Microprocessors and Microsystems 38 (2014) 976–990

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

TERAFLUX: Harnessing dataflow in next generation teradevices

Roberto Giorgi [a,*], Rosa M. Badia [b], François Bodin [c], Albert Cohen [d], Paraskevas Evripidou [e], Paolo Faraboschi [f], Bernhard Fechner [g], Guang R. Gao [h], Arne Garbade [g], Rahul Gayatri [b], Sylvain Girbal [i], Daniel Goodman [j], Behran Khan [j], Souad Koliaï [h], Joshua Landwehr [h], Nhat Minh Lê [d], Feng Li [d], Mikel Lujàn [j], Avi Mendelson [k], Laurent Morin [c], Nacho Navarro [b], Tomasz Patejko [b], Antoniu Pop [d], Pedro Trancoso [e], Theo Ungerer [g], Ian Watson [j], Sebastian Weis [g], Stéphane Zuckerman [h], Mateo Valero [b]

[a] Dip. di Ingegneria dell'Informazione e Scienze Matematiche, Universitá di Siena, Italy
[b] Barcelona Supercomputing Center, Spain
[c] CAPS Enterprise, France
[d] INRIA, France
[e] Dept. of Computer Science, University of Cyprus, Nicosia, Cyprus
[f] Intelligent Infrastructure Lab, Hewlett Packard, Barcelona, Spain
[g] University of Augsburg, Germany
[h] University of Delaware, DE, USA
[i] THALES, France
[j] University of Manchester, United Kingdom
[k] Technion, Israel

ARTICLE INFO

Article history:
Received 4 November 2013
Revised 5 March 2014
Accepted 7 April 2014
Available online 18 April 2014

Keywords:
Dataflow
Programming model
Compilation
Reliability
Architecture
Simulation
Many-cores
Exascale computing
Multi-cores

ABSTRACT

The improvements in semiconductor technologies are gradually enabling extreme-scale systems such as teradevices (i.e., chips composed by 1000 billion of transistors), most likely by 2020. Three major challenges have been identified: programmability, manageable architecture design, and reliability. TERAFLUX is a Future and Emerging Technology (FET) large-scale project funded by the European Union, which addresses such challenges at once by leveraging the dataflow principles. This paper presents an overview of the research carried out by the TERAFLUX partners and some preliminary results. Our platform comprises 1000+ general purpose cores per chip in order to properly explore the above challenges. An architectural template has been proposed and applications have been ported to the platform. Programming models, compilation tools, and reliability techniques have been developed. The evaluation is carried out by leveraging on modifications of the HP-Labs COTSon simulator.

1. Introduction

Silicon manufacturing technologies, such as FinFET [1] transistors and 3D-die stacking [2] that are currently available, will allow new chips (that we call teradevices) with a huge number of transistors (for current ITRS [3] projections, 1 Tera or 10^{12} transistors), therefore opening the doors to the possibility of exploiting the extremely large amount of parallelism in different ways. It is expected that such systems will be able to perform at least one Exa-FLOPS or 10^{18} floating-point operations per second.

In such future exascale machines, the number of general purpose cores (i.e., compute elements) per die will exceed those of current systems by far. This suggests a major change in the software layers that are responsible of using all such cores. The three

* Corresponding author. Tel.: +39 0577 191 5182; fax: +39 0577 195 9064.
E-mail addresses: giorgi@dii.unisi.it (R. Giorgi), rosa.m.badia@bsc.es (R.M. Badia), francois.bodin@caps-entreprise.com (F. Bodin), albert.cohen@inria.fr (A. Cohen), skevos@cs.ucy.ac.cy (P. Evripidou), paolo.faraboschi@hp.com (P. Faraboschi), bernhard.fechner@informatik.uni-augsburg.de (B. Fechner), ggao@capsl.udel.edu (G.R. Gao), arne.garbade@informatik.uni-augsburg.de (A. Garbade), sylvain.girbal@thalesgroup.com (S. Girbal), koliai@eecis.udel.edu (S. Koliaï), josh@eecis.udel.edu (J. Landwehr), mikel.lujan@manchester.ac.uk (M. Lujàn), avi.mendelson@tce.technion.ac.il (A. Mendelson), laurent.morin@caps-entreprise.com (L. Morin), nacho@bsc.es (N. Navarro), antoniu.pop@inria.fr (A. Pop), pedro@cs.ucy.ac.cy (P. Trancoso), theo.ungerer@informatik.uni-augsburg.de (T. Ungerer), watson@cs.man.ac.uk (I. Watson), sebastian.weis@informatik.uni-augsburg.de (S. Weis), szuckerm@eecis.udel.edu (S. Zuckerman), mateo.valero@bsc.es (M. Valero).

http://dx.doi.org/10.1016/j.micpro.2014.04.001
0141-9331/

major challenges: programmability, reliability and complexity of design are here briefly introduced. Also, a new Program eXecution Model [4–6] seems suited in order to address such challenges.

Given the large number of transistors and the diversity in the requirements for different applications, it is natural to expect that these massively parallel (or concurrent teradevice) systems will be composed of heterogeneous cores. Thus, programmability of such large-scale systems will be a major challenge. Moreover, such large systems are expected to become more and more susceptible to failures, due to the increasing sensibility to process variations and manufacturing defects. Thus, this extreme scale of device integration represents a second major concern, in terms of reliability, for future many-core systems. Finally, the software industry is lagging behind as general purpose applications cannot take advantage of more than a handful number of cores compared to the larger degree of parallelism offered by the current and future processors. Starting from this premise, there is the need for new ways to exploit the large parallelism offered by future architectures as expected to be a reality beyond the year 2020.

The dataflow concept is known to overcome the limitations of the traditional control-flow model by exploring the maximum parallelism and reducing the synchronization overhead. As recalled by Jack Dennis [7], dataflow is "A Scheme of Computation in which an activity is initiated by presence of the data it needs to perform its function". The dataflow paradigm is not new, but recently it has met mature silicon technology and architectural models to take advantage from the large intrinsic parallelism.

TERAFLUX [8] is a Future Emerging Technologies (FET) large-scale project funded by the European Union. The aim is to exploit the dataflow paradigm in order to address the three major challenges presented above (i.e., programmability, reliability, and manageable architecture design). Since we are targeting 1000+ core systems, the dataflow paradigm enables us to use the increased degree of parallelism which emerges in future teradevices.

The rest of the paper is organized as follows. Section 2 provides a general overview of the project. Remaining sections are focused on describing the concepts together with the major achievements resulting from our research activity. In particular, Section 3 describes possible applications based on the OmpSs programming model, while Section 4 details a further possibility of using a productivity language such as Scala thanks to a dataflow runtime called DFScala. Another common layer (OpenStream, presented in Section 5) is used for mapping feed-forward dataflow into lower-level dataflow threads as expressed by the T∗ Instruction Set Extension, described in Section 6, together with the architecture of our target system. Section 7 describes the Fault Detection Units (FDUs), which provide fault detection management through monitoring techniques and redundant execution of dataflow threads. The experiments are integrated into a common simulator based on the HPLabs COTSon [9] , presented in Section 8. Finally, Section 9 introduces the codelet model, while Section 10 concludes the paper.

2. General overview of the TERAFLUX project

To investigate our concepts, we use dataflow principles at any level of a complete transformation hierarchy, starting from general complex applications (able to load properly a teradevice system) through programming models, compilation tools, reliability techniques and architecture. Fig. 1 shows the TERAFLUX layered approach.

Different layers allow to transform application source code into a dataflow-style binary, and to execute it on the target architecture (which is at the current moment based on off-the-shelf cores like x86_64, even if our approach is Instruction Set agnostic—see Section 8 for more details). The top level of this hierarchy is represented by real world applications, which allow us to stress

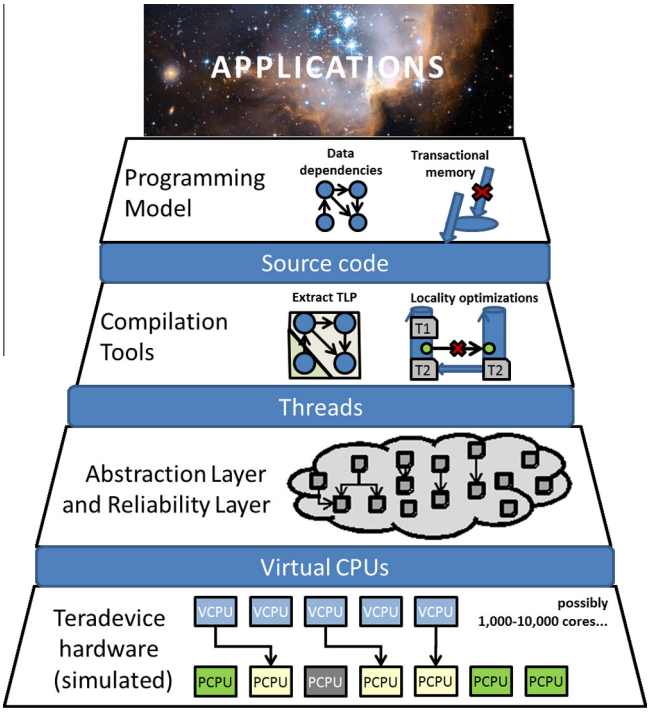

Fig. 1. The TERAFLUX transformation hierarchy.

the underlying teradevice hardware. In the TERAFLUX project, implicit parallelism refers to the set of constraints on the concurrent execution of threads, and the expression of these constraints in the source code. These constraints can be dependencies, atomic transactions, synchronization barriers, privatization attributes, memory layout and extent properties, and a wide variety of hints. An explicitly parallel program, on the other hand, is made of concurrency constructs making the thread creation, termination, and possibly some target-specific aspects of the execution explicit [10–12].

A dataflow oriented programming model allows expressing data dependencies among the concurrent tasks of an application. Such concurrent tasks can be subdivided even more—at lower levels—into *DataFlow Threads* (or *DF-Threads*), also simply referred as threads when clear from the context. Nevertheless, applications use large data structures with in-place updates, for efficient memory management [13–15] and copy avoidance. Such applications may require a mechanism to express the non-interference of concurrent updates to shared data. To meet such need, we selected Transactional Memory (TM), as the most promising programming construct and concurrency mechanism for specifying more general forms of synchronization among threads, while preserving the composability of parallel dataflow programs and promising a high level of scalability [16]. We achieve this by defining a specific layer for studying the integration between the TM and dataflow programming models [17–19].

Besides the programming model, implicit parallelism must be exploited by a compilation tool-chain [20–22], being able to convert dependencies and transactions, into scalable target-specific parallelism. It is also responsible for properly managing the inter-node communications and a novel memory model. Compiler effectiveness is guaranteed by the implementation of a generalization of the state-of-the-art algorithms to expose fine-grained dataflow threads from task-parallel OpenMP-, StarSs- or HMPP-annotated [23,24] programs. The algorithm generalization leverages a new dependence-removal technique to avoid the artificial synchronizations induced by in-place updates in the source program [25,26].

Our goals in designing an efficient compilation tool-chain are to capture the important data reuse patterns, to optimize locality and