저전력 비동기 곱셈기를 위한 배열 구조

박찬호, 최병수, 이동익
광주과학기술원 정보통신공학과 병행시스템연구실
전화 : (062) 970-2248 / 팩스 : (062) 970-2204

Array Structure for Asynchronous Low Power Multiplier

Chan-Ho Park, Byung-Soo Choi, and Dong-Ik Lee
Concurrent Systems Research Lab., Dept. of Info. & Comm., K-JIST
E-mail : {chpark, bschoi, dilee}@csrl.kjist.ac.kr

Abstract

In this paper, a new parallel array structure for the asynchronous array multiplier is introduced. This structure is designed for a data dependent asynchronous multiplier to reduces power which is wasted in conventioanl array structure. Simulation shows that this structure saves 30% of power and 55% of computation time comparing to conventional booth encoded array multiplier.

1. 서론

곱셈기는 DSP나 마이크로 프로세서의 데이터 처리부에서 연산시간이 길고, 많은 전력을 소모하는 부분이다. 따라서 동기식 시스템에서는 파이프라인 형태로 만들거나, 복수클럭을 사용하여 연산을 하는 경우가 많으며, 성능에 있어서 병목으로 작용하기도 한다.

파이프라인 구조의 비동기 프로세서에서 데이터 의존적인 연산시간을 가진 단계가 존재하여도 그 단계에서의 가변적인 지연시간이 다른 단계의 고정연산시간보다 적으면 효과를 볼 수 없다. 따라서 데이터 의존적인 회로는 곱셈기처럼 전체 시스템에서 비교적 큰 지연시간을 차지하는 부분일수록 효과가 크다.

곱셈기에는 여러가지가 있으나, 대표적으로 Wallace 트리 형태, 배열구조 형태, 그리고 iterative 형태의 곱셈기가 있다. Wallace 트리 형태의 곱셈기는 연산속도가 빠르나 불규칙적인 구조로 구현이 어렵다는 단점을 가지고 있으며, 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 많은 단점을 가지고 있다.

이 논문에서는 기존의 배열구조의 단점을 보완하여 빠른 연산시간과 적은 전력소모를 가지는 비동기 곱셈기를 위한 구조를 제안하였다. 이 논문의 구성은 다음과 같다. 2장에서는 벤치마크 프로그램의 데이터 패턴을 분석하고, 기존의 배열구조가 가지는 문제점에 대하여 설명한다. 3장에서는 이를 해결하기 위한 배열구조를 제안하고 연산속도를 높이기 위하여 Booth 인코딩을 사용한 경우, 제안된 구조의 성능에 대하여 설명한다. 제안된 배열구조의 비동기 시스템의 적용에 대하여 4장에서 설명하고 있으며, 각각의 시뮬레이션 결과는 5장에 나타나 있다. 마지막으로 결론은 6장에서 기술한다.

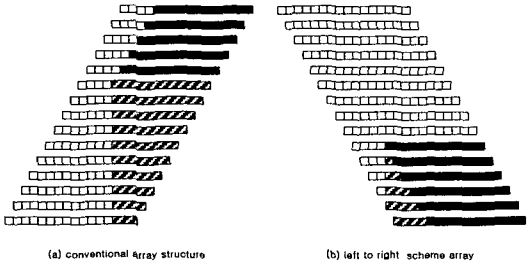
2. 기존 배열구조의 문제점

표 1은 SPEC INT95 프로그램을 SUN SPARC 와 Simplescalar architecture의 기반에서 추출한 입력 데이터의 평균 길이를 보여주고 있다. 이 표를 살펴보면 곱셈 명령 입력 데이터의 평균 길이가 약 7~8bit 정도로 길지 않음을 알 수 있다.

이러한 데이터 양상에서 기존의 배열구조를 사용하면 낭비되는 전력이 많다. 그림 1 (a)는 16bit 곱셈기에서 12×6bit의 곱셈이 일어날 때 사용되는 CSA(Carry

[표 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



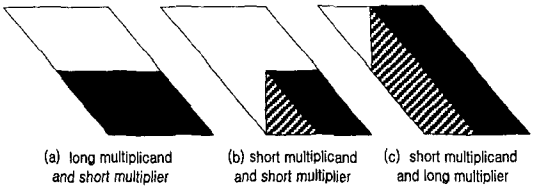
[그림 1] Power consumption in array structure

Save Adder) 셀들을 보여준 그림이다. 그림을 보면 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 하나이다.

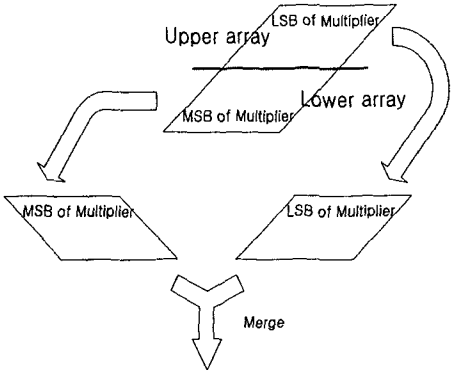
3. 병렬 배열구조

앞에서 말한 배열구조에서 낭비되는 전력은 상하를 나누어서 데이터의 전파를 막거나, 그림 1 (b)와 같이 LR(Left to Right) 방식을 사용하면 줄일 수 있다. 그림 2는 이 LR 방식의 입력패턴에 따른 특성을 보여주고 있다. 역시 실제 사용되는 셀은 검은색으로, 낭비되는 셀은 빗금으로 표시되어 있다. 그러나 이 LR 방식은 승수의 길이가 피승수의 길이보다 긴 경우에는 그림 2의 (c)와 같이 오히려 전력의 낭비가 커지며, 최종 덧셈 단계에서 더해주어야 하는 비트수가 늘어나는 단점이 있다.

두 번째의 해결책으로 배열의 상·하를 나누어 계산할 경우에는, 두 결과를 합하는 단계가 더 필요하지만 병렬적인 연산을 수행하여 최대 두 배의 연산속도



[그림 2] Properties of Left to Right Scheme



[그림 3] Parallel array structure

를 낼 수 있다. 전력 측면에서는 배열의 윗부분에서만 이루어지는 연산이 있다면 아래부분으로 전파되는 데이터가 없어 전력의 낭비를 줄일 수 있으나, 승수의 길이가 절반을 넘어갈 경우에는 데이터의 전파가 기존의 배열구조와 동일하고, 마지막 합산 단계의 추가적인 전력 소모가 필요한 단점이 있다.

지금까지 기술한 문제점을 해결하기 위하여 그림 3과 같이 상위 부분은 기존의 배열구조를 사용하고 하위부분은 LR 방식을 사용한 병렬구조의 곱셈기를 고안하였다. 이 구조는 배열을 분할하여 승수의 길이가 짧은 경우에는 하위 배열로의 데이터 전파를 막고, 승수의 길이가 긴 경우라도 하위 배열에 입력되는 승수의 길이는 줄어든 상태이므로 LR 방식을 사용하여 전력소모를 줄인 형태이다.

한편 제안된 구조에서는 빠른 연산을 위하여 흔히 사용되는 Booth 인코딩을 적용, 부분합의 숫자를 줄여 연산속도를 높였다. Booth 인코딩을 이용할 경우, 인코더의 결과에 따라서 피승수를 감산해야하는 경우가 있다. 이 경우 피승수의 2의 보수를 더해주게 되므로 결과적으로 데이터의 길이를 늘리는 효과를 가져와 그림 2의 (a)와 같은 연산에 가깝게 된다. 이러한 경우 일반적인 구조를 사용하면 그림 1의 (a) 처럼 전력을 많이 낭비하게 되며, 제안된 병렬구조의 곱셈기를 적용하면 낭비되는 전력 소모를 크게 줄일 수 있다.

4. 비동기 병렬 배열구조

제안된 곱셈기를 비동기 시스템에 적용할 때 중요한 점은 평균연산시간을 최소화하는 것이다. 데이터 의존적인 연산시간을 가지는 곱셈기는 이미 제안되었으나 [1], 여기서 제안된 구조에 적용하는 것은 쉽지 않다. 그 이유는 Booth 인코딩을 사용할 때, 인코딩 된 결과가 0이라 할지라도 부호확장을 위하여 앞의 2 비트는 1을 더해주어야 하기 때문이다[2]. 그러나 데이터 특성을 살펴보면 leading 0이 많으며, 하위 배열은 승수의 MSB부터 연산을 시작하기 때문에 leading 0이 끝날 때까지의 합은 0임을 쉽게 알 수 있다. 따라서 하위 배열은 상위 배열보다 훨씬 빠른 연산이 가능하다. 그러나 하위 배열의 연산이 끝나더라도 상위 배열의 연산이 끝나지 않으면 두 부분합을 더할 수는 없다. 따라서 상위 배열과 하위 배열의 연산시간의 밸런싱은 전체 평균연산시간을 높이는 중요한 요소이며, 상·하위 배열의 연산시간을 비슷하게 해 주기 위해서는 하위 배열의 길이를 더 길게 하여야 한다.

상위 배열에서도 기존의 방식[1]을 사용하여 데이터 의존적인 연산시간을 얻는 것이 가능하다. 즉 인코딩 된 결과가 0일 때 연산을 하지 않고 다음단계로 바로 데이터를 통과시켜 더 빠른 연산을 할 수 있다. 그러나 상위 배열에는 데이터들이 비교적 균일하게 분포해 있고, 따라서 인코딩 된 결과가 0일 확률은 25%이다. 또한 이를 위한 부가적인 회로와 그 전력소모가 있으므로, 상위 배열에서 데이터 의존적인 연산을 하는 것은 큰 효과를 얻을 수 없다. 따라서 제안된 구조에서 사용한 방식은 연산시간이 비교적 고정된 상위 배열은 짧게 놓고, 데이터에 따라 빠른연산이 가능한 하위 배열을 길게 놓아 평균연산시간을 상위 배열과 밸런싱하는 방법을 사용하였다.

곱셈이 비교적 많은 ijpeg, ksim등을 기준으로, 32×32bit Booth 인코딩 배열의 경우 상위 배열에서 연산되는 부분합의 개수가 5개 정도에서 최적화 된 속도를 내는 것을 시뮬레이션을 통해 알 수 있었다. 이와 같은 경우에는 Wallace 트리와 비교하여도 거의 같은 단계의 연산을 거치며, CSA의 배열이 비교적 균일하므로 구현이 용이한 장점을 가진다.

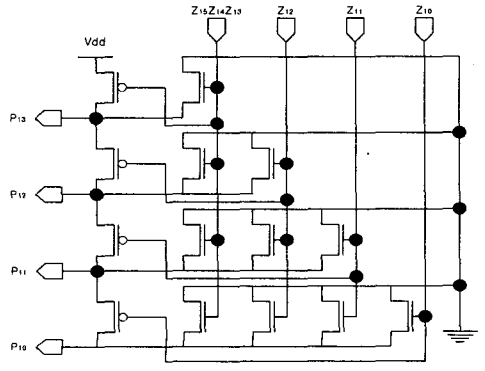
비동기 회로에 적용하기 위해 필요한 또 한가지는 연산종료신호이다. 비동기 회로는 클럭을 사용하지 않으므로 연산종료를 알아내기 위한 신호가 필요하며, 하위 배열에서 데이터 의존적인 연산시간 측정을 위하여 그림 4와 같은 회로를 구현하였다. *Zj* 는 *j*번째 부분합이 0일 때 0이고, *Pi* 는 최상위 부분합부터 *i*번째까지가 모두 0일때 1이 된다. 이 *Pi* 신호가 1이 되면 CSA 셀들을 pull down하여 즉시 연산결과를 만든다.

그림 5는 이 신호를 이용한 지연소자이다. *Enable* 신호가 인가되는 순간, *Pi* 신호가 1인 상태이면 바로 지연신호가 발생된다. 즉, 연산이 필요한 부분에서부터 지연신호를 발생시키게 된다.

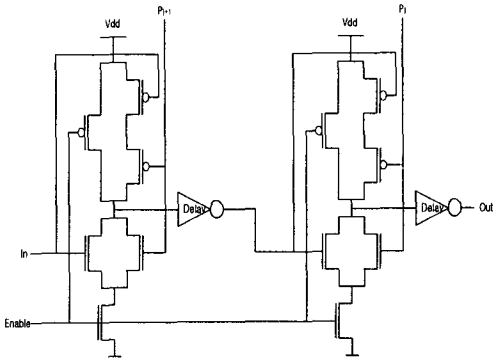
이들은 데이터 의존적인 연산시간을 추출하기 위한 부가적인 회로이며, 인코딩 시간 내에 *Pi* 신호들이 안정화되어야 한다. 그러나 그림 4의 회로에서 *Zj* 입력이 많아지면, 회로의 전달시간이 길어져, 인코딩 시간 내에 안정화되지 못하고, 지연소자에서 glitch가 생길 수 있다. 이러한 경우에는 인코딩이 끝났다고 하더라도, 신호들이 안정이 될 때까지 기다린 후, *Enable* 신호가 인가되어야 한다.

5. 실험결과

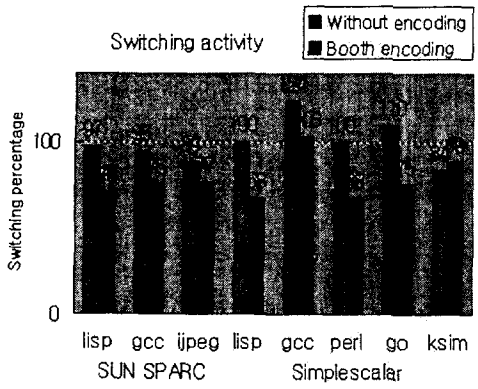
그림 6은 각 경우에 스위칭 횟수를 측정한 결과이다. 전력소비는 각 소자의 스위칭 횟수와 비례하기 때문에 스위칭 횟수를 전력소비의 기준으로 정하였으며, 인코더나 최종 덧셈 단계는 포함되지 않은 값이다. 이 그림에서 표시된 결과는 인코딩을 사용한 경우와, 사



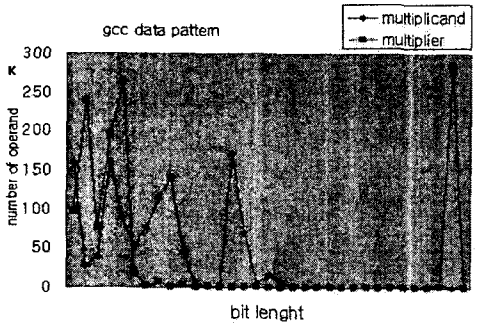
[그림 4] Additional encoding logic



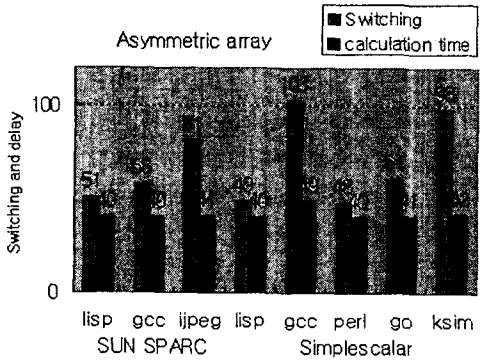
[그림 5] Delay element



[그림 6] Switching activity



[그림 7] Data pattern of gcc(benchmark)



[그림 8] Asymmetric array structure

용하지 않은 각각의 경우에 대하여, 기존 배열구조에서의 스위칭 횟수를 100으로 하였을 때, 상·하위를 절반으로 나눈 병렬 배열구조에서의 상대적인 스위칭 횟수를 표시한 것이다. 전반적으로 인코딩을 사용한 경우의 스위칭 감소효과가 큼을 알 수 있다.

Simplescalar 기반의 gcc와 같은 경우는 다른 벤치마크와는 다르게, 승수의 길이가 긴 경우가 많아서 효과를 보지 못했다. 그림 7은 이 벤치마크 프로그램의 입력 데이터의 패턴을 보여주고 있다. 즉 제안된 구조는 이처럼 승수의 길이가 긴 데이터가 많을 경우, 속도 측면에서만 이득을 볼 수 있을 뿐, 전력측면에서는 이득을 볼 수 없다.

그림 8은 Booth 인코딩 배열구조의 스위칭 횟수와 연산시간을 100으로 놓았을 때, 제안된 비동기 Booth 인코딩 병렬 배열구조의 스위칭 횟수와 연산시간을 표시한 그래프이다. 데이터 의존적인 연산시간의 추출을 위한 부가적인 인코딩 회로에서 소모되는 전력은 전체 Booth 인코딩 회로의 1~2%에 불과하며, 긴 승수가 없는 몇몇의 벤치마크 프로그램은 전혀 전력을 소모하지 않기 때문에 전체전력에 미치는 영향은 거의 없다. 그래프를 보면 평균적으로 약 30%정도의 스위칭 횟수의 감소와, 55%정도의 연산시간의 감소를 보여준다.

6. 결론

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 필요 없는 데이터의 전파를 막고, 상위 배열에는 기존의 방식을, 하위 배열에는 LR 방식을 사용하여 전력소비를 줄였다. 하위 배열에서 LR 방식의 사용은 전력소비 감소의 효과뿐만 아니라, leading 0이 많은 입력 패턴에서 매우 빠른 연산을 가능케 하였다. 또한, 입력 패턴에 따른 평균 연산속도의 향상을 위하여 하위 배열의 단계를 상위 배열보다 길게 하고, 연산시간의 종료검출을 위한 지연소자를 고안하였다. 그리고, Booth 인코딩을 사용하여 전체적인 속도를 향상시켰다.

시뮬레이션 결과 제안된 비대칭 Booth 인코딩 병렬 배열구조의 곱셈기는 기존의 Booth 인코딩 배열구조 곱셈기에 비해 스위칭의 횟수는 약 30%정도, 연산시간은 약 55%정도 감소하였다.

참고문헌

[1] 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.

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