This is SPATEM! A Spatial-Temporal Optimization Framework for Efficient Inference on ReRAM-based CNN Accelerator

Yen-Ting Tsou, Kuan-Hsun Chen†, Chia-Lin Yang*, Hsiang-Yun Cheng‡, Jian-Jia Chen§, Der-Yu Tsai**

*National Taiwan University, Taiwan,
†University of Twente, Netherlands,
‡Academia Sinica, Taiwan,
§Technical University of Dortmund, Germany

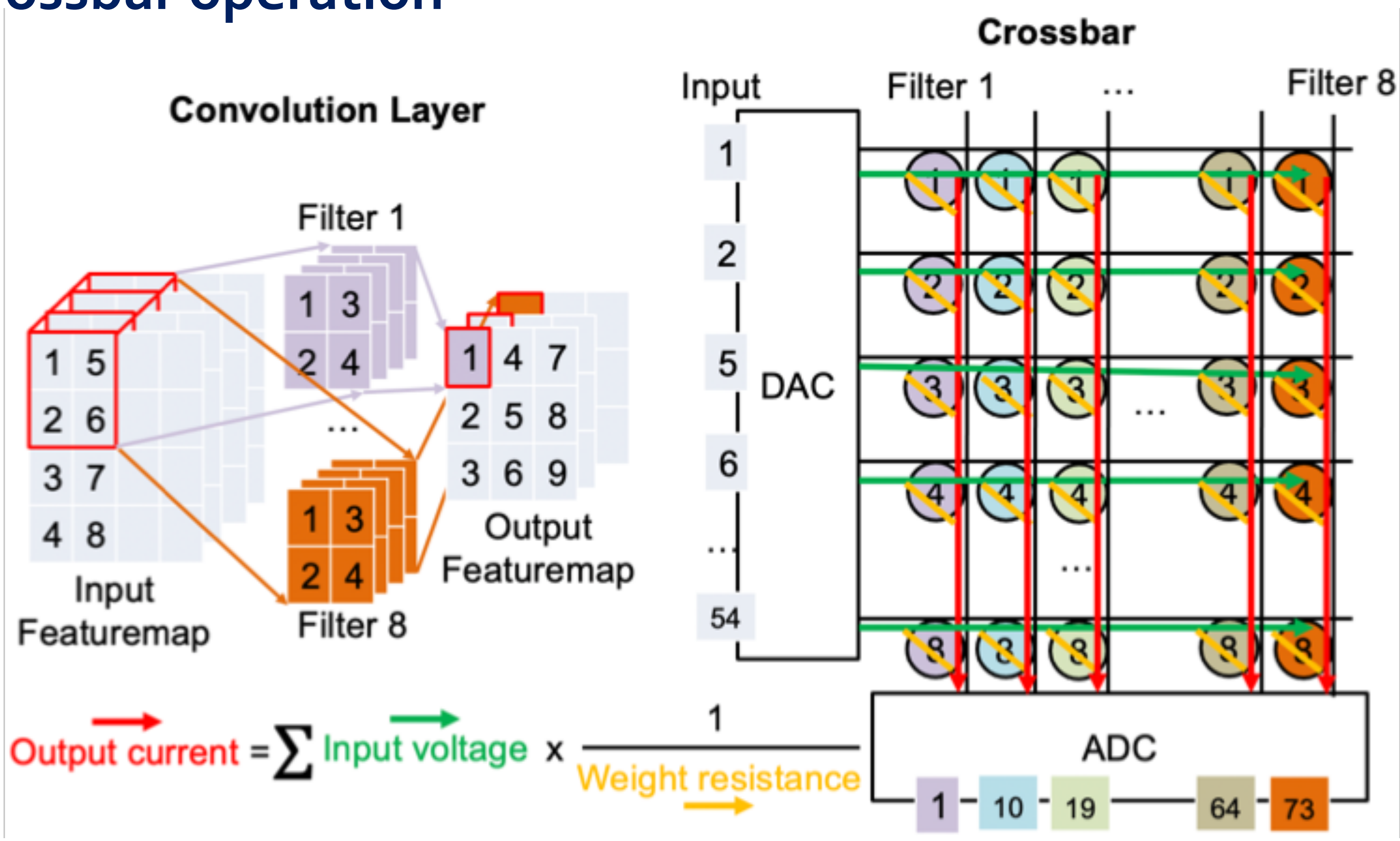
ECLab
Embedded Computing Lab.

Outline

- Introduction
- Background
- Design space exploration
- SPATEM
- Evaluation result
- Conclusion

ReRAM-based In-Memory Computing for NN Inference

- Crossbar operation

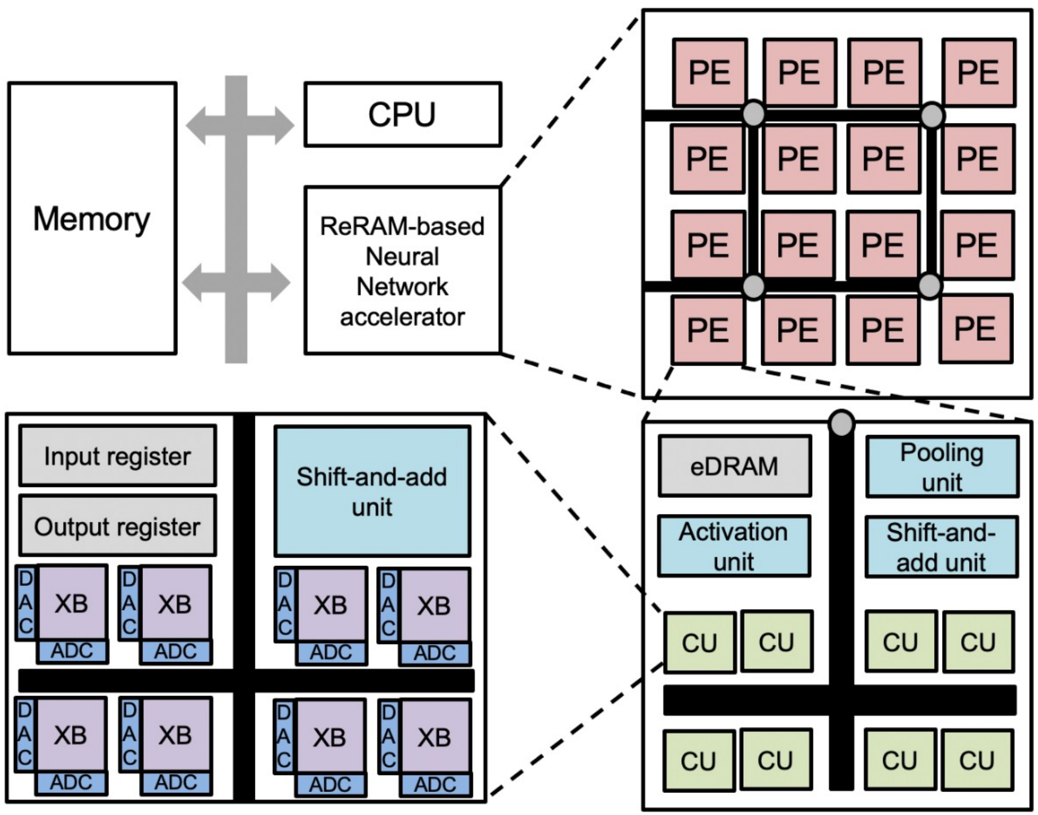


😊 Reduce data movement via weight stationary computation

😊 High degree of parallel computation

Architecture of ReRAM-based NN Accelerators

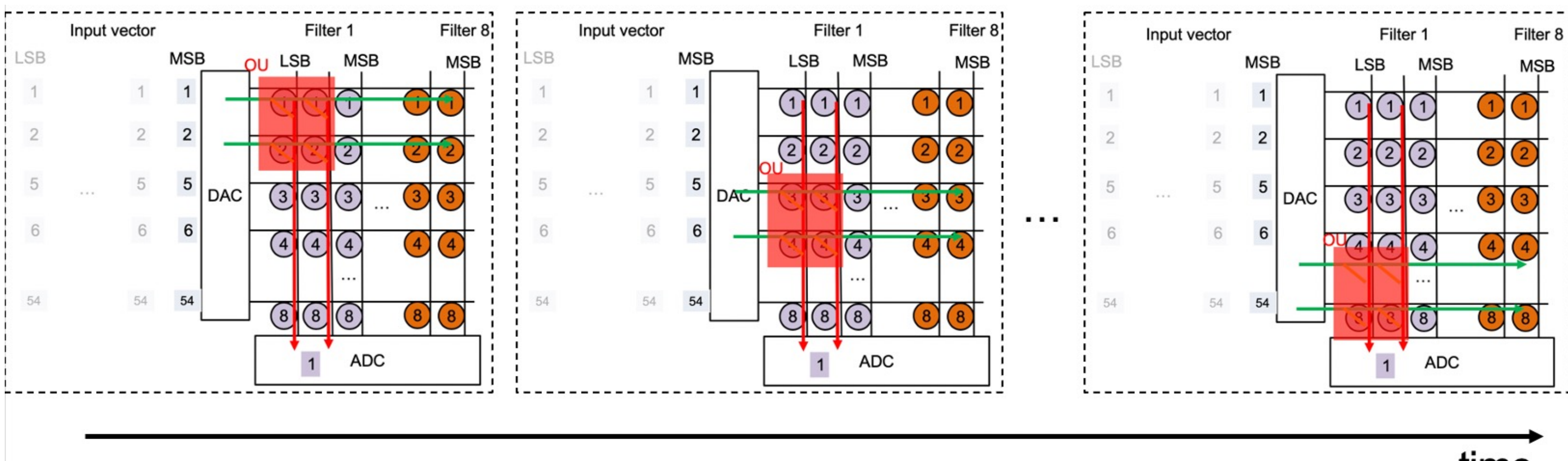
- Aggregate multiple crossbars to form a hierarchical spatial architecture
  - Enable high computation parallelism degree with lots of hardware computing unit



[ISCA'16] ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars

The Imperfect Circuit and Device

- OU-based operation
  - Limited wordlines and bitlines could operate at a time (Operation Unit)
  - Effect
    - Originally, one crossbar operation parallelly executes all weights stored in a crossbar. With considering the imperfect circuit and device, it requires multiple sequentially executed OU-based operations to complete the computation equivalent to one crossbar operation.

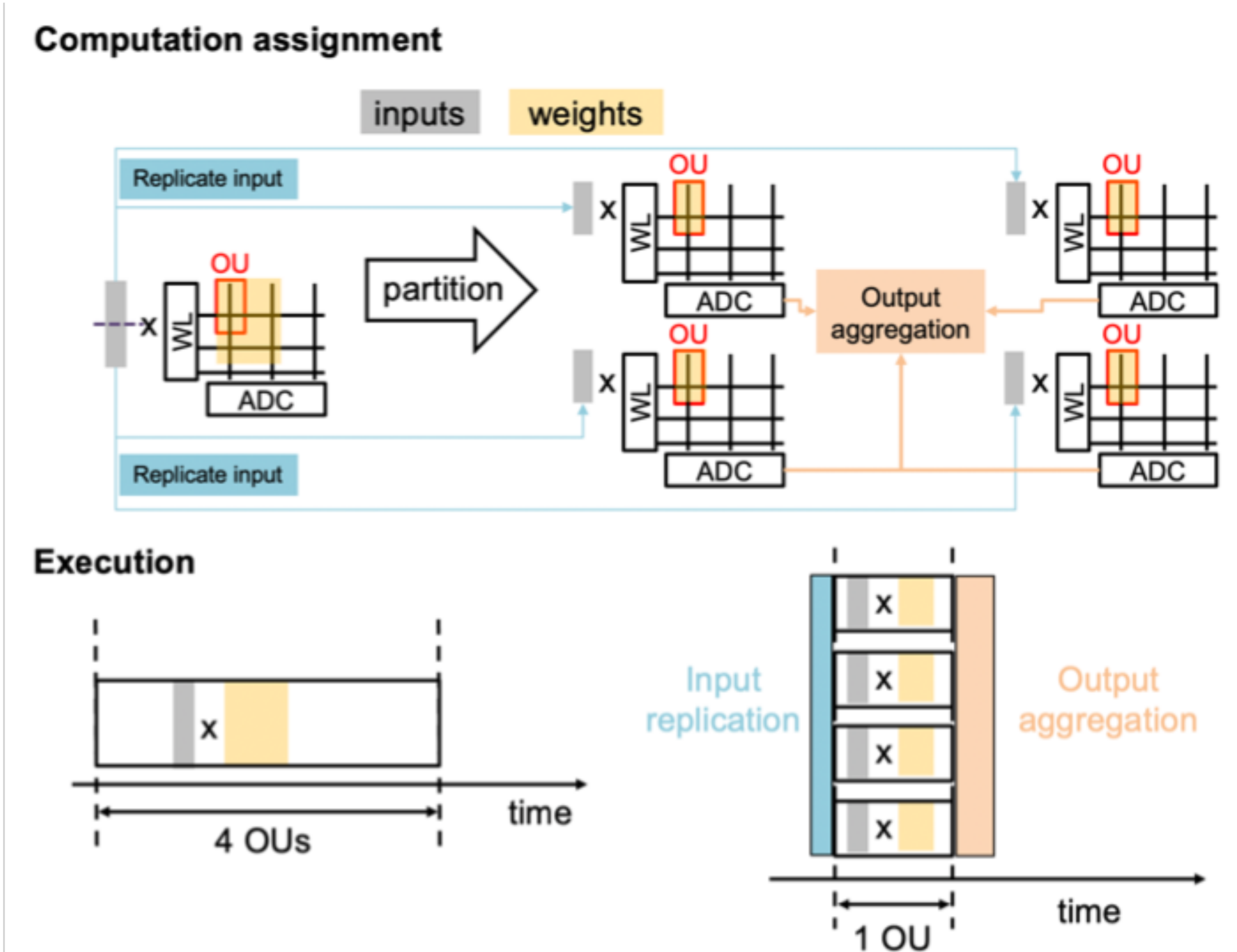


OU-based operations

[ICCAD'18] DL-RSIM: a simulation framework to enable reliable ReRAM-based accelerators for deep learning

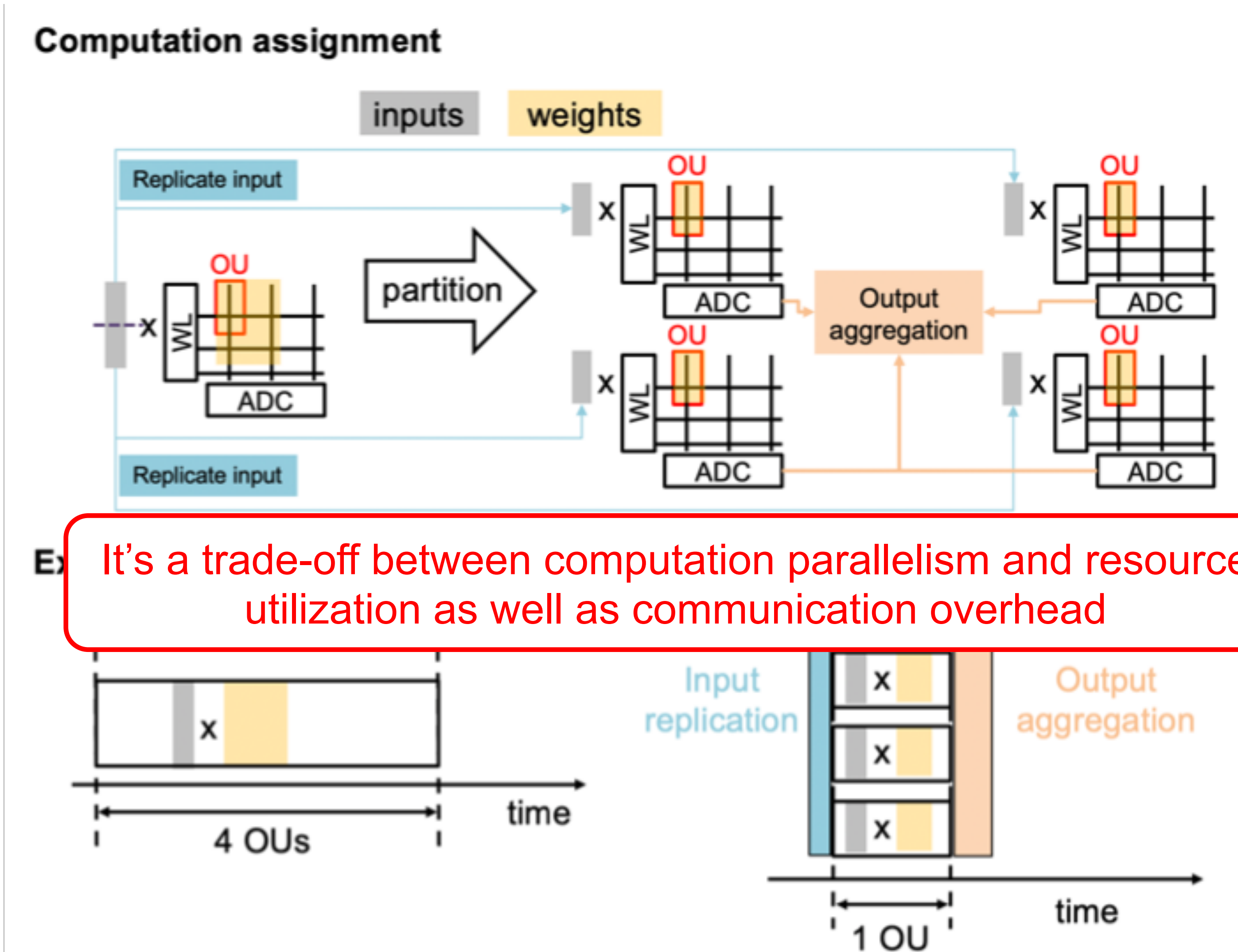
Impact of OU-based Operation

- non OU-based partitioning vs OU-based partitioning



Impact of OU-based Operation

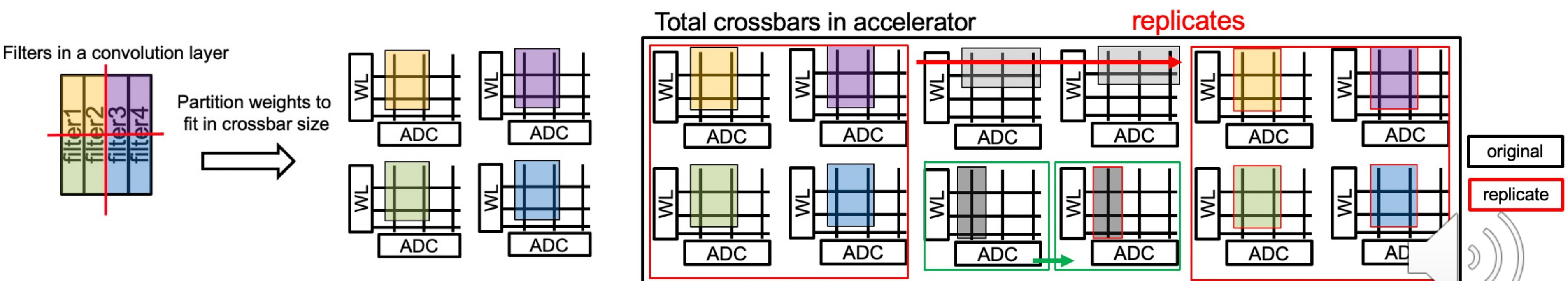
- non OU-based partitioning vs OU-based partitioning



It's a trade-off between computation parallelism and resource utilization as well as communication overhead

Limitation on Previous Work

- There are two steps in the weight deployment strategies to optimize execution time
  - Partition weights to fit in crossbar size
  - Replicate partitioned weights to utilize unused crossbars
    - ISAAC [1] proposed that the number of replicates is proportional to the throughput of layers.
    - HitM [2] proposed a dynamic-programming method to decide the number of replicates for all layers.

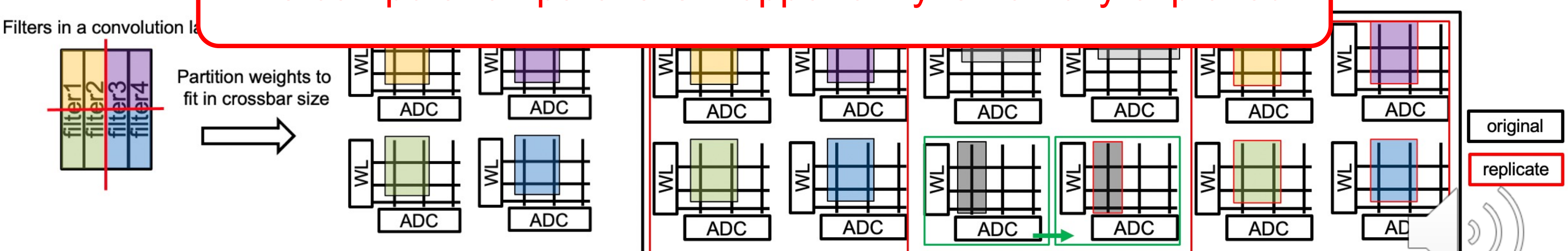


1. [ISCA'16] ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars
2. [ICCAD'20] HitM: High-Throughput ReRAM-based PIM for Multi-Modal Neural Networks

Limitation on Previous Work

- There are two steps in the weight deployment strategies to optimize execution time
  - Partition weights to fit in crossbar size
  - Replicate partitioned weights to utilize unused crossbars
    - ISAAC [1] proposed that the number of replicates is proportional to the throughput of layers.
    - HitM [2] proposed a dynamic-programming method to decide the number of replicates for all layers.

The computation parallelism opportunity is not fully exploited.



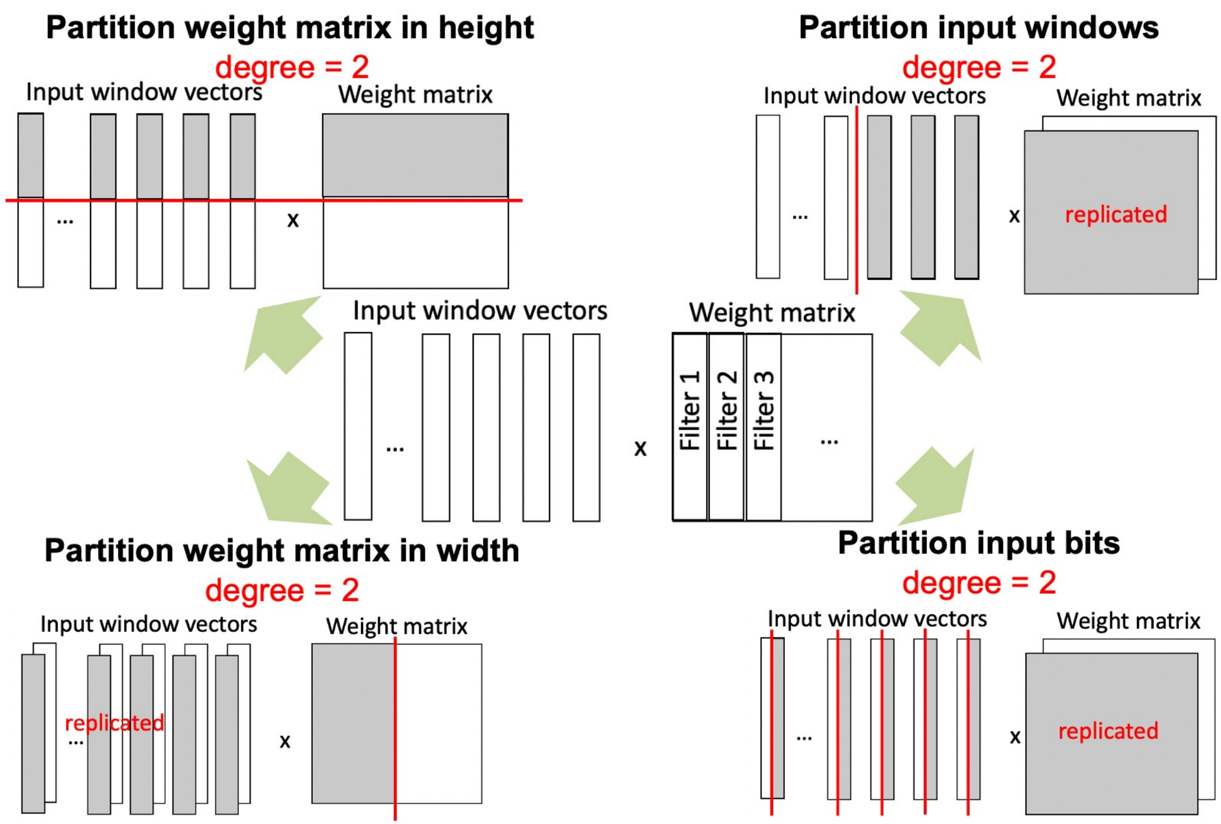
1. [ISCA'16] ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars
2. [ICCAD'20] HitM: High-Throughput ReRAM-based PIM for Multi-Modal Neural Networks

Design Space Exploration

- The design space for the deployment of CNN inference on ReRAM-based accelerator can be divided into two parts:
  - The *spatial deployment* copes with the mapping of weight values and multiplication results onto crossbar cells. In our studied problem, one crossbar cell stores one weight value and generates one multiplication result.
  - The *temporal deployment* deals with the execution order of MVMs. In our studied problem, MVMs are partitioned into multiple parts on different crossbars, so the execution order should preserve the data dependencies between adjacent layers.

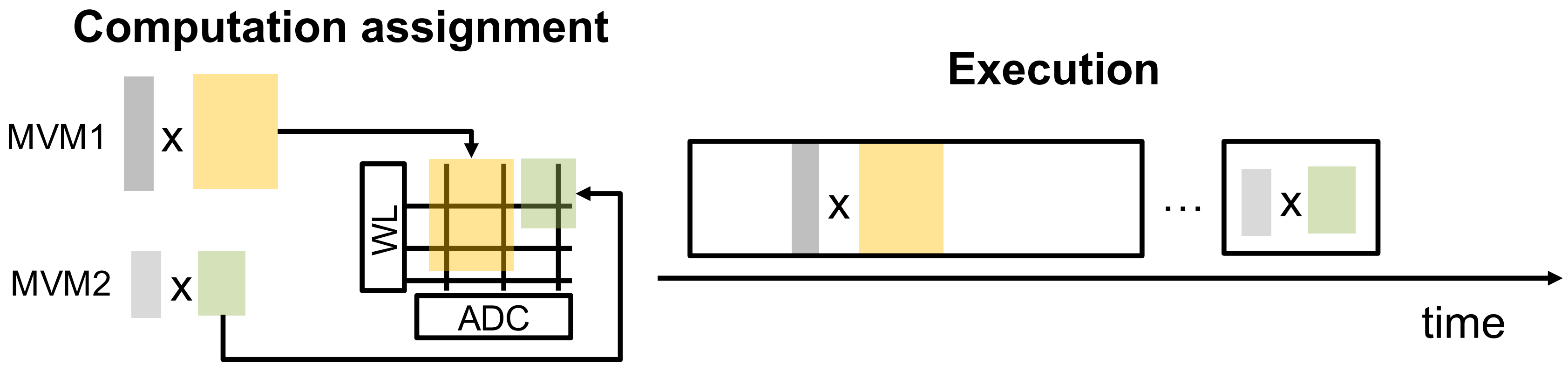
Spatial Deployment

- Three steps to decide the spatial deployment in our framework.
  - Step 1. The partitioning step decides how to partition the MVMs and how many parts should be partitioned.



Spatial Deployment (cont.)

- Three steps to decide the spatial deployment in our framework.
  - Step 2. The packing step decides how to pack partitioned MVMs onto virtual crossbars.



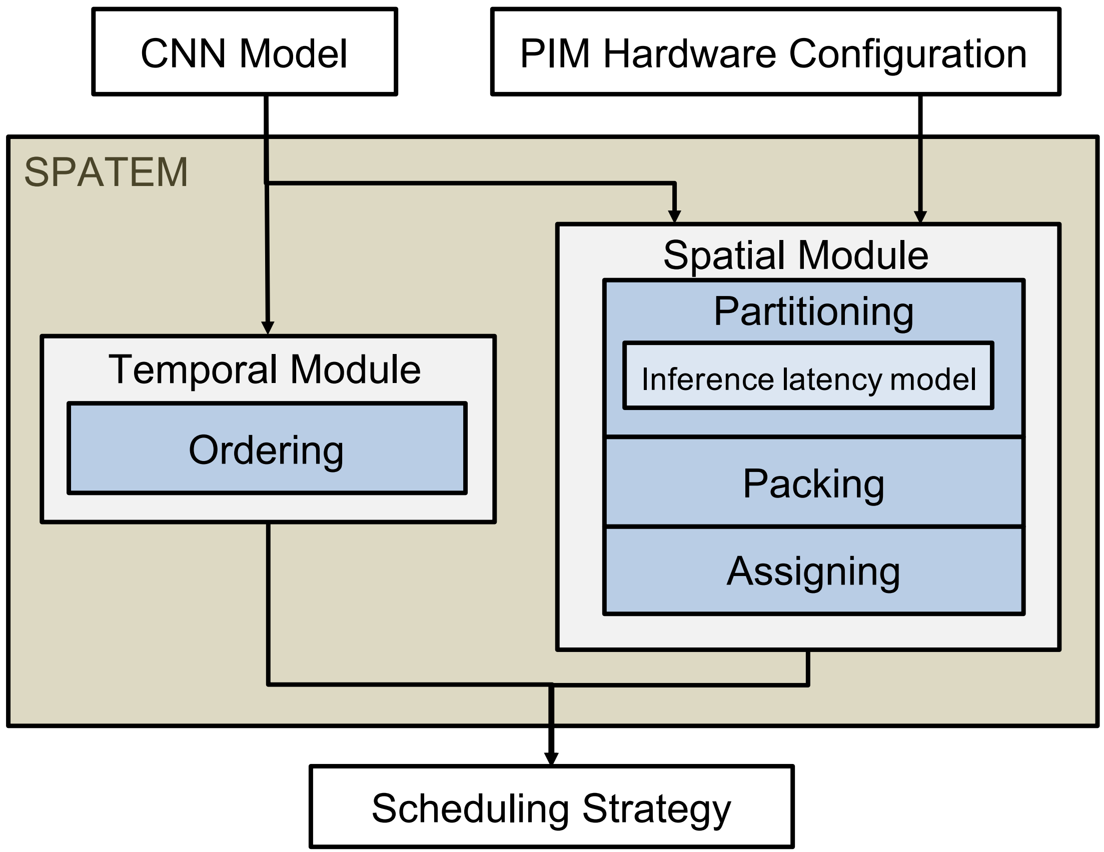
  - Step 3. The assigning step decides the assignment of virtual crossbars onto physical crossbars.

Temporal Deployment

- The Hardware components on the ReRAM-based CIM architecture follow the generated script to execute all MVM computations in a predefined order.
  - The ordering step decides the MVM execution sequence to ensure that the precedence constraint imposed by the data dependencies between MVMs must be preserved.

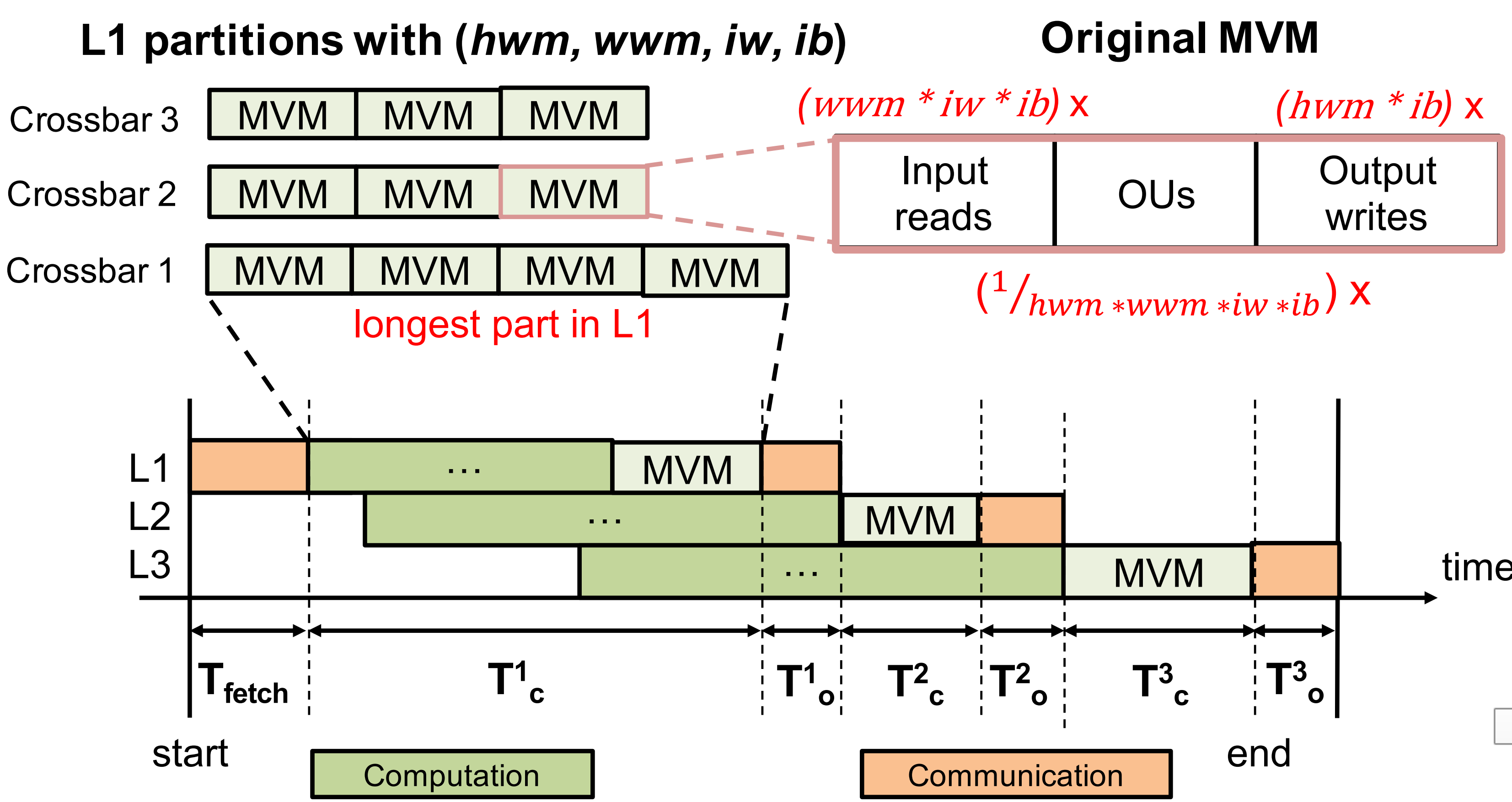
SPATEM

- Input: pretrained CNN model and PIM hardware configuration.
- Output: Scheduling strategy to drive the execution.



Partitioning Step

- Partitioning step
  - Inference latency estimation model based on a partition strategy.



Partitioning Step (cont.)

- Decide the partition strategy with the best inference latency.
- Dynamic-Programming Algorithm for Relaxed Bounded Knapsack Problem
  - Capacity: relaxed available crossbars
  - Bounded items: all layers' partition dimensions
  - Value of items: inference latency reduction

Algorithm 1: Partitioning Algorithm

Input: Number of layers L, number of available crossbars C
Output: Best partitioning strategies table ILR
Initialize ILR$[x][y]$ to 0, $\forall x \in [0, L \times 4)$, $y \in [0, L \times C)$;
for $l \leftarrow 0$ ***to*** $L-1$ **do**
    for $d \leftarrow \{hwm = 0, wwm = 1, iw = 2, ib = 3\}$ **do**
        for $r \leftarrow 0$ ***to*** $L \times C$ **do**
            tmp_ILR = ILR$[l \times 4 + d][r]$;
            while *True* **do**
                increase layer l's dimension d with new requirement tmp_r and latency reduction tmp_ILR.
                if $tmp\_r > L \times C$ **then**
                    break;
                else
                    if $tmp\_ILR > ILR[l \times 4 + d + 1][tmp\_r]$ **then**
                        ILR$[l \times 4 + d + 1][tmp_r]$ = tmp_ILR
                    end
                end
            end
        end
    end
end

Packing Step

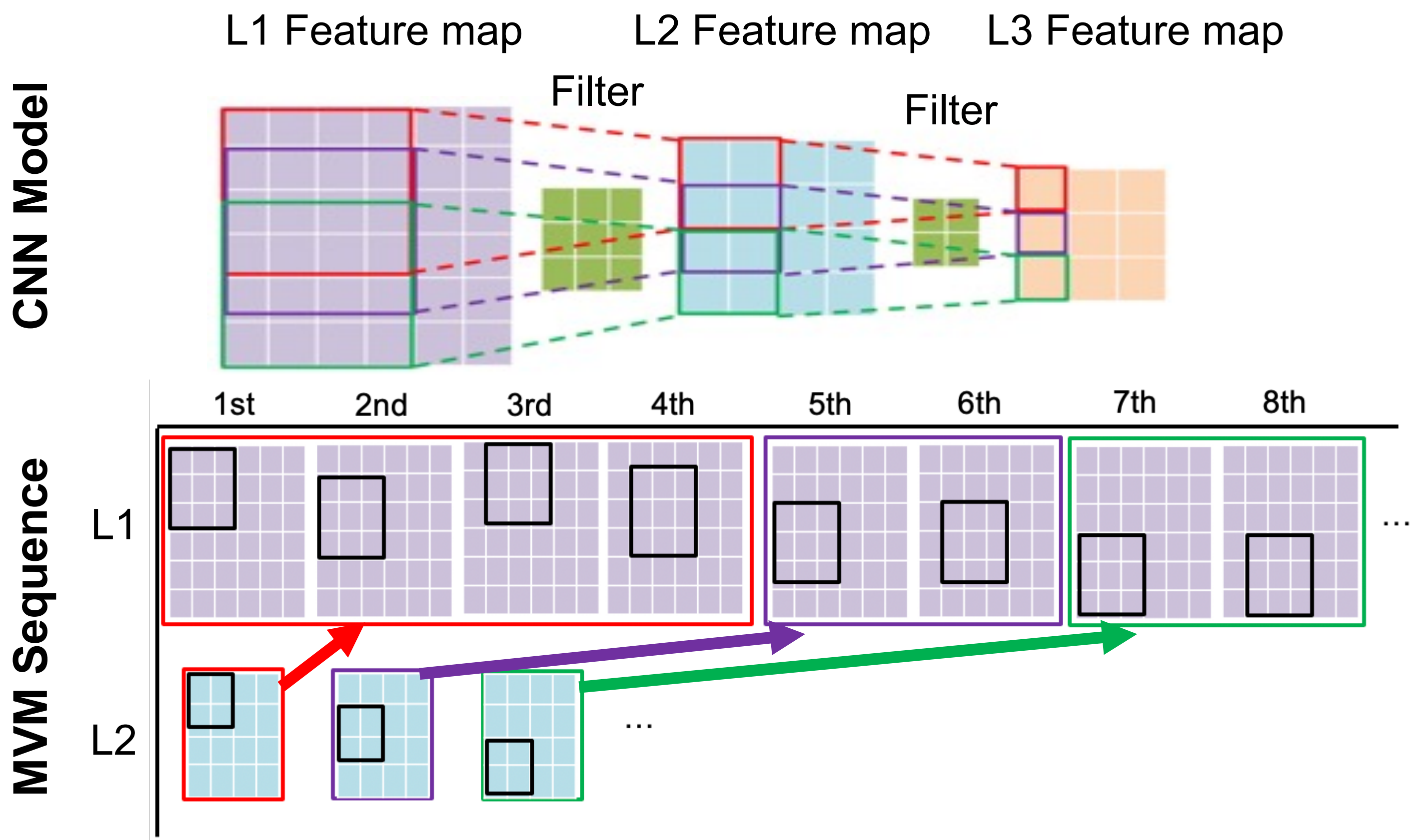
- Decide how to arrange virtual crossbars based on the partition degrees to meet resource limitation.
- Greedily try the partitioning strategies found by the previous step in inference latency ascending order.
  - Greedy Worst-fit bin-packing algorithm
    - Sequentially pack partitioned MVMs to the emptiest virtual crossbar.
    - Virtual Crossbar: a set of computations that will be assigned to one unique physical crossbar.

Assigning Step

- Decide the assignment of virtual crossbars onto physical crossbars.
- Sequentially place the virtual crossbar with the most amount of shared data to physical crossbar in ascending order.
  - Step 1 – merge virtual crossbars to virtual CUs
  - Step 2 – merge virtual CUs to virtual PEs
  - Step 3 – assign virtual PE to physical PE in ascending order of distance between physical PEs.

Ordering Step

- Decide the MVM sequence.
- Arrange each layer's MVM to fulfill data dependency as soon as possible.



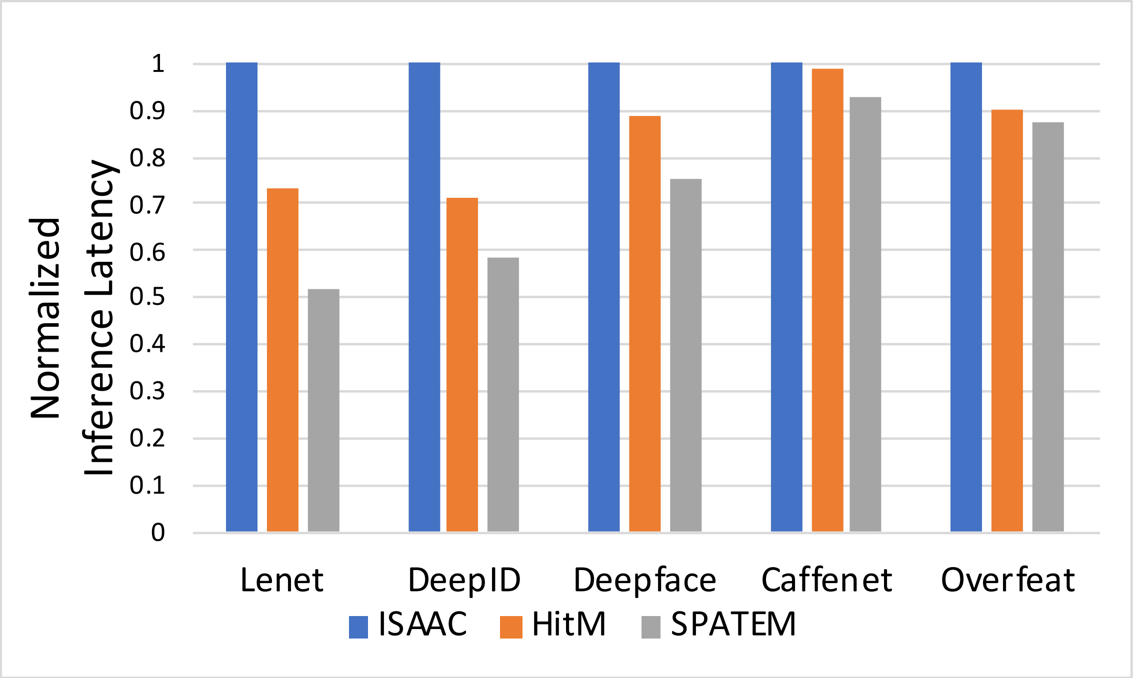
Experimental Setup

- We use an in-house event-driven simulator to estimate the inference latency of all scheduling strategies.
- CNN models: Lenet, DeepID, Deepface, Caffenet, Overfeat
- Hardware Configuration:
  - 12 x 14 PEs, 12 CUs per PE, 8 128x128 Crossbars per CU.
  - OU size: 9 x 8
  - 1-bit DAC, 2-bit ReRAM cell
- Comparison objects
  - To fairly evaluate our work, we implement ISAAC[1] and HitM[2] with adapting the throughput models to consider OU

1. [ISCA'16] ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars
2. [ICCAD'20] HitM: High-Throughput ReRAM-based PIM for Multi-Modal Neural Networks

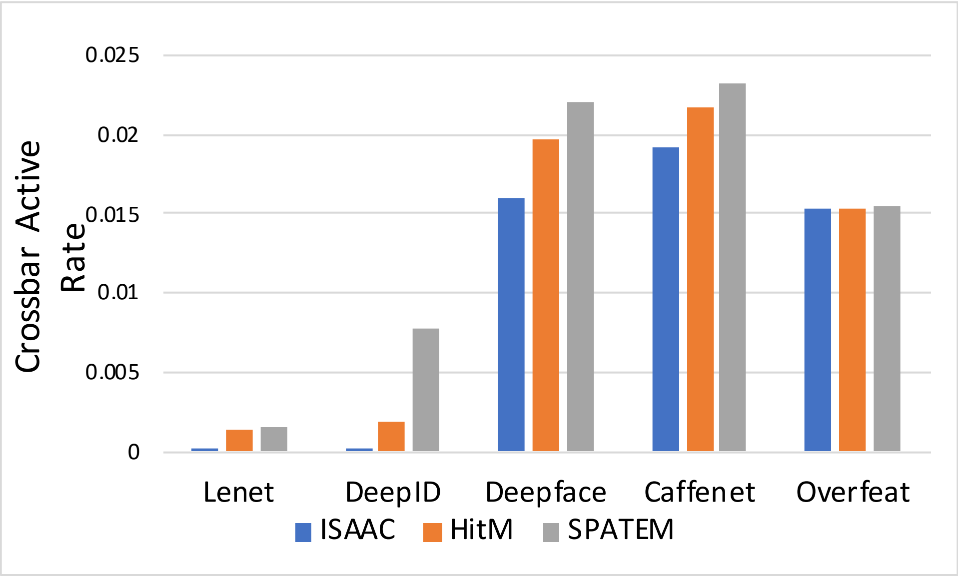
Evaluation Result

- Inference Latency
  - SPATEM achieves 29.24% improvement in average.
  - In general, the larger the given CNN structure is, the lower the improvement is.



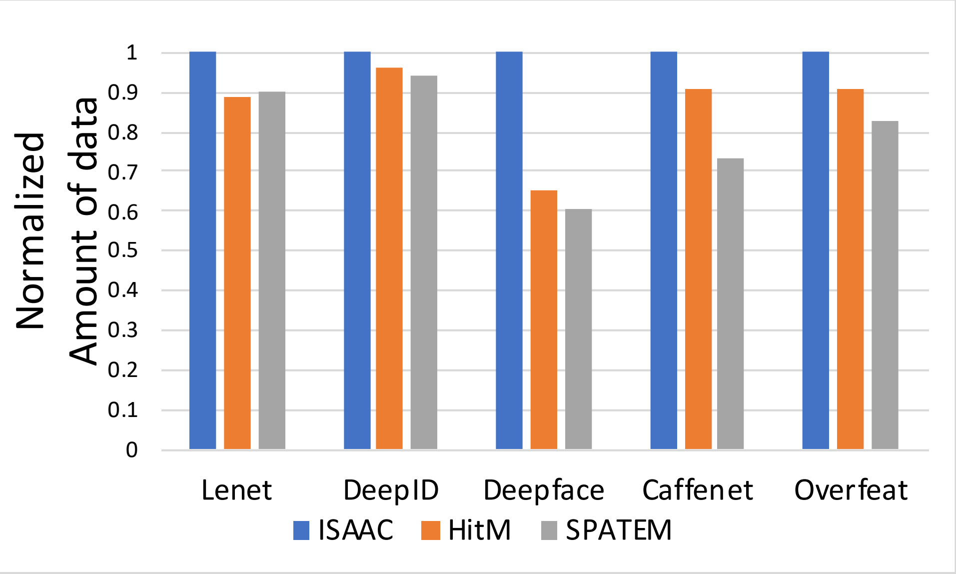
Evaluation Result (cont.)

- Computation parallelism
  - SPATEM increases computation parallelism.
  - Allowing storing weights of different layers in a single crossbar through the packing step significantly increases the computation parallelism since there are more effective crossbars to be allocated for computations of layers.



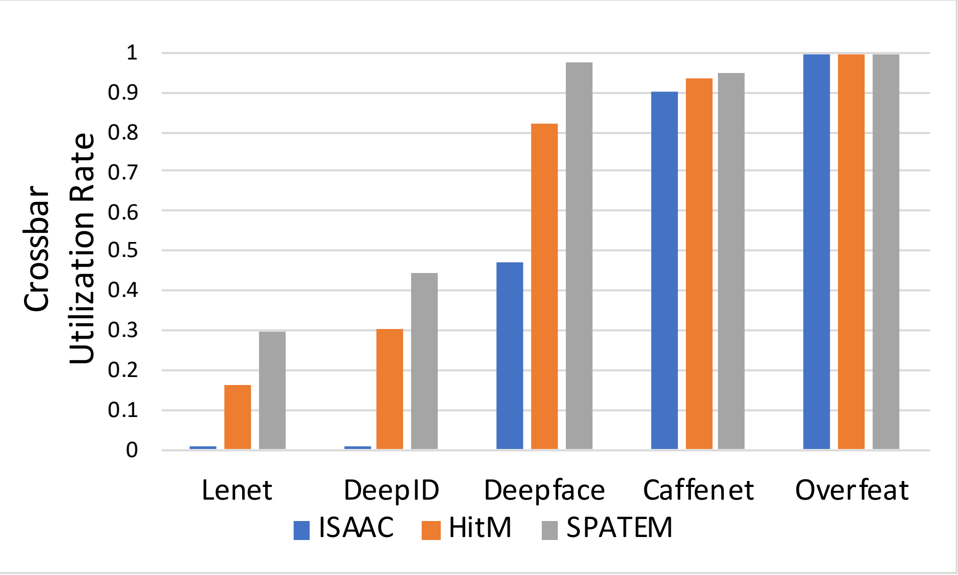
Evaluation Result (cont.)

- Communication overhead
  - SPATEM reduces communication overhead.
  - We observe that the amount of data when executing Deepface has a different trend from other large CNNs. One explanation is that Deepface possesses a great number of neurons in fully connected layers, producing lots of intermediate data with partitioning weights.



Evaluation Result (cont.)

- Resource utilization
  - SPATEM utilizes more available resources.
  - When the size of NNs is smaller like Lenet and DeepID, the improvement is not significant since the maximal parallelism degree is reached while leaving lots of unused crossbars.



Conclusion

- SPATEM shows that spatial and temporal issues must be overcome on OU-based ReRAM accelerators. Our framework decouples the design space into tractable steps, models the expected inference latency for partitioned MVMs, and addresses each step thoughtfully.
- Comparing to the state-of-the-arts, we show that the derived scheduling strategy over five representative CNNs achieves 29.24% inference latency reduction on average, by utilizing 3.19x more originally unused crossbar cells with 31.28% less communication overhead.

THANKS FOR LISTENING