NewCARD: A Vertically Integrated Teaching Tool for Microarchitecture

Mária Ďuračková
University of Edinburgh

Nigel Topham
University of Edinburgh

Background - How are practical exercises done in computer architecture classes?

Two types of teaching tools:

1. Low Level

   - focusing on HW microarchitectural implementation (e.g. using Chisel Or Verilog).

2. High Level

   - software simulation, which provides a good overview of the instruction execution.

   - architectural simulators such as gem5 or ChampSim.

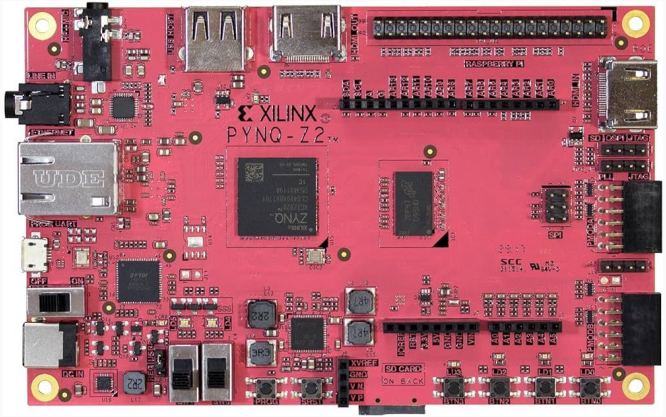
How it’s done in Edinburgh in the Computer Architecture and Design class (CARD)?

The computer lab has around 60 PYNQ-Z2 FPGAs.

We have 5-stage, in-order RISC-V core written in Verilog.

In the practical exercises, students implemplement different components of the RISC-V core (e.g. ALU, regfile, pipeline bypassing).

In the advanced tasks, students can choose an arbitrary way of optimising the core.



How can we improve it?

To measure the performance gain, students are limited to simulating the behaviour in software simulation.

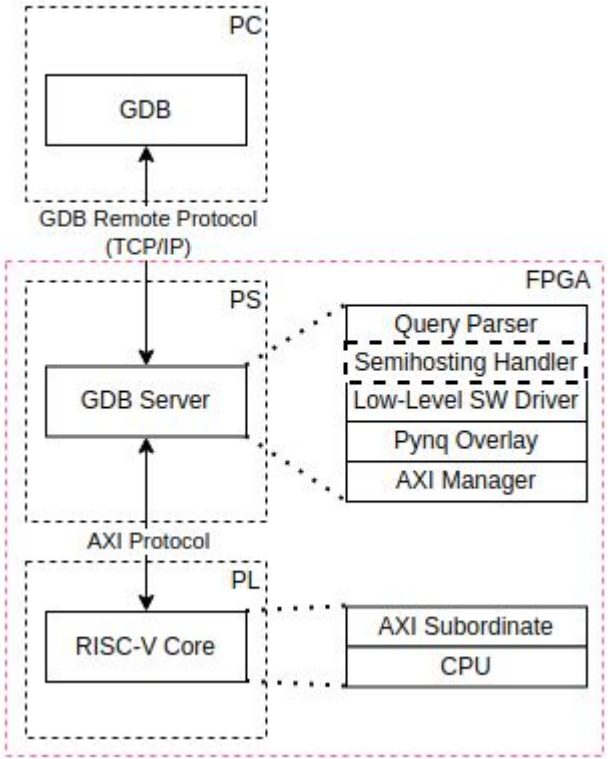
- **limited number of instructions** that can be executed.

By connecting the core implemented in HW to a debugger, students can run more advanced benchmarks and measure the performance more accurately on real HW.

+ **Opportunity to examine the insides of the core during execution**
+ **Opportunity to measure performance.**

… Insert NewCARD

System Design



- GDB runs on a regular PC and allows the student to:
  - Load an arbitrary program to memory.
  - Set breakpoints.
  - Run the program.
  - View and modify the state of the PC, registers and memory.
- GDB Connects to a Remote GDB Server, which acts as a front-end to the RISC-V core.
- Remote Server knows how to communicate with RISC-V core.
- Why is Remote Server not on the PC, but on the PS?
  - We're taking advantage of the AXI protocol.

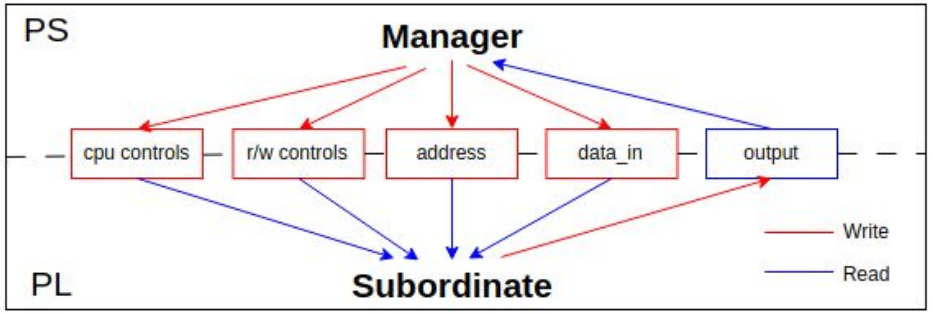
PS/PL Interface

Hardware interface (AXI)

Far more lightweight than JTAG with RISC-V debug specifications implemented.

a. Run/halt core
b. Instruction Single Step
c. Read/Write Registers
d. Read/Write PC
e. Read/Write Instruction memory
f. Read/Write Data memory

\+ EBREAK instruction



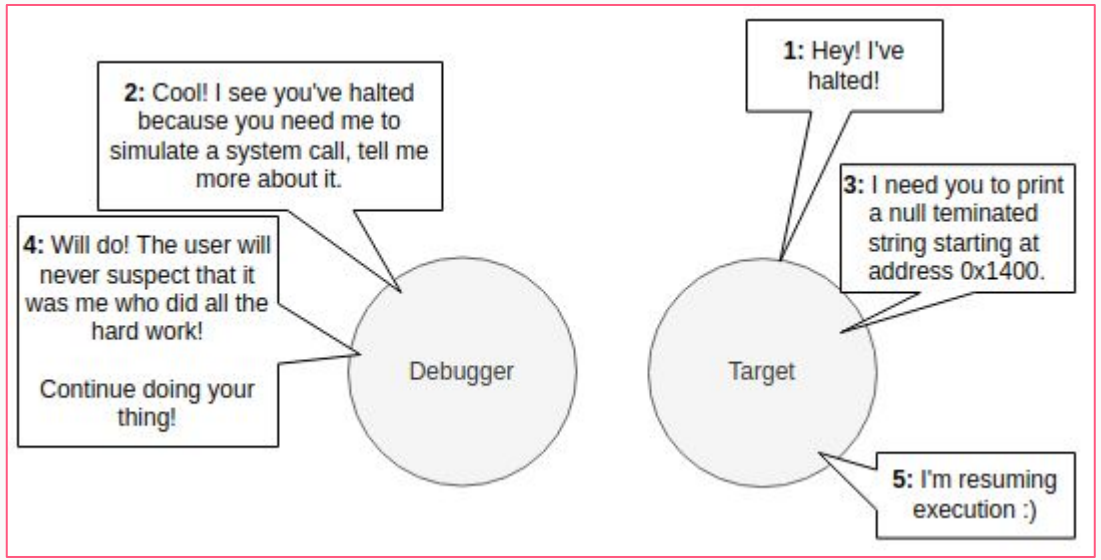
Software driver

Python Overlay (does a lot of the magic for you)

- Students supply a bitstream and hardware description file (from Vivado)
- The Python Overlay programs the board and creates the driver for you.

```
def write_DCCM(self, addr, data):
    self.halt_core()
    self.rv32_ip.write(0x0008, addr)
    self.rv32_ip.write(0x000c, data)
    self.rv32_ip.write(0x0004, 0b10000000)
```

Semi-hosting (What we do when there's no OS on the RISC-V)

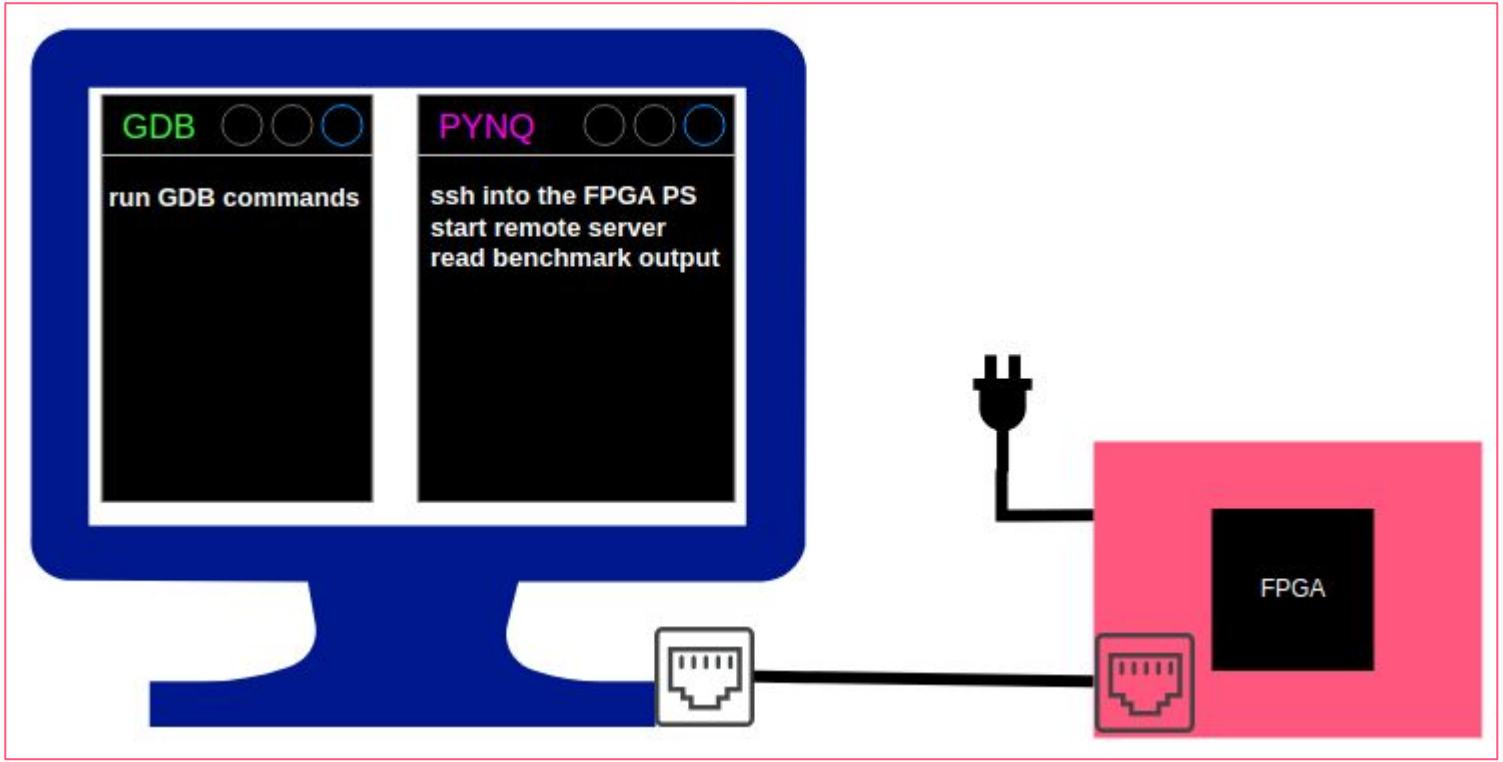


	time[microseconds]
arm cortex 9 / c	0.32040
arm cortex 9 / python	7.0792
RISC-V (semi-hosted) / c	1362.3

Single Character Print Comparison

- A way of simulating system calls on bare-metal cores.
- Uses the EBREAK instruction with special padding to indicate a semi-hosting call instead of regular breakpoint.
- Handled by the Remote Server.
- It's slow. But still good enough. (~26ms to print a 20 char long message)

User Setup


GDB
run GDB commands
PYNQ
ssh into the FPGA PS
start remote server
read benchmark output
FPGA

What can the students do?

Mini-studies in real hardware

- E.g. compare performance of different core implementations

	time [s]	DMIPS/MHz	normalised
baseline (no data forwarding)	10.25	0.9255	1
data forwarding	7.484	1.267	0.7301

The performance gain of adding pipeline forwarding on 1,000,000 iterations of the Dhrystone benchmark.

Experiment in GDB

```
(gdb) break *0xa10
Breakpoint 1 at 0xa10
(gdb) continue
Continuing.

Breakpoint 1, 0x00000a10 in sh_writei ()
(gdb) i r $pc
pc             0xa10    0xa10 <sh_writei>
(gdb) i r $sp
sp             0x0      0x0
(gdb) i r $ra
ra             0x19c    0x19c <main+316>
(gdb) x/4i $pc
=> 0xa10 <sh_writei>:    mv       a1,a0
   0xa14 <sh_writei+4>:  li       a0,64
   0xa18 <sh_writei+8>:  nop
   0xa1c <sh_writei+12>:         nop
```

Summary

NewCARD is a vertically integrated teaching tool which integrates both low-level HW implementation opportunities for students which high-level, experimentation and benchmarking framework.

It uses the GDB interface with Remote Server to debug and benchmark the RISC-V core.

To our best knowledge, it is the only tool which provides a high-level instruction execution overview on a core implemented in real hardware.

Thank you for your attention?

Questions and suggestions are much appreciated :)