CHAPTER 5

REDUCING WIRE DELAY PENALTY THROUGH VALUE PREDICTION

In this chapter we show how value prediction can be used to avoid the penalty of long wire delays by providing the receiver of the communication with a predicted value and validating the prediction locally, where the value is produced. Only in the case of misprediction, the long wire delay is experienced. This concept is applied to a clustered microarchitecture in order to reduce inter-cluster communications. In addition, the predictability of values allows the dynamic instruction partitioning hardware to have less constraints to optimize the trade-off between communication requirements and workload balance, which is the most critical issue of the partitioning scheme. In particular, we show that the performance of a realistic implementation of a four-cluster architecture may be improved by 14% through a simple value prediction scheme and a new steering logic designed to take advantage of the value predictor.

5.1 Introduction

As discussed in Chapter 1, the increasing impact of global wire delays will become soon a major problem in the design of future microarchitectures because signals that cross a large portion of the die will require multiple cycles to propagate. By predicting the value to communicate, the receiver may proceed without being penalized by the communication delay. The actual communication may occur later on, out of the critical path of execution, or it may be replaced by a simple verification signal, depending on where the verification takes place.

Value prediction has been largely investigated in the context of superscalar processors, and it is not our purpose to design another predictor but to investigate its potential to reduce the penalties of slow communications. Value prediction has a great potential to eliminate data

dependences and to increase program parallelism, but it sometimes does not fulfill the expectations because the misprediction overhead offsets almost completely the performance gains. Therefore, in this chapter we propose to revisit the value prediction technique in the context of long communication latencies caused by wire delays.

As a sample application, we will show how value prediction helps reducing inter-cluster communication penalties in a clustered architecture, and enables a new source of performance improvements. In conventional value prediction approaches, instructions are dispatched speculatively with a predicted source register only if its actual value is still unavailable. Our proposal differs from this because an instruction may be dispatched speculatively with a predicted source register even though the computed value is already available, if this value is in a remote cluster.

In addition, we will show how value prediction significantly improves the effectiveness of the steering logic by providing a less dense data dependence graph which results in less communication requirements and better opportunities to balance the workload. Since both inter-cluster communications and workload imbalance usually produce a high IPC loss, a clustered architecture may benefit from value prediction more than a centralized one.

5.2 Microarchitecture

This section describes the basic value prediction mechanism assumed in this chapter, as well as the required extensions that we propose for a clustered architecture, and presents a performance evaluation.

5.2.1 Value Prediction

We assume that the microarchitecture implements a stride value predictor [25, 37, 38, 88] that predicts the source operands of the instructions. It has a tagless value prediction table indexed by the PC and by the operand order (left/right). We first assume a very large table (64K entries) to isolate the results from the effects of a limited table size, and we later evaluate the impact of a table with sizes ranging from 1K to 64K entries (section 5.4.3). Each entry contains the last value, the last observed stride and a 2-bit counter that assigns confidence to the prediction. On a misprediction, the stride is updated, but only if the confidence counter is lower than 3. Such an updating policy [39] avoids mispredicting twice on many inner loops, and has a similar purpose and performance as the 2-delta stride technique [25]. Since each prediction involves a table access and an addition, we assume that value predictions are available 1 cycle after the fetch, i.e. at the decode stage. Table updates are done at decode time.

When a source operand is not yet available at dispatch time, and its predicted value is confident (the confidence counter is greater than 1), the instruction is dispatched speculatively and may use the predicted value. The instruction that will produce this value is identified, and it is assigned the task of verifying that its output matches the prediction. The verification occurs

during the writeback stage of the producer instruction, and it takes one cycle. If it fails, the dependent misspeculated instruction is invalidated and reissued.

We have assumed a selective invalidation and reissue mechanism [57, 82], i.e. after the mispredicted instruction is reissued and executed, a new value is produced and propagated to dependent instructions, which in turn reissue, and so on. Only the instructions that depend on the mispredicted instruction are invalidated. The mechanism is in fact the existing issue mechanism, and therefore we have assumed no additional penalty for each instruction restart.

Since mispredictions are found late in the pipeline, during the writeback stage of the producer instruction, the misspeculated dependent instructions are actually re-issued several cycles later than they would do if they were not speculative (see diagram in figure 5-6). Hence, they are effectively delayed by as many cycles as pipeline stages between issue and writeback. Even though speculation is restricted by the confidence bits to the most predictable values, the penalty incurred by mispredictions may still be so high that it offsets most of the performance gains of correct predictions. Improving the predictor accuracy and restricting speculation to those instructions with a higher impact on the critical path length [31] are valid approaches to reduce these overheads. It is beyond the scope of this thesis to study these alternatives, however we found that speculating only on values that are the results of loads (many of which are likely in the critical path, due to cache misses) provides an additional 2.7% average performance improvement, so this simple constraint is assumed for the rest of the experiments in this chapter.

5.2.2 Speculation on Remote Operands

For a clustered architecture, the above speculation procedure is further extended, in order to reduce inter-cluster communications, which is a major goal of our proposal. The extension applies to the case when a source register is not currently mapped on the cluster where the instruction is being dispatched. In this case, the instruction is dispatched speculatively with the predicted value, even if the register is unavailable, and a special *verification-copy* instruction is dispatched to the cluster where the operand is to be produced, instead of a normal copy instruction. During the cycle following the read stage, the verification-copy compares locally the prediction with the computed register value, and it sends the corresponding validation signal through the interconnect. The actual communication of the correct value is only required in case of comparison mismatch, and then the remote misspeculated instructions must re-issue with the correct input.

5.2.3 The Baseline Steering Algorithm

The cluster assignment algorithm assumed for our baseline clustered architecture is the Priority RMB scheme (PRMB), since it was shown to be the most effective (see chapter 4). In normal operation, this algorithm firstly selects from among clusters to minimize communication penalties and secondly, it chooses the least loaded cluster. However, when the workload imbalance exceeds a given threshold, then the first criterion is ignored, and the least loaded cluster is always chosen. We reproduce the algorithm here for clarity:

1. To minimize communication penalties:
   1.1. If there is any unavailable operand, choose its producer cluster
   1.2. Else, select clusters with highest number of source registers mapped
2. Choose the least loaded among the above selected clusters

Except: if imbalance is greater than a given threshold, then ignore rule 1

5.2.4 Performance Evaluation

We ran a set of experiments to compare the performance impact of value prediction on a clustered architecture, with two and four clusters, and also on a centralized architecture, all of them with similar resources and identical pipeline length (see table 2-1 for details). These experiments use the Mediabench benchmark suite [56, 62] and assume all the experimental setup described in section 2.2, including the simulator and the architectural parameters as well as the stride value prediction and the PRMB steering scheme described above.

Figure 5-1 shows the IPC of these three architectures without value prediction, which will be referred to as the baseline architectures. These baseline IPC results will be used for further speedup results in this chapter. Figure 5-2 shows the speedups achieved when value prediction is used in each of the three architectures. These results show that the impact of value prediction on a centralized architecture is very small, just a 2.9% speedup on average, and negative for several benchmarks. In contrast, the average speedup on a four-clustered architecture is 9.5% (5.5% for two clusters), because value prediction drastically reduces inter-cluster communications per instruction from 0.22 to 0.13 on average (from 0.12 to 0.08, for two clusters). There is one exception, *rawdaudio*, which loses 4% IPC on a 2-cluster architecture due to a slight increase of communications.

5.3 A Steering Scheme for Value Prediction

In this section we focus on how the steering logic of a clustered processor may improve its effectiveness by being aware of the existing value prediction mechanism. Let us assume that predicted source registers will never cause communications or delays, which is true if the prediction does not fail. Then, we could relax the constraints imposed by dependence-based steering criteria (rule 1, in section 5.2.3) for predicted operands, in order to concentrate on improving the workload balance (rule 2). As far as the misprediction rate is kept low, this policy may improve significantly the workload balance.

5.3.1 Enhancing the Partitioning through Value Prediction

In more detail, we propose the following two modifications to the baseline PRMB steering heuristic. First, when the source register of an instruction is predicted and it is not yet available, the steering algorithm considers it as available, thus applying rule 1.2 instead of rule 1.1. In other words, the algorithm does not force to steer the instruction to the cluster where this operand is going to be produced, since it is unlikely that this dependence is in the critical path.

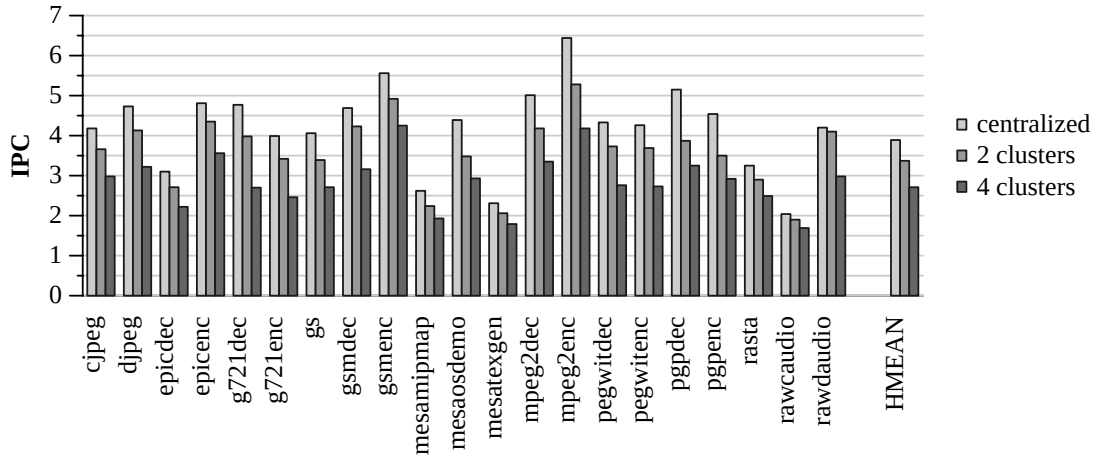


Figure 5-1: IPC of baseline architectures, without value prediction

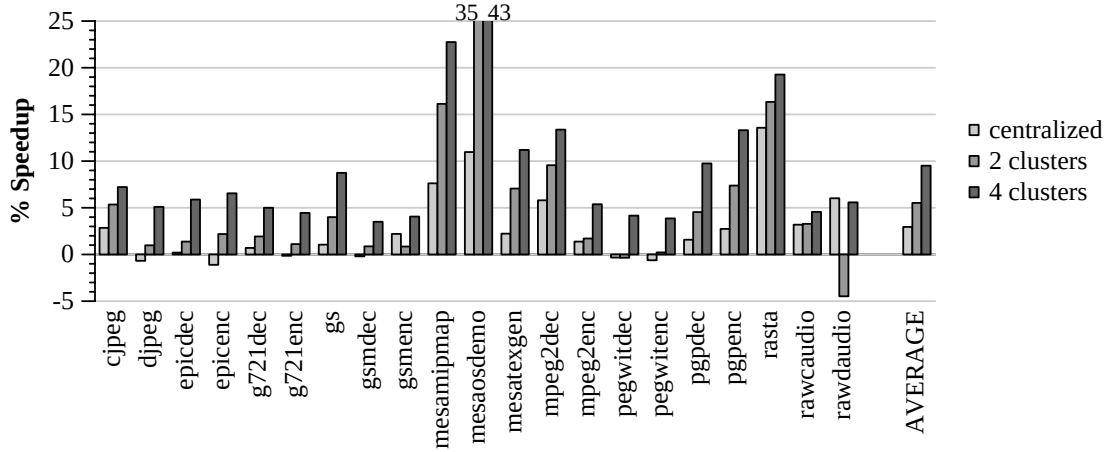


Figure 5-2: Impact of using value prediction on IPC

Second, rule 1.2 of the steering algorithm considers any value-predicted source operand as being mapped in all clusters. As a consequence, this operand does not constrain the set of candidate clusters because, regardless of the cluster it is sent to, it will not cause any additional inter-cluster communication (unless the prediction fails and the operand is remote).

In summary, these two modifications to the steering algorithm eliminate in some cases the constraints imposed by communications/delays issues so that the algorithm has better opportunities for balancing the workload (since rule 2 selects one cluster from a wider choice of clusters).

We evaluated the impact of these two modifications on a four-cluster configuration, and found that they produce worse IPC speedups than the baseline PRMB steering scheme (6.9% average speedup instead of 9.5%). The average NREADY workload imbalance metric (defined in section 4.1.2) is reduced by 32% and, since imbalance correction actions (which ignore communication issues) are less frequent, one would expect also to have less inter-cluster communications. However, the communications ratio (which mostly influences the IPC)

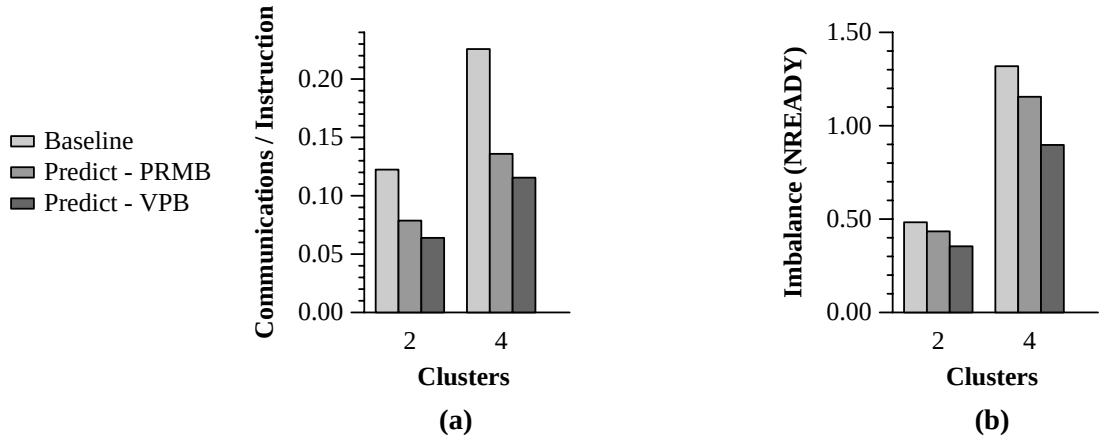


Figure 5-3: (a) Average inter-cluster communications ratio (b) Average workload imbalance

remains almost constant because there is also a communications increase due to an indiscriminate use of the optimistic initial assumptions. More specifically, with the proposed modification to rule 1.2 of the baseline PRMB steering scheme, an instruction that uses a predicted source operand may be steered to a cluster where it is not mapped. In this case, if the prediction fails, the instruction will be re-issued non-speculatively, and a communication will be required to read the correct operand from a remote cluster.

5.3.2 The VPB Steering Scheme

To minimize the above mentioned communications increase, the above mentioned modification to rule 1.2 should only apply to those cases in which there is a potential for improving the workload balance. In particular, we propose that the steering logic considers predicted source registers to be mapped in all clusters only when the DCOUNT workload imbalance counter (see definition in section 4.1.2) is higher than a given threshold. We set this threshold empirically to 16 for four clusters, and 8 for two clusters. In other words, if the workload is very well balanced, the steering does not rely on value prediction to improve workload balance, since it may increase the communication requirements. We refer to this technique as the Value Prediction Based (VPB) steering scheme.

Figure 5-3 compares the average workload imbalance and communication rates for three different cluster architectures: the baseline without value prediction, value prediction with PRMB steering and value prediction with VPB steering, each one configured with either two or four clusters. Figure 5-3 shows the IPC speedups of the two architectures with value prediction over the baseline.

For an architecture with four clusters, it is shown that value prediction with VPB steering requires only 0.11 communications per instruction, which is less than with PRMB steering, and 49% less than the baseline architecture without value prediction. It is also shown that the workload imbalance of VPB steering is lower than that of PRMB steering, and 32% lower than that of the baseline architecture without prediction. Accordingly, while value prediction with PRMB steering achieves only a 9.5% average speedup, with VPB steering it achieves a 14.4%.

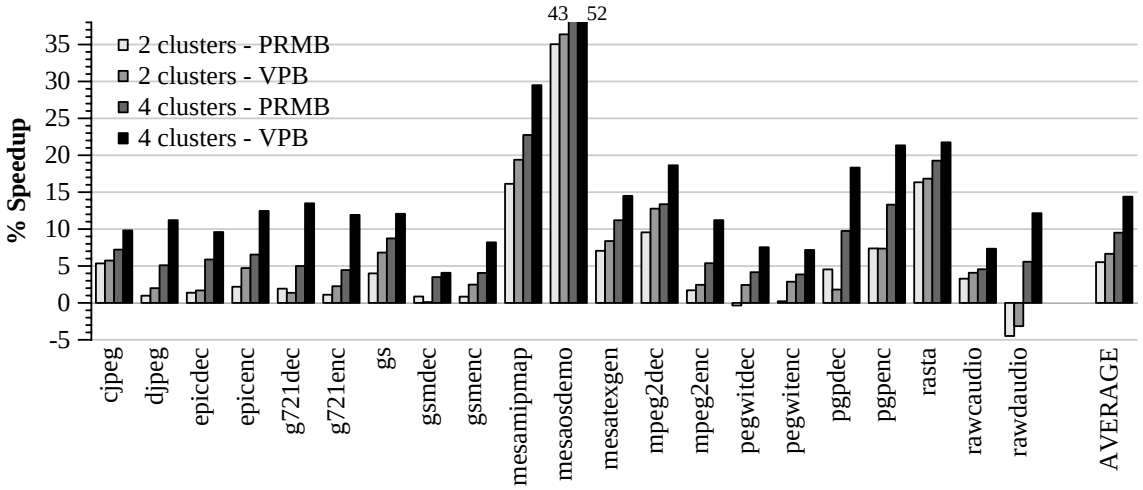


Figure 5-4: Speedups of value prediction, with PRMB and VPB steering schemes

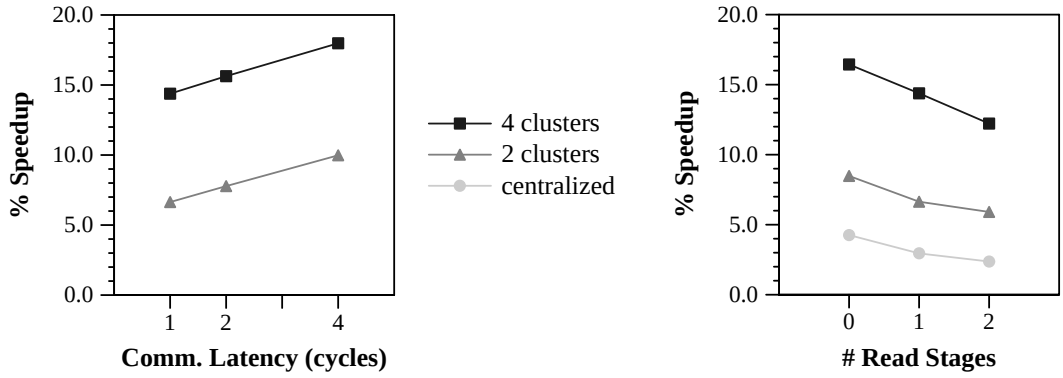
For an architecture with two clusters, the results show similar trends: 0.06 communications per instruction, which is 48% less than the baseline, and a 27% imbalance reduction. However, the performance increase is smaller than for four clusters, a 6.6% average speedup, because there are less communications to be removed with value prediction.

In the above experiments, it was assumed a very conservative value prediction scheme. In order to estimate the potential of this technique with a more accurate predictor, we modelled a perfect predictor and allowed value speculation on any kind of integer values, not only load results. We found speedups of 58%, 38% and 27% for four clusters, two clusters and a single cluster respectively, which suggests that the performance of the VPB steering may significantly be improved by a more effective value predictor, and confirms that the benefit of value prediction increases with the number of clusters.

In summary, we observed that value prediction drastically halves the amount of inter-cluster communications on a cluster organization, and produces significant performance improvements which are higher as there are more clusters, and much higher than for a centralized architecture. We also observed that performance is further increased when adequate steering techniques are implemented that take advantage of value prediction for improving the workload balance. We can thus conclude that value prediction is a very effective technique to reduce the communication requirements of clustered processors.

5.4 Sensitivity Analysis

In this section we analyze the impact on performance of several critical design parameters like inter-cluster communication latency, misprediction penalty and predictor table size.



(a) **(b)**

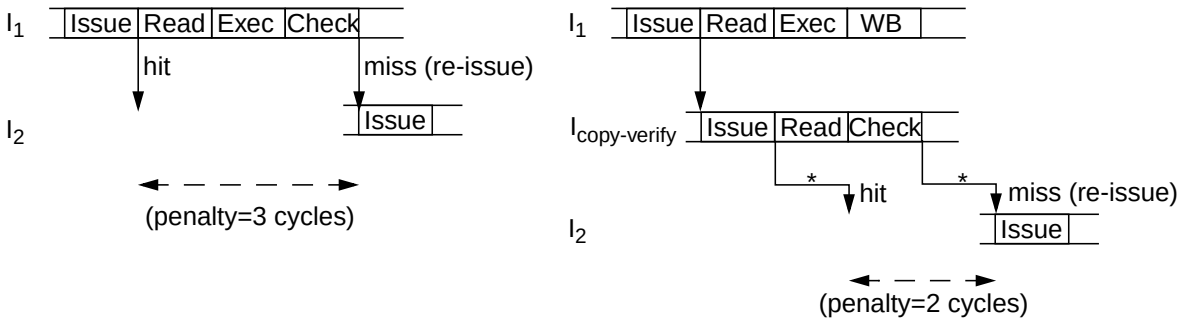
Figure 5-5: Sensitivity of value prediction speedups to (a) communication latency, and (b) latency of the register read stage

5.4.1 Communication Latency

In future technologies the widening gap between the relative speeds of gates and wires will decrease dramatically the percentage of on-chip transistors that a signal can travel in a single clock cycle [1]. Using high clock rates may require not only to reduce the capacity of many components like register files and issue windows, but also to pipeline more deeply the access to other structures, and it may imply a longer inter-cluster communication latency.

In previous sections we assumed that inter-cluster communications take 1 cycle (there is a 1 cycle “bubble” between the copy instruction and the dependent instruction, in another cluster). In section 4.3.3 it was analyzed the impact of the communication latency on performance for several steering schemes, and it was shown how the steering schemes that produced more communications suffer a higher performance degradation. Since value prediction helps reduce the communications demands, one would expect that value prediction also reduces the sensitivity to the communication latency. In other words, the longer the communication penalty, the higher the benefit of eliminating communications. In the following experiments we modelled inter-cluster communication latencies from 1 to 4 cycles, on architectures with 2 and 4 clusters, and measured the speedups of value prediction with VPB steering over the corresponding baseline without prediction.

Figure 5-5a shows that the speedup of value prediction increases with longer inter-cluster communication latencies. When the latency is extended from 1 to 4 cycles, the speedup increases from 14.4% to 18.0% with four clusters (and from 6.6% to 10.0%, for two clusters). Thus, we can conclude that value prediction may become still more useful as wire delays tend to grow with future technologies.



(a) I1, I2 steered to the same cluster

(b) I1, I2 steered to different clusters
(*) inter-cluster latency=1 cycle

Figure 5-6: Timing diagram of two instructions I_1 and I_2, where I_2 mispredicts the value produced by I_1, and must re-issue non-speculatively (the arrows show wakeup signals in case of hit and miss)

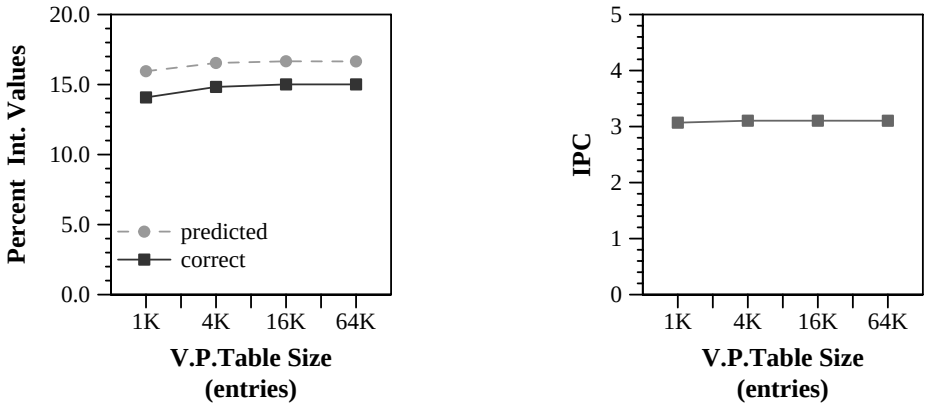
5.4.2 Register Read Latency and Misprediction Penalty

As discussed in section 5.2.1, since mispredictions are discovered during the writeback stage of the producer instruction, the mispredicted instruction must re-issue 3 cycles later (2 cycles, for remote operands) than it would do with a correct prediction, as shown in the timing diagram of figure 5-6. As shown, the pipeline depth between the issue and writeback stages determines the minimum misprediction penalty, so it may have a direct impact on IPC. We have modelled several pipelines, having 0, 1 and 2 read stages. The first model does the issue and register read in a single stage (like it occurs in short pipeline layouts), while the other two have 1 and 2 read stages respectively. The first model also corresponds to an architecture that reads the register file before inserting the instruction into the issue queue, like a PowerPC [94].

Figure 5-5b shows the value prediction speedups for these three pipeline depths. The longer the pipeline, the higher the misprediction penalty. Therefore, with read stages ranging from 0 to 2, speedups vary from 16.4% to 12.2% with four clusters, from 8.5% to 5.9% with two clusters, and from 4.3% to 2.4% in a centralized architecture. We can conclude that processors with fewer stages between issue and exec will benefit more from this technique than other more deeply pipelined ones. More generally, this result shows that value prediction is quite sensitive to the misprediction penalty.

5.4.3 Value Predictor Table Size

The predictor table size determines the prediction accuracy, which has a significant influence on the performance. We have evaluated the impact of the predictor table size on a clustered architecture. Figure 5-7a shows the predictor rate and accuracy for several table sizes. With a 64K entry table, the prediction rate (number of predictions used by speculative instructions over



(a) **(b)**

Figure 5-7: Impact of value predictor table size for 4 clusters (a) prediction rate and accuracy (b) IPC.

total number of integer values) is 16.6%, which is very low, due to predicting only load results and remote operands. The prediction accuracy is 90.1%. Reducing the table size to 1K entries reduces the prediction rate to 15.9% and the accuracy to 88.2%.

Figure 5-7b shows that on average, for a four-cluster configuration, there is less than 1.1% IPC degradation when the predictor table size is reduced from 64K to just 1K entries.

5.4.4 Experiments with the SpecInt95

In previous sections, the Mediabench [56, 62] benchmark suite was used for all the experiments. For the sake of higher generality of our conclusions, we run an identical set of experiments with the SpecInt95 [97] benchmark suite.

We found that the speed-ups of value prediction, with PRMB steering, are 6.7% for 4 clusters, 3.9% for 2 clusters and 3.6% for a conventional centralized architecture. We also found that, by using the enhanced VPB steering scheme, the speed-ups of value prediction are 7.5% for 4 clusters and 6.2% for 2 clusters.

Compared to the results with the Mediabench suite shown in previous sections, these results show similar trends, although the speed-ups are smaller. However, the same overall conclusions hold for both benchmark suites.

5.5 Conclusions

Future microprocessors are likely to be communication bound due to the increasing penalty of wire delays. In this chapter we showed that value prediction can be an effective instrument to improve communication locality. In particular, we have presented an approach to reduce inter-

cluster communication by means of a dynamic steering logic that leverages value prediction. Values produced in one cluster and consumed in another one may not require long wire delays to propagate from the producer to the consumer if the consumer can correctly predict the value. The validation required by the prediction is locally performed in the producer cluster.

We have shown that value prediction removes communications even for previously proposed steering schemes not specially designed to exploit value prediction. However, performance is higher if the steering logic exploits the predictability of values to improve the workload balance. We have presented a novel steering 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. Value prediction, 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 in a four-clustered architecture. In contrast, an identical value predictor achieves only a 3% speedup in a centralized architecture.

We have also shown that this technique may produce even better improvements in future technologies, as wire delay - and hence communication latency - increases. The performance improvement of value prediction is quite sensitive to misprediction penalty, but it is less sensitive to the predictor table size, for the considered set of benchmarks and table sizes.