논문 2005-42SD-5-9

로봇을 위한 운영체제 수준의 동적 전력 관리

(Operating System level Dynamic Power Management for Robot)

최 승 민*, 채 수 익**

(Seungmin Choi and Sooik Chae)

요 약

본 논문에서는 가정용 로봇에 적용 할 수 있는 운영체제 수준의 동적 전력 관리 기법인 EAJS(Energy-Aware Job Scheduler)를 제안한다. EAJS는 workload가 일정하지 않은 IO 장치를 사용하는 job들을 스케줄하여 장치의 idle주기를 가능하면 연속적이게 만든 후, 장치를 *shutdown* 시켜서 에너지 소모를 줄이는 능동적인 저전력 스케줄러이다. EAJS는 기존의 저전력 스케줄러와 달리, IO 장치를 사용하는 job의 workload, job의 buffering에 사용 가능한 메모리의 크기, buffering으로 인해 발생하는 시간 지연 등을 동시에 고려하여 job을 스케줄 하기 때문에, 에너지를 절약하면서도 시스템의 성능 저하를 최소화 할 수 있다. EAJS의 prototype을 본 연구에서 개발한 가정용 로봇인 AFM(Autonomous Family Machine)에 구현하였으며, H.263 인코더를 수행시키는 실험을 통해 무선랜과 DSP의 에너지 소모가 최대 44% 가량 줄어듬을 확인 하였다.

Abstract

This paper describes a new approach for the operating system level power management to reduce the energy consumed in the IO devices in a robot platform, which provides various functions such as navigation, multimedia application, and wireless communication. The policy proposed in the paper, which was named the Energy-Aware Job Schedule (EAJS), rearranges the jobs scattered so that the idle periods of the devices are clustered into a time period and the devices are shut down during their idle period. The EAJS selects a schedule that consumes the minimum energyamong the schedules that satisfy the buffer and time constraints. Note that the burst job execution needs a larger memory buffer and causes a longer time delay from generating the job request until to finishing it. A prototype of the EAJS is implemented on the Linux kernel that manages the robot system. The experiment results show that a maximum 44% power saving on a DSP and a wireless LAN card can be obtained with the EAJS.

Keywords: Energy, Energy-Aware Job Scheduler, DPM, Robot, Operating System

I. 서 론

최근 다양한 형태로 연구되는 지능형 로봇은 이동을 하면서 다양한 기능을 수행하기 때문에 전지로부터 동작전원을 공급 받는다. 그러나 제한된 용량의 전지로부터 장시간 로봇을 작동 시킬 수 있는 충분한 에너지를 공급 받는데 한계가 있기 때문에, 로봇을 설계하는 과정에서 전지의 용량을 증가시키는 노력과 더불어, 효율적으로 전력을 관리하는 방법에 대해 연구하는 일은 매우 중요하다.

전력 관리 기법은 칩 설계 과정에서 합성이나 컴파일러 설계 과정에서 명령어 스케줄링 등을 통해서 구현될 수 있는 정적인 기법과, 운영 체제 (Operating System, OS)나 응용프로그램이 수행 되는 도중에 적용될 수 있는 동적인 기법으로 나눌 수 있다[1][2]. 본 논문에서는 자율 로봇처럼 사용자와의 상호 작용이 많은 대화형 시스템에서 효과적인 OS 수준의 동적 전력 관리 기법에 대해 중점적으로 논의 한다.

본 논문에서는, 새롭게 개발한 가정용 자율 로봇인 AFM(Autonomous Family Machine)에 적용 할 OS 수준의 전력 관리 기법으로 저전력 job 스케줄러를 제안한다. 이 기법은 DSP나 무선랜처럼 workload가 일정하

* 정회원 한국전자통신연구원 지능형로봇연구단
(Electronics and Telecommunications Research Institute Intelligent Robot Research Divison)
** 정회원 서울대학교 전기.컴퓨터공학부
(School of Electrical and Computer Engineering, Seoul National University)
접수일자: 2004년8월9일, 수정완료일: 2005년4월18일

지 않은 IO 장치를 사용하는 job들을 재배치하여 장치의 idle 주기를 가능하면 연속적으로 길게 모은 후, 장치를 shutdown 시켜서 에너지 소모를 줄이는 능동적인 저전력 스케줄러이다. 이 기법이 기존의 저전력 스케줄러와 다른 점은 장치를 사용하는 job의 workload, job의 buffering에 사용 가능한 메모리의 크기, buffering으로 인해 발생하는 시간 지연 등을 동시에 고려하여 job을 스케줄함으로써 시스템의 성능저하를 최소화 하면서도 에너지 소비를 줄일 수 있다는 점이다.

II. Case Study : AFM 로봇 플랫폼

그림 1은 본 연구에서 개발한 전력 관리가 가능한 가정용 로봇인 AFM(이하 AFM)의 사진이다. AFM은 무선통신 및 시스템 제어를 담당하는 AHPB (AFM Hardware Platform Board), 스텝 모터의 제어 역할을 하는 MCB (Motor Control Board), 에너지 측정 시스템인 EMB (Energy Measurement Board)등으로 구성된다. 다음의 각 절에서는 앞에서 열거한 AFM의 구성요소들에 대해 간략히 기술한다.

1. AHPB(AFM Hardware Platform Board)

그림 2는 AHPB의 Block Diagram과 사진을 보여 준다. AHPB는 각 주변 모듈의 제어를 위한 저전력 RISC 프로세서인 SA1110[3] MPU, 영상과 음향 압축을 위한 TM1300[4] DSP, HOST PC와의 무선 통신을 위한 Orinoco 무선랜 카드 등으로 구성된다. 또한, AHPB는 V320USC PCI 브릿지를 탑재하여 고속으로 주변 장치와 통신이 가능한 PCI 인터페이스를 제공하며, 이를 통해 주변 모듈을 다양하게 확장 할 수 있다.

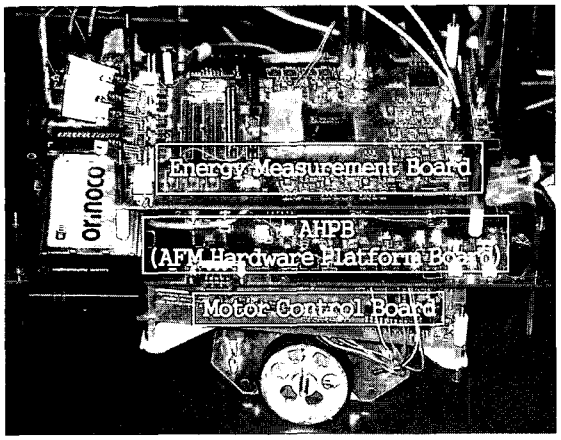
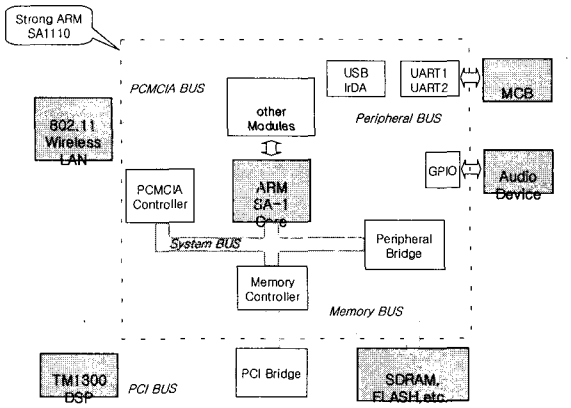
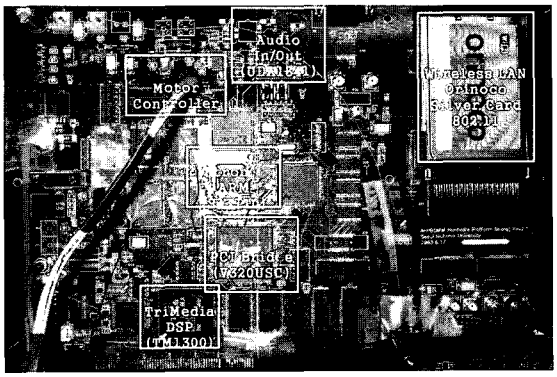


그림 1. AFM 로봇 플랫폼
Fig. 1. AFM Robot Platform.

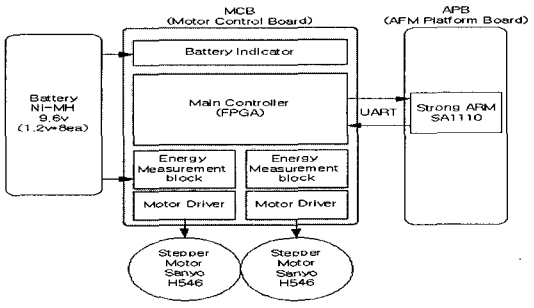


(a) AHPB Block Diagram

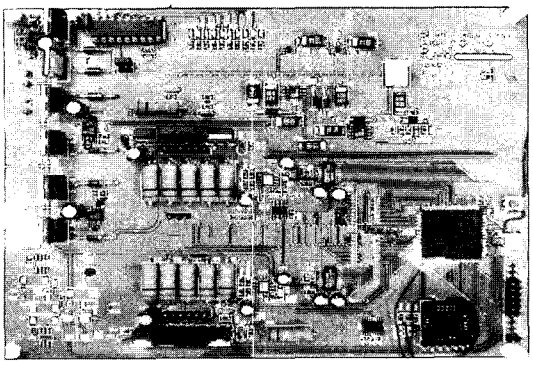


(b) AHPB의 PCB 사진

그림 2. AFM Hardware Platform Board의 PCB
Fig. 2. PCB of AFM Hardware Platform Board.



(a) MCB Block Diagram



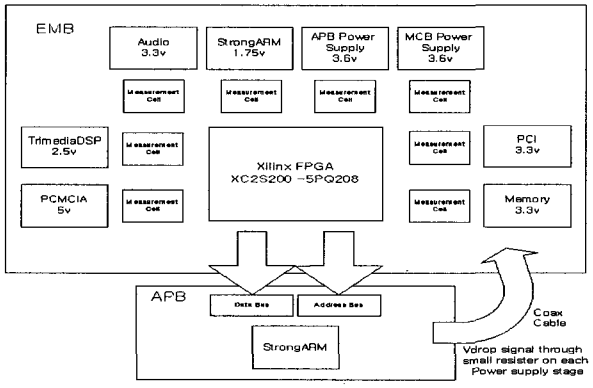
(b) MCB의 PCB

그림 3. Motor Control Board
Fig. 3. Motor Control Board.

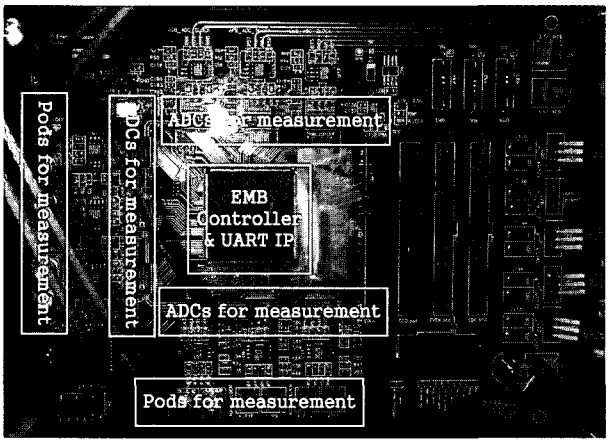
그러나 제한된 전력과 AHPB의 하드웨어 구성 요소만으로는 AFM의 주요 기능인 영상 및 음향 인식, 감성 인식, 항법 등의 복잡한 알고리즘을 원활히 수행할 수 없기 때문에, 연산량이 많은 응용 프로그램들은 HOST PC에서 실행 되도록 설계 하였다. 따라서 AFM의 주요 작업은 HOST PC에서 처리될 음성 및 영상 등의 생 자료(raw data) 수집 및 압축, 압축된 자료를 무선 전송, HOST PC로부터 결과를 수신하여 시스템에 적용하는 등의 일이다.

2. MCB(Motor Control Board)

AFM의 주행을 위해 그림 3과 같은 스텝모터 제어 모듈인 MCB를 제작하였다. MCB의 메인 컨트롤러인 Spartan FPGA에는 UART 통신, 모터제어, 전력 측정 등을 수행하는 로직이 합성 되어 있다. 따라서 AHPB와 MCB는 UART 인터페이스를 통해서 모터의 이동속도와 방향, 모터의 소모 전력, 전지의 남은 용량 등에 대한 명령과 정보를 주고받을 수 있다.



(a) EMB Block Diagram



(b) EMB의 PCB 사진

그림 4. Energy Measurement Board
Fig. 4. Energy Measurement Board.

3. EMB(Energy Measurement Board)

AFM은 전력 관리 기능을 구현하기 위해, 설계 단계에서 전원을 각 하드웨어 블록 별로 분리 시켰으며, SA1110 MPU가 각 하드웨어 블록의 전력 상태를 제어할 수 있게 하였다. 또한 그림 4와 같이 8개의 10bit ADC를 통한 에너지 측정 시스템인 EMB를 개발하여, 최대 8개의 구성 하드웨어가 소모하는 전력을 동시에 측정 할 수 있다. EMB에 의해 측정된 각 하드웨어 콤포넌트들의 소모 전력은 UART를 통해 AHPB나 HOST PC로 전송되며, 에너지 계산 프로그램에 의해 분석된다.

Ⅲ. AFM의 전력 관리 정책

1. 배경 지식

가. 동적 전력 관리(Dynamic Power Management)

한 시스템 내에서 특정 IO 장치들은 일정 시간 동안만 동작(busy)하고, 그 외의 시간에는 휴지(idle) 상태로 남아서 불필요한 전력을 소모한다. 이것은 무선랜이나 DSP 혹은 하드디스크와 같이 workload의 양이 일정하지 않은 장치들에게서 자주 발생되는 일인데, 만약 workload가 없을 때 장치를 sleep 상태로 만들게 되면 상당량의 에너지 절약 효과를 볼 수 있을 것이다.

그림 5는 전력 관리에 대한 기본 개념을 보여 준다. 그림 5에서 장치가 처리해야 할 사용 요청(request)이 있는 경우에 그 장치의 동작 상태는 흑색인 busy이며, 그렇지 않은 경우 백색의 idle이다. 따라서 t_1과 t_3사이에서 장치의 동작 상태는 idle이며, 불필요한 전력 소모를 막기 위해서 idle인 장치를 sleep 상태로 만들 수 있다. 그러나 전력 상태를 변화 시킬 때는 그림5와 같이 shutdown delay (t_{sd})와 wakeup delay(t_{wu})라는 시간상의 지연뿐만 아니라 전력 상태의 천이 과정에서 발생하

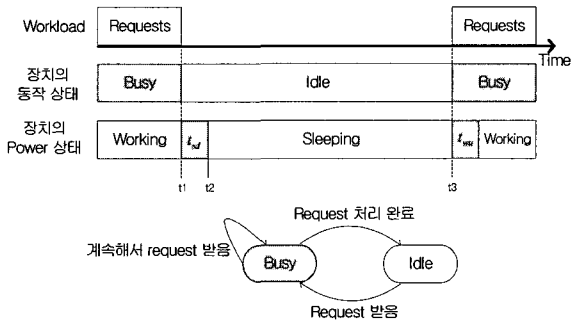
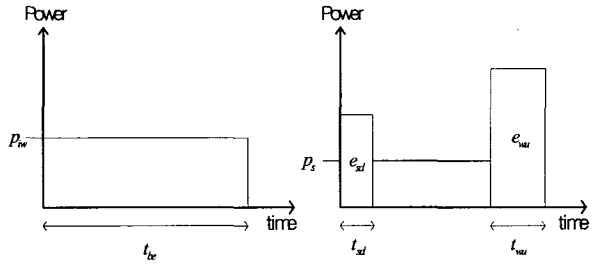


그림 5. Idle 구간에서 장치의 sleep
Fig. 5. Sleep of device for its idle period.

표 1. Symbol들의 의미
Table 1. Meaning of symbols defined here.

Symbol	의 미
t_{sd} (t_{wu})	Shutdown (wakeup) 지연
t_o	전력 상태 천이 지연 ($t_{sd} + t_{wu}$)
$t_{be,k}$	장치 d_k 의 break-even time
T_u	Task u의 장치사용job의 burst 주기
p_{iw} (p_s)	Idle working(sleeping)상태의 소모전력
$p_{iw,k}$ ($p_{s,k}$)	장치 d_k 의 idle working(sleeping)소모 전력
$p_{bw,k}$	장치 d_k 의busy working 소모 전력
p_{nKbuf}	n Kbyte buffer memory 소모 전력
e_{sd} (e_{wu})	Shutdown (wakeup) 소모 에너지
e_o	전력 상태 천이 소모 에너지($e_{sd} + e_{wu}$)
E_{split}	요청 발생 즉시 처리할 때 소모 에너지
E_{burst}	요청을 burst로 처리 할 때 소모 에너지
E_{Buf}	Buffer memory가 소모하는 에너지
BW_k	장치 d_k 의 자료 처리 능력(bps, fps, etc)
$S_{k,u}$	Task u가 d_k 에 한번에 처리를 요청 하는 자료의 크기 (Byte)
$N_{k,u}$	Task u가 초당 $S_{k,u}$ 크기의 자료를 요청 하는 횟수



(a) idle working 유지 (b) shutting down

그림 6. Workload가 없을 때 장치의 전력 관리
Fig. 6. Power management for devices when there aren't workloads.

는 추가적인 에너지 소모 때문에, 일정 시간동안 장치가 idle 상태로 있을 때 만 sleep 시켜야 실질적인 에너지를 이득을 볼 수 있다. 이렇게 전력 관리를 통해 에너지 이득을 볼 수 있는 최소의 idle 길이를 Break-Even Time(t_{be})이라고 하는데, 이 시간은 장치가 가지고 있는 고유한 특성 상수이다[5].

그림 6은 장치가 idle일 때, 전력 상태를 계속 idle working(p_{iw})으로 유지하는 경우와 shutting down (p_s) 하는 경우의 소모 전력을 보여 준다. 이때, t_{be}는 그림 6의 두 경우에서, 각각 계산된 소모 에너지의 양이 동일할 때의 시간이 된다. 장치의 idle working과 sleeping 상태의 소모 전력을 각각 $p_{iw}, p_s (p_{iw} \geq p_s)$, 장치의 shutdown과 wakeup에 소요되는 시간과 소모되는 에너지를 t_o, e_o라 하면, t_{be}값은 다음과 같이 계산 된다.

$$p_{iw} \times t_{be} = e_o + p_s \times (t_{be} - t_o) \quad t_{be} > t_o$$
$$t_{be} = \max\left(\frac{e_o - p_s \times t_o}{p_{iw} - p_s}, t_o\right) \tag{1}$$

식(1)에서 알 수 있듯이 t_{be}값은 각 장치에 의존적인 값으로, 수행되는 프로그램이나 전력 관리 정책과는 무관한 값이다. 장치가 최소한 t_{be}이상의 시간을 idle로 있었거나 있을 것이라는 예측이 가능 한 장치를 shutting down 하면 에너지 절약 효과를 얻을 수 있다. 하지만 그 장치가 언제 다시 쓰이게 될지를 정확하게 예측 하기란 불가능 하므로, wakeup delay에 의한 어느 정도의 성능 저하는 피할 수가 없다.

나. JOB의 정의

본 논문에서 사용된 job이란 용어는 특정 작업을 끝내는 명령어들의 집합으로써 특정 시각에 시작 되도록 스케줄 가능한 최소의 단위이다. 그리고 DPM이 적용되는 장치(DSP, 무선랜)를 사용하는지 여부에 따라 process를 여러 개의 job으로 구분 하였으며, job들 간에는 선행(precedence)관계의 제약 조건이 존재하기도 한다. 예를 들어, MP3 디코더 process는 무선랜을 통해 mp3 data를 수신 받는 job, 수신 받은 mp3 data를 디코딩하여 wave로 변환 시키는 job, wave data를 DAC로 출력 하는 job 등으로 구성 되며, 디코딩 job보다 mp3를 무선으로 수신하는 job이 선행되어야 하는 제약이 존재 한다. 이것은 job을 스케줄 할 때 꼭 지켜져야 하는 제약조건으로 작용한다.

다. 기존의 동적 전력 관리 기법

지금까지 연구 된 OS수준에서의 동적 전력 관리 기법은 크게 2종류로 분류 할 수 있는데, 첫 번째는 I/O 장치의 idle 주기를 예측하여 shutdown을 결정하는 기법이며[6]~[9], 두 번째는 job 스케줄링을 통해 장치의 idle주기를 연속적으로 만든 후 shutdown 하는 기법이

다[10]~[12]. 첫 번째 기법은 장치 사용에 대한 요청이 언제 발생 할지를 사전에 알기 힘든 대화형 시스템에 적합한 기법으로써 응용프로그램을 수정하지 않아도 되는 장점이 있다. 그러나 예측의 정확도가 낮을 경우 시스템의 성능이 떨어지며, 큰 에너지 절약 효과를 기대 할 수 없다. 반면 스케줄링을 통해 shutdown의 기회를 늘이는 기법은, idle 주기의 예측 기법에 비해 상대적으로 많은 에너지를 절약 할 수 있지만 사전에 장치 사용에 대한 정보를 알고 있어야 하며, 응용프로그램을 수정해야 하는 단점이 있다.[13]

2. Energy-Aware Job Scheduler

이번 절에서는 기존의 저전력 스케줄러가 고려하지 않았던, buffer의 소모 에너지, runtime에 스케줄러가 사용 가능한 buffer의 크기, 그리고 buffering에 의해 발생하는 시간지연 등을 동시에 고려하여, 시스템의 성능을 저하시키지 않고, 실질적으로 에너지를 절약 할 수 있는 저전력 스케줄러인 Energy-Aware Job Scheduler (EAJS)를 제안한다. 저전력 스케줄 기법은 job의 수행 순서를 재배치하기 위해 job이 처리할 data를 buffering 할 수 있는 메모리 buffer를 필요로 한다. 그런데, buffer가 소모하는 에너지는 스케줄을 통해 절약되는 에너지와 trade-off 관계에 있기 때문에, 실질적인 에너지 절약 효과를 얻기 위해서는 buffer가 소모 하는 에너지도 고려되어야 한다. 또한 스케줄에 사용 가능한 buffer의 크기가 시스템의 상황에 따라 runtime에 변하기 때문에, 시스템의 성능을 저하 시키지 않으면서 에너지를 절약하기 위해서는 현재 사용 가능한 버퍼의 크기를 고려하여 job을 스케줄 해야 한다. 마지막으로, job의 수행 순서를 바꾸게 되면 사용자가 원하는 deadline을 만족하지 못할 가능성이 있기 때문에, 시간 제약에 대한 고려도 필요하다.

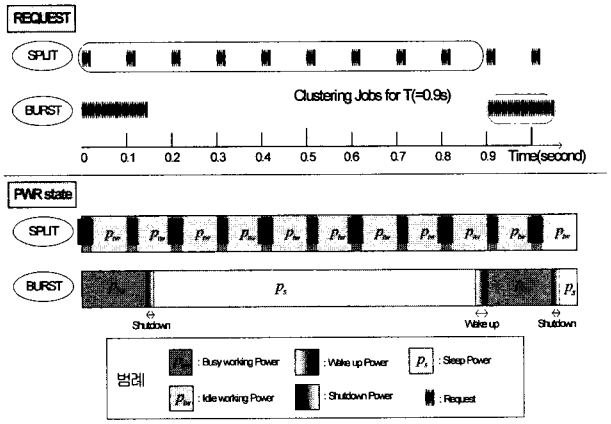


그림 7. 스케줄 방식에 따른 무선랜의 호출 패턴 및 전력 상태

Fig. 7. Requests calling pattern and power state of wireless LAN at split and burst scheduling policy.

그림 7은 AFM에서, fps가 10인 H.263 인코더의 output을 HOST PC로 전송하는 job이 split과 burst의 두 가지 방식으로 스케줄 될 때, 무선랜이 호출되는 빈도와, 무선랜의 전력 상태를 보여준다. 그림에서 split 스케줄의 경우 무선랜이 0.1초마다 호출 되는데, 이것은 무선랜의 t_{be}값인 0.67초보다 작은 값이기 때문에, 정확하게 idle 주기를 예측 했다고 하더라도, 장치를 shutdown 하는 것은 오히려 에너지 소모를 증가하게 만든다. 반면, 그림에서 burst 스케줄의 경우 무선랜이 0.75초마다 호출되므로, shutdown을 통해 에너지를 절약 할 수 있다. 다음의 식들은 그림 7에서 split과 burst 스케줄에서 burst 주기인 T 시간 동안 무선랜이 소모하는 에너지에 대한 계산식이다.

$$E_{split} = \frac{T \times S \times N}{BW} \times p_{bw} + \left(T - \frac{T \times S \times N}{BW}\right) \times p_{iw} \quad (2)$$

$$E_{burst} = \frac{T \times S \times N}{BW} \times p_{bw} + \left(T - \frac{T \times S \times N}{BW} - t_o\right) \times p_s + e_o + E_{Buf} \quad (3)$$

$$E_{Buf} = \left\{Integer\left(\frac{S \times N}{nKbyte}\right) + 1\right\} \times T \times p_{nKbuf} \quad (4)$$

식 (2)~(4)에 사용 된 기호들은 표 1에 정리되어 있으며, 장치 이름은 생략하였다. 식 (2), (3)의 공통으로 포함된 첫 항은 T 시간 동안 request를 처리하는데 소모되는 에너지 값이다. 식 (2)에서 두 번째 항은request가 없을 때에도 전력 관리를 하지 않고 무선랜을 idle working 상태로 유지 시켰을 때 소모되는 에너지이다. 그리고 식 (3)에서 두 번째 항은 sleep구간에서 소모되는 에너지이고, e_o는 shutdown과 wakeup과정에서 소모되는 상태 천이 에너지이다.

식 (3)의 마지막 항인 E_{Buf}는 식 (4)로 계산 할 수 있다. E_{Buf}는 burst 스케줄을 위해 대기하는 request가 buffering 되는데 쓰이는 메모리의 소모 에너지이다. 현재 AFM이 스케줄에 사용하는 buffer 메모리는 bank 별로 전력 관리가 불가능한 SDRAM이기 때문에, E_{Buf} 값을 0으로 계산 하였다. 그러나 향후 진행 할 연구에서는 AFM을 SoC로 구현하고, n KB 단위로 공급 전원의 조절이 가능한 On-Chip SRAM을 buffer로 사용하여, 스케줄로 절약한 에너지와, buffering으로 소모되는

에너지의 trade-off 관계를 식 (2), (3)을 통해 분석 할 수 있다. Buffer 설계시, on/off가 가능한 SRAM의 크기인 n 값이 작을 경우 전력조절 회로를 구현하는데 많은 비용이 소모되며, n 값을 너무 크게 했을 경우 SRAM의 저장 공간과 소모되는 에너지가 비효율적으로 활용될 수 있다. 따라서 전력 조절이 가능한 SRAM buffer의 크기를 합리적으로 선택하는 작업이 선행 되어야 한다.

EAJS가 에너지 최적의 스케줄을 찾기 위해서 runtime에 고려해야 할 사항은 slpit과 burst 스케줄이 각각 소모하는 에너지, burst 스케줄이 사용 가능한 buffer 크기, process의 deadline등의 세 가지이며, 이들은 각각 식 (5)~(7)의 관계식으로 나타낼 수 있다. 각각의 식에서 우변의 상수 값(소문자)은 표 2로부터, 변수값(대문자)은 runtime에 process와 OS로부터 얻을 수 있다.

$$T_{eq} = \frac{e_o - t_o \times p_s}{p_i - p_s} \times \frac{1}{1 - \frac{N \times S}{BW}}; E_{burst} = E_{split} \quad (5)$$

$$T_{buf} = \frac{Buffer\ Limit}{S \times N}, \text{ 메모리제약} \quad (6)$$

$$T_{dl} = Deadline, \text{ 시간제약} \quad (7)$$

예제 : 표 2에서 측정한 무선랜의 특성 파라미터값과, Bandwidth=5.5Mbps, fps=10, deadline =4를 대입하여 S와 T의 관계식을 풀면 식 (5′~7′)과 같으며, 이 식들을 그래프로 나타내면 그림 10과 같이 T가 S에 따라 I~III의 세 영역으로 구별된다.

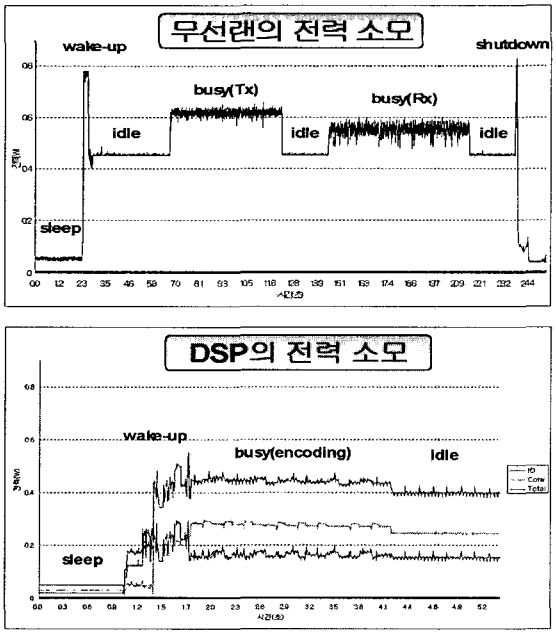


그림 8. 무선랜과 DSP의 전력 소모
Fig. 8. Power dissipation of WLAN and DSP.

그림 10에서 I, II의 경우는 burst 스케줄 방식이 유리하며, III의 경우는 split 스케줄 방식이 유리하다. 이것은 스케줄별 에너지 소모량을 나타낸 그래프인 그림 9에서 두 평면의 교점의 그래프인 식 (5)에 의해 결정된다. 예제에서는 runtime에 process의해 결정되는 S값에 따라 에너지 최적의 T값이 달라지게 되는데 I의 경우 응용 프로그램이 S값을 $0 < S \leq 26KB$범위로 생성 하므로, 이때 에너지 최적의 burst 주기 T는 시간제약인 4초가 됨을 알 수 있다. 같은 방식으로 II의 경우 S값이 $26KB < S \leq 45.5KB$일 때, T는 식 (4′)가 된다. 반면 III의 경우처럼, S가 45.5KB보다 클 경우 스케줄러는 burst 스케줄링을 하지 않고 무선랜을 사용을 요청하는 job이 생성 되는 즉시 수행 시키는 split 스케줄링을 선택 한다.

$$T = \frac{e_o - t_o \times p_s}{p_i - p_s} \times \frac{1}{1 - \frac{N \times S}{BW}} = \frac{0.3 - 0.5 \times 0.05}{0.46 - 0.05} \times \frac{1}{1 - \frac{10 \times S}{5.5Mbps}}$$

$$= 0.67 \times \frac{1}{1 - \frac{10 \times s}{720896}} \quad (5')$$

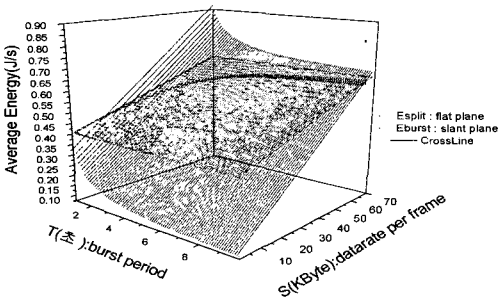
$$T = \frac{BufferLimit}{S \times N} = \frac{1 \times 1024 \times 1024}{S \times 10} = \frac{104857.6}{S} \quad (6')$$

$$T = Deadline = 4 \quad (7')$$

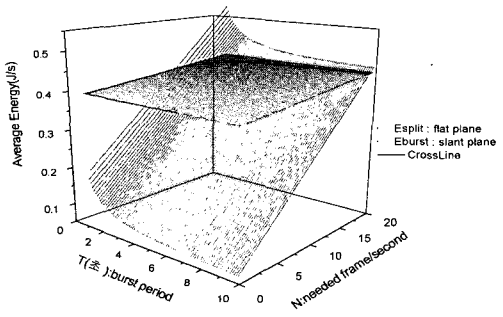
앞의 예제는 DSP에도 동일하게 적용된다. 그러나 무선랜과 DSP를 동시에 사용하는 H.263 인코더와 같은 process의 경우 job들간에 수행 순서의 제약이 존재하기 때문에, 한 장치를 위해 구한 T값을 다른 장치의 시간 제약(Deadline)으로 대입하여 나머지 장치의 T값을 구하는 순차적 방법을 이용한다. 그림 11은 이러한

표 2. 무선랜과 DSP의 특성 파라미터 측정치
Table 2. Measured characteristic parameters of DSP and WLAN.

장치명 / 파라미터	TM1300 DSP	Orinoco 무선랜
p_{iw}	0.40	0.46
p_s	0.05	0.05
p_{bw}	0.44	0.65
t_0	0.59	0.5
e_0	0.17	0.3



(a) 무선랜(Wireless LAN)



(b) DSP

그림 9. Split과 burst 스케줄의 에너지 소모
Fig. 9. Energy consumption of devices at split and burst scheduling.

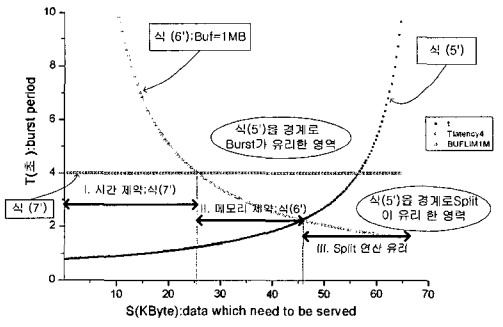


그림 10. S에 따른 에너지 최적의 T 선택
Fig. 10. Energy optimized T for S.

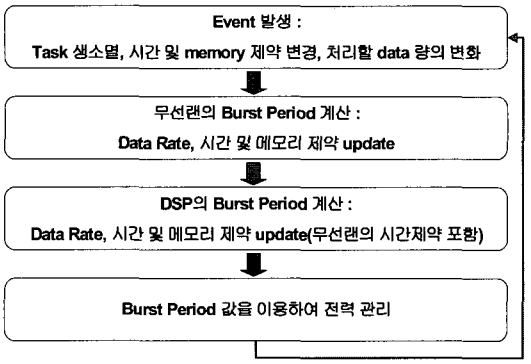


그림 11. EAJS의 outline
Fig. 11. Outline of EAJS.

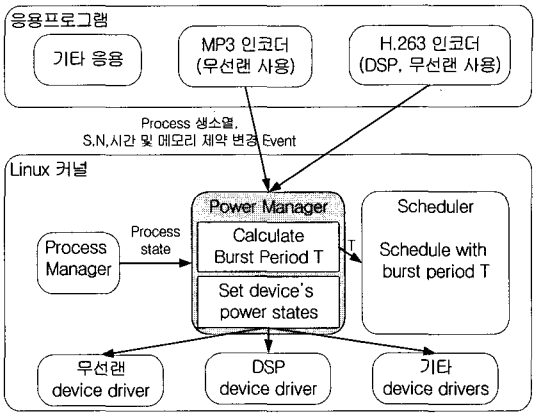


그림 12. AFM의 소프트웨어 모듈별 통신
Fig. 12. Communication between software modules of AFM.

```
==========================================
    /* Parameter Initialization : S,N,Bw,Tlatency */
/* Burst period 'T' 선택 */
If (add tasks to run queue) or (remove tasks from run queue) or
(time constraint change)
or (memory constraint change) or (task's S*N change) {
  switch(event) {
      case "add task Tn to run queue":
        S*N = S*N + (S*N of Tn);
        Tlatency = min(Tlatency, Tn's Tlatency);
      case "remove task Tn from run queue" :
        S*N = S*N  (S*N of Tn);
        Tlatency=min(all running tasks's Tlatency except Tn's);
      case "new time constraint"          :
        Tlatency = (new Tlatency);
      case "new buffer constraint"        :
        BufSize = (new BufSize);
      case "Task Tn's S*N change"         :
      S*N = S*N - (S*N of Tn) + (new S*N of Tn);
}
```

$$T_{BufLimit} = \frac{BufSize}{S \times N};$$

$$T_{(Eburst = Esplit)} = \frac{E_0 - T_0 \times p_{sleep}}{p_{idle} - p_{sleep}} \times \frac{1}{1 - \frac{N}{Bw} \times S};$$

$$T = min(T_{BufLimit}, T_{Latency});$$

$$IF((T_{E_{burst}=E_{split}} \geq T_{BufLimit}) or (T_{E_{burst}=E_{split}} > T_{Latency}))$$

```
        then, Split으로 스케줄링;
else T값을 가지고 burst로 스케줄링;}
==========================================
```

그림 13. Power manager의 pseudo code
Fig. 13. Pseudo code of power manager.

EAJS의 outline을 보여주고, 그림 12는 EAJS를 사용할 때 AFM의 소프트웨어 모듈들간의 통신을 보여준다. 마지막으로 그림 13은 EAJS의 outline을 토대로 만든 pseudo 코드이며, 그림 12에서 전력 관리기(Power

Manager)에 해당 하는 부분이다.

Ⅳ. 실 험

이번 장에서는 Ⅲ장에서 설명한 EAJS를 AFM에 구현하여 그림 14와 같은 측정 환경에서 실험한 결과에 대해 논의 한다. 그림에서 에너지 측정 board는 동시에 8개 하드웨어 장치들의 에너지 소모량을 측정 할 수 있는 기능을 갖추었다.

1. EAJS의 특성 실험

실험은 AFM에서 무선랜과 DSP를 사용하는 H.263 인코더가 수행 될 때, DSP와 무선랜이 소모하는 에너지를 10분간 측정하는 방식으로 이루어 졌다. 이 실험은 사용 가능한 buffer의 크기, time latency, workload와 EAJS의 관계를 밝힘으로써 상황에 따라 EAJS를 적절히 활용할 수 있게 하는 정보를 제시하는데 목적이 있다. 그림 15는 FPS가 변할 때 무선랜과 DSP의 에너지 소모를 보여준다. 이 그래프의 결과에 따르면, 무선랜과 DSP의 사용량이 많아질수록(즉, FPS가 증가할수록) EAJS에 의한 에너지 절약 비율이 낮아짐을 알 수 있는데, 이것은 사용 가능한 buffer의 크기와 deadline 제약 때문이다. 그림 15(a)에서 FPS가 15 이상일 때는 더 이상 에너지 절약 효과를 나타나지 않는데, 이것은 DSP의 처리능력(bandwidth)보다 많은 workload가 부과되었기 때문이다.

그림 16은 사용 가능한 Buffer의 크기를 변화 시키면서 소모되는 에너지를 측정한 결과이다. (a)에서 DSP의 경우 사용 가능한 buffer의 크기가 1Mbyte미만인 경우에는 split 스케줄이 더 유리하고, 1Mbyte 이상인 경우에는 burst 스케줄이 더 유리하다는 것을 알 수 있다.

한편 buffer 크기가 8Mbyte 이상일 경우는 시간의

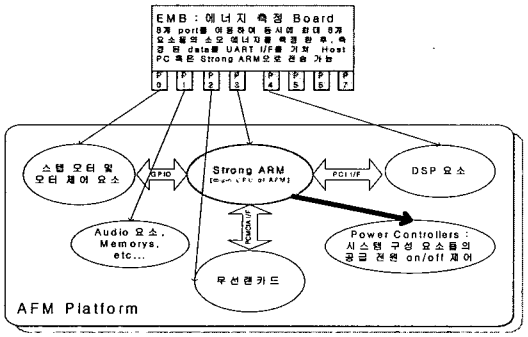
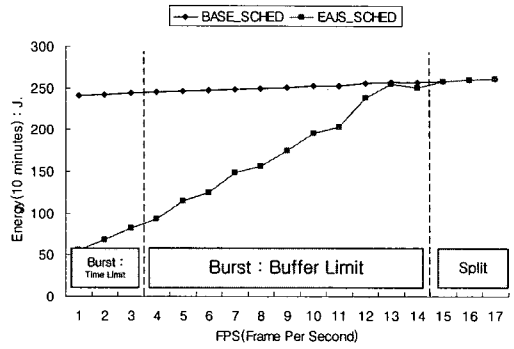
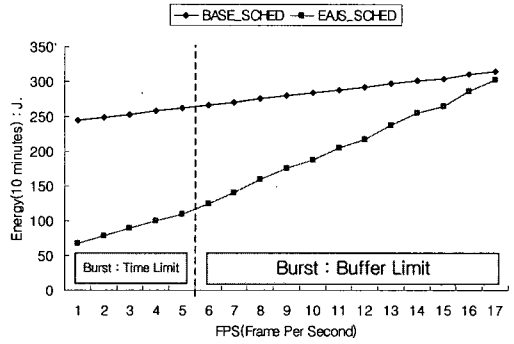


그림 14. 실험 환경
Fig. 14. Platform for experiment.

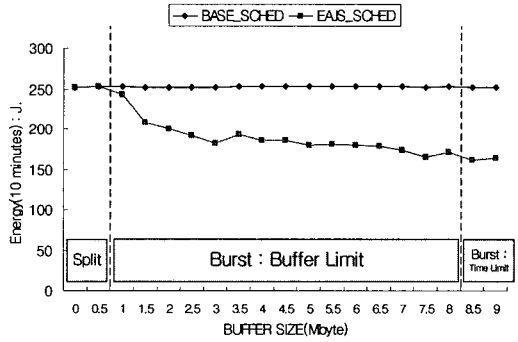


(a) DSP

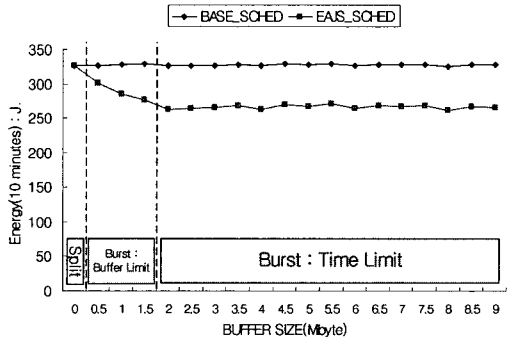


(b) 무선랜

그림 15. FPS에 따른 에너지 소모
Fig. 15. Energy consumption versus frame per second.

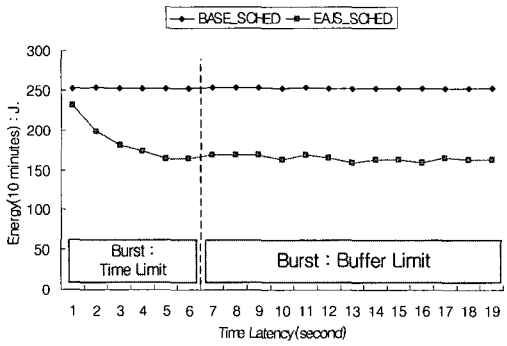


(a) DSP

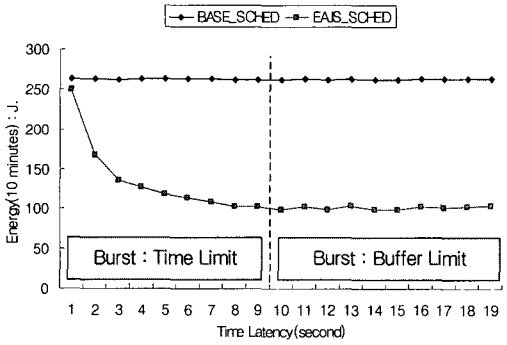


(b) 무선랜

그림 16. Buffer 크기에 따른 소모 에너지
Fig. 16. Energy consumption versus available buffer size.



(a) DSP



(b) 무선랜

그림 17. Time Latency에 따른 소모 에너지
Fig. 17. Energy consumption versus time latency.

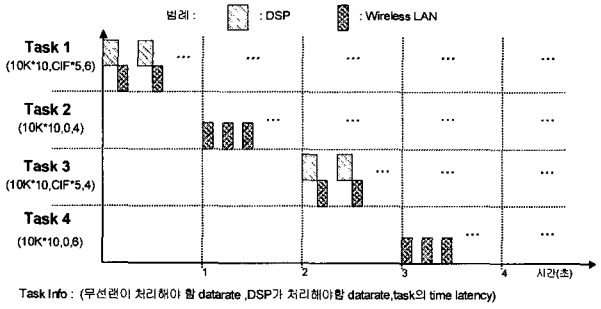


그림 18. EAJS 실험에 쓰인 Workload 정보
Fig. 18. information of workload used in experiment.

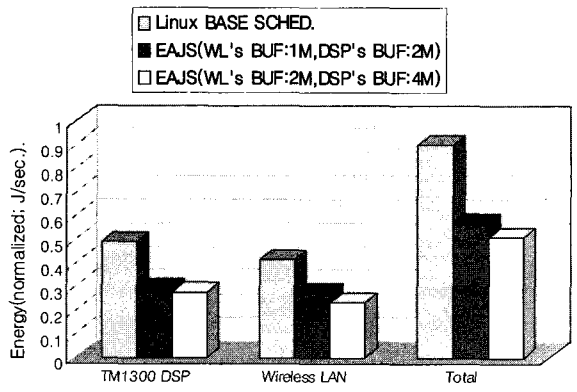


그림 19. 스케줄 방식별 에너지 소모량
Fig. 19. Energy consuption versus schedule policy.

제약을 받기 때문에, buffer 크기를 더 늘리더라도 에너지를 추가로 절약 할 수는 없다. 그러나 8Mbyte라는 수치는 process의 time latency와 FPS값에 의해 결정되기 때문에 응용프로그램에 따라 변하는 값임을 알 수 있다. 그림 17은 time latency가 변할 때 소모 되는 에너지를 나타낸 그래프이며, 사용 가능한 buffer의 크기와 EAJS와의 관계와 유사한 형태를 띠고 있다.

2. EAJS 성능 실험

이번 실험은 실제로 여러 개의 workload를 수행 시키면서, EAJS에 의한 에너지 절약 효과를 확인하기 위한 실험이다. 실험에 사용될 workload는 그림 18과 같이 주기적으로 무선랜과 DSP를 사용하는 4개의 task(혹은 process)이다. 그림에서 task info는 task가 무선랜과 DSP를 사용하는 것에 대한 정보를 나타낸다. Task 1을 예로 들면, (10K*10,CIF*5,6)이며 이것은 "무선랜을 통해 초당 10Kbyte의 data를 10번 전송하며, DSP를 통해 CIF 영상 5fps로 압축을 수행 하며, time latency는 6초"라는 것을 의미 한다. 이렇게 4개의 task는 1초 간격으로 생성 된 후 10분간 수행 되는데, 이때 무선랜과 DSP가 소모한 초당 평균 에너지를 측정한 결과가 그림 19의 막대그래프 이다.

그림 19의 막대는 리눅스의 기본 스케줄, 3Mbyte (무선랜 1Mbyte, DSP 2Mbyte) buffer를 가진 EAJS, 6Mbyte (무선랜 2Mbyte, DSP 4Mbyte) buffer를 가진 EAJS 등의 3가지의 스케줄 정책에 대한 결과를 에너지 소모를 보여준다. EAJS로 스케줄 되었을 경우, 기본 스케줄러 보다 44%의 에너지를 절약됨을 알 수 있다. EAJS의 특성 실험에서 확인 한 바와 같이 buffer의 크기가 클수록 에너지 절약이 많이 되며, 시간제약에 제한 될 경우에는 buffer의 크기를 무한정 늘리더라도 에너지가 더 이상 절약 되지는 않음을 알 수 있다.

V. 결 론

본 논문에서는 가정용 로봇에 적용 할 수 있는EAJS라는 저전력 job 스케줄링 기법을 제안하고, 새롭게 개발한 AFM 로봇에 적용하여 그 효과를 검증 하였다. EAJS는 기존의 저전력 스케줄러와 달리, job 스케줄에 사용되는 buffer가 소모하는 에너지와, 사용가능한 buffer의 크기, buffering으로 인해 발생하는 시간 지연 등을 동시에 고려하여 스케줄에 반영하기 때문에, 실질적으로 에너지의 소모를 줄이면서 동시에 시스템의 성

능저하를 최소화 하는 장점을 가지고 있다. AFM에 EAJS의 prototype을 구현하여 실험한 결과, H.263 인코더를 수행 시킬 때 무선랜과 DSP에서 최대 44%의 에너지 소모를 줄일 수 있음을 확인 하였다. 또한 사용 가능한 버퍼의 크기가 클수록, process가 허용 가능한 시간 지연이 길수록 EAJS는 더 많은 에너지를 절약 할 수 있음을 확인 하였다.

참 고 문 헌

[1] J. M. Rabaey and M. Pedram, Eds., Low Power Design Methodologies. Norwell, MA: Kluwer, 1996.

[2] L. Benini and G. De Micheli, Dynamic Power Management: Design Techniques and CAD Tools. Norwell, MA: Kluwer, 1997.

[3] StrongARM Processor. [Online]. Available : http://developer.intel.com/design/strong/

[4] "TriMedia TM-1300 Media Processor Data Book," 2000 Sep 30, Philips.

[5] L. Benini, A. Bogliolo, and G. Demicheli, "A survey of design techniques for system-level dynamic power management," IEEE Trans. VLSI Syst. Vol.8 June 2000.

[6] G. De Micheli and L. Benini, "System level power optimization: Techniques and tools," ACM Trans. Design Automation Electron. Syst., vol. 5, no. 2, pp. 115-192, Apr. 2000.

[7] A. R. Karlin, M. S. Manasse, L. A. McGeoch, and S. Owicki, "Competitive randomized algorithms for nonuniform problems," Algorithmica, vol. 11, no. 6, pp. 542-571, June 1994.

[8] E.-Y. Chung, L. Benini, and G. De Micheli, "Dynamic power management using adaptive learning tree," in Int. Conf. Comput.-Aided Design, San Jose, CA, Nov., 1999, pp. 274-279.

[9] C.-H. Hwang and A. C. H. Wu, "A predictive system shutdown method for energy saving of event driven computation," ACMTrans. Design Automation Electron. Syst., vol. 5, no. 2, pp. 226-241, 2000.

[10] J. R. Lorch and A. J. Smith, "Scheduling techniques for reducing processor energy use in MacOS," Wireless Networks, vol. 3, no. 5, pp.311-324, 1997.

[11] Y. Shin and K. Choi, "Power conscious fixed priority scheduling for hard real-time systems," in Proc. Design Automation Conf., New Orleans, LA, June 1999, pp. 134-139.

[12] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy," in Proc. Symp. Operating Syst. Design Implementation, Monterey, CA, Nov. 1994, pp. 13-23.

[13] Yung-Hsiang Lu, Luca Benini, and Giovanni De Micheli, "Power-Aware Operating Systems for Interactive Systems," IEEE Trans. VLSI Syst., vol 10, no.2, pp.119-134,April 2002.

저 자 소 개

최 승 민(정회원)

2002년 중앙대학교 전기전자제어공학과 학사 졸업

2004년 서울대학교 전기컴퓨터공학과 석사 졸업

2005년 현재 한국전자통신연구원(ETRI) 지능형로봇연구단 내장형하드웨어컴포넌트 연구팀 연구원

<주관심분야 : SoC 설계, 비디오 코덱, 네트워크 로봇>

채 수 익(정회원)

1976년 서울대학교 전기공학과 학사 졸업

1978년 서울대학교 전기공학과 석사 졸업

1987년 Stanford 대학교 전기공학과 박사 졸업

2005년 현재 서울대학교 전기컴퓨터공학부 교수 산자부 SoC설계특화연구기반구축사업단 단장

<주관심분야: 반도체, 저전력 VLSI 설계, 저전력 SoC 설계, 비디오 코덱>