영상 디코더의 제한된 버퍼를 고려한 전력 최소화 DVFS 방식

준회원 정 승 호*, 종신회원 안 희 준*°

Power-Minimizing DVFS Algorithm for a Video Decoder with Buffer Constraints

Seungho Jeong* *Associate Member*, Heejune Ahn*° *Lifelong Member*

요 약

DVFS (Dynamic Voltage and Frequency Scaling)에 기초한 저전력 기법은 배터리를 사용하는 모바일 장치에서 동작시간 향상을 위하여 매우 중요하다. 본 연구에서는 DVFS기법에 기반을 둔 영상디코더의 에너지 소비를 최소화하는 스케줄링 알고리즘을 제안 한다. 특히, 기존연구에서 간과된 디코더와 디스플레이 사이에 위치한 버퍼의 크기 제약을 모델에 포함하여 버퍼 넘침을 방지 하도록하며, 이 모델에서 수학적으로 에너지를 최소화하는 알고리즘을 제안하고 증명하였다. 실제 영상을 통한 시뮬레이션 결과 버퍼의 크기가 10 프레임정도에서 이득이 포화상태가 되며, 제안된 알고리즘이 기존의 직관적인 알고리즘들에 비하여 평균 10%정도의 전력소모 절약을 얻을 수 있음을 확인하였다.

Key Words : video decoder, DVFS, optimal theory, majorization, embedded system

ABSTRACT

Power-reduction techniques based on DVFS(Dynamic Voltage and Frequency Scaling) are crucial for lengthening operating times of battery powered mobile systems. This paper proposes an optimal DVFS scheduling algorithm for decoders with memory size limitation on display buffer, which is realistic constraints not properly touched in the previous works. Furthermore, we mathematically prove that the proposed algorithm is optimal in the limited display buffer and limited clock frequency model, and also can be used for feasibility check. Simulation results show the proposed algorithm outperformed the previous heuristic algorithms by 7% in average, and the performance of all algorithms using display buffers saturates at about 10 frame size.

I. 서 론

최근 노트북, PMP, 스마트폰 등 휴대, 모바일 장치의 사용이 급격히 증가하고 있다[1]. 반도체 집적 기술의 발달로 모바일 장치가 성능 면에서 크게 발달한 것에 반해, 일반적으로 복잡도와 시스템의 클럭 주파수의 증가에 비례하여 에너지 소모가 증가하기 때문에, 배터리의 수명은 일반 사용자가 체감으로 느낄 수 있을 정도로 짧아졌다. 그러나, 배터리의 용량의 증가 속도는 매우 느리므로, 모바일 장치의 사용시간을 늘리기 위한 연구가 매우 중요하다. 셋톱박스(Set-top box) 등과 같이 배터리를 전원으로 사용하지 않는 임

* 서울과학기술대학교 제어계측공학과 임베디드 네트워크 연구실(snghojeong@gmail.com, heejune@seoultech.ac.kr) (° : 교신저자)
논문번호 : KICS2011-06-254, 접수일자 : 2011년 6월 16일, 최종논문접수일자 : 2011년 9월 5일

베디드 시스템에서도 에너지소모는 발열로 이어져 시스템의 신뢰성이 저하되므로 효율적인 에너지 소모 기법은 중요하다[2-5].

컴퓨터 시스템에서 저전력 설계 방식은 크게 DPM(Dynamic Power Management)와 DVFS으로 구분할 수 있다[6]. DPM은 시스템의 사용하지 않는 모듈의 동작을 중지하는 방식으로 상대적으로 시간지연이 크기 때문에 영상 디코더에 적용하기는 어렵고 적용시간이 짧은 DVFS 방식을 통한 방식이 주로 사용된다. DVFS는 각각 다른 계산량이 필요한 태스크(Task)들을 각기 다른 클럭 속도와 전압으로 처리하는 방식이다.

DVFS 알고리즘은 크게 비실시간 알고리즘과 실시간 알고리즘으로 분류된다. 비실시간 알고리즘의 경우 보통 OS에서 사용도를 감시하고 CPU 클럭 주파수를 제어하는 방식을 사용하는 반면, 실시간 응용인 경우는 응용프로그램의 데드라인 시간 등으로 구성되는 QoS 정보를 적극 활용하는 방향으로 연구되고 있다. 대표적으로 Pillai 등은 실시간 시스템의 스케줄링 알고리즘인 RM (rate-monotonic) 과 EDF (early deadline first) 알고리즘을 확장하여 QoS를 만족하면서 DVFS를 통하여 전력소모를 줄이는 알고리즘을 제안하였다[2,6]. 대부분의 연구에서 영상디코더의 태스크인 한 영상화면(프레임)의 디코딩시간도 데드라인(deadline)을 만족하여야 하므로 실시간 시스템의 특성을 갖으나 디코딩 시간의 가변적인 특성과 상대적인 태스크 수행 간격이 중요하므로 영상디코더에 특화된 DVFS 알고리즘들이 제안되었다[6-13]. 관련된 대부분의 연구[7,10-13]들은 모두 한 프레임 간격 동안 프레임의 디코딩을 진행하여야하는 데드라인 조건을 가정하고 있다. 따라서 일반적으로 영상 디코더 시스템에서 사용할 수 있는 디코더와 디스플레이 사이에 버퍼를 적절히 활용하지 못했다. 반면, 연구 [8, 9]에서는 디스플레이 버퍼를 활용하면 더효과적으로 에너지를 절감할 수 있음을 보였다. 연구 [8]에서는 태스크의 데드라인까지 남은 시간과, 최악의 경우 태스크의 계산 양과 같은 간단한 통계치를 이용하여, 데드라인을 보장하는 DVFS 알고리즘을 제안하였고, Lu 등은 [9]의 결과에 제어이론을 접목시켜, 버퍼의 크기를 정해진 데드존(dead-zone)으로 유지시키는 알고리즘을 제안하여 성능의 향상을 이끌어냈다.

하지만 현재까지 발표된 연구들은 두 가지 분명한 제약이 존재한다. 첫째는 디스플레이 버퍼의 크기의 제약을 고려하지 않는다는 점이다. 그 결과 버퍼가 일정 크기로 유지되도록 노력하지만, 버퍼의 크기가 제한됐을 때 오버플로의 방지를 보장해주지 못하다. 연구 [9]에서는 최대 버퍼요구량이 시스템 변수에 포함되어 있지 못하고, 일정 버퍼이상에서 넘침을 막기 위하여 제어를 하지만 PID방식의 제어기를 사용하므로 최대 버퍼의 크기에 대한 확정적인 고려가 되어 있지 못하다. 실제로 노트북과 같은 환경은 디스플레이 버퍼가 상당히 클 수 있으나 스마트폰 PMP 및 세탑박스 등의 장치는 디스플레이버퍼에 사용되는 메모리의 크기는 수~ 수십 프레임 정도로 제한 적이다. 두 번째는 기존 알고리즘들은 직관적인 해결방법으로서 우수한 성능을 보이기는 하지만, 최적알고리즘이 아니라는 점이다. 본 논문은 디스플레이 버퍼가 제약된 조건에서 최적 DVFS 알고리즘을 제안한다. 본 연구는 Salehi등이 네트워크에 전송하는 영상 트래픽의 평활하기위한 알고리즘[14]의 모델과 DVFS 모델과의 유사성에 착안하였으며, Salehi의 알고리즘의 모델과 본 연구 모델의 차이점인 최대 클럭 주파수의 제약이 있는 경우로 이론을 확장하였다.

본 논문의 구성은 다음과 같다. 제 2장에서는 영상 디코더시스템의 수학적인 모델을 정의한다. 제 3장에서는 제안하는 전력최소화 알고리즘을 제시한다. 특히 이 알고리즘과 연구[14]의 알고리즘간의 수학적 공통점과 차이점을 설명하며, 제안된 알고리즘이 주어진 제약 조건 하에서 최소의 전력을 소모하는 클럭 주파수 운영경로를 생성한다는 것을 증명한다. 제 4장에서 기존의 대표적인 알고리즘들과 최적 알고리즘의 성능을 실제 영상 데이터를 사용하여 비교 한다. 제 5장에서는 본 논문의 결과를 실제 시스템에 적용하기위한 제약사항과 이를 위한 추후 연구에 대하여 제안한다.

Ⅱ. 시스템 모델

2.1 비디오 디코더 시스템

그림 1은 본 논문에서 사용하는 비디오 디코더 시스템의 모델을 도시한 것이다. 압축된 프레임들은 디코더 버퍼(decoder buffer)에 저장되어 있으며, 이 버퍼의 크기에는 제약이 없다고 가정한다. 실제로 디스크나 롬에 저장되어 있음을 가정한다. 디코더는 프레임을 하나씩 디코더 버퍼에서 가져와 디코딩(decoding)하며, 디코딩에 필요한 계산량은 프레임 별로 가변적이고, 디코딩이 완료된 프레임은 디스플레이 버퍼(display buffer)에 저장되며 일정 주기로 화면에 보이고 버퍼에서 제거된다. DVFS 제어기는 시스템에서 가장 많은 계산량을 필요로 하는 부분인 디코더의 클럭 속도와 전압을 조절하여 디코딩에 소요되는 시간을 조절

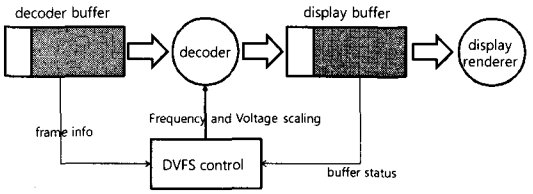


그림 1. DVFS를 적용한 비디오 디코더 모델

한다. 화면에 보이는 시간이 각 프레임 태스크의 데드라인이라 할 수 있다. 실제 시스템에서 디코더는 CPU에 의하여 구동되는 소프트웨어일 수도 있고, 동작 클럭이 변화될 수 있는 형태의 하드웨어 가속기일 수도 있다.

그림 2는 해당 시스템의 이해와 수학적 분석의 편의성을 위하여 계산량을 시간에 따른 누적 분포의 형태로 도시한다. 여기에 사용한 변수의 정의는 표 1과 같고, 해당 함수의 수학적인 표현은 다음 식 (1), (2), (3)과 같다.

$$\begin{cases} D(t)=0 & ,t<delay \\ D(t)=\sum_{i=delay}^{t} d(i-delay)u(t-i+delay) & ,t\geq delay \end{cases} \quad (1)$$

$$B(t)=\sum_{i=1}^{t+B_u} d(i)u(t-i) \quad (2)$$

$$a(t)=A(t)-A(t-1) \quad (3)$$

$$A(t)=\sum_{i=1}^{t} a(i) \quad (3a)$$

여기서 $u(t)$ 는 단위 스텝함수를 의미한다. $A(t)$

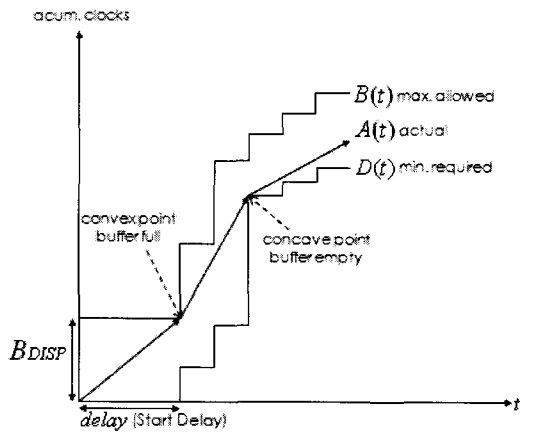


그림 2. DVFS를 적용한 비디오 디코더의 수학모델

표 1. 디코더 시스템 모델의 변수

변수	정의
N	디코딩 영상 스트림의 총 프레임 수.
t	프레임 간격 단위로 정규화된 시간 (단위 frame), $0\leq t\leq N+D$, t 정수인 경우 프레임 시점
B_u	디스플레이 (완충) 버퍼에 저장 가능한 최대 프레임 수 (단위 frame)
$delay$	디코딩 시작 이후 디스플레이되는 시간지연 (단위 frame)
$d(t)$	t 번째 프레임을 디코딩하기 위해 필요한 클럭 수
$D(t)$	t 시점까지 디스플레이버퍼가 고갈되지 않기 위한 누적 클럭 수
$a(t)$	시점 [t-1, t) 동안 실제 디코딩에 사용된 시스템의 클럭 수.
$A(t)$	시작부터 시점 t 까지의 총 디코딩에 사용되는 클럭 수.
$B(t)$	버퍼가 넘치지 않는 조건에서의 최대 가상 CPU 클럭 수.

는 실제로 디코더가 사용한 클럭 수를 의미한다. $D(t)$는 화면 시점 t에서 필요한 프레임이 디코딩되기 위해 디코더가 사용해야하는 최소의 클럭 수이다. 주목할 점은 화면출력이 일정 시간간격으로 발생하고 이때 마다 하나의 프레임이 출력되므로 일정 시간간격으로 $D(t)$는 계단 함수의 모양을 따른다. 따라서 $A(t)$가 $D(t)$와 같게 되면, 디스플레이 버퍼가 고갈된 상태(underflow)를 의미한다. 반면 $B(t)$는 시점 t에서 디스플레이 버퍼가 넘치지 않는 최대의 클럭 사용량이다. 따라서 버퍼 넘침과 고갈을 방지하기위한 조건은 $D(t)\leq A(t)\leq B(t)$이 된다. 또한 여기서 시스템 변수 $delay$와 B_u는 의미적으로 $delay\leq B_u$의 관계를 만족하여야 한다.

본 논문에서 다루는 DVFS 멀티미디어 시스템 모델은 Salehi의 전송률 연구[14]에서의 네트워크상의 디코더 버퍼 모델과 수학적으로 유사하다. 디코더의 계산량 $A(t)$는 전송속도에 대응하고, $D(t)$는 디코더에서 해당 시간까지 사용하는 압축데이터의 양, 디스플레이 버퍼 제약에 의한 $B(t)$는 디코더 버퍼의 제약과 일치한다. 그러나 두 가지 점에서 분명한 차이점이 있다. 우선, DVFS의 최대 계산량은 전송속도와 달리 속도의 제약이 있다.* 또한, 논문 [14]에서는 $B(t)-D(t)$는 디코더 버퍼의 크기의 관계를 갖고 있으나, DVFS 알고리즘에서는 단위가 clock이므로 해당 값은 가변적이다.

* 전송률에도 최대 전송률이 있을 수 있으나, 연구[5]에서는 수학적 증명이 가능한 모델을 가정하기위하여 이에 대한 제약을 주지 않았다.

2.2 제약 조건

이 시스템 모델의 성능상의 특징은, 제한된 디스플레이 버퍼를 이용하여, DVFS 제어기로 디코더의 디코딩 속도를 제어함으로써 에너지 절감을 달성하는데 있다. 일반적으로 DVFS 멀티미디어 시스템 모델은 표 2. 와 같이 5가지를 가정한다[7, 9].

가정 4의 C_k와 $d(t)$의 관계는 아래와 같다.

$$C_k = \frac{d(D+kT)}{f_{\max}} \tag{4}$$

앞서 정의한 모델에서 모든 입력 영상 스트림과 모든 알고리즘이 모두 버퍼의 고갈이나 범람이 없이 동작할 수 있는 것은 아니다. 즉, 최적의 알고리즘에 앞서, 실현 가능한 조건은 다음의 두 가지 조건을 만족하여야한다.

$$\text{(실현조건-1)}\, D(t) \le A(t) \le B(t) \text{ for } \forall t \tag{5}$$

$$\text{(실현조건-2)} \max[a(1), a(2), \ldots, a(N)] \le a_{\max} \tag{6}$$

여기서, $a_{\max}$는 디코더 시스템에서 허용되는 단위 시간 (프레임)당 최대 수행 클럭 수를 의미한다. 첫 번째 제약 조건은 디코딩과 버퍼제약에 의한 조건으로 연구 [14]에서의 실현조건과 동일하며, (실현조건-2)는 최대속도 제약에 의한 조건으로 DVFS 시스템에서 저자들에 의해 추가된 조건이다. 중요한 사항은 기존의 연구와 달리 조건 식 (6)로 인하여 모든 입력 스트림은 실현 가능한 스케줄링 경로가 존재하지는 않을 수 있다는 점이다. 예를 들어 현재 시스템의 성능을 초과하는 프레임들이 연속으로 입력되는 경우는 데드라인 안에 복호화가 불가능할 것이다. 반면, 지나치게 빠른 속도로 디코딩을 하는 경우는 버퍼, 즉 메모리가 넘치게 될 것이다.

표 2. 멀티미디어 시스템 모델의 가정

가정1) 디코더 버퍼는 고갈(Underflow) 없이 항상 디코딩이 가능한 상태로 파일 형태를 가정한다.
가정2) 각 프레임은 화면에 일정한 시간 간격 T마다 출력되어야 한다. 즉, 디스플레이 버퍼에서 일정한 시간 간격 T마다 프레임이 하나씩 소비되며, T는 동영상의 프레임율 (fps)의 역수이다.
가정3) 디스플레이 버퍼의 크기에는 제한이 있다.
가정4) 최대 CPU 클럭 속도 $f_{\max}$로 k번째 프레임을 디코딩 했을 때의 디코딩 시간이 C_k라면, $f_k = r \cdot f_{\max}$인 CPU 클럭 속도로 디코딩한 k번째 프레임의 디코딩 시간은 C_k/r이다. (클럭 보존규칙)
가정5) DVFS를 위해 변환 가능한 CPU 클럭 속도는 (0, fmax]의 범위를 갖는 연속적은 수이다.

2.3 에너지 소모 모델

일반적으로 CMOS 회로를 사용하는 디지털 시스템에서 소모되는 누설전류에 의한 영향을 제외하면, 동작전력은 식 (7) 에 의하여 모델링 된다.

$$P_{dyn} = \alpha C_{eff} V^2 f_{clk} \tag{7}$$

따라서 시스템이 요구량에 따라서 클럭 주파수를 줄일 수 있으면 그 양에 비례해서 전력사용을 줄일 수 있다. 뿐만 아니라, 식 (8) 와 같이 시스템 전압 V에 따라 스위칭 신호의 전달 속도가 비례하기 때문에, 클럭 주파수는 전압과 비례하도록 되어 있다. 따라서 클럭 주파수를 낮추면 이에 따라 동작전압도 낮출 수가 있으므로, 이를 종합하면 제곱 또는 세제곱에 비례하는 전력감축을 할 수 있다[2-4].

$$\tau \propto \frac{1}{f_{clk}} \propto \frac{V}{(V - V_T)^2} \tag{8}$$

DVFS 스케줄링을 통해 디코더로 동작하는 CPU 클럭 속도는 CPU의 최대 클럭 속도에 비해 실제 속도의 비율을 나타내는 주파수 스케일링 계수(frequency scaling factor) $r(0 \le r \le 1)$을 정의한다. 따라서 C_k는 최대 클럭 속도로 디코딩 했을 때 소모되는 시간이고, 프레임 k의 실제 디코딩 시간은 C_k/r 이다. 그리고 각 r은 최적의 전압으로 설정되어 있다고 가정한다. 그리고 일반적으로 에너지 소모량은 식 (7)과 (8)에 따라서 클럭 속도의 제곱 또는 세제곱에 비례하나 기존 연구[9]에서는 제곱에 비례하는 것으로 사용하므로 다음 식 (9)와 같이, 변수 r로 에너지 소모량을 정의한다.

$$E(r) = r^2 E_0 \tag{9}$$

여기서, E_0는 최대 클럭 속도로 동작할 때의 에너지 소모량이며, $E(r)$은 실제 설정된 클럭 속도로 동작 할 때의 에너지 소모량이다. 장치에 실제로 DVFS 알고리즘을 사용하여 주파수를 클럭을 증가시키는 경우 해당 전압이 안정적인 상태에 도달할 때에 필요한 50~100usec 정도의 오버헤드가 발생한다. 하지만 이

때 발생하는 오버헤드는 프레임 디코딩 시간의 1% 미만이므로 이전 연구들 [7,10-13]에서는 무시할 수 있는 것으로 가정한다. 본 논문에서도 이전 연구들과 같이 오버헤드는 고려하지 않는다.

Ⅲ. 전력 최소화 디코딩 알고리즘

3.1 알고리즘

2장에서 소개한 바와 같이 DVFS 멀티미디어 시스템은 기본적으로 연구 [14]의 트래픽 평활화 모델과 유사하다. 그리고 최소화 목표인 에너지와 스케줄링 대상인 클럭 주파수의 관계가 식 (9)와 같이 선형적이지 않기 때문에, 가능하다면 하나의 클럭 주파수를 유지하면서 모든 프레임을 디코딩하는 것이 최소의 에너지를 사용하는 것이 될 것이다. 그러나 일반적으로 $D(t) \le A(t) \le B(t)$를 만족하는 직선은 존재하지 않으며, 비실용적으로 큰 B와 D를 사용하는 경우에만 가능하다. 따라서 가능한 많은 프레임을 클럭 주파수 변동 없이 디코딩 하고, 클럭 속도에 변화를 줄 때는 그 변화량을 최소화하도록 하는 것이 에너지를 최소화 하는 방법일 것이라 추론해 볼 수 있다. 이러한 방법을 알고리즘으로 정리하면, 다음 변경 클럭 속도가 현재보다 높은 경우 $A(t) = B(t)$를 만족하는 조건에서 변경하며(그림 2에서 표시된 convex 지점), 다음 변경 클럭 주파수가 현재보다 낮아지는 경우 $A(t) = D(t)$이 되도록 설정하는 것이다. 이렇게 가장 변화량이 없이 실현 가능한 스케줄링을 하는 알고리즘이 [14]에서 제안한 트래픽 평활화 알고리즘이며, 개념적으로 유사한 DVFS 멀티미디어 시스템의 모델에도 적용이 가능하고 또 최적임을 다음절에서 증명한다.

변화된 수식과 논문의 자체 완결성을 위하여 다음 그림 3에 알고리즘의 의사코드를 정리하였다. 그리고 모델 변수는 표 1에서 정의로 대치한 것과 동일하다. 여기서 추가로 사용된 변수들은 다음 식 (10a), (10b), (11a), (11b)로 정의된다.

$$c_{\max} = \min_{a+1 \le t \le b} \left\{ \frac{B(t) - D(a) - q}{t - a} \right\} \quad (10a)$$

$$c_{\min} = \max_{a+1 \le t \le b} \left\{ \frac{D(t) - D(a) - q}{t - a} \right\} \quad (10b)$$

$$t_B = \max_{a+1 \le t \le b} \left\{ t : \frac{B(t) - D(a) - q}{t - a} = c_{\max} \right\} \quad (11a)$$

```
PROCEDURE OPTIMAL_PATH_FIND(d(t),b)
  t_s = 0, t_e = 1, q = 0
  c_max = b, t_B = 1, c_min = d(1), t_D = 1
  REPEAT
    t_e' = t_e + 1
    IF  c_max < (D(t_e') - D(t_s) - q)/(t_e' - t_s),  end segment at t_B:
        OUTPUT segment < t_B - t_s, c_max >
        Start a new segment at t_B:
            t_s = t_B, t_e = t_B + 1, q = B(t_B) - D(t_B)
    ELSE  IF  c_min < (B(t_e') - D(t_s) - q)/(t_e' - t_s),
OR t_e' = N end segment at t_D:
        OUTPUT segment < t_D - t_s, c_min >
        Start a new segment at t_D:
            t_s = t_D, t_e = t_D + 1, q = 0
    ELSE
        SET t_e = t_e'
    END IF
    Compute  c_max, c_min, t_B, t_D  over [t_s, t_e]
  UNTIL t_s = N
END PROCEDURE
```

그림 3. 최소전력알고리즘[14]

$$t_D = \max_{a+1 \le t \le b} \left\{ t : \frac{D(t) - D(a) - q}{t - a} = c_{\min} \right\} \quad (11b)$$

3.2 알고리즘의 최소 에너지 소모에 대한 증명

본 논문에서 다루는 DVFS 멀티미디어 시스템 모델은 전송률 스무딩 방안에서 다루는 네트워크 모델과 수학적으로 유사하다. 때문에 [14]에서 제안한 스무딩 알고리즘을 DVFS 스케줄링에 사용하면, 에너지 소모를 최소화하는 주파수 스케일링 계수의 집합을 찾을 수 있다. 본 절에서는 [14]에서 다루는 네트워크 모델과 DVFS 멀티미디어 시스템 모델에서 수학적으로 같은 의미를 지니는 것들을 매핑하고, DVFS 멀티미디어 시스템 모델에서도 [14]에서 제안한 알고리즘이 최적임을 증명한다. 표 1. 은 본 논문에서 사용할 표기법을 요약한 것이다. 혼란을 피하기 위해 [14]에서 사용하는 표기법을 그대로 사용하고 DVFS 멀티미디어 시스템 모델과 수학적으로 같은 의미를 지니는 것을 사상했다. 그리고 그림 3의 알고리즘을 이용하기 위해 멀티미디어 스트림의 모든 프레임 계산량을 알고 있다고 가정한다.

DVFS 멀티미디어 시스템은 (실현조건-2)가 추가됨으로써 [14]에서 다루는 모델과 달라진다. 하지만 정리 2는 두 번째 조건이 추가된 뒤에도 [14]의 스무

딩 알고리즘이 최적의 스케줄을 찾아내는 알고리즘임을 증명한다. 이 경우 majorization 이론에 따라 Schur convex 함수 인 어떤 함수에 $f(\bullet)$대해서도 해당 경로 $A^*(t)$가 어떤 다른 경로 $A(t)$에 대하여 $A^*(t) < A(t)$이면 $f(A^*(t)) \le f(A(t))$을 만족한다. 그러나 여기 한 가지 중요한 문제가 남는데, 제안된 알고리즘으로 설정된 경로는 (실현조건-2)를 만족하지 않고, (실현조건-1)을 만족하는 다른 모든 경로 중에서 (실현조건-2)를 만족하는 경로가 존재하는 가이다. 스무딩 알고리즘을 제안한 [14]에서는 경로의 최댓값을 제한하는 조건이 없기 때문에, DVFS 멀티미디어 시스템에서도 해당 알고리즘이 최적임을 증명하기 위해선, 제시된 문제가 반드시 해결되어야 한다.

우선, [14]에서 증명한 정리 1을 이용해야 하므로, 독자의 편의를 위하여 연구 [14]의 정의를 제시한다.

[정리 1] 최소전력알고리즘으로 생성한 클럭속도의 시퀀스 $S^* = [a^*(1), ..., a^*(N)]$는 (실현조건-1)을 만족하는 모든 시퀀스 $S = [a(1), ..., a(N)]$는 $S^* < S$를 만족한다.

증 명: 참고 문헌 [14]를 참고한다. 부호 $<$는 majorization을 의미한다[15].

[정리 2] (실현조건-1)과 (실현조건-2)를 만족하는 $S = [a(1), ..., a(N)]$가 존재하는 경우, 최소전력알고리즘으로 생성한 클럭 속도의 시퀀스 $S^* = [a^*(1), ..., a^*(N)]$는 (실현조건-1)과 (실현조건-2)를 만족하며, 모든 S에 대하여 $S^* < S$를 만족한다.

증 명: 위의 증명은 (실현조건-1)을 만족하는 스케줄 S들의 집합을 Ω_1라 하고, 두 가지 조건을 모두 만족하는 스케줄 S들의 집합을 Ω_2라고 할 때, $S^* \notin \Omega_2$이면, $\Omega_2 = \varnothing$임을 증명하는 것과 동일하다. 우선, 정의에서부터 Ω_2의 원소는 모두 (실현조건-1)을 만족하므로 만족하게 되므로 $\Omega_1 \supset \Omega_2$이다.

(정리2)에 의해, 집합 A에 속한 모든 $S = [a(1), ..., a(N)]$에 대하여, $\max\{a^*(1), ..., a^*(N)\} \le \max\{a(1), ..., a(N)\}$이다. 또한, 정리 2에 의하여 $\Omega_1 \neq \varnothing$이면, $S^* \in \Omega_1$이므로 (즉, 실현 조건-1을 만족), $S^* \notin B$이면, $\max[a^*(1), ..., a^*(N)] > a_{\max}$(실현조건-2를 불만족)이 된다. 따라서 $S^* \notin B$이면 모든 S에 대하여, $a_{\max} < \max\{a^*(1), ..., a^*(N)\} \le \max\{a(1), ..., a(N)\}$이다. 즉, $S^* \notin B$이면, 집합 A의 모든 S에 대하여, $a_{\max} < \max\{a(1), ..., a(N)\}$이기 때문에 (실현조건-2)를 만족하는 스케줄 S가 존재할 수 없어 $B \neq \varnothing$이다. (증명)

정리 2가 의미하는 것은 만약 집합 Ω_1의 스케줄 S^*가 (실현조건-2)를 만족하지 못한다면, 다른 어떤 집합 Ω_1의 스케줄 S들도 (실현조건-2)를 만족할 수 없으므로, 실현조건-1과 2를 모두 만족하는 스케줄 S가 존재하는 시스템은 최소전력알고리즘으로 스케줄링된 S^*가 존재하며 이때 $S^* < S$이다.

$S^* < S$, 즉 S가 S^*를 majorize 하는 경우에 Schur convex 함수 (에너지 소모모델인 $f(x) = x^2$ 포함)의 경우, $f(S^*) \le f(S)$ 즉, 일종의 부등호 관계가 유지된다[15]. 때문에 제안한 알고리즘이 최소의 에너지를 소모하는 스케줄임을 알 수 있다.

Ⅳ. 시뮬레이션 결과

본 논문에서는 알고리즘의 성능을 검증하기위하여 MATLAB 시뮬레이션을 통해 실험하였다. 시뮬레이션에 사용한 디코딩 시간은 공개소프트웨어로 널리 사용하는 ffmpeg[16]을 이용하여 리눅스 상에서 CPU 클럭을 최대로 고정한 후 측정한 것이다. 각 프레임의 계산량은 인텔 I386의 명령어인 'rdsc'(read time stamp counter)[17]를 이용하여 측정했으며, 다양한 비디오 스트림을 실험하였으며, 본 논문에는 대표적인 결과만을 제시한다. 시뮬레이션에 사용한 영상의 특성을 표 3에 제시하였다.

시뮬레이션은 각 프레임을 스케줄링된 클럭 주파수로 디코딩 할 때, 디코딩 완료 시간을 표 2. 의 가정 4)를 이용하여 구하고, 디코딩이 완료 될 때마다 디스플레이 버퍼를 1씩 증가시킨다. 그리고 일정 시간 단위로 버퍼를 1씩 감소하여 디스플레이 버퍼의 상태를 시뮬레이션한다. 그리고 식 (9)를 이용하여 일정 시간 단위로 에너지 소모량을 측정하고, 각 영상, 스케줄링 방법 별로 에너지 소모량을 합산하여 비교하였다.

시뮬레이션에 사용한 영상은 총 두 가지로 Drama 컨텐츠의 영상인 시트콤 "The Big Bang Theory"의 시즌4 에피소드20의 0:09:57 ~ 0:12:57를 캡쳐한 영상과 SF 영화 컨텐츠의 영상인 영화 "Avatar"의 2:27:15 ~ 2:30:15를 캡쳐한 영상이다 (그림 3). 두 영상을 캡쳐를 한 뒤, H.264코덱의 인코더로 널리 사용되는 x264[18]를 이용하여 Baseline 프로파일과 Main 프로파일로 다양한 Bit-rate와 해상도를 비교할 수 있도록 인코딩했다. 그리고 GOP크기는 15로 고정했으며, Main 프로파일의 경우 B프레임이 2번씩 연속되도

표 3. 시뮬레이션에 사용된 영상 특성

Media clips	영상 특성					디코딩 계산양의 통계 특성			
	Playback rate (fps)	재생시간 (min:sec)	해상도	Bit-rate (kbps)	Codec	평균 계산양 (cycle)	최고 계산양 (cycle)	최소 계산양 (cycle)	Std./Avg. 비율
SF Movie (Avatar)	23.98	3:00	HD 1080	5M	H.264 MP	43.83M	86.69M	22.88M	0.1607
Drama (The big bang theory)	23.98	3:00	SD	2.5M	H.264 BP	3.92M	13.81M	0.86M	0.3831
	23.98	3:00	SD	2.5M	H.264 MP	7.48M	40.43M	1.10M	0.8745
	23.98	3:00	SD	1M	H.264 BP	2.97M	11.54M	0.95M	0.4483
	23.98	3:00	SD	1M	H.264 MP	4.67M	24.27M	1.07M	0.7865

록 고정하여 규칙적인 GOP 패턴으로 인코딩되도록 설정했다.

본 논문에서 제안하는 알고리즘을 디스플레이 버퍼를 사용한 DVFS 연구 중에 가장 성능이 좋은 것으로 알려진 2가지 방법과 비교한다. 제안된 알고리즘이 수학적으로 최적이므로 실험 결과는 제안된 알고리듬의 성능 평가일 뿐 아니라, 기존 알고리즘의 상대적 성능을 확인하는 결과이기도 하다.

4.1 Panic factor[8]

논문 [8]에서 제안한 방식으로 QoS를 보장하면서 디스플레이 버퍼를 이용해 DVFS 스케줄링을 한다. 실시간성을 보장하기 위해 시뮬레이션에선 최악의 경우 디코딩 시간을 알고 있다고 가정한다. 각 프레임별로 식 (12)와 같이 주파수 스케일링 계수를 구해서 k번째 프레임을 디코딩 할 때, $f_{\max} \times r_k$로 CPU 클럭을 할당한다.

$$r_k = \frac{WCET}{\Delta t + bT} \tag{12}$$

식 (12)에서 WCET는 최악의 경우 디코딩 시간이며, Δt는 다음 프레임이 디스플레이 될 때까지 남은 시간이다. 그리고 b는 현재 버퍼에 저장된 프레임의 개수를 의미한다.

(a) Big-Bang Theory (b) Avatar

그림 4. 실험에 사용된 영상

4.2 Dead-zone 알고리즘[9]

논문 [9]에서 제안한 방식으로 DVFS 스케줄링에 제어이론을 이용한다. 버퍼의 설정된 영역으로 PID 제어기가 주파수 스케일링 계수를 조절하며, Panic factor를 알고리즘에 추가하여 실시간성을 보장한다. PID 제어기의 두 가지 Gain값은 실험적으로 얻은 값을 사용하며, K_p값은 0.05, K_i값은 0.0001을 설정했다. 그리고 버퍼의 데드존은 3 ~ 10, 예측에 사용할 윈도우 크기는 100을 설정하고 시뮬레이션을 했다.

$$r_c = K_p e + K_i \sum e \tag{14}$$

$$e = \begin{cases} B_h - b & \text{if } b > B_h; \\ B_l - b & \text{if } b < B; \\ 0 & \text{if } B_l \le b \le B_h. \end{cases} \tag{15}$$

$$Max(r_e + r_c, r_p) \tag{16}$$

Dead-zone 알고리즘은 식 (15)와 같이 에러를 정의하여 PID제어기를 이용한다. 그리고 식 (16)과 같이 예측된 계산 양인 r_e와 식 (13)과 같이 계산되는 r_p로 최종적인 주파수 스케일링 계수를 구한다.

시뮬레이션에서 에너지 소모량은 수식 (9)를 사용하여 계산하고 Panic factor에 표준화하여 비율로 나타낸다. 표 3은 시뮬레이션 결과 각 알고리즘의 에너지 소모량을 나타낸다. 제안된 알고리즘은 버퍼 한계를 5, 10, 15로 각각 에너지 소모량을 계산했다. 실험에 사용된 모든 영상은 약 10 프레임 정도면 에너지소모 측면에서 성능향상이 거의 없어지는 것을 확인하였다. 따라서 5~10 프레임 정도의 버퍼를 사용하는 것이 실용적인 것으로 예측된다. 시뮬레이션 결과, 제안된 알고리즘은 이전 연구 [8]에 비해 약 4 ~ 9%의 성능향상을 보였으며, [9]에 비해선 약 3 ~ 12%의 성능향상을 보였다. Dead-zone 알고리듬은 PID 계수를 결정해야하기 때문에 영상과 버퍼 크기 등에 따라 성능이 차이가 나는 것을 확인하였다. 하지만 해당 연구 결과에서 파라미터 설정에 대한 알고리즘이 명시되지

않았기 때문에 실험에는 고정된 계수를 사용했다. 그리고 버퍼의 최대 크기가 클수록 성능이 향상됐으며, 각 영상 별로 요구되는 최대 버퍼 크기가 달랐다. 이는 시스템에서 할당할 수 있는 최대한 버퍼를 할당하면 에너지소모를 절감할 수 있음을 나타낸다. 하지만 영상에서 요구하는 최대 버퍼 크기보다 할당된 버퍼가 클 경우 사용하지 않는 메모리로 낭비됨을 알 수 있다.

에너지 감소 외에도 중요한 성능 척도로 디스플레이 버퍼의 오버플로와 언더플로, 즉 실시간성 보장(deadline miss ratio)이 있다. 시뮬레이션에 사용된 알고리즘들은 모두 실시간성이 보장되는 알고리즘으로 데드라인을 넘긴 프레임은 하나도 없었다. 하지만 버퍼의 제한은 제안한 알고리즘만이 보장하고 이는 그림 5를 통해 확인할 수 있다. 제안한 알고리즘은 설정된 버퍼의 제한인 10을 지키며 디코딩 했지만, [9]의 알고리즘은 Deadzone을 8로 사용한 경우 버퍼의 제한을 보장하지 못하고 오버플로우되는 것을 보여준다. 또한, 이 경우 제안된 알고리즘보다도 많은 버퍼를 사용하면서 에너지 절감 성능은 떨어지며 Deadzone을

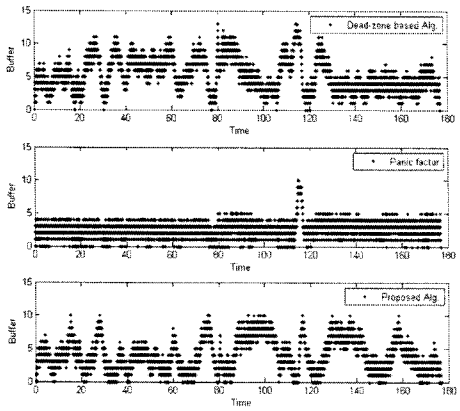


그림 5. 디스플레이 버퍼의 상태 그래프

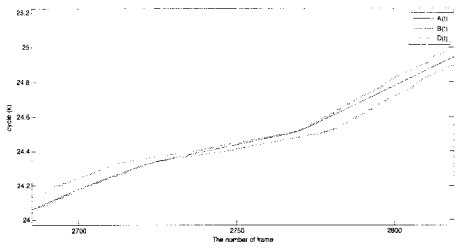


그림 6. 시뮬레이션 결과 D(t), A(t), B(t) 그래프

그 이상 감소시키면 더욱 성능에 많은 차이가 보이며, 이 경우에도 분명한 오버플로 방지를 보장 할 수 없다.

버퍼의 제한을 보장하지 못하는 것은 실제 시스템에 적용할 때 문제가 발생할 가능성이 크기 때문에 제안된 알고리즘이 이전의 알고리즘들 보다 실제적이라고 할 수 있다. 제안된 알고리듬의 실행과정의 세부예는 그림 6에 도시한 바와 같이 전체 버퍼공간을 모두 이용하는 것을 보인다. 반면 [8]에서 제안한 알고리즘의 경우, 버퍼자원을 충분히 활용하지 못하는 것을 볼 수 있다. 때문에 본 논문에서 제안하는 알고리즘에 비해서 에너지 소모량이 더 크게 스케줄링을 하는 것을 확인할 수 있다.

V. 결 론

본 논문에서는 디코더의 디스플레이 버퍼크기와 디코더 계산속도의 최대치의 제약을 고려한 DVFS 최소전력 알고리즘을 제안했다. 특히, 제안된 알고리듬이 제안한 알고리즘이 최적임을 증명하기 위해 DVFS 멀티미디어 시스템 모델을 수학적으로 정의했으며, [14]의 정리를 참조하여 제안한 알고리즘이 버퍼가 제한된 환경에서 최적임을 증명했다. 또한 시뮬레이션을 통해 이전 연구들과 제안한 알고리즘의 성능을 비교했으며, 제안한 알고리즘이 제한된 버퍼의 크기를 지키면서도 이전 연구들에 비해 에너지 소모량을 약 3

표 4. 각 알고리즘의 에너지 소모량

Media clips	영상 특성					에너지 소모량				
	Playback rate (fps)	재생시간 (min:sec)	해상도	Bit-rate (kbps)	Codec	Panic factor	Dead zone	Proposed Bu=5	Proposed Bu=10	Proposed Bu=15
SF Movie (Avatar)	23.98	3:00	HD 1080	5M	H.264 MP	1	0.9990	0.9680	0.9653	0.9639
Drama (The big bang theory)	23.98	3:00	SD	2.5M	H.264 BP	1	1.0320	0.9162	0.9118	0.9096
	23.98	3:00	SD	2.5M	H.264 MP	1	0.9939	0.9602	0.9576	0.9562
	23.98	3:00	SD	1M	H.264 BP	1	1.0297	0.9241	0.9199	0.9180
	23.98	3:00	SD	1M	H.264 MP	1	0.9687	0.9531	0.9466	0.9441

~ 12% 정도 절감하는 것을 보였다. 본 논문의 결과는 기존에 간과되었으나, 현실적으로 필요한 제약조건인 디스플레이 버퍼크기를 고려한 알고리즘을 개발하였다는 점과 해당 알고리즘이 최적화 알고리즘임을 기존은 연구와 접목하여 증명했다는 점이다. 최적알고리즘이 존재하므로 이후의 다른 직관적인 알고리즘의 성능을 평가에 객관적인 기준이 될 수 있다.

본 논문에서는 알고리즘의 수학적인 측면과 제안된 알고리즘의 성능 특성을 확인하기위하여 시뮬레이션에 의한 결과를 보였다. 현재 리눅스를 탑재한 노트북과 안드로이드 단말 장치에 구현하고, cpu 및 HW 가속장치의 클럭을 조절하여 실험을 수행하고 있다. 이때 디코더 계산에 의한 요인 뿐 아니라, 디스플레이 메모리 등의 요인에 의한 전력소모에 대한 고려에 대한 부분을 해결하여야한다. 또한 현재 각 프레임 당 계산 양을 미리 알고 있다는 가정을 하였다. 이는 동일한 플랫폼에서 사전에 실행을 한 경우에는 적용이 가능하나, 그렇지 않은 경우에는 예측에 의한 방식에 의존할 수 밖에 없다. 본 논문에서 비교대상으로 사용된 알고리듬들은 모두 예측 방식을 사용하고 있다. 그러나 본 논문에서 제안한 알고리즘을 위해서는 좀 더 정확도가 높은 알고리즘이 필요할 것으로 판단되면 이에 대한 연구를 현재 진행 하고 있다.

참 고 문 헌

[1] 노시봉, 안희준, 이명진, 오혁준, "임베디드 DSP 기반 시스템을 위한 H.264 소프트웨어 부호기의 실시간 최적화", *한국통신학회*, 10호, 2009

[2] J. Henkel, Sri Parameswaran, "Designing embedded processor", *Springer*, 2007

[3] D. Sigh, and V. Tiwari, "Power challenge in the Internet world", in Cool Chips, tutorial, in conjunction with the 32th Int. Symp. Microarchitecture, pp.8-15, Nov. 1999

[4] T.D. Burd, T. A. Perimg, A. J. Stratakos, and R.W. Brodersen, "A Dynamic Voltage Scaled Microprocessor System", *IEEE* Journal of Solid-State Circuits, Nov. 2000.

[5] K. Govil, E. Chan, and H. Wasserman, "Comparing Algorithms for Dynamic Speed-Setting of a Low Power CPU", Proc. 1st Int''l Conference on Mobile Computing and Networking, Nov. 1995

[6] O.S. Unsal and I. Koren, "System-level power-aware design techniques in real-time systems," Proceedings of *IEEE* Vol.91, No.71, pp.1055-1069, 2003

[7] W. Yuan, K. Nahrstedt, "Practical voltage scaling for mobile multimedia devices", in Proc. of ACM Multimedia'04, New York, Oct. 2004

[8] C. Im, H. Kim, and S. Ha. "Dynamic voltage scheduling technique for low-power multimedia applications using buffers", In International Symposium on Low Power Electronics and Design, pages 34-39, Aug. 2001

[9] Z. Lu, J. Lach, M. Stan, and K. Skadron, "Reducing multimedia decode power using feedback control", in Proc. Int. Conf. on Computer Design, pp. 489-796, 2003

[10] M. Mesarina and Y. Turner, "Reduced Energy Decoding of MPEG Streams", ACM/SPIE Multimedia Computing and Networking 2002 (MMCN ''02), Jan. 2002

[11] J. Pouwelse, K. Langendoen, R. Lagendijk, and H. Sips, "Power-Aware Video Decoding", Picture Coding Symposium (PCS ''01), April 2001

[12] D. Son, C. Yu, and H. Kim, "Dynamic Voltage Scaling on MPEG Decoding", International Conference of Parallel and Distributed System (ICPADS), Jun. 2001

[13] E. Nurvitadhi, B. Lee, C. Yu, and M. Kim, "A Comparative Study of Dynamic Voltage Scaling Techniques for Low-Power Video Decoding", International Conference on Embedded Systems and Applications, Jun. 2003

[14] J. D. Salehi, Z.-L. Zhang, J. Kurose, D. Towsley, "Supporting stored video: reducing rate variability and end-to-end resource requirements through optimal smoothing", IEEE/ACM trans. on networking, Vol.6, No.4, Aug. 1998

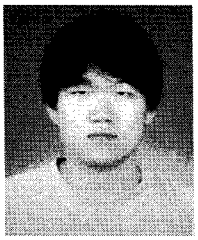
[15] A. W. Marshall and I. Olkin, Inequalities: Theory of Majorization andits Applications, New York, Academic, 1979

[16] ffmeg, online: www.ffmpeg.org.

[17] B. Case, "Intel Reveals Pentium Implementation Details", Microprocessor Report, pp.9-13, Mar. 1993

〔18〕 x264, online: http://www.videolan.org/developers/x264.html

정 승 호 (Seungho Jeong) 준회원

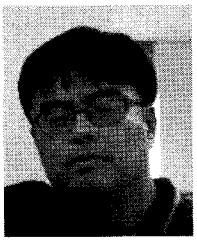


2010년 2월 서울산업대학교 컴퓨터공학과 졸업

2010년 3월~현재 서울과학기술대학교 제어계측공학과 석사과정

<관심분야> 비디오 코딩, 멀티미디어 시스템, 임베디드 시스템

안 희 준 (Heejune Ahn) 종신회원



2000년 2월 KAIST 전기및전자공학과(공학박사).

2000년~2002년 LG 전자 선임연구원.

2002년~2003년 Tmax Soft 책임연구원.

2004년~현재 서울과학기술대학교 제어계측공학과 부교수

<관심분야> 임베디드 소프트웨어, 영상통신, 인터넷 프로토콜