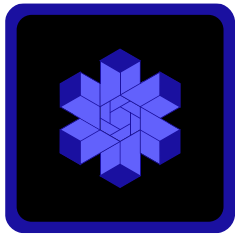
semidynamics

Tailor-Made High Performance RISCV64 IP

In Order Core

OOO Core



OOO Vector Unit

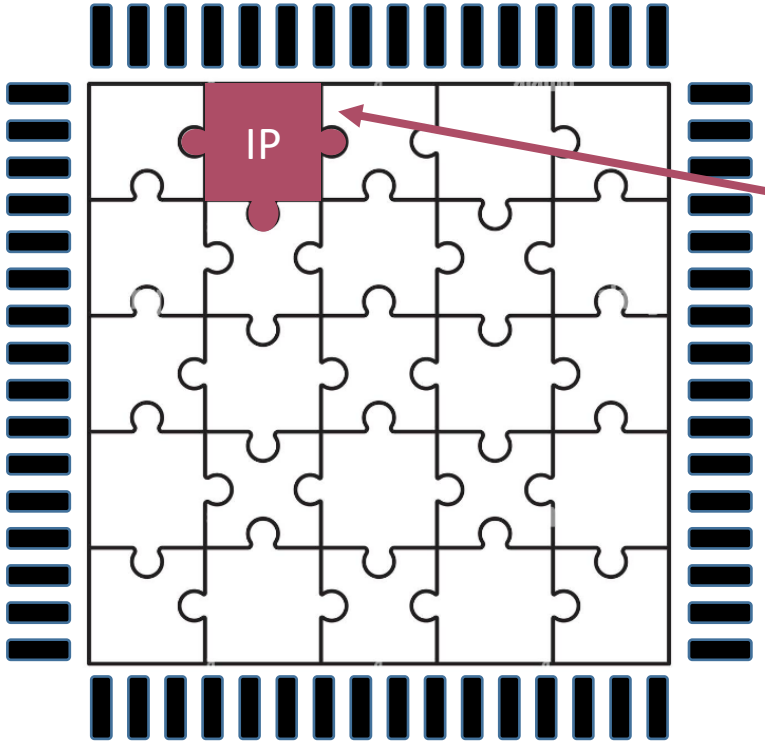
About Semidynamics

Semidynamics, founded in 2016, is a **100% European** supplier of RISC-V IP cores, HQ in **Barcelona**, specializing in **customization** of **high bandwidth high performance cores with vector units** for **tailored projects**

Experts in open core surgery

semidynamics

What do we do? What do we sell?

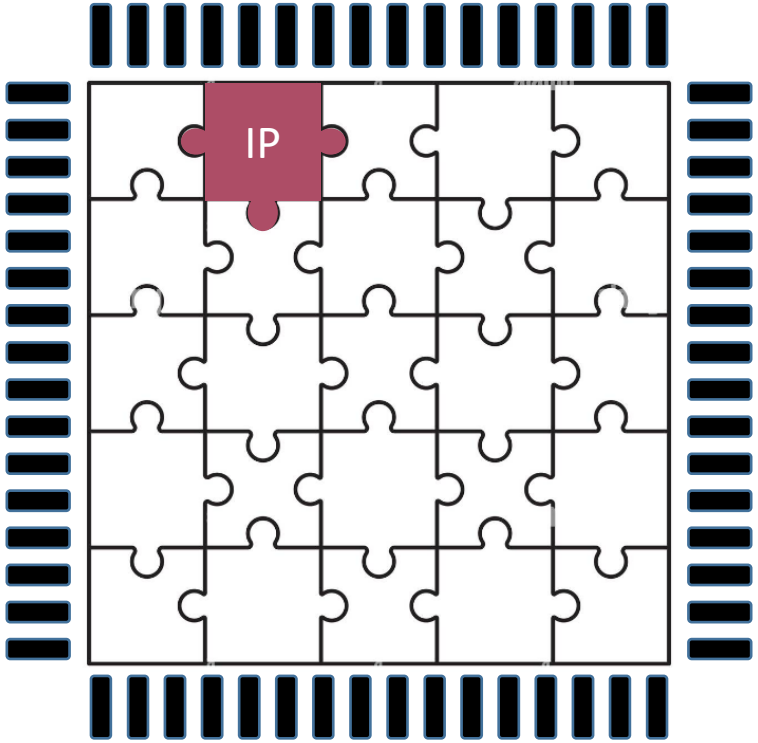


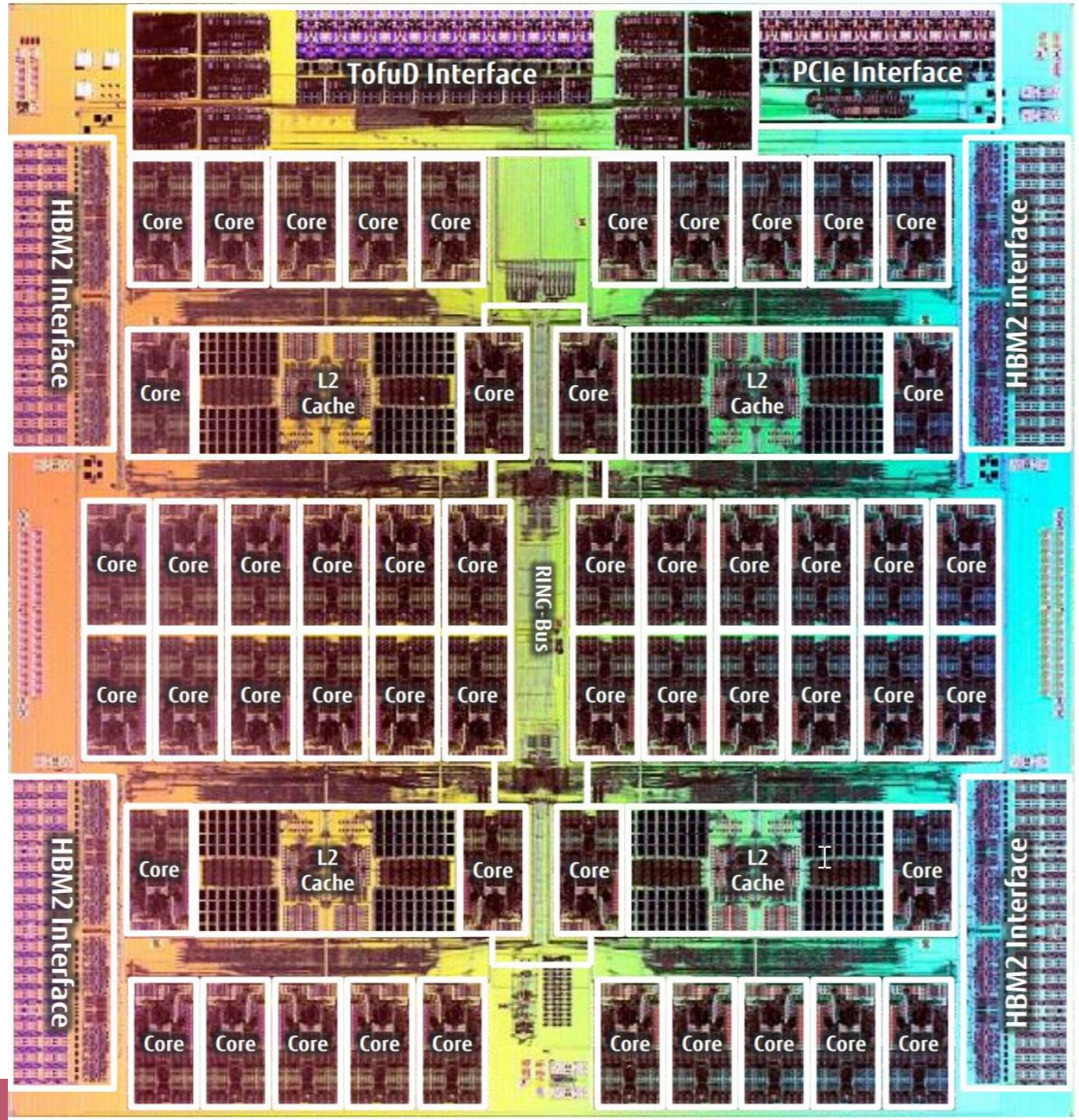
semidynamics

The BLUEPRINT of a portion of a silicon chip

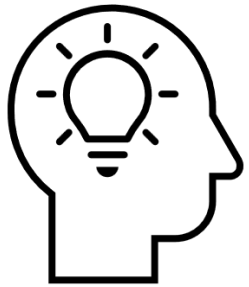
semidynamics

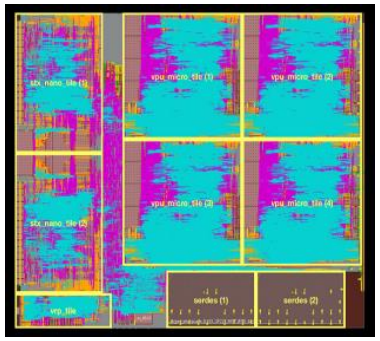
Example:
FUJITSU A64FX (2019)

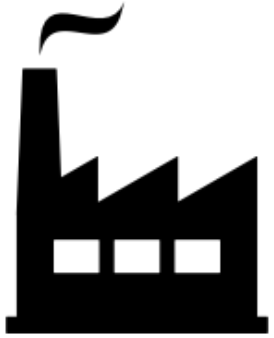


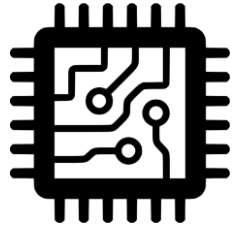


Designing a chip...

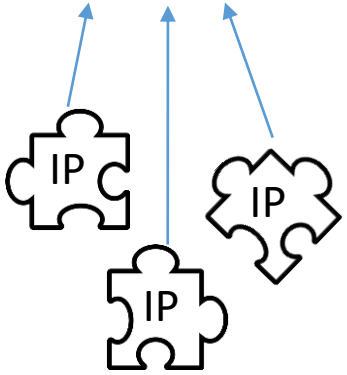






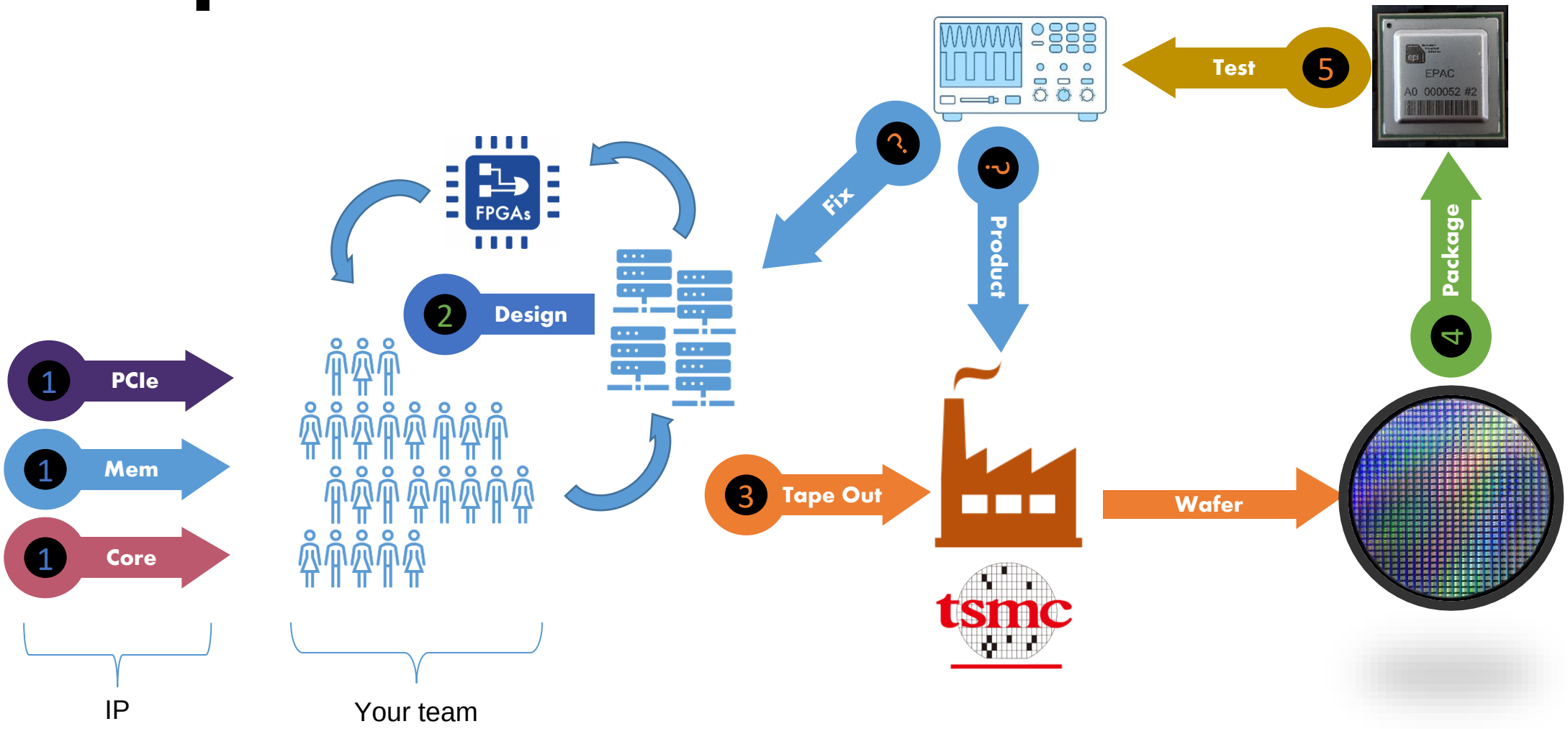


Idea → Exploration → R&D(Frontend) → R&D(Backend) → Foundry → Chip



semidynamics

Chip Fabrication Process



Our RISC-V Core IP Families

Atrevido

无所畏惧 Fearless 용감한

2, 3 or 4-wide **out-of-order**
RISCV64GCV AXI and CHI

Avispado

机智 Smart 똑똑한

2-wide **in-order**
RISCV64GCV AXI and CHI

World's first, **fully customizable**, 64-bit RISC-V cores for ultra fast, big memory applications, optimized for a companion RISC-V **vector unit**

Unique tailor-made PPA solutions include customer's secret sauce for product differentiation and IP protection.

semidynamics

What's special about our RISC-V Cores?

We fully customize each core to the customer's precise application needs

We can include unique customer features in a few weeks

Support for OOO RISC-V Vector Unit

Gazzillion™ turbo-charges memory retrieval

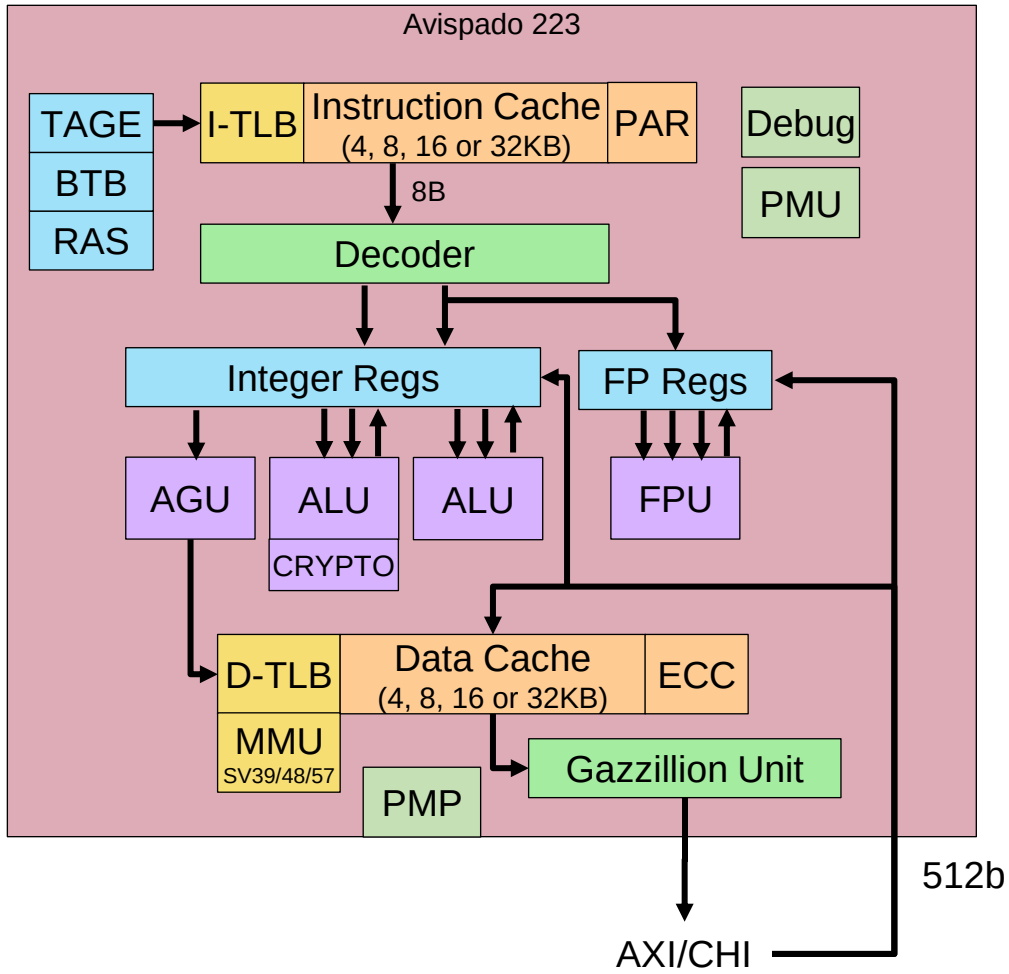
Fastest cores on the market for moving big data

Process agnostic — already done 5nm

semidynamics

Avispado 223

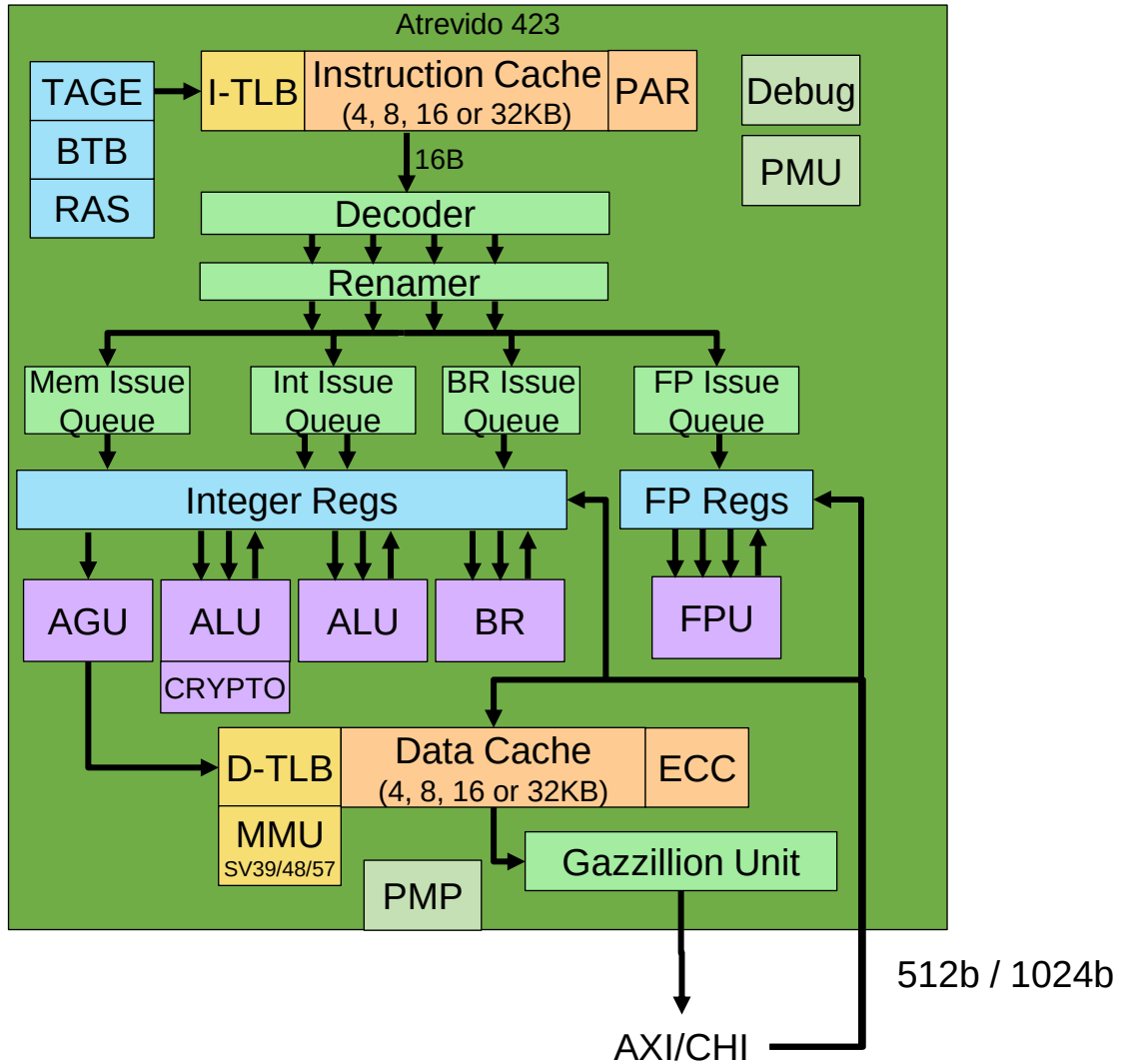
- 64b RISC-V Core
- Mid-Range Performance
- **In-order execution**
- User, Supervisor and Machine privilege levels
- Hypervisor support available Q4
- Linux-Ready memory subsystem
  - Memory Management Unit (MMU)
  - Supports SV39/48/57
  - Coherent caches with ECC, Parity
  - Hardware support for unaligned accesses
  - Hardware support for Atomics
- PMP Regions (0 to 16)
- AXI4 or AMBA CHI.B compliant interface
- Advanced Debug Capabilities
  - RISC-V debug spec compliant interface over JTAG
  - HW/SW Breakpoint support
- RISC-V Extensions supported:
  - **Vector, Crypto, Bit Manipulation, CMOs, Zifencei**
- Quad-Core Ready



semidynamics

Atrevido 423

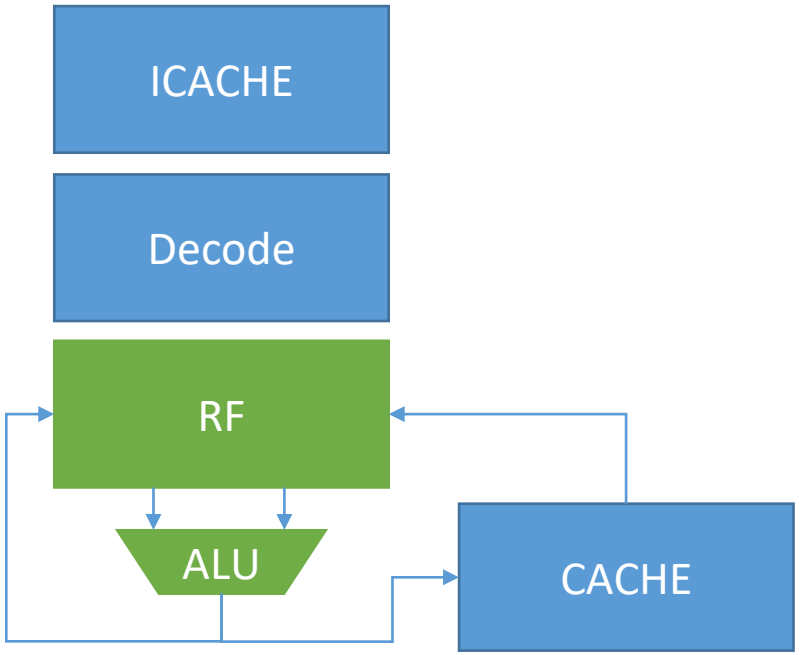
- 64b RISC-V Core
- High-End Performance
- **Out-of-order** execution
- **4-wide decode, rename, retire**
- User, Supervisor and Machine privilege levels
- Hypervisor available Q4
- Linux-Ready memory subsystem
  - Memory Management Unit (MMU)
  - Supports SV39/48/57
  - Coherent caches with ECC, Parity
  - Hardware support for unaligned accesses
  - Hardware support for Atomics
- PMP Regions (0 to 16)
- AXI4 or AMBA CHI.B compliant interface
- Advanced Debug Capabilities
  - RISC-V debug spec compliant interface over JTAG
  - HW/SW Breakpoint support
- RISC-V Extensions supported:
  - **Vector, Crypto, Bit Manipulation, CMOs, Zifencei**
- Quad-Core Ready



RISC-V vectors: Why?

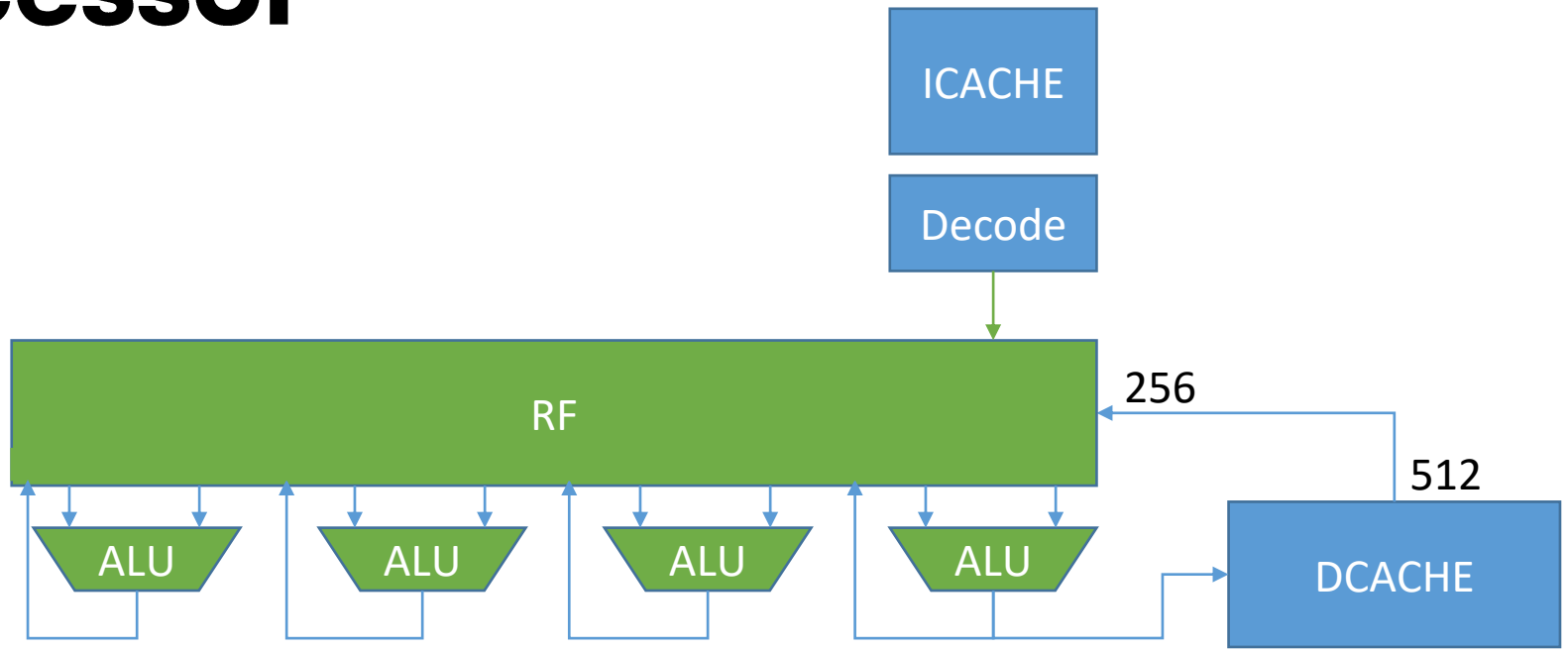
A scalar processor

- Load (r1) -> r2
- Load 4(r1) -> r3
- Add r2, r3 -> r4
- Store r4 -> (r1)

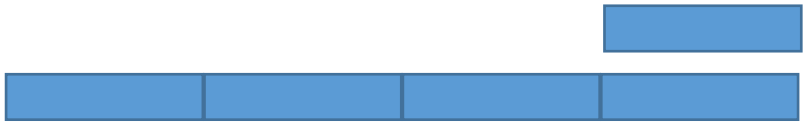


A vector processor

- vLoad (r1) -> v2
- vLoad 4(r1) -> v3
- vAdd v2, v3 -> v4
- vStore v4 -> (r1)

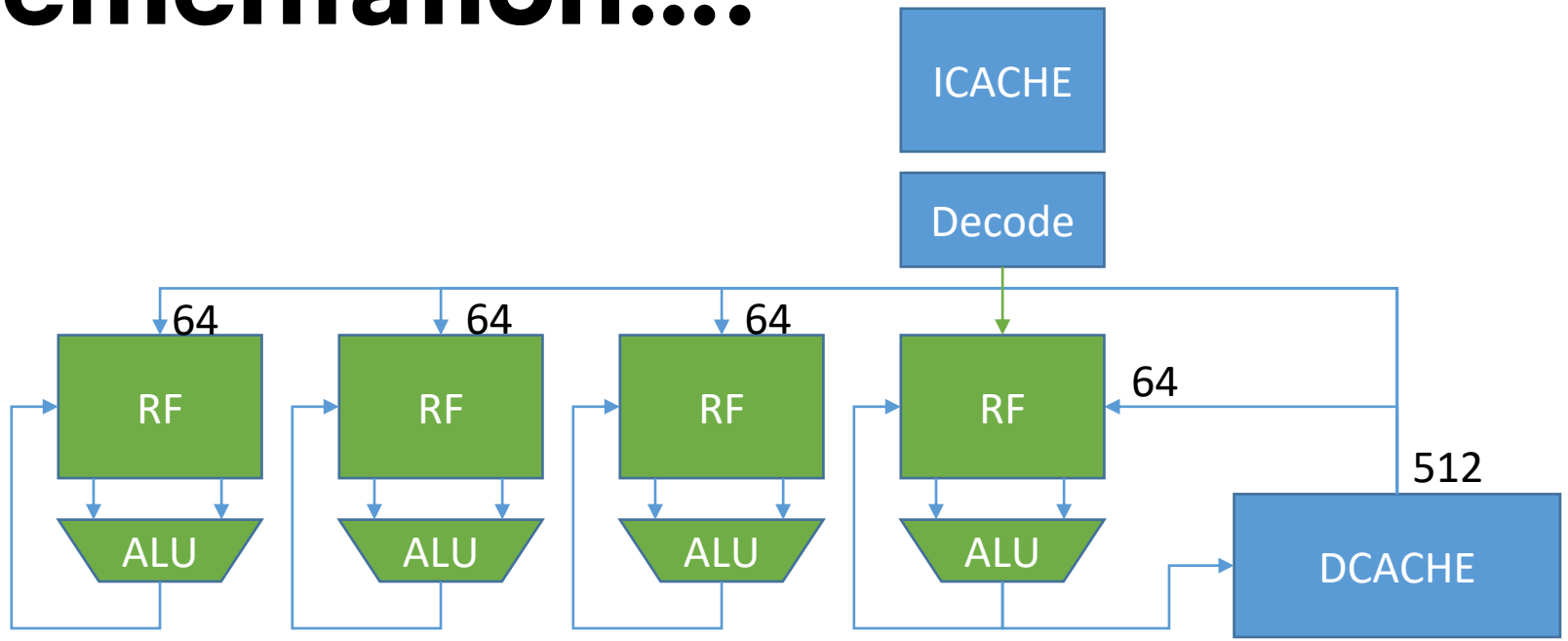


Scalar register 64 bits
Vector register 256 bits

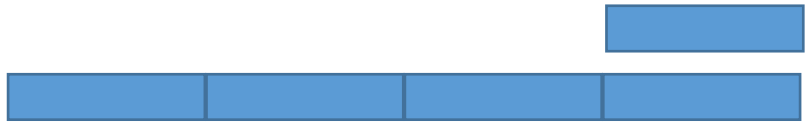


The real implementation....

- vLoad (r1) -> v2
- vLoad 4(r1) -> v3
- vAdd v2, v3 -> v4
- vStore v4 -> (r1)

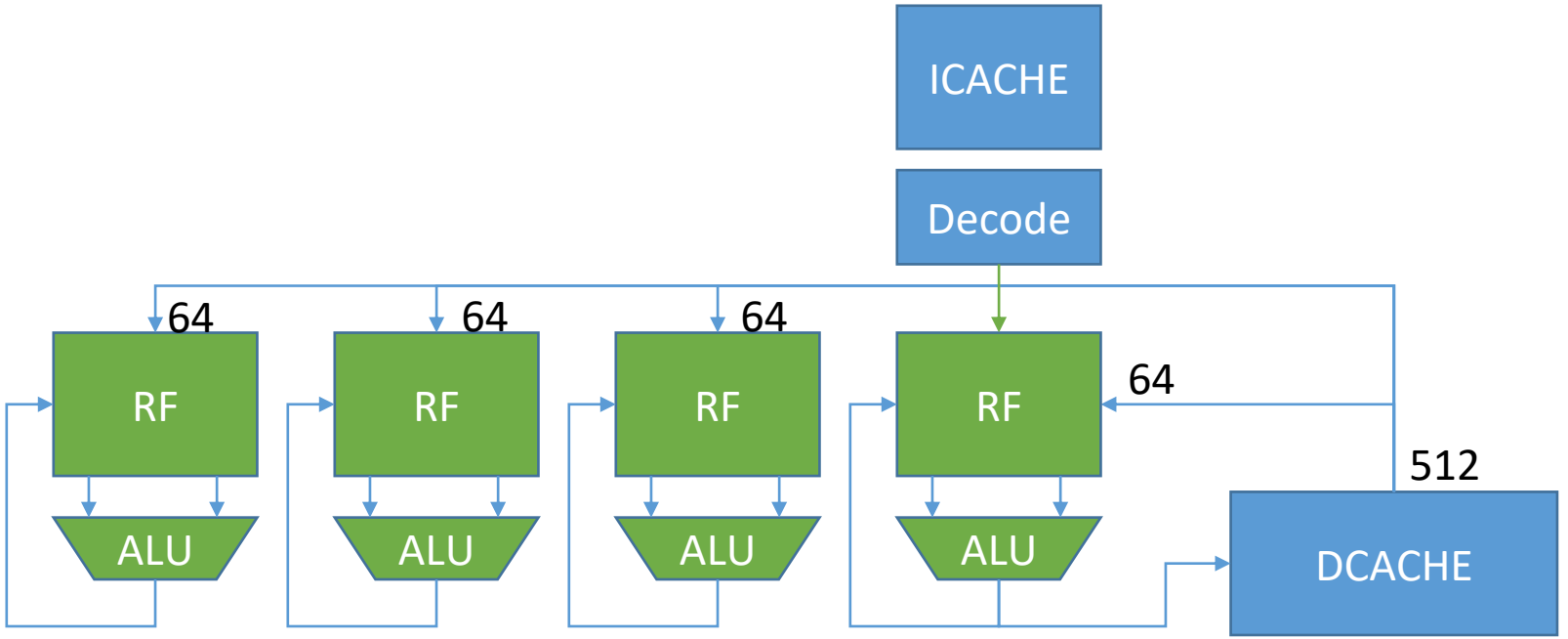


Scalar register 64 bits
Vector register 256 bits



Now... imagine we make vectors 512b....

- vLoad (r1) -> v2 // 512b
- vLoad 4(r1) -> v3 // 512b
- vAdd v2, v3 -> v4 // 512b
- vStore v4 -> (r1) //512b

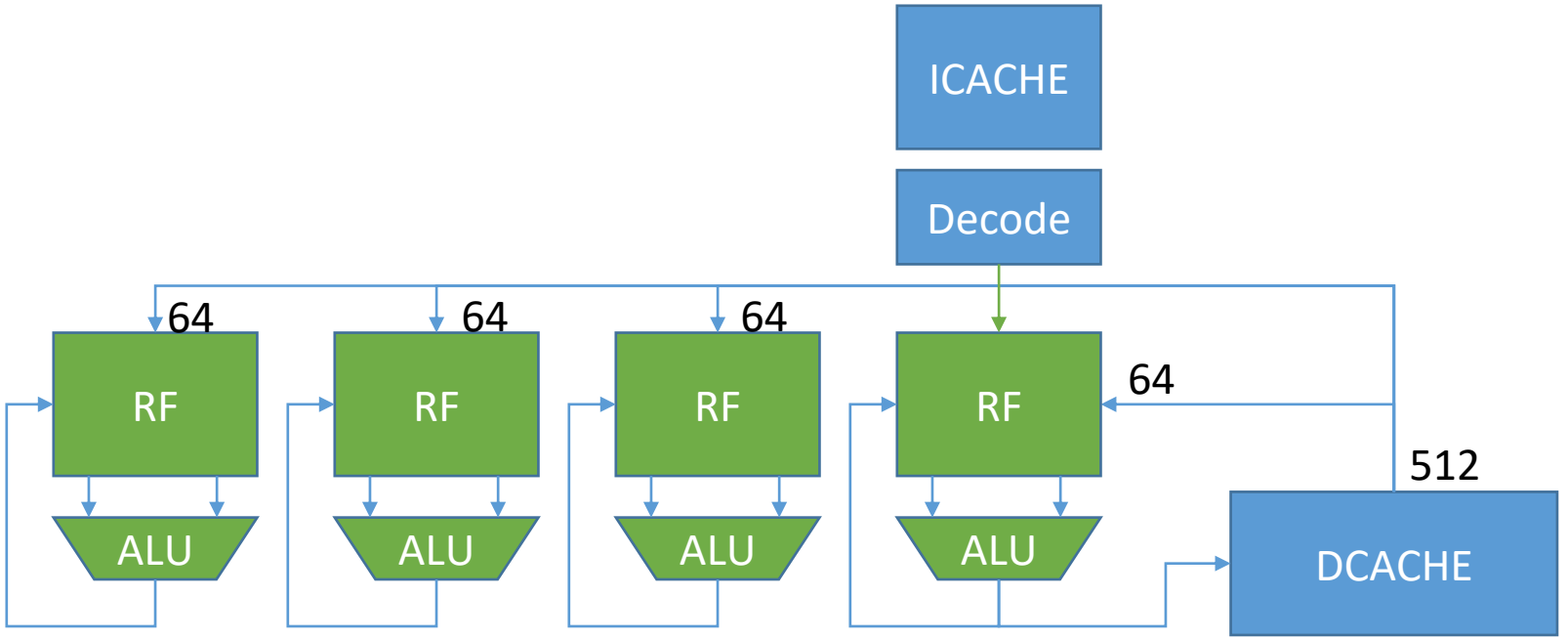


Scalar register 64 bits
Vector register 512 bits

semidynamics

Now… imagine we make vectors 512b….

- vLoad (r1) -> v2 // 512b
- vLoad 4(r1) -> v3 // 512b
- vAdd v2, v3 -> v4 // 512b
- vStore v4 -> (r1) //512b

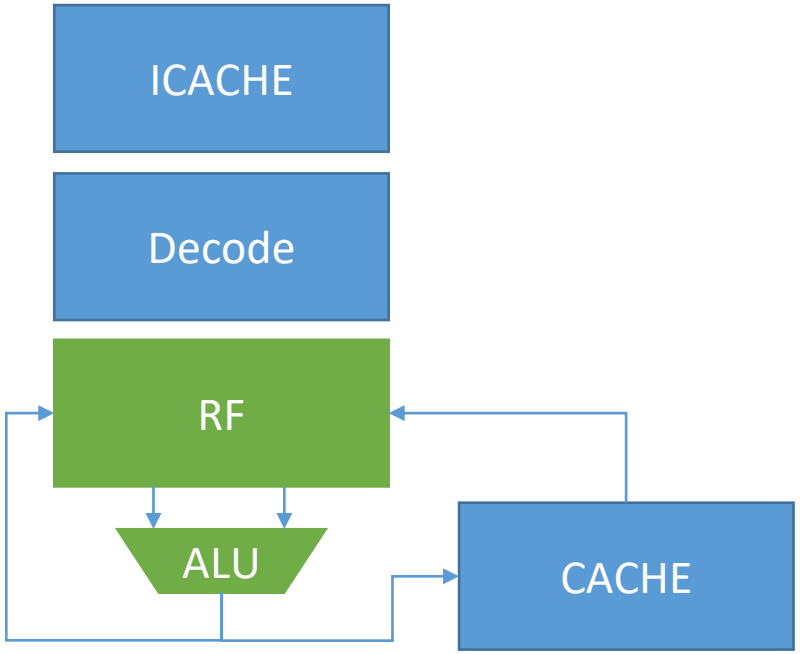


Each operation takes two clocks! 8 clocks for 8 adds

semidynamics

Back to our scalar processor

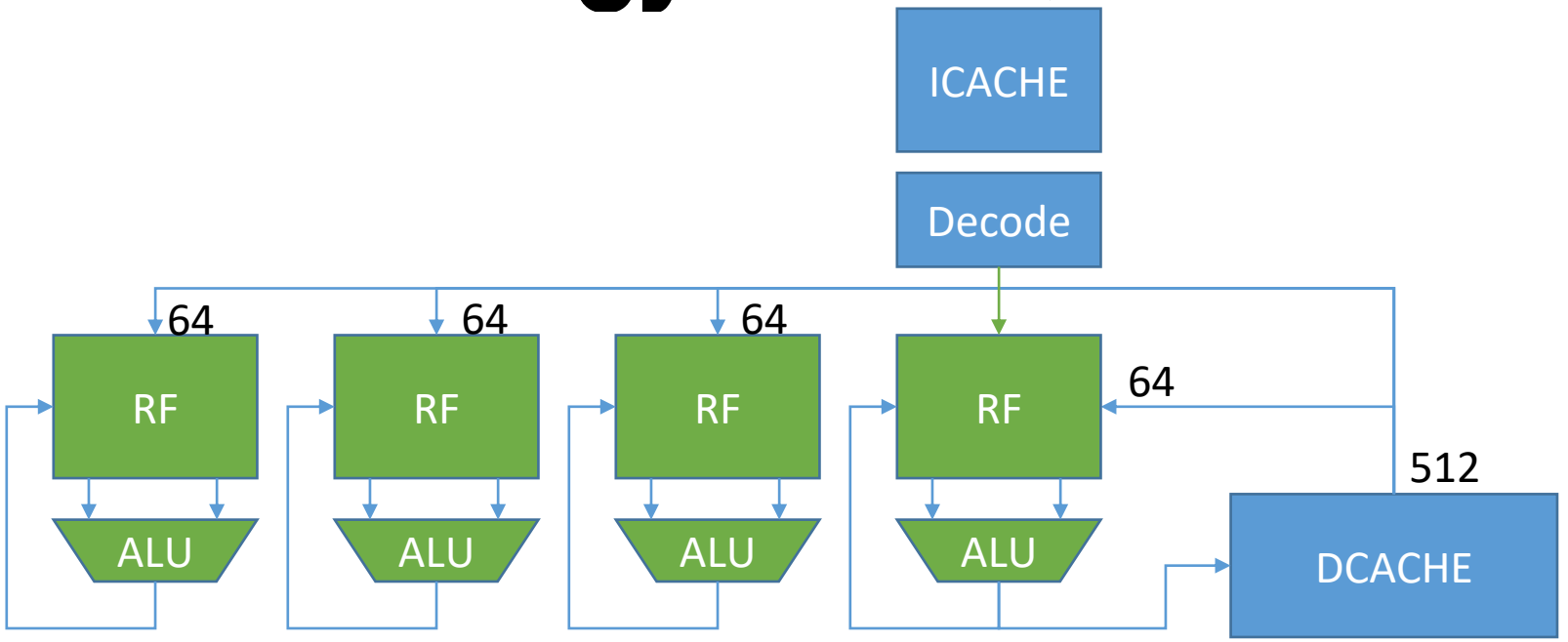
- Load (r1) -> r2
- Load 4(r1) -> r3
- Add r2, r3 -> r4
- Store r4 -> (r1)



32 clocks for 8 adds

Speedup > 4X

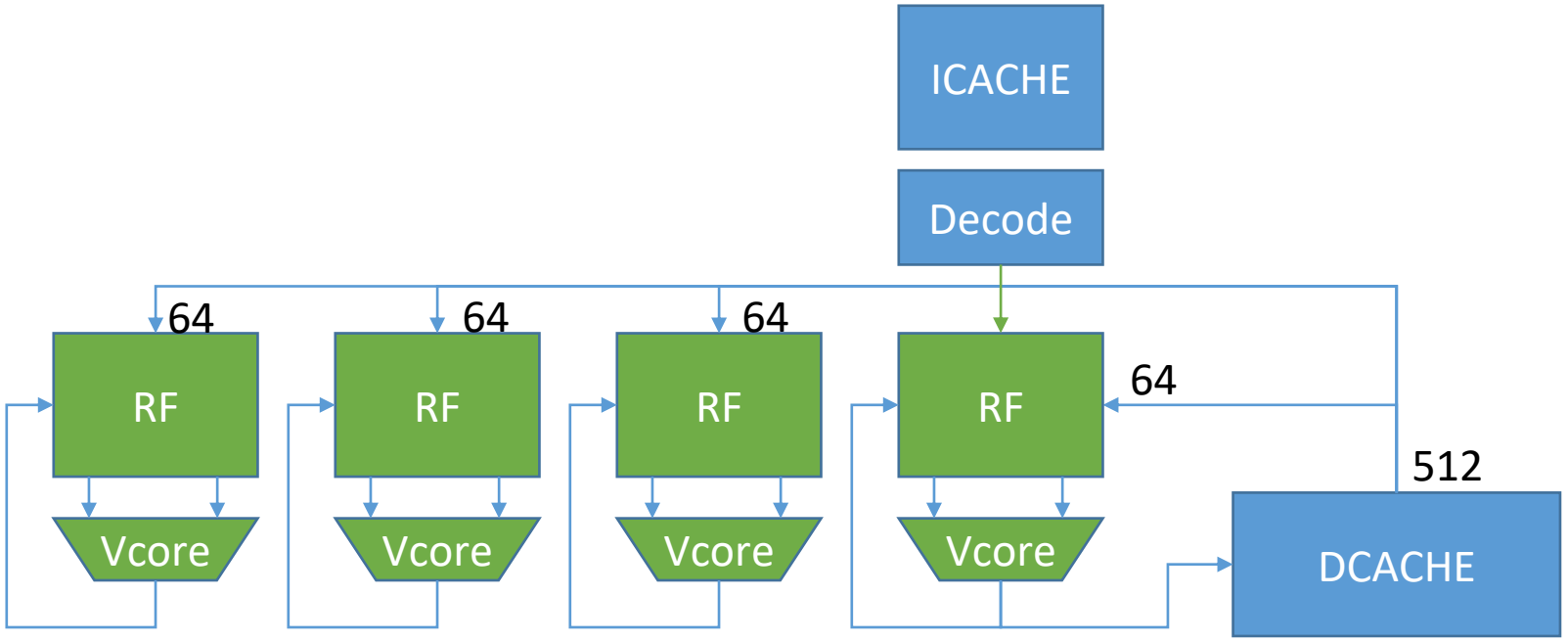
Important RISC-V Terminology: VLEN, DLEN



- VLEN = Vector Length = Number of bits in vector register = 512b
- DLEN = Datapath Length = Number of "alu bits" = 256b
- When VLEN == DLEN → You have a SIMD machine
- When VLEN > DLEN → You have a VECTOR machine

semidynamics

Oh, and we call an “ALU” a “Vcore”

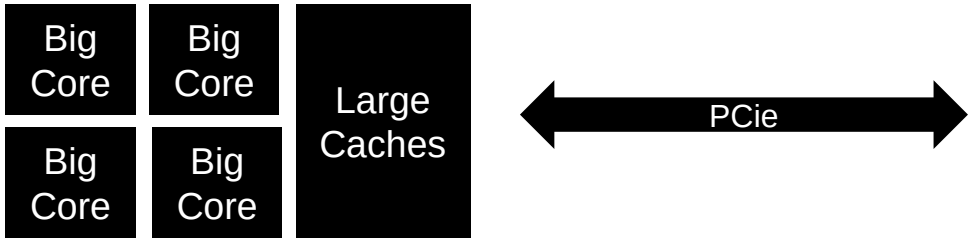


- VLEN = Vector Length = Number of bits in vector register = 512b
- DLEN = Datapath Length = Number of “alu bits” = 256b
- When VLEN == DLEN → You have a SIMD machine
- When VLEN > DLEN → You have a VECTOR machine

semidynamics

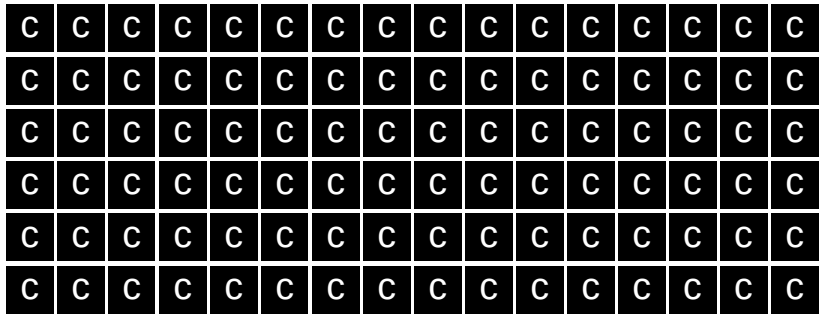
Before the vector unit...

CPU



- Few, large cores
- Easy to program

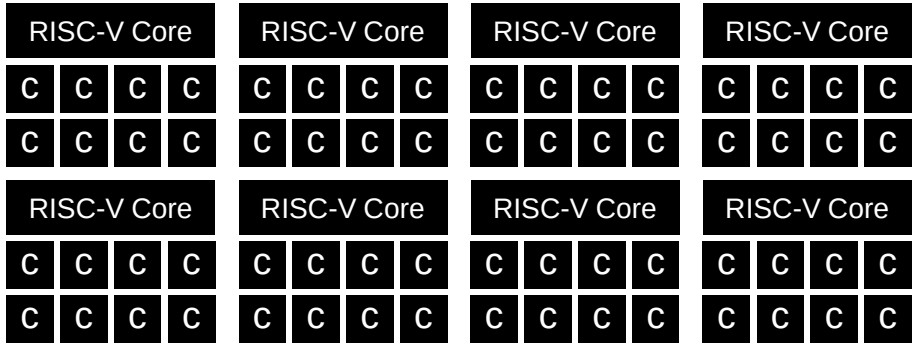
GPU



- Many tiny cores
- Hard to program
- High Performance for Parallel Code
- Communication Latency

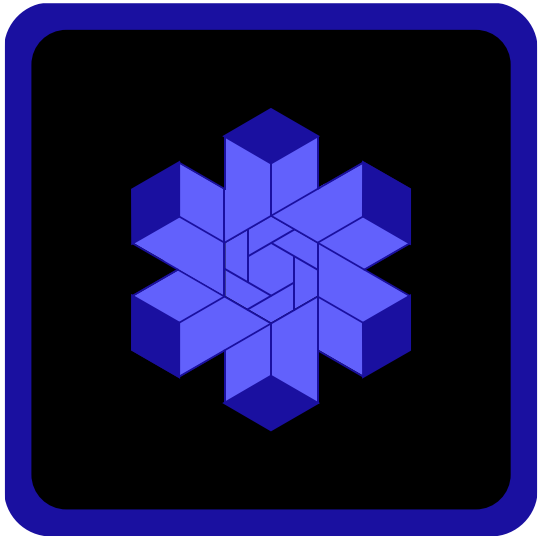
semidynamics

CPU + Vector Unit: best of both worlds



- Bring the GPU compute cores next to the CPU cores
- **Easy** to program
- **High Performance** for Parallel Codes
- **Zero** Communication Latency

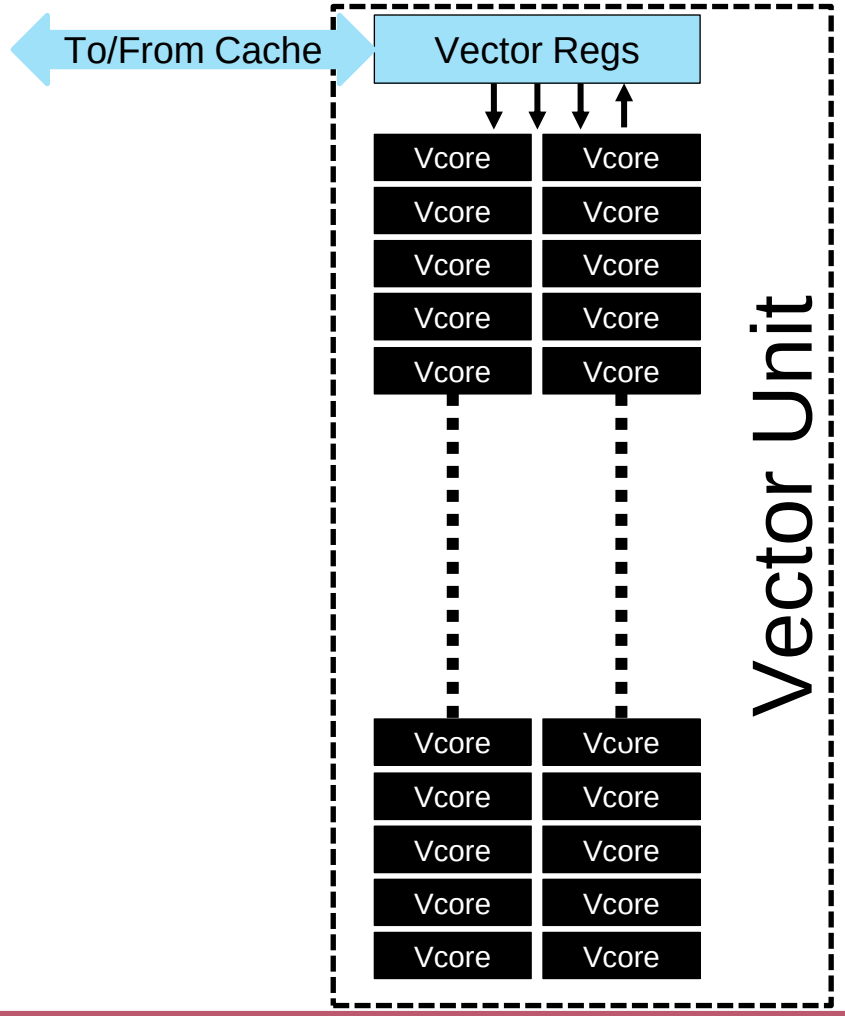
semidynamics



Semidynamic's Highly Configurable OOO Vector Unit IP

semidynamics

What's inside a vector unit?



- 32 vector registers in RISC-V
- A number of "vector cores"
- A (wide) bus from/to the data cache

Three Key Customizations offered by Semidynamics

Customization #1: Data Types

Select from the following data type to configure your vcore:

ELEN=16

INT8
INT16, FP16, BF16

ELEN=32

INT8
INT16, FP16, BF16
INT32, FP32

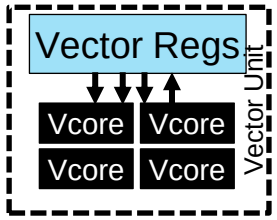
ELEN=64

INT8
INT16, FP16, BF16
INT32, FP32
INT64, FP64

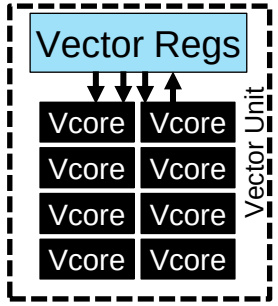
semidynamics

Customization #2: Number of Vector Cores

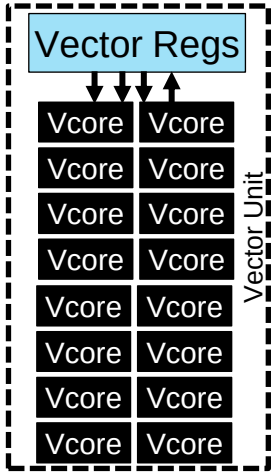
V4



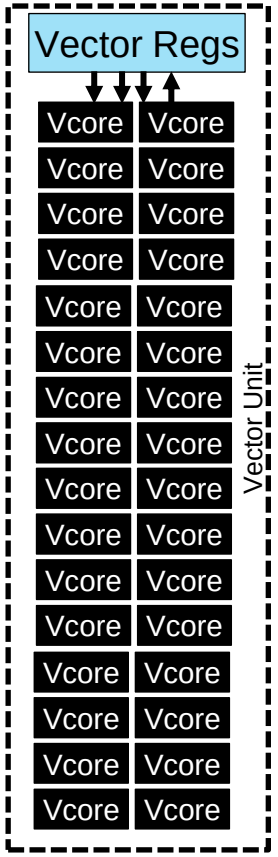
V8



V16



V32



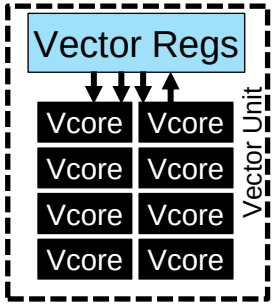
DLEN

VCore	V4	V8	V16	V32
ELEN=16	64	128	256	512
ELEN=32	128	256	512	1024
ELEN=64	256	512	1024	2048

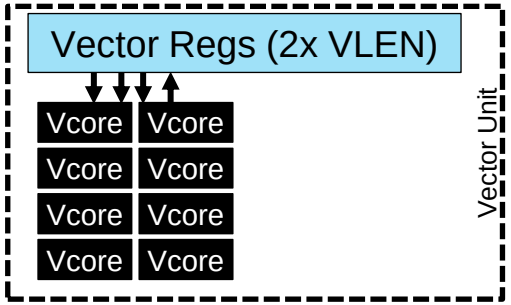
semidynamics

Customization #3: Vector Length (VLEN)

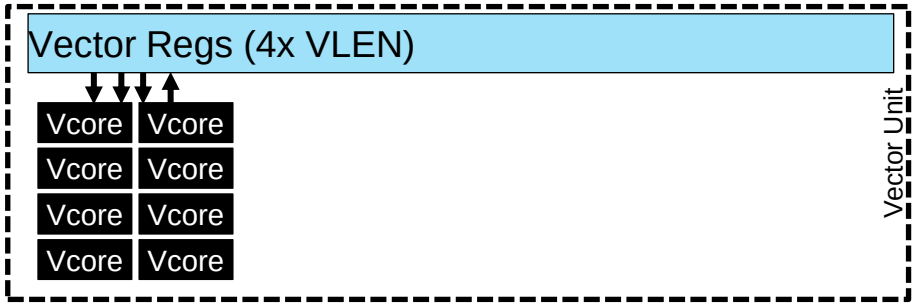
V8 - 1X



V8 - 2X



V8 - 4X



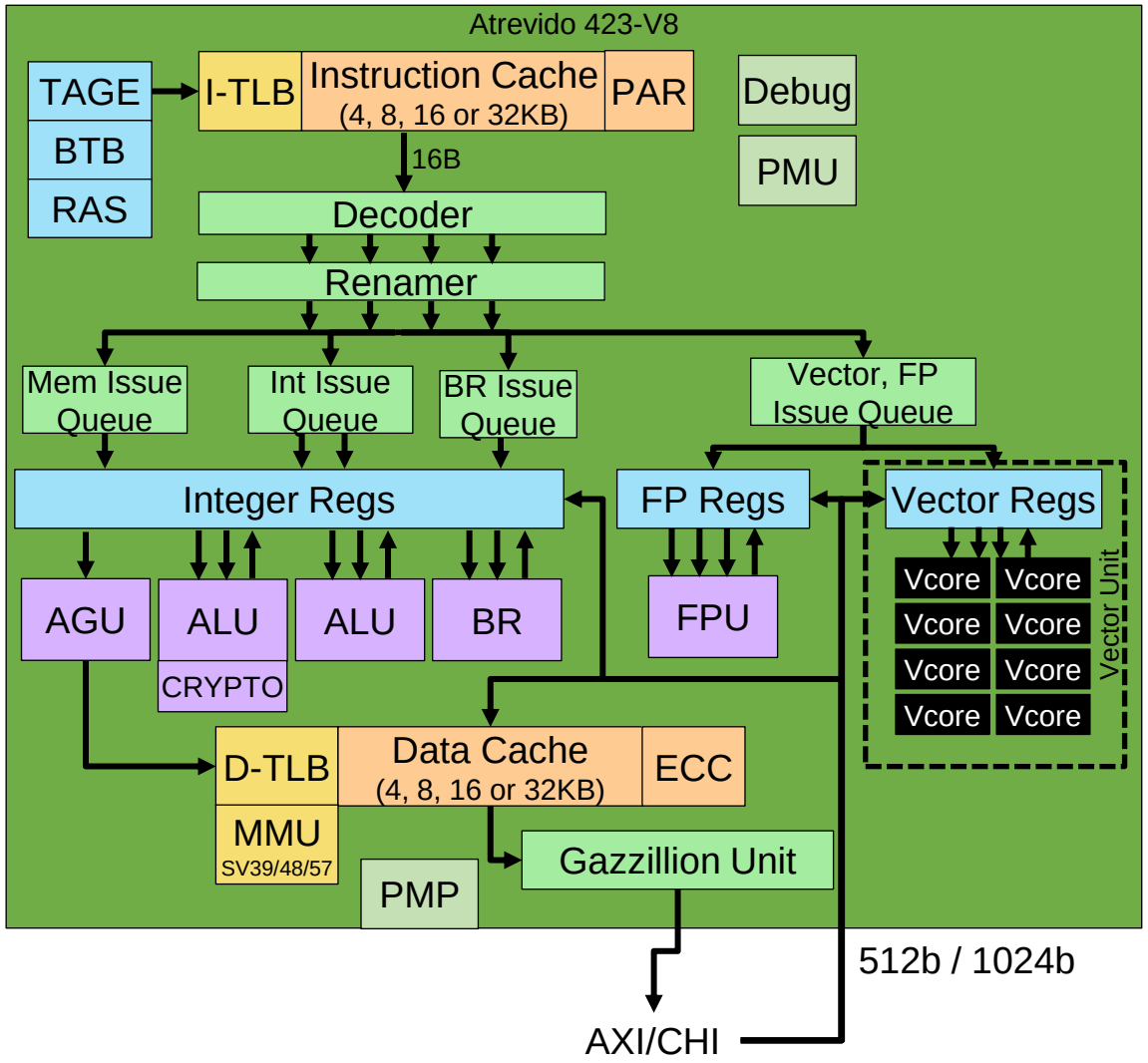
V8 - 8X

1X, 2X, 4X or 8X the number of vector cores

Great for Performance and Power reduction

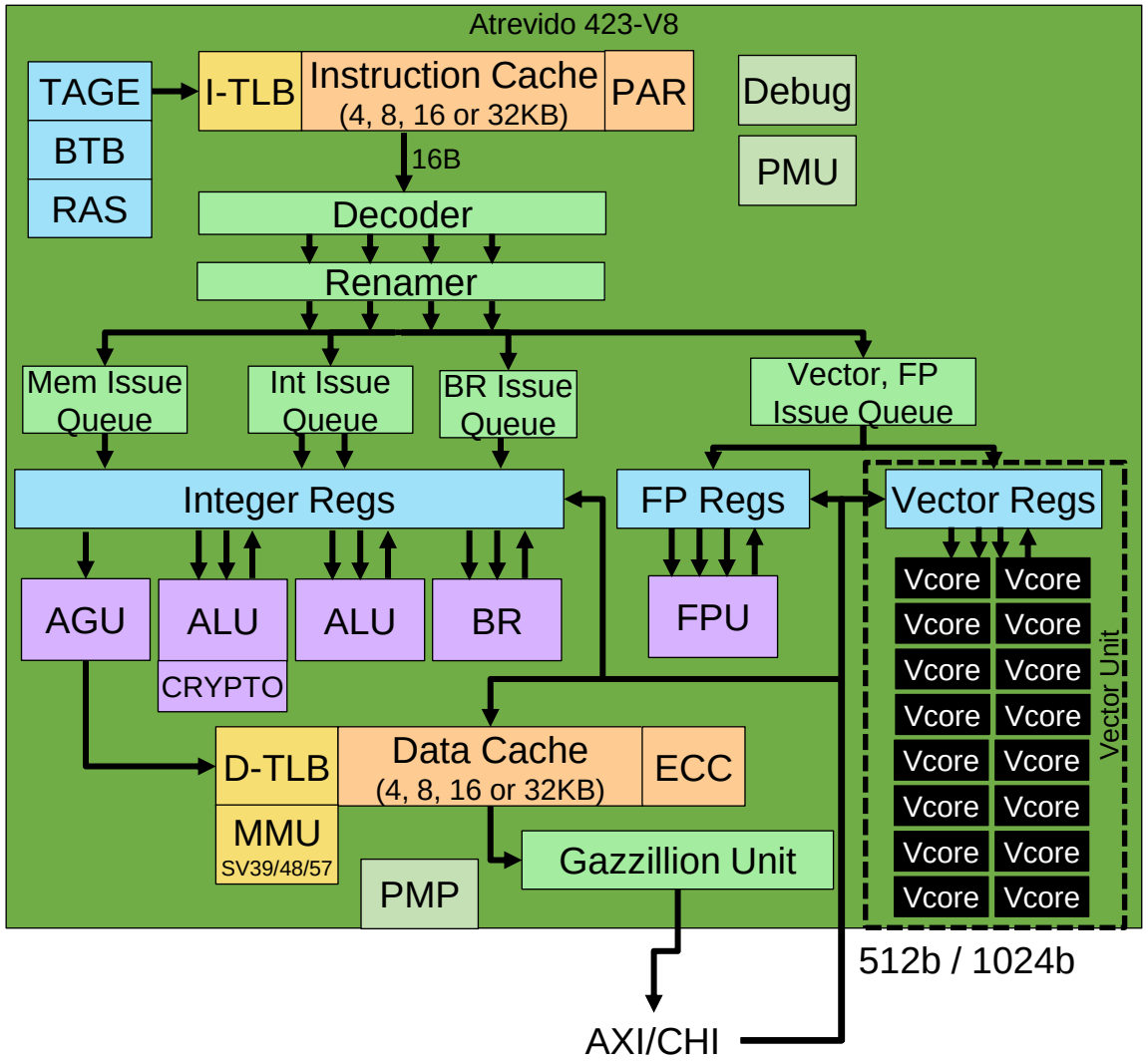
How is the vector unit connected to the core?

Atrevido 423 + V8 Vector Unit



semidynamics

Atrevido 423 + V16 Vector Unit



semidynamics

Gazzillion™ Technology

SERIAL

Traditional method of memory retrieval is stop and go. Request some piece of data from memory and wait doing nothing over several hundred clock cycles for it to come back.

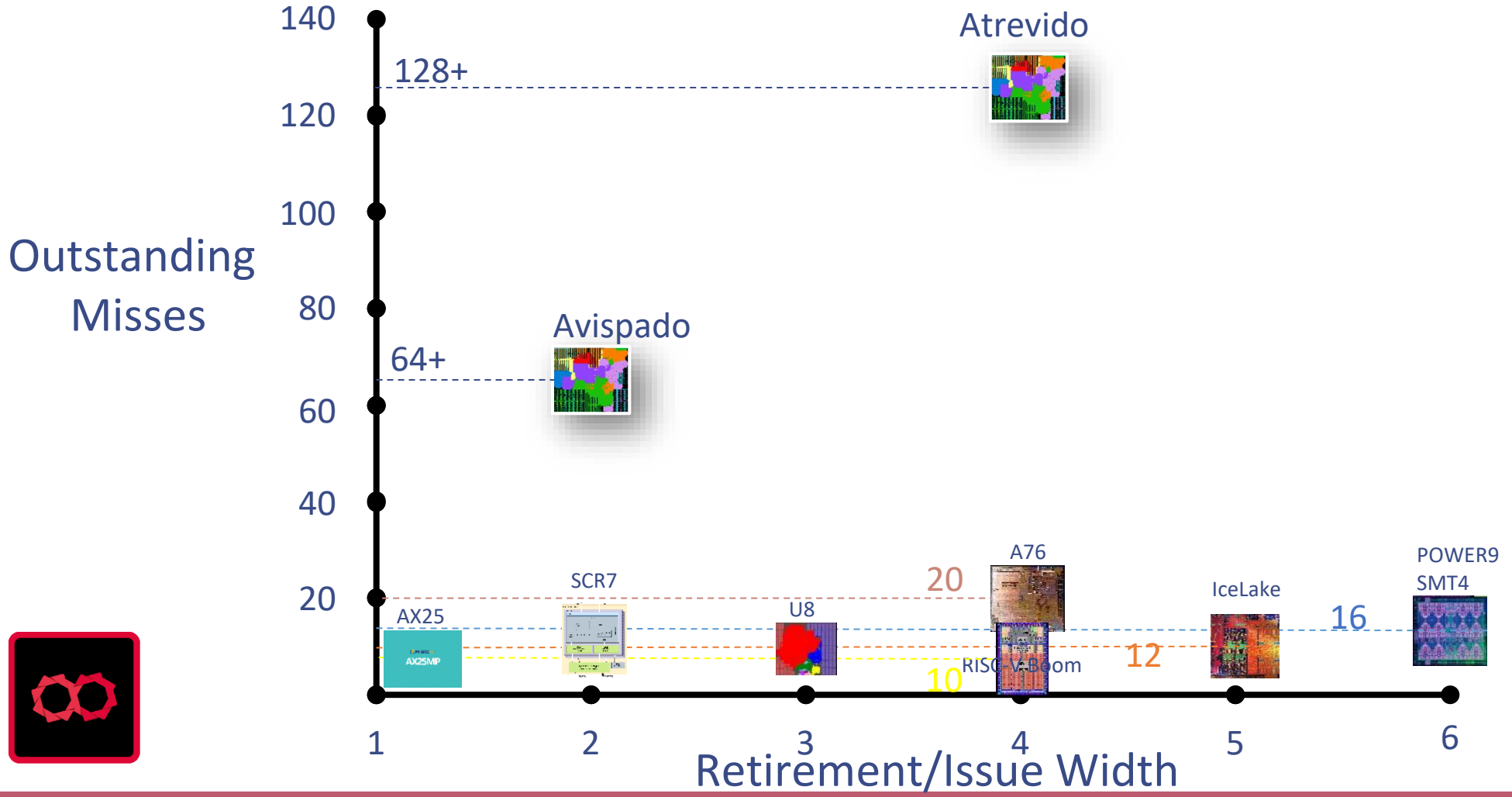
PARALLEL

Gazzillion™ sends out up to 128 simultaneous requests for data from memory. Whichever data request comes back first is worked on and then the next one back so that core is always working. 128 of these streams in parallel bringing data back to the core really turbo charges performance.

Turbo charges memory retrieval
Tolerates memory latency like no other

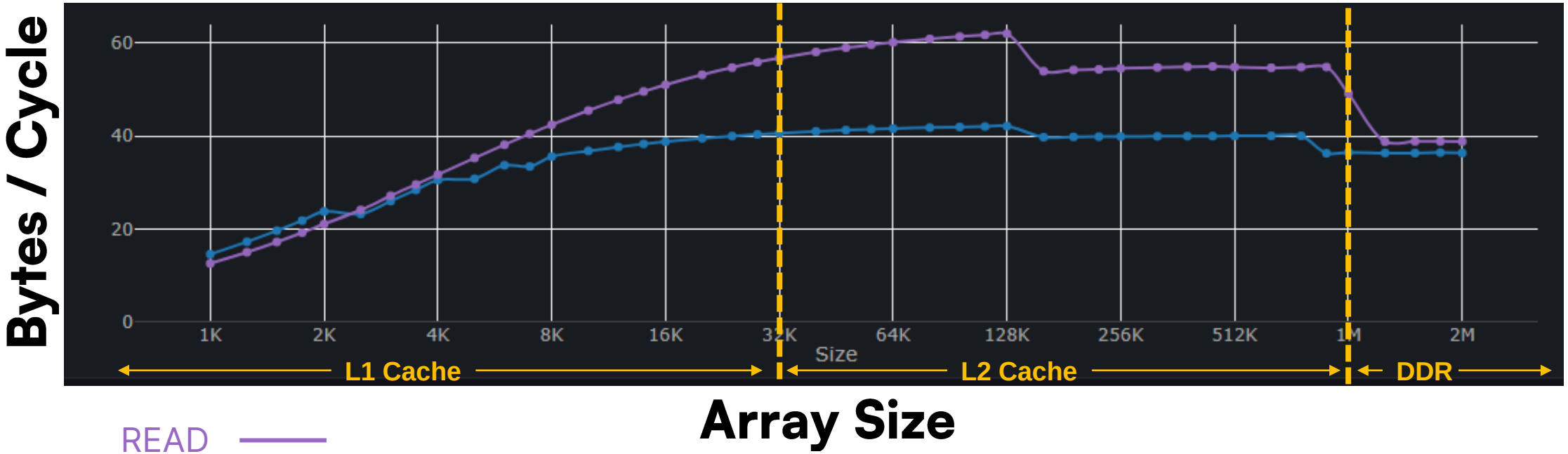
semidynamics

Gazzillion compared to other CPUs...



Vector + Gazzillion: A bandwidth rocket!

Can you find a core out there capable of streaming data at over 60 Bytes/cycle?
And from main DDR memory (not from your cache)? We don't think so ☺



8 vector cores, 32X vector length

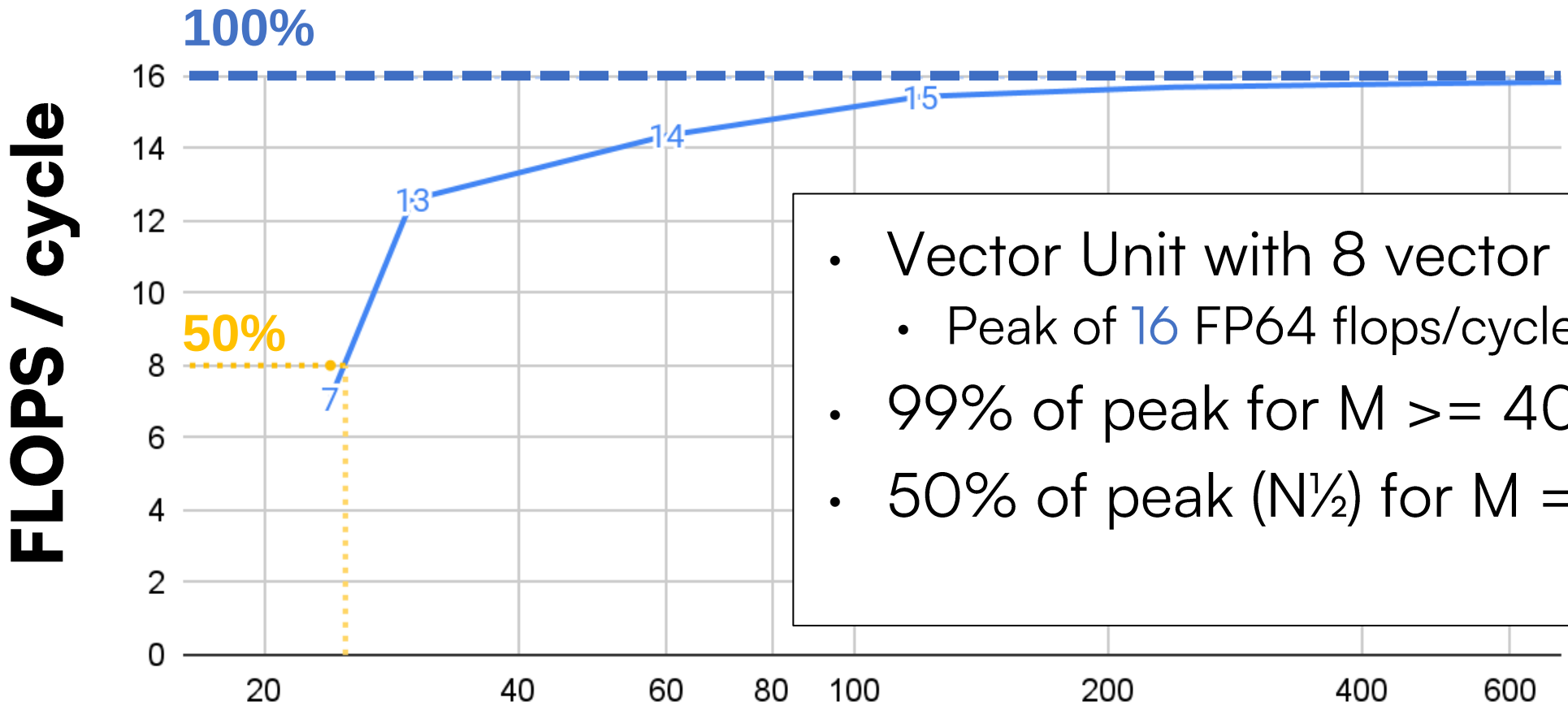
semidynamics

Semidynamic's Vector Performance

semidynamics

DGEMM on OOO V8 Vector Unit

(FP64 matrix multiply)



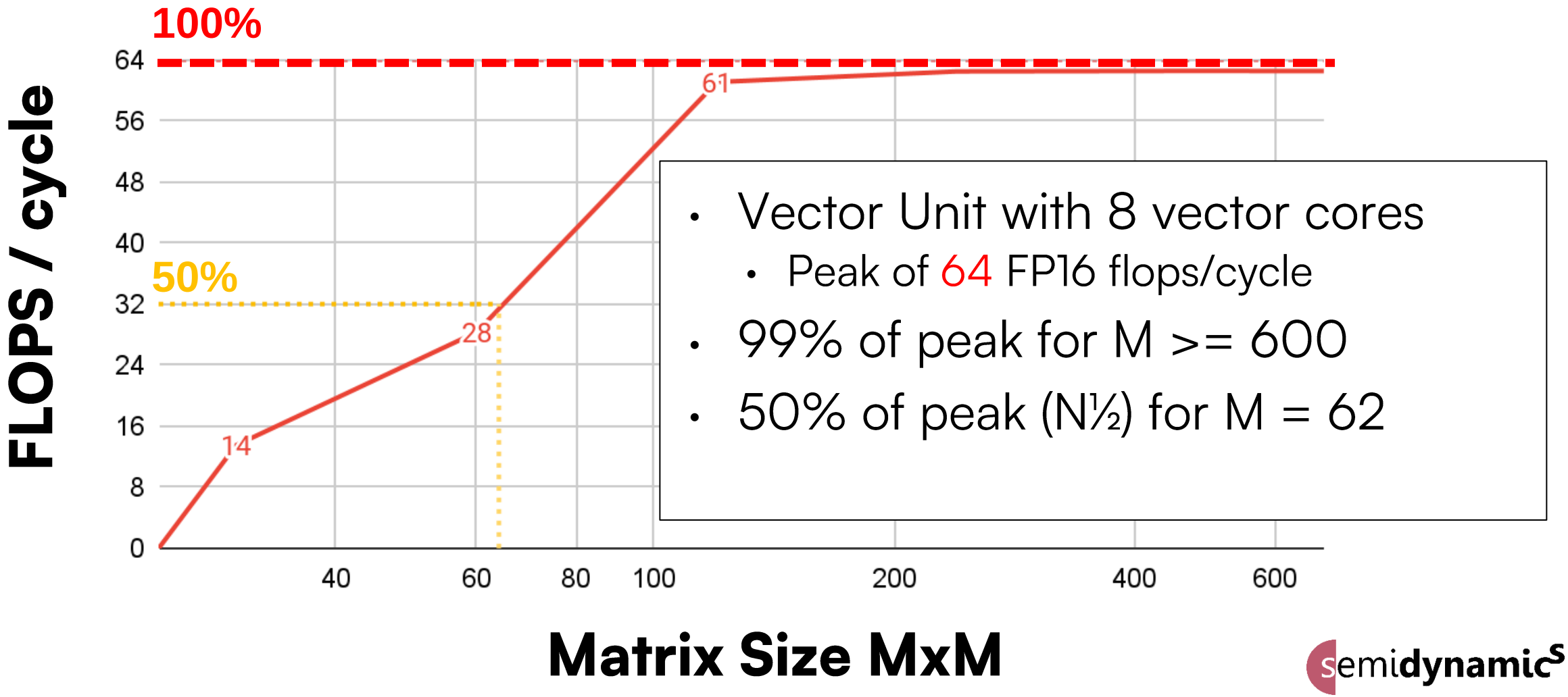
- Vector Unit with 8 vector cores
  - Peak of 16 FP64 flops/cycle
- 99% of peak for M >= 400
- 50% of peak (N½) for M = 24

Matrix Size MxM

semidynamics

HGEMM on OOO V8 Vector Unit

(FP16 matrix multiply)

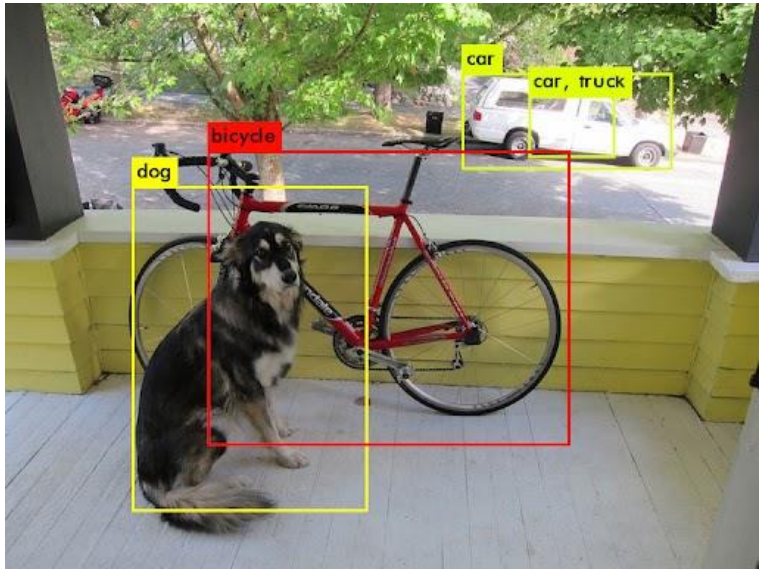


- Vector Unit with 8 vector cores
  - Peak of 64 FP16 flops/cycle
- 99% of peak for M >= 600
- 50% of peak (N½) for M = 62

semidynamics

Yolo on OOO V8 Vector Unit

- YOLOv3-tiny:
  - 24 layers, 5.56 Gops/frame, ~9M params
  - Using SGEMM (FP32) for Matrix Multiplication



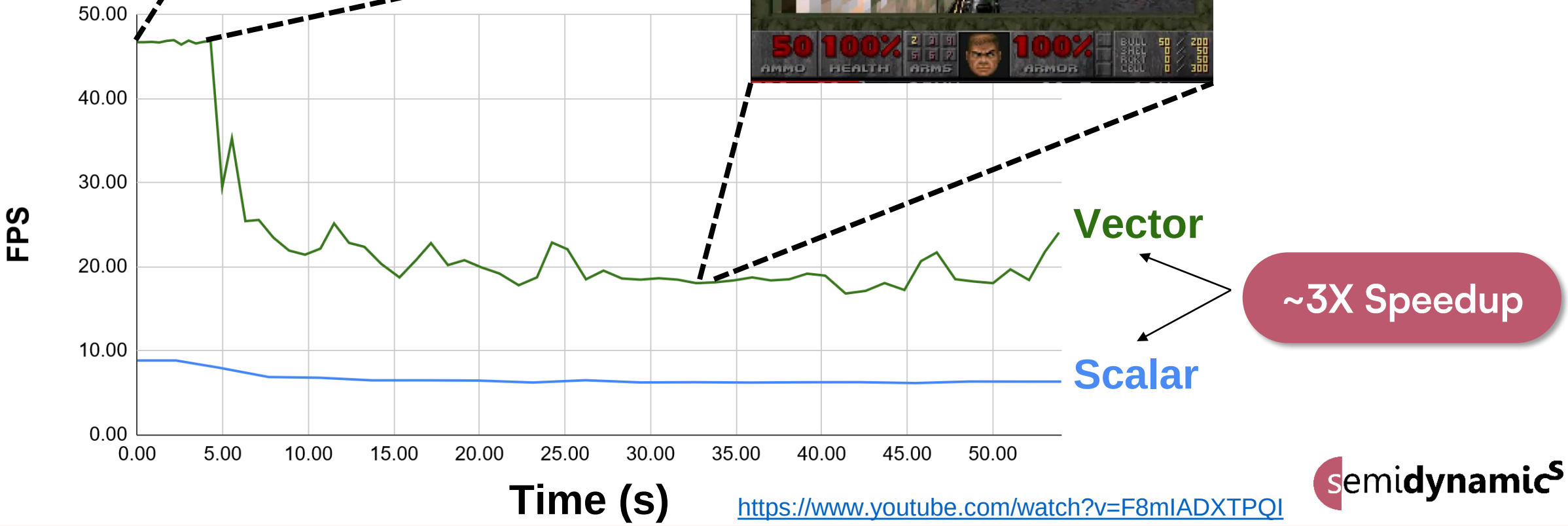
Platform	Vector/Cuda Cores	Frequency (Ghz)	FPS	FPS per 8 vector cores @ 1Ghz
Jetson TX2	256	1.30	19[1]	0.46
Jetson AGX Xavier	512	1.38	32[1]	0.36
GTX Titan X	3072	1.09	220[2]	0.53
Atrevido 423-V8	**8**	**1.00**	**0.84**	**0.84**

58% higher performance per vector core

semidynamics

[1] https://www.researchgate.net/publication/351347699_Real-Time_On_Board_Deep_Learning_Fault_Detection_for_Autonomous_UAV_Inspections
[2] https://pireddie.com/darknet/yolo/

Vectorized Doom on OOO V8 Vector Unit



https://www.youtube.com/watch?v=F8mIADXTPQI

semidynamics

THANK YOU!

semidynamics