DINO CPU
A Teaching-Focused RISC-V Design in Chisel

UC Davis

https://github.com/jlpteaching/dinocpu

Jason Lowe-Power
@JasonLowePower

Christopher Nitta
DINO CPU

A suite of RISC-V CPU designs
  Single cycle
  Five stage pipeline
  + Branch predictor
All designs can run rv32i code compiled with mainline GCC

A set of assignments
  One for each design

Tools for classroom use
  Chisel development
  Auto grading

Open source
  https://github.com/jlpteaching/dinocpu
Open source hardware construction language

Embedded in Scala

Main benefit: Parameterizable

Used in industry: SiFive, Google, IBM, others...
Why Chisel?

Vs Logisim

From an evaluation:
“\textit{I hate Logisim with a passion}”

Scaling designs difficult

Grading time consuming
Why Chisel?

Vs Verilog

- More modular design
  - Built-in unit tests
  - + easy to add auto grading

Scala-based
- More familiar (to me)
DINO CPU Design

Closely follows Patterson and Hennessy’s textbook’s design

Simple and modular
Not fast, small, synthesizable

Complete-ish
Hide complexity when possible
DINO CPU Assignment 1: R-types

Step 1: Logic for ALU control

```scala
/** *
 * The ALU control unit
 *
 * Input: add, if true, add no matter what the other bits are
 * Input: immediate, if true, ignore funct7 when computing the operation
 * Input: funct7, the most significant bits of the instruction
 * Input: funct3, the middle three bits of the instruction (12-14)
 * Output: operation, What we want the ALU to do.
 *
 * For more information, see Section 4.4 and A.5 of Patterson and Hennessy
 * This follows figure 4.12
 */

class ALUControl extends Module {
  val io = IO(new Bundle {
    val add    = In(Bool())
    val immediate = In(Bool())
    val funct7 = In(UInt(7.W))
    val funct3 = In(UInt(3.W))

    val operation = Output(UInt(4.W))
  })

  io.operation := 15.U // invalid operation

  // Your code goes here
}
```

The following table details the operation input and which values produce which results.

<table>
<thead>
<tr>
<th>funct7</th>
<th>rs2</th>
<th>rs1</th>
<th>funct3</th>
<th>rd</th>
<th>opcode</th>
<th>R-type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000</td>
<td>rs2</td>
<td>rs1</td>
<td>000</td>
<td>rd</td>
<td>011011</td>
<td>ADD</td>
</tr>
<tr>
<td>0100000</td>
<td>rs2</td>
<td>rs1</td>
<td>000</td>
<td>rd</td>
<td>011011</td>
<td>SUB</td>
</tr>
<tr>
<td>0000000</td>
<td>rs2</td>
<td>rs1</td>
<td>001</td>
<td>rd</td>
<td>011011</td>
<td>SLL</td>
</tr>
<tr>
<td>0000000</td>
<td>rs2</td>
<td>rs1</td>
<td>010</td>
<td>rd</td>
<td>011011</td>
<td>SLT</td>
</tr>
<tr>
<td>0000000</td>
<td>rs2</td>
<td>rs1</td>
<td>011</td>
<td>rd</td>
<td>011011</td>
<td>SLTU</td>
</tr>
<tr>
<td>0000000</td>
<td>rs2</td>
<td>rs1</td>
<td>100</td>
<td>rd</td>
<td>011011</td>
<td>XOR</td>
</tr>
<tr>
<td>0000000</td>
<td>rs2</td>
<td>rs1</td>
<td>101</td>
<td>rd</td>
<td>011011</td>
<td>SRL</td>
</tr>
<tr>
<td>0100000</td>
<td>rs2</td>
<td>rs1</td>
<td>101</td>
<td>rd</td>
<td>011011</td>
<td>SRA</td>
</tr>
<tr>
<td>0000000</td>
<td>rs2</td>
<td>rs1</td>
<td>110</td>
<td>rd</td>
<td>011011</td>
<td>OR</td>
</tr>
<tr>
<td>0000000</td>
<td>rs2</td>
<td>rs1</td>
<td>111</td>
<td>rd</td>
<td>011011</td>
<td>AND</td>
</tr>
</tbody>
</table>
```
DINO CPU Assignment 1: R-types

Step 1: Logic for ALU control

Step 2: Draw R-type circuit
Before implementation!
DINO CPU Assignment 1: R-types

Step 1: Logic for ALU control

Step 2: Draw R-type circuit
Before implementation!

Step 3: Write Chisel

```
val writereg = instruction(11,7)
registers.io.writereg := writereg
registers.io.wen := true.B

aluControl.io.add := false.B
aluControl.io.immediate := false.B
aluControl.io.Funct7 := instruction(31,25)
aluControl.io.Funct3 := instruction(14,12)
```
DINO CPU Assignment 2: Single-cycle

Given diagram:
- Implement control
- Wire control lines
- Wire whole datapath

Assignment takes one instruction type at a time
DINO CPU Assignment 3: Pipelined

Significant increase in complexity

Define all pipeline registers

Implement hazard and forwarding logic
DINO CPU Assignment 4: Branch prediction

Extensions!
We chose branch prediction

We updated pipeline

Students implemented two predictors

Ran benchmarks and compared results
Tools included

Singularity container
Many dependencies
Safe and secure

Gradescope scripts
Autograder
Open source
Everything on GitHub
https://github.com/jlpteaching/dinocpu
Assignments
Documentation
Source code
Tools
Future Improvements

Main feedback: Need better debugging

"How much time did this assignment take?"

Percent of Students

<table>
<thead>
<tr>
<th>Hours</th>
<th>&lt; 5</th>
<th>5 - 10</th>
<th>10 - 20</th>
<th>&gt; 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>WQ L2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQ L2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WQ L3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQ L3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Future Improvements

Main feedback: Need better debugging

More RISC-V support
  Privileged ISA for machine-mode rv32i (e.g., for embedded)

More assignments
  Non combinational memory + cache
  Multi-issue?

Open source community!
Questions/Comments?

Thanks to:
Students of ECS154B WQ19
Jared Barocsi, Filipe Eduardo Borges, Nima Ganjehloo, Daniel Grau, Markus Hankins, and Justin Perona

DINO CPU
A Teaching-Focused RISC-V Design in Chisel

https://github.com/jlpteaching/dinocpu

Jason Lowe-Power
@JasonLowePower
Christopher Nitta