Yun et al. EURASIP Journal on Embedded Systems 2013, 2013:13
http://jes.eurasipjournals.com/content/2013/1/13

EURASIP Journal on Embedded Systems
a SpringerOpen Journal

RESEARCH Open Access

Dynamic voltage and frequency scaling over delay-constrained mobile multimedia service using approximated relative complexity estimation

Jihyeok Yun*, Deepak Kumar Singh and Doug Young Suh

Abstract

This paper deals with dynamic voltage and frequency scaling (DVFS) in mobile multimedia services. The multimedia services that consume a large amount of energy cannot be continuously used in mobile devices because of battery limitation. The DVFS has been applied to multimedia services in previous studies. However, they have only addressed the issue of power saving and overlooked the fact that mobile multimedia services are sensitive to delays. The proposed method is intended to apply DVFS to multimedia services considering potential delays. Another problem with previous studies is that either separate devices have been employed or appropriate frequency scaling values have been determined through complicated calculation processes to apply DVFS to multimedia services. On the contrary, the proposed method determines appropriate frequency scaling values using the characteristics of multimedia contents without employing any separate devices or undergoing complicated calculation processes. This has the advantage of allowing DVFS to be applied to real-time multimedia content. The present paper proposes a DVFS application method that divides multimedia services into video conferences, which are real-time services, and video streaming, which is a non-real-time service, and that reduces energy consumption in a simple manner while considering the constraints of service delays.

Keywords: Dynamic voltage and frequency scaling; Mobile; Multimedia; Real-time; Power saving

Introduction

The quality requirements for handheld devices' video services have been continuously increasing. As a result, it has been challenging to maintain the high level of quality required to satisfy consumers. In this paper, we propose a dynamic voltage and frequency scaling (DVFS) complexity estimation algorithm that can produce power-saving effects close to those produced using previous DVFS with no additional devices or complexity. DVFS is a method of reducing a processor's power consumption by adjusting applied voltage to a processor dynamically.

According to [1-3], the power consumption of a processor is proportional to the square of its supply voltage, and supply voltage is proportional to frequency. Based on these relationships, power consumption can be reduced by adjusting voltage and frequency appropriately. After estimating the complexity of a processor, which is required for decoding, voltage and frequency will be applied appropriately to the estimated complexity.

In the case of [1], although DVFS was adopted as a power-saving method for wireless mobile devices with limited power, video quality was allowed to deteriorate to reduce the complexity of the codec as with [4]. However, the methods proposed by [1,4] are not suitable for the recent trend of mobile video services in which high-resolution and high-definition video services are preferred.

Ma et al. [2] proposed modeling the complexity of video frames by analyzing the individual module units of video decoders using an appropriate complexity model proposed by [5,6], which adds a complexity profiler to video decoders. This model increases complexity due to the added extra profiler and does not consider the frame drop or buffering phenomenon, which is generated due

* Correspondence: jiyheok.yun@gmail.com
Department of Electronics and Information, Kyunghee University, Yongin 446-701, South Korea

to an estimation error that may occur because of the jittering.

Cho and Cho [3] proposed an algorithm to find the optimum combination of frequency and voltage in which the decoding slack time is 0 while decoding time information is stored because of the frequency and voltage applied to a processor. To this end, Cho and Cho [3] used complexity interpolation based on the frame information (e.g., frame size, frame type) under [7] and the feedback control proposed by [8]. However, since Cho and Cho [3] evaluated the performance of this complexity estimation conducted through interpolation based on super low-resolution images (e.g., 240 × 128, 192 × 144, 192 × 112) with small differences in complexity between frames, its applicability to the current trend of using high-resolution images is uncertain. High-resolution and high-definition images involve significant differences in complexity between frames, and thus, complexity estimation errors using linear interpolation are substantial. In addition, this algorithm has to store decoding information between certain periods and cannot prevent a frame drop or a buffering phenomenon since an estimation of the next frame is calculated after taking into account the overhead due to estimation errors.

In this paper, we propose an estimation method that requires simple profilers or calculations for complexity estimation as well as a DVFS method that prevents frame drop and buffering, which is in contrast with the methods proposed in [2,3]. Our proposed estimation method does complexity estimation with simple profilers or calculations by using the characteristic of multimedia content requiring the repeated processing of similar calculations.

For video content, as the coded frame type is the same and frames are temporally nearer, the similarity becomes higher. This principle is based on the most representative principle of video codec compression. As described in [9], the H.264/AVC standard, which is the most widely used reference codec, also increases the compressibility of the codec using the similarities between temporally close frames. Therefore, it is effective to perform voltage and frequency scaling by taking advantage of complexity information of frames that have been decoded most recently and the same coded frame types requiring no extra calculation processing. In addition, our proposed method sets the limited delay bound of the delays caused by estimation errors, and if the limited delay bound is exceeded, the corresponding frame is decoded with a processor's maximum frequency. As such, if one frame is decoded with maximum frequency, the delay problem is solved, but overhead still exists in terms of power. However, since the number of estimation errors is small due to the offset caused by adding and subtracting repeatedly, and decoding time is reduced significantly when decoded with the maximum frequency, the related overhead is minimal. With such a small amount of energy consumption, frame drop or buffering caused by estimation errors does not occur, and energy-saving effects can be obtained, unlike in existing methods.

Complexity estimation with delay control

As shown in [9], the complexity estimation method proposed in this paper is performed with regard to frame types constituting videos used in H.264/AVC such as intra-frames (I-frame), unidirectionally predicted frames (P-frame), and bidirectionally predictive frames (B-frame), respectively, as shown in Figure 1. It refers the same type of complexity information that has been decoded most recently without extra modeling or estimation algorithms.

The equations for estimation are dependent on the structure of the group of pictures (GOP). The GOP structures usually used in H.264/AVC are shown in Figure 1a, b.

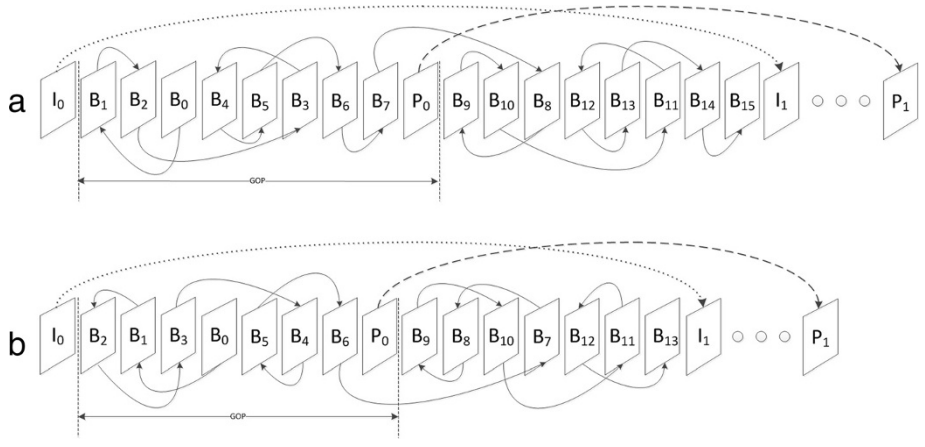

Figure 1 The structures of GOP usually used in H.264/AVC with complexity estimation order. (a) Non-dyadic hierarchical structure. **(b)** Dyadic hierarchical structure. The round dot dashed line shows the reference structure for the I frame, the long dashed line shows the reference structure for the P frame, and the solid dashed line shows the reference structure for the B frame.

Figure 1a is a non-dyadic hierarchical structure, and Figure 1b is dyadic hierarchical structure.

Since our proposed model targets video service, it takes advantage of one characteristic of videos: their similarity between temporally close frames. In our method, referenced frames for complexity can be searched using the GOP size and the intra-period, which are parameters of the video coder according to [9,10].

If the GOP size is s and the intra-period is p, then the nth expected complexity of the I-frame $c_{\exp\_i}[n]$ can be calculated using Equation 1. Similarly the nth expected complexity of the P-frame $c_{\exp\_p}[n]$ can be calculated using Equation 2. The nth expected complexity of the B-frame $c_{\exp\_b}[n]$ can be calculated using Equations 3 and 4. Equations 3 and 4 are the case of non-dyadic hierarchical structure and the case of dyadic hierarchical structure, respectively.

$$\text{I-frame}: c_{\exp\_i}[n] = c[n-p] \quad \text{for} \quad (n\%p) = 0, \tag{1}$$

where $(n\%p)$ means (n modulo p).

$$\text{P-frame}: c_{\exp\_p}[n] = c[n-s] \ \text{ for } \ (n\%s) = 0, (n\%p) \neq 0 \tag{2}$$

$$\text{B-frame}(\text{non Dyadic hierarchy with 4 temporal levels}) : c_{\exp\_b}[n] = \begin{cases} c[n-4] & \text{for } (n\%3) = 0 \\ c[n+2] & \text{for } (n\%3) = 1 \\ c[n-1] & \text{for } (n\%3) = 2 \end{cases} \tag{3}$$

Equation 2 represents the non-dyadic hierarchical B-frame with four temporal levels.

$$\text{B frame (Dyadic hierachy with 4 temporal levels)} : c_{\exp\_b}[n] = \begin{cases} c\left[n-\left(\frac{s}{2^m}+1\right)\right] \text{ for } \left(n\%\frac{s}{2^m}\right) = 0, m = 1 \\ \begin{cases} c[n-\left(\frac{s}{2^m}+1\right) \text{ for } n > \frac{s}{2^{m-1}}, \left(n\%\frac{s}{2^m}\right) = 0, 2 \le m < \log_2 s \\ c[n+\left(\frac{s}{2^m}\right) \text{ for } n < \frac{s}{2^{m-1}}, \left(n\%\frac{s}{2^m}\right) = 0, 2 \le m < \log_2 s \end{cases} \\ \begin{cases} c[n-2] \text{ for } (n\%4) = 3, (n\%2) \neq 0 \\ c[n+1] \text{ for } (n\%4) = 1, (n\%2) \neq 0 \end{cases} \end{cases} \tag{4}$$

Equation 4 represents the dyadic hierarchical B-frame with four temporal levels.

$$\Delta = t[n-1] - \{t_{\text{slack}} \cdot (n-1)\}, \tag{5}$$

where Δ is the discrepancy between $t_{\text{slack}} \cdot (n-1)$, which is the time to be decoded, and $t[n-1]$, which is the time when decoding is finished. $|\Delta|$ will be controlled to converge to 0 and to be no larger than $J_{\max}$, the jitter limit.

$$f_{\exp}[n] = c_{\exp}[n]/t_{\text{slack}}, \tag{6}$$

where $c_{\exp}[n]$ and $f_{\exp}[n]$ represented by Equation 6, $c[n]$ respectively, estimate the complexity of frame n and frequency, which enables a frame to be decoded for t_{slack}. In order to lower jitter, the expected frequency $f_{\exp}[n]$ is modified according to Δ.

$$f[n] = \begin{cases} f_{\max} & \text{for } J_{\max} < \Delta \\ f_{\exp}[n] + \left(f_{\max} - f_{\exp}[n]\right)\frac{\Delta}{J_{\max}} & \text{for } 0 < \Delta \le J_{\max} \\ c_{\exp}[n]/(t_{\text{slack}} - \Delta) & \text{for otherwise} \end{cases}. \tag{7}$$

If the process is significantly behind schedule, as in Figure 2a, then the highest frequency is used, as shown in Equation 7 for $J_{\max} < \Delta$, while the process is accelerated, as in Figure 2b, in Equation 4 for $0 < \Delta \le J_{\max}$ and the process is decelerated, as shown in Figure 2c, in Equation 7 for *otherwise* to control delay variation.

Ma et al. [2] assumed that jitters would not occur, as shown Figure 2c, and thus, that spare time that could be utilized for frame decoding would always exist. However, in the present study, jitters that may occur when DVFS is utilized in the process of video decoding were considered in preparation for situations, as shown in Figure 2a, b.

In our proposed estimation method, when there is no anchor for estimation such as the start time for decoding or the changing of a channel, decoding is performed by applying maximum frequency to a processor. For each frame type, after one frame undergoes performance decoding, the previous frame can act as an anchor so that estimation can be done.

In the present study, approximated complexity is used as the reference complexity to estimate the next frame's decoding complexity. Whereas the reference complexity was determined using the number of bits applied to all compositors of the decoder in the case of [2], in the present study, the reference complexity is determined using main modules' major processing time compared to the decoding time. The approximated complexity can be relatively expressed, as shown in Figure 3, if the share of the main modules in the entire process is known. Therefore, the reference complexity can be drawn without decoding an entire video's data. As the main modules, the inverse transform, inverse quantization, interpolation, motion compensation, and loop filter can be selected, which are the decoder's most frequently used modules among the parameters in Tables two and three in [5].

On measuring the calculations of all modules of H.264/AVC using [11], it can be seen that the aforementioned main modules spend an average of 80% of the entire execution time. After getting the values of execution time of each module and the current frequency of the processor, we use them in Equation 6 to calculate the complexity of frame. This process is known as complexity profiling used in [2,10,12,13]. Since this value hardly changes with changes in diverse compression options supported by H.264/AVC, in the present study, an approximate value estimated based

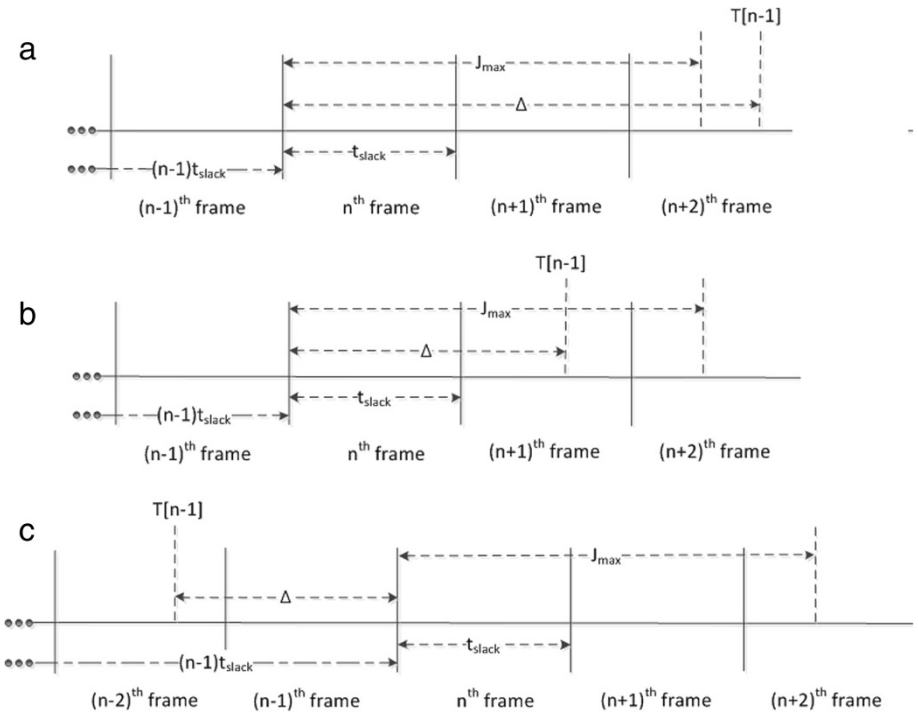

Figure 2 The concept of proposed frequency scaling. (a) The condition where the decoding should be done in maximum frequency without DVFS. **(b)** The condition where the decoding should be done in DVFS within the jitter limit. **(c)** The condition where the decoding should be done in DVFS within the slack time.

on the main modules' complexity, as shown in Figure 3, is used as the anchor complexity information.

By using the anchor complexity measured as such, the next frames can be estimated with approximated relative complexity, as shown in Figure 4, until the next scene-to-scene transition occurs and the similarity between frames disappears.

In Figure 5, the whole operating process of the proposed estimation algorithm, profiling algorithm, and frequency scaling algorithm while decoding is shown.

Below, Figure 6 shows a scatter graph of the actual complexity compared to the estimated complexity for all frames (167,857 frames) of [12], using our proposed method. Figure 6a shows a case composed of only I-frames

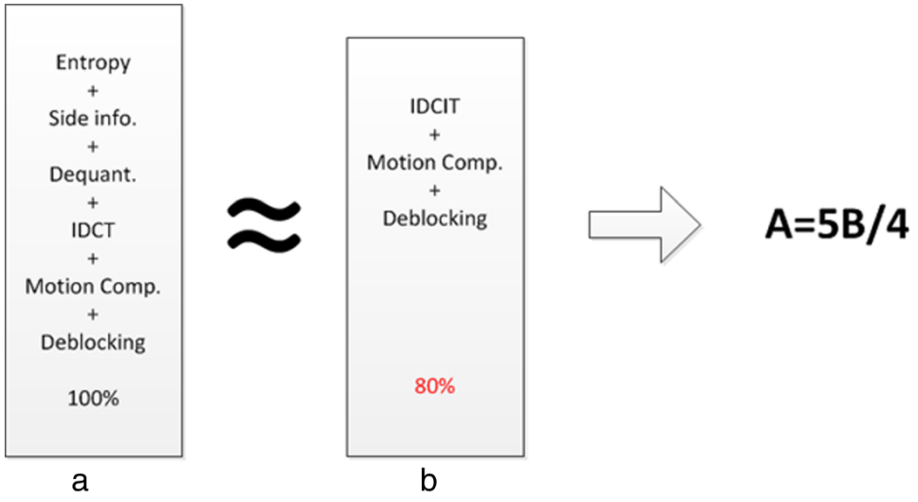

Figure 3 The concept of approximated complexity.

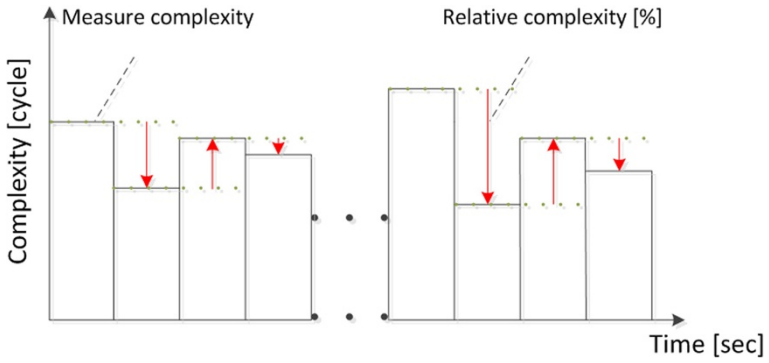

Figure 4 The concept of relative complexity. In this figure, the red arrow shows the relative complexity.

and P-frames, a combination of frame types used for general real-time video services (e.g., video conversations), and Figure 6b shows a case composed of I-frames, P-frames, and B-frames, a combination of frame types used for general non-real-time video services (e.g., videos on demand).

If many scene changes occur, the proposed complexity estimation method generates estimation errors as a result of reduced similarity between neighboring frames. If the estimation value is larger than the actual value, it generates energy wasting, and if the estimation value is smaller than the actual value, it generates delay. As shown in Figure 6, since the estimation error shows a symmetrical form between the right and left sides, the estimation error is offset. As a result, we can see that delay and power-saving efficiency have a tradeoff relationship with each other.

Figure 7 shows a logarithmic scaled negative cumulative density function (cdf) of estimation error. The cdf becomes almost 1 near the zero error value. This shows that the proposed estimation is valid, while large errors also exist even at low probability. Large errors may be caused by scene changes. In the case of Figure 7a, b for real-time services, despite the numerous scene-to-scene transitions occurring in 167,857 frames, approximately 90% of the frames show estimation errors of approximately 10%, indicating a high level of similarity between adjacent frames.

Figure 7a illustrates a case where two types of frames - I-frames and P-frames - were used to compose image sequences, and Figure 7b depicts a case where three types of frames - I-frames, P-frames, and B-frames - were used to compose image sequences.

Figures 6 and 7 show the estimation accuracy of the proposed method. From these figures, we can see that the proposed method can show the unexpected large estimation errors that may occur. Usually, the frame correlation is high because of the same frame type and being adjacent to each other. However, during cases like scene change, the frame correlation becomes low. This is the case where there occur large estimation errors. In

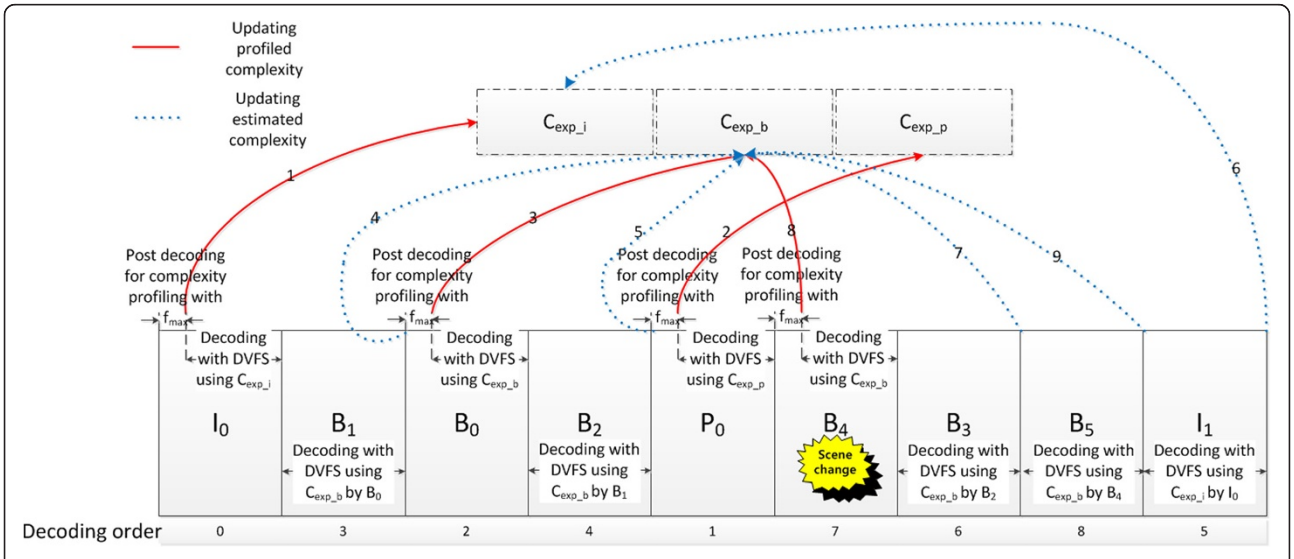

Figure 5 Decoding process with proposed complexity profiling, estimating, and frequency scaling method.

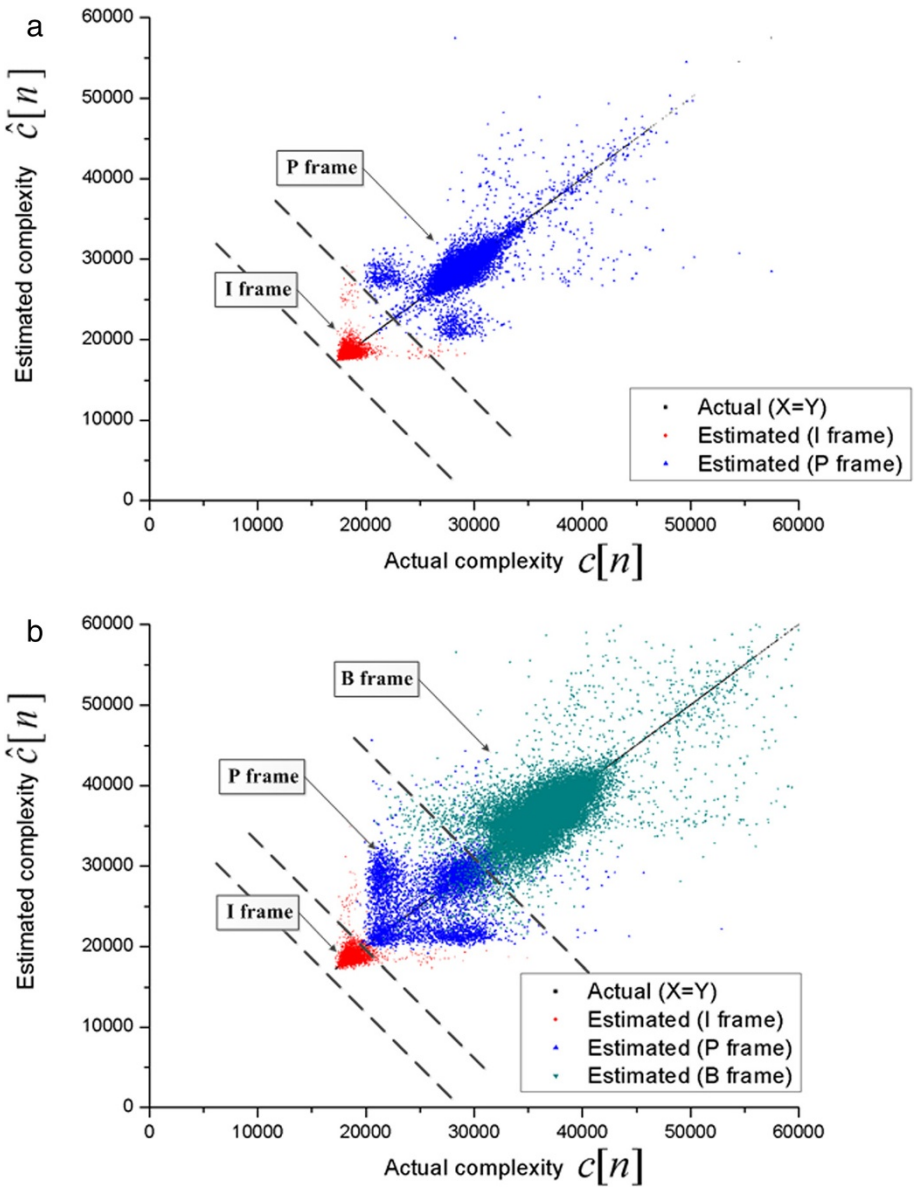

Figure 6 The variance of error of the proposed complexity estimation. (a) Composed of only I-frames and P-frames and **(b)** composed of I-frames, P-frames, and B-frames.

this case, complexity information for estimation for the next frame cannot be used, and we should update the complexity information. So, we use complexity profiling with max frequency of processor which is explained in Equation 7.

Regarding delay and power wasting, which are generated despite the offset between estimation errors while using our proposed method, we compare them by considering the accepted delay bound according to the characteristics of buffer and service [14,15]. Our proposed method for delay control, as shown in Figure 2, is performed using Equation 7.

The proposed complexity profiling is operated at the beginning of decoding or when there is scene change. The profiling is done for each frame type (i.e., I-, P- and B-frames). The proposed complexity profiler detects the execution time of modules of the decoder (i.e., inverse transform, inverse quantization, interpolation, motion compensation, and loop filter) same as that of [2,10,12]. When the correlation between the frames of the same type is high, we do not use the profiler and just estimate by execution time of the previous frame.

In [2], profiling is performed in all frames. Each module has the complexity coefficient where complexity coefficient

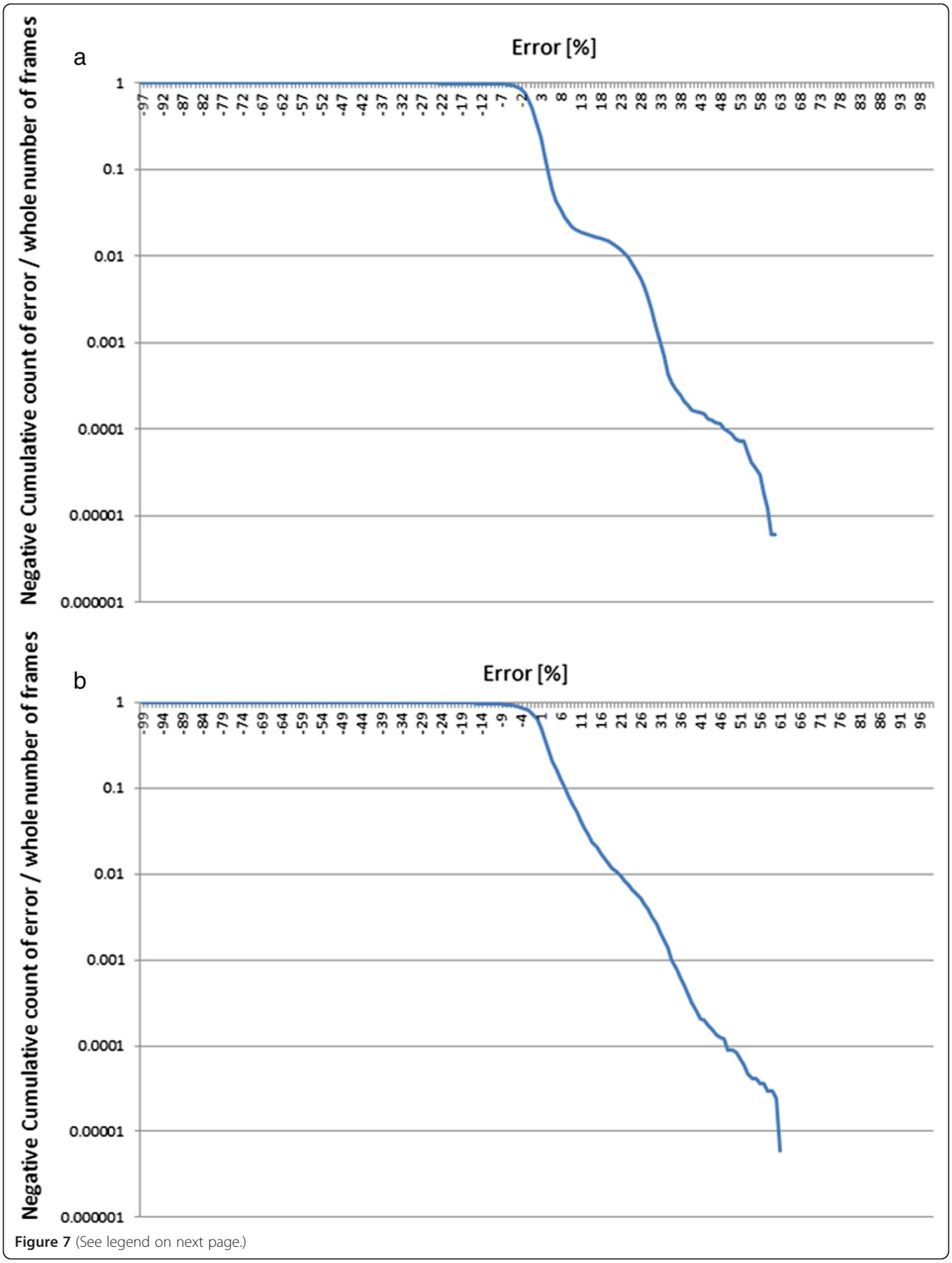

Figure 7 (See legend on next page.)

(See figure on previous page.)
Figure 7 Negative cumulative density function of complexity estimation error. (a) Where I-frames and P-frames were used to compose image sequences and **(b)** where I-frames, P-frames, and B-frames were used to compose image sequences.

means complexity of the module operating in 1 bit. During decoding, the number of bits in each module is known; thus, we can determine the total complexity of the frame.

In [10], complexity profiling is done in every frame using the execution time like that mentioned in our proposed profiling method.

In [12], complexity profiling is done in every frame using the execution cycle of each module.

In [2,10,12], complexity is predicted using different algorithms, but all of them performed post-decoding for all of the frames. However, in our proposed method, we perform post-decoding when there is scene change and on the first frame of each frame type (i.e., I, P, B). Therefore, the proposed method reduces the post-decoding of all frames. This reduction of post-decoding may reduce the access to profiling memory.

Energy consumption

DVFS is a method used to control a processor's calculation frequency to reduce the amount of energy used for calculations. Basically, when the number of calculations is large, the voltage is amplified to increase the processor's frequency. The correlation formula can be calculated using formulas 8 and 9 as with [2,16,17].

Frequency f is determined using formula 8 below as with [2,16,17].

$$f = c/t \tag{8}$$

As shown in the above formula, the frequency (Hz) of the processor is inversely proportional to calculation time t (s) and proportional to the number of calculations c (cycle). Supply voltage in complementary metal-oxide-semiconductor (CMOS) circuits can be expressed as shown in formula 9.

$$V_{dd} = \omega f^{\varphi} + \theta, \tag{9}$$

where, ω, φ, θ are coefficients determined by the underlying platform. As a representative example, Intel Pentium M1.6GHz (Intel Corporation, Santa Clara, CA, USA) of 90-nm processes has values $\theta = 0.61$, $\varphi = 1$, and $\omega = 5.6 \times 10^{-10}$, as shown in [18]. As mentioned in [16], the power (W) of CMOS circuits is expressed by formula 10 below.

$$P_{\text{total}} = P_{\text{dyn}} + P_{\text{DC}} + P_{\text{on}} \tag{10}$$

According to [16], P_{dyn} is the dynamic power consumption, which is determined by the supply voltage and the frequency, and P_{DC} is the static power consumption, which is the leakage power consumption of CMOS devices. This is determined by the supply voltage and the constant value. P_{on} is the power that maintains the 'power on' state of the processor. This is assumed as 0.1 W in the present study.

Dynamic power consumption P_{dyn} is calculated using the following formula in Watt units.

$$P_{\text{dyn}} = K_{\text{eff}} V_{dd}^2 f \tag{11}$$

Formula 11 can be expressed as formula 12 using formula 9.

$$P_{\text{dyn}} = K_{\text{eff}} (\omega f^{\varphi} + \theta)^2 f, \tag{12}$$

where K_{eff} is the effective circuit capacitance.

Dynamic calculation energy (J) is calculated as dynamic power P_{dyn} multiplied by time t, as shown in formula 13.

$$E_{\text{dyn}} = P_{\text{dyn}} t \tag{13}$$

Formula 13 can be expressed as formula 14 using formula 12, and formula 14 can be re-expressed as formula 15 using formula 8 so that the proposed estimation method can be applied to obtain dynamic energy.

$$E_{\text{dyn}} = K_{\text{eff}} (\omega f^{\varphi} + \theta)^2 f t \tag{14}$$

$$E_{\text{dyn}} = K_{\text{eff}} \left(\omega \left(\frac{c}{t} \right)^{\varphi} + \theta \right)^2 c \tag{15}$$

Static power consumption P_{DC} can be calculated in voltage units using formula 16 and Table 1 based on [17].

$$P_{\text{DC}} = V_{dd} I_{\text{subn}} + |V_{\text{bs}}| I_{\text{j}} \tag{16}$$

In formula 16, V_{bs} is the body bias voltage, and this is assumed to be –0.7 V in the present invention. I_{j} is the reverse bias junction current, which is a constant value. I_{subn} is the sub-threshold current, which is calculated in Ampere units using formula 17 below and Table 1.

$$I_{\text{subn}} = K_3 e^{K_4 V_{dd}} e^{K_5 V_{\text{bs}}} \tag{17}$$

Leakage power consumption P_{DC} and P_{on} can be obtained using the above formulas, and leakage energy

Table 1 Underlying coefficient of CMOS circuit

Constant	Value	Constant	Value
K_1	0.063	K_6	5.26×10^{-12}
K_2	0.153	K_7	−0.144
K_3	5.38×10^{-7}	I_{j}	4.8×10^{-10}
K_4	1.83	V_{bs}	−0.7
K_5	4.19		

consumption E_{DC} and E_{on} can be obtained by multiplying calculation time t.

The total calculated energy is as shown in formula 18, and it can be calculated in Joule units using the above formulas in Joule units.

$$E_{total} = E_{dyn} + E_{DC} + E_{on} \quad (18)$$

Figure 7 below shows the energy consumption experiment results when coefficients ω, φ, and θ determined by the underlying platform were set to 5.6×10^{-10}, 1, and 0.61, respectively, and a certain complexity was decided at different frequencies. On reviewing Figure 8, it can be seen that when the same complexity is decoded, the energy decreases as the frequency is reduced, and the energy increases exponentially as the frequency is increased.

Results of theoretical simulation

In this paper, we use underlying coefficients of the Intel Pentium mobile processor 1.6 GHz (Intel Corporation), and 167,857 frames of video [19] (DVD ver. 720 × 480 pixel quality) are decoded using the H.264/AVC reference software 18.3 version by [20]. Warner Bros. Entertainment's The Matrix [19] is the famous movie with intensive scene change subject to the worst scenario in our proposed method. A snapshot of [19] is shown in Figure 9.

To display raw files that are the decoder's outputs, the raw files should be transformed into RGB files. This work conducts float computations in pixel units. Therefore, no differences in the complexity between frames are caused by the transformation work in general cases where images of the same frame size are continued.

Figure 9 Snapshot of motion-intensive video 'The Matrix'.

However, even if the complexity necessary for transformation work remains constant, since DVFS is conducted, the transformation work will be affected by frequency scaling, and thus, the energy consumption necessary will vary by frame.

Complexity estimation

The first simulation compares the decoding of energy consumption of the processor between methods that use DVFS and the method that does not use DVFS [2], using complexity modeling and our proposed method. This simulation uses a science fiction, action movie [19] which is the worst simulation environment for our proposed method. We also assume that the estimation error of the comparison counterpart method [2] is 0%, which represents the best possible estimation.

As shown in Figure 10, the proposed method saves 73% more energy for decoding than the conventional non-DVFS method. This performance is almost the same as

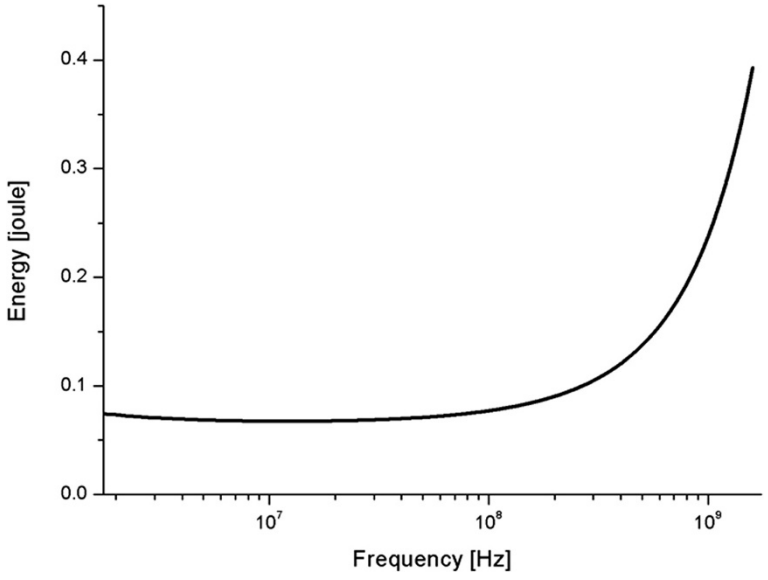

Figure 8 Decoding energy with varying frequency.

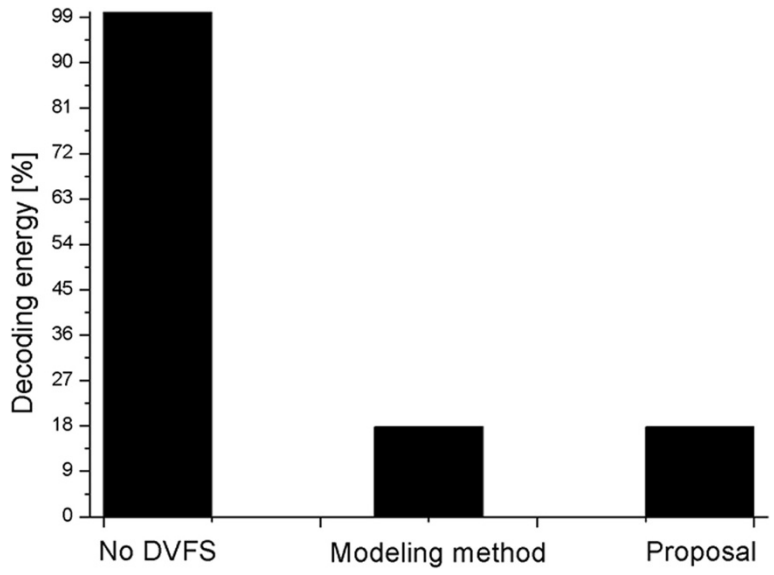

Figure 10 Performance comparison between the modeling method and proposed method.

those of previous methods [2] in which the control algorithms used are much more sophisticated than ours.

Delay control

The second simulation compares energy consumption while a method of frame drop prevention, which is generated by the estimation error of DVFS to support QoE, is used. Since Ma et al. [2] does not consider delay, in order to overcome an estimation error of 3%, DVFS shall be performed with a margin of 3% of complexity estimation. Our proposed method overcomes frame drop by setting a delay threshold as a buffer during DVFS operation to overcome the estimation error. The large buffer can overcome a large estimation error. This use of the buffer can make the proposed method adapted to real-time video service. When there is burst scene change, the total time required for post-decoding is large. So, the buffer can be used to overcome the delay caused by this time requirement.

Threshold values for delay, D_{th}, are set as three values such as 0.01 s (=10 ms), 0.1 s (=100 ms), and 1 s (=1,000 ms).

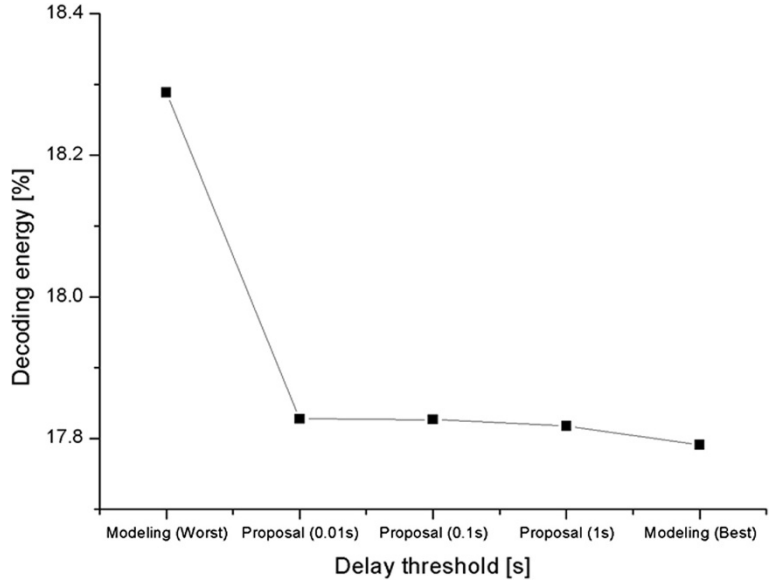

Figure 11 Performance comparison between the modeling method and proposed method with varying delay thresholds.

The results of the second simulation show that energy efficiency improves 1.94% at a threshold of 0.01 s, 1.96% at 0.1 s, and 2% at 1 s, respectively, compared to the one using [2], as shown in Figure 11.

The third simulation compares reduction of post-decoding. We assume the scene change with a larger error rate than [2,10,12]. So, we perform the post-decoding to make the estimation error rate equal to the average estimation error of [2,10,12].

In case of [2], the average estimation error rate was 3%, so our assumed estimation error rate was higher than 3%. Similarly, for [10,12], the estimation rate was assumed based on their average estimation error rate.

Figure 12 shows reduction of post-decoding of the proposed method between the direct memory access (DMA) algorithm in [2]; RE-EST, Markov-1, and LMSE algorithm in [10]; and bitstream shaping algorithm in [12,13]. The result of Figure 12 shows that when compared with DMA performance for the same accuracy, the total number of post-decoding of the proposed method was more than that of the DMA algorithm. Though it is the bad result but when the target estimation error rate was increased to the average error rate of algorithms in [10,12], the proposed method shows better performance as the total number of post-decoding goes on decreasing. In this case, the threshold value for the optimal performance is 4%. So, if the achieved estimation error rate is more than 4%, then the proposed method can reduce the number of post-decoding. Therefore, the proposed method reduces the post-decoding and hence reduces the access to profiling memory.

Table 2 Specific embedded testbed for experiment

	Description
Processor	ARM Coretex™ - A9 dual core (2 GHz)
Memory	1 GB LPDDR2
LCD	7″ 800 × 400 resolution
System software	Linux Kernel 2.6.35.7, Android 2.3.5 (Gingerbread)

Results of experimental simulation

On reviewing the results in theoretical simulation, it can be seen that quite excellent energy-saving effects can be obtained if the proposed method is used when the movie [19] is decoded using the H.264/AVC reference software 18.3 version. However, the results in the theoretical simulation only related to the energy consumed in the central processing unit (CPU) and did not consider the energy consumed in numerous background programs (e.g., display, system software, widget) as with cases where video services are used in actual mobile devices.

Energy-saving effects obtained when the method proposed in the theoretical simulation was applied to actual mobile devices were measured.

The system configuration of the embedded test board used in this experiment is shown in Table 2 below.

Linux Kernel-based android operating systems provide many forms of governors that support DVFS. The characteristics of representative governors are shown in Table 3.

Android-based mobile devices generally produced for commercial purposes use the on-demand governor that changes frequencies gradually depending on the load

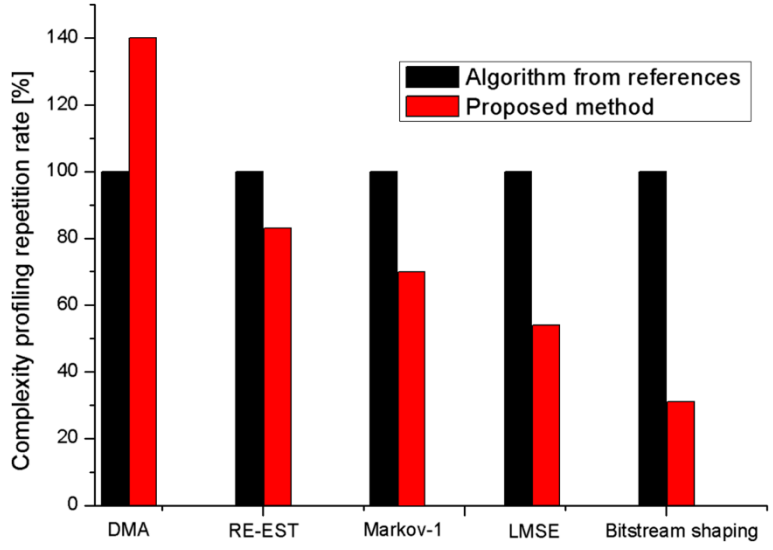

Figure 12 Reduction of post-decoding between the complexity estimation algorithm from references and proposed method.

Table 3 Characteristics of CPU governors on Linux-based operating system

Governor	Frequency scaling method
Performance	Uses only the maximum clock frequency that can be provided by the CPU
Powersave	Uses only the minimum clock frequency that can be provided by the CPU
Ondemand	Changes between the maximum/minimum frequencies depending on load conditions
Conservative	Gradually changes between the maximum ~ minimum frequencies depending on load conditions
Userspace	Uses the frequency designated by the user

conditions of the CPU. However, since gradual frequency changes are not suitable for cases where loads vary according to the kind of frames that constitute videos, the userspace governor that changes frequencies as requested by the user was used in this experiment.

Please note that the frequencies that can be designated using the userspace governor in the aforementioned system configuration of the embedded testbed are limited to 2 GHz, 1.6 GHz, 1 GHz, and 400 MHz.

Figure 13 below shows the results of experiments using the proposed method in the aforementioned experimental environment.

As with the experiment in the theoretical simulation, this experiment compared decoding energy consumption among cases where DVFS was not used, cases where the method under [2] used complexity modeling, and cases where the proposed method was used. This experiment was conducted with [12], which is a science fiction, action movie with the lowest level of similarity among frames; this experimental environment can show the worst performance of the proposed method, while the method under [2] with which the proposed method was compared was a case of best performance with an estimation error of 0%.

On reviewing Figure 13, it can be seen that the amount of energy corresponding to 91.08% of the energy consumed in cases where DVFS was not used was consumed in the case of best performance under [2], and 91.41% of the energy was consumed in the case where the proposed method was used despite the fact that no separate computation for estimation was conducted. Given these results, it can be seen that although the energy-saving efficiency was lower in this experiment than in the theoretical simulation where only the energy efficiency of the CPU was tested, since this experiment was conducted in an environment where numerous background programs operated together, as in actual use environments, when compared with [1], almost the same energy-saving effect could be identified despite the fact that the experimental environment was the worst for the proposed method similar to the results in the theoretical simulation.

Conclusion

As discussed in the 'Introduction', conventional methods that apply DVFS to video decoding use extra estimation profilers or post-decoding (i.e., calculations). These methods generate additional complexity and estimation errors. This paper proposes an estimation method that takes advantage of the correlation between video frames of the same frame type, which is in contrast to other estimation methods that uses only post-decoding. This

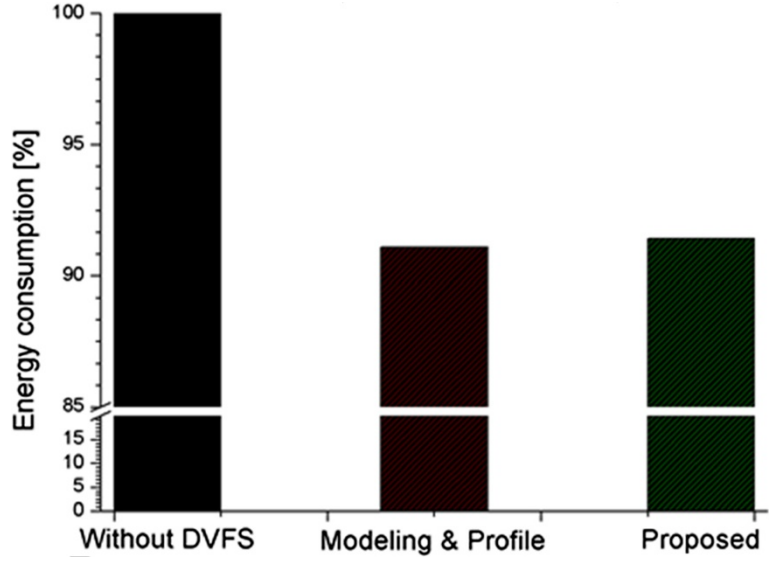

Figure 13 Performance comparison between the modeling method and proposed method on realistic testbed.

proposed method shall have larger estimation errors compared to the conventional methods because it uses a less number of post-decoding than the conventional method.

When the estimation value is larger than the actual value, our proposed method controls decoding jitter from the estimation error if the range (i.e., J_{max} in Figure 2) of estimation error is within the threshold value. On the other hand, since the conventional methods such as those of [2,3] do not consider delay, they can cause frame drop as a result of estimation errors. As shown in our simulation result, our proposed method, which uses fewer post-decoding, showed comparable performance with the performance of [2] that uses more post-decoding than the proposed method.

As a result of Figure 11, the proposed method, which sets the delay threshold (i.e., J_{max} in Figure 2), is applied according to [14]. So, the proposed method can be used on different video types and services with delay acceptance (e.g., delay threshold for video conversation is 100 ms; delay threshold for video streaming is 1 s) and thus can be used to save energy.

Therefore, our proposed method reduces the number of post-decoding unlike the conventional algorithms where post-decoding is done to all frames. It may reduce the memory access of the profiler, but memory analysis was not performed to confirm it.

In this paper, we have discussed DVFS, which only considers video decoding. However, our future research can develop further energy-saving methods that can be used in the overall video service systems such as video data receiving, memory access, video decoding, and video display.

Competing interests

The authors declare that they have no competing interests.

Acknowledgments

This research was supported by the MKE (Ministry of Knowledge Economy), Korea under the ITRC (Information Technology Research Center) support program (NIPA-2012-H0301-12-1006) supervised by the NIPA (National IT Industry Promotion Agency). This research was funded by the MSIP (Ministry of Science, ICT & Future Planning), Korea in the ICT R&D Program 2013.

Received: 28 January 2013 Accepted: 11 June 2013
Published: 4 September 2013

References

1. Z He, Y Liang, Power-rate-distortion analysis for wireless video communication under energy constraints. IEEE Trans. Circuits Syst. Video Technol. **15**(5), 645–658 (2005)
2. Z Ma, H Hu, Y Wang, On complexity modeling of H.264/AVC video decoding and its application for energy efficient decoding. IEEE Trans. Multimedia **13**, 1240–1255 (2011)
3. J Cho, I Cho, A combined approach for QoS-guaranteed and low-power video decoding. IEEE Trans. Consumer Elec. **57**, 651–657 (2011)
4. M van der Schaar, Y Andreopoulos, Rate-distortion-complexity modeling for network and receiver aware adaption. IEEE Trans. Multimedia **7**(3), 471–479 (2005)
5. M Horowitz, A Joch, H.264AVC baseline profile decoder complexity analysis. IEEE Trans. Circuits Syst. Video Technol. **13**(7), 704–716 (2003)
6. Z Ma, Z Zhang, Complexity modeling of H.264 entropy decoding, in *Proc. ICIP, San Diego, October 2008*
7. K Choi, W Cheng, Frame-based dynamic voltage and frequency scaling for an MPEG decoder, in *Proc. ICCAD, San Jose, November 2002*
8. Z Lu, J Lach, Reducing multimedia decode power using feedback control, in *Proceedings of International Conference on Computer Design, San Jose, October 2003*
9. T Wiegand, GJ Sulivan, Overview of the H.264/AVC video coding standard. IEEE Trans. Circuits Syst. Video Technol. **13**(7), 560–576 (2003)
10. N Kontorinis, Y Andreopoulos, Statistical framework for video decoding complexity modeling and prediction. IEEE Trans. Circuits Syst. Video Technol. **19**(7), 1000–1013 (2009)
11. Intel Parallel Studio, [Online], Available: software.intel.com/en-us/intel-parallel-studio-xe. Accessed 8 February 2012
12. E Akyol, M Scharr, Complexity model based proactive dynamic voltage scaling for video decoding systems. IEEE Trans. Multimedia **9**(7), 1475–1492 (2007)
13. Y Andreopoulos, M van der Scharr, Complexity-constrained video bitstream shaping. IEEE Trans. Signal Process. **55**(5), 1967–1974 (2007)
14. Nortel Networks, *QoS performance requirements for UMTS, 3GPP TSG SA S1* (Copenhagen, 1999)
15. S Wee, W Tan, Optimized video streaming for networks with varying delay, in *Proc. ICME, Lausanne, August 2002*
16. R Jejurikar, C Pereira, Leakage aware dynamic voltage scaling for real-time embedded systems, in *Proc. DAC, San Diego, July 2004*
17. S Martin, K Flautner, Combined dynamic voltage scaling and adaptive body biasing for optimal power consumption in microprocessors under dynamic workloads, in *Proc. ICCAD, San Jose, November 2002*
18. Intel Pentium Mobile Processor, [Online]. Available: http://www.intel.com/content/www/us/en/intelligent-systems/previous-generation/embedded-pentium-m.html. Accessed 15 January 2012
19. Warner Bros. Entertainment, *The Matrix* (United States, 1999)
20. Dolby Laboratories Inc., Fraunhofer-Institute HHI, Microsoft Corporation, *H.264/MPEG-4 AVC Reference Software Manual*. ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6, JVT-X072, Geneva, 29 June–5 July 2007

doi:10.1186/1687-3963-2013-13
Cite this article as: Yun *et al.*: **Dynamic voltage and frequency scaling over delay-constrained mobile multimedia service using approximated relative complexity estimation.** *EURASIP Journal on Embedded Systems* 2013 **2013**:13.