MPEG 동영상 재생기를 위한 윈도우 기반 동적 전압조절 알고리즘

(A Window-Based DVS Algorithm for MPEG Player)

서 영 선 † 박 경 환 †† 백 용 규 † 조 진 성 †††
(Youngsun Seo) (Kyunghwan Park) (Yonggyu Baek) (Jinsung Cho)

요 약 휴대용 단말기의 기능은 점차 더 높은 사양의 멀티미디어를 처리할 수 있도록 진화하고 있다. 특히, 고화질의 동영상과 게임 등을 지원할 수 있는 높은 성능은 사용자의 끊임없는 요구와 밀접하게 관련이 있다. 따라서 요구되는 높은 성능을 충족시키기 위한 휴대용 임베디드 시스템의 동작과정에서 전력소비는 기존보다 상대적으로 커지게 되었으며, 이를 효율적으로 관리하는 전력 관리 기법이 필요하게 되었다. 본 논문에서는 휴대용 미디어 플레이어에서의 동적 전압 조절 알고리즘을 제안하고자 한다. 제안하는 알고리즘은 최근 프레임의 정보와 실행 시간 등을 적절한 크기의 윈도우로 유지하며 이를 기반으로 프로세서가 지원하는 (주파수, 전압)레벨을 조절하여 프로세서의 전력소비를 낮추게 된다. 이 알고리즘은 간단한 모듈 형태로 구현이 되었으며 일반적인 동영상 재생기에 손쉽게 추가시킬 수 있다. 성능 측정은 실제 환경에서 많이 사용되고 있는 MPlayer를 사용하였으며 수행 결과, 최대 56%의 프로세서의 전력소비 감소 효과를 얻을 수 있었다.

키워드 : 동적전압조절, 휴대용 단말기, 연성 실시간 시스템

Abstract As the functionality of portable devices are being enhanced and the performance is being greatly improved, power dissipations of battery driven portable devices are being increased. So, an efficient power management for reducing their power consumption is needed. In this paper, we propose a window-based DVS algorithm for MPEG Player. The proposed algorithm maintains the recently frame information and execution time received from MPEG player in window queue and dynamically adjusts (frequency, voltage) level based on window queue information. Our algorithm can be implemented in the common multimedia player as a module. We employed well-known MPlayer for the measurement of performance. The experimental result shows that the proposed algorithm reduces energy consumption by 56% on maximal performance.

Key words : DVS(Dynamic Voltage Scaling), Portable device, Soft realtime system

· 본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터지원사업의 연구결과로 수행되었음(IITA-2008-(C1090-0801-0002))
· 이 논문은 2008 한국컴퓨터종합학술대회에서 '동영상 재생기를 위한 윈도우 기반 동적 전압조절 알고리즘'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 경희대학교 컴퓨터공학과
lamooss@mesl.khu.ac.kr
ica28@khu.ac.kr
†† 정 회 원 : (주)NHN 개발 사원
hicom@khu.ac.kr
††† 정 회 원 : 경희대학교 컴퓨터공학과 교수
chojs@khu.ac.kr

논문접수 : 2008년 8월 25일
심사완료 : 2008년 10월 23일

정보과학회논문지 : 시스템 및 이론 제35권 제11호(2008.12)

1. 서 론

최근의 휴대용 단말기의 기능은 점차 더 높은 사양의 멀티미디어를 처리할 수 있도록 진화하고 있다. 휴대폰(cellular phone), PDA, PMP(portable multimedia player), 스마트폰과 같은 휴대용 임베디드 디바이스들은 동영상과 게임 등 광범위한 멀티미디어 기능과 서비스를 제공한다. 또한 휴대용 임베디드 시스템은 단순하게 멀티미디어 서비스를 제공한다는 개념을 넘어 고화질, 고품질의 서비스를 지원할 수 있는 높은 성능을 갖추는 것에 관심을 가지고 있으며, 이는 보다 높은 성능의 휴대용 임베디드 시스템을 사용하고자 하는 사용자의 끊임없는 요구와 밀접하게 관련이 있다.

이러한 휴대용 임베디드 시스템은 주로 제한된 용량의 에너지를 가지고 있는 배터리로 동작하게 되어 있다. 하지만, 사용자의 요구가 증가할수록 더 높은 성능을 처리하기 위한 동작과정이 필요하며, 동영상과 게임 등과 같은 일정 시간 이상의 Play time을 요구하는 컨텐츠의 사용시간에 비례하여 배터리 전력소비는 기존보다 상대적으로 커질 수밖에 없다. 증가된 배터리 전력소비는 곧 배터리 수명이 빠르게 줄어들게 되는 것을 의미하며, 이러한 배터리 수명을 증대시키기 위해서 시스템의 전력 소모를 효과적으로 줄이기 위한 효율적인 전력 관리기법이 필요하다.

이미 시스템의 전력 소모를 줄이기 위한 방법으로 많은 연구가 이루어져 왔다. 대표적인 방법으로는 동적 전력관리 기법(Dynamic Power Management: DPM)과 동적 전압 조절(Dynamic Voltage Scaling: DVS)이 있다. 동적 전력관리 기법은 시스템을 구성하는 컴포넌트들의 사용 패턴을 기반으로 하여 특정 모듈이 장시간 혹은 일정 임계값 이상 사용될 계획이 없다면 전력 소비 모드를 비활성화 상태로 전이함으로써 프로세서의 전력소비를 줄이는 방법이다. 동적 전압조절 기법은 태스크가 수행이 될 때 프로세서에 인가되는 클록(clock frequency)과 전압(voltage)을 동적으로 조절하여 프로세서의 전력소비를 줄이는 방법이다. 일반적으로 프로세서의 전력소비는 공급전압의 제곱에 비례하고 인가되는 클록은 공급전압에 비례하는 특성을 가진 CMOS회로로 구성된 프로세서에서 DVS기법을 이용하면 효율적인 배터리 전력소비가 가능하다.

이 DVS기법은 많은 연구가 이루어져 왔으나 대부분은 경성 실시간 시스템(Hard Real-Time System)을 기반으로 하고 있고 또한 알고리즘 위주의 시뮬레이션이 대다수이므로 본 논문에서 고려하고자 하는 연성 실시간 시스템(Soft Real-Time System)에는 맞지 않다. 연성 실시간 시스템이란 데드라인(deadline)의 허용한계가 경성 실시간 시스템보다 다소 융통적인 멀티미디어 응용프로그램과 같은 시스템을 말한다. 기존의 연구는 실시간 시스템 모델을 기반으로 알고리즘이 기술되어 태스크의 실행시간을 정확하게 알고 있거나, 최악 실행시간을 대상으로 이루어진다. 그러나 동영상 재생기의 태스크는 주기적으로 수행되기는 하나, 수행시간이 프레임별로 매우 차이가 나기 때문에 기존 연구를 그대로 활용할 수가 없다. 본 논문에서는 실제 사용과 같이 휴대용 단말기에서 멀티미디어 응용과 같은 연성 실시간 시스템을 대상으로 효율적인 프로세서의 전력소비를 할 수 있는 동적 전압조절 알고리즘을 개발하고 구현하고자 한다.

제안하는 윈도우 기반 동적 전압조절 알고리즘은 최근 프레임의 정보와 실행 시간 등을 적절한 크기의 윈도우로 유지하며 이를 기반으로 CPU의 주파수와 전압 레벨을 조절하여 프로세서의 전력소비를 낮추게 된다. 이 알고리즘은 간단한 모듈 형태로 구현이 되었으며 일반적인 동영상 재생기에 손쉽게 추가시킬 수 있다. 제안된 알고리즘의 성능 측정 결과 최대 56%의 프로세서의 전력소비 감소 효과를 얻을 수 있었다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대한 내용을 기술한다. 3장에서는 본 논문에서 제안하는 윈도우 기반 알고리즘을 기술하고, 4장에서는 구현 및 실험을 통해 성능을 평가하고 분석한다. 마지막으로 5장에서는 결론을 내리고 향후 과제에 대해 기술한다.

2. 관련 연구

2.1 동적 전력관리 기법(DPM)

동적 전력관리 기법은 시스템의 구성 요소별로 사용된 패턴에 기반 하여 특정 모듈이 장시간 혹은 일정 임계값 이상 사용될 계획이 없다면 비활성화 상태로 전이하게 됨으로써 전력 소모를 줄이는 방법이다[1-8]. 동적 전력관리 기법이 적용되는 대표적인 시스템 모듈로는 프로세서, 메모리, 통신 인터페이스 등이 있다. 이들은 일반적으로 시스템 전체의 전력 소모 중 가장 큰 비율을 차지하고 있는 구성요소들이다. 따라서 이들의 상태 전이가 적절하게 이루어진다면 많은 에너지 효율을 얻을 수 있게 된다. 하지만 이렇게 시스템을 구성하는 다양한 모듈은 각기 다른 특성을 가지게 되고 이들의 사용 패턴을 정확히 예측할 수 없다면 상태 전이에 따른 전력 소모가 더 커지게 될 수 있으므로 어려움이 있다. 그러므로 시스템의 동작을 고려한 상태에서 상태 전이에 따르는 시간 및 에너지의 부대비용을 적절히 고려한 동적 전력관리 기법이 이루어져야 한다.

2.2 동적 전압조절 기법(DVS)

동적 전압조절 기법은 프로세서에 인가되는 전압을

동적으로 조절하여 프로세서의 전력소비량을 줄이는 방법이다. 보통 프로세서의 전력소비는 공급 전압의 제곱에 비례하고 또한 전압은 주파수와 비례하는 관계에 있다[3,6,9-21]. 그러므로 주어진 작업의 요구 조건을 만족하면서 전압과 주파수의 관계를 적절히 조절하면 프로세서의 전력소비를 효율적으로 줄일 수 있다. 일반적으로 동적 전압조절 기법은 실시간 작업들이 가지는 임계점 내에서의 여유 시간을 잘 분배하여 공급되는 전압과 주파수를 동적으로 변화하게 된다. DVS 기법은 크게 실시간 시스템과 비실시간 시스템 두 가지로 분류되어질 수 있다. 실시간 시스템에서의 DVS 기법은 태스크들이 가지는 여유 시간(idle time, slack time)을 정확히 예측하여 공급 전압과 주파수를 바꾸는 방법이 일반적이며 크게 Intra-Task DVS, Inter-Task DVS로 나뉠 수 있다[10-13]. 비실시간 시스템에서의 DVS 기법은 시스템에 대한 작업 부하량을 정확히 예측, 공급 전압과 주파수를 바꾸는 방법이 사용되며 이때는 프로세서의 작업량을 주기적으로 모니터링 하는 Interval-based DVS 기법이 있다[6,9,14-17,21]. 이 밖에도 프로세서 하나만을 고려하는 것이 아니라 (CPU, Memory), (CPU, WNIC), (CPU, LCD), (CPU, Battery) 등 프로세서와 다른 시스템 모듈 사이의 관계를 고려한 다양한 기법들이 존재한다[14,20].

2.3 Video-specific DVS

최근에는 멀티미디어 동영상을 위한 DVS 기법이 제안되고 있다[9,15,17]. Burchard와 Altenbernd는 두 단계에 걸쳐 프레임의 디코딩을 나누어 공급 전압과 주파수를 낮추는 방법으로, 프레임을 디코딩하기 위한 여러 정보들을 알고 있어야 하며 일반적인 동영상 재생기에 많은 수정을 가해야 하는 단점이 있다. Poiwelse는 프레임의 타입과 프레임의 크기로부터 실행시간을 측정하고 측정된 값을 기반으로 공급 전압과 주파수를 낮추는 방법이다. 이 방법 역시 실행 시간을 측정하기 위해서는 디코더에 수정을 가해야 하는 단점이 있다. Richard와 Gilles는 Kiosk(공공장소에 설치된 터치스크린 방식의 정보전달 시스템)과 같이 일정한 크기와 동일한 구간으로만 되어 있는 멀티미디어 동영상에서의 DVS 기법을 제안하였다[17]. 이 알고리즘은 동영상을 재생할 때 프레임의 실행 시간을 보고 공급 전압과 주파수를 변경하게 되며 변경되는 정보를 History에 저장하게 된다. kiosk의 특성상 일정한 구간의 정해진 동영상을 재생하는 것이기 때문에 한번 수행이 되고 나면 History에 DVS가 수행된 값이 저장이 되므로 다음번부터 실행할 때는 해당 값을 참조하여 그대로 수행하면 되므로 많은 전력소모 감소 효과를 얻을 수 있다. 다만 일반 사용자들이 이용하는 휴대용 단말기에서는 일정하게 정해진 동영상이 아니기 때문에 최적의 성능을 낼 수 없는 단점이 있다. 여기까지 관련 연구를 살펴보았다. 이를 바탕으로 본 논문에서는 실제 많은 사람들이 사용하는 휴대용 단말기를 대상으로 하며 멀티미디어 동영상에 특화된 DVS 알고리즘을 제안하고자 한다.

3. 윈도우 기반 DVS 알고리즘

본 논문은 군사무기, 공장의 작업 제어 시스템과 같이 주어진 마감시간(deadline)내에 반드시 작업을 처리해야 하는 경성 실시간 시스템(Hard Real-Time System)을 대상으로 하는 것이 아니라 멀티미디어 응용 프로그램과 같이 일정한 주기로 동작하면서 마감시간에 비교적 덜 제약적인 연성 실시간 시스템(Soft Real-Time System)에서의 동적 전압조절 알고리즘을 제안하고자 한다. 연성 실시간 시스템은 마감시간을 준수하지 못할 경우 작업의 결과의 가치가 일시적으로는 상실되나 시스템에 지속적으로 문제를 발생시키지 않는 특징을 가지고 있으므로 멀티미디어 응용 프로그램이 대표적인 예라 할 수 있다. 따라서 본 장에서는 수행되고 있는 멀티미디어 응용 프로그램의 품질을 최대한 보장하는 범위 내에서 프로세서의 전력소비를 최소화할 수 있는 윈도우 기반 DVS 알고리즘을 제안한다.

3.1 문제 정의

일반적으로 시스템에서 제공되는 DVS 가능한 프로세서는 정해진 범위의 주파수와 전압 레벨이 있다. 따라서 클록 주파수는 f_i, 인가되는 전압을 v_j라고 한다면 (f_i, v_j) 집합으로 나타낼 수 있다. 이 때 수행되는 태스크(동영상 재생 프로그램)를 T라고 한다면 하나의 프레임이 재생되는데 걸리는 시간을 T_{Exe}, 수행되고 있는 동영상의 프레임 당 주기는 T_{Period}라고 하며, 여유 시간 T_{Slack}은

$$T_{Slack} = T_{Period} - T_{Exe} \quad (1)$$

위의 식 (1)과 같이 표현할 수 있다. T_{Slack}은 프로세서가 수행되는 (f_i, v_j) 레벨이 낮을수록 0 에 가까워지거나 음수(프레임의 주기를 넘어서게 됨)가 된다. 따라서 사용자의 QoS를 만족하는 범위 내에서 프로세서의 전력소비를 최소한으로 줄이는 방법은 T_{Slack}이 음수가 되지 않으면서 제로에 가깝도록 하는 최적의 (f_i, v_j) 레벨을 찾아야 한다. 멀티미디어 동영상 프로그램의 특성상 매 프레임마다 크기가 달라지기 때문에 일정하게 (f_i, v_j) 레벨이 유지되지 않을 수 있으므로 동영상 재생동안 동적으로 지속적인 (f_i, v_j) 레벨을 구하여 적용하도록 한다.

따라서 본 논문에서는 표 1과 같이 문제를 정의한다.

표 1 WB-DVS 알고리즘 문제 정의

• 동영상이 재생되는 동안 사용자의 QoS를 만족하는 범위 내에서, $\sum_{i=1}^{k} T_{Slack[i]}\ (k = Total\ frame)$이 최소화 될 수 있도록 하는 최적의 (f_i, v_j) 레벨을 구한다.

3.2 제안하는 WB-DVS 알고리즘

제안하는 WB(window-based)-DVS 알고리즘은 최근까지 수행된 프레임의 정보(프레임이 출력되는 시간과 다음 프레임의 종류)를 일정한 크기의 윈도우 큐 (W= *Window Queue Size*)에 유지하여 그 값을 바탕으로 여유 시간 평균값(T_{Slack}^{Avg})을 구한다.

$$T_{Slack}^{Avg} = \sum_{i=1}^{W} (a_i \cdot T_{Slack[i]}) (a_i < a_{i+1},\ \sum_{i=1}^{W} a_i = 1) \quad (2)$$

$$a_i = i \cdot \frac{2}{W(W+1)} (W = Window\ Queue\ Size,\ W > 0) \quad (3)$$

여유시간 평균 값(T_{Slack}^{Avg})을 구하는 식 (2)에서는 가중치 a_i를 정의한 식 (3) 연산을 통해 일정한 크기의 윈도우 큐 중에서 가장 최근에 가까운 윈도우부터 선형적으로 가중치를 부여하였다. 식 (3)에서 i · 2/W(W+1)는 가중치의 합을 1이 되도록 하기 위함이며, 선형적으로 증가하는 a_i는 윈도우 큐의 각 윈도우에 가중치로 적용된다. a_i는 T_{Slack}^{Avg}을 구할 때 W 중 가장 최근의 값에 더 많은 가중치를 두어 계산 값의 정확도를 높이기 위한 변수이며 W 의 값은 가변적으로 변할 수 있기 때문에 실험 결과를 통해 적당한 값을 얻을 수 있다. 이 값을 통해 구해진 T_{Slack}^{Avg} 값은 초기에 정해진 임계값(Th_{Down}, Th_{Up})과 비교하여 (f_i, v_j) 레벨을 올릴지 낮출지를 결정하게 되며 이에 따라 임계값도 조절하게 된다. 또한 시스템에 무리가 가지 않도록 최소한의 프레임 Drop도 결정한다. 그림 1은 WB-DVS 알고리즘의 전체 동작에 대한 개요를 나타낸 것이고 그림 2는 윈도우 큐 크기가 3일 때 동작하는 모습을 간략하게 나타낸 그림이다.

그림 1을 보면 프레임이 수행될 때마다 프로세서의 속도(f_i, v_j)가 낮아져 T_{Slack}이 줄어드는 모습이 보이고 4 프레임에서는 거의 제로에 가깝게 동작한다. 이에 따라 수행되는 프로세서의 속도는 가장 낮은 1레벨에서 동작하므로 많은 전력 감소 효과를 가져 오게 된다.

물론 4 프레임 이후로 지속적으로 1 레벨에서 동작하리라는 보장은 없다. 일반적으로 동영상은 무수히 많은 GOP(group of pictures)로 구성되어 있고 GOP의 가장 앞 쪽부터 프레임의 비트량이 가장 많은 I frame으로 구성되어 있으므로 새로운 GOP가 시작되면 다시 프로세서의 속도를 증가시켜야 하며, 하나의 GOP안에서도

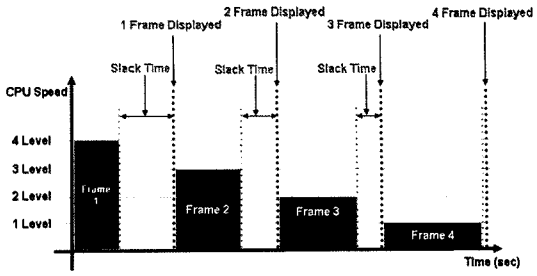

그림 1 WB-DVS 알고리즘 개요도

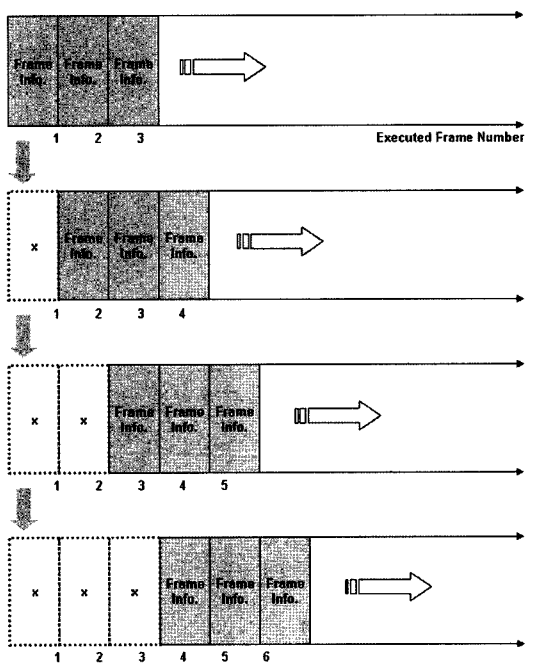

그림 2 윈도우 큐의 동작 예제(W=3)

프레임의 비트량에 따라 프로세서의 속도를 적절하게 변화시켜야 QoS를 보장할 수 있다. 이때 비트량이 클수록 디코더의 수행 시간이 길어지므로 결과적으로 프레임이 화면에 출력되는 실행 시간과 비례한다고 할 수 있다.

그림 3은 이러한 멀티미디어 동영상의 특성으로 인해 (f_i, v_j) 레벨이 바뀌게 되는 모습을 예로 나타낸 것이다. 그림 3과 같이 하나의 GOP가 끝나고 새로운 GOP가 시작될 때 (f_i, v_j) 레벨이 올라간 모습을 볼 수 있다.

그러나 위 그림과 같이 동작할 때 발생될 수 있는 문제점은 너무 빈번한 프로세서의 속도 변화는 오히려 시스템의 성능을 악화시켜 사용자가 원하는 QoS를 보장할 수 없다는 것이다. 따라서 일정한 임계값(Th_{Down}, Th_{Up})을 두어 해당 범위 내에서 동작한다면 프로세서의

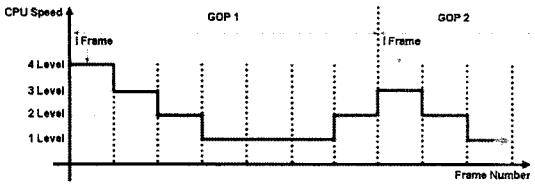

그림 3 WB-DVS 알고리즘 개요도

표 2 WB-DVS 알고리즘 입력 값

Input Value
① T_{Exe} (Frame Execution Time), $N_{frame\ Type}$ (Next Frame Type) ② $T_{Period} = \frac{1}{fps}$, ③ Current (f, v) Level

표 3 WB-DVS 알고리즘 출력 값

Output Value
① Adaptation (f, v) Level ② Deadline Miss

속도를 변화시키지 않으며 이것이 누적되어 동영상 재생에 무리를 일으킬 수 있다면 적절한 프레임 Drop을 통해 성능을 유지하도록 한다. 본 논문에서 제안하는 WB-DVS 알고리즘은 간단한 모듈 형태로 구현이 되기 때문에 손쉽게 일반적인 동영상 재생기에 추가할 수 있다. 하지만 알고리즘이 동작하기 위해 필요한 정보는 동영상 재생기로부터 넘겨받아야 한다. 이 값들은 동영상 재생기에서 꼭 필요한 정보이므로 기존 소스의 수정이 없이 넘겨주기만 하면 된다. 표 2는 이와 같이 넘겨받아야 하는 Input Value를 나타낸 것이고 표 3은 알고리즘이 동작한 후에 발생하는 출력 값을 나타낸 것이다.

제안하는 알고리즘의 동작은 동영상 재생시 매 프레임이 화면에 출력된 직후에 호출이 된다. 따라서 하나의 동영상이 수행될 때 알고리즘의 동작 횟수는 해당 동영상의 전체 프레임 수와 동일하다. 지금까지 기술한 본 논문의 알고리즘의 전체 동작은 표 4와 같이 나타낼 수 있다.

표 4의 WB-DVS 알고리즘 의사 코드에서 알고리즘 I은 여유시간 평균값(T_{Slack}^{Avg})을 구한다. 여유시간 평균값은 표 1, 식 (2), 식 (3)의 정의와 수식을 이용하여 구할 수 있다. 알고리즘 I 연산에 필요한 정보는 표 2의 WB-DVS 알고리즘 입력 값으로부터 얻을 수 있으며, 연산의 결과인 여유시간 평균값은 알고리즘 II와 알고리즘 III을 수행하는 파라미터로 사용된다. 알고리즘 II는 프레임 Drop을 결정하는 단계이다. $F_{Drop\ Number}$는 알고리즘 상에서 실험 수행 중에 Drop되는 프레임의 개수를 저장하는 카운트 변수를 의미한다. 여유시간 평균값(T_{Slack}^{Avg})이 음수이고 현재 프로세서에 인가되는 전압이 가장 높은 (f_i, v_j)레벨이라면 Drop이 발생하며, Drop되는 해당 프레임의 개수를 누적한다. 알고리즘 III는 적정한 (f_i, v_j)레벨을 정하는 단계이다. 여유시간 평균값이 양수이면서 임계값(Th_{Down})보다 크다면, 프로세서에 인가되는 (f_i, v_j)레벨을 1단계 낮추는 조건을 갖추게 되며, 이 때 $N_{frame\ Type}$(Next Frame Type)의 값에 I-frame이 올 경우 (f_i, v_j)레벨을 낮추지 않는 방법을 취함으로써 I-frame을 고려한다. $N_{frame\ Type}$이 I-frame이 아니면, Th_{Down}조정과 함께 (f_i, v_j)레벨이 1단계 낮추는 과정이 수행된다. 반대로 여유시간 평균값이 음수이면서 임계값(Th_{Up})보다 작다면, 프로세서에 인가되는 (f_i, v_j)레벨을 1단계 높이는 과정을 수행한다.

표 4 WB-DVS 알고리즘 의사 코드

I. Average Slack Time Computation Phase II. Frame Drop Decision Phase III. (f, v) Level Adaptation Phase	
I	$T_{Slack[i]} = T_{Period[i]} - T_{Exe[i]}$ $(0 \le i \le W)$ $T_{Slack}^{Avg} = \sum_{i=1}^{W} (a_i \cdot T_{Slack[i]})$ ($a_i < a_{i+1}$, $\sum_{i=1}^{W} a_i = 1$)
II	*If* ($T_{Slack}^{Avg} < 0$ and $Freq = Highest$) $F_{Drop\ Number} = F_{Drop\ Number} + 1$ *End If*
III	*If* ($T_{Slack}^{Avg} > 0$ and $T_{Slack}^{Avg} > Th_{Down}$) *If* ($N_{frame\ Type} \neq I_{frame}$) $Th_{Down} = (Th_{Down} + T_{Slack}^{Avg}) / 2$ $(f_i, v_j) = (f_i, v_j) - 1$ *End If* *Else If* ($T_{Slack}^{Avg} < 0$ and $T_{Slack}^{Avg} < Th_{Up}$) $(f_i, v_j) = (f_i, v_j) + 1$ *End If*

4. 구현 및 실험

4.1 구현 및 실험 환경

본 논문에서 제안하는 WB-DVS 알고리즘을 구현하기 위해 두 종류의 실험 환경을 구현하였다. 그림 4와 같이 PXA255 프로세서가 탑재되어 있는 보드환경과 실험의 신뢰도를 위해 노트북 환경에서도 본 논문이 제안하는 WB-DVS 알고리즘을 동일하게 적용하여 실험을 진행하였다.

그림 4의 좌측에 있는 보드는 저전력을 지원하는 인텔사의 PXA255 프로세서가 탑재되어 있고, 특별히 프

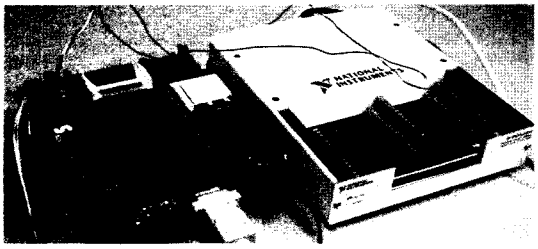

그림 4 실험 환경

표 5 구현 환경(Board)

Board	Tynux-Box xe Board (Palm Palm Tech.)
Processor	PXA255
Supported (f, v) Level	(99.5MHz, 1.0V), (199.1MHz, 1.0V) (298.6MHz, 1.1V), (398.1MHz, 1.3V)
OS	Embedded Linux 2.4.19
MPEG Player	MPlayer 0.90

표 6 구현 환경(Laptop)

Processor	Intel Mobile Centrino 900MHz
Supported (f, v) Level	(600MHz, 0.844V), (800MHz, 0.988V) (900MHz, 1.004V)
OS	Fedora core 5 (Version: 2.6.16.1)
MPEG Player	MPlayer 0.90

로세서의 전력소비를 측정하기 위한 핀이 주요 모듈에 장착되어 있어 그림 4의 우측에 있는 DAQ(data acquisition) 장비를 사용해 프로세서의 전력소비 측정이 가능하다. 프로세서가 지원하는 실제 (f_i, v_j) 레벨은 4단계이며, 표 5, 6은 보드와 노트북에서의 본 논문에서 제안하는 알고리즘의 구현환경에 대한 정보이다.

또한 제안한 알고리즘의 구현 모듈을 추가할 동영상 재생기는 신뢰성을 높이기 위해 많은 사용자들이 사용하는 MPlayer-0.90 버전을 사용하였으며 소스코드 수정이 거의 없이 본 논문의 WB-DVS 알고리즘이 적용된 모듈을 추가하여 구현하였다.

4.2 실험 방법 및 결과 분석

제안하는 알고리즘의 성능 측정을 위해 National Instruments사의 NI DAQPad-6015라는 그림 4의 우측과 같은 데이타 수집 하드웨어 장비를 사용하였다. 이 장비는 Host PC에 USB 타입으로 연결이 되며, 프로세서가 사용하는 전력 값을 직접 측정하는 것이 아니라 Tynux Board에 전류가 공급되면 프로세서를 통과한 후에 측정이 되는 전압(전압 강하)을 실시간으로 저장하게 된다.

따라서 아래의 전력 측정 공식을 사용하여 프로세서가 사용한 전력소비량을 알아낼 수 있다.

$$P = I \times V = \frac{V_1 \times V_2}{R} \quad (V_1: \text{공급 전압}, V_2: \text{출력 전압}) \quad (4)$$

R은 프로세서에 달려있는 저항의 값이며 V_1의 값은 프로세서에 인가되는 공급 전압이므로 저항을 거쳐 강하된 V_2(출력 전압)을 알 수 있다면 P의 값을 측정할 수 있다. 성능 측정을 위해 사용된 동영상 클립은 표 8에 나타내었듯이 60초의 동영상 클립을 4개의 fps (Frame Rate : frame per second)로 만들었으며 이 동영상 클립을 이용하여 보드와 노트북 환경에서 실험하였다.

실험방법은 우선 알고리즘이 적용되지 않은 일반 MPlayer-0.90의 소비 전력을 구하였고, 제안한 알고리즘에서 T_{Slack}^{Avg} 값을 계산할 때 가중치를 적용하는 식 (3)의 비교를 위해 $a_i = \frac{1}{W}$라는 평균값을 적용해 따로 측정하도록 하였다.

또한 알고리즘이 가장 우수한 성능을 발휘할 수 있는 윈도우큐($W = Window\,Queue\,Size$) 크기를 알기 위해 윈도우큐(W)의 크기를 1부터 13까지 차례로 변화시키면서 실험을 하였고, 알고리즘의 성능 검증을 위해 표 7과 같이 측정 기준과 측정 변수를 분류하여 윈도우큐 사이즈에 따른 프로세서의 전력소비와 표 8에서의 Frame Rate(fps)와 같이 일정한 크기의 변화를 가진 4개의 fps 프레임의 수에 따른 윈도우큐 사이즈별 프로세서의 전력소비 결과를 분석하고 최적의 프로세서의 전력소비를 나타내는 각 fps에 따른 윈도우큐 사이즈를 통해 본 논문이 제안하는 WB-DVS알고리즘의 성능을 살펴보고 검증한다.

표 7 측정 기준 및 측정 변수의 분류

① Power Consumption of each MPEG-Clip (10, 15, 20, 25 fps)
② Number of (f, v) Level change
③ Make a comparison between Non-DVS and WB-DVS

표 8 실험에 사용된 동영상 기본 정보

Name	Superman-Returns.avi			
Frame Rate(fps)	10	15	20	25
Total Frames	611	916	1221	1525
Data Rate(Kbps)	336	376	424	464
Encoder	Divx 4(MPEG-4)			
Screen Size	320 x 240			
Play Time	60 seconds			

4.2.1 프로세서의 소비전력 측정 결과

그림 5는 superman-returns 동영상을, 알고리즘을 적용하지 않은 일반 MPlayer에서의 소비전력 측정값, 그리고

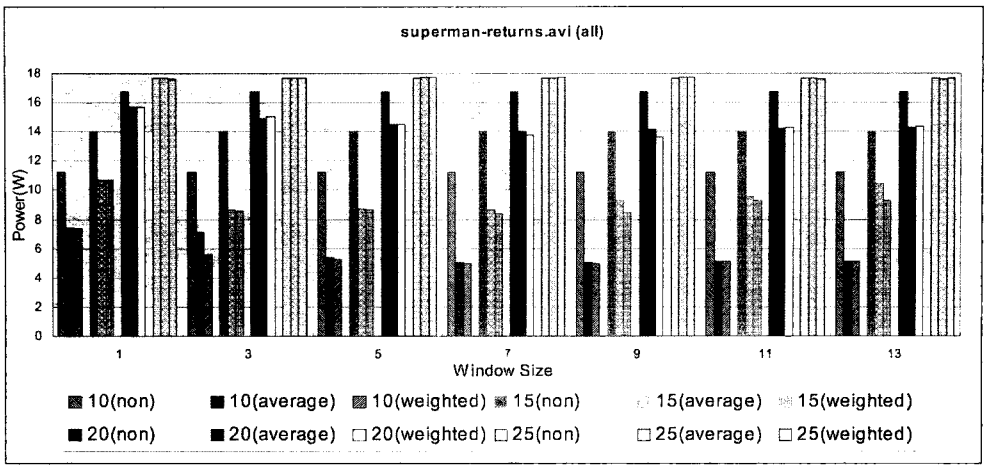

그림 5 보드에서의 동영상 소비 전력 측정값

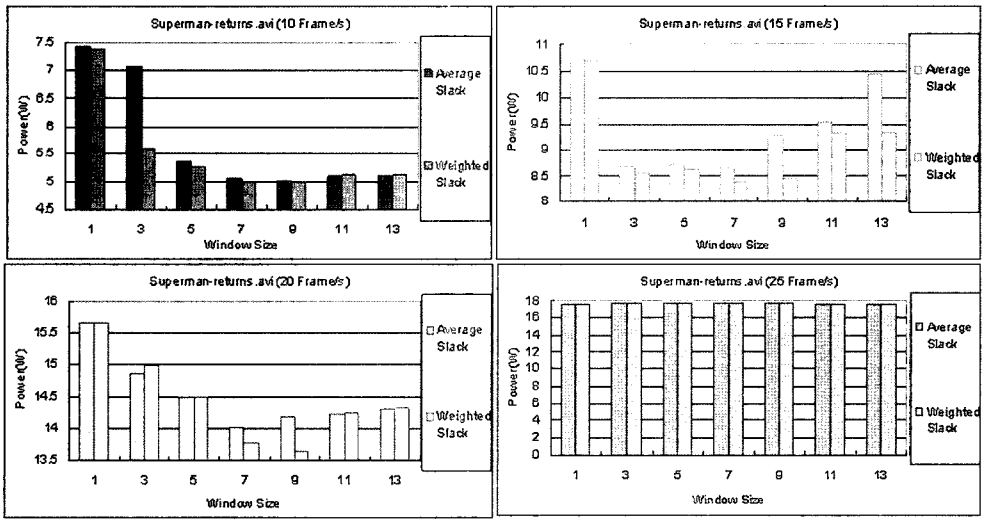

그림 6 보드에서의 fps별 소비 전력 측정값

제안한 알고리즘에서 $a_i = \frac{1}{W}$ 일 때와 $a_i = i \cdot \frac{2}{W(W+1)}$ 일 때의 각각의 소비전력 값을 계산한 그래프이다. 측정된 값에 의하면 모든 경우에서 W 가 약 5에서 7정도 크기일 때 가장 소비전력이 낮음을 알 수 있다. 10 fps의 경우 W 가 7이고 가중치를 적용한 a_i를 사용했을 경우 일반 MPlayer보다 최대 56%의 전력소모 절감효과를 보이며, 20 fps에서 일반 MPlayer보다 17%의 전력소모 감소효과를 얻을 수 있었다. 주목할 만한 점은 일반 MPlayer에서는 25 fps 동영상을 수행할 때 비디오와 오디오의 싱크가 맞지 않아 종료시 약 5초 정도의 차이를 보인다는 것이다. 이는 하드웨어의 성능이 25 fps 동영상을 재생시키는 데 무리가 있다는 것을 나타내며, 본 논문에서 제안한 알고리즘을 적용시키면 25 fps 동영상도 무리 없이 재생이 가능하다. 그 이유는 표 4의 3단계에서 II.Frame Drop Decision Phase에 의해 알고리즘 내부에서 여유 시간이 지속적으로 초과할 때에는 프레임 dropping이 일어나기 때문이다. 그림 6은 측정된 결과값을 fps별로 구분한 것이다.

그림 6에서 25 fps를 제외하고 소비전력 값이 W 가 약 7을 기점으로 감소하다가 다시 증가하게 된다. 이 결과 값으로 알 수 있는 것은 W 의 크기가 무조건 큰 것이 성능 향상에 도움이 되지 않으며, 소비전력이 증가한다는 것은 W의 크기가 증가함으로 인해 최적의 프로세서 속도 레벨을 찾지 못한다는 얘기와 동일하다.

이 결과는 추가적으로 실시한 노트북 환경에서의 실

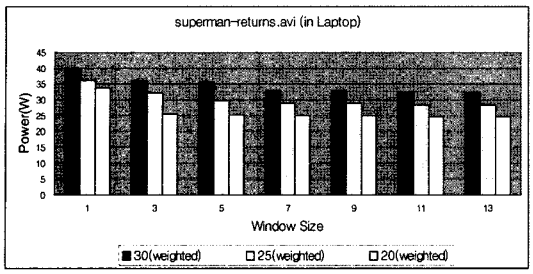

그림 7 노트북에서의 동영상 소비 전력 측정값

험 결과에도 유사한 결과를 보인다. 그림 7의 노트북에서 식 (3)을 적용한 소비 전력 측정값 결과에서도 Window Size가 7일 때 가장 소비 전력이 낮음을 볼 수 있다. 이는 그림 5와 그림 7에서 동일한 결과를 나타내고 있다.

4.2.2 프로세서 (f, v)레벨 변경 측정 결과

그림 8은 제안한 알고리즘이 동작할 때 프로세서의 (f, v) 레벨 변경이 얼마나 이루어졌는지 결과를 나타낸다. 25 fps의 경우 (f, v) 레벨의 변경은 한 번도 이루어지지 않았다. 25 fps 자체가 일반 MPlayer로 수행시켰을 때에도 비디오와 오디오의 동기화가 맞지 않는 높은 fps를 보이므로, 프로세서의 속도를 변경하는 대신 프레임을 일정 구간마다 지속적으로 dropping하여 무리 없이 재생이 가능하도록 하였기 때문이다. 10, 20 fps의 경우 W 가 증가할수록 변경 횟수가 줄어들며 7 이후에는 거의 비슷한 형태를 보이게 된다. 15 fps의 경우 낮아지다가 다시 변경 횟수가 증가하게 되는데, 계산 값의 정확성을 위해 식 (3)의 W 중 가장 최근의 값에 더 많은 가중치(weight) 적용한 알고리즘이 식 (2)의 결과보다 윈도우 사이즈가 7이상에서의 프로세서 (f, v) 레벨 변경 횟수의 증가폭이 더 적음을 알 수 있다. 이와는 반대로 4.2.1절에서의 소비전력 결과를 보면 알 수 있듯이 W 가 7 이후에는 프로세서의 전력소비량이 증가하게 된다. 그 이유는 W의 사이즈가 크다는 것은 그만큼 버퍼의 크기를 크게 잡는 것을 의미하며, W의 증가로 부정확하게 계산된 여유 시간(T_{Slack}) 때문에 낮은 단계로

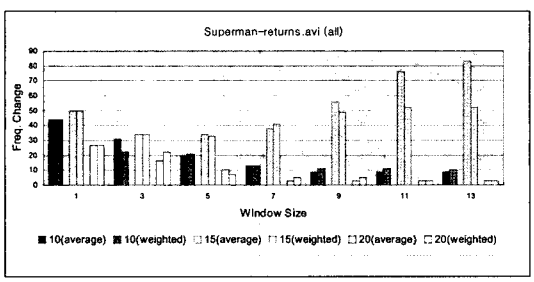

그림 8 보드에서의 (f, v)레벨 변경 측정값

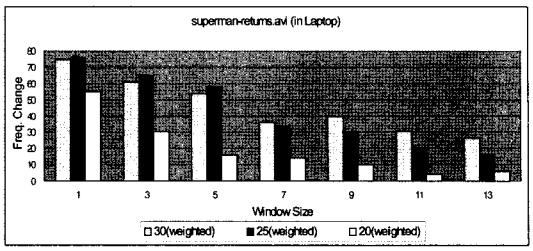

그림 9 노트북에서의 (f, v)레벨 변경 측정값

감소하였다가 다음과 그 다음 프레임에 연속적으로 (f, v) 레벨을 올리는 등 오히려 불필요한 레벨 변경으로 인해 프로세서의 전력소비를 증가시키기 때문이다.

그림 9는 노트북에서 (f, v) 레벨 변경이 얼마나 이루어졌는지 결과를 나타낸다. 3종류의 fps 동영상 파일에 모두 식 (3)의 가중치(weight)를 적용한 결과이며, 그림 8에서와 같이 윈도우 사이즈 7일 때 최적의 프로세서 (f, v) 레벨 변경 횟수를 나타낸다.

4.2.3 Deadline Miss 측정결과

본 논문이 제안한 알고리즘이 동작할 때 프레임의 주기를 얼마나 넘었는지에 대한 deadline miss 횟수를 측정하였다. 이때, deadline은 프레임의 주기, 즉 T_{Period}가 되며 식 (1)의 계산에서 프레임의 실행시간(T_{Exe})이 T_{Period}를 넘을 경우 $T_{Slack} < 0$ 이 되므로 deadline miss가 된 것으로 간주한다. 일반적으로 동영상을 재생할 때 fps가 높을 경우 fps에 거의 근접하게 약간 넘거나 넘지 않으면서 동영상을 수행시킨다. deadline miss 발생과 그 횟수의 증가는 QoS의 저하를 유발하지만, 본 논문에서 제안하는 WB-DVS 알고리즘이 적용된 시스템을 실험한 결과 발생하는 deadline miss의 90% 정도가 10ms 이내로 측정되었으며, 실제로 동영상 플레이어의 재생시에 시청각적으로 동영상 재생에 큰 문제를 발견할 수 없었다. 이런 사실을 감안하여 그림 10의 10 fps를 분석하면 W 가 약 7일 경우를 기점으로 증가하다가 일정해짐을 볼 수 있다. 그림 8에서 10 fps의 (f, v) 레벨은 윈도우 사이즈가 7 기점까지 감소하다가 일정해지는 것을 살펴보면, 여유시간이 deadline을 약간 넘어서고 넘지 않으면서 적정선의 (f, v)레벨을 찾았기 때문에 나타나는 결과이다. 이것으로 다른 fps들의 그래프 값의 결과의 타당성을 알 수가 있다.

4. 결론 및 향후 연구과제

최근 멀티미디어를 제공하는 휴대폰, PDA와 같은 휴대용 임베디드 시스템이 널리 사용되고 있다. 멀티미디어의 특성상 동영상, 게임 등의 컨텐츠는 일정 시간 이상의 Play time을 요구하며, 이에 따라 배터리를 기반

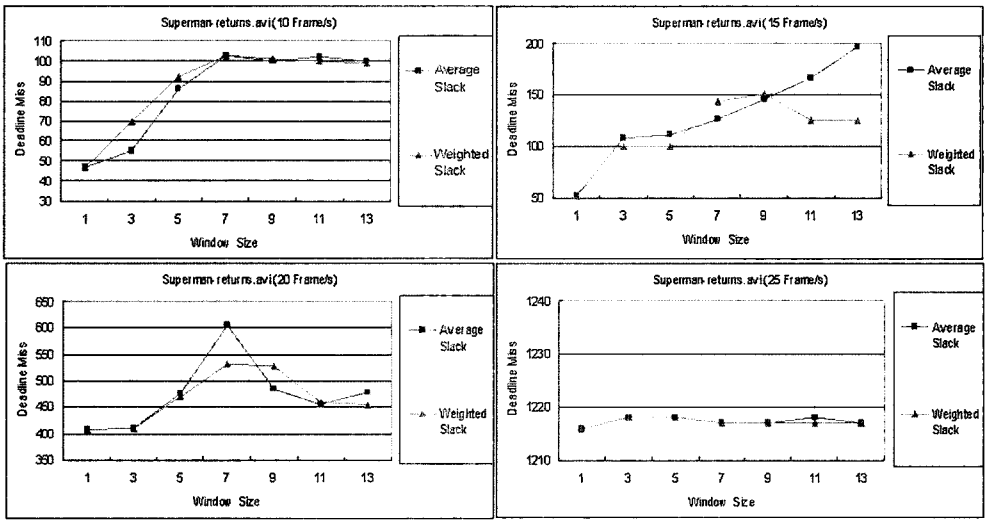

그림 10 보드에서의 Deadline Miss 측정 결과

으로 동작하는 휴대용 임베디드 시스템의 전력 관리는 최근 중요한 문제로 대두되고 있다. 이에 따라 본 논문에서는 경성 실시간 시스템(Hard Real-Time System)을 위주로 하는 다소 실제 휴대용 시스템에 적용되지 않는 기존 연구를 바탕으로 일정한 주기로 동작하면서 마감 시간에 비교적 덜 제약적인 연성 실시간 시스템(Soft Real-Time System)에 해당하는 멀티미디어 동영상 재생기에서의 동적 전압조절 알고리즘을 제안하였다. 제안한 WB(window-based)-DVS 알고리즘은 프레임의 타입과 실행 시간이라는 정보를 일정한 크기의 윈도우 큐(window queue)로 유지하면서 프레임이 가지는 주기에 최대한 근접하게 동작, 여유 시간을 제로에 가깝도록 최적의 프로세서 레벨을 찾아 적용한다. 이 알고리즘은 간단한 모듈 형태로 구현되어 있기 때문에 일반적인 동영상 재생기에 손쉽게 추가시킬 수 있다. 보드 환경과 노트북 환경에서의 실험을 통해 본 논문이 제안하는 알고리즘의 성능을 측정한 결과, 최소 17%, 최대 56%의 프로세서의 전력소비 감소효과를 얻을 수 있었다. 본 논문의 향후과제로는 프로세서뿐만 아니라 LCD, 메모리, 통신 인터페이스 등 시스템에서 주요한 모듈들의 전력 소모를 줄이기 위해 동적 전력관리 기법과의 효과적인 연계 방안을 모색하여 운영체제 레벨에서의 전력관리 프레임워크를 구현하고자 한다.

참 고 문 헌

[1] L. Benini, A. Bogliolo, and G.D. Micheli, "A Survey of Design Techniques for System-Level Dynamic Power Management," IEEE Trans. on Very Large Scale Integration Systems, Vol.10(2), pp. 299-316, 2000.6.

[2] B. Brock and K. Rajamani, "Dynamic Power Management for Embedded Systems," Proceedings of the IEEE SoC Conference, 2003.

[3] T. Simunic, L. Benini, A. Acquaviva, P. Glynn, G. Micheli, "Dynamic Voltage Scaling and Power Management for Portable Systems," Proceedings of the 38th Conference on Design Automation, pp. 524-529, 2001.

[4] D. Bertozzi, L. Benini, B. Ricco, "Power Aware Network Interface Management for Streaming Multimedia," Proceedings of Wireless Communications and Networking Conference, Vol.2, pp. 926-930, 2003.

[5] L. Benini, G. Castelli, A. Macii, R. Scarsi, "Battery-Driven Dynamic Power Management of Portable Systems," International Symposium on System Synthesis, pp. 25-33, 2000.9.

[6] 임성수, 신동윤, "리눅스 기반 이동통신 단말의 설계 및 구현", SK Telecommunications Review, 15권4호, pp. 573 584, 2005.8.

[7] 민정희, 차호정, "WiFi기반 모바일 임베디드 시스템을 위한 통합 전력 제어 기법", 정보과학회논문지, 컴퓨터 시스템, 2006.9.

[8] 이원규, 황선영, "실시간 시스템에서 효율적인 동적 전력 관리를 위한 태스크 스케줄링 알고리듬에 관한 연구", 한국통신학회 논문지 31권4A호, pp. 393-401, 2006.4.

[9] C.S. Im, S.H. Ha, and H.S. Kim, "Dynamic Voltage Scheduling with Buffers in Low-Power Multimedia Applications," ACM Trans. on Embedded Computing Systems, pp. 686 705, 2004.

[10] D. Shin, J. Kim, "Intra-task voltage scheduling on DVS-enabled hard real time systems," IEEE Trans, Computer-Aided Design, Vol.24, No.10, pp. 1530-

1549, 2005. 10.
[11] R. Xu, D. Mosse, "Minimizing Expected Energy in Real-Time Embedded Systems," EMSOFT, pp. 19-22, 2005.
[12] V. Rao, G. Singhal, A. Kumar, "Real Time Dynamic Scaling for Embedded Systems," International Conference on VLSI Design 17th, pp. 650-653, 2004.
[13] R. Jejurikar, R. Gupta, "Dynamic Voltage Scaling for Systemwide Energy Miminization in Real-Time Embedded System," International Symposium on Low Power Electronics and Design, pp. 78-81, 2004.
[14] Y.J. Kim, J.H. Kim, "Exploration of Memory-Aware Dynamic Voltage Scheduling for Soft Real-Time Applications," IEEE International Conference on RTCSA, pp. 170-180, 2005.
[15] B. Lee, E. Nurvitadhi, R. Dixit, C.S. Yu and M.C. Kim, "Dynamic Voltage Scaling Techniques for Power Efficient Video Decoding," JSA: the EURO-MICRO Journal, pp. 633-652, 2005.
[16] W. Yuan, K. Nahrstedt, "Practical Voltage Scaling for Mobile Multimedia Devices," Proceedings of the ACM international Conference on Multimedia, pp. 924-931, 2004.
[17] R. Urunuela, G. Muller, J. Lawall, "Energy Adaptation for Multimedia Information Kiosks," EMSOFT'06, pp. 22-25, 2006.10.
[18] J. Yu, W. Wu, X. Chen, H. Hsieh, J. Yang, "Assertion-Based Design Exploration of DVS in Network Processor Architectures," Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, 2005.
[19] 연세대학교, "커널 모니터링에 기반한 저전력 운영체제 기법 개발", 정보통신기초기술연구과제 04-기초-065.
[20] 최진욱, 차호정, "이동 단말기를 위한 전력 소모 관리 기법", SK Telecommunications Review, 15권4호, pp. 626-637, 2005.8.
[21] 김웅걸, "준연성 실시간 시스템을 위한 동적 전압 조절 기법", 서울대:공학석사학위논문, 2005.2.

서 영 선
2008년 용인대학교 컴퓨터정보학부 학사
2008년~현재 경희대학교 대학원 컴퓨터공학과 석사과정. 관심분야는 임베디드 시스템, 무선 센서 네트워크

박 경 환
2005년 경희대학교 대학원 컴퓨터공학과 학사. 2007년 경희대학교 대학원 컴퓨터공학과 석사. 2007년~현재 NHN corp. 관심분야는 임베디드 시스템, 임베디드 소프트웨어, 웹 프로그래밍

백 용 규
2007년 경희대학교 컴퓨터공학과 학사
2007년~현재 경희대학교 대학원 컴퓨터공학과 석사과정. 관심분야는 임베디드 시스템, 실시간 운영체제, 저전력 시스템

조 진 성
1992년 서울대학교 컴퓨터공학과 학사
1994년 서울대학교 대학원 컴퓨터공학과 석사. 2000년 서울대학교 대학원 컴퓨터공학과 박사. 1998년 IBM T.J. Watson Research Center Visiting Researcher. 1999년~2003년 삼성전자 책임연구원. 2003년~현재 경희대학교 컴퓨터공학전공 조교수. 관심분야는 모바일 네트워크, 임베디드 시스템