Chapter 1
An Introduction to Controlflow and Dataflow Supercomputing

Miloš Kotlar
School of Electrical Engineering, University of Belgrade, Serbia

ABSTRACT

In the controlflow paradigm, based on the finite automata theory, one writes a program in order to control the flow of data through the hardware. In the dataflow paradigm, one writes a program in order to configure the hardware. Then, the question is what moves data through the hardware, if that is not a stored program? Ideally, in the dataflow paradigm, data get moved by the voltage difference between the system input and the system output, with flip-flops and register barriers in-between (FPGAs), or without these barriers (referred as ultimate dataflow), possibly but not necessarily, with the computing infrastructure based on the analog hardware. See the references for one specific viewpoint related to the subject.

THE CONTROLFLOW ESSENCE

The controlflow paradigm dates back to the times of von Neumann in 1940s (vonNeumann, 1951), when the first machines were implemented based on that paradigm, but its rapid popularity came up with the invention of microprocessors, especially those based on the RISC and MIPS concepts (Hennessy, 1982). In both cases, as already indicated, the essence is in the theory of finite automata.

The more recent advances include a number of architectural advances (e.g., Grujic, 1996; Milenkovic, 2000; Trobec, 2016), technology advances (e.g., Milutinovic1996), extensions of the paradigm (e.g., Babovic, 2016; Gavrilovska, 2010), extensive research in the domains of system software (e.g., Knezevic, 2000), tools for effective code developments (e.g., Trifunovic, 2015), and the ready to use library routines for a plethora of different applications, with a recent stress on Artificial Intelligence, Deep Learning, and Data Mining (e.g., Radivojevic, 2003). These advances span the time from 1990s till today, and are here presented briefly, stressing only the concepts of interest for this edited volume, which all originated from the laboratory that synergizes the contributing authors of this book.

DOI: 10.4018/978-1-7998-7156-9.ch001

Programs predominantly based on the transactional code are well suited for execution on machines implemented using a controlflow architecture. However, controlflow architectures are often times not the best suited for highly parallel code operating on big data, with low power, low volume, and high precision requirements. For such environments, accelerators are needed.

THE DATAFLOW ESSENCE

The dataflow paradigm dates back to the MIT research of professors Dennis and Arvind in 1970s, when the first machines were implemented based on that paradigm, but the machines with a really great potential for speed up and power savings started to appear with the research in and around Maxeler Technologies, in 1990s. In the early MIT research, the essence is in data that flow using standard finite automata hardware underneath, while in the Maxeler research (e.g., see an overview at Milutinovic, 2015), as well as in the early Purdue University research (e.g., Milutinovic, 1987), the essence is in mapping the algorithms into hardware, so that the dataflow process is fully implemented in hardware.

The more recent advances include a number of extensions of the paradigm (e.g., Jovanovic, 2012), extensive research in the domains of system software (e.g., see an overview at Milutinovic, 2015), tools for effective code developments (e.g., Trifunovic, 2015), and the ready to use library routines for large data volumes (e.g., Flynn, 2013). These advances span the time from 1990s till today, and are here presented briefly, stressing only the concepts of interest for this edited volume, which all originated from the laboratory that synergizes the contributing authors of this book.

Programs predominantly based on the highly parallel code are well suited for execution on machines implemented using a dataflow architecture and serving as accelerators. Such accelerators are especially well suited for highly parallel code operating on big data, with low power, low volume, and high precision requirements. For such environments, found in many applications covered in this book, accelerators are an ideal solution. Another type of environment in which the Dataflow paradigm could help is in applications based on approximate computing for low latency (e.g., Milutinovic1980).

CONCLUSION

The major conclusion of this introduction to controlflow and dataflow supercomputing is that the best results are obtained with hybrid computers that synergize the two paradigms. The controlflow part is best used as the host, while the dataflow part is best used as the accelerator. A number of emerging applications are best implemented on hybrid computers.

Further advances in controlflow, dataflow, and hybrid computing need the creativity that is closely coupled with realities in the domains of underlying technologies and emerging applications. One study of creativity utilized in the related research so for could be found, together with related activities in the dissemination domain, in (Blagojevic2017, Bankovic2020).

However, the major challenge is to map efficiently the most sophisticated emerging algorithms and applications onto a properly synergized infrastructure that combines dataflow and controlflow paradigms.