The FOS (FPGA Operating System) Demo

The FOS (FPGA Operating System) Demo

Anuj Vaishnav, Khoa Dang Pham, Kristiyan Manev, and Dirk Koch
School of Computer Science, The University of Manchester, UK
Email: {anuj.vaishnav, khoa.pham, kristiyan.manev, dirk.koch}@manchester.ac.uk

***Abstract*—With the introduction of Zynq FPGAs that provide an ARM SoC with an attached FPGA fabric, it is possible to build complex software-centric systems that are software and hardware programmable. To harness the full potential of this approach, we developed FOS an FPGA Operating System which is built on open-source FPGA community and Xilinx vendor components. A distinct feature shown in this demo is a heterogeneous resource elastic scheduler that can dynamically and automatically adjust the allocation of tasks to hardware and software resources with respect to the present load scenario.**

We will also show the FOS ecosystem that allows easily implementing relocatable partially reconfigurable modules directly from RTL or HLS.

I. Introduction

FPGA technology is currently experiencing an exciting time as FPGAs are becoming core citizens in datacenters and SoCs are now available for building sophisticated embedded systems that couple powerful 64-bit ARM SoCs with an FPGA fabric. As compared to earlier CPU-FPGA-hybrid solutions (e.g., Altera Excalibur or Xilinx Virtex-II Pro), today's devices are now software-centric, i.e. operating the FPGA under full software control.

One issue that we identified in this approach is that hardware accelerators cannot be deployed with the flexibility that is inherently available in the FPGA and, for example, PYNQ accelerators cannot be arbitrarily instantiated at runtime. This essentially treats the FPGA as a reprogrammable ASIC rather than a flexible dynamically reprogrammable hardware platform.

This is exactly what FOS the FPGA Operating System is addressing and the operation and ecosystem of FOS along with its user-friendliness will be demonstrated.

II. Demo Setup

Figure 1 shows an ULTRA96 board featuring a Xilinx Zynq UltraScale+ ZU3EG FPGAs. The shell is based on ZUCL [1] and tiles the FPGA resources into three slots for hosting relocatable modules of different size. Modules can be implemented in RTL or HLS, including support for OpenCL. Each slot provides a 32-bit AXI slave and a 128-bit AXI master. AXI stream and other AXI interfaces of different size are supported through wrappers. The system allows accelerators to access the 2GB system memory in a protected way using the ARM System Memory Management Unit (SMMU). FOS had been ported to other Zynq UltraScale+ boards (e.g., UltraZed, ZCU102), see `https://github.com/khoapham/fos.git`.

FOS is based on PetaLinux and Ubuntu and provides additional runtime services for compilation, (partial) configuration and runtime management.

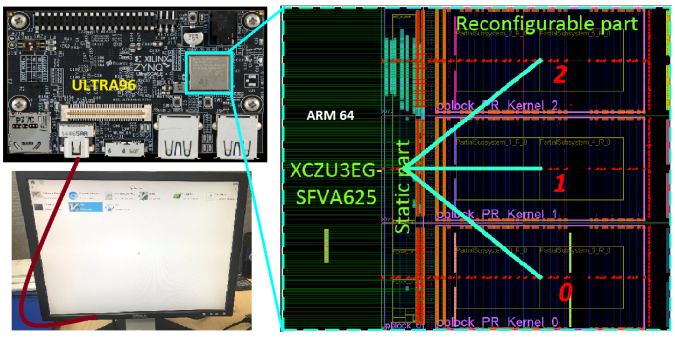

Fig. 1: FOS Demo providing a shell with 3 slots.

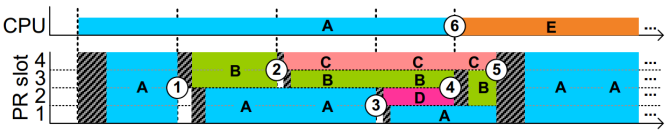

Fig. 2: Heterogeneous resource elastic scheduling (HRES).

III. FOS Demo

The FOS runtime provides a Heterogeneous Resource Elastic Scheduler (HRES) [2]. The demo includes scenarios as illustrated in Figure 2 where, for example, Task `A` is long-running and compute bound (e.g., a Mandelbrot deep dive). In FOS, applications can fully utilize all hardware and software resources and if other tasks arrive (see ①, ②, ③, ⑥) or terminate (see ④, ⑤), the application can shrink or expand at any point in time to maximize utilization and therefore throughput. For OpenCL acceleration tasks, the resource reallocation is performed entirely transparent and similar to what is know from software-only systems and the process is only noticeable by the application throughput (e.g., the speed of the Mandelbrot computation).

FOS provides further features that someone expects from a complete Linux distribution, including a compilation path from Verilog to a partial bitfile [3] such that modules can be compiled, configured and executed directly on the Ultra96.

Acknowledgment

This work is kindly supported by the European Commission through the H2020 projects ECOSCALE & EuroEXA (grants 671632 & 754337). We also thank Xilinx for tools and boards.

References

[1] K. D. Pham, A. Vaishnav, M. Vesper, and D. Koch, "ZUCL: A ZYNQ UltraScale+ Framework for OpenCL HLS Applications," in *FSP*, 2018.
[2] A. Vaishnav, K. D. Pham, D. Koch, and J. Garside, "Resource Elastic Virtualization for FPGAs using OpenCL," in *FPL*, 2018.
[3] K. D. Pham, M. Vesper, D. Koch, and E. Huang, "EFCAD – an Embedded FPGA CAD Tool Flow," in *FCCM*, 2019.