Thermal-Effective Clustered Microarchitectures

Pedro Chaparro
IBRC – Intel Labs - UPC
pedro.chaparro.monferrer@intel.com

José González
IBRC – Intel Labs - UPC
pepe.gonzalez@intel.com

Antonio González
IBRC – Intel Labs - UPC
antoniox.gonzalez@intel.com

Abstract

As frequencies and feature size scale faster than operating voltages, power density is increasing in every processor generation. Along with that, leakage (highly dependent on temperature) has become an important source of power. Due to the non uniformity of on-chip power density, localized hot spots may create transient high temperature in a restricted area of the chip. These temperatures are source of errors and reduce chip reliability. This paper evaluates clustered architectures as an effective way to distribute power across the chip in order to reduce chip temperature. The proposed quad-cluster architecture reduces 33% peak temperature and 12% average. Along with this, "cluster-hopping" decreases temperature in the chip because of disabling some of the clustered backends during a period of time: peak temperatures are reduced 37% and average temperature of the processor 14% with an extra penalty of 3%.

1. Introduction

Power dissipation is one of the major hurdles in the design of next-generation microarchitectures. Power density is increasing in each generation due to the fact that feature size and frequency are scaling faster than operating voltage. Power density directly translates into heat, and this heat must be removed from the processor die in order to keep the silicon temperature below a certain limit. In fact, the cost of removing heat is increasing at the same rate as power density. This increase is affecting the processor design in many different ways. For instance, the cooling system of a processor is targeted to support a peak temperature, even though the processor spends most of the time running at much lower temperatures. The cost of the cooling system has been quantified in the order of 1-3$ or more per Watt when the average power exceeds 40 Watts [4][10], which represents an important cost.

In order to reduce dynamic power dissipation, chip designers have relied on scaling down the supply voltage. To counteract the negative effect of a lower supply voltage on gate delay, the threshold voltage is also scaled down along with the supply voltage. However, lowering threshold voltage has a significant impact on leakage current due to the highly strong relationship between them. In fact, it is expected that within a few process generations the contribution of leakage power to the total power will be comparable to the contribution of dynamic power [4][8].

On the other hand, wire delays scale much slower than gate delays [1][3][16] and will become a serious obstacle to the scalability of superscalar processors. Clustered microarchitectures are an effective paradigm to deal with the problem of wire delays and complexity by means of partitioning some of the processor resources [7][9], as for instance the processor backend, and attempting to maximize local (and fast) communications and reduce global (and slow) communications.

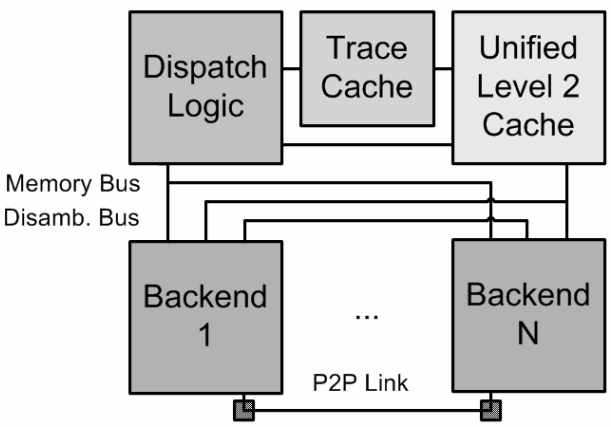
This paper studies the benefit of clustered microarchitecture from the thermal point of view when they are compared with monolithic architectures. Besides, we introduce an architectural modification, cluster hopping, with the aim of reducing processor's temperature. Cluster hopping refers to the architectural feature that disables (i.e., V_{dd} is gated-off) some of the clusters during a time interval, in order not to dissipate power and to reduce temperature. The clusters that are V_{dd}-gated rotate every interval to achieve a better activity and, though, heat distribution.

The rest of the paper is as follows: Section 2 describes the processor architecture and the power and thermal models. Section 3 provides baseline temperature results for a monolithic architecture. Section 4 introduces our proposals for study of the thermal effectiveness of clustered microarchitectures and cluster hopping. Section 5 presents the performance (execution time and temperature) results. Section 6 highlights the related work and Section 7 concludes the paper.

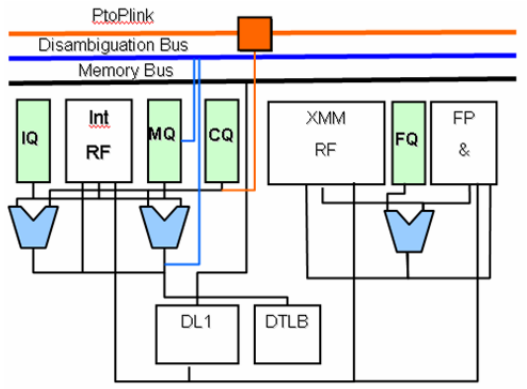
2. Processor Architecture

2.1. Clustered Architecture

This Section briefly describes the baseline clustered microarchitecture.



(a)



(b)

Figure 1. (a) Block diagram of the clustered microarchitecture (b) Backend detail

Figure 1 depicts the block diagram of the clustered microarchitecture. A high-level picture can be seen in Figure 1a, in which the two main parts of the processor are distinguished: the frontend and the clustered backends. The frontend reads IA32 instructions from the UL2, translates them into *uops* and stores them in the Trace Cache, from where they are read, decoded, renamed and steered to any of the backends, according to a steering policy. Figure 1b shows the details of one of the backends (a.k.a. clusters). Each of them has its own register file, integer and floating point issue queues and a memory order buffer along with a data TLB and a first-level data cache.

uOps are first handled by the dispatch logic, where the steering unit decides the destination cluster based on some policy. Once the destination cluster is decided, the logical output register is mapped into a free register belonging to that cluster and the instruction is steered.

After steering, instructions remain in an issue queue until their inputs become available, and then, they are executed and results are written back to the register file.

Special *copy* instructions are generated by the dispatch logic when an instruction requires a register value generated in a cluster other than the one in which it will be executed ([7], [17]). This *copy uop* is dispatched to the cluster generating the value and it is in charge of sending the data through a point-to-point link to the cluster where the consumer resides.

Data caches are distributed and a load can be steered to any cluster. If there is a cache miss, the UL2 is accessed using the memory bus and the line is written in the cache of the cluster where the requesting load resides. Store instructions are steered to a cluster according to the steering policy to compute the effective address, but they allocate a slot in all memory order buffers in order to disambiguate stores from subsequent loads [2]. When the store address is computed, it is sent through the disambiguation bus and copied to all clusters, so disambiguation can be performed locally.

Integer and floating point instructions leave the issue queue after being issued. Store instructions remain in the memory order buffer until commit and loads are stored in the memory order buffer until they are disambiguated. After executed, instructions send a *completed* signal to the reorder buffer and they can be committed once they reach the head of the buffer.

The monolithic implementation considered in this paper is the equivalent to the clustered one but without the need of communication and coherency among clusters (basically the copy instructions and the copy scheduler are avoided). The aggressiveness of the backends depends, basically, on the number of them that are implemented in each particular configuration.

2.2. Power Model

This Section introduces both the dynamic and the leakage power model that we have utilized to carry out the experiments.

The dynamic power model is very similar to those existing in the literature [5]. Basically, an activity counter is associated to each functional block (e.g. register files, data cache, etc) and it is incremented in each block operation. In order to compute the energy, the activity counter is multiplied by its corresponding energy-per-operation value (obtained from the physical design or an analytical model). An aggressive clock gating technique is assumed: unused structures just dissipate 10% of their

peak dynamic power, since a perfect gating is not usually possible [5] (except in the case of UL2 where it is considered a perfect clock gating).

For each functional block of the processor leakage power has been modeled as the average dynamic power multiplied by a factor dependent on the temperature. More precisely, it is assumed that leakage power is going to be roughly 30% of dynamic power at ambient –inside box- temperature (45º [15], [20]). Then, in order to establish a strong dependence between temperature and leakage, this percentage is varied according to the current temperature of the functional block. Figure 2 shows a curve similar to that reported by Zhang et *al* in [21] (generated using BSIM3). Our analysis is adjusted to fit the expected trend for 65nm and reports a leakage equal to 100% of the average dynamic power (i.e., leakage is 50% of total power) at 100ºC. In this analysis, the curve is divided into linear segments of 5 degrees ranging from 45 to 100 degrees (interpolating each interval by a straight line). When the temperature of a block changes, leakage is recomputed getting the proper factor from the corresponding segment.

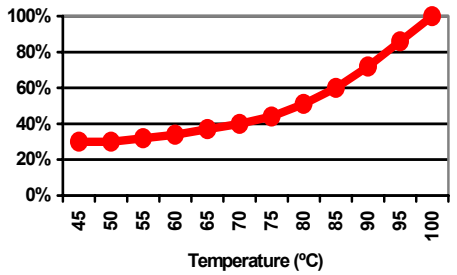


Figure 2. Relation between temperature and leakage power

This approximation of leakage over dynamic power (and not over area) is reasonably good for memory-like structures, which constitute most of the processor's leakage, although it may be a bit inaccurate for functional units. Note that the objective of the leakage estimation presented in this paper is to model the relationship between leakage power and temperature. This model may not be as accurate as other ways of measuring leakage (e.g. HSPICE simulations) but it accomplishes our main requirements.

2.3. Temperature Model

The temperature model is similar to the one by Skadron *et al.* [19][20]. It is based on the duality of the thermal and the electrical phenomena (Table 1). The temperature is estimated using a RC model that represents the system, also known as dynamic compact model (dynamic because it includes thermal capacitors modeling the transient response of the system). At the microarchitectural level, it models heat conduction and the removal of heat in the heat sink.

Table 1. Duality between the electrical and the thermal phenomena

Electrical	Thermal
Voltage (*V*)	Temperature (*K*)
Current (*A*)	Power (*W*)
Resistance ($V / A = \Omega$)	Resistance (K / W)
Capacity ($J / V = F$)	Capacity (J / K)
Time constant $\tau = R \cdot C$ (*s*)	

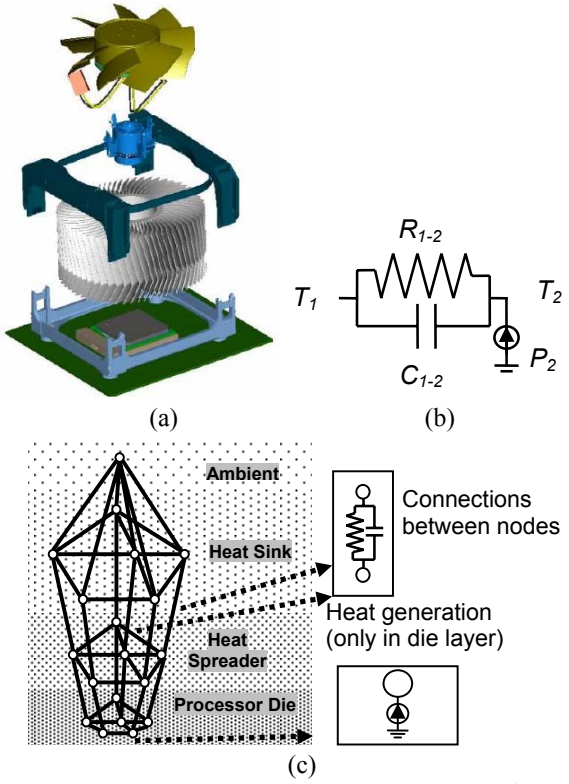


Figure 3. (a) Thermal model (b) Model of heat generation and spreading (c) Layer division of the whole system

In order to build an equivalent RC circuit the system is divided into layers. In our case these layers are: the die, the heat spreader and the heat sink, as shown in Figure 3a. Each one of these layers is divided then into blocks: the functional block division (register file, data caches, instruction queues, etc.) is used for splitting the die, and the rest of the layers are divided accordingly, that is, mapping the division of the die on them (note that this approach is different from the one followed in [19]). Some other mapping schemes were studied and this one was found to be the most accurate among them. Each of

these blocks is represented by a node in the RC circuit. The upper layers division models the thermal heterogeneity of the heat spreader and heat sink.

If two blocks are in contact then an RC connection (shown in Figure 3b) is placed between the nodes representing them. A RC connection consists of a resistor in parallel with a capacitor. The resistor models the opposition to the heat flow and the capacitor models the time component, i.e. how long it takes to reach the steady-state situation. Some connections link the nodes representing the heat sink with a special node for the ambient (the only node whose temperature does not change because of heat flow). Heat generation is modeled using the same duality: a current source for each functional block is placed in the circuit, as shown in Figure 3b).

The thermal properties of the materials and the relative positions and area of contact among blocks are used to estimate the R and C values that model the heat flow. The physical definition of a resistor value is used applying the thermal resistivity of the material (ρ), the distance between the central point of a block and the center of the contact area with the neighbor block and the area of contact itself (different studies on how to estimate resistors and capacitors were carried out in order to find the most accurate approach). For convection, which models the removal of heat through air, the same approach is used but applying the heat transfer coefficient of the ambient (v). The counterpart equation for capacitors is derived by using the same area and distance values and the heat capacitance (γ). At the simulated time scale, capacitors to ambient are useless because the time constant is huge compared to the simulation step, therefore these capacitors are neglected.

Table 2. Equations for the equivalent resistor and capacitor connecting two nodes

	Partial resistor	Partial capacitor
Conduction	$Ri = \rho i \cdot l\, i / A\, i\text{-}j$	$C_i = \gamma_i \cdot l_{\,i} \cdot A_{\,i\text{-}j}$
Convection	$Ri = 1 / (vi \cdot A\, i\text{-}j)$	$Ci = 0$
Total value	$Ri\text{-}j = Ri + Rj$	$1 / C_{\,i\text{-}j} = 1 / C_i + 1 / C_j$

The RC circuit behavior is controlled by the equations of each one of its components. For each thermal node, Kirchoff's law is applied to derive for every RC pair connection a differential equation that involves heat flow, the thermal resistor and the thermal capacitor. At simulation time the discrete version of the equation is used:

$$\Delta T_i(t) - \Delta T_j(t) = \\ = P_{i \to j}(t) \cdot \Delta t / C_{i\text{-}j} - (T_i(t) - T_j(t)) \cdot \Delta t / (R_{i\text{-}j} \cdot C_{i\text{-}j})$$

Each one of the unknowns (increment of temperature in each node -$\Delta Ti(t)$- and power flow through each connection -$P_{i \to j}(t)$-) is linked to an equation so the number of unknowns is equal to the number of equations. Given the state of the circuit and the power dissipated in the time interval, the resulting linear system of equations is solved, which generates the increment of temperature for each one of the nodes.

3. Evaluation of the Microarchitecture

3.1. Experimental Framework

Experiments have been conducted using an execution-driven simulator that runs IA32 binaries. The processor can fetch, dispatch and commit up to 8 *uops* per cycle. Table 3 summarizes the main parameters of the baseline monolithic architecture.

We have selected sixteen SPEC2000 applications for the evaluation process. Each execution trace (from the *test* input set) is divided in 10 equal-size slices (i.e., slices of different applications have different size) and the fourth of them is selected to be run in the simulator (the whole slice or up to 200 millions of instructions). This limit was reached by *ammp, art, bzip2, crafty, eon, gzip, mesa, mgrid, parser, swim, vortex* and *wupwise* but *applu, gcc, twolf* and *vpr* were run for 115, 187, 64 and 150 millions of instructions respectively.

Table 3. Processor configuration

Frontend	
Trace cache/Fetch	32Kuops, 4-way, 4 cycle fetch-to-dispatch latency
Decode, rename and steer	8 cycles (regardless of the destination cluster)
UL2	2 MB/8-way, 12 cycle hit, 500+ miss
Communi-cations	2 memory buses, 2 disambiguation buses, 4-cycle latency + 1-cycle arbiter, 2 bidirectional p2p link (1 cycle per hop; 2 from side to side of the chip)
Each backend	
Queues	80-entry IQueue 4 inst/cycle, 80-entry FPQueue 3 inst/cycle, 384-entry MemQueue 4inst/cycle, 10 cycle dispatch latency; 20 entries per prescheduler queue (integer, FP, memory and copies)
Register file	544 int. registers (10 read and 8 write ports) and 544 FP registers (10 read and 8 write ports)
Data cache	64 KB/2-way, 3 cycle hit, 2 read and 2 write ports, write through

As far as the thermal model is concerned, at the beginning of the simulation we assume that the processor has already been running for a long time dissipating its nominal average dynamic power and the leakage power at 80ºC. In this way, simulations are started with the processor already warmed. Then, during normal execution, every 10 million of cycles temperature is updated using the per-block dissipated power.

Figure 4b shows the layout of the processor and Figure 4a and Figure 4c detail, respectively, the frontend and the backend area for the monolithic superscalar processor; the 2MB UL2 completes the processor. We assume a processor designed at 65nm, running at 10GHz with a V_{dd} of 1.1V with a total area of 56.34mm^2, which, again, make it feasible to be included in a bigger CMP configuration (i.e. multiple processors cores on the same chip). For the sake of simplicity a configuration with a single core is analyzed. Areas were computed using and enhanced version of Cacti ([23]) for cache-like structures, and scaling down rest of 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 Pentium® 4 processors [13]). On top of it there is a copper heat sink of 7x8.3x4.11cm ([13]).

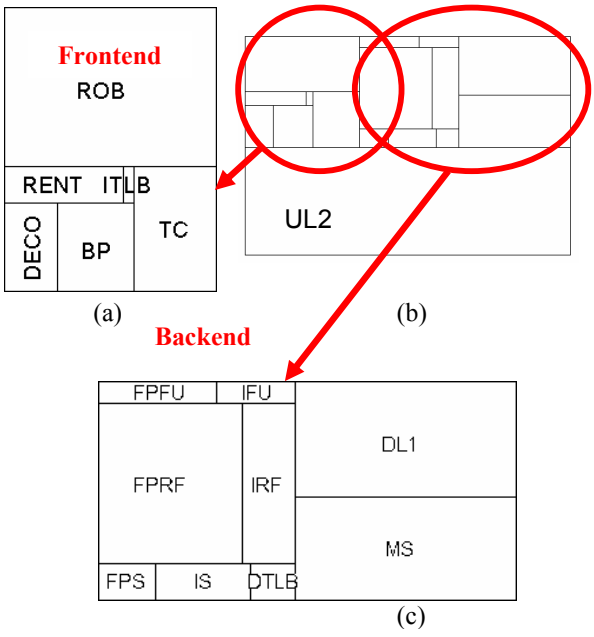


Figure 4. (a) Frontend layout detail, (b) monolithic processor layout and (c) backend detail

3.2. Results for the Monolithic Processor

This Section shows the initial results for the baseline architecture. The temperatures obtained for the simulated benchmarks are depicted, averaged, in Figure 5 (the Figure shows temperatures with respect to 45 ºC). Four different metrics are shown for four different scopes ("sets") of the processor (UL2, frontend area, backend area and global processor):

- *LocalMax*: The average temperature of a set of blocks (i. e. backend area) is computed as the weighted average (by area) temperature of each one of the functional units that are included in that set. This means that, for instance, having a very high temperature in a block in the backend does not mean that the average temperature of the backend itself is high if the rest of the blocks are cold. Its a general metric for reducing average temperature.
- *AbsMax*: This metric represents the maximum peak temperature in any of the blocks included in a set.
- *Average*: The average temperature of the set (over time). This metric is important because of the impact of temperature in leakage. Reducing the average temperature of the chip helps reducing leakage.
- *AverageMax*: The average over time of the maximum temperatures of any block inside a set accounts for the evolution of the highest temperature in time. With this metric we can detect, in opposition to a reduction in the absolute peak, a reduction along time of the peak temperature. These last three are the main metrics.

Looking at the Figure, we observe that the backend (i.e. instruction queues, schedulers, register files, etc) is the area that determines peak temperatures. Therefore, clustering that area, which reduces power density, seems an interesting approach to control temperature.

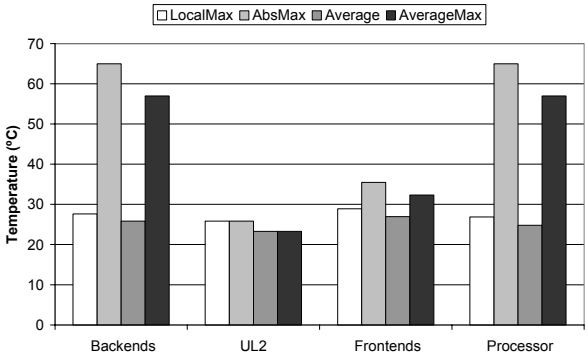


Figure 5. Baseline temperature values (over ambient in-box temperature, 45 ºC)

4. Thermal-Effective Clustered Organizations

This section presents two different proposals to deal with the high temperatures that arise during runtime processor execution in monolithic configurations.

- Clustered processors: By means of partitioning processor's resources a better distribution of power and activity may be accomplished reducing temperature.
- Cluster hopping: When clustering resources, by alternatively gating some of the clusters of the processor during certain periods of time power and power density can be reduced helping cooling the chip.

4.1. Clustered Processors

Table 4. Configuration of each backend

	Bi-cluster	Quad-cluster
Queues	40-entry IQueue, 2 inst/cycle, 40-entry FPQueue 2 inst/cycle, 40-entry CopyQueue 2inst/cycle , 192-entry MemQueue 2inst/cycle, 10 cycle dispatch latency; 20 entries per prescheduler queue	40-entry IQueue 1 inst/cycle, 40-entry FPQueue 1 inst/cycle, 40-entry CopyQueue 1inst/cycle , 96-entry MemQueue 1inst/cycle, 10 cycle dispatch latency; 20 entries per prescheduler queue
Register file	272 int. registers (8 read and 4 write ports) and 272 FP registers (7 read and 4 write ports)	160 int. registers (6 read and 3 write ports) and 160 FP registers (5 read and 3 write ports)
Data cache	32 KB/2-way, 2 cycle hit, 2 read and 2 write ports, write update	16 KB/2-way, 1 cycle hit, 1 read port, 1 write port, write update

In addition to the well-known benefits of clustered architectures [7][9], the inherent distribution of resources and activity achieved by such organizations may help to better distribute heat.

Figure 6 shows configuration and layout details for the two different clustered architectures studied in this paper. The areas are 60.19 mm2 for the bi-clustered and 58.42 mm2 for the quad-cluster architecture (again small enough to be part of a CMP configuration). In the same figure, the details for the backend are given. The frontend layout and configuration as well as the thermal solution remain unchanged from the monolithic processor.

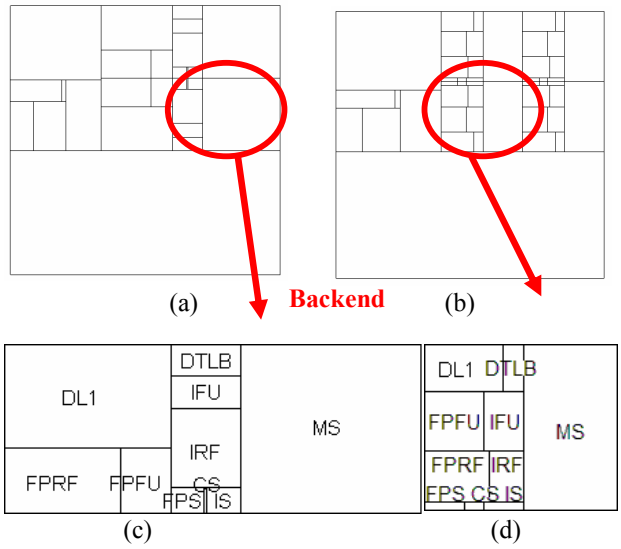


Figure 6. Layout of the bi-cluster processor (a) and its backend (c) and quad-cluster processor (b) with its backend (d)

4.2. Cluster Hopping

Asanović *et al.* [11] study the impact on power density of activity migration among replicated units. In opposition, this paper focuses on the reduction of temperature. We analyzed hopping among clusters. In a clustered configuration, when hopping, a subset of the clustered backends is V_{dd} -gated during certain periods of time to help reducing the power dissipation of the chip. In addition, the power density is also reduced because of the rotation of the unit gated. We evaluated the impact on temperature (and not on power) of cluster hopping in clustered architectures.

Figure 7 shows the different alternatives. The block distribution is the same as in the normal quad-cluster architecture. Gray painted clusters represent those clusters that are disabled. At the end of the interval (10 millions of cycles), the configuration switches to the next step in the cycle. Changing active clusters requires a stream of *copy* instructions to be dispatched in order to copy all registers just mapped in the clusters being disabled (i.e. that are not present in any other cluster). Once the cluster is empty, it can be V_{dd} –gated safely. When a cluster is woke up local data cache and data TLB are empty (data caches are write-through, so next memory level has always an up-to-date copy). Since the periods of time between hops are large enough, the performance impact of copy streams is negligible.

Two different configurations are tested: in *HOP3* (Figure 7a) only one out of the four clusters is gated. In

HOP2 (Figure 7b) two clusters are gated. Moreover, in the *HOP2* configuration the frontend is located in the middle of the layout. In this layout copy instructions that go from clusters placed in both sides of the frontend require an extra cycle. Note that this will just happen for those copies generated at reconfiguration time, since with *HOP2* just clusters placed in the same side are always active.

This layout could have been also probed along with the *HOP3* scheme, but we believe that it makes no sense since having three active clusters will cause a lot of traffic from one side of the frontend to the other side, increasing the latency of communications and thus its impact on the final performance.

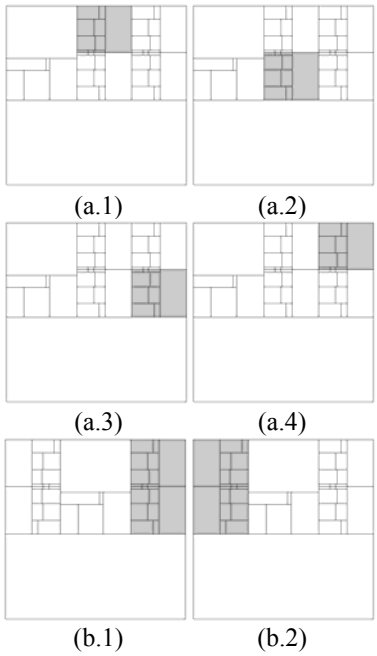


Figure 7. Cluster-hopping alternatives analyzed: (a) *HOP3* (b) *HOP2*

5. Evaluation

This section presents the results obtained when comparing the baseline with the different proposals.

We evaluate the benefits associated to both clustered architectures and cluster hopping. Any mechanism such fetch toggling, throttling or voltage/frequency scaling are orthogonal to our schemes and can be applied on top of them so none of them have been evaluated. Also, the usage of different physical parameters (threshold voltages, frequency) has not been compared because can be applied on both monolithic and clustered architectures (the less complexness of clustered architectures may permit, even, higher frequency).

5.1. Clustered Organizations

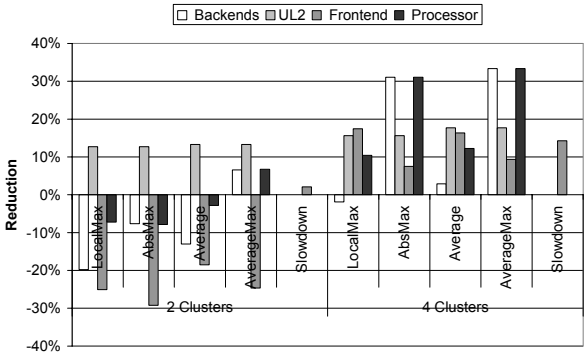


Figure 8. Performance (slowdown and temperature reductions) of the clustered architectures

Figure 8 shows the runtime results (averaged for all sixteen benchmarks) obtained for the evaluated clustered architectures. It can be seen that architectures with 2 backend clusters are not as effective as a quad-cluster architecture. This is due to the fact that, when the resources are partitioned in just two clusters they are still complex and large enough not to show benefits in temperature. In addition, the explored layout keeps critical structures (such the schedulers) together so for heat spreading looked like a monolithic scheduler. Actually, temperature is increased due to a worse power density: activity is not reduced at the same rate as area (halving the capacity of some units does not necessarily mean halving their area).

Only the L2 cache has visibly reduced its temperature. On the other hand the performance penalty (measured in instructions per cycle) paid by a bi-cluster processor is small (2%) compared to the quad-cluster (14%). The quad-cluster is clearly reducing processor temperature except for the case of the average backend highest (2%). In fact, the highest temperature reported inside the backends is reduced by 31% and the average through the whole simulation is also reduced by 37%. On the other hand both UL2 and frontend areas have also their temperature decreased. In general there is a reduction of 33% in processor's peak temperature and 12% average temperature.

5.2. Cluster Hopping

Figure 9 gives details of the results for both cluster-hopping alternatives proposed in this paper. Analyzing *HOP3*, we can see a big impact on the temperature of the backends: peak temperature is reduced 31% and the average over time is reduced (37%). UL2 and frontend

temperature is reduced between 1-2% with respect to the quad-cluster configuration. Summarizing, *HOP3* reduces processor peak temperature by 37% (as the average of all applications studied in this work) and the average temperature by 14%, with a slowdown of 17% over the baseline (only 3% more than the standard quad-cluster architecture). When 2 clusters are gated together (*HOP2* configuration) the average performance penalty increases quite a lot (29%). Temperature is also decreased, which suggests that this approach can be applied to effectively reduce temperature right before the processor reaches a thermal crisis, since the penalty paid is lower than that of throttling. Nevertheless, this is an opened issue and it is part of our future work.

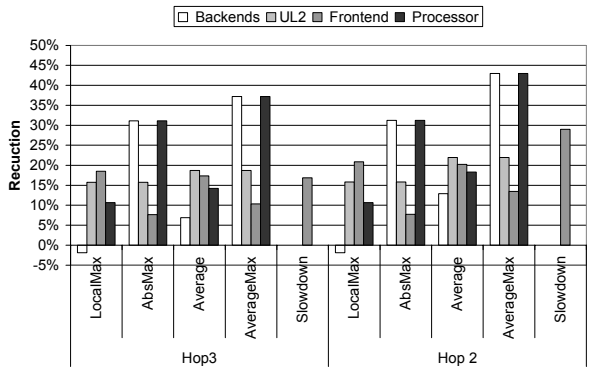


Figure 9. Performance stats for the cluster-hopping alternatives

6. Related Work

Controlling temperature through microarchitectural techniques is a fairly new area. Huang *et al* [12] 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, sub-banked data cache, among others. Brooks and Martonosi [6] propose a set of control techniques evaluated on top of different triggering mechanisms with the aim of reducing thermal emergencies. They use the average power in an interval as a proxy of temperature. Skadron *et al.* [19][20] propose a thermal simulator based on the duality between heat transfer and the electrical phenomena. Several techniques are evaluated to control peak temperature and reduce thermal emergencies: PI controllers, frequency scaling, fetch toggling and replication of the register file. Cai *et al* [14] propose a secondary ultra-low power pipeline that is used when a given temperature threshold is exceeded. Asanović *et al.* [11] study the impact of activity migration among replicated units on power density.

The main contribution of this paper with respect to previous proposals is that, to the best of our knowledge, this is the first work that analyzes the thermal behavior clustered architectures, presenting this microarchitecture paradigm as an effective way of reducing both average and peak temperature.

7. Conclusions

The distribution of processor's resources through clustered architectures helps to spread the activity and energy dissipation through chip's area. In order to achieve temperature reductions, partitioning processor's backend into 2 clusters is not enough for reducing temperature, but it is even increased. Four clusters are enough to see better results: ignoring some other benefits of clustered architectures, the distribution analyzed in this paper is able to reduce processor peak temperature 33 % (average of all applications analyzed), and 12% average temperature with a performance penalty of 14%. Improving the quad-cluster architecture with a hopping scheme, peak temperature is reduced 37% and average temperature of the processor 14% with an extra penalty of 3%.

8. References

[1] V. Agarwal, M.S. Hrishikesh, S.W. Keckler and D. Burger. "Clock Rate versus IPC: the End of the Road for Conventional Microarchitectures". In Proceedings of the 27th International Symposium on Computer Architecture, 2000.

[2] R. Balasubramonian, S. Dwarkadas and D.H. Albonesi. "Dynamically Managing the Communication Parallelism Trade-off in Future Clustered Processors. ". Proceedings of the International Symposium on Computer Architecture, 2003.

[3] M. Bohr. "Interconnect Scaling - the Real Limiter to High-Performance ULSI". In Proceedings of the International Electron Devices Meeting, pp. 241-244, Dec. 1995.

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

[5] D. Brooks, V. Tiwari V. and M. Martonosi. "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations", in Proceedings of the 27th International Symposium on Computer Architecture, pp. 83-94, 2000

[6] D. Brooks and M. Martonosi. "Dynamic Thermal Management for High-Performance Microprocessors". Proceedings of the International Symposium on High-Performance Computing, 2001.

[7] R. Canal, J.M. Parcerisa and A. González. "A Cost-Effective Clustered Architecture" Proceedings of the International Conference on Parallel Architectures and Compilation Techniques, 1999.

[8] 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.

[9] K. Farkas, P. Chow, N. Jouppi and Z. Vranesic. “The Multicluster Architecture: Reducing Cycle Time through Partitioning”. Proceedings of the International Symposium on Microarchitecture, 2000.

[10] S. Gunther, F. Binns, D. M. Carmean and J.C. Hall. “Managing the Impact of Increasing Microprocessor Power Consumption”. Intel Technology Journal, Q1, 2001.

[11] S. Heo, K. Barr, K. Asanović “Reducing power density through activity migration” Proceedings of the 2003 International Symposium on Low Power Electronics and Design, 2003.

[12] M. Huang, J. Renau, S-M. Yoo and J. Torrellas. “A Framework for Dynamic Energy Efficiency and Temperature Management”. Proceedings of the International Symposium on Microarchitecture, , pp. 202-213, 2000

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

[14] C. H. Lim, W. R. Daasch, G. Cai, “A Thermal-Aware Superscalar Microprocessor” Quality Electronic Design, 2002. Proceedings. International Symposium on , 18-21 March 2002

[15] R. Majan “Thermal management of CPUs: A perspective on trends, needs and opportunities”, Oct. 2002. Keynote presentation, THERMINIC-8.

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

[17] J.-M. Parcerisa, J. Sahuquillo, A. González, J. Duato "Efficient Interconnects for Clustered Microarchitectures" Proc. of the Int. Conf. on Parallel Architectures and Compilation Techniques (PACT 2002).

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

[19] K.Skadron, T. Abdelzaher and M. Stan. “Control-Theoretic Techniques and Thermal-RC Modelling for Accurate and Localized Dynamic Thermal Management”. Proceedings of the International Symposium on High Performance Computing, 2002.

[20] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. “Temperature-Aware Microarchitecture”. In Proceedings of the 30th Annual International Symposium on Computer Architecture, Apr. 2003.

[21] Y. Zhang, D. Parikh, K. Sankaranarayanan, K. Skadron and M. Stan. “Hotleakage: A Temperature-Aware Model of Subthreshold and Gate Leakage for Architects”. Technical Report CS-2003-05, University of Virginia Department of Computer Science, Mar. 2003. 34.