# 6

# CONCLUSIONS AND OPEN-RESEARCH AREAS

*In this Chapter, the main conclusions of this thesis are presented. Moreover, some open-research areas in this topic are pointed out.*

## 6.1. CONCLUSIONS

In this thesis, a processor architecture that boosts up the performance of single applications by means of exploiting speculative thread-level parallelism has been presented. The execution model proposed is based on identifying pairs of instructions in the program, -the spawning and the control quasi-independent point,- at where the speculative threads are created. When a spawning point is reached by a thread, a new speculative thread is spawned starting its execution at the control quasi-independent point. Thus, both threads proceed in parallel until the spawner thread reaches the control quasi-independent point of a more speculative thread. Threads are validated and committed in program order. Any thread is allowed to spawn more speculative threads.

Parallel threads are speculative since there may be data and control dependences among them. Two mechanisms to deal with interthread data dependences have been studied in this thesis. The former one is based on synchronizing the consumer thread with the producer one. With this mechanism, the speculative thread stalls its execution until the dependent value is forwarded from the producer thread. The performance reported for this kind of mechanisms is very poor in comparison with the amount of hardware used. A 16-thread unit clustered speculative multithreaded processor only achieves a 43% speed-up over a single-threaded execution for the SpecInt95 with perfect synchronization mechanisms.

On the other hand, value prediction may help to break interthread data dependence chains among concurrent threads. If the processor is able to correctly predict all the interthread data dependent values for a given speculative thread, then the spawner and the spawned threads will be executed as if they were independent.

Different value predictors have been analyzed in this thesis and a new one, which is referred to as increment predictor has been proposed. This predictor takes into account the control flow followed by the thread to perform the predictions. Very high prediction accuracies are reported for a predictor of a small size.

Overall, the performance of speculative multithreaded processors with register value prediction presents impressive speed-ups even for relatively small sizes of the predictor.

Finally, the impact of the selection of the spawning pairs in the performance of speculative multithreaded processors has been studied. Two families of spawning schemes have been analyzed. In the first one, speculative threads are assigned to well-known program constructs such as loops or subroutines. These constructs provide certain control independence and the speed-ups obtained by such spawning poli-

cies for the SpecInt95 are quite impressive, higher than 7 for a 16-thread unit configuration with perfect register value prediction.

On the other hand, a new systematic spawning identification scheme based on quantifying the thread features through an off-line analysis is presented. This scheme uses profile information to characterize some run-time features of programs. Main code considered by this spawning scheme are the distance between the spawning and the control quasi-independent point, the number of dependences and the predictability of the dependent values. The speed-up reported for this spawning scheme is close to 9 compared with a single-threaded execution with perfect value prediction and it outperforms the obtained for the spawning schemes based on heuristics by a 20%.

Overall, a mechanism to boost up the performance of difficult-to-parallelize applications has been presented. This mechanism speculatively partitions the program into threads and is able to find huge amounts of thread level parallelism in applications that are known to be hard to parallelize such as the SpecInt95. Performance results report an average speed-up higher than 5 over a single-threaded execution for a 16-thread unit clustered speculative multithreaded processor with a 16-KB stride value predictor and a fixed 8-cycle thread initialization overhead.

## 6.2. OPEN-RESEARCH AREAS

### 6.2.1. Memory Subsystems for Speculative Multithreaded Processors

The profile-based spawning scheme may collect a significant amount of information such as the memory access patterns of the memory instructions as well as dependent store-load instructions. However, this information is not used to reduce memory misspeculations. Traditional mechanisms to detect interthread memory dependences and misspeculations are based on extensions of cache coherence protocols. The MultiValue Cache, proposed in this thesis, provides certain data dependence speculation by delaying the execution of some memory instructions if a memory dependence is predicted to occur. However, this mechanism only works for the sequential thread ordering spawning model and it is quite complex to adapt for the unrestricted spawning model.

Therefore, in order to obtain benefit of such information collected by the spawning scheme, a new design for the memory subsystem may be needed. In this design, interthread memory dependent instructions would be delayed until the producer instruction finishes its execution in the same way as the MultiValue cache does.

### 6.2.2. Data Value Prediction for Memory Values

In this thesis, the study of value prediction has been limited to predict live-in register values even though the potential speed-up reported for perfect live-in memory value prediction was impressive.

Memory value predictors can be implemented in different ways, just predicting the value that will be produced by a dependent load or combining with data dependence speculation and predicting the value that will be in a predicted memory location.

### 6.2.3. Evaluate the Sensibility of the Profile-Based Spawning Scheme

When profile information is used to implement a given technique, it is always present the same question: Is the profile information used representative? In this thesis, profile information has been used to select the spawning pairs based on quantifying some criterias. Obviously, the spawning pairs selected are statistically the best for the input used in the profile analysis, but it is possible that for any other input results were worse that for any other spawning pair set. In fact, it is very likely that the optimal set of spawning pairs for two different inputs of a given program are completely different.

Then, to improve the quality of the selection of the spawning pairs, different inputs to obtain the profile information can be considered in order to refine the spawning pair selection.

### 6.2.4. Processor-Aware Spawning Schemes

The profile-based spawning scheme may collect any dynamic information of the candidate spawning pairs in order to select which are the most appropriate pairs. In this thesis, three selection mechanisms have been analyzed: maximizing the distance between the spawning and the control quasi-independent point, maximize the number of independent instructions between the threads and maximizing the number of independent plus input-operand predictable instructions. Some others can also be taken into account such as minimizing the number of live-in input values, etc.

Moreover, the selection of the spawning pairs can also be improved if the selection process take into account information of the configuration of the speculative multithreaded processor such as the number of thread units, latencies for forwarding register and memory values, etc.

We believe that the approach presented in this thesis opens the possibility of studying other important criteria that can enhance the thread spawning scheme of speculative multithreaded processors.