Power Management of Variation Aware Chip Multiprocessors

Abu Saad Papa
Center for VLSI and Embedded System Technologies
International Institute of Information Technology
Hyderabad - 500032, India
abu_saad@research.iiit.ac.in

Madhu Mutyam
Department of Computer Science and Engineering
Indian Institute of Technology Madras
Chennai - 600036, India
madhu@cse.iitm.ac.in

ABSTRACT

Faced with the challenge of finding ways to use an ever-growing transistor budget, microarchitects have begun to move towards the chip multiprocessors (CMPs) as an attractive solution. CMPs have become a common way of reducing chip complexity and power consumption while maintaining high performance. Multiple cores are replicated on a single chip, resulting in a potential linear scaling of performance. Cores are becoming sufficiently small with technology scaling. As technology continues to scale, inter-die and intra-die variations in process parameters can result in significant impact on performance and power consumption, leading to asymmetry among the cores that were designed to be symmetric. Adaptive voltage scaling can be used to bring all cores to the same performance level leaving only core-to-core power variations. The goal of our work is to find the optimal frequency that balances performance with power against asymmetry. We also demonstrate that traditional task scheduling techniques need to be revisited to mitigate the effects of process variations.

Categories and Subject Descriptors

C.1 [**Processor Architectures**]: Parallel Architectures; C.4 [**Performance of Systems**]: Performance attributes; D.4.1 [**Operating Systems**]: Process Management—*Scheduling, Threads*

General Terms

Performance

Keywords

Adaptive Voltage Scaling, Chip Multi-Processor, Process Variation, Power-Aware

1. INTRODUCTION

Technology scaling is increasing the gap between design and manufacturing expectations. As technology continues to scale beyond 65nm, inter-die and intra-die variations in process parameters (e.g., channel length and threshold voltage) can result in significant variations in the circuit characteristics. With the rapid advent of chip multiprocessors (CMPs), these manufacturing uncertainties may lead to significant variations in the operating frequencies and power consumptions of on-chip cores. As a result, it leads to asymmetry among the cores that were designed to be symmetric in performance causing core-to-core (C2C) variations [16]. This can cause reduced throughput, missed real-time deadlines, or excessive thermal throttling if more computationally intensive threads are mapped to higher-power cores or assigning too much work to slow cores and too little to fast cores.

Thus, to have a uniform frequency and equal performance across all cores, we can reduce the frequency of fast cores to the slowest running core (by techniques such as adaptive body biasing or V_{DD} adjustments) but this leads to performance loss or we can increase the frequency of slow cores to the fastest running core which results in increased *thermal throttling*. Thermal throttling is a mechanism that reduces power consumption when the core temperature exceeds a threshold temperature limit. Excessive thermal throttling affects the system performance.

In this work we find the optimal frequency which balances the performance with power against asymmetry and make an analysis to find the minimum power for a fixed performance deadline or maximum performance for a fixed power budget. The aim of this analysis is a system design perspective. If a system is aware of its power excesses due to inter-component process variation, it can make variation-aware decisions for allocating cores in a power-efficient manner. We also show the need for revisiting traditional scheduling techniques so as to mitigate the effects of process variation. We consider only C2C variations in this work.

The remaining paper is organized as follows. Section 2 gives the related work. Section 3 provides the analytical model used for simulating the 8-core system with process variations. Section 4 describes our frequency analysis model. Section 5 describes the experimental methodology and the performance metrics that we use to derive the results. Section 6 stresses on the need for revisiting traditional scheduling techniques and Section 7 concludes the paper.

2. RELATED WORK

Process variations have been considered as a major design challenge at the circuit and microarchitectural level [3]. Many circuit level techniques have been proposed to mitigate the effects of process variations [9, 31]. Variation tolerant architectural approaches include variation-tolerant register files [21], caches [1, 17, 26], issue queues [29], and pipeline organizations [12]. Chandra *et.al.* provided a methodology for modeling variations during system-level power analysis [7].

Very little work has considered the impact of process variations on CMPs. Humenay *et.al.* proposed a model for variations in multicore architectures [15]. In another work [16], Humenay *et.al.* showed that systematic within-die (WID) variations can cause large performance, power, and thermal variations among cores which were intended to be identical. In a recent work, Das *et.al.* [10] developed a variation-aware scheme for power optimization using single/multiple voltage islands across different cores in a CMP. Their work emphasized on the importance of multiple voltage islands in CMPs to reduce power dissipation and performed a detailed analysis of advantages for various voltage island formations.

Among the power management techniques, Isci *et.al.* [20] proposed and evaluated a dynamic power management perspective for CMP systems. Their work focused on the chip-level global monitoring, control, and dynamic management of power for the multi-core systems. Similar work is pursued by Sharkey *et.al.* [30], where design trade-offs associated with various CMP power management alternatives have been examined. They showed that chip-wide solutions are necessary to take the advantage of power optimization and concluded that on-chip hardware global power manager designs are necessary for effective power management. Note that both [20] and [30] have not taken process variations into consideration.

Donald *et.al.* [11] examined power and performance characteristics of multicore architectures, when running multiprogrammed workloads and parallel applications, in the context of process variation. Meng *et.al.* [24] presented a strategy for thread assignment that reduces the overall power consumption for chip multiprocessors fabricated under extreme parameter variations.

Our work is an extension of [16] and different from the other studies that have considered process variations in CMPs, in that

- We develop a frequency model that allows us to examine various performance-power configurations of a variation aware chip-multiprocessor in a full-system simulation environment.
- We make an analysis to find the optimal frequency for all cores which balances the performance with power against asymmetry.

3. MODEL

Systematic WID variations are the main source for C2C variations. In order to estimate the magnitude of frequency variations across the cores, we develop a model used by Humenay *et.al.* [16]. An assumption is made that chief source of systematic WID variation is variability in effective gate length (L_{eff}) due to optical variations across the exposure field. The optical component that Humenay *et.al.* [16] modeled is chiefly due to lens aberrations and is modeled as a simple polynomial function of position within the exposure field. The cross-chip systematic variation in L_{eff} (in *nm*), Δ_{sys}, for a die positioned in the lower-left hand quadrant of the reticle can be approximated by:

$$\Delta_{sys} = ax^2 + by^2 + cx + dy + exy + Intercept \qquad (1)$$

We use the same scaled coefficients as used by Humenay *et.al.* [16] in order to model variations at the 45nm technology node. The L_{eff} variations are then used to estimate the gate delay variations. The gate delay (D) and L_{eff} have a dependency as given in [27], *i.e.*,

$$D \sim L_{eff}^{1.5} . V_{dd} / (V_{dd} - V_{th})^{\alpha} \qquad (2)$$

where V_{dd} is supply voltage, V_{th} is threshold voltage, and α is velocity saturation. α approaches 1 as channel length becomes shorter. We assume $\alpha = 1.3$ for 45nm technology as given in [16].

Also L_{eff} is related to V_{th} by the equation shown below [6]

$$V_{th_{eff}} = V_{th_0} - V_{dd} . e^{(\alpha_{DIBL} - L_{eff})} \qquad (3)$$

where V_{th_0} is the threshold voltage for long channel transistors and α_{DIBL} is the DIBL coefficient. The default values for $V_{th_0} = 0.22$ and $\alpha_{DIBL} = 0.15$ were provided by the Predictive Technology Model [28].

Thus we can see that variation in L_{eff} leads to performance and power asymmetry among the cores. Hardware and software techniques can be employed to remove these asymmetries. One method is to design a simple software assuming symmetric core performance and apply hardware techniques such as adaptive voltage scaling (AVS) to reduce the frequency spread among the cores. This can increase power asymmetry and lead to either expensive cooling solutions or thermal throttling [16]. In another method, one can deal only with software to mitigate the effects of process variation. For example one can develop a process-variation aware task scheduling algorithm which takes care of both performance and power asymmetry among the cores, but the performance asymmetry can complicate processes such as scheduling, synchronization, and load balancing in addition to the thermal-management issues.

In this work, we opt for a method which employs both hardware and software techniques. We first remove the performance asymmetry by applying AVS technique. The power asymmetry is then taken care by software scheduling technique proposed in Section 6.

4. FREQUENCY MODELING

We would like all cores to have the same performance by making their operating frequencies equal. The two most common techniques to achieve symmetrical performance are adaptive body biasing (ABB) and AVS. Both the techniques do affect the leakage power of the cores. Also AVS has a cubic impact on dynamic power and ABB affects the dynamic power linearly.

In our work we use only AVS to compensate the core frequencies as AVS requires a much smaller change (percentage-wise) in supply voltage than ABB requires in threshold voltage. As a result, AVS has much milder impact on leakage and is more power-efficient and thermally compatible solution than ABB [16]. Also ABB requires the complicated deep N-well or triple well process for its implementation.

AVS can be implemented by providing a different supply voltage to each core (multiple voltage islands [10]) and measuring the core's maximum frequency during testing and then computing the necessary supply voltage scaling. The implementation of AVS is beyond the scope of our work and we assume that AVS has been implemented in our simulated multicore processor.

Using AVS we boost the frequency of slow running cores and reduce the frequency of high performance cores. The aim of this analysis is to find the optimal frequency which balances the performance loss against power asymmetry. For this we define a parameter β known as the *performance-power* *factor*. By assuming $f_1,\ f_2,\ f_3,\ \ldots f_N$ as the frequencies of N cores of a CMP, arranged in the ascending order without applying any frequency compensation techniques, we define $\beta = \frac{i}{2N}$ such that the frequency for all cores for a particular β value can be found by using the formula as shown below:

$$Freq, F = \begin{cases} \frac{f_1+f_2+\cdots+f_i}{i}, & i \le N \\ \frac{f_N+f_{N-1}+\cdots+f_{i-N}}{(2N-i)+1}, & i > N \end{cases} \qquad (4)$$

When β is '1', the performance is maximum and power saving is minimum, *i.e.*, all cores operate at the frequency of the highest speed core. When β is approximately '0', the power saving is maximum and performance is minimum, *i.e.*, all cores operate at the frequency of the slowest core.

As we show in the next section, the optimal value of frequency is obtained when β is around 0.5, *i.e.*, the optimal frequency is the mean of the original frequencies of the N cores. Thus by varying the value of β, we can get different performance-power configurations for our CMP. For an application with a specific power budget we can achieve the maximum performance by using higher values of β, in such a way that the power consumption is within the specified power budget. For an application with a specific performance deadline we can achieve maximum power savings by using lower values of β such that the execution time of the application is within the performance deadline. In an extreme case, if an application requires maximum performance, we can set the value of β to 1 so that all cores operate at the frequency of the highest performance core, and if we have a power sensitive application which does not require a performance deadline, we can set the value of β to $1/2N(\approx 0)$ and get the maximum power savings. Thus using different values of β, we can get the minimum power for a fixed performance deadline or maximum performance for a fixed power budget. The β value decides the common operating frequency of all the cores which is achieved by supplying different voltages to each core. Note that this β can correspond to different levels as considered in [30] for their chip-wide DVFS approach but in our work we supply different voltages to each core to achieve the same performance as we take process variations into consideration.

5. EXPERIMENT METHODOLOGY

5.1 Architectural Model

We use a modified version [14] of CMPFlex.OoO of the Flexus [32] family, which adds a detailed timing model to Virtutech Simics [22] to model performance and power of an 8-core Sparc based processor running Solaris OS. Flexus is a family of component based computer architecture simulators that enables full system timing-accurate simulation of uni and multiprocessor systems running unmodified commercial applications and operating systems. The CMPFlex.OoO simulator of the Flexus family simulates a Piranha [2] based chip-multiprocessor with out-of-order execution. Each core has a private L1 cache and a shared L2 cache. The enhanced simulator has a hybrid instruction-microarchitecture level Wattch-like [4] dynamic power model integrated with Flexus. The process and architecture parameters used in our experiments are given in Table 1.

5.2 Simulation

For validating our techniques, we use the following benchmarks of SPLASH-2 [33] application suite, namely, *cholesky, fft, lu, radix, raytrace, ocean,* and *volrend*. For *lu* and *ocean*,

Global Design Parameters	
Process Technology	45nm
Target Supply Voltage	1V
Clock Rate	3 GHz
Organization	8-core, shared L2 cache
Core Configuration	
Decode Width	4
Issue Width	4
Commit Width	4
Functional Units	3 FP ADD/SUB Units, 2 FP MUL/DIV Units, 6 Integer ALUs
Physical Registers	128 GPR, 128 FPR
Branch Predictor	16K-entry bimodal, gshare, selector
Memory Hierarchy	
L1 Dcache	64 KB, 2-way, 64 byte blocks, 1-cycle latency
L1 Icache	64 KB, 2-way, 64 byte blocks, 1-cycle latency
L2 cache	16 MB, 8-way Piranha, 64 byte blocks, 12-cycle latency
Main Memory	200-cycle latency

Table 1: Design parameters for modeled 8-core CMP

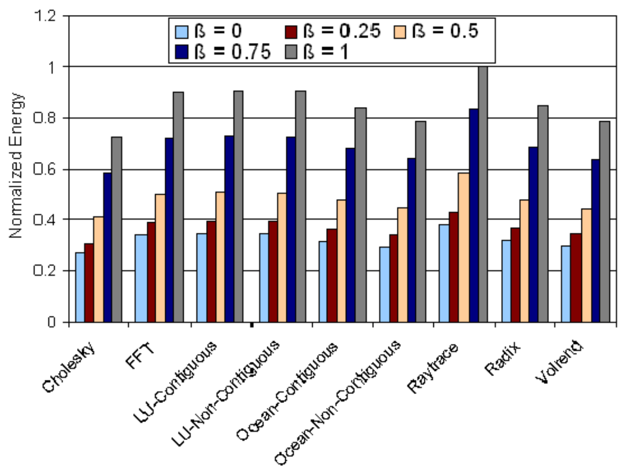
we use both the contiguous and non-contiguous blocks for our simulation. The desired clock rate of our processor is 3GHz [16, 25] but due to process variations each of the 8-cores has different operating frequency. We model our processor as an evenly distributed floor plan as in [16] with four rows and two columns. For our simulation we use the common frequency to all cores for five different values of β namely $0.125(\approx 0)$, 0.25, 0.5, 0.75, and 1.

5.3 Performance Metrics

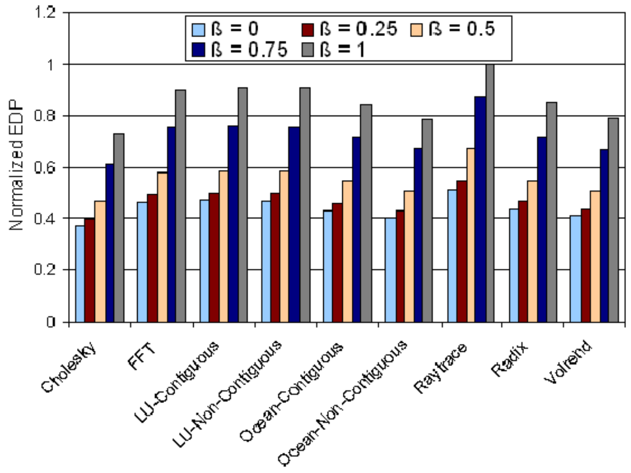
This section describes the metrics that we use to characterize performance and power-efficiency of various performance-power configurations.

We measure the performance of the system when running the given parallel applications by using the execution time or delay, which is computed as the time taken for the application to complete its execution. Power-efficiency is evaluated based on the total system power consumed by the application to complete its execution.

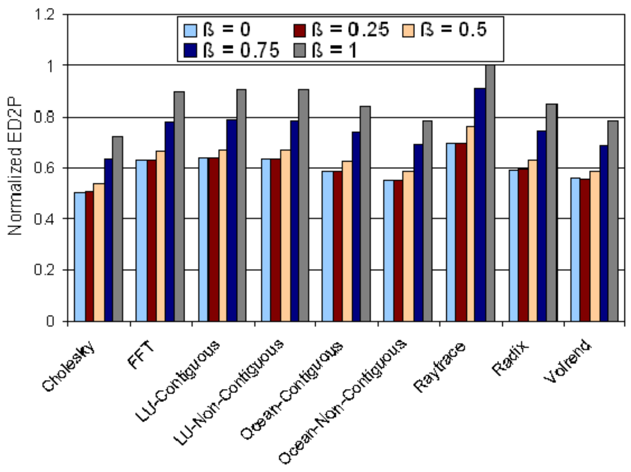
Common energy-time metric, the Energy-Delay^n Product (ED^{n}P), includes the Power-Delay Product (Energy) [8], when n=0, the Energy-Delay Product (EDP) [13], when n=1, the Energy-Delay^2 Product (ED^{2}P) [23], when n=2 and so on. Brooks *et.al.* [5] suggest using EDP for high-end workstations and ED^{2}P for high performance servers, *i.e.*, ED^{n}P metrics, when $n < 2$, are usually preferred when targeting low power systems, while ED^{n}P metrics, when $n > 2$, are used when focus is on high performance. In this work we use ED^{2}P and ED^{3}P to choose the optimal performance-power configuration (*i.e.*, the β value that has the minimum ED^{2}P or ED^{3}P). The ED^{2}P metric is independent of the supply voltage, but only to the first approximation [23]. We know that the power consumption is roughly related to V^3. The performance, however, varies better than linearly with the frequency because the effective memory latency is reduced as the frequency goes down. As voltage and frequency have a linear variation, we see the performance to scale better than V^3 (as V decreases) and power scales with V^3 in the ED^{2}P



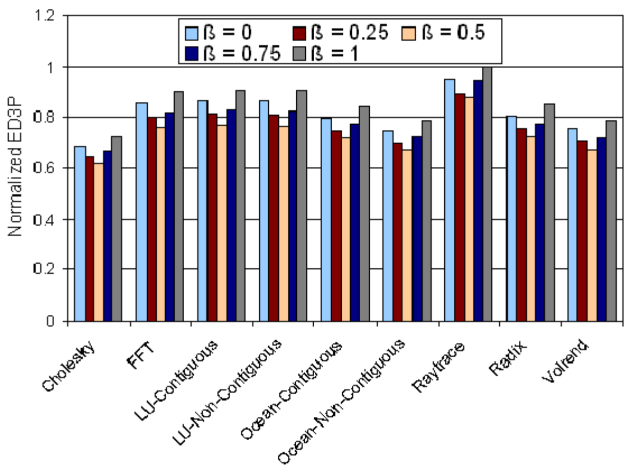
(a) **Energy ($n = 0$)**



(b) **Energy Delay Product ($n = 1$)**



(c) **Energy Delay^2 Product ($n = 2$)**



(d) **Energy Delay^3 Product ($n = 3$)**

Figure 1: ED^{n}P behavior for different values of β where $n = 0, 1, 2,$ and 3

metric. The net result is a decrease in ED^{2}P with lower β values although at the cost of significant performance degradation [30]. Thus we have also focussed on ED^{3}P to put more performance constraint. The other metrics suffer because they are too energy dominant ($n < 2$) resulting in preference given to low performance systems or too performance dominant ($n > 3$) resulting in preference given to high power consuming systems.

5.4 Results

The ED^{n}P values of the benchmarks for different values of β when $n = 0, 1, 2,$ and 3 are shown in Figure 1. The values are normalized to the respective maximum ED^{n}P value which is achieved for *raytrace* application when $\beta = 1$. Each of the benchmarks is run to its completion using 8 threads on our 8-core simulated system. Figure 1(a) and 1(b) show the Energy and EDP, respectively. As expected, maximum energy is used when β is 1 and also the performance is maximum. The Energy and EDP metrics are biased towards energy and hence show the optimal configuration when $\beta = 0$. We then focus on the ED^{2}P metric as shown in Figure 1(c). For the ED^{2}P metric, the optimal performance with acceptable energy consumption (minimum ED^{2}P value) is achieved for lower values of β as shown in Figure 1(c). The optimal configuration for ED^{2}P metric at low β values at the cost of significant performance degradation is due to the better scaling of performance than V^3 as discussed in the previous section. Thus to put more performance constraint we use the ED^{3}P metric as shown in Figure 1(d), where we get the optimal configuration when $\beta = 0.5$. It is seen that for higher values of n, *i.e.*, $n > 3$, the optimal configuration is when $\beta \geq 0.75$.

The power variation among the cores for different values of β for the *raytrace* application is shown in Figure 2. The power values of each core are normalized to the maximum core power value achieved when $\beta = 1$ for core 7. The normalized power value of '0' indicates the power of the cores for the ideal case when there is no process variations, *i.e.*, all cores have a supply voltage of 1V and frequency of 3GHz. The power deviation from the ideal case for various β values is shown in Table 2 and the minimum power deviation is found to be 8.82% for $\beta = 0.5$. The other applications have a similar variation pattern. We find the power asymmetry among the cores by statistically calculating the standard de-

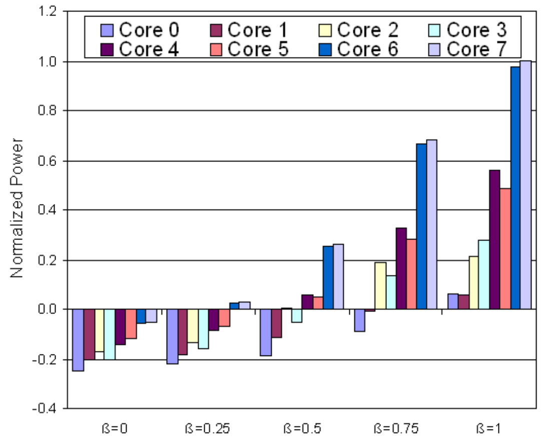


Figure 2: Power variation of each core for *raytrace* application

β	Deviation(%) from the Ideal Case	SD in Watts
0	39.55	0.90
0.25	26.67	1.16
0.5	8.82	1.92
0.75	71.89	3.33
1	118.64	4.40

Table 2: Power asymmetry for *raytrace* application

viation (SD) of the power values of the 8 cores. Table 2 also shows the SD for *raytrace* application for different β values. The minimum SD is 0.90W for $\beta = 0$ *i.e.*, the power asymmetry is minimum for $\beta = 0$. Though the power asymmetry is minimum for $\beta = 0$, the performance of the CMP is very poor (Figure 1(d)). The power asymmetry among the cores for $\beta = 0.25$ and $\beta = 0.5$ is not much as compared to the higher β values as shown in Table 2, but the power deviation from the ideal case is on the higher side (26.67%) for the configuration of $\beta = 0.25$. For the configuration of $\beta = 0.5$ the power deviation from the ideal case is minimum, and also ED^{3}P value is the least (Figure 1(d)), with only the power asymmetry slightly higher (1.92W) when compared to $\beta = 0.25$. We can thus conclude that the optimal frequency which balances performance with power against asymmetry is when $\beta = 0.5$.

6. OS SCHEDULING

The aim of this section is not to validate any new scheduling algorithm but to stress on the need for revisiting traditional task scheduling techniques so as to achieve maximum power savings. We use the same set of benchmarks as considered in Section 5, but this time we make each application to run with 4 threads on 8 cores. We then use the processor utility commands of Solaris to execute the benchmarks in three different modes. In the *normal* mode we allow the OS to execute the 4 threads in any 4 of the available 8 cores without any additional changes. In the *best* mode, using the command 'psradm' of Solaris, we make 4 threads to run on the low power consuming cores so as to achieve power savings. We also show the *worst* mode where 4 threads execute on the high power consuming cores. Energy behavior of the three different modes for a particular value of β, namely, when the performance-power factor(β) is 0.5, is shown in Figure 3. The energy values are normalized to the maximum energy value, achieved for the *raytrace* application in the worst mode. The energy savings from the worst case to the best case for the same β value is shown in Table 3, where we can see savings upto 30% for the Ray Tracer Application. Note that in all the three modes the performance of the cores is the same and hence the energy savings we obtain are without any performance degradation. If the OS assumes that all the cores are same then there are chances that it may schedule the threads in the worst manner by utilizing the high power consuming cores instead of the low power consuming cores.

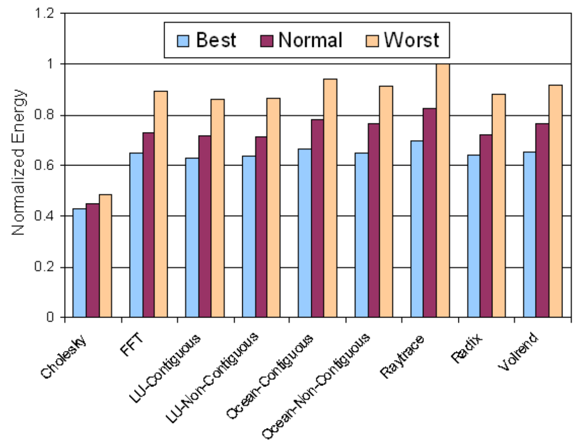


Figure 3: Energy behavior when $\beta = 0.5$ for the Best, Normal, and Worst mode

BenchMarks	Energy Savings in %
Cholesky	11.49
FFT	27.43
LU - Contiguous	26.76
LU - Non-Contiguous	26.60
Ocean-Contiguous	29.01
Ocean-Non-Contiguous	28.82
Ray Tracer	30.16
Radix	27.34
Volume Renderer	28.48

Table 3: Energy Savings when $\beta = 0.5$

Since our processor has equal performance on all the cores but unequal power consumption, the OS must schedule threads by having prior knowledge of the power consumption capabilities of each core. This information is already available to the OS scheduler during the testing phase of the processor, eliminating the time overhead in selecting the low power consuming core. The OS scheduler should be designed in such a way that it gives high priority to the low power consuming core and low priority to the high power consuming core. If there are more threads than the cores, the power savings can be achieved by scheduling dynamically the more computationally intensive threads (*i.e.*, high power consuming thread) to the low power consuming core (since all our cores have equal performance, there is no performance loss). Computationally intensive threads can be identified dynamically with the help of hardware performance counters which allow measuring thread specific runtime statistics. These counters are found in modern day processors [18, 19].

Thus the OS cannot assume all cores to be equal due to process variations and the techniques used to mitigate these variations. The traditional scheduling techniques need to be revisited and new techniques have to be evolved for efficient power management of chip multiprocessors. For the frequency model developed by us, the OS scheduling is made simple as the OS has prior information of the power consumption capability of each core. We leave the exploration and design of OS scheduler for future work.

7. CONCLUSIONS

Process variation effects on modern microprocessors are of utmost concern due to the reliability and thermal issues that arise if these variations are not taken into consideration. As the trend towards many-core CMPs has started, it is highly important to develop techniques to mitigate the impact of process variations.

To this end, we modeled a variation aware chip multiprocessor where the performance of all cores is same but there is power variation among the cores. This enabled the OS to schedule the threads in a power-efficient manner without any extra overhead (in time) in selecting the core for thread assignment as OS has prior information of power utilization of each core. We have shown that running 4 threads in our 8 core system with proper utilization of the cores gave us energy savings of upto 30% as compared to the worst utilization of cores in terms of the power consumption. Also we found the optimal frequency which balances the performance with power against asymmetry and made an analysis to find the minimum power for a fixed performance deadline or maximum performance for a fixed power budget.

We also demonstrated that traditional scheduling techniques need to be revisited and new scheduling techniques are needed to map threads to the proper cores. In future, software cannot be designed assuming symmetric core performance as due to process variations there will be asymmetry among the cores which should be taken into account.

8. ACKNOWLEDGEMENTS

This work is supported in part by grants from the Department of Science and Technology (DST), Govt. of India, Project No. SR/S3/EECE/80/2006, and the Indian Institute of Technology, Madras, under the New Faculty Scheme, Project No. CSE/07-08/227/NFSC/MADU.

9. REFERENCES

[1] A. Agarwal *et.al.*, "A Process-Tolerant Cache Architecture for Improved Yield in Nanoscale Technologies." *IEEE Transactions on VLSI Systems*, 13(1), pp. 27-38, 2005.

[2] L. A. Barroso *et.al.*, " Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing." *Proc. of the 27th Intl. Symp. on Computer Architecture*, pp. 282-293, 2000.

[3] S. Borkar *et.al.*, "Parameter Variations and Impact on Circuits and Microarchitectures." *Proc. of the Design Automation Conf.*, pp. 338-342, 2003.

[4] D. Brooks *et.al.*, "Wattch: A Framework for Architectural-level Power Analysis and Optimizations." *Proc. of the 27th Intl. Symp. on Computer Architecture*, pp. 83-94, 2000.

[5] D. Brooks *et.al.*, "Power-Aware Microarchitecture: Design and Modeling Challenges for Next-Generation Microprocessors." *IEEE Micro*, 20(6), pp. 26-44, 2000.

[6] Y. Cao and L. T. Clark, "Mapping Statistical Process Variations Toward Circuit Performance Variability: An Analytical Modeling Approach." *Proc. of 42nd Design Automation Conf.*, pp. 658-663, 2005.

[7] S. Chandra *et.al.*, "Considering Process Variations During System-Level Power Analysis." *Proc. of the Intl. Symp. on Low Power Electronics and Design*, pp. 342-345, 2006.

[8] A. P. Chandrakasan *et.al.*, "Low Power CMOS Digital Design." *IEEE Journal of Solid-State Circuits*, 27(4), pp. 473-484, 1992.

[9] S. H. Choi *et.al.*, "Novel Sizing Algorithm for Yield Improvement under Process Variation in Nanometer Technology." *Proc. of the Design Automation Conf.*, pp 454-459, 2004.

[10] A. Das *et.al.*, "Evaluating Voltage Islands in CMPs under Process Variations." *Proc. of the Intl. Conf. of Computer Design*, 2007.

[11] J. Donald and M. Martonosi, "Power Efficiency for Variation-Tolerant Multicore Processors." *Proc. of the Intl. Symp. on Low Power Electronics and Design*, pp. 304-309, 2006.

[12] D. Ernst *et.al.*, "Razor:A Low-Power Pipeline Based on Circuit-Level Timing Speculation." *Proc. of the Intl. Symp. on Microarchitecture*, pp. 7-18, 2003.

[13] R. Gonzales and M. Horowitz, "Energy Dissipation in General Purpose Microprocessors." *IEEE Journal of Solid-State Circuits,* 31(9), pp. 1277-1284, 1996.

[14] S. Herbert and D. Marculescu, "Analysis of Dynamic Voltage/Frequency Scaling in Chip-Multiprocessors." *Proc. of the Intl. Symp. on Low Power Electronics and Design*, pp 38-43, 2007.

[15] E. Humenay *et.al.*, "Impact of Parameter Variations on Multicore Chips." *Proc. of the 1st Wkshp. On Architectural Support for Gigascale Integration*, 2006.

[16] E. Humenay *et.al.*, "Impact of Process Variations on Multicore Performance Symmetry." *Proc. of Intl. Conf. on Design, Automation and Test in Europe* , pp. 1653-1658, 2007.

[17] Md. A. Hussain and M. Mutyam, "Block Remap with Turnoff: A Variation-Tolerant Cache Design Technique." Proc. of the Asia and South Pacific Design Automation Conf., pp. 783-788, 2008.

[18] IBM PowerPC 970FX RISC Microprocessor User's Manual, Version 1.6, Dec. 2005.

[19] Intel 64 and IA-32 Architectures Software Developer's Manual, System Programming Guide, Part 2, Vol-3B, Aug. 2007.

[20] C. Isci *et.al.*, "An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget." *Proc. of the Intl. Symp. on Microarchitecture*, pp. 347-358, 2006.

[21] X. Liang and D. Brooks, "Latency Adaptation of Multiported Register Files to Mitigate Variations." *Proc. of the 1st Wkshp. on Architectural Support for Gigascale Integration*, 2006.

[22] P. S. Magnusson *et.al.*, "Simics: A full system simulation platform." *IEEE Computer*, 35(2), pp. 50-58, 2002.

[23] A. J. Martin, "Towards an Energy Complexity of Computation." *Information Processing Letters*, 77(2-4), pp. 181-187, 2001.

[24] K. Meng *et.al.*, "Physical Resource Matching Under Power Asymmetry." *P=ac2 Conf.*, IBM TJ Watson Research Center, 2006.

[25] M. Monchiero *et.al.*, "Design Space Exploration for Multicore Architectures: A Power/Performance/Thermal View." *Proc. of the 20th Intl. Conf. on Supercomputing*, pp. 177-186, 2006.

[26] M. Mutyam and N. Vijaykrishnan, "Working with Process Variation Aware Caches." *Proc. of Intl. Conf. on Design Automation and Test in Europe*, pp. 1152-1157, 2007.

[27] M. Orshanksy *et.al.*, "Characterization of spatial intrafield gate CD variability, its impact on circuit performance, and spatial mask-level correction." *IEEE Transactions on Semiconductor Manufacturing*, 17(1), pp. 2-11, 2004.

[28] Predictive Technology Model, http://www.eas.asu.edu/∼ptm.

[29] K.Raghavendra and M. Mutyam, "Process Variation Aware Issue Queue Design." *Proc. of Intl. Conf. on Design Automation and Test in Europe*, 2008.

[30] J. Sharkey *et.al.*, "Evaluating Design Tradeoffs in On-Chip Power Management for CMPs." *Proc. of the Intl. Symp. on Low Power Electronics and Design*, pp. 44-49, 2007.

[31] J. Tschanz *et.al.*, "Adaptive Body Bias for Reducing Impacts of Die-to-Die and Within-Die Parameter Variations on Microprocessor Frequency and Leakage." *IEEE Journal of Solid-State Circuits*, 37(11), pp. 1396-1402, 2002.

[32] T. Wenisch *et.al.*, "SimFlex: Statistical Sampling of Computer Architecture Simulation." *IEEE Micro Special Issue on Computer Architecture Simulation*, 26(4), pp. 18-31, 2006

[33] S. C. Woo *et.al.*, "The SPLASH-2 Programs: Characterization and Methodological Considerations." *Proc. of the 22nd Intl. Symp. on Computer Architecture*, pp. 24-36, 1995.