UPDATING SODOR

INTRODUCTION

Currently sodor is written in out-dated Chisel2, complies to out-dated Priv ISA spec v1.7, not ported to FPGA

The following are the benefits that I am speculating by "updating sodor":
- I have seen that some individuals(from the mailing list and other sources) start off with sodor and then move to rocket possible reasons for this could be since they are quite new to chisel/RISC-V and starting directly with the large codebase of rocket(which includes a lot more than the instruction pipeline) could be difficult. Keeping sodor updated to latest standards of RISC-V ISA / Chisel would help ease their transition from sodor to rocket(which is continuously being updated to latest standards)
- It is already used for educational purpose so having it updated would help teach the latest content and also a better documentation would definitely help in having an even wider deployment in terms of number of schools using sodor in undergraduate computer architecture courses with the port of 3 stage on FPGA providing an even more complete overview of comp arch

PROJECT GOALS

- Updating to Chisel3/FIRRTL which requires making the Sodor testharness speak to Verilator (since Chisel3/FIRRTL can only generate Verilog, no more C++).
- Updating to Privileged Spec v1.9.1
- Use the External Debug to load programs into Sodor, like how Rocket works.
- Provide a Verilog test harness, and put the 3-stage on an FPGA(PYNQ-Z1) which would require an AXI port so that binaries can be loaded into Sodor's scratchpad also allowing others to use Sodor's cores in their project
- Add support for the ma_addr, ma_fetch ISA tests. This requires detecting misaligned address exceptions
- Make common/csr.scala file clearer and more understandable.
- Refactor the stall, kill, fencei, and exception logic of the 5-stage to be more understandable
- Documentation of codebase along with microarchitecture diagrams displaying data/control path for all 3,5 and ucode

IMPLEMENTATION

To port to Chisel3, Wiki with updated usage instructions and guide given at Chisel3-vs-Chisel2 would be appropriately utilized. Along with that I also plan to go through commit history of other projects which transited to Chisel3 like BOOM and Rocket-chip to get an overview of transition and if incase any undocumented changes need to be made.
Priv Spec v1.9.1 would require changes in the CSRs (CSR1.7->1.9.1) used along with that use of External Debug Module similar to (rocket-chip/debug) instead of HTIF atleast wrt sodor.
AXI wrapper/port to allow communicate/load binaries in the core's memory on FPGA using riscv-fesvr. AXI wrapper would allow others to use Sodor cores in their project
Provide in-depth documentation which would explain the execution flow for all the cores and also how most of it is implemented in Chisel3 since I am speculating that majority of users would be new to Chisel3. Also provide microarchitecture diagrams to summarize.

TIMELINE

Week 1-3 : Update to Chisel3
Week 4-5: Update to Priv Spec v1.9.1 as part of which Support for External Debug would be developed
Week 6-8: Implement a Verilog Test harness and test it on zynq
Week 9: Buffer (just if in case any of the above deadlines are missed)
Week 10: Support to detect misaligned address exceptions and cleanup/refactor few files to make them more understandable
Week 11-12: Document the codebase in separate latex(/doc) and develop microarchitecture diagrams

ABOUT ME

My Previous projects
I have also contributed to this repo with the aim to develop support for Princeton mode in 3 stage
https://github.com/ucb-bar/riscv-sodor/pull/20

1. **Communication between zc706 and host motherboard with intel i7-5960x via PCIe** - it involved making an appropriate block diagram in vivado[cdma(support for both scatter-gather and normal transfers was achieved),axi2pcie and many other IP's were required] device driver for host running Linux kernel 4.6 and a userspace application. Was able to achieve a read speed of 500MiB/s(wrt host).
2. modify already existing support for OoO processor simulating RISC-V ISA to a one with desired specifications which are currently not met just by changing the parameters in exported python file.
3. Implement a noise cancellation adaptive filter using TI TMS320C6748
4. IMPLEMENTING PERFECT HASHING FOR MAPPING XID TYPES TO LOADED PRINCIPALS Implemented hash function which used Jenkin's Lookup3 as the hash generating function and Hopscotch based collision resolution mechanism

I am pre-final year undergraduate pursuing B.E. Hons Electrical and Electronics Engineering at BITS PILANI K.K. BIRLA GOA CAMPUS.

Relevant Coursework: Embedded System Design, Digital Signal Processing, Analog & Digital VLSI, Microprocessor Programming and Interfacing, Digital Design

Relevant OpenCourseWork:

1. COMPUTER ARCHITECTURE 6.004.2 BY MIT ON EDX - Designed a 32-bit single cycle processor in MIT's tool Jade
2. ALGORITHMS AND DATA STRUCTURES BY STANFORD ON COURSERA - Implemented all the algorithms taught in the course in C

Alternative Email-Id : bhimanikritik@hotmail.com
Github : codelec
Contact # : +919819901417
LinkedIn : https://www.linkedin.com/in/kritikbhimani/
For IM : Available on Hangouts(bhimanikritik@gmail.com), also on IRC