FLIP2M: Flexible Intra-layer Parallelism and Inter-layer Pipelining for Multi-model AR/VR Workloads

Gabriele Tombesi
Je Yang
Joseph Zuckerman
Davide Giri
William Baisi
Luca Carloni

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK
CS @CU

Background – AR/VR Workloads

- AR/VR applications require multiple concurrent DNNs running on the same SoC:

Background – AR/VR Workloads

- AR/VR applications require multiple concurrent DNNs running on the same SoC:
  - **Object Detection, Hand Tracking, Segmentation, etc**


Eye Tracking


Object Detection

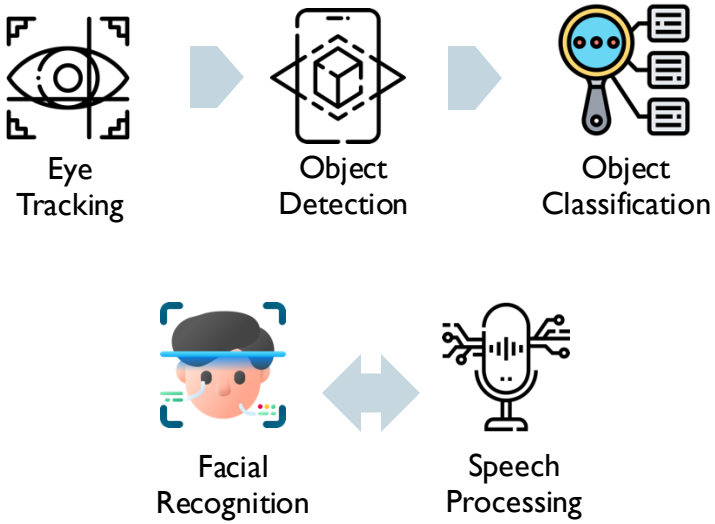

Object Classification


Facial Recognition


Speech Processing

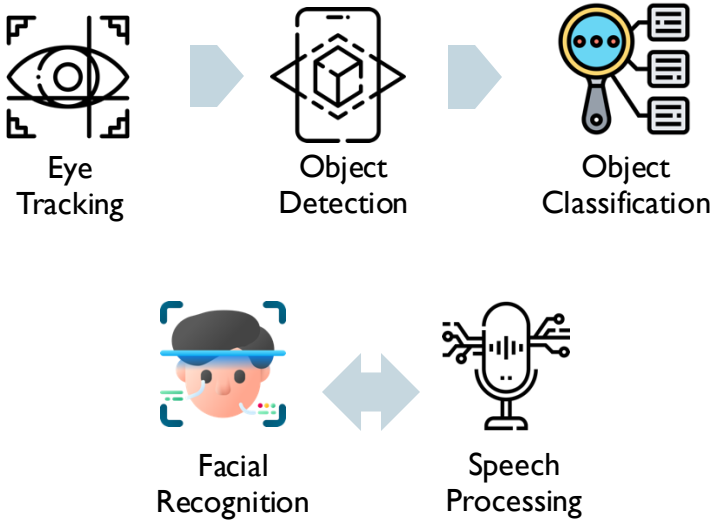
Background – AR/VR Workloads

- AR/VR applications require multiple concurrent DNNs running on the same SoC:
  - **Object Detection, Hand Tracking, Segmentation, etc**
  - **Complex intra- and inter-task dependencies (both static/dynamic)**



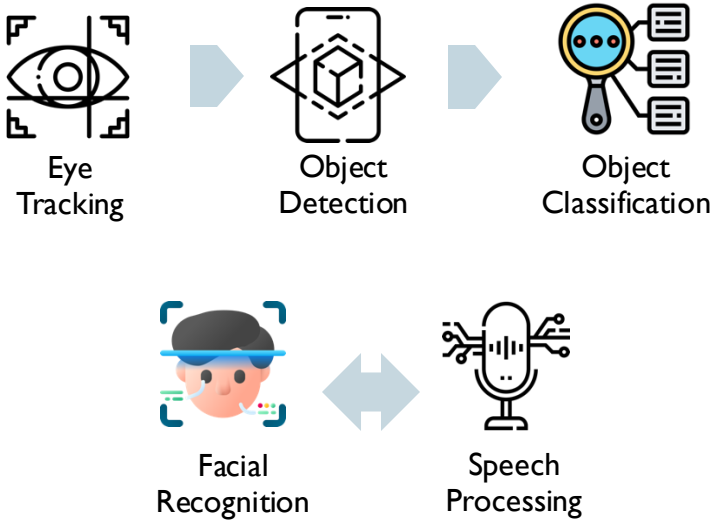
Background – AR/VR Workloads

- AR/VR applications require multiple concurrent DNNs running on the same SoC:
  - **Object Detection, Hand Tracking, Segmentation, etc**
  - **Complex intra- and inter-task dependencies (both static/dynamic)**
  - **Often coming with tight real-time constraints**



Background – AR/VR Workloads

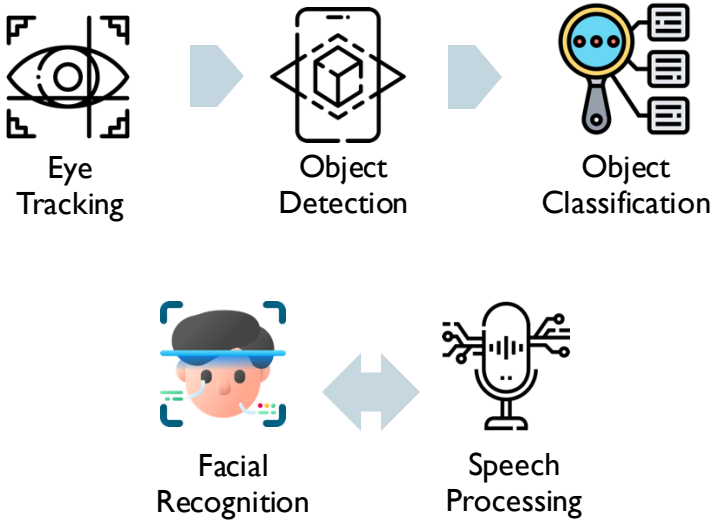
- AR/VR applications require multiple concurrent DNNs running on the same SoC:
  - **Object Detection, Hand Tracking, Segmentation, etc**
  - **Complex intra- and inter-task dependencies (both static/dynamic)**
  - **Often coming with tight real-time constraints**



- Main Execution Challenge →
  High **heterogeneity** in:

Background – AR/VR Workloads

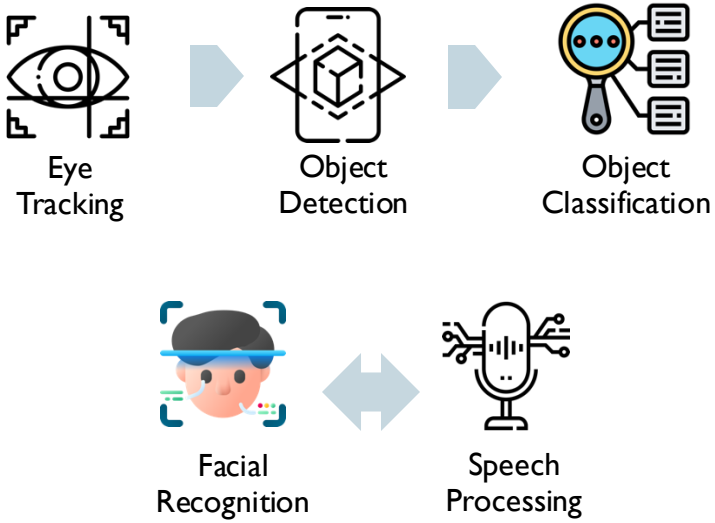
- AR/VR applications require multiple concurrent DNNs running on the same SoC:
  - **Object Detection, Hand Tracking, Segmentation, etc**
  - **Complex intra- and inter-task dependencies (both static/dynamic)**
  - **Often coming with tight real-time constraints**



- Main Execution Challenge →
  High **heterogeneity** in:
  - Layer shapes / operations

Background – AR/VR Workloads

- AR/VR applications require multiple concurrent DNNs running on the same SoC:
  - **Object Detection, Hand Tracking, Segmentation, etc**
  - **Complex intra- and inter-task dependencies (both static/dynamic)**
  - **Often coming with tight real-time constraints**

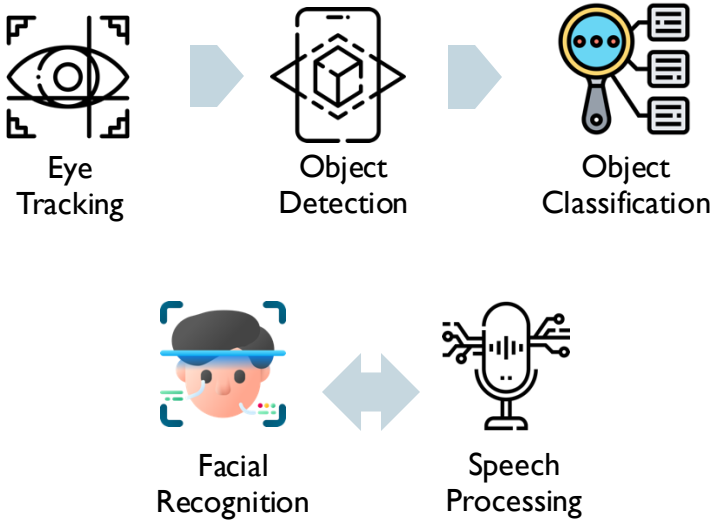


- Main Execution Challenge → High **heterogeneity** in:
  - Layer shapes / operations
  - Intra-model /

Model	Per-Layer Activation-Weight Size Ratio	Layer Operations
MobileNet-v2 [39]	Min: 0.0057, Median: 2.04, Max: 784	Conv2D, PW-Conv, DW-Conv, Residual

Background – AR/VR Workloads

- AR/VR applications require multiple concurrent DNNs running on the same SoC:
  - **Object Detection, Hand Tracking, Segmentation, etc**
  - **Complex intra- and inter-task dependencies (both static/dynamic)**
  - **Often coming with tight real-time constraints**



- Main Execution Challenge → High **heterogeneity** in:
  - Layer shapes / operations
  - Intra-model / Inter-model

Model	Per-Layer Activation-Weight Size Ratio	Layer Operations
MobileNet-v2 [39]	Min: 0.0057 Median: 2.04, Max: 784	Conv2D, PW-Conv, DW-Conv, Residual
ResNet-50 [19]	Min: 0.0106, Median: 0.68, Max: 49	Conv2D, FullyConnected, Residual
Vgg16 [40]	Min: 0.043, Median: 1.36, Max: 87.111	Conv2D, FullyConnected
SqueezeNet [20]	Min: 0.073, Median: 2.64, Max: 189.06	Conv2D, FullyConnected, Residual
Unet [38]	Min: 0.13, Median: 8.88, Max: 568	Conv2D, FullyConnected, UpConv, Concat
MobileBert [43]	Min: 0.125, Median: 0.5, Max: 8	Matrix-matrix multiplication, Residual

Background – Tiled Architectures

AR/VR Heterogeneity →

- **Over-specialized hardware:** latency/energy inefficiency across diverse DNN layers.
- **Model concurrency:** amplified range of operational requirements / resource demands.

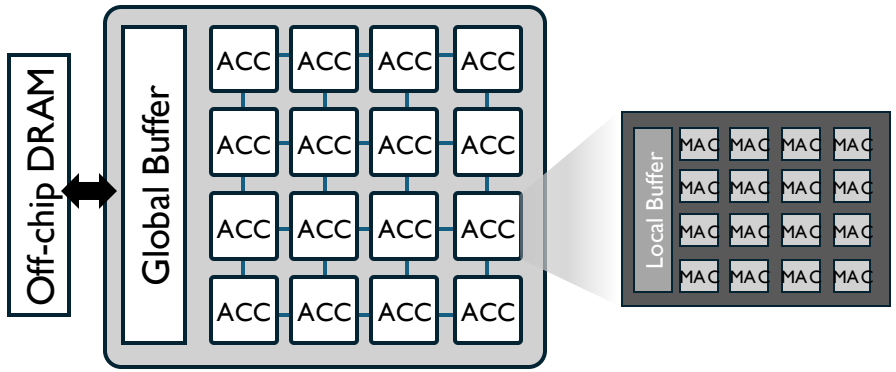
Background – Tiled Architectures

AR/VR Heterogeneity →

- **Over-specialized hardware:** latency/energy inefficiency across diverse DNN layers.
- **Model concurrency:** amplified range of operational requirements / resource demands.

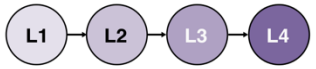
Recent advancements in SoC design from monolithic accelerators towards **tiled architectures:**

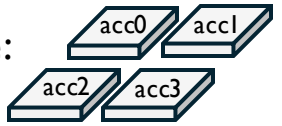
- Multiple accelerator tiles + on-chip global buffer
- Network-on-chip (NoC) based interconnect
- Multiple accelerator tiles cooperate to execute a task



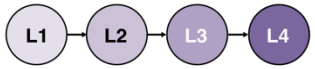
Background – Execution Strategies in Tiled Architectures

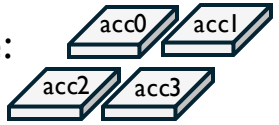
Background – Execution Strategies in Tiled Architectures

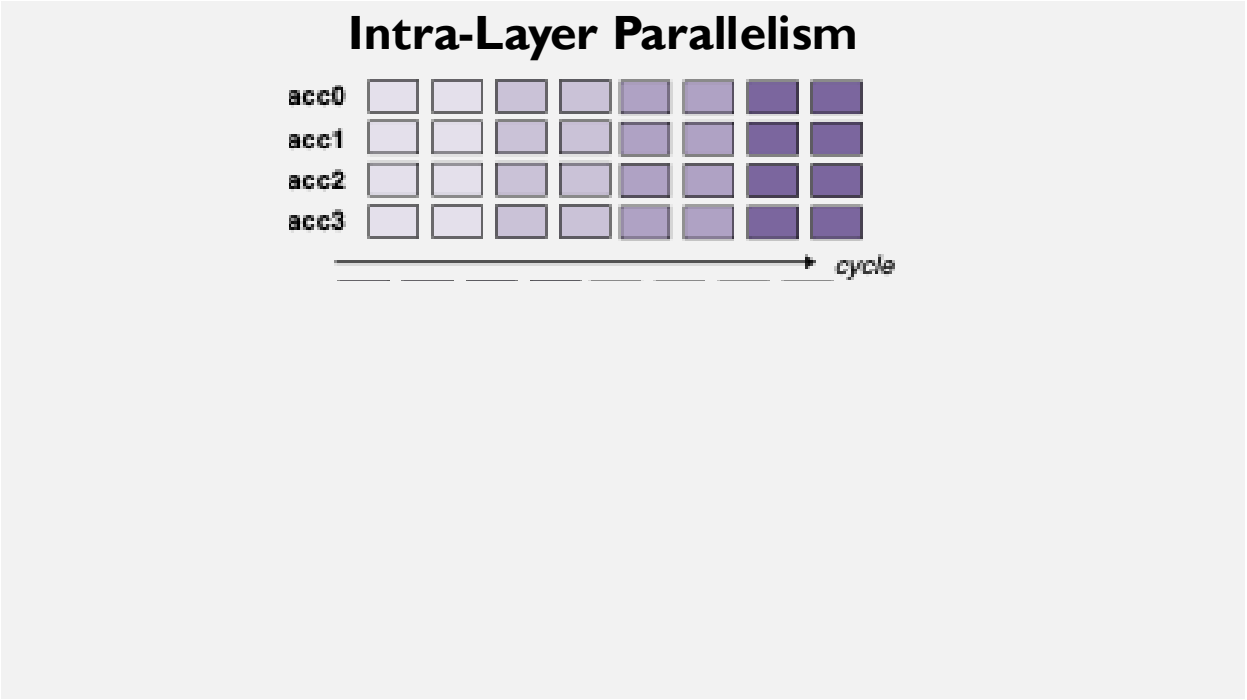
Network : 

Architecture: 

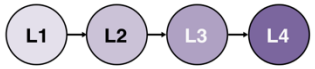
Background – Execution Strategies in Tiled Architectures

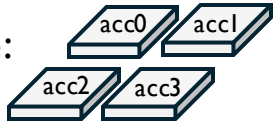
Network : 

Architecture: 

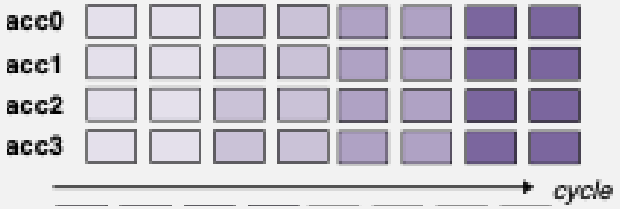


Background – Execution Strategies in Tiled Architectures

Network : 

Architecture: 

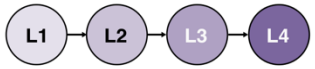
Intra-Layer Parallelism

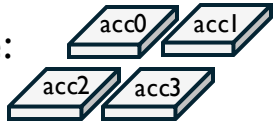


Input Channel (INP)

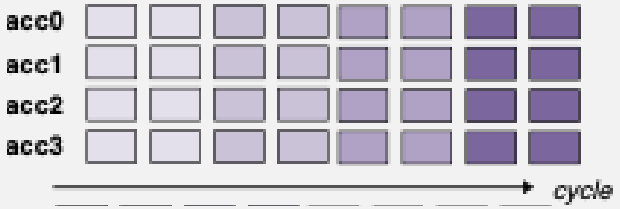
Each accelerator processes a subset of input channels → partial outputs combined.

Background – Execution Strategies in Tiled Architectures

Network : 

Architecture: 

Intra-Layer Parallelism



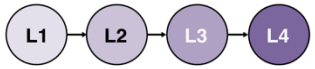
Input Channel (INP)

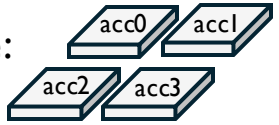
Each accelerator processes a subset of input channels → partial outputs combined.

Output Channel (OUTP)

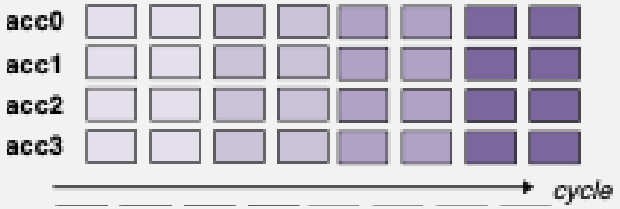
Filters split across accelerators → each outputs full feature maps.

Background – Execution Strategies in Tiled Architectures

Network : 

Architecture: 

Intra-Layer Parallelism



Input Channel (INP)

Each accelerator processes a subset of input channels → partial outputs combined.

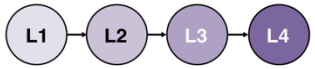
Output Channel (OUTP)

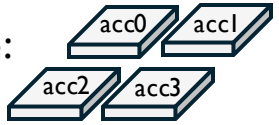
Filters split across accelerators → each outputs full feature maps.

Feature Map (FMP)

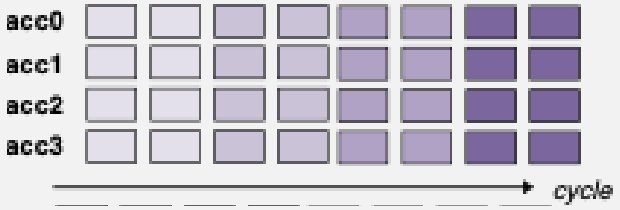
Input feature map divided spatially → each computes a slice of outputs.

Background – Execution Strategies in Tiled Architectures

Network : 

Architecture: 

Intra-Layer Parallelism



Input Channel (INP)

Each accelerator processes a subset of input channels → partial outputs combined.

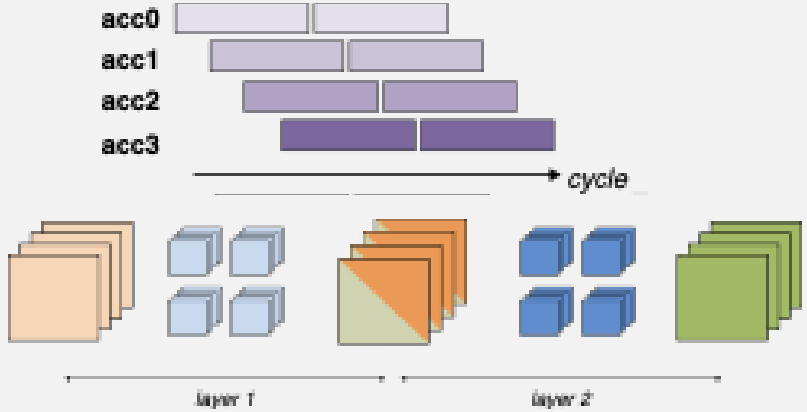
Output Channel (OUTP)

Filters split across accelerators → each outputs full feature maps.

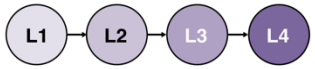
Feature Map (FMP)

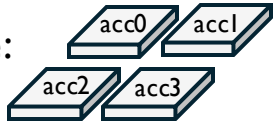
Input feature map divided spatially → each computes a slice of outputs.

Inter-Layer Pipelining

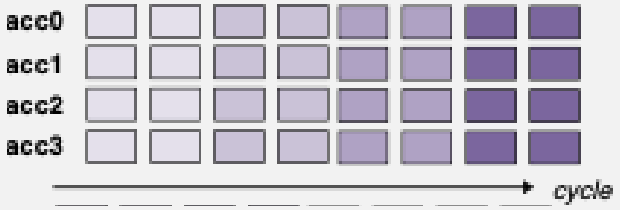


Background – Execution Strategies in Tiled Architectures

Network : 

Architecture: 

Intra-Layer Parallelism



Input Channel (INP)

Each accelerator processes a subset of input channels → partial outputs combined.

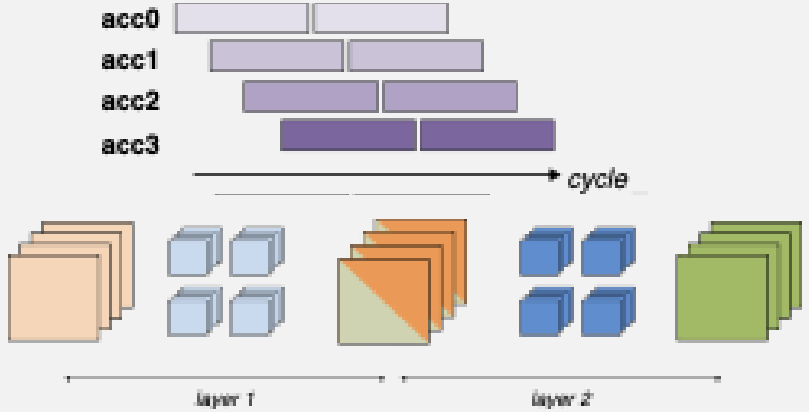
Output Channel (OUTP)

Filters split across accelerators → each outputs full feature maps.

Feature Map (FMP)

Input feature map divided spatially → each computes a slice of outputs.

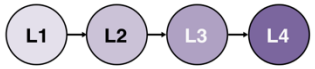
Inter-Layer Pipelining

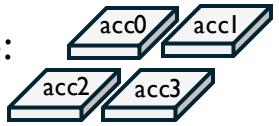


Adjacent layer fused into ***segments***, forwarding intermediate activations on-chip.

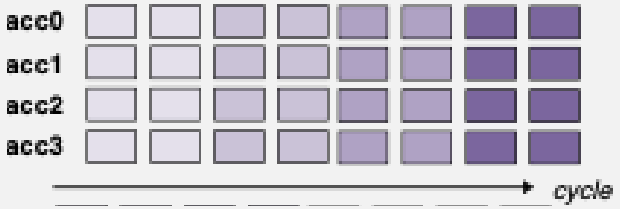
- **Benefits**: off-chip accesses reduction
- **Cost**: pipeline fill/flush overhead and larger on-chip buffers.

Background – Execution Strategies in Tiled Architectures

Network : 

Architecture: 

Intra-Layer Parallelism



Per-Layer Resource Allocation

Input Channel (INP)

Each accelerator processes a subset of input channels → partial outputs combined.

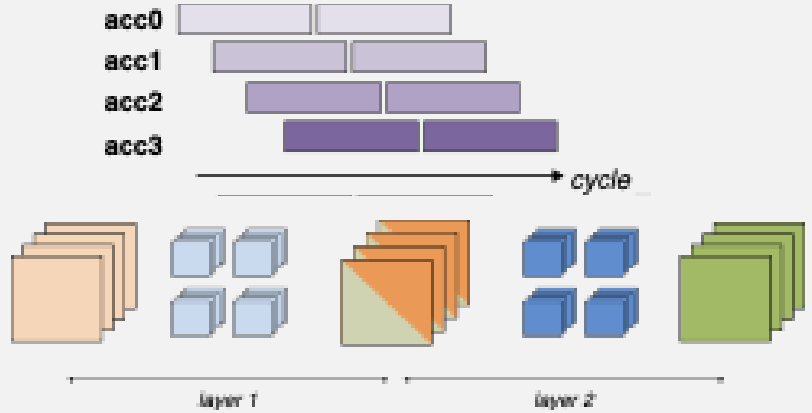
Output Channel (OUTP)

Filters split across accelerators → each outputs full feature maps.

Feature Map (FMP)

Input feature map divided spatially → each computes a slice of outputs.

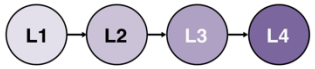
Inter-Layer Pipelining

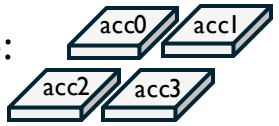


Adjacent layer fused into ***segments***, forwarding intermediate activations on-chip.

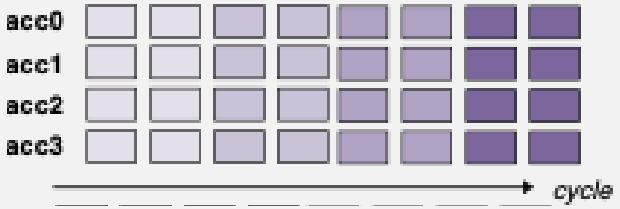
- **Benefits**: off-chip accesses reduction
- **Cost**: pipeline fill/flush overhead and larger on-chip buffers.

Background – Execution Strategies in Tiled Architectures

Network : 

Architecture: 

Intra-Layer Parallelism



Per-Layer Resource Allocation

Input Channel (INP)

Each accelerator processes a subset of input channels → partial outputs combined.

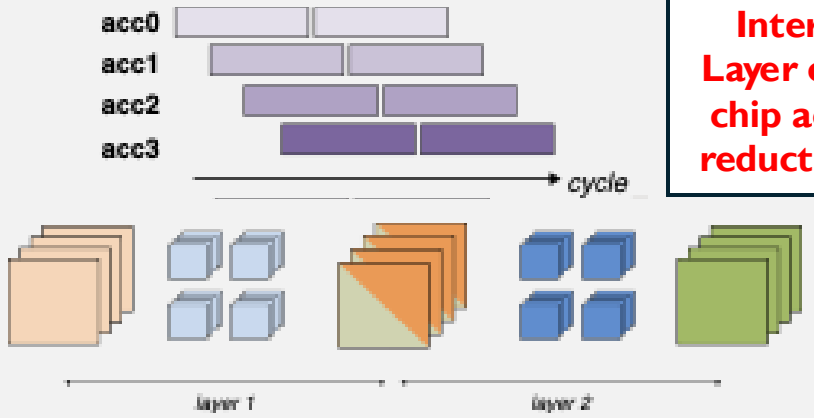
Output Channel (OUTP)

Filters split across accelerators → each outputs full feature maps.

Feature Map (FMP)

Input feature map divided spatially → each computes a slice of outputs.

Inter-Layer Pipelining

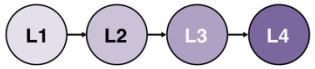


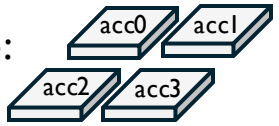
Inter-Layer off-chip acc. reduction

Adjacent layer fused into ***segments***, forwarding intermediate activations on-chip.

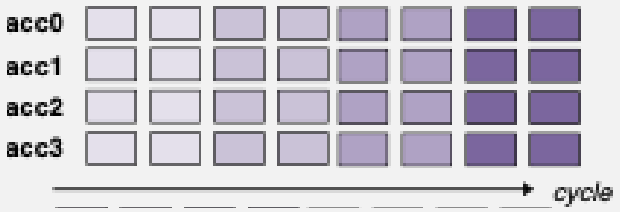
- **Benefits**: off-chip accesses reduction
- **Cost**: pipeline fill/flush overhead and larger on-chip buffers.

Background – Execution Strategies in Tiled Architectures

Network : 

Architecture: 

Intra-Layer Parallelism



Per-Layer Resource Allocation

Input Channel (INP)

Each accelerator processes a subset of input channels → partial outputs combined.

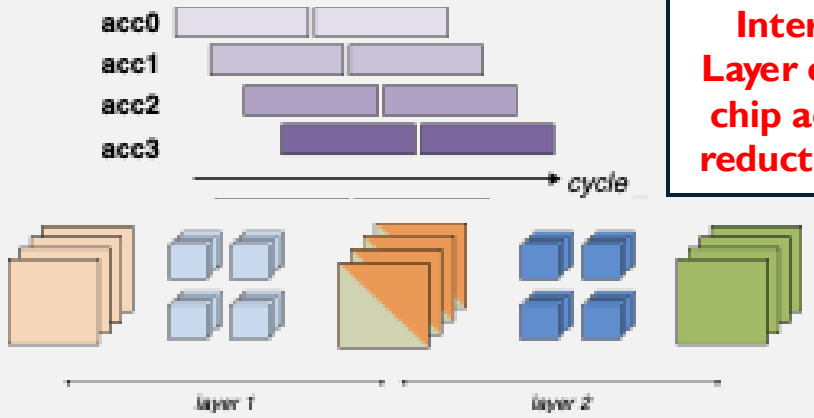
Output Channel (OUTP)

Filters split across accelerators → each outputs full feature maps.

Feature Map (FMP)

Input feature map divided spatially → each computes a slice of outputs.

Inter-Layer Pipelining



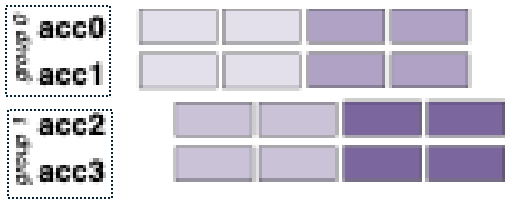
Inter-Layer off-chip acc. reduction

Adjacent layer fused into ***segments***, forwarding intermediate activations on-chip.

- **Benefits**: off-chip accesses reduction
- **Cost**: pipeline fill/flush overhead and larger on-chip buffers.

→ **Combination** can strike the right balance but:

- Further increases the mapping space
- Requires advanced architectural support



Challenges - Summary

- Intra/Inter - model **heterogeneity** (layers vary in intensity and size)
- Multi-model **concurrency** (contention for compute and off-chip bandwidth)
- **Large mapping space** and **architectural challenges** in tiled architectures (Intra/Inter-layer mapping)

Challenges - Summary

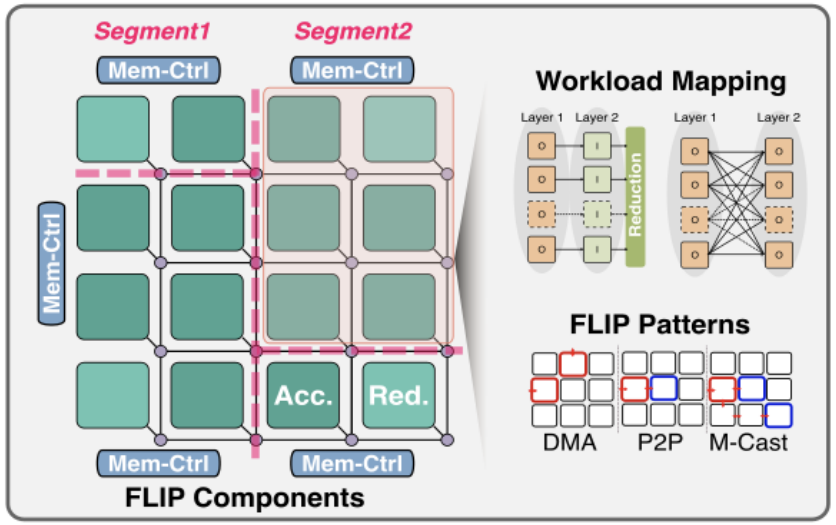
- Intra/Inter - model **heterogeneity** (layers vary in intensity and size)
- Multi-model **concurrency** (contention for compute and off-chip bandwidth)
- **Large mapping space** and **architectural challenges** in tiled architectures (Intra/Inter-layer mapping)

→ We propose **FLIP2M**: a holistic solution combining intra- and inter-layer optimization for multi-model AR/VR workloads on tiled architecture

Challenges - Summary

- Intra/Inter - model **heterogeneity** (layers vary in intensity and size)
- Multi-model **concurrency** (contention for compute and off-chip bandwidth)
- **Large mapping space** and **architectural challenges** in tiled architectures (Intra/Inter-layer mapping)

→ We propose **FLIP2M**: a holistic solution combining intra- and inter-layer optimization for multi-model AR/VR workloads on tiled architecture

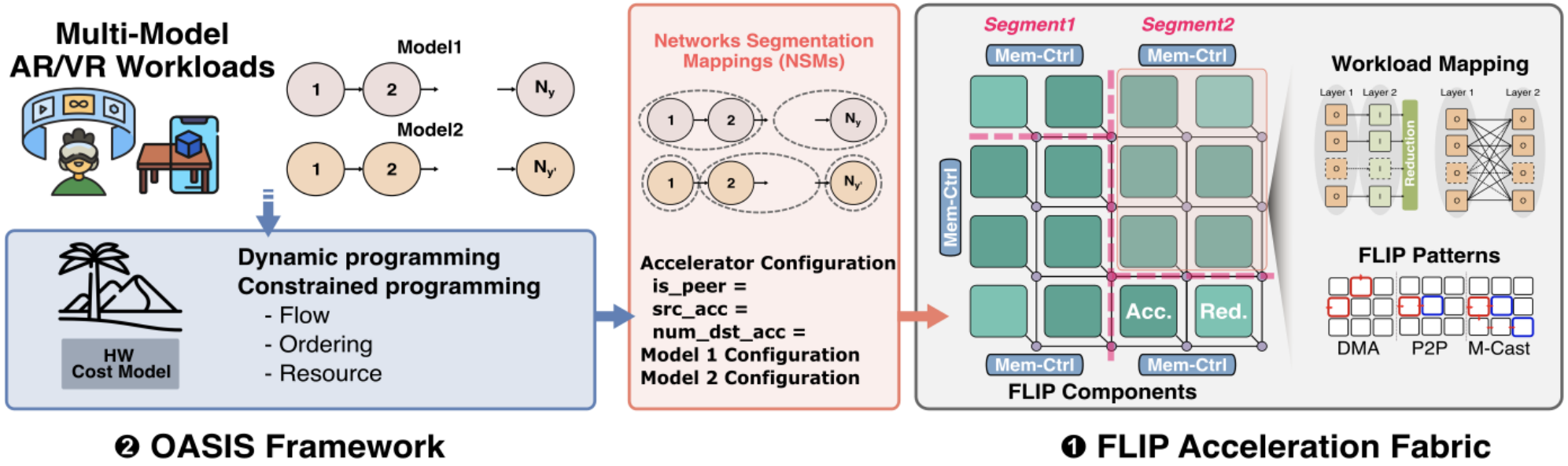


❶ FLIP Acceleration Fabric

Challenges - Summary

- Intra/Inter - model **heterogeneity** (layers vary in intensity and size)
- Multi-model **concurrency** (contention for compute and off-chip bandwidth)
- **Large mapping space** and **architectural challenges** in tiled architectures (Intra/Inter-layer mapping)

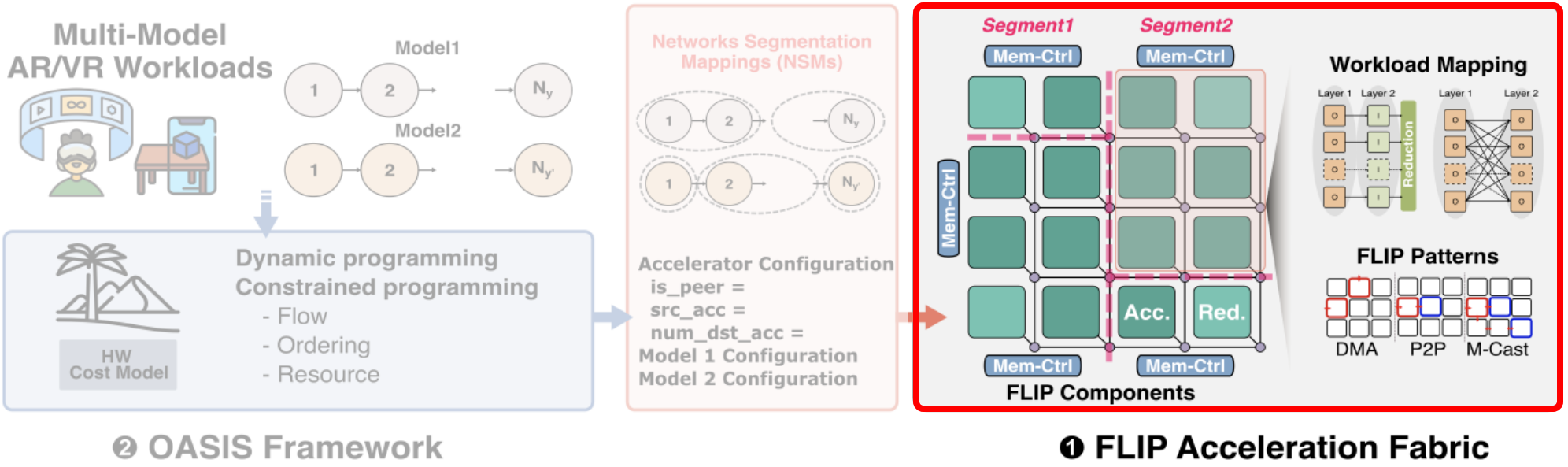
→ We propose **FLIP2M**: a holistic solution combining intra- and inter-layer optimization for multi-model AR/VR workloads on tiled architecture



Challenges - Summary

- Intra/Inter - model **heterogeneity** (layers vary in intensity and size)
- Multi-model **concurrency** (contention for compute and off-chip bandwidth)
- **Large mapping space** and **architectural challenges** in tiled architectures (Intra/Inter-layer mapping)

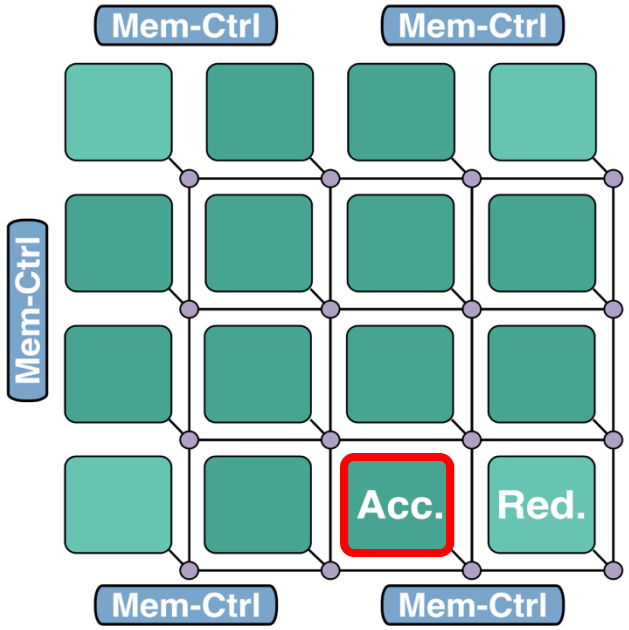
→ We propose **FLIP2M**: a holistic solution combining intra- and inter-layer optimization for multi-model AR/VR workloads on tiled architectures



❷ OASIS Framework

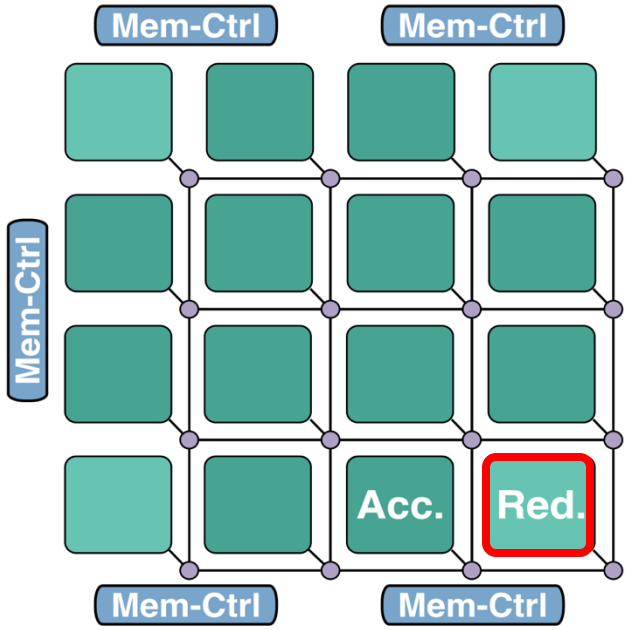
❶ FLIP Acceleration Fabric

❶ FLIP Architecture - Overview



- *Accelerator Tiles:*
  - Coarse-grained engines → execute full layers or large portions.
  - Include private buffers, issue long load/store bursts.
  - Flexible design (vector lanes, systolic, etc.), to support DNN kernels

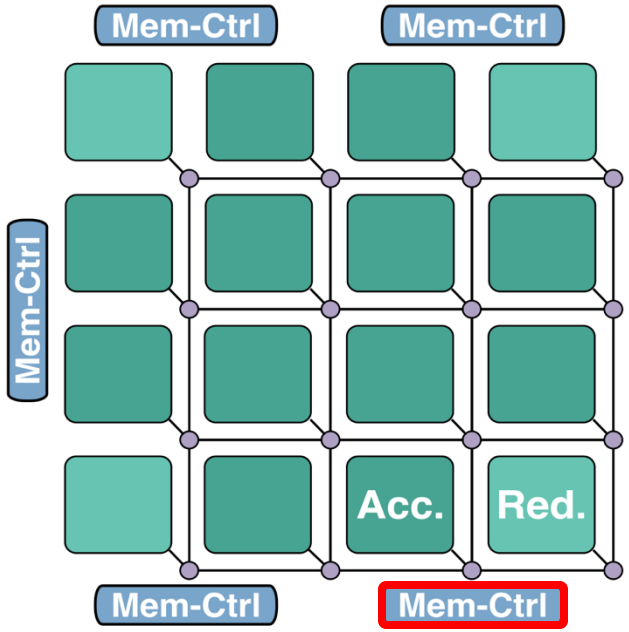
❶ FLIP Architecture - Overview



- *Accelerator Tiles:*
  - Coarse-grained engines → execute full layers or large portions.
  - Include private buffers, issue long load/store bursts.
  - Flexible design (vector lanes, systolic, etc.), to support DNN kernels

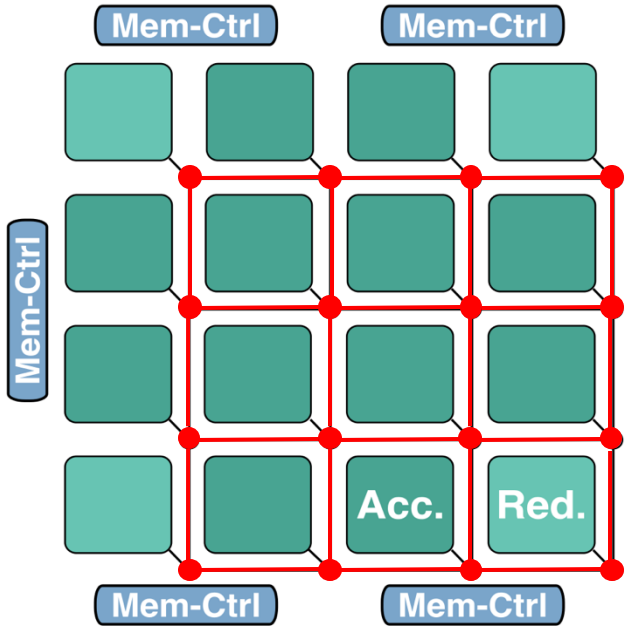
- *Reduction Tiles:*
  - Specialized for tensor addition (e.g., INP partial sums, residual merges).
  - Hardware support avoids software bottlenecks.

❶ FLIP Architecture - Overview



- *Accelerator Tiles:*
  - Coarse-grained engines → execute full layers or large portions.
  - Include private buffers, issue long load/store bursts.
  - Flexible design (vector lanes, systolic, etc.), to support DNN kernels
- *Reduction Tiles:*
  - Specialized for tensor addition (e.g., INP partial sums, residual merges).
  - Hardware support avoids software bottlenecks.
- *Memory Controllers:*
  - Multiple DRAM controllers scale bandwidth and reduce contention. Partitioned address space → better isolation between segments.

❶ FLIP Architecture - Overview

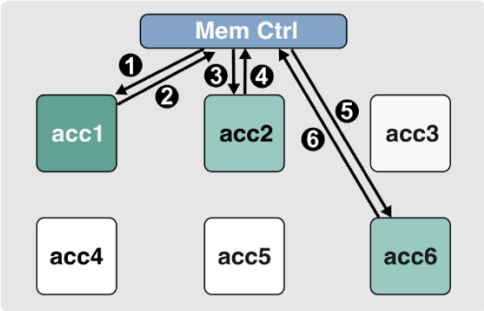


- *Accelerator Tiles:*
  - Coarse-grained engines → execute full layers or large portions.
  - Include private buffers, issue long load/store bursts.
  - Flexible design (vector lanes, systolic, etc.), to support DNN kernels
- *Reduction Tiles:*
  - Specialized for tensor addition (e.g., INP partial sums, residual merges).
  - Hardware support avoids software bottlenecks.
- *Memory Controllers:*
  - Multiple DRAM controllers scale bandwidth and reduce contention. Partitioned address space → better isolation between segments.
- *NoC-based Interconnect* supporting a variety of communication patterns

❶ FLIP Architecture – Communication Patterns

❶ FLIP Architecture – Communication Patterns

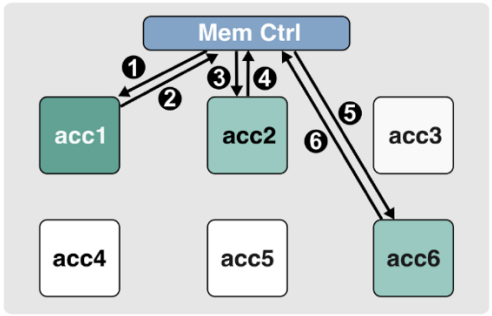
Direct Memory Access (DMA)



- Moves data between **DRAM** and accelerator **buffers**.
- Sustains **high bandwidth** for large transfers (weights, features).

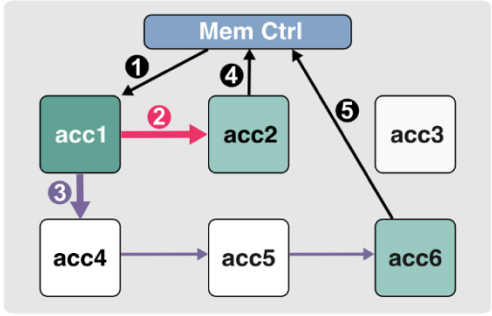
❶ FLIP Architecture – Communication Patterns

Direct Memory Access (DMA)



- Moves data between **DRAM** and accelerator **buffers**.
- Sustains **high bandwidth** for large transfers (weights, features).

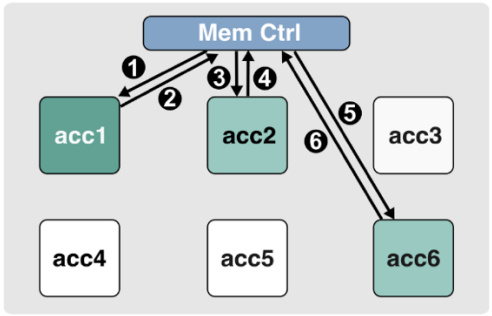
Peer to Peer (P2P)



- Direct transfer between compute tiles. Key for **inter-layer pipelining** → avoids round-trip to DRAM.
- Requires **synchronization** to prevent deadlock.

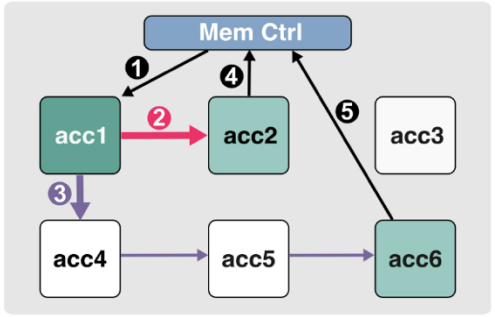
❶ FLIP Architecture – Communication Patterns

Direct Memory Access (DMA)



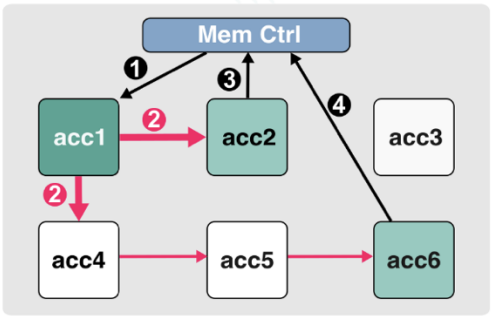
- Moves data between **DRAM** and accelerator **buffers**.
- Sustains **high bandwidth** for large transfers (weights, features).

Peer to Peer (P2P)



- Direct transfer between compute tiles. Key for **inter-layer pipelining** → avoids round-trip to DRAM.
- Requires **synchronization** to prevent deadlock.

Multicast



- One tile sends output to **multiple downstream tiles** in parallel.
- Crucial for intra + inter-layer optimizations

❶ FLIP Architecture – Workload Mapping

- Single-layer segments: OUTP / INP

❶ FLIP Architecture – Workload Mapping

- Single-layer segments: OUTP / INP
- Multi-layer segments : Layers forward outputs directly to next layer's accelerators (pipelined)

❶ FLIP Architecture – Workload Mapping

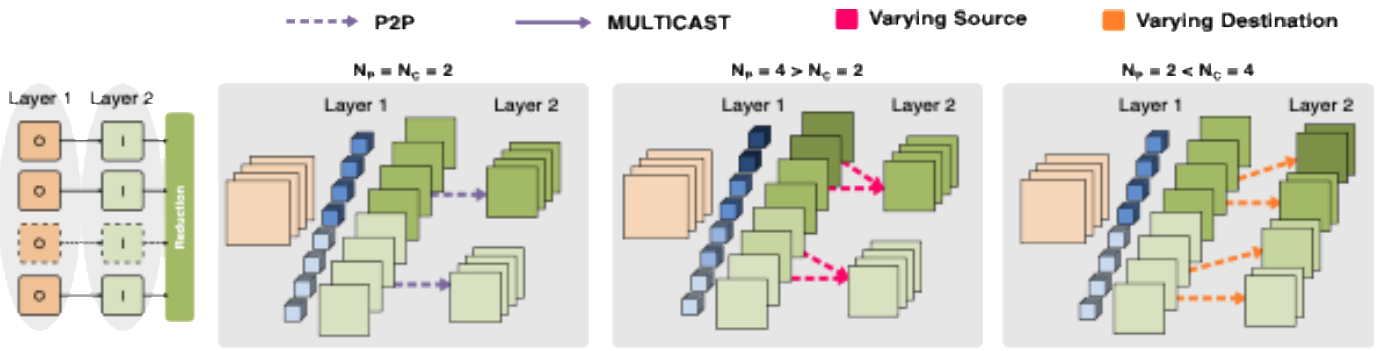
- Single-layer segments: OUTP / INP
- Multi-layer segments : Layers forward outputs directly to next layer's accelerators (pipelined)
  - INP→OUTP bottlenecked by reduction tile → excluded.
  - Too many pipelined layers increase fill/flush overhead → capped at 3.

❶ FLIP Architecture – Workload Mapping

- Single-layer segments: OUTP / INP
- Multi-layer segments : Layers forward outputs directly to next layer's accelerators (pipelined)
  - INP→OUTP bottlenecked by reduction tile → excluded.
  - Too many pipelined layers increase fill/flush overhead → capped at 3.

OUTP → INP

- Producers split output channels (OUTP).
- Consumers use INP → need only subsets of channels.
- Communication depends on producer/consumer ratio



❶ FLIP Architecture – Workload Mapping

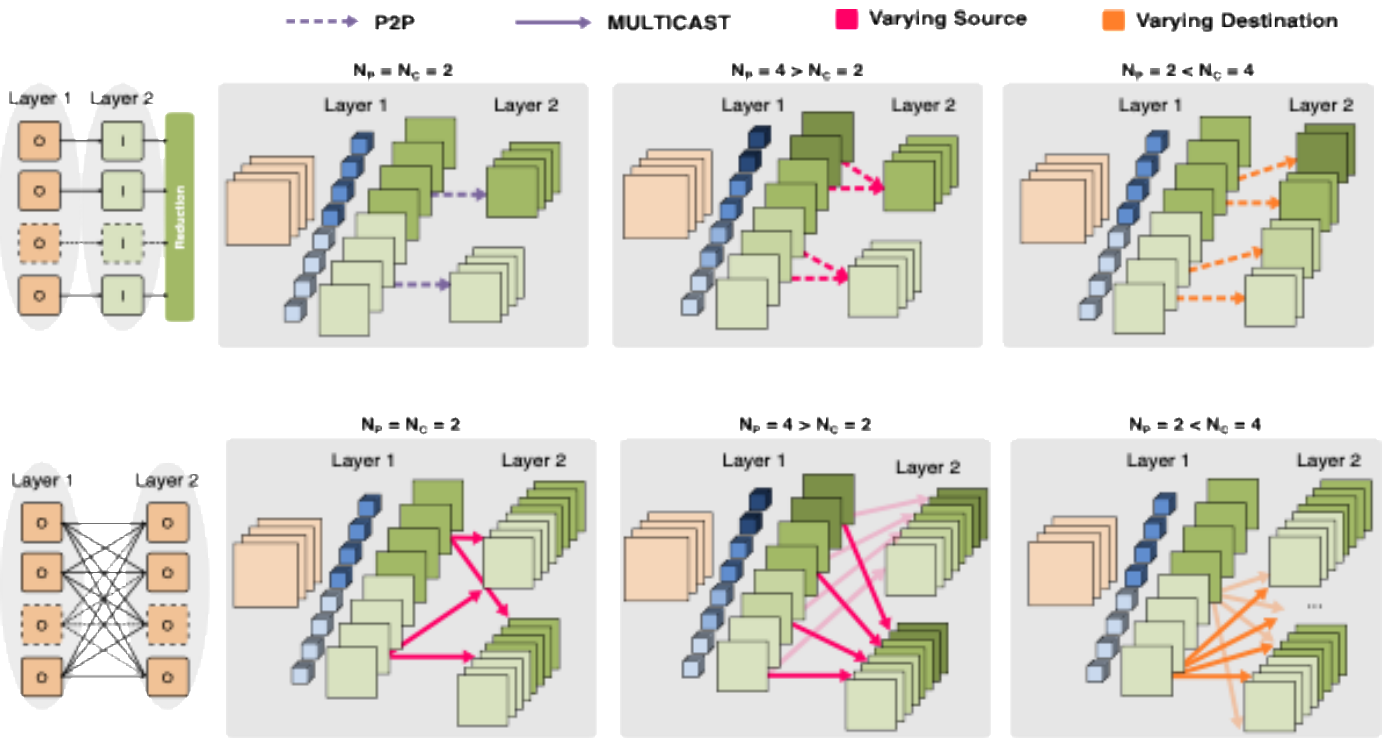
- Single-layer segments: OUTP / INP
- Multi-layer segments : Layers forward outputs directly to next layer's accelerators (pipelined)
  - INP→OUTP bottlenecked by reduction tile → excluded.
  - Too many pipelined layers increase fill/flush overhead → capped at 3.

OUTP → INP

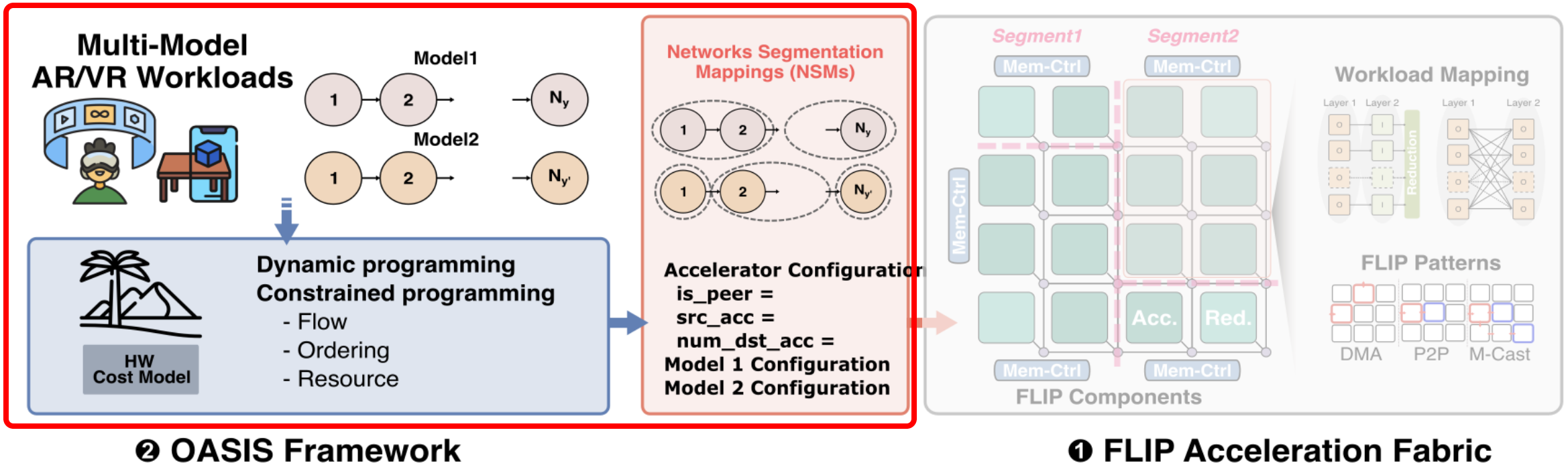
- Producers split output channels (OUTP).
- Consumers use INP → need only subsets of channels.
- Communication depends on producer/consumer ratio

OUTP → OUTP

- Both layers parallelized by filters (OUTP).
- Each consumer needs *all* input channels.
- Requires **multicast:** producers send to all consumers, each consumer gathers from all producers.



OASIS Optimization Framework - Overview



❷ OASIS Framework

❶ FLIP Acceleration Fabric

❷ OASIS Optimization Framework - Overview

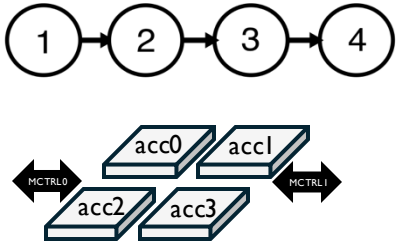
- **Goal:** Optimize segmentation, resource allocation, and scheduling for multi-model AR/VR on FLIP.

❷ OASIS Optimization Framework - Overview

- **Goal:** Optimize segmentation, resource allocation, and scheduling for multi-model AR/VR on FLIP.
- **Inputs:**
  - DNN topologies

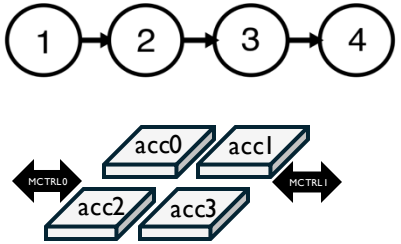
❷ OASIS Optimization Framework - Overview

- **Goal:** Optimize segmentation, resource allocation, and scheduling for multi-model AR/VR on FLIP.
- **Inputs:**
  - DNN topologies
  - FLIP prototype



❷ OASIS Optimization Framework - Overview

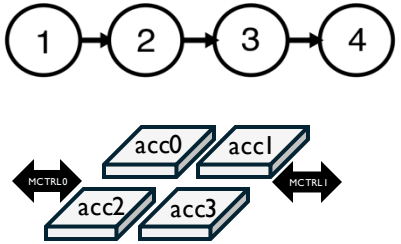
- **Goal:** Optimize segmentation, resource allocation, and scheduling for multi-model AR/VR on FLIP.
- **Inputs:**
  - DNN topologies
  - FLIP prototype
- **Segment:**
  Execution mode from layer i:



❷ OASIS Optimization Framework - Overview

- **Goal:** Optimize segmentation, resource allocation, and scheduling for multi-model AR/VR on FLIP.
- **Inputs:**
  - DNN topologies
  - FLIP prototype
- **Segment:**

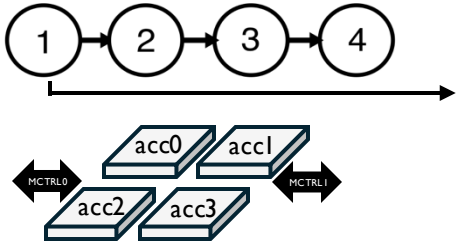
  Execution mode from layer i:
  - Segment depth (*sd*)
  - Segment parallelism (*sp*)
  - Segment resource (*SR*)



❷ OASIS Optimization Framework - Overview

- **Goal:** Optimize segmentation, resource allocation, and scheduling for multi-model AR/VR on FLIP.
- **Inputs:**
  - DNN topologies
  - FLIP prototype
- **Segment:**

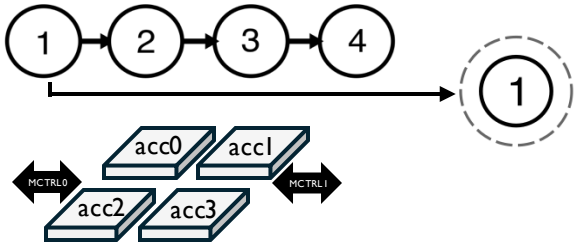
  Execution mode from layer i:
  - Segment depth (*sd*)
  - Segment parallelism (*sp*)
  - Segment resource (*SR*)



❷ OASIS Optimization Framework - Overview

- **Goal:** Optimize segmentation, resource allocation, and scheduling for multi-model AR/VR on FLIP.
- **Inputs:**
  - DNN topologies
  - FLIP prototype
- **Segment:**

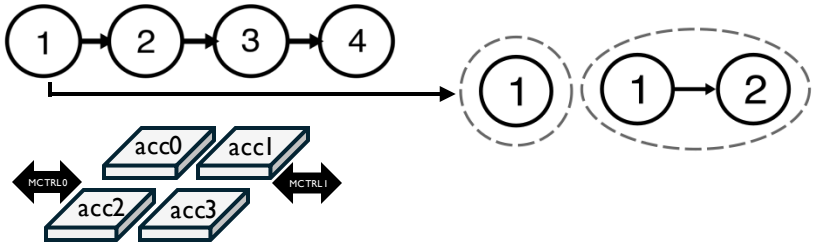
  Execution mode from layer i:
  - Segment depth (*sd*)
  - Segment parallelism (*sp*)
  - Segment resource (*SR*)



❷ OASIS Optimization Framework - Overview

- **Goal:** Optimize segmentation, resource allocation, and scheduling for multi-model AR/VR on FLIP.
- **Inputs:**
  - DNN topologies
  - FLIP prototype
- **Segment:**

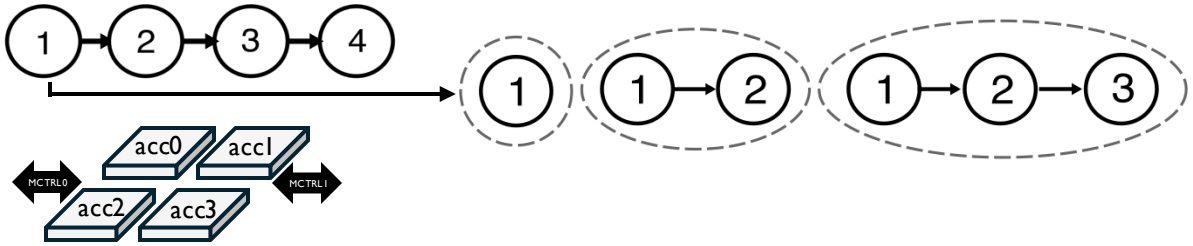
  Execution mode from layer i:
  - Segment depth (*sd*)
  - Segment parallelism (*sp*)
  - Segment resource (*SR*)



❷ OASIS Optimization Framework - Overview

- **Goal:** Optimize segmentation, resource allocation, and scheduling for multi-model AR/VR on FLIP.
- **Inputs:**
  - DNN topologies
  - FLIP prototype
- **Segment:**

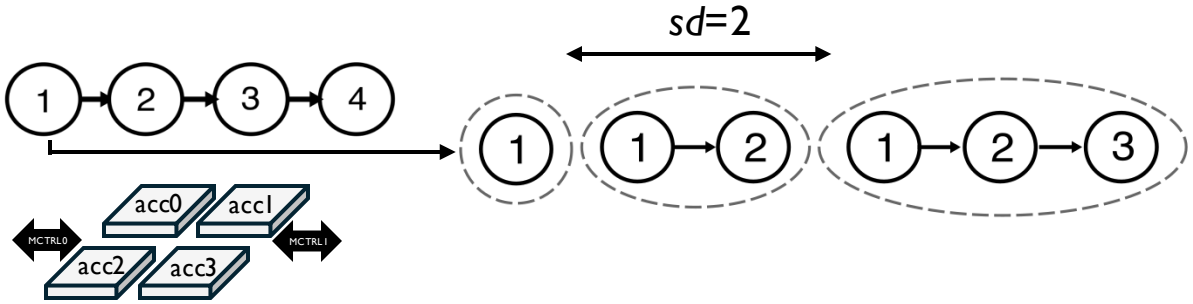
  Execution mode from layer i:
  - Segment depth (*sd*)
  - Segment parallelism (*sp*)
  - Segment resource (*SR*)



❷ OASIS Optimization Framework - Overview

- **Goal:** Optimize segmentation, resource allocation, and scheduling for multi-model AR/VR on FLIP.
- **Inputs:**
  - DNN topologies
  - FLIP prototype
- **Segment:**

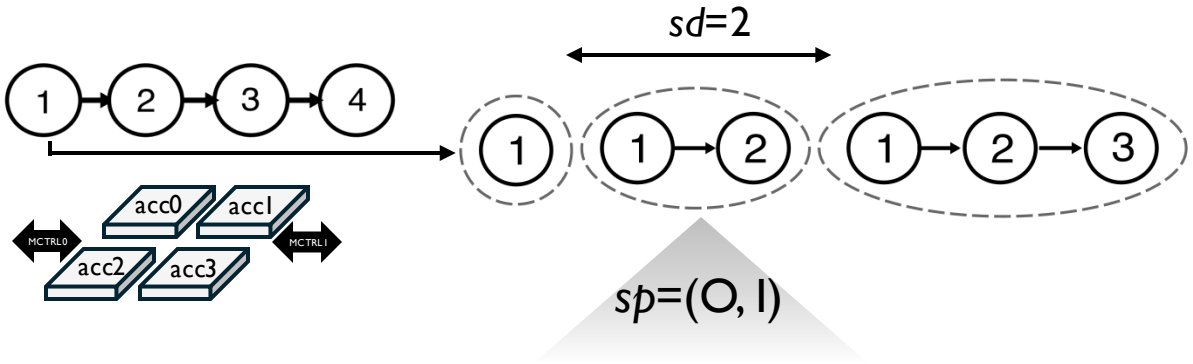
  Execution mode from layer i:
  - Segment depth (*sd*)
  - Segment parallelism (*sp*)
  - Segment resource (*SR*)



❷ OASIS Optimization Framework - Overview

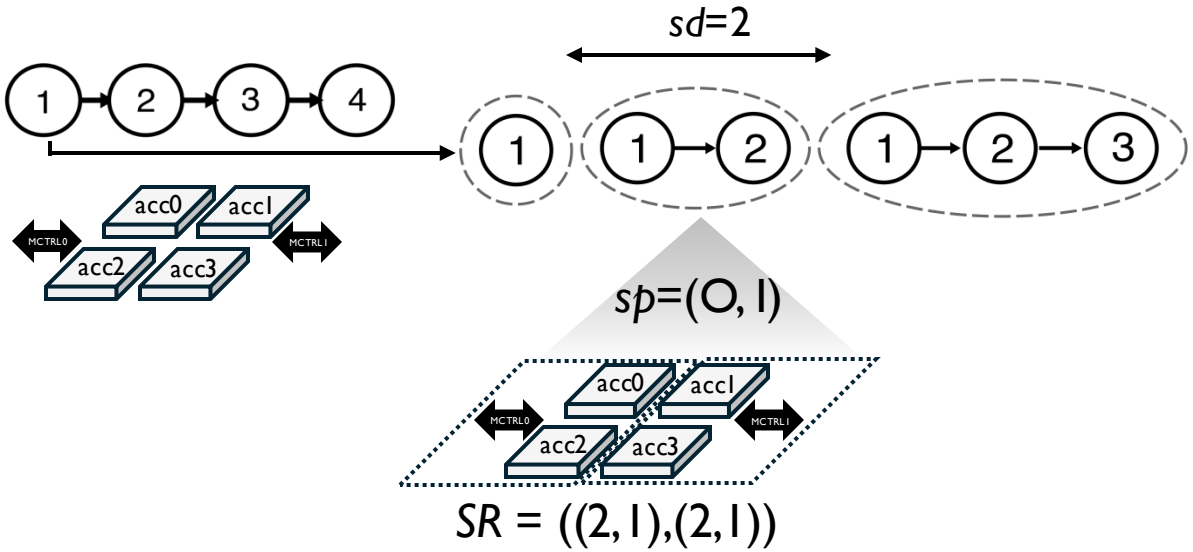
- **Goal:** Optimize segmentation, resource allocation, and scheduling for multi-model AR/VR on FLIP.
- **Inputs:**
  - DNN topologies
  - FLIP prototype
- **Segment:**

  Execution mode from layer i:
  - Segment depth (*sd*)
  - Segment parallelism (*sp*)
  - Segment resource (*SR*)



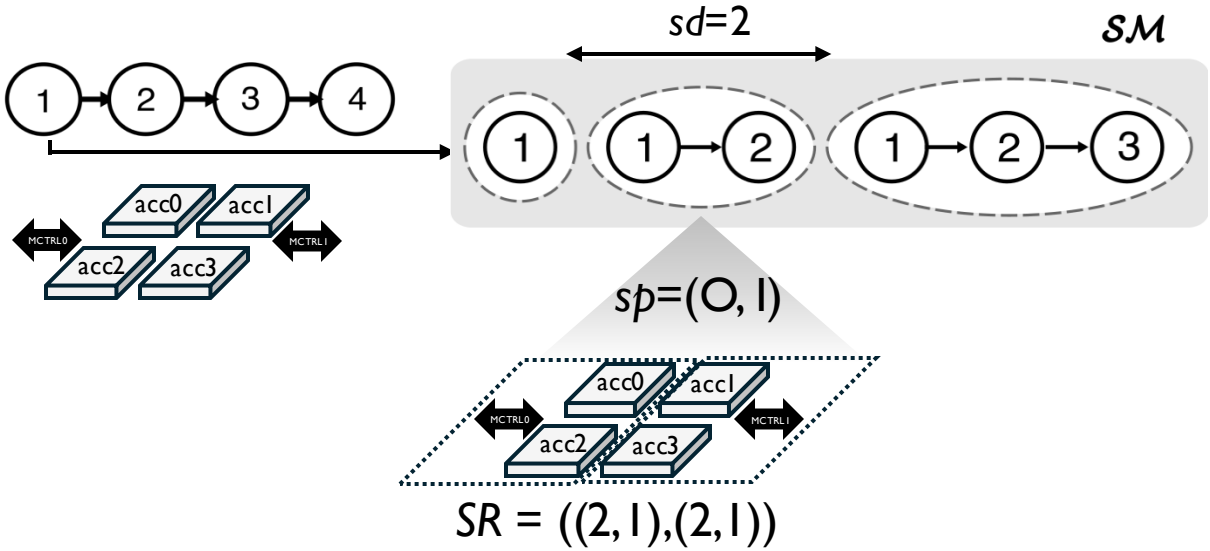
❷ OASIS Optimization Framework - Overview

- **Goal:** Optimize segmentation, resource allocation, and scheduling for multi-model AR/VR on FLIP.
- **Inputs:**
  - DNN topologies
  - FLIP prototype
- **Segment:**
  Execution mode from layer i:
  - Segment depth (*sd*)
  - Segment parallelism (*sp*)
  - Segment resource (*SR*)



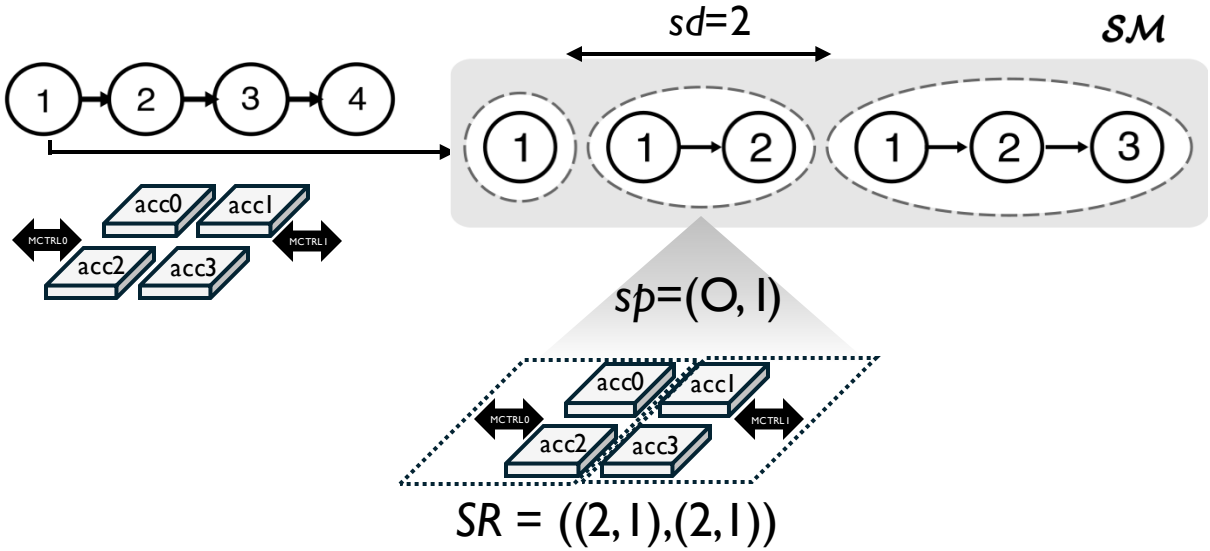
❷ OASIS Optimization Framework - Overview

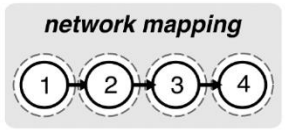
- **Goal:** Optimize segmentation, resource allocation, and scheduling for multi-model AR/VR on FLIP.
- **Inputs:**
  - DNN topologies
  - FLIP prototype
- **Segment:**
  Execution mode from layer i:
  - Segment depth (*sd*)
  - Segment parallelism (*sp*)
  - Segment resource (*SR*)
- **Mapping Space:**
  - Segment mapping space (*SM*) = all possible segments from a layer



❷ OASIS Optimization Framework - Overview

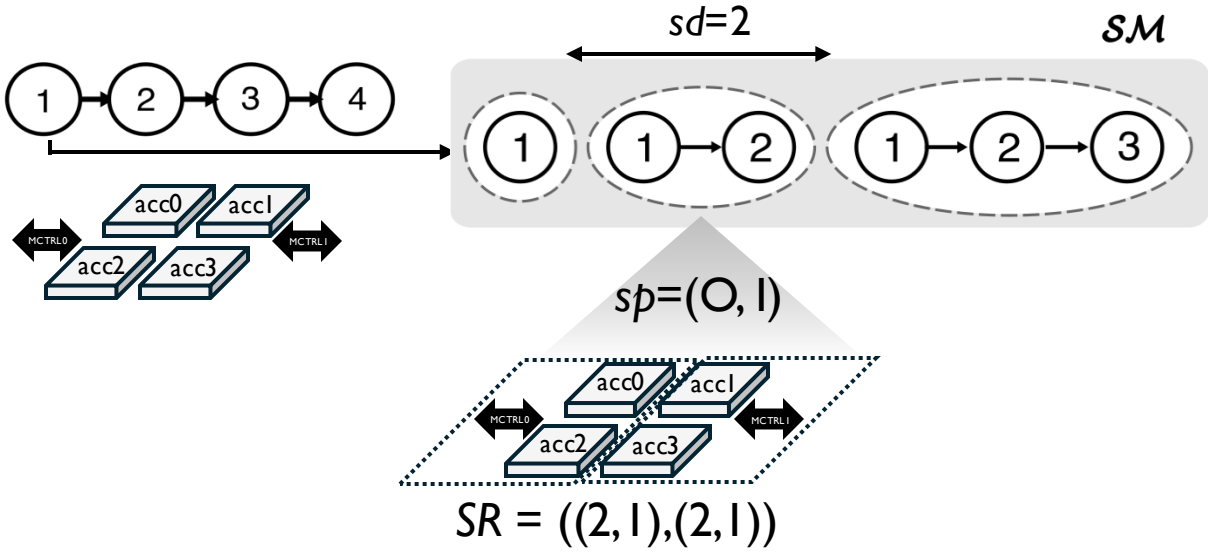
- **Goal:** Optimize segmentation, resource allocation, and scheduling for multi-model AR/VR on FLIP.
- **Inputs:**
  - DNN topologies
  - FLIP prototype
- **Segment:**
  Execution mode from layer i:
  - Segment depth (*sd*)
  - Segment parallelism (*sp*)
  - Segment resource (*SR*)
- **Mapping Space:**
  - Segment mapping space (*SM*) = all possible segments from a layer
  - Network mapping = sequence of segment mappings covering all layers.

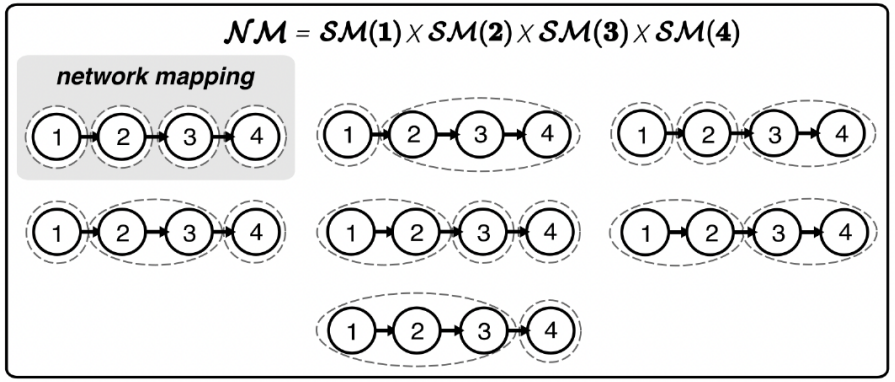




❷ OASIS Optimization Framework - Overview

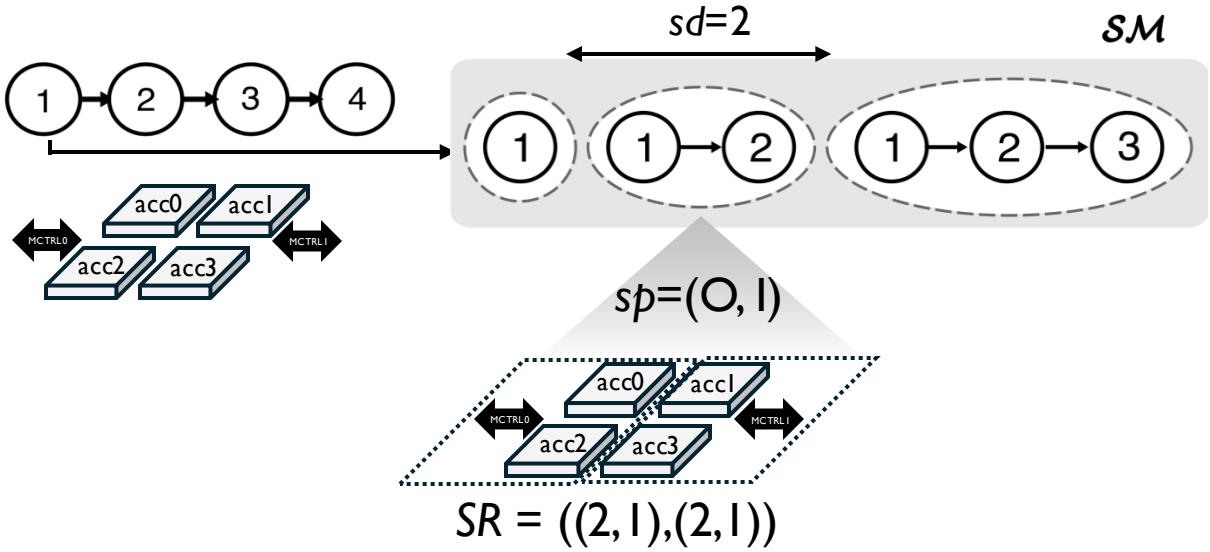
- **Goal:** Optimize segmentation, resource allocation, and scheduling for multi-model AR/VR on FLIP.
- **Inputs:**
  - DNN topologies
  - FLIP prototype
- **Segment:**
  Execution mode from layer i:
  - Segment depth (*sd*)
  - Segment parallelism (*sp*)
  - Segment resource (*SR*)
- **Mapping Space:**
  - Segment mapping space (*SM*) = all possible segments from a layer
  - Network mapping = sequence of segment mappings covering all layers.
  - Network mapping space (*NM*) = all possible network mappings

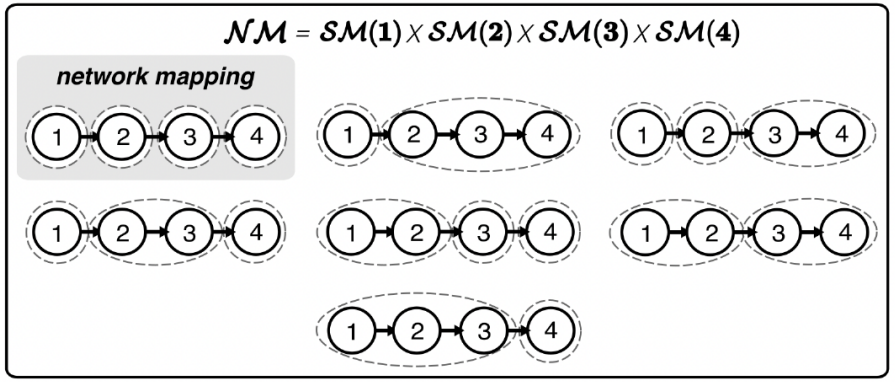




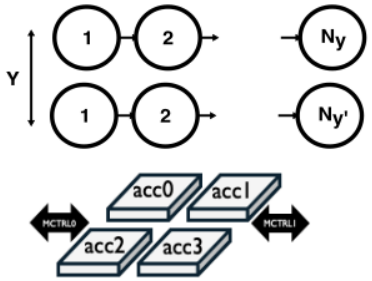
❷ OASIS Optimization Framework - Overview

- **Goal:** Optimize segmentation, resource allocation, and scheduling for multi-model AR/VR on FLIP.
- **Inputs:**
  - DNN topologies
  - FLIP prototype
- **Segment:**
  Execution mode from layer i:
  - Segment depth (*sd*)
  - Segment parallelism (*sp*)
  - Segment resource (*SR*)
- **Mapping Space:**
  - Segment mapping space (*SM*) = all possible segments from a layer
  - Network mapping = sequence of segment mappings covering all layers.
  - Network mapping space (*NM*) = all possible network mappings

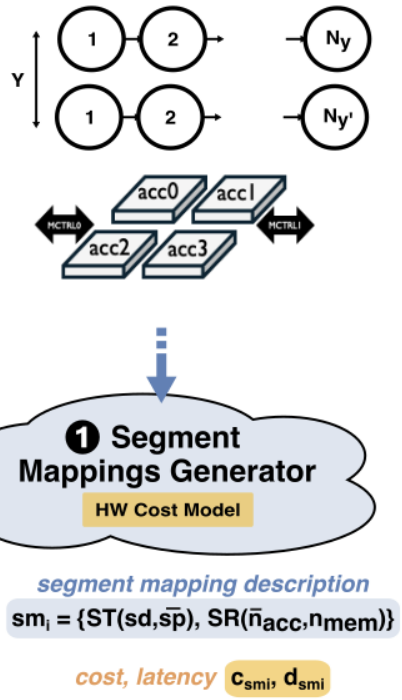




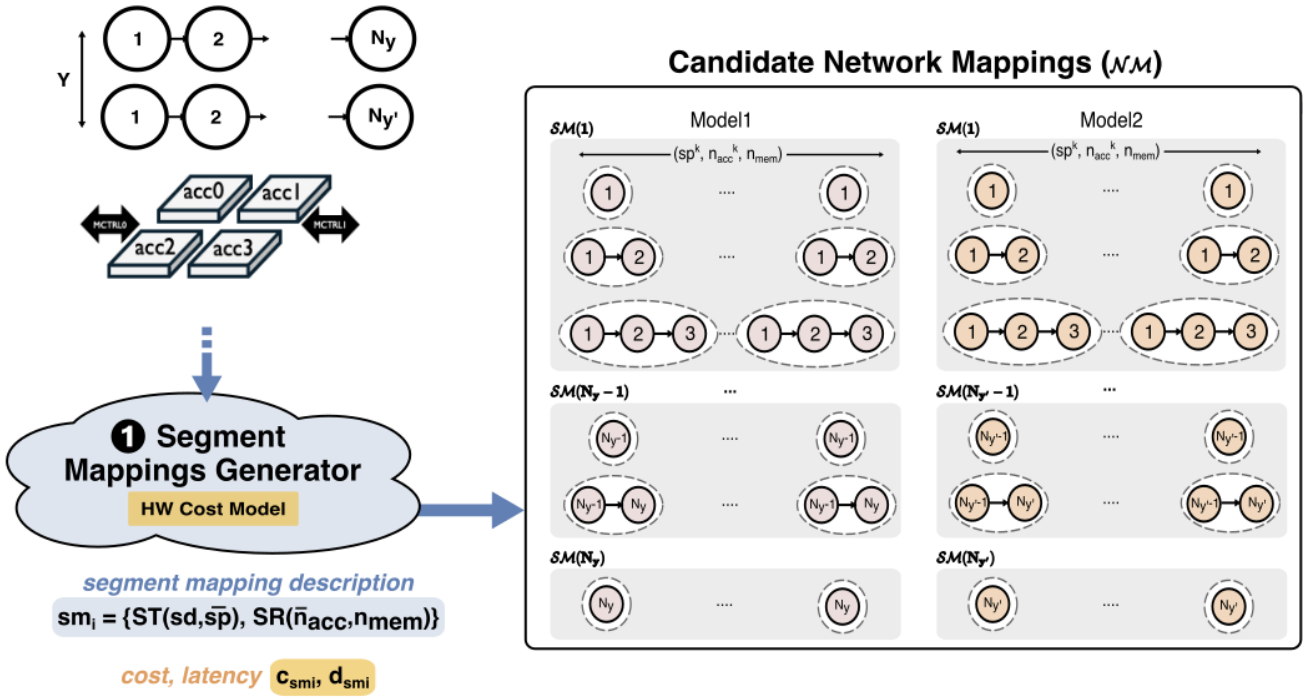
❷ OASIS Solver



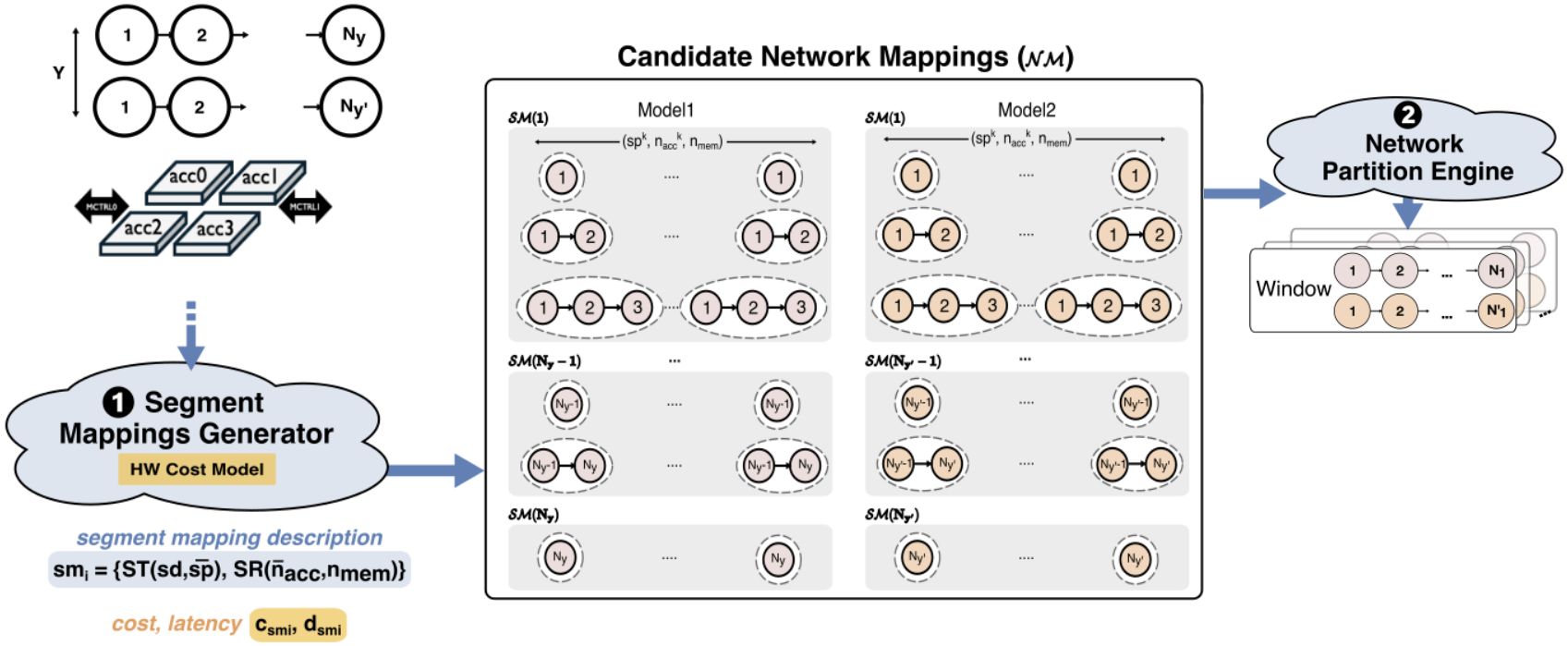
❷ OASIS Solver



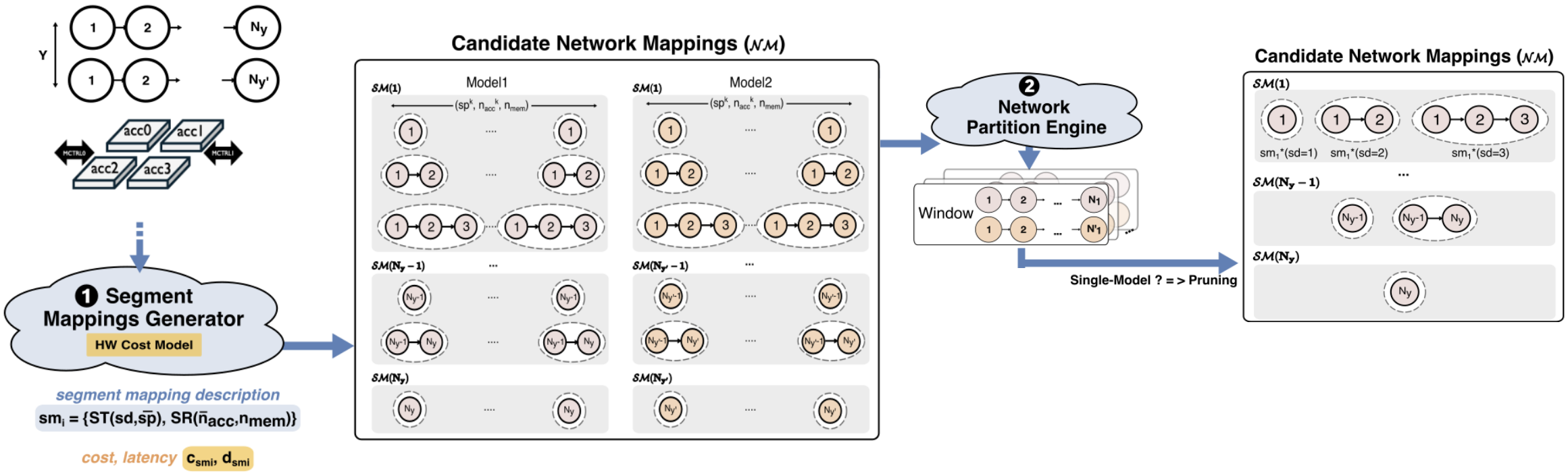
❷ OASIS Solver



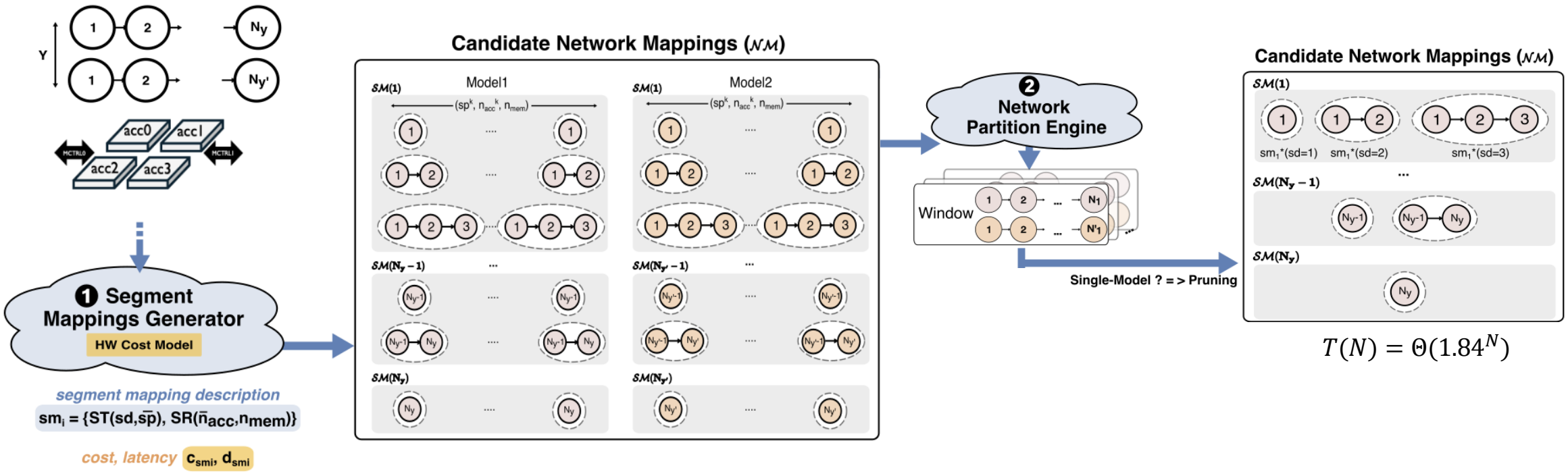
❷ OASIS Solver



❷ OASIS Solver

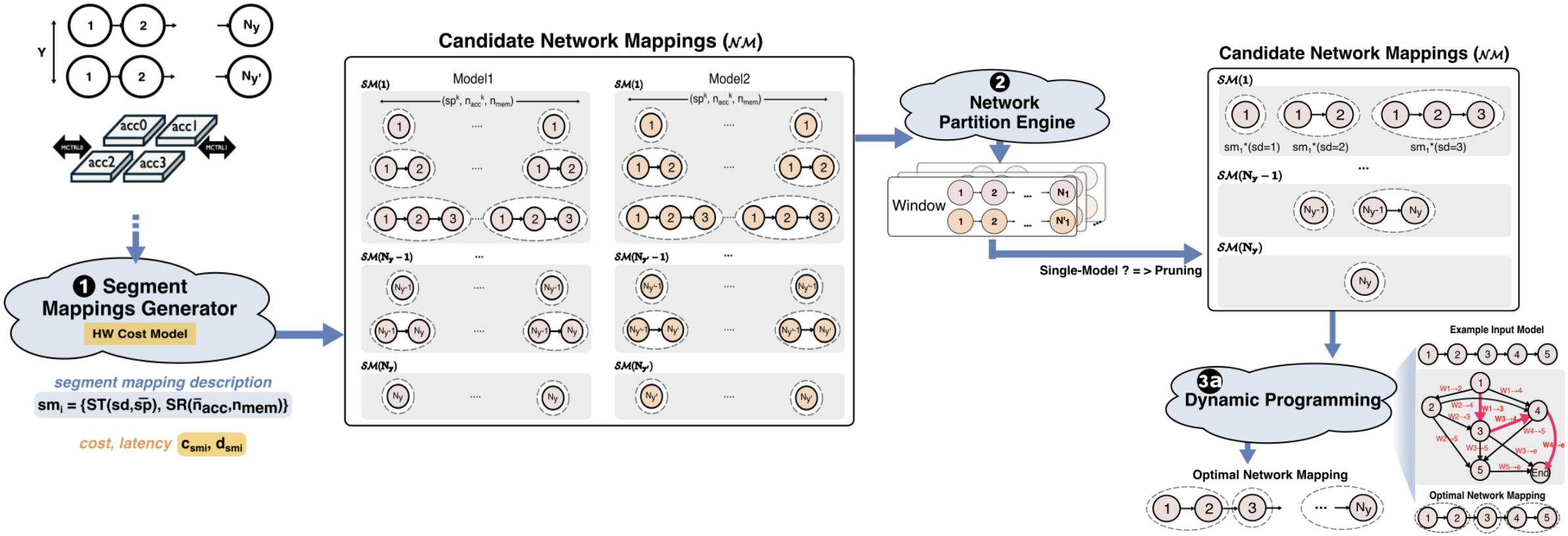


❷ OASIS Solver

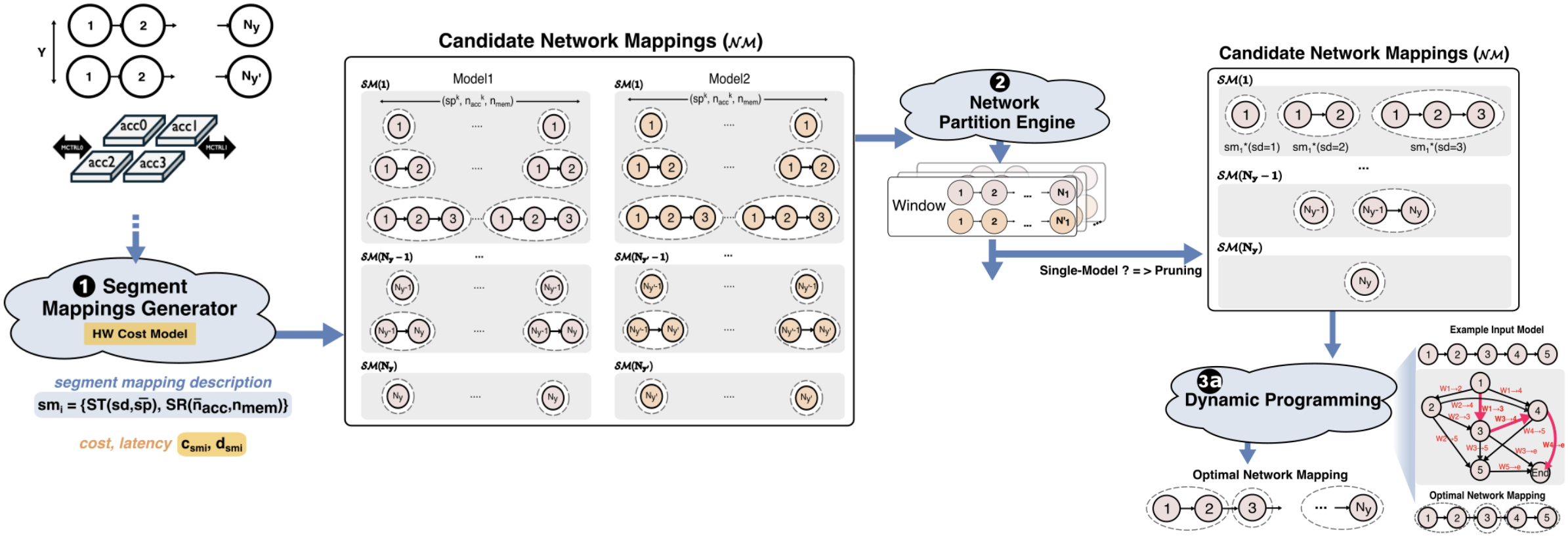


$$T(N) = \Theta(1.84^N)$$

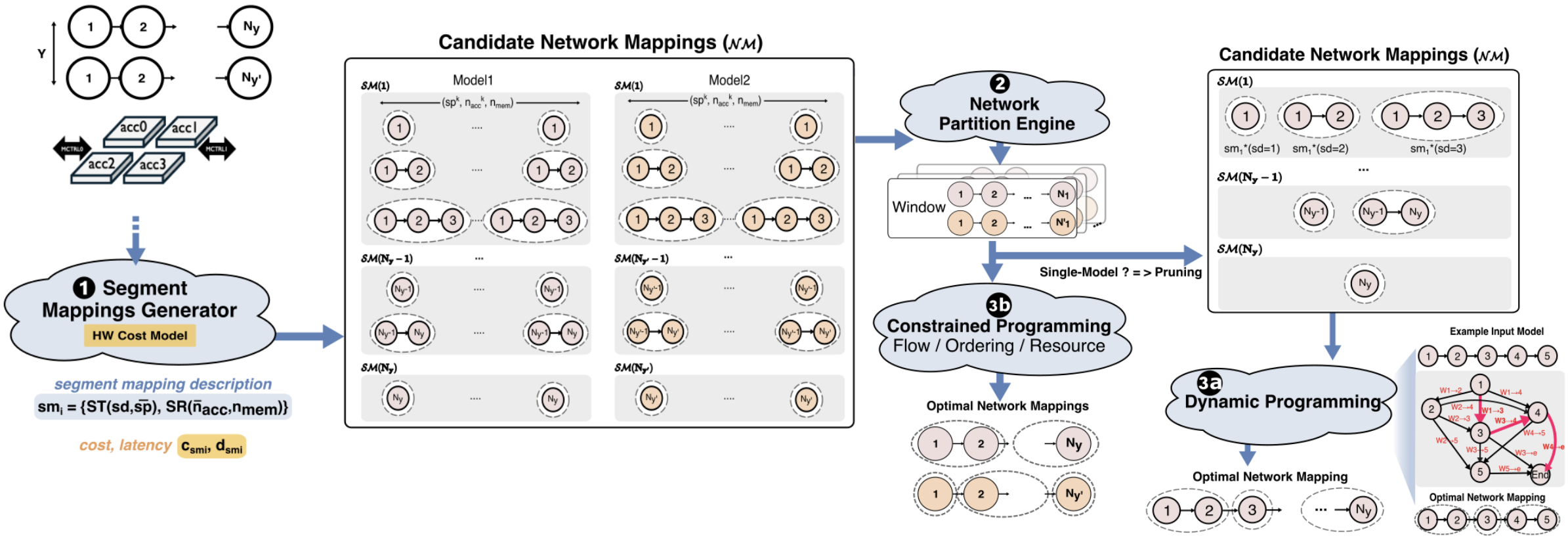
❷ OASIS Solver



❷ OASIS Solver



❷ OASIS Solver



Evaluation - FLIP Prototype

Platform: Built on open-source ESP:

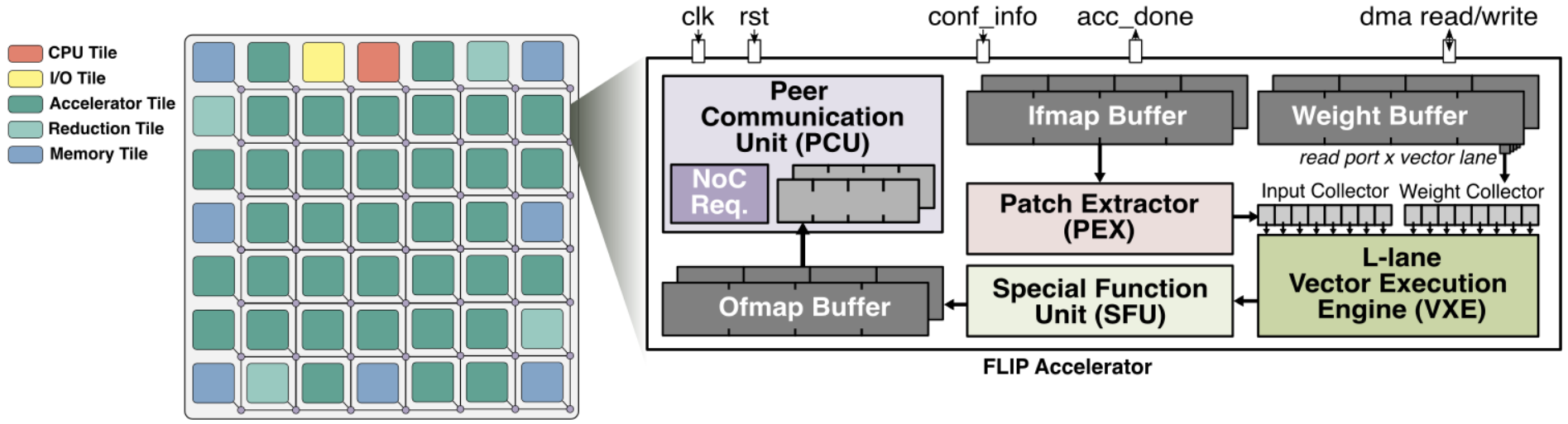
esp.cs.columbia.edu

Implemented on Xilinx XCVU19P FPGA.
Tile architecture (7×7 grid = 49 tiles): 36 acc tiles / 4 red.Tiles / 7 memory tiles / 1 processor tile (CVA6 host) / 1 auxiliary tile (peripherals).

Custom features:

- Multi-level data reuse (weights & inputs) for acc tiles.
- Dynamic communication patterns: DMA, P2P, multicast.
- Modified ESP NoC for flexible producer/consumer access + multicast support.

Resource utilization: ~48% LUTs, ~21% registers, ~60% DSPs, ~74% BRAMs.
Power: Dominated by accelerators, reduction tiles, and NoC



Evaluation – Experimental Setup

Benchmarks: Multi-model AR/VR workloads (from XRbench) → 3 scenarios combining different models & batch sizes:

AR/VR1			AR/VR2			AR/VR3		
Task	Model	Batch size	Task	Model	Batch size	Task	Model	Batch size
Gaze Estimation	ResNet-18	2	Hand Tracking	ResNet-34	4	Eye Segmentation	Vgg16	2
Keyword Detection	SqueezeNet	1	Plane Detection	Unet	2	Depth Refinement	MobileNet-v2	2
Object Detection	MobileNet-v2	2	Speech Recognition	MobileBert	1	Action Segmentation	ResNet-18	4
			Depth Estimation	ResNet-50	2	Speech Recognition	MobileBert	1

Evaluation:

- Single- vs. multi-model deployments
- Metrics include latency, energy, and EDP (from Vivado power + performance counters).

Evaluation – Multi-Model Performance

Configurations:

- Baseline → fixed per-model resources, no pipelining.
- FlexIntra1/2 → flexible per-model accelerator & memory BW allocation.
- Full → + inter-layer pipelining.

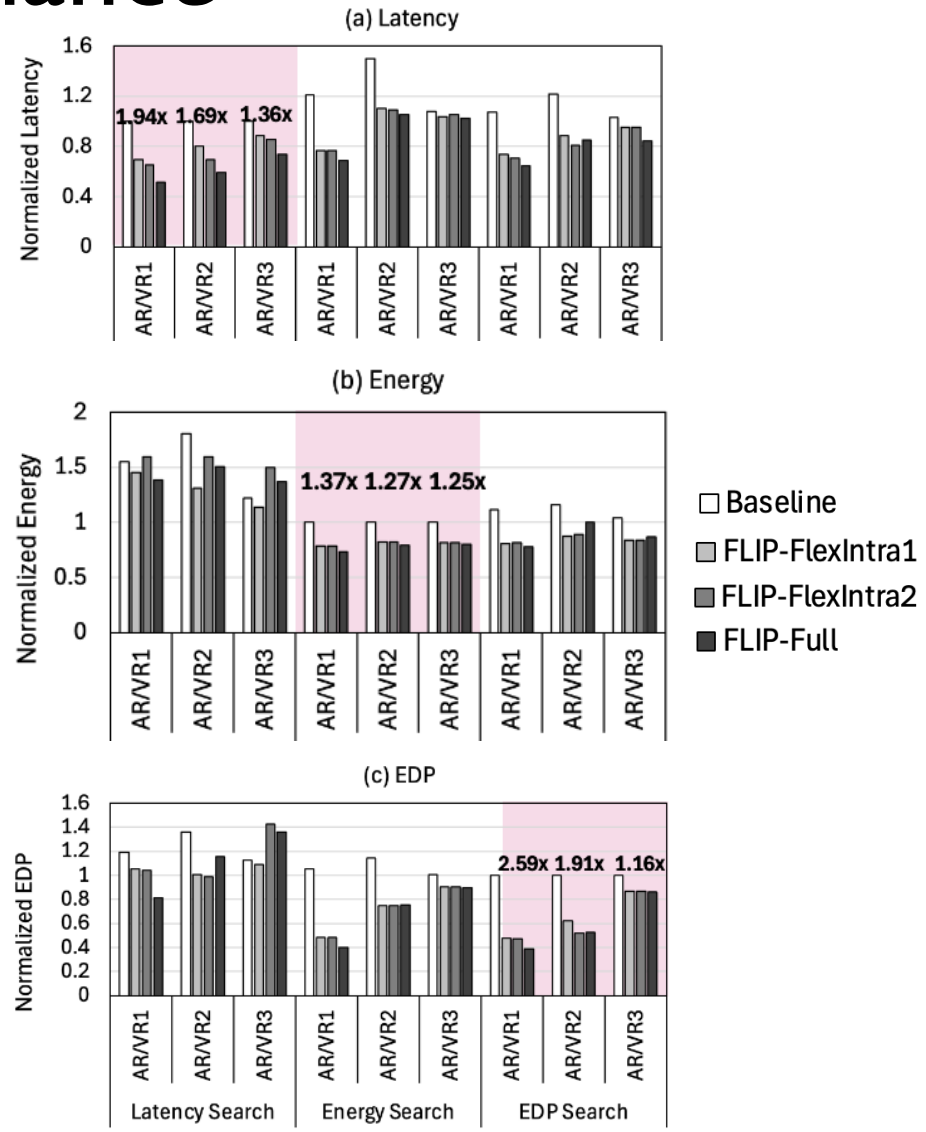
Evaluation – Multi-Model Performance

Configurations:

- Baseline → fixed per-model resources, no pipelining.
- FlexIntra1/2 → flexible per-model accelerator & memory BW allocation.
- Full → + inter-layer pipelining.

Key Results:

- **Latency:** Up to **1.9× speedup** (higher than single-model gains).
- **Energy:** FlexIntra variants give most savings (≤1.4×).
- **EDP:** Consistently improved by reduced idle cycles & better allocation



Evaluation – Multi-Model Performance

Configurations:

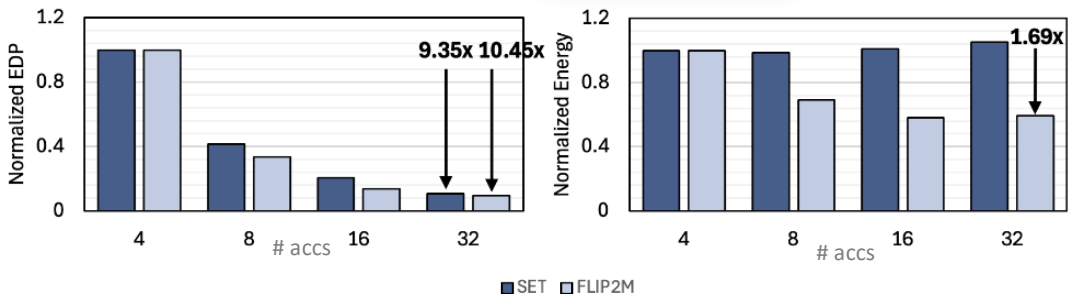
- Baseline → fixed per-model resources, no pipelining.
- FlexIntra1/2 → flexible per-model accelerator & memory BW allocation.
- Full → + inter-layer pipelining.

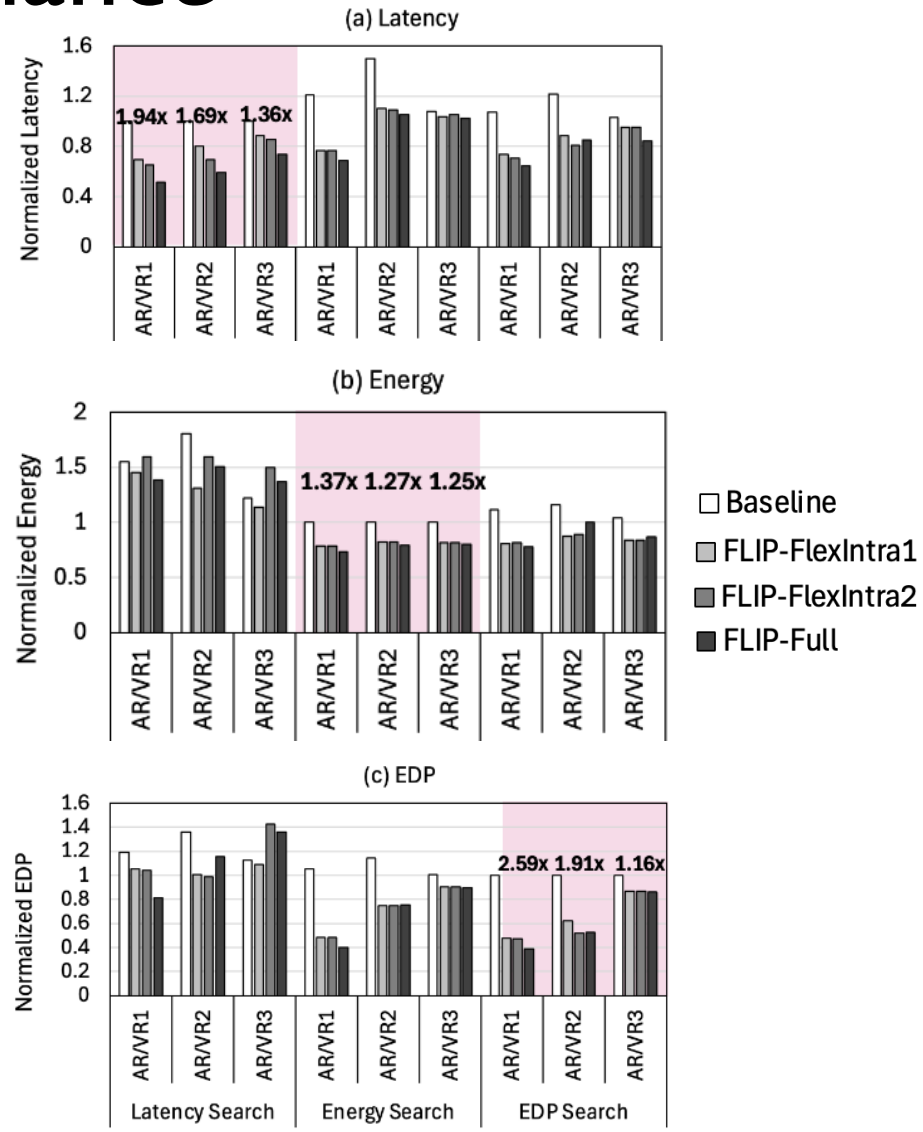
Key Results:

- **Latency:** Up to **1.9× speedup** (higher than single-model gains).
- **Energy:** FlexIntra variants give most savings (≤1.4×).
- **EDP:** Consistently improved by reduced idle cycles & better allocation

Comparison to SET [1] (state-of-the-art temporal multi-model):

- FLIP2M achieves **10.5× EDP reduction vs. 9.4×** at 32 tiles.
- Better energy scaling (up to **1.7× lower**).
- Advantage: concurrent pipelines via spatial partitioning.





[1] J . Cai, ISCA'23

Conclusions

- **FLIP2M**: integrates intra-layer parallelism + inter-layer pipelining → efficient multi-model AR/VR on tiled accelerators.

Conclusions

- **FLIP2M**: integrates intra-layer parallelism + inter-layer pipelining → efficient multi-model AR/VR on tiled accelerators.
- **FLIP accelerator fabric**: flexible architecture enabling scalable parallelism & diverse on-chip communication.

Conclusions

- **FLIP2M**: integrates intra-layer parallelism + inter-layer pipelining → efficient multi-model AR/VR on tiled accelerators.
- **FLIP accelerator fabric**: flexible architecture enabling scalable parallelism & diverse on-chip communication.
- **OASIS framework**: navigates vast mapping space for single- & multi-model workloads.

Conclusions

- **FLIP2M**: integrates intra-layer parallelism + inter-layer pipelining → efficient multi-model AR/VR on tiled accelerators.
- **FLIP accelerator fabric**: flexible architecture enabling scalable parallelism & diverse on-chip communication.
- **OASIS framework**: navigates vast mapping space for single- & multi-model workloads.
- **Prototype**: 49-tile FPGA SoC on ESP platform.

Conclusions

- **FLIP2M**: integrates intra-layer parallelism + inter-layer pipelining → efficient multi-model AR/VR on tiled accelerators.
- **FLIP accelerator fabric**: flexible architecture enabling scalable parallelism & diverse on-chip communication.
- **OASIS framework**: navigates vast mapping space for single- & multi-model workloads.
- **Prototype**: 49-tile FPGA SoC on ESP platform.
- **Results**: up to **1.9× latency speedup**, **1.4× energy reduction**, **2.6× EDP improvement** over baseline.

In Memory of Davide Giri (1990-2021)

Thank you!

FLIP2M: Flexible Intra-layer Parallelism and Inter-layer Pipelining for Multi-model AR/VR Workloads

Gabriele Tombesi
Je Yang
Joseph Zuckerman
Davide Giri
William Baisi
Luca Carloni

COLUMBIA UNIVERSITY IN THE CITY OF NEW YORK
CS @CU