



Discussion 1



Office Hour

Thursday 2-3 pm

Room Kemper 3106 (3rd floor)





Goal today

1. Create a github Codespace for assignment 1
2. Introduction to Chisel
3. Create a simple hardware with Chisel
4. Test and debug the hardware
5. Introduction to assignment 1



Github CodeSpace

Assignment 1 page: <https://ilpteaching.github.io/comparch/modules/dino%20cpu/assignment1/>

Monthly included storage and core hours for personal accounts [↗](#)

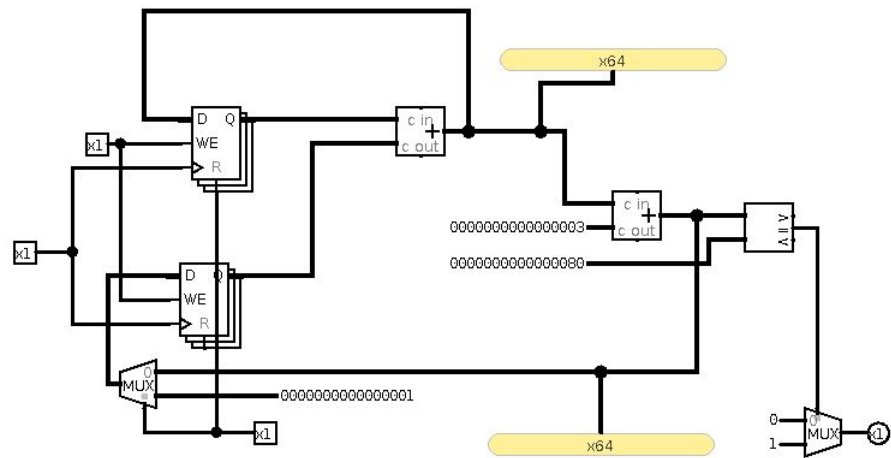
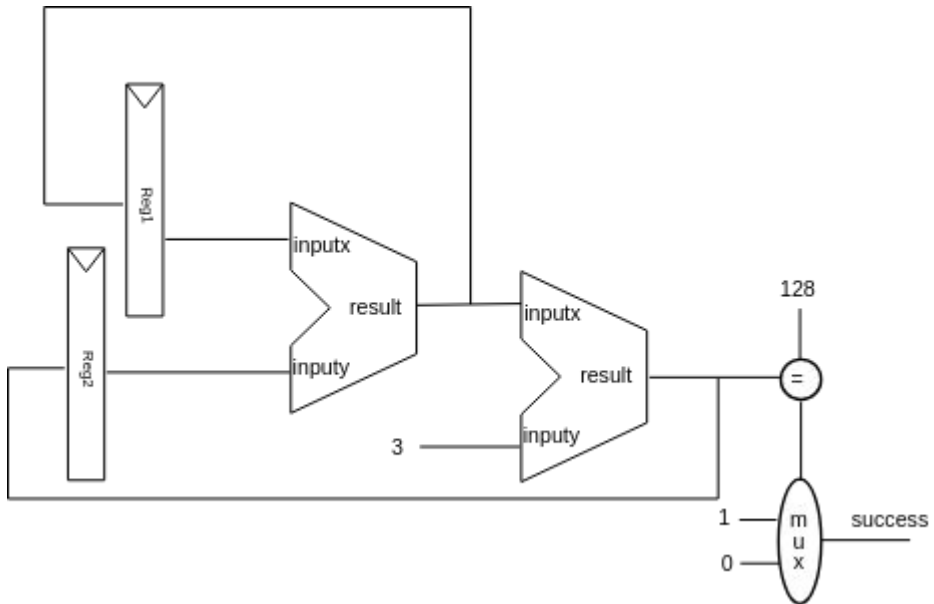
The following storage and core hours of usage are included, free of charge, for personal accounts:

Account plan	Storage per month	Core hours per month
GitHub Free for personal accounts	15 GB-month	120



What is Chisel

“Chisel, short for **Constructing Hardware in a Scala Embedded Language**, is an open-source hardware description language (HDL) used to describe digital electronics and circuits at the register-transfer level. It is based on Scala as an embedded domain-specific language (DSL), bringing the power of object-oriented and functional programming to type-safe hardware design and generation. Chisel facilitates advanced circuit generation and design reuse for both ASIC and FPGA digital logic designs. It enables agile, expressive, and reusable hardware design methodologies.”





Chisel Basics

Chisel Cheat Sheet:

<https://github.com/ECS154B-WQ24/dinocpu-assignment1/blob/main/documentation/chisel-notes/cheat-sheet.md>

Wires

```
val x = Wire(UInt())
```

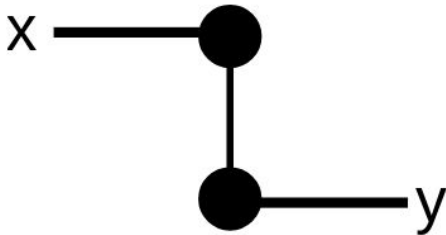


X 

Create a wire (named `x`) that is of type `UInt`. The width of the wire will be inferred. **Important:** this is one of the few times you will use `=`, and not `:=`.

Connect two wires

```
y := x
```



Connect wire `x` to wire `y`. This is "backwards" in that the input is on the right and the output is on the left. However, it's forwards in the way you say it out loud. (For the example above, think "`y` is connected to `x`.")



Muxes

Muxes

Mux

```
val x = Wire(UInt())  
  
x := Mux(selector, true_value, false_value)
```



This creates a wire `x` which will have the `true_value` on it if `selector` is true and the `false_value` otherwise.

When-elsewhen-otherwise

```
val x = Wire(UInt(3.W))  
  
when(value === 0.U) {  
  x := "b001".U  
} .elsewhen (value > 0.S) {  
  x := "b010".U  
} .otherwise { // value must be < 0  
  x := "b100".U  
}
```



The above creates a one-hot value on the wire `x` depending on whether the wire `value` is 0, greater than 0, or less than 0.



Types

Types

Boolean

- `Bool()` : a 1-bit value.
- `true.B` : to convert from a Scala boolean to Chisel, use `.B`.
- `false.B` : to convert from a Scala boolean to Chisel, use `.B`.

Integers

- `UInt(32.W)` : an unsigned integer that is 32 bits wide.
- `UInt()` : an unsigned integer with the width inferred. (You may get an error saying it can't infer the width.)
- `77.U` : to convert from a Scala integer to a Chisel unsigned int, use `.U`. (You may get type incompatible errors if you don't do this correctly.)
- `3.S(2.W)` : signed integer that is 2 bits wide (e.g., -1).
- `"b001010".U` : to create a binary literal, use a string of 1's and 0's starting with "b". Then, you can convert this string to an unsigned int with `.U` or a signed int with `.S`.

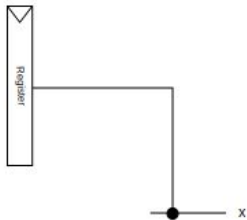
Registers

State elements (registers)

- `Reg(UInt(64.W))` : A 64-bit register
- `RegInit(1.U(32.W))` : A 32-bit register that has the value 1 when the system starts.

Registers can be connected to other wires.

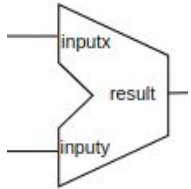
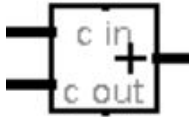
```
val register = Reg(UInt(32.W))  
x := register
```



This takes the value coming out of the register and connects it to the wire `x`.

Modules

Example: A 64 bit adder module

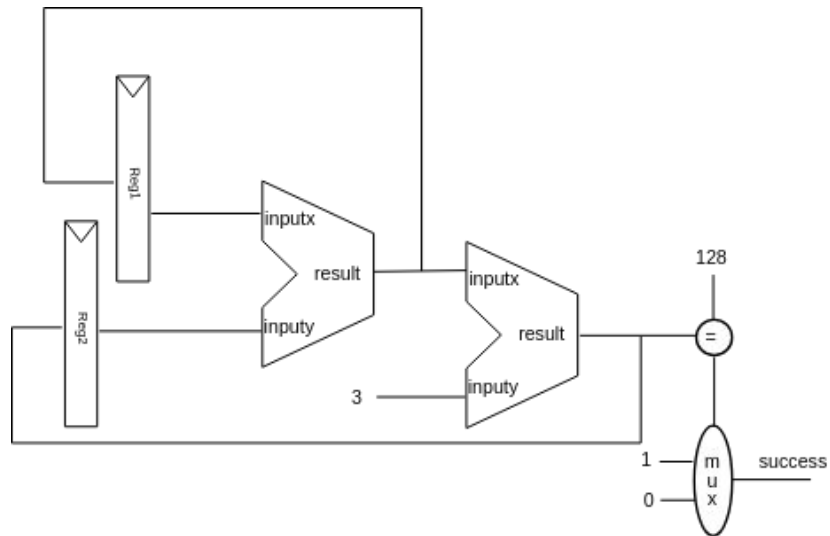


```
class Adder extends Module {  
  val io = IO(new Bundle{  
    val inputx = Input(UInt(64.W))  
    val inputy = Input(UInt(64.W))  
  
    val result = Output(UInt(64.W))  
  })  
  
  io.result := io.inputx + io.inputy  
}
```

Build your first hardware in Chisel

Documentation:

<https://github.com/ECS154B-WQ24/dinocpu-assignment1/blob/main/documentation/chisel-notes/first-hardware.md>

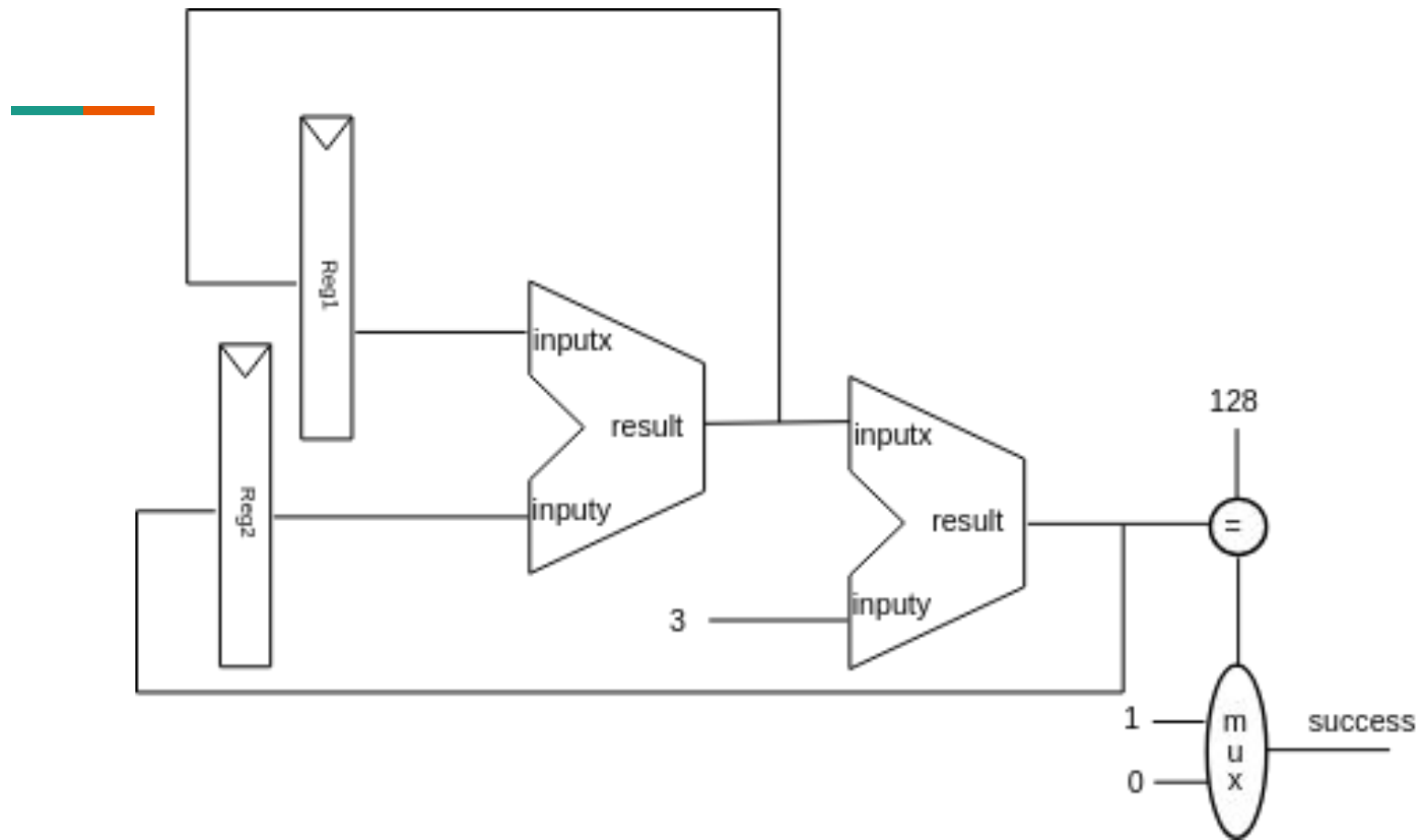




Now let's create the hardware

In the dinocpu assignment directory,

go to ``src/main/scala/simple.scala``



Assignment 1

Part 1: Draw out your hardware design

Part 2: Use Chisel to describe it

