논문 2009-46SD-7-4

멀티프로세서상의 에너지 소모를 고려한 동적 전압 스케일링 및 전력 셧다운을 이용한 태스크 스케줄링

(Energy-Aware Task Scheduling for Multiprocessors using Dynamic Voltage Scaling and Power Shutdown)

김 현 진*, 홍 혜 정*, 김 홍 식**, 강 성 호***

(HyunJin Kim, Hyejeong Hong, Hong-Sik Kim, and Sungho Kang)

요 약

멀티프로세서가 임베디드 시스템에서 널리 쓰임에 따라 지원되는 전력 최소화 기법을 이용하여태스크를 수행하기 위해 필요한 에너지의 소모량을 줄여야 할 필요성이 대두된다. 본 논문은 동적 전압 스케일링 및 전력 셧다운을 이용하여 에너지 소모를 최소화 하는 태스크 스케줄링 알고리즘을 멀티프로세서 환경을 위해 제안하였다. 제안된 알고리즘에서는 전력 셧다운시의 에너지 및 타이밍 오버헤드를 고려하여 반복적으로 태스크 할당 및 태스크 순서화를 수행한다. 제안된 반복적인 태스크 스케줄링을 통해 전체 에너지 소모를 줄이는 가장 좋은 해를 얻을 수 있었다. 전체 에너지 소모는 리니어 프로그래밍 모델 및 전력 셧다운의 임계 시간을 고려하여 계산되었다. 실제 어플리케이션으로부터 추출된 표준 태스크 그래프에 기반을 둔 실험 결과를 통해 하드웨어 자원 및 시간제한에 따른 에너지 소모 관계를 분석하였다. 실험 결과를 볼 때 제안된 알고리즘은 기존의 우선권 기반의 태스크 스케줄링에 대해서 의미 있는 성능 향상을 얻을 수 있었다.

Abstract

As multiprocessors have been widely adopted in embedded systems, task computation energy consumption should be minimized with several low power techniques supported by the multiprocessors. This paper proposes an energy-aware task scheduling algorithm that adopts both dynamic voltage scaling and power shutdown in multiprocessor environments. Considering the timing and energy overhead of power shutdown, the proposed algorithm performs an iterative task assignment and task ordering for multiprocessor systems. In this case, the iterative priority-based task scheduling is adopted to obtain the best solution with the minimized total energy consumption. Total energy consumption is calculated by considering a linear programming model and threshold time of power shutdown. By analyzing experimental results for standard task graphs based on real applications, the resource and timing limitations were analyzed to maximize energy savings. Considering the experimental results, the proposed energy-aware task scheduling provided meaningful performance enhancements over the existing priority-based task scheduling approaches.

Keywords: Task scheduling, Dynamic Voltage Scaling, Power Shutdown

I. 서 론

서브마이크론 이하의 반도체 공정의 발달로 다양한 멀티프로세서 시스템이 상용화 되고 있다. 멀티프로세서는 여러 개의 태스크를 동시에 수행하는 병렬화를 통해 어플리케이션의 수행 성능을 향상시킨다. 그러나 처리소자 (Processing Element: PE)의 개수가 증가하면서 멀티프로세서 시스템의 에너지 소모 또한 증가하게 되므로 에너지 소모량을 줄이는 것은 매우 중요하다.

시스템의 저전력화를 위해 시스템 단계에서부터 하위 단계까지 다양한 기법들이 연구되어 왔다. 시스템 단계에서 에너지를 최소화 하는 기법은 하위 단계에서

* 학생회원, ** 정회원, *** 평생회원,
연세대학교 전기전자공학과
(Department of Electrical and Electronic Engineering, Yonsei University)
접수일자: 2009년2월19일, 수정완료일: 2009년6월26일

수행되는 기법보다 훨씬 효율적이다[1]. 시스템 단계에서 수행되는 에너지 최소화 기법중 대표적으로 전력 셧다운 (Power Shutdown: PS[2])과 동적 전압 스케일링 (Dynamic Voltage Scaling: DVS[3])이 있다. 전력 셧다운 기법은 프로세서가 동작을 수행하지 않으면 필요 없는 스위칭 동작을 제어하는 것으로써 대용량의 주변 장치를 제어하는데 주로 사용된다. 동적 전압 스케일링은 전원 전압을 낮추고 클락 주파수를 낮춤으로써 에너지의 소모를 줄이는 것이다. 현재의 공정기술에서는 적절한 동적 전압 스케일링은 전력 셧다운 기법보다 에너지 소모를 줄이는데 효과적이다[3]. 그러나 복잡한 멀티프로세서의 에너지 소모를 줄이기 위해 단순히 동적 전압 스케일링만을 사용하는 것만을 고려해서는 안 된다. 왜냐하면 이 두 방법이 시스템 단계에서 동시에 지원 가능 할 수 있기 때문이다. 예를 들어 AMD의 Opteron 멀티프로세서의 경우 파워 스케줄러가 각각의 코어에 독립적으로 전력 셧다운을 적용할 수 있다. 또한 입력 전압은 외부 전압 레귤레이터에서부터 제어되는 기능을 갖추고 있다[4]. 그러나 전력 셧다운은 에너지 공급에 필요한 시간 및 에너지의 오버헤드를 가지고 있다는 점을 고려해야 한다. 동적 전압 스케일링을 적용 시에는 프로세서가 동작이 필요 없을 시에도 누설 전류에 의한 에너지 소모가 계속 일어난다.

멀티프로세서 상에서 소모되는 에너지를 줄이기 위해서는 어플리케이션의 태스크를 어느 프로세서에서 수행할 것인지와 (태스크 맵핑) 태스크의 순서를 어떻게 배열할 것인지가 (태스크 순서화) 포함된 태스크 스케줄링이 이루어 져야 한다. 이를 위해서 데드라인을 고려한 우선권 기반의 연구가 진행되었다[5~8]. 이 경우 각 태스크의 입출력에 따른 의존적인 관계 때문에 한 태스크에 동적 전압 스케일링이 적용되면 이후의 태스크의 시작 시간 및 전체 수행 시간이 모두 영향을 받게 된다. 이와 같은 문제를 해결하기 위해 멀티프로세서 상에서 어떻게 태스크를 스케줄링하고 동적 전압 스케일링을 적용할지에 대한 몇몇 연구가 진행되어 왔으나 멀티프로세서가 다중의 시스템 에너지 최소화 기법을 지원하면서 파워 셧다운 적용시의 오버헤드 및 동적 전압 스케일링을 동시에 고려하는 경우의 에너지 소모를 줄이기 위한 태스크 스케줄링의 연구가 요구된다.

본 연구에서는 멀티프로세서 상에서의 태스크 수행에 필요한 에너지 소모를 최소화시키기 위해 동적 전압 스케일링 및 전력 셧다운을 적용하는 태스크 스케줄링 기법을 제안하였다. 전력 셧다운시의 에너지 및 타이밍 오버헤드를 고려하여 우선권 기반의 반복적인 태스크 할당 및 태스크 순서화를 제안하였다. 실험 결과는 실제 어플리케이션에 기반을 둔 태스크 그래프에 대해 2.85%~6.24%의 에너지를 기존의 우선권 기반의 스케줄링 대비 줄일 수 있었다. 또한 실험 결과를 통해 하드웨어 자원 및 시간제한에 따른 에너지 소모 관계를 분석하였다.

Ⅱ. 배경 지식

어플리케이션 모델: 통신-태스크 그래프 (CTG)는 $G(V, E)$로 정의된다. 이때 각 꼭지점(vertex) v_i, $v_j \in V$은 태스크 T_i와 T_j를 각각 나타낸다. 각각의 모서리 (edge) $e(i, j) \in E$는 출발지 꼭지점인 v_i 와 목적지 꼭지점 v_j 간의 제어 및 데이터의 의존성을 의미한다. 다시 말하면 v_j에 해당되는 태스크가 수행되기 위해서는 v_i에 해당되는 태스크가 수행이 된 이후에 수행이 가능하며 또한 v_i로부터 출력 데이터가 v_j에 도착한 후에 v_j에 해당되는 태스크가 수행될 수 있다는 것을 의미한다. 이 때 각각의 인덱스 i와 j는 태스크 T_i와 T_j를 의미한다. 꼭지점 v_i에 해당되는 태스크 수행에는 NC_i의 사이클이 필요하다. 데드라인(deadline)은 태스크 수행의 시각 제약을 의미하는 것으로 d_i로 나타내 진다.

하드웨어 모델: 멀티프로세서 시스템은 N개의 처리소자로 이루어져 있으며 각각의 처리소자는 지연 없이 서로 상호연결을 가지고 있다고 가정할 것이다. 각 처리소자는 전원 전압을 가변할 수 있는 내장 프로세서 또는 독자적인 시스템일 수 있다. 각각의 처리소자 즉 PE들은 전압 스케일링에 따른 수행 동작 주파수를 채택한다. 본 논문에서는 기존의 논문들[5~8]과 같이 전압 스케일링에 따른 시간 및 에너지 오버헤드는 무시할 것이다. 또한 하나의 처리소자는 한 시간대에 하나의 태스크를 수행한다.

에너지 모델: 각 프로세서의 전력 소모 P는 식 (1)과 같이 스위칭에 의한 동적 전력 소모 P_{ac}와 누설 전류에 의한 정적 전력 소모 P_{dc}로 구성된다.

$$P = P_{ac} + P_{dc} \tag{1}$$

동적 전력 및 정적 전력 소모는 태스크가 프로세서에

서 수행되는 경우에 모두 일어나며 태스크가 프로세서에서 수행되지 않을 시에는 누설 전류에 의한 정적 전력 소모만이 일어나게 된다. 동적 전압 스케일링이 적용되는 경우 파워 스케줄러에 의해 미리 결정된 비연속적인 전압 및 동작 주파수에 따라 각 사이클에 소모되는 전력량 또한 미리 결정된다. 전압 및 주파수를 낮춤으로써 해당 사이클에 해당되는 총 동적 전력 소모가 줄어들게 된다.

전력 셧다운이 적용될 시에는 태스크가 수행되지 않는 시간의 정적 전력 소모를 제거한다. 그러나 파워 라인 및 기타 초기화를 위한 에너지 및 시간이 요구되어지므로 전력 셧다운을 위한 에너지 오버헤드 및 타이밍 오버헤드가 존재한다. 오버헤드를 고려하여 태스크가 수행되지 않는 시간에 전력 셧다운을 적용할지의 여부를 결정하기 위해 식 (2)를 통해 전력 셧다운 임계 시간($T_{threshold}$)을 구해야 한다.

$$T_{threshold} = \max(T_o, E_o / E_{low}) \quad (2)$$

T_o, E_o 및 E_{low}는 각각 전력 셧다운의 타이밍 오버헤드, 에너지 오버헤드, 가장 낮은 동작 주파수의 한 사이클의 정적 에너지 소모를 의미한다. 태스크가 수행되지 않는 시점의 연속적인 시간 간격이 전력 셧다운 임계 시간 보다 큰 경우만 이득을 볼 수 있다.

III. 제안된 에너지 소모를 고려한 태스크 스케줄링

태스크 스케줄링 및 동적 전압 스케일링의 결과에 따라 태스크를 수행하지 않는 시간에 전력 셧다운 임계 시간을 고려하여 전력 셧다운을 적용시킬지 결정한다. 전체적으로 줄어든 에너지 소모량은 동적 전압 스케일링에 의해서 줄어든 동적 에너지 소모량과 전력 셧다운을 통해 줄어든 정적 에너지 소모량의 합이므로 단순한 결정적인 (deterministic) 알고리즘으로 해결하는 것은 문제가 있다. 그러므로 반복적으로 태스크 스케줄링의 결과를 바꾸면서 최소의 에너지 소모를 가지는 태스크 스케줄링을 찾도록 한다. 제안된 태스크 스케줄링 알고리즘의 개요를 그림 1과 같이 도식화하였다.

주어진 통신-태스크 그래프 및 우선순위 튜닝 파라미터 (priority tuning parameter)를 기반으로 에너지 소모를 고려한 태스크 스케줄링 (energy-aware task scheduling)이 수행된다. 각 태스크 스케줄링에서는 우선순위 기반 (priority-based)의 태스크 선택 (task selection), 태스크 맵핑과 순서화 (task mapping & ordering)를 통해 각각의 태스크를 스케줄링 한다. 모든 태스크들의 스케줄링이 끝나면 리니어 프로그래밍 (linear programming: LP)을 통해 동적 전압 스케일링의 값을 구하고 이에 따라 임계 시간을 고려하여 전력 셧다운을 적용한다. 최종적으로 전체 에너지 소모를 계산하게 된다. 반복횟수 등의 종료 조건 (end condition)이 만족하지 않을 경우에는 우선순위 튜닝 파라미터의 값을 정해진 만큼 증가시키면서 반복적으로 루프를 수행한다. 이를 통해서 태스크 스케줄링의 결과를 바꾸고 이에 따른 에너지 소모를 반복적으로 계산하여 최종적으로는 최소의 에너지 소모를 가지는 태스크 스케줄링을 찾게 된다.

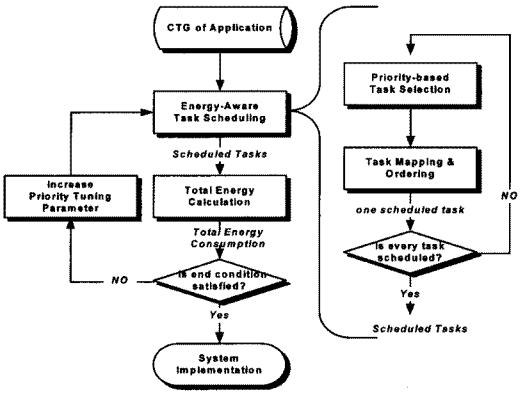

그림 1. 제안된 태스크 스케줄링의 개요
Fig. 1. Outline of Proposed Task Scheduling Algorithm.

가. 우선순위 기반의 태스크 선택

우선순위에 기반을 둔 태스크 스케줄링의 경우 불필요한 의존적인 패스의 길이를 작게 만드는 장점을 가지고 있다[5~7]. 이를 통해 태스크에 가능한 많은 여유시간을 부여함으로써 동적 전압 스케일링을 통해 태스크의 에너지 소모를 줄일 수 있는 확률을 높여준다. 우선순위 튜닝 파라미터의 값에 따른 우선순위 기반 태스크 선택의 과정을 요약하면 다음과 같다.

태스크 Ti의 제일 늦은 완료시간(latest finish time: lft_i)은 다음과 같이 정의된다.

$$lft_i = \min(d_i, lft_j - NC_j \mid \forall i, e(i,j) \in E. \quad (3)$$

각 태스크의 제일 늦은 완료시간은 통신-태스크 그래프의 최종 태스크의 데드라인에서 쉽게 계산될 수 있다. 태스크 T_i의 태스크 준비완료 시간(task ready

time: r_i)은 T_i의 모든 부모 태스크들이 수행이 완료된 시점을 의미한다. 또한 T_i가 PE_k에 맵핑되었을 때 가장 빠른 수행 가능 시간을 PE 수행 가능 시간(available PE time, ap_{ik})이라 하자. 이 때 태스크 T_i의 가장 이른 시작 시간 (earliest starting time: es_i)를 다음과 같이 정의할 것이다.

$$es_i = \max(r_i, \min(ap_{ik} | \forall PE_k \in \boldsymbol{PE}\, s.t. k \le N) \quad (4)$$

여기서 k, ap_{ik}는 각각 PE의 인덱스, k번 PE의 T_i를 수행하기 위한 가능한 가장 빠른 시간을 의미한다.

식 (3)과 (4)에 의거하여 태스크 T_i의 우선값 (PRI_i)은 다음과 같이 정의된다.

$$PRI_i = es_i \times K + lft_i \quad (5)$$

여기서 K는 우선순위 튜닝 파라미터이다. [5]의 우선순위 기반 방식에서는 K의 값이 1이다. 제안된 알고리즘에서는 K의 값을 변경하면서 각 태스크의 우선값을 변경하게 되고 이에 따라 각기 다른 태스크 스케줄링을 얻게 된다.

나. 태스트 맵핑 및 순서화

식 (5)를 기반으로 스케줄링 되지 않은 태스크들 각각의 우선값이 계산되고 가장 적은 우선값을 가지는 태스크가 선택된다. 태스크 T_i가 맵핑되어질 순서라 가정할 때 태스크 준비완료 시간 r_i가 PE의 수행 가능 시간보다 이를 때는 가장 이른 PE 수행 가능 시간을 가지는 프로세서에 맵핑한다. r_i가 PE의 수행 가능 시간보다 같거나 늦으면 가장 최근에 수행 가능한 PE에 태스크 T_i를 맵핑한다.

태스크 T_i가 맵핑될 프로세서를 결정한 후에 태스크 순서화를 통해 맵핑된 태스크의 시작 시간을 결정하게 된다. 맵핑된 태스크의 시작 시간은 태스크 순서화를 통해 결정할 것이다. 태스크의 준비완료 시간 r_i가 PE의 수행가능 시간보다 이후이거나 같으면 태스크는 r_i에서부터 수행을 시작할 것이다. 그렇지 않으면, PE의 수행가능 시간이 T_i의 시작시간이 된다. 태스크 순서화 이후에 맵핑 가능한 태스크의 리스트에서 스케줄링 된 태스크를 삭제할 것이다. 이후에 태스크 스케줄링이 적절하게 되었는지를 데드라인을 고려하여 체크한다.

그림 2에서는 태스크 맵핑과 순서화의 한 예를 보여주고 있다. 여기에서 그림 2. (a)는 통신-태스크 그래프

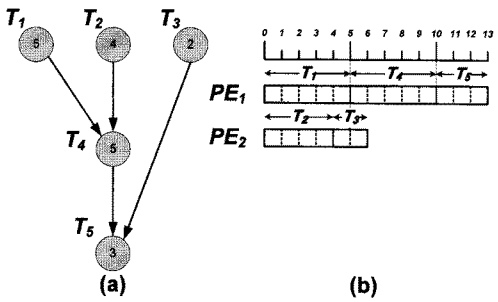

그림 2. 태스크 맵핑 및 순서화의 예
Fig. 2. Example of Task Mapping and Ordering.

이며 원안의 숫자는 동적 전압 스케일링이 적용되지 않은 태스크의 수행된 시간을 의미한다. 태스크 T_1, T_2를 맵핑한 후에 T_3를 PE_2에 맵핑하게 되는데 이는 T_3의 준비완료 시간이 PE_1의 수행 가능 시간보다 빠르기 때문이다. 또한 T_2와 T_3는 가능한 가깝게 순서가 배열된다. T_5는 PE_1에 맵핑되는데 PE_1의 수행 가능 시간이 PE_2의 수행 가능 시간보다 T_5의 준비 완료 시간에 가깝기 때문이다.

다. 전체 에너지 소모 계산

모든 태스크가 스케줄링 된 후에 리니어 프로그래밍을 통해서 동적 전압 스케일링을 통한 각 태스크의 전압 레벨이 결정된다. [5]에서 언급된 0-1 리니어 프로그래밍 식을 적용할 수 있으며 만약 전압 레벨은 high와 low의 두 가지가 있는 경우는 그림 3과 같이 표현된다.

그림 3에서 Ex는 전압 레벨에 따른 수행 시간을 의미한다. 심볼 D와 d는 태스크의 시작 시간 및 태스크의 데드라인을 의미한다. 전압 레벨이 다수인 경우 또한 정수 리니어 프로그래밍 식으로 해결할 수 있다[5]. 결정된 동적 전압 스케일링의 결과에 대해서 태스크를 수행하지 않는 시간에 대해서 전력 셧다운 임계 시간을 고려하여 전력 셧다운을 적용한다. 이를 통해서 정적 에너지 소모를 구할 수 있으며 최종적으로 총 에너지 소모의 합을 구하도록 한다.

$$Ex_i \ge Ex_i(v_h)$$
$$Ex_i \le Ex_i(v_l);$$
$$D_i - D_j \ge Ex_i(v_h) \text{ s.t. } edge_{ij} \in edge;$$
$$D_i + Ex_i \le dl_i;\}$$

그림 3. 리니어 프로그래밍의 제한식
Fig. 3. LP Formulation Constraints.

V. 실험 결과

제안된 태스크 스케줄링에 대한 평가를 위해 기존 및 제안된 알고리즘을 구현하였다. 또한 몇가지 파라미터를 변경하면서 이에 대한 결과 값을 비교 분석하였다.

가. 실험 환경

제안된 알고리즘을 구현하기 위해서 C++ 언어 및 부스트 라이브러리[9]를 사용하였다. 실험 대상은 Standard Task Graph(STG)[10]의 실제 어플리케이션 기반의 태스크 그래프를 이용하였다. 표 1은 각각의 통신-태스크 그래프의 특성을 나타내고 있다. sparse의 경우 높은 병렬화 수행 때문에 많은 양의 처리소자를 필요로 한다. robot의 경우는 각 태스크와 연결된 최대 입력의 개수가 3밖에 되지 않았다. fpppp의 경우는 최대 입력의 수가 무려 81이나 되고 태스크의 수가 300이상이다. 각 통신-태스크 그래프들은 tight, normal, 및 loose의 세 종류의 데드라인을 가진다. 각 데드라인은 통신-태스크 그래프의 임계 경로에 기반을 하여 결정되었다.

또한 리니어 프로그래밍 계산기로써 lp_solve[11]를 선택하고 어플리케이션 프로그램 인터페이스 (API) 함수를 사용하여 C++에 리니어 프로그래밍을 적용할 수 있었다. 우선순위 튜닝 파라미터의 초기에는 0으로 설정되어 있다. 이 값을 일정하게 증가시키면서 전체 에너지 소모의 값을 이웃 솔루션과의 비교한다. 본 실험의 종료 조건은 100번의 루프를 실행하는 것이다. 우선순위 튜닝 파라미터의 값은 0.1씩 증가되는 것으로 설정한다.

멀티프로세서에 채택된 처리소자의 에너지 소모에 관련된 데이터는 [6]에 언급된 ARM 11 프로세서의 데이터에 기반한다. 이경우 두 개의 다른 전압 모드를 제공한다. high 전압 모드 및 low 전압 모드에서의 태스크의 수행시간은 각각 1과 2의 단위 시간이 걸리며 40pJ/cycle 및 13.3pJ/cycle의 에너지가 소모된다. 정적 에너지의 경우 low 전압 모드의 20%의 에너지를 소모한다고 가정한다. 이 때 셧다운시의 에너지 및 타이밍 오버헤드의 경우 100pJ 및 10 단위 시간으로 가정시 이 때 임계 시간은 약 37.5 단위 시간으로 설정되므로 38 단위 시간 이상의 연속된 태스크가 수행되지 않을 시에 전력 셧다운이 적용된다.

표 1. STG에 기반한 통신-태스크 그래프의 특성
Table 1. Properties of CTGs from Standard Task Graphs.

이름	태스크수	edge수	데드라인		
			tight	normal	loose
fpppp	334	1196	1593	2124	2655
robot	88	130	854	1138	1423
sparse	96	128	183	244	305

나. 성능 평가

그림 4에서는 [7]과 [8]의 논문을 통해 선택된 처리소자의 개수 및 데드라인을 변경하면서 얻어지는 에너지 소모량을 보여주고 있다. 아무런 값이 나타나지 않은 경우는 처리소자의 개수가 너무 적거나 또는 모든 태스크가 데드라인 이전에 실행될 수 없는 경우를 의미한다. 적합한 해를 얻은 모든 경우에 있어서 데드라인이 동일한 경우에는 처리소자의 개수를 늘릴수록 전체 에

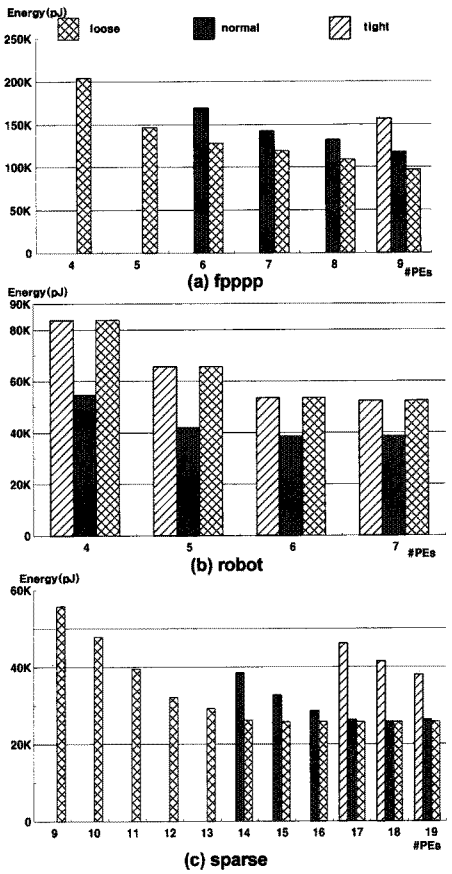

그림 4. 채택된 CTGs에 대해 처리소자의 개수 및 데드라인을 변경하면서 얻어진 에너지 소모량
Fig. 4. Total Energy Consumptions by Varying the Number of Adopted PEs and Deadlines.

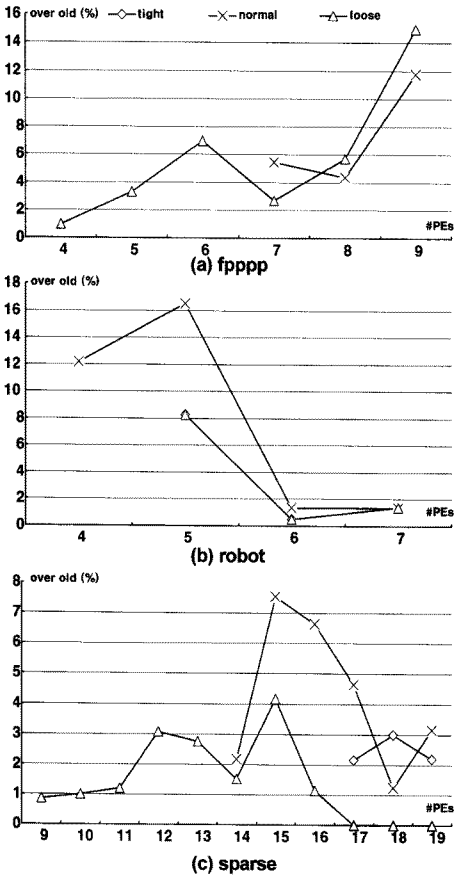

그림 5. 기존 우선권 기반 태스크 스케줄링 대비 성능 향상 결과

Fig. 5. Summary of Performance Enhancements over Previous Priority-based Task Scheduling

너지 소모가 줄어든다는 것을 알 수 있었다. 또한 처리 소자의 개수를 늘리면서도 일정 수준이 되면 에너지의 소모가 줄어들지 못하는 경우나 감소량이 줄어드는 경우가 생기게 된다. 이는 전체적인 에너지 소모가 전력 셧다운 및 동적 전압 스케일링을 통해서 충분하게 감소된 경우 처리소자의 개수를 증가시켜서 얻을 수 있는 에너지 소모량의 감소 비율이 줄어든다는 것을 의미하게 된다.

그림 5에서는 제안된 알고리즘과 [5]의 기존의 우선권 기반의 스케줄링 결과와 비교하였다. 제안된 알고리즘과 기존의 태스크 스케줄링이 모두 값을 가지는 경우만 그림 5에 표현된다. 용어 "over old"는 기존의 우선권 기반의 스케줄링 결과대비의 성능 향상을 의미한다. 성능 향상의 값은 처리소자의 개수와는 상관없이 불규칙적으로 성능 향상의 값이 나타났다. 또한 통신-태스크 그래프에 따라 일정한 값을 가지지도 않았다. 평균적으로 기존의 방식에 비해 평균적으로 6.24%, 5.18%, 그리고 2.85%의 에너지를 fpppp, robot과 sparse의 경우에 대해서 줄일 수 있었으며 적절한 처리소자가 선택된 경우에는 에너지 소모를 상당량 줄일 수 있었다.

VI. 결 론

본 연구에서는 멀티프로세서 상에서의 태스크 수행에 필요한 에너지 소모를 최소화시키는 동적 전압 스케일링 및 전력 셧다운을 적용하는 태스크 스케줄링 기법을 제안하였다. 전력 셧다운시의 에너지 및 타이밍 오버헤드를 고려하여 반복적인 태스크 할당 및 태스크 순서화가 이루어진다. 이 때 우선권 기반의 태스크 스케줄링을 반복 적용하여 에너지 최소화를 위한 최상의 해를 얻을 수 있었다. 실제 어플리케이션에 기반한 태스크 그래프를 대상으로 한 실험 결과에서 제안된 알고리즘은 멀티프로세서상의 에너지 소모를 효율적으로 줄일 수 있었다.

참 고 문 헌

[1] R. Jejurikar et al., "Leakage aware dynamic voltage scaling for real-time embedded systems," *in Proc. Design Automation Conf.*, pp. 5-280. 2004.

[2] L. Benini et al., "A survey of design techniques for system-level dynamic power management," *IEEE Trans. on VLSI systems*, vol. 8, pp. 299-316, 2000.

[3] I. Hong et al., "Power optimization of variable-voltage core-based systems," *IEEE Trans. on CAD*, vol. 18, pp. 1702-1714. 1999.

[4] J. Dorsey et al., "An integrated quadcore opteron processor," *in Proc. Int'l Solid State Circuits Conf.*, 2007, pp. 102-103.

[5] Y. Zhang et al., "Task scheduling and voltage selection for energy minimization," *in Proc. Design Automation Conf.*, pp. 183-188, 2000.

[6] G. Varatkar and R. Marculescu, "Communication -aware task scheduling and voltage selection for total systems energy minimization," *in Proc. Int'l Conf. on CAD*, pp. 510-517. 2003.

[7] Hyunjin Kim et al., "Total energy minimization of real-time tasks in an on-chip multiprocessor using dynamic voltage scaling efficiency metric," *IEEE Trans. CAD*, vol. 27, no. 11, pp. 2088-2092, 2008.

[8] P. de Langen and B. Juurlink, "Leakage-aware multiprocessor scheduling for low power," *in Proc. Int'l Parallel and Distributed Processing Symp.*, pp. 8-15. 2006.
[9] J.G. Siek, L.Q. Lee, and A. Lumsdaine, "The boost graph library user guide and reference manual," Addison-Wesley Professional, 2001.
[10] "Standard task graph," http://www.kasahara.elec.waseda.ac.jp/schedule
[11] "lp_solve reference guide," http://sourceforge.net/project/, ver. 5.5.0.10.

저 자 소 개

김 현 진(학생회원)
1997년 연세대학교 전기공학과 학사 졸업.
1999년 연세대학교 전기 및 컴퓨터 공학과 석사 졸업.
2005년 삼성전기 중앙연구소 선임연구원.
2009년 현재 연세대학교 전기전자공학과 박사과정.
<주관심분야: SoC 설계 및 응용, CAD>

김 홍 식(정회원)
1997년 연세대학교 전기공학과 학사 졸업.
1999년 연세대학교 전기 및 컴퓨터공학과 석사 졸업.
2004년 연세대학교 전기전자 공학과 박사 졸업.
2004년~2005년 Virginia 공대 박사 후 연구원.
2006년 삼성전자 시스템 LSI 사업부 책임연구원.
2009년 현재 연세대학교 TMS 사업단 연구교수.
<주관심분야 : SoC 설계, 테스트>

홍 혜 정(학생회원)
2006년 연세대학교 전기전자 공학과 학사 졸업.
200년 현재 연세대학교 전기전자 공학과 석박통합과정.
<주관심분야: SoC 설계 및 응용, 테스트>

강 성 호(평생회원)
1986년 서울대학교 제어계측 공학과 학사 졸업.
1988년 The University of Texas, Austin 전기 및 컴퓨터 공학과 석사 졸업.
1992년 The University of Texas, Austin 전기 및 컴퓨터 공학과 박사 졸업.
1992년 미국 Schlumberger Inc. 연구원.
1994년 Motorola Inc. 선임 연구원.
2009년 현재 연세대학교 전기전자공학과 교수.
<주관심분야 : SoC 설계, SoC 테스트>