멀티미디어 참조 예측을 고려한 임베디드 시스템의 성능 분석 모델

이춘희† 문현주‡ 유현배‡
† 충북대학교 전자계산학과
‡ 나사렛대학교 정보과학부

Analytical Performance Models for Embedded Systems Using Multimedia Reference Prediction

Chun-Hee Lee, Hyun-Ju Moon, Hyun-Bae Yoo
Department of Computer Science, Chungbuk National University
Division of Information and Science, Korea Nazarene University

요 약

최근 들어 개발 및 보급이 확산되고 있는 포터블 임베디드 시스템에서는 전력이 중요한 성능 지표로 작용한다. 특히 메모리 시스템의 전력 소모량은 전체 전력 중 매우 큰 비중을 차지하므로, 정확한 전력 분석에 기초하여 전력 소모를 줄일 수 있는 구조에 관한 연구가 수행되고 있다. 본 논문에서는 포터블 임베디드 시스템의 주요 처리 대상인 멀티미디어 응용 프로그램의 낮은 데이터 재사용성을 극복하기 위하여 참조 예측기를 포함하는 메모리 시스템의 전력 분석 모델을 제안한다.

1. 서론

최근 소규모 이동 가능한 포터블 임베디드 시스템들의 개발 및 보급이 확산되면서 이들 시스템에 대한 관심이 증대되고 있다. 포터블 임베디드 시스템은 수명이 한정된 베터리에 의존하게 되어 전력 사용에 상당한 제약을 받고 있다. 따라서 전력 사용의 제약을 해결하기 위한 많은 연구들이 수행되고 있다[1,2]. 특히 메모리 시스템의 전력 소모량이 전체 시스템 전력 소모의 대부분을 차지하고 있어 메모리 시스템의 상호작용에 대한 분석이 디지털 시스템의 에너지를 분석하는데 중심이 되고 있다. 이러한 관점에서 Wen등은 메모리 시스템의 전력 분석 모델을 제시한 바 있다[3,4].

한편 포터블 임베디드 시스템의 주요 처리 대상인 멀티미디어 데이터는 스트리밍 형태로 메모리를 참조하며 캐시에 의한 재사용성이 낮은 특성을 가지고 있다. 이러한 특성은 메모리 접근을 위한 CPU의 대기 시간을 증가시켜 응용 프로그램의 수행 성능을 저하시키는 원인이 되며, 이를 극복하기 위한 방법으로 데이터 선인출 기법이 사용되고 있다[5-7]. 데이터 선인출 기법은 참조된 메모리 주소를 바탕으로 다음에 참조될 데이터를 예측하고 미리 인출하여 메모리 참조에 대한 CPU의 대기시간을 단축시키고 캐시의 미스율을 감소시키는 방법이다.

포터블 임베디드 시스템이 데이터 선인출을 채용할 경우 선인출을 담당하는 참조예측기가 시스템에 포함되며, 전력 소모 측면에서 선인출로 인한 전력 소모의 영향을 분석할 필요가 있다. 따라서 본 논문에서는 기존의 전력 분석 모델을 기반으로 참조예측기가 포함된 포터블 임베디드 시스템의 전력 소모량 분석을 위한 새로운 에너지 모델을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 캐시를 채용한 메모리 시스템의 전력 분석 모델을 소개하고 3장에서는 데이터 선인출 기법과 참조 예측기의 구성 및 수행에 관하여 설명한다. 4장에서는 데이터 선인출을 채용하는 메모리 시스템의 전력 분석모델을 제시하고, 5장에서는 결론을 맺는다.

2. 메모리 시스템의 전력 분석 모델

메모리 시스템과 전체 시스템의 전력 소모량은 거의 완전 상관관계에 근접하고, 전력의 주 소모층인 메모리 시스템의 전력 소모는 캐시 히트율에 종속되어 있다. 메인 메모리와 캐시로 인한 기존의 전력 분석 모델[4]에서는 메모리 시스템의 전체 에너지를 그림 1과 같이 정의하였다.

그림 1에서 전체 에너지는 캐시 적중에 의한 에너지(Hit rate * Energy_hit)와 캐시 미스에 의한 에너지(Miss rate * Energy_miss)의 합으로 정의된다. 여기에서 Energy_hit은 디코더 과정에 필요한 에너지 소모량과 캐시에 의해 소모되는 에너지를 나타낸다. 또한 Energy_miss는 CPU에 의해 요청되는 데이터를 캐시에서 확인하는 과정은 Energy_hit와 같고, 캐시의 블록 크기를 데이터 버스에서 전송하는 워드 크기로 나눈 몫 만큼의 시행 회수로 메모리에서의 읽기 동작을 반복하는 작업(E_io)과 캐시의 write back 쓰기 정책에 의해 소모와 캐시 블록에 포함되는 워드 수만큼의 메인 메모리 액세스에 필요한 에너지(E_main)의 합으로 나타낸다.

$$Energy = Hit\ rate*Energy\_hit + Miss\ rate*Energy\_miss$$

$$Energy\_hit = E\_dec + E\_cell$$
$$Energy\_miss = E\_dec + E\_cell + E\_io + E\_main$$
$$= Energy\_hit + E\_io + E\_main$$

$$E\_dec = \alpha*(Add\_bs)$$
$$E\_cell = \beta*(Word\_line\_size)*(Bit\_line\_size)$$
$$E\_io = \gamma*(Data\_bs*Cache\_line\_size+Add\_bs)$$
$$E\_main = \gamma*(Data\_bs*Cache\_line\_size)+Em*Cache\_line\_size$$

그림 1. 메모리 시스템의 전력 분석 모델

또한 E_dec은 CPU가 요구하는 데이터를 캐시에서 찾기 위해 해당 주소를 디코드 하는 과정에 소모되는 전력을 나타내며 E_cell은 캐시 자체에서 소모되는 전력을 나타낸다. α,β,γ는 반도체 기본소자의 구현에 사용되는 CMOS 기술에 의해 변경되는 인수 값으로 0.8μm CMOS를 예로 들면, α=0.001, β=2, γ=20으로 사용할 수 있다. Word_line_size는 한 워드 라인상에 존재하는 메모리 셀들의 수를 Bit_line_size는 한 비트 라인상에 존재하는 메모리 셀들의 수를 나타낸다. Em은 한번의 메인 메모리 액세스에 의해 소모되는 전력을 나타낸다.

이와 같은 메모리 시스템 중심의 상호작용을 그림 2에 나타내었다. 그림 2에서 Add_bs는 주소 버스를 나타내고 Data_bs는 데이터 버스를 나타낸다.

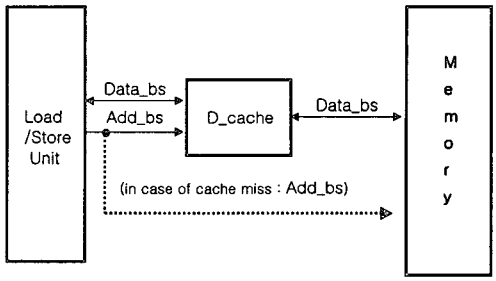

그림 2. 메모리 시스템 구조

3. 데이터 선인출

멀티미디어 응용 프로그램은 포터블 임베디드 시스템에서 매우 중요한 처리 대상이다. 이들은 스트리밍 패턴으로 데이터를 참조하는 특성이 있어 빈번한 메모리 액세스로 인한 전체 수행시간의 지연이 발생되고 캐시를 적절하게 활용하기 어렵다. 이러한 문제를 효율적으로 해결하기 위한 방법으로 스트리밍 패턴 데이터의 메모리 참조 특성을 활용하는 선인출 기법이 적합하다. 데이터 선인출은 프로세서가 앞서 인출된 데이터를 처리하는 동시에 미래에 사용될 데이터를 인출함으로써 응용 프로그램의 수행시간에 대한 메모리 참조 시간의 영향을 줄이는 것이다[7]. 이 기법은 선인출 명령을 발생시키는 시점을 기준으로 정적 선인출 기법과 동적 선인출 기법으로 구분된다. 정적 선인출 기법은 프로그래머나 컴파일러가 프로그램 구조를 분석하여 실행 시간 이전에 선인출 명령어를 프로그램에 삽입하는 기법이다. 이와 비교하여 동적 선인출 기법은 하드웨어가 실행 시간에 메모리 참조 명령어의 수행을 분석하여 이를 바탕으로 선인출할 메모리 주소를 계산한 후 선인출 명령을 발생시키는 것이다.

정적 선인출 기법은 컴파일 정보를 활용하여 데이터의 참조 패턴과 참조 시점을 정확하게 예측할 수 있는 반면 실행 처리기가 선인출 명령어를 수행하는 만큼 실행 사이클이 증가하는 문제점이 있다. 이와는 달리, 동적 선인출 기법은 복잡한 패턴의 데이터는 참조는 예측할 수 없으나 단순한 패턴에 대한 선인출에서는 우수한 성능을 발휘하며 실행 사이클의 증가를 유발하지 않아 멀티미디어 응용프로그램과 같이 단순한 규칙성에 의한 스트리밍 패턴의 데이터

참조에 적합하다.

동적 선인출 기법의 대표적인 예로는 OBL(One Block Lookahead) 기법[5], Stream Buffer 기법[6], RPT 기법[7]이 있다. 이들 중 멀티미디어 응용 프로그램의 수행에 가장 우수한 성능 향상을 나타내는 RPT 기법은 메모리 참조에 대한 과거 기록을 유지하면서 이로부터 참조의 규칙성을 찾아 미래에 참조될 메모리 주소를 예측하는 기법이다. 이를 위하여 참조예측표(Reference Prefetching Table)를 구성하고 아래와 같은 정보를 저장 및 갱신하면서 이들로부터 미래에 참조될 메모리 주소를 예측한다.

- Tag: 메모리 참조 명령어의 주소, PC값
- Prev_addr: 메모리 참조 명령어가 참조한 피연산자의 주소
- Stride: 메모리 참조 명령어가 반복적으로 수행될 때 연속적으로 참조된 피연산자 주소간의 간격
- State: 규칙성의 성립 상태(Init, Trasient, Steady)

본 논문에서는 RPT 기법을 채용하는 임베디드 시스템을 대상으로 메모리 시스템의 전력 분석 모델을 제시한다.

4. 선인출을 채용한 메모리 시스템의 전력 분석 모델

데이터 선인출을 채용하는 포터블 임베디드 시스템의 메모리 구조는 그림 3과 같다. CPU와 메모리 사이에 위치한 참조 예측기는 하드웨어로 구현되며, 참조예측표의 정보를 바탕으로 선인출 할 메모리 주소를 계산하고 선인출 명령을 메모리 제어기로 전달한다. 참조 예측표는 참조 예측기와 같이 별도의 하드웨어로 구현하거나 데이터 캐시의 일부를 이용하여 구현할 수 있다. 본 논문에서는 데이터 캐시의 일부를 이용하여 참조 예측표를 구현한 시스템을 가정한다.

참조예측기가 발생한 선인출 명령은 CPU에 의한 메모리 참조 명령과 마찬가지로 메모리 제어기에 의하여 처리된다. 즉, 선인출 할 주소의 데이터 블록이 이미 캐시에 적재된 경우에는 선인출 명령이 그대로 종료되며, 데이터 블록이 캐시에 존재하지 않는 경우에 메모리로부터 인출된다. 따라서 메모리 시스템의 전력 소모는 CPU의 데이터 참조 요구에 대한 캐시의 적중/미스율과 참조 예측기 수행의 결과에 대한

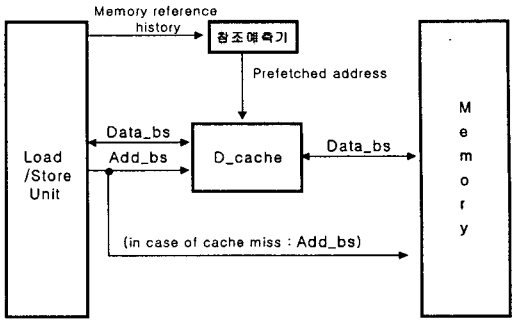

그림 3. 참조예측기가 포함된 메모리 시스템 구조

선인출 적중/미스율에 의하여 결정된다.

그림 4는 본 논문에서 제안하는 메모리 시스템의 전력 분석 모델을 나타낸다. 메모리 시스템에서 소요되는 총 전력은 메모리-캐시간의 데이터 교환에 소요되는 전력과 참조예측기가 선인출을 수행하는데 소요되는 전력의 합으로 표현하였다.

메모리-캐시간의 데이터 교환에 소요되는 전력을 분석하는 과정에서 메모리 참조 명령이 수행되는 총 횟수는 CPU에 의한 메모리 참조와 선인출에 의한 메모리 참조의 합이다. 따라서 메모리 참조 중 CPU에 의한 메모리 참조 명령의 캐시 적중률과 미스율을 각각 *C_Hit rate, C_Miss rate*로 나타내었으며, 선인출로 인한 메모리 참조 명령의 캐시 적중률과 미스율을 각각 *P_Hit rate, P_Miss rate*로 나타내었다. *Energy_hit*와 *Energy_miss*, *E_dec*, *E_cell*, *E_io*는 기존의 전력 분석 모델과 동일하게 적용하였다.

RP_Energy는 참조예측기가 선인출을 수행하는데

$$
\begin{aligned}
Energy &= \{(C\_Hit\ rate * Energy\_hit) + (C\_Miss\ rate * Energy\_miss)\} + \{(P\_Hit\ rate * Energy\_hit) \\
&\quad + (Prefetch\ Miss\ rate * Energy\_miss)\} + RP\_Energy \\
Energy\_hit &= E\_dec + E\_cell \\
Energy\_miss &= E\_dec + E\_cell + E\_io + E\_main \\
&= Energy\_hit + E\_io + E\_main \\
E\_dec &= \alpha * (Add\_bs) \\
E\_cell &= \beta * (Word\_line\_size) * (Bit\_line\_size) \\
E\_io &= \gamma * (Data\_bs * Cache\_line\_size + Add\_bs) \\
E\_main &= \gamma * (Data\_bs * Cache\_line\_size + Add\_bs) \\
&\quad + Em * Cache\_line\_size \\
\\
RP\_Energy &= E\_RPT\_dec + E\_RPT\_cell \\
E\_RPT\_dec &= 2 * E\_dec \\
E\_RPT\_cell &= \beta * (RPT\_Word\_line\_size) * (RPT\_Bit\_line\_size)
\end{aligned}
$$

그림 4. 선인출을 채용한 시스템의 전력 분석 모델

소요되는 전력을 의미한다. *E_RPT_cell*은 CPU가 메모리 참조 명령을 수행하면 참조예측기가 참조 예측표(RPT)를 확인하여 현재 수행된 메모리 참조 명령의 PC가 RPT에 존재하는가를 확인하는데 소요되는 전력을 의미한다. RPT는 데이터 캐시의 일부 또는 별도의 SRAM으로 구현되므로 캐시의 참조와 동일한 전력 모델을 적용하여 β *(RPT_Word_line_size)*(RPT_Bit_line_size) 으로 정의하였다. 여기서 RPT의 워드라인 크기와 비트라인 크기가 캐시와 다르므로 이들을 각각 RPT_Word_line_size와 RPT_Bit_line_size으로 나타내었다.

*E_RPT_dec*은 CPU가 메모리 참조 명령을 수행하면 참조 예측기가 참조 예측표(RPT)의 내용을 참조하여 선인출 주소를 계산하고 RPT를 갱신하기 위한 전력을 나타낸다. 즉, 해당 PC가 RPT에 존재할 경우 참조 예측기는 1) RPT로부터 Prev_addr을 읽어 현재 참조된 메모리 주소와의 관계를 확인한 후 선인출 할 주소를 계산하고 2) RPT의 Prev_addr를 현재 참조된 메모리 주소로 갱신한다. 이 과정에서 참조 예측기와 참조 예측표간에 2회의 주소 전송이 발생하므로 *2*E_dec*로 표현하였다.

본 논문에서 제시하는 캐시는 Set associative 매핑 함수를 사용하는 것으로 가정하였으며, 각 기능의 수행에 따른 구성요소들의 전력 소모 중 비중이 큰 구성요소들을 중심으로 분석하였다. 예를 들어, E_dec은 주소의 디코딩 과정에서 발생되는 전력은 디코딩 로직보다 주소 버스상에서 소모량이 크기 때문에 주소 버스에 의한 전력(E_dec)으로 단순화 하였으며 캐시의 태그 비교기, 주소 비교기등은 전력 소모량이 적어서 전력 분석에서 제외하였다.

5. 결론 및 향후 과제

본 논문에서는 스트리밍 패턴의 데이터 참조를 효율적으로 처리하는 참조예측기가 포함된 포터블 임베디드 시스템의 전력 문제를 해결하기 과정으로 기존의 디지털 시스템의 전력 분석 모델을 기반으로 참조예측기가 포함된 포터블 임베디드 시스템의 전력 분석 모델을 설계하였다. 향후 연구에서는 포터블 임베디드 시스템의 주요 처리 대상이 되는 멀티미디어 응용 프로그램들을 대상으로 기존의 메모리 시스템 구조에서 발생하는 전력 소모량과 참조예측기가 추가된 메모리 시스템에서 발생하는 전력 소모량을 비교·분석할 것이다. 또한 이를 바탕으로 포터블 임베디드 시스템의 전력 소모를 줄이면서 수행 성능을 향상시킬 수 있는 다양한 메모리 운용 기법을 연구할 것이다.

참고문헌

[1] N. Vijaykrishnan, M. Kandemir, M. Irwin, H. Kim and W. Ye, "Energy-Driven Integrated Hardware-Software Optimizations Using SimplePower". *Proc. of the Int. Symp. on Computer Architecture*, 2000.

[2] M. Kandemir, U. Sezer and V. Delauz, "Improving Memory Energy Using Accesses Pattern Classification," *Proc. of the IEEE/ACM Int. Conf. on Compiler Construction*, pp.276-292, April, 2002.

[3] M. B. Kamble and K. Ghose, "Analytical Energy Dissipation Models for Low Power Caches," *Proc. of the Int. Symp. on Low Power Electronics and Design, 1997.*

[4] W. T. Shiue and C. Charabarti, "Memory Exploration for Low Power, Embedded Systems," *Proc. of the IEEE/ACM Conf. on Design Automation*, pp.140-145, June 1999.

[5] A. Smith, "Sequential Program Prefetching in Memory Hierarchies," *IEEE Computer*, Vol. 11, No. 2, pp.7-21, 1978.

[6] F. Dahlgren, M. Dubois and P. Stenstrom, "Fixed and Adaptive Sequential Prefetching in Shared-memory Multiprocessors," *Proc. of the Int. Conf. on Parallel Processing*, pp.156-63, August, 1993.

[7] T. F. Chen and J. L. Baer, "Effective Hardware-Based Data Prefetching for High Performance Processors," *IEEE Transactions on Computers*, Vol. 44, No. 5, pp.609-623, May, 1995.