HW-extended Containers on FPGA-based RISC-V SoC

Konstantinos Amplianitis[1], Katerina Tsimpirdoni[1], Andreas Brokalakis[1], George Christou[1], Konstantinos Georgopoulos,[1] and Sotiris Ioannidis[1]

[1]School of Electrical and Computer Engineering, Technical University of Crete

Abstract

This paper describes a technology that brings together three key elements of reconfigurable hardware (FPGAs) prototyping, namely, Docker containers, RISC-V architectures, and runtime (dynamic) partial reconfiguration. The work envisaged serves the purpose of further expanding FPGA capabilities by allowing these processing platforms to support state-of-the-art development of prototypes with RISC-V soft-cores at their centre. The RISC-V processor features an Operating System (OS) that fully supports the execution of HW-extended Docker containers, which in turn contain all the necessary libraries, software, firmware and bitstreams for the implementation and utilisation of accelerator cores/modules within the reconfigurable fabric of the FPGA. Hence, different Docker containers, representing separate services and clients, will be able to deploy on-demand part of their functionality directly into the FPGA fabric benefiting from the parallel execution capabilities that this type of technology has to offer while the overall system will be based on a soft instance of a RISC-V processor core.

Introduction

Modern FPGAs, along with their variations such as the Multi-Processor System-on-Chip (MPSoC) [1] are at the forefront of technological advancements. They are currently used in various solutions, e.g., IoT, Supply Chains, Industry, Telecoms, High Performance Computing (HPC), and Security [2] both in the form of prototype development as well as part of system deployment. Their main advantages are found in *i)* that they can offer highly parallel architectures at various levels of granularity, thereby accelerating computationally heavy algorithms and tasks, *ii)* extreme versatility and, *iii)* low power consumption. Moreover, the ability to re-visit and update an existing implementation in-field, makes them an attractive solution since active designs can be updated into newer and more robust versions of their original implementations without the need to replace the actual host hardware platform.

Consequently, bridging container-like technology, which offers OS virtualisation, on the hardware level by utilising FPGAs, results in a highly attractive development environments for users that want to benefit from FPGA-based acceleration on compute-intensive tasks while making the shared demand for access to the FPGA fabric a more manageable and secure task.

Concept & Implementation

The concept of the HW-extended container is based upon the fundamental FPGA fabric split into *static* and *dynamic* designs, i.e. an architecture that is comprised of a collection of permanent modules (static) that form the core system functionality, and a collection of temporary modules that are replaceable on-demand and correspond to different services (dynamic).

Commonly, a realisation of this split is performed using devices such as MPSoC chips where the static portion of the design is *"hard"*, i.e. it has been fabricated this way and cannot, therefore, be modified. Hence, only the dynamic section of the chip is reconfigurable and can be re-designed at will. Typically, under these schemes, the static section includes ARM processor cores along with a number of additional functional blocks, e.g. a management unit, on-chip memory, GPU and I/O. Due to the vendor support that usually accompanies these devices, it is an easier task for a designer to select a suitable OS that includes all necessary libraries and drivers and deploy it to the static processor(s). This allows for a greater focus on the harder task at hand, i.e. the development of (optimised) custom hardware modules, which represent the functionality required by a user, as well as the demarcation of suitable Programmable Region (PR) [3] placeholders that are meant to host said custom modules introduced by the various system users. Nevertheless, the architecture that exists on the static side is fixed and remains beyond the control of the developer, in addition, the majority of the *"hard"* functional blocks are most of the time redundant for a given application.

Hence, this work replaces the static portion of the system with a RISC-V processing core [4] along with necessary blocks for the system's operation. The benefits of this approach are twofold. First, it offers an FPGA-based prototype development platform that has a RISC-V processor at its core, free from proprietary architecture dependencies and practical considering the popularity of RISC-V in recent years [5]. The second has to do with the ability to modify the static part in accordance with further developments by the RISC-V foundation [6] and affiliated community so that the system processor is always up-to-date. That is because the static portion of the design is also implemented on reconfigurable hardware and it can be updated if so desired.The overall architecture of the concept is shown in Figure-1 and presents the essential prototype building blocks. First, the FPGA fabric is divided into a static and a dynamic portion. The static side is also

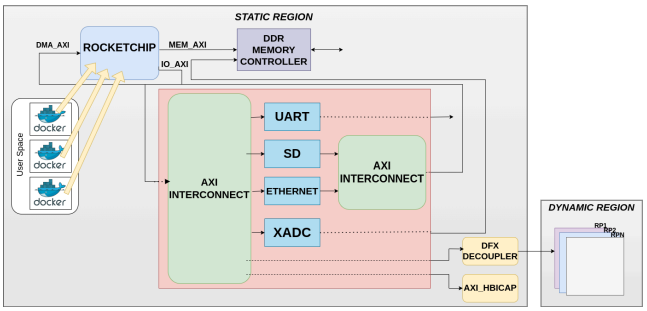

Figure 1: High-level overview of the static and dynamic regions.

hosted on reconfigurable fabric and can, therefore, be modified according to the developers' needs. Overall, it contains a linux capable RISC-V core, which the users can utilise to use part of the dynamic reconfigurable fabric to implement and accelerate parts of their process by exploiting the parallelism capabilities. The dynamic section is designed to offer a number of Programmable Regions (PRs) able to host a dedicated AXI connected hardware module that is user/Docker image-specific. The corresponding PR region will become available when the task is completed.

The prototype's static side, consists of a multi-core RISC-V processor along with utilitarian functional blocks, such as, *i)* I/O responsible for interfacing the RISC-V with the development board's various peripherals, *ii)* DDR responsible for interfacing with the on-board memory, and, *iii)* DMA responsible for supporting direct access to the DDR memory. All these blocks, along with the RISC-V and the different PR regions, are connected to a crossbar that supports all-to-all connection so, for instance, a custom PR module can talk to the RISC-V and/or read/write data directly from/to memory. Finally, the static side will also contain a configuration block, e.g. ICAP, responsible for transferring the partial bitstream to the dynamic side of the FPGA.

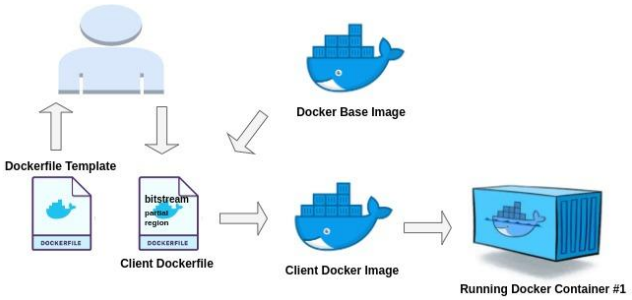

Figure 2: HW-extended Docker flow.

System users can request access to the service through a dedicated utility that reports the availability of reconfigurable regions. Subsequently, the host's OS will reply back with i) a Dockerfile template that the client has to adapt by performing minor modifications in order to make it user-specific and ii) the currently available programmable regions on the FPGA. The client's Dockerfile will be based on a Docker image created by the system, containing all the needed tools in order to perform the reconfiguration of a specific region. The information that the client will have to provide are the bitstream file and the target PR region.

This will produce a client-specific Docker Image that utilises the existing tools in the system's Docker Base Image in order to deploy a new client-specific Docker container. The client's bitstream will be programmed through the FPGA vendor's method in the specific region that was available and chosen by the client. After this procedure, the client will have access to the container by the default Docker commands.

Conclusions & Future Work

This short paper introduces an attractive concept that combines a number of key technologies in modern computer technology. These are Docker containers, RISC-V processors and FPGA partial reconfiguration. The idea attempts to enable a better utilisation of FPGA devices in the context of services and research by offering their resources to clients that wish to combine a RISC-V based management and execution system of processes and tasks with reconfigurable hardware that can offer computation speedups in addition to low-power execution. Hence, users are offered the opportunity to deploy HW-extended Docker containers on a soft RISC-V processor that aside from software execution can also introduce hardware modules in pre-assigned programmable regions within the reconfigurable fabric of the FPGA. During the next implementation steps we aim to evaluate the overall performance of the system by deploying cryptographic and AI accelerators.

References

[1] "Description of xilinx us+ multi-processor system-on-chip,"https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html

[2] O. Mencer, D. Allison, E. Blatt, M. Cummings, M. J. Flynn, J. Harris, C. Hewitt, Q. Jacobson, M. Lavasani, M. Moazami, H. Murray, M. Nikravesh, A. Nowatzyk, M. Shand, and S. Shirazi, "The history, status, and future of fpgas: Hitting a nerve with field-programmable gate arrays," Queue, vol. 18, no. 3, p. 71–82, jul 2020. [Online]. Available: https://doi.org/10.1145/3411757.3411759.

[3] "Dynamic function exchange," https://www.xilinx.com/support/documentation-navigation/design-hubs/dh0017-vivado-partial-reconfiguration-hub.html, accessed: 07-02-2025.

[4] "Out of order risc-v processor based on rocketchip," https://github.com/eugene-tarassov/vivado-risc-v, accessed: 07-02-2025.

[5] "Risc-v pushes into the mainstream," https://semiengineering.com/risc-v-pushes-into-the-mainstream/, accessed: 07-02-2025.

[6] "Risc-v foundation," https://riscv.org/, accessed: 07-02-2025.