IPSJ Transactions on System LSI Design Methodology Vol.12 46–49 (Feb. 2019)

[DOI: 10.2197/ipsjtsldm.12.46]

Short Paper

An OpenCL-based Software Framework for a Heterogeneous Multicore Architecture on Zynq-7000 SoC

Takafumi Miyazaki[1,a)] Shunsuke Takai[1] Ittetsu Taniguchi[2,b)] Hiroyuki Tomiyama[1,c)]

Received: May 31, 2018, Revised: September 7, 2018,
Accepted: October 22, 2018

Abstract: This paper presents an OpenCL-based software framework which we have developed for a heterogeneous multicore architecture on Zynq-7000 SoC. In this work, the heterogeneous architecture is designed with two hard-macro Cortex-A9 cores and two soft-macro MicroBlaze cores. A major advantage of our OpenCL framework is that it can execute OpenCL kernel programs in three ways. Experiments show the usefulness of the OpenCL framework.

Keywords: heterogeneous multicore, OpenCL, FPGA, hard core, soft core, Zynq-7000 SoC

1. Introduction

Many of recent FPGA devices, such as Xilinx Zynq-7000 series, are pre-equipped with multiple CPU cores as hard-macros. Such devices are often called programmable systems-on-chip (PSoCs). In many cases of embedded system design with PSoCs, most functionalities are implemented in software running on the hard-cores, while some computationally-expensive functionalities are implemented in hardware accelerators on the FPGA fabric of PSoCs. Another design alternative for performance improvement with PSoCs is to develop heterogeneous multicores by implementing soft-cores (such as Xilinx MicroBlaze and Intel Nios-2) on the FPGA fabric in addition to the pre-equipped hard-cores.

The goal of this work is development of a parallel computing software framework for heterogeneous multicores on FPGAs. In this work, we have developed an OpenCL-based software framework for a heterogeneous multicore architecture on Xilinx Zynq SoCs. The contribution of this work is that our OpenCL framework supports three methods for load balancing on the heterogeneous cores. To our knowledge, this is the first study which develops an OpenCL-based software framework for heterogeneous multicore architecture on Zynq SoCs.

2. Preliminaries

2.1 Zynq-7000 SoC

A Zynq-7000 SoC consists of two parts, i.e., *Processing System* (*PS*) and *Programmable Logic* (*PL*). The PS part is implemented as hard-macros and includes dual Cortex-A9 cores and a set of peripherals. The PL part is an FPGA fabric. Application software and OS run on PS, and customized logics are implemented and executed on PL.

In this work, we use Digilent's ZYBO board, which is equipped with Zynq XC7Z010.

2.2 OpenCL

OpenCL is a standardized software framework for heterogeneous parallel-computing systems. OpenCL assumes two kinds of processing elements, i.e., *host* and *device*. A device consists of multiple *compute units* (*CUs*). In case of GPGPU, the host core corresponds to a main CPU, while the CUs correspond to GPU cores. An OpenCL program consists of two kinds of program fragments, i.e., a *host* program running on a host core and *kernel* functions running on device cores. A kernel function is typically executed on device cores in a data-parallel manner. The data is partitioned into a number of small pieces, called *work-items*. The device cores execute the same kernel function on different work-items.

3. The Heterogeneous Multicore Architecture

We have designed a heterogeneous multicore architecture which has totally four CPU cores by implementing dual MicroBlaze soft-cores in PL in addition to dual Cortex-A9 hard-cores in PS. The overall block diagram of our architecture is shown in the right side of **Fig. 1**. The host in Fig. 1 includes dual Cortex-A9

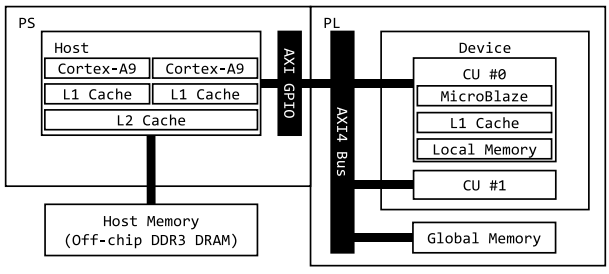


Fig. 1 An overview of a heterogeneous multicore architecture for Zynq-7000 SoC.

[1] Graduate School of Science and Engineering, Ritsumeikan University, Kusatsu, Shiga 525–8577, Japan
[2] Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565–0871, Japan
[a)] takafumi.miyazaki@tomiyama-lab.org
[b)] i-tanigu@ist.osaka-u.ac.jp
[c)] ht@fc.ritsumei.ac.jp

cores, and the device consists of dual MicroBlaze cores which are connected to an AXI4 bus.

As shown in the left side of Fig. 1, the host contains dual Cortex-A9 hard-cores. A Cortex-A9 core has an L1 cache and two Cortex-A9 cores share an L2 cache. The L1 cache is composed of an instruction cache (32 KB) and a data cache (32 KB). The size of the L2 cache is 512 KB. Programs and data for the host are stored in a host memory implemented by off-chip DDR3 DRAM. The size of the DRAM is 512 MB.

As shown in the left side of Fig. 1, there are two Compute Units (CUs) in the device. A CU is composed of a soft-macro MicroBlaze core, an L1 cache and a local memory. The L1 cache is comprised of an instruction cache (32 KB) and a data cache (32 KB). The size of a local memory is 32 KB.

The on-chip global memory is accessible by both Cortex-A9 and MicroBlaze cores. The global memory is used for communication between the host and the device. The size of the global memory is 32 KB.

4. The OpenCL-based Software Framework

We have ported PetaLinux on the host of our heterogeneous multicore architecture. The Linux is booted from SD card on the FPGA board. Also, we have developed an OpenCL-based framework. We have significantly modified and extended the RuCL framework [1], [2] in order to fully utilize the parallelism of our heterogeneous architecture. Specifically, our new OpenCL-based framework supports three methods to execute OpenCL programs. Programmers can select one of the three methods at the compilation stage.

As explained in Section 2.2, an OpenCL program is composed of a host program and kernel programs. In our framework, the host program runs on the host part of the architecture. The kernel programs are executed in either of the three ways, which are named Only_MicroBlaze, Only_Cortex and Cortex_MicroBlaze. The three methods are illustrated in **Fig. 2**.

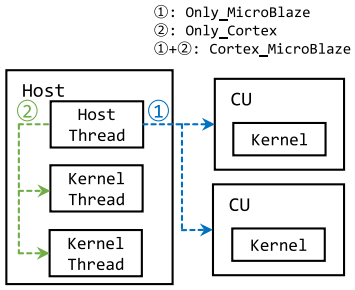


Fig. 2 An overview of three execution methods.

4.1 The Only_MicroBlaze Method

With the Only_MicroBlaze method, kernel programs run on two CUs (MicroBlaze cores), and work-items are statically distributed to the two CUs. In our system, Linux is not running on the CUs. Instead, our in-house runtime software is linked to the OpenCL kernel code. The runtime software receives activation signals from the host program, executes the kernel programs, and sends completion signals to the host. Communication and synchronization between the host and CUs are performed through the off-chip DRAM.

Conceptually, the Only_MicroBlaze method is similar to popular OpenCL frameworks for GPGPU in the sense that kernel programs are executed on CUs only and are not executed on the host CPUs.

4.2 The Only_Cortex Method

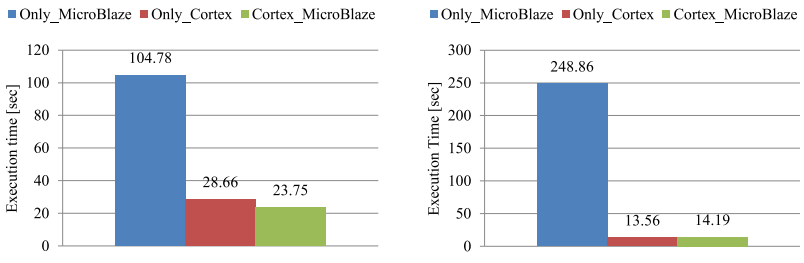
With the Only_Cortex method, both a host program and kernel programs are executed on the host (i.e., dual Cortex-A9 cores). CUs do not execute any kernel program. The pthread library is used to execute the kernel programs in parallel on the host. Since our architecture has two Cortex-A9 cores, our OpenCL framework creates two kernel threads. Work-items are statically allocated to the two threads.

4.3 The Cortex_MicroBlaze Method

With the Cortex_MicroBlaze method, both the host and the CUs run kernel programs. Work-items are dynamically allocated to two Cortex-A9 cores and two MicroBlaze cores.

5. Preliminary Experiments

In order to evaluate the usefulness of our OpenCL framework, we have conducted experiments using two OpenCL benchmark programs. The benchmark programs are Black-Scholes and Gaussian from the BEMAP benchmark suite developed in industry [3]. We ran the Black-Scholes and Gaussian with three configurations of kernel execution and measured execution times. The experiments are conducted on an actual FPGA board, i.e., Digilent's ZYBO board. The heterogeneous multicore architecture presented in Section 3 is implemented on the board, where the Cortex-A9 hard-cores and MicroBlaze soft-cores operate at 650 MHz and 100 MHz, respectively. Through the experiments, we have confirmed that our OpenCL framework with the three execution methods works correctly for the benchmark programs



(a) Black-Scholes (b) Gaussian

Fig. 3 Comparison of three execution methods.

on the actual FPGA board.

The execution time of Black-Scholes is shown in **Fig. 3** (a). This graph shows that Cortex_MicroBlaze is the fastest and Only_MicroBlaze is the slowest. The result indicates that load balancing between the host and the device is nicely done with the Cortex_MicroBlaze method.

The execution time of Gaussian is shown in Fig. 3 (b). The graph shows that Only_Cortex is the fastest and Only_MicroBlaze is the slowest. In this program, Cortex_MicroBlaze is slower than Only_Cortex. This is because of the communication overhead between the host and the device. The communication overhead is larger than the benefit of the parallel execution on both the host and the device.

6. Related Work

Kondo et al. developed a 128-core NoC architecture and implemented it on 16 FPGA boards [4]. Mori developed and implemented a 16-core NoC architecture on an FPGA [5]. Takenae et al. developed a bus-based 32-core architecture on an FPGA [6]. Their manycore architectures are homogeneous in a sense that the cores have the same instruction set. They implemented their architectures on FPGA without hard-macro CPU cores.

Past studies on heterogeneous multi/manycore architectures on FPGA with pre-equipped hard-cores include Refs. [7], [8] and [9]. However, Refs. [7] and [8] focus on hardware architectures and do not describe software frameworks. In Ref. [9], the authors developed a software framework based on SysML and Java.

This paper is an extended version of our previous work in Refs. [1] and [2]. In Ref. [1], we developed a lightweight OpenCL framework, named RuCL, for homogeneous multicore processors for embedded systems, and tested RuCL on Altera's Cyclone V SoC device which embeds two Cortex-A9 cores as hard macros. Although the device has FPGA fabrics, we did not use the FPGA part in Ref. [1]. In essence, the work in Ref. [1] corresponds to the Only_Cortex method of this paper. In Ref. [2], we designed and implemented a heterogeneous multicore architecture on Cyclone V SoC. We implemented four Nios-2 soft-cores on the FPGA part of the device, and connected them with the Cortex-A9 hard-cores. Then, we extended the RuCL framework for the heterogeneous multicores. With the extended RuCL, OpenCL kernels are executed on both Cortex-9 and Nios-2 cores for dynamic load-balancing. The work in Ref. [2] corresponds to the Cortex_MicroBlaze method of this paper. In this paper, we have further extended the RuCL framework in such a way that OpenCL kernels are executed on the soft-cores only, i.e., the Only_MicroBlaze method. Furthermore, we have integrated the three execution methods into the single RuCL framework so that users can choose the kernel execution method out of the three at compilation time.

7. Conclusions

In this work, we have developed an OpenCL framework for a heterogeneous multicore architecture on Zynq-7000 SoC, and confirmed the usefulness of our OpenCL framework using two OpenCL benchmark programs. In future, we plan to evaluate our system with further benchmark programs. Considering the results, we will improve our system in the both aspects of hardware and software.

Acknowledgments This work is in part supported by KAKENHI 15H02680.

References

[1] Takai, S., Nishiyama, N., Taniguchi, I. and Tomiyama, H.: A Lightweight OpenCL Framework for Embedded Multicore Processors, *Proc. International Technical Conference on Circuits/Systems, Computer and Communications* (*ITC-CSCC*), pp.355–357 (2015).

[2] Takai, S., Taniguchi, I., Tomiyama, H. and Parameswaran, S.: An OpenCL Framework for FPGA-based Heterogeneous Multicore Architecture, *Proc. International Technical Conference on Circuits/Systems, Computers and Communications* (*ITC-CSCC*), pp.443–444 (2016).

[3] Ardila, Y., Kawai, N., Nakamura, T. and Tamura, Y.: Support Tools to Porting Legacy Applications to Multicore, *Proc. Asia and South Pacific Design Automation Conference* (*ASP-DAC*), pp.568–573 (2013).

[4] Kondo, M., Nguyen, S.T., Hirao, T., Soga, T., Sasaki, H. and Inoue, K.: SMYLEref: A Reference Architecture for Manycore-Processor SoCs, *Proc. Asia and South Pacific Design Automation Conference* (*ASP-DAC*), pp.561–564 (2013).

[5] Mori, H. and Kise, K.: Design and Performance Evaluation of a Manycore Processor for Large FPGA, *Proc. International Symposium on Embedded Multicore/Manycore SoCs* (*MCSoC*), pp.207–214 (2014).

[6] Takenae, M., Taniguchi, I. and Tomiyama, H.: A Case Study on Exploration of FPGA-based Multicore/Manycore Architectures, *Proc. International Symposium on Low-Power and High-Speed Chips* (*COOL Chips*) (2016).

[7] Rettkowski, J. and Göhringer, D.: RAR-NoC: A Recofigurable and Adaptive Routable Network-on-Chip for FPGA-based Multiprocessor Systems, *Proc. International Conference on ReConFigurable Computing and FPGAs* (*ReConFig*) (2014).

[8] Makni, M., Baklouti, M., Niar, S., Biglari-Abhari, M. and Abid, M.: Heterogeneous Multi-Core Architecture for a 4G Communication in High-Speed Railway, *Proc. International Design & Test Symposium*, pp.26–31 (2015).

[9] Nittala, R., Acquaviva, A. and Macii, E.: A Software Toolchain for Variability Awareness on Heterogeneous Multicore Platforms, *IEEE Trans. Emerging Topics in Computing*, Vol.5, No.1, pp.95–107 (2017).

Takafumi Miyazaki received his B.E. degree in Electronic and Computer Engineering from Ritsumeikan University in 2018. He is in Master's degree program in Ritsumeikan University. His research interests include, but not limited to, multicore architectures, design space exploration, parallel software for embedded systems. He is a member of IEEE.

Shunsuke Takai received his B.E. and M.E. degrees in electronic and computer engineering from Ritsumeikan University in 2015 and 2017, respectively. He currently works for Hitachi-Omron Terminal Solutions, Corp. His research interests include multicore architectures and parallel software for embedded/cyber-physical systems.

Ittetsu Taniguchi received his B.E., M.E., and Ph.D. degrees from Osaka University in 2004, 2006, and 2009, respectively. He is currently an associate professor at Graduate School of Information Science and Technology, Osaka University, Japan. From 2007 to 2008, he was a Ph.D. researcher at IMEC, Belgium. His research interests include system level design methodology, design methodologies for cyber-physical systems, etc. He is a member of IEEE, ACM, IEICE, IPSJ, and IEEJ.

Hiroyuki Tomiyama received his B.E., M.E. and D.E. degrees in computer science from Kyushu University in 1994, 1996 and 1999, respectively. He worked as a visiting researcher at UC Irvine, as a researcher at ISIT/Kyushu, and as an associate professor at Nagoya University. Since 2010, he has been a full professor with College of Science and Engineering, Ritsumeikan University. He has served on program and organizing committees for a number of premier conferences including DAC, ICCAD, DATE, ASP-DAC, CODES+ISSS, CASES, ISLPED, RTCSA, FPL and MPSoC. He has also served as editor-in-chief for IPSJ TSLDM, as an associate editor for ACM TODAES, IEEE ESL and Springer DAEM, and as chair for IEEE CS Kansai Chapter and IEEE CEDA Japan Chapter. His research interests include, but not limited to, design methodologies for embedded and cyber-physical systems.

(Recommended by Associate Editor: *Masayuki Hiromoto*)