지연 시간 및 화질 제약이 있는 비디오 응용을 위한 에너지 최적화 기법

(An Energy Optimization Technique for Latency and Quality Constrained Video Applications)

임 채 석 † 하 순 회 ††
(Chaeseok Im) (Soonhoi Ha)

요 약 이 논문은 지연 시간 및 화질 제약이 있는 비디오 응용을 위한 에너지 최적화 기법을 제안한다. 이는 두 가지 핵심 기법 - 프레임 생략 기법 및 버퍼링 기법 - 으로 구성되어있다. 버퍼링은 운영 체제 수준에서 유휴 시간 이용률을 증가시키고, 프레임 생략은 응용 수준에서 유휴 시간 자체를 증가시키며, 양쪽 모두 동적 전압 조절 기법의 효과를 향상시킨다. 이 논문에서는 제안한 기법을 적용하기 위해 H.263 부호기 응용을 사용한다. 실험에서는 제안한 기법이 주어진 지연 시간 및 화질 제약을 만족하면서 괄목할 만한 에너지 절감을 얻을 수 있음을 보인다.

키워드 : 에너지 최적화 기법, 지연 시간 제약, 화질 제약, 비디오 응용, 동적 전압 조절, 프레임 생략, 버퍼링, H.263 부호기

Abstract This paper proposes an energy optimization technique for latency and quality constrained video applications. It consists of two key techniques: frame-skipping technique and buffering technique. While buffering increases the slack time utilization at the OS level, frame skipping increases the slack time itself at the application level, and both enhance the effectiveness of the dynamic voltage scaling technique. We use an H.263 encoder application as a test vehicle to which the proposed technique is applied. Experiments demonstrate that the proposed technique achieves noticeable energy reduction satisfying the given latency and video quality constraints.

Key words : energy optimization technique, latency constraint, video quality constraint, video application, dynamic voltage scaling, frame skipping, buffering, H.263 encoder

1. 서 론

최근 이동 환경에서 비디오 전화나 비디오 카메라 등의 비디오 응용이 중요하게 부각되고 있다. 이러한 이동 멀티미디어 응용에서 중요한 문제 중의 하나는 바로 저전력 설계인데, 시스템의 에너지원인 전지의 용량에는 한계가 있기 때문이다. 그래서 많은 저전력 설계 기법들이 회로 수준에서부터 운영 체제 수준에 이르기까지 다양한 추상화 단계 (abstraction level)에서 제안되었다[1].

동적 전압 조절 (DVS: dynamic voltage scaling)은 마이크로프로세서 기반의 시스템에서 가장 주목받는 기술 중의 하나이다[2]. 이는 태스크 실행 후 남은 유휴 시간(slack time)을 활용하여 가변 전압 프로세서(variable voltage processor)의 전압 및 동작 주파수를 낮춤으로써 시스템의 에너지 소모를 줄이는 것이다. DVS에서 핵심 사항은 유휴 시간 이용률(utilization)을 증가시켜서 주어진 성능 제약(performance constraint)내에서 공급 전압을 최대한 낮추는 것이다.

멀티미디어 응용은 일반적으로 다음과 같은 특징을 갖는다. 1) 스트림 형태의 입력 데이타에 대해 반복적으로 실행하며, 실행 시간은 가변적이다. 2) 지연 시간(latency)에 대한 고려보다 일정한 속도의 처리 능력(throughput)이 더 중요하다. 3) 사람이 인지 못하는 범위 내에서 화질 (video quality)에 대한 내성(tolerance)이 있다. 이러한 특징 중에서 특별히 지연 시간과 화질에 대한 내성을 활용하여 유휴 시간과 이의 이용률을 높일 수 있으면, 시스템의 에너지 소모를 최적화할 수

† 학생회원 : 서울대학교 전기컴퓨터공학부
csim@iris.snu.ac.kr
†† 정 회 원 : 서울대학교 컴퓨터공학부 교수
sha@iris.snu.ac.kr
논문접수 : 2003년 9월 9일
심사완료 : 2004년 7월 28일

있는데 이것이 이 논문의 주제이다. 이 논문에서는 제안한 기법을 적용하기 위해 H.263 부호기(encoder) 응용을 사용한다.

제안한 기법은 두 가지로 구성된다. 우선, H.263 부호화 과정에서 화질을 관찰해서 주어진 화질 제약(video quality constraint)의 범위 안에서 부호기 알고리즘의 프레임 속도(frame rate)를 변화시킨다. 화질은 비디오 시퀀스(video sequence)의 시간적(temporal)이고 공간적(spatial)인 자연스러움을 함께 지칭하는 표현이다. 시간적 화질, 즉 움직임의 부드러움은 비디오 시퀀스의 프레임 속도와 움직임 정도에 따라 다르다. 만약 프레임 속도가 너무 낮다면, 움직임의 끊김 현상으로 보는 사람이 불편할 것이다. 이런 문제를 없애려면, 가장 높은 움직임 정도에 맞춰 응용의 프레임 속도를 충분히 높여야 한다. 이는 프레임 속도가 낮은 움직임 정도의 화면에 대해서는 지나치게 높아져 있음을 의미한다. 따라서 이 논문에서는 화면의 움직임 정도에 따른 중복적인(redundant) 프레임 생략(frame skipping)을 통해서 프레임 속도를 조절할 것을 제안한다. 이는 화질이 크게 떨어지지 않은 한 의도적으로 유휴 시간을 생성할 수 있음을 의미한다.

두 번째 제안한 기법은 지연 시간 제약(latency constraint) 내에서 입력 데이타를 버퍼링(buffering)하는 것이다[4]. 일정한 처리 능력을 보장하기 위해서 시스템은 최악 실행 시간(WCET: worst-case execution time)을 가정하고 설계되는데, 실제 실행 시간이 그것보다 작게 되면 부하 가변 유휴 시간(VST: workload-variation slack time)이 발생한다. 하지만 입력 데이타가 도착하지 않아서 태스크를 실행할 수 없을 경우에는 이 VST를 이용할 수 없게 된다. 제안한 입력 버퍼링 기법은 대기 중인 입력 데이타의 수를 증가시켜서 VST가 발생했을 때 태스크가 바로 실행 가능하도록 만든다. 입력 버퍼링은 종종 입력 프레임의 도착 간격(inter-arrival time)이 변동하는 것을 안정화하기 위해서 이미 사용되기 때문에 제안한 버퍼링 기법으로 추가되는 자원 부가(overhead)는 그리 크지 않다.

버퍼링은 운영 체제 수준에서 유휴 시간 이용률을 증가시키고, 프레임 생략은 응용 수준에서 유휴 시간 자체를 증가시키며, 양쪽 모두 동적 전압 조절 기법의 효과를 향상시킨다. 제안한 기법은 기존 부호기 알고리즘을 조금만 수정하여 구현할 수 있다. 게다가 복호기(decoder) 쪽에서는 아무런 수정이 필요하지 않다. 결국 제안한 기법은 지연 시간 및 화질과 같은 서비스 품질(QoS: quality of service) 제약 내에서 이동 비디오 응용의 에너지 소모를 감소할 수 있는 효율적이고 효과적인 기법이다.

이 논문은 다음처럼 구성된다. 2장에서는 이전 관련 연구들을 간단하게 살펴볼 것이다. 3장에서는 버퍼링 기법에 대해서 설명하고, 개발 동기 및 제안한 프레임 생략 기법을 각각 4장과 5장에서 설명할 것이다. 6장에서 실험 결과를 보이고 7장에서 결론을 맺겠다.

2. 관련 연구

DVS 기반의 에너지 최적화 연구는 많이 있었다. 실시간 시스템을 위한 정적 혹은 동적 방식의 DVS 기법은 일반적으로 엄격한 종료 시점(deadline)을 지킬 수 있게 고안되었다. [5][6]의 연구에서는 엄격한 종료 시점 제약 하에서 최악 실행 시간을 최대한 이용하도록 하는 최적의 전압 스케줄 (voltage schedule)을 찾아내었다. 이 연구의 한계는 실행 가능한 태스크가 없는 시점에서는 VST를 사용하지 못한다는 데 있다. [4]의 연구에서는 버퍼를 사용하여 지연 시간의 제약 하에서 VST를 모두 사용하는 버퍼링 기법을 제안하였다.

기존의 기법들이 멀티미디어 응용에 적용 가능하다고 하더라도, QoS의 여러 인자를 동시에 고려한다면 좀더 복잡한 최적화 문제가 발생한다. 최근 QoS 제약을 고려한 몇몇 에너지 최적화 기법들이 소개되었다. 예를 들면, [3]의 연구에서 저자는 JPEG 이미지 압축 알고리즘에 대해서 지연 시간 및 이미지 품질(image quality) 제약을 고려한 에너지 최적화 기법을 제안하였다. 이들은 JPEG 알고리즘의 몇몇 인자(parameter)들과 지연 시간 및 이미지 품질과 같은 QoS 요소들 간의 관계를 나타내는 표를 구성하여서, 실행 시간(run-time)에 에너지와 QoS 요소들간에 최적의 교환(trade-off)이 이루어지도록 인자 값을 정하도록 했다. [7]의 연구에서 저자는 MPEG-2 비디오 복호기 프로그램의 핵심인 IDCT의 계산 정밀도를 변화시켜서, 주어진 주관적/객관적 화질 제약 하에서의 에너지 및 면적 최소화 기법을 제공하였다. 제안하는 방법은 알고리즘을 변형시키지 않고 QoS 제약 하에서 프레임 생략을 통한 프레임 속도 변화를 사용한다는 점에서 이들 기존 연구와 다르다.

프레임 생략은 [8]의 연구처럼 비디오 데이타를 사용 가능한 대역폭(bandwidth)에 적응시키기 위해서 잘 사용되는 기술이다. [9]의 연구에서 저자는 주어진 프레임 생략 비율(frame dropping ratio)을 활용하여 프로세서의 속도 및 에너지 소모를 감소시킨다. 하지만 여기서는 화질을 무시했으므로 이 연구에서 제안한 기법과는 차이가 있다. [10]의 연구에서는 대역폭의 제약 하에서 시간적 화질과 공간적 화질 사이의 교환을 고려하였다. 이 연구에서 저자는 화질 변화를 줄이기 위해서 이동 윈도우(sliding window)에서 이웃한 프레임 간의 움직임 정보에 근거하여 프레임 속도를 적응적으로 조절한다.

이 논문은 지연 시간과 화질 제약이 있는 비디오 응용에서, 위의 버퍼링 기법과 프레임 속도 조절 기법을 확장하여 에너지 소모를 줄이는 목적으로 사용하였다는 점에서 독창적인 내용을 담고 있다.

3. 버퍼링 기법

프레임 생략 기법은 유휴 시간을 많이 증가시켜준다. 두 번째로 제안하는 것은 입력 데이타를 버퍼링해서 지연 시간 제약 하에서 유휴 시간을 최대한 활용하는 기법이다. 이 장에서는 [4]에서 제안한 이 버퍼링 기법을 간략하게 살펴보겠다.

최악 실행 시간이 주기와 같고 실제 실행 시간이 최악 실행 시간의 반인 주기적인 태스크를 가정해보자. 입력 버퍼링이 없는 전형적인 태스크 간(inter-task) DVS 기법은 여기서 발생하는 유휴 시간을 최대한 활용하지 못한다. 왜냐하면 태스크는 자신의 방출 시간(release time) 이전에 실행될 수 없기 때문이다. 그러므로 그림 1(a)에 보인 것처럼 프로세서는 전체 시간의 반을 휴지 상태(idle state)에서 보내게 된다. 반면 그림 1(b)처럼 입력 데이타를 미리 한 번 버퍼링한다면, 다음 태스크는 그 유휴 시간을 전부 활용할 수 있게 된다. 전자의 상황에서 휴지 상태의 프로세서를 전력 차단 상태(power-down mode)로 보낸다 하더라도 후자의 경우가 에너지를 더 많이 절감하게 된다. 그림에서 화살표로 표시했듯이 버퍼링은 지연 시간을 증가시킨다. 그래서 버퍼링 기법은 버퍼링 지연 시간이 지연 시간 제약에 의해서 허용되는 한 사용될 수 있다. 자세한 논의는 [4]에서 다루고 있다.

단일 태스크의 경우에, WCET와 BCET를 각각 최악 실행 시간과 최소 실행 시간이라고 했을 때, 최소 버퍼 크기는 $\min\left(\left|\frac{WCET}{BCET}-1\right|, latency constraint\right)$가 된다. [4]의 버퍼링 기법에서는 실행 가능한 태스크의 수를 최대한 늘려서 VST를 최대한 활용하기 위해서 입력 프레임 버퍼를 완전히 채운 다음에 실행을 시작한다. 버퍼링 기법이 프레임 생략 기법 이후에 적용된다면, 프레임 간격이 더 이상 1이 아니기 때문에, 입력 프레임의 부호화 작업을 버퍼가 완전히 찰 때까지 미루지 않아도 된다. 대신 버퍼의 맨 앞에 있는 프레임에 대한 부호화 작업을 언제 시작할지를 결정하기 위해서 프레임 간격을 계산해야 한다.

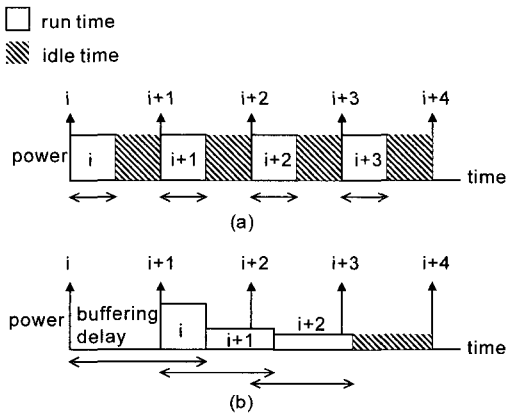

그림 1 (a) 버퍼링이 없는 DVS 기법 (b) 버퍼링 DVS 기법

단일 태스크의 경우, 버퍼링 기법의 성능을 과거의 에너지 최적화 기법들 - 태스크 간 DVS, 태스크 내 DVS - 과 비교한 결과가 그림 2에 나타나 있다. 버퍼링 기법이 다른 기법들보다 뛰어난 성능을 보이고 최적(optimal)에 가까운 결과를 보임을 알 수 있다. 다중 태스크의 경우, 버퍼링 기법을 실시간 다중 태스크 환경으로 확장한 연구 결과를 [11]에 발표하였다. 여기서도 제안한 버퍼링 기법은 기존의 다중 태스크 DVS 기법들에 비해 좋은 성능을 보인다.

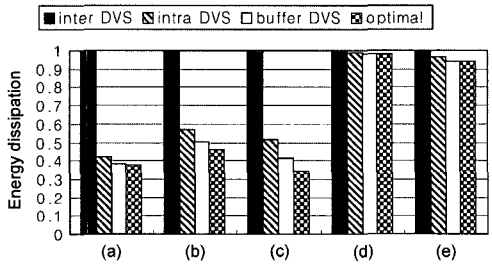

그림 2 단일 태스크에서 버퍼링 기법의 성능 비교 (a) MPEG-2 복호기 (b) MPEG-4 부호기 (c) MPEG-4 복호기 (d) VSELP 부호기 (e) VSELP 복호기

4. 프레임 생략의 동기

일반적으로 비디오 응용은 최악의 화면 변화에도 화질을 만족시키기 위해 고정 프레임 속도 - 이 논문에서는 30FPS(frames per second) - 로 동작한다. 정량적인 분석을 위해 이 논문에서는 화질에 대한 객관적인 척도로 동작의 부드러움(motion smoothness)을 사용하는데, 식은 PSNR(peak signal noise ratio)과 비슷하다. 프레임 차이(FD: frame difference)가 이웃한 두 프레임 간의 평균 제곱 오차(mean-square error)일 때, 화질은 아래 식과 같이 계산된다.

$$Quality = 10 \times \log_{10}\left(\frac{1}{FD} \times 255^2\right)[dB] \quad (1)$$

그림 3은 여러 비디오 시퀀스에 대해서 부호화 프레임 속도에 따른 평균 화질을 보여준다. 이 실험에서는 프레임을 주기적인 방법으로 생략해서 프레임 속도를 감소시킨다. 예를 들어 30/4 FPS의 프레임 속도는 한 프레임 부호화 후 다음 세 프레임을 생략함으로써 얻을 수 있다. 모든 비디오 시퀀스에 대해서 프레임 속도가 감소함에 따라 화질이 선형적으로 감소함은 주목할 만한 일이다. 비디오 시퀀스마다 움직임 정도가 다르기 때문에 프레임 속도에 따른 화질과 그 변동은 모두 다르게 나타난다. 예를 들어 'Coastguard'와 'Foreman'은 다른 두 비디오 시퀀스보다 움직임 정도가 훨씬 크기 때문에 모든 프레임 속도에 대해서 낮은 화질을 보여준다. 한편 'Hall monitor'와 'Container'는 움직임 정도가 작기 때문에 높은 화질을 보여준다.

그러므로 다양한 장면으로 구성된 비디오 시퀀스가 고정 프레임 속도(FFR: fixed frame rate)로 부호화되었다면, 화질은 그림 4에서처럼 요동이 있을 것이고, 따라서 상당 부분의 시간 동안 화질이 화질 제약보다 크게 형성이 될 것이다.

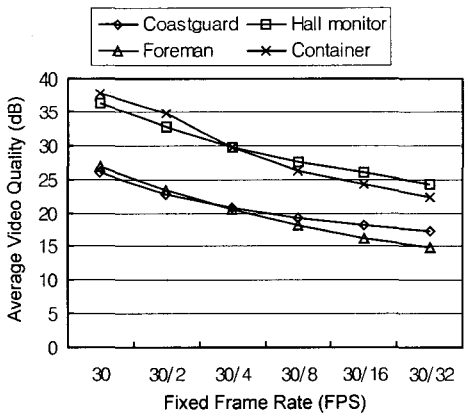

그림 3 고정 프레임 속도에 따른 화질의 변화

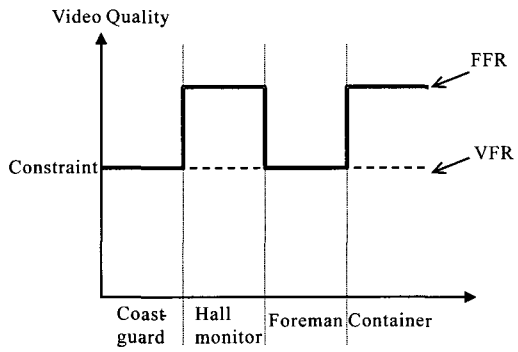

그림 4 혼합 비디오 시퀀스의 화질 변화

제안한 기법의 기본 아이디어는 부호화 과정에서 화질이 주어진 제약보다 크게 형성되면 해당 프레임을 생략하는 것이다. 이런 프레임 생략 기법은 부호화 태스크의 실질적인 프레임 속도를 감소시켜서 유휴 시간을 증가시키고 결과적으로 DVS의 도움을 받아서 에너지 소모를 줄일 수 있게 한다. 이 논문은 그림에서처럼 화질을 제약에 가깝게 좀더 균등하게 분포시키기 위해서 'Hall monitor'와 'Container'에서 좀더 많은 프레임을 생략하는 가변 프레임 속도(VFR: variable frame rate)를 사용하였다.

5. 프레임 생략 기법

이 장에서는 두 가지 프레임 생략 기법 - 직접 비교(direct comparison) 기법 및 예측(prediction) 기법 - 에 대해서 설명하겠다.

5.1 직접 비교 프레임 생략

그림 5는 간단하며 직관적인 직접 비교 프레임 생략 기법을 보여준다. 이는 개념적으로 두 개의 병렬적인 작업(process) - 프레임 생략 작업 및 부호화 작업 - 으로 나눠진다. 프레임 생략 작업은 카메라에 담긴 입력 프레임 이미지를 읽어서 현재 선택된 프레임 이미지(프레임 0)와 새롭게 입력된 프레임 이미지를 두 프레임 간의 화질이 주어진 화질 제약 또는 지연 시간 제약을 위반할 때까지 차례로 비교한다. 위반이 발생하면 (프레임 5)이 작업은 바로 이전 입력 프레임 (프레임 4)을 부호화해야 할 프레임으로 선택한다. 부호화 작업은 프레임 생략 작업에 의해서 선택된 프레임만을 부호화한다. 부호화된 이웃 두 프레임 간의 간격을 프레임 간격(frame interval)이라고 하는데, 이는 1과 최대 프레임 간격(MFI: maximum frame interval) 사이의 범위에서 동적으로 변하고, MFI는 주어진 지연 시간 제약에 의해서 한계가 주어진다.

이 방법은 미리 검증 절차를 거치기 때문에 주어진 화질 제약을 위반하지 않는다. 하지만 그렇게 하기 위해서 추가 버퍼 오버헤드가 필요하다. 그러므로 이 방법은 버퍼링 기법과 자연스럽게 결합될 수 있고, 최대 버퍼 요구량은 MFI가 된다. 이 추가적인 버퍼는 지연 시간을 증가시키는데, 최대 지연 시간은 $2\times MFI\times T$(초기 버퍼

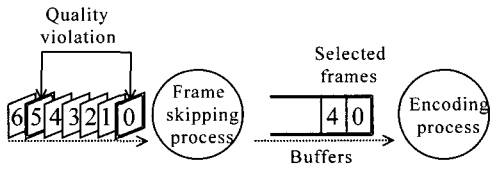

그림 5 직접 비교 프레임 생략 기법

링 $MFI \times T$, 부호화 작업 $MFI \times T$, T는 태스크의 주기)가 된다. 그러므로 LC(latency constraint)가 지연 시간 제약일 때, MFI는 다음 부등식을 만족해야 한다.

$$MFI \leq \frac{LC}{2T} \quad (2)$$

제안한 프레임 생략 기법의 알고리즘을 아래에 제시하였다.

```
void Direct_Comparison_Frame_Skipping
{
   for ( ; ; ) {
      currentImage = readImage(currentFrame);
      frameInterval = 0;
      do {
        nextFrame = currentFrame + (++frameInterval);
        nextImage = readImage(nextFrame);
        quality = PSNR(currentImage, nextImage);
      } while (quality ≥ constraint && frameInterval
      ≤ MFI);
      frameInterval--;
      boundCheck(&frameInterval);
      send_to_encode_process(currentImage, frameInterval);
      currentFrame += frameInterval;
   }
}
```

제안한 알고리즘의 에너지 오버헤드는 프레임 생략에 필요한 추가 연산에 의해서 발생하는데, 이 추가 연산은 화질을 정량화하기 위한 PSNR 계산이 대부분을 차지한다. 실험을 위해서 최적화된 H.263 부호기 코드를 100MHz ARM720T 프로세서에서 실행했을 때, 한 프레임 부호화에 소모되는 시간은 456ms~637ms(45.6 Mcycle~63.7Mcycle)인데 비해, 추가 연산의 오버헤드는 19ms(1.9Mcycle) 미만이었다. 일반적인 전압 전환(voltage switch)의 오버헤드가 0.52ms [12]이고, 작업 전환(process context switch)의 오버헤드가 1ms (0.1Mcycle)를 넘지 않는다고 가정하면, 전체 실행 시간 오버헤드는 최대 4.5%($\frac{19ms+0.52ms+1ms}{456ms}$)에 해당한다. 이는 미미한 정도의 값이어서 이 실행 시간 증가에 따른 에너지 오버헤드는 프레임 생략에 의해서 얻는 에너지 이득에 비해서 무시할 수 있을 정도로 작다고 할 수 있다.

그림 6에 직접 비교 프레임 생략 기법의 적용 예를 보였다. MFI는 4라고 가정하자. 우선 프레임 생략 작업에서 초기 4회 버퍼링을 시행하면서 부호화될 프레임을 선택한다. 여기서는 프레임 0과 1이 선택되었다고 가정한다. 그러면 부호화 작업에서 이들 프레임을 해당 프레임 간격동안 DVS를 사용하여 부호화한다. 동시에 프레임 생략 작업에서는 이어지는 프레임을 읽어서 이전에 선택되었던 프레임과 비교하는 작업을 계속 실행한다.

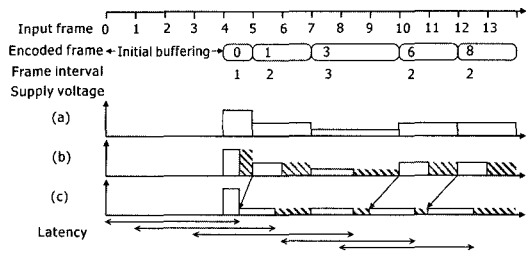

그림 6 직접 비교 프레임 생략 기법의 부호화 예 (a) WCET에서 프레임 생략 결과 (b) AET에서 프레임 생략 결과 (c) AET에서 프레임 생략과 버퍼링을 동시에 적용한 결과

버퍼링될 수 있는 최대 크기는 MFI가 된다. 그림에서는 이어서 프레임 3, 6, 8이 부호화되었다. 그림 6(a)는 모든 프레임에 대해서 최악 실행 시간이 발생했을 때 공급 전압의 변화를 나타내고 있다. 만약 실제 실행 시간(AET: actual execution time)이 WCET의 절반이라고 가정한다면, 그림 6(b)에 보인 것처럼 프로세서는 절반의 시간동안 대기 상태(idle)에 있게 된다. 하지만 그림 6(c)에 보인 것처럼 버퍼링 기법을 이용하게 되면 이 유휴 시간도 활용할 수 있게 된다. 직접 비교 방법에서는 다음에 부호화할 프레임 선택된 다음에야 현재 프레임을 부호화할 수 있다. 예를 들면, 프레임 3은 시간 7에서 프레임 6이 선택될 때까지 기다려야 한다. 이러한 제약은 버퍼링 기법을 적용할 때 반드시 고려되어야 한다. 버퍼링 기법을 이용한다면 화살표로 나타낸 것처럼 프레임 1, 6, 8은 유휴 시간을 활용할 수 있게 된다. 그러므로 그림 6(c)에 나타난 것처럼 공급 전압은 더욱 감소할 수가 있는 것이다. 그림 6은 또한 지연 시간이 초기 버퍼링 때문에 증가함을 보여준다. 최대 지연 시간은 $2 \times 4T$가 된다. 이런 긴 지연 시간은 직접 비교 방법의 단점이 된다. 또 하나의 단점은 4개의 입력 버퍼가 필요하다는 점을 들 수 있겠다.

5.2 예측 프레임 생략

직접 비교 기법이 현재 프레임의 부호화를 다음 프레임이 선택될 때까지 미루는 반면, 예측 기법은 과거에 기초하여 앞으로의 프레임 간격을 예측한다. 처음에 프레임 간격은 프레임 생략이 이루어지지 않는다는 의미에서 1로 설정한다. 이후 프레임이 도착하면 그 프레임과 이전에 재구성된(reconstructed) 프레임 사이의 프레임 차이를 계산한다. 그 차이가 화질 제약보다 작으면 프레임 간격을 증가시키고, 그 프레임 간격을 이용하여 조절된 전압과 동작 주파수로 현재 프레임에 대한 부호화를 실행한다. 여기서는 지연 시간 및 버퍼 오버헤드가 없기 때문에 MFI는 아래 식처럼 직접 비교 방법에 비

해서 두 배로 증가한다.

$$MFI \leq \frac{LC}{T} \quad (3)$$

따라서 지연 시간 제약이 엄하게 주어진 경우라면 직접 비교 방법보다 예측 방법이 나을 수 있다.

하지만 이 방법은 예측이 빗나갔을 경우 화질 제약을 위반할 수 있다. 이것은 좋은 예측 알고리즘이 필요한 이유이기도 하다. 본 논문에서는 화질 위반의 비율을 낮추기 위한 예측 알고리즘을 제시하였다. 이 알고리즘은 실험에서 1% 미만의 위반율을 보였다. 제안한 예측 기법의 알고리즘을 아래에 제시하였다.

```
void Prediction_Frame_Skipping
{
    frameInterval = fixedFrameInterval;
    for ( ; ; ) {
        currentImage = readImage(currentFrame);
        quality = PSNR(currentImage, previouslyRecon-
        structedImage);
        if (quality > constraint + α)
                frameInterval++;
        else if (constraint ≤ quality ≤ constraint + α)
                ; // preserve the current frame interval
        else if (constraint - α ≤ quality < constraint)
                frameInterval /= 2;
        else if (quality < constraint - α)
                frameInterval = fixedFrameInterval;
        boundCheck(&frameInterval);
        DVS(frameInterval);
        doEncoding(currentImage);
        skipFrames(frameInterval - 1);
        currentFrame += frameInterval;
    }
}
```

현재 프레임을 읽은 다음, 그 프레임과 이전에 재구성된 프레임(reconstructed frame) 간의 화질을 구한다. 이 화질에 근거하여 프레임 간격을 조절한다. 화질의 안정적인 변화를 위해서 임계(threshold), α를 사용한다.

Case 1. 화질 > 제약 + α: 이는 이전 프레임에서 갑작스런 화면 변화가 없었음을 의미한다. 따라서 이런 상태가 다음 프레임에서도 지속될 것이라는 가정 하에 프레임 간격을 1만큼 증가시킨다.

Case 2. 제약 ≤ 화질 ≤ 제약 + α: 이는 주어진 응용이 동작하기를 바라는 바람직한 화질 영역이다. 그러므로 프레임 간격을 조절하지 않고 이전 프레임 간격을 계속 유지한다.

Case 3. 제약 - α ≤ 화질 < 제약: 이는 이전 프레임 간격이 현재 화면의 변화를 감당하기에 너무 큼을 의미한다. 따라서 화질을 주어진 제약보다 크게 하기 위해서 프레임 간격을 줄일 필요가 있다. 빠른 적응을 위해서 이 논문에서는 프레임 간격을 반 씩 감소시킨다. 즉 프레임 속도는 2배가 된다.

Case 4. 화질 < 제약 - α: 이는 이전 프레임에서 움직임의 부드러움이 갑자기 심하게 나빠졌음을 의미한다. 이는 일반적으로 화면 전환 시에 발생한다. 이 경우에는 화질을 FFR의 그것과 같게 만들기 위해서 프레임 속도를 최대값으로 복귀한다.

제안한 알고리즘은 간단하지만 화질 제약의 비율을 낮추기 위해 매우 효과적으로 동작한다. 우선 Case 2의 경우처럼 화질이 좋은 상태에서도 α (threshold) 이상 좋아지지 않을 경우에는 현재 프레임 간격을 유지함으로써 프레임 간격의 변화가 자주 발생하지 않게 하였다. 또한 Case 1의 경우처럼 화질이 충분히 좋은 경우에도 프레임 간격을 점진적으로 증가시키고, Case 3이나 4처럼 화질이 나빠지는 경우에는 프레임 간격을 재빠르게 줄여서 빠른 화면 전환에도 화질이 많이 나빠지지 않고 전체적으로 화질이 나빠지는 가능성을 줄이는 방향으로 고안되었다. 그리고 프레임 간격이 커지면 다음 프레임 간격에서 화면 전환이 발생할 확률도 커지는데 이 논문에서는 정적인 화면 상태가 오래 지속되더라도 어느 한계 이후로는 최대 프레임 간격을 계속 유지하는 정책을 취함으로써 화면 전환에 빠르게 대처할 수 있게 하였다.

그럼에도 불구하고 Case 3과 4의 경우에서는 몇몇 프레임에서 화질 위반이 발생할 수 있다. 이는 입력 프레임 데이타의 불확실성 때문에 피할 수 없는 결과이고 이 예측 기법의 주된 단점이다. 그러나 제약 위반이 드물게 발생한다면 전체 화질은 수용 가능한 것일 수도 있다. 제약 위반의 확률은 α에 의존하기 때문에 이 값을 안정적(conservative)으로 설정하는 것이 중요하다.

이 기법은 직접 비교 기법에 비해 다음처럼 몇 가지 장점이 있다. 1) 프레임 생략 작업이 따로 분리되지 않고 대신 부호화 작업에 덧붙여진다. 그러므로 제안한 알고리즘의 총 시간 부가는 작아진다. 2) 최대 지연 시간이 반으로 줄어든다. 3) 추가 버퍼가 필요하지 않다.

그림 7은 예측 프레임 생략 기법의 적용 예를 보여준다. 직접 비교 기법의 경우와 비슷하나 여기서는 버퍼링

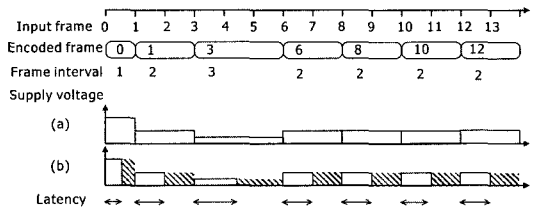

그림 7 예측 프레임 생략 기법의 부호화 예 (a) WCET에서 프레임 생략 결과 (b) AET에서 프레임 생략 결과

기법이 적용되지 않는 차이점이 있다. 그러므로 그림 7(b)에 보인 것처럼 AET가 WCET보다 작을 때 여기서는 그 유휴 시간을 활용할 수 없게 된다.

6. 실 험

이 논문에서는 H.263 부호화 알고리즘으로 TMN 3.0 소스 코드[13]를 고정 소수점(fixed-point) 버전으로 최적화하여 사용하였다. 이 소스 코드에 제안한 기법을 구현하고, ARM720T 프로세서를 대상으로 컴파일하여, ARM 프로세서 시뮬레이터인 ARMulator 상에서 실행시켰다.

에너지 모델은 SimDVS[14]를 이용하였다. SimDVS는 다양한 DVS 알고리즘의 성능을 평가하기 위해 고안된 시뮬레이션 환경이다. 부호화 태스크의 각 프레임 실행 시간을 ARMulator로부터 추출하여 SimDVS의 에너지 모델에 넣으면, 각 프레임 부호화에 소모된 에너지가 계산된다. 프로세서의 최대 전력 소모은 20mW이고, 동작 주파수는 최대 100MHz에서 하한(lower bound)으로 최대값의 0.5배(50MHz)를 가정하였다.

화면 변동의 실제적인 시나리오를 흉내내기 위해서 300 프레임짜리 CIF(common intermediate format) 비디오 시퀀스 네 개 - Coastguard, Hall monitor, Foreman, Container - 를 이어서 1200 프레임짜리로 만들어 사용하였다. 최대 프레임 속도는 30FPS로 설정하였다.

그림 8에 보인 것처럼, 예측 프레임 생략 기법의 화질 제약 위반율(quality constraint violation ratio, 위반된 프레임 수 / 전체 프레임 수)은 알고리즘의 임계 (threshold)가 커질수록 감소한다. 반대로, 그림 9에 보인 것처럼, 프레임 속도는 임계값이 커질수록 증가한다. 따라서 적절한 임계값의 설정은 화질과 프레임 속도 (또는 에너지 소모) 사이의 교환(tradeoff) 문제와 관련이 있다. 실험에서는 이 값을 2dB로 설정하여 사용하였다.

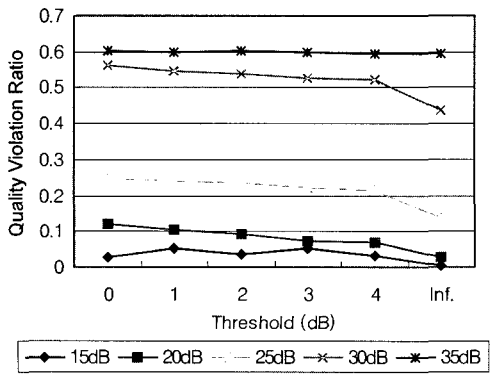

그림 8 임계(threshold)에 따른 화질 위반율

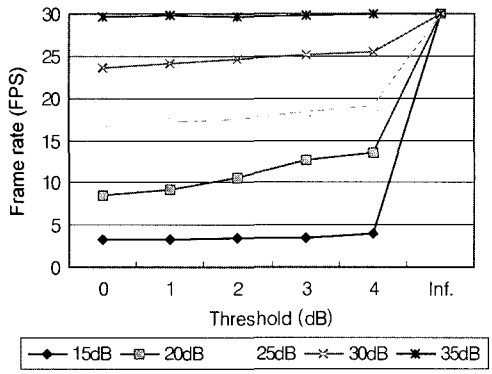

그림 9 임계(threshold)에 따른 프레임 속도

6.1 화질 분포

제안한 VFR 기법이 주어진 화질 제약 하에서 어떻게 동작하는가 보여주기 위해서, 그림 10과 11에 각각 FFR 기법과 25dB 화질 제약 하에서의 VFR 기법에 대한 화질 분포를 도시하였다. 그림 11의 DR 및 PRE는 각각 직접 비교 기법과 예측 기법을 나타낸다. 30FPS의 최대 프레임 속도에서 화질이 10에서 50까지 분포함을

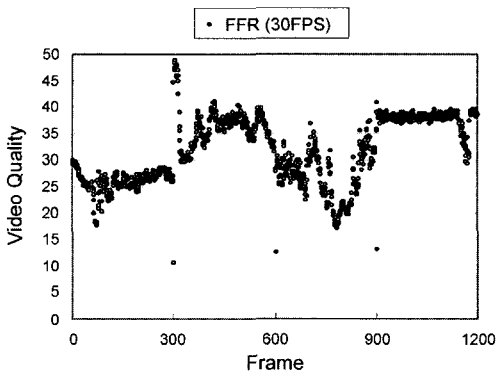

그림 10 FFR 기법(30FPS)의 화질 분포

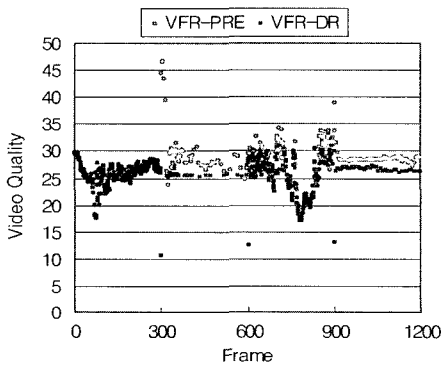

그림 11 VFR 기법(25dB 제약)의 화질 분포

주목하라. 이 실험에서는 지연 시간 제약은 고려하지 않았다.

직접 비교 기법에서는 모든 프레임이 화질 제약을 만족하지만, 예측 기법에서는 몇몇 프레임이 이를 위반한다. 또 다른 차이는 직접 비교 기법이 화면 변동에 독립적으로 꽤 안정적인 화질을 유지하는 반면 예측 기법은 프레임 간격의 동적인 적응으로 인해 좀더 폭넓은 변동을 보인다는 점이다.

'Hall monitor'(301-600)와 'Container'(901-1200)처럼 정적인 장면에서는 많은 프레임이 생략된 점이 두드러진다. 실제로 제공된 화질 제약에 대한 평균 프레임 속도는 직접 비교 기법에서는 15.5FPS, 예측 기법에서는 17.6FPS로 감소하였다. 지연 시간 제약이 고려가 된다면 평균 프레임 속도는 다소 증가할 것이다.

6.2 화질 제약 하에서의 성능 비교

이 절에서는 여러 가지 에너지 최적화 기법들의 성능을 다양한 화질 제약 조건 하에서 비교하였다. 지연 시간 제약은 없는 것으로 가정하였는데, 이는 다음 절에서 고려할 것이다.

모두 6가지 기법을 사용하였는데, 1) 전력 차단(power-down)만을 이용하는 FFR 기법 2) 버퍼링과 DVS를 이용하는 FFR 기법 3) 전력 차단만을 이용하는 VFR-DR 기법 4) 버퍼링과 DVS를 이용하는 VFR-DR 기법 5) 전력 차단만을 이용하는 VFR-PRE 기법 6) DVS를 이용하는 VFR-PRE 기법이다. 1)은 모든 에너지 소모 평가의 기준이 되는, 최대 공급 전압에서의 FFR 기법이다. 2)는 FFR에서 VST를 이용하기 위해 버퍼를 이용하는 기법이다. 3)과 4)는 직접 비교 프레임 생략 기법으로 DVS의 이용여부에 따라 구분한다. 5)와 6)은 예측 프레임 생략 기법으로 마찬가지로 DVS의 이용여부에 따라 구분한다. 앞서 설명한 바와 같이 버퍼링은 VFR에서 DR 기법에만 적용하여 사용하고, PRE 기법은 버퍼 오버헤드가 없다. 모든 기법은 프로세서가 휴지 상태일 때 전력 공급을 차단한다.

VFR 기법에서는 화질 제약을 각각 15, 20, 25, 30, 35dB로 주어 실험하였다. 모든 기법에 대해서 화질 제약에 따른 평균 프레임 속도와 에너지 소모를 구하였다. 결과는 그림 12와 13에 나타내었다.

그림 12는 20dB의 화질 제약 하에서 평균 프레임 속도가 직접 비교 기법에서는 2.25FPS까지, 예측 기법에서는 3.5FPS까지 감소함을 보여준다. 예측 기법은 α(threshold)의 사용으로 인해 직접 비교 기법에 비해 적은 수의 프레임을 생략한다.

그림 13은 버퍼링과 DVS를 이용하는 VFR-DR 기법(VFR-DR + BUF)이 가장 좋은 성능을 얻고 따라서 실제 지연 시간 제약이 있는 경우에 대해서 하한

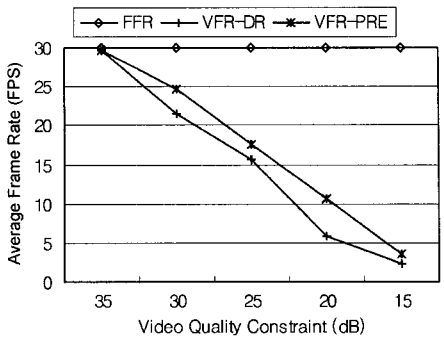

그림 12 화질 제약에 따른 평균 프레임 속도

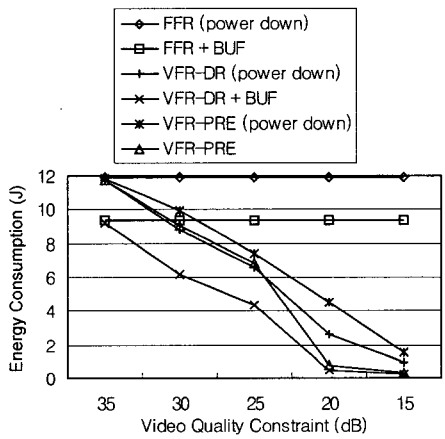

그림 13 화질 제약에 따른 에너지 소모

(lower bound)을 설정함을 보여준다. 다음 절에서 볼 수 있듯이, 지연 시간 제약이 고려된다면 모든 기법의 성능은 감소하게 된다. 지연 시간이 고려되지 않는다면, 예측 기법은 직접 비교 기법에 비해 나쁜 성능을 보인다.

그림 13은 또한 프레임 생략을 하지 않고 버퍼링과 DVS를 이용하는 기법(FFR + BUF)이 22%의 에너지 감소를 가져옴을 보여준다. 이것과 비교하여, 프레임 생략 기법들은 전력 차단만 적용하더라도 화질 제약이 낮을 때에는 훨씬 효과적으로 동작함을 알 수 있다.

6.3 지연 시간 제약 하에서의 성능 비교

이 절에서는 지연 시간 제약에 따른 성능 변화 결과를 제시하겠다. 그림 14와 15는 지연 시간 제약에 따른 평균 프레임 속도와 에너지 소모를 보여준다. 화질 제약은 20dB로 가정하였고 앞 절에서의 결과는 지연 시간 제약이 무한대인 경우의 성능으로 사용된다.

그림 15는 지연 시간 제약이 엄하게 주어졌을 때 예측 기법이 직접 비교 기법보다 좋은 성능을 나타냄을 보여준다. 이는, 같은 지연 시간 제약 하에서, 예측 기법의 MFI가 직접 비교 기법의 그것보다 크기 때문에, 그

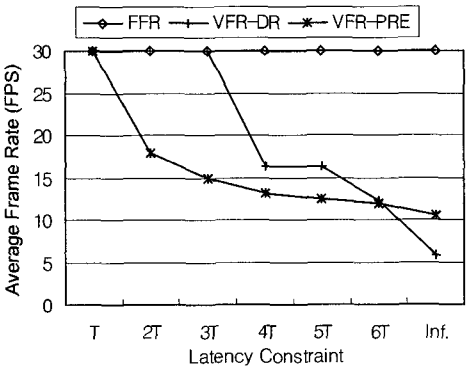

그림 14 지연시간 제약에 따른 평균 프레임 속도

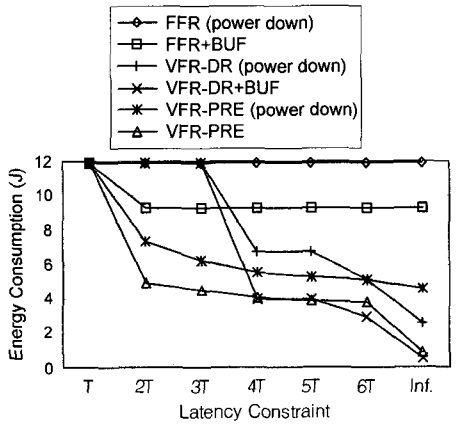

그림 15 지연시간 제약에 따른 에너지 소모

림 14에 나타난 것처럼 전자의 기법이 더 많은 프레임을 생략할 수 있기 때문이다. 사실 3T의 지연 시간 제약 하에서, 직접 비교 기법은 전혀 성능 향상이 없는 반면, 다른 기법들은 괄목할 만한 성능 향상을 보인다. 한편 4T 이상의 지연 시간 제약 하에서는 직접 비교 방법이 최고의 성능 향상 기법으로 부상한다. 이러한 현상은 직접 비교 기법이 프레임 생략의 구현을 위해서 최소한 두 개의 추가 버퍼(4T 지연 시간)를 필요로 하기 때문에 발생한다.

7. 결 론

이 논문은 VFR 기법이 FFR 기법보다 화질 제약 하에서의 에너지 감소에 훨씬 효과적이라는 점의 관찰에서 출발한다. 이 논문에서는 버퍼링 기법과 프레임 생략 기법을 제안하였고, 실험을 통해서 제안한 기법들이 지연 시간과 화질 제약 하에서 프레임 속도 및 에너지 소모를 상당히 감소시킬 수 있음을 보였다. 또한 지연 시간 제약이 없을 때는 직접 비교 기법이, 반면 지연 시간 제약이 엄할 때는 예측 기법이 최대 에너지 감소를 얻게 됨을 보였다.

참 고 문 헌

[1] Pedram, M. and Rabaey, J. M., Power Aware Design Methodologies, Kluwer Academic Publishers, 2002.

[2] Chandrakasan, A. P., Sheng, S., and Brodersen, R. W., "Low-Power CMOS Digital Design," IEEE Journal of Solid-State Circuits, Vol.27, No.4, pp. 473-484, 1992.

[3] Taylor, C. N., Dey, S., and Panigrahi, D., "Energy/Latency/Image Quality Tradeoffs in Enabling Mobile Multimedia Communication," Proc. of Software Radio: Technologies and Services, pp. 55-66, 2001.

[4] Im, C., Kim, H., and Ha, S., "Dynamic Voltage Scheduling Technique for Low-Power Multimedia Applications Using Buffers," Proc. of International Symposium on Low Power Electronics and Design, pp.34-39, 2001.

[5] Hong, I., Kirovski, D., Qu, G., Potkonjak, M., and Srivastava, M. B., "Power Optimization of Variable Voltage Core-Based Systems," Proc. of Design Automation Conference, pp. 176-181, 1998.

[6] Shin, Y. and Choi, K., "Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems," Proc. of Design Automation Conference, pp. 134-139, 1999.

[7] Cao, Y. and Yasuura, H., "Quality-Driven Design by Bitwidth Optimization for Video Applications," Proc. of Asia and South Pacific Design Automation Conference, 2003.

[8] Cha, H., Oh, J., and Ha, R., "Dynamic Frame Dropping for Bandwidth Control in MPEG Streaming System," Multimedia Tools and Applications, Vol.19, No.2, pp. 155-178, 2003.

[9] Hua, S., Qu, G., and Bhattacharyya, S. S., "Energy Reduction Techniques for Multimedia Applications with Tolerance to Deadline Misses," Proc. of Design Automation Conference, pp. 131-136, 2003.

[10] Song, H., Kim, J., and Kuo, C.-C. J., "Real-Time Encoding Frame Rate Control for H.263+ Video over the Internet," Signal Processing: Image Communication, Vol.15, No.1-2, pp. 127-148, 1999.

[11] Im, C., and Ha, S., "Dynamic Voltage Scaling for Real-Time Multi-task Scheduling Using Buffers," Proc. of Conference on Languages, Compilers, and Tools for Embedded Systems, pp. 88-94, 2004.

[12] Burd, T. D., and Brodersen, R. W., "Design Issues for Dynamic Voltage Scaling," Proc. of International Symposium on Low Power Electronics and Design, pp.9-14, 2000.

[13] University of British Columbia Image Processing

Laboratory, http://www.ee.ubc.ca/image/
[14] Shin, D., Kim, W., Jeon, J., Kim, J., and Min, S. L., "SimDVS: An Integrated Simulation Environment for Performance Evaluation of Dynamic Voltage Scaling Algorithms," Proc. of Workshop on Power-Aware Computer Systems, 2002.

임 채 석

1997년 2월 서울대학교 컴퓨터공학과 학사. 1999년 2월 서울대학교 컴퓨터공학과 석사. 2004년 8월 서울대학교 전기컴퓨터공학부 박사. 관심분야는 저전력 멀티미디어 시스템 설계, 시스템 수준의 에너지 예측, 저전력 내장형 운영체제

하 순 회

1985년 2월 서울대학교 전자공학과 학사 1987년 2월 서울대학교 전자공학과 석사 1992년 미국 UCB 전기컴퓨터공학과 박사. 1993년~1994년 현대전자 근무. 1994년~현재 서울대학교 컴퓨터공학부 부교수. 관심분야는 하드웨어-소프트웨어 통합설계, 내장형 시스템을 위한 설계방법론