Burst Mode Processing: An Architectural Framework for Improving Performance in Future Chip MultiProcessors

Huan Zhang* ecezhang@ucdavis.edu
Rajeevan Amirtharajah* ramirtha@ucdavis.edu
Christopher Nitta† cjnitta@ucdavis.edu
Matthew Farrens† farrens@cs.ucdavis.edu
Venkatesh Akella* akella@ucdavis.edu

ABSTRACT

A new family of chip-level multiprocessor architectures called Bright Core Multicore Processor (BCMP) is presented, in which individual cores can operate either in normal mode at nominal operating clock frequency or in burst mode (at frequencies in the range of 15 GHz or more). BCMP architectures represent a class of *temporally overprovisioned* computing systems that allow trade-offs between latency, ease of programming, and scalability of parallel programs with reliability and power consumption by taking advantage of emerging packaging and cooling technologies in the extreme nanoscale CMOS regime. Results from a preliminary analysis of BCMP are presented to demonstrate the opportunities and challenges with burst-mode processing.

1 Introduction

We have now reached an interesting junction in terms of technology scaling wherein using normal voltages (in the range of 1V to 1.2V) the logic circuits in processors will work correctly at a significantly higher clock frequency than what is permitted by the power dissipation budget. In fact, our preliminary simulations indicate that in today's soon-to-be mainstream 16nm technology, a processor with reduced-size functional units can run correctly at 15 GHz,which is roughly 5 times the nominal frequency of operation of processors today. Obviously, for thermal reasons a processing core cannot be run *continuously* at 15 GHz, but it can be run at this higher frequency for a short interval (a burst) of time. In this work we are interested in the exploring the limits and trade-offs of burst-mode processing.

A rudimentary form of burst mode operation is used in the Intel Turbo Boost 2.0 technology [11], where (when in Boost mode) the clock frequency is raised about 25% to 30% higher than the nominal mode. This mode is managed carefully by an internal PCU (package control unit), to make sure the overall reliability of the chip is not compromised. Access to the mode is also largely hidden from the programmer and applications.

There are three main differences between what we propose to do and Turbo boosting: (a) We intend to operate at a significantly higher clock frequency (as high as the basic underlying technology allows - about 15 GHz or so in 16nm technology), (b) We propose to ignore the Thermal Design Power (TDP), which is an artificial package driven constraint, and focus on a reliability constraint that is based on the underlying technology and Mean Time To Failure (MTTF), and (c) We want to expose the burst-mode capability to the programmer, so that applications can take advantage of it.

Specifically, we propose a new architecture called a BCMP (Bright Core Multi/Manycore Processor). The main feature of a BCMP is that one or more cores can temporarily operate at a significantly higher clock frequency (called F_{high}) than the nominal clock frequency (F_{nom}) of the core - we call this higher frequency mode *bright* mode[1]. We define a Thermal Duty Cycle (or TDC) to be the fraction of time a core can be in bright/burst mode before the junction temperatures of the transistors exceed a critical temperature limit. Note that the TDC of a core is not fixed - it depends on the mode the core has been running in, how long it has been running in that mode, and what is/was running on others cores (since cores are thermally coupled). Furthermore, a BCMP can be implemented in many ways. It could be a single die with every core capable of operating in both bright and regular mode, or it could be accomplished by using special cores on separate dies which are more resilient to thermal shocks mounted on an interposer using 2.5D integration. By using 2.5D packaging, it is also possible to physically distribute dies so that some of the thermal issues can be alleviated, therefore in general 2.5D packaging will provide us a better TDC (as shown in Figure 1). We expect different configurations to have different TDCs, so BCMPs offer a very rich and complex design space to investigate.

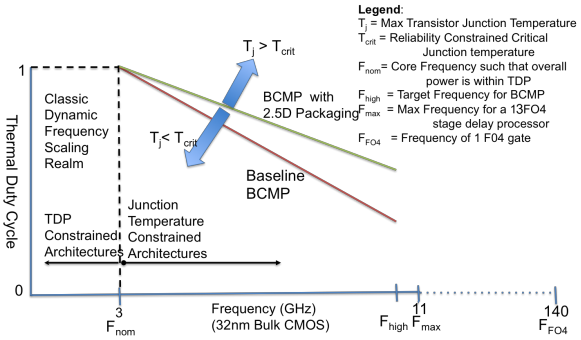


Figure 1: BCMP Architecture Design Space

Note that a BCMP is not a single design point, but rather a family of architectures (or a design space) with different

*Department of Electrical & Computer Engineering
University of California, Davis

†Department of Computer Science
University of California, Davis

WMOS In conjunction with ASPLOS 2014, Salt Lake City, UT, USA

[1]We use the words "bright" and "burst" interchangeably in this paper.

trade-offs, as shown in Figure 1. The figure assumes the use of a 32nm bulk CMOS technology, where the FO4 gate delay at 1.2V is approximately 7 ps (according to ITRS 2011 projections for high-speed logic), which represents a theoretical maximum frequency (F_{FO4}) of 140 GHz. High performance modern processors, for example, IBM Power 6, is implemented in a 13 FO4 design with 14-stage integer pipelines [18]. Assuming a pipeline stage delay of 13 FO4 gates, the maximum frequency at which a processor can be clocked in 32nm technology is around 11 GHz. For newer technology node the maximum frequency can be even higher. For simplicity, we assume TDC is a linear function of the operating frequency. Note that not all architectures in the junction temperature constrained region (to the right of the black dotted line) are legal- only those architectures that do not allow the junction temperatures T_j to exceed the critical threshold (T_{crit}) that is dictated by a reliability constraint.

2 Challenges and Applications

One of the biggest challenges has to do with handling thermal issues - dynamic power consumption increases linearly with frequency, which results in higher on-die temperatures, which in turn (exponentially) increases the static leakage power. How does one prevent burst-mode processing from leading to a thermal runaway? And how damaging is thermal cycling, and what can be done to ameliorate the problem? We have reached a point where we cannot use all the transistors that we can pack on a practical sized die simultaneously, because the heat generated by the transistors cannot be dissipated without using expensive/complex packaging techniques. However, if burst-mode operation does accelerate the aging of the cores, it may be that extra cores can be used to spread the aging, as is done in solid-state non-volatile phase change and/or flash memories.

There are also questions regarding which applications will be able to use burst-mode, how much it can be used, and how significant the resultant performance improvements will be. We believe there are a number of areas where the benefits of burst-mode processing will be apparent - for example, there are some tasks that are inherently sequential in nature. Even in the part of the program that can be parallelized there may be shared data structures that require mutually exclusive access. This constraint results in the serialization of the threads, and can seriously impact the performance and scalability of parallel programs as the number of cores on a chip increases. In the widely used MySQL database, for example, there is a shared data structure `open_cache` which tracks all tables opened by all transactions. This is protected by a global cache mutex `LOCK_open`. On average, the critical section which accesses the data structure constitutes 670 instructions out of the 40K instructions needed for the complete transaction. The contention for this critical section is quite high, and on average 5 threads are waiting to access it when run with 32 threads [19]. The contention increases exponentially as the number of threads increases, causing a severe bottleneck [20] which is unacceptable for interactive applications like Facebook. Table 2 is reproduced from [19] and shows the nature and distribution of critical sections over the set of benchmarks they chose to examine. Clearly there appears to be opportunities for improvement if one can execute these critical sections in less time, and this data is particularly promising for our burst-mode processing because these critical sections are short. Also, note that as the number of threads increases from 4 to 32, the average number of threads waiting is increasing proportionally, which means that the benefits of our proposed research are likely to be higher as the number of cores per chip increases.

Workload	% of Non-parallel instr.	% of parallel instr. in critical sections	# of disjoint critical sections	Avg. instr. in critical section	Shared/ Private	Contention			
						4	8	16	32
ep	13.3	14.6	3	620618.1	**1.0**	1.4	1.8	4.0	8.2
is	84.6	8.3	1	9975.0	**1.1**	2.3	4.3	8.1	16.4
pagemine	0.4	5.7	1	531.0	**1.7**	2.3	4.3	8.2	15.9
puzzle	2.4	69.2	2	926.9	**1.1**	2.2	4.3	8.3	16.1
qsort	28.5	16.0	1	127.3	0.7	1.1	3.0	9.6	25.6
sqlite	0.2	17.0	5	933.1	**2.4**	1.4	2.2	3.7	6.4
tsp	0.9	4.3	2	29.5	0.4	1.2	1.6	2.0	3.6
iplookup	0.1	8.0	4	683.1	0.6	1.2	1.3	1.5	1.9
oltp-1	2.3	13.3	20	277.6	0.8	1.2	1.2	1.5	2.2
oltp-2	1.1	12.1	29	309.6	0.9	1.1	1.2	1.4	1.6
specjbb	1.2	0.3	39	1002.8	0.5	1.0	1.0	1.0	1.2
webcache	3.5	94.7	33	2257.0	**1.1**	1.1	1.1	1.1	1.4

Figure 2: Characteristics of Critical Sections: Contention is the average number of threads waiting for critical sections when the workload is executed with 4, 8, 16, and 32 threads.

In addition, many data processing applications are organized as *pipelines*. For example, the famous map-reduce application has a map or filter phase which is highly parallel, followed by a reduce phase that is inherently sequential. Many signal processing, image recognition, and video encoding tasks are also performed using multiple stages with each stage requiring a different amount of processing, and the throughput is limited by the slowest stage. Thus, using burst-mode (if possible) in the slowest stage could improve the overall throughput in a power efficient way, as shown in [14].

3 BCMP Architecture

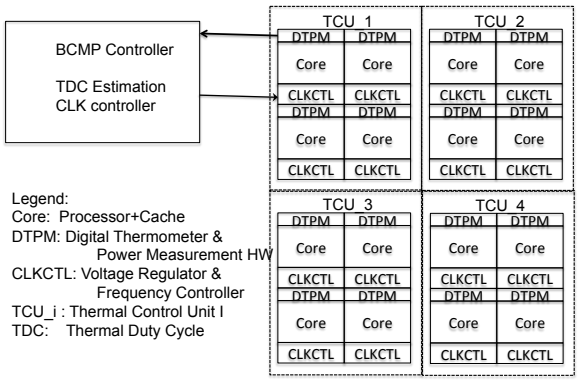


Figure 3: High Level View of Chip Multiprocessor with Bright Cores (BCMP) - 16 Core Version is shown.

A generic 16-core version of the BCMP architecture is shown in Figure 3, although as feature sizes continue to shrink future designs will almost certainly employ more cores than shown. The figure shows four Thermal Control Units (TCUs), each containing 4 cores. The number of cores per TCU is up to the designer - figure 3 is used for ease of illustration. Each core has an on-die "CLKCTL" block, which is the clock control and voltage regulation unit for the core, and a digital thermometer [12] and power measurement block called "DTPM". (Note that on-die voltage regulator modules (VRM) and digital thermometers are now integral parts of commercially available processors like Intel's Haswell.) The digital thermometer [16] has multiple sensors to track and report the hotspots, as well as the maximum temperature on the core. These temperature and power readings are sampled by the BCMP controller to estimate the Thermal Duty Cycle (TDC) of the core.

Each core has the ability to operate at two frequencies

- F_{nom}, the nominal (sustainable) clock frequency at the technology node, and F_{high}, which is typically several times higher than the nominal frequency. The choice of F_{high} depends on many factors - the technology, the FO4 gate delay (which in turn depends on whether the actual device is based on bulk CMOS, or SOI, or multigate FETs), the associated Vdd, etc. The value of F_{high} directly impacts the dynamic power consumption, which becomes even more critical if Vdd has to be raised to support the chosen F_{high} value. At higher frequencies the clock network design also becomes complicated and could consume a large fraction of the core power, though that would depend on the size of the core. The design of the memory hierarchy is also a factor, since misses become relatively more expensive when operating at F_{high}. However, a lower value of F_{high} may not result in a sufficient overall improvement in performance.

In order to run a processor at F_{high}, we are not going redesign the execution pipeline, since some modern processors like IBM Power 6 have already been designed in a way that each pipeline stage only has a dozen of FO4 delays. Thanks to the latest process technology, these processors can potentially run at a very high frequency if there is no thermal issue. However, there still exist some array-like structures (caches, ROB, etc) that must be resized such that a processor can run at F_{high} without any timing issues. In other words, during the bright mode operation, the pipeline depth does not increase; instead, some functional units are dynamically shrunk to satisfy the timing requirements at F_{high}.

The Thermal Duty Cycle (TDC) is the fraction of time that a given core can operate at F_{high} (i.e. in burst mode) without exceeding the critical junction temperatures of the core. It depends on many factors: the physical location of the core, the average power consumption, the nature of the program, the layout of the processor, the execution history, etc. For example, if the core has operated in high frequency mode recently it is likely to have a lower TDC than if it has been idle for a while (since it may still have a slightly elevated temperature.) When we wish to run in burst mode the TDC needs to be large enough to allow the core to execute a useful number of instructions, so calculating the optimal TDC is an interesting optimization problem.

There are many control decisions that must be made regarding when and how long to run in burst mode. When should a core be switched to burst-mode? How long should it stay in that mode? If there are several cores ready to switch, how does one decide which one should run in burst-mode? Can/should more than one be in burst mode at a time? For instance, memory intensive applications are not suitable to operate in bright mode; if the controller detects a memory intensive application, it should not boost the clock frequency. The control strategy is responsible for making these decisions, and there are many possible strategies that could be used. As an example, if we assume each core i has access to its current $TDC_i(t)$, the number of instructions being retired ($IR_i(t)$) and the L1 cache miss-rate ($MR_i(t)$), then a simple distributed control strategy would be to switch a core to burst-mode and keep it in burst mode as long as $(TDC_i(t) > \theta_1) \wedge (IR_i(t) < \theta_2) \wedge (MR_i(t) < \theta_3)$, where $\theta_1, \theta_2, and \theta_3$ are threshold parameters either dynamically computed or predefined by the runtime system for a given BCMP implementation. Furthermore, since accelerating serial sections is a major application of BCMP, it would be interesting to study memory behaviors of typical serial sections and identify if they are truly memory-bound.

To maximize performance improvements, programmers' inputs are rather important, since in most cases they know which part of a program can be potentially sped up much better than hardware, and the available thermal capacitance is precious in the BCMP architecture. The BCMP controller exposes a few management APIs to operating systems so that a specific amount of bright-mode time and F_{high} can be reserved and requested (in a best effort basis) from an user-mode application. By carefully scheduling bright mode processing time explicitly in a program, more performance gain could be achieved than previous burst mode processing approaches (e.g., Turbo Boosting).

4 Preliminary Analysis of BCMP

We performed preliminary simulations and analysis to study the potential and feasibility of BCMP architectures. In this analysis we were focused on answering three questions:

1. What modules need to be modified in order to allow the operation at F_{high}?
2. How large is the TDC for a 16nm processor?
3. Will there be thermal runway, and/or are current densities in burst-mode within reasonable limits?

4.1 Experimental Setup And Methodology

We performed a set of preliminary experiments and analyses to study the potential feasibility of BCMP architectures at the 16nm technology nodes. In our experiments we used McPAT v1.0 [8] for timing and power estimation. The technology parameters in McPAT were updated with ITRS 16nm DG (Dual Gate) data generated by MASTAR (The Model for Assessment of CMOS Technologies And Roadmaps) version 5.0.51 [17]. Given a processor configuration, McPAT generates area, dynamic power numbers and leakage power numbers at a given temperature and reports timing violations, if any. Although McPAT is not a transistor level simulator, it does model major components of a processor with adequate gate level details, giving fairly reasonable estimations of area, timing and power within a short time. McPAT can establish a proof of concept that a core can be clocked at a chosen frequency, and, if not, it can identify the components in the critical path that need to be redesigned. We also used a modified version of HotSpot v5.02 [4,5] to model temperature-dependent static leakage power in a closed loop manner, and validated the results against publicly available data on the modeled processors.

Our experiments were performed by first generating a floorplan for a quad-core Nehalem-like processor, based on published designs from the literature. To operate the core at 15GHz, we reduced L1 caches, branch target buffer size and number of ROB entries of a typical Nehalem processor. The values of the various processor parameters for normal and burst (bright) modes are listed in Table 1. The lowest voltage to operate at 15GHz without any timing violations reported by McPAT is 1.25V. We also performed simulations using 1.5V Vdd as a worst-case scenario. Initial simulations showed that the MASTAR model for Dual-Gate/Multi-Gate transistors overestimates leakage power (especially for temperatures higher than 370 K), since it predicts thermal runaway for 4 GHz air-cooled operation. We scaled the leakage power computed by McPAT so that our quad-core processor could run without thermal runaway at 4 GHz, consistent

	Normal Mode	Bright Mode
Processor Type	Intel Nehalem (x86-64)	
Technology Node	ITRS 16nm DG	
Core Frequency	2.66 Ghz	15.00 Ghz
Core Voltage	1.00 V	1.25/1.5 V
ROB entries	64	
BTB	18.5KB 2-way set associative	
L1 Inst. Cache	8KB directly mapped	
L1 Data Cache	16KB 2-way set associative	
L2 Cache	256KB 8-way set associative	
L3 Cache	8MB 16-way set associative	

Table 1: Simulation Setup - Nehalem x86-64 based core

with physical experiments[2].

In order to estimate the TDC, we first generate peak power numbers using both normal and bright mode processor models. The processor is started with the normal mode steady state temperature as its initial temperature. We then feed the bright mode power numbers into the HotSpot thermal simulator, and monitor the output temperature trace of HotSpot. HotSpot gives average temperatures for each functional unit, and we use the maximum one to determine if we are still able to operate in bright mode or not. When the maximum temperature climbs above a certain value (100 °C in our experiment), we switch and feed the normal mode power numbers into the simulator, and the temperature decreases as a result. After the maximum temperature falls to a certain point (75 °C in our experiment), we begin to feed the bright mode numbers again and start a new cycle. TDC is measured as the time in bright mode divided by the period of one bright-normal thermal cycle. During the first few bright-normal cycles the processor is warming up and the length of the cycle varies, so we run the simulation through enough TDC cycles for the period to become stable.

4.2 Results and Analysis

Figure 4 shows the TDC estimate for the Nehalem-like processor. In our experiment, one of the four cores can operate in bright mode if temperature permits, while other three cores are always running at the normal frequency. Thermal coupling issues between cores are not included in this preliminary analysis. Our preliminary results are encouraging - they indicate that for about 7.5% of the time one core can be continuously operated in bright mode at 1.25V. Even at 1.5V bright mode V_{dd}, the estimated TDC is still about 4.3%.

We measure TDCs with different bright mode frequencies (F_{high}), and the result is shown in Figure 5. This figure also shows the product of TDC and F_{high}, reflecting the relative number of instructions that can be executed in bright mode during a fixed time period. When $F_{high} = 5GHz$ and $V_{dd} = 1.25V$, the processor is able to spend half of the time in bright mode, while a higher F_{high} will decrease TDC to about 0.1. It is also noticeable that the decrease in TDC is much slower when $F_{high} > 10GHz$, and the product of TDC and F_{high} tends to be a constant in high-frequency region. Different applications can request different F_{high} based on their characteristics to achieve optimal performance. For example, a program with a relatively long critical section can choose a lower F_{high} during its critical section such that the entire section can be executed in bright mode. The choice of F_{high} adds a new dimension for performance optimization.

[2]`http://blog.stuffedcow.net/2012/10/intel32nm-22nm-core-i5-comparison/`

During our simulations, the Floating-Point Register Alias Table (FP-RAT) is the hottest functional unit and has the maximum power density. Since we are using peak power numbers, McPAT assumes all functional units are working at full rate (although it is very unlikely in reality). When $F_{high} = 15GHz$ and $V_{dd} = 1.25V$, the power density of FP-RAT given by McPAT is $1.82 \times 10^{8}\mathrm{W/m^2}$. We used CACTI (included in McPAT) models to estimate the worst case power density of a single CMOS gate given a typical gate width under the same condition with a activity factor of 1, and the power density ranges from $2.0 \times 10^{8}\mathrm{W/m^2}$ to $2.3 \times 10^{8}\mathrm{W/m^2}$, very close to the peak power number of FP-RAT. These numbers indicate that our simulations do reflect the worst-case scenario.

4.3 Device-level Feasibility Analysis

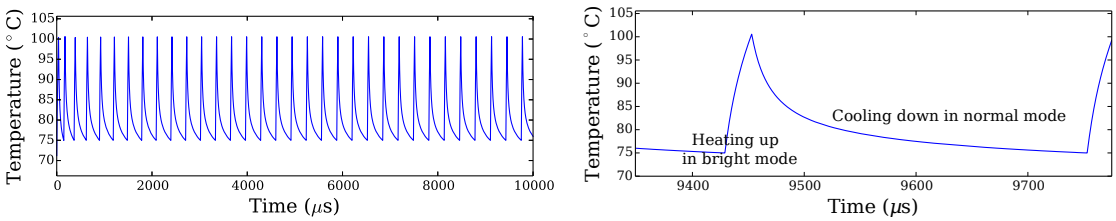
When considering bright mode operation at 15 GHz, the peak currents can potentially initiate a positive feedback loop between temperature and power consumption that leads to thermal runaway, or exceeds power supply or electromigration limits. Fortunately, thermal runaway can be eliminated as long as the processor power consumption can be drastically reduced (e.g., by entering idle mode) before a critical temperature is sustained for long periods. This critical temperature is 140-175 °C at maximum dynamic power [2] for subthreshold leakage consistent with 22 nm FinFET technology [1]. As long as this temperature is exceeded only briefly, immediately idling the processor will prevent catastrophic failure from occurring.

Aggressive pipelining and voltage scaling will likely be needed to implement 15 GHz bright mode operation. A single FO4 22 nm inverter delay can be estimated as 4.7 ps, neglecting interconnect capacitance. For an Ivy Bridge core running at 4 GHz and 1.26 V, dissipating 101 W and drawing 80 A, a clock period corresponds to 54 FO4 delays. Bright mode operation at 15 GHz can be achieved by reducing each pipe stage to 15 FO4 delays and raising V_{DD} to 1.5 V, assuming α-law MOSFET drain current scaling with $\alpha = 1.2$ [17]. Such a processor would consume 4.1× the present TDP (533 W and 130 W, respectively) and draw 3.8× the present max current (356 A and 95 A, respectively). This may necessitate approximately 3000 package balls, a 1.5× increase over present LGA 2011 socket-compatible packages [3].

High currents imply high current densities on-chip, potentially leading to interconnect failures induced by electromigration. At 1.26 V, the current density for a single via is approximately 11.8 MA/cm^2, below the critical current density of 14 MA/cm^2 at which the probability of failures due to electromigration are considered negligible [13]. At 1.5 V (for 15 GHz operation), the current density becomes 15.1 MA/cm^2, just exceeding the critical current density and yielding a small increase in failures. Quantifying the magnitude of this increase is part of our future work.

5 Related Work

As described in the introduction burst-mode processing can be viewed as generalizing Turbo Boosting [11] by decoupling it from TDP and exposing the capability to the programmer. Computational sprinting [15] is based on the insight that many mobile applications do not demand sustained performance but instead perform short bursts of computation in response to sporadic user activity. To improve responsiveness for such applications, cores that are otherwise powered down (to stay within the TDP limit) are activated



(a) Temperature Profile (10ms operation)

(b) Zoom into One Cycle

Figure 4: Thermal Duty Cycle of a Nehalem-like core in ITRS 16nm dual gate when bright mode $F_{high} = 15Ghz, V_{dd} = 1.25V$. Thermal Duty Cycle is approximately 7.5% and is quite steady.

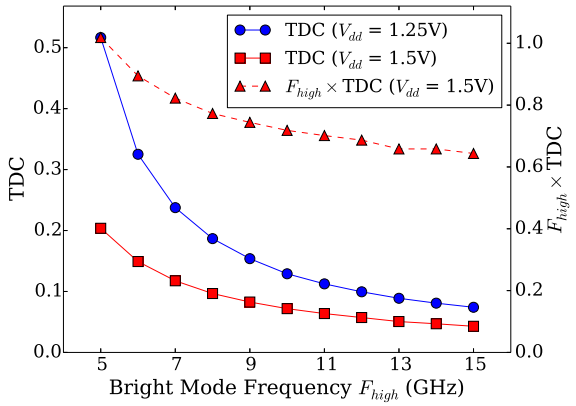


Figure 5: TDC vs. Bright Mode Frequency (F_{high})

for short periods of intense parallel computation, exceeding the thermal power budget temporarily. The key difference between the computational sprinting work and BCMP is that that we are trying to run the processors at a significantly higher clock frequency for very short periods of time. Our motivation is to improve the performance of parallel and pipelined programs by operating the cores in burst-mode to power through serial bottlenecks (such as during synchronization, pipeline imbalances, etc.), as opposed to improving silicon utilization or addressing the dark silicon problem. Given the significantly higher frequencies used in our approach, we have to address reliability issues due to thermal cycling and exploit 2.5D packaging which computational sprinting does not have to.

There has been work recently in the area of combating process variations that shares some superficial similarity with burst mode processing. For example, in Booster [10] performance heterogeneity due to process variations is reduced by dynamically switching the voltage/frequency of a core to one of the two available voltages without exceeding the overall power budget. In BubbleWrap [7], cores that are rendered power efficient due to process variations are used for improving throughput while others treated as expendable by running them at higher voltages for accelerating sequential code till they fail. Both of these approaches are clearly different from BCMP, where we focus on managing the thermal and reliability issues raised by running cores at 15 GHz and beyond.

There has also been work in the heterogeneous multicore processor area [9] where the authors propose the use of some small cores for executing *parallel* code only, in addition to larger cores for exploiting ILP. Ipek et al [6] propose Core Fusion, where multiple small cores can be combined (fused) to form a powerful core at runtime that can be used to speed up the serial portion of programs. Suleman et al. [19] also propose a heterogeneous CMP with a large core to accelerate critical sections of parallel programs, although it has some drawbacks such as false serialization when critical sections that are protected by different locks are all executed sequentially on the single large core, which could degrade performance (and with nested critical sections possibly introduce deadlocks and complications with interrupts). Our approach does not suffer from these drawbacks. In addition, we do not have to ship serial sections of code to a single large core, eliminating the associated communication costs and modifications to the ISA and libraries. Furthermore, our work differs from existing work because we are proposing a more aggressive approach that violates the TDP of a processor. All of the existing research addressing the serial bottleneck problem still relies on staying within the TDP. We are intentionally willing to expend more power to reduce latency, which introduces many new challenges that we propose to address.

6 Conclusions and Future Work

Burst mode processing represents a class of *temporally overprovisioned* computing architectures with a rich design space to exploit the trade-offs between ease of programming, latency, and scalability of parallel programs along with power consumption and reliability by leveraging emerging packaging, cooling, and extreme nanoscale CMOS technologies. Our preliminary analysis indicates that burst-mode processing and the abstraction of thermal duty cycle are promising and can be supported with modest changes to the existing processor architectures but a more detailed analysis of the impact of reliability due to thermal cycling needs to be investigated. Future work includes developing tools for modeling thermal transients and the reliability of processors operating in the burst-mode and improvements in performance at the application level.

Acknowledgement: Part of the research reported in this paper is supported by NSF grant CCF:1116897.

7 References

[1] C. Auth, "22-nm fully-depleted tri-gate CMOS transistors," in *IEEE 2012 Custom Integrated Circuits Conf. (CICC)*, September 2012.

[2] J. H. Choi, A. Bansal, M. Meterelliyoz, J. Murthy, and K. Roy, "Self-consistent approach to leakage power and temperature estimation to predict thermal runaway in FinFET circuits," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 11, pp. 2059–67, November 2007.

[3] I. Corporation, "Intel® core™ i7 processor family for the lga-2011 socket," Datasheet, November 2012, available online (556 pages). [Online]. Available: http://www.intel.com/content/dam/doc/datasheet/

core-i7-lga-2011-datasheet-vol-2.pdf

[4] W. Huang, S. Ghosh, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "Hotspot: Thermal modeling for cmos vlsi systems," *IEEE Transactions on Component Packaging and Manufacturing Technology*, 2005.

[5] W. Huang, K. Skadron, S. Gurumurthi, R. Ribando, and M. Stan, "Differentiating the roles of ir measurement and simulation for power and temperature-aware design," in *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*, 2009, pp. 1–10.

[6] E. Ipek, M. Kirman, N. Kirman, and J. F. Martinez, "Core fusion: accommodating software diversity in chip multiprocessors," in *Proceedings of the 34th annual international symposium on Computer architecture*, ser. ISCA '07. New York, NY, USA: ACM, 2007, pp. 186–197. [Online]. Available: http://doi.acm.org/10.1145/1250662.1250686

[7] U. Karpuzcu, B. Greskamp, and J. Torrellas, "The bubblewrap many-core: Popping cores for sequential acceleration," in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, 2009, pp. 447–458.

[8] S. Li, J. H. Ahn, J. B. Brockman, and N. P. Jouppi, "McPAT 1.0: An Integrated Power, Area, and Timing Modeling Framework for Multicore Architectures," HP Laboratories, Palo Alto, CA, USA, Tech. Rep., 2013. [Online]. Available: http://www.hpl.hp.com/research/mcpat/McPATAlpha_TechRep.pdf

[9] P. Michaud, Y. Sazeides, and A. Seznec, "Proposition for a sequential accelerator in future general-purpose manycore processors and the problem of migration-induced cache misses," in *Proceedings of the 7th ACM International Conference on Computing Frontiers*, ser. CF '10. New York, NY, USA: ACM, 2010, pp. 237–246. [Online]. Available: http://doi.acm.org/10.1145/1787275.1787330

[10] T. Miller, X. Pan, R. Thomas, N. Sedaghati, and R. Teodorescu, "Booster: Reactive core acceleration for mitigating the effects of process variation and application imbalance in low-voltage chips," in *High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on*, 2012, pp. 1–12.

[11] A. Naveh, D. Rajwan, A. Ananthakrishnan, and E. Weissmann, "Power management architecture of the 2nd generation Intel® Core™ microarchitecture, formerly codenamed Sandy Bridge," in *Hotchips*, 2011.

[12] A. Naveh, E. Rotem, A. Mendelson, S. Gochman, R. Chabukswar, K. Krishnan, and A. Kumar, "Power and Thermal Management in the Intel® Core™ Duo Processor," *Intel® Technology Journal*, vol. 10, pp. 109–122, May 2006. [Online]. Available: http://www.intel.com/technology/itj/2006/volume10issue02/vol10_iss02.pdf

[13] A. Oates and M. Lin, "Electromigration failure distributions of Cu/Low-k dual-damascene vias: Impact of the critical current density and a new reliability extrapolation methodology," *Device and Materials Reliability, IEEE Transactions on*, vol. 9, no. 2, pp. 244–254, 2009.

[14] J. Oliver, R. Rao, P. Sultana, J. Crandall, E. Czernikowski, L. W. Jones IV, D. Franklin, V. Akella, and F. T. Chong, "Synchroscalar: A multiple clock domain, power-aware, tile-based embedded processor," in *Proceedings of the 31st annual international symposium on Computer architecture*, ser. ISCA '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 150–. [Online]. Available: http://dl.acm.org/citation.cfm?id=998680.1006714

[15] A. Raghavan, Y. Luo, A. Chandawalla, M. Papaefthymiou, K. Pipe, T. Wenisch, and M. Martin, "Computational sprinting," in *High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on*, feb. 2012, pp. 1 –12.

[16] E. Rothem, J. Hermerding, C. Aviad, and C. Harel, "Temperature measurement in the intel core duo processor," in *Proceedings of the Twelfth International Workshop on Thermal Investigations of ICs (THERMINIC'06)*, Aug. 2006.

[17] Semiconductor Industry Association, "International technology roadmap for semiconductors 2011," 2011. [Online]. Available: http://public.itrs.net/Links/2011ITRS/Home2011.htm

[18] B. Stolt, Y. Mittlefehldt, S. Dubey, G. Mittal, M. Lee, J. Friedrich, and E. Fluhr, "Design and Implementation of the POWER6 Microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 21–28, Jan 2008.

[19] M. A. Suleman, O. Mutlu, M. K. Qureshi, and Y. N. Patt, "Accelerating critical section execution with asymmetric multi-core architectures," in *Proceedings of the 14th international conference on Architectural support for programming languages and operating systems*, ser. ASPLOS '09. New York, NY, USA: ACM, 2009, pp. 253–264. [Online]. Available: http://doi.acm.org/10.1145/1508244.1508274

[20] P. Zaitsev, "Opening Tables scalability," Online Article, MySQL Performance Blog, November 2006, available online. [Online]. Available: http://www.mysqlperformanceblog.com/2006/11/21/-opening-tables-scalability