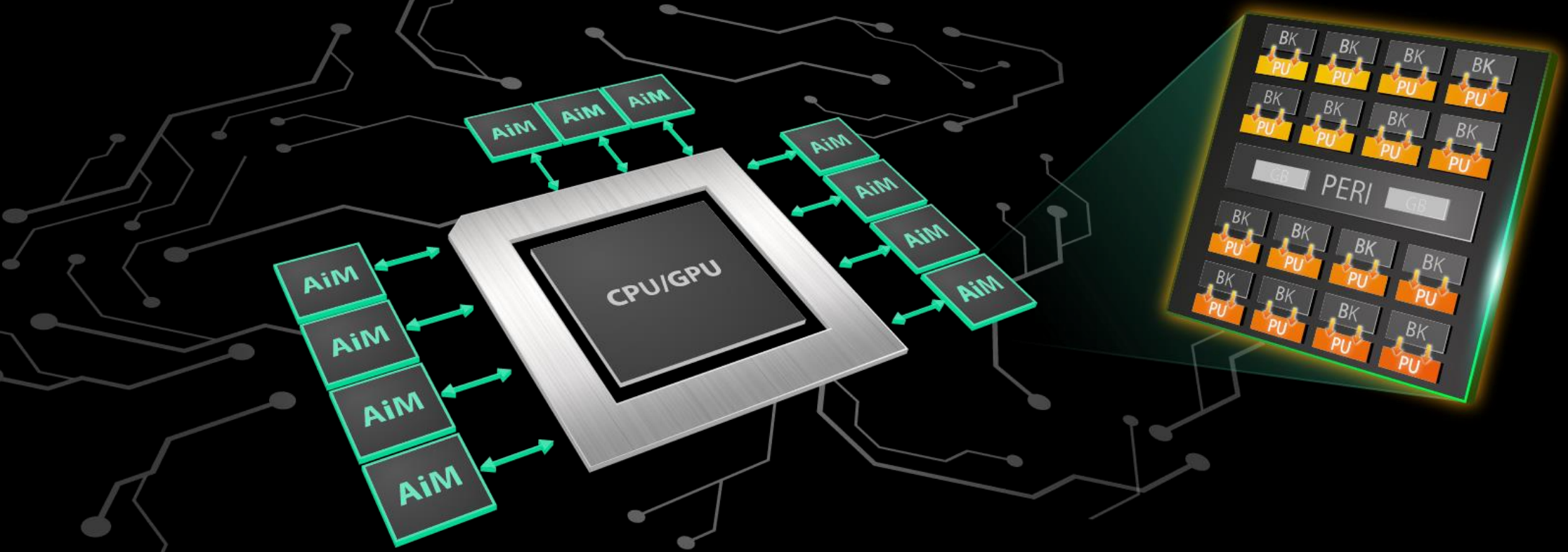
System Architecture and Software Stack for GDDR6-AiM



Yongkee Kwon, Kornijcuk Vladimir, Nahsung Kim, Woojae Shin, Jongsoon Won, Minkyu Lee, Hyunha Joo, Haerang Choi, Guhyun Kim, Byeongju An, Jeongbin Kim, Jaewook Lee, Ilkon Kim, Jaehan Park, Chanwook Park, Yosub Song, Byeongsu Yang, Hyungdeok Lee, Seho Kim, Daehan Kwon, Seongju Lee, Kyuyoung Kim, Sanghoon Oh, Joonhong Park, Gimoon Hong, Dongyoon Ka, Kyudong Hwang, Jeongje Park, Kyeongpil Kang, Jungyeon Kim, Junyeol Jeon, Myeongjun Lee, Minyoung Shin, Minhwan Shin, Jaekyung Cha, Changson Jung, Kijoon Chang, Chunseok Jeong, Euicheol Lim, Il Park, and Junhyun Chun, SK hynix

Text Abstract

This poster presents system architecture, software stack, and performance analysis for SK hynix's very first GDDR6-based processing-in-memory (PIM) product sample, called **Accelerator-in-Memory** (**AiM**).

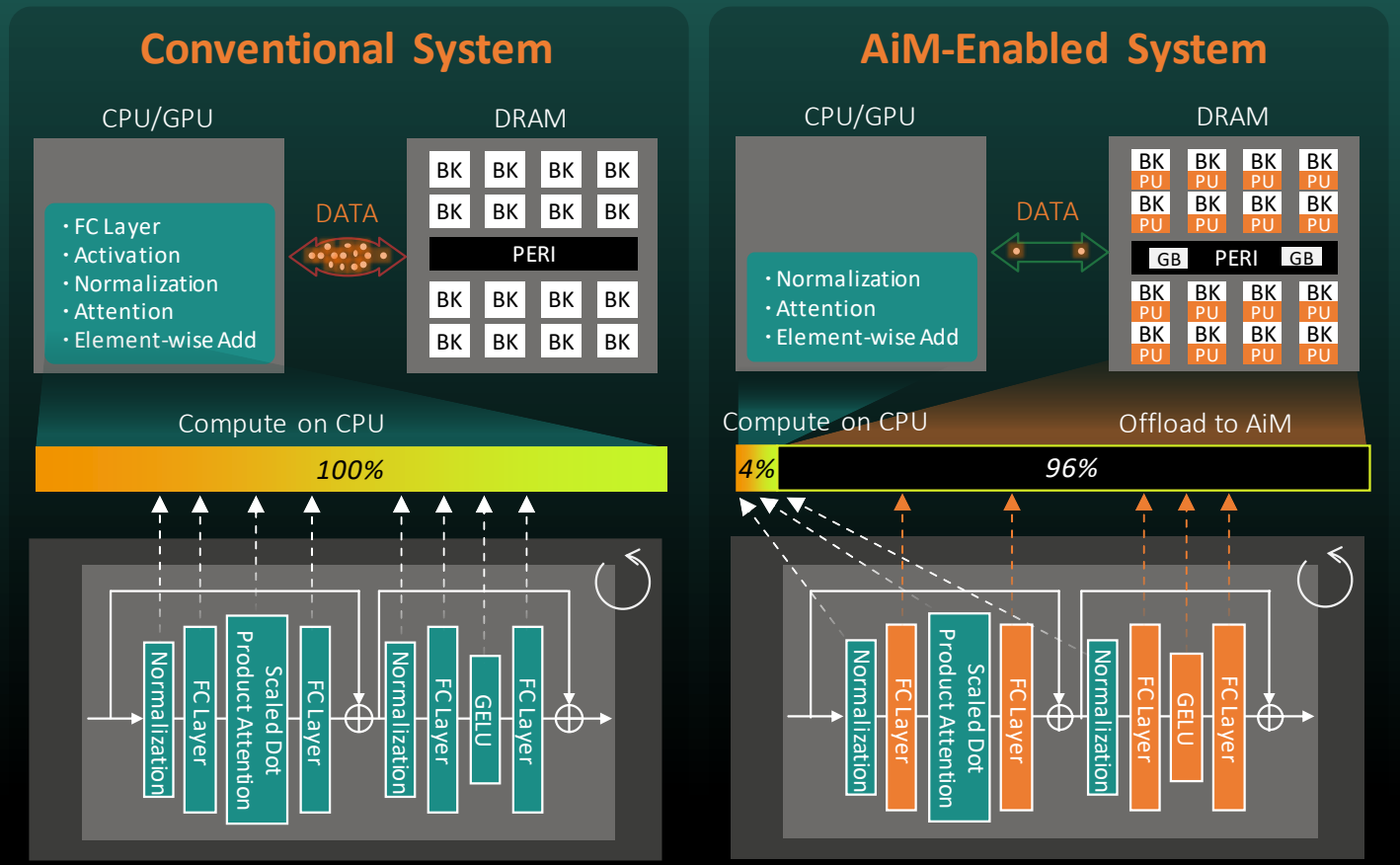
AiM is designed for the in-memory acceleration of matrix-vector product operations, which are commonly found in machine learning applications. The strength of AiM primarily comes from the two design factors, which are 1) **all-bank operation support** and 2) **extended DRAM command set**. All-bank operations allow AiM to fully utilize the abundant internal DRAM bandwidth, which makes it an attractive solution for memory-bound applications. The extended command set allows the host to address these new operations efficiently and provides a clean separation of concerns between the AiM architecture and its software stack design.

We present a dedicated **FPGA-based reference platform** with a software stack, which is used to validate AiM design and evaluate its system-level performance. We also demonstrate **FMC-based AiM extension cards** that are compatible with the off-the-shelf FPGA boards and serve as an open research platform allowing potential collaborators and academic institutes to access our hardware and software systems.

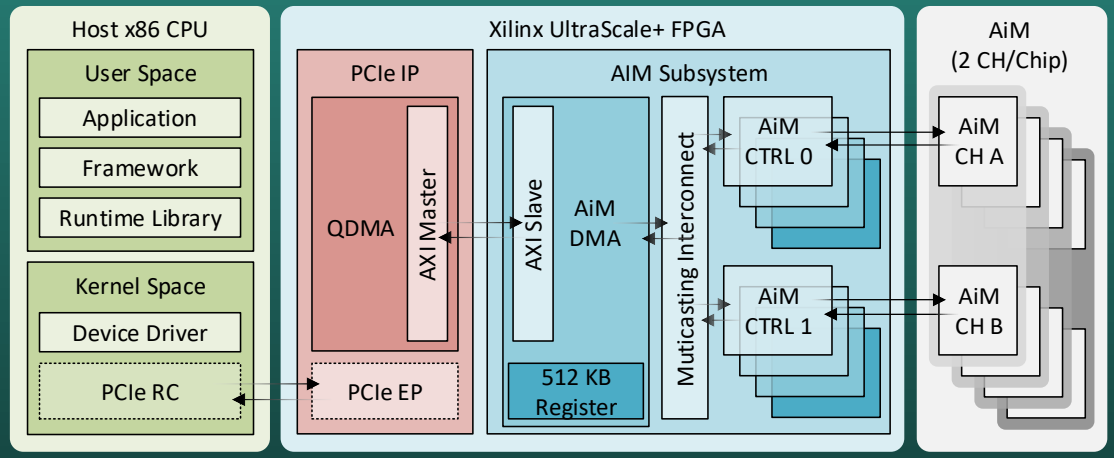
AiM Concept

The Accelerator-in-Memory (AiM) is a GDDR6-based Processing-in-Memory device designed to accelerate memory-intensive Machine Learning applications in memory.

Conventional System vs. AiM System

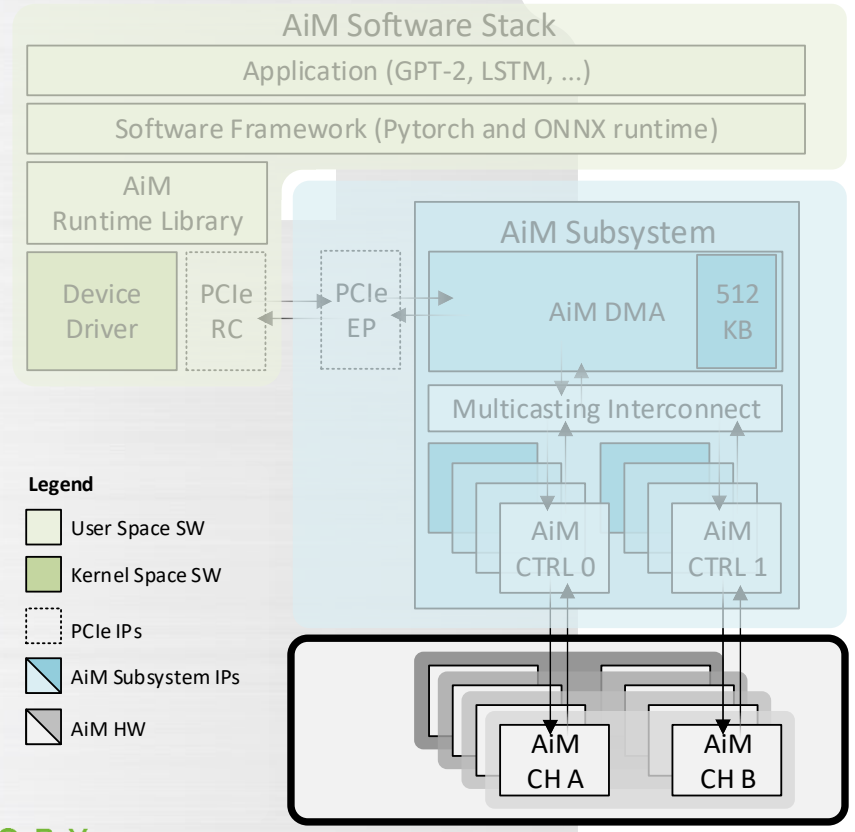


AiM Subsystem and Software Stack



AiM FPGA Platform (w/ CPU Host)

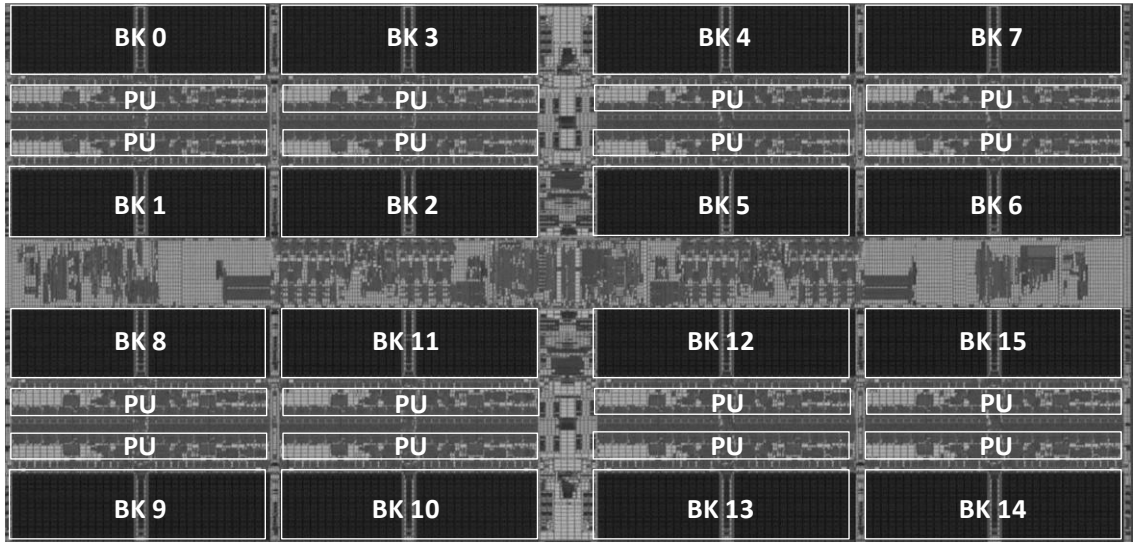
CONTENTS



GDDR6-AiM Feature Summary

SK hynix's very first GDDR6-based processing-in-memory (PIM) product sample, called Accelerator-in-Memory (AiM)

GDDR6-AiM*	
DRAM Type	GDDR6
Process Technology	1y
Memory Density	8Gb (4Gb DDP)
Organization	2CH/Chip, x16 mode only
IO Data rate	16 Gb/s/pin (@1.25V)
Bandwidth	64 GB/s
Processing Unit (PU)	16 PU/die, 32 PU/Chip
Operating Speed	1 GHz
Compute Throughput	1TFLOPS/Chip
Numeric Precision	Brain Floating Point 16 (BF16)
Activation Function support**	Sigmoid, tanh, GELU, ReLU, Leaky ReLU, …
Targets	Memory-bound DNN applications



[GDDR6-AiM Floorplan]

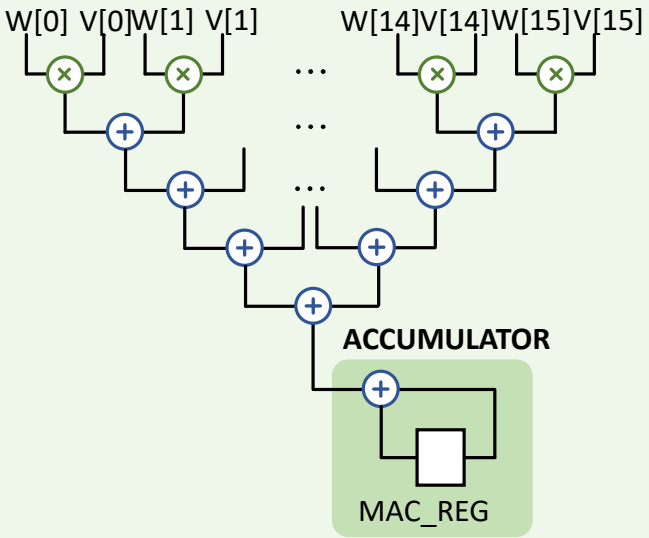
* S. Lee, et al. "A 1ynm 1.25V 8Gb, 16Gb/s/pin GDDR6-based Accelerator-in-Memory supporting 1TFLOPS MAC Operation and Various Activation Functions for Deep-Learning Applications", 2022 INTERNATIONAL SOLID-STATE CIRCUITS CONFERENCE (ISSCC). IEEE, 2022

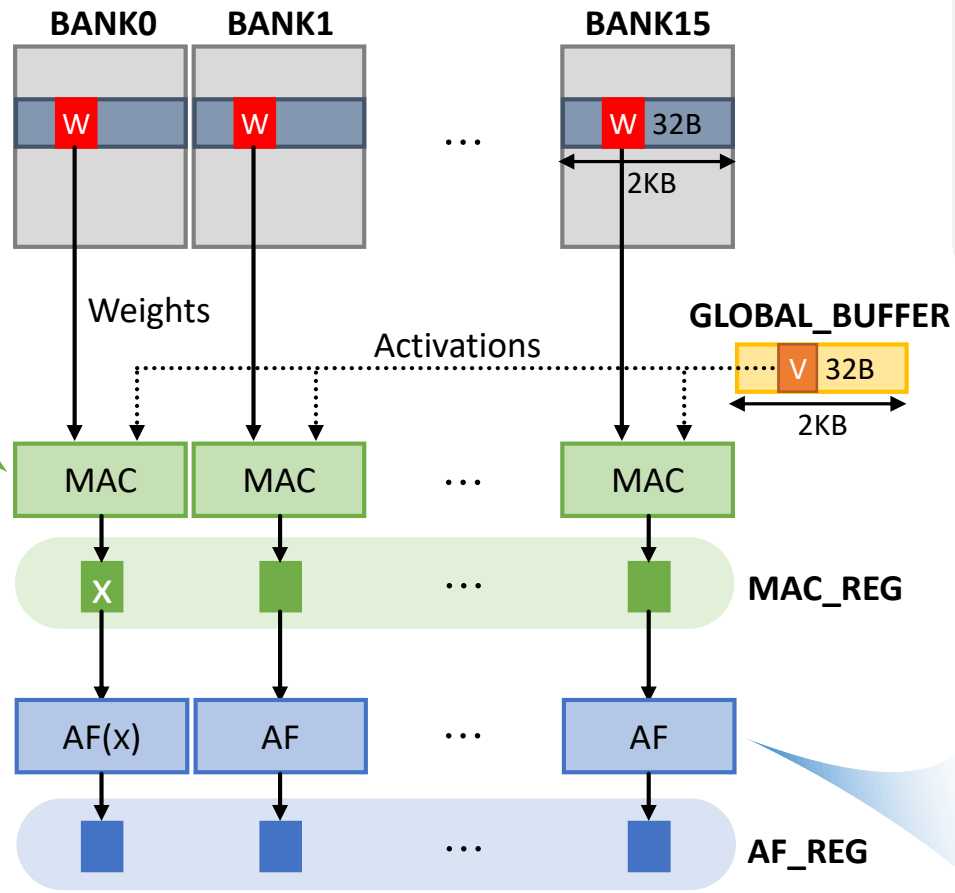
** With using internal lookup table and linear interpolation unit. Any customized function may apply with accuracy limitation.

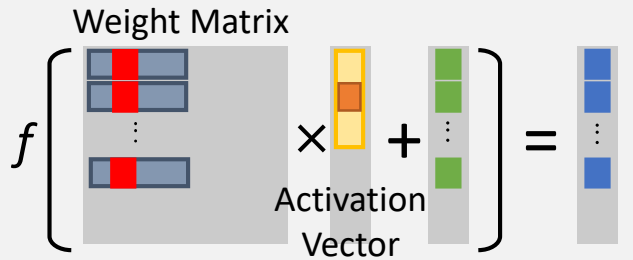
GDDR6-AiM Key Operation: Matrix × Vector

Multiply-And-Accumulate (MAC)

- Performs MAC operation on **sixteen** BF16 weight matrix and vector elements (corresponds to a single DRAM column access, i.e. 32B).
- Computation results are stored in a dedicated **MAC_REG** set and can be later accessed by the user.

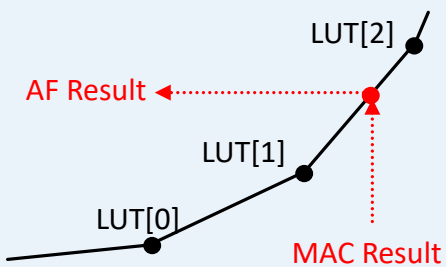






Activation Function Module

- Performs **Activation Function (AF)** computation by linearly interpolating pre-stored AF template data using MAC calculation results.
- Interpolation results are stored in a dedicated **AF_REG** set and can be later accessed by the user.



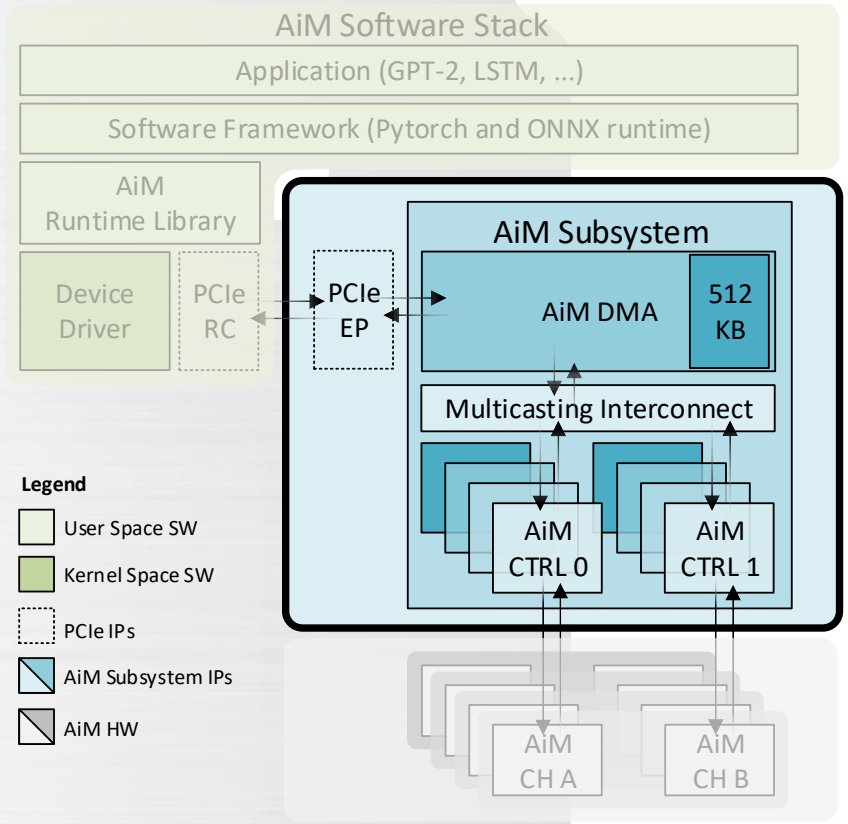
- **MAC** and **Activation Function** operations can be performed in all banks in parallel.
- **Weight** matrix data is sourced from **Banks**; **Vector** data is sourced from the **Global Buffer**.
- **MAC** results are stored in latches collectively referred to as **MAC_REG**.
- **Activation Function** results are stored in latches collectively referred to as **AF_REG**.

New Commands introduced in AiM

Bank Activation	
ACT4, ACT16	Activate four/sixteen banks in parallel
ACTAF4, ACTAF16	Activate rows storing Activation Functions LUTs in four/sixteen banks in parallel
Compute Commands	
MACSB, MAC4B, MACAB	Perform MAC in one/four/sixteen banks in parallel
AF	Compute Activation Function in all banks
EWMUL	Perform element-wise multiplication
Data Commands	
RDCP	Copy data from a bank to the Global Buffer
WRCP	Copy data from the Global Buffer to a bank
WRGB*	Write to Global Buffer *(often Activation vector data)*
RDMAC*	Read from MAC result register
RDAF*	Read from Activation Function result register
WRMAC*	Write to MAC result register *(or WRBIAS as often BIAS data is written)*
WRBK	Write to all activated banks in parallel

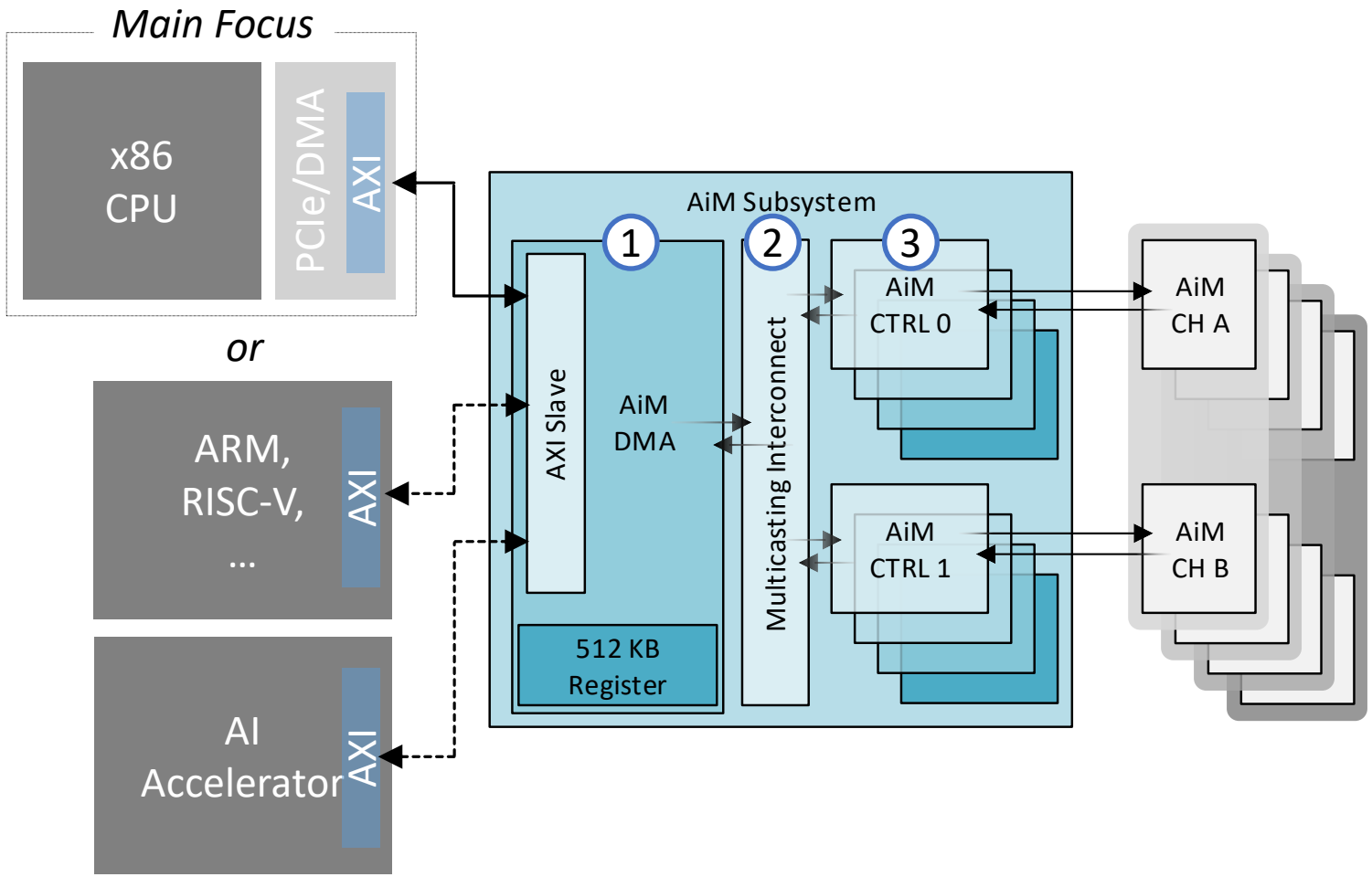
- Commands marked as **CMD*** require a special **Mode Register** set to be recognized.

CONTENTS



AiM Subsystem: High-Performance Portable Reference Design

AiM Subsystem is a hardware bridge between the host and the AiM devices. It is designed to 1) maximize compute throughput for a set of AiM devices and to 2) minimize software stack overhead.

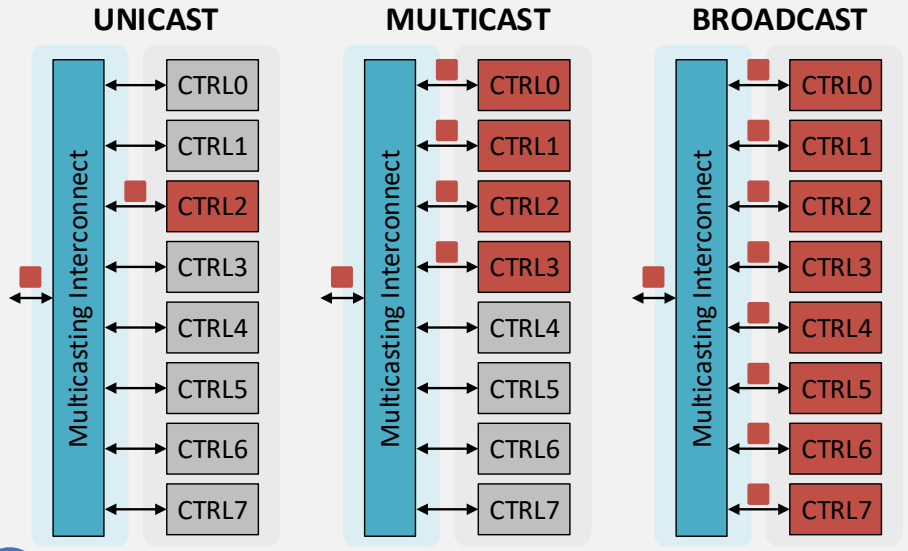


① AiM Command/Data DMA

Decodes AiM instructions from software and provides direct memory access for the host.

② AiM Multicasting Interconnect

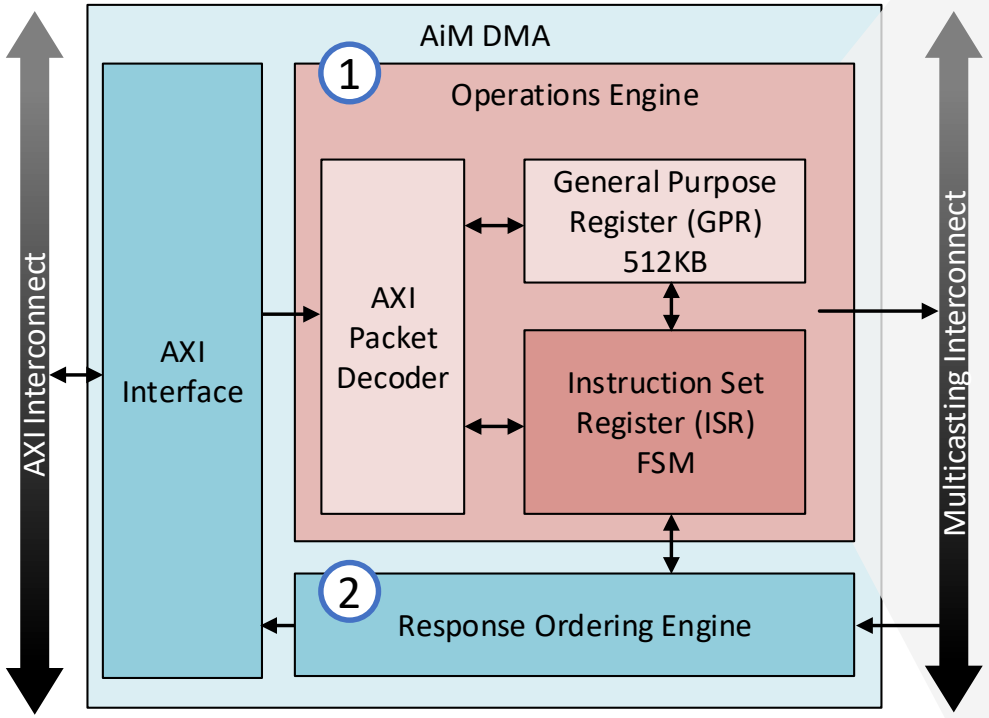
Enables efficient workload distribution through flexible instruction parallelism. Supports unicast, multicast, and broadcast modes.



③ AiM Controller

Generates and schedules low-level AiM and typical DRAM commands.

AiM Command/Data DMA



DRAM Operations

- DRAM_WRITE
- DRAM_READ

GPR Operations

- GPR_WRITE
- GPR_READ

ISR Operations

- ISR_WRITE_SBK
- ISR_WRITE_GB
- ISR_WRITE_MAC
- ISR_WRITE_AFLUT
- ISR_READ_MAC
- ISR_READ_AF
- ISR_COPY
- ISR_MAC_SBK
- ISR_MAC_ABK
- ISR_AF
- ISR_EWMUL

1 Operations Engine

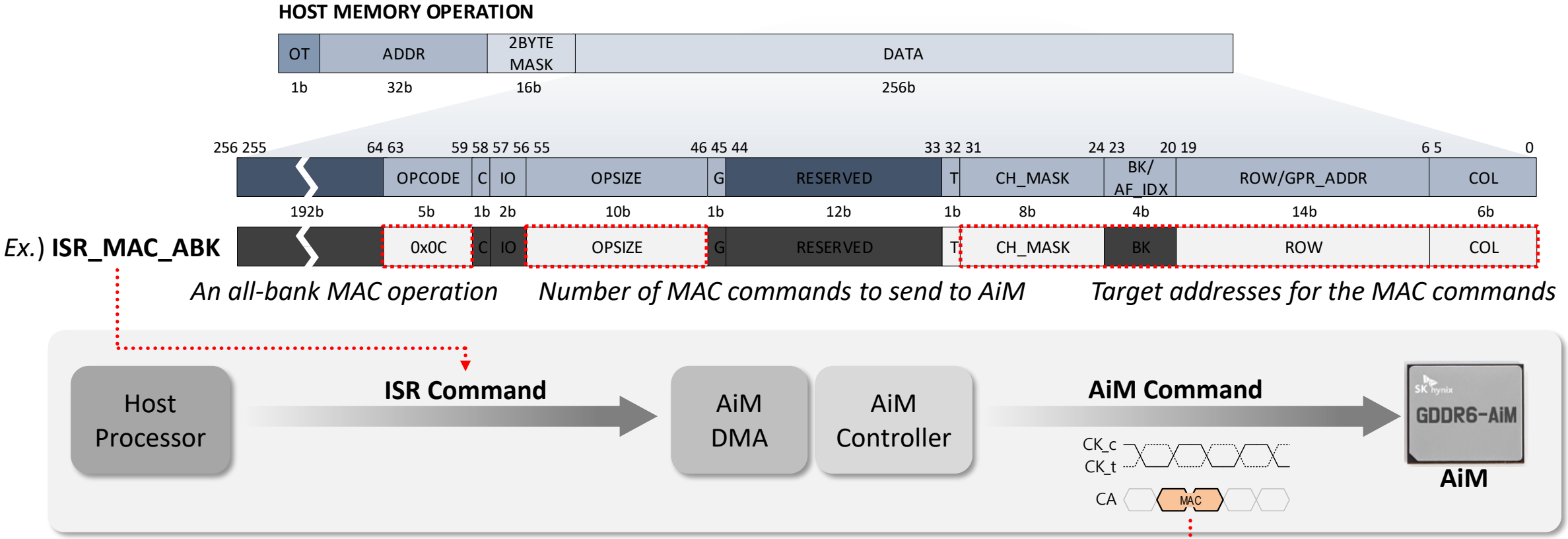
Interprets AXI operations from the host and accordingly generates either memory access or AiM compute requests. Memory requests are passed to the memory directly, while AiM requests are additionally processed by the dedicated FSM called ISR.

- **AXI Packet Decoder** performs host (software) packet interpretation.
- **General Purpose Register (GPR)** is an SRAM used for 1) holding bias data to be rewritten multiple times during GEMV computation, 2) holding activation results to be used as vectors for the following layers, 3) holding diagnostic and debugging data, e.g. temperature.
- **Instruction Set Register (ISR) FSM** generates streams of low-level AiM requests based on the higher-level software commands. See next slide for an example.

2 Response Ordering Engine

An optional block used for restoring the order of DRAM read transactions, which can be reordered by the AiM controllers.

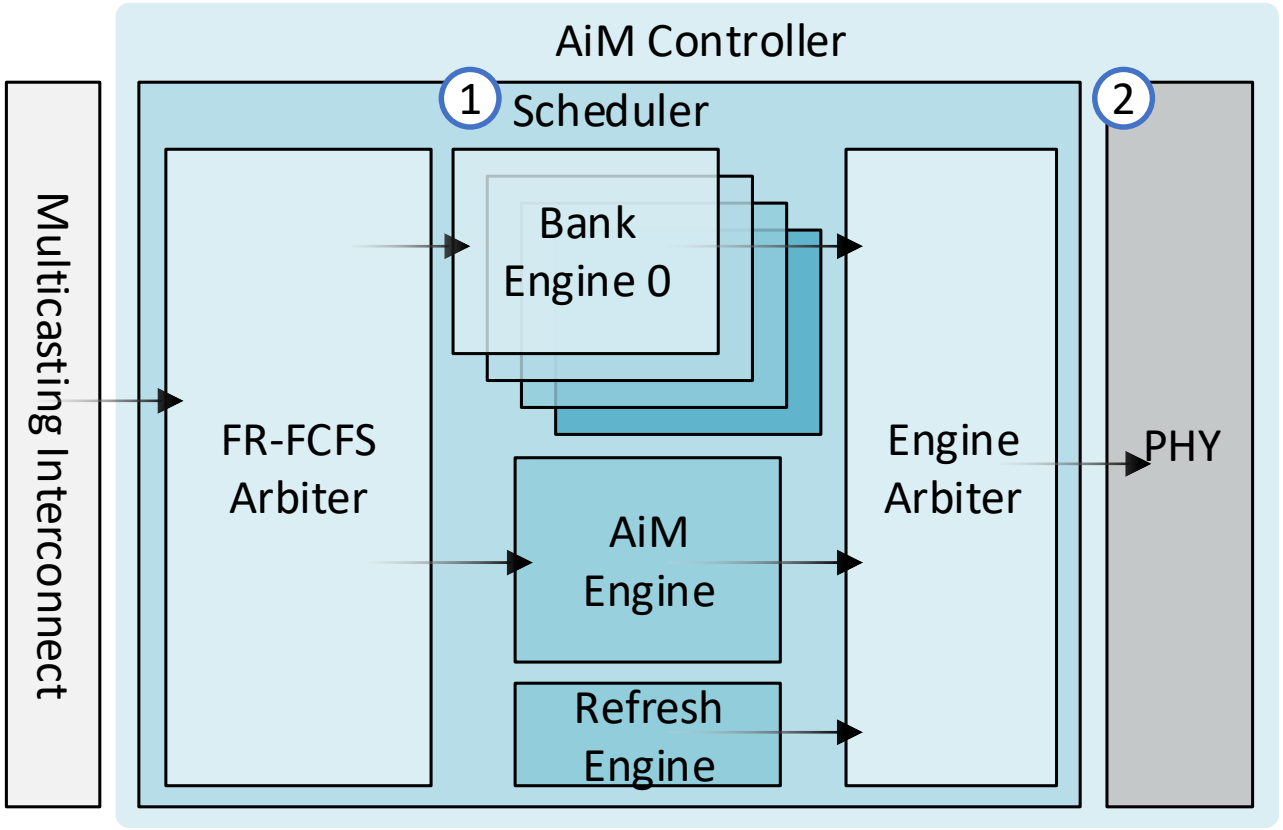
AiM HW/SW Interface: ISR Command Format



Bank Activation	
ACT4, ACT16	Activate four/sixteen banks in parallel
ACTAF4, ACTAF16	Activate rows storing Activation Functions LUTs in four/sixteen banks in parallel
Compute Commands	
MACSB, MAC4B, MACAB	Perform MAC in one/four/sixteen banks in parallel

AiM Controller

GDDR6-AiM Controller = GDDR6 Controller + AiM Command Scheduling



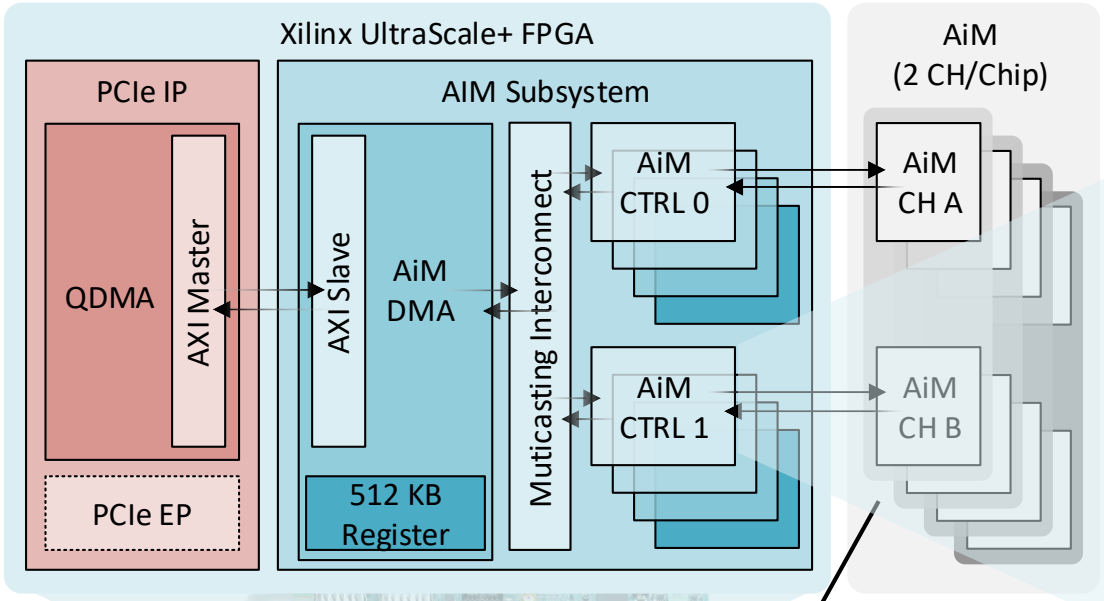
1 Scheduler

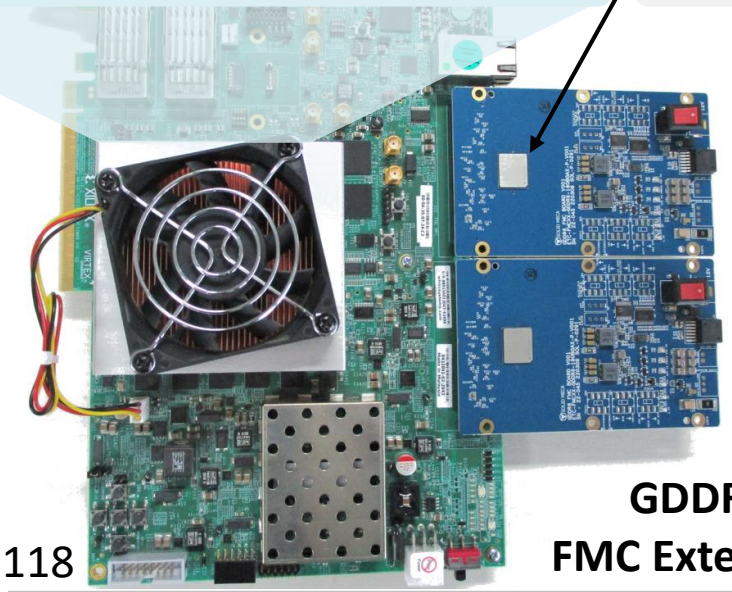
- **FR-FCFS Arbiter** rearranges memory requests to reduce the number of row activations. AiM requests are always scheduled in-order.
- **Bank Engines** and **AiM Engine** generate and schedule DRAM commands (ACT, PREPB, MAC, …) from the memory requests.
- **Refresh Engine** periodically generates and schedules refresh commands.
- **Engine Arbiter** multiplexes between the engines and issues prioritized DRAM commands to PHY.

2 PHY

A standard GDDR6 DRAM PHY block. Details depend on the selected implementation technology (e.g. FPGA).

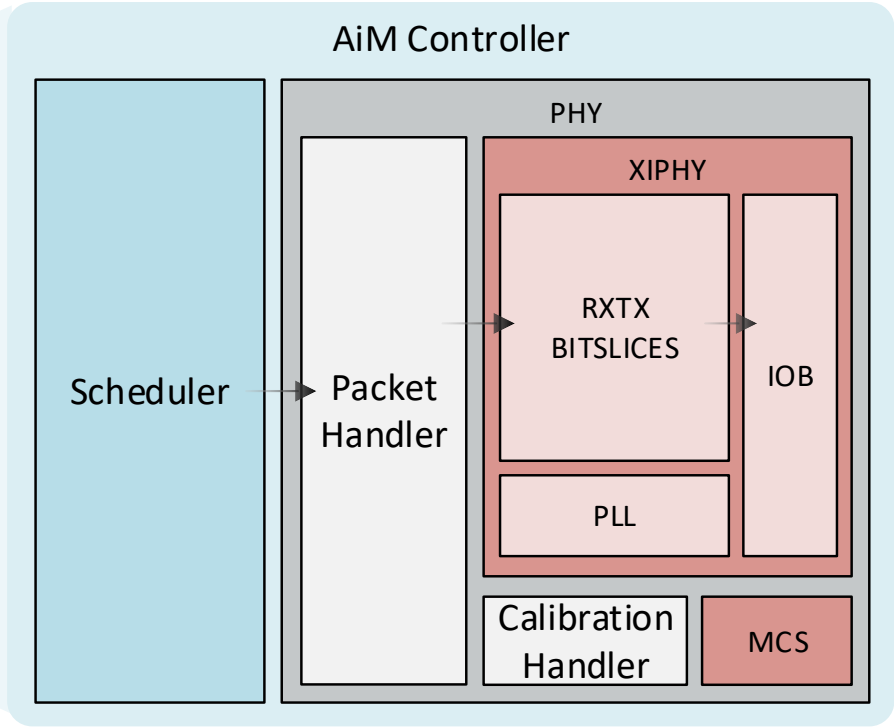
AiM FPGA Platform: AiM Subsystem Prototyped in Xilinx FPGA





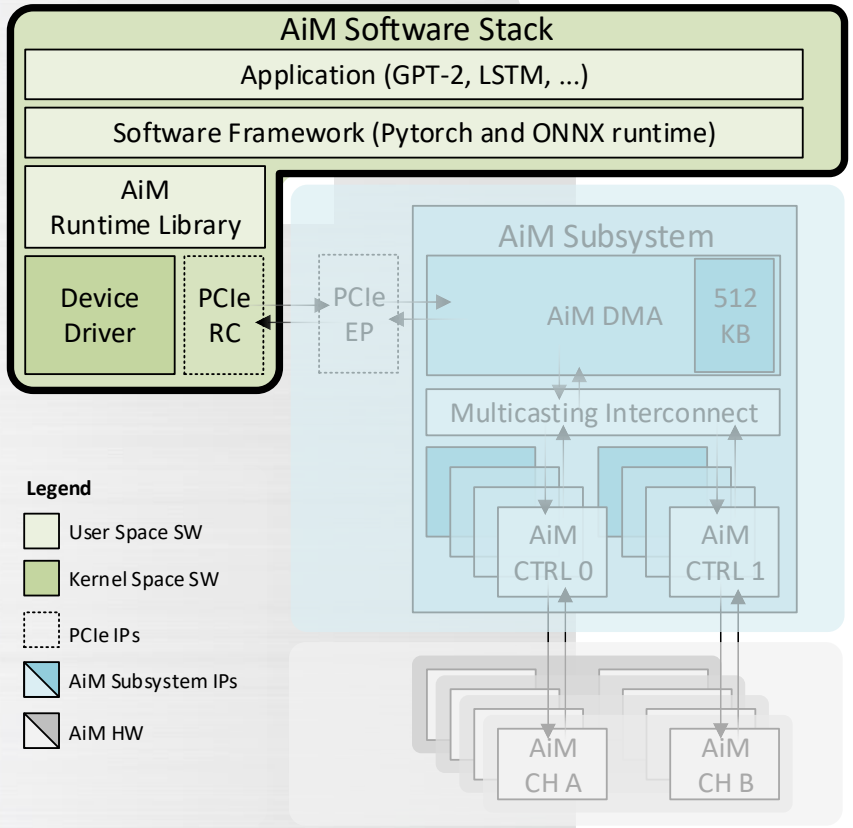
Xilinx VCU118

GDDR6-AiM FMC Extension Card

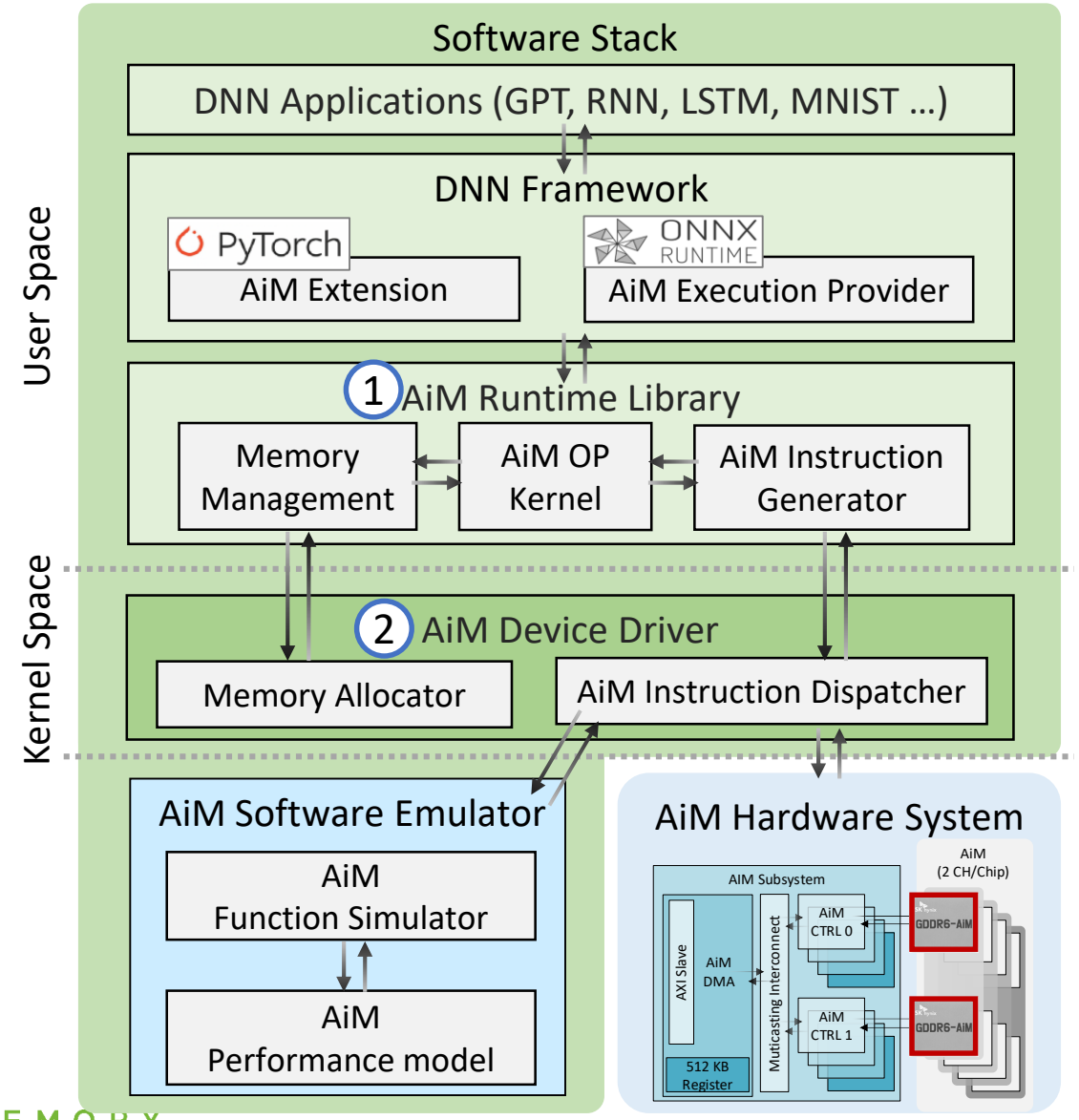


- PHY is implemented by using configurable source-synchronous interface technology (*SelectIO*) available in Xilinx FPGAs.
- Interface calibration is performed by a specialized RTL FSM (Calibration Handler) controlled by an MCS microcontroller.

CONTENTS



AiM Software Stack



1 AiM Runtime Library

- **AiM OP Kernel** includes memory-bound DNN operations such as Linear Op, Sequential OP, RNN OP, and LSTM OP.
- **Memory Management** allocates buffers and reshapes data.
- **AiM Instruction Generator** generates AiM Instruction (ISR Command) Stream for AiM DMA.

2 AiM Device Driver

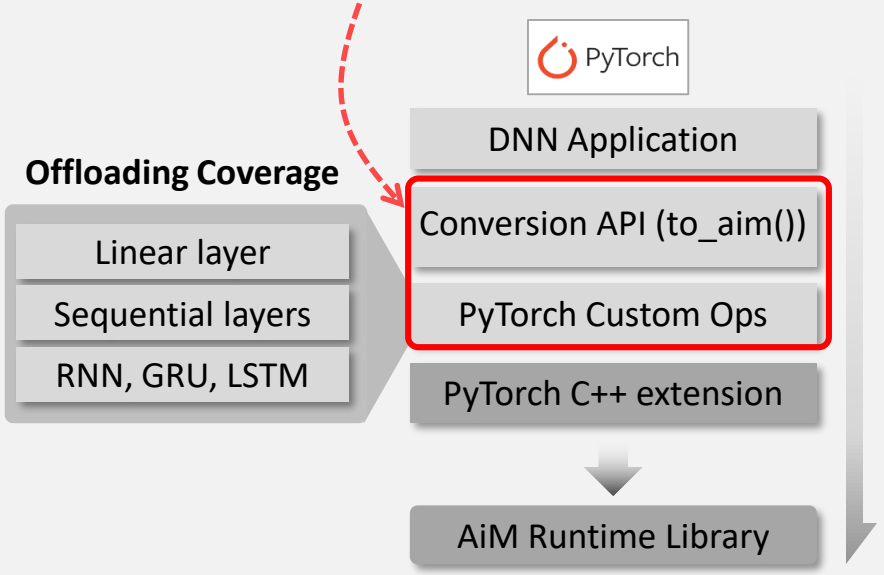
- **Memory Allocator** allocates chunk memories and manages buffers
- **AiM Instruction Dispatcher** dispatches AiM Instruction Stream to AiM DMA

PyTorch and ONNX Runtime Support

PyTorch

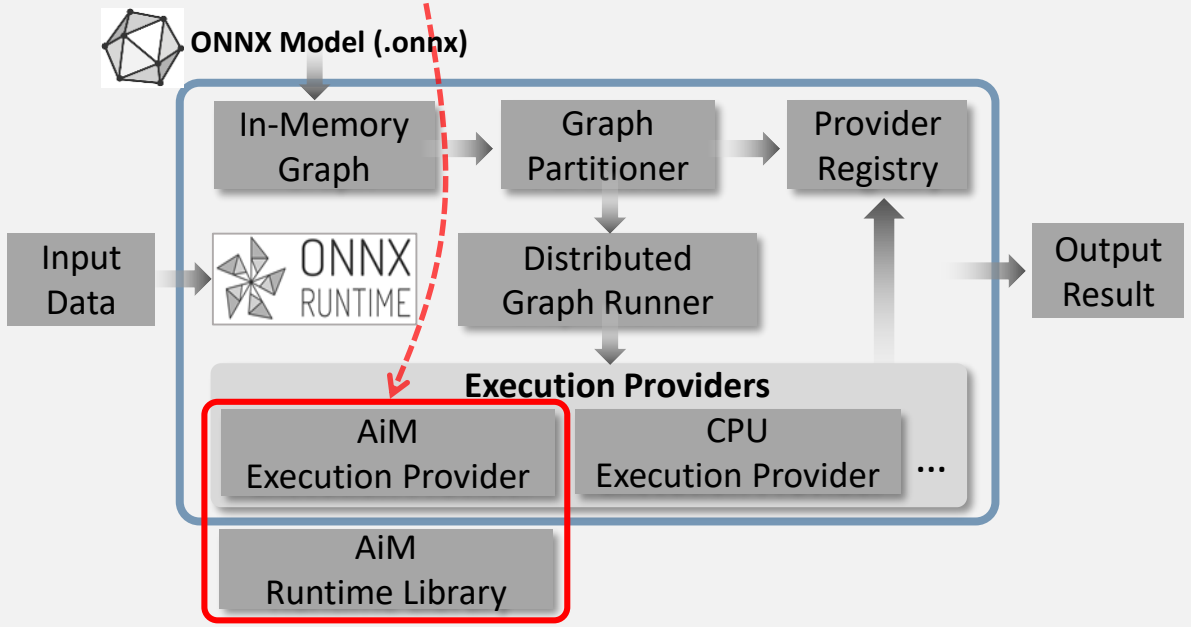
```
model_cpu = torch.nn.LSTM(input_size, hidden_size,
            num_layers).to(torch.device("cpu"), torch.float32)

from aim.pytorch.utils import to_aim
model = to_aim(model_cpu)
...
outputs, (hidden, cell) = model(X)
```



ONNX Runtime

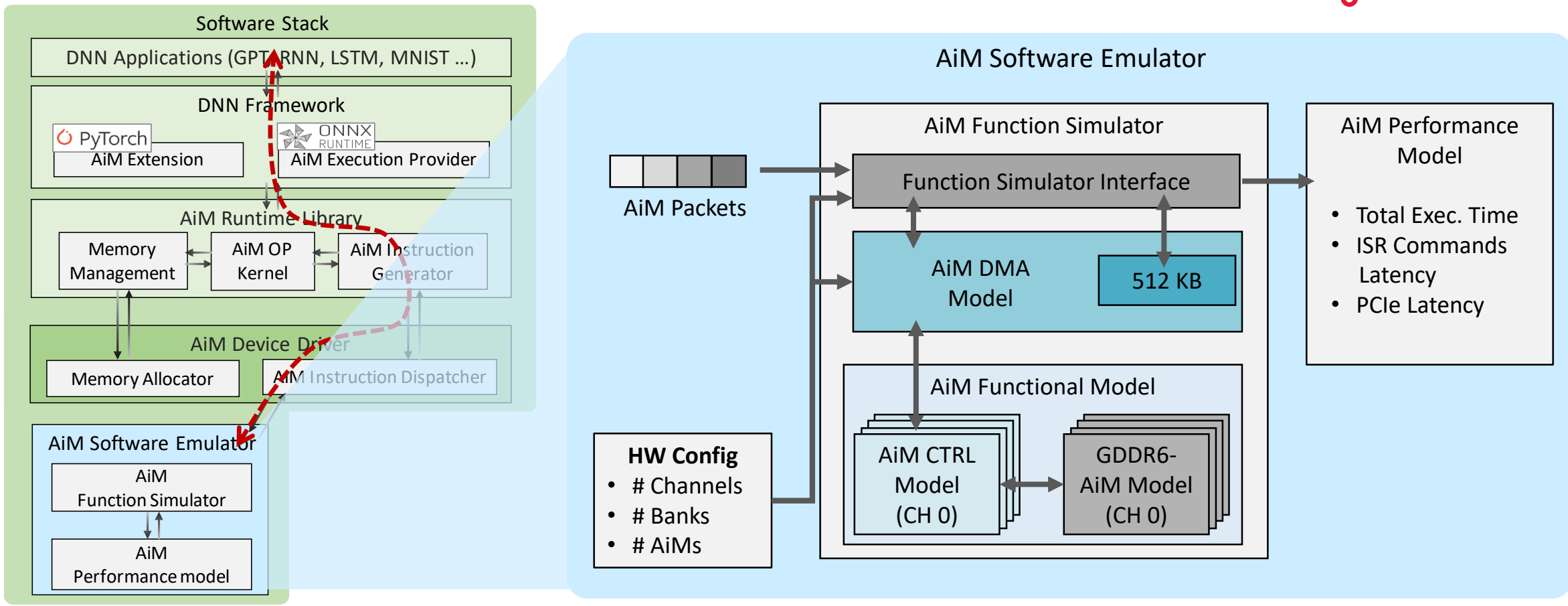
```
onnx_file = './onnx/mnist_'+str(size)+'.onnx'
EP_list = ['AimExecutionProvider','CPUExecutionProvider']
...
sess_aim = rt.InferenceSession(onnx_file, sess_options, providers=EP_list)
...
res_aim = sess_aim.run(None, {input_name: input}, )
```



- Implemented AiM extension using C++ extension
- Exposes AiM Operators to PyTorch via the AiM extension
- Provides ease of programmability with "*to_aim*" API

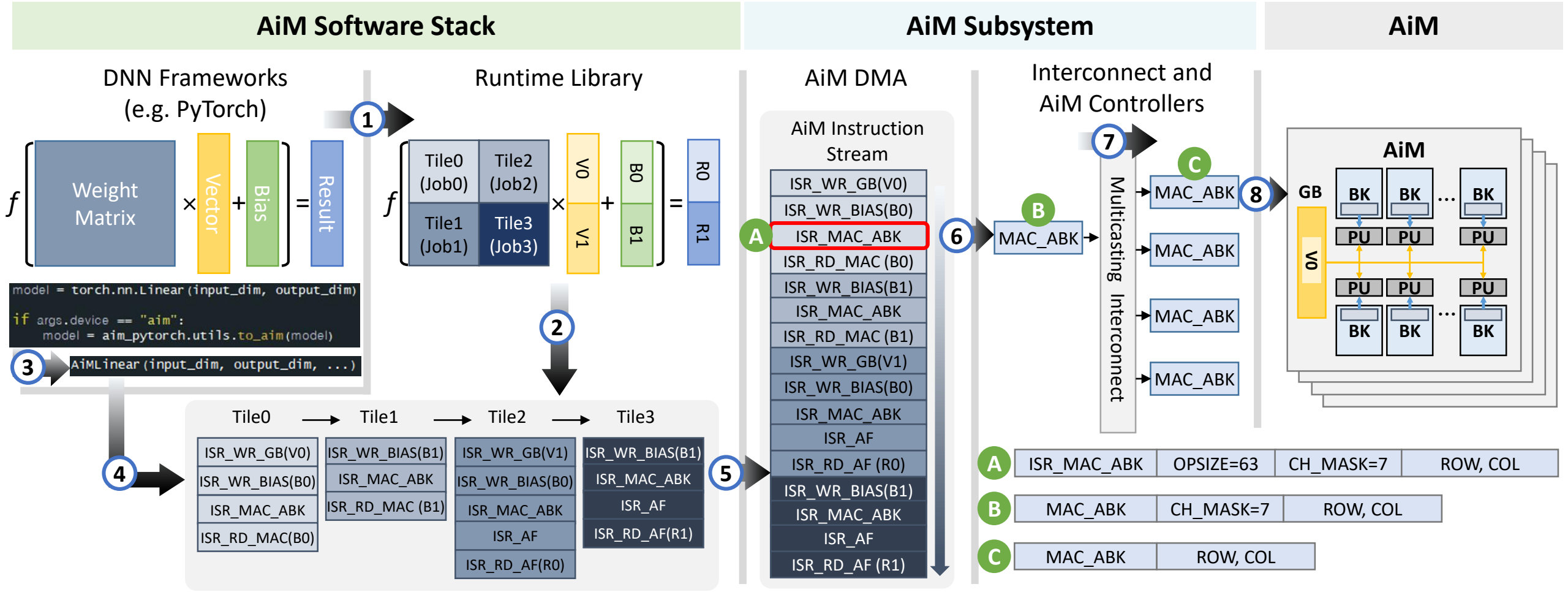
- Implemented using AiM EP (Execution Provider)
- Some nodes in ONNX graph are offloaded, depending on the capability of AiM EP
- Add "*AiMExeucutionProvider*" into the "*EP_List*"

AiM Software Emulator



- Supports AiM function simulator that emulates the functionalities of the AiM FPGA platform
- Offers flexibility of the hardware model with a HW configuration file
- Supports AiM performance analytical model
- Allows to develop their own AI application to run on AiM and to estimate performance without the AiM FPGA platform

ISR Command Flow: From DNN Framework to AiM



① Send the weight matrix information to AiM runtime library. AiM runtime library reshapes the matrix and stores it into AiMs

② Generate an ISR command stream from the weight matrix information

③ Convert the DNN model to the AiM OPs

④ Call the AiM OPs defined in AiM runtime library

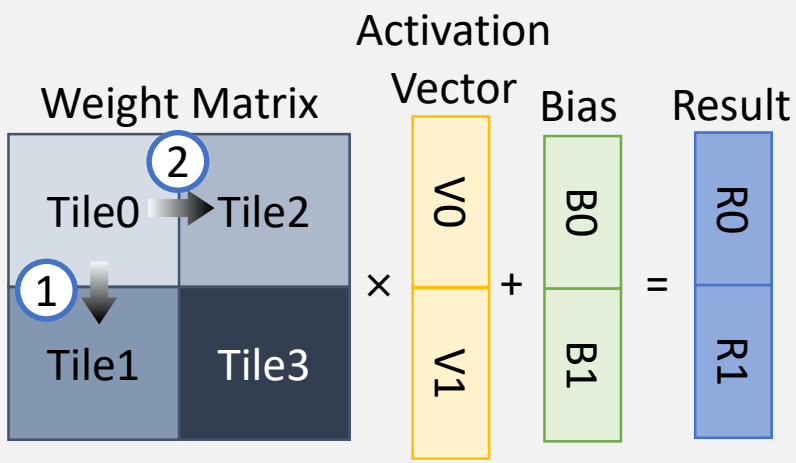
⑤ Send the ISR command stream to AiM DMA via QDMA

⑥ Decode ISR command stream and generate AiM packets. (e.g., ISR_MAC_ABK generates 64 AiM MAC_ABK packets)

⑦ Send or broadcast AiM packets to the AiM controllers. AiM controllers decode AiM packets into AiM commands regarding their state

⑧ Send AiM commands to the AiM devices and execute the operation (e.g., execute all-bank MAC)

AiM Matrix Tiling: Column-major vs. Row-major Tiling



The tradeoffs between column- and row-major tiling may result in different performance, depending on the size and shape of the matrix.

1 **Column-major tiling** keeps activation vector in Global Buffer

Tile 0	Tile 1	Tile 2	Tile 3
ISR_WR_GB(V0)		ISR_WR_GB(V1)	
ISR_WR_BIAS(B0)	ISR_WR_BIAS(B1)	ISR_WR_BIAS(B0)	ISR_WR_BIAS(B1)
ISR_MAC_ABK	ISR_MAC_ABK	ISR_MAC_ABK	ISR_MAC_ABK
ISR_RD_MAC(B0)	ISR_RD_MAC(B1)	ISR_RD_MAC(R0)	ISR_RD_MAC(R1)

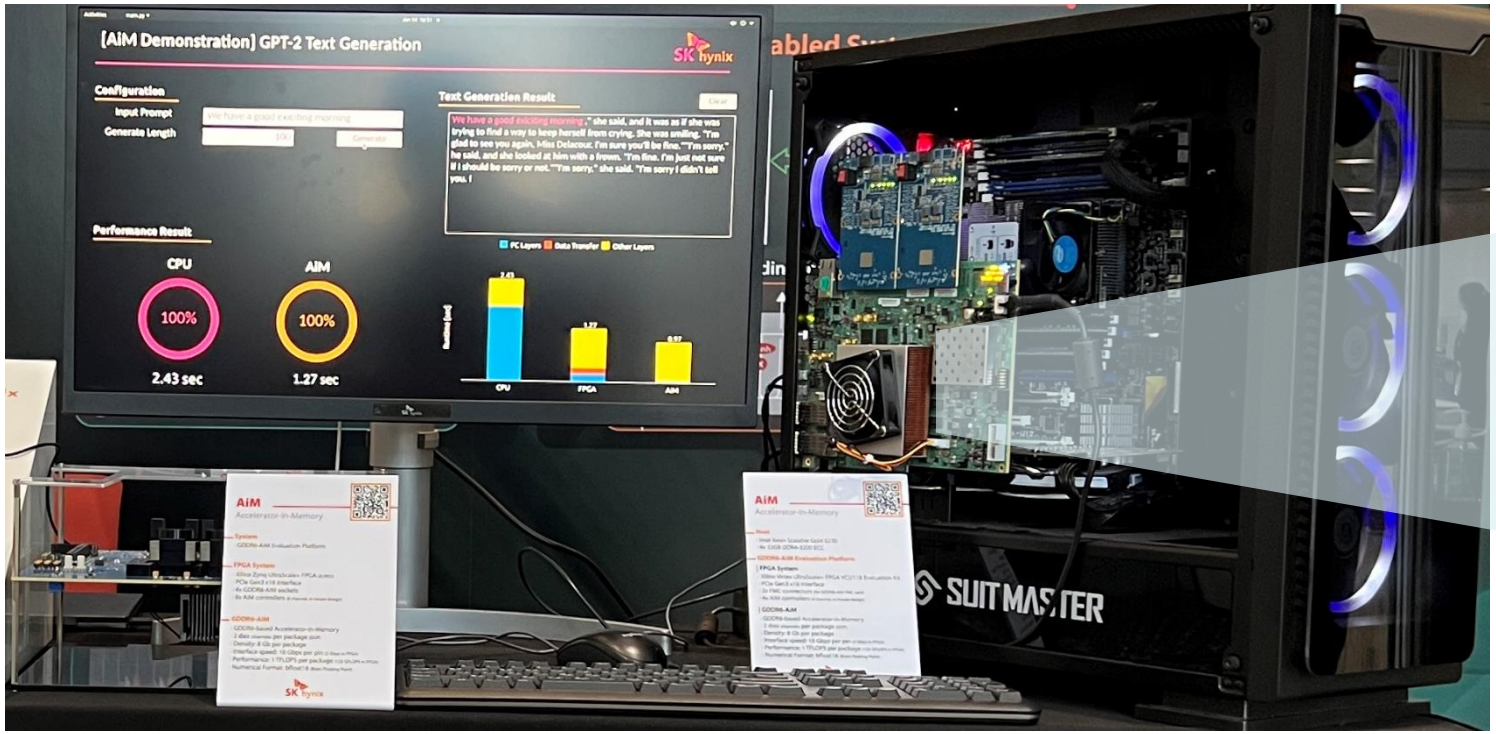
2 **Row-major tiling** keeps partial sums accumulated in MAC_REG

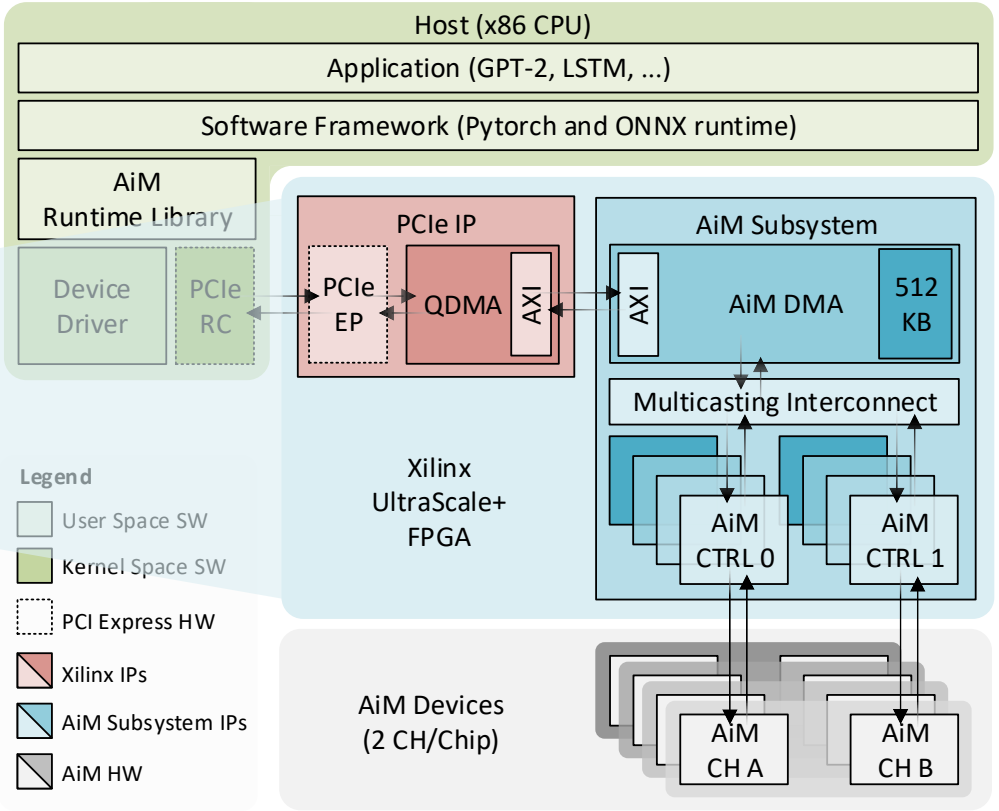
Tile 0	Tile 2	Tile 1	Tile 3
ISR_WR_GB(V0)	ISR_WR_GB(V1)	ISR_WR_GB(V0)	ISR_WR_GB(V1)
ISR_WR_BIAS(B0)		ISR_WR_BIAS(B1)	
ISR_MAC_ABK	ISR_MAC_ABK	ISR_MAC_ABK	ISR_MAC_ABK
	ISR_RD_MAC (R0)		ISR_RD_MAC (R1)

CONTENTS

MEMORY FOREST

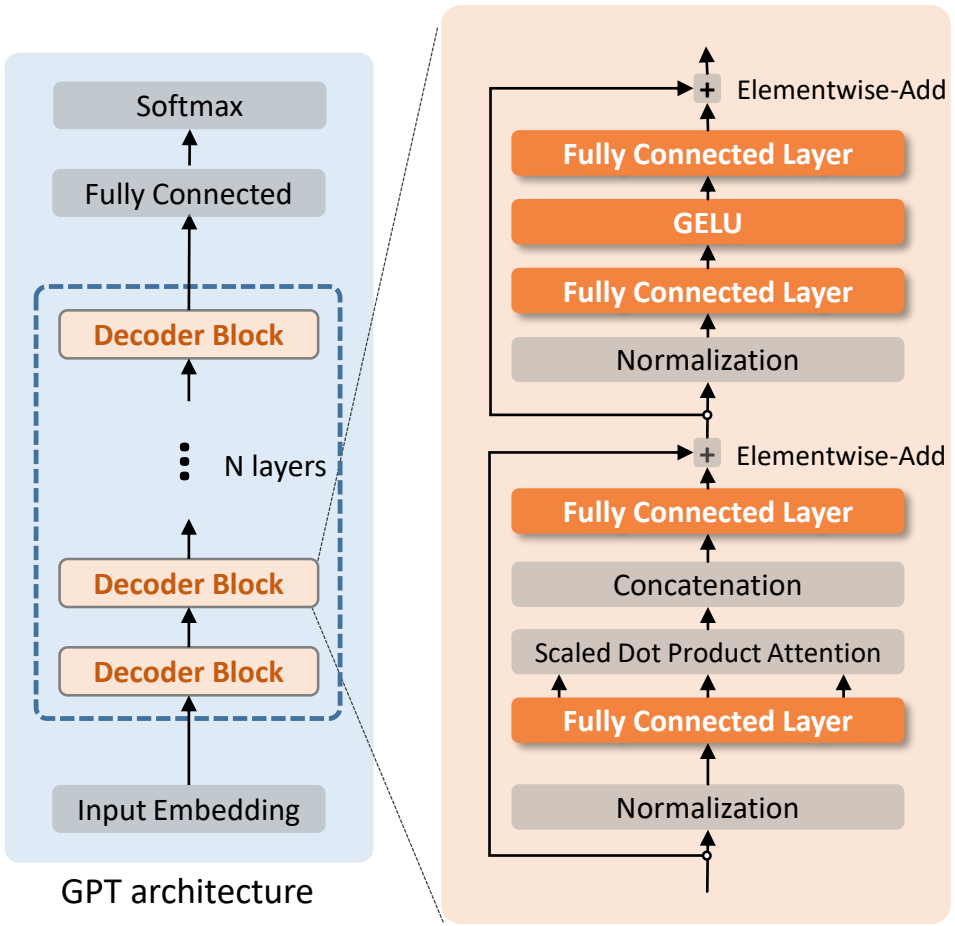
AiM FPGA Platform: Xilinx FPGA board + x86 CPU



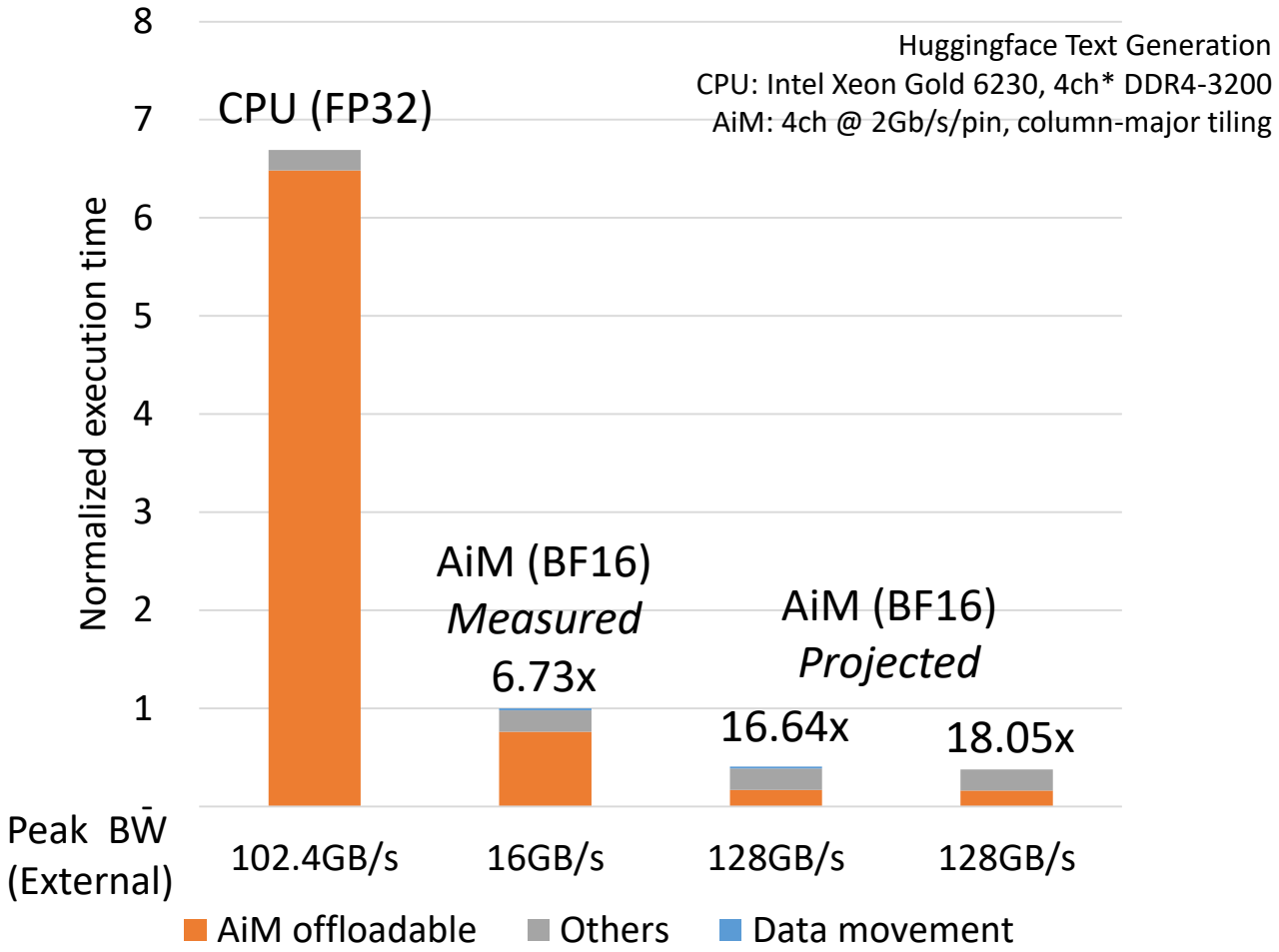


- Xilinx VCU118 + 2 AiM FMC cards (**2 AiM chips** (**4 channels**, **2GB**))
- Running at **2Gb/s/pin** (4GB/s per channel, **256GFLOPs** per chip)
  - *Note: GDDR6-AiM can run at up to 16 Gb/s/pin*

Performance Evaluation: GPT-3 13B configuration

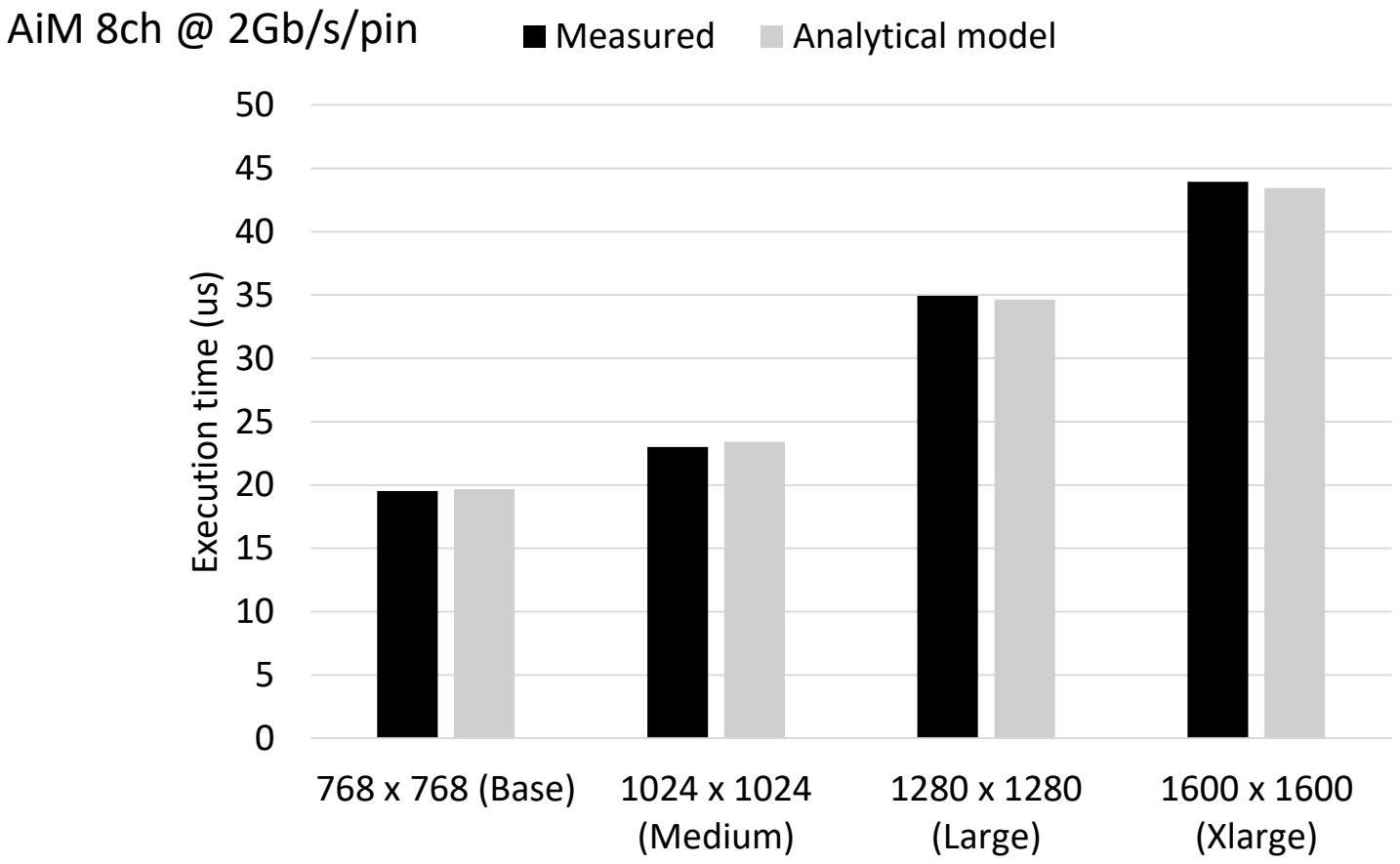


GPT architecture



- Higher gains are expected if AiM is directly deployed on the memory channels running at 16Gb/s/pin, as demonstrated by "*AiM Projected*" with our performance analytical model
- Even higher gains can be achieved when unnecessary data movement over PCIe interface is eliminated

AiM Analytical Performance Model Validation



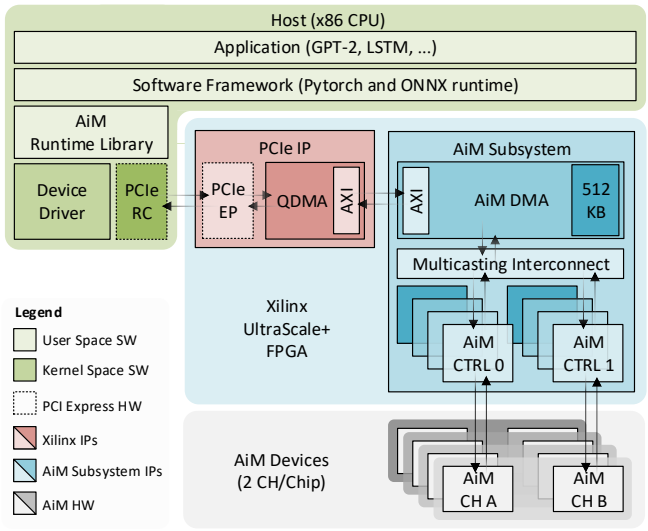
Our performance analytical model matches the measured performance within 5% for varying the number of channels and frequency of AiMs

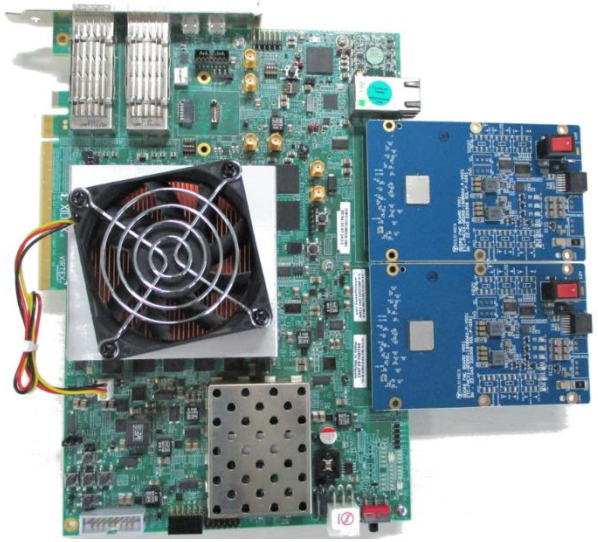
CONTENTS

MEMORY FOREST

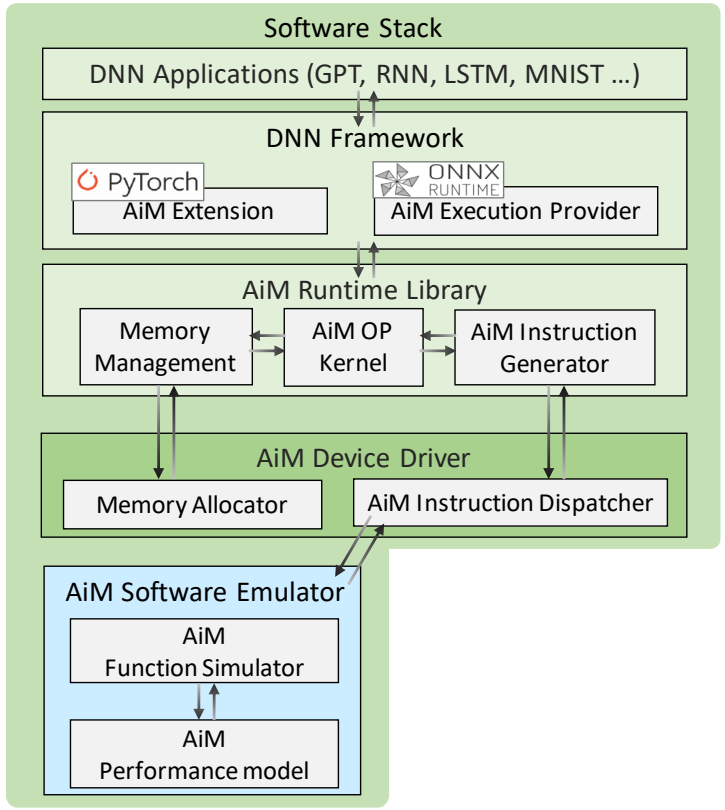
Open Research Platform

AiM FMC extension card with AiM Subsystem (FPGA bitstream)





AiM SDK with AiM Function Simulator



Future Research Topics

- AiM Architecture Exploration
- AiM Controller
  - Scheduling algorithm: Normal DRAM Read/Write vs. Refresh vs. AiM commands
- AiM Software Stack and Applications
  - More applications mapping to AiMs
  - AIM optimal tiling algorithm
  - Automatic decision at runtime whether or not to offload to AiM

SKhynix_PIM@skhynix.com