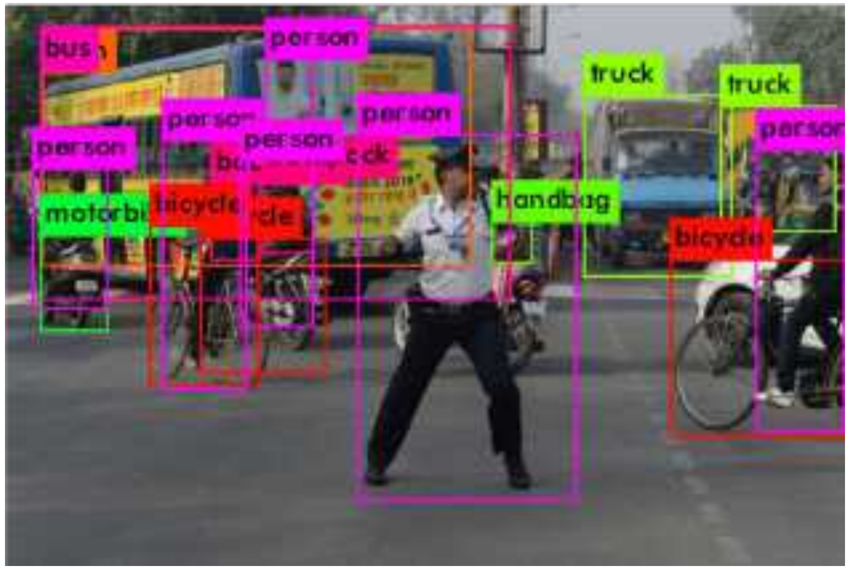
From TOPS to Throughput:
Getting the most throughput from the least hardware

Dr. Cheng C. Wang
Co-Founder & Senior VP Architecture/Software/Engineering
Flex Logix Technologies, Inc.

cheng@flex-logix.com

AI Hardware Summit
September 17-18, 2019, Mt. View, CA

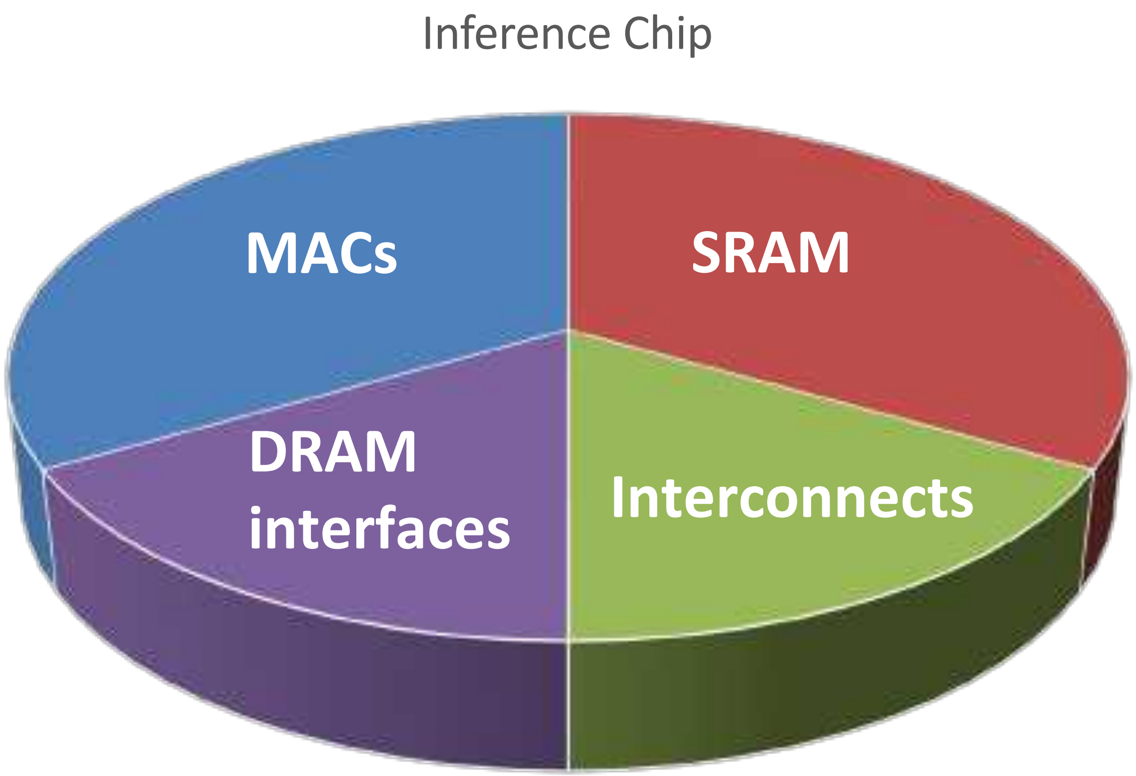


Customer Wish List for an Edge Inference Chip

Target neural network applications	Typically object detection (e.g. YOLOv3, SSD, **not** ResNet50)
Batch = 1	Lowest latency
Preferred resolution	Typically 1-4 Megapixels (**not** 224x224)
High prediction accuracy	No modifications to the model (**no** forced sparsity)
Targeted performance	Highest inferences / sec (**not** highest TOPS)
Within power and cost budget	No fans, low cost, highest inferences / W (**not** highest TOPS/W)
Major supported frameworks	No custom frameworks that requires porting

Efficiency is key: highest **inferences / $** and **inferences / W** is what customers look for

What contributes to inference efficiency?



Chip cost & power are dominated by:

- Compute (MACs)
- Local weight/activation (SRAM)
- Non-local weight/activation (DRAM)
- Data movement between them all (Interconnect)

DRAM chip cost & power are **not** included above

Only MAC & MAC utilization (%) contribute to inference performance.
Everything else is overhead

How many MACs do we need? And how do we run them?

Short answer:

- >100x more than what people are running today

Challenge:

- Run the 100x models with much lower power & cost

How?

- Reduce memory access & data movement, especially to/from DRAM

Highest Accuracy

>100 GOPs per frame

>100x

Lowest Accuracy

<1 GOP / frame

5-10 GOPs per frame

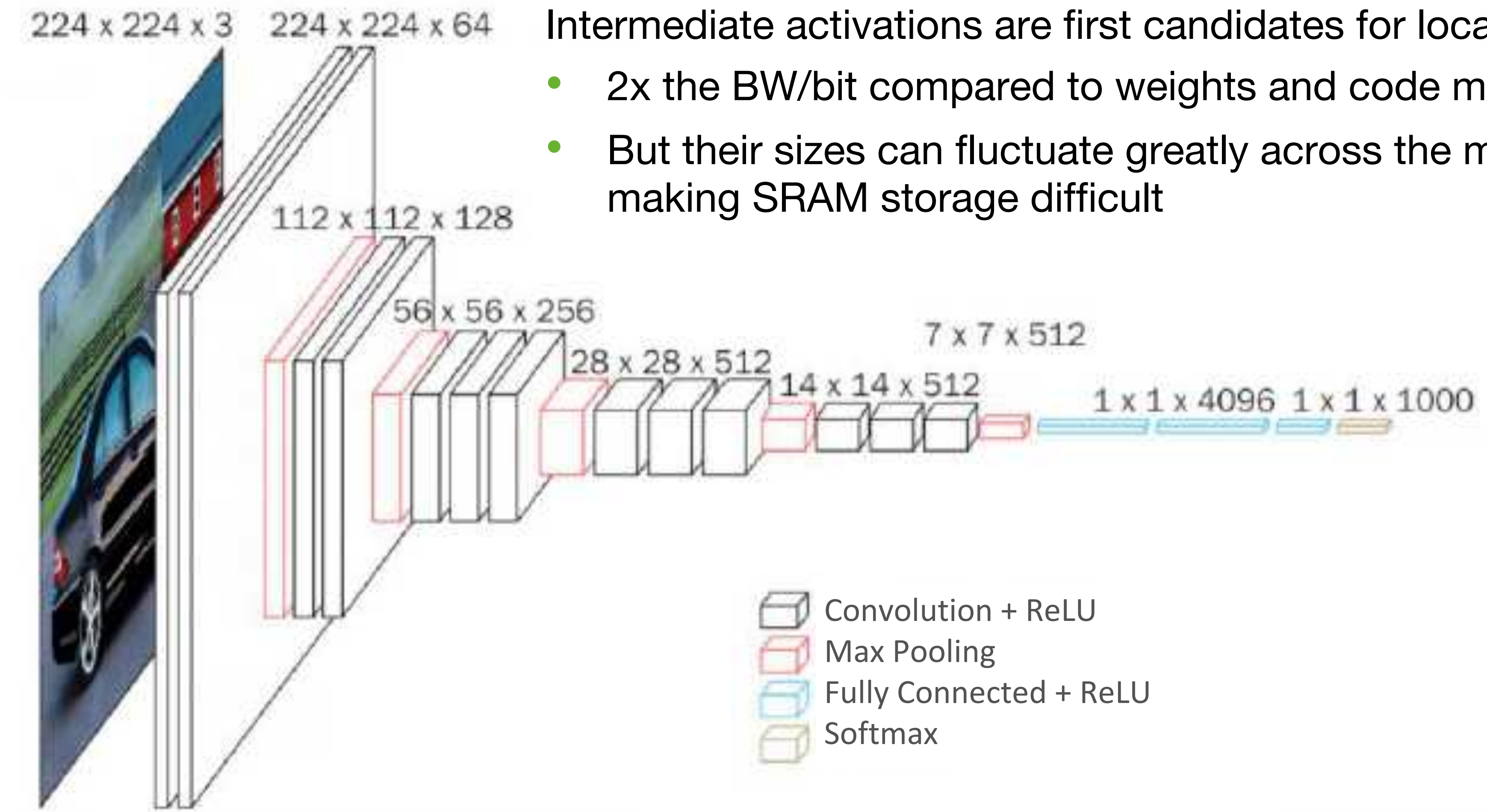
MobileNetV2 SSD 224x224	TinyYOLOv2 416x416	YOLOv3 1920x1080
<1 GOP / frame	5-10 GOPs per frame	>100 GOPs per frame

What needs to be stored in Neural Network Inference?

1. The input image
2. The weights
3. The intermediate activations
4. The code that controls the inference processor

Storage	On-chip SRAM	Off-chip DRAM
Power	Lower power	Higher power
Cost	Higher cost/bit	Lower cost/bit
Capacity	Limited capacity Not expandable	Higher capacity Expandable
Application	Intermediate activations Small Weights Small Processor code	Input image Large Weights Large Processor Code

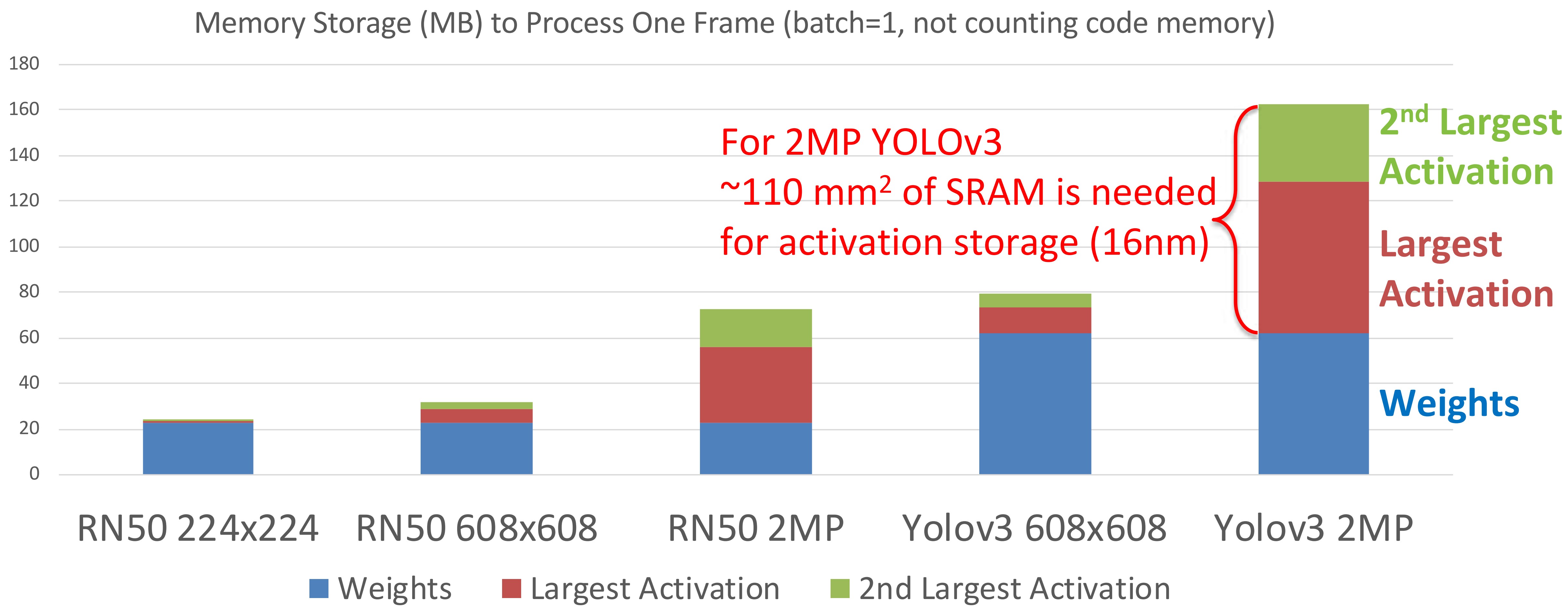
Activation Output Size Varies by Layer



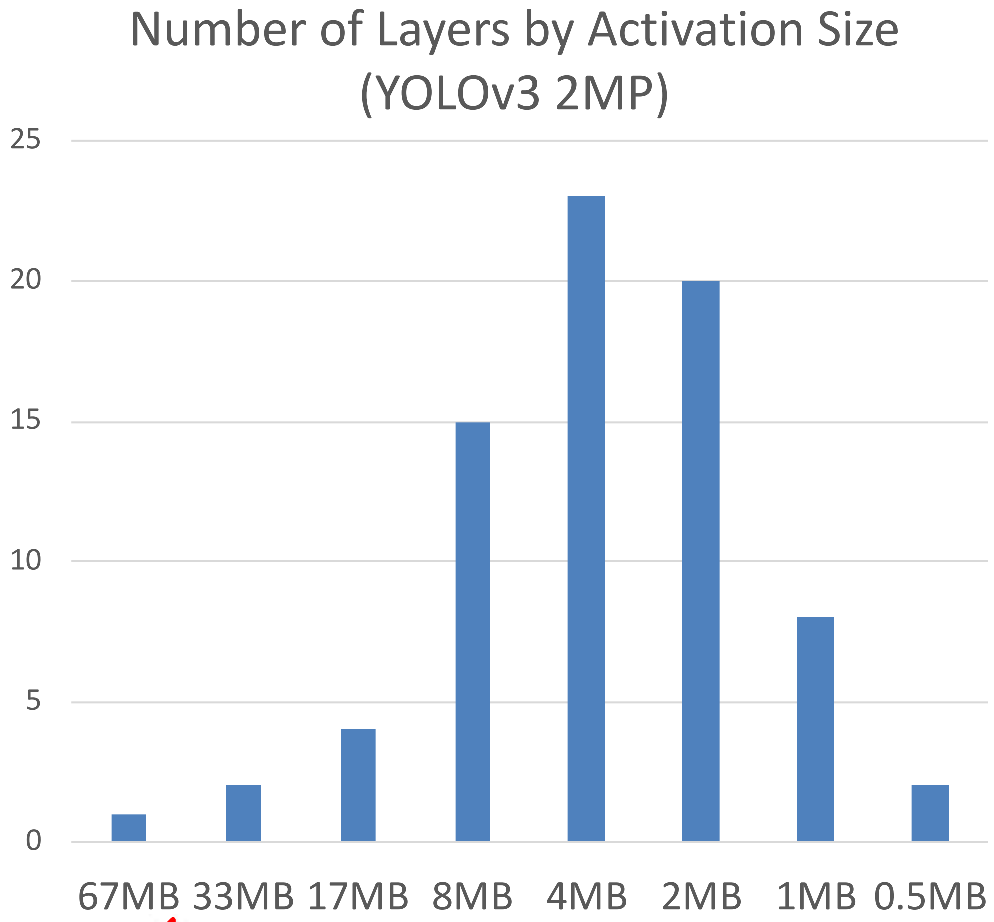
Intermediate activations are first candidates for local storage

- 2x the BW/bit compared to weights and code memory
- But their sizes can fluctuate greatly across the model, making SRAM storage difficult

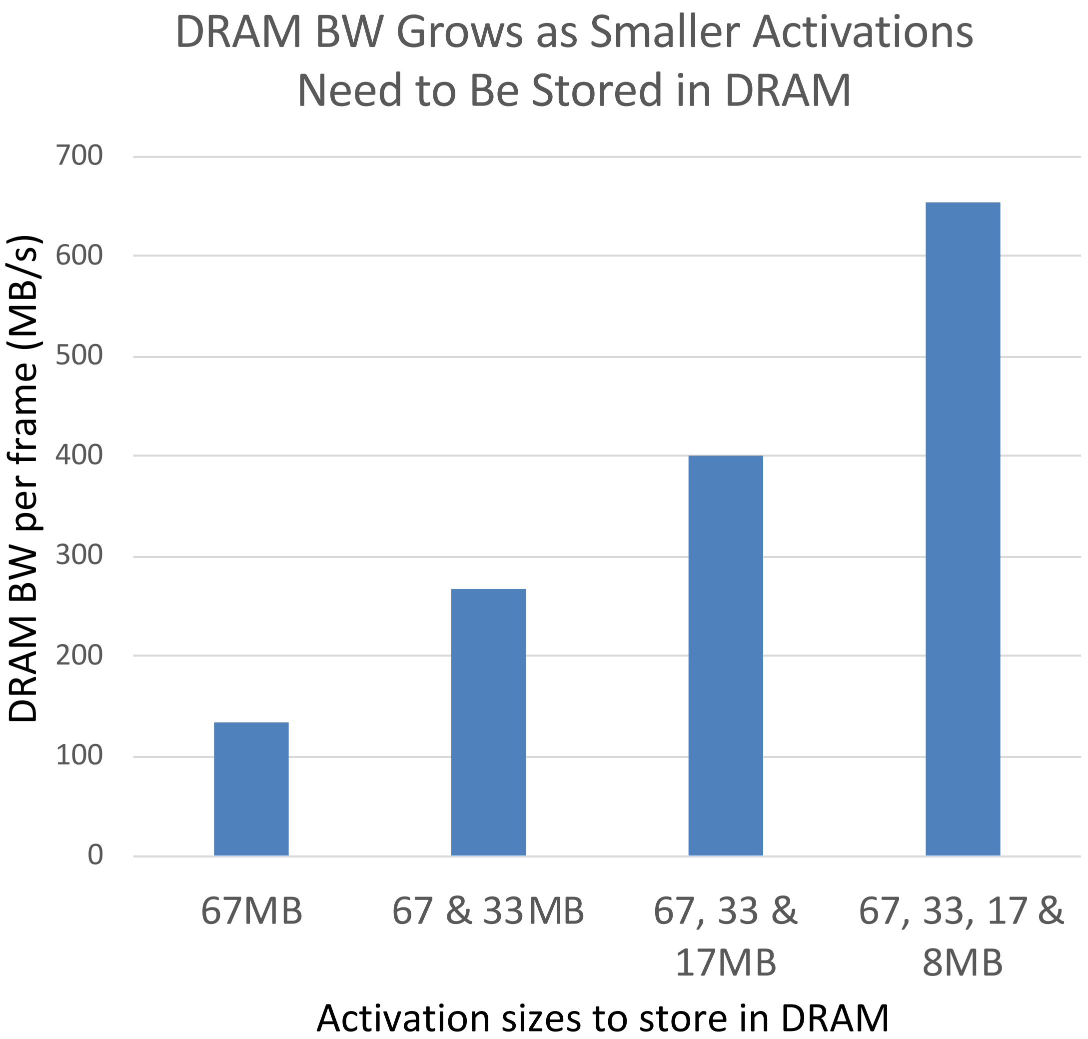
Activation Storage Size >> Weights for Megapixel images



Balancing SRAM capacity vs. DRAM BW

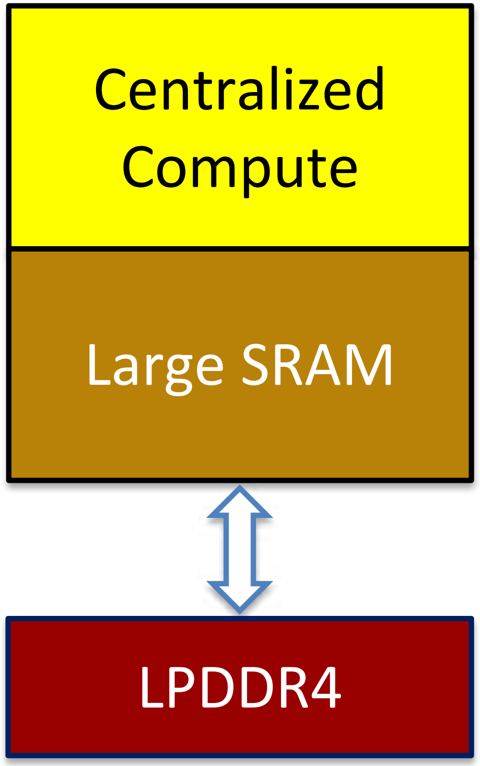


More SRAM reduces DRAM BW



Memory Architecture: from Centralized to Distributed

- ✓ SRAM reduces >10x energy/bit over DRAM
- ✓ Distributed, local RAM with each compute reduces energy/bit by another 10x
- ✗ But **interconnect becomes the new problem** (power, delay & SW programming complexity)



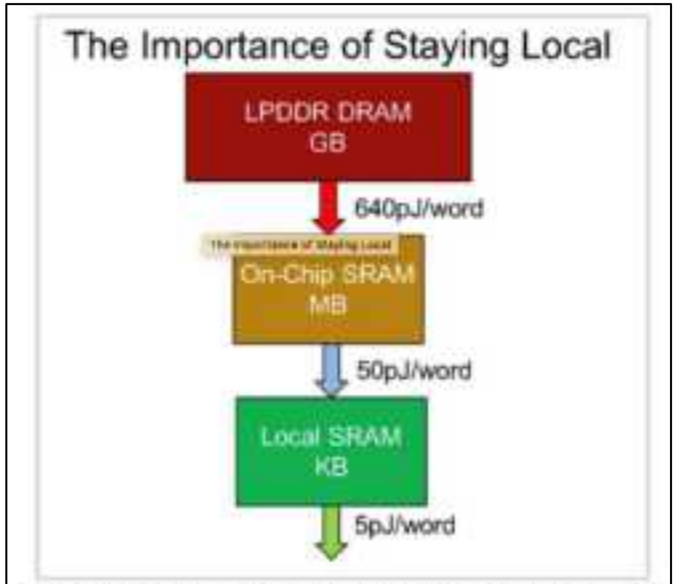
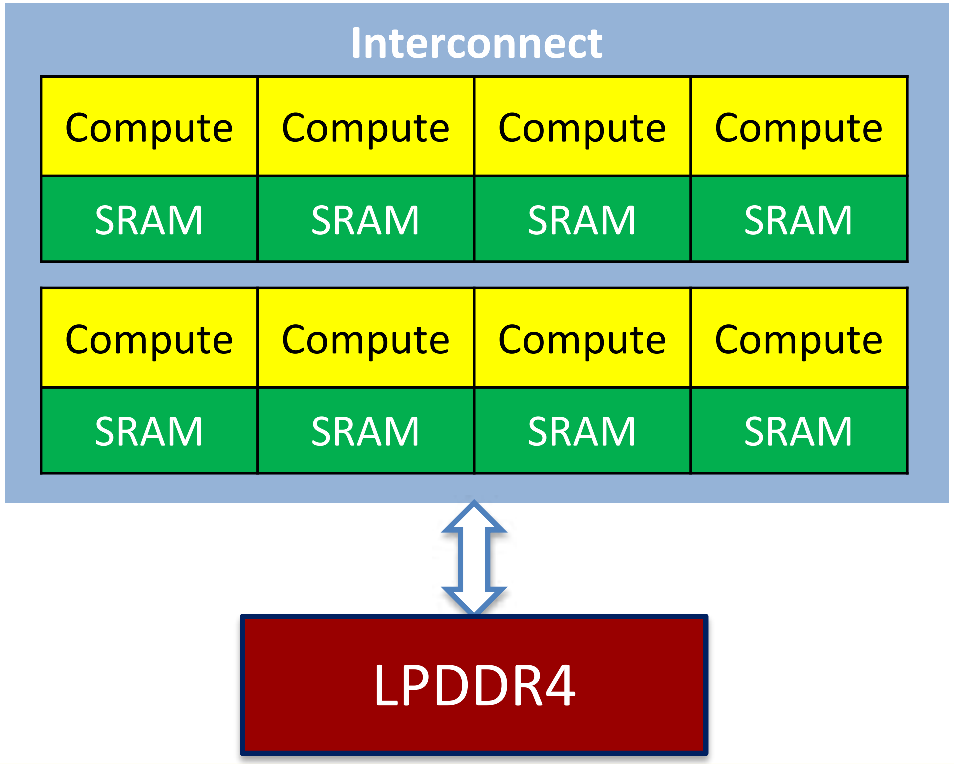


Diagram of Data Fetch Energy vs Distance (from Horowitz)

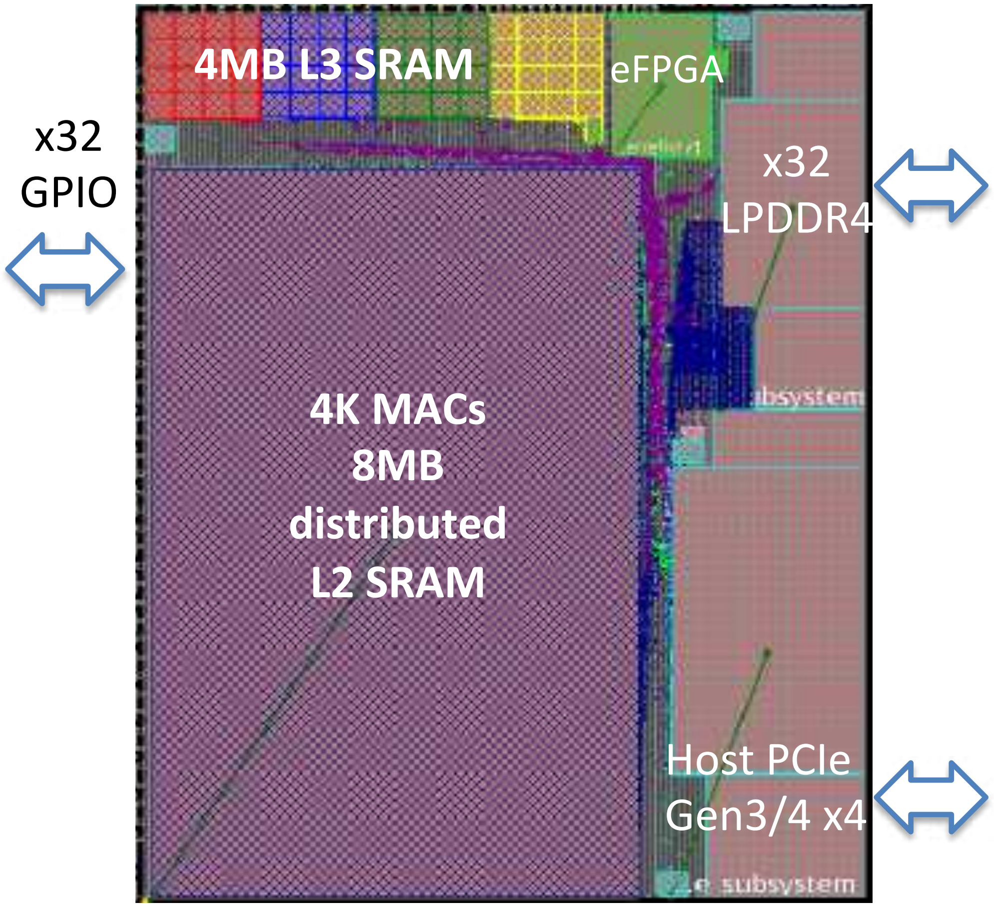


Keys to Efficient Inference Throughput

- Maximize MAC utilization

- Minimize everything else
  - Use smaller, distributed SRAM for compute
  - Use efficient, high bandwidth interconnects
  - Minimize off-chip DRAM access whenever possible
    - But keep 1 DRAM to allow for model growth

InferX X1 Key Specs, Die Plot



- 50mm^2 TSMC 16FFC
- 21x21mm FCBGA
- 1.067GHz Operation

- 4K MACs @ INT8x8/16x8
  or 2K MACs @ INT16x16/BF16
- Winograd acceleration for INT8

- 8MB L2 SRAM + 4MB L3 SRAM
- x32 LPDDR4 (16GB/s peak BW)

- Partners: TSMC, GUC, Synopsys, Arteris, Analog Bits, Cadence, Mentor

- Available as Chip & PCIe Board

ResNet-50 throughput comparison

	TOPS (INT8)	Number of DRAM	ResNet-50 (batch=1) Inferences / s
Nvidia Tesla T4	130	8	961
Nvidia Xavier AGX	32	8	480
InferX X1	8.5	1	293
Google Edge TPU	4	1?	21 (batch=?)

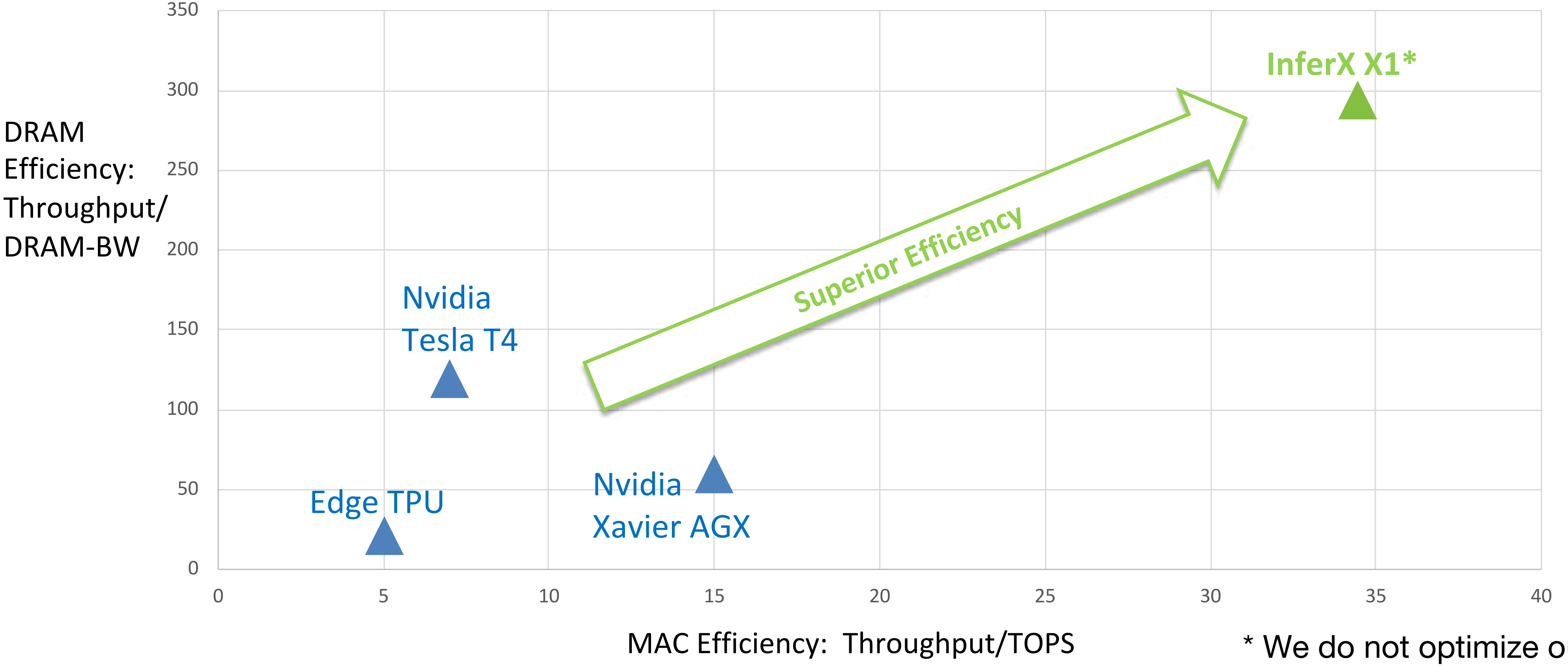
Low correlation between TOPS, DRAM & throughput!

But, high correlation between TOPS, SRAM, DRAM & Cost:

- More TOPS = more silicon area = cost
- More SRAM = more silicon area = cost
- DRAM = silicon area (PHY), package & BOM cost

Efficiency is Throughput/$ - correlates with throughput/TOPS & throughput/DRAM

DRAM Efficiency & MAC Efficiency for ResNet-50, batch=1



* We do not optimize our performance for ResNet-50

2MP YOLOv3 Throughput Comparison

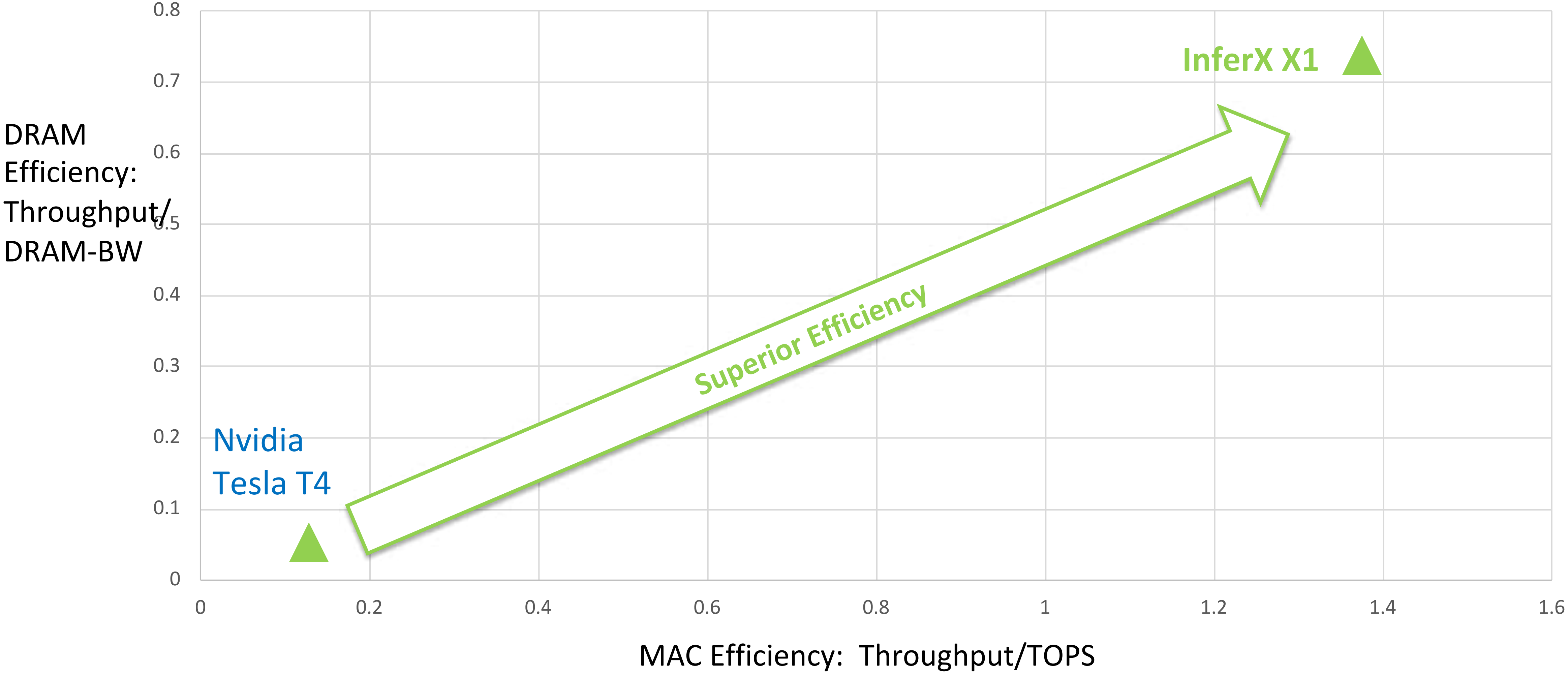
	TOPS (INT8)	Number of DRAM	YOLOv3 2Megapixel Inferences / s
Nvidia Tesla T4 *	130	8 (320 GB/s)	16
InferX X1	8.5	1 (16 GB/s)	12

X1 has 7% of the TOPS and 5% of the DRAM bandwidth of Tesla T4

Yet it has 75% of the inference performance running YOLOv3 @ 2MP

* through TensorRT framework

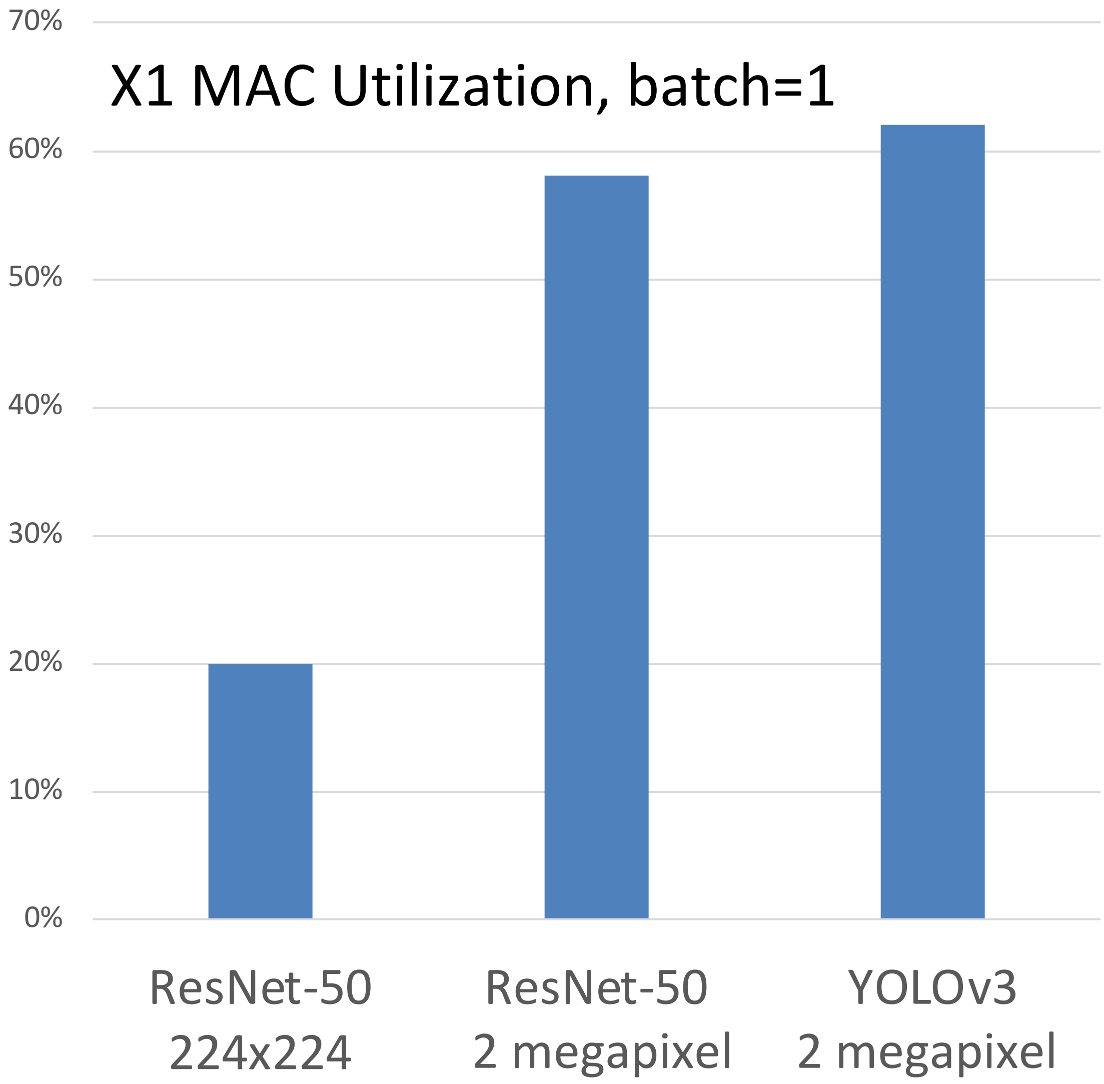
Throughput/TOPS & Throughput/DRAM for YOLOv3, 2Megapixel, batch=1



What Makes InferX X1 Efficient?

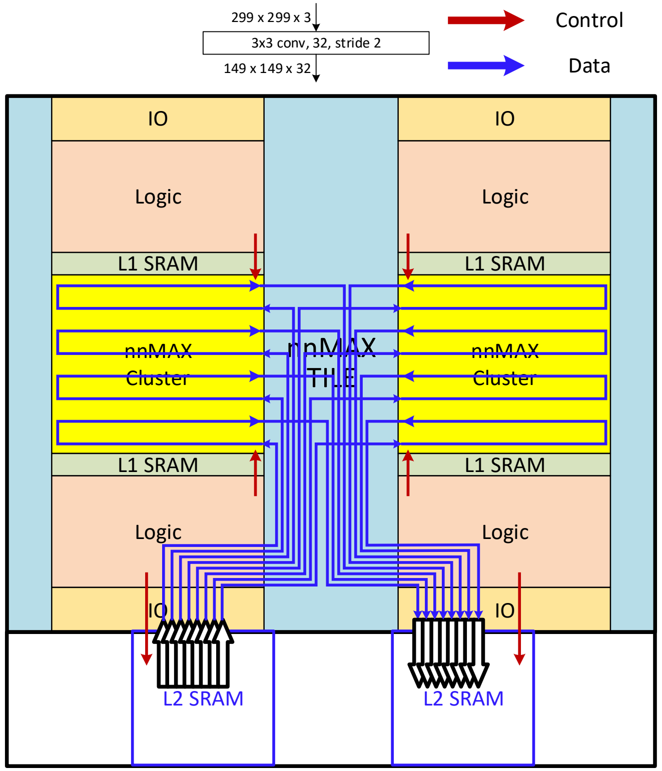
- InferX X1 is optimized for megapixel images & tough models

- How do we achieve high throughput/low cost?
  1. ASIC-like MAC efficiency:
     - ✓ High MAC utilization % = inference perf.
     - × Idle MACs = cost & power
  2. Programmable, efficient interconnect
  3. Reducing memory accesses via deep layer fusion
  4. “Hide” DRAM access time in background



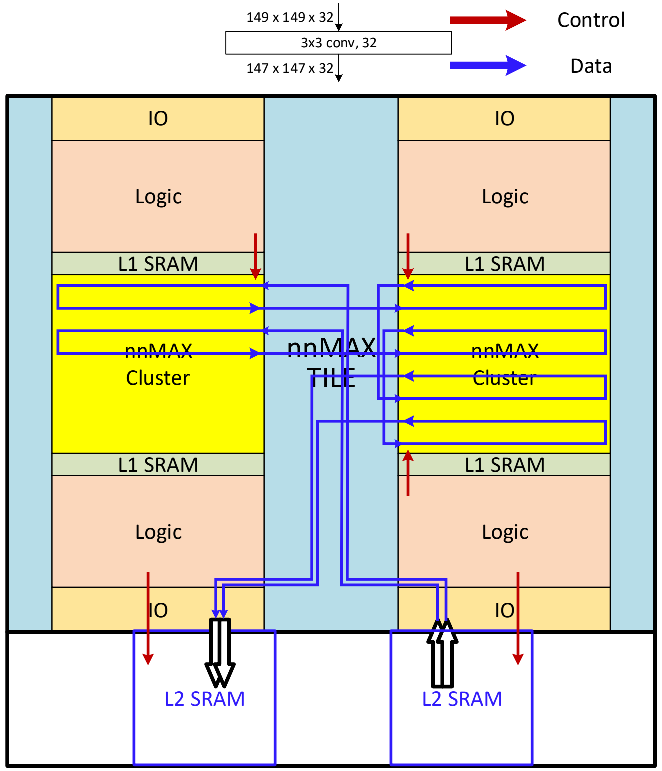
#1 & 2 Dedicated path: memory to compute to memory, programmed for each layer

Layer 0:
16 nnMAX in parallel
L2 SRAM → L2 SRAM



Reconfigure
in <2 μs

Layer 1:
4 nnMAX in parallel, 3 in series
L2 SRAM → L2 SRAM

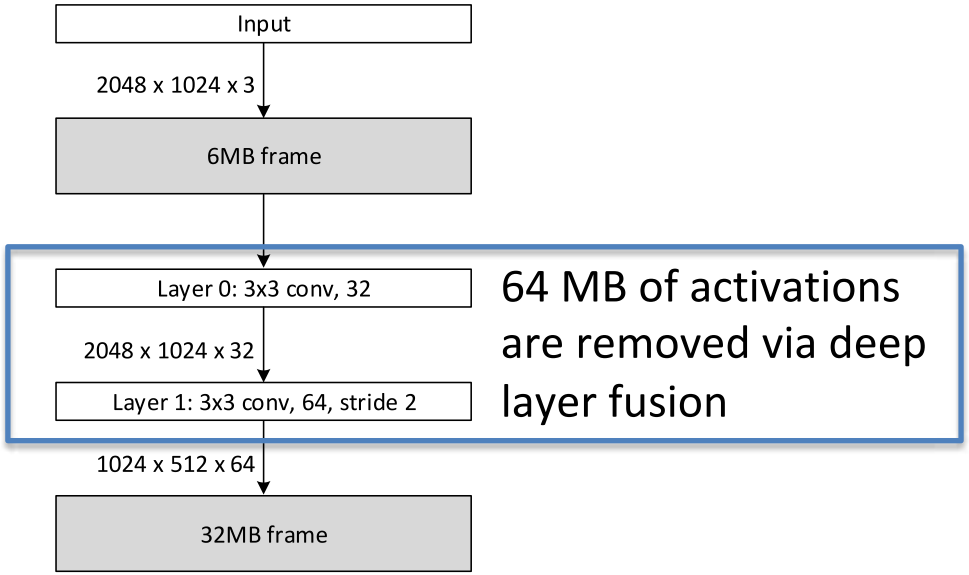


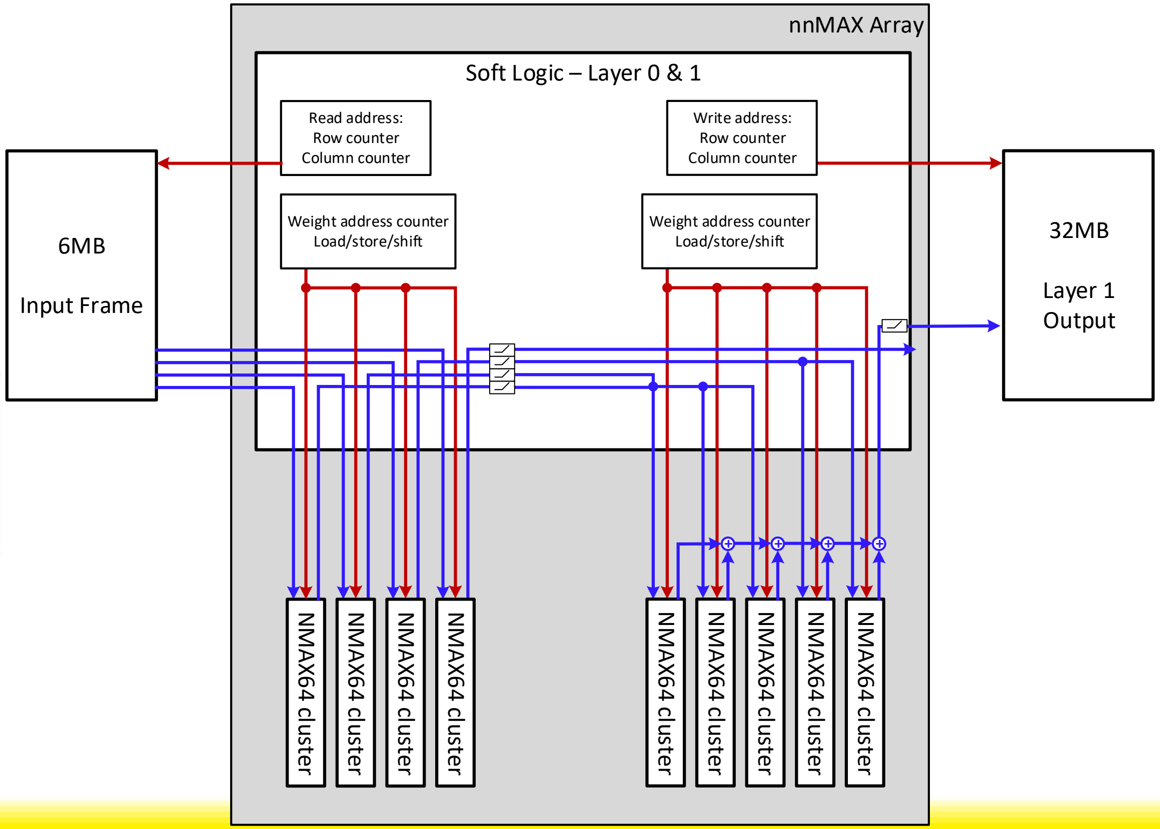
Localized data access & compute
ASIC-like performance yet fully reconfigurable

*architectural diagram, not to scale

#3 Deep layer fusion reduces memory requirement

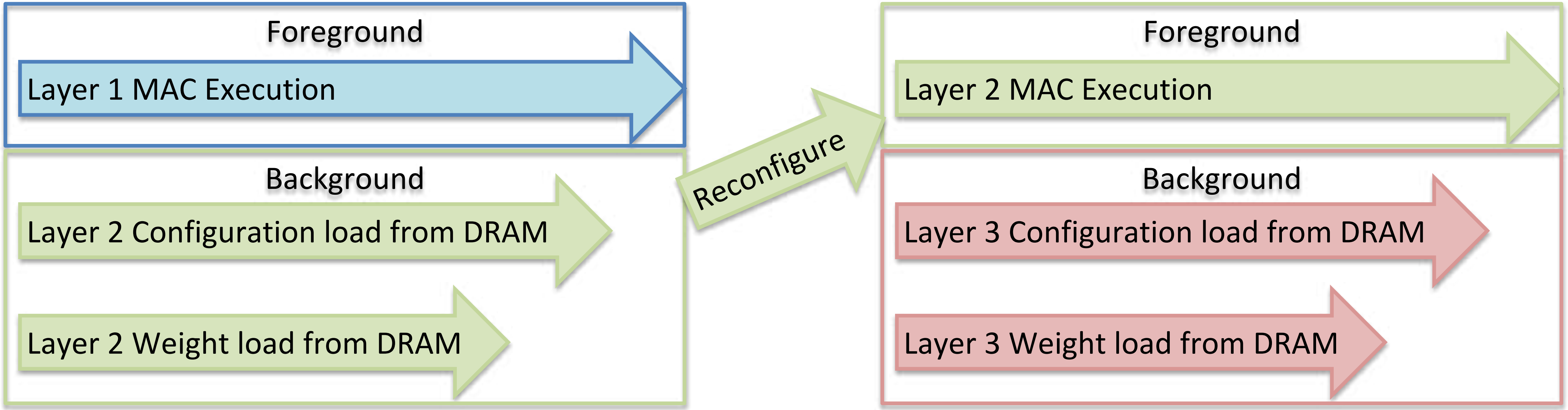
- Deep Layer Fusion combines multiple layers (not just activation layers) to eliminate reads/writes for some of the largest activations
  - In YOLOv3 2MP: DLF can reduce memory requirement by 2x





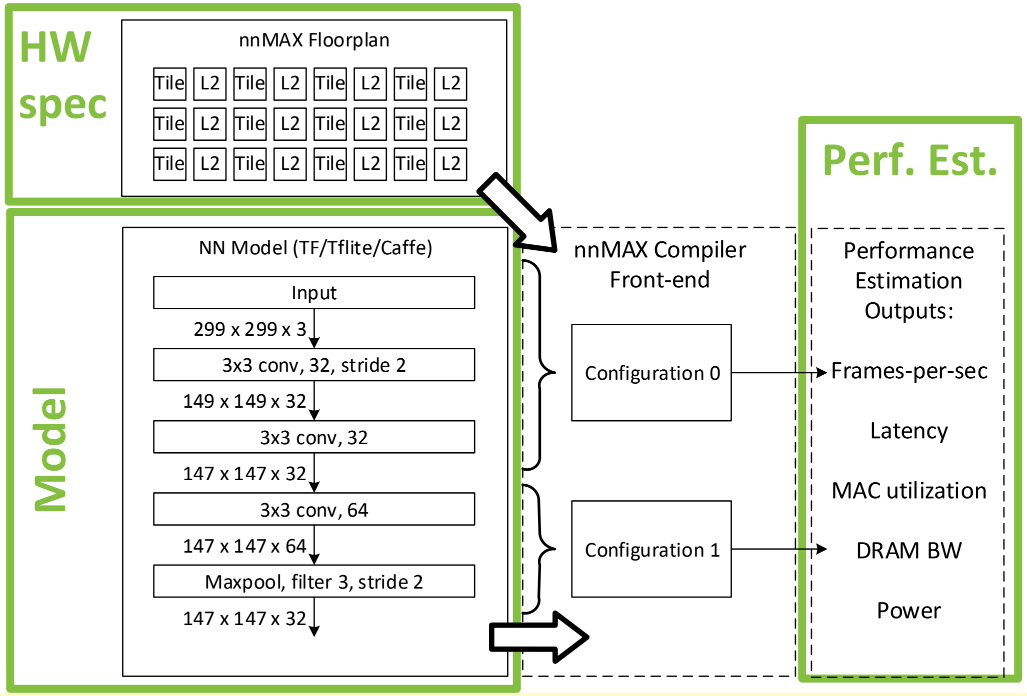
#4 “hiding” DRAM access in background

- Next layer’s weights and configuration are loaded in background while current layer runs
  - During reconfiguration, the background data is quickly moved to the front
- With a small amount of SRAM, performance is kept very high by minimizing DRAM stalls
  - Most of the time, DRAM access time is “hidden” behind layer execution time
  - For 2MP Yolov3, just 4% of cycles are DRAM overhead (stalls MACs)



InferX X1 Performance Estimation – Available Now; Demo @ Booth 28

- First part of the compiler is the performance estimation
- Accepts X1 floorplan and TF-lite/ONNX model as input
  - Automatically partitions model across multi-layer configurations
  - Computes performance, latency, MAC utilization, DRAM BW per layer and per model




```
[INFO] *******Group Layer Native Mapping *********
[INFO] **********Performance Estimation***********
[INFO] Frame/s @ 1GHz                           : 931.223
[INFO] DRAM Bandwidth (GB/s)                    : 6.79634
[INFO] Latency (ms)                             : 10.7386
[INFO] TMAC/s                                   : 2.46525
[INFO] MAC Utilization (%)                      : 60.1869
[INFO] Area (mm^2)                              : 24.8
[INFO] SRAM Bandwidth (GB/s)                    : 13.7782
[INFO] Actual Batch Size                        : 10
[INFO] Store Config in SRAM                     : false
[INFO] Store Weights in SRAM                    : false
[INFO] ****************************************
```

nnMAX Compiler tested on many popular models

imagenet_resnet_v1_50	nasnet_large
imagenet_resnet_v1_152	resnet_v2_50
imagenet_resnet_v2_101	resnet50_v1.5
imagenet_resnet_v2_152	resnet_v2_101_299
inception_v1_224	squeezenet
inception_v2_224	xeption
inception_v3_299	yolov2
inception_v4_299	yolov2_tiny
mobilenet_v1_224	yolov3
mobilenet_v2_224	yolov3_tiny
mobilenet_v1_COCO_SSD	deeplabv3_257