Parallel Computing 40 (2014) 1–33

Contents lists available at ScienceDirect

Parallel Computing

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

Review

Carrying on the legacy of imperative languages in the future parallel computing era

Mohammad Reza Selim [a,*], Mohammed Ziaur Rahman [b,1]

[a] Dept. of Computer Sc. and Engineering, Shahjalal University of Science and Technology, 3114 Sylhet, Bangladesh
[b] Zifern Ltd, Kuala Lumpur 59200, Malaysia

ARTICLE INFO

Article history:
Received 25 March 2013
Received in revised form 28 January 2014
Accepted 1 February 2014
Available online 6 February 2014

Keywords:
Dataflow computing
Dataflow model
Parallel computing
High performance computing
Imperative language
Compiler

ABSTRACT

There has been a renewed interest in dataflow computing models in recent years of technology scaling. Potentiality of exploiting huge parallelism, with the expense of low power, simpler circuit, less silicon area, is the main characteristic of a dataflow model. Growing trends in housing large number of functional units in a single chip, making use of local clocks, reducing energy consumptions, avoiding global wires are the main reasons behind the resurgence of dataflow models. To program a dataflow machine, new architectures suggest *imperative languages* rather than functional type *dataflow languages* or parallel languages because this is the right way to make the new architectures popular among the general community. Although for several decades scientists have been working on how imperative languages can be used in dataflow models efficiently, there is no systematic review on those works. Existing reviews on dataflow paradigm mainly focus on the architectures. Although few papers review programming languages of dataflow architectures, their discussions are limited to only dataflow languages and visual programming languages which are fundamentally different from imperative languages. In this paper, we conduct a systematic review on those works that attempt to provide a way to use imperative languages in any type of dataflow architectures. Our survey of compilers and related architectures cover the aspects like translation mechanisms of program construct, their optimization techniques, memory ordering methods, program allocation and scheduling and special architectural features. We also present some of our observations and future research directions obtained by exploring the literature.

Contents

* Corresponding author.
E-mail addresses: selim@sust.edu (M.R. Selim), m.ziaur.rahman@ieee.org, zia@zifern.com (M.Z. Rahman).
URL: http://www.zifern.com (M.Z. Rahman).

[1] The author was a visiting senior lecturer at University of Malaya during when this research was carried out.

http://dx.doi.org/10.1016/j.parco.2014.02.001
0167-8191/

1. Introduction

Traditionally, the von Neumann computing model is treated as an inherent sequential model of computation [13]. The sequentiality comes from the fact that execution of the current instruction decides which instruction will be the next representative. This is the reason it is called a *control flow model*. In this model, we do not have a natural way to perform the execution of an instruction before the execution of the previous instruction, in the control flow path, is finished. Next instruction to be executed is pointed to and triggered by the program counter. Besides, a global updatable store, the media through which data is exchanged between instructions, is assumed to be present in the model. These two properties are the main bottlenecks to exploit parallelism in the control flow model. The superscalar processors [109] break the original control flow model in their architectures to gain efficiency. The multiprocessor architectures [42,69] based on the von Neumann computing model also break the flow of control with the help of explicit language statements/derivatives inserted by compilers or programmers. Despite the involvement of compilers and programmers, it has not been possible to reach to a performance level that was expected to by the early scientists. Nonetheless, the von Neumann computing model is the most popular model in the world.

Dataflow computing models [105,57,106,69], on the other hand, provide a natural way to initiate the execution of more than one instruction simultaneously. Unlike control flow models, no program counter or global updatable memory is used here. The execution is driven by the availability of the operands [94]. As soon as all the operands of an instruction is available, the instruction is issued, provided that the execution resource is available. In dataflow computing, parallelism is implicit and