

Computer Communications and Networks

Series editor

A.J. Sammes

Centre for Forensic Computing

Cranfield University, Shrivenham Campus

Swindon, UK

The **Computer Communications and Networks** series is a range of textbooks, monographs and handbooks. It sets out to provide students, researchers, and non-specialists alike with a sure grounding in current knowledge, together with comprehensible access to the latest developments in computer communications and networking.

Emphasis is placed on clear and explanatory styles that support a tutorial approach, so that even the most complex of topics is presented in a lucid and intelligible manner.

More information about this series at <http://www.springer.com/series/4198>

Veljko Milutinovic • Jakob Salom • Dragan Veljovic
Nenad Korolija • Dejan Markovic • Luka Petrovic

DataFlow Supercomputing Essentials

Research, Development and Education



Springer

Veljko Milutinovic
University of Belgrade
Belgrade, Serbia

Dragan Veljovic
Motionlogic GmbH
Berlin, Germany

Dejan Markovic
University of Belgrade
Belgrade, Serbia

Jakob Salom
Serbian Academy of Sciences and Arts
Belgrade, Serbia

Nenad Korolija
University of Belgrade
Belgrade, Serbia

Luka Petrovic
University of Belgrade
Belgrade, Serbia

ISSN 1617-7975 ISSN 2197-8433 (electronic)
Computer Communications and Networks
ISBN 978-3-319-66127-8 ISBN 978-3-319-66128-5 (eBook)
DOI 10.1007/978-3-319-66128-5

Library of Congress Control Number: 2017951392

© Springer International Publishing AG 2017

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

In general, there are two major approaches to classical computing: control flow and dataflow. In the control flow approach to computing, the program micro-controls the flow of data through the hardware. In the dataflow approach to computing, the program configures the hardware; ideally, it is the voltage difference between the input and output of the hardware that micro-controls the flow of data through the hardware in the theoretically ideal dataflow machines (in reality, for now, this theoretical ideal has not been achieved yet).

Dataflow Taxonomy

According to Flynn's taxonomy, control flow machines could be of the SISD type (single instruction and single data) and of the MIMD type (multiple instructions and multiple data). The MIMD type is basically subdivided into multicores (e.g., Intel) and manycores (e.g., NVidia) and WSN (wireless sensor networks). The control flow approach is used in almost all computers of today.

According to taxonomies used in dataflow literature, the dataflow concept could be defined on a number of different levels of abstraction related to computing. The three levels relevant for this book are the software level (application software), the hardware level (digital hardware), and analog levels used for implementation of digital hardware (transistors and wires).

On the software level, the concept exists for a long time now, but became extremely popular only recently, since Google announced that it gave up MapReduce and switched to dataflow. Software dataflow is not a subject of this book. For the reader of this book, it is only important to understand that a number of different software approaches to dataflow could be defined and that the best hardware support for dataflow in software is dataflow in hardware.

On the hardware level, the concept exists for decades now, in three basic forms: (a) general purpose dataflow related to the early research at MIT, (b) special purpose

dataflow related to the early research in systolic arrays, and (c) the dataflow that accelerates loops, which is applicable to both special purpose and general purpose computing and whose effectiveness depends on the contribution of loop execution times to the overall execution time of the application or the multiscale dataflow (MSDF). The first and the second approach (a and b) are outside the scope of this book. The third approach (MSDF) offers the best potentials and is the focus of this book.

Maxeler

The MSDF approach could be implemented using (a) FPGAs (Altera or Xilinx or any other FPGA vendor) or (b) aFPGAs (asynchronous FPGAs) (a research concept exemplified by recent research at Caltech, Cornell, Yale, and others) or (c) SoG (sea-of-gates) that enables the most effective mapping of execution graphs onto the hardware infrastructure underneath, which combines analog communications and digital storage.

Examples in this book are based on the Maxeler MSDF concept. The Maxeler machines are currently implemented using FPGAs. However, Maxeler is not an FPGA company. It uses FPGAs as the least bad hardware infrastructure of all those currently available for the implementation of the MSDF concept. Therefore, FPGAs, looking from the Maxeler viewpoint, are a “necessary evil” that has to be used until a more appropriate hardware support reaches the production level. Although FPGAs do not allow the Maxeler MSDF concept to blossom fully, they are tolerated, since they, in spite of all their drawbacks, do allow the Maxeler concept to demonstrate superiority over the existing control flow approaches, mostly in the amount of computation per cubic foot and Watt.

The Maxeler MSDF concept covers relevant issues in three different aspects of software: (a) operating system, (b) compiler, and (c) application software. The first two issues (a and b) are outside the scope of this book. The focus of this book is on the research that could improve the overall concept and on the applications thereof.

Research Issues

The major inspiration for future research in dataflow computing of the MSDF type comes from the heritage of four different Nobel laureates. Their heritage includes not only scientific results but also observations that torch the light onto possible future research avenues.

Richard Feynman, in his book on lecture notes in computer science, commented that speed and power could be traded and also shared the observation that arithmetic and logic could be performed with power that converges to zero, while communi-

cating the results to the next usage point in memory or in another processor does consume power that could diverge to infinity for NP complete problems.

These observations mean that the ultimate goal of computer architects is to come up with an architecture that possibly does not move data at all, or if it has to move data around, which is unavoidable since the dimensions of arithmetic and logic units are non-zero, then the movements should go only over the shortest possible distances.

Ilya Prigogine, in his book on chaos, commented that injecting energy could decrease entropy of a system, which could be understood as the reminder of the fact that one could also define the entropy of a computing system, meaning that the lower entropy of such a system creates better conditions for the system software to do optimizations more effectively.

These observations led to the conclusion that the energy spent in separation of temporal and spatial data could help both the speedups and power reductions. Temporal data are those that get revisited over and over again (loop control variables, counter control variables, semaphores, etc.). Spatial data are those that get traversed, one by one or in strides, such as 4 by 4 and the like (vectors, matrices, complex data structures of any kind, etc.). If temporal and spatial data are separated, then a hybrid approach based on control flow and dataflow could become very effective, if the cranking of temporal data is somehow attributed to control flow and of spatial data to dataflow.

Daniel Kahneman, in his book on thinking fast and thinking slow, indicated about the importance of approximate computing. One possible explanation of the importance of approximate computing is through chess analogy. In chess, from time to time, we sacrifice something that is of importance, but not of ultimate importance, like the tower or queen, and by doing that we establish potential energy that enables us to achieve what is of ultimate importance—to win the game. In computing, that means that from time to time we can do computation with lower precision, which enables us to release resources that could be reinvested into a benefit that is of a much higher importance.

These observations bring one to a conclusion that the wisdom of Daniel Kahneman could and should be built into the compiler of dataflow machines, together with a set of mechanisms that enable one to select the tradeoffs of interest.

Andre Geim, in his research related to thin films, studied the relationships between latency (the speed at which a process could be done) and precision (with which the process achieves the final results). An interested reader could port these conclusions into the world of computing.

These observations, if implemented in the compiler of a dataflow machine, could enable users to trade latency and precision at compile time, which could be especially of importance for applications that require low latency, like those in online trading and those in defense.

Application Issues

One can port an application from control flow to dataflow by applying rules, but that will not bring major speedups, especially if the contribution of loops is negligible and the level of data reusability within loops is relatively low.

One can play with issues like input data choreography, changes or orders of operations without jeopardizing the integrity of results (commutation, association, distribution), changes in the floating-point precision or precision in general, and on the wise utilization of internal pipelines that the compiler generates, in the effort to create a consistent execution graph.

One can compare existing algorithms for the same application. Sometimes, the algorithm which is the best in control flow is the worst in dataflow and vice versa. One such example is sorting. Bitonic sorting is the worst in control flow and the best in dataflow.

One can think of new algorithms for new applications that did not exist before. Creating a new algorithm from scratch is a challenge, and it can turn into a success only and especially if it is done by an experienced dataflow programmer.

Conclusion

What was the goal and what was achieved? The goal was to underline what are the basic potentials for dataflow computing and to explain what types of research would lead to the ultimate goals: high speed, low power, high precision, and small sizes of computing equipment.

To whom may it be of benefit? The above is of benefit to those who plan to invest into further developments of dataflow computing, as well as to those who are eager to select the topics for their future research.

What are the new open research problems? The new goals could be classified as: (a) developing a new type of sea-of-gates that would enable a more effective mapping of the execution graphs onto the hardware infrastructure, (b) developing new aspects of the operating system that would enable the dataflow paradigm to become more effective, (c) implementing into the compiler all the wisdoms of the four Nobel laureates (Feynman, Prigogine, Kahneman, and Geim) who served as the major sources of inspiration, and (d) working on new algorithms for existing and new applications that would enable the paradigm to demonstrate its full strengths (appgallery.maxeler.com).

Belgrade, Serbia
July 2017

Veljko Milutinovic

Contents

Part I Research

1	Maxeler AppGallery Revisited	3
1.1	Introduction	3
1.2	Maxeler AppGallery	4
1.2.1	Data Analytics Category	4
1.2.2	Engineering Category	7
1.2.3	Low Latency Transaction Processing Category	10
1.2.4	Networking Category	11
1.2.5	Science Category	14
1.2.6	Security Category	17
	References	18
2	Discrepancy Reduction Between the Topology of Dataflow	
	Graph and the Topology of FPGA Structure	19
2.1	Introduction	19
2.2	Dataflow Graph	20
2.3	Getting to Accelerated Application: Maxeler Way	20
2.4	Simulation Debugging	21
2.4.1	Simulation Watches	21
2.4.2	Simulation printf	23
2.5	Hardware Debugging	24
2.5.1	DFE printf	25
2.6	Advanced Debugging	26
2.6.1	Introduction	26
2.6.2	Kernel Halted on Input	27
2.6.3	Kernel Halted on Output	29
2.6.4	Stream Status Blocks	30
2.6.5	Deadlock	31
2.7	Effects of Inconsistency Between Simulation and Hardware	37
2.7.1	Uninitialized Elements	37
2.7.2	Race Condition	37

2.8	Embedded DFE Optimizations	38
2.8.1	Kernel Optimizations	38
2.8.2	Manager Optimizations	42
2.9	Global DFE Optimization Practices: Getting Maximum Performance	44
2.9.1	Introduction	44
2.9.2	Dataflow Computing Strategy	45
2.9.3	Fitting Procedure	45
2.9.4	How to Make it Fit?	47
2.9.5	Optimizing Memory Bound Applications: Data Size	50
2.9.6	Optimizing Compute Performance: Clock Frequency	51
2.10	The More We Know About the Universe, It Is Harder to Believe in Determinism	55
2.10.1	Place and Route Non-deterministic Process	56
2.11	Topology of the Execution Graph	56
2.11.1	Simulated Annealing	56
2.11.2	Cost Tables	57
2.11.3	Constraints	57
2.12	Topology Inconsistencies	61
2.12.1	Consequences	62
2.13	Compiling Two-Input and Tri-Input Adders	62
2.13.1	Initial Compiling	63
2.13.2	Optimization TriAD Graph Pass	64
2.14	Conclusion	65
	References	65

Part II Development

3	Polynomial and Rational Functions	69
3.1	Introduction	69
3.1.1	Term of a Real Number	70
3.1.2	Term of a Complex Number	70
3.1.3	The Term of Polynomial	70
3.1.4	The Term of Rational Functions	71
3.2	Existing Solutions	72
3.3	Essence of the Dataflow Implementation	73
3.4	Details of the Implementation	74
3.4.1	Lexical Analyzer	90
3.5	Syntax Analyzer	92
3.5.1	Syntax	96
3.6	Demonstrating the Calculation of a Rational Function's Value	96
3.7	Some Performance Indicators	97
3.8	Conclusions	104
	References	105

4 Transforming Applications from the Control Flow to the Dataflow Paradigm	107
4.1 Introduction	107
4.1.1 Potentials for Improving Computational Efficiency	107
4.1.2 Potentials for Reducing Power Consumption	111
4.2 Problem Description	112
4.3 Available Dataflow Applications and Technology	112
4.4 Transforming Applications from Control Flow to Dataflow	113
4.5 Analysis of Potentials	116
4.5.1 Huxley Muscle Model	117
4.5.2 Network Sorting	121
4.6 Conclusions	127
References	128
 Part III Education	
5 Mini Tutorial	133
5.1 Introduction	133
5.2 Dataflow Programming	134
5.2.1 Application Types	135
5.2.2 Educational Value	135
5.2.3 Visual Versus Textual	136
5.3 Installing the Virtual Machine and Starting the Integrated Development Environment	137
5.3.1 Installing the Virtual Machine	137
5.3.2 Integrated Development Environment	139
5.4 First Maxeler Program: “Hello World”	139
5.5 Testing the “Hello World” Program	142
5.6 Complex Maxeler Program: “Moving Average”	143
5.7 Tutorials on YouTube	146
References	146
 Index	149