Continuous Frequency Adjustment Technique Based on Dynamic Workload Prediction[1]

Hwisung Jung and Massoud Pedram
University of Southern California
Department of Electrical Engineering
{*hwijung, pedram*}*@usc.edu*

Abstract

Real-time embedded systems increasingly rely on dynamic power management to balance between power and performance goals. In this paper, we present a technique for continuous frequency adjustment (CFA) which enables one to adjust the frequency values of various functional blocks in the system at very low granularity so as to minimize energy while meeting a performance constraint. A key feature of the proposed technique is that the workload characteristics for functional blocks are effectively captured at runtime to generate a frequency value that is continuously adjusted, thereby eliminating the delay and energy penalties incurred by transitions between power-saving modes. The workload prediction is accomplished by solving an initial value problem (IVP). Applying CFA to a real-time system in 65nm CMOS technology, we demonstrate the effectiveness of the proposed technique by reporting 13.6% energy saving under a performance constraint.

1. Introduction

As more power-managed functional blocks (FBs) are being built to realize power-saving opportunities by utilizing dynamic voltage and frequency scaling (DVFS) techniques, the task of integrating multiple power management policies in a single system is becoming ever more challenging. Furthermore, although current CMOS technologies allow an increasing number of different clock and voltage domains to be specified on the same chip, traditional dynamic power management (DPM) methods have not been able to take full advantage of DVFS techniques due to intricate trade-offs between the power-savings and performance constraints. This is because a system-level power manager (PM) has only limited control over power-saving techniques due to additional power and delay costs incurred during power-mode transitions [1]. In addition, the power management routine, most likely residing in the operating system, can itself become a heavy duty since it has to continually monitor the workload of FBs and send DVFS assignment commands to them [2].

DVFS-enabling techniques depend not only on the configuration of the voltage/frequency control circuits, but also on the efficiency of the prediction mechanism used by the PM to set the voltage/frequency levels. As shown in [3]-[8], the problem of determining a power management policy with DVFS techniques at system-level has received a lot of attention. In [3], the authors present a frequency management method based on variable updated intervals, instead of using a fixed update interval, where a frequency scheduling method is based on effective deadline mechanism. The analytical models for selecting an optimal DVFS under tight performance constraints are presented in [4][5]. Reference [6] presents an online DVFS technique by utilizing interface queues to guide the DVFS control in a multiple clock and voltage domain architecture. A voltage island-based power management technique is proposed in [7] to meet a performance constraint in multi-threshold CMOS technologies. Authors of [8] present an optimization technique for power mode transitions under timing constraints.

Although all of the above techniques perform DVFS based on power management policies, little attention has been paid to handle variable frequency adjustment and prediction techniques by using hardware-control mechanisms, which minimizes the computational efforts by the PM. Furthermore, traditional approaches for DPM, mainly based on the software-control of power-saving techniques, are highly dependent on the speed of operating system, which incur non-negligible overhead. Thus, minimizing the overhead of power-mode transitions in real-time with the hardware-control architecture is an important step to guarantee the quality of DPM techniques. This is precisely the contribution of the present paper.

In this paper, we present a power management framework for dynamic continuous frequency adjustment which provides power-saving opportunities by dynamically and continuously adjusting a variable operating frequency. The basic idea of CFA is to eliminate the power and delay costs incurred by the power-mode transitions which involve clock generators (e.g., PLL). Predicting a workload of tasks is formulated as an initial value problem (IVP) [9], where the frequency is adjusted by the proposed dynamic frequency adapter. Note that the IVP formulation in the paper determines the workload value of future time subsequent to a given time.

The remainder of this paper is organized as follows. Section 2 provides a motivational example, while section 3 describes the proposed architecture. In section 4, we present the details of the workload prediction technique. Experimental results and conclusion are given in section 5 and 6.

[1] This research is supported in part by NSF grant no. 0509564.

2. Motivational Example

According to conventional DPM approaches [10], where many FBs in a system are equipped with multiple power-performance states (e.g., sleep, idle, and active modes), a PM sends an command, i.e., DFS (Dynamic Frequency Scaling) values, to each FB. For example, as shown in Figure 1, where we assume each FB has three active states which is controlled by DFS values, the service provider (SP) or the service requestor (SR) can switch between the different speed-levels, where $DFS_1 < DFS_2 < DFS_3$ in terms of frequency values. The PM monitors the current workload of the system by looking into the corresponding service queue (SQ) to adjust the DFS value for each FB. A transition into or out of a power-performance state (i.e., dotted arrows in the figure), however, consumes energy and/or incurs delay penalty that may not be negligible.

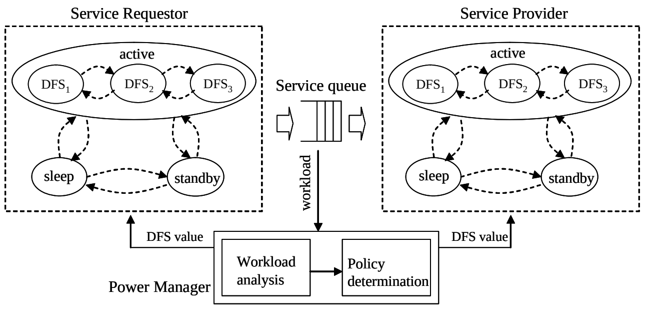

Figure 1. Conventional dynamic power management approach.

Attempting to greedily respond to the workload changes so as to provide an optimal DVFS value can result in significant energy and delay overheads associated with mode transitions. To solve this problem, a software component, which can predict the required future performance level of the system to prevent frequent power mode transitions, has been incorporated into the power managers of [2][3]. Although these prediction methods help reduce the energy overheads, there are some disadvantages because i) a software-oriented prediction algorithm increases the computational overhead of the PM that resides in the driver or the operating system, and ii) when using a PLL (Phase-Lock Loop) to effect a frequency change, the FB may be stalled during the lock time of the PLL. Consequently, use of the PLL to realize the DFS setting commanded by the PM may result in sizeable performance penalty [11]. The main contribution of our work is that we predict the workload level for the next time step and ramp up (or down) the system frequency in a continuous manner until the target frequency value is achieved, where there is never a need for stalling the FB.

3. Power Management Architecture

In this section we present a platform-specific continuous frequency adjustment (CFA) technique. The target system is comprised of various software and hardware modules, which include a power manager (PM), a performance monitoring unit (PMU), and a dynamic frequency adapter (DFA), as depicted in Figure 2. Note that the DFA is implemented as a hardware module to minimize software-oriented computational efforts.

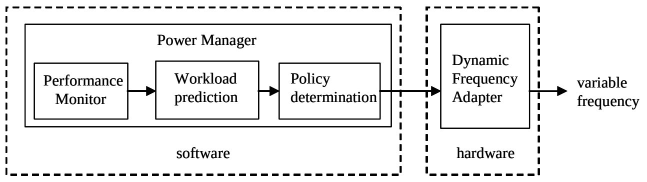

Figure 2. The proposed power management architecture.

3.1 Power Manager

The main goal of the PM is to determine and execute a power management policy (i.e., one that maps workloads to power state transition commands so as to minimize the system energy dissipation), based on the information provided by the PMU. The PM consists of a workload prediction and a policy determination. In this paper, we focus on the selection of optimal frequency value and continuous (gradual) change in system frequency moving from current value to target value.

3.2 Performance monitoring unit

The PMU profiles and analyzes the workload (e.g., the arrival rate of tasks) characteristics by looking into the SQ. In our problem setup, the SQ of each FB is represented by the G/M/1 queuing model, whereby the inter-arrival times are arbitrarily distributed and the service times are exponentially distributed [12]. Note that the service time behavior of each FB is captured in the form of the service time distribution for the FB when it is in the active mode. Similarly, the input request behavior (i.e., workload) for each FB is modeled by the request inter-arrival time distribution at the corresponding input queue. Details of the G/M/1 queuing model are omitted here to save space. Interested reader may refer to [12].

3.3 Dynamic Frequency Adapter

When the workload of an FB changes frequently, the task of deciding what frequency value to assign to the FB becomes increasingly difficult. Furthermore, the conventional PLL-based frequency scaling techniques waste energy and delay when they change the frequency values. To overcome these shortcomings, we present a workload-aware DFA to generate a continuously varying frequency for each FB.

One benefit of using a variable frequency is that the DFA enables each FB to remain functional even when its frequency is being adjusted, while satisfying the required performance. Furthermore, the DFA is able to increase (or decrease) the operating frequency value at a slow or fast rate with the help of the PMU, depending on how slow or fast the workload is changing and what the user-specified preferences are, as depicted in Figure 3.

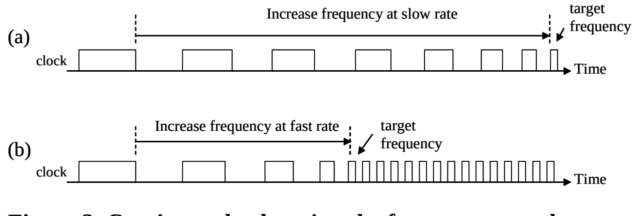
(a)
Increase frequency at slow rate
target frequency
clock
Time
(b)
Increase frequency at fast rate
target frequency
clock
Time

Figure 3. Continuously changing the frequency at a slow pace (a) or fast pace (b).

The procedure for continuously adjusting the frequency is explained as follows. The PM examines the workload of each FB at decision epoch n+1 for the time interval ranging from decision epoch n to n+1, and subsequently, sets the frequency value of each FB for the next time ranging from n+1 to next decision epoch at time n+2 (see the next section for details of the frequency prediction algorithm). Note that each time-based or interrupt-based event occurrence is called a decision epoch. Assume that a mapping table for selecting an optimal operating frequency as a function of the workload has been provided. If the workload change is fast (slow), the interval during the frequency adjustment is performed will be shortened (lengthened) to improve DFA responsiveness. In the proposed framework, determining which frequency level to use in what time interval is implemented in hardware.

Figure 4 shows the block diagram of the proposed DFA which generates a variable frequency by using a pulse width modulation technique. The DFA is implemented inside a chip, where we effectively manage noise and signal integrity problems. Note that we apply this architecture to a specific target system (e.g., high-speed networking controller), where the operating frequency is rather low (e.g., around 200MHz), yet the system provides high throughput.

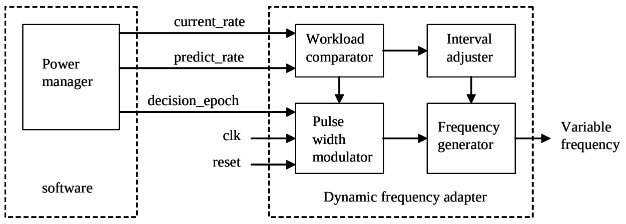

Figure 4. Block diagram of DFA.

4. Workload-Driven Frequency Adjustment

In this section, we present a workload prediction technique based on the initial value problem (IVP) formulation and a procedure of dynamic frequency adjustment.

4.1 IVP-based Workload Prediction

Assume that, by utilizing the PMU, a PM is able to monitor the current workload of the tasks at decision epochs $t_1, \ldots, t_n$ where $t_{i+1} = t_i+T$. Let $w(t)$ denote the workload (i.e., the arrival rate of tasks) of a target FB at time t and let f be a function providing the operating frequency for the FB in terms of time and workload in every interval $[t_i, t_{i+1}]$. Then, an initial value problem (IVP) may be defined to predict $w(t)$ as follows:

$$\partial w / \partial t = f(t, w), \quad w(t_i) = w_i \tag{1}$$

where $t \in [t_i, t_i + T]$, and w_i denotes the workload at the beginning of the current interval. The IVP limits the solution by an initial condition, which determines the value of solution at all future time t in the current interval. Although f can be any general function, in practice, we assume a linear function form: $f=aw(t)+b$ where a and b are appropriately-calculated slope and offset coefficients.

Since the initial workload value, w_i, is provided by the PMU, it is possible to integrate Eqn. (1) to obtain $w(t)$ in the interval $[t_i, t_{i+1}]$. The standard solution method for the IVP is to approximate the solution of the ordinary differential equation (ODE) by calculating the next value of w, i.e., $w(t+h)$ as the summation of the present value $w(t)$ plus the product of the size of a time step h and an estimated slope $w'(t)$ i.e.,

$$w(t+h) = w(t) + hw'(t) \tag{2}$$

where the smaller this time step h is, the more accurate the results will be. The difference between different ODE solvers is in how they approximate $w'(t)$ and whether and how they adaptively adjust h.

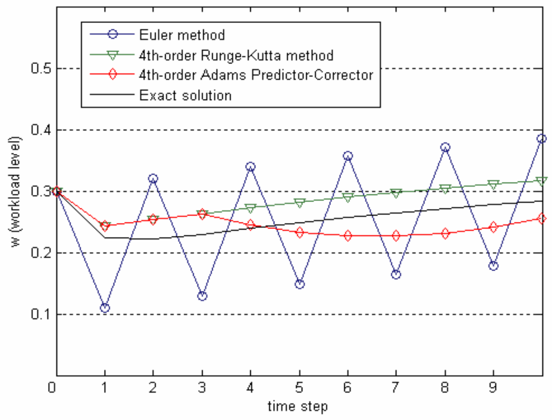

Figure 5. Evaluation of various IVP solutions.

Considering the accuracy and overhead, we have evaluated a number of methods for solving the IVP, which include the Euler's method, the 4th-order Runge-Kutta method, and the 4th-order Adams predictor-corrector method [9]. In Figure 5, we assume that $w(0) = 0.3$ as an initial value. The time step size is defined as $h = T/K$, where the time interval $[t_i, t_i + T]$ is divided into K equal-length segments. It is clearly seen that the Euler method, the simplest approach for solving the IVP, shows low accuracy (i.e., high error) in predicting the workload value, where the error is defined as the difference between the exact values and the computed approximates. However, the 4th-order Runge-Kutta method exhibits low error and consistent stability in predicting the workload value. The 4th-order Adams predictor-corrector is also accurate, but has higher computational complexity.

Figure 6 shows the trade-off between the accuracy and time step h in terms of performance of workload prediction, where time (x-axis) is defined in terms of successive time steps. In this evaluation, the 4th-order Runge-Kutta method is used with an initial value $w(0) = 0.3$. The determination of the time step size is crucial since smaller time step increases the computational overhead in the software (e.g., operating system). We use various values for time step size h (= 2, 5, and 10), where T is fixed, while monitoring the error in predicting the workload values. The time step of size 2 indicates great accuracy, but increases computational efforts by the software (due to more computations in the same interval), whereas step size of 10 exhibits lower computational efforts with lower accuracy. In our problem setup, we have empirically observed that a time step size of 5 provides a reasonable trade-off point.

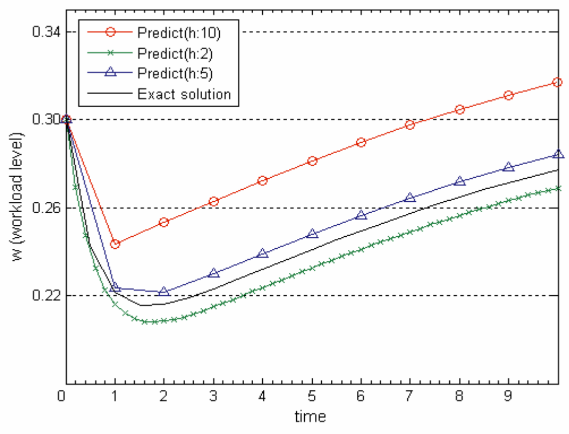

Figure 6. Trade-off between the performance and time step *h*.

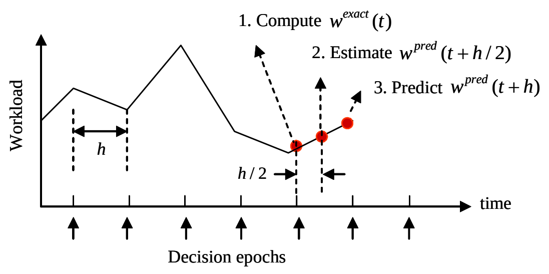
1. Compute $w^{exact}(t)$
2. Estimate $w^{pred}(t+h/2)$
3. Predict $w^{pred}(t+h)$
Workload
h
h / 2
time
Decision epochs

Figure 7. Workload prediction technique based on the midpoint method and IVP.

To make the workload prediction technique more suitable for online implementation, an efficient one-step method known as the *midpoint method* is utilized to solve the IVP. Specifically, at time instance t in $[t_i, t_i + T]$, we predict the workload value for time $t + h$, based on the value at time $t + h/2$, which is obtained by using the midpoint method, as depicted in Figure 7. First, the current workload at time t is monitored by the PMU and a frequency value is read from a pre-characterized workload-frequency mapping table by the power manager. Note that we do not want to use the predicted value for time t, which was previously computed at time $t – h$, because we can achieve the exact frequency value at time t. Next, the workload value at time $t + h/2$ is estimated by using a moving average method, for example, if the window size of the moving average calculator is 2, then, $w^{pred}(t + h/2) = 1/2\cdot(w^{exact}(t) + w^{exact}(t – h))$. This workload value is subsequently used as the midpoint estimate of the workload in the upcoming period. In particular, it is used along with $w^{exact}(t)$ to compute $w^{pred}(t + h)$ by applying the IVP.

The advantage of this prediction method is that we do not attempt to predict $w^{pred}(t + h)$ directly by using a moving average method only. Instead, we estimate the workload value for a nearer time in the future (which should provide higher accuracy) and use that value to initially estimate the rate of workload change in the upcoming period, followed by finally computing $w^{pred}(t + h)$ by solving the IVP.

4.2 Workload-Driven Frequency Adjustment

The decision about the frequency adjustment interval is made based on the difference between $w^{exact}(t)$ and $w^{pred}(t + h)$. For example, if $w^{pred}(t + h) >> w^{exact}(t)$, then the DFA will increases the frequency. Clearly, if $w^{pred}(t + h) << w^{exact}(t)$, then the DFA will decrease the frequency. On the other hand, the DFA increases (decreases) the frequency slowly if $w^{pred}(t + h)$ is only a little larger (smaller) than $w^{exact}(t)$.

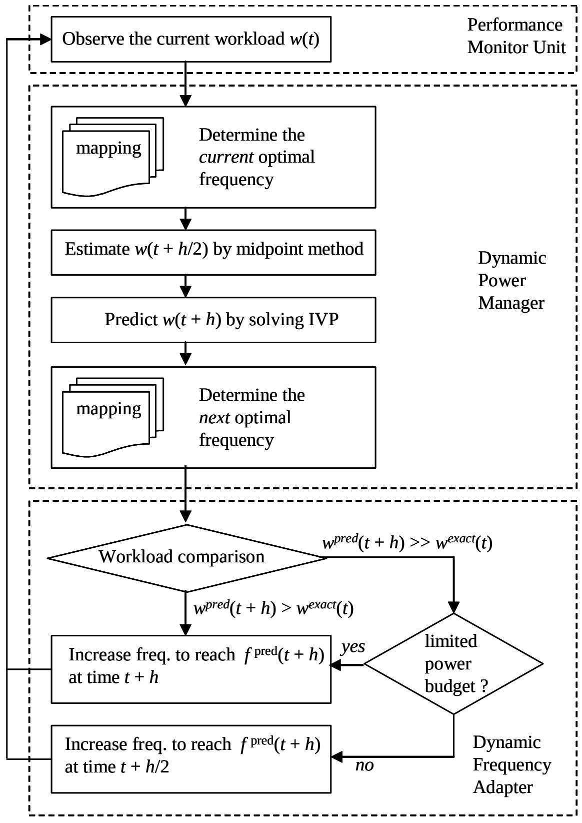
Observe the current workload w(t)
Performance Monitor Unit
mapping
Determine the current optimal frequency
Estimate w(t + h/2) by midpoint method
Predict w(t + h) by solving IVP
Dynamic Power Manager
mapping
Determine the next optimal frequency
Workload comparison
$w^{pred}(t + h) >> w^{exact}(t)$
$w^{pred}(t + h) > w^{exact}(t)$
Increase freq. to reach $f^{pred}(t + h)$ at time $t + h$
yes
limited power budget ?
Increase freq. to reach $f^{pred}(t + h)$ at time $t + h/2$
no
Dynamic Frequency Adapter

Figure 8. The flow of dynamic frequency adjustment method.

Figure 8 shows the flow of dynamic frequency adjustment technique, where $f^{pred}(t + h)$ is the frequency value obtained from the predicted workload and the workload-frequency mapping table for the two cases where $w^{pred}(t + h) >$ or $>> w^{exact}(t)$,. In this flow, we have omitted the case of $w^{pred}(t + h) <$ or $<< w^{exact}(t)$, which can be handled in a similar way. Note that when $w^{pred}(t + h) = w^{exact}(t)$, the current frequency value is maintained. It is worthwhile to mention that the DFA is capable of handling the throughput and power budget. If there is a target throughput, for example, the DFA will slowly increase the frequency up to a target frequency value that results in just-enough throughput and minimum power dissipation.

4.3 Mapping of Workloads to Optimal Frequency

The entries of the workload-frequency mapping table correspond to various values of workload (i.e., the arrival rate of tasks). Figure 9 illustrates the mapping process from workloads to an optimal operating frequency, assuming that 0.1 ≤ the arrival rate ≤ 0.9. In this figure, the pre-characterized mapping table is achieved through extensive offline simulations during design time, considering performance characteristics of each FB provided by the user or application, in a similar way as [5][14]. For example, when a power manager predicts the workload for the near future, an optimal frequency value for the next decision epoch is selected and provided to the DFA which will continuously change the operating frequency from its present value to the target value. Note that mapping from workload to operating frequencies is achieved by a simple linear function while considering the maximum and minimum operating frequencies that can be applied to the FB in question.

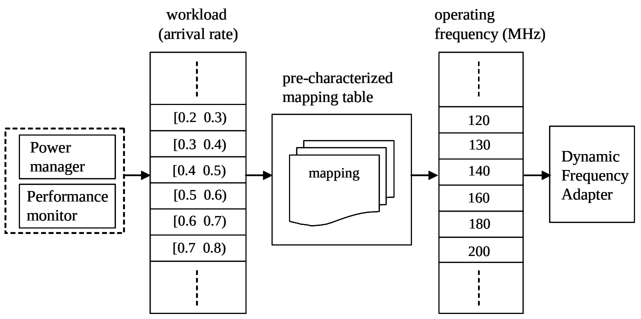

Figure 9. Mapping of workloads to optimal operating frequency values for each FB.

5. Experimental Results

In the experimental setup, we applied the proposed CFA technique to a high-speed network controller (i.e., gigabit Ethernet controller) which includes IEEE 802.3 PHY/MAC blocks, RISC processor, direct memory access (DMA) engine, PCI-E core, etc. as shown in Figure 10. This embedded system is implemented with TSMC 65nmLP library. To capture power-saving opportunities by using the proposed technique, we consider a part of the process of receiving packets inside the system, which involves Ethernet MAC (EMAC) block and control block. Note that it will not hurt the quality of the paper if we concentrate on these blocks (i.e., inside dotted box in Figure 10) to simplify the experimental setup, since they sufficiently exhibit the characteristics of the SR, the SP, and the SQ. Thus, the continuously varying frequency value is applied to the control block (i.e., the SP) by the hardware-implemented DFA, where its frequency is optimally adjusted.

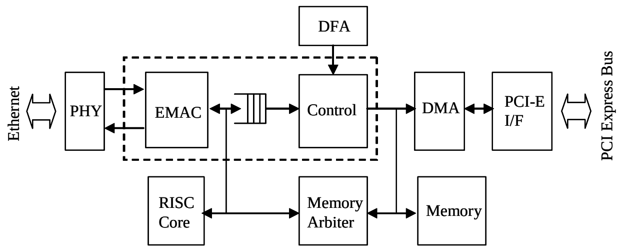

Figure 10. Block diagram of simplified Ethernet controller.

The functions performed by these blocks (i.e., EMAC and control blocks) are explained briefly as follows. The EMAC receives a data stream from the selected physical layer interface and performs address checking, CRC calculation, and CSMA/CD functions [15]. The control block calculates checksum and parses TCP/IP headers and classifies the frames based on a set of matching rules. While processing the control data in the control block, the frame data is temporarily stored in memory buffers before being sent to local interconnect through the PCIE interface.

Table 1. Energy dissipation (normalized) and utilization of the control block.

1000Base-T	Arrival rate of tasks								
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Energy (normalized)	1.96	2.62	4.37	5.04	5.85	6.67	8.89	9.86	10.83
Utilization (%)	9.5	14.2	20.6	32.1	41.8	54.0	63.2	67.5	73.5

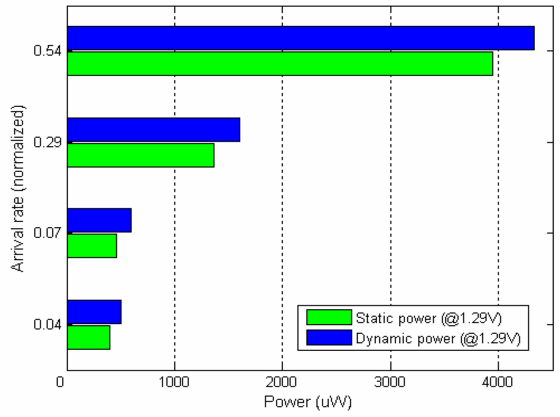

Figure 11. Power consumption of the service queue.

We first achieve the power and energy dissipation of the service provider (i.e., control block) and the service queue (i.e., memory) by using TSMC 65nmLP library, which has 3 optional operating voltages (e.g., 1.08V, 1.20V, and 1.29V). To calculate accurate power values for static and dynamic power consumption, we used SAIF (Switching Activity Interchange File) based on back-annotated RTL simulation of the system with Power compiler [16]. To achieve the energy dissipation of the control block, different workloads (e.g., the arrival rate of traffic) are used to generate the multiple columns in Table 1, where the dynamic and static power values are considered. For simulation setup, we set that the maximum full duplex bandwidth (e.g., 1000Base-T) is achieved. Note that the overhead of designing the DFA block inside the system is negligible due to its small number of gate counts (around 150 standard cells) and power dissipation (around 2uW including dynamic and static power). Figure 11 shows power consumption of the service queue in terms of the normalized arrival rate of the traffic. We set the packet size to 64bytes and the service time to 1 (by using the G/M/1 queuing model) for simplicity. For example, when the arrival rate of the tasks is 0.29 (normalized), the memory size necessary for buffering the incoming data is 5.8 times greater than the case of where the arrival rate is 0.04.

Next, we evaluate the effectiveness of the proposed CFA technique. We assume that the workload changes dynamically from 0.1 to 0.9. For comparison purpose, we implemented a couple of power management policies (denoted by PM1 and PM2 and described below) as representatives of the conventional methods, similar to [7][13]. We use three set of frequency values to simplify the experimental setup ($F_1 < F_2 < F_3$ in terms of frequency values).

PM1: Utilize dynamic frequency scaling technique, while accounting for a 100us power-mode transition overhead; the frequency assignment policy is as follows.

- Use the lowest F_1 value when $0.1 \le$ the arrival rate ≤ 0.3, i.e., low workload.
- Likewise, use F_2 and F_3 values when $0.3 <$ the arrival rate ≤ 0.6 and $0.6 <$ the arrival rate ≤ 0.9, respectively.

PM2: The same as PM1 except that a frequency change is avoided when the same frequency changes occurs consecutively. More precisely, let F^i denote the value of frequency at time *i*.

- Set $F^{i+1} = F^i$, if the predicted F^{i+1} value is the same as F^{i-1} value.

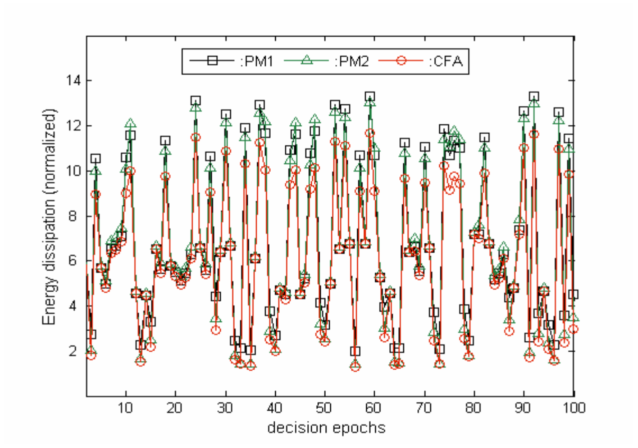

Figure 12. Evaluation of the proposed CFA technique.

Table 2. Power and energy savings of the CFA.

No. of decision epoch	Workload distribution			Average Power (mW)			Power saving over		Energy saving over	
	Low	Mid	High	PM1	PM2	CFA	PM1	PM2	PM1	PM2
100	26	43	31	13.2	11.8	11.4	12.9%	2.3%	11.4%	8.8%
500	143	196	161	13.3	11.7	11.5	13.3%	2.2%	12.0%	9.4%
1000	277	417	306	13.4	11.8	11.5	13.6%	2.1%	12.3%	9.5%
5000	1494	1954	1552	13.4	11.9	11.6	13.1%	2.1%	12.1%	9.6%

Then, we generate dynamic workloads randomly with 100, 500, 1000, and 5000 numbers of power management decision epochs and apply both above-mentioned conventional policies and the proposed power management technique to the control block. The simulation results in Figure 12, which corresponds to the case of 100 decision epochs, show that the proposed CFA technique achieves energy savings compared to the conventional methods. Results in Table 2, which also reports the characteristics of the workload distribution (e.g., Low = $0.1 \le$ the arrival rate ≤ 0.3), demonstrate that, compared to the PM1 policy, our approach achieves power and energy savings up to 13.6% and 12.3%, respectively.

6. Conclusion

In this paper, we addressed the problem of power management techniques in the context of handling dynamic frequency management, where power-mode transition cost is no longer negligible in the nano-scaled systems. We proposed a continuous frequency adjustment technique based on a workload prediction method, which minimizes the transition cost. Experimental results with a 65nm design show that the proposed technique ensures robust energy savings under dynamic workloads.

References

[1] D. Li, Q. Xie, and P.H. Chou, "Scalable Modeling and Optimization of Mode Transitions Based on Decoupled Power Management Architecture," *Proc. of DAC*, Jun., 2003.

[2] Y-H. Lu, and G. De Micheli, "Comparing System-Level Power Management Policies," *IEEE Design & Test of Computers*, Vol. 18, Issue 2, Mar.-Apr., 2001.

[3] M. Najibi, et al., "Dynamic Voltage and Frequency Management Based on Variable Update Intervals for Frequency Setting," *Proc. of ICCAD*, Nov., 2006.

[4] Y. Cho, N. Chang, C. Chakrabarti, and S. Vrudhula, "High-Level Power Management of Embedded Systems with Application-Specific Energy Cost Functions," *Proc. of DAC*, Jul., 2006.

[5] P. Rong and M. Pedram, "Power-aware Scheduling and Dynamic Voltage Setting for Tasks Running on a Hard Real-time System," *Proc. of ASP-DAC*, Jan., 2006.

[6] A. Iyer, and D. Marculescu, "Power Efficiency of Voltage Scaling in Multiple Clock, Multiple Voltage Cores," *Proc. of ICCAD*, Nov. 2002.

[7] Q. Wu, P. Juang, M. Martonosi, and D.W. Clark, "Voltage and Frequency Control with Adaptive Reaction Time in Multiple-Clock Domain Processors," *Proc. of 11th Symposium on HPCA*, Feb., 2005.

[8] J. Liu, and P.H. Chou, "Optimizing Mode Transition Sequences in Idle Intervals for Component-Level and System-Level Energy Minimization," *Proc. of ICCAD*, Nov., 2004.

[9] J.R. Dormand, *Numerical Methods for Differential Equations: A Computational Approach*, CRC Publisher, Feb., 1996.

[10] L. Benini, and G. De. Micheli, *Dynamic Power Management: Design Techniques and CAD Tools*, Kluwer Academic Publishers, 1998.

[11] R. Zhang, and G. La Rue, "Fast Acquisition Clock and Data Recovery Circuit with Low Jitter," *IEEE Journal of Solid-State Circuits*, Vol. 41, No. 5, May, 2006.

[12] S. M. Ross, *Introduction to Probability Models*, Academic Press, 8th edition, Dec., 2002.

[13] P. Choudhary, et al., "Hardware Based Frequency/Voltage Control of Voltage Frequency Island Systems," *Proc. of CODES*, Oct., 2006.

[14] H. Jung, and M. Pedram, "A Unified Framework for System-level Design: Modeling and Performance Optimization of Scalable Networking System," *Proc. of ISQED*, Mar., 2007.

[15] http://www.ieee802.org/802_tutorials IEEE 802.3 tutorial. Jul., 2005.

[16] http://www.synopsys.com Synopsys Power Compiler Documents.