Journal of Korea Multimedia Society Vol. 11, No. 5, May 2008(pp. 641-650)

전압 변경 오버헤드를 고려한 전력 관리 알고리즘

권혁성†, 안병철††

요 약

휴대 기기의 보급 확대에 따라 작은 배터리의 사용 시간 연장을 위해 배터리의 전력 관리 기법이 필요하다. 기존의 전력 관리 기법들은 전압 변경에 따른 오버헤드를 고려하지 않거나 부분적으로 고려하였다. 따라서 전압 변경 오버헤드는 개인용 멀티미디어 시스템의 실시간 태스크의 스케줄을 보장 못하는 경우가 발생할 수 있다. 본 논문은 전압 변경 오버헤드를 고려하면서 시스템의 이용률 정보를 바탕으로 사용 가능한 주파수의 개수와 연속된 주파수 사이의 간격을 조절하여 소비 전력을 줄이는 전력 관리 알고리즘을 제안한다. 본 알고리즘은 주파수 변경 회수를 줄여 기존의 CC RT-DVS 방법과 smoothing 방법에 비해 10에서 25% 정도 소비 전력을 절감할 수 있다.

A Power-Aware Scheduling Algorithm with Voltage Transition Overhead

Hyek-Seong Kweon†, Byoung-Chul Ahn††

ABSTRACT

As portable devices are used widely, power management algorithm is essential to extend battery use time on small-sized battery power. Although many methods have been proposed, they assumed the voltage transition overhead was negligible or was considered partially. However, the voltage transition overhead might not guarantee to schedule real-time tasks in portable multimedia systems. This paper proposes the adaptive power-aware algorithm to minimize the power consumption by considering the voltage transition overhead. It selects only a few discrete frequencies from the whole frequencies of a system and adjusts the interval between two consecutive frequencies based on the system utilization to reduce the number of frequency change. This algorithm saves the power consumption about 10 to 25 percent compared to a CC RT-DVS method and a frequency-smoothing method.

Key words: Low Power Management(저전력 관리), Real time System(실시간 시스템), Dynamic Voltage Scaling(동적 전압 조절 기법), Scheduling(스케줄링)

1. 서 론

최근 무선 인터넷 환경의 확산과 더불어 스마트폰 및 PDA(Personal Digital Assistant)와 같은 휴대용 시스템이 대중화 되고 있다. 이러한 시스템은 기존의 고유 기능 외에 멀티미디어 시청과 같은 복잡한 응용 프로그램을 수행한다. 장시간 사용하기 위해서 대용량의 전원 장치가 필요하지만, 배터리의 용량이 한정되어 배터리의 수명 연장을 위한 효율적인 전력 관리 기법이 필요하다. 전력 관리 기법 중 동적 전력 관리 기법(Dynamic Power Management)과 동적 전압 조절 기법(Dynamic Voltage Scaling)은 현재까지 많이 이용되고 있다[1,2].

동적 전압 조절 기법은 실행하는 프로그램의 동작

※ 교신저자(Corresponding Author) : 안병철, 주소 : 경북 경산시 대동 214-1(712-749), 전화 : 053)810-2556, FAX : 053)819-1976, E-mail : b.ahn@yu.ac.kr
접수일 : 2008년 2월 11일, 완료일 : 2008년 4월 16일

† 준회원, 영남대학교 컴퓨터 공학과 박사과정
(E-mail :) hskwon@ynu.ac.kr
†† 영남대학교 컴퓨터 공학과 교수

유형이나 특성을 파악하여 시스템의 이용률을 예측하고 이를 바탕으로 전압을 줄여 전력 소비를 줄이는 기법이다. 일반적으로 CMOS 회로의 전력 소비는 전원의 제곱과 주파수에 비례한다[3]. 현재 널리 사용되고 있는 프로세서들은 CMOS 회로를 많이 사용하고 있어서 동적 전압 조절 기법은 시스템의 이용률이 높지 않는 시스템에서는 소비 전력을 많이 줄일 수 있다[1].

연구 초기에는, 동적 전압 조절 기법을 이용한 전력 관리 기법은 주로 이상적인 프로세서 환경, 즉, 연속적인 전압을 공급하고 전압을 변화시킬 경우 실행 지연이 발생하지 않는 프로세서 환경 하에서 소비 전력을 줄이는데 초점을 맞추고 있다[1]. 그러나 이러한 전력 관리 알고리즘을 실 시스템에 구현할 때, 두 가지 제약 사항이 있다[4-6]. 하나는 프로세서에 공급되는 전압이 연속적이지 않고 이산적이라는 것이고 다른 하나는 전압을 변경할 경우 실행 지연과 불필요한 전력 소비가 발생한다는 것이다.

MPEG Player와 같은 실시간 특성을 가진 시스템에서도 전력 관리 알고리즘을 이용하여 소비 전력을 줄이는 많은 연구가 이루어 졌다[4,5,7-12]. [10]에서는 Static, Cycle-Conserving, Look-Ahead Real-time DVS 기법을 제안하였다. Cycle-Conserving DVS 기법은 응용 프로그램의 실시간 특성을 만족하기 위해서 최악 실행 시간을 고려하여 시스템의 유휴 시간을 충분히 활용할 수 없지만, 구현이 용이하여 많은 시스템에서 이용되고 있다[9]. Look-Ahead DVS 기법은 다른 기법에 비해 소비 전력을 많이 줄이지만, 알고리즘이 복잡하여 구현이 어려운 단점이 있다[9]. 이러한 초기 연구들은 주로 전압의 제약 사항만 고려하였다[7,9,10,11]. 그러나 이러한 알고리즘들은 전압 변경으로 인한 실행 지연을 고려하지 않아서 실시간 시스템에 적용할 경우, 마감 시간을 만족하지 못하는 경우가 발생할 수 있다[4,5]. 실시간 특성을 만족하기 위해서는 이산적 전압 공급과 더불어 주파수 변경에 따른 실행 지연을 고려한 전력 관리 알고리즘이 필요하다.

[5]에서는 동적 전압 조절 기법에서 전압을 변경할 경우, 세 가지의 오버헤드가 발생한다고 정의하였다. 이는 전압 변경을 위한 명령어 실행 시간, 전압 변경에 따른 실행 지연, 그리고 실행 지연으로 인한 전력 소비이다. 이를 바탕으로 오버헤드를 최소한 할 수 있는 TE-VAOpt 알고리즘을 제안하였다. 이 알고리즘은 그래프 이론을 바탕으로 하여 전압의 개수가 증가할 경우, 소비 전력을 최소하는 전압들을 찾는데 많은 시간이 소요된다. [12]에서는 태스크의 스케줄링 정보를 바탕으로 전압 변경에 따른 오버헤드를 최소화 하는 알고리즘을 제안하였다. 이는 전압 변경 횟수를 최소를 할 수 있도록 태스크의 실행 순서를 시스템에 영향을 주지 않는 범위 내에서 재배치한다. 그러나 각 태스크의 최악 실행 시간을 이용하므로 실 시스템에서 발생하는 유휴 시간(slack)을 이용할 수 없다는 단점이 있다. [4]에서는 프로세서에서 제공하는 전압의 개수가 많을 경우, 시스템의 이용률이 조금만 변화하더라도 전압을 변경해야 하는 문제를 전압 개수와 연속된 전압 사이의 간격을 조절함으로써 전압 변경 횟수를 줄이는 알고리즘을 제안하였다. 그러나 오버헤드를 일정하게 적용하여 실 시스템에 구현할 경우, 마감 시간을 만족하지 못하거나 불필요한 전력을 소비하는 문제가 발생할 수 있다.

따라서 전력 관리 기법을 실 시스템에서 구현하기 위해서는, 우선적으로 전압 변경에 따른 오버헤드를 고려해야 한다. 또한, 전력 소비를 최소화 할 수 있는 전압을 선택하기 위해 소요되는 시간이 적어야 운영체제에 부담을 줄일 수 있다. 본 논문에서는 이와 같은 사항을 고려하면서 내장 시스템에서 사용되는 평균 이용률의 차이가 크지 않다는 점을 이용하여 평균 이용률의 변화에 따라 사용 가능한 전압의 개수와 전압 사이의 간격을 조절하여 전압을 변경하는 회수를 최소화하여 전력 소비를 줄이는 알고리즘을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 사용될 시스템과 용어에 대하여 논한다. 3장에서는 본 논문에서 제안하는 전력 관리 알고리즘에 대하여, 4장에서는 3장에서 언급된 알고리즘의 성능을 평가하기 위한 모의실험 환경 및 결과에 대하여 논한다. 마지막으로 5장에서는 결론에 대하여 기술한다.

2. 시스템 모델

2.1 프로세서 모델

본 논문에서는 실 시스템의 환경을 최대한 반영하기 위해서 PXA270에서 제공하는 주파수 범위를 사

용한다[13]. 주파수는 식 (1)과 같이 n개의 이산적 주파수들의 집합으로 구성되고 식 (2)의 연속된 주파수 사이의 간격(Δf)은 같다고 가정한다. 주파수 f는 프로세서마다 달라질 수 있으므로, 각 주파수를 최대 주파수 f_n(또는 f_{max})에 의하여 정규화(normalization)하여 사용한다.

$$\{J_1, J_2, \cdots, J_n\} \ (J_k > J_{k-1}, 1 < k \le n) \tag{1}$$

$$= J_k - J_{k-1} \tag{2}$$

본 논문에서 사용하는 전력 모델은 현재 프로세서에서 많이 사용하는 CMOS 회로 특성을 고려한 전력 모델을 이용한다[3]. 프로세서에서 소비되는 전력은 주로 gate의 switching에 따른 동적 전력 소비 (dynamic power consumption)이다. 본 논문에서 소비 전력 평가는 동적 전력 소비만을 고려한다. 소비 전력은 식 (3)과 같이 전압(V_{dd})의 제곱에 비례하고 주파수(f)에 비례한다[3]. 주파수는 식 (4)와 같이 전압에 비례한다. 따라서 주파수를 낮추면 전압을 낮출 수 있고 이는 식 (3)에 따라 소비 전력을 급격히 줄일 수 있다. C_{ef}는 프로세서의 전체 커퍼시터 용량이고 k는 비례 상수이고, V_{th}는 문턱(threshold) 전압이다.

$$_{ynamic} = \kappa \cdot \cup_{ef} \cdot \bar{V}_{dd} \cdot J \tag{3}$$

$$f \propto \frac{(V_{dd} - V_{th})^2}{V_{dd}} \tag{4}$$

이상적인 주파수 f에 대한 전력 소비를 $P_{ideal}(f)$이라 하고 식 (1)에서 얻은 주파수 f에 의한 전력 소비를 $P_{pract}(f)$ 또는 $P(f)$라 한다. 이 두 전력의 차이(ΔE)는 식 (5)와 같이 유도된다.

$$^{\prime}{}_{ideal}(J) - {}_{pract}(J) \tag{5}$$

2.2 태스크 모델

본 논문에서 사용되는 태스크 모델은 주기를 가진 실시간 태스크 집합을 이용한다[14]. 태스크 집합은 $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$로 표기한다. 각 태스크는 주기($T$), 마감 시간($D$) 그리고 최악 실행 시간(WCET, worst case execution time)(C)로 구성되고, $\tau_i = \{T_i, D_i, C_i\}$로 표기 한다. 각 태스크는 동일한 주기를 가지며, 선점적(Preemptive)이고, 상호 독립적이다. 시간 t_0에서 시작되고 시작 시간과 마감 시간은 태스크의 주기와 일치한다고 가정한다. Hyper Period는 H로 표기하며, 모든 태스크 주기의 최소 공배수로 정의한다. 각 시스템의 성능 평가는 시간 t_0에서 H_n까지를 기준으로 한다. 시간 t에서의 시스템의 이용률(u_t)는 식 (6)을 이용하여 산출한다[10]. t_{si}와 t_{fi}는 각각 태스크 τ_i의 시작 시간과 마감 시간이다.

$$\sum_{i=1}^{t} \frac{}{T_i} = \sum_{i=1}^{t} \overline{t_f^i - t_s^i} \tag{6}$$

이상적인 시스템은 u_t를 연속적인 주파수로 환산하여 사용할 수 있으나 실 시스템에서는 이산적인 주파수를 사용해야 하므로 식 (7)을 만족하는 주파수를 찾는다.

$$\lceil u_t \rceil = \min\{f : f \in \{f_1, f_2, \cdots, f_n\} \ and \ f \ge u_t\} \tag{7}$$

이전 시스템 이용률에 대응되는 이산적 주파수와 식 (6)을 이용하여 산출한 시스템 이용률과의 차이(Δu)는 식 (8)을 이용하여 산출할 수 있다.

$$\Delta u = \lceil u_{t-1} \rceil - u_t \tag{8}$$

시간 t에서 사용될 주파수는 위의 유도된 식을 바탕으로 식 (9)를 이용하여 정할 수 있다.

$$f_t = \begin{cases} \lfloor u_{t-1} \rfloor, & \text{if } 0 \le \Delta u < \Delta f, \\ \lceil u_t \rceil, & \text{otherwise.} \end{cases} \tag{9}$$

현 시스템의 이용률과 이전에 사용된 주파수의 차이가 주파수 간격(Δf)보다 적으면 이전에 설정된 주파수를 이용할 수 있지만, 차이가 Δf보다 크다면 새로운 주파수로 변경해야 한다. 본 논문에서는 주파수가 변할 경우 전압도 같이 변한다고 가정한다.

3. 적응적 전력 관리 기법

본 논문에서는 주파수를 평활(smoothing)하게 관리하여 주파수를 선택할 때 시스템의 이용률과 이산적인 주파수 사이의 차이로 인한 전력 손실과 주파수 변경에 따른 실행 지연에 의한 전력 손실을 줄이는 기법을 제안한다. 또한 시스템의 평균 이용률 변화에 따라 사용 가능한 전압의 개수와 전압 사이의 간격을 조절하여 주파수 변경 횟수를 줄일 수 있다. 이를 적

응적 전력 관리 기법(Adaptive Power-Aware Algorithm)이라 한다.

3.1 주파수 변경에 따른 전력 소비

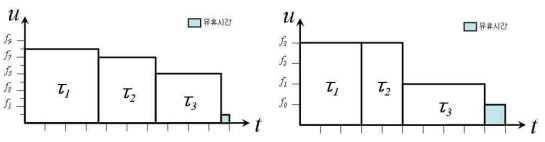
동적 전압 관리 기법을 실 시스템에서 구현할 경우 시스템의 제약 사항에 대한 고려가 필요하다. 그림 1과 그림 2는 주파수를 변경할 경우 발생하는 실행 지연과 이로 인한 전력 소비를 고려하지 않을 경우와 고려한 경우의 시스템 이용률을 예시한 것이다. 예시에서는 Cycle-Conserving RT-DVS 기법(이하 cc-EDF)과 3개의 실시간 태스크를 이용하였다[10]. 각 태스크는 τ_1 = {10, 10, 4}, τ_2 = {10, 10, 3}과 τ_3 = {10, 10, 2}로 구성되며, 실제 실행 시간은 3, 2, 2이다. 주파수의 간격은 0.25와 0.1이다.

유휴 시간은 식 (10)과 같이 시스템의 이용률과 이산적인 주파수 사이에 차이가 발생한다. 이는 프로세서에서 제공하는 주파수가 연속적인 주파수가 아니라 이산적인 주파수이기 때문이다. 그림 1의 (a)처럼 Δ가 크면 시스템의 유휴시간이 많이 발생하여 전력 소비가 증가한다. 따라서 유휴 시간을 줄이기 위해서는 Δ를 줄여야 하고 이를 위해서는 주파수 간격을 줄여야 한다. 그림 1의 (b)처럼 주파수 간격이 작을 경우, 시스템 이용률의 변화가 크지 않고 유휴 시간도 작음을 알 수 있다.

$$\Delta = \lceil u_t \rceil - u_t, \quad (0 \le \Delta \le \Delta f) \tag{10}$$

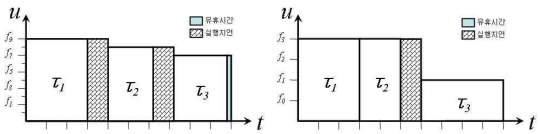
그러나 그림 2와 같이 주파수 변경에 따른 실행 지연이 존재하는 실 시스템의 경우 전력 소비를 줄이기 위해 주파수 간격과 더불어 주파수 변경에 따른 실행 지연을 고려해야 한다. 주파수 변경에 따른 오버헤드는 1ms로 가정한다.

그림 2의 (b)는 그림 2의 (a)보다 주파수 간격이 작음에도 불구하고 주파수 변경 횟수가 많아진다. 따라서 같은 태스크 집합에서 이상적인 시스템과는 달



(a) Δf = 0.25 (b) Δf = 0.1

그림 1. 주파수 변경 시간을 고려하지 않은 이상적인 시스템에서 이용률 변화



(a) Δf = 0.25 (b) Δf = 0.1

그림 2. 주파수 변경 시간을 고려한 실 시스템에서 이용률 변화

리 시스템의 이용률이 더 증가함을 알 수 있다. 이는 더 많은 전력을 소비하는 것을 의미한다. 따라서 주파수를 f_i에서 f_k로 변경할 경우, 오버헤드와 비교하여 소비 전력을 절감할 수 있는지 확인해야 한다. 식 (11)과 같이 주파수를 변경하여 줄일 수 있는 소비 전력이 주파수 변경에 따른 실행 지연 시간(ΔT)보다 적어야 한다.

$$T_i \cdot |P(f_i) - P(f_k)| > \Delta T \cdot (P(f_i) + P(f_k)) \tag{11}$$

이를 태스크의 주기(T_i)에 대하여 정리 하면 식 (12)을 유도할 수 있다.

$$T_i > \Delta T \cdot \frac{}{|P(f_i) - P(f_k)|} \tag{12}$$

식 (12)로부터 주파수 변경을 통하여 전력 소비를 줄이기 위해서는 아래 세 가지 조건 중에 하나를 만족해야 한다.

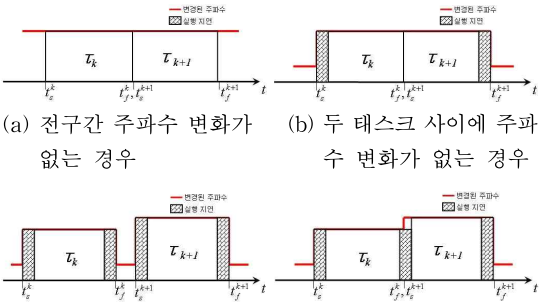
1) 태스크 주기(T_i)의 값이 아주 크거나,
2) 주파수 변경에 따른 실행 지연(ΔT)이 아주 작거나
3) $P(f_i) \gg P(f_k)$ or $P(f_k) \ll P(f_i)$를 만족해야 한다.

1) 항과 2) 항의 조건들은 응용 프로그램이나 시스템에 의하여 결정되는 경우가 많다. 따라서 본 논문에서는 3) 항의 조건을 이용하여 변경하고자 하는 주파수 사이의 소비 전력을 측정하여 새로운 주파수로 변경할 지를 결정하는 알고리즘을 제안한다. 이는 주파수 변경에 따른 오버헤드를 줄일 수 있다.

3.2 알고리즘

태스크를 스케줄링 할 때 주파수가 변경되는 경우는 그림 3과 같다. 대개 태스크들은 연속적으로 실행되거나 단독으로 실행된다. 연속적으로 실행되는 태스크들 사이에서 주파수가 변경될 수 있다.

실시간 시스템에서 마감 시간을 만족하기 위해서



(a) 전구간 주파수 변화가 없는 경우 (b) 두 태스크 사이에 주파수 변화가 없는 경우

(c) 두 태스크 사이에 주파수 변화가 있는 경우 (d) 태스크가 단독으로 실행되는 경우

그림 3. 실행 지연을 고려한 주파수 변경에 따른 태스크 스케줄링

주파수 변경으로 인한 실행 지연을 고려해야 한다. 따라서 주파수를 설정할 경우 식 (13)과 같이 실행 지연을 시스템의 이용률에 포함해야 한다.

$$u_k = \begin{cases} \dfrac{C_k}{t_f^k - t_s^k}, \\ \dfrac{C_k}{t_f^k - t_s^k - \Delta t}, \\ \dfrac{C_k}{t_f^k - t_s^k - 1.5 \cdot \Delta t}, \\ \dfrac{C_k}{t_f^k - t_s^k - 2 \cdot \Delta t}. \end{cases} \quad (13)$$

본 논문에서 제안하는 전력 관리 알고리즘은 [1]에서 제안된 주파수 평활화 기법을 바탕으로 하였다. [4]의 연구는 주파수 변경에 따른 일정한 실행 지연을 태스크의 최악 실행 시간에 더하여 시스템의 마감 시간을 만족시켰으나 태스크 실행 순서에 따라 발생할 수 있는 실행 지연의 차이를 충분히 반영하지 못하였으며, 소비 전력의 절감 여부에 관계없이 주파수를 변경하여 전력 소비를 증가시켰다.

본 논문에서는 기존 연구의 단점을 해결하기 위해 식 (13)을 이용하여 태스크의 실행 순서에 따라 발생하는 실행 지연을 각각 다르게 적용하여 실시간 특성을 만족시키며, 식 (12)을 이용하여 소비 전력의 절감 여부를 판단하여 주파수를 변경하였다. 이를 바탕으로 그림 4와 같은 알고리즘을 제안한다.

프로그램의 시작 초기(t_0)에서 H_1까지 저장된 시스템의 이용률 정보가 없으므로 시스템에서 설정한 주파수 테이블을 사용한다. H_1이후에는 이전의 시스템 이용률 정보를, U_{max}와 U_{ave}, 바탕으로 주파수 테이블을, F = {f_0, f_1, f_2, f_3}, $f_3 = f_{max}$, $f_2 = U_{max}$, $f_1 = (U_{max} + U_{ave})/2$, $f_0 = U_{ave}$로 다시 설정한다. 시스템의 소비 전력은 식 (3)과 같이 변경된 주파수와 전압의 제곱

```
// 매 Hyper Period 마다 주파수 테이블을 다시 조정함
adjust_frequency_table (Max, Ave) {
    F0 = Ave; F1 = (Max + Ave)/2; F2 = Max; F3 = fmax;
}

// 최초 사용될 주파수 테이블
set_default_frequency_table () {
    F0 = 0.25; F1 = 0.5; F2 = 0.75; F3 = fmax;
}

// 시스템 시작 전에 Task Queue와 주파수 Table을 초기화함.
On_initial() {
    empty tQ;
    set_default_frequency_table();
}

// 시스템 이용률(오버헤드 포함)에 적합한 최저 주파수 선택
set_speed(Utotal) {
    use lowest frequency fi ∈ { f0, ⋯, fmax | f0 < ⋯ < fmax} such that Utotal ≤ fi/fmax;
    if (Ti > ΔT - (P(fprev) + P(fi)) / | P(fprev) - P(fi)| )
        return fi;
    else return fprev;
}

// 태스크가 시작될 때
on_task_start () {
    if (!(time mod HyperPeriod))
        adjust_frequency_table(Max, Ave);
    if (no task in tQ) set Ui to Ci/(Ti + 2 ΔT);
    else if (0 ≤ (fprev - Ci/Ti) < Δf) set Ui to Ci/Ti;
    else set Ui to Ci/(Ti + 1.5 ΔT);
    set_speed(Utotal);
}

// 태스크 마감 할 때 각 태스크의 실행 profile을 마련함
on_task_end() {
    gather the Max/Ave of Ui;
    if (no new task in tQ) set Ui to Ci/(Ti + 2 ΔT);
    else if (0 ≤ (fprev - Ci/Ti) < Δf) set Ui to Ci/Ti;
    else set Ui to Ci/(Ti + 1.5 ΔT);
    set_speed(Utotal);
}

// 실행할 태스크가 없을 경우
on_idle() {
    set_speed(0);
}
```

그림 4. 제안한 전력 관리 알고리즘

의 곱으로 이루어진다. 따라서 소비 전력이 최소가 되기 위해서는 전압의 값이 같거나 차이가 최소가 되어야 한다. 즉, 전압은 주파수에 비례하므로, 동일한 주파수로 시스템을 실행하거나 변경되는 주파수의 차이가 적어야 한다. 따라서 최저 주파수인 f_0를 시스템의 평균 이용률인 U_{ave}로 설정하여 주파수가 자주 변경되는 것을 막는다.

4. 모의실험

본 논문에서 제안한 전력 관리 기법의 성능은 모의실험을 통하여 평가한다. 성능 평가는 주파수 변경으로 인한 시스템 오버헤드와 시스템의 이용률 변화에 따른 소비 전력 평가와 알고리즘의 확장성을 평가하기 위해 태스크 수의 변화에 따른 소비 전력 평가로 이루어진다.

4.1 실험 환경

본 논문에서 제안한 전력 관리 알고리즘의 성능을 평가하기 위하여 RTSim 0.5.1을 이용한다[15]. RTSim은 RETIS Lab에서 실시간 시스템의 스케줄링 평가를 위하여 개발되었으며 GPL에 의거하여 누구나 이용하거나 수정이 가능하다[15]. 모의실험에서 사용하는 주파수 테이블은 표 1과 같이 PXA 270 프로세서에서 제공하는 주파수를 이용한다[13]. PXA 270 프로세서는 휴대용 멀티미디어 시스템과 PDA 등 많은 휴대 장비에 사용되며, 교육용 장비로도 많이 보급되어 본 논문에서 제안하는 알고리즘을 쉽게 구현 할 수 있다.

본 논문에서는 모의실험을 통하여 제안하는 전력 관리 알고리즘의 성능 향상을 증명하고자 한다. 이를 위해 Pillai가 제안한 cc-EDF 기법과 주파수 평활화를 통한 전력 관리 알고리즘을 비교 대상으로 선정하였다[4,10]. cc-EDF 기법의 경우, 표 1의 PXA 270에서 제공하는 48개의 주파수를 이용한 것과 표 1에서 제공하는 주파수 중 25%, 50%, 75%, 100%의 4개를 이용한 것을 고려한다. 이를 cc-EDF 48과 cc-EDF 4로 각각 표시한다. 주파수 평활화 기법을 이용한 방법은 smoothing으로 표시한다. 본 논문에서 제안한 전력 관리 알고리즘은 그림 4의 알고리즘을 바탕으로 표 1에서 제공하는 주파수 중에 4개를 선택하여 설정한다. 이를 A-DVS로 표시한다.

표 1. 모의실험에서 사용될 주파수 테이블

주파수[1]	전압[1]	주파수	전압	주파수	전압
13	0.85	221	1.16	429	1.36
26	0.86	234	1.17	442	1.37
39	0.87	247	1.19	455	1.39
52	0.87	260	1.20	468	1.40
65	0.88	273	1.21	481	1.41
78	0.89	286	1.22	494	1.42
91	0.89	299	1.24	507	1.44
104	0.9	312	1.25	520	1.45
117	0.93	325	1.26	533	1.46
130	0.96	338	1.27	546	1.47
143	0.99	351	1.29	559	1.49
156	1.02	364	1.30	572	1.50
169	1.06	377	1.31	585	1.51
182	1.09	390	1.32	598	1.52
195	1.12	403	1.34	611	1.54
208	1.15	416	1.35	624	1.55

[1]주파수의 단위는 MHz이고 전압의 단위는 V이다

본 논문에서 사용되는 태스크의 집합은 표 2와 같으며, 4개의 실시간 태스크로 구성되어 있다. 실제 사용 환경에 근접한 모의실험 환경을 구축하기 위해 시스템의 이용률을 10%에서 90%까지 변화시켜서 알고리즘의 성능을 평가한다. 이 경우, 최악 실행 시간은 시스템 이용률에 따라 변하며, 실제 사용 시간은 최악 실행 시간의 50%에서 90% 사이의 값으로 적용한다.

4.2 모의실험 결과 평가

그림 5는 전압 변경에 따른 오버헤드의 변화에 따라 전력 관리 알고리즘의 성능을 평가한 것이다. 실제 이용 시간을 최악 실행 시간의 70%으로 정한다.

표 2. 모의실험에서 사용되는 테스트 집합

TASK	PERIOD[1]	WCET[2]	AET
A	10	2	(50%~90%) * WCET
B	20	3	(50%~90%) * WCET
C	30	6	(50%~90%) * WCET
D	40	6	(50%~90%) * WCET

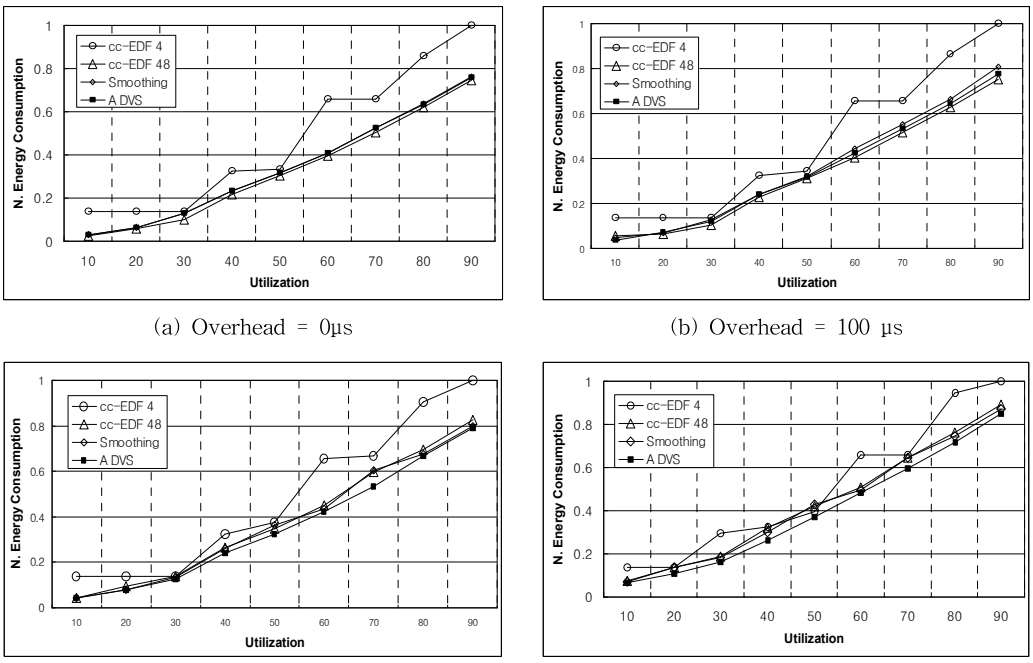
[1]단위는 ms임.
[2]전체 이용률 70% 기준임.

식 (13)에서 사용되는 전압 변경에 따른 오버헤드(ΔT)의 값을 0μs, 100μs, 500μs 그리고 1ms로 각각 설정한다.

그림 5의 (a)와 같이 전압 변경에 대한 오버헤드가 없는 경우에는, 주파수의 개수가 많은 cc-EDF 48 방식이 소비 전력을 가장 많이 절감함을 알 수 있다. 이는 전압 변경의 횟수가 시스템의 전력 소비에 영향을 주지 않음을 의미한다. 따라서 이상적인 시스템에서는 사용 가능한 주파수의 개수가 많을수록 전력 소비를 줄임을 알 수 있다. 그러나 그림 5의 (b), (c), (d)와 같이 오버헤드의 값이 증가할수록 시스템의 전력 소비가 증가함을 알 수 있는데, 이는 주파수의 개수가 많을수록 주파수의 변경 횟수가 많아지고 이로 인한 시스템의 전력 소비가 증가하기 때문이다. cc-EDF 4 방식은 시스템 이용률이 30%, 50%, 70%의 경우 주파수 변경 횟수가 적은 반면 smoothing 기법은 소비 전력 절감 여부에 관계없이 주파수를 변경하여 cc-EDF 4 방식 보다 더 많은 전력을 소비하는 것으로 나타났다. 그러나 본 논문에서 제안한 A-DVS 방식은 식 (12)에 의거하여 전압 변경에 따른 오버헤드와 전압 변경에 따른 소비 전력 절감을 비교하여 전압 변경의 횟수를 줄여서 다른 방식에 비해 전압 변경에 따른 전력 소비를 줄일 수 있다.

그림 6은 알고리즘의 확장성을 평가하기 위해 태스크의 개수를 증가시키면서 알고리즘의 성능을 평가한 것이다. 모의실험 조건은 그림 5의 실험 조건과 같으나 태스크의 개수를 4개와 8개로 한다. 주파수 변경에 따른 오버헤드는 PXA 270의 주파수 변환 오버헤드와 외부 전압 정류 장치인 Texas Instrument 사의 TSP62400을 이용한 전압 변환 오버헤드를 적용한다. 각 오버헤드는 150μs와 350μs이다[14,16]. 그러나 모의실험에서는 주파수가 변할 경우 전압도 같이 변한다고 가정하고 오버헤드를 500μs로 정한다.

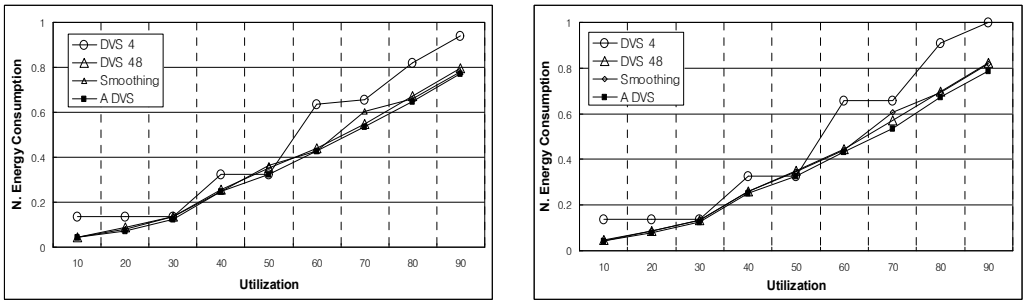
주파수는 대개 태스크의 시작과 종료 때 설정한다. 태스크의 수가 증가하면 주파수를 변경하는 횟수도 증가할 수 있다. 그림 6의 (a)와 (b)에서 시스템 이용률은 같지만, 많은 태스크를 가진 그림 6의 (b)에서 더 많은 전력을 소비함을 할 수 있다. 이는 태스크의 수가 많아서 태스크의 실 사용 시간을 시스템에 빨리 반영할 수 있기 때문에 주파수를 변경할 수 있는 횟수가 증가하기 때문이다. Smoothing 방식의 경우, 전압 변경이 적은 시스템 이용률 30%, 50%, 70%



(a) Overhead = 0μs (b) Overhead = 100 μs

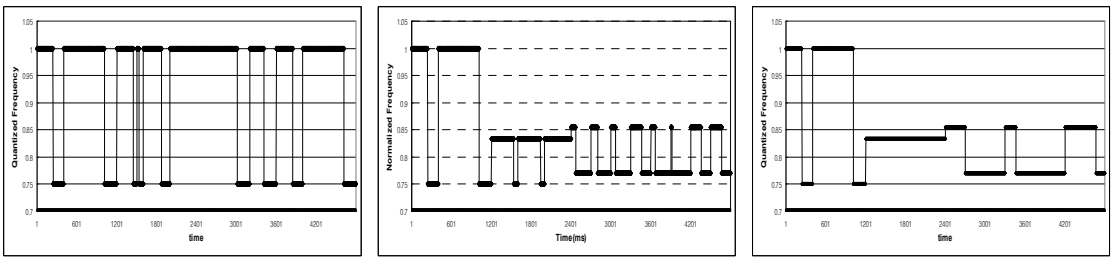
(c) Overhead = 500 μs (d) Overhead = 1ms

그림 5. 전압 변경에 따른 오버헤드 변화에 따른 성능 평가



(a) 4 tasks in CC RT-DVS (b) 8 tasks in CC RT-DVS

그림 6. 태스크 수 증가에 따른 성능 평가



(a) cc-EDF 4 기법 (b) Smoothing 기법 (c) Adaptive DVS 기법

그림 7. 시스템 이용률 변화에 따른 주파수 설정

에서 cc-EDF 4 방식 보다 더 많은 전력을 소비하는 것으로 나타났다. 본 논문에서 제안한 A-DVS방식은 태스크의 수가 증가하더라도 다른 방식에 비해 소비 전력을 약 2 ~ 25% 정도 절감함을 알 수 있다.

그림 7은 cc-EDF 4 방식, smoothing 방식과 본 논문에서 제안하는 A-DVS 방식에서 시스템의 이용률에 따라 주파수를 어떻게 설정하는지를 보여주는 것이다. 모의실험 조건에서 실제 이용 시간은 최악 실행 시간의 80%로 정하며, 시스템 이용률은 70%이고 다른 조건은 그림 6의 실험 조건과 같다. 그림 7의 (a)에서 cc-EDF 4 방식은 주파수가 시간이 지남에 따라 시스템 이용률에 상관없이 주파수 100%와 75%, 두 단계에서만 반복한다. 그러나 주파수 변경 횟수가 많지 않음을 알 수 있다. 그림 7의 (b)에서 smoothing 방식은 시간이 지남에 따라 시스템 이용률에 따라 주파수를 다시 설정하여 실제 시스템 이용률에 근접하게 주파수를 설정하지만, 주파수를 자주 변경함을 알 수 있다. 그림 7의 (c)에서 A-DVS 방식은 smoothing 방식과 비슷하게 주파수를 설정하지만, 소비 전력 절감 여부를 판단하여 주파수를 변경하므로 주파수의 변경 횟수가 다른 방식에 비해 현저히 적음을 알 수 있다.

5. 결 론

본 논문에서 동적 전압 조절 기법을 실 시스템에 적용할 경우 고려해야 하는 제약 사항을 고려하면서 매 Hyper Period마다 시스템의 이용률 정보를 바탕으로 사용 가능한 주파수의 개수와 연속된 주파수 사이의 간격을 조절하여 소비 전력을 줄이는 전력 관리 알고리즘을 제안하였다. 본 알고리즘은 전압 변경에 따른 오버헤드와 전압 변경에 따른 소비 전력 절감을 비교하여 주파수를 변경하여 주파수 변경 횟수와 시스템의 유휴 시간을 줄여 기존의 cc-EDF 4, cc-EDF 48 알고리즘, smoothing 알고리즘과 비교하여 최대 25%, 평균 10% 정도 소비 전력을 절감할 수 있다. 본 논문에서는 PXA270과 같은 현재 많은 사용되고 있는 프로세서의 특성을 바탕으로 하였으므로 향후 실 시스템에 별 다른 변경 없이 구현 가능하다.

그러나 본 논문은 Hyper Period를 기준으로 시스템의 평균 이용률을 산출하기 때문에, 각 태스크의 주기가 서로 달라서 Hyper Period가 길어지면 시스템의 평균 이용률 산출 시간이 길어지는 단점이 있다. 따라서 향후 이러한 단점을 개선하고자 한다. 또한, 제안한 전력 관리 알고리즘을 휴대용 내장 시스템에서 많이 사용되는 PXA 270 프로세서와 embedded Linux에서 구현하여 성능을 평가할 필요가 있다.

참 고 문 헌

[1] F. Yao, A. Demers, and S. Shenker, "A Scheduling Model for Reduced CPU Energy," In IEEE Annual Symposium on Foundations of Computer Science, pp. 374-382, 1995.

[2] IBM and Montavista Software, "Dynamic Power Management for Embedded Systems," http://www.research.ibm.com/arl/projects/dpm.html, 2002.

[3] S. Borkar, "Design Challenges of Technology Scaling," IEEE Micro, Vol.19 No.4, pp. 23-29, 1999.

[4] 권혁성, 안병철, "주파수 평활화 기법을 이용한 전력 관리 알고리즘," 전자공학회 논문지, 제 45권 CI편 제1호, pp. 78-85, 2008.

[5] Jong-U Shin and Taewhan Kim, "Technique for Transition Energy-Aware Dynamic Voltage Assignment," IEEE transactions on circuits and systems, Vol.53, No.5, pp. 956-960, 2006.

[6] Tomas D. Burd and R.W.Brodersen, "Design Issues for Dynamic Voltage Scaling," In Proceedings of the 2000 International Symposium on Low Power Electronics and Design, pp. 9-14, 2000.

[7] C.M. Krishna and Yann-Hang Lee, "Voltage-clock-scaling Adaptive Scheduling Techniques for Low Power in Hard Real-time Systems," IEEE transactions on computers, Vol.52, No.12, pp. 1586-1593, 2003.

[8] Jaewon Seo, Taewhan Kim and Joonwon Lee, "Optimal Intratask Dynamic Voltage-Scaling Technique and Its Practical Extensions," IEEE transactions on computer-aided design of integrated circuits and systems, Vol.25, No. 1, pp. 47-57, 2006.

[9] Lin Yuan and Gang Qu, "Analysis of Energy Reduction on Dynamic Voltage Scaling-Enabled Systems," IEEE transactions on computer-aided design of integrated circuits and systems, Vol.24, No.12, pp. 1827-1837, 2005.

[10] P. Pillai and K.G. Shin, "Real-time Dynamic Voltage Scaling for Low-power Embedded Operating Systems," In Proceedings of the 18th ACM Symposium on Operating Systems Principles, pp. 89-102, 2001.

[11] Saowanee Saewong and Ragunathan Rajkumar, "Practical Voltage-Scaling for Fixed-Priority RT-Systems," In Proceedings of the ninth IEEE Real-Time and Embedded Technology and Applications Symposium, pp. 224-233, 2003.

[12] Yumin Zhang, Xiaobo S. Hu and Danny Z. Chen, "Energy Minimization of Real-time Tasks on Variable Voltage Processors with Transition Energy Overhead," In Asia and South Pacific Design Automation Conference, pp. 65-70, 2003.

[13] Intel, PXA270 Processor Electrical, Mechanical, and Thermal Specification.

[14] C. L. Liu, and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a hard real time environment," Journal of the Association for Computing Machinery, Vol. 20, No.1, pp. 46-61, 1973.

[15] RTsim homepage, http://rtsim.sssup.it/

[16] Texas Instrument Inc, TPS62400 user's guide.

권 혁 성

1995년 영남대학교 전산공학과 학사 졸업
1997년 영남대학교 전산공학과 석사 졸업
1997년～2006년 LG전자 DDC연구소 책임연구원
2005년～현재 영남대학교 컴퓨터공학과 박사과정
관심분야 : 실시간 시스템, 저전력 관리, sensor network, 디지털 TV

안 병 철

1976년 영남대학교 전자공학과 학사 졸업
1986년 오레곤주립대 전기 및 컴퓨터공학 석사 졸업
1989년 오레곤주립대 전기 및 컴퓨터공학 박사 졸업
1976년～1984년 국방과학연구소 연구원
1989년～1992년 삼성전자 수석연구원
1992년～현재 영남대학교 전자정보공학부 교수
관심분야 : 임베디드시스템, 실시간운영체제, 멀티미디어처리