

Foundations for Massively Parallel on-chip Architectures using Microthreading

Chris Jesshope (PI): Computer Systems Architecture (CSA)
 Jan Bergstra and Paul Klint: Programming Research Group (PRG)
 University of Amsterdam (UvA), Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

Introduction

Moore's law predicts that transistor density doubles every 18 to 24 months and this scaling presents both problems and opportunities to computer architects. Current micro-architectures use clock speed as a means of increasing performance rather than concurrency causing them to hit the power wall. The aim of this project is to develop a micro-architectural model microprocessors that exploits concurrency through chip-multiprocessors giving performance and avoiding the power wall. More importantly it must do so from existing sequential code. The model is based on code fragmentation and uses microthreads. It was first introduced in the mid 1995 [3]. The code fragments (micro-threads) are interleaved on a single processor and provide latency tolerance and are distributed over multiple processors to obtain performance or reduce power by frequency and voltage scaling.

Families of Microthreads

The model is incremental and adds a few instructions to an existing ISA that implement explicit concurrency controls. These define the parametric sets of code fragments - *Families of Microthreads* - to execute loops concurrently [4]. The code fragments are dynamically scheduled on multiple processors (see figure 1).

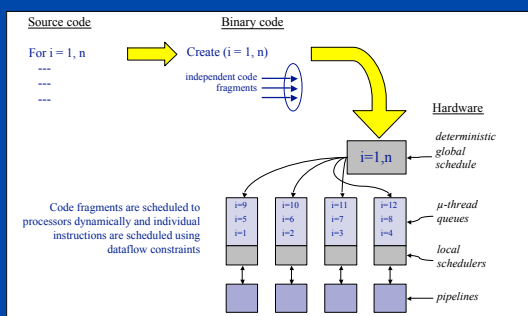


Figure 1. Illustration of Micro-Threaded scheduling

Parametric concurrency and dynamic schedules mean the same code will run on any number of processors up to some practical limit defined by the parameters.

The model uses synchronised shared registers with weakly consistent shared memory synchronised by barriers. In [2] an implementation of the model has been shown to be completely scalable when executing independent loops.

Registers are separated into a global part and a micro-context for each iteration, which allows code to be shared between iterations. Micro-contexts provide the mechanism for inter-iteration communication (loop-carried dependencies). This is achieved by an on-chip network, where different models of dependencies require different communication patterns. Currently the model supports fixed offsets between iterations and uses a ring network.

Chip - Multiprocessors

A Microgrid is a chip multi-processor comprising M Microthreaded. Each processor has each own local register file and access to a network for shared-register communication (figure 2).

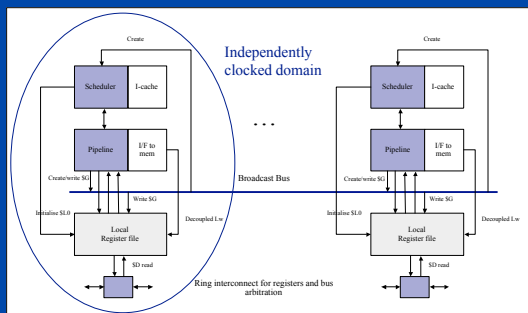


Figure 2. Chip multi-processor based on an asynchronous collection of micro-threaded pipelines.

Early Simulation Results

We have built a cycle-accurate simulator of an implementation of the proposed model based on the Alpha ISA. This is able to execute micro-threaded code on different numbers of processors. The simulated system uses a seven-stage, in-order pipeline with no branch prediction and a relative slow but non-blocking second level of shared memory.

Figure 3 shows the results from the execution of the Livermore hydro-fragment for a fixed number of iterations, running the same hand-compiled binary code on from 1 to 2048 processors. Each processor has a set of 1024 local registers.

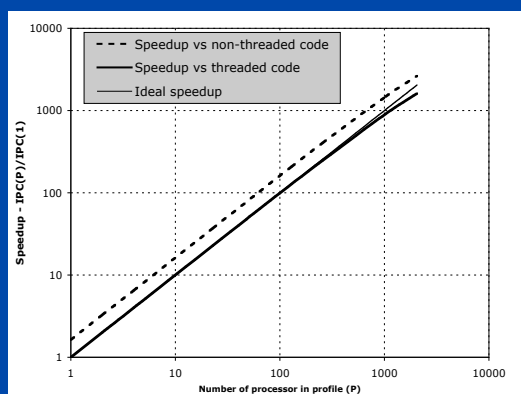


Figure 3. Speedup of hydro-fragment against number of processors

These results show a speedup for 256 processors, which is within 3% of the ideal. Speedup is still within 20% of the ideal for 2048 processors. The onset saturation is because the code is executed for a fixed number of iterations (64K), which represents 32 iterations per processor, or a 20% utilization of the chip's resources for the largest number of processors (2048) in the example.

Research Questions

We have set ourselves the following research question. "Is it possible, through the introduction of simple and explicit concurrency controls, to develop a systematic approach to:

- incrementally designing new processor architectures (i.e. based on an existing ISA and infrastructure);
 - dynamically managing and optimising the available resources for a variety of goals such as performance, power and reliability (i.e. resulting in autonomous and self-adaptive microgrids);
 - formally defining the architectures' execution properties;
 - incrementally developing the architectures' infrastructure (i.e. simulators, compilers, binary-to-binary translators and even silicon intellectual property);
- all within the context of ten to fifteen years of silicon-technology scaling - i.e. maintaining scalability over a thousand fold increase in chip density?"

References

- [1] C Jesshope (2004) Micro-grids - the exploitation of massive on-chip concurrency, invited paper, Cetraro HPC workshop 2004, Cetraro, Italy, to be published in *Advances in Parallel Computing*, 2005 (<http://staff.science.uva.nl/~jesshope/Papers/HPC-paper.pdf>)
- [2] A Bolychevsky, C R Jesshope and V B Muchnick, (1996) Dynamic scheduling in RISC architectures, *IEE Trans. E, Computers and Digital Techniques*, **143**, pp309-317
- [3] C. R. Jesshope and B. Luo (2000) Micro-threading: A New Approach to Future RISC, *Proc ACAC 2000*, pp34-41, ISBN 0-7695-0512-0 (IEEE Computer Society press), Canberra Jan 2000
- [4] Luo B. and Jesshope C. (2002) Performance of a Micro-threaded Pipeline, in *Proc. 7th Asia-Pacific conference on Computer systems architecture*, **6**, (Feipei Lai and John Morris Eds.) Australian Computer Society, Inc. Darlinghurst, Australia, ISBN ~ ISSN:1445-1336 , 0-909925-84-4 , pp83-90