Dynamic Voltage and Frequency Management Based on Variable Update Intervals for Frequency Setting

M. Najibi [I] M. Salehi [II] A. Afzali Kusha [II] M. Pedram [III] S. M. Fakhraie [II] H. Pedram [I]

{najibi, pedram}@ce.aut.ac.ir , {mersali, fakhraie}@ut.ac.ir , pedram@ceng.usc.edu

[I] Amirkabir University of Technology, 424 Hafez Ave , Tehran. [II] School of ECE, University of Tehran, North Kargar AVE. 1439957131

[III] Department of EE-Systems , University of Southern California, EEB-344 3740 McClintock Ave. Los Angeles

Abstract

An efficient adaptive method to perform dynamic voltage and frequency management (DVFM) for minimizing the energy consumption of microprocessor chips is presented. Instead of using a fixed update interval, the proposed DVFM 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 method, which is based on introducing the concept of an effective deadline, utilizes the correlation between consecutive values of the workload. In practice because the frequency and voltage update rates are dynamically set based on variable update interval lengths, voltage fluctuations on the power network are also minimized. The technique, which may be implemented by simple hardware and is completely transparent from the application, leads to power savings of up to 60% for highly correlated workloads compared to DVFM systems based on fixed update intervals.

Categories and Subject Descriptors

C.3 [SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS]: *Real-time and embedded systems* C.5.4 [COMPUTER SYSTEM IMPLEMENTATION]: *VLSI Systems*

Keywords

Power Management, Dynamic Voltage Scaling, Update Intervals

1. Introduction

The high demand from users for higher performance and more integrated applications and functions in portable devices has encouraged the designers to build faster and larger microelectronic systems such as System on Chips (SoCs) and Network on Chips (NoCs.) In these systems, energy consumption is a critical design concern from several points of view such as the battery life in case of portable systems where the long battery life is key important design parameter or heat dissipation in high end computing platforms. This has been the driving force for many design efforts devoted to reducing the power consumption while keeping the same performance. Due to the quadratic dependence of dynamic power consumption on the voltage, one of the most effective ways to reduce the energy consumption in CMOS processors is the dynamic voltage (and frequency) scaling (DVS or DVFS) that dynamically changes the frequency and the supply voltage of the processor.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ICCAD'06, November 5-9, 2006, San Jose, CA
Copyright 2006 ACM 1-59593-389-1/06/0011...$5.00.

Major chip makers including Transmeta, AMD, and Intel have recently presented commercial processors with the DVS capability [1]. Several research groups have also implemented their own DVS systems on commercial processors. For example, a DVS prototype processor system using ARM8 processor core has been proposed in [2][3]. In this system, software controls the clock frequency by writing to a register in the system control state. An SoC based on the PowerPC processor that manages both voltage and frequency under a software control has been implemented in [4][5]. In [6], an SoC based on ARM9 that uses the DVS technique is described. This SoC, includes software components that can predict the minimum necessary performance level of the processor for a running workload to prevent any performance degradation. A DVS architecture that adjusts the supply voltage according to throughput requirements is described in [7]. In this method, performance is controlled by software and voltage setting is achieved through a combination of a look up table (LUT) and performance monitoring via a critical path replica. To initialize the LUT, a calibration phase is required to identify the process corners under the worst case temperature conditions. In [8], the possibility of modifying a DVS algorithm to minimize the energy consumption without adversely affecting performance is reported.

All of the above techniques rely on software for estimating the required frequency and voltage, and thus, they are not completely transparent from the Operating System. In addition, different hardware conditions, such as the range of temperature and process variations, may not be easily observed by the software. This imposes the worst case values for frequency and voltage, which means that the voltage and the frequency may not be adjusted to the minimum possible values when the system does not experience the worst case conditions. In [9], a dynamic voltage and frequency management (DVFM) method for an ARM9 microprocessor is introduced. In this DVFM method, without any software intervention, the hardware dynamically controls the clock frequency and supply voltage with a fixed update interval.

In this paper, we present a frequency management method for DVFM approaches which is similar to [9]. The difference is that in our approach, the frequency update decisions are made based on variable timing intervals. The remainder of this paper is organized as follows. Section 2 presents the overall structure of dynamic voltage and frequency management system. In Section 3, we briefly discuss the concept of the dynamic frequency scheduling and Effective Deadline. Our proposed frequency adjustment algorithm based on the concept of the effective deadline is given in Section 4. Section 5 belongs to Results Analysis and finally in section 6 we sum up our work.

2. Dynamic Voltage and Frequency Management

The DVFM system in our work is based on the proposed method in [9]. Figure 1 shows the block diagram of their system which consists of a dynamic frequency controller (DFC) and dynamic voltage controller (DVC) units. The DVC emulates the critical-path characteristic of the system by using a delay synthesizer and controls the dynamic supply voltage. The DFC adjusts the clock frequency by monitoring the system activity. Since the voltage and frequency are predicted according to the performance monitoring of the system, the DVFM system can track the required performance with a high level of accuracy over the full range of temperature and process deviations.

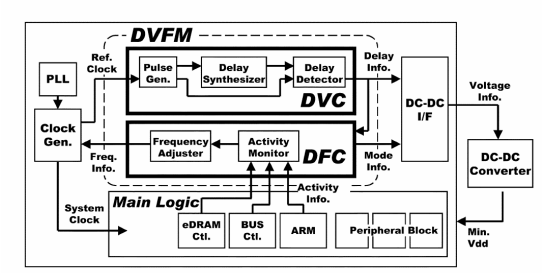


Figure 1. The block diagram of the DVFM system [9].

The DFC dictates the new working frequency according to the performance monitoring of the system. Since the new voltage is adaptively adjusted based on the predicted frequency, DFC plays the key role in the DVFM method. The next section describes the proposed method for the frequency scheduling.

3. DVFM: Background and a Key Concept

In synchronous dynamic voltage and frequency management systems, the clock frequency is usually adjusted according to the amount of the workload and the specified deadline. Next, in proportion to this frequency, a safe voltage level will be supplied to the system i.e., the voltage management is affected directly by the frequency scheduling. One of the major challenges of DVFM systems is how to determine the optimal working frequency required for maintaining the system load without disturbing the deadlines. While frequency underestimation will prevent the system from meeting the deadlines, an overestimated frequency results in a higher required voltage in addition to more idle CPU cycles and consequently more wasted power. From another point of view, undesired frequency fluctuations should be minimized in the frequency scheduling regime. This requirement minimizes the amount of the power loss in the power distribution network by minimizing unwanted voltage changes. The frequency scheduling circuitry is normally composed of two major units: an activity monitor and a frequency adjuster. By inspecting a subset of the system control signals, the activity monitor determines whether the system is in active (useful) or idle (useless) mode in each cycle. The output of the activity monitor is one for an active cycle and zero for the idle cycle. ALU control signals, register file read/write, or DMEM signals usually are included in the monitoring subset. Intuitively the output of the activity monitor can be considered as a signal with ones indicating active and zeros showing idle cycles. This signal is expected to be periodic due to the repetitious nature of some real-time applications, such as processing of MPEG4 encoded video streams.

3.1 Concept of an Effective Deadline

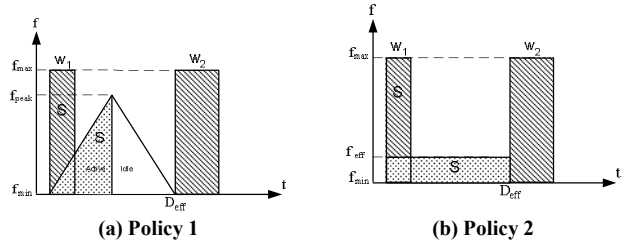
We introduce the concept of an "*effective deadline*" for periodic soft real-time systems. In hard real-time system, there exists an explicitly specified deadline for each workload and if this deadline is not met, the system may not function properly. In some applications, the real-time requirement is not that strict. i.e., it is simply required that the system is able to process the workload in an acceptable amount of time. For such applications, the deadlines may not even be mentioned explicitly. To meet the soft real-time constraint, it must be guaranteed that during the system operation, a hypothetical FIFO of moderate size placed at the input of the system never overflows. This can be assured by following a simple conservative rule, which states that the processing of the previous 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 previous workload. Notice that for the workloads that are highly correlated, it is possible to have a computational unit which can predict the exact arrival time of the next workload or equivalently the effective deadline of the current workload. Knowing the effective deadline for each workload, the frequency may be adjusted to its lowest value needed to meet the effective deadline. Clearly for an optimally adjusted frequency, size of the FIFO must be kept to a minimum to behave more like a real-time system.

3.2 Importance of the Effective Deadline

To clarify the importance of the effective deadline concept in the frequency adjustment, let us consider two different frequency scheduling policies shown in Figure 2. Amount of the workload can be represented by the area under the frequency-time plot.

Policy 1: As soon as the workload arrives, the frequency scheduler, without any consideration for the effective deadline, begins to increment the working frequency. The frequency incrementing will continue until the workload is processed completely and then scheduler will decrement the frequency. In the analysis, for the sake of simplicity, we assume that when the frequency returns to its initial value, the next workload arrives. Figure 2 (a) shows the frequency-time plot for this system in steady state (considering a sequence of identical workloads.) In this policy, f_{peak} can be calculated from $f_{peak} = k_f \frac{D_{eff}}{2}$ where k_f is the slope of frequency increase and D_{eff} is the effective deadline. Note that the amount of the workload is related to the deadline and the peak frequency through: $S = \frac{1}{2}\frac{D_{eff}}{2} \times f_{peak}$.

Policy 2: The scheduler in each period adjusts the frequency based on the knowledge of the effective deadline (arrival time of the next workload) as shown in Figure 2 (b)



(a) Policy 1 **(b) Policy 2**

Figure 2. Frequency-time plot.

We can compute the f_{eff} from $S = D_{eff} \times f_{eff}$.

To compare the two policies, we assume that the frequency is directly proportional to voltage ($f \propto V_{DD}$) which changes the power relation from $C.f.V_{DD}^2$ to $C.f^3$. The energy consumptions, E_1 and E_2, under these two policies may be computed as

$$E_1 = 2\alpha C \int_0^{\frac{D_{eff}}{2}} (k_f t)^3 \, dt = \frac{1}{4}\alpha C D_{eff} f_{peak}^3 \quad (1)$$

$$E_2 = \alpha.C.f_{eff}^3 D_{eff} \quad (2)$$

where α is the switching activity factor. Assuming the same workload for both policies, we must have the same area under the frequency-time plot which leads to $f_{peak} = 4f_{eff}$. Therefore, $E_1 = 16E_2$. This first order analysis shows that a significant reduction in the energy consumption is possible with an accurate prediction of the effective deadline.

4. Frequency Scheduling Methods

In this section, we briefly discuss the frequency adjustment method exploited in [9] and then present our frequency scheduling method based on the concept of the effective deadline.

4.1 Fixed Update Intervals

The method employed in [9] is proposed to reduce the idle cycles in fixed intervals. In this method, the frequency adjuster counts the number of active cycles in a predetermined fixed interval (e.g., 1μs.) When the number of the idle cycles in an interval exceeds a threshold value, frequency is lowered; otherwise, the frequency is increased. Figure 3 shows the details of this algorithm. The frequency update rate in this method directly depends on the value of the fixed interval. Larger values for the interval lead to lower slope of the frequency change and, hence, a weaker workload tracking ability. Consequently, there exists an upper limit for the fixed interval to maintain a good workload tracking capability.

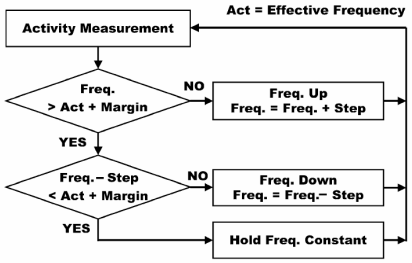


Figure 3. Details of fixed interval frequency adjustment [9].

Choosing small values for the fixed interval results in yet another undesirable effect for the periodic workloads (a periodic workload refers to a situation in which the workload has a fixed length and appears in a periodic manner with periods of inactivity in between.) If the fixed frequency update interval is set to a fraction of the workload period, then unnecessary updates will be performed during each workload period. More precisely, when the activity signal is high at the start of the active mode, the frequency is gradually increased until the workload is processed and the CPU becomes idle or until the frequency reaches its maximum allowed value. On the other hand, when the CPU enters its idle mode, its frequency is gradually decreased until it reaches the minimum allowed frequency or until a new workload arrives. Clearly, this frequency updating method is wasteful. 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 (it will eliminate them if the frequency values is set continuously and the next arrival time prediction is perfect.) Unnecessary updates may give rise to two difficulties. First, extravagant frequency changes can result in wasteful voltage fluctuations on highly capacitive power distribution network and, hence, a significant power loss. Second, since the activity signal will be '1' most of the time if the system frequency is scheduled properly, the number of frequency increases in a period will be usually higher than the number of frequency decreases. This means that, in the steady state, the average frequency cannot reach its minimum value. Figure 4 shows the behavior of the fixed interval frequency adjuster for a workload with a workload period of 66μs and a fixed frequency update interval of 1μs. With a DC-DC converter with maximum slope of 5mV/μs the voltage ripple is about 0.1V which can give rise to considerable power dissipation. Although increasing the value of the fixed interval may reduce the amount of voltage rippling it also reduces the ability of the system to track the workload.

4.2 Variable Update Intervals

We propose to use a simple adaptive frequency scheduling algorithm that decreases the frequency update interval to be able to track abrupt workload changes and increases this interval to minimize unwanted fluctuations for slowly changing workload. The algorithm works based on the concept of effective deadline introduced previously. It is thus critical to develop an online algorithm to calculate the effective deadline.

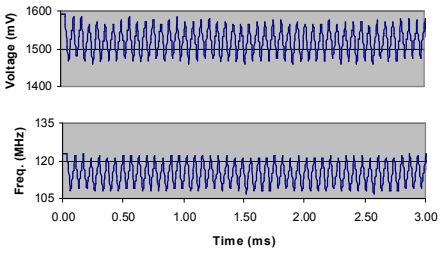


Figure 4. frequency-voltage variations in FI method.

4.2.1 Effective Deadline Prediction

In each workload cycle, a prediction of the arrival time of the next workload (effective deadline) is made based on an adaptive algorithm. The difference between two effective deadlines is considered as an adaptive interval for the frequency scheduler both to compute and update the frequency. The value of the adaptive interval is expressed in terms of the system clock cycle to eliminate the need for a separated fixed frequency clock generator for the time measurement.

A *counter* is used to compute the workload period in units of the clock cycles. The counter is always reset at positive edge of the

activity signal which corresponds to the arrival of next workload (see Figure 5.) As will be explained later, choosing update intervals that are well matched to workload characteristics leads to more optimum system frequencies and voltages. Updating the interval is performed either when the counter reaches the current value of the interval or at the positive edge of the activity signal, whichever comes first. When the former occurs (see Figure 5 (a)), since no positive edge of the activity signal has been observed, the *Update Interval* length should be increased by the amount of the *Interval Step*. When the latter condition occurs (see Figure 5 (b)), it is reasonable to decrease the *Update Interval* length by the amount of the *Interval Step*. It is important to point out that the frequency is updated at the end of the interval unless the positive edge occurs earlier.

The value of the interval step is determined adaptively such that when two consecutive interval increases (decreases) occur, the interval step will be multiplied (divided) by 2 using simple shift registers with zero detection capability. The upper limit for the interval step should be entered to the algorithm as an input parameter while its lowest limit is assumed to be zero. Using this adaptive method while the system behaves in quite averaging stable manner, the abrupt changes are tracked quickly.

4.2.2 Proposed Frequency Scheduling Method

The procedure for determining the proper value for the interval was explained above. Using the value of the interval, the optimum frequency scheduling may be performed by employing a shift register which always samples the activity signal to construct the workload history online. Figure 6 shows the details of the frequency adjustment using the workload history register. In this figure, the two predetermined thresholds values of *thH* and *thL* are used to determine the number (*thH* – *thL* + 1) of cycles that the workload history is examined. At the end of the interval the workload history is examined. If the positive edge of the activity signal resides between *thL* and *thH* in the workload history register (transition from 0 bits to 1 bits occurs), it is concluded that the number of idle cycles is not higher than the acceptable limit and, hence, there is no need to increase the frequency. If all the bits of the workload history register between *thL* and *thH* history are high, the frequency should be increased while if all the bits between *thL* and *thH* are zero, the frequency should be decreased to reduce the number of idle cycles.

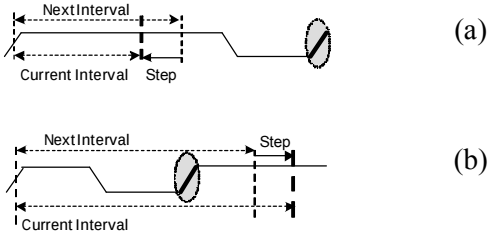


Figure 5. Effective deadline prediction. (a) interval decrement, (b)interval increment.

The margin for acceptable idle cycles is determined by the workload history register size (i.e., the difference between *thH* and *thL*) which can also affect the number of the required frequency updates.

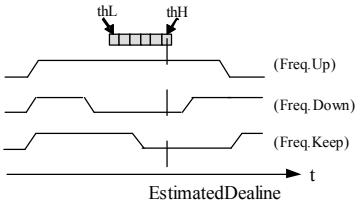


Figure 6. Adaptive frequency scheduling.

4.2.3 Overload and Underload States

It is worth mentioning that since the system frequency is adjusted based on the effective deadline, a quick yet accurate estimation of the effective deadline is crucial. If the effective deadline is not determined correctly, the frequency adjustment may malfunction and in some special cases the system cannot recover from its incorrect frequency scheduling. Two special cases that must be looked upon are overload and underload situations.

Overload State**:** When the frequency scheduler can not track the workload well and sets the frequency of the system to a value lower than the required value, the activity signal may continuously remain high i.e., there will be no idle cycles. Hence, the algorithm begins to increase the interval due to an implied assumption about the existence of a positive edge at infinity. Although a faster update is required to overcome this situation, the frequency changes become slower and slower which leads to much lower performance than required. To detect the overload situation, it is simply required to remember the direction of the last few interval updates which is five in our current system. If the overload condition is detected, the predicted update interval length will be replaced by the minimum possible interval value and all the frequency updates will be done in the direction of frequency increase. This leads to the frequency increase with the highest possible rate, and, hence, the workload can be tracked well until some idle cycles are generated by the activity signal which returns the system to the normal effective deadline prediction method. The effect of applying the overload mechanism on the interval changes is depicted in Figure 7 which shows the instantaneous value of the interval. As it can be seen unnecessarily large update intervals are avoided.

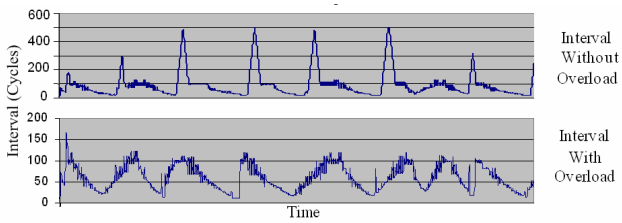


Figure 7. Overload effect on interval adjustment.

Underload State**:** If an abrupt workload decrease occurs when the update interval is large, the system cannot reduce its frequency at the rate of the workload change. For a better tracking of the workload in this situation, an underload detection method is used. The underload state can be detected by monitoring the *thH* bit of the workload history. By observing a predetermined number of consecutive zeros on *thH,* the underload condition is easily detected. In the underload situation, frequency and interval adjustment methods similar to those used in the overload situations are employed.

The overload and underload methods can be implemented by using shift registers and zero/one detection logic circuits.

5. Results and Discussion

To assess the efficiency of the proposed frequency scheduling method, we apply the method to periodic workloads. An important attribute of the DVFM system is how fast it can reach the final values of the frequency and the voltage. For the fixed update interval DVFM system [9], it is expected that small intervals yield faster convergence. Small intervals, however, give rise to large unnecessary updates and so larger final voltages and frequencies. On the other hand, larger intervals yield lower final frequencies and voltages with slower convergence. For comparison, we first adjust both the fixed update interval and the adaptive update interval DVFM systems such that they both can follow the workloads with nearly the same speed. To be able to do this, we initially apply a maximum frequency swing workload to both systems and adjust their parameters to track this workload at the same speed. When the value of the fixed interval is 7μs, the fixed interval system can work at the same speed (1.5ms for min to max workload swing) as the adaptive interval system with minimum interval of 1μs (123 cycles @123MHz.)

Figure 8 (a) and (b) depict the behaviors of the two systems for workloads of 20% and 80% of the maximum computational loads. Again the adaptive DVFM system can reduce the working frequency and the voltage significantly by utilizing the periodic nature of the workload. While it is expected that an 80% workload requires an optimal frequency of about 80% of the maximum frequency, the fixed interval system cannot distinguish between 80% and 100% workloads and, hence, the working frequency for 80% workload is not much lower than the maximum frequency. In the other words, assuming 100 frequency updates during an 80% workload period, 80 frequency increases and 20 frequency decreases are performed. Since the number of the frequency increases is higher than the number of the frequency decreases, it is expected that the system frequency eventually reaches its maximum. This behavior is expected for workloads with over 50% of the maximum computational load.

Table 1 details a comparison between the two DVFM systems for periodic workloads under different loads. As can be seen from the table, the working frequency for each workload is adjusted to the least required frequency. Since the maximum computational load can be processed with the maximum system frequency of 123MHz, it is expected that a 50% computational workload requires about half of this maximum frequency which can be verified in the table. The power consumption of the process for both fixed and adaptive update interval DVFM systems are shown in Figure 9(a) Our proposed method gives rise, on average, to about 60% power saving for the periodic workloads with different computational loads as shown in Figure 9 (b)

To explore the dynamic nature of our adaptive system, synthesized random workload patterns with different levels of auto-correlations are applied to the system. Since the effective deadline is predicted based on previous values of the amount of workload, it is expected that increasing the auto correlation leads to better prediction of the effective deadline and as a result more effective power reduction. In our experiment, a random workload with uniform probability density function in the range of 10% to 90% of the maximum allowed workload is produced. By reordering the sequence of the randomly generated workloads, different amount of auto-correlation in the workload is produced. Figure 10 shows the power and performance reduction of both fixed and adaptive interval frequency schedulers. Our system can reduce the power per operation metric 20-40% while keeping the performance nearly constant. There are exceptions for 0% and 10% auto correlations which are considered as totally random workloads. For these special cases our system will operate in lower performance while reducing the energy considerably.

6. Conclusion

In this work, an efficient adaptive method for the dynamic voltage and frequency management was proposed. The method which minimizes the energy consumption optimally schedules the frequency and voltage of the system for the periodic workloads 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 consequent workloads. The effective deadline was predicated using an adaptive technique tracking the workload. An overload/underload strategy in the frequency management was introduced for a fast tacking of abrupt workload changes. It should be noted that the new adaptive interval method usually converges to the optimum frequency and voltage in each working interval preventing incorrect predictions of the voltage which occurs in the fixed interval approach. A comparison between the adaptive and fixed interval methods showed power savings of up to 60% by lowering the average working frequency and voltage for highly correlated workloads. Depending on the amount of correlation for randomly generated workloads ranging from 10-90% of the maximum workload a power reduction from 20% to 40% for 50% or higher correlated workloads is obtained.

References

[1] B. C. Mochocki, X. S. Hu, and G. Quan, “A Unified approach to variable voltage scheduling for nonideal DVS processors,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 23, no. 9, Sept. 2004, pp. 1370-1377.

[2] T. D. Burd, R. W. Brodersen. “Design issues for dynamic voltage scaling,” *Proc. 2000 Int'l Symp. Low Power Electronics and Design,* July 2000, pp. 9-14.

[3] T. Burd, T. Pering, et al., "A Dynamic voltage scaled microprocessor system," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, Feb. 2000, pp.294-295.

[4] K. Nowka, G. D. Carpenter, et al., “A 0.9 V to 1.95 V dynamic voltage-scalable and frequency-scalable 32-bit PowerPC processor,” *in IEEE Int. Solid-State Circuits Conf.* Feb. 2002, pp. 340–341.

[5] K. J. Nowka, G. D. Carpenter, et al., "A 32-bit PowerPC system-on-a-chip with support for dynamic voltage scaling and dynamic frequency scaling," *IEEE Journal of Solid-State Circuits,* vol. 37, no. 11, Nov. 2002, pp. 1441-1447.

[6] K. Flautner, D. Flynn, et al., “IEM926: an energy efficient SoC with dynamic voltage scaling,” *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition Designers’ Forum,* Feb. 2004, pp. 324-329.

[7] M. Elgehaly, A. Fahim, et al., “Robust and efficient dynamic voltage scaling architecture,” *IEEE International System on Chip Conference,* Sep 2003, pp. 155-158.

[8] J.R. Lorch, A.J. Smith, "PACE: A new approach to dynamic voltage scaling," *IEEE Transactions on Computers,* vol. 53, no. 7, July 2004, pp. 856-869.

[9] M. Nakai, S. Akui, et al., "Dynamic voltage and frequency management for a low-power embedded microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 1, Jan. 2005, pp 28-35.

Table 1. Comparison results for the fixed and adaptive interval methods.

	Fixed Interval			Adaptive Interval			
Workload	**Frequency (MHz)**	**V_{DD} (mV)**	**Power (mW)**	**Frequency (MHz)**	**V_{DD} (mV)**	**Power (mW)**	**Power Reduction**
0%	8	600	0.95	8	600	0.5	47.37%
10%	21	875	2.16	12	810	0.92	57.41%
20%	41	990	4.57	25	895	2.17	52.52%
30%	61	1135	8.51	36	945	3.73	56.17%
40%	82	1285	14.01	49	1050	5.89	57.96%
50%	102	1430	21.07	61	1121	8.38	60.23%
60%	123	1580	30.02	74	1230	11.51	61.66%
70%	123	1585	30.59	86	1315	15.32	49.92%
80%	123	1585	30.77	98	1400	19.52	36.56%
90%	123	1585	30.89	110	1485	24.68	20.10%
100%	123	1580	30.92	123	1585	30.88	0.13%

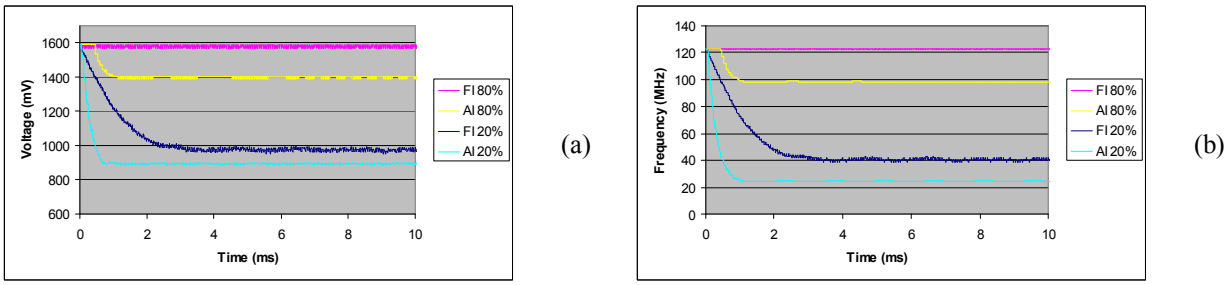


Figure 8. Voltage-Frequency variations for 20% and 80% workload in fixed interval (FI) and adaptive interval (AI) methods.

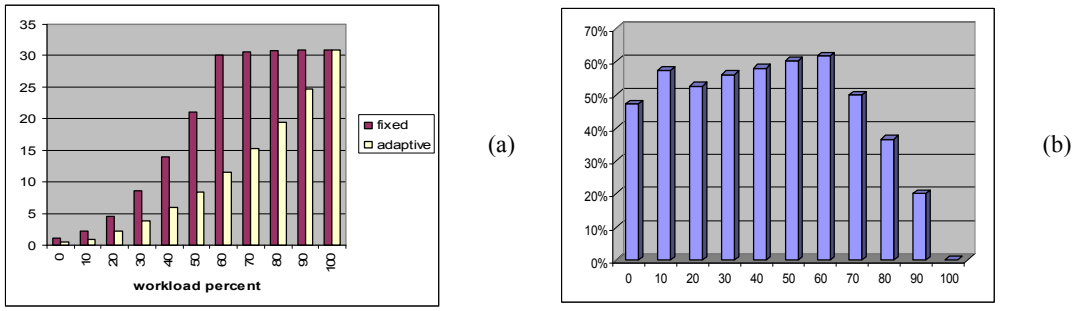


Figure 9. Fixed and adaptive intervals (a) Power consumption and (b) Power saving as a function of the workload percentage

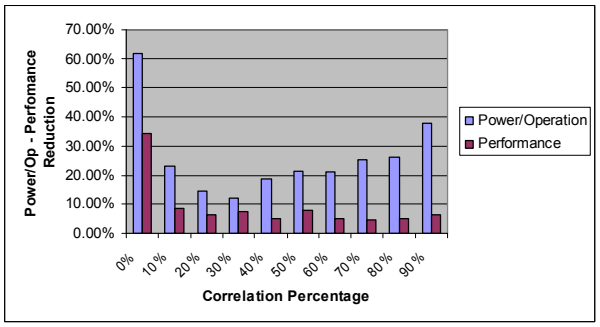


Figure 10. Power per operation and performance reduction.