RISC V - Architecture and Interfaces

The RocketChip

Moritz Nöltner-Augustin

Institut für Technische Informatik
Lehrstuhl für Rechnerarchitektur
Universität Heidelberg

February 6, 2017

Table of Contents

Table of Contents

What is RISCV?

Source: `https://riscv.org`

- Open-source Instruction Set Architecture (ISA)
- One ISA to cover all computer devices
- 3 ISAs (32-128 bits)
- Meant for implementation

- Modular
- Extensible
- Actively developed
- A good investment?

Why should I care?

lowRISC aims to create the processor pendant of linux

SiFive and Open-V create custom silicon

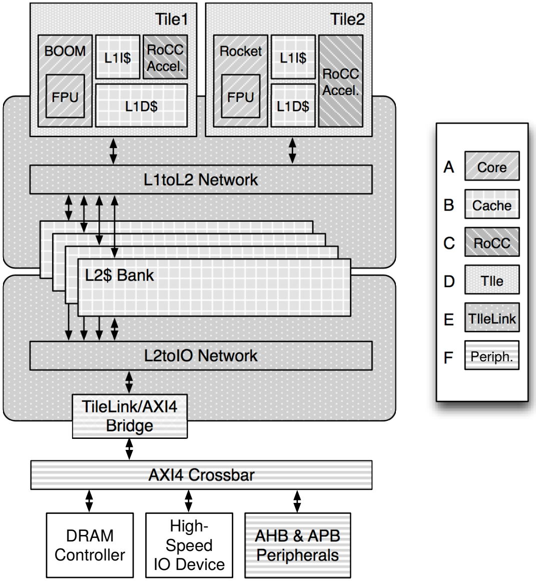
ETH Zurich and Università di Bologna create a state of the art microcontroller

- NVIDIA replaces Falcon processor
- IIT Madras creates a processor
- UC Berkeley uses it for reasearch

Sources: [1]

Table of Contents

What is RocketChip?



Source: [2]

How can I "use" RocketChip?

```
git clone https://github.com/ucb-bar/rocket-chip.git
cd rocket-chip
#git checkout boom
git submodule update --init --recursive
# Now the repo is about 3GB
cd riscv-tools
export RISCV=/path/to/toolchain/install
export PATH="${PATH}:$RISCV/bin"
./build.sh # Takes about 45 min
# The toolchain is another 800MB
cd ../emulator
make run CONFIG=BOOMConfig
make run CONFIG=ExampleSmallConfig
cd ../vsim
make -jN CONFIG=ExampleSmallConfig # about 20 min
```

Source: Collected from [3] and [4]

What can be configured?

A selection from
$(ROCKET-CHIP)/src/main/scala/coreplex/Configs.scala:

Tile (RC || Boom):

- Instruction width
- IFetch width
- Use debug
- #Breakpoints
- #Perf. Counters
- FPU key
- Mult. div. key
- Use atomics

Caches:

- #Sets
- #Ways
- #TLB-Entries
- #row bits
- #id bits
- split metadata
- ECC code
- Replacement policy

Uncore:

- TileLink config
- L1toL2 config
- Broadcast config
- Banked L2 config
- Bootrom
- #Tiles
- $block #bytes
- Build Core?

How to configure RocketChip? /1

$(RC)/src/main/scala/coreplex/Configs.scala:

```
case CacheBlockBytes => 64
case CacheName("L1D") => CacheConfig(
  nSets         = 64,
  nWays         = 4,
  rowBits       = site(L1toL2Config).beatBytes*8,
  nTLBEntries   = 8,
  cacheIdBits   = 0,
  splitMetadata = false)
...
class WithL1ICacheSets(sets:Int) extends Config((site,here,up)=>{
 case CacheName("L1I")=>up(CacheName("L1I"), site).copy(nSets=sets)
}) // Likewise for nWays
class WithCacheBlockBytes(linesize:Int)
                extends Config((site,here,up)=>{
 case CacheBlockBytes => linesize
})
```

How to configure RocketChip? /2

$(ROCKET-CHIP)/src/main/scala/rocketchip/Configs.scala

```
class DualCoreConfig2way extends Config(
  new WithNCores(2) ++ new WithL1ICacheWays(2)
  ++ new WithL1ICacheSets(32) ++ new WithCacheBlockBytes(64)
  ++ new WithL2Cache ++ new BaseConfig)

class DualCoreConfig4way extends Config(
  new WithNCores(2) ++ new WithL1ICacheWays(4)
  ++ new WithL1ICacheSets(64) ++ new WithCacheBlockBytes(32)
  ++ new WithL2Cache ++ new BaseConfig)
```

How to configure RocketChip? /2

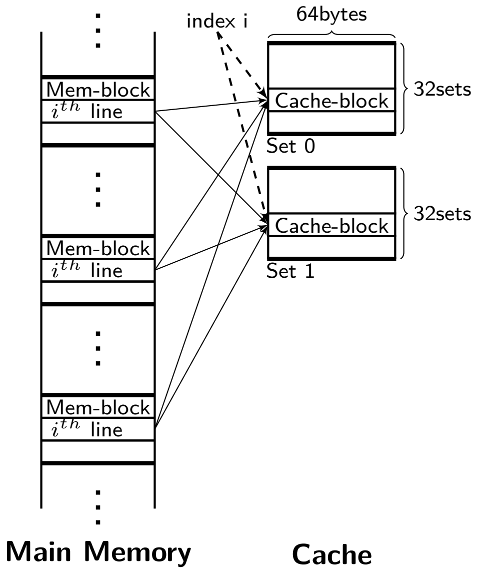
$(ROCKET-CHIP)/src/main/scala/rocketchip/Configs.scala

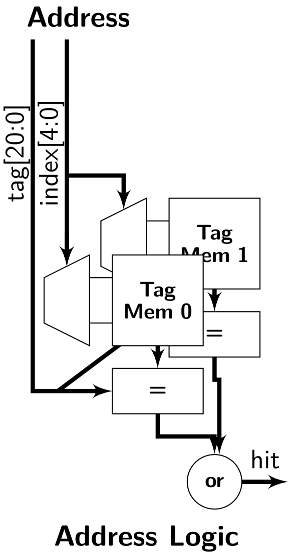
```
class DualCoreConfig2way extends Config(
  new WithNCores(2) ++ new WithL1ICacheWays(2)
  ++ new WithL1ICacheSets(32) ++ new WithCacheBlockBytes(64)
  ++ new WithL2Cache ++ new BaseConfig)

class DualCoreConfig4way extends Config(
  new WithNCores(2) ++ new WithL1ICacheWays(4)
  ++ new WithL1ICacheSets(64) ++ new WithCacheBlockBytes(32)
  ++ new WithL2Cache ++ new BaseConfig)
```

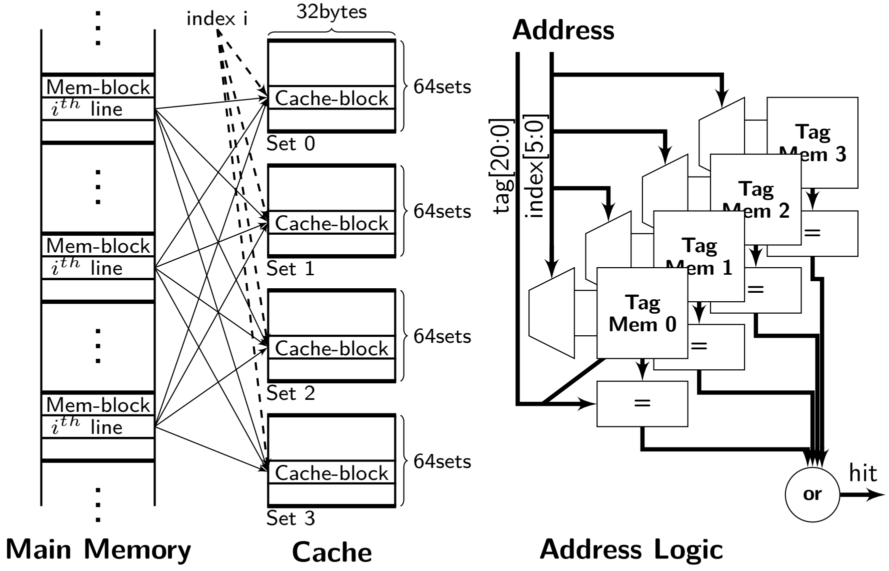
```
...
cd rocket-chip/vsim
make -j5 CONFIG=DualCoreConfig2way
...
make -j5 CONFIG=DualCoreConfig4way
```

Cache Example, 2-Way





Cache Example, 4-Way



Results: Module ICache_icache (2 ways, 32 sets, 64 bytes)

```
module tag_array( // x1
  input   [4:0] RW0_addr,
  input    RW0_en,
  input    RW0_clk,
  input    RW0_wmode,
  input   [20:0] RW0_wdata_0,
  input   [20:0] RW0_wdata_1,
  output  [20:0] RW0_rdata_0,
  output  [20:0] RW0_rdata_1,
  input    RW0_wmask_0,
  input    RW0_wmask_1
);
reg [41:0] ram [31:0];
```

```
module _T_772( // x2
  input   [7:0] RW0_addr,
  input    RW0_en,
  input    RW0_clk,
  input    RW0_wmode,
  input   [63:0] RW0_wdata,
  output  [63:0] RW0_rdata
);
reg [63:0] ram [255:0];
```

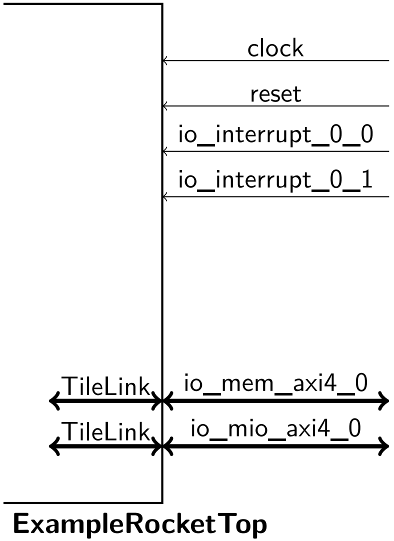
Results: Module ICache_icache (4 ways, 64 sets, 32 bytes)

```
module tag_array( // x1
  input   [5:0] RW0_addr,
  input    RW0_en,
  input    RW0_clk,
  input    RW0_wmode,
  input   [20:0] RW0_wdata_0,
  input   [20:0] RW0_wdata_1,
  input   [20:0] RW0_wdata_2,
  input   [20:0] RW0_wdata_3,
  output [20:0] RW0_rdata_0,
  output [20:0] RW0_rdata_1,
  output [20:0] RW0_rdata_2,
  output [20:0] RW0_rdata_3,
  input    RW0_wmask_0,
  input    RW0_wmask_1,
  input    RW0_wmask_2,
  input    RW0_wmask_3
);
reg [83:0] ram [63:0];
```

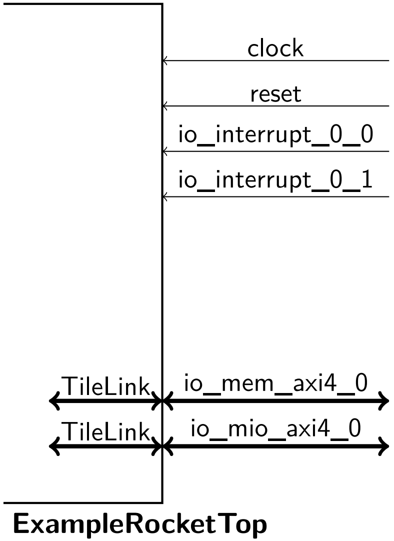
```
module _T_850( // x4
  input   [7:0] RW0_addr,
  input    RW0_en,
  input    RW0_clk,
  input    RW0_wmode,
  input   [63:0] RW0_wdata,
  output [63:0] RW0_rdata
);
reg [63:0] ram [255:0];
```

Table of Contents

RocketChip Interfaces



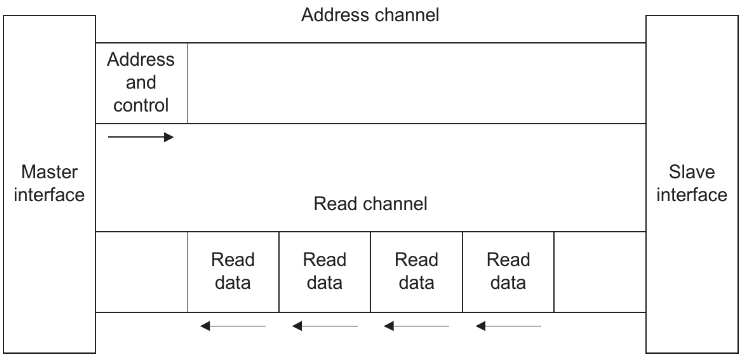
RocketChip Interfaces



- 451 signals per TileLink
- TileLink: Data + Opcode
- 282 signals per AXI interface
- io_l2_axi4_0 optional

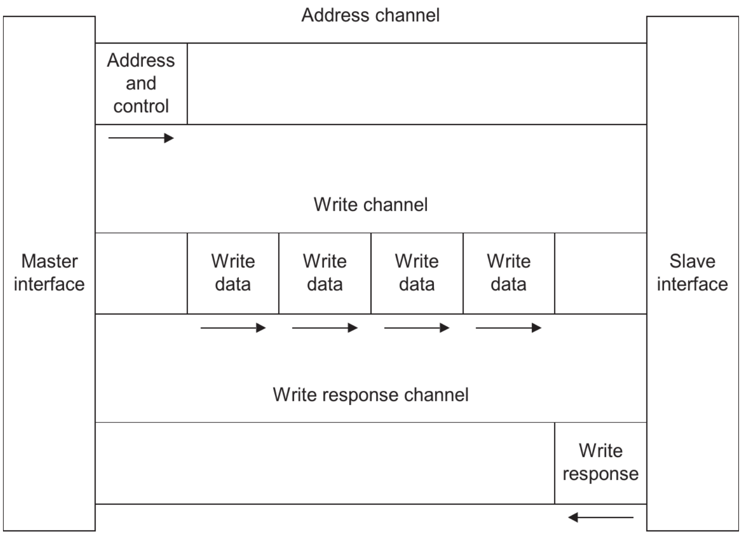
Signal Bundle	Prefix
Write Address Channel	..._aw_...
Write Data Channel	..._w_...
Write Response	..._b_...
Read Address Channel	..._ar_...
Read Data Channel	..._r_...
(Low Power Interface)	..._c_...

AXI Read



Source: Taken from [5]

AXI Write



Source: Taken from [5]

AXI Features

- Burst up to 256 transfers
- Memory attributes
- Transaction buffering
- Privileged, secure, instruction bits
- Transaction IDs
- Exclusive Access
- Atomicy size
- Quality of Service signals

TileLink: Inter-tile cache coherence network

- Agents

Client	Request/hold \$ block
Manager	Oversee permission and data flow

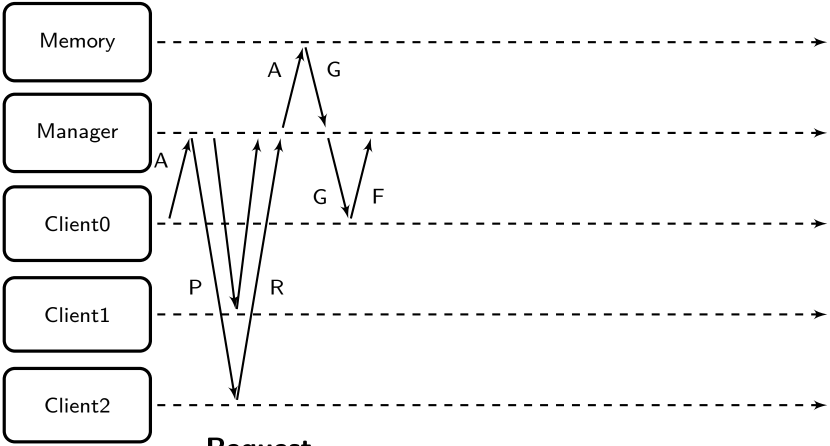
- Channels

Aquire	Start read/write uncached
Probe	Check if client has \$-block/revoke permissions
Release	Free data/write back
Grant	Provide data/permissions
Finish	Final acknowledgement from requestor

Refer to the specification for definitions of the exact operations

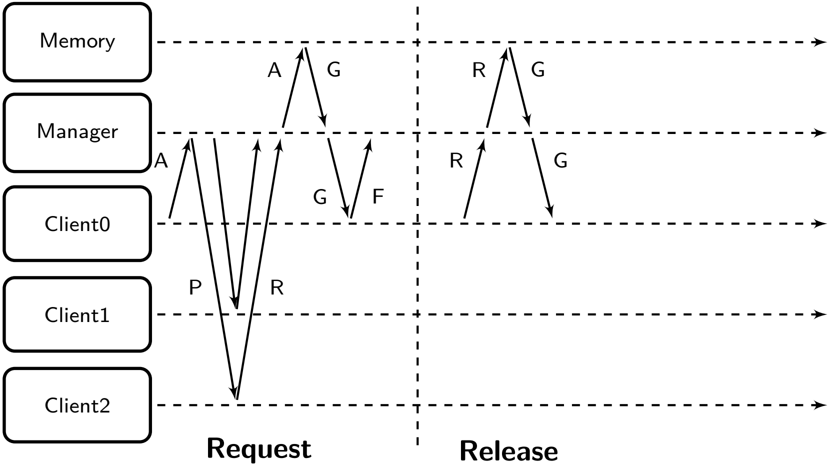
TileLink: Operations

TileLink: Operations

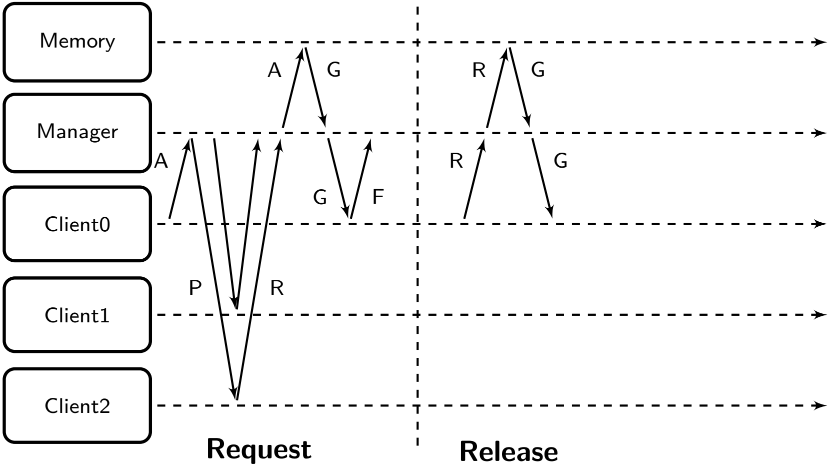


Request

TileLink: Operations

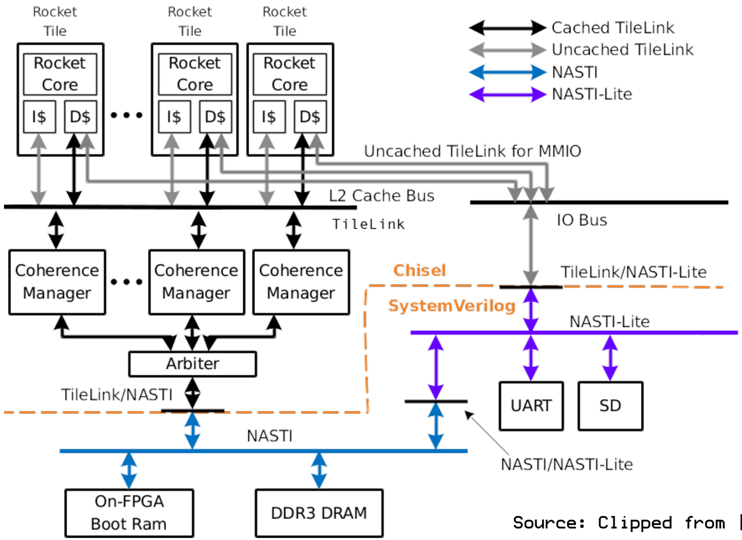


TileLink: Operations



Manager	Don’t accept transactions for in-flight blocks
Client	Don’t release block with outstanding voluntary write-back

RocketChip Interfaces



Source: Clipped from [6]

Table of Contents

Conclusion

- RISC-V is here to stay
- RocketChip generates SoC
- Can use Rocket or Boom
- Lots of configuration options

- Chisel is hard to read
- Generated Verilog is worse
- RocketChip exposes AXI
- Many standard AXI IP

References

Some users of the RISC-V ISA
lowRISC: http://www.lowrisc.org/; SiFive: https://www.sifive.com/
Open-V: https://www.crowdsupply.com/onchip/open-v Pulp: http://www.pulp-platform.org;
IID Madras: http://rise.cse.iitm.ac.in/shakti.html
Nvidia: https://riscv.org/wp-content/uploads/2016/07/Tue1100_Nvidia_RISCV_Story_V2.pdf
UC Berkeley: https://www.youtube.com/watch?v=WJndUQssFBg&t=1539s

Asanović, Krste/Avizienis, Rimas/Bachrach, Jonathan et at. (2016)
The Rocket Chip Generator.
Technical Report No. UCB/EECS-2016-17, EECS Department, University of California, Berkeley.
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html

The RocketChip github pages
https://github.com/ucb-bar/rocket-chip
https://github.com/ucb-bar/project-template

The Berkeley Out Of Order Machine github page
https://github.com/ucb-bar/riscv-boom

AXI Reference Guide, Xilinx 2011
http://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf

Xuan Guo, Nathanael Davison, Profir-Petru Partachi, Alistair Fisher (2016)
Technical report from the lowRISC Summer Internship 2016
http://www.lowrisc.org/docs/internship-2016/report

AMBA® AXI™ and ACE™ Protocol Specification, ARM Limited 2011
http://infocenter.arm.com/help/topic/com.arm.doc.ihi0022d/index.html
http://www.gstitt.ece.ufl.edu/courses/fall15/eel4720_5721/labs/refs/AXI4_specification.pdf

The TileLink Specification, Version 0.3.3
https://docs.google.com/document/d/1Iczcjigc-LUi8QmDPwnAu1kH4Rrt6Kqi1_EUaCrfrk8/pub

End

Time for your questions