future of AI-HW & GPU in SoCET

April 30th, 2026

background

~~future~~ of AI-HW & GPU in SoCET + little bit about future

April 30th, 2026

motivation

history & growth

future plans

who should join?

motivation

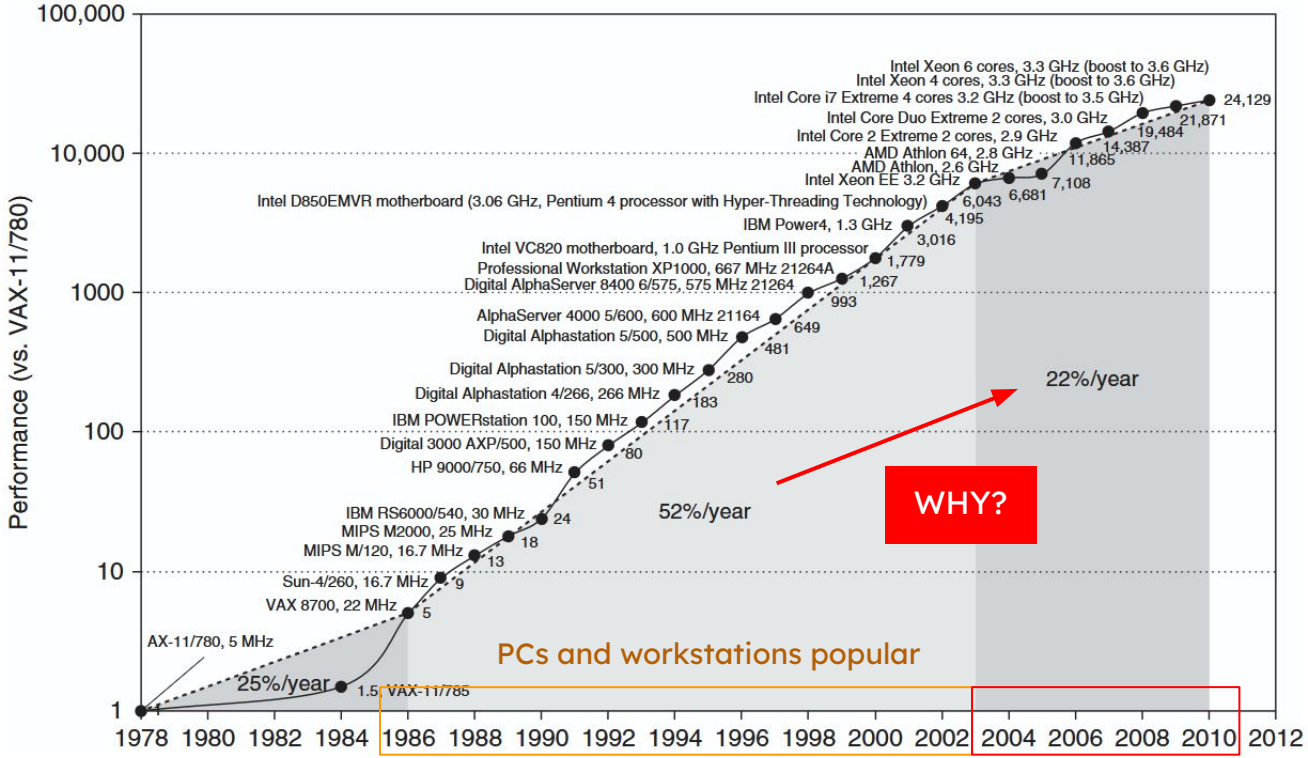
history & growth

future plans

who should join?

key ideas:

- ❑ specializing silicon is important
- ❑ CPUs are *not enough* today
- ❑ programmability required
- ❑ transparency limited



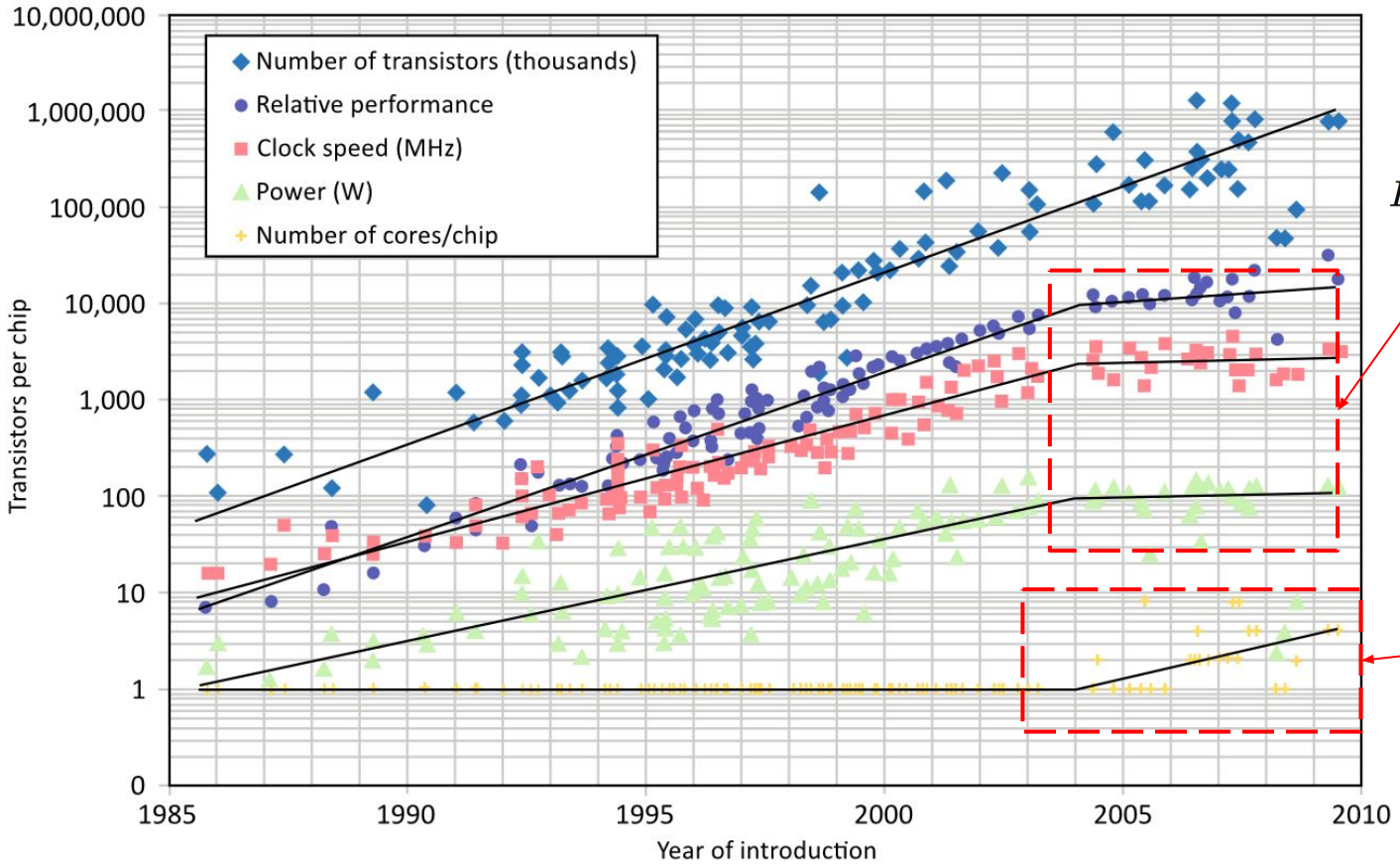
SPECInt performance slowed down

Hennessy, J. L., & Patterson, D. A. (2019). *Computer Architecture: A Quantitative Approach* (6th ed.). Morgan Kaufmann.

?

?

("Dennard Scaling")



$$P_{total} = \alpha C V_{dd}^2 f + I_{leak} V_{dd}$$

computer architects in the late 1990s had "free lunch" by scaling (1) voltage (2) frequency (etc.)

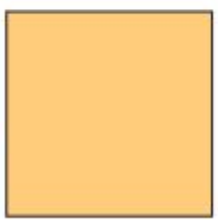
we moved to multi-core systems instead (more cores running at same *V* and *f*)

Hennessy, J. L., & Patterson, D. A. (2019). *Computer Architecture: A Quantitative Approach* (6th ed.). Morgan Kaufmann.

benefits of scaling were limited by the *“power wall”*

Node	45nm	22nm	11nm
Year	2008	2014	2020
Area^{-1}	1	4	16
Peak freq	1	1.6	2.4
Power	1	1	0.6
		$(4 \times 1)^{-1} = 25\%$	$(16 \times 0.6)^{-1} = 10\%$

Exploitable Si
(in 45nm power budget)



(2) complex hardware in CPUs could not be kept busy enough to justify its power and area cost.

(1) "dark silicon" (had to be switched off)

C. Akass, "ARM wrestles with dark silicon," Mar. 2010, Available: http://www.theinquirer.net/inquirer/feature/1598659/arm-wrestles-dark-silicon 7

benefits of scaling were limited by the *“power wall”*

adding functionality to single core limited by *“utilization wall”*

Solution:
Specialized domain-specific cores sitting alongside general-purpose core

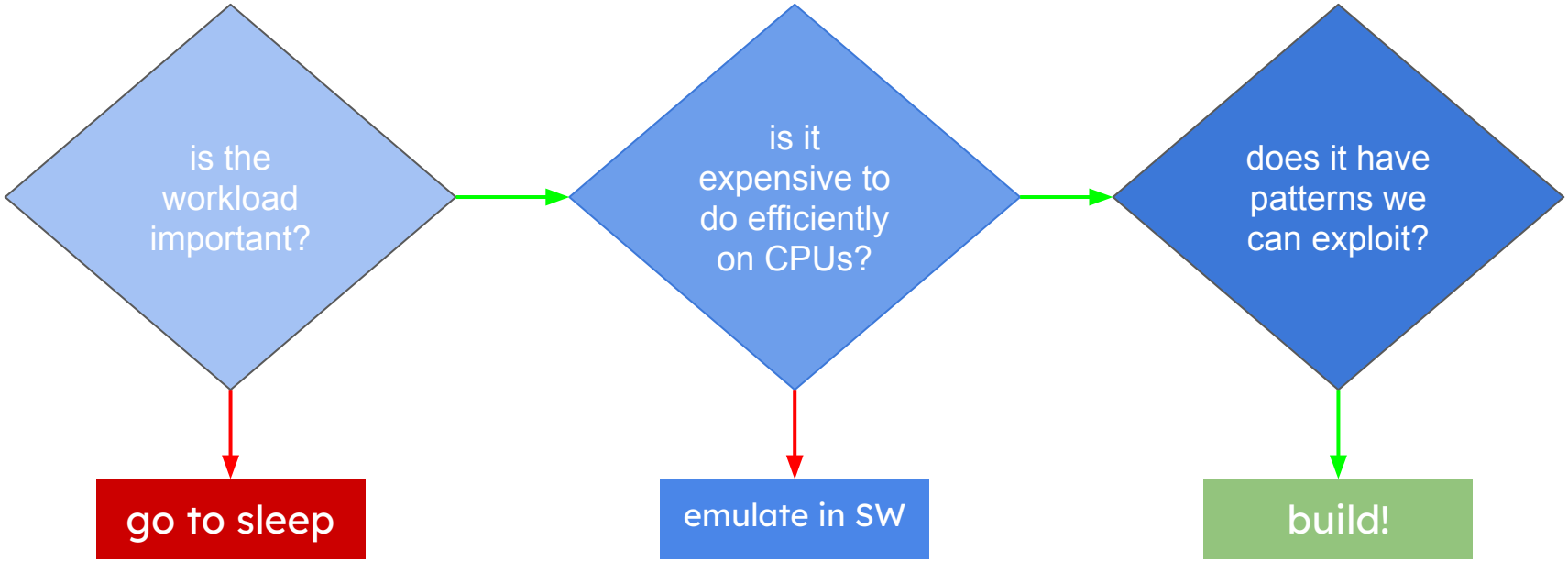
examples of “specialized” cores?

TPU (Tensor Processing Unit)

GPU (Graphics Processing Unit)

+ iPhone Neural Engine,
+ Pixel Mali GPU,
+ Snapdragon Hexagon NPU

when do you build a specialized core?

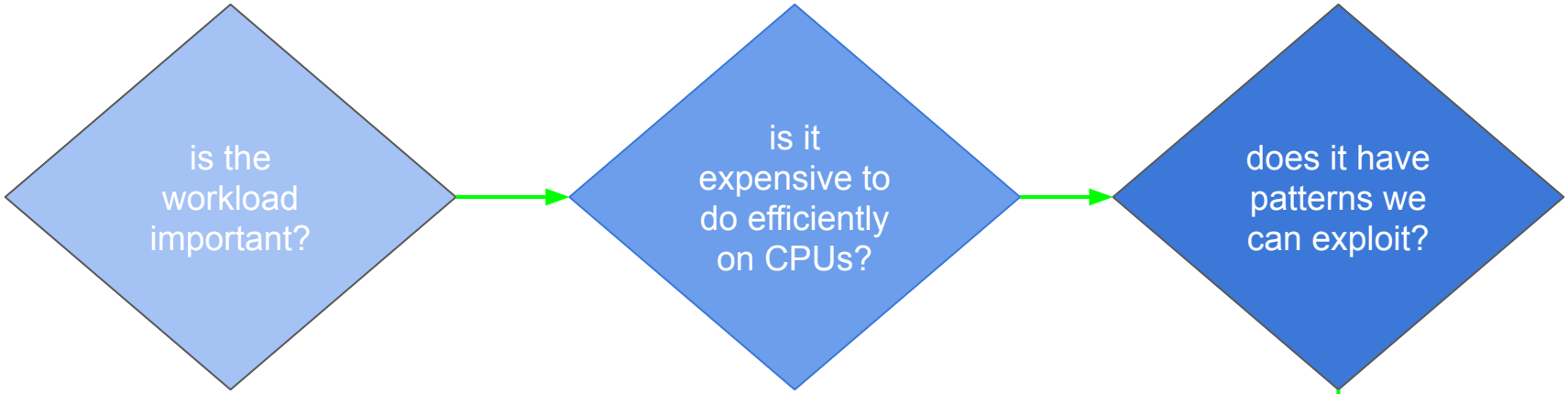


nerds build useless fun stuff;
engineers don't

why build TPUs (ML/AI)?

ML just FC/Conv layers;
Those layers are repeated Tensor math;

Regular compute/mem patterns!



AI is used everywhere;
- Robotics uses VLAs
- Google uses it for Search
- We use it for testbenches!

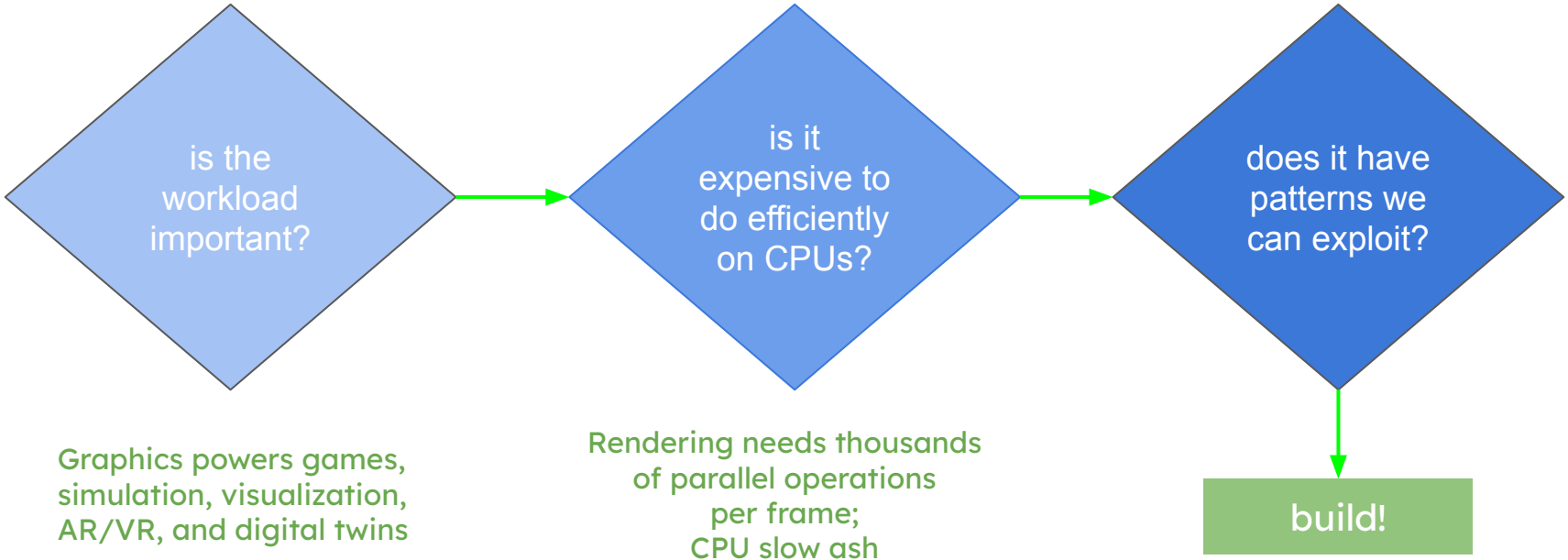
needs v low p99 latency though

Throughput can be maximized by (very) SIMD units

Too power hungry + area needed to add to CPU

why build GPUs (graphics)?

Vertices, pixels, shaders,
Textures repeat;
Independent math and
streaming data;



Graphics powers games,
simulation, visualization,
AR/VR, and digital twins

Rendering needs thousands
of parallel operations
per frame;
CPU slow ash

Nvidia & Google don’t open source “secret-sauce” – why?

VAGUELY Open Sourced!

programming model,
compiler interface,
performance guidance,
high-level arch

NOT EXPOSED!

Exact core microarchitecture,
Precise scheduler behavior,
Exact cache replacement,
Operand collector details,
cache PnR information,
etc.

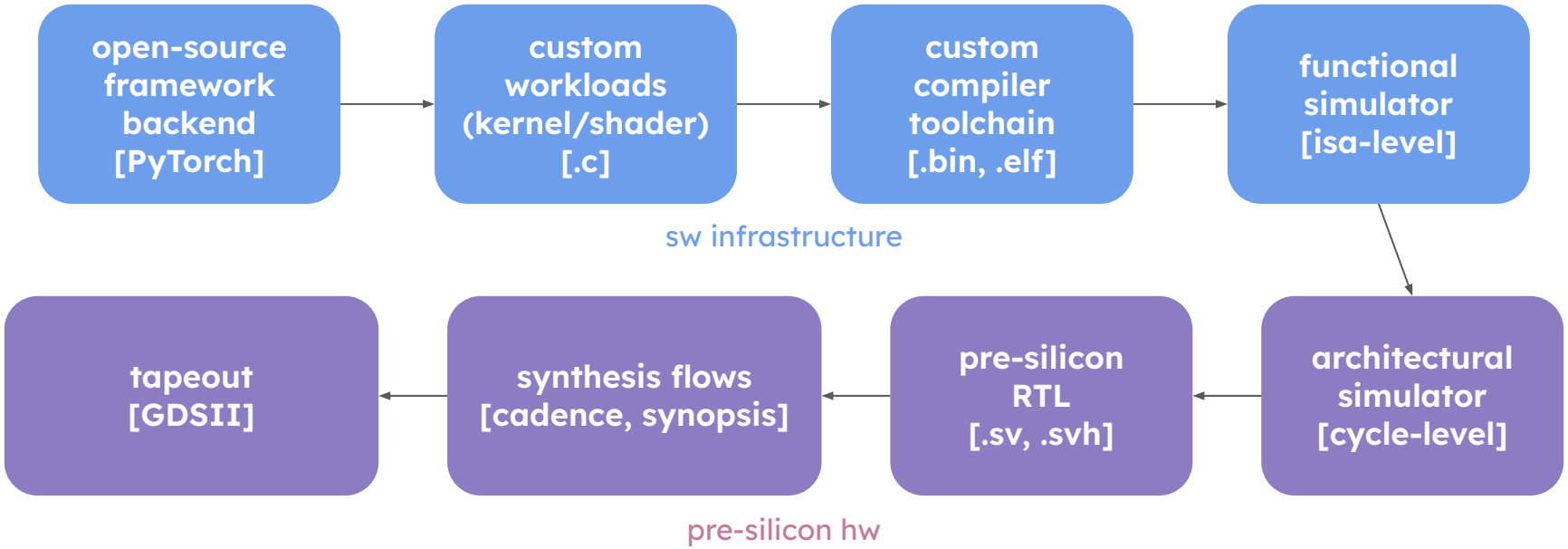
promote end-users	competitive advantage
define ecosystem/moat	security
create the engineering trends shaping the market	freedom to escape hardware backwards compatibility

and so,

- ❑ industry trends showed potential for DSAs
- ❑ AI & Graphics are perfect for custom silicon
- ❑ Best industrial systems are *closed off*
- ❑ DSA startups are everywhere (Taalas!)
- ❑ academic tape-outs and FOSS toolchains rare

that's where we come in!

Build open-source custom *architectures* and *toolchains;*
inspired by industrial systems, but designed for research



The goal is that anyone can understand the architecture, modify it, run workloads on it, and eventually help us tape it out!

motivation

history & growth

future plans

who should join?

many many weeks ago,

in the MSEE 180 boardroom,

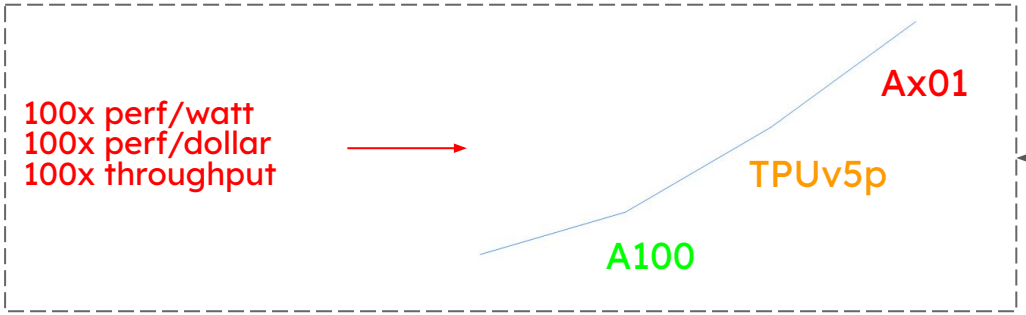
a new sub-team was formed!

aptly named as AI Hardware

Sooraj, Timmy and Malcolm were

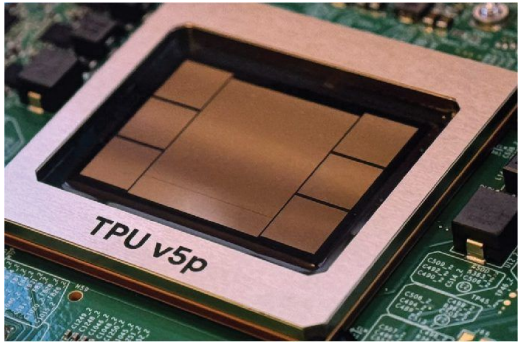
inspired by Nvidia and Google to

build a custom ASIC



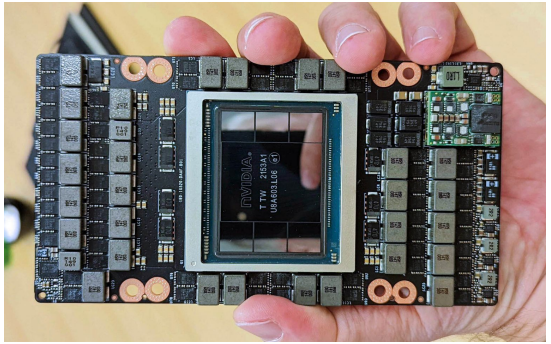
SoCET's Atalla Ax01!

>

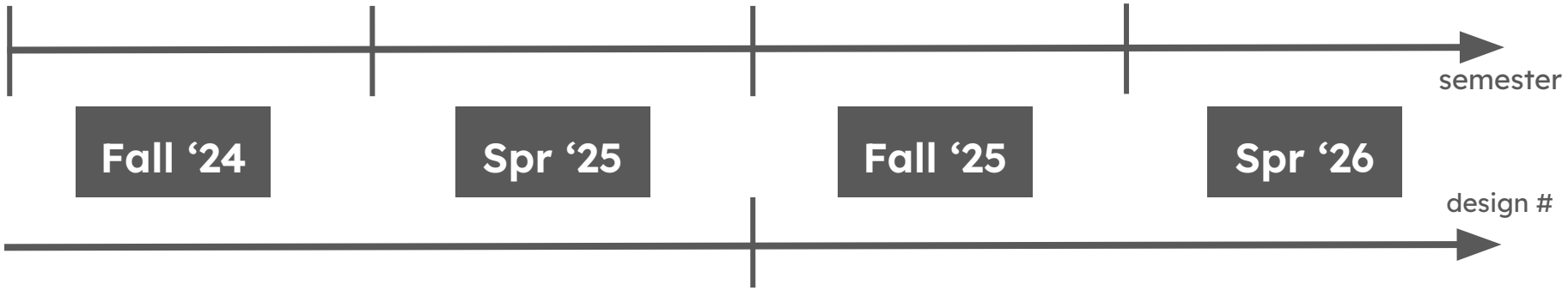


Google's TPUv5p

>

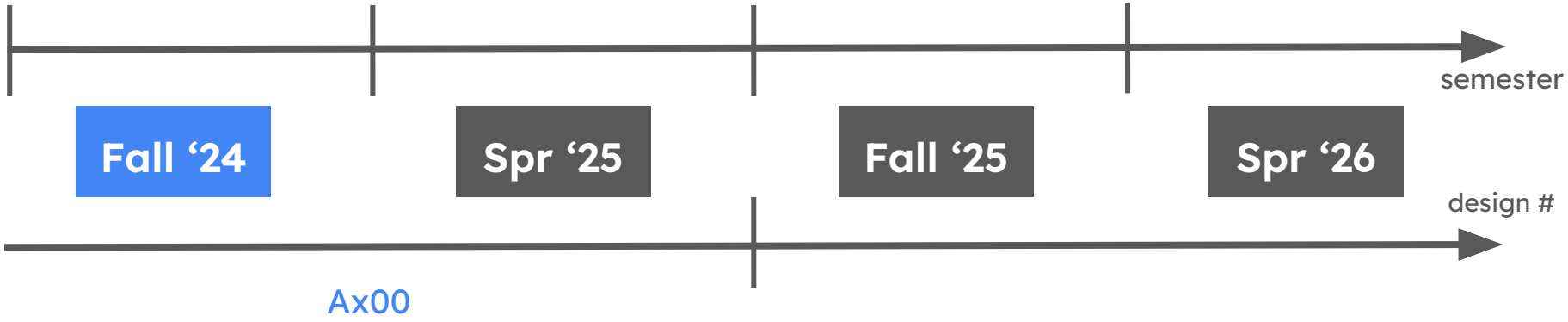


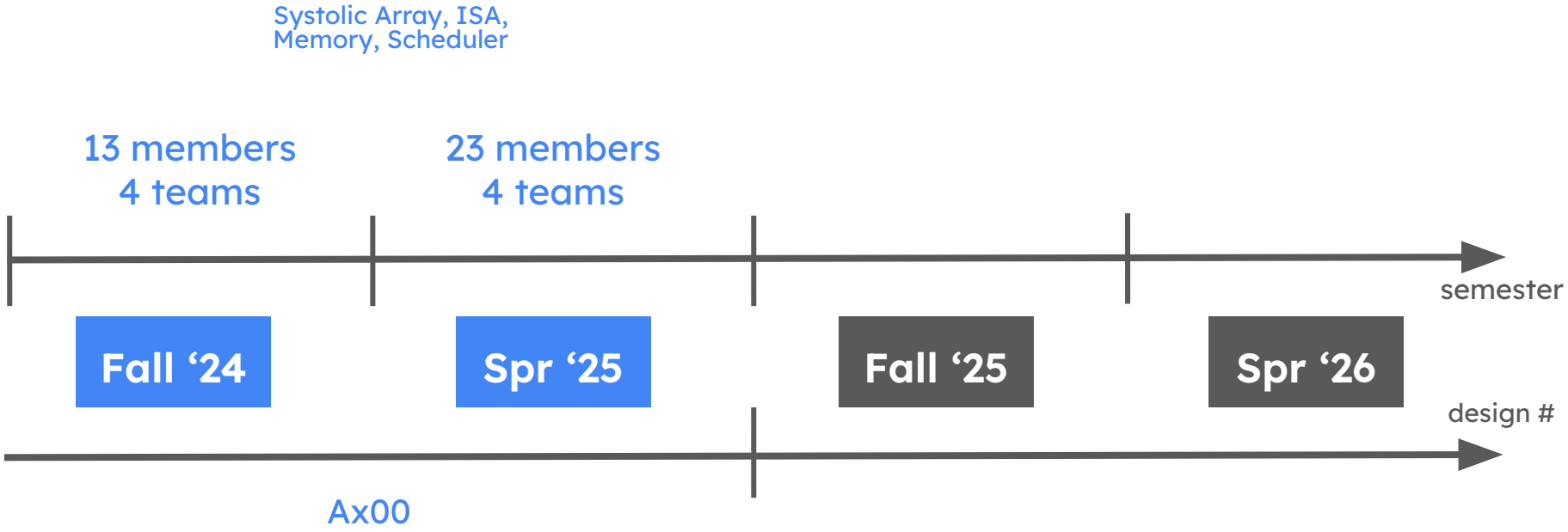
Nvidia's A100


semester
Fall ‘24
Spr ‘25
Fall ‘25
Spr ‘26
design #

Systolic Array, ISA,
Memory, Scheduler

13 members
4 teams

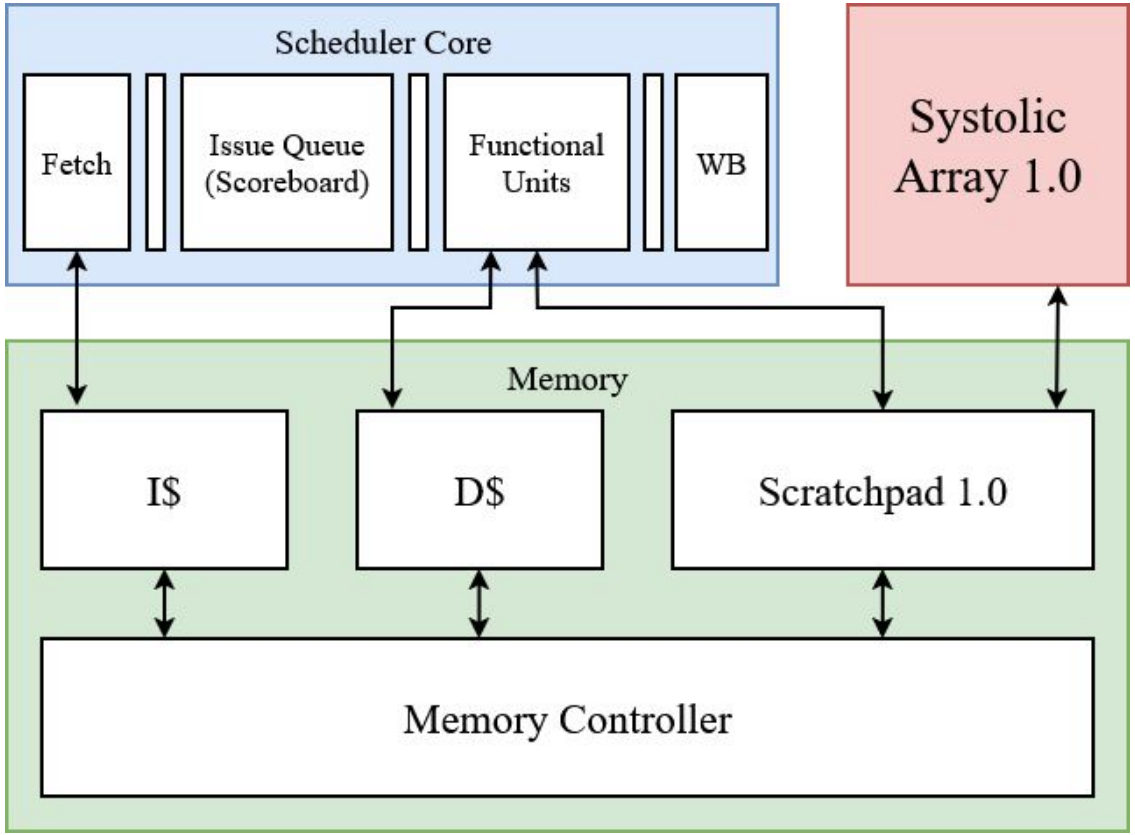



Systolic Array, ISA,
Memory, Scheduler
13 members
4 teams
23 members
4 teams
semester
Fall ‘24
Spr ‘25
Fall ‘25
Spr ‘26
design #
Ax00

scoreboard O3

blocking caches

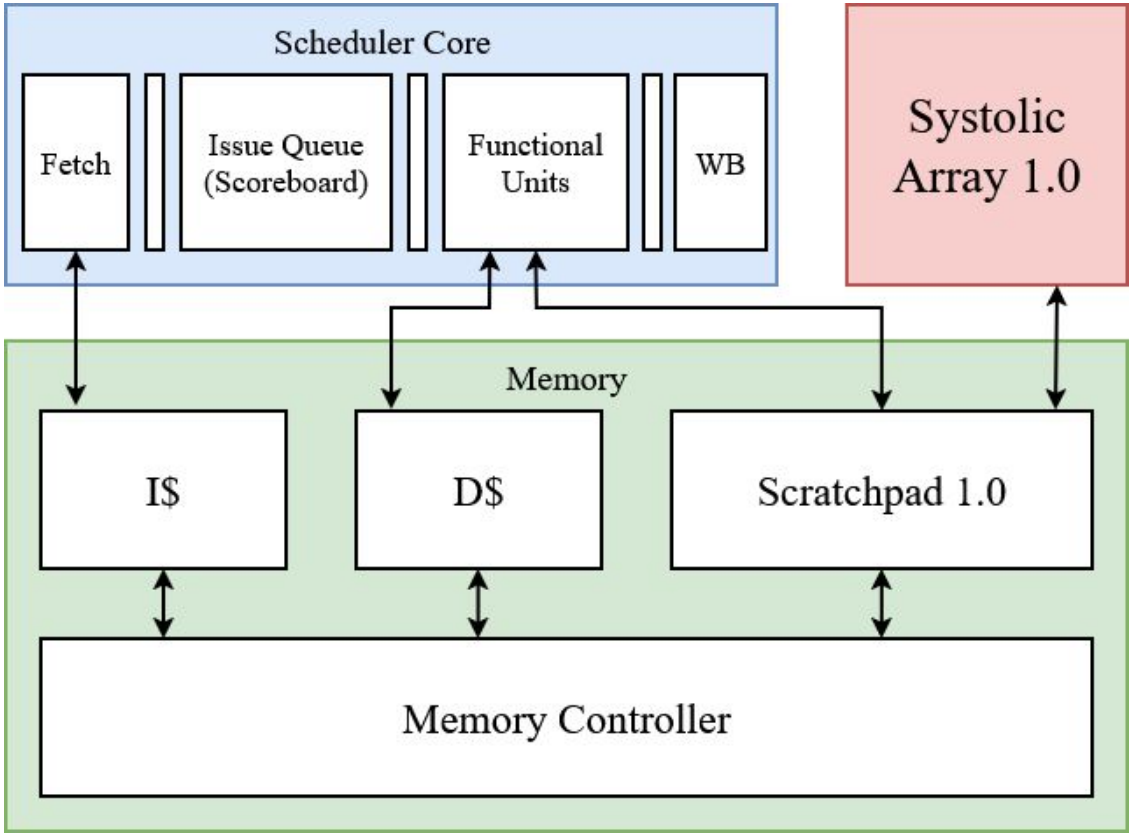
blocking ddr4-cntrl



32x32 SA

matrix registers

Atalla Ax00 Top-Level Architecture



limited throughput

blocking FSM, low bandwidth

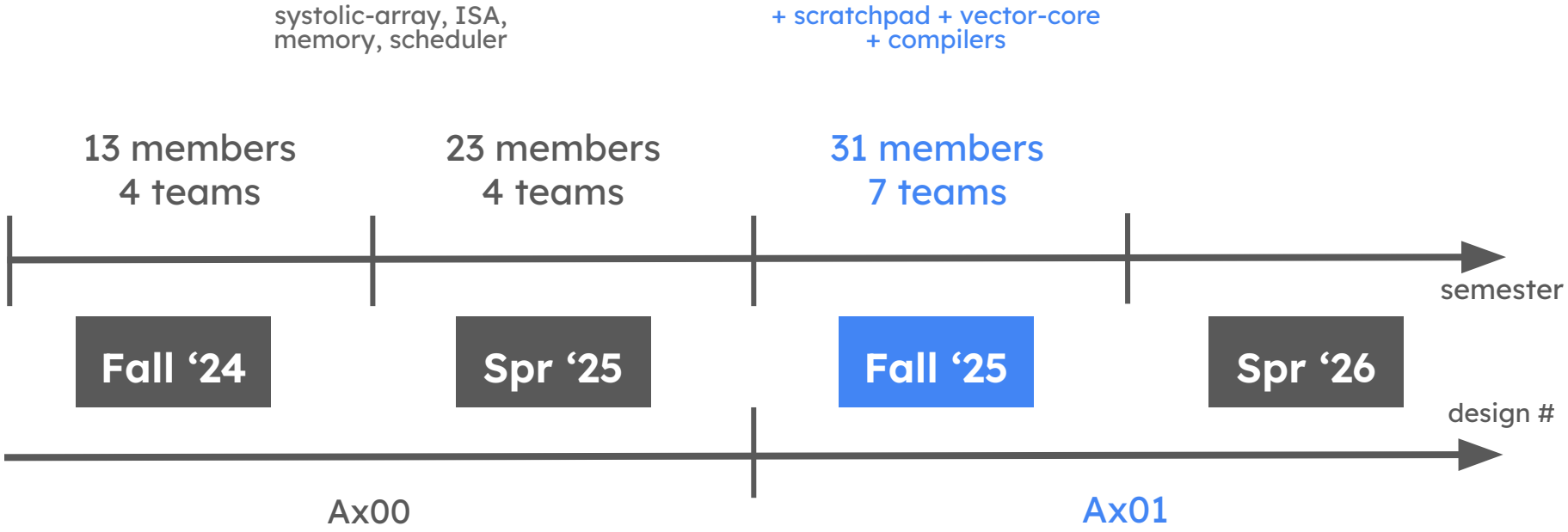
no memory parallelism

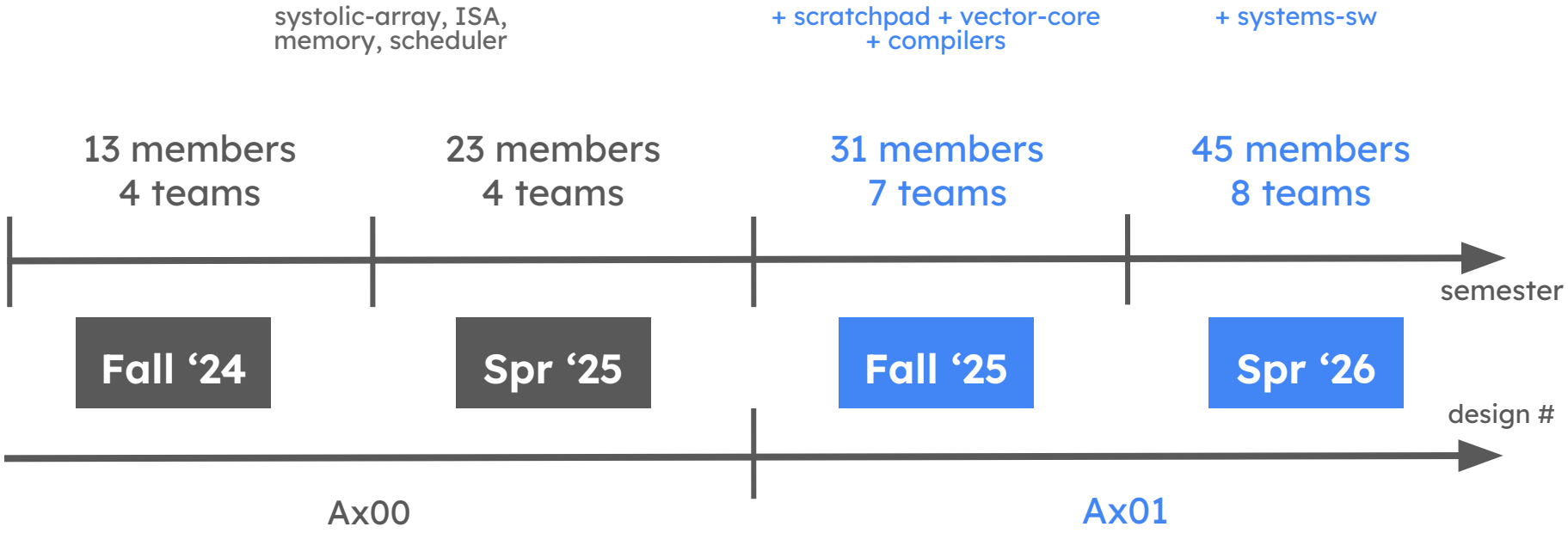
no split-transaction bus

no overlapping work

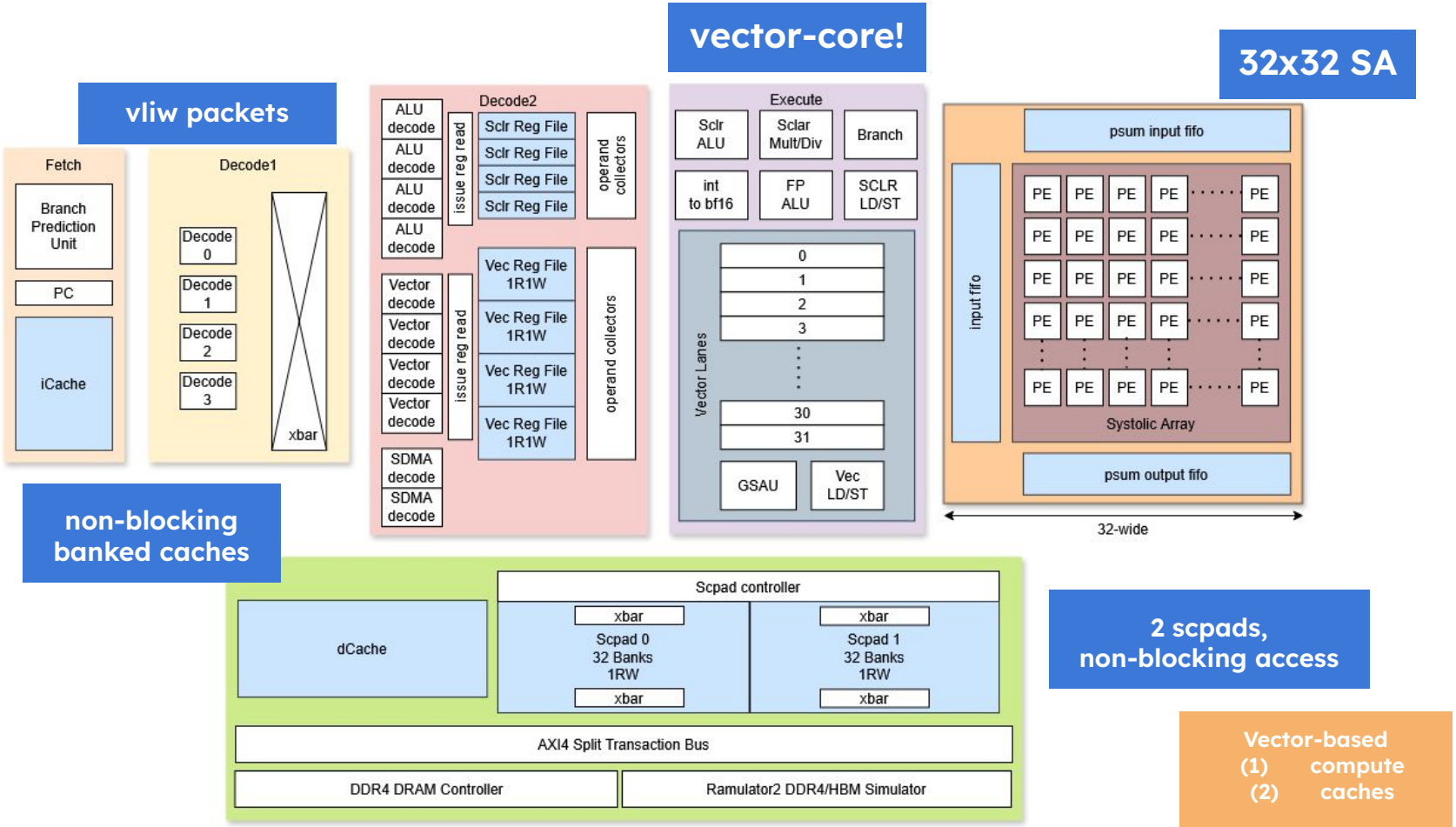
Tile-based
(1) compute
(2) caches
(3) off-chip ld/st

Atalla Ax00 Top-Level Architecture

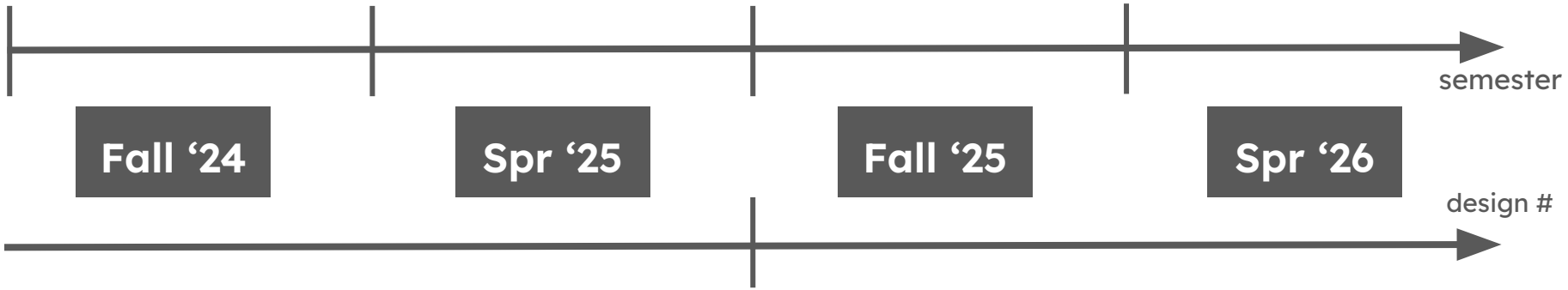

systolic-array, ISA,
memory, scheduler
+ scratchpad + vector-core
+ compilers
13 members
4 teams
23 members
4 teams
31 members
7 teams
semester
Fall ‘24
Spr ‘25
Fall ‘25
Spr ‘26
design #
Ax00
Ax01


systolic-array, ISA,
memory, scheduler
+ scratchpad + vector-core
+ compilers
+ systems-sw
13 members
4 teams
23 members
4 teams
31 members
7 teams
45 members
8 teams
semester
Fall ‘24
Spr ‘25
Fall ‘25
Spr ‘26
design #
Ax00
Ax01

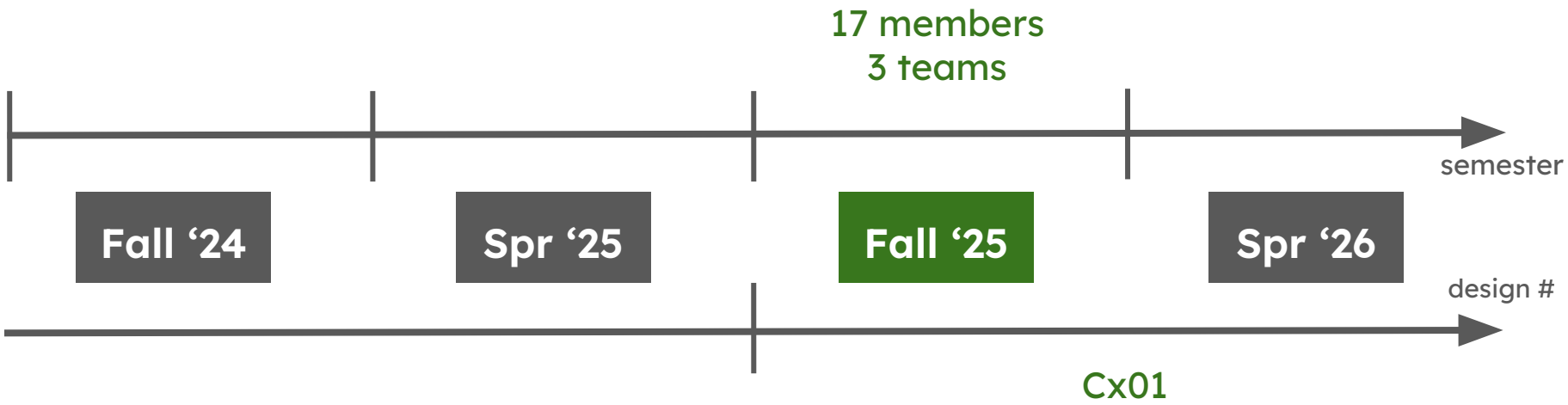

Purdue



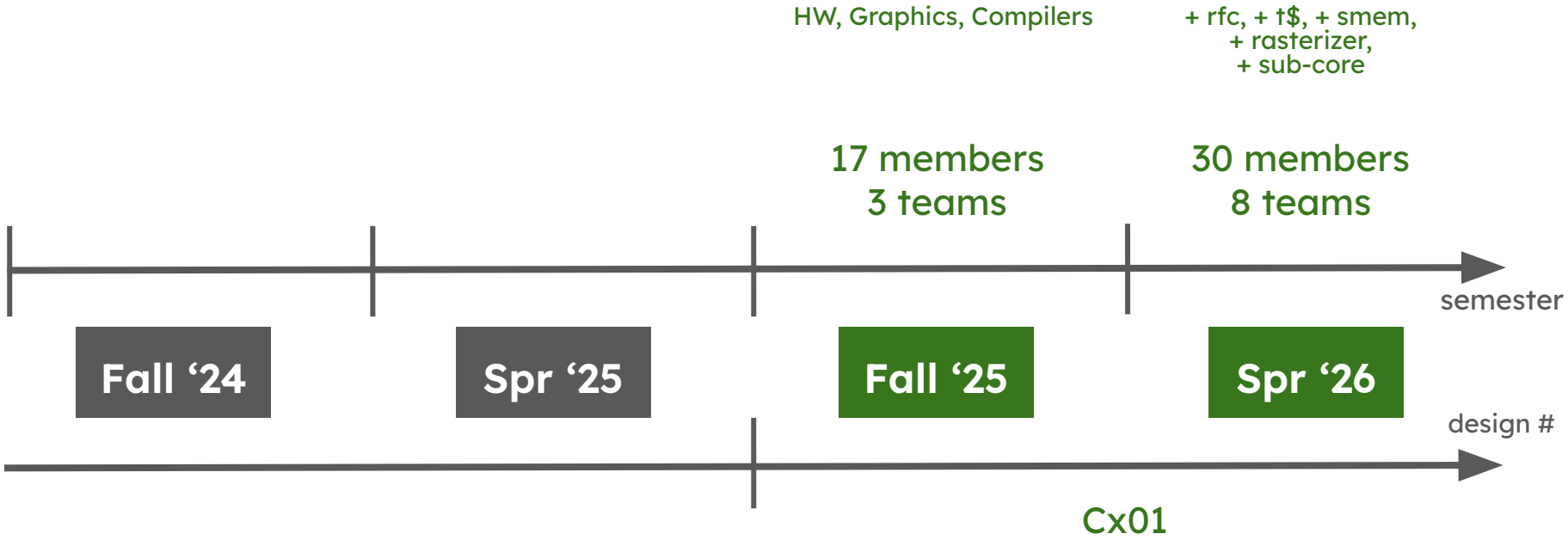
Atalla Ax01 Top-Level Architecture


semester
Fall ‘24
Spr ‘25
Fall ‘25
Spr ‘26
design #

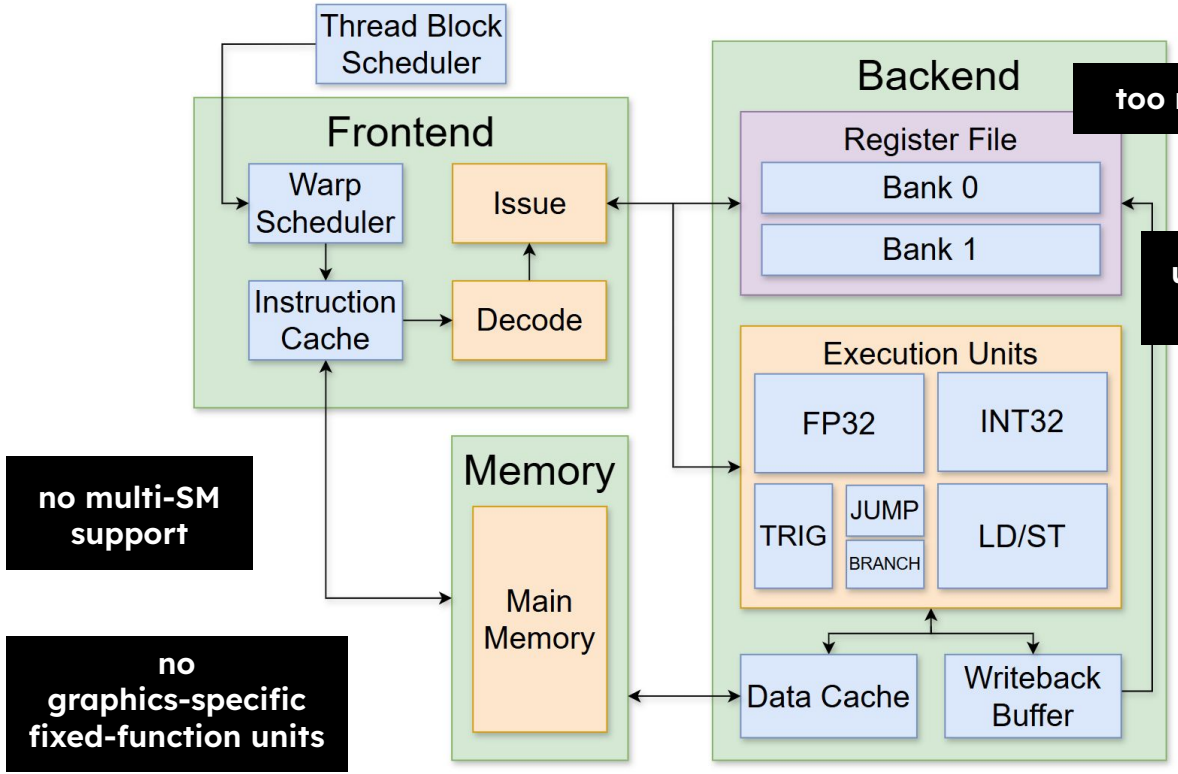
HW, Graphics, Compilers



GPU


HW, Graphics, Compilers
+ rfc, + t$, + smem,
+ rasterizer,
+ sub-core
17 members
3 teams
30 members
8 teams
semester
Fall ‘24
Spr ‘25
Fall ‘25
Spr ‘26
design #
Cx01

GPU



Cardinal Cx01 Top-Level Architecture

Credits: GPU Hardware Cx01 Sim group!

motivation

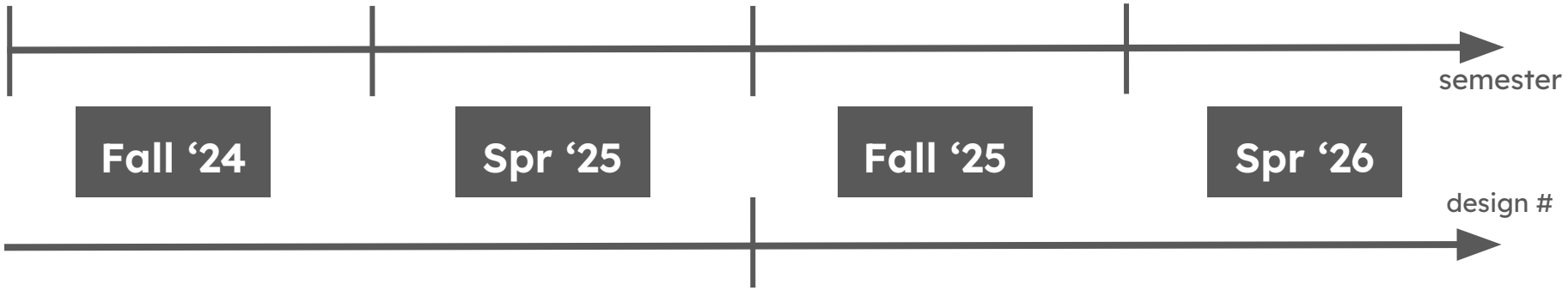
history & growth

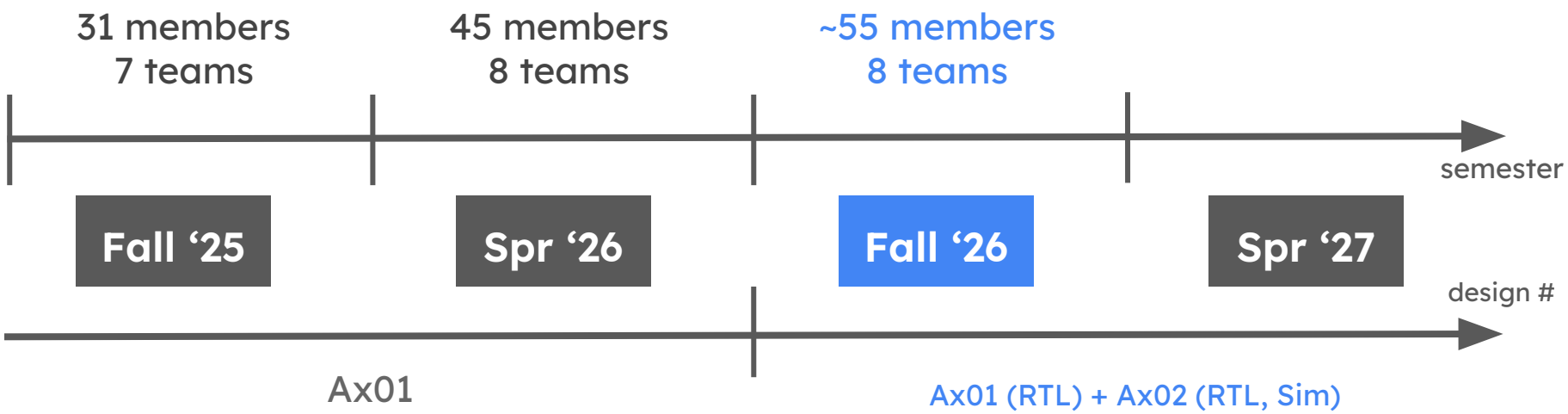
future plans

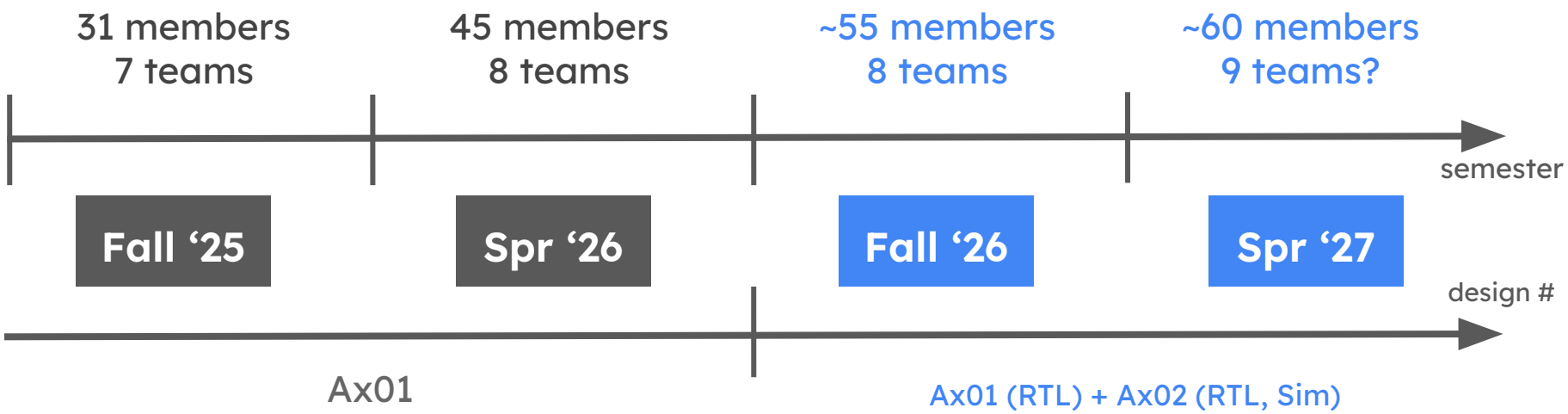
who should join?

lessons learned:

- ❏ architecture changes first in functional sim
- ❏ RTL needs minimum 98% code coverage
- ❏ every arch edit needs compiler and sw efforts (hw/sw co-design)
- ❏ physical design imperative in design cycle
- ❏ communication skills most important


semester
Fall ‘24
Spr ‘25
Fall ‘25
Spr ‘26
design #


31 members
7 teams
45 members
8 teams
~55 members
8 teams
semester
Fall ‘25
Spr ‘26
Fall ‘26
Spr ‘27
design #
Ax01
Ax01 (RTL) + Ax02 (RTL, Sim)

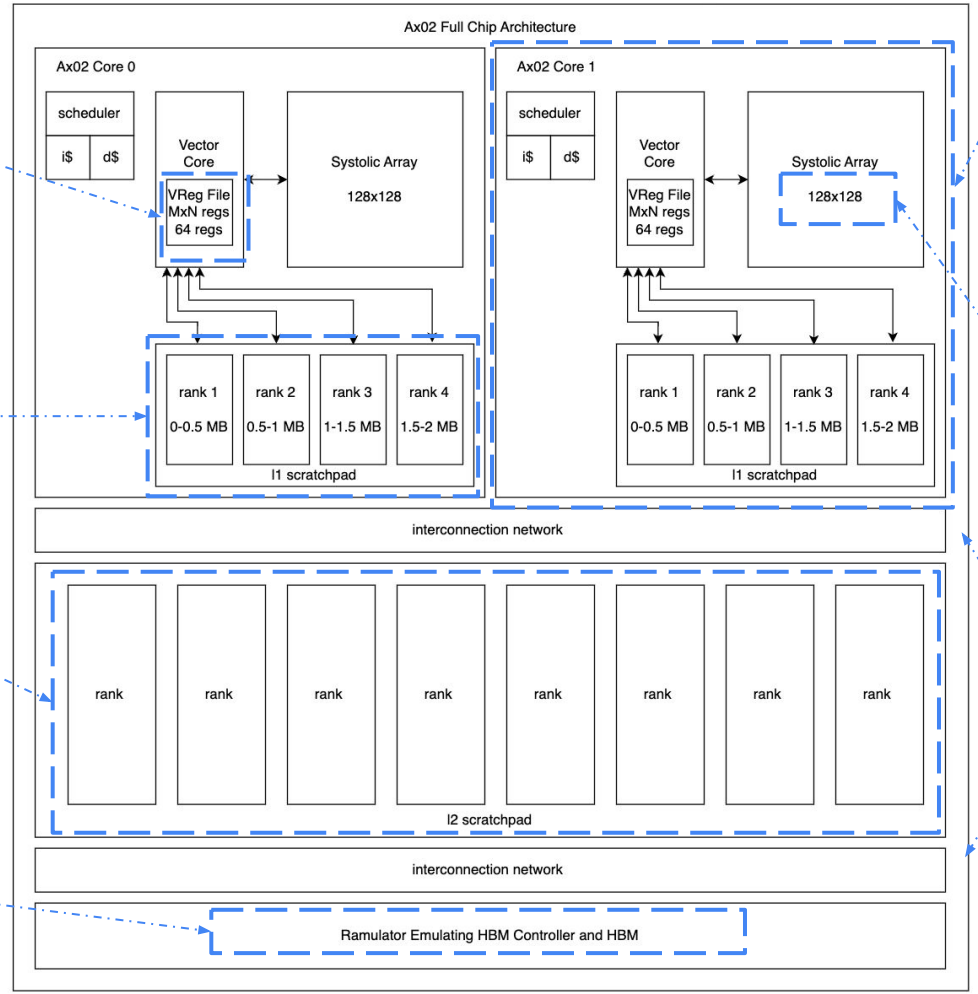

31 members
7 teams
45 members
8 teams
~55 members
8 teams
~60 members
9 teams?
semester
Fall ‘25
Spr ‘26
Fall ‘26
Spr ‘27
design #
Ax01
Ax01 (RTL) + Ax02 (RTL, Sim)

more DLP with MxN reg

more b/w with 4-split

l2 for cross-core sync

can finally utilize the real width of HBM

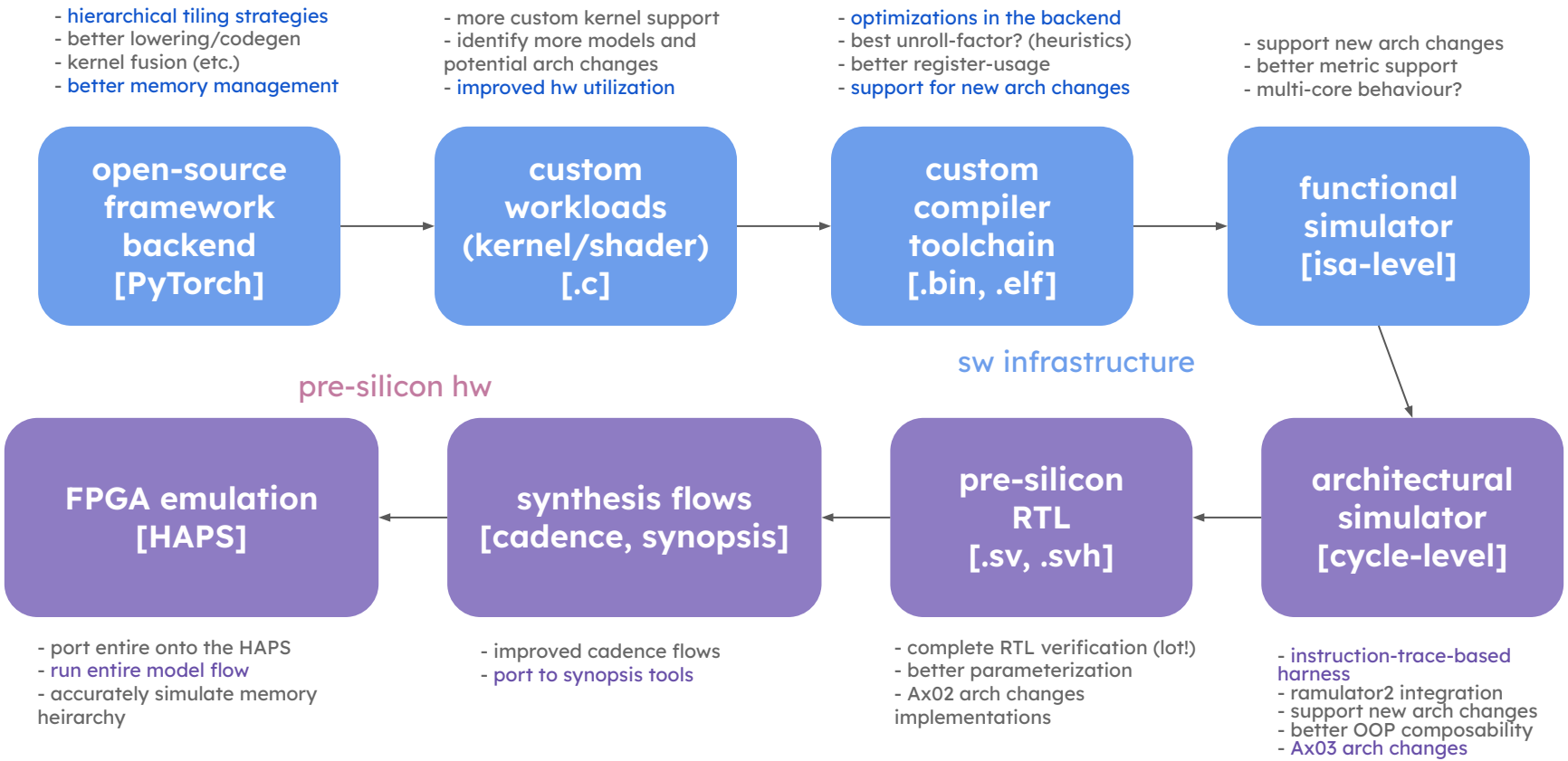


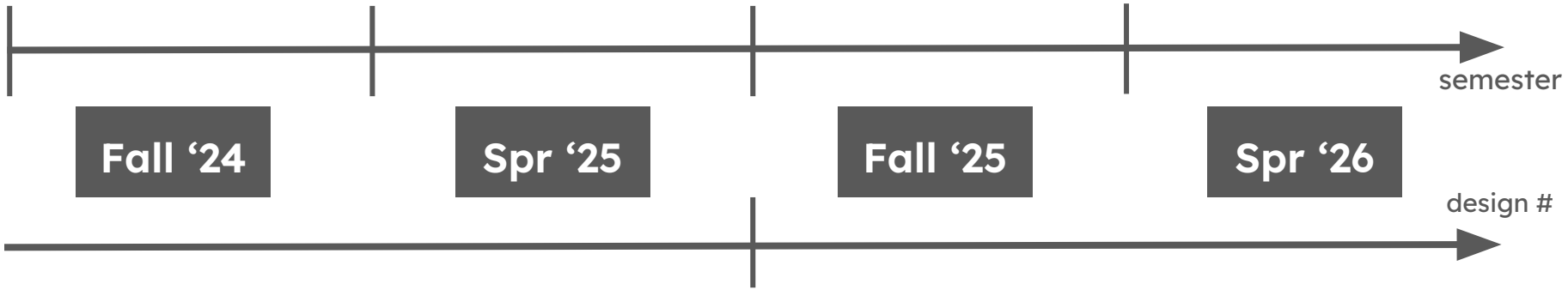
dual-core chip

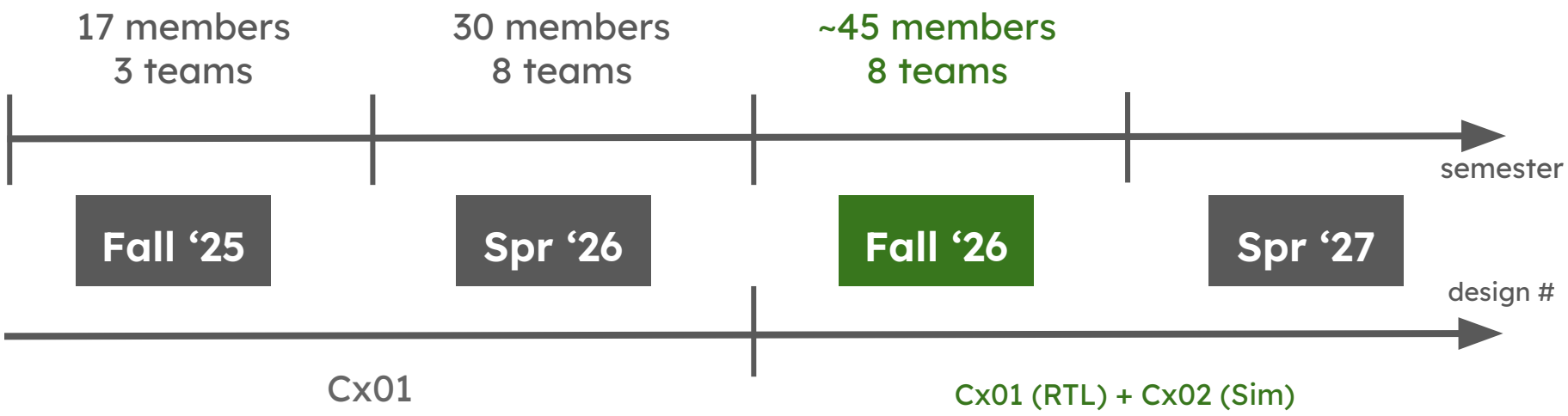
16x larger SA

two buses

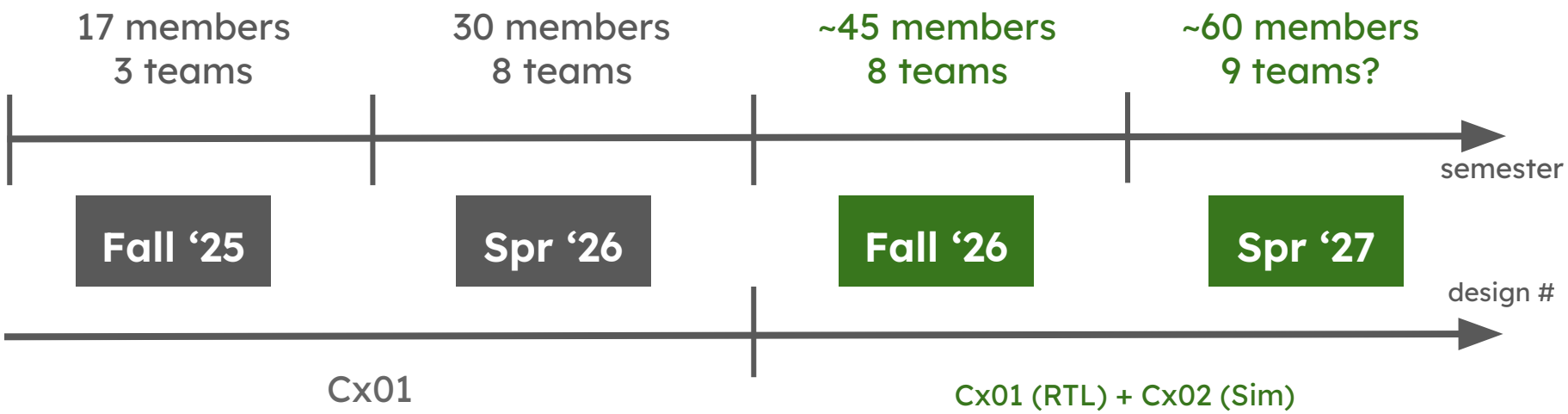
Atalla Ax02 Top-Level Architecture


- hierarchical tiling strategies
- better lowering/codegen
- kernel fusion (etc.)
- better memory management
- more custom kernel support
- identify more models and potential arch changes
- improved hw utilization
- optimizations in the backend
- best unroll-factor? (heuristics)
- better register-usage
- support for new arch changes
- support new arch changes
- better metric support
- multi-core behaviour?
open-source framework backend [PyTorch]
custom workloads (kernel/shader) [.c]
custom compiler toolchain [.bin, .elf]
functional simulator [isa-level]
sw infrastructure
pre-silicon hw
FPGA emulation [HAPS]
synthesis flows [cadence, synopsis]
pre-silicon RTL [.sv, .svh]
architectural simulator [cycle-level]
- port entire onto the HAPS
- run entire model flow
- accurately simulate memory heirarchy
- improved cadence flows
- port to synopsis tools
- complete RTL verification (lot!)
- better parameterization
- Ax02 arch changes implementations
- instruction-trace-based harness
- ramulator2 integration
- support new arch changes
- better OOP composability
- Ax03 arch changes

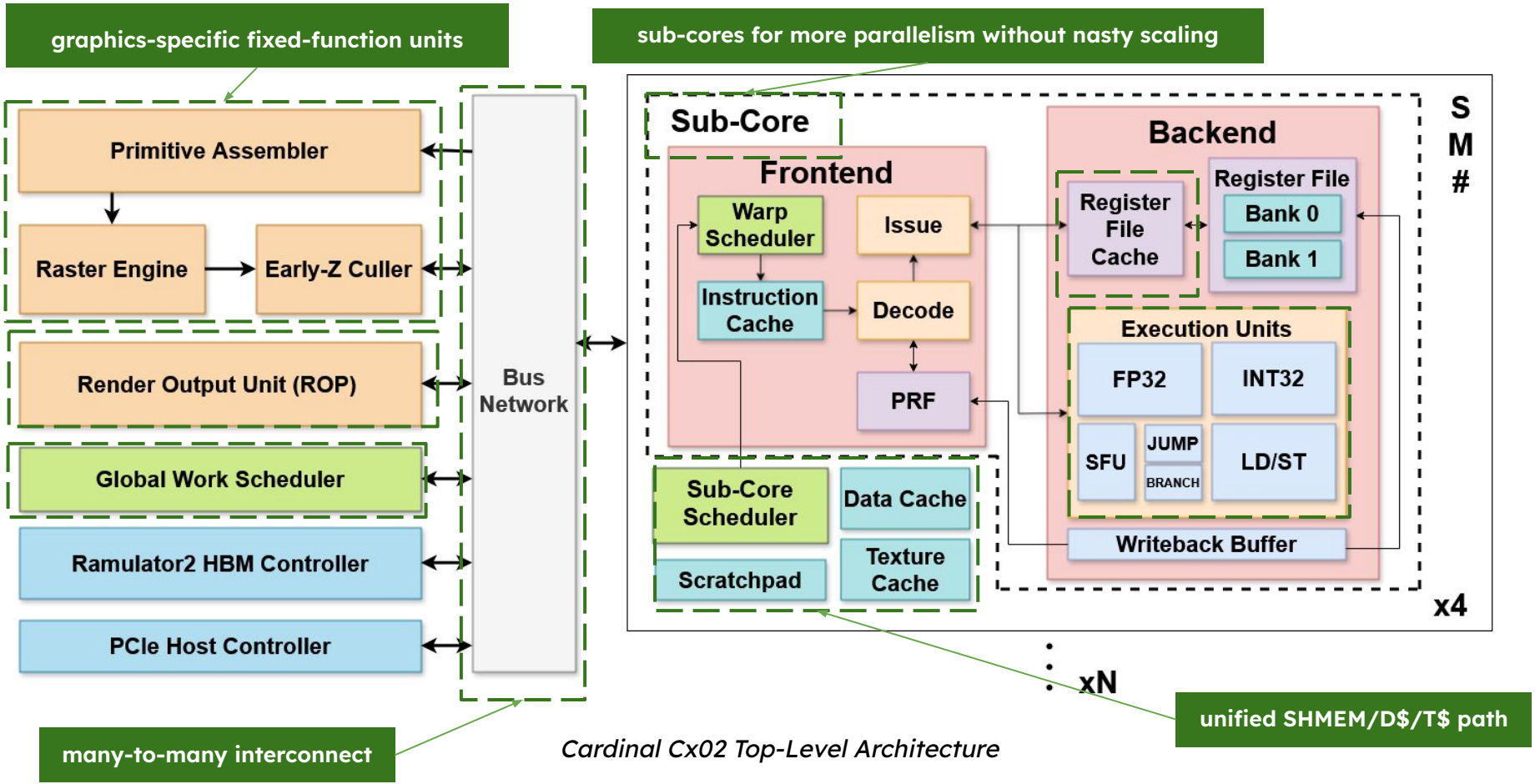

semester
Fall ‘24
Spr ‘25
Fall ‘25
Spr ‘26
design #


17 members
3 teams
30 members
8 teams
~45 members
8 teams
semester
Fall ‘25
Spr ‘26
Fall ‘26
Spr ‘27
design #
Cx01
Cx01 (RTL) + Cx02 (Sim)

GPU

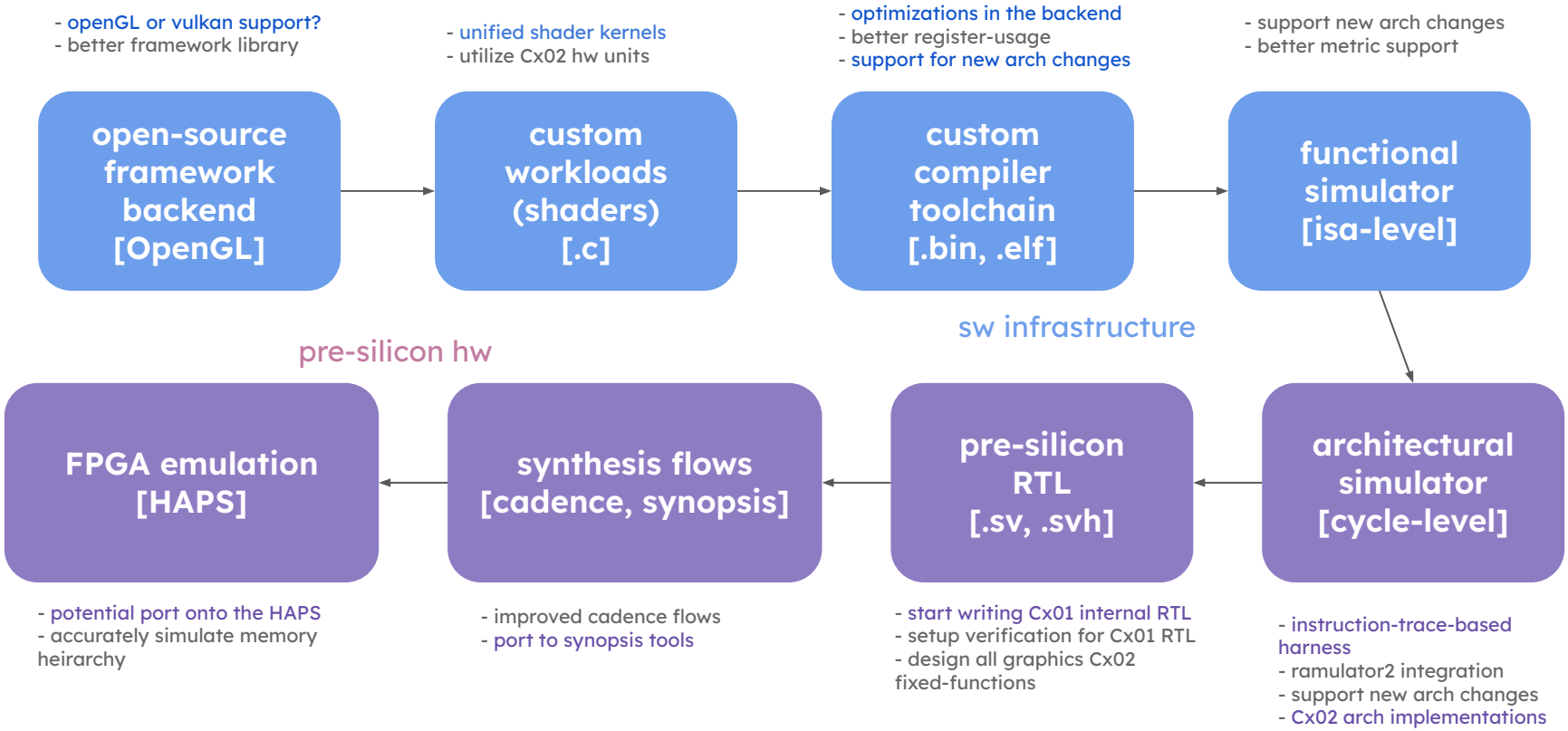

17 members
3 teams
30 members
8 teams
~45 members
8 teams
~60 members
9 teams?
semester
Fall ‘25
Spr ‘26
Fall ‘26
Spr ‘27
design #
Cx01
Cx01 (RTL) + Cx02 (Sim)

GPU



Cardinal Cx02 Top-Level Architecture

Inspired by Spencer + Syd's top-level


- openGL or vulkan support?
- better framework library
open-source framework backend [OpenGL]
- unified shader kernels
- utilize Cx02 hw units
custom workloads (shaders) [.c]
- optimizations in the backend
- better register-usage
- support for new arch changes
custom compiler toolchain [.bin, .elf]
- support new arch changes
- better metric support
functional simulator [isa-level]
sw infrastructure
architectural simulator [cycle-level]
- instruction-trace-based harness
- ramulator2 integration
- support new arch changes
- Cx02 arch implementations
pre-silicon RTL [.sv, .svh]
- start writing Cx01 internal RTL
- setup verification for Cx01 RTL
- design all graphics Cx02 fixed-functions
pre-silicon hw
synthesis flows [cadence, synopsis]
- improved cadence flows
- port to synopsis tools
FPGA emulation [HAPS]
- potential port onto the HAPS
- accurately simulate memory heirarchy

GPU

motivation

history & growth

future plans

who should join?

what you will gain!

communication skills
- interviews/internships/fte
- teamwork
- how to present

technical skills
- RTL design + verification
- computer architecture
- software infrastructure

portfolio-building
- real results to showcase
- shows awareness of latest trends
- improves credibility

mentorship
- how to think
- how to identify opportunities
- create your own projects

apply to join, if you are:

- ❑ excited about *comp.arch* & *software*
- ❑ willing to work across disciplines
- ❑ want to understand chip design flow
- ❑ capable of being consistent (!)
- ❑ good at communicating with others
- ❑ quick to onboard

DON’T apply to join, if you are:

- ❑ low motivation
- ❑ not willing to learn-by-doing
- ❑ have busy semesters
- ❑ not doing senior-design with us
- ❑ not wanting to take ownership

who are we looking for? (any)

- ❑ good at compilers (ppci, llvm exp.)
- ❑ wonderful at RTL & verification
- ❑ capable in graphics/ml workloads
- ❑ proficient in python/cpp (for sim)
- ❑ background in physical design
- ❑ fpga emulation (HAPS pref.) experience
- ❑ pytorch or openGL/vulkan background
- ❑ coursework in compiler/arch/DL/graphics

questions?