SPECULATIVE MULTITHREADED PROCESSORS

Pedro Marcuello

DEPT. OF COMPUTER ARCHITECTURE
UNIVERSITAT POLITÈCNICA DE CATALUNYA
Barcelona (SPAIN)

A THESIS SUBMITTED IN FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
Doctor en Informática

July, 2003

SPECULATIVE MULTITHREADED PROCESSORS

Pedro Marcuello

DEPT. OF COMPUTER ARCHITECTURE
UNIVERSITAT POLITÈCNICA DE CATALUNYA
Barcelona (SPAIN)

Advisor: Antonio González

A THESIS SUBMITTED IN FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
Doctor en Informática

July, 2003

ABSTRACT

Several studies on the limits of the instruction-level parallelism have shown that current superscalar organizations are approaching to diminishing return points. Such studies have shown that just taking benefit of the ever increasing number of transistors to simply scale up such architecture hardly provides any improvement. These limits in instruction-level parallelism have motivated the research on alternative techniques to increase the performance of the processors. An approach that is being considered by several vendors is exploiting coarse-grain parallelism in addition to the conventional instruction-level parallelism so that future processor generations provide capabilities to exploit both types of parallelism. These processors are known as Multithreaded Processors

The way thread-level parallelism is being exploited in multithreaded processors is usually in a non-speculative manner. The hardware support for maintaining different contexts simultaneously are used to execute different applications in each of these contexts or to execute threads corresponding to parallelizable applications. In the former case, threads are independent among them and the processor obtains a better throughput due to the better usage of the resources of the processor even though the individual execution time of applications is increased by the fact that resources are now shared among different processes. In the latter case, applications executed in different threads reduce their execution time but the partitioning process may be a difficult task for irregular or non-numerical applications. This kind of applications hardly benefit of this type of thread-level parallelism since traditional parallel compilers usually fail to find parallel threads.

Instruction-level parallelism has traditionally used speculation techniques to increase its performance. In this thesis we propose to exploit thread-level parallelism speculatively. Thus, those applications the compiler is unable to find non-speculative threads can be partitioned into speculative threads to reduce their execution time. Threads are speculative since they are data and control dependent on previous threads. Then, in the case speculation is correctly performed, applications drastically reduce their execution time due to the additional exploitation of speculative thread-level parallelism. However, if a misspeculation occurs, roll-back mechanisms are necessary to return the processor to a correct state. Processors that are able to execute speculative threads are referred to as Speculative Multithreaded Processors.

In this thesis, the execution model of speculative multithreaded processors and the necessary requirements to implement it are studied. This execution model is based on inserting spawn instructions into the sequential code. The execution of programs in a speculative multithreaded processor is similar to any other conventional processor until a spawn instruction is reached, then a speculative thread is created at the corresponding point indicated by the spawn instruction and both threads proceed in parallel. When the spawner thread reaches where the speculative thread has started, then a verification process checks for the

correctness of the speculation. If the speculation was correctly done, the context of the non-speculative thread is committed and its context freed for a future usage of new speculative threads. If not, a recovery mechanism is started. In this execution model, there must always be a non-speculative thread and there may be multiple speculative threads.

To support this execution model, several requirements are needed: i) hardware support for spawning and managing speculative threads and ii) a partitioning mechanism to divide programs into speculative threads.

In this thesis, different platforms to manage concurrent threads are discussed. Clustered processors benefit from low wire delays, power and complexity even though communication among concurrent threads can suffer from delays. Centralized processors may benefit from sharing resources and low communication cost but the complexity of the required hardware is higher. In any case, hardware has to be able to simultaneously execute several contexts taking into account that some of the values used by the threads are shared while others are private in such a way that a variable may have different values simultaneously, one for each of the concurrent threads.

Regarding the hardware support, this thesis has paid special attention to the management of interthread data dependences since they have a high relevance on the overall performance of such processors. As finding data and control independent threads is almost impossible for irregular applications, speculative threads depends on the values produced by previous ones, either register or memory values. Different mechanisms to deal with interthread data dependences are analyzed such as synchronizing the producer and the consumer thread, or predicting the dependent values. In the former, schemes to early forward the dependent value to the consumer thread are studied, especially for memory values. On the other hand, the latter proposal is the most promising one since if values are correctly predicted, speculative threads are executed as if they were independent. Different well-known value predictors are evaluated and a new one especially targeted for this type of architectures is presented, the increment value predictor. This predictor takes into account information of the control-flow of the thread to predict the values and it obtains impressive results for small sizes of the predictor.

Finally, the way applications are partitioned affects the performance of these processors. In this thesis, different spawning schemes have been proposed and analyzed. One family of schemes spawn speculative threads associated to well-known program constructs whose characteristics may provide significant benefits. Spawning schemes that belong to this family are those that creates speculative threads at loop iterations, loop continuations and subroutine continuations. In another family of spawning schemes, speculative threads are created through a profile-based analysis of each particular code. Here, those parts of the code that present the most appropriate characteristics are selected to spawn speculative threads. Selection criterias such as control independence, minimum size, data independence and data predictability are considered in order to choose the most effective spawning pairs. Performance results for both families of spawning policies are quite impressive, and the profile-based spawning scheme is shown to outperform those based on heuristics.

AGRADECIMIENTOS

En primer lugar, quiero agradecer a mi director de tesis, Antonio González, su confianza, su paciencia y todo lo que me ha enseñado a lo largo de estos últimos años. Agradecerle también a José González, Pepe, por haberme metido en esta aventura, por haberse acordado de mi nombre cuando hubo una beca disponible parar empezar a estudiar el doctorado.

Dar gracias también al CEPBA por permitirme utilizar las máquinas con las que he hecho todas las simulaciones de esta tesis.

También me gustaría agradecer el apoyo recibido por todos los miembros del departamento con quienes he pasado muy buenos momentos. No pretendo dar una lista exhaustiva y espero que nadie se moleste por no aparecer aquí, pero si querría agradecer especialmente a Josep-Llorenç, Carlos, Suso, Ramon, Alex, Dani y Joan Manel los grandes ratos que me han hecho pasar estos años.

Por otro lado, también agradecer el apoyo del resto de amigos de fuera del departamento: mis colegas del barrio con quienes perpetro partidos de baloncesto tres veces por semana, así como aquellos que forman la JerboChat. Además, de forma especial quiero agradecerle a Raquel estos últimos años. Espero que sea por mucho tiempo.

También agradecer a Walter Bays y Bodo Parady, de Sun Microsystems y John Shen y Hong Wang, de Intel, que me ofrecieran la posibilidad de hacer sendos internships en California, así como todo el grupo del MRL de Santa Clara por facilitarme la estancia allí.

Finalmente, no existen palabras para expresar el agradecimiento a toda mi Familia, con mayúsculas y en el sentido más amplio de la palabra, por su confianza, apoyo moral y haber estado allí siempre que los he necesitado, tanto los que siguen aquí como los que nos han abandonado en estos últimos años como mi abuela, mi tio Blas y en especial, mi padre. Y dentro de este gran conjunto, especial gracias a mi núcleo familiar más cercano, a mi madre y mis hermanos por su ayuda, esto no habría podido realizarlo sin estar vosotros detrás, muchas gracias.

A mis padres,
Luís y Concha,
mis hermanos,
Álex y Conchi,
y Raquel

CONTENTS