Jianbin FANG, Peng ZHANG, Chun HUANG, Tao TANG, Kai LU, Ruibo WANG, Zheng WANG, 2023. Programming bare-metal accelerators with heterogeneous threading models: a case study of Matrix-3000. *Frontiers of Information Technology & Electronic Engineering*, 24(4):509-520.
https://doi.org/10.1631/FITEE.2200359

Programming bare-metal accelerators with heterogeneous threading models: a case study of Matrix-3000

Key words: Heterogeneous computing; Parallel programming models; Programmability; Compilers; Runtime systems

Corresponding author: Chun HUANG
E-mail: chunhuang@nudt.edu.cn
ORCID: https://orcid.org/0000-0002-0317-8192

Motivation

1. Heterogeneous many-core processors are now commonplace in computer systems, but the potential of accelerators can be unlocked only if the software can make good use of the hardware.
2. Writing and optimizing code for many-core accelerators is challenging for many application developers, because the current hardware architecture and programming model of accelerators significantly differ from those of the conventional multi-core processors.
3. Our work will focus on addressing the programming issues of Matrix-3000 and similar accelerators.

Main idea

1. We share our experience of designing and implementing a programming model and its supporting compiler and runtime system for the Matrix-3000 (also coined as MT-3000) heterogeneous many-core accelerator.

Method

1. To support software development for MT-3000, we have developed a piece of full-stack system software from scratch, where we focus on two programming modules: the hthreads low-level programming interface and the MOCL3 OpenCL compiler.

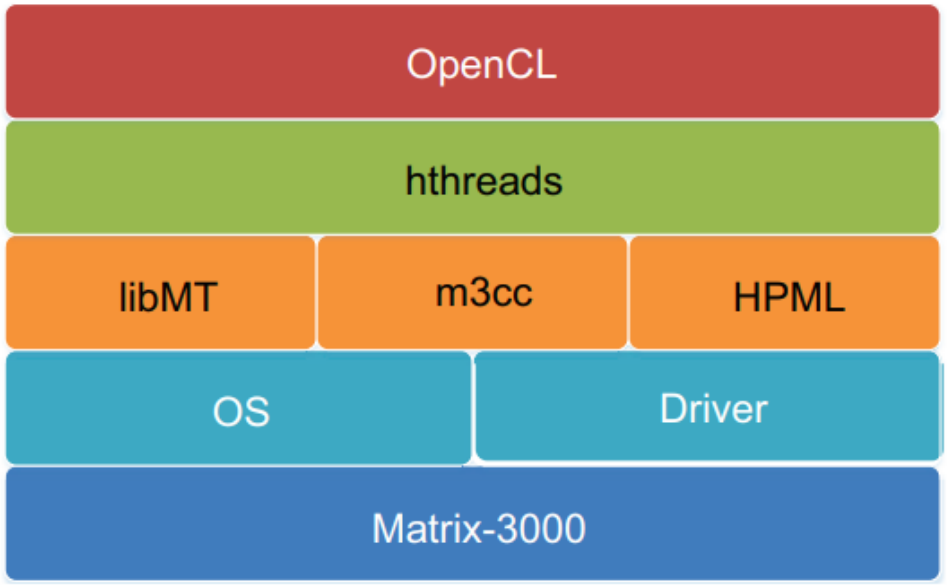


Fig. 3 The MT-3000 programming stack (OS: operating system)

Method (Cont'd)

2. We develop a low-overhead high-availability heterogeneous programming interface (hthreads). At the core of hthreads is a heterogeneous threading model introduced for the bare-metal MT-3000 accelerator.

- The hthreads interface consists of the general-purpose (GP) zone side application programming interfaces (APIs) and the acceleration (ACC) zone side APIs.
- hthreads exposes as many performance-related architecture features as we can, aiming to fully tap its computing potentials.
- We introduce the threading model to hide the metal-related uses such as the native direct memory access (DMA) usage.

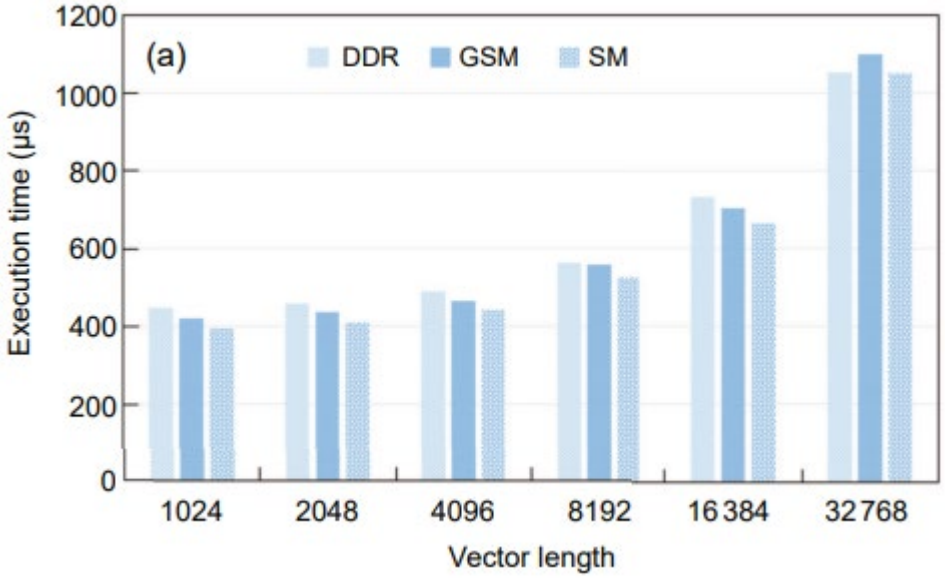
Method (Cont'd)

3. At a higher level, we provide the implementation of the OpenCL standard parallel programming interface (MOCL3) for the MT-3000 architecture.

- MOCL3 follows the programming specification of OpenCL 1.2.
- While ensuring to explore the computing potential of MT-3000, MOCL3 can be effectively compatible with OpenCL legacy code and significantly improve programmability.

4. With these two programming interfaces, we aim to achieve a balance among performance, programmability, and portability for our bare-metal accelerator.

Major results



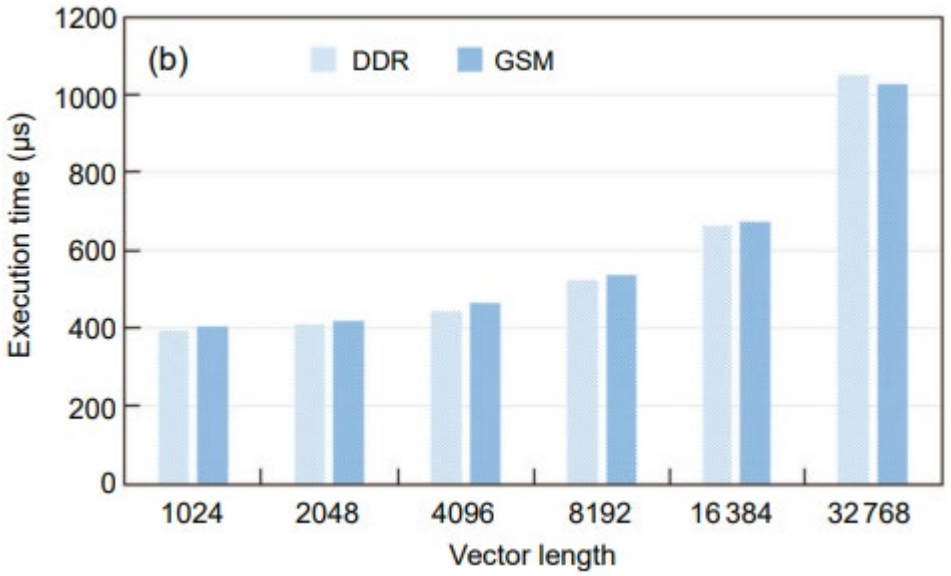


Fig. 10 Performance comparison of different design choices: (a) memory layout (the performance obtained from placing stack on SM over DDR and GSM); (b) code layout (performance comparison between placing code on DDR and GSM). SM: scalar memory; DDR: double data rate; GSM: global shared memory

Major results (Cont'd)

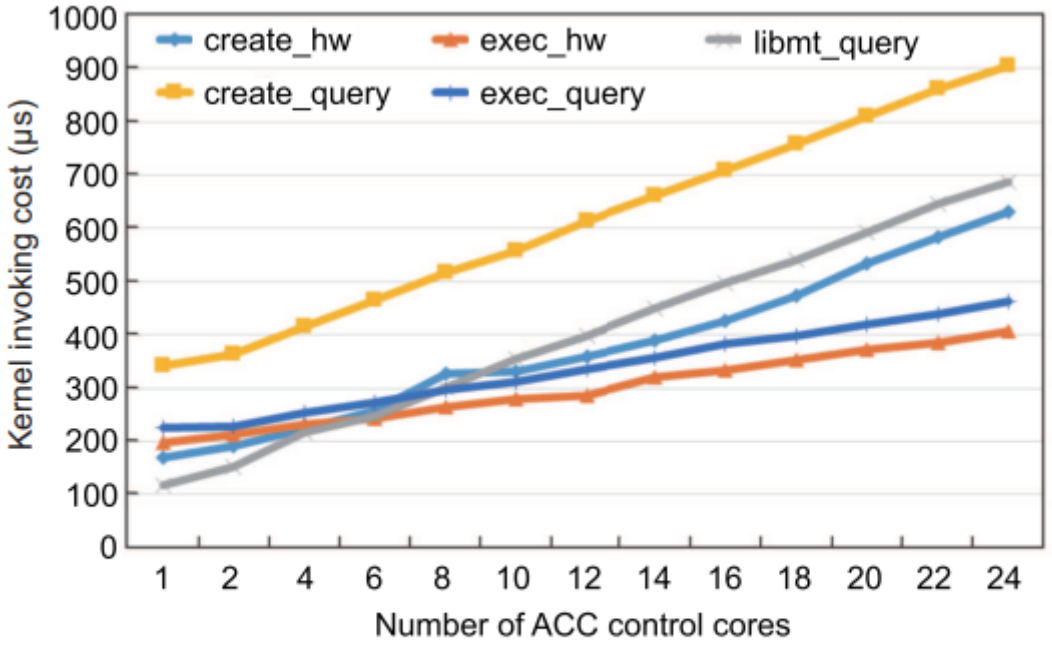


Fig. 11 Invoking cost of using hardware interrupt over querying

Major results (Cont'd)

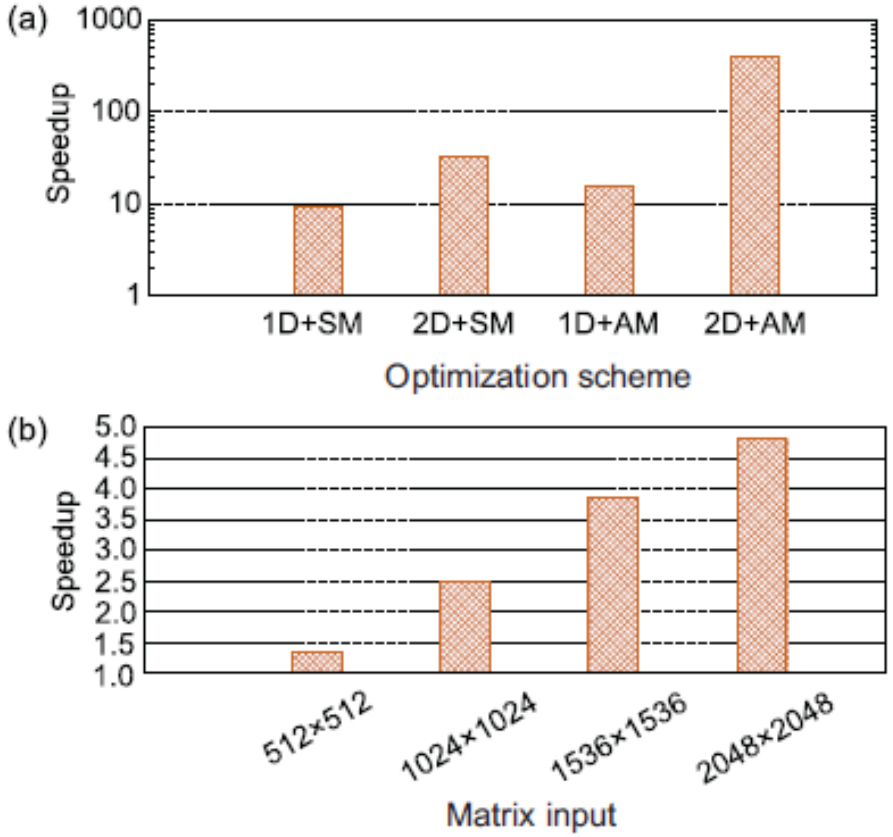


Fig. 12 Speedup over baseline implementations for matrix multiplications: (a) with various optimizations in hthreads; (b) with local memory in MOCL3

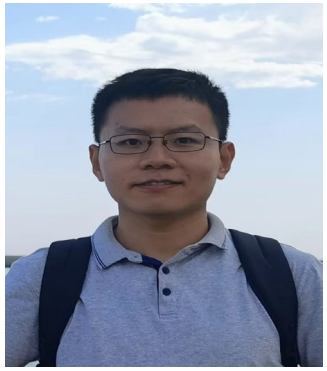
Conclusions

1. We have presented the design and implementation of the programming and compiler tools for the Matrix-3000 accelerator.

2. Given the complex memory hierarchy and processor core organization of Matrix-3000, the use of microarchitecture design in a way that enables complete realization of potential hardware performance depends on the effectiveness of the customized software employed on the device.

3. We share our experience on how a low-level threading-based programming interface can be developed to support the high-level OpenCL programming standard.

Jianbin FANG is an assistant professor in computer science at National University of Defense Technology (NUDT). He received his PhD degree from Delft University of Technology in 2010. His research interests include parallel programming for many-cores, parallel compilers, performance modeling, and scalable algorithms. He is a member of CCF.



Peng ZHANG is an assistant professor in computer science at National University of Defense Technology (NUDT). He received his PhD degree in computer science from NUDT in 2020. His research interests include heterogeneous programming, and performance optimization of parallel programs and compilers.

Chun HUANG is a full processor in computer science at National University of Defense Technology (NUDT). Her research interests are high-performance computing, system software, parallel compilers, parallel programming, performance optimization, and high-performance math libraries.

Tao TANG is now an associate processor in computer science at NUDT. He received his BSc, MSc, and PhD degrees all from NUDT. His research interests are compilers, parallel programming, and high-performance computing.

Kai LU is a full professor in computer science at NUDT. His research interests include high performance computing, parallel programming, operating system, and security.

Ruibo WANG received the BS and PhD degrees in computer science from National University of Defense Technology (NUDT) in 2003 and 2011, respectively. He is now a full professor at NUDT. His research interests include high performance computing and operating system.

Zheng WANG is currently a (full) professor of Intelligent Software Technology at the School of Computing at the University of Leeds. He received his PhD degree in computer science from The University of Edinburgh in 2011. From 2005 to 2007, he worked as an R&D engineer in IBM China. His research focus is in the areas of parallel compilers, runtime systems, systems security, and the application of machine learning to tackle the challenging optimization problems within these areas. He received three best paper award for his work on machine learning based compiler optimization (PACT'10, CGO'17, and PACT'17).