Low-Complexity Policies for Energy-Performance Tradeoff in Chip-Multi-Processors

Avshalom Elyada, Ran Ginosar, Uri Weiser

Abstract— **Chip-Multi-Processors (CMP) utilize multiple energy-efficient Processing Elements (PEs) to deliver high performance while reducing energy-consumption. Dynamic frequency-Voltage Scaling (DVS) balances performance and energy consumption by varying PEs' frequency-voltage *workpoints* to save energy while meeting performance requirements. We consider multi-task CMP applications with unknown workloads, and dynamically set workpoints to minimize ET^2. Heuristic policies for serial/parallel task-graphs are investigated. We compare these policies to a theoretical bound and show that they achieve good results with low complexity. In most cases the simplest policy, which usually assigns constant workpoints, is also the most cost-effective one.**

Index Terms—**Chip-Multi-Processors, Dynamic Voltage Scaling, Slack Utilization, Low Power.**

I. Introduction

CHIP-Multi-Processors (CMP) architectures integrate multiple relatively small energy-efficient PEs, to enable energy-efficient performance scaling [1-3]. Dynamic frequency-Voltage Scaling (DVS) is a widely practiced [4] and researched [5] technique for energy-performance tradeoff in which a PE's frequency is altered dynamically to meet performance requirements while consuming limited energy. The supply voltage of each PE is kept at the lowest feasible value for the selected frequency. Scaling the frequency-voltage workpoint (*f,V*) can result in near-quadratic energy savings [6].

In a CMP running multiple dependent tasks, DVS may save energy without degrading performance. At any given time, one or few tasks typically constitute(s) the performance bottleneck while other PEs can be slowed down without affecting total performance. We refer to slowing down non-critical tasks in CMP as *slack-utilization* [7, 8].

When task workloads are known in advance, the *DVS policy* sets workpoints to utilize precisely all available slack. But when task workloads are unknown in advance, slack utilization is non-trivial. Assuming worst-case workloads achieves limited energy savings since overly aggressive workpoints are needed. Conversely, overestimating slack can lead to performance degradation. Workload estimation is a primary factor of the efficiency of DVS policies.

Measuring DVS policy efficiency depends on the energy/power/performance requirements of target applications. Mobile battery-operated systems strive to extend battery life, while desktop systems and servers typically optimize power rather than energy. While a maximum performance system would always run at the maximum frequency (the *f-max* policy), a minimal energy system should execute at the minimum frequency (the *f-min* policy). A system which strives to balance energy and performance could maneuver *(f,V)* between justifiable energy consumption at the high end and tolerable performance at the low end. This choice is reflected in the criterion selected to assess alternative DVS policies.

If the required calculations for policy implementation are to be integrated into the system itself and performed in real-time, then it is essential that energy-consumption and delay of the policy calculation itself be minimal. Spending a substantial amount of execution time and energy merely to calculate workpoints may considerably offset the savings aimed at, making it impractical for a performance- and energy-aware system. Thus, low computational complexity is a key concern.

Previous DVS policies [9-12] for CMP attempt to formulate optimization problems which have high computational complexity. In contrast, we introduce lightweight heuristic policies, and show that they achieve good results compared to theoretical bounds. We further show that in most cases the simplest policy is the most cost-effective one.

This paper is organized as follows. Section II defines the minimization criterion and formulates the DVS minimization problem. The DVS policies are presented in Section III, and analyzed in Section IV. Conclusions are drawn in Section V.

II. Definitions and Problem Formulation

Consider a CMP running a multi-task application. Assume for simplicity that at any time, each PE accommodates one software task. Each PE's *(f,V)* workpoint is controlled independently of other PEs and all PEs are identical, although the method of this study may be generalized to *heterogeneous*-PE systems [3, 13, 14] with few modifications [15].

Each PE can operate in the frequency range $f \in [f_{min}, f_{max}]$ cycles/sec, or be in a standby mode. The PE operates at the minimum feasible supply voltage for its frequency, defining a frequency-voltage (*f,V*) curve [15]. A continuous frequency model is employed. Furthermore, the rate of changing frequency is limited in practice due to the energy and time penalties of making the transition, and the added complexity of calculating a new workpoint [9]. We neglect workpoint

Manuscript received March 24, 2007. Revised manuscript received July 20, 2007.

The authors are with the Electrical Engineering Department, Technion–Israel Institute of Technology, Haifa 32000, Israel (e-mail: avshael@tx.technion.ac.il).

transitions here, and later mention the effect of considering them on the final conclusions.

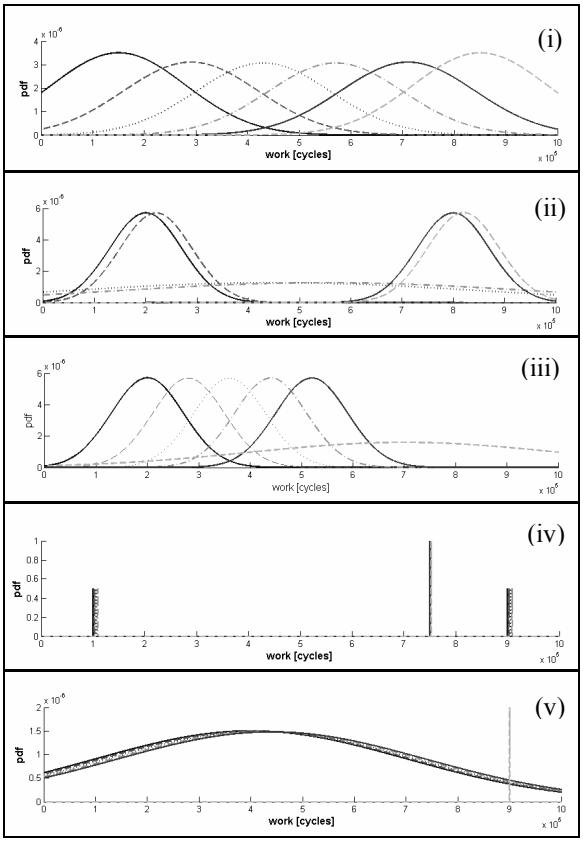

Figure 1: Workload distributions: Examples on a 6-PE CMP.

For a task of unknown workload, the cumulative density function $cdf_W(w)$ of the workload W is the probability that the task will be completed within w or fewer cycles: $cdf_W(w)=Pr(W \leq w)$. Hence the probability that the task will take w cycles or more to execute is $1\text{-}cdf_W(w\text{-}1)$. Some example distributions are displayed in Figure 1 above, which shows probability density functions $pdf_W(w)=cdf'_W(w)$. These distributions are used in our simulations, as described in Section IV. In practical cases, the *pdf* is estimated based on previous task instances [15, 16].

The total power consumption of a PE running at frequency f is $P(f)$ joules/sec, where power consists of active and standby power: $P(f) = P_{act}(f) + P(0)$. The *active energy-per-cycle* of the PE is $e_{act}(f) = P_{act}(f) / f$. Suppose that the task starts at time $t = 0$ and completes by time $t = \mathbb{T}$. If it completes before that time, the PE goes to standby until $t = \mathbb{T}$. Given $e_{act}(f)$ and $cdf_W(w)$ or their estimations, we formulate the expected energy required to execute a task of workload W on a PE p:

$$E_p = \sum_{w=1}^{\infty} e_{act}\left(f_w\right)\left[1 - cdf_W(w-1)\right] + \mathbb{T}P(0), \qquad (1)$$

where f_w is the frequency at cycle w. Eq. (1) sums the active energy-per-cycle times the probability that the task will still be running at that cycle, adding the total standby energy. Following similar arguments, the expected execution time is:

$$T_p = \sum_{w=1}^{\infty} \frac{1 - cdf_W(w-1)}{f_w}. \qquad (2)$$

The application is modeled as alternating serial and parallel phases [3, 17-19] (Figure 2). We focus on DVS policies for the parallel phases. Tasks of equal workloads running on identical PEs at the same *(f,V)* workpoint will incur identical run-times. While DVS of symmetrical parallel phases (Figure 2a) necessarily results in performance trade-off, asymmetrical parallel phases (2b) offer a potential for energy saving *without* performance degradation (2c).

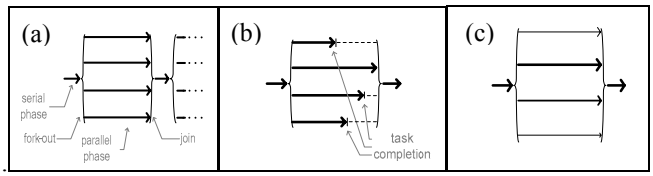

Figure 2: An execution timeline with (a) equal workloads, (b) unequal workloads (c) unequal workloads where slack is utilized to balance run times (thinner lines represent lower *(f,V)* workpoints).

In a system of N PEs, the total system energy $\mathbb{E}$ and the combined execution time $\mathbb{T}$ of a parallel phase are:

$$\mathbb{E} = \sum_{p=1}^{N} E_p \quad , \quad \mathbb{T} = \max_{p=1..N} T_p \,. \qquad (3)$$

All tasks that complete before the execution time $\mathbb{T}$ of the critical PE waste energy, since they could have run slower without degrading performance. We utilize the slack of non-critical PEs in order to save energy (Figure 2c).

We employ the criterion of minimal expected value of $\mathbb{ET}^2$, which has the useful characteristic of *frequency invariance* [6], since $\mathbb{E} \propto f^2$ and $\mathbb{T} \propto 1/f$. $\mathbb{ET}^2$ is a widely used criterion for design-spaces that consider scaling of frequency-voltage, both at the circuit [20] and system [21] levels. It is therefore specifically suited for fair comparison of DVS policies. We have not found it useful to use $\mathbb{ET}^{\rho}$ with $\rho \neq 2$. Using $\rho < 2$ gives inherent advantage to the *f-min* policy, while $\rho > 2$ inherently favors the *f-max* policy.

Thus, the optimal policy can be formulated as:

$$\begin{aligned} &\min \mathbb{ET}^2 \\ &s.t.\ f_p(w) \in 0 \cup \left[f_{\min}, f_{\max}\right], \\ &\forall p = 1..N,\ \forall w = 1, 2..\infty \end{aligned} \qquad (4)$$

Rigorous analysis of this stochastic optimization problem is beyond the scope of this work. In the following we employ heuristic policies to achieve computationally feasible solutions.

III. DVS POLICIES

In this section we describe a group of simple DVS policies for a CMP. We first describe a common outline (Figure 3) for all policies, and then specify the differences for each policy.

At t=0 fork-point of the parallel phase in the timeline, all policies perform the same steps:

Step (1): We compute the expected value of the (remaining) estimated work for all PEs:

$$\hat{W}_{p,rem} = E[W_p \mid W_{p,comp}] \quad (5)$$

where $W_{p,comp}$ is zero at t=0. We subsequently find the *estimated-critical PE* (ECP), namely the PE (task) estimated to have the most work:

$$\hat{W}_{ECP,rem} = \max_{p=1..N}\{\hat{W}_{p,rem}\}$$
$$ECP = \arg\max_{p=1..N}\{\hat{W}_{p,rem}\} \quad (6)$$

Step (2): The ECP is assigned f_{max} in order to minimize standby power. If standby power were neglected, any frequency could be used because $\mathbb{ET}^2$ is frequency-invariant. However, when f_{max} is assigned to ECP the total run time is minimized and thus the contribution of standby power to the total energy is also minimized [15].

The joint-target-time (JTT) for completion of all tasks is the expected completion time of the ECP:

$$JTT = \widehat{W}_{ECP,rem} \Big/ f_{\max} \quad (7)$$

Frequency assignment of the other PEs differs per policy and is described per each policy below.

Step (3): All PEs run at their assigned frequencies until they complete (except in the case of the *Interval* policy described below), or until the ECP completes. The ECP is expected to complete last but since workloads are stochastic this may not be the case.

Step (4): If the ECP completes (or, for *Interval*, a time-interval elapses) while other PEs still have remaining work, re-estimation is performed, taking into account the work done by each PE so far. The cycle is repeated until all PEs complete their work (Figure 3).

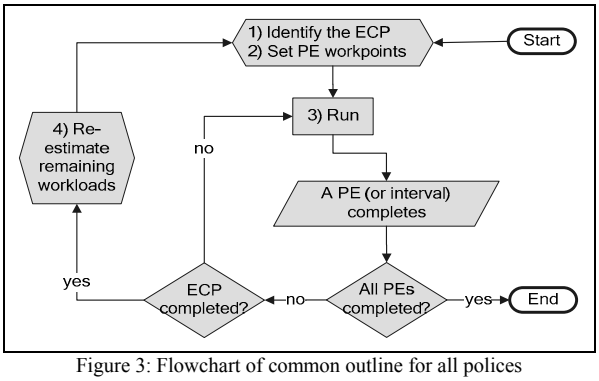

Figure 3: Flowchart of common outline for all polices

We now consider each individual policy. The *Oracle* policy is a non-causal, hypothetical policy for which workloads are known in advance. We use Oracle as a best-case for comparing to real policies. Workloads are known, so energy can be minimized by running each PE at the slowest constant workpoint that is still fast enough to finish the task at the JTT. This well established result is due to the convexity of $e(f)$, the energy-per-cycle [22]. The optimal frequencies f_p assigned to each PE are:

$$f_p = W_p / T = f_{\max} W_p \Big/ \max_p(W_p),\ p = 1..N. \quad (8)$$

The *Constant* policy assigns a constant frequency to each PE. We set the frequencies of non-critical PEs with an aim to complete at the joint-target-time:

$$f_p = \frac{\widehat{W}_{p,rem} + \lambda\widehat{\sigma}_{p,rem}}{JTT} = \frac{\widehat{W}_{p,rem} + \lambda\widehat{\sigma}_{p,rem}}{\widehat{W}_{ECP,rem}} f_{\max} \quad (9)$$

where $\widehat{W}_{p,rem}$ is the remaining work estimation of PE p, $\hat{\sigma}_{p,rem}$ is the standard deviation of $W_{p,rem}$, and $\lambda \geq 0$ is the *bias parameter*. At time *t=0* no work has been done so the remaining work is the total work.

For $\lambda=0$, f_p is set so that PE p completes $\widehat{W}_{p,rem}$ work during the time it takes the ECP to complete $\widehat{W}_{ECP,rem}$ work. However since delay outweighs energy for the $\mathbb{ET}^2$ criterion, we can achieve better results by setting the bias parameter $\lambda>0$ [15].

If the ECP completes while other PEs still have remaining work (step (3) above), then the assumptions by which frequencies were assigned in step (2) no longer hold. We therefore update the estimations $\widehat{W}_{p,rem}$ and $\widehat{\sigma}_{p,rem}$ to reflect the work completed so far, and repeat steps (1) to (4) until all PEs complete. Figure 4 shows a Constant policy example.

If we regard the complexity of one (f,V) workpoint assignment as $O(1)$, then the complexity of Constant is $O(N)$, N being the number of PEs in the system. [15].

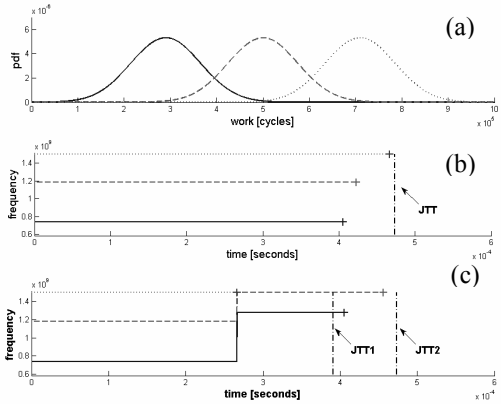

Figure 4: Constant policy example with three PEs: (a) workload distributions, (b) the ECP (dotted line) completes last as expected, (c) the ECP completes first, and re-estimation is performed.

The *Interval* policy assigns constant workpoints as in the Constant policy (Eq. (9)), but the critical PE is re-chosen and workpoints are re-assigned at intermediate fixed time intervals. At each time interval, estimated *remaining* workloads are used to choose the critical PE and frequencies for the next interval. Re-estimation may offer a significant advantage since the ratios between the estimated workloads at time t=0 may vary significantly from those at a later time.

A bias parameter λ for the *Interval* policy allows tuning as in the Constant policy. Figure 5 shows an Interval policy example. The frequency of non-critical PEs typically increases with time, similar to the behavior of PACE [16], but decreasing frequencies can also occur [15]. Since the complexity of Constant is $O(N)$, the complexity of Interval is $O(kN)$, where k is the number re-estimation intervals.

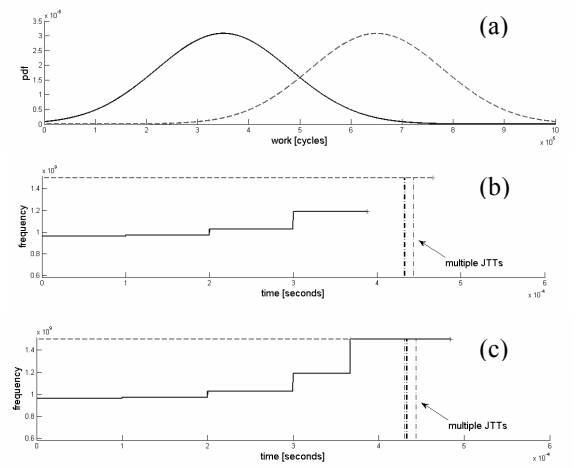

Figure 5 : Interval policy example with two PEs: (a) workload distributions, (b) the ECP (green, dashed) finishes last as expected, (c) the ECP finishes first.

Energy-performance tradeoff in a single PE has been studied extensively [16, 23, 24]. A new *Multi-PACE* policy presented here is a heuristic generalization of the single-PE PACE policy [16] for CMPs. PACE stands for Processor Acceleration to Conserve Energy, reflecting the fact that the optimal frequency function is increasing when workloads are unknown. PACE minimizes energy subject to meeting a deadline D with probability PMD (Probability of Meeting the Deadline). The PACE optimal frequency function is:

$$f(w) = PACE([f_{\min}, f_{\max}], pdf_W(w), D, PMD) \qquad (10)$$

In Multi-PACE, non-critical PE frequencies are set according to PACE with the JTT as the deadline:

$$f_p(w) = PACE([f_{\min}, f_{\max}], pdf_{W_{p,rem}}(w), JTT, PMD) \qquad (11)$$

Note that JTT is computed following Eq. (7) and is *not* an application deadline. Multi-PACE uses the PACE deadline mechanism to synchronize completion times between PEs. PACE does not specifically define which frequency to use for post-deadline cycles ($w>w_{PMD}$). Multi-PACE runs at f_{max} during the cycles following w_{PMD} to minimize delay past JTT. shows a Multi-PACE example.

PMD has a significant effect on the $\mathbb{ET}^2$ criterion, similarly to the λ bias parameter used in Constant and Interval. Setting it too high incurs excessive energy consumption, while setting it too low increases the probability of missing the JTT, thereby increasing overall execution time [15].

Multi-PACE requires that PEs have a continuous frequency range and be able to change frequency every cycle, which is impractical. Practical methods of implementing PACE, which can apply to Multi-PACE as well, are described in [16, 24]. The number of frequency changes in Multi-PACE is in practice proportional to some number B (which may represent the number of histogram bins used to collect previous instance data), thus the complexity of Multi-PACE is $O(BN)$ [16].

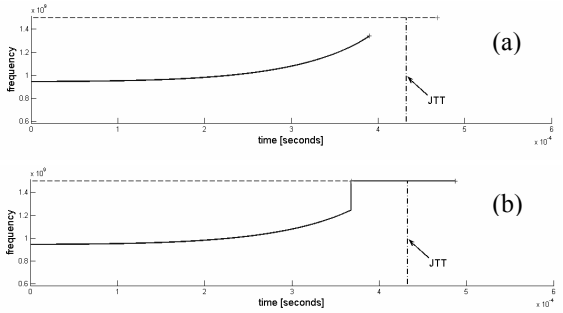

Figure 6: Multi-PACE example with two PEs, workload distributions as in Figure 5: (a) ECP (dashed line) completes last as expected; (b) ECP completes first.

IV. Simulations and Results

We simulated the various DVS policies on a system with six identical PEs, each with a continuous frequency range of 0.32GHz to 1.5GHz, using the workloads of Figure 1. Energy was calculated according to the power function $P(f) = af^3+b$ with a=0.8 [watt/Ghz^3], b=0.2 [watt]. For the Interval policy 16, 8, 4, and 2 intervals were simulated. We chose λ=0.8 for Constant, λ=0.5 for Interval and PMD=80% for Multi-PACE, values which empirically minimize $\mathbb{ET}^2$ [15].

For each distribution, Figure 7 compares $\mathbb{ET}^2$ across all policies, normalized to the results of Oracle. Any fixed-workpoint policy, such as the *f-min* and *f-max* policies, reaches the same results in $\mathbb{ET}^2$ for all distributions, owing to the frequency-invariance of $\mathbb{ET}^2$. The results of all policies lie between Oracle and *f-max*/*f-min*, i.e., better than running at an arbitrary fixed frequency, but worse than the ideal case where workloads are known in advance. The Interval policy usually achieves the best results while Constant usually achieves the worst results, and Multi-PACE typically lies in-between them. However, the difference between these policies is generally quite small (4-13%).

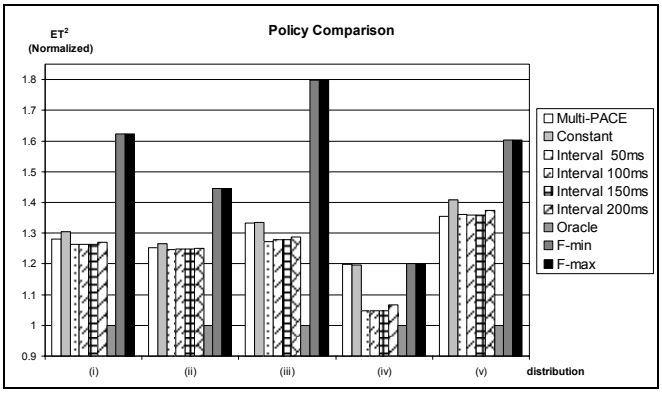

Figure 7: ET^2 compared across all policies

Distribution (*iv*) of Figure 1 is an example where the Interval policy stands out. If the bimodal tasks in (*iv*) complete $1 \cdot 10^5$ cycles of work and do not finish, their remaining estimated workload changes sharply to a deterministic $9 \cdot 10^5$ cycles. The re-estimation performed by Interval takes full

advantage of this, setting a slow, constant frequency to complete the task precisely at the JTT. The results show that a few re-estimations over a relatively large period can generally make a significant difference, while additional re-estimations have only a marginal effect.

Multi-PACE typically achieves better results than Constant and worse than Interval in the simulated examples. Multi-PACE is more dependent on accurate estimation of the critical task: consider for instance distributions (*v*) versus (*iii*).

Regarding computational complexity, observe that a relatively small number of re-estimations k are needed for Interval, and that since Multi-PACE performs no re-estimations it needs a relatively large number of histogram bins B [16]. With $k<<B$ the relative complexity of the policies is $O(N) < O(kN) << O(BN)$.

V. Summary and Conclusions

In this work we presented several DVS policies for CMPs, and demonstrated that simple policies achieve good results compared to more complex ones, and are within approximately 35% of optimal bounds. We started by formulating an energy-performance optimization problem of an application running on a CMP and noted the complexity of the problem, which makes it impractical for implementation. As an alternative, we described several heuristic DVS policies which utilize available time-slack to save energy in a performance-aware manner: *Constant*, *Interval*, and *Multi-PACE*. The frequency-invariant $\mathbb{ET}^2$ criterion was employed for comparing the policies. Except for isolated cases, all policies reach comparable results. Increasing the number of re-estimations (using Interval) bears only a marginal improvement. Multi-PACE produces results that are anywhere between Interval and Constant, depending on the distribution. Since the results are usually quite close for all policies, we conclude that the least complex policy, Constant, is usually preferred. Use of Interval is justified only for certain distributions which highly benefit from re-estimation, and should be weighed against the added complexity. Based on these findings, a scheme could be contemplated whereby the number of intervals is chosen dynamically based on certain characteristics of the distribution or on past results. Multi-PACE generally does not achieve better results than any of the other two, and thus is not preferred due to its high complexity.

Taking workpoint transitions into account may degrade the results since each transition incurs performance and energy penalties [9]. When transition costs are considered, simple policies such as Constant become even more attractive because they employ fewer transitions.

Several topics are proposed for future research. More complex task graphs may be considered. Discrete (f,V) workpoints could be studied. The Interval policy may be enhanced to consider re-estimation at flexible times. Test cases based on real application traces can be employed. Applications may be modified to estimate their own remaining work.

References

[1] F. Pollack, "New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies," *Micro 32*, 1999. Available: http://www.intel.com/research/mrl/Library/micro32Keynote.pdf

[2] E. Grochowski *et al.*, "Best of Both Latency and Throughput," *Proc.* ICCD, pp., 2004.

[3] T. Y. Morad *et al.*, "Performance, Power Efficiency and Scalability of Asymmetric Cluster Chip Multiprocessors," *IEEE CAL'06*, vol. 5.

[4] "Intel Enhanced SpeedStep(R) Technology". Available: http://www.intel.com/support/processors/mobile/pentium4/sb/CS-007499.htm

[5] T. Pering *et al.*, "The simulation and evaluation of dynamic voltage scaling algorithms," *Proc.* ISPLED'98, pp. 76-81.

[6] A. J. Martin *et al.*, "ET^2: A metric for time and energy efficiency of computation," *Power Aware Computing*, *Series in Computer Science*. Kluwer Academic Publishers, 2002, pp. 293-315.

[7] J. Li *et al.*, "The Thrifty Barrier: Energy-Aware Synchronization in Shared-Memory Multiprocessors," *Proc.* HPCA'04, pp. 14-23.

[8] C. Liu *et al.*, "Exploiting Barriers to Optimize Power Consumption of CMPs," *Proc.* IPDPS'05, pp. 5.1, 2005.

[9] A. Andrei *et al.*, "Overhead-conscious voltage selection for dynamic and leakage energy reduction of time-constrained systems," *IEE Proc. Comp. & Digital Tech. '05*, vol. 152, pp. 28-38.

[10] R. Xu *et al.*, "Minimizing expected energy in real-time embedded systems," *Proc.* EMSoft'05, pp. 251-254.

[11] S. Yaldiz *et al.*, "Characterizing and exploiting task load variability and correlation for energy management in multi core systems," *Proc.* ESTIMedia'05, pp. 135-140.

[12] D. Zhu *et al.*, "Scheduling with Dynamic Voltage/Speed Adjustment Using Slack Reclamation in Multi-Processor Real-Time Systems," *IEEE TPDS '03*, vol. 14, pp. 686-700.

[13] R. Kumar *et al.*, "Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload Performance," *Proc.* ISCA'04, pp. 64-75.

[14] S. Ghiasi *et al.*, "Scheduling for heterogeneous processors in server systems," *Proc.* CF'05, pp. 199-210.

[15] A. Elyada *et al.*, "Low Complexity Policies for Energy-Performance Tradeoff in Chip-Multi-Processors," *EE*. Techion IIT, 2007. Available: http://avshalom.elyada.googlepages.com/CmpDvsIE3VLSI_full.pdf

[16] J. R. Lorch and A. J. Smith, "PACE: a new approach to dynamic voltage scaling," *IEEE Trans. Comp.*, vol. 53, pp. 856-869, 2004.

[17] M. L. Crow and M. Ilic, "The parallel implementation of the waveform relaxation method for transient stability simulations," *IEEE Trans. Power Sys. '90*, vol. 5, pp. 922-932.

[18] R. A. Saleh *et al.*, "Parallel circuit simulation on supercomputers," *Proc. IEEE '89*, vol. 77, pp. 1915-1931.

[19] L. Dagum and R. Menon, "OpenMP: an industry standard API for shared-memory programming," *IEEE Comp. Science & Eng. '98*, vol. 5, pp. 46-55.

[20] J. Hensley *et al.*, "An area- and energy-efficient asynchronous Booth multiplier for mobile devices," *Proc.* ICCD'04, pp. 18-25.

[21] V. Salapura *et al.*, "Power and performance optimization at the system level," *Proc.* CF'05, pp. 125-132.

[22] J. Blazewicz *et al.*, *Scheduling Computer and Manufacturing Processes*. Berlin: Springer-Verlag, 1996.

[23] R. Jejurikar *et al.*, "Leakage aware dynamic voltage scaling for real-time embedded systems " *Proc.* DAC'04, pp. 275-280.

[24] R. Xu *et al.*, "Practical PACE for embedded systems," *Proc.* EMSoft'04, pp. 54-63.