Evaluation of RISC-V RTL with FPGA-Accelerated Simulation

Donggyu Kim, Christopher Celio, David Biancolin,
Jonathan Bachrach, Krste Asanović

Berkeley Architecture Research

CARRV 2017

10/14/2017

Evaluation Methodologies For Computer Architecture Research

Microarchitectural Cycle-by-Cycle Software Simulators

GPGPU-Sim

Flexus

MARSSx86

Analytic Power/Energy Modeling

mcpat

GPUWattch Energy Model

Simulation Sampling

SMARTS

SimPoint

How To Do Computer Architecture Research In The Past?

Few lines of C/C++ code

100M~1B instructions

spec

gem5

Paper

How Easy Single-Cycle Perfect Caches?


```
541  590          if(cacheLines_->get_port(queueEntry->request)) {
     591  +               /***** by vteori *****/
     592  +               // for perfect caches
     593  +               int robid = queueEntry->request->get_robid();
     594  +               bool icache_walk = queueEntry->request->is_instruction();
     595  +               bool itlb_walk = memoryHierarchy_->is_itlb_miss();
     596  +               bool dtlb_walk = memoryHierarchy_->is_dtlb_miss(robid);
     597  +               bool perfect_l2_icache = config.perfect_l2_icache && type_ == L2_CACHE && icache_walk && !itlb_walk;
     598  +               bool perfect_l2_dcache = config.perfect_l2_dcache && type_ == L2_CACHE && !icache_walk && !dtlb_walk;
     599  +
     600  +               // for debug by vteori
     601  +               /*if (!icache_walk) ptl_logfile
     602  +                       << (type_ == L2_CACHE ? "L2 $ " : "L1 $ ")
     603  +                       << "access => rob : " << robid << " addr : "
     604  +                       << (void *) queueEntry->request->get_physical_address() << endl;*/
     605  +
542  606          CacheLine *line = cacheLines_->probe(queueEntry->request);
543  607          bool hit = (line == NULL) ? false : line->state;
     608  +               hit |= perfect_l2_icache;
     609  +               hit |= perfect_l2_dcache;
```

- ***10 lines of C++ code in MARRSx86***

➔ ***Easy to implement unrealistic, non-cycle-accurate models***

No µarch Simulation Any More!

Validation of one design instance[1]

Other designs points?

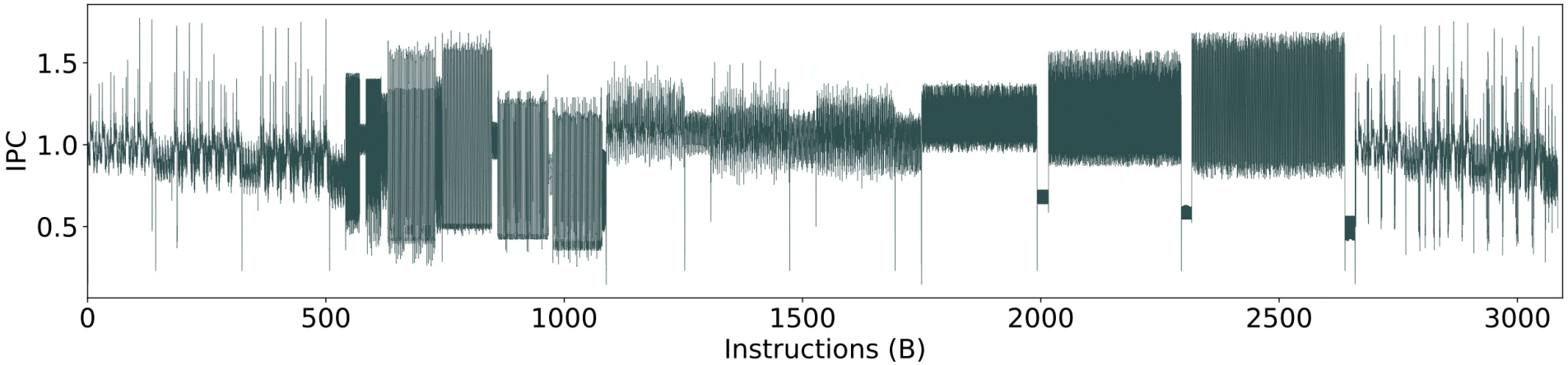
Your target modeling?

- ***Validation is difficult***
  - ***Is your modeling still cycle-accurate?***
- ***Simulation is too slow → Simulation Sampling***

[1] Gutierrez et al. Sources of error in full-system simulation, ISPASS 2014

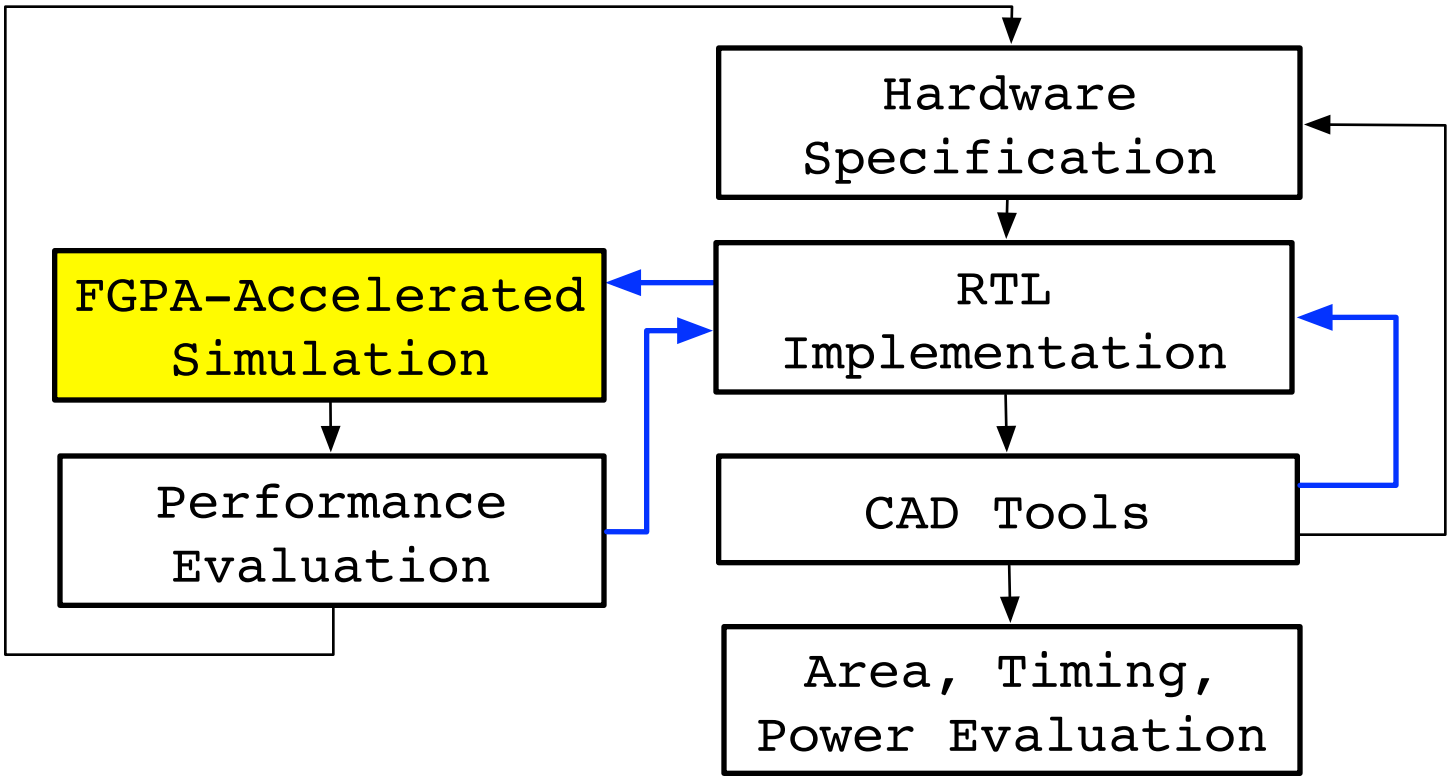
No Simulation Sampling!

- Phase-based Sampling (e.g. SimPoint)
  - No theoretically guaranteed error bounds
  - Short periods of phases show similar IPC whenever repeated
  - 401.bzip2 with its reference input in BOOM-2w



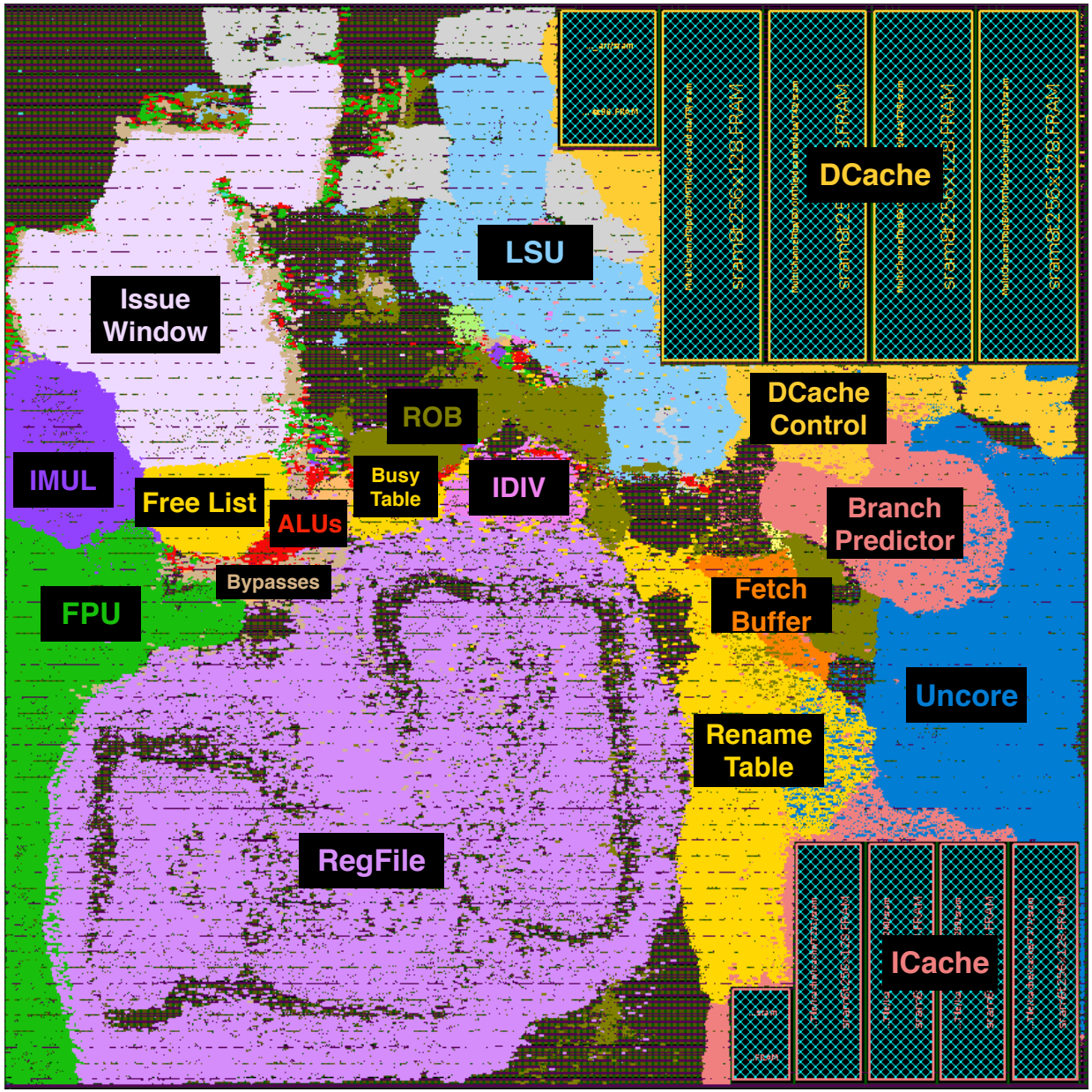
- Statistical Sampling (e.g. SMARTS)
  - Statically bounded errors
  - **State warming problems**
    - µarchitectural state should be recovered from functional simulators
- What about managed-language applications?

Build RTL To Validate Your Design Ideas!

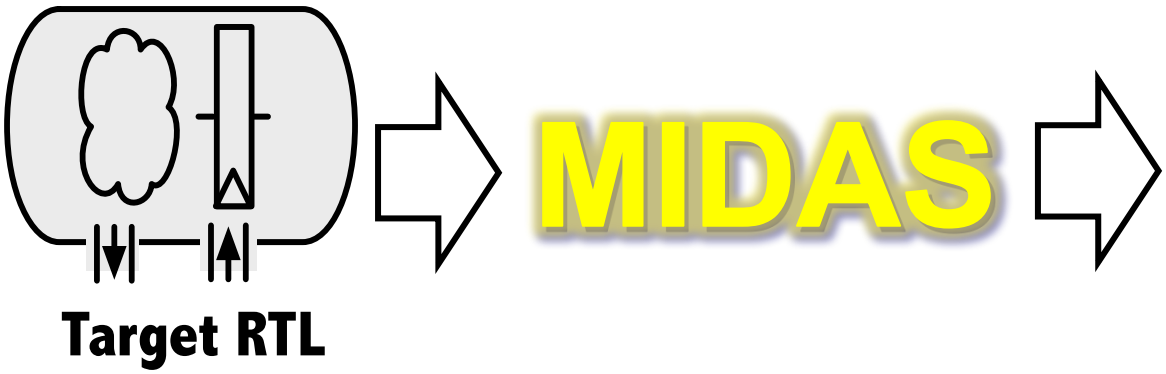


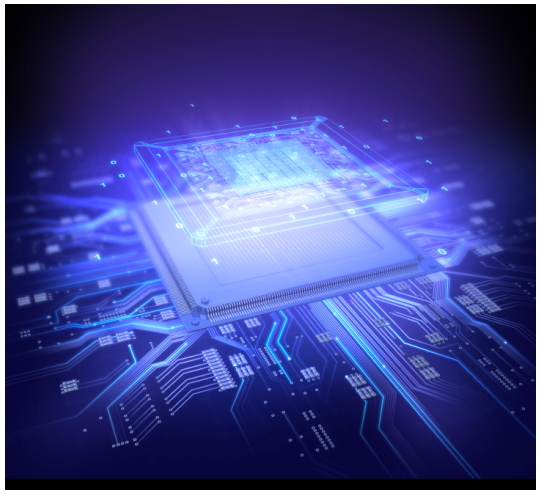
- ***RTL is no longer difficult***
  - ***Hardware construction languages (e.g. Chisel)***
  - ***RISC-V implementation code base (e.g. RocketChip)***
- ***RTL simulation is no longer slow***
  - ***Tens of MIPS using FPGAs***

Old BOOM Layout



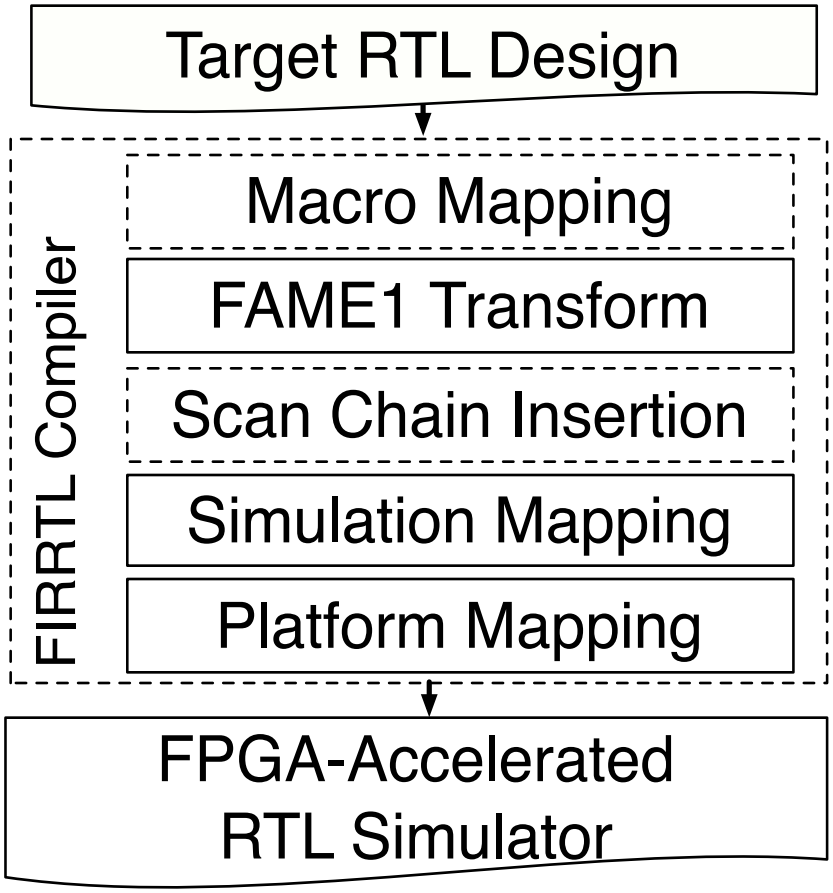
MIDAS: Turn Your RTL into Gold





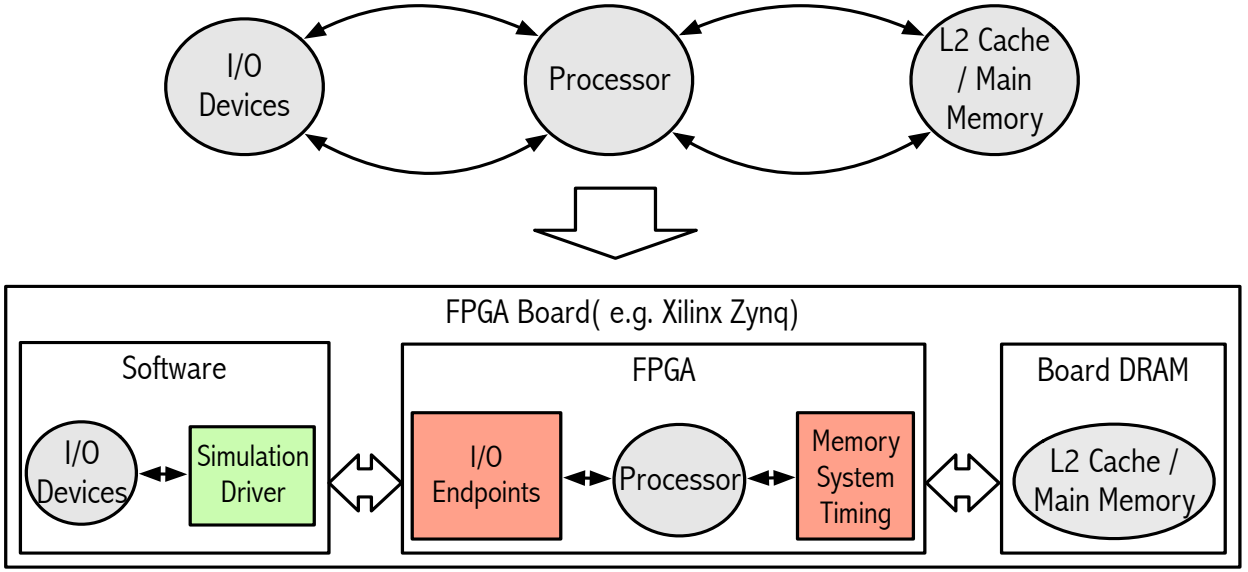
- *Automatically* transforms any RTL designs into FPGA-Accelerated RTL simulators
- *Quickly* evaluates performance, power, and energy of RTL designs with realistic software applications.
- *Flexibly* Co-simulates abstract HW & SW models

MIDAS Custom Compiler Passes



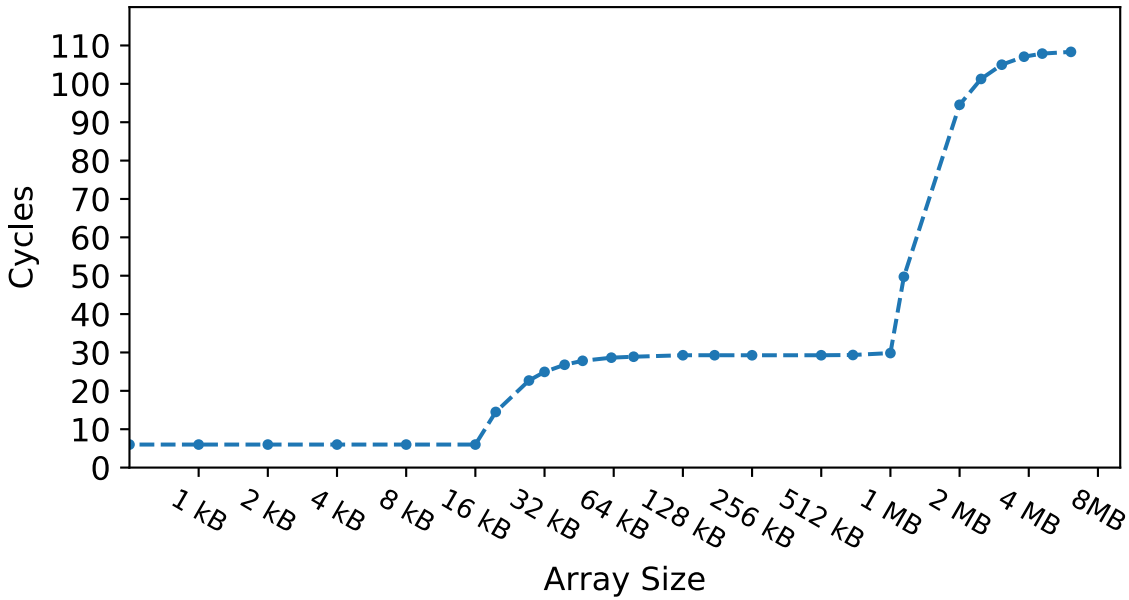
- FIRRTL: IR for RTL transforms
  https://github.com/freechipsproject/firrtl
- Instrument RTL designs for
  - Accurate performance modeling
  - Easy simulation control in FPGA
  - Interactions with abstract timing models
  - RTL state snapshots for energy modeling

Mapping Simulation to the FPGA Host



- Simulation Speed: 3.56MHz(ISCA`16) → ***~40MHz***(CARRV`17)
- I/O Endpoints
  - Low-level timing tokens <-> High-level transactions
  - Optimize the communications between SW & FPGA
- L2 / Main Memory
  - Abstract timing model(L2$ tags) in FPGA
  - Actual data in Board DRAM
  - Set size, associativity, block size, latency: runtime configurable

Memory Timing Model Validation



- A pointer-chase µbenchmark of *ccbench* running in BOOM (https://github.com/ucb-bar/ccbench)
- L1$: ***16KiB***, ***6 cycles***
- L2$: ***1MiB***, 6 + ***23 cycles*** (runtime configurable)
- DRAM: 6 + 23 + ***80 cycles*** (runtime configurable)

Target Design: Rocket Chip Generator

RISC-V	Rocket (In-order Processor)	BOOM-2w (Version 1) (Out-of-order Processor)
Fetch-width	1	2
Issue-width	1	3
Issue slots		20
ROB size		80
Ld/St entries		16/16
Physical registers	32(int)/32(fp)	110
Branch predictor		gshare: 16KiB history
BTB entries	40	40
RAS entries	2	4
MSHR entreis	2	2
L1 I$ / D$	16 KiB or 32 KiB	
ITLB / DTLB reaches	128 KiB / 128 KiB	
L2 $	1MiB / 23 cycles	
DRAM latency	80 cycles	

SPEC2006int Benchmark Suite

- Instruction count with RISC-V for the reference inputs

Benchmarks	Instruction Count(T)	Benchmarks	Instruction Count(T)
400.perlbench	2.48	458.sjeng	2.85
401.bzip2	3.08	462.libquantum	2.09
403.gcc	1.37	464.h264ref	5.07
429.mcf	0.29	471.omnetpp	0.61
445.gobmk	2.04	473.astar	1.05
456.hmmer	2.95	483.xalanbmk	1.10

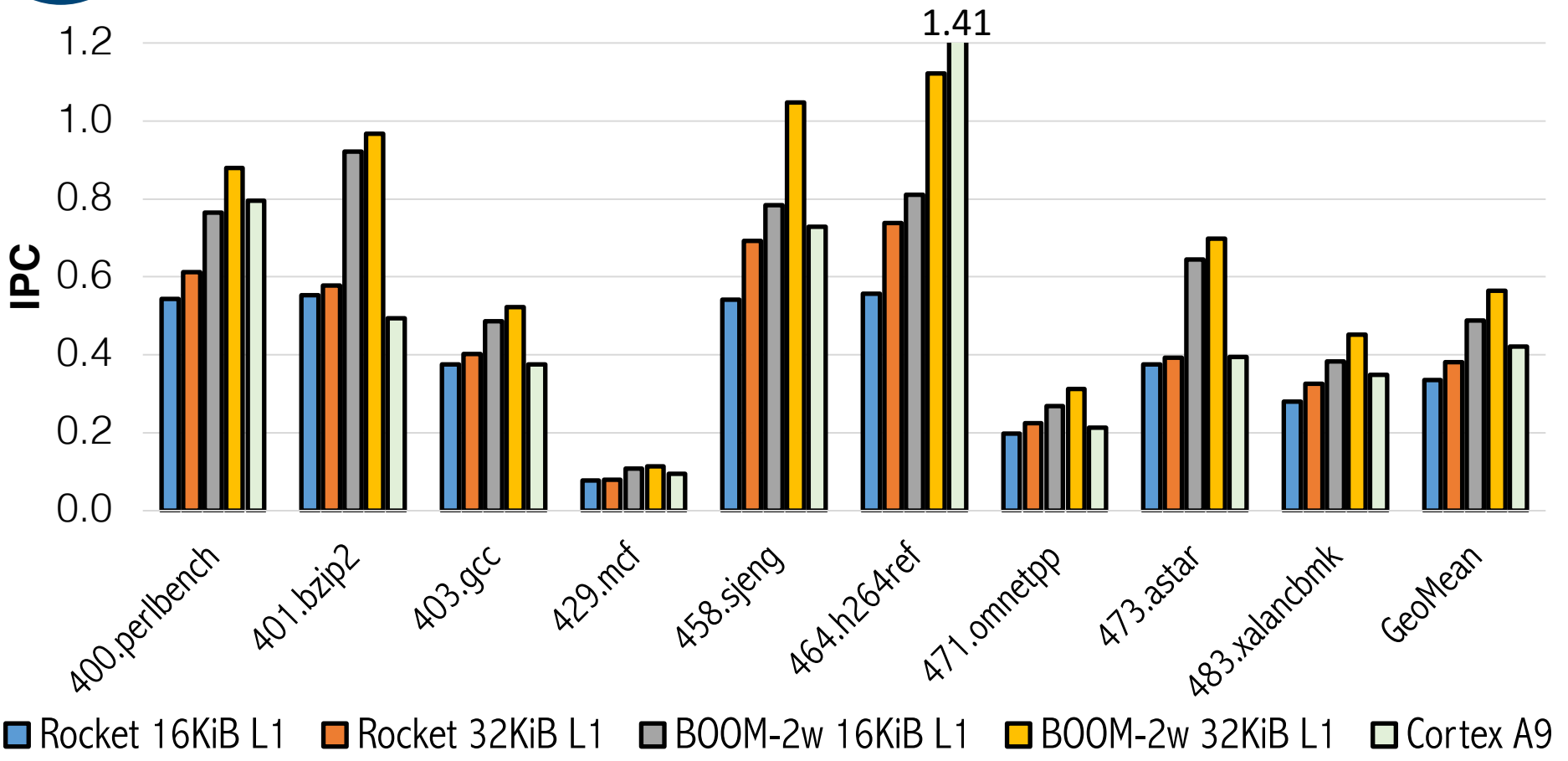
- Instruction Count Average: ***2.08 Trillion***
- 445.gobmk, 456.hmmer, 462.libquantum: fail in BOOM

Simulation Time for SPEC2006int

Simulators	Speed	Average	Max (464.h264ref: 5T)
GEM5+Ruby	100 KIPS[1]	240 days (8 months)	640 days (1.8 years)
MARSSx86	400 KIPS[2]	60 days (2 months)	160 days (5.3 months)
MIDAS	18 MIPS (40MHz)	1.4 days	3 days

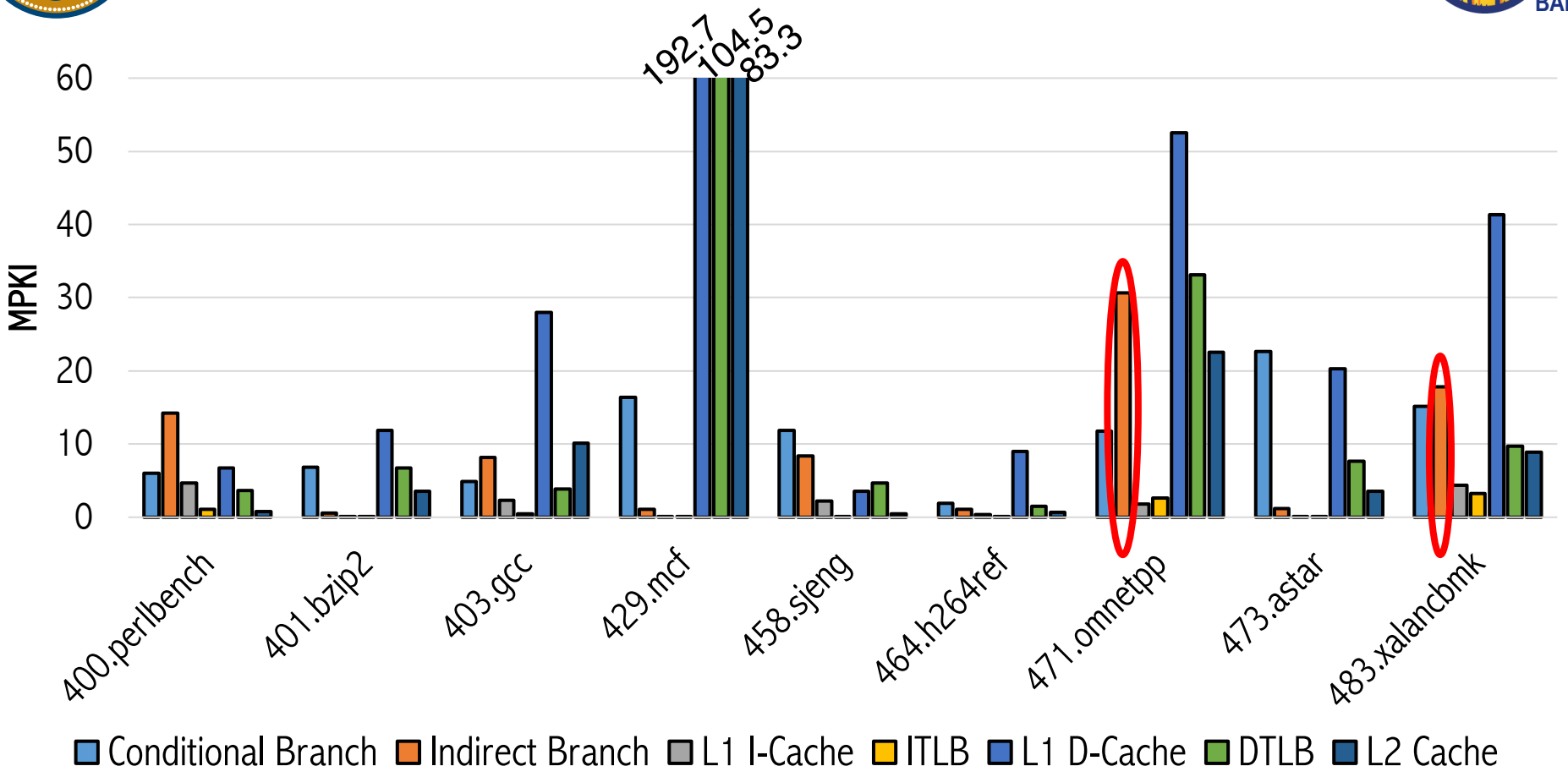
[1] http://gem5-users.gem5.narkive.com/hCJL602Q/gem5-simulation-speed
[2] Patel et al. MARSSx86: A full system simulator for x86 CPUs, DAC 2011.

Case Study: IPC



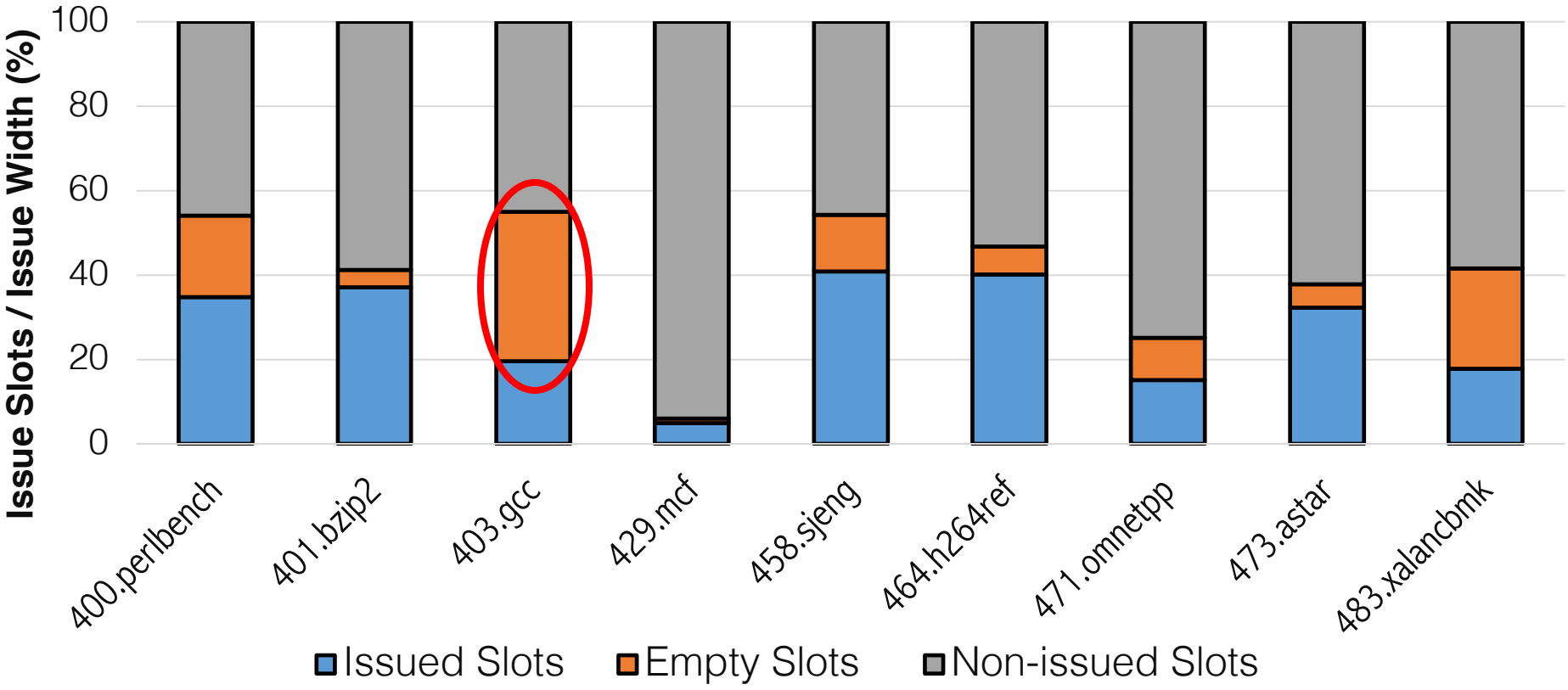
- Rocket is comparable to Cortex A9
- BOOM outperforms Cortex A9
- Performance improvement: 16KiB L1$ → 32KiB L1$

Case Study: MPKIs of BOOM-2w



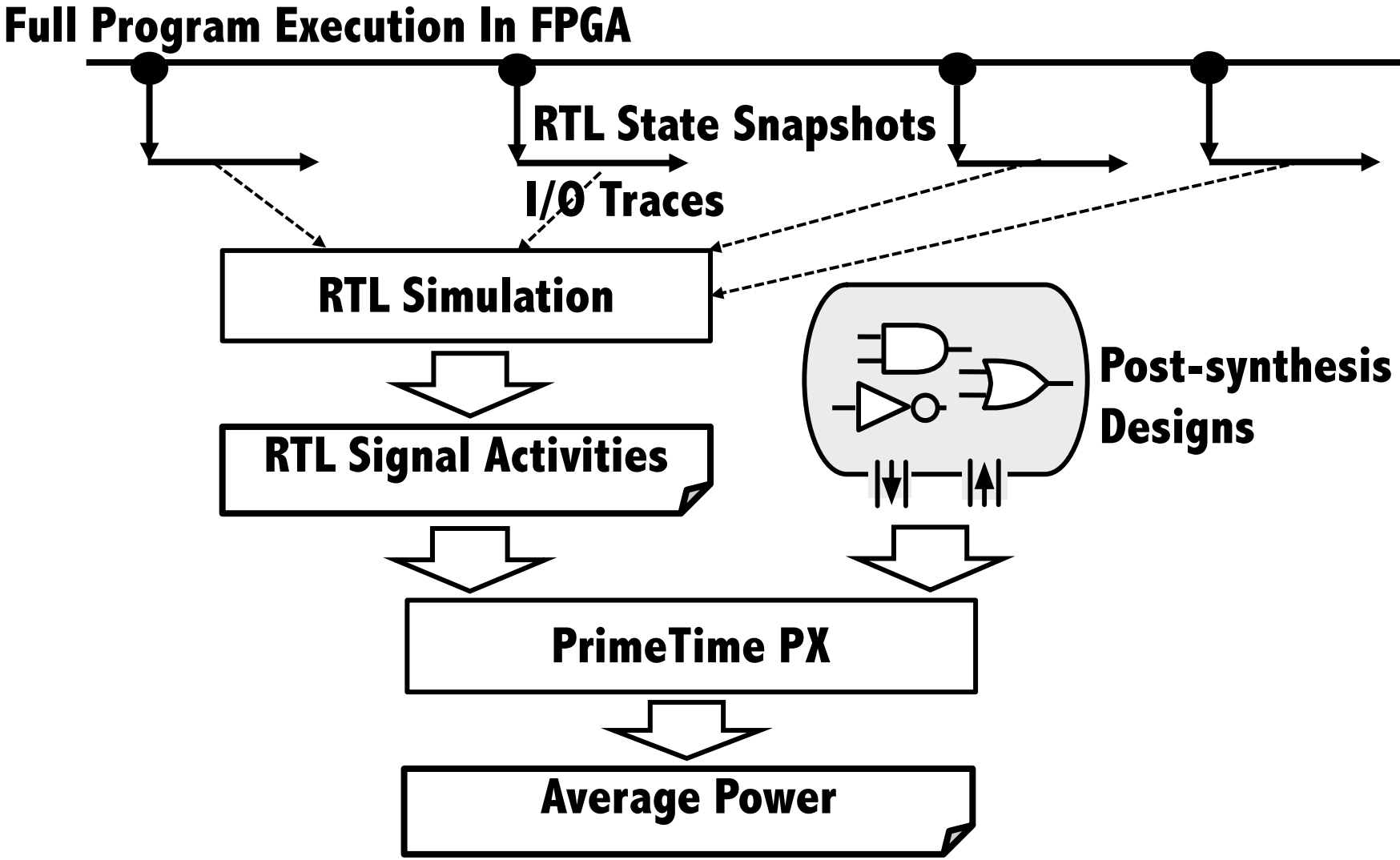
- 40 BTB entries, 32KiB L1$, 1MiB L2$, TLB reach = 128KiB
- 473.omnetpp, 483.xalancbmk:
  Large indirect branch MPKIs → Bigger BTBs

Case Study: Issue Queue Utilization



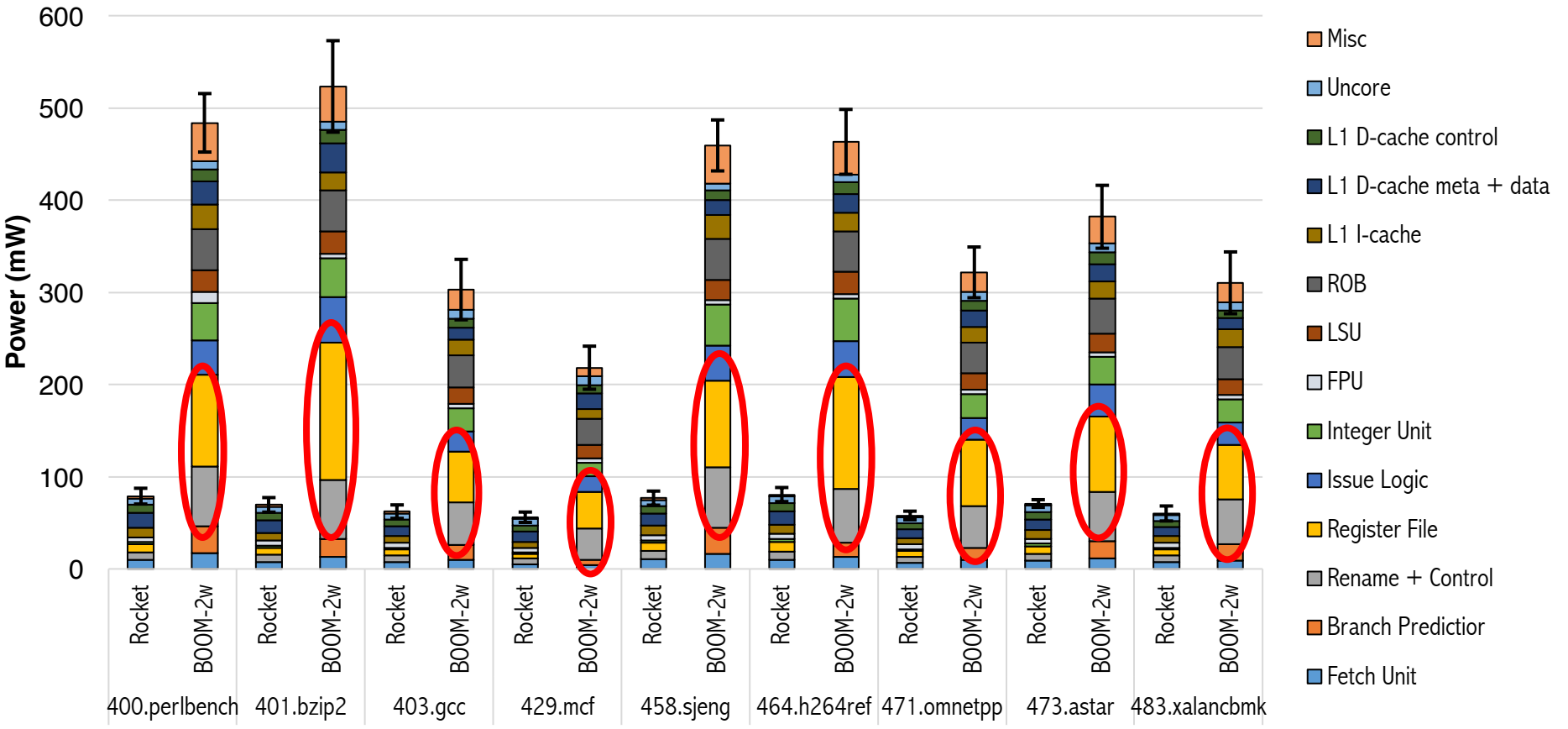
- Issued Slots / Cycle = IPC
- Empty Slots: frontend hazards, *lack of resources*(403.gcc)
- Non-issued slots: backend hazards

Strober Power/Energy Modeling



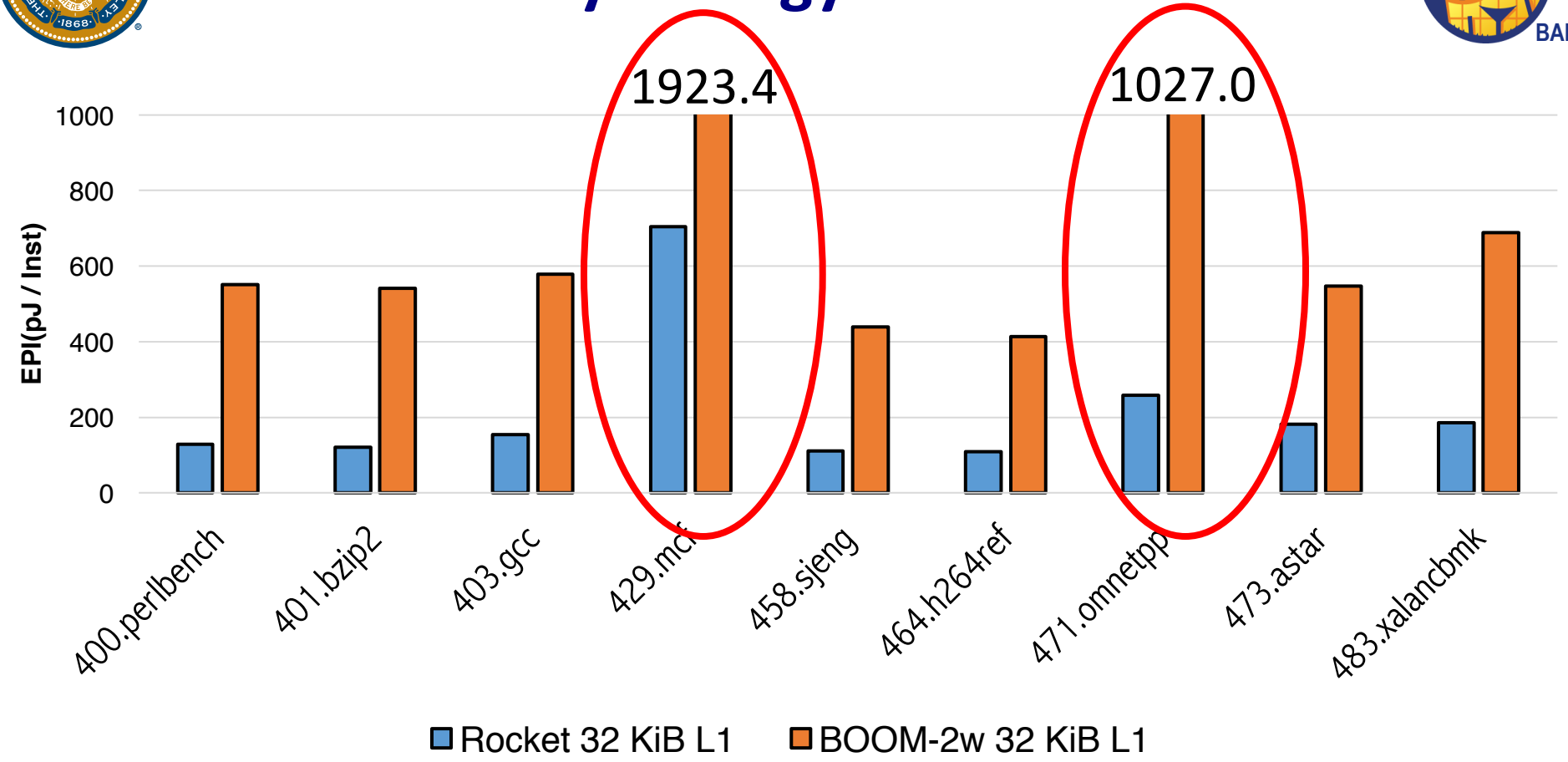
- No state warming is necessary in RTL/gate-level simulation

Case Study: Power Breakdown



- Synopsys 32nm educational technology
- 50 random sample RTL state snapshots
- Pipeline Utilization ↑ → Power ↑
- BOOM: 40% power from Rename Logic & Register File

Case Study: Energy Per Instruction



- BOOM-2w is performant, Rocket is more energy efficient
- 429.mcf, 471.omnetpp: low power, less energy efficient
  Pipeline Utilization ↑ → Energy Efficiency ↑

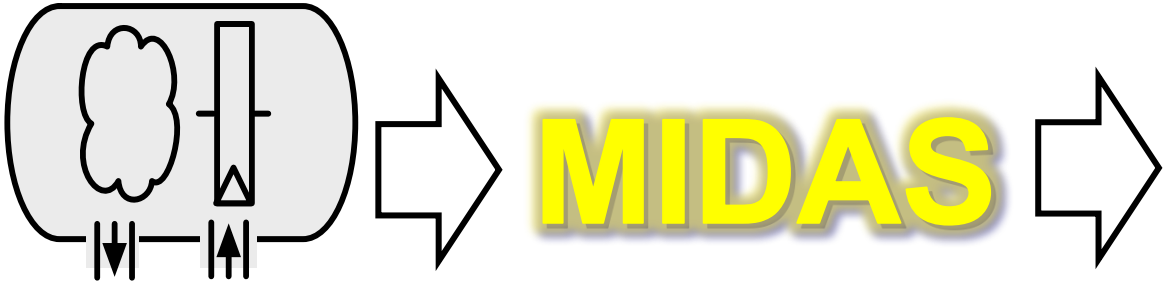
On-going MIDAS Research In UCB

- Debugging, validation for RTL designs
  - Assertion detection
  - Commit log comparison with Spike
- Memory system timing modeling
  - Realistic DRAM timing models in FPGA
- FireSim: datacenter simulation(fires.im)
  - Connect RocketChips with NICs & switch timing models
  - Amazon blog post, 7th RISC-V workshop

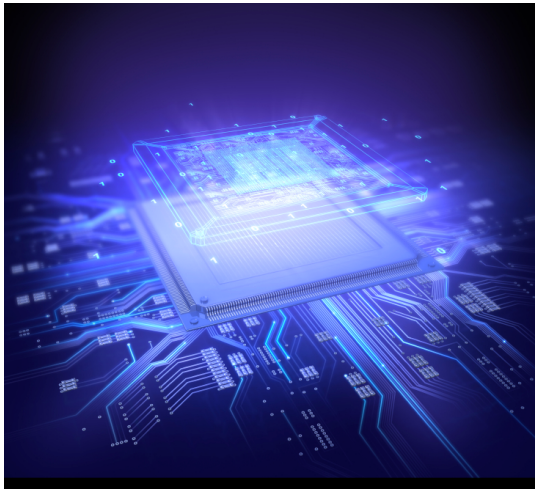
MIDAS(Strober) Is Open-Source Now

https://github.com/ucb-bar/midas-release

strober.org



Your Processors / Accelerators



- Examples:
  https://github.com/ucb-bar/midas-examples
- RocketChip template:
  https://github.com/ucb-bar/midas-top-release
- Will support various host platforms
  (**Xilinx Zynq**, Amazon EC2 F1, Intel Xeon + In-package FPGA)

Acknowledgements

- Funding:
  - DARPA Award Number HR0011-12-2-0016
  - The Center for Future Architecture Research, a member of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA
  - ASPIRE Lab industrial sponsors and affiliates: Intel, Google, HPE, Huawei, LGE, Nokia, NVIDIA, Oracle, and Samsung
  - Kwanjeong Scholarship