Exploring Processing-in-Memory for memory-bound applications in computing systems

Byeongho Kim

Samsung Electronics

DRAM Design Team 1

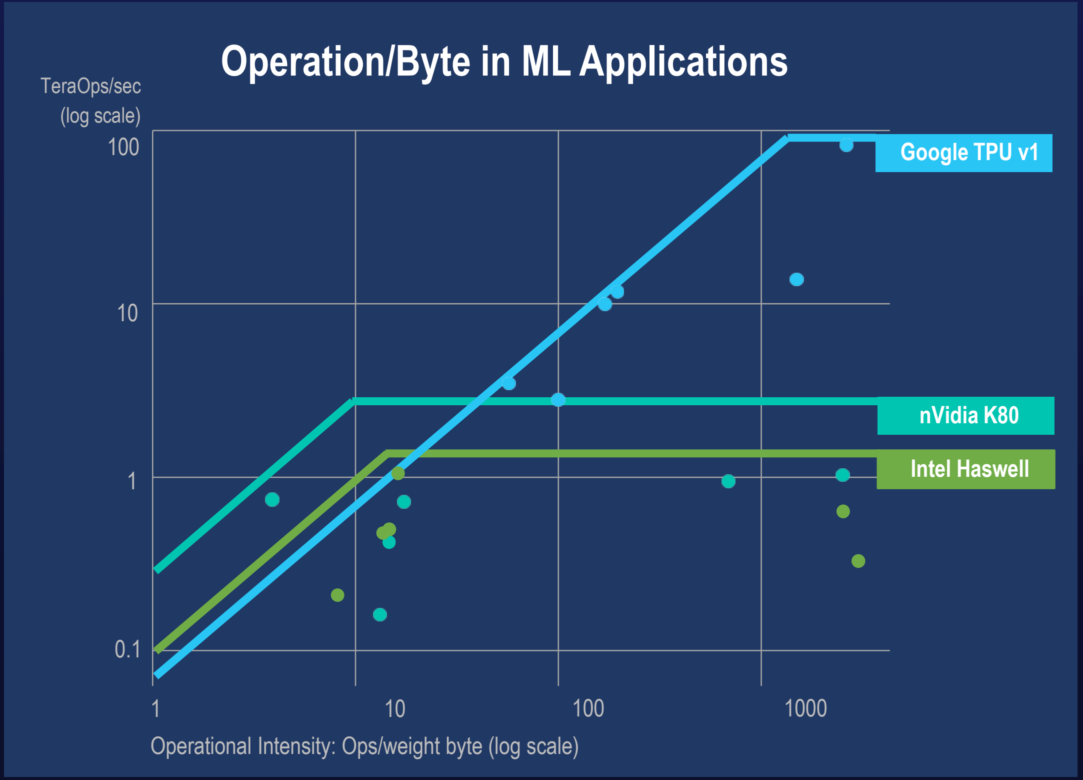
Memory Technology Trends and Challenges

AI system challenges

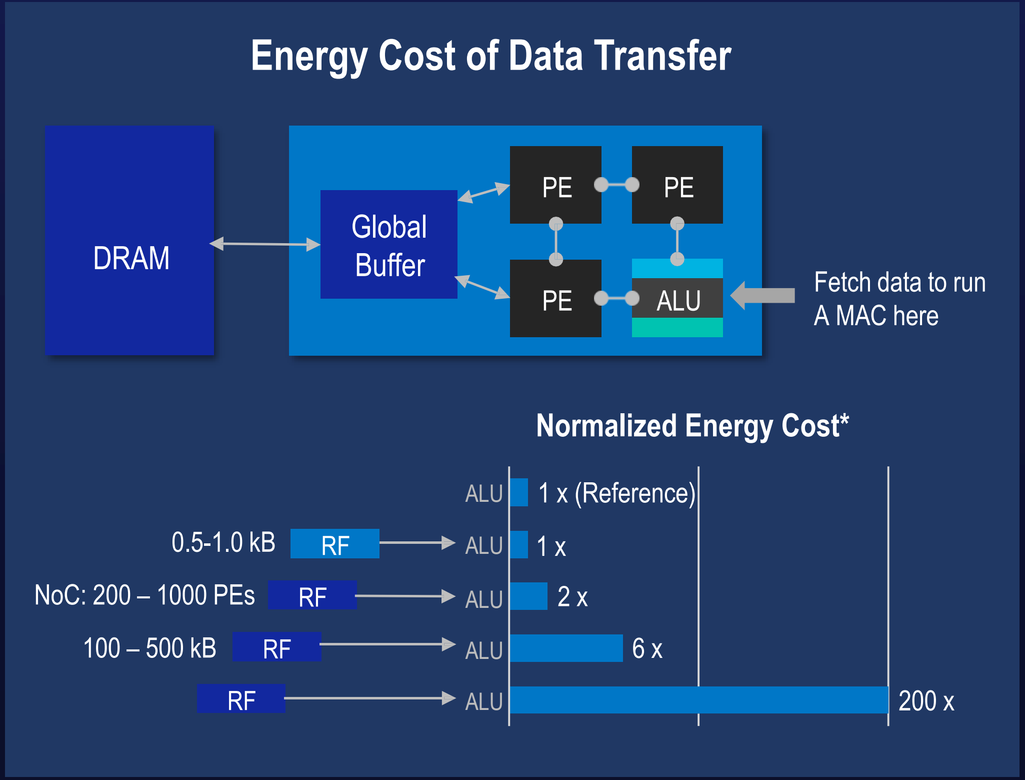
Memory bandwidth-starved processors (memory-bound application)

High data movement energy from(to) memory

Low memory bandwidth utilization (random, sparsity)



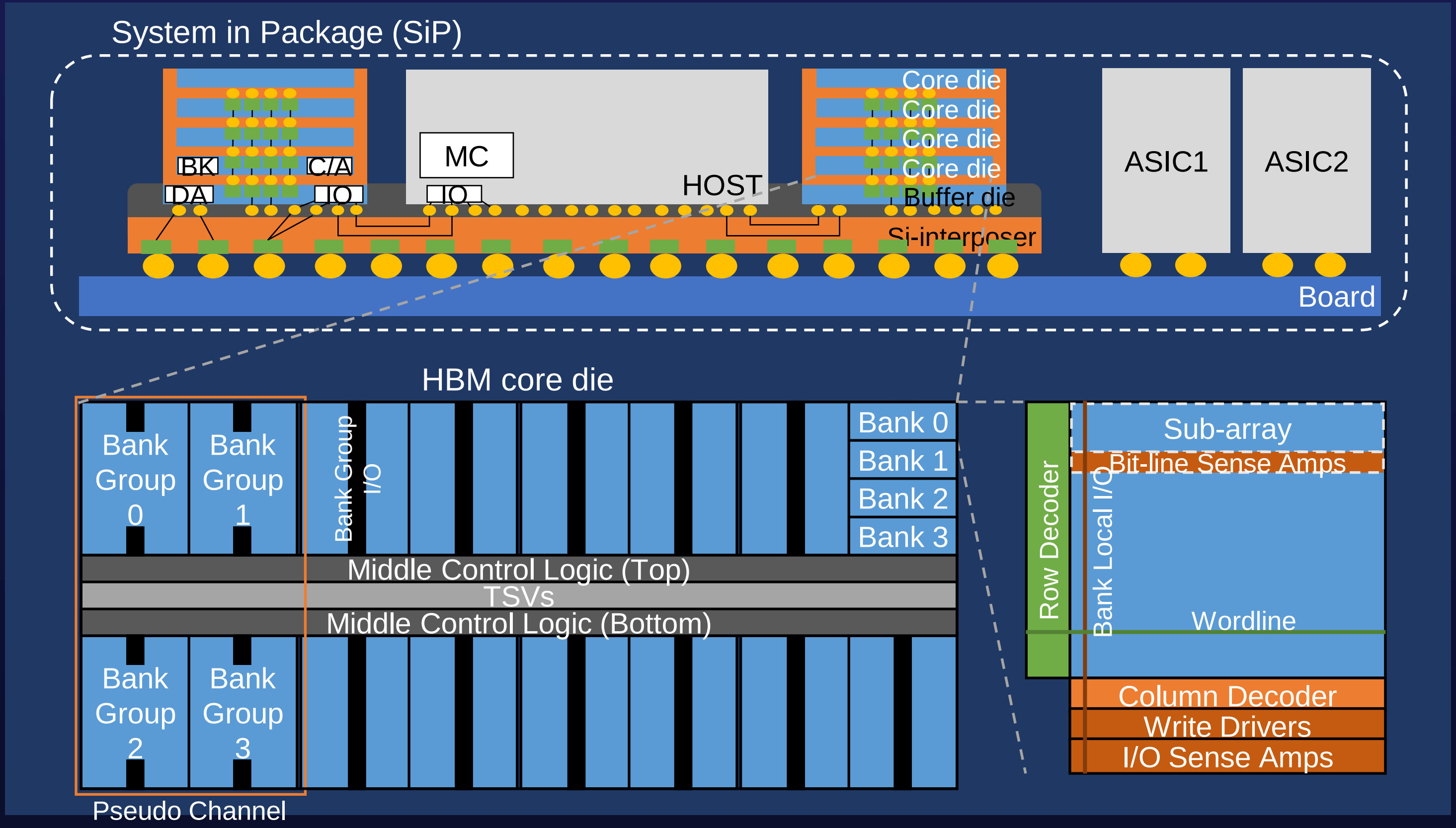
Source : In-Datacenter Perf Analysis of a Tensor Processing Unit @ 2017, ISCA



Source: * Y.-H. Eyeriss, 2016 ISCA

High Bandwidth Memory, but still need more

High bandwidth memory (HBM) was introduced to provides higher pin density with silicon interposer and satisfies the demand for the off-chip memory bandwidth.

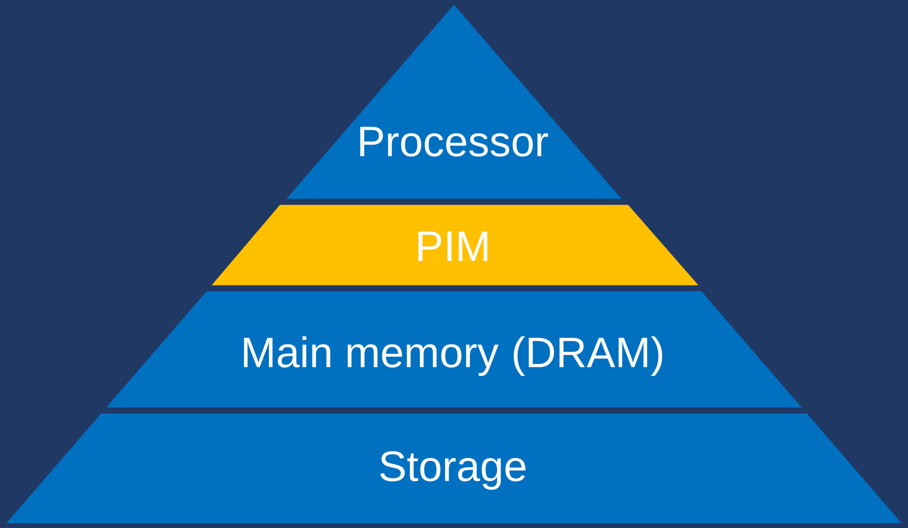


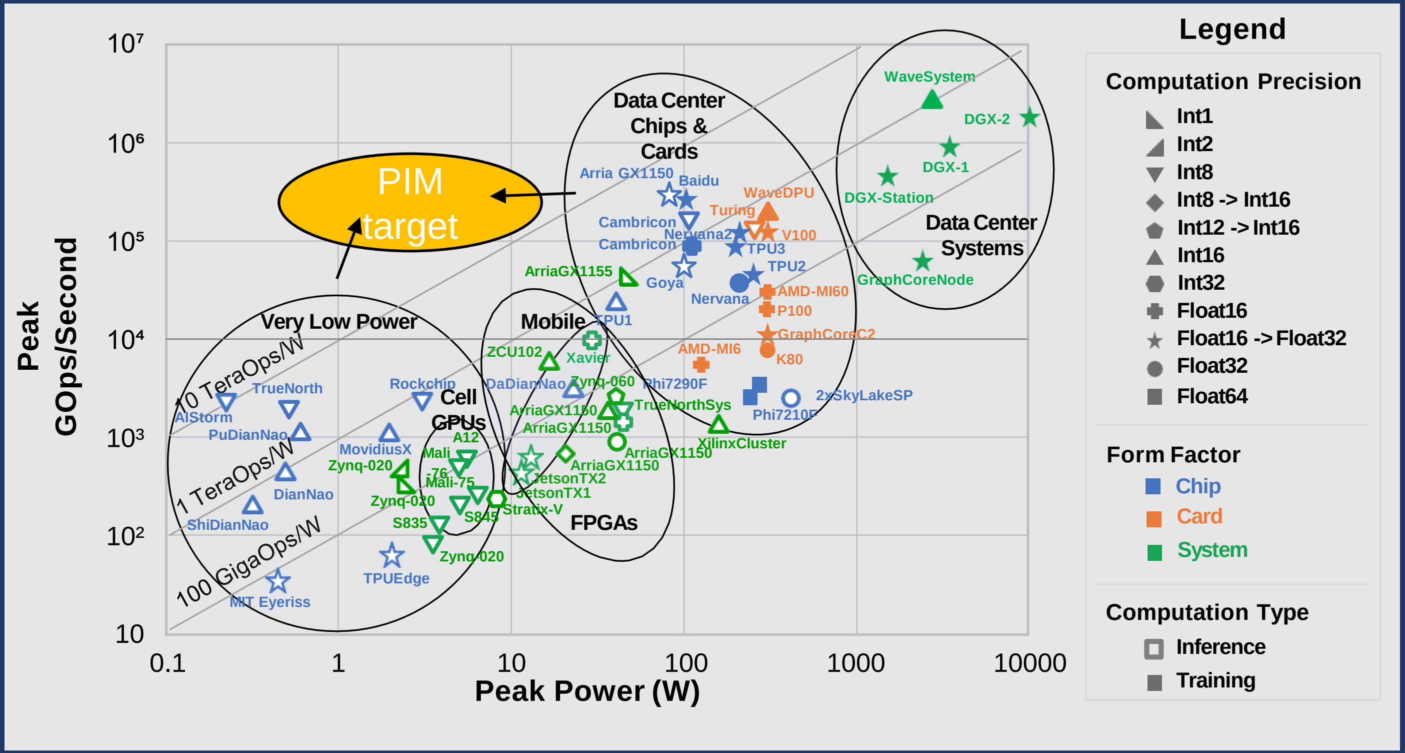
HBM helped breaking the memory wall, but system still requires more bandwidth for emerging applications

PIM: Renewed interest

Processing-In-Memory (PIM)

- Fill the performance gap and deliver energy-efficient solutions [Hotchips 16, Samsung]
- PIM provides high ops/second and low power [Survey and Benchmarking of Machine Learning Accelerators]





Challenges in developing commercial PIM

Driven by costs and hegemony struggle among industry stakeholders

- **Processor design companies:** We don't have time and resource to change memory subsystems of our processors for unproven technologies, especially something that …
- **DRAM manufacturers:** We don't want to change DRAM core design for PIM as it has been optimized over decades and thus hard and expensive to change
- **Customers:** We don't want to change our application code just for PIM. We want homogeneous systems, i.e., PIM also need to serve as standard DRAM

Then, we propose PIM architecture without changes on the hosts by keeping current JEDEC interface and timing parameters, same memory organization as previous DRAM, and provides full software stacks and several applications to prove on a HBM2 based system

Introduction to HBM-PIM Architecture

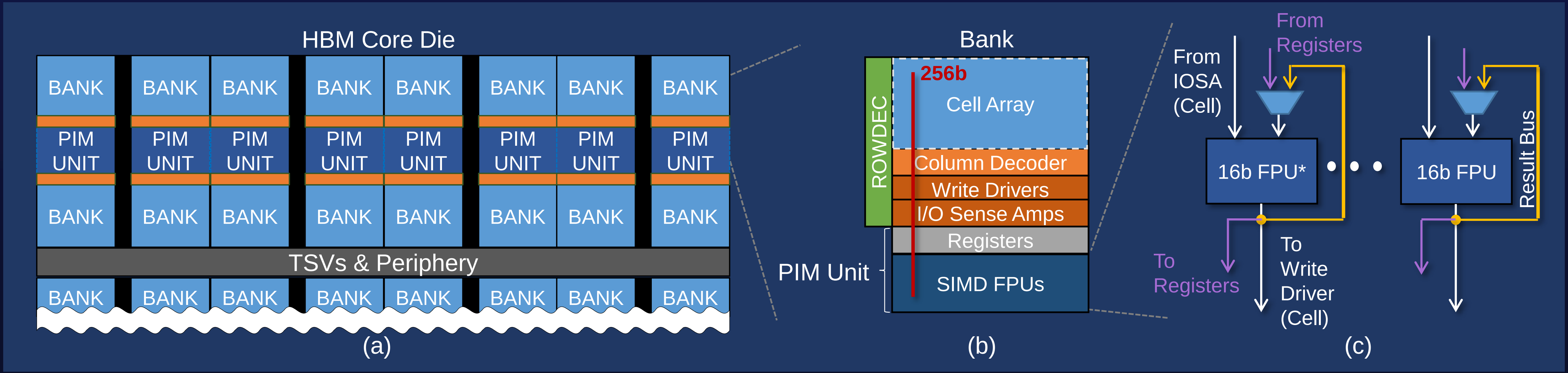
Overview of PIM architecture

High on-chip compute bandwidth w/o changing DRAM core circuitry

- Place SIMD FPU at bank IO boundary
- Exploit bank-level parallelism: access multiple banks/FPUs in a lockstep manner

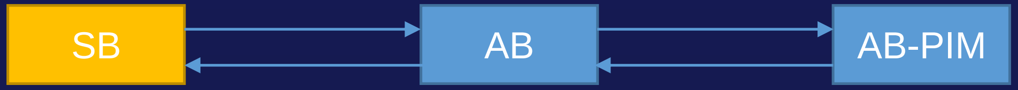
Expose high on-chip bandwidth of standard DRAM to processors

- Build on industry standard DRAM interfaces and preserve deterministic DRAM timing
- i.e., a DRAM RD/WR command triggers execution of a PIM instruction

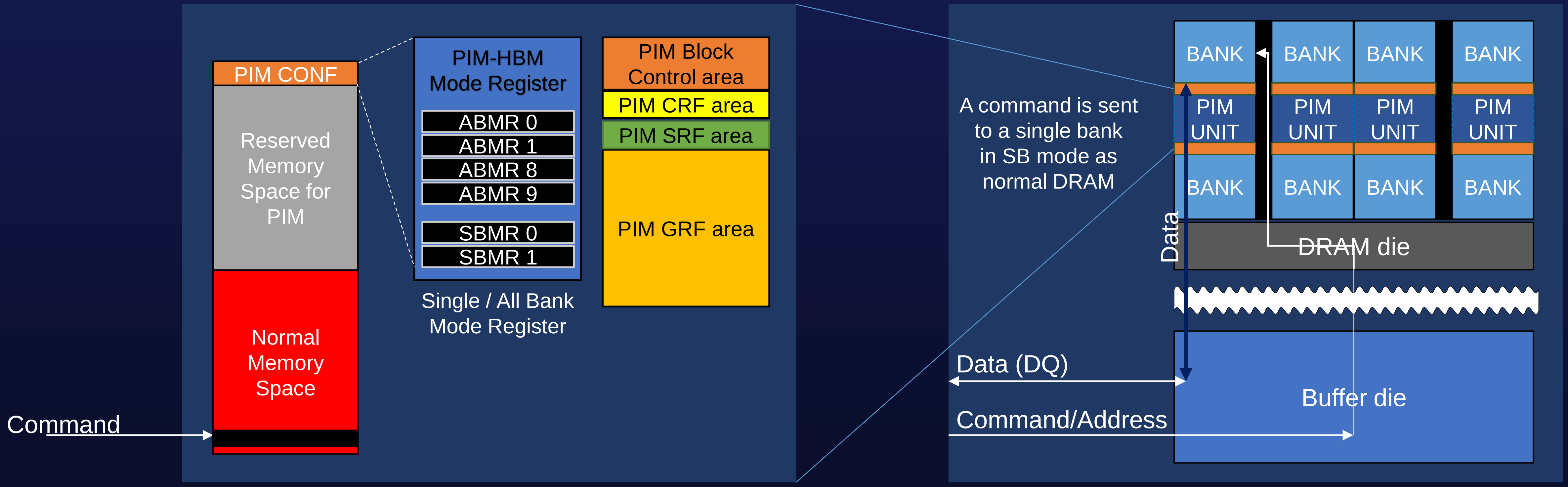


HBM-PIM: Exploiting bank-level parallelism

Single Bank (SB) mode:



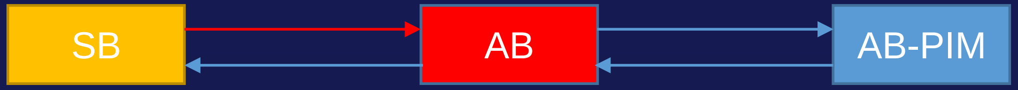
- HBM-PIM works exactly same as normal DRAM (timing parameters, commands and address) but cannot access reserved space for HBM-PIM



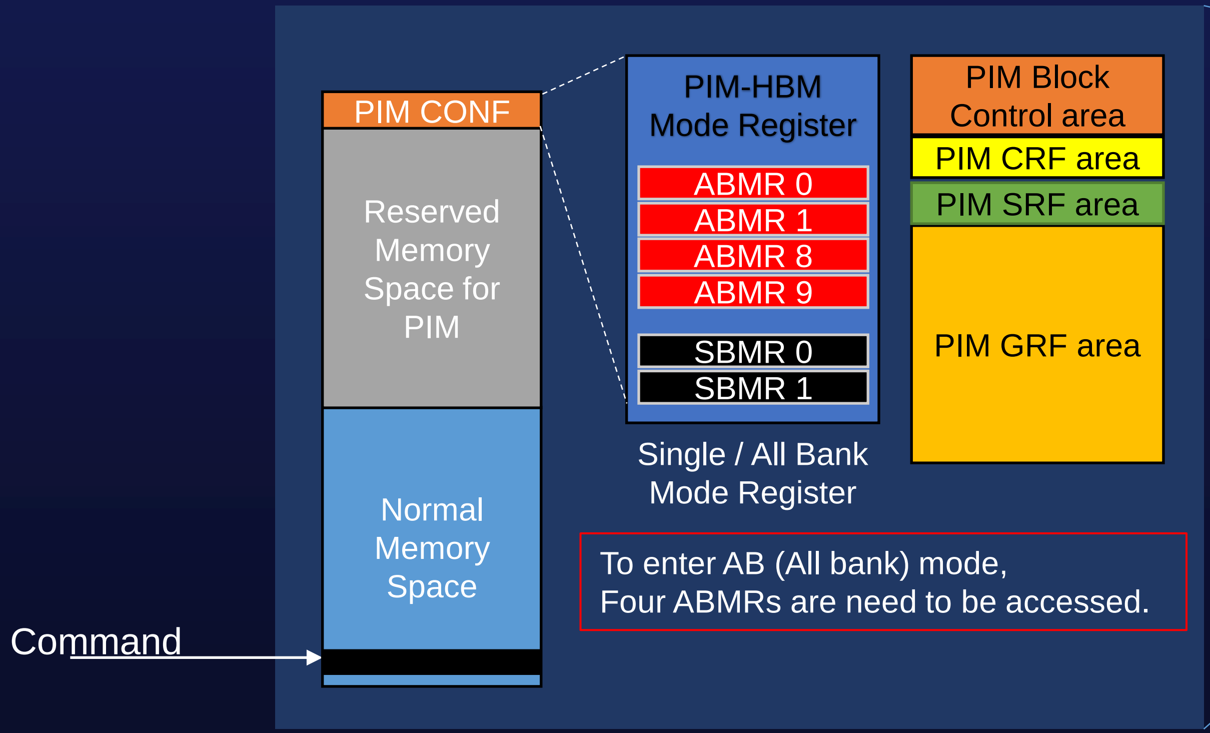
Address targets a specific bank and operates command

HBM-PIM: Exploiting bank-level parallelism

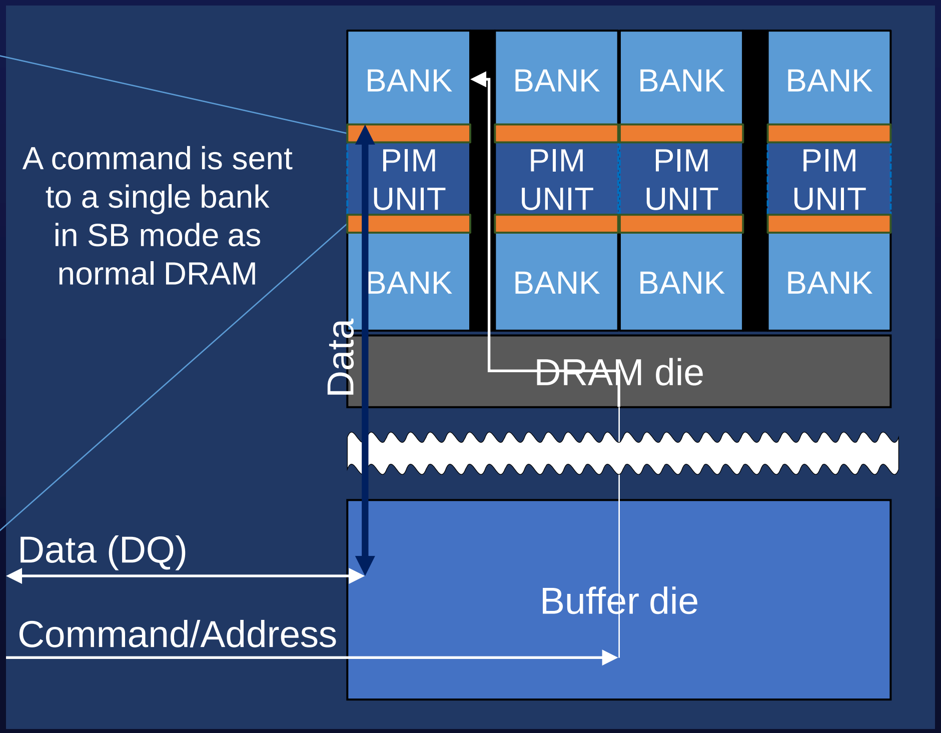
Single Bank (SB) mode:



- HBM-PIM works exactly same as normal DRAM (timing parameters, commands and address) but cannot access reserved space for HBM-PIM



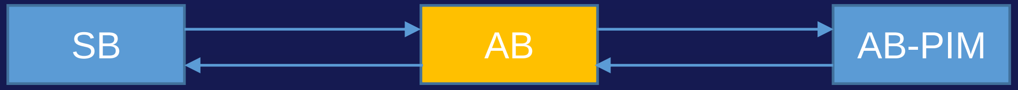
To enter AB (All bank) mode,
Four ABMRs are need to be accessed.



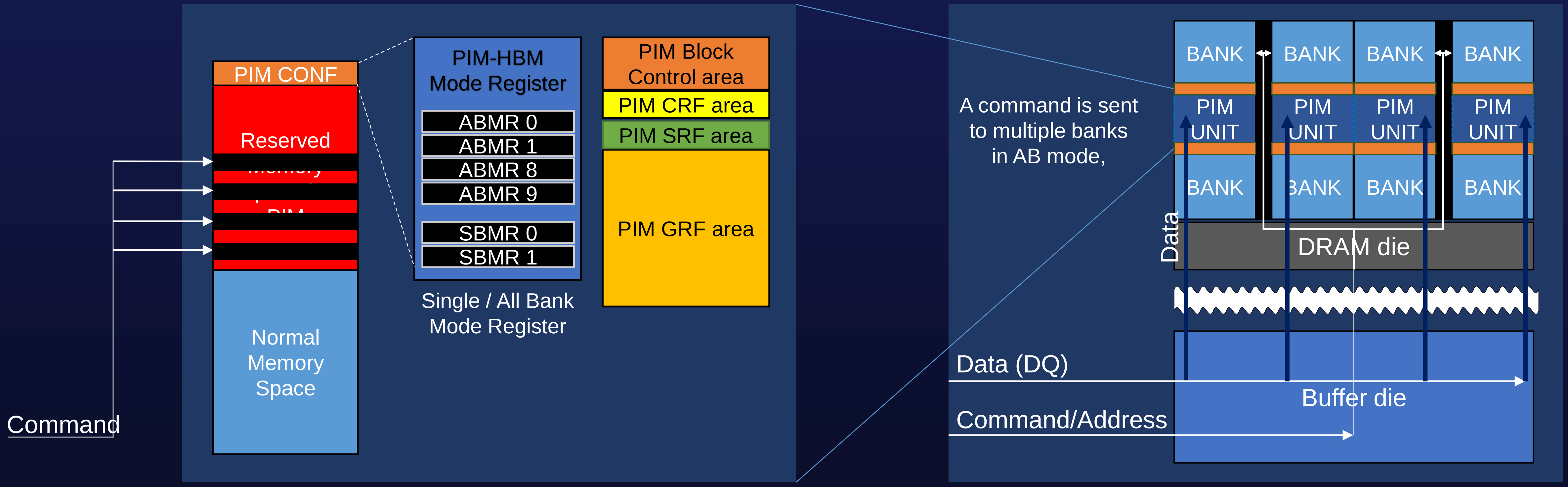
Address targets a specific bank and operates command

HBM-PIM: Exploiting bank-level parallelism

All Bank (AB) mode:



- All banks (data) / PIM units are working at the same time by one command (i.e., ACT, WR, RD, PRE)

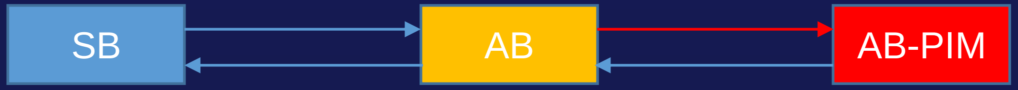


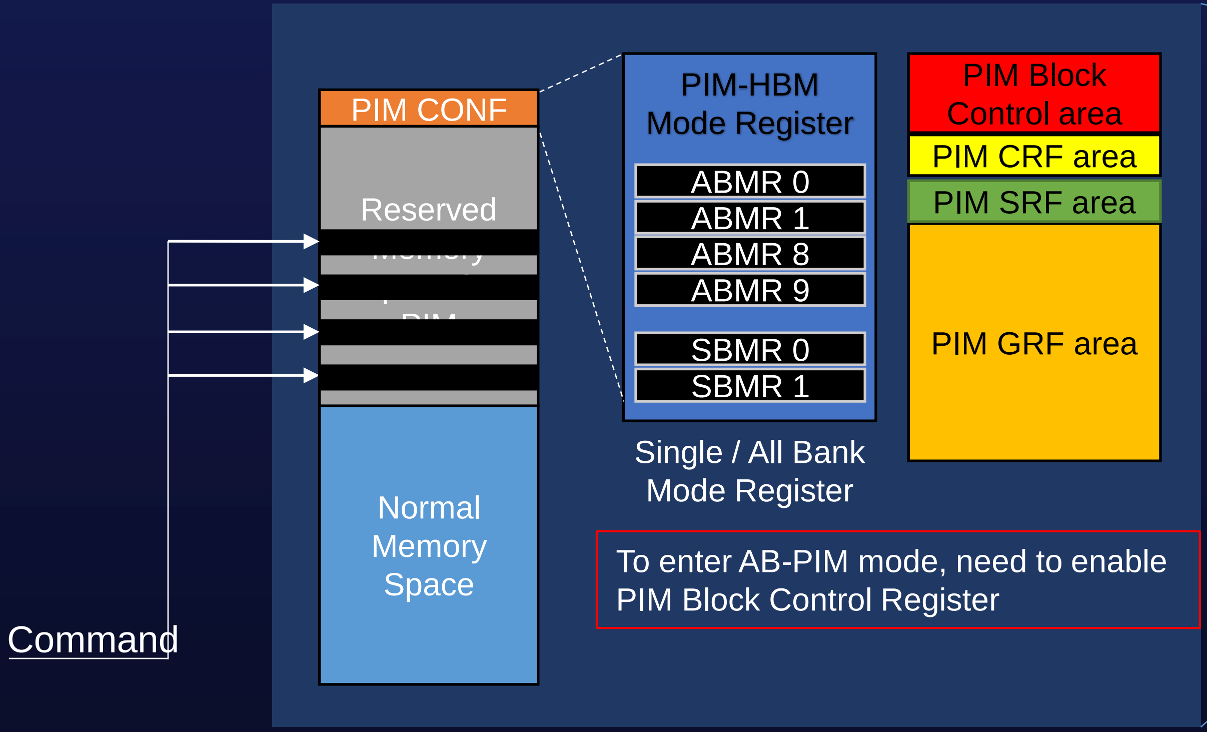
Address targets a specific bank and operates command

HBM-PIM: Exploiting bank-level parallelism

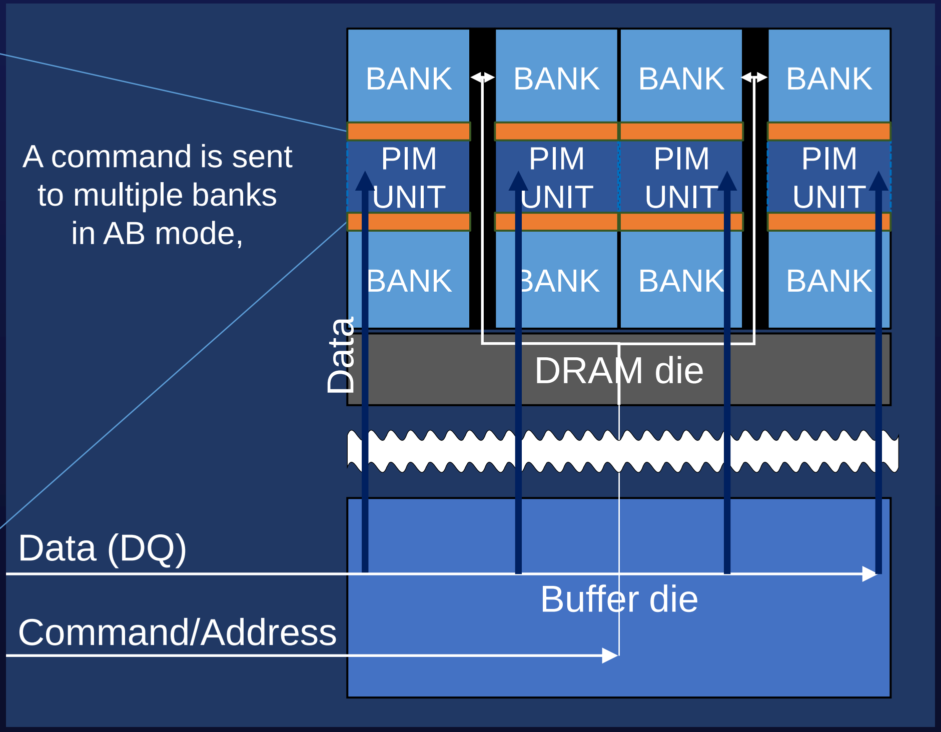
All Bank (AB) mode:

- All banks (data) / PIM units are working at the same time by one command (i.e., ACT, WR, RD, PRE)



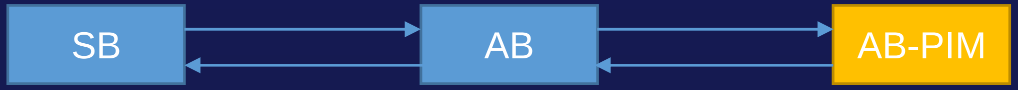


To enter AB-PIM mode, need to enable PIM Block Control Register



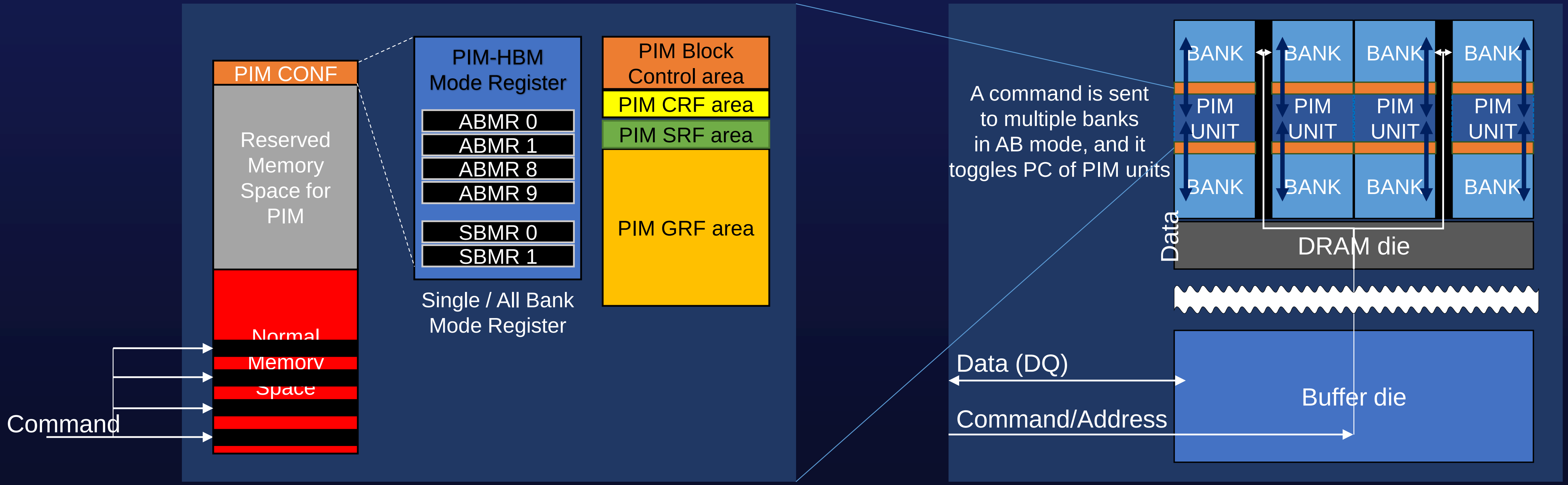
Address targets a specific bank and operates command

HBM-PIM: Exploiting bank-level parallelism



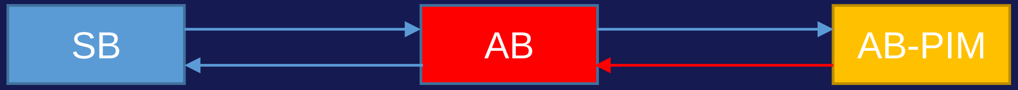
All Bank (AB-PIM) mode:

- All banks (data) / PIM units are working at the same time by one command (i.e., ACT, WR, RD, PRE)
- Same as AB mode, but PIM unit is only enabled in this mode to prevent unexpected data pollution.



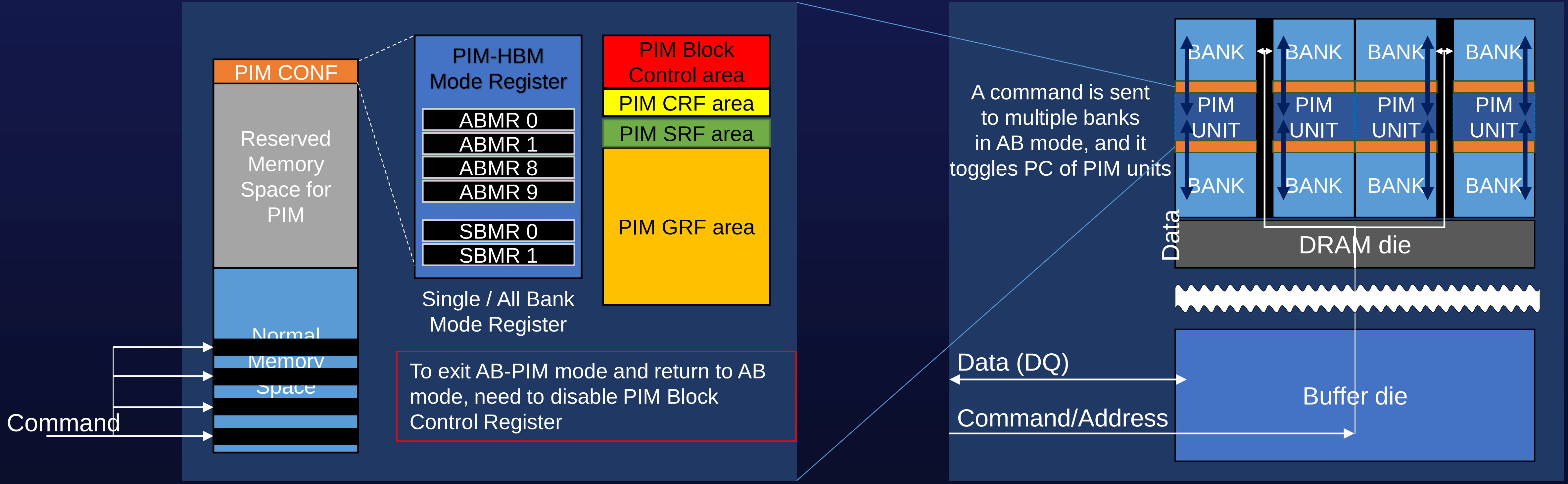
Address targets a specific bank and operates command

HBM-PIM: Exploiting bank-level parallelism



All Bank (AB-PIM) mode:

- All banks (data) / PIM units are working at the same time by one command (i.e., ACT, WR, RD, PRE)
- Same as AB mode, but PIM unit is only enabled in this mode to prevent unexpected data pollution.

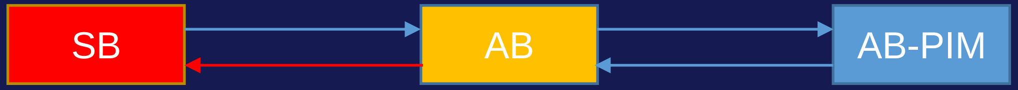


To exit AB-PIM mode and return to AB mode, need to disable PIM Block Control Register

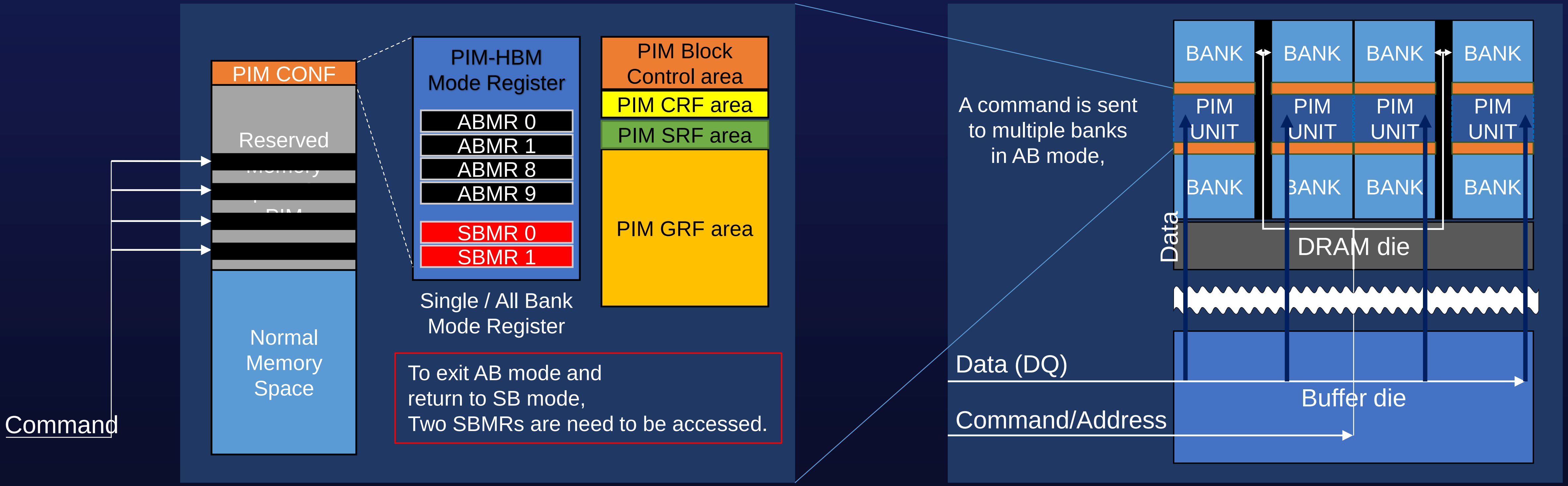
Address targets a specific bank and operates command

HBM-PIM: Exploiting bank-level parallelism

All Bank (AB) mode:



- All banks (data) / PIM units are working at the same time by one command (i.e., ACT, WR, RD, PRE)



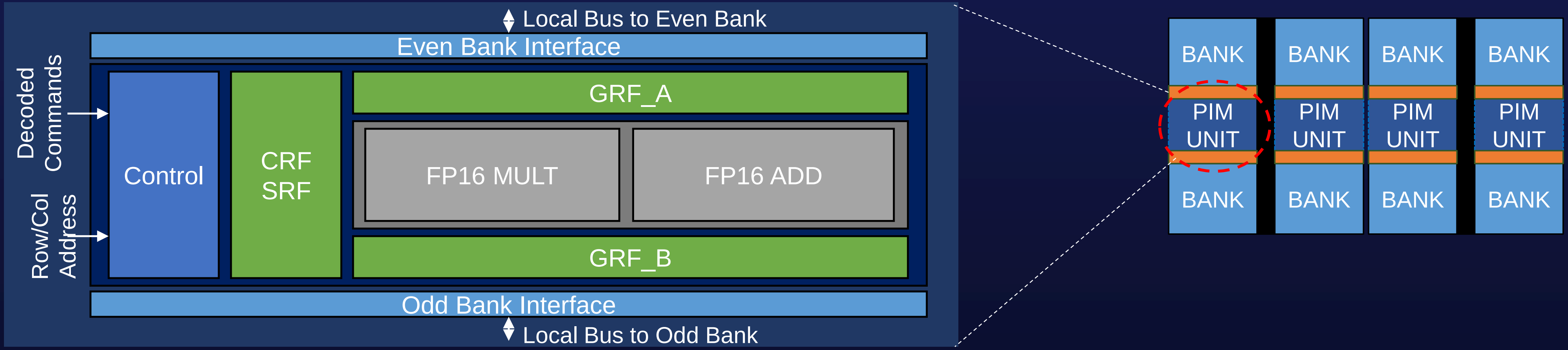
To exit AB mode and
return to SB mode,
Two SBMRs are need to be accessed.

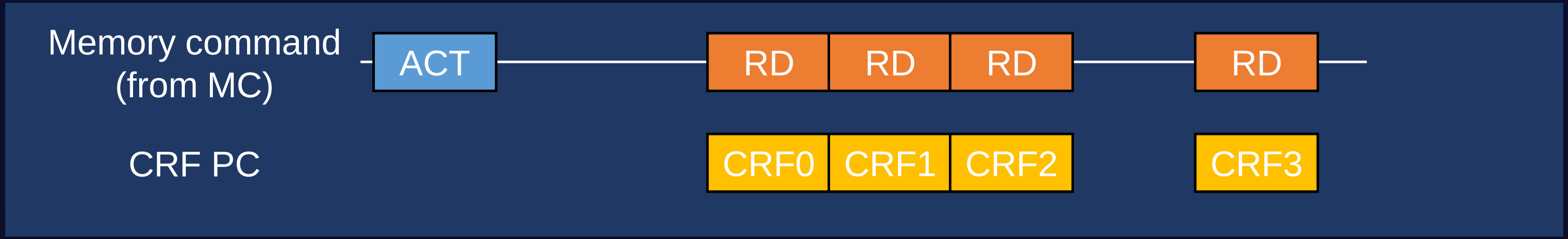
Address targets a specific bank and operates command

HBM-PIM: Microarchitecture

Consist of three major components with DRAM local bus interface:

- A 16-lane FP16 SIMD FPU array: a pair of 16 FP16 multipliers and adders
- Register files: Command, General, and Scalar register files (CRF, GRF, and SRF)
- A PIM unit controller (fetch and decode, controls pipeline signals, forward)





External data (CAS) commands increase CRF PC and read (write) data from to column address at the same time.
From this, PIM unit do not interfere timing parameters.

HBM-PIM : Microarchitecture

Five pipeline stages:

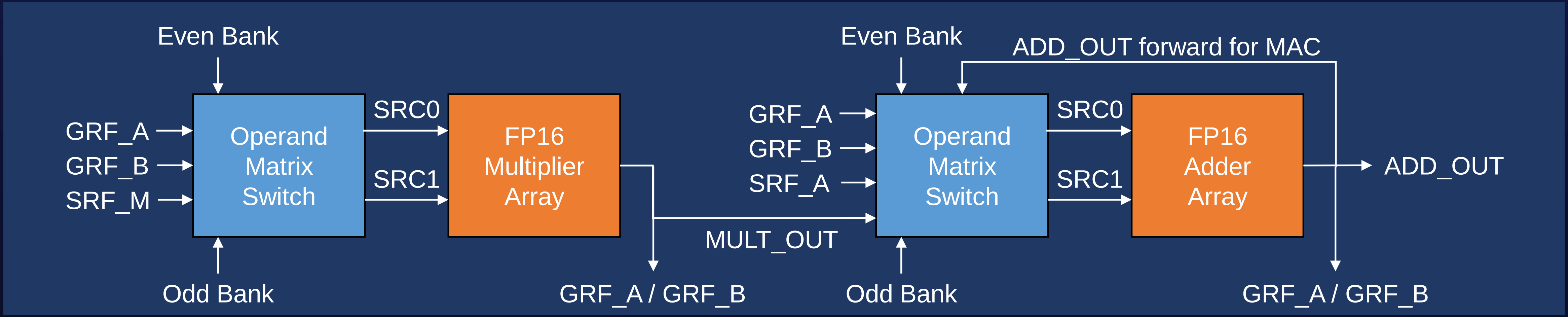
- 1) instruction fetch/decode, 2) data fetch, 3) MULT execution, 4) ADD execution, and 5) write back GRF

Arithmetic opcodes:

- MUL, ADD, MAC, MAD

Operands:

- GRF_A, GRF_B, SRF_M (Mult), SRF_A (Add), Even Bank, Odd Bank



Data path block diagram of PIM execution unit

HBM-PIM : Microarchitecture

Five pipeline stages:

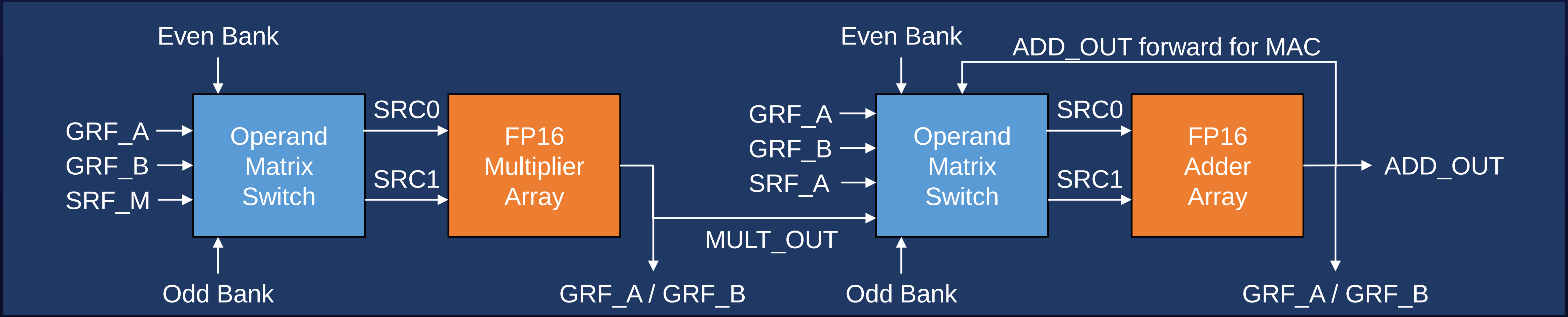
- 1) instruction fetch/decode, 2) data fetch, 3) MULT execution, 4) ADD execution, and 5) write back GRF

Arithmetic opcodes:

- MUL, ADD, MAC, MAD

Operands:

- GRF_A, GRF_B, SRF_M (Mult), SRF_A (Add), Even Bank, Odd Bank

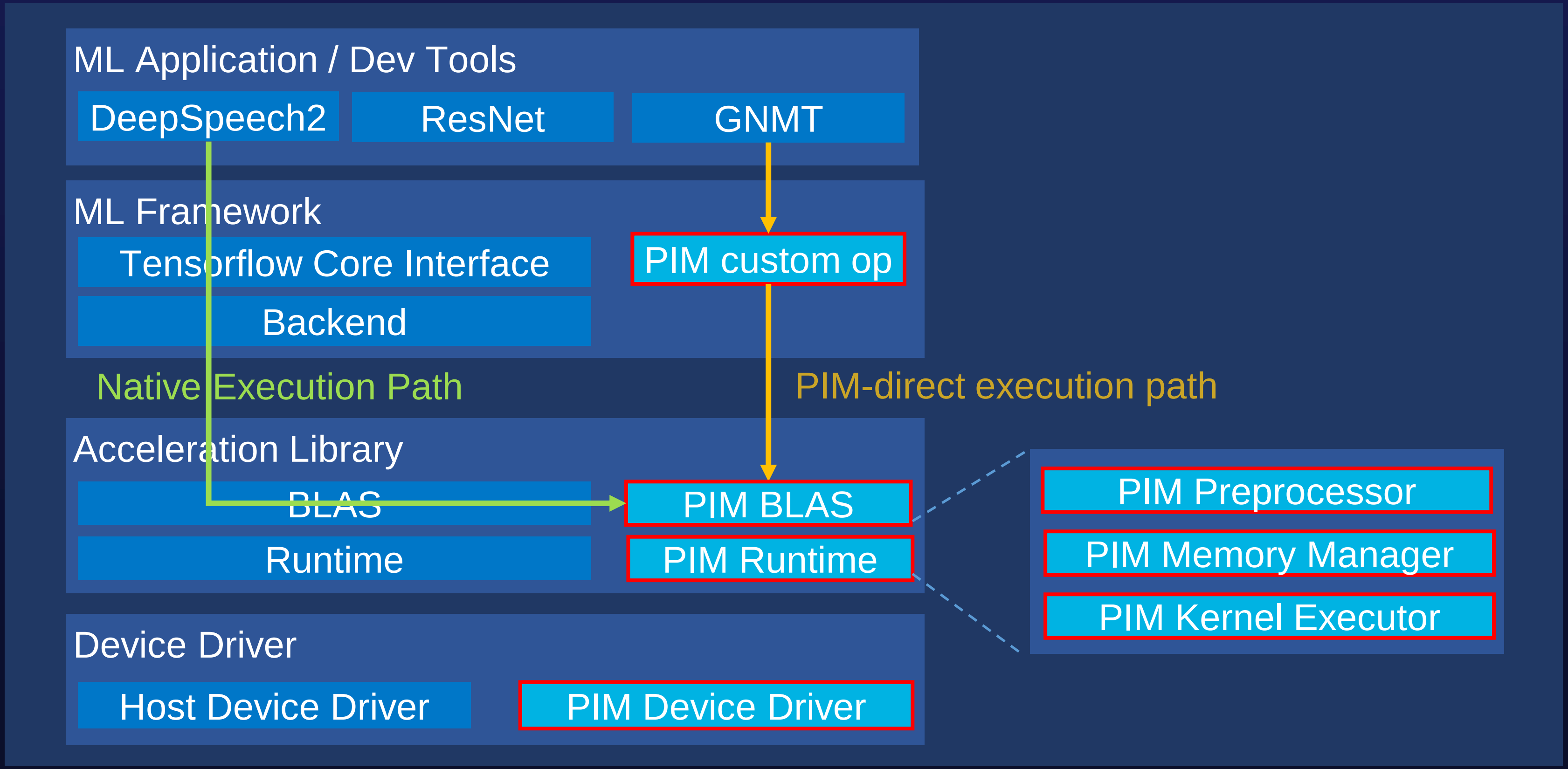


Data path block diagram of PIM execution unit

PIM Software stack

Develop software stack allowing existing ML application source code to run w/o any change

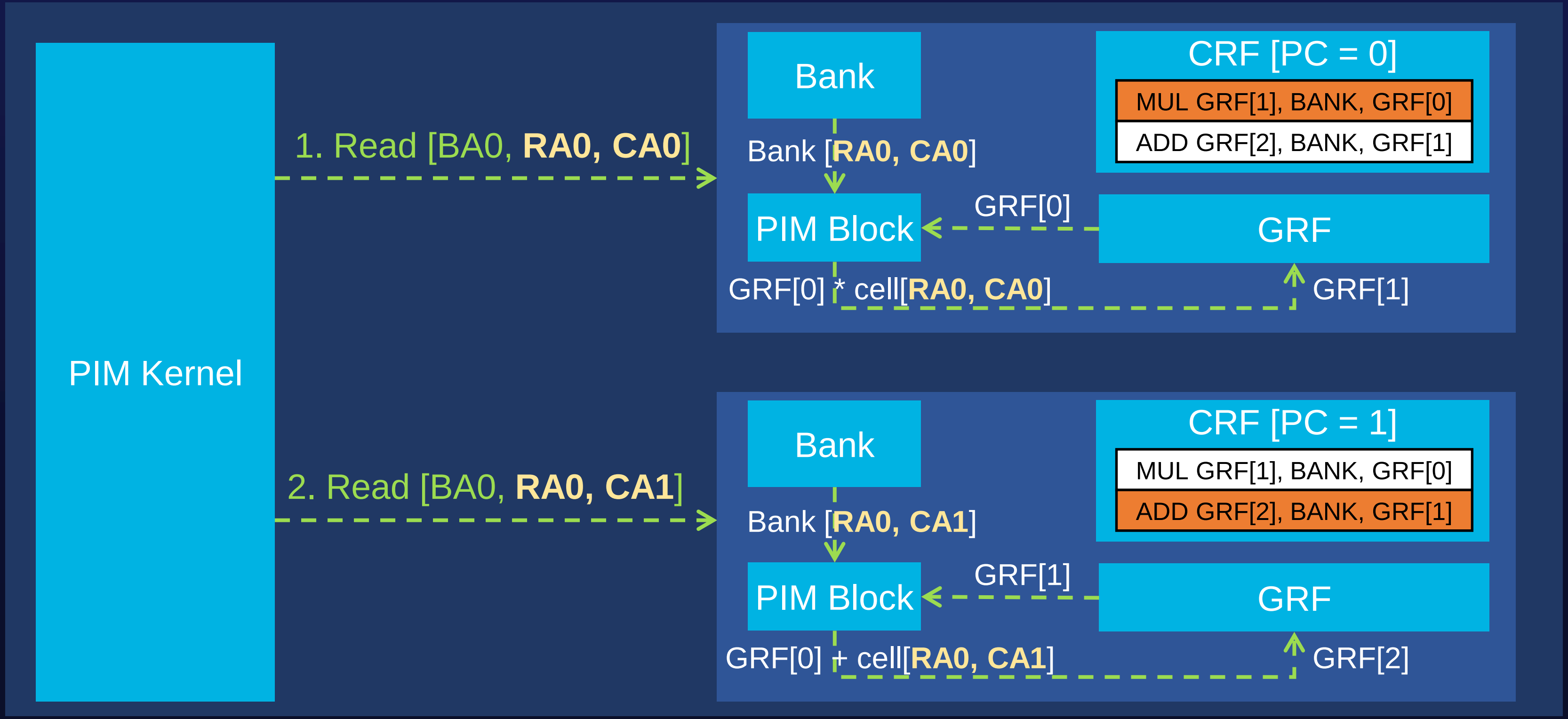
Support PIM-direct execution path: a programmer can explicitly call PIM custom TF operations.



PIM Programming Model

PIM programming consists of CRF programming and PIM kernel programming.

The key of PIM kernel programming is
(1) to generate memory requests to the correct addresses
(2) to send them out in the correct order

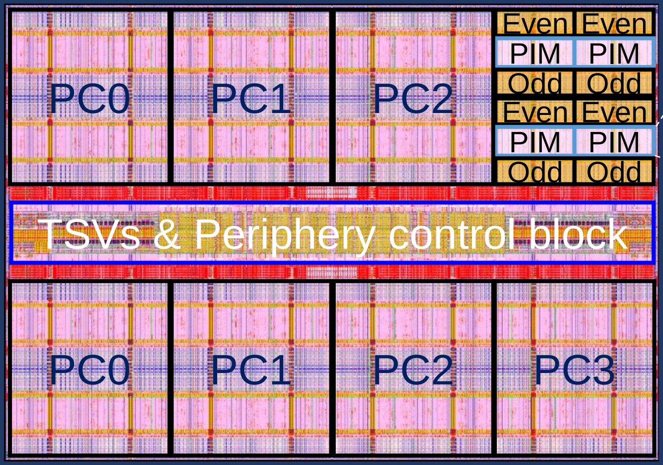


Chip Implementation and Integration with systems

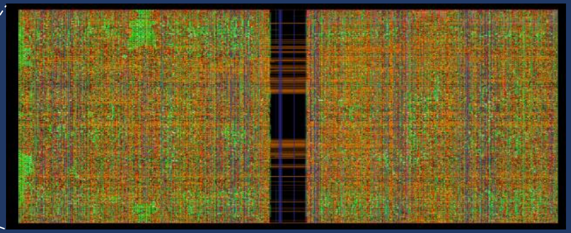
Implemented PIM by modifying a commercial HBM2 design (Aquabolt). Resulting HBM-PIM device codenamed Aquabolt-XL

Integrated the fabricated Aquabolt-XL with an unmodified GPU and Xilinx FPGA

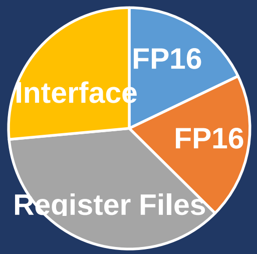
- Validated fabricated HBM-PIM in system with unmodified HBM controller
- Off-chip and on-chip PIM compute bandwidth is 1.23 TB/s and 4.92 TB/s, respectively.



PIM DRAM die



PIM block



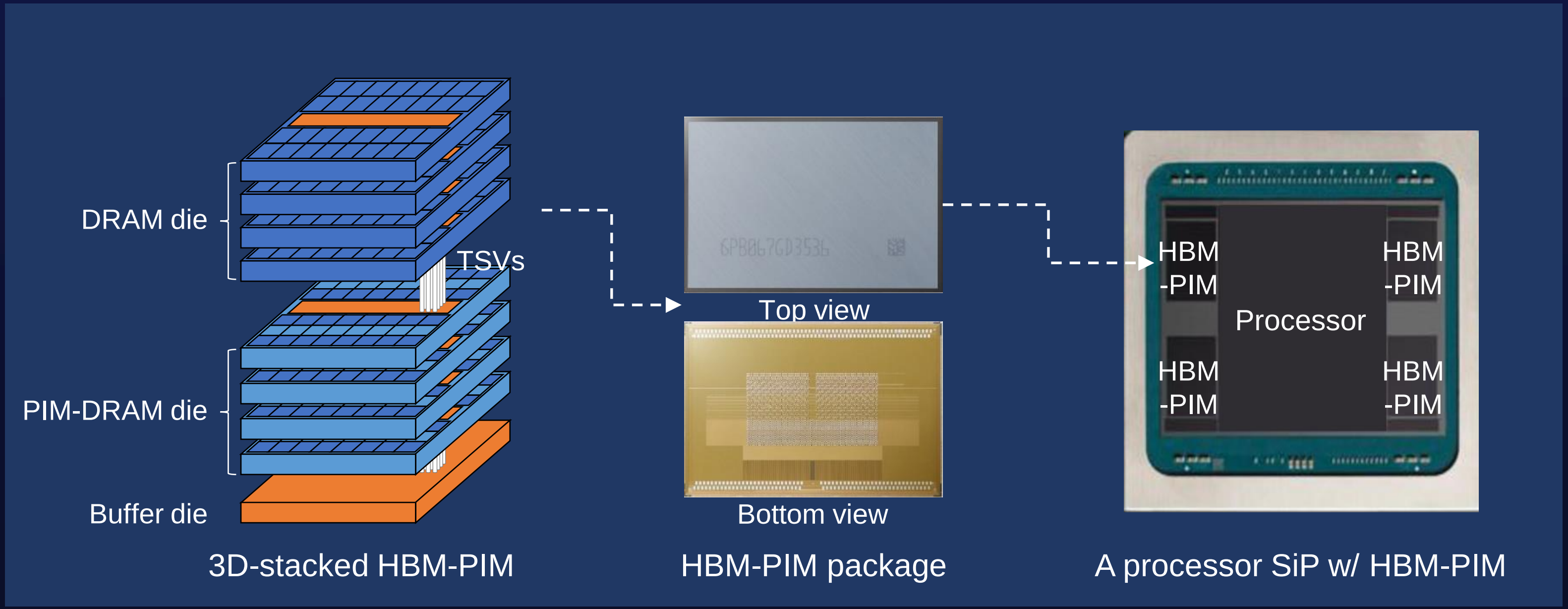
PIM block area breakdown

Chip Implementation and Integration with systems

Implemented PIM by modifying a commercial HBM2 design (Aquabolt). Resulting HBM-PIM device codenamed Aquabolt-XL

Integrated the fabricated Aquabolt-XL with an unmodified GPU and Xilinx FPGA

- Validated fabricated HBM-PIM in system with unmodified HBM controller
- Off-chip and on-chip PIM compute bandwidth is 1.23 TB/s and 4.92 TB/s, respectively.


DRAM die
TSVs
PIM-DRAM die
Buffer die
3D-stacked HBM-PIM
Top view
Bottom view
HBM-PIM package
HBM -PIM
HBM -PIM
Processor
HBM -PIM
HBM -PIM
A processor SiP w/ HBM-PIM