2020-1 컴퓨터 조직론

Project 1. Getting Started

Juhee Kim

System & software security Lab.
Seoul National University

Apr. 10, 2020

Table of Contents

Project Goals

- Learn how to use Chisel
- Learn how to run tests and debug Chisel
- Make your first hardware with Chisel

Step 1: Getting Started with Chisel

- Chisel Overview
    - **Chisel** is a hardware design language that facilitates **advanced circuit generation and design reuse for both ASIC and FPGA digital logic designs**.

- Chisel References
    - Chisel Homepage
      https://www.chisel-lang.org/
    - Chisel Wiki
      https://github.com/freechipsproject/chisel3/wiki

Step 1: Getting Started with Chisel

- Datatypes in Chisel

```
Bool()                  // Bool literals (1-bit)
  true.B
  false.B

UInt()                  // unsigned int, width inferred
UInt(32.W)              // 32-bit unsigned decimal
  1.U                   // unsigned decimal 1-bit literal
  5.U                   // unsigned decimal 3-bit literal
  8.U(4.W)              // 4-bit unsigned decimal, value 8
 “b001010”.U            // 6-bit binary literal

SInt()                  // signed int, width inferred
  5.S                   // signed 4-bit literal
 -8.S                   // signed negative 4-bit literal
```

You may get an error if it can’t infer the width

```
// variable declaration

val b = true.B
val sint = 3.S(4, W)
```

Step 1: Getting Started with Chisel

- Wires
  - **Create** a new wire

use '=' operator to create a new variable

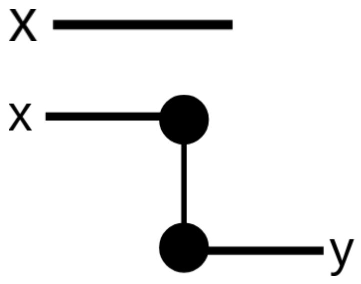
```
val x = Wire(UInt())
```

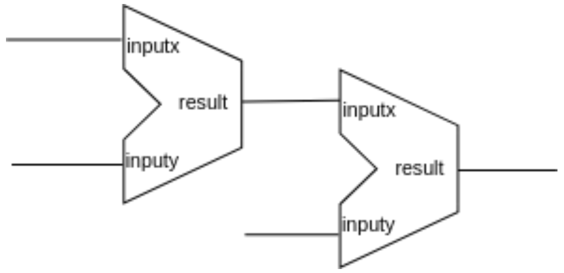
  - **Connect** two wires

```
y := x
```

```
val adder1 = Module(new Adder())
val adder2 = Module(new Adder())

adder2.io.inputx := adder1.io.result
```



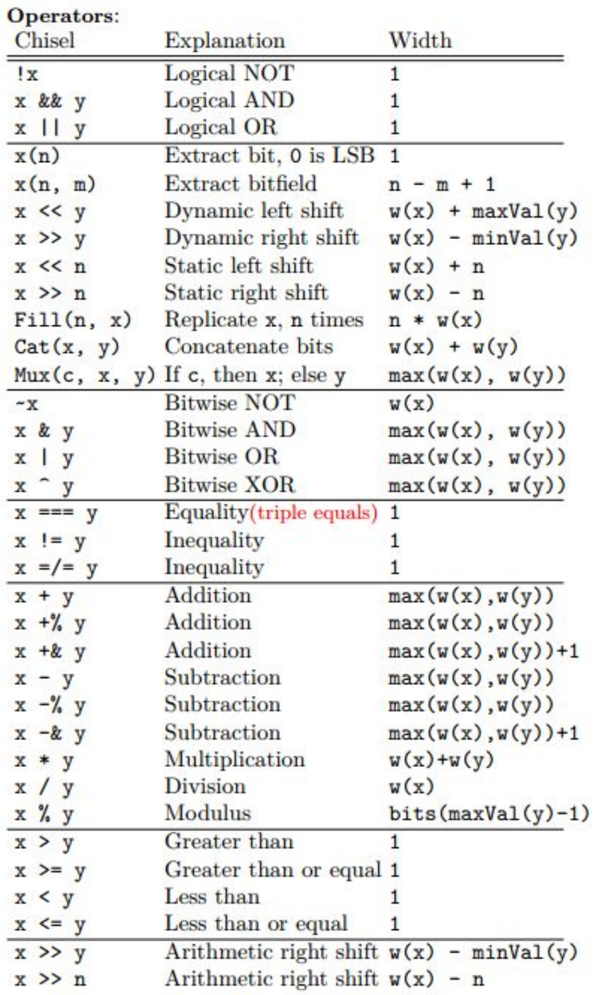


Step 1: Getting Started with Chisel

- Getting parts of a wire

```
val x = Wire(UInt(32.W))
val lower_5 = x(4,0)
val top_3 = x(31,29)
val rd_in_riscv_instruction = x(11,7)
```

- Built-in Operators →
    - *Equality is '==='.*



Operators:

Chisel	Explanation	Width
`!x`	Logical NOT	1
`x && y`	Logical AND	1
`x || y`	Logical OR	1
`x(n)`	Extract bit, 0 is LSB	1
`x(n, m)`	Extract bitfield	`n - m + 1`
`x << y`	Dynamic left shift	`w(x) + maxVal(y)`
`x >> y`	Dynamic right shift	`w(x) - minVal(y)`
`x << n`	Static left shift	`w(x) + n`
`x >> n`	Static right shift	`w(x) - n`
`Fill(n, x)`	Replicate x, n times	`n * w(x)`
`Cat(x, y)`	Concatenate bits	`w(x) + w(y)`
`Mux(c, x, y)`	If c, then x; else y	`max(w(x), w(y))`
`~x`	Bitwise NOT	`w(x)`
`x & y`	Bitwise AND	`max(w(x), w(y))`
`x | y`	Bitwise OR	`max(w(x), w(y))`
`x ^ y`	Bitwise XOR	`max(w(x), w(y))`
`x === y`	Equality(triple equals)	1
`x != y`	Inequality	1
`x =/= y`	Inequality	1
`x + y`	Addition	`max(w(x),w(y))`
`x +% y`	Addition	`max(w(x),w(y))`
`x +& y`	Addition	`max(w(x),w(y))+1`
`x - y`	Subtraction	`max(w(x),w(y))`
`x -% y`	Subtraction	`max(w(x),w(y))`
`x -& y`	Subtraction	`max(w(x),w(y))+1`
`x * y`	Multiplication	`w(x)+w(y)`
`x / y`	Division	`w(x)`
`x % y`	Modulus	`bits(maxVal(y)-1)`
`x > y`	Greater than	1
`x >= y`	Greater than or equal	1
`x < y`	Less than	1
`x <= y`	Less than or equal	1
`x >> y`	Arithmetic right shift	`w(x) - minVal(y)`
`x >> n`	Arithmetic right shift	`w(x) - n`

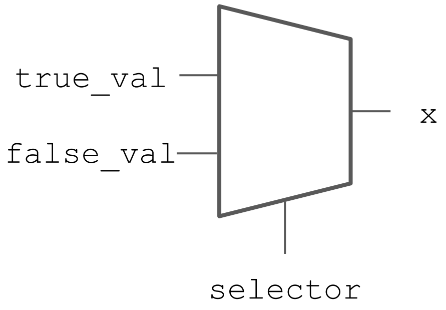
Step 1: Getting Started with Chisel

- Muxes

```
val x = Wire(UInt())
x := Mux(selector, true_val, false_val)
```

```
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"
}
```



Don’t forget ‘.’ in front of ‘.elsewhen’ and ‘.otherwise’.

Step 1: Getting Started with Chisel

- Muxes
    - switch-is statement for logic table

You cannot have a "nested" switch-is statement. (use Mux or when statements instead.)

input	output
0001	true
0100	false
0101	true
1101	true

```
// since we aren’t fully specifying the
// output, Dontcare is requried.

output := DontCare

switch(input) {
  is (“b0001”.U) { output := true }
  is (“b0100”.U) { output := false }
  is (“b0101”.U) { output := true }
  is (“b1101”.U) { output := true }
}
```

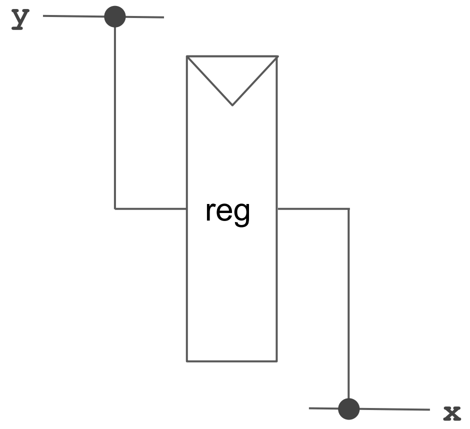
Step 1: Getting Started with Chisel

- State elements (Registers)

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

This will set the register to the value to on wire y at the end of the clock cycle

```
// initializing register
val register = RegInit(false.B)
```



Step 1: Getting Started with Chisel

- Bundles
  - group a set of wires together. (similar with struct in C)

```
class Complex extends Bundle {
  val real = SInt(32.W)
  val imag = SInt(32.W)
}
```

```
val wire = Wire(new Complex())          // create a wire with Complex type
val myreg = Reg(new Complex())          // create a register with Complex type

wire.real := 3.S                                   // set each component of the wire
wire.imag := -5.S                       // or, set the entire bundle to 0
wire := 0.U.asTypeOf(new Complex)       // using an explicit cast

wire := myreg                           // connect the entire bundle of wires to
                                        // other objects.
```

Step 1: Getting Started with Chisel

- Modules

```
class Mux2 extends Module {
  val io = IO (new Bundle {
    val sel = Input(UInt(1.W))
    val in0 = Input(UInt(1.W))
    val in1 = Input(UInt(1.W))
    val out = Output(UInt(1.W))
  })
  io.out := (io.sel & io.in1) | (~io.sel & io.in0)
}
```

```
class Mux4 extends Module {
  val io = IO(new Bundle {
    val in0 = Input(UInt(1.W))
    val in1 = Input(UInt(1.W))
    val in2 = Input(UInt(1.W))
    val in3 = Input(UInt(1.W))
    val sel = Input(UInt(2.W))
    val out = Output(UInt(1.W))
  })
  val m0 = Module(new Mux2)
  m0.io.sel := io.sel(0)
  m0.io.in0 := io.in0
  m0.io.in1 := io.in1

  val m1 = Module(new Mux2)
  m1.io.sel := io.sel(0)
  m1.io.in0 := io.in2
  m1.io.in1 := io.in3

  val m3 = Module(new Mux2)
  m3.io.sel := io.sel(1)
  m3.io.in0 := m0.io.out
  m3.io.in1 := m1.io.out

  io.out := m3.io.out
}
```

Step 2: Create your First Chisel Hardware

- Download the project file from http://compsec.snu.ac.kr:40404/downloads/proj1_hw.tar.gz
- SimpleSystem Preparation
  - Create a new file (from the base directory)

```
touch src/main/scala/simple.scala
```

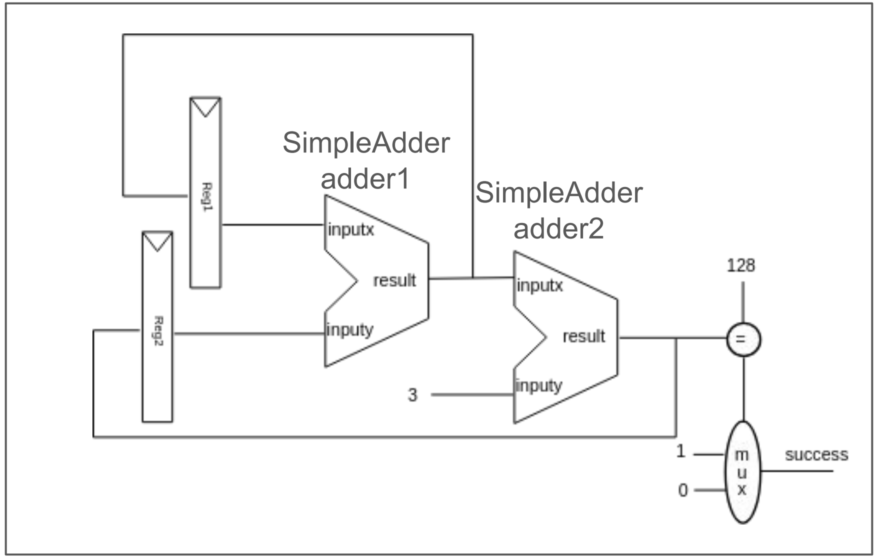
  - Import the chisel libraries. Add the following lines to the `src/main/scala/simple.scala`

```
package dinocpu
import chisel3._
import chisel3.util._
```

Step 2: Create your First Chisel Hardware

- System configuration
  - Module SimpleAdder
    - Input
      - inputx: 32bit unsigned
      - inputy: 32bit unsigned
    - Output
      - result: 32bit unsigned
  - Module SimpleSystem
    - Goal: described in the figure.
    - Output
      - success: bool
    - Variables
      - adder1, adder2: SimpleAdder Modules

SimpleSystem



Step 3: Simulating and testing the circuit

- Run singularity and test the circuit
  - Run singularity

```
singularity run library://jlowepower/default/dinocpu
```

You should see the following:

```
...
[info] Loading settings for project root from build.sbt ...
[info] Set current project to dinocpu (in build file:...)
[info] sbt server started at ...
sbt:dinocpu>
```

Step 3: Simulating and testing the circuit

- Run singularity and test the circuit
  - Run test

```
sbt:dinocpu> test
```

```
[info] Compiling 1 Scalar source to ...
...
test SimpleSystem Success: 0 tests passed in 15 cycles taking
0.016592 seconds
[info] [0.007] RAN 10 CYCLES PASSED
...
[info] All tests passed
```

You can add `printf` to `SimpleSystem` to see what's going on inside it.

```
printf(p"reg1: $reg1, reg2: $reg2, success: ${io.success}\n")
```

Evaluation Criteria

- Results for 10 cycles should be same with below.

```
reg1: 1, reg2: 4, success: 0
reg1: 5, reg2: 8, success: 0
reg1: 13, reg2: 16, success: 0
reg1: 29, reg2: 32, success: 0
reg1: 61, reg2: 64, success: 1
reg1: 125, reg2: 128, success: 0
reg1: 253, reg2: 256, success: 0
reg1: 509, reg2: 512, success: 0
reg1: 1021, reg2: 1024, success: 0
reg1: 2045, reg2: 2048, success: 0
```

Submission

- Deadline: 2020. 04. 17 금요일 23:59

- 제출 방법: 서버에 본인이 구현한 **src 디렉토리를 그대로 복사해서** 업로드
    - 제출 서버: 147.46.114.111 (port 40404) ID: arch-<학번> / PW: dinocpu

    - **서버 패스워드 바꾸기**
    - $ ssh arch-<학번>@147.46.114.111 -p 40404
    - $ passwd