Industrial Ranting about Current Computer Micro-Architectures

Or – I've done this a while now in the Silicon Industry so you probably should listen as I say stuff

September 28, 2022
Eric Quinnell, Ph.D.
Tesla AI Hardware

Topics and Caveats

Topics

- CPU Selection in One Clear Graph
- Branch Predictors and Natural Law
- Variable Length Decoding and Why OP Caches are The Worst™
- SMT is The Worst™, except when it's The Best™
- Tesla AI Dojo D1
- Apple A14
- Quinnell's 2nd Law of Computer Arithmetic
- Shameless Tesla AI Day-2 Plug
- Q&A

Caveats

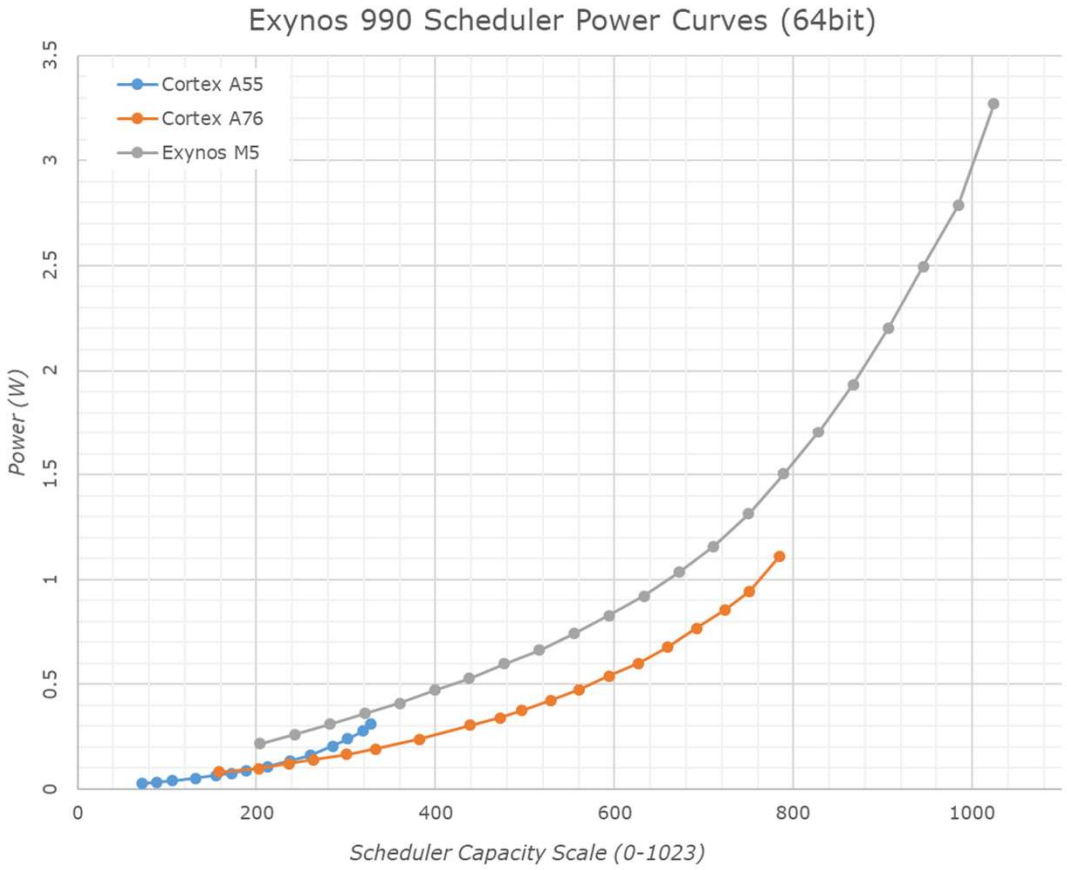
- I do not speak on behalf of Tesla. My opinions are my own and are expressed merely to educate new engineers on why they're wrong.
- All content presented here is public domain. Any speculation on CPU reverse-engineering is mine alone, but probably right
- I'm happy to talk to the Tesla HotChips 2022 "Dojo System" paper (am co-author), but you'll not hear anything not already disclosed

CPU Selection
In One Easy Graph

Evolution of the Samsung Exynos CPU Microarchitecture, ISCA 2020

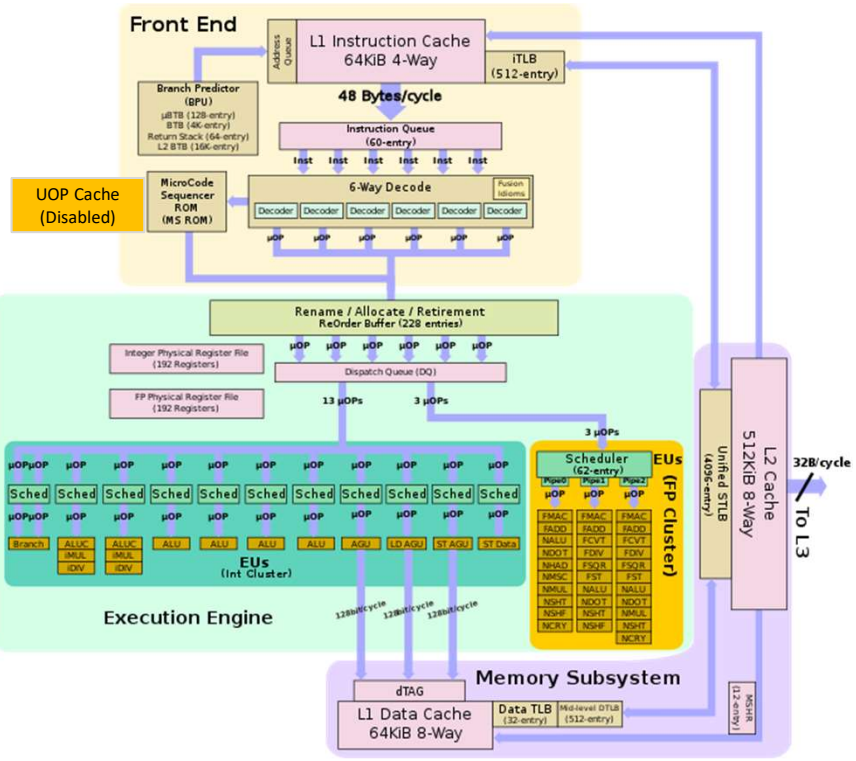
- The Power-Perf plot can determine the entire selection of an industrial CPU. In this case, this graph represents the end of the Samsung custom CPU effort
- "Burn Down" paper we wrote before shutting project down, publish the good, *future lecture on the bad/ugly (like right now)*
- ARM A76 (Ares) was a new uarch, a design triumph, labeled the "custom CPU killer"

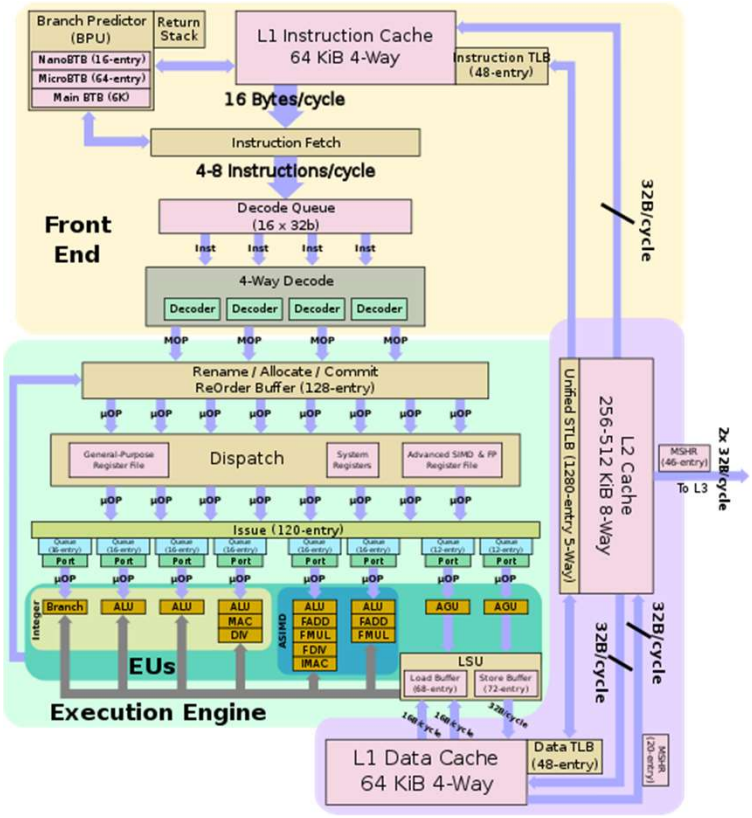
Project shutdowns are complex decisions, and the M5 could have fixed this…but quite frankly, with this single perf/power curve, would YOU fund a custom team ($100sM/year) or buy it off-the-shelf?



https://images.anandtech.com/doci/15603/sched-990.png

Mongoose M5 vs Ares A76





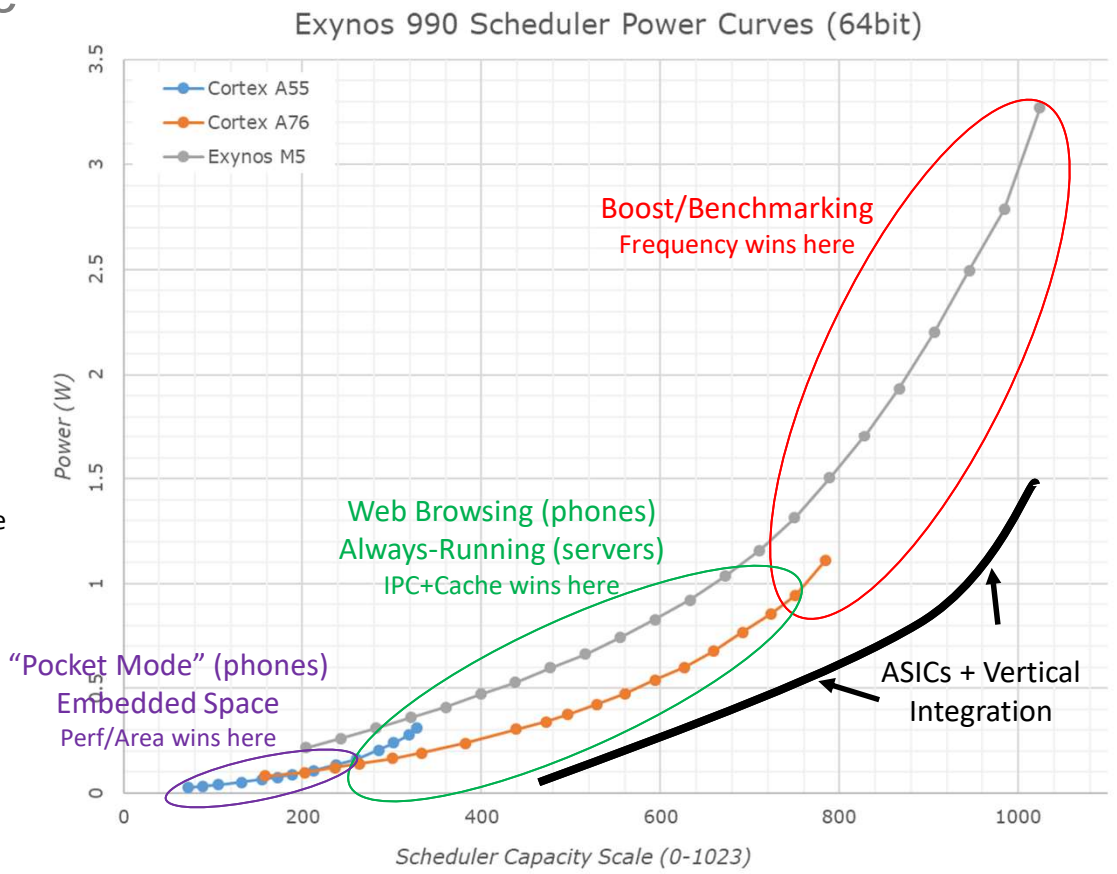
https://en.wikichip.org/wiki/File:mongoose_5_block_diagram.svg

https://en.wikichip.org/wiki/File:cortex-a76_block_diagram.svg

Reading the Power/Perf Curve

- Thermal constraints are the current greatest threat to compute
- Boost Mode and Benchmarking isn't useable compute – greater than 2GHz sustained is rare
- Performance is won by IPC and fundamentals
  - Cache size matters
    - Not just IPC. Power to DRAM >> power to Cache. Less I/O bandwidth.
  - IPC Latency matters more than frequency
    - Mongoose M5 16-deep mis-predict vs Ares A76 11-deep mis-predict
  - ROI on performance features matters
    - "Will this feature provide more IPC than the cache-area equivalent?"

- ASICs with Vertical Integration ***always win*** on their application, full stop. Just have to pay the engineers for it.
  - Citations? Sure!
    - Apple A6-M1 with IOS
    - Google's TPU with Deep-Mind, Search
    - GPU Shaders with API drivers
    - Is this really in question?



https://images.anandtech.com/doci/15603/sched-990.png

“Real Code” is Indirect Threading

```
start:
  ip = &thread  // points to '&i_pushA'
  jump *(*ip)  // follow pointers to 1st instruction of 'push', DO NOT advance ip yet
thread:
  &i_pushA
  &i_pushB
  &i_add
  ...
i_pushA:
  &push
  &A
i_pushB:
  &push
  &B
i_add:
  &add
push:
  *sp++ = *(*ip + 1)  // look 1 past start of indirect block for operand address
  jump *(*++ip)  // advance ip in thread, jump through next indirect block to next subroutine
add:
  addend1 = *--sp
  addend2 = *--sp
  *sp++ = addend1 + addend2
  jump *(*++ip)
```

https://en.wikipedia.org/wiki/Threaded_code#Indirect_threading

For Whom are you building machines?

- Governments use Fortran
- HPC, Gaming use C/C++ (and Excel spreadsheets)
- AI uses MACs and Python. (That’s pretty much it.)
- The Internet, VMs use JS and JITs ***(this is most of the world)***
  - https://browserbench.org/Speedometer2.0/
  - That’s ~1B instructions / iteration to sort a 100-deep text list.
  - Try it on your phone and laptop right now, see which is faster
  - Indirect branches and integer ALUs
  - This is for SW’s dev benefit, not HW. There are exponentially more of them (SW), they win.
- We knew about this phenomenon in 2005. What is SPEC benchmarking again?
  - https://www.amazon.com/Virtual-Machines-Versatile-Platforms-Architecture/dp/1558609105
- The newest CPU uarchs are **heavy** with indirect-branch target storage, algorithms like IT-TAGE (Seznec).

Side Quest – Speaking of Branch Prediction…

- Dr. Daniel Jimenez (Texas A&M) in his perceptron-based branch-prediction research shows that global-history (ghist) auto-correlation on generic code follows a power-law. (I told you, Daniel and Jim that I'd bring this up someday)

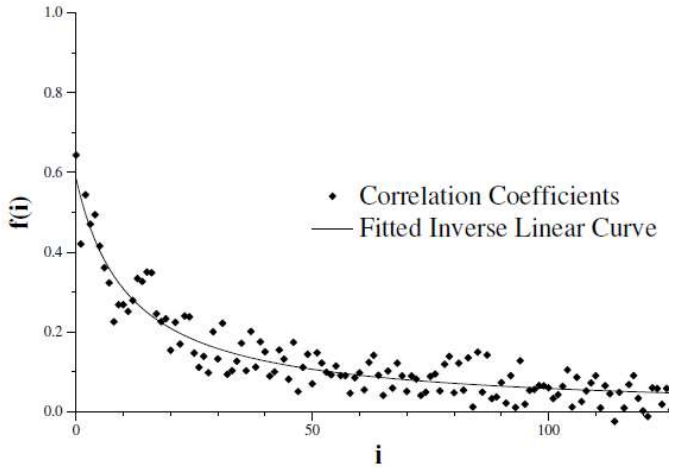
- Dr. Tarjan and Dr. Skadron (Univ of VA) noted "Theta" for generic branches to be

$$\theta = 1.93 numTables + \frac{numTables}{2}$$
$$= (1.93 + 0.5) numTables$$
$$= 2.43 * numTables$$

- Dr. James Dundas (ARM) recognized some time ago that:

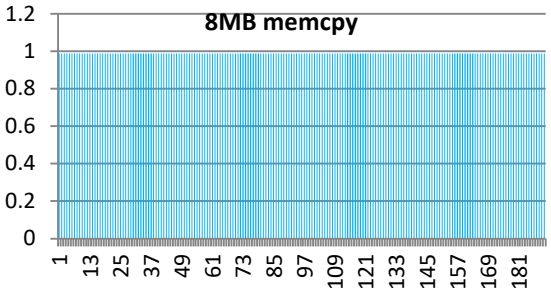
$$\varphi = \frac{1-\sqrt{5}}{2} = \sim 1.618 = \text{"golden ratio"}$$
$$\varphi = \frac{3\varphi}{2} = \sim 2.427$$
$$\theta = \frac{3\varphi}{2} * numTables$$

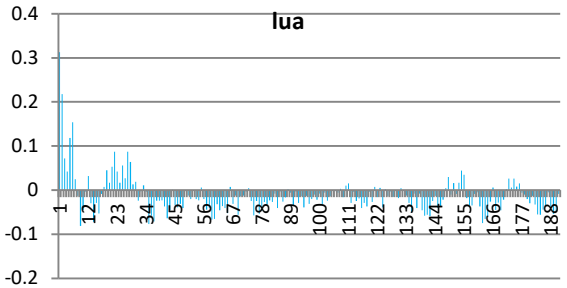
- Dr. Seznec has published GEometric History Length and O-GEHL predictors
- Dr. Yasuo Ishii (ARM) extended GEHL to local history
- Dr. Quinnell (me) asks – "Is code, therefore, constrained by natural law!?"
- Apparently you need a PhD to notice this kind of thing…

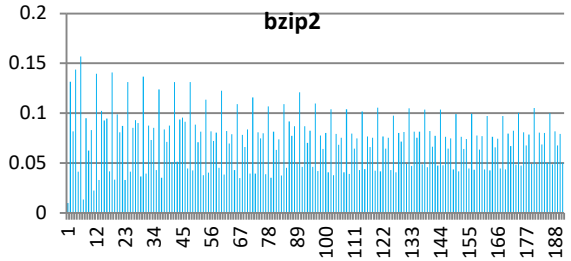


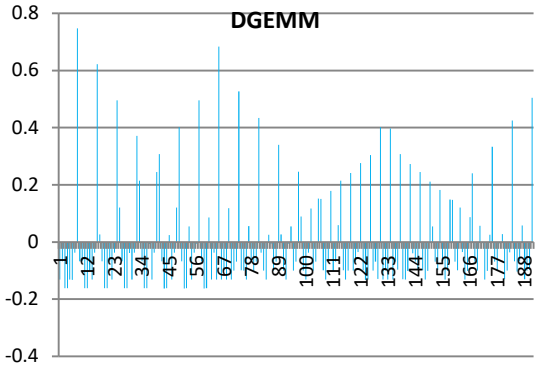
[Jimenez, lots of papers]

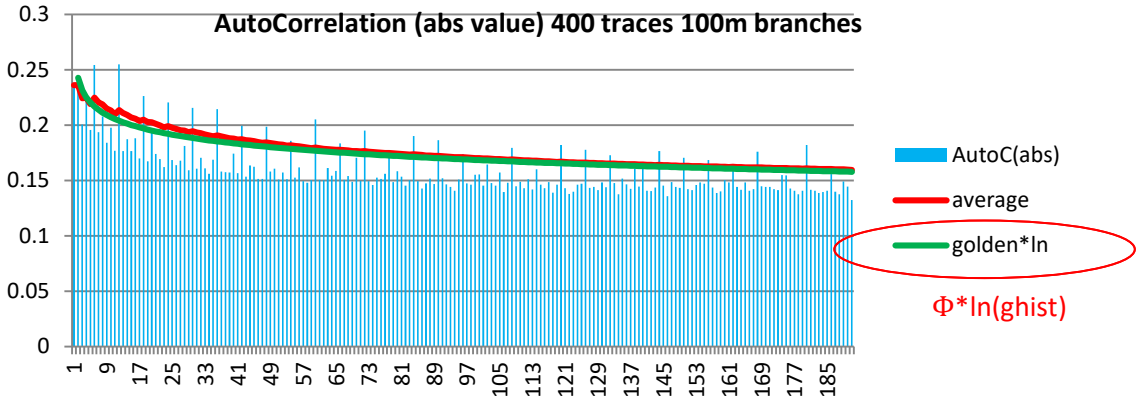
Auto-Correlation of Ghist vs Branch Time Lag, CBP-5











Fibonacci in CBP-5

{0-3,0-5,1-8,2-13,3-21,5-34,8-55,13-89}

- I replaced all TAGE parameters with Fibonacci geometric series equivalents.
  - MPKI was exactly the same
  - Don't believe me? Do it yourself! CBP-5 is public and available
- SHP – vs several successful hashes and their variants, various sizes, tables.
  - Example below. Wasn't always "lowest lowest" mpki, but certainly unbeatable when constraining ghist length

Run	MPKI	#Ghist bits
Variant of published SHP hash	4.720	160
Fibonnaci	4.718	89

- Code, and therefore AI Robots, is constrained by Natural Law!
  - We can (maybe) defeat them if the take over!
  - Asimov is smiling.
- See me after lecture for a tin-foil hat

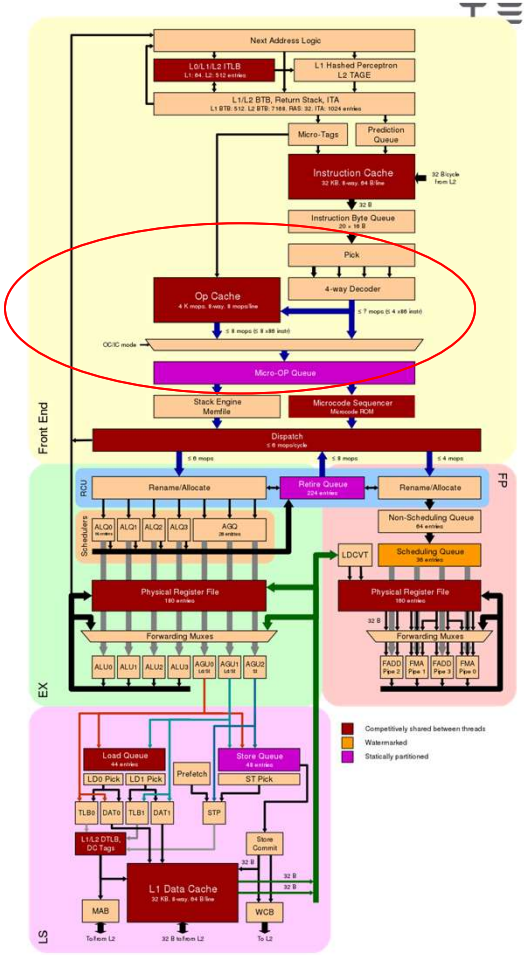
Op Caches are The Worst™

AMD's Zen 2

- Notice the "ITA" – Indirect Target Array. Notice its expansion in subsequent CPUs..
- 4x Int ALUs and 3x AGUs, LD/LD/ST
- 6-wide INT dispatch, 8-wide Retire

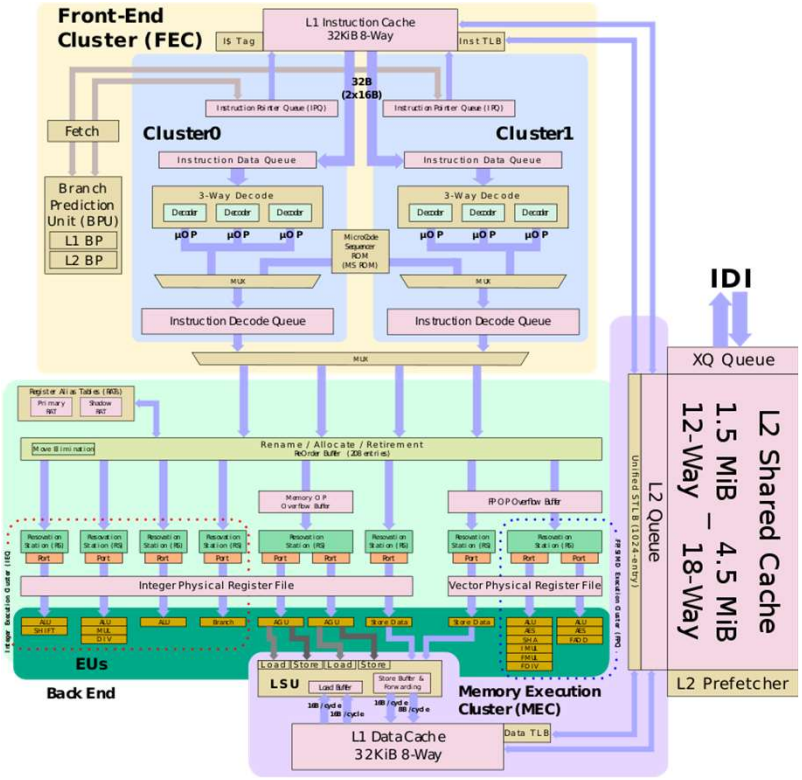
Wait – 4-wide Decode!?! For 8-wide Retire?!

- Why? B/c x86 ISA is a variable-length decode
  - Various techniques to up IPC – Marker bits, pre-decoders, collapsing instruction queues, parallel speculative decoding
- Solution here is the OP Cache
  - Repeated kernels can read out up to 8-mops / cycle
  - OP Caches derive historically from Trace Caches, originally intended to increase zero-bubble branches and instruction throughput
- Similar Block-Diagram for wider MOP decode in Intel's Skylake uarch
  https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(server)



https://en.wikichip.org/wiki/amd/microarchitectures/zen_2#Block_Diagram

Op Caches are The Worst ™



OP caches are large, complex, make branch prediction logic replicate, and use lots of power.

1. ~100-bit read/write array (vs the 16-32-bit original instruction)
2. Redundant branch prediction hardware
3. OP cache specific fetch-line buffering
4. OP cache micro-tags and CAMs in parallel with decode path

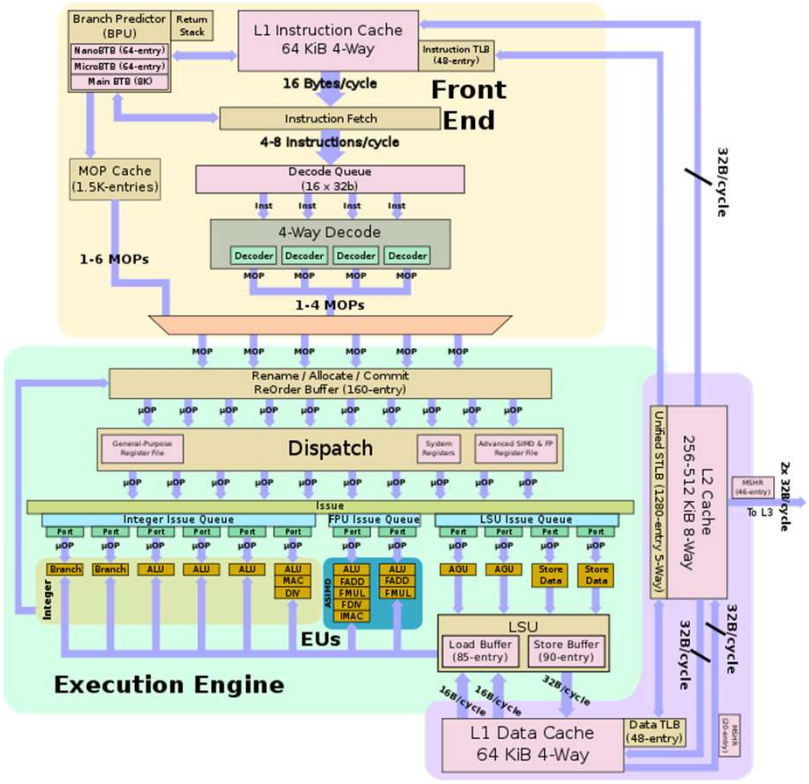
1-4 vs saving Decode power. Are we sure it's less overall?
(Mongoose M5's UOP Cache was **not** lower power overall)

Intel's Tremont – No-OP Cache

- Intel's Tremont **avoided OP caches** altogether, solving the x86 decode/throughput by alternating basic-block fetches across 2x fetch paths
- Result is an effective "2x3-wide decode" on two basic-block fetch
- Note – are the "Reservation Stations" classic Tomasulo's algorithm, or are they clever PRFs? Each has a tradeoff.

https://en.wikichip.org/wiki/intel/microarchitectures/tremont

OP Caches in ARM? RISC-V?



https://en.wikichip.org/wiki/arm_holdings/microarchitectures/cortex-a77

- Cortex A-77 did add a “MOP Cache”
- 6-wide MOP “decode” vs 4-wide native decode
- A-77 still supports AArch32
  - Samsung Mongoose M1-5 supported AArch32 and Thumb
- ARM is dropping support for Thumb/AArch32 in Makalu (2022) generation, making all instructions 32-bit aligned

- RISC-V has optional variable-length instructions (16-bit) for code compression. There is reserved space in case future instructions want to go > 32-bits.

Variable length vs OP Cache tradeoff

- Variable length instructions allow code compression – great for embedded applications
- High-perf CPU indirect-threaded code needs > 4-wide decode to compete.
- **OP caches are a variable-length instruction solution**

Trends

- ARMv9 is transitioning to fixed length while RISC-V is adding variable-length with reserved expansion
- Is a non-fixed instruction alignment considered “RISC”?

@ RISC-V – perhaps the proper way to support future instructions is not to infinitely expand the space, but rather allow ***for deletion*** of antiquated instructions.

SMT – the good, bad, and always ugly

This really should be an entire talk
(and I need to take more time with citations here…maybe a grad student should do this as a paper)

SMT bad

Claim: *SMT in an out-of-order, virtualized, non-shared context will be beaten by single-threaded SMP*

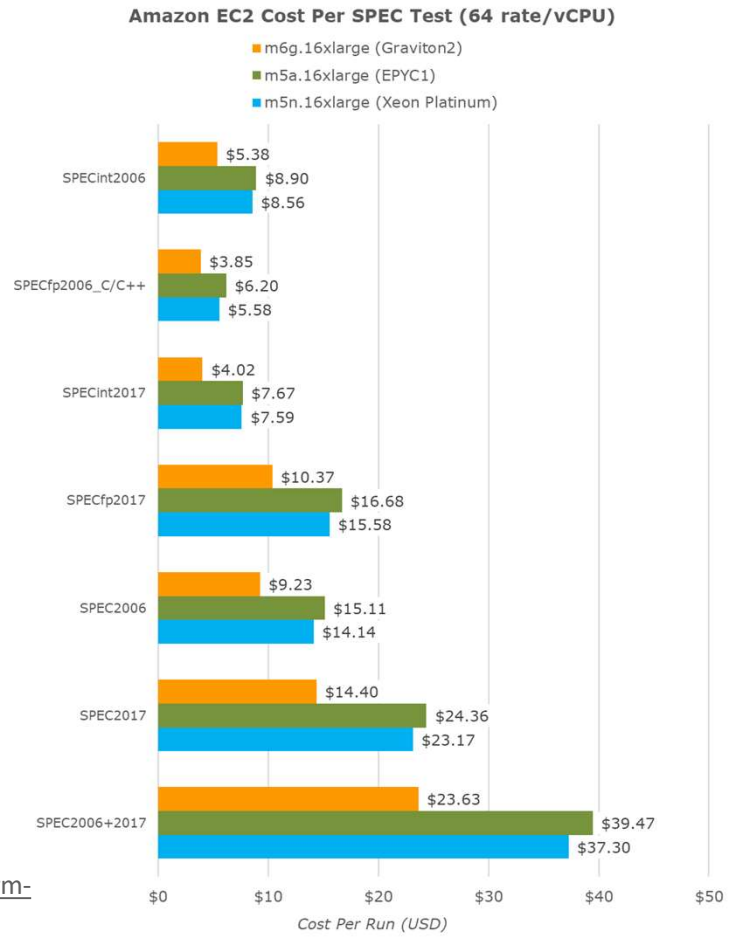
- 2x85% IPC SMP cores at ½ the size beat ~120% IPC SMT
- Full-context SMT takes >> 10% area in reality. Find a die shot (not a paper or claim) where this is untrue.
- "Noisy neighbor effect" with cross-polluting caches
- Spectre-style security problems
- HW-multi thread scheduling unable to predict SW multi-thread intent, forecasting, prefetching
- All threads must sleep before powering down
- Out-of-order resources lost to hold variable SMT retire state
- Turn off hyperthreading and see for yourself. Did you lose anything?

SMT good

("Poor-man's out of order")

Claim: *SMT in a shared-context, SW controlled, scalar and in-order throughput machine gains scale-able benefits of concurrent control and execution*

- Qualcomm's DSP Hexagon
- Tesla's Dojo AI D1



https://www.anandtech.com/show/15578/cloud-clash-amazon-graviton2-arm-against-intel-and-amd/9

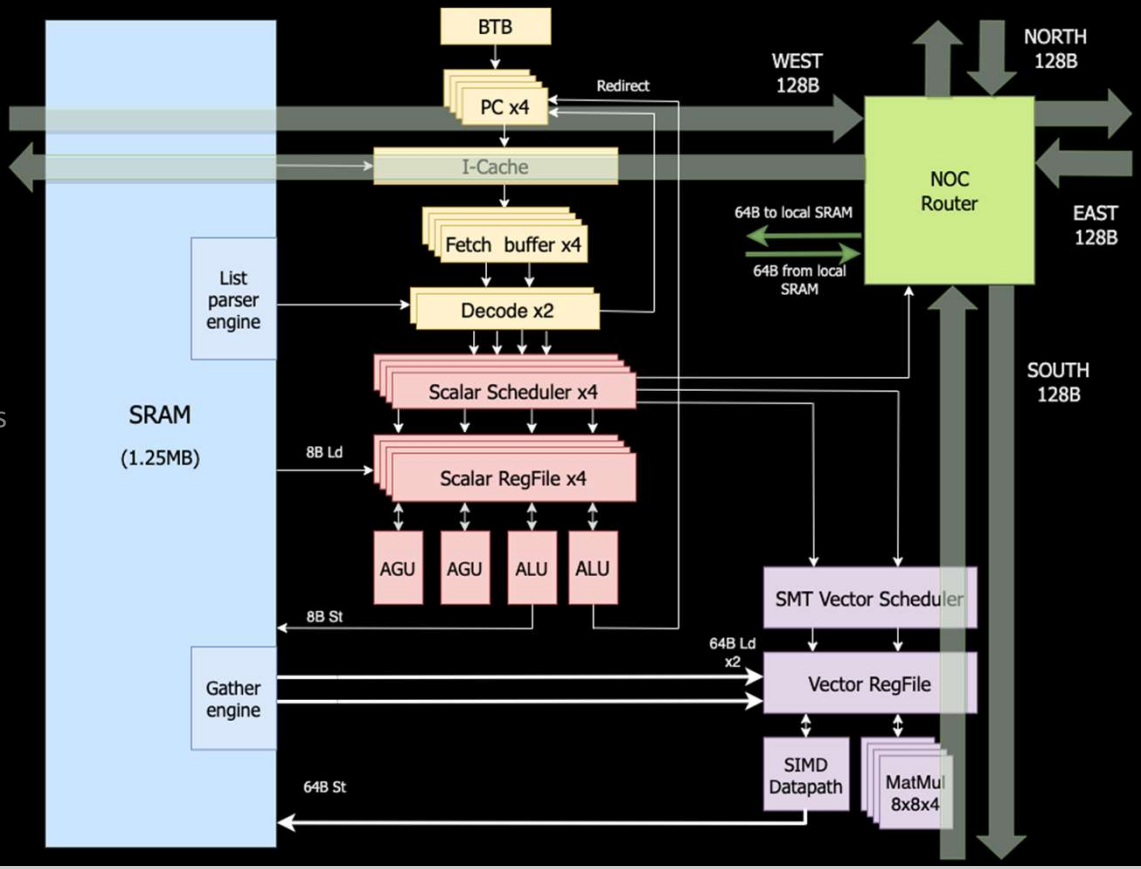
Microarchitecture of the DOJO node

High throughput, general purpose CPU

DOJO nodes are full-fledged computers
- Dedicated CPU, local memory, communication interface

Superscalar, multi-threaded organization
- Optimized for high-throughput math applications rather than control heavy code

Custom ISA optimized for ML kernels



Processing pipeline

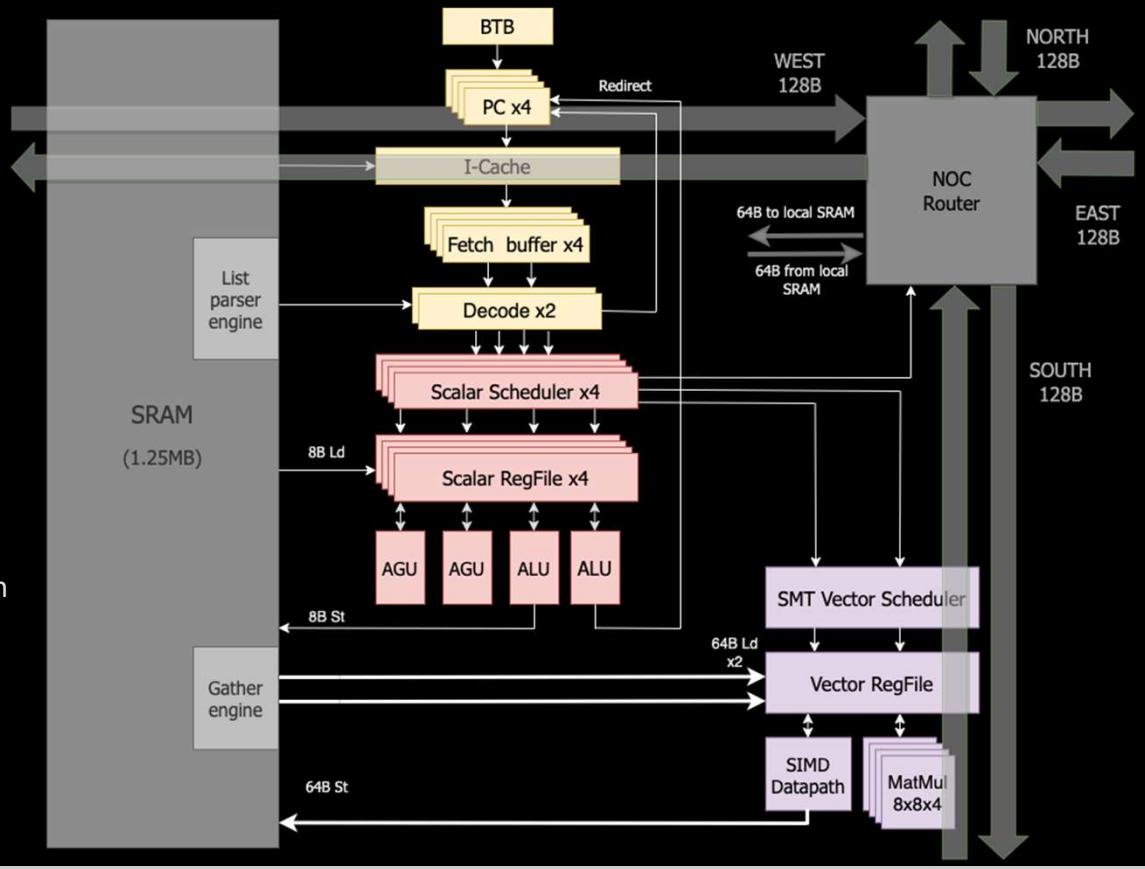
32B fetch window holding up to 8 instructions

8-wide decode handling 2 threads per cycle

4-wide scalar scheduler, 4-way SMT
- 2 integer ALUs
- 2 address units
- Register file replicated per thread

2-wide vector scheduler, 4-way SMT
- 64B wide SIMD unit
- 8x8x4 matrix multiplication units

SMT support focuses on single threaded application
- No virtual memory, limited protection mechanisms, SW-managed sharing of resources
- Typical application uses 1 or 2 compute threads and 1-2 communication threads



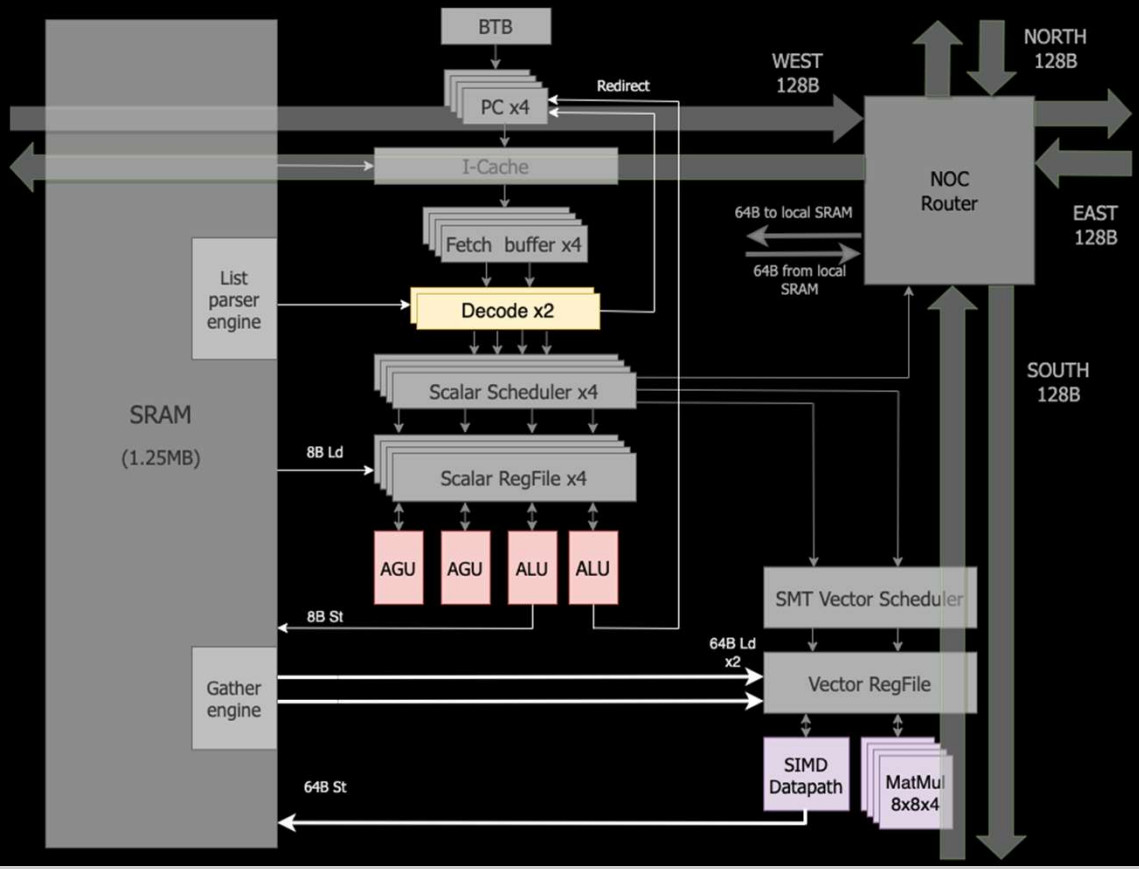
Datapath

Pipeline width reduces progressively
- 8-way in Decode
- 4-way in the Scalar engine
- 2-way in the Vector engine

Simple primitives can execute early in Decode
- Looping, list parsing
- Predication

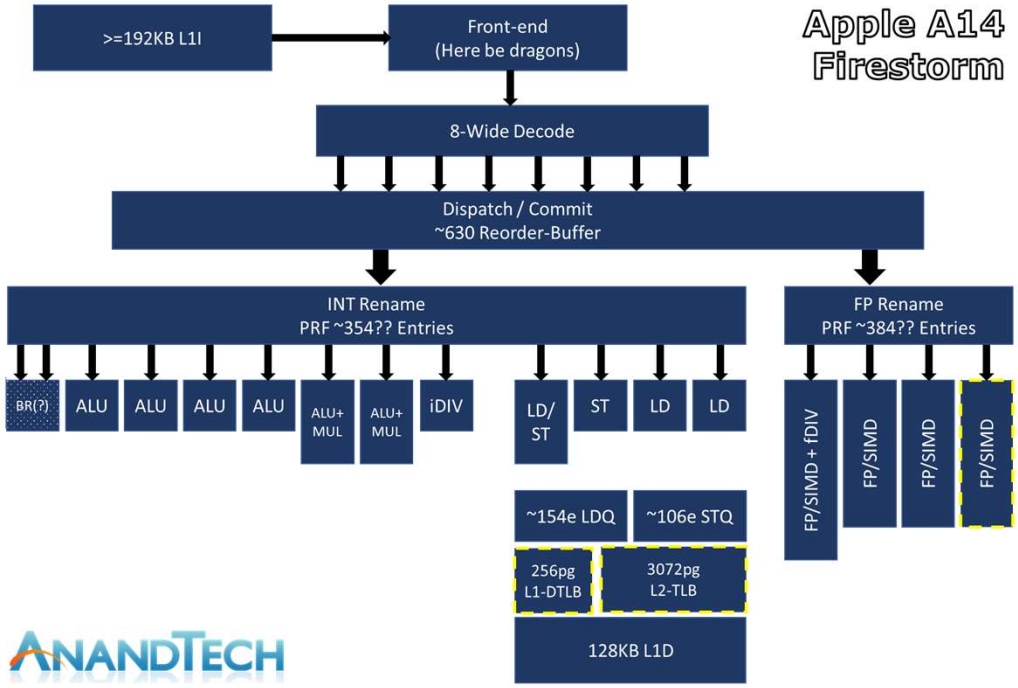
Scalar instructions
- Regular integer code, address generation
- Network synchronization primitives

Vector datapath
- 8x8 matrix multiplication instructions
- 64B SIMD pipeline
- Special ML formats (CFP8, storage CFP16)
- Special ML instructions (e.g., stochastic rounding, etc.)



What about Apple?

- Massive Caches
  - Not really a mystery – seen from space
  - Likely the greatest contributor to low energy – extremely fewer accesses to DRAM than competitors
  - No other ARM competitor comes even close to paying the area for this significant advantage
- Measured massive Indirect Branch Target chains
- Tons of simple ALUs, 3xLD to feed it
- Extremely short mis-predict pipe (IPC >> frequency)
- **No OP cache**, 8-Wide decode fixed-width instructions.
  - Why not 10-wide? Why not more? Fixed width decode is cheap
- Vertical Integration with SW
  - Adding cute ISA instructions does not make up for lacking this
- SMP, not SMT



https://images.anandtech.com/doci/16226/Firestorm.png

Bonus Round: Quinnell's 2nd Law of Computer Arithmetic

You cannot add faster than you can add

Fallacies include:

- Trying LUTs/ROMs/case statements instead of an adder for your "but I don't need all adder bits" adder
- Attempting generate/kill derivatives on LOPs, but wanting a precise result (we call that an "adder")
- Trying to be clever with carry-only trees combined with other carry only trees and mux inferring a sum bit (we call that an adder)
- Trying to deduce an adder answer or bit without "the carry". Works for 99% of all possible cases you say, and only not that 1% "propagate carry" case
  - The most common "compares" in all code are "== 0" or the identity compare. Which is a full carry. Code Gaussian doesn't match the theoretical "universal hash" math spread.
- Trying to be clever with circuit tricks like domino logic or cascaded FETs (they're adders with not-cmos!)
- "End-around-carry (EAC)" adders (it's an adder and an incrementor that go as fast as…themselves)
- Thinking a $popcnt isn't just a MUL compression and adder
- Trying "higher radix" and writing papers about non-memory formats to avoid carries
  - Which you translate to/from 2s complement or IEEE format, which requires…yup, an adder and full carry BOTH ways. So your total time processing same answer is worse, congrats!!
  - Nobody does kernels resident only in regfiles

If it has ANYTHING to do with adding, do "assign z = a + b" and **watch synthesis beat you at it in all forms**.

Really, I can't read any more of these. Research somewhere else, put a fork in it, this one's done. Very little in uarch has been scrutinized quite like the adder.

(What's the 1st law? I don't remember, nobody remembers the 1st law of anything)

Shameless Plug: AI Day 2

- Sept 30, 2022
- ~7:30-ish PM livestream
- Robots and Cars and Supercomputers
- You're not watching "sportsball" anyway, you nerds, you

Awesome Tesla technical links

- https://www.youtube.com/watch?v=jPCV4GKX9Dw – CVPR '22
- https://www.youtube.com/watch?v=hx7BXih7zx8 – ScaledML '20 – at 8:40, see the "stop sign" problem
- https://www.youtube.com/watch?v=j0z4FweCy4M – AI Day 1, '21
- https://www.youtube.com/watch?v=rsBbt3TxKGg – 3rd party video, but Giga Presses are cool
- See Tesla HotChips '21 and '22 (Dojo slides are from the '22 talks)

Car Picture