AI Engines to Accelerate the AI Age

Alex Rico

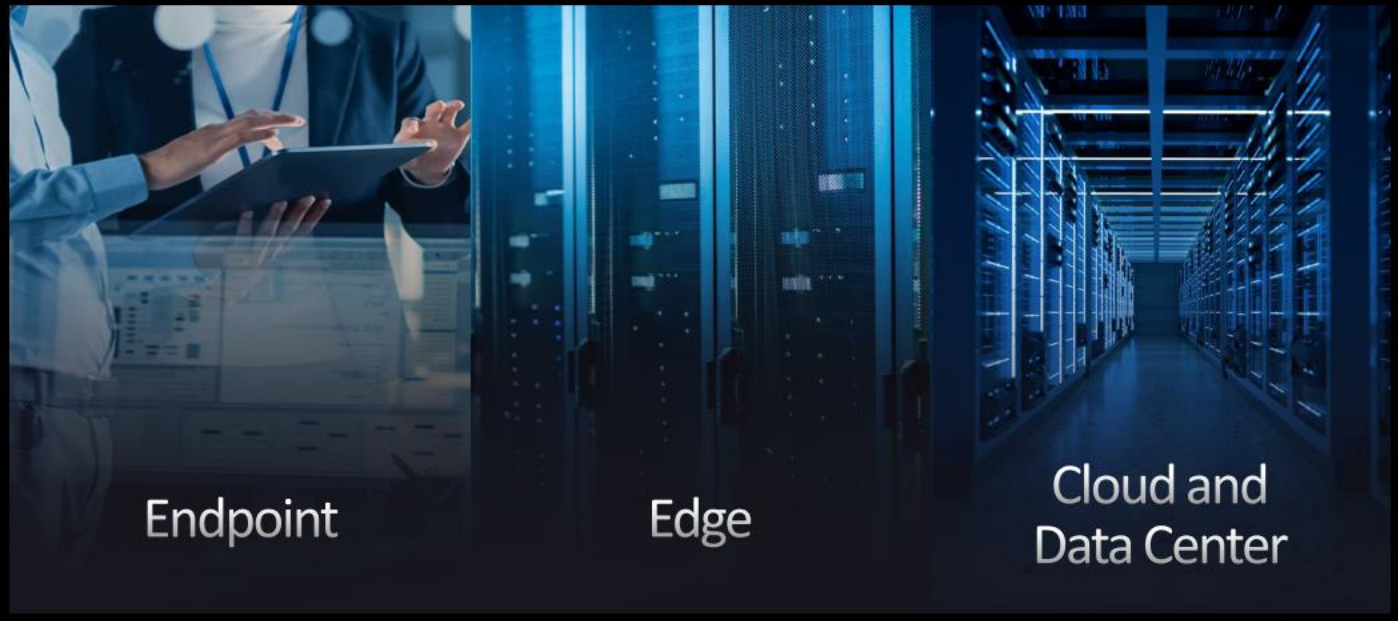
EXCELENCIA SEVERO OCHOA

BSC Barcelona Supercomputing Center
Centro Nacional de Supercomputación

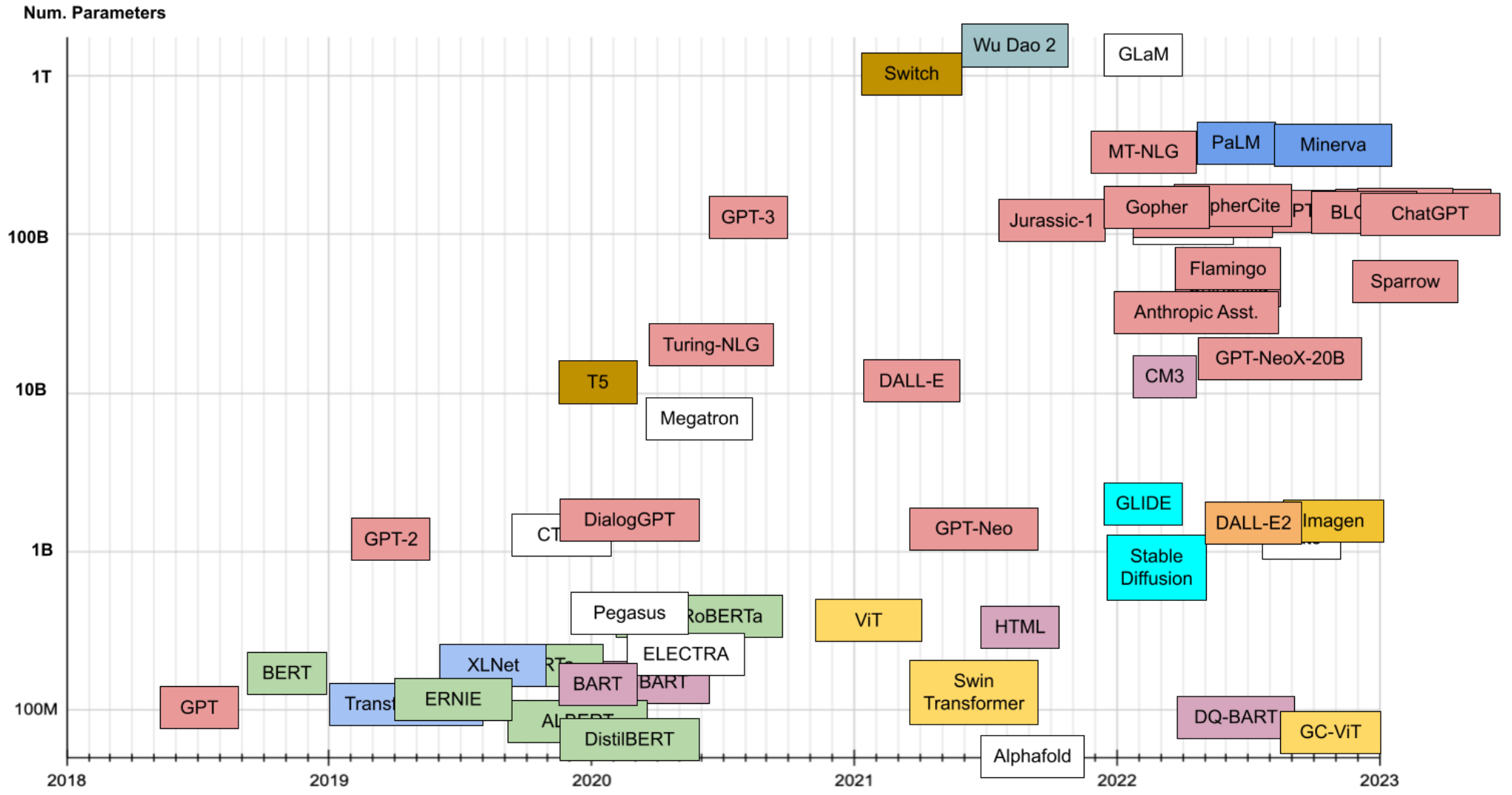
AMD
together we advance_

Challenge: Delivering Massive AI Compute at Low Latency and Low Power

Unlocking the potential of AI requires AI engines everywhere



AI Evolves Rapidly

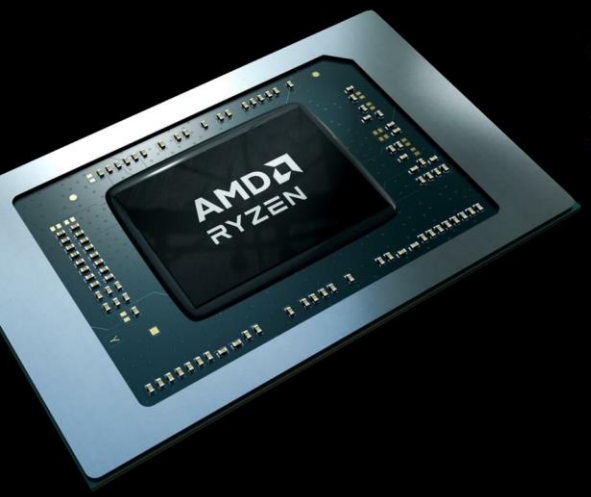


X. Amatriain. Transformer Models: an Introduction and Catalog. 2023

Pervasive AI

Cloud-to-Client Symmetry

CES 2023 Keynote



Announcing

AMD RYZEN™ 7040 Series

The Ultimate Ultrathin Processor

ZEN 4
PROCESSOR ARCHITECTURE

AMD RDNA 3
GRAPHICS ARCHITECTURE

AMD XDNA
AI ENGINE ARCHITECTURE

4nm
MANUFACTURING TECHNOLOGY

Featuring Ryzen™ AI

26

AMD ALVEO™ V70

AI inference accelerator

AMD XDNA AI Engine | 400 TOPS of AI compute | PCIe® 5.0 | 75W TDP | Cloud-to-client symmetry for AI developers

20

AMD
together we advance_

Scalable Architecture with Unified Software Stack

AMD
together we advance_

INTRODUCING

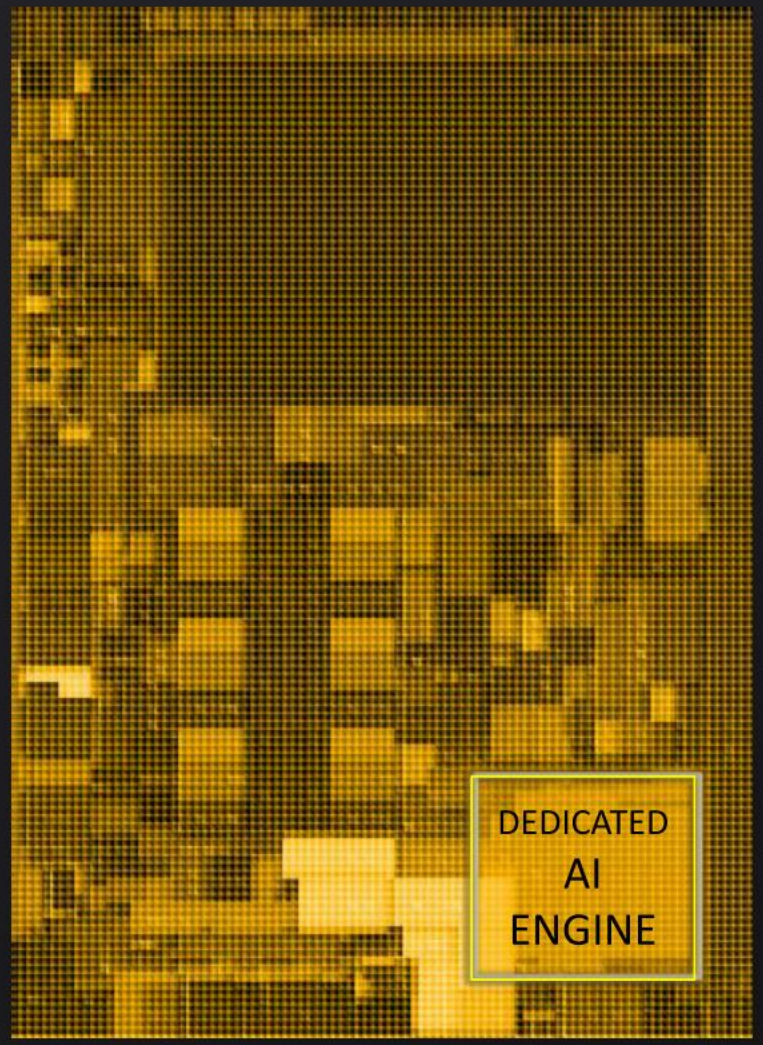
AMD RYZEN™ AI

THE WORLD'S FIRST INTEGRATED AI ENGINE ON AN X86 PROCESSOR

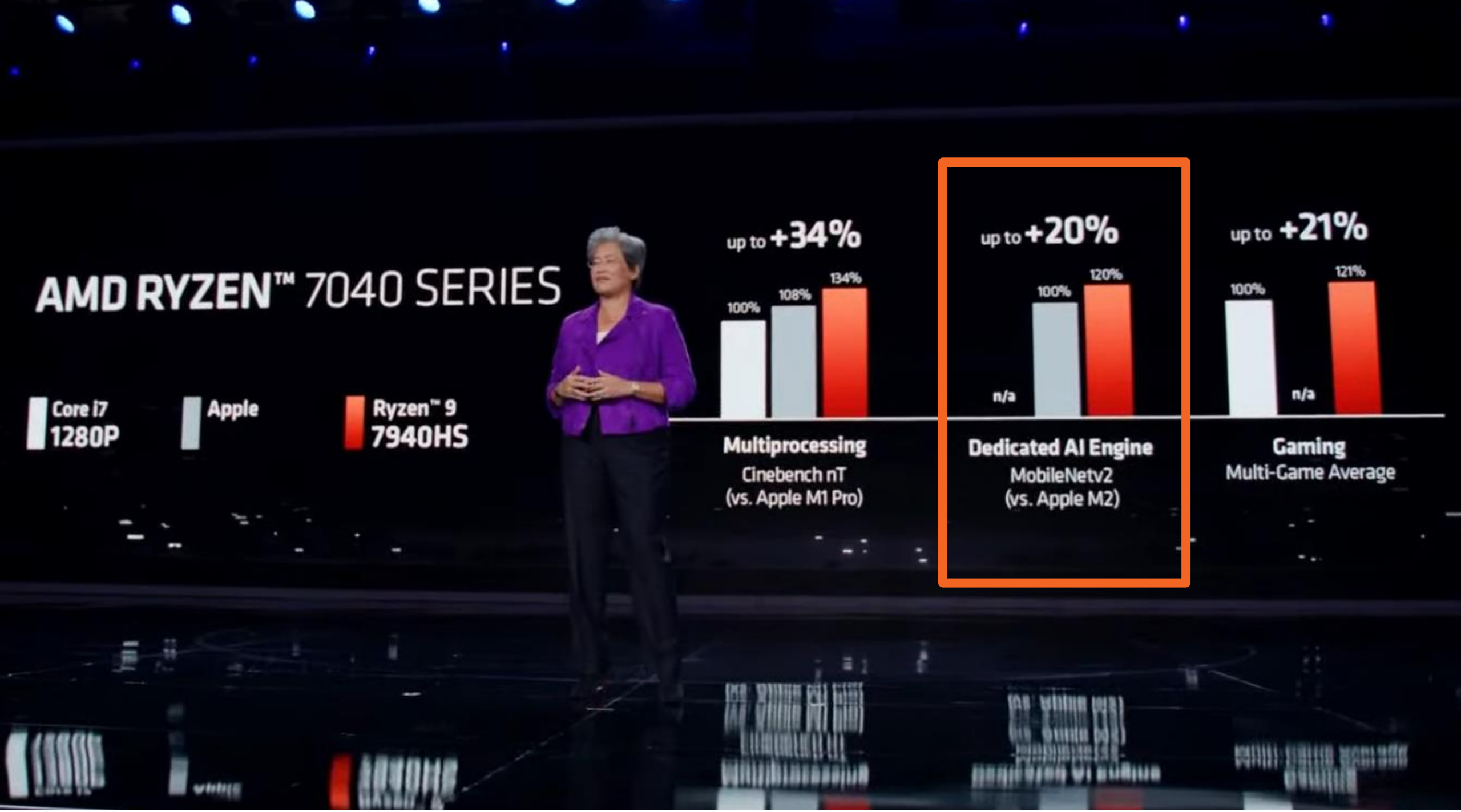
Up to 4 concurrent dedicated AI streams for real-time multi-tasking

Up to >35% more responsive AI experiences compared to a single AI stream

Designed for incredible efficiency



SEE ENDNOTES: PHX-3


AMD RYZEN™ 7040 SERIES
Core i7 1280P
Apple
Ryzen™ 9 7940HS
up to +34%
100%
108%
134%
Multiprocessing
Cinebench nT
(vs. Apple M1 Pro)
up to +20%
n/a
100%
120%
Dedicated AI Engine
MobileNetv2
(vs. Apple M2)
up to +21%
100%
n/a
121%
Gaming
Multi-Game Average

Versal™ AI Edge: Architecture Overview

XILINX VERSAL™ | AI CORE

XILINX VERSAL™ | PREMIUM

XILINX VERSAL™ | AI EDGE

ADAPTABLE TO MULTIPLE WORKLOADS

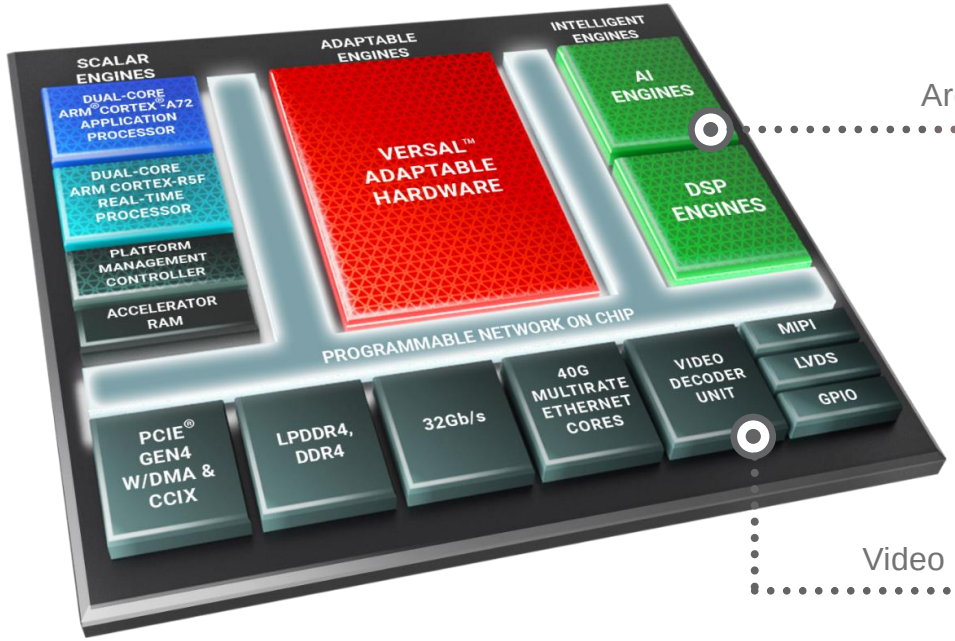
- Versal AI Core
- Versal Premium
- Versal AI Edge

COMPUTE ACCELERATION

- Scalar Engines
- Adaptable Engines
- Intelligent Engines

PLATFORM

- Software programmable NoC
- Platform Management Controller
- Dedicated Interfaces (e.g., PCIe, DDR)



AIE-ML Architecture

Video Decode Unit

Technology Node: TSMC 7nm FinFET

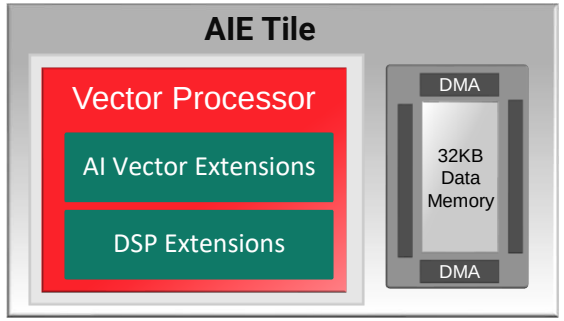
tsmc

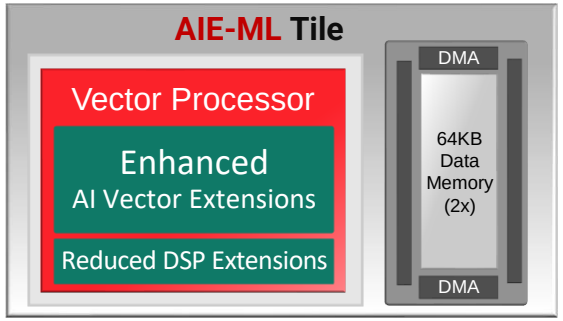
7nm

AMD

together we advance_

AIE Compute Architecture Features





		AIE Tile		AIE-ML Tile
Target Markets		Wireless 5G, AI/ML inference, A&D		AI/ML inference
Compute (Mults / tile)	BFLOAT16	—	New	128
	INT8	128	2X	256
	INT4	—	New	512
Tile Local Data Memory		32 KB	2X	64 KB
AIE Array Interconnect B/W		1X	1X	1X
Compression and Sparsity		No		Yes
Scratchpad On-Chip Memory		PL uRAM		AIE Memory (512KB/tile)

AIE Optimized for Machine Learning Inference

Compute core optimized for ML

- Doubled the multipliers, doubled INT8 performance
- Native support for INT4 and BFLOAT16

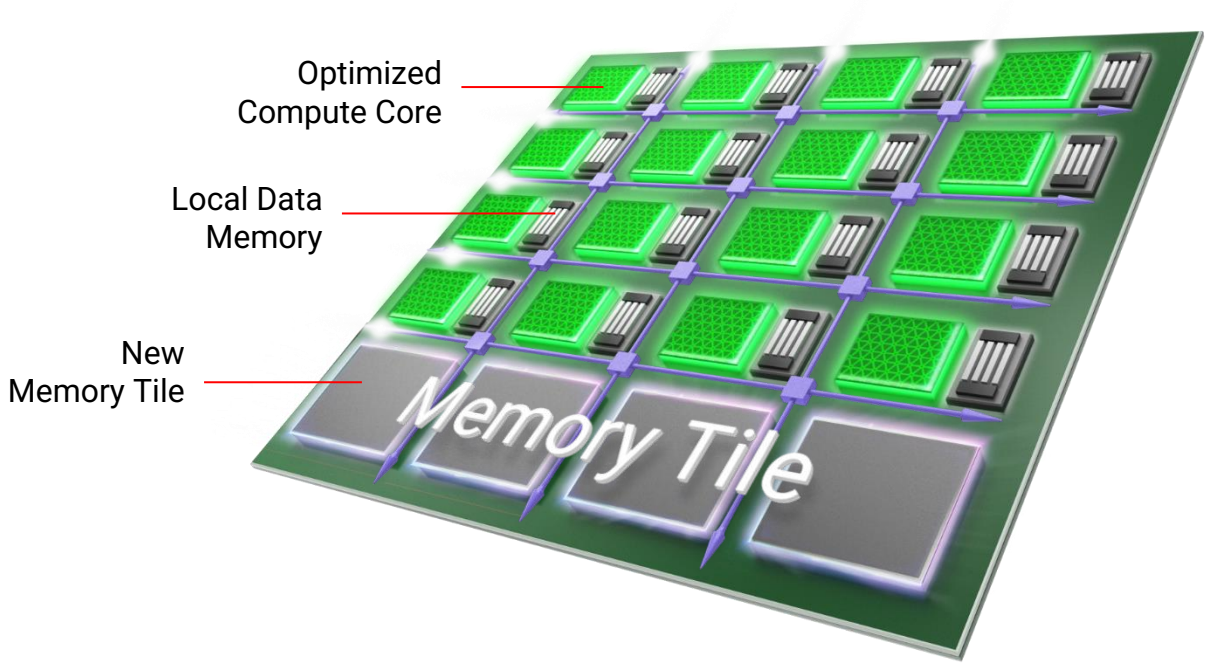
Compute local data memory

- 64 kB data memory local to compute core
- Improved localization of data

Memory Tiles

- Up to 38 Megabytes across the AIE array
- Higher bandwidth memory access

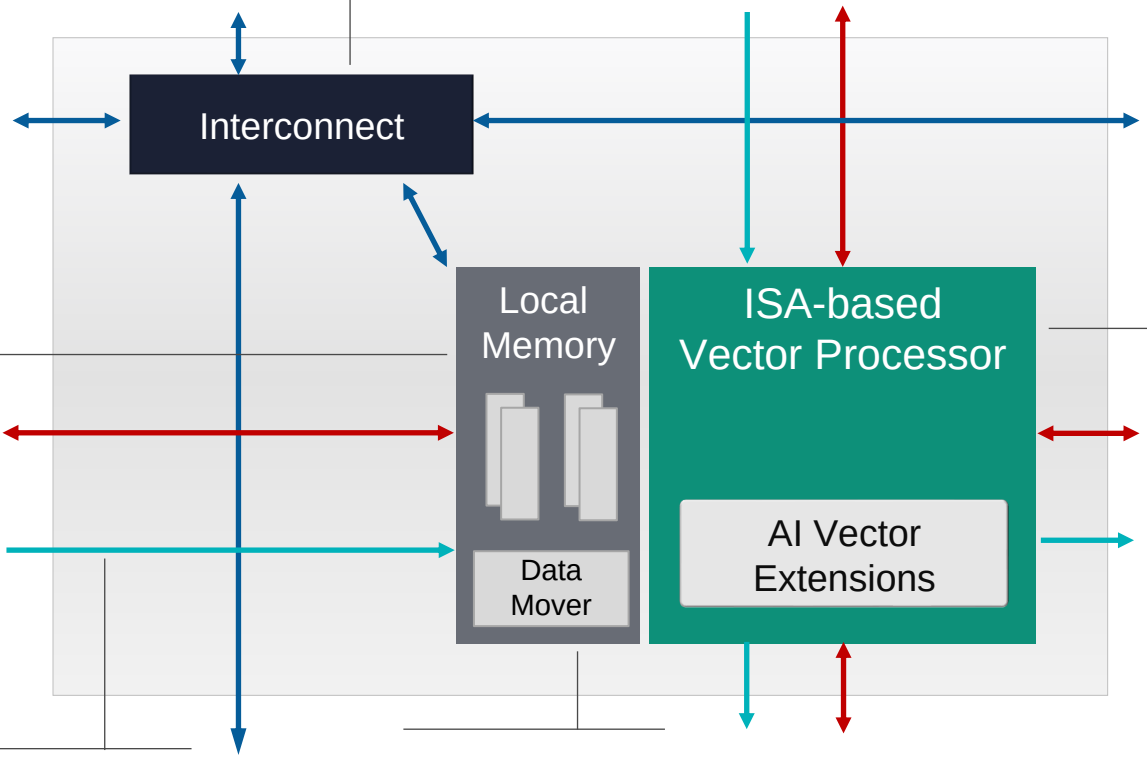
Optimized
AIE-ML Array



Scalable and Modular Architecture

AIE-ML Tile Architecture

Non-Blocking Interconnect
Up to 200+ GB/s bandwidth per tile



Local Memory
Multi-bank implementation
Shared across neighbor cores

ISA-based Vector Processor
Software Programmable
(e.g., C/C++)

Cascade Interface
Partial results to next core

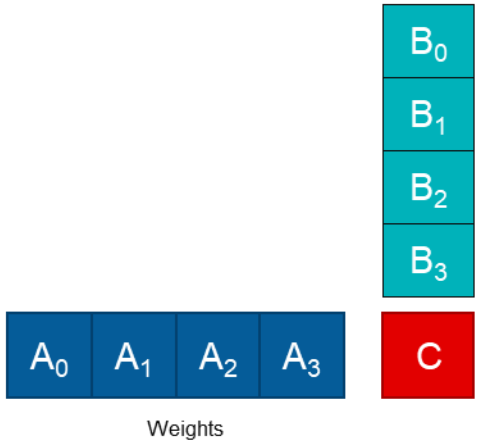
Data Mover (DMA)
Non-neighbor data communication
Integrated synchronization primitives

Flexible Dataflow: AIE-ML Array Examples

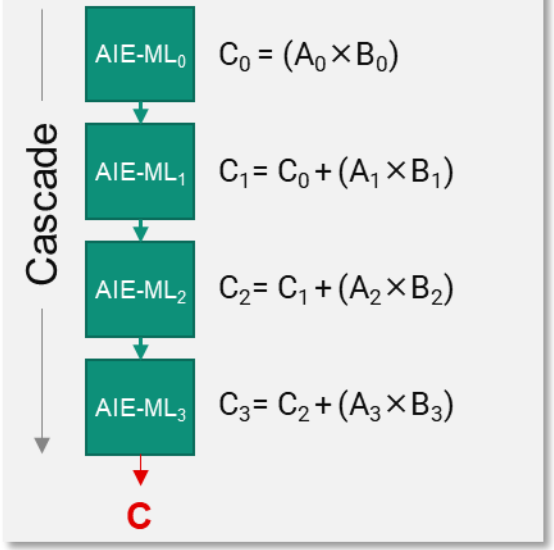
DATA REDUCTION

$$C = (A_0 \times B_0) + (A_1 \times B_1) + (A_2 \times B_2) + (A_3 \times B_3)$$

COMPUTE **SINGLE** OUTPUT MATRIX

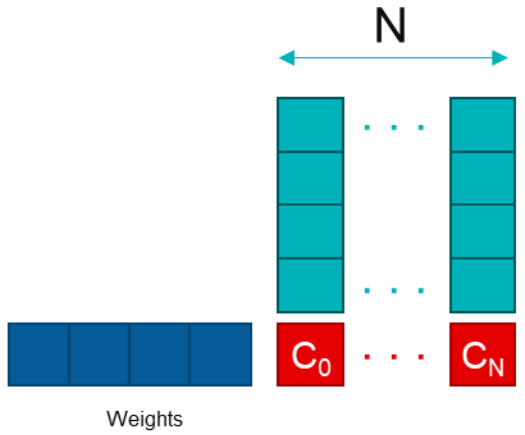


AIE-ML IMPLEMENTATION

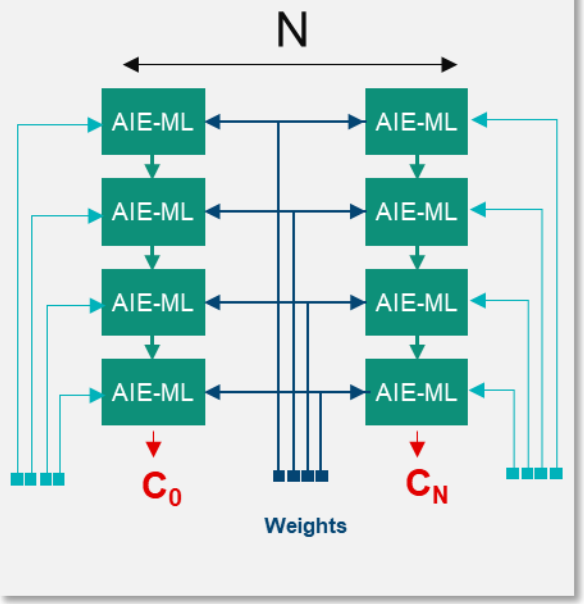


DATA MULTICASTING

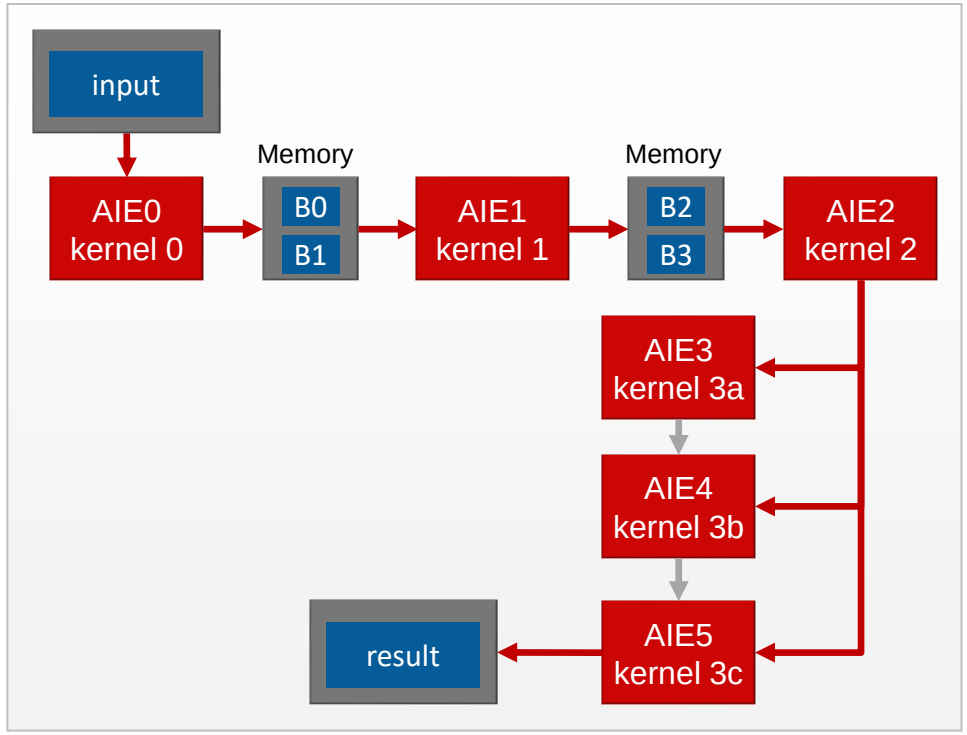
COMPUTE **MULTIPLE** OUTPUT MATRICES



AIE-ML IMPLEMENTATION

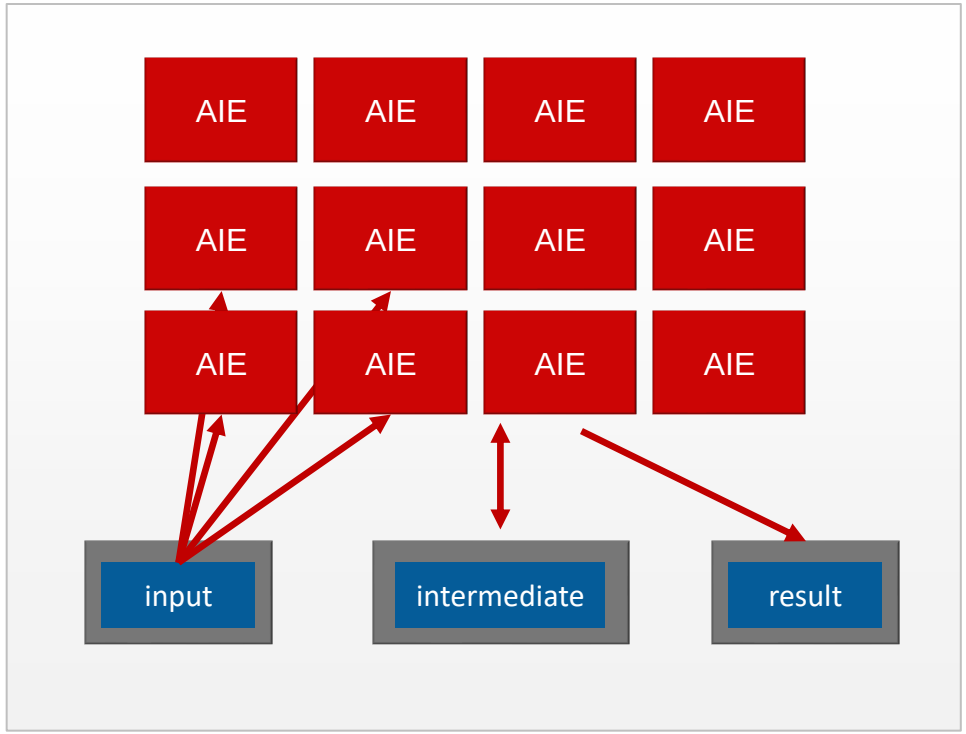


AIE Supported data-flow styles



Data-flow style

- application unrolled in space
- different PE working on different kernels



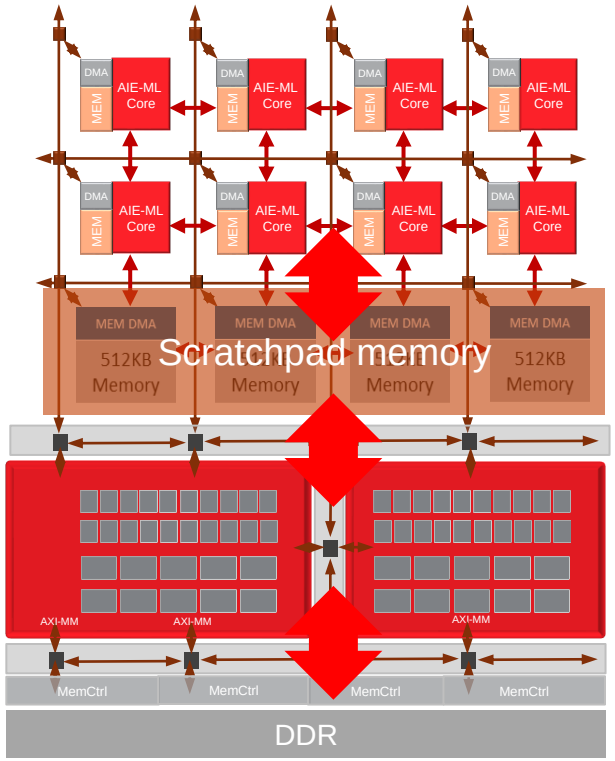
Distributed-kernel style

- application folded in time
- all PE working on same sub-kernel at the same time

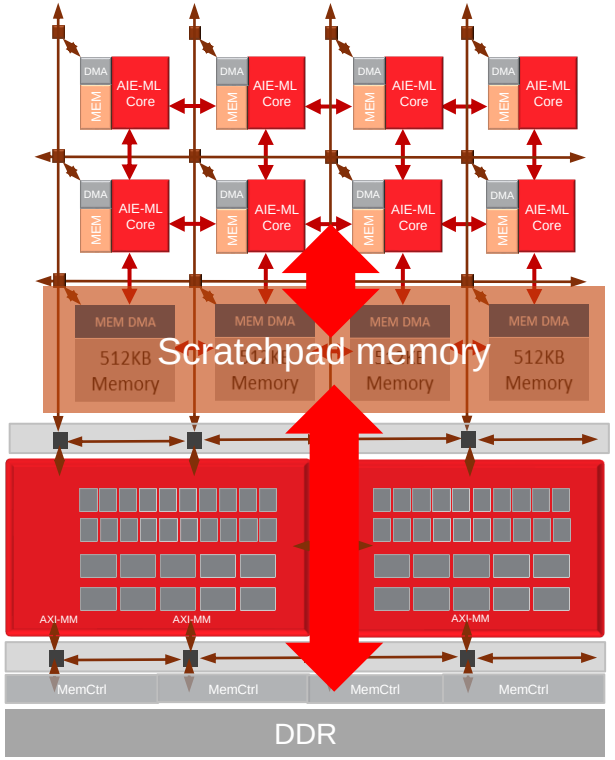
Hybrid of two: Coarse-Grain-Data-flow style

- Subset of whole application allocated to a block of AIE tiles (sub-array)
- Data-flow of intermediate data between these blocks

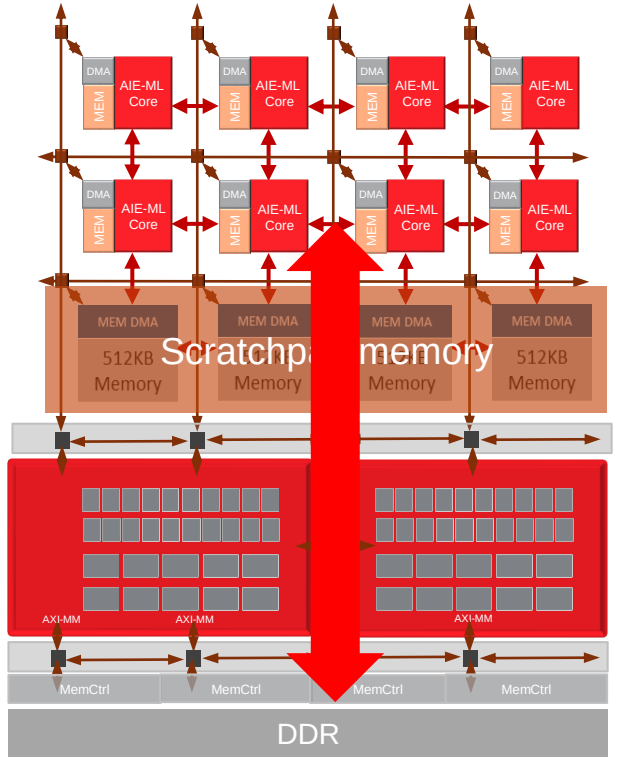
Flexible Dataflow: Device-level Examples



DDR ↔ Scratchpad using Programmable Logic



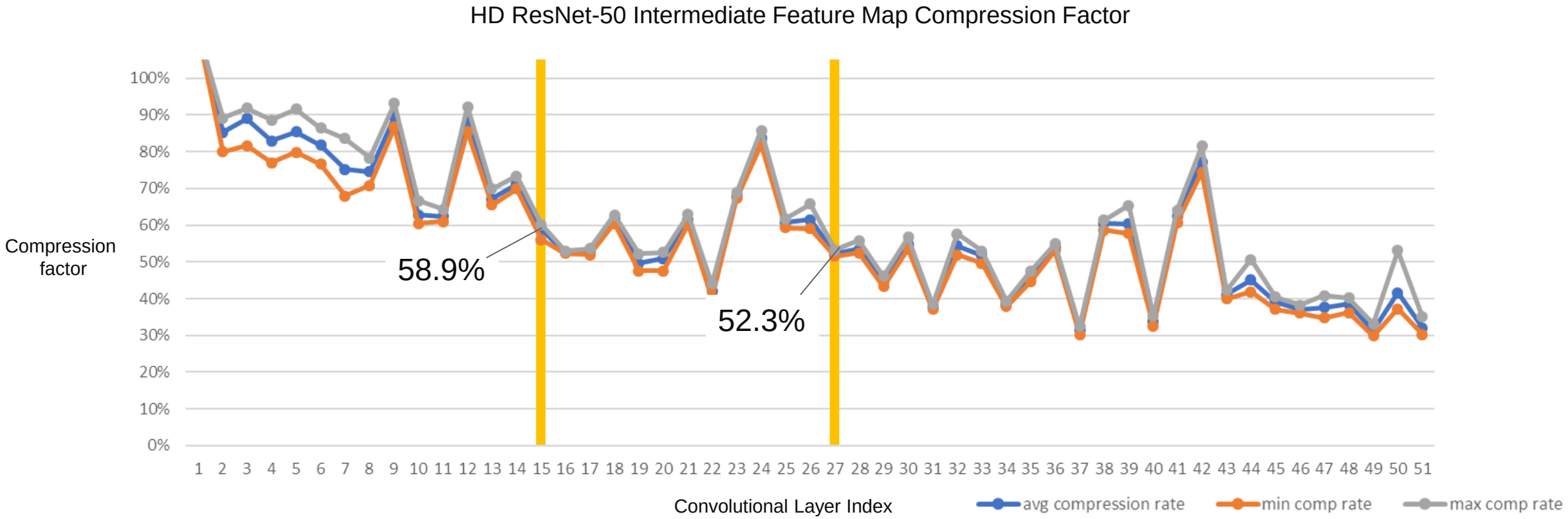
DDR ↔ Scratchpad using NoC



DDR ↔ AIE-ML Tiles using NoC

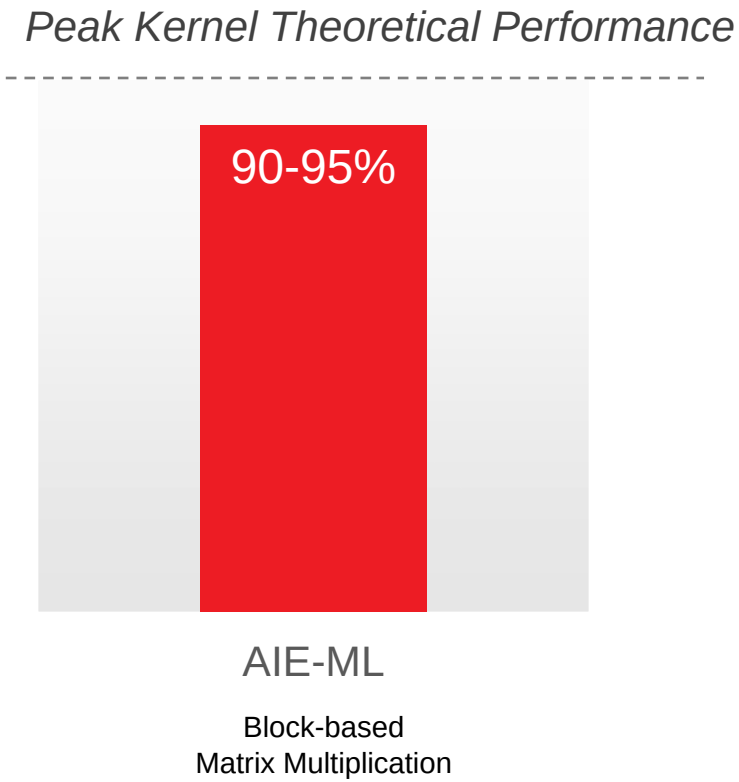
Compression to DDR: Acceleration of activation Spill/Restore

- Reduction in external memory bandwidth using compression
  - Quantized int8 weights, int8 activations
  - HD images require tiling, with spill/restore to DDR
    - Without compression **~56GB/s**
    - With compression **~29GB/s**

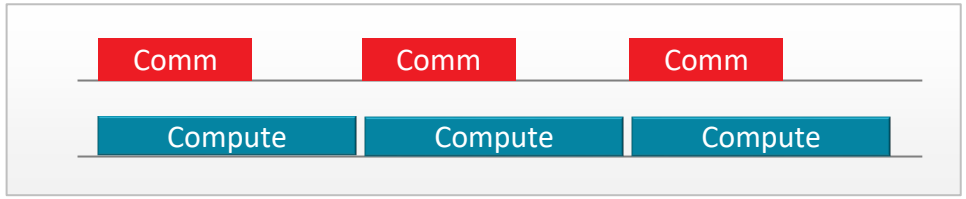


AIE-ML Delivers High Compute Efficiency

Vector Processor Efficiency

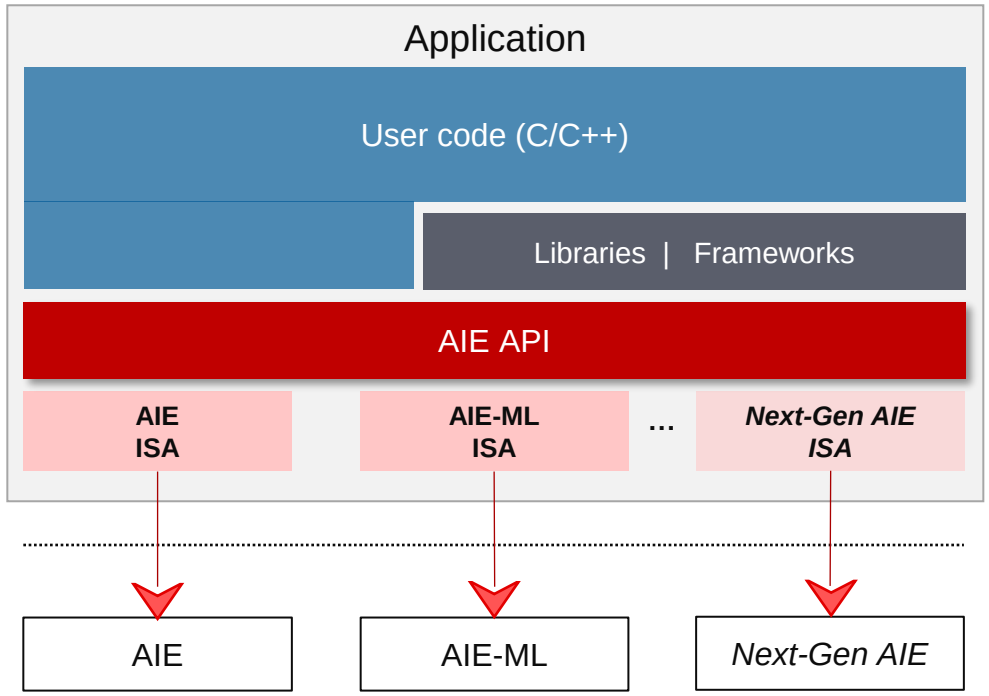


- Adaptable, non-blocking interconnect
  - Flexible data movement architecture
  - Avoids interconnect “bottlenecks”
- Adaptable memory hierarchy
  - Local, distributed, shareable = extreme bandwidth
  - No cache misses or data replication
  - Extend to PL memory (BRAM, URAM)
- Transfer data while AIE-ML computes in parallel



Overlap Compute and Communication

AIE Architecture: Source Code Portability



- Backward compatible
- Evolving
  - New devices can add new functionality
- Efficient
  - Applications can be optimized to maximize compute efficiency

Portable common high-level API across the AIE architecture

API Example

```
inline void elementwise_mul(int8* ifm0_ptr, int8* ifm1_ptr,
  int8* ofm_ptr, const params_t& params) {
    int shf_wr = 7 - params.shf_wr;
    int8* restrict i_0_ptr = ifm0_ptr;
    int8* restrict i_1_ptr = ifm1_ptr;
    int8* restrict out_ptr = ofm_ptr;
    //
    //Dynamically use a relu for output depending on parameter
    int8_t i8relu = params.relu == 1 ? 0 : -128;
    aie::vector<int8, 64> v64i8_relu = aie::broadcast<int8,
64>(i8relu);
    //
```

```
for (int i = 0; i < params.loop; i += 1)
  chess_loop_range(2, )
  chess_prepare_for_pipelining
  {
    aie::vector<int8,64> xvec = aie::load_v<64>( i_0_ptr);
    i_0_ptr+=64;
    aie::vector<int8,64> yvec = aie::load_v<64>( i_1_ptr);
    i_1_ptr+=64;
    aie::accum<acc32,64> acc0;
    aie::vector<int8,64> out_v;

    acc0 = aie::mul( xvec, yvec );

    out_v = acc0.template to_vector<int8_t>(shf_wr);

    out_v = aie::max(v64i8_relu, out_v);
    aie::store_v( (int8*)out_ptr, out_v); out_ptr+=64;
  }
}
```

AMD UNIFIED AI STACK 1.0

Unified Inference Frontend (UIF) for AI Developers

Optimized Inference Models

WinML* | ONNX Runtime | Pytorch | TensorFlow

Unified AI Development & Deployment Tools (Quantization, Pruning, Inference Server)

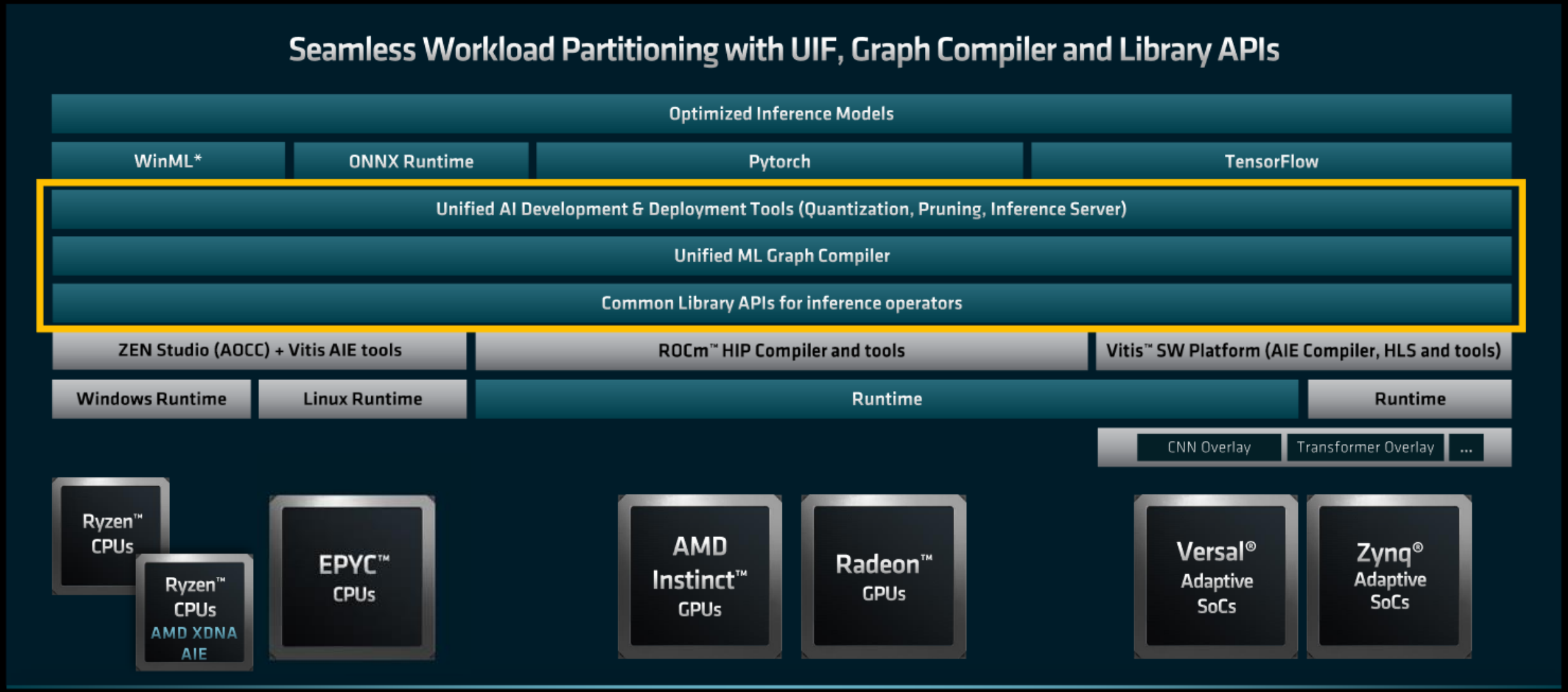
ML Graph Compiler	MIGraphX: ML Graph Compiler	Vitis AI ML Graph Compiler
ZenDNN, AOCL Optimized Library	MIOpen, ROCBLAS	Vitis ML Libraries
ZEN Studio (AOCC)	ROCm™ HIP Compiler and tools	Vitis™ SW Platform (AIE Compiler, HLS and tools)
Windows Runtime / Linux Runtime	Runtime	Runtime
		CNN Overlay / Transformer Overlay / ...

Ryzen™ CPUs	EPYC™ CPUs	AMD Instinct™ GPUs	Radeon™ GPUs	Ryzen™ CPUs AMD XDNA AIE	Versal® Adaptive SoCs	Zynq® Adaptive SoCs

*WinML for Ryzen Only

All roadmaps are subject to change.

AMD UNIFIED AI STACK 2.0



*WinML for Ryzen Only

All roadmaps are subject to change.

Collaborate – How to Get Your Hands on It

Supporting High End Compute Research

HACC community

Remote access to Adaptive Compute hardware

Growing community of over 350 researchers at over 100 institutions

www.amd-haccs.io

AMD University Program

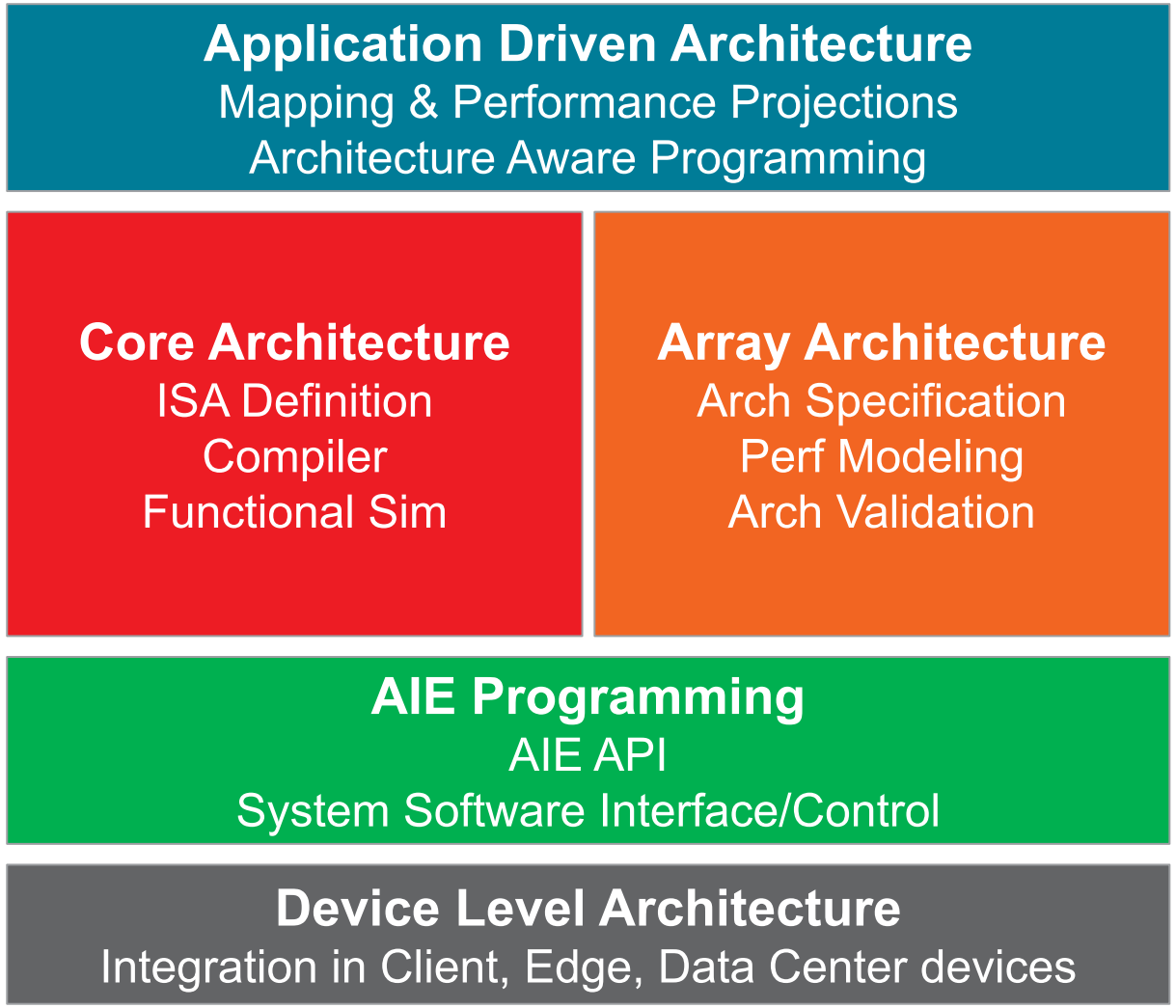
Development Boards

Donation Program

VCK5000

www.xilinx.com/support/university.html

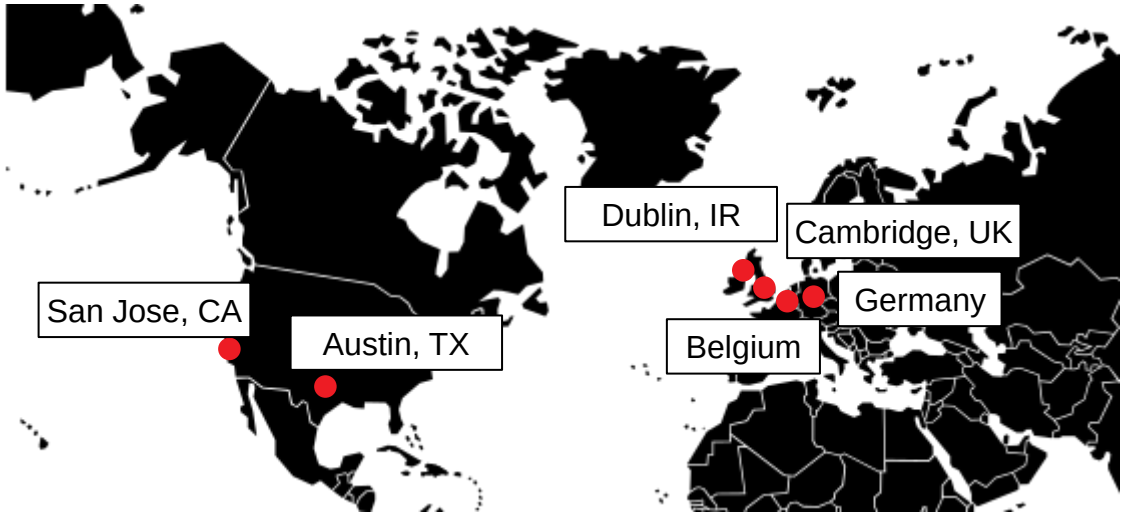
AIE Architecture Team



Join the AMD AIE Architecture Team

- Main sites in San Jose and Dublin
- Hiring across geographies and teams

Talk to me, reach out alex.rico@amd.com




AMD