

**Prof Simon McIntosh-Smith**

University of Bristol

@simonmcs



# Enabling Processor Design Space Exploration with SimEng

# Some history

- Started my career at Inmos in Bristol in 1994
  - Transputers, Occam, ...
- Worked as an architect on “Chameleon” designing a SIMD instruction set for a dual-core, 64-bit, dual-issue, out-of-order CPU
- Very advanced workflow for the time
  - A single ‘master’ instruction set database drove everything
    - Documentation
    - Simulator
    - Compiler / assembler
    - Test / verification / ...



# Early design space exploration

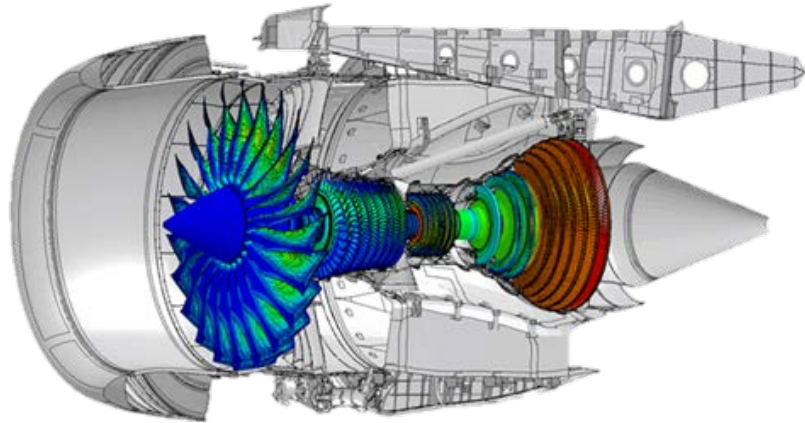
- The electronic spec-led workflow enabled rapid CPU design space exploration
- We could change most parameters about the architecture and microarchitecture, and regenerate everything quickly to try rigorous experiments
  - From the ISA to the number and spec of execution units etc.
  - Size and structure of reservation stations, memory hierarchy, ...
- I rejoined academia in 2009 and wanted to try these kinds of experiments – this wasn't as straightforward as I expected...

# Motivation – designing gas turbines ‘in silico’

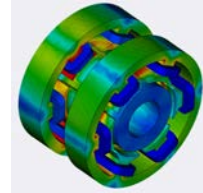
ASiMoV 5-year project with Rolls-Royce

Aiming to design new gas turbines completely in simulation

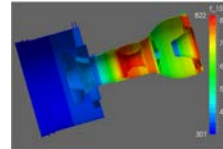
Many different kinds of physics need to be modelled simultaneously



1 Trillion degrees of freedom  
A commercial Exascale problem

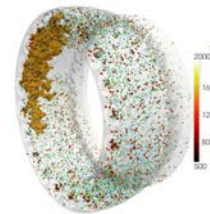


Electromagnetic



Thermo-mechanical

Contact and Friction



Computational Fluid Dynamics

Combustion

# So what do we want to be able to do for ASiMoV?

Explore the design of an “optimal” processor for 5–10 years' time?

- Core level
  - OoO parameters, number and width of vector units, prefetch capability...
- Co-processor level
  - Accelerators for vector–matrix math, FFTs, ...
- Memory hierarchy level
- Network level

## To address these questions...

... we need a fast, easy to modify, accurate-enough simulator to support semi-automated design space exploration.

In theory, we could do this with gem5 or a number of other simulators

But we found they didn't have the specific combination of speed and accuracy to let us do the things we needed.

The "Simulation Engine" was born to investigate these issues...

# SimEng design goals

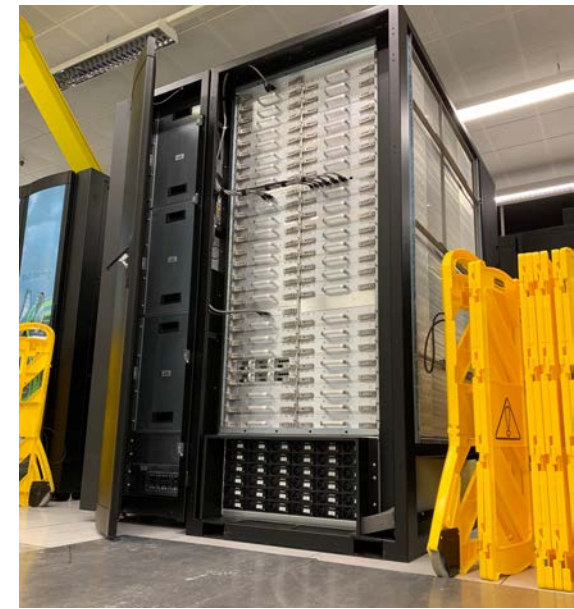
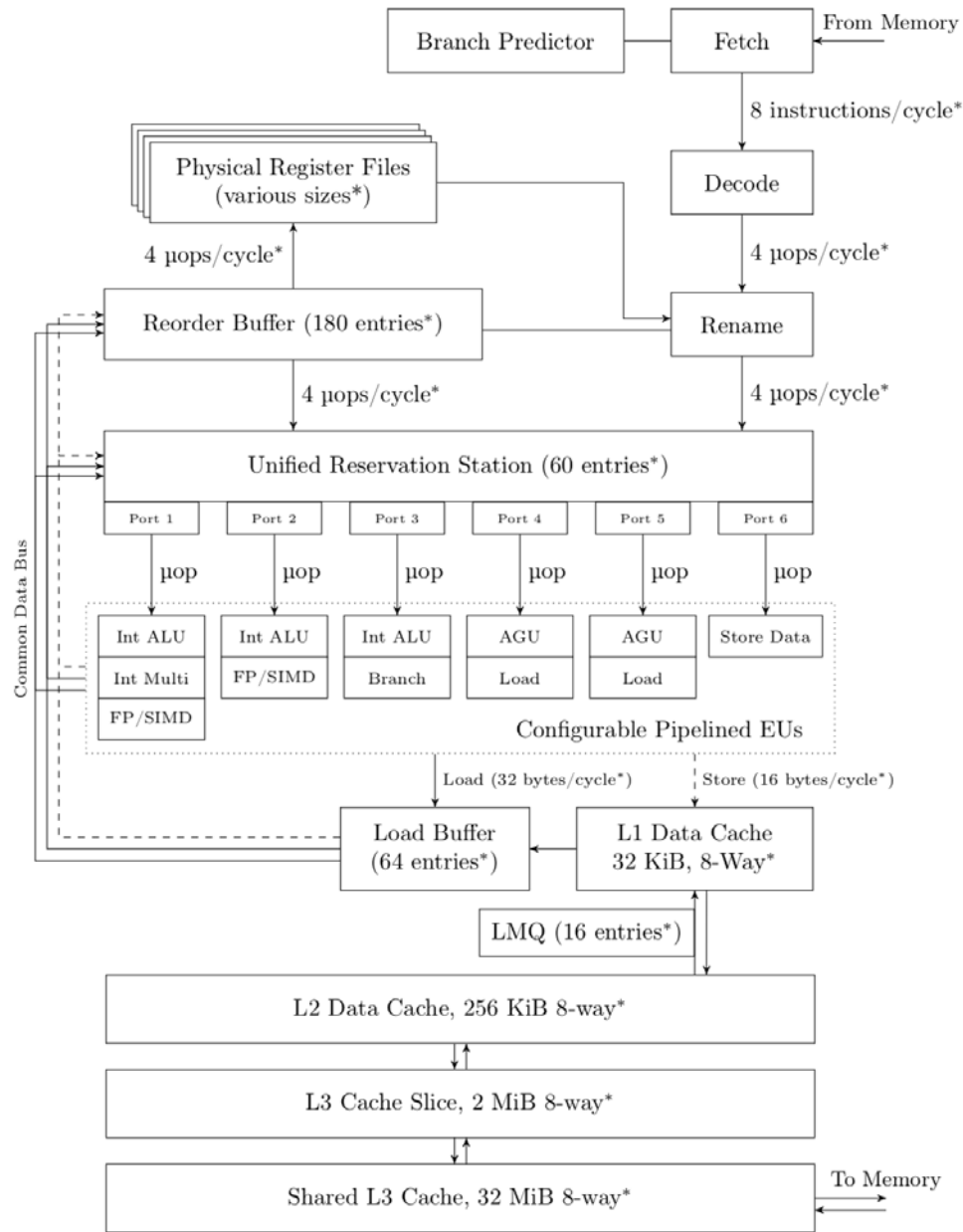
## Primary goals:

- **Fast** – millions of OoO instructions per second on a single core
- **Accurate** – within 10–20% of hardware
- **Easy to modify** – days for a radically different processor model

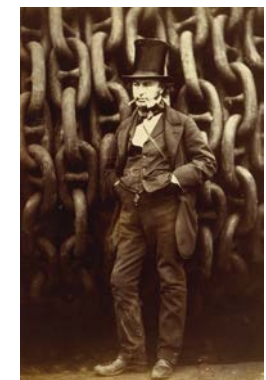
## Secondary goals:

- Use existing frameworks where possible
  - CAPSTONE for instruction decode, SST for memory hierarchy / multicore
  - Gem5-compatible tracing, checkpointing, ...

An early prototype targeted ThunderX2

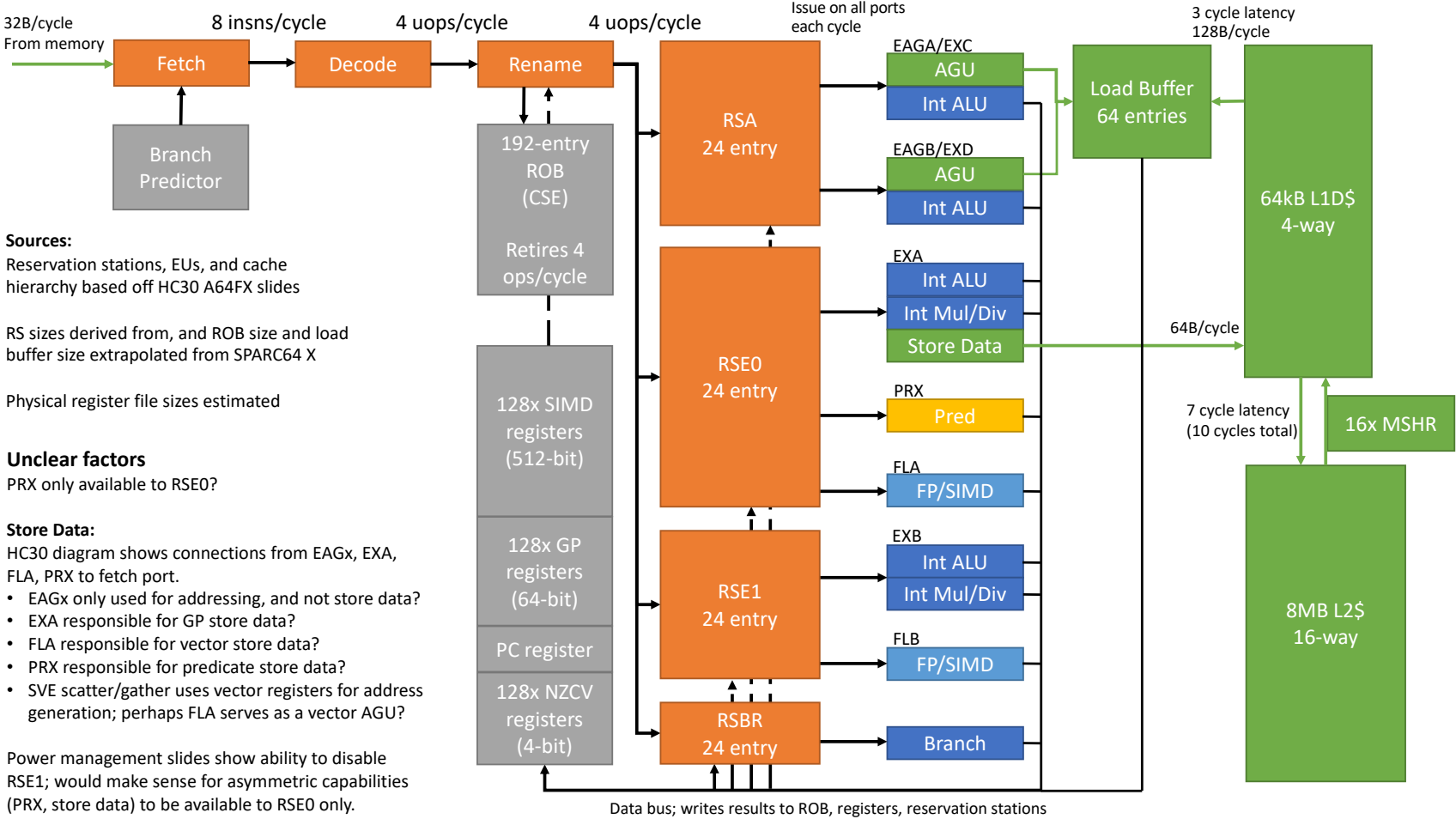


The ThunderX2 simulation was within 5-10% of the real hardware in Isambard

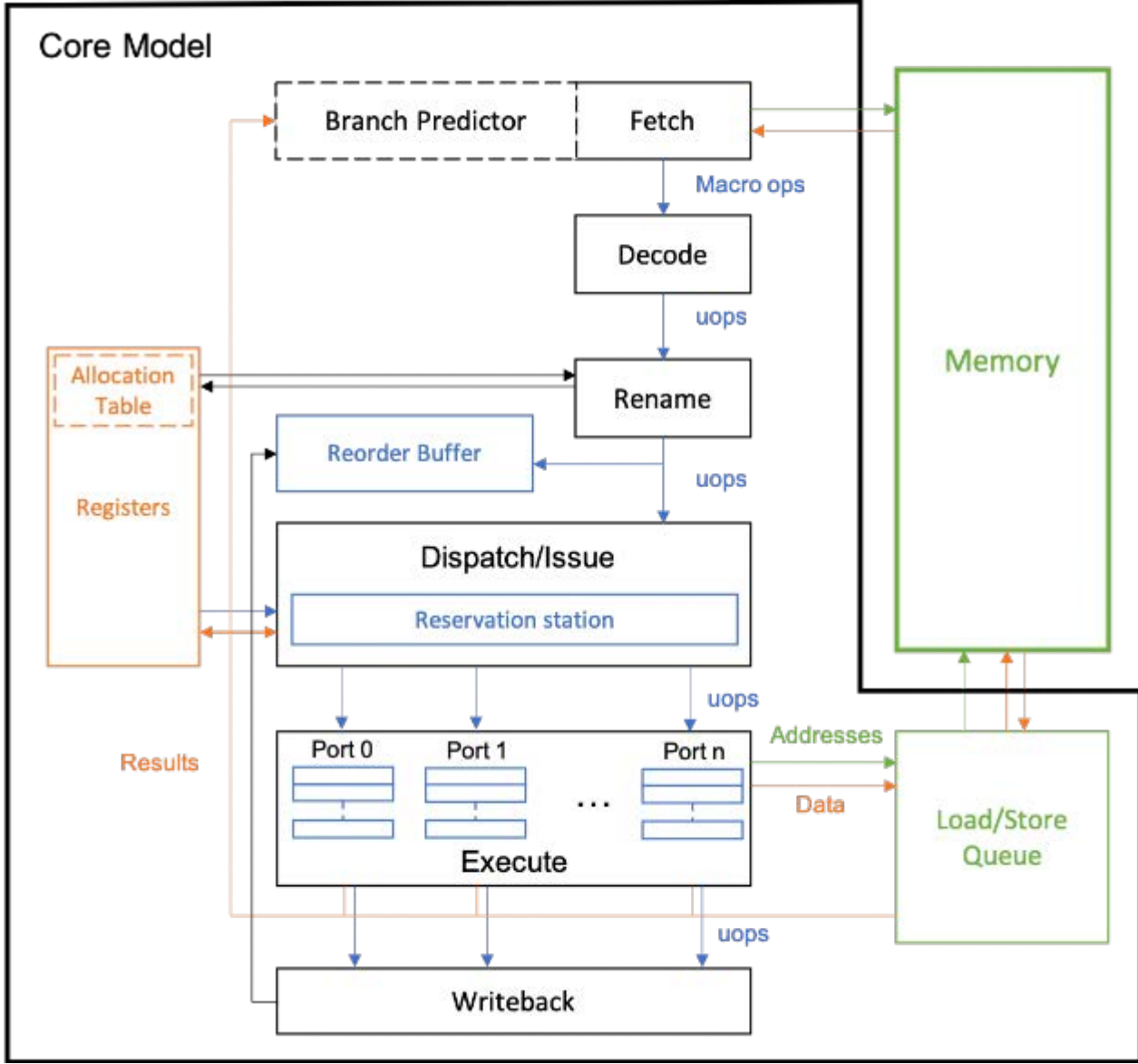




# A later version targeted Fujitsu's upcoming A64fx



# SimEng top-level design

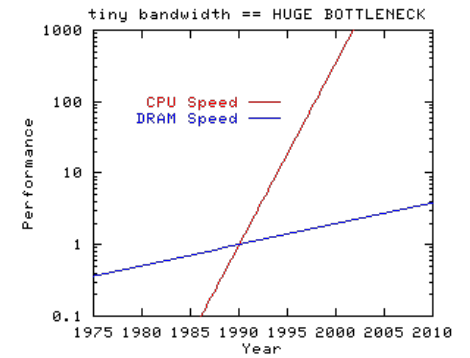


## Current status (10 months in)

- Targeting Armv8.1 initially, using CAPSTONE, which also supports x86, RISC-V, POWER, ...
  - Currently supports 230+ instructions, ~10% of the ISA
- Basic syscall emulation
  - Enough to handle libc startup routines in real binaries (compiled from C)
  - Basic printf support
  - malloc and file I/O in progress
- Current limitations:
  - Requires static binaries
  - Models up to the load/store units, planning to plug in existing models for the memory hierarchy (SimEng includes its own infinite L1 cache model)
  - Single-core only

# Early experiments

- Running McCalpin's STREAM benchmark
  - Run a problem small enough to fit in L1D cache
  - Using an out-of-order/superscalar core model, parameterized for ThunderX2
  - The STREAM run takes ~10ms on a real ThunderX2 core
- **SimEng** running on an AMD Ryzen 7 2700 @ 4.0 GHz
  - OoO takes ~26 seconds → **738 kHz / 1.84 MIPS**
  - Atomic mode runs at around **6.4 MIPS**
  - Cycle count error is **3.7%** versus real ThunderX2 hardware
- **gem5.fast** (built from Arm's sve/beta1 branch, same AMD host CPU)
  - OoO takes ~105 seconds → **171 kHz / 0.45 MIPS** (**SimEng 4.3X / 4.1X**)
  - Atomic mode runs at around **2.4 MIPS** (**SimEng ~2.7X**)
  - Cycle count error is **9.1%** versus real ThunderX2 hardware

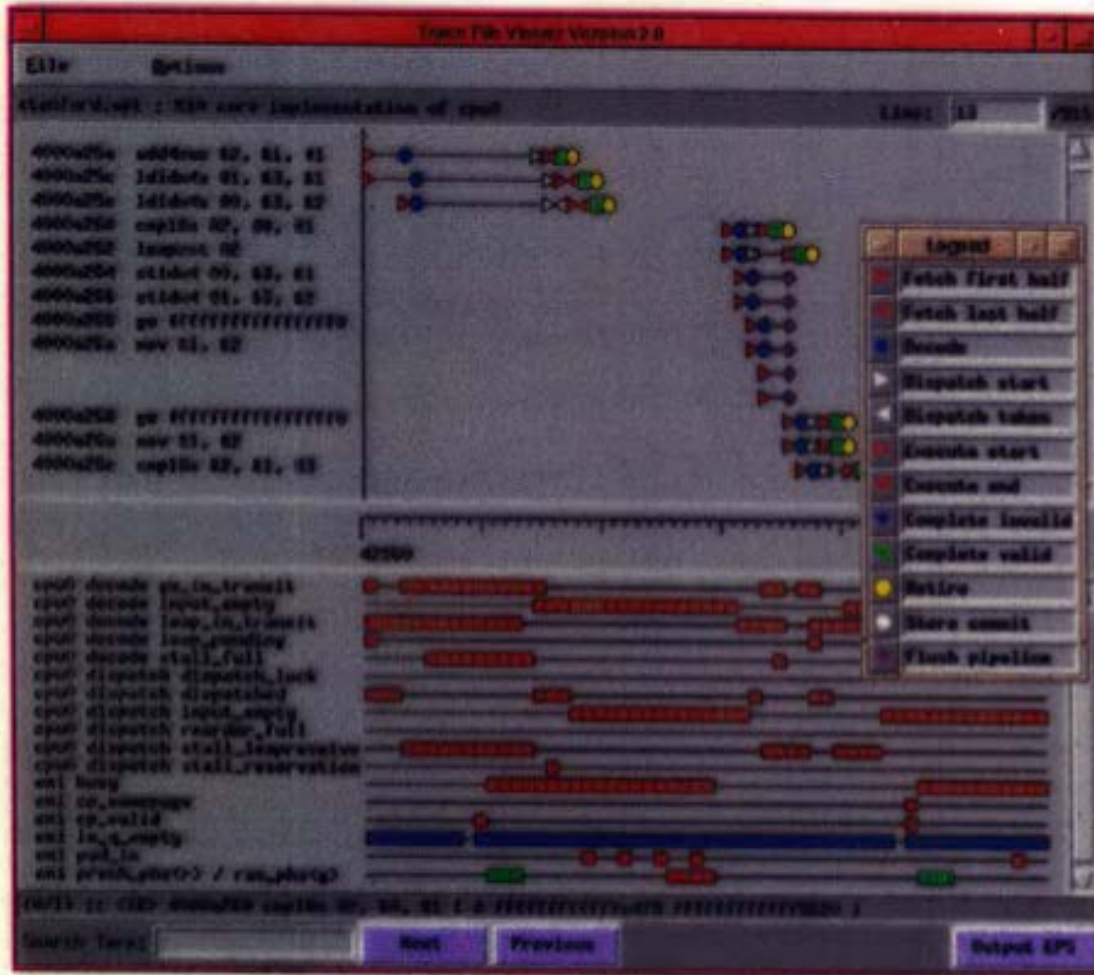


# Stats about the project

- ~10,000 lines of simple, modern C++
  - ~3,000 lines are specific for Armv8 support
  - Another ~5,000 lines of test code across nearly 200 tests
- Includes a full Continuous Integration (CI) workflow
  - CircleCI, Googletest
- Supported host platforms include: Ubuntu, CentOS and macOS
- Will be released under a permissive LLVM-style license



# Example Processor Trace



stanford.opt.trc

```

DEFINE TRACETYPE prouter
(REQ Pkt Request Send) circle yellow
(REQ Pkt Header Send) rightarrow red
(REQ Pkt EOP Send) leftarrow red
(RSP Pkt Request Send) circle yellow
(RSP Pkt Header Send) rightarrow blue
(RSP Pkt EOP Send) leftarrow blue
END

DEFINE TRACETYPE M10
(Fetch first half) rightarrow red
(Fetch last half) leftarrow red
(Decode) circle blue
(Dispatch start) rightarrow white
(Dispatch taken) leftarrow white
(Execute start) rightarrow red
(Execute end) leftarrow red
(Complete invalid) diamond blue
(Complete valid) square green
(Retire) circle yellow
(Store commit) circle white
(Flush pipeline) diamond purple
END

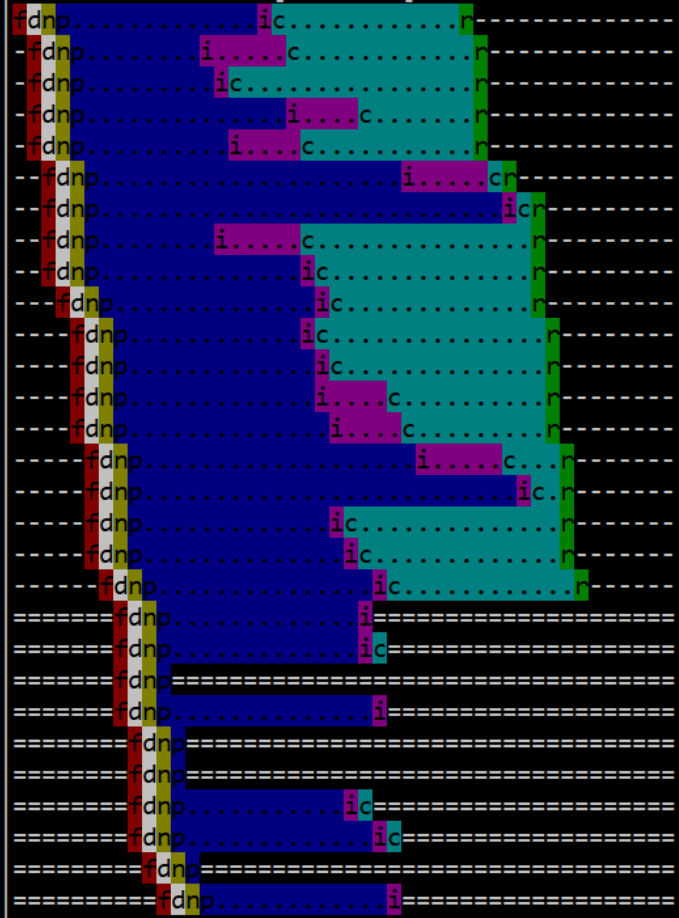
TRACE prouter pkt0 "packets on prouter"
TRACETEXT txt0
TRACE M10 pkt1 "M10 core implementation of cpu0"
TRACETEXT txt1

PROBE pcb
  
```

Trace simulator from 1996. Written in Tcl/Tk



[TIMELINE]



[INSN\_NUM]---[PC]---[DISASM]

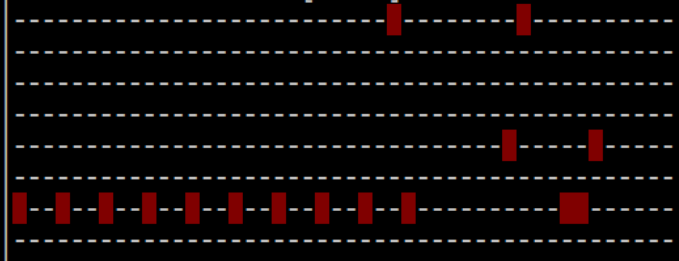
```

217282 0x00000510 b.ne #0xffffffffffffffe0
217283 0x000004f0 add x2, x0, x26
217284 0x000004f4 add x1, x0, x19
217285 0x000004f8 ldr q1, [x2]
217286 0x000004fc ldr q0, [x1]
217287 0x00000500 fmla v0.2d, v1.2d, v2.2d
217288 0x00000504 str q0, [x27, x0]
217289 0x00000508 add x0, x0, #0x10
217290 0x0000050c cmp x0, #2, lsl #12
217291 0x00000510 b.ne #0xffffffffffffffe0
217292 0x000004f0 add x2, x0, x26
217293 0x000004f4 add x1, x0, x19
217294 0x000004f8 ldr q1, [x2]
217295 0x000004fc ldr q0, [x1]
217296 0x00000500 fmla v0.2d, v1.2d, v2.2d
217297 0x00000504 str q0, [x27, x0]
217298 0x00000508 add x0, x0, #0x10
217299 0x0000050c cmp x0, #2, lsl #12
217300 0x00000510 b.ne #0xffffffffffffffe0
217301 0x000004f0 add x2, x0, x26
217302 0x000004f4 add x1, x0, x19
217303 0x000004f8 ldr q1, [x2]
217304 0x000004fc ldr q0, [x1]
217305 0x00000500 fmla v0.2d, v1.2d, v2.2d
217306 0x00000504 str q0, [x27, x0]
217307 0x00000508 add x0, x0, #0x10
217308 0x0000050c cmp x0, #2, lsl #12
217309 0x00000510 b.ne #0xffffffffffffffe0
217310 0x000004f0 add x2, x0, x26

```

**f** - fetch  
**d** - decode  
**n** - rename  
**p** - dispatch  
**i** - issue  
**c** - complete  
**r** - retire  
**=** - flushing

[PROBE]



[PROBES SELECTED]

```

branch.mispredict
L1D.cache.miss
L1I.cache.miss
rename.allocationStalls
decode.earlyFlushes
dispatch.rsStalls
fetch.branchStalls
issue.portBusyStalls

```



# Acknowledgments

- Key development team in Bristol:
  - Hal Jones, James Price, Andrei Poenaru, Jack Jones
- Funders:
  - EPSRC ASiMoV project (Advanced Simulation and Modelling of Virtual systems) - EP/S005072/1
  - Arm via a Centre of Excellence in HPC at University of Bristol

# Conclusions

- Using SimEng to explore how fast we can make a microarchitecture level simulator
  - Hope to provide useful input for the RE-gem5 project
- Also exploring how easy we can make it to make major changes to a microarchitecture to enable rapid design space exploration
- Early experiments suggest >4X speedup over gem5 is possible for a single core OoO model of ThunderX2
- We now have a fast, fairly accurate, stand-alone, single-core model in O(10,000) lines of code – what else is this useful for?