Journal of the Korean Institute of Illuminating and Electrical Installation Engineers Vol. 22, No.4, pp. 93~99 April 2008 논문 22-4-12

응용프로그램의 작업량을 고려한 임베디드 프로세서의 동적 전압 조절

(Dynamic Voltage Scaling based on Workload of Application for Embedded Processor)

왕홍문* · 김종태**
(Hong-Moon Wang · Jong-Tae Kim)

요 약

휴대용 기기의 다양한 기능으로 인해 에너지 절약은 더욱 중요한 문제가 되고 있다. Dynamic Voltage Scaling(DVS)는 임베디드 기기에서 대표적으로 사용되는 에너지 절약 방법이다. 본 논문에서는 응용프로그램의 작업량 변화에 따라 프로세서의 동작 전압과 속도를 조절할 수 있는 DVS 알고리즘을 제안한다. 제안된 DVS 알고리즘은 커널의 DVS 모듈과 응용프로그램의 작업량 변화를 관찰하는 함수로 구성되어 있으며 작업량이 급격히 증가하거나 감소하는 경우 이에 알맞은 프로세서의 동작 수준을 결정함으로서 작업의 데드라인을 넘기지 않으면서도 전력 소비를 줄일 수 있도록 하였다. 제안된 DVS 알고리즘은 Linux 2.6 커널과 PXA270프로세서를 이용한 임베디드 시스템에서 구현되었다.

Abstract

Portable devices generally have limited energy sources, so there is a need to minimize the power consumption of processor using energy conservation methods. One of the most common energy conservation methods is dynamic voltage scaling (DVS). In this paper, we propose a new DVS algorithm which uses workload of application to determine frequency and voltage of processors. The proposed DVS algorithm consists of DVS module in kernel and specified function in application. The DVS module monitors the processor utilization and changes frequency and voltage periodically. The other part monitors workload of application. With these two procedures, the processor can change the performance level to meet their deadline while consuming less energy. We implemented the proposed DVS algorithm on PXA270 processor with Linux 2.6 kernel.

Key Words : Dynamic voltage scaling, Workload

* 주저자 : 성균관대학교 전자전기컴퓨터공학과
** 교신저자 : 성균관대학교 정보통신공학부 교수
Tel : 031-290-7130, Fax : 031-299-4613
E-mail : jtkim@skku.ac.kr
접수일자 : 2007년 12월 20일
1차심사 : 2007년 12월 28일
심사완료 : 2008년 1월 11일

1. 서 론

임베디드 모바일 기기들의 특징은 배터리에서 전력을 공급받으며 과거에 데스크 탑에서 수행했던 작

업들을 수행하는 것이다. 현재 개발되고 있는 휴대기기의 경우 음성통신, 화상통신, 데이터 전송, 비디오 요청 서비스 등과 같은 다양한 기능들의 요구에 따라 더욱 고성능의 프로세서를 탑재하고 많은 전력을 소비한다. 하지만 프로세서가 항상 고성능으로 동작한다면 전력을 낭비하는 것이다. 고성능을 요구하는 작업과 천천히 수행해도 되는 작업을 구분하여 동작속도를 조절한다면 소비전력의 많은 부분을 절약할 수 있을 것이다.

Dynamic voltage scaling(DVS) 기술은 일정한 작업량에 필요한 에너지의 양은 프로세서에 공급되는 전압의 제곱에 비례하는 특징을 이용한다. 따라서 프로세서에 공급되는 전압을 낮추면 작업의 수행 시간은 늘어나는 것과는 반대로 필요한 에너지는 낮아진 공급 전압의 제곱에 비례하여 줄어든다[1].

임베디드 모바일 기기에서 수행되는 응용프로그램의 경우 대부분 사용자와의 인터페이스가 중요하기 때문에 어느 정도의 실시간성은 보장이 되어야 한다. 하지만 실시간성을 보장하기 위해 빠른 속도로 동작한다면 많은 전력 소비로 인해 동작가능 시간은 단축될 것이다. 반면에 전력 소비를 줄이기 위해 동작속도를 낮춘다면 응용프로그램의 수행 및 반응 시간이 늘어나 사용자가 불편을 느낄 수 있다. 따라서 응용프로그램을 수행함에 있어 전력 소비를 줄임과 동시에 데드라인을 넘기지 않기 위해서 임베디드 기기에서는 응용프로그램의 작업량에 따라 프로세서의 동작속도가 조절 가능한 DVS 알고리즘을 적용해야 한다.

따라서 본 논문에서는 응용프로그램의 동적인 작업량에 따라 프로세서의 동작 전압과 속도를 조절할 수 있는 DVS 알고리즘을 제안한다. 응용프로그램에서 작업량이 급격히 증가하거나 감소하는 경우 이에 알맞은 프로세서의 동작 수준을 결정함으로써 작업의 데드라인을 넘기지 않으면서도 전력 소비를 줄일 수 있도록 하였다. 제안된 DVS 알고리즘은 Linux 2.6 커널을 이용하여 PXA270 프로세서에서 구현되었다. 제안된 DVS 알고리즘과 기존의 알고리즘과의 비교를 위해 non-DVS 그리고 Vertigo[2], PEAK[3] 알고리즘과 비교하였다.

2장에서 전압의 조절에 대해 그리고 3장에서 응용프로그램의 작업량에 따른 DVS 알고리즘에 대해 설명하였다. 4장에 구현된 알고리즘의 실험 결과를 서술하였으며 마지막으로 5장에서 결론을 맺는다.

2. 전압의 조절

디지털 CMOS 프로세서의 동적인 전력 소비는 회로에 전하를 충전했다가 방전하는 작업의 반복이다. 정적인 전력 소비를 제외한 동적인 전력 소비는 아래와 같이 나타낼 수 있다.

$$P_{dynamic} = \sum_{k=1}^{M} C_k \cdot f_k \cdot V_{DD}^2 \quad (1)$$

M은 회로의 게이트의 수, C_k는 전하량, f_k는 스위칭 주파수 그리고 V_{DD}는 공급 전압을 나타낸다. 위의 식 (1)에서 보이는 바와 같이 프로세서의 소비전력은 V_{DD}의 제곱과 비례 관계에 있다. 따라서 V_{DD}를 낮추는 것이 CMOS 회로에서 가장 효과적으로 동적인 소비전력을 줄이는 결과를 가져온다. 그러나 낮은 전압은 신호의 전달 속도를 느리게 한다. CMOS 회로에서 V_{DD}를 낮추는 것은 전체 동작 속도를 느리게 하는 문제를 야기한다. 회로의 동작 지연은 아래와 같이 단순화 될 수 있다.

$$\tau \propto \frac{V_{DD}}{(V_G - V_T)^2} \quad (2)$$

τ는 회로의 전달 속도, V_T는 문턱전압, 그리고 V_G는 입력 게이트 전압이다[4]. 낮은 공급 전압으로 인해 발생한 전달 지연은 프로세서의 클럭 주파수에 제한을 가해서 프로세서가 낮은 속도로 동작해야 한다는 것을 뜻한다. 식 (1), (2)로부터 특정 작업에 대해 프로세서의 동작시간이 늘어나는 만큼 에너지의 소비는 V_{DD}의 제곱에 비례하여 절약될 수 있다는 것을 알 수 있다.

위에서 제시한 동작 주파수, 공급 전압, 그리고 소비전력의 관계는 표 1에서 확인할 수 있다. 표 1은 Transmeta TM5400 프로세서의 소비전력의 관계를 보여주고 있다[5]. 프로세서의 동작 주파수와 공급전

압이 낮아지면 동작시간이 늘어나는 대신 사용하는 에너지의 양이 줄어드는 것을 확인할 수 있다.

표 1. TM5400의 클럭 주파수와 공급전압의 관계
Table 1. Relative power of TM5400

Freqeuncy ([MHz])	Voltage ([V])	Relative Power ([%])
700	1.65	100
600	1.60	80.59
500	1.50	59.03
400	1.40	41.14
300	1.25	24.60
200	1.10	12.70

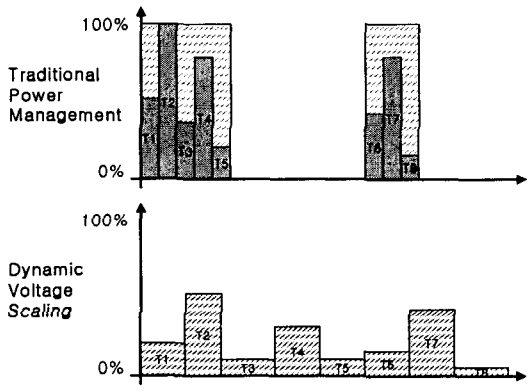

그림 1. 일반적인 전력관리 기법과 DVS의 전력 소비 비교
Fig. 1. Comparison of traditional power management and DVS

Dynamic voltage scaling(DVS)은 프로세서의 동작 중 동작 전압과 주파수를 변경함으로써 프로세서의 소비전력을 줄인다. DVS 기술은 시스템의 전체적인 시간 제약을 어기지 않고 동작속도와 공급전압의 적절한 조합을 통해 소비전력을 절약한다. 시스템이 필요로 하는 동작수준이 프로세서의 최대 동작수준보다 낮을 경우 동작속도와 공급전압을 낮춰 전력 소비를 최소화 할 수 있다. 그림 1에서 보는 바와 같이 수행해야 하는 작업이 T1~T8까지 있을 경우 전통적인 전력관리 기법은 모든 작업들을 같은 동작수준으로 수행하고 IDLE 상태일 때는 SLEEP 모드로 전환하여 전력 소비를 줄이는 방법을 사용한다. 반면에 DVS 기법의 경우 각 작업별로 필요한 동작속도와 공급전압을 변화시킴으로써 전통적인 전력관리 기법보다 적은 양의 전력을 소비하면서도 같은 양의 작업을 수행할 수 있다.

3. 작업량에 따른 DVS

임베디드 모바일 기기의 응용프로그램은 과거에 비해 고성능이 필요하고 외부의 환경에 따라 수행하는 작업의 양이 변경되는 경향이 있다. 예를 들어 외부 서버에서 비디오 전송을 통해 영화를 시청할 경우 일반적인 프레임의 양이 30[frames/s]라고 가정할 때 서버 또는 통신 채널에 발생한 병목현상으로 인해 수신되는 프레임의 양이 15[frames/s]로 현저하게 적어진다면 이 경우 30[frames/s]를 처리할 수 있는 속도로 프로세서가 동작하는 것은 전력의 낭비를 의미한다. 15[frames/s]일 경우 프로세서의 공급전압을 낮추고 낮은 속도로 동작시킴으로써 동일한 시간에 소비되는 전력을 줄일 수 있다. 또한 멀티미디어 데이터의 경우 처리해야 하는 정보의 양이 일정하지 않고 불규칙적으로 증가하고 감소한다. 그림 2는 MPEG 동영상의 프레임별로 디코딩되어야 하는 데이터 크기 변화를 보여주고 있다. MPEG의 경우 화면의 변경속도와 프레임의 종류에 따라 특정 시간에 처리되어야 하는 작업량이 불규칙하게 변화한다. 따라서 그림 2와 같은 멀티미디어 데이터를 처리할 때 프로세서의 IDLE 시간을 최소화하는 DVS 알고리즘을 사용하여 동작 속도를 낮출 경우 순간적으로 늘어난 데이터를 처리하지 못해서 프로세서는 해당 프레임의 데드라인을 넘기게 될 것이다.

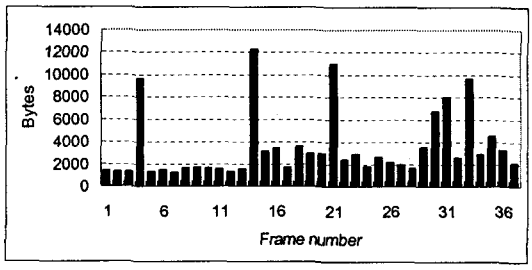

그림 2. MPEG 동영상에서 프레임의 데이터 크기
Fig. 2. Data size of MPEG video for each frame

위의 예에서 보인바와 같이 특정 응용프로그램이 필요로 하는 프로세서의 수행속도가 가변적이기 때문에 과거의 DVS 기술 보다는 응용프로그램의 작업량 변화에 따라 적응할 수 있는 DVS 기술이 더욱 에너지 효율적이고 서비스의 질을 떨어뜨리지 않는 방법일 것이다.

따라서 본 논문에서는 전력 소비의 효율을 높이기 위해 응용프로그램의 작업량에 따른 DVS 방식을 제안한다. 제안된 알고리즘은 DVS 모듈과 응용프로그램에서의 함수로 구성된다. DVS 모듈은 프로세서의 이용률을 주기적으로 관찰하여 프로세서의 동작 전압과 속도를 변경한다. 응용프로그램의 함수는 DVS 모듈에 의해 설정된 동작 전압과 속도를 작업량에 맞추어 수정하는 작업을 수행한다.

3.1 DVS 모듈

동작 전압과 주파수의 변경이 가능한 대부분의 프로세서들은 미리 정해진 몇 개의 동작 레벨만을 사용가능하도록 제공한다[5-6]. 사용가능한 동작 레벨마다 프로세서는 다른 양의 전력을 소비하며 같은 작업에 대해서도 *RUNNING*과 *IDLE* 시간에 차이를 보인다. *RUNNING*과 *IDLE* 시간을 이용하여 프로세서의 이용률을 얻기 위해 서로 다른 동작 레벨에서 측정된 시간을 최고 동작 레벨의 값으로 변화시킨다. 프로세서에서 수행된 작업의 시간(Ptime)은 아래의 식으로부터 구해진다.

$$Ptime = \sum_{i=1}^{n} t_i f_i \tag{3}$$

i는 프로세서의 여러 동작 레벨중의 하나를 뜻한다. t_i는 i 동작 레벨에서 작업의 수행시간이고 f_i는 i 동작 레벨의 최고 동작 레벨의 동작 주파수에 대한 비례 값이다. 식 (3)과 같이 동작 레벨별로 수집된 작업의 수행시간을 최고 동작 레벨에서의 값으로 변경함으로서 각각의 수행시간을 일괄적으로 더하는 방법보다는 정확한 결과를 얻을 수 있다.

일정한 시간(T)동안의 프로세서의 이용률(U_T)은 아래의 식 (4)와 같이 정의될 수 있다.

$$U_T = \frac{Ptime_{run}}{Ptime_{run} + Ptime_{idle}} \tag{4}$$

$Ptime_{run}$과 $Ptime_{idle}$은 프로세서의 *RUNNING*과 *IDLE* 시간을 나타낸다. 시간간격(T)는 임의로 정해진다. 그러나 프로세서의 이용률을 계산하는데 일정한 오버헤드가 존재하기 때문에 시간간격(T)가 일정 시간보다 짧으면 오버헤드가 크게 나타난다. 반면에 시간간격(T)이 너무 길면 DVS 알고리즘의 효율성이 낮아지게 된다.

```
Procedure of DVS module in kernel
Input: running and idle time for each performance
       level
U_i ← {u | u is pre-defined processor utilization
       range}
S_i ← {(f,v) | (f,v) is frequency and voltage for each
       performance level}
for each interval T
    calculate Ptime_running and Ptime_idle
    calculate U_T
    if (U_T is within U_i)
        next frequency and voltage = S_i
        change performance level

Procedure of DVS system-call
Input: ratio of next workload to average of workload
       (WL_t)
R_j ← {r | r is pre-defined ratio range}
M_j ← {t | t is the amount of transition of
       performance level}
for each system-call
if (WL_t is within R_j)
    next amount of transition = M_j
    adjust performance level

Procedure in user space
Input: next workload
for every workload
    calculate WLaverage_t
    calculate WL_t
    inform the DVS module of WL_t
```

그림 3. 의사코드
Fig. 3. Pseudo-code

DVS 모듈은 커널에 위치하여 프로세서의 동작 수준을 조절한다. 그림 3는 제안된 DVS 알고리즘의

의사코드를 보여주고 있다. DVS 모듈은 스케줄러에서 각 동작 수준별로 *RUNNING*과 *IDLE* 시간을 기록한다. 기록된 동작시간을 이용하여 시간간격(T)마다 주기적으로 $Ptime_{run}$과 $Ptime_{idle}$을 계산하여 식 (4)로부터 프로세서의 이용률을 얻는다. 계산된 프로세서 이용률은 다음 시간구간(T_{t+1})동안의 프로세서의 동작 수준을 결정하는데 사용된다. 프로세서의 동작 수준을 결정하기 위해 사전에 각 동작 수준별로 프로세서 이용률의 구간을 정의해 놓는다. 식 (4)로부터 구해진 프로세서 이용률이 특정 구간에 포함될 경우 그 구간에 정의되어 있는 동작 수준으로 프로세서의 동작 전압과 주파수를 변경한다. 위의 작업은 시간간격(T)마다 주기적으로 이루어지기 때문에 프로세서의 이용률을 일정 수준으로 유지하는 기능을 한다. 또한 결정된 동작 수준은 응용프로그램으로부터 제공되는 작업량의 변화를 적용하는 기준이 된다.

DVS 시스템 콜은 응용프로그램으로부터 제공되는 작업량에 관한 정보를 이용하여 프로세서의 동작 수준을 수정하는 작업을 수행한다. 응용프로그램으로부터 전달되는 정보는 작업량의 증가 혹은 감소 비율이다. 전달 받은 비율을 사전에 정의되어 있는 값과 비교하여 동작 수준의 변경 값을 결정한다. 변경 값이 결정되면 DVS 모듈에서 결정된 기준 동작 수준에서 변경 값만큼 프로세서의 동작 수준을 변경한다.

3.2 응용프로그램에서의 함수

응용프로그램에서는 DVS 시스템 콜을 이용하여 커널에 다음 작업량의 비율에 대한 정보를 전달한다. 작업량의 비율을 계산하기 위해 응용프로그램은 현재까지 수행한 작업량의 평균을 계산해야 한다. 현재시간(t)의 작업량의 평균($WLaverage_t$)은 식 (5)를 통해 구해진다.

$$WLaverage_t = \alpha \times WLaverage_{t-1} + (1-\alpha) \times WL_{t+1} \quad (5)$$

WL_{t+1}은 다음에 수행해야할 작업양이다. α는 과거 작업량의 평균과 다음 작업량에 대한 비중을 결정하는 인자 이다. α이 0.5보다 작으면 작업량의 평균($WLaverage_t$)은 과거 작업량의 평균($WLaverage_{t-1}$)보다 다음에 수행할 작업량(WL_{t+1})에 의해 더욱 영향을 받는다.

작업량의 평균에 대한 다음 작업량의 비율($WLrate_t$)은 식 (6)으로부터 구해진다.

$$WLrate_t = \frac{WL_{t+1}}{WLaverage_{t-1}} \quad (6)$$

위의 식으로부터 구해진 작업량의 비율($WLrate_t$)은 DVS 시스템 콜을 통해서 프로세서의 동작 전압과 속도를 변경하는데 사용된다. DVS 시스템 콜은 응용프로그램의 작업량이 순간적으로 증가했을 경우 전달 받은 작업량의 증가 비율을 통해 프로세서의 동작 전압과 속도를 변경함으로서 응용프로그램의 동작 시간을 보장한다.

그림 3의 응용프로그램에서 수행되어야 하는 작업의 의사코드와 같이 응용프로그램은 다음에 수행해야할 작업량을 이전에 수행한 작업량의 평균과 비교하여 그 비율을 커널의 DVS 모듈에 전달한다.

4. 실험 및 결과

제안된 DVS 알고리즘의 검증을 위한 응용프로그램으로 작업량의 변화가 외부에 의해 영향을 받는 스트리밍 비디오를 이용하였다. 실험에서 응용프로그램의 작업량은 스트리밍 비디오의 프레임 크기로 정의하였다.

프로세서의 동작 수준의 변경은 하드웨어의 상태를 변경해야 하기 때문에 시간이 소요되며 또한 전력을 소비한다. 잦은 동작 수준의 변경은 전체 시스템에 많은 오버헤드를 유발하기 때문에 DVS 모듈의 시간간격(T)은 적당한 값으로 선택되어야 한다. 인터럽트 및 외부의 신호에 빠르게 대응해야 하는 응용프로그램의 경우 시간간격(T)이 길면 반응시간이 느려지고 반면에 짧으면 오버헤드가 증가하기 때문에 본 실험에서는 시간간격(T)를 500[ms]로 하였다. 또한 응용프로그램의 작업량 평균에 영향을 미치는 (5)의 α는 0.5로 하였다.

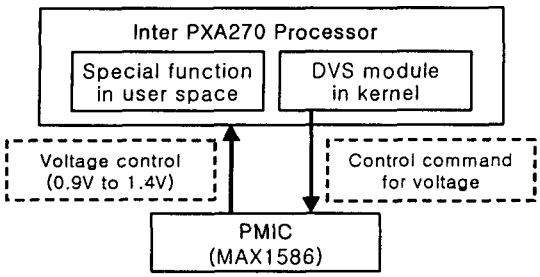

그림 4. 실험 환경
Fig. 4. Experiment environment

제안된 DVS 알고리즘은 PXA270프로세서를 이용한 임베디드 시스템에서 구현되었다. 그림 4는 실험을 위한 환경을 보여주고 있다. PXA270프로세서는 8개의 서로 다른 동작 수준을 이용하여 동작할 수 있다. 하지만 본 논문에서는 6개의 동작 수준만을 이용하였다. DVS 모듈은 리눅스 2.6 커널에 위치한다. 이 모듈은 프로세서 이용률을 이용하여 주기적으로 동작 수준을 결정하고 응용프로그램으로부터 전달받은 작업량의 비율을 이용하여 작업량의 변화에 따라 동작 수준을 변경한다. 프로세서 코어에 전원을 공급하는 전력관리 칩은 DVS 모듈에 의해서 제어된다. 공급 전압은 0.9[V]부터 1.4[V]까지 변경이 가능하다.

실험을 위해서 4가지 MPEG-4 비디오 파일을 준비하였다. 각 파일은 실시간 스트리밍 프로토콜에 의해 스트리밍 서버로부터 수신된다. 스트리밍 비디오는 MPlayer를 이용하여 재생되었고 제안된 DVS 알고리즘의 성능 비교를 위해 non-DVS 방식, Vertigo 그리고 PEAK 알고리즘과 비교하였다. 프로세서의 각 동작 수준에서 소비하는 전력은 [7]에 나타난 *RUNNING*과 *IDLE* 시간의 소비전력 값을 이용하였다.

표 2는 스트리밍 비디오 재생 시 발생하는 데이터 손실률을 non-DVS 방식을 기준으로 비교 작성한 것이다. 스트리밍 비디오의 재생은 작업량의 변화를 예측할 수 없고 그 변화량도 크기 때문에 데이터 손실률이 크게 나타나는 것은 모든 데이터의 수신과 처리가 이루어지지 않았다는 것을 뜻한다. 결국 데이터 손실률로 부터 DVS 알고리즘이 프로세서의 동작 수준을 결정하는데 응용프로그램의 동작 상황을 반영했는지를 확인할 수 있다. PEAK 알고리즘의 평균 데이터 손실률은 12.5[%]로 제안된 방식이 0.9[%]를 나타내는 것과 비교해서 많은 차이를 보인다. PEAK 알고리즘의 결과와 같이 데이터 손실률이 높을 경우 각 프레임의 디코딩 데드라인 안에 작업이 완료되지 못해서 응용프로그램의 동작 상황을 반영하지 못했음을 확인할 수 있다.

표 2. non-DVS 방식을 기준으로한 데이터 손실률
Table 2. Data loss rates normalized to the non-DVS method

비디오 종류	데이터 손실률([%])		
	Vertigo	Proposed	PEAK
Animation	0.3	1.3	14.1
Movie	0.1	1.6	13.5
Car racing	0.1	0.3	10.0
Sport	0.0	0.5	12.1
Average	0.1	0.9	12.5

표 3. 평균 소비 전력
Table 3. Average power consumption

비디오 종류	소비 전력([mW])		
	non-DVS	Vertigo	Proposed
Animation	353.0	316.7	304.2
Movie	351.1	306.4	304.2
Car racing	365.1	345.6	325.9
Sport	357.0	324.6	313.9
Average	356.6	323.3	312.0

DVS 방식별로 평균 전력 소비량은 표 3에서 보는 바와 같다. PEAK의 경우 손실된 데이터가 많아 소비 전력의 비교에서 제외하였다. 제안된 DVS 알고리즘은 평균 312[mW]를 소비해서 non-DVS 방식의 87.5[%]의 전력만으로 비디오를 재생한다. 이것은 Vertio 알고리즘과 비교해서 3.2[%] 성능이 높은 수치로서 제안된 DVS 알고리즘이 Vertigo 알고리즘보다 적은 에너지로 작업을 수행했음을 보여준다.

5. 결 론

제안된 DVS 알고리즘은 작업량이 불규칙적으로

변화하는 응용프로그램의 소비 전력을 줄이도록 설계되었으며 PXA270 프로세서와 Linux 2.6 커널을 이용한 임베디드 시스템에서 구현되었다. 제안된 DVS 알고리즘의 성능 분석을 위해 non-DVS 그리고 Vertigo, PEAK 알고리즘과 실험 결과를 비교하였다. 실험결과 제안된 DVS 알고리즘이 non-DVS 방식보다 12.5[%] 적은 에너지를 소비하였으며 Vertigo 알고리즘과 비교했을 때도 3.2[%] 적은 에너지를 소비하는 것을 보였다. 실험에 사용한 스트리밍 비디오의 데이터 손실률은 낮아진 프로세서의 수행 속도에 의해 응용프로그램의 수행 중 얼마나 많은 데이터가 처리되지 못했는지를 보여준다. PEAK 알고리즘의 경우 12.5[%]의 데이터 손실률을 보여 프로세서의 동작 수준의 결정이 응용프로그램의 동작을 반영하지 못했음을 보여주고 있다. 반면에 제안된 알고리즘은 데이터 손실률이 적으면서도 프로세서의 소비 전력을 줄이고 있음을 확인할 수 있다.

References

[1] J. Pouwelse, K. Langendoen and H. Sips, "Dynamic voltage scaling on a low-power microprocessor", International Conference on Mobile Computing and Networking, 2001.

[2] K. Flautner and T. Mudge, "Vertigo: automatic performance-setting for linux", Proceedings 5th Symposium on Operating Systems Design and Implementation, Boston, 2002.

[3] K. Govil, E. Chan, and H. Wasserman, "Comparing algorithms for dynamic speed-setting of a low-power CPU", Proceedings the First International Conference on Mobile Computing and Networking, California, 1995.

[4] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors", Proceedings of the International Symposium on Low Power Electronics and Design, Aug. 1998.

[5] Transmeta Corporation, "LongRun2 technologies", http://www.transmeta.com/tech/longrun2.html, visited on 19/9/2007.

[6] Intel Corporation, "Enhanced intel speedstep technology", http://www.intel.com/support/processors/mobile/pentium4/sb/CS-007499.htm.

[7] "Intel PXA270 processor electrical, mechanical and thermal specification", data sheet, Intel.

◇ 저자소개 ◇

왕홍문 (王鴻文)

1979년 12월 28일생. 2006년 성균관대학교 정보통신공학부 졸업. 2006년~현재 성균관대학교 일반대학원 전자전기컴퓨터공학과 석사과정 재학.

김종태 (金鍾兌)

1959년 11월 11일생. 1982년 성균관대학교 전자공학과 졸업. 1987년 University Of California at Irvine, 전기 및 컴퓨터공학과 대학원 졸업(석사). 1992년 University Of California at Irvine, 전기 및 컴퓨터공학과 대학원 졸업(박사). 1991~1993년 The Aerospace 연구원. 1993~1995년 전북대학교 컴퓨터공학과 교수. 1995년~현재 성균관대학교 전자전기공학과 교수.