GSoC Phase-II Report

Linux Kernel Driver for Lattice MachXO2 programming/debugging

Swaraj Hota (@bluez_)

Mentor: Herbert Poetzl (@Bertl)

July 31, 2020

Contents

1 Introduction

The aim of this task (T729) was to make a Linux kernel driver to program/debug MachXO2 FPGAs (used as routing fabrics) in AXIOM Beta main board. This is a report of my progress in the *second* phase of GSoC 2020.

Two main goals of this project are:

- To implement an "upload" interface to program MachXO2
- To implement a debug interface for OpenOCD

2 Progress

By the end of the *first* phase I was able to successfully implement an upload interface to program the routing fabrics, along with sysfs attributes to read out idcode and status register bits. My milestone for the *second* phase (as I mentioned in my proposal) was to figure out a debug interface for OpenOCD and implement at least some debug functionalities by the end of the phase. I, unfortunately, was not able to meet this milestone due to some issues that came up and some underestimation on my part.

I spent most of this phase improving various aspects of the driver, added more useful sysfs entries, and spent much time tackling with issues that came up while implementing the mux switching between the two PICs using an existing kernel driver (pca954x). Devicetree entries are used to bind the mux driver as well as our rfdev driver. Although the mux driver binds successfully and creates two multiplexed buses for the two PICs, our rfdev driver could not be registered with those buses due to some reason which is still being debugged. Besides that I did a lot of reading on OpenOCD and a possible implementation for the debug interface. Herbert and I had a discussion with people on OpenOCD IRC, they suggested us to look into their current USB driver on OpenOCD side and also a proposed patchset for linux kernel about an ioctl based implementation [link]. So in the next phase I will be picking it up from there.

Some important issues that needed to be addressed before making the debug interface where taken care of. A way to reset the FPGAs through JTAG, parsing of status register, more abstract functions, two different FPGA manager instances, and sysfs entries like traceid, usercode, digest, statstr which will be useful for debugging, were all implemented. As we have different types of muxes/switches across different power board versions, we will need to address each of these in a proper way in the next phase as quickly as possible, before moving on to the most important part of the project i.e. the debug interface.

3 Phase Timeline

In this phase:

- *Week 1:*
  - Investigated a way to optimize the byte reverse function (did not find a suitable way)
  - Added sysfs entry to show md5 digest of the latest uploaded firmware
  - Read about OpenOCD and explored a possible debug interface
- *Week 2:*
  - Added code to parse status register bits, along with debug messages and a sysfs entry for human readable status (statstr)
  - Added basic operational state reporting through sysfs entry 'state' (needs to be improved)
  - Added two separate FPGA managers (currently both work for same FPGA)

- *Week 3 & 4:*
  - Added function to reset FPGA through JTAG (currently resets on driver exit)
  - Added sysfs entries to read out 'traceid' (unique MachXO2 id) and 'usercode'
  - Cleaned up code and added more abstract functions like JTAG command in and command out
  - Tried to add mux switching functionality through devicetree overlays (still ongoing)

4 Challenges Faced

This phase was quite challenging as I had to work with things that I was very unfamiliar with. First was to properly understand how OpenOCD works and how to go about making a debug interface. After a lot of reading and discussion with OpenOCD people I was finally able to imagine a possible implementation. The second big challenge was working with devicetree to properly load the mux driver along with our rfdev driver. This task is still ongoing and I have been stuck at it since more than a week. Herbert and I have been trying to debug what the issue could be, as of yet we suspect it could also be some bug in linux i2c subsystem, as everything seems to be in place and it should theoretically work but it just doesn't. Trying to find a suitable way to debug the devicetree was also a challenge as there is no good support and very little up-to-date documentation.

5 Modified Timeline

Major task in the next phase would also be to finally implement a debug interface for OpenOCD as I was not able to do it in this phase. Before that, the muxes/switches also needs to be taken care of.

In the upcoming phase:

- *Week 1*:
  - Fix the mux related issue where the driver does not get probed
  - Add support for the different muxes/switches across different power boards we have
- *Week 2*:
  - Change the PIC firmware to allow configuration of its JTAG port and possibly also reset the FPGAs, directly in response to a command byte
  - Finalize idea of the debug interface implementation (possibly the ioctl interface mentioned previously)
- *Week 3 & 4*:
  - Implement the debug interface for OpenOCD
  - Take care of other deferred work, clean up the code

6 In Conclusion...

The driver was improved in various aspects in this phase. I have been trying my best to keep up with the steep learning curve. I got stuck with unexpected issues yet again, but learned a lot along the way as well. Next phase is going to be even more challenging and I am not going to underestimate that this time. Nevertheless, even if for some reason I could not fully complete the project by the end of next phase, I will definitely work on it even after GSoC ends. I'll make sure the driver is fully usable and helps to accelerate the development process of routing fabrics.