데이터 선인출을 채용한 임베디드 시스템의 성능 분석

문현주* · 유현배**

A Performance Analysis of Embedded Systems adapting Data Prefetching

Hyun Ju Moon* · Hyun Bae Yoo**

요 약

멀티미디어를 주요 처리 대상으로 하는 포터블 임베디드 시스템은 데이터에 대한 빈번한 메모리 접근으로 인하여 처리 속도가 저하되는 문제점에 직면하고 있으며, 이를 해결하기 위하여 임베디드 프로세서 설계 과정에서 데이터 선인출 기법을 채택하고 있다. 이 논문에서는 임베디드 시스템의 주요 성능 척도인 전력 소모 측면에서 데이터 선인출이 시스템의 성능에 미치는 영향을 분석하였다. 이를 위하여 데이터 선인출 기법이 추가된 메모리 시스템의 전력 분석 모델을 제안하고, 응용 프로그램 수행에 소모되는 전력을 모의 측정하였다. 실험 결과, 데이터 선인출은 응용 프로그램의 처리 시간을 단축하는 반면 전력 소모를 크게 증가시키는 것을 확인하였다. 더불어 데이터 선인출을 채용한 임베디드 시스템에 대하여 처리 속도와 전력 소모를 함께 고려하는 성능 분석 모델을 제안하였다.

ABSTRACT

Portable embedded systems which mainly handle multimedia applications involve the problem that frequent accesses to fetch data from memory make running time increased. To cope with the problem, embedded processors have adopted data prefetching schemes. From a power point of view, which is a main performance indicator of embedded systems, this paper analyzed to investigate how data prefetching schemes influence on system's performance. To solve the problem, we proposed a power-consumption analysis model of a memory system with data prefetching scheme and measured the power dissipated during running application programs. As a result, data prefetching schemes have application program's running time reduced but have system's power increased. Also we proposed a performance analysis model considering execution time and power consumption for embedded system with data prefetching schemes.

키워드

Embedded system, Embedded Processor, Data Prefetching, Energy Analysis Model, On-Chip Memory

Ⅰ. 서 론

최근 모바일 컴퓨팅 시스템은 이에 관한 관심이 증폭되면서 IT분야에서 그 시장성이 매우 커지고 있으며, 임베디드와 유비쿼터스의 확장으로 앞으로도 큰 증가세를 보일 것이라 예측된다. 일반 디지털 시스템과 비교해 볼 때, 모바일 컴퓨팅 시스템 등의 휴대용 기기들은 멀티미디어 응용 프로그램의 실시간 서비스를 위한 빠른 처리가 요구되며 배터리에 의존적인 전력 한계의 문제점을 갖고 있다[1-2]. 특히, 멀티미디어 처리에 따른 수행 시간의 지연이나 집중적인 전력 소모가 대용량의 데이터를 참조하는 과정에서 발생한다는 점을 주목할 만 하다. 즉, 멀티미

* 남서울대학교 컴퓨터학과
** 나사렛대학교 IT학부

접수일자 : 2005. 8. 31

디어 데이터는 스트리밍 패턴으로 사용되는 특성이 있어, 데이터의 재사용성을 활용하는 캐시의 효과를 극대화 할 수 없다. 그 결과로 빈번한 메모리 참조가 발생하며 이는 프로그램의 처리 시간 및 전력 소모를 증가시키는 원인이 된다.

Intel을 포함한 임베디드 프로세서 설계자들은 캐시의 적중률을 높여 이러한 문제를 해결하기 위한 방안의 일환으로 데이터 선인출 기법을 채용하고 있다[2-5]. 데이터 선인출은 미래에 참조될 메모리 주소를 예측하고 이를 참조 시점보다 앞서 캐시로 인출하는 기법이다. 그 결과, 캐시의 적중률을 높임으로써 메모리 참조를 위한 CPU의 대기 시간을 단축시키고자 하는 것이다. 이러한 측면에서 기존의 많은 연구 결과[6-9]들이 데이터 선인출로 인하여 수행 시간이 단축됨을 증명하였다.

이 논문에서는 기존 연구에서 간과되었던 데이터 선인출의 전력 소모에 대한 영향을 분석하고자 한다. 데이터 선인출로 인한 캐시의 적중률 향상은 메모리 참조를 위한 전력 소모를 줄이는 효과가 있는 반면, 미래에 참조될 메모리 주소를 예측하고 인출하는 과정에서 추가적인 전력을 소모하게 된다. 이를 분석함으로써 처리 속도를 향상시킬 뿐 만 아니라 전력 소모를 지양하는 임베디드 프로세서 설계의 토대를 마련하고자 한다.

전력 분석을 위한 첫 번째 과정으로 이 논문에서는 데이터 선인출을 채용한 캐시와 메모리 시스템의 상호작용에 대한 전력 분석 모델을 제안한다[2,3,10-12]. 또한 데이터 선인출로 인한 프로그램 처리 속도 향상과 전력 소모 증가의 영향을 동일하게 반영하는 프로세서의 성능 분석 모델을 제안한다. 이들을 이용하여 데이터 선인출로 인한 임베디드 시스템의 성능 변화를 분석한다.

이 논문의 구성은 다음과 같다. 제 2장에서는 대표적 데이터 선인출 기법인 RPT(Reference Prediction Table) 기법을 설명한다. 제 3장에서는 기존 메모리 시스템의 전력 분석 모델을 기반으로, RPT 기법을 채용하는 캐시 및 이와 상호 작용하는 메모리 시스템의 전력 분석 모델을 제안한다. 제 4장에서는 대표적인 멀티미디어 응용 프로그램을 대상으로 이들의 수행에 소모되는 전력과 처리 속도를 측정하기 위한 실험 및 분석 결과를 제시한다. 제 5장에서는 결론을 도출한다.

Ⅱ. 데이터 선인출

데이터 선인출은 앞서 인출된 데이터를 처리하는 동시에 미래에 사용될 데이터를 인출함으로써 응용 프로그램의 수행시간에 대한 메모리 참조 시간의 영향을 줄이는 것이다[8-9]. 이 기법은 선인출 명령을 발생시키는 시점을 기준으로 정적 선인출 기법과 동적 선인출 기법으로 구분된다. 정적 선인출 기법은 컴파일 정보를 활용하여 데이터 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 데이터 참조는 할 수 없으나 단순한 참조 패턴에 대한 선인출에서는 우수한 성능을 발휘하며 실행 사이클의 증가를 유발하지 않아 멀티미디어 응용 프로그램과 같이 단순한 규칙성에 의하여 스트리밍 패턴으로 데이터를 참조하는 경우에 적용하기 적합하다. 동적 선인출 기법의 대표적인 예로는 OBL(One Block Lookahead) 기법[7], Stream Buffer 기법[8], RPT 기법[9]이 있다. 이 논문에서는 이들 중 멀티미디어 응용 프로그램의 수행에 가장 우수한 성능을 나타내는 RPT 기법을 분석 대상으로 한다.

RPT 기법은 프로그램 내의 특정한 주소의 명령어가 반복해서 참조하는 메모리 주소를 기억하였다가 이 주소가 일정한 차이를 두고 변화하는 경우에 이를 예측하여 메모리 블록을 캐시로 적재하는 기법이다. 이를 위하여 참조 예측표(Reference Prediction Table)를 포함하는 참조 예측기를 메모리 시스템에 추가하고 다음과 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, Transient, Steady)

즉, 멀티미디어 데이터는 루프에 의하여 반복적으로 참조되는 특징을 지니고 있으므로 RPT 기법은 데이터를 참조하는 명령어마다 명령어의 적재 주소를 기억해두고 이 명령어가 데이터를 참조할 때마다 앞서 명령어가 실행될 때에 참조하였던 데이터 주소와 비교하여 두 데이터

주소의 차이, 즉 거리(stride)를 계산하고, 이 차이를 현재 참조하려는 데이터의 주소에 더하여 앞으로 참조할 것으로 예상되는 데이터의 선인출 주소를 계산한다.

예를 들어, 그림 1의 (a)는 C 언어로 작성된 프로그램 모듈이며, 그림 1(b)는 그림 1(a)의 프로그램 4~5번 문장의 내부 루프 반복문을 행 우선으로 수행하는 어셈블리 프로그램 모듈이다. 그림 1의 (a)~(b)는 내부 반복문의 곱셈 연산에서 일정한 간격에 의해 적재되어 있는 데이터를 참조하는 것을 보여준다. 그림 2(a)~(d)는 RPT 기법에 근거하여 그림 1(b)의 500, 504, 512에 해당하는 적재 명령문이 루프에 반복적으로 수행되는 과정이다. 그림 2(a)는 가장 안쪽의 내부 루프의 반복문을 수행하기 전에 RPT 참조 예측표의 상태를 나타내고 그림 2(b)는 내부 반복을 처음 시행할 경우 프로그램 카운터가 처음에 500, 504, 512 주소를 참조하는 경우의 stride와 state를 나타낸다. 다음 반복문에서 프로그램 카운터가 같은 주소(예:500, 504, 512)를 참조하는 경우 stride는 시스템 메모리에서 배열에 선

```
1. int A[100,100],B[100,100],C[100,100]
2. for i=1 to 100
3.   for j=1 to 100
4.     for k=1 to 100
5.       A[i,j]+=B[i,k]×C[k,j]
```

(a) 행렬 곱셈 프로그램

```
주소    명령어                  비 고
500   lw  r4, 0(r2)       load B[i,k]  stride 4 B
504   lw  r5, 0(r3)       load C[k,j]  stride 400 B
508   mul r6,r5,r4        B[i,k]×C[k,j]
512   lw  r7,0(r1)        load A[i,j]  stride 0
516   addu r7,r7,r6       +=
520   sw  r7,0(r1)        store A[i,j] stride 0
524   addu r2,r2,4        ref B[i,j]
528   addu r3,r3,400      ref C[k,j]
532   addu r11,r11,1      increase k
536   bne r11,r13,500     loop
```

(b) 어셈블리 코드

그림 1. 행렬 곱셈의 예
Fig. 1 An Example of Matrix Multiplication

tag	prev_addr	stride	state

(a) 초기 상태

tag	prev_addr	stride	state
500	50,000	0	init
504	90,000	0	init
512	10,000	0	init

(b) 반복 수행 (1'st)

tag	prev_addr	stride	state
500	50,004	4	transient
504	90,400	400	transient
512	10,000	0	steady

(c) 반복 수행(2'nd)

tag	prev_addr	stride	state
500	50,008	4	steady
504	90,800	400	steady
512	10,000	0	steady

(d) 반복 수행(3'rd)

그림 2. RPT 수행 과정
Fig. 2 RPT Processing

언된 데이터 형태의 크기만큼의 간격을 stride 값으로 정하고 state는 transient 상태가 되고, 그 다음 반복에서 프로그램 카운터가 같은 메모리 주소의 명령어를 참조하면서 상태가 transient이면 steady 상태로 전환 된다. 따라서 그림 2(a)~(d)의 경우는 정수형 데이터의 크기를 4byte로 정의하였을 경우 가장 내부 루프가 한 번 반복되면 배열 B는 배열 인덱스 첨자 k가 하나씩 증가하면서 다음에 수행되는 반복문의 명령어의 주소도 동일한 주소가 되면 정수형 데이터 크기 간격만큼 증가된 메모리 주소의 데이터를 참조하게 된다. 즉 배열 B는 stride가 4가 되고, 배열 C의 경우는 가장 내부 인덱스 첨가 k가 하나씩 증가할 때마다 행 우선으로 수행되는 행렬이므로 다음 반복 수행에서는 참조되는 주소가 열의 전체 크기(100)에 정수형 데이터의 크기(4)를 곱한 값이 참조하고자 하는 데이터의 주소가 된다. 즉, 배열 C의 stride는 400이 되고, 배열 A의 경우는 가장 내부 루프의 반복문 수행에서 참조되는 주소는 동일한 주소를 이용하므로 stride는 0이 되는 것이다. 이러한 RPT 기법은 일정한 간격으로 메모리를 참조하는 스트리밍 데이터의 처리에 매우 유용하다.

Ⅲ. 메모리 시스템의 전력 분석 모델

3.1 시스템 구조

시스템의 전력 소모량은 온-칩 캐시와 오프-칩 메모리로 구성되는 메모리 시스템의 전력 소모량에 매우 의존적이다. 따라서 시스템 설계 과정에서 메모리 시스템의

전력 소모에 대한 면밀한 분석이 필요하다. 이를 위하여 메모리 시스템의 각 구성 요소에서 소모되는 전력을 측정하는 시뮬레이터가 개발 된 바 있으나 이것의 적용 범위가 하드웨어적 메모리 개발 방법으로 제한되는 문제점이 있다. 이에 대한 대안으로 메모리 시스템에 관한 다양한 소프트웨어적 접근을 평가할 수 있는 전력 분석 모델이 연구되었다[10-14]. 메모리 시스템의 전력 소모는 캐시 적중률에 종속적이며, Wen[9] 등은 캐시 적중률과 미스율을 고려하는 임베디드 시스템의 전력 분석 모델을 제안하였다. 이 논문에서는 Wen의 모델을 바탕으로 RPT 기법을 채용하는 임베디드 시스템의 메모리 시스템 전력 분석 모델을 제안한다.

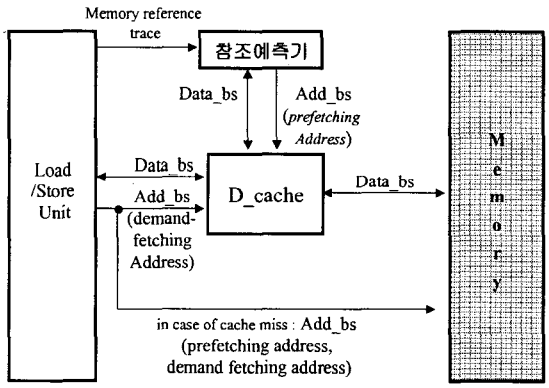

그림 3. 참조 예측기가 포함된 메모리 시스템 구조
Fig. 3 A memory system including RPT

RPT 기법을 채용하는 메모리 구조는 그림 3과 같다. CPU와 메모리 사이에 위치한 참조 예측기는 하드웨어로 구현되며, 참조 예측표의 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.

참조 예측기가 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.

3.2. 전력 분석 모델

그림 4는 이 논문에서 제안하는 메모리 시스템의 전력 분석 모델이다. 메모리 시스템의 전력 소모는 CPU의 데이터 참조 요구에 대한 캐시의 적중/미스율과 참조 예측 수행의 결과에 대한 선인출 적중/미스율에 의하여 결정된다. 즉, 메모리 시스템에서 소요되는 총 전력 소모량은 메모리-캐시간의 데이터 교환에 소요되는 전력 소모량(Energy_Demand)과 참조 예측기가 선인출을 수행하는데 소요되는 전력 소모량(Energy_Prefetch)의 합으로 나타낼 수 있다. 분석 모델에서는 메모리 참조 중 CPU에 의한 메모리 참조 명령의 캐시 적중률과 미스율을 각각 C_Hit rate, C_Miss rate로 나타내었으며, 선인출로 인한 메모리 참조 명령의 캐시 적중률과 미스율을 각각 P_Hit rate, P_Miss rate로 나타내었다. Energy_hit은 메모리 참조가 캐시에서 적중된 경우의 전력 소모량으로서, CPU가 요구하는 데이터를 캐시에서 찾기 위해 해당 주소를 디코

$$Energy = Energy\_Demand + Energy\_Prefetch$$
$$Energy\_Demand = (C\_Hit\ rate * Energy\_hit) + (C\_Miss\ rate * Energy\_miss)$$
$$Energy\_Prefetch = (P\_Hit\ rate * Energy\_hit) + (P\_miss\ rate * Energy\_miss) + RP\_Energy$$

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

$$RP\_Energy = 2(E\_dec + E\_cell)$$

Add_bs : 명령어 당 주소 버스상의 비트 스위치 수
Data_bs : 명령어 당 데이터 버스상의 비트 스위치 수
Word_line_size : 한 워드 라인의 메모리 셀 수
Bit_line_size : 한 비트 라인의 메모리 셀 수
Em : 주 메모리 참조의 전력 소모량
α, β, γ : *CMOS 기술에 의해 변경되는 인수*

그림 4. 선인출을 채용한 메모리 시스템의 전력 분석 모델
Fig. 4 Power-consumption analysis Model for a memory system including RPT

드 하는 과정에서 필요한 전력 소모량(E_dec)과 캐시 참조시 소모되는 전력량(E_cell)을 나타낸다. 또한 Energy_miss는 캐시 미스가 발생한 경우의 전력 소모량으로서, CPU에 의해 요청되는 데이터를 캐시에서 확인하는 과정은 Energy_hit과 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 정책 수행에 요구되는 전력 소모량과 캐시 블록에 포함되는 워드 수만큼의 주 메모리 접근에 필요한 전력 소모량(E_main)의 합으로 나타낸다. 이 때, Em은 한 번의 주 메모리 접근시 발생하는 전력 소모량이다. α, β, γ는 반도체 기본소자의 구현에 사용되는 CMOS 기술에 의해 변경되는 인수 값으로 0.8μm CMOS를 예로 들면, $\alpha = 0.001$, $\beta = 2$, $\gamma = 20$ 이다.

RP_Energy는 CPU가 메모리 참조 명령을 수행한 경우에 참조 예측기가 참조 예측표의 기록을 사용하여 선인출 여부를 판단하는 과정에서 필요한 전력 소모량을 의미한다. CPU가 메모리 참조 명령을 수행하면, 참조 예측기는 그 때의 프로그램 카운터(PC) 값을 이용하여 참조 예측표에 해당 기록이 있는가를 검색한다. 이 때, 참조 예측표는 데이터 캐시의 일부로 구현되었으므로 캐시 적중에 의하여 데이터를 참조하는 것과 동일한 전력을 소모한다. 또한 참조 예측기는 참조 예측표에서 검색한 기록을 갱신하고 그 결과를 참조 예측표에 재기록 해야 하므로 캐시 참조를 한 번 더 수행한다. 따라서 참조 예측기에 의한 전력 소모량 RP_Energy는 2(E_dec+E_cell)으로 정의하였다.

이 논문에서는 전력 소모량의 비중이 큰 구성요소들을 중심으로 분석 모델을 정의하였다. 예를 들어, 주소의 디코딩 과정에서 발생되는 전력 소모량 E_dec의 경우, 디코딩 로직보다 주소 버스상에서 소모량이 크기 때문에 주소 버스에 의한 전력 소모량으로 단순화 하였으며 캐시의 태그 비교기, 주소 비교기 등은 상대적으로 전력 소모량이 적은 이유로 전력 분석에서 제외하였다.

Ⅳ. 메모리 시스템 성능 분석

4.1 실험 환경

데이터 선인출이 임베디드 시스템의 성능에 미치는 영향을 분석하기 위하여 데이터 선인출 기능을 추가한 캐시 시뮬레이터를 구현하고 벤치마크 프로그램의 수행 결과를 측정하였다. 캐시 시뮬레이터는 트레이스 구동형으로 구현하였으며, 실험은 벤치마크 프로그램의 명령어 트레이스를 생성하는 과정과 데이터 캐시의 동작을 시뮬레이션 하는 과정으로 구성된다.

응용 프로그램의 명령어 트레이스를 생성하기 위하여 Alpha CPU용 목적 코드를 이용하여 벤치마크 프로그램을 분석할 수 있도록 개발한 ATOM 시뮬레이터[16]를 사용하였다. ATOM 시뮬레이터가 생성하는 명령어 트레이스는 연산, 분기 및 메모리 참조 명령어가 혼합된 형태로 생성된다. 연산 및 분기 명령어는 태그와 명령어 종류를 기록하고, 메모리 참조 명령어는 태그, 명령어 종류, 참조한 데이터 메모리의 유효 주소를 기록하게 하였다. 메모리 참조 명령어는 캐시의 수행 과정을 모의 실험하는 목적으로 사용되었으며, 연산 및 분기 명령어는 메모리 참조로 인한 지연시간을 측정하기 위한 목적으로 사용되었다.

캐시 시뮬레이터는 위스콘신 대학에서 개발한 Dinero III[15]를 기반으로 RPT 선인출 기법을 추가하였다. 이는 명령어 트레이스를 입력으로 하여 load/store 명령어별 메모리 참조 횟수와 캐시 적중 횟수, 적중률, 선인출 명령에 대한 캐시 적중 횟수, 적중률과 메모리 참조 지연 시간 등을 산출한다. 또한 캐시 크기, 라인 크기, 연관도, 블록 교체 정책 등은 수행시 매개 변수로 지정할 수 있다.

실험 대상은 멀티미디어 처리의 대표적 프로그램인 MPEG과 JPEG을 사용하였다. 각 프로그램은 압축 모듈과 해제 모듈로 구성되며 각 워크 그룹이 제공하는 입력 데이터 중 각각 크기와 특징이 서로 다른 5개의 입력 데이터를 선정하여 실험하였다. 입력 데이터 및 프로그램의 특징에 관하여 표 1에 나타내었다.

표 1. 벤치마크 정보
Table. 1 Experimental Parameters

	Description	Program module	Input/Frame size
MPEG	디지털 비디오 스트림 표준 변환 도구	mpeg2enc	352×240 4~8 frames
		mpeg2dec	
JPEG	흑백 및 칼라 이미지를 위한 표준 압축 방식	cjpeg	660Kbytes ~6Mbytes
		djpeg	

4.2 실험 결과

임베디드 시스템의 성능은 일반적으로 전력 소모량과 처리 시간의 함수로 표현된다. 따라서 시스템의 성능 분석을 위하여 응용 프로그램의 처리 시간을 측정하였으며, 캐시 시뮬레이션의 수행 결과 얻어진 캐시 적중률 및 선인출 명령에 대한 캐시 적중률을 앞서 제안한 전력 분석 모델에 적용하여 전력 소모량을 계산하였다. 캐시 블록 크기와 블록 교체 정책은 각각 32바이트와 LRU 방식을 채택하였으며, 캐시 미스가 발생한 경우의 메모리 참조 지연을 25사이클로 가정하였다. 연산 및 분기 명령의 수행 사이클은 덧셈, 뺄셈 및 분기 연산은 1사이클, 곱하기 4사이클, 나누기 12 사이클로 가정하였다.

그림 5는 데이터 선인출에 의한 처리 시간의 변화를 나타낸다. 각 그림에서 NP는 선인출을 수행하지 않은 경우를 나타내며, P는 선인출을 수행한 경우의 처리 시간을 나타낸다. 또한 s1~s5는 서로 다른 입력 데이터를 나타내며, y축은 수행 사이클이다. 그림 5의 결과는 데이터 선인출을 적용함으로써 각 벤치마크의 처리 시간이 매우 미세한 폭으로 단축됨을 나타낸다. 이들 중 그림 5(b)에 나타난 djpeg의 수행 결과가 비교적 뚜렷한 처리 시간의 단축을 보여 준다.

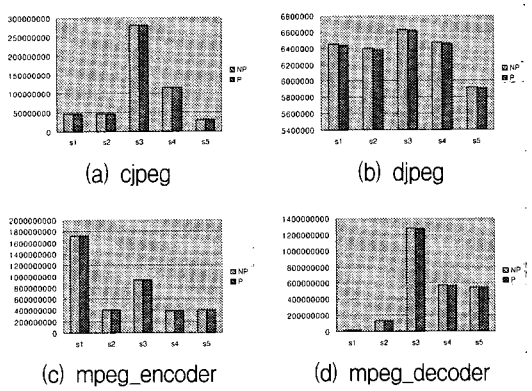

그림 5. 데이터 선인출에 의한 처리 시간 변화
Fig. 5 Variation of execution time

그림 6은 데이터 선인출에 의한 전력 소모량의 변화를 나타낸다. 각 그림에서 NP와 P는 각각 선인출을 수행하지 않은 경우와 선인출을 수행한 경우의 처리 시간을 나타낸다. 또한 s1~s5는 서로 다른 입력 데이터를 나타내며, y축은 전력 소모량(nJ)이다. 그림 6의 결과는 데이터 선인출을 적용함으로써 각 벤치마크 수행시 전력 소모량이 매우 큰 폭으로 증가함을 나타낸다.

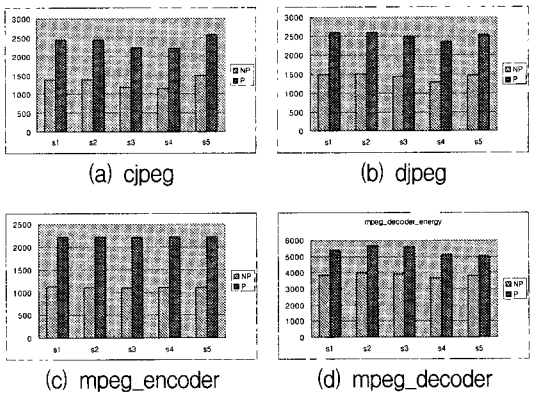

그림 6. 데이터 선인출로 인한 전력 소모량 변화
Fig. 6 Variation of power consumption

4.3 고찰

데이터 선인출 기법은 멀티미디어 데이터의 낮은 재사용성을 극복하고 메모리 참조를 위한 지연 시간을 단축하기 위하여 임베디드 프로세서 설계시 채택되고 있다. 앞서 제시한 실험 결과에서 데이터 선인출이 메모리 참조를 위한 지연 시간을 단축하는 것을 확인하였으나 그 효과가 매우 적은 이유는 실험 대상으로 사용한 응용 프로그램이 입력 데이터를 원본 형태대로 참조하지 않는 특성을 지녔기 때문으로 판단된다. 반면, 전력 소모의 측면에서는 평균 41.5%의 뚜렷한 전력 소모량 증가를 나타냈다. 이는 CPU가 처리하는 메모리 참조 명령 외에 선인출에 의한 메모리 참조가 추가적으로 발생하며, CPU가 연산 및 분기 명령을 처리하는 동시에 수행되므로 수행 시간에 대한 영향은 없으나 메모리 시스템을 동작시키기 위한 전력 소모는 메모리 참조 횟수에 비례적으로 증가하기 때문인 것으로 분석할 수 있다.

따라서 데이터 선인출이 궁극적으로 임베디드 시스템의 성능을 향상시키기 위해서는 처리 속도를 높일 뿐 아니라 전력 소모의 증가를 최대한 억제해야 한다. 일반적으로 임베디드 시스템 성능에 대한 전력 소모와 처리 시간의 영향이 동일한 것으로 간주할 때, 전체 시스템의 성능은 식 (1)과 같이 전력 소모량(E)과 처리 시간(T)의 곱에 반비례하는 것으로 나타낼 수 있다.

$$P = \frac{1}{E \cdot T} \quad (1)$$

식 (1)에 의하면, RPT 기법에 의한 데이터 선인출은 임베디드 시스템의 성능을 오히려 저하시키는 것으로 판단할 수 있다. 데이터 선인출을 채택하지 않은 기존의 메모리 시스템을 A라 하고 참조 예측기가 포함되는 메모리 시스템을 B라 할 때, 이들 간의 전력 소모량(E)과 처리 시간(T)의 관계는 각각 식 (2), 식 (3)과 같다.

$$\text{소모 전력}(E) \ : \ E_A < E_B \quad (2)$$

$$\text{처리 시간}(T) \ : \ T_B < T_A \quad (3)$$

여기서, E_A와 T_A는 각각 시스템 A의 전력 소모량과 처리 시간을 나타낸다. 식 (2)와 식 (3)으로부터 전력 소모 측면에서는 시스템 A의 성능이 우수하며 처리 시간 측면에서는 시스템 B가 적합함을 알 수 있다.

한편, 각 메모리 운용 기법에 의한 시스템 성능은 식 (4) 및 식(5)와 같다

$$P_A = \frac{1}{E_A \times T_A} \quad (4)$$

$$P_B = \frac{1}{E_B \times T_B} \quad (5)$$

식 (4)와 식 (5)에서 두 시스템은 전력과 처리 시간 측면에서 상충 관계에 있으므로 선인출을 채용한 시스템 B가 기존 시스템 A에 비하여 항상 우수한 것으로 판단할 수 없다. 즉, 처리 시간이 단축되는 효과보다 전력 소모의 증가율이 더 클 경우에는 전체 시스템 성능에 역효과를 수반할 수 있다.

그러므로 데이터 선인출이 임베디드 시스템의 전체적인 성능 향상에 기여하기 위해서는 선인출로 인한 전력 소모량의 변화율(ϕ_E)과 처리 시간의 변화율(ϕ_T)이 식 (8)과 같은 조건을 만족해야 한다.

$$\text{전력 소모량 변화율} \ : \ \phi_E = \frac{E_B}{E_A} \quad (6)$$

$$\text{처리시간 변화율} \ : \ \phi_T = \frac{T_A}{T_B} \quad (7)$$

전력 소모량 변화율 & 처리시간 변화율 :

$$\phi_E < \phi_T \quad (8)$$

V. 결 론

이 논문에서는 전력 소모량과 처리 시간을 성능 지표로 하여 데이터 선인출을 채용한 임베디드 시스템의 성능을 분석하였다. 이를 위하여 데이터 선인출을 채용한 임베디드 시스템의 전력 분석 모델을 제안하였다. 제안된 전력 분석 모델은 메모리 계층을 중심으로 전력 소모량을 측정하기 위하여 설계된 기존 분석 모델에 데이터 선인출을 위하여 추가된 참조 예측기의 전력 소모를 고려한 분석 모델이다. 이를 토대로 기존의 캐시 시뮬레이터에 데이터 선인출을 위한 모듈을 구현하여 선인출의 사용 여부에 따른 전력 소모량과 처리 시간을 모의 실험하였다. 실험 결과, 데이터 선인출은 전력 소모와 처리 시간의 측면에서 상반되는 영향을 미치는 것으로 나타났다. 즉, 데이터 선인출로 인하여 처리 시간은 단축되는 반면 전력 소모가 크게 증가됨이 입증되었다. 또한 데이터 선인출로 인한 전력 소모와 처리 시간의 관계를 정리하고, 임베디드 시스템의 성능 분석 모델을 제안하였다. 이는 다양한 선인출 기법을 채용한 임베디드 시스템의 성능 분석에 적용 가능할 것으로 판단된다. 향후에는 임베디드 시스템을 대상으로 멀티미디어 응용 프로그램의 처리 속도를 향상시킴과 동시에 전력 소모의 증가를 극소화 할 수 있는 메모리 운용 기법을 연구하고자 한다.

본 연구는 2005년도 나사렛대학교의 연구비 지원에 의하여 이루어진 연구로서, 관계부처에 감사드립니다.

참고문헌

[1] T. Mudge, "Power: A First-Class Architectural Design Constraint," *IEEE Computer*, Vol. 34, No. 4, pp.52-58, Apr. 2001.

[2] W. T. Shiu, "Memory Exploration for Low Power Embedded Systems", *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems*, pp.250-253, May. 1999.

[3] E. Witchel, S. Larsen and C. Scott, "Direct Addressed Caches for Reduced Power Consumption", *Proceedings of the 34th ACM/IEEE International Symposium on Microarchitecture*, pp.124-134, Dec. 2001.

[4] K. Defendorff and P. K. Dubey, "How Multimedia Workloads Will Change Processor Design," *IEEE Computer*, Vol. 30, No. 9, pp.43-45, Sep. 1997.

[5] A. Bona, M. Sami, D. Sciuto, C. Silvano, V. Zaccaira and R. Zafalon. "Energy Estimation and Optimization of Embedded VLIW Processors based on Instruction Clustering," *Proceedings of the 39th ACM/IEEE International Conference on Design automation*, pp.886-891, Jun. 2002.

[6] M. Kandemir, U. Sezer and V. Delaluz. "Improving Memory Energy Using Access Pattern Classification", *Proceedings of the ACM/IEEE International Conference on Computer Aided Design*, pp.201-206, 2001.

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

[8] F. Dahlgren, M. Dubois and P. Stenstrom, "Fixed and Adaptive Sequential Prefetching in Shared-memory Multiprocessors," *Proceedings of the International Conference on Parallel Processing*, pp.I56-63, Aug. 1993.

[9] 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.

[10] J. R. Lorch, and A. J. Smith. "Software Strategies for Portable Computer Energy Management," *IEEE Personal Communications*, Vol. 5, No. 3, pp.60-73, Jun. 1998.

[11] M. B. Kamble and K. Ghose, "Analytical Energy Dissipation Models For Low Power Caches", *Proceedings of the 1997 International Symposium on Lowpower Electronics and Design*, pp.143-148, Aug. 1997.

[12] P. Hicks, M. Walnock and R. M. Owens, "Analysis of Power Consumption in Memory Hierarchies", *Proceedings of the 1997 International Symposium on Low Power Electronics and Design*, pp.239-242, Aug. 1997.

[13] B. Mathew and A. Davis, "An Energy Efficient High Performance Scratch-pad Memory System" *Proceedings of the Design Automation Conference (DAC)*, 2004

[14] 이정훈, 최진혁, 김신덕, "저전력 온칩 메모리에 관한 연구 동향 및 개발 방향," 정보과학회지, 제20권, 제10호, pp.37-44, 2002. 10.

[15] M. D. Hill, Dinero III Cache Simulator, *http://www.ece.cmu.edu/~ece548/tools/dinero*.

[16] A. Srivastava and A. Eustace, "ATOM: A System for Builing Customized Program Analysis Tools," *Proceedings of the ACM SIGPLAN 94*, pp.196-205, 1994.

저자소개

문 현 주(Hyun Ju Moon)

1995~1997 충북대학교 전자계산학과 이학석사
1997~2003 충북대학교 전자계산학과 이학박사
2003~2005 나사렛대학교 전산정보학과 전임강사
2005 유비쿼터스바이오정보기술연구센터 책임연구원
2005~현재 남서울대학교 컴퓨터학과 전임강사
※ 관심분야 : 임베디드 시스템, 저전력 메모리 시스템 구조, 유비쿼터스 컴퓨팅, 센서 네트워크

유 현 배(Hyun Bae Yoo)

1992년 부경대학교 응용수학과 졸업
1994년 부경대학교 대학원 응용수학과 졸업(이학석사)
2000년 일본 Tsukuba대학교 대학원 공학연구과 졸업(공학박사)
2000년 일본 통신종합연구소(CRL) 연구원
2002년~ 현재 나사렛대학교 IT학부 조교수
※ 관심분야 : 임베디드 시스템, 저전력 메모리 시스템 구조, 유비쿼터스 컴퓨팅, 센서 네트워크, 영상처리 및 이해, 가상현실, 유니버설디자인