DVS를 이용한 저전력 MPEG 디코더

손동환° 이형석 김선자
한국전자통신연구원

Low Power MPEG Decoder with DVS Algorithms

Donghwan Son
ETRI-Computer & Software Technology Laboratory
E-mail : dhson@etri.re.kr, hyslee@etri.re.kr, sunjakim@etri.re.kr

요 약

동적 전압 조정(DVS)은 모바일 환경에서 프로세서에서의 전력 소모를 줄일 수 있는 가장 효율적인 방법으로 많은 연구가 진행중이다. 또한 MPEG 디코딩은 모바일 기기에서 가장 중요하고 또한 전력 소모가 큰 어플리케이션 중 하나이다. 본 논문에서는 모바일 환경에 적합한 MPEG 디코더를 DVS를 이용하여 구현하였고 전력 소모를 측정하였다. 제안된 첫번째 DVS 알고리즘은 이전의 workload에 의해 다음 workload를 예측하여 전압을 조정하는 것이고, 두번째 알고리즘은 MPEG 프레임의 종류 및 크기를 이용하여 다음 프레임의 디코딩 시간을 예측 한 후, 전압을 조절하는 것이다. 실험을 통하여 두번째 알고리즘에 의한 MPEG 디코더가 더 정확한 workload 예측을 통하여 QoS의 저하를 최소화 하면서 전력 소모를 더 많이 줄일 수 있었다.

1. 서론

이동 기기의 프로세서에서의 에너지 소모는 이동컴퓨팅 작업이 복잡해지고 내장형 시스템의 증가로 인하여 전체 시스템의 에너지 소모의 많은 부분을 차지하고 있다. 프로세서의 에너지 소모를 줄이는 가장 효율적인 방법은 전압을 낮추는 것인데 이러한 전압의 감소는 프로세서 스피드의 감소를 가져오므로 일정 전압으로 낮추는 대신 시스템 작업량에 따라 동적으로 전압을 조정하는 방법에 대한 많은 연구가 이루어지고 있다[1,2].

동적 전압 조정(Dynamic Voltage Scaling, DVS)은 프로세서가 run-time에 동적으로 클럭 속도와 동작 전압을 변화하도록 함으로써 에너지 효율을 높이는 기술이다. Interval 기반의 DVS는 시간을 일정한 간격으로 나눈 후, 이전 interval들에서의 workload에 관한 정보를 가지고 다음 interval에서의 workload를 예측한 후, 그에 따라서 전압을 변동하는 기술로써 이러한 방식의 DVS는 간단하고 구현이 쉬우나 workload의 변화가 크면 효율성이 떨어질 수 있다.

한편 이동컴퓨팅의 어플리케이션에서 점차 멀티미디어의 비중이 커지고 있으며 그 핵심에 있는 MPEG 디코더는 많은 프로세서 파워를 요구하고

따라서 전력소모가 큰 어플리케이션이다. MPEG 디코더의 특징은 프레임의 디코딩 시간의 변화가 크다는 것인데 이로 인하여 이전 단계의 작업량에 기초하여 다음 단계에 대한 작업량을 예측하는 것이 어렵기 때문에 이에 기반 한 동적 전압 조정은 효율적이지 않다. 하지만 MPEG 프레임의 유형 및 크기를 통한 디코딩 시간 예측이 가능하다.

이 논문은 MPEG 디코딩에서의 동적 전압 조정 알고리즘을 기존 MPEG 디코더에 구현 및 실험하여 에너지 소모를 비교하는 것을 목적으로 하며, 첫번째 알고리즘은 디코딩을 수행하는 동안 이전 단계의 작업량을 나타내는 delay와 drop rate를 이용하여 다음 interval에서의 전압을 결정하는 알고리즘이고, 두 번째 알고리즘은 MPEG 프레임 사이즈를 이용하여 디코딩 시간 예측을 통해 다음 interval의 전압을 조정하는 것이다. 이 알고리즘들은 Boston University에서 구현한 MPEG Decoder에 적용이 되었고 여섯 개의 샘플 스트림을 통해 실험되었다. 나타난 결과에 의하면 디코딩 시간 예측을 통한 전압 조정이 다른 알고리즘에 비해 많은 에너지 절약을 가져왔고 에너지 소모가 디코딩 시간 예측의 정확도에 영향을 받는다는 사실을 확인할 수 있었다.

2. 관련연구

2.1 동적 전압 변동 (DVS)

디지털 CMOS 회로에서의 동작에 의한 전력 소모는 아래의 식으로 나타난다[3].

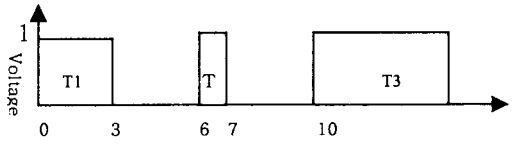
$$E \propto C_{eff} V^2 f_{CLK}$$

여기서 C_{eff} 는 유효 스위칭 정전 용량이고, V 는 전압, 그리고 f_{CLK} 는 클럭 주파수이다. 다른 한편으로, 회로의 지연(d)와 클럭 주파수는 아래의 식으로 주어진다.

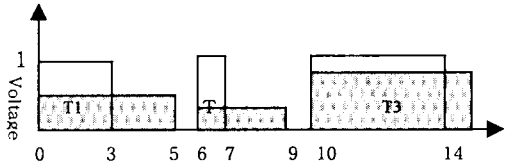
$$d \propto \frac{V}{(V - V_k)^{\alpha}} \qquad f_{CLK} \propto \frac{(V - V_k)^{\alpha}}{V}$$

여기서 α는 1에서 2 사이의 값이고, V_k 는 포화가 일어나는 임계전압에 의존한다. 이 식들은 전압을 낮추면 지연이 길어지면서 프로세서의 clock speed가 낮아지지만, 동시에 에너지 소모가 전압의 제곱에 비례하여 감소하는 것을 보여준다.

동적 전압 변동 (Dynamic Voltage Scaling, DVS)는 전압 scheduler로 하여금 프로세서의 동작 전압을 run-time에 바꿔줌으로써 에너지 소모를 줄이는 기술이다. 프로세서에서의 전력 소모를 줄이는 기존의 방법으로는 idle 시간에 CPU를 멈추는 shutdown 메커니즘이 있다. 이 방법은 에너지 소모가 전압의 제곱에 비례하므로 아래 예에서와 같이 DVS보다 효율적이지 않다. 그림 1은 shutdown 메커니즘과 DVS의 예를 보여준다. X축은 시간을 나타내고, Y축은 전압과 clock frequency를 나타낸다. task에 필요로 하는 작업량은 각 작업의 면적과 같다. 그림 1(a)는 3개의 작업이 최대 전압으로 수행되고 idle시에는 shutdown되며 전체 에너지 소모는 8 energy unit이다. 그림 1(b)에서는 전압이 각각 3/5, 1/3, 2/3으로 설정되고 에너지 소모는 2.97[1]로 계산되어진다.



(a) shutdown 알고리즘에 의한 CPU workload



(b) DVS 알고리즘에 의한 CPU workload

[그림 1] 두 알고리즘에서의 전압 조정 그래프

2.2 MPEG 디코딩

MPEG 1 스트림은 I, P, B 프레임들로 구성된 GOP(Group of Pictures)로 구성된다. MPEG 디코딩의 실시간 특성을 위하여 디코더는 현재 시스템의 속도

[1] 이 값은 $(3/5)^2 \times (3/5)\times 5 + (1/3)^2 \times (1/3)\times 3 + (2/3)^2 \times (2/3)\times 6 = 2.97$ 로 계산된다. C_{eff}는 1 로 가정한다.

를 모니터하고 각 프레임을 실시간을 위해 play 할 것인지, 또는 다음 프레임 까지 delay를 줄 것인지를 결정한다. 이를 위하여 디코더는 4개의 real-time phase(DELAY-PHASE, B-PHASE, P-PHASE, I-PHASE)를 가지고 있다. DELAY-PHASE는 프레임 디코딩 사이에 delay를 줘서 play를 늦추는 것이고 B-PHASE는 실시간성을 맞추기 위하여 B 프레임을 drop할 수 있는 phase이고 P-PHASE는 시스템이 더 느려서 B 뿐만 아니라 P 프레임도 drop 할 수 있는 phase이며 I-PHASE는 모든 프레임들을 drop할 수 있는 phase이다.

디코딩 시간에 관련하여서는 I 프레임이 가장 많은 시간을 필요로 하고 B 프레임이 가장 작지만 byte 당 디코딩 시간(Decoding Time Per Byte, DTPB)는 I가 가장 작고 B가 가장 크다. 이것은 B 프레임의 경우 motion prediction을 위한 계산 시간이 필요하기 때문이다. 이와 같은 특성을 이용하여 프레임 디코딩 시간은 프레임 종류와 크기에 따라서 예측될 수 있고 정확도는 25% 이내이다[4].

2.3 가정

실제로 DVS가 구동되는 하드웨어를 구하는 것은 아직 현실적으로 어려움이 있으며, 따라서 본 실험에서는 전압을 조정하는 대신에 frames per second를 변화시킴으로써 같은 효과를 가져왔다. 또한 전압 및 clock speed는 0.5 ~ 1 범위 내에서 0.1 간격으로 6 단계가 있다고 가정하였다.

3. MPEG 디코딩에서의 DVS 알고리즘

3.1 delay 및 drop rate 최소화 DVS 알고리즘(DDM)

delay와 drop rate은 시스템의 workload를 나타내는 두개의 중요한 변수이다. delay값은 시스템의 디코딩 속도가 너무 빨라서 주어진 frames per second보다 더 많은 수의 프레임이 play될 경우 프레임 디코딩 사이에 delay를 주기 위해 설정된다. 큰 delay 값은 시스템 속도가 스트림을 디코딩하기 위해 필요한 시스템 속도보다 크다는 것을 의미한다. delay 값이 0 이라면, 시스템 속도는 디코딩의 실시간성을 만족시키는 것이거나 느려서 프레임 drop이 일어나는 경우이다. drop rate는 이 시스템에서 모든 프레임의 디코딩이 불가능할 때, drop되는 프레임과 play되는 프레임의 비로써, 각각의 프레임 종류에 따라 B, P 및 I-frame drop rate으로 되어있다.

DDM은 delay값과 drop rate 모두를 최소화하려고 시도한다. GOP의 시작 점에서 전압은 delay와 drop rate에 의해 조정된다. delay 값이 양이면(DELAY-PHASE), 전압과 주파수는 delay값에 비례하여 낮아지고, 시스템이 B-PHASE이면 전압과 주파수가 drop rate에 비례하여 높아지고, I 및 P-PHASE이면, 전압 및 주파수는 최대값으로 설정된다. 이 알고리즘은 그림 2와 같이 구현되어 있다.

```
static void scale_voltage_delay( )
{
   /*voltage is set to maximum voltage*/
   if((phase==I_PHASE)||(phase ==P_PHASE))
      set_max_voltage( );
   else if(phase == B_PHASE)
   {   /* scale up voltage */
      current_voltage = current_voltage
               + drop_rate*increase_factor;
      set_voltage(current_voltage);
   }
   else if(phase == DELAY_PHASE)
   {
      if(delay<100)/*keep current voltage*/
         return;
      else {  /* scale down voltage */
         current_voltage =current_voltage
                  - delay*decrease_factor;
        set_voltage(current_voltage);
      }
   }
}
```

[그림 2] DDM 알고리즘

3.2 예측된 MPEG 디코딩 시간에 의한 DVS(PDT)

DDM은 delay 및 drop rate과 같은 workload history에 근거한 변수에 의존하므로 MPEG 디코딩과 같이 workload의 변동이 큰 응용 프로그램에서는 효율적이지 않다. 이러한 큰 workload의 변동은 MPEG 비디오 스트림이 여러 프레임 타입으로

구성되어 있는 것과 장면들 사이의 변동이 크다는 것에 기인한다. 이것은 과거의 상태에 근거한 디코딩 시간의 예측을 어렵게 한다. 따라서 과거의 history와 예측된 디코딩 시간에 의한 DVS 알고리즘(PDT)이 제안되었다.

디코딩 시간은 MPEG 프레임 종류와 크기 및 과거의 workload 정보에 의해 각 GOP의 시작 시에 예측된다. 2.2 장에서 설명되었듯이, 스트림을 디코딩할 때, 각 프레임 종류는 서로 다른 바이트 당 디코딩 시간 (decoding time per byte, DTPB)을 가진다. 즉, I, P 및 B 프레임의 DTPB는 같지 않으며, 더욱이 이 값들은 장면의 역동성에 비례하여 변화한다. 장면의 특성과 workload의 변화에 적응하기 위하여, PDT는 우선 update_statistics 루틴에서 과거 DTPB 값들을 이용하여 GOP의 평균 DTPB를 계산한다. weight_factor는 DTPB들의 가장 최근 값과 과거의 history에 의한 값을 적절히 분배하며, 특정한 값으로 설정된다. 이 실험에서의 weight_factor는 0.4로 주어졌으나, 최적의 weight_factor는 스트림에 의존적이다. 그 다음에 다음 GOP의 I, P 및 B 프레임의 크기를 구하고, 이 프레임 크기와 DTBP 값을 곱함으로써 다음 GOP 디코딩 시간을 예측하고 이에 따라서 전압과 주파수를 설정한다.

balance_factor는 최근 interval의 예측된 workload와 실재 workload의 차이를 나타내는 값으로, I_P_balance, B_balance 및 delay와 drop rate에 의해 계산된다. I_P_balance, B_balance 그리고 DELAY_balance는 각각 I, P, B-PHASE 및 DELAY_PHASE일 경우 balance_factor를 구하기 위한 상수이다. balance_factor가 음수값이면 알고리즘이 이전 interval의 workload를 실재보다 낮게 예측했다는 것을 의미하며, 양수이면 그 반대로 예측했다는 것을 나타낸다. 따라서 예측된 전압에 balance_factor를 더해줌으로써 알고리즘이 MPEG 디코더 밖의 다른 어플리케이션에 의한 workload 변화에 대하여 대처할 수 있도록 하였다.

```
static void scale_voltage_predict( )
{
  update_statistics( );
  get_GOP_size( );
  decode_time= est_decode_time( );
  volt=set_volt_predict( decode_time );
  set_voltage(volt);
}
void update_statistics( )
{
   /* weighted average */
  avg_I_DTPB=avg_I_DTPB*(1-weight_factor)
              + I_DTPB *weight_factor;
  avg_P_DTPB=avg_P_DTPB*(1-weight_factor)
              +P_DTPB*weight_factor;
  avg_B_DTPB=avg_B_DTPB*(1-weight_factor)
             + B_DTPB * weight_factor;
}
void get_GOP_size( )
{
  I_size = get_GOP_I_size( );
  P_size = get_GOP_P_size( );
  B_size = get_GOP_B_size( );
}
double est_decode_time( )
{
  I_decode_time = I_size * avg_I_DTPB;
  P_decode_time = P_size * avg_P_DTPB;
  B_decode_time = B_size * avg_B_DTPB;
  return I_decode_time + P_decode_time
        + B_decode_time;
}
void set_volt_predict( d_time )
{
  if((phase==I_PHASE)||(phase==P_PHASE))
     balance_factor = I_P_balance;
  else if(phase == B_PHASE)
     balance_factor = drop_rate*B_balance;
  else if(phase == DELAY_PHASE){
     balance_factor = delay*DELAY_balance;
/*scaled well, thus no volt. change occurs*/
  if(delay<100)
    {
/*standard voltage and time are set to
current values to adjust to current
workload*/
     std_voltage = voltage;
     std_dtime = d_time;
    }
  return std_voltage*d_time/std_dtime
        + balance_factor;
}
```

[그림 3] PDT 알고리즘

4. 성능 평가

4.1 실험 환경

MPEG 디코더

본 실험에서는 Boston University에서 Berkeley MPEG 디코더를 기반으로 구현한 MPEG 디코더를 기본 디코더로 한 4개의 MPEG 디코더가 비교되었다. 첫번째 디코더는 변형하지 않은 Boston University 디코더이고, 두번째 디코더는 2.1장에서 설명된 shutdown 알고리즘을 적용한 디코더이다. 세번째와 네번째 디코더는 Boston University의 디코더에 DDM과 PDT를 각각 적용한 디코더이다.

MPEG 스트림

실험을 위하여 6개의 각기 다른 샘플 스트림을 선택했다. 이 스트림들은 평균 프레임 크기, 평균 디코딩 시간, GOP 디코딩 시간의 편차 그리고 디코딩 시간의 변동(fluctuation)으로써 아래 표와 같이 분류되었다. 디코딩 시간의 변동은 아래식으로 표현된다.

$$fluct = \frac{\sqrt{\sum(decode\_time_i - decode\_time_{i-1})^2}}{num\_GOP}$$

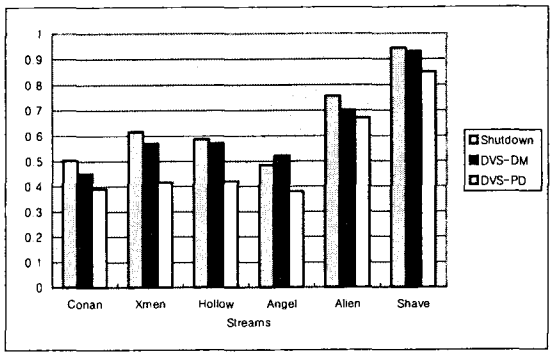
여기서 $decode\_time_i$ 는 i번째 GOP의 디코딩 시간이고, num_GOP 는 스트림 안의 GOP 수이다. 따라서 변동은 인접한 GOP 디코딩 시간의 차이의 정도를 나타낸다.

Movies	Avg. GOP Size	Frames Per Second	Avg Dec. Time	Dev. of GOP Dec. Time	Fluctuation
Alien	96066	23.976	0.378	0.00920	0.0000745
Conan	76778	30	0.232	0.00057	0.000007
Shave	96065	23.976	0.531	0.00115	0.000025
X-men	76775	30	0.272	0.00029	0.000001
Hollow	96066	23.976	0.313	0.00094	0.000013
Angel	92131	25	0.267	0.00834	0.0000565

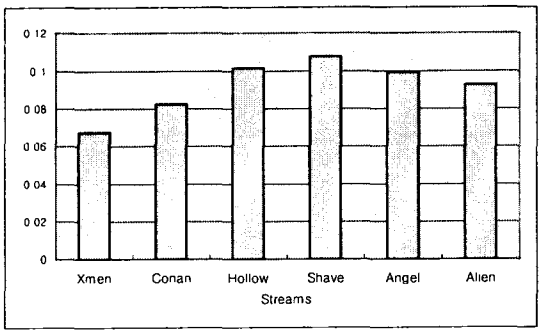
[표 1]실험에 사용된 샘플 스트림의 특성

4.2 실험 결과

그림 4는 6개의 샘플 스트림들에 의한 4개의 MPEG 디코더에서의 상대적인 에너지 소모를 나타낸다. 여기서 기본 디코더의 에너지 소모를 1로 가정하였다. 4개의 디코더 중에서 PDT 알고리즘에 의한 디코더가 가장 좋은 에너지 효율을 보여주고 있다. DDM 역시 shutdown 알고리즘보다 더 효율적임을 알 수 있다.



[그림 4] 에너지 소모
(original MPEG 디코더가 1의 에너지를 소모한다고 가정함.)



[그림 5] PDT에서의 *error_rate*
(스트림은 fluctuation의 순서로 정렬되었음)

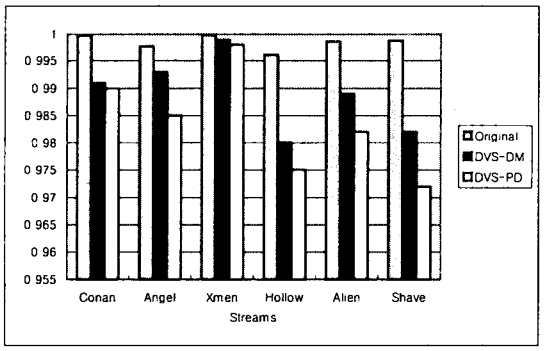
PDT의 좋은 성능은 다음 workload에 대한 예측이 정확하다는 사실에 기인한다. 이러한 정확성을 측정하기 위하여 *error_rate*를 다음과 같이 정의하였다.

$$err\_rate = \frac{\sqrt{\sum(est\_dec\_time - dec\_time)^2}}{num\_GOP}$$

여기에서 est_dec_time 는 예측된 GOP 디코딩 시간이고, dec_time 은 실재 GOP 디코딩 시간이며, num_GOP 는 GOP의 개수이다. PDT 알고리즘에서의 각

예제 스트림에 대한 *error_rate*는 그림 5와 같다. 그림에서의 스트림은 fluctuation 값의 순서로 나열되어 있고, 이것은 *error_rate*가 fluctuation과 관계가 없음을 나타내고 있다. 따라서, PDT는 스트림의 fluctuation이 큰 경우에도 좋은 성능을 보여준다. 그림 4에서와 같이 Xmen에서는 shutdown이나 DDM에 비하여 PDT에서의 에너지 소모가 현격하게 줄어들 수 있었고, 반대로 Shave에서는 그 차이가 크지 않음을 볼 수 있는데, 이것은 *error_rate*의 차이에서 예측될 수 있다. 따라서 예측이 정확할수록 에너지 소모를 줄일 수 있다고 결론지을 수 있다.

그림 6은 PDT에서의 에너지 소모가 적은 반면에 다른 알고리즘에 비하여 많은 QoS의 저하를 가져올 수 있음을 보여준다. DDM에서의 높은 QoS는 이 알고리즘 자체가 drop-rate를 최소화 하려는 것이기 때문이다. 그림 6의 스트림은 평균 GOP 디코딩 시간의 순으로 정렬되어 있고, 이것은 디코딩 시간이 클수록 QoS 가 저하됨을 보여준다.



[그림 6] 전체 프레임에 대한 play된 프레임의 비율

5. 결론

DVS는 프로세서의 전력 소모를 줄이는 강력한 방법이다. 그러나 MPEG 디코딩과 같이 workload의 변화가 큰 어플리케이션에 적용하기 위해서는 다음 workload의 정확한 예측을 위한 알고리즘이 필수적이다. 이 논문에서는 workload의 변화에 적응하기 위하여 DDM과 PDT 알고리즘을 제안하였다. 이들은 각각 이전의 workload에 대한 정보와 예측된 MPEG 디코딩 시간에 의존하여 다음 interval에서의 전압을 결정함으로써 에너지 소모를 줄인다. 우리는 6개의 예제 스트림을 통하여 Boston University의 MPEG 디코더와 이 디코더에 shutdown, DDM 및 PDT알고리즘을 구현한 MPEG 디코더의 에너지 소모 및 QoS를 측정하였고 PDT에서 약간의 QoS의 저하가 있었지만 에너지 소모가 가장 적음을 알 수 있었다. PDT에서 에너지를 절약할 수 있었던 이유는 미래의 workload에 대하여 다른 알고리즘에 비해 정확히 예측을 할 수 있었기 때문이다. 실험 결과를 통하여 에너지 절약은 평균 디코딩 시간과 fluctuation에 관련되어 있음을 알 수 있었다. DDM에서는 MPEG의 큰 fluctuation으로 인하여 미래의 workload에 대한 예측의 정확도가 떨어져서 에너지 소모가 상대적으로 컸다. 반대로 PDT는 fluctuation에 거의 영향을 받지 않는 예측의 정확도로 효율적인 DVS를 수행할 수 있음을 알 수 있었다.

[참고문헌]

[1] T. Burd and R. Brodersen, "Energy Efficient CMOS Microprocessor Design," *28th Hawaii Int'l Conf. on System Science*, Vol. 1, pp. 288-297, Jan. 1995.

[2] T. Burd and R. Brodersen, "Design Issues for Dynamic Voltage Scaling," *Int'l Symp. on Low power Electronics and Design*, pp. 9 - 14, 2000.

[3] K. Govil, E. Chan, and H. Wasserman, "Comparing Algorithms for Dynamic Speed-Setting of a Low-Power CPU," *Technical Report TR-95-017, ICSI Berkeley*, Apr. 1995.

[4] A. Bavier, B. Montz, and L. Perterson, "Predicting MPEG Execution Times," *SIGMETRICS/ PERFORMANCE '98, Int'l Conf. on Measurement and Modeling of Computer Systems*, pp. 131-140, Jun. 1998.

[5] MPEG Decoder from Boston University http://hulk.bu.edu/pubs/software.html#system-player