Towards General Purpose Tagged Memory

Wei Song, Alex Bradbury, and Robert Mullins
Computer Laboratory, University of Cambridge

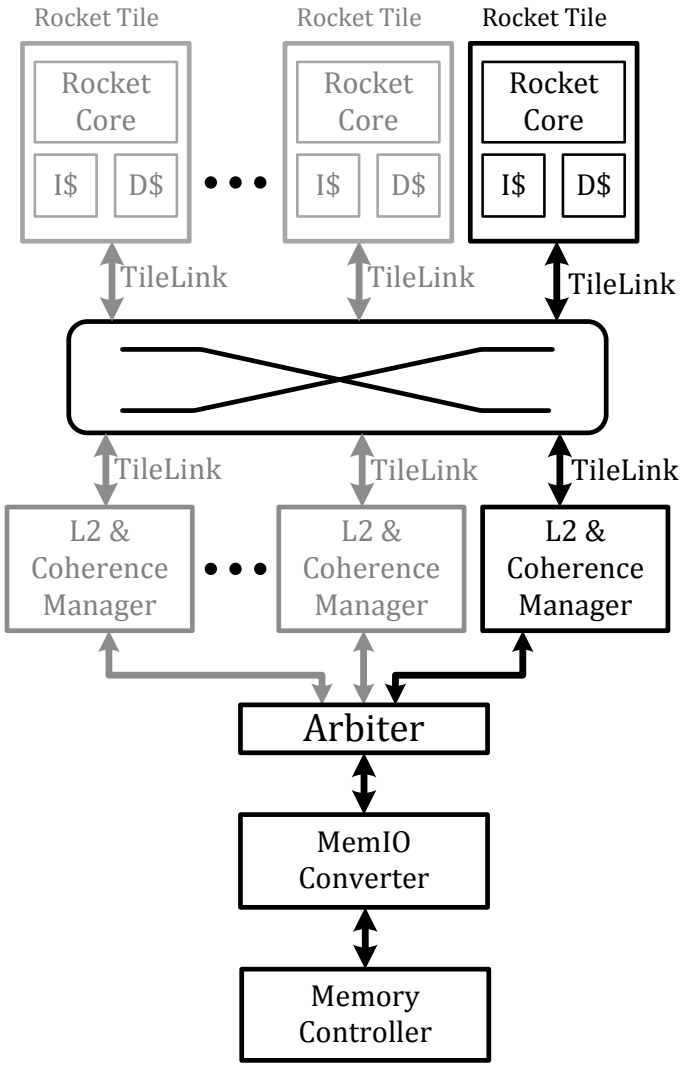
2nd RISC-V workshop, 30/06/2015

Code release and tutorial:
http://www.lowrisc.org/docs/tutorial/

lowRISC Project

- lowRISC
  - Open source SoC provider
  - 64-bit RISC-V ISA, Rocket core
  - Key features: tagged memory and “minion” cores
- Tagged memory
  - Adding tag bits for each 64-bit word.
  - [Permission] Protection against control-flow hijacking
  - [Lock] Memory synchronization
  - [Type] Garbage collection
  - [Breakpoint] Debugging

Rocket Chip



Rocket Tile:
Rocket core, private I$ and D$

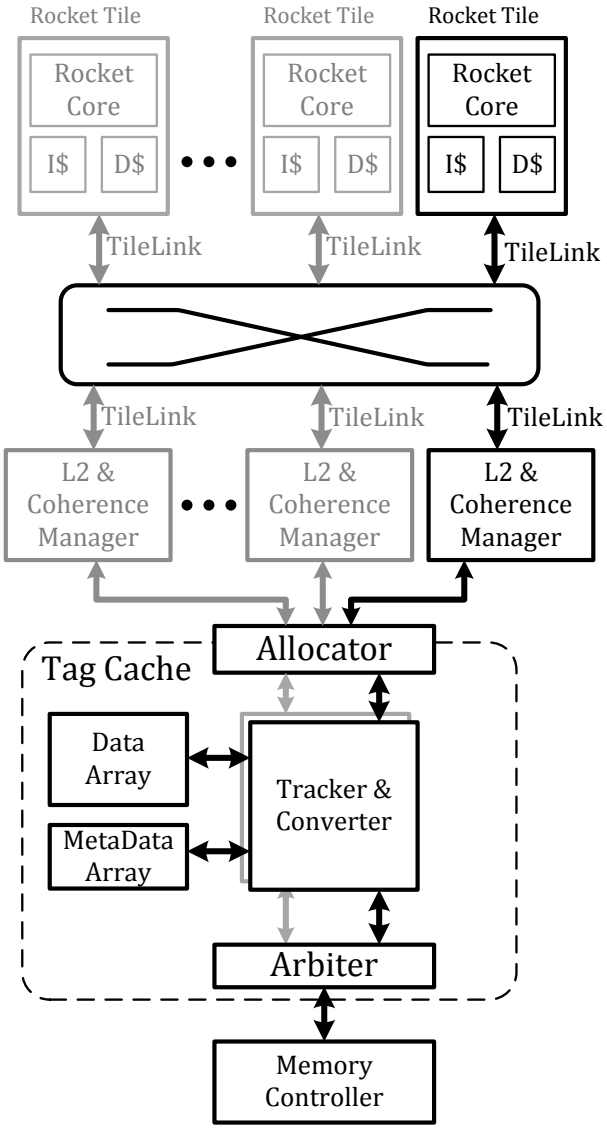
Crossbar between L1 and banked L2

Banked L2 (coherence manager)

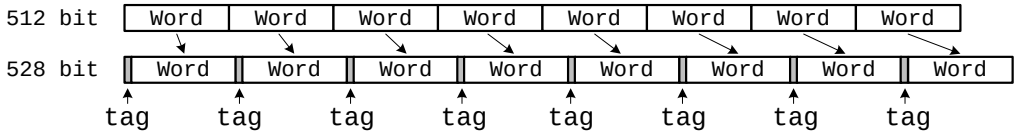
Single memory port, format converter

Memory

Support Tagged Memory (1)

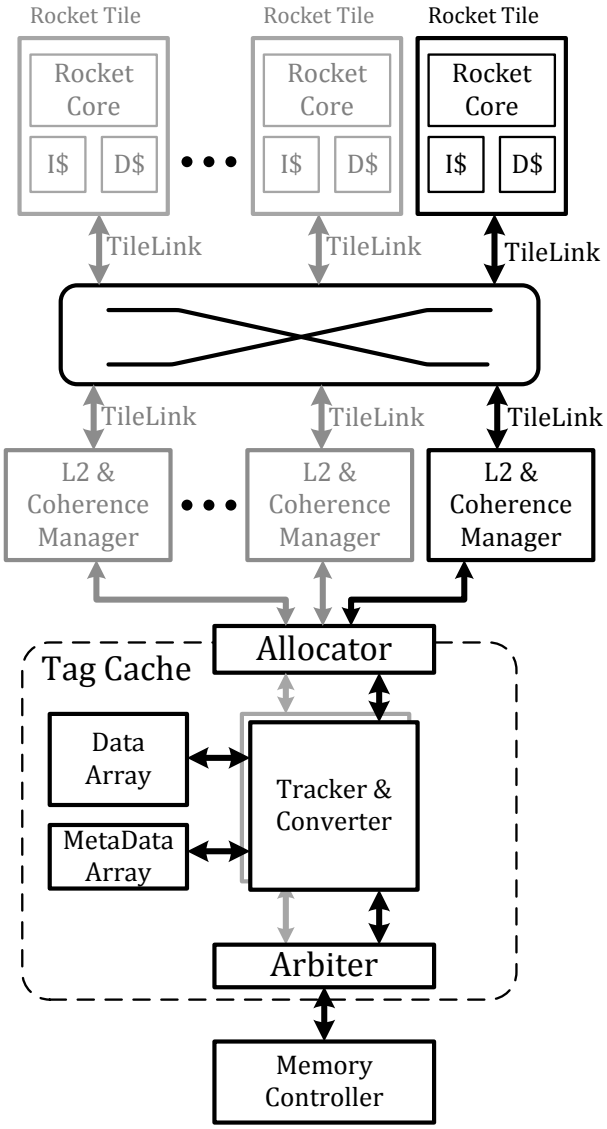


Augment each 64-bit word with tag bits

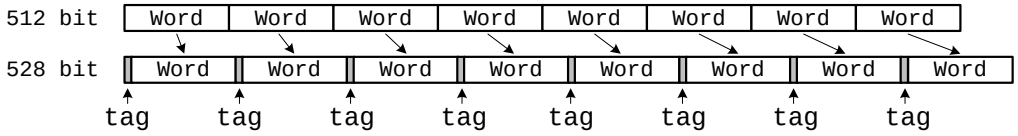


Augmented cache line is transparent to coherence control.

Support Tagged Memory (1)



Augment each 64-bit word with tag bits



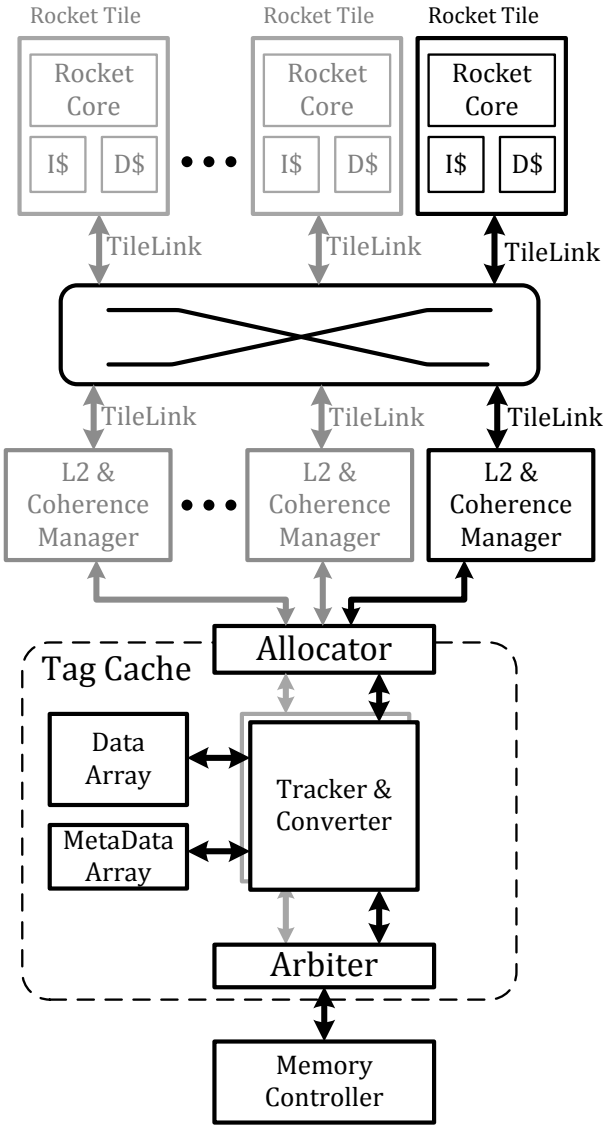
Augmented cache line is transparent to coherence control.

Memory is partitioned into data and tag regions.

Every memory access needs a data access and an extra tag access.

To reduce the number of tag access, a tag cache is added.

Support Tagged Memory (2)



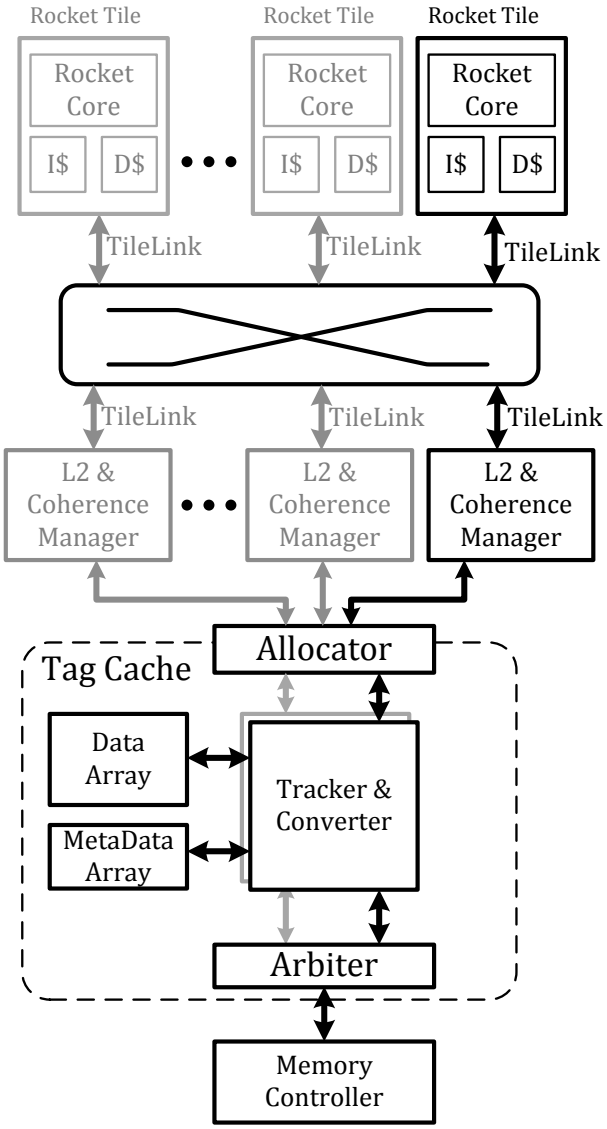
New instructions for load/store tag:

LTAG rd, imm(rs1) # load tag @ rs1 + imm to rd

STAG rs2, imm(s1) # store tag rs2 @ rs1 + imm

Adding a new memory op type M_T in D$.
No change in core pipeline.

Support Tagged Memory (2)



New instructions for load/store tag:

LTAG rd, imm(rs1) # load tag @ rs1 + imm to rd

STAG rs2, imm(s1) # store tag rs2 @ rs1 + imm

Adding a new memory op type M_T in D$.
No change in core pipeline.

Multiple trackers (transaction handlers) to serve multiple memory access in parallel.

Non-intrusive to current Rocket chip.
Easy to implement.
But not efficient.

Function and Performance Tests

- Function tests:
  - Inline assembly to use LTAG/STAG
  - Bare metal assembly test cases (riscv-tests)
  - Larger C programs (proxy kernel) to memory sweep with tags
  - Boot Linux with separated tag region
  - SystemVerilog unit test for the tag cache (random package generation)
- Performance tests:
  - SPECInt 2006 benchmark (9 out of 12)

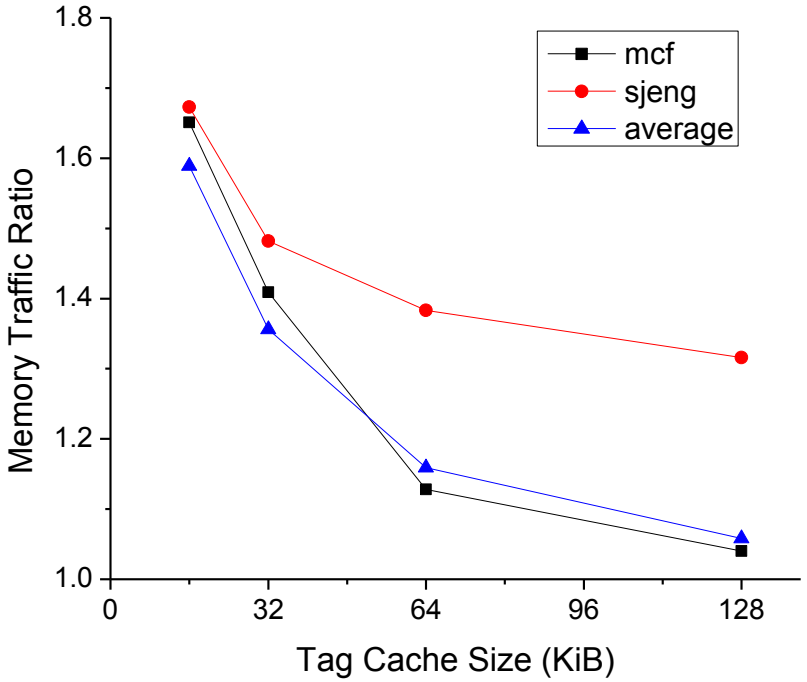
SPECInt 2006 Result (1)

	I$ 8 KiB (MPKI)	D$ 16 KiB (MPKI)	L2 256 KiB (MPKI)	Mem Traffic No Tag (TPKI)	Tag $ 16 KiB (MPKI)	Traffic Ratio	Tag $ 32 KiB (MPKI)	Traffic Ratio	Tag $ 64 KiB (MPKI)	Traffic Ratio	Tag $ 128 KiB (MPKI)	Traffic Ratio
perlbench	20	5	<1	2	<1	1.289	<1	1.089	<1	1.025	<1	1.011
bzip2	<1	14	10	16	10	1.941	7	1.688	3	1.281	<1	1.007
gcc	15	11	4	6	2	1.497	<1	1.240	<1	1.072	<1	1.023
mcf	<1	168	104	136	67	1.651	40	1.409	11	1.128	3	1.040
gobmk	24	8	3	6	1	1.368	<1	1.146	<1	1.073	<1	1.046
sjeng	11	5	1	3	1	1.673	<1	1.482	<1	1.383	<1	1.316
h264ref	1	3	2	3	<1	1.480	<1	1.265	<1	1.109	<1	1.028
omnetpp	40	5	<1	<1	<1	1.653	<1	1.415	<1	1.190	<1	1.042
astar	<1	21	5	9	4	1.750	2	1.471	<1	1.173	<1	1.009
average	12	27	14	20	10	1.589	6	1.356	2	1.159	<1	1.058

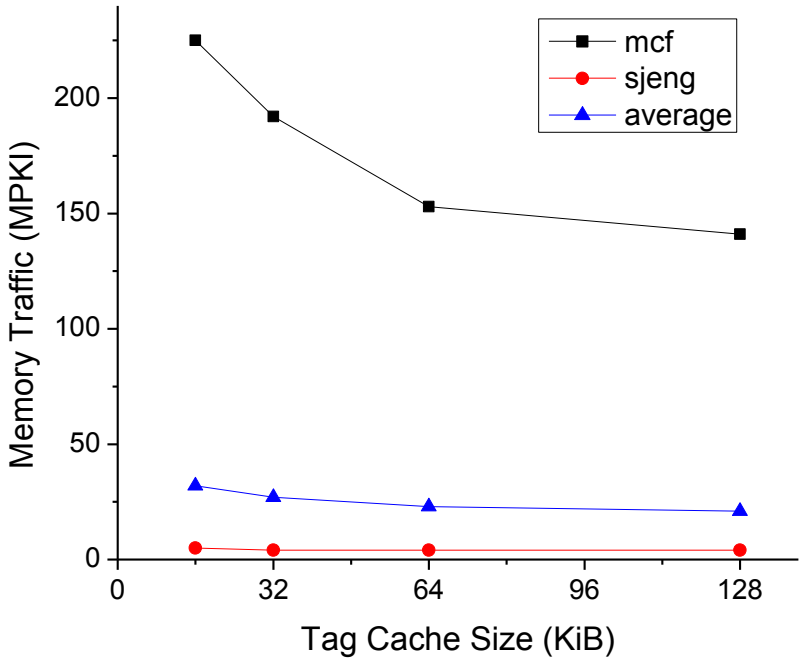
MPKI: misses per 1000 instructions

TPKI: transactions per 1000 instructions

SPECInt 2006 Result (2)



Memory traffic ratio
Tagged/no tag



Memory Traffic (MPKI)

Future Work (1)

- Optimizing the tag cache
  - Identify non-tag pages
  - Automatic check for non-tag (zero tags) cache line
  - Write back only revised tags
- Further ISA support
  - Atomic data + tag load/store
  - I$, trap instructions lack of permissions in tags
  - Set/reset tags for a whole cache line

Future Work (2)

- General tag interpretation
  - Propagate tags through core pipeline
  - General tag interpretation/check/operation
  - Rule configuration
    - Static table
    - Controlled by minion cores

Conclusion

- Add tagged memory in Rocket chip
  - Augment cache lines with tag bits
  - Add a tag cache to reduce memory traffic
  - New instructions for tag load/store
  - More optimization in the future

Fro more information and get in touch:

1. **Talk to Alex or me.**
2. **http://www.lowrisc.org/**
3. **E-Mail to lowrisc-dev@lists.lowrisc.org**