Systems and Computers in Japan, Vol. 36, No. 6, 2005
Translated from Denshi Joho Tsushin Gakkai Ronbunshi, Vol. J87-D-I, No. 4, April 2004, pp. 452–461

Frequency–Voltage Cooperative CPU Power Control: A Design Rule and Its Application by Feedback Prediction

Keisuke Toyama,[1] Satoshi Misaka,[1] Kazuo Aisaka,[1] Toshiyuki Aritsuka,[1] Kunio Uchiyama,[1] Koichiro Ishibashi,[1] Hiroshi Kawaguchi,[2] and Takayasu Sakurai[2]

[1]Central Research Laboratory, Hitachi, Ltd., Tokyo, 185-8601 Japan

[2]Center for Collaborative Research, University of Tokyo, Tokyo, 153-8904 Japan

SUMMARY

Frequency–voltage cooperative power control (FVC) is a powerful method to reduce the CPU power consumption of a program during execution, because it utilizes the information on the software workload dynamically. In this paper, we first show through a mathematical analysis the design rule to determine the necessary frequencies and its effect. Then we show experimental results of implementing an FVC with a feedback algorithm on MPEG-4 video and MP3 audio decoders with two sets of frequency and voltage. The FVC gave a 72% reduction in CPU power consumption during execution. In addition, we show a "Cool-Start" method that begins the FVC at a lower frequency and improves the power reduction effect. © 2005 Wiley Periodicals, Inc. Syst Comp Jpn, 36(6): 39–48, 2005; Published online in Wiley InterScience (www.interscience.wiley.com). DOI 10.1002/scj.20263

Key words: frequency–voltage cooperative power control (FVC); feedback control; MPEG-4; MP3.

1. Introduction

As the integration scale grows, power consumption has become one of the major problems in VLSI design, especially in chips for mobile applications. The CPU power consumption is a major factor, and many hardware and software techniques have been introduced to reduce it. Frequency–voltage cooperative power control (FVC) is one such technique, which lowers both clock frequency (F) and power supply voltage (V) when a lower speed is enough during an execution [1, 2]. This technique is effective because a lower F and a lower square of V contribute to reducing the power; it offers a large reduction in CPU power consumption during program execution.

FVC is applied to a whole program, task, etc. When applied to fragments of a program (slices), it utilizes the program's characteristics to control F and V. Here, we are mainly concerned about its application to program slices [3, 4].

From a mathematical analysis, we have derived a design rule that gives the proper value of F and V [5, 6] and have applied the rule to an MPEG-4 decoder. Furthermore, we have verified that the rule is valid as expected for an MP3 decoder.

2. Background on Frequency–Voltage Power Control

2.1. Basic concepts

Power consumption P of a CPU is expressed as

$$P = \alpha \times C \times F \times V^2$$

where α is the switching probability, C the capacity, F the clock frequency, and V the core voltage. Any reduction in these factors will lead to lower power consumption, and especially, a reduction in V will have a large effect. The fundamental power consumption characteristics of multivoltage controlled CPUs when F and V are varied are as follows [2]:

- The ideal voltage V_{ideal} that executes process while consuming the minimum power within a specified deadline time exists.
- When a CPU is supplied with multiple discrete core voltages, the two adjacent voltages to the V_{ideal} have the minimum power consumption.

To apply the above characteristics to actual system designs, the following issues need to be considered.

(1) Although CPUs have lower limit voltages to run, clock frequency can be decreased at the lowest voltage.

(2) The F–V relation should be expressed because voltage is controlled in terms of throughput.

Thus, we analyzed the problem in relation to the operation clock frequency F and the consumed power P with F as the main control parameter.

2.2. Modeling of frequency–power relation

We first define the frequency–power (F–P) characteristics of a CPU. Figure 1 shows a typical F–P characteristic curve showing the minimum power consumption (P) of a chip when its operation clock frequency (F) is given. Although the F–P curve depends on many design parameters in an actual LSI design, we assume that the curve can be modeled by

$$P = \begin{cases} kF^{\gamma} & (F \geq F_m) \\ k'F & (F \leq F_m) \end{cases} \quad (1)$$

where k and k' are proportionality constants.

The F–P curve by Eq. (1) consists of two parts: the left half is a straight line beginning at the origin, and the right half is an algebraic curve of order γ. Both halves meet at the point defined as frequency F_m, which is the highest at the minimum operation voltage (V_{min}) in the chip. In the left half, the power is linearly proportional to frequency, meaning that the operation voltage is constant. In the right half, the operation voltage increases in accordance with frequency showing a curve raised to the power γ. Though there is no physical evidence that supports the use of Eq. (1), from experience, we consider it accurate enough for modeling a real LSI.

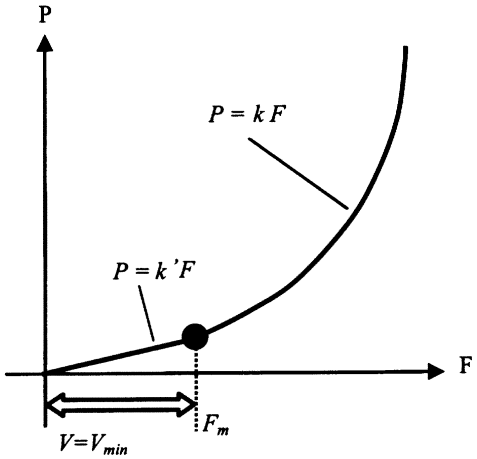


Fig. 1. Modeling of F–P relation.

3. Discrete FVC

3.1. F–P relationship with discrete frequencies

The F–P relationship (1) assumes that F and V of a CPU can be changed continuously. Implementing this feature requires a large and complicated circuit, and is not practical in LSIs for commercial use. We thus assume that the clock frequencies of the clock generator are defined discretely, namely, F_1, F_2, and F_3 in Fig. 2, and that the corresponding minimum voltages are provided. Accordingly, the F–P curve is represented by the dotted line in Fig. 2, that is, approximating the curve piecewise-linearly with nodes at the defined frequencies [1, 2]. We can find out the influence of the discrete frequency to power consumption by evaluating the difference between the dotted line and the original F–P curve. Below we demonstrate a sufficient reduction in power consumption for discrete frequencies.

Suppose that a CPU requires the ideal minimum operation frequency F_i to run a specific program. If the frequency (and voltage) can be set arbitrarily, the program can be executed with a power consumption P_i in Fig. 2. P_i is the lowest power consumption to run the program. Under the discrete frequency condition, the power consumption corresponding to F_i increases to P_r which is on the dotted line, because we can get a power reduction proportional to the frequency decrease by realizing F_i on average with a combination of F_1 and F_2. Thus, the relative power loss for discrete frequencies with respect to the minimum power

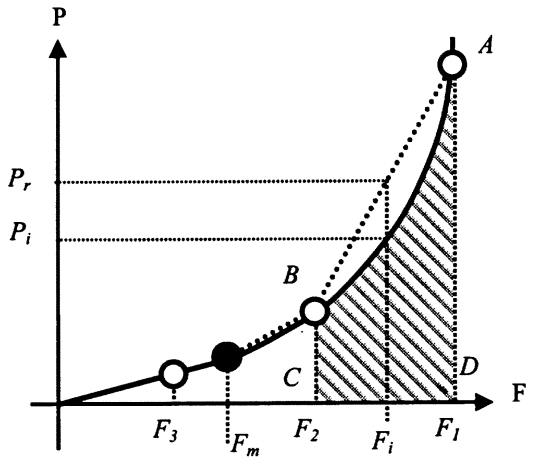


Fig. 2. Influence of discrete frequency.

consumption for continuous frequencies can be obtained by calculating the ratio of P_r to P_i.

3.2. Power loss estimation

We estimate realistic values for the power loss above by introducing some more definitions. First, the highest frequency F_1 in Fig. 2 for the CPU is given a priori. Second, we introduce a parameter β representing the ratio of adjacent frequencies such as F_1/F_2, F_2/F_3, and so on in the power loss expressions.

3.2.1. Maximum power loss

As mentioned above, the discrete-frequency power loss at a certain frequency F_i, which is relative to the minimum value when frequency changes continuously, equals P_r/P_i in Fig. 2. This value is calculated as follows: If F_m is not in the span of the polygonal line (the case in which F_i exists between F_2 and F_1 in Fig. 2), by substituting F_1 and F_2 in the expression for line AB, we get

$$P_r = \frac{kF_1^\gamma - kF_2^\gamma}{F_1 - F_2} + \frac{F_1 \cdot kF_2^\gamma - F_2 \cdot kF_1^\gamma}{F_1 - F_2}$$

As P_i equals kF_i^γ, we substitute F_2/F_i with α and F_1/F_2 with β, to get

$$\frac{P_r}{P_i} = \frac{\alpha^\gamma(\beta - \beta^\gamma) + \alpha^{\gamma-1}(\beta^\gamma - 1)}{\beta - 1} \qquad (2a)$$

If F_m is in the span of the polygonal line (the case F_i exists between F_3 and F_2 in Fig. 2), we have

$$\frac{P_r}{P_i} = \frac{\alpha^\gamma(K\beta - \beta^\gamma) + \alpha^{\gamma-1}(\beta^\gamma - K)}{\beta - 1} \qquad (2b)$$

where $\alpha = F_3/F_i$, $\beta = F_2/F_3$, $K = (F_m/F_3)^{\gamma-1}$. In both expressions, F_i is represented by the relative value α with respect to F_2 or F_3. Equation (2a) represents the case where F_m is in the lowest end of the interval in Eq. (2b), and thus $K = 1$. We examine only Eq. (2b) hereafter.

As Eq. (2b) represents a power loss for a specific F_i, its maximum value in the interval $[F_3, F_2]$ is obtained by analyzing the increase and decrease relative to a change in F_i, that is, α in Eq. (2b). The domain of α is $[1, \beta]$, which corresponds to the domain of F_i of $[F_3, F_2]$. The maximum value in this interval is obtained by differentiating Eq. (2b) by α,

$$\alpha = (\gamma - 1)(\beta^\gamma - K)/\gamma\beta(\beta^{\gamma-1} - K) \qquad (3)$$

By substituting Eq. (3) in Eq. (2b), the maximum power loss for a specific β and K is obtained.

3.2.2. Average power loss

In real applications, programs do not run with only the F_i that gives the maximum power loss. We evaluate the average power loss when a program is executed in fragments (we call them slices) where various F_i exist.

Suppose that the CPU executes n program slices in the considered time frame, and the minimum frequencies for their executions are $F_i(1), F_i(2), \ldots, F_i(n)$, respectively. The necessary power that completes the whole program is the sum of the powers for each program slice, and the average power loss is

$$\frac{\sum^n P_r(F_i(.))}{\sum^n P_i(F_i(.))} \qquad (4)$$

The denominator denotes the power if the frequency can be changed continuously, and the numerator denotes the sum if discrete frequencies are to be used. If we assume n is large enough and the $F_i(.)$ are distributed uniformly in the interval $[F_2, F_1]$, the denominator corresponds to the hatched area in Fig. 2 and the numerator to the trapezoid ABCD. Thus, we can get the following:

$$\frac{\sum P_r}{\sum P_i} = \frac{(\gamma + 1)(\beta^\gamma + 1)(\beta - 1)}{2(\beta^{\gamma+1} - 1)} \qquad (5a)$$

The above equation is for the interval $[F_2, F_1]$ in Fig. 2, and the equation below is for $[F_3, F_2]$:

$$\frac{\sum P_r}{\sum P_i} = \frac{\rho^{\gamma-1}(\gamma + 1)\left(1 + \frac{1}{\beta}\rho^{\gamma-1}\right)\left(1 - \frac{1}{\beta}\right)}{(\gamma + 1)\left(\frac{1}{\rho^2} - \frac{1}{\beta^2}\right) + 2\rho^{\gamma-1}\left(1 - \frac{1}{\rho^{\gamma+1}}\right)} \qquad (5b)$$

where $\rho = F_2/F_m$. Equation (5a) is the case where F_m is at the lower end of the interval, and this reduces to $\rho = \beta$.

That is the basic execution model for the FVC; it divides a program's execution into intervals (slices) and reduces the CPU frequency for each interval according to its required performance which has the effect of reducing CPU power while preserving the execution performance as a whole.

3.2.3. Examples of numerical result

(1) F_m fixed

Table 1 lists the calculated power loss in accordance with β and γ of the *F–P* relation. In each column, the upper row lists the average power loss from Eq. (5b), and the lower row lists the maximum power loss from Eqs. (2b) and (3). The number is the loss ratio: 0 means no power degradation and 10 means 10% more power consumed than the ideal value. Table 1(a) is for the case where F_m is not contained in the evaluated interval, and Table 1(b) is where F_m is contained in it. Typical values at "the junction point" where $F_m = (F_2 + F_3)/2$ are also presented.

An experimental measurement on a real CPU yielded approximately $\gamma = 2.0$. A frequency altering mechanism enabled us to control the CPU easily and efficiently with $\beta = 2.0$ meaning one-half of the maximum frequency. This is considered to be a typical case. As shown in Table 1, in the case of (a) with $\beta = 2.0$, for a typical example of $\gamma = 2.0$, the average power loss is 7% and the maximum power loss is 13%. Although these values are a little worse in (b), the average power loss remains 11%.

(2) F_m varies

Figure 3 shows the relative power loss when F_m varies with $\beta = F_1/F_2$ set to 2.0. F_m is represented by the parameter q:

Table 1. Example value for power loss ratio

(a) Interval $[F_2, F_1]$

	1.5	2.0	2.5	3.0
1.5	1	3	5	8
	2	4	8	13
2.0	3	**7**	13	20
	5	**13**	24	41
3.0	6	15	27	40
	12	33	69	126

(b) Interval $[F_3, F_2]$

$F_m = (F_3+F_2)/2$

	1.5	2.0	2.5	3.0
1.5	3	6	9	13
	6	12	19	26
2.0	5	**11**	17	24
	10	**22**	36	52
3.0	9	18	28	39
	17	38	63	94

Upper row Average loss (%)
Lower row Maximum loss (%)

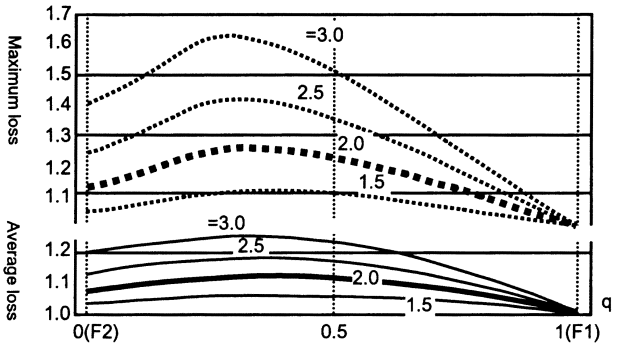


Fig. 3. Power loss estimation with F_m altered.

$$q = (F_m - F_2)/(F_1 - F_2)$$

q denotes the relative position of F_m in the frequency range $[F_2, F_1]$ under analysis. $q = 0$ is $F_m = F_2$ and $q = 1$ is $F_m = F_1$, which means F_m is at either the lowest or highest end of the interval, respectively. The relative power loss is 1.0, that is, no loss, at $q = 1$. This shows that FVC has no effect because the CPU must run at the minimum core voltage in this frequency range.

As shown in the figure, power loss reaches a maximum at $q = 0.3$ to 0.4, and although a little worse than typical, the maximum value should have no critical effect on the LSI design around $\gamma = 2.0$, which is dominant in real applications.

3.3. Design rule

Based on the above analysis and evaluation, we developed a design rule that determines *F* and *V* so as to get the most appropriate power consumption according to the operation frequencies the program requires.

For typical chips with $\gamma = 2.0$, Section 3.2.3 indicates that the power loss against optimal reduction is around 10% when two adjacent frequencies are determined at $\beta = 2$. Consequently, one-half of the highest frequency and its corresponding core voltage will be able to reduce the CPU power consumption during program execution by about a 10% loss against the ideal case. If the core voltage can be further reduced, the next frequency chosen is one-half of the previous one (and the corresponding voltage). Dividing by 2 determines the set of frequencies that realizes power reduction of around 10% loss against the ideal case. The rule for determining the frequencies is shown below.

In designing an FVC system when the highest frequency F_1 is given, the lower frequencies selected are

$$F_2 = 1/2\ F_1,\ F_3 = 1/4\ F_1,\ F_4 = 1/8\ F_1, \ldots$$

Programs should be executed with these sets of frequencies, which simulate ideal execution with the minimum frequency F_i.

For actual chips, the F–P relationship is a straight line through the origin in the range of $V < V_{min}$ as in Fig. 1. The terminal frequency $F = 0$ and at least another frequency are enough for the range. This means that if the selected frequency by the rule above is lower than or equal to F_m, no other frequencies are necessary and the application of the rule terminates. FVC is possible with the frequencies obtained by the rule so far. When there is time to spare for the execution, the frequency should be 0. As the core voltage of recent chips has become lower, it is more likely that the rule will terminate at $F_2 = 1/2F_1$ because the voltage cannot be lowered any further. In other words, a single frequency should be enough for addition in practical circumstances.

The above discussion presumes the availability of the 0 frequency, that is, stopping a program execution with no power consumption. For this purpose many CPUs have a low-power mode such as standby or sleep which is equivalent to halting the clock.

3.4. Application to real CPUs

We applied the design rule in the previous section to actual CPUs.

(1) SH-Mobile [7]

Here, the highest operation frequency F_1 and the corresponding core voltage are 120 Mhz and 1.5 V, respectively. γ is obtained as 1.9. The selected frequency F_2 is 60 Mhz and the core voltage is 1.25 V according to the rule. Although this voltage is outside of the operation guaranteed range of the chip's specification, the experiments showed prompt operation at 1.25 V. We thus identified 1.25 V as V_{min}. As F_2 = 60 Mhz is lower than F_m, the application of the rule terminates. We get an FVC system with two frequencies of 120 and 60 Mhz, and a "sleep mode" which corresponds to $F = 0$.

(2) Crusoe [8, 9]

The Crusoe chip of Transmeta Corporation is the first CPU with a built-in FVC function. It offers five to seven frequency–voltage sets [8, 9]. We tested a five-set chip (Table 2). F_1 = 933 MHz and F_2 = 467 MHz are required, and the voltage for 467 MHz is 1.35 V. F_2 does not reach F_m, so $F_3 = F_2/2$ = 233 MHz is added. At F_3, 0.9 V is the lowest voltage, after which the process terminates. FVC can be performed with three frequency–voltage sets (and sleep mode).

Table 2. Presumed frequency on Crusoe

Original design [8]		This design	
F [MHz]	V [V]	F [MHz]	V [V]
933	1.35	933	1.35
800	1.25		
667	1.2		
533	1.1	467	1.1
300	0.9	233	0.9

4. System Application

4.1. Feedback FVC

We applied our design rule to an actual CPU and the programs running on it. From the obtained frequency and voltage set, the ideal frequency F_i corresponding to program workload was approximated to realize a power reduction. We needed to know the workload information of the program. As program workloads are altered dynamically, the changes in frequencies are also dynamic. Accurate predictions of workload thus have to be fed to the processor in some way. Feedback FVC predicts program workloads by comparing the actual elapsed time and the scheduled time calculated beforehand on the basis of the program's real-time property [3].

For feedback FVC to work, we assume that the following about the program it controls is known:

(1) The deadline of the program completion
(2) The execution steps or time for program completion

The control shown in Fig. 4 for the MPEG decoder as an example works under these conditions.

Suppose that the MPEG decoder starts decoding a certain picture frame. The decoding process must end by the time that frame is displayed (i.e., the deadline). To complete a decoding, the initial setting and macro block processing are necessary. A macro block (MB) is a 16×16 pixel subpartition of moving picture data. The QCIF image for MPEG has 99 MBs. As the flowchart in Fig. 4 (left) shows, the processing time is controlled with the checkpoints placed in the initialization and at each start of 99 MB processes. That is, each MB process is taken to be the slice for FVC in Section 3.2.2. Figure 4 (right) shows the progress chart.

Given extremely heavy moving picture data, the decoder consumes the Worst Case Execution Time (WCET) for all MBs, as shown by the double-dashed line in the

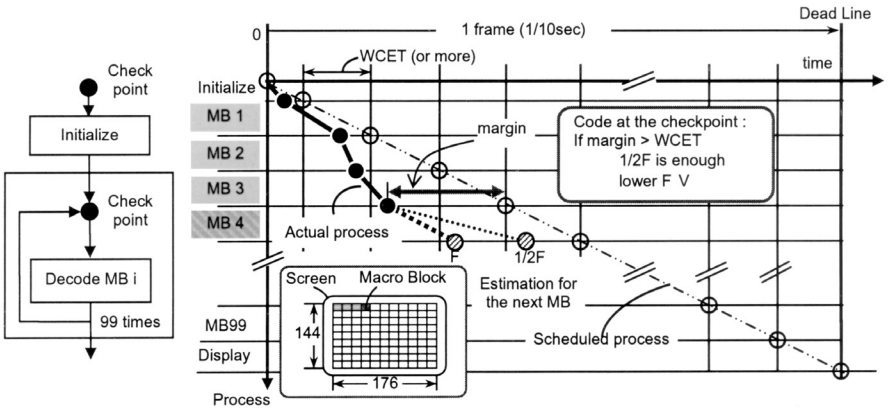


Fig. 4. Decoding process of MPEG-4.

figure. The last MB is decoded at the rightmost position of the diagonal. As can be seen, the decoding time does not exceed the deadline. The system is designed so that the whole process ends within the time frame, even in the worst case. In most cases, the workload is not so heavy and the process follows the thick line in the figure. Each decoding process terminates before WCET; thus, the time margin between the double-dashed line and the thick one can be calculated when starting a decoding step. When the margin is greater than WCET, the next step can use twice as much time than usual without delaying the worst case for the whole frame, and thus, we can judge that it is safe to drop the operation frequency to 1/2. In the same way, when the margin is twice, 3 times, . . . , n times larger than WCET, the frequency can be dropped safely to 1/3, 1/4, . . . , $1/(n+1)$. When the margin is small, on the other hand, the frequency should be increased.

With this method, we can evaluate the progress of the program against the worst case. When there is no time margin, it is possible to raise the frequency in order to make it on time for the schedule even in the worst case. The frequency is controlled precisely by evaluating the time margin from the elapsed time at that time. We call this feedback control.

4.1.1. Application to MPEG-4 decoder

We implemented a prototype FVC system based on the above design rule and evaluated its behavior. From the result of Section 3.4(1), the operation uses 120 MHz/1.5 V and 60 MHz/1.25 V. When there is still a time margin after low-frequency operation, the CPU goes into sleep mode. The target decoder is implemented only with software. The checkpoints for the time margin are inserted into the initializing process and each macro block process. The program is divided into 100 slices. The time margin is estimated at each slice and when the margin is large enough, the frequency is set to Low; otherwise it is set High. The time margin is calculated under the condition that the frame rate of the moving pictures is 10 fps and WCET for the whole frame is 100 ms.

When a typical MPEG-4 moving picture stream (average workload: 31.0%) is input, the system behaves as in Table 3. Here, the average workload is relative to the maximum performance of the CPU needed to properly decode the pictures. In the table, 10 out of 71 moving picture frames, for example, can be decoded using around 10 to 20% of the CPU's performance. In these frames, the frequency changes once on average. 8.76% of the running time is in High operation (120 MHz/1.5 V), 26.06% is in Low (60 MHz/1.25 V), and 65.18% is in sleep mode. The relative power in the bottom line is calculated by multiplying the ratio of the mode and its appearance probability. The power consumption is calculated to be 26%. In this simulation of 71-frame data, we calculated the ratio by assuming

Table 3. Simulated power reduction

relative workload	#frames	#FV change	High mode (%)	Low mode (%)	Sleep (%)
0 - 10%	0	0	0	0	0
- 20	10	1.00	8.76	26.06	65.18
- 30	31	1.45	9.40	41.10	49.50
- 40	22	1.36	13.24	50.42	36.34
- 50	7	1.00	16.51	56.30	27.19
- 60	0	0	0	0	0
- 70	0	0	0	0	0
- 80	1	1.00	71.50	14.00	14.50
- 90	0	0	0	0	0
- 100	0	0	0	0	0
relative power (%)			11.67	14.50	0

that Low operation consumes one-third of the power of High operation and that zero power is consumed in sleep mode.

4.2. Application to MP3 decoder

We also verified the effect of FVC on MP3 by installing almost the same process as for MPEG-4. While the macro blocks were the basis for the slices in the MPEG-4, we chose MP3's starting point for L/R channels and the granule processes to insert the checkpoints. There were eight slices. As in the MPEG-4 case, frequency and voltage were set to 120 MHz/1.5 V and 60 MHz/1.25 V. When there was enough time margin, the CPU would go into sleep mode.

The calculation of the time margin and the judgment were the same as in the MPEG-4 case. The workload for MP3 is low (around 20%), and its fluctuation is smaller than in MPEG-4. The difference between maximum and minimum workload is 3:2 in our trial data. We set WCET for the whole frame as 26 ms, considering one frame of sound output.

The power evaluation was done with the trace data from the experiment described in the following section. Table 4 shows data for 199 sound output frames. Most of the frames (86%) change frequency and voltage only once in the frame. From those data, the power consumption for MP3 was calculated to have been reduced to 15.3% compared with that without FVC.

4.3. Experimental results

As the target system of an implementation of FVC, we used the system board (SolutionEngine) embedded with Hitachi's SH-Mobile1 and configured the system in Fig. 5. In the figure, ① shows a core voltage change circuit offering two voltages. The frequencies are changed by modifying the ratio of the PLL output. The controlled frequencies were the maximum one and half of it (120 MHz/60 MHz), and the voltages were the lowest ones (1.5 V/1.25 V) that could sustain those frequencies. System calls to set the frequency and voltage cooperatively were implemented as

Table 4. Simulated power reduction by trace data

Appearance ratio (%)	#FV change	High mode (%)	Low mode (%)	Sleep (%)
85.9	1	2.45	34.3	63.25
1.0	2	14.36	85.7	0
13.1	3	7.35	44.1	48.55

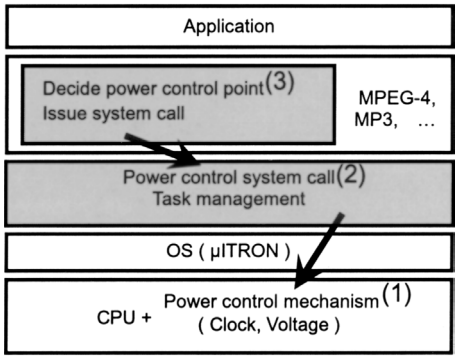


Fig. 5. System architecture.

SetFV_Low (1/2 frequency) and SetFV_High (maximum frequency) in ②. The MPEG-4 video decoder and the MP3 decoder were executed synchronously on μITRON. The "decide power control" point of ③ in Fig. 5 corresponds to the checkpoint in Fig. 4.

The number of *F* and *V* changes in 188 MPEG-4 frames is shown in Table 5(a). For a moving picture decode of Sub-QCIF at 10 fps, 74% of the frames change *F* and *V* once. Most of the execution is in Low mode. There are frames that change *F* and *V* more than 10 times. After running at low frequency, if the time margin is not large enough, the feedback control increases the frequency. When the workload is around 50%, this process can be repeated because a single Low mode execution would not leave a large enough time margin. The number of *F* and *V* changes in MP3 199 frames is shown in Table 5(b). In MP3, a single *F* and *V* change is enough for most of the frames, and the change takes place in the second frame. Indeed, execution at 60 Mhz is enough for this decoder.

Figure 6 shows the waveforms of voltage and current during moving picture and sound decoding. Average power consumption with FVC [in (b)] is 65 mW, which is a reduction to 28% compared with that of 230 Mw without FVC [in (a)].

4.4. Cool-Start method

So far, we have assumed that the initial frequency/voltage is in High mode at the beginning of each

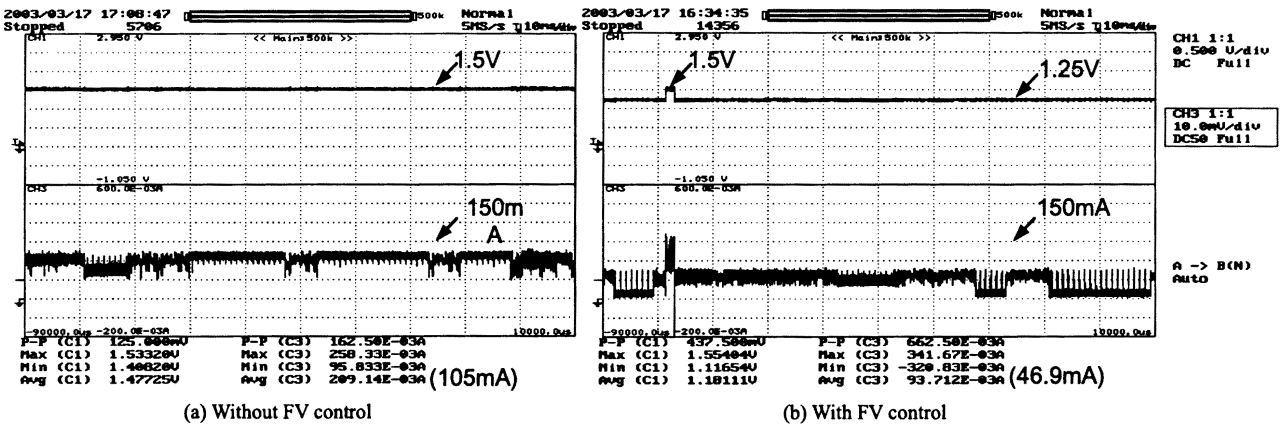
Table 5. No. of FV changes in frame execution

#FV change	1	3	5	7	9	11	13	15	-	27
#frames	139	27	12	2	2	3	1	1	0	1

(a) MPEG-4

#FV change	1	2	3
#frames	171	2	26

(b) MP3



(a) Without FV control (b) With FV control

Fig. 6. Voltage and current waveforms during execution.

frame of MPEG-4 and MP3. If the CPU's frequency is high enough for the estimated workload (by 10% as shown below), we can start each frame in Low mode (i.e., "Cool-Start"). Assuming that each slice executes in the same time, if the FVC has a 1% larger margin for MPEG-4 or 13% for MP3 in the FVC, execution within WCET is guaranteed by the feedback control after the second slice. MP3 in our experiment always runs in Low mode using this control. Supposing that the power consumption in Low mode is one-third of High mode, MP3 gets a 5.9% power reduction over that without Cool-Start.

Table 5 shows that the first F and V change in a frame occurs at the 19th slice on average. If Cool-Start reduces this by one slice, we can expect a 1.4% power reduction supposing that only one F and V change is dominant. Cool-Start is an efficient method if the number of slices is small as it is in MP3.

5. Conclusion

We have developed a frequency selection rule for our FVC by performing an analytical power loss estimation. Based on the result, we found that the set of frequencies provided by the one-half rule is sufficient for practical purposes. In addition, we clarified the termination condition for applying the steps of the one-half rule. We have also shown for recent low-voltage chips that the addition of only one voltage corresponding to one-half of the maximum frequency is sufficient because a narrower voltage range does not allow us to decrease the voltage further.

We have implemented an FVC system for MPEG-4 and MP3 based on the rule. Both simulation and actual executions showed that the power consumption of an MPEG-4 decoder could be reduced to 26% and that the power consumption of synchronously running MPEG-4 and MP3 moving picture/sound decoders could be reduced to 28% overall. This indicates the FVC is practically efficient.

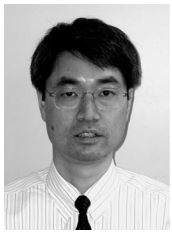
Acknowledgments. The authors appreciate the help and advice of Mr. Hideaki Chaki, Mr. Tsuguji Tachiuchi, Mr. Kazuo Kondoh, and Mr. Koji Tokunaga of Renesas Technology Corporation, and Mr. Ikuo Shibata, Mr. Keiji Moki, Mr. Hidekazu Mishima, and Mr. Shoei Tamura of Hitachi ULSI Systems.

REFERENCES

1. Chandrakasan A, Gutnik V, Xanthopoulos T. Data driven signal processing: An approach for energy efficient computing. ISLPED, p 347–352, 1996.
2. Ishihara T, Yasuura H. Voltage scheduling problem for dynamically variable voltage processors. ISLPED, p 197–202, 1998.
3. Lee S, Sakurai T. Run-time voltage hopping for low-power real-time systems. DAC, p 806–809, 2000.
4. Aisaka K, Aritsuka T, Misaka S, Toyama K, Uchiyama K, Ishibashi K, Kawaguchi H, Sakurai T. Design rule for frequency–voltage cooperative power control and its application to an MPEG-4 decoder. 2002 Symp VLSI Circuits, p 216–217.
5. Aisaka K, Aritsuka T, Misaka S, Toyama K, Uchiyama K, Ishibashi K, Kawaguchi H, Sakurai T. Design rule for frequency–voltage cooperative power control and its application to an MPEG-4 decoder. Tech Rep IEICE 2002;ICD2002-37. (in Japanese)
6. Aisaka K, Aritsuka T, Misaka S, Toyama K, Uchiyama K, Ishibashi K, Kawaguchi H, Sakurai T. Design rule and its algorithm for frequency–voltage

cooperative power control. Tech Rep IEICE 2002;ICD2002-72. (in Japanese)
7. Yamada T, Ishikawa M, Ogata Y, Tsunoda T, Irita T, Tamaki S, Nishiyama K, Kamei T, Tatezawa K, Arakawa F, Nakazawa T, Hattori T, Uchiyama K. A 133 MHz 170mW 10μA standby application processor for 3G cellular phones. ISSCC Digest of Technical Papers, p 370–371, 2002.
8. Transmeta Corporation. Crusoe Processor Model TM5500/TM5800 Data Book V1.0.
9. Transmeta Corporation. Processor Product Brief Model TM5800, 2003.

AUTHORS (from left to right)





Keisuke Toyama received his M.S. degree in information science from Kyoto University in 1980 and joined Hitachi, Ltd. He has been engaged in research and development of compiling technologies, optimizing compilers, Java systems, and low-power software for system LSIs at the Systems Development Laboratory and the Central Research Laboratory. From 1987 to 1988 he was a research associate in the Department of Computer Science, University of Copenhagen. He is a member of IPSJ.

Satoshi Misaka (regular member) received his M.S. degree in information and communication engineering from Tokyo Denki University in 1994 and joined the Central Research Laboratory, Hitachi, Ltd. He has been engaged in research and development related to standardization for MPEG-4, reuse of audio/visual middleware for embedded microprocessors, and low-power software.

Kazuo Aisaka received his M.S. degree in information and communication engineering from the University of Tokyo in 1982. From 1982 to 1998, he was with the Central Research Laboratory, Hitachi, Ltd., and was involved in research and development related to image processing systems for medical treatment. After that, he was engaged in software engineering and low-power software for embedded microprocessors. Currently, he is a senior engineer of the planning division of the Central Research Laboratory.

Toshiyuki Aritsuka received his B.S. and M.S. degrees in electrical and electronic engineering from Sophia University in 1985 and 1987 and joined the Central Research Laboratory, Hitachi, Ltd. He has mainly been engaged in research and development on acoustics, voice information processing, and system LSIs. From 1989 to 1992, he was a researcher at the Advanced Telecommunications Research Institute International (ATR). Currently, he is a senior researcher of the Central Research Laboratory.

Kunio Uchiyama (regular member) received his B.S. and M.S. degrees in information science from Tokyo Institute of Technology in 1976 and 1978, and Ph.D. degree in advanced applied electronics in 2001. Since 1978, he has been working for the Central Research Laboratory, Hitachi, Ltd., on design automation, small-scale mainframe, cache memory, and microprocessors. From 1985 to 1986, he was a visiting researcher in the Department of Computer Science, Carnegie-Mellon University. He won the Ichimura Award, the R&D100 Award, and the Chief Officer's Award of the Japanese Science and Technology Agency in 1998, 1999, and 2000, respectively. He is a member of IEEE.

AUTHORS (continued) (from left to right)



Koichiro Ishibashi (regular member) received his M.S. and Ph.D. degrees in engineering from Tokyo Institute of Technology in 1982 and 1985 and joined the Central Research Laboratory, Hitachi, Ltd., where he was involved in research and development of highly integrated SRAMs, low-power microprocessors, and low-power circuit technology. He led the Low Power Technology Group, Design Technology Development Department of the Semiconductor Technology Academic Research Center (STARC) from 2001 to 2004. He has been senior manager of the Advanced Design Framework Development Department of Renesas Technology Corporation. He won the R&D100 Award in 1999 for development of the SH4 microprocessor. He is an IEEE senior member.

Hiroshi Kawaguchi received his B.S. and M.S. degrees in electronic engineering from Chiba University in 1991 and 1993 and joined Konami Corporation, where he developed arcade entertainment systems. He moved to the Institute of Industrial Science, University of Tokyo, in 1996 as a technical associate, and is currently a research associate. His research interests include low-voltage VLSI designs, low-power hardware systems, and wireless circuits. He is a member of IEEE and ACM.

Takayasu Sakurai (regular member) received his B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Tokyo in 1976, 1978, and 1981 and joined Toshiba Corporation, where he designed CMOS DRAM, SRAM, and BiCMOS ASICs. From 1988 to 1990, he was a visiting researcher at the University of California at Berkeley, doing research in the field of VLSI CAD. Since 1996, he has been a professor at the University of Tokyo, working on low-power and high-performance system LSI designs. He served as a program committee member for CICC, DAC, ICCAD, ICVC, ISLPED, ASP-DAC, TAU, CSW, VLSI and FPGA Workshops. He is a technical committee chairperson for the Symposium on VLSI Circuits, an IEEE AdCom member, an IEEE fellow, and an IEEE distinguished lecturer.