韓國컴퓨터情報學會 論文誌
第15卷 第3號, 2010. 3.

2010-15-3-1-1

버퍼 메모리 접근 정보를 활용한 동적 전압 주파수 변환 기법

곽 종 욱*, 김 주 환**

Dynamic Voltage and Frequency Scaling based on Buffer Memory Access Information

Jong Wook Kwak*, Ju-Hwan Kim**

요 약

프로세서 플랫폼이 무선의 모바일 시스템으로 변화하면서 내장형 모바일 프로세서들의 성능은 계속적으로 향상되었으며 기능은 보다 더 강력해 지고 있다. 무선의 휴대용 장비들은 유선 장비에 비해 휴대용 전원에 의한 제한된 전력을 공급받기 때문에, 이러한 시스템들에 대한 효율적 에너지 관리 기술의 중요성은 점차 증가하고 있다. 한편, 메모리 시스템은 프로세서 관점에서 시스템 전체의 성능을 저하 시키는 주된 요소 가운데 하나이다. 비록 휴대용 전원의 효과적 활용을 위한 DVFS 기법과 관련된 많은 연구들이 존재하지만, 프로세서와 메모리 사이의 상호 관계에 대한 최근의 연구는 부족한 실정이다. 본 연구에서는 무선의 모바일 장치들에서 활용되는 내장형 응용 프로그램의 장단기 메모리 접근 특성을 반영하기 위한 새로운 DVFS 레벨 예측 알고리즘을 소개한다. 모의 실험 결과 본 논문에서 제시하는 DVFS 정책은 메모리 접근이 많은 벤치마크 프로그램의 경우 5.86%의 소비 에너지 감소 효과를 보여주고 있으며, 평균적으로는 3.60%의 소비 에너지 감소 효과를 보여주고 있다.

Abstract

As processor platforms are continuously moving toward wireless mobile systems, embedded mobile processors are expected to perform more and more powerful, and therefore the development of an efficient power management algorithm for these battery-operated mobile and handheld systems has become a critical challenge. It is well known that a memory system is a main performance limiter in the processor point of view. Although many DVFS studies have been considered for the efficient utilization of limited battery resources, recent works do not explicitly show the interaction between the processor and the memory. In this research, to properly reflect short/long-term memory access patterns of the embedded workloads in wireless mobile processors, we propose a memory buffer utilization as a new index of DVFS level prediction. The simulation results show that our solution provides 5.86% energy saving compared to the existing DVFS policy in case of memory intensive applications, and it provides 3.60% energy saving on average.

• 제1저자 : 곽종욱
• 투고일 : 2010. 02. 02, 심사일 : 2010. 02. 22, 게재확정일 : 2010. 03. 18.
* 영남대학교 컴퓨터공학과 조교수 ** 서울대학교 전기컴퓨터공학과 박사과정
※이 논문은 2009년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 2009-0071583).

▸ Keyword : 저전력(low-power), 내장형시스템(embedded system), 동적전압주파수변환(DVFS), 메모리접근(memory access), 메모리버퍼(memory buffer)

Ⅰ. 서 론

가히 유비쿼터스(ubiquitous) 시대라 불리어질 정도의 현 상황에서, 내장형 기기(embedded device) 및 이동 기기(mobile device)와 관련된 시장이 급격하게 팽창하고 있다. 이로 인해 내장형 프로세서(embedded processor)에 대한 관심도 더불어 증가하였다. 내장형 프로세서는 사실상 무한하게 전력을 공급받는 데스크탑용 프로세서나 서버용 프로세서와는 달리 배터리에 의해서 제한된 전력만을 공급받기 때문에, 성능뿐만 아니라 전력 소모량도 전체 시스템의 설계에 있어서 매우 중요한 고려의 대상이 된다.

한편, 빠르게 발전하고 있는 초고밀도 집적회로(Very Large Scale IC, VLSI) 기술은 프로세서의 처리 속도를 급격하게 높여왔다. 또한, 명령어 수준의 병렬 처리 기술(Instruction Level Parallelism, ILP)의 발전과 클록(clock) 속도의 향상은 계산 속도의 증가를 더욱 가속화시켜 프로세서의 성능을 매년 60% 이상 향상시켜 오고 있다. 그러나 컴퓨터 시스템 상에서 또 하나의 중요한 축을 이루는 메모리 시스템(memory system)은 동작 속도의 측면에서 크게 개선되지 않았다. 그 예로, 대표적인 메모리 타입인 DRAM의 경우 집적도의 향상과 비교하여, 매년 10% 미만의 동작 속도 증가만을 보여주고 있다. 비록, VLSI 기술의 발전으로 인해 동적 램의 용량은 비약적으로 늘어났지만, 접근 속도의 느린 발전으로 인해서 프로세서와의 속도 차이는 갈수록 증가하고 있는 현실이다. 이와 같은 프로세서와 메모리 시스템간의 큰 속도 차이는 시스템 성능 향상의 결정적인 저해 요인으로 작용한다[1].

메모리 병목 현상의 문제는 기존의 프로세서 환경 하에서 시스템의 동작 속도, 즉 성능 측면에서 큰 영향을 미쳐왔다. 하지만, 내장형 기기 및 이동 기기가 보편화되는 오늘날의 환경에서는 성능뿐만 아니라 에너지 소비적 측면에서도 큰 문제를 야기한다. 시스템의 동작에 있어서 메모리로의 접근이 필수적 요소라면, 해당 메모리로의 접근 동안 프로세서의 환경을 에너지 소비 측면에서 보다 더 최적의 상태로 관리하는 기법이 필요하다.

본 연구에서는 내장형 시스템에서 사용될 수 있는 효율적인 에너지 관리 정책의 하나로, 동적 전압 주파수 조절 (Dynamic Voltage and Frequency Scaling, 이하 DVFS) 정책을 위해 제공되어 질 수 있는 프로세서의 새로운 성능 평가 지표를 제안한다. 효율적인 에너지 관리 정책을 위한 DVFS 기법의 필요성은 자명하다[2]. 특히, 현재의 프로세서가 제공하는 프로세서 자체적인 이벤트 정보 이외에 DVFS에서 효과적으로 활용할 수 있는 추가적인 성능 평가 지표로써 메모리 접근 요청 버퍼의 사용량은 메모리로의 접근에 대한 효율적인 관리를 가능케 하여, 시스템의 동작 속도와 함께 소비 전력에도 큰 영향을 미치게 된다[3][4].

본 연구에서는 다양한 실험을 통해 현재 내장형 프로세서에 의해 제공되고 있는 다양한 메모리 접근 요청 버퍼의 사용량을 관찰하고, 이들이 프로세서의 에너지 효율적 측면에서 효과가 있는지 확인한다. 이를 통해 저전력 내장형 시스템에서 효과적으로 활용할 수 있는 새로운 DVFS 정책 및 동작 알고리즘을 제안하며, 이를 위해 요구되는 새로운 하드웨어 모듈을 설계한다. 구체적으로 메모리 접근 요청 버퍼의 사용량을 개수하기 위한 카운터 기반 하드웨어 모듈과 이를 모니터링 하기 위한 성능감시장치를 소개한다. 특히 이들은 단기 메모리 접근 정보를 분석하기 위한 정보로 활용된다.

아울러, 제안된 시스템의 성능 평가 지표로서, 각 응용 프로그램별 실행 시간과 소비 에너지를 분석하여 기존 연구와의 차별성을 제시한다. 이를 위해 본 연구에서는 각 응용 프로그램별 메모리 접근 패턴을 분석한 뒤, 이를 통해 프로세서 집중형 작업 부하와 메모리 집중형 작업 부하를 판단하며, 각 작업 부하별 성능 향상의 정도를 구분하여 소개한다.

이하 본 논문의 구성은 다음과 같다. 2장에서는 배경 지식 및 관련 연구에 대해 소개한다. 3장에서는 메모리 접근 정보를 활용하는 DVFS 기법의 개선에 대해 소개한다. 그리고 4장에서는 본 논문에서 제안한 주제의 검증을 위한 모의 실험을 수행하며, 실험 결과를 분석한다. 끝으로 5장에서는 결론 및 향후 연구 과제를 제시한다.

Ⅱ. 배경 지식 및 관련 연구

프로세서의 전력 소비 문제와 함께 프로세서의 발열 문제 또한 시스템의 안정성과 맞물려 중요한 문제로 부각되고 있다. 즉, 시스템의 크기가 점점 작아지면서 프로세서의 발열 문제는 시스템의 전체적인 안정성을 좌우하는 중요한 요소가 되었다. 이러한 소비 전력과 발열 문제를 해결하기 위해 오늘날까지 다양한 연구가 계속해서 진행되고 있다.

$$P \propto f\, V^2 \quad \text{식 (1)}$$

프로세서의 소비 전력은 일반적으로 다음과 같은 식으로 표현된다. 식 (1)에서 f 는 프로세서의 동작 주파수이며, V 는 동작 전압이다. 따라서 식 (1)에서 보이는 바와 같이, 프로세서의 소비전력을 감소시키기 위해서는 프로세서의 동작 주파수와 전압을 낮춰야한다. 하지만 프로세서의 동작 전압과 주파수를 낮추는 것은 프로세서의 성능을 저하시키는 원인으로 작용하므로, 그 영향을 고려하여 선택에 신중을 기해야 한다. 또한, 식 (1)을 살펴보면 소비 전력은 동작 주파수에 비례하고, 동작 전압의 제곱에 비례하므로, 동작 전압을 조절해주는 것이 소비 전력의 감소와 비교하여 보다 더 큰 효과를 보인다는 사실을 알 수 있다.

한편, 본 논문의 근간을 이루는 DVFS 기법은 프로세서의 동적 에너지(dynamic energy) 관리 분야에 있어서 매우 효과적인 기법 중 하나로 알려져 있으며, 이 분야에 대해서는 지금도 활발하게 많은 연구가 진행되고 있다. DVFS 기법은 프로세서의 작동 중에 필요에 따라 동적으로 프로세서의 동작 전압과 주파수를 변경할 수 있는 기술로, 이를 활용하면 보다 낮은 전력을 소비하면서도 이전과 같은 수준의 성능을 얻을 수 있게 된다. 따라서 DVFS 기법은 모바일 시스템의 전력소비와 발열문제에 대한 효과적인 해결책으로 알려져 있다. 이를 반영하듯 DVFS 기술은 대표적인 내장형 모바일 프로세서들이라 할 수 있는 Intel의 XScale, AMD의 Athlon, Trensmeta의 Crusoe등의 프로세서에 적극 활용되어 구현 및 상용화 되었다[5].

DVFS의 기본적인 원리는 프로그램 실행 중에 프로세서가 항상 최고의 성능을 발휘할 필요가 없다는 사실에 기반한다[6]. 프로세서가 가진 자원은 한정되어 있고, 이러한 점은 프로세서의 성능을 제한할 수밖에 없다. 자원적 제약 요소가 강한 내장형 시스템의 경우 이러한 사실은 더욱 중요하게 부각된다. 따라서 효율적인 에너지 관리를 위해 순간순간 필요한 만큼의 전압과 주파수로 프로세서를 작동시켜 프로세서가 필요로 하는 최소수준의 성능만을 발휘하게 하고, 이를 통해 전체 전력소비를 낮추면서도 일정 수준의 성능을 보장하는 것이 DVFS의 핵심이다. 그림 1은 “시스템이 발휘할 수 있는 최대 성능”과 “실제 수행에 필요한 성능”의 관점에서 DVFS의 효과를 잘 보여준다.

그림 1의 (a)는 내장형 시스템에서 요구하는 “실제 필요 성능”과 비교하여 프로세서의 “발휘 성능”이 높게 설계된 경우이다. 사실상 무한한 전력을 공급받는다고 할 수 있는 일반적인 컴퓨터 시스템 혹은 서버 환경에서는 시스템의 발휘 성능이 높을수록 우수한 환경이라 할 것이다. 이에 비해 그림 1의 (b)는 시스템의 “발휘 성능”을 “실제 필요 성능” 수준으로 조절함으로, 응용 프로그램의 수행 관점에서 시스템이 요구하는

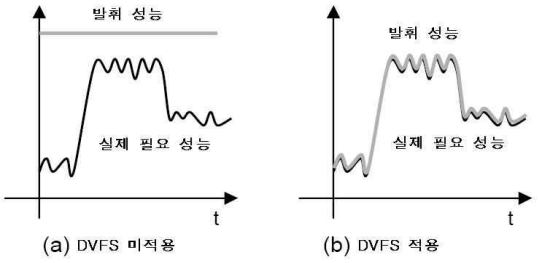

그림 1. DVFS의 효과
Fig. 1. DVFS Effect

적정 수준의 성능을 발휘하게 된다. 이러한 설계 원리는 내장형 시스템의 설계에 있어서 시스템의 과도 설계(over-design)를 막아 낭비되는 자원 및 비용을 효과적으로 줄이는 역할을 한다. 결론적으로 DVFS 기법의 효과적인 활용은 에너지 효율적인 저전력 시스템의 설계를 가능하게 할 뿐만 아니라, 시장 가격에 민감한 내장형 시스템의 주된 응용 분야인 각종 휴대용 무선 단말기기 시장에서 최소 요구 자원의 효율적 활용을 통한 가격 경쟁력 측면에서도 우수한 제품의 생산에 기여하게 된다.

이상에서 살펴본 바와 같이 DVFS 정책의 효율적 활용에 대한 필요성은 자명하다. 한편, 효율적인 DVFS 정책의 구현을 위해서는 프로세서의 상태를 정확하게 파악하는 것이 중요하다. 현재의 프로세서들은 내부에서 발생하는 다양한 이벤트에 관한 정보를 외부에 알려주는데, 일반적으로 이벤트의 발생 시점에 특정 레지스터 값을 조절하는 방식을 취한다. 일반적인 DVFS 정책에서는 이러한 이벤트 정보를 취합하여 프로세서의 상태를 평가한다. 하지만 기존의 방식에서는 프로세서가 제공하는 이벤트의 종류와 정보의 개수가 제한되어있기 때문에 프로세서의 상태를 정확하게 평가하기에는 제약이 있다. 따라서 보다 더 효율적으로 프로세서의 상태를 정확히 평가할 수 있는 성능평가 지표를 찾는 것이 중요하다. 이를 위해 본 연구에서는 효율적인 DVFS 정책을 위해 제공되어 질 수 있는 프로세서의 새로운 성능 평가 지표를 제안한다.

III. 메모리 접근 정보를 활용한 DVFS

3.1 시스템 구조

본 연구에서는 대표적인 내장형 프로세서라 할 수 있는 XScale 프로세서를 기반으로 해당 프로세서의 개선된 형태를 제안한다. XScale 프로세서에는 Write 버퍼, Fill 버퍼, Fetch

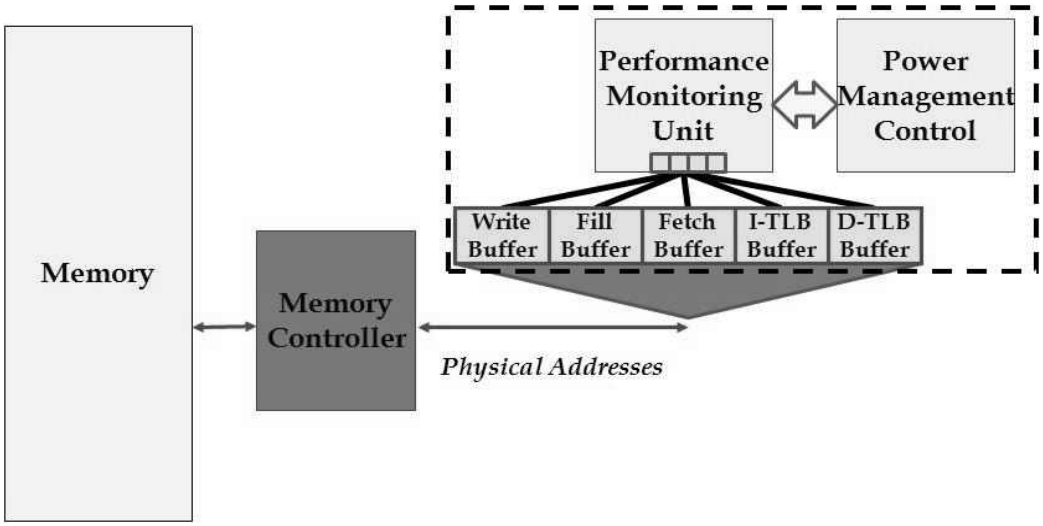

그림 2. XScale 프로세서의 변경된 메모리의 구조
Fig. 2. XScale Processor and its Memory Structure

버퍼, 명령어 TLB 버퍼, 데이터 TLB 버퍼 등의 메모리 접근 요청 버퍼가 존재한다. 이들 버퍼에서는 프로세서에서 메모리로 전송되는 메모리 접근 요청을 저장하여 두었다가 프로세서 코어와 메모리 사이를 연결하는 버스가 유휴(idle) 상태일 때 이를 하나씩 버스를 통해 메모리 제어기로 전송하게 된다[7].

이들 메모리 접근 요청 버퍼에 저장될 수 있는 요청의 최대 개수는 각 버퍼마다 다르다. Write 버퍼는 8개, Fill 버퍼는 4개, 2개의 Fetch 버퍼는 각 1개씩, 명령어 TLB와 데이터 TLB 역시 각 1개씩이며, 전체 크기는 16(= 8+4+2*1+1+1)이 된다. 따라서 기존의 코어 내부에 존재하는 성능감시장치(PMU)에 4 비트 크기의 카운터를 추가한다. 각 버퍼들을 관리하는 제어기로부터 각 버퍼의 현재 사용량을 전달받을 수 있으면 전체 메모리 접근 요청 버퍼의 사용량을 측정할 수 있다. 그림 2에 변경된 프로세서 구조가 소개되어 있다.

각 메모리 접근 요청 버퍼는 자기 자신의 버퍼에 새로운 메모리 접근 요청이 들어오거나 메모리 접근 요청이 버퍼에서 나가게 되어 버퍼의 사용량이 변하게 되면 이를 성능감시장치에 알린다. 성능감시장치는 각 버퍼의 사용량을 집계하여 전체 메모리 접근 요청 버퍼의 사용량을 4 비트 카운터에 기록한다. 이러한 4 비트 카운터의 정보를 다른 성능 관련 이벤트 정보들과 마찬가지로 별도의 레지스터를 통해 외부로 알려주게 되며, 이 정보를 직접 전력관리장치에 제공하게 함으로, 프로세서 수준의 DVFS 조절을 가능하게 한다.

3.2. DVFS 정책

일정한 주기를 기반으로 분석, 평가, 설정 과정을 반복하는 DVFS 정책을 사용하게 되면, 하나의 주기 내에서 초기에 정해진 동작 전압과 주파수를 유지하기 때문에 메모리 접근 정도가 급격하게 변동하는 상황에 효과적으로 대응할 수 없다. 따라서 이로 인해 발생하는 예측 실패 비용은 DVFS 정책의 효율성을 떨어뜨리는 원인으로 작용한다. 또한 하나의 프로그램 안에서도 메모리 접근 요청 버퍼의 사용량은 매 순간 급격하게 변할 수 있기 때문에, 이것만을 기반으로 하여 비교적 긴 시간의 예측을 통해 수행되는 DVFS 정책은 오히려 전체적인 시스템 전체적인 성능 저하를 초래할 가능성이 높다. 하지만, 메모리 접근 요청 버퍼에 저장되어 있는 메모리 접근 요청의 개수를 통해 메모리 접근 요청을 예측할 경우, 예측 시간의 범위는 비교적 좁은 편이다.

따라서 본 논문에서는 DVFS 정책을 장기 정책과 단기 정책으로 나누어 서로 다른 DVFS 정책을 적용하는 방식을 제안한다. 장기 DVFS 정책으로는 논문[8]에서 제안한 DVFS 정책을 사용하고, 단기 DVFS 정책으로는 메모리 접근 요청 버퍼의 사용량에 기반을 둔 방식을 사용한다.

단기 DVFS 정책의 기본적인 의사결정은 본 논문에서 제안한 메모리 접근 요청 버퍼의 사용량을 활용하는 것으로 앞 절에서 소개된 카운터의 값을 이용하여 단기 메모리 접근 정도를 효과적으로 파악할 수 있다. 결론적으로 본 논문에서 제안하는 기법은, 장기 DVFS 정책이 짧은 시간에 급변하는 메

모리 접근 정도의 변동에 대처할 수 없는 단점을 단기 DVFS 정책을 통해 보완할 수 있으며, 단기 DVFS 정책이 긴 시간을 예측할 수 없는 단점을 장기 DVFS 정책을 통해 보완할 수 있다. 이러한 시나리오를 그림 3을 통해 나타내었다. 그림 3에서는 장기 DVFS 정책으로 인한 주기적인 동작 주파수의 변화와 함께, 각 주기 내부에서의 단기 DVFS 정책으로 인한 추가적인 동작 주파수의 변화를 보여주고 있다.

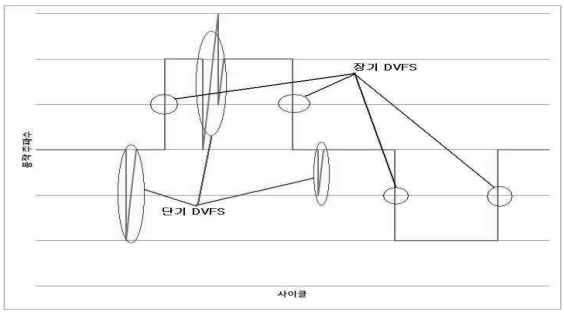

그림 3. 장단기 DVFS 정책으로 인한 동작 주파수의 변화
Fig. 3. Long/Short Term DVFS Policy vs. Operation Frequency

한편, 메모리 접근 정보를 바탕으로 하는 DVFS 조절 기법은 다음과 같이 구현된다. 메모리 접근 요청 버퍼의 사용량이 높다는 것은 조만간 많은 메모리 접근이 발생할 것을 암시한다. 따라서 메모리 접근에 따른 프로세서의 지연이 발생할 가능성이 높으므로 현재보다 프로세서의 동작 전압과 주파수를 낮추는 것이 성능과 에너지 면에서 유리하다. 반면 메모리 접근 요청 버퍼의 사용량이 낮다는 것은 당분간 메모리 접근 횟수가 적을 것을 암시한다. 이러한 상황에서 프로세서의 동작 전압과 주파수를 낮추는 것은 직접적으로 프로세서의 성능에 영향을 미치므로 현재의 동작 전압과 주파수를 유지하거나 혹은 올려주는 것이 유리하다. 따라서 메모리 접근 요청 버퍼의 사용량에 기반을 둔 DVFS 정책의 기본 방향은 메모리 접근 요청 버퍼의 사용량이 높은 경우에는 프로세서의 동작 전압과 주파수를 낮춰서 소비 에너지를 줄이고, 이후에 메모리 접근 요청 버퍼의 사용량이 낮아지게 되면 다시 동작 전압과 주파수를 올려서 프로세서의 성능을 보장하는 것이다. 본 논문에서는 위에서 언급한 것처럼 기본적으로 논문[8]에서 제시한 방법을 사용하여 장기 DVFS 정책을 관리하며, 메모리 접근 요청 버퍼 사용량이 일정량 이상 변할 때마다 단기 DVFS 정책을 적용하게 된다. 이를 최적화하기 위해서는 동작 전압과 주파수의 변환 시점, 그리고 변환 정도를 결정하는 것이 중요하다. 하지만 프로세서의 성능 변화가 전체 시스템의 성능에 미치는 영향은 여러 요소에 의해 동적으로 변화할 뿐만 아니라 동적으로 전압과 주파수를 변환하는데 필요한 오버헤드(overhead)도 계산해야 하기 때문에 분석적인 접근법 보다는 다양한 조건 값을 사용한 반복 실험을 통해 휴리스틱(heuristic)에 의한 방법을 적용하였다[9][10].

Algorithm DVFS
input :
buffer the amount of memory access request occupied (0 ~ 15)
current current processor performance level (0 ~ 5)
base base processor performance level (0 ~ 5)
output :
next next processor performance level (0 ~ 5)

if ***buffer*** < 6 then
    if ***current*** < ***base*** - 3 then ***next*** = ***current*** + 2
    else if ***current*** < ***base*** then ***next*** = ***current*** + 1
    else if ***current*** ≥ ***base*** then ***next*** = ***current***
else ***buffer*** ≥ 6 then
    next = ***current*** - 5
if ***next*** < 0 then ***next*** = 0
else if ***next*** > 5 then ***next*** = 5
return ***next***

그림 4. DVFS 정책
Fig. 4. DVFS Policy

그림 4에 본 논문에서 제안하고자 하는 DVFS 정책이 소개되어 있다. 이를 위하여 현재 메모리 접근 요청 버퍼의 사용량(buffer)과 프로세서의 현재 성능 수준(current), 기본 성능 수준(base)이라는 세 가지 요소를 가지고 동작 전압과 주파수를 설정한다. 메모리 접근 요청 버퍼의 사용량은 본 논문에서 제안하는 방식으로 성능감시장치에 추가된 4 비트 카운터를 통해 얻을 수 있고, 현재 성능 수준은 프로세서의 전력관리장치를 통해 얻을 수 있으며, 기본 성능 수준은 장기 DVFS 정책에 의해 결정된 주기 내의 성능 수준으로 외부로부터 입력 받아야 한다. 프로세서의 성능 수준은 실험에 사용한 6단계의 동작 전압과 주파수를 기준으로 나누었다(표 2 참조). 이를 그림 4를 통해 정리하였으며, 실험에 사용된 기준 값은 반복 실험을 통해 얻어진 결과값이다. 그림 4에서는 현재 메모리 접근 요청 버퍼 사용량이 일정 수준 이상일 경우에는 프로세서의 성능을 최소로 발휘하게 하였으며, 일정 수준 이하일 경우에는 현재 성능과 기본 성능을 고려하여 현재 성능 수준을 유지하거나 일정 수준 상승시키도록 하였다.

IV. 성능 평가

4.1 실험 환경

본 논문의 성능 평가를 위해 다음과 같은 모의실험(simulation)을 수행하였다. 본 논문에서는 SimpleScalar[11]를 기반으로 XScale의 아키텍처를 구현한 XEEMU를 활용한다[12]. XEEMU는 전력 소비를 시뮬레이션하기 SimpleScalar에 Sim-panalyzer의 기능을 추가한 시뮬레이터다[13].

실험에 사용된 XScale 프로세서 모델의 기본 설정 값은 표 1과 같다. 본 논문에서는 일반적인 XScale 프로세서의 구조적 특성을 따랐으며, XScale에서 제공하는 대부분의 기능을 활용하도록 하였다. DVFS에 사용되는 전압, 주파수 조합은 시뮬레이터에서 지원하는 조합 가운데 전압차(voltage level)가 분명한 6가지를 선택하였으며, 표 2를 통해 정리하였다.

표 1. 시뮬레이터의 설정 값
Table 1. Simulation Environment

항목	설정 값
Data L1 Cache	32KB
Instruction L1 Cache	32KB
Data L2 Cache	None
Instruction L2 Cache	None
Integer ALU	4
Integer Multiplier / Divider	1
Floating Point ALU	4
Floating Point Multiplier / Divider	1
Memory Frequency	100 MHz
Memory Voltage	3.3 V
Memory Access Bus Width	8 bytes
Memory Access Request Bus Width	2 bytes

제안된 시스템의 성능 평가 지표로서, 본 연구에서는 각 응용 프로그램별 메모리 접근 패턴을 분석한다. 이를 통해 프로세서 집중형 작업 부하와 메모리 집중형 작업 부하를 판단하며, 각 작업 부하별 대표적 응용 프로그램을 선택한 뒤, 해당 응용 프로그램의 작업 부하별 IPC의 변화를 분석한다. 최종적으로 본 논문에서 제안한 방식과 기존의 방식, 관련 연구의 방식을 상호 비교하여 실행 시간과 소비 에너지 측면에서 성능 향상의 정도를 소개한다.

표 2. 동작 전압과 주파수 조합
Table 2. Performance Level vs. Voltage and Frequency

성능 수준	동작 전압	동작 주파수
0	1.0	333
1	1.1	400
2	1.215	533
3	1.32	600
4	1.4	666
5	1.5	733

벤치마크 프로그램은 GCC Code-Size Benchmark Environment (CSiBE)를 선택하였으며, XEEMU 시뮬레이터에 수행 가능한 형태로 컴파일되었다. 본 논문의 실험에서 사용되는 6가지 대표 벤치마크의 특징은 다음과 같다[14].

- png_decode : png 형식의 이미지 파일을 읽어 들여서 ppm 형식의 이미지 파일로 전환한다.
- zlib : 1.4 Mbytes 크기의 이진 파일을 읽어 들여서 압축하는 프로그램을 수행한다.
- diff : 두 텍스트 파일을 읽어 들여서 두 파일 사이의 다른 부분을 찾아 그 결과를 또 다른 텍스트 파일로 출력한다.
- png_encode : ppm 형식의 이미지 파일을 읽어 들여서 png 형식의 이미지 파일로 전환한다.
- lp_solve : simplex 방법과 B&B 방법을 기반으로 혼합 정수 선형 프로그래밍의 해답을 찾는다.
- flex : 어휘 분석기 생성 프로그램으로, 정규표현식과 C 코드가 쌍으로 이루어진 rule의 읽어 들여 yylex() 루틴을 정의하는 C 소스 파일(lex.yy.c)을 출력을 생성한다.

4.2 실험 결과 및 분석

본 논문의 실험 결과를 분석하기 위해, 우선 각 벤치마크의 실행 패턴을 분석할 필요가 있다. 응용 프로그램들은 해당 프로그램의 실행 특성에 따라 메모리 종속적 프로그램과 계산

종속적 프로그램으로 구분 지을 수 있다. 표 3은 각 벤치마크 프로그램 별로 전체 실행 명령어 중에서 캐시 등의 영향을 고려하여 실제 메모리 접근이 발생한 횟수를 정리한 것이다. png_encode, zlib, diff의 경우 다른 벤치마크 프로그램에 비해 메모리 접근 비율이 높음을 알 수 있다. 본 논문에서는 이러한 프로그램을 메모리 집중형, 이에 속하지 않는 다른 벤치마크 프로그램들을 프로세서 집중형이라고 명명한다.

표 3. 벤치마크 프로그램의 메모리 접근 특성 비교
Table 3. Memory Access Pattern for Each Benchmark

벤치마크 프로그램	메모리 접근 횟수 / 명령어	특성
png_encode	0.029621464	메모리 집중형 작업부하
zlib	0.024541132	
diff	0.012309369	
jpegtran1	0.007743764	프로세서 집중형 작업부하
egrep	0.005455382	
png_decode	0.00471568	
jpegtran2	0.004227549	
jikes	0.002609948	
djpeg	0.002234698	
lp_solve	0.001219924	
cjpeg	0.000973206	
flex	0.00031107	
lame	7.51738E-05	
compiler	3.25576E-05	
c4.5	2.07691E-05	
wc	6.44793E-06	
teem_enhex	1.3904E-06	
teem_dehex	1.31386E-06	

다음으로, 메모리 접근 명령어로 인해 발생한 프로세서의 지연이 프로세서의 전체적인 성능에 어떤 영향을 미치는지 분석한다. 이를 위해 벤치마크 중에 메모리 집중형 프로그램인 png_encode와 프로세서 집중형 프로그램인 flex를 선택하여 1ms 단위로 메모리 접근 요청 수와 처리 명령어 수를 얻어 이에 대한 IPC를 계산하였다.

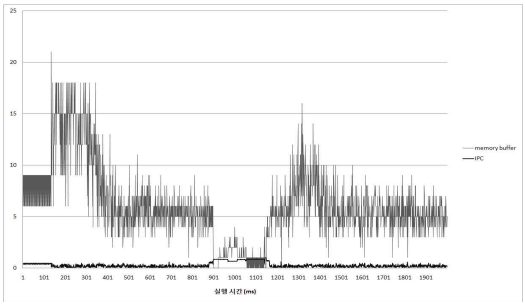

그림 5. 메모리 접근 요청 버퍼 사용량과 IPC의 변화 - png_encode
Fig. 5. Memory Buffer Usage and IPC - png_encode

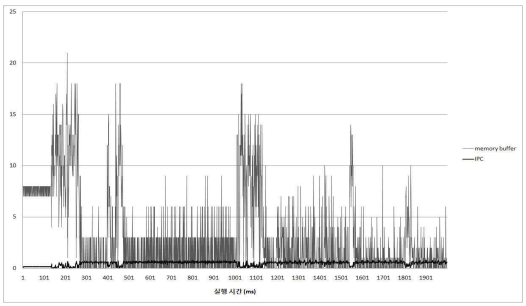

그림 6. 메모리 접근 요청 버퍼 사용량과 IPC의 변화 - flex
Fig. 6. Memory Buffer Usage and IPC - flex

그림 5를 살펴보면 메모리 집중형에 속하는 png_encode의 경우, 메모리 접근 요청 버퍼 사용량은 3과 7사이에, IPC는 0.3 이하에 주로 분포하고 있다. 전체 프로그램의 평균 IPC도 0.2201로 상당히 낮은 편이다. 반면, 그림 6의 프로세서 집중형 프로그램에 해당하는 flex의 경우에는 메모리 접근 요청 버퍼 사용량은 0과 5사이에, IPC는 0.5와 0.7 사이에 주로 분포하고 있다. 전체 프로그램의 평균 IPC도 0.6324로 png_encode에 비해 상당히 높다. 이상의 결과를 통해 메모리 집중형 프로그램은 프로세서 집중형 프로그램에 비해 메모리 접근 요청 버퍼 사용량이 높고 변동의 폭이 크며, 그로 인해 평균 IPC도 상당히 낮고 IPC의 편차도 큼을 확인할 수 있다.

png_encode와 flex 사이에 전체적인 수치의 차이는 있지만 두 가지 모두 메모리 접근 요청 버퍼의 사용량에 따른 IPC의 변화 경향은 동일하게 나타난다. 메모리 접근 요청의 사용량이 높을수록 IPC가 감소하고, 반대로 메모리 접근 요청의 사용량이 낮아지면 IPC는 증가하는 경향을 두 그래프에서 확인할 수 있다. 이를 통해 메모리 접근 요청 버퍼의 사용량이 높

을수록 프로세서에서는 메모리 접근으로 인한 지연이 길게 발생하고, 이로 인해 프로세서 전체의 성능이 더 많이 감소한다는 것을 알 수 있다.

또한 그림 5와 그림 6의 실험 결과를 통해 다음과 같은 사실도 유추할 수 있다. 프로세서의 동작 주파수를 증가시키는데 있어서 메모리 집중형의 경우에는 프로세서 집중형에 비해 성능 향상의 폭이 작고, 에너지 소비 증가의 폭이 크다는 것이다. 결국 메모리 접근이 빈번하게 발생하는 경우에는 굳이 높은 주파수로 동작시키기보다 낮은 주파수로 동작시키는 것이 성능과 에너지 측면에서 모두 효율적이라는 점을 알 수 있다. 본 연구에서도 이러한 실험 결과에 대한 분석을 바탕으로 새로운 DVFS 정책을 결정하는데 있어서, 작업부하의 메모리 접근 정도를 중요한 요소로 반영하였다. 이상의 실험을 통해 본 논문에서 제안하는 주제의 타당성을 검증할 수 있다. 다음으로 메모리 접근 정보를 활용하는 DVFS 기법의 각 벤치마크별 실행시간과 소비에너지의 변화를 분석해 본다.

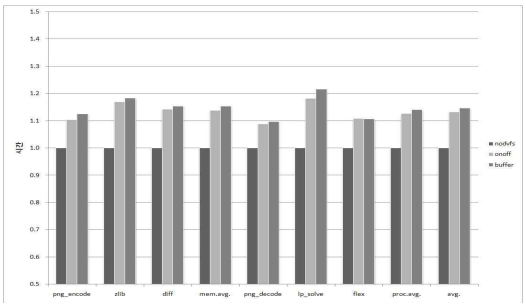

그림 7. nodvfs, onoff, buffer의 벤치마크별 실행 시간
Fig. 7. Execution Time : nodvfs, onoff, buffer

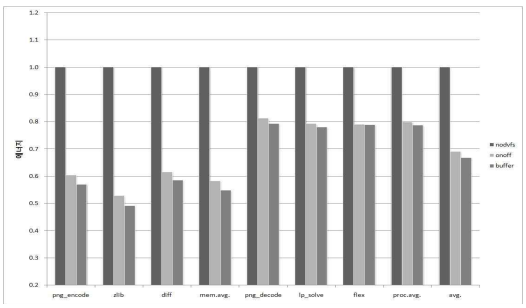

그림 8. nodvfs, onoff, buffer의 벤치마크별 소비 에너지
Fig. 8. Energy Consumption : nodvfs, onoff, buffer

그림 7과 그림 8은 각 벤치마크별 실행 시간과 소비 에너지를 측정한 실험 결과이다. 전체적인 실험은 적용한 DVFS 정책에 따라 3가지로 나누어 수행했으며, 이들은 각각 nodvfs, onoff, buffer로 표시하였다. nodvfs는 추가적인 DVFS 정책을 적용하지 않고 프로세서의 최고 성능을 발휘하여 프로그램을 수행했을 경우로 동작 전압 1.5V, 동작 주파수 733 MHz로 설정하였다. onoff는 논문[8]에서 제안한 방식으로 온칩 접근과 오프칩 접근 사이의 비율에 기반을 둔 DVFS 정책이다. PFloss 값은 10%, 윈도우 크기는 25, 한 주기의 길이는 50ms로 설정하였다. buffer는 본 논문에서 제안하는 메모리 접근 요청 버퍼의 사용량에 기반을 둔 DVFS 정책으로, 기본적인 장기 DVFS 정책으로는 onoff와 같은 정책을 사용하지만, 한 주기 내에서도 메모리 접근 요청 버퍼 사용량의 변화에 따른 단기 DVFS 정책을 적용한 방식이다. 6가지의 벤치마크 프로그램 항목 이외에 추가적으로 메모리 집중형 벤치마크 프로그램인 png_encode, zlib, diff의 평균은 mem.avg.로, 프로세서 집중형 벤치마크 프로그램인 png_decode, lp_solve, flex의 평균은 proc.avg.로 표시하였으며, 전체 벤치마크의 평균은 avg.로 표시하였다.

그림 7의 그래프에는 3가지 DVFS 정책에 따른 벤치마크의 실행 시간과 평균을 표시하였다. 세로축은 실행 시간을 표시하며, 이 시간은 nodvfs 정책에 따른 실행 시간을 기준으로 정규화(normalization) 하였다. onoff 정책은 nodvfs와 비교하여 메모리 집중형에서는 13.79%, 프로세서 집중형에서는 12.57%의 시간 지연이 발생하였고, 평균적으로 13.18% 정도의 시간이 지연되었다. 이는 10%로 설정한 PFloss와 비교하여 큰 차이가 없었으며, 이러한 결과는 논문[8]에 제시된 결과와 유사하다. buffer 정책을 적용했을 경우에는 nodvfs와 비교하여 메모리 집중형에서는 15.34%, 프로세서 집중형에서는 13.95%의 시간이 지연되었고, 평균적으로 14.64%의 시간 지연이 발생하였다. buffer 정책과 onoff 정책을 비교해보면 buffer 정책을 적용했을 경우에는 onoff 정책을 적용했을 경우보다 메모리 집중형은 1.36%, 프로세서 집중형은 1.19%, 평균 1.27%의 실행 시간이 지연되었다. 특이하게 flex에서는 buffer 정책을 적용했을 경우에 onoff 정책을 적용했을 경우보다 0.08% 정도 실행 시간이 단축되는 결과를 보여주었다. 이는 flex 벤치마크 자체의 전체 실행 시간이 짧아서 onoff 정책의 회귀분석 접근법이 효과적으로 적용되지 못 하였고, 그로 인해 발생한 오차를 buffer 정책을 통해 수정할 수 있었던 것으로 예측된다. 그림 8은 3가지 DVFS 정책에 따른 벤치마크의 소비 에너지와 평균에 관한 그래프이다. 세로축은 프로세서의 소비 에너지를 표시하며, 이 에너지는 그림 7의 실행 시간 그래프와 마찬가지로 nodvfs 정책에 따른 소비 에너지를 기준으로 정규화 하였다. 우선 onoff 정책은

nodvfs 정책과 비교하여 평균적으로 69.04% 정도의 에너지를 소비하여 30.96% 정도의 에너지 소비를 감소시켰다. 메모리 집중형에서는 평균 58.27%, 프로세서 집중형에서는 평균 79.81%의 에너지를 소비하여 메모리 집중형에서 프로세서 집중형보다 뛰어난 에너지 소비 절감 효과를 보여주었다. buffer 정책을 적용했을 경우에는 nodvfs와 비교하여 메모리 집중형에서는 54.88%, 프로세서 집중형에서는 78.73%의 에너지가 소비되어 평균적으로는 66.80%의 에너지가 소비되었다. 다시 말해서 33.20%의 에너지 소비 절감 효과가 있었다. 프로세서 집중형의 경우에는 onoff 정책의 79.81%와 비교하여 에너지 소비 절감의 폭이 그리 크지 않은데, 이는 프로세서 집중형의 특성상 메모리 접근 요청이 빈번하게 발생하지 않아서 onoff 정책이외의 추가적인 DVFS 작업이 수행될 여지가 없었기 때문이다. buffer 정책을 적용함으로서 평균적으로 33.20% 정도의 에너지 소비를 줄인 것은 onoff 정책을 적용했을 때의 30.96% 보다 향상된 에너지 소비 절감 효과를 보여주고 있다. buffer 정책과 onoff 정책을 비교해보면, 메모리 집중형에서는 5.86%, 프로세서 집중형에서는 1.34%, 평균적으로 3.60%의 에너지 소비 절감 효과를 보여준다. 프로세서 집중형에서보다 메모리 집중형에서 DVFS 정책의 에너지 소비 절감 효과가 향상되는 것을 확인할 수 있다.

표 4. nodvfs, onoff, buffer의 실행 시간 분석
Table 4. Execution Time : nodvfs, onoff, buffer

시간 (s)	png_ encode	zlib	diff	mem. avg.	
nodvfs	2.618300	1.813100	0.252200	1.561200	
onoff	2.889697	2.118303	0.288000	1.765333	
buffer	2.944544	2.144383	0.290762	1.793230	
onoff : buffer	0.018980	0.12312	0.009591	0.013628	
시간 (s)	png_ encode	lp_ solve	flex	proc. avg.	avg.
nodvfs	0.833800	0.812700	0.120900	0.589133	1.075167
onoff	0.906853	0.960589	0.133900	0.667114	1.216224
buffer	0.914037	0.988148	0.133789	0.678658	1.235944
onoff : buffer	0.007922	0.028690	-0.00083	0.011927	0.012778

표 5. nodvfs, onoff, buffer의 소비 에너지 분석
Table 5. Energy Consumption : nodvfs, onoff, buffer

에너지 (W)	png_ encode	zlib	diff	mem. avg.	
nodvfs	1.683000	1.202500	0.176700	1.020733	
onoff	1.017003	0.635543	0.108731	0.587092	
buffer	0.959632	0.591137	0.103328	0.551366	
onoff : buffer	-0.05641	-0.06987	-0.04969	-0.05865	
에너지 (W)	png_ encode	lp_ solve	flex	proc. avg.	avg.
nodvfs	0.642500	0.630900	0.094000	0.455800	0.738267
onoff	0.522025	0.500006	0.074200	0.365410	0.476251
buffer	0.509705	0.492401	0.074092	0.358732	0.455049
onoff : buffer	-0.02360	-0.01521	-0.00146	-0.01342	-0.03604

실험의 결과를 종합해보면 다음과 같다. buffer 정책은 onoff 정책에 비하여 에너지 소비 절감 측면에서 메모리 집중형의 경우 5.86%, 프로세서 집중형의 경우 1.34%, 평균적으로 3.60%의 에너지 소비를 절감시키는 효과가 있다. 프로세서 집중형의 경우에는 그 영향이 미미하지만 메모리 집중형의 경우에는 1차적으로 다른 DVFS 정책을 적용한 이후임에도 불구하고 5.86%의 추가적인 에너지 소비 절감 효과를 얻을 수 있었다. 표 4와 표 5에 이상의 실험 결과를 정리하였으며, 표의 항목 중 “onoff : buffer” 항목은 onoff 정책과 buffer 정책의 효과를 비교한 것이다.

V. 결론

본 논문에서는 DVFS에서 활용할 수 있는 새로운 프로세서 성능 평가 지표로써 메모리 접근 요청 버퍼의 사용량을 제안하고 그에 따른 새로운 DVFS 정책의 성능 향상을 소개하였다. 메모리 접근 요청 버퍼에 쌓여있는 메모리 접근 요청의 개수는 메모리 접근으로 인한 지연의 발생과 그에 따른 프로세서의 성능 저하를 예측할 수 있는 지표이며, 이는 DVFS의 효과에도 영향을 미친다는 사실을 실험을 통해 확인하였다. 이러한 메모리 접근 요청 버퍼의 사용량에 기반을 둔 DVFS 정책은 예측 시간의 범위가 좁기 때문에 보다 더 정교한 예측이 가능하며, 기존의 DVFS 정책을 보완하는 형태로도 사용되어 효율적인 결과를 얻을 수 있다. 기존의 DVFS 정책과 본

논문에서 제시하는 DVFS 정책을 비교하면 메모리 접근이 많은 벤치마크 프로그램의 경우 5.86%의 소비 에너지 감소 효과를 보여주고 있으며, 평균적으로는 3.60%의 소비 에너지 감소 효과를 보여주고 있다.

참고문헌

[1] Patterson, D.A., and Hennessy, J.L. "*Computer architecture: a quantitative approach*," Morgan Kaufman, 4th Edition, 2007.

[2] Gaurav Dhiman, Tajana Simunic Rosing, "Dynamic Voltage Frequency Scaling for Multi-Tasking Systems Using Online Learning," Proceedings of the 2007 International Symposium on Low Power Electronics and Design, ISLPED '07, Portland, OR, USA, August 27 - 29, 2007.

[3] Funaoka K., Takeda, A. et al., "Dynamic voltage and frequency scaling for optimal real-time scheduling on multiprocessors," Proceedings of International Symposium on Industrial Embedded Systems, pp. 27-33, 2008

[4] David C. Snowdon, Stefan M. Petters, and Gernot Heiser, "Accurate On-Line Prediction of Processor and Memory Energy Usage Under Voltage Scaling," Proceedings of the 7th ACM & IEEE international conference on Embedded software 2007, Salzburg, Austria, September 30 - October 3, 2007.

[5] Sebastian Herbert, Diana Marculescu, "Analysis of Dynamic Voltage/Frequency Scaling in Chip-Multiprocessor," Proceedings of the 2007 International Symposium on Low Power Electronics and Design ISLPED '07, Portland, OR, USA, August 27 - 29, 2007.

[6] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," Proceedings of the international symposium on low power electronics and design, pp. 38-43, 2007

[7] Intel XScale Core Developer's Manual (Available : *http://www.intel.com/design/intelxscale/273473.htm*)

[8] Kihwan Choi, Soma, R. and Pedram, M., "Fine-Grained Dynamic Voltage and Frequency Scaling for Precise Energy and Performance Tradeoff Based on the Ratio of Off-Chip Access to On-Chip Computation Times," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, January, 2005.

[9] G. Dhiman and T. S. Rosing, "Dynamic voltage frequency scaling for multi-tasking systems using online learning," Proceedings of International Symposium on Low Power Electronics and Design, pp. 207-212, 2007

[10] A. Cheng and Y. Wang, "A Dynamic Voltage Scaling Algorithm for Dynamic Workloads," Journal of Signal Processign Systems, Vol. 52, No. 1, pp. 45-57, 2008

[11] SimpleScalar LLC, *http://www.simplescalar.com*

[12] Zoltán Herczeg, Ákos Kiss, Daniel Schmidt, Norbert Wehn and Tibor Gyimóthy, "XEEMU: An Improved XScale Power Simulator," Proceedings of 17th International Workshop, Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation PATMOS 2007, Gothenburg, Sweden, September 3-5, 2007.

[13] Gilberto Contreras, Margaret Martonosi, Jinzhang Peng, Guei-Yuan Lueh and Roy Ju, "The XTREM Power and Performance Simulator for the Intel XScale Core: Design and Experiences," ACM Transactions on Embedded Computing Systems, TECS, Volume 6, Issue 1, February, 2007.

[14] GCC Code-Size Benchmark Environment, (Available : *http://www.inf.u-szeged.hu/csibe*)

저 자 소 개

곽 종 욱

1998 : 경북대학교 공학사
2001 : 서울대학교 공학석사
2006 : 서울대학교 공학박사
2006 - 2007 : 삼성전자 SOC 연구소 책임 연구원
2007 - 현재 : 영남대학교 컴퓨터공학과 조교수
관심분야 : 컴퓨터 구조, 임베디드 시스템, 그린 컴퓨팅

김 주 환

2001 : 서울대학교 공학사
2001 - 현재 : 서울대학교 석박사 통합과정
관심분야 : 컴퓨터 구조, 고성능 컴퓨팅