USING MCD-DVS FOR DYNAMIC THERMAL MANAGEMENT PERFORMANCE IMPROVEMENT

Pedro Chaparro, Grigorios Magklis, José González and Antonio González
Intel Barcelona Research Center
Intel Labs - UPC
Ed. Nexus II, Jordi Girona 29 3a
08034 Barcelona (SPAIN)
Phone: +34 93413 77 63
Fax: +34 93 413 77 55
Email: {pedro.chaparro.monferrer, grigorios.magklis, pepe.gonzalez, antonio.gonzalez}@intel.com

ABSTRACT

With chip temperature being a major hurdle in microprocessor design, techniques to recover the performance loss due to thermal emergency mechanisms are crucial in order to sustain performance growth. Many techniques for power reduction in the past and some on thermal management more recently have contributed to alleviate this problem. Probably the most important thermal control technique is dynamic voltage and frequency scaling (DVS) which allows for almost cubic reduction in power with worst-case performance penalty only linear. So far, DVS techniques for temperature control have been studied at the chip level. Finer grain DVS is feasible if a Globally-Asynchronous Locally-Synchronous (GALS) design style is employed. GALS, also known as Multiple-Clock Domain (MCD), allows for an independent voltage and frequency control for each one of the clock domains that are part of the chip. There are several studies on DVS for GALS that aim to improve energy and power efficiency but not temperature. This paper proposes and analyses the usage of DVS at the domain level to control temperature in a clustered MCD microarchitecture with the goal of improving the performance of applications that do not meet the thermal constraints imposed by the designers.

KEY WORDS: Multiple clock domain architectures, GALS, DTM, dynamic frequency and voltage scaling

INTRODUCTION

Power directly translates into heat which must be removed from the processor die in order to keep the silicon temperature inside a "safe" range. Power density is increasing due to the fact that frequency and leakage current are scaling up so much that their effect on power cannot be offset by decreasing the supply voltage. Such trend makes the cost of the cooling system grow and challenges the performance benefits that can be obtained by the ever growing transistor density. This results in a cooling system cost in the order of $1-$3 or more per Watt when the average power exceeds 40W [1][2], which represents a significant part of the total cost of the chip. This is especially important for data centers where air conditioning is a main contributor in the total cost [3]. In addition, circuit reliability depends exponentially on operating temperature. Temperature variations account for over 50% of electronic failures [4].

Another problem is due to the scaling down of supply voltage to reduce dynamic power consumption. To counteract the effect on gate delay, the threshold voltage is also scaled down. However, lowering the threshold voltage impacts leakage exponentially. Furthermore, leakage power is also exponentially dependent on temperature. This is the reason that projections show leakage power reaching the same levels as dynamic power [1][5].

Traditionally the cooling system of a processor has been designed to support the worst case temperature so that peak performance is guaranteed. Because of both the increasing cost of the cooling solution and form factor constraints—especially in mobile computers—the cooling system is nowadays designed for common case power dissipation. In case of a temperature rise, a thermal emergency mechanism is in charge of restoring the processor to its operating temperature. Despite the penalty of this mechanism, this solution has been adopted because the processor spends most of the time running at much lower temperatures than the worst-case scenario. Additional proactive techniques try to get some of the performance back by avoiding triggering the emergency mechanism.

Dynamic voltage and frequency scaling (DVS for short) has long been used to deal with thermal emergencies [6]. Whenever the processor starts heating up, a controller decides to slow the processor down to avoid triggering the emergency mechanism.

Globally Asynchronous Locally Synchronous (GALS) systems have the unique ability to operate different parts of the chip (called *domains*) at different frequency and voltage, which allows applying DVS independently to different parts of the processor [7]. It has been shown that per-domain adaptation is significantly more energy efficient compared to global adaptation [8][6][9][10]. GALS architectures also reduce complexity and save power dissipation of the clock distribution, which constitutes a large part of the total processor power [11][12].

On the other hand, it has been shown that clustering reduces the complexity of large structures, such as issue queues and register files. This allows for faster clock frequency and reduced power dissipation [13][14][15][16][17]. Clustering also facilitates run-time power control through fine-grained adaptation of resources and achieves a significant reduction of temperature due to an effective distribution of the activity

among the different clusters both in the frontend [18] and the backend [19][20].

Combining clustering with GALS results in a highly energy-efficient design with the capability of fine-grain adaptation [21]. So far, DVS techniques for temperature control have been studied at the chip level [6]. This work aims to improve performance for thermally constrained designs. In particular it takes advantage of the fine-grain DVS capabilities of GALS microarchitectures to avoid thermal crisis situations and their associated performance penalty.

CLUSTERED GALS MICROARCHITECTURE

Figure 1 shows the details of our clustered GALS microarchitecture. The processor has three clock domains, shaded light grey in Figure 1: frontend, backend and memory. The frontend domain contains the fetch and dispatch logic. Fetch utilizes a branch predictor, a trace cache, and an IA32 decoder that decodes complex x86 instructions into simple micro-ops. Dispatch renames register operands, allocates resources for new instructions and steers micro-ops to one of the backend execution clusters [17]. The frontend also includes the reorder buffer and the commit logic of the processor. The reorder buffer and the rename table have been partitioned in order to make them more thermal efficient [18].

The backend domain contains the out-of-order execution and the first-level data cache of the microprocessor. It follows a clustered design, with two execution clusters. Each cluster includes the integer and floating-point issue queues, their corresponding register files, and the integer and floating-point execution units. The clusters share both the load-store queue and the first-level data cache. Address calculations occur at the execution clusters. Special *copy* micro-ops communicate register values among the clusters using point-to-point links [16][22].

GALS systems, by design, assume unrestricted clock skew among domains. This allows utilizing local-only clocking in the first place and gives us the ability to run each domain at a different frequency. The disadvantage is that inter-domain communication must be correctly synchronized to avoid meta-stability [23]. In our microarchitecture, we use the mixed-clock FIFO design of Chelcea and Nowick [24], with the synchronizer circuit by Nyström and Martin [25].

Figure 2 shows the timing of the synchronizer for a mixed-clock FIFO with write-clock CLK_{IN} and read-clock CLK_{OUT}. A data value is written at clock edge 1. If the time difference between edges 1 and 2 is greater than the synchronizer's delay then the data will be visible at the read interface at edge 2. Otherwise, the synchronizer will not allow the data to be visible until edge 4. In our simulations, this delay is set to 30% of CLK_{OUT}, following Sjogern and Myers [26].

Due to the clustered nature of our design, all the mixed-clock FIFOs utilized in the GALS design (dark grey queues in Figure 1) already existed, as regular FIFOs, in the fully synchronous design. This allows for a natural separation with minimal changes in the microarchitecture. Moreover, since FIFOs provide natural buffering and usually reside off the critical path of the microprocessor, our GALS modifications result in minimal performance loss due to synchronization.

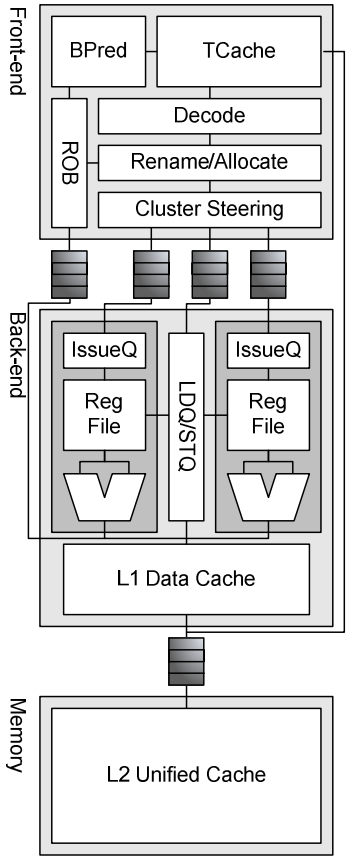


Figure 1. Clustered GALS Microarchitecture

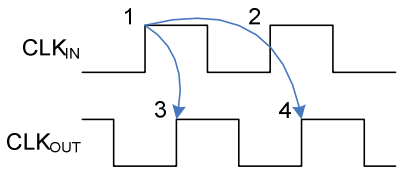


Figure 2. Synchronizer timing

Table 1 Voltage and frequency levels

Level	mV	MHz
0	700	3100
1	734	3400
2	767	3700
3	800	3900
4	834	4200
5	867	4400
6	900	4700
7	934	4900
8	967	5100
9	1000	5400
10	1034	5600
11	1067	5800
12	1100	6000

In addition, each one of the domains has independent voltage and frequency control. Similar to previous studies, we assume domains can execute through voltage and frequency changes [12][26][27][28]. Our microprocessor has a limited range of voltages and frequencies, shown in Table 1.

DYNAMIC THERMAL MANAGEMENT

Skadron et al. evaluate in [6] different thermal control mechanisms with the goal of maximizing performance in the presence of potential thermal threshold violations. In their evaluation, the best performing technique was *Temperature-Tracking Frequency Scaling* (TTDFS). In TTDFS the processor is clocked above the conservative frequency that

guarantees no timing errors. The algorithm detects when the temperature is growing excessively so that correct timing cannot be guaranteed and scales frequency down to a safe level. This technique is unique in the sense that can exceed the thermal threshold as long as the frequency is scaled to meet the timing constraints.

TTDFS is orthogonal to thermal management techniques. It is a performance-improvement technique based on relaxing the maximum frequency limit due to the circuit timing. This technique is not aimed at guaranteeing reliability and is not designed to manage thermal crisis situations.

The rest of the techniques from the same study are aimed at guaranteeing physical reliability. The best of them is *Migrating Computation*, which consists of using of spare units to migrate the activity if the temperature of a unit grows excessively [6][18][19][29]. This is not a viable technique if there are no spare units to migrate the activity.

A third technique, with less slowdown is DVS with an "ideal" PID controller [30]. In that scheme, it is assumed that the processor can continue executing while changing the voltage and frequency levels. It seems a valid assumption based on already existing data and products [12][26][27][28]. The authors claim that the DVS scheme they study is penalized because of slowing down the full chip compared to other techniques that are fine-grained.

FINE-GRAIN DVS FOR THERMAL MANAGEMENT

In this work we propose to use GALS microarchitectures to reduce the granularity at which DVS is applied, in order to achieve fine grain adaptation. In particular, it is assumed that the frontend and the backend can run simultaneously at different frequency and voltage levels and that both can continue executing while changing the voltage/frequency levels.

The algorithm for GALS uses an independent PID controller for each domain to decide the proper voltage-frequency level to run at, depending on the proximity of the peak temperature to a given threshold. The same PID configuration is used in all the domains—the one used in the global DVS.

A configuration with different PIDs per domain requires the presence of several thermal sensors. This is needed in order to measure the temperature in different functional blocks. This way, both the frontend and the backend can decide which the peak temperature in each domain is. This is a reasonable assumption since existing microprocessors already include several of these thermal sensors [32].

The DVS algorithm is invoked every 100K cycles. At that time, the information regarding the temperature of each block of the processor is gathered and is sent to the PID controller. The PID computes the frequency-voltage execution level for the next interval (independently for the frontend and the backend) and the changes are applied to the upcoming execution interval. It takes some cycles for the processor to reach the new frequency and voltage but execution is never stopped.

A backup mechanism is assumed in case the PID is unable to contain the temperature inside the safe margins. The mechanism consists of an operating system (O.S.) context saving mechanism that resumes execution after a cool-down interval. This mechanism is used to "penalize" the mechanisms when they are not able to guarantee execution under the thermal threshold. However, the goal is to avoid reaching that situation.

Table 2. Processor configuration

Frontend	
Fetch	24K micro-op trace cache, 6 micro-ops/cycle, 5 cycle fetch-to-dispatch
Dispatch: decode, rename and steer	6 micro-ops/cycle, 1 cycle latency, plus 1 cycle wire delay to mixed-clock FIFOs
Reorder Buffer	512 entries, commit 6 micro-ops/cycle
Backend (configuration shown per cluster)	
Mixed-clock FIFOs	1 FIFO per issue queue, 24 entries each
Issue queues	48-entry INT, 2 micro-ops/cycle 48-entry FP, 2 micro-ops/cycle 96-entry MOB, 1 micro-ops/cycle 24-entry COPY, 1 micro-ops/cycle
Register file	256-entry INT register file 256-entry FP register file
Inter-cluster communication	bi-directional point-to-point link, 1 cycle latency, 1 copy/cycle
First level cache	32KB, 4-way, 3 cycle hit, 2 read ports, 1 write port, 256-entry Load/Store Queue
Memory	
Second level cache	2MB, 16-way, 13 cycle hit, ≥ 500 cycle miss 1 read port, 1 write port

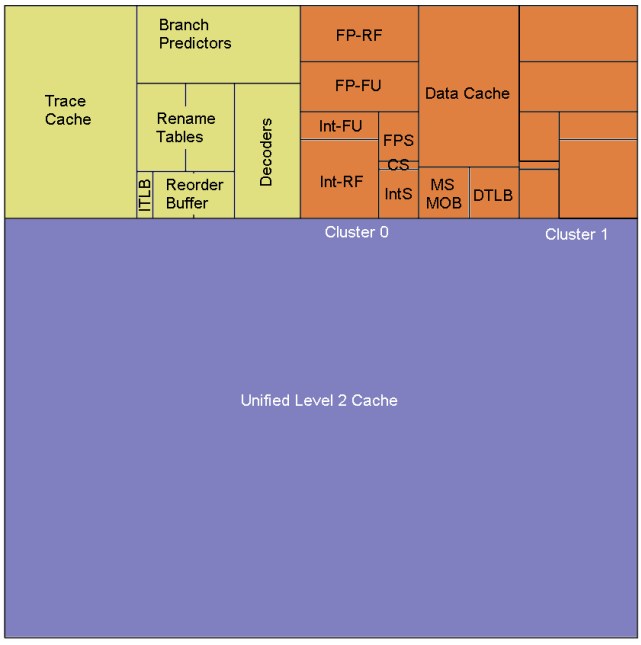
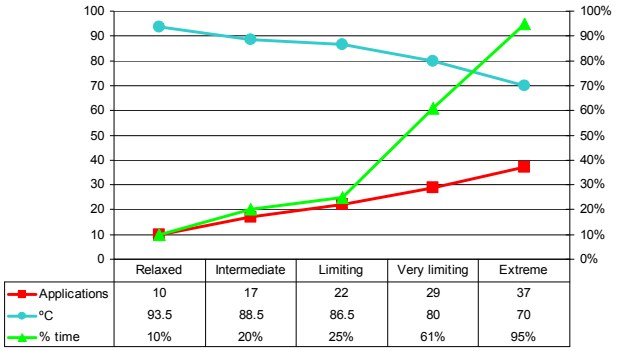


Figure 3. Processor floorplan

EVALUATION

Experiments have been conducted using an execution-driven simulator that runs IA32 binaries. Table 2 summarizes the main parameters of the architecture. The simulator includes a



	Relaxed	Intermediate	Limiting	Very limiting	Extreme
Applications	10	17	22	29	37
°C	93.5	88.5	86.5	80	70
% time	10%	20%	25%	61%	95%

Figure 4. Experiment details

dynamic power model similar to Wattch [35], a leakage model (including the exponential relationship with temperature) and a full-system thermal model similar to some others proposed in the literature [6][20].

As far as the thermal model is concerned, at the beginning of the simulation we assume that the processor has been running for a long time, dissipating its nominal average dynamic power (obtained for 50M instructions) and the leakage corresponding to its temperature, until temperature converges. In this way, simulations are started with the processor already warm. During normal execution, every 100K cycles the temperature is recomputed using the per-block dissipated power.

Figure 3 shows the floorplan of the processor. We assume a processor designed at 45nm. Areas were computed using an enhanced version of Cacti [31] for cache-like structures, and scaling down the rest of the structures from current designs. The thermal solution attached to the die of the processor consists of a copper heat spreader, in contact with the die, whose size is 3.1x3.1x0.23cm (similar to the one used in the Pentium® 4 Northwood processor [33]). On top of it there is a copper heat sink of 7x8.3x4.11cm [33].

For the evaluation process, 25 SPEC2000, 6 MediaBench and 6 MineBench applications are run. Standard reference input sets are used to select a trace from the middle of the execution. Traces are run up to a billion instructions when available (always at least 400M instructions are run).

For each thermal limit, benchmarks are classified according to the performance loss incurred when using only the backup mechanism. The 1/3 with the highest loss is classified in the "Very High" category. The 1/3 with the smallest loss is classified in the "Low" category. The intermediate benchmarks are classified in the "High" category. Note that this classification depends on the thermal limit selected. It is not a static classification depending on the properties of each application but depends instead on its response to the thermal limit and the backup mechanism. A configuration that only uses the backup mechanism (O.S. context saving) is also simulated for comparison purposes and in order to do the benchmark classification previously indicated.

To select the limits, all benchmarks were run with no thermal constraints to obtain the different profiles to be able to set a proper thermal threshold. In addition, this reports the peak performance each benchmark can provide. Different thermal limits are studied to obtain a representative evaluation. Figure 4 details the percentage of time the suite would run beyond the threshold if there were no thermal constraints. Also, the number of applications that reach each one of the limits is depicted.

Local vs. Global DVS

Figure 5 depicts the performance of the different applications and sets of applications normalized to the peak performance (with no thermal constraints). Results are only shown for the intermediate thermal threshold. Results for the other thermal thresholds are consistent with the ones in Figure 5.

Different conclusions can be extracted from these results. First, it can be observed that the O.S. mechanism penalizes performance a lot (more than 20% in some applications). Second, it can be seen that global DVS is able to recover most of this performance loss (for the limited applications, only 11% of the performance is lost). Third, Local DVS outperforms global DVS both on average and for every single application (as well as the O.S. context saving-only scheme).

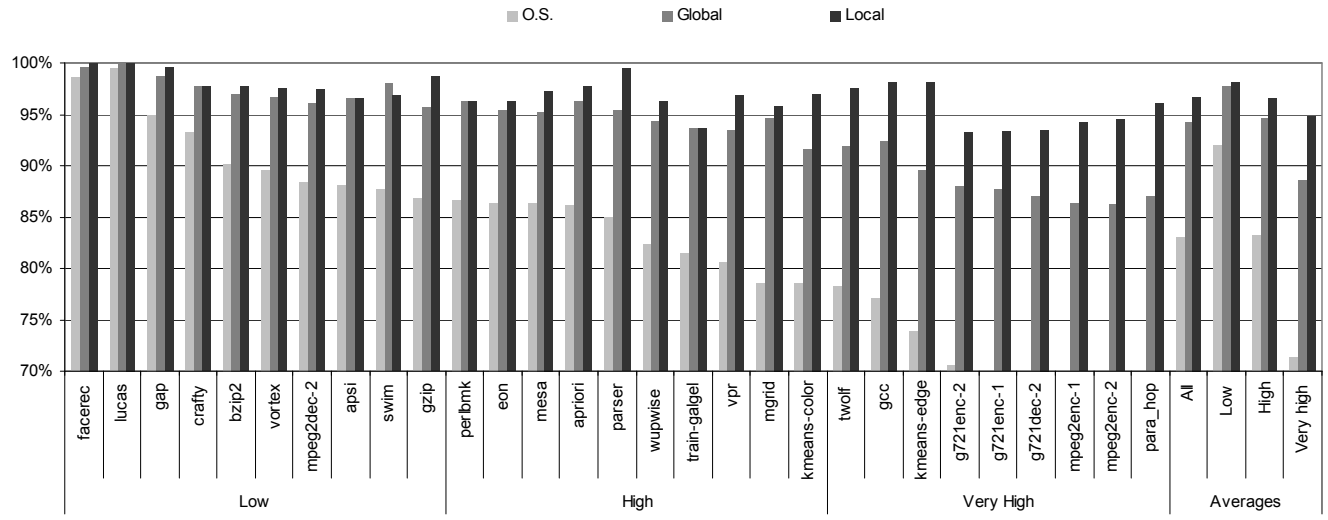


Figure 5. Performance normalized to peak performance for the Very Limiting threshold

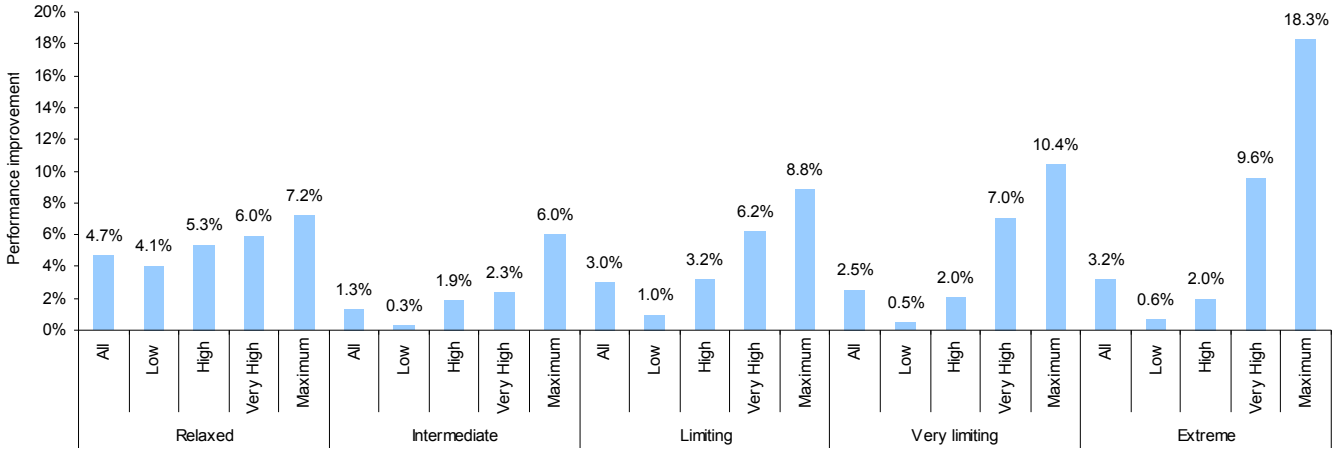


Figure 6. Performance improvement of local DVS compared to global DVS

For the most limited applications the average performance lost compared to the peak is only 5%. Obviously, the less limited an application is the higher its performance is and the smaller the room for performance improvement.

In addition, the thermal emergency threshold is violated neither in the global nor in the local version of the controller.

Thermal Threshold Sensitiveness

Figure 6 shows the performance improvement of local DVS compared to global DVS for each one of the thermal limits and benchmark groups. It can be observed that the lower the thermal threshold the higher the performance improvement, for constrained applications. Note that this does not apply when comparing the "Relaxed" and the "Intermediate" threshold performance. Remember that the applications tested vary according to the thresholds. In particular, there are few applications that violate the thermal threshold in the "Relaxed" threshold so that the averages are less meaningful than for the rest of the limits. If this analysis is done in particular applications that are tested for different thresholds, the relationship between thermal threshold and performance is valid.

As pointed out before, the stated trends are maintained for the different thermal thresholds. Only when the threshold is set extremely high ("Relaxed") or extremely low ("Extreme") can we find some particular applications showing strange behavior. This is not representative since it occurs for very few applications under very particular conditions.

Our proposed technique exhibits good performance improvement across the whole range of applications and performs especially well for the most constrained ones. For example, in the "Very limiting" threshold although the average performance improvement of all applications that spend some time beyond the threshold is 2.5%, for applications really constrained this average grows up to 7% with some applications improving by more than 10%.

RELATED WORK

Dynamic thermal management and temperature control is a hot topic presently. Huang et al. [34] propose a framework to maximize energy savings and to guarantee that temperature remains under a certain threshold. The framework combines a number of energy-management techniques, such as voltage-frequency scaling and sub-banking of the data cache among others. Brooks and Martonosi [35] propose a set of control techniques evaluated on top of different triggering mechanisms aiming at reducing thermal emergencies. They use the average power in an interval as a proxy for temperature.

Skadron et al. [6][30] propose a thermal simulator based on the duality between heat transfer and electrical phenomena. Several techniques are proposed to control peak temperature and to reduce thermal emergencies. Lim et al. [36] propose a secondary ultra-low power pipeline that is used when a given temperature threshold is exceeded. Heo et al. [29] study the impact of activity migration, among replicated units, on power density. Donald et al. [37] address design issues for SMT and CMP architectures, and Ghiasi et al. [38] for dual-core processors. Current commercial processors such as the Pentium® M [39] or the PowerPC [32] implement thermal monitors to control the temperature of the chip.

Globally Asynchronous Locally Synchronous systems were first introduced by Chapiro [7]. Since then there have been several published works on GALS and dynamic voltage and frequency scaling. Iyer and Marculescu [12] propose a superscalar microprocessor with five domains: fetch, decode and rename, integer pipeline, floating-point pipeline, and memory pipeline (includes first level cache). Semeraro et al. [27][40] propose a Multiple Clock Domain (MCD) processor, with four domains: frontend (fetch and dispatch), integer, floating-point, and memory (with first and second level cache).

Zhu et al. [10] propose an enhanced MCD microarchitecture. Magklis et al. [21] combine clustering with GALS into a Clustered Multiple Clock Domain (CMCD) design. The CMCD consists of four backend clusters (each with a local first level cache), a shared frontend, and a shared second level cache each in a separate domain. They also propose a mathematical model that relates the fetch queue utilization, the branch prediction accuracy, the frontend frequency and the application performance. They use this model to construct a control mechanism to adapt the voltage and frequency of only the frontend domain, achieving close to optimal results.

Semeraro et al. propose an interval-based microarchitecture-level control mechanism for the domains of the MCD (all but the frontend), called the Attack/Decay [40]. Wu et al. [42] model the MCD domains as queue systems and propose a feedback control DVS system based on a Proportional-Integral (PI) controller. The controller uses the occupancy of the domain input queue over some interval of time and responds with a frequency for the upcoming interval. The goal is to maintain occupancy close to a pre-defined nominal value. The authors also provide a rigorous analysis of their control system and its stability. We et al. [28] propose an event-driven DVS mechanism for the MCD that reacts to workload changes instead of making decisions at fixed time intervals. The controller utilizes both the queue occupancy and the rate of change of the occupancy.

All of the above designs separate the pipeline very differently from our work. We separate the pipeline in between logical pipeline stages. The above studies divide a logical pipeline stage according to the type of operation performed (integer, floating-point, memory). The latter separation results in non-deterministic latency in the issue/wake-up loop of operations of different types (e.g. an integer operation depending on a load).

CONCLUSIONS

Fine-grain DVS is feasible if a Globally-Asynchronous Locally-Synchronous (GALS) design style is employed. GALS allows for an independent voltage and frequency control for each one of the clock domains that are part of the chip. Several studies on DVS for GALS aim to improve energy and power efficiency but not temperature. This paper proposes and analyses the usage of DVS at the domain level to control temperature in a clustered MCD microarchitecture with the goal of improving the performance of applications that do not meet the thermal constraints imposed by the designers.

To the best of our knowledge, this is the first work that proposes the usage of GALS microarchitectures for thermal control. This is also the first work that quantifies the performance improvement of doing fine-grain DVS over global DVS. Our experiments show that local DVS achieves better results compared to global DVS: some high-power applications have a performance improvement ranging from 6% to 18% depending on thermal threshold.

In this work, the same PID configuration is employed per-domain to quantify the benefit of applying localized DVS. This mechanism although it achieves good performance it, may not be best one since a better tuning of the PIDs could result in even better performance. In the future we plan to investigate designs where the PIDs are tuned per domain or per thermal threshold.

REFERENCES

[1] S. Borkar. “Design Challenges of Technology Scaling”. IEEE Micro, 19(4), pp. 23-29, 1999.

[2] S. Gunther et al. “Managing the Impact of Increasing Microprocessor Power Consumption”. Intel Technology Journal, Q1 2001.

[3] J. Moore et al. “Going beyond CPUs: The Potential of Temperature-Aware Solutions for the Data Center”. Proceedings of the First Workshop of Temperature-Aware Computer Systems at ISCA, 2004.

[4] L.-T. Yeh and R.C. Chu. “Thermal Management of Microelectronic Equipment: Heat Transfer Theory, Analysis Methods and Design Practices”. ASME Press, New York, NY, 2002.

[5] V. De and S. Borkar. “Technology and Design Challenges for Low Power and High Performance”. Proceedings of the International Symposium on Low Power Electronics Design pp. 163-168, 2000.

[6] K. Skadron et al. “Temperature-Aware Microarchitecture”. Proceedings of the 30th Annual International Symposium on Computer Architecture, April 2003.

[7] D. M. Chapiro. “Globally Asynchronous Locally Synchronous Systems”. PhD thesis, Stanford University, 1984.

[8] G. Magklis et al.. “Profile-based Dynamic Voltage and Frequency Scaling for a Multiple Clock Domain Processor”. Proceedings of the 30th Annual International Symposium on Computer Architecture, June 2003.

[9] Q. Wu et al. “Voltage and Frequency Control with Adaptive Reaction Time in Multiple-Clock-Domain Processors”. Proceedings of the 11th International Symposium on High-Performance Computer Architecture, February 2005.

[10] Y. Zhu et al. “A High Performance, Energy Efficient, GALS Processor Microarchitecture with Reduced Implementation Complexity”. Proceedings of the International Symposium on Performance Analysis of Systems and Software, March 2005.

[11] A. Hemani et al. “Lowering Power Consumption in Clock by Using Globally Synchronous Locally Synchronous Design Style”. Proceedings of the 36th Conference on Design Automation, June 1999.

[12] A. Iyer and D. Marculescu. “Power and Performance Evaluation of Globally Asynchronous Locally Synchronous Processors”. Proceedings of the 29th Annual International Symposium on Computer Architecture. May 2002.

[13] V. Agarwal et al. “Clock Rate versus IPC: the End of the Road for Conventional Microarchitectures”. Proceedings of the 27th International Symposium on Computer Architecture, 2000.

[14] M. Bohr. “Interconnect Scaling - the Real Limiter to High-Performance ULSI”. Proceedings of the International Electron Devices Meeting, pp. 241-244, December 1995.

[15] D. Matzke. “Will Physical Scalability Sabotage Performance Gains?” Computer Magazine, Vol. 30, No. 9, pp 37-39.

[16] R. Canal et al. “A Cost-Effective Clustered Architecture”. Proceedings of the International Conference on Parallel Architectures and Compilation Techniques, 1999.

[17] K.-I. Farkas et al. “The Multicluster Architecture: Reducing Cycle Time through Partitioning”. Proceedings of the International Symposium on Microarchitecture, 2000.

[18] P. Chaparro et al. “Distributing the Frontend for Temperature Reduction”. Proceedings of the eleventh International Symposium on High Performance Computer Architecture, February 2005.

[19] P. Chaparro et al. “Thermal-Aware Clustered Microarchitectures”. Proceedings of the 22nd International Conference on Computer Design, October 2004.

[20] P. Chaparro et al. “Thermal-Effective Clustered Microarchitectures”. Proceedings of the First Workshop of Temperature-Aware Computer Systems at ISCA 04.

[21] G. Magklis et al. “Frontend Frequency-Voltage Adaptation for Optimal Energy-Delay^2”. Proceedings of the 22nd International Conference on Computer Design, October 2004.

[22] J.-M. Parcerisa et al. “Efficient Interconnects for Clustered Microarchitectures”. Proceedings of the International Conference on Parallel Architectures and Compilation Techniques, 2002.

[23] T. J. Chaney and C. E. Molnar. “Anomalous Behavior of Synchronizer and Arbiter Circuits”. IEEE Transactions on Computers, C-22(4), April 1973.

[24] T. Chelcea and S. M. Nowick. “Robust Interfaces for Mixed-Timing Systems with Application to Latency-Insensitive Protocols”. Proceedings of the 38th Design Automation Conference, June 2001.

[25] M. Nyström and A. J. Martin. “Crossing the Synchronous-Asynchronous Divide”. Proceedings of the Workshop on Complexity-Effective Design, May 2002.

[26] A. E. Sjogern and C. J. Myers. “Interfacing Synchronous and Asynchronous Modules within a High-Speed Pipeline”. Proceedings of the 17th Conference on Advanced Research in VLSI, Sept. 1997.

[27] G. Semeraro et al. “Hiding Synchronization Delays in a GALS Processor Microarchitecture”. Proceedings of the 10th International Symposium on Asynchronous Circuits and Systems, April 2004.

[28] Q. Wu et al. “Voltage and Frequency Control with Adaptive Reaction Time in Multiple-Clock-Domain Processors”. Proceedings of the 11th International Symposium on High-Performance Computer Architecture, February 2005.

[29] S. Heo et al. “Reducing Power Density through Activity Migration”. Proceedings of the International Symposium on Low Power Electronics and Design, 2003.

[30] K. Skadron et al. “Control-Theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management”. Proceedings of the International Symposium on High Performance Computing, 2002.

[31] P. Shivakumar and N. P. Jouppi. “CACTI 3.0: An Integrated Cache Timing, Power and Area Model”. WRL Research Report 2001/2.

[32] B. Sinharoy. “POWER5 Architecture and Systems”. Keynote presentation, International Symposium on High Performance Computer Architecture, February 2004.

[33] Intel Corporation Intel® Pentium ® 4 Processor in the 423-pin Package Thermal Solution Functional Specification http://www.intel.com/design/pentium4/guides/249204.htm.

[34] M. Huang et al. “A Framework for Dynamic Energy Efficiency and Temperature Management”. Proceedings of the International Symposium on Microarchitecture, pp. 202-213, 2000.

[35] D. Brooks et al. “Wattch: A Framework for Architectural-Level Power Analysis and Optimizations”, in Proceedings of the 27th International Symposium on Computer Architecture, pp. 83-94, 2000.

[36] C. H. Lim et al. “A Thermal-Aware Superscalar Microprocessor”. Proceedings. International Symposium on Quality Electronic Design, 18-21, March 2002.

[37] J. Donald and M. Martinosi. “Temperature-Aware Design Issues for SMT and CMP Architectures”. WCED Workshop at ISCA-31, June 2004.

[38] S. Ghiasi and D. Grunwald. “Design Choices for Thermal Control in Dual-Core Processors”. WCED Workshop at ISCA-31, June 2004.

[39] E. Rotem et al. “Analysis of Thermal Monitor Features of the Intel Pentium M Processor”, TACS Workshop at ISCA-31, June 2004.

[40] G. Semeraro et al. “Energy Efficient Processor Design Using Multiple Clock Domains with Dynamic Voltage and Frequency Scaling”. In Proceedings of the 8th International Symposium on High-Performance Computer Architecture. February 2002.

[41] G. Semeraro et al. “Dynamic Frequency and Voltage Control for a Multiple Clock Domain Microarchitecture”. In Proceedings of the 35th Annual International Symposium on Microarchitecture, November 2002.

[42] Q. Wu et al. “Formal Online Methods for Voltage/Frequency Control in Multiple Clock Domain Microprocessors”. In Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems, October 2004.