DVFS Algorithm Exploiting Correlation in Runtime Distribution

Jungsoo Kim*, Sungjoo Yoo, and Chong-Min Kyung***

Abstract—**Dynamic voltage and frequency scaling (DVFS) is an effective method to achieve low power design. In our work, we present an analytical DVFS method which judiciously exploits correlation information in runtime distribution while satisfying deadline constraints. The proposed method overcomes the previous distribution-aware DVFS method [2] which has pessimistic assumption on which runtime distributions are independent. Experimental results show the correlation-aware DVFS offers 13.3% energy reduction compared to existing distribution-aware DVFS [2].**

Index Terms—**Dynamic voltage and frequency scaling (DVFS), energy optimization, runtime distribution, correlation**

I. INTRODUCTION

Dynamic voltage and frequency scaling (DVFS) is one of the effective methods to reduce energy consumption. In DVFS, frequency is set to the ratio of remaining work-to-do to remaining time-to-deadline. Thus, prediction of remaining work-to-do (in short, workload prediction) plays a central role in DVFS. Most of previous DVFS methods perform workload prediction based on the worst case execution cycle (WCEC) of software to satisfy imposed real-time constraint [1]. However, software workload is varied and most of the workload is much less than WCEC. The reason why software workload has distribution is largely divided into three: 1) data dependency (e.g., data dependent loop iteration), 2) control-flow dependency (e.g., if/else, switch/case, etc), and 3) architectural dependency (e.g., cache hit/miss, TLB hit/miss, etc) Thus, WCEC-based workload prediction results in pessimistic voltage and frequency scaling since the predicted remaining workload is much higher than the actual workload.

To exploit the distribution of software workload, thereby achieving further energy savings, several works have been proposed [2,3]. In [2], it presents a workload prediction method which analytically exploits the software workload distribution of each program region with assuming that the workload of each program region is independent. However, in a real-life software program, especially, multimedia application (e.g., H.264 decoder, MPEG4, etc), upcoming software workload can be predicted using actual workload history of previous iteration (temporal correlation) and that of previous program region (spatial correlation). Therefore, we can save more energy consumption by exploiting the correlation information of software program along with workload distribution. In this paper, we present an analytical DVFS method which exploits workload distribution and its (temporal and spatial) correlation information. This paper is organized as follows. Section II defines some terms and problem which will be solved in this paper. Section III presents the proposed method. Section IV gives experimental results followed by conclusion in Section V.

II. TERMINOLOGY AND PROBLEM DEFINITION

First, we define some terms which are frequently used in entire this paper.

Node (n_i): Program region where voltage/frequency is set at the start and maintained until the end of the

Manuscript received Jun. 2, 2009; revised Jun. 11, 2009.
* Dept. EE., KAIST Daejeon, Korea
** Dept. EE., POSTECH Pohang, Korea
E-mail : jskim@vslab.kaist.ac.kr

program region

PDF$_i$ (<x_i, p_i>): probability distribution function of n_i: x_i represents the number of processor clock cycles of i-th node (n_i) and p_i represents the corresponding probability when workload of n_i amounts to x_i

Bin ($n_{i,j}$): a certain range of execution cycle (as Fig. 1 (b) shows); $n_{i,j}$ represent j-th bin of i-th node (n_i)

Transition probability ($p(n_{i,j}, n_{i+1,k})$): probability of which workload of n_{i+1} is at the k-th bin ($n_{i+1,k}$) when the workload of n_i is at the j-th bin ($n_{i,j}$)

The input of our solution consists of 1) software program partitioned into several program regions, 2) PDF of each program region, and 3) an analytical energy model. In the partition of program region, we set the program region where time and energy overhead of the voltage/ frequency transition can be negligible, since voltage/frequency transition occurs at the start of each node.[1] Using the inputs, first, we partition each program region into several bins according to the number of execution cycles of each program region as illustrated in Figs. (b) and (c). Then, we calculate remaining workload ($w_{i,j}$) of each bin which minimizes average energy consumption by exploiting workload distribution and its (temporal/spatial) correlation information in design-time. By using the calculated energy-optimal remaining workload, we set frequency level in runtime as the ratio of the remaining workload to remaining time-to-deadline. Supply voltage is also scaled to the level where a processor is operated with the frequency with the minimum energy consumption.

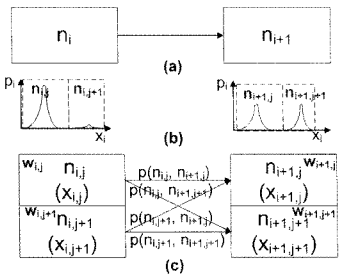

Fig. 1. Problem definition: (a) software program partitioned into program regions, (b) PDF of each program region and bin partition, and (c) bin partitioned software program.

[1] There is a sophisticated software partitioning method [5]. However, in this paper, we partition an application into several functional blocks, e.g., CAVLD, macroblock decoding, and deblocking filter, in H.264 decoder.

III. DVFS Exploiting Correlation Of Runtime Distribution

1. Bin Partitioning

A program consists of several operation modes. For example, the operation of H.264 baseline decoder can largely be classified into two modes: I- and P-frame decoding modes. According to the operation mode, different operation is performed. It can be known by analyzing the target program. Based on the analysis, energy consumption can be further reduced by optimizing and setting voltage/frequency corresponding to each operation mode. However, since the analysis takes a lot of time and need to be performed for each program, it is time-consuming.

According to an operation mode, runtime can distinctly be classified. That is, we can infer the operation mode of a program by only examining the runtime, not analyzing program. Fig. 2 shows runtime distribution of three program regions of H.264 decoder [6]. As this figure shows, runtimes are distinctly different when H.264 decoder performs I- or P-frame decoding.

We partition the runtime of each program region into a finite number of groups, and each group is called bin. The procedure is called *bin partitioning*. By the bin portioning, we can apply DVFS more effectively since we can utilize transition probability of across bins and runtime distribution of a bin is narrower than that of a node.

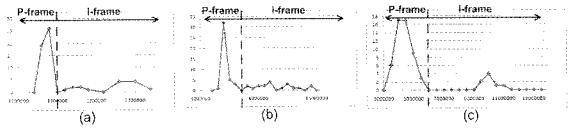

Fig. 2. Runtime distribution of H.264 decoder: (a) CAVLD, (b) MB decoding, and (c) deblocking filter

2. Design-Time Remaining Workload Prediction Exploiting Spatial Correlation

In this subsection, we present a method which calculates the remaining workload ($w_{i,j}$) of each bin that gives minimum energy consumption. Since transition probabilities, i.e., $p(n_{i,j}, n_{i+1,k})$, are different according to the bins which are executed, the energy-optimal remaining workload can be different at each bin. Thus, we

calculate the remaining workload at each bin, not at each node. Energy consumption (E_i) of the task graph in Fig. 1 (c) is calculated as follows.[2]

$$\begin{aligned} E_i = & p(n_{i,j}, n_{i+1,j}) \cdot \left(f_{i,j}^2 x_{i,j} + f_{i+1,j}^2 x_{i+1,j}\right) \\ & + p(n_{i,j}, n_{i+1,j+1}) \cdot \left(f_{i,j}^2 x_{i,j} + f_{i+1,j+1}^2 x_{i+1,j+1}\right) \\ & + p(n_{i,j+1}, n_{i+1,j}) \cdot \left(f_{i,j+1}^2 x_{i,j+1} + f_{i+1,j}^2 x_{i+1,j}\right) \\ & + p(n_{i,j+1}, n_{i+1,j+1}) \cdot \left(f_{i,j+1}^2 x_{i,j+1} + f_{i+1,j+1}^2 x_{i+1,j+1}\right) \end{aligned} \quad (1)$$

By replacing the frequency of each bin, i.e., $f_{i,j}$, into the ratio of remaining workload to remaining time, i.e., $f_{i,j}=w_{i,j}/T_i$, and integrating the energy consumption with respect to PDF of each node, average energy consumption ($\overline{E_i}$) is calculated as follows.

$$\overline{E_i} = w_{i,j}^2 \overline{x_{i,j}} + Z_{i,j} \cdot \int \frac{p_i}{\left(1 - \frac{x_{i,j}}{w_{i,j}}\right)^2} dx_i + w_{i,j+1}^2 \overline{x_{i,j+1}} + Z_{i,j+1} \cdot \int \frac{p_i}{\left(1 - \frac{x_{i,j+1}}{w_{i,j+1}}\right)^2} dx_i \quad (2)$$

where

$$Z_{i,j} = p(n_{i,j}, n_{i+1,j}) \cdot w_{i+1,j}^2 \overline{x_{i+1,j}} + p(n_{i,j}, n_{i+1,j+1}) \cdot w_{i+1,j+1}^2 \overline{x_{i+1,j+1}}$$

$$Z_{i,j+1} = p(n_{i,j+1}, n_{i+1,j}) \cdot w_{i+1,j}^2 \overline{x_{i+1,j}} + p(n_{i,j-1}, n_{i+1,j+1}) \cdot w_{i+1,j+1}^2 \overline{x_{i+1,j+1}}$$

Since the average energy consumption ($\overline{E_i}$) is continuous and convex with respect to $w_{i,j}$ and $w_{i,j+1}$, $w_{i,j}$ and $w_{i,j+1}$ which minimize the average energy consumption are calculated by finding points which satisfy $\partial \overline{E_i} / \partial w_{i,j} = 0$ and $\partial \overline{E_i} / \partial w_{i,j+1} = 0$, respectively. Since remaining workloads of successor node (n_{i+1}), i.e., $w_{i+1,j}$ and $w_{i+1,j+1}$, are known[3] when we calculate $w_{i,j}$ and $w_{i,j+1}$, and assuming that $w_{i,j}$ and $w_{i,j+1}$ are independent, we can rearrange the derivatives as follows.

$$\begin{aligned} \frac{\partial \overline{E_i}}{\partial w_{i,j}} = 0 \Rightarrow \overline{x_{i,j}} + Z_{i,j} \cdot \left[\sum_{K=1}^{M} \frac{-x_{i,j}(k) p_i(k)}{\left(w_{i,j} - x_{i,j}(k)\right)^3}\right] = 0 \\ \frac{\partial \overline{E_i}}{\partial w_{i,j+1}} = 0 \Rightarrow \overline{x_{i,j+1}} + Z_{i,j+1} \cdot \left[\sum_{K=1}^{M} \frac{-x_{i,j+1}(k) p_i(k)}{\left(w_{i,j+1} - x_{i,j+1}(k)\right)^3}\right] = 0 \end{aligned} \quad (3)$$

When N nodes are cascaded, we calculate the remaining workload of each bin as the similar method in [2]. When conditional branches are present, it can be easily applied by just multiplying branch-taken probability into the transition probability ($p(n_{i,j}, n_{i+1,k})$), then applying the same procedure.

3. Runtime Voltage/Frequency Selection Exploiting Temporal Correlation

In runtime, we first predict the range of runtime, i.e., bin index, of each node using runtime history of each node. In the runtime history, temporal correlation is considered. The procedure is called *bin prediction*. After bin prediction, we set remaining workload of each node corresponding to the predicted bin index. Then, frequency (f_i) is set to the ratio of the remaining workload (w_i) to remaining time (T_i) which is measured at the start of each node, i.e., $f_i = w_i / T_i$.

In the frequency setting, we check whether deadline violation happens when WCEC of target program occurs. The procedure is called *feasibility check*. It is performed as follows.

$$\left(\frac{WCEC_i}{f_i} + T_{tr}\right) + T_{i+1}^{\min\_req} \le T_i \quad (4)$$

where $T_{i+1}^{\min\_req} = \frac{WCEC_{i+1,end}}{f_{\max}} + T_{tr}$

In the above equation, $WCEC_i$ and $WCEC_{i+1,end}$ are, respectively, worst-case execution cycle of i-th and from (i+1)-th to the end of the program region. T_{tr} is the time overhead of voltage/frequency transition. $T_{i+1}^{\min\_req}$ is the minimum required time from (i+1)-th to the end of program, not to violate deadline. At the start of i-th node, i.e., v_i/f_i scaling, we first set f_i as w_i/T_i. If the inequality in Eqn. (4) is hold, we set the frequency of i-th node as f_i; if not, we set f_i as the lowest value which satisfies the deadline constraint, i.e., less than T_i in Eqn. (4). The voltage is also scaled to the pre-characterized level which gives minimum energy consumption when a processor is operated at the calculated frequency level.

IV. Experimental Result

We use real-life multimedia program, H.264 decoder [6] (when decoding 60 frames of QCIF images) in the

[2] Per-cycle energy consumption is proportional to the square of frequency, i.e., $E_{cycle} \sim f^2$ [2].

[3] Since remaining workload is calculated from the last to the first node, i.e., reverse topological order, we have already known the remaining workload of n_{i+1}, i.e., w_{i+1}, when calculating that of n_i, i.e., w_i. Details are presented in [2].

experiments. We partition H.264 program into eight nodes: the first node is a program region for performing CAVLD, the following four nodes are program regions for macroblock (MB) decoding, and the last three nodes are program regions for deblocking filter. We run H.264 decoder application and profile the number of processor clock cycles of each node using cycle-accurate ISS of MPCore in commercial SoCDesigner [7]. The frequency ranges 300MHz ~ 2.5GHz with 11 levels. We assume that voltage/frequency transition takes 20μs. We calculated energy overhead in the voltage/frequency transition using the model presented in [8].

We compared the following four methods.
D-DVFS [2]: runtime distribution-aware DVFS
DC-HIST-DIST: runtime distribution- and correlation-aware DVFS using history based bin predictor
DC-ORACLE-DIST: runtime distribution- and correlation-aware DVFS using oracle bin predictor
ORACLE-DVFS: ideal DVFS based on perfectly accurate workload prediction

The Table 1 shows the energy consumption results normalized into the energy consumption of D-DVFS method. DC-HIST-DVFS offers 13.3% energy savings compared to D-DVFS due to the effectiveness of exploiting correlation of runtime. When bin prediction is perfectly accurate, i.e., DC-ORACLE-DVFS, it offers 1.1% (=0.867-0.856) further energy savings. The energy consumption is just larger 11.9% (=0.856-0.737) compared with oracle DVFS, i.e., the lowest energy consumption obtained from DVFS.

Table 1. Energy savings in various DVFS methods

D-DVFS	DC-HIST-DVFS	DC-ORACLE-DVFS	ORACLE-DVFS
1.000	0.867	0.856	0.737

V. CONCLUSIONS

In this paper, we presented an analytical DVFS method which performs workload prediction exploiting workload distribution of software program and its temporal and spatial correlation information in uni-processor system. Compared to other correlation-aware DVFS method, it does not require time-consuming software analysis effort. In design-time, it performs workload prediction with exploiting spatial correlation information. In runtime, it predicts a range of software workload (bin index) by utilizing temporal correlation information and scales voltage/frequency by finding corresponding pre-calculated energy-optimal workload. The experimental results show that the proposed method offers up to 13.3% further energy savings compared to the DVFS method without exploiting correlation information.

REFERENCES

[1] S. Lee and T. Sakurai, "Run-time voltage hopping for low-power real-time systems," In DAC, 2000.
[2] S. Hong, et al., "Runtime distribution-aware dynamic voltage scaling," in ICCAD, 2006.
[3] C. Xian and Y.-H. Lu, "Dynamic voltage scaling for multitasking real-time systems with uncertain execution time," in GLSVLSI, 2006.
[4] M.Roitzsch, "Slice-balancing H.264 video encoding for improved scalability of multicore decoding," in EMSOFT, 2007.
[5] S. Oh, J. Kim, S. Kim, and C.-M. Kyung, "Task partitioning algorithm for intra-task dynamic voltage scaling," in ISCAS 2008.
[6] JM reference software. [Online]. Available: http://iphome.hhi.de/suehring/tml/
[7] SoCDesigner, ARM Co. Ltd. [Online]. Available: http://www.arm.com/products/DevTools/
[8] A. Azevedo, I. Issenin, R. Cornea, R. Gupta, N. Dutt, A. Veidenbaum, and A. Nicolau, "Profile-based dynamic voltage scheduling using program checkpoints," in Proc. DATE, 2002.

Jungsoo Kim received the B.S. degree in Electrical Engineering from KAIST in 2005. Since 2005, he has been pursuing the unified course of the M.S. and the Ph.D. degree in the Department of Electrical Engineering and Computer Science at KAIST. His research interests include dynamic power and thermal management, MPSoC design and massive parallel processing.

Sungjoo Yoo received the B.S. degree, the Master's degree, and the Ph.D degree in electronics engineering from Seoul National University, Seoul, Korea, in 1992, 1995, and 2000. He was a researcher at TIMA laboratory, Grenoble, France from 2000 to 2004. He was a senior and principal engineer with Samsung Electronics, Korea, from 2004 to 2008. He is with the Pohang University of Science and Technology (POSTECH), Korea since 2008. His research interests include dynamic power and thermal management, on-chip network, multi-threaded software and architecture, and fault tolerance of solid state disk.

Chong-Min Kyung received the B.S. degree in Electronics Engineering from Seoul National University in 1975, the M.S. and Ph.D. degree in Electrical Engineering from KAIST in 1977 and 1981, respectively. From April 1981 to January 1983, he worked at Bell Telephone Laboratories, Murray Hill, New Jersey as a postdoc. Since he joined KAIST in 1983, he has been working on System-on-a-Chip design and verification methodology, processor and graphics architectures for high-speed and/or low-power applications including mobile video codec.

He received the Most Excellent Design Award, and Special Feature Award in the University Design Contest in the ASP-DAC 1997 and 1998, respectively. He received the Best Paper Awards in the 36th DAC held in New Orleans, LA, the 10th ICSPAT(International Conference on Signal Processing Application and Technology), Orlando, FL, in September 1999, and the 1999 ICCD (International Conference on Computer Design), Austin, TX. He was General Chair of A-SSCC(Asian Solid-State Circuits Conference) 2007, and ASP-DAC 2008. In 2000, he received National Medal from Korean government for his contribution to research and education in IC design. He is a member of NAEK(National Academy of Engineering Korea) and KAST(Korean Academy of Science and Technology). He is an IEEE fellow and Hynix Chair Professor at the KAIST.