[6] S. Rusu, J. Stinson, S. Tam, J. Leung, H. Muljono, and B. Cherkauer, "A 1.5-GHz 130-nm Itanium 2 Processor with 6-MB on-die L3 cache," *IEEE J. Solid-State Circuits*, vol. 38, no. 11, pp. 1887–1895, Nov. 2003.
[7] E. Safi, P. Akl, A. Moshovos, and A. Veneris, "On the latency and energy of checkpointed, superscalar register alias tables," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 3, pp. 365–377, Jul. 2009.
[8] E. Sprangle and D. Carmean, "Increasing processor performance by implementing deeper pipelines," in *Proc. Int. Conf. Comput. Arch.*, 2002, pp. 25–34.
[9] D. Tarjan, S. Thoziyoor, and N. P. Jouppi, "CACTI 4.0," HP Labs Technical Report, HPL-2006-86, 2006.
[10] J. Wang and C. Huang, "High-speed and low-power CMOS priority encoders," *IEEE J. Solid-State Circuits*, vol. 35, no. 10, pp. 1511–1514, Oct. 2000.
[11] S. Wasson and A. Brown, "Pentium 4 'Northwood' 2.2 GHz vs. Athlon XP 2000+," 2002. [Online]. Available: www.techreport.com
[12] V. Zyuban, D. Brooks, V. Srinivasan, M. Gschwind, and P. Bose, "Integrated analysis of power and performance for pipelined microprocessors," *IEEE Trans. Comput.*, vol. 53, no. 8, pp. 1004–1016, Aug. 2004.

Dynamic Voltage and Frequency Scheduling for Embedded Processors Considering Power/Performance Tradeoffs

Mostafa E. Salehi, Mehrzad Samadi, Mehrdad Najibi, Ali Afzali-Kusha, Masoud Pedram, and Sied Mehdi Fakhraie

***Abstract*—An adaptive method to perform dynamic voltage and frequency scheduling (DVFS) for minimizing the energy consumption of microprocessor chips is presented. Instead of using a fixed update interval, the proposed DVFS system makes use of adaptive update intervals for optimal frequency and voltage scheduling. The optimization enables the system to rapidly track the workload changes so as to meet soft real-time deadlines. The technique, which can be realized with very simple hardware, is completely transparent to the application. The results of applying the method to some real application workloads demonstrate considerable power savings and fewer frequency updates compared to DVFS systems based on fixed update intervals.**

***Index Terms*—Dynamic frequency scheduling, dynamic power management, dynamic voltage scheduling (DVS).**

I. Introduction

In recent years, researchers have proposed several static and dynamic techniques to scale the operating frequency and voltage of embedded processors. These techniques address power and performance tradeoffs at either hardware (fabricated chips) [1]–[3] or software levels [4]–[9]. Software-based scheduling techniques rely on precollected offline statistical information of different applications and adjust the voltage and frequency accordingly. In [10], static voltage-scheduling techniques for intra-task voltage-scheduling in hard real-time tasks based on the execution profile of the tasks are presented. The optimal intra-task voltage/frequency scheduling for single task real-time systems is found in [11]. The technique used statistical workload information. Both of these works use static voltage scaling approaches. A quasi-static voltage scaling proposed in [12] considers worst-case execution times (WCET) to guarantee the fulfillment of deadline constraints. In reality, the actual execution times of the tasks, for most of the cases, are shorter than their WCETs and may vary by up to 87% relative to the measured WCET execution times in the cases of real-world embedded tasks [13]. In order to track the dynamic changes of the workload and reduce the power consumption accordingly, it is essential to dynamically adjust the voltage during the application run-time. The researchers in [14] and [15] have proposed power management approaches which track the critical path changes and consider the impact of process variations. There are also feedback-based online dynamic voltage and frequency scheduling (DVFS) solutions that dynamically control the clock frequency and supply voltage considering the real operating conditions of the underlying processing hardware [16]–[19]. Traditional feedback-based hardware modules for online voltage scaling such as [19] and [20] are computationally expensive, and can hamper the possible energy savings. In this paper we propose an adaptive DVFS method with low area and power overhead to overcome the hardware complexity of online methods. The proposed DVFS not only scale the frequency and voltage to a nearly optimum value, but also reduce the frequency and voltage update rates considering both power consumption and system responsiveness.

The remainder of this paper is organized as follows. In Section II, we briefly review the related works while Section III explains the proposed frequency adjustment algorithm based on the effective deadline. The results are discussed in Section IV. Finally, the summary and conclusion are given in Section V.

II. Related Works

In this section, we briefly review some of the hardware-based DVFS systems which are most relevant to our proposed scheme. A DVFS method that dynamically controls the clock frequency and supply voltage with a fixed update interval is proposed in [16]. Fixed interval dynamic voltage scheduling (DVS) scheme for multiple clock domain processors has also been presented in [17]. The online DVS method proposed in [19], exploits a scheduling algorithm based on fixed update intervals. This method is implemented with complex proportional-integral-differential (PID) controllers that may compensate the power saving of the DVS scheme. The Razor DVS technique [4] uses a delay-error tolerant flip-flop for scaling the supply voltage to minimum allowed value for a given frequency. This method also works based on fixed update intervals. An online hardware-based DVFS scheme for dynamically selecting operating frequencies and voltages in multiprocessor globally asynchronous locally synchronous (GALS) systems is proposed in [20]. This DVFS approach monitors the application workload at predefined times called T_{sample} and scales frequency and voltage values accordingly. The frequency prediction algorithm of this method exploits multipliers and dividers which complicate the hardware realization of this DVFS.

All of these work use fixed update intervals for scheduling voltage and frequency. The optimum value of the fixed update interval strictly depends on the application and patterns of the workloads. Therefore, the value of fixed interval should be carefully tuned for different applications. Fixed update interval DVFS methods can be used when the behavior of the application is predictable for various applications and

Manuscript received October 22, 2009; revised March 04, 2010; accepted July 02, 2010. Date of publication August 09, 2010; date of current version August 10, 2011.

M. E. Salehi, M. Samadi, A. Afzali-Kusha, and S. M. Fakhraie are with the Nanoelectronics Center of Excellence, School of Electrical and Computer Engineering, University of Tehran, Tehran 14395, Iran (e-mail: mersali@ut.ac.ir; afzali@ut.ac.ir; fakhraie@ut.ac.ir).

M. Najibi is with the Department of Computer Engineering, Amirkabir University of Technology, Tehran 15875-4413, Iran (e-mail: najibi@ce.aut.ac.ir).

M. Pedram is with the Department of Electrical Engineering Systems, University of Southern California, Los Angeles, CA 90089 USA (e-mail: pedram@ceng.usc.edu).

Digital Object Identifier 10.1109/TVLSI.2010.2057520

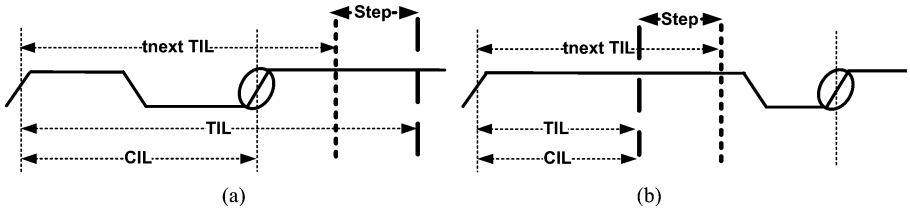

Fig. 1. Effective deadline prediction concept for cases corresponding to (a) interval decrement and (b) interval increment.

input conditions and the worst-case behavior is not very different from the average-case behavior.

Our previous work on DVFS [21] exploits an adaptive update interval for scheduling the voltage and frequency of the processor to near optimum values. This paper is the extension of the work presented in [21]. The main differences of this paper with [21] includes using low-overhead hardware instead of long history registers to reduce the required controller hardware, presenting a method for tuning the parameters of the algorithm, and considering power consumption and system responsiveness tradeoffs in voltage and frequency scheduling. It addition, this paper presents realistic results for MPEG2 decoder and some packet-processing applications, the effective deadline is redefined based on new concepts, and discusses the area overhead of the proposed DVFS components. At last, it should be mentioned that the current work assume continuous voltage values are available in the system. The use of discrete values may reduce the amount of achievable energy saving through this technique, though not significantly [18].

III. Frequency Scheduling Methods

In DVFS algorithms, the supply voltage is adaptively adjusted based on the predicted frequency. Therefore, one of the major challenges of DVFS systems is how to determine the optimal working frequency without violating the deadlines. This can be assured by following a simple conservative rule, which states that the processing of the current workload must be finished prior to the arrival of the next workload. In this way, we may consider the arrival time of the next workload as an *effective deadline* for the current workload in soft real time applications. Knowing the effective deadline for each workload, the frequency may be adjusted to the lowest required value. The importance of the effective deadline in the frequency adjustment is shown in [21] by considering greedy and deadline-aware frequency scheduling policies.

The frequency scheduling circuitry presented in [16] exploits an activity monitor and counts the number of idle cycles of the processor in fixed update intervals. When the number of idle cycles in an interval exceeds a threshold value, the frequency is lowered; otherwise, the frequency is increased or held steady. The frequency update rate in [16] directly depends on the value of the fixed interval, i.e., larger values for the interval lead to lower rates of the frequency changes, and hence, a weaker workload tracking ability. On the other hand, choosing small values for the fixed interval leads to unnecessary frequency and voltage updates. The optimum strategy is to set the CPU clock frequency to a value that will finish the workload just in time (i.e., just before the next workload arrives). This optimum strategy will thus minimize the number of idle cycles. We have developed an adaptive frequency scheduling algorithm that decreases the frequency update interval in order to track abrupt workload changes while increasing this interval to minimize unwanted voltage fluctuations for slowly varying workloads. The algorithm works based on the concept of effective deadline introduced previously.

In the proposed deadline prediction, for each workload, a prediction of the arrival time of the next workload (effective deadline) is made based on an adaptive algorithm. We use effective deadline or simply *target interval length* (TIL) interchangeably in the remainder of this paper. During each workload, the system clock cycles are counted by a *cycle counter* whose value is considered as the *current interval length* (CIL). The counter saturates when its value reaches the TIL and resets to zero at the arrival of the next workload. If at the arrival of the next workload, $\mathrm{CIL} < \mathrm{TIL}$ [see Fig. 1(a)], we decrease TIL by an interval step. If the next workload is not arrived until the CIL reaches the TIL [see Fig. 1(b)], we increase TIL by the interval step.

The value of the interval step (k_{step}) is updated adaptively to (i) track the pace of changes in the workload and (ii) avoid unnecessary changes of the TIL when the system workload is not changed much. When "k" consecutive interval increases (decreases) occur, it means that the TIL should reach a much larger (smaller) value. In both cases, the interval step will be multiplied by two to speed up the move to higher or lower TILs. If these cases do not occur, the interval step is divided by two to minimize the interval variations. Using this adaptive method for the interval step adjustment, when the system behaves in an averaging stable manner, unwanted changes in frequency setting periods are avoided while at the same time providing the ability to quickly track abrupt changes in the workload arrival rate. By choosing update intervals that are well matched to the average workload characteristics, optimum system frequencies and voltages may be achieved.

A. Proposed Frequency Scheduling Method

The predetermined value of k_{history} is used to determine the number of clock ticks that the workload history is examined. k_{history} thus represents the maximum number of acceptable idle cycles between consequent workloads. At the update interval, if all of the workload history bits are "1" ("0"), the estimated frequency is lower (higher) than the required value, and hence, should be increased (decreased) for the next workload. Finally, when some of these history bits are "1," and some are "0," it shows that the system is working with a proper frequency and should not be changed.

A simple zero/one detector logic such as one introduced in [21] may be used for frequency scheduling. For large sizes of the k_{history}, the area overheads of the zero/one detection logic increase linearly with the size of the k_{history}. To overcome the problem, we propose a count-reset-hold (CRH) circuit whose area overhead is proportional to $\log_2 k_{\mathrm{history}}$. For this purpose, we need two types of the CRH counters denoted by CRH0 and CRH1. The proposed counter is an up counter that is controlled by *hold* and *reset* signals. At each clock cycle, CRH0 (CRH1) counts the subsequent "0" ("1") bits of the input signal. It counts up while the *reset* and *hold* signals are "0." When the *reset* is high, the counter output is reset to 0, and when the *hold* signal is high, the counter output remains unchanged. Fig. 2(a) and 2(b) show both the CRH0 and CRH1 circuits.

The frequency scheduling circuit based on the proposed counter is also depicted in Fig. 2(c). When the number of consequent "0" ("1") in the activity signal reaches the k_{history}, it means that for the last k_{history} clock cycles the CPU has been idle (active), and hence, the frequency *decrease (increase)* signals are activated. Finally, note that the perfect

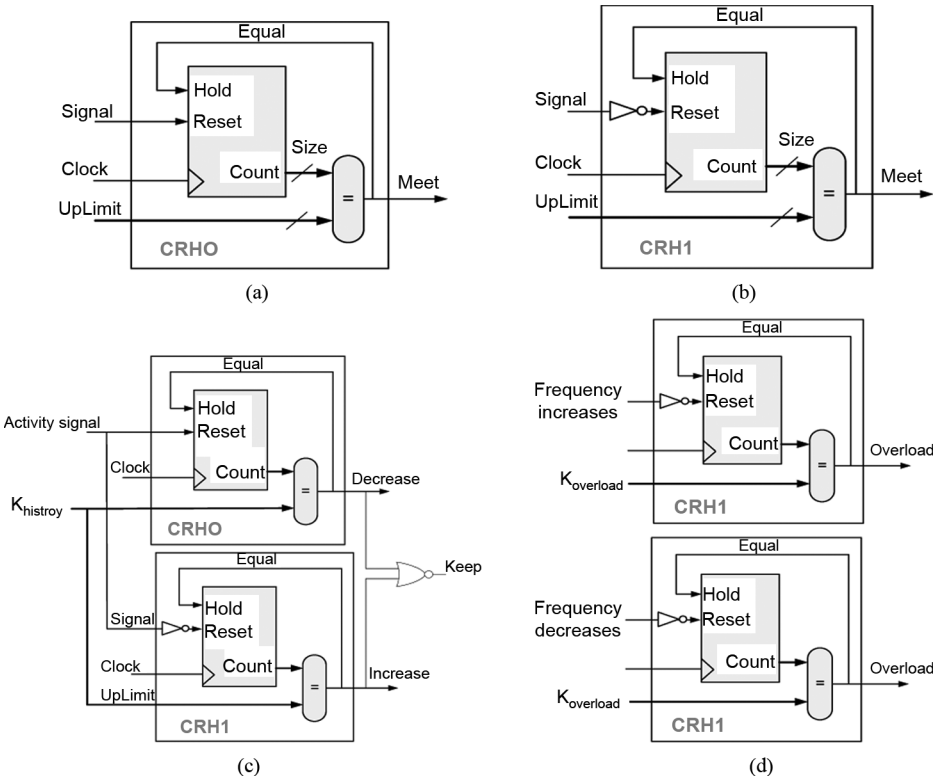

Fig. 2. Circuits for (a) CRH0, (b) CRH1, (c) frequency scheduling, (d) overload/underload detection.

value for the frequency is the amount that provides some idle cycles before the arrival of the next workload [16], [21].

B. Overload and Underload States

The concepts of overload and underload states were suggested in [21]. We detect the overload/underload situation by counting the consequent frequency increases/decreases, i.e., if the number of consequent frequency increases/decrease reaches a value denoted by $k_{\mathrm{overload}}/k_{\mathrm{underload}}$, we conclude that an abrupt increase/decrease in the workload has occurred and the system is overloaded/underloaded. If the overload or underload condition is detected, TIL will be replaced by the minimum possible value and all the frequency updates will be done in the direction of frequency increase/decrease. This small update interval leads to the highest frequency update rate. The overload/underload detection circuits are shown in Fig. 2(d).

IV. Results and Discussion

To assess the efficiency of the proposed frequency scheduling method, we applied the fixed update interval [16], adaptive update interval, and oracle DVFS methods to a workload set obtained from real workloads based on realistic computational load of the MPEG2 decoder and two packet-processing applications. We have used MIPS-based SimpleScalar [22] simulations to determine the computational load of the MPEG2 decoding in terms of instructions per frame. The computational loads of the packet-processing applications were also extracted from [24]. The packet-processing applications which were selected from the PacketBench [25] included an IPv4 lookup (IPv4-trie) and flow classification (Flow-class) algorithms that are widely used in network processing nodes. We have also exploited Wattch technique [23] in our Verilog model for estimating the power consumption based on the effective capacitance concept.

The DVFS algorithms and the power estimation methods were modeled in Verilog and simulated in Modelsim. The MPEG2 decoder workload is obtained from decoding three different MPEG2 movies. The computational load is related to the number of instructions required for decoding a frame. In the MPEG2 decoder, frames should be processed in 33 μs. Each frame-processing time violating this specified period causes a deadline miss. It should be noted that in our simulations we applied the proposed DVFM technique to many frames of the selected movies obtaining not very different results.

In ADAPTIVE, there are four parameters whose values determine the power saving capability and deadline misses (responsiveness) of the system. The parameters are the size of the workload history (k_{history}), overload/underload history $(k_{\mathrm{overload}}/k_{\mathrm{underload}})$, and the *Interval step* (k_{step}). For MPEG decoding workloads, we used some simulations to obtain the best values for the parameters. The priority of finding the values of the parameter was based on their effects on the power and deadline misses. The results showed that the most effective parameter was k_{overload}. Fig. 3 shows the deadline misses and power consumption of the selected movies as a function of the k_{overload}. As shown in Fig. 3, a good value for k_{overload} is between 2 and 7, and hence, the overload detection circuit can be implemented with a 3-bit CRH1 counter leading to the deadline misses in the range of 5%–15%, and the power consumption reduction of 40%–60%.

The next parameter that is tuned is the $k_{\mathrm{underload}}$. As the results reveal, when $k_{\mathrm{underload}}$ is around 10, the deadline miss is about 6% and the power saving is nearly saturated. This value needs a 4-bit CRH1 counter for underload detection. Finally, we have found that setting k_{history} to 3200 reduces the deadline miss to below 2% which requires a 12-bit CRH counter. The last parameter of the system is k_{step} which is used for properly tuning the interval value. We have found the power consumption and deadline miss of the selected movies as a function of k_{step}. Since the power consumption is a weak function of k_{step}, this parameter may be used for fine tuning of the deadline misses.

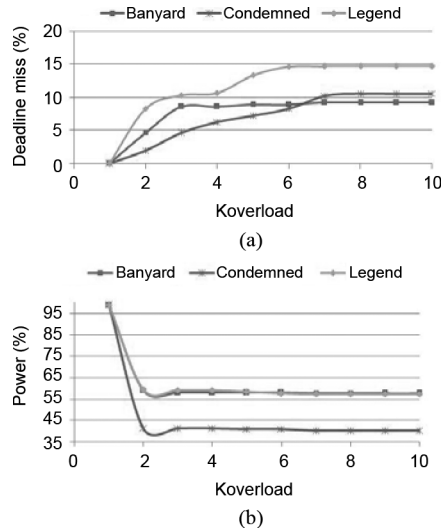

Fig. 3. Power consumption and deadline miss of the selected movies versus k_{overload}: (a) deadline miss and (b) power.

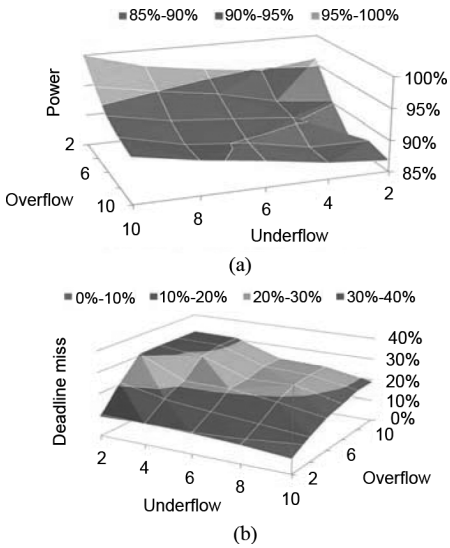

Fig. 4. Power consumption and deadline misses of the Legend movie for a wide range of k_{overload} and $k_{\mathrm{underload}}$ values: (a) power consumption and (b) deadline miss.

Our study showed that k_{overload} and $k_{\mathrm{underload}}$ were the most effective parameters. We started with the most effective parameter and fine tune the system with the remaining parameters based on the significance of the effect. Also, to assess the accuracy of the approach, we calculated the powers and deadline misses using exhaustive simulations considering wide ranges of values for different parameters. The results showed that the priority-based tuning worked well. Fig. 4 shows the effect of k_{overload} and $k_{\mathrm{underload}}$ parameters on the power consumption and deadline misses of one of the selected movies. The power values are normalized to the maximum values of the power consumption. We can observe that higher $k_{\mathrm{underload}}$ and lower k_{overload} values yield lower deadline misses (more responsiveness) while lower $k_{\mathrm{underload}}$ and higher k_{overload} values give rise to lower power consumptions.

We have tuned the parameters of ADAPTIVE based on MPEG workloads and used these parameters for the packet-processing applications as well. Therefore, we have used the results of Fig. 4 and set the values of k_{overload} and $k_{\mathrm{underload}}$ to 2 and 6, respectively. The other parameters including k_{history} and k_{step}, were also set to 1000 and 5, respectively. In Fig. 5, we have compared the power consumptions

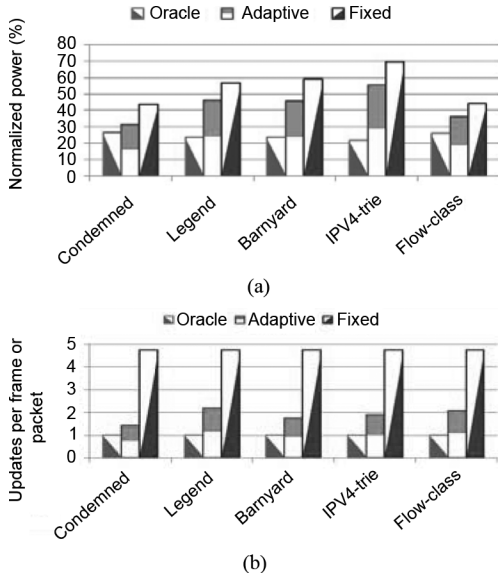

Fig. 5. (a) Power consumption and (b) update rates per frame/packet for the realistic workloads in different techniques.

and the frequency update rates of the FIXED, ADAPTIVE, and oracle (ideal) techniques for the MPEG2 decoding and two packet-processing applications.

With the oracle we mean an ideal DVFS algorithm in which the workload computational complexity is known at the beginning of processing and the frequency is set to the minimum value accordingly. Therefore, the update rate of oracle is one update per frame and the deadline miss is zero. Besides, FIXED and ADAPTIVE parameters were set such that the deadline misses of both systems were the same. As shown in Fig. 5, on average, ADAPTIVE leads to about 12% more power savings, compared to that of the FIXED, while the update rate of ADAPTIVE is also about 2.6 times lower than that of the FIXED. According to the presented results in [16], exploiting the DVFS scheme does not increase the power consumption of subsystems such as memory. Therefore, excluding the memory power consumption does not affect the results.

To evaluate the hardware overhead of the proposed adaptive DVFS, we synthesized the MIPS processor [26], frequency scheduler, overload/underload detection components and a multiplier/divider modules using a 90-nm CMOS standard cell library. The area (power) overheads for the frequency scheduler and underload/overload detector were 1.2% and 0.4% (2.3% and 0.9%), respectively. The results were normalized to the MIPS synthesis results. The selected MIPS processor was a five-stage single issue pipeline processor. The proposed algorithms in [19] and [20] require multipliers and dividers in their scheduling algorithm. The area (power) overhead for them was 11% (6.3%).

V. Conclusion

In this work, an efficient and adaptive update interval method for dynamic voltage and frequency management was proposed. In this method, the frequency and voltage of the system for the periodic workloads were scheduled while maintaining soft real-time deadlines. The saving was achieved by introducing the concept of the effective deadline and taking advantage of the possible correlation between the consecutive workloads. To show the efficacy of the method, comparisons between oracle, adaptive, and fixed interval DVFS systems were performed using realistic workloads of the MPEG2 decoder and packet-processing applications. The results showed that the proposed adaptive interval DVFS technique could save power more with fewer frequency updates as compared to the fixed interval DVFS systems.

REFERENCES

[1] B. C. Mochocki, X. S. Hu, and G. Quan, "A unified approach to variable voltage scheduling for nonideal DVS processors," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 23, no. 9, pp. 1370–1377, Sep. 2004.

[2] S. Das, D. Roberts, S. Lee, S. Pant, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "A self-tuning DVS processor using delay-error detection and correction," *IEEE J. Solid-State Circuits*, vol. 41, no. 4, pp. 792–804, Apr. 2006.

[3] K. Choi, R. Soma, and M. Pedram, "Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 24, no. 1, pp. 18–28, Jan. 2005.

[4] T. Burd, T. Pering, T. Stratakos, and R. W. Brodersen, "A dynamic voltage scaled microprocessor system," *IEEE J. Solid-State Circuits*, vol. 35, no. 2, pp. 294–295, Feb. 2000.

[5] K. J. Nowka, G. D. Carpenter, E. W. MacDonald, H. C. Ngo, B. C. Brock, K. I. Ishii, T. Y. Nguyen, and J. L. Burns, "A 32-bit powerPC system-on-a-chip with support for dynamic voltage scaling and dynamic frequency scaling," *IEEE J. Solid-State Circuits*, vol. 37, no. 11, pp. 1441–1447, Nov. 2002.

[6] K. Flautner, D. Flynn, D. Roberts, and D. I. Patel, "IEM926: An energy efficient SoC with dynamic voltage scaling," in *Proc. Des., Autom. Test Euro.*, Feb. 2004, pp. 324–329.

[7] J. R. Lorch and A. J. Smith, "PACE: A new approach to dynamic voltage scaling," *IEEE Trans. Comput.*, vol. 53, no. 7, pp. 856–869, Jul. 2004.

[8] M. Marinoni and G. Buttazzo, "Elastic DVS management in processors with discrete voltage/frequency modes," *IEEE Trans. Ind. Inform.*, vol. 3, no. 1, pp. 51–56, Feb. 2007.

[9] S. Liu, Q. Qiu, and Q. Wu, "Energy aware dynamic voltage and frequency selection for real-time systems with energy harvesting," in *Proc. Des., Autom. Test Euro.*, 2008, pp. 236–241.

[10] J. Seo, T. Kim, and J. Lee, "Optimal intratask dynamic Voltage-Scaling technique and its practical extensions," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 25, no. 1, pp. 47–57, Jan. 2006.

[11] Z. Lu, Y. Zhang, M. Stan, J. Lach, and K. Skadron, "Procrastinating voltage scheduling with discrete frequency sets," in *Proc. Des., Autom. Test Euro.*, 2006, pp. 456–461.

[12] A. Andrei, M. Schmitz, P. Eles, Z. Peng, and B. M. Al Hashimi, "Quasi-static voltage scaling for energy minimization with time constraints," in *Proc. Des. Autom. Test Euro.*, 2005, pp. 514–519.

[13] J. Wegener and F. Mueller, "A comparison of static analysis and evolutionary testing for the verification of timing constraints," *Real-Time Syst.*, vol. 21, no. 3, pp. 241–268, Nov. 2001.

[14] M. Elgebaly and M. Sachdev, "Variation-Aware adaptive voltage scaling system," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 5, pp. 560–571, May 2007.

[15] S. Chandra, K. Lahiri, A. Raghunathan, and S. Dey, "Variation-tolerant dynamic power management at the system-level," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 9, pp. 1220–1232, Sep. 2009.

[16] M. Nakai, S. Akui, K. Seno, T. Meguro, T. Seki, T. Kondo, A. Hashiguchi, H. Kawahara, K. Kumano, and M. Shimura, "Dynamic voltage and frequency management for a Low-Power embedded microprocessor," *IEEE J. Solid-State Circuits*, vol. 40, no. 1, pp. 28–35, Jan. 2005.

[17] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," in *Proc. Low Power Electron. Des.*, Aug. 2007, pp. 38–43.

[18] J. Rabaey, *Low-Power Design Essentials*. New York: Springer, 2009.

[19] Q. Wu, P. Juang, M. Martonosi, and D. W. Clark, "Formal online methods for voltage/frequency control in multiple clock domain microprocessors," in *Proc. ASPLOS-XI*, Oct. 2004, pp. 248–259.

[20] P. Choudhary and D. Marculescu, "Power management of voltage/frequency Island-based systems using hardware-based methods," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 3, pp. 427–438, Mar. 2009.

[21] M. Najibi, M. E. Salehi, A. Afzali Kusha, M. Pedram, S. M. Fakhraie, and H. Pedram, "Dynamic voltage and frequency management based on variable update intervals for frequency setting," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2006, pp. 775–760.

[22] D. Burger and T. Austin, "The SimpleScalar tool set version 2.0," *Comput. Arch. News*, vol. 25, no. 3, pp. 13–25, Jun. 1997.

[23] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for Architectural-Level power analysis and optimizations," in *Proc. ISCA*, Jun. 2000, pp. 83–94.

[24] M. E. Salehi and S. M. Fakhraie, "Quantitative analysis of Packet-Processing applications regarding architectural guidelines for Network-Processing-Engine development," *J. Syst. Arch.*, vol. 55, no. 7–9, pp. 373–386, Jul.-Sep. 2009.

[25] R. Ramaswamy, N. Weng, and T. Wolf, "Analysis of network processing workloads," *J. Syst. Arch.*, vol. 55, no. 10–12, pp. 421–433, Oct.-Dec. 2009.

[26] J. L. Hennessy and D. A. Patterson, *Computer Organization and Design: The Hardware/Software Interface*, 3rd ed. Burlington, MA: Morgan Kaufmann, 2005.