Contact Info

Name: Elias Kouskoumvekakis
Postal Address: Sarantapixou 77, Lycabetus, 11471 Athens, Greece
E-Mail: eliaskousk@gmail.com, eliask@di.uoa.gr
LinkedIn: http://www.linkedin.com/in/eliaskousk
Github: http://github.com/eliaskousk
Telephone No: +30-6907-598950, +30-210-6412077

GSoC 2016 Project: RISC-V port to Parallela (FOSSi Organization - Parallela)

Abstract

With this proposal I hope to benefit the open-source hardware enthusiast community with work related to the incredible Parallela board used by thousands of students and hobbyists around the world. This project will focus on the integration of the *RISC-V* rocket core, inside the *Zynq FPGA* device of *Parallela*. The *RISC-V* rocket core is an implementation of the *RISV-V* ISA that has gotten a lot of attention and support due to being clean, modular and power efficient. This project will allow owners of *Parallela* boards to write and execute *RISC-V* programs with minimal effort from their side. The system will work out of the box with a prebuilt binary image ready to be placed in an SD card and users will be able to re-build it with minimal effort. Moreover, a tutorial document will be created to aid inexperienced users make the most of this work and allow them to modify it for their own needs and purposes with custom hardware and / or software code.

Background

This document describes my proposal to the FOSSi organization for the *RISC-V* port to *Parallela*. Since *FOSSi* has the mission to promote and assist free and open digital hardware designs and their related ecosystems, it motivates me greatly to join their efforts and contribute to their cause. Coming from a pure software background with zero hardware knowledge, I know first-hand how difficult it was to get started with hardware design and this is my chance to further improve on this situation that is still somewhat true. I haven't been active with the *FOSSi* community until now but I was a regular user of *OpenCores* which as I understand will be succeeded by *FOSSi LibreCores*. With this project I hope to contribute to *FOSSi* and *Parallela* communities as much as I can and improve my skills at the same time.

I have actually used open source hardware IP cores from *OpenCores* and especially *Plasma MIPS CPU*, which my B.Sc. thesis was based some years ago and involved creating a simple teaching SoC with *Plasma*, *DDR*, *UART* and *VGA* functionality. Since then and while working on my M.Sc. I have participated in an EU funded project called *StochSoCs* (http://stochsocs.di.uoa.gr) that involved building *FPGA* accelerators for parallel stochastic simulations of large bio-models described in *SBML* format. The communication of the accelerator with its Host-PC was handled by a *PCIe* core that I built on top of existing *Xilinx* IP and 3rd party *PCIe* DMA. This improved my *FPGA* hardware knowledge further and got me thirsty for more. I am also really motivated for this project since I wanted to study and use both the Parallela and *RISC-V* for a couple of years but due to work I never had the chance. Now that my work with *StochSoCs* is done it is the time to deep dive into both of them!

Implementation Plan - Milestones

1. Build the GCC/Newlib toolchain and a companion Linux image for RISC-V from https://github.com/riscv/riscv-tools. The repository seems to have excellent documentation of all the steps involved and I already have experience building my own cross compilers for Intel SCC NoC many-core processor (48 P54C Pentium cores) and linking with static or dynamically libraries and their issues. At the end of the build we should have built the following RISC-V tools / components:

- GNU toolchain (GCC C/C++ compiler, GDB, GAS, LD)
- Spike ISA simulator for RISC-V
- Proxy kernel (alternative to Linux for single Newlib linked binaries)
- Linux kernel image
- BusyBox utilities binary
- Root disk image
- Frontend server library (for HTIF I/O interface service calls)

I can optionally build a Clang – LLVM toolchain and the rest of the RISC-V tools (opcodes, proxy kernel, tests & benchmarks). This step shouldn't take more than a few days of work.

2. Configure and generate a 64-bit RISC-V rocket core logic from https://github.com/ucb-bar/rocket-chip. I intend to generate code for two out of three targets supported (C++ emulator, Verilog for FPGAs, no Verilog for VLSI since this is an FPGA design).

The configuration will maybe need some file editing (related to variables in Configs.scala) to parameterize the rocket core with different cache size and / or associativity parameters, memory and I/O interfaces, FPU existence, TLB size.

I will use the C++ emulator to test and benchmark what I generated and also inspect its functionality with the optionally generated VCD waveforms using GTKWave Alternatively I could use Modelsim, Xilinx Isim or the open-source simulator tools Icarus Verilog or Verilator.

If time permits, I will consider a second RISC-V implementation like Pulpino's project RI5CY core from https://github.com/pulp-platform/riscv which is freely available in synthesizable SystemVerilog. This will allow us to have the ability to skip the Chisel intermediate layer and have more flexibility in our final system.

3. Create an IP block that will contain the above created RISC-V rocket core so that it can be placed in the OH repository and easily instantiated by others. The https://github.com/ucb-bar/fpga-zynq repository has good scripts that I can I can take parts of them to automate the process of integrating the RISC-V rocket core inside a Zynq IP block of my own. This can happen either by creating an IP-XACT definition file or directly use the Vivado toolset to create a Xilinx IP core (.xci file). I will then have to remove any Zedboard specific parts that are still present and not needed for our system. The HTIF-I/O and Mem-I/O that the RISC-V rocket chip uses to communicate with I/O and memory respectively should use the provided AXI interfaces to communicate with the Zynq ARM cores, memory controller and the rest of the Parallela components present in the Zynq FPGA.

4. Create a complete SoC by instantiating the RISC-V IP block created in 3 with other parallela/oh devices. The end result will be a complete SoC inside the Zynq FPGA device of Parallela. The minimum required devices besides RISC-V rocket core is the GPIO and a UART to communicate with the user and / or easily load programs on the RISC-V rocket core. The OpenCorors.org adv_debug_sys could be used for the latter. Another possibility would be to connect the SPI core from parallela/oh repository in order to have the ability to load programs from Flash memory during boot. It is important that a lot of tests happen here to validate the desired functionality between the components and the functionality of the system as a whole. More cores could be added if the above are stable and we have time, probably at the end of the project when everything else is finished.

5. Boot Linux on the RISC-V rocket core and run the Epiphany SDK host software that was originally meant to run on the Zynq ARM cores to communicate with other IP blocks and the Epiphany chip. On the E-SDK, e-hal is the most important software component for this task and maybe I will also need e-lib to interface the RISC-V rocket core with the rest of the hardware components. This would certainly require good familiarity with the E-SDK and porting the necessary code from ARM -> RISC-V. Hopefully it will run on RISC-V without any problems. If not, I should debug and attempt to fix any issues.

The next step would be to boot the Linux image created in milestone 1. With Linux booted I could then proceed to run the E-SDK software on top of it. This will allow me to run more advanced test programs to verify that all is well with the core and that it can communicate with the other parallela/oh hardware, at least the GPIO and UART for communication with the user.

6. Create a new IP block that interfaces the RISC-V rocket core to a simple 104 bit emesh interface, instead of AXI. This will allow communication with the Epiphany 16 core chip present on each Parallela board. This is the last step of the whole project and the details are not yet finalized with my prospective mentors. We will discuss this further during the weeks leading to the official start of work.

General Note: My workstation is pretty fast with quad-core Intel Core-i7 4790K, 32GB RAM and latest SSD and I have access to 2 8-core 5960X at my university office so build times for all milestones shouldn’t be a problem at all.

Deliverables

The goal with this project is to demonstrate open source integration of a RISC-V rocket core with software running on it and communication with other FPGA cores. An important aspect of the work is to have the system “run out of the box” and with good documentation. There are many universities using Parallela and for them to be productive in their research, it's not enough to have the code work, it must be also easy to use. With those things in mind, the deliverables of this project will be the following:

1. Source code for everything developed in hardware and software with good comments.
2. Build scripts to automate the building procedure of the final SoC without requiring to open Vivado GUI. The build scripts would generate a final binary image, optionally copying to the SD card of Parallela. This in short means the system will “run out of the box”.
3. Final binary image produced with the above scripts.
4. Complete documentation especially of what the build scripts do and what a determined user can try to change and what he cannot. This will be a tutorial style document for using the RISC-V rocket core inside the Zynq FPGA device and running simple example programs or custom hardware.
5. Optionally: If time permits maybe also create a very simple GUI application in Qt for users not experienced enough to use the console to run the top level build script.

Timeline

This is a timeline that I will use to stay on track with the above milestones.

Before official start of work (April 22 – May 22):

- Get up to speed with the Zynq FPGA device with enough small Vivado projects since I don’t have any experience with Zynq besides a very simple tutorial. I should focus on communication between the ARM cores and FPGA IP cores (Xilinx or custom).
- Learn everything important there is to learn about Parallela and mentioned inside the user guide, the SDK guide and the forums.
- Bond with the community and read as many Parallela FPGA topics as I can. Stay in constant touch with my mentors.
- Optional: Read the Chisel papers / tutorials from https://chisel.eecs.berkeley.edu/ and https://github.com/ucb-bar/chisel-tutorial to understand the rationale for creating it and learn how to modify Chisel (Scala) files. It will be good but hopefully not necessary knowledge for all UCB BAR related work and the tools they use, especially the Rocket-Chip core generator. Moreover, it seems cool enough to know for any future projects.

May 23 – May 29

- Build the RISC-V toolchain.
- Generate the RISC-V rocket core.
- There should be time for testing with a good HDL simulator and / or the companion C++ emulator.

May 30 – June 5

- Start the integration work of the RISC-V rocket core as mentioned in milestone 3.
- Use AXI buses to handle the out of core communication (HTIF-I/O, Mem-I/O).

June 6 – June 12

- Finish the IP block of the RISC-V rocket core inside Vivado.
- Possibly build an IP-XACT file to make the IP vendor neutral.
- Simulate the IP block and verify that it runs correctly with very simple programs.
- Make sure this RISC-V IP block and its toolchain can be automatically generated from a top level script and using only Github repositories for source.

June 13 – June 26

- Work on the SoC design that will contain the RISC-V rocket core and the supported devices from the parallela/oh repository described in milestone 4.
- Document progress thus far for the upcoming review.

June 27 – July 10

- Start work with milestone 5, by porting the necessary E-SDK software
- Test that the SoC can successfully boot Linux.
- Create and run programs that use the devices to test communication.

July 11 – July 17

- Finalize the complete SoC hardware and software design.
- Fix any bugs and instabilities.

July 18 – July 24

- Work on milestone 6 and build an IP block for the interface of the RISC-V rocket core with the Epiphany chip via the emesh interface.
- Test the communication between them.

July 25 – July 31

- Work on the automation of the whole build.
- This might be already in place and only need refinement.
- Perform more testing.

August 1 – August 7

- Write any remaining documentation of the project.
- The most important part are the tutorials for new users of Parallela that don’t have much experience with all the tools.

August 8 – August 23

- Since the project involves hardware I leave this period as a buffer for any unpredictable delays.
- If project is finished early I will use this time to refine the tests and the documentation (especially figures) of the whole project.