Versal Adaptive SoC AIE-ML v2

Architecture Manual

AM027 (v1.1) October 23, 2025

AMD

Table of Contents

AMD

Chapter 1

Overview

Introduction to Versal Adaptive SoCs

AMD Versal™ adaptive SoCs combine programmable logic (PL), processing system (PS), and AI Engines with leading-edge memory and interfacing technologies to deliver powerful heterogeneous acceleration for any application. The hardware and software are targeted for programming and optimization by data scientists and software and hardware developers. A host of tools, software, libraries, IP, middleware, and frameworks enable Versal adaptive SoCs to support all industry-standard design flows.

The Versal portfolio is the first platform to combine software programmability and domain-specific hardware acceleration with the adaptability necessary to meet today's rapid pace of innovation. The portfolio is uniquely architected to deliver scalability and AI inference capabilities for a host of applications across different markets—from cloud—to networking—to wireless communications—to edge computing and endpoints.

The Versal architecture has a wealth of connectivity and communication capability and a programmable network on chip (NoC) to enable seamless memory-mapped access to the full height and width of the device. AI Engines are SIMD VLIW vector processors for adaptive inference and advanced signal processing compute. The PL combines configurable logic blocks, memory, and DSP Engines architected for high-compute density. The PS includes application and real-time processors from Arm® for intensive compute tasks.

The Versal AI Edge Series Gen 2 delivers end-to-end acceleration for AI-driven embedded systems—all in a single device built on a foundation of enhanced safety and security. Combining world-class programmable logic with a new high-performance processing system of integrated Arm CPUs and next-generation AI Engines, these devices enable all three phases of compute in embedded AI applications: preprocessing, AI inference, and postprocessing.

The Versal AI Edge Series focuses on AI performance per watt for real-time systems in automated drive, predictive factory and healthcare systems, multi-mission payloads in aerospace & defense, and a breadth of other applications. More than just AI, the Versal AI Edge Series accelerates the whole application from sensor to AI to real-time control, all with the highest levels of safety and security to meet critical standards such as ISO 26262 and IEC 61508.

The Versal AI Core Series delivers breakthrough AI inference acceleration and compute performance. This series is designed for a breadth of applications, including cloud for dynamic workloads and network for massive bandwidth, all while delivering advanced safety and security features. AI and data scientists, as well as software and hardware developers, can all take advantage of the high-compute density to accelerate the performance of any application.

The Versal Prime Series Gen 2 combines world-class programmable logic from AMD with a new high-performance processing system of integrated Arm CPUs—offering significantly greater scalar compute than existing Versal or Zynq™ adaptive SoCs. This powerful combination of flexible, real-time sensor processing and the ability to handle complex embedded computing workloads allows designers to maximize system performance while avoiding the overhead of a multi-chip solution.

The Versal Prime Series is the foundation and the mid-range of the Versal portfolio, serving the broadest range of uses across multiple markets. These applications include 100G to 200G networking equipment, network and storage acceleration in the data center, communications test equipment, broadcast, and aerospace & defense. The series integrates mainstream 58G transceivers and optimized I/O and DDR connectivity, achieving low-latency acceleration and performance across diverse workloads.

The Versal Premium Series Gen 2 offers new levels of memory and data bandwidth with CXL® 3.1, PCIe® Gen6, and DDR5/LPDDR5X interfacing capabilities, tailored to fit the application requirements of tomorrow's data center, communications, test & measurement, and aerospace & defense data-intensive applications. As a heterogeneous compute platform, the series is engineered to help users reach high levels of acceleration for a wide range of compute-intensive workloads by providing high compute density, custom memory hierarchy, and DSP Engine resources.

The Versal Premium Series provides breakthrough heterogeneous integration, very high-performance compute, connectivity, and security in an adaptable platform with a minimized power and area footprint. The series is designed to exceed the demands of high-bandwidth, compute-intensive applications in communications, data center, test & measurement, and other applications. The Versal Premium Series includes 112G PAM4 transceivers and integrated blocks for 600G Ethernet, 600G Interlaken, PCI Express® Gen5, and high-speed cryptography.

The Versal RF Series incorporates high-frequency, high sample-rate RF-sampling data converters with integrated signal processing IP for high-performance RF systems. These IP tiles provide a low power consumption and latency data interface between the DACs/ADCs and programmable logic. RF-DACs and RF-ADCs also include programmable filters and digital up and down converters (DUCs/DDCs).

The Versal HBM Series enables the convergence of fast memory, adaptable compute, and secure connectivity in a single platform. The series is architected to keep up with the higher memory needs of the most compute intensive, memory-bound applications, providing adaptable acceleration for data center, communications, test & measurement, and aerospace & defense applications. Versal HBM adaptive SoCs integrate the most advanced HBM2e DRAM, providing high memory bandwidth and capacity within a single device.

The Versal architecture documentation suite is available at: https://www.amd.com/versal.

Navigating Content by Design Process

AMD Adaptive Computing documentation is organized around a set of standard design processes to help you find relevant content for your current development task. You can access the AMD Versal™ adaptive SoC design processes on the Design Hubs page. You can also use the Design Flow Assistant to better understand the design flows and find content that is specific to your intended design needs. This document covers the following design processes:

- **System and Solution Planning:** Identifying the components, performance, I/O, and data transfer requirements at a system level. Includes application mapping for the solution to PS, PL, and AI Engine. Topics in this document that apply to this design process include:
  - Chapter 1: Overview provides an overview of the AIE-ML v2 architecture and includes:
    - AIE-ML v2 Array Overview
    - AIE-ML v2 Array Hierarchy
    - Performance
  - Chapter 2: AIE-ML v2 Tile Architecture describes the interaction between the memory module and the interconnect and between the AIE-ML v2 and the memory module.
  - Chapter 3: AIE-ML v2 Array Interface Architecture is a high-level view of the AIE-ML v2 array interface to the PL and NoC.
  - Chapter 4: AIE-ML v2 Architecture describes the processor functional unit and register files.
  - Chapter 5: AIE-ML v2 Memory Tile Overview and Features describes the features and functionality of the AIE-ML v2 memory tile, which is an additional functional unit in the AIE-ML v2 architecture.
  - Chapter 6: AIE-ML v2 Configuration and Boot describes configuring the AIE-ML v2 array from the processing system during boot and reconfiguration.
- **AI Engine Development:** Creating the AI Engine graph and kernels, library use, simulation debugging and profiling, and algorithm development. Also includes the integration of the PL and AI Engine kernels. Topics in this document that apply to this design process include:

- Chapter 2: AIE-ML v2 Tile Architecture
- Chapter 4: AIE-ML v2 Architecture
- Chapter 5: AIE-ML v2 Memory Tile Overview and Features

Introduction to AI Engines and AIE-ML v2

Next-generation wireless, data center, automotive, and industrial applications demand significant increases in compute acceleration without increasing board area and while remaining power efficient. With Moore's Law and Dennard Scaling no longer following their traditional trajectories, simply porting existing hardware architectures to next-generation process nodes is not sufficient to meet the system-level performance requirements of these applications – architectural innovation is critical to overcome the diminishing gains from process scaling. To address this need, AMD has developed an innovative processing technology, the AI Engine, as part of the Versal architecture.

AI Engines are very long instruction word (VLIW), single instruction multiple data (SIMD) vector processors optimized for both machine learning and advanced signal processing workloads. AI Engines are arranged in 2D arrays within a given device, with the number of individual processors varying by device, allowing for scalability to meet the compute needs of a breadth of applications. AMD offers two primary types of AI Engines within the Versal portfolio: AI Engines (AIE), which are balanced between signal processing and machine learning workloads, and AI Engines-Machine Learning (AIE-ML), which are optimized for machine learning workloads. Although both types of AI Engines can be used for either machine learning or signal processing functions, differences in native data type support, memory, and other capabilities impact which workloads each AI Engine type is best suited for.

AI Engines-Machine Learning enable high performance per watt with low latency for inference workloads. AIE-ML v2 is the second generation of the AI Engines-Machine Learning architecture. AIE-ML v2 offers several architectural enhancements over the first-generation AIE-ML architecture, including up to two times compute/tile, new native data type support, and improved energy efficiency.

This architecture manual details the specifics of the AIE-ML v2 architecture.

AIE-ML v2 Array Features

Versal AI Edge Series Gen 2 adaptive SoCs feature an array of AIE-ML v2 tiles, AIE-ML v2 memory tiles, and the AIE-ML v2 array interface. The following lists the features of each. A pictorial view of the array organization is shown in AIE-ML v2 Array Overview.

AIE-ML v2 Tile Features

- A separate building block, integrated into the silicon, outside the programmable logic (PL).
- One AIE-ML v2 incorporates a high-performance very-long instruction word (VLIW) single-instruction multiple-data (SIMD) vector processor optimized for many applications including machine learning applications.
- From a hardware perspective, data memory is 64 KB organized as eight banks of 8 KB (256-bit wide and 256 words deep). From a programmer's perspective, every two banks are interleaved to form one bank, that is, a total of four banks of 512-bit wide.
- Streaming interconnect for deterministic throughput, high-speed data flow between AIE-ML v2 tiles and/or the programmable logic in the Versal device.
- Direct memory access (DMA) in the AIE-ML v2 tile moves data from incoming stream(s) to local memory and from local memory to outgoing stream(s). It has two S2MM channels and two MM2S channels and supports 4-D address generation.
- Configuration interconnect (through memory-mapped AXI4 interface) with a shared, transaction-based switched interconnect for access from external masters to internal AIE-ML v2 tile.
- Hardware synchronization primitives provide synchronization of the AIE-ML v2, between the AIE-ML v2 and the tile DMA, and between the AIE-ML v2 and an external manager (through the memory-mapped AXI4 interface).
- Debug, trace, and profile functionality.

AIE-ML v2 Memory Tile Features

- A tile containing 512 KB of high-density, high-bandwidth memory to reduce the use of PL resources. Memory is arranged in eight physical banks of 256-bit wide and 2K words deep.
- The memory tile DMA has six MM2S and S2MM channels with 64-bit stream interfaces and 256-bit memory interfaces. The DMA also supports compression and decompression of zeroes in data before sending onto the stream. It also supports 4-D tensor address generation.
- The AXI4-Stream interconnect has north ports, south ports, and the DMA MM2S and S2MM channels that are 64-bit wide. It also has 32-bit trace, control response, and control that are connected using adapters.
- Memory-mapped AXI4 configuration is the same as the AIE-ML v2 tile.

AIE-ML v2 Array Interface to NoC and PL Resources

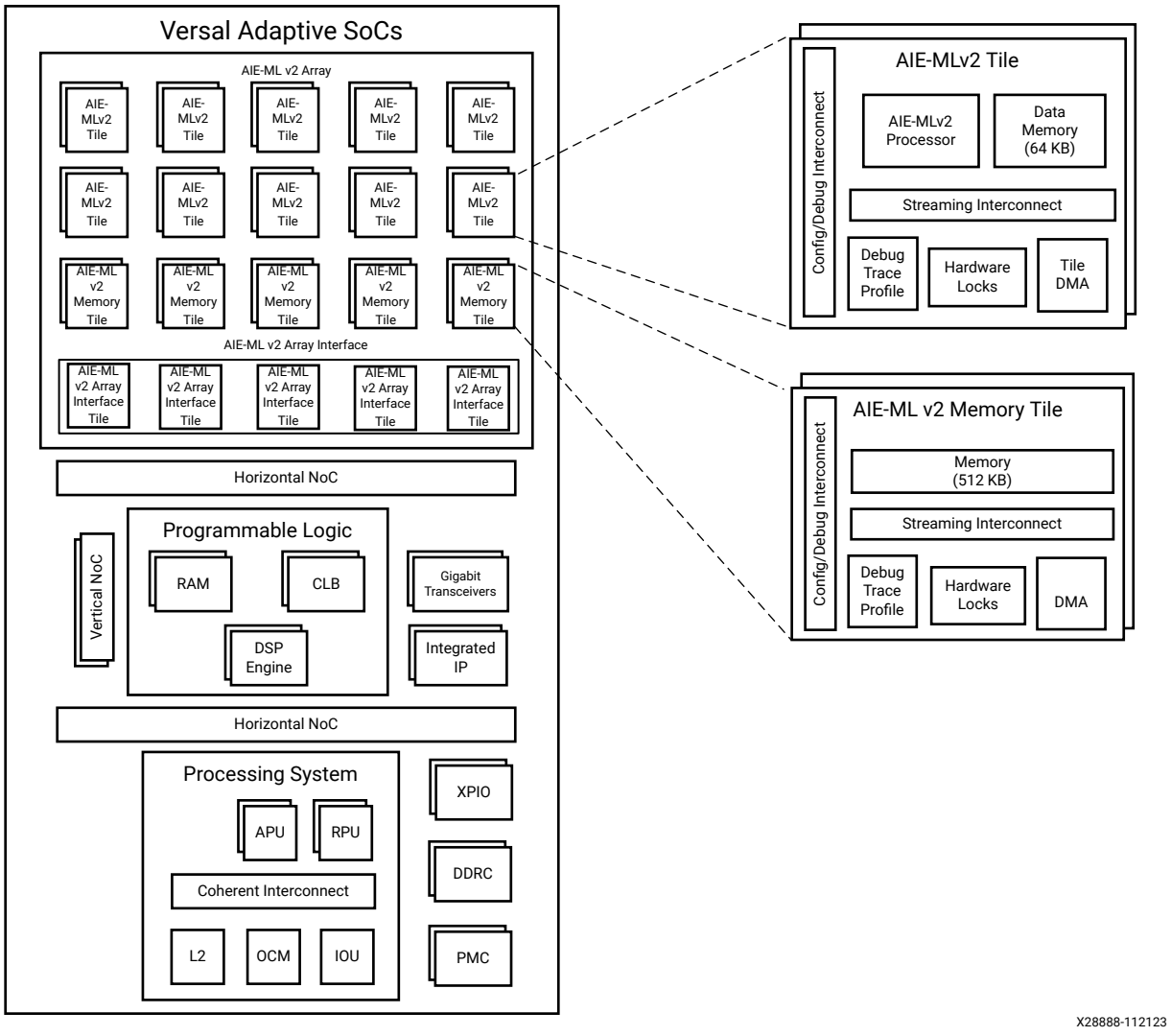
- A control module used for setting up the application, orchestrate execution and data movement during runtime for each column. This is also used to allow synchronization with external host.
- Configuration through the memory-mapped AXI4 interface of local registers and control module.

- Streaming interconnect provides routing between PL resources and array tiles.
- AIE-ML v2 to programmable logic (PL) interface that provides asynchronous clock-domain crossing between the AIE-ML v2 clock and the PL clock.
- AIE-ML v2 to NoC interface logic to the NoC manager unit (NMU) and NoC subordinate unit (NSU) components.
- Hardware synchronization primitives leverage features from the AIE-ML v2 tile locks module.
- Debug, trace, and profile functionality that leverage all the features from the AIE-ML v2 tile.

AIE-ML v2 Array Overview

The following figure shows the high-level block diagram of a Versal adaptive SoC with an AIE-ML v2 array in it. The device consists of the processor system (PS), programmable logic (PL), and the AIE-ML v2 array.

Figure 1: **Versal Device (with AIE-ML v2) Top-Level Block Diagram**



The AIE-ML v2 array is the top-level hierarchy of the AIE-ML v2 architecture. It integrates a two-dimensional array of AIE-ML v2 tiles. Each AIE-ML v2 tile integrates a very-long instruction word (VLIW) processor, integrated memory, and interconnects for streaming, configuration, and debug. The AIE-ML v2 array introduces a separate functional block, the memory tile, that is used to significantly reduce PL resources (LUTs and UltraRAMs) for ML applications. The memory tile has 512 KB data memory, 12 DMA channels (eight can access neighboring memory tiles), and stream interfaces. Depending on the device there can be one or two rows of memory tiles. The AIE-ML v2 array interface enables the AIE-ML v2 array to communicate with the rest of the Versal device through the NoC or directly to the PL. The AIE-ML v2 array also interfaces to the processing system (PS) and platform management controller (PMC) through the NoC.

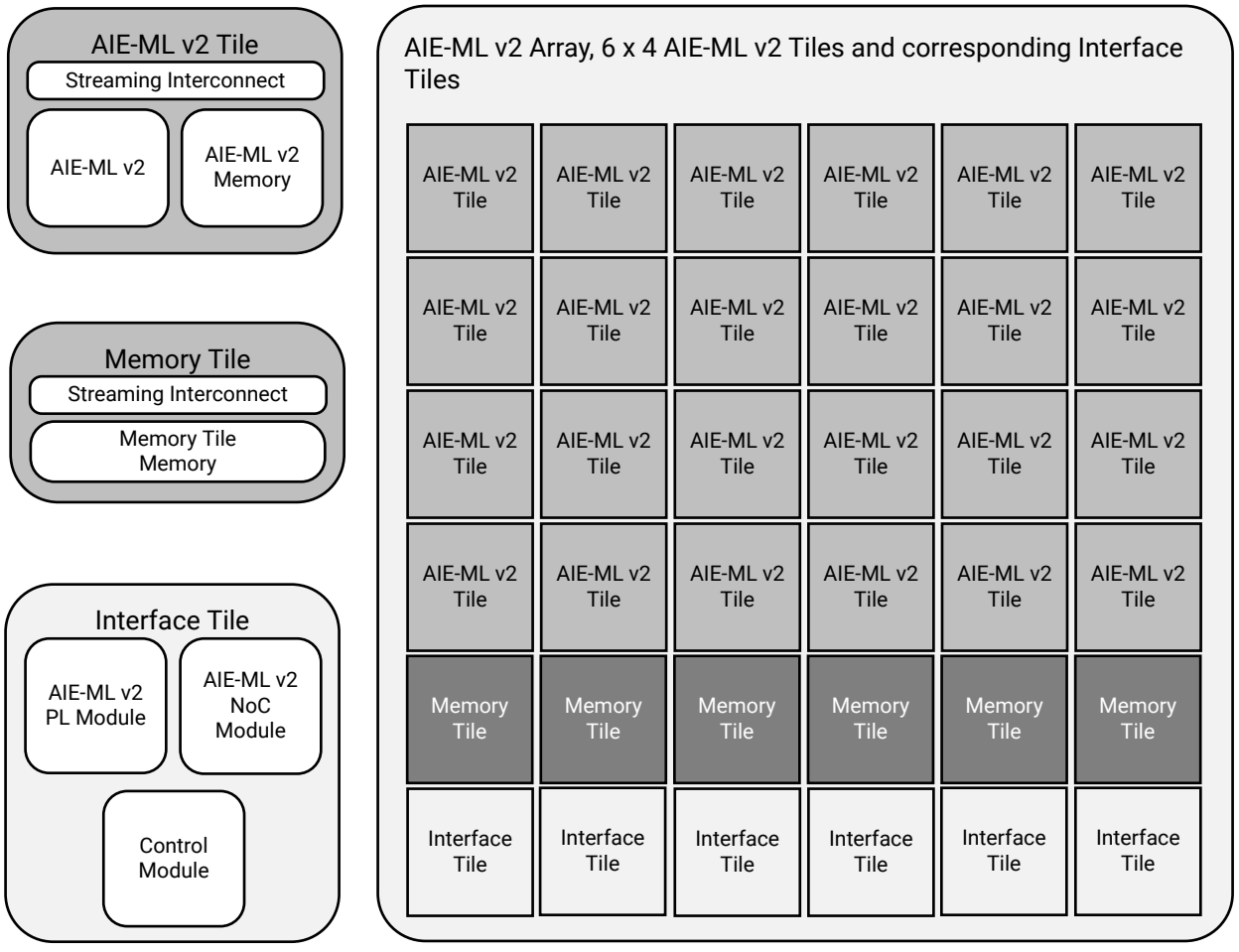
Versal adaptive SoC devices that integrate AIE-ML v2 tiles have access to the following types of memory:

- External DDR memory (via NoC)
- On-chip PL memory resources (UltraRAM/block RAM)
- On-chip shared memory in AIE-ML v2 memory tiles
- On-chip local data memory in AIE-ML v2 tiles

AIE-ML v2 Array Hierarchy

The AIE-ML v2 array is made up of AIE-ML v2 tiles, one or two rows of AIE-ML v2 memory tiles, and AIE-ML v2 array interface-tiles (the last row of the array). The following figure shows a conceptual view of the complete tile hierarchy associated with the AIE-ML v2 array. See Chapter 2: AIE-ML v2 Tile Architecture, Chapter 3: AIE-ML v2 Array Interface Architecture, and Chapter 4: AIE-ML v2 Architecture for detailed descriptions of the various tiles.

Figure 2: **Hierarchy of Tiles in an AIE-ML v2 Array**



Performance

The AIE-ML v2 array has a single clock domain for all the tiles and array interface blocks. The performance targed of the AIE-ML v2 array for the –1 speed grade devices is 1 GHz with V_{CC_AIE} at 0.725V. In addition, the AIE-ML v2 array has clocks for interfacing to other blocks. The following table summarizes the various clocks in the AIE-ML v2 array and their performance targets.

Table 1: **AIE-ML v2 Interfacing Clock Domains**

Clock	Target for –1L	Source	Relation to AIE-ML v2 Clock
AIE-ML v2 array clock	1 GHz	AIE-ML v2 PLL	N/A
NoC clock	1000 MHz	NoC clocking	Asynchronous, clock domain crossing (CDC) within the NoC
PL clocks	500 MHz	PL clocking	Asynchronous, CDC within AIE-ML v2 array interface
NPI clock	300 MHz	NPI clocking	Asynchronous

The following table shows the minimum bandwidth of the AXI4 memory-mapped interface by target block, under the following conditions:

- The AXI4 memory-mapped manager operates ideally.
- Access occurs in bulk and contiguously.
- Bandwidth measures at the AIE-ML v2 array AXI4 NoC subordinate interface (NSU).
- AIE-ML v2 array clock runs at 1 GHz.

The actual bandwidth varies depending on:

- Internal bottlenecks and other non-idealities of the AXI4 memory-mapped manager (for example, PS, DMA in PMC, custom AXI4 manager in PL).
- Contention in the NoC.
- The AXI4 memory-mapped burst size and data access pattern.
- The AIE-ML v2 array clock frequency.

Table 2: **AIE-ML v2 Internal AXI4 Memory-mapped Bandwidth at an AIE-ML v2 Clock of 1 GHz**

	128-bit Access		32-bit Access	
Block	Target Write BW (GB/s)	Target Write BW (GB/s)	Target Write BW (GB/s)	Target Write BW (GB/s)
Tile and memory tile data memory	3.0	1.1	0.80	0.32
Tile program memory	2.9	1.1	0.32	0.32

Table 2: **AIE-ML v2 Internal AXI4 Memory-mapped Bandwidth at an AIE-ML v2 Clock of 1 GHz** *(cont'd)*

	128-bit Access		32-bit Access	
Tile, memory tile, and array interface tile stream-switch and BD registers	3.0	2.0	1.7	0.80

Clocking Structure

Each AIE-ML v2 interface tile has a column clock gate control register that controls the clock to all the memory tiles and AIE-ML v2 tiles in the same column. The register does not affect the clock of the AIE-ML v2 interface tile. When all the memory tiles and AIE-ML v2 tiles in a column are unused, disabling its clock through this control register reduces the power consumption of the AIE-ML v2 array.

Memory Error Handling

Each AIE-ML v2 tile has 64 KB of data memory and 16 KB of program memory. Due to the large amount of memory in the AIE-ML v2 tiles, protection is provided against soft errors. The AIE-ML v2 program memory 16 KB has ECC protection. The 64 KB of data memory is divided into eight memory banks each of size 8 KB. Two of these banks have ECC protection and the remaining six banks has parity protection. The memory tile data memory has ECC protection.

The program memory is protected with two 8-bit ECC. Each of the 8-bit ECC covers 64-bits. The 8-bit ECC can detect 2-bit errors and detect/correct a 1-bit error within the 64-bit word.

There are eight memory banks in data memory module. The first two memory banks have 7-bit ECC protection for each of the eight 32-bit lanes in the 256-bit word. The 7-bit ECC can detect 2-bit errors and detect/correct a 1-bit error. The last six memory banks have an odd parity for each 32-bit within 256-bit word. If parity error occurs, events are generated from the memory module.

The AIE-ML v2 memory tile has 512 KB of memory formed from eight banks of 64 KB. All memory banks have 7-bit ECC protection on each of the eight 32-bit lines in the 256-bit word. This ECC corrects 1-bit errors within 32-bit words and detects 2-bit errors within 32-bit words.

The control module contains three SRAMs. It has 32 KB of program memory and a data memory of 16 KB with a 32-bit interface. It also has a shared memory of 32 KB with a 128-bit interface. The ECC is generated and checked at 32-bit data granularity for the shared memory. In case there are correctable and uncorrectable errors, events are generated. The ECC is checked at 32-bit granularity for program memory and data memory.

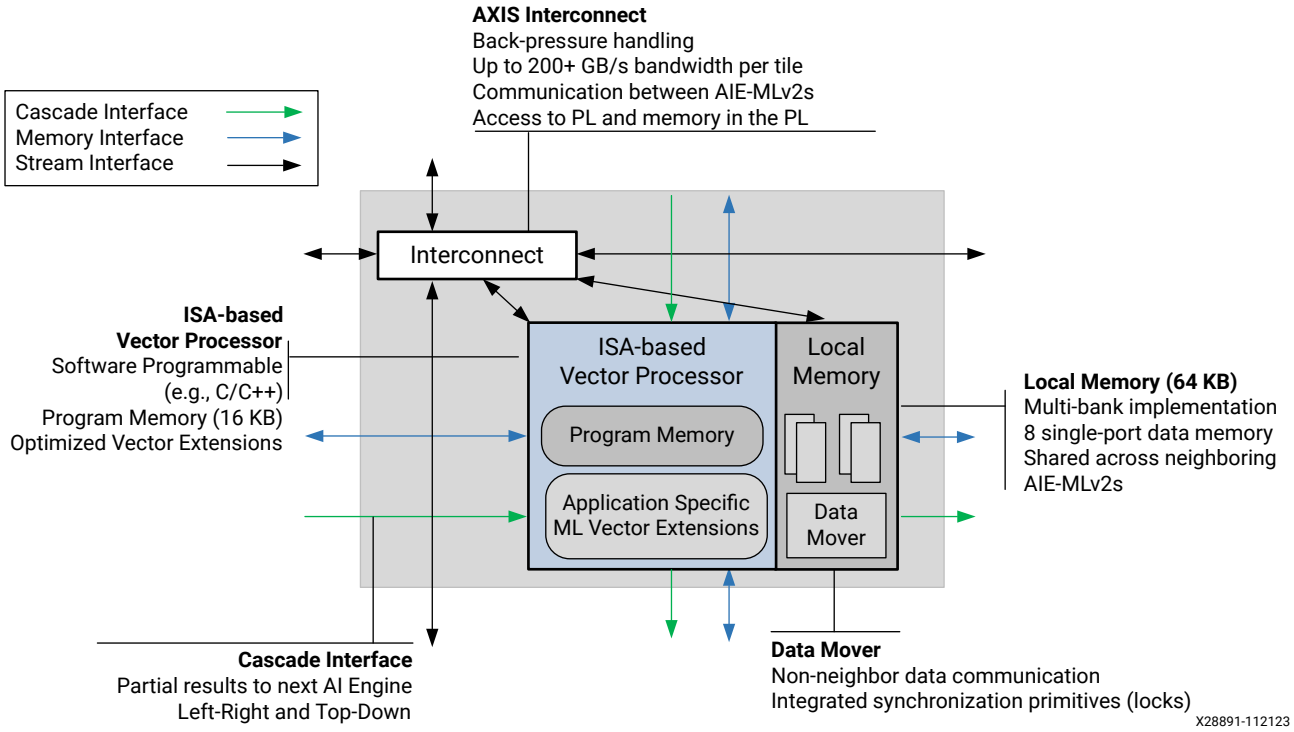
Privileged Register Access Control

AIE-ML v2 has a few privileged registers. These registers exist in the AIE-ML v2 tile, memory, and array interface-tiles. The security mechanism controls write requests to these registers and has no effect on reads to these registers. Per the AXI4 protocol, an AXI4 write transaction with `AWPROT[1]=0` indicates that the transaction is secure. AIE-ML v2 allows secure transactions to write to privileged registers in the array-interface tile registers. It also allows an external privileged manager to have secured write transactions.

AIE-ML v2 Tile Architecture

The top-level block diagram of the AIE-ML v2 tile architecture, key building blocks, and connectivity for the AIE-ML v2 tile are shown in the following figure.

Figure 3: **AIE-ML v2Tile Block Diagram**



The AIE-ML v2 tile consists of the following high-level modules:

- Tile interconnect
- AIE-ML v2
- AIE-ML v2 memory module

The tile interconnect module manages incoming and outgoing AXI4-Stream and memory-mapped AXI4 traffic. The subsequent sections elaborate on the functioning of the memory-mapped AXI4 and AXI4-Stream interconnect. Within the AIE-ML v2 memory module, there are eight memory banks totaling 64 KB of data memory. This module includes a memory interface, DMA functionality, and locks. Both incoming and outgoing directions are equipped with DMA, while each memory module contains a locks block. The AIE-ML v2 tile can access memory modules in

all four directions as a single contiguous block of memory. The memory interface appropriately directs memory accesses based on the address generated from the AIE-ML v2. Featuring a scalar datapath, a vector datapath, three address generators, and 16 KB of program memory, the AIE-ML v2 facilitates efficient processing. Upon boot and reset, the program and data memory of the AIE-ML v2 tile initialize to zero. Additionally, the AIE-ML v2 includes a cascade stream access mechanism, allowing the forwarding of accumulator output to the subsequent AIE-ML v2 tile.

The AIE-ML v2 is described in more detail in Chapter 4: AIE-ML v2 Architecture. Both the AIE-ML v2 tile and the AIE-ML v2 memory module have control, debug, and trace units. Some of these units are described later in this document:

- Control and status registers
- Events, event broadcast, and event actions
- Performance counters for profiling and timers

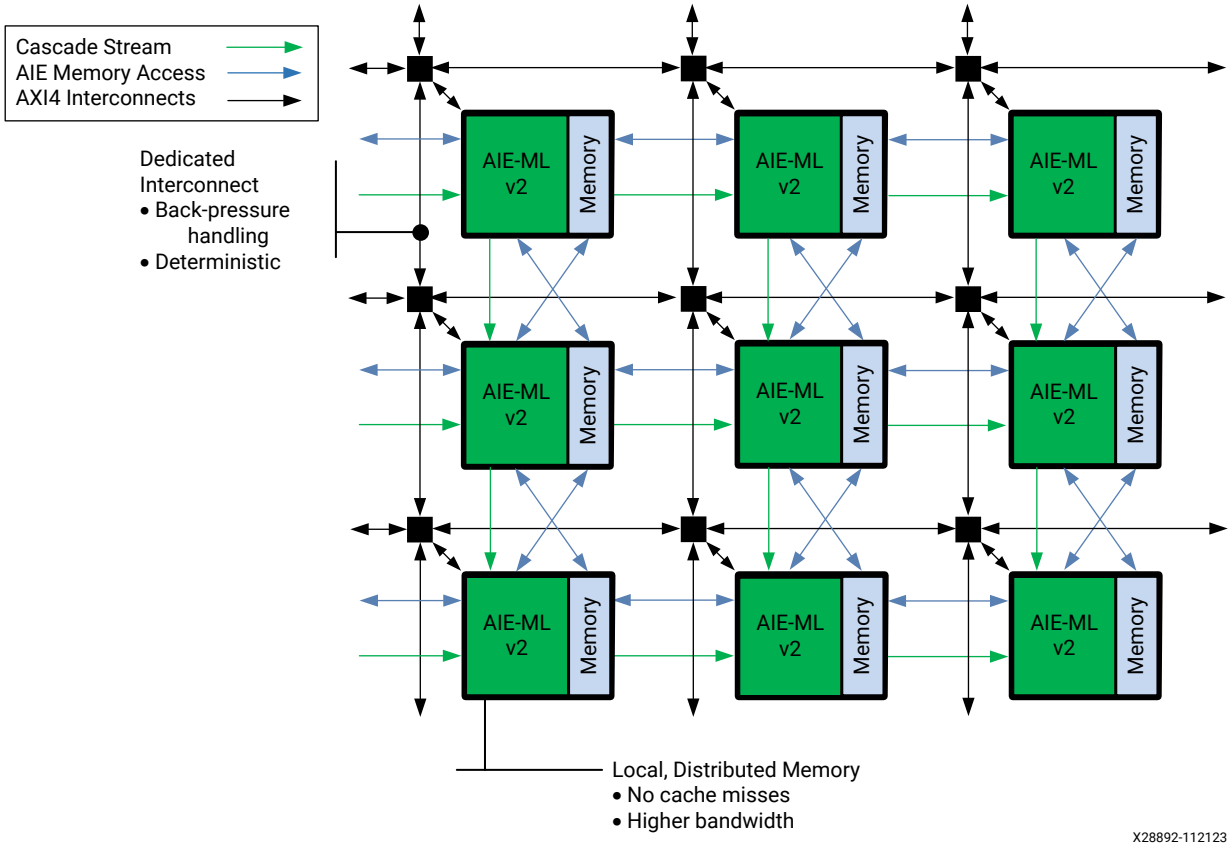
Figure 4 shows the connectivity across AIE-ML v2 tiles within the AIE-ML v2 array. This structure enables neighboring AIE-ML v2 to exchange data via shared data memory.

The architecture is designed to enable each AIE-ML v2 unit to interface with up to four distinct memory modules. These modules encompass:

- Its own local memory module
- The module positioned to the north
- The module situated to the south
- The module placed towards the west

However, the AIE-ML v2 units located at the edges of the array might have access to a slightly limited memory configuration compared to those centrally positioned within the array. This difference results in edge units having access to one or two fewer memory modules than their counterparts positioned more centrally. This distinction arises due to the bordering placement of these units within the array's structure.

Figure 4: **AIE-ML v2 Array**



Together with the flexible and dedicated interconnects, the AIE-ML v2 array provides deterministic performance, low latency, and high bandwidth. The modular and scalable architecture allows more compute power as more tiles are added to the array.

The AIE-ML v2 has both horizontal and vertical cascade connections, directed from north to south and from west to east. The cascade start points and end points are tied off at the array edges.

Memory-mapped AXI4 Interconnect

Each AIE-ML v2 engine tile contains a memory-mapped AXI4 interconnect for use by external blocks to write to or read from any of the registers or memories in the AIE-ML v2 tile. The memory-mapped AXI4 interconnect inside the AIE-ML v2 array can be driven from outside of the array by any memory-mapped AXI4 manager that can connect to the NoC. All internal resources in an AIE-ML v2 tile including memory, and all registers in an AIE-ML v2 tile and AIE-ML v2 memory module, are mapped onto a memory-mapped AXI4 interface.

Each AIE-ML v2 tile has a memory-mapped AXI4 switch that accepts all memory-mapped AXI4 requests from the south direction. If the address is for the tile, the request is served locally. Otherwise, the request is passed to the next tile in the north direction. Each tile has 1 MB address space.

The following table shows the addressing scheme of memory-mapped AXI4 in the AIE-ML v2 tile. The lower 20 bits represent the tile address range of 0x00000 to 0xFFFFF, followed by five bits that represent the row location and seven bits that represent the column location.

Table 3: **AIE-ML v2 Memory-Mapped AXI4 Tile Addresses**

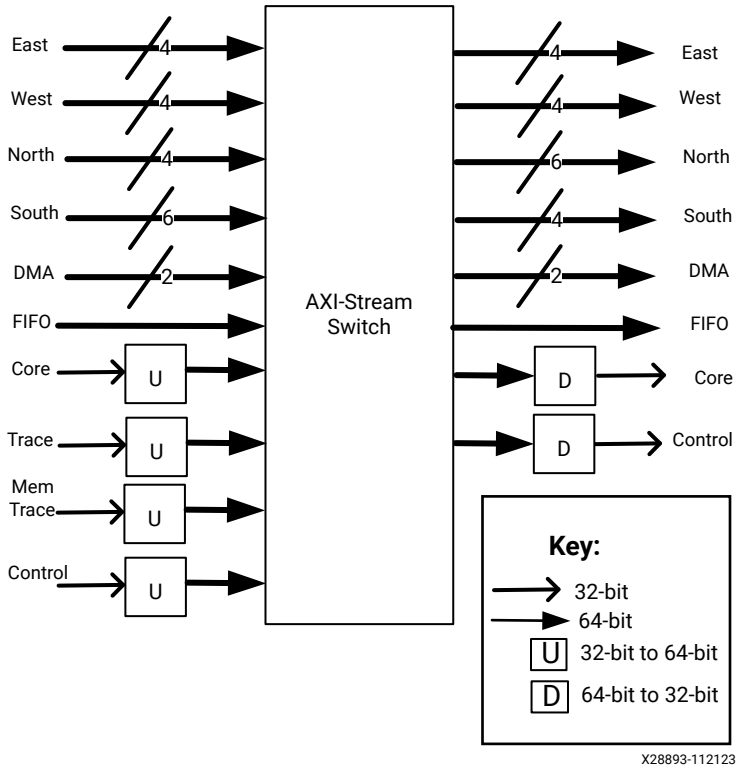
Column	Row [22:18]	AIE-ML v2 Tile/AIE-ML v2 Memory Tile/AIE-ML v2 Array Interface-Tile
[31:25] (7b)	[24:20] (5b)	[19:0] (20b)

AXI4-Stream Interconnect

Each AIE-ML v2 tile has an AXI4-Stream interconnect (alternatively called a stream switch) that is a fully programmable, a 64-bit AXI4-Stream crossbar, and is statically configured through the memory-mapped AXI4 interconnect. It handles backpressure and is capable of the full bandwidth on the AXI4-Stream. The AXI4-Stream interconnect has the following properties:

- All elements of the switch data-path are 64-bit, which supports the transport of two 32-bit-words in parallel per cycle.
- A `TKEEP` signal is added to support transfer of an odd number of 32-bit-words.
- Stream parity is calculated on the 32-bit word and two parity bits transported with the data.
- All external south, west, north, east, and stream-FIFO ports are capable of full throughput of 64-bit per cycle.
- Sources and destinations of DMA which can run at full throughput of 64-bit per cycle.
- Core, trace, and control-packet blocks have a 32-bit stream interface, so an adapter is placed for 32 <-> 64-bit conversion.

Figure 5: **AXI4-Stream Switch High-Level Block Diagram**



The tile ports are split into external ports and local ports. The external ports are south, west, north, and east. The local ports are core, DMA, FIFO, and trace. In the following description, a double-word refers to two adjacent 32-bit words on the stream, and in the case of `TLAST=1` and `TKEEP=0`, a double-word refers to 32-bit word and the adjacent null-word.

- External and local slave ports have two cycles of latency and four double-words of buffering.
- Local master ports have one cycle of latency and two double-words of buffering.

Therefore, excluding packet switch arbitration overhead, the latency and buffering crossing the switch is external to external.

- Local slave to local master: three cycles of latency, six double-words of buffering
- Local slave to external master: four cycles of latency, eight double-words of buffering
- External slave to local master: three cycles of latency, six double-words of buffering
- External to external: four cycles of latency, eight double-words of buffering

The stream switch contains one 16-deep, 68-bit (64-bit data + 2-bit parity + 1-bit `TLAST` + 1-bit `TKEEP`) wide FIFO. Each stream port operates in either circuit-switched or packet-switched mode, selected by a packet-switching bit in the port's configuration register. A port cannot use both modes at the same time. In packet-switched mode, multiple logical streams can share the same physical port and wires.

Table 4: **AIE-ML v2 AXI4-Stream Tile Interconnect Bandwidth**

Connection Type	Number of Connections	Data Width (Bits)	Clock Domain (1 GHz)	Bandwidth per Connection (GB/s)	Aggregate Bandwidth (GB/s)
To North/From South	6	64	AIE-ML v2 array clock	8	48
To South/From North	4	64	AIE-ML v2 array clock	8	32
To West/From East	4	64	AIE-ML v2 array clock	8	32
To East/From West	4	64	AIE-ML v2 array clock	8	32

***Note*:** The previous table assumes the slowest speed grade. Higher bandwidth can be achieve with a higher speed grade.

A circuit-switched stream has a one-to-one or one-to-many topology: a single source port feeds one or more destination ports, and all data entering the source is broadcast to all configured destinations. Latency is deterministic. When bandwidth is constrained, built-in backpressure throttles the source, which reduces overall performance.

Packet-switched streams can share ports with other packet-switched streams, which introduces potential resource contention and therefore non-deterministic latency. Each packet-switched stream uses a 5-bit stream ID that must be unique among streams sharing the same ports; this ID determines the packet's destination(s). Packet switching supports flexible topologies, enabling any combination of single or multiple manager and subordinate ports within a stream.

A packet-switched packet has:

- **Packet header:** Routing and control information for the packet
- **Data:** Actual data in the packet
- **TLAST:** Last word in the packet must have `TLAST` asserted to mark the end of packet

Table 5: **Packet Header**

Odd Parity	3'b000	Source Column	Source Row	1'b0	Packet Type	7'b0000000	Stream ID
[31]	[30:28]	[27:21]	[20:16]	[15]	[14:12]	[11:5]	[4:0]

AIE-ML v2 Tile Program Memory

The AIE-ML v2 tile has a local 16 KB of program memory with a data-width of 128-bit that can be used to store VLIW instructions. There are two interfaces to the program memory:

- Memory-mapped AXI4 interface
- AIE-ML v2 interface

An external manager can read or write to the program memory using the memory-mapped AXI4 interface. The AIE-ML v2 tile has 128-bit wide interfaces to the program memory to fetch instructions. The AIE-ML v2 tile can read from, but not write to, the program memory. To access the program memory simultaneously from the memory-mapped AXI4 and AIE-ML v2 tile, divide the memory into multiple banks and access mutually exclusive parts of the program memory. Arbitration logic is needed to avoid conflicts between accesses and to assign priority when accesses are to the same bank. The core program memory load gets priority and the other requests get a `SLVERR` if a concurrent access is made to the program memory.

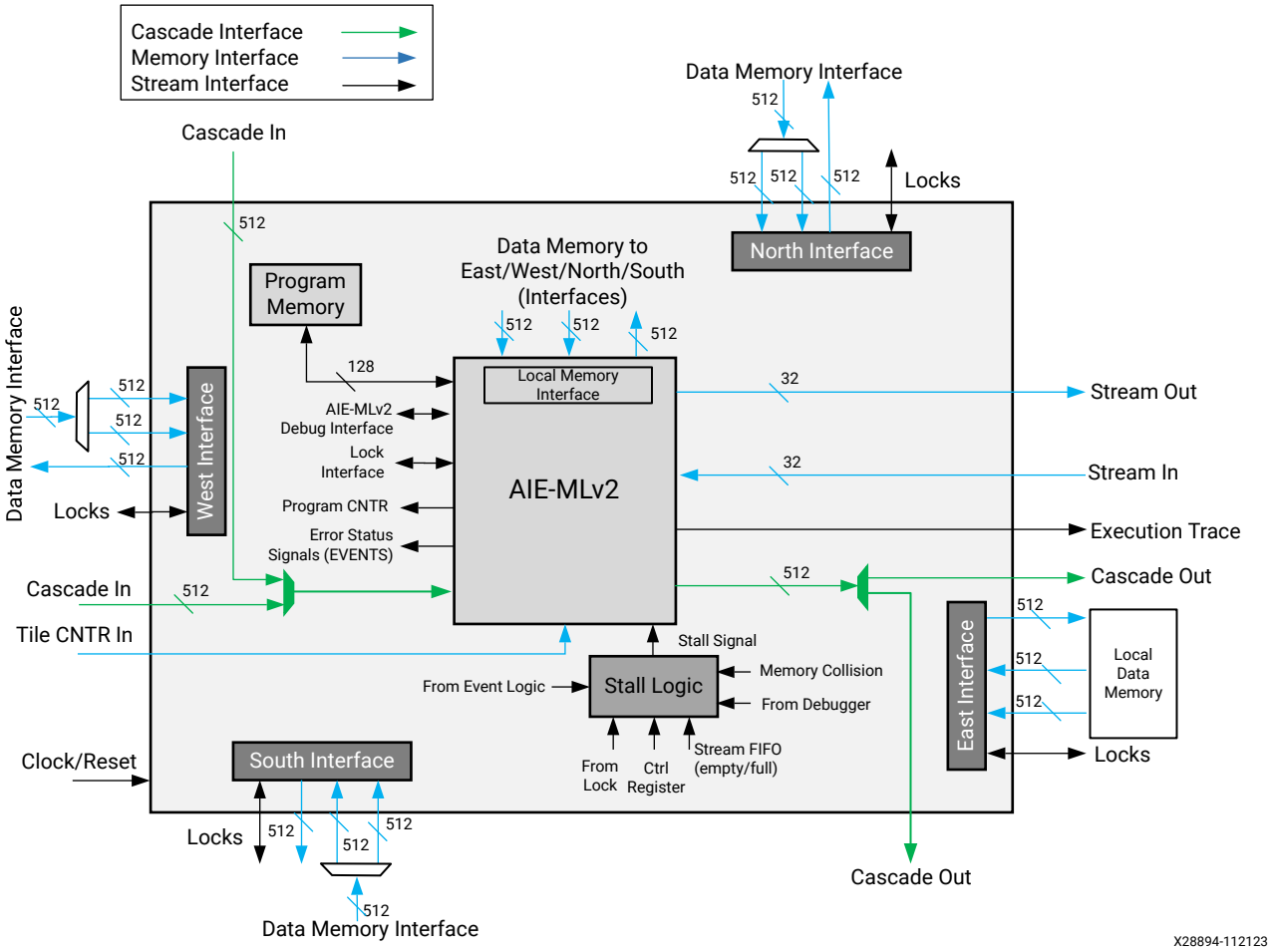
AIE-ML v2 Interfaces

The AIE-ML v2 has multiple interfaces. The following block diagram shows the interfaces.

- **Data Memory Interface:** The AIE-ML v2 core can access data memory modules on all four directions. They are accessed as one contiguous memory. The AIE-ML v2 core has two 512-bit wide load units and one 512-bit wide store unit. From the AIE-ML v2 perspective, the throughput of each of the loads (two) and store (one) is 512 bits per clock cycle.
- **Program Memory Interface:** This 128-bit wide interface is used by the AIE-ML v2 core to access the program memory. A new instruction can be fetched every clock cycle.
- **Direct AXI4-Stream Interface:** The AIE-ML v2 core has one 32-bit input AXI4-Stream interface and one 32-bit output AXI4-Stream interface. Each stream allows the AIE-ML v2 core to have a four-word (128-bit) access every four cycles, or a one-word (32-bit) access every cycle.
- **Cascade Stream Interface:** The 512-bit accumulator data from one AI Engine can be forwarded to another by using these cascade streams to form a chain. Using this stream, the core can transfer accumulator data to the next core. There is a small FIFO which accumulates up to two 512-bit wide words on both input and output streams. This allows the storing of four accumulator words in FIFO between each AIE-ML v2 core.
- **Debug Interface:** This interface is able to read or write all AIE-ML v2 core registers over the memory-mapped AXI4 interface.
- **Hardware Synchronization (Locks) Interface:** This interface allows synchronization between two AIE-ML v2s or between an AIE-ML v2 and DMA. The AIE-ML v2 can access the lock modules in all four directions. There is also added support for semaphore locks.
- **Stall Handling:** An AIE-ML v2 can be stalled due to multiple reasons and from different sources. Examples include external memory-mapped AXI4 manager (for example, PS), lock modules, empty or full AXI4-Stream interfaces, data memory collisions, and event actions from the event unit.

- **AIE-ML Event Interface:** This 16-bit wide `EVENT` interface can be used to set different events. The tile provides 128 internal core module and 128 memory module events. Each event has a unique number.
- **Tile Timer:** The input interface to read the 64-bit timer value inside the tile.
- **Execution Trace Interface:** A 32-bit wide interface where the AIE-ML v2 generated packet-based execution trace can be sent over the AXI4-Stream.

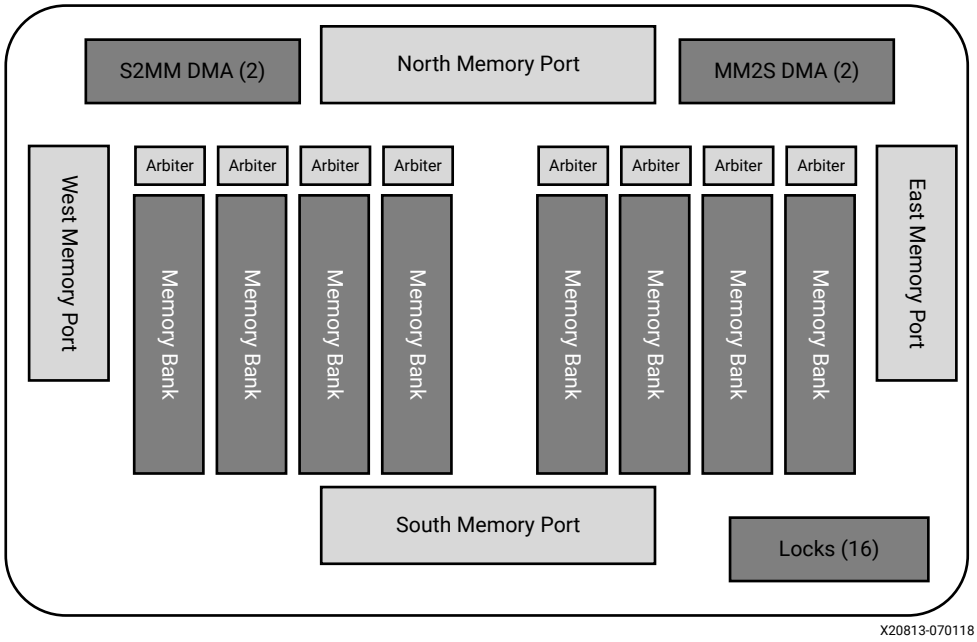
Figure 6: **AIE-ML v2 Interfaces**



AIE-ML v2 Memory Module

The AIE-ML v2 memory module (shown in the following figure) contains eight memory banks, two input streams to memory map (S2MM) DMA, two memory-map to output DMA streams (MM2S), and a hardware synchronization module (locks).

Figure 7: **AIE-ML v2 Memory Module**



- **Memory Banks:** The AIE-ML v2 data memory is 64 KB, organized as eight memory banks, where each memory bank is a 256 word x 256-bit single-port memory. From a programmer's perspective, a pair of hardware banks are interleaved at 256-bit granularity to present four programmer banks each being 512-bit wide. Banks 0 and 1 have ECC error correction. Banks 2–7 have parity protection.
- **Memory Arbitration:** Arbitration is performed independently for each bank with parallel accesses to different banks supported. Therefore, there are eight independent arbiters per tile DM. Arbitration is between all accessors with active requests. The arbitration is hierarchical, with accessors arranged into fixed groups.
- **Tile DMA Controller:** The tile DMA connects to the AIE-ML v2 stream switches with two incoming and two outgoing streams. It consists of two modules: S2MM, which writes 64-bit stream data to memory, and MM2S, which reads 64-bit data from memory to a stream. DMA transfers are defined by buffer descriptors; the controller supports 16 descriptors, accessible and configurable from any memory-mapped AXI4 manager. Each descriptor contains all

parameters for a transfer and can optionally chain to the next descriptor for continuous operation. The controller also manages 16 locks used for synchronization between the AIE-ML v2 array, the DMA, and any external memory-mapped AXI4 managers. Descriptors can be configured to use these locks through the same AXI4 interface. The DMA datapath sustains a throughput of two 32-bit words per cycle.

The DMA controller supports:

- 4D tensor address generation (including iteration-offset)
- Task queues and task complete tokens
- S2MM finish on `TLAST` and handling of out-of-order packets
- Decompression on the two S2MM channels
- Compression to the two MM2S channels

- **Lock Module:** The AIE-ML v2 memory module contains a lock module to achieve synchronization amongst the AIE-ML v2s, tile DMA, and an external memory-mapped AXI4 interface manager (for example, the processor system (PS)). The AIE-ML v2 features 16 semaphore locks, each carrying a 6-bit unsigned as state. The lock module handles lock requests from the AIE-ML v2s in all four directions, the local DMA controller, and memory-mapped AXI4.

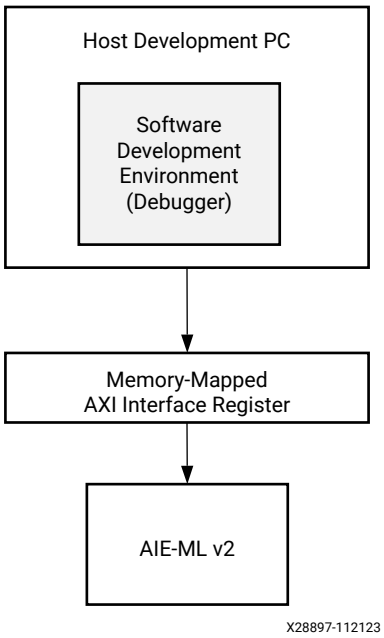
AIE-ML v2 Debug

Debugging the AIE-ML v2 uses the memory-mapped AXI4 interface. All the major components in the AIE-ML v2 array are memory-mapped:

- Program memories
- Data memories
- AIE-ML v2 registers
- DMA registers
- Lock module registers
- Stream switch registers
- AIE-ML v2 break points registers
- Events and performance counters registers

These memory-mapped registers can be read and/or written from any manager that can produce memory-mapped AXI4 interface requests (PS, PL, and PMC). These requests come through the NoC to the AIE-ML v2 array interface, and then to the target tile in the array. The following figure shows a typical debugging setup involving a software development environment running on a host development system combined with its integrated debugger.

Figure 8: **Overview of the AIE-ML v2 Debug Interface**



The debugger connects to the platform management controller (PMC) on an AIE-ML v2 enabled Versal device either using a JTAG connection or the AMD high-speed debug port (HSDP) connection.

AIE-ML v2 Trace and Profiling

The AIE-ML v2 tile has provisions for trace and profiling. It also has configuration registers that control the trace and profiling hardware.

Trace

The AIE-ML v2 tiles have provisions for trace streams. There are two trace units in the tile, one for the core module and another for the memory-module. Both these trace streams are connected to the tile stream switch. There is a trace unit in an AIE-ML v2 array interface tile. Each trace unit can collect trace data and send it out via the AXI4-Stream network.

The units can operate in the following modes:

- **AIE-ML v2 Tile Core Module Modes:**
  - Event-time
  - Event-PC
- **AIE-ML v2 Tile Memory Module Modes:** Event-time

- **AIE-ML v2 Memory Tile Module Mode:** Event-time
- **AIE-ML v2 Array Interface Tile Mode:** Event-time

The trace is output from the unit through the AXI4-Stream as an AIE-ML v2 packet-switched stream packet. The packet size is 8x32 bits, including one word of header and seven words of data. The stream ID in the packet header is set via a configuration register. The information contained in the packet header is used by the array AXI4-Stream switches to route the packet to any AIE-ML v2 destination it can be routed to, including AIE-ML v2 local data memory through the AIE-ML v2 tile DMA, external DDR memory through the AIE-ML v2 array interface DMA, and block RAM or UltraRAM through the AIE-ML v2 to PL AXI4-Stream.

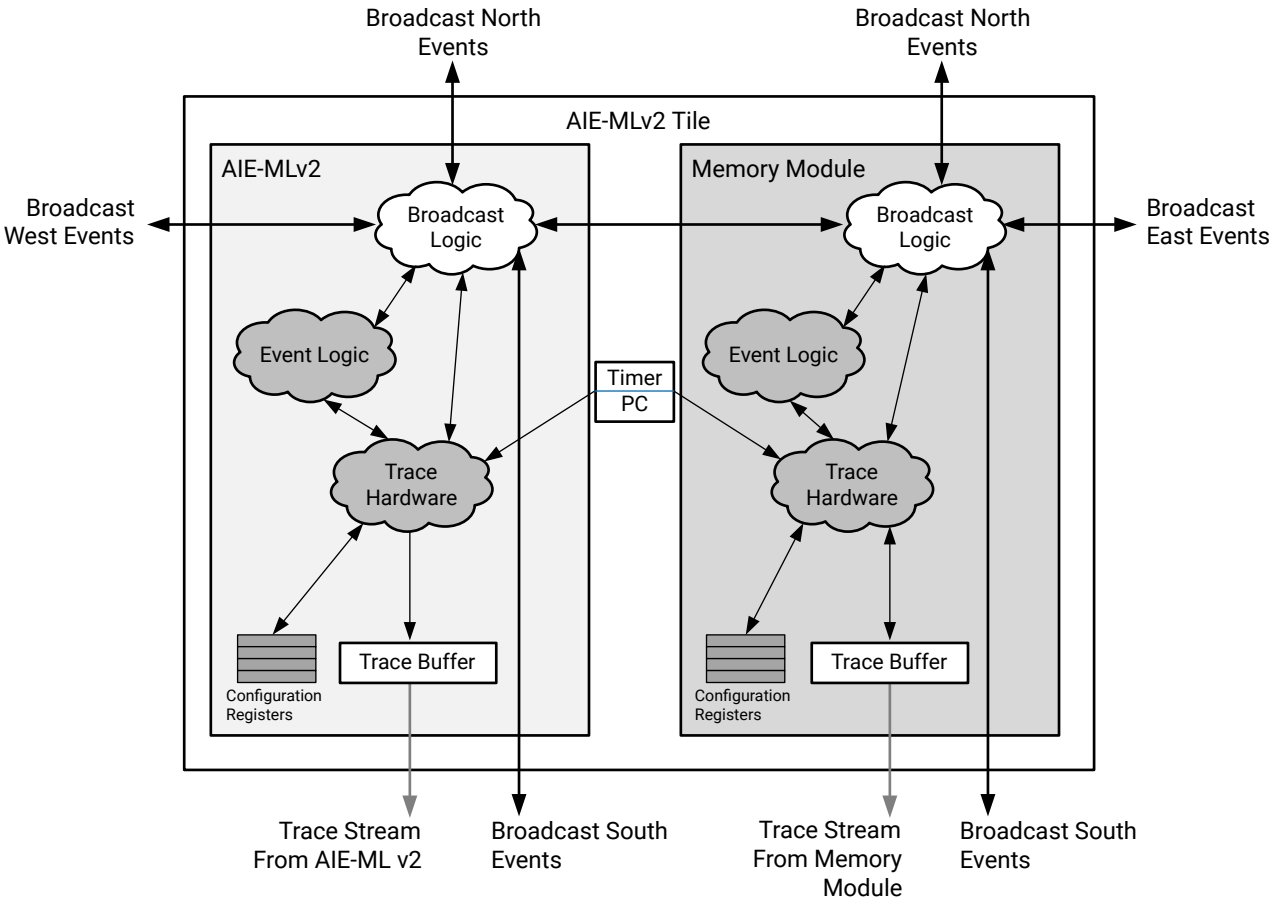
The event-time mode tracks up to eight independent numbered events on a per-cycle basis. A trace frame is created to record state changes in the tracked events. The frames are collected in an output buffer into an AIE-ML v2 packet-switched stream packet. Multiple frames can be packed into one 32-bit stream word, but they cannot cross a 32-bit boundary (filler frames are used for 32-bit alignment).

In the event-PC mode, a trace frame is created each cycle where any one or more of the eight watched events are asserted. The trace frame records the current program counter (PC) value of the AIE-ML v2 together with the current value of the eight watched events. The frames are collected in an output buffer into an AIE-ML v2 packet-switched stream packet.

The following figure shows the logical view of trace hardware in the AIE-ML v2 tile. The two trace streams out of the tile are connected internally to the event logic, configuration registers, broadcast events, and trace buffers.

Note: The different operating modes between the two modules are not shown.

Figure 9: **Logical View of AIE-ML v2 Trace Hardware**



To control the trace stream for an event trace, there is a 32-bit `trace_control0/1` register to start and stop the trace. There are also the `trace_event0/1` registers to program the internal event number to be added to the trace.

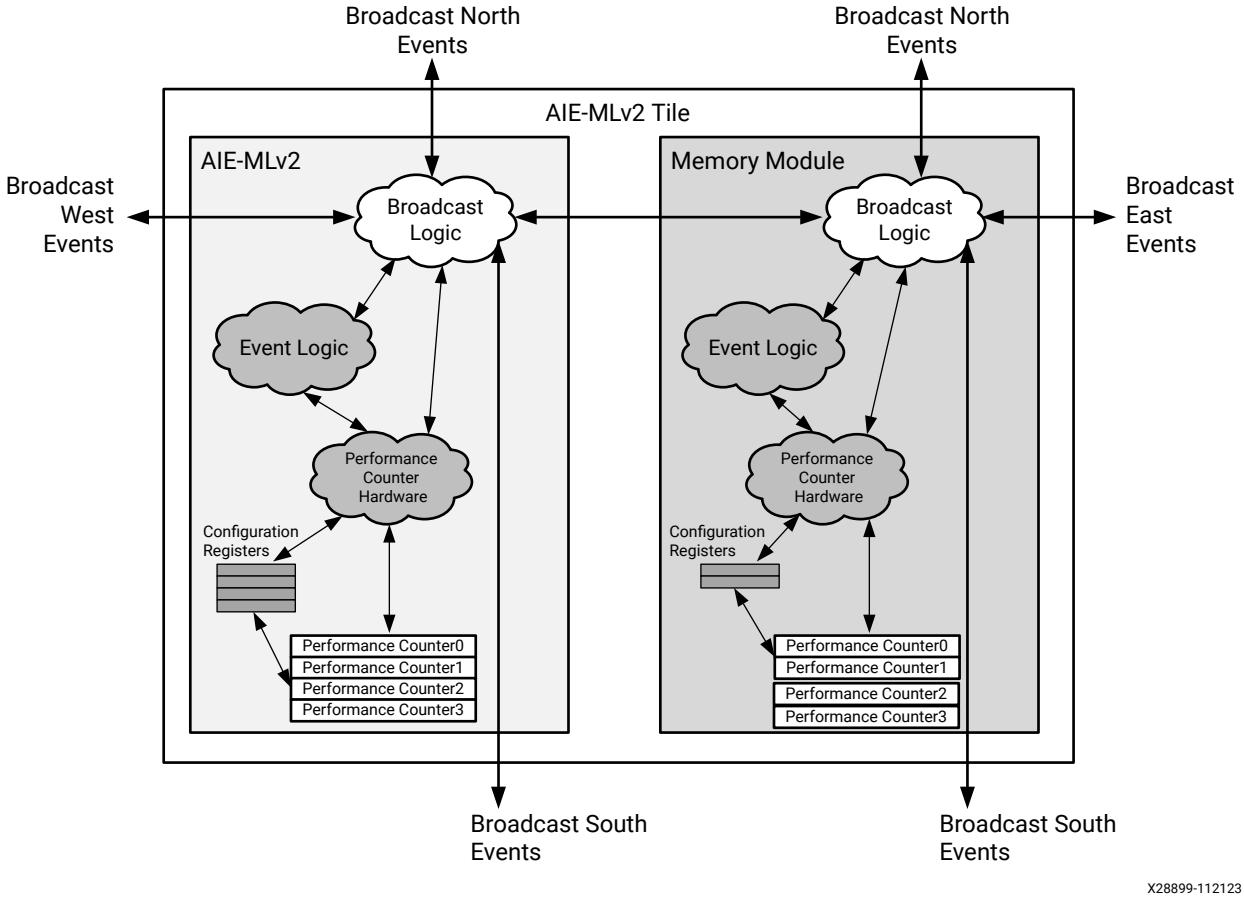
Profiling (Performance Counters)

The AIE-ML v2 array incorporates performance counters tailored for profiling purposes. Within the AIE-ML v2 and its associated memory module, each has four versatile performance counters available for configuration. These counters are capable of monitoring various internal events within the system. You can set them to either tally the occurrence of specific events or calculate the number of clock cycles between two defined events.

Moreover, the AIE-ML v2 memory tile and the AIE-ML v2 array interface are equipped with six performance counters, aligning their functionalities with those of the aforementioned counters. These counters are also customizable to execute similar monitoring tasks.

To offer a clearer understanding, the following figure illustrates a high-level logical representation of the profiling hardware integrated within the AIE-ML v2 tile. This hardware configuration is instrumental in facilitating performance monitoring and analysis within the AIE-ML v2 array.

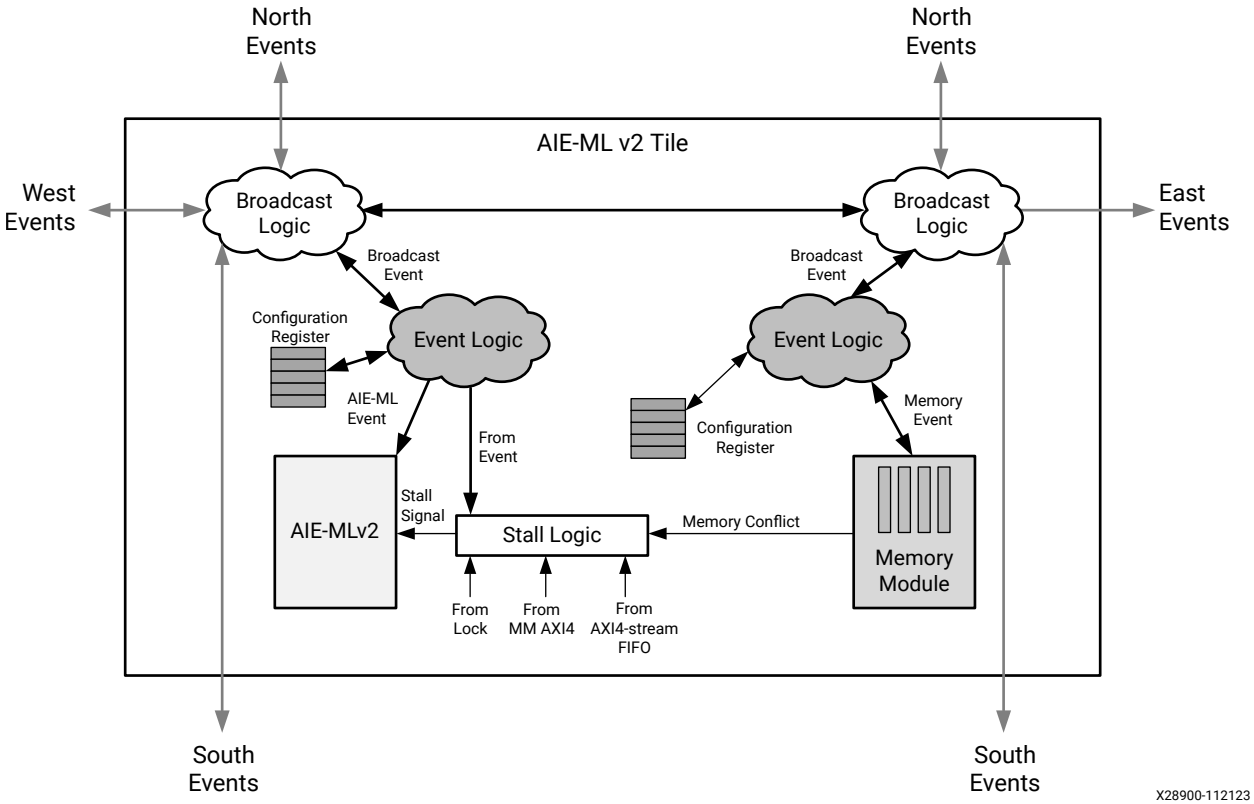
Figure 10: **Logical View of AIE-ML v2 Profiling**



AIE-ML v2 Events

The AIE-ML v2 and memory modules each have an event logic unit. Each unit has a defined set of local events. The following diagram shows the high-level logical view of events in the AIE-ML v2 tile. The event logic needs to be configured with a set of action registers that can be programmed over the memory-mapped AXI4 interface. There are separate sets of registers associated with event logic for the AIE-ML v2 and memory modules. Event actions can be configured to perform a task whenever a specific event occurs. Also, there is separate broadcast logic to send event signals to neighboring modules.

Figure 11: **Events in an AIE-ML v2 Tile**



Event Actions

An event itself does not have an action, but events can be used to create an action. Event broadcast and event trace can be configured to monitor the event. Examples of event actions include:

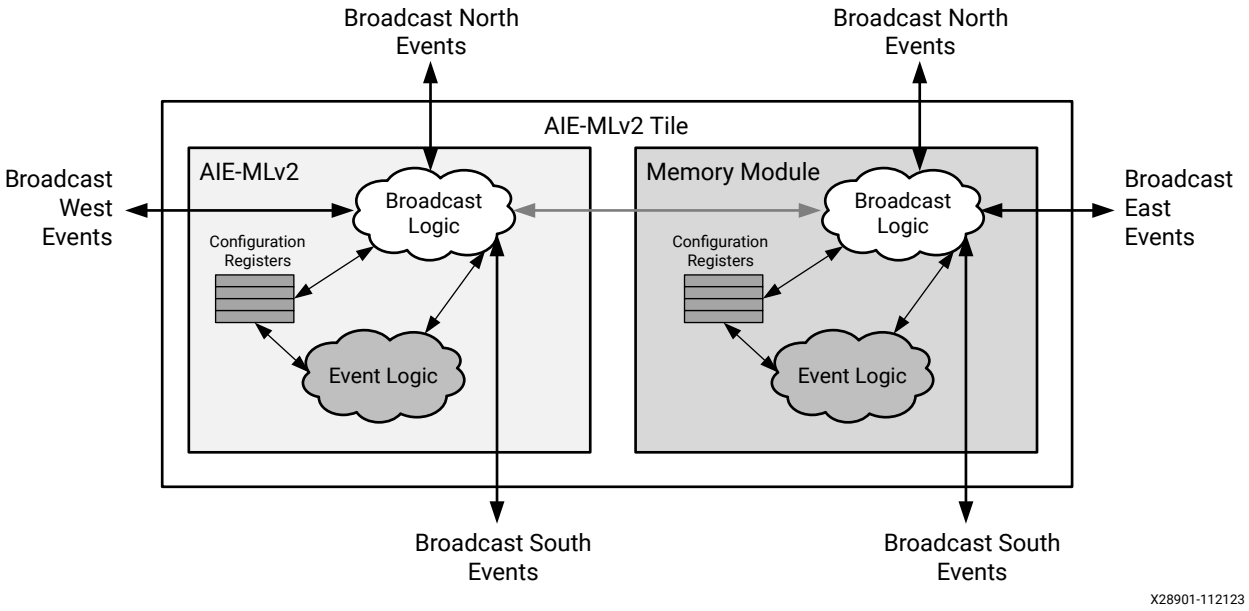
- Enable, disable, or reset of an AIE-ML v2
- Debug-halt, resume, or single-step of an AIE-ML v2
- Error halt of an AIE-ML v2
- Resynchronize timer
- Start and stop performance counters
- Start and stop trace streams
- Generate broadcast events
- Drive combo events
- ECC scrubbing event

For each of these event actions there are associated registers where a 7-bit event number is set and is used to configure the action to trigger on a given event.

Event Broadcast

Broadcast events are both the events and the event actions because they are triggered when a configured event is asserted. The following figure shows the logical view of the broadcast logic inside the AIE-ML v2 tile. The units in the broadcast logic in the AIE-ML v2 and memory modules receive input from and send out signals in all four directions. The broadcast logic is connected to the event logic, which generates all the events. There are configuration registers to select the event sent over, and mask registers to block any event from going out of the AIE-ML v2 tile.

Figure 12: **AIE-ML v2 Broadcast Events**



Each module has an internal register that determines the broadcast event signal to broadcast in the other directions. To avoid broadcast loops, the incoming event signals are ORed with the internal events to drive the outgoing event signals according the following list:

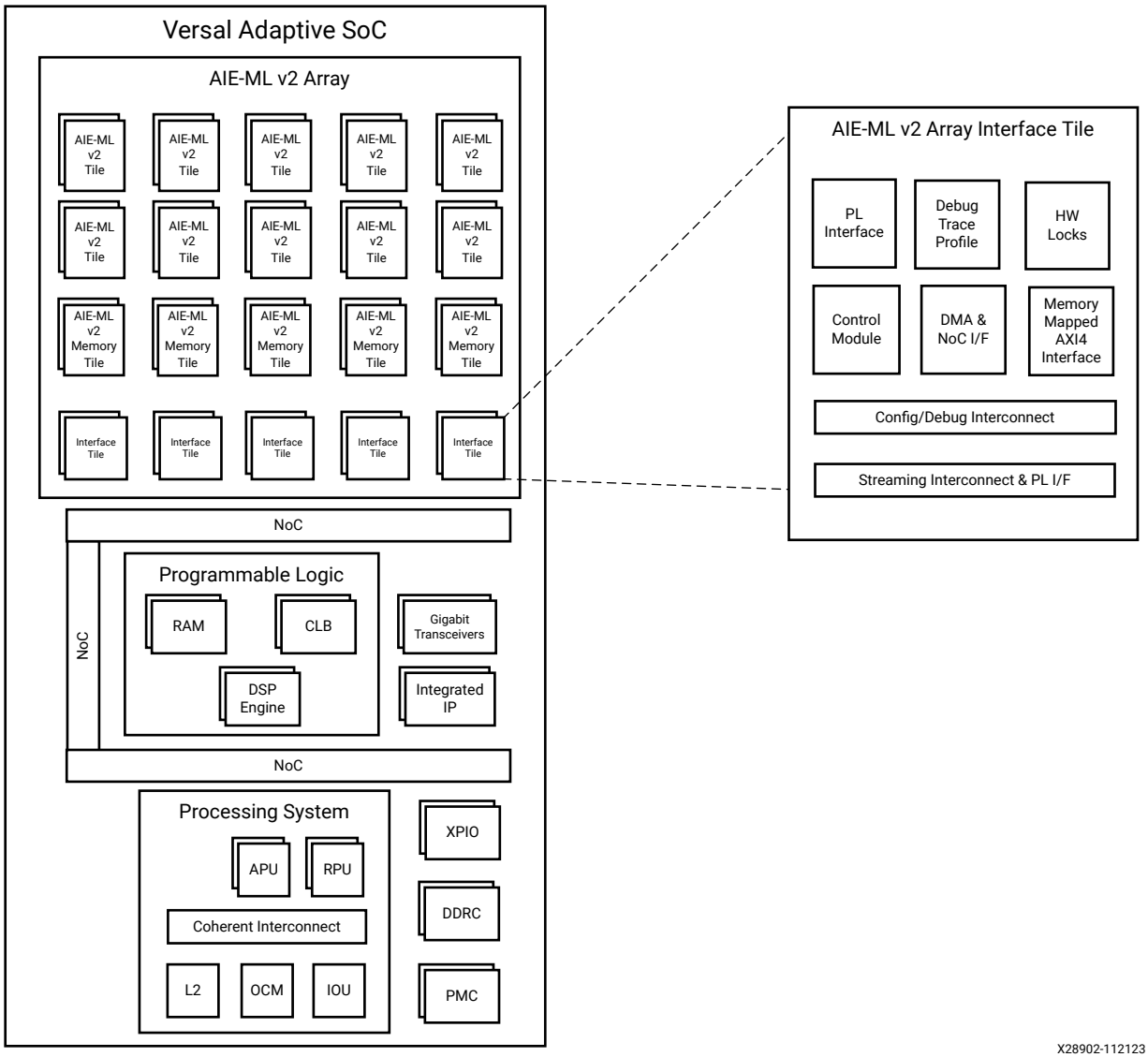
- Internal, east, north, south → west
- Internal, west, north, south → east
- Internal, south → north
- Internal, north → south

TIP: *The AIE-ML v2 module east broadcast event interface is internally connected to the memory module west broadcast event interface and does not go out of the AIE-ML v2 tile. In the AIE-ML v2 module, there are 16 broadcast events each in the north, south, and west directions. In the memory module, there are 16 broadcast events each in the north, south, and east directions.*

AIE-ML v2 Array Interface Architecture

The AIE-ML v2 is arranged in a 2D array as shown in the following figure. The AIE-ML v2 array interface provides the necessary functionality to interface with the rest of the device. There is a one-to-one correspondence of interface tiles for every column of the AIE-ML v2 array. The interface-tiles form a row and move memory-mapped AXI4 and AXI4-Stream data horizontally (left and right) and vertically up an AIE-ML v2 tile column. The AIE-ML v2 interface-tiles are based on a modular architecture, but the final composition is device specific. Refer to the following figure for the internal hierarchy of the AIE-ML v2 array interface in the AIE-ML v2 array.

Figure 13: **AIE-ML v2 Array Interface Hierarchy**



AIE-ML v2 Array Interface

The AIE-ML v2 array interface tile has NPI, POR, PLL, stream switch, CDT, event switch, control module, NoC, PL interface, two NMU interfaces, and an NSU interface. The array interface tile also supports AXI_MM isolation, which means AXI-MM transactions coming from west/east into array interface tiles are blocked.

The functionality of each of the modules in the array interface tiles are described as follows:

- **Power on Reset (POR):** The POR module is powered by the NoC voltage rail. This provides the necessary hardware reset signal to the AIE-ML v2 array.
- **AXI-MM Switch:** The AXI-MM switch provides functionality to move AXI4 horizontally within the AIE-ML v2 array interface tiles. The traffic can be directed vertically to the columns. This can also be used as a local interface for configuring local registers and has 1 MB of address space. This is a 32-bit interface.
  - Array interface tile west interface: requests to lower index columns
  - Array interface tile east interface: requests to higher index columns
  - Array interface tile north interface: requests to all other tiles in same column (AIE-ML v2 memory and AIE-ML v2 tile)
- **AXI4-Stream PL Interface:** This interface allows streams to and from the PL through the BLI interface. Each stream has a 64-bit interface but can be configured to operate as a 32-bit or 64-bit stream, or two physical streams can be configured to operate as a single 128-bit stream.Streams from the PL support a subset of the AXI4-Stream protocol:
  - When TLAST = 0, TKEEP is ignored and assumed to be all ones
  - When TLAST = 1, TKEEP is only supported at 32-bit granularity and valid words must be contiguous:
    - 32-bit streams: TKEEP must be 0xF
    - 64-bit streams: TKEEP can be 0x0F or 0xFF
    - 128-bit streams: TKEEP can be 0x000F, 0x00FF, 0x0FFF, 0xFFFF

Table 6: **AIE-ML v2 Array Interface to PL Interface Bandwidth Performance**

Connection Type	Number of Connections	Data Width (bits)	Clock Domain	Bandwidth per Connection (GB/s)	Aggregate Bandwidth (GB/s)
PL to AIE-ML v2 array interface	8	64[1]	PL (500 MHz)	4	32
AIE-ML v2 array interface to PL	6	64	PL (500 MHz)	4	24
AIE-ML v2 array interface to AXI4-Stream switch	8	64	AIE-ML v2 (1 GHz)	8	64
AXI4-Stream switch to AIE-ML v2 array interface	6	64	AIE-ML v2 (1 GHz)	8	48
Horizontal interface between AXI4-Stream switches[2]	4	64	AIE-ML v2 (1 GHz)	8	32

Notes:

1. If only 32 bits are used, the data must be on the lower half of the 64-bit bus.
2. The aggregate bandwidth shown is in the east/west direction. There are two sets of connections going in and out of each AXI4-Stream switch.

***Note*:** The previous table assumes the slowest speed grade. Higher bandwidth can be achieve with a higher speed grade.

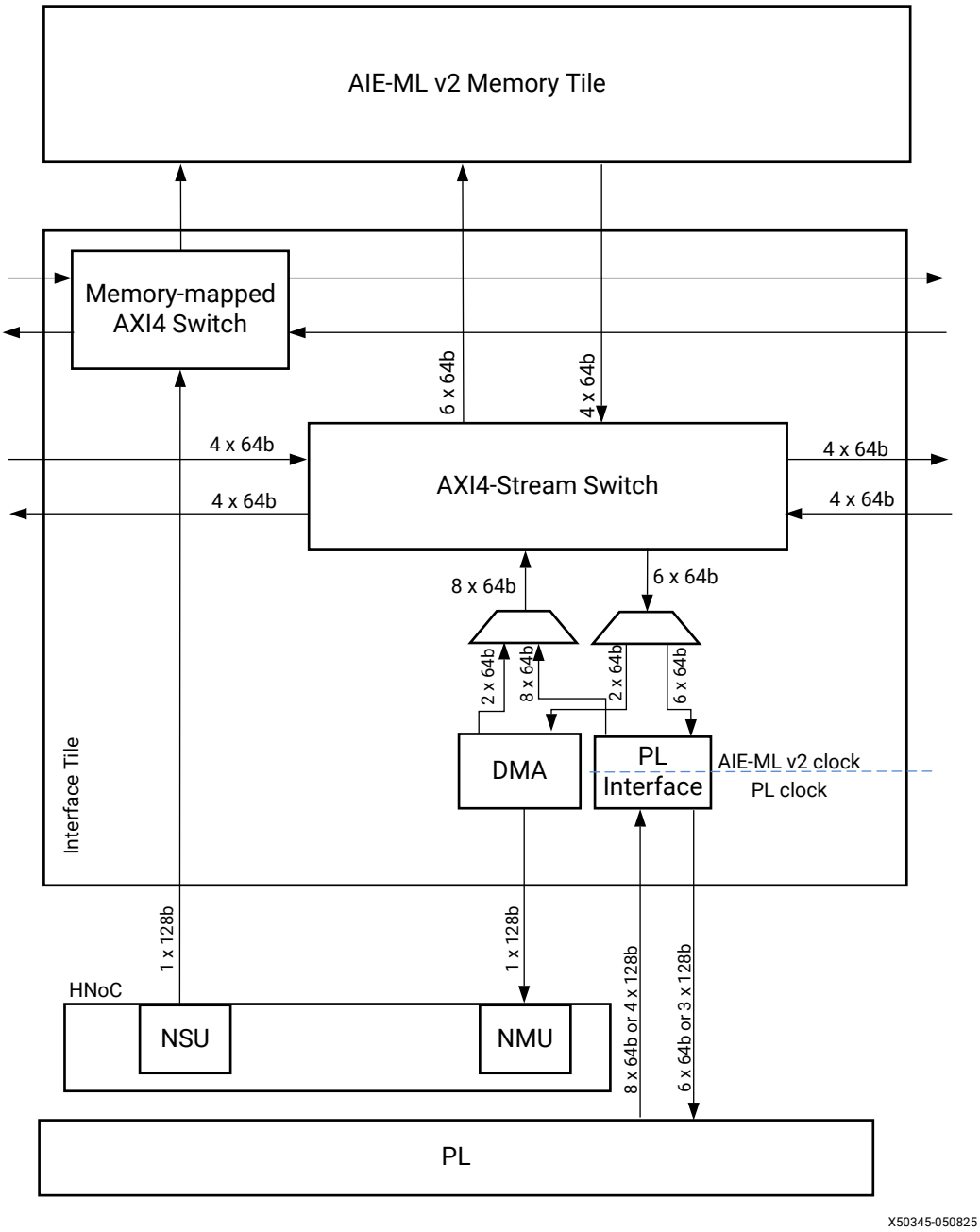
- **AXI4 Manager Interface (NMU):** This interface is used to carry AXI4 traffic from array interface tile DMA and control-interface. Array interface tile supports up to two of these interfaces and support passing traffic through neighbor NMU. This is a 128-bit interface and supports 48-bit addresses.
- **AXI4 Subordinate Interface (NSU):** This interface is used to carry AXI4 configuration, control, status and debug traffic from external controller and host debugger to the array interface tiles.
- **Event Broadcast Switch:** Broadcast events are both events and event actions because they are triggered when a configured event is asserted. The switch carries events to/from PL via BLI interface.
- **NPI and Interrupt:** The AIE-ML v2 array provides a single NPI subordinate (end-point) interface in the AIE-ML v2 array interface tile, which provides access to the AIE-ML v2 PLL configuration registers, global reset registers for AIE-ML v2 array and global AIE-ML v2 array register settings (security setting). Four of the interrupts are driven from the array interface tiles on to NPI Interrupts lines to PMC.

  AIE-ML v2 array interface tile contains hardware activity monitors that can be leveraged to provide telemetry data to allow power management schemes. This activity monitor shall capture the count of vector instructions executed by all the AIE-ML v2 tiles over a time period. The counter is of 40-bit width and can be read via two 32-bit NPI registers.
- **PLL:** A single, low-jitter dedicated PLL for the AIE-ML v2 clock generation. Depending on the device some devices require a PLL internal to the AIE-ML v2 array.
- **Stream Switch:** The main task of the AXI4-Stream switch is to carry deterministic throughput and high-speed circuit or packet data-flow between AIE-ML v2 and the programmable logic or NoC. It is designed to carry the bulk of the data movement to/from the AIE-ML v2 array. The switch data path has a 64-bit width and supports two 32-bit words in parallel per cycle. AXI4-Stream allows for data-flow to/from east, west, and north directions. It uses the PL or the DMA interface to/from south directions. All these paths have 64-bit data width. The switch also allows flow from/to the control module, trace, and control response. Each of these have a data width of 32-bit.
- **Control Module:** The control module performs application set-up, orchestration of execution and data movement during runtime, and synchronization with external host. This control module block has an AXI4 manager port connecting to the NoC, or read/write configuration in the array. An AXI4 subordinate port allows read/write to registers. This also includes an AXI4-Stream interface to handle task-complete-tokens and issue control-packets.
- **Control, Debug, and Trace Unit:** This unit leverages the features from the AIE-ML v2 tile for local event debugging, tracing, and profiling. The number of performance counters is six. The number of events has been increased to 256 to accommodate the additional events generated by control module. An input is added to the second level interrupt handler, driven by the control module.

- **NoC Interface:** This module handles the interface to and from the NoC in AIE-ML v2. The NoC module has the following modules:
  - DMA 2xS2MM, 2xMM2S channels, and BDs
  - Lock unit (16 locks)
  - Interface to NMU
  - Second level interrupt handler
- **DMA:** This DMA is composed of two S2MM channels, two MM2S channels, a shared pool of buffer descriptors, and a lock interface. The DMA supports a burst of up to 512B. The S2MM channels share access to the write ports of the AXI4 manager interface. The MM2S channels share the read ports of the AXI4 manager interface. This AXI4 manager interface is connected to the NoC NMU. There is a shared lock module that is equivalent to the one present in the AIE-ML v2 tiles. The S2MM channel consumes data coming from the incoming stream, creates AXI4 write transactions based on the address generation in the BD, and issues a burst on the write ports of the AXI4 manager interface. The MM2S channel requests AXI4 read transfers from the NoC based on the information in the DMA BDs and pushes data to stream. It handles the backpressure of the stream. Both channels have a task queue and can issue task-complete-tokens.
- **Lock Unit:** Inside the DMA there are 16 semaphore hardware locks. These locks are like the locks inside the AIE-ML v2 tile and are used by the DMA to handle synchronization for BD usage.

The following figure shows the array interface connectivity that the AIE-ML v2 array uses to communicate with other blocks in the Versal architecture. Also specified are the number of streams in the AXI4-Stream interconnect interfacing with the PL, NoC, or AIE-ML v2 memory tiles, and between the AXI4-Stream switches.

Figure 14: **AIE-ML v2 Array Interface**



Interrupt Handling

It is possible to setup interrupts to the PS and the platform management controller (PMC) triggered by events inside the AIE-ML v2 array. This section introduces the types of interrupts from the AIE-ML v2 array.

AIE-ML v2 has the capability to report hardware errors at the hypervisor level and are under privileged register control. To allow this AIE-ML v2 has additional logic to propagate these hardware errors to the array interface-tile and privileged registers for control and logging. The hardware errors are divided into three different categories.

- **Uncorrectable hardware errors:** 2-bit ECC errors and parity errors
- **Correctable hardware errors:** 1-bit ECC errors
- **AXI errors:** SLVERR/DECERR

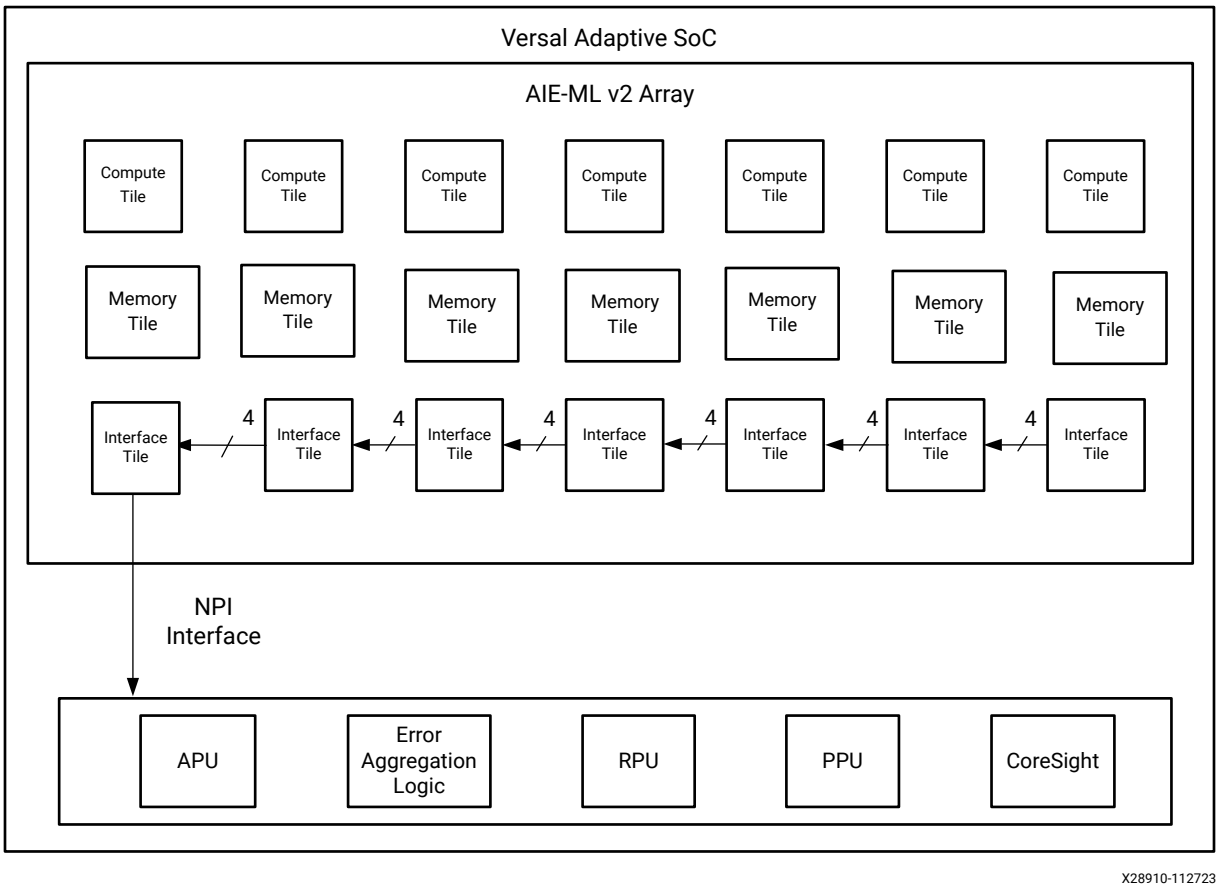
These are hardwired and cannot be configured. All the events in a category are Ored together inside each tile and then along every column. The Array interface tiles contain logic to latch, mask, clear, and generate an interrupt from hardware/AXI errors received from the column.

The AIE-ML v2 array generates four interrupts that can be routed from the AIE-ML v2 array to the PMC, application processing unit (APU), and real-time processing unit (RPU). The overall hierarchy for interrupt generation from AIE-ML v2 array is as follows:

- Events get triggered from any of the AIE-ML v2 tiles or AIE-ML v2 array interface-tiles.
- Each column has first-level interrupt handlers that can capture the trigger/event generated and forward it to the second-level interrupt handler. Second-level interrupt handler, HW-error interrupt handler in AIE-ML v2 array interface-tile drive the interrupt.
- A second-level interrupt handler can drive any one of the four interrupt lines in an AIE-ML v2 array interface-tile.
- These four interrupt lines are eventually connected to the AIE-ML v2 array interface-tile.

The following figure is a high-level block diagram showing the connections of the NPI interrupts from the AIE-ML v2 array to other blocks in the AMD Versal™ device. The diagram does not show the actual layout/placement of the array interface-tiles and the AIE-ML v2 tiles.

Figure 15: **Connecting Interrupts from the AIE-ML v2 Array to Other Functional Blocks**



As indicated in the previous diagram, four interrupts originate from an NoC interface tile. These signals traverse through the PL interface tile and ultimately arrive at a configuration interface tile. Upon reaching this point, internal errors such as the loss of PLL lock, are combined (ORed) with the incoming four interrupts. Subsequently, the resulting four interrupts are directly linked to the NPI interrupt signals on the NPI interface, which operates as a 32-bit wide memory-mapped AXI4 bus.

At the device level, the four NPI interrupts are designated as 4 to 7. There exist three distinct groups of NPI registers denoted as IMR0...IMR3, IER0...IER3, and IDR0...IDR3. Each set of registers (IMR, IER, and IDR) serves the purpose of configuring the behavior of the four NPI interrupts. The IMR registers are read-only, while the IER and IDR registers are write-only. Only the registers corresponding to NPI interrupt 4 can be programmed.

For NPI interrupts 5, 6, and 7, the three sets of registers—IMR, IER, and IDR—do not have any impact, rendering their programming ineffective. Consequently, these three interrupts cannot be controlled or masked using the NPI registers.

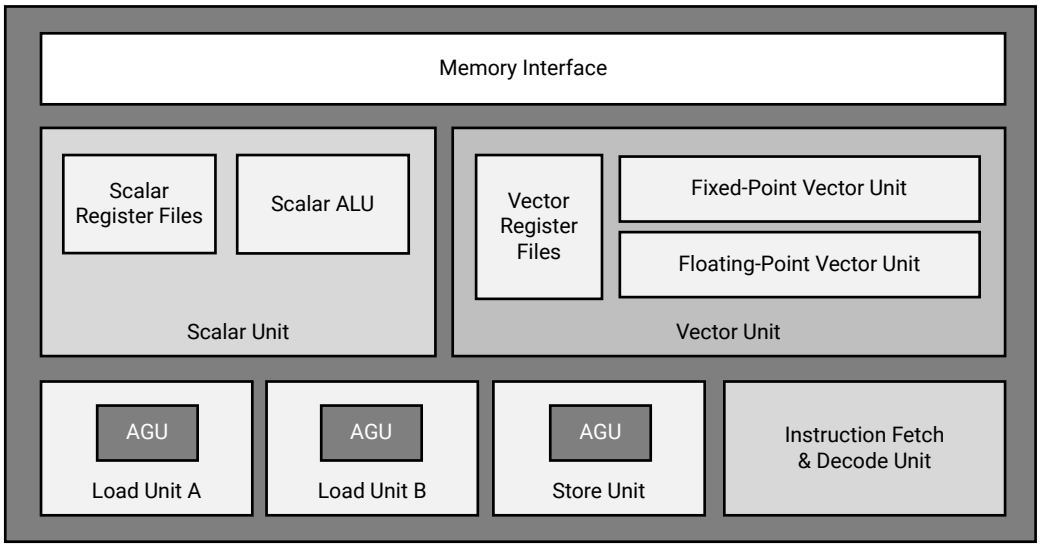
Chapter 4

AIE-ML v2 Architecture

Functional Overview

The AIE-ML v2 is a highly-optimized processor featuring single-instruction multiple-data (SIMD) and very-long instruction word (VLIW) processor that supports both fixed-point and floating-point precision. As shown in the following figure, the AIE-ML v2 has a memory interface, a scalar unit, a vector unit, two load units, one store unit, and an instruction fetch and decode unit.

Figure 16: **AIE-ML v2**



The features of the AIE-ML v2 include:

- Instruction-based VLIW SIMD processor
- 32-bit scalar RISC processor
  - Scalar register files and special registers
  - 32 x 32-bit multiplier
  - 32-bit add/subtract

  - ALU operations like shifts, compares, and logical operations.
  - Hardware acceleration for inverse, square root, and inverse square root.

- Vector Multiplication Unit
  - Vector unit supporting MAC operations for multiple precisions (for example, 512x 8-bit × 8-bit and 512x 4-bit × 4-bit)
  - Sparsity is supported for all integer and floating point modes except for MX block floating point (50% sparsity)
  - Support for floating point multiplication (float8, bfloat16, and float16) accumulating in single precision floating point (fp32). For fp8, several formats are supported, such as E4M3 and E5M2, and with different representation capabilities for infinity, NaNs and zeros.
  - Support for multiplying MX block floating point types and accumulating in floating point. The MX9 type has eight bits per block element (sign and mantissa in twos complement), eight bits for the shared exponent, and eight bits for shared sub-tile shifts. The MX6 type has five bits per block element (sign and mantissa in twos complement), eight bits of shared exponent, and eight bits of shared sub-tile shifts. The MX4 type has three bits per block element (sign and mantissa in twos complement), eight bits of shared exponent, and eight bits of shared sub-tile shifts.
  - The multiplier|multiplicand can be signed or unsigned. The accumulator is always signed.
  - The accumulation can be performed in several operation modes: 64 lanes of 32 bits, or 32 lanes of 64 bits.
  - The total number of multipliers and the number of accumulation lanes determine the depth of the post-adding.

- *Table 7:* **Supported Precision Width of the Vector Data Path**

Precision 1	Precision 2	Number of Accumulator Lanes	Bits per Accumulator Lane	Number of MACs
int8	int8	64	32	512
int16	int16	64	32	128
int16	int16	32	64	128
int32	int16	32	64	64
bfloat16	bfloat16	32	SPFP	256
float16	float16	32	SPFP	256
float8	float8	64	SPFP	512
MX6	MX6	64	SPFP	1024
MX9	MX9	64	SPFP	512

Notes:

1. Multiplication of 4-bit by 4- bit can be emulated.
2. Multiplication of 32-bit by 32-bit numbers can be emulated by decomposition into multiplications of 32x16-bit.
3. Single precision floating point (SPFP) per the IEEE standard.
4. MX4 multiplication is emulated using MX9.

- Vector Addition Unit
  - Vector unit supporting 8, 16 or 32-bit addition, subtraction, comparison and min/max computation
  - Support for non-linear functions: tanh, exp2 (bfloat16 with tanh and exp2, float16 with exp2)
  - Support processing of two 512-bit wide vectors
  - Includes comparisons and min/max computation for bfloat16/float16 vector
- Load/Store Units
  - For loading/storing data and weights
  - AGU handles optimized address generation for ML functionality
  - Two 512-bit load and one 512-bit store units with aligned addresses
  - Supports 2D/3D addressing modes for ML functionality
- Ports to Streaming interconnect switch
  - 2×32-bit subordinate port
  - 1×32-bit manager port
- Processor bus interface: The processor bus allows the AIE-ML v2 to perform direct read/write access to local tile memory mapped registers.

Register Files

The AIE-ML v2 has several types of registers. Some of the registers are used in different functional units. This section describes the various types of registers.

Scalar Registers

Scalar registers include configuration registers. See the following table for register descriptions.

Table 8: **Scalar Registers**

Syntax	Number of bits	Description
r0..r31	32 bits	General-purpose registers
m0..m7	20 bits	Modifier registers
p0..p7	20 bits	Pointer registers

Special Registers

Table 9: **Special Registers**

Syntax	Number of bits	Description
dn0..dn7	20 bits	AGU dimension size register
dj0..dj7	20 bits	AGU dimension stride (jump) register
dc0..dc7	20 bits	AGU dimension count register
s0..s3	6 bits	Shift control
sp	20 bits	Stack pointer
lr	20 bits	Link register
pc	20 bits	Program counter
fc	20 bits	Fetch counter
	32 bits	Status register[1]
	32 bits	Mode control register[1]
ls	20 bits	Loop start
le	20 bits	Loop end
lc	32 bits	Loop count
lci	32 bits	Loop count (PCU)
F	128 bits	MX6 data MSB registers
G	64 bits	MX6 sub-title shift registers
E	64 bits	MX6 exponent registers

Notes:

1. The status and control registers are each separate registers of a small number of bits each. Only through the debug interface are the various individual registers accessed together as one 32-bit wide SR and CR register.

Vector Registers

Vector registers are wide to allow SIMD instructions and to be used as operand storage. These registers are prefixed with a *W*. There are 24 x 256-bit registers: wln and whn, $n = 0..11$. Two W registers can be grouped to form a 512-bit register prefixed with an X. Two X registers then can be grouped to form a 1024-bit register with the prefix Y.

Table 10: **AIE-ML v2 Vector Registers**

256-bit	512-bit	1024-bit
wl0	x0	y0
wh0		
wl1	x1	
wh1		

Table 10: **AIE-ML v2 Vector Registers** *(cont'd)*

256-bit	512-bit	1024-bit
wl2	x2	y1
wh2		
wl3	x3	
wh3		
wl4	x4	y2
wh4		
wl5	x5	
wh5		
wl6	x6	y3
wh6		
wl7	x7	
wh7		
wl8	x8	y4
wh8		
wl9	x9	
wh9		
wl10	x10	y5
wh10		
wl11	x11	
wh11		

Mask Registers

In addition to the vector registers, there are 4 x 128-bit mask registers (Q0 to Q3) used for sparsity.

Accumulator Registers

Accumulator registers are used to store the results of the vector data path and are 512-bit wide, they can be viewed as sixteen lanes of 32-bit data or eight lanes of 64-bit data. The main reason to have wider accumulator registers is to have high precision multiplication and accumulate over those results without bit overflows. The accumulator registers are prefixed with `bm`. Two of them are aliased to form a 1024-bit register prefixed with `cm`, and two `cm` can be aliased to form a 2048-bit register prefixed with `dm`.

Table 11: **AIE-ML v2 Accumulator Registers**

512-bit	1024-bit	2048-bit
bmll0	cml0	dm0
bmlh0		
bmhl0	cmh0	
bmhh0		
...	...	...
...		
...	...	
...		
bmll7	cml7	dm7
bmlh7		
bmhl7	cmh7	
bmhh7		

Instruction Fetch and Decode Unit

The instruction fetch and decode unit sends out the current program counter (PC) register value as an address to the program memory. The program memory returns the fetched 128-bit wide instruction value. The instruction value is then decoded, and all control signals are forwarded to the functional units of the AIE-ML v2. The program memory size on the AIE-ML v2 is 16 KB.

The AIE-ML v2 instruction size ranges from 16 to 128 bits and supports multiple instruction formats and variable length instructions to reduce the program memory size. In most cases, the full 128 bits are needed when using all VLIW slots. However, for many instructions in the outer loops, main program, control code, or occasionally the pre and post-ambles of the inner loop, the shorter format instructions are sufficient.

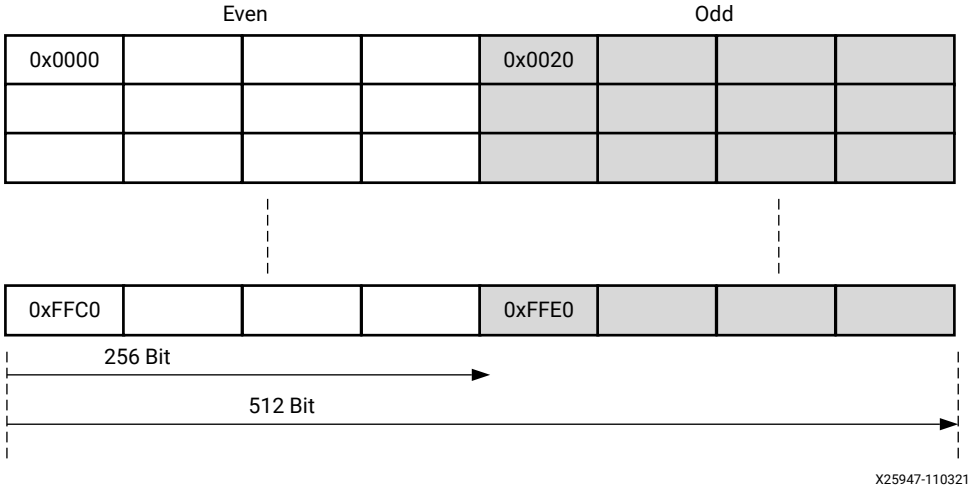
Load and Store Unit

The AIE-ML v2 has two load units and one store unit for accessing data memory. Data is loaded or stored in data memory.

Each of the load or store units has an address generation unit (AGU). AGUA and AGUB are the load units and the store unit is AGUS. Each AGU has a 20-bit input from the P-register file and a 20-bit input from the M-register file (refer to the pointer registers and the modifier registers in Register Files). The AGU has a one cycle latency.

An individual data memory block is 64 KB. The AIE-ML v2 accesses four 64 KB data memory blocks to create a 256 KB unit. These four memory blocks are located on each side of the AIE-ML v2 and are divided and interleaved as odd and even banks (see the following figure).

Figure 17: **Interleaving in Data Memory (64 KB per Block)**



In a logical representation the 256 KB memory can be viewed as one contiguous 256 KB block or four 64 KB blocks, and each block can be divided into odd and even banks. The memory can also be viewed as eight 32 KB banks (four odd and four even). The AGU generates addresses for data memory access that span from `0x0000` to `0x3FFFF` (256 KB).

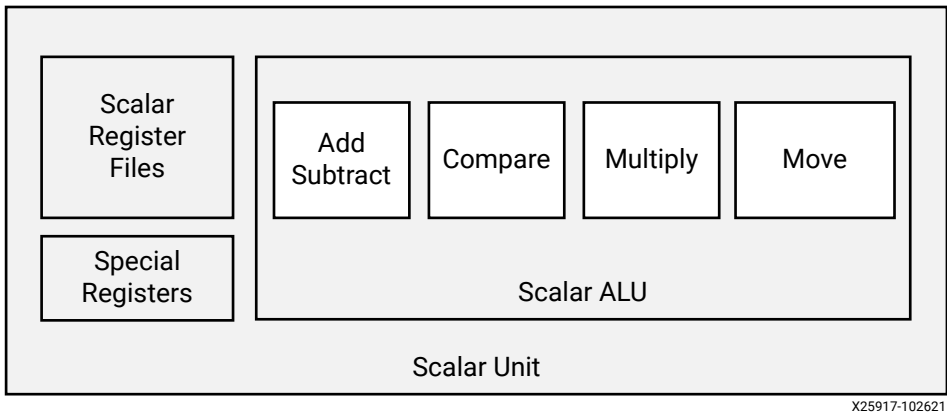
Scalar Unit

The following figure shows a block diagram of the scalar unit, including the scalar register files and scalar functional units. The scalar unit contains the following functional blocks:

- Register files and special registers
- Scalar arithmetic and logical unit (ALU)

Integer add, subtract, compare, and shift functions are one-cycle operations. The integer multiplication operation has a two-cycle latency.

Figure 18: **AIE-ML v2 Scalar Unit**



Arithmetic Logic Unit and Scalar Functions

The arithmetic logic unit (ALU) in the AIE-ML v2 manages the following operations. In all cases the issue rate is one instruction per cycle:

- Integer addition and subtraction: 32 bits. The operation has a one cycle latency.
- Bit-wise logical operation on 32-bit integer numbers (BAND, BOR, BXOR). The operation has a one cycle latency.
- Integer multiplication: 32 x 32 bit with output result of 32 bits stored in the R register file. The operation has a two cycle latency.
- Shift operation: Both left and right shift are supported. A positive shift amount is used for left shift and a negative shift amount is used for right shift. The shift amount is passed through a general purpose register. A one bit operand to the shift operation indicates whether a positive or negative shift is required. The operation has a one-cycle latency.
- Hardware acceleration for inverse, square root, and inverse square root, with support for integer and floating-point values at both input and output.

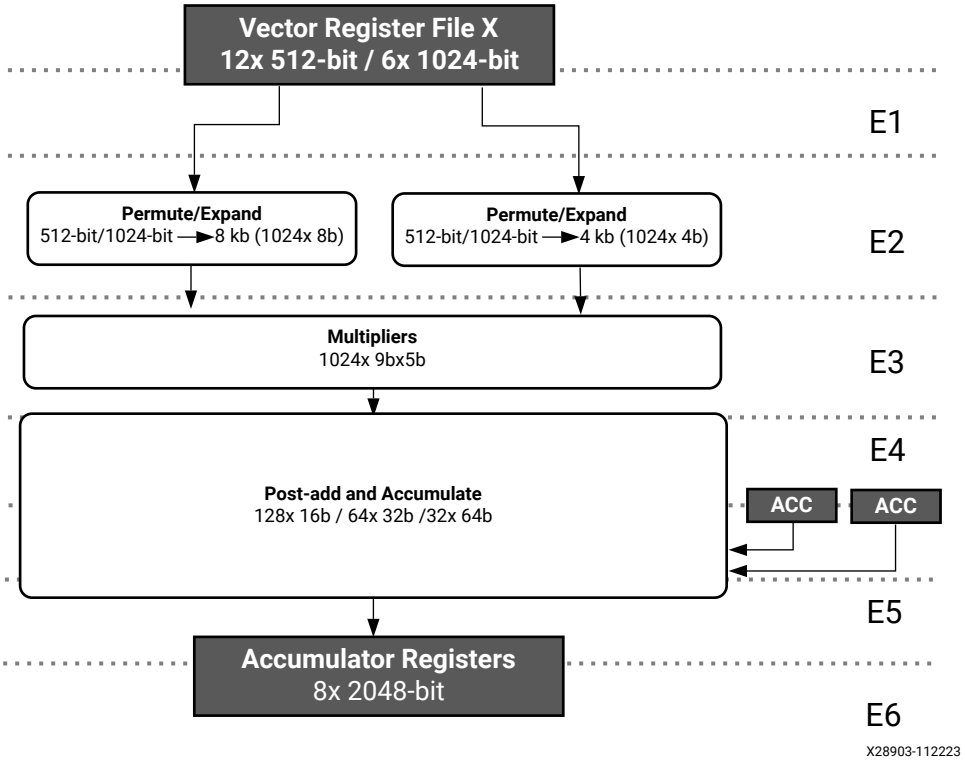
There is no floating point arithmetic unit in the scalar unit. The floating point operations are supported through emulation. In general, it is preferred to perform add and multiply in the vector unit.

Vector Unit

Fixed-point Vector Unit

This is the data path for integer vector data. The multiplication data path is split into six pipeline stages as shown in the following diagram.

Figure 19: **Pipeline Diagram of AIE-ML v2 Fixed-Point Vector Multiplication Paths**

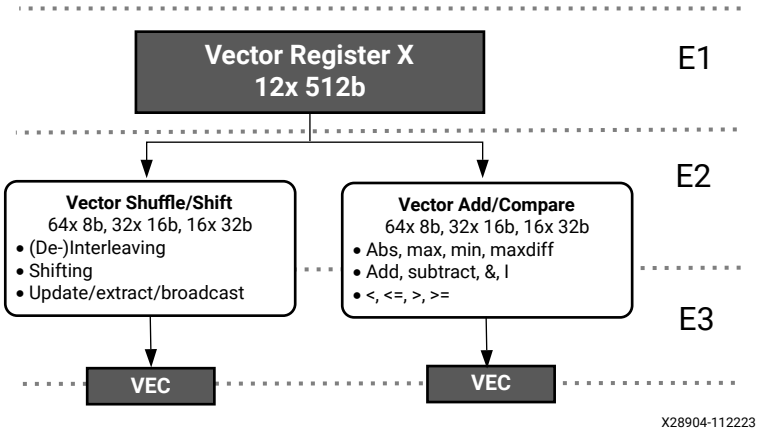


The features of the units in the multiplication datapath are as follows:

- There are two permute units that handle a set of permutes of X vector registers.
- The multiplier unit is fed by the output of the permute blocks.
- The post-add and accumulate unit adds the result of the multipliers with up to two accumulator inputs.
-
  - 16 lanes of 32-bit
  - 32 lanes of 16-bit
  - 64 lanes of 8-bit

In addition to the multiplier datapath, there are two additional vector units: shuffle/shift and add/compare. The input comes directly from two vector registers and the results are stored back in the vector registers.

Figure 20: **Pipeline Diagram of AIE-ML v2 Fixed-point Vector Unit Shuffle/Shift and Adder Paths**



The vector add/compare units support the following bit-width modes (both signed and unsigned):

- 16 lanes of 32-bit
- 32 lanes of 16-bit
- 64 lanes of 8-bit

The vector add/compare unit supports lane-by-lane control whether addition or subtraction is performed. This unit also has limited support for 16-bit and 8-bit floating point formats. This includes the following modes in a lane-by-lane fashion:

- $x < y$, $x > y$, $x \leq y$, $x \geq y$, $x < 0$
- $\min(x,y)$, $\max(x,y)$

The vector shift unit takes one or two 512-bit vector registers as an input and produces one 512-bit output vector. It supports the following modes:

- Standard right shift with 8-bit granularity
- Shift and push in scalar value either at the left or right-hand side. An 8, 16, or 32-bit lane can be shifted into the LSB lane of a 512-bit vector register, and all existing values are shifted one lane up. The value of MSB lane is dropped.

The shuffle unit allows different modes to transform the input vectors. It supports the following features:

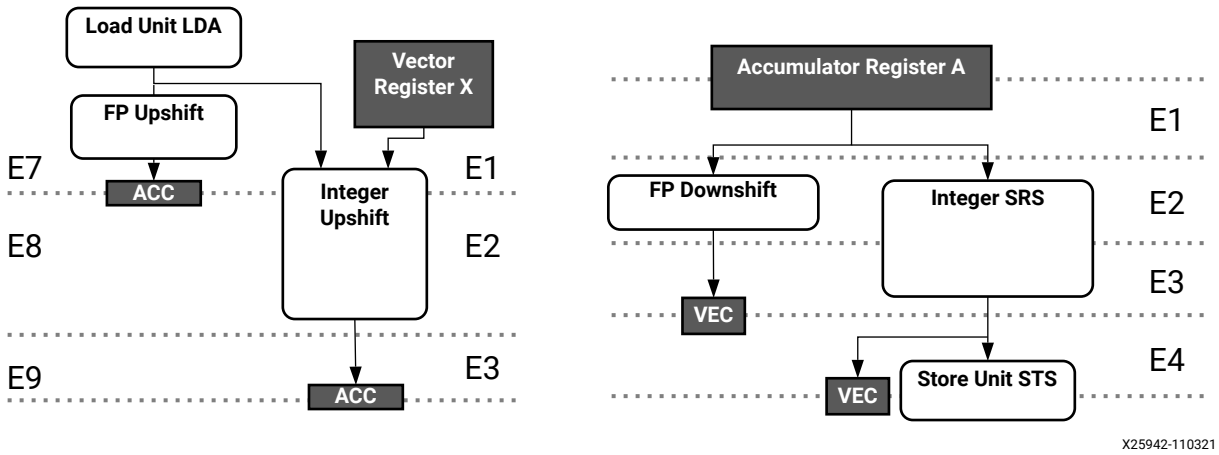
- Interleaving and de-interleaving of values at 8-bit, 16-bit, and 32-bit

- Extraction of upper and lower half of the transformed input.
- The shuffle unit shuffles the exponents and sub-tile shifts from MX block floating-point data.

Fixed-Point SRS and UPS Conversions

A block diagram of the units is shown in the following figure.

Figure 21: **Fixed-point SRS and UPS Datapath**



The SRS unit reads an accumulator register, performs the conversion, and restores the result either back to the vector register or directly to memory. The UPS unit reads a vector register directly from memory or a register and stores the result into an accumulator register. The supported modes include:

- 8-bit to/from 32-bit conversion
- 16-bit to/from 32-bit conversion
- 16-bit to/from 64-bit conversion
- 32-bit to/from 64-bit conversion

The core supports a floating-point conversion mode which converts bfloat16/float16/bfloat8/float8 to a single precision or the other way around. The following modes are supported:

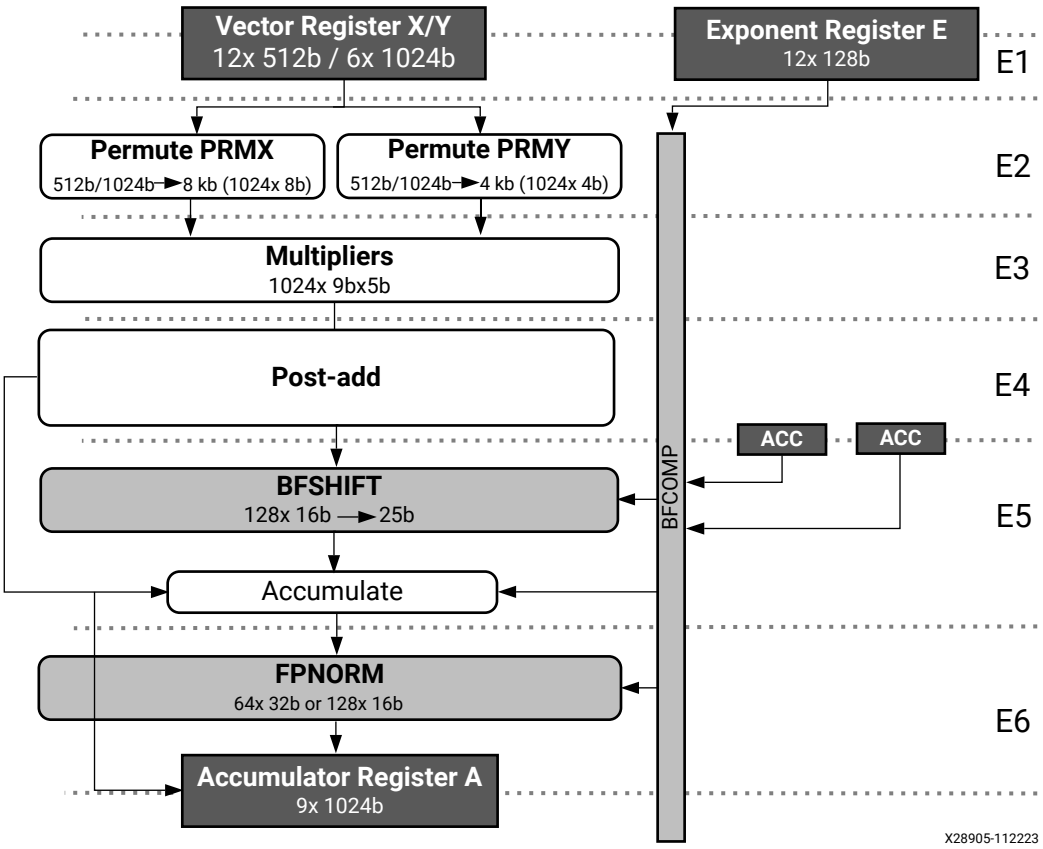
- Floating-point to floating-point conversions:
  - 16/32 lanes of fp32 accumulators to bfloat16/float16 vector registers
  - 16/32 lanes of bfloat16/float16 vector registers to fp32 accumulators
  - 32 lanes of fp32 accumulators to bfloat8/float8 vector registers
- Floating-point to integer conversions: 16 lanes of bfloat16 vector registers to 32-bit signed integers

- Block Floating-point to floating-point conversions:
  - A block of MX9 data converts to single precision floating-point accumulators
  - A block of MX6 data converts to single precision floating-point accumulators
  - A block of MX4 data converts to single precision floating-point accumulators
- Floating-point to MX conversions:
  - A block of single precision floating-point accumulators converts to a block of MX9 with 16 8-bit mantissas with a shared exponent and sub-tile shift values.
  - A block of single precision floating-point accumulators converts to a block of MX6 with 16 5-bit mantissas with a shared exponent and sub-tile shift values.
  - A block of single precision floating-point accumulators converts to a block of MX4 with 16 3-bit mantissas, a shared exponent and sub-tile shift values.

Floating-point Vector Unit

The floating-point vector data path is split in six pipeline stages. The mantissas from the MX and floating-point datablocks are pre-processed in the same fashion as the integer data path through the permute units. The MX exponents come from the E register file and are used in the alignment stage to prepare for addition and normalization using the fp32 accumulation data path. The MX sub-tile shift values come from the G register file.

Figure 22: **AIE-MLv2 Floating-Point Vector Datapath**



The MX and floating-point shift unit shifts down the post-add output and the two accumulator lanes. The accumulator unit supports addition/subtract/negate of accumulator registers in single-precision FP32 format. All floating-point additions are done in one go, by aligning all mantissas to the one with the largest exponent and with 23 bits of fractional bits. The floating-point normalization unit converts the accumulation result to FP32.

Register Move Functionality

The register move capabilities of the AIE-ML v2 are covered in this section (refer to the Register Files section for a description of the naming of register types.

- **Scalar to scalar:**
  - Move scalar values between R, M, P, and special registers.
  - Move immediate values to R, M, P, and special registers.
  - Move a scalar value to/from an AXI4-Stream.

- **Vector to vector:**

  Move one 256-bit W-register to an arbitrary W-register in one cycle. It also applies to the 512-bit X-register and the 1024-bit Y-register. However, vector sizes must be the same in all cases.

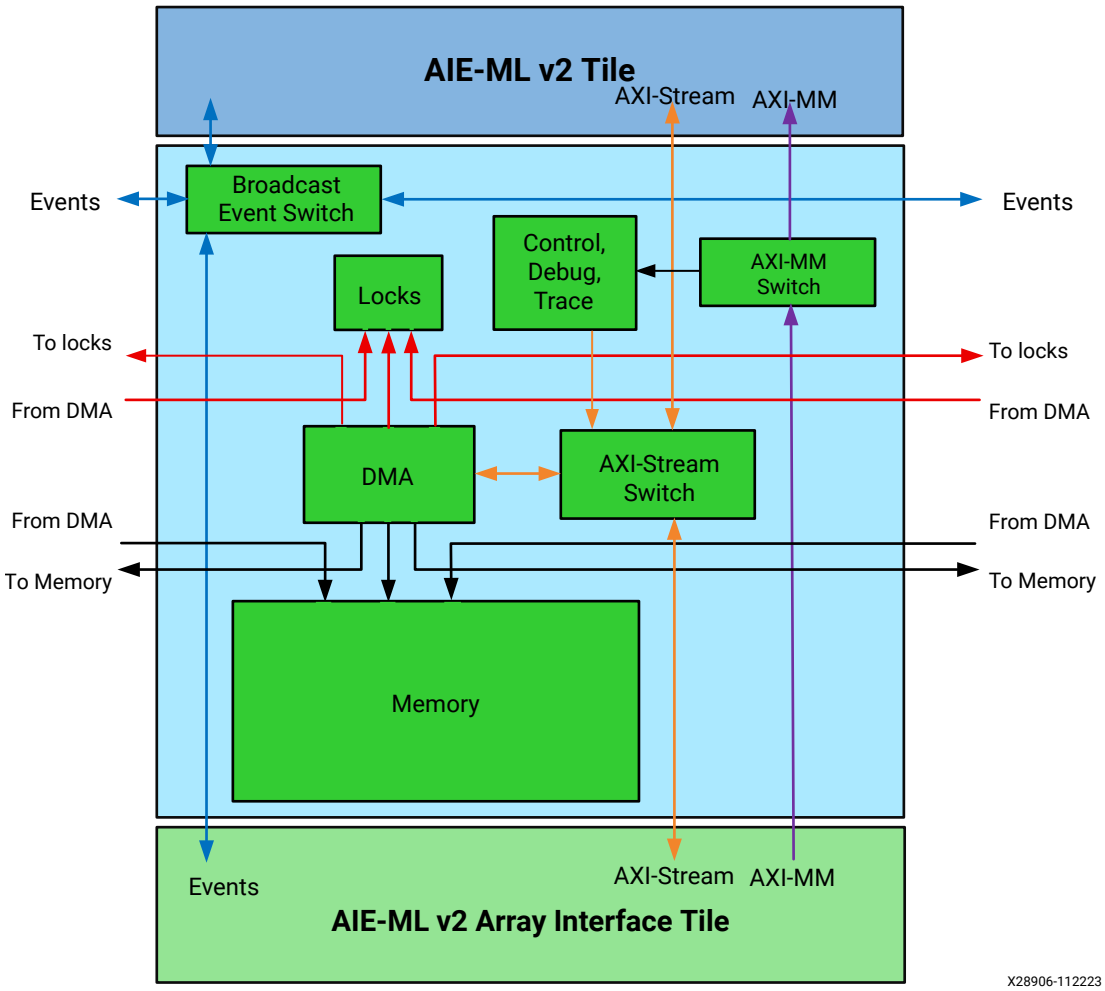
- **Accumulator to accumulator:** Move one 512-bit accumulator (BM) register to another BM-register in one cycle. There is also register CM to CM accumulator register move (1024 bits).
- **Vector to accumulator:** There are three possibilities:
  - Up shift path takes 16 or 32-bit vector values and writes into an accumulator.
  - Use the normal multiplication datapath and multiply each value by a constant value of 1.
  - Move between BM and X registers.
- **Accumulator to vector:** Shift-round saturate datapath moves the accumulator to a vector register. There is also a direct register move from accumulator to vector register.
- **Accumulator to cascade stream and cascade to accumulator:** Cascade stream connects the AIE-ML v2 in the array in a chain and allows the AIE-ML v2 to transfer an accumulator register (512-bit) from one to the next. A small two-deep 512-bit wide FIFO on both the input and output streams allows storing up to four values in the FIFOs between the AIE-ML v2s.
- **Scalar to vector:** Moves a scalar value from an R-register to a vector register.
- **Vector to scalar:** Extracts an arbitrary 8, 16, or 32-bit value from a 512-bit vector register and writes results into a scalar R-register.

Chapter 5

AIE-ML v2 Memory Tile Overview and Features

The AIE-ML v2 architecture incorporates memory tiles that provide substantial on-chip storage within the array, improving data locality and bandwidth while reducing the utilization of memory resources in the PL. The AIE-ML v2 memory tile is similar to the AIE-ML v2 tile but without the core and program memory. Instead, it features a larger (512 KB) high-density and high-bandwidth data memory, together with an integrated DMA. All DMA channels can access the local memory and a subset of them can directly access the memory in the nearest neighboring memory tiles to the east and west. A subset of DMA channels can directly access the memory in the nearest neighboring memory tiles to the east and west.

Figure 23: **AIE-ML v2 Memory Tile Architecture**



The AIE-ML v2 memory tile has the following functional blocks. They are either the same, or very similar to the equivalent blocks in AIE-ML v2 tile:

- Memory
- DMA
- Locks
- Stream switch
- AXI4 switch
- Control, debug, and trace
- Events and event broadcast

The following are the features of the AIE-ML v2 memory tile:

- Memory
  - 512 KB of memory arranged in eight physical banks. Each bank is 256-bit wide and 2K words deep.
  - ECC protected
- AIE-ML v2 memory tile DMA
- Six memory to stream (MM2S) DMA channels, each with:
  - 64-bit stream interface
  - 256-bit memory interface
  - Support for 4D tensor address generation
  - Support for inserting constant value into the stream data called constant padding.
  - Access memory and locks in the neighboring tiles (Channels 0 – 3)
  - Support task queue and task-complete-tokens with task repeat-count
  - Compression of zeros in data before sending on stream
  - Store incremental address offset between BD calls
  - TLAST suppress feature.
- Six memory to stream (MM2S) DMA channels, each with:
  - 64-bit stream interface
  - 256-bit memory interface
  - Support for 4D tensor address generation
  - Access memory and locks in the neighboring tiles (Channels 0 – 3)
  - Support task queue and task-complete-tokens with task repeat-count
  - Decompression of stream data to re-insert compressed zeroes before storing to memory.
  - Store incremental address offset between BD calls
  - Support for out of order packet transfer
  - Support for Finish on TLAST
- Buffer descriptors (BD): 48 shared buffer descriptors across all 12 DMA channels
- Stream switch:
  - Shares the same design as AIE-ML v2 tile with 17 manager and 18 subordinate ports.
  - The data path has 64-bit throughput which supports two 32-bit words in parallel per cycle.
  - `TKEEP` signal added to support transfer of an odd number of 32-bit-words.

    - North and south ports and DMA can run at full throughput of 64-bit per cycle.
    - Trace and control ports have 32-bit interface and adapters are placed to convert them to 64-bit.
- Lock module:
    - Accessible from local and neighboring AIE-ML v2 memory tile DMA channels
    - There are 64 semaphore locks and each lock state is 6-bit unsigned.
- Additional control and status registers:
    - Events, event actions, event broadcast, and combo events
    - Task-complete-tokens logic
- Configuration/debug interconnect (AXI4)
    - 1 MB address space per tile
    - Stream control-packet support
- Debug and trace
    - Event trace streams
    - Six performance counters and a 64-bit tile timer

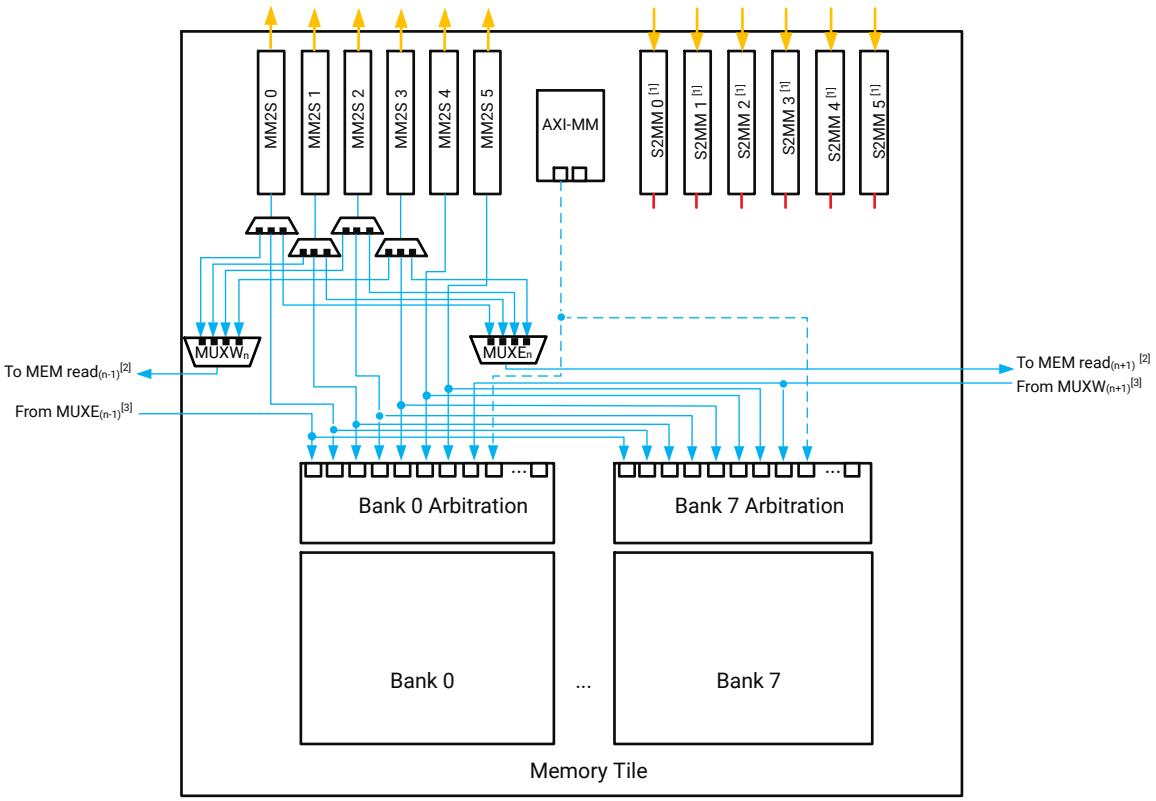
AIE-ML v2 Memory Tile Memory

Each memory tile has 512 KB of memory formed from eight banks of 64 KB. Each bank is 256-bit wide and 2K words deep. Each bank can be accessed by ten read and ten write interfaces.

The memory tile read interfaces are:

- AXI4 read (including control packets)
- One MM2S interface from the MM2S channels in the neighboring memory tile to the left
- 6x local MM2S channels [0 – 5]
- One MM2S interface from the MM2S channels in the neighboring memory tile to the right
- ECC scrubber

Figure 24: **Memory Tile Memory Read Interfaces**



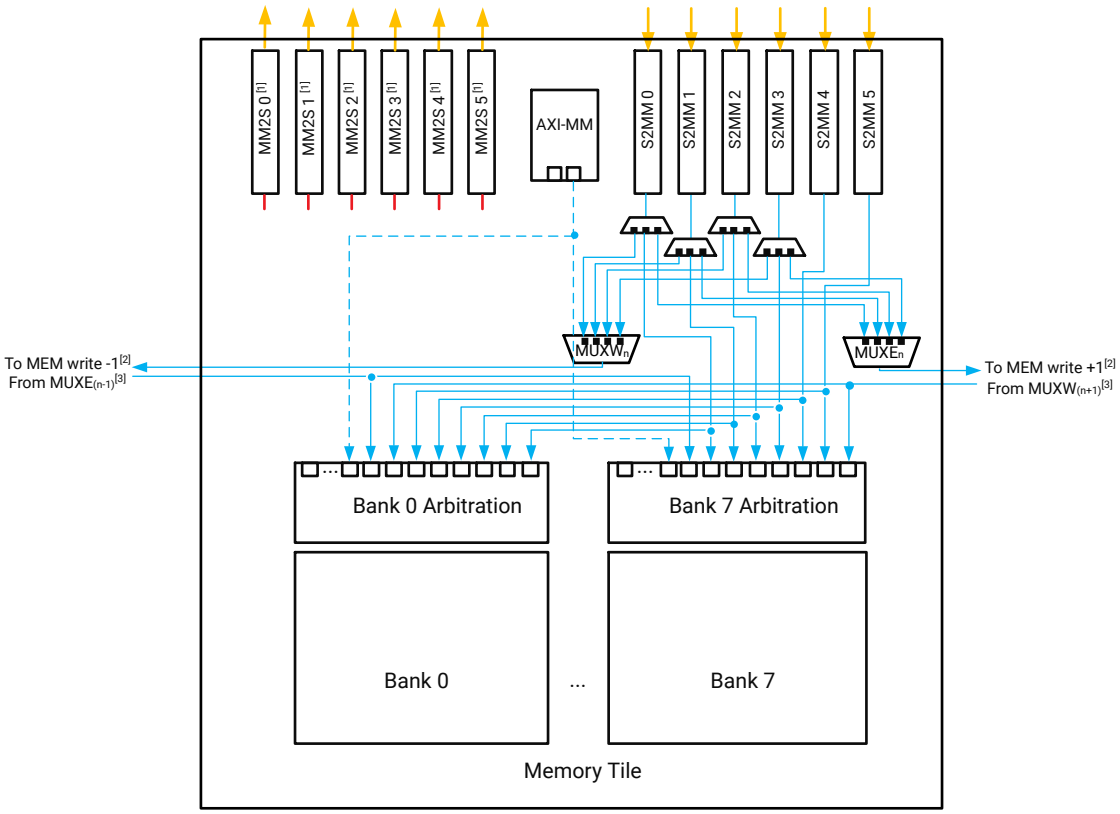
[1] The S2MM interfaces exist for the memory tile, but are not used in the memory read operation.
[2] MEM read$_{(n-1)}$ and MEM read$_{(n+1)}$ correspond to the neighboring tile memory banks on the east and west.
[3] MUXE$_{(n-1)}$ refers to the multiplexer of the tile on the east and MUXW$_{(n+1)}$ refers to the multiplexer of the tile on the west.

X25865-101921

The memory tile write interfaces are:

- AXI4 write (including control packets)
- One MM2S interface from the MM2S channels in the neighboring memory tile to the left
- 6x local S2MM channels [0 – 5]
- One MM2S interface from the MM2S channels in the neighboring memory tile to the right
- ECC scrubber/zeroization

Figure 25: **Memory Tile Memory Write Interfaces**



[1] The MM2S interfaces exist for the memory tile, but are not used in the memory write operation.
[2] MEM write -1 and MEM write +1 correspond to the neighboring tiles on the east and west.
[3] $MUXE_{(n-1)}$ refers to the multiplexer of the tile on the east and $MUXW_{(n+1)}$ refers to the multiplexer of the tile on the west.

X25880-101921

DMA S2MM channels (0 – 3) and MM2S channels (0 – 3) can access the local memory banks and the memory banks of the Memory-tile to the east and the west. The memory in memory tile supports bank interleaving. Interleaving is done at 256-bit granularity, such that sequential 256-bit accesses map to different banks and wrap around after the eight banks (every 256B). The memory tile banks have ECC protection and ECC scrubbing.

AIE-ML v2 Memory Tile DMA

The list of features in the AIE-ML v2 memory tile DMA is covered in Chapter 5: AIE-ML v2 Memory Tile Overview and Features. The memory tile DMA is similar to the AI Engine AIE-ML v2 tile DMA with the following enhancements:

- Supports 5D tensor address generation (including iteration-offset)

- Allows out-of-order buffer descriptor (BD) processing based on incoming packet header information
- Supports compression and decompression

The memory tile DMA has 12 independent channels: six S2MM and six MM2S. Each channel has an input task queue. It can load a BD, generate address, access memory over a shared interface, and read or write to and from its stream port. Each channel can also trigger the issuing of a task-complete-token upon completing a task. Each S2MM channel can be configured to work in either in-order or out-of-order mode.

DMA S2MM channels 0 – 3 and MM2S channels 0 – 3 can access the memory banks in the tile to the west and east, in addition to the local memory banks. These same channels can also access lock modules in tile to the east and west. Both MM2S and S2MM channels 4 – 5 can only access local memory banks and local lock modules. All 12 channels use the same address scheme and lock indexes, as shown in the following table.

Table 12: **Address and Lock Ranges for Memory Tile DMAs**

	Address Ranges	Lock Indexes	Description
West	0x0_0000 – 0x7_FFFF	0 – 63	Channels 0 – 3 only
Local	0x8_0000 – 0xF_FFFF	64 – 127	
East	0x10_0000 – 0x17_FFFF	128 – 191	Channels 0 – 3 only

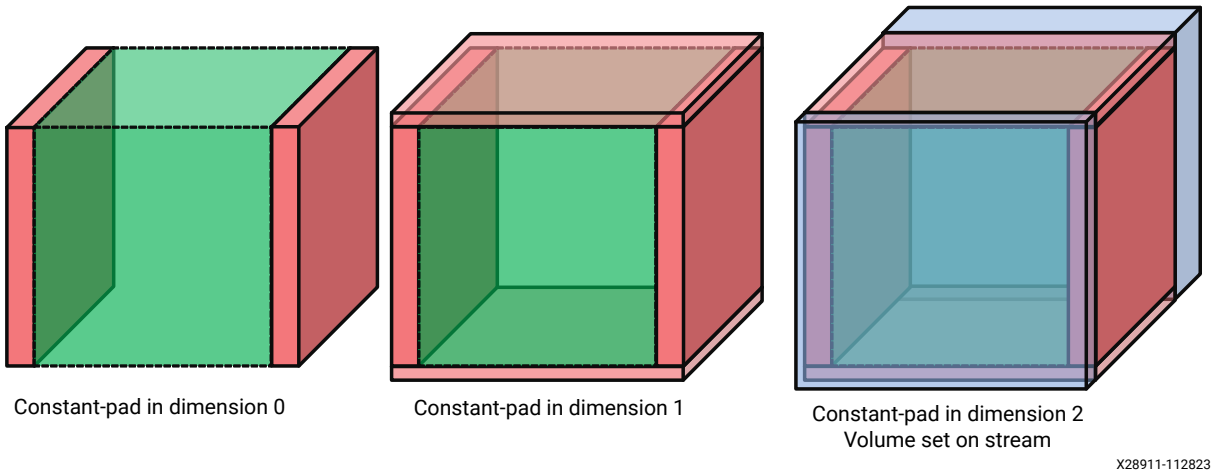
With this addressing scheme, it is possible to configure the hardware where the address and lock requests could be out of range for a specific DMA channel. This condition can result in the DMA channel stalling requiring a channel reset to proceed.

The memory tile MM2S channels support constant-padding insertion into the stream data. There is a 32-bit register per MM2S channel which can be set to any value, this value is inserted by hardware during padding. This feature satisfies two application requirements:

- Algorithmic padding: To recreate surrounding data on the edge of valid data.
- Granularity padding: An optimized kernel can operate on 16 channels while a layer can have 24 channels, the MM2S channel pads the channel dimension up to 32 channels.

The constant-padding insertion is linked to 4D address generation. For the lower three dimensions there is a field for padding before and after that dimension. The following figure illustrates constant padding in three dimensions. The padding on a dimension is added on top of the wrap for that dimension.

Figure 26: **Constant Padding in 3D**



AIE-ML v2 Memory Tile Locks Module and Stream Switch

The memory tile semaphore locks are identical to those in the AIE-ML v2 tile. The memory tile supports up to 64 semaphore locks each carrying a 6-bit unsigned lock state. All the locks are accessible from local DMA channels and from S2MM and MM2S channels 0 – 3 from both east and west neighboring memory tiles. Locks are also accessible through memory mapped AXI4.

The memory tile stream switch is similar to the tile stream switch with 17 manager ports and 18 subordinate ports. Unlike the tile stream switch, this design has no east or west stream ports.

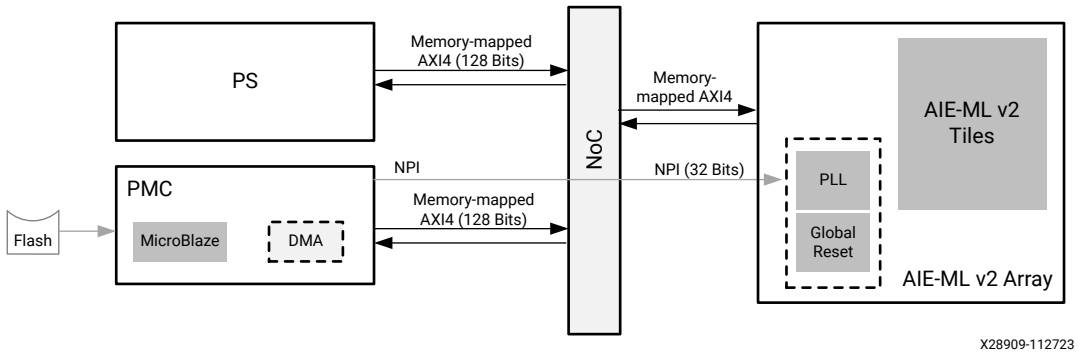
Chapter 6

AIE-ML v2 Configuration and Boot

AIE-ML v2 Array Configuration

There are two top-level scenarios in the AIE-ML v2 array configuration: AIE-ML v2 array configuration from power-up and AIE-ML v2 array partial reconfiguration. The following figure shows a high-level view of the AIE-ML v2 array and configuration interface along with the registers to the PS and the platform management controller (PMC) through the NoC.

Figure 27: **AIE-ML v2 Array Configuration using NoC and NPI**



Any memory-mapped AXI4 manager can configure any memory-mapped AXI4 register in the AIE-ML v2 array using the NoC (for example, the PS and PMC). The global registers (including PLL configuration, global reset, and security bits) in the array configuration interface tile can be programmed using the NPI interface because the global registers are mapped onto the NPI address space.

AIE-ML v2 Boot Sequence

This section describes the steps involved in the boot process for the AIE-ML v2 array.

- The column clock enable value is disabled by default.

- Memory zeroization hardware logic is added that applies to each of the tile program memory, tile data memory, memory tile data memory and control module memory. For each of the memories there is a 1-bit memory-mapped AXI4 register that is set to 1 when zeroization starts. When the process is completed, the internal hardware sets this bit to 0. For the control module memory it is a 3-bit memory mapped register that is set to 0'b111.

1. Power-on and power-on-reset (POR) deassertion: Power is turned on for all modules related to the AIE-ML v2 array, including the PLL. After power-on, the PLL runs at a default speed. The platform management controller (PMC) and NoC need to be up and running before the AIE-ML v2 boot sequence is initiated. After the array power is turned on, the PMC can deassert a POR signal in the AIE-ML v2 array.
2. AIE-ML v2 array configuration using NPI: After power-on, the PMC uses the NPI interface to program the different global registers in the AIE-ML v2 array (for example, the PLL configuration registers). The AIE-ML v2 configuration image that is required over the NPI for AIE-ML v2 array initialization comes from a flash device.
3. Enable PLL: After the PLL registers are configured (after POR), the PLL-enable bit can be enabled to turn on the PLL. The PLL then settles on the programmed frequency and asserts the `LOCK` signal. The source of the PLL input (`ref_clk`) is from `hsm_ref_clk` and is generated in the control interfaces and processing system (CIPS).
4. Column clock and column reset assertion/de-assertion: After the PLL is locked, all column clocks are enabled by writing a 1 to a memory-mapped AXI4 register bit. All column resets are then asserted writing a `1` to a memory-mapped AXI4 register bit. After waiting for a number of cycles, all column resets are de-asserted by writing a 0 to the same register bit.
5. The control module should be setup.
6. The array is partitioned into one or more independent partitions, with an integer number of AIE-ML v2 columns per partition. Isolation is enabled by default in all tiles, therefore must be disabled on the internal edges of each partition.
7. AIE-ML v2 array programming: The AIE-ML v2 array interface needs to be configured over the memory-mapped AXI4 from the NoC interface. This includes all program memories, AXI4 stream switches, DMAs, event, and trace configuration registers.

AIE-ML v2 Array Reconfiguration

The AIE-ML v2 configuration process writes a programmable device image (PDI) produced by the bootgen tool into AIE-ML v2 configuration registers. The AIE-ML v2 configuration is done over memory-mapped AXI4 via the NoC. For more information on generating a PDI with the bootgen tool, refer to *Bootgen User Guide* (UG1283).

The following table summarizes the reset controls available for the global AIE-ML v2 array.

Table 13: **Categories of AIE-ML v2 Resets**

Type	Trigger	Scope
Internal power-on-reset	Part of boot sequence	AIE-ML v2 array
System reset	NPI input	AIE-ML v2 array
INITSTATE reset	PCSR bit	AIE-ML v2 array
Array soft reset	Software register write over NPI	AIE-ML v2 array
AIE-ML v2 tile column reset	Memory-mapped AIE-ML register bit in the array interface tile	AIE-ML v2 tile column
AIE-ML v2 core reset	Memory-mapped AIE-ML v2 register bit in the core tile	
AIE-ML v2 array interface reset	From NPI register	AIE-ML v2 array interface tile

The combination of column reset and array interface tile reset (refer to AIE-ML v2 Array Hierarchy) enables a partial reconfiguration use case where a sub-array that comprises AIE-ML v2 tiles and array interface tiles can be reset and reprogrammed without disturbing adjacent sub-arrays. The specifics of handling the array splitting and adding isolation depend on the type of use case (multi-user/tenancy or single-user/tenancy multiple-tasks).

Appendix A

Comparison of AIE Generations

Table 14: **Comparison of AIE Generations**

	AI Engine	AIE-ML	AIE-ML v2
Compute			
Multipliers per Tile (non-exhaustive list)	128 (INT8xINT8) 8 (fp32)	256 (INT8xINT8) 128 (bfloat16)	512 (INT8xINT8) 256 (bfloat16, fp16) 512 (fp8) 512 (MX9) 1024 (MX6)
Bandwidth to Local Memory			
Load from data memory in same Tile	64 B/cycle	64 B/cycle	128 B/cycle
Load from data memory in neighboring Tiles	64 B/cycle	64 B/cycle	64 B/cycle from a single neighbor 128 B/cycle from two neighbors
Store	32 B/cycle	32 B/cycle	64 B/cycle
Cascade bandwidth	48 B/cycle	64 B/cycle	64 B/cycle
On-chip Data Memory			
Per Compute Tile	32 KB 128 B/cycle (8 x 16B banks)	64 KB 128 B/cycle (8 x 16B banks)	64 KB 256 B/cycle (8 x 32B banks)
Per Memory Tile	No memory tiles	512 KB 256 B/cycle (16 x 16B banks)	512 KB 256 B/cycle (16 x 32B banks)
DMAs			
Compute Tile	2 MM2S 2 S2MM	2 MM2S 2 S2MM	2 MM2S 2 S2MM
Memory Tile	No memory tiles	6 MM2S 6 S2MM	6 MM2S 6 S2MM
Interface Tile	2 MM2S 2 S2MM	2 MM2S 2 S2MM	2 MM2S 2 S2MM

Table 14: **Comparison of AIE Generations** *(cont'd)*

	AI Engine	AIE-ML	AIE-ML v2
Stream Interconnect			
	32b	32b	64b
Compute Tile	South to North: 6 (24 B/cycle) North to South: 4 (16 B/cycle) East to West: 4 (16 B/cycle) West to East: 4 (16 B/cycle)	South to North: 6 (24 B/cycle) North to South: 4 (16 B/cycle) East to West: 4 (16 B/cycle) West to East: 4 (16 B/cycle)	South to North: 6 (48 B/cycle) North to South: 4 (32 B/cycle) East to West: 4 (32 B/cycle) West to East: 4 (32 B/cycle)
Memory Tile	No memory tiles	South to North: 6 (24 B/cycle) North to South: 4 (16 B/cycle) East to West: 0 West to East: 0	South to North: 6 (48 B/cycle) North to South: 4 (32 B/cycle) East to West: 0 West to East: 0
Interface Tile	From South: 8 (32 B/cycle) To South: 6 (24 B/cycle) From North: 4 (16 B/cycle) To North: 6 (24 B/cycle) East to West: 4 (16 B/cycle) West to East: 4 (16 B/cycle)	From South: 8 (32 B/cycle) To South: 6 (24 B/cycle) From North: 4 (16 B/cycle) To North: 6 (24 B/cycle) East to West: 4 (16 B/cycle) West to East: 4 (16 B/cycle)	From South: 8 (64 B/cycle) To South: 6 (48 B/cycle) From North: 4 (32 B/cycle) To North: 6 (48 B/cycle) East to West: 4 (32 B/cycle) West to East: 4 (32 B/cycle)
PL/Array Interface	From PL: 8 (64B/cycle @ PL clock) To PL: 6 (48 B/cycle @ PL clock)	From PL: 8 (64B/cycle @ PL clock) To PL: 6 (48 B/cycle @ PL clock)	From PL: 8 (64B/cycle @ PL clock) To PL: 6 (48 B/cycle @ PL clock)
Includes Control Module?			
	No	No	Yes

Appendix B

Additional Resources and Legal Notices

Finding Additional Documentation

Technical Information Portal

The AMD Technical Information Portal is an online tool that provides robust search and navigation for documentation using your web browser. To access the Technical Information Portal, go to https://docs.amd.com.

Documentation Navigator

Documentation Navigator (DocNav) is an installed tool that provides access to AMD Adaptive Computing documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the AMD Vivado™ IDE, select **Help → Documentation and Tutorials**.
- On Windows, click the **Start** button and select **Xilinx Design Tools → DocNav**.
- At the Linux command prompt, enter `docnav`.

***Note*:** For more information on DocNav, refer to the *Documentation Navigator User Guide* (UG968).

Design Hubs

AMD Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In DocNav, click the **Design Hubs View** tab.
- Go to the Design Hubs web page.

Support Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Support.

References

These documents provide supplemental material useful with this guide:

1. *Versal AI Edge Series Gen 2 and Prime Series Gen 2 Technical Reference Manual* (AM026)
2. *Versal Architecture and Product Data Sheet: Overview* (DS950)
3. *Versal AI Edge Series Gen 2 Data Sheet: DC and AC Switching Characteristics* (DS1021)
4. *Bootgen User Guide* (UG1283)
5. *Versal Adaptive SoC AI Engine Architecture Manual* (AM009)
6. *Versal Adaptive SoC AIE-ML Architecture Manual* (AM020)
7. *AI Engine Tools and Flows User Guide* (UG1076)
8. AI Engine Technology

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
10/23/2025 Version 1.1	
Performance	Added a section and table.
Chapter 2: AIE-ML v2 Tile Architecture	Updated the section.
Memory-mapped AXI4 Interconnect	Updated the section.
AXI4-Stream Interconnect	Updated the section.
AIE-ML v2 Memory Module	Updated the Tile DMA Controller bullet.
Trace	Updated the section.
AIE-ML v2 Array Interface	Updated the first table note.
Functional Overview	Updated sparsity bullet in the Vector Multiplication Unit section.
Chapter 5	Updated the first paragraph.
Appendix A: Comparison of AIE Generations	Updated the AIE-ML v2 entry for Bandwidth to Local Memory: Load from data memory in neighboring Tiles.

Section	Revision Summary
07/24/2025 Version 1.0.2	
General updates	Editorial updates only. No technical content updates.
06/11/2025 Version 1.0.1	
Introduction to Versal Adaptive SoCs	Updated the section.
05/30/2025 Version 1.0	
Initial release.	N/A

Please Read: Important Legal Notices

Copyright