DOI: 10.3745/KIPSTA.2011.18A.3.093

저전력 시스템을 위한 BET기반 태스크 분할 스케줄링 기법

박 상 오[†] · 이 재 경[††] · 김 성 조[†††]

요 약

최근 배터리로 동작하는 임베디드 시스템의 사용이 급격히 증가하고 있지만, 현재 배터리 기술의 발전 속도는 임베디드 시스템의 전력 사용량의 증가를 따라가지 못하여, 장시간 사용을 위해서는 배터리의 크기가 커져야 하는 단점이 있다. 내장형 시스템에서 소모하는 전력량은 시스템을 구성하는 하드웨어와 시스템을 구동하는 소프트웨어에 의해 결정된다. 그러나 하드웨어적으로 저전력을 지원하더라도 운영체제 등 소프트웨어 수준에서 이를 활용하지 못하면 절전 효과를 극대화할 수 없다. 따라서 본 논문에서는 모바일 임베디드 시스템 환경에서 멀티미디어 애플리케이션 구동시 BET(Break Even Time)기반 태스크 분할을 이용하여 소비 전력을 감소시키는 스케줄링 기법을 제안한다.

키워드 : 저전력, 임베디드 시스템, 스케줄링, 실시간 운영 체제

A Scheduling Method using Task Partition for Low Power System

Sang Oh Park[†] · Jae Kyoung Lee[††] · Sung Jo Kim[†††]

ABSTRACT

While the use of battery-powered embedded systems has been rapidly increasing, the current level of battery technology has not kept up with the drastic increase in power consumption by the system. In order to prolong system usage time, the battery size needs to be increased. The amounts of power consumption by embedded systems are determined by their hardware configuration and software for manipulating hardware resources. In spite of that, the hardware provides features for lowering power consumption, if those cannot be utilized efficiently by software including operating system, reduction in power consumption is not optimized. In this paper, we propose a BET(Break Even Time)-based scheduling method using task partition to reduce power consumption of multimedia applications in a mobile embedded system environment.

Keywords : Low Power, Embedded System, Scheduling, Real-Time Operating System

1. 서 론

최근 모바일 임베디드 시스템은 멀티미디어 애플리케이션을 구동하기 위해 높은 속도의 프로세서에 대한 요구가 증가하고 있다. 또한, 시스템 하드웨어가 소모하는 자원이 늘어나면서 시스템이 소비하는 전력 역시 증가하고 있으며, 소비전력의 증가는 제한된 전원 용량을 가진 밧데리를 기반으로 운용되는 모바일 임베디드 시스템의 사용시간 단축을 초래하였다. 이런 문제를 해결하기 위해 하드웨어적인 측면과 소프트웨어적인 측면에서 저전력을 실현하는 기법에 대해 그 동안 많은 연구[1][2][3][4][5]가 진행되어 왔다.

※ 이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. 2010-0000414).
† 정 회 원 : 중앙대학교 박사 후 과정
†† 정 회 원 : 삼성 SDS ESDM 개발지원 그룹 사원
††† 정 회 원 : 중앙대학교 컴퓨터공학부 교수
논문접수 : 2010년 10월 28일
수 정 일 : 1차 2011년 3월 11일
심사완료 : 2011년 3월 12일

기존의 전력 관리 기법에는 프로세서만을 고려한 DVS (Dynamic Voltage Scaling)[1][2][5]와 시스템의 구성 요소별 사용 패턴을 기반으로 해당 디바이스가 비활성화 상태에 있는 경우 해당 디바이스가 제공하는 구분된 파워 상태 중 주어진 수행의 요구 조건에 맞추어 공급 전력을 조절하는 DPM(Dynamic Power Management)[3][4]등이 있다. 하지만 이러한 기법들은 프로세서나 주변 디바이스들 중 하나만을 고려함으로써 전력 소모량을 감소시키는데 한계가 있다. 프로세서와 디바이스 모두를 고려한 기법[6]이 연구되고 있지만 단일 태스크에서의 전력 소비량만을 고려하고 있다.

본 논문에서는 디바이스가 갖는 고유 속성인 BET(Break Even Time)중 최대값을 기준으로 태스크를 Job 단위로 분할한 후 기존 기법의 특징을 고려하여 DPM과 DVS를 혼합한 BTPS(BET-based Task Partition Scheduling)기법을 제안한다. 이 기법은 기존 기법의 장점은 그대로 유지하고, 단점은 보완함으로써 전력 소모량을 줄일 수 있는 방안을 제

안한다. 또한 BTPS가 적용되었을 때 추가로 발생하는 시간 오버헤드로 인해 태스크의 데드라인(Deadline)을 만족시킬 수 없어 시스템의 QoS(Quality of Service)를 보장하지 못하는 문제가 발생할 수 있으므로 스케줄러 호출에 따른 시간 오버헤드를 극복하며 데드라인을 준수할 수 있는 프로세서의 처리 속도를 찾아 실시간성을 보장한다. 또한 BTPS의 시간 오버헤드로 인해 추가 발생하는 전력 소비량과 BTPS를 통해 감소되는 전력 소비량을 비교 및 분석하여 추가 발생하는 전력 소비를 보상받을 수 있도록 데드라인을 준수하는 프로세서의 처리 속도 범위에서 전력 소비를 최소로 하는 프로세서 처리 속도를 선택하여 우선순위에 따라 선택된 태스크를 수행시킴과 동시에 주변 디바이스의 유휴 시간과 전력 소비량을 고려하여 앞으로 수행할 태스크를 선택한다.

본 논문의 구성은 다음과 같다. 2절에서는 기존 전력 관리 기법의 문제점에 대해 알아보고, 3절에서는 제안하는 BTPS 기법에 대하여 설명한다. 4절에서는 제안한 기법에 대한 성능 평가와 분석을 통해 제안된 기법의 타당성을 검증하며, 5절에서는 결론과 향후 연구 과제에 대해서 기술한다.

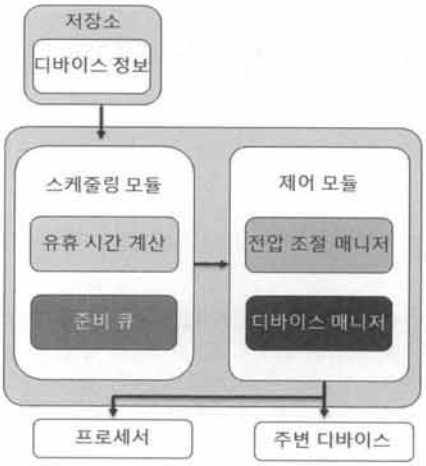
2. 관련 연구

DPM은 미리 정해진 시간(Decision Time) 이상으로 디바이스가 유휴(Idle)상태에 있을 경우 디바이스의 전압 상태를 할당된 정책에 따라 동작상태와 유휴상태 등 단계별로 변화시킨다. 하지만 프로세서가 동적으로 전압 조절이 가능할 경우, DVS가 DPM에 비해 프로세서 전력 소모 측면에서 효율적이기 때문에 일반적으로 DPM은 주변 디바이스에 주로 적용된다.

DVS는 CMOS 회로로 구성되는 프로세서의 공급 전압을 동적으로 조절하여 에너지 소모량을 줄이는 기법으로 널리 사용되고 있다. 일반적으로 CMOS 회로의 전력 소모(P)는 공급 전압(V_{DD})에 대해 $P \propto V_{DD}^2$인 관계를 가지므로, 공급 전압의 감소는 전력 소모량 측면에서 매우 효과적이다. 하지만 DVS는 공급 전압 조절을 지원하는 디바이스에만 적용이 가능하다는 단점이 있다.

LPFPS(Low Power Fixed Priority Scheduling)[3]는 DPM과 DVS를 혼합하여 사용하는 기법이며 스케줄링이 태스크 단위로 이루어지기 때문에 태스크의 실행 시간(Execution Time)과 최악 수행 시간(Worst Case Execution Time : WCET) 사이에 발생하는 작업 부하 변경에 따른 유휴시간(Workload-Variation Slack Time : WVST)을 충분히 이용하지 못하는 단점이 있다.

마지막으로, RVH(Run-time Voltage Hopping)[6]는 LPFPS의 한계를 극복하고자 제안된 기법으로 타임슬롯(Timeslot)개념을 도입하였다. 하지만 RVH는 디바이스의 잦은 Wakeup으로 인한 시간 지연과 Wakeup 시 추가로 소비되는 전력은 고려하지 않는 문제가 있다.



(그림 1) 전력 관리 모듈

3. BTPS (BET-based Task Partition Scheduling)

본 논문에서 제안하는 전력 관리 시스템은 다음에 수행될 태스크를 결정하는 스케줄링 모듈, 프로세서 전력 및 클럭을 할당하고 주변 디바이스의 상태를 변경하는 제어 모듈, 그리고 디바이스의 상태와 소비 전력에 대한 정보가 저장되는 저장소로 구성된다(그림 1).

3.1 스케줄링 모듈

스케줄링 모듈은 (그림 1)과 같이 태스크가 수행된 후 유휴 시간을 계산하는 유휴 시간 계산기와 수행될 태스크들의 집합인 준비 큐로 구성된다. 스케줄링 모듈에서 사용하는 스케줄링 기법은 전력 사용량을 줄이기 위해 가장 긴 BET(Break-Even Time)[7]를 기준으로 태스크를 연속된 몇 개의 Job으로 분할하여 WVST를 최대한 이용할 수 있도록 한다. BET는 디바이스의 전자기적 특성에 의해 결정되는 고유값으로 디바이스의 상태를 변화시키기 위해 소비하는 전력량 즉, 디바이스를 On/Off 할 때 소비하는 전력 소모량과 디바이스가 유휴 상태를 유지함으로써 절약할 수 있는 전력량이 같아지는 시간(단위: 클럭 사이클)을 의미한다. 디바이스가 갖는 각각의 BET보다 유휴 시간이 길다면 디바이스를 유휴 상태로 변화시킴으로써 전력 소비량의 감소를 보장한다. 본 논문에서는 디바이스의 BET 중 최대값을 기준으로 태스크를 분할함으로써 기준값보다 오랜 시간동안 시스템이 유휴하다면 모든 디바이스를 유휴 상태로 유지함으로써 모든 디바이스의 전력 소비량 감소를 보장하도록 설계하였다. 또한 Job간의 디바이스 연속 사용 여부를 파악하여 디바이스 Wakeup시 소모되는 전력이 최소화 되도록 스케줄링 한다.

3.1.1 유휴 시간 계산기 (Slack Time Calculator)

태스크들의 유휴 시간(Slack Time)은 정적 유휴 시간과 동적 유휴 시간 등 두 가지로 분류된다. 전자는 태스크가 최악 실행 시간 동안 동작했을 때 데드라인까지 남는 시간

이며 정적으로 파악이 가능하다. 후자는 태스크들의 실제 실행 시간과 최악 실행 시간(WCET)과의 차이이다. 예를 들어, 태스크의 데드라인이 5초이고, WCET가 3초인 태스크의 경우, 주어진 작업은 최악의 경우에도 매 5초마다 2초의 정적 유휴 시간이 발생한다. 만일 주어진 작업의 실제 실행 시간이 1초라면, 시스템은 최악 실행에 비해 2초의 추가 동적 유휴 시간을 얻게 된다.

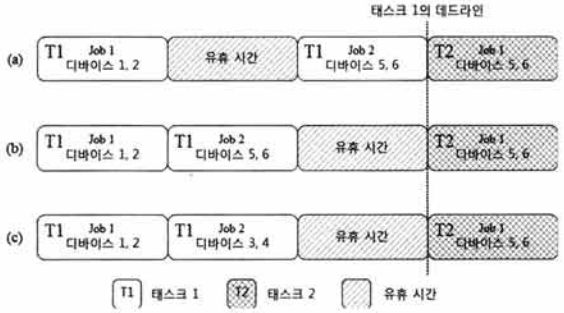
정적 유휴 시간은 (Deadline - WCET)이며, 동적 유휴 시간은 (WCET - Execution Time)이 된다. 따라서 전체 유휴 시간(τ_s)은 아래 식(1)과 같이 표현된다. 여기서 τ_d는 데드라인, τ_e는 수행 시간을 나타내며 단위는 클럭 사이클이다.

$$\tau_s = \tau_d - \tau_e \qquad (1)$$

3.1.2 준비 큐 (Ready Queue)

준비 큐는 수행될 태스크들의 Job들을 관리한다. 같은 태스크에 속한 Job들은 순서쌍을 가지며, 한 태스크에 속한 Job들의 수행 순서에 따라 태스크 수행 결과가 달라질 수 있기 때문에 태스크에서 Job의 순서는 변경될 수 없다.

준비 큐에서 프로세서의 유휴 시간을 고려하여 다음에 수행할 태스크를 결정할 때, (그림 2 (a))에서 태스크 1의 Job 2와 태스크 2의 Job 1과 같이 동일한 디바이스를 사용하는 Job을 선택하여 수행한다. 만약 (그림 2 (b))와 같이 동일한 디바이스를 사용하는 태스크 1의 Job 2와 태스크 2의 Job 1을 고려하지 않고, 유휴 시간을 태스크1의 수행 시간 중 가장 마지막에 배치하여 활용하게 되면 유휴시간 동안 디바이스 5와 6의 장치를 지속적으로 사용하거나 전원을 차단했다가 다시 공급해야 함으로 오버헤드가 더 커질 수 있다. 따라서 동일한 디바이스를 사용하는 Job의 존재 유무를 확인하고, 존재할 경우, (그림 2 (a))와 같이 동작시키고, 존재하지 않을 경우, (그림 2 (c))와 같이 동작시켜 디바이스의 상태 변경을 최소화하고 유휴시간을 최대화하여 전력 소모를 줄인다.



(그림 2) 디바이스 연속 사용에 따른 Job수행 순서의 예

3.2 제어 모듈 (Control Module)

제어 모듈은 (그림 1)과 같이 프로세서의 공급 전압을 조절하는 전압 조절 매니저와 주변 디바이스에 공급되는 전압을 변경하는 디바이스 매니저로 구성된다.

전압 조절 매니저(Voltage Scaling Manager)는 동적 전압 조절 기법(DVS)을 이용하여 프로세서에 공급되는 전압을 조절함으로써 처리 속도를 변경함으로써 전력 소모를 최소화한다. 공급 전압의 증가는 프로세서의 클럭 사이클을 증가시켜 처리 속도가 빨라지기 때문에 공급 전압과 프로세서의 처리 속도는 선형관계를 갖는다. 이때 전압의 단위는 Voltage이고 프로세서의 처리 속도 단위는 Hertz이다.

본 논문에서 제안한 BTPS는 스케줄러 호출에 따른 시간 오버헤드를 포함한 실행시간 분석을 통해 스케줄러 호출에 따른 시간 오버헤드를 극복하며 데드라인을 준수할 수 있는 프로세서의 클럭 사이클을 찾아 실시간성을 보장한다. 또한 BTPS를 통해 시간 오버헤드로 인해 추가로 발생하는 전력 소비량과 BTPS를 통해 감소되는 전력 소비량을 비교 및 분석하여 추가로 소비되는 전력을 보상받을 수 있도록 데드라인을 준수하는 프로세서의 처리 속도 범위에서 전력 소비가 최소가 되는 프로세서 처리 속도를 선택한다. 식(2)는 Job 단위 스케줄러 호출로 인한 시간 오버헤드(τ_o)이다.

$$\tau_o = \tau_{sch} + \tau_{slk} + \tau_{search} + \tau_{vol} + \tau_{trans} \qquad (2)$$

식(2)에서 τ_{sch}, τ_{slk}, τ_{search}, τ_{vol}, τ_{trans}는 각각 스케줄링에 필요한 클럭 사이클, 유휴시간(Slack Time)계산에 필요한 클럭 사이클, 수행 중인 주변 장치 검색에 필요한 클럭 사이클, Voltage Scaling 계산에 필요한 클럭 사이클, 주변장치 상태 변경에 필요한 클럭 사이클을 의미한다.

$$\frac{\tau_e + \tau_o}{f_{scal}} \le \tau_d \qquad (3)$$

식 (3)은 주어진 Job 또는 태스크가 어떤 주어진 데드라인(τ_d)을 만족시키기 위해 필요한 최소한의 프로세서 처리 속도를 구하는 식으로서 f_{scal}은 전압 조절로 인해 변화된 프로세서 처리 속도이며, τ_e는 Job 또는 태스크가 필요로 하는 클럭 사이클을 의미한다. Job 또는 태스크가 f_{scal}의 속도로 수행되고, 스케줄러 호출로 인한 τ_o가 발생하기 때문에 전체 수행시간은 식(3)의 좌변과 같고, 이 시간은 τ_d를 넘지 않아야 한다. 따라서 전체 수행 시간은 전압 조절에 따라 변경된 프로세서의 처리 속도에 따라 결정된다. 결과적으로 프로세서의 f_{scal}은 아래 식(4)를 만족해야 한다.

$$f_{scal} \ge \frac{\tau_e + \tau_o}{\tau_d} \qquad (4)$$

식(4)는 추가로 발생할 스케줄링 오버헤드를 고려했을 때, 전체 시스템의 데드라인을 만족할 수 있는 프로세서 처리 속도인 f_{scal}의 최소 허용범위를 구하는 식이다. 이와 같이 소비되는 전력의 분석을 통해 하나의 Job이 수행될 때 BTPS를 이용하여 전력 소비량의 증감을 분석함으로써 전력 소비가

감소됨을 보일 수 있다. 하나의 Job 또는 태스크가 수행될 때 전력 소비량의 증감을 식으로 표현하면, 식(5)와 같다.

$$\tau_e \times f_{\max} \times (\rho^{p_v} + \rho^{d_v}) - \tau_e \times f_{scal} \times (\rho^{p_s} + \rho^{d_s}) \geq \tau_o \times f_{scal} \times (\rho^{p_j} + \rho^{d_j}) \quad (5)$$

식(5)에서 $f_{\max}$는 프로세서의 최대 처리속도, ρ^{p_v}는 프로세서가 소비하는 전력 소비량, ρ^{d_v}는 디바이스가 소비하는 전력 소비량, ρ^{p_s}는 스케줄링 시 프로세서가 소비하는 전력 소비량, ρ^{d_s}는 스케줄링 시 디바이스가 소비하는 전력 소비량, ρ^{p_j}는 Job 단위 스케줄링 후 프로세서가 소비하는 전력 소비량, ρ^{d_j}는 Job 단위 스케줄링 후 디바이스가 소비하는 전력 소비량이며 각 전력 소비량의 단위는 Watt이다.

식(5) 좌변의 $\tau_e \times f_{\max} \times (\rho^{p_v} + \rho^{d_v})$는 어떠한 기법도 적용하지 않았을 때의 전력 소비량이고, $\tau_e \times f_{scal} \times (\rho^{p_j} + \rho^{d_j})$는 전압 조절 기법을 적용했을 때의 전력 소비량이다. 이 두 식의 전력 소비량 차이는 전압 조절 기법을 사용했을 때 감소하는 전력 소비량에 해당한다. 우변의 $\tau_o \times f_{scal} \times (\rho^{p_s} + \rho^{d_s})$은 Job단위 스케줄링으로 인해 추가로 소비되는 전력량이다. 따라서 좌변이 우변보다 크거나 같다면, 전압 조절 기법을 적용한 시스템이 어떠한 기법도 사용하지 않은 시스템 보다 전력 소모가 적다고 할 수 있다. 식(5)를 이용하여 f_{scal}의 최대값을 구하는 식(6)을 얻을 수 있다.

$$f_{scal} \leq \frac{\tau_e \times f_{\max} \times (\rho^{p_v} + \rho^{d_v})}{\tau_o \times (\rho^{p_s} + \rho^{d_s}) + \tau_e \times (\rho^{p_j} + \rho^{d_j})} \quad (6)$$

어떤 태스크에 속하는 Job 수를 n이라고 가정했을 경우, 식(6)을 전체 Job 수행 시 소모되는 전력량을 계산하는 식으로 확장하면 식(7)과 같다.

$$f_{scal} \leq \frac{n \times \tau_e \times f_{\max} \times (\rho^{p_v} + \rho^{d_v})}{\tau_o \times (\rho_1^{p_s} + \rho_1^{d_s}) + \left((\tau_e + \tau_o) \sum_{k=2}^{n-1} (\rho_k^{p_j} + \rho_k^{d_j}) \right) + \tau_e \times (\rho_n^{p_j} + \rho_n^{d_j})} \quad (7)$$

따라서 식(4)를 이용하여 데드라인을 만족시키는 f_{scal}의 최소 범위를 정하고, 식(7)을 이용하여 전체 전력 소비량과의 비교를 통해 f_{scal}을 구해 실시간성을 보장하면서 전력 소비량을 최소화할 수 있다.

디바이스 매니저(Device Manager)는 전압 조절 매니저에서 계산된 Job의 실행 시간과 주변 디바이스들의 BET(Break Even Time)를 비교하여 후자가 전자보다 짧을 경우 디바이스를 Power-down모드로 전환한다. 만약 BET (단위 : 초)가 Job의 실행 시간보다 길거나 같을 경우, 주변 디바이스의 공급 전력을 Power-down 모드로 변화시키면 잦은 Wakeup으로 인해 전력 소모량이 늘어날 수 있기 때문에 디바이스 상태를 그대로 유지한다.

3.3 저장소 (Repository)

저장소에는 디바이스의 상태와 전력 소비에 대한 정보가 저장된다. 전력 관리 시스템의 동작이 시작되면 가장 먼저 주변 디바이스의 고유 정보를 저장소에 저장한다. 각 디바이스의 BET는 Run, Idle, Wakeup 상태 등에서 요구되는 전력 소모량을 이용하여 얻을 수 있다.

$$BET = \frac{2 \times T_{wakeup} \times (\rho_{wakeup} - \rho_{run})}{\rho_{run} - \rho_{idle}} \quad (8)$$

식(8)에서 T_{wakeup}은 디바이스가 Wakeup 시 걸리는 시간, ρ_{wakeup}, ρ_{run} 그리고 ρ_{idle}은 각각 디바이스가 Wakeup, Run 그리고 Idle 시 소비하는 전력 소비량을 의미한다. 이 식을 통해 BET는 디바이스가 Run상태를 유지하는 시간과 관계가 있다는 사실을 알 수 있다. 그러므로 특정 디바이스를 사용하는 Job의 실행 시간이 BET와 비교하여 더 길수록 디바이스의 공급 전압을 Idle로 변경하는 것이 전력 소모 측면에서 유리하며, Job의 실행 시간이 길어질수록 Power-down 모드를 유지하는 시간이 길어지므로 많은 이득을 얻을 수 있다.

4. 성능 평가

3절에서 제안된 스케줄링 기법은 리눅스 커널 버전 2.6.31에서 시뮬레이션을 통해 테스트되었다. 멀티미디어(MP3)파일의 90%이상이 크기가 16MB이하[8]이므로 테스트에서 사용된 파일의 크기는 16MB이다. MP3 파일의 비트율(Bit Rate)는 최소 32Kbps, 최대 448Kbps이다. 평균 비트율인 240Kbps는 정해진 규격이 아니므로 192kbps를 비트율로 가정하였다. 사용된 멀티미디어 태스크의 기본 마감시간은 비트율의 시간 단위인 1초이며, 1초에 처리되어야 하는 데이터의 크기는 파일의 비트율로 알 수 있다. 태스크가 소모하는 클럭 사이클은 효율을 높이기 위한 캐쉬정책이나 파이프라이닝에 의해 가변적이므로 WCET와 BCET의 균등 분포를 따른다고 가정하였다. 소모되는 전력량을 측정하기 위해 프로세서와 파일의 저장 장치로는 각각 EZ-PXA270 보드[9]에서 사용되는 PXA270 프로세서와 삼성 SUM-UWB(16G)[10]를 모델링하였다(<표 1>참조).

〈표 1〉 EZ-PXA270 세부 사양

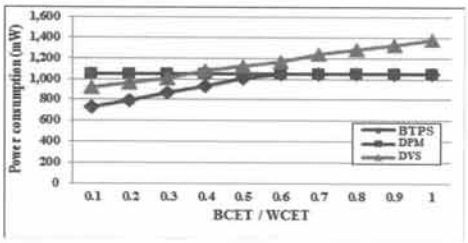
종류	세 부 내 역
규격	90mm X 60mm
MCU	520MHz PXA270 ARM RISC Chip
RAM	64Mbyte SDRAM
ROM1	512Kbyte Boot Flash
ROM2	1Gbyte NAND-Flash
USB	USB Host 1Port, Client 1Port

Sound	AC'97(Line-Out, Head-Phone)
Internal Connector	288-pins Board to board Connector
Extension Connector	160-pins Board to board Connector
JTAG	On-Board JTAG Convertor
Power	권장 DC 5V/1A

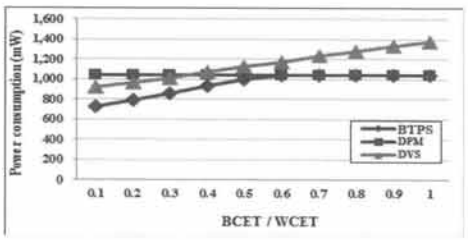
WCET는 모델링한 PXA270 프로세서의 최대 처리 속도를 기준으로 적용하였으며, BCET는 태스크의 실행 사이클을 WCET의 10%~100%까지 변화시키면서 측정하였다. 버퍼 크기는 리눅스에서 사용되는 Mplayer의 버퍼 크기와 동일한 2Mbytes를 사용하였다. (그림 3)~(그림 5)는 단일 태스크만을 수행하였을 때 전력 소비량을 비교한 결과이다.

(그림 3)~(그림 5)에서 볼 수 있는 바와 같이 DPM의 경우, 디바이스가 사용되지 않을 때는 전력 소비를 줄이기 위해 Idle상태로 변경되지만, 프로세서에 대해서는 적용되지 않기 때문에 전력 소비량은 태스크의 BCET에 의해 영향을 크게 받지 않고, 단지 디바이스가 사용된 횟수만 전력 소모량에 영향을 주게 된다. DVS의 경우, BCET가 작을 때, 즉, 필요한 클럭 사이클이 적을 때, 프로세서의 처리 속도를 많이 낮출 수 있기 때문에 DPM보다 성능이 좋다. 하지만, BCET가 커질수록 태스크가 필요로 하는 클럭 사이클이 늘어나기 때문에 프로세서의 처리 속도를 낮출 수 없어 DPM보다 성능이 떨어졌다. 본 논문에서 제안한 BTPS의 경우, BCET가 작을 때, DPM과 DVS를 동시에 사용하기 때문에 전력 소모량이 가장 적었으나, BCET가 큰 경우, 프로세서의 전압 조절이 불가능하여 DPM과 동일한 성능을 보였다. 측정 결과 BTPS의 전력 소비량은 DPM과 DVS와 비교할 때 각각 13.5%와 12.9%감소하였다.

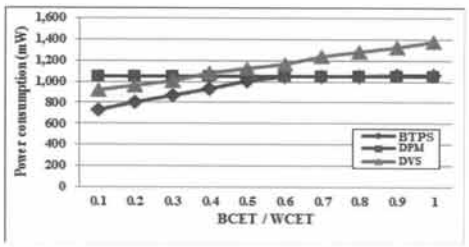
(그림 6)은 멀티 태스크의 전력 소모를 측정한 결과이다. 멀티 태스크 측정을 위해 먼저 한 태스크의 BECT/WCET



(그림 3) 32Kbps

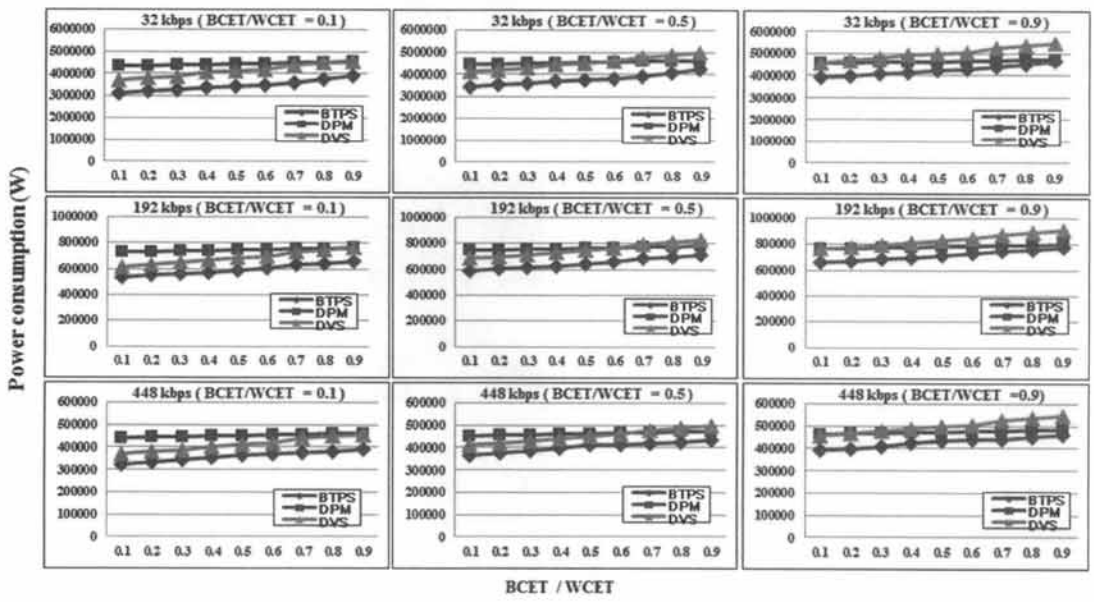


(그림 4) 192Kbps



(그림 5) 448Kbps

값 0.1, 0.5, 0.9에 대해 다른 태스크의 실행 사이클을 0.1부터 0.9까지 변화하면서 전력 소비량을 측정한 하였다. 또한 단일 태스크에 대해 MP3 파일의 비트율을 32kbps, 192kbps, 448kbps로 변경하면서 동일하게 전력 소비량을 측정하였다. 그 결과 BCET가 커질수록 멀티 태스크의 경우도



(그림 6) 멀티 태스크 수행 결과

싱글 태스크와 마찬가지로 BCET가 증가할 경우, 프로세서의 전압 조절이 불가능하여 DPM과 동일한 성능을 보였다. 평균적으로 멀티태스크의 경우의 BTPS 전력 소비량은 DPM과 DVS와 비교할 때 각각 8.48%와 16.58% 좋은 성능을 보였다.감소하였다. 또한, 일반적으로 DPM보다 DVS가 효율적인 전력 관리 기법이지만 본 논문의 실험 결과에서는 전력 소모량의 차이가 크지 않은 것으로 측정되었다. 이는 본 실험이 BCET를 WCET의 10~90%까지 변화하면서 측정한 값의 평균이므로 DVS의 기법이 적용되지 못하는 구간까지 모두 더해지기 때문이다.

5. 결론 및 향후 연구

본 논문에서는 기존의 저전력 기법을 기반으로 주변 디바이스를 고려하여 프로세서의 전압을 조절한 스케줄링 기법을 구현하고, 그 결과를 분석하였다. 분석된 결과를 바탕으로 실시간성을 보장함과 동시에 전력 소비량을 줄일 수 범위에서 프로세서의 전압을 조절하여 두 가지 조건을 모두 만족시키도록 하였다. 또한 디바이스가 연속적으로 사용되는 경우, 이를 고려한 스케줄링을 통해 wakeup 빈도를 줄임으로써 DPM과 DVS와 비교하여 단일 태스크에서는 각각 13.5%와 12.9%, 멀티 태스크에서는 각각 8.48%와 16.58%의 전력 소비량을 감소시켰다. 향후 연구로는 확률 모델을 기반으로 정형화된 디바이스 사용 패턴 정보를 통해 시스템의 전력 소비량을 감소시킬 수 있는 패턴 분석 모듈의 추가가 요구된다.

참 고 문 헌

[1] D. Shin and J. Kim, "Dynamic voltage scaling of mixed task sets in priority-driven systems", IEEE Transactions Computer-Aided Design Integrated Circuits Systems, Vol.25, No.3, pp.438-453, Mar., 2006.

[2] S. Bang, K. Bang, S. Yoon, E. Chung, "Run-time adaptive workload estimation for dynamic voltage scaling", IEEE Transactions Computer-Aided Design Integrated Circuits Systems, Vol.28, No.9, pp.1334-13747, Mar., 2009.

[3] Y. Lu, E.Y. Chung, T.Simunic, L.Benini and G.D. Micheli, "Quantitative Comparison of Power Management Algorithms", Proc. Of Design Automation and Test In Europe, pp.20-26, 2000.

[4] A.Sesic, S. Dautovic and V.Malbasa, "Dynamic power management of a system with a two-priority request queue using probabilistic-model checking", IEEE Transactions Computer-Aided Design Integrated Circuits Systems, Vol.27, No.2, pp.403-407, 2008.

[5] J. Mao, C. Christos, Q. Zhao. "Optimal dynamic voltage scaling in energy-limited nonpreemptive systems with real-time constraints", IEEE Transactions on Mobile Computing, Vol.6, No.6, pp.678-688, 2007.

[6] S. Lee and T. Sakurai, "Runtime voltage Hopping for Low-power Real-time Systems", Annual ACM IEEE Design Automation Conference, pp.806-809, 2000.

[7] Y. Lu and G.D. Micheli, "Comparing system level power management policies", IEEE Design & test of Computers, Vol.18, pp.10-19, 2001.

[8] K.M. Evans and G.H. Kuenning, "A Study of Irregularities in File-Size Distributions", In Proceedings of the 2nd International Symposium on Performance Evaluation of Computer and Telecommunication Systems Paper, July, 2002.

[9] FALINUX Co., Ltd. http://www.falinux.com

[10] SAMSUNG Co., Ltd. http://samsung.com

박 상 오

e-mail : sj1st@cs.cau.ac.kr

2005년 중앙대학교 컴퓨터공학과(학사)

2007년 중앙대학교 컴퓨터공학과(공학석사)

2010년 중앙대학교 컴퓨터공학과(공학박사)

2010년~현 재 중앙대학교 박사 후 과정

관심분야 : 저전력 시스템, NAND 플래시 메모리 파일시스템, 임베디드 시스템 등

이 재 경

e-mail : jklee@cs.cau.ac.kr

2008년 중앙대학교 컴퓨터공학과(학사)

2010년 중앙대학교 컴퓨터공학과(공학석사)

2011년~현 재 삼성 SDS ESDM 개발 지원 그룹 사원

관심분야 : 저전력 시스템, 임베디드 시스템 등

김 성 조

e-mail : sjkim@cau.ac.kr

1975년 서울대학교 응용수학과(학사)

1977년 한국과학기술원 전산과(이학석사)

1977년~1980년 국방과학연수소 연구원

1980년~현 재 중앙대학교 컴퓨터공학부 교수

1987년 Univ. of Texas at Austin(공학박사)

1987년~1988년 Univ. of Texas at Austin(Research Fellow)

1996년~1997년 Univ. California-Irvine(Visiting Professor)

관심분야 : 이동컴퓨팅, 임베디드 소프트웨어, 유비쿼터스 컴퓨팅 등