IPSJ Transactions on System LSI Design Methodology Vol.12 42–45 (Feb. 2019)

[DOI: 10.2197/ipsjtsldm.12.42]

Short Paper

Design and Evaluation of Asymmetric and Symmetric 32-core Architectures on FPGA

Seiya Shirakuni[1,a)] Ittetsu Taniguchi[2,b)] Hiroyuki Tomiyama[1,c)]

Received: May 31, 2018, Revised: September 5, 2018,
Accepted: October 22, 2018

Abstract: Due to the advances in semiconductor technologies, recent FPGA devices are able to implement a number of CPU cores to realize high-performance embedded systems. This paper presents a case study on design, implementation and evaluation of manycore architectures on an FPGA. Two types of 32-core architectures with different topologies, i.e., asymmetric and symmetric architectures, are designed and implemented on an FPGA, together with an OpenCL-based software framework. The performance of the two architectures is evaluated based on actual measurement using various application programs.

Keywords: manycore architecture, FPGA, MicroBlaze, OpenCL

1. Introduction

The progress of semiconductor technologies enables to implement a large scale system on a single chip. This trend appears in the field of not only custom systems-on-chip (SoCs) but also field-programmable gate arrays (FPGAs). A recent FPGA device is capable of implementing a number of CPU cores, so-called manycores, to take advantage of parallel soft-ware execution, and FPGAs are used as platforms for final implementation as well as custom SoC prototyping.

Manycore architectures on FPGAs have been studied actively in the last decade. Matthews et al. developed a multicore architecture, named PolyBlaze, consisting of up to eight cores for Xilinx FPGA [1]. Tumeo et al. developed an FPGA-based multicore architecture for automotive applications [2]. Shibata et al. developed a system-level design tool for FPGA, named Advanced SystemBuilder [3]. A common feature of Refs. [1], [2] and [3] is that OS is executed on every core and the cores are evenly important. Mori and Kise developed a sixteen-core architecture on a Xilinx FPGA [4]. Takenae et al. developed manycore architectures consisting of up to 32 cores on a Xilinx FPGA [5]. One of the differences between Refs. [4] and [5] is the location of the host core. The SMYLEref architecture developed by Nguyen et al. [6] takes the similar approach to the one in Ref. [5]. One of the conceptual differences between Refs. [5] and [6] is the symmetry of the host and slave cores. Makni et al. developed two types of multicore architectures [7]. These architectures are proposed of different kinds of soft cores. Miyazaki et al. proposed a heterogeneous multicore architecture consisting of dual Cortex-A9 cores and dual MicroBlaze cores [8].

This paper studies the accelerator-based manycore architectures on FPGA. As mentioned above, there exist several research efforts on this topic in the past. However, few work presented quantitative comparison with other architectural alternatives based on actual FPGA implementation and measurement.

In this work, we have designed two types of 32-core architectures with different topology: One is asymmetric, and the other is symmetric. We have implemented the two architectures on an FPGA, ported Linux on the host core, executed several application programs on the slave cores, and actually measured their execution times.

2. Design of 32-core Architectures

2.1 Asymmetric 32-core Architecture

Figure 1 depicts our asymmetric 32-core architecture, called ASYM32. ASYM32 has two types of cores, i.e., a host core and slave cores, both of which are based on the MicroBlaze architecture. The host core is the main CPU core in ASYM32, which executes the Linux operating system and most of application programs. The slave cores are used to accelerate specific applica-

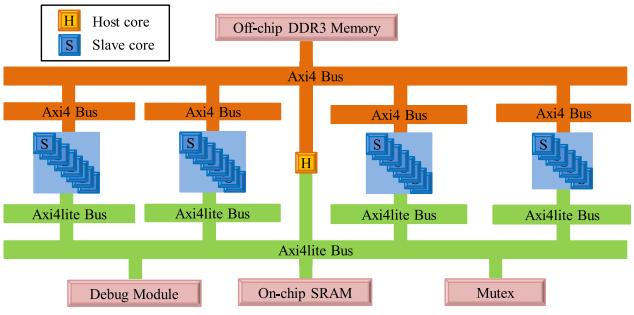


Fig. 1 Asymmetric 32-core architecture (ASYM32).

[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) seiya.shirakuni@tomiyama-lab.org
b) i-tanigu@ist.osaka-u.ac.jp
c) ht@fc.ritsumei.ac.jp

tions. No operating system runs on the slave cores. Instead, simple runtime software is executed on each slave core. Each of the host core and the slave cores has local memory, instruction cache and data cache, whose sizes are configurable. By default, the host core has 16 KB local memory, 16 KB instruction cache and 16 KB data cache. Each slave core has 4 KB local memory, 1 KB instruction cache and 1 KB data cache. The local memory is dedicated to the core, and is not shared with the other cores. Only the host core has a memory management unit (MMU) in order to run Linux.

Off-chip DDR3 DRAM, shown in the upper part of Fig. 1, is the main memory of the entire system, and programs to be executed by all cores and the host core's Linux kernel are stored in the DRAM. The Linux image file is transferred from a host PC to the DRAM on the FPGA board through a USB cable. The host core and the slave cores access the DRAM through their instruction and data caches. Communication between the cores can be realized using on-chip SRAM, which is depicted at the bottom of Fig. 1.

In ASYM32, the host core is connected to the top-level AXI4 bus, and the slave cores are hierarchically grouped into four clusters, each of which consists of eight or seven slave cores.

2.2 Symmetric 32-core Architecture

Figure 2 presents the symmetric 32-core architecture, named SYM32. SYM32 also consists of a host core and 31 slave cores. Similar to ASYM32, the Linux operating system runs only on the host core, and the slave cores are used to accelerate specific applications. In this way, from a viewpoint of software, the SYM32 architecture is similar to ASYM32. The essential difference between SYM32 and ASYM32 is the location of the host core. In SYM32, the host core is located within the same cluster as slave cores. Thirty-two cores including the host core are partitioned into four clusters, each of which consists of eight cores.

3. Parallel Software Framework

In this work, we have ported the OpenCL-like framework presented in Ref. [9] onto our architectures, and we extended the slave activation mechanism. Some applications which have poorly-scalable parallelism may not need as many as 31 slave cores. Using a smaller number of slave cores may achieve the higher performance. Let us assume a program which uses only four slave cores. There exist several options to select four slave cores out of 31. One simple way is to select four slave cores from the same cluster. We call this the *packed* method. In case of SYM32, when the required number of slave cores is less than or equal to 24, slave cores are selected from clusters without the host core. Another simple way to select four slave cores is to select one slave core from each cluster. We call this the *balance* method. We implemented both of the two methods in our software framework so that programmers can choose either one.

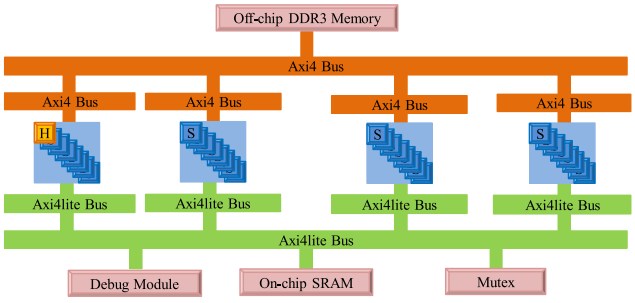


Fig. 2 Symmetric 32-core architecture (SYM32).

4. Experiments

We have implemented the ASYM32 and SYM32 architectures on Xilinx's Kintex-7 FPGA KC705 board, and evaluated their performance. We used Vivado Design Suite 2016.2 as a design toolkit.

4.1 Linux Boot Time

We measured the boot time of Linux on ASYM32 and SYM32 in order to evaluate the performance of the host core. Linux image file is pre-stored in the DRAM on the FPGA board, and the host core boots the Linux directly from the DRAM. The Linux image is not stored in HDD, SSD or SD card, and no boot loader is necessary in our study. The host core accesses the DRAM through its instruction and data caches. Since the cache size is one of important parameters which directly affect the Linux boot time, we changed the size of instruction and data caches of the host core. The results are summarized in **Table 1**. Table 1 clearly shows that the large caches improve the Linux boot time on both architectures. Also, the Linux boot time on ASYM32 is shorter than that on SYM32. In SYM32, when cache misses occur, off-chip DRAM is accessed through two AXI4 buses, resulting in longer access latency. On the other hand, in ASYM32, the host core and the DRAM controller are connected to the same AXI4 bus, resulting in shorter access latency.

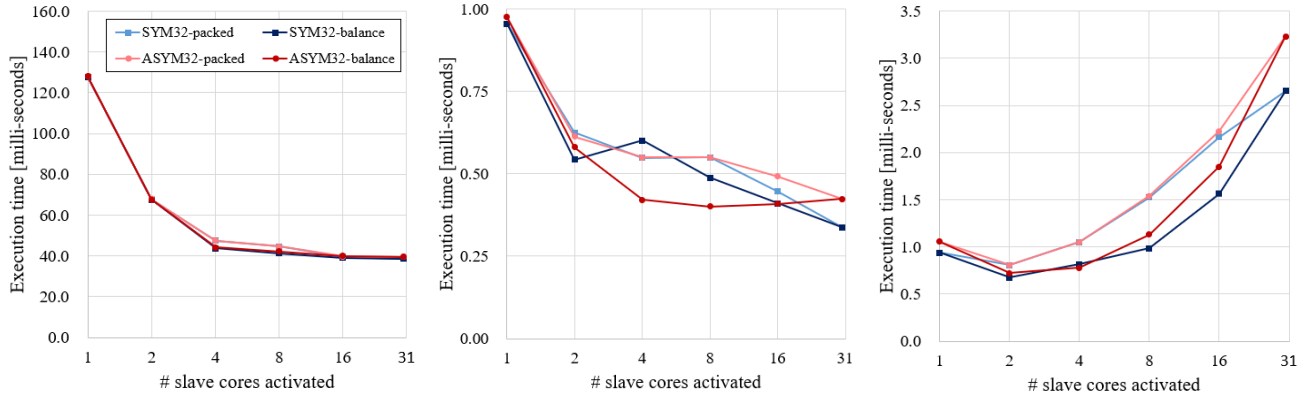
4.2 Execution Times of Applications

We executed three benchmark programs on the two architectures. The programs are Gaussian filter, grayscale conversion and runlength encoding from BEMAP [10]. When we executed the programs, we change the number of slave cores activated from one to 31 cores. In addition, we tested two execution methods, i.e., the packed method and the balance method as described in Section 3. **Figure 3** shows the execution time of three benchmark programs.

The graph (a) shows the execution time of the Gaussian filter. The graph (a) shows that an increase in the activated slave cores generally leads to performance improvement, but the amount of improvement is small when more than four slave cores are used. This is mainly because of the memory-intensive nature of the program. Data is stored in the on-chip SRAM and all of the activated slave cores access the SRAM. Also, the off-chip DRAM is also congested. The capacity of the instruction cache in the slave

Table 1 Linux boot time [seconds].

Host Cache Size	ASYM32	SYM32
1KB each	35.98	37.30
4KB each	32.34	33.23
16KB each	30.31	31.16



(a) Gaussian filter (b) grayscale conversion (c) runlength encoding

Fig. 3 Performance comparison.

cores is only 1 KB, and is not sufficient to store the whole program. Therefore, instruction cache misses occur often in the slave cores, resulting in the poor scalability of performance. Between ASYM32 and SYM32, and also between the packed method and the balance one, little difference of performance is observed.

The graph (b) shows the execution time of the grayscale conversion program. In case all of the 31 slave cores are used, SYM32 achieves better performance than ASYM32. The reason is as follows. After the host core activates the slave cores for parallel execution, the host core waits for the completion of the slave cores. This synchronization between the host core and the slave cores is implemented by polling specific addresses in the on-chip SRAM. This means that, while the slave cores are running, the host core also accesses the on-chip SRAM frequently. These SRAM accesses by the host core often disturb data accesses from the slave cores. This problem is more severe in ASYM32 than in SYM32, since in ASYM32 the host core is directly connected in the SRAM. It should be noted that, in each bus, accesses are arbitrated in a round-robin manner. Therefore, in SYM32 with 31 active slave cores, SRAM accesses from the host core are often blocked at the lower-level bus, and therefore the slave cores can access the SRAM faster than in ASYM32. It is also observed in the graph (b) that, when the activated slave cores are less than 31, the balance method outperforms the packed method in most cases. This is because memory access contention is less severe in the balance method than in the packed method.

The graph (c) shows the execution time of the runlength encoding program. SYM32 using two slave cores achieves the best performance. Compared with the Gaussian filter and grayscale conversion programs, the runlength encoding program requires synchronization among cores more often, which limits the parallelization opportunities. Similar to the grayscale conversion program, the balance method outperforms the packed method.

5. Conclusions

This paper presented a case study on design, implementation and evaluation of manycore architectures on an FPGA. Two types of 32-core architectures, i.e., asymmetric and symmetric architectures, were designed and implemented on an FPGA. The experimental results show both advantages and disadvantages of the two architectures, and balance method is better than packed method. In addition, their effectiveness highly depends on the characteristics of the programs. Our future works include automatic design space exploration of the best architecture for given application programs.

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

References

[1] Matthews, E., Shannon, L. and Fedorova, A.: Polyblaze: From One to Many Bringing the MicroBlaze into the Multicore era with Linux SMP Support, *International Conference on Field Programmable Logic and Applications* (2012).

[2] Tumeo, A., Branca, M., Camerini, L., Ceriani, M., Monchiero, M., Palermo, G., Ferrandi, F. and Sciuto, D.: A Dual-Priority Real-Time Multiprocessor System on FPGA for Automotive Applications, *Design, Automation and Test in Europe* (2008).

[3] Shibata, S., Honda, S., Tomiyama, H. and Takada, H.: Advanced SystemBuilder: A Tool Set for Multiprocessor Design Space Exploration, *International SoC Design Conference* (2010).

[4] Mori, H. and Kise, K.: Design and Performance Evaluation of Manycore Processor for Large FPGA, *International Symposium on Embedded Multicore/Many-Core Systems-on-Chip* (2014).

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

[6] Nguyen, S.-T., Kondo, M., Hirao, T. and Inoue, K.: A Prototype System for Many-Core Architecture SMYLEref with FPGA Evaluation Boards, *IEICE Trans. Information and Systems*, Vol.E96-D, No.8, pp.1645–1653 (2013).

[7] Makni, M., Niar, S., Baklouti, M., Wassim, M. and Abid, M.: A Comparison and Performance Evaluation of FPGA Soft-cores for Embedded Muti-core Systems, *International Design & Test Symposium* (2016).

[8] Miyazaki, T., Taniguchi, I. and Tomiyama, H.: A Heterogeneous Multicore Architecture and a Parallel Software Environment for Zynq SoC, *International Symposium on Advanced Technologies and Applications in the Internet of Things* (2018).

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

[10] Ardila, Y., Kawai, N., Nakamura, T. and Tamura, Y.: Support Tools for Porting Legacy Applications to Multicore, *Asia and South Pacific Design Automation Conference* (2013).

Seiya Shirakuni received his B.E. degree in Electronic and Computer Engineering from Ritsumeikan University in 2017. He is in Master's degree program in Ritsumeikan University. His research interests include, but not limited to, many-core architectures on FPGA and design space exploration.

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: *Hideho Arakida*)