Principles of High-Performance Processor Design

Junichiro Makino

Principles of High-Performance Processor Design

For High Performance Computing, Deep Neural Networks and Data Science

Springer

Junichiro Makino
Kobe University
Kobe, Hyogo, Japan

ISBN 978-3-030-76870-6 ISBN 978-3-030-76871-3 (eBook)
https://doi.org/10.1007/978-3-030-76871-3

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

The future cannot be predicted, but futures can be invented. — Dennis Gabor

Preface

In this book, I tried to theorize what I have learned from my experience of developing special- and general-purpose processors for scientific computing. I started my research career as a graduate student in astrophysics, studying the dynamical evolution of globular clusters. The research tool was the N-body simulation, and it was (and still is) important to make simulations faster so that we can handle larger number of stars. I used vector supercomputers such as Hitac S-810, Fujitsu VP-400, NEC SX-2, Cray X-MP, Cyber 205, and ETA-10, and also tried parallel computers such as TMC CM-2 and PAX. Around the end of my Ph.D. course, my supervisor, Daiichiro Sugimoto, started the GRAPE project to develop special-purpose computers for astrophysical N-body simulations, and I was deeply involved in the development of numerical algorithms, hardware, and software. The GRAPE project is a great success, with hardware achieving 10–100 times better price- and watt-performance compared to general-purpose computers at the same time and used by many researchers. However, as semiconductor technology advanced into deep-submicron range, the initial cost of development of ASICs had become too high for special-purpose processors. In fact, it has become too high for most general-purpose processors, and that was clearly the reason why first the development of parallel computers with custom processors and then the development of almost all RISC processors were terminated. Only x86 processors from Intel and AMD had survived. (Right now, we might be seeing the shift from x86 to Arm, though) The x86 processors in the 2000s were not quite efficient in the use of transistors or electricity. Nowadays, we have processors with very different architectures such as GPGPUs and Google TPU, which are certainly more efficient compared to general-purpose x86 or Arm processors, at least for a limited range of applications. I also was involved in the development of a programmable SIMD processor, GRAPE-DR, in 2000s, and more recently a processor for deep learning, MN-Core, which was ranked #1 in the June 2020 and June 2021 Green500 lists.

In this book, I discuss how we can make efficient processors for high-performance computing. I realized that we did not have a widely accepted definition of the efficiency of a general-purpose computer architecture. Therefore, in the first three chapters of this book, I tried to give one possible definition, the ratio between

the minimum possible energy consumption and the actual energy consumption for a given application using a given semiconductor technology. In Chapter 4, I overview general-purpose processors in the past and present from this viewpoint. In Chapter 5, I discuss how we can actually design processors with near-optimal efficiencies, and in Chapter 6 how we can program such processors. I hope this book will give a new perspective to the field of high-performance processor design.

This book is the outcome of collaborations with many people in many projects throughout my research career. The following is an incomplete list of collaborators: Daiichiro Sugimoto, Toshikazu Ebisuzaki, Yoshiharu Chikada, Tomoyoshi Ito, Sachiko Okumura, Shigeru Ida, Toshiyuki Fukushige, Yoko Funato, Hiroshi Daisaka, and many others (GRAPE, GRAPE-DR, and related activities); Piet Hut, Steve McMillan, Simon Portegies Zwart, and many others (stellar dynamics and numerical methods); Kei Hiraki (GRAPE-DR and MN-Core); Ken Namura (GRAPE-6, GRAPE-DR, and MN-Core); Masaki Iwasawa, Ataru Tanikawa, Keigo Nitadori, Natsuki Hosono, Daisuke Namekata, and Kentaro Nomura (FDPS and related activities); Yutaka Ishikawa, Mitsuhisa Sato, Hirofumi Tomita, and many others (Fugaku development); Michiko Fujii, Takayuki Saito, Junko Kominami, and many others (stellar dynamics, galaxy formation, and planetary formation simulation on large-scale HPC platforms); Takayuki Muranushi and Youhei Ishihara (Formura DSL); many people from PFN (MN-Core); many people from PEZY Computing; and ExaScaler (PEZY-SC). I would like to thank all the people above. In addition, I'd like to thank Miyuki Tsubouchi, Yuko Wakamatsu, Yoshie Yamaguchi, Naoko Nakanishi, Yukiko Kimura, and Rika Ogawa for managing the projects I was involved. I would also like to thank the folks at Springer for making this book a reality. Finally, I thank my family, and in particular my partner, Yoko, for her continuous support.

Kobe, Japan Junichiro Makino

Contents

1 Introduction 1
References 5

2 Traditional Approaches and Their Limitations 7
2.1 History 7
2.1.1 CDC 6600 and 7600 7
2.1.2 Cray-1 13
2.1.3 The Evolution of Vector Processors 17
2.1.4 Lessons from the History of Vector-Parallel Architecture 21
2.1.5 The Evolution of Single-Chip Processors 21
2.1.6 The Impact of Hierarchical Cache 24
2.1.7 Alternatives to Cache Memories 28
2.2 The Need for Quantitative Approach 31
2.3 What Is Measured and What Is Not 34
References 35

3 The Lower Limit of Energy Consumption 37
3.1 Range of Applications We Consider 37
3.1.1 Structured Mesh 39
3.1.2 Unstructured Mesh 40
3.1.3 Particles 40
3.1.4 Random Graphs 41
3.1.5 Dense Matrices 41
3.1.6 Miscellanies 41
3.1.7 Distribution of Application Types 42
3.2 Definition of Efficiency 42
3.3 Structured Mesh 46
3.3.1 Choice of the Numerical Methods 46
3.3.2 The Design of An Ideal Processor Architecture for Structured Mesh Calculations 48
3.4 Unstructured Mesh 48

3.5 Particles 54
3.5.1 The Overview of Particle-Based Methods 54
3.5.2 Short-Range Interactions 55
3.5.3 Long-Range Interactions 56
3.6 Random Graphs 57
3.7 Dense Matrices 58
3.8 Summary 59
References 61

4 Analysis of Past and Present Processors 65
4.1 CDC 6600 65
4.2 Cray-1 and Its Successors 67
4.3 x86 Processors 69
4.3.1 i860 69
4.3.2 From Pentium to Skylake 70
4.4 NEC SX-Aurora and Fujitsu A64fx 74
4.5 SIMD Supercomputers—Illiac IV and TMC CM-2 79
4.5.1 Illiac IV 79
4.5.2 CM-2 80
4.5.3 Problems with Large-Scale SIMD Processors 82
4.6 GPGPUs 85
4.7 PEZY Processors and Sunway SW26010 87
4.7.1 PEZY Processors 87
4.7.2 Sunway SW26010 88
4.8 Conclusion 90
References 92

5 "Near-Optimal" Designs 95
5.1 The Special-Purpose Designs: GRAPE Processors 95
5.2 The Baseline Design: GRAPE-DR 100
5.2.1 Design Concept and Architecture 100
5.2.2 The Efficiency 105
5.2.3 Software 108
5.3 Functions Necessary to Widen Application Area 110
5.3.1 Particles 111
5.3.2 Dense Matrices 114
5.3.3 Other Applications 118
5.4 An Extreme for Deep Learning: MN-Core/GRAPE-PFN 120
5.5 A "General-Purpose" Design 121
5.5.1 On-Chip Network for Sorting 121
5.5.2 Off-Chip DRAM Access 123
5.5.3 Chip-to-Chip Communications Network for Deep Learning and Unstructured-Mesh Calculations 124
5.5.4 Support for FP64, FP32 and FP16 or Other Mixed-Precision Operations 124

5.6 The Reference SIMD Processor 125
5.6.1 PE 127
5.6.2 BM 130
5.6.3 TBM 131
5.6.4 DRAM Interface 132
5.6.5 Host Data Interface 132
5.6.6 Instruction Fetch/Issue 132
References 133

6 Software 135
6.1 Traditional Approaches 135
6.2 How Do We Want to Describe Applications? 138
6.2.1 Structured Mesh 138
6.2.2 Unstructured Mesh 141
6.2.3 Particles 141
6.2.4 Dense Matrices 143
6.3 Summary 144
References 144

7 Present, Past and Future 147
7.1 Principles of High-Performance Processor Design 147
7.2 The Current Practice 148
7.3 Our Past 149
7.4 GPGPUs and Deep Learning Processors 151
7.5 The Future 153
References 155

Index 157

Acronyms

ASIC	Application-specific integrated circuit
B/F	Bytes per flop
BB	Broadcast block
BM	Broadcast memory
CISC	Complex instruction-set computer
CG	Conjugate gradient (method)
CNN	Convolutional neural network
CPE	Computing processing element of sunway SW26010
DCTL	Direct coupled transistor logic
DEM	Distinct (or discrete) element method
DDM	Domain decomposition method
DDR	Double data rate (DRAM)
DMA	Direct memory access
DSL	Domain-specific language
EFGM	Element-free Galerkin method
FEM	Finite element method
FLOPS	Floating-point operations per second
FMA	Floating-point multiply and add
FMM	Fast multipole method
FPGA	Field-programmable gate array
FPU	Floating-point arithmetic unit
GaAs	Gallium arsenide
GPGPU	General-purpose computing on graphics processing units
HBM	High-bandwidth memory
HPC	High-performance computing
HPL	High-performance Linpack benchmark
ISA	Instruction-set architecture
MIMD	Multiple instruction streams, multiple data streams
MPE	Management processing element of sunway SW26010
MPS	Moving particle simulation
NUMA	Non-uniform memory access

OoO	Out-of-order (execution)
PCI	Peripheral component interconnect
PCIe	PCI express
PE	Processing element
RISC	Reduced instruction-set computer
SERDES	Serializer/deserializer
SIMD	Single instruction stream, multiple data dreams
SMT	Simultaneous multithreading
SPH	Smoothed particle hydrodynamics
SVE	Scalable vector extensions