International Journal of Parallel Programming, Vol. 34, No. 3, June 2006 (© 2006)
DOI: 10.1007/s10766-006-0013-2

Special issue on Micro-grids – Guest Editor Introduction

This special issue of the International Journal of Parallel Processing contains papers selected from the first open international Micro-grids workshop. This event was hosted by the Computer Systems Architecture group (http://www.science.uva.nl/research/csa/) at the University of Amsterdam on July 1–2, 2005 at the Science Park in Amsterdam. The theme of the workshop series is scalable on-chip parallelism, and it aims to bring together experts in micro-architecture, languages and compilers with the goal of disseminating longer-term research into the significant problems associated with scalable on-chip concurrency.

This workshop emerged from the following historical perspective. For many years there has been uniformity in computer architecture with little diversity and no new directions. This had led to a monoculture within the computer architecture research community, where every research group seemed to be designing and testing marginally better mousetraps, without any realisation that mice were no longer an issue. We refer of course to the overwhelming body on literature on speculative, out-of order execution in wide-issue pipelines. With a few exceptions, research into alternative concurrent architectures was marginalised by a neat pincer movement. On the one side, the rate of change of clock speed in mainstream processors was being driven by two factors: an exponential increase in speed from the reduction in device dimensions and, from the 1990s onwards, an additional one-off boost by the use of superpipelining in the micro-architecture. This massive increase in clock speed was matched on the other flank with a well-oiled manufacturing process that brought us new fabrication facilities and new and faster products on an almost annual basis. This combination of circumstances did not allow any diversity in the form of parallel architecture to enter into the market, as it was always cheaper and

faster to simply wait for the next generation of superscalar processors or to use many of them concurrently in a variety of configurations depending on the application.

Concurrent architecture research from the 1980s, such as dataflow, was forgotten; the superscalar processor reigned supreme and perhaps for very good reason. This form of concurrency exploitation required no input from the programmer. Moreover it gave us the holy grail of compatibility, namely that the same binary code could run on each new generation of processor. What is interesting to review over this period is the effect of device-dimension changes in the characteristics of the processors that were implemented in this technological roller coaster. Over a 12 year period, from the early 1990s, Moore's law predicted a packing density increase of 256, and the corresponding speed increase around 16, based on device-dimension scaling. If we look at the Power PC processor as an example of progress (see http://www.rootvg.net/RSmodels.htm), clock speed increased at about twice the predicted rate during this period; i.e., from 33 MHz to 1 GHz. Circuit density in the die has also grown as predicted; but if we look at how it has contributed to concurrency in instruction issue, then we see a very different picture. Here we get an increase of only a factor of 10, with the PPC moving from a 32-bit, single-instruction-issue implementation to a 64-bit, five-way-instruction-issue implementation. If those gates had been used to replicate the single issue base line, we would expect 256 such processors to be possible. So, what has happened?

The faster than predicted clock speed is due to the finer slicing of the pipeline. The smaller than predicted instruction-issue concurrency (a factor of 25) is more worrying and can be attributed to a number of factors, namely: a larger proportion of chip die being used for memory, to mitigate against the memory wall; the fact that out-of-order instruction issue is not scalable; and finally the fact that the larger multi-port register files required are not scalable either. Scalability of this architectural cul-de-sac was not the only issue. Like all CMOS parameters, exponential growth cannot continue unabated without some form of population control flattening out the curve.

Gate dimension still has some scaling left, different sources predict different end points but we are still several orders of magnitude from the fundamental limit of atomic dimensions. Signal propagation and power dissipation are the two parameters which are beginning to cause problems. Power dissipation has been growing exponentially in each new generation of commodity processor, but power densities have already reached the point at which removing the excess heat is no longer economic. The power dissipation has been exacerbated by the large central support structures required for out-of-order issue. It was this, rather than the lack of

scalability, that rang the death knoll for continued increase in issue width in superscalar architectures.

These facts have led to the emergence of multi-core processor chips over the last few years. However, it is clear that applying the same approach to multi-processors-on-chip as has been used at the system level is not an optimal solution. Perhaps the biggest challenge is that multi-processing has only been used by a small percentage of the application market: the large-scale scientific application and the simpler server-application domains. By far the largest segment of the market for processors still demands binary-code compatibility, but the experience from the number crunchers is that each new generation/configuration of multi-processor required some rewriting of the application to tune it for concurrency. This then is the crux of the issue. For too long we have ignored the genericity of research into concurrency and its exploitation; and we are now faced with a world where if we are to exploit what is left of silicon CMOS scaling, we are forced to use multi-processors. Thus the questions these proceedings try to answer are whether scalability can be achieved in on-chip multi-processors? If so, then can the grail of binary-code compatibility be resurrected in a concurrent environment? If not, are there language or compiler solutions that will give some relief from the hands-on approach currently used in large-scale grid computing? These are all pressing issues as we have arrived at the point of departure with little foundation in theoretical research that would give us insight into the future.

This first part of the two-part special issue contains four papers from the Micro-grids workshop which have been significantly expanded and include three on architecture and one on compiler theory. The paper from Ramon Beivide's group looks at scalability in chip multi-processors from the perspective of networks on chip and explores the suitability of dense circulant graphs of degree four for the design of on-chip interconnection networks. Networks based on these graphs have better characteristic measures than planar toroidal graphs, especially in managing collective communications. The paper from Skevos Evripidou's group looks at architectures based on minimal change from current micro-architecture and evaluates the case for data-driven scheduling of threads as a mechanism of exploiting them. This paper considers both performance and power dissipation and concludes that a significant performance increase, or indeed power savings, can be made by backtracking to earlier micro-architectures and running chip multi-processors based on their data-driven methodology. Stamatis Vassiliadis' group's paper is a study into the management of data in concurrent SIMD architectures, such as multimedia processors. This is a very pragmatic paper, which shows significant advantages over the current state of the art in this area. Finally, the paper from Olivier

Temam's group is the odd one out in this issue and is the first of our compiler papers. This paper looks at a unified polyhedral representation of iteration spaces, illustrates how complex the transformation sequences required for significant performance benefits may be and illustrates how the framework may be used in accomplishing those transformations.

We are pleased to present these papers to you, as well as the papers that will follow in the subsequent edition that complete the spectrum of research which was presented in Amsterdam.

Chris Jesshope, Professor
Institute of Informatics
University of Amsterdam
Kruislaan 4031098 SJ Amsterdam
The Netherlands
E-mail: jesshope@science.uva.nl

Alex Shafarenko, Professor
Department of Computer Science
University of Hertfordshire
College Lane, Hatfield, AL10 9AB, UK
E-mail: A.Shafarenko@herts.ac.uk