RISC-V “Rocket Chip” SoC Generator in Chisel

Yunsup Lee
UC Berkeley
yunsup@eecs.berkeley.edu
What is the Rocket Chip SoC Generator?

- Parameterized SoC generator written in Chisel
- Generates Tiles
  - (Rocket) Core + Private Caches
- Generates Uncore (Outer Memory System)
  - Coherence Agent
  - Shared Caches
  - DMA Engines
  - Memory Controllers
- Glues all the pieces together
“Rocket Chip” SoC Generator

- Generates n Tiles
  - (Rocket) Core
  - RoCC Accelerator
  - L1 I$
  - L1 D$

- Generates HTIF
  - Host DMA Engine

- Generates Uncore
  - L1 Crossbar
  - Coherence Manager
  - Exports MemIO Interface
Why SoC Generators?

- Helps tune the design under different performance, power, area constraints, and diverse technology nodes

- Parameters include:
  - number of cores
  - instantiation of floating-point units, vector units
  - cache sizes, associativity, number of TLB entries, cache-coherence protocol
  - number of floating-point pipeline stages
  - width of off-chip I/O, and more
Why Chisel?

- RTL generator written in Chisel
  - HDL embedded in Scala
- Full power of Scala for writing generators
  - object-oriented programming
  - functional programming
Rocket Scalar Core

- 64-bit 5-stage single-issue in-order pipeline
- Design minimizes impact of long clock-to-output delays of compiler-generated RAMs
- 64-entry BTB, 256-entry BHT, 2-entry RAS
- MMU supports page-based virtual memory
- IEEE 754-2008-compliant FPU
  - Supports SP, DP FMA with hw support for subnormals
## ARM Cortex-A5 vs. RISC-V Rocket

<table>
<thead>
<tr>
<th>Category</th>
<th>ARM Cortex-A5</th>
<th>RISC-V Rocket</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISA</td>
<td>32-bit ARM v7</td>
<td>64-bit RISC-V v2</td>
</tr>
<tr>
<td>Architecture</td>
<td>Single-Issue In-Order</td>
<td>Single-Issue In-Order 5-stage</td>
</tr>
<tr>
<td>Performance</td>
<td>1.57 DMIPS/MHz</td>
<td>1.72 DMIPS/MHz</td>
</tr>
<tr>
<td>Process</td>
<td>TSMC 40GPLUS</td>
<td>TSMC 40GPLUS</td>
</tr>
<tr>
<td>Area w/o Caches</td>
<td>0.27 mm²</td>
<td>0.14 mm²</td>
</tr>
<tr>
<td>Area with 16K Caches</td>
<td>0.53 mm²</td>
<td>0.39 mm²</td>
</tr>
<tr>
<td>Area Efficiency</td>
<td>2.96 DMIPS/MHz/mm²</td>
<td>4.41 DMIPS/MHz/mm²</td>
</tr>
<tr>
<td>Frequency</td>
<td>&gt;1GHz</td>
<td>&gt;1GHz</td>
</tr>
<tr>
<td>Dynamic Power</td>
<td>&lt;0.08 mW/MHz</td>
<td>0.034 mW/MHz</td>
</tr>
</tbody>
</table>

- PPA reporting conditions
  - 85% utilization, use Dhrystone for benchmark, frequency/power at TT 0.9V 25C, all regular VT transistors
  - 10% higher in DMIPS/MHz, 49% more area-efficient
HTIF: Host-Target Interface

- UC Berkeley specific block mainly used to emulate devices for simple test chips
  - Emulates system calls, console, block devices, frame buffer, network devices
  - No need for this block once the SoC has actual devices on the target machine
- Consider it as a “host DMA engine”
- A port for host system to read/write
  - Core CSRs (control and status registers)
  - Target Memory
Important Interfaces in the Rocket Chip

- **ROCCIO**
  - Interface between Rocket/Accelerator

- **HTIFIO**
  - Read/Write CSRs

- **TileLinkIO**
  - Coherence Fabric

- **MemIO**
  - Simple AXI-like memory interface

- **HostIO**
  - Host Interface to HTIF
TileLinkIO consists of Acquire, Probe, Release, Grant, Finish
UncachedTileLinkIO consists of Acquire, Grant, Finish
- Convertors for TileLinkIO/UncachedTileLinkIO in uncore library
MemIO

- MemReqCmd consists of addr, rw (write=true), tag
- MemData consists of 128 bit data payload
- MemResp consists of 128 bit data payload, tag
- Decoupled(interface) means an interface with ready/valid signals
Rocket sends coprocessor instruction via the Cmd interface
- Accelerator responds through Resp interface
- Accelerator sends memory requests to L1D$ via CacheIO
- busy bit for fences
- IRQ, S, exception bit used for virtualization
- UncachedTileLinkIO for instruction cache on accelerator
- PTWIO for page-table walker ports on accelerator
- reset signal and core_id routed from HTIF (historical reasons nothing technical)
- CSR Read/Write requests go through CSRReq/CSRResp
- IPI Requests go through IPIReq/IPIResp
- HTIFIO likely to be modified in the near future
Rocket Chip C++ Emulator Setup
Rocket Chip Berkeley Test Chip Setup

Rocket Chip

HostIO/AXI Convertor

MemIO/AXIHP Convertor

MemIO Deserializer

AXI Master

AXI HP Slave

ARM Processing System

RISC-V Frontend Server

MemIO Serializer

MemIO

HostIO

AXI

AXI HP

DDR3

DRAM
Rocket Chip “SoC” Setup

Rocket Chip

LPDDR3 Memory Controller

Interrupts
TiileLinkIO
Uncached TiileLinkIO

MemI0

Devices

DRAM

MemI0
LPDDR3
Who should use the Rocket Chip Generator?
People who would like to develop ...

- A RISC-V SoC
  - Look into Chisel parameters
- New Accelerators
  - Drop in at ROCCIO level
- Own RISC-V Core
  - Drop in at TileLinkIO level or MemIO level
- Own Device
  - Drop in at TileLinkIO or UncachedTileLinkIO
New Features: L2$ with Directory Bits

- Shared L2$ with multiple banks
- Each L2$ will act as a coherence manager with directory bits (snoop filter)
- These caches can be composed to build outer-level caches such as an L3$
New Features: ROCC interfaces with L2$

- ROCC talks directly to the L2$ to address more data
New Features on the Deck

- Dual-issue Rocket Core
- Hwacha Vector Unit (checkout hwacha.org)
- Dump MemIO and use AXI