A Self-Compilation Flow Demo on FOS – the FPGA Operating System

Demo title

A Self-Compilation Flow Demo on FOS – the FPGA Operating System

Author names and affiliations

Khoa Dang Pham, Anuj Vaishnav, Joseph Powell, and Dirk Koch
Department of Computer Science, The University of Manchester, UK
khoa.pham@manchester.ac.uk
anuj.vaishnav@manchester.ac.uk
joseph.powell@manchester.ac.uk
dirk.koch@manchester.ac.uk

Description of the main objectives and relevance of the demo to the FPL community

The main objective is demonstrating the self-compilation flow (EFCAD) integrated in the FPGA Operating System (FOS) to provide a user-friendly development environment and a fast prototyping solution for developing FPGA applications.
We packaged EFCAD in the FOS repository on GitHub to share with the FPGA community.

Short biographies of the presenters

- Khoa Dang Pham is a PhD student in CS, the University of Manchester, UK. He holds a BSc in Mechanical Engineering from Ho Chi Minh City University of Technology, and a MEng (research-based program) in Computer Engineering from Nanyang Technological University, Singapore. Khoa has his working experiences in both academic and industrial sectors including University of Houston and Applied Micro (now Ampere Computing). His research interests include reconfigurable computing, partial reconfiguration, computer architecture, and embedded systems.
- Anuj Vaishnav is currently a PhD candidate at the University of Manchester. He received his BEng (Hons) in Computer Systems Engineering from the University of Manchester in 2017. His current research focus include resource management, virtualization and programmability of FPGAs.
- Joseph Powell got his Bachelor and Master degrees from the University of Manchester. He is working as a research software engineer at the Advanced Processor Technologies Group at the University of Manchester.
- Dirk Koch is a senior lecturer in the Advanced Processor Technologies Group at the University of Manchester. His main research interest is on run-time reconfigurable systems based on FPGAs, embedded systems, computer architecture, VLSI and hardware security. Dirk developed techniques and tools for self-adaptive distributed embedded control systems based on FPGAs. Current research projects include database acceleration using FPGAs-based stream processing, HPC and Exascale computing, as well as reconfigurable instruction set extensions for CPUs and using FPGAs in datacenters.

Any logistical requirements the demonstration may have (e.g. Internet access...)
Screen, power, small desk, poster board…
This Demo can be presented as a video, recording, if necessary.

A Self-Compilation Flow Demo on FOS – the FPGA Operating System

Khoa Dang Pham, Anuj Vaishnav, Joseph Powell, and Dirk Koch
Department of Computer Science, *The University of Manchester*, UK
{khoa.pham, anuj.vaishnav, joseph.powell, dirk.koch}@manchester.ac.uk

***Abstract*—With the introduction of Zynq UltraScale+ MPSoCs equipped with powerful 64-bit ARM CPUs along with a 16nm UltraScale+ FPGA fabric in the same die, we can now build full hardware-software programmable systems and configure the FPGA with accelerators, as needed. Traditionally, accelerators had been developed offline using powerful servers running heavy lifting CAD toolchains.**

In this demo, we show a self-compilation system supporting a user-friendly Jupyter Notebook GUI and multi-tenancy use of the FPGA for educational purposes. From a user perspective, this system compiles accelerators at run-time directly on the ARM CPU without any involvement of the vendor tools. The final bitstreams then execute on the FPGA fabric using PR.

I. Introduction

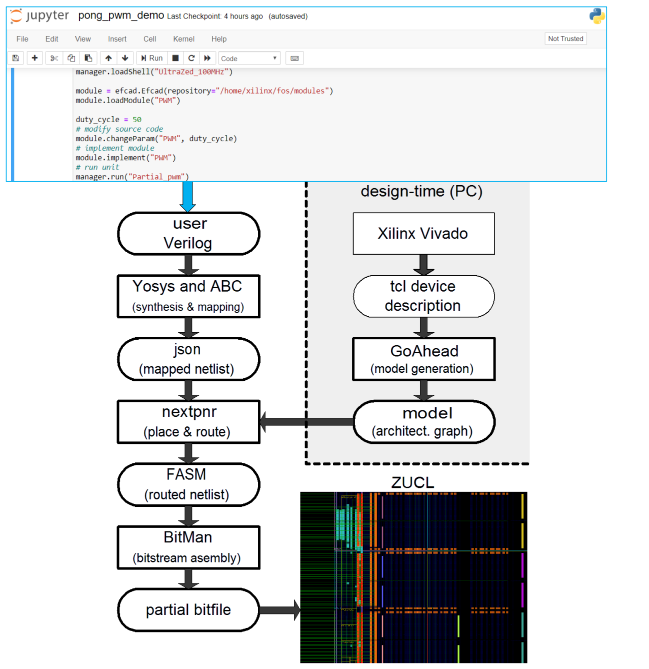
Starting to use FPGA systems is a non-trivial task as users may need to learn one of the hardware description languages (HDLs) as well as to download, install, maintain, and operate complex FPGA toolchains. With the introductions of the Pynq initiative [1] and an academic self-compilation EFCAD flow [2], there is a solution to build lightweight self-contained and out-of-the-box FPGA systems for educational purposes. Everything that is needed to run an FPGA hardware programming labs consists of a small FPGA board fitted with a mico-SD card (installed with FOS and EFCAD).

This demo shows how to design or modify FPGA designs from a Jupyter Notebook with a GUI interface to FOS, the FPGA Operating System [3] to allow operating designs through Python. Designs can be synthesized and implemented using the EFCAD flow to generate partial bitstreams at run-time and deploy them on the FPGA fabric, as shown in Fig. 1.

II. Demo Setup

The demo is running on a lightweight UltraZed/Ultra96 platform. The FOS boot image integrates the EFCAD flow and the Jupyter Notebook GUI. Further, we have developed a new EFCAD Python API in order to let the Python code written in the Jupyter Notebook GUI to modify the underlying Verilog source code. Moreover, the FOS Ponq API will be used to load bitstreams onto the FPGA for accelerator deployment. The use of FOS allows multiple users to compile and run their applications concurrently and in isolation with its run-time system and shell. This demo is open-source and integrated into the FOS repository at `https://github.com/khoapham/fos.git`.

We would like to thank the UK NCSC project rFAS (grant 4212204/RFA 15971), the H2020 project EuroEXA (grant 754337) for supporting this work, and the Xilinx University Program for tools and boards.



```
manager.loadShell("UltraZed_100MHz")

module = efcad.Efcad(repository="/home/xilinx/fos/modules")
module.loadModule("PWM")

duty_cycle = 50
# modify source code
module.changeParam("PWM", duty_cycle)
# implement module
module.implement("PWM")
# run unit
manager.run("Partial_pwm")
```

Fig. 1. The EFCAD demo on an Ultra96 with the Jupyter Notebook GUI. Initially, Vivado and GoAhead are used to extract the architectural graph, which will be used by the EFCAD flow running entirely on the ARM CPU.

III. EFCAD Demo on FOS

Several templates of FPGA applications, including Pulse-Width Modulation (PWM), UART (RS232), and stopwatch will be provided. Users will load one of these templates to their Jupyter Notebook on the FOS machine (a Zynq UltraScale+ FPGA equipped with the FOS software and hardware stack) and be able to change some parameters of the applications. The modified parameters will be reflected to the original Verilog source codes of the application. The updated Verilog code will go through synthesis, implementation, and bitstream generation steps using the underlying EFCAD flow, as shown in Fig. 1. The final partial bitstream of the modified application will then be configured to the FOS shell and executed.

With this, students can design own relocatable Verilog modules using a pre-defined interface and partial regions.

References

[1] Xilinx Inc., “PYNQ: Python productivity,” http://www.pynq.io.
[2] K. Pham et al., “EFCAD – An Embedded FPGA CAD Tool Flow for Enabling On-Chip Self-Compilation” in *FCCM*, 2019.
[3] A. Vaishnav et al., “FOS: A Modular FPGA Operating System for Dynamic Workloads,” *arXiv preprint arXiv:2001.09990*, 2020.