실험 시간 프로파일을 이용한 저전력 경성 실시간 프로그램용 동적 전압 조절 알고리즘

(A Dynamic Voltage Scaling Algorithm for Low-Energy Hard Real-Time Applications using Execution Time Profile)

신동균† 김지홍‡

(Dongkun Shin) (Jihong Kim)

요약 테스코브내부에서 공급 전압을 조절하는 테스코브 전압 스케일링(IntraVS)은 저전력 프로그램을 구현하는 데 효과적인 방법이다. 본 논문에서는 실험 실시간 운영프로그램에서 평균 실험 시간에 대한 정 보를 이용하여 전력 소모를 효과적으로 줄이는 새로운 테스코브 전압 스케일링 알고리즘을 제시한다. 최적 실험 시간을 사용하여 전압 조절의 결정을 내렸던 기존의 테스코브 전압 스케일링은 달리, 제안된 알고 리즘은 평균 실행 시간에 바탕을 두고 실행 속도를 조절함으로써 주어진 시간 제약 조건을 만족시키면서 도 기존 방법보다 에너지 효율성을 높일 수 있다. MPEG-4 디코더를 이용한 실험 결과, 제안된 알고리즘 은 기존의 테스코브 전압 스케일링에 비해 전력 소모를 최대 34% 감소시켰다.

키워드 : 동적 전압 조절, 저전력, 실시간

Abstract Intra-task voltage scheduling (IntraVS), which adjusts the supply voltage within an individual task boundary, is an effective technique for developing low-power applications.

In this paper, we propose a novel intra-task voltage scheduling algorithm for hard real-time applications based on average-case execution time. Unlike the conventional IntraVS algorithm where voltage scaling decisions are based on the worst-case execution cycles, the proposed algorithm improves the energy efficiency by controlling the execution speed based on average-case execution cycles while meeting the real-time constraints. The experimental results using an MPEG-4 decoder program show that the proposed algorithm reduces the energy consumption by up to 34% over conventional IntraVS algorithm.

Key words : dynamic voltage scaling, low-power, real-time

1. 서론

정보통신 기술이 발달함에 따라 휴대전화와 개인용 휴 대 단말기(PDA), 그리고 동영상 휴대전화와 같은 이동 음 시스템(mobile system)의 중요성은 낭로 증가하고 있다. 한편 배터리를 충전했을 때 얼마나 오랫동안 사용 할 수 있는지를 나타내는 인속 동작 가능 시간은 상업적 인 이동음 시스템에 있어서 가장 중요한 성능 지표의 하나

이동음 시스템의 전력 소모를 줄이는 것은 VLSI 시스템 설계에서 중요한 요소로 부각되고 있다. 특히, VLSI 시스템의 고성능, 고장밀화 되어있다고 전력 소모는 지속적으로 증가하는 반면에 배터리의 전력 용량은 상습적인 증가에 극히 쉽고 있어 배터리 자체 개량보다는 VLSI 시스템의 전력 소모를 줄이는 저전력 기법이 점점적으로 연구되는 주제이다. 고성능 마이크로프로세 사와 같은 이동음 시스템에 있어서 전력 소모는 중 요한 설계 목표의 하나가 된다. 높은 전력 소모는 VLSI 내부에서 많은 열을 발생시켜 반도체의 성능 저하가 가능, 설계자는 고장을 입으키기도 하기 때문이다. 이러한 문제점 때문에 최근 들어 소프트웨어 및 하드웨 어 측면에서 다양한 저전력 기법이 요구되고 있다.

일반적인 VLSI 시스템의 전력 소모는 CMOS 회로의
동적 전력 소모(dynamic power)가 대부분이며, 이때의 전력 소모 \( P = V \cdot I \equiv C \cdot V_s \cdot \frac{dV_s}{dt} \) 로 주어진다. 여기서 \( C \Omega \)은 CMOS 회로의 부하, \( \tau \)은 시간 상수, \( V_s \)은 프로세서의 입력 입력 신호, \( V_{TH} \)은 공급 전압을 의미한다. 전력 소모 \( E \)이 \( V_{TH} \)의 제곱에 비례하기 때문에 공급 전압 \( V_{TH} \)을 낮추는 것은 전력 소모를 줄이는 데 매우 효과적인 방법이다. 그러나 공급 전압을 낮추면 클록 속도도 낮아지는데, 이를 이유로 CMOS의 회로는 제한 시각 \( T_c \)을 \( V_{TH} \)\( / \cdot (V_{TH} - V_T) \)에 비례하기 때문이다. 여기서 \( V_T \)는 임계 전압, \( g \)는 속도 포화 계수(velocity saturation index)가 나타난다.

실시간 시스템(real-time system)의 경우, 시스템의 외부 클록 속도는 모든 태스크를 주어진 마감 시간(deadline)이내에 클램 수 있는 클록 속도보다 크거나 같아야 실시간 동작을 보장할 수 있다. 이때, 각 태스크의 동작 상태를 살펴보면 마감 시간 전까지 클록 속도가 클록 속도의 최고 클록 속도를 초과해버려서 클램 수 없으며, 클록 속도가 빠른 공급 전압도 함께 클램에 전력 소모를 크게 증가할 수 있다. 이것이 동적 전압 조절(dynamic voltage scaling: DVS) 기법의 기본 개념이다. 예를 들어 그림 1(a)와 같 이, 어떤 태스크가 50MHz의 클록 속도와 5V의 공급 전압이 기반 프로세서 상에서 실행한다고 가정하자. 만약 이 태스크가 실행되다면 \( 5 \times 10^7 \) 사이클이 걸려며 바갈 시간 조건이 25msec이라면, 프로세서는 주어진 태스크를 10msec만에 클램 수 있고 태스크의 마감 시간까지 15msec의 유의 시간(idle time)을 가지게 된다. 그러므로 그림 1(b)에서와 같이 클록 속도와 공급 전압이 20MHz와 2.0V로 낮아지면 유의 시간 없이 주어진 태스크를 마감 시간에 맞추어 클램 수 있고 전력 소모는 \( 2 \times 10^7 \) 사이클이 줄어들게 된다.

### 1.1 동적 전압 조절(DVS) 기법

동적 전압 조절 기법으로 가장 중요한 부품은 실시간 시스템에서 요구되는 마감시간 제한 조건(discipline constraint)을 만족시키기 위해서 공급 전압을 어떻게 효과적으로 조절하느냐 하는 것인데, 공급 전압을 점차 수준에서 수행하려는 것에 따라서 크게 두 종류로 구분된다. 태스크간 전압 스케줄링(Inter-VS)[3,4,5]가 각각의 태스크 단위로 공급 전압을 조절하여 하나의 태스크에 부서는 동일한 공급 전압을 사용하는 반면, 태스크내 전압 스케줄링(Inner-VS)[7,8]은 태스크에서 공급 전압을 조절하며, 하나의 태스크 내에서도 공급 전압이 여러 밸유동가능하다. 최근까지는 태스크간 전압 스케줄링 기법이 주로 연구되었지만, 실제로 실시간 시스템에 적용되는 과정에서는 및 가지의 문제점을 보여주고 있다.

태스크간 전압 스케줄링에서는 OS 내의 태스크 스케줄러가 태스크의 공급 전압을 결정하기에 때문에 태스크 스케줄링을 적용하더라도 OS 자체를 수정해야 하며 각각의 태스크마다 공급 전압이 하나의 고정값으로 결정되기 때문에 단일 태스크 환경에서는 적용될 수 없다[7,8]. 소형 내장형 이동 시스템(embedded mobile system)에서는 대부분의 응용프로그램이 단일 태스크 모델로 기반을 두고 있다는 것을 고려하면 이 문제를 태스크간 전압 스케줄링 기법과 실제 내장형 시스템에 적용되는 제 한을 넘어선다. 다중 태스크 환경에서도 만약 하나의 태스크가 전체 실행 시간대에 대한 부분이 있을 경우 태스크 간 전압 스케줄링 기법으로 공급 전압을 효과적으로 낮추지 못하는 경우가 발생하기도 한다[9]. 태스크 내 전압 스케줄링은 태스크간 전압 스케줄링에 이론한 한계점을 극복하기 위해서 제안되었다. 태스크간 전압 스케줄링은 다른 실행 조건에 따른 실행 시간 변경을 인한 모든 유의 시간(idle time)을 이용하기 때문에 스케줄링 프로세서가 실행을 마쳤을 때 전체 유의 시간이 남지 않게 되어 에너지 효율성을 크게 향상시킨다. 또한 OS가 공급 전압을 조절하지 않기 때문에 기존의 OS를 수정할 필요가 없다는 장점도 가지고 있다.

### 1.2 연구의 의의

본 논문은 실시간(real-time) 응용프로그램
의 평균 실행 시간(Average Case Execution Time: ACET)에 대한 정보를 이용하여 기존의 태스크 내 장 
스케줄링의 예측 효율성을 크게 높이 새로운 알고리 
즘을 제시한다. 최악 실행 시간(Worst Case Execution 
Time: WCET)를 근거로 공급 전압을 조절하는 기존의 
태스크 내 장 스크루טור의, 태스크 내 장 스크루تور(1)과 달리, 제시하는 알고리즘은 
평균 실행 경로(Average Case Execution Path: 
ACEP)를 이용하여 실행 속도를 조절한다. 여기에서 평 
균 경로로 다스크 내에서 프로그램 코드가 수행되 
는 여러 경로중에서 예측 프로그램이 수행될 때 가장 
자주 수행되는 경로를 의미한다. 제안된 알고리즘은 프 
로그램 실행 시에 가장 신뢰성 높은 평균 실행 경로 
에서의 전압 감소에 최적화되어 있기 때문에 기존의 
알고리즘보다 전체 예측율을 좀 더 효과적으로 감소시 
킬 수 있다. 기존의 태스크 내 장 스크줄링은 정상 실 
시간 시스템의 바람직한 제어 조건을 만족시키기 위해 
서 프로그램의 남은 부분이 항상 최악 실행 경로(Worst 
Case Execution Path: WCET)로 수행된다고 가정하고 
공급 전압을 결정하지만, 제안하는 알고리즘은 프로그램 
의 남은 부분이 평균 실행 경로로 수행되므로 가정하고 
전력 소모가 최소로 되도록 공급 전압을 결정하면서도 
바람직한 제어 조건을 완벽하게 지키는 것이 가장 큰 
특징이다. MPEG-4 디코더 프로그램을 사용한 실험 결 
과, 제안된 알고리즘은 기존의 태스크 내 장 스크줄링과 
비교하여 전력 소모를 최대 34% 감소시킬 수 있었다. 
본 논문은 다음과 같이 구성되어 있다. 2장에서는 기 
존의 태스크 내 장 스크줄링 기법에 대하여 설명하고, 
3장에서는 제안되는 새로운 태스크 내 장 스크줄링에 
대하여 자세히 설명한다. 4장에서는 MPEG-4 디코더 
을 이용한 실험 결과가 논의되며, 5장에서는 결론과 향후 
연구에 대한 방향을 제시한다.

2. 기존 태스크 내 장 스크줄링 알고리즘

정상 실시간 태스크에 있어서 태스크 내 장 스크줄 
링 알고리즘의 목표는 시간 제한 조건을 만족시키면서도 
전력 소모를 최소화하므로 각각의 기본 블록(basic 
block)에 적당한 클록 속도를 할당하는 것이다. 기존의 
태스크 내 장 스크줄링에 대해서와 상황보다 앞서, 본 논문에서는 태스크가 수행될 하드웨어 플랫폼인 기반 
장 프로세서(variable voltage processor)에 대해서 
다음과 같은 가장을 한다. (1) 기본 장 프로세서는 
change_f_v(x, f, x)라는 특별한 없이 제안한 알고리즘을 이용함으로써, 이 알고리즘은 프로그램의 클록 속도 f_CET와 이에 따른 해당 

그림 2 예제 프로그램: (a) 예제 실시간 프로그램 P와 
(b) CFG G_P

하는 공급 전압 V_IE를 조절한다. (2) f_CET와 V_IE는 
가변 전압 프로세서가 정상적으로 동작할 수 있는 최대 
값과 최소값 내에서 임의의 연속값으로 조절될 수 있다. (3) 가변 전압 프로세서가 클록 속도와 공급 
전압을 바꾼 때, 클록과 공급 전압을 바꾼 때는 임정 
한 사이클이 소요되며, 이 사이클 동안에 가변 전압 프로 
세서는 실행을 멈춘다(1)

그림 2(a)에 나타난 것과 같이 2μsec의 바람직한 시간 
가진 정상 실시간 프로그램 P를 생각하자. 프로그램 
P의 제어 흐름 그래프(CFG) G_P는 그림 2(b)에 나타 
난다. G_P에서 각각의 노드(node) b_i는 기본 
과목을 나타내며 각 여기(edge)는 기본 방향이 '재 
호를 원호(scalar control program)를 나타낸다. 각 
노드 내부에 있는 숫자는 해당 기본 과목의 비용 b_i를 실행하 
하는 데 필요한 사이클 수인 C_IE(b_i)를 나타낸다. 
재료에서 b_i의 역에 back edge는 프로그램 P의 while 
반복문은 나타난다.

프로그램에서 while 반복문의 최대 반복 횟수가 사용 
자에 의해서 3으로 주어졌다고 가정할 때, WCET 분석 
도구를 사용하여 최악 실행 경로(WCET)를 얻을 수는 
1) 이 가정은 클록 속도와 공급 전압을 바꾸 때, 프로세서의 실행 
시간을 나타낸다. 프로세서의 만들 시간(Transact)에서 시작되는 시간(Transact) 또는 수집 
전시에서의 프로세서의 실행 시간(WCET)이 더 지연을 
기 때문에 사용자는 시간에 따라 대략 30%의 시간을 줄 수 있음을 
알고리즘의 경우는 가변 전압 프로세서의 경우와 동일하게 수용되나 그 가변 범위도 보장 
지어지는 전자장 프로세서가 제안된 것으로 가정한다. 본 논문의 
전극과 공급 전압의 테마트에 대한 모델링을 지정할 수 
제약하여 얻어진 재료를 모델을 모든 지정할 수 있다. 

2) 파일시간(deadline)과 같은 입력변수 같은 해지 조건을 가진 경 
실 시간 시스템을 가정할 때, 요구하는 시스템 제한 조건을 보장 
하기 위해서는 시스템의 최악 실행 시간(WCET)이 미리 계산되는 
기 때문에 사용자는 높은 WCET 분석 도구는 사용자가 
루프의 최대 반복 횟수 값을 결정이 간단히 입력하면 스스로 
장실 시간 프로그램에서 분석하여 최악 실행 경로(WCET) 및 
최악 실행 시간을 안전하고 정확하게 예측해낼 수는
\[ b_{worst} = (b_1, b_2, b_3, b_4, b_5, b_6, b_7) \]

given. 이 때의 실행 사이클 160 사이클의 최악 실행 사이클 수 (Worst Case Execution Cycles; WCEC)가 된다. 만약 목표 프로세스가 최대 80MHz의 클록 주파수로 동작한다면, 프로그램 \( P \)는 유니버설리티(Idle time)을 발생시키지 않으며 실행을 \( 2\mu s \)에 걸리며, 여기서 실행 시간 대신에 실행 사이클 수를 사용했다는데, 이것은 기존 전자 프로세싱에는 클록 속도가 조절될 때 따라 실행 시간은 변하지 않기 때문에이다.

d스케일 내 스케일링 방법은 서로 다른 실행 경로 사이에 실행 시간의 차이가 크다는 점에 착안하고 있다. 그림 2(b)에 있는 예제 프로그램을 보면 32개의 서로 다른 경로가 존재한다. 최악 실행 경로 \( b_{worst} \)는 160 사이클이 걸리지만 32개 중에서 8개의 실행 경로는 80 사이클이 절단되지 않는다. 만약 프로그램을 시작할 때 어떤 경로로 실행할 것이라는 것을 알 수 있다면 클록 속도를 낮추어 상당한 양의 에너지를 줄일 수 있다.

속도 조절을 위해서, 대스케일 전단 스케일링은 최악 실행 시간에 대해 정적(static) 프로그램 분석 기법을 이용하여 적응적(adaptive) 접근법을 사용한다. 기본 클록 \( b_7 \)에서의 전안 최악 실행 사이클수 (Remaining Worst Case Execution Cycles; RWCEC) \( C_{RWCEC}(b_7) \)를 \( b_7 \)로부터 출발하는 모든 실행 경로 중에서는 가장 긴 경로의 수행 시간을 수로 정의하면, WCEC 분석 도구는 각각의 기본 클록 \( b \)에 대해, 컴퓨터 시간에 \( C_{RWCEC}(b) \)를 구할 수 있다. 그림 2(b)에서 기호 [ ]는 각 기본 플록의 \( C_{RWCEC}(b) \) 값을 나타낸 것이다.

while 반복문과 관련된 기본 플록(즉, \( b_1, b_2, b_3, b_4 \))에 대해서는 while 반복문이 최대 2번 반복되기 때문에 여러 개의 \( C_{RWCEC}(b) \) 값이 나타난다.

계산된 \( C_{RWCEC}(b) \) 값을 이용하면 CFG \( G_0 \)에서 \( C_{RWCEC}(b_7) > C_{RWCEC}(b_6) > C_{RWCEC}(b_5) \)이에 \( (b_6, b_5, b_7) \)를 찾을 수 있다. 예를 들어, 그림 2(b)에서 이러한 에지로 \( (b_4, b_5), (b_4, b_6), (b_4, b_7) \)과 \( (b_5, b_8), (b_6, b_7) \)의 4개가 존재하며 그림에서 ● 기호로 표시되어 있다. 이 경우에는 에지 \( (b_7, b_8) \)가 최악 실행 경로가 아니며 따라서 클록 속도 및 공급 전압을 조절할 수 있는 가능성이 있다고 할 수 있다. 이렇게 표시된 에지들은 진단 조절 에지 (Voltage Scaling Edges; VSE)의 후보로 선택된다. 만약 에지 \( (b_4, b_7) \)가 VSE로 선택되면, 프로그램의 제어 흐름이 \( b_7 \)에서 \( b_4 \)로 갈 때 클록 속도 및 공급 전압이 바뀌는 것을 의미한다. 예를 들어, 기본 플록 \( b_1 \) 이후에 \( b_2 \)가 실행된다면 낮은 클록 속도 1/5 \( C_{RWCEC}(b_7) \)에 대한 \( C_{RWCEC}(b_6) \)의 비율로 줄어들기 때문에 클록 속도는 낮추어 진다. VSE는 B형과 L형의 두 가지 종류로 구분된다. B형 \( \text{VSE} \)는 \( r \)을 고정한 조건된 CFG 예제(예를 들어, \( (b_4, b_5), (b_4, b_6), (b_5, b_7) \))에 해당하며 L형 \( \text{VSE} \)는 CFG의 반복문 출구 부분(예제 들어, \( (b_4, b_7) \))에 해당한다.

선택된 VSE에서 마치는 새로운 클록 속도는 앞으로 남아있는 작업량이 얼마나 빠르게 줄어들었는지에 따라 결정된다. 예를 들어, 프로그램의 제어 흐름이 VSE \( (b_4, b_7) \)에 도착했을 때 클록 속도는 잔여 작업량이 \( 1 - \frac{C_{RWCEC}(b_7)}{C_{RWCEC}(b_6)} \)만큼 줄어들었기 때문에 낮추어 진다. 클록 속도 \( r \)으로 \( b_7 \)가 실행될 후에, 클록 속도는 잔여 작업량의 감소를 반영하여 \( b_7 \)에 대한 새로운 클록 속도는 \( \frac{C_{RWCEC}(b_7)}{C_{RWCEC}(b_6)} \)로 된다.

여기서, \( \frac{C_{RWCEC}(b_7)}{C_{RWCEC}(b_6)} \)를 예시 \( b_7 \rightarrow b_6 \)에 대한 속도 변경 비율 (Speed Update Ratio; SUR)이라고 하며 \( \text{SUR} \)로 나타낸다.

그림 3은 그림 2의 예제 프로그램에 대해 대스케일 전단 스케일링이 사용되었을 때와 사용되지 않았을 때의 원래 클록 속도와의 전압이 변하는 것을 비교하여 보여주고 있다. 예시 \( (b_4, b_7) \)에서 클록 속도는 80 MHz로부터 16 MHz(= 80 MHz \times \frac{30}{160})로 바꼈다. 유동상태에서 전압이 전혀 상황되지 않았다는 전압 \( E \propto C_{v} \cdot N_{v} \cdot V_{b}^{2} \)라고 가정할 때, 프로그램 실행

그림 3 대스케일 스케일링에 의한 클록 속도와 전압의 변화
이 \( p_i = (b_i, b_i, a_i, a_i, a_i, a_i) \)의 경로로 수행될 경우, 태스크 내 전압 스케줄링 알고리즘은 에너지 소모를 60% 줄일 수 있다.

위에서 본 VSE 후보들이 완벽하게 VSE로 선택되지는 않는다는 사실에 주목해야 한다. 가변 전압 프로세서에서 클록 속도 및 공급 전압을 변환시키는 대야는 어느 정도의 시간 및 전력이 추가로 소요되므로 프로그램 코드 내에도 클록 속도 및 공급 전압을 계산하고 변환시키기 위한 명령어가 추가로 삽입되기 때문에, 클록 속도와 공급 전압을 바꾸었을 때 마진시간 계약 조건이 보상되고, 줄어든 전력 소모가 오버헤드보다 더 클 때만 귀중한 VSE 후보가 완벽적으로 VSE로 선택된다.

대스크래프 전압 스케줄링은 VSE 후보 \((b_i, a_i, a_i)\)에서 프로그램의 제어 흐름이 \(b_i\)에서 \(b_i\)로 바뀔 때 전력 작용량이 줄어든 정도, 즉 여분 사이클(saved cycle)과 주어진 임계 값(threshold)보다 커야 \((b_i, a_i, a_i, a_i, a_i)\)가 완벽적으로 VSE로 선택하게 된다. 이때의 임계 값은 클록 속도 및 공급 전압을 변환시키면서 결정한 시간과 전력의 오버헤드와 프로그램 코드 크기의 증가에 대한 함수로 주어진다.

3. 프로파일을 이용한 테스크내 전압 스케줄링

3.1 동기

프로파일을 이용한 테스크내 전압 스케줄링 알고리즘은 생성하기 전에 먼저 2단계에서 산출한 기존의 테스크 내 전압 스케줄링 알고리즘을 일반화해 보았다. 테스크내 전압 스케줄링에서는 프로그램의 낮은 부분이 어떤 경로로 수행될 것이던지 예상한 만큼보다 VSE에 클록 속도를 조절하는 데, 예상한 수행 경로를 참조 실행 경로(Reference Execution Path)라고 부르기로 한다. 앞으로 자세하게 설명할만이지만 이 단계에서 충분한 실행 경로를 선택하려면 따라서 전력 소모의 감소율이 크게 차이 낼 수 있다.

일단 참조 실행 경로가 결정되면, 테스크 내 스케줄러는 프로그램의 첫 실행 경로를 간단하게 가정하고 초기 일차 전압과 클록 수동을 결정한다. 분기문 등에 의해서 실행이 예측된 참조 실행 경로를 찾지 못했다면, 클록 속도는 참조 실행 경로와 손쉽게 찾을 수 있는 실행 경로의 간락 실행 사이클 수의 차이 값만큼 조절한다. 만약 첫 실행 경로가 참조 실행 경로보다 월 수 긴 경로라면 클록 속도는 마감시간을 지키기 위해 높여야 한다. 반면에, 만약 첫 실행 경로가 참조 실행 경로보다 실행을 일찍 끝낼 수 있다면 클록 속도는 전력 소모를 줄이기 위해서 낮추어 줄 수 있다. 일반 실행 이 참조 실행 경로와 다른 경로를 취하게 되면, 베타인 기본 플랫폼에서 시작하는 새로운 참조 실행 경로가 만들어진다.

테스크래프 전압 스케줄링은 전역 프로그램 노적 기법을 이용하여, 클록 속도가 높거나 낮아지는 억류된 프로그램 위치를 찾아내기, 그리고 클록 속도를 실행 시간에(run-time) 조정하기 위해서 선택된 프로그램 위치에 전압 조절 코드를 삽입한다. 전압 조절 코드가 삽입된 후보 위치는 CFG에서 분기문이나 반복문에 해당하는 분기 위치에 해당한다.

기존의 태스크 내 전압 스케줄링의 경우에는 프로그램의 낮은 부분이 어떤 경로로 수행되더라도 마감시간 계약 조건을 지키기 위해서 주의 깊게 실행 경로(RWEP)을 참조 실행 경로로 선택하게 된다. 본 논문에서는 기존의 태스크 내 전압 스케줄링을 RWEP 스케줄러라고 부르기로 한다. RWEP 스케줄러에서는 클록 속도가 모든 VSE에서 단조적으로 감소한다. 그러나 해당하는 알고리즘에서는 RWEP 스케줄러는 다른 방법으로 참조 실행 경로를 선택하기 때문에, 어떻게 참조 실행 경로가 선택되느냐에 따라 클록 속도는 어떤 VSE에서는 올바로고 어떤 VSE에서는 나타나지 않는다. 그러므로, VSE를 Up-VSE와 Down-VSE로 구분한다. Up-VSE에서는 클록 속도가 증가하며, Down-VSE에서는 클록 속도는 감소하다. 모든 후보 VSE는 최종 VSE로 선택된다. 그렇지 않으면 요구되는 마감시간이 지켜지지 않는다.

RWEP 스케줄러는 마감시간을 지키면서 전력 소모를 줄이지만, 이 방법은 항상 가장 긴 경로가 실행되더라도 예상하지 못한 예외적인 접근법으로 볼 수 있다. 좀 더 납득할 수 있는 접근법은 참조 실행 경로로 평균 실행 경로(ACEP)를 사용하는 것이다. 평균 실행 경로는 가장 실행될 가능성이 높은 경로로 정의되며, 실행에 대한 프로파일 정보를 통해 결정된다.

WCEP 내에 ACEP를 사용하는 주요 동기는 프로그램 수행 시간이 아주 짧게 발생하는 경우에 대해서 적절함으로서 전력 소모량에서 좀 더 높은 효율을 얻기 위한 것이다. 일반적인 프로그램에서는 프로그램 실행 시간의 약 80%는 프로그램 코드 크기에서 단지 20%에 불과한 부분에서 수행되며, 이러한 코드를 웅 과패(hot path)이라고 부른다[10]. 테스크래프 전압 스케줄링의 전력 소모를 보다 효과적으로 하기 위해서는 우선적으로 웅 과패 부분에서의 전력 소모량을 줄이는 것이 우선이다. 만약 웅 과패 중에서 하나나 테스크래프 전압 스케줄링을 위한 참조 경로로 사용한다면, 속도 변경 그래프는 웅 과패에

본 논문에서는 프로그램의 Acep을 참조 실행 경로를 이용하여 테스크루 엔진에서 실행한 피드백을 사용하는 방법을 사용했다. 이 방법은 개발 예제의 실행 별도의 테스크루 엔진에서 실행한 피드백을 사용하는 방법을 사용한다. 이 방법은 개발 예제의 실행 별도의 테스크루 엔진에서 실행한 피드백을 사용하는 방법을 사용한다. 이 방법은 개발 예제의 실행 별도의 테스크루 엔진에서 실행한 피드백을 사용하는 방법을 사용한다. 이 방법은 개발 예제의 실행 별도의 테스크루 엔진에서 실행한 피드백을 사용하는 방법을 사용한다. 이 방법은 개발 예제의 실행 별도의 테스크루 엔진에서 실행한 피드백을 사용하는 방법을 사용한다.

하지만 좀 더 정확한 프로파일링을 위해서는 최근에 제시된 개선된 기법을 사용할 수도 있다. 예를 들어, 최근에 제시된 acyclic 경로에 대해서 직접 프로파일링을 하기 위해서는 오차에 대한 부분을 SPEC의 train과 ref의 각 위치를 사용한 실행 결과를 통해서 보여주고 있는데, train 데이터를 이용한 관찰된 경로의 41.8-96.3%의 경로가 ref 데이터에서도 발견된다.[12] 그리고, 이러한 공통의 경로가 전체 실행 경로의 71.9-100.0%를 차지했다. 즉, train 데이터를 이용한 프로파일링에서 ref 데이터를 사용한 경우 실행 경로 대부분 갈라지게 된다. 이것은 프로파일 기법의 오차가 크다고 느끼는 것을 보여주고 있다.


그림 4는 b로부터 출력하는 모든 경로중에서 전처리 평균 실행 시간으로 나타나는 C_bout(b) 값을 가진 RAEP를 이용한 CFG G_b^PV를 보여주고 있다. G_b^PV에서는 숫자 두 개의 분기 예제에서 실행시에 선택될 확률이 높은 것을 의미한다. 그림 4에서 최초의 참조 실행 경로는 (b_1, b_5, b_6, b_7, b_8, b_9, b_10, b_11, b_12, b_13)가 된다. 이 참조 실행 경로를 통해, C_bout(b_4)가 계산된다. 예를 들어, C_bout(b_4) = C_10(b_4) + C_bout(b_4)가 된다. RAEP 스케줄링에서는 그림에서 빨간색 Down~VSE뿐만 아니라 오로 표시된 Up~VSE도 있다. 그림 5는 RAEP 스케줄링에 의해서 이OfType 종류의 속도의 간격이 변화하는 것을 보여주고 있다. 클록 속도는 300 MHz 단 1.5 ( = 15/10)를 가진 Up~VSE(b_1, b_2)에서 14MHz에서 21MHz로 바꾼다. 그림 3(b)에 나타난 IVT와 RAEP 스케줄링의 속도 소프트웨어와 비교하여 RAEP를 이용한 테스크루 스케줄링은 55%의 발전을 더 줄일 수 있다.
RAEP 스케줄링이 RIWE 스케줄링보다 전력 소모를 감소시키는 데에는 더욱 효과적이지만, 순수하게 RAEP 만을 이용한 해결법은 총 실행 시간을 총적으로 더 낮추지 못한다. WCEP가 Acep과 WCEP가 Acep
가 실험 시할 수에서 상당히 큰 차이를 보이고 있는 경우, 프로그램의 종류의 프로세서의 최대 속도를 사용해도
만약 실험 시할 수를 만족시키지 않기 때문이다. 예를 들어, WCEP와 Acep
가 실험 시할 수에서 상당히 큰 차이를 보이는 경우
을 생각해보자. 프로그램의 실행이 WCEP를 지나가고 있는 경우, 프로그램의 종류의 프로세서의 최대 속도를 사용하지만 실험 시할 수를 만족시키지 못할 수도 있다. 이러한 문제를 해결하기 위해서
다음 절에서는 RAEC를 사용하면서도 모든 실험 경로에
대해서 마감 시간 제한 조건을 보장할 수 있는 새로운
망 방법을 설명한다.

3.2 참조 경로 수행

본 논문에서 제안하는 알고리즘은 원래의 RAEP 스케줄링이 마감시간을 지키지 못할 수 있다는 문제를 해결하기 위해 마감시간이 지키지 않을 수 있는 상황이 전체 참조 실험 경로를 수행한다. 참조 실험 경로가
마감시간 조건에 충족되지 않는 경우에만 참조 실험 경로가 진행된다. 참조 실험 경로가 새로운

\[ T_{ref} = \frac{C_{WEC}(b_{ref})}{S} \]

\[ T_{max} < T_{ref} \times C_{WEC}(b_{max}) \]

가 됨

기 때문에, 즉, \( b_{max} \)에서부터 시작되는 새로운 참조 실험 경로를 따라 프로그램이 실행할 때 마감시간 이후에
\( M = [C_{WEC}(b_{max}) - T_{ref} \times C_{WEC}(b_{max})] \)의 사이클이 추가를 시행되어야 한다. 마감시간이 지키지 않을 경우를 위해 정수만큼 \( k \)에 대해서 \( C_{WEC}(b_{max}) \)를
만들어주어야 한다. 즉, 참조 경로의 \( b_{max} \)이

\[ \text{virtual basic block} \]

에 새로운 기반 기본 블록이 추가되어 있는 경우에 마감시간이 지키지 않을 수 있는 경우가 있다. RAEP 스케줄링의

\[ \text{virtual basic block} \]

에 새로운 기반 기본 블록이 추가되어 있는 경우에

그림 6에 여러 가지 경로의 수행이 이루어지는 지

를 보여주고 있다. 원래의 CPU \( G^{CPU} \)가 주어졌을 때,

Acep가 (\( b_{max} \times b_{ref} \))가 참조 실험 경로로 이용된다. (공은 애지는 실행시간에 선택될 가능성이 다 높은 애지를

\[ 3) \text{20 cycles} \times 100 \text{MHz} \times \frac{1}{100 \text{MHz}} \]
그림 7 참조 실행 경로 수명 알고리즘

하위 7는 참조 경로 수명 과정을 의사 코드(pseudo code)로 나타낸 것이다. 주어진 CFG G에서 프로세서의 Fragments, Instruction, Instruction 인지 기반(Instruction-based) 값을 찾아서 각 VSE graph에 속도 변화를 계산한다. 만약에 속도가 프로세서의 제한 임계 값 MaxVse 보다는 높거나 하는 문제가 있는 VSE를 찾으면, 그 VSE 이후의 속도 변화가 없다고 가정하고 MissedCycles을 계산한다. 그리고 참조 실행 경로의 가장 빠른 값이 초기값으로 설정 VSE의 속도 변화 비도수계를 수행한다. 이러한 방법은 여가가 있는 VSE가 발견되지 않음을 때까지 반복한다.

3.3 온라인 속도 할당

장치에서 설정하기 위하여 원래의 테스트와 알고리즘에서 VSE의 속도 변화를 고려한 오류를 받을 때 정의된 속도 할당 방식은 오프라인(offline) 속도 할당 방식이라고 부른다. 오프라인 속도 할당 방식에서는 모든 VSE의 초기값이 최대의 VSE로 설정되지만 그 후의 속도 변화는 없기 때문에 선택되지 않은 후보 VSE에 의해 합치지 않는 사이클이 존재한다. 오프라인 방식은 속도 변화를 불과한 때 탐색되는 사이클만 이용하고 이로 인해 설명되지 않은 사이클은 이 후의 VSE에 의해 합치지 않아 유 효시간(idle time)을 발생시킨다.

이러한 유 효시간을 이용하기 위해서, 목표 시스템의 효과적인 신간 시간을 개선하고 오프라인 (on-line) 속도 할당 방식을 제안한다. 오프라인 속도 할당 방법에서는 새로운 클록 속도가 잔여 사이클 수를 증가 시킨 것으로 결정한다. 새로운 속도가 실행 시간까지 시간에 대한 정보를 사용하기 때문에 음의 한Attrs 속도 할당 방법에서는 속도 할당 비율이 필요하지 않다.

전단 변환 오래화이가 작고 선택되지 않는 VSE의 개수가 적을 때에는 오프라인 할당 방법이 온라인 할당 방법에 비해 결과가 상대적으로 많이 뒤떨어지지 않는다는. 하지만 전단 변환 오래화이가 크 때 선택되지 않은 VSE의 개수가 늘어 때문에 온라인 속도 할당 방법이 오프라인 속도 할당 방법보다 더 효율적이다.

4. 실험 결과

본 연구에서는 제안한 테스트에 대한 간단한 실험을 수행하였으며, 실험은 프로그램의 스테이지의 테스트와 알려진 프로그램의 결과가 비슷한 결과를 얻고 있는 프로그램을 확인하였다. 본 실험에서는 실제 시스템에 대한 실험을 수행하기 위해서, 한국 önüne 애플리케이션(MPEG-4) 사용하였고, 이것에 시뮬레이터를 이용하여 결과를 검증하였다. 시스템이 유 효시간에 있을 때는 DVS 시스템이나 DVS를 사용하지 않는 시스템이나 모두 전력 저감(power down) 모드로 바뀌면 전력 보증 모드에서는 전력 소모가 0이라고 가정한다. 주어진 클록 주파수에 대한 공급전압은 V_{DD} = 1/T_{ref} = (V_{DD}-V_{TH})/I_{TH}에서 비록 다음과 같은 결과를 얻는다. DVS 시스템의 경우는 2.5V, 0.5V, 1.2V으로 사용하였다. RAEP 스케일링을 위해서 MPEG-4 비디오 디코더의 셀에 들어가 본 연구에서의 실험은 셀의 결과를 원하는 결과를 얻기 위해서는 본 연구의 결과를 통해 계산되었다. 프로세스를 통해서 이러한 정보를 구축한 결과는 다음과 같다. 클록과 전압의 변경에 소요되는 시간은 일반적인 DC-DC 변환기의 속도에는 1.0V/200μs의 값으로 설정되었다.

그림 8은 다음에 설명한 slack factor를 변환시키는 정규화된 실행 속도가 어떻게 변하는지를 보여주고 있다. 다음은 우선 deadline = WCET deadline로 정의되며, 프로세스가 WCET 이후에 유 효한(idle) 상태가 되는 시간에
비율을 나타낸다. 3.2절에서 설명한 과정에 의해서 수정된 ACEP에 대한 실험 시간이 MPEG-4 디코더에 있어서 WCET보다 35%까지 작을 수 있다. 이는 프로세서가 처음 프로그램을 수행할 때 RWE의 스케줄링이 요구하는 속도보다 35% 더 느린 속도로 시작할 수 있으며, 그 결과 전략 소모의 차이가 줄 수 있다는 것을 의미한다.

그림 8은 이번 비율을 변화시키며 두 가지 테스트 내 스케줄링 알고리즘의 전역 소모를 비교한 것이다. 모든 결과는 DVS를 사용하지 않는 시스템에서 실행되는 원래 프로그램의 전략 소모값에 대해서 정규화 되어있다. 실험을 위해서 VSE 선택을 위한 임계값은 1,000사 이클을 사용했다. MPEG-4 디코더에 대해서, RAEP 스케줄링은 RWE의 스케줄링에 비해 34%까지의 전략 소모 값을 감소시킨다.

여기서 이번 비율이 0인 경우(즉, 마감시간과 WCET가 같은 경우)에도 RWE의 스케줄링과 RAEP의 스케줄링의 전략 소모 사이에 많은 차이가 있다는 것을 알 수 있다. 이것은 시작 속도는 RWE 스케줄링에서의 같은

속도로 결정되지만 RAEP를 이용하여 속도가 결정된 부분을 이용할 수 있는 경우가 프로그램 중간에 많이 있기 때문이다. 즉, 시작 제약 조건을 만족시키기 위해서 가장 빠른게 수행되어 시작 속도는 RWE을 이용한 방법과 같지지만 가장 빠른게 수행의 일부 경로는 ACEP를 이용한 속도 결정을 활용할 수 있게 된다. 여 유 비율이 증가할수록 전략 소모의 차이가 감소하는 데, 이것은 두 가지 테스트 내 전역 스케줄링 알고리즘의 모두 낮은 전략을 사용하기 때문이다. 전략 소모는 \( V_{\text{th}}^2 \)에 비례하므로 낮은 전략을 사용할수록 전략 소모에 더 작은 차이를 발생시키는 것이다.

그림 10은 VSE 선택을 위한 임계값을 라지어 가며 어떻게 전략 소모가 변하는지를 보여주고 있다. 여유 비율은 0으로 가정했다. 온라인 속도 할당법은 추가로 40 사이클의 오버헤드를 받는다. 임계값이 작을 때는 온라인과 오프라인 속도 할당법 사이에 전략 소모에 있어 큰 차이가 있으나 임계값이 커질수록 신호화되지 않는 VSE들이 많이져 전략 소모 차이가 10%까지 생기는 것을 볼 수 있다.

5. 결론

본 논문에서는 RAEP 는 대역을 바탕으로 한 새로운 테 스트 내 전역 스케줄링 알고리즘을 제시한다. 제안된 알 고리즘은 프로그램이 실행될 때 평균 실행 경로, 즉 꼭 페스가 최적 실행 경로(WCET)보다 실행될 가능성이 많이 이러한 랜덤 페스에 대해서 전략 소모를 최적화하는 것이 좋다는 사실에 착안하였다. 제안된 알고리즘의 주된 기여는 마감시간을 보장하면서도 각 실행 경로의 확률을 이용하여 전체의 테스테이션 전역 스케줄링 알고 리즘을 개선시켰다는 점이다. MPEG-4 비디오 디코더를 이용한 실험 결과에서 RAEP 스케줄링이 RWE의 스
김지홍

김용환
1994년 서울대학교 전산공학과 학사, 2000년 서울대학교 전산공학과 석사, 현재 서울대학교 전기 컴퓨터공학부 박사과정, 전산공학부, 전자공학, 전기공학, 시스템, 전자기학, 컴퓨터 과학

김지홍