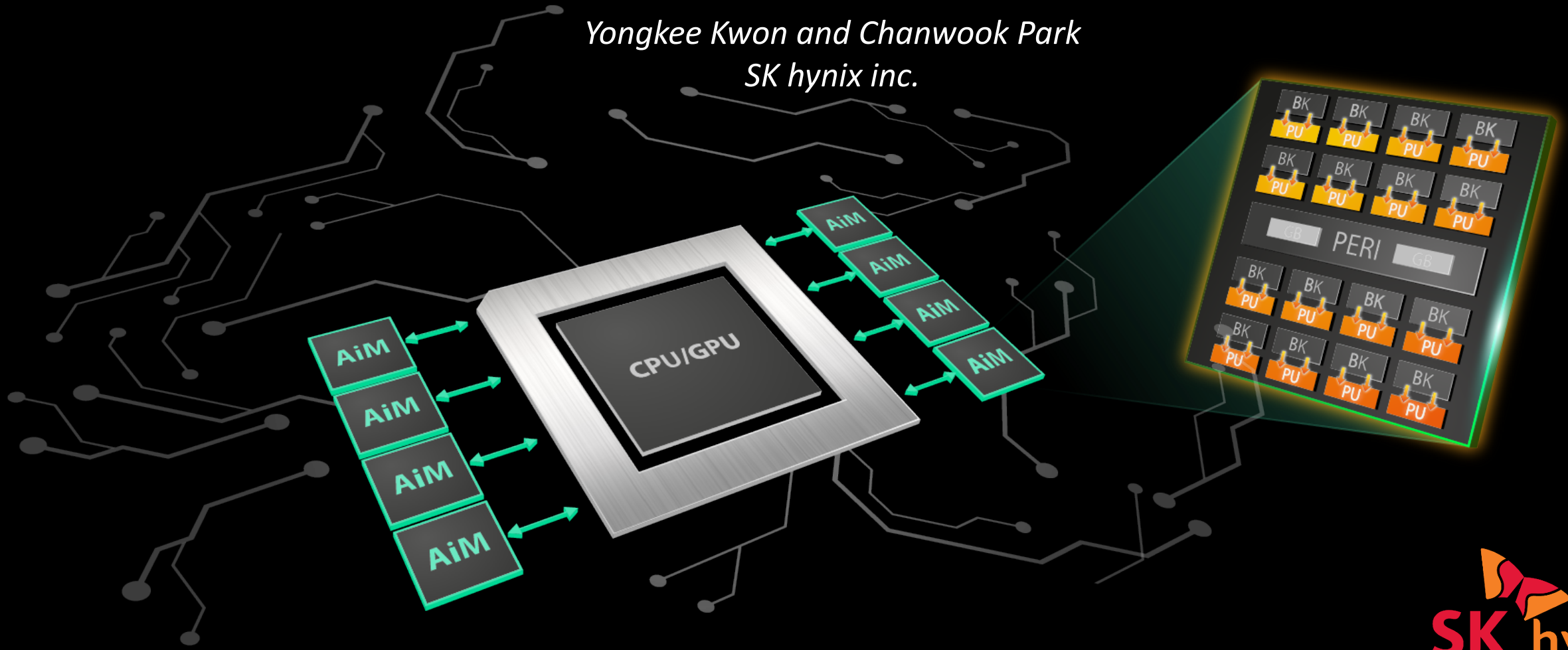
System Architecture and Software Stack for GDDR6-AiM

Yongkee Kwon and Chanwook Park
SK hynix inc.

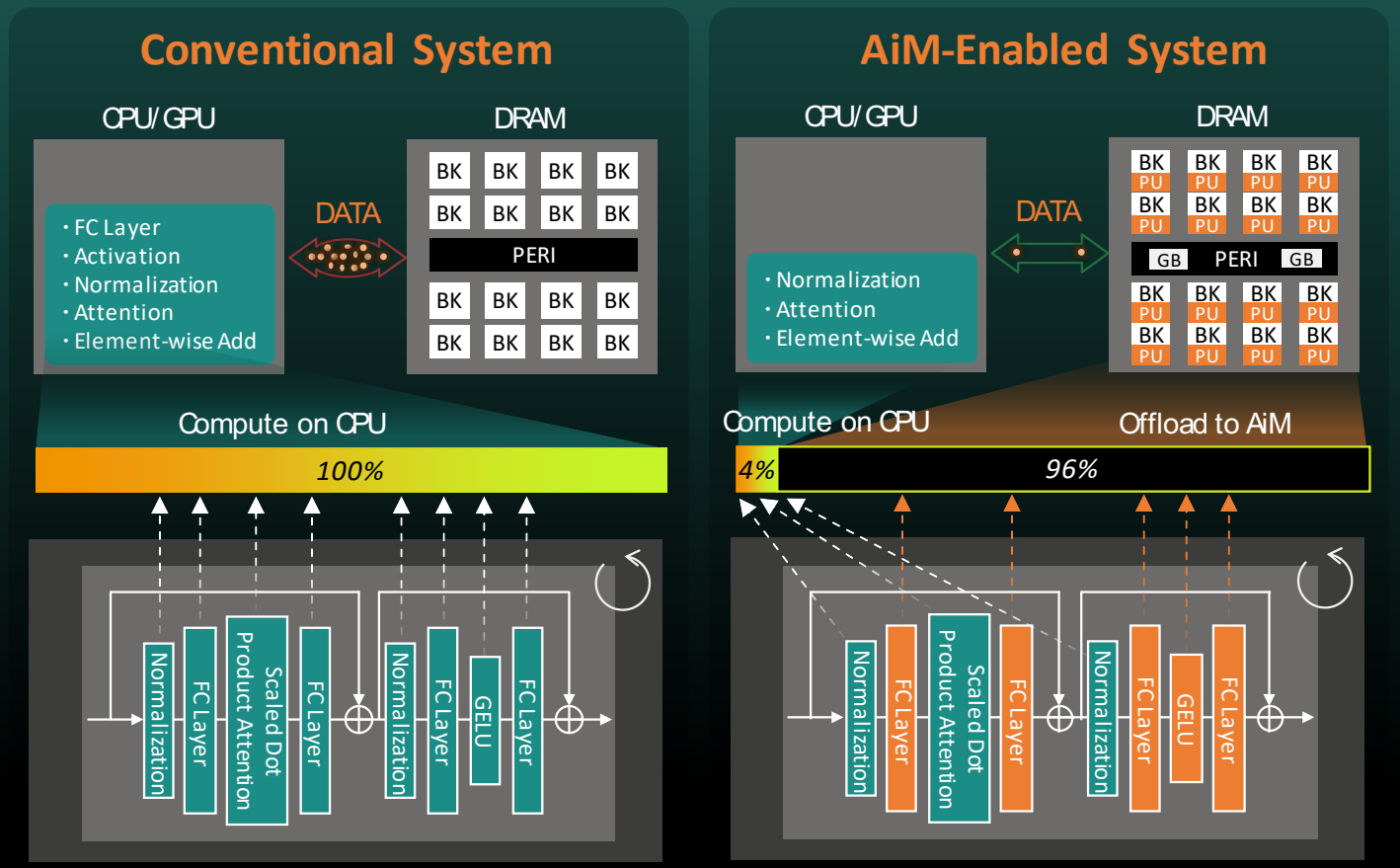


AiM : Accelerator-in-Memory

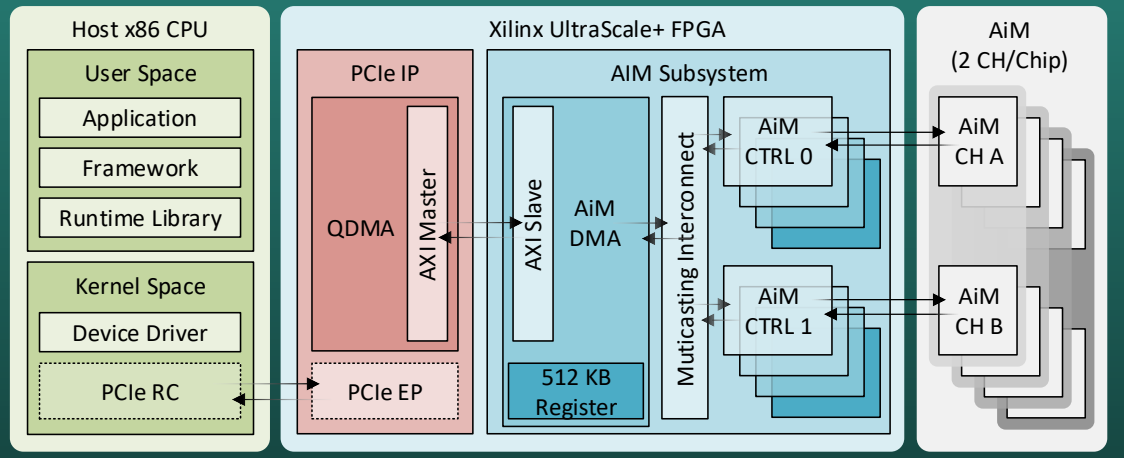
AiM Concept

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

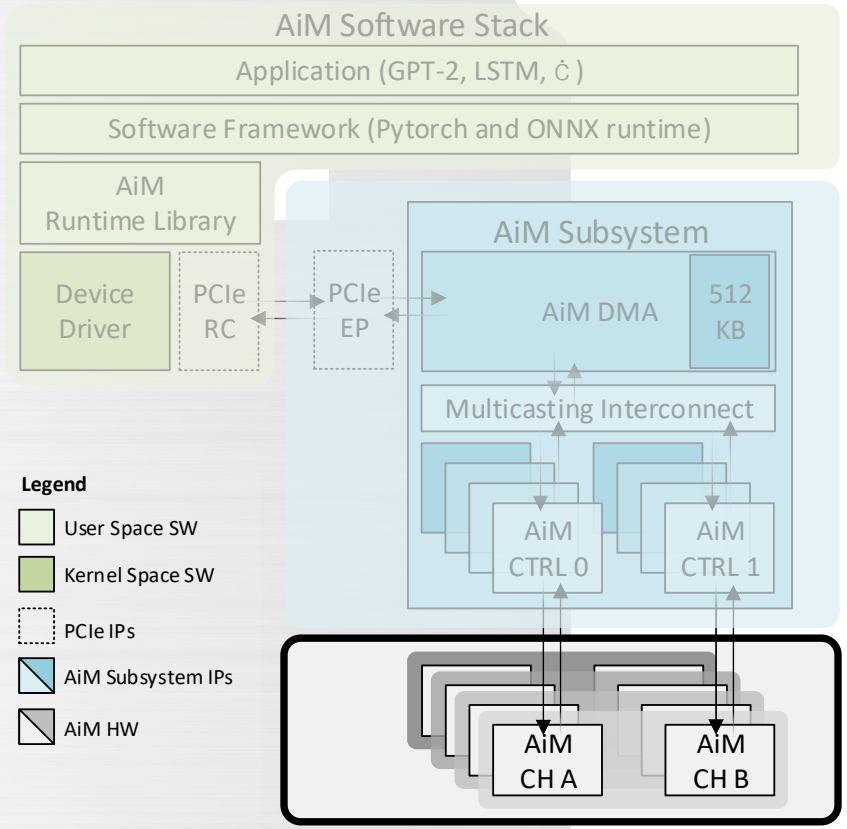


AiM Subsystem and Software Stack



AiM FPGA Platform (w/ CPU Host)

CONTENTS



CACM'20: Domain Specific Accelerators

DOI:10.1145/3361682

DSAs gain efficiency from specialization and performance from parallelism.

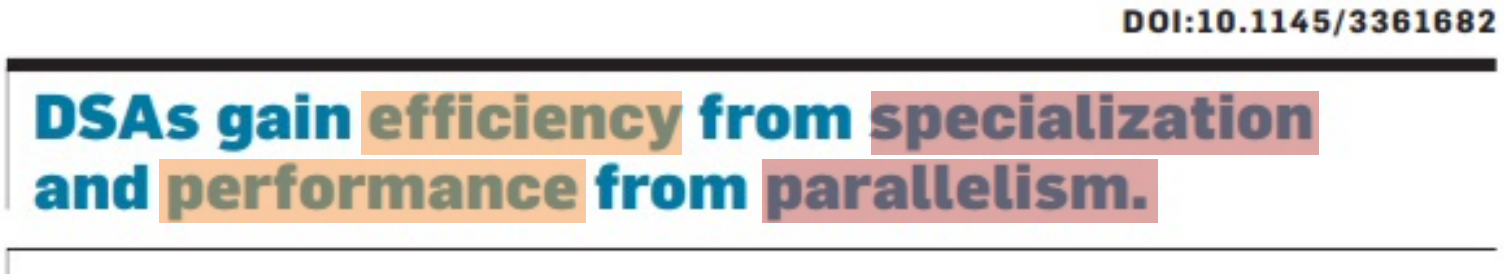
BY WILLIAM J. DALLY, YATISH TURAKHIA, AND SONG HAN

Domain-Specific Hardware Accelerators

look to alternative architectures with lower overhead, such as domain-specific accelerators, to continue scaling of performance and efficiency. There are several ways to realize domain-specific accelerators as discussed in the sidebar on accelerator options.

A domain-specific accelerator is a hardware computing engine that is specialized for a particular domain of applications. Accelerators have been designed for graphics,[26] deep learning,[16] simulation,[2] bioinformatics,[49] image processing,[38] and many other tasks. Accelerators can offer orders of magnitude improvements in performance/cost and performance/W compared to general-purpose computers. For example, our bioinformatics accelerator, Darwin,[49] is up to 15,000× faster than a CPU at reference-based, long-read assembly. The performance and efficiency of accelerators is due to a combination of specialized operations,

AiM as a Domain Specific Memory



Accelerator-in-Memory

Efficiency: specialization for **a particular domain** of applications
Target domain: memory bound DNN applications

Main goals:

1) Performance: high degrees of **bank-level parallelism**

2) Power: reduce significant **data movement**

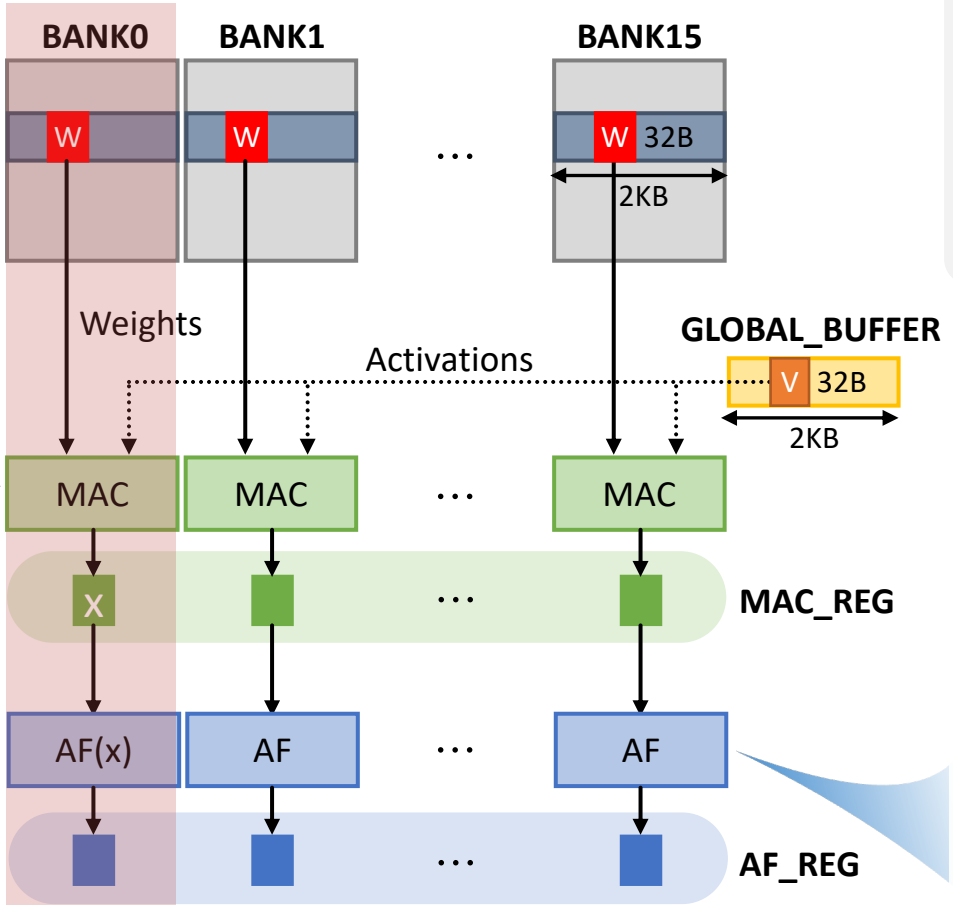
3) Cost: commodity DRAM-based (**GDDR6**)

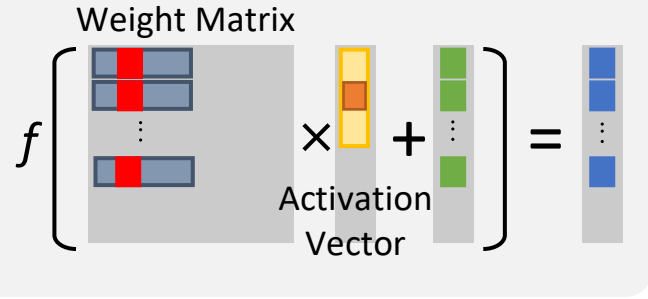
bination of specialized operations,

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

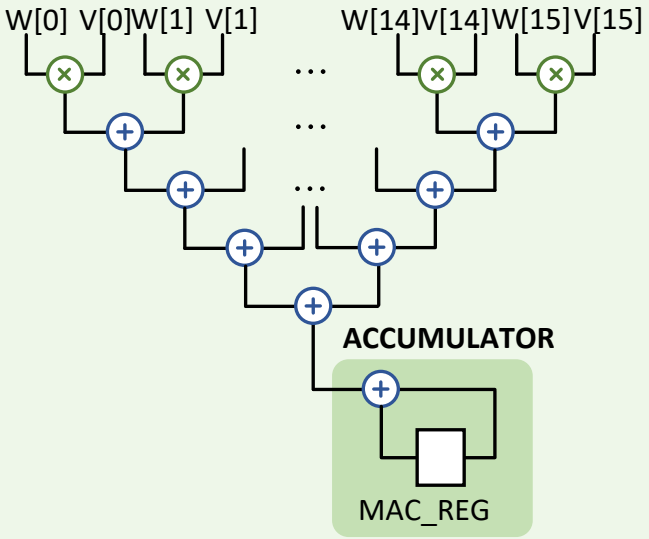
- Performs **Activation Function (AF)** computation by linearly interpolating pre-stored AF template data using MAC calculation results.
- Interpolation results are stored in a dedicated **AF_REG** set and can be later accessed by the user.

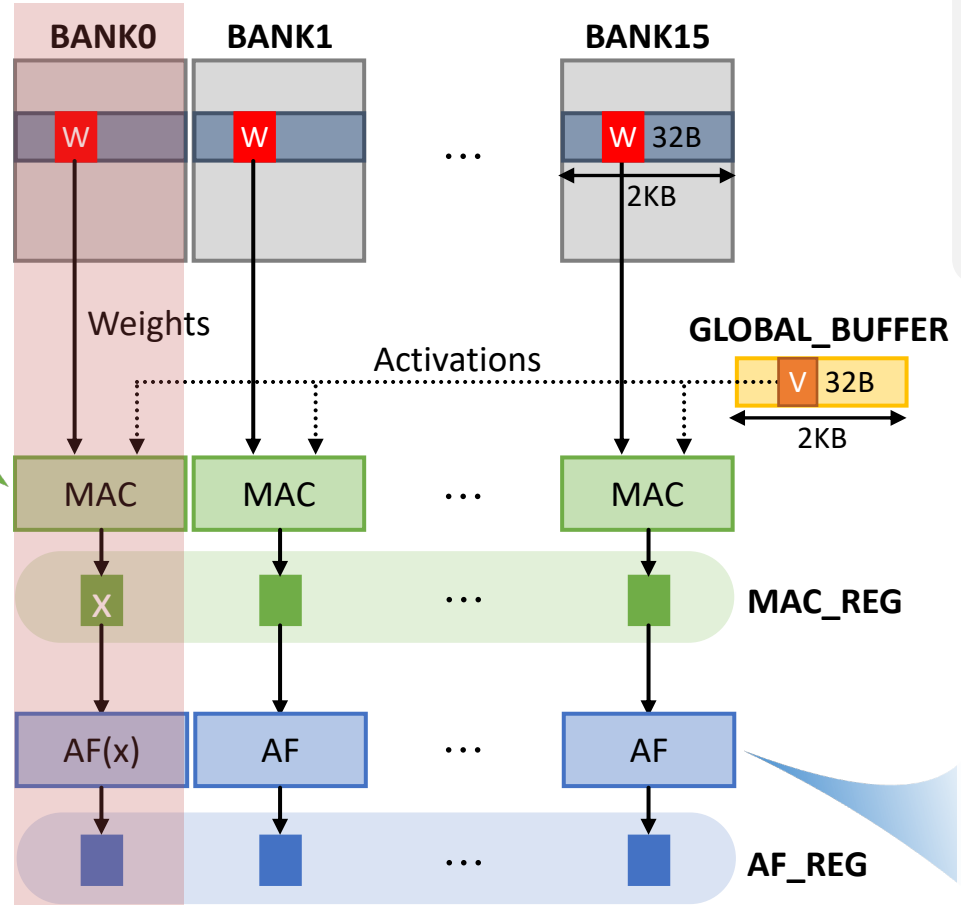
- **MAC** and **Activation Function** operations can be performed in all banks in parallel.
- **Weight** matrix data is sourced from **Banks**; **Vector** data is sourced from the **Global Buffer**.
- **MAC** results are stored in latches collectively referred to as **MAC_REG**.
- **Activation Function**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

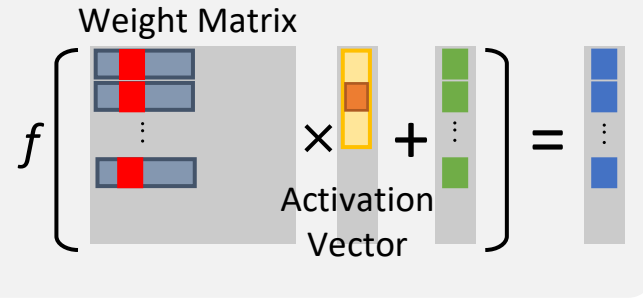
GDDR6-AiM Key Operation: Matrix × Vector

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

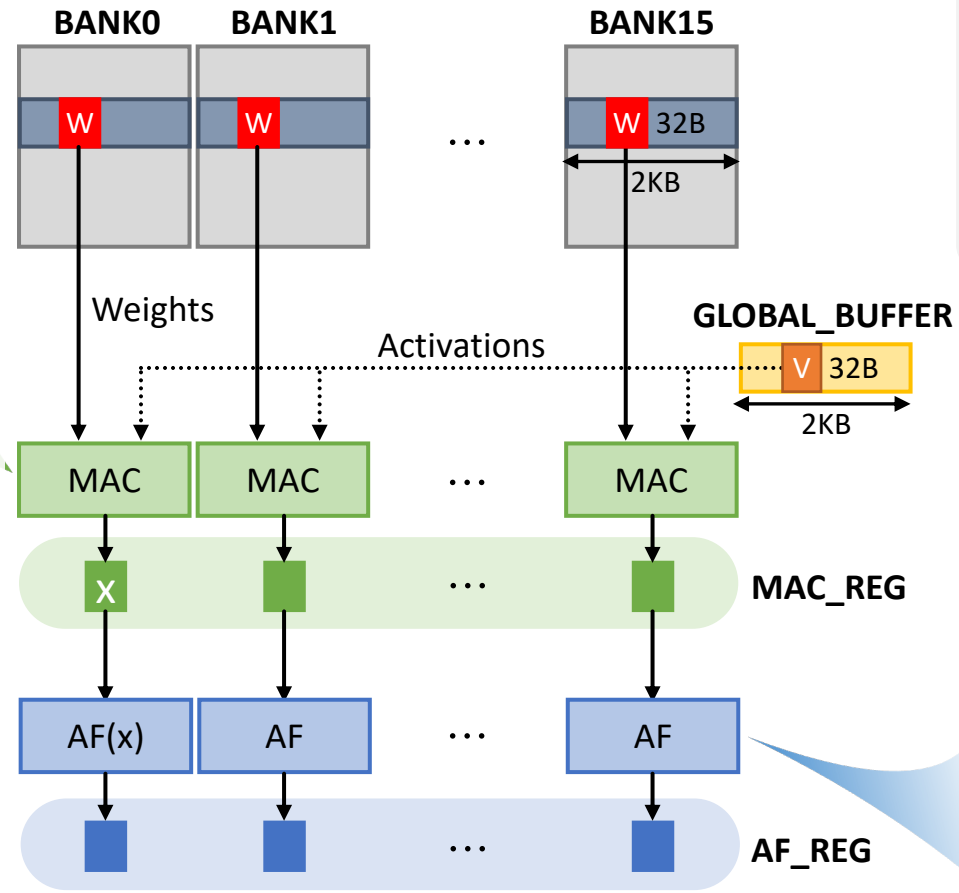
- Performs **Activation Function (AF)** computation by linearly interpolating pre-stored AF template data using MAC calculation results.
- Interpolation results are stored in a dedicated **AF_REG** set and can be later accessed by the user.

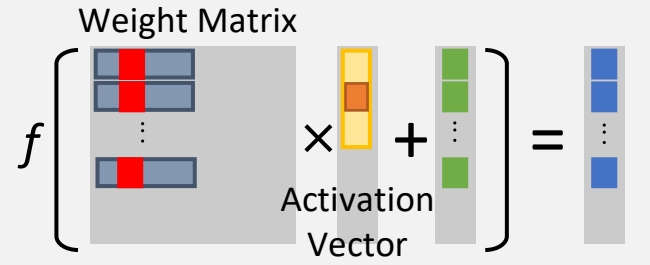
- **MAC** and **Activation Function** operations can be performed in all banks in parallel.
- **Weight** matrix data is sourced from **Banks**; **Vector** data is sourced from the **Global Buffer**.
- **MAC** results are stored in latches collectively referred to as **MAC_REG**.
- **Activation Function**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

GDDR6-AiM Key Operation: Matrix × Vector

Multiply-And-Accumulate (MAC)

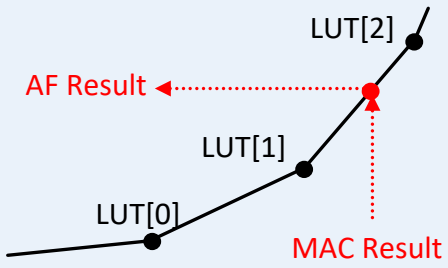
- Performs MAC operation on **sixteen** BF16 weight matrix and vector elements (corresponds to a single DRAM column access, i.e. 32B).
- Computation results are stored in a dedicated **MAC_REG** set and can be later accessed by the user.





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.

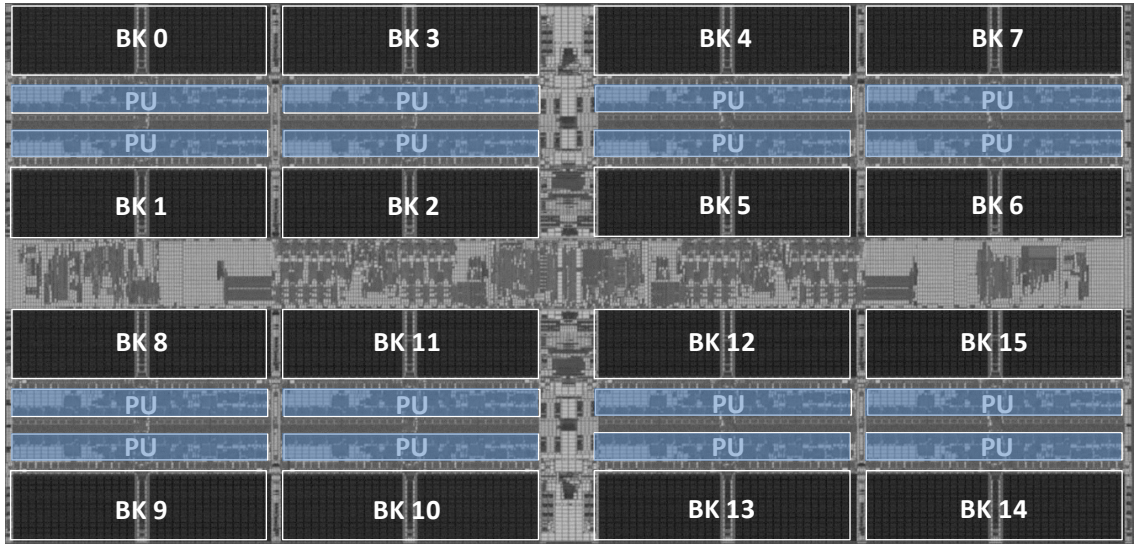


- **MAC** and **Activation Function** operations can be performed in all banks in parallel.
- **Weight** matrix data is sourced from **Banks**; **Vector** data is sourced from the **Global Buffer**.
- **MAC** results are stored in latches collectively referred to as **MAC_REG**.
- **Activation Function**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

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 suppo rt**	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

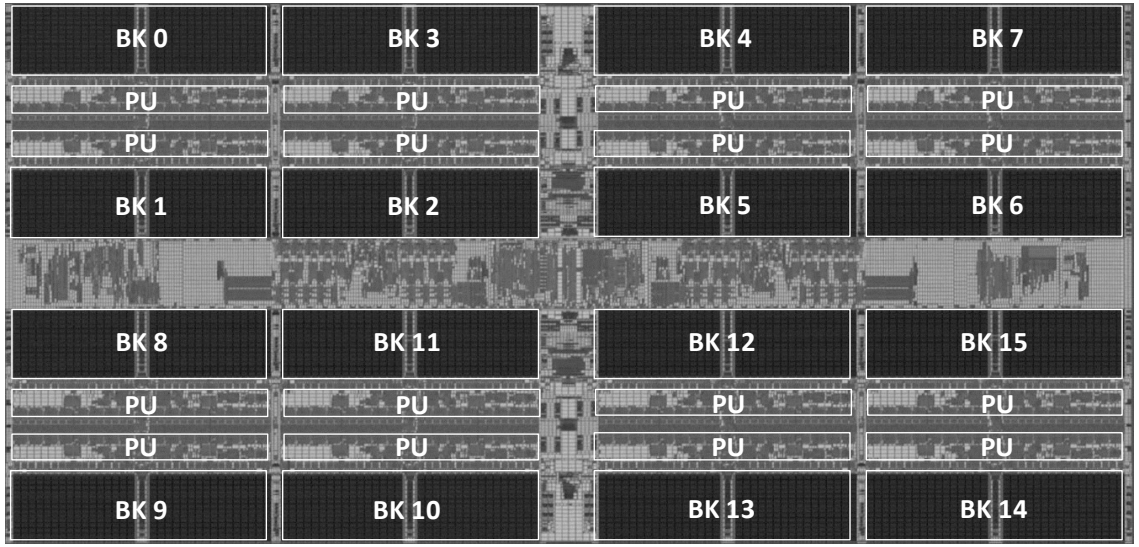
** With using internal lookup table and linear interpolation unit. Any customized function may apply with accuracy limitation.

GDDR6-AiM Feature Summary

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

No compromise on parallelism = performance

	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

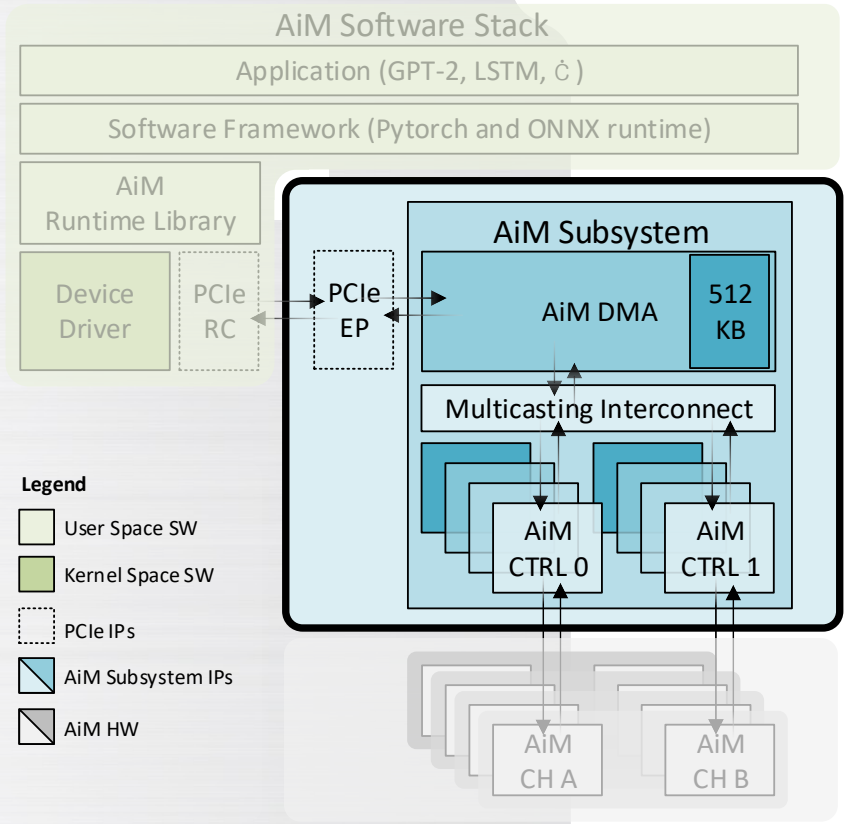
** With using internal lookup table and linear interpolation unit. Any customized function may apply with accuracy limitation.

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

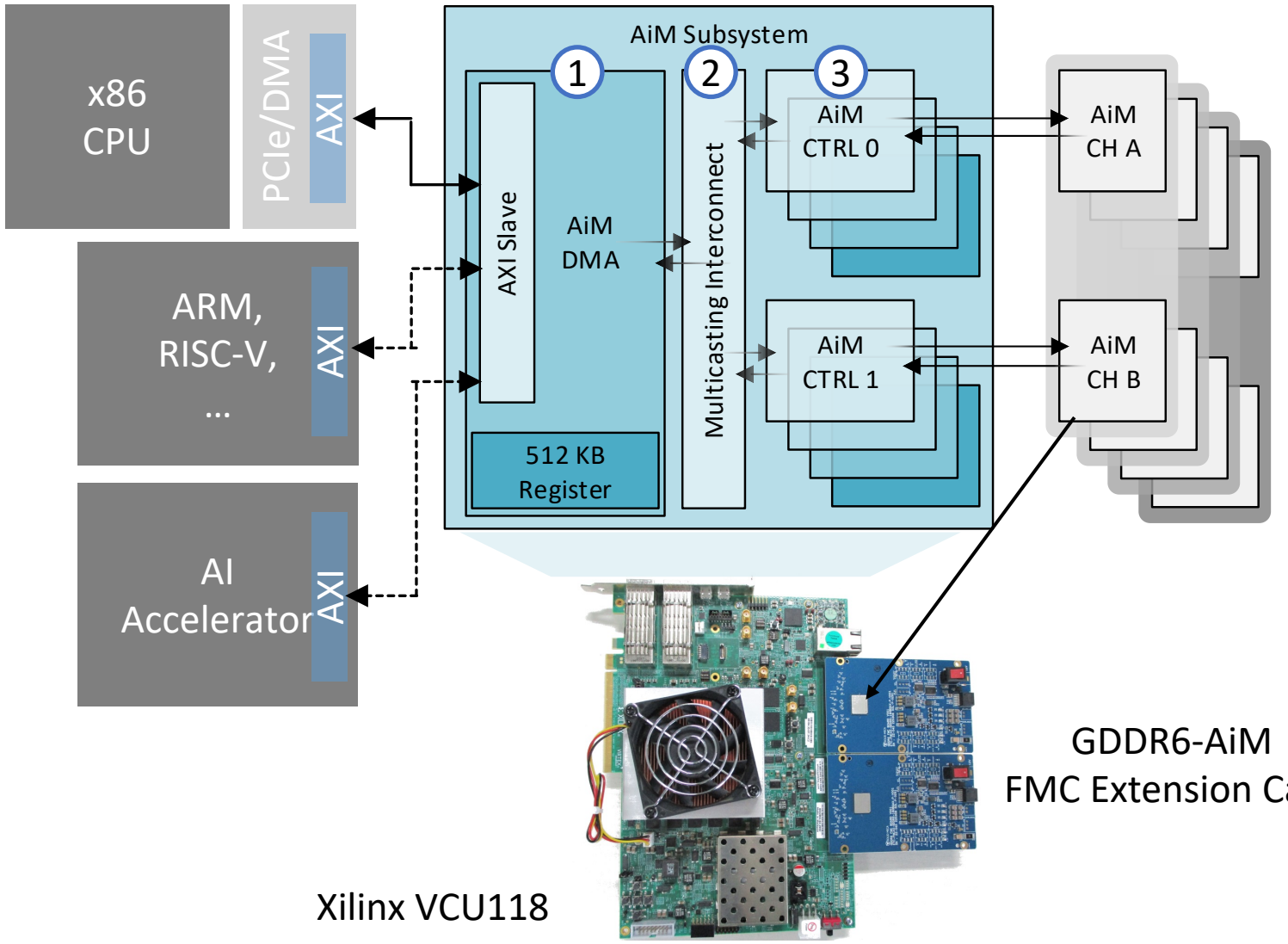
- Commands marked as **CMD*** require a special **Mode Register** set to be recognized.

CONTENTS



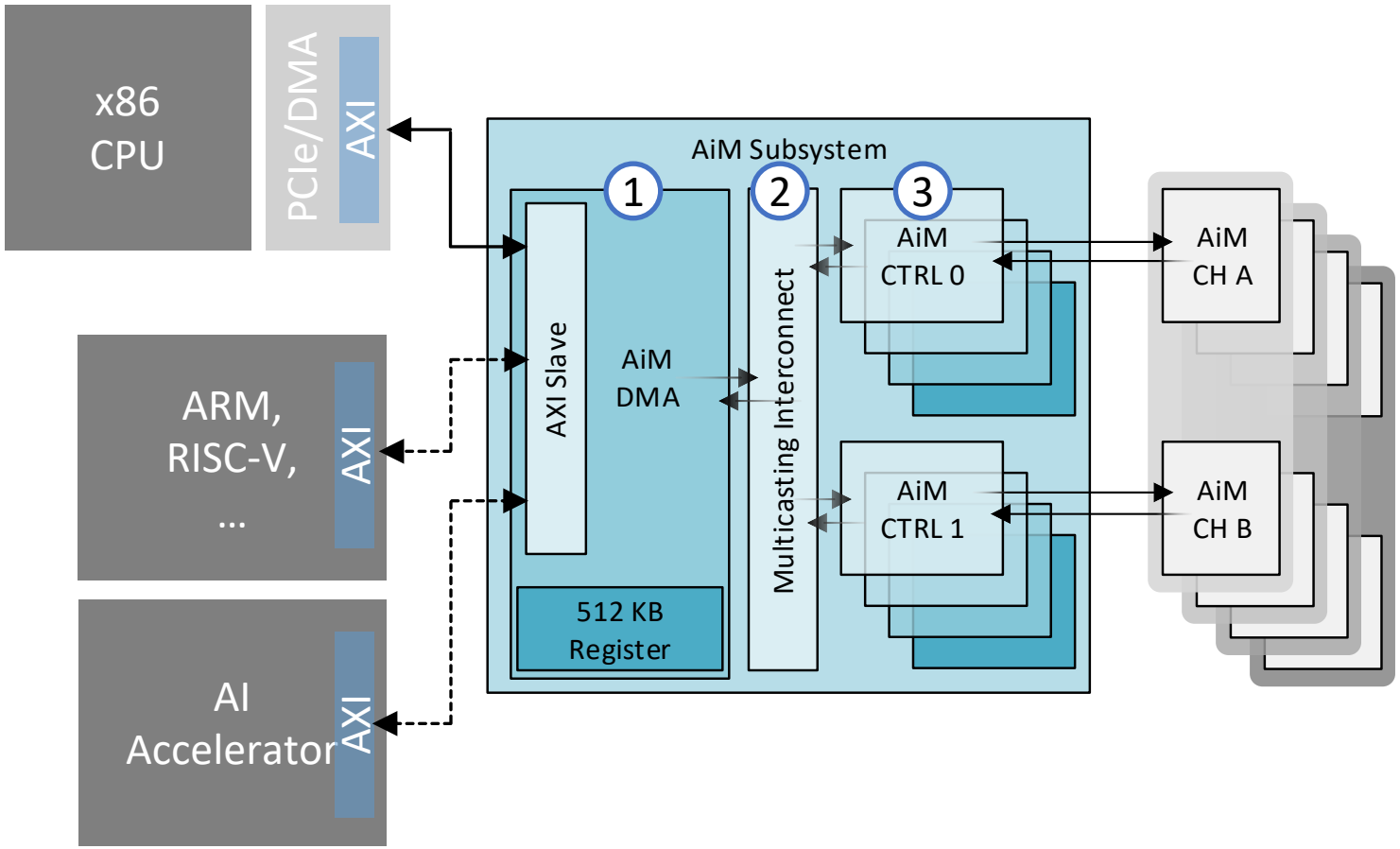
AiM Subsystem: High-Performance Scalable Reference Design

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.



AiM Subsystem: High-Performance Scalable 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.

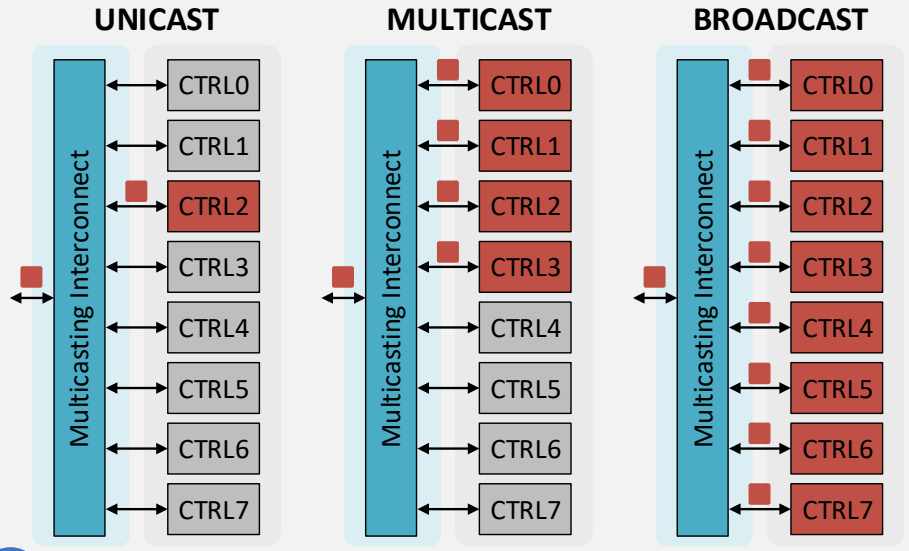


1 AiM Command/Data DMA engine

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 Control Flow from Host Processor

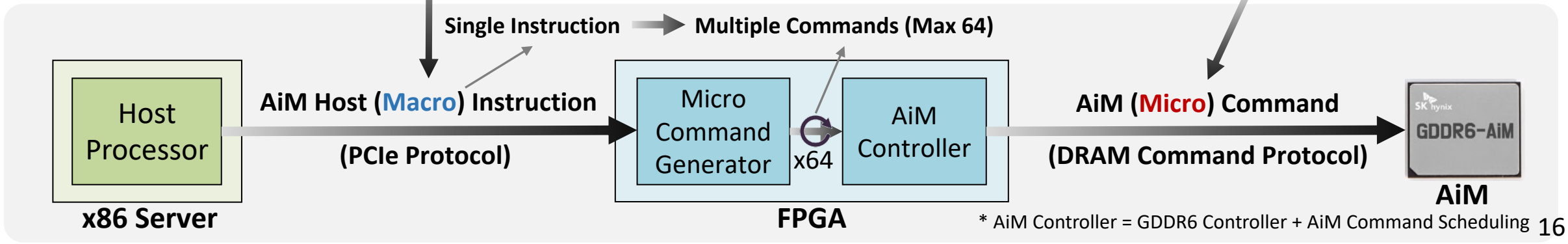
- *New commands named **AiM commands** are introduced to control computing operations in AiM*
- *AiM software stack generates **AiM Host (Macro) Instructions** and each **Macro Instruction** is converted into the corresponding **multiple AiM (Micro) commands** by the Command Generator and AiM Controller*

ISR Instructions

ISR_WR_GB		0x03	C	IO	OPSIZE	G	RESERVED	T	CH_MASK	GPR_ADDR[5]	COL
ISR_WR_BIAS[2]		0x04	C	IO	OPSIZE	G	RESERVED	T	CH_MASK	GPR_ADDR	COL
ISR_WR_AFLUT[1]		0x05	C	IO	OPSIZE	G	RESERVED	T	CH_MASK	AF_IDX	ROW	COL
ISR_RD_MAC[2]		0x06	C	IO	OPSIZE	G	RESERVED	T	CH_MASK[3]	GPR_ADDR	COL
ISR_RD_AF[2]		0x07	C	IO	OPSIZE	G	RESERVED	T	CH_MASK[3]	GPR_ADDR	COL
ISR_MAC_ABK		0x0C	C	IO	OPSIZE	G	RESERVED	T	CH_MASK	BK	ROW	COL
ISR_AF		0x0D	P	57	RELU_SLOPE	42	RESERVED	T	CH_MASK	AF_IDX	ROW	COL

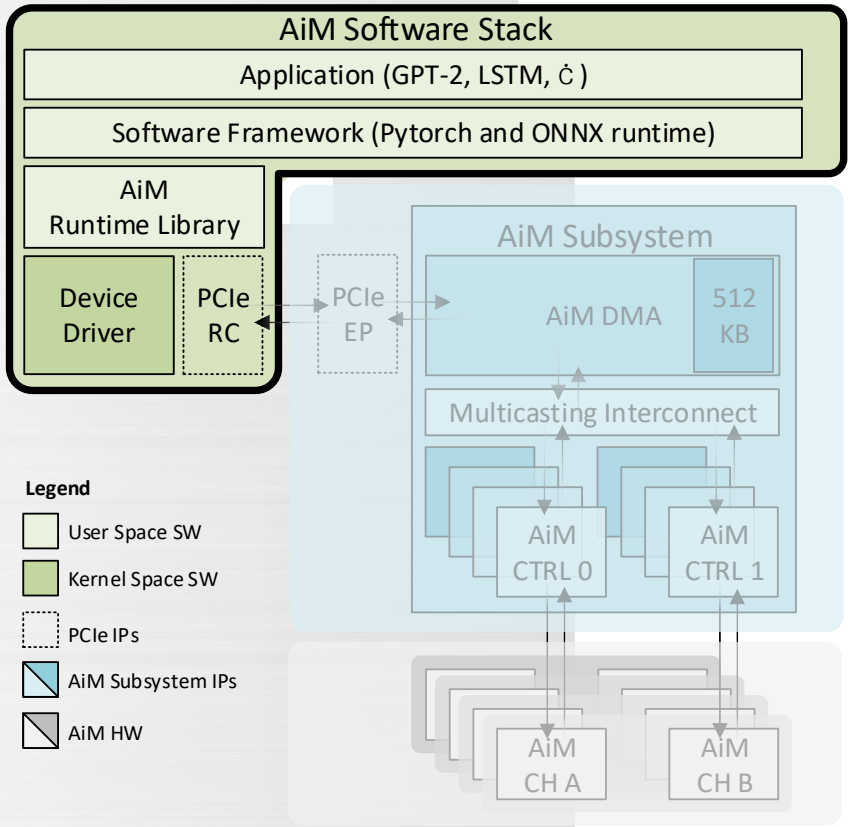
AiM Command

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



* AiM Controller = GDDR6 Controller + AiM Command Scheduling

CONTENTS



Motivation

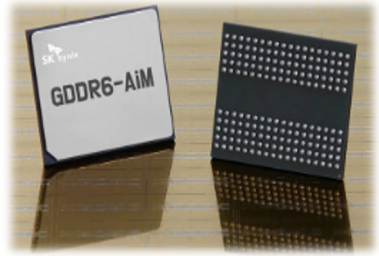
- *In order to increase the value of new memory solutions such as AiM, expanding the ecosystem is of paramount importance.*
- *To achieve this goal, we have developed an SDK to allow any DNN models to be easily adapted by operator-level APIs.*
- *The proposed API abstracts the functionality of PIM operations and provides ease of programmability to users.*

AiM SDK

+

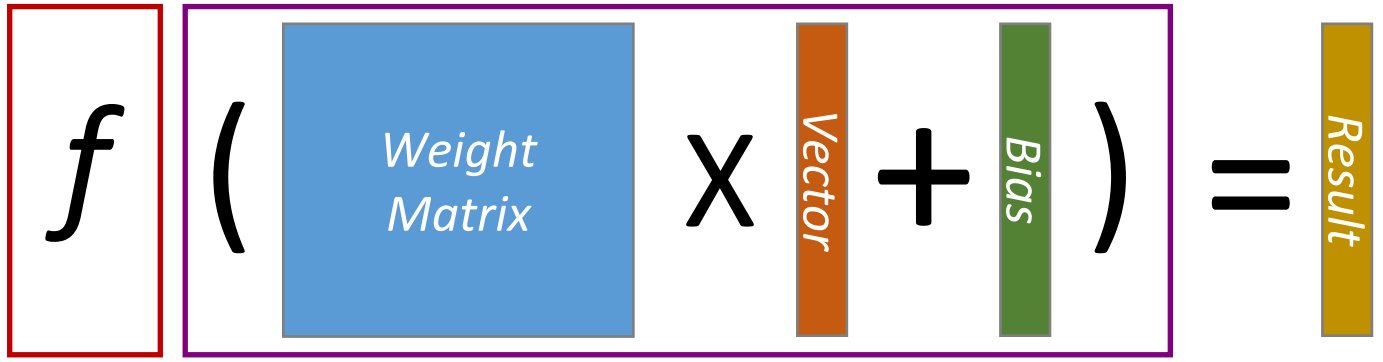
(AiM HW Platform)

distribute

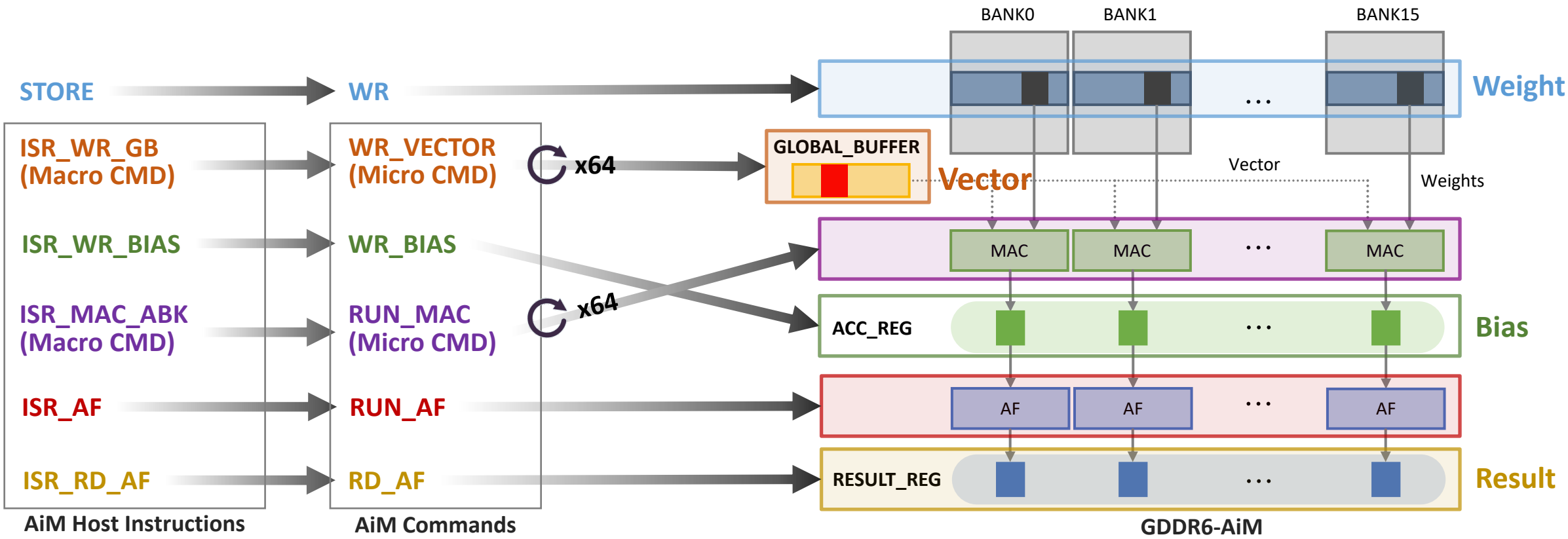


Expanding Ecosystem

AiM Execution Sequence for Fully Connected Layer

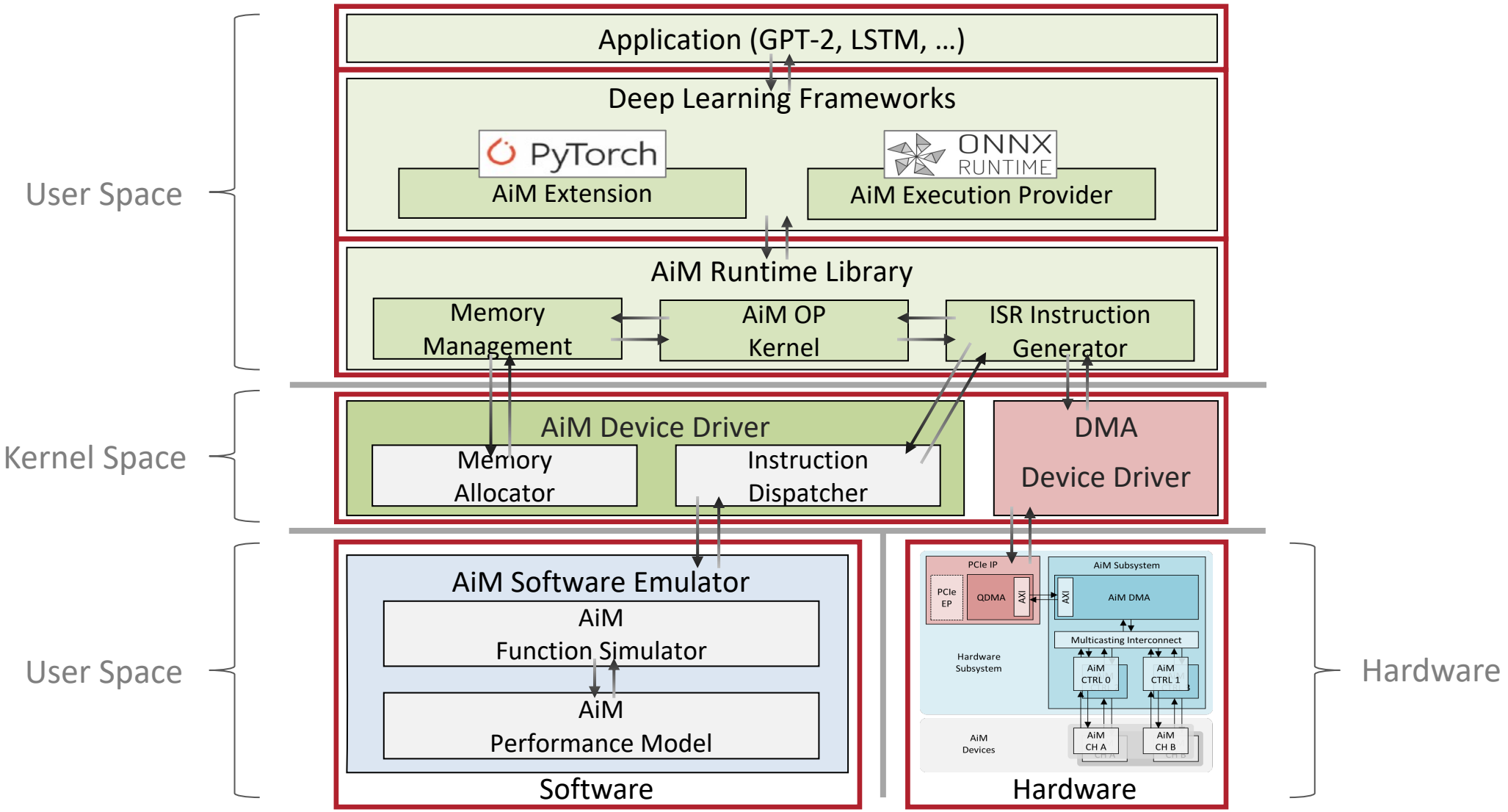


* AF : Activation Function



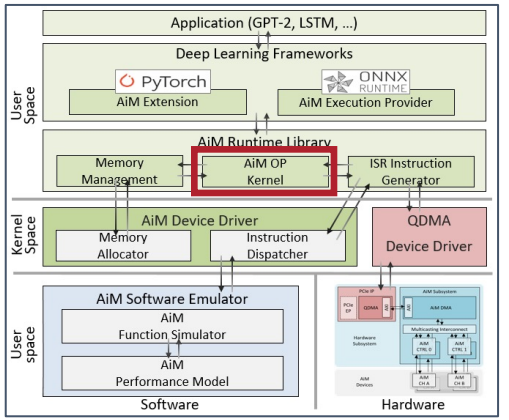
AiM Software Stack

- *SK hynix has implemented full software stack (device drivers, runtime library, frameworks and applications)*
- *Supports an AiM software emulator that allows users to develop AI applications without evaluation board*



AiM Operation Kernels

- *The AiM Runtime Library provides a number of AiM OP kernels for Deep Learning Frameworks and AI Applications by exposing operator-level APIs.*

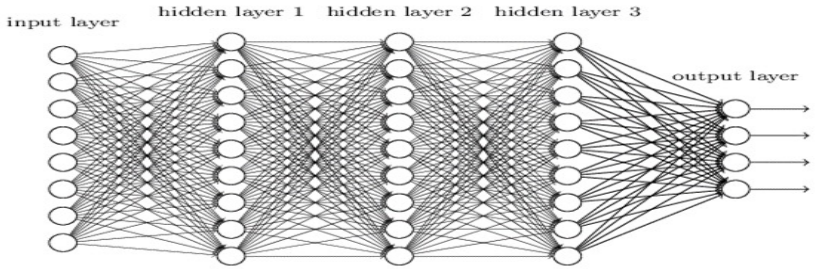


AiM OP Kernel API

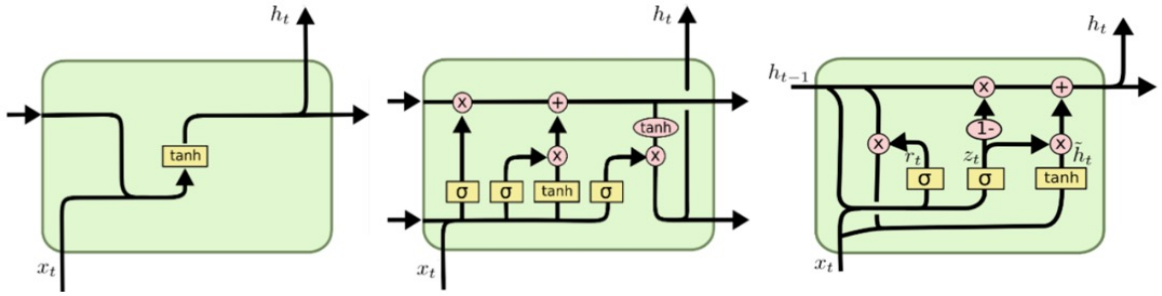
- *CreateAiMLinearOp*
- *RunAiMLinearOp*

$$sigmoid\left(\begin{bmatrix} w^{11} & w^{12} & w^{13} & w^{14} \\ w^{21} & w^{22} & w^{23} & w^{24} \\ w^{31} & w^{32} & w^{33} & w^{34} \end{bmatrix} \times \begin{bmatrix} x^1 \\ x^2 \\ x^3 \\ x^4 \end{bmatrix} + \begin{bmatrix} b^1 \\ b^2 \\ b^3 \end{bmatrix}\right) = \begin{bmatrix} s^1 \\ s^2 \\ s^3 \end{bmatrix}$$

- *CreateAiMSeqOp*
- *RunAiMSeqOp*



- *CreateAiMRNNOp*
- *RunAiMRNNOp*
- *CreateAiMLSTMOp*
- *RunAiMLSTMOp*
- *CreateAiMGRUOp*
- *RunAiMGRUOp*



Basic RNN *LSTM* *GRU*

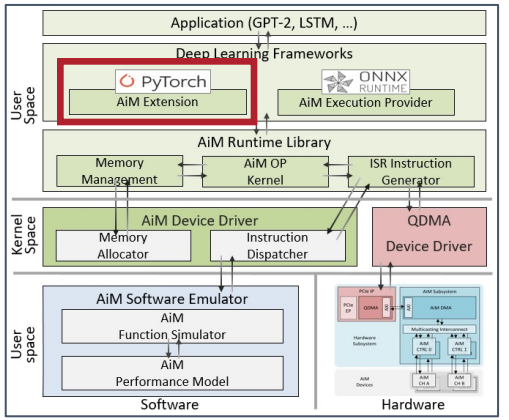
AiM OP Kernel API

- *CreateAiMCustomOp*
- *RunAiMCustomOp*

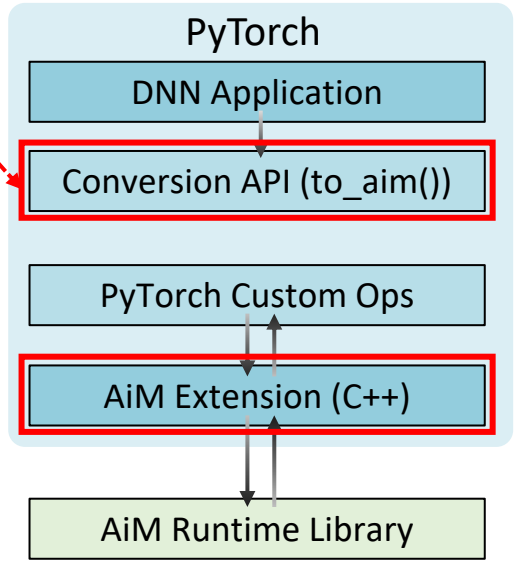
⋮

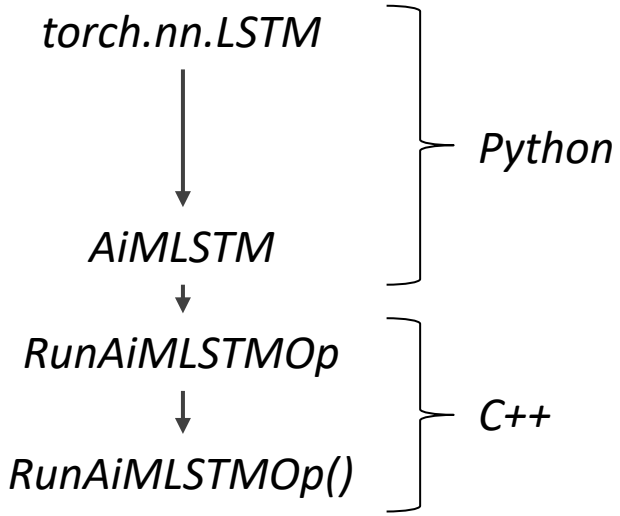
AiM Integration on PyTorch

- *The framework provides abstraction functions for various PIM operations and easy programmability for developers.*
- *If users simply apply "to_aim" API to any PyTorch operations, the runtime library will convert them to the operator-level AiM APIs defined in the runtime library.*



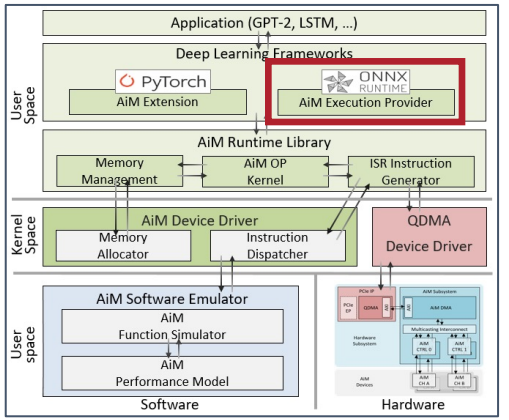

```
model_cpu = torch.nn.LSTM(input_size, hidden_size,
                          num_layers).to(torch.device("cpu"), torch.float32)

from aim_pytorch.utils import to_aim
model = to_aim(model_cpu)
...
output, (hidden, cell) = model(X)
```



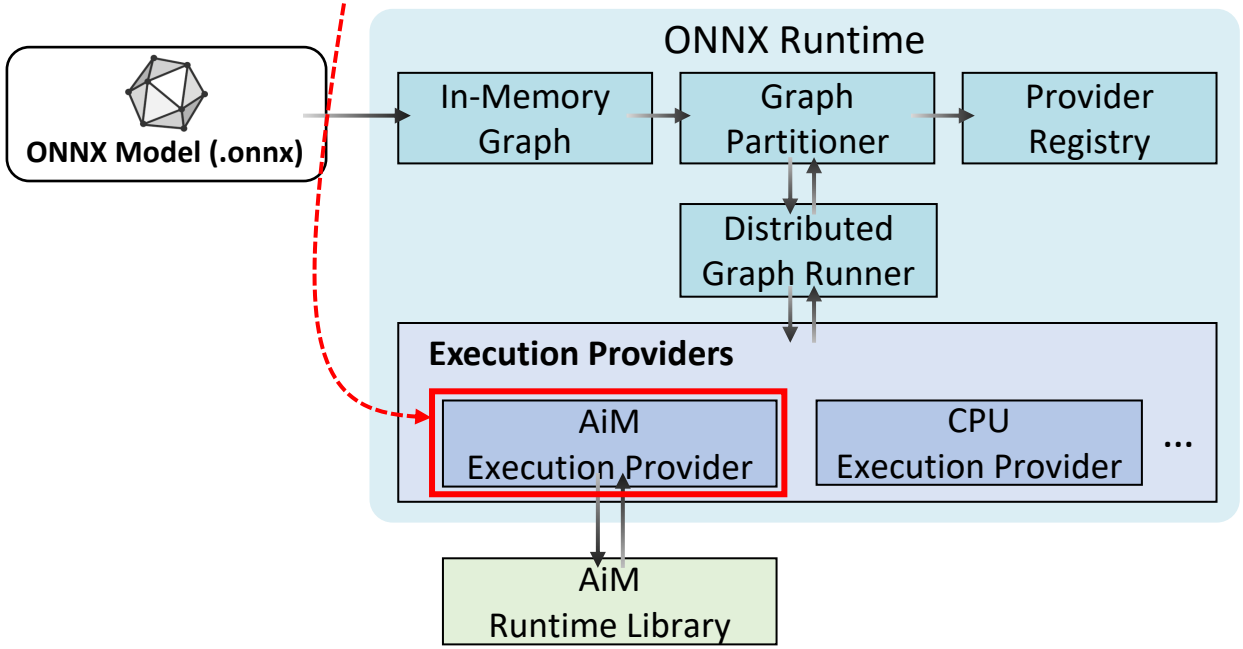


AiM Integration on ONNX Runtime

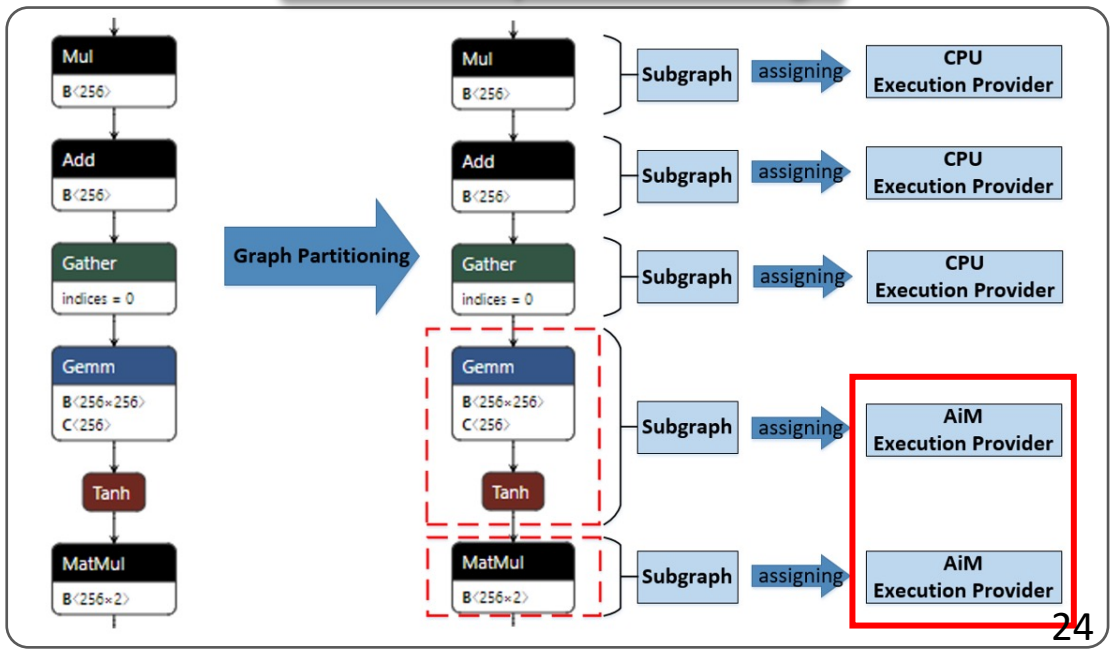
- *In order to run ONNX Runtime applications on AiM, developers simply add "AiMExecutionProvider" to the "EP_List", which will make the AiM execution provider the default provider, and some nodes in the ONNX graph will be offloaded to AiM.*




```
onnx_file = 'my_model.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}, )
```

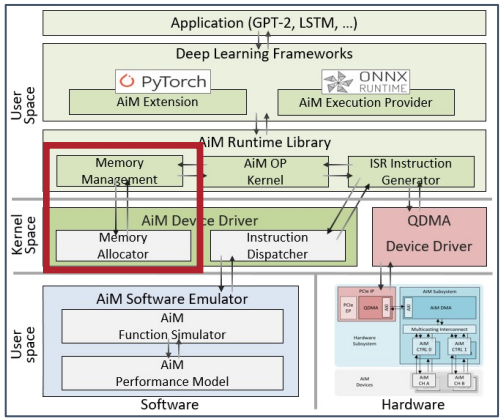


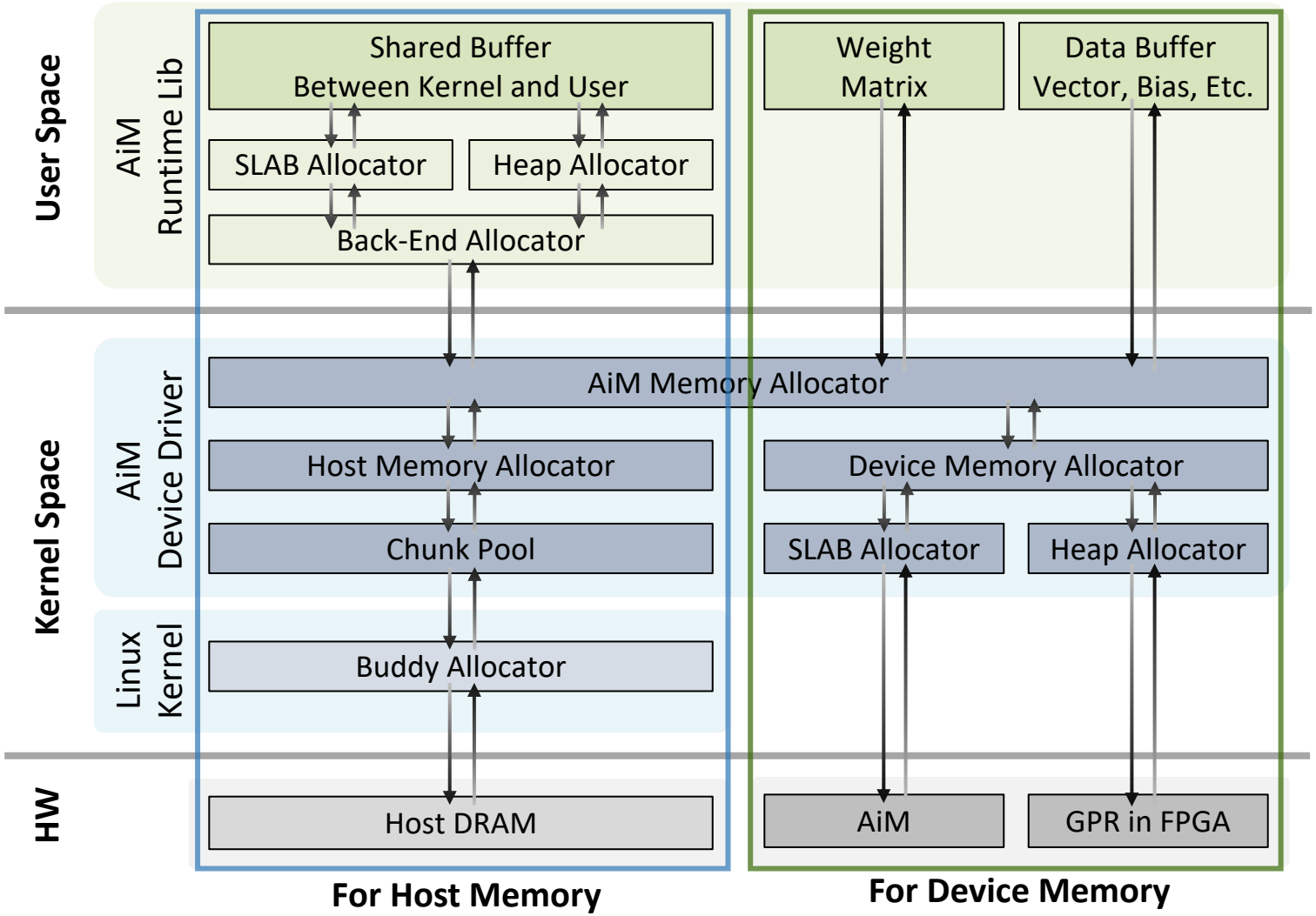
ONNX Graph Partitioning



AiM Memory Allocator

- *The memory allocator allocates buffers from the AiM and host DRAM, and manages the virtual address of each buffer for each process. This allocator manages three types of memory (Host DRAM, AiM, and FPGA GPR) and has a hierarchical structure.*



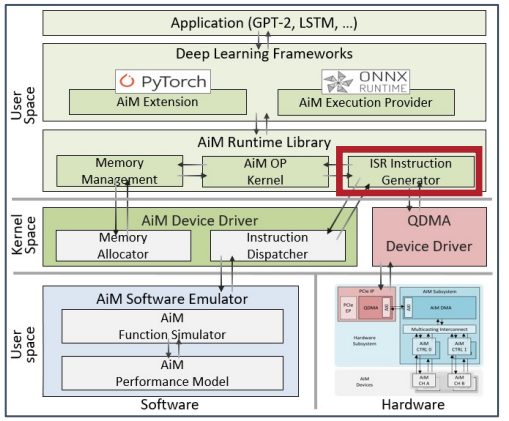


AiM Memory Allocator API

- *AiMMalloc*
- *AiMMatrixMalloc*
- *AiMMemcpy*
- *AiMFree*

AiM Host (ISR) Instruction Generator

- *The process of generating the host instructions and dispatching them to the hardware.*



Only once at Application Initialization Time

DL Frameworks (e.g. PyTorch)

$f($ Weight Matrix $\times$ vector $+$ Bias $) =$ Result

1 *CreateAiMLinearOp*

Runtime Lib

Weight Matrix		Vector	Bias	Result
Tile0 (Job0)	Tile2 (Job2)	V0	B0	R0
Tile1 (Job1)	Tile3 (Job3)	V1	B1	R1

2 ***Generate ISR Instruction Stream***

at Every Inference Time

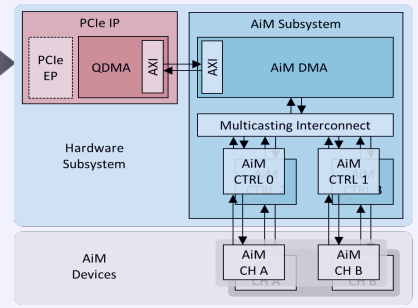
3 *RunAiMLinearOp*

Host (Macro) Instructions

Tile0	Tile1	Tile2	Tile3
ISR_WR_BIAS (B0)	ISR_WR_BIAS (B1)	ISR_WR_BIAS (B0')	ISR_WR_BIAS (B1')
ISR_WR_GB (V0)	ISR_MAC_ABK	ISR_WR_GB (V1)	ISR_MAC_ABK
ISR_MAC_ABK	ISR_RD_MAC (B1')	ISR_MAC_ABK	ISR_AF
ISR_RD_MAC (B0')		ISR_AF	ISR_RD_AF (R1)
		ISR_RD_AF (R0)	

Host DRAM

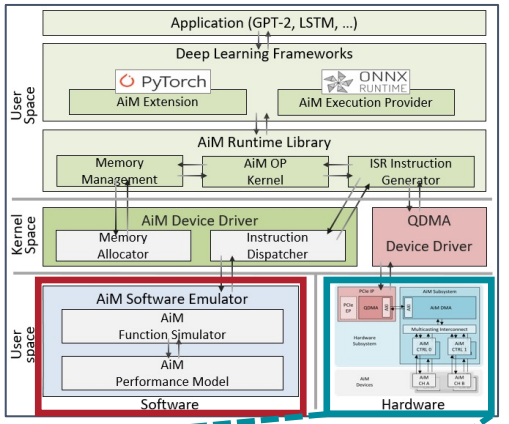
4 ***DMA***



Hardware

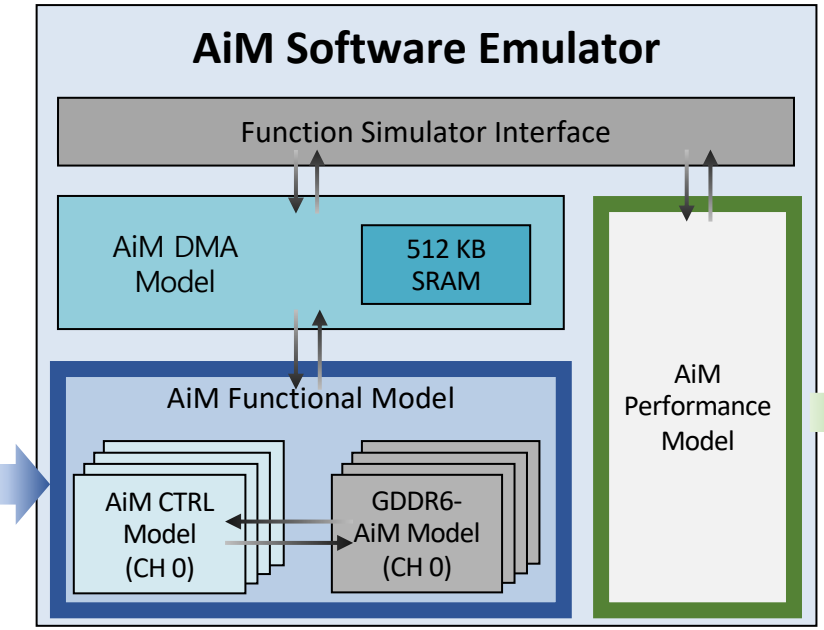
AiM Software Emulator

- *With the AiM performance model, users can estimate not only the total execution time of AI applications but also the execution time of each individual host instruction.*
- *The other benefit of using the software emulator is the flexibility of the hardware.*



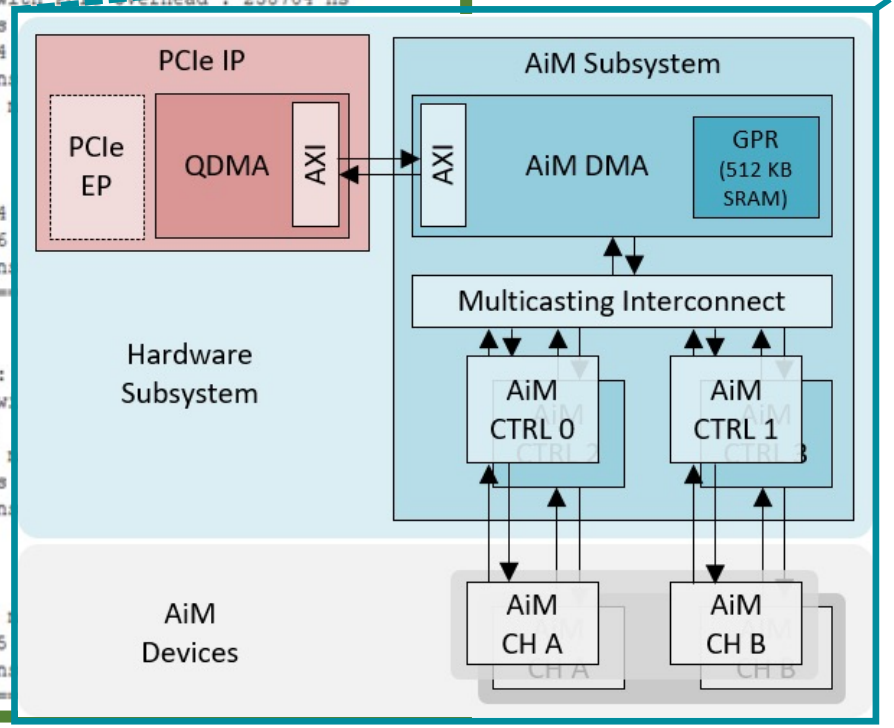

```
[memctrl]
addrmapping="RoRaBaCoCh"

[mem]
burstsize=32
rowsize=2048
row=524288
bank=16
bg=4
pseudo=1
chip=1
rank=1
channel=4

[aim]
datatype="bf16"
gbuffer=2048
```

Configuration File



AiM Software Emulator

```
==========================================================
== [OP Kernel] Estimated Performance (PID : 43809) ==
==========================================================
== Timing Parameters from Python Analytic Model ==
==========================================================
channel : 4
Gbps : 2
total latency : 246704 ns
total latency with PCIe Overhead : 258704 ns
tWRGB : 2240 ns
tWRBIAS : 24464
tMAC : 145408 n
tRDMAC : 23552
tAF : 0 ns
tRDAF : 0 ns
tEWADD : 0 ns
tsystem : 41984
trefresh : 9056
tPCIe : 12000 n
==========
channel : 4
Gbps : 16
total latency :
total latency w
tWRGB : 352 ns
tWRBIAS : 8894
tMAC : 28928 ns
tRDMAC : 9600 n
tAF : 0 ns
tRDAF : 0 ns
tEWADD : 0 ns
tsystem : 5248
trefresh : 3375
tPCIe : 12000 n
==========
```



Estimated Performance from Analytical Model

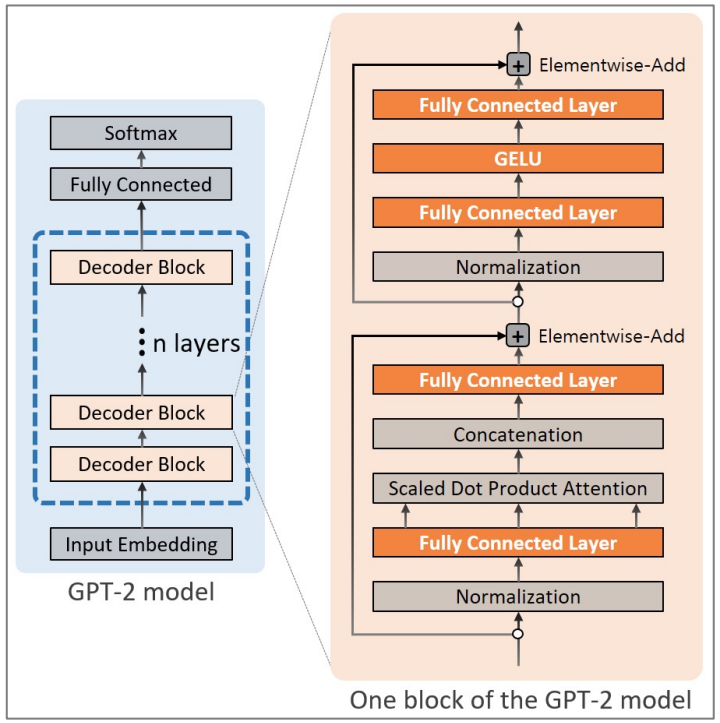
AiM Evaluation Board

CONTENTS

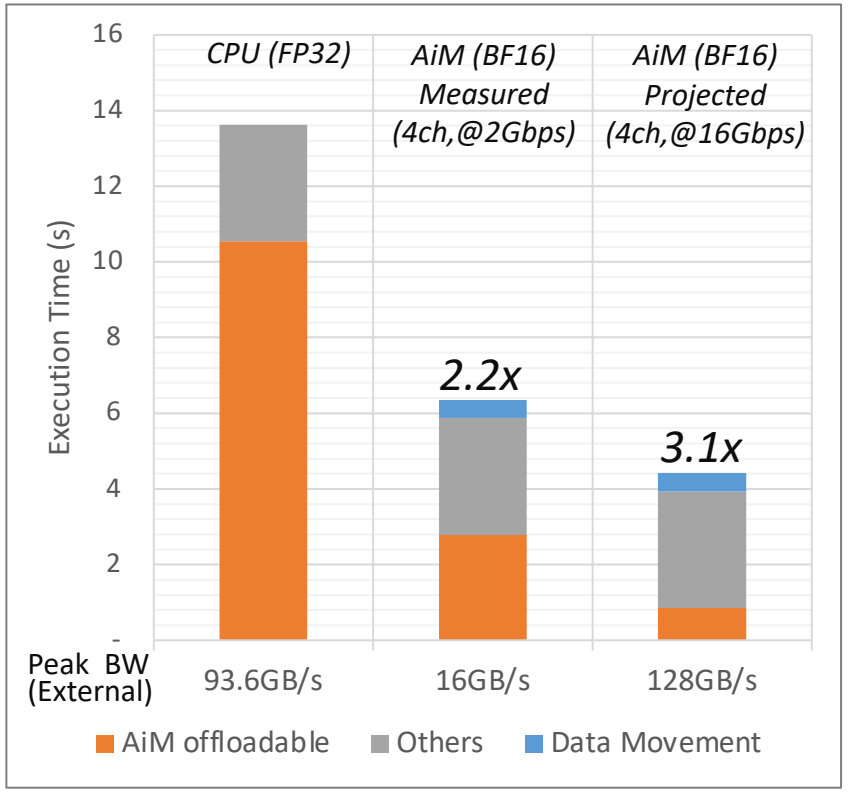
Perfomance Evaluation: GPT-2 and GPT-3

- *Performance Evaluation on GPT-2 and GPT-3 (Memory intensive workload)*
- *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*
- *By having a more memory intensive workload, AiM will be more effective*

GPT-2 Model Architecture

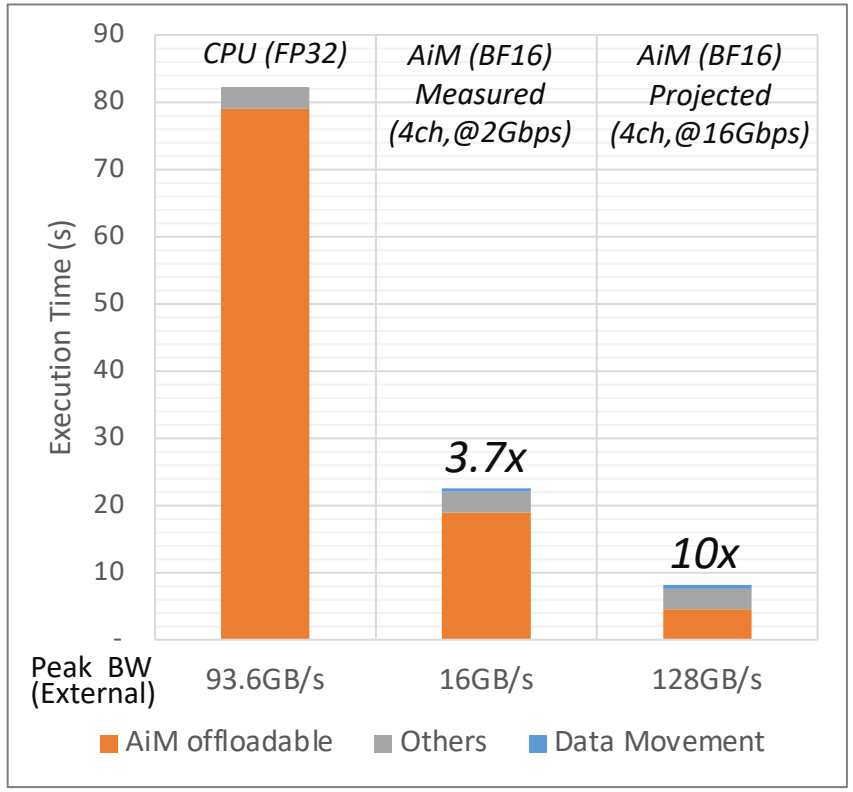


*Performance on **GPT-2 XL***



Parameter Size : 1.5B

*Performance on **GPT-3 13B Configuration***



Parameter Size : 13B

SK hynix can provide the following items for your research

- ✓ *AiM SDK (including AiM Software Emulator)*
- ✓ *[Optional] AiM FMC card × 2*

Please contact us at **SKhynix_PIM@skhynix.com**

AiM Platform Distribution

SK hynix

References

- **Circuit-level**
  - *[ISSCC'22] 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*
    - *ISSCC'22 demo. (short video): https://www.youtube.com/watch?v=3LbvwrJFYoA*
  - *[JSSC'23] A 1ynm 1.25V 8Gb 16Gb/s/Pin GDDR6-Based Accelerator-in-Memory Supporting 1TFLOPS MAC Operation and Various Activation Functions for Deep Learning Application*

- **Architectures/Software stack**
  - *[Hot Chips'22] System Architecture and Software Stack for GDDR6-AiM*
  - *White papers (released with AI Hardware Summit'22):*
    - *(1) Accelerator-in-Memory: SK hynix's GDDR6-based Processing-in-Memory Device*
    - *(2) Accelerator-in-Memory Platform: Hardware Perspective*
    - *(3) Accelerator-in-Memory Platform: Software Perspective*
    - *(From https://product.skhynix.com)*