Exercise 0 Introductory Lab & Chisel Lecture TDT4255 - 2023

Joseph Rogers, David Metz, Magnus Jahre

Outline

- Modern CPU Design in Practice
- Hardware Description Languages (HDL)
- What is Chisel
- Exercise 0 Overview
- Chisel Demo

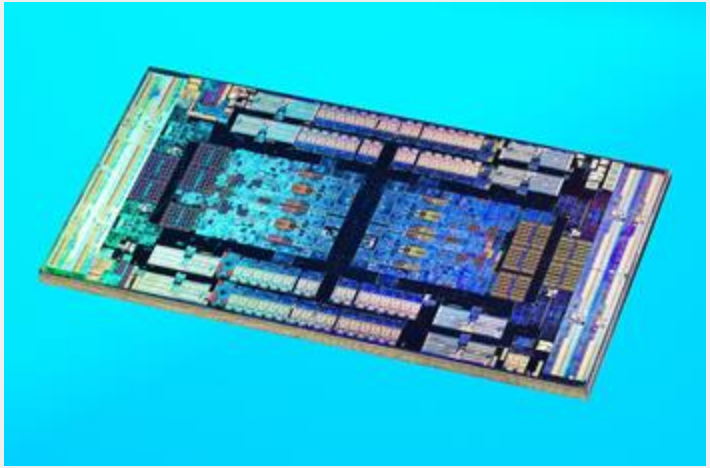
Please stop me to ask questions!

Housekeeping

- Check BB announcements
- Get on MS Teams & Piazza

Modern CPU Design in Practice

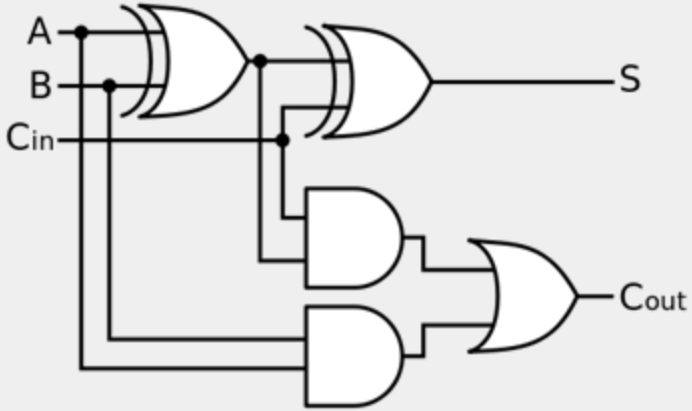
- CPUs are immensely complicated digital logic circuits.
- Modern designs can consist of millions, if not billions, of components.
- Do to this complexity, engineers have developed tools to assist with development, instead of drawing designs by hand.
- These take the form of Hardware Description Languages (HDLs).



2019 AMD Zen 2 64-Core EPYC Server CPU
8.34 Billion Transistors!!

Hardware Description Languages (HDLs)

- While programming languages are used to create programs, HDLs are used to create digital circuit designs (hardware!).
- When code from traditional programming languages gets compiled, it produces a binary. When HDL code is compiled, it produces RTL (Register-Transfer Level).
- RTL files are a representation of the actual circuits, they can be used by a manufacturer to make a real chip.



Full adder with carry-in and carry-out.

HDLs Cont...

VHDL and Verilog:

- Historically the two most popular HDLs. Developed in the 1980s, still widely used today.
- C-like syntax, easy to read, not very powerful.

A few cons…

- Can have a fair amount of boilerplate code.
- Lack of advanced language features, Oop, Fp, libs...
- Made for simulation – not everything is synthesizable

```
module toplevel(clock,reset);
  input clock;
  input reset;

  reg flop1;
  reg flop2;

  always @ (posedge reset or posedge clock)
    if (reset)
      begin
        flop1 <= 0;
        flop2 <= 1;
      end
    else
      begin
        flop1 <= flop2;
        flop2 <= flop1;
      end
endmodule
```

Two flip-flops in Verilog.

Chisel Programming Language

- New HDL implemented in Scala. Developed by University of California, Berkeley (2012). New version is Chisel 3, which we will be using.
- Designed for faster development time compared to other HDLs. Made possible by object-oriented, functional, parameterizable, etc… features, and a large set of libraries.
- Used to create RTL for RISC-V processors such as Rocket Core, which has been taped.
- Sometimes difficult to Google about due to an over enthusiastic woodworking community.

https://www.chisel-lang.org/
https://riscv.org/

Chisel RISC-V Example

PolarFire SoC Icicle Kit:

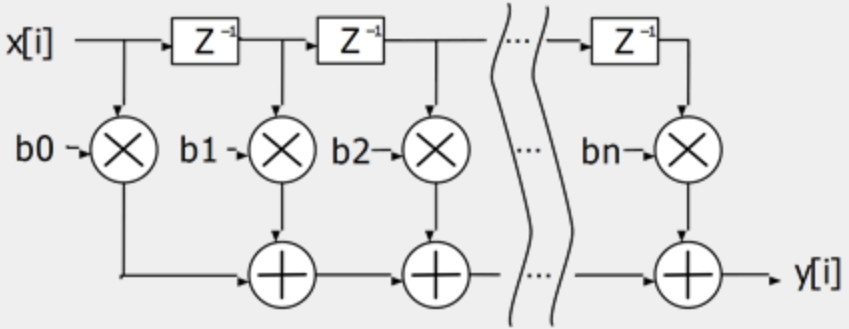
- Linux capable RISC-V CPU.
- Also has an FPGA.
- Similar to Xilinx Zynq, Ultra96 or RaspPi.

Chisel runs in Scala:

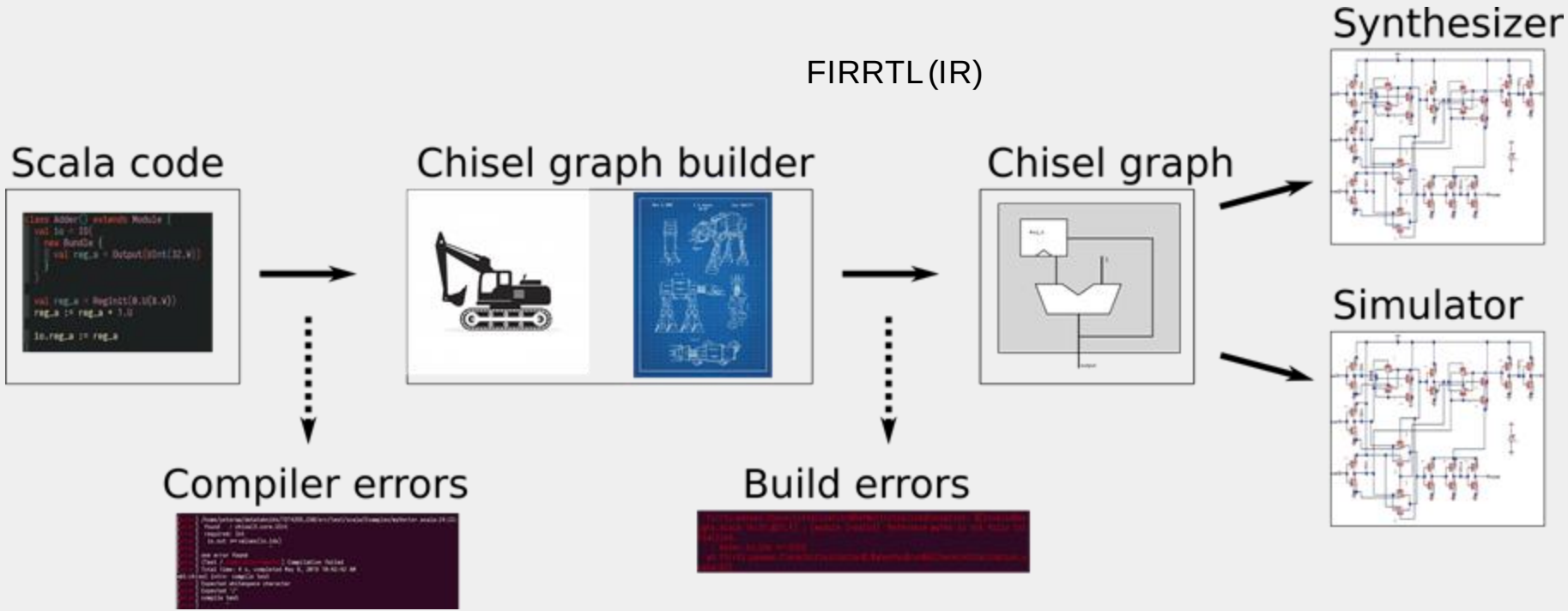
- OOP and FP features, strong statically typed, etc…
- Runs on the JVM, giving development portability.
- Leveraging these high-level features enables fast development, and **code reuse**.
- Distinguishing between generator (Scala) and hardware (Chisel) is important

Chisel Example - FIR Filter

```
class FirFilter(bitWidth: Int, coeffs:
Seq[UInt]) extends Module {

    val io = IO(new Bundle {

        val in = Input(UInt(bitWidth.W))

        val out = Output(UInt(bitWidth.W))

    })

        /* Implementation Here… */

}
```



Chisel Development Toolchain:



Essential Chisel Concepts:

- Wires
- Registers
- Modules

Essential Chisel Concepts, Part 1

Wires:

```
val wireA = Wire(UInt(4.W))
val wireB = Wire(UInt(4.W))
wireA := 2.U // wireA driven by a signal
wireB := wireA // wireA drives wireB
```

Essential Chisel Concepts, Part 2

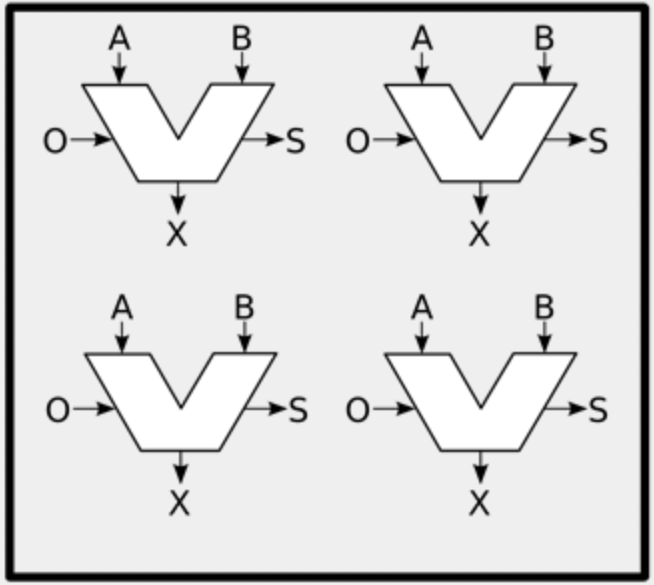
Registers (Simple Memory):

```
val regA = RegInit(2.U(4.W))
val regB = RegInit(1.U(4.W))
regA := regB
regB := regA
```

Essential Chisel Concepts, Part 3

Modules:

- Allows designs to be organized into discrete parts.
- Modules can contain sub-modules.
- Can be exported as libraries.
- Allows the for division of labor.



Submodules inside bigger module.

Things to Watch out for:

- Chisel is **NOT** a standalone programming language. It is an HDL embedded Scala.
- This means that you need to be clear on when you are writing Scala code vs Chisel code.
- In HDLs, all code lines are being executed **simultaneously**.
    - Except for things in not active when/otherwise clauses
    - Syntactic sugar around muxes

Exercise 0 Overview:

- Get familiar with basic HDL concepts
- Set up Chisel and get some basic examples working.
- Available at: https://github.com/EECS-NTNU/tdt4255-chisel-intro
    - Clone and push to a private repo – don't work in a public repo!
- Lab Session 8-10 Mondays in EL23 and 12-14 Wednesdays in R41.
- 20 Points, show solutions at the lab sessions before Sep. 20th
- Ask for help on Teams/Piazza!

Useful Resources:

- https://www.chisel-lang.org/ -- Official Chisel website
- https://mybinder.org/v2/gh/freechipsproject/chisel-bootcamp/master -- Official online Chisel bootcamp
- https://github.com/freechipsproject/chisel3 -- Official Chisel Github page
- https://github.com/schoeberl/chisel-book -- Great Chisel textbook
- https://people.eecs.berkeley.edu/~krste/papers/chisel-dac2012.pdf -- The original research paper introducing Chisel
- https://github.com/freechipsproject/chisel-cheatsheet/releases/latest/download/chisel_cheatsheet.pdf -- Chisel coding cheat sheet

Other Notes:

- We want feedback!
- If there are mistakes, let us know, we will try to fix them.
- It’s okay to help each other, please discuss exercises on Teams so that others in the course may benefit.
    - Discuss – not copy

Exercise 0 Demo