Exploring Power Management in Multi-Core Systems

Reinaldo Bergamaschi[1], Guoling Han[2], Alper Buyuktosunoglu[1], Hiren Patel[3], Indira Nair[1], Gero Dittmann[1], Geert Janssen[1], Nagu Dhanwada[4], Zhigang Hu[1], Pradip Bose[1], John Darringer[1]

[1] IBM T. J. Watson Research Center, Yorktown Heights, NY 10598; [2] University of California, Los Angeles, CA 90095;

[3] Virginia Tech, Blacksburg, VA 24060; [4] IBM STG, East Fishkill, NY 12533

berga@us.ibm.com

Abstract— Power dissipation has become a critical design metric in microprocessor-based system design. In a multi-core system, running multiple applications, power and performance can be dynamically traded off using an integrated power management (PM) unit. This PM unit monitors the performance and power of each core and dynamically adjusts the individual voltages and frequencies in order to maximize system performance under a given power budget (usually set by the operating system). This paper presents a performance and power analysis methodology, featuring a simulation model for multi-core systems that can be easily reconfigured for different scenarios and a PM infrastructure for the exploration and analysis of PM algorithms. Two algorithms have been implemented: one for discrete and one for continuous power modes based on non-linear programming. Extensive experiments are reported, illustrating the effect of power management both at the core and the chip level.

I. Introduction

Technology scaling has continuously driven towards higher levels of integration, higher frequencies and lower operating voltages. For technologies down to 90 nm it has been possible to continue increasing performance while reducing power for the same functionality from one processor generation to the next. However, for 65 nm and below, the effect of increased interconnection length and resistance, coupled with a relatively flatter operating voltage, has caused a significant dynamic and static power increase in complex chips.

As the power / performance curve for microprocessor-based systems is flattening across processor generations [1], techniques for managing power and performance are needed both at the low-end and high-end. Low-end embedded systems need to be low power because of battery constraints. High-end systems can benefit greatly from lower power consumption, reducing packaging and cooling costs from individual processors to complete data centers.

Several power management approaches have been developed, covering a wide spectrum of system characteristics, including: (a) high-level operating-system-driven policies for turning devices (e.g., disks, modems) on and off according to predicted usage [2], (b) dynamic management of processor resources according to activity demands [3, 4], (c) dynamic scheduling of tasks to processors in a chip multi-processor (CMP) environment to manage power and temperature [5, 6], (d) hardware techniques for dynamic voltage and frequency scaling [7, 8, 9], and (e) global dynamic power management for multi-core chips [10].

Given the recent trend towards single-chip, multi-core systems, dynamic power management techniques that were designed for single-core microprocessors must be augmented at the chip-level to exploit the larger design space. This enables a better power-performance trade-off under top-level constraints, e.g., a power budget. The hardware actuators available for power management include: (a) joint voltage and frequency scaling, (b) frequency scaling, and (c) microarchitectural switches, e.g., instruction fetch throttling [11].

In multi-core systems these actuators can be applied to the chip as a whole (i.e., same value for all cores), or individually per core. From a hardware implementation point of view, applying voltage and frequency scaling on a per-core basis is significantly more expensive than chip-wide, because it requires expensive analog voltage regulators and phase-locked loops for each core. However, the per-core variant provides more precise control of the performance / power curve, leading to a better trade-off.

While substantial research has been done on power management techniques at the hardware and architectural levels for microprocessor-based systems, little has been done at the tools level for power management. There are tools for power estimation of processor cores [12], caches [13] and interconnects [14], as well as simulators for performance and power [15, 16]. One of the few works on specification and analysis of power-managed systems is [17], but it focuses on static definitions of states / transitions and modes for each system component, as opposed to values dynamically updated according to varying workloads. Most of these tools focus on individual components (e.g., core, cache), and not on complete multi-core systems, including buses and memory hierarchy. However, it is at the system level that power management can be the most effective.

This paper presents a performance and power analysis methodology based on a simulation model for multi-core systems with integrated power management. The methodology is implemented in SLATE (System-Level Analysis Tool for Early Exploration) [18]. SLATE allows designers to assemble, configure and simulate multi-core systems with L1 and L2 caches and memory controllers interconnected by a coherent bus, and under the control of a global on-chip power manager. The components are implemented in SystemC using cycle-accurate transaction-level abstractions. To the best of the authors' knowledge, SLATE is the first tool that combines detailed performance and power models of the core, system components, and power management unit for CMP systems.

The power manager queries the performance and power of all components at regular time intervals and decides how to best control the available actuators of each component in order to comply with a given power management policy, e.g., a fixed power budget. The power manager currently implemented in SLATE uses dynamic voltage and frequency scaling (DVFS) as the power / performance actuator. Two PM algorithms have been implemented, namely: the MaxBIPS algorithm from [10] which uses a discrete set of possible voltage and frequency pairs, and a continuous algorithm based on non-linear program-

978-1-4244-1922-7/08/$25.00 ©2008 IEEE

ming which finds the best possible voltage and frequency pair within a continuous range of values. In addition, this paper applies both algorithms on a per-core basis (i.e., each core is allowed a unique voltage / frequency setting) and chip-wide (i.e., all cores operate at the same voltage / frequency setting). In this way we can quantify the performance advantage of per-core DVFS to be traded off against the extra implementation cost, compared with chip-wide DVFS.

This paper is organized in the following way. Section II presents details on the SystemC modeling, including models for the core, system and power management. Section III describes the power management algorithms, and Section IV presents the experimental results and discussion. Section V gives the conclusions.

II. System Modeling

SLATE provides a library of performance models which can be easily connected together forming complex multi-core systems, including: processor models, cache models, memory model and a coherent bus model [22]. SystemC-based transaction-level models (TLM) are used, and all components communicate via a well-defined set of ports and channels, which makes it flexible and easy to assemble different system configurations, e.g., varying the number of cores. A wide range of parameters on the core, cache and memory models are supported. More details of the models are described in [18].

The models are based on the POWER family of processors and systems. All models, except for the Power Manager model, are cycle-accurate performance models. The core model is based on the POWER4 processor [19], running instruction traces as the input to the simulation. Through simulation over millions of instructions we obtain results for performance, such as cycles-per-instruction (CPI), and power for a given set of application traces.

A. Core Model

The core model is a pipeline-accurate performance model, based on the POWER4 processor which is a single-thread, out-of-order execution, in-order completion microarchitecture [19]. The units inside the core (e.g., Decode, Dispatch, Issue queues) are modeled functionally accurately, and the execution delay of each unit is modeled by a cycle-accurate pipeline channel, as illustrated in Figure 1. This approach implements a clear separation between computation and communication, with all the delay (computation delay + communication delay) being accurately modeled in the pipeline channel. The pipeline channel itself is a C++ template and can transport any data type. The delays are parameterized and multi-ports are used wherever applicable, e.g., out of the Dispatch unit. The number of certain units and the delays between them can be changed, allowing for extensive architectural exploration. For example, more fixed-point issue queues and execution units (FXQ, FXU) can be added just by instantiating them and connecting them to the generic pipeline channels.

SLATE's core model was tested and tuned using instruction traces generated from the SPEC CINT2000 benchmarks [20]. The CPI numbers were compared against a detailed production-quality performance simulator used in IBM [21]. On average for all 12 benchmarks SLATE results were within 16% of the production simulator results, which is acceptable for an early analysis system.

The core model also computes the dynamic and static power dissipated by the core as the simulation progresses. The power

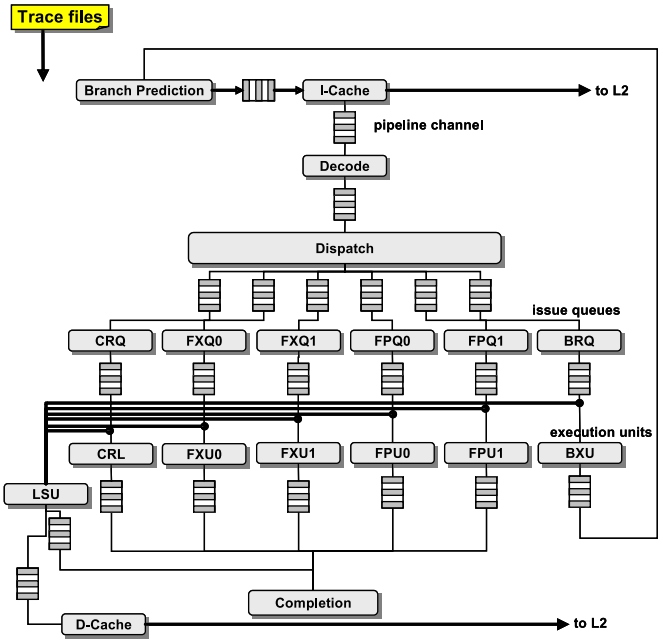


Fig. 1.: Internal organization of core model.

model used is based on [12]. Briefly, each unit inside the core has a formula for power that takes switching factors to produce the power for the unit. These formulas are generated by detailed logic and circuit-level simulations of the actual design or the previous version of the design, properly scaled for technology changes. The performance simulation provides the switching factors for each unit, which are then fed into the formulas as the simulation progresses, and the power is computed [12].

B. System Model

SLATE is targeted at building and simulating CMP systems, running multiprogrammed workloads, i.e., each core runs an independent application trace. Each core is connected to a dedicated level-2 cache, which is connected to a coherent bus. The bus is also connected to a memory controller model. The proprietary coherence protocol in the bus is modeled cycle-accurately. An accurate bus model is important in a multi-core system to simulate coherency and contention to memory precisely, as these factors may affect overall performance significantly. Parameterizable performance and power models for the caches (L1, L2), bus and memory controller were developed in SystemC. Power models include both dynamic and static power.

The components are interconnected using specialized SystemC communication channels capable of modeling both synchronous and asynchronous communication. The asynchronous channels enable communication between components operating at different frequencies which is a prerequisite for per-core DVFS. The channel delays are fully parameterizable and can implement a range of structures, including pipelines, queues, FIFOs and asynchronous handshakes, facilitating the exploration of different communication schemes.

The basic system example which will be used in the remainder of this paper is shown in Figure 2. It depicts four POWER4-like cores, four dedicated L2 caches, a bus, and a memory controller, all connected by communication channels, marked *CC*. The different shadings indicate that different internal implementations and different interfaces are used between each dif-

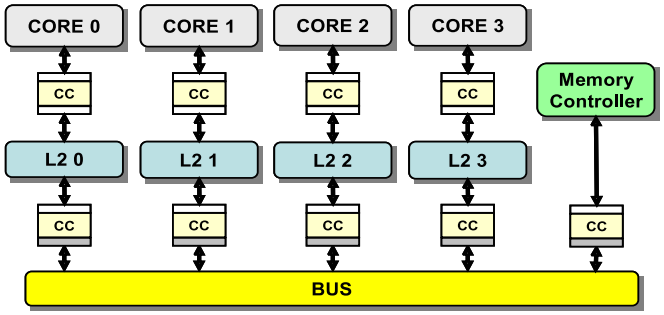


Fig. 2.: System example with 4 cores.

ferent pair of components. Connecting a component to another component is simply a matter of instantiating the communication channel that has the right pair of predefined interfaces to the two components.

C. Power Manager Model

The power management problem addressed here is the following: given a total chip power budget, and input application traces running on the cores, how to dynamically assign power modes (i.e., Vdd-frequency pairs) to each core, such that the overall chip performance (i.e., total number of instructions completed on all cores per time interval) is maximized. For the sake of simplicity, in this paper we only consider changing the Vdd and frequency of the cores while keeping the other components running at a nominal system Vdd and frequency. This is acceptable in this case because for the parameters and technology used in this paper the core power dominates the system power. However, the approach can be easily extended to handle all components in the system individually, or to group them into Vdd and clock domains.

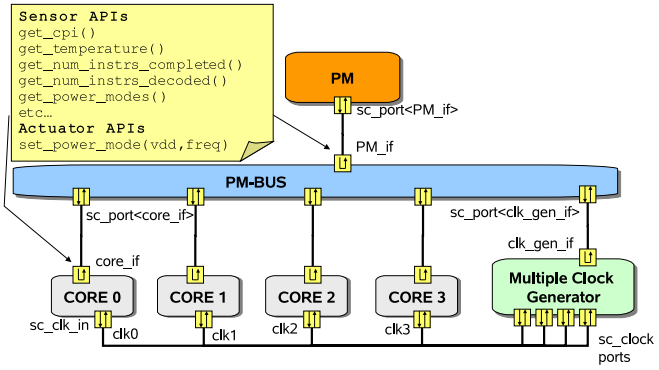


Fig. 3.: Power manager components and interfaces.

A power management infrastructure has been implemented in SLATE, as illustrated in Figure 3. The networks in Figure 2 and Figure 3 are both part of the same system model. The function of the PM module is to decide whether and how to apply DVFS to the cores in the design. It does this by periodically sampling sensor information from the cores, and running algorithms which decide on the next set of Vdds and frequencies to apply. The period, or observation window, is usually determined by a higher-level supervisor software layer and is normally in the order of a few hundred microseconds. Given the large periodicity of the PM loop, there was no need to implement the PM module in a cycle-accurate manner. We use a SystemC behavioral model triggered at every observation window.

In real hardware, the sensors are usually performance and power measurement proxies that each core keeps track of as it runs. This can be done either using digital counters or more complex analog circuitry. In the core and PM models, these sensors are implemented as function calls (APIs) over the SystemC interfaces connecting them, as shown in Figure 3. The PM component connects to a PM bus that decodes the command, sends the right Vdd and frequency to the designated core, and sends commands to the clock generator to modify the frequency of the clock that is connected to the designated core.

The multiple clock generator component is parameterized on the number of clocks. All clocks may change frequency simultaneously, if so requested by the PM module. The clock generator can also take into account a time penalty for changing the frequency if necessary, although current voltage and frequency-changing hardware techniques can perform a smooth transition between modes, and there is no need to stall the execution during voltage and frequency changes [8].

III. Power Management Algorithms

A. MaxBIPS

As a reference point, and to test our PM framework, we implemented an existing algorithm, MaxBIPS, proposed in [10] for DVFS. The algorithm assumes a set of discrete power modes (Vdd-frequency pairs) for each core which the PM can control individually. The goal is to maximize the overall chip performance, as measured by the total number of completed instructions by all cores per time period, under a given power budget.

The MaxBIPS algorithm relies on the fact that when a given core switches from power mode A (VddA, freqA) in observation window N to power mode B (VddB, freqB) in observation window N+1, the future performance and power is predictable using simple formulas, as shown below. This assumes that the workload characteristics do not change significantly from one window to the next.

Observation Window	N	$N+1$
Mode	(v, f)	(v', f')
Performance	I	$I * (f'/f)$
Dynamic Power	P	$P * (v'/v)^2 * (f'/f)$
Static Power	L	$L * (v'/v)^3$ (approx.)

Based on these formulas, the MaxBIPS algorithm computes the estimated power and performance for all possible tuples $[\text{core}_i, \text{mode}_j]$ and selects the mode that maximizes performance while not exceeding the power budget. The worst-case complexity of the algorithm is $O(M^N)$, with M the number of modes and N the number of cores. In practice many $[core, mode]$ tuples can be pruned out during the search as soon as the total power exceeds the current budget, and both N and M are limited for practical reasons; as a result the algorithm can afford a simple search on all tuples for the optimal solution.

MaxBIPS uses a discrete number of power modes because it limits complexity and it is implementable in hardware. However, the best number of power modes to use is still an open question that depends on the silicon technology used, on the implementation costs of voltage regulator modules (VRMs) and phase locked-loops (PLLs), and on whether per-core or chip-wide DVFS is used.

B. Continuous Power Modes

To understand the impact of the fixed number of power modes and to compare with the discrete MaxBIPS solution, we used non-linear programming to model the same DVFS problem with continuous power modes. Under this Continuous Power Modes (CPM) algorithm, cores can run at any frequency and voltage within predefined upper and lower bounds. To understand the formulation, a few definitions are needed:

N Number of cores in the design.

$Interval$ Exploration interval.

$v_{\min}, f_{\min}, v_{\max}, f_{\max}$ The minimum and maximum voltages and frequencies.

k The slope of the frequency change function. It is assumed that voltage and frequency bear a linear relationship given by $k = (f_{\max} - f_{\min})/(v_{\max} - v_{\min})$

$v_0[1:N], f_0[1:N], P_0[1:N], L_0[1:N], I_0[1:N]$ Vectors storing the voltage, frequency, average dynamic and static power dissipation, and performance (I for instructions executed) in the last interval. The i-th value of these vectors is the corresponding value for $core_i$.

$v[1:N], f[1:N]$ Vectors representing the voltages and frequencies respectively, for the next interval.

The following linear constraint equation describes the relationship between the voltage change and frequency change:

$$f[i] = f_{\min} + k * (v[i] - v_{\min}) \quad \forall i \in \{0 \dots N\}$$

The sum of the predicted power (dynamic + static) must not exceed the power budget, which translates into the following non-linear constraint:

$$\sum_{1 \le i \le N} P_0[i] * (v[i]/v_0[i])^2 * f[i]/f_0[i] + L_0 * (v[i]/v_0[i])^3 \le \text{PowerBudget}$$

The objective function is to maximize the overall chip performance as measured by the sum of the predicted performance for the next interval, expressed as:

$$\sum_{1 \le i \le N} I_0[i] * f[i]/f_0[i]$$

Since the constraints and the objective functions have analytic gradient and Hessian information, the continuous per-core DVFS problem can be efficiently and optimally solved using the interior point method. We used OPT++ [22], a freely available non-linear program solver, to optimize the power modes in every observation window. The CPM algorithm is too complex for a hardware implementation, but it is useful in a design exploration methodology since it produces optimal results which allow us to evaluate heuristic algorithms with practical implementations.

Both MaxBIPS and CPM can be applied to chip-wide DVFS as well by restricting the search space to solutions in which all cores apply the same power mode (same voltage and frequency).

IV. Experimental Results

All experimental results were obtained by running simulations of the model shown in Figure 4, with the power management infrastructure of Figure 3, and under the power management algorithms described in Section III. Realistic values of voltage and frequency were used. For each core, the voltage range was [1.1 V to 0.9 V]. The frequency range was [3.5 GHz to 2.7 GHz]. These ranges were evenly divided into 9 levels, or power modes, resulting in voltage and frequency steps of 0.025 V and 0.1 GHz, respectively. The voltage transition rate was 10 mV/µs, resulting in a transition time of 2.5 µs for a Vdd step of 0.025 V.

Varying the PM observation window, when the PM algorithm is run, between 100 µs and 500 µs had only little effect on the experiments. Therefore, we show results for a fixed 500 µs window, corresponding to approximately 1.5 million cycles.

Experiments were run for power budgets varying from the highest power, i.e., all cores at max Vdd and frequency (*all_high* mode) to the lowest power, i.e., all cores at min Vdd and frequency (*all_low* mode). For each power budget the simulation was run for around 60 million cycles, covering a wide range of workload phases. Performance was measured as the total number of completed instructions in all cores over a fixed time period. Chip performance is highest at the all_high mode and lowest at the all_low mode. All performance values are normalized with respect to the performance measured in the all_high mode. In order to produce realistic results, we employed four SPEC CINT2000 benchmark traces with different workload characteristics: *eon*, *gzip*, *perl*, and *twolf*.

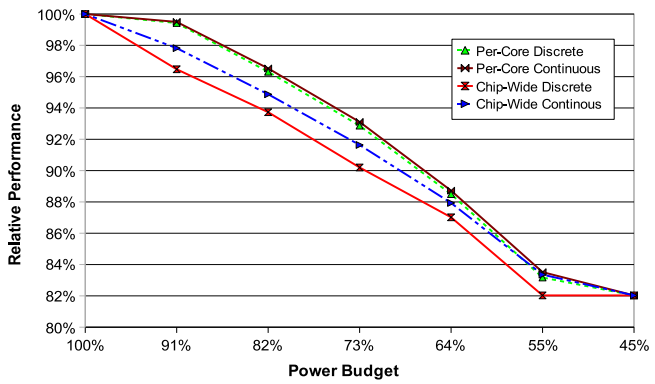
Figure 4 shows several graphs of power and performance for the whole range of power budgets (from all_high to all_low), comparing the discrete algorithm (MaxBIPS) with the continuous algorithm (CPM), both applied per-core as well as chip-wide. The labels with "-c" indicate results using the CPM; absence of "-c" means that MaxBIPS was used. Detailed explanations of each graph are given below.

Figure 4a—Overall Chip Performance: This chart contains four lines representing the overall chip performance under DVFS for seven different power budgets, ranging from 100% (all_high mode) to 45% (all_low mode). The four lines correspond to the discrete (using 9 power modes) and continuous algorithms applied on a per-core and chip-wide basis. As expected, the *chip-wide discrete* line shows the worst performance degradation as the power budget is reduced, followed by *chip-wide continuous* and, almost matching, *per-core discrete* and *per-core continuous*.

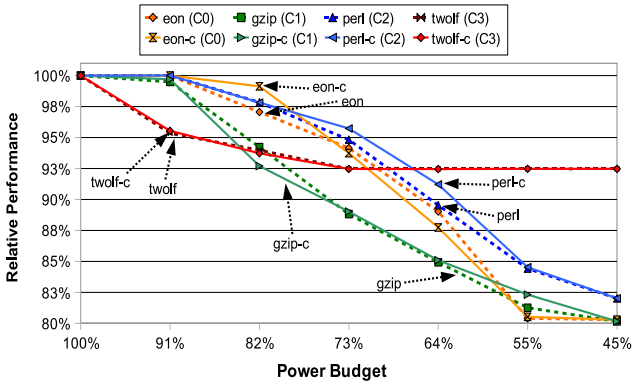
The *per-core discrete* approach has a distinct advantage over the *chip-wide discrete* of about 2.5 percentage points, using 9 levels for Vdd and frequency. We ran several experiments using only 3 levels (a more cost-effective solution, not shown here) and the difference grew to about 6 percentage points. This is a valuable insight because it can help designers evaluate the cost-benefit of comparable solutions (for example, 3-level per-core VRMs, versus 9-level single VRM for the whole chip).

The *per-core discrete* and *per-core continuous* lines almost match, which indicates that using 9 levels for Vdd and frequency is practically optimal and there is no need to use more levels.

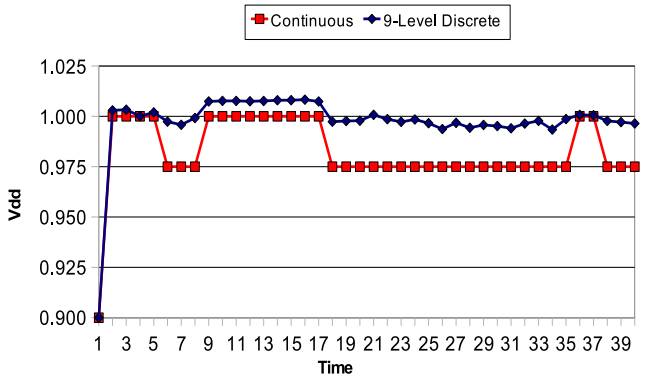
Figure 4b—Individual Core Relative Performance: This chart presents the results of applying DVFS on a per-core basis for different power budgets, using both discrete (9 levels) and continuous algorithms. Instead of showing the overall chip per-



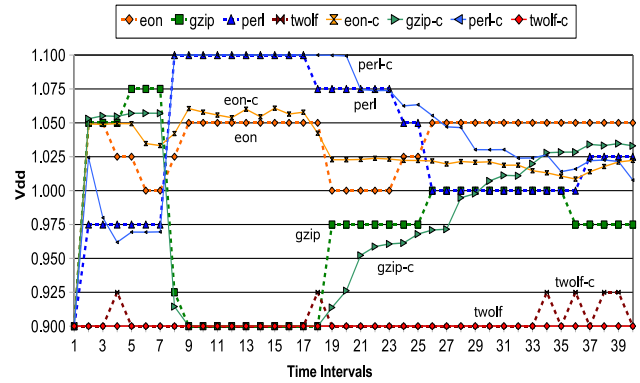
(a) Overall chip performance for decreasing power budgets.



(b) Individual core performance under per-core DVFS.



(c) Chip-wide DVFS transitions for 73% power budget.



(d) Per-core DVFS transitions for 73% power budget.

Fig. 4.: Performance and power analysis results for 4-core design under power management.

formance, this chart compares the performance of each core, running a specific trace. The chart illustrates how the algorithms choose modes for each core, thus affecting their relative performance in different ways, in order to maximize overall chip performance.

The performance of an application running on a core, measured in instructions per cycle (IPC), depends on how much time the application spends doing computations versus time spent waiting for memory accesses. The frequency of a core directly affects its computation speed but has little influence on the memory latency. Therefore, the performance of computationally intensive applications is more sensitive to voltage and frequency scaling than that of memory-bound applications.

As the power budget decreases, the algorithms decrease the voltages and frequencies of the cores with lowest IPC first—core 3 running twolf, followed by core 1 running gzip—and decrease less the voltages and frequencies of the cores with higher IPC. As the power budget continues to decrease, not enough power can be saved by only the lowest IPC cores and the algorithms further lower the voltages and frequencies of the higher IPC cores. No core drops below 80% of their maximum performance.

By comparing the discrete and continuous ("-c") lines for each core one can see how much performance can be gained by using a continuous solution. For twolf the difference is virtually zero while for the other cores the difference is 1 to 2 percentage points, indicating again that using 9 levels is almost as good as the optimal continuous solution.

Figures 4c/4d—Chip-wide and Per-core DVFS Transitions: These graphs show the evolution of both algorithms over time, and how they triggered each core to switch, or not, to a different voltage and frequency at every observation interval, for a fixed power budget of 73% of all_high. As the simulation progresses and the workload characteristics change, the algorithms may have to trigger power mode changes in order to not exceed the given power budget. Figure 4c shows the transitions for the case of chip-wide DVFS for both discrete and continuous algorithms. It can be noticed that the discrete algorithm causes very definite step-wise transitions, whereas the continuous algorithm applies more frequent and much more gradual changes to Vdd. A similar effect can be observed when per-core DVFS is applied, as shown in Figure 4d.

Figure 5 shows the average power overshoot and undershoot, i.e., the average deviation from the power budget, over the entire simulation for different power budgets. To keep track of power at a finer granularity rate, we measured the chip power every 10,000 cycles and computed the averages of all overshoot and undershoot values. This figure plots these averages for all four cases studied (per-core and chip-wide, using both discrete and continuous algorithms). From this figure it can be seen that the average overshoots and undershoots are fairly small, and per-core DVFS overshoots and undershoots are smaller than in chip-wide DVFS. This represents how well the algorithm is able to adapt to workload changes and stay within the power budget.

Execution times for these experiments were around 120 min for 60 million instructions. The actual power management algorithms were very fast and not significantly affecting the overall simulation time.

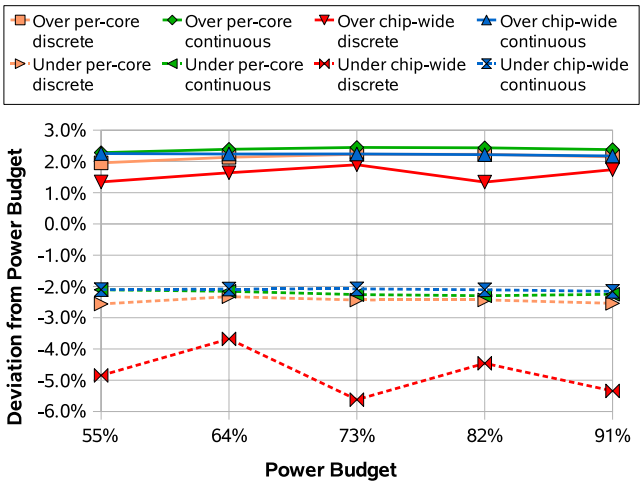


Fig. 5.: Average power overshoots, undershoots.

V. Conclusions

This paper presented a performance and power analysis methodology based on a simulation model for multi-core microprocessor systems with an integrated power management module applying dynamic voltage and frequency scaling. The novelty in this paper is three-fold. First, it presented a flexible, componentized environment capable of simulating the performance and power of systems, with detailed models of cores, caches, buses and memory controllers, and a variety of interconnection paradigms. Secondly, it described an integrated power management infrastructure which, coupled with the performance and power models, allows designers to explore different power management policies and algorithms. Thirdly, it presented a formulation and solution to the DVFS problem using continuous voltage and frequency ranges, using non-linear optimization methods. This algorithm, although not practical for hardware implementation, is very useful for analyzing the quality of other algorithms.

Extensive results were given analyzing different facets of the multi-core power management problem, including comparisons of per-core and chip-wide approaches and discrete and continuous algorithms.

As a next step we plan to investigate the effect of systems software such as task scheduling on the integrated power manager and interactions between them.

References

[1] L.A. Barroso, "The price of performance". ACM Queue, vol. 3, no. 7, pp. 48–53, September 2005.

[2] G.A. Paleologo, L. Benini, A. Bogliolo, and G. De Micheli, "Policy optimization for dynamic power management". In *Proceedings of the Design Automation Conference (DAC'98)*, IEEE/ACM, pp. 182–187, San Francisco, June 15–19, 1998.

[3] E. Grochowski, R. Ronen, J. Shen, and H. Wang, "Best of both latency and throughput". In *Proceedings of IEEE International Conference on Computer Design (ICCD 2004)*, pp. 236–243, San Jose, October 11–13, 2004.

[4] S. Heo, K. Barr and K. Asanovic, "Reducing power density through activity migration". In *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 217–222, Seoul, August 25–27, 2003.

[5] M. Powell, M. Gomaa, and T.N. Vijaykumar, "Heat-and-run: leveraging SMT and CMP to manage power density through the operating system". In *Proceedings of 11th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XI)*, pp. 260–270, Boston, October 7–13, 2004.

[6] E. Kursun, C.Y. Cher, A. Buyuktosunoglu, and P. Bose, "Investigating the effects of task scheduling on thermal behavior". *3rd Workshop on Temperature-Aware Computer Systems (TACS'06)*, Boston, June 18, 2006.

[7] K. Nowka, G. Carpenter, E. MacDonald, H.C. Ngo, B. Brock, K. Ishii, T. Nguyen, and J. Burns, "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, pp. 1600–1608, November 2002.

[8] G. Magklis, G. Semeraro, D.H. Albonesi, S.G. Dropsho, S. Dwarkadas, and M.L. Scott, "Dynamic frequency and voltage scaling for a multiple-clock-domain microprocessor". *IEEE Micro*, vol. 23, no. 6, pp. 62–68, Nov/Dec 2003.

[9] R. McGowen, C. Poirier, C. Bostak, J. Ignowski, M. Millican, W. Parks, and S. Naffziger, "Power and temperature control on a 90-nm Itanium family processor". IEEE Journal of Solid-State Circuits, vol. 41, no. 1, pp. 229–237, January 2006.

[10] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi, "An analysis of efficient multi-core global power management policies: maximizing performance for a given power budget". In *Proceedings of the 39th Annual International Symposium on Microarchitecure (MICRO'06)*, IEEE, pp. 347–358, Orlando, December 9–13, 2006.

[11] S. Manne, A. Klauser, and D. Grunwald, "Pipeline gating: speculation control for energy reduction". In *Proceedings of the 25th International Symposium on Computer Architecture (ISCA'98)*, pp. 132–141, Barcelona, June/July 1998.

[12] D. Brooks, P. Bose, V. Srinivasan, M.K. Gschwind, P. Emma, and M. Rosenfield, "New methodology for early-stage microarchitecture-level power-performance analysis of microprocessors". IBM Journal of Research & Development, vol. 47, no. 5/6, pp. 653–670, September/November 2003.

[13] S.J. Wilton, and N.P. Jouppi, "CACTI: an enhanced cache access and cycle time model". IEEE Journal of Solid-State Circuits, vol. 31, no. 5, pp. 677–688, May 1996.

[14] N. Magen, A. Kolodny, U. Weiser, and N. Shamir, "Interconnect-power dissipation in a microprocessor". In *Proceedings of the 2004 International Workshop on System-Level Interconnect Prediction (SLIP'04)*, pp. 7–13, Paris, February 14–15, 2004.

[15] T. Austin, E. Larson, D. Ernst, "SimpleScalar: an infrastructure for computer system modeling". IEEE Computer, vol. 35, no. 2, pp. 59–67, February 2002.

[16] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimization". In *Proceedings of the 27th annual international symposium on Computer architecture (ISCA 2000)*, pp. 83–94, Vancouver, June 10–14, 2000.

[17] A. Bogliolo, L. Benini, E. Lattanzi, and G. De Micheli, "Specification and analysis of power-managed systems". Proceedings of the IEEE, vol. 92, no. 8, pp. 1308–1346, August 2004.

[18] R. Bergamaschi, G. Han, A. Buyuktosunoglu, H. Patel, I. Nair, G. Janssen, G. Dittmann, N. Dhanwada, Z. Hu, P. Bose, J. Darringer, "Performance modeling for early analysis of multi-core systems." In *Proceedings of the 5th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS 2007)*, pp. 209–214, Salzburg, September 30–October 5, 2007.

[19] J. Tendler, J.S. Dodson, J.S. Fields Jr., H. Le, B. Sinharoy, "POWER4 system microarchitecture". IBM Journal of Research & Development, vol. 46, no. 1, pp. 5–26, January 2002.

[20] SPEC Standard Performance Evaluation Corporation, `www.spec.org`.

[21] S.R. Kunkel, R.J. Eickemeyer, M.H. Lipasti, T.J. Mullins, B. O'Krafka, H. Rosenberg, S.P. VanderWiel, P.L. Vitale, and L.D. Whitley, "A performance methodology for commercial servers". IBM Journal of Research & Development, Vol.44, No.6, November, 2000.

[22] "OPT++: an object-oriented nonlinear optimization library", `http://csmr.ca.sandia.gov/opt++/`