Guarded Power Gating in a Multi-Core Setting

Niti Madan, Alper Buyuktosunoglu, Pradip Bose, Murali Annavaram

Guarded Power Gating in a Multi-Core Setting

Niti Madan, Alper Buyuktosunoglu, Pradip Bose
IBM T.J. Watson Research Center

Murali Annavaram
University of Southern California

Abstract

Power gating is an increasingly important actuation knob in chip-level dynamic power management. In a multi-core setting, a key design issue in this context, is determining the right balance of gating at the unit-level (within a core) and at the core-level. Another issue is how to architect the predictive control associated with such gating, in order to ensure maximal power savings at minimal performance loss. We use an abstract, analytical modeling framework to understand and discuss the fundamental tradeoffs in such a design. We consider plausible ranges of software/hardware control latencies and workload characteristics to understand when and where it makes sense to disable one or both of the gating mechanisms (i.e. intra- and inter-core). The overall goal of this research is to devise predictive power gating algorithms in a multi-core setting, with built-in "guard" mechanisms to prevent negative outcomes: e.g. a net increase in power consumption or an unacceptable level of performance loss.

1. Introduction

Power gating is a circuit-level technique that enables one to cut off the power supply to a logic macro. Power gating is implemented with the help of a sleep transistor ("switch") that is inserted as a series header or footer device in the V_{dd}-to-Ground circuit path that includes the targeted macro. With the help of microarchitectural predictive control, such gating is effected, when it is deemed that the macro is *likely* to be idle for a relatively long duration. While dynamic voltage and frequency scaling (DVFS) continues to be one of the most successfully deployed power management techniques, the dynamic range of (voltage-frequency) operational points is getting smaller, with the scaling downward of the supply voltage (V_{dd}). As such, predictive power-gating is emerging as an increasingly important actuation knob in chip-level dynamic power management. Functionally, this can be thought of as a special case of DVFS, with just two operating points: nominal or maximum voltage-frequency point and "OFF" state (zero voltage and zero frequency).

Recently, Intel's Nehalem processor family [1] has made per-core power gating available as a power management facility within a multi-core processor chip setting. Research to quantify the benefits of unit-level (intra-core) [2, 4, 5] and per-core (inter-core) [3] power gating algorithms have been pursued in the past. However, the trade-offs between unit-level (intra-core) gating and per-core gating in a multi-core chip setting have not been studied so far. In this paper, we present an analytical modeling approach to understand and discuss the fundamental tradeoffs that would be of interest to an early-stage definition team. Our simple queueing theory based model is a good starting point for such early evaluation as it can represent a real-world system with sufficient accuracy.

Predictive power-gating is a promising control knob for power management. However, mispredictions, if frequent, can lead to significant negative impact on power-performance, since there are overheads for switching on and off a gated macro. Lungu [4] et al. make a case for and propose built-in guard mechanisms for intra-core power gating algorithms to guarantee power-performance bounds. We demonstrate in this paper that inter-core power-gating algorithms also suffer from the same problem and will require guard mechanism to prevent power over-runs.

The paper is organized as follows. We first give the problem background in Section 2 and then describe the power-gating algorithms in Section 3 . We discuss the methodology and results in Sections 4 and 5, respectively, and finally our conclusions in Section 6.

2. Problem Background

Depending on the size of the macro that is targeted for power gating, the overhead in terms of "wake-up" latency or the power cost for turning it back on from a "gated off" state may be quite significant. In prior work, Hu [2] et al. have discussed the overhead costs of the power gating process in some detail, and have described ways of quantifying the so-called "breakeven point" (BEP) in terms of circuit and technology parameters. The BEP stands for the minimum number of consecutive processor execution cycles that the gated macro needs to remain in idle state (before being woken up back to active state), in order to ensure a net positive power savings. With predictive gating algorithms, the risk is that if mispredictions about idle period durations are frequent, the net power savings benefit may turn out to be negative. As discussed by Lungu [4] et al., such scenarios may be rare in typical workloads, but are quite common in loop-intensive, periodic application phases, in which the idle duration of a targeted unit (e.g. the floating point execution pipe) is periodic and short enough to cause a repeated under-cutting of the BEP threshold. As such, Lungu [4] et al. proposes the use of "guard" mechanisms to detect the onset of such "negative benefit" scenarios, so that the main power-gating sense-and-

control algorithm can itself be disabled in time to avoid a net increase in system power. This work was limited to unit-level power gating within a single core.

In this paper, we explore the use of predictive power gating mechanisms in the context of a multi-core processor chip, with N cores. We assume the availability of unit-level (intra-core) power-gating facility as well as core-level power-gating, termed as small and big knobs respectively in this paper. The small knobs allow the power manager to turn off unused units within a core, based on predictions of idle durations of macros within a core; and, the big knobs allow it to turn off entire cores, based on overall system utilization that is monitored. Fundamentally, we expect the big knob to be used when the system utilization is below a certain threshold level: a scenario in which it would make sense to enable "core folding": i.e. turn off a certain number of cores, and force a workload consolidation across a smaller number of cores. In such a core folding procedure (which may be applied recursively, until the desired power-performance balance is achieved), the individual core utilizations would increase, resulting in decreased power savings opportunity via the small knob.

There are actually two dimensions of utilization that come into play in the context of considering the benefits of power gating. From an operating system (task scheduler) viewpoint, a core is fully engaged (or utilized) during the time it is assigned an execution time slice. So, a 60% system utilization (SU) would mean that on average 40% of the cores were without assigned tasks. However, as a core executes a task during an assigned time slice, the individual units (and the full core itself) is typically not fully utilized, because of the particular instruction frequency mix and pipeline stalls, etc. So, the system may be 60% core-utilized from an OS-viewpoint, but within that 60% the unit-level, physical resource utilization (RU) may be only 10% (for example) for a particular unit. The first dimension (SU) would influence the degree of benefit derived from per-core power gating. The second dimension (RU) would influence the degree of benefit derived from intra-core power gating. The per-core power gating involves larger latency and energy overheads. So, intuitively, if SU is quite large, the big control knob should probably not be used, while the small knob may be useful to exploit if RU levels are small. If SU is quite small, recursive use of the big control knob is probably very useful; but as RU levels rise as a result, the additional value of the small control knob may become questionable (especially without a suitable guard mechanism).

Using a simple, analytical modeling framework, we conduct experiments to understand: (a) the workload and design point parameter spaces in which one or both knobs should be engaged for maximal benefit; (b) how to define and tune the parameters of a particular power gating algorithm that would result in convergence to the right level of core folding (using the big knob) supplemented perhaps by the "right" level of unit-level power gating within each core. The broader objective is to understand if the small knob is of limited value, once the big knob is available (or vice versa), or if there are realistic, common-case scenarios in which both knobs are essential.

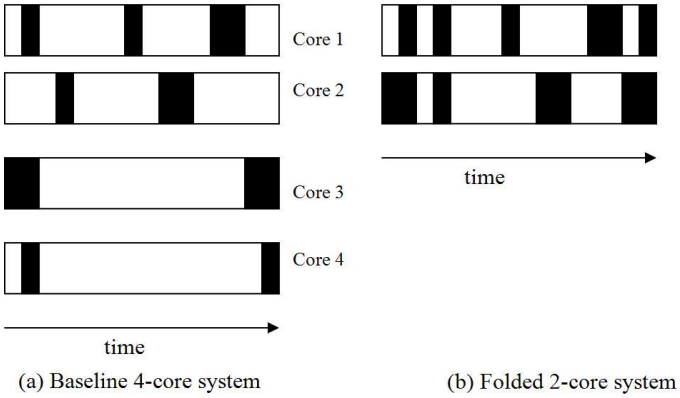

Figure 1. Example Power Gating Scenario

3. Power-gating Scenarios

In this section we describe the opportunities for power gating in a multi-core setting, by introducing the two dimensions of core utilization that are relevant in that context.

For illustration, consider a 4-core baseline system. The black time segments indicate a core's busy periods, which in this case exhibits periods of single time slice activity or back-to-back (double) time slice engagements. The overall time period shown is roughly that of 16 timeslices (or about 16 ms, if each OS time slice is 1 ms). Across the 4 cores, the total number of potential time slices to be utilized is then $16 \times 4 = 64$. This is clearly a lightly loaded system with the system-level utilization (from an OS perspective) of:

$SU = 13/64$ or 0.203

If we had applied core-folding using the big knob during this time period, we would in the first step of that folding have only 2 cores, and the system utilization would double to $SU = 13/32$ or 0.406 for the 2-core system. SU would remain 20.3% over the full 4-core system, of course. Note that, in this particular illustrative example, there is no performance change as we move over to the 2-core system.

Now, let's assume the average resource utilization (for a particular unit, or averaged over all units) within the core, RU is only 20% when it is assigned a task and 0% when it is not assigned a task. Then for the 4-core system, the net (average) RU's over the full 16ms time period are:

$$RU(1) = 0.8/16 = 0.05$$

$$RU(2) = 0.6/16 = 0.0375$$

$$RU(3) = 0.8/16 = 0.05$$

$$RU(4) = 0.4/16 = 0.025$$

For the 2-core system, the RU's are:

$$RU(1) = 1.2/16 = 0.075$$

$$RU(2) = 1.4/16 = 0.0875$$

In this initial study, we use simple heuristics to understand the tradeoffs of a system that provides both the actuation knobs. These algorithms can be replaced by more sophisticated power-gating or even other power-saving mechanisms such as DVFS. Both big knob and small knob heuristics use a predictive algorithm that is based on the prior-published concept [2], in which the idle state of a resource is monitored

for up to a pre-determined number of cycles.If the idle state persists for more than this threshold, C_T, then the resource is predicted to be experiencing a long duration idle state sequence. Therefore, the predictive control initiates the process of gating off that resource. The resource is woken back up at the onset of new work. For our big-knob or inter-core model, we power-off the cores. In the case of small-knob, we gate off the targeted execution units. We later evaluate the impact of overheads associated with wake-up delay and the idleness threshold on performance and power savings.

4. Modeling Methodology

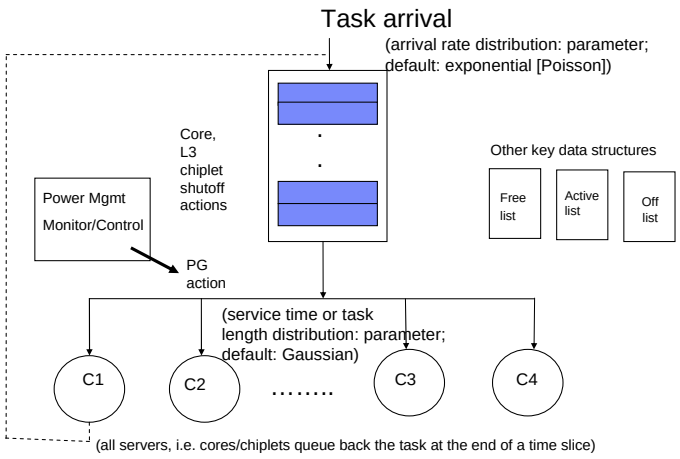

Figure 2. Overview of QUTE Framework

In this section, we briefly describe the modeling approach that we adopt for understanding the fundamental tradeoffs of the multi-core power gating problem, as formulated in the previous sections. Figure 2 depicts the high-level overview of a simple queuing model, called Qute (Queuing based timing estimator) that has been implemented in C/C++ for the purposes of this study.

The model uses a centralized task arrival queue. The task arrival process can be modeled using any arrival distribution of choice: e.g. either the well-known Poisson process (with exponentially distributed inter-arrival time distances) or even the one derived empirically from a real, measured task arrival process at a server node. For the purposes of this paper, we assume a Poisson arrival process. Tasks are issued from the head of the central queue to waiting cores in a round robin fashion. Each core services an assigned task for a pre-determined time slice (which is a model parameter).If the task does not complete within that time slice then the core simply queues the task back to the tail of the centralized queue. A given task is removed from the queue once it is completed. Each task may require several time slices of processing depending upon its length. Task lengths (i.e. time durations) are picked from a user-specified probability distribution: e.g. Gaussian. There is monitoring code in our model to keep track of the average number of utilized cores, the onset and duration of idle periods in each core for simulating inter-core or big-knob power gating heuristic.

To simulate the effect of applying the small control knob, we assume that each core can operate in one of n different "power" states. There is one nominal state in which there is no unit-level power-gating applied and the core operates at the nominal or rated power level, P. The other n-1 power states represent different low power modes of operation, depending on the degree of engagement of the small knobs across all units within the core. The duration of a particular power state is simulated statistically, while assuming that the initial state at the start of a time slice is always "nominal". In the analysis reported in this work, each power state duration is assumed to obey a Gaussian distribution, the mean being a model parameter. Furthermore, in the initial data presented, we assume that the number of power states, n is just 2. The first is the nominal, full power mode (*Hi*) and the second state represents a single low power mode (*Lo*) that the core is driven to (with a fixed number of execution resources that are gated off) during the onset of an idle phase. When a core is in the Lo mode, we assume it to be consuming a fraction of the nominal power, P. This fraction, F is itself another model parameter, that we conservatively set to 0.75 (or 75%) in the analysis that we present. For example, as in Lungu [4] et al., if the per-core power gating is only limited to the back-end execution units, $F = 0.75$ would be a reasonable choice.

We use "average response time" derived from our analytical model as the metric for evaluating performance and the "average number of cores switched on" for computing power savings for our big-knob heuristic. To model the power savings of our small knob heuristic, we measure the average percentage of time a core spends in each of the power states. For example, if we have a 32-core system and half the cores are switched off and on average a core spends 50% of its execution time in the low power mode (assuming 0.75 power ratio), then we compute the system power to be 14×P where P is per core nominal power. The equation below gives the power consumption of a hybrid system when both the knobs are enabled:

$$Total\_System\_Power = Num\_Cores\_switched\_on \times (\%time\_Hi + F \times \%time\_Lo) * P \quad (1)$$

If only the small-knob is enabled, we compute the total system power using the following equation:

$$Total\_System\_Power = Num\_Active\_Cores \times (\%time\_Hi + F \times \%time\_Lo) * P + (N - Num\_Active\_Cores)F * P \quad (2)$$

Table 1 shows the simulation parameters used for our experiments. All experiments run 1 million tasks which is long enough to reach a steady-state environment in our queueing model. We chose some of the parameters such as N, arrival rate and task rate such that our given queueing model is moderately utilized. The analytical system utilization of our baseline model is 0.62 based on the following equation:

$$SystemUtilization(SU) = \lambda/(\mu * N) \quad (3)$$

where λ is the mean arrival rate and μ is the mean service time. We evaluate our power gating techniques for different utilization levels in the system by varying the arrival rate.

Number of Cores (N)	32
Mean Task Length	5 ms (Gaussian)
Mean Inter-Arrival Rate	0.25 ms (Exponential)
Time Slice	1 ms
Core switch on latency	0.5 ms
OnLat	
Idleness Threshold C_T	0.5 ms
Number of Power Modes	2
Hi mode mean	300μs (Gaussian)
Lo mode mean	100μs (Gaussian)
F	0.75
Intra-core transition overhead	1μs

Table 1. Experiment Parameters

	Experiment	Response time (μs)	Number of cores on
	Base	5002.22	32
OnLat = 0.5ms	C_T = 0.5 ms	5038.46	24.99
	C_T = 0.3 ms	5070.12	23.33
	C_T = 0.1 ms	5158.51	21.83
	C_T = 10μs	5244.43	21.68
OnLat = 10μs	C_T = 0.5 ms	5002.93	24.82
	C_T = 10 μs	5007.07	20.77

Table 2. C_T Idleness Threshold Sensitivity Analysis

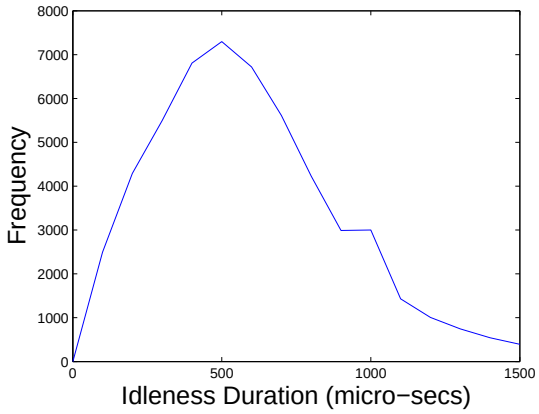

Figure 3. Idle Duration Histogram

OnLat	Response time (μs)	Number of cores switched on
10μs	5002.93	24.82
100μs	5009.36	24.86
500μs	5038.46	24.99
1ms	5075.69	25.15
5ms	5377.49	26.19

Table 3. OnLat Sensivity Analysis

5. Results

We first discuss the results when either of the two power gating knobs are enabled and then we evaluate a hybrid system where both the knobs are enabled.

5.1 Inter-core Power Gating Results

Table 2 shows the performance and power savings impact of enabling the big knob. The baseline case is when all cores are active (on) and none of the knobs are enabled. In this case, the average response time to get a task finished is 5.002 ms. The second set of experiments show the sensitivity of our heuristic to C_T when OnLat is 0.5ms. We find that the power savings is sensitive to our choice of C_T. With a lower idleness threshold, we get more opportunity to power gate cores. However, the average response time can degrade up to 4.8% if the OnLat is high as we switch cores more frequently. The third set of experiments show that for a lower OnLat latency (10μs), we can get better power savings as much as 34% with a negligible performance hit of 0.01%. Figure 3 shows idle duration histogram for our baseline model. As can be seen, the idle duration peaks around 0.5ms. However, there are still significant number of idle periods less than 0.5ms. This shows that we can reduce the C_T value further. However, we choose the conservative value of $C_T = 0.5ms$ as the baseline for all experiments.

We next perform a sensitivity analysis to study the impact of wake-up delay or core switch on latency in our model. As can be seen from Table 3, the power savings are independent of the wake-up delay for the given C_T but performance degrades by up to 7.4% as wake up delay increases to 5ms. These results help us understand the tradeoffs between implementation choices of the big-knob. For example, if we devise a software-based approach to power gating, then we will see a very high wake-up delay and hence more performance degradation. However, a hardware based technique will be faster to react to waking up cores.

5.2 Intra-core Power Gating Results

Table 4 shows the power and performance results for intra-core power gating. We consider a plausible range of workload phases where each core can be in either of the two power states. Note that we assume that a workload has higher or lower ILP depending upon how long it is in *Hi* or *Lo* power states as these states indicate resource utilization (RU) levels. We use equation (1) to compute the power savings assuming all the cores are switched on. We also assume that the *Lo* power mode has 0.75$\times$ nominal core power and the overhead of each mode transition is 1μs. Amongst all the workloads, the workload with short phases has the worst performance impact of 1% (although still negligible) due to more frequent transitions between the power modes. If a workload has long *Hi* phases, the power savings are minimal being only 3% . Similarly, long *Lo* phases yield maximum power savings of up to 20%.

We also study the sensitivity of the intra-core power gating transition overhead on performance. Table 5 shows the sensitivity analysis for a range of delay overheads for our base workload case where mean *Hi* = 300μs and mean *Lo* = 100μs. As shown in the results, even if we assume a very high penalty, the performance impact is still tolerable.

5.3 Hybrid Power Gating

We discuss the tradeoffs of invoking different power gating knobs and evaluate it as a function of inter-arrival rate or utilization. Table 6 shows the measured SU for a given inter-arrival rate. Figures 4(a) and (b) show normalized power consumption when compared to baseline with no power gating

Workload Behavior	Hi Mean (μs)	Lo Mean (μs)	Hi %	Lo %	Response time (μs)	Avg. Number of Cores on
Short phases	100	100	52%	48%	5050.51	28.16
	200	200	57%	43%	5027.36	28.48
High ILP	300	100	79%	21%	5026.46	30.08
Low ILP	100	300	30%	70%	5028.23	26.24
Very High ILP	500	100	89%	11%	5013.67	31.04
Very Low ILP	100	500	21%	79%	5019.95	25.6

Table 4. Power and Performance Analysis of Intra-core Power Gating

Overhead	Response time	Performance Impact
Base (No Intra-core)	5002.22	
0.5 μs	5008.3	0.11 %
1 μs	5014.68	0.23%
5 μs	5056.67	1.07%
10 μs	5107.08	2.01%

Table 5. Sensitivity Analysis for Overhead of Intra-core Power Gating

Inter-Arrival Rate (μs)	System Utilization Measured SU
50	1.0
100	1.0
300	0.52
500	0.31
1000	0.16
2000	0.08

Table 6. System Utilization

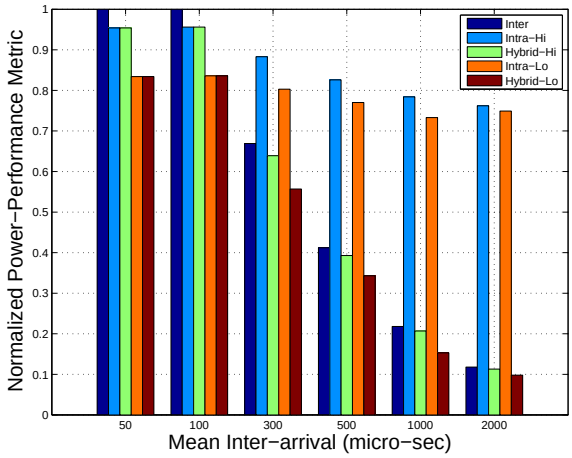

Figure 5. Normalized Performance-power Metric for High ILP and Low-ILP Workloads

for a system that supports only the small knob (*intra*), the big knob (*inter*) and a *hybrid* scheme with both the knobs enabled for different workload scenarios. Figure 4(a) shows the impact of running a high-ILP workload. Since a high-ILP workload doesn't have too much opportunity for doing unit-level power gating, we find that beyond 1.0 utilization or 100μs arrival rate, intra-core power gating helps only marginally. In the case of low-ILP workload as shown in Figure 4(b), intra-core power gating can help even if the system is lightly loaded. The *hybrid* system continues to perform better than (*inter*) yielding at least 2% more power savings for even the lowest utilization level.

We next explore the power-performance tradeoffs for all these models. We use the metric $Response\_time^2 * Num\_cores\_on$ similar to ED metric for our evaluation purposes. Figure 5 shows the results for both high ILP workload represented by *Hi* and low ILP workload represented by *Lo*. Since the performance penalty for power gating technique is not very high, these results are very similar to expected power savings discussed earlier. For a high ILP workload, the difference between *inter* and *hybrid* begins to diminish at arrival rate 500μs whereas for the low ILP workload, this trade-off point is at 2000μs.

5.4 Discussion

Based on the experimental results and equations (1) and (2), we can deduce that if the number of active cores is very close to N due to high SU or if our power factor F is very small, then the big knob does not help much. While big knob is useful to reduce core count through core folding, the opportunities for core folding occur only when SU is low. If SU is high then one has to rely exclusively on the small knob to reduce power consumption. The model parameter F determines the limit of power savings that can be achieved with the small knob. While SU is workload dependent, F is dependent on the hardware implementation, in addition to workload behavior. Due to implementation complexity of achieving very fine grain power gating at the intra-core level we surmise that F will not be less than 0.5. In any case, irrespective of the size of F, the small knob is always helpful due to its ability to take advantage of dynamic variation in resource utilization within a core.

5.5 A Case for Guard Mechanism

We found that when the workload behavior is predictable and follows a particular distribution mean, then our inter-core or big-knob power gating algorithm does not cross the power/performance bounds. However, if we want to implement an aggressive policy with lower C_T to get maximum power savings, then our core switching on/off frequency increases and we may end of expending more power in the process. We also observed that if the workload's arrival mean changed frequently then our power gating algorithm was taking incorrect decisions and switching cores frequently. For example, Table 7 illustrates a scenario where the mean arrival rate keeps toggling between 300μs and 1000μs. If each switching transition consumes say p% power then we end up losing 2.5X more power when compared to fixed arrival rate.

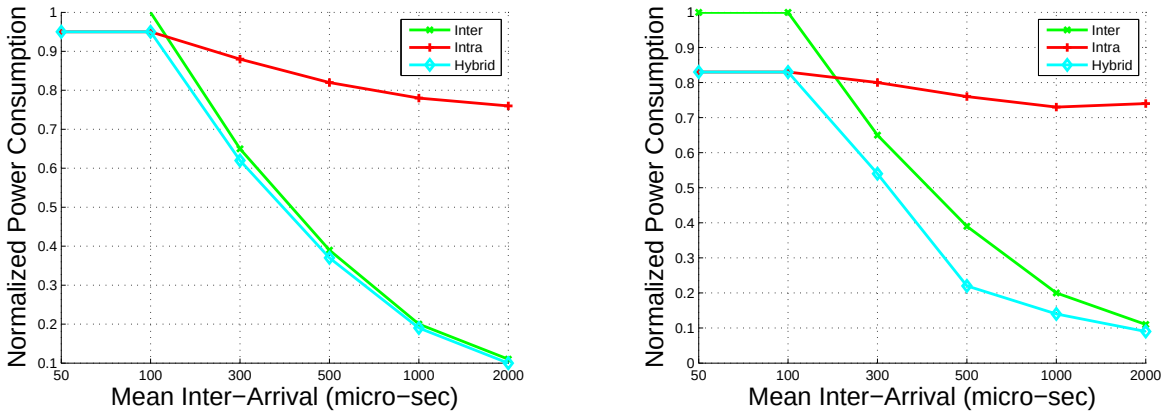

Figure 4. Normalized Utilization and Power Consumption as a function of Inter-arrival rate for high-ILP and low-ILP Workloads

While such worst case behaviors may be rare, when it comes to power consumption capping worst case behavior is critical for a variety of reasons, such as chip reliability. Hence, a guard mechanism must be designed in conjunction with the power gating approach.

Experiment	Response time	Switching Frequency
Fixed Arrival Rate	5043.88	91482
Toggling Arrival Rate	5111	226372

Table 7. A case for guard mechanism

6. Conclusions and Future Work

In this paper, we presented a simple analytical (queuing model) to develop basic understanding of the fundamental tradeoffs available during the definition of a multi-core power management architecture, in which the actuation mechanism is power-gating. We assumed the presence of both the small knob and the big knob as available facilities to a chip-level power manager. Our initial model-based results point to the following general conclusions:

- In a fully loaded system, where the task arrival queue is never empty, typically, the only knob of value is the small knob: i.e. by engaging the unit-level, intra-core gating, one can achieve the lowest possible system power level at close to the maximum performance level that the system is capable of.
- In a lightly loaded system, where the task arrival queue is often empty, the big knob is most effective, and the added benefit of having the small knob is minimal.
- In scenarios where the system load is intermediate, the optimal degree of core-folding (aided by the big knob) depends very much on the degree of fine-grain unit-level power gating that is available for use within each core. The value of the small knob may be minimal if the workload has higher ILP or resource utilization levels. However, if the workload has low ILP or low RU, then the small knob can yield moderate power savings along with the big knob.
- If the small knob can be implemented with lower power factor F, then it will always be helpful to have this feature in any of the above scenarios.

For future work, we plan to extend our framework to support real server utilization traces and study the power-performance tradeoffs for real workload scenarios. We will explore other power-management mechanisms such as DVFS as well as a system that can have multiple P-states. We will also architect guard mechanisms for a multi-core system.

Acknowledgements: The authors wish to thank Dr. Sriram Vajapeyam for developing the initial version of the Qute model that forms the basis of the analysis framework in this study. The work reported in this paper is supported in part by the National Science Foundation under Grant #0937060 to the Computing Research Association for the CI Fellows project that supports the first author.

References

[1] First the Tick, Now the Tock: Next Generation Intel Microarchite cture (Nehalem). Technical report, Intel Whitepaper, 2008.

[2] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose. Microarchitectural techniques for power gating of execution units. In *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED)*, August 2004.

[3] J. Leverich, M. Monchiero, V. Talwar, P. Ranganathan, and C. Kozyrakis. Power management of datacenter workloads using per-core power gating. *IEEE Computer Architecture Letters*, vol.8(2), July-December 2009.

[4] A. Lungu, P. Bose, A. Buyuktosunoglu, and D. Sorin. Dynamic power gating with quality guarantees. In *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED)*, August 2009.

[5] A. Youssef, M. Anis, and M. Elmasry. Dynamic standby leakage prediction for leakage-tolerant microprocessor functional units. In *Proceedings of International Symposium on Microarchitecture (MICRO)*, December 2006.