

Chisel in a NutShell

**Institute of Computing Technology
Chinese Academy of Sciences
(ICT, CAS)**

**Kaifan Wang Huaqiang Wang
2021.6**

Outlines

- NutShell: a brief introduction
- Chisel develop experience sharing
 - Chisel in NutShell: Examples
 - Hints for Chisel beginners

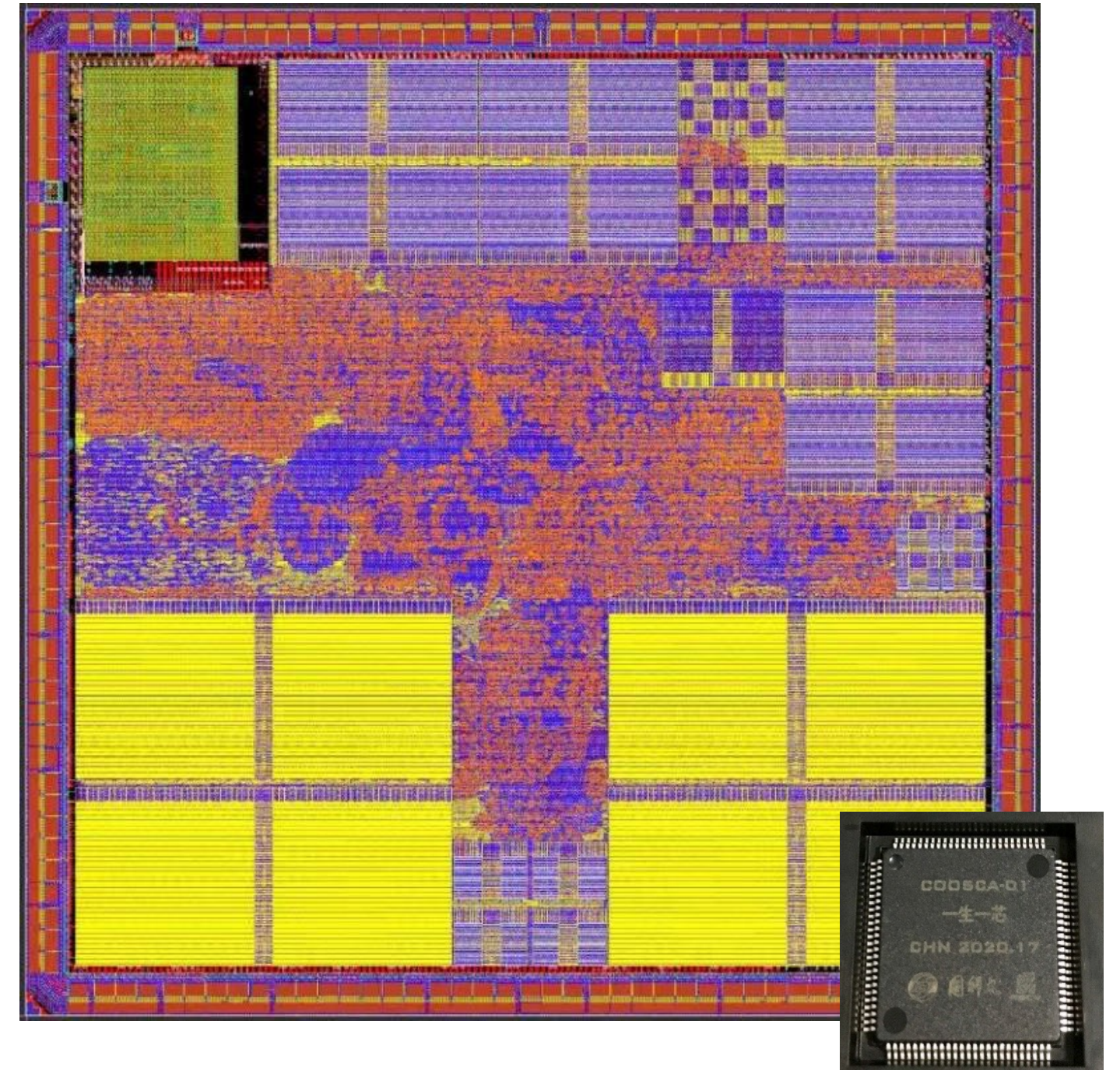
NutShell: a brief introduction

NutShell: Overview

- Developed by **5 undergraduates** in 4 months
 - Use online differential testing framework to speed up development
- Support SDRAM, SPI flash, UART_[1]
- Support **Linux** 4.18.0 Kernel

- Support **Debian 11 / Fedora 32** in simulation/FPGA environment

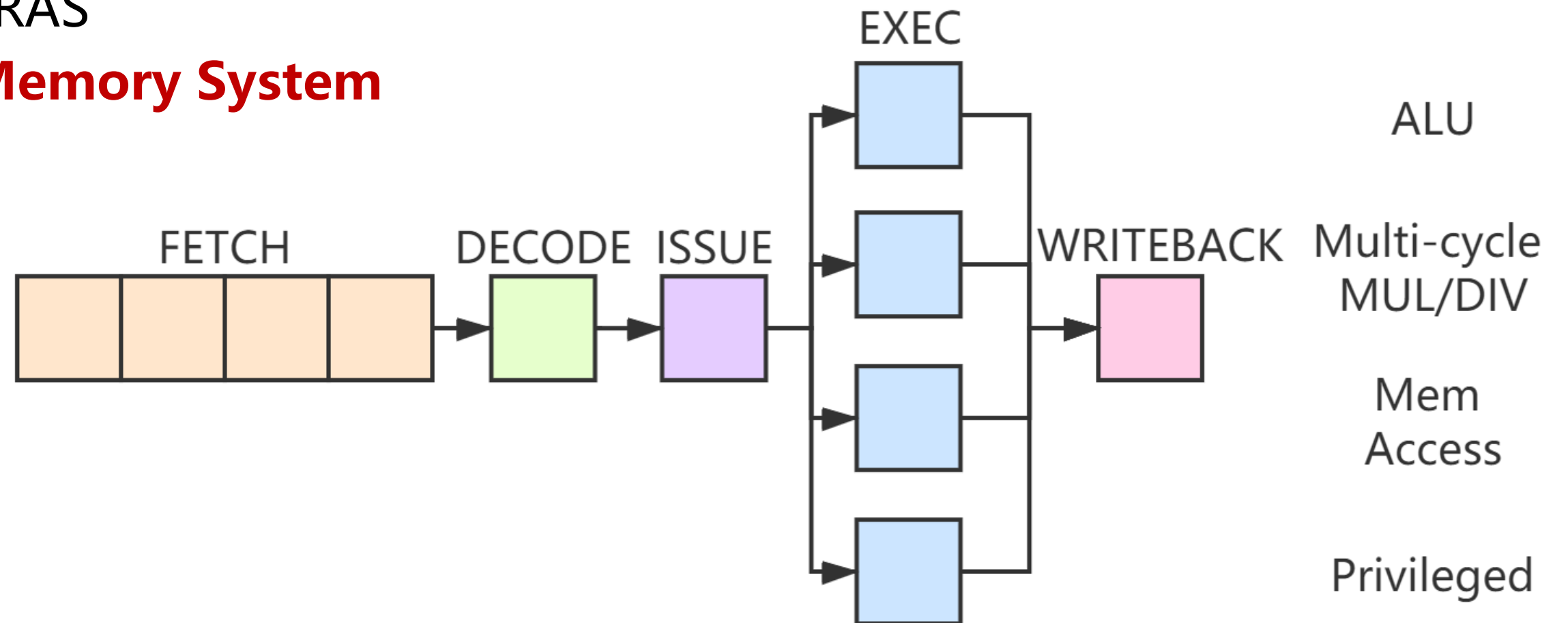
- SMIC 110nm
- 10mm²
- **200mw@350MHz** Typical
- TQFP100



[1] We used peripheral devices from OpenCores community.

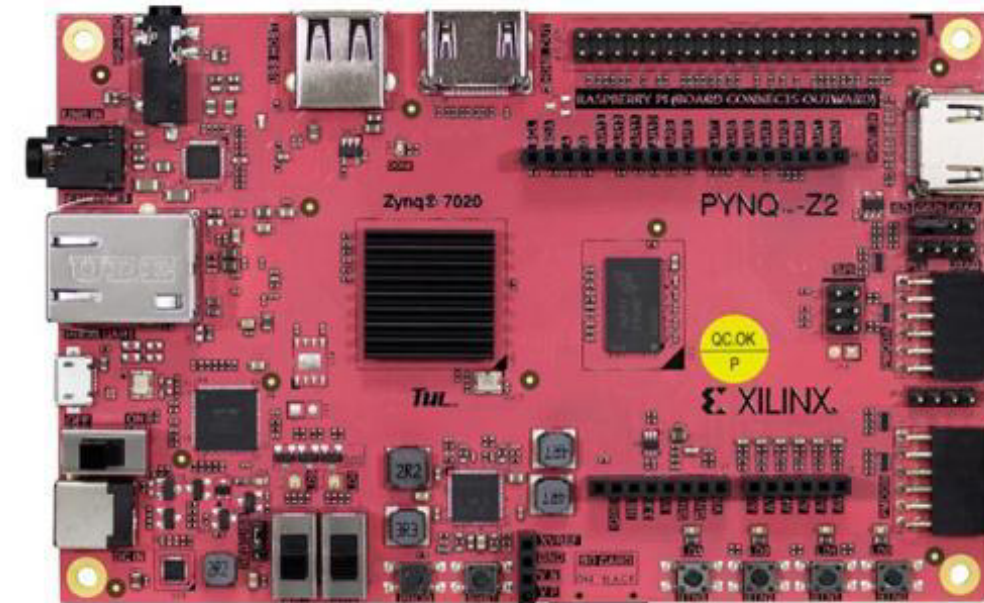
NutShell: Processor Core

- **Single-issue in-order core**
- **Developed in Chisel**
- Support RV64IMAC, Zifence, Zicsr
- Support M/S/U Mode
- 2-bit saturation counter for branch prediction
- 512 entry BTB, 16 entry RAS
- **Support Sv39 Virtual-Memory System**
- L1/L2 Cache support



NutShell: Frequency

- Only optimize TLB and compressed instruction decoder specially:
 - xc7z020-1-clg400 **60MHz**
 - xczu3cg-sfvc784-1-e **200MHz**
 - SMIC 110nm **350MHz**



Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	0.525	14	14	217	system_top_i..._0/CLKARDCLK	system_top_i...RBWRADDR[8]	15.203	4.945	10.258
Path 10	0.687	14	14	217	system_top_i..._0/CLKARDCLK	system_top_i...RARDADDR[6]	15.165	4.945	10.220
Path 11	0.687	14	14	217	system_top_i..._0/CLKARDCLK	system_top_i...RARDADDR[6]	15.165	4.945	10.220
Path 9	0.686	14	14	217	system_top_i..._0/CLKARDCLK	system_top_i...RARDADDR[6]	15.162	4.945	10.217

Vivado timing report for Pynq board

* Source: <http://xilinx.eetrend.com/d6-xilinx/article/2018-11/13898.html>

Chisel in NutShell: Examples

Use MaskedRegMap to Generate CSRs

- To implement RISC-V CSR:
 - Different CSRs have different read/write **side effects**
 - CSR addresses are **not consecutive**
 - Different read / write **mask**
- We designed MaskedRegMap abstract

```
MaskedRegMap(  
    reg address, reg data source (hardware),  
    writemask, write side effect,  
    readmask, read side effect  
)
```


Use MaskedRegMap to Generate CSRs

- MaskedRegMap: usage

MaskedRegMap(
 reg address, reg data source,
 writemask, write side effect,
 readmask, read side effect
)

```
1 //fu/CSR.scala
2 //实例化控制和状态寄存器
3 val mstatus = RegInit(UInt(XLEN.W), "h00001800".U)
4 val mie = RegInit(0.U(XLEN.W))
5 val medeleg = RegInit(UInt(XLEN.W), 0.U)
6 .....
7 val mapping = Map(
8   MaskedRegMap(Mstatus, mstatus, "hffffffffffffffff".U, mstatusUpdateSideEffect),
9   MaskedRegMap(Misa, misa),
10  MaskedRegMap(Medeleg, medeleg, "hbbff".U),
11  .....
12 )
13 .....
14 MaskedRegMap.generate(mapping, addr, rdata, wen, wdata)
15 val isIllegalAddr = MaskedRegMap.isIllegalAddr(mapping, addr)
```

Implement CSRs in NutShell

Use MaskedRegMap to Generate CSRs

- How to implement MaskedRegMap
 - **MaskedRegMap.apply()**
 - generate reg setting
 - MaskedRegMap.generate()
 - use reg settings to generate real circuit

```
def apply(  
  addr: Int, reg: UInt,  
  wmask: UInt = WritableMask, wfn: UInt => UInt = (x => x),  
  rmask: UInt = WritableMask  
) = (addr, (reg, wmask, wfn, rmask))
```

Use MaskedRegMap to Generate CSRs

- How to implement MaskedRegMap
 - MaskedRegMap.apply()
 - generate reg setting
 - **MaskedRegMap.generate()**
 - **use reg settings to generate real circuit**

```
chiselMapping.map { case (a, r, wm, w, rm) =>
  if (w != null && wm != UnwritableMask)
    when (wen && waddr === a) {
      r := w(MaskData(r, wdata, wm))
    }
}
```

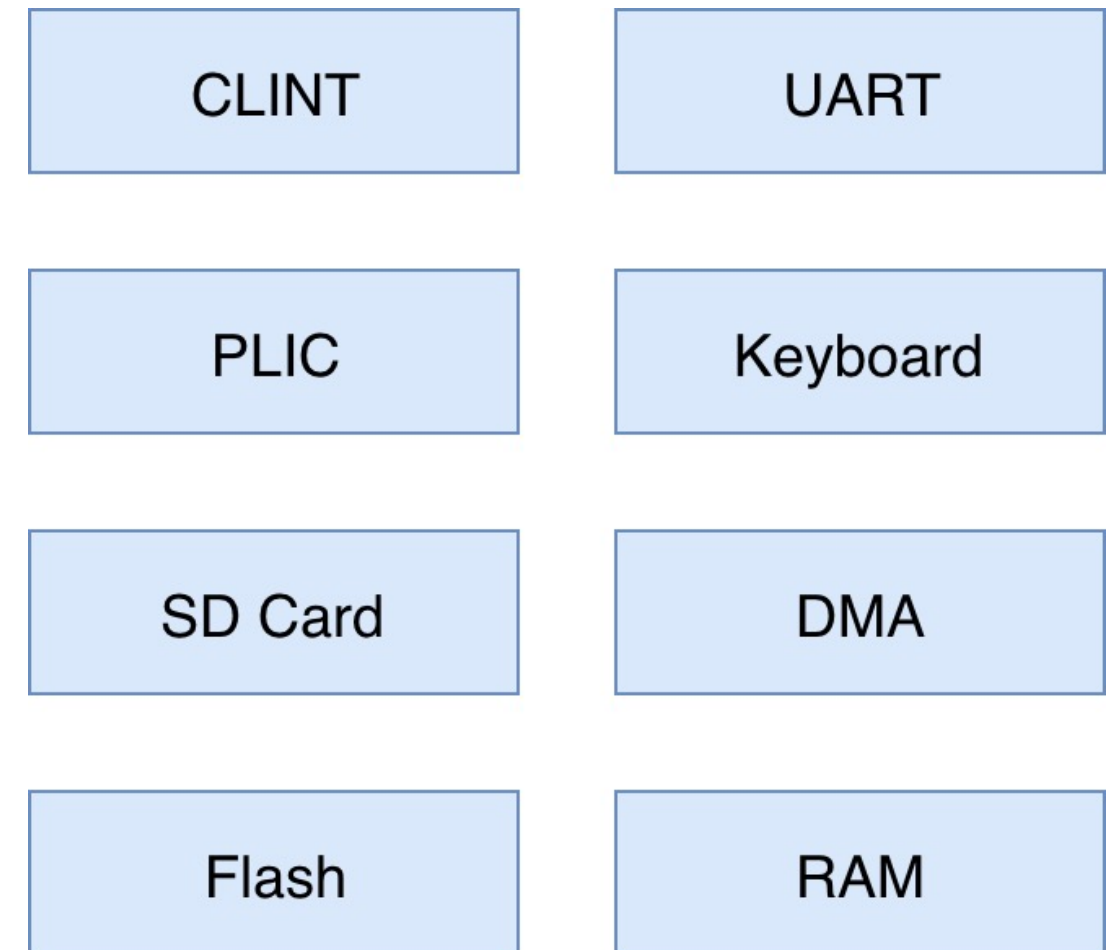
Use MaskedRegMap to Generate CSRs

```
object MaskedRegMap { // TODO: add read mask
  def Unwritable = null // You, 9 months ago • chore(CSR): substitute RegMap in CSR with MaskedRegMap
  def NoSideEffect: UInt => UInt = (x=>x)
  def WritableMask = Fill(64, true.B)
  def UnwritableMask = 0.U(64.W)
  def apply(addr: Int, reg: UInt,
    wmask: UInt = WritableMask, wfn: UInt => UInt = (x => x),
    rmask: UInt = WritableMask, rfn: UInt => UInt = x=>x
  ) = (addr, (reg, wmask, wfn, rmask, rfn))
  def generate(mapping: Map[Int, (UInt, UInt, UInt => UInt, UInt, UInt => UInt)], raddr: UInt, rdata: UInt,
    waddr: UInt, wen: Bool, wdata: UInt):Unit = {
    val chiselMapping = mapping.map { case (a, (r, wm, w, rm, rfn)) => (a.U, r, wm, w, rm, rfn) }
    rdata := LookupTree(raddr, chiselMapping.map { case (a, r, wm, w, rm, rfn) => (a, rfn(r & rm)) })
    chiselMapping.map { case (a, r, wm, w, rm, rfn) =>
      if (w != null && wm != UnwritableMask) when (wen && waddr === a) { r := w(MaskData(r, wdata, wm)) }
    }
  }
  def isIllegalAddr(mapping: Map[Int, (UInt, UInt, UInt => UInt, UInt, UInt => UInt)], addr: UInt):Bool = {
    val illegalAddr = Wire(Bool())
    illegalAddr := LookupTreeDefault(addr, true.B, mapping.map { case (a, _) => (a.U, false.B) })
    illegalAddr
  }
  def generate(mapping: Map[Int, (UInt, UInt, UInt => UInt, UInt, UInt => UInt)], addr: UInt, rdata: UInt,
    wen: Bool, wdata: UInt):Unit = generate(mapping, addr, rdata, addr, wen, wdata)
}
```

MaskedRegMap implementation using 27 lines

Reuse Code in AXI4 Device

- We designed multiple fake devices for simulation
 - MMIO fashion
 - AXI4 or AXI4Lite bus
- The logic that handles bus read and write requests can be reused
 - But how?



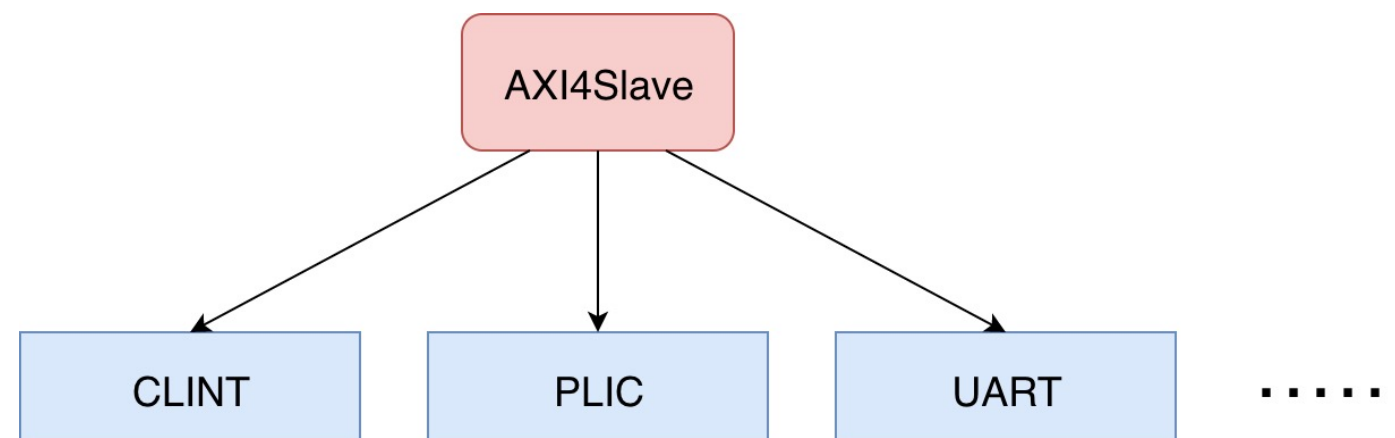
Reuse Code in AXI4 Device

- Step 1: Construct abstraction class “AXI4SlaveModule”
 - IO and AXI4 handler are implemented inside

```
val r_busy = BoolStopWatch(in.ar.fire(), in.r.fire() && rLast, startHighPriority)
in.ar.ready := in.r.ready || !r_busy
in.r.bits.resp := AXI4Parameters.RESP_OKAY
ren := RegNext(in.ar.fire(), init=false.B) || (in.r.fire() && !rLast)
in.r.valid := BoolStopWatch(ren && (in.ar.fire() || r_busy), in.r.fire(), startHighPriority)

val w_busy = BoolStopWatch(in.aw.fire(), in.b.fire(), startHighPriority)
in.aw.ready := !w_busy
in.w.ready := in.aw.valid || (w_busy)
in.b.bits.resp := AXI4Parameters.RESP_OKAY
in.b.valid := BoolStopWatch(in.w.fire() && wLast, in.b.fire(), startHighPriority)
```

- All fake devices are child class of this abstract class



Reuse Code in AXI4 Device

- Step 2: Construct AXI4 bus abstraction

- Pass bus type as an argument

```
abstract class AXI4SlaveModule[T <: AXI4Lite](_type :T = new AXI4) extends Module {  
  val io = IO(new Bundle{  
    val in = Flipped(_type)  
  })  
}
```

- Use pattern matching to implement differentiated functions

Hints for Chisel beginners

Distinguish Scala and Chisel

- [marco] vs [RTL]
- Example
 - MaskedRegMap

```
chiselMapping.map { case (a, r, wm, w, rm) =>
  if (w != null && wm != UnwritableMask)
    when (wen && waddr === a) {
      r := w(MaskData(r, wdata, wm))
    }
}
```

Hardware

Patterned Writing in Chisel

- Read more! Patterned writing exists
- Finite State Machine
- Pipeline connection
- Employ useful components
 - Queue
 - Arbiter
 - BitPat
 - BoringUtils

```
val s_idle :: s_req :: s_wait :: s_end :: Nil = Enum(4)
val state = RegInit(s_idle)
switch (state) {
  is (s_idle) {
    .....
  }
  is (s_req) {
    .....
  }
  .....
}
```

You, seconds ago • Uncommitted changes

```
when (rightOutFire) { valid := false.B }
when (left.valid && right.ready) { valid := true.B }
when (isFlush) { valid := false.B }

left.ready := right.ready
right.bits := RegEnable(left.bits, left.valid && right.ready)
right.valid := valid //&& !isFlush
```


Care about Abstraction Level

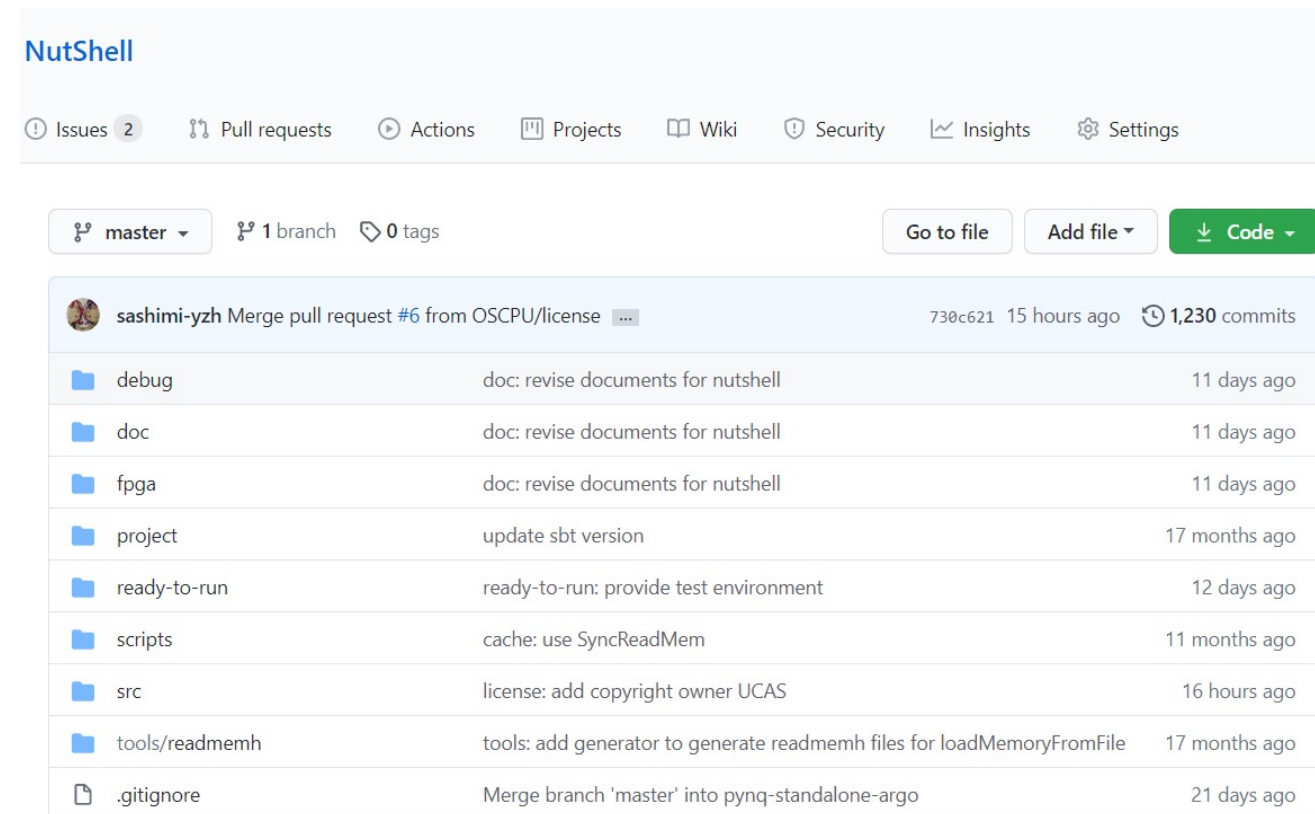
- Verilog-like Chisel
 - Avoid line-by-line translation
- Software-like Chisel
 - Be careful about "var"
- Find a balance!
 - Keep in mind that we are describing RTL
 - On this basis, explore flexibility of high-level language

Chisel doc for beginners

- [Chisel Bootcamp]
 - (<https://github.com/freechipsproject/chisel-bootcamp>)
- [Chisel Users Guide]
 - (<https://github.com/freechipsproject/chisel3/wiki/Short-Users-Guide-to-Chisel>)
- [Chisel Cheat-Sheet]
 - (<https://chisel.eecs.berkeley.edu/doc/chisel-cheatsheet3.pdf>)
- [Chisel API]
 - (<https://chisel.eecs.berkeley.edu/api/latest/index.html>)

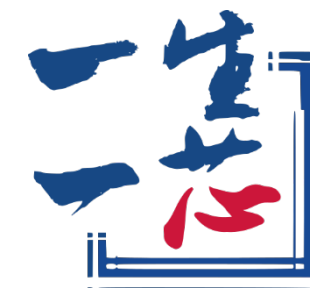
About NutShell

- NutShell has been open sourced on GitHub



The screenshot shows the GitHub repository page for NutShell. At the top, there are navigation links for Issues (2), Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below this, there are buttons for 'Go to file', 'Add file', and 'Code'. The main content area displays a list of commits, with the most recent one being a merge pull request #6 from OSCP/UCAS. The commit history table is as follows:

Commit	Message	Time
730c621	15 hours ago	1,230 commits
debug	doc: revise documents for nutshell	11 days ago
doc	doc: revise documents for nutshell	11 days ago
fpga	doc: revise documents for nutshell	11 days ago
project	update sbt version	17 months ago
ready-to-run	ready-to-run: provide test environment	12 days ago
scripts	cache: use SyncReadMem	11 months ago
src	license: add copyright owner UCAS	16 hours ago
tools/readmemh	tools: add generator to generate readmemh files for loadMemoryFromFile	17 months ago
.gitignore	Merge branch 'master' into pynq-standalone-argo	21 days ago



<https://github.com/OSCPU/NutShell>