ハードウェア記述言語Chiselを
もっと活用するためのDiplomacy概説

Chisel & Diplomacy Deep Dive

FPGA開発日記 著者 : msyksphinz

FPGA Development Diary Author

@msyksphinz_dev

https://msyksphinz.hatenablog.com

RISC-V Day Tokyo 2020 (Nov.5 - Nov.6)

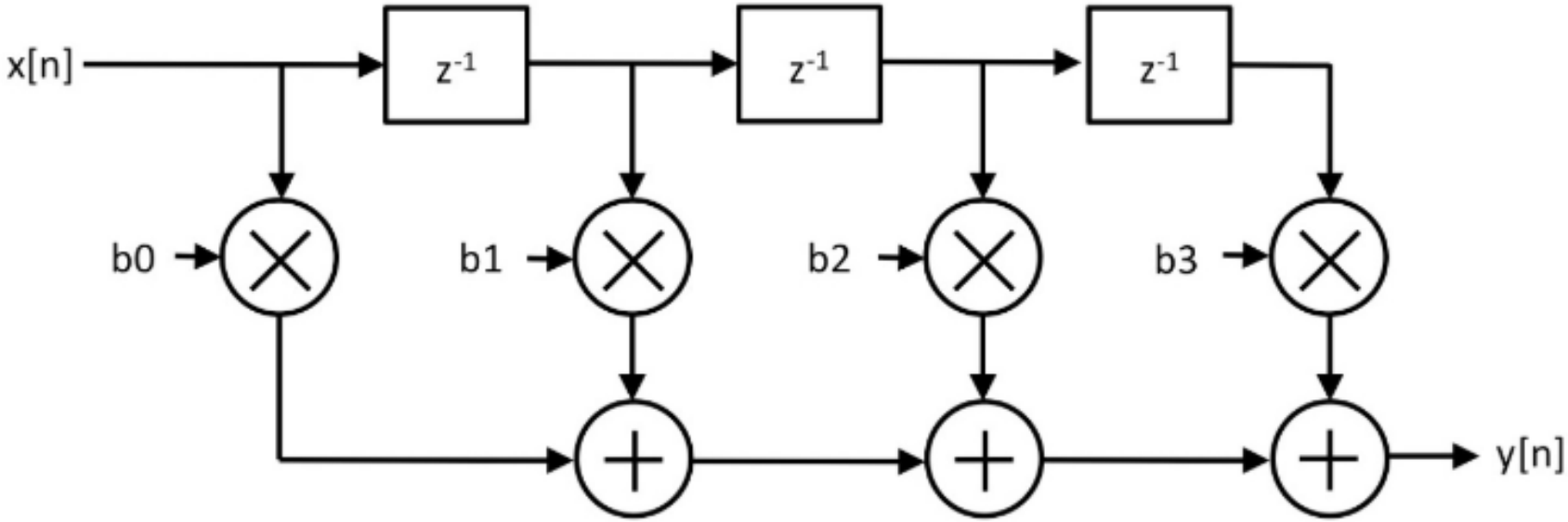
Do you use Chisel?

- Hardware Construction language based on Scala.
  - Not High-level synthesis Language
- Used in SiFive's RISC-V IP
- Rocket-Chip
  - https://github.com/chipsalliance/rocket-chip
- BOOM
  - https://github.com/riscv-boom/riscv-boom

What part of Chisel is Good?

Software-like description using Scala syntax.
Many Verilog engineers get confused with “unique” description from Scala.

FIR Filter: How to write it?



Written by Sophisticated Chisel Engineer

```
class MyManyDynamicElementVecFir(length: Int) extends Module {
  val io = IO(new Bundle {
    val in = Input(UInt(8.W))
    val valid = Input(Bool())
    val out = Output(UInt(8.W))
    val consts = Input(Vec(length, UInt(8.W)))
  })


  val taps = Seq(io.in) ++ Seq.fill(io.consts.length - 1)(RegInit(0.U(8.W)))
  taps.zip(taps.tail).foreach { case (a, b) => when (io.valid) { b := a } }

  io.out := taps.zip(io.consts).map { case (a, b) => a * b }.reduce(_ + _)
```

https://github.com/freechipsproject/chisel-bootcamp/blob/master/2.5_exercise.ipynb

Why you use Chisel?

- In recent years, the period during technology is "state-of-the-art" is:
  - Much shorter than the latency required for engineer’s design.
- Can it take two or three years to make an AI chip?
  - High mix, Low volume is one of the trends in DSA era.
  - Agile hardware design is a big importance of the future HW design.
- What is the “complexity” of Hardware Design?

設計中悪いんだけどバスの
スレーブ数1つ増えたから
Hey, spec is changed. # of bus slave is changed.

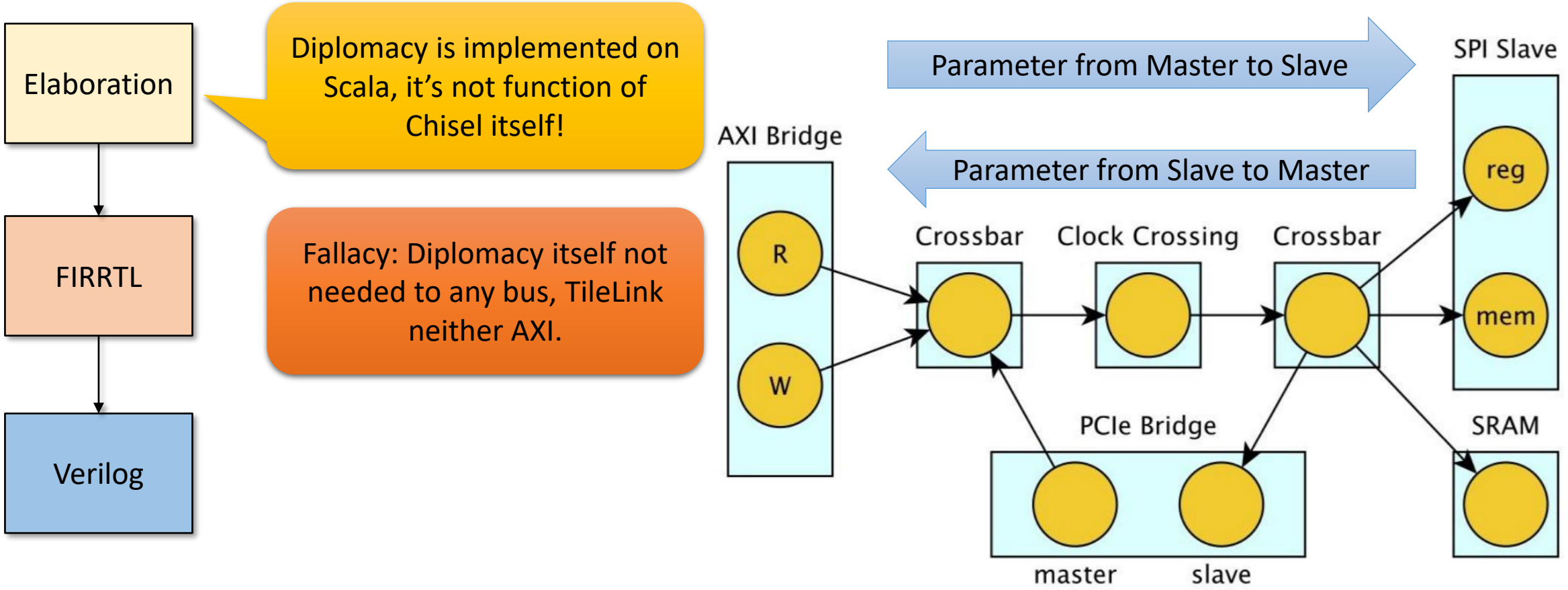
え... じゃあバスArbiterもう一回
接続し直さなきゃ
いけないんじゃないですか...
Oh, it means we must redesign bus arbiter again ...?

線が一本増えるだけでしょ？（違うわ）
よろしくたのむよ
You only adding one signal. Only few minutes you need.

How to Manage complex Module Networks in Design?

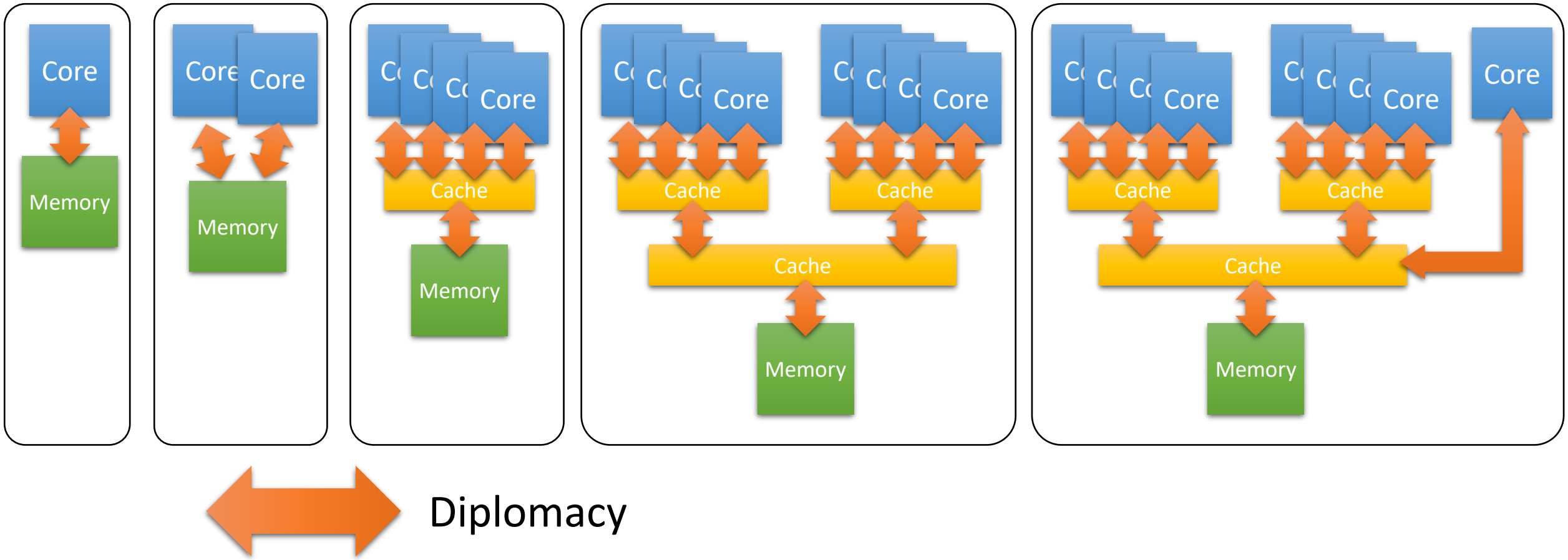
Parameter Passing with Diplomacy

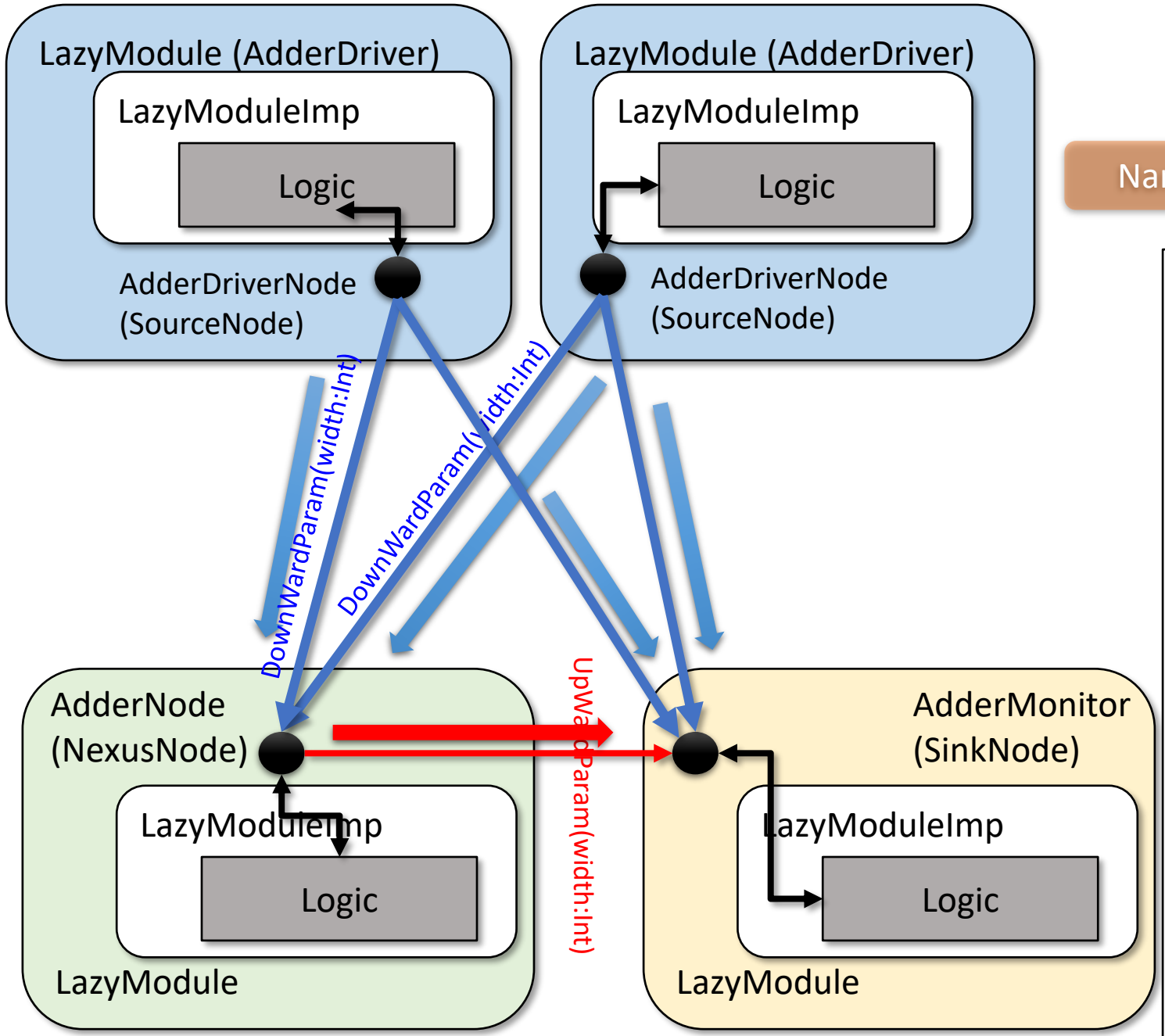
Parameter Negotiation Framework for generating parameterized protocol implementation.



https://carrv.github.io/2017/slides/cook-diplomacy-carrv2017-slides.pdf

How to realize rapid HW development?
Many HW configuration from one design, using Diplomacy!





Node: Connecting to each node

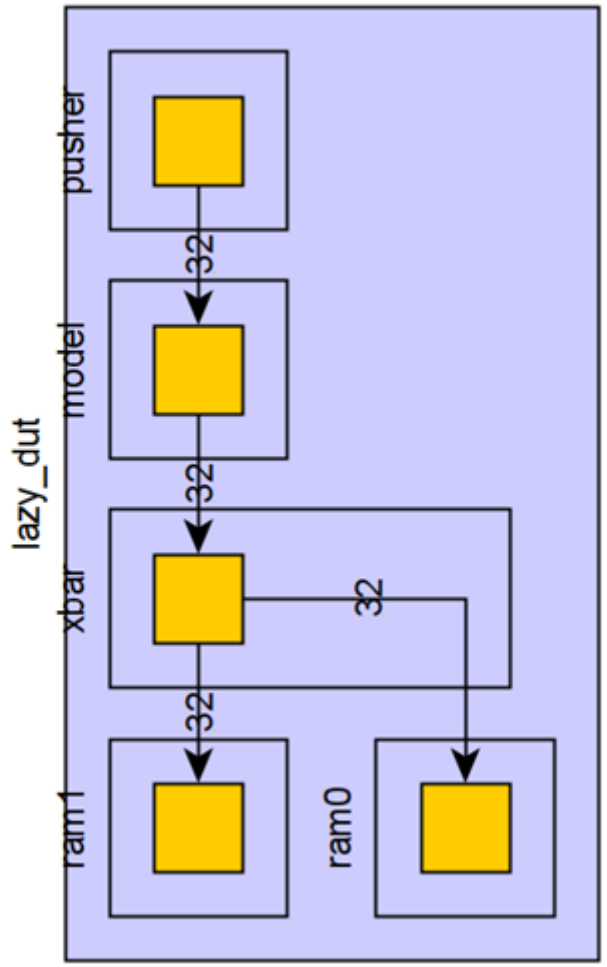
Name of LazyModule

LazyModuleImp

```scala
class AdderDriver(width: Int, numOutputs: Int)
    (implicit p: Parameters) extends LazyModule {
  val node = new AdderDriverNode(Seq.fill(numOutputs)
                                 (DownwardParam(width)))

  lazy val module = new LazyModuleImp(this) {
    // check that node parameters converge after negotiation
    val negotiatedWidths = node.edges.out.map(_.width)
    require(negotiatedWidths.forall(_ == negotiatedWidths.head),
            "outputs must all have agreed on same width")
    val finalWidth = negotiatedWidths.head

    // generate random addend
    // (notice the use of the negotiated width)
    val randomAddend = FibonacciLFSR.maxPeriod(finalWidth)

    // drive signals
    node.out.foreach { case (addend, _) =>
                       addend := randomAddend }
  }

  override lazy val desiredName = "AdderDriver"
}
```

Using GraphML for Diplomacy

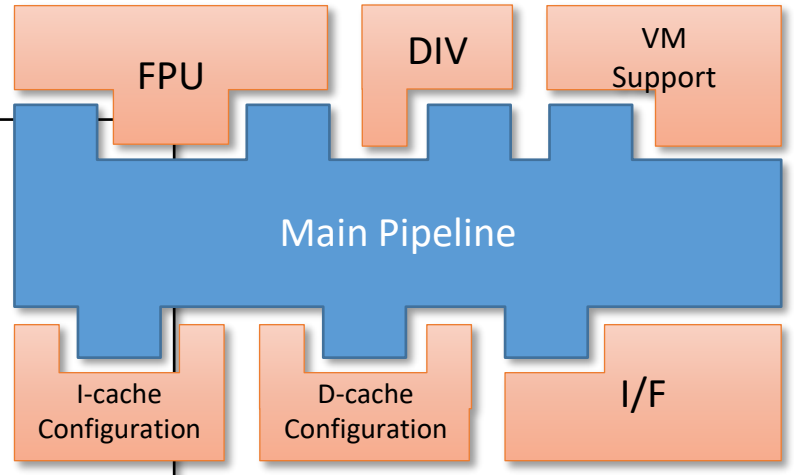
```
object Generator {
    final def main(args: Array[String]) {
        val p = (new Default2Config).toInstance
        // val p = (new Default1Config).toInstance
        Driver.emitVerilog(
            new TestHarness()(p)
        )
        ElaborationArtefacts.files.foreach { case (extension, contents) =>
            val f = new File(".", "TestHarness." + extension)
            val fw = new FileWriter(f)
            fw.write(contents())
            fw.close
        }
    }
}
```



Configurable Parameterization using Config/Parameter

Rocket-Chip Configuration Files : (main/scala/system/Configs.scala)

```
class BaseConfig extends Config(
  new WithDefaultMemPort() ++
  new WithDefaultMMIOPort() ++
  new WithDefaultSlavePort() ++
  new WithTimebase(BigInt(1000000)) ++ // 1 MHz
  new WithDTS("freechips,rocketchip-unknown", Nil) ++
  new WithNExtTopInterrupts(2) ++
  new BaseSubsystemConfig()
)

class DefaultConfig extends Config(new WithNBigCores(1) ++ new WithCoherentBusTopology ++ new BaseConfig)

class DefaultBufferlessConfig extends Config(new WithBufferlessBroadcastHub ++ new DefaultConfig)
class DefaultSmallConfig extends Config(new WithNSmallCores(1) ++ new WithCoherentBusTopology ++ new
BaseConfig)
class DefaultRV32Config extends Config(new WithRV32 ++ new DefaultConfig)

class DualBankConfig extends Config(new WithNBanks(2) ++ new DefaultConfig)
class DualCoreConfig extends Config(new WithNBigCores(2) ++ new WithCoherentBusTopology ++ new BaseConfig)
class DualChannelConfig extends Config(new WithNMemoryChannels(2) ++ new DefaultConfig)
class EightChannelConfig extends Config(new WithNMemoryChannels(8) ++ new DefaultConfig)

class DualChannelDualBankConfig extends Config(
  new WithNMemoryChannels(2) ++
  new WithNBanks(4) ++ new DefaultConfig)

class RoccExampleConfig extends Config(new WithRoccExample ++ new DefaultConfig)
```



Parameter Passing with “Config”

```
class Default1Config extends Config(
  new Bus32BitConfig ++ new IfuConnectConfig
)

class Default2Config extends Config(
  new BaseConfig ++
  new Bus64BitConfig ++
  new IfuNotConnectConfig
)

class Bus64BitConfig extends Config((site, here, up) => {
  case BusWidthBytes => 64 / 8
  case AddrSize => 0x200 * here(BusWidthBytes)
})

class IfuConnectConfig extends Config((site, here, up) =>
{
  case ConnectIfu => true
})
```

```
object Generator {
    final def main(args: Array[String]) {

        val p = (new Default2Config).toInstance

        Driver.emitVerilog(
            new TestHarness()(p)
        )
```

```
case object BusWidthBytes extends Field[Int]
case object ConnectIfu extends Field[Boolean]
case object AddrSize extends Field[Int]

class core_complex(txns: Int)(implicit p: Parameters)
extends LazyModule {

  val xbar    = LazyModule(new TLXbar)
  val memory = LazyModule(new TLRAM(AddressSet(0x0,
p(AddrSize)-1), beatBytes = p(BusWidthBytes)))
```

Parameter Passing with “Config”

```
class Default1Config extends Config(
  new Bus32BitConfig ++ new IfuConnectConfig
)

class Default2Config extends Config(
  new BaseConfig ++
  new Bus64BitConfig ++
  new IfuNotConnectConfig
)

class Bus64BitConfig extends Config((site, here, up) => {
  case BusWidthBytes => 64 / 8
  case AddrSize => 0x200 * here(BusWidthBytes)
})

class IfuConnectConfig extends Config((site, here, up) =>
{
  case ConnectIfu => true
})
```

```
object Generator {
    final def main(args: Array[String]) {

        val p = (new Default2Config).toInstance

        Driver.emitVerilog(
            new TestHarness()(p)
        )
```

```
case object BusWidthBytes extends Field[Int]
case object ConnectIfu extends Field[Boolean]
case object AddrSize extends Field[Int]

class core_complex(txns: Int)(implicit p: Parameters)
extends LazyModule {

  val xbar    = LazyModule(new TLXbar)
  val memory = LazyModule(new TLRAM(AddressSet(0x0,
p(AddrSize)-1), beatBytes = p(BusWidthBytes)))
```

Today’s Example

https://github.com/msyksphinz-self/minimal-diplomacy

Based-on Chisel 3.2
No need to download Rocket-Chip (Libraries are come from Maven)