시스템 온칩에서 스크래치 패드 메모리의 크기 탐색연구

조중석*, 조두산*, 김용주**
*순천대학교 전자공학과
**한국전자통신연구원
e-mail : dscho@scnu.ac.kr

A Study of Scratchpad memory size exploration of System-on-a Chip

Jungseok Cho*, Doosan Cho*, Yongjoo Kim**
*Dept. of Electronic Engineering, Sunchon National University
**ETRI

요 약

멀티미디어를 비롯한 많은 스트리밍 어플리케이션은 에너지 소비의 상당한 부분을 데이터 접근 연산 실행 명령어에 의해서 소비된다. 이러한 어플리케이션에서는 데이터 재사용성을 이용하여 에너지 소모량을 절감할 수 있다. 빈번히 사용되는 데이터를 고속의 상위 계층 메모리에 상주시켜 메인메모리 접근 횟수를 줄인다. 결과적으로 메모리 서브시스템에서 에너지 소모를 절감할 수 있게 된다. 본 연구에서는 어플리케이션의 재사용성을 분석하여 해당 어플리케이션에 특화된 스크래치패드 메모리 서브시스템 구성을 탐색하는 기법을 제안하고자 한다. 제안된 기법을 사용하면 하드웨어 제어 캐시 메모리와 비교하여 약 49% 에너지 소모를 절감하는 것이 가능하다.

1. 서론

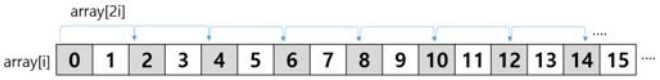
루프 지배적인 어플리케이션에서 데이터 재사용성을 이용하는 것은 에너지 효율 측면에서 매우 중요하다. 데이터 재사용성을 이용하는 전통적인 방식은 하드웨어 제어 캐시 (hardware controlled caches)를 사용하는 것이다. 범용 프로세서에서 주로 사용되는 하드웨어 제어 캐시는 하드웨어 제어 (controller) 로직이 차지하는 칩 면적, 이에 공급되는 전력량이 지수 증가한다는 단점을 지닌다. 또한 향후 데이터 접근 정보의 부족으로 인하여 캐시 메모리의 크기와 위치가 최적으로 설계되지 못하고 이에 따라 캐시 미스 비율이 높아질 수 있게 된다. 이러한 캐시 메모리의 단점을 대체하기 위하여 소프트웨어 제어 캐시 (software controlled cache)가 제안되었다. 소프트웨어 제어란 재사용되는 데이터를 컴파일러가 분석하여 캐시에 상주하도록 결정하는 것을 의미한다. 재사용 데이터를 메인 메모리에서 상위 메모리로 복사 전송하는 코드와 메인 메모리로 쓰기 명령하는 코드는 컴파일러에 의하여 자동 삽입된다. 컴파일러는 재사용 데이터 분석 기술을 적용하여 코드를 생성한다. 이러한 분석 기술은 어플리케이션에 특화된 소프트웨어 제어 캐시의 크기 탐색에도 적용하는 것이 가능하다. 본 연구에서는 이러한 데이터 재사용 분석 기술을 이용한 소프트웨어 제어 캐시 (스크래치 패드 메모리)의 크기 탐색 기법에 대하여 논의할 것이다.

2. 관련연구

많은 연구들이 캐시 메모리에서 데이터 재사용성에 대한 연구결과들을 발표하였다 [1][2]. 기존의 연구들은 주로 루프 코드의 변환을 통한 지역성 (locality) 개선에 초점을 맞추고 있다. 우리는 여기서 기존의 연구와는 다르게 어레이 변수들을 대상으로 스크래치패드 메모리를 효율적으로 설계 할 수 있는 크기 탐색 기법을 제시하고자 한다. [3]은 컴파일 시간 분석을 통하여 어레이를 분할하여 재사용 가능한 부분을 스크래치패드 메모리에 상주하도록 하는 기법을 제안하였다. 하지만 어플리케이션에 특화된 메모리 구조가 아니기 때문에 최적의 해결책은 아니다. 벨기에의 IMEC 연구소에서 스크래치패드 메모리 설계 탐색에 대한 연구와 동시에 재사용성에 대한 분석 연구를 진행하였다 [4]. [5]에서는 정적 위치 결정 기법이 아닌 재사용 데이터의 동적 위치 결정 기법을 제안하였다. 우리는 [4][5]의 연구를 어플리케이션에 적용해보았다. 이들은 복잡하여 실용적이지 못하다는 단점을 갖는다. 여기서는 이를 개선한 기법을 제안하고자 한다.

3. 메모리 액세스 기술자 기반 데이터 재사용 분석

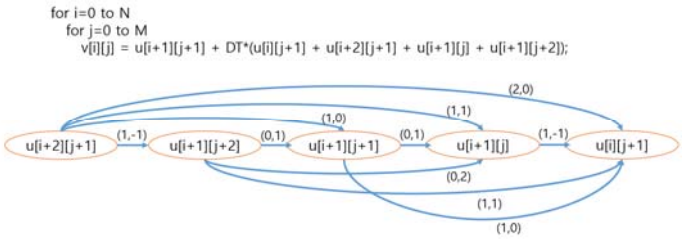
```
for …
    for (i=0; i<=50; i++)
        …=a[i] – a[2i]
```



(그림 1) 데이터 재사용 분석

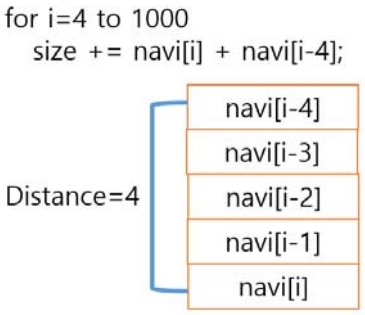
본 연구에서 데이터의 재사용 분석을 위하여 선형 메모리 액세스 기술자 (Linear Memory Access Descriptor, LMAD) [6]를 사용하였다. LMAD 를 데이터 재사용 분석 기술에 기본 자료구조로서 선택하여 사용한 이유는 메모리 액세스의 시간에 따른 발자취 (footprint)를 정확하고 단순하게 표현하여 분석을 용이하게 하기 때문이다. LMAD 는 메모리 액세스 패턴을 start, stride, span 세 가지 정보로 기술한다. 즉, start + [stride, span] 와 같이 표기한다. Start 는 특정한 어레이 변수에서 액세스를 시작하는 원점 '0'으로부터의 옵셋을 나타낸다. Stride 는 각 액세스들 사이의 간격을 나타내며, span 은 메모리 공간에서 처음 액세스와 마지막 액세스 사이의 거리를 나타낸다. 예를 들면, 그림 1 에 나타난 예제 코드의 액세스 a[2i]를 LMAD 로 표기하면 0 + [2, 50]이 된다. 재사용 정보는 코드 분석을 통하여 선형 메모리 기술자들 사이의 교집합을 통하여 계산된다. 그림 1 의 예제 코드의 경우 두 개의 선형 메모리 액세스 기술자인 {0 + [1,50], 0 + [2,50]}의 교집합을 통하여 재사용된 데이터 0 + [2,50]을 계산하게 된다. 재사용 LMAD 는 요구되는 스크래치패드 메모리의 크기 정보를 나타낸다. 이 예제의 경우 한 개의 워드가 4byte 일 경우 25*4=100byte 가 요구된다. 어플리케이션의 데이터 플로우를 따라 루프 코드 단위로 분석하며, 한 루프에 필요한 최대 용량이 해당 어플리케이션에 최적화된 스크래치 패드 메모리 크기로 결정된다.

4. 그래프 기반 데이터 재사용 분석



(그림 2) 데이터 재사용 그래프

데이터 재사용 그래프는 그림 2 와 같이 구성된다. 우선 루프 코드를 대상으로 분석하여 각각의 메모리 접근 코드가 그래프의 노드가 되는데, 그림 2 의 경우 메모리 접근이 어레이를 통해 이루어지기 때문에 각각의 어레이 접근 코드가 그래프의 노드가 된다. 각 노드와 노드사이에 관계를 에지가 나타낸다. 에지는 노드 사이의 재사용이 있을 때 재사용 사이의 거리를 루프반복(iteration) 사이의 거리로 나타낸다. 예를 들어, u[i+1][j+1]과 u[i][j+1]사이는 재사용 거리 (1,0)의 관계가 성립한다. 메인메모리에서 스크래치패드 메모리로 재사용 데이터의 복사는 재사용 이득을 얻을 수 있도록 데이터 전송 지연 시간을 고려하여 진행되어야 한다. 일반적으로 CPU 와 독립적으로 데이터 전송을 진행하는 것이 시스템 효율을 높이기 때문에 DMA (Dynamic Memory Access) 엔진을 사용하여 프리페치(prefetch)와 유사하게 데이터 전송이 이루어 진다. 재사용 그래프를 이용하면 해당 루프에서 재사용을 완전히 이용할 수 있도록 필요한 스크래치패드 메모리의 정확한 크기를 얻을 수 있게 된다.



(그림 3) 데이터 전송 크기 결정

스크래치패드 메모리로 한번에 전송하는 데이터의 크기가 스크래치패드 메모리 크기를 결정하는 주요 인자가 된다. 그림 3 에 예제를 나타내었다. 예제 코드를 보면 처음 사용된 navi[4]의 데이터는 4 번의 반복이 진행된 이후에 navi[i-4]에 의해서 재사용된다. 즉, 재사용 거리가 4 가된다. 따라서 루프 반복 4 번마다 데이터 4 개 단위로 전송하고 8 개 크기의 메모리를 가지고 있으면 재사용 데이터 모두 스크래치패드 메모리에서 접근되는 것이 가능하여 최대의 이득을 얻을 수 있게 된다. 아래 재사용 그래프의 에지를 따라 재사용을 최대로 하는 메모리 크기 결정 알고리즘을 나타내었다.

Reuse aware SPM size estimation algorithm

1. U: iteration space of loop nest l
2. G: reuse graph constructed for loop nest l
3. E: set of edges in G
4. S: minimum size of SPM storage

5. Set initial Scratchpad memory size to be sum of the total number of nodes' data sizes

6. While 1 do
7. Traverse all the untraversed edges in G and calculate their reuse distance
8. Update the largest reuse distance D
9. Mark edge (u,v) to traversed edge
10. If there are no untraversed edges in G then
11. Break;
12. Endif

13. End while
14. Multiply S and D
15. Update S with the multiplied result under SPM size constraint

알고리즘은 루프 U 에 대해서 재사용 그래프 G 를 구성하여 진행된다. G 는 U 가 완전 계층 루프(perfect nested loop)로 되어있는 경우에 한하여 구성된다. G 는 노드 집합 v 와 에지 집합 e 로 구성되어 있다. S 는 최종적으로 계산된 스크래치패드 메모리의 크기를 저장한다. S 의 초기값은 모든 노드들의 데이터 크기 합으로 결정한다. G 의 헤드 노드에서부터 시작하여 방문하지 않은 에지들을 따라 노드들을 차례로 방문한다. 방문할 때 노드들 사이의 재사용 거리의 최대치를 갱신하며 D 에 저장한다. 모든 노드들이 재사용 기간 동안 메모리에 상주할 수 있도록 재사용 거리와 상주 노드들의 데이터 크기의 곱으로 메모리의 크기를 결정한다. 더 이상 방문하지 않은 노드가 없을 때 알고리즘을 종료하고 S 를 최종 메모리의 크기로 결정한다.

5. 실험 결과

실험의 목표는 제안된 기법을 통하여 최적의 크기로 구성된 스크래치패드메모리와 같은 크기의 하드웨어 제어 캐시 메모리를 비교하여 에너지 소모비를 측정하는 것이다. 이러한 실험을 통하여 어플리케이션에 특화된 크기의 스크래치패드 메모리의 에너지 효율을 확인할 수 있을 것이다.

인텔 펜티엄 프로세서에 리눅스를 설치하여 실험을 진행하였으며, Simplescalar[7] 시뮬레이터를 사용하여 결과를 확인하였다. CACTI [8]에너지 모델을 사용하여 130nm 공정상에서 에너지 소모량을 측정하였다. 실험 대상 어플리케이션은 H.263 비디오 인코더, QSDPCM 인코더 이미지 처리용 라플라스 알고리즘에서 루프 커널을 추출하여 사용하였다. 앞서 제안된 재사용 분석기술을 적용하여 각 어플리케이션에 최적화된 스크래치패드메모리 크기를 할당하여 동일한 크기의 하드웨어 제어 캐시와 비교 실험을 진행하였다.

<표 1> 실험 결과

프로그램	메모리 크기	에너지 절감
H.263	36K	41%
QSDPCM	1K	49%
Laplace	4K	58%

스크래치패드메모리의 사용은 메인메모리 액세스 횟수를 감소시킴으로 에너지 절감을 얻게한다. 스크래치패드메모리 기반의 부메모리 시스템을 Directed mapped cache 타입의 하드웨어 제어 캐시 메모리와 비교하는 것은 소프트웨어 제어 메모리가 어플리케이션 코드를 정교한 분석을 통하여 최적의 데이터 위치 선정을 결정할 수 있는지를 판단하는 작업이다. 기본적으로 스크래치패드 메모리가 얻는 에너지 절감은 두 가지 원인으로 분석된다. 첫번째는 하드웨어 제어 캐시에 비하여 제어 로직 자체가 없는 소프트웨어 제어 캐시가 일회 액세스당 소모 전력이 절반으로 작다는 점이다. 두번째는 하드웨어 제어 캐시의 데이터 선정 정책인 LRU 와 같은 기법과 비교하여 컴파일러 재사용 분석을 통한 데이터 선정이 더 정확히 데이터의 재사용성을 획득하여 이용 가능하다는 점이다. 이러한 두가지 요인으로 인하여 소프트웨어 제어 캐시는 모든 실험에서 하드웨어 제어 캐시에 비하여 에너지 효율면에서 우위를 나타내었다.

6. 결론

본 논문에서 우리는 데이터의 재사용 분석을 통한 소프트웨어 제어 캐시 메모리 혹은 스크래치패드 메모리를 위한 효율적인 크기 탐색 기법을 제안하였다. 제안된 기법은 데이터의 재사용성을 정확히 분석하여 최적의 부메모리 크기정보를 제공하여 에너지 절감효율을 개선시킨다. 제안된 기법은 크게 두가지를 기반으로 구성되었다. 첫째, 메모리 접근 기술자 기반으로 각 메모리 접근 패턴을 요약하여 이를 기반으로 크기를 결정한다. 둘째, 재사용 그래프 기반으로 재사용 거리를 측정하여 모든 재사용을 활용할 수 있도록 메모리 크기를 결정한다.

7. 감사의 글

본 연구는 2010 년도 정부 (미래창조과학부)의 재원으로 한국연구재단 기초연구사업(2010-0024529)의 연구수행으로 인한 결과물임을 밝힙니다.

참고문헌

[1] D. F. Bacon, S. L. Graham et al. “Compiler Transformations for High Performance Computing.” ACM Computing Surv., 26(4), 1994.

[2] K. McKinley, S. Carr, and C.-W. Tseng. “Improving Data Locality with Loop Transformations.” ACM Trans. on Programming Languages and Systems, 18(4), July 1996.

[3] P. R. Panda, N. D. Dutt, and A. Nicolau. “Efficient Utilization of Scratch-Pad Memory in Embedded Processor Applications.” DATE, Paris, March 1997.

[4] J. Diguet, S. Wuytack, F. Catthoor, and H. De Man. “Formalized Methodology for Data Reuse Exploration in Hierarchical Memory Mappings.” In Proceedings of the IEEE International Symposium on Low Power Design, pages 30-35, Monterey, CA, August 1997.

[5] T. Van Achteren, F. Catthoor, R. Lauwereins, G. Deconinck. “Search Space Definition and Exploration for Nonuniform Data Reuse Opportunities in Data-Dominant Applications.” ACM Trans. On Design Automation of Electronic Systems, Vol. 8, No. 1, Jan. 2003.

[6] Yunheung Paek, Jay Hoeflinger, and David Padua, "Simplification of array access patterns for compiler optimizations", In PLDI’*98*, pages60–71.

[7] D. Burger and T. M. Austin. “The SimpleScalar tool set, version 2.0.” In Technical Report 1342, University of Wisconsin-Madison, CS Department, June 1997.

[8] P. Shivakumar, N. Jouppi. “CACTI 3.0: An Integrated Cache Timing, Power, and Area Model.” WRL Technical Report 2001/2, Aug. 2001.