곱셈기에서 12×6bit의 곱셈이 일어날 때 사용되는 CSA 셀들을 보여준 그림이다. 실질적으로 연산에 필요한 CSA 셀은 검은색으로 표시되어 있으며, 빗금 친 부분은 실제 연산에는 필요치 않으나 배열구조상 데이터가 전파되어 스위칭이 일어나는 CSA 셀들, 즉 낭비되는 전력이다. 이러한 낭비는 배열구조의 전력소모를 크게 만드는 원인중 하나이며 피승수의 길이가 길어질수록 심해진다.

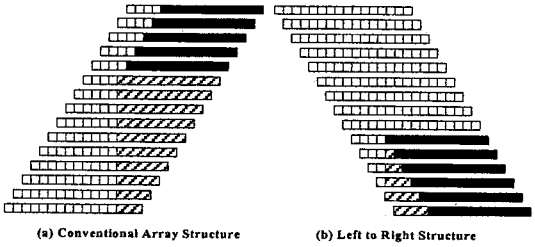

그림 2. 배열구조에서의 전력소모
Fig. 2. Power consumption in array structure.

Ⅲ. 병렬 배열 구조

1. Left to Right Scheme

앞에서 말한 배열구조에서 낭비되는 전력을 줄이는 한 방법으로 그림 2(b)와 같이 LR(Left to Right) 방식[7]을 사용할 수 있다. 일반적인 배열구조는 부분합 중에서 Weight가 가장 낮은 것부터 먼저 더해가는 RL(Right to Left) 방식이나, LR방식은 Weight가 가장 높은 부분합부터 더해가는 방식이다. 그림 3은 이 LR 방식과 일반적인 RL 방식의 입력패턴에 따른 특성을 비교해 보여주고 있다. 역시 실제 사용되는 셀은 검은색으로, 낭비되는 셀은 빗금으로 표시되어 있다. 그러나 이 LR 방식은 승수의 길이가 피승수의 길이보다 긴 경우에는 그림 3(c)와 같이 오히려 전력의 낭비가 커지며, 최종 덧셈 단계에서 더해주어야 하는 bit수가 늘어나는 단점이 있다.

2. Divided Array Structure

배열구조의 느린 연산속도에 대한 해결책으로 그림 4와 같이 배열의 상·하를 나누어 계산한 후, 두 결과를 더하는 방법을 생각할 수 있다. 이와 같은 경우에는, 두 결과를 합하는 단계가 더 필요하지만 병렬적인 연산을 수행하여 최대 두 배의 연산속도를 낼 수 있다.

소비 전력 측면에서 이 구조의 효과는 다음과 같다. 그림 4처럼 배열의 윗부분에서만 이루어지는 연산이

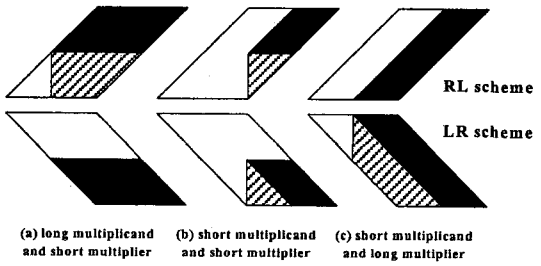

그림 3. RL방식과 LR방식의 전력소모 비교
Fig. 3. Comparison between LR scheme and RL scheme.

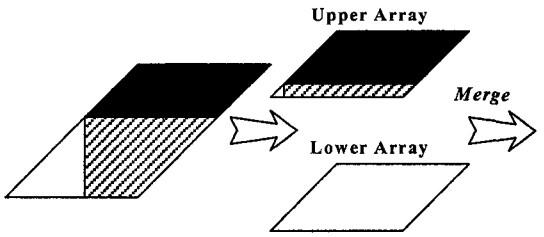

그림 4. 분할된 배열구조
Fig. 4. Divided Array Structure.

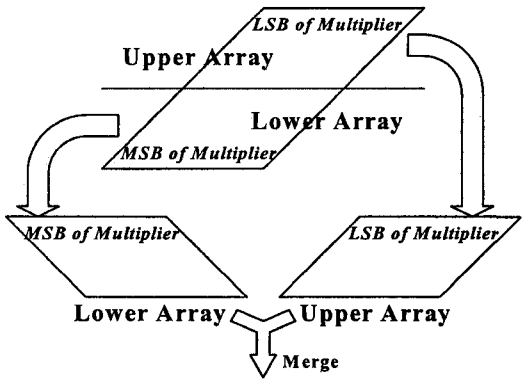

그림 5. 병렬배열구조
Fig. 5. Parallel Array Structure.

있다면 아래부분으로 전파되는 데이터가 없어 기존방식의 단점으로 지적되었던 전력의 낭비를 줄일 수 있으나, 승수의 길이가 절반을 넘어갈 경우에는 데이터의 전파가 기존의 배열구조와 동일하고, 마지막 합산 단계가 필요하여 전력소모가 오히려 늘어나는 단점이 있다.

3. 병렬 배열 구조

그림 5와 같은 병렬배열구조는 지금까지 기술한 문제점을 보완하기 위하여 사용될 수 있다. LR 방식과 RL방식을 조합한 Hybrid 형태의 곱셈기[8]는 전력 소모 측면에서 이득을 볼수 있는 구조이다. 병렬배열구조는 이 Hybrid 방식을 divided array structure에 적용한 형태로써 상위 부분은 기존의 배열구조를 이용하고 하위 부분은 LR 방식을 이용하여 빠른 연산시간과, 승수의 길이가 짧은 경우에 전력낭비를 줄일 수 있는 장점을 가지고 있다. 또한 승수의 길이가 긴 경우에 전력소모가 커지는 단점을 보완하기 위하여 하위 배열에 LR 방식을 사용하였다. 승수의 길이가 길다고 할지라도 상위 배열에 의하여 16bit(32bit operand의 경우)가 잘려나간 상태이므로 하위 배열에 입력되는 승수의 길이는 상당히 짧다. 이런경우 LR 방식을 사용하여 전력낭비를 줄일 수 있다.

4. Booth 인코딩

제안된 곱셈기는 빠른 연산을 위하여 흔히 사용되는 Booth 인코딩[9]을 적용, 부분합의 숫자를 줄여 연산속도를 높였다. 곱셈기를 구현할 때 사용한 인코딩 방법은 One-hot 인코딩 방식을 사용하였다. 즉 Mux를 구동하는 신호로 negative, shift, zero의 세가지를 사용한 것이 아니라, -2A, -A, zero, +A, +2A의 다섯 가지를 사용하였다.

승수나 피승수가 양수로 변환되어 계산되는 경우는 피승수의 상위 bit가 대부분 0이므로 데이터의 길이가 짧다. 그러나 Booth 인코딩을 이용할 경우, 결과에 따라서 피승수를 감산해야하는 경우가 있다. 이 경우 피승수의 2의 보수를 더해주게 되므로 연산을 할 때 더해지는 수는 양수와 음수가 모두 필요한 것을 알 수 있다. 음수의 경우 상위 bit가 모두 1로 채워지므로 결과적으로 데이터의 길이를 늘리는 효과를 가져온다. 승수의 경우 상위 bit는 양수, 음수에 관계없이 대부분 인코더의 결과를 0으로 만들게 되므로 짧은 승수와 긴 피승수를 가진 연산형태에 가깝게 된다. 이러한 경우 일반적인 구조를 사용하면 전력을 많이 낭비하게 되며, 병렬배열구조를 사용하면 낭비되는 전력을 크게 줄일 수 있다.

Ⅳ. 비동기 곱셈기

1. 비동기 곱셈기

회로의 연산은 항상 같은 시간을 소모하지는 않는다. 즉, 입력이 바뀜에 따라서 결과가 안정적으로 나오는데 걸리는 latency는 입력 데이터에 따라서 달라진다. 동기식 시스템은 클럭의 주기를 회로의 critical path에 맞추어 동작시킨다. 반면 비동기 시스템은 각 블록에서 연산 종료 신호를 만들어내고 이 신호가 다음단계에 request 신호로 들어간다. 따라서 입력 패턴에 따른 연산종료 시간을 찾아낼 수 있다면 평균 연산시간을 가지는 빠른 시스템을 만들 수 있다. 그러므로 병렬배열구조를 사용하여 비동기 시스템에 적용할 곱셈기를 디자인 할 때 중요한 점은 평균연산시간을 최소화하는 것이다.

데이터 의존적인 연산시간을 가지는 곱셈기는 이미 제안되었다.[10] 이 곱셈기는 Booth 인코딩을 이용하지 않은 unsigned 곱셈기로서 승수가 1이면 덧셈을 하고 0이면 윗단계에서 전달된 데이터를 다음단계로 바로 pass하는 배열구조를 가지고 있다. 데이터를 바로 pass하는 시간은 덧셈을 하는 시간보다 짧아서 데이터 의존적인 연산이 가능하다. Booth 인코딩을 사용하더라도 이와같은 방법을 적용할 수 있다. 즉 인코딩 된 결과가 0이면 pass하는 방법을 사용할 수 있다. 그러나 랜덤한 데이터라고 가정하면 인코딩된 결과가 0일 확률은 25% 밖에 되지 않는 단점이 있다. 게다가 Booth 인코딩을 사용할 때, 인코딩 된 결과가 0이라 할지라도 부호확장을 위하여 앞의 두개의 bit는 1을 더해주어야 한다[11]. 이 때문에 병렬배열구조는 LR 방식을 사용하는 하위 배열에서 기존방식의 적용이 불가능하다. 따라서 여기서는 leading 0 또는 1을 이용한 데이터 의존적 연산을 구현하였다.

2. 비대칭 병렬 배열 구조

곱셈연산의 데이터 패턴을 살펴보면 상위 bit은 대부분 부호 bit로 채워져 있으며, 하위 배열은 승수의 MSB(Most Significant Bit)부터 연산을 시작하기 때문에 부호 bit이 끝날 때까지의 합은 0임을 쉽게 알 수 있다. 따라서 하위 배열은 상위 배열보다 훨씬 빠른 연

산이 가능하다. 그러나 하위 배열의 연산이 끝나더라도 상위 배열의 연산이 끝나지 않으면 두 부분합을 더할 수는 없다. 따라서 상위 배열과 하위 배열의 연산시간의 균형은 전체 평균연산시간을 높이는 중요한 요소이며, 상·하위 배열의 연산시간을 비슷하게 해 주기 위해서는 하위 배열의 길이를 더 길게 하여야 한다.

상위 배열에서도 기존의 방식[10]을 사용하여 데이터 의존적인 연산시간을 얻는 것이 가능하나 상위 배열에는 데이터들이 비교적 균일하게 분포해 있고, 따라서 인코딩 된 결과가 0일 확률은 약 25%밖에 되지 않는다. 또한 이를 위한 부가적인 회로와 그 전력소모가 있으므로, 상위 배열에서 데이터 의존적인 연산을 하는 것은 큰 효과를 얻을 수 없다. 결과적으로 제안된 구조에서는 연산시간이 비교적 고정된 상위 배열은 짧게 놓고, 데이터에 따라 빠른연산이 가능한 하위 배열을 길게 놓아 평균연산시간을 상위 배열과 밸런싱하는 방법을 사용하였다.

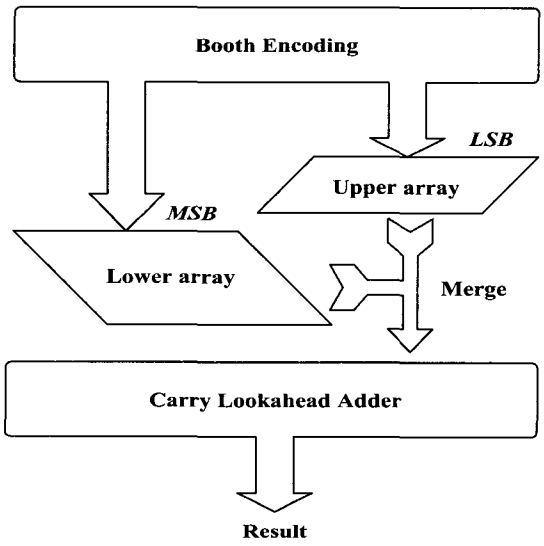

그림 6. 비대칭 병렬배열구조를 사용한 비동기 곱셈기
Fig. 6. Asynchronous Multiplier with Asymmetric Parallel Array Structure.

곱셈이 비교적 많은 ijpeg등을 기준으로, 32×32bit Booth 인코딩 배열구조의 경우 상위 배열에서 연산되는 부분합의 개수가 5개 정도에서 최적화 된 속도를 내는 것을 실험을 통해 알 수 있었다. 이와 같은 경우에는 Wallace 트리와 비교하여도 거의 같은 단계의 연산을 거치며, CSA의 배열이 비교적 균일하므로 구현이 용이한 장점을 가진다. 그러나 일반적인 배열구조에 비해서는 라우팅이 조금 복잡해지고 마지막 덧셈기에서 더해주어야 하는 비트 수가 많아지며 이로 인하여 회로 면적이 약간 늘어나는 단점이 있다. 그림 6은 이 같은 비대칭 병렬배열구조를 이용한 곱셈기이다.

3. 연산 종료 신호

비동기 회로에 적용하기 위해 필요한 또 한가지는 연산종료신호이다. 비동기 회로는 클럭을 사용하지 않으므로 연산종료를 알아내기 위한 신호가 필요하며, 하위 배열에서 데이터 의존적인 연산시간 측정을 위하여 그림 7과 같은 회로를 구현하였다. 이 회로는 Booth 인코딩의 결과를 사용하여 부분합이 0인 부분이 어디까지인지 찾아내기 위한 회로이다. Zj 는 j번째 부분합이 0일 때 '0'이고, Pi는 최상위 부분합부터 i번째까지가 모두 0일때 '1'이 된다. 이 Pi 신호가 '1'이 되면 그

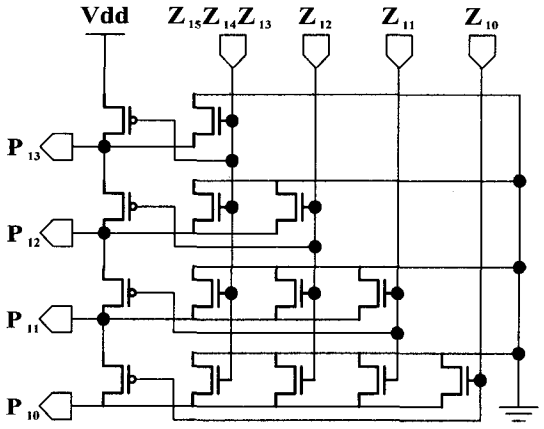

그림 7. 부호 비트 체크회로
Fig. 7. Leading Sign Bit Check Logic.

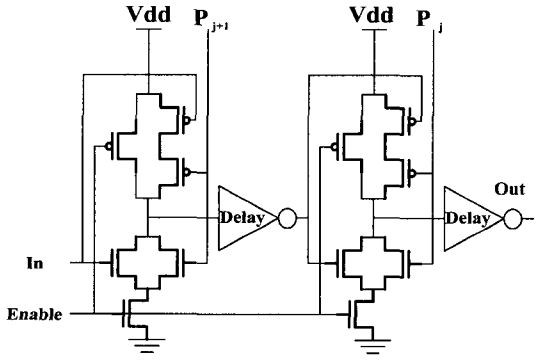

그림 8. 지연회로
Fig. 8. Delay Logic.

level의 CSA 셀들의 출력을 pull-down하여 즉시 연산결과를 0으로 만든다.

그림 8은 이 신호를 이용한 지연소자이며 2단계를 보여주고 있다. 각 단계의 Out 신호는 Enable 신호가 '0' 이면 항상 '0' 인 상태에 있으며, Enable 신호가 인가되는 순간, 그 단계의 Pi 신호가 '1'인 상태이면 바로 '1'로 변하고, 그렇지 않을 경우는 이전단계의 Out 신호가 '1'이되고 나서 '1'로 바뀌게 된다. 즉, 연산이 필요한 부분에서부터 지연신호를 발생시키게 된다. 이들은 데이터 의존적인 연산시간을 추출하기 위한 회로이며, Enable 신호는 Booth encoding이 끝나고 Pi 신호들이 안정화되고 난 후 인가되는 신호이다.

표 2. 곱셈 명령어의 개수
Table 2. Number of multiplications.

Architecture	benchmark program	Number of signed multiplication
SPARC	lisp	2 thousand
	gcc	240 thousand
	ijpeg	82 million
Simplescalar	lisp	15 thousand
	gcc	1 million
	perl	42 thousand
	go	13 million
	ksim	5 million

V. 실험 결과

CMOS 회로의 경우 전력소모의 요소는 몇가지가 있지만 그중에서 주된 요소는 Dynamic Power Consumption 이며 이는 회로에 사용된 Transistor의 크기와 그 Switching 회수에 비례한다. 표 2는 실험에 사용된 벤치마크 프로그램들의 곱셈명령의 개수이다. ijpeg의 경우는 곱셈 명령이 상당히 많아서 그중의 1/4정도만을 추출하여 실험하였다. 그림 9는 상·하위 배열에서 더해지는 부분합의 개수가 같은 Booth 인코딩 병렬배열구조의 에너지 소모를 비율로 나타낸 것이다. 기준인 기존의 Booth 인코딩 배열구조의 에너지 소모는 100으로 표시되어 있으며, 인코더나 최종 덧셈 단계는 포함되지 않은 값이다. Simplescalar의 gcc와 같은 경우는 다른 벤치마크와는 다르게, 승수의 길이가 긴 경우가 많아서 효과를 보지 못했다.

그림 10은 기존의 Booth 인코딩 배열구조의 에너지 소모와 연산시간을 100으로 놓았을 때, 제안된 Booth 인코딩 비대칭 병렬배열구조의 에너지 소모와 연산시간을 표시한 그래프이다. 데이터 의존적인 연산시간의 추출을 위한 회로에서 소모되는 에너지는 전체 Booth 인코딩 회로의 1~2%에 불과하며, 긴 승수를 갖지 않는 몇몇의 벤치마크 프로그램에서는 전혀 에너지를 소모하지 않기 때문에 전체 에너지 소모량에 미치는 영향은 거의 없다. 그래프를 보면 평균적으로 약 20%정도의 에너지 소모의 감소와, 55%정도의 연산시간의 감소를 보여준다.

그림 11은 비대칭 병렬배열구조를 이용한 비동기 곱셈기의 평균연산시간 및 에너지 소모량을 보여주고 있다. 기준은 역시 기존의 Booth 인코딩 배열구조 곱셈기이다. 0.35㎛ CMOS 공정을 사용하여 실험하였으며,

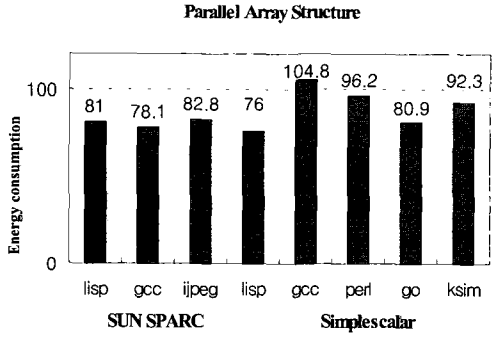

그림 9. 병렬배열구조의 에너지 소모
Fig. 9. Energy Consumption of Parallel Array Structure.

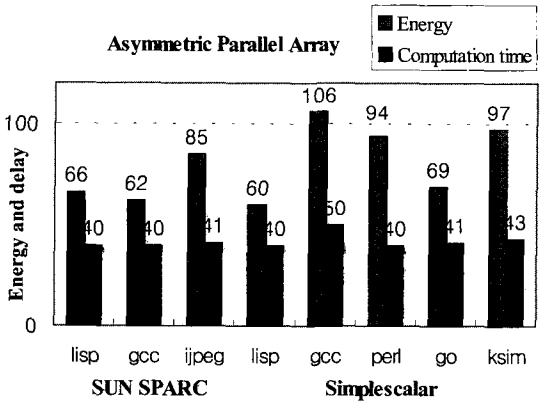

그림 10. 비대칭 병렬배열구조에서의 연산시간 및 에너지 소모
Fig. 10. Computation time and Energy consumption of Asymmetric Parallel Array Structure.

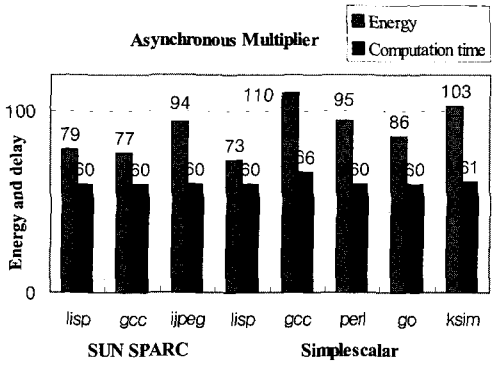

그림 11. 비대칭 병렬배열구조를 이용한 비동기 곱셈기의 연산시간 및 에너지 소모

Fig. 11. Computation time and Energy Consumption of Asynchronous Multiplier with Asymmetric Parallel Array Structure.

CSA 셀로는 "28T" Adder[12]가 사용되었다. 또한 Booth 인코딩은 One hot 인코딩 방법을 사용하였으며, CLA (Carry Look-ahead Adder)가 마지막 덧셈에 사용되었다. 실험결과 평균적으로 약 40%정도의 연산시간 단축 효과와 6%정도의 전력 감소 효과가 있음이 확인되었다.

이 평균연산시간의 감소는 이 곱셈기가 이상적인 비동기시스템에 사용되었을 경우에 볼 수 있는 성능의 향상이다. 만일 동기식 시스템에 사용된다고 하면 항상 worst-case의 속도로 동작시켜야 하므로 그다지 이득을 볼 수 없다. 이 경우에, 즉 worst-case의 경우에는 약 13%정도의 연산시간 감소를 보여준다.

VI. 결 론

이 논문에서는 비동기 구조에 적합하도록 상·하위 배열에서 계산되는 부분합의 개수가 다른 비대칭 곱셈기를 고안하였다. 먼저 곱셈기의 입력데이터 패턴을 조사하고, 그 패턴에서 발생하는 기존 배열구조의 문제점을 제시하였다. 이 문제점을 해결하기 위해 기존의 배열구조를 상·하 둘로 나누어 병렬적인 연산을 통해 속도를 높임과 동시에 필요없는 데이터의 전파를 막았다. 또한 상위 배열에는 기존의 방식을, 하위 배열에는 LR 방식을 사용한 Hybrid 방식을 채용하여 전력소비를 줄였다. 하위 배열에서 LR 방식의 사용은 전력소비 감소의 효과뿐만 아니라, leading sign bit이 많은 입력 패턴에서 빠른 연산을 가능케 하였다. 또한, 입력 패턴에 따른 평균 연산속도의 향상을 위하여 하위 배열의 연산단계를 상위 배열보다 길게 하고, 연산시간의 종료 검출을 위한 지연소자를 고안하였다. 그리고, Booth 인코딩을 사용하여 전체적인 속도를 향상시켰다.

실험결과 Booth 인코딩 비대칭 병렬배열구조는 기존의 Booth 인코딩 배열구조에 비해 에너지 소모는 약 20%정도, 연산시간은 약 55%정도 감소하였으며 이를 사용한 비동기 곱셈기는 평균 약 40%정도의 연산시간 단축효과가 있음을 보여주었다.

한편 Wallace 트리 구조는 6단계의 CSA 단계를 거치므로 평균연산시간이 CSA 6단계의 지연시간을 조금 넘는 비대칭 배열구조에 비하여 조금 더 빠른 연산속도를 가진다. 그러나 불규칙적인 구조로 인하여 더 긴 wire의 연결과 더 넓은 면적을 필요로 한다. 공정기술의 발전에 따라 회로의 전력소모와 연산시간에서 wire의 비중이 점차 커지고 있다. 따라서 정확한 비교는 두 곱셈기의 layout을 한 후 가능할 것으로 보인다.

참 고 문 헌

[1] C.S. Wallace, A suggestion for fast multiplication technique, IEEE Transaction on Electronic Computer, Vol. EC-13, pp. 14~17, Feb. 1964.

[2] I.E.Sutherland, Micropipelines, Communications of ACM, Vol. 32, No. 6, pp. 720~738, Jan. 1989.

[3] The standard Performance Evaluation Corporation. SPEC CPU95 Benchmarks. SPEC Newsletter, September 1995.

[4] D. Burger, T.M. Austin, and S. Bennett, Evaluating future microprocessors : The simplescalar tool set, Tech. Rep. CS-TR-96-1308, University of Wisconsin-Madison, July, 1996.

[5] Introduction to Shade, Sun Microsystem Laboratories, Inc. TR 415-960-1300, Revision A of 1/Apr/92.

[6] 김영수, 차영호, 조경연, 최혁환, Leading 0/1 검출 기능을 부가한 곱셈기, IEEK Summer Conference 2000 컴퓨터그룹, pp. 85~88, June 2000

[7] S. E. McQuillan, J. V. McCanny, A, VLSI

architecture for multiplication, division and square root, International Conference on Acoustics, Speech, and Signal Processing, pp. 1205~1208, 1991.

[8] A. D. Booth. A signed binary multiplication technique, Quarterly J. Mechanics, Appl. Math, Vol. 4, Part 2, pp. 236~240, 1951.

[9] K. Muhammad, D. Somasekhar and K. Roy, Switching characteristics of generalized array multiplier architecture and their application to low power design, International Conference on Computer Design (ICCD), 1999.

[10] David Kearney and Neil W. Bergmann, Bundled Data Asynchronous Multiplier with Data Dependent Computation times, Third International Symposium on Advanced Research in Asynchronous Circuits and Systems, pp. 186~197, April 1997.

[11] Israel Koren, Computer Arithmetic Algorithm, Prentice Hall International, 1993.

[12] Pascal C.H. Meier, Rob A. Rutenbar, L.Richard Carley, Inverse Polarity Techniques for High-Speed/Low-Power Multipliers, International Symposium on Low Power Electronics and Design, pp. 264~266, August 1999.

저 자 소 개

朴 贊 鎬(正會員)

1999년 서강대학교 전자공학과 학사. 2001년 광주과학기술원 정보통신공학과 석사. 2001년~현재 *ETRI* 컴퓨터시스템연구부 연구원. <주관심분야 : 비동기 VLSI 디자인, SOC 디자인 및 Network 시스템 설계 등>

崔 炳 秀(正會員)

1996년 충남대학교 컴퓨터공학과 학사. 1998년 광주과학기술원 정보통신공학과 석사. 1998년~현재 광주과학기술원 정보통신공학과 박사과정. <주관심분야 : 고성능 비동기 프로세서 설계, 명령어 수준 병렬성, 고성능 프로세서 구조 및 시스템 등>

李 東 翊(正會員)

1985년 영남대학교 전기공학과 학사. 1989년 일본 Osaka 대학 전자공학과 석사. 1993년 일본 Osaka 대학 전자공학과 박사. 1993년~1994년 University of Illinoise 객원 연구원. 1990년~1995년 일본 Osaka 대학 문부교관. 1995년~현재 광주과학기술원 정보통신공학과 조교수, 부교수. <주관심분야 : 비동기 시스템 CAD/설계, 자율 분산 시스템, 프로토콜 공학, 페트리넷 이론, 정형 기법, 병행시스템 설계 및 분석 등>