Untethered lowRISC,
Memory Mapped IO and TileLink/AXI

Wei Song
27/07/2015
Time Line

Nov. 2014
Rocket-Chip release from Berkeley

Apr. 2015
First lowRISC release.
Initial tagged memory support.

Now
Memeory Mapped IO.

Oct. 2015
Untethered lowRISC release.

- Untethered SoC.
- Support Kintex KC705.
- Support MMIO.
- Support SD, UART, DDRAM.
- Open simulation environment.

- Added tags in L1 D$, L2.
- Added a tag cache.
- Added 2 instructions to load/store tag.
- A tutorial about Rocket-chip.
Rocket-Chip Release (Berkeley)
lowRISC Release (tagged memory)

Tag in L1 D$, L2 $
Tag Cache
LTAG/STAG instructions
Latest Rocket-Chip (Berkeley)

- Rocket Tile
  - Rocket Core
  - L2 & Coherence Manager
  - Arbiter
  - AXI Bus
  - Memory Controller

- Host Interface
- ARM
- UART
- SD
- EtherNet

- L2 Bus
- Cached TileLink
- Uncached TileLink
- AXI
- MemIO

- Multi-beat TileLink
- Standardize TileLink transactions
- Possible coherence support of L3
- Code refactoring
- AXI/AXI interface (NASTI)
Untethered lowRISC SoC (First Version)
Current Status

Rocket Tile
Rocket Core
L2 & Coherence Manager

Rocket Tile
Rocket Core
L2 & Coherence Manager

Rocket Tile
Rocket Core
L2 & Coherence Manager

L2 Cache Bus

DMA
coherent
Boot Minion

DMA
incoherent

Cached TileLink
Uncached TileLink
AXI
AXI-Lite

AXI Bus
TileLink/AXI

On-FPGA Boot Ram
Memory Controller

Tag Cache
Arbiter

UART
SD
EtherNet
Memory Mapped IO

• Target
  – IO load/write (B/HW/W/DW)
  – In-order uncached load/store
  – Side effect
    • None for all write in units of byte
    • None for all read in units of word (32-bit AXI-Lite)
  – No change in current L2 coherent manager
Untethered lowRISC SoC (First Version)
L1 Data Cache

![Diagram of L1 Data Cache](image)
L1 Data Cache with IO Handler

mem.req -> mshrs
mem.grant -> mshrs
io.req -> io_addr
io.grant -> io_addr

mshrs_meta_write -> mshrs
mshrs_request -> mshrs

io_data -> s1_io_data
s1_io_data -> s1_req
s1_req -> dtlb

meta_read resp -> meta
meta_read resp -> data

amoalu_rhs_out -> data

Stage 1
Stage 2
Stage 3
Stage 4

cpu.resp -> Stage 4

write -> data
write -> meta

cpu.resp -> data
write -> data

TileLink Channels

• Manager/Client
  – Manager: Coherent manager or next level cache/device
  – Client: upper level cache

• 5 Channels
  – Acquire: [C -> M]
    • Read, uncached write (write-through, IO), permission update
  – Grant: [M -> C]
    • Ack to Acquire (with data when read)
  – Finish: [C -> M]
    • Finish a transaction
  – Probe: [M -> C]
    • Coherence probe (snoop, invalidate)
  – Release: [C -> M]
    • Write-back (replace or invalidate)
Untethered lowRISC SoC (First Version)
Use a SuperChannel to store all types of TileLink channels.
Current Status of TileLink/AXI

- TileLink/AXI (Berkeley, Rocket-chip)
  - only a whole cache line
- TileLink/AXI-Lite (lowRISC)
  - 1,2,4,8 byte write; 4,8 byte read
- AHB/APB (Berkeley, Z-Scale)

Still needed:
- AXI/AXI-Lite compatible, auto width SerDes switch
  - The AXI-Node from PULP
  - May be in Chisel for its parameterization capability
- AXI/Wishbone, TileLink/Wishbone
Remain Issues

• Interrupt controller
• Open Sourced, License compatible IPs
  – UART (Flexpret, BSD)
  – SD host controller
  – Ethernet controller (Xilinx IP for now)
  – Memory controller (difficult to get)
• Open Source EDA tools
  – Current environment:
    • VCS (DRAMSim, Front-end server, DirectC)
    • Vivado+SDK (SDK not available for Kintex)
  – Target environment:
    • Verilator (SystemVerilog 2009, SystemC, VPI, DPI)
    • Vivado only
After the Untethered SoC

- Implementing the hierarchical tag cache (hardware)
- Debug interface
- Integrating minions (PULP)
- Tag support in Rocket cores (Lucas)