논문 13-38B-01-06

한국통신학회논문지 '13-01 Vol.38B No.01
http://dx.doi.org/10.7840/kics.2013.38B.1.46

영상 프레임 디코딩 복잡도 예측을 통한 DVFS 전력감소 방식

안 희 준◆°, 정 승 호*

Power-Minimizing DVFS Algorithm Using Estimation of Video Frame Decoding Complexity

Heejune Ahn◆°, Seungho Jeong*

요 약

최근 영상 디코더 시스템에서 소모전력을 절감하기 위한 방안으로 DVFS (Dynamic Voltage and Frequency Scaling) 방식을 적용한 알고리즘 들이 제안되고 있다. 이에 저자들은 논문[1]에서 전력소모를 최소화할 수 있는 스케줄링 알고리즘을 제시하였다. 이 알고리즘은 수학적으로 최적의 결과를 보장하지만, 사전에 화면 당 디코딩 계산양을 알 수 있다는 조건이 만족하여야 한다. 그러나 실제응용에서 이 조건은 만족되기 어려운 경우가 종종 존재한다. 본 논문에서는 이 제약사항을 극복하는 방안으로, 프레임의 데이터크기로 프레임의 디코딩 계산양을 예측하는 기법에 기초한 수정된 알고리즘을 제안한다. 실제 영상에서 추출된 데이터를 이용한 결과, 계산양 예측 알고리즘은 평균적으로 90%이상의 정확도를 보였으며, 따라서 계산양 예측 기법과 임계점에서의 프래임 크기 20% 내외의 완충버퍼 마진을 적용한 수정한 알고리즘은 버퍼 고갈과 넘침이 일어나지 않으며, 최적알고리즘과 비교할 때 거의 동일한 성능 (1~2% 이하의 성능저하)을 보이는 것을 확인하였다.

Key Words : video decoder, DVFS, prediction, energy consumption reduction, scheduling algorithm

ABSTRACT

Recently, intensive research has been performed for reducing video decoder energy consumption, especially based on DVFS (Dynamic Voltage and Frequency Scaling) technique. Our previous work [1] has proposed the optimal DVFS algorithm for energy reduction in video decoders. In spite of the mathematical optimality of the algorithm, the precondition of known frame decoding cycle/complexity limits its application to some realistic scenarios. This paper overcomes this limitation by frame data size-based estimation of frame decoding complexity. The proposed decoding complexity estimation method shows over 90% accuracy. And with this estimation method and buffer underflow margin of around 20% of frame size, almost same power consumption reduction performance as the optimal algorithm can be achieved.

Ⅰ. 서 론

저 전력 시스템 설계 기술은 최근 컴퓨터 시스템이 당면한 가장 중요한 문제 중의 하나이다[2]. 특히, 노트북, PMP, 스마트폰 등 휴대, 모바일 장치는 배터리를 에너지원으로 사용하고 있으며, 점차로 이러한 모

※ 본 연구는 서울과학기술대학교 2011년도 해외파견 연구교수 지원 사업의 지원을 받았음.
◆° 주저자 겸 교신저자 : 서울과학기술대학교 전기정보시스템공학과, heejune@seoultech.ac.kr, 종신회원
* (주) 윈포넷, snghojeong@gmail.com, 정회원
논문번호 : KICS2012-10-474, 접수일자 : 2012년 10월 2일, 최종논문접수일자 : 2012년 12월 25일

바일 장치에서 동영상 소비가 증가하고 있다. 따라서 영상 디코더시스템의 저전력화 기법은 실용적인 측면에서 중요성이 매우 높다. 시스템 레벨 저전력 기술의 좀 더 일반적인 소개는 참고 문헌 [1]과 [2]에 소개되어 있으며, 본 논문에서는 DVFS 방식을 영상디코더에 사용한 방법에 대해서만 다루도록 한다.

영상 디코더의 DVFS 알고리즘을 이해하기 위해서는 다음과 같은 사항의 이해가 필요하다. 영상디코더의 태스크는 한 영상화면(프레임)의 디코딩으로 정의하는데, 일정한 시간간격으로 화면에 출력되기 위하여 데드라인(deadline)을 만족하여야 하므로 실시간 시스템의 특성을 갖는다. 그러나 영상 프레임의 디코딩 시간의 가변적인 특성과 절대적 데드라인 시간이아니라 상대적인 시간간격이 중요하다. 이러한 차이점을 고려하여 영상디코더에 특화된 DVFS 알고리즘들이 제안되었다[3-9]. 기존의 대부분의 연구, [4,7-9] 등은 모두 한 프레임 간격 동안 프레임의 디코딩을 진행하여야하는 데드라인 조건을 가정하고 있다. 예를 들어 연구 [4]에서 저자들은 동영상의 화면 당 계산 양 히스토그램에 기초하여 일정 수준의 QoS를 만족하고, 한 프레임을 디코딩하면서 CPU 클럭을 조절하는 알고리즘을 제안하였다. 즉, 이 일련의 연구들은 디코더의 영상입력시점과 출력시점의 간격이 고정되어있다고 가정하였으며, 따라서 일반적으로 영상 디코더 시스템에서 사용할 수 있는 디코더와 디스플레이 사이에 버퍼를 적절히 활용하지 못했다. 연구 [4,5]에서는 이러한 디스플레이 버퍼의 사용을 활용하면 DVFS를 통해 더 많은 에너지 절감을 달성할 수 있음을 보였다. 연구[4]에서는 태스크의 데드라인까지 남은 시간과, 최악의 경우 태스크의 계산 양과 같은 간단한 통계치를 이용하여, 데드라인을 보장하는 DVFS 알고리즘을 제안하였고, Lu 등은 논문 [5]의 결과에 제어이론을 접목시켜, 버퍼의 크기를 정해진 데드존(dead-zone)으로 유지시키는 알고리즘을 제안하여 성능의 향상을 이끌어냈다.

기존 연구 결과들이 상당한 성능개선의 결과가 있었지만, 1) 발견적 (heuristic) 해결방법으로서 이론적인 최적성능의 한계를 제시하지 못하였다는 점과 디스플레이 버퍼의 크기의 제약을 고려하지 않는다는 점 등의 문제점을 갖고 있었다. 저자들은 논문[1]은 이러한 점에 주목하고, 디스플레이 버퍼가 제약된 조건에서 수학적으로 최소전력을 사용하는 DVFS 알고리즘을 제안하였다. 이 알고리즘은 수학적으로 최적의 결과를 가져오지만, 사전에 정확한 화면당 디코딩 계산양을 알고 있다는 조건을 만족하여야 하는데, 이는 디코더의 응용에 따라 만족되기 어려운 경우가 존재한다. 본 논문에서는 프레임의 데이터크기로 프레임의 디코딩 계산양을 예측하는 알고리즘을 활용하여, 이 제약사항을 극복하는 알고리즘을 제안한다. 이 아이디어의 근간에는 상당 수준이상의 정확도가 보장된 계산양 예측 알고리즘을 고안하게 된다면 이를 기반으로 최적알고리즘을 적용하였을 때 약간의 완충 버퍼를 사용하면 버퍼 고갈과 넘침이 일어나지 않고, 기존의 최적 DVFS 알고리즘의 성능을 확보할 수 있을 것이라는 점이다.

본 논문은 이와 같은 아이디어를 만족하는 구체적인 동작 알고리즘을 구성하고 실제 데이터를 사용하여 실용성을 검증하는 것이 목적이며, 이에 따라 다음과 같이 구성하였다.. 제 2장에서는 제안하는 프레임의 디코딩 복잡도 예측 알고리즘과 영상 데이터를 통한 성능을 제시한다. 3장에서는 이 예측 알고리즘을 최적알고리즘에 결합한 새로운 DVFS 알고리즘을 소개하고, 이 알고리즘을 사용한 성능을 최적 알고리즘 및 기존 대표알고리즘과 비교하여 제시한다. 제 4장에서는 이 논문의 의미와 추후 연구 방향에 대하여 제안한다.

Ⅱ. 프레임 복잡도 예측 알고리듬

2.1. 실험 데이터

우선 본 논문에서 사용한 실험데이터는 일관성을 위하여 논문[1]에서와 같은 데이터를 사용하였다. 각 영상의 프레임별 디코딩 시간은 ffmpeg[11]을 이용하여 노트북의 리눅스 상에서 CPU 클럭을 최대로 고정한 후 측정한 것이다. 각 프레임의 계산 양은 인텔 IA386의 명령어인 ‘rdsc’(read time stamp counter) [12]를 이용하여 CPU 클럭수를 이용하여 계산하였다. 표 1은 시뮬레이션에 사용한 영상의 특성을 제시하였다.

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. 두 영상을 캡쳐를 한 뒤, H.264코덱의 인코더로 널리 사용되는 x264[13]를 이용하여 Baseline 프로파일과 Main 프로파일로 다양한 Bit-rate와 해상도를 비교할 수 있도록 인코딩했다. 그리고 GOP크기는 15로 고정했으며, Main 프로파일의 경우 B프레임이 2번씩 연속되도록 고정하여 규칙적인 GOP 패턴으로 인코딩되도록 설정했다.

표 1. 시뮬레이션에 사용된 영상 특성
Table 1. Video trace statistics used for our simulations

Media clips	video characteristics					complexity charactersitics			
	Playback rate (fps)	play time (min:sec)	resultion	Bit-rate (kbps)	coding std.	mean (cycle)	max (cycle)	min (cycle)	std./avg. ratio
SF Movie (Avatar)	23.98	3:00	HD 1080	5M	H.264 MP	43.83M	86.69M	22.88M	0.1607
	23.98	3:00	HD 720	5M	H.264 MP	25.74M	74.92M	11.63M	0.2294
Drama (The big bang theory)	23.98	3:00	HD 720	5M	H.264 BP	13.00M	33.38M	3.35M	0.3138
	23.98	3:00	HD 720	5M	H.264 MP	20.04M	79.45M	4.12M	0.6043
	23.98	3:00	HD 720	2.5M	H.264 BP	10.69M	29.19M	3.38M	0.3341
	23.98	3:00	HD 720	2.5M	H.264 MP	15.61M	56.76M	4.31M	0.5027
	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

2.2. 계산 복잡도 예측 모델

그림 1는 각 영상 프레임의 크기와 계산량 (시스템 clock cycle 수)의 상관관계를 보여준다. 그래프 (a)를 통하여 압축영상의 한 프레임의 데이터의 크기와 디코딩에 필요한 계산량 간에는 높은 상관관계가 있음을 확인 할 수 있다. 프레임의 크기와 계산량이 높은 상관관계를 보이는 이유는 크게 두 가지로 생각할 수 있다. 우선 비디오 코딩 시스템의 엔트로피 디코딩부는 프레임의 사이즈가 클수록 찾아야할 심볼의 수가 많아지므로 비례 관계의 큰 이유이다. 그리고 화면간 부호화의 경우 프레임의 크기가 크면 건너띤(skipped) 마크로블럭의 숫자가 적을 확률이 높으므로 이러한 특징도 비례관계가 나타나는 원인이 된다. 프레임 크기와 계산양간의 상관관계의 연구는 [10]에 처음 발표되었으며, 이 연구에서는 동일한 부호화를 가정하여 상관계수가 매우 높은 것으로 보고되었다.

본 연구에서는 보다 정밀도가 높은 예측이 필요하기 때문에 프레임의 타입으로 구분하고, 각 타입별 선형모델로 예측했을 때 어떻게 예측이 되는 지를 실험하였다(그림 1의 (b), (c), (d)). I 프레임의 경우, 다른 타입의 프레임들에 비해 큰 크기와 계산양으로 넓게 분포하는 것을 볼 수 있다. 그리고 I화면은 엔트로피 디코딩으로 인한 계산양이 많은 부분을 차지하므로 다른 타입에 비해서 더욱 선형적인 분포를 보인다. 이는 I 프레임이 다른 프레임들에 비해 더욱 좋은 예측성능을 보이는 이유가 된다. 그리고 다른 타입의 프레임들은 I 프레임에 비해 좁게 분포하며 P 프레임에 비해 B 프레임이 더욱 작은 크기로 분포하는 것을 볼 수 있다. 여기서 주목할 것은 프레임의 타입을 나누어 분포를 그려보면, 통계적 특성이 전혀 다른 분포를 보인다는 것이다. 이는 프레임의 타입정보를 예측에 이용했을 때, 크기 정보만을 이용 했을 때보다 예측 성능이 더욱 좋아지는 이유가 된다.

따라서 본 논문에서 제안하는 DVFS 알고리즘의 계산양 예측 시스템은 각 부호화 모드를 고려하여 선별된 선형 예측 모델을 적용하여 다음과 같이 선형 예측 모델로 구성하였다.

$$\widehat{d_M}(n) = \alpha_M s_M(n) + \beta_M \tag{1}$$

여기서 모델 파라메터는 다음과 같이 구해진다.

$$\alpha_M = \frac{Cor[s_M(n), d_M(n)]}{VAR[s_M(n)]} \tag{2}$$

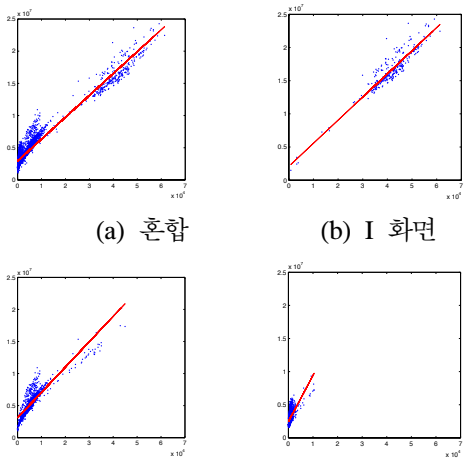
$$\beta_M = E[d_M(n)] - \alpha_M E[s_M(n)] \tag{3}$$

확률변수 $s_M(n)$는 모드 M = I , P, B 화면 또는 T (화면 통합)에서 n-번째 프레임의 크기를 나타내고, $\widehat{d_M}(n)$는 해당 프레임의 디코딩 계산양을 나타낸다. 각 확률변수의 평균과 분산을 구하기 위한 통계데이터는 시간적으로 현재 프레임에 윈도우 크기만큼 앞선 프레임들의 통계데이터를 이용한다.

표 3은 예측된 프레임 계산양의 예측오차 (디코딩 클럭수의 표준편차/디코딩 클럭수 평균)를 정리한 것이다. 모든 경우에 대해서 타입을 구분해 예측하는 것이 적게는 5%에서 많게는 두 배 더 좋은 성능을 보인다.

그리고 해상도와 데이터양이 높을수록 타입을 구분했을 때 성능이 더욱 좋아지며, I 프레임 > P 프레임

> B 프레임 순서로 예측이 더 잘되는 것을 알 수 있다. 표 2의 결과를 통해 타입을 구분해서 예측하는 것이 구분하지 않는 것에 비해서 매우 좋은 예측 성능을 보이는 것을 알 수 있다.



(a) 혼합 (b) I 화면

(c) P 화면 (d) B 화면

그림 1. 화면 타입별 데이터양과 계산 디코딩 복잡도 상관관계 그래프
Fig. 1. The correlation between frame data size and decoding clock cycle counts

2.3. 슬라이딩 윈도우기반 모델 파라메터 추정

일반적으로 선형예측 모델의 파라메터는 전체 통계 데이터를 이용해서 계수를 구하고 구해진 계수를 이용해서 예측을 하게 되지만, 영상 시퀀스는 데이터의 통계 특성이 시간에 따라 변화하는 비정상적(non-stationary)한 특성을 보인다. 이러한 통계특성 변화의 직관적인 원인은 영상 데이터의 통계적 특성은 일반적으로 영상의 장면(scene)에 크게 의존적인 것으로 알려져 있다. 때문에 전체 데이터를 이용한 예측보다는, 통계 특성이 비슷한 동일 장면으로 데이터들로 나누어서 예측에 이용하는 것이 더 좋은 예측 성능을 보여줄 수 있다. 하지만 장면을 나누려면 장면전환을 결정하는 알고리듬이 필요하다. 데이터양을 기초로 한 장면전환 검출 알고리즘은 기존에 발표된 것이 있으나 충분히 안정적이지 못하고 계산양도 추가로 필요하다. 따라서 본 연구에서는 장면전환을 검출하는 알고리즘을 사용하는 대신 슬라이딩 윈도우를 사용하여 시간적으로 가까운 화면들의 정보만을 예측에 사용하는 방법을 선택하였다. 슬라이딩 윈도우를 이용하는 장점은 예측 알고리즘의 계산 양이 줄어든다는 장점도 있다.

여기서 슬라이딩 윈도우의 크기는 다수의 데이터들을 바탕으로 실험적으로 MSE(Mean Square Error)가 가장 작은 크기로 고정하여 식 (2)와 (3)의 계산에 적용하였다. 표 3는 다양한 영상 시퀀스에서 예측을 통해 발생하는 가장 작은 슬라이딩 윈도우의 크기를 정리한 것이다. 타입 별로 보면 최적 윈도우 크기가 I 프레임 < P 프레임 < B 프레임 순서로 작은데, 이는 각 타입의 출현 빈도 때문이다. 빈도가 높을수록 같은 장면에서 많은 프레임이 존재하고 때문에 윈도우의 크기가 큰 것이 적절해진다. 그리고 해상도가 작을수록, 데이터양이 클수록 최적 윈도우 크기가 큰 것을 관찰할 수 있다. 이 실험을 통해 구한 윈도우 크기는 DVFS 스케줄링 실험에서 그대로 사용하여 계산양 예측으로 인해 발생하는 오차를 최소화했다.

표 2. 예측된 프레임 계산양의 정확도
Table 2. the accuracy of estimation of decoding cycle from the frame data size

Media clips	video statistics		est. error (std(e)/mean) %		I-type pic. est. error (std(e)/mean) %		P-type pic. est. error (std(e)/mean) %		B-type pic. est. error (std(e)/mean) %	
	resol.	Bit-rate (kbps)	combined	picture-dep.	combined	picture-dep.	combined	picture-dep.	combined	picture-dep.
Drama BP	HD 720	5M	14.94	12.99	5.27	3.81	21.07	15.23	0.00	0.00
	HD 720	2.5M	11.68	10.50	5.21	4.69	20.83	18.76	0.00	0.00
	SD	2.5M	16.13	14.65	6.57	5.63	26.26	22.53	0.00	0.00
	SD	1M	15.43	13.88	7.90	7.24	31.59	28.96	0.00	0.00
Drama MP	HD 720	5M	11.01	8.93	4.00	1.68	16.00	6.73	22.81	18.90
	HD 720	2.5M	12.97	9.86	5.46	3.53	21.86	14.12	21.05	16.33
	SD	2.5M	13.10	12.40	6.58	6.23	26.33	24.94	26.06	23.73
	SD	1M	18.30	15.88	9.12	8.26	36.47	33.03	31.03	26.66
SF Movie MP	HD 1080	5M	10.68	7.43	5.09	1.95	20.36	7.78	23.21	16.37
	HD 720	5M	8.82	6.45	4.14	2.42	16.56	9.69	20.72	14.67

표 3. 최적 예측 슬라이딩 윈도우 크기
Table 3. estimation window size selection

Media clips	video statistics		부호화 타입 통합		frame type					
					I type frame		P type frame		B type frame	
	resol.	Bit-rate (kbps)	(std(e) /mean) %	opt. win size	(std(e)/ mean) %	opt. win size	(std(e)/ mean) %	opt. win size	(std(e)/ mean) %	opt. win size
Drama Baseline profile	HD 720	5M	10.55	15	10.99	9	43.95	317	0.00	
	HD 720	2.5M	10.77	15	11.83	20	47.30	317	0.00	
	SD	2.5M	12.63	15	13.78	27	55.12	317	0.00	
	SD	1M	12.51	15	14.97	30	59.89	317	0.00	
Drama Main profile	HD 720	5M	10.87	15	16.06	11	64.26	96	16.91	161
	HD 720	2.5M	13.05	15	14.23	26	56.93	110	16.66	155
	SD	2.5M	13.66	46	19.86	71	79.44	122	21.60	161
	SD	1M	18.70	75	19.49	52	77.97	310	26.23	112
SF Movie Main profile	HD 1080	5M	9.40	15	7.78	16	31.14	81	18.37	147
	HD 720	5M	8.61	15	9.43	29	37.72	27	16.67	27

Ⅲ. 예측기법을 결합한 최적 알고리즘

3.1. 예측 기법을 적용한 최적 DVFS 알고리즘

예측 알고리즘을 사용하여 누적 프레임 복잡도 $D(t) = D(t_{current}) + \sum_{t=t_{current}+1}^{t} \hat{d}(i)$ 를 예측하게 되면 이를 이용하여 논문 [1]의 최적 DVFS 알고리듬을 적용할 수 있다. 그러나, 실제 프레임 디코딩 계산양은 예측 값과의 차이가 발생하며 이 경우에 대한 대책이 필요하다. 특히, 다른 경우보다 버퍼의 최대치나 최소치와 접하는 지점, 즉 convex 및 concave 지점에서 버퍼고갈이나 버퍼 범람를 방지하기 위한 수정, 즉 완충 버퍼영역 설정이 필요하게 된다.

본 연구에서는 응용에서 정한 확률적 허용오차를 정하면 이 완충영역의 양, 즉 마진을 선형 예측모델을 사용하여 계산하여 제공한다. 예측 에러를 가우시안분포 모델로 가정한 경우의 통계특성과 예측된 계산양의 오차 정도가 표 3에 정리되어 있다. 표 5을 보면 95% 신뢰 수준으로 버퍼고갈확률을 만족하기 위해선 약 11%에서 25% 정도 예측된 계산양에 마진을 주는 것이 적합하다. 이를 반영하여 본 논문에서 제안한 최적 알고리즘에 예측된 계산양을 10%나 20%정도 여유를 두고 경로를 스케쥴링함으로써 예측으로 인해 발생하는 버퍼고갈을 방지한다. 단 실제로 버퍼고갈이 발생활률은 스케쥴링에서 컨벡스 포인트이며 따라서 컨벡스 포인트의 빈도와 범위를 벋어날 확률 둘을 곱한 확률으로 본 연구의 경우 통상 1% 미만이 된다.

기존 알고리즘과 차이를 보이는 컨벡스 및 컨케이브 지점에서의 동작을 그림 2에 별도로 예를 들어 설명하였다. 계산양 예측 시스템을 적용한 DVFS 알고리즘은 그림 3에 제시했다. 그림 3의 알고리즘은 기존의 최적 알고리즘과 기본적인 구성은 비슷하지만 버퍼와 마진의 상태를 고려한 점이 추가되어 있다. 그리고 최적 알고리즘과 달리 컨벡스 지점까지만 경로를 찾고 디코더를 동작시킨 후 스케줄링된 경로까지 동작을 하고나면, 다음의 경로를 마진과 버퍼 점유상태를 고려해서 다시 스케줄링 한다. 그리고 버퍼의 언더플로가 발생하면 다시 알고리즘을 호출해서 재 스케줄링을 함으로써 예측으로 인한 QoS 저하를 최대한 방지한다. 표 4에 알고리즘에서 사용되는 변수들이 정리하였다.

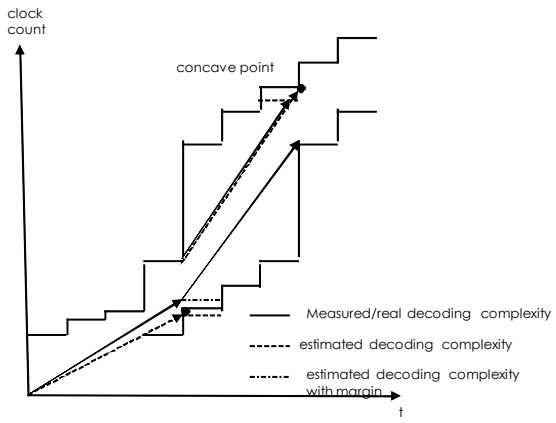


그림 2. 예측오류에 대응하기위한 컨벡스, 컨케이브 지점에서의 마진을 이용한 알고리즘의 수정
Fig. 2. the illustration of the modified algorithm at convex and concave points for overcoming the estimation error

3.2. 실험 결과

본 논문에서 제안하는 알고리즘을 논문[1]의 최적 알고리즘 및 대표적인 영상 DVFS 알고리즘 2가지와 비

교한다. 최적알고리즘은 [1]을 참고로 하고, 나머지 두 알고리듬에 대하여만 추가로 설명한다.

표 4. 계산양 예측 시스템 모델의 변수
Table 4. the variables for estimation-based, modified optimal DVFS algorithm

변수	정의
$d_{est}(t)$	식 (12)를 이용해서 예측된 t번째 프레임을 디코딩하기 위해 필요한 클럭 수.
$D_{est}(t)$	t 시점까지 디스플레이 버퍼가 고갈되지 않기 위한 예측된 누적 클럭 수.
$B_{est}(t)$	버퍼가 넘치지 않는 조건에서의 예측된 최대 가상 CPU 클럭 수.
$Margin$	버퍼 공간에서의 스케줄링 마진 (단위 %)
$buf(t)$	각 시점에 버퍼로 인해 생기는 계산양 공간 $B_{est}(t)-D(t)$ (단위 cycle)
$D_{margin}(t)$	버퍼를 마진 이상으로 유지시키기 위한 예측된 누적 클럭 수. $D_{est}(t)+(Margin/100)\bullet buf(t)$
$buf_{occ}(t)$	각 시점에 버퍼에 누적된 프레임들의 클럭 수.

● Panic factor 알고리즘

논문[4]에서 제안한 방식으로 성능을 보장하면서 디스플레이 버퍼를 이용해 DVFS 스케줄링을 한다. 실시간성을 보장하기 위해 시뮬레이션에선 최악의 경우 디코딩 시간을 알고 있다고 가정한다. 각 프레임 별로 식 (12)와 같이 주파수 스케일링 계수를 구해서 k번째 프레임을 디코딩 할 때, $f_{\max}\times r_k$로 CPU 클럭을 할당한다.

$$r_k = \frac{WCET}{\triangle t + bT} \tag{4}$$

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

● Dead-zone 알고리즘.

논문 [5]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.

STEP 0. Set the default estimation parameters with system parameters and variables
STEP 1. Get the next frame size,
STEP 2. Estimate the complexity of the next frame to decode. (using eqn. (1))
STEP 3. Calculate optimal freq. and the corresponding voltage. (using modified algorithm below)
STEP 4. Decode one frame: measure the decoding clock count and decoding time that frame, and the result buffer occupancy.
STEP 5. Update the complexity estimation parameters. (egn (2) and (3)

PROCEDURE DYNAMIC_SMOOTHING_PATH_FIND $(t_s, d(t), b)$
  IF $buf_{occ}(t) < D_{margin}(t) - D_{est}(t)$, apply margin:
    $D_{tmp}(t) = D_{margin}(t) - D_{est}(t)$
  END
  $t_e = t_s, q = 0$
  $c_{\max} = b, t_B = 1, c_{\min} = d(1), t_D = 1$
  REPEAT
    $t_e' = t_e + 1$
    IF $c_{\max} < (D_{tmp}(t_e') - D_{tmp}(t_s) - q)/(t_e' - t_s)$, end segment at t_B:
      OUTPUT segment $< t_B - t_s, c_{\max} >$
      Start a new segment at t_B:
        $t_s = t_B, t_e = t_B + 1, q = B(t_B) - D_{tmp}(t_B)$
    ELSE IF $c_{\min} < (B(t_e') - D_{tmp}(t_s) - q)/(t_e' - t_s)$, OR $t_e' = N$ end segment at t_D:
      OUTPUT segment $< t_D - t_s, c_{\min} >$
      BREAK
    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. 예측 기법을 적용한 최적 DVFS 설정 알고리즘
Fig. 3. the estimation based, modified optimal DVFS scheduling algorithm

$$r_c = K_p e + K_i \sum e \tag{5}$$

$$e = \begin{cases} B_h - b & \text{if } b > B_h; \\ B_l - b & \text{if } b < B; \\ 0 & \text{if } B_l \le b \le B_h. \end{cases} \tag{6}$$

$$Max(r_e + r_c, r_p) \tag{7}$$

Dead-zone 알고리즘은 식 (6)와 같이 에러를 정의하여 PID제어기를 이용한다. 그리고 식 (6)과 같이 예측된 계산 양인 r_e와 식 (5)과 같이 계산되는 r_p로 최종적인 주파수 스케일링 계수를 구한다.

성능의 비교는 에너지 절감 효과와 버퍼문제 발생

확률을 살펴보았다. 특히, 버퍼 범람의 조건에서는 디코더를 잠시 중지할 수 있으므로 버퍼고갈 확률만을 고려하였다. 표 5은 제안된 알고리즘과 다른 알고리즘의 에너지 소모량을 비교한 표이다. 모든 에너지 소모량은 최적 알고리즘을 기준으로 정규화 하였다. 모든 경우에 대해서 제안한 알고리즘이 비교대상 알고리즘인 Dead-zone 알고리즘보다 좋은 성능을 보인다. 그리고 버퍼범람과 고갈확률을 낮춤에 따라, 즉, 마진을 증가함에 따라, 적용함에 따라 에너지 성능이 조금씩 감소하는 것을 볼 때, 직관적인 해석과도 일치하는 결과이다. 데드존 알고리즘 보다 좋은 결과를 보이는 것은, 최적 알고리즘의 사용과 함께, 예측 알고리즘이 가져온 일종의 피드포워드 특성 때문인 것으로 분석 된다. 참고로 데드존 알고리즘은 매우 간단한 예측 기법과 간단한 피드백 알고리즘으로 구성되어 있다.

표 6은 계산양 예측 시스템을 이용했을 때의 성능(버퍼 범람 및 고갈 횟수)을 제시한다. 마진의 적용양이 많아짐에 따라 성능은 좋아지는 것을 통해 직관적인 해석과 일치하는 결과를 볼 수 있다. 그 결과 마진을 20% 이상으로하게 되면 버퍼 범람은 실제적으로 거의 발생하지 않는 것을 확인할 수 있다. 또한 버퍼이상이 생기는 경우를 관찰하여보면 언더플로가 발생하는 프레임은 밀집된 형태로 나타나는 것을 관찰할 수 있다. 이는 계산양 예측 시스템에서 사용하는 영상의 통계특성이 변하는 부분에서 발생하는 것으로 추정되는데, 프레임의 크기와 계산양 간의 관계에 대한 통계특성이 급격히 변하는 구간이 존재하기 때문이다. 따라서 장면의 전환과 같은 영상의 통계 특성이 변하는 구간을 감지해서 이를 예측에 반영하는 노력을 함으로써 문제의 해결이 가능할 것으로 생각된다.

Ⅳ. 결 론

본 논문에서 저자들은 논문[1]에서 제안하고 증명한 최소 전력소모 DVFS 스케쥴링 알고리즘을 화면당 계산 복잡도를 알지 못하는 (실시간) 상황으로 확장하기 위한 방안과 알고리즘을 제시하였다. 프레임당 디코딩양과 계산 복잡도가 상관도가 높다는 사실을 이용하였다. 특히 화면별 예측 모델을 통하여 예측의 정확도를 향상시켰으며, 화면별 예측모델은 평균 90%~95%이상의 높은 정확성을 보이는 것을 확인하였다. 계산양을 예측하기 위하여 해당 프레임의 데이터 이 모델을 바탕으로 수정된 알고리즘은 기존 최적알고리즘에 예측모델 부분을 확장한 형태로 적용가능하며, 확률적 예측에서 발생하는 오차에 따른 버퍼고갈의 문제는 10%에서 20%정도의 마진을 적용하여 해결할 수 있음을 보였다.

표 5. 계산양 예측 시스템을 적용한 알고리즘의 에너지 성능
Table 5. Energy consumption performance in comparison with the optimal DVFS algorithm

genre	resolution	Bitrate (bps)	optimal	Dead-zone	Proposed 0% Margin	Proposed 10% Margin	Proposed 20% Margin
SF	1080	5M	1	1.1249	1.0142	1.0165	1.0173
	720	5M	1	1.0898	1.0144	1.0150	1.0166
Drama	720	5M	1	1.0894	1.0116	1.0136	1.0154
	720	2.5M	1	1.0846	1.0112	1.0107	1.0112
	VGA	2.5M	1	1.09508	1.0144	1.0181	1.0198
	VGA	1M	1	1.0782	1.0162	1.0270	1.0271

표 6. 계산양 예측 시스템을 적용한 알고리즘의 버퍼고갈 확률
Table 6. Buffer underflow frequency for all test durations

genre	resolution	Bitrate (bps)	Panic factor	Dead-zone	Proposed 0% Margin	Proposed 10% Margin	Proposed 20% Margin
SF	1080	5M	3	0	36	14	1
	720	5M	1	0	21	6	0
Drama	720	5M	5	0	12	3	0
	720	2.5M	2	0	12	4	0
	VGA	2.5M	2	0	19	5	0
	VGA	1M	2	0	2	0	0

그 결과 최적 알고리즘의 성능을 유지하면서 실시간 알고리즘으로 활용이 가능하게 되었다. 제안된 수정 알고리즘이 기존의 대표적 알고리즘보다 좋은 성능을 보이는 것은, 이 알고리즘의 근간이 되는 최적 알고리즘의 특성과 정교한 디코딩 계산양 예측기법에 기인한다고 판단된다.

마지막으로, 예측 오차로 인하여 간헐적으로 발생할 수 있는 버퍼 고갈의 문제는 주로 장면전환시 발생하는 특성 변화로 생기는 문제로 추정되며, 이는 적합한 장면전환 검출 기법과 접목하면 좀 더 높은 신뢰성을 확보할 수 있을 것으로 예상한다. 또한, 본 알고리즘을 CPU나 DSP기반 이외의 하드웨어로 구성된 디코더에 적용하기위해서는 각 요소별 계산양 예측기법이 추가로 개발 되어야 할 것이다.

감사의 글

이 연구는 서울과학기술대학교 2011학년도 해외파견 연구교수 지원 사업의 지원으로 수행되었습니다.

참 고 문 헌

[1] S. Jeong, and H. Ahn, “Power-minimizing DVFS algorithm for a video decoder with buffer constraints,” *J. KICS*, vol. 36, no. 9, pp. 1082-1091, Sep. 2011.

[2] O. S. Unsal and I. Koren, “System-level power-aware design techniques in real-time systems,” *Proc. of IEEE* vol. 91, no. 71, pp. 1055-1069, Jul. 2003.

[3] W. Yuan, and K. Nahrstedt, “Practical voltage scaling for mobile multimedia devices,” in *Proc. ACM Multimedia’04*, pp. 924-931, New York, Oct. 2004.

[4] C. Im, H. Kim, and S. Ha. “Dynamic voltage scheduling technique for low-power multimedia applications using buffers,” in *Proc. Int. Symp. on Low Power Elec. and Design*, pp. 34-39, Aug. 2001.

[5] 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, Oct., 2003.

[6] M. Mesarina and Y. Turner, “Reduced energy decoding of MPEG streams,” in *proc. ACM/SPIE Multimedia Comp. and Netw, 2002 (MMCN’02)*, pp. 202-213, Jan. 2002.

[7] J. Pouwelse, K. Langendoen, R. Lagendijk, and H. Sips, “Power-aware video decoding,” in *proc. Picture Coding Symposium (PCS ’01)*, pp. 303-306, Apr. 2001.

[8] D. Son, C. Yu, and H. Kim, “Dynamic voltage scaling on MPEG decoding,” in *proc. Int‘l Conf. of Parallel and Dist. Sys.*, pp. 633-640, Jun. 2001.

[9] E. Nurvitadhi, B. Lee, C. Yu, and M. Kim, “A comparative study of dynamic voltage scaling techniques for low-power video decoding,” in *proc. Int. Conf. Embed. Syst. and App.*, pp. 23-26, Jun. 2003.

[10] A. C. Bavier, A. B. Montz, Larry and L. Peterson, “Predicting MPEG execution times,” in *proc. SIGMETRICS’98/PERFORMANCE ’98 pp.131-140, Jun.* 1998.

[11] ffmeg, from www.ffmpeg.org.

[12] B. Case, ‘‘Intel reveals pentium implementation details,” *Microprocessor Report*, pp. 9-13, Mar. 29, 1993.

[13] x264, from http://www.videolan.org/ developers/ x264.html

안 희 준 (Heejune Ahn)

2000년 2월 KAIST 전기및전자공학과 (공학박사).
2000년~2002년 LG전자
2002년~2003년 Tmax Soft
2004년~현재 서울과학기술대학교 전기정보공학과 (교수)
<관심분야> 임베디드 SW

정 승 호 (Seungho Jeong)

2010년 2월 서울산업대학교 컴퓨터공학과 졸업
2012년 2월 서울과학기술대학교 제어계측공학과 졸업(공학석사)
2012년 3월~현재 (주)윈포넷 연구원
<관심분야> 멀티미디어 시스템, 임베디드 시스템