실시간 임베디드 운영체제 TMO-eCos의 데드라인 기반 CPU 소비 전력 관리

(A Deadline_driven CPU Power Consumption Management Scheme of the TMO-eCos Real-Time Embedded OS)

박 정 화 † 김 정 국 ††
(Jeonghwa Park) (Jungguk Kim)

요 약 본 논문은 실시간 임베디드 OS인 TMO-eCos의 데드라인 기반 CPU 저전력 관리 기법을 다루고 있다. 해당 저전력 관리 기법은 경성 실시간 시스템인 TMO 시스템을 위한 태스크 순차화 기법에서 도출된 스케줄링 시나리오를 사용한다. 본 연구팀에서 개발한 스케줄링 사전 분석기는 주기적으로 동작하는 태스크의 주기, 데드라인, WCET를 기반으로 오프라인 분석을 실시한다. 최종적으로 TMO-eCos 커널은 CPU의 전력 소모를 줄이기 위하여 주기적인 태스크의 데드라인을 위반하지 않는 범위에서 CPU의 속도를 조절하여 시스템에서 사용하는 소비전력은 줄이게 된다. 본 논문은 이와 같은 과정과 실제 실험결과를 기술한다.

키워드 : 저전력, 임베디드, TMO-eCos, 실시간, 데드라인, Real-time, DVS

Abstract This paper presents the deadline driven CPU-Power management scheme for the Real-Time Embedded OS: named TMO-eCos. It used the scheduling scenarios generated by a task serialization technique for hard real- time TMO system. The serializer does a off-line analysis at design time with period, deadline and WCET of periodic tasks. Finally, TMO-eCos kernel controls the CPU speed to save the power consumption under the condition that periodic tasks do not violate deadlines. As a result, the system shows a reasonable amount of power saving. This paper presents all of these processes and test results.

Key words : Deadline-driven power management, TMO-eCos, Hard Real-time, TMO, Dynamic voltage scaling

· 본 연구는 방위사업청, 국방과학연구소의 지원(UD060048AD) 및 지식경제부, 정보통신연구진흥원의 대학 IT연구센터 지원사업의(IITA-2008-C1090-0804-0015) 연구결과로 수행되었음
· 이 논문은 제35회 추계학술대회에서 '실시간 임베디드 운영체제 TMO-eCos의 데드라인 기반 CPU 저전력 관리'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 한국외국어대학교 컴퓨터공학과
vesuvius@hufs.ac.kr
†† 종신회원 : 한국외국어대학교 컴퓨터공학과 교수
jgkim@hufs.ac.kr

논문접수 : 2008년 12월 23일
심사완료 : 2009년 2월 21일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제15권 제4호(2009.4)

1. 서 론

일반적으로 임베디드 시스템에서 프로세서는 가장 주요한 전원소모 장치이다. 이전 연구의 결과를 참고하면 일반 랩탑 컴퓨터의 전원 소모에서 프로세서는 작게는 15%에서 52%정도의 비중을 차지한다. 과거 시스템의 전원관리를 위해 UNIX sleep() 함수와 같이 시스템 idle 시간에 대하여 스케줄링 상에서 구현되는 방법이 일반적 이었지만, 배터리로 동작하는 소형 모바일 장치들의 등장에 따라 보다 다양한 전원관리 방법이 고려되어 왔다. 특히, 저전력 CMOS 설계 기술의 발전함[1,2]에 따라 프로세서의 구현 시 가변 전압과 클럭으로 동작하는 기능이 제공되기 시작했고, 스케줄러 상에서 이를 반영한 보다 효율적인 방법들이 제안되었는데, 그들 중 가장 대표적인 방법은 Dynamic voltage scaling(DVS) 기반의 기술들이다. DVS 방법은 컴퓨팅 시스템의 소모 전원의 관계를 나타내는 다음 식에서 설명될 수 있다.

$$P \cong p_t \cdot C_L \cdot V_{dd}^2 \cdot f_{clk}$$

식에서 p_t는 전원모드 전환 확률이며, C_L은 장착된 콘덴서 값(Loading capacitance), V_{dd} 구동 전압, f_{clk}는 입력 클럭 값이다. 위의 식을 고려해볼 때 시스템의 저전력 동작을 위한 파라미터로서 구동전압과 입력 클럭을 생각해 볼 수 있다. DVS는 필요한 작업에 따라 필요한 전압과 입력 클럭을 가변적으로 운영함으로써, 프로세서의 전원소모를 최적화하는 방법이다. DVS의 방법은 프로세서뿐만 아니라 전원 관리가 가능한 모든 시스템 장치에 적용이 가능하다. 배터리로 운영되는 여러 임베디드 시스템을 위한 DVS기반 스케줄링 알고리즘들이 제안되었다. 이러한 대부분의 스케줄링 알고리즘들은 주기적인 작업의 우선순위(Priority) 또는 WCET에 근거하여 시스템 전원상태를 판단하며, 비주기적인 작업은 응용의 확률기반 모델링에 따라 시스템의 전원상태를 최적화 시키게 된다[3-13].

본 논문에서는 방위사업청과 국방과학연구소 지원으로 개발 중인, 대표적 분산 실시간 객체 모델인 TMO (Time-triggered Message-triggered Object)[14]의 실행을 제공하는 초경량 경성 실시간 임베디드 운영체제 TMO-eCos2.1를 위한 데드라인 기반 CPU 저전력 관리 기법을 제안하고 구현하였다. 본 논문에서 제안하는 전원관리 방법은 기존 DVS 기반 스케줄링 알고리즘들과 비교할 때, 실시간 객체 모델인 TMO의 주기적 태스크들의 보장성 컴퓨팅을 선결 전제로 순차화된 스케줄링 시나리오에 대해 저전력 관리 기법을 적용하는 점에서 기존의 기법들과의 차이를 갖는다. 이러한 차이로 인해 순차화된 시나리오와 데드라인을 기반으로 함으로서 스케줄링 예측성이 가능하고, 태스크 수행 도중 문맥 교환이 일어나지 않아 태스크 수행 단위 전체에 대한 사용 자원 예측을 기반으로 한 저전력 관리의 효과를 얻을 수 있다. 본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로 TMO와 그의 초경량 실시간 임베디드 엔진인 TMO-eCos[15] 및 각 태스크의 스케줄링 가능 여부와 시작시간과 순서를 정할 수 있는 태스크 순차화 기법에 대해 소개한다. 3장에서는 TMO-eCos의 저전력 설계와 구현 방법 및 성능 평가를 기술하고 4장에서는 결론 및 향후 연구 방향에 대해 논의한다.

2. 관련연구

2.1 실시간 객체 TMO

TMO(Time-triggered Message-triggered Object)는 객체지향, 실시간, 분산 컴퓨팅을 한 객체 내에서 모두 제공하는 모델로 보장성 컴퓨팅 패러다임을 지향 한다. TMO는 경성 실시간 응용 프로그램뿐만 아니라 병렬 컴퓨팅 응용 프로그램에서도 사용할 수 있는 유연한 구조를 가졌으며 시스템 설계 시의 정시 보장성을 추구한다.

TMO의 특성을 요약하면 다음과 같다.

- TMO는 ODS(Object Data Store)와 이들을 공유하는 메소드인 객체 내의 스레드 군으로 구성된다.
- TMO의 메소드는 그 특성에 따라, 사전에 주어지는 시간 조건(ACC: Autonomous Activation Candition)에 의해 구동되는 SpM(Spontaneous Method)들과, 분산 환경에서 다른 클라이언트 TMO가 보내는 메시지에 의해 구동되는 SvM(Service Method)들로 분류된다. SpM은 실행 주기와 수행 데드라인이 주어지며 SvM은 수행 데드라인을 가지며 둘은 모두 객체 내의 스레드들이다.
- SpM과 SvM이 객체 내의 공유 데이터에 동시에 접근하여 충돌이 발생할 경우 SpM은 SvM 보다 높은 우선순위를 가지는데, 이것은 설계 시 시간 보장의 개념을 도입하기 위해 SpM과 SvM의 시간 선점을 계층화한 것으로 BCC(Basic Concurrency Constraints)라 한다.

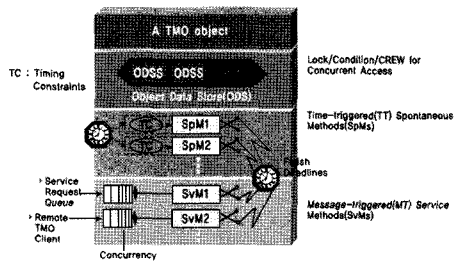

그림 1 TMO 구조

2.2 TMO-eCos2.1

TMO-eCos2.1은 TMO를 활용한 분산 실시간 애플리케이션의 수행을 지원하는 공개 소스 운영체제인 eCos기반의 실시간 커널이다. TMO-eCos2.1은 TMO의 지원을 위해 다음과 같은 기능을 제공한다.

- CPU의 성능 및 클럭 구성에 따라 최저 30us에서 10ms까지의 정밀도로 SpM과 SvM의 마감시간 구동 실시간 스케줄링 기능
- 논리적인 멀티캐스트, IPC 서브시스템을 제공하며, 원격 및 로컬 메시지를 통한 SvM의 구동 기능
- 분산 노드간의 클럭 동기화 기능
- TMOSL[16](TMO Support Library)을 이용한 TMO 기반의 프로그래밍 기능

그림 2는 TMO-eCos 커널의 구조이다.

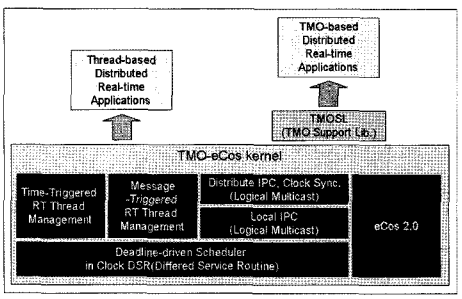

그림 2 TMO-eCos2.1의 구조

2.3 태스크 순차화 기법

본 연구팀에 의해 이전에 개발된 태스크 순차화 기법은[17] 주어진 SpM 들의 주기와 WCET(Worst-case execution time)를 기반으로 경성 실시간 보장성 컴퓨팅이 가능하도록 모든 태스크 인스턴스의 시작 지연 및 선점이 전혀 없도록 태스크 주기들의 최소 공배수인 하이퍼 피리어드(hyper period) 내에서 각 태스크의 초기 위상 (initial offset)을 정하는 기법을 말한다. 단, 이때 주어지는 WCET는 구동 프로세서가 동작할 수 있는 최대 속도에서의 WCET를 말한다.

본 태스크 순차화 기법은 하이퍼 피리어드를 주기의

최대 공약수 크기의 기본 슬롯으로 분할하고, 이러한 슬롯을 기반으로 각 태스크의 초기 위상을 할당하되 태스크들이 슬롯별로 잘 분산되는 시나리오를 먼저 시도하는 방법으로, 시행착오를 최소화하기 위해 재귀적 태스크 클러스터링, 서로소 주기의 제거 등 여러 기법이 활용되었으며, 그 최초 시나리오 도출 시간이 예전의 기법에 비해 상당히 단축된 알고리즘이다.

이러한 태스크의 순차화는 다음과 같은 장점을 갖는다.

- 문맥 교환의 오버헤드가 없다.
- 병행실행 시의 상호배제에 의한 불확실성이 제거되어 보장성 컴퓨팅의 전제조건인 예측성을 제공한다.
- 태스크의 순차적 실행은 태스크별 데드라인 만족 범위 내에서 CPU 속도 조절 등을 통한 저전력 관리 시나리오의 도출이 가능하다.

3. TMO-eCos2.1를 위한 저전력 관리 기법

3.1 TMO-eCos2.1의 저전력 관리 기법

TMO-eCos2.1을 위한 저전력 관리 기법은 2.3절에서 설명한 태스크 순차화 도출 시나리오를 기반으로, 그 수행의 예측이 가능한 SpM에만 적용한다. 즉, 도출된 시나리오는 경성 실시간 시스템을 위한 것으로 모든 태스크 인스턴스 시작 시간에 대한 지연이 일절 허용되지 않고, 다만 각 태스크 마다 주어지는 데드라인 내에서는 태스크의 저속 수행이 허용되는 시스템이다. 이러한 시스템에서의 CPU 저전력 관리는 도출된 시나리오에 의해 모든 태스크를 FTFS(First-triggered First Scheduled)방식으로 스케줄 하되, 도출 시나리오 상에 슬랙 타임이 있을 경우, 허용된 데드라인 기점까지 가능한 태스크의 수행 속도를 늦추는 방식을 사용한다. CPU의 속도 조절이 수행되는 각 시점에서는 각 태스크의 데드라인과 WCET 및 현재 구동중인 태스크의 수, CPU 속도 변경의 필요여부, 다음 동작할 태스크의 구동시간이 활용된다.

저전력 관리는 클럭 인터럽트가 발생하면 구동을 사작할 태스크가 있을 때 다음과 같은 순서로 진행된다.

i) CPU 속도 변경 횟수를 측정하여 시스템 시간을 보정한다.
ii) 각 태스크의 시간 속성 확인 시 구동중인 태스크의 개수와 구동을 기다리는 태스크중 가장 빠른 시작시간을 측정한다.
iii) 구동중인 태스크(새로 구동하는 태스크여야 한다) 가 한 개인 경우 슬랙 타임을 계산하여 최적의 CPU 속도를 계산한다.
iv) 계산된 CPU 속도와 현재의 CPU 속도가 다르다면 CPU 속도를 변경한다.

CPU의 속도 변경이 이루어지는 시점은 각 태스크가 구동되는 시점에서 현재 구동될 태스크 이후에 슬랙 타임이 존재할 경우이다. 따라서 이러한 슬랙 타임의 존재 여부를 확인하고 데드라인을 지키는 범위 안에서 CPU의 속도를 변경하여 최대한 그 동작속도를 늦추며 전력 소모를 줄인다. 슬랙 타임이 존재하지 않을 경우에는 CPU동작 속도는 최대가 된다. 슬랙 타임의 유무를 구분하는 방법은 각 태스크가 클럭 인터럽트로 인해 각각의 상태를 체크하고 동작 유무를 확인할 때 현재 구동할 태스크를 제외한 태스크 중 가장 빠른 구동 시간으로 계산할 수 있다. 이 때 구동되는 태스크의 개수가 한 개 이상일 경우, 스케줄러 분석기에 의해 사전에 정해진 순서로 슬랙 타임 없이 구동된다. 따라서 슬랙 타임이 생기는 경우는 현재 구동할 태스크가 한 개일 경우 다음 구동할 태스크의 시작시간보다 구동될 태스크의 WCET가 작은 경우에 해당한다.

슬랙 타임이 존재하는 경우 CPU 변환 속도는, 구동 프로세서가 동작할 수 있는 최대 속도를 M이라 하고, 다음 스케줄까지의 슬랙 타임을 t라 할 때

$$M \times WCET \div t$$

와 같이 계산할 수 있다. 속도 값이 정확히 정수로 나누어떨어지지 않거나 적용할 수 있는 CPU 속도가 없을 경우에는 한 단계 높은 CPU 속도로 동작하게 하여 데드라인을 준수한다.

그림 3과 그림 4는 SpM1의 슬랙 타임 t가 경우에 따

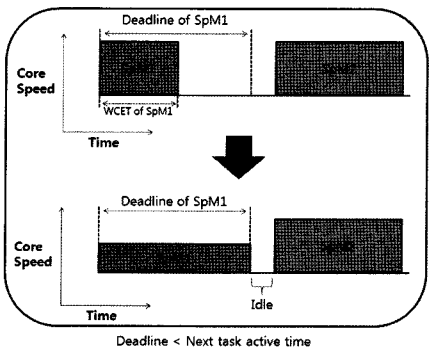

그림 3 Deadline < Next schedule time

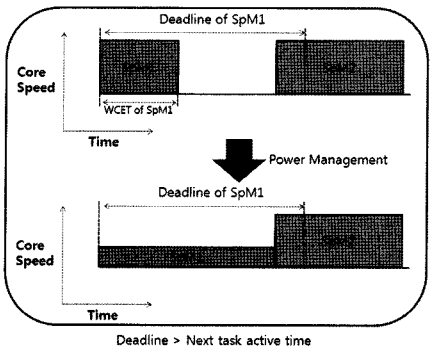

그림 4 Deadline > Next schedule time

라 어떻게 결정되는지를 보여준다. 현재 구동될 SpM의 데드라인보다 추후 구동될 SpM의 구동 시간이 늦을 경우는 현재 구동될 SpM의 데드라인까지의 시간이 t의 값이 되고, 그 반대의 경우 t의 값은 추후 구동할 SpM의 구동시간까지의 시간이 된다. 구동될 태스크가 없는 시스템 유휴 모드의 경우에는 구동될 마이크로프로세서가 지원하는 저전력 지원모드 또는 구동할 마이크로프로세서의 최저 속도록 전환하여 추가적으로 소모되는 전력을 줄일 수 있다.

일반적으로 마이크로프로세서의 속도를 변경할 때 변경한 속도에 맞는 클럭과 공급되는 전압을 안정시키기 위하여 약간의 지연 시간을 지닌다. 이러한 CPU 속도 변경 지연 시간은 시스템에서 어떤 클럭 인터럽트를 사용하는가에 따라 다르므로 마이크로프로세서의 속도를 변경할 때 해당하는 하드웨어의 특성에 따라 지연시간을 보정해 주어야 한다. 이러한 지연 시간으로 인하여 빈번한 마이크로프로세서의 속도 변경은 시스템의 성능을 저하시킬 수도 있다. 따라서 본 저전력 관리 시스템에서는 태스크가 구동될 때와 종료될 경우에 요구되는 CPU 속도와 현재 구동되는 CPU 속도가 다를 경우 이외에는 CPU의 속도 변경을 하지 않는다.

3.2 구 현

TMO-eCos는 다양한 아키텍처를 지원하는데 이중 ARM920T 기반의 SAMSUNG S3C2410A 칩을 사용한 보드에서 구현 및 테스트를 수행하였다.

TMO-eCos는 해당 아키텍처에서 운영체제의 시스템 타이머로 PWM(Pulse Width Modulation) Timer4를 사용한다. 이러한 PWM Timer4는 CPU의 클럭을 특정 분주값으로 나눈 PCLK(Peripheral Clock)를 사용하게 되는데 CPU의 클럭 변경 시 안정된 전압의 공급을 위해 CPU에 공급되는 클럭을 이미 설정된 클럭 시간 (Lock-Time) 만큼 멈추게 한다. 그 후 전압이 안정된 후 해당 속도의 CPU 클럭을 공급하게 된다. 따라서 CPU 클럭의 변경은 시스템 시간에 LockTime 만큼의 지연을 가져오게 된다. 한편 CPU 클럭의 변경은 UART의 baud rate와 Timer4의 동작에도 영향을 미치므로 UART와 Timer4를 포함한 기타 하드웨어를 다시 설정해 주는 것이 필요하다. 따라서 LockTime 레지스터를 1/6000sec로 설정하고 CPU 클럭 변경 횟수를 저장하여 60회 이상의 변경이 이루어졌을 경우 시스템에서 사용하는 기본적인 최소 단위인 1틱(tick : 1/100sec)이상의 차이가 발생한 것으로 1틱을 인위적으로 보정해 주는 방법으로 시스템에서 발생하는 시간의 지연을 방지하였다. 또한 불필요한 CPU 클럭 변경은 시스템에 부하를 발생시키기 때문에 SpM의 구동 시작과 종료 시 필요할 경우를 제외한 다른 부분에서의 CPU 속도 변경은 제외시켰다.

3.3 성능평가

이렇게 구현된 시스템의 전력의 소모는 시스템에서 사용되는 SpM의 숫자와 각 SpM의 시작시간, WCET, 주기, 데드라인 등의 요소에 따라 복합적으로 변하기 때문에 하나의 인자 값만을 변화시켜 시스템의 전력 소모를 비교, 분석하는 데는 한계가 있다. 따라서 본 논문에서는 최대 CPU 속도로 동작할 때 하이퍼 피리어드내의 태스크가 구동할 수 있는 최대 시간과 전체 슬랙타임의 비율을 비교 분석할 수 있는 조건으로 설정하여 이러한 조건이 변화함에 따라 시스템의 전력 소모가 어떻게 변화하는지를 측정하였다.

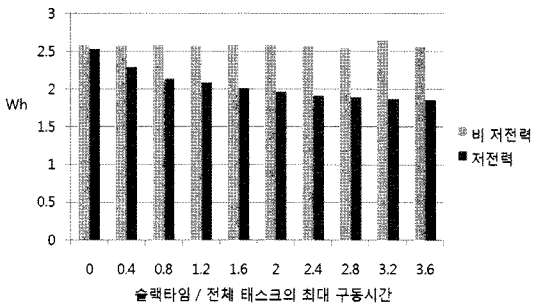

그림 5 소비전력 측정결과

각 태스크의 하이퍼 피리어드 안에서 최대 구동시간은 다음과 같다.

하이퍼 피리어드 ÷ 해당태스크의 주기 × WCET

실험 결과는 그림 5에서 확인할 수 있듯이 전체 태스크의 최대 구동시간과 전체 슬랙타임의 비율에 따라 전력의 소모가 변화함으로 슬랙타임의 존재하지 않을 경우 기존의 전력 소모와 유사한 전력 소모를 보이지만, 그 이외의 경우 슬랙타임의 비율이 높아짐에 따라 전력 소모량은 기존의 저전력을 적용하지 않은 시스템보다 줄어드는 것을 확인할 수 있다. 이와 같은 결과는 순차화된 시나리오와 데드라인을 기반으로 함으로 스케줄링 예측성이 가능하고, 태스크 수행 도중 문맥교환이 일어나지 않아 태스크 수행 단위 전체에 대한 사용 자원 예측을 기반으로 한 저전력 관리가 가능하기 때문이다.

한편 시스템 입력 전압은 동작 클럭 주파수에 거의 비례하기 때문에 전압과 주파수 가변이 모두 가능한 프로세서에서의 소비 전력은 다음과 같이 나타낼 수 있다[18].

전력 ≒ 부하 커패시턴스 × 클럭 주파수^3

따라서 동적 전압 주파수 조절기법(DVS : Dynamic Voltage Scaling)[19]을 지원하는 마이크로프로세서에 본 저전력관리를 적용할 경우 그림 8의 결과보다 많은 양의 전력이 절약될 것으로 예상된다.

4. 결론 및 향후 과제

본 논문에서는 초경량 실시간 운영체제인 TMO-eCos에 태스크 순차화 기법을 바탕으로 한 데드라인 기반 CPU 저전력 관리 방법을 제안하고 구현, 실험하여, 경성 실시간 컴퓨팅이 우선시되는 임베디드 시스템에서의 적합성 및 효율성을 입증하였다.

이와 같은 기법은 실시간 컴퓨팅을 고려하지 않는 일반 저전력 관리 기법에 비해 보장성 컴퓨팅의 제한적 유연성으로 그 전력 관리의 효율성이 떨어지지만, 무기체계와 같이 보장성 컴퓨팅 우선의 임베디드 시스템에서는 그 활용이 기대된다. 향후의 연구는 본 논문에서 활용한 태스크 순차화 기법을 저전력 관리를 고려한 시나리오를 사전에 도출하도록 개선하여 활용하는 것이다.

참 고 문 헌

[1] BURD, T. D., AND BRODERSEN, R. W. Energy efficient CMOS microprocessor design. In Proceedings of the 28th Annual Hawaii International Conference on System Sciences. Volume 1: Architecture(Los Alamitos, CA, USA, Jan. 1995), T. N. Mudge and B. D. Shriver, Eds., IEEE Computer Society Press, pp. 288-297.

[2] CHANDRAKASAN, A. P., SHENG, S., AND BRODERSEN, R. W. Low power CMOS digital design. IEEE Journal of Solid-State Circuits, 27, 4(Apr. 1992), 473-484.

[3] GOVIL, K., CHAN, E., AND WASSERMAN, H. Comparing algorithns for dynamic speed-setting of a low-power CPU. In Proceedings of the First Annual International Conference on Mobile Computing and Networking (Mobi-Com'95), 1995, 13-25.

[4] GRUIAN, F. Hard real-time scheduling for low energy using stochastic data and DVS processors. In Proceedings of the International Symposium on Low-Power Electronics and Design ISLPED'01 (Huntington Beach, CA, Aug. 2001).

[5] LORCH, J., AND SMITH, A. J. Improving dynamic voltage scaling algorithms with PACE. In Proceedings of the ACM SIGMETRICS 2001 Conference(Cambridge, MA, June 2001), pp. 50-61.

[6] MOSSE, D., AYDIN, H., CHILDERS, B., AND MELHEM, R. Compiler-assisted dynamic power-aware scheduling for real-time applications. In Workshop on Compilers and Operating Systems for Low-Power(COLP'00)(Philadelphia, PA, Oct. 2000).

[7] GPERING, T., AND BRODERSEN, R. Energy efficient voltage scheduling for real-time operating systems. In Proceedings of the 4th IEEE Real-Time Technology and Applications Symposium RTAS'98, Work in Progress Session (Denver, CO, June 1998).

[8] GPOUWELSE, J., LANGENDOEN, K., AND SIPS, H. Energy priority scheduling for variable voltage processors. In Proceedings of the International Symposium on Low-Power Electronics and Design ISLPED'01 (Huntington Beach, CA, Aug. 2001).

[9] SWAMINATHAN, V., AND CHAKRABARTY, K. Real-time task scheduling for energy-aware embedded systems. In Proceedings of the IEEE Real-Time Systems Symp. (Work-in-Progress Session) (Orlando, FL, Nov. 2000)

[10] WEISER, M., WELCH, B., DEMERS, A., AND SHENKER, S. Scheduling for reduced CPU energy. In Proceedings of the First Symposium on Operaing Systems Design and Implementation (OSDI) (Monterey, CA, Nov. 1994), pp. 13-23.

[11] AYDIN, H., MELHEM, R., MOSSE, D., AND MEJIA-ALVAREZ, P. Dynamic and aggressive scheduling techniques for power-aware real-time systems. In Proceedings of the 22nd IEEE Real-Time Systems Symposium (RTSS'01), 2001.

[12] ISHIHARA, T., AND YASUURA, H. Voltage scheduling problem for dynamically variable voltage processors. In Proceedings of the International Symposium on Low Power and Electronics Design (ISLPED'98), 1998, 197-202.

[13] KATCHER, D., ARAKAWA, H., AND STOSNIDER, J. Engineering and analysis of fixed priority schedulers. IEEE Transactions on Software Engineering, 19, 9 (1993), 920-934.

[14] Kim, K.H. and Kopetz, H., "A Real-Time Object Model RTO.k and an Experimental Investigation of Its Potentials," Proc. 18th IEEE Computer Software & Applications Conference, pp. 392-402, November 1994.

[15] 김광, "TMO-eCos : 분산 실시간 객체 모델을 지원하는 eCos 기반의 마이크로 운영체제", 한양대학교 컴퓨터공학과 박사학위논문, 2005년 12월.

[16] J.G. Kim, M.H. Kim, B.J. Min, and D.B Im, "A Soft Real-time TMO platform - WTMOS - and Implementation Techniques," Proc. 1st IEEE International Symposium on object-oriented Real-time Distributed Computing, pp. 254-264, April 1997.

[17] 김현주, "경성 실시간 TMO 시스템을 위한 효율적인 태스크 순차화 기법", 한국외국어대학교 컴퓨터공학과 박사학위논문, 2008년 7월.

[18] 이정환, 김명준, "Design and Implementation for Porable Low-Power Embedded System", 한국정보과학회논문지, 제13권 제7호 pp.455, 2007년 12월.

[19] T. Sakurai and A. Newton. Alpha-power Law MOSFET Model and Its Application to CMOS Inverter Delay and Other Formulas. IEEE Journal of Solid-State Circuits, 25(2), pp. 584-593, April 1990.