# Introduction to the AMD Versal™ AI Engine Architecture

2024.1

AMD

together we advance_

# Objectives

After completing this module, you will be able to:

OBJECTIVE 01

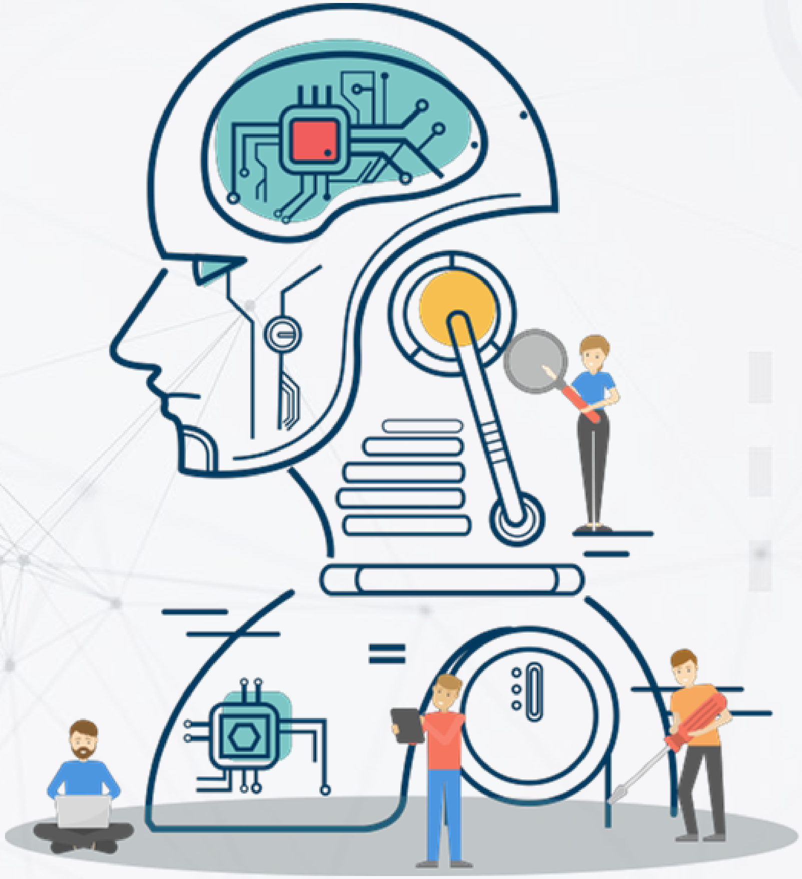
Identify the major blocks in the AMD Versal™ architecture

OBJECTIVE 02

Describe the architecture of the Versal AI Engine as well as its interfaces

OBJECTIVE 03

Describe the AI Engine memory modules and memory access structure



# Versal Architecture: Overview

Processor System
- Platform control
- Embedded edge compute

Programmable Logic
- High compute density
- Voltage scaling for perf/watt

AI Engine
- AI compute
- Diverse DSP workloads

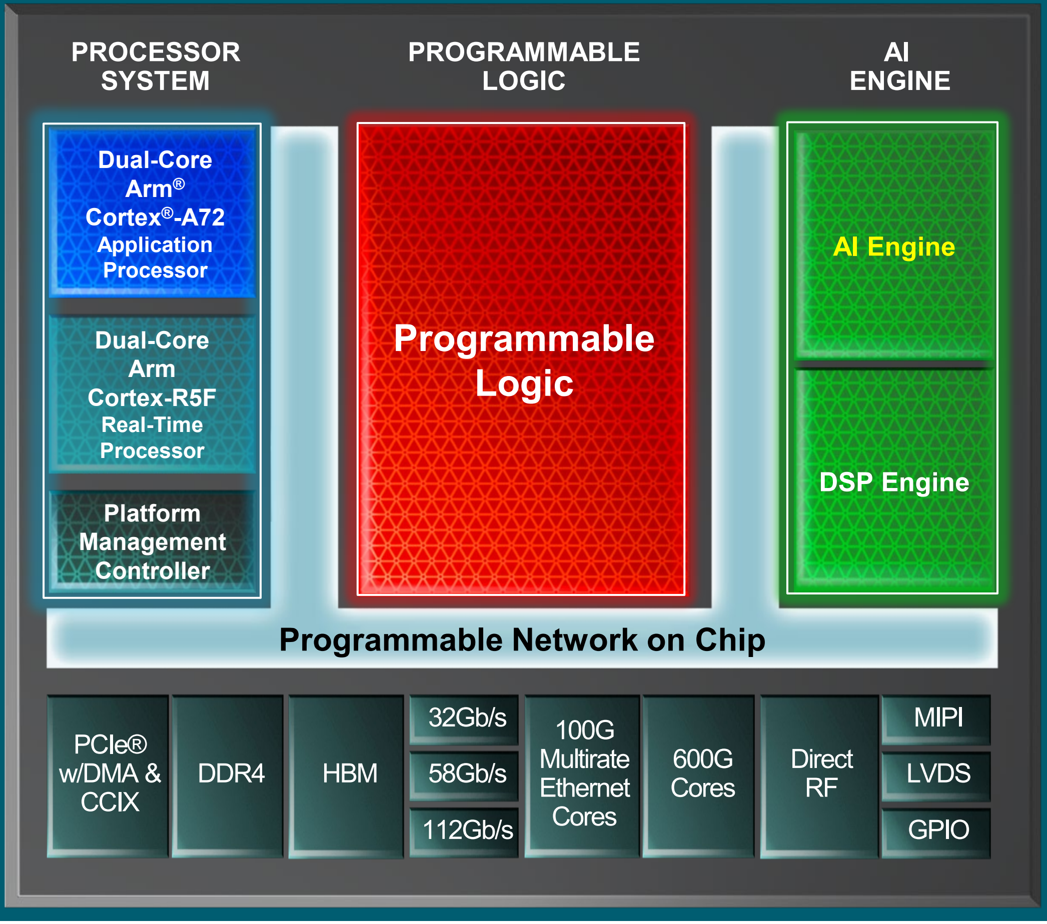
## Connectivity

**PCIe Gen5 & CCIX**
- High PCIe and DMA bandwidth
- Cache coherency

**Protocol Engines**
- 400G/600G cores
- Power optimized

**Programmable I/O**
- Any interface or sensor
- Includes 3.2Gb/s MIPI

**Transceiver Leadership**
- Broad range, 25G →112G
- 58G in mainstream devices



PROCESSOR SYSTEM

Dual-Core Arm® Cortex®-A72 Application Processor

Dual-Core Arm Cortex-R5F Real-Time Processor

Platform Management Controller

PROGRAMMABLE LOGIC

Programmable Logic

AI ENGINE

AI Engine

DSP Engine

Programmable Network on Chip

PCIe® w/DMA & CCIX

DDR4

HBM

32Gb/s

58Gb/s

112Gb/s

100G Multirate Ethernet Cores

600G Cores

Direct RF

MIPI

LVDS

GPIO

## Various Domain Connections

**Programmable NoC**
- Guaranteed bandwidth
- Enables SW programmability

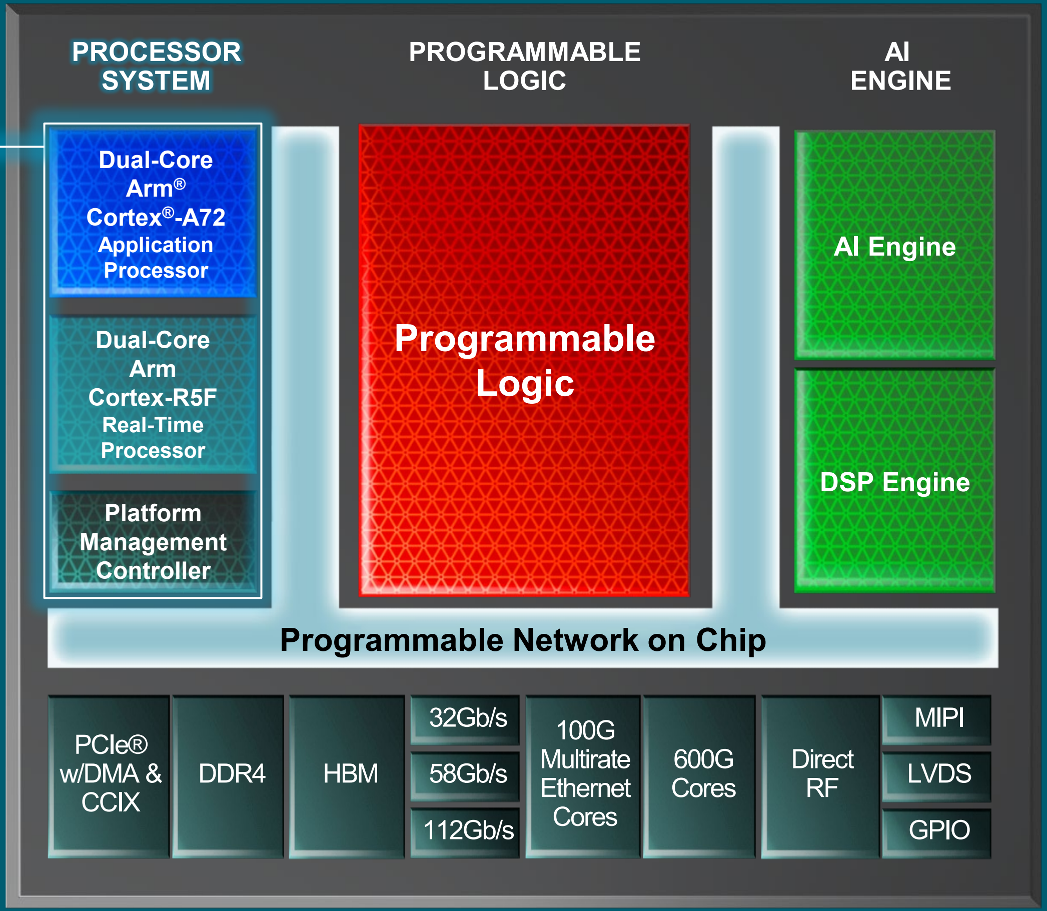
## Memory Options

**DDR4 Memory**
- 3200-DDR4, 4266-LPDDR4
- High bandwidth/pin

# Versal Architecture: Overview

## Processor System

- Execute complex algorithms and decision making
- Provide safety processing and redundancy for mission- and safety-critical applications
- Manage the entire platform
- Load each aspect of the Versal™ device and monitor status
- Support capability extension
  - PL-instantiated MicroBlaze™ processor

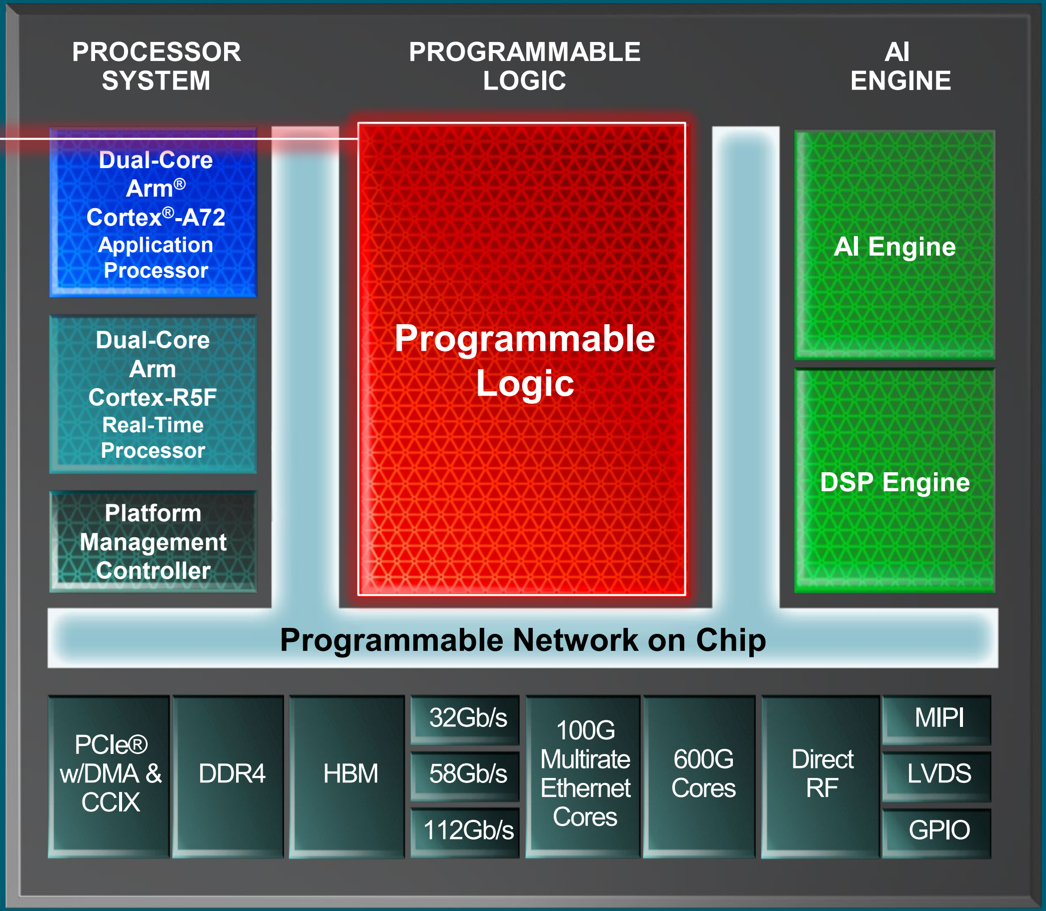

PROCESSOR SYSTEM
PROGRAMMABLE LOGIC
AI ENGINE
Dual-Core Arm® Cortex®-A72 Application Processor
Dual-Core Arm Cortex-R5F Real-Time Processor
Platform Management Controller
Programmable Logic
AI Engine
DSP Engine
Programmable Network on Chip
PCIe® w/DMA & CCIX
DDR4
HBM
32Gb/s
58Gb/s
112Gb/s
100G Multirate Ethernet Cores
600G Cores
Direct RF
MIPI
LVDS
GPIO

AMD together we advance_

# Versal Architecture: Overview

## Programmable Logic

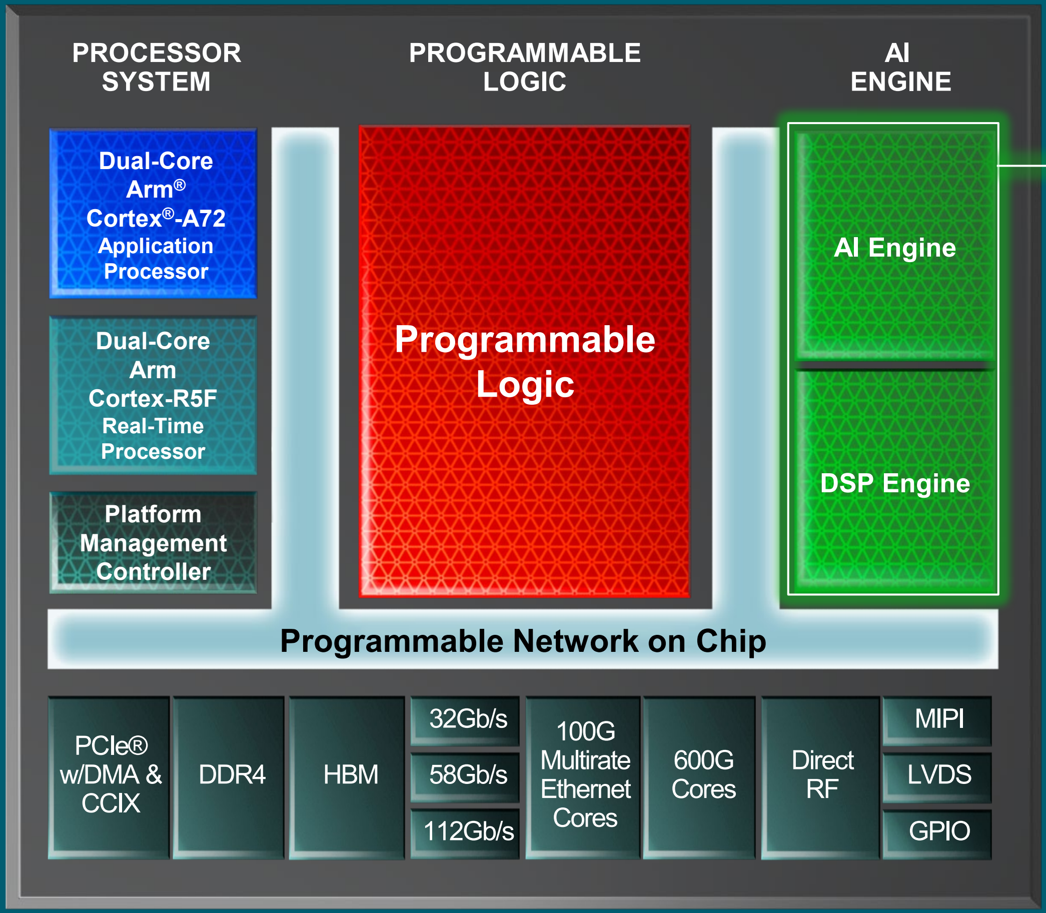
- Millions of reconfigurable system logic cells
- Support for parallel, pipelined, and hybrid architectures
- Wide variety of memory elements


PROCESSOR SYSTEM
PROGRAMMABLE LOGIC
AI ENGINE
Dual-Core Arm® Cortex®-A72 Application Processor
Dual-Core Arm Cortex-R5F Real-Time Processor
Platform Management Controller
Programmable Logic
AI Engine
DSP Engine
Programmable Network on Chip
PCIe® w/DMA & CCIX
DDR4
HBM
32Gb/s
58Gb/s
112Gb/s
100G Multirate Ethernet Cores
600G Cores
Direct RF
MIPI
LVDS
GPIO

AMD
together we advance_

# Versal Architecture: Overview

*This is not a physical representation!*


PROCESSOR SYSTEM
PROGRAMMABLE LOGIC
AI ENGINE
Dual-Core Arm® Cortex®-A72 Application Processor
Dual-Core Arm Cortex-R5F Real-Time Processor
Platform Management Controller
Programmable Logic
AI Engine
DSP Engine
Programmable Network on Chip
PCIe® w/DMA & CCIX
DDR4
HBM
32Gb/s
58Gb/s
112Gb/s
100G Multirate Ethernet Cores
600G Cores
Direct RF
MIPI
LVDS
GPIO

## AI Engine

- AI compute
- Diverse DSP workloads

### AI Is Everywhere

- Wired communications, automotive, and consumer markets

### DSP Engines

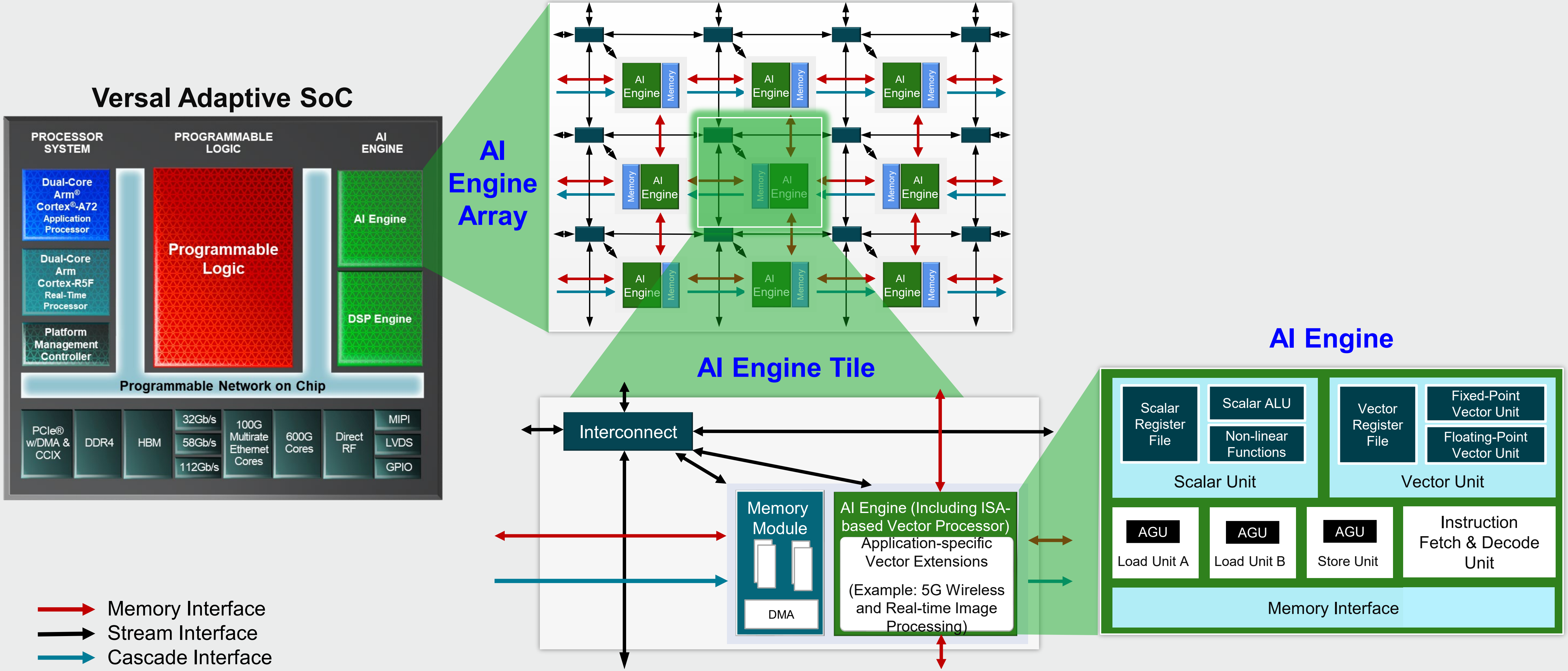
- High-precision, floating-point computation support
- Offload additional functions for acceleration

### AI Engines

- High throughput, low latency, and power efficient
- Ideal for AI inference and advanced signal processing
- Array of VLIW processors with SIMD vector units
- Instruction-level and data-level parallelism

AMD
together we advance_

# AI Engine: Terminology



# AI Engine: Array Architecture

## Modular & Scalable Architecture

- More tiles = more compute
- Up to 400 per device in Versal™ AI Core series
- Up to 472 per device in Versal Premium series

## Distributed Memory Hierarchy

- Maximize memory bandwidth

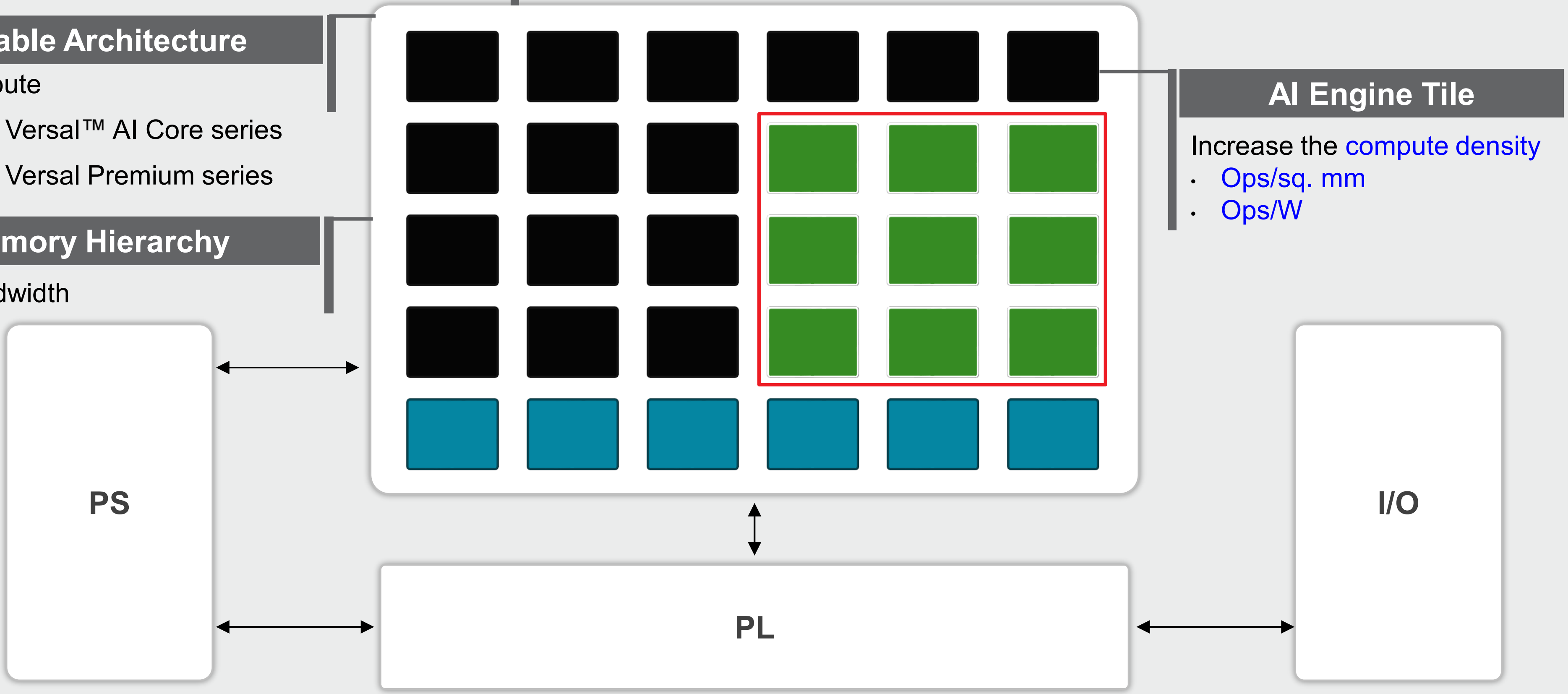
## AI Engine Array

- Top-level hierarchy of AI Engine architecture
- Integrates 2D array of AI Engine tiles

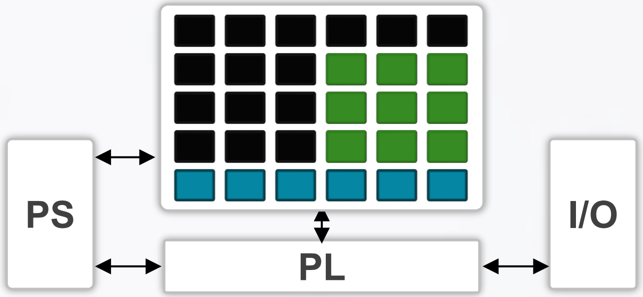
## AI Engine Tile

Increase the compute density

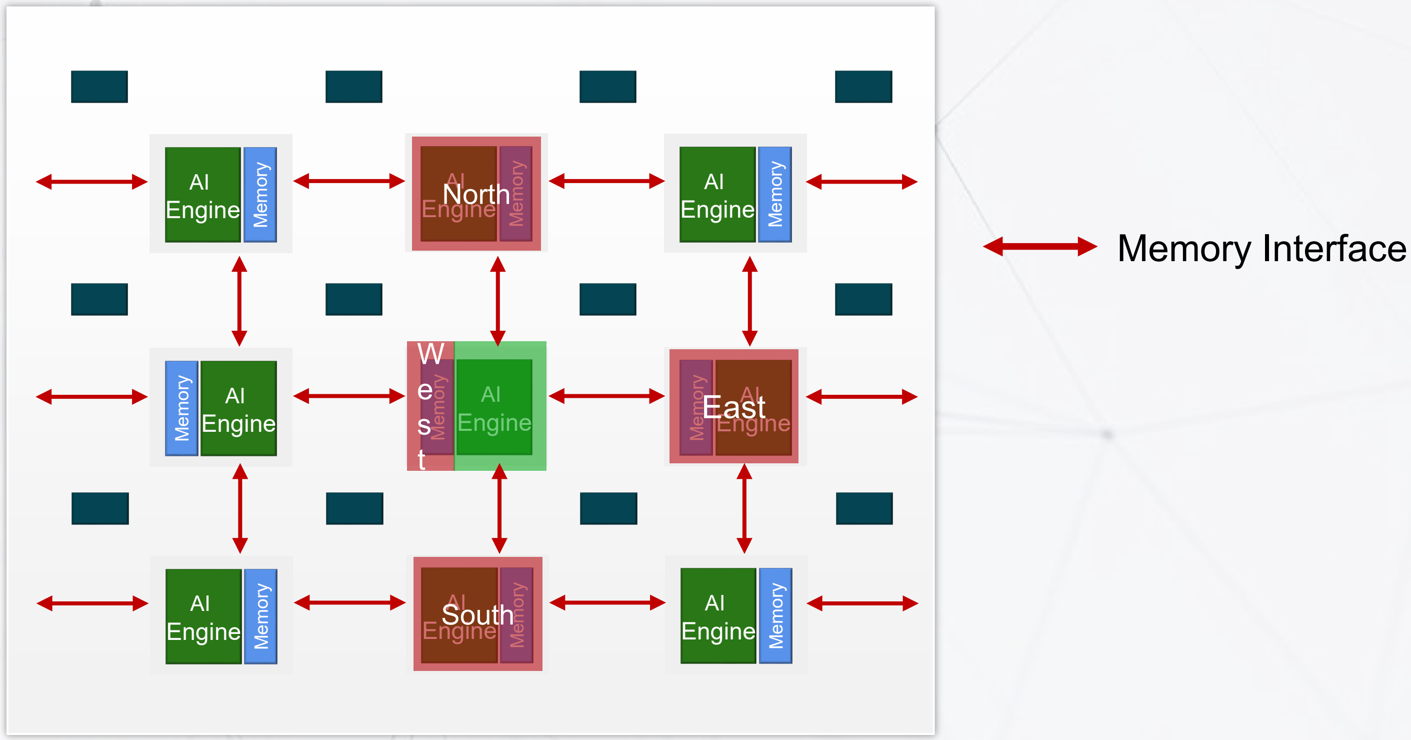
- Ops/sq. mm
- Ops/W



# AI Engine: Array Architecture

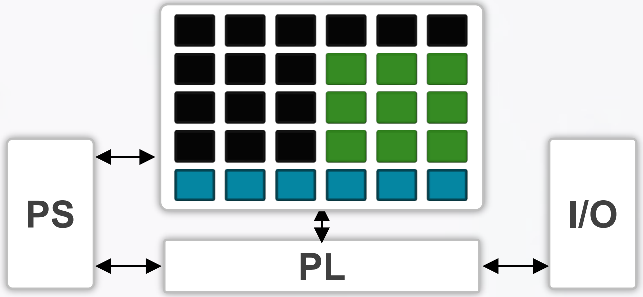


3 x 3 array of AI Engines tiles

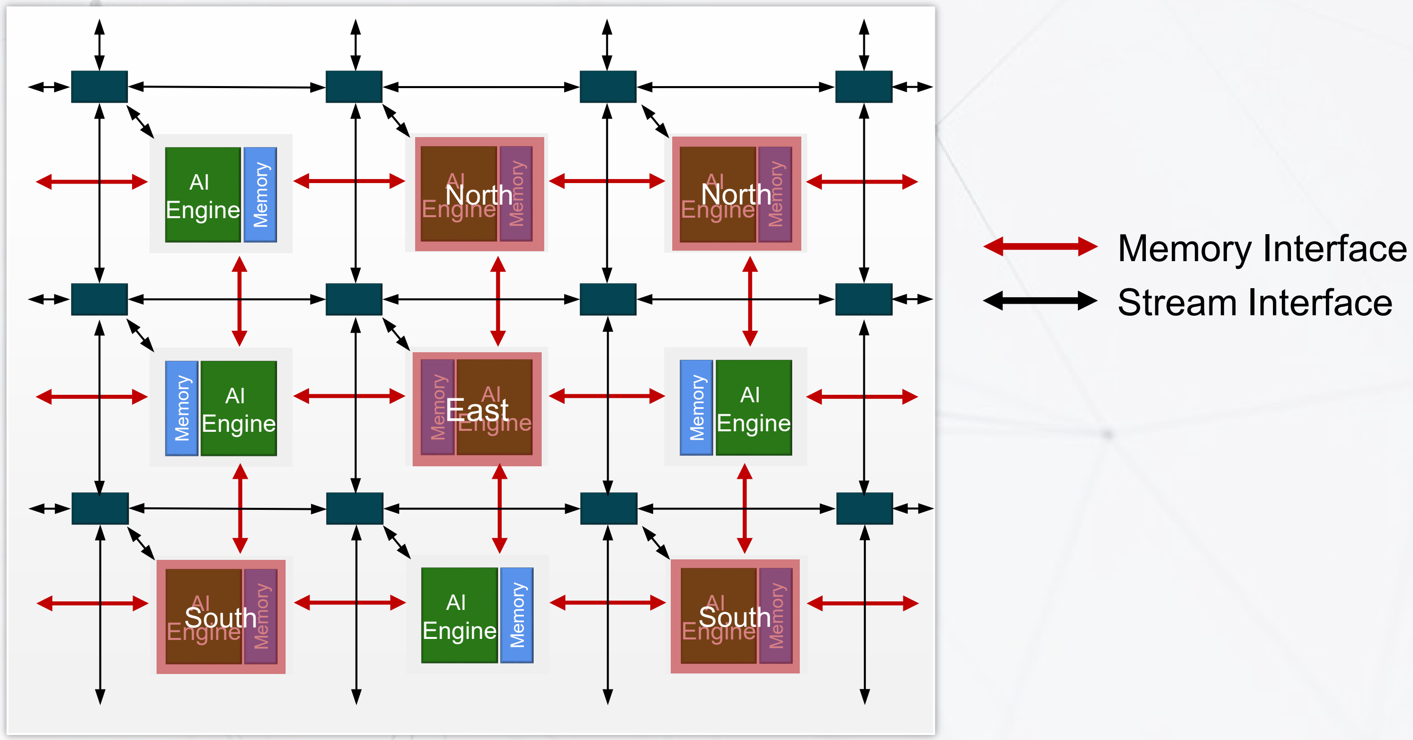


The tile can communicate to the local memory on each neighboring tile: north, south, east, or west

# AI Engine: Array Architecture

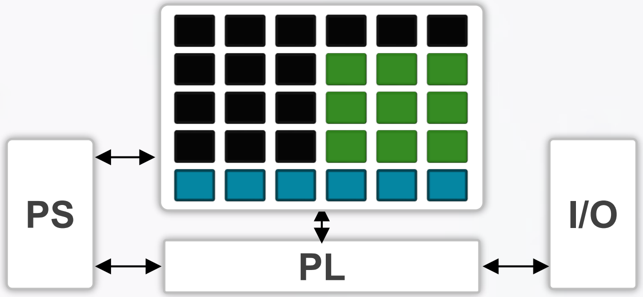


## 3 x 3 array of AI Engines tiles

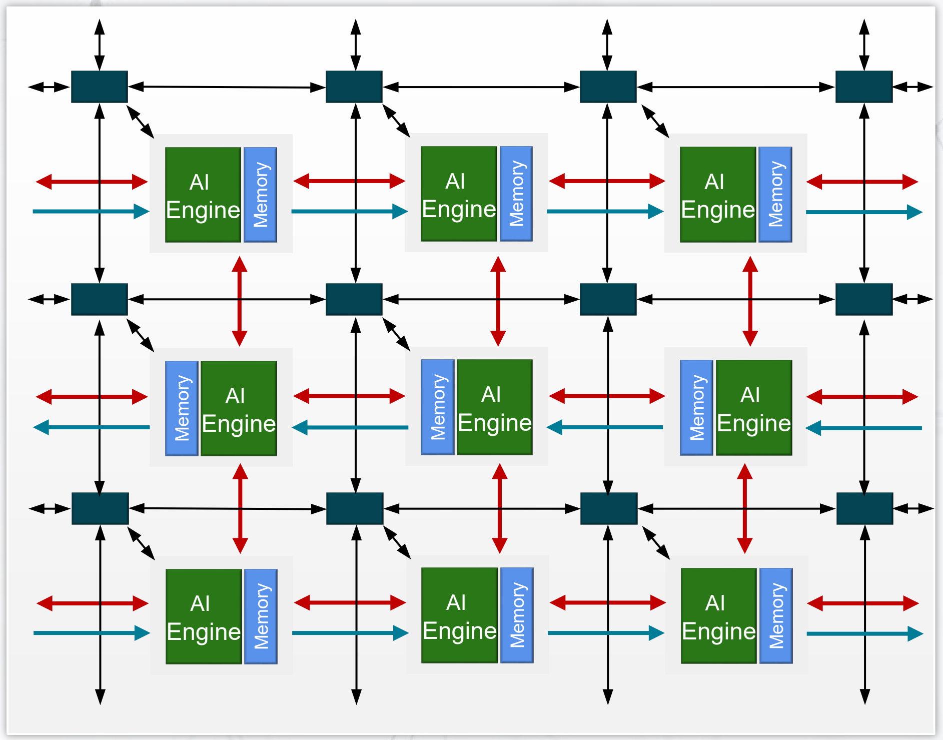


The non-blocking interconnect allow data communications between non-neighboring tiles throughout the array.

# AI Engine: Array Architecture

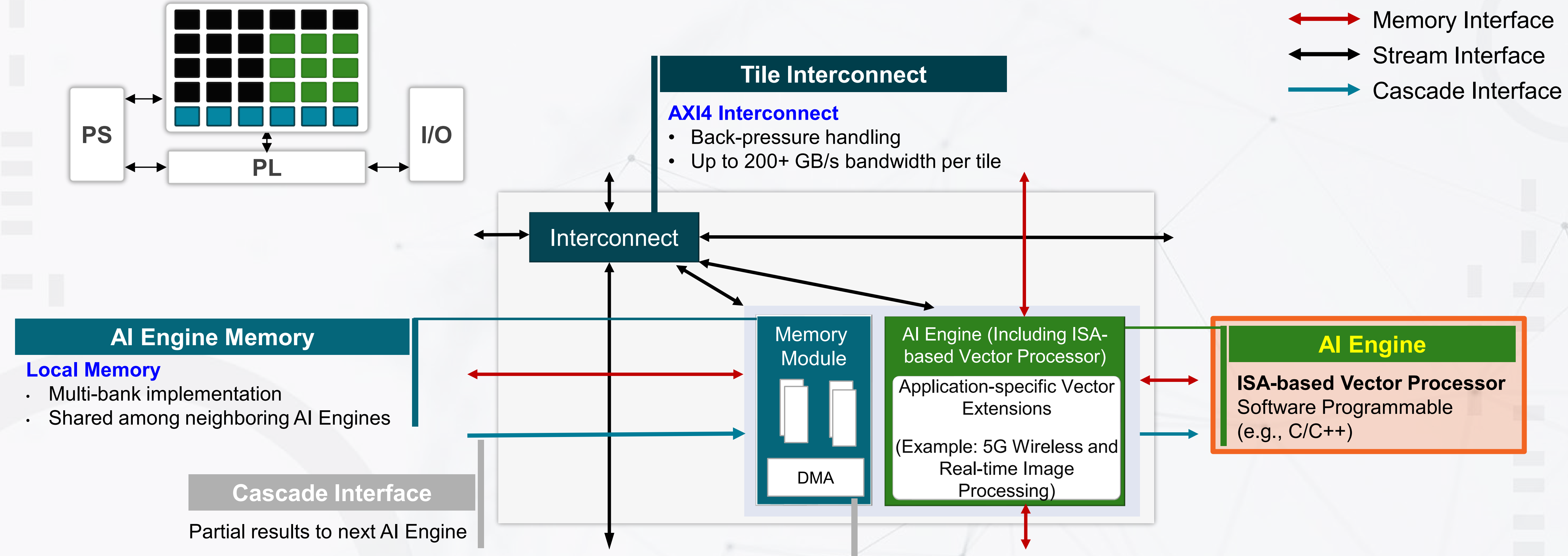


3 x 3 array of AI Engines tiles



Similarly, the blue arrows indicate the cascading of partial compute to neighboring tiles.

# AI Engine: Tile-based Architecture


PS
PL
I/O
Memory Interface
Stream Interface
Cascade Interface
Interconnect
Memory Module
DMA
AI Engine (Including ISA-based Vector Processor)
Application-specific Vector Extensions
(Example: 5G Wireless and Real-time Image Processing)

## Tile Interconnect

**AXI4 Interconnect**

- Back-pressure handling
- Up to 200+ GB/s bandwidth per tile

## AI Engine Memory

**Local Memory**

- Multi-bank implementation
- Shared among neighboring AI Engines

## AI Engine

**ISA-based Vector Processor**

Software Programmable (e.g., C/C++)

## Cascade Interface

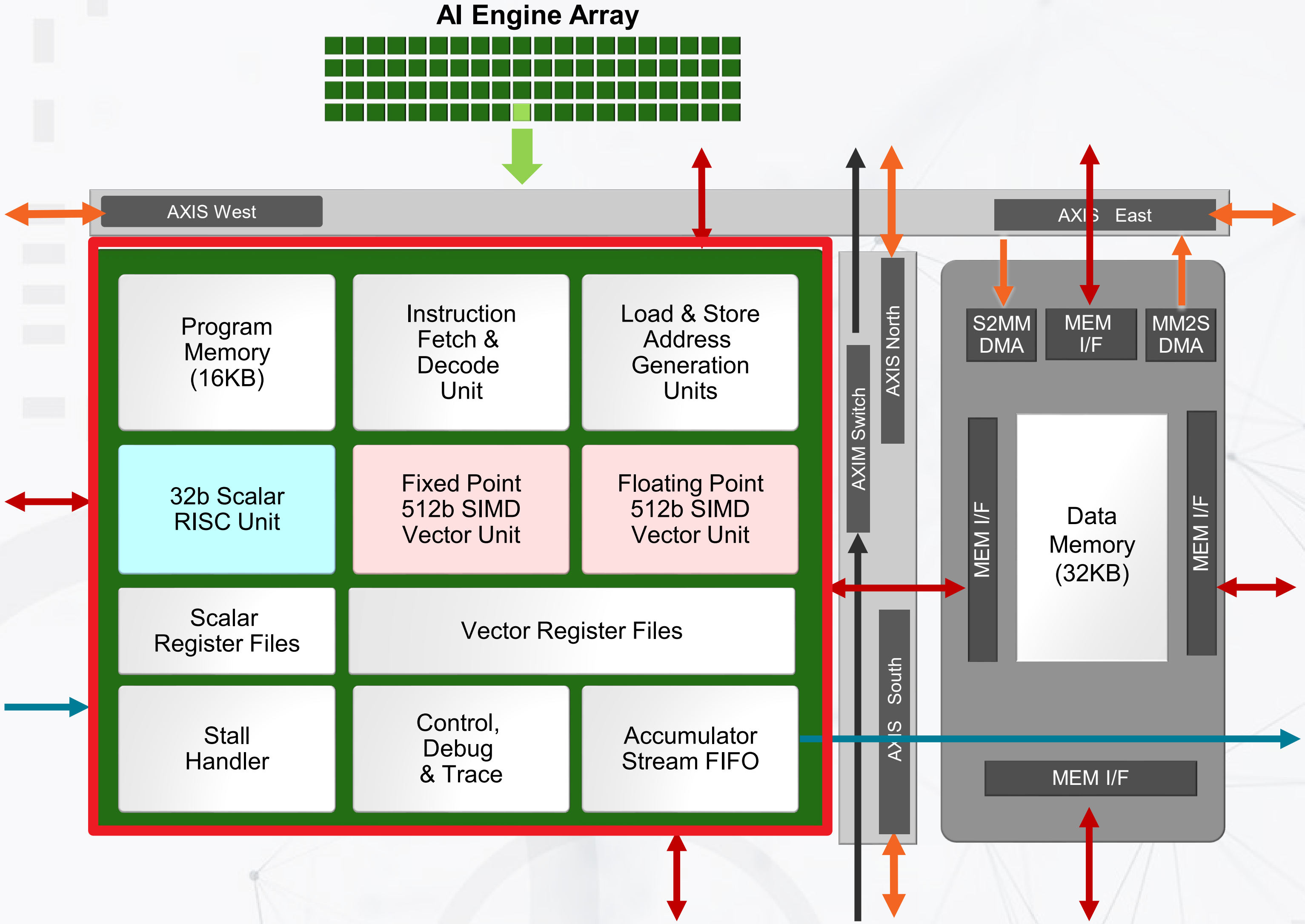
Partial results to next AI Engine

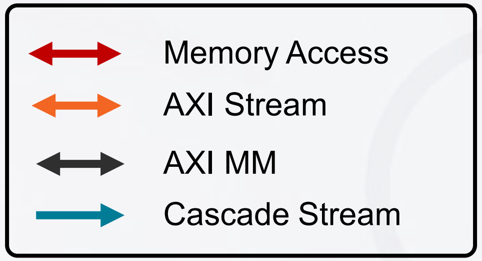
## DMA

- Non-neighbor data communication
- Integrated synchronization primitives

**AI Engine Tile**

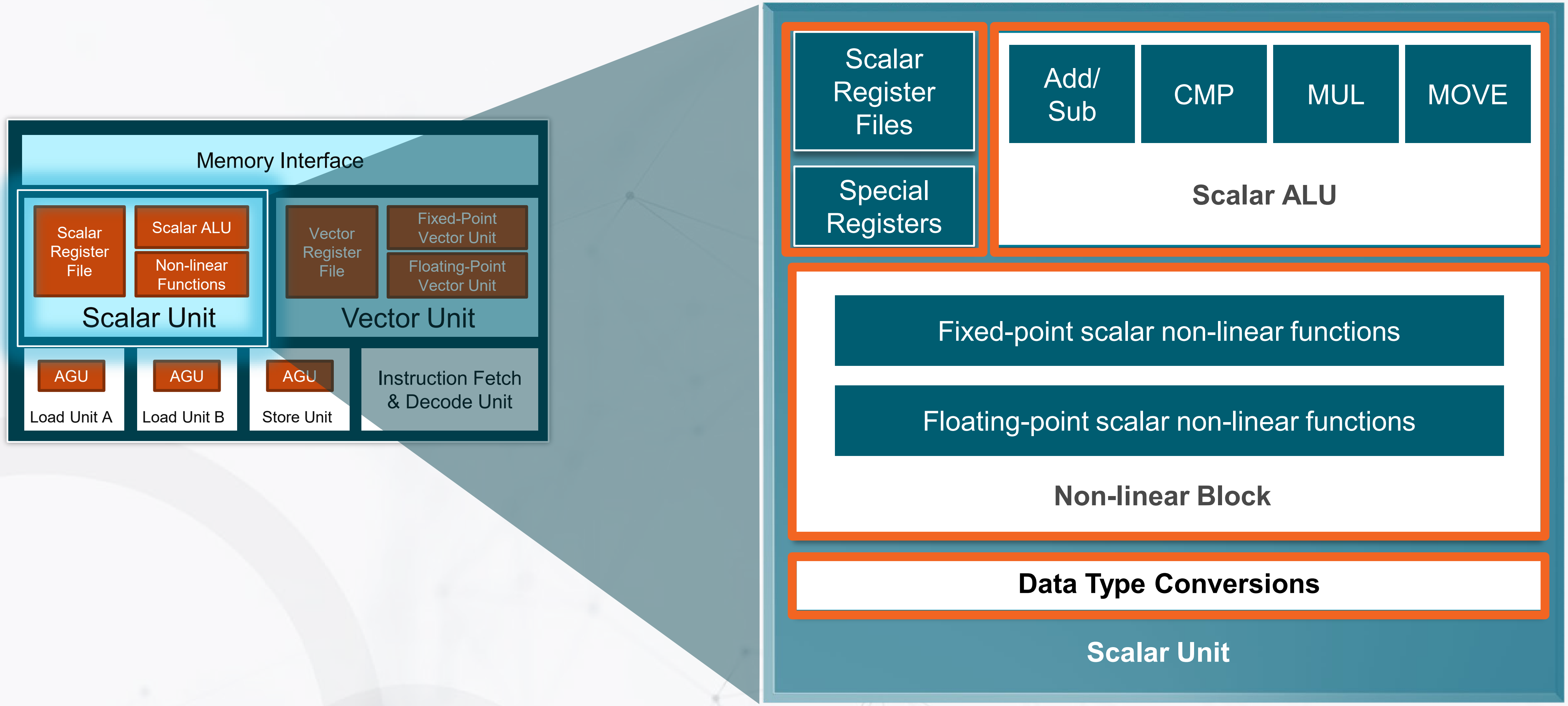
# AI Engine: Processor





- VLIW processor with 512-bit SIMD vector units
  - Application-specific vector extensions
    - Optimized for DSP, 5G, etc.
  - Hardened in 7nm @ 1 GHz
  - Software programmable (e.g., C/C++)
- Debug/trace/profile functionality
  - Debug using memory-mapped AXI4 interface
  - Connect to PMC via JTAG or HSDP

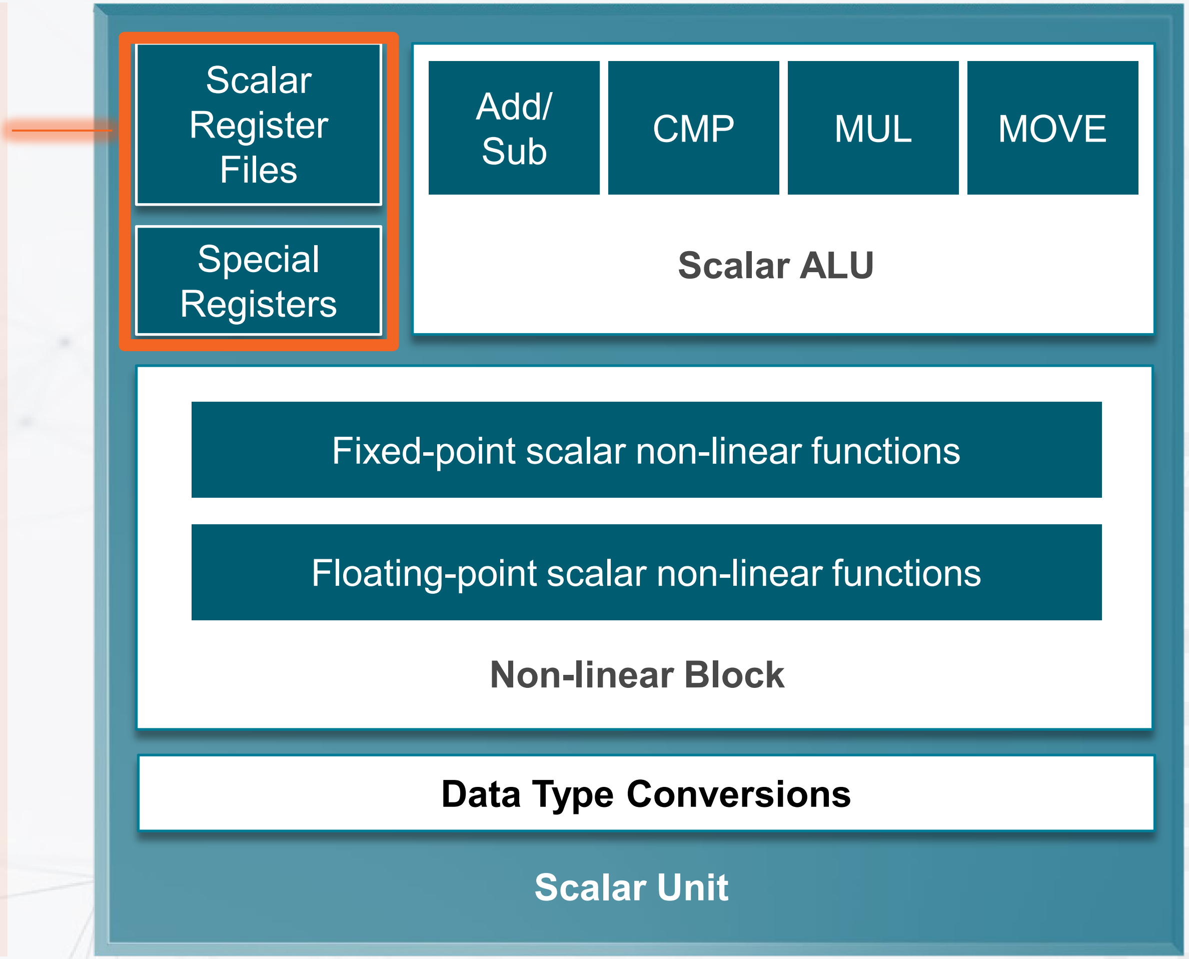
# AI Engine – Scalar Unit



# Scalar Unit – Registers

Scalar registers include configuration registers

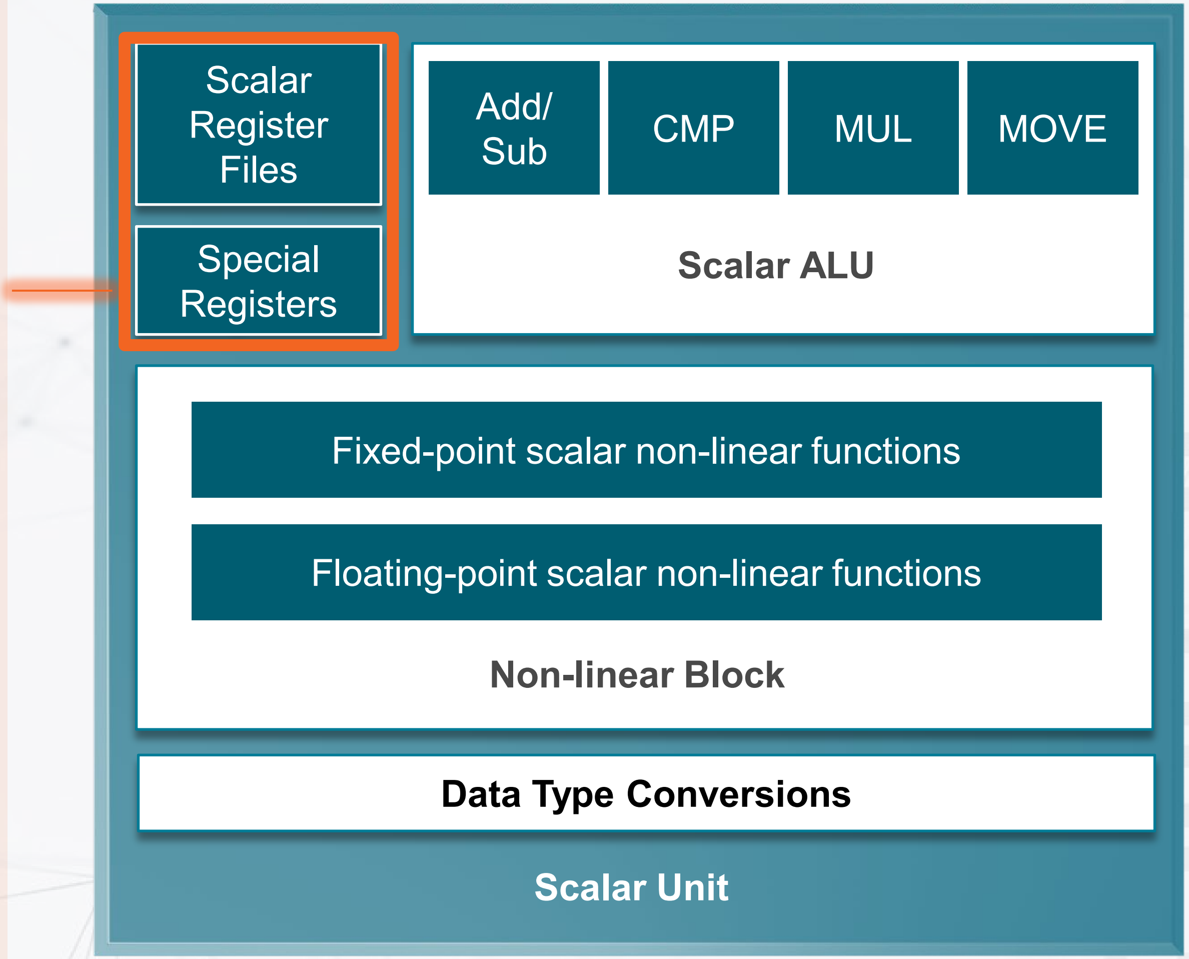
| Syntax | No. of Bits | Description |
|---|---|---|
| r0….r15 | 32 bits | General-purpose registers |
| m0…m7 | 20 bits | Modifier registers |
| p0…p7 | 20 bits | Pointer registers |
| cl0…cl7 | 32 bits | Configuration registers |
| ch0…ch7 | | |
| c0…c7 | 64 bits | |



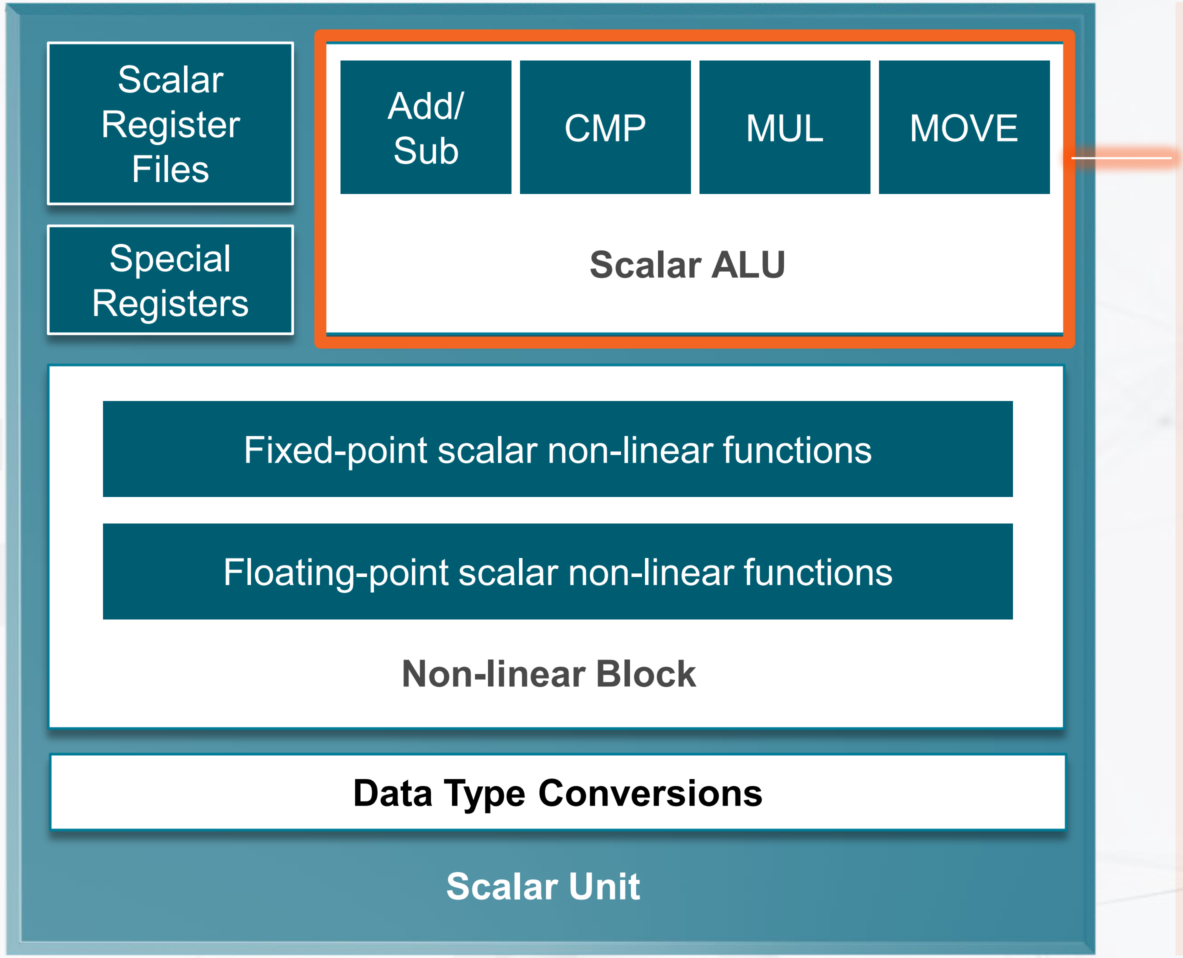
# Scalar Unit – Registers

## Special registers

| Syntax | No. of Bits | Description |
|---|---|---|
| cb0…cb7 | 20 bits | Circular buffer start address |
| cs0…cs7 | 20 bits | Circular buffer size |
| wcs0…wcs3 | 40 bits | Wide circular buffer size |
| s0…s7 | 8 bits | Shift control |
| sp | 20 bits | Stack pointer |
| lr | 20 bits | Link register |
| pc | 20 bits | Program counter |
| fc | 20 bits | Fetch counter |
| mc0…mc1 | 32 bits | Status register |
| md0…md1 | 32 bits | Mode control register |
| ls | 20 bits | Loop start |
| le | 20 bits | Loop end |
| lc | 32 bits | Loop count |
| lci | 32 bits | Loop count (PCU) |
| S | 8 bits | Shift control |



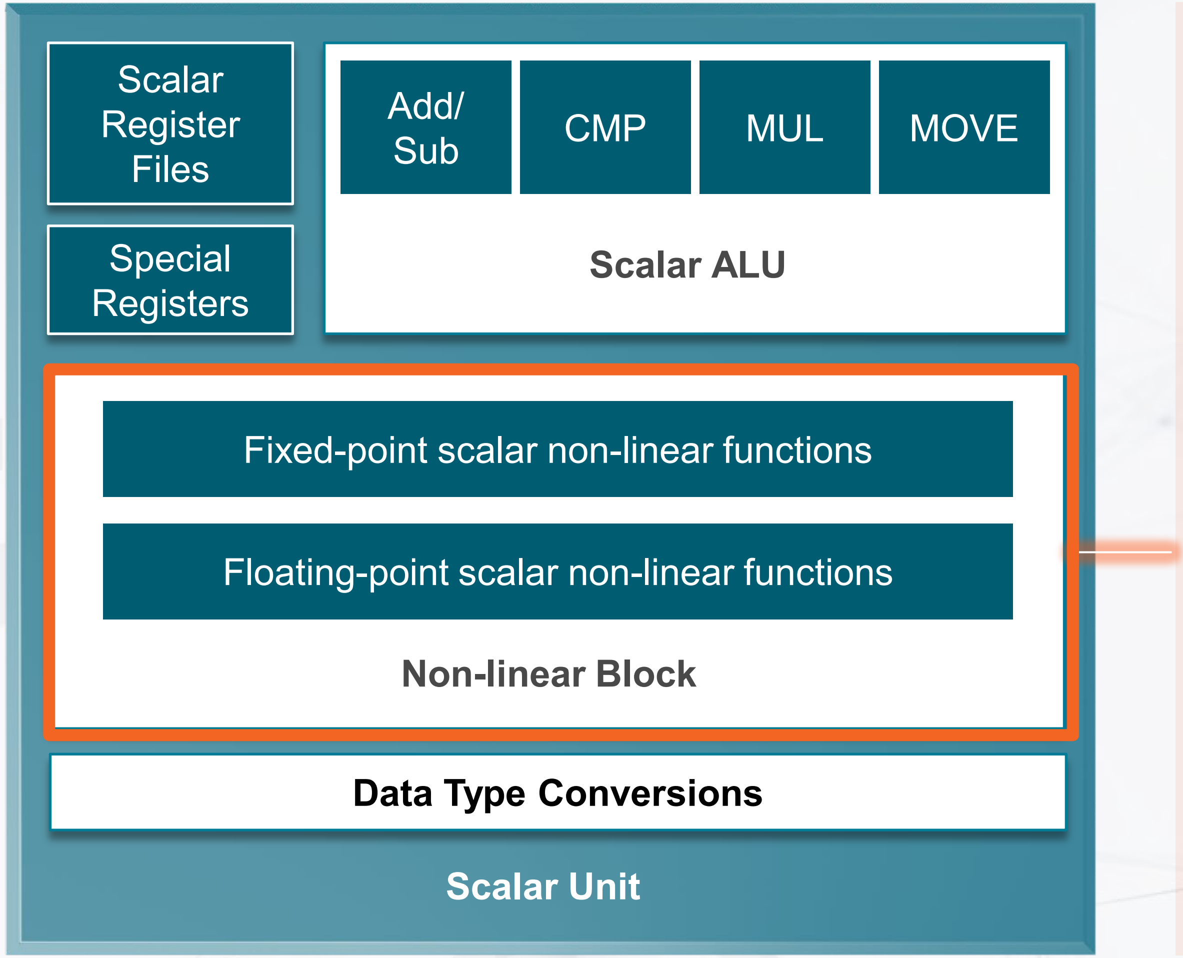
# Scalar Unit – ALU



AIE compiler supports standard C data types

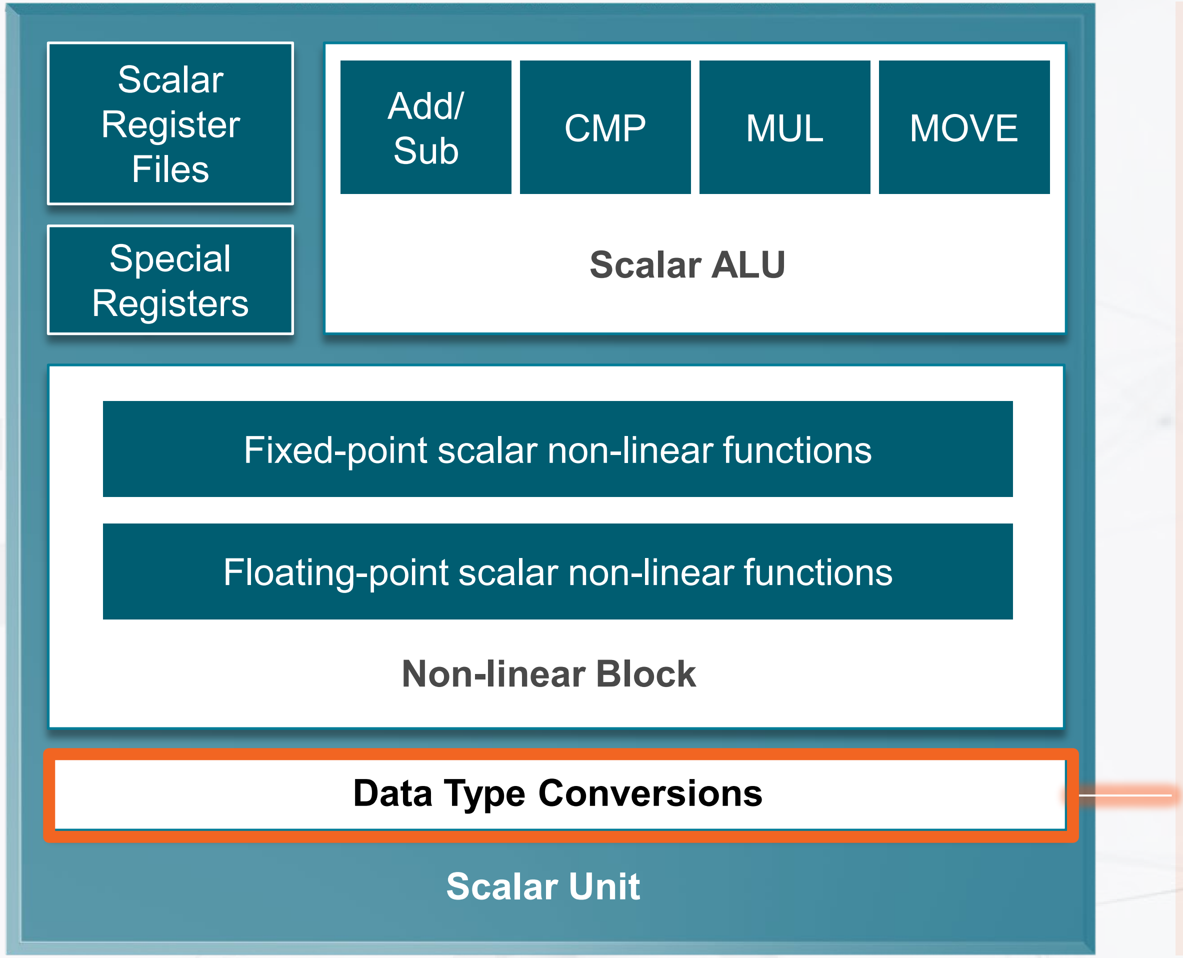
| Operations | Cycles |
|---|---|
| 32-bit add/subtract | 1 |
| 32-bit logical operation | 1 |
| 32x32-bit mult (signed & unsigned) | 3 |
| Shift and compare | 1 |

# Scalar Unit – Non-linear Blocks



- Fixed point/integer non-linear functions
  - sin/cos, abs (real input), clz, min/max
  - sqrt, inv sqrt & inv – supported via data conversion
    - Fix2float and float2fix
- Floating-point, non-linear functions
  - sqrt, inv sqrt, inv, abs, min/max, comparison

# Scalar Unit – Data Type Conversions

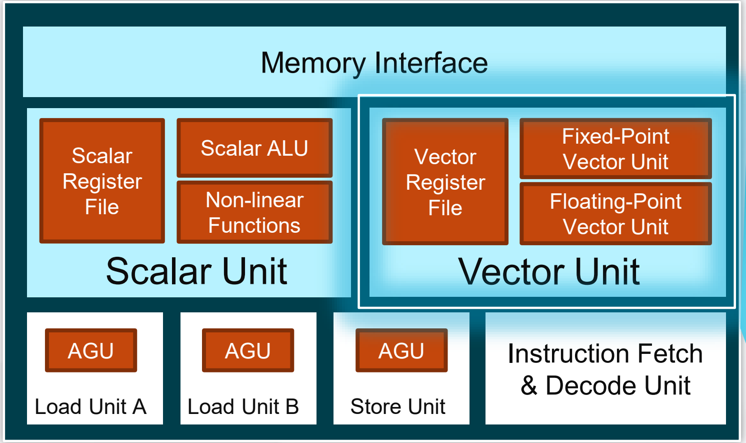


AIE compiler supports standard C data types

- short
- char
- int
- long
- float
- double

AMD
together we advance_

# Vector Unit



1 Fixed-point Vector Unit

2 Floating-point Vector Unit

3 Vector Register File

4 Multi-precision Support

# Fixed-point Vector Unit

**Multiply Accumulator (MAC) path**

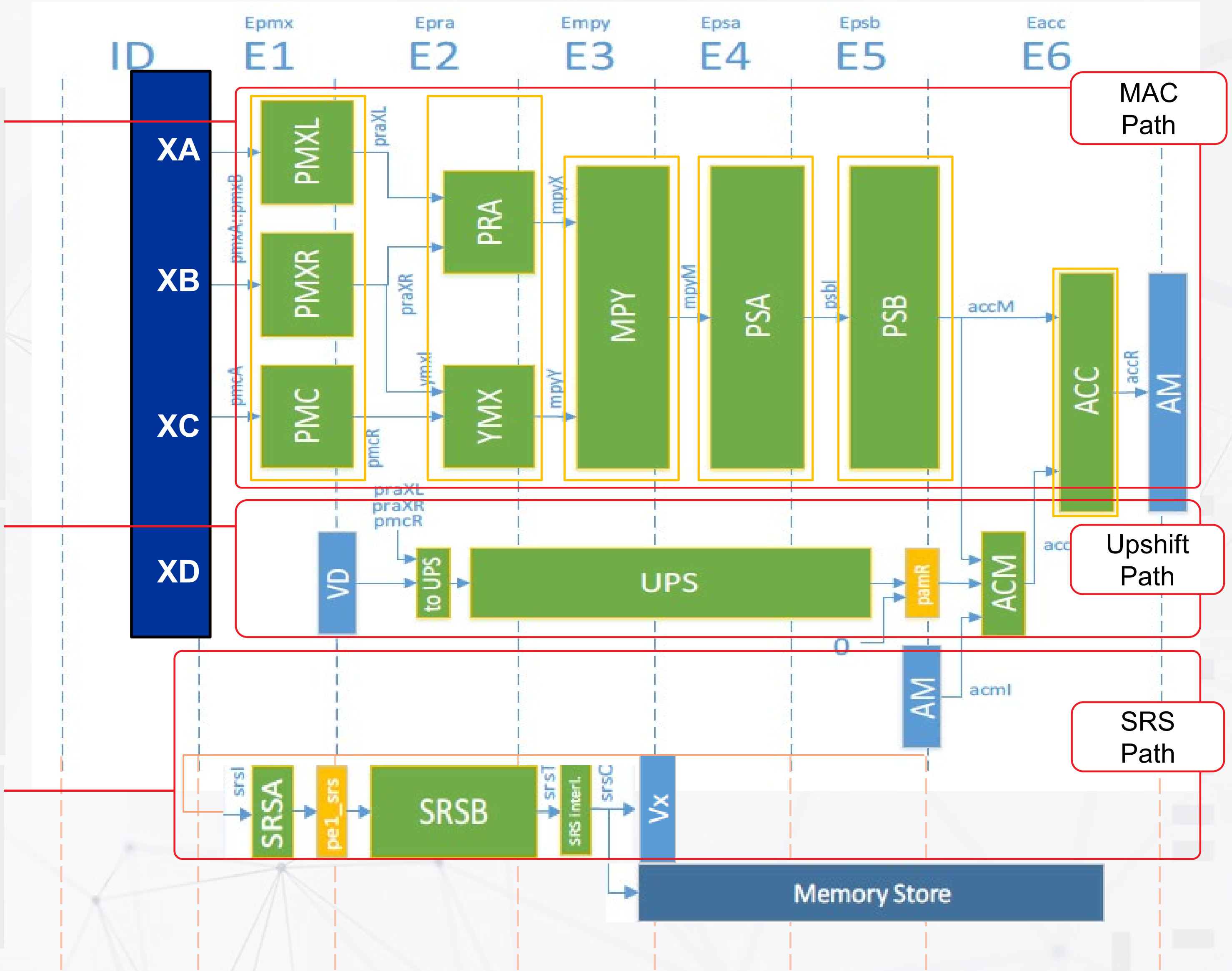
- Reading data from vector registers
- Full permute unit (32-bit granularity)
- Pre-adder
- Multiplier
- Two-step post-adding
- Accumulator

**Upshift path**

- Reading from permute units/vector registers
- Left shifting
- Accumulator

**Shift-Round-Saturate (SRS) path**

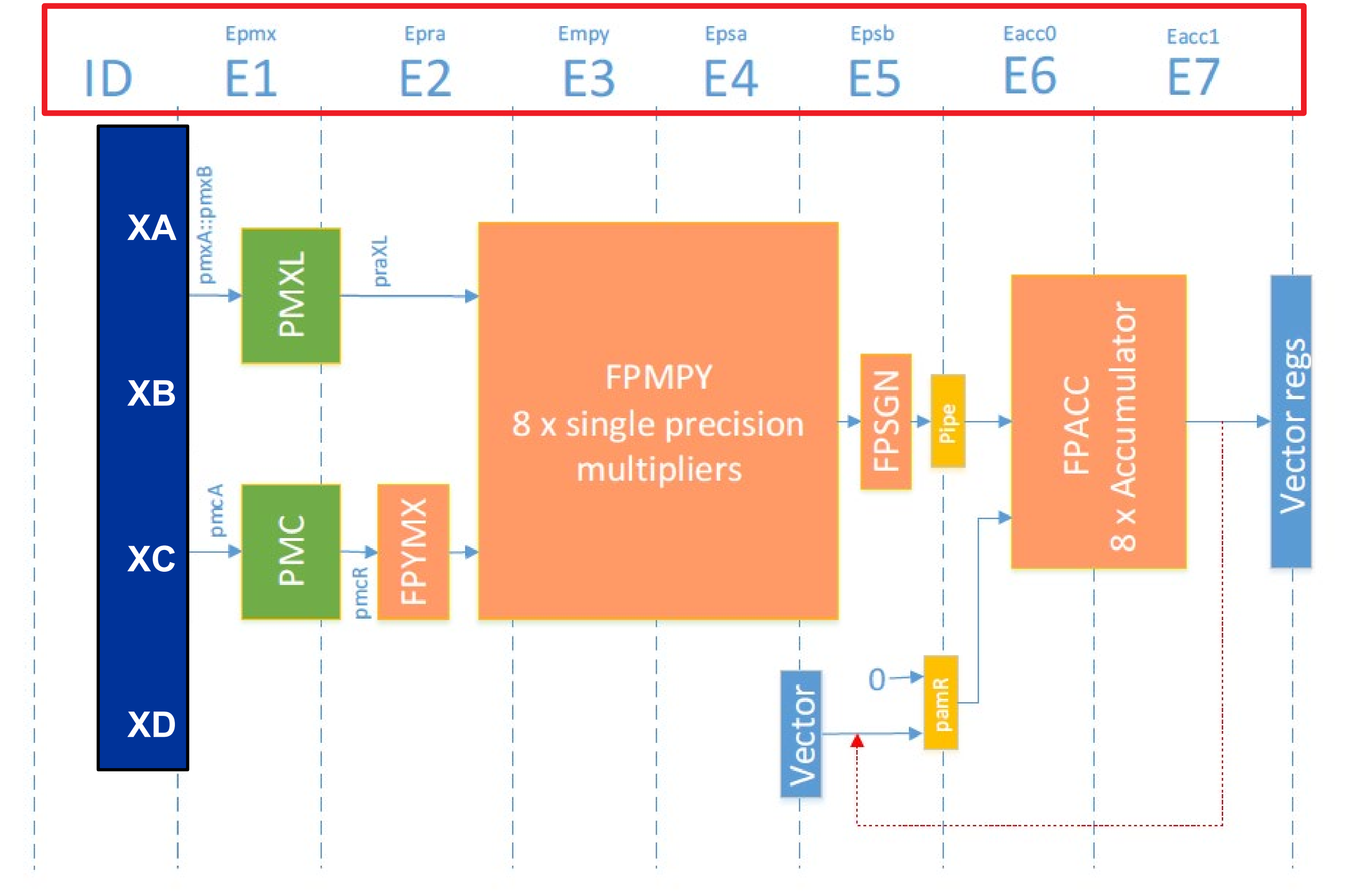
- Reading from accumulator registers
- Storing to registers/memory



AMD
together we advance_

# Floating-point Vector Unit

- Eight multiplications computed in parallel
- Vector elementary functions
  - Comparison, minimum, and maximum
- Fixed-point to floating-point conversion
- Floating-point to fixed-point conversion

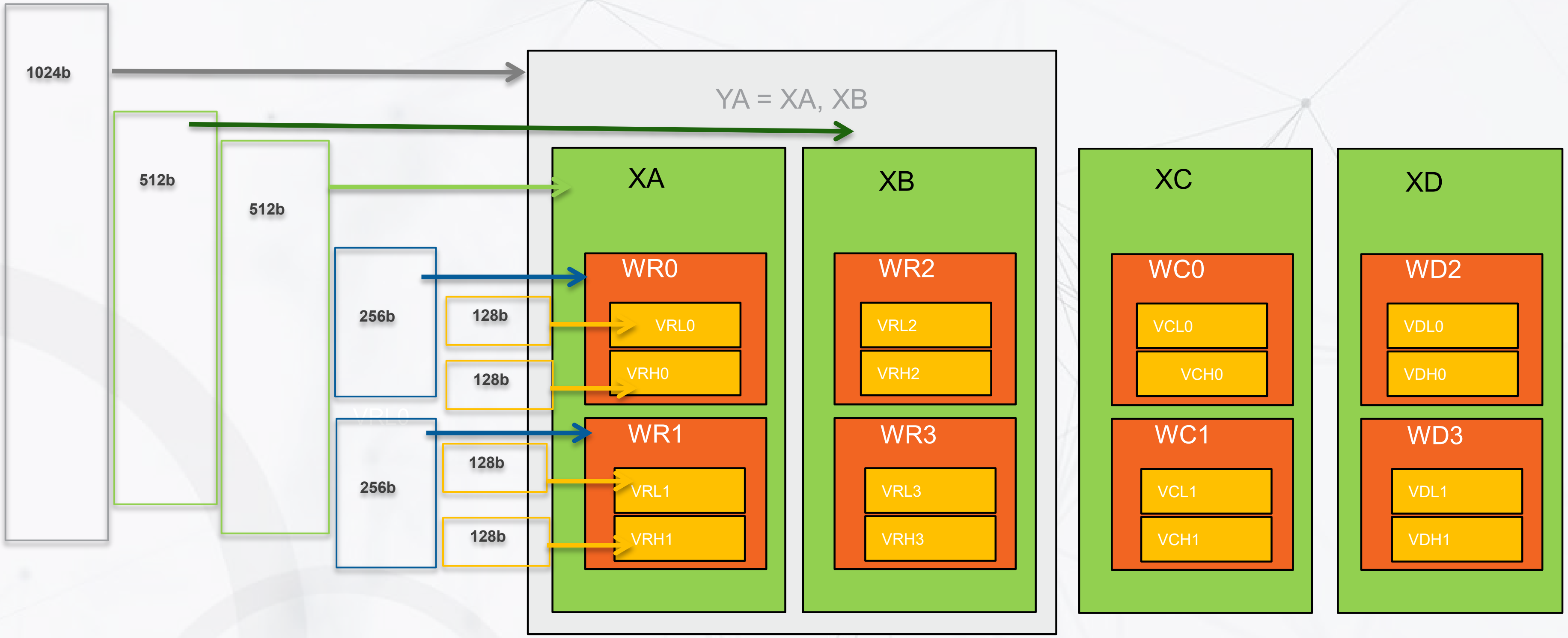


AMD
together we advance_

# Vector Register File

High-width registers enabling SIMD instructions

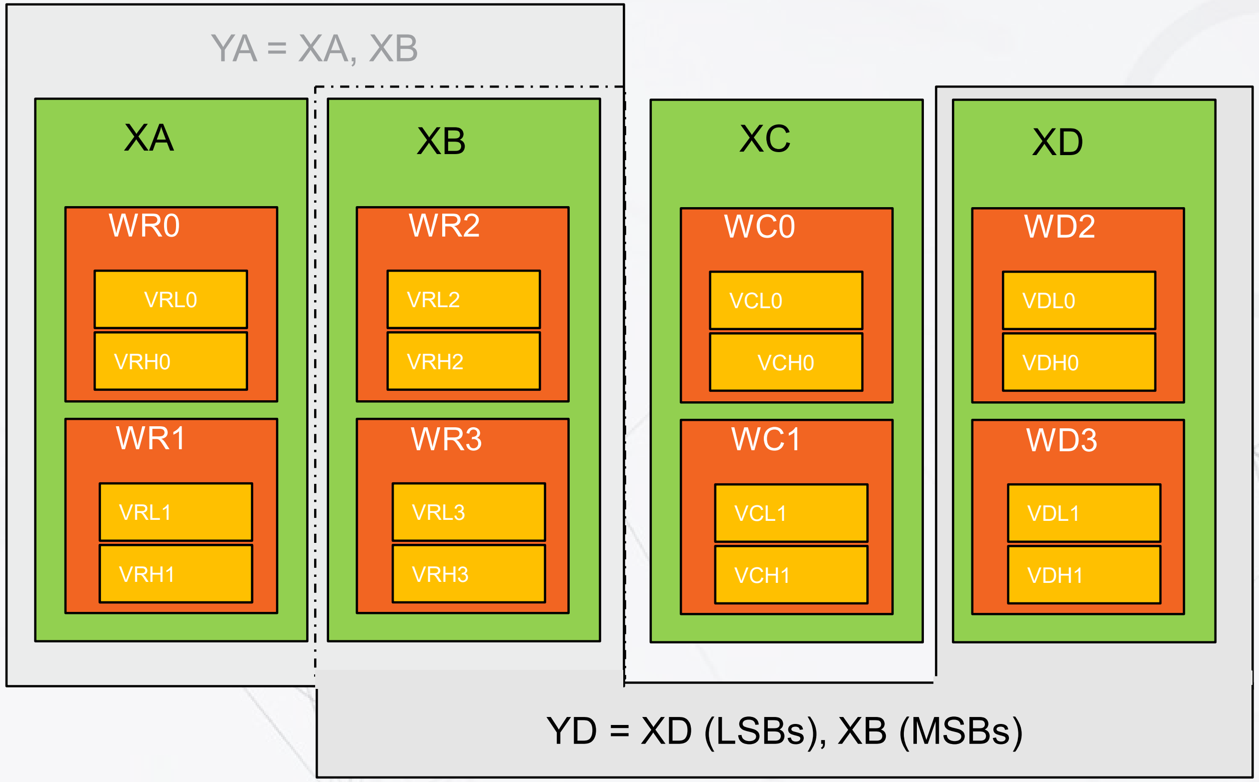
- Incrementally grouped for higher widths



# Vector Register File

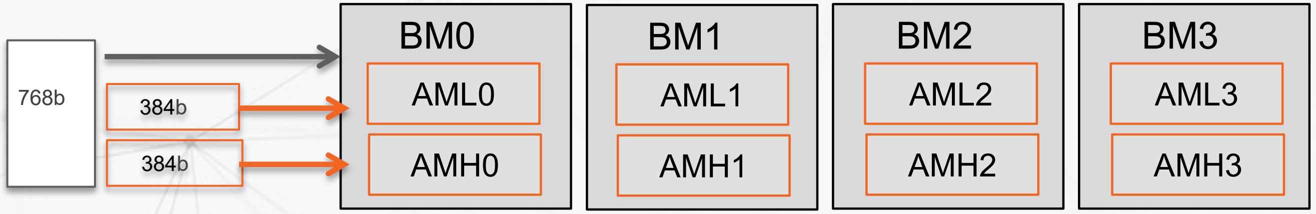
High-width registers enabling SIMD instructions

- Incrementally grouped for higher widths
- Basic hardware registers are 128-bits wide (V registers)



Accumulator registers

- Store the results of the vector data path
- 8 vector lanes of 48 bits
- 32-bit mult. results and acc. without bit overflow



# Multi-precision Support

| X Operand | Z Operand | Output | Number of GMACs @ 1 GHz |
|---|---|---|---|
| 8 real | 8 real | 48 real | 128 |
| 16 real | 8 real | 48 real | 64 |
| 16 real | 16 real | 48 real | 32 |
| 16 real | 16 complex | 48 complex | 16 |
| 16 complex | 16 real | 48 complex | 16 |
| 16 complex | 16 complex | 48 complex | 8 |
| 16 real | 32 real | 48/80 real | 16 |
| 16 real | 32 complex | 48/80 complex | 8 |
| 16 complex | 32 real | 48/80 complex | 8 |
| 16 complex | 32 complex | 48/80 complex | 4 |
| 32 real | 16 real | 48/80 real | 16 |
| 32 real | 16 complex | 48/80 complex | 8 |
| 32 complex | 16 real | 48/80 complex | 8 |
| 32 complex | 16 complex | 48/80 complex | 4 |
| 32 real | 32 real | 80 real | 8 |
| 32 real | 32 complex | 80 complex | 4 |
| 32 complex | 32 real | 80 complex | 4 |
| 32 complex | 32 complex | 80 complex | 2 |
| 32 SPFP | 32 SPFP | 32 SPFP | 8 |

**Example:**

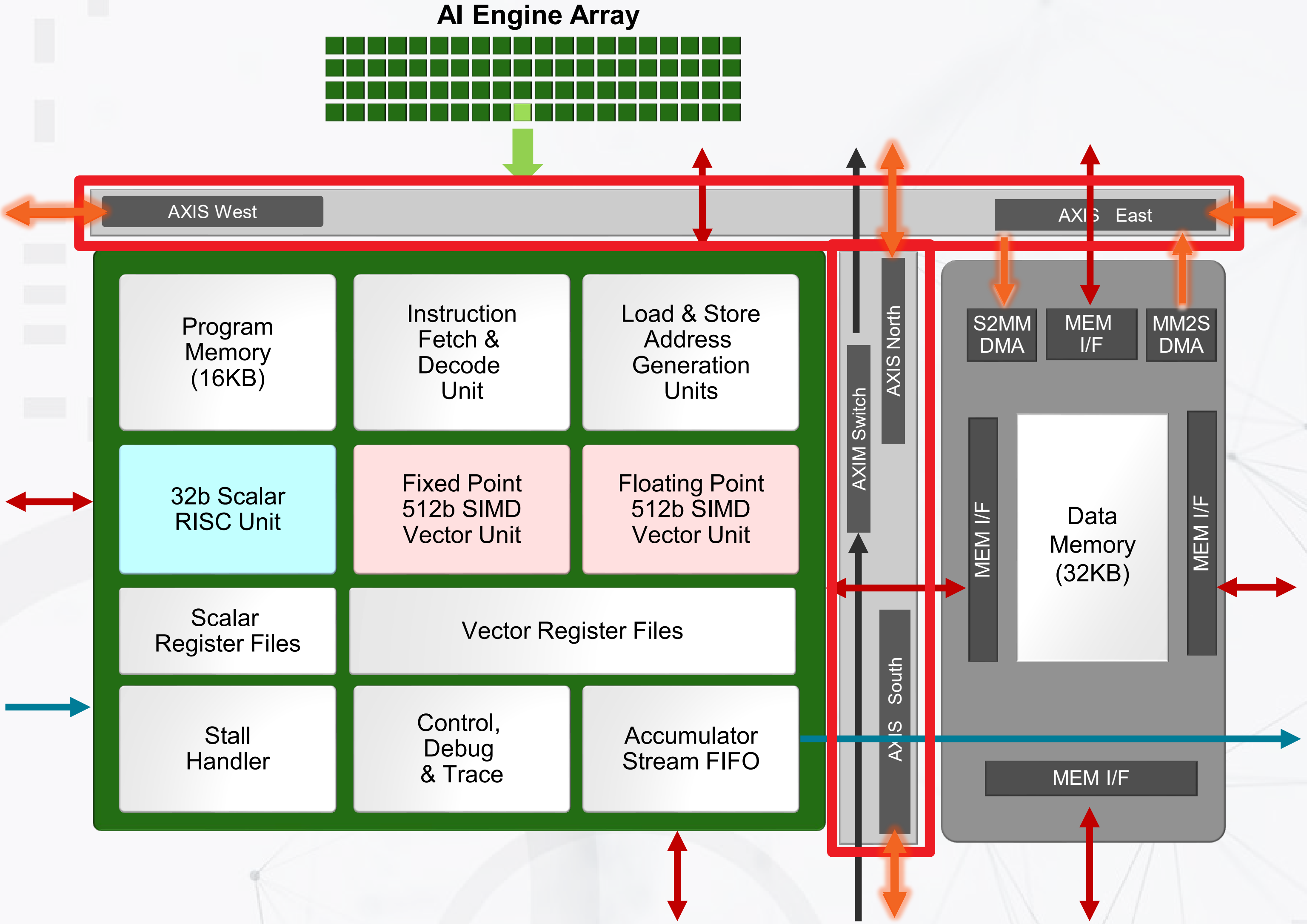
X and Z =>16-bit input vectors

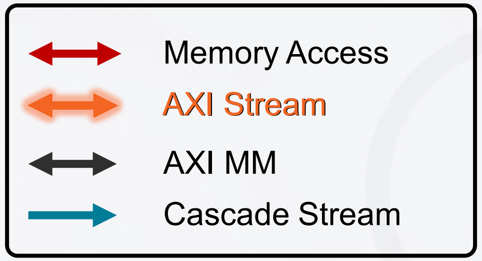
No. of GMACs @ 1 GHz
(32 MACs * 1 GHz clock frequency) =
32 GMAC operations/second.

**cfloat** is a vector type but is not directly supported by the AI Engine vector processor

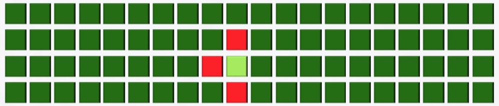
Two instructions have to be issued to perform a mult

# AI Engine: Tile Interconnect





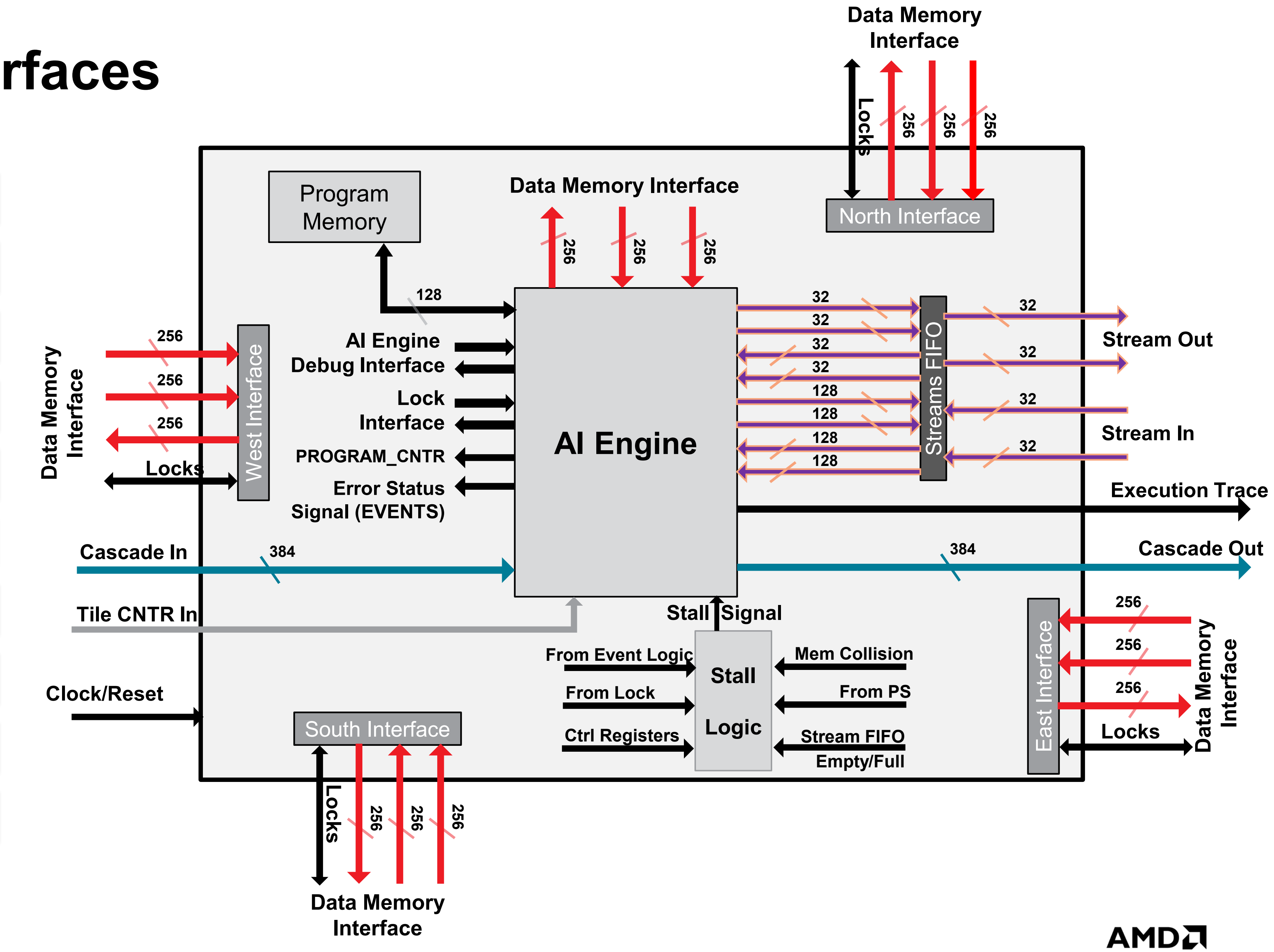
- Communication among AI Engines
- Access to the PL and memory in the PL
- Streaming interconnects
  - AXI-Memory Mapped (AXI-MM) switch
    - Configuration, control, and debug
  - AXI-Stream crossbar switch
    - Routing N/S/E/W around the array



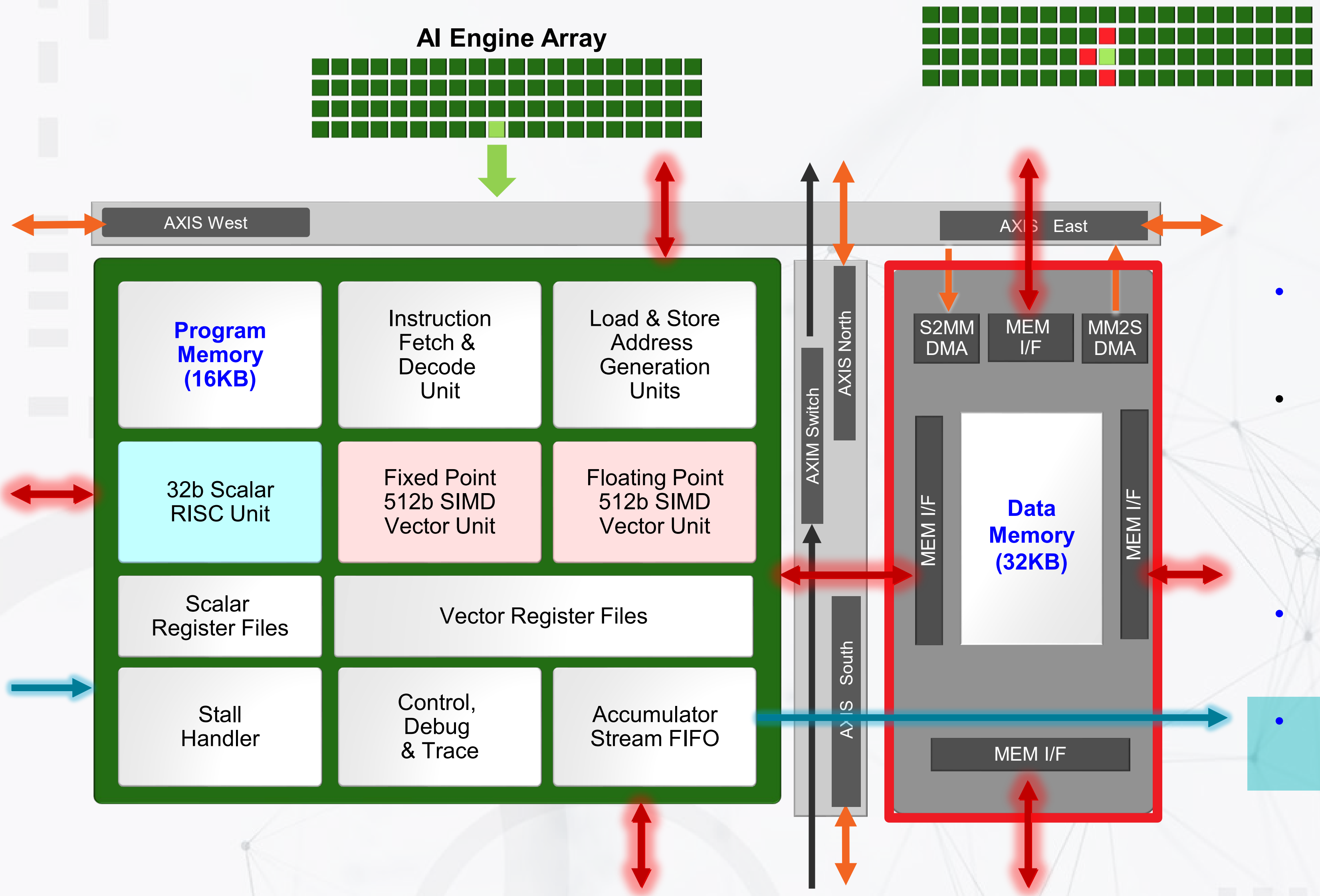
AMD
together we advance_

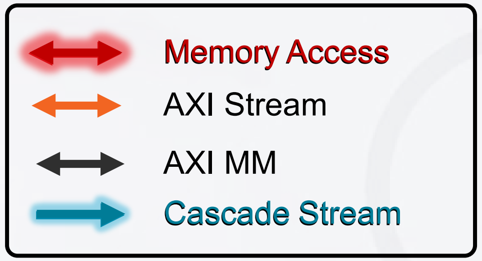
# AI Engine Tile Interfaces

- Program Memory Interface
- Data Memory Interface
- Locks Interface
- Debug Interface
- Cascade Stream Interface
- Direct AXI4-Stream Interface
- Stall Handling Interface
- AI Engine Event Interface
- Tile Timer Interface
- Execution Trace Interface



# AI Engine: Memory





- **Local Data Memory (32KB)**
  - 8 memory banks, memory interface, DMA (incoming/outgoing), and locks
- **Memory Access:**
  - 128KB direct core memory access
    - 32KB local
    - 32KB north, 32KB south
    - 32KB east or west (depending on row)
- **Data Mover**
  - Non-neighbor data communication
- **Cascade Interface**
  - Partial results to next core

AMD
together we advance_

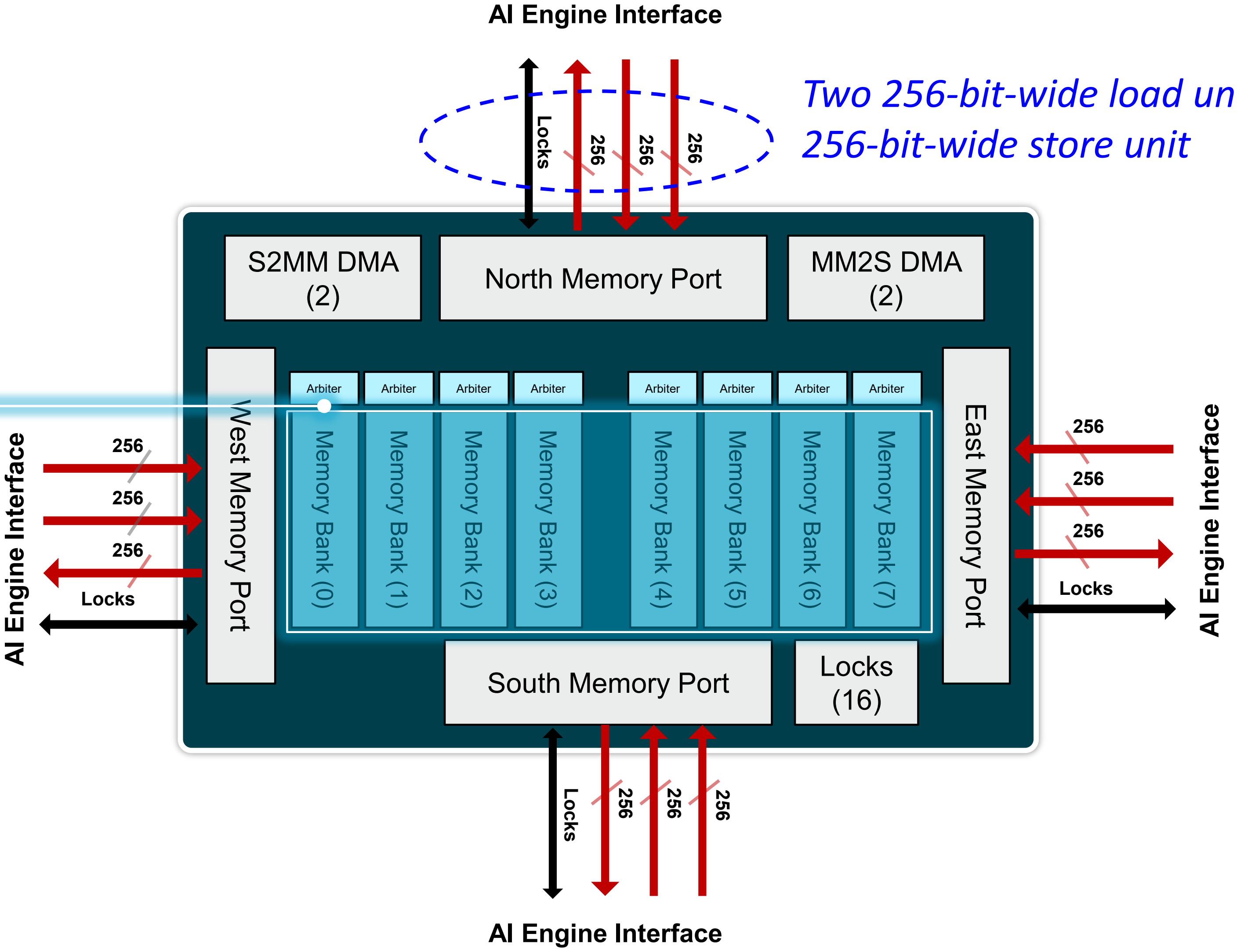
# AI Engine Data Memory: Banks

Eight memory banks:

- 256-word x 128-bit single-port memory
- Write enable for each 32-bit word
- Banks [0-1]: ECC protection
- Banks [2-7]: Parity check

ECC protection:

- 1-bit error detector/corrector
- 2-bit error detector per 32-bit word



*Two 256-bit-wide load units and one 256-bit-wide store unit*

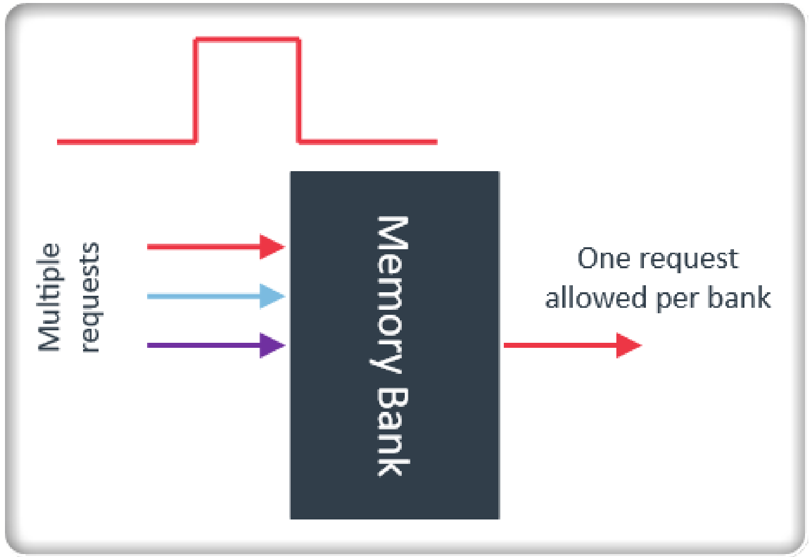
| ←128 bits →<br>Bank 0 | ← 128 bits →<br>Bank 1 |
|---|---|
| 0 | 1 |
| 2 | 3 |
| 4 | 5 |
| 6 | 7 |
| 8 | 9 |
| 10 | 11 |
| .. | .. |
| .. | .. |

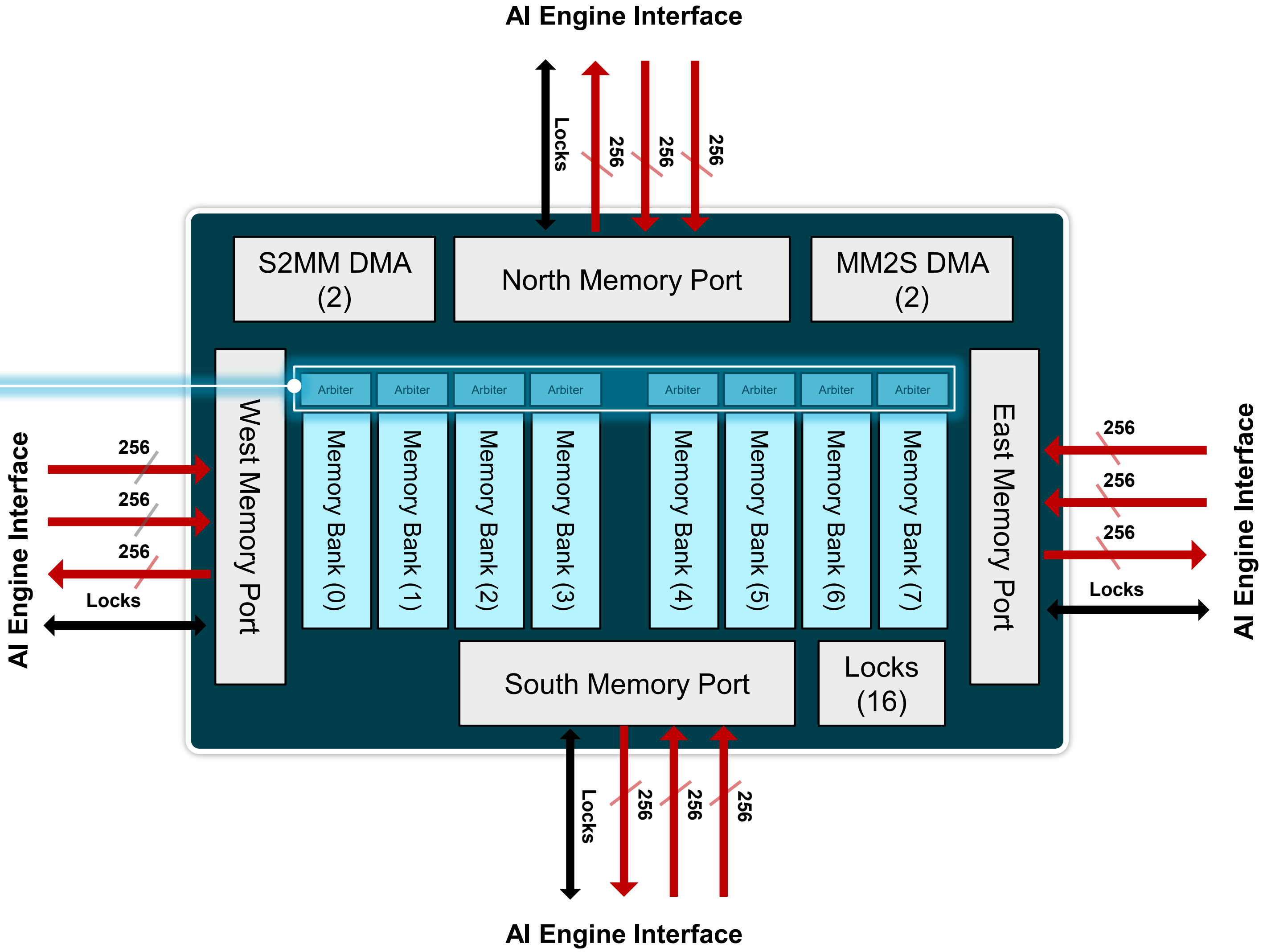
AMD
together we advance_

# AI Engine Memory: Arbitration

### Memory Arbitration

- Round-robin to avoid starving any requester
- Handles a new request every cycle
- Only one requestor at a time allowed to access the memory per bank



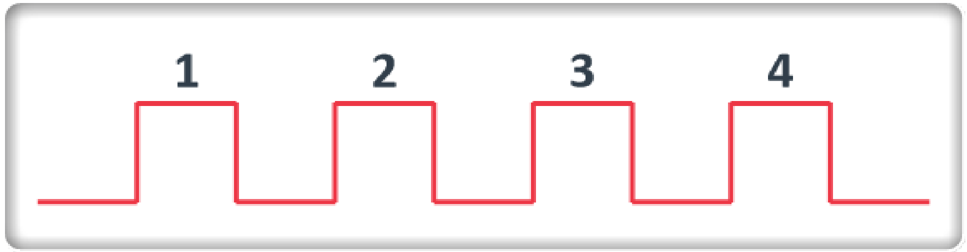


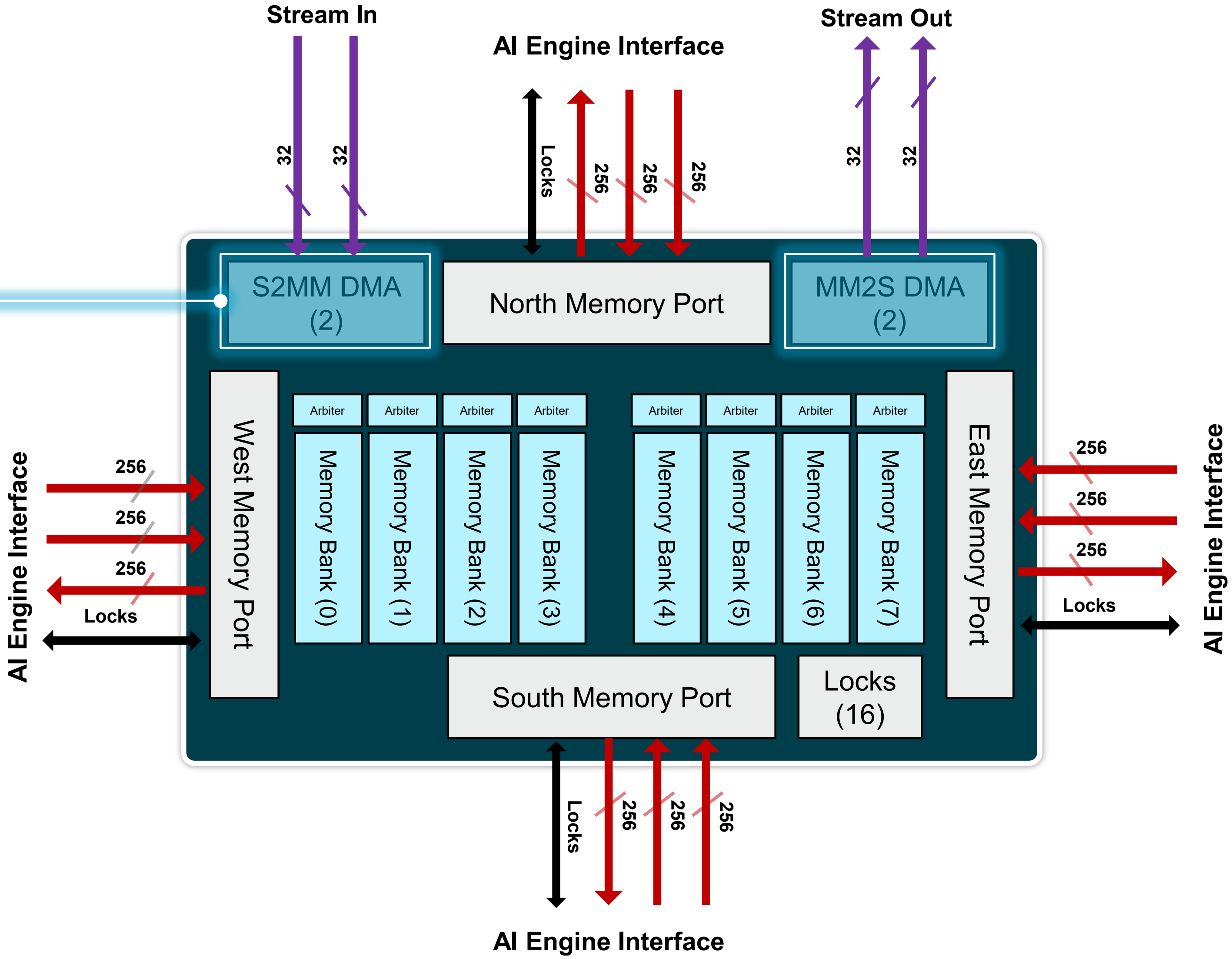
AMD
together we advance_

# AI Engine Memory: Tile DMA Controller

## DMA Controller

- Two incoming and two outgoing streams to the stream switches
- Divided into two modules
  - S2MM → store 32-bit streams of data to memory
  - MM2S → write the contents of the memory to a 32-bit stream
- Waits four clock cycles (128 bit) to access memory
- Has access to the 16 buffer descriptors
- Has access to the 16 locks



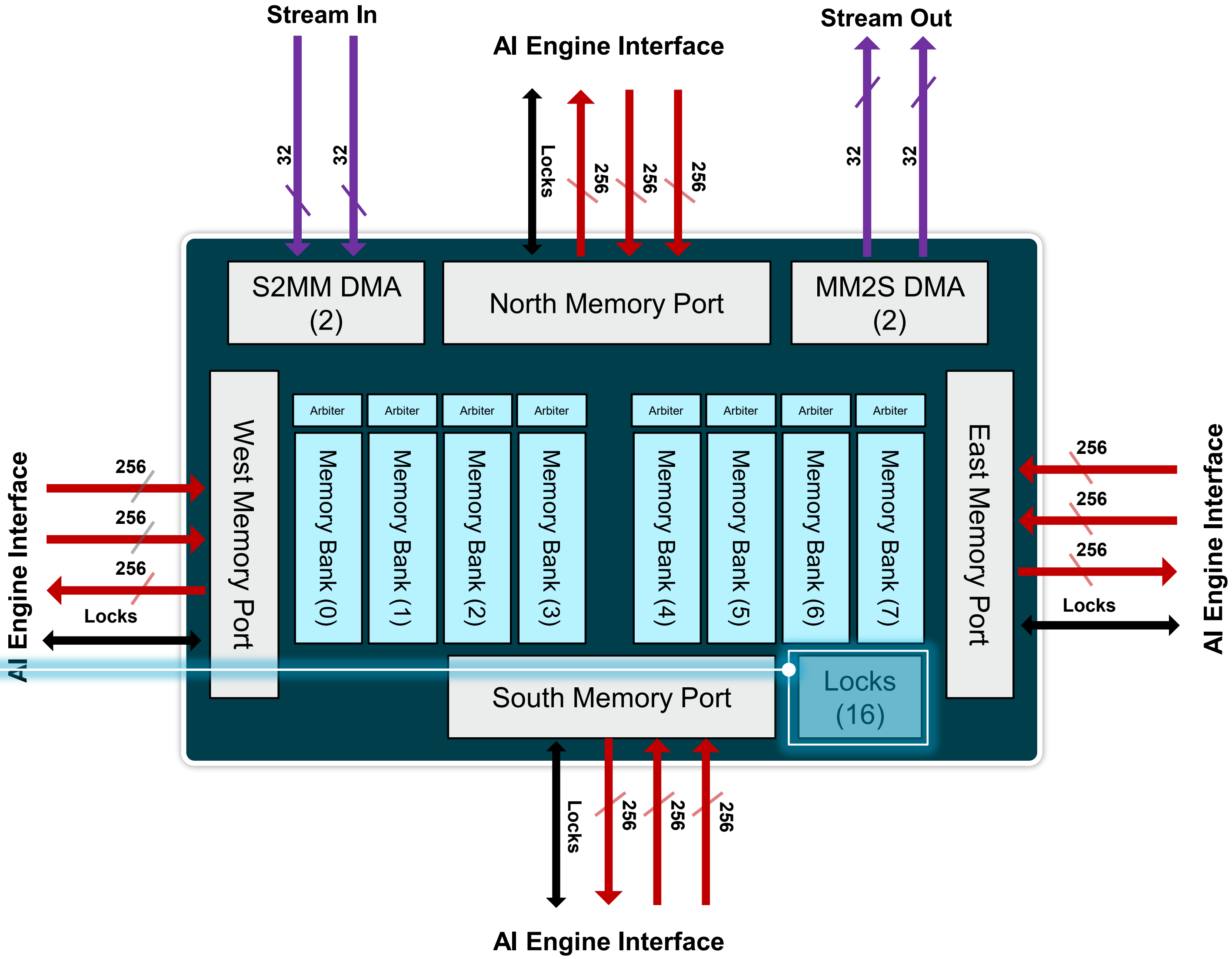


AMD
together we advance_

# AI Engine Memory: Lock Module

### Lock Module

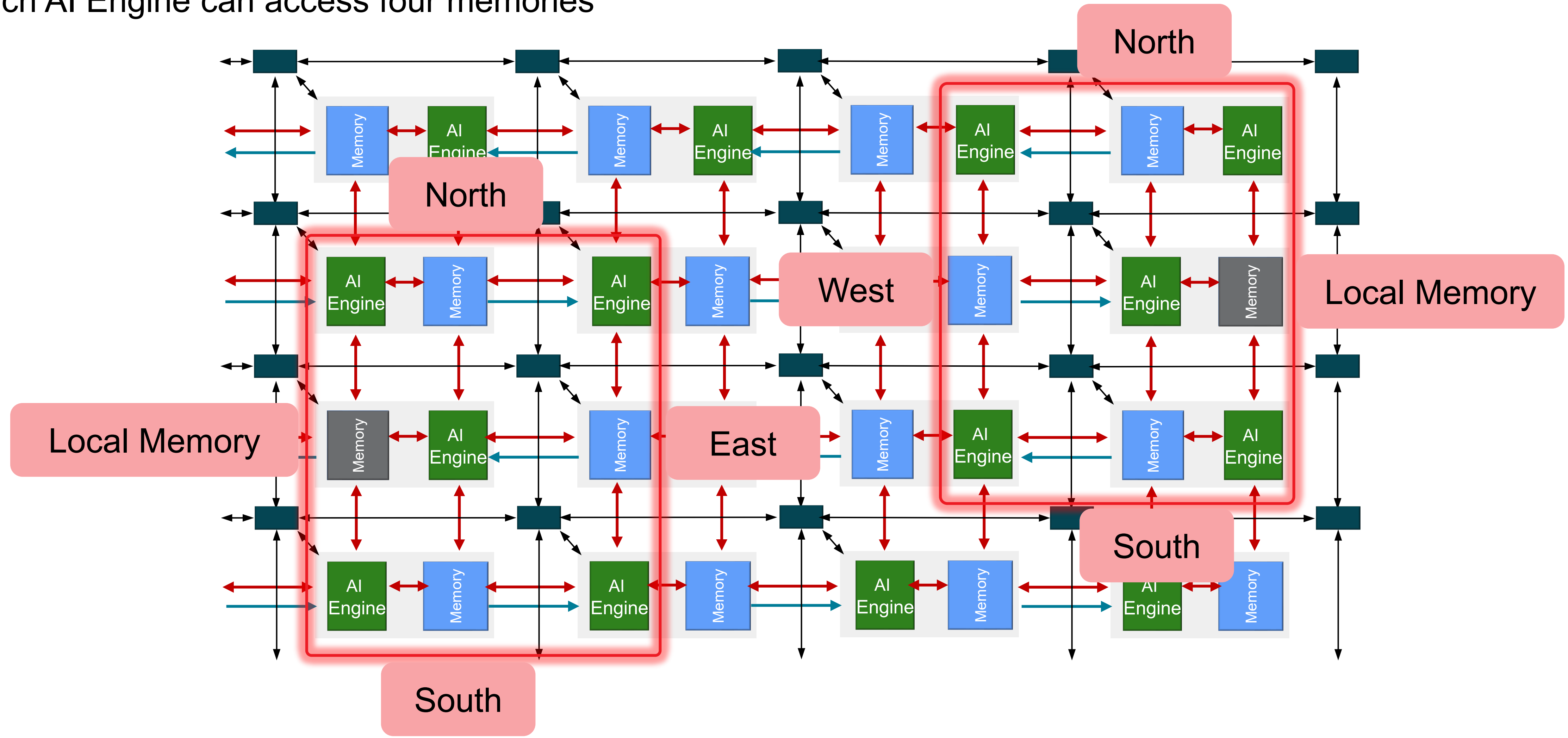
- Helps achieve synchronization among the AI Engines, tile DMA, and external memory-mapped AXI4 interface master
- Sixteen hardware locks with a binary value for each AI Engine memory module lock
- One arbitrator for each lock
- Handles lock requests from the AI Engines, local DMA controller, and memory-mapped AXI4



AMD
together we advance_

# Memory Accesses in the AI Engine Array

Each AI Engine can access four memories



Result: All the diagonals do not have the same memory access structure

AMD together we advance_

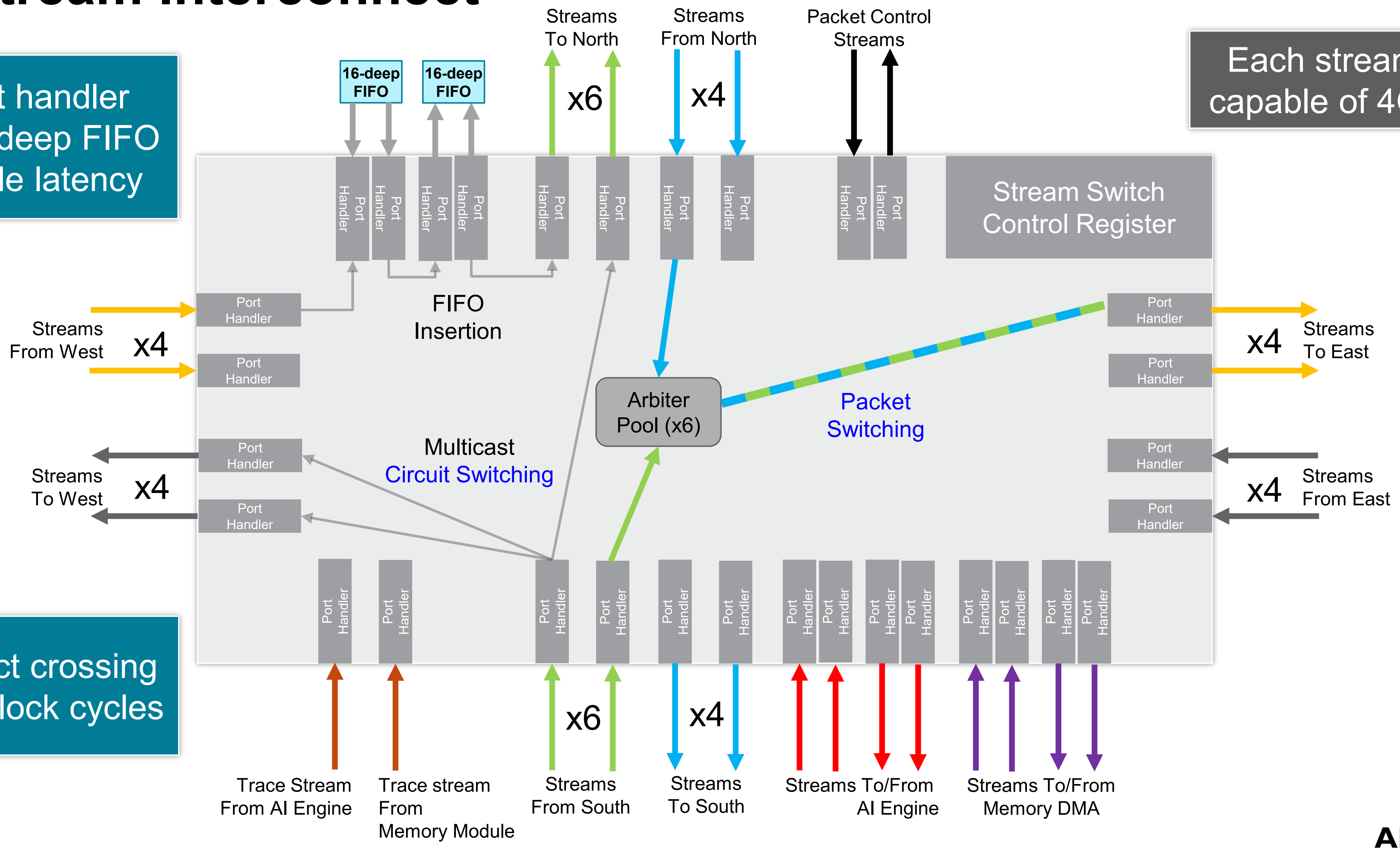
# AXI4-Stream Interconnect

*circuit-switched / packet-switched*

Each port handler contains 4-deep FIFO with 2-cycle latency

Each stream is capable of 4GB/s

Interconnect crossing latency: 4 clock cycles



together we advance_

# AXI-MM Interconnect

*The AI Engine MM AXI4 interconnect allows external access to internal AI Engine tile resources such as memories and registers for configuration and debug.*

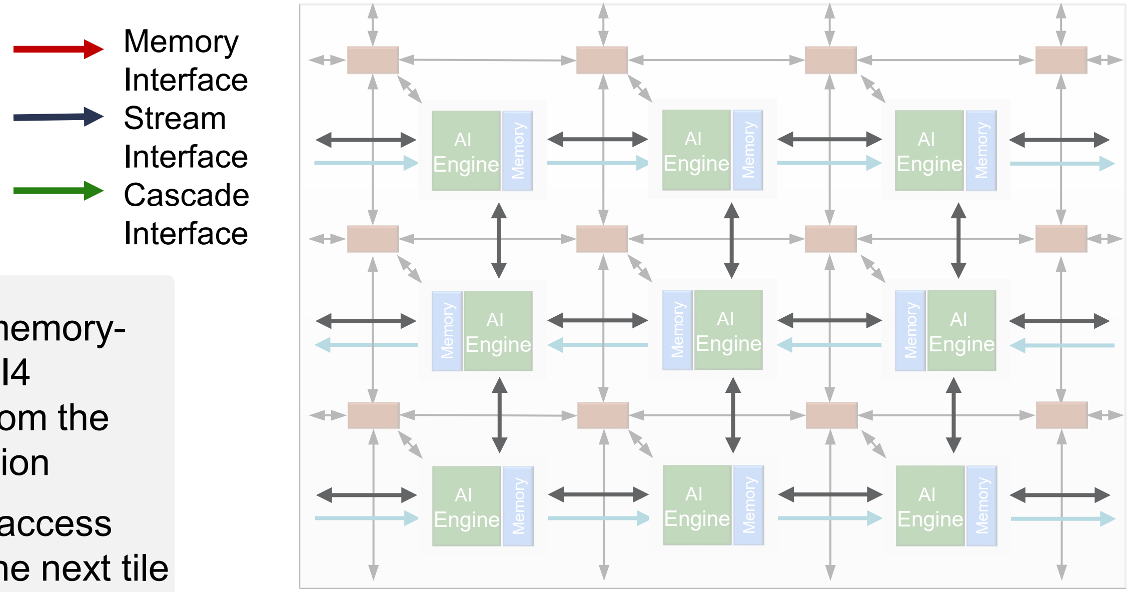
AIE Tile

AXI4-MM

- Write to or read from any of the registers or memories

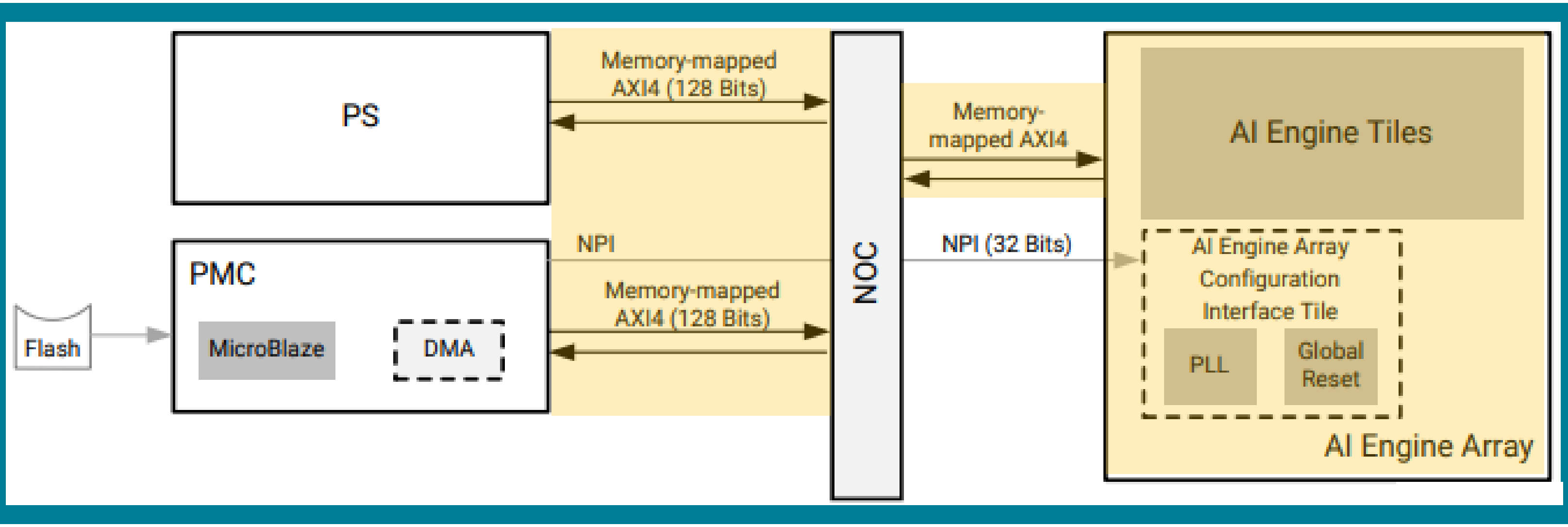
- All internal resources mapped to MM AXI4

- Accept all memory-mapped AXI4 accesses from the south direction
- Otherwise, access passed to the next tile in north direction

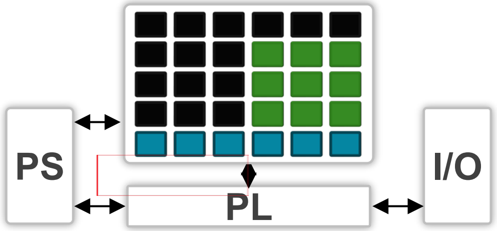


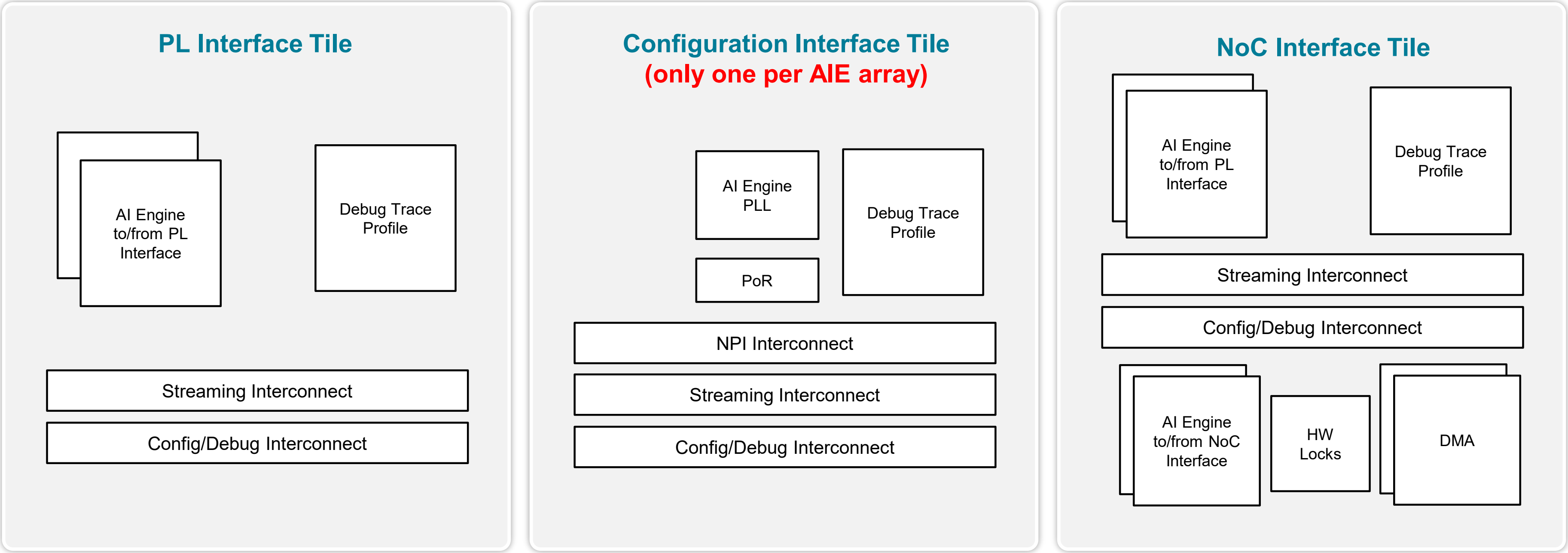
AI Engine Array

Driven from outside by any AXI4 master



# AI Engine Array Interface Tiles *(shim)*





PL Interface Tile

AI Engine to/from PL Interface
Debug Trace Profile
Streaming Interconnect
Config/Debug Interconnect

Configuration Interface Tile (only one per AIE array)

AI Engine PLL
Debug Trace Profile
PoR
NPI Interconnect
Streaming Interconnect
Config/Debug Interconnect

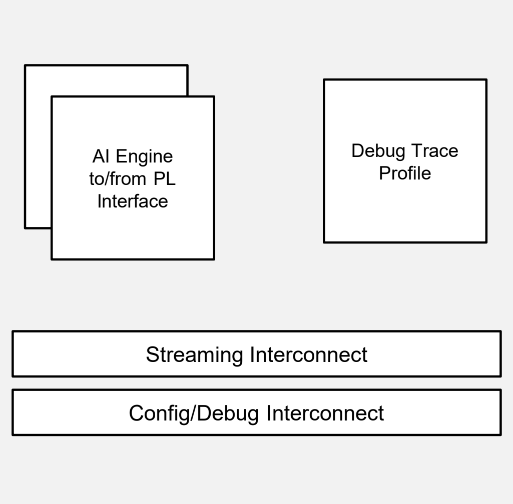
NoC Interface Tile

AI Engine to/from PL Interface
Debug Trace Profile
Streaming Interconnect
Config/Debug Interconnect
AI Engine to/from NoC Interface
HW Locks
DMA

- One-to-one correspondence for every column of the AI Engine array
- Form a row and move memory-mapped AXI4 and AXI4-Stream data horizontally and vertically up
- Based on a modular architecture with device-specific final composition

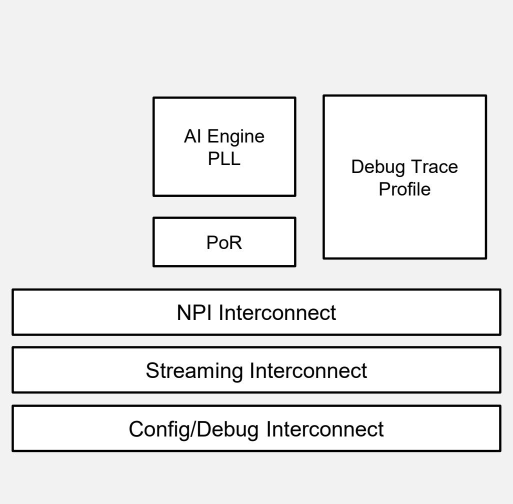
# AI Engine Array Interface Tiles

### PL Interface Tile



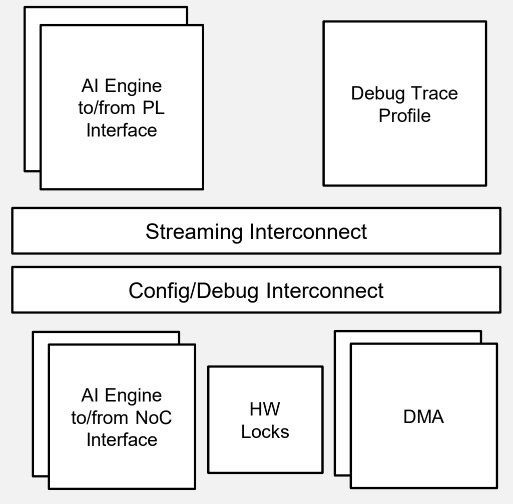
- AXI4-Stream switch
- Memory-mapped AXI4 interface
- AI Engine to PL stream interface
- Control, debug, and trace unit

### Configuration Interface Tile



- PLL for AI Engine clock generation
- Power-on-reset (POR) unit
- Interrupt generation unit
- DFX logic
- NoC peripheral interconnect (NPI) unit
- AI Engine array global registers

### NoC Interface Tile


AI Engine to/from PL Interface
Debug Trace Profile
Streaming Interconnect
Config/Debug Interconnect
AI Engine to/from NoC Interface
HW Locks
DMA

- PL module
- NoC modules with interfaces to the NoC master unit (NMU) and NoC slave unit (NSU)
  - Bi-directional NoC streaming interface
  - Array interface DMA

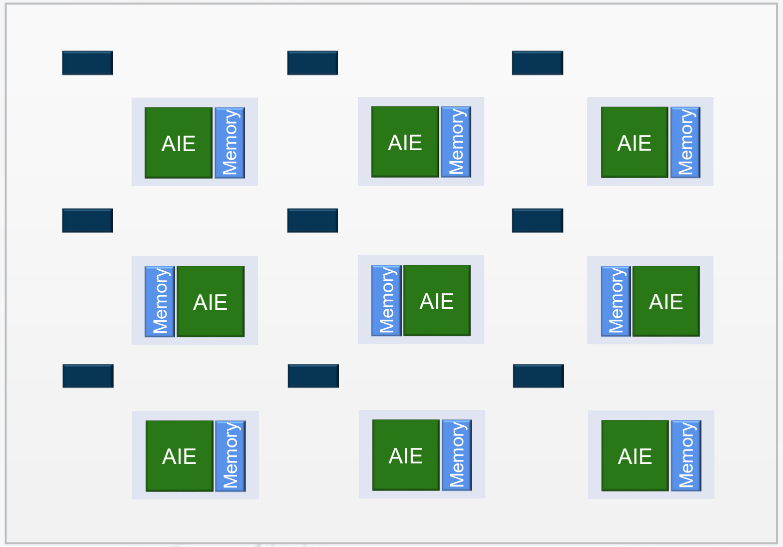
# Key Differences between AIE-ML and AIE

**AIE-ML features:**

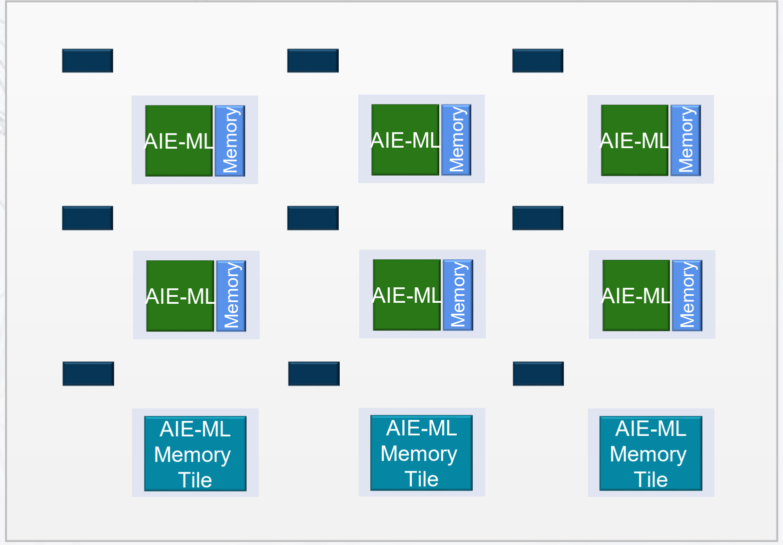
- Increased throughput for ML/AI inference workloads
- Optimized precision
- Increased on-chip memory
- Increased multiplier performance
- Power efficiency

| | AIE | AIE-ML |
|---|---|---|
| Stream interface | Two 32-bit incoming and two 32-bit outgoing | One 32-bit incoming and one 32-bit outgoing |
| int32 | Native support | No native support but can be emulated |
| FP32 | Native support | No native support but can be emulated using bfloat16 |
| BFLOAT16, INT4 | Not supported | Supported |
| Memory tiles | N/A | Available |

AIE 3x3 Array



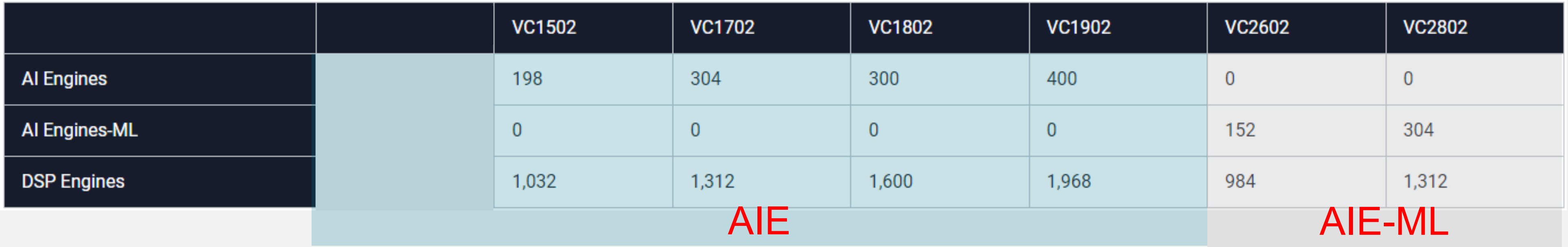
AIE-ML 3x3Array



# Availability of AIE and AIE-ML

Versal™ series devices with the availability of AIE and AIE-ML

**Versal AI Core series:** (based on the devices, AIE or AIE-ML is available)

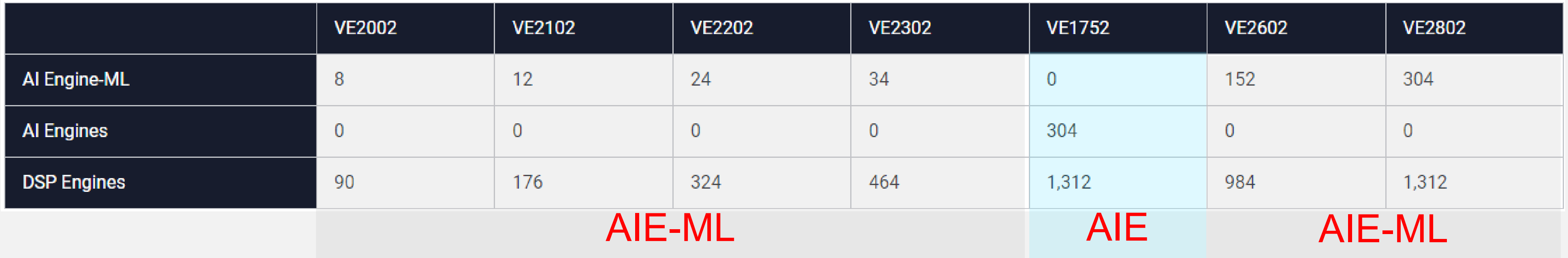


| | | VC1502 | VC1702 | VC1802 | VC1902 | VC2602 | VC2802 |
|---|---|---|---|---|---|---|---|
| AI Engines | | 198 | 304 | 300 | 400 | 0 | 0 |
| AI Engines-ML | | 0 | 0 | 0 | 0 | 152 | 304 |
| DSP Engines | | 1,032 | 1,312 | 1,600 | 1,968 | 984 | 1,312 |

AIE

AIE-ML

**Versal AI Edge series:** (based on the devices, AIE or AIE-ML is available)



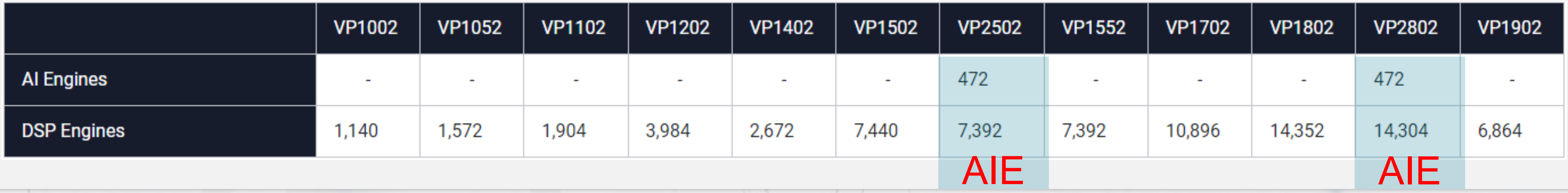
| | VE2002 | VE2102 | VE2202 | VE2302 | VE1752 | VE2602 | VE2802 |
|---|---|---|---|---|---|---|---|
| AI Engine-ML | 8 | 12 | 24 | 34 | 0 | 152 | 304 |
| AI Engines | 0 | 0 | 0 | 0 | 304 | 0 | 0 |
| DSP Engines | 90 | 176 | 324 | 464 | 1,312 | 984 | 1,312 |

AIE-ML

AIE

AIE-ML

**Versal Premium series:** (only AIE is available)



| | VP1002 | VP1052 | VP1102 | VP1202 | VP1402 | VP1502 | VP2502 | VP1552 | VP1702 | VP1802 | VP2802 | VP1902 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AI Engines | - | - | - | - | - | - | 472 | - | - | - | 472 | - |
| DSP Engines | 1,140 | 1,572 | 1,904 | 3,984 | 2,672 | 7,440 | 7,392 | 7,392 | 10,896 | 14,352 | 14,304 | 6,864 |

AIE

AIE

# Summary

01

The hierarchy of the AI Engine architecture consists of:

- AI Engine array
- AI Engine tile
- AI Engine, memory module, and interconnects
- Scalar and vector units

02

The AI Engine tile has the following interfaces:

- Memory, lock, debug, stream, stall, event, timer, execution trace

03

The AI Engine memory module contains:

- Eight memory banks
- Two input streams to memory-map DMA (S2MM)
- Two memory-map to output DMA streams (MM2S)
- Hardware synchronization module (locks)

# GENERAL DISCLOSURE AND ATTRIBUTION STATEMENT

The information contained herein is for informational purposes only and is subject to change without notice. While every precaution has been taken in the preparation of this material, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this material, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD’s products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18.

AMD