Enabling Fine-Grained Spatial Multitasking on Systolic-Array NPUs Using Dataflow Mirroring

Jinwoo Choi, Yeonan Ha, Junghoo Lee,
Sangsu Lee, Jinho Lee, Hanhwi Jang, and Youngsok Kim
Yonsei University

IEEE Transactions and Computers (**TC**), Aug. 2023

Systolic-Array NPUs

- Run Neural Network (NN) operations on a **systolic array**
  - A two-dimensional array of Processing Elements (PEs)
  - PEs execute one Multiply-ACcumulate (MAC) operation per cycle.

- Suited for **matrix multiplication**, a key operation in NNs
  - e.g., convolutional and fully-connected layers

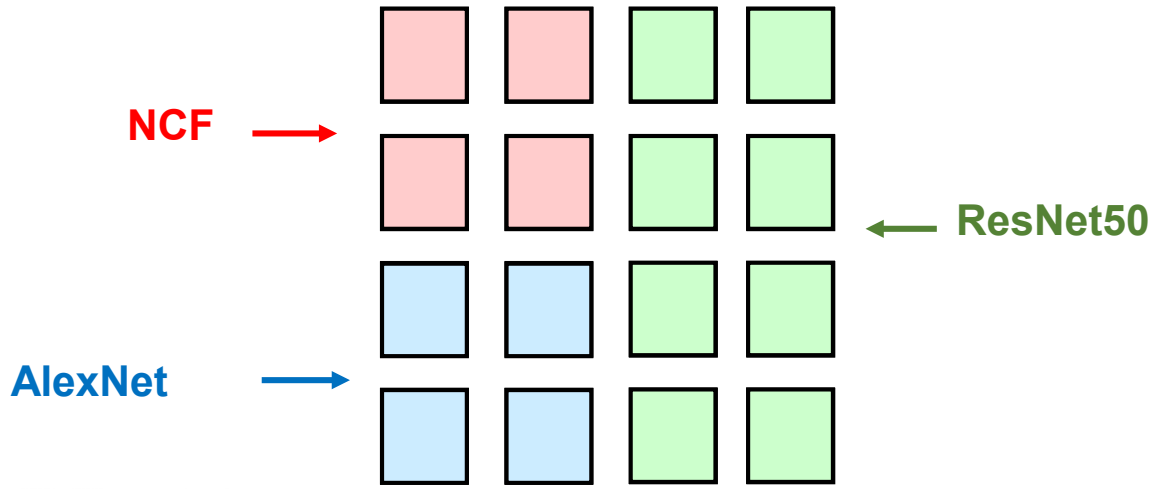
Underutilized Hardware Resource

- Prior NPUs allocate systolic array to **one NN at a time**.
  - e.g., single-NN execution, temporal multitasking
- **Highly difficult** to fully utilize NPU with only a single NN
  - # of output channels ≥ PE width, Filter size ≥ PE height
- Prior NPU shows the low HW resource utilization
  - **22.0%** of PEs and **33.4%** of the off-chip DRAM bandwidth

We need **Spatial Multitasking** to improve HW utilization!

Spatial Multitasking on NPUs

- **Co-locate multiple NNs** on the same systolic array
- Advantages
  - Higher **HW utilization**
  - Higher **multi-program performance**
    - System throughput (STP)
    - Average normalized turn-around time (ANTT)



Limitations of Prior Work

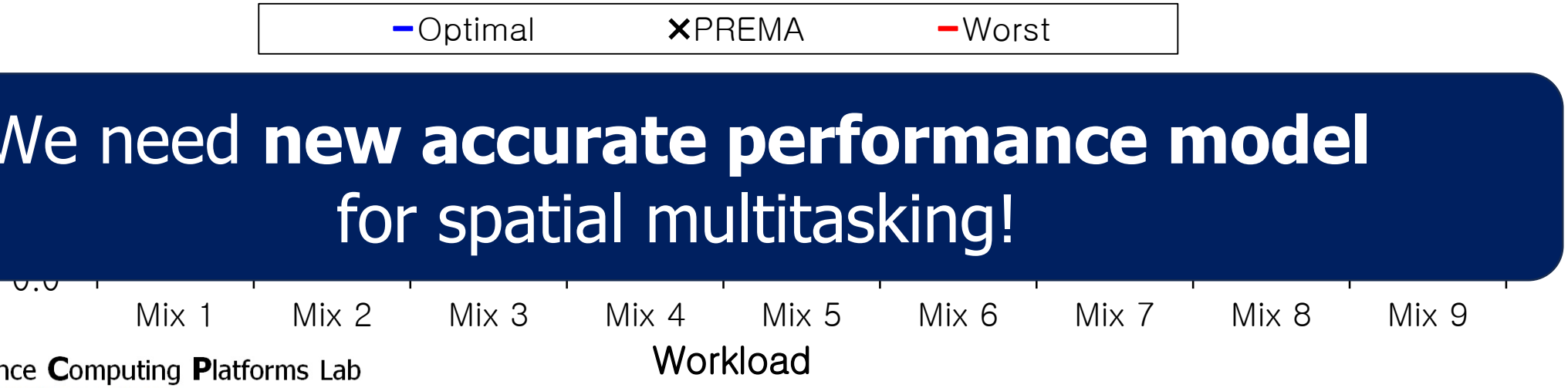
- **Coarse-grained** systolic array allocation
  - Partition the systolic array into multiple sub-arrays
    - e.g., 128x128 systolic array → 4 64x64 sub-arrays
  - Allocate the sub-arrays to co-located NNs

- **High hardware cost**
  - e.g., all-to-all high radix crossbar

Need a new flexible, fine-grained systolic-array allocation for **fine-grained** spatial multitasking on NPUs

Lack of Shared NPU Resource Modeling

- The optimal NPU resource allocation is crucial
  - Allocate to maximize the performance benefits of spatial multitasking
  - Consider DNN's characteristic
- Existing performance model results in sub-optimal alloc.
  - It achieves much lower STP than the optimal allocation
- It doesn't consider the contention on the NPU resource



We need **new accurate performance model** for spatial multitasking!

Goal: "Fine-Grained" Multitasking

- **Fine-grained** systolic-array allocation granularity
  - Systolic array allocation should not be bounded by sub-arrays.
- **Low** hardware implementation cost
  - Easily employ the new architecture to the existing NPUs
- Support a **high number of co-located NNs**
  - Maximize the performance of spatial multitasking
- **High accurate** performance model
  - Find optimal allocation, considering HW resource contention

Key Idea: Reverse the Dataflows

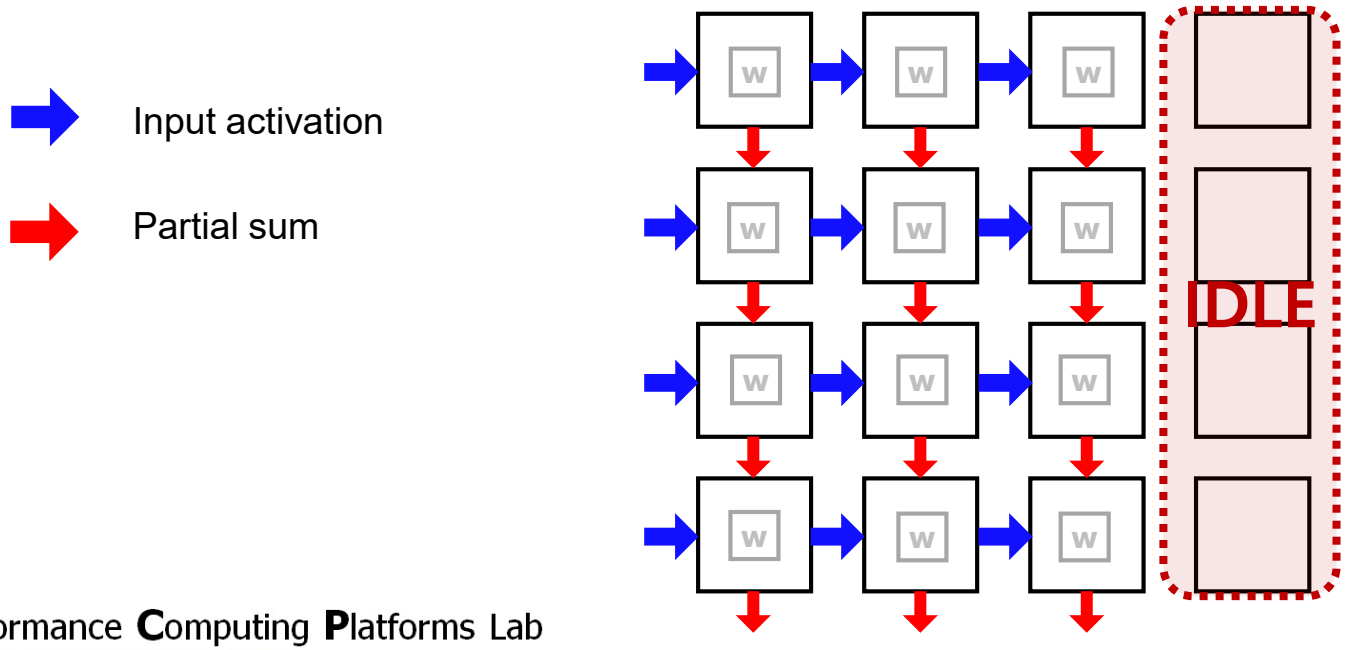
- Fine-grained systolic array allocation granularity
  - **Input activation mirroring** for PE column distribution
  - **Partial sum mirroring** for PE row distribution

- Low hardware implementation costs
  - Only **7.29%** overhead for the 128x128 Google TPU

- Support **up to 4 NNs**
  - Reverse the dataflows of the input activations and partial sums at the same time

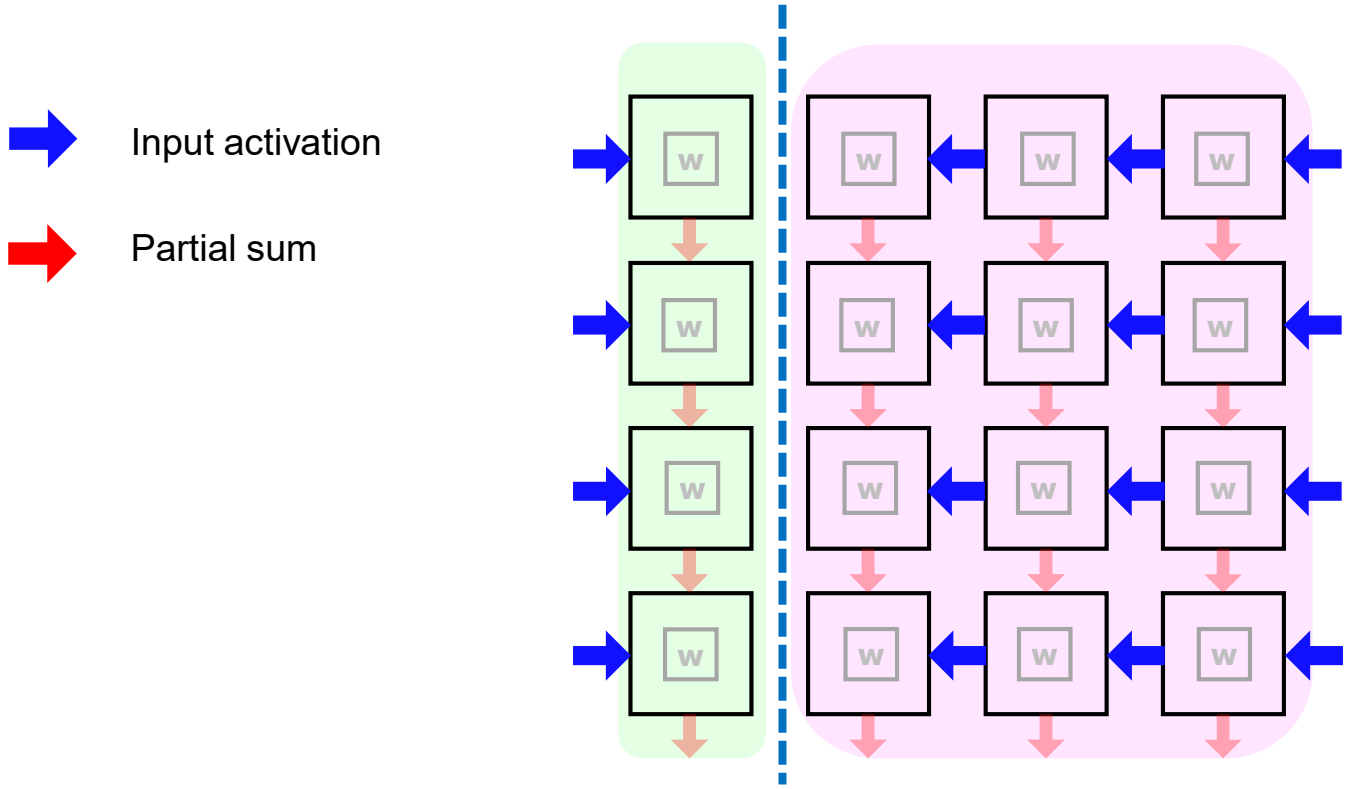
Weight-Stationary Dataflow

- Each filter weight remains stationary at one PE.
- Stream iacts **left-to-right** and psums **top-to-bottom**
  - Each PE row processes one input patch.
  - Each PE column processes one output channel.



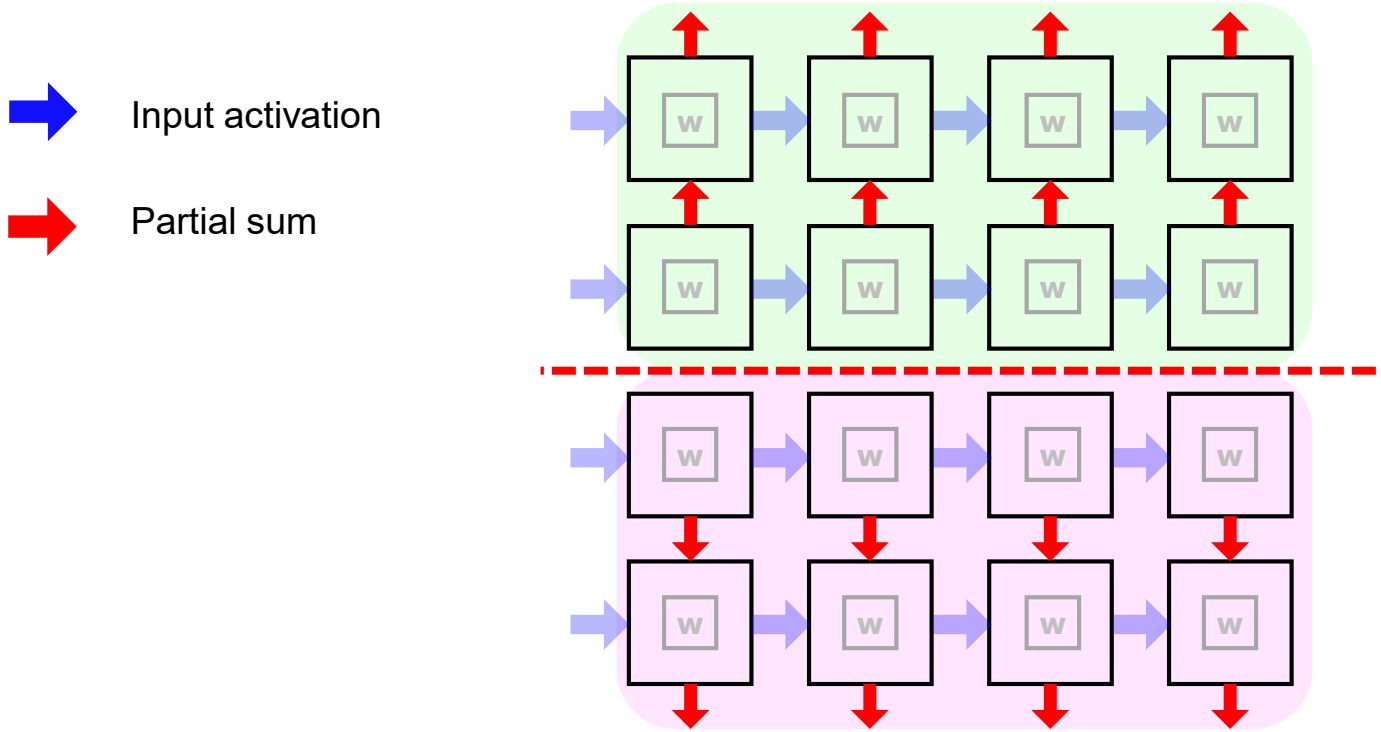
PE Column Distribution

- **Reverse the iact dataflows** of co-located NNs
  - One NN's iacts flow left-to-right and the other's flow right-to-left.
  - Both NNs' psums flow top-to-bottom.



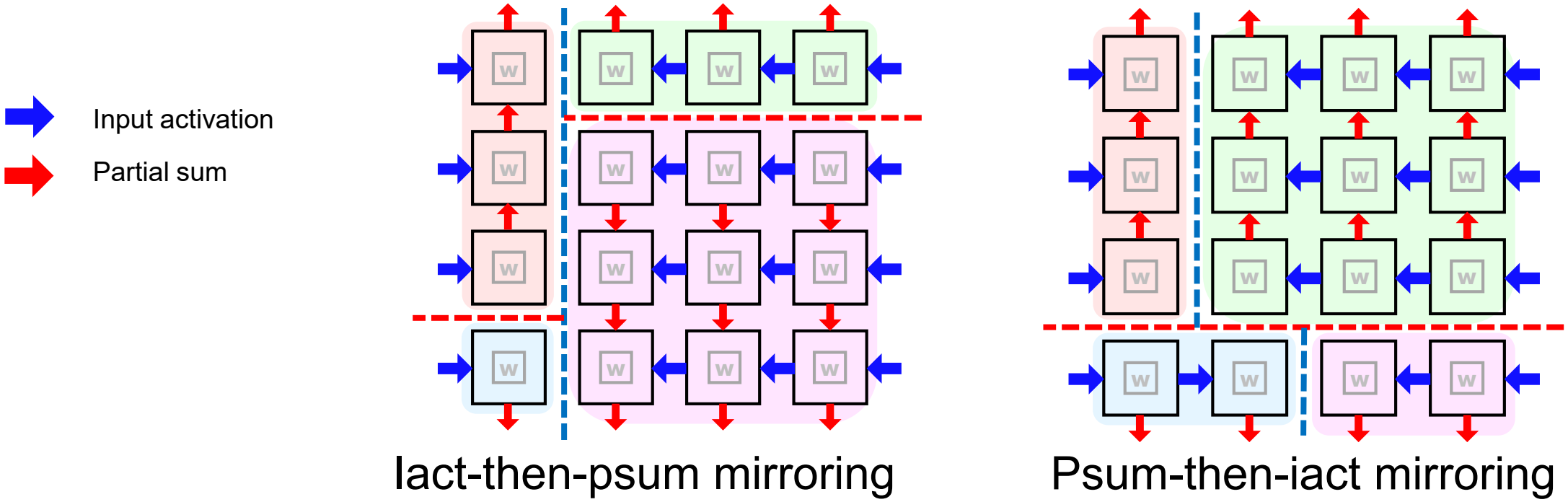
PE Row Distribution

- **Reverse the psum dataflows** of co-located NNs
  - Both NNs' iacts flow left-to-right.
  - One NN's psums flow upwards and the other's flow downwards.



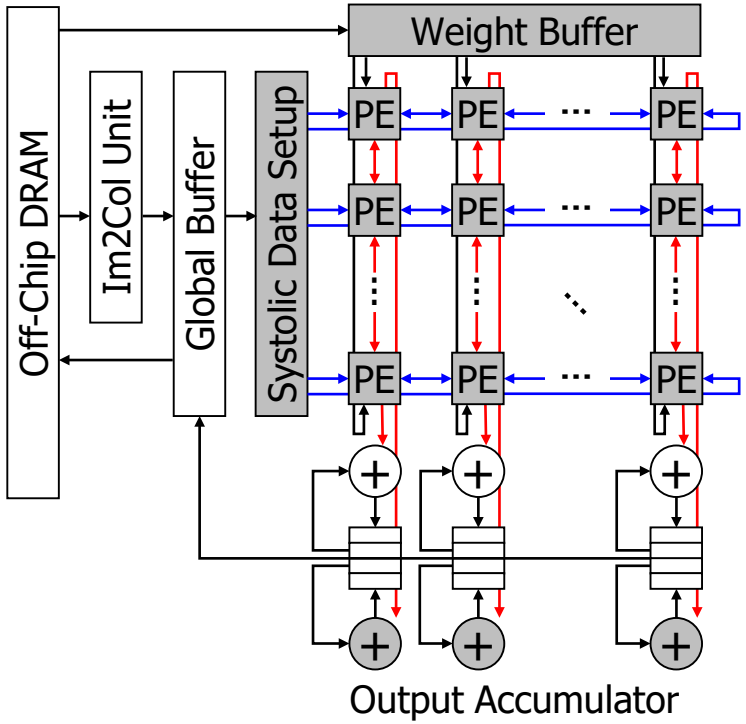
Dataflow Mirroring

- **Up to 4 NNs** by enabling both iact and psum mirroring
- Fine-grained allocation of both PE rows and columns

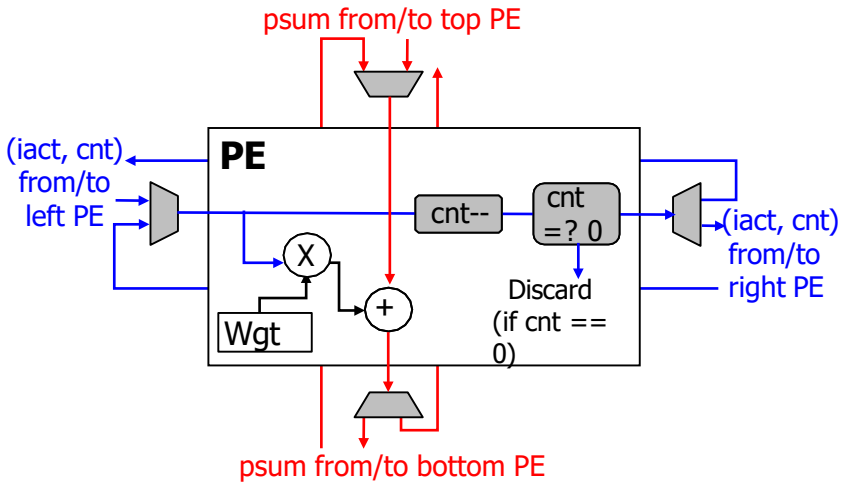


DM-NPU: NPU Architecture

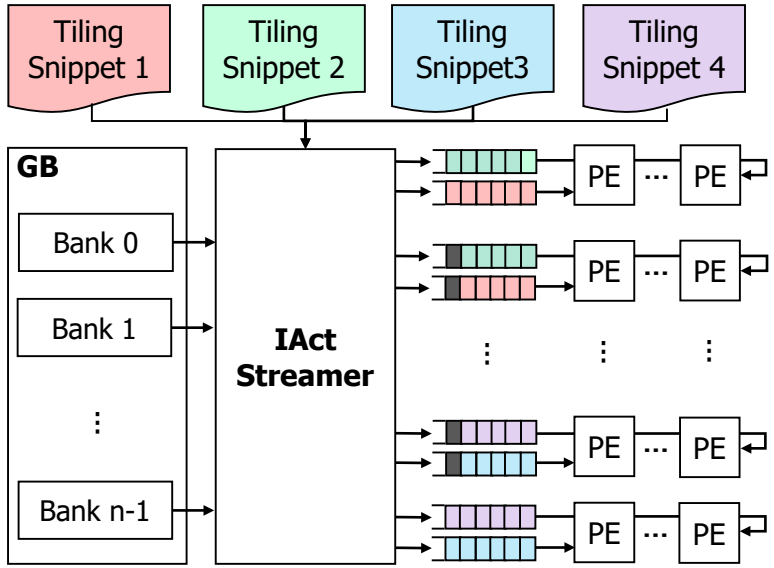
- Extend the baseline Google TPU with dataflow mirroring
  - Bidirectional bus, additional accumulators, extended SDS
  - Only **7.29%** overhead on top of 7-nm 128x128 TPU



(a) DM-NPU overall Architecture



(b) PE Architecture



(c) Systolic Data Setup Unit

DM-Perf: Performance Model

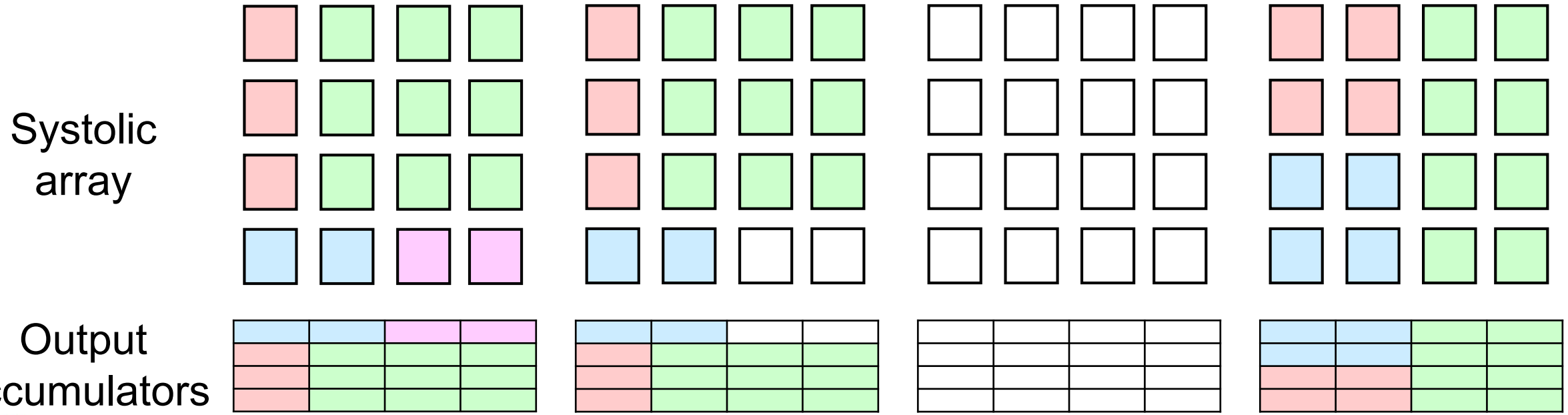
- PE Contention
  - Calculate computation latency using tile's size
- Off-chip DRAM BW contention
  - Consider DRAM access characteristics and the contention on DRAM BW
  - We use the profiled DRAM utilization as the layer's DRAM utilization
- On-chip GB contention
  - Define three cases following GB capacity
    - Non-prefetch, data reuse, and prefetch
  - Differently calculate execution latency following the case

Algorithm 1 DM-Perf

GB: Allocated GB capacity
BW_{DRAM}: off-chip DRAM Bandwidth
$Util_{DRAM}$: Utilization of DRAM of layer
M, K, N: Im2col data size of layer
m, k, n: Tiled im2col data of tile
$C_{latency}$: Compute latency of tile
$M_{latency}$: Memory fetch latency of tile
$end_{iact}, start_{iact}$: End, start address of input activation of tile
$Time_{estimated}$: Total execution latency of a DNN
procedure DM-PERF.ESTIMATEEXECTIME
    $Time_{estimated} = 0$
    for Tiles in Layers **do**
        $GB_{Leftover} = GB$
        $Data_{Layer} = M \times K + K \times N$
        for each (m,k,n) in Tiles **do**
            $C_{latency} = m + 2 \times k + n$
            $iact_{size} = end_{iact} - start_{iact}$
            $M_{latency} = (iact_{size} + k \times m)/(BW_{DRAM} \times Util_{DRAM})$
            if $GB_{Leftover} \leq 0$ **then**
                $Time_{estimated} += (C_{latency} + M_{latency})$
            else if $(GB_{Leftover} > 0)$ and $(Data_{Layer} < GB)$ **then**
                $Time_{estimated} += C_{latency}$
            else
                $Time_{estimated} += \max(C_{latency}, M_{latency})$
                $GB_{Leftover} -= (iact_{size} + k \times m)$
            end if
        end for
    end for
return $Time_{estimated}$
end procedure

DM-Scheduler: Scheduler

- Support **dynamic re-allocation** of the systolic array
  - DRAIN the executing layers when an NN arrives or finishes
  - Re-allocate the systolic array after the preemption completes



Simulation Configuration

- SCALE-Sim/DRAMsim3 + Accelergy/CACTI
  - SCALE-Sim, CACTI and Accelergy for PEs and on-chip SRAMs
  - DRAMsim3 for off-chip DRAM

Parameter	Low Performance	TPU-Like	High Performance
Clock frequency	1 GHz		
Systolic Array	64x64	128x128	256x256
Output accumulators	2048 entries / column		
On-chip SRAM buffer	32 banks, 32 B/cycle		
	4 MB	8 MB	16 MB
Off-chip DRAM	HBM2, 8 channels, 256GB/s		
Computation order	Filter-major		
Memory scheme	Working sets of filter and activations		

Spatial Multitasking Workloads

- Nine representative MLPerf DNNs with batch sizes 4
- 2-, and 4-way multitasking workloads

Name	PE Util.	DRAM Bandwidth Util.	Category
AlexNet	7.70%	26.94%	M
GoogleNet	35.78%	41.15%	X
ResNet-50	39.37%	48.41%	X
AlphaGoZero	63.53%	21.57%	C
NCF	0.21%	8.19%	L
FasterRCNN	44.71%	47.52%	M
SeqCNN	5.06%	47.21%	M
SeqLSTM	1.00%	32.30%	M
Transformer	0.79%	22.07%	M

: memory-intensive / C: compute-intensive / X: mixed / L: lightweight

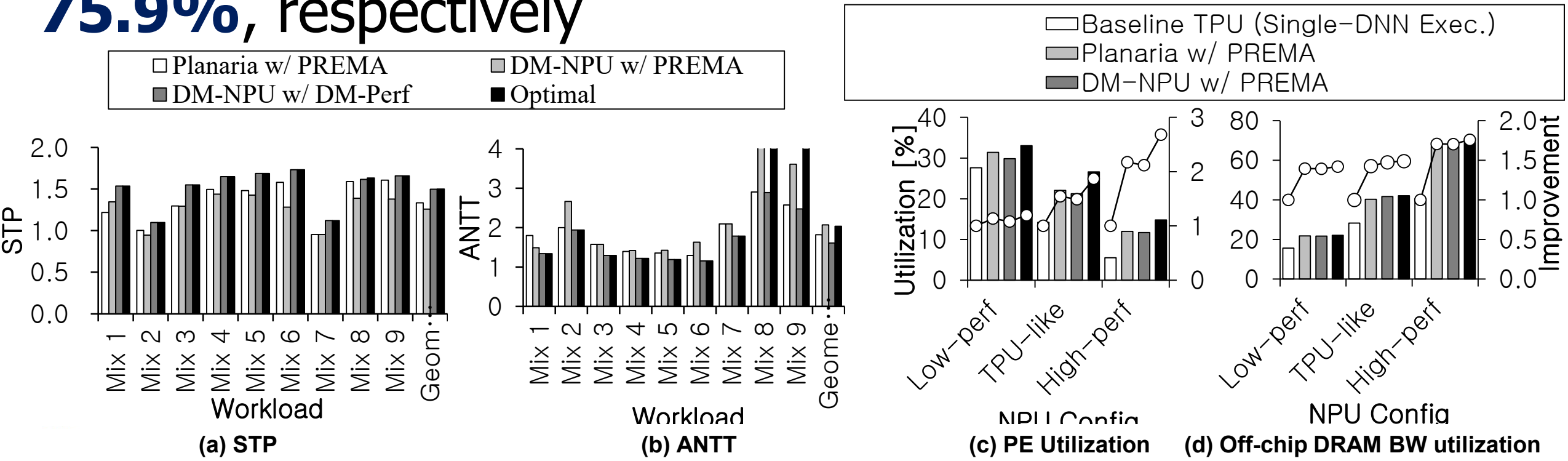
(a) Characteristics of the evaluated MLPerf DNNs

Workload	Benchmarks	Scenario
Mix 1	AlphaGoZero, NCF	CL
Mix 2	AlphaGoZero, SeqCNN	CM
Mix 3	NCF, FasterRCNN	LX
Mix 4	NCF, SeqLSTM	LM
Mix 5	NCF, Transformer	LM
Mix 6	NCF, AlexNet	LM
Mix 7	FasterRCNN, ResNet-50	XX
Mix 8	AlphaGoZero, ResNet-50, NCF, Transformer	CXLM
Mix 9	GoogleNet, ResNet-50, NCF, Transformer	XXLM

(b) Evaluated spatial-multitasking workloads

Evaluation - STP, ANTT, and HW resource util.

- STP improves **up to 31.9%** over Planaria w/ PREMA
- Geometric mean improvement in ANTT **achieves 13.0%**
- PE & DRAM BW utilization improve up to **2.68x** and **75.9%**, respectively

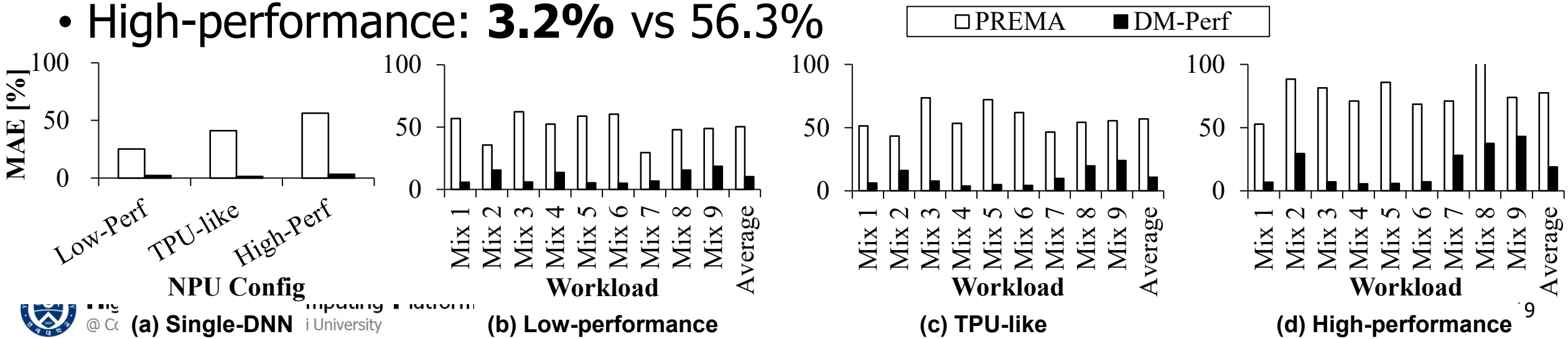


(a) STP (b) ANTT (c) PE Utilization (d) Off-chip DRAM BW utilization

Evaluation result of TPU-like DM-NPU

Evaluation - Performance Modeling

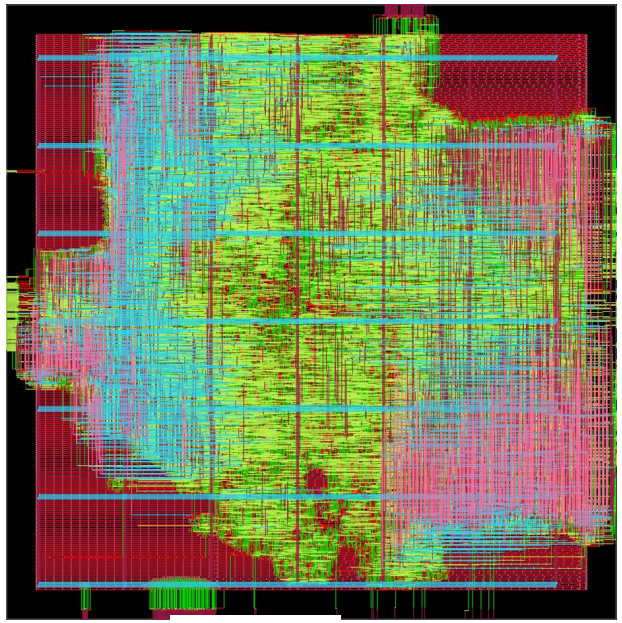
- Use a mean absolute error (MAE) as the accuracy metric
  - $MAE = \frac{1}{n}\sum_{i=0}^{n-1}|latency_i - predicted_i|$
- DM-Perf achieves high accuracy over PREMA
  - Single-DNN: **2.91%** vs 44.36%
  - Low-performance: **2.3%** vs 25.2%
  - TPU-like: **1.4%** vs 41.1%
  - High-performance: **3.2%** vs 56.3%



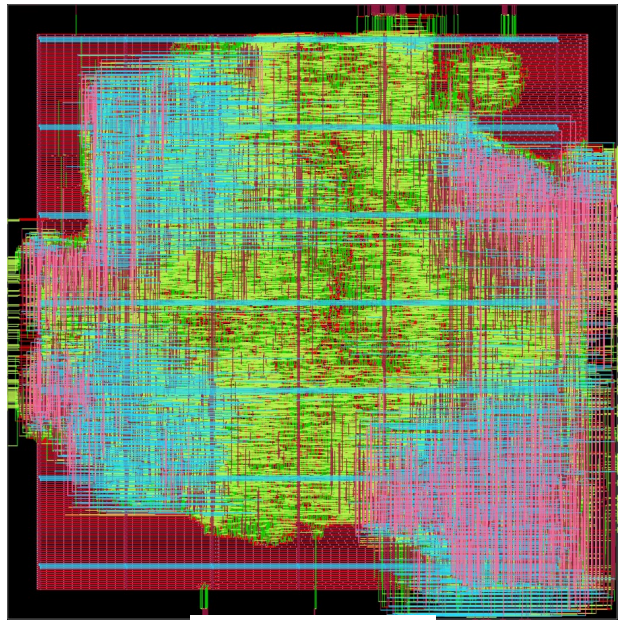
(a) Single-DNN (b) Low-performance (c) TPU-like (d) High-performance

Evaluation - HW Implementation Costs

Component	TPU	DM-NPU	Overhead
PE	0.1697 mm^2	0.2024 mm^2	+0.0327 mm^2 (+19.27%)
SDS+WB	0.0543 mm^2	0.0606 mm^2	+0.0063 mm^2 (+11.66%)
ACCQ	0.0320 mm^2	0.0357 mm^2	+0.0037 mm^2 (+11.44%)
Others	0.3294 mm^2	0.3294 mm^2	-
Total	0.5854 mm^2	0.6280 mm^2	+0.0427 mm^2 (+7.29%)



(a) TPU



(b) DM-NPU

Post-PnR layouts of low-performance TPU and DM-NPU implementations produced by OpenROAD Flow with the 7-nm ASAP7 PDK

Conclusion

- **Difficult** to fully utilize the systolic array
  - The existing coarse-grained systolic array allocation limits the potential of spatial multitasking on NPUs.
- **Dataflow mirroring & DM-NPU**
  - Reverse the dataflows of co-located NNs
  - Achieve highly flexible and efficient spatial multitasking
- **DM-Perf: Accurate Contention-aware perf. model**
  - Capture the shared NPU HW resource contentions using per-layer profiles
  - Achieve high accurate latency calculation
- Up to **31.9%** performance improvement over SotA
  - Optimal systolic-array allocation with fine-grained PE distribution