효율적인 슬랙 분석 방법에 기반한 경성 실시간 시스템에서의 동적 전압 조절 방안

(Dynamic Voltage Scaling Algorithms for Hard Real-Time Systems Using Efficient Slack Time Analysis)

김 운 석 † 김 지 홍 †† 민 상 렬 †††

(Woonseok Kim) (Jihong Kim) (Sang Lyul Min)

요 약 동적 전압 조절(DVS: Dynamic Voltage Scaling)은 온라인 상태에서 CMOS 회로의 공급 전압과 클락 속도를 동적으로 조절하는 것으로 내장형 실시간 시스템의 에너지 소모량을 줄이는데 매우 효과적인 기법이다. 일반적으로 DVS 알고리즘의 에너지 효율은 이의 슬랙 측정 방법에 의해 크게 좌우된다. 본 논문에서는, 향상된 슬랙 측정 방법에 기반한 주기적 경성 실시간 태스크들을 위한 새로운 DVS 알고리즘을 제안한다. 기존의 방법들과는 달리, 제안된 온라인 슬랙 측정 방안은 우선순위 기반 스케줄링의 기본적인 특성을 이용하며, 따라서 대부분의 우선순위 기반 스케줄링 정책에 대해 적용이 가능하다. 본 논문에서는, 이른종료시한우선(EDF: Earliest Deadline First) 스케줄링 정책과 주기-단조(RM: Rate Monotonic) 스케줄링 정책으로 대변되는 동적 및 고정 우선순위 스케줄링 정책에 대해 제안된 슬랙 측정 방안을 적용하는 방안을 제시한다. 또한, 모의 실험을 통해, 제안된 알고리즘은 기존의 DVS 알고리즘에 비해 프로세서의 에너지 소모량을 효과적으로(20~40% 정도) 줄일 수 있음을 보인다.

키워드 : 저전력 시스템, 실시간 시스템, 동적 전압 조절

Abstract Dynamic voltage scaling(DVS), which adjusts the clock speed and supply voltage dynamically, is an effective technique in reducing the energy consumption of embedded real-time systems. The energy efficiency of a DVS algorithm largely depends on the performance of the slack estimation method used in it. In this paper, we propose novel DVS algorithms for periodic hard real-time tasks based on an improved slack estimation algorithm. Unlike the existing techniques, the proposed method can be applied to most priority-driven scheduling policies. Especially, we apply the proposed slack estimation method to EDF and RM scheduling policies. The experimental results show that the DVS algorithms using the proposed slack estimation method reduce the energy consumption by 20~40 % over the existing DVS algorithms.

Key words : Low power systems, Real-time systems, Dynamic voltage scaling

1. 서 론

최근 무선 인터넷 환경의 확산과 더불어 스마트폰 및 PDA(Personal Digital Assistant)와 같은 이동 내장형 시스템들의 이용과 응용이 확대되고 있다. 이들 시스템들은 대부분 제한된 전원 용량을 가지는 베터리를 기반으로 운용되기 때문에, 이들을 위한 저전력 설계 기술의 개발은 매우 중요하다.

여러 저전력 설계 기법들 중 동적 전압 조절 기법(DVS: Dynamic Voltage Scaling)[1]은 CMOS 회로로 구성되는 프로세서의 공급 전압을 온라인 상태에서 조절하여 에너지 소모량을 줄이는 기법이다. 일반적으로 CMOS 회로의 전력(P) 소모는 공급 전압 V_{DD}에 대해 $P \propto V_{DD}^2$인 관계를 가지므로, 공급 전압의 감소는 에너지 소모량 측면에 있어서 매우 효과적이다. 하지만, 공급 전압이 낮아질 때에는 회로의 클락 속도(f_{clk}) 또한 함께 늦추어져야 하기 때문에, 즉 $f_{clk} \propto V_{DD}$인 관계를 가지므로 프로세서의 처리량은 줄어들게 된다. 이러한 프로세서 성능과 에너지 소모량 간의 이해득실 관계를

† 비 회 원 : 서울대학교 전기컴퓨터공학부
wskim@archi.snu.ac.kr
†† 종신회원 : 서울대학교 전기컴퓨터공학부 교수
jihong@davinci.snu.ac.kr
종신회원 : 서울대학교 전기컴퓨터공학부 교수
symin@archi.snu.ac.kr
논문접수 : 2002년 8월 20일
심사완료 : 2003년 8월 22일

이용하여 주어진 작업에 대한 최적의 전압 및 클락 속도를 설정하는 것을 동적 전압 스케줄링이라 한다.

태스크들이 엄격한 종료 시한을 가지는 경성 실시간 시스템의 경우, 프로세서에 요구되는 성능은 태스크들의 시간 제약성에 의해 주어진다. 프로세서의 공급 전압이 낮아져 태스크들의 실행 시간이 길어질 경우, 태스크들의 실행이 이들의 종료 시한을 초과할 수 있다. 실시간 시스템에서 태스크들의 종료 시한 초과는 심각한 시스템 문제를 유발할 수 있기 때문에, 동적 전압 조절은 태스크들의 슬랙(혹은 유휴 구간)을 활용하는 한도 내에서 행해질 수 있다. 따라서, 실시간 DVS 알고리즘의 에너지 효율성은 그러한 슬랙을 얼마나 정확히 계산해내느냐에 의해 크게 좌우된다.

슬랙 분석은 기존의 실시간 서버 시스템에서 광범위하게 연구되어왔다[2,3]. 이러한 시스템들에서는 주기적 태스크들과 함께 스케줄 되는 비주기적 태스크들의 응답 시간을 향상시키거나 이들에 대한 수용률을 높이기 위해 슬랙 분석이 필요하였다. 이러한 기존의 슬랙 분석 기법들은 일반적으로 높은 시간적/공간적 간접비용이 요구되기 때문에, 메모리 및 프로세서의 성능에 제약을 가지는 이동 내장형 시스템들에 대해서는 이러한 기법들이 이용되기 어렵다. 이러한 이유로, 대부분의 이동 내장형 시스템들에 대한 온라인 DVS 알고리즘들은 슬랙 계산에 있어서 간단한 경험적 방법(heuristic)을 이용해 왔다.

실시간 태스크들에 대한 DVS 알고리즘들은 크게 오프라인 알고리즘과 온라인 알고리즘으로 구분될 수 있다. 오프라인 알고리즘의 경우, 태스크들이 최악실행시간(WCET: Worst Case Execution Time)을 요구하는 최악 실행 시나리오에 대해 각 태스크에 대한 최적의 공급 전압과 클락 속도를 계산한다. 하지만, 일반적으로 태스크들의 실제 실행 시간은 오프라인에서 분석된 이들의 최악 실행 시간보다 적으므로, 온라인 상태에서는 태스크들의 실행 시간 변화에 따른 작업량(workload) 변화 슬랙(VST: Variation Slack Time)이 발행하게 된다[4]. 온라인 DVS 알고리즘들은 이러한 VST들을 온라인 상태에서 측정하여 스케줄 되는 태스크의 실행 전압을 낮추고자 한다.

기존의 실시간 시스템에서의 슬랙 분석에 관한 연구에서 알 수 있듯이, 태스크들이 가지는 슬랙은 태스크들이 어떤 스케줄링 정책에 의해 스케줄 되느냐에 따라 달라진다. 따라서, 기존의 온라인 DVS 알고리즘들과 오프라인 DVS 알고리즘들은 이들이 대상으로 하는 스케줄링 정책에 따라 달리 구현되어 왔다. 그러한 이유로 기존의 어떤 특정 스케줄링 정책에 대한 DVS 알고리즘은 다른 스케줄링 정책들에 대해 쉽게 적용되기 어렵다. 본 논문에서는 우선순위를 기반으로 하는 스케줄링 정책들에 대해 일반적으로 적용될 수 있는 태스크들의 슬랙 측정 방안을 제시한다.

본 논문에서는 주기적 실시간 태스크들이 우선순위를 기반으로 스케줄 되는 이동 내장형 시스템에서 효과적인 전압 조절을 위한 온라인 슬랙 측정 방안을 제시한다. 제안된 방안은 우선순위 기반 스케줄링 정책의 기본적인 특성과 태스크들의 주기성을 이용하여 태스크들이 이용할 수 있는 슬랙을 낮은 간접비용으로 측정한다. 특히, 본 논문에서는 EDF와 RM 스케줄링 정책으로 대변되는 동적 우선순위 스케줄링 정책과 고정 우선순위 스케줄링 정책에 대해 슬랙 측정 방안을 제시하고, 이를 이용한 DVS 알고리즘을 제시한다. 모의 실험 결과, 제안된 슬랙 측정 방안을 이용하는 DVS 알고리즘의 경우, EDF와 RM을 대상으로 하는 기존의 알고리즘들에 비해 각각 5% 와 40% 의 에너지 소모량 감소 효과를 보였다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 DVS 알고리즘들에 대해 살펴보고 이들의 문제점을 지적한다. 3장에서는 제안하는 우선순위 기반 슬랙 측정 방안의 기본 개념을 설명하고, 4장에서는 EDF와 RM에 대한 구현 방안을 설명한다. 5장에서는 실험을 통해 얻어진 제안된 방안의 에너지 효율성을 보이고, 6장에서 향후 연구 과제와 함께 결론을 맺도록 한다.

2. 관련 연구

DVS 알고리즘들은 [5]에서 Weiser 등에 의해 처음 제시된 이후, 실시간 시스템을 포함한 다양한 시스템 환경을 대상으로 확대 개발되었다. 우선, DVS 알고리즘들은 태스크들에 대한 전압 및 클락 속도가 언제 결정되느냐에 따라 크게 오프라인 알고리즘과 온라인 알고리즘으로 구분된다. 오프라인 알고리즘의 경우, 태스크들의 실행 시간을 최악 실행 시간으로 가정하고, 이에 대한 최적의 전압 및 클록 속도를 계산한다. 이와 같이, 태스크들의 도착 시간, 종료 시한, 그리고 실행 시간이 주어진 경우에는 최적의 해를 구하는 것이 가능한데, EDF 스케줄링에 대한 최적 계산 방법은 [6]에 제시된 바 있으며, 고정 우선순위 스케줄링 정책에 대한 최적 계산 방법은 [7]에 제시된 바 있다.

실제 태스크들의 실행 시간은 일반적으로 이들의 최악 실행 시간보다 적으며, 실행이 종료되기 이전에는 그 값을 알기 어렵다. 따라서, 오프라인에서 얻어진 최적의 해만으로는 최적의 결과를 얻을 수 없으며, 온라인 상태에서 이를 계산하는 방법이 필요하다. 온라인 DVS 알고리즘들은 대상으로 하는 태스크의 특성에 따라 주기적 태스크 집합을 대상으로 하는 것들[8-13]과 비주기

적 태스크 집합을 대상으로 하는 것들[14,15]로 나뉜다. 이러한 알고리즘들은 다시 대상으로 하는 시스템의 시간 제약성이 얼마나 엄격하느냐에 따라 경성 실시간 시스템을 위한 것들과 연성 실시간 시스템을 위한 것들로 구분된다. 본 논문에서는 주기적 태스크 집합을 대상으로 하는 경성 실시간 시스템에 대한 DVS 알고리즘들에 초점을 맞추도록 하겠다.

주기적 태스크 집합을 대상으로 하는 경성 실시간 시스템을 위한 DVS 알고리즘들은 [8-13]에서 제안되었는데, 이러한 기존의 알고리즘들의 문제점은 어떤 특정 스케줄링 정책에 대해 특화되어 있어, 다른 스케줄링 정책에서는 이용이 어렵다는 것이다. 예를 들어, [10]에 제안된 ccEDF와 laEDF 알고리즘과 같은 동적 우선순위 스케줄링 정책에 대한 DVS 알고리즘들의 경우에는 RM 스케줄링 정책과 같은 고정 우선순위 스케줄링 정책에 적용되기 어렵다. 반면, [9]에서 제안된 알고리즘의 경우에는, EDF와 RM 모두에 대해 그대로 이용이 가능하지만, 슬랙 측정 방법의 정확성이 떨어져 에너지 소모량 감소 측면에서 큰 효과를 기대하기 어렵다.

따라서, 본 논문에서는 우선순위 기반 스케줄링 정책들에 대해서 보편적으로 이용될 수 있는 슬랙 측정 방안을 제시하고, 대표적인 우선순위 기반 실시간 스케줄링 정책인 EDF와 RM에 대해 제시된 방안이 어떻게 활용될 수 있는지를 보이도록 하겠다.

3. 기본 개념

3.1 시스템 모델

본 논문에서는 주기적 태스크들이 EDF와 RM으로 스케줄 되는 경성 실시간 시스템을 가정한다. 대상이 되는 가변 전압 프로세서는 이의 공급 전압과 클록 속도를 이의 운용 범위 $[V_{\min}, V_{\max}]$와 $[f_{\min}, f_{\max}]$ 내에서 임의의 실수 값으로 결정될 수 있다고 가정한다. 태스크 집합 T는 n 개의 주기적 태스크들로 구성되며 $T = \{\tau_1, \tau_2, \cdots, \tau_n\}$와 같이 표현한다. 여기서 태스크들은 상호 독립적인 것으로 가정한다. 각 태스크 τ_i는 이의 주기 p_i와 최악 실행 시간 w_i를 가지며, 이의 상대 종료 시한 d_i는 이의 주기의 길이와 같은 것으로 가정한다. 각 태스크는 이의 인스턴스를 주기적으로 활성화시키는데, τ_i의 j번째 인스턴스는 $\tau_{i,j}$와 같이 표시한다. 문맥의 흐름상 의미가 명확한 경우에는 한 태스크 인스턴스를 τ_α와 같이 간단히 표기토록 한다. 각 태스크 인스턴스 τ_α는 이의 활성화 시점 a_α와 절대 종료 시한 d_α를 가진다.

3.2 우선순위 기반 슬랙 측정 방안

EDF와 RM과 같은 우선순위 기반 스케줄링 정책하에서 태스크들은 각각의 우선순위에 따라 스케줄 된다. 즉, 각 스케줄링 시점에서 활성화 된 태스크 인스턴스들 중 가장 높은 우선순위를 가진 태스크 인스턴스가 스케줄 되며, 이보다 낮은 태스크 인스턴스들은 상위 태스크의 실행이 종료되기 이전에는 스케줄될 수 없다. 또한, 한 태스크 인스턴스의 실행중 보다 높은 우선순위를 가지는 상위 태스크 인스턴스가 활성화 되는 경우에는 설령 이전에 스케줄된 태스크 인스턴스의 실행이 아직 종료되지 않았더라도, 이의 실행은 중단되고 새로이 활성화된 태스크 인스턴스가 스케줄된다. 태스크 인스턴스들의 우선순위는 각 스케줄링 정책 마다 달리 설정되는데, 예를 들어, EDF 스케줄링 정책에서는 절대 종료 시한이 가장 이른 태스크가 가장 높은 우선순위를 가지게 되며, RM 스케줄링 정책에서는 주기의 길이가 가장 짧은 태스크가 가장 높은 우선순위를 가지게 된다. 전자의 경우는 동일 태스크로부터 활성화된 인스턴스들인 경우에도 매 활성화 시점마다 서로 다른 우선순위를 가질 수 있으므로, 동적 우선순위 스케줄링이라 하며, 후자의 경우는 동일 태스크로부터 활성화된 인스턴스들은 모두 동일한 우선순위를 가지게 되므로, 고정 우선순위 스케줄링이라 한다.

그림 1(a)에서와 같이 세 개의 태스크 인스턴스들이 최악 실행 시간을 가지고 스케줄 되는 작업 구간을 고려해 보자. 태스크 인스턴스 τ_α가 가장 높은 우선순위를 가지고, τ_γ가 가장 낮은 우선순위를 가진다고 하자. 이 경우, τ_β의 활성화 시점과 종료 시한 사이의 구간 $[a_\beta, d_\beta]$는 다음 세가지의 작업 구간으로 구분될 수 있다. (1) 상위 태스크 인스턴스 τ_α의 실행을 위한 구간 $[a_\beta, s^e_\beta]$, (2) τ_β 의 실행을 위한 구간 $[s^e_\beta, d^e_\beta]$, 그리고 (3) 하위 태스크 인스턴스 τ_γ의 실행을 위한 구간

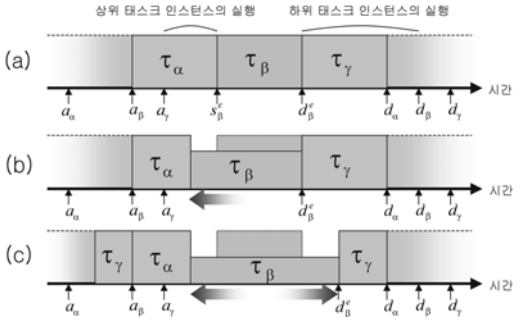

그림 1 태스크의 슬랙 분류; (a) 슬랙이 없는 경우, (b) 상위 태스크로 인스턴스로부터 슬랙이 발생한 경우, 그리고 (c) 하위 태스크 인스턴스로부터 슬랙이 발생한 경우

[d^e_β, d_β]이다. 여기서 s^e_β와 d^e_β는 각각 태스크들이 최악 실행 시간을 요구하는 최악 실행 시나리오에서의 τ_β의 예상실행 시작 시점과 예상종료 시점을 의미한다.

τ_α가 s^e_β보다 일찍 종료되는 경우, 즉 τ_α가 최악 실행 시간보다 적은 실행 시간으로 종료되는 경우 그림 1(b)에서와 같이 τ_β는 이의 실행 시작 시간을 앞당길 수 있으며 τ_α에 의해 이용되지 않은 시간을 활용할 수 있다. 또한, τ_β가 이의 유효 종료 시한[1]을 늦출 수 있는 경우도 가능하다. 예를 들어, τ_γ의 활성화 시점이 a_β보다 이르다면, τ_β의 실행은 a_β 이전에 (부분적으로) 완료될 수 있다. 이러한 경우, τ_β 는 그림 1(c)에서와 같이 이의 유효 종료 시한을 늦출 수 있다.

정리를 하면, 온라인 상태에서 어떤 한 태스크에게 유용한 VST는 다음 두가지의 경우로 구분될 수 있다. 1) 상위 태스크로 부터의 슬랙, 그리고 2) 하위 태스크로부터의 슬랙이다. 본 논문에서는 이 두가지 슬랙을 효과적으로 구하는 슬랙 측정 방안을 제시하고자 한다.

우선, 위와 같이 상·하위 태스크들로부터 발생하는 슬랙을 제대로 측정하고 활용하는 경우, 프로세서의 에너지 효율성이 좋아질 수 있음을 다음 예를 통해 확인토록 하자. 표 1에 있는 주기적 태스크 집합 T가 EDF 스케줄링 정책하에서 스케줄 되는 경우를 예로 들어보자. T의 경우, 태스크들이 모두 최악 실행 시간을 요구할 때, 프로세서 활용률은 1이 되고, 스케줄 구간 내에 슬랙이 전혀 발생하지 않는다. 하지만, 태스크들이 이보다 적은 실행 시간을 요구하는 경우에는 온라인 상태에서 슬랙이 발생하게 되고, 태스크들은 이러한 슬랙을 이용하여 낮은 전압 및 클록 속도로 스케줄될 수 있다.

태스크들이 모두 이들의 평균 실행 시간으로 실행된다고 가정하자. $\tau_{1,1}$이 $t=0.5$에서 종료하였을 때, $\tau_{2,1}$은 이의 실행을 $t=0.5$에서 시작한다. 최악 실행 시나리오에서도 알 수 있듯이, $\tau_{2,1}$이 $t=1.0$에서 실행이 시작되어도 이후 스케줄 되는 태스크들은 모두 스케줄 가능하다. 따라서, $\tau_{2,1}$이 $\tau_{1,1}$이 남긴 시간 0.5를 이용

표 1 실시간 태스크 집합의 예

	주기 (p_i)	최악실행시간 WCET (w_i)	평균실행시간 ACET (a_i)
τ_1	2	1	0.5
τ_2	3	1	0.5
τ_3	6	1	0.5

1) 여기서의 유효 종료 시한(effective deadline)은 이후 스케줄될 태스크들의 스케줄 가능성을 보장하는 한도 내에서 현재 스케줄 될 태스크가 완료되어야만 하는 가장 늦은 시간을 의미한다.

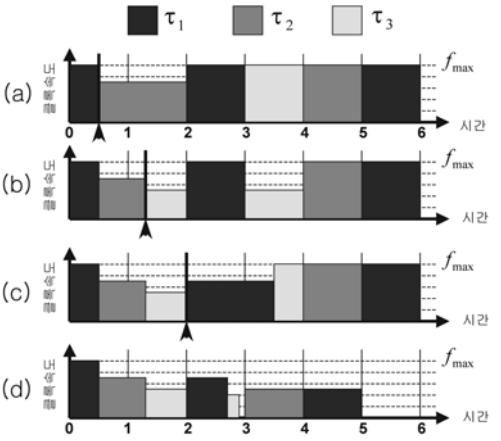

그림 2 전압 및 클록 조절의 예

하여 이의 실행 시간을 늘인다고 하여도, 모든 태스크들은 스케줄 가능하다. 따라서, 그림 2(a)에서와 같이 $\tau_{2,1}$은 낮은 클록 속도($\frac{1}{(1+0.5)} f_{\max}$)로 이의 실행 시간을 늘일수 있다. $\tau_{2,1}$이 $t=1.25$에서 종료하였을 때, $\tau_{3,1}$ 또한 그림 2(b)에서와 같이 낮은 클록 속도 ($\frac{1}{1+0.75} f_{\max}$)로 스케줄 될 수 있다.

$\tau_{1,2}$가 $t=2.0$에서 활성화 되었을 때, $\tau_{3,1}$은 여전히 실행 중이지만, $\tau_{1,2}$의 우선순위가 $\tau_{3,1}$ 보다 높으므로, $\tau_{1,2}$가 $\tau_{3,1}$의 실행을 선점할 것이다. 이 시점에서 $\tau_{3,1}$이 이미 부분적으로 실행되었기 때문에, $\tau_{3,1}$의 잔여 실행 시간은 이의 최악 실행 시간 보다는 적다. 따라서, $\tau_{1,2}$는 이의 유효 종료 시한을 늦출 수 있게 된다. 이 경우, $\tau_{3,1}$의 잔여 실행 시간은 0.57 이므로, $\tau_{1,2}$는 그림 2(c)에서와 같이 낮은 클록 속도($\frac{1}{1+0.43} f_{\max}$)로 이의 실행을 $t=3,43$까지 늘일 수 있다.

나머지 태스크 인스턴스들($\tau_{3,1}$, $\tau_{2,2}$, $\tau_{1,3}$)도 비슷한 방법으로 낮은 클록 속도와 전압으로 스케줄될 수 있다. 최종적인 스케줄은 그림 2(d)와 같다. 프로세서의 전력 소모가 클록 속도의 제곱에 비례한다고 가정했을 때, 그림 2(d)의 스케줄은 동적 전압 조절을 행하지 않고 전력 차단 기법만을 이용하는 경우에 비해 34.5% 적은 에너지를 소모한다.

다음 장에서는 위와 같은 슬랙 측정을 EDF와 RM 스케줄링 정책 하에서 어떻게 낮은 간접비용으로 구현할 수 있는지를 설명한다.

4. 우선순위 기반 슬랙 측정 방안

[9,13]에서 설명되었듯이, 태스크들이 가질 수 있는 슬

랙은 크게 두 부류로 구분할 수 있다. 오프라인 상태에서 계산될 수 있는 정적으로 주어지는 것(static slack time)과 온라인 상태에서 얻어지는 동적인 것(VST)으로 구분된다. 주어진 태스크 집합내 태스크들이 항상 최악 실행 시간으로 요구하는 경우를 최악 실행 시나리오라 할 때, 최악 실행 시나리오 상에서 프로세서 활용률이 1보다 작은 경우에는 태스크 스케줄상에 항상 유휴 구간(idle time)이 존재하게 된다. 이러한 슬랙은 오프라인 상태에서 분석이 가능하므로 정적 슬랙이라 한다. 또한, 태스크들의 실제 실행 시간이 이들의 최악 실행 시간보다 적은 경우, 온라인 상태에서 정적 슬랙 이외의 슬랙이 발생하게 되는데, 이러한 슬랙을 동적 슬랙 혹은 작업 부하량 변화에 따른 슬랙이라 한다. 동적 슬랙은 정적 슬랙과는 달리 오프라인 상태에서 계산이 불가능하다.

본 장에서는 주어진 태스크 집합이 EDF와 RM 스케줄링 정책에 의해 스케줄 되는 경우, 정적 및 동적 슬랙을 분석하는 방법을 제시하고, 이를 이용하는 동적 전압 조절 기법을 제시토록 한다.

4.1 정적 슬랙 분석

4.1.1 EDF 스케줄링 정책에서의 정적 슬랙 분석

주기적 태스크 집합이 EDF로 스케줄 되는 경우, 식 (1)에서와 같이 이의 최악 프로세서 활용률이 1보다 작거나 같으면 주어진 태스크 집합은 스케줄 가능하다.

$$U\ (=\sum_{i=1}^{n}\frac{w_i}{p_i})\ \le\ 1 \qquad (1)$$

만일, 주어진 태스크 집합이 $f_{\max}$ 상에서 위의 식을 만족하고 스케줄 가능하다면, 동적으로 전압이 조절되는 가변 전압 프로세서 상에서는 $f_{\max}' = U \cdot f_{\max}$ 인 클록 속도 상에서도 주어진 태스크 집합은 여전히 스케줄 가능하다. 즉, 각 태스크의 실행 시간이 $w_i' = \frac{w_i}{U}$ 가 되더라도 주어진 태스크 집합의 최악 프로세서 활용률은 1

표 2 실시간 태스크 집합의 예

	주기 (p_i)	최악실행시간 WCET (w_i)	평균실행시간 ACET (a_i)
τ_1	3	0.8	0.4
τ_2	4	0.8	0.4
τ_3	6	1.6	0.8

과 같으므로 여전히 스케줄 가능하다. 따라서, EDF 로 스케줄 되는 주기적 태스크 집합의 경우에는, 최악 프로세서 활용률 계산식을 이용하여 최고 클록 속도를 $f_{\max}'$로 설정하여 각 태스크에게 정적인 슬랙을 고르게 할당할 수 있다. 이때, 각 태스크는 $w_i(\frac{1-U}{U})$ 만큼의 슬랙을 할당 받게 된다.

4.1.2 RM 스케줄링 정책에서의 정적 슬랙 분석

기존의 실시간 시스템 연구에서도 알 수 있듯이, EDF 스케줄링에서와는 달리, RM 스케줄링 정책에서는 정확한 정적 슬랙 계산이 다소 복잡하다. 본 논문에서는 기존의 최적 슬랙 분석 기법을 이용하여 오프라인 상태에서 계산 가능한 슬랙들을 분석토록 한다[3,16].

개념적으로, 슬랙 구간들은 다음과 같이 정의된다. 태스크들이 최악 실행 시간을 요구하는 경우에도 태스크들의 스케줄 가능성을 보장하면서 프로세서가 유휴할 수 있는 가장 이른 구간들로 정의될 수 있다. 표 2와 같이 주어진 태스크 집합을 예로 들어보자. 이 태스크 집합의 최악 실행 시나리오는 그림 3과 같다. 그림에서 보여진 듯이, 태스크들이 모두 최악 실행 시간을 요구하는 경우, 프로세서는 [11, 12] 구간에서 유휴 시간을 가지게 된다. 이러한 시간들은 태스크들을 낮은 클록 속도로 실행시킬 때 활용될 수 있는데, [2]에서 제안된 최적 슬랙 분석 방법을 이용할 경우, 그림 4에서 보여지는 것과 같이 프로세서가 [6, 7] 구간에서 유휴하더라도 전체 태스

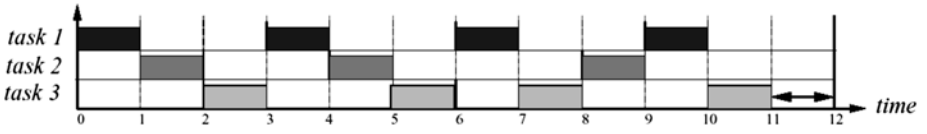

그림 3 최악 실행 시나리오의 예

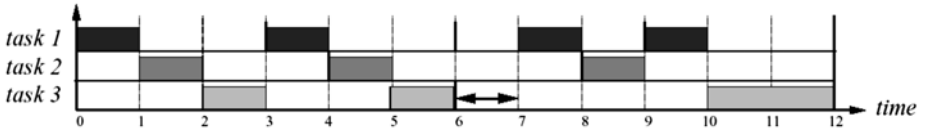

그림 4 최적 슬랙 분석 방법에 의해 결정된 가장 이른 슬랙 구간

크들의 스케줄 가능성은 여전히 보장된다. 즉, 본 예에서는 [6, 7] 구간의 1만큼의 슬랙이 최악 실행 시나리오 상에서 프로세서가 유휴할 수 있는 가장 이른 시간 구간이 된다.

기존의 최적 슬랙 분석 방법을 이용할 경우, 위와 같이 주어진 태스크들에 의해 이용될 수 있는 정적 슬랙을 오프라인 상태에서 정확히 분석할 수 있다. 주어진 태스크 집합에 대한 정적 슬랙 정보는 슬랙 구간 집합 T_S로 표현될수 있으며, 각 슬랙 구간 s_i는 (b_i, m_i)와 같이 표현될 수 있다. 여기서 b_i와 m_i는 각각 슬랙 구간의 시작점과 구간의 길이를 의미한다. 본 논문에서는 RM 스케줄링에 대해 위와 같이 기존의 최적 슬랙 분석 방법을 이용하여 정적 슬랙 정보 T_S를 구하고, T_S 내 각 슬랙 구간들은 b_i를 기준으로 정렬하여 온라인 상태에서 이를 이용한다.

4.2 동적 슬랙 분석

본 절에서는 온라인 상태에서 동적 슬랙을 효과적으로 계산하는 방법을 설명한다. 우선순위 기반 스케줄링에서 한 태스크가 가질 수 있는 동적 슬랙은 이의 원천이 어떠한 태스크로부터 비롯되었느냐에 따라 상위 태스크로부터의 슬랙과 하위 태스크로부터의 슬랙으로 구분될 수 있다. 이 두 타입의 슬랙을 측정하기 위해 다음 표기를 이용한다.

U_i^{rem} : τ_i에 의해 이용되지 않은 시간

W_i^{rem} : τ_i의 잔여 최악 실행 시간

본 논문에서는 실시간 스케줄러가 두 개의 태스크 큐, 즉 대기큐(wait-queue)와 준비큐(ready-queue)를 가지고 있는 것으로 가정한다. 대기큐는 실행이 완료된 태스크들이 속하는 큐로서 여기에 속한 태스크들은 다음 주기에 다시 활성화된다. 준비큐에는 활성화는 되었지만, 아직 실행이 완료되지 못한 태스크들이 속하게 된다. 모든 태스크들은 초기에 대기큐에 속하게 되며, 이때, 대기큐내의 태스크들은 각각의 다음 활성화 시점을 기준으로 정렬된다. 한 태스크가 활성화 되면, 그 태스크는 대기큐에서 준비큐로 옮겨지고, 준비큐는 태스크들의 우선순위에 따라 정렬된다. 예를 들어, EDF 스케줄링과 RM 스케줄링의 경우, 활성화된 태스크들은 각각 종료 시한과 주기의 길이를 기준으로 정렬된다. 각 태스크 활성화 시점에서 해당 태스크의 잔여 최악 실행 시간은 이의 최악실행시간으로 초기화되고, 미사용 시간은 EDF 스케줄링과 RM 스케줄링에 대해 다음과 같이 각각 초기화 된다.

EDF 스케줄링 : $U_i^{rem} = \frac{w_i}{U}$

RM 스케줄링 : $U_i^{rem} = w_i$

준비큐에 있는 태스크들 중 가장 높은 우선순위를 가진 활성태스크 τ_α가 실행을 위해 스케줄 되는데, EDF 스케줄링의 경우 가장 이른 종료 시한을 가진 태스크가 τ_α에 해당되며, RM 스케줄링의 경우에는 가장 짧은 주기의 길이를 가진 태스크가 τ_α로 선정되어 스케줄 된다.

스케줄된 태스크 τ_α는 스케줄 된 상태에서 실행이 완료되거나 상위 태스크에 의해 선점될 수 있다. τ_α가 선점되었을 때에는 τ_α는 다시 준비큐에 소속되고 다음 재실행을 기다리게 된다. τ_α가 상위 태스크의 도착 이전에 종료할 경우, 이의 잔여 실행 시간은 0으로 재설정된다. 하지만, 이의 미사용 시간 U_α^{rem} 은 재설정되지 않으며, 다른 태스크들을 위한 가용시간 측정에 이용된다. τ_α가 실행될 때, 이의 W_α^{rem}은 감소하게 되고, 이의 가용 시간 또한 소모되는데, 가용시간은 다음과 같이 계산된다.

현재의 스케줄링 시점 t_{cur}에서 τ_α의 가용시간은 다음 세가지 유형의 시간들로 구성된다.

$S_H(t_\alpha, t_{cur})$: t_{cur} 이전에 종료된 상위 태스크들의 미사용 시간들의 합

U_α^{rem} : τ_α가 가진 본래의 잔여 가용 시간

$S_L(\tau_\alpha, t_{cur})$: 하위 태스크들로부터의 슬랙 합

$T_H(\tau_\alpha, t_{cur})$를 t_{cur} 이전에 이미 종료한 태스크 인스턴스들의 집합이라고 할 때, $S_H(t_\alpha, t_{cur})$는 다음과 같이 계산된다.

$$S_H(t_\alpha, t_{cur}) = \sum_{\tau_i \in T_H(t_\alpha, t_{cur})} U_i^{rem} \quad (2)$$

여기서, U_α^{rem}와 W_α^{rem}는 서로 다를 수 있다. 예를 들어, 만일 τ_α가 이전에 선점된 후에 재실행 된 것이라면, τ_α에 대한 일부 작업이 선점되기 이전에 상위 태스크들의 미사용 시간을 이용하여 실행되었을 수 있다. 이러한 경우엔, U_α^{rem}가 W_α^{rem} 보다 클 수 있다.

$S_H(t_\alpha, t_{cur})$와 U_α^{rem}이 쉽게 계산될 수 있는 것에 비해 $S_L(\tau_\alpha, t_{cur})$를 정확히 계산하는 것은 상당한 오버헤드를 유발할 수 있다. 예를 들어, [16]에서의 최적 슬랙 계산 알고리즘을 이용하여 이를 측정하려 한다면, 한 초주기 내 각 태스크 인스턴스의 종료 시한 이전에 얼만큼의 작업이 반드시 종료되어야 하는지에 대한 정보를 항상 유지하여야 한다. 이러한 정보를 유지한다면, 각 스케줄링 시점에서 유용한 슬랙 값들을 정확히 계산할 수 있다. N을 초주기 내 태스크 인스턴스들의 수라 할 때, 이 최적 계산 방법은 $O(N)$의 계산 및 공간 복잡도를 요구한다.

본 논문에서는, $S_L(\tau_\alpha, t_{cur})$를 다음과 같이 근사치로

구한다. U_α^{rem}와 $S_H(t_\alpha, t_{cur})$는 τ_α의 실행을 위해 이용될 수 있는 시간 자원이므로, τ_α는 이 시간 자원을 활용하여 낮은 클록 속도로 실행 시간을 늘일 수 있다. 즉, τ_α는

$$\frac{W_\alpha^{rem}}{U_\alpha^{rem}+S_H(\tau_\alpha, t_{cur})} f_{\max} \quad (3)$$

의 클록 속도로 스케줄이 가능하다. τ_α가 이 클록 속도로 스케줄 되는 경우를 가정해 보자. 만일 τ_α가 t_{cur}에서 이의 최악 실행 시간을 요구한다면, τ_α는 $t_a = t_{cur} + S_H(\tau_\alpha, t_{cur}) + U_\alpha^{rem}$에 종료할 것이다. 상위 태스크가 t_a 이전에 활성화 되는 경우에는 τ_α는 선점될 것이고, t_a 이전에 종료할 수 없다. τ_α가 언제 재실행 될 것이고 그때 τ_α에게 유용한 시간은 얼만큼 되는지를 계산하는 것은 계산 복잡도가 높으므로, 그러한 상위 태스크 인스턴스가 존재할 때에는 τ_α의 가용 시간을 $S_H(\tau_\alpha, t_{cur}) + U_\alpha^{rem}$로 제한한다. 만일 그러한 태스크 인스턴스가 존재하지 않는다면, 하위 태스크로부터의 슬랙이 τ_α에게 유용한지를 확인한다. t_a 시점에서 만일 활성화되어 있는 태스크가 없다면, τ_α는 대기큐 내 태스크들의 활성화 시점중 가장 이른 시점(t_a')까지 이의 실행 시간을 늘일 수 있다. 더 나아가, t_a (혹은 t_a') 시점에서, 준비큐 내 태스크 인스턴스들이 슬랙을 가진다면, τ_α는 그 슬랙들을 이용하여 이의 실행을 더 늘일 수 있다. τ_β를 t_a 이전에 활성화 되었지만 완료되지 못한 태스크들 중 가장 높은 우선순위를 가지는 태스크 인스턴스라고 하자. 만일 $T_L(\tau_\alpha, t_{cur}) \neq \emptyset$ 이라면 (여기서 $T_L(\tau_\alpha, t_{cur})$는 τ_α 보다 낮은 우선순위를 가지는 태스크 인스턴스들 중 t_{cur} 이전에 종료된 태스크 인스턴스들의 집합), τ_β는 슬랙을 가질 수 있으며, 이는 다음과 같이 계산될 수 있다.

$$T_L'(\tau_\beta, t_a) = \{\tau_i \mid \tau_i \in T_L(\tau_\alpha, t_{cur}) \text{ and } d_i \le d_\beta\} \quad (4)$$

$$S_L(\tau_\alpha, t_{cur}) = (U_\beta^{rem} - W_\beta^{rem}) + \sum_{\tau_i \in T_L'(\tau_\beta, t_a)} U_i^{rem} \quad (5)$$

상기 식에서 $S_L(\tau_\alpha, t_{cur})$의 계산은 그림 5(a)에서와 같이 τ_β 보다 높은 우선순위의 태스크 인스턴스가 $[t_a, t_a + S_L(\tau_\alpha, t_{cur})]$(혹은 $[t_a', t_a' + S_L(\tau_\alpha, t_{cur})]$) 사이에 활성화 되지 않을때만 유효하다. 즉, $S_L(\tau_\alpha, t_{cur}) = 0$이 된다. 이러한 경우, 만일 그림 5(b)에서와 같이 그러한 태스크 인스턴스 τ_γ가 존재한다면 τ_α의 실행 시간 확장은 τ_γ의 활성화 시점까지로 제한된다. 즉, $S_L(\tau_\alpha, t_{cur}) = a_\gamma - t_a$가 된다.

EDF 스케줄링 정책하에서는 정적 슬랙들이 오프라인 상태에서 각 태스크 인스턴스에게 분배될 수 있기 때문에, 위와 같이 세 유형의 슬랙들을 계산하는 것으로 슬랙 계산은 완료된다. 하지만, RM 스케줄링의 경우에는 오프라인에서 분석된 정적 슬랙이 오프라인 상태에서 태스크 인스턴스들에게 배분 될 수 없으므로, 아래와 같이 온라인 상태에서 이를 활용하여야 한다.

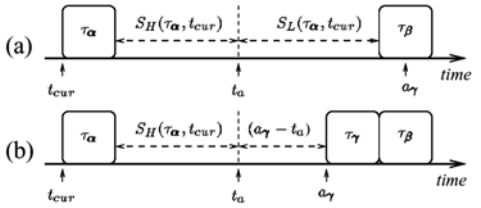

그림 5 하위 태스크로부터의 슬랙을 얻는 경우의 예

$S_S(t_{cur})$를 현재의 시점에서 유용한 정적 슬랙들의 값이라 할 때, 슬랙 테이블 T_S 내 각 슬랙 구간 정보 $s_i = (b_i, m_i)$를 참조하여 다음과 같이 t_{cur}에서 유용한 정적 슬랙의 값을 계산한다.

$$S_s(t_{cur}) = m_i \quad (6)$$

(여기서, $s_i \in T_S$, $b_i \le t_{cur}$,이고 $m_i > 0$이다.)

결과적으로, τ_α의 가용 시간 $S(\tau_\alpha, t_{cur})$은 EDF와 RM 스케줄링에 대해 각각 $S_H(\tau_\alpha, t_{cur}) + U_\alpha^{rem} + S_L(\tau_\alpha, t_{cur})$과 $S_S(t_{cur}) + S_H(\tau_\alpha, t_{cur}) + U_\alpha^{rem} + S_L(\tau_\alpha, t_{cur})$이 되고, 클록 속도는 아래와 같이 조절된다.

$$f_{clk}(\tau_\alpha, t_{cur}) = \frac{W_\alpha^{rem}}{S(\tau_\alpha, t_{cur})} f_{\max} \quad (7)$$

이때, 공급 전압은 이에 상응하는 값으로 결정되어 조절된다.

제안하는 알고리즘의 전체 과정은 아래와 같이 알고리즘 1과 알고리즘 2로 요약된다. 알고리즘에서 슬랙을 계산하기 위해 이용되는 부속 서브루틴들은 모두 $O(n)$의 계산 복잡도로 계산이 가능하다(여기서 n은 주기적 태스크 수를 말한다.).

태스크 인스턴스들의 미사용 시간을 재활용함에 있어서 일관성을 유지하기 위해서는 각 스케줄링 시점에서마다 슬랙 테이블과 태스크 인스턴스들의 미사용 시간들을 갱신해 주어야 한다. 이때, 이전에 스케줄된 태스크에 의해 소모된 시간을 반영할 때에는, 우선 슬랙 테이블의 값이 갱신되고, 이후 태스크들의 미사용 시간은 태스크들의 우선순위 순서대로 갱신된다.

5. 모의 실험

제안된 전압 조절 알고리즘의 에너지 효율성을 평가하기 위해, 기존의 DVS 알고리즘들과 더불어 모의 실

표 3 클록 속도 및 공급 전압 조절 알고리즘

알고리즘 2. 현재 스케줄된 태스크에 대한 클록 및 전압 설정

1. 입력
   τ_α = 현재 스케줄된 태스크 인스턴스
   τ_p = 바로전에 스케줄된 태스크 인스턴스
   t_{prv} = 이전 스케줄링 시점
   t_{cur} = 현재 스케줄링 시점
   f_{clk} = 현 클록 속도
2. 출력 : 전압 및 클록 속도 설정
3. τ_p 가 소모한 시간 $I=|t_{cur}-t_{prv}|$을 시간 자원 소모에 반영한다.
   (1) RM 스케줄링의 경우, 우선 슬랙 테이블을 갱신한다.
   (2) I 값이 모두 반영될 때까지, 높은 우선순위의 태스크들서부터 이들의 U_i^{rem} 값을 갱신한다.
4. 만일, τ_p가 t_{cur}에 종료되었다면,
   $W_p=0$으로 하고, τ_p를 준비큐에서 대기큐로 옮긴다.
5. 만일, τ_p가 τ_α에 의해 선점되었다면,
   W_p를 τ_p의 실행 시간 만큼 감소시킨다. ($W_p = W_p - I\frac{f_{clk}}{f_{\max}}$)
6. 만일, 활성화된 태스크 인스턴스가 없다면,
   프로세서를 다음 태스크 활성화 시점까지 전원 차단 모드로 전환한다
7. 아니면, 알고리즘 1을 이용하여 가용 슬랙 $S(\tau_\alpha)$를 측정하고 클록 속도를
   $f_{clk}=(W_\alpha/S(\tau_\alpha))f_{\max}$로 설정하고, 공급 전압은 이에 상응하는 값으로 설정한다.

표 4 슬랙 측정 알고리즘

알고리즘 1. 현재 스케줄된 태스크에 대한 슬랙 측정

1. 입력 : 활성화 태스크 τ_α, 대기큐(wait queue), 준비큐(ready queue), 현재시간 t_{cur}
   (* RM 스케줄링의 경우, 슬랙 테이블 T_S도 입력에 포함)
2. 출력 : τ_α에 대한 가용 시간 $S(\tau_\alpha)$
3. $T_H(\tau_\alpha, t_{cur})$ = 이미 종료된 상위 태스크 인스턴스들의 집합
4. $T_L(\tau_\alpha, t_{cur})$ = 이미 종료된 하위 태스크 인스턴스들의 집합
5. (* RM의 경우 : $S_S(t_{cur})$ = T_S를 참조하여 얻어진 현재의 시점에서 가용한 정적 슬랙의 양
6. $S_H = \sum_{\tau_i \in T_H(\tau_\alpha, t_{cur})} U_i^{rem} + U_\alpha^{rem}$
7. $S_L = 0$
8. t_α^h = τ_α 보다 우선순위가 높은 태스크 인스턴스들의 다음 활성화 시점중 가장 이른 시간
9. 만일, $t_{cur} + S_H < t_\alpha^h$이면
   {
10. t_a = τ_α가 $\frac{W_\alpha^{rem}}{S_H} f_{\max}$로 스케줄되고 최악실행시간을 요구하였을 때 종료예정시간
11. 만일, t_a까지 활성화되는 태스크 인스턴스가 없다면, (즉, 준비큐가 비어있으면),
12. $t_a' = \min(a_i \mid \tau_i \in$ 대기큐$)$로 하고,
13. $t_a = \max(t_a, t_a')$로 한다.
14. τ_β = t_a시점에서 스케줄될 태스크 인스턴스
15. $T_L'(\tau_\beta, t_{cur})$ = τ_β 보다는 우선순위가 높고 τ_α 보다는 우선순위가 낮은 이미 종료된 태스크 인스턴스들의 집합
16. $S_L = (U_\beta^{rem} - W_\beta^{rem}) + \sum_{\tau_i \in T_L'(\tau_\beta, t_{cur})} U_i^{rem}$
17. a_χ = τ_β 보다 높은 우선 순위를 가지는 태스크 인스턴스들중 다음 활성화 시점이 가장 이른 태스크 τ_χ의 다음 활성화 시점
18. $S_L = \min(S_L, a_\chi - t_a)$
    }
19. $S(\tau_\alpha) = \min(S_H + U_\alpha^{rem} + S_L \ , \ d_\alpha - t_{cur})$

험을 실시하였다. 실험은 SimDVS (통합 DVS 실험 환경)을 이용하여 실시하였다[17]. SimDVS의 에너지 시뮬레이터는 ARM8 마이크로 프로세서를 기반으로 한다. 클록 속도는 [8, 100] MHz 범위 내에서 1 MHz 단위로 조절이 가능하며, 공급 전압은 [1.1, 3.3] V 범위내에서 클록 속도에 상응하는 값으로 조절이 가능하다. 시스템이 유휴한 상태에서는 시스템이 전원 차단 모드로 들어가 전력을 전혀 소모치 않는 것으로 가정하였다. 실험에서, 전압 조절 및 문맥 교환에 따른 에너지 및 지연시간 오버헤드는 없는 것으로 가정하였다.

5.1 EDF 스케줄링에 대한 결과

제안된 EDF 스케줄링을 위한 DVS 알고리즘을 실제 실시간 응용물들과 인위적으로 합성된 응용물들을 이용하여 실험하였다. 실제의 실시간 응용들의 특성은 표 5에 요약되어 있다[18,19,20].

각 실험에서, 각 태스크 인스턴스의 실행 시간은 [최단실행시간,최악실행시간] 내에서 정규 분포를 따르는 것으로 가정하였다. 실제 실시간 응용물들에 대한 실험 결과는 그림 6(a)~(c)에 보여진다. 이 실험에서는 태스크들의 최단실행시간(BCET: Best Case Execution Time)을 최악실행시간(WCET)의 10%에서 100%로 변화 시키면서, 이에 따른 에너지 효율성 변화를 관찰하였다. 본 실험에서 비교 대상이 되는 DVS 알고리즘들은 신영수 등에 의해 제안된 lppsEDF[9]와 최적 슬랙 측정 방안[2]을 이용하는 lpOPT로 선정하였다. 각 그림에서 가로축은 최악실행시간에 대한 최단실행시간의 비율을 의미하며, 세로축은 전원차단기법 만이 이용된 경우에 대한 상대적인 에너지 소모량을 나타낸다.

표 5 실험에 이용된 실제 실시간 응용물들의 속성

응용물	태스크 수	최악실행시간 (ms)	주기 (ms)	최악 프로세서 활용율
CNC	8	0.035~0.72	2.5~9.6	0.489
Avionics	17	1~9	25~1,000	0.848
Videophone	4	1.4~50.4	40~66.7	0.986

태스크의 평균 실행시간은 최단실행시간이 작아질수록 적어지므로, 태스크들의 슬랙들은 최단실행시간이 줄어들수록 늘어나게 된다. 따라서, 그림에서 보여지듯이, 각 DVS 알고리즘의 에너지 효율성은 최악실행시간에

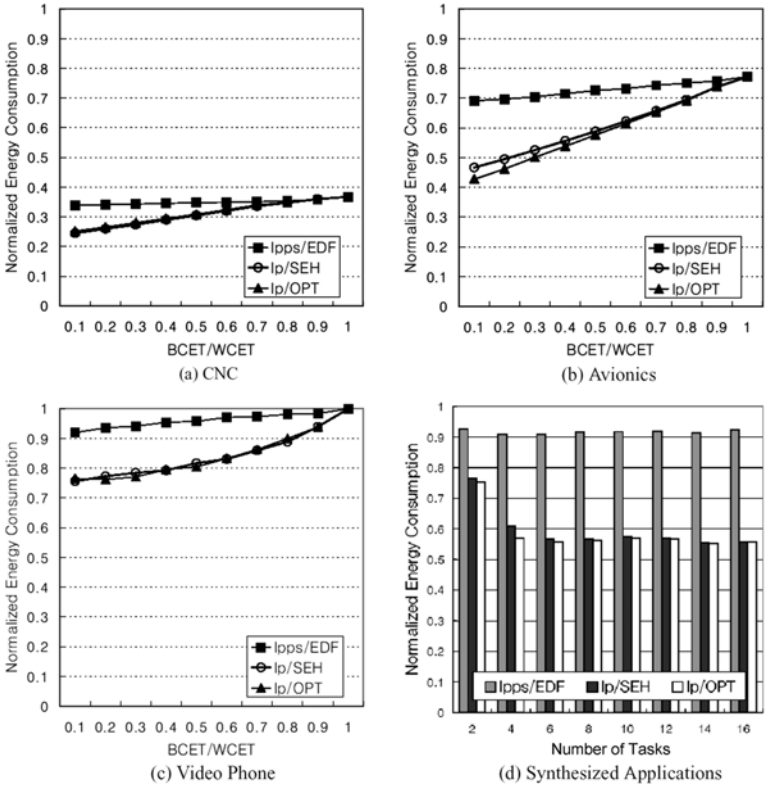

그림 6 EDF 스케줄링 정책에 대한 DVS 알고리즘들의 비교 실험 결과

대한 최단실행시간의 비율이 작아질수록 높아진다. 하지만, 제안된 lpSEH 알고리즘의 슬랙 측정이 더 효율적이기 때문에, lppsEDF 보다 더 빨리 에너지 효율성이 높아짐을 알 수 있다. 본 실험에서는 lpSEH가 lppsEDF에 비해 20~40% 정도 에너지 소모량을 감소 시킬 수 있음을 보였다. 본 실험에서는 또한, 비록 lpOPT가 높은 계산 복잡도로 보다 정확하게 슬랙을 측정하지만, 제안된 lpSEH 알고리즘이 lpOPT 알고리즘에 대해 필적할 만한 성능을 보였다.

그림 6(d)는 인위적으로 합성된 실시간 응용물들을 태스크 집합내 태스크 수를 변화시키면서 실험한 결과이다. 각 태스크 수에 대해 최악 프로세서 활용률이 1인 태스크 집합 100개를 생성하였다. 태스크들의 주기와 최악실행시간은 [10, 100]ms와 [1, 주기]ms의 범위 내에서 균등분포를 따르도록 하였다. 실험결과, 태스크의 수가 증가할 때, lppsEDF의 에너지 효율성은 변함이 없는 반면, lpSEH와 lpOPT의 에너지 효율성은 향상되었다. 이는 태스크의 수가 증가할수록 lpSEH와 lpOPT의 경우, 슬랙을 얻을 수 있는 태스크 인스턴스들의 수가 증가하는 반면, lppsEDF의 슬랙 측정은 다음 태스크의 도착 시간에 의해 결정되며 이는 태스크의 수와 무관하기 때문이다.

그림 6에서의 결과에서는 또한, lpOPT의 경우 각 스케줄링 시점에서 모든 가용한 슬랙을 정확히 계산하지만, 그 에너지 효율성은 lpSEH에 비해 좋지는 못하다. lpOPT의 경우, 현재 스케줄된 태스크는 이의 가용한 슬랙을 모두 이용토록 클록 속도를 설정하는데 이러한 슬랙 이용 정책이 공급 전압 및 클록 속도의 불균형을 가져올 수 있다. 예를 들어, 현재 가용한 슬랙을 다음 스케줄될 태스크를 위해 일부 남겨두는 것이 보다 좋은 결과를 가져올 수도 있다. 그림 6의 결과는 이와 같이, 보다 효과적인 슬랙 배분 정책이 슬랙 측정 방법 만큼 중요함을 보여주는 것이다.

5.2 RM 스케줄링 정책에 대한 결과

RM 스케줄링 정책을 위해 구현된 DVS 알고리즘 lpSHR을 이용하여 제안된 알고리즘의 효율성 또한 실험을 통해 확인하였다. 그림 7(a)와 (b)는 그림 6에서 이용된 실제 실시간 응용물들을 이용하여 실시된 실험결과이며, 그림 7(c)는 인위적으로 합성된 실시간 태스크들에 대한 실험 결과이다. 비교 대상이 되는 DVS 알

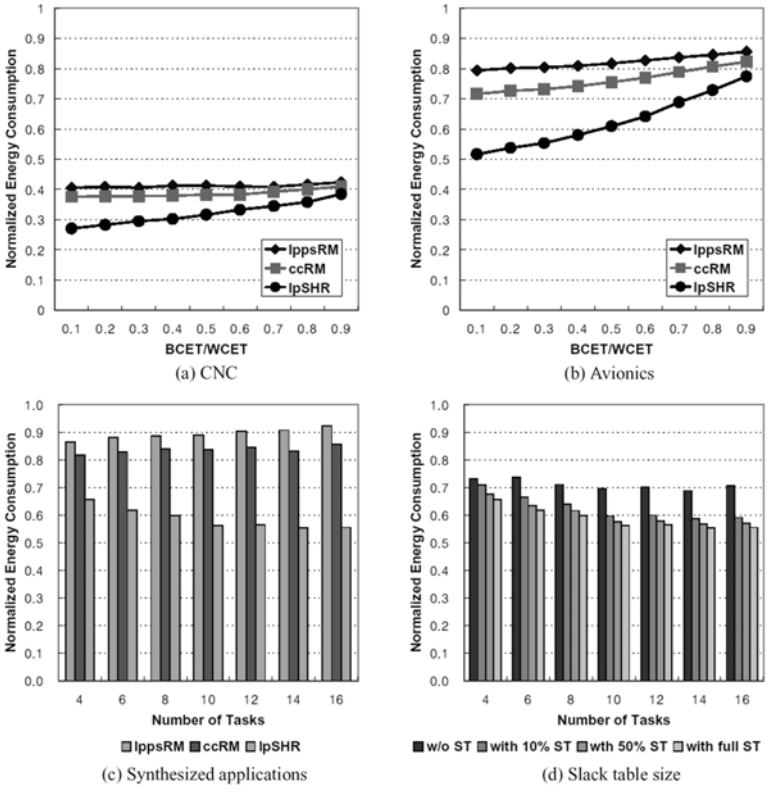

그림 7 RM 스케줄링 정책을 위하 DVS 알고리즘들과의 비교 실험 결과

고리즘들은 lppsRM[9]과 ccRM[10]을 선정하였다. 그림에서 보여지듯이, RM 스케줄링 정책 하에서도 제안된 알고리즘이 기존 알고리즘에 비해 30~37% 정도의 에너지 소모량 감소를 얻을 수 있음을 보였다.

제안된 알고리즘의 경우, 오프라인에서 분석된 슬랙 구간 정보를 구하고, 온라인 상태에서 이를 활용한다. 태스크 집합에 따라 오프라인에서 분석된 슬랙 구간 정보의 양은 달라질 수 있다. 일반적으로 내장형 시스템에서 메모리 허용량이 적음을 감안할 때, 과다한 오프라인 정보는 시스템 설계 요건에 있어서 수용되기 어려울 수 있다. 따라서, 오프라인 정보의 양을 조절할 수 있는 여과 기법이 요구된다. 실험을 통해 알아본바, 초주기내 정적 슬랙의 대부분이 긴 슬랙 구간에 속하고, 작은 슬랙 구간들이 다수 존재함을 확인하였다. 이는 일부 긴 슬랙 구간에 대한 정보만을 유지하더라도, 정적 슬랙 정보의 유용성은 크게 저하되지 않음을 의미한다. 이를 확인하기 위해, 얻어진 슬랙 정보의 양을 조절하였을 때의 에너지 효율성의 변화를 실험을 통해 관찰하였다. 그림 7(d)는 그 결과를 보여준다. 그림 7(d)에서는 얻어진 슬랙 구간 정보의 양을 전혀 이용하지 않는 경우(w/o ST), 10% 만을 이용하는 경우, 절반만을 이용하는 경우, 그리고 모두 이용하는 경우에 대한 제안된 알고리즘의 에너지 효율성의 변화를 보여주고 있다. 실험 결과, lpSHR 알고리즘이 오프라인 정보에 의해 과다하게 영향을 받지 않음을 보여주었으며, 90% 정도의 정보가 무시된 경우(즉, 10% 정도의 슬랙 구간 정보만을 활용하는 경우)에 5% 정도의 성능 저하가 있었다. 이는 제안된 알고리즘의 공간 복잡도는 조절이 가능하며, 크게 문제가 되지 않음을 말한다.

5.3 다른 알고리즘들과의 비교

최근에는 각 스케줄링 정책에 특화된 DVS 알고리즘들이 다수 제안되었는데, 특히 EDF 스케줄링 정책의 경우, 많은 효과적인 알고리즘들이 제시되었다. 그림 8은 그러한 알고리즘들과의 비교 실험 결과를 보여준다. 그림에서 비교 대상이 되는 알고리즘들은 모두 EDF 스케줄링 정책에 특화된 DVS 알고리즘들[9,10,11]로서 lppsEDF를 제외하고는 RM 스케줄링 정책에 적용이 불가능하다. 본 실험에서는 8개의 주기적 태스크들로 구성된 100개의 태스크 집합을 이용하여 실험을 실시하였다. 그림 8(a)와 (b)에서 주어진 태스크 집합들의 최악 프로세서 활용률(WCPU: Worst Case Processor Utilization)과 평균 프로세서 활용률(ACPU: Average Case Processor Utilization)은 각각 1.0 과 0.55 그리고 0.6 과 0.33 으로 하였다. 그림 8의 그림들에서 가로축은 설정될 수 있는 클록 속도의 하한선을 최고 클록 속도에 대한 상대적인 비율을 나타내며, 세로축은 전력 차단 기법에 대한 상대적인 에너지 소모 비율을 나타낸다.

스케줄 된 태스크 인스턴스에 대해 측정된 슬랙을 모두 이용할 수 있도록 하는 경우, 한 태스크 인스턴스가 다른 인스턴스에 비해 슬랙을 과다하게 이용하여 태스크 간의 슬랙 활용의 불균형이 발생할 수 있다. 이러한 슬랙 활용의 불균형이 커질 경우, 태스크 인스턴스들에 적용되는 클록 속도 및 공급 전압의 편차가 커져 에너지 소모량 감소폭은 줄어든다. 따라서, 본 실험에서는 각 스케줄링 시점에서 설정될 수 있는 클록 속도의 하한선을 변화시키면서 실험을 실시하여 가장 적합한 슬

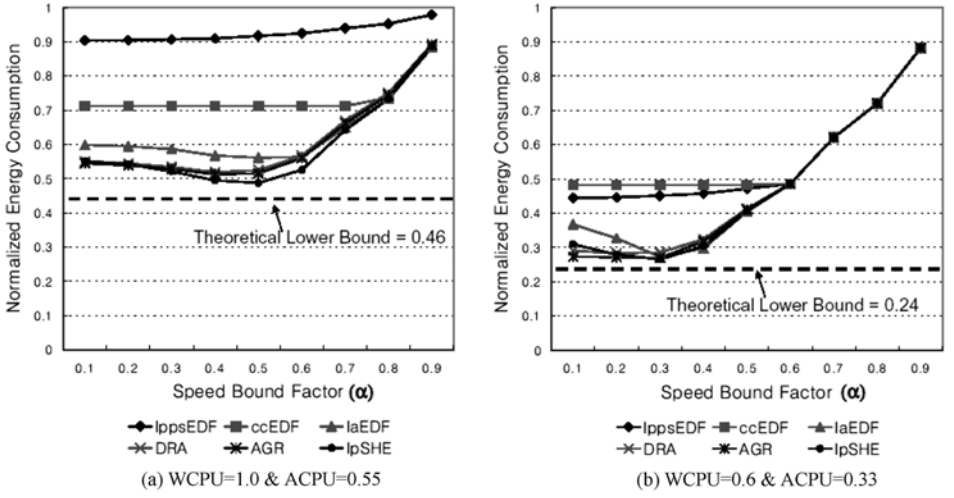

그림 8 EDF 정책에 특화된 DVS 알고리즘들과의 성능 비교

랙 배분 방법을 찾고자 하였다. 실험 결과, DVS 알고리즘들은 클록 속도 하한선이 최고 클록 속도에 대해 평균 프로세서 활용률의 비율로 설정되었을 때 가장 좋은 에너지 효율성을 보였다. 이는, 클록 속도의 하한선이 너무 낮은 경우, 태스크 인스턴스들 간의 클록 속도 편차가 에너지 효율성이 저하되고, 클록 속도의 하한선이 너무 높은 경우, 태스크 인스턴스들이 주어진 슬랙을 거의 활용하지 못하기 때문에 헤어지 효율성이 저하되는 반면, 적절한 하한선으로 제한된 경우에는 태스크 인스턴스들의 클록 속도 편차가 줄어들어 에너지 효율성이 높아졌지 때문이다.

실험결과, 제안된 알고리즘은 EDF 및 RM에 대해 보편적으로 이용 가능함과 동시에, EDF 에 특화된 DVS 알고리즘들과 비교해 그 에너지 효율성이 상당히 우수함을 보였다. 본 실험에서는 또한, 제안된 알고리즘을 비롯한 AGR[11], 그리고 laEDF[10]의 경우, 이론적 최저점에 대해 7~10% 정도의 에너지 효율성 차이만을 보였는데, 이론적 최저점의 경우, 초주기내 평균 작업량을 바탕으로 설정된 클록 속도와 공급 전압을 바탕으로 계산된 값이며 태스크들의 시간 제약성이 무시된 상태에서 계산된 값이므로 실제 얻기 어려운 값이다. 즉, 실질적으로 온라인 상태에서 얻을 수 있는(태스크들의 시간 제약성을 항상 보장하면서) 하한선은 이론적 최저점보다 높다. 따라서, 제안된 알고리즘을 포함한 기존의 EDF 스케줄링 정책에 대한 DVS 알고리즘들의 에너지 효율성은 이미 온라인 최적값에 근접하고 있음을 의미한다.

6. 결 론

본 논문에서는 효율적인 슬랙 측정 방법을 바탕으로 하여 EDF 및 RM 스케줄링 정책에 모두 활용될 수 있는 새로운 동적 전압 조절 알고리즘을 제안하였다. 실제 제안된 알고리즘은 이들 스케줄링 정책 이외에도 다른 우선순위 기반 스케줄링 정책에 모두 쉽게 이식될 수 있다. 또한, 제안된 알고리즘은 일반 스케줄링 정책에 필요한 계산 및 공간 복잡도 정도로 구현이 가능하다. 실험을 통해 보여졌듯이, 제안된 알고리즘은 EDF 및 RM 스케줄링 정책을 위해 제안되었던 기존의 알고리즘들에 비해 향상된 에너지 효율성을 보인다. 특히, RM 스케줄링 정책을 위해 구현된 lpSHR 의 경우, 기존의 가장 높은 효율성 보였던 ccRM 에 비해 40% 정도의 에너지 소모량 감소 효과를 보였다.

제안된 알고리즘이 높은 에너지 효율성을 보였지만, RM 스케줄링 정책의 경우, 이론적 최저점에 비해 아직 15% 정도의 차이를 보인다. 즉, 아직 성능 향상의 여지가 있으며, 보다 높은 에너지 효율성을 얻기 위해서는 슬랙 측정 부분 뿐만 아니라 슬랙 배분 문제에 있어서도 보다 효과적인 정책이 필요하다. 따라서, 본 연구에서는 향후 보다 효과적인 슬랙 배분 정책에 관한 연구를 진행하고 있다.

참 고 문 헌

[1] T. Sakurai and A. R. Newton, "Alpha-Power Law MOSFET Model and Its Application to CMOS Inverter Delay and Other Formulas." *IEEE Journal of Solid-State Circuits*, vol. SC-25, No. 2, pp. 584-589, Apr. 1990.

[2] I. Ripoll, A. Crespo and A. G. Fornes, "An Optimal Algorithm for Scheduling Soft Aperiodic Tasks in Dynamic-Priority Preemptive Systems." *IEEE Transactions on Software Engineering*, vol 23(6), pp. 388-400, 1997.

[3] J. P. Lehoczky and S. Ramos-Thuel, "An Optimal Algorithm for Scheduling Soft Aperiodic Tasks Fixed-Priority Preemptive Systems." in *Proceedings of the IEEE Real-Time Systems Symposium*, pp. 110-123, Dec. 1992.

[4] S. Lee and T. Sakurai, "Run-time Voltage Hopping for Low-power Real-Time Systems." in *Proceedings of the 37th Design Automation Conference*, pp. 806-809, Jun. 2000.

[5] M. Weiser, B. Welch, A. Demers, and S. Shenker, *"Scheduling for Reduced CPU Energy." in Proceedings of the 1st USENIX Symposium on Operating Systems Design and Implementation,* pp. 13-23, Nov. 1994.

[6] F. Yao, A. Demers, and A. Shenker, "A Scheduling Model for Reduced CPU Energy." in *Proceedings of the IEEE Foundations of Computer Science*, pp. 374-382, 1995.

[7] G. Quan and X. S. Hu, "Minimum Energy Fixed-Priority Scheduling for Variable Voltage Processors." in *Proceedings of the Design Automation and Test in Europe (DATE'02)*, pp. 782-787, Mar. 2002.

[8] Y. Shin and K. Choi, "Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems." in *Proceedings of the Design Automation Conference*, pp. 134-139, Jun. 1999.

[9] Y. Shin, K. Choi, and T. Sakurai, "Power Optimization of Real-Time Embedded Systems on Variable Speed Processors." in *Proceedings of the International Conference on Computer-Aided* Design, pp. 365-368, Nov. 2000.

[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 (SOSP'01)*, pp. 89-102, 2001.

[11] H. Aydin, R. Melhem, D. Mosse, and P. M.

Alvarez, "Dynamic and Aggressive Scheduling Techniques for Power-Aware Real-Time Systems." in *Proceedings of IEEE Real-Time Systems Symposium*, Dec. 2001.

[12] C. M. Krishna and Y.-H. Lee, "Voltage-Clock Scaling Adaptive Scheduling Techniques for Low Power in Hard Real-Time Systems." in *Proceedings of the Sixth IEEE Real-Time Technology and Applications Symposium*, pp. 156-165, Jun. 2000.

[13] G. Flavius, "Hard Real-Time Scheduling Using Stochastic Data and DVS Processors." in *Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 46-51, Aug. 2001.

[14] I. Hong, G. Qu, M. Potkonjak, and M. B. Srivastava, "Synthesis Techniques for Low-Power Hard Real-Time Systems on Variable Voltage Processor." in *Proceedings of the IEEE Real-Time Systems Symposium*, pp. 178-187, Dec. 1998.

[15] T. Pering and R. Brodersen, "Energy Efficient Voltage Scheduling for Real-Time Operating Systems." in *Proceedings of the 4th IEEE Real-Time Technology and Applications Symposium*, 1998.

[16] T. S. Tia, J. W. -S. Liu, and M. Shankar, "Algorithms and Optimality of Scheduling Soft Aperiodic Requests in Fixed-Priority Preemptive Systems." *Real-Time Systems*, 10(1):23-43, Jan. 1996.

[17] D. Shin, W. Kim, J. Jeon, J. Kim, and S. L. Min, "SimDVS: An Integrated Simulation Environment for Performance Evaluation of Dynamic Voltage Scaling Algorithms." in *Proceedings of Workshop on Power-Aware Computer Systems (PACS'02)*, Feb. 2002.

[18] N. Kim, N. Ryu, S. Hong, M. Saksena, C. Choi, and H. Shin, "Visual Assesment of a Real-Time System Design: a Case Study on a CNC Controller." in *Proceedings of IEEE Real-Time Systems Symposium*, pp. 300-310, Dec. 1996.

[19] C. Locke, D. Vogel, and T. Mesler, "Building a Predictable Avionics Platform in Ada: a Case Study." in *Proceedings of IEEE Real-Time Systems Symposium*, pp. 181-189, Dec. 1991.

[20] D. Shin, J. Kim, and S. Lee, "Intra-Task Voltage Scheduling for Low-Energy Hard Real-Time Applications." *IEEE Design and Test of Computers*, vol. 18(2), pp. 20-30, Mar. 2001.

김 운 석

1997년 홍익대학교 컴퓨터공학과 학사 1999년 서울대학교 컴퓨터공학과 석사 2000년~현재 서울대학교 전기컴퓨터공학부 박사과정 재학중. 관심분야는 실시간 시스템, 저전력 시스템, 멀티미디어 시스템

김 지 홍

1986년 서울대학교 계산통계학과 학사 1988년 University of Washington 컴퓨터과학과 석사. 1995년 University of Washington 컴퓨터과학 및 공학과 박사 1995년~97년 미국 Texas Instruments 사 선임연구원. 1997년~현재 서울대학교 전기컴퓨터공학부 부교수. 관심분야는 컴퓨터구조, 내장형시스템, 저전력시스템, 멀티미디어시스템

민 상 렬

1983년 서울대학교 컴퓨터공학과 졸업 1985년 서울대학교 컴퓨터공학과 석사 1989년 University of Washington 전산학 박사. 1989년~90년 IBM T.J. Watson Research Center 객원 연구원. 1990년~92년 부산대학교 컴퓨터공학과 조교수. 1992년~현재 서울대학교 전기컴퓨터공학부 교수. 관심분야는 Computer Architecture, Parallel Processing, Computer Performance Evaluation