# 전력소모 민감도와 실행시간 불확실성을 고려한 동적 전압 조절 기법†

한양대학교 정기석 · 황영시

## 1. 소 개

최근 급속도로 복잡해지고 다양해지는 임베디드 시스템의 설계에 있어 전력 소모를 최소화하는 것은 매우 중요한 문제이다. 임베디드 시스템이 고성능, 다기능화 되면서 임베디드 시스템을 구성하는 소프트웨어의 비중은 매우 커지고 있으며, 그 중에서도 운영체제의 중요성은 강조되고 있다. 과거에는 단지 실시간 처리를 위해 존재하던 임베디드 운영체제는 이제는 실시간 처리 뿐 아니라, 복잡한 응용 프로그램을 효과적으로 관리하기 위해 임베디드 시스템의 필수적인 구성요소가 되었다.

정적(static) CMOS 기술로 설계된 부분이 상당부분을 차지하는 현대의 마이크로프로세서에서 상태 변이에 따라 발생하는 전력 소모의 양(dynamic switching power consumption)은 공급 전압의 제곱에 비례하며, 이는 곧 공급 전압을 낮추면 그 낮춘 양의 제곱에 비례하여 전력 소모가 줄어든다는 것을 의미한다. 다만 전압을 낮추면 낮은 공급 전압에서 구현 가능한 한 클럭 주파수가 줄어들기 (즉, 속도가 줄어들기) 때문에 속도 지연이 있을 수 있는 문제가 있다. 하지만, 모든 구성요소가 항상 최대 속도로 동작할 필요가 없기 때문에 적절한 때 전압을 줄이면 전체 시스템의 성능은 유지되면서 전력소모를 줄일 수 있다. 이와 같이 동적으로 공급 전압을 조절하여 전력 소모를 줄이는 연구는 매우 활발히 진행되고 있으며, 실제로 Transmeta사의 Crusoe [12] 등 프로세서에서 구현되어 효과가 입증되어 왔다.

본 논문에서는 동적 전압 조절에 대해 새로운 모델링 및 설계 기법을 제시한다. 현재까지 많은 동적 전압 조절 기법에 대한 연구가 있어왔지만, 기존의 연구는 몇 가지 한계점을 갖는다. 첫째, 공급 전압과 실행시간과의 관계가 지나치게 단순화 되어 있다는 점이다. 시스템의 소프트웨어는 다양한 종류의 작업을 서로 다른 하드웨어 구성요소들에서 실행된다. 그래서 일정하게 공급 전압이 유지된다고 하더라도 각 태스크의 실행 시간은 매번 조금씩 달라질 수 있다. 이런 실행 시간의 불확실성은 각 태스크의 고유한 특성이라고 볼 수 있다. 특히 운영체제는 태스크 관리, 메모리 관리, 파일 시스템 관리, 입출력/통신 관리 및 예외 또는 인터럽트 처리 등 다양한 작업을 동시에 수행하기 때문에 운영체제의 역할이 큰 태스크의 실행일수록 실행 시간의 불확실성은 커진다 하겠다. 둘째, 가변 공급 전원에 의한 전력 소모의 모델링이 매우 간단한 방법에 의거하고 있다는 점이다. CMOS회로에서 동적 전력 소모가 전압의 제곱에 비례한다는 것은 잘 알려진 사실이지만, 각 태스크들의 실행에 있어, 모든 태스크들이 각기 다른 프로세서의 구성요소들을 활용할 수 있기 때문에 공급 전압의 증감에 대한 전력 소모 증감의 상대적 비율은 (이를 본 논문에서는 "전력 소모 민감도"라 부른다) 각각 다른 것이 일반적이다. 셋째로, 많은 논문들이 동적 전압 조절에 대한 오버헤드를 심각하게 고려하지 않고, 때론 이 오버헤드의 존재를 무시하거나, 무시할 정도로 크기가 작다고 가정하고 문제를 해결하는 경우가 많다. 이는 심각한 제한점이라 볼 수 있다. 기존의 많은 연구결과는 이와 같은 제한점들로 인해, 실제 시스템에서 전력 소모를 줄이는 것에 효과적이지 못할 수 있다는 한계가 있다. 본 연구는 이러한 한계를 극복하기 위한 운영체제 상에서의 새로운 전력 사용 모델을 제시함으로써 시스템의 전력 특성을 가장 효율적으로 파악하고, 그 특성의 올바른 이해를 전력 소모 최소화 방법에 적용한다.

본 논문의 가치는 크게 두 가지로 요약할 수 있다. 먼저, **기존의 연구에서 먼저 전역적(global)인 태스크 그래프(task graph)를 분석하여 최소 횟수의 전압 조정으로 최대의 전력 소모 효과를 얻을 수 있도록 해준다. 두 번째는 태스크의 실행 시간의 불확실성(runtime**

† 이 논문은 2004년도 한국학술진흥재단의 지원에 의하여 연구되었음(KRF- 2003-041-D00471).

**uncertainty)과 전력 소모의 민감도(power sensitivity)를 고려한다는 것이다.**

본 논문의 구성은 다음과 같다. 2절에서는 이 논문에서 해결하는 문제의 정의와 기본 정의와 가정에 대해 논하며, 3절에서는 간단한 예를 통하여 전체적인 알고리즘을 소개하고, 4절에서는 관련연구를 소개할 것이다. 5절에서는 DVS_SUC 알고리즘에 대해 자세하게 소개하고, 6절에서 실험을 통해 효율성을 입증하고, 7절에서 결론을 맺는다.

## 2. 문제의 정의 및 기본 용어 정의

### 2.1 문제 정의

이 논문에서는 저전력 설계를 위해 동적 전압 조절(dynamic voltage scaling)을 가장 효과적으로 수행하는 방법을 제시한다. 최적의 전압 조절이란 최소의 전력 조절 오버헤드로 최대의 전력 감소 효과를 올리는 것으로 정의된다. 현재까지의 많은 연구가 이 문제를 다루었지만, 대부분 실제로 오버헤드에 대한 고려가 아예 없거나 무시할 수준이라는 가정 하에, 주어진 상황에서 최적의 전압 수준을 구하는 문제에 대해 주로 연구되어 왔다. 하지만, 실제로 전압 조절에는 프로세서 자체에서만은 $10-20\mu s$ 의 시간이 걸리지만, 시스템 전체의 제반 구성 간의 오버헤드를 생각해 볼 때, 수 ms정도의 시간적 오버헤드가 있다[6]. 또한 전력 소모면에서도 수 $\mu J$ 수준의 전력소모를 유발하며, 고속/저전력 시스템에서 이 오버헤드는 결코 무시할 수 있는 것이 아니다. 다시 말해, 동적 전압 조절은 매우 효과적이기는 하지만, 그 적용에 있어 적지 않은 시간적, 전력 소모적 오버헤드가 따른다.

본 논문은 이 문제를 보다 일반적인 상황에서 풀기 위해서 몇 가지 가정을 한다. 먼저, 주어진 전압에 따른 전력 소모량의 많고 적음을 고려하기 것에서 그치지 않고, 전압이 바뀌는 양에 따른 전력 소모량의 변화량이 얼마인가를 나타내는 전력 소모 민감도(power sensitivity)를 고려한다. 둘째는 주어진 전압 및 클럭 속도에 항상 같은 실행시간이 걸린다는 가정을 없애고, 각 태스크는 같은 전압에도 다른 주변 상황에 따라 실행시간이 가변적이 된다고 가정한다. 셋째는 태스크 그래프상의 태스크 들은 전체 태스크가 모두 처리되어야 할 실시간 제한 시간이 있다고 가정한다.

본 연구는 기존의 연구와는 차별성을 갖고 다음의 문제에 대한 해결책을 제시한다.

**"주어진 태스크 그래프에서 각 태스크가 실행 시간에 대한 불확실성과 전원 전압의 조정에 따른 전력 소모 민감도가 각각 다르게 주어졌을 때, (사용자가 정의하는) N번의 전압 수준을 변화시켜 태스크들의 전력 소모량이 최소가 되게 한다면, 어디에서 어떻게 공급 전압을 바꾸어 주는 것이 최적일 것인가?"**

### 2.2 용어 정의

위의 문제를 풀기위한 입력 정보와 출력에 대한 정보, 그리고 각각이 어떤 의미로 어떻게 주어지고 계산되는지에 대해 설명하기 위한 몇 가지 기본 용어를 정의한다.

**태스크 그래프 G(V, E) TG**

본 논문에서 가정하는 G(V,E)는 싸이클이 없는(즉, 루프(loop)가 없는) 방향성 그래프이다. V는 노드의 집합이며, 각 노드들은 하나의 태스크에 상응한다. 각 노드들 간의 순서 (order)는 에지로 나타낸다. 만일 노드 v에서 노드 u로 가는 에지가 있을 경우, 반드시 노드 u는 노드 v가 시행되고 나서 시행되어야 한다. 그림 1은 간단한 태스크 그래프의 예를 보여준다.

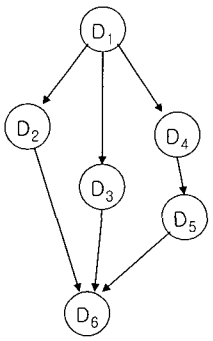



그림 2 태스크 그래프

**전력관리 포인트(Power Management Point) PMP**

동적 전압 조절 시스템에서는 전압을 동적으로 바꾸어 줌으로써, 그 전압에 가능한 클럭 주파수를 계산하게 된다. 이 논문에서는 전압을 바꾸어주는 시점을 전력 관리 포인트라고 정의하며, 어떤 태스크의 실행 전에 그 태스크에 맞는 전압 수준을 계산하여 전압 수준을 바꾸어주는 것이 효과적이라 판단될 때, 전압을 바꾸어 준다. 본 연구에서, 전력 관리 포인트는 실행 전에 명령어 형태로 실행 코드에 삽입된다고 가정하며, 일정한 전압과 전력 소모의 오버헤드가 존재한다고 가정한다. 즉, 정적인 분석을 통하여, 전체 태스크 그래프에서 어떤 부분이 가장 전력 감소를 위한 핵심 전력 관리 포인트인가를 계산한다.

**전력소모 민감도 (Power Sensitivity) PS**

기본적으로 스위칭에 의한 동적 전력소모는 공급 전압의 제곱에 비례한다. 특히 CMOS 설계가 많은 부분

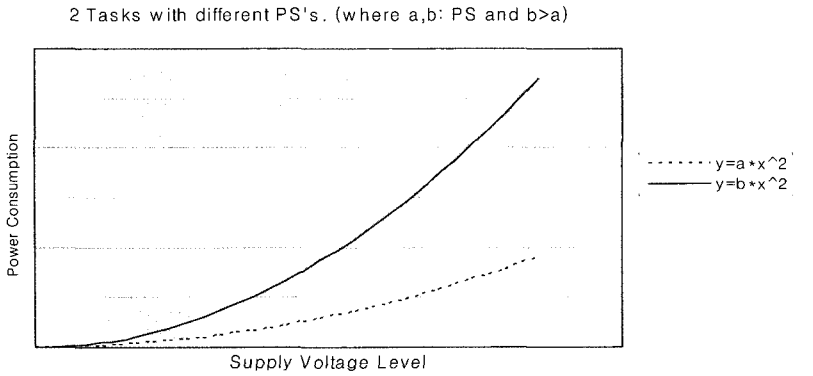



그림 2 전력소모민감도(PS)가 다른 두 태스크들의 전압에 따른 전력 소모 패턴

을 차지하는 프로세서에서는 공급 전압과 동적 전력 소모는 비교적 정확한 상관관계를 갖는다. 즉 power$\propto V^2$ 이다. 하지만, 태스크의 특성상 그 비례 상수는 다를 수 있다. 즉 그림 2에서의 경우처럼 두 태스크의 비례 상수가 다를 경우, 전력 소모량의 변화률은 다르다. 이런 현상이 나타나는 이유는 여러 가지가 있을 수 있으며, 대표적으로 각 태스크가 수행하는 작업에 따라 프로세서의 서로 다른 구성요소들을 이용하게 되고, 따라서 전력 소모의 증감이 속도에 대비되는 정도가 다르기 때문이라고 생각할 수 있다. 이를 이 논문에서는 전력 소모 민감도(power sensitivity)라고 부른다. 전력 소모 민감도의 정확한 모델링을 위해서는 그 자체가 하나의 비선형 수식이어야 하지만, 이 논문에서는 간단히 각 태스크에 대해 실행하는 프로세서에서의 프로파일링을 통해서 구해진 하나의 상수로서 취급한다. 즉, V를 공급전압이라고 할때, 전력 소모 $P = PS * V^2$ 으로 계산된다.

**실행시간 불확실성(Runtime Uncertainty) RU**

실행 시간 불확실성(runtime uncertainty)은 어떤 태스크가 매회 실행될 때마다 실행시간의 차이가 나는 정도를 모델링하는 개념으로 실행시간의 분산을 의미하며, 실행시간 불확실성(RU)은 최대 실행 시간(WCET, Worst Case Execution Time)과 실제 실행시간($TIME_{EXE}$)과의 차이의 제곱의 평균으로 계산된다. 즉,

$$RU = \frac{1}{n}\sum^{n}(WCET - Time_{EXE})^2 \qquad (1)$$

**태스크 실행 여유분(Task Slack): TS**

본 논문에서의 태스크 그래프는 하드 실시간 제한(hard realtime constraint)을 갖고 있다고 가정한다. 즉, 전체 태스크가 실행될 최대 허용 클럭 싸이클의 수는 결정되어 있다고 가정한다. 각 태스크들의 평균 실행 시간(ACET)와 최대 실행 시간(WCET) 역시 이미 알려져 있다고 가정한다. 이때 각 실행의 최대 임계 경로를 계산(critical path) 했을 때, 임계 경로의 모든 태스크가 최악의 실행시간을 갖고 실행해도 전체의 전역적(global) 제한시간은 반드시 만족할 수 있어야 한다. 임계 경로의 총 실행 시간과 전체 수행 제한 시간과의 차이를 여유분(slack)이라고 정의하며 이를 태스크 그래프 전체의 여유분(slack)이라고 정의할 수 있다. 그림 3에서는 어떻게 여유분이 계산되는지를 보여준다. 이 여유분은 프로세서의 클럭 스피드를 줄여서 전력소모를 줄이는 동적 전압 조절(dynamic voltage scaling)에서는 매우 중요한 요소라고 할 수 있다. 본 연구에서는 이 여유분(slack)값을 탐욕적(greedy)으로 사용하기 보다는 전체적인 분석을 통해 가장 효과적인 태스크의 실행에 적절히 분배(slack distribution)하기 위한 분석 기법을 제안한다.

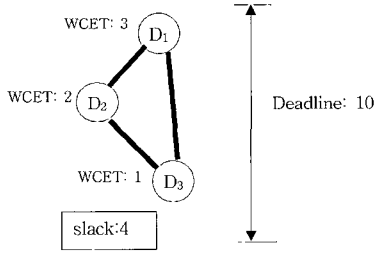



그림 3 태스크 그래프에서의 여유분(slack) 계산

## 3. 간단한 예

이번 장에서는 간단한 예를 통하여 논문이 제시하는 해결 방법을 설명한다. 그림 4의 태스크 그래프는 6개의 태스크를 포함한다.

이 예에서의 각 태스크들에 대해 우리는 프로파일링(profiling)을 통해 전력소모민감도(PS), 실행시간 불확실성(RU)을 계산해 두었다고 가정한다. 또한 각 태

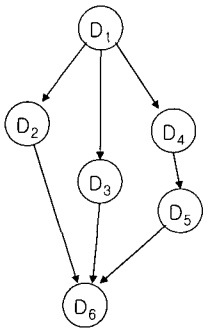



그림 4 태스크 그래프 예

스크의 최대실행시간은 알려져 있다고 가정하고, 그 값이 12 단위 시간이라고 가정한다. 논의를 간단히 하기 위해서 태스크를 수행하는 프로세서의 수에는 제한이 없다고 가정한다. 이 가정을 바탕으로 표 1에서 보인 바와 같이 전력 소모 효율성에 대한 중요도를 계산할 수 있다. 표 1의 계산의 통해 전력 소모를 궁극적으로 줄이기 위해 가장 중요한 것은 시간 여유분의 분배이다. 이 시간 여유분은 각 태스크에게 할당된 여유분을 사용했을 때, 얼마나 전역적 의미에서 전력 소모를 줄일 수 있는지에 따라 배분된다. 본 연구의 문제는 $N$개의 전력 관리 포인트를 삽입하여 태스크 전체의 전력 사용을 최소화하려는데 목적이 있으며, 전력 관리의 중요도를 나타내는 Power Management Criticality (PMC) 값의 계산을 통해 가장 효과적으로 전력을 줄일 수 있는 곳을 정하고, 이 PMC값에 따라 시간 여유분을 분배하여, 각 태스크의 실행 속도를 시간 여유분을 이용하여 늦추어 줌으로써 전체 전력 소모를 최소화하려는데 본 연구의 목적이 있다. 표 1에서는 전체적인 전력 소모를 최소의 오버헤드로 줄이기 위해서는 $D_6$의 실행 전이 가장 중요한 전력 관리 포인트임을 보인다. 자세한 알고리즘에 대한 설명은 5장에서 이루어진다.

표 1 그림 5의 태스크그래프에 대한 전력소모 중요도 계산 예

| Task | WCET | PS | RU | slack | Power Management Criticality |
|---|---|---|---|---|---|
| $D_1$ | 3 | 1 | 1 | 2 | $\min\{(1\cdot\frac{1}{2}\frac{5}{6}),(1\cdot\frac{1}{2}\frac{3}{4}),(1\cdot\frac{1}{2}\frac{8}{9})\}$ $= 0.3$ |
| $D_2$ | 2 | 2 | 3 | 5 | $(2\cdot\frac{1}{2}\frac{3}{6}) = 0.5$ |
| $D_3$ | 3 | 2 | 1 | 4 | $(2\cdot\frac{1}{2}\frac{3}{4}) = 0.75$ |
| $D_4$ | 2 | 2 | 4 | 2 | $(2\cdot\frac{1}{3}\frac{5}{9}) = 0.518$ |
| $D_5$ | 3 | 3 | 2 | 2 | $(3\cdot\frac{1}{3}\frac{7}{9}) = 0.77$ |
| $D_6$ | 2 | 4 | 2 | 2 | $\min\{(4\cdot\frac{1}{2}\frac{4}{6}),(4\cdot\frac{1}{2}\frac{2}{4}),(4\cdot\frac{1}{3}\frac{7}{9})\} = 1$ |

위의 PMC값에 따라 여유분을 분배한 결과와 결과적으로 분배된 여유분을 갖고 속도를 낮추어 진행될 수 있는 새로 할당된 실행 시간은 아래와 같다.

| | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ |
|---|---|---|---|---|---|---|
| 분배된 slack | 0 | 4 | 3 | 0 | 1 | 1 |
| 할당 실행 시간 | 3 | 6 | 6 | 2 | 4 | 3 |

## 4. 관련 연구

저전력 시스템 설계에 대한 연구와 개발은 최근 수년간 매우 활발히 진행되어 많은 발전을 이루었다[8,11]. 하지만, 현재까지의 연구는 대부분이 트랜지스터나 로직 수준, 또는 컴퓨터 구조 단계에서의 전력 소모 모델링 및 전력 소모 최소화에 국한된 부분이 없지 않다. 이는 시스템의 구성요소로서의 임베디드 운영체제의 역할이 매우 큰 것에 비해 연구 노력이 충분하지 못했다는 것을 의미한다. 현재 운영체제 단계에서 가장 활발히 연구되는 저전력 설계 기법은 크게 동적 전력 관리 기법 (dynamic power management)과 동적 가변 전압 조절 기법(dynamic voltage scaling)으로 나눌 수 있다.

동적 전력 관리 기법은 우선 프로파일링(profiling)과 같은 기법을 통하여 시스템의 각 동작 상태에서의 평균적 전력 소모량 및 주로 사용하는 시스템 구성 요소 등을 파악하고, 또 각 동작 상태간의 전이 패턴 및 각 모드전이에서 발생하는 전력적/시간적 오버헤드를 측정한다. 이 정보를 이용하여 부분적인 시스템의 셧다운(shutdown) 및 대기(stand-by) 상태를 유지 운영함으로써 전력 소모를 최소화하는 방법이다. 잦은 셧다운(shutdown)과 실행 준비(wakeup)의 상태 전이는 상당한 시간적/전력 소모적 오버헤드를 발생시키기 때문에 되도록 한번 셧다운 상태가 되면 되도록 오래 유지되도록 하는 것에 초점을 맞춘다. 하지만 일반적으로 복잡한 응용 프로그램에 있어 어떤 시스템 구성요소가 얼마나 오랫동안 셧다운 상태를 유지 할 것인가를 예측하는 것은 매우 힘들다. 여러 연구들이 이 분야에서 있지만[8,10,13], 기본적인 한계점이 정확한 미래 실행 예측의 어려움에 있다고 하겠다.

두 번째로 매우 활발히 연구되고 있는 방법이 동적 가변 전압 조절 기법이다[9,14,15]. 이 방법은 전압과 프로세서의 클럭 주파수를 PLL 및 VCO 등의 회로를 통하여 동적으로 바꿀 수 있다는 기술을 근거로 한다. 즉 비교적 적은 오버헤드로 점차적 속도 조절(slowdown)로 전력 소모를 줄이는 기법이다. 여러 연구가 있는데, 스케쥴링이 실행시 동적으로 이루어지는 on-line 스케쥴링인지 그렇지 않은 off-line인지, 각

태스크의 우선순위가 고정인지, 또는 가변인지, 또 경성(hard) 실시간 시스템인지, ,연성(soft) 실시간 시스템인지, 또한, 한 태스크 안에서 전압을 동적으로 조절하는 태스크 내의 전압 조절(intra-task voltage scheduling) 기법[4,7,14]과 태스크 단위로 일정한 전압을 조절하는 태스크 간의 전압조절(inter-task voltage scheduling) 기법[5,9]인지 등 매우 다양한 연구가 있었다. 이들 대부분의 연구는 각 주어진 실시간 환경에서 전력 소모를 최소화하기 위해 최적의 전압수준을 찾는 다양한 기법과 그 전압 하에서 어떤 실시간 스케쥴링 알고리즘이 효과적인지를 보였다. 하지만, 대부분은 아쉽게도 실제 전압의 조절에서 오는 시간적, 전력 소모적 오버헤드를 다루지 않았으며, 비교적 최근에 이르러서야, 몇몇 연구에서 시간적, 전력적 전압 조절 오버헤드를 다루고 있다. [7]에서는 태스크의 남아있는 여분의 WCET에 따라 태스크의 전압 조정의 오버헤드를 고려한 패러미터를 정하고, 이에 따라서 전압을 조절한다. [6]에서는 저전력 작업 요구 분석과 그에 따른 전력 조절 오버헤드를 고려하여, 각 전압 조절 포인트에서 실시간 제한 시간을 맞추면서 전압을 최적으로 줄일 수 있는 전압 수준을 정해준다. [2]에서는 공급 전압과 트랜지스터의 임계 전압(threshold voltage)를 동시에 정해서 동적 전력 소모와 누설 전력 소모(leakage power consumption)을 linear programming 기법을 사용하여 줄이고자 하였다. [1]에서는 태스크 그래프에서의 태스크 상호간의 실행 연관성에 관한 통계를 바탕으로 태스크의 실행 여유분을 효과적으로 사용하여 전체적으로 전력 소모를 줄이는 기법을 제안하였다. 본 연구는 다음과 같은 점에서 다르다고 할 수 있다. (i) 실행 시간의 불확실성, 전력 소모의 민감도를 태스크 공급 전압 결정에 고려했고, (ii) 전역적 분석을 통한 태스크 그래프의 전체 시간 여유분을 적절히 분배하는 방법을 제안했다는 점이 기존의 방법과는 차별화되는 독창적이 접근이라고 할 수 있다.

## 5. 전력 민감도와 실행시간 불확실성을 고려한 전압조절: DVS_SUC 알고리즘

이 장에서는 본 논문이 제시하는 알고리즘인 DVS_SUC (Dynamic Voltage Scaling with Sensitivity and Uncertainty consideration)이 어떻게 실행되는지에 대해 설명한다.

주어진 태스크들은 하나의 태스크 그래프 상에서 각 태스크의 실행 의존 관계가 정의된다. 태스크들은 위에서 언급한 전력 소모 민감도(PS), 실행 시간 불확실성(RU), 그리고 시간 여유분(TS)의 세 가지 특성을 갖는다. 각각의 특성에 대한 값은 크게 프로파일링을 통해 얻고, 또 각 태스크 그래프의 분석을 통해 계산된다.

### 5.1 에너지 소모 모델링

일반적으로 시스템 수준에서 하나의 태스크를 실행하는데 소요되는 에너지양 $E$는 $E \propto V^2 \cdot N_{cycle}$로 표현된다[14]. 또한 일반적으로 CMOS회로에서 회로에 공급되는 클럭(clock) 주파수는 공급 전압에 비례한다고 가정된다. 다만, 이 논문에서는 각 태스크 마다 같은 프로세서 상에서 실행된다고 해도 그 때 그 때 활성화되는 프로세서 구성요소 및 소프트웨어 구조의 다양성으로 인해 $E \propto V^2 \cdot N_{cycle}$ 식의 비례 상수는 각 태스크 마다 다르며, 이를 전력 소모 민감도(power sensitivity)라고 한다. 또한, 일정한 전압 레벨이 유지되어 클럭 주파수가 일정하게 유지되더라도 실행 시간 ($N_{cycle}$)은 매 실행시마다 다를 수 있으며, 얼마나 달라지는가는 태스크의 특성으로 간주한다. 즉, 태스크의 실행 시간은 실행 때마다 달라지는 실행시간 불확실성(uncertain execution time)을 갖는다고 가정한다. 이런 가정 하에 본 논문에서는 주어진 태스크의 평균 총 에너지 소모량을 다음과 같이 계산한다.

$$E_{AVG} = \sum_{\forall task\ T} E(T) \qquad (2)$$

이때, $E(T)$는 태스크 $T$의 에너지 소모량을 나타낸다.

### 5.2 전력 관리 중요도(Power Management Criticality, PMC)의 계산

각 태스크에 대해 PS, RU, TS의 값을 정한 후, 전력 관리 중요도(Power Management Criticality)를 계산하며 이에 따라 여유분(slack)을 분배한다. 일반적인 태스크 그래프에 대해 최적의 전력 관리 중요도를 구하는 문제는 NP-hard 문제이므로, 이 논문에서는 효과적인 휴리스틱 알고리즘을 제안한다.

표 1에서 보는 바와 같이 여유분이 하나의 태스크 그래프 상의 태스크들은 같은 여유분을 갖는 집합으로 나눌 수 있다. 여유분의 분배는 실행시간 불확실성과 반비례하여 이루어진다. 이는 실행시간이 보다 확실한 태스크에 많은 시간 여유분을 배당하려는 노력이다. 이는 불확실한 실행 시간을 갖는다는 것은 여유분을 할당해도 그 만큼 전력 소모가 감소될 가능성이 적으며, 또 따라서 실행 시간 지연으로 인해 제한 시간을 만족못할 가능성도 커지기 때문이다. 실행시간 불확실성(runtime uncertainty, RU)은 실제 실행시간과 최악(최대) 실행 실행시간과의 분산으로 계산된다. (수식 (1) 참조) 즉, 실행 시간이 매번 실행 시에 많이 달라

지는 태스크가 높은 RU 값을 갖는다고 하겠다.

실제로 주어진 태스크 그래프의 경로가 있을 때, $t_1$에서 $t_N$까지 N개의 태스크로 이루어진 실행 경로에서, 임의의 태스크 ti의 총 실행시간 불확실성에 대한 상대적 불확실성을 계산하여 시간 여유분 분산에 사용한다. 이와 같은 상대적 불확실성 $R(t_i)$은

$$R(t_i)=\frac{1}{N-1}\frac{\sum_{k=1,k\neq i}^{N} t_k's\,RU}{\sum_{k=1}^{N}(t_k's\,RU)} \quad (3)$$

로 계산된다. 위의 식을 계산해보면, $R(t_i)$값은 상대적으로 각 RU값에 반비례함을 알 수 있다. 즉, RU 값이 크면 클수록 실행시간의 불확실성이 커지기 때문에 상대적으로 전력 소모면에서의 중요도가 떨어지게 되고, 결국에는 적은 여유분 값을 분배받는다. 이 상대적인 불확실성에 대해 각각 태스크가 갖고 있는 전력소모 민감도(power sensitivity)를 기준으로 하여 얼마의 전력소모가 감소될 수 있는지를 계산한다. 즉, 상대적 실행시간 불확실성에다 전력 소모 민감도 (power sensitivity, PS)를 곱하면, PMC(Power Management Criticality)를 구할 수 있다.

$$PMC(t_i)=R(t_i)\times PS(t_i) \quad (4)$$

만일 하나의 태스크가 여러 실행 경로에 포함되어 있을 경우 PMC값은 각 경로에 대한 최소 값으로 결정된다. 이는 실제로 시간 여유분은 가장 여유가 없는 경로에 의해 제한을 받기 때문이다.

### 5.3 여유시간의 분배(Slack Distribution) 및 총 에너지 소모 계산

전체 태스크 그래프가 수행되는 데는 하드 실시간 제한시간(hard real time deadline)이 존재한다고 가정한다. 즉, 각 태스크가 최악의 경우 실행되는 시간에는 상한선(upper limit)이 있으며 이 상한선은 주어진다고 가정한다. 최악의 실행 시간으로 모든 태스크가 수행한다고 해도 하드 실시간 제한시간(hard real-time deadline)을 만족한다고 가정한다. 하지만, 일반적으로 최악의 실행시간이 동시에 모든 태스크에서 발생할 확률은 매우 적기 때문에 일반적으로 최악의 총 임계 실행시간과 실제 하드 제한 시간(hard deadline)과 일치한다고 해도 실제로는 실행에 대한 실행시간 여유분이 있게 마련이다. 이 여유분을 이 논문에서는 여유분(slack)이라고 부른다. 이 여유분은 평균적으로 얼마나 각 태스크의 실행시간을 늦추어 실행할 수 있는지를 가늠하는 정도이기 때문에 전력소모를 줄이는데 있어 매우 중요한 근거 값(metric)라고 할 수 있다. 그러므로 문제는 어떻게 하면 이 여유분을 각 태스크에 분배하여 적절히 실행시간을 가장 효과적으로 조정하느냐에 있다고 할 수 있다.

전력 관리 중요도(PMC)가 계산된 이후에 여유분의 분배는 PMC 값에 의거하여 분배된다. 몇 개의 전력 관리 포인트를 삽입할 것인가에 대한 결정이 이미 정해져 있다고 할 때, 여유분의 분배는 PMC값에 비례하여 분배된다. 즉, 하나의 실행 경로에 PMC값이 커서 전력 관리 포인트가 삽입될 큰 M개 태스크 $t_1 \ldots t_M$이 있을 때, 특정 $t_i$ 태스크에 대하여 전체 여유분에 대한 분배의 여유분 $slack(t_i)$는

$$slack(t_i)=\left\lceil slack_{total}\times\left(\frac{PMC(t_i)}{\sum_{k}^{M}PMC(t_k)}\right)\right\rceil \quad (5)$$

로 계산된다. 즉, PMC 값이 가장 큰 M개의 태스크를 정하고, 이에 대해 $slack(t_i)$값을 계산하게 된다. slack 값은 정수값을 갖는다고 가정하며, 이를 위해 ceiling 함수를 적용한다. 이때, 우선순위에 따라 각 태스크를 위해 slack을 할당하며, 총 slack이 소진될 때까지 분배된다. 분배 받은 $slack(t_i)$의 값이 0이 아닌 태스크들 중, 효과가 가장 큰 태스크들로부터 N개의 전력 관리 포인트가 해당 태스크 실행 시에 삽입되며, 이를 통해 전력 소모가 감소되는 결과를 얻는다. 만일 $V_{adjust}(t_i)$를 $slack(t_i)$만큼의 여유시간에 따라 조정된 전압 수준이라면, 이와 같은 전압 수준 조정에 따라, 실제 실행되는 클럭 싸이클의 수에는 변함이 없다고 가정하면, 태스크 그래프 TG의 총 에너지 소모량 $Energy_{total}(TG)$는

$$Energy_{total}(TG)=\sum_{i}^{N}(V_{adjust}(t_i)^2\times N_{cycle}) \quad (6)$$

로 계산된다.

### 5.4 전체 알고리즘 요약

*ALGORITHM DVS_SUC:*

1. Obtain profiling information for each task: WCET, PS, RU
2. Compute the task slack (TS) value for each task
3. Analyze the task graph, TG to compute "power management criticality (PMC)"
4. Based on PMC, distribute the slacks to the tasks with highest priorities.
5. Insert PMP in the task graph with the optimal voltage level

## 6. 실험 설정 및 결과

본 연구에서 제안된 기법의 효용성을 입증하기 위해서 모의 실험용 시뮬레이터 소프트웨어인 DVS_SUC를 C++ 언어로 작성하였다. 이를 Intel Pentium 4 PC와 Redhat 리눅스 운영체제 상에서 실행하였다. 실행시간은 전 실험에 걸쳐 1,2 초안에 결과가 나왔으며, 실행시간은 매우 빠르기 때문에 본 연구 결과에는 구체적으로 나타내지 않았다. DVS_SUC는 주어진 태스크 그래프에서 각 태스크의 전력 소모 민감도, 실행의 불확실성에 변화를 랜덤으로 생성하는 프로그램, 그리고, 각 태스크에서 시간 여유분을 계산하는 프로그램 등으로 구성된다. 이를 바탕으로 각 태스크들에 대하여 전력 관리 중요도 및 분배된 시간 여유분을 계산하고, 이를 바탕으로 태스크 그래프의 총 에너지 소모량을 계산하도록 하였다. 에너지 소모량은 수식 (3)에 의해 나타낸 바와 같이 $E_{AVG} = \sum_{\forall 태스크\ T} E(T)$로 계산하였다. 또한 다양한 프로세서 모델에 대한 실험을 위하여, Intel사의 XScale 프로세서 모델[3]을 이용하여 전형적인 태스크 실행 시간, 전력소모량에 관한 정보 및 전력 관리 오버헤드에 대한 정보를 정하였다. 프로세서의 기본 클럭은 공급 전압이 1.3volt에서 200MHz로 하였으며, 전압 조절 기법에 의해 배분된 시간 여유분을 이용하여, 최대 50MHz까지 클럭 빠르기가 늦추어 질 수 있다고 가정하였으며, 공급 전압에 대한 클럭 주파수는 비례한다고 가정하였다. 또한 정적인 전압 조절 기법의 한계를 극복하기 위해 최악의 경로로 최악의 실행시간 발생 시 실시간 제한 조건을 만족하지 못하는 경우를 방지하기 위해 실제 동적 전압 관리를 위해 최대 300MHz까지 클럭 주파수가 빨라 질 수 있다고 가정하였다. 제안된 기법에서는 프로세서의 개수에는 제한이 있어도 되고 없어도 되는데, 실험의 용이성을 위해 프로세서의 개수는 두 개로 제한하였다. 전력 관리 포인트의 삽입으로 인한 오버헤드는 주어진 프로세서 속도에서 50 싸이클이 걸린다고 가정하였으며, 전압 변환에 대한 전력 소모적 오버헤드는 상수라고 가정하였다. 이런 전압 변환에 대한 전력 소모는 전체 에너지 소모량에서 계산하지 않았으나 전력 관리 포인트의 개수에 비례한다고 가정하였다.

본 연구의 독창적인 문제 설정으로 인해 기존의 벤치마크 프로그램을 사용할 수 없었으며, 기존의 벤치마크를 수정하여 자체적으로 만들어진 태스크 그래프를 실험에 사용하였다. 총 5개의 태스크 그래프를 만들어 실험 대상으로 사용하였으며, 이들 태스크 그래프들이 포함하는 태스크의 개수는 7개에서 20개까지 다양하게 하였으며, 각 태스크의 전력 소모 민감도 및 실행 시간 불확실성의 값을 다양하게 변화시켜 실험하였다. 실험 결과는 각 태스크 그래프에 대해 1만 번 이상의 랜덤 시뮬레이션과 본 논문에서 제안된 정적인 전압 조절 기법과의 비교를 통해 오차를 계산함으로써 제안된 기법의 효율성에 대해 입증하였다. 다양한 가정과 실험 환경을 충분히 고려하기 위해 제안된 기법의 효용성을 위해 다음과 같은 실험을 시행하였다.

### 6.1 전력 관리 중요도의 효율성 및 DVS_SUC 정확성

이 실험은 PMC 값의 계산에 대한 실제 태스크 그래프에서의 에너지 소모 감소와의 상관 관계 및 효율성을 검증하기 위한 실험이다. 각 태스크 그래프에 대해서 PMC 값을 구하고, 그 PMC 값에 의해서 전력 관리 포인트를 3개씩 정했다. 그리고 각 전력 관리 포인트의 할당된 최대의 시간 여유분을 이용하여 프로세서의 속도를 늦추고 에너지 소모를 줄인 후에 에너지 소모 값을 (1만 번 수행한 후 얻은) 최적의 값과 비교해 보았다. 단, 실행 시간의 경우 실행시간 불확실성 값을 근거도 WCET와는 다른 실제 실행 시간을 갖도록 하였다. 다양한 태스크 그래프에 대해서 상대적 에

표 2 DVS_SUC 효용성 분석

| TG | 태스크수 | DVS_SUC | OPTIMAL | RelError |
|---|---|---|---|---|
| TG1 | 10 | 2203 | 1980 | 0.11 |
| TG2 | 7 | 1038 | 980 | 0.06 |
| TG3 | 13 | 3574 | 3003 | 0.19 |
| TG4 | 13 | 5434 | 4698 | 0.16 |
| TG5 | 10 | 1253 | 1100 | 0.14 |
| TG6 | 15 | 4563 | 4043 | 0.13 |
| TG7 | 14 | 5432 | 5099 | 0.07 |
| TG8 | 20 | 10293 | 9098 | 0.13 |
| TG9 | 10 | 1235 | 1034 | 0.19 |
| TG10 | 17 | 8098 | 7098 | 0.14 |
| 평균 | | | | 0.13 |

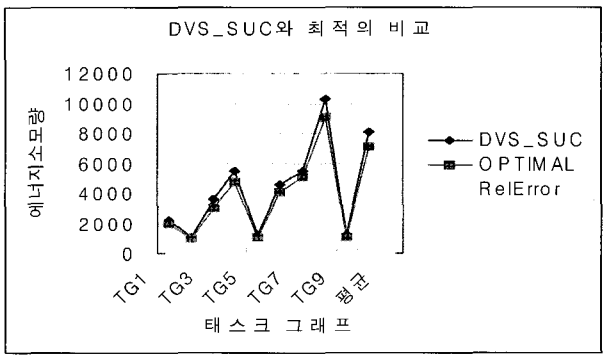


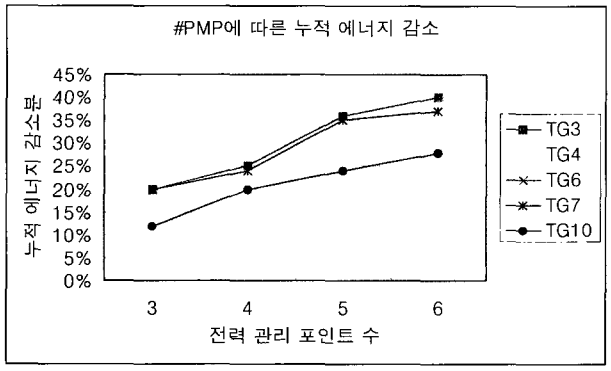

그림 5 DVS_SUC의 효율성 결과

러는 13% 정도로 나타났다. 이는 적은 에러는 아니지만, 상대적으로 매우 동적으로 변하는 태스크 그래프들을 실험 대상으로 사용했다는 점과 또한 PMC 값을 구하는 시간이 매우 빠르다는 점에서 효율성이 뛰어나다고 할 수 있다. 결과는 표 2와 그림 5에서 보여준다.

### 6.2 전력 관리 포인트의 수에 관한 실험

이 실험에서는 전력 관리 포인트 수에 따른 에너지 감소분의 추이를 살펴본다. 전력 관리 포인트를 3개에서 6개까지 증가시켜 봄으로써 전체적인 오버헤드의 증가 추이와 에너지 소모 감소 추이를 살펴 보았다. 태스크 그래프는 5개를 테스트 했으며 모두 10개 이상의 태스크를 갖고 있고 PS, RU 값은 다 다르게 정했다. 그림 6에서의 결과를 보면, 전력 관리 포인트의 수가 증가함에 따라 예상한 바와 같이 추가 에너지 소모 감소 효과가 나타났으며 이에 PMC 값의 효용성이 잘 나타난다고 할 수 있다. 하지만, 경우에 따라서는 PMC값에 따라 전력 소모 감소 효과가 나타나지 않을 수 있다는 것을 TG3의 경우를 보면 알 수 있다.

그림 6 전력 관리 포인트 수에 따른 에너지 감소량 분석




### 6.3 태스크 개수의 다양화에 대한 실험

본 실험에서는 하나의 태스크 그래프(TG2)를 두 번에서 네 번 복사하여 서로 연결함으로써 하나의 큰 태스크 그래프를 구성한 후, 태스크 개수가 증가했을 때 DVS_SUC 알고리즘의 효용성에 대한 검증을 한다. 먼저 실행 시간 면에서는 정적 분석의 알고리즘인 관계로 큰 태스크 그래프에 대해서도 실제 알고리즘의 수행시간은 크게 변하지 않았다. 또한 전력 관리 포인트를 전체 태스크 수의 10%로 제한하여 비교적 소수의 전력 관리 포인트를 삽입한 후 실험 하였다. 실험한 결과에 따르면 태스크 개수가 늘어나도 DVS_SUC에 의한 에너지 감소 효용성은 크게 떨어지지 않는 것으로 나타났다. 또한 같은 태스크 그래프가 중복하여 나타남으로써, 시간 여유분의 불균형도가 심해져서 더 많은 에너지 감소를 유발하는 가능성이 더 커진 면이 발견되었다. 하지만, PMC값에 의한 여유분의 분배 값은 경우에 따라 에너지 소모 감소 효과와 일치하지 않는 경우가 나타났으며, 이는 실제 실행에 있어 태스크 개수가 증가함에 따라 다양한 가능 실행 경로에 대한 정적인 분석이 정확히 하지 못하는 한계에 기인한 것으로 판단된다. 결과는 그림 7에 요약되어 있다.

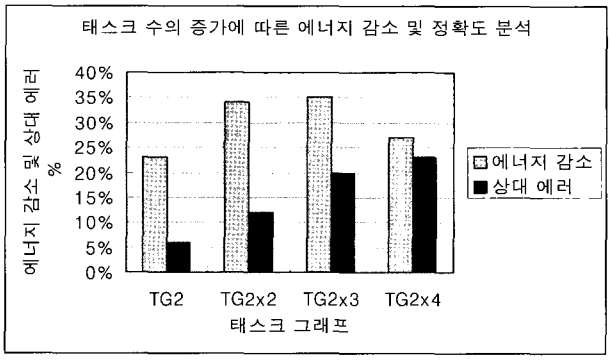



그림 7 태스크 수의 증가에 따른 에너지 감소 및 정확도

## 7. 결론 및 향후 계획

본 논문에서는 동적 전압 조절에 대해 보다 새로운 모델링 및 설계 기법을 제시하였다. 본 연구가 다른 기존의 연구와 차별화되는 것은 크게 (i) 실행 시간의 불확실성, 전력 소모의 민감도를 태스크 공급 전압 결정에 고려했고, (ii) 전역적 분석을 통한 태스크 그래프의 전체적 시간 여유분을 적절히 분배하는 방법을 제안했다는 점이다. 본 연구에서 제안된 방법의 효용성을 입증하기 위해 다양한 실험을 통하여 실제로 주어진 태스크 그래프에서의 총 에너지 소모량이 줄어드는 것을 확인할 수 있었으며 매우 적은 수의 전력 관리 포인트의 삽입에도 전체 에너지 소모량이 20-30%씩 감소하는 효과를 입증할 수 있었다. 하지만, 제안된 기법은 정확한 각 태스크의 모델링에 의존하기 때문에 정확한 입력 모델링을 위해서 많은 노력이 필요하다. 모의실험을 통해 실험한 결과, 특히 전력 민감도, 실행시간 불확실성 및 시간 여유분을 정확히 얻기 위한 프로파일링 (profiling) 기법이 더 연구되어야할 것으로 나타났다. 또한 프로세서의 모델에 따라 실행 결과가 매우 크게 달라짐을 알 수 있었고, 이에 따라 보다 자세하고 정확한 프로세서 모델링이 필요하다는 것을 알았다. 앞으로는 실제 운영체제의 모델링과 결합하여 응용 태스크 그래프의 실행과 운영체제 자체의 전력 소모 패턴을 동시에 고려한 데이터를 바탕으로 보다 면밀한 전력 소모 민감도 및 실행 불확실성에 관한 데이터를

수집하여 본 연구에서 제시한 알고리즘의 효용성과 정확도를 보다 발전시킬 계획이다.

## 참고문헌


[1] D. Zhu, D. Mosse, and R. Melhem, "Power-Aware Scheduling for AND/OR Graphs in Real-Time Systems," in IEEE Trans. on Parallel and Distrubuted Systems, Vol. 15, No. 9, Sep. 2004.

[2] Alexandru Andrei, et al. "Overhead-Conscious Voltage Selection for Dynamic and Leakage Energy Reduction of Time-Constrained Systems," in Proc. of DATE, Feb. 2004.

[3] http://developer.intel.com/design/intelxscale/2004

[4] D. Shin and J. Kim, "A Profile-based Energy-Efficient Intra-task Voltage Scheduling Algorithm for Hard Real-Time Applications," in Proc. of ISLPED, 2001.

[5] J. Luo and N. K. Jha, "Battery-aware static scheduling for distributed real-time embedded systems," in Proc. of Design Automation Conferences, pp.444-449, June 2001.

[6] B. Mochocki, et al. "Practical On-line DVS Scheduling for Fixed-Priority Real-Time Systems," in Proc. of IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2005), March 2005.

[7] B. Walsh, et al. "Parametric Intra-Task Dynamic Voltage Scheduling," in Proc. of Workshop on Compilers and Operating Systems for Low Power, 2003.

[8] L. Benni, A. Bogliolo, G. D. Micheli, "A Survey of Design Techniques for System-Level Dynamic Power Management," IEEE Trans. on VLSI, Vol.8, No.3, pp.299-316, Jun. 2000.

[9] P. Pillai and Kang G. Shin, "Real-Time dynamic voltage scaling for Low-Power embedded operating systems," In Proc. of the 18th ACM Symp. on Operating Systems Principles (SOSP-01), Oct. 2001.

[10] Eui-Young Chung, et. al, "Dynamic Power Management for Nonstationary Service Requests," IEEE Trans. on Computers, Vol.51, No.11, pp.1345-1361, Nov. 2002.

[11] Massoud Pedram, "Power Minimization in IC Design: Principles and Applications," ACM Transactions on Design Automation of Electronic Systems, Vol.1, No.1 pp. 3-56, 1996.

[12] Transmeta Corp. "www.transmeta.com/crusoe"

[13] D. Li, P. Chou, and N. Bagherzadeh, "Mode Selecdtion and Mode-Dependency Modeling for Power-Aware Embedded Systems," in Proc. of ASP-DAC/VLSI Design, 2002, Jan 2002.

[14] Dongkun Shin, Jihong Kim, and Seongsoo Lee, "Intra-task Voltage Scheduling for Low-Energy Hard Real-time Applications," in IEEE Design & Test of Computers, vol. 18, Issue 2, Mar/Apr. 2001.

[15] Y. Zhang, X. Hu and D. Chen, "Task Scheduling and Voltage Selection for Energy Minimization," in Proc. of Design Automation Conference, June 2002.



정 기 석

1989 서울대학교 컴퓨터공학과(학사)
1998 University of Illinois at Urbana-Champaign 전산학 박사
1997~1998:University of Illinois at Urbana-Champaign, 강의전담교수
1998~2000 Synopsys, Inc. Sr. R&D Engineer
2000~2001 Intel Corp. Staff Engineer
2001~2004 홍익대학교 컴퓨터공학과 조교수
2004~현재 한양대학교 정보통신대학 조교수
관심분야 : 임베디드 시스템, System-on- Chip 설계, 저전력 시스템 설계, VLSI/CAD
E-mail : kchung@hanyang.ac.kr

황 영 시

2003 대진대학교 컴퓨터공학과(학사)
2003~2005 (주)한국정보통신 연구원
2005~현재 한양대학교 정보통신학과 석사과정
관심분야 : 임베디드 시스템, 저전력 시스템
E-mail : ysturtle@hotmail.com