esperanto.ai

# Why Esperanto picked RISC-V for HPC Computing

Dave Ditzel
Founder and CTO
Esperanto Technologies

November 18, 2024
RISC-V Workshop Keynote
Supercomputing24

email: dave@esperanto.ai

# Agenda - Some history for fun and a bit of Esperanto's products too

esperanto.ai

History of instruction sets leading us to RISC-V

Why RISC-V was the right choice for Esperanto

Esperanto ET-SoC-1 at the beginning of RISC-V

Esperanto 4nm ET-SoC-2 and 2nm ET-SoC-3 roadmap with RVV Vector support

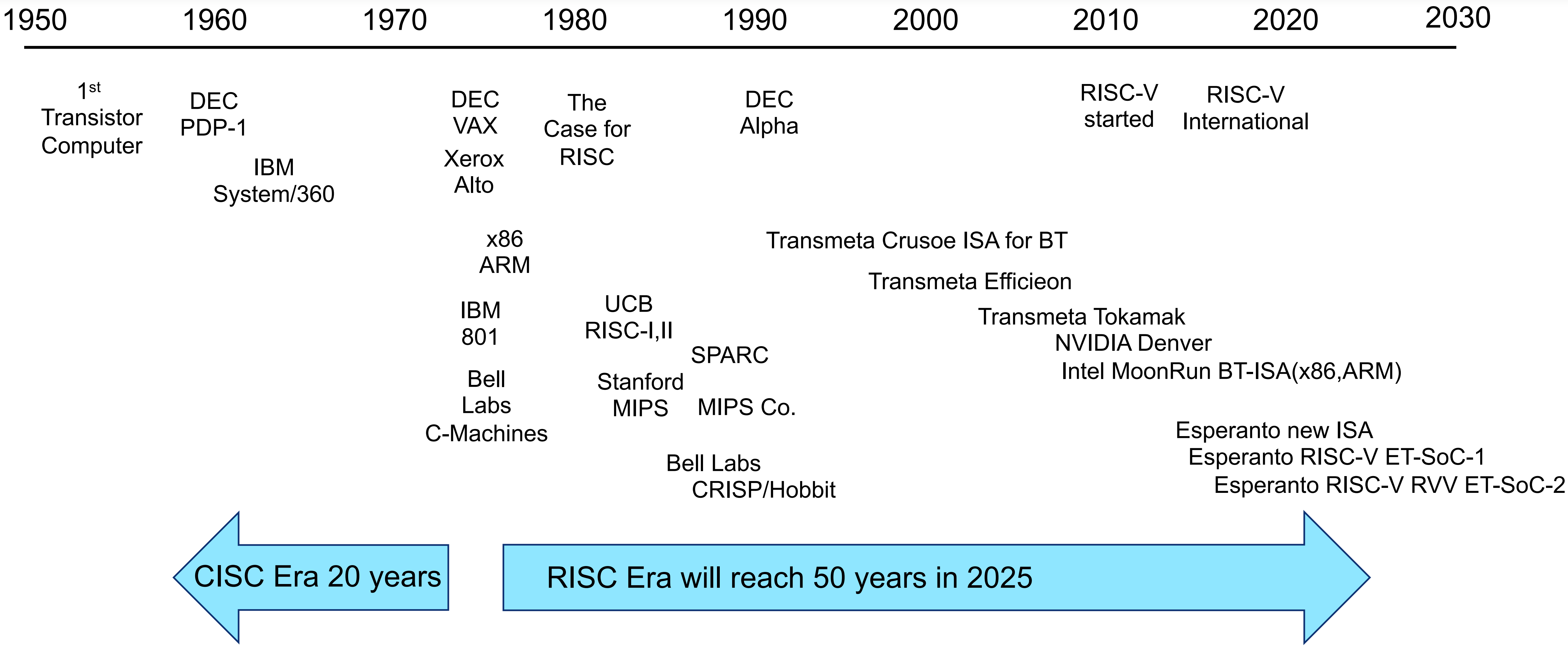
Conclusion: RISC-V is the right instruction set if you want to run both AI and HPC.

# Useful quote for computer architects

“Those who cannot remember the past are condemned to repeat it.”

**George Santayana,**
**The Life of Reason,**
**1905**.

# My Computer Development Timeline



1950 1960 1970 1980 1990 2000 2010 2020 2030

1st Transistor Computer

DEC PDP-1

IBM System/360

DEC VAX

Xerox Alto

x86 ARM

IBM 801

Bell Labs C-Machines

The Case for RISC

UCB RISC-I,II

Stanford MIPS

SPARC

MIPS Co.

Bell Labs CRISP/Hobbit

DEC Alpha

Transmeta Crusoe ISA for BT

Transmeta Efficieon

Transmeta Tokamak

NVIDIA Denver

Intel MoonRun BT-ISA(x86,ARM)

RISC-V started

RISC-V International

Esperanto new ISA

Esperanto RISC-V ET-SoC-1

Esperanto RISC-V RVV ET-SoC-2

CISC Era 20 years

RISC Era will reach 50 years in 2025

# Era of High Level Language Computer Architecture

later called CISC (Complex Instruction-set Computers)

# DEC VAX-11/780: a 32-bit CISC

**A real 70’s architecture:**

- VAX: Transitioned from 16-bit PDP-11 assembly to 32-bit addresses and compiled code
- ISA made it easy to take high level language statements and cast into VAX assembly code

**Instruction set heavily influenced by availability of 8-bit wide integrated circuits**

**VAX ISA composed of variable length instructions**

- Instruction opcode (1 or 2 bytes) followed by up to 6 operands
- First operand descriptor (1 byte)
- First operand data (0-4 bytes
- Second operation descriptor
- Second operand data
- Third operand descriptor
- Third operand data (0-4 bytes)
- …

**This was great when decoding serially one byte at a time,**
**but a nightmare for pipelined or later superscalar implementations**

# Xerox Palo Alto Research Center: Alto

Earliest Graphical User Interface with mouse

5.8 MHz microcoded CPU

User loadable instruction sets led to several HLLCA type bytecoded instruction sets for languages like BCPL, Smalltalk and MESA

Enabled exploration of highly tailored instruction sets

But raised the question
> “Why not compile to lowest microcode level?”

# SYMBOL:
# The Ultimate CISC

# SYMBOL: The ultimate Complex Instruction Set Computer

**SYMBOL was a High Level Language Computer announced in 1971 by Fairchild Semiconductor**

**Supported by Gordon Moore and Robert Noyce at Fairchild until they left to form Intel**

**Goal was to reduce cost of software by using hardware instead, literally “programmers cost too much”.**

**Implemented Compiler, Text Editor and Operating System entirely in logic gates**

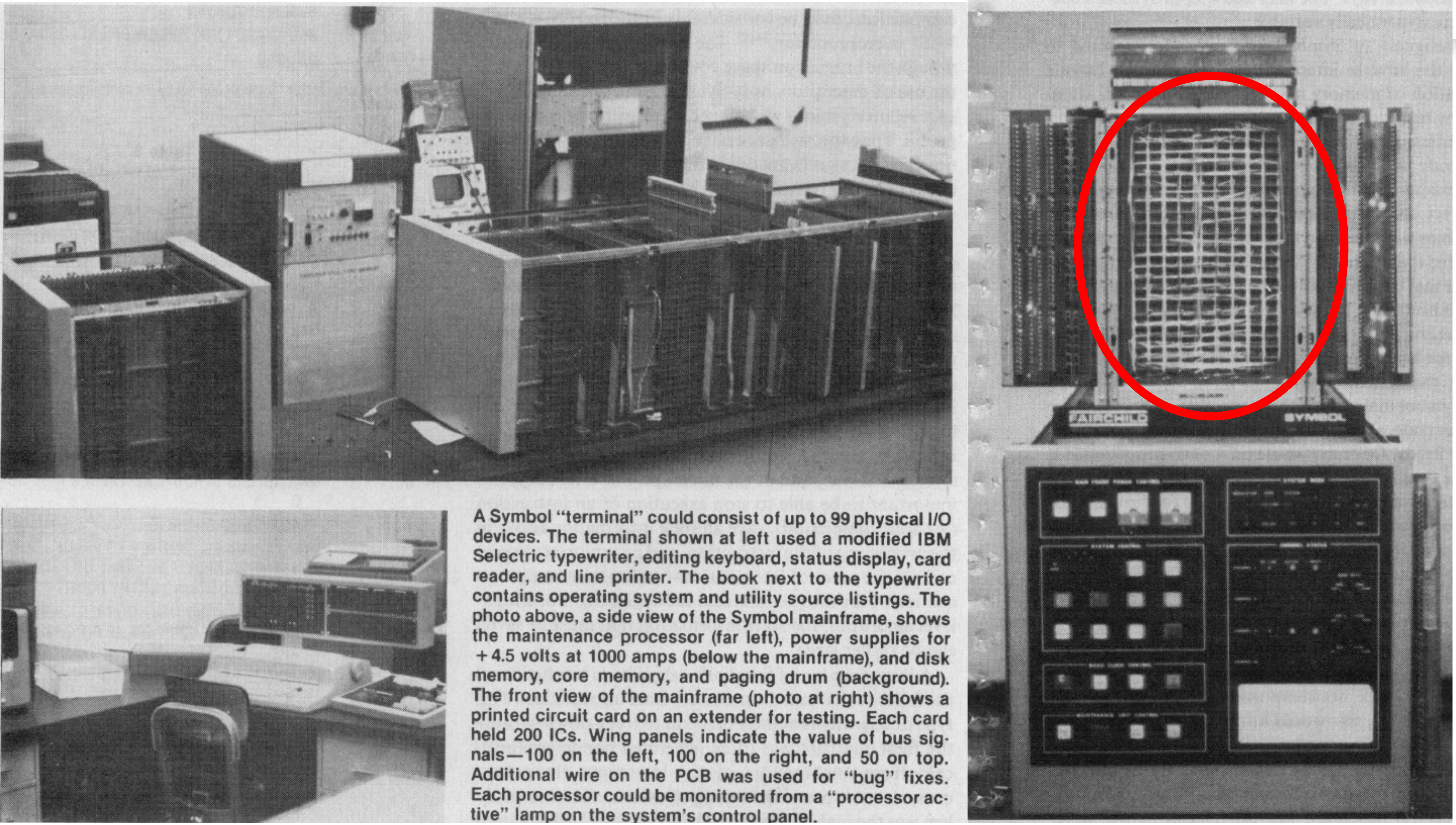
- 2 gates or 1 Flip-Flop per 14-pin DIP package, no ROM
- 20 thousand chips – took **several years** to debug at ISU
- Instruction set was bytecode mapped almost 1-1 from high level ALGOL/LISP like typeless language
- Tagged architecture with descriptors, and “logical memory” that was not linearly addressable.
- Five years of my life thinking about CISC vs RISC, i.e. better ways to use 20K chips, programming this machine.

**I got to work on this computer while a student for 5 years, after it was donated to ISU**

**Lessons:**

- Don’t build your compiler or OS in logic gates
- Use right combination of Hardware, Software or Microcode

# Debugging SYMBOL, the inspiration for RISC



A Symbol "terminal" could consist of up to 99 physical I/O devices. The terminal shown at left used a modified IBM Selectric typewriter, editing keyboard, status display, card reader, and line printer. The book next to the typewriter contains operating system and utility source listings. The photo above, a side view of the Symbol mainframe, shows the maintenance processor (far left), power supplies for +4.5 volts at 1000 amps (below the mainframe), and disk memory, core memory, and paging drum (background). The front view of the mainframe (photo at right) shows a printed circuit card on an extender for testing. Each card held 200 ICs. Wing panels indicate the value of bus signals—100 on the left, 100 on the right, and 50 on top. Additional wire on the PCB was used for "bug" fixes. Each processor could be monitored from a "processor active" lamp on the system's control panel.

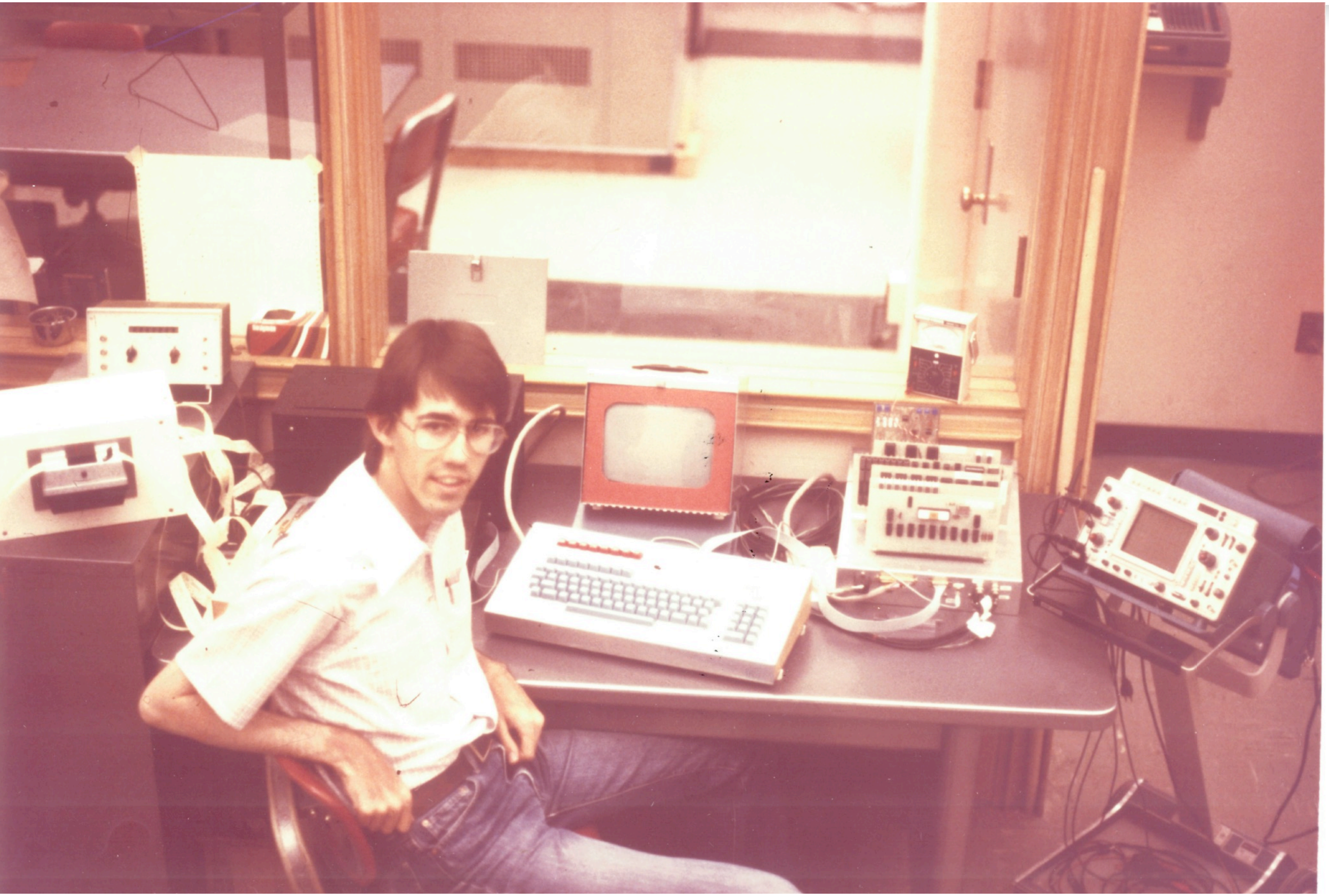
Inside red circle are 220 chips.

In order to fix a bug, white wires added on top of blue pc board to make logic changes.

Many of the 100 boards were covered with white bug fixes.

Source: D. Ditzel, Reflections on the High Level Language SYMBOL Computer Systems, IEEE Computer, July 1981.

# 1978 Photo: My career started at the same time as the first microprocessors



My personal hobby computer

8-bit 6502 Microprocessor

4 MHz

4K Bytes of main memory

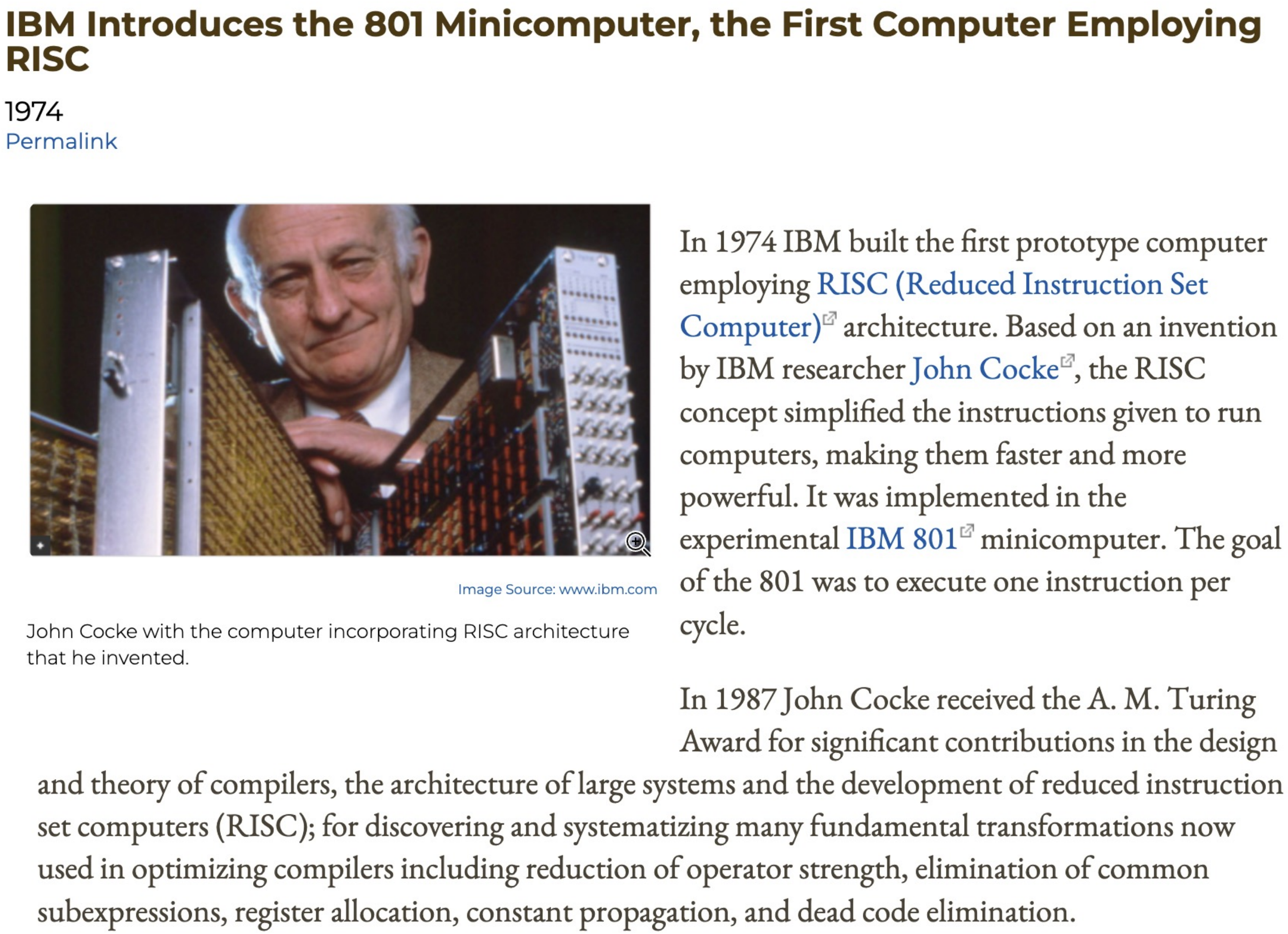
Hand “wire-wrapped”

Paper tape reader

Hand wound transformer on power supply.

Note – base of SYMBOL computer
in background

# IBM 801: In my view, the first true RISC



## IBM Introduces the 801 Minicomputer, the First Computer Employing RISC

1974
Permalink

Image Source: www.ibm.com

John Cocke with the computer incorporating RISC architecture that he invented.

In 1974 IBM built the first prototype computer employing RISC (Reduced Instruction Set Computer) architecture. Based on an invention by IBM researcher John Cocke, the RISC concept simplified the instructions given to run computers, making them faster and more powerful. It was implemented in the experimental IBM 801 minicomputer. The goal of the 801 was to execute one instruction per cycle.

In 1987 John Cocke received the A. M. Turing Award for significant contributions in the design and theory of compilers, the architecture of large systems and the development of reduced instruction set computers (RISC); for discovering and systematizing many fundamental transformations now used in optimizing compilers including reduction of operator strength, elimination of common subexpressions, register allocation, constant propagation, and dead code elimination.

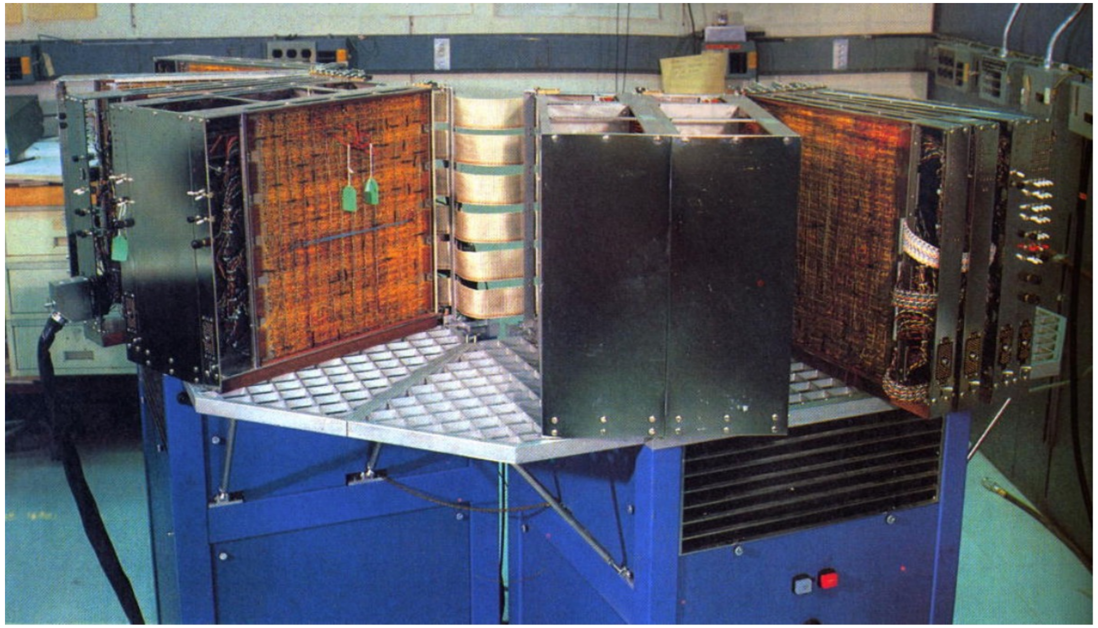
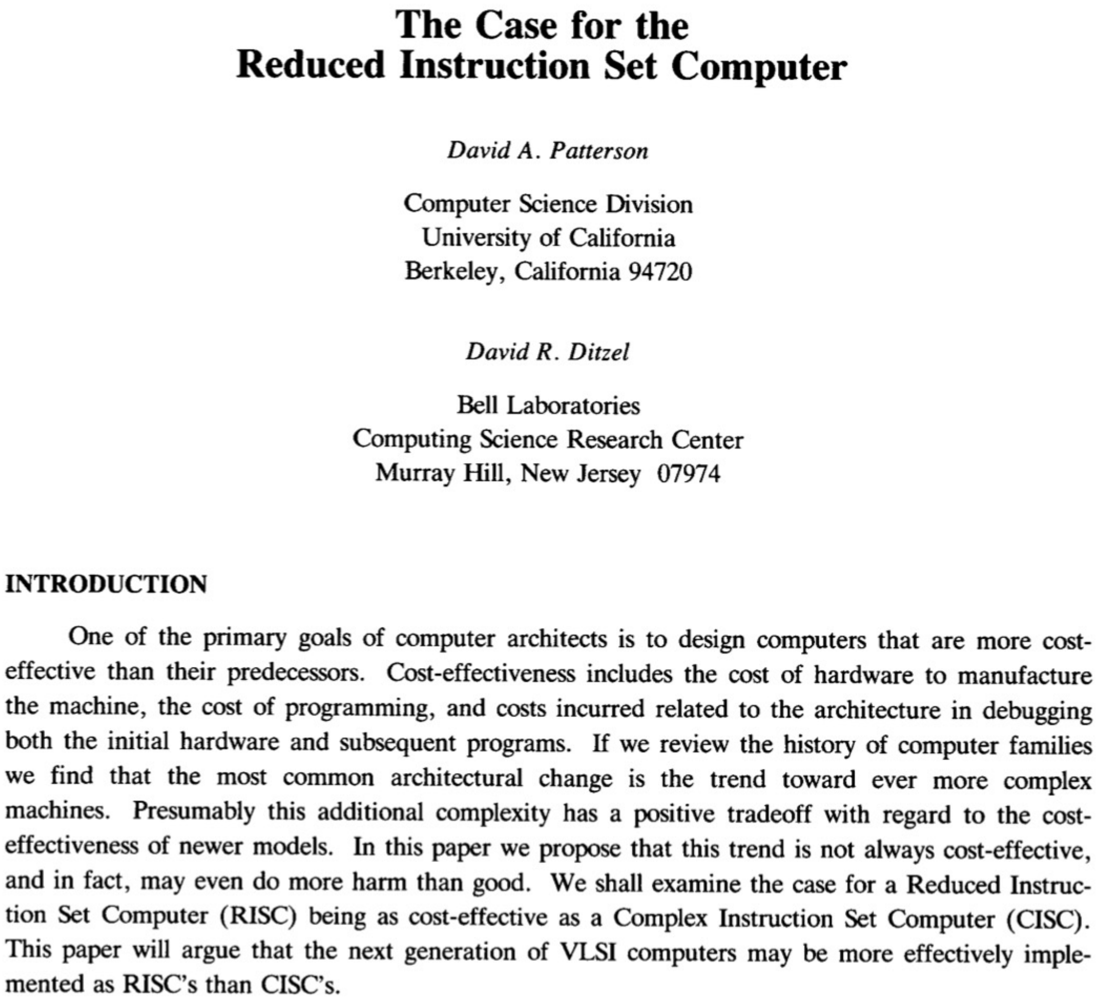


Photo of IBM 801 minicomputer.

The name 801 was from the IBM building number
of the T.J. Watson Research Center in Yorktown Heights

First described at ASPLOS-1 in 1982

# 1980: The Case for RISC Published



## The Case for the Reduced Instruction Set Computer

*David A. Patterson*

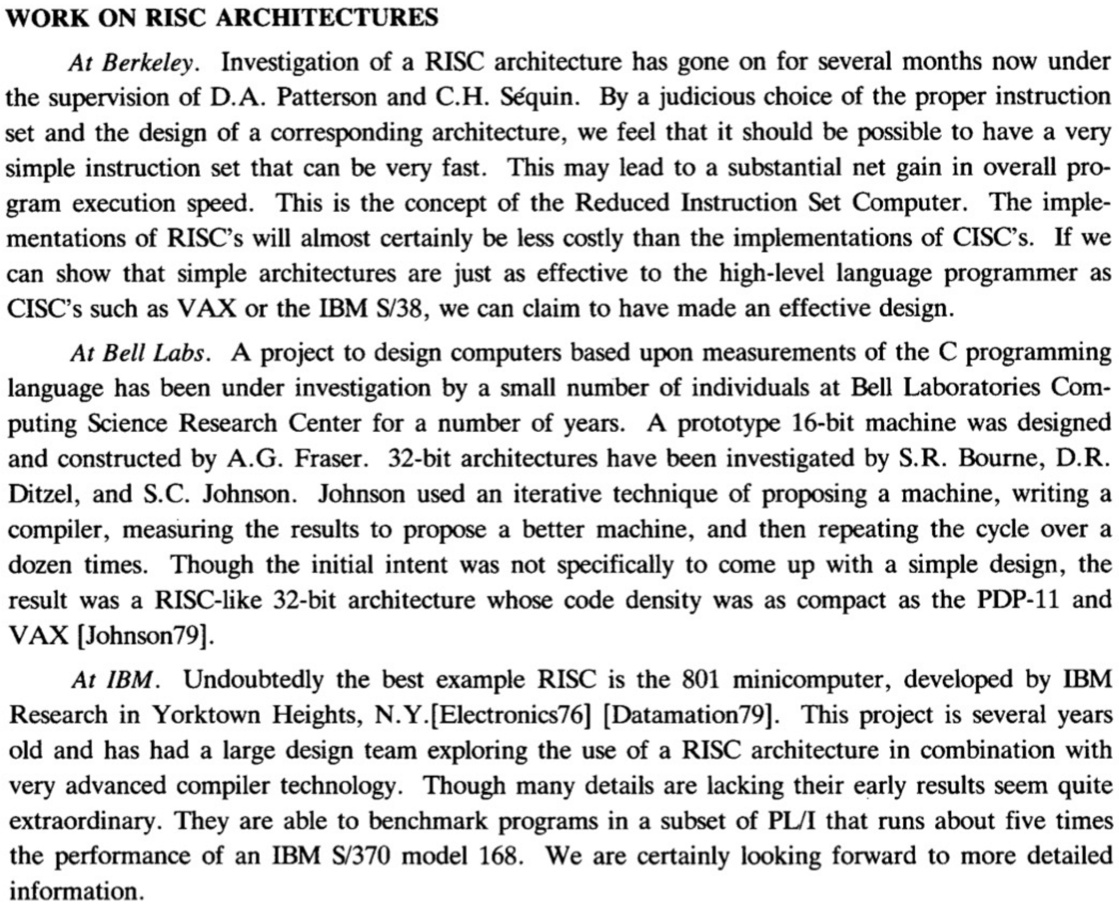
Computer Science Division
University of California
Berkeley, California 94720

*David R. Ditzel*

Bell Laboratories
Computing Science Research Center
Murray Hill, New Jersey 07974

### INTRODUCTION

One of the primary goals of computer architects is to design computers that are more cost-effective than their predecessors. Cost-effectiveness includes the cost of hardware to manufacture the machine, the cost of programming, and costs incurred related to the architecture in debugging both the initial hardware and subsequent programs. If we review the history of computer families we find that the most common architectural change is the trend toward ever more complex machines. Presumably this additional complexity has a positive tradeoff with regard to the cost-effectiveness of newer models. In this paper we propose that this trend is not always cost-effective, and in fact, may even do more harm than good. We shall examine the case for a Reduced Instruction Set Computer (RISC) being as cost-effective as a Complex Instruction Set Computer (CISC). This paper will argue that the next generation of VLSI computers may be more effectively implemented as RISC's than CISC's.



### WORK ON RISC ARCHITECTURES

*At Berkeley*. Investigation of a RISC architecture has gone on for several months now under the supervision of D.A. Patterson and C.H. Séquin. By a judicious choice of the proper instruction set and the design of a corresponding architecture, we feel that it should be possible to have a very simple instruction set that can be very fast. This may lead to a substantial net gain in overall program execution speed. This is the concept of the Reduced Instruction Set Computer. The implementations of RISC's will almost certainly be less costly than the implementations of CISC's. If we can show that simple architectures are just as effective to the high-level language programmer as CISC's such as VAX or the IBM S/38, we can claim to have made an effective design.

*At Bell Labs*. A project to design computers based upon measurements of the C programming language has been under investigation by a small number of individuals at Bell Laboratories Computing Science Research Center for a number of years. A prototype 16-bit machine was designed and constructed by A.G. Fraser. 32-bit architectures have been investigated by S.R. Bourne, D.R. Ditzel, and S.C. Johnson. Johnson used an iterative technique of proposing a machine, writing a compiler, measuring the results to propose a better machine, and then repeating the cycle over a dozen times. Though the initial intent was not specifically to come up with a simple design, the result was a RISC-like 32-bit architecture whose code density was as compact as the PDP-11 and VAX [Johnson79].

*At IBM*. Undoubtedly the best example RISC is the 801 minicomputer, developed by IBM Research in Yorktown Heights, N.Y.[Electronics76] [Datamation79]. This project is several years old and has had a large design team exploring the use of a RISC architecture in combination with very advanced compiler technology. Though many details are lacking their early results seem quite extraordinary. They are able to benchmark programs in a subset of PL/I that runs about five times the performance of an IBM S/370 model 168. We are certainly looking forward to more detailed information.

# Early RISC Principles

**RISC: Reduced Instruction Set Computer**

- RISC and CISC terms coined by UC Berkeley professors Dave Patterson and Carlo Sequin

**Simple orthogonal instructions**

- Often fixed 32-bit length
- Easy for compiler code generation
- Easy to implement with small (10-20) gates per cycle
- Easily pipelined

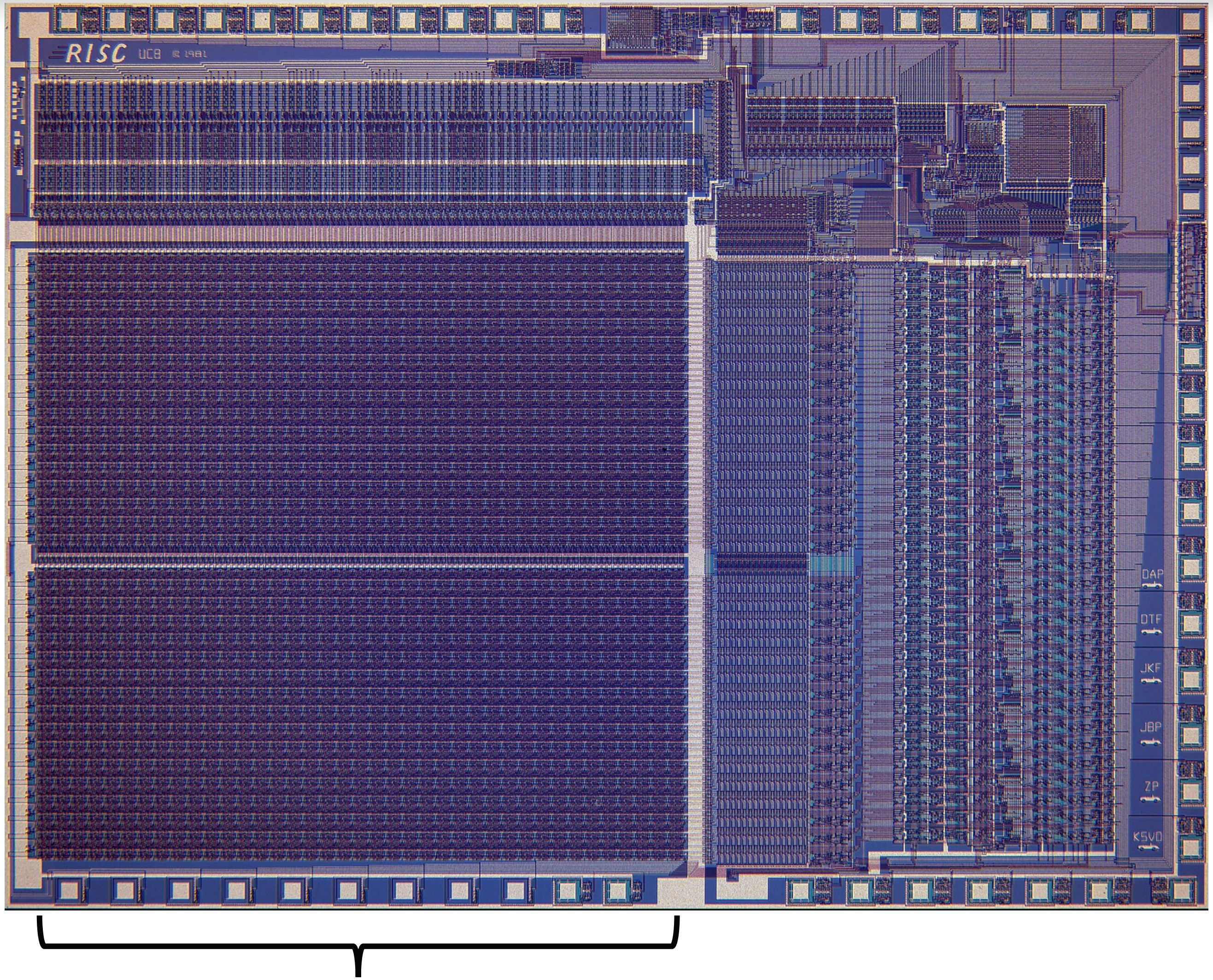
**General purpose register file**

- Enough to keep data in registers for current and another called subroutine
- Variation in register file style: Flat, Windowed, or invisible (stack cache)

**Single load or store per instruction**

**Register to register arithmetic operations**

# UC Berkeley RISC-I ~1981



Register Windows

Most of die consumed with register windows

More registers to keep more operands on die

Single load or store to/from a register

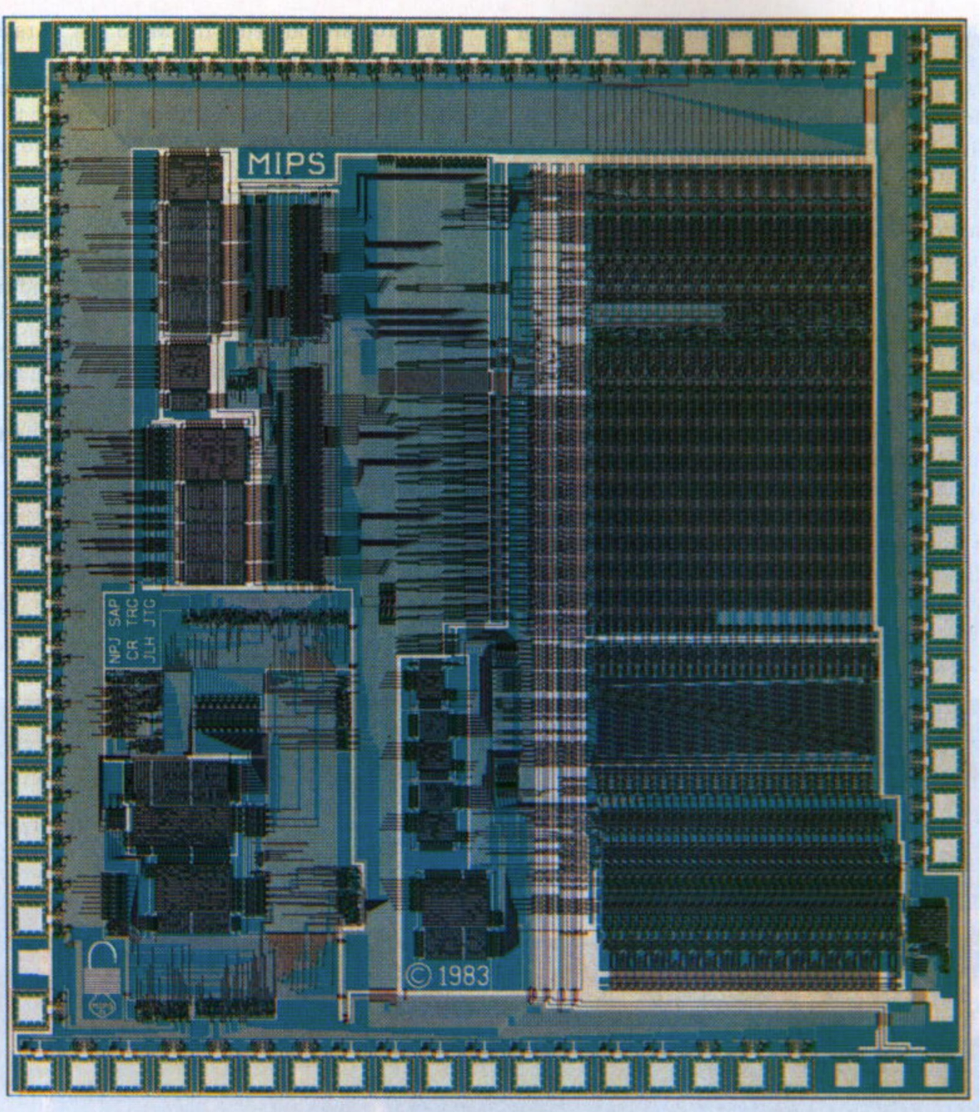
3-address register to register operations

Integer only

No caches

But simple enough for students to design!

# Stanford MIPS ~1983



The MIPS instruction set consists of about 111 total instructions

Each instruction encoded in 32-bits

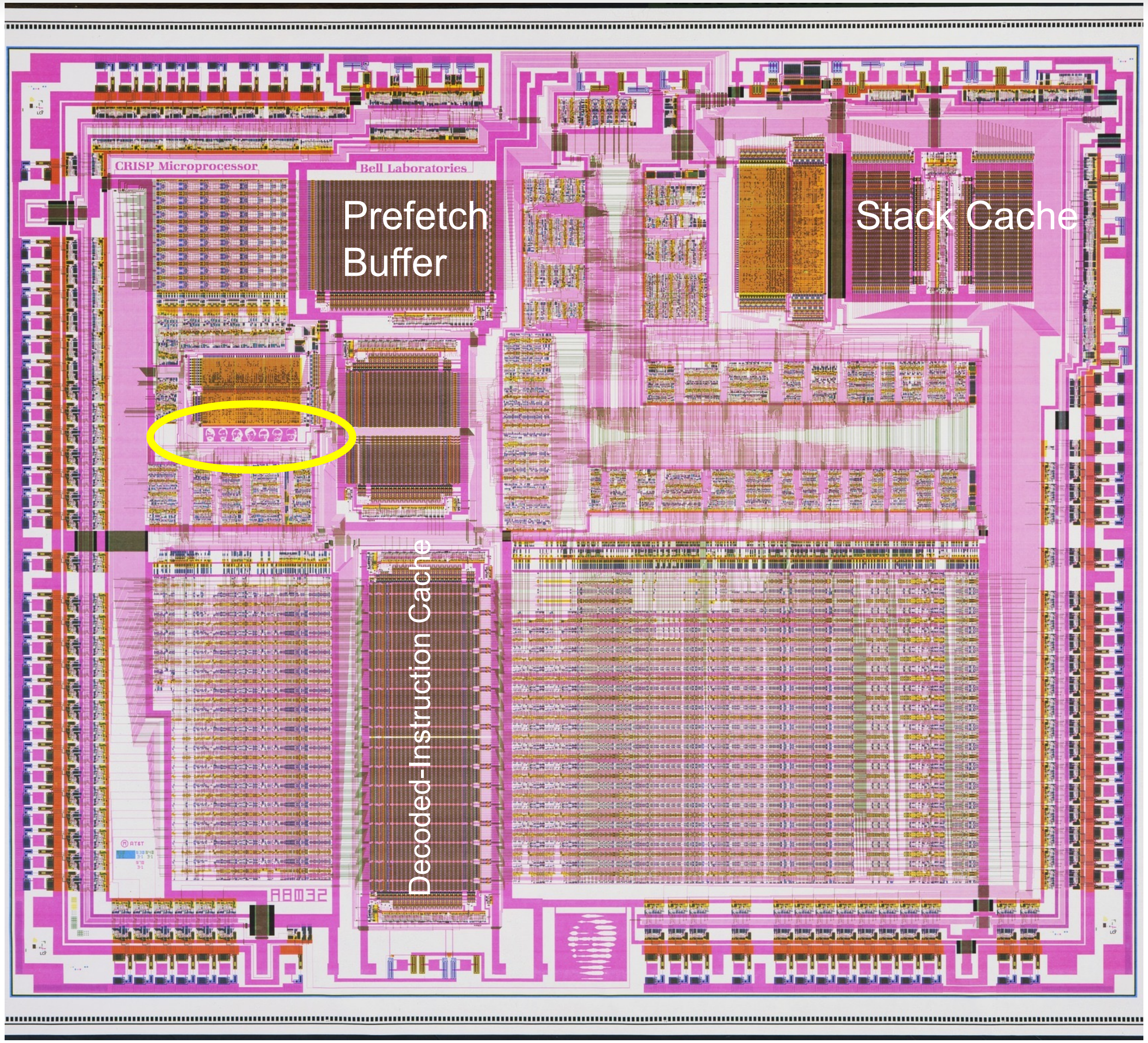
Pipelined to execute one instruction per clock

The instruction set includes:

- 21 arithmetic instructions (+, -, *, /, %)
- 8 logic instructions (&, |, ~)
- 8 bit manipulation instructions
- 12 comparison instructions (>, <, =, >=, <=, ¬)
- 25 branch/jump instructions
- 15 load instructions
- 10 store instructions
- 8 move instructions
- 4 miscellaneous instructions

Delayed branch with 2 delay slots

# AT&T Bell Labs CRISP: C language Reduced Instruction Set Processor



C-Language Reduced Instruction Set Processor

1.75 micron CMOS

Announced 1987

1st CMOS superscalar (more than 1 instr/clock) chip

Hardware translated compact instructions into 180-bit wide decoded micro-Op cache

Branch folding eliminated the overhead of branches

“Stack Cache” performed automatic register allocation in hardware

Low power chip, used in EO Tablet computer

Software binary translator used to move software from CISC WE32100

Lessons

- External to internal translation worked well
- Decouple external and internal ISA

# So why did Esperanto pick RISC-V and why you should too

Esperanto initially had its own ”universal” instruction set, but we switched to RISC-V as it gained momentum

- We needed freedom to use and innovate with an ISA

RISC-V scalar ISA not that different from early RISC designs

- UC Berkeley team did a good job of taking the best from prior RISC chips to create RISC-V
- Notably missing: delayed branches, fixed length instructions (mix of 16/32-bit), no register windows
- Now the committees are adding features so complexity will grow... but still cleaner than alternatives.

The real IMPORTANT reasons to use RISC-V

- RISC-V is the only free and open instruction set that currently has critical mass of adoption
  - Intel and ARM work to keep non-licensed outsiders from creating new CPU implementations
  - RISC-V is where the innovation will occur
- RISC-V allows and encourages innovation with both private and official extensions
- RISC-V has a cleaner instruction set that enables simpler more energy efficient implementations
- RISC-V is good enough that inventing yet another RISC instruction set doesn’t make sense
- For anyone wanting to do a new CPU, RISC-V is not merely the best choice, its the only practical choice.

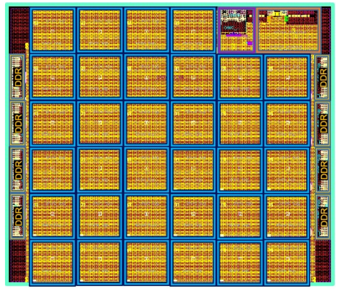
esperanto.ai

# Esperanto’s CPU Roadmap
# 7nm
# 4nm
# 2nm

**November 2024**
**Subject to change**

# Silicon Roadmap from SoC to Chiplets

## ET-SoC-1 7nm

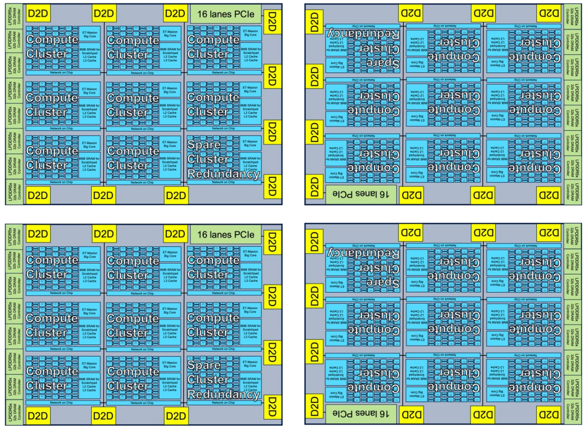


Product shipping today

7nm Single chip SoC 15-60W
1K ET-Minion1 RISC-V cores
100 TOp Int8
25 TFlops FP16
12.5 TFlops FP32

PCIe Gen4
Up to 32GB LPDDR4x DRAM

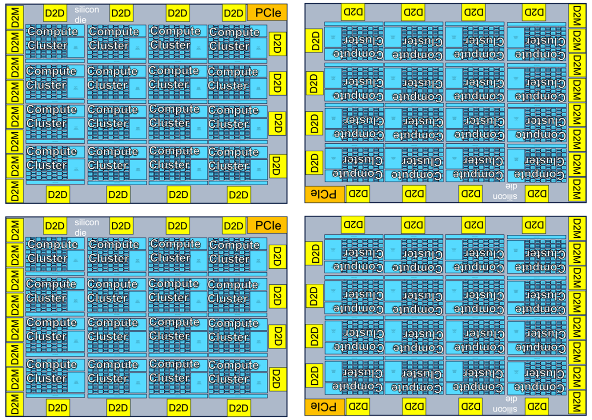
## ET-SoC-2x 4nm



Production 2026

4nm Compute Fabric Chiplet 15-60W
256 ET-Minion2 CPU/Chiplet @2 GHz
256 TFlops FP8/Int8 per chiplet
128 TFlops FP16 per chiplet
32 TFlops FP32 per chiplet
16 TFlops FP64 per chiplet
256 to 2048 RISC-V cores per package
Up to 8 ET-SoC-2x chiplets per package

## ET-SoC-3x 2nm



Production 2027

2nm Compute Fabric Chiplet 10-60W
512 ET-Minion3 CPU/Chiplet @2GHz
512 TFlops FP8/Int8 per chiplet
256 TFlops FP16 per chiplet
64 TFlops FP32 per chiplet
32 TFlops FP64 per chiplet
512 to 4096 RISC-V cores per package
Up to 8 ET-SoC-3x chiplets per package

# ET-SoC-1

# 7nm

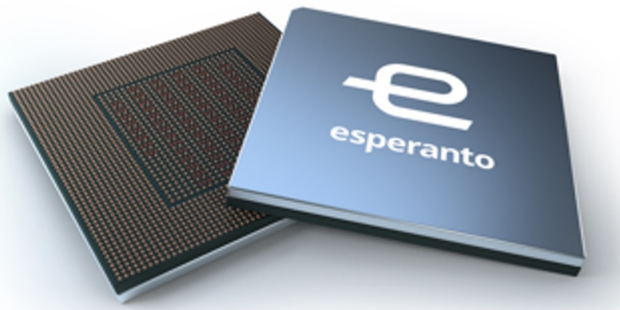
## Intended for Machine Learning Inference

## Protyping low power advances

# ET-SoC-1: Esperanto’s RISC-V Supercomputer on a Chip

esperanto.ai

## Over 1,000 64-bit RISC-V CPUs per Chip



As low as 15W per SoC
(workload dependent)

High bandwidth interconnect network

Dynamic, tiered architecture of 160 MB on-die SRAM for caches

Up to 32GB LPDDR4x DRAM

4 ET-Maxion high-performance OOO RISC-V cores

TSMC 7nm

High efficiency operation (inferences/sec/watt)

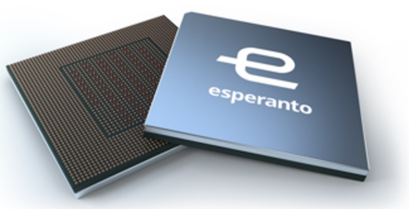
**Accelerates wide range of AI/ML workloads**
- Language Models (NLP/LLM)
- Visual Models (Detection, Segmentation)
- Recommendation Models (RecSys, DLRM)
- Generative AI – (Diffusion, Video semantic analysis)

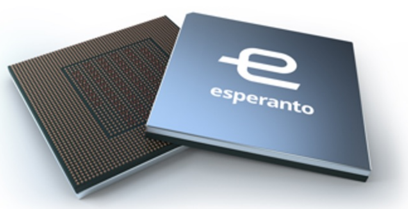
Allows flexible general-purpose computing

Enables pre- and post-processing

Highly efficient HPC workload through massive parallelism

# ET-SoC-1: 7nm First Generation Customer Roadmap





| Key Parameters | ET-SoC-1: 600 MHz | ET-SoC-1: 800 MHz |
|---|---|---|
| Typical application | • AI inference<br>• HPC (Single precision) | • AI inference<br>• HPC (Single precision) |
| Int8 Peak TOps | 77 | 102 |
| FP16 Peak TFlops | 19 | 26 |
| FP32 Peak TFlops | 10 | 13 |
| Max DRAM Capacity | 32 GB | 32 GB |
| Memory Bandwidth | 120 GB/s | 137 GB/s |
| Typical Power running ML | < 30 Watts | < 40 Watts |
| Sample date | Shipping Now | now |
| FCS | Shipping Now | 2025 |
| FP16 Relative Performance including software improvements | 1x | 1.4x |

# ET-Minion1 is an Energy-Efficient RISC-V CPU with a Vector/Tensor Unit

*CPU is tailored for Massively Parallel ML Applications*

## ET-MINION1 IS A CUSTOM BUILT 64-BIT RISC-V PROCESSOR

- In-order pipeline with low gates/stage to improve MHz at low voltages
- Architecture and circuits optimized to enable low-voltage operation
- Two hardware threads of execution
- Software configurable L1 data-cache and/or scratchpad

## ML OPTIMIZED VECTOR/TENSOR UNIT

512-bit wide integer per cycle

- 128 8-bit integer operations per cycle, accumulates to 32-bit Int

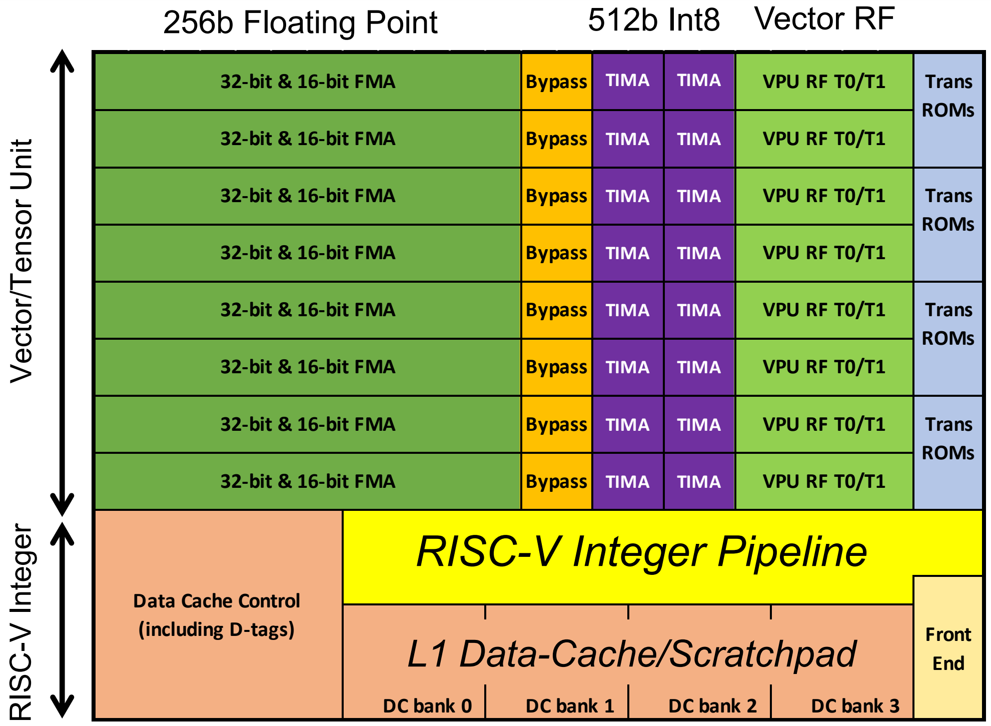
256-bit wide floating point per cycle

- 16 32-bit single precision operations per cycle
- 32 16-bit half precision operations per cycle

New multi-cycle Tensor Instructions

- Can run for up to 512 cycles (up to 32K operations) with one tensor instruction
- Reduces instruction fetch bandwidth and reduces power
- RISC-V integer pipeline put to sleep during tensor instructions

Vector transcendental instructions



*ET-Minion1 RISC-V Core and Tensor/Vector unit optimized for low-voltage operation*
to improve energy-efficiency

Optimized for energy-efficient ML operations. Each ET-Minion1 can deliver peak of 128 Int8 GOPS per GHz

# 32 RISC-V ET-Minion CPUs and 4 MB Memory Form a “Minion Shire”

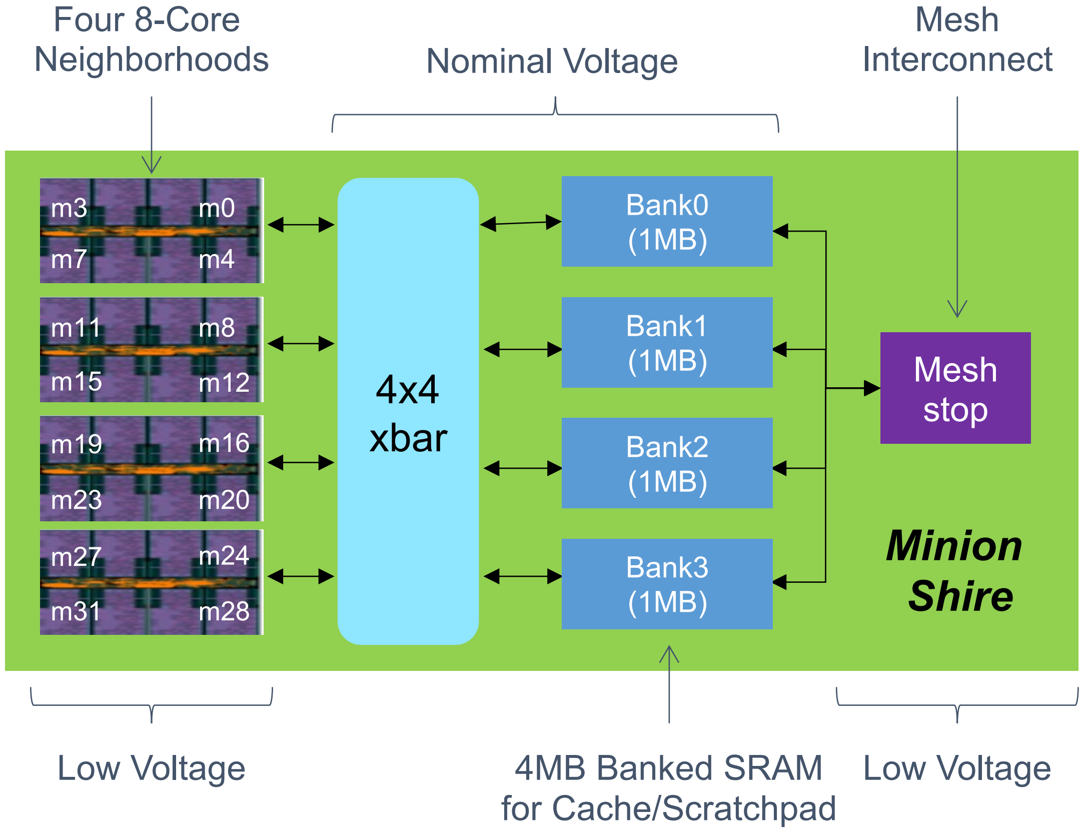
## 32 ET-Minion RISC-V cores per Minion Shire

- 64-bit RISC-V integer CPU cores each with 512-bit vector/tensor units
- Arranged in four 8-core neighborhoods

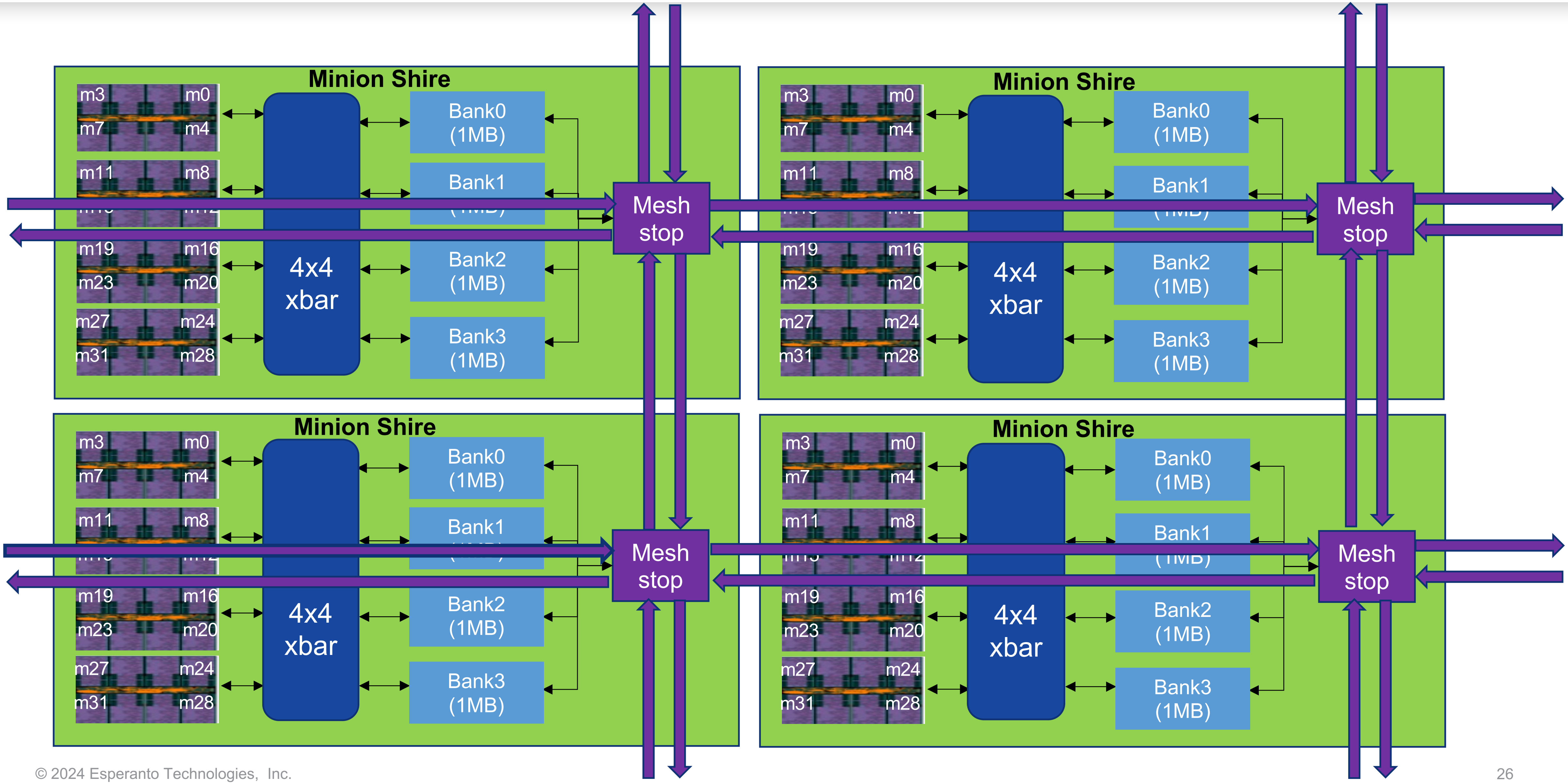
## Software configurable memory hierarchy

- L1 data cache can also be configured as scratchpad
- Four 1MB SRAM banks can be partitioned as private L2, shared L3 and scratchpad

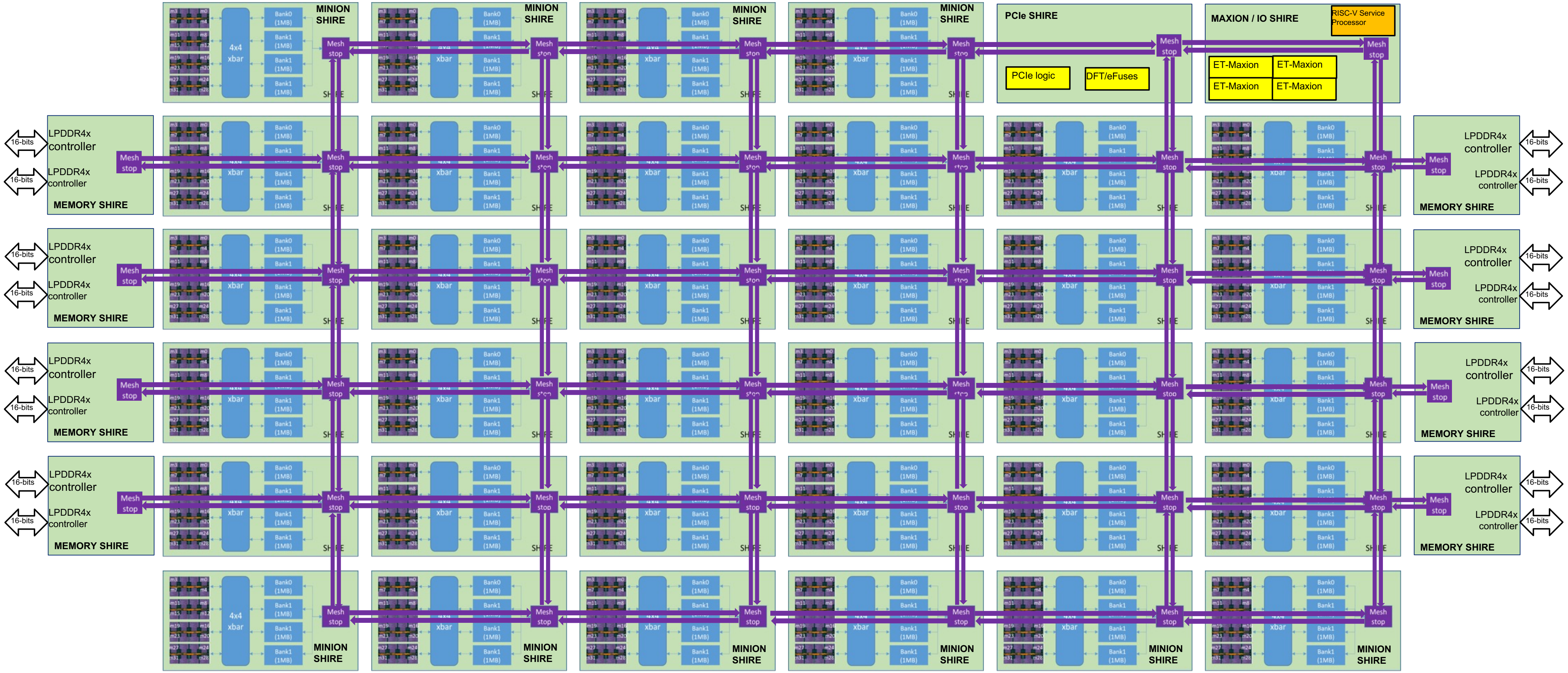
## Shires connected with mesh network-on-chip



# Shires are connected to each other and to external memory through Mesh Network



# ET-SoC-1: Compute Fabric with Over a Thousand RISC-V CPU Cores



**34 MINION SHIRES**

- **1088 ET-Minions**

**8 MEMORY SHIRES**

- **LPDDR4x DRAM controllers**
- **up to 32 GB DRAM**

**1 MAXION / IO SHIRE**

- **4 ET-Maxions**
- **1 RISC-V Service Processor**

**PCIe SHIRE**

**160 million bytes of on-die SRAM**

**8 lanes of PCIe Gen4**

**Secure Root of Trust**

# ET-SoC-1 Physical Design

## The ET-SoC-1 is fabricated in TSMC 7nm

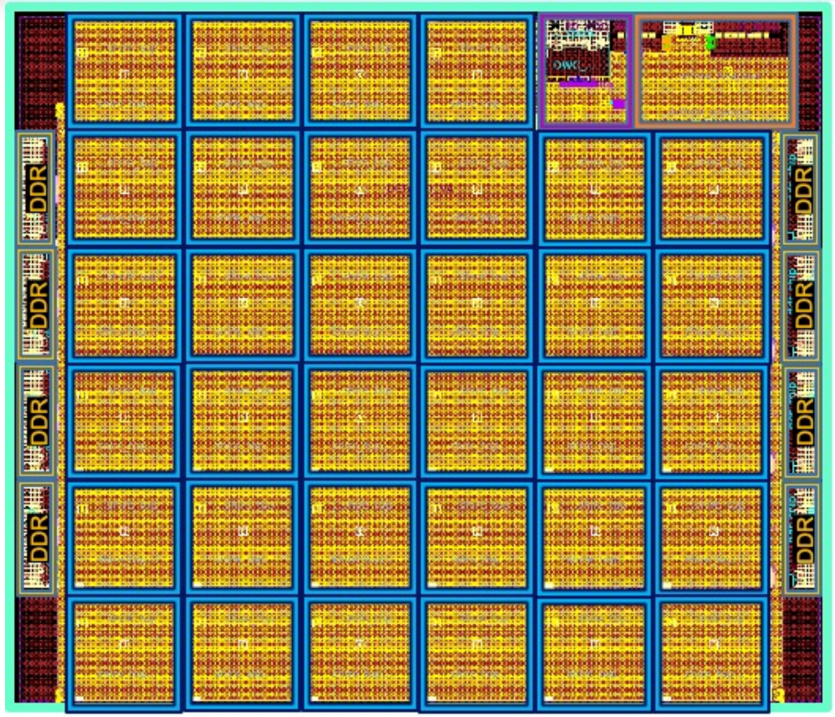
- 24 billion transistors
- Die-area: 570 mm$^2$
- Low power: adjustable between 15 to 60 watts

## Energy-efficient 64-bit RISC-V processors

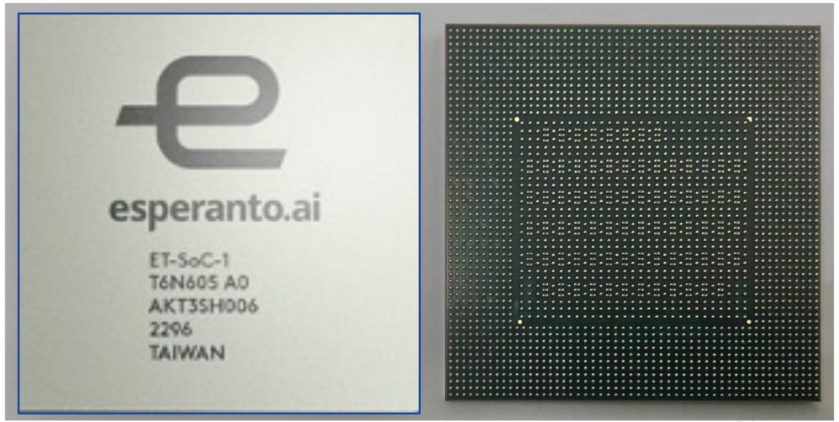
- 1088 “ET-Minion” 64-bit RISC-V processors; vector/tensor unit
- 4 “ET-Maxion” 64-bit high-performance RISC-V out-of-order processors
- 1 64-bit RISC-V service processor

## Other highlights

- 160MB of on-die SRAM used for caches and scratchpad memory
- Root of trust for secure boot
- Package: 45x45mm with 2494 balls to PCB, over 30,000 bumps to die



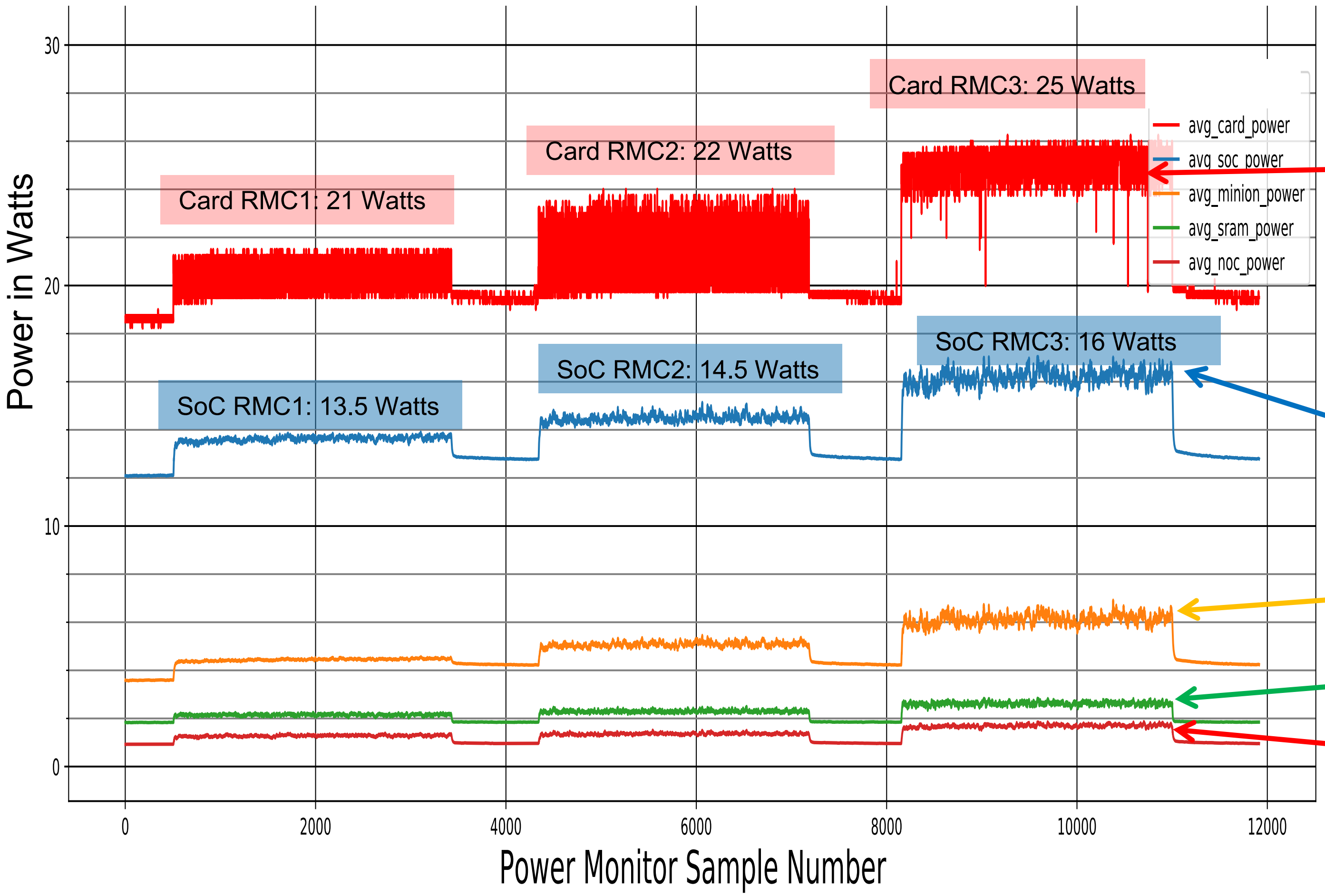
ET-SoC-1 Die Plot



ET-SoC-1 Package

# Example Card and ET-SoC-1 Power on ML Recommendation Benchmarks

DLRM FP16 RMC1, RMC2, RMC3



Power will vary depending on benchmark and operating conditions, but here is recently measured data.

Total PCIe card power under 25 watts for DLRM

- ET-SoC-1
- Eight  LPDDR4x 32-bit wide DRAM die
- Flash memory and other misc components
- Power supplies

ET-SoC-1 power about 13.5 to 16 watts

A thousand 600 MHz ET-Minion RISC-V processors: 6 watts

128 MB of on-die SRAM for caches: 2.8 watts

44 Network-on-Chip connecting compute shires: 1.8 watts

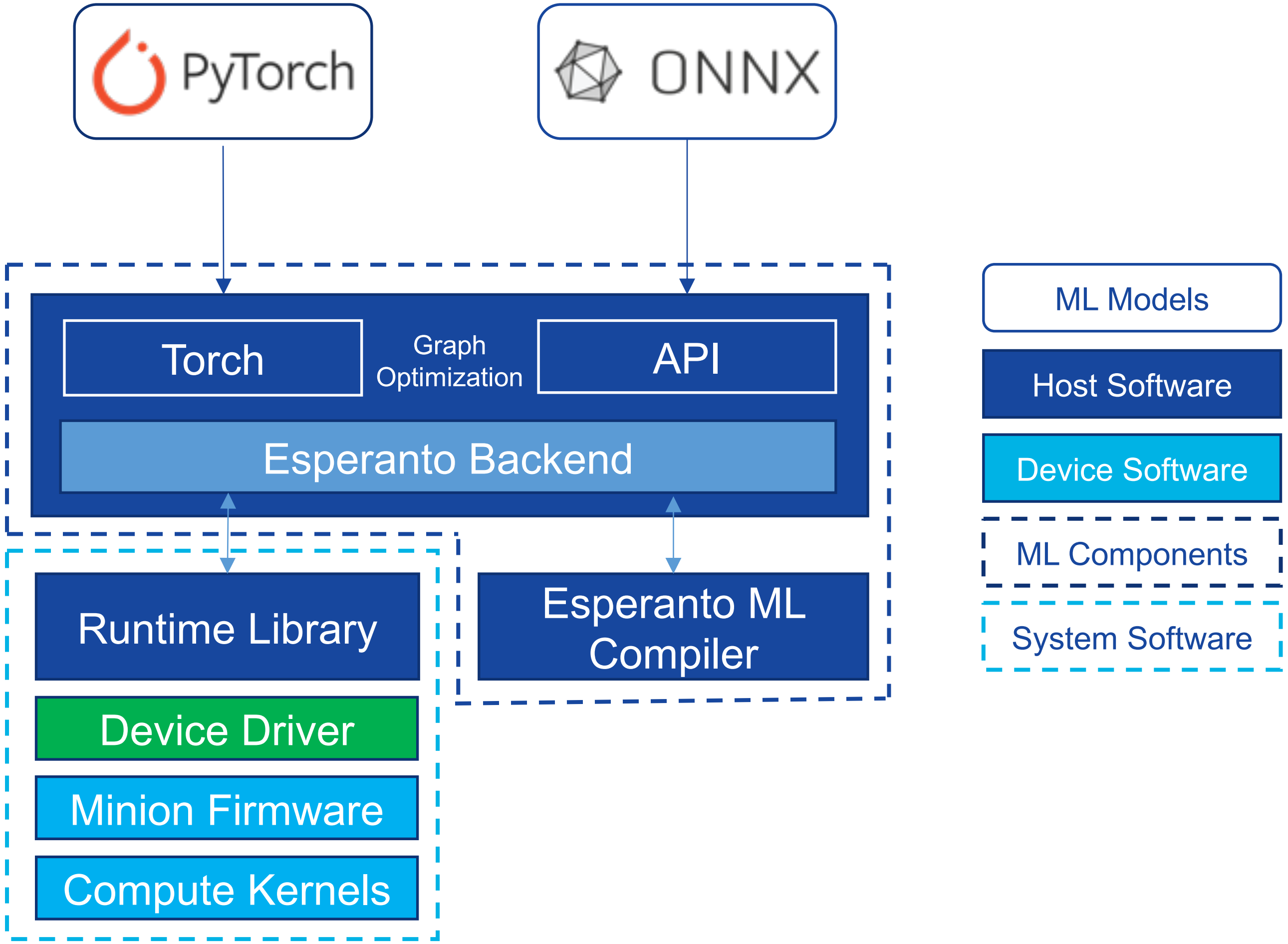
# The Esperanto ML Software Stack

**Delivered and integrated with each AI / HPC compute server**

**The ML framework compiler accepts a trained model expressed at a high level – PyTorch or TensorFlow (via ONNX) – and generates RISC-V executable code to run on ET-SoC-1(s)**

**Esperanto's ML stack performs device-independent graph optimization and then the Esperanto Backend perform device-specific optimization**

**Esperanto continues to adapt to market trends, new model directions and open-source software evolution**



# Esperanto General Purpose Software Development Kit (GP-SDK)

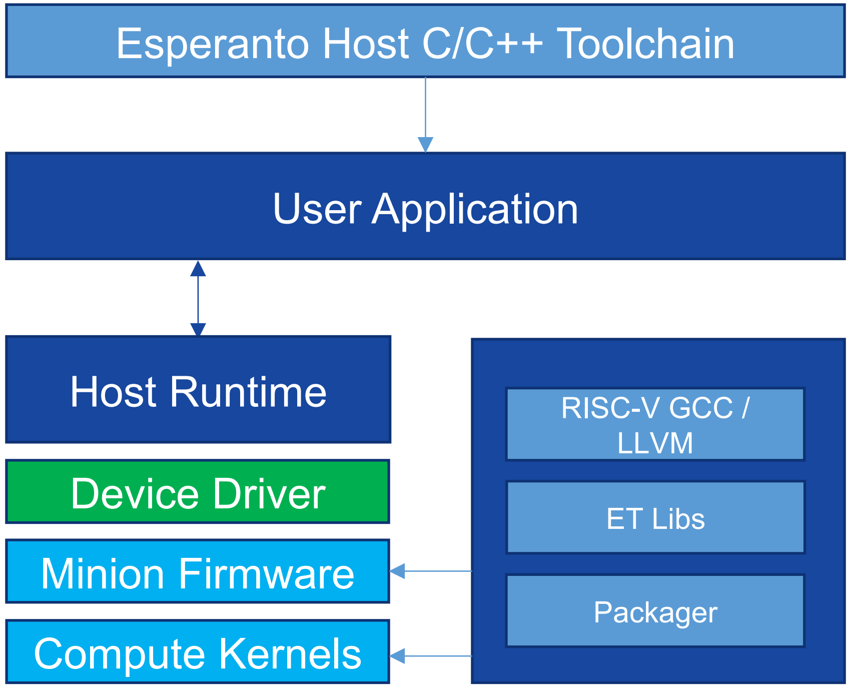
**Comes integrated with each evaluation server**

**Enables general purpose parallel programming of 1024 64-bit ET-Minion cores**

- All Minions can be individually programmed and can share common data address space
- Great for DSP and other highly parallel compute acceleration applications

**x86 host-based user application is compiled using Esperanto's Host C/C++ Toolchain**

**ET-Minion compute kernels and firmware are compiled using the RISC-V toolchain**



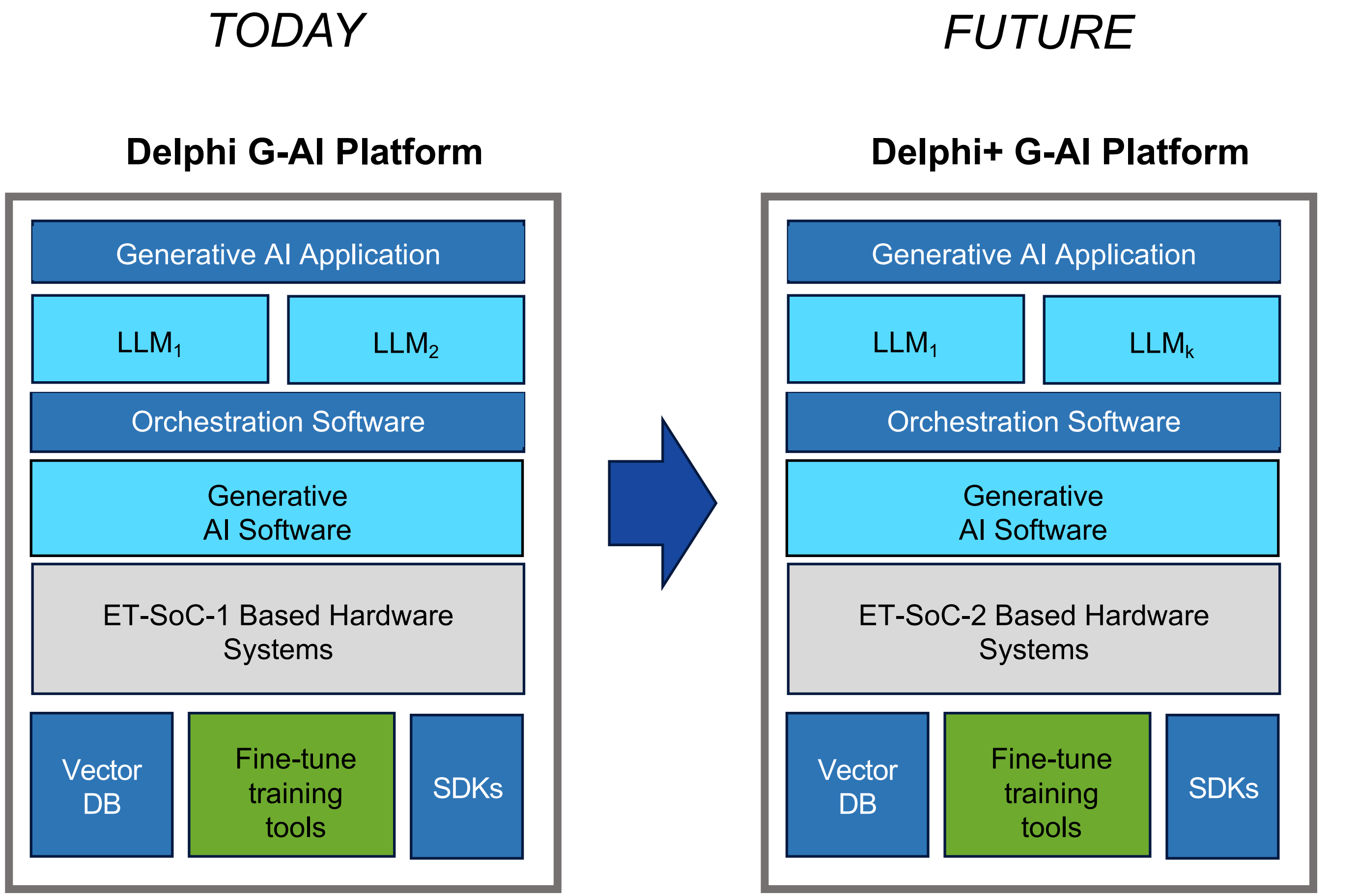
# Esperanto Generative AI Strategy

- **Esperanto strategy is focused on targeting ET-SoC-1 based systems with open-source models to bring differentiated value to vertical markets – Faster time to value and lower TCO**

- **Leverage Esperanto's technology for compute efficiency (low power) and scale-out computing**

- **Future support of MoE LLMs for inference as based model size is projected to increase beyond 1T parameters[1]**

[1] Note: Nvidia's recent Blackwell based GPU systems are targeted at enabling training of 1T+ MoE LLMs



- Vertical market specific tuning
- Dense open-source LLMs
- ET-SoC-1 hardware
- Edge and small datacenter

- Vertical market specific tuning
- Dense and MoE (Sparce) LLMs
- ET-SoC-2 hardware
- Edge to large datacenter

# Generative AI Language Models Supported by Esperanto

| Use Case | LLM Model | Parameters | Precision | Open-sourced since | ET Supported since |
|---|---|---|---|---|---|
| Text-to-text | Llama2, Vicuna | 7B, 13B | fp16, int4 | April 2023 | September 2023 |
| | Mistral | 7B | fp16, int4 | September 2023 | November 2023 |
| | Llama3.1 | 8B | fp16, int4 | July 2024 | August 2024 |
| | Phi-3.5 | Mini | fp16 | August 2024 | August 2024 |
| | Gemma | 2B, 7B | fp16 | February 2024 | June 2024 |
| | Mixtral | 8x7B | int4 | January 2024 | August 2024 |
| Text-to-code | Starcoder | 15B | fp16 | May 2023 | November 2023 |
| Text-to-image | OpenJourney | | fp16, fp32 | November 2022 | September 2023 |
| | Stable Diffusion | 1.5, 2.1 | fp16 | December 2022 | November 2023 |
| Image-to-text | LLAVA | 1.5 | fp16, int4 | October 2023 | July 2024 |
| Speech-to-text | Whisper | V3 Medium, Large | fp16 | November 2023 | July 2024 |

# ET-SoC-1 PCIe Production Card

**PCIe Gen4 8-lane low-profile form factor**

**ET-SoC-1 and 32 GB LPDDR4x memory**

**Safety and emissions certification for multiple geographies**

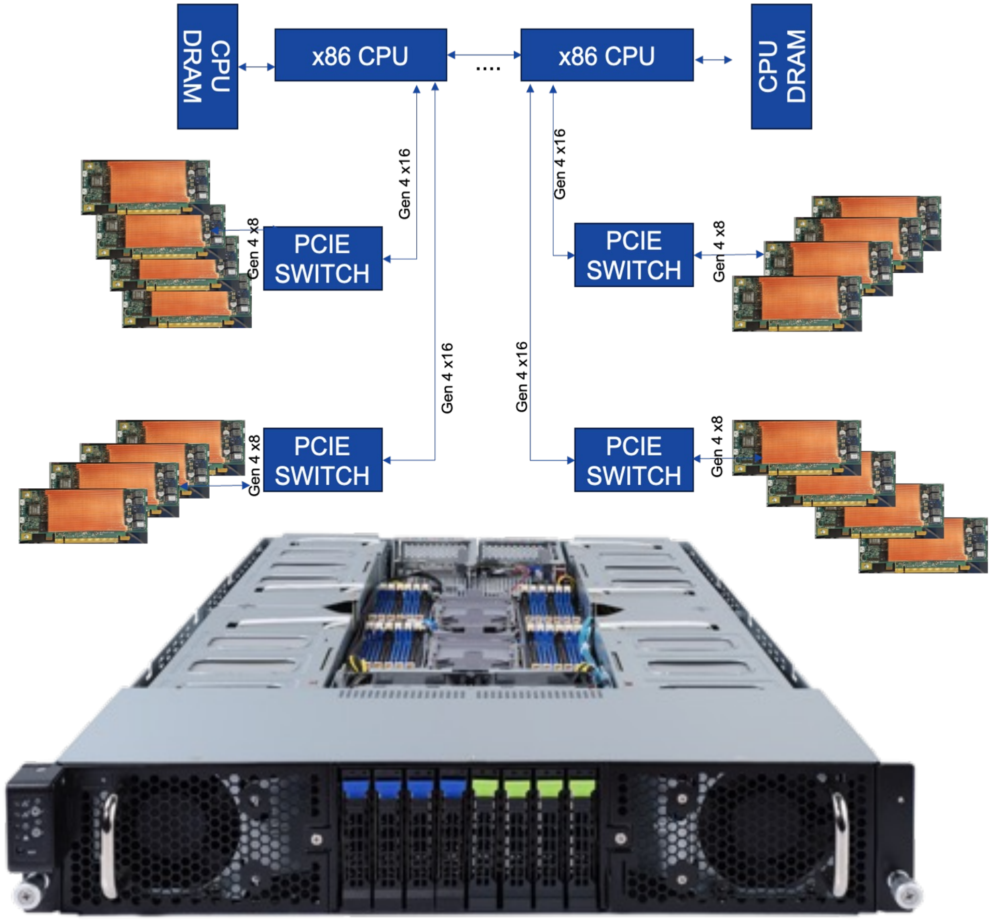
**Developed, tested, certified and delivered by Esperanto's partner Penguin Solutions / Smart Global Holdings (SGH)**

**Implemented across various server form factors**

- Single card (ruggedized edge server)
- Up to 6 cards (short-form 2U server for enterprise edge)
- Up to 16 cards (ultra-high density 2U server for data center)

**Penguin Computing production ET-SOC-1 PCIe card with and without heatsink**

# Esperanto’s ET-SoC-1 RISC-V Production System – Shipping Now



**2U server with up to 16 Esperanto Accelerator Cards**

| | | |
|---|---|---|
| Target environment | Data center | |
| Server configuration | Standard 2U 19” rack-mount chassis | |
| ET-SoC-1 PCIe cards (Gen4x8) | 8 Cards | 16 Cards |
| System host processor | 2x Intel Xeon® Gold 6326 16-core | 2x Intel Xeon Platinum 8358P 32-core |
| System memory | 512GB DDR4-3200 | 1TB DDR4-3200 |
| Storage | 2x Samsung® PM9A3 3.84TB NVMe U.2 SSDs | |
| ET-SoC-1 frequency | 600 MHz | |
| ET-SoC-1 power consumption | 15W to 60W (workload dependent) | |
| ET-SoC-1 RISC-V CPUs | 8,448 or 16,896 (high performance, low power) | |
| Pre-installed AI models | LLMs, Computer Vision, Recommendation, Transformer models | |
| ML SDK | Jupyter Notebook and command-line tools | |
| GP SDK | Included | |
| Performance, power, and trace analysis tools | Included (et-powertop, Perfetto) | |
| Training and documentation | Included | |
| Connectivity | 2 x 10Gb/s LAN ports built in, 2 free PCIe Gen4 x8 slots | |

# Example ET-SoC-1 Rack Implementations

esperanto.ai

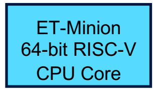
|  | Low Power | High Density | Ultra High Density | Notes |
|---|---|---|---|---|
| ET chip | ET-SoC-1 | ET-SoC-1 | ET-SoC-1 |  |
| Enclosure | GigaByte G292 | GigaByte E252 | GigaByte G292 | US country of origin |
| Host | x86 | x86 | x86 | "Self hosting" optional |
| PCIe | Gen 4 | Gen 4 | Gen 4 |  |
| Server form factor | 2U | 2U | 2U |  |
| # of cards / server | 8 | 7 | 16 | Customizable |
| # of server / rack | 10 | 20 | 20 | Customizable |
| # of 64bit RISC-V cores | 80,000+ | 140,000+ | 320,000+ | Low power cores |
| Vector / Tensor | Yes | Yes | Yes |  |
| Rack power | 8 kW* | 16 kW** | 35kW** | * measured average; DLRM @ FP16;<br>** estimated peak |
| Int8 | 6 PetaOps | 10.5 PetaOps | 24 PetaOps | Est. peak |
| FP16 | 1.5 PetaFlops | 2.6 PetaFlops | 6 PetaFlops | Est. peak |
| FP32 | 0.75 PetaFlops | 1.3 PetaFlops | 3 PetaFlops | Est. peak |
| Interconnect | Ethernet / InfiBand | Ethernet / InfiBand | Ethernet / InfiBand | Customizable |
| Software | Dual SDKs included | Dual SDKs included | Dual SDKs included | AI inference; single precision HPC |
| Pre-installed LLMs | Optional | Optional | Optional | Current: Vicuna, Llama, Mistral, StarCoder, OpenJourney, StableDiffusion, LLaVa |
| Analytics tools | Included | Included | Included |  |
| Support / warranty | Included | Included | Included | Via Smart Global Holdings / Penguin Computing |

**80,000 low power RISC-V processors in Esperanto's Silicon Valley office**

# ESPERANTO´S NEXT GENERATIONS:

## Improvements for an energy-efficient Chiplet Compute Fabric based on the latest RISC-V  Vector Instruction Set for both HPC and AI applications

# Compute Cluster for Parallel Programming (step 1 of 5)



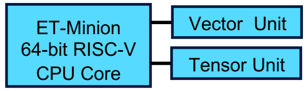
Starts with a

- Custom ET designed
- 64-bit RISC-V ISA compatible
- General-purpose
- FMA with 2 ops/cycle
- Simple in-order pipeline
- Low area
- Low power
- Dual hardware threads
- Supports a globally shared address space
- Circuits that work at low-Voltage
- Processor Core
- Esperanto calls this the ET-Minion2

With 1 GHz cores

| | |
|---|---|
| Int8: | .002 TOps |
| FP8: | |
| FP16: | |
| FP32: | .002 TFlops |
| FP64: | .002 TFlops |

# Compute Cluster for Parallel Programming (step 2 of 5)



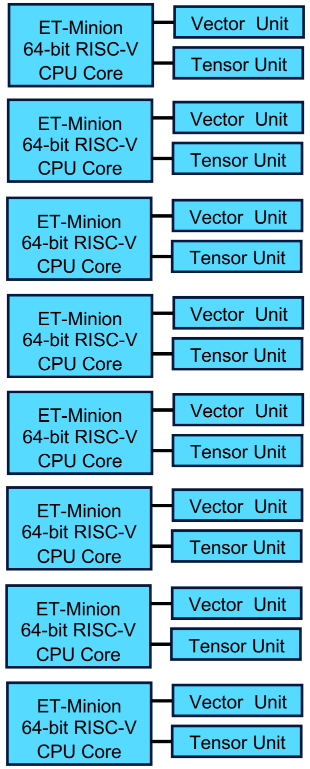
Add a

- RISC-V Vector Unit
  - 512 bits per cycle all RISC-V data types
- Tensor Unit
  - 1024 bits/cycle FP64/FP32
  - 2048 bits/cycle Int8/FP8/FP16

With 1 GHz cores
Int8: 0.512 TOps
FP8: 0.512 TFlops
FP16: 0.256 TFlops
FP32: 0.064 TFlops
FP64: 0.032 TFlops

Tensor FMA operations dominate power and area for ML,
meaning that providing the general-purpose RISC-V ISA has little penalty
if a simple, small and energy efficient design is used.

# Compute Cluster for Parallel Programming (step 3 of 5)



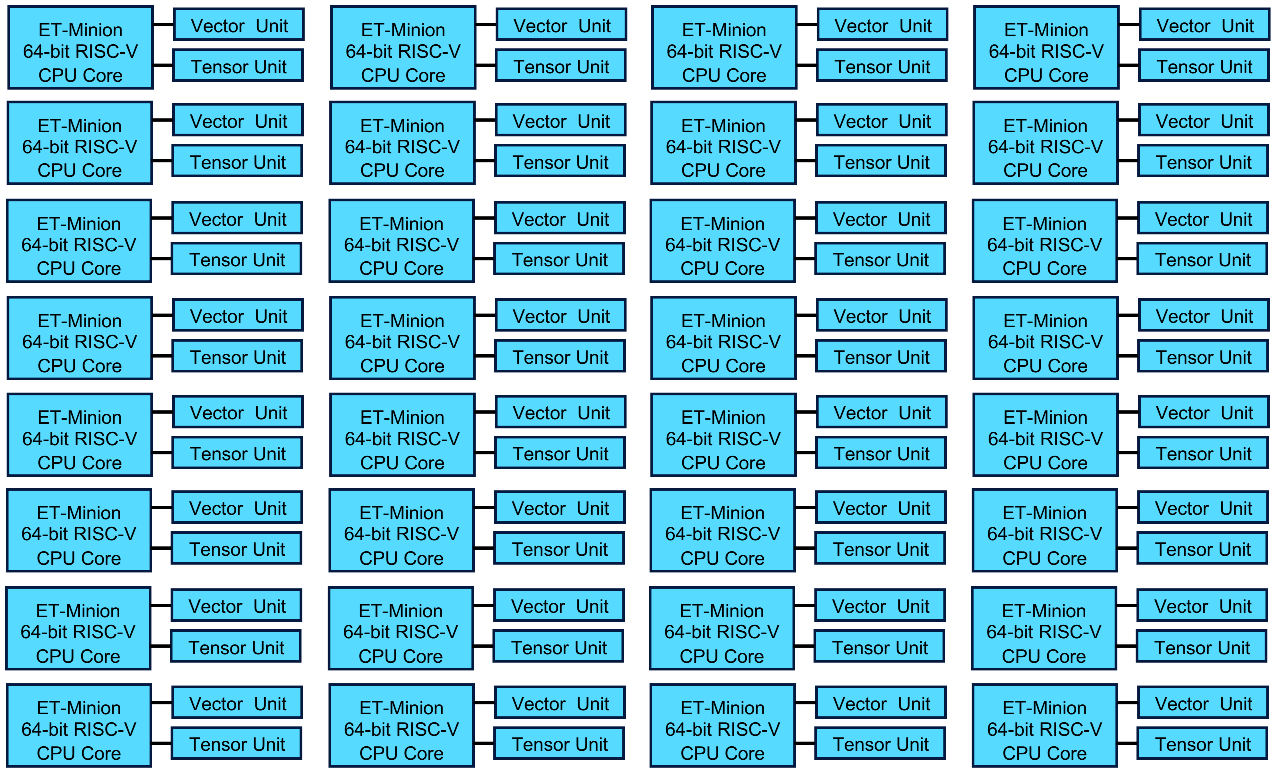
Add 8 ET-Minions each with

- Vector Unit
  - 512 bits per cycle all RISC-V data types
- Tensor Unit
  - 1024 bits/cycle FP64/FP32
  - 2048 bits/cycle Int8/FP8/FP16

With 1 GHz cores
Int8: 4.096 TOps
FP8: 4.096 TFlops
FP16: 2.048 TFlops
FP32: 0.512 TFlops
FP64: 0.256 TFlops

# Compute Cluster for Parallel Programming (step 4 of 5)

Build up to 32 ET-Minion Processors

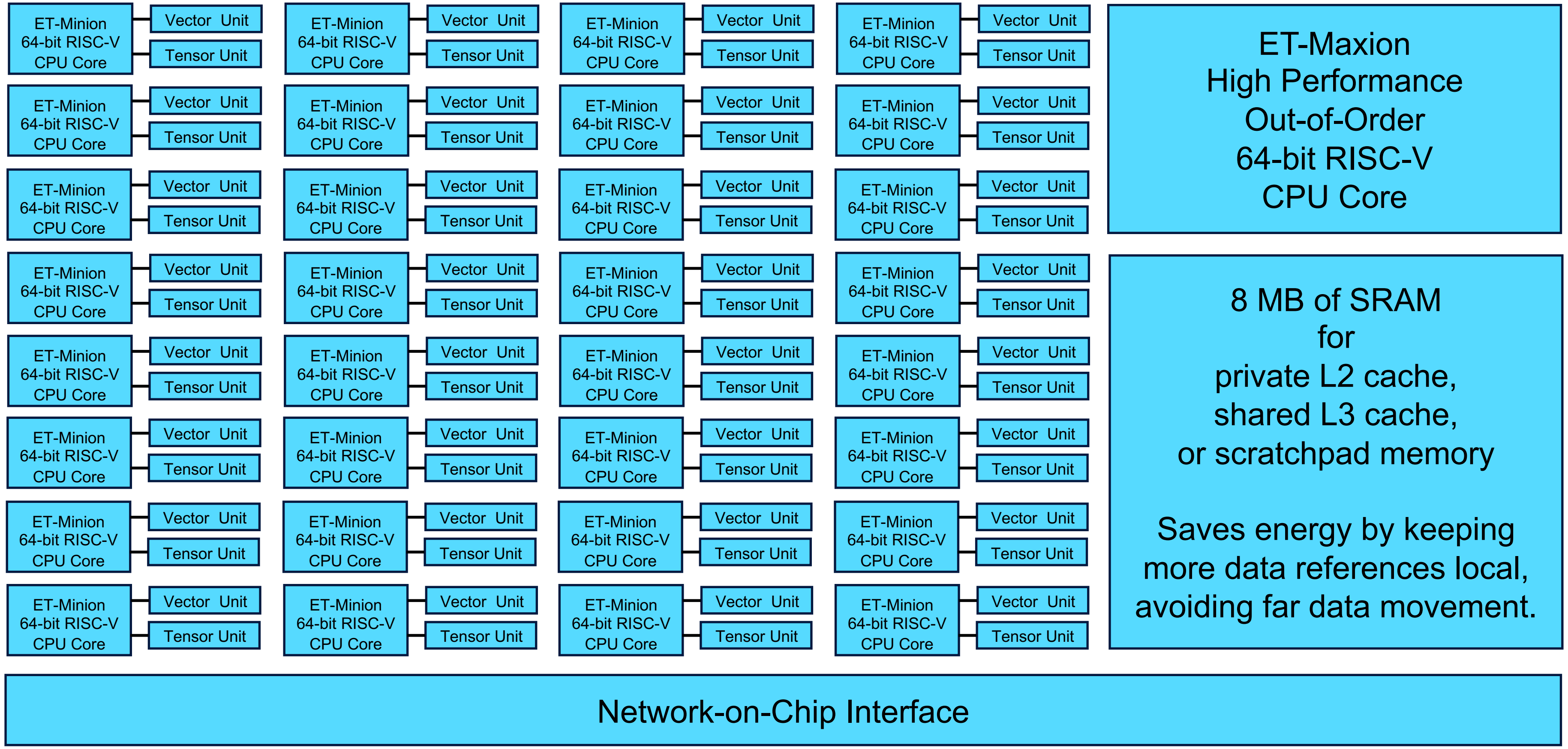

ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit

With 1 GHz cores

| | |
|---|---|
| Int8: | 16.384 TOps |
| FP8: | 16.384 TFlops |
| FP16: | 8.192 TFlops |
| FP32: | 2.048 TFlops |
| FP64: | 1.024 TFlops |

# Compute Cluster for Parallel Programming (step 5 of 5)

Build up to 32 ET-Minion Processors



| | | | |
|---|---|---|---|
| ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit |
| ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit |
| ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit |
| ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit |
| ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit |
| ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit |
| ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit |
| ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit | ET-Minion 64-bit RISC-V CPU Core<br>Vector Unit<br>Tensor Unit |

ET-Maxion
High Performance
Out-of-Order
64-bit RISC-V
CPU Core

8 MB of SRAM
for
private L2 cache,
shared L3 cache,
or scratchpad memory

Saves energy by keeping more data references local, avoiding far data movement.

Network-on-Chip Interface

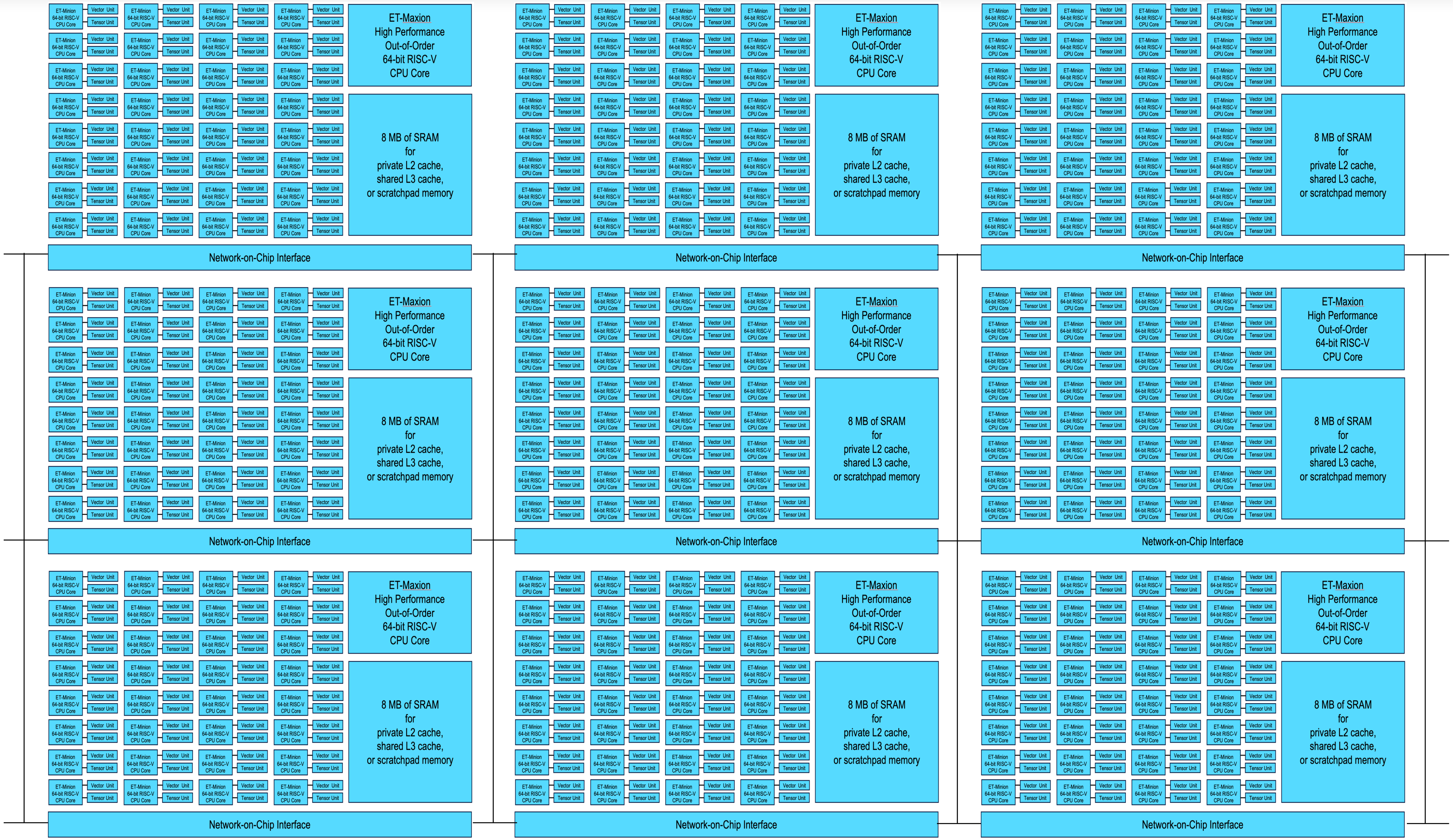
With 1 GHz cores

| | |
|---|---|
| Int8: | 16.384 TOps |
| FP8: | 16.384 TFlops |
| FP16: | 8.192 TFlops |
| FP32: | 2.048 TFlops |
| FP64: | 1.024 TFlops |

This is the cluster architecture for programmers/compiler which is then replicated

# Add compute clusters to form a compute fabric on a die

esperanto.ai


ET-Minion 64-bit RISC-V CPU Core
Vector Unit
Tensor Unit
ET-Maxion High Performance Out-of-Order 64-bit RISC-V CPU Core
8 MB of SRAM for private L2 cache, shared L3 cache, or scratchpad memory
Network-on-Chip Interface

With 1 GHz cores
Int8: 131.1 TOps
FP8: 131.1 TFlops
FP16: 65.5 TFlops
FP32: 16.4 TFlops
FP64: 8.2 TFlops

(note – only using 8 of the clusters, 1 is a spare)

# ET-SoC-2x
# 4nm

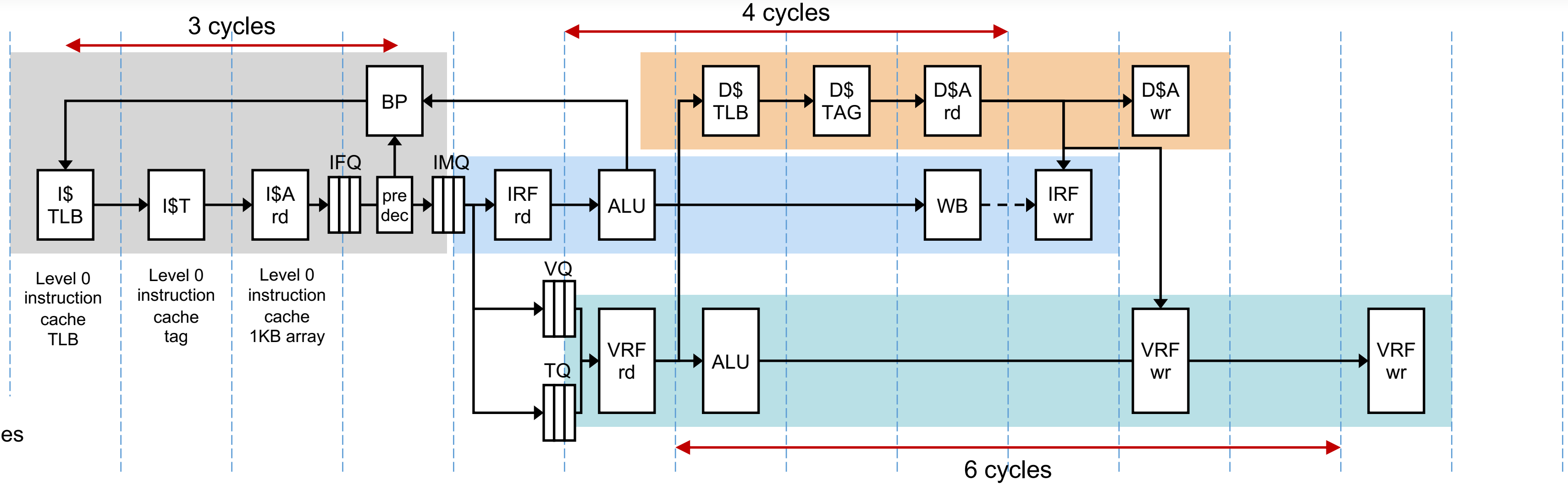
## Add full support for HPC and larger AI tasks

# New ET-Minion2 CPU: Small, low power and high vector/tensor throughput

esperanto.ai

## Minion2 Scalar Pipe

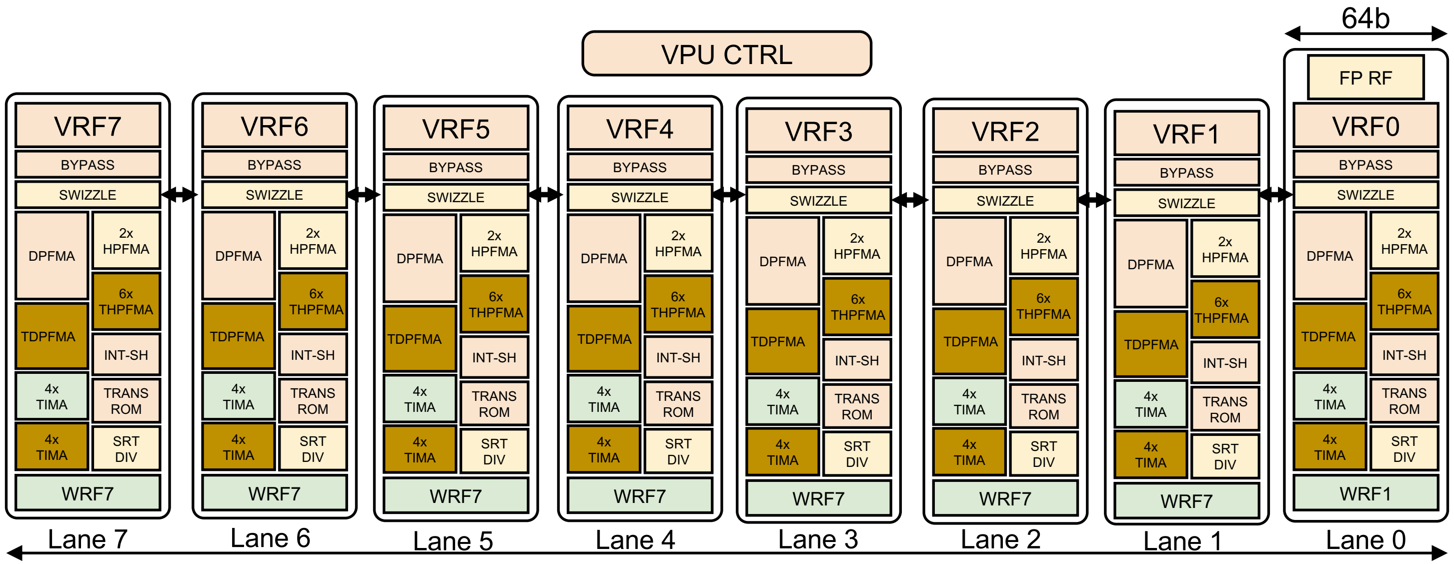
- 64-bit RISC-V compatible CPU up to 2 GHz
- 2 hardware threads
- Low-power in-order core
- Short 3 cycle branch bubble if unpredicted
- Decoupled Vector/Tensor unit
- In-order appearance of Tensor instructions
- 4 KB L1 Data Cache
- 4-way issue
  - Memory
  - Int
  - FP/Vector
  - Tensor
- 48-bit Virtual Addresses / 46-bit Physical Addresses



## Minion2 Vector Unit

- RISC-V RVV Compatible
- Hand optimized for speed and low power
- 4 KB Vector Register File (32 64B reg x 2 threads)
- 16 KB Tensor Register File (16 1KB Tensor regs)

| Data Type | Vector (RVV instructions) Ops/cycle | Tensor (Tensor instructions) Ops/cycle |
|---|---|---|
| FP64 DP | 16 | 32 |
| FP32 SP | 32 | 64 |
| FP16/BF16 HP | 64 | 256 |
| FP8 QP | 128 | 512 |
| INT8 | 64 | 512 |



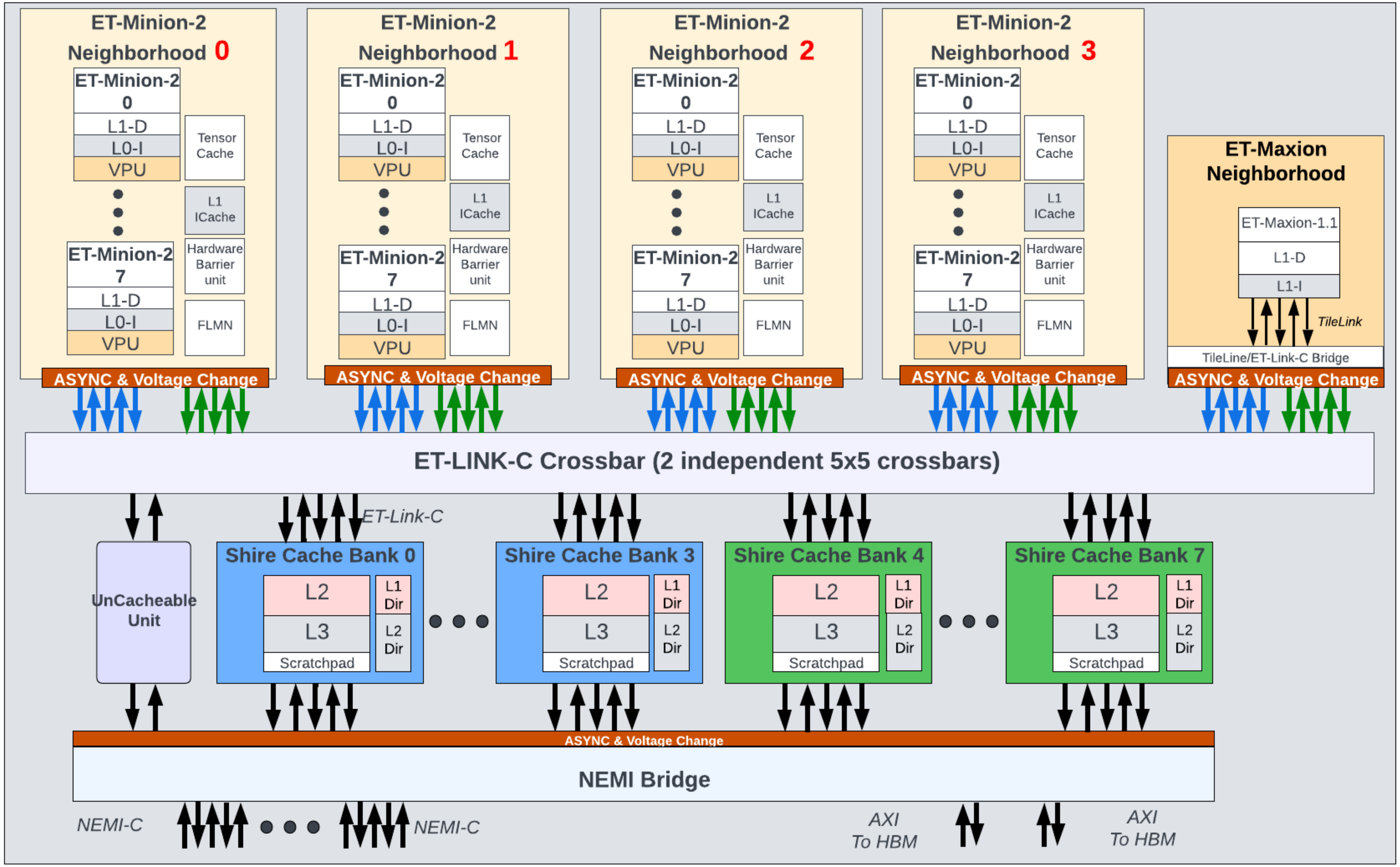
# New ET-SoC-2 SuperShire/Compute Cluster :
# 32 ET-Minion-2s, 1 ET-Maxion, 8 MB SRAM and NEMI NoC

Compute Fabric is built
out of multiple **SuperShires**.

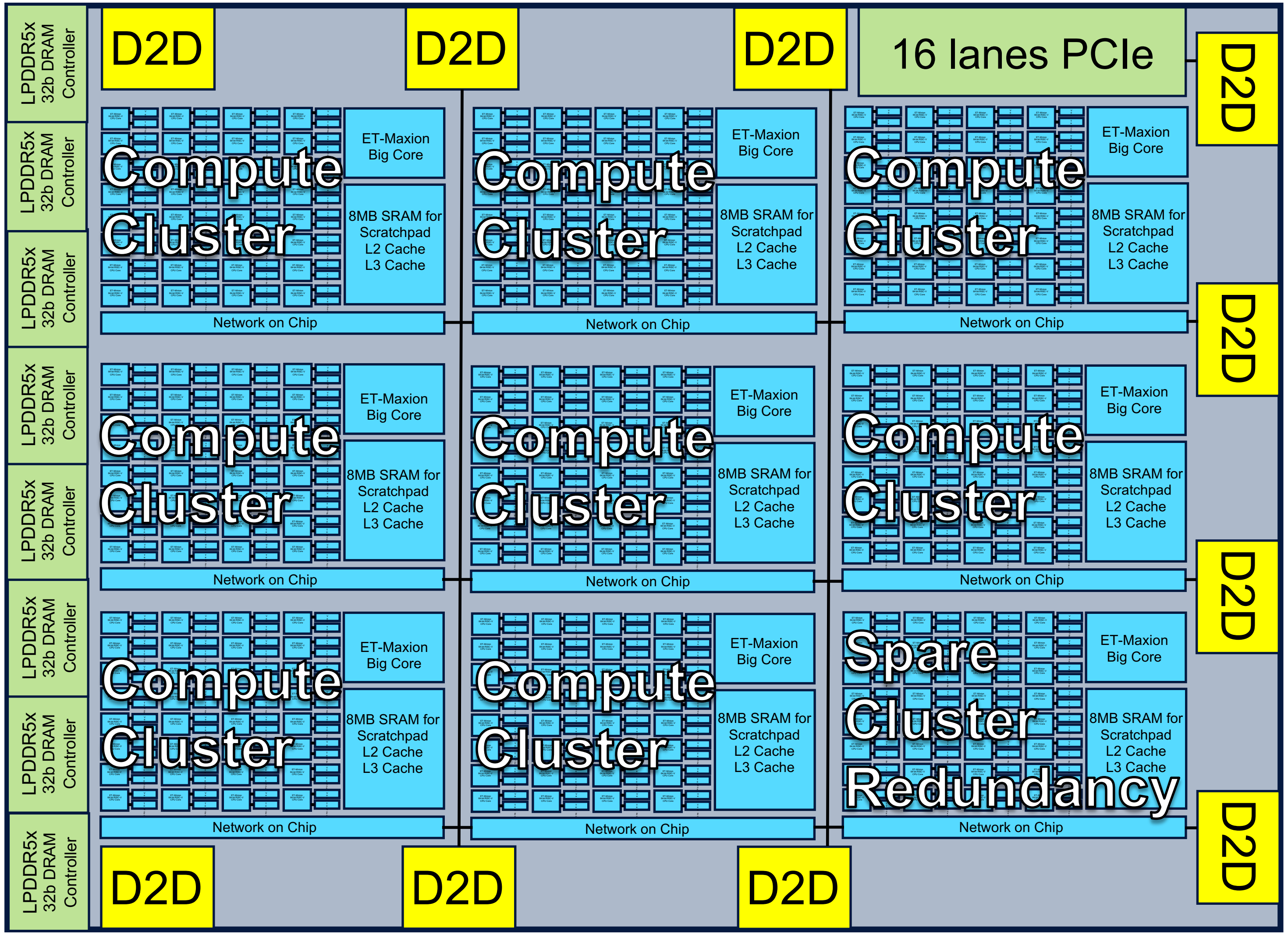
There are **9** SuperShires in the
4nm ET-SoC-2x.

Each SuperShire has:

- **1** ET-Maxion CPU
- **32** ET-Minion2 CPUs (Grouped in 4 Neighborhoods)
- Eight 1 MB SRAM Banks (software configurable as L2, L3 or scratchpad memory)
- Two 5x5 crossbar links
- NEMI NoC Bridge to other SuperShires or IO



# ET-SoC-2x 4nm Compute Fabric Chiplet



LPDDR5x 32b DRAM Controller
LPDDR5x 32b DRAM Controller
LPDDR5x 32b DRAM Controller
LPDDR5x 32b DRAM Controller
LPDDR5x 32b DRAM Controller
LPDDR5x 32b DRAM Controller
LPDDR5x 32b DRAM Controller
LPDDR5x 32b DRAM Controller
D2D
D2D
D2D
16 lanes PCIe
D2D
D2D
D2D
D2D
D2D
D2D
D2D
Compute Cluster
ET-Maxion Big Core
8MB SRAM for Scratchpad L2 Cache L3 Cache
Network on Chip
Spare Cluster Redundancy

ET-SoC-2x chiplet has:

- 8 Compute Clusters + 1 Spare
- 256 ET-Minion2 processors
- 8 ET-Maxion processors
- 64 MB on die SRAM
- 256 bit wide LPDDR5x-8500 controllers
- 16 lanes of PCIe Gen5

8 active compute clusters, each with

- One ET-Maxion “big core”
- 32 Minion2 “efficiency cores”
    - Each with its own Vector and Tensor units
- 8 MB of SRAM for L2, L3 and scratchpad
- Network on Chip to other compute clusters

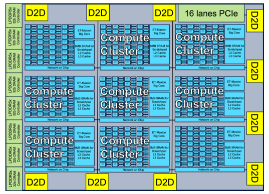
1 compute cluster used for redundancy
to increase yield and lower cost.

Die to die (D2D) interfaces allow fabric to grow
each has similar bandwidth as on die NoC to NoC

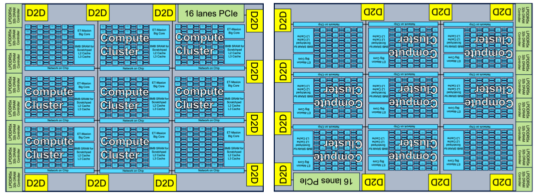
# ET-SoC-2x Allowed Compute Fabric Chiplet Configurations

esperanto.ai

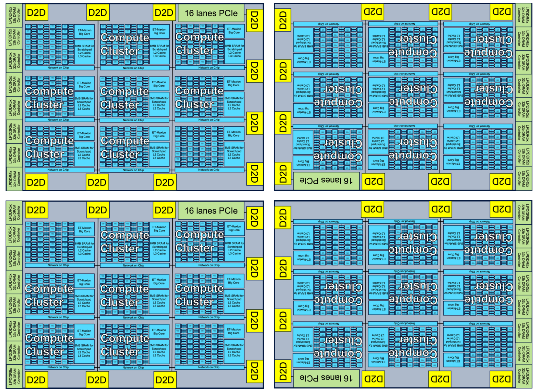
ET-SoC-2x1



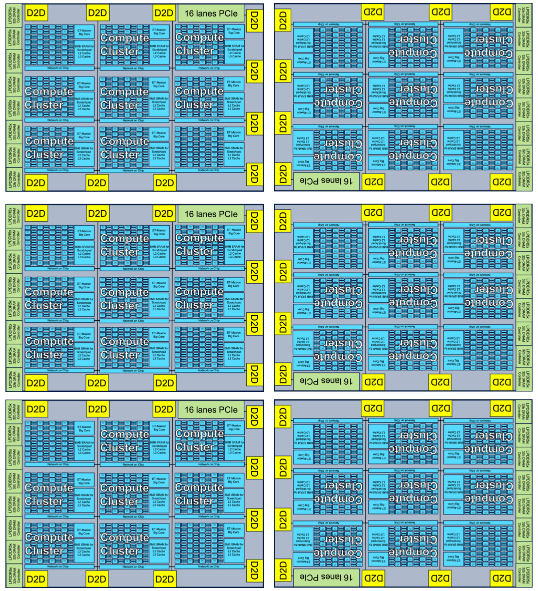
ET-SoC-2x2



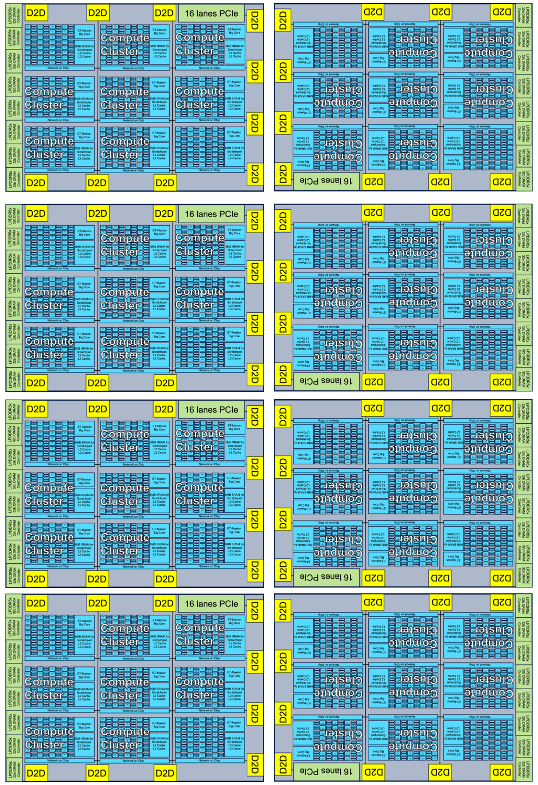
ET-SoC-2x4



ET-SoC-2x6



ET-SoC-2x8

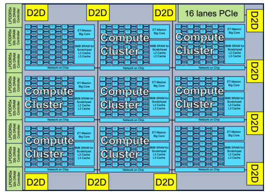


| | ET-SoC-2x1 | ET-SoC-2x2 | ET-SoC-2x4 | ET-SoC-2x6 | ET-SoC-2x8 |
|---|---|---|---|---|---|
| # Chiplets in package | 1 | 2 | 4 | 6 | 8 |
| LPDDR5x lanes | 256 | 512 | 1024 | 1536 | 2048 |
| Memory GB(Rank=4) | 256 | 512 | 1024 | 1536 | 2048 |
| Memory Bandwidth GB/s | 272 | 544 | 1088 | 1632 | 2176 |
| | | | | | |
| PCIe Gen 5 Lanes | 16 | 32 | 64 | 96 | 128 |
| PCIe Bandwidth GB/s | 64 | 128 | 256 | 384 | 512 |
| | | | | | |
| Peak TFlops GHz-> | 2 | | | | |
| FP8 or Int8 | 262 | 524 | 1048 | 1572 | 2096 |
| FP16 / BF16 | 131 | 262 | 524 | 786 | 1048 |
| FP32 | 32.8 | 65.6 | 131.2 | 196.8 | 262.4 |
| FP64 | 16.4 | 32.8 | 65.6 | 98.4 | 131.2 |

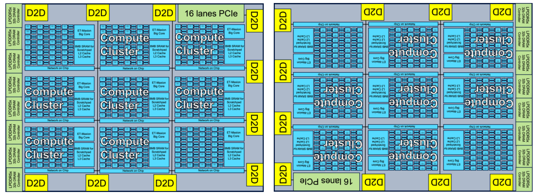
# ET-SoC-2x Allowed Compute Fabric Chiplet Configurations

esperanto.ai

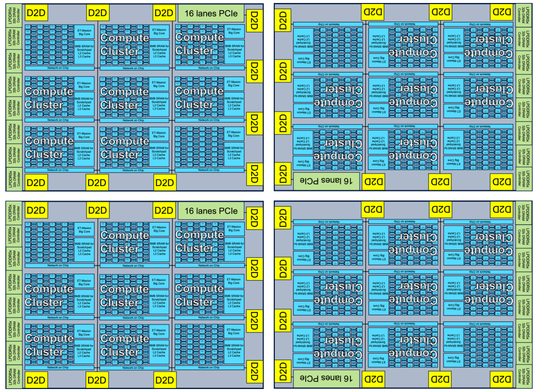
ET-SoC-2x1



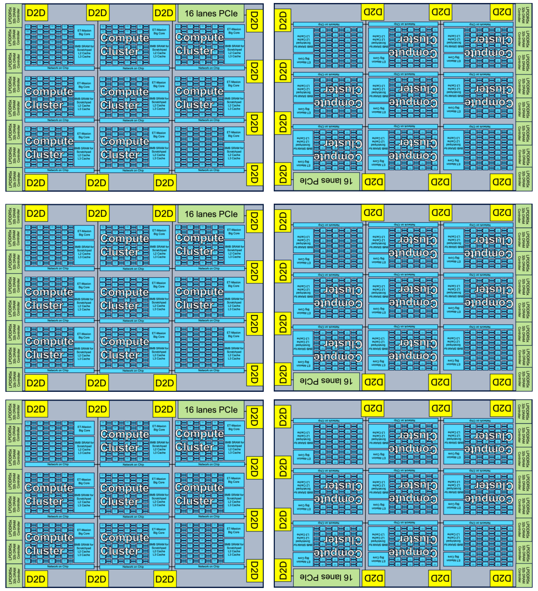
ET-SoC-2x2



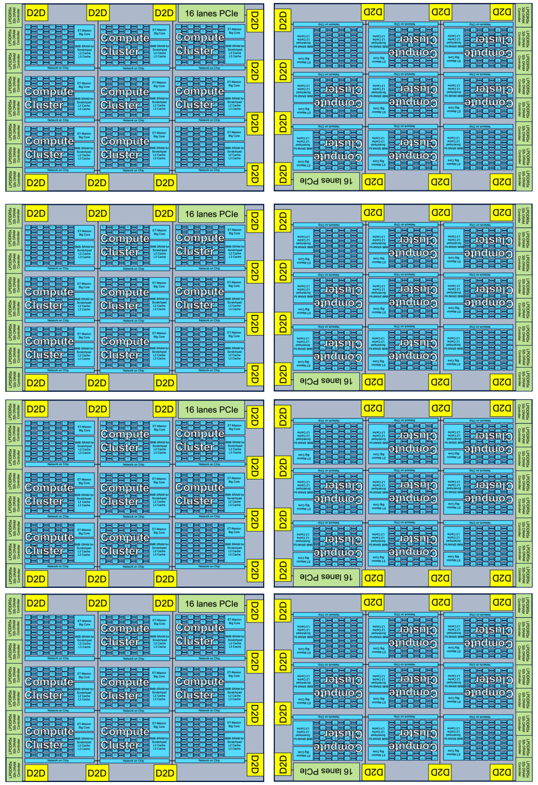
ET-SoC-2x4



ET-SoC-2x6

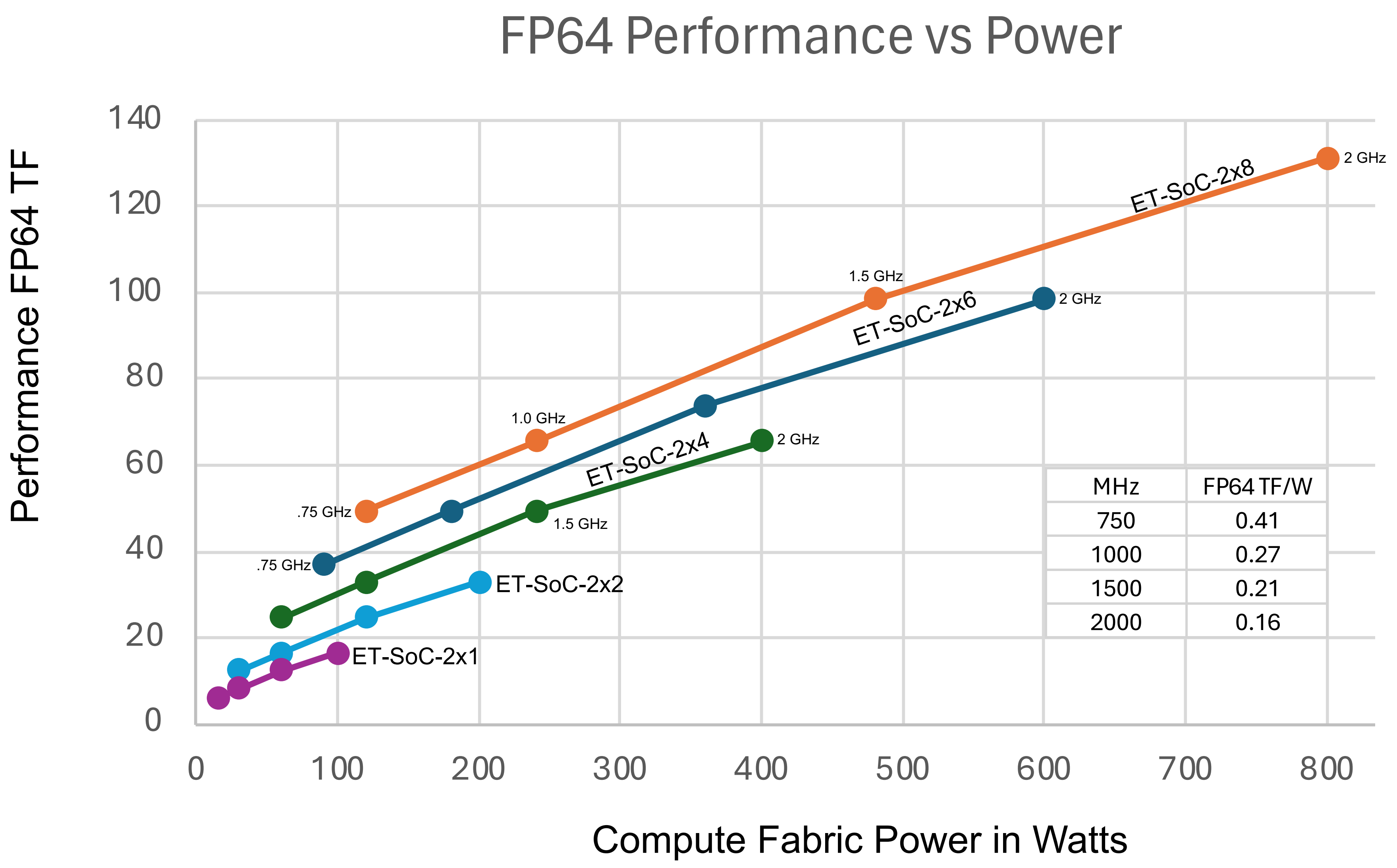


ET-SoC-2x8



| | ET-SoC-2x1 | ET-SoC-2x2 | ET-SoC-2x4 | ET-SoC-2x6 | ET-SoC-2x8 |
|---|---|---|---|---|---|
| # Chiplets in package | 1 | 2 | 4 | 6 | 8 |
| LPDDR5x lanes | 256 | 512 | 1024 | 1536 | 2048 |
| Memory GB(Rank=4) | 256 | 512 | 1024 | 1536 | 2048 |
| Memory Bandwidth GB/s | 272 | 544 | 1088 | 1632 | 2176 |
| | | | | | |
| PCIe Gen 5 Lanes | 16 | 32 | 64 | 96 | 128 |
| PCIe Bandwidth GB/s | 64 | 128 | 256 | 384 | 512 |
| | | | | | |
| Peak TFlops GHz-> | 2 | | | | |
| FP8 or Int8 | 262 | 524 | 1048 | 1572 | 2096 |
| FP16 / BF16 | 131 | 262 | 524 | 786 | 1048 |
| FP32 | 32.8 | 65.6 | 131.2 | 196.8 | 262.4 |
| FP64 | 16.4 | 32.8 | 65.6 | 98.4 | 131.2 |

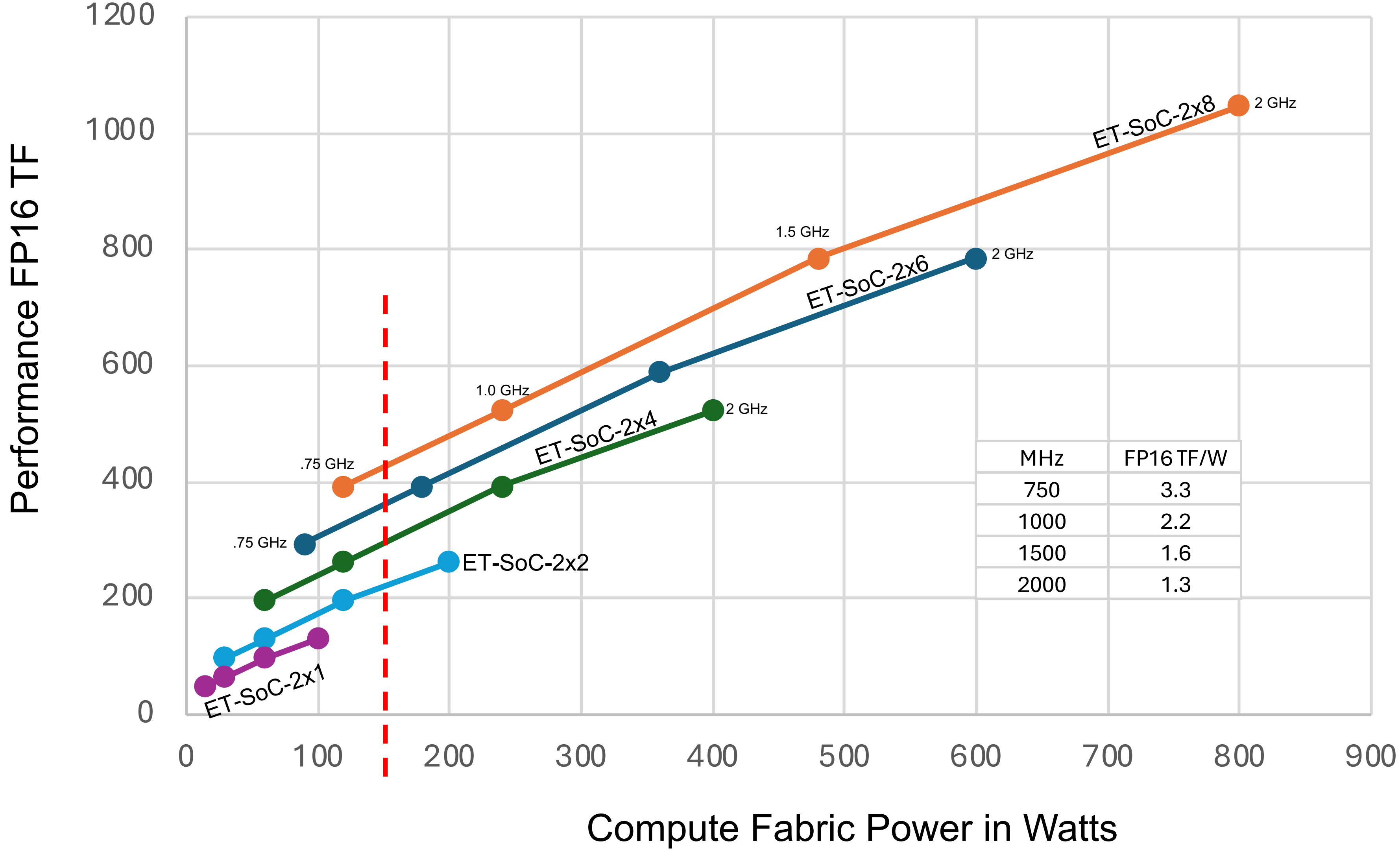
# FP64 Performance vs Power for different 4nm ET-SoC-2 Packages


FP64 Performance vs Power

| MHz | FP64 TF/W |
|---|---|
| 750 | 0.41 |
| 1000 | 0.27 |
| 1500 | 0.21 |
| 2000 | 0.16 |

Performance and Power projections June 10 2024, subject to change

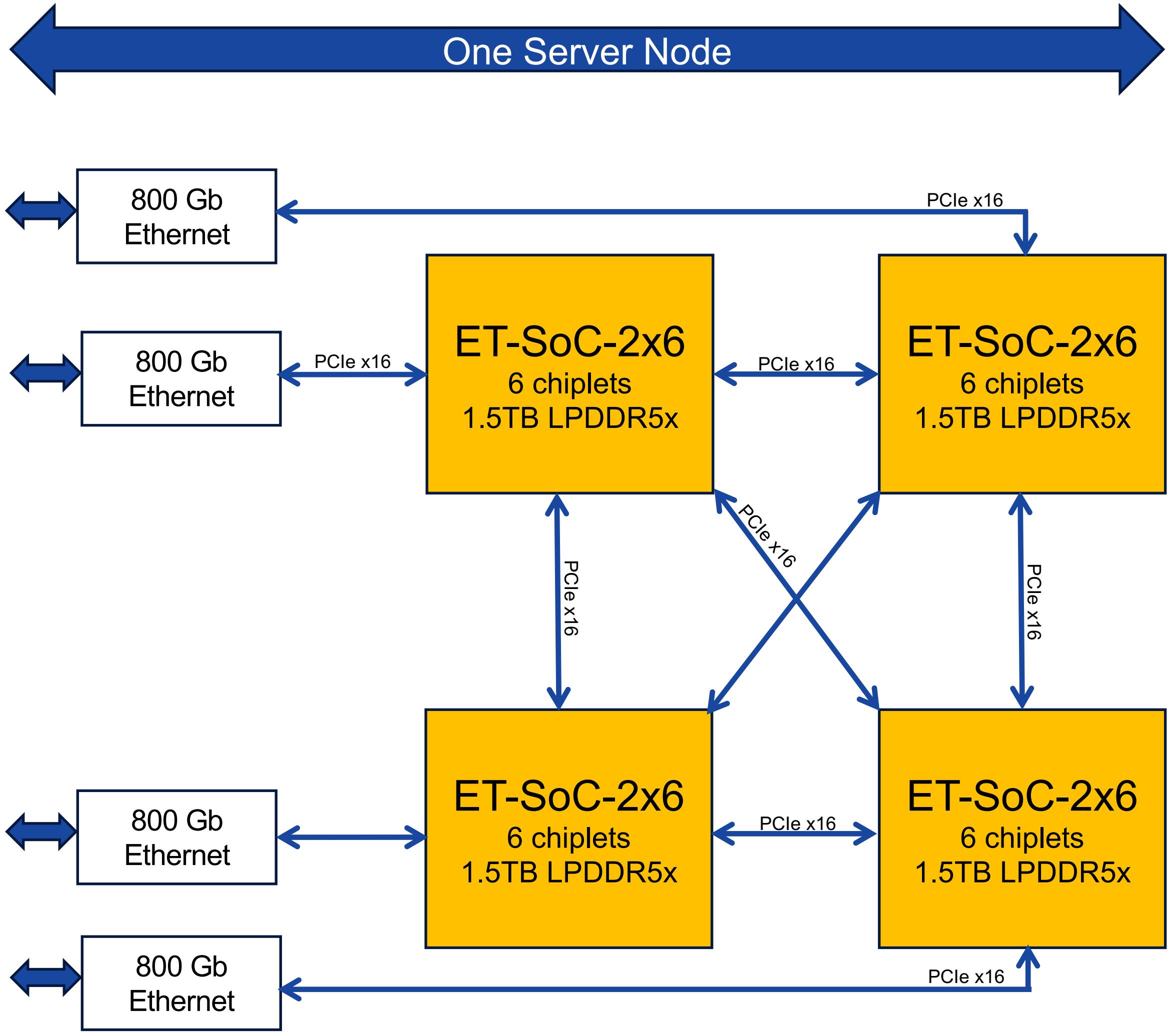
# FP16 Performance vs Power for different 4nm ET-SoC-2 Packages



| MHz | FP16 TF/W |
|---|---|
| 750 | 3.3 |
| 1000 | 2.2 |
| 1500 | 1.6 |
| 2000 | 1.3 |

Performance and Power projections June 10 2024, subject to change

# Second Generation 4nm ET-SOC-2x General Purpose RISC-V Server Example



Four CPU packages on a motherboard per node

Each package has six ET-SoC-2x chiplets

All CPUs share single address space and memory

6144 RISC-V vector/tensor processors

6 TB of shared DRAM

Peak performance per server:

- 384 TF FP64
- 768 TF FP32
- 3072 TF FP16
- 6144 TF FP8

Performance per chiplet doubles with 2nm ET-SoC-3

About 36 racks to provide 1 ExaFlop DP peak

# ET-SoC-3x

# 2nm

# Main changes to ET-SoC-3x Chiplet

Similar software interface between 4nm ET-SoC-2x and 2nm ET-SoC-3x

2nm technology allows 512 ET-Minion-3 RISC-V processors, double that of 4nm ET-SoC-2x

Improved memory system to support up to 2 TB of useable bandwidth per chiplet

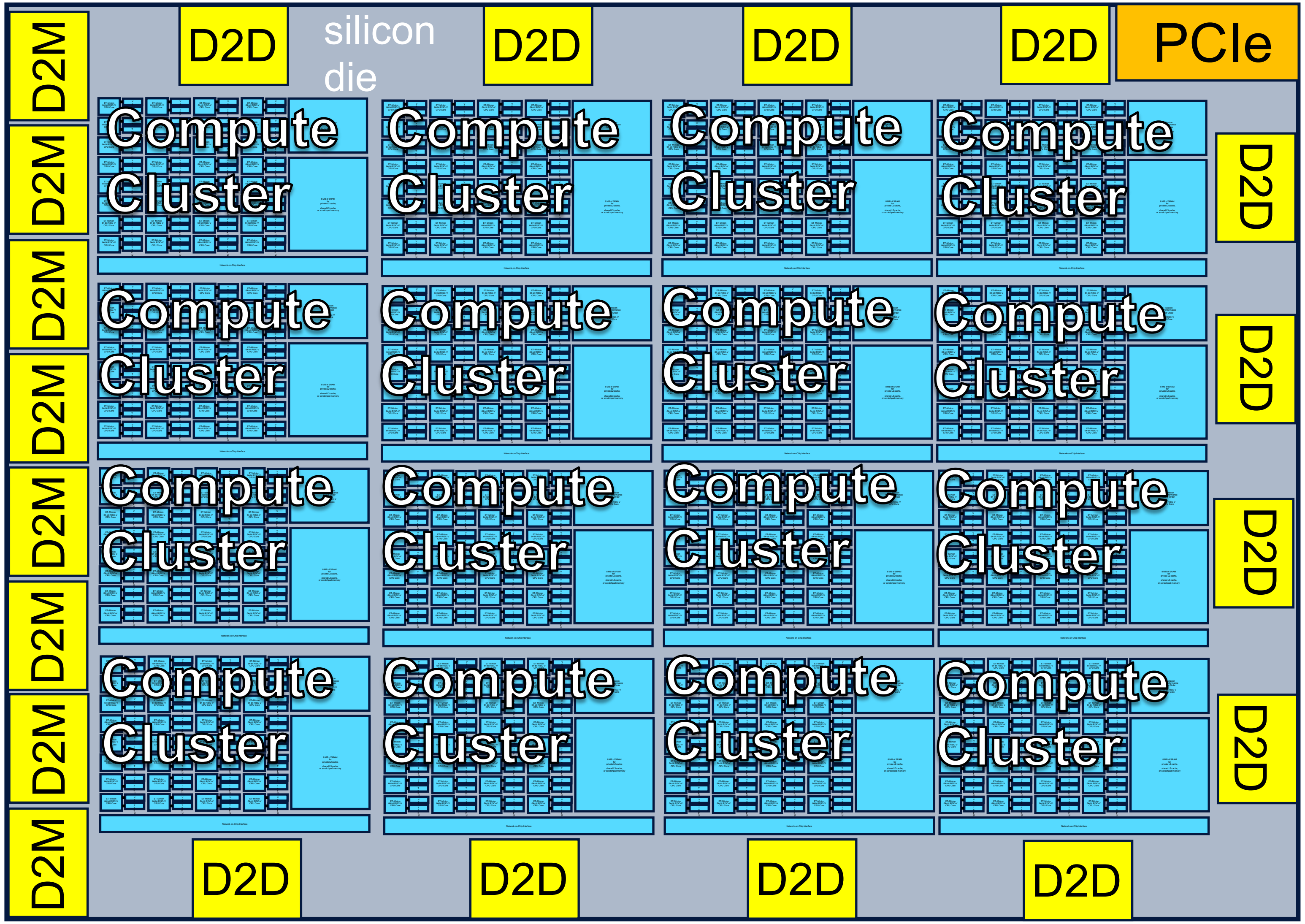
Higher speed/bandwidth for chiplet die to die interconnect

Expanded physical and virtual address space (57 bits)

PCIe Gen6 or Gen7 IO

Other chiplet support, eg optical IO, TBD, depending on customer interest

# ET-SoC-3x 2nm Compute Fabric Chiplet: Twice the compute vs 4nm



ET-SoC-3x chiplet has:

- 16 Compute Clusters with
- 512 ET-Minion2 processors
- 16 ET-Maxion processors
- 128 MB on die SRAM

16 compute clusters, each with

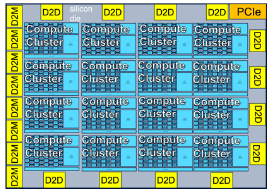
- One ET-Maxion “big core”
- 32 Minion “efficiency cores” in each cluster
- 8 MB of SRAM for L2, L3 and scratchpad
- Network on Chip to other Compute Clusters

Die to die (D2D) interfaces allow fabric to grow.

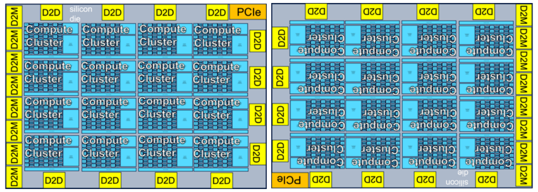
Die to Memory allows up to 2 TB/s bandwidth direct to memory side cache on die

# ET-SoC-3x Allowed Compute Fabric Configurations

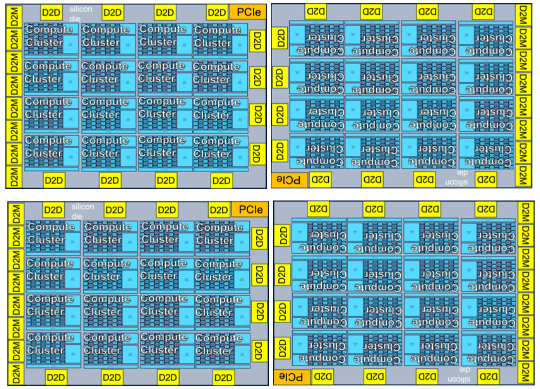
ET-SoC-3x1



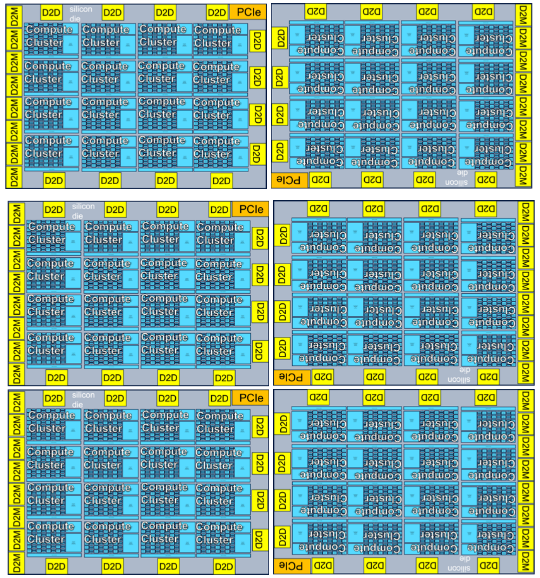
ET-SoC-3x2



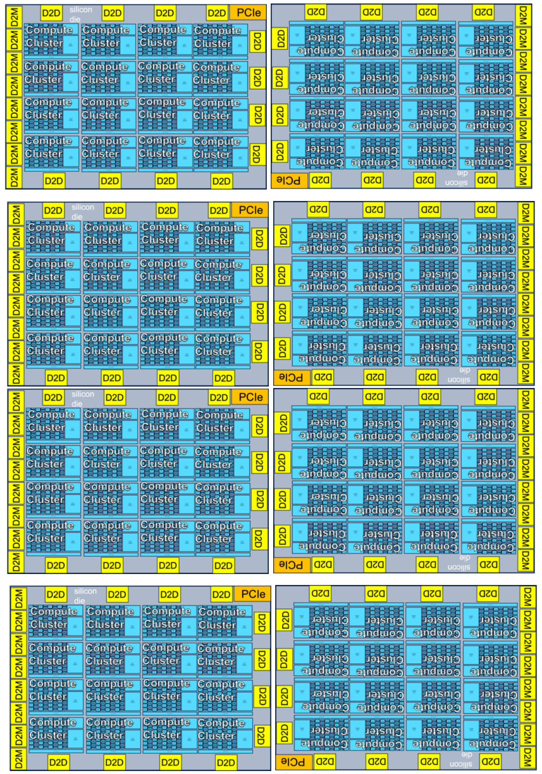
ET-x4



ET-SoC-3x6

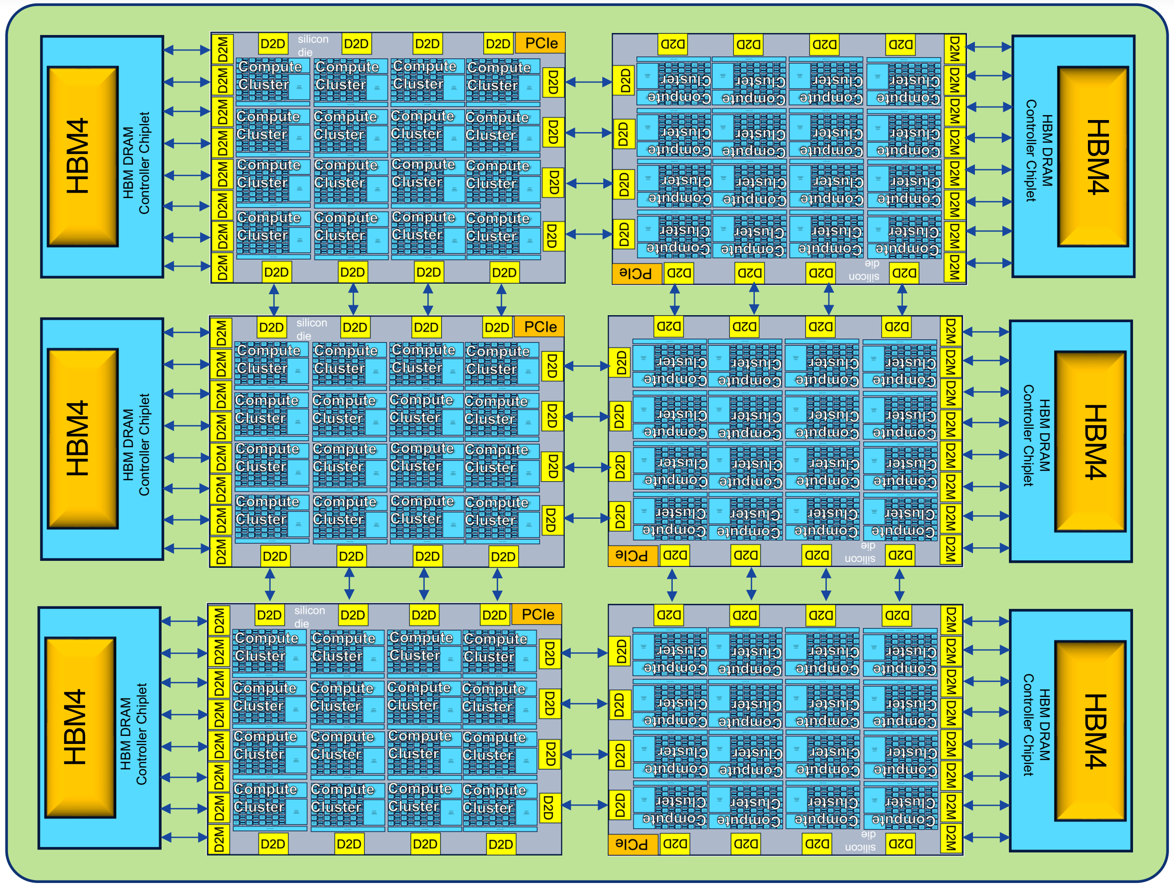


ET-SoC-3x8



| | ET-SoC-3x1 | ET-SoC-3x2 | ET-SoC-3x4 | ET-SoC-3x6 | ET-SoC-3x8 |
|---|---|---|---|---|---|
| # Chiplets in package | 1 | 2 | 4 | 6 | 8 |
| # of HBM4e supported | 1 | 2 | 4 | 6 | 8 |
| HBM4e Memory GB | 64 | 128 | 256 | 384 | 512 |
| Memory Bandwidth TB/s | 2 | 4 | 8 | 12 | 16 |
| | | | | | |
| PCIe Gen 6/7 Lanes | 16 | 32 | 64 | 96 | 128 |
| PCIe Bandwidth GB/s | 128 | 256 | 384 | 512 | 1024 |
| | | | | | |
| Peak TFlops GHz-> | 2 | | | | |
| FP8 or Int8 | 524 | 1048 | 2096 | 3144 | 4192 |
| FP16 / BF16 | 262 | 524 | 1048 | 1572 | 2096 |
| FP32 | 65.6 | 131.2 | 262.4 | 394 | 524 |
| FP64 | 32.8 | 65.6 | 131.2 | 196 | 262 |

# 2nm ET-SoC-3 (x6) package configuration example product



Six 2nm ET-SoC-3 Chiplets

- Max perf example @2 GHz
- 77 Watts per Compute Chiplet
- 460 Watts for 6 Compute Chiplets

Peak Performance:

- 200 TFlops FP64
- 400 TFlops FP32
- 1600 TFlops FP16/BF16
- 3200 TFlops FP8
- 3200 TOps Int8

Six HBM4e on module (2TB/s 64 GB)

- 384 GB capacity
- 12 TB/sec bandwidth

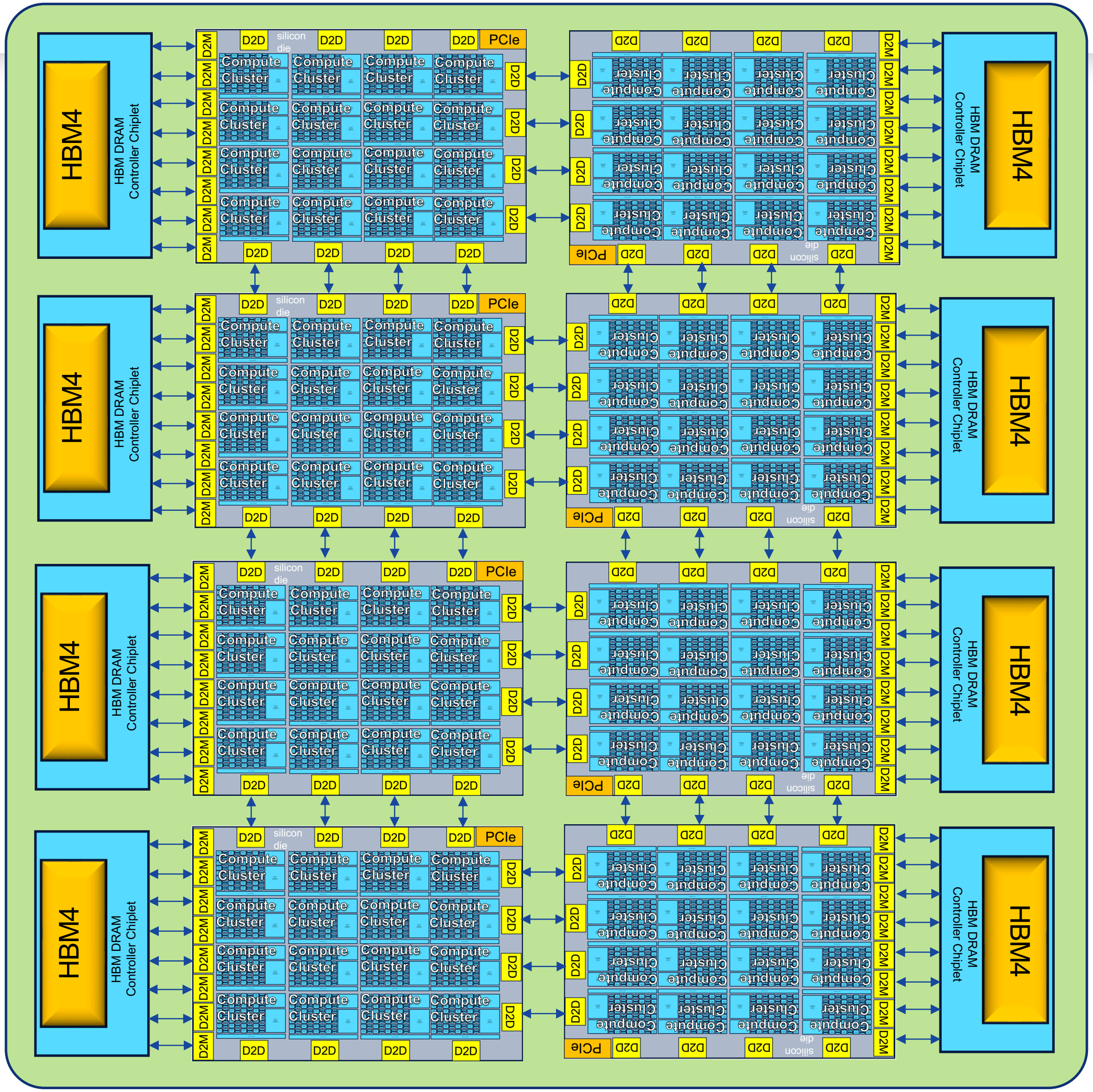
96 lanes PCIe Gen6/7

Expansion DRAM off module options

Six 2 GHz chiplets have 200 TFlops FP64 / 460 watts = 434 GFlops/watt

Better FP64 performance than new NVIDIA Blackwell 1KW B200 GPU which is only 40 TFlops FP64

# 2nm ET-SoC-3 (x8) package configuration example product



Eight 2nm ET-SoC-3 Chiplets

- Efficient example @1.5 GHz
- 44 Watts per Compute Chiplet
- 350 Watts for compute chiplets

Peak Performance:

- 200 TFlops FP64
- 400 TFlops FP32
- 1600 TFlops FP16/BF16
- 3200 TFlops FP8
- 3200 TOps Int8

Eight HBM4e on module (2TB/s 64 GB)

- 512 GB capacity
- 16 TB/sec bandwidth

128 lanes PCIe Gen6/7

Expansion DRAM off module options

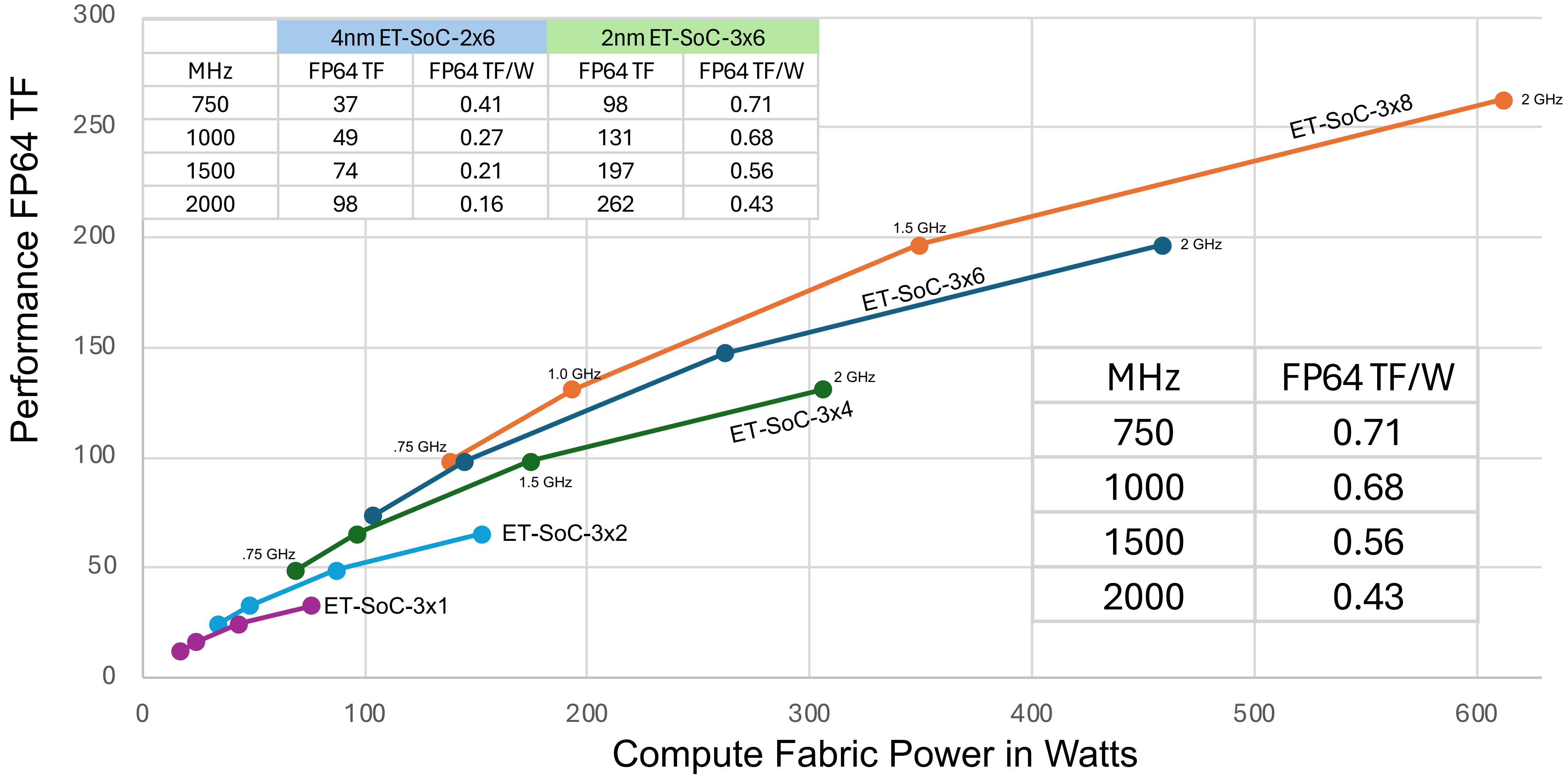
Better FP64 performance than new NVIDIA Blackwell 1KW B200 GPU which is only 40 TFlops FP64

Eight 1.5 GHz chiplets have 200 TFlops FP64 / 350 watts = 571 GFlops/watt

# FP64 Performance vs Power for different 2nm ET-SoC-3 Packages

esperanto.ai

## 2nm FP64 Performance vs Power

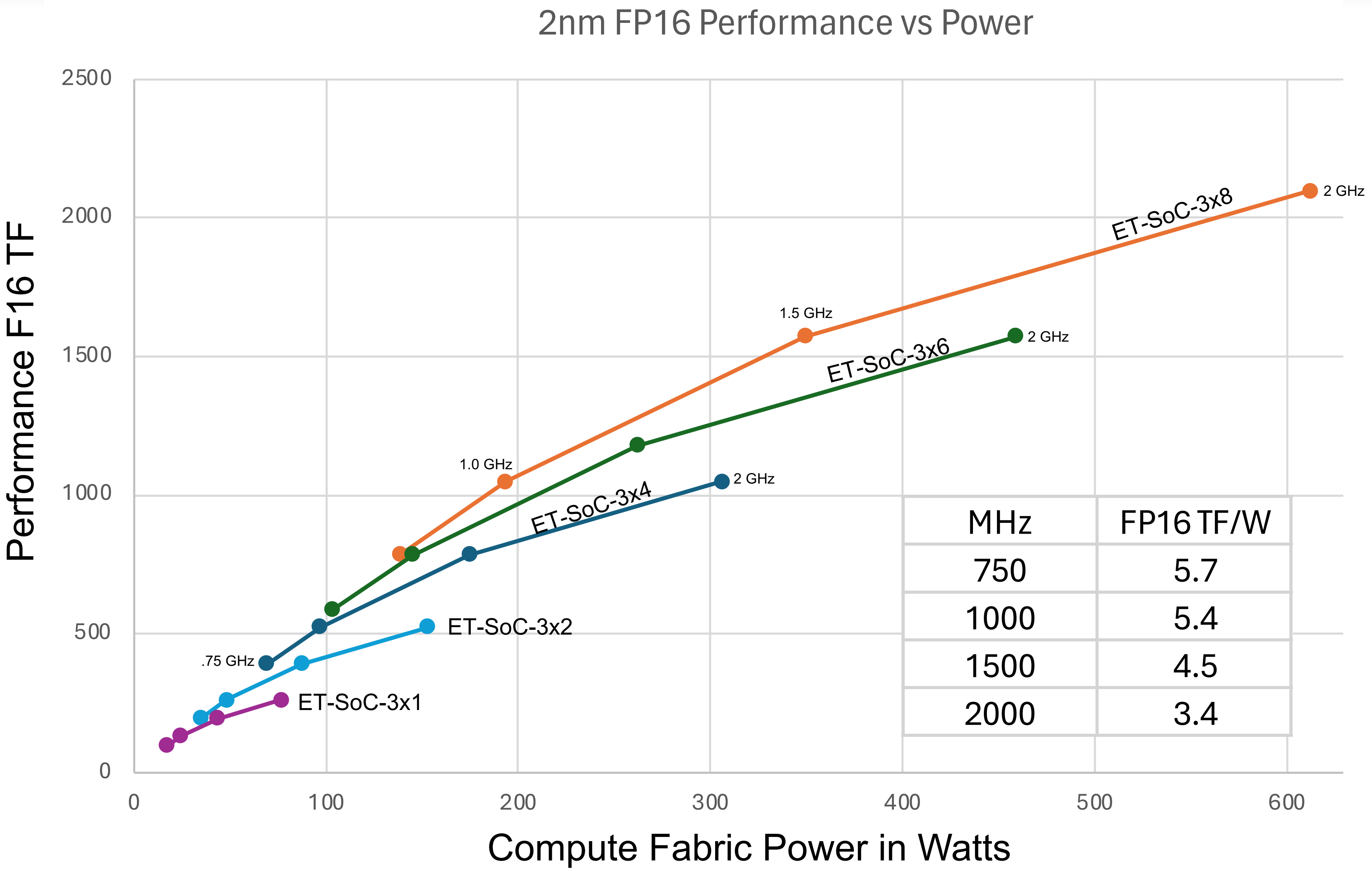


| | 4nm ET-SoC-2x6 | | 2nm ET-SoC-3x6 | |
|---|---|---|---|---|
| MHz | FP64 TF | FP64 TF/W | FP64 TF | FP64 TF/W |
| 750 | 37 | 0.41 | 98 | 0.71 |
| 1000 | 49 | 0.27 | 131 | 0.68 |
| 1500 | 74 | 0.21 | 197 | 0.56 |
| 2000 | 98 | 0.16 | 262 | 0.43 |

| MHz | FP64 TF/W |
|---|---|
| 750 | 0.71 |
| 1000 | 0.68 |
| 1500 | 0.56 |
| 2000 | 0.43 |

Performance and Power projections June 10 2024, subject to change

# FP16 Performance vs Power for different 2nm ET-SoC-3 Packages



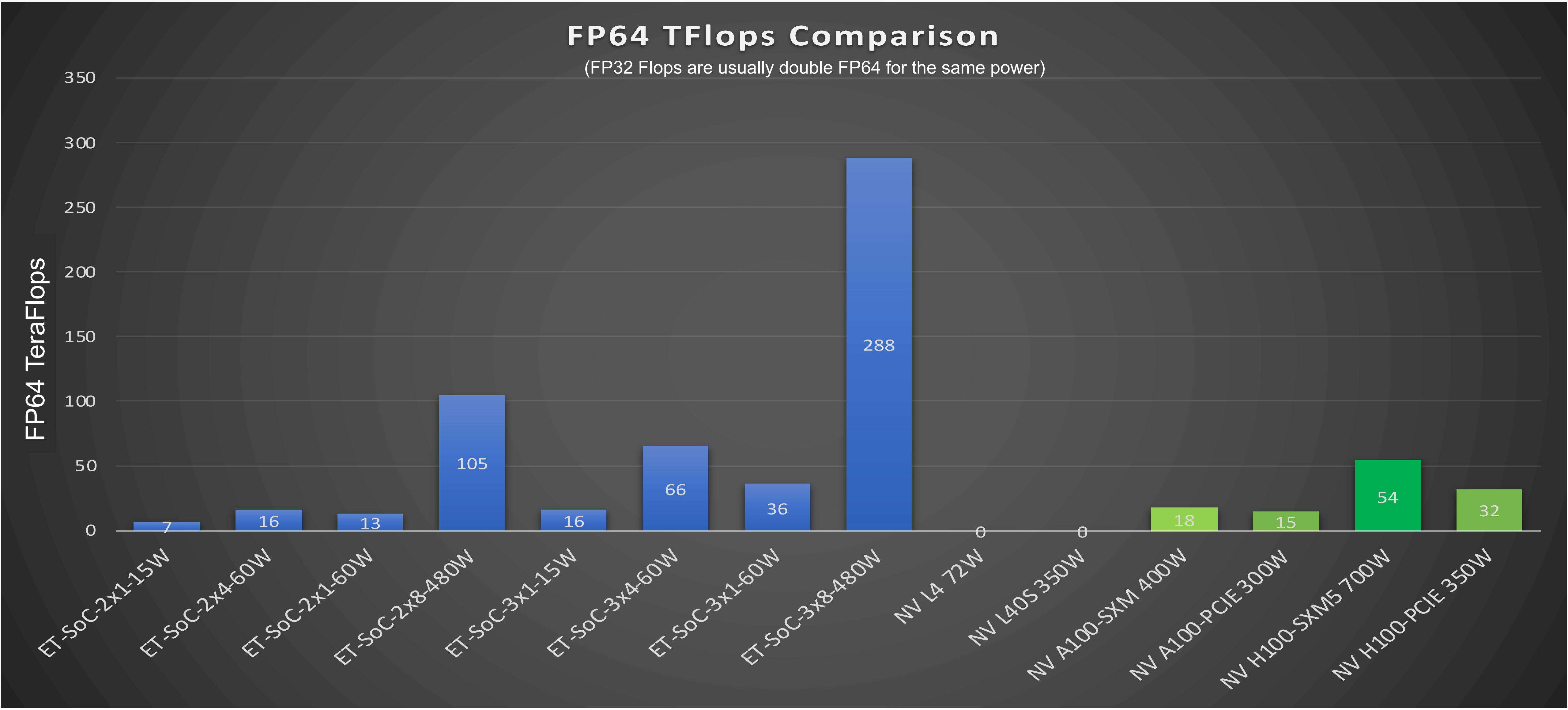
| MHz | FP16 TF/W |
|---|---|
| 750 | 5.7 |
| 1000 | 5.4 |
| 1500 | 4.5 |
| 2000 | 3.4 |

Performance and Power projections June 10 2024, subject to change

# Performance, Power and Performance per Watt Comparisons

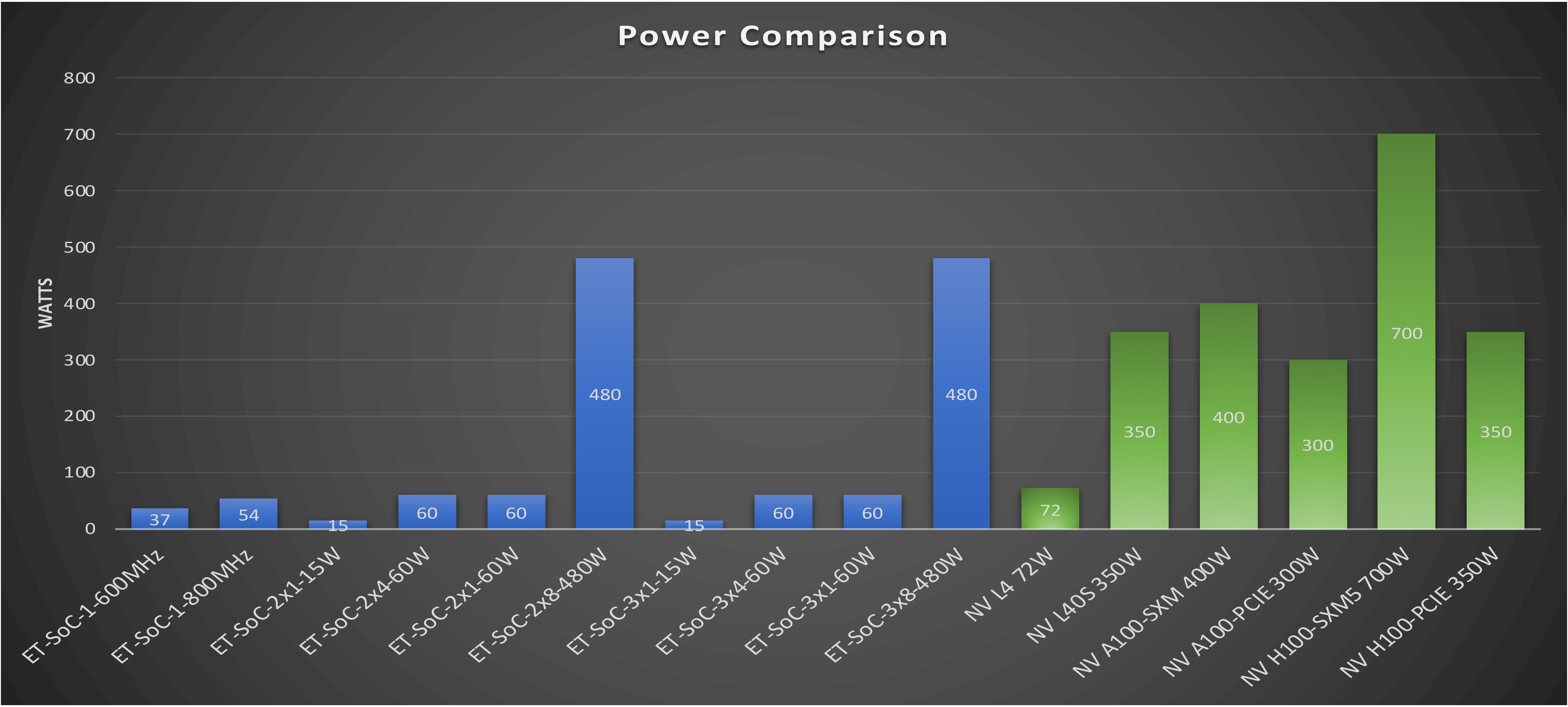
Note – to provide more apples to apples comparisons, the following graphs compare computation rates assuming no sparsity and sustainable base clock rates that do not exceed TDP.

# 64-bit double-precision floating point comparisons



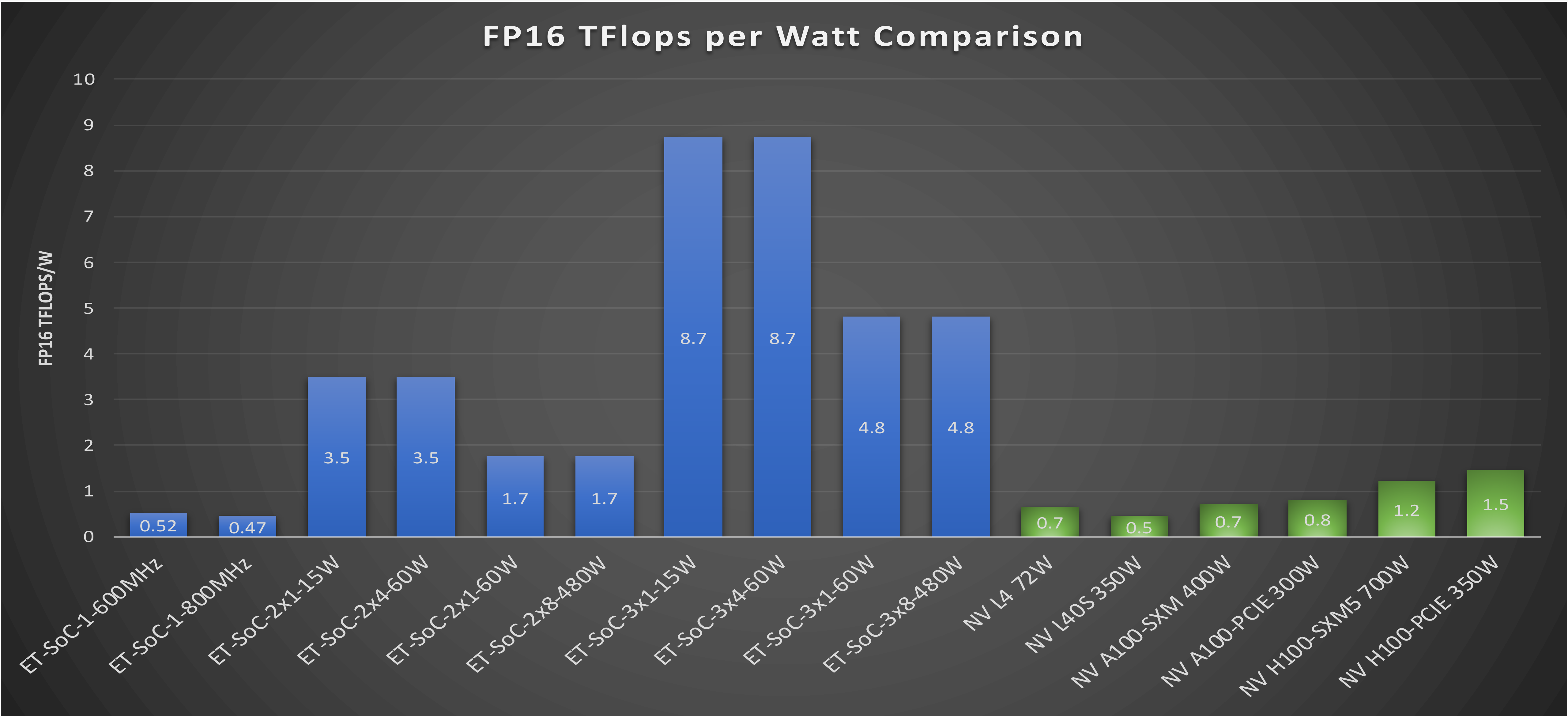
# Thermal Design Power (TDP) comparisons

esperanto.ai



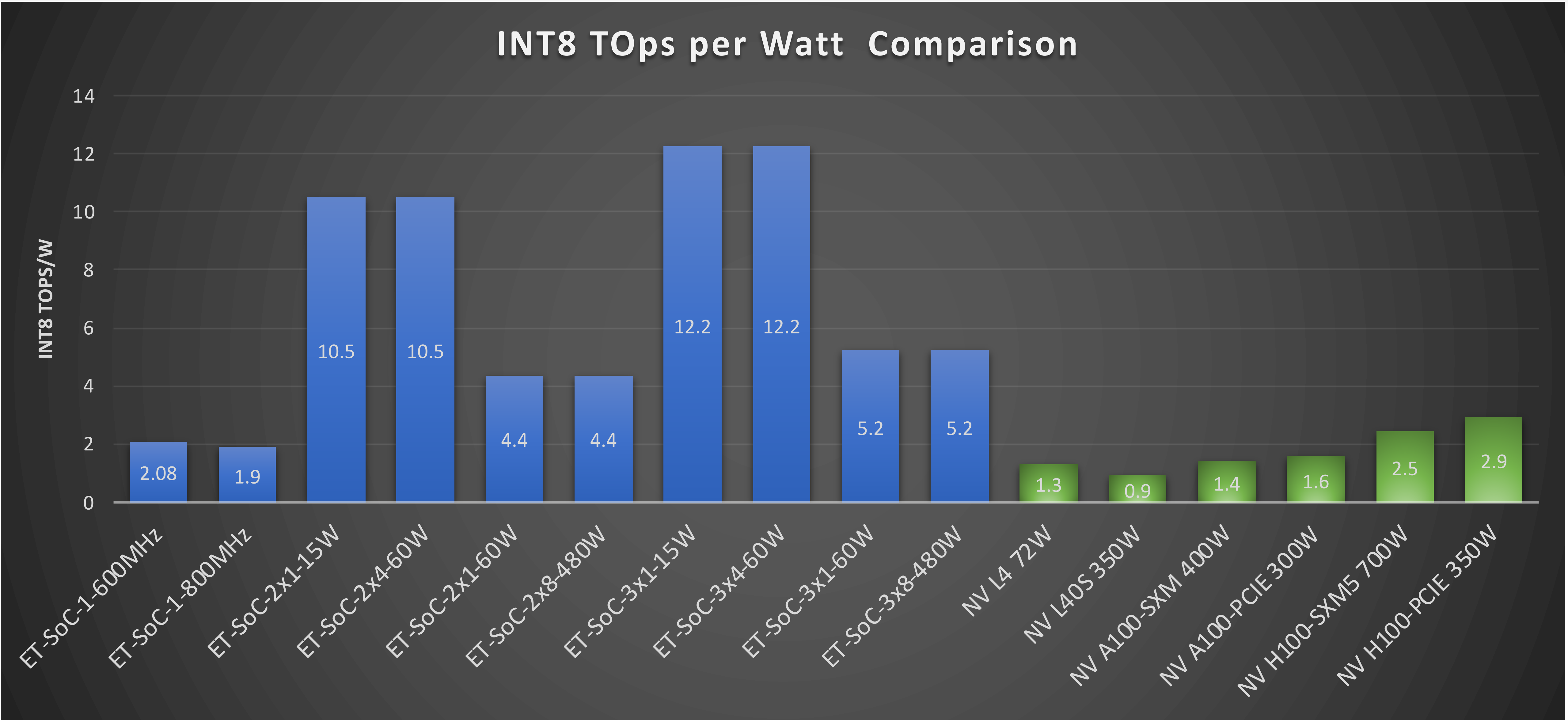
# Energy Efficiency: FP16 Tera Operations per Watt comparison

esperanto.ai

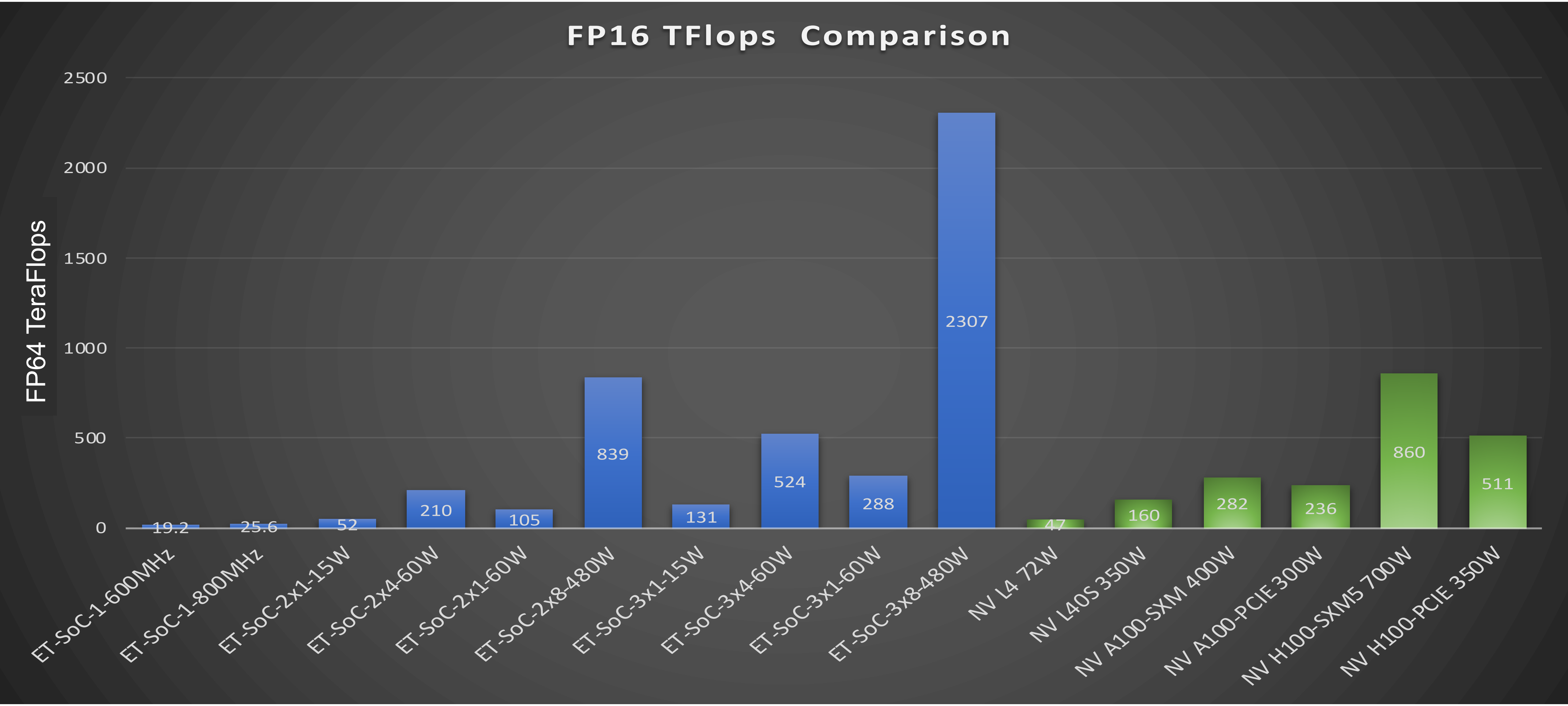


# Energy Efficiency: Int8 Tera Operations per Watt comparison

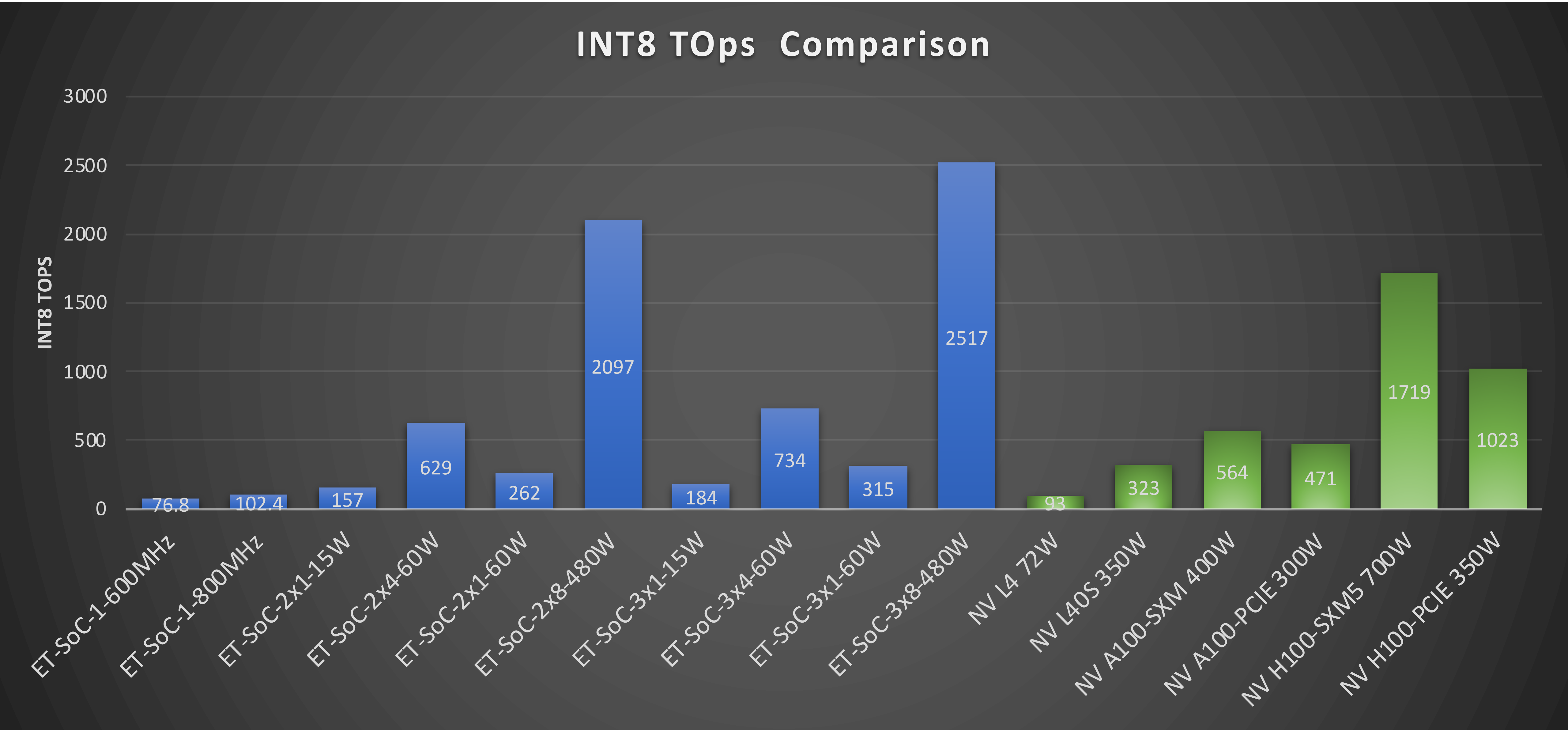
esperanto.ai



# 16-bit half-precision floating point comparisons.



# 8-bit integer Tera Operations per second



# Summary

**Have worked with many instruction sets: CISC, VLIW, x86, ARM, and RISC-V**

**RISC-V is the best of the alternatives**

- RISC-V is better technically
- RISC-V is better legally (¨Free and Open¨, i.e. no license needed)
- RISC-V is still maturing and has a lot of room to grow

**Join us in making RISC-V successful in the HPC space!**

esperanto.ai

www.esperanto.ai