CHAPTER 8

CONCLUSIONS

In this chapter, the main conclusions of this thesis are presented, as well as some open-research areas for future work.

8.1 Conclusions

First, in this thesis, several dependence-based dynamic cluster assignment schemes have been proposed for a clustered superscalar processor.

The first main proposal is a family of slice-based, dynamic cluster assignment schemes. This kind of schemes assign the set of instructions involved in a load or store address calculation (Ld/St slice), or in a branch condition calculation (Br slice), to the same cluster, because these instructions are likely to be critical. First, a dynamic scheme based on a previous "load/store slice" compile-time algorithm is proposed, and it is shown that its major weakness is that it often produces poor workload balance. Then, other more efficient new dynamic slice-based schemes are presented which greatly improve the ability to keep the workload balanced among clusters.

All of these algorithms are evaluated for a cost-effective architecture which is a two-cluster architecture that results of extending the FP unit of a conventional superscalar to execute simple integer operations. It is shown that the dynamic load/store slice scheme substantially outperforms the static one. It is also shown that our best slice-based scheme outperforms a state-of-the-art previous dynamic proposal because it generates a significantly lower number of inter-cluster communications. Our results also prove that a cost-effective architecture with our cluster assignment schemes substantially outperforms a conventional one when running the SpecInt95 benchmarks, because it can profitably exploit idle FP resources to speed-up integer programs with minimal hardware support and no ISA change. It is also shown that this cost-effective architecture also outperforms a conventional architecture when running the SpecFP95 benchmarks, even though the FP-cluster datapath is shared by integer and FP instructions.

The second main proposal is a family of new dynamic schemes (referred to as RMB) that assign instructions to clusters in a per-instruction basis, based on prior assignment of the source register producers, the cluster location of the source physical registers, and the workload of clusters. Compared to the slice-based algorithms, this class of schemes have lower implementation cost and are more flexible to keep the workload balanced since they operate at a finer granularity. The proposed RMB schemes follow a primary and secondary criteria that specifically address the goals of minimizing communications and maximizing workload balance respectively. Our proposal includes a precise definition for the workload balance among N clusters, as well as a hardware mechanism to estimate it.

The experiments show that the proposed AR-Priority RMB steering scheme is more effective than all the existing slice-based schemes, and it significantly outperforms the best dynamic schemes previously proposed in the literature, because it achieves the best trade-off between communications and workload balance. It is also shown that this cluster assignment scheme scales better than other RMB proposals, with growing number of clusters and communication latency, because it avoids spreading blindly the instructions among clusters thus producing less communications. Two more conclusions are extracted from our evaluation: on the one hand, a clustered architecture is quite sensitive to the inter-cluster communication latency, which emphasizes the importance of reducing communications with appropriate steering decisions. On the other hand, a clustered architecture with the AR-Priority RMB assignment scheme is hardly sensitive to the cluster interconnect bandwidth because it has a low bandwidth demand, which suggests the feasibility of efficient cluster interconnects based on simple hardware.

The third main proposal of this thesis is to reduce the penalties of wire delays through the prediction of the communicated values. It is motivated by the increasing penalty of wire delays in future microprocessors. The proposed strategy is applied to inter-cluster communications, in a clustered superscalar architecture. The proposal includes a novel cluster assignment scheme (VPB) that avoids data dependence constraints on the assignment algorithm when a value is going to be predicted and there is a potential for improving the workload balance. This algorithm exploits the less dense data dependence graph that results from predicting values to achieve a better workload balance.

It is shown that value prediction removes communications even for previously proposed steering schemes not specially designed to exploit value prediction. However, performance is higher with VPB cluster assignment because it exploits the predictability of values to improve the workload balance. A simple stride value predictor, together with VPB steering, removes on average 50% of the communications, and reduces substantially the workload imbalance, which translates into an average 14% IPC speedup for a four-clustered architecture. In contrast, an identical value predictor achieves only a 3% speedup for a centralized architecture. It is proven that, because of the large penalties of inter-cluster communications, the benefit of breaking dependences with value prediction grows with the number of clusters and the communication latency, so this technique may produce even better improvements in future technologies, as wire delay - and hence communication latency - increases. On the other hand, it is also shown that the performance improvement of value prediction is quite sensitive to misprediction penalty, although it is less sensitive to the predictor table size, for the considered set of benchmarks and table sizes.

The fourth proposal is motivated by the observation that a clustered architecture (with a good steering scheme, such as the AR-Priority RMB) is hardly sensitive to the cluster interconnect bandwidth. Therefore, cluster interconnects can be implemented with simple hardware and with very low complexity impact on the register files, the issue windows and the bypasses, which results in short cycle times and low power consumption. Several cost-effective point-to-point interconnects are proposed, both synchronous (a ring) and partially asynchronous (a ring, a mesh and a torus). Included with these interconnect models are some proposals of possible router implementations that illustrate their feasibility with very simple and low-latency hardware solutions. For instance, for a synchronous ring interconnect, the router hardware is as simple that it only requires five registers and three multiplexers per cluster. In addition, a new topology-aware improvement to the cluster assignment scheme is proposed to reduce the distance (and latency) of inter-cluster communications.

It is shown that the simple point-to-point interconnects, together with our effective steering schemes, achieve much better performance than bus-based interconnects. Besides, the former do not require a centralized arbitration to access the transmission medium. It is also shown that a partially asynchronous ring performs better than the synchronous one at the expense of some additional cost/complexity due to the additional queue required per cluster, although a tiny queue would practically never overflow. Regarding the connectivity and bandwidth, we extract the following two conclusions. First, as expected, interconnects with higher connectivity perform better because they have shorter communication latency. However, point-to-point partially asynchronous interconnects with moderate connectivity/complexity perform close to an idealized crossbar with unlimited bandwidth and all nodes at one-cycle distance: e.g., a four-cluster ring performs within 2% of the ideal, and an eight-cluster torus performs within 4% of the ideal. Second, despite the low hardware requirements of partially asynchronous interconnects, they achieve a performance close (within 1%) to an equivalent idealized interconnect with unlimited bandwidth and number of write ports to the register files.

Finally, a novel design to fully distribute the processor’s front-end is proposed, which extends the advantages of clustering to structures like the I-cache, the branch predictor, the steering logic and the renaming map table. Several techniques are proposed to partition these structures with the goal of reducing their complexity, and avoiding replication. These techniques minimize the wire delay penalties caused by broadcasting recursive dependences in two critical hardware loops: the fetch address generation, and the cluster assignment. First, it is shown that the branch predictor latency, which is in the critical path of the fetch address generation loop, may be reduced by partitioning it into clusters, in such a way that cross-structure wire delays are left out of the critical path and converted to cross-cluster communications that may be smoothly pipelined. Second, it is shown that the latency of a dependence-based steering scheme, which is inherently a serial process that forms a critical hardware loop, may be greatly reduced by partitioning it into clusters. The negative impact of using outdated assignment information is effectively mitigated with an overriding scheme driven by the dependence analysis.

In summary, clustering reduces the latency of these hardware structures and expose wire delays so that they can be dealt with effectively, which results in shorter pipelines and/or faster clock rates and translates into significant performance improvements. Ignoring these performance benefits, and assuming a full clock cycle for any cross-cluster communication, partitioning the front-end produces just a 4% IPC loss for SpecInt95. This result is quite uniform for all benchmarks, and may be broken into three components. About 1.65% IPC loss is caused by the one-cycle increase in branch misprediction latency, which results from adding a stage to broadcast the cluster assignments. About 1% is due to the clustered branch predictor, because of the bubble generated when switching the predictor banks, and the remaining 1.35% is caused by using outdated assignment information in the partitioned steering.

8.2 Open-Research Areas

For a clustered architecture, inter-cluster communication latency is a major source of performance loss. The baseline architecture assumed in this thesis uses a simple explicit copy mechanism to transfer values from one register file to another, and these copy instructions are handled in the same way as regular instructions. Though simple, this mechanism has two important drawbacks: first, it consumes issue queue entries and issue bandwidth; second, copy instructions augment the data dependence graph with an additional node that increases the length of the dependence chain, and may contribute to the length of the critical path of execution. As suggested in section 2.1.3, the first problem may be solved with minimal hardware cost by using separate issue queues and register file read ports for copy instructions. However, the second problem still remains: copy instructions increase the total communication latency by one cycle, which is spent in the wake-up and selection task of the copy instruction itself. Therefore, further research on improved register copy mechanisms with the goal of avoiding the copy issue cycle penalty is ongoing.

Along this thesis we deliberately obviate the question of which is the optimal cluster configuration for a given technology scenario, and a given design goal. Our approach considers configurations with homogeneous clusters, and with a moderate number of clusters (2 or 4) and issue width per cluster (2 or 4). However, other configurations could be explored. On the one hand, whereas increasing the amount of execution resources per cluster has a positive effect on IPC, it also increases the delay of some critical hardware loops, such as the wakeup and select task or the execute and bypass loop. Since both effects have opposite directions, an accurate delay estimation could be useful to study the impact on performance of many design trade-offs between parallelism and complexity. On the other hand, it is ineffective to replicate in all clusters execution resources for complex operations that are seldom used. Thus, exploring configurations with heterogeneous clusters could be an interesting approach.

The design of the cache hierarchy for a clustered architecture is another important open research area. There are two main architectural components that should be studied: L1 caches and disambiguation hardware. Current technology projections anticipate that first level data caches will tend to reduce their size to keep latency low, although it will come at the cost of a higher miss ratio. With clustered architectures, a partitioned cache design may seek to achieve both objectives, a reduced latency and a large cache size. There exist already some proposals to design a clustered L1 data cache. However, there are less work done that focuses on the complexity of the dynamic disambiguation hardware. Although the issues of partitioning the disambiguation logic and data caches are independent of other architectural features, it is likely that both may take advantage of the instruction distribution mechanisms already included in a clustered architecture, so it may be viewed as a further extension of the clustered approach.

This thesis has shown the benefits of value prediction to reduce inter-cluster communications in a clustered architecture. However, our approach focused on a rather simple stride based value prediction scheme. It is likely that even better performance could be achieved with more clever prediction schemes that are both more accurate and more selective, to focus on those values that are more predictable, and have the highest potential for reducing the critical path of execution.