HW/SW CO-DESIGN FOR THE XILINX ZYNQ PLATFORM

Jan Viktorin
Master Degree Programme (2), FIT BUT
E-mail: xvikto03@stud.fit.vutbr.cz

Supervised by: Pavol Korček
E-mail: ikorcek@fit.vutbr.cz

Abstract: This work describes a novel approach of HW/SW co-design on the Xilinx Zynq and similar platforms. It summarizes interconnections between the Processing System (ARM Cortex-A9 MPCore) and the Programmable Logic (FPGA) to find an abstract and universal way for developing applications that are partially offloaded into the programmable hardware and that run in the Linux Operating System. No such universal framework is currently available.

Keywords: Zynq, FPGA, SoC, HW/SW Codesign, Linux, AXI

1 INTRODUCTION

Today the market of computing systems is led by embedded systems as they are almost everywhere. They are often equipped with a CPU, peripherals and some application specific units (e. g. graphic coprocessor). All of them are integrated together on one physical chip called System-on-Chip (SoC). This work explores a new approach of designing embedded systems using a new generation of SoCs that consist of a CPU (with peripherals) and Field-Programmable Gate Array (FPGA) logic. The FPGA logic enables to interface with different peripherals and to integrate application specific units without affecting the rest of the system. For those new SoCs, and the Xilinx Zynq is one of them, a framework that simplifies and speeds up the development of an embedded system is introduced.

2 XILINX ZYNQ

The Xilinx Zynq SoC can be divided into two parts (see the Figure 1): the *Processing System* (PS) and the *Programmable Logic* (PL). [4, p. 23]

The PS consists of

- two ARM Cortex-A9 cores at 667–1000 MHz[1] with the NEON coprocessor,
- the Snoop Control Unit (SCU) that ensures coherency among L2 cache, L1 caches of both cores and the Accelerator Coherency Port (connected to the PL),
- the On-Chip Memory (OCM) containing 256 KB of RAM,
- a set of peripherals (SPI, I^2C, Gigabit Ethernet, UART, CAN, USB Host, etc.),
- a DDR memory controller and an interrupt controller. [4, p. 28]

The PL is an FPGA (equivalent to the Xilinx Artix-7 or Kintex-7 family) that is tightly interconnected with the PS. It enables the implementation of hardware accelerators to speed up the processor's computation. Depending on the version, the PL offers 17,000–218,000 6-inputs look-up tables (LUTs), 60–545 Block RAM memories each with capacity of 36 Kb, 80–900 DSP slices, PCIe interface (only the Kintex-7 compatible devices), multi-gigabit transceivers and others. [3, 2]

[1] Depends on the selected chip.

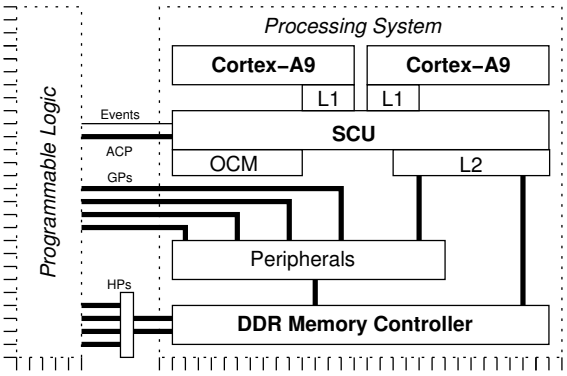

Figure 1: The architecture of Zynq (simplified)

3 INTERFACES BETWEEN PS AND PL

The interfaces between both parts of the Zynq platform can be divided into three categories:

- **Low-performance interface** is represented by four General Purpose (GP) AXI ports connected to the internal bus of the PS. Two of them are in the *slave* and the other two in the *master* mode. The PS can access any hardware component in the PL by its address using the slave ports and the PL part can do bus master operations using the master ports. [4, p. 529, 543–548]
- **High-performance interface** is represented by four High Performance (HP) AXI ports connected to the main memory of the PS and by the Accelerator Coherency Port (ACP) connected to the Snoop Control Unit (i. e. L2 cache). All the ports are in *master* mode and provides 64 bit data bus. The ACP simplifies PS-PL communication because of the hardware coherency control. As opposed to the ACP, the HPs need additional software overhead because of explicit flushing of caches before a transaction is performed. [4, p. 529, 543–548]
- **Signalization interface** provides ways in which to notify both the PS and PL when an event occurs. The most traditional way is interrupts. The PL can use 20 dedicated lines to interrupt the processor. Few of them are of a high priority. There is also the *Event Interface*. Any Cortex-A9 core can perform the `SEV` instruction to send a notification into the SCU. Such an event is sent to other cores and to the PL. The same type of event can be generated from PL. Similarly the `WFE` instruction is used to put the core into a low-power mode while waiting for an event to occur. The PL can detect when a core is put into the low-power mode.[4, p. 53, 102]

The HPs, GPs and ACP interfaces are implemented using the ABMA AXI, the family of protocols suitable for creating a bus system in the SoC architectures. The details can be found in [1].

4 FRAMEWORK FOR HW/SW CO-DESIGN ON THE ZYNQ

To simplify development of applications for the Zynq platform (and also for other *Reconfigurable SoC* devices) a framework of common hardware components and software drivers is designed. Its main idea is presented in the Figure 2. The platform dependent parts (*PS Driver* and *PL Endpoint*) ensure DMA transfers and correct signalization between PS and PL. These components are hidden from the application developer who needs to know just the *Accelerators Interface* and *Userspace interface*. The latter one provides a duplex (or in special cases just a simplex) streaming interface and a configuration bus. The transfers and the interrupts/events generation are hidden from the accelerators. The *Userspace interface* enables an application to write data into a selected accelerator and read back the results. The configuration address space of each particular accelerator can be exposed to the

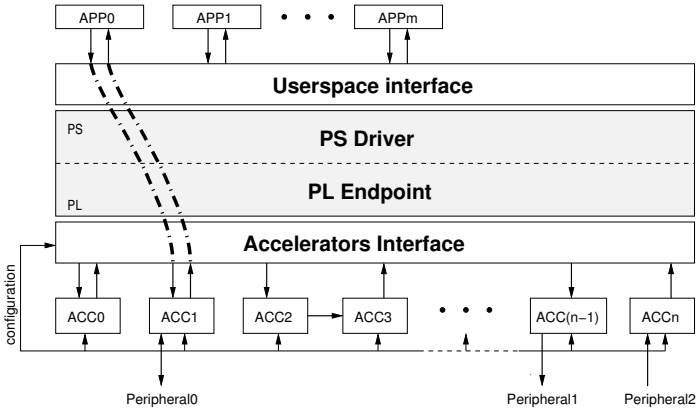

Figure 2: *The Reconfigurable SoC Bridge* design

application using a memory mapping capabilities of the used OS (the Linux OS provides the system call *mmap* that is suitable for this purpose).

As a result, the developer uses the concept of streams between the software and hardware and this enables a faster development of a target application. At the same time, the concept does not depend on the Xilinx Zynq but can be implemented for any similar platform.

5 CONCLUSION AND FUTURE WORK

The proposed solution provides a way to develop applications for *Reconfigurable SoC* devices while taking advantage of both the software (easier and faster to develop but less efficient) and the hardware (more complex to develop but much more efficient). It also simplifies the development of hardware accelerators by introducing a simple interface and hiding the complexity of DMA communication. The concept is independent on the Xilinx Zynq platform and it is expected to be ported to other similar systems. The well-specified interfaces allow to apply the *Partial Dynamic Reconfiguration* of the soft-core accelerators in the Programmable Logic.

ACKNOWLEDGEMENT

This work was supported by the grant BUT FIT-S-12-1 and Sec6net project no. VG20102015022.

REFERENCES

[1] ARM. *AMBA AXI and ACE Protocol Specification*, 2011. Online: `https://silver.arm.com/download/download.tm?pv=1198016`.

[2] Xilinx. *7 Series FPGAs Overview*, November 2012. Online: `http://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf`.

[3] Xilinx. *Zynq-7000 All Programmable SoC Overview*, August 2012. Online: `http://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf`.

[4] Xilinx. *Zynq-7000 All Programmable SoC Technical Reference Manual*, November 2012. Online: `http://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf`.