Dynamic Cluster Resizing

José González and Antonio González
Intel Barcelona Research Center
Intel Labs Barcelona – Universitat Politècnica de Catalunya
{pepe.gonzalez,antoniox.gonzalez}@intel.com

Abstract

Processor resources required for an effective execution of an application vary across different sections. We propose to take advantage of clustering to turn-off resources that do not contribute to improve performance. First, we present a simple hardware scheme to dynamically compute the energy consumed by each processor block and the energy-delay^2 product for a given interval of time. This scheme is used to compute the effectiveness of the current configuration in terms of energy-delay^2 and evaluate the benefits of increasing/decreasing the number of active issue queues. Performance evaluation shows an average energy-delay^2 product improvement of 18%, and up to 50% for some applications, in a quad-cluster architecture.

1. Introduction

Clustered microarchitectures are an effective approach to solve the problem of the impact of wire delays and the increase of the processor complexity [1][6], by means of partitioning some of (or all) the processor resources [5]. The impact of wire delays is reduced by keeping local communications inside a cluster fast. Clustering also provides for a simple approach to turning-off resources that do not contribute to improve performance when the requirements of applications vary across different sections of the code.

This work focuses on designing power-efficient clustered microarchitectures. We first propose a technique to dynamically compute the energy consumed by each issue queue during a particular time interval. This estimation is used to compute the effectiveness of the current configuration (i.e. the current number active issue queues), and decide whether the number of issue queues must be decreased (by gating off a queue in a particular cluster) or increased. So far, resizing schemes for low-power rely on estimating the exploited ILP, resource utilization or idle time to take the resizing decisions. The novelty of this work is the utilization of an estimated energy-delay^2 product (ED^{2}P) metric as the input to the resizing schemes.

2. Processor Architecture

Figure 1 depicts the block diagram of the processor. The front-end includes a unified L2 cache, a Trace Cache

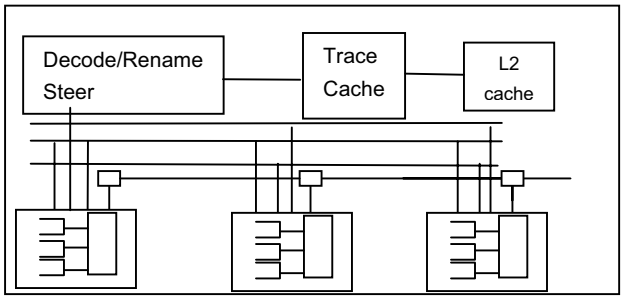

Figure 1. Block diagram of the clustered processor.

that acts as an Instruction cache for micro-operations, in a similar way to the Intel® Pentium® 4 processor and the logic needed to translate macroinstructions into micro-operations in the case of a Trace Cache miss. After the fetch stage, micro-operations are decoded, registers are renamed and the steering unit is in charge of sending the instruction to the appropriate cluster according to the implemented policy. Each cluster has a local register file and register values are not replicated unless needed in more than one cluster. Special *copy* instructions are generated at the steer stage to communicate register values among clusters [5].

Each cluster has its own register files, data cache, data TLB, functional units and four different issue queues: Integer queue, Floating Point queue, Memory queue and Copy queue.

3. Dynamic Energy Estimation

Figure 2 shows the block diagram of the mechanism that we propose to measure the energy consumed by a functional block. We assume that each block has its own activity counter (AC), usually present for performance monitoring. The second component is a constant value, *Energy per Access Register* (EAR), that is set up at design time and represents the energy consumed per operation of that particular block. The activity counter will be updated every cycle according to the number of accesses to that particular block. Then, every certain interval, the total energy consumed by the block is computed by multiplying the activity counter by the *EAR* and it is stored in the *Energy Consumed Register.*

Proceedings of the 21st International Conference on Computer Design (ICCD'03)
1063-6404/03 $ 17.00 © 2003 IEEE

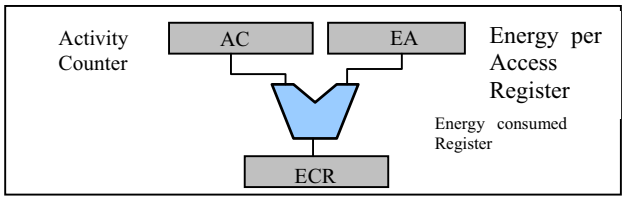

Figure 2. Diagram of Dynamic Energy Estimation

In this work we propose to use Dynamic Energy Estimation to compute Energy-Delay^2 Product (ED^2P for short), since dynamic power dissipation is roughly proportional to V_{dd}^3. Thus ED^2P provides a voltage-independent metric suitable for high-end microprocessors.

4. Dynamic Cluster Resizing

Processor resources are not fully utilized during the execution of one application [2]. We propose a new clustered microarchitecture in which the number of Active Issue Queues (AIQs for short) devoted to each type of instructions is dynamically adapted with the objective of minimizing ED^2P. For each individual IQ, the ED^2P is dynamically computed (using all ECRs associated to that IQ) and, according to its value compared with that obtained in previous intervals, the active number of IQs is changed by enabling or disabling IQs.

4.1. Single Interval Scheme

This scheme requires a *direction bit* associated to each IQ type. This bit indicates the resizing direction of the last interval. At the end of each interval, the ED^2P is compared with that of the previous interval. In order to perform a correct comparison, the delay in the current interval is scaled by dividing the number of instructions executed in the previous interval by the IPC of the current interval, so that the amount of work (instructions) considered for both statistics is the same. If ED^2P has decreased, the scheme is in the right direction, and the number of AIQ is updated according to the *direction bit*. Otherwise, the last interval has been less "ED^2P effective" than the previous one. Thus, the direction bit is reversed and the number of active IQs is updated accordingly.

4.2. Double Interval Scheme

The main drawback of the previous scheme is that the number of active IQs is changed in each interval. This may hurt performance since communication increases every time the number of active queues changes. We propose a second scheme that does not change the configuration for a large interval of time, and then, it tries two alternative configurations for shorts intervals, at that point the ED^2 products obtained are compared. Large interval lasts 256 K cycles and short interval 16 K cycles.

In particular, assume that the current number of AIQ is *N*. The processor runs with *N* AIQs for the large interval. Then it runs for a short interval with *N-1* AIQs and for another short interval with *N+1* AIQs. At this point the scaled ED^2P for the three configurations are compared. And the one with the lowest ED^2P is chosen for the next interval.

Trace Cache	16K uops
Branch Predictor	Hybrid
Decode/Commit Width	4
Unified 2nd Level cache	1 MB
Pipeline length	22+ stages
Cluster Information: 4 homogeneous clusters	
Register File	128 x 2
Integer issue queue size/ issue width	16 / 1
FP issue queue size/issue width	16/1
Mem issue queue / issue width	96/1
Data cache	8 KB/ DM/2 read/1 write
Data TLB	128 entries/FA

Table 1. Processor Parameters

5. Performance Evaluation

Table 1 shows the main characteristics of the simulated architecture. Performance metrics were obtained for 14 applications. from Spec2000 and a *tpc* trace. Applications are run for 200M instructions after initializations.

Figure 3 shows the performance of both schemes proposed in this work (S-Int stands for Single-Interval and D-Int stands for Double-Interval). First, it can be seen that both schemes provide similar results in terms of execution time, with an average slowdown of 3%. However power savings are quite different. D-Int achieves and average power reduction of 22% whereas S-Int provides 15% of reduction.

Figure 3 (d) shows the percentage of time that each IQ is active for *ammp* and *gzip*. For instance, for *gzip* the Integer IQ of Cluster 0[1] is always active, the one in Cluster 1 is active 86% of the time, and the one in Cluster 2 is active 60% of the time.Looking at Figure 3(a) we observe that the speed-up obtained by both schemes for *ammp* is quite similar. However, at Figure 3 (b) we observe that D-Int provides much better power savings. The reason for that is depicted in Figure 3 (d). D-Int consumes fewer resources than S-Int, which means that obtains the same execution time with less active Issue Queues. Same trend is observed for *gzip*: using fewer resources the execution time is the same and more power is saved. Regarding ED^2P results, all applications but

[1] Cluster 0 has all issue queues always active.

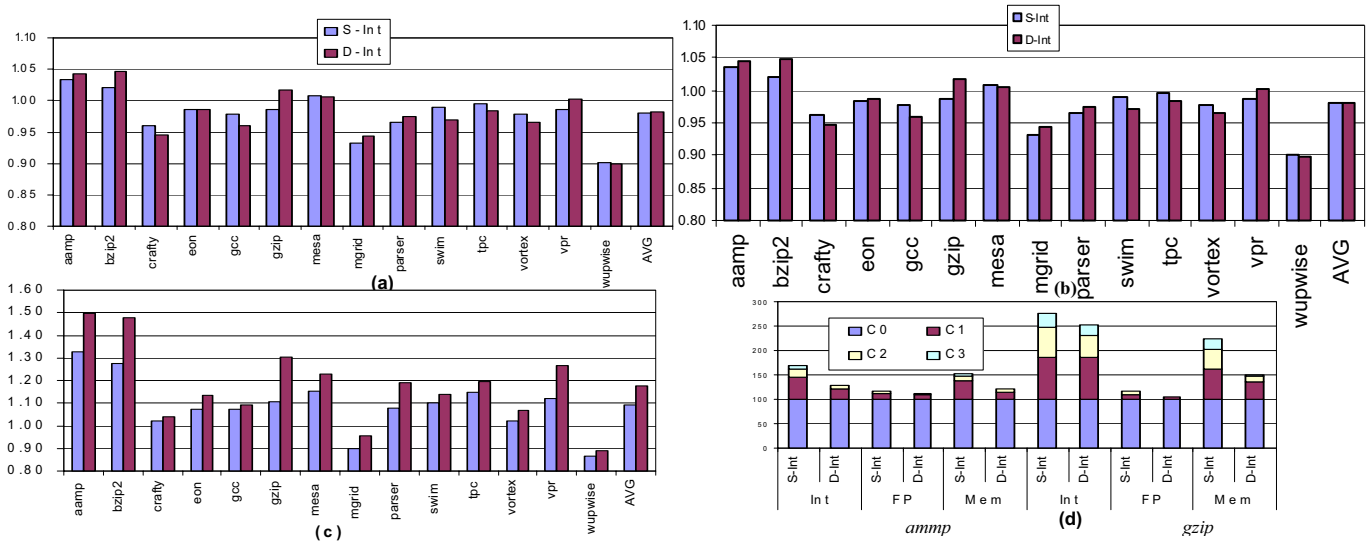

Figure 3. (a) Speed-up (b) Power savings and (c) ED2P improvement of the proposed schemes. (d) Resource utilization for ammp and gzip. Performance results are compared to the baseline architecture with 4 clusters

mgrid experience an improvement. Comparing both schemes, D-Int achieve an average ED^2P improvement of 18%, whereas S-Int obtains 9% improvement. *ammp* and *bzip* obtain up to 50% improvement in ED^2P.

6. Related Work

There have been many proposals regarding cluster architectures ([5] among others). The base architecture presented in this work borrows some features of some of them, trying to build the best possible baseline. There have been a plethora or microarchitectural proposals to control power dissipation. Seng *et al* [9] propose slower functional unit to execute non-critical instructions. Recent studies [4][7] focus on dynamically adapting the size of the issue queue to reduce its power consumption. These schemes usually analyze the activity of the queue and re-size it in the case that there are useless entries. Some recent works propose dynamic frequency and voltage scaling for multi-clock domain architectures as a method to reduce power dissipation [8].

7. Conclusions

This work presents a novel ED^2P-effective clustered microarchitecture. We first propose a mechanism to dynamically compute the Energy-Delay^2 product using the activity counters already included in most current processors, and some energy constants set up at design time. Using this estimation we propose to dynamically adapt the number of active issue queues according to the ED^2P computed across different execution intervals. Results show average ED^2P improvements of 18%, and up to 50% for some applications.

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 Architectures", in *Proc. of the Int. Symp. on Computer Architecture*, 2000.

[2] D. Albonesi. "Dynamic IPC/Clock Rate Optimization. *Proc. of the Int. Symp. on Computer Architecture.* pp. 282-292, 1998.

[3] A. Baniasadi, A. Moshovos. "Asymmetric-Frequency Clustering: A Power Aware Back-End for High-Performance Processors". *Proc. Int. Symp. on Low Power Electronics Design*, 2002.

[4] A. Bayuktosunoglu, S.Shuster, D. Brooks, P. Bose, P. Cook and D. Albonesi. "An Adaptive Issue Queue for Reduced Power at High Performance". *Workshop on Power-Aware Computer Systems*, 2000.

[5] R. Canal, J.M. Parcerisa and A. Gonzalez. "A Cost-Effective Clustered Architecture". In *Proc. of the Int. Conf. on Parallel Architectures and Compilation Techniques,* 1999.

[6] S. Palacharla. "Complexity-Effective Superscalar Processors". PH.D. Thesis, Univ. of Wisconsin-Madison, 1998.

[7] D. Ponomarev, G. Kucuk, K. Ghose. "Reducing Power Requirements of Instruction Scheduling through Dynamic Allocation of Multiple Datapath Resources". *Proc of Int. Symp on Microarchitecture*, 2000.

[8] G. Semeraro, D. H. Albonesi, S.G.Dropsho, G. Magklis, S. Dwarkadas and M.L. Scott. "Dynamic Frequency and Voltage Control for a Multiple Clock Domain Microarchitecture". *Proc. Int. Symp. on Microarchitecture*, 2002.

[9] J.S. Seng, E. S. Tune, D.M. Tullsen. "Reducing Power with Dynamic Critical Path Information". *Proc. International Symp. on Microarchitecture,* 2000.

[i] Large interval lasts 256 K cycles and short interval 16 K cycles

[ii] Cluster 0 has all issue queues always active.