Processing Near/In Memory

ISCA Tutorial (2019)

Website: http://eyeriss.mit.edu/tutorial.html

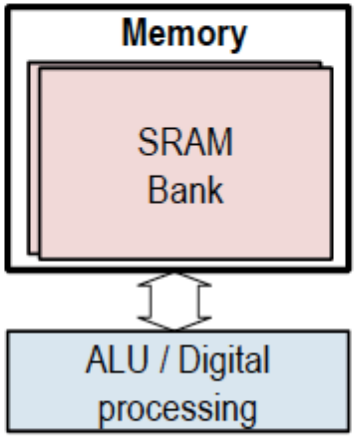
Joel Emer, Vivienne Sze, Yu-Hsin Chen

Processing Near vs. In Memory

Processing Near Memory

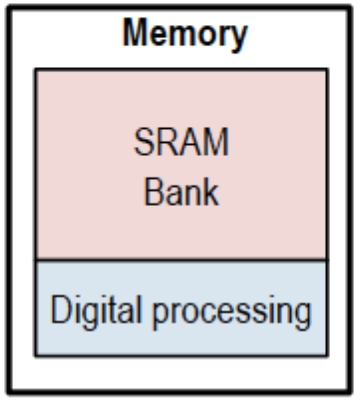
Processing In Memory

digital



- data access energy and latency dominating
- data reuse and data compression

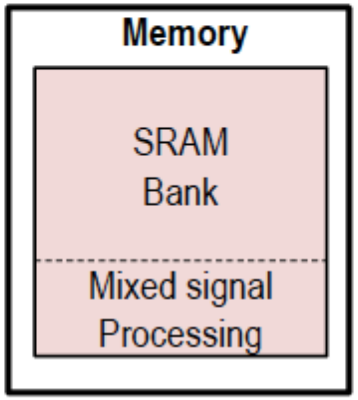
near memory



- computation still digital
- eliminates data transfer costs
- memory read energy dominates

Higher BW

deep in-memory



- memory access and computation combined
- mixed signal computation
- significant energy & latency reduction

Highest BW

Modified image from [Gonugondla, ISSCC 2018]

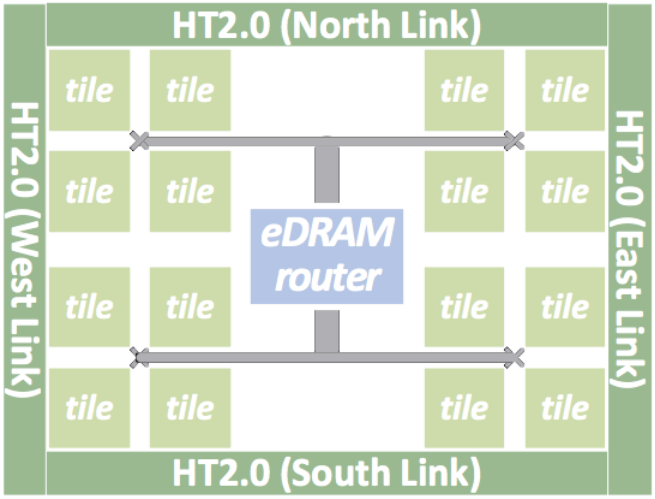
Processing Near Memory

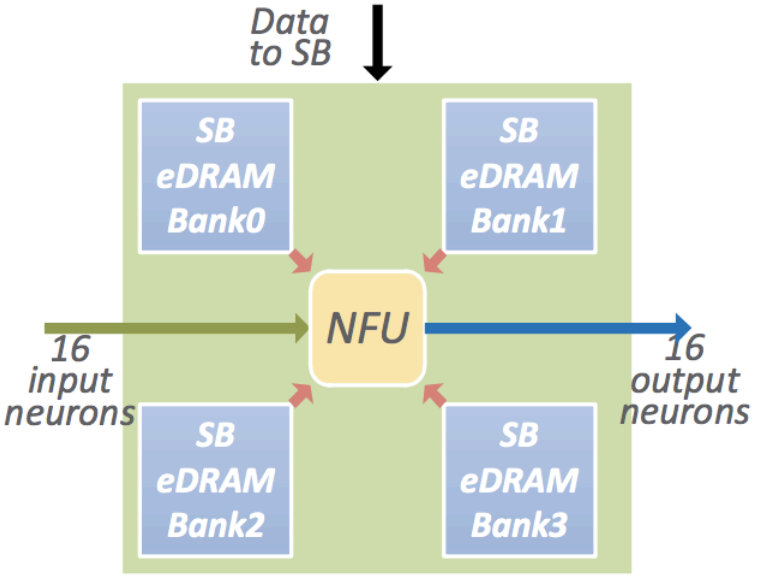
- Bring compute closer to the memory
- Benefits
  - Increase memory bandwidth
  - Reduce energy per access
- Challenges
  - Cost
- Embedded DRAM (eDRAM)
- 3D Stacked Memory (DRAM, SRAM)

eDRAM (DaDianNao)

- Advantages of eDRAM
  - 2.85x higher density than SRAM
  - 321x more energy-efficient than DRAM (DDR3)
- Store weights in eDRAM (36MB)
  - Target fully connected layers since dominated by weights

16 Parallel Tiles

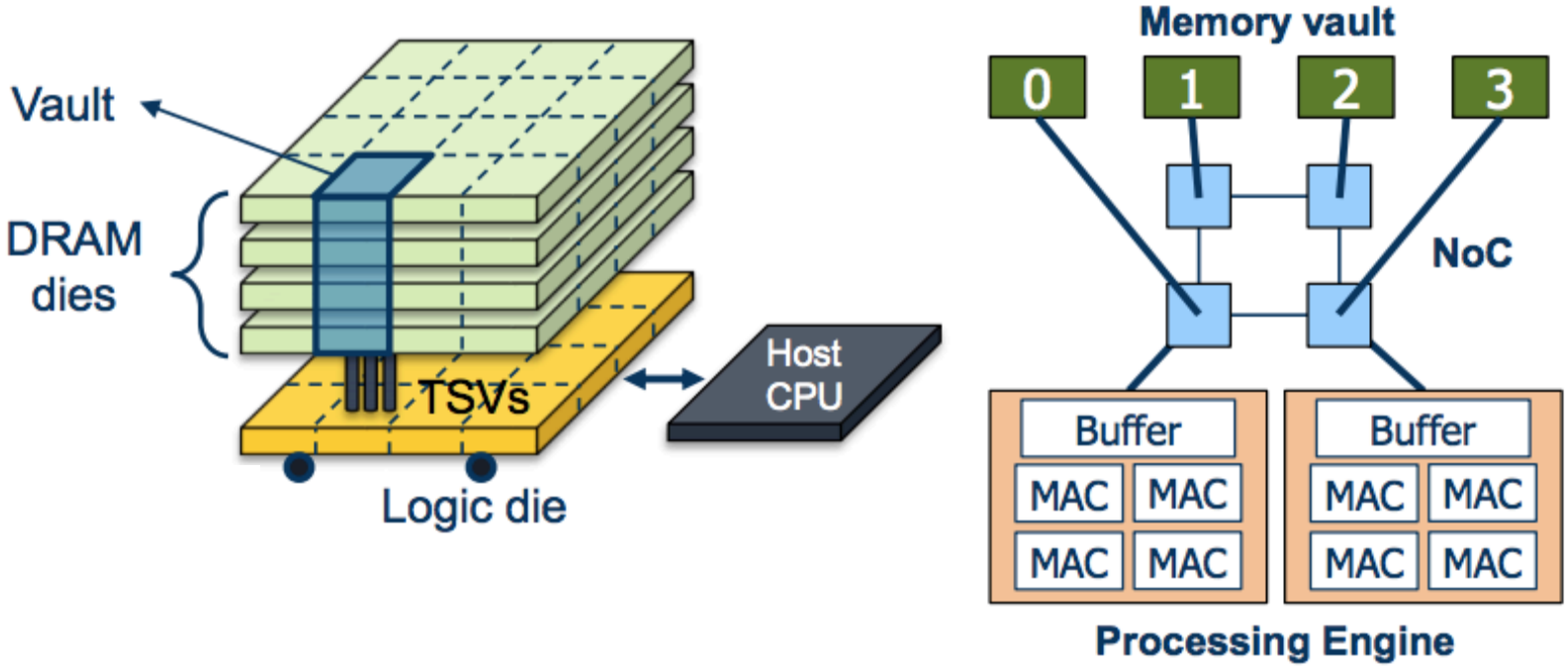




[Chen et al., DaDianNao, MICRO 2014]

Stacked DRAM (NeuroCube)

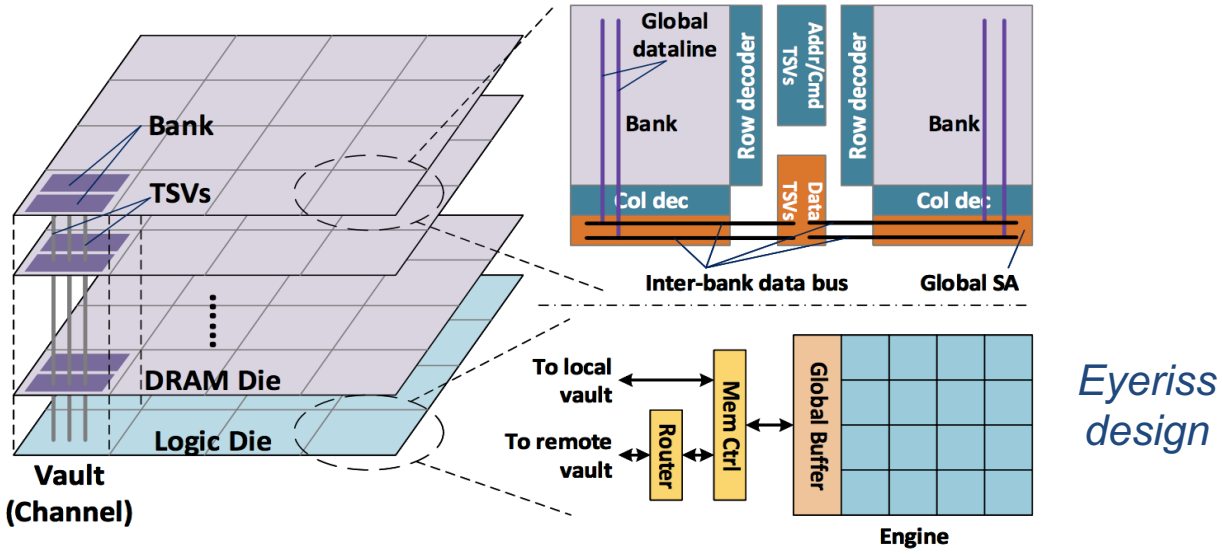
- NeuroCube on Hybrid Memory Cube Logic Die
  - 6.25x higher BW than DDR3
    - HMC (16 ch x 10GB/s) > DDR3 BW (2 ch x 12.8GB/s)
  - Computation closer to memory (reduce energy)



[Kim et al., NeuroCube, ISCA 2016]

Stacked DRAM (TETRIS)

- Explores the use of HMC with the Eyeriss **spatial architecture** and **row stationary dataflow**
- Allocates **more area to the computation (PE array) than on-chip memory (global buffer)** to exploit the low energy and high throughput properties of the **HMC**
  - **1.5x energy reduction, 4.1x higher throughput** vs. 2-D DRAM

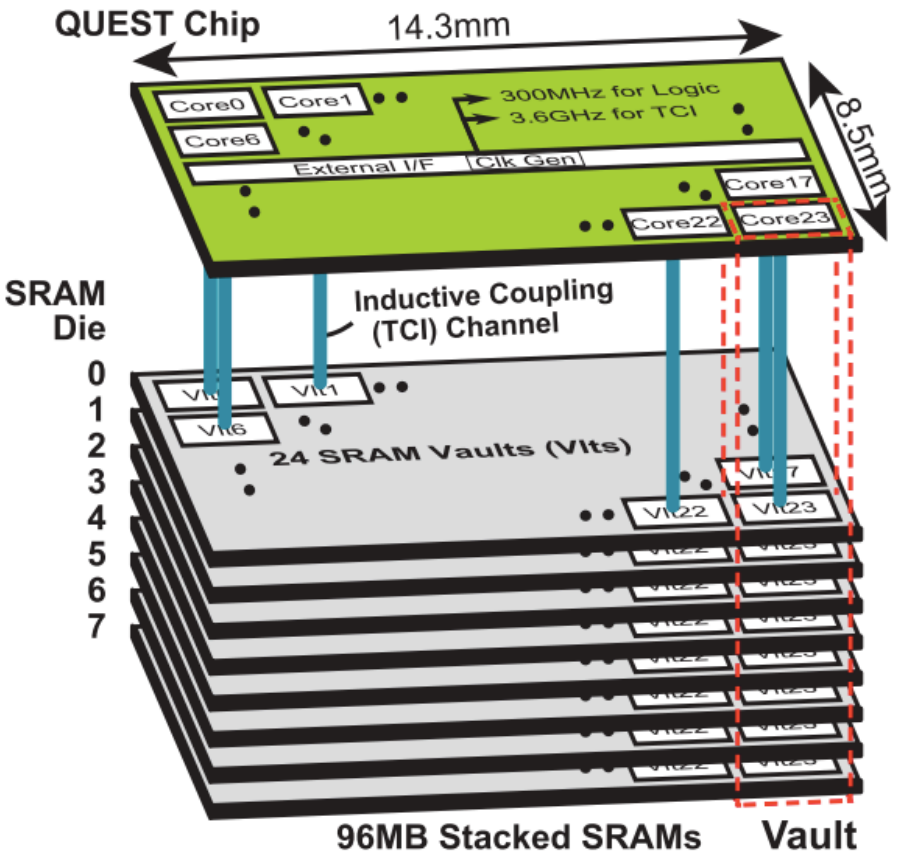


Eyeriss design

[Gao et al., Tetris, ASPLOS 2017]

Stacked SRAM (Quest)

- Explores use of stacked SRAM which has lower latency and power dissipation than DRAM
  - Lower storage density than DRAM but can stack more due to low power density
- Use inductive coupling to connect to SRAM rather than TSV for lower integration cost
  - 9.6Gb/s per channel x 24 channels = 28.8 GB/s



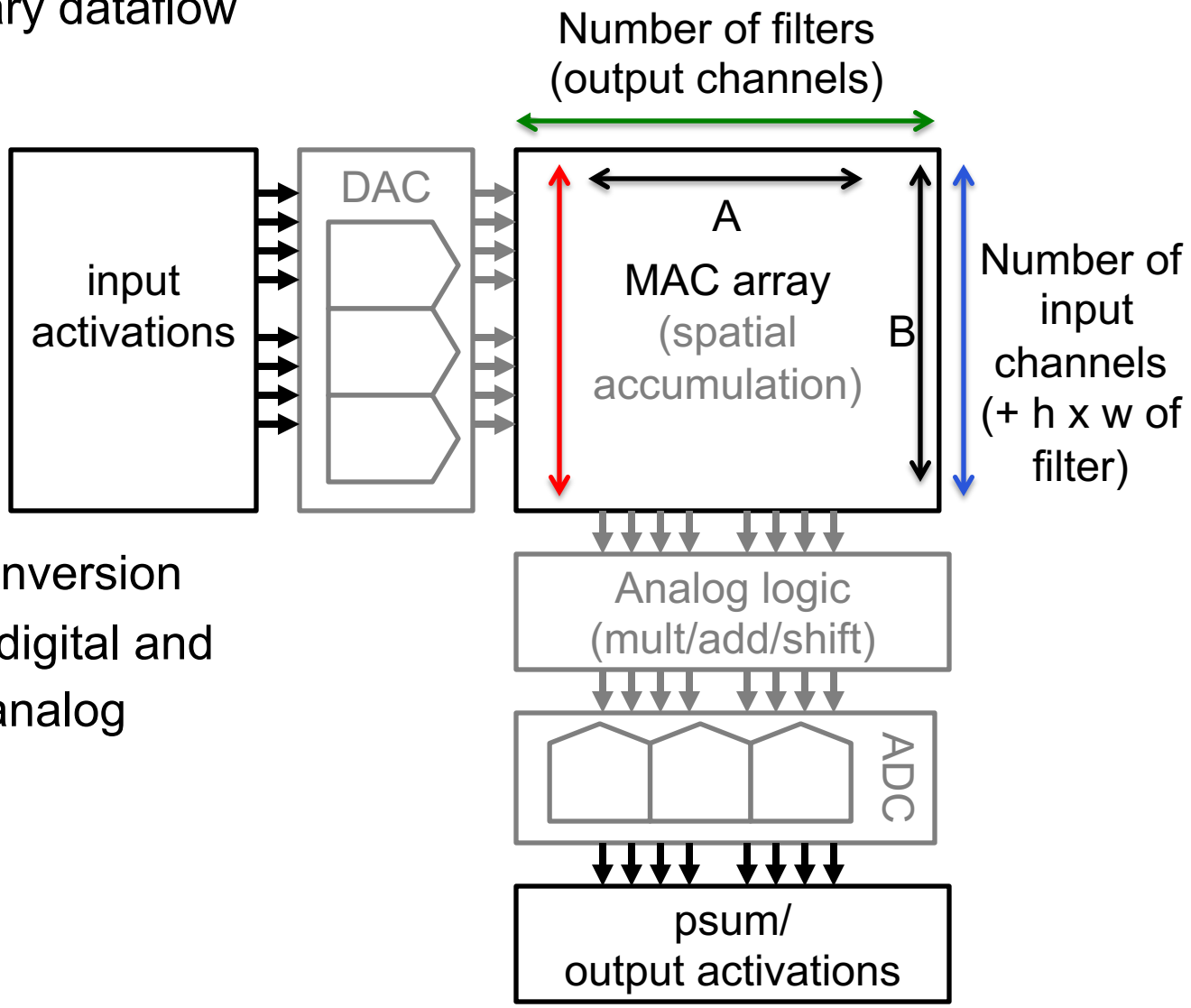
[Ueyoshi, ISSCC 2018]

Processing In Memory

- Embed processing in peripheral or perform compute in analog domain
- Benefits
  - read multiple weights in parallel (high bandwidth)
  - perform multiple computations in parallel (high throughput)
  - Increase density of compute engines (lower capacitance on input delivery)
  - reduce number of conversions (sense amp only applied to final accumulated value)
- Challenges
  - non-idealities of analog compute (non-linearity, sensitivity to process and temperature variations)
  - conversion cost from analog to digital (fewer conversions, but each one is more expensive)
  - increase size of bit cell and increase size in peripheral (reduce storage density)
  - limited precision that can be stored in each storage element (e.g. bit cell, device); need to use bit serial or multiple elements if want to support higher precision; also may need to use two arrays to enable differential signal to represent negative values
- Memory Technology
  - SRAM, DRAM, NVM (Flash, memristors)

Dataflow for PIM

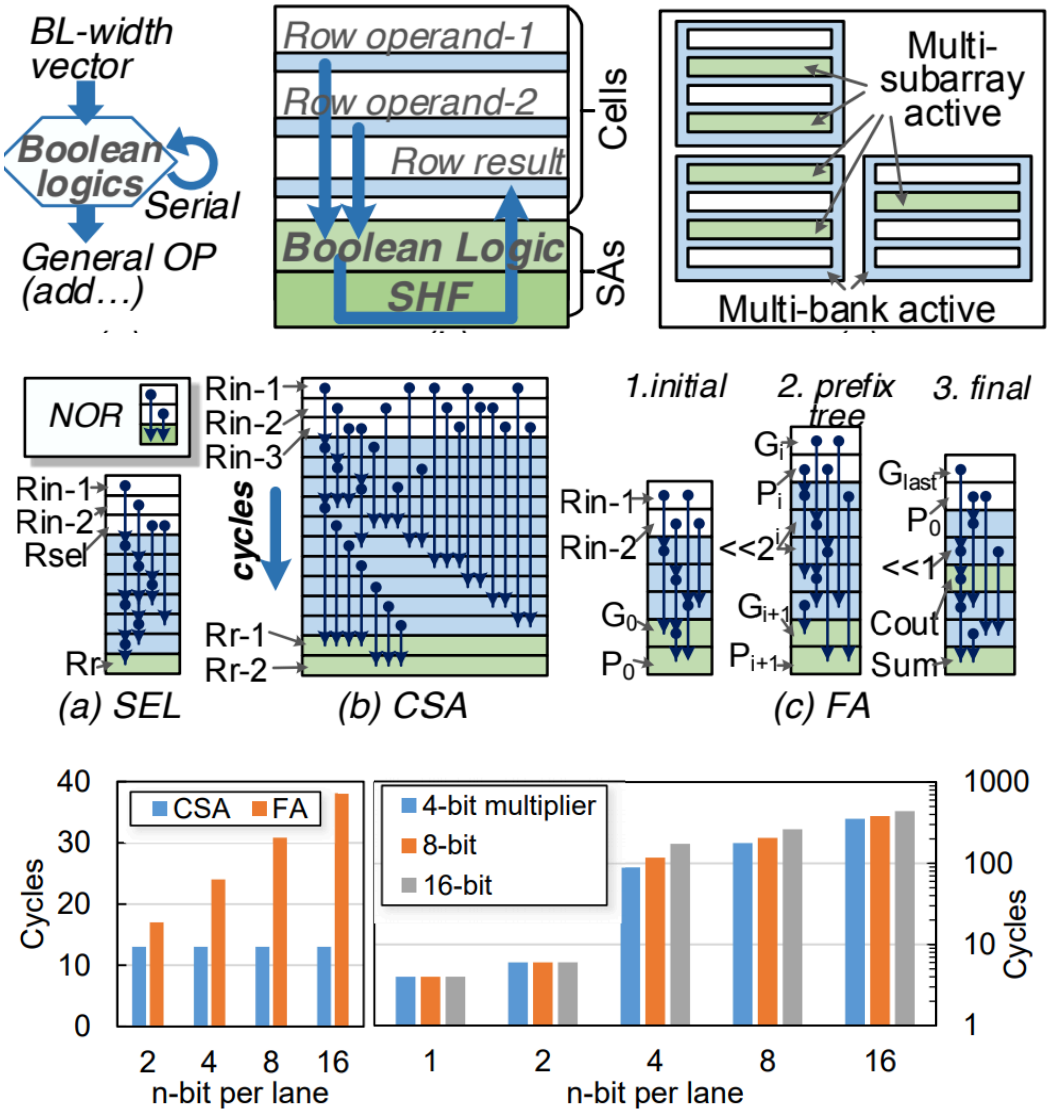
- Weight stationary dataflow



- Overhead in conversion from analog to digital and digital back to analog

DRAM (DRISA)

- Build multiplier and adders from NOR logic in DRAM
  - Add shift logic to output of DRAM before write back
- Requires multiple cycles
  - Number of cycles depends on bit width
  - Tradeoff parallelism for cycles
- Parallelism dictated by array width and number of arrays
- Demonstrate on BNN: 1b weights, 8 bit activation for AlexNet, VGG-16, VGG-19, ResNet



[Li, MICRO 2017]

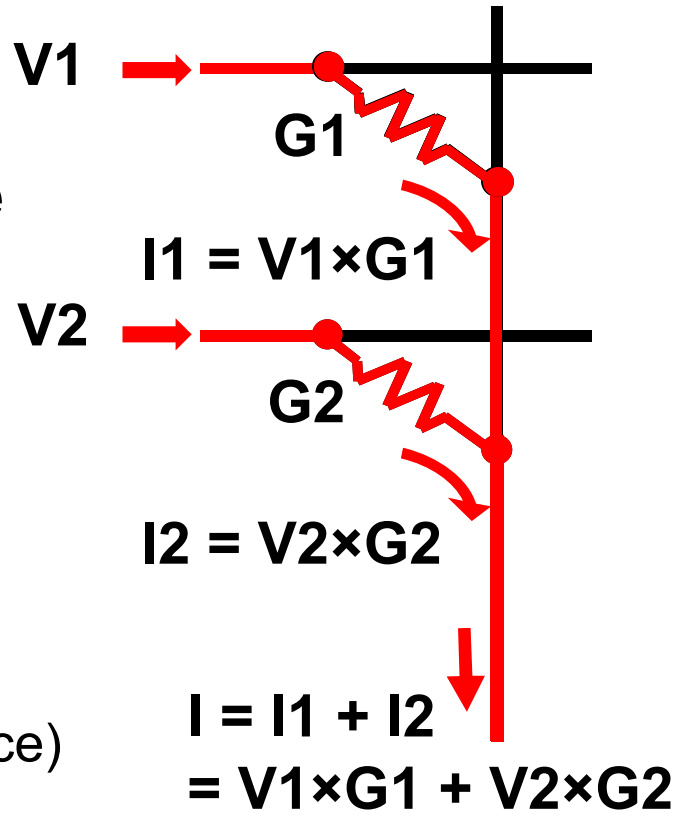
Analog Computation

- **Conductance = Weight**
- **Voltage = Input**
- **Current = Voltage × Conductance**
- **Sum currents for addition**

$$Output = \sum Weight \times Input$$

Input = V1, V2, …

Filter Weights = G1, G2, … (conductance)

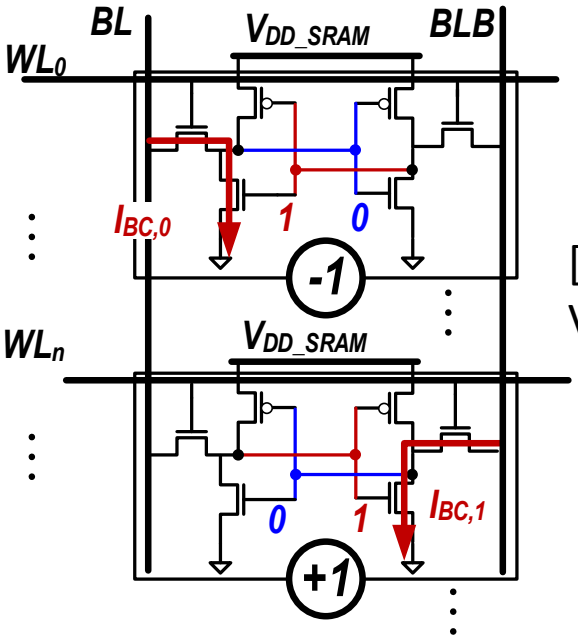


Weight Stationary Dataflow

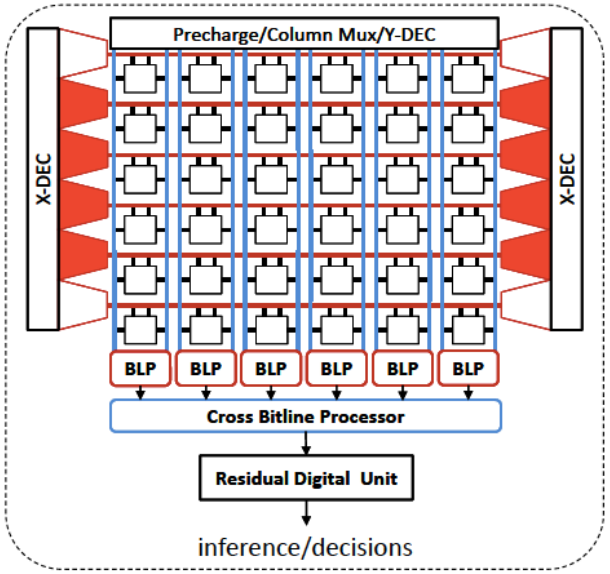
Figure Source: ISAAC, ISCA 2016

Compute Dot Product in SRAM

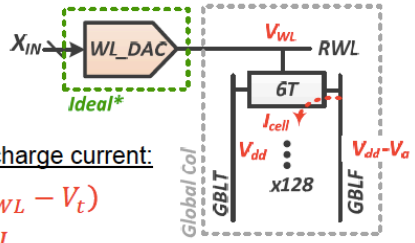
Approaches vary in terms of how data applied to BL and/or WL



[J. Zhang, VLSI 2016]



[S. Gonugondla, ISSCC 2018]



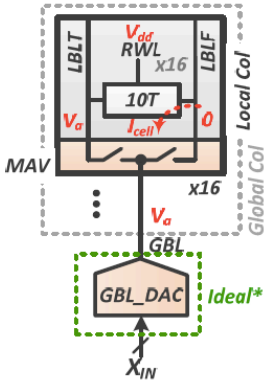
Bit-cell discharge current:

$I_{cell} \propto (V_{WL} - V_t)$

$V_a \propto I_{cell}$

***Conventional* [4]**

V_a varies widely due to I_{cell} variation



Assumption:
Both DACs are ideal

Directly apply V_a on GBL(LBL) by DAC

V_a ✗ I_{cell}

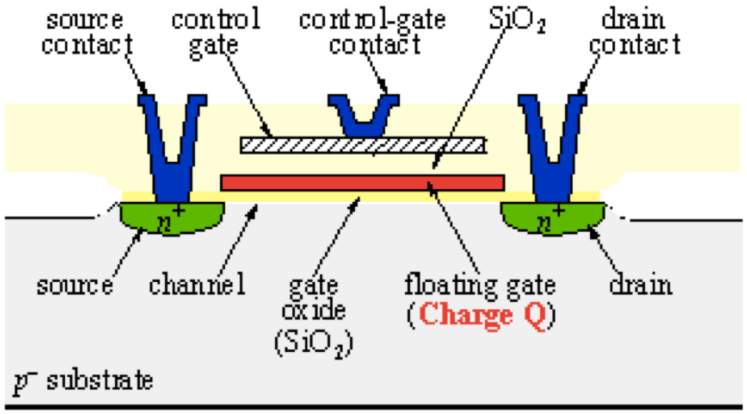
Proposed

V_a has no variation due to I_{cell}

[A. Biswas, ISSCC 2018]

Analog Compute with Flash Memory

Floating Gate: Program by changing threshold voltage (shift I-V curve)



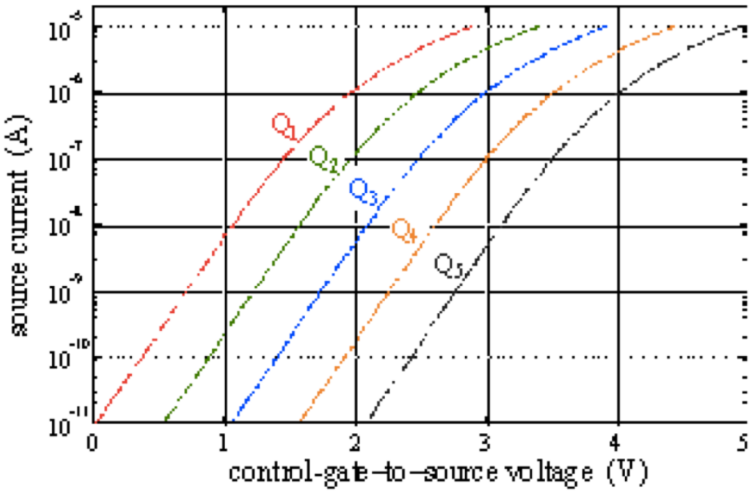


Image source: Diorio, ISFIIE 1999

	Enterprise With DRAM	Enterprise No-DRAM	Edge With DRAM	Edge No-DRAM	Mythic NVM
SRAM	<50 MB	100+ MB	< 5 MB	< 5 MB	< 5 MB
DRAM	8+ GB	-	4-8 GB	-	-
Power	70+ W	70+ W	3-5 W	1-3 W	1-5 W
Sparsity	Light	Light	Moderate	Heavy	None
Precision	32f / 16f / 8i	32f / 16f / 8i	8i	1-8i	1-8i
Accuracy	Great	Great	Moderate	Poor	Great
Performance	High	High	Very Low	Very Low	High
Efficiency	25 pJ/MAC	2 pJ/MAC	10 pJ/MAC	5 pJ/MAC	0.5 pJ/MAC

Source: Mythic, Hot Chips 2018

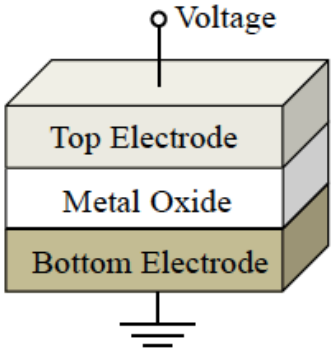
Memristor Computation

Use memristors as programmable weights (resistance)

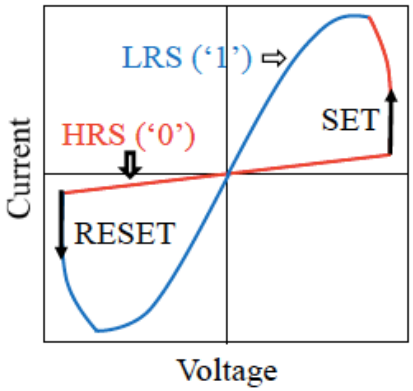
- **Advantages**
  - **High Density (< 10nm x 10nm size*)**
    - **~30x smaller than SRAM****
    - **1.5x smaller than DRAM****
  - **Non-Volatile**
  - **Operates at low voltage**
  - **Computation within memory (in situ)**
    - **Reduce data movement**

*[Govoreanu et al., IEDM 2011], **ITRS 2013

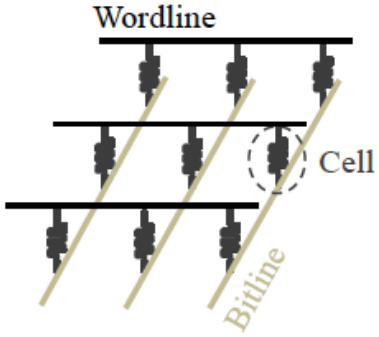
Memristor



(a) Conceptual view of a ReRAM cell

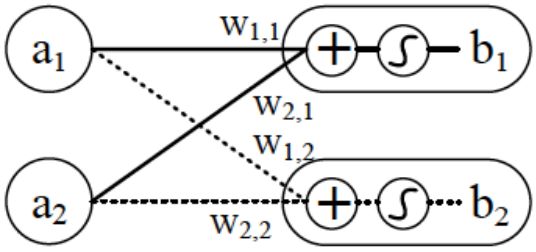


(b) I-V curve of bipolar switching

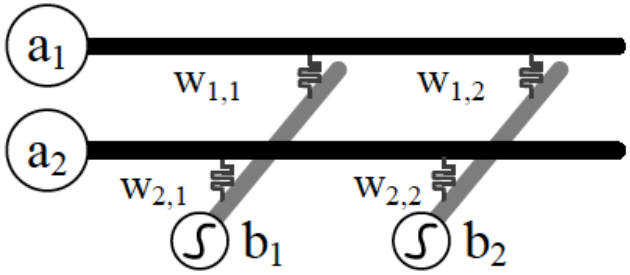


(c) schematic view of a crossbar architecture

$$b_j = \sigma\left(\sum_{\forall i} a_i \cdot w_{i,j}\right)$$



(a) An ANN with one input and one output layer



(b) using a ReRAM crossbar array for neural computation

[Chi et al., ISCA 2016]

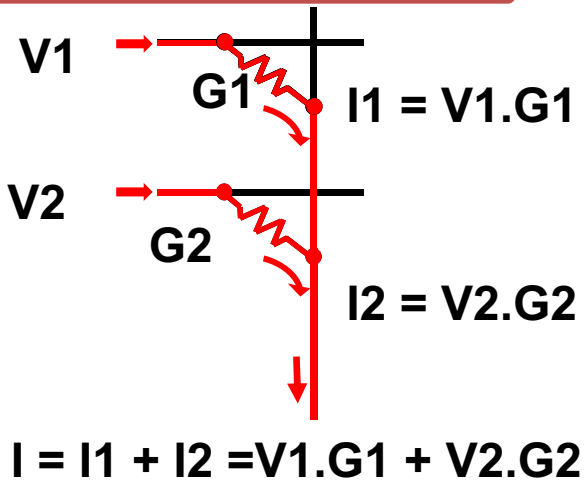
Challenges with Memristors

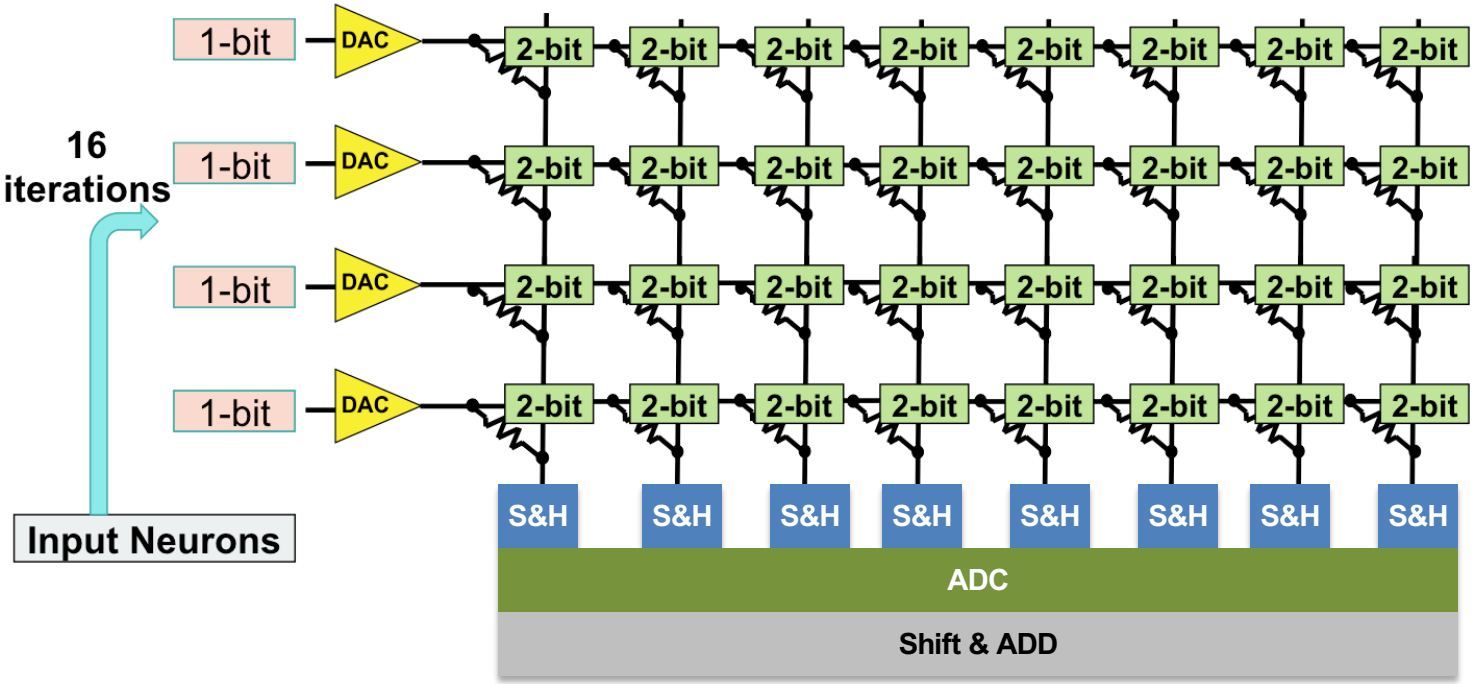
- **Limited Precision**
- **A/D and D/A Conversion**
- **Array Size and Routing**
  - **Wire dominates energy for array size of 1k × 1k**
  - **IR drop along wire can degrade read accuracy**
- **Write/programming energy**
  - **Multiple pulses can be costly**
- **Variations & Yield**
  - **Device-to-device, cycle-to-cycle**
  - **Non-linear conductance across range**

[Eryilmaz et al., ISQED 2016]

ISAAC

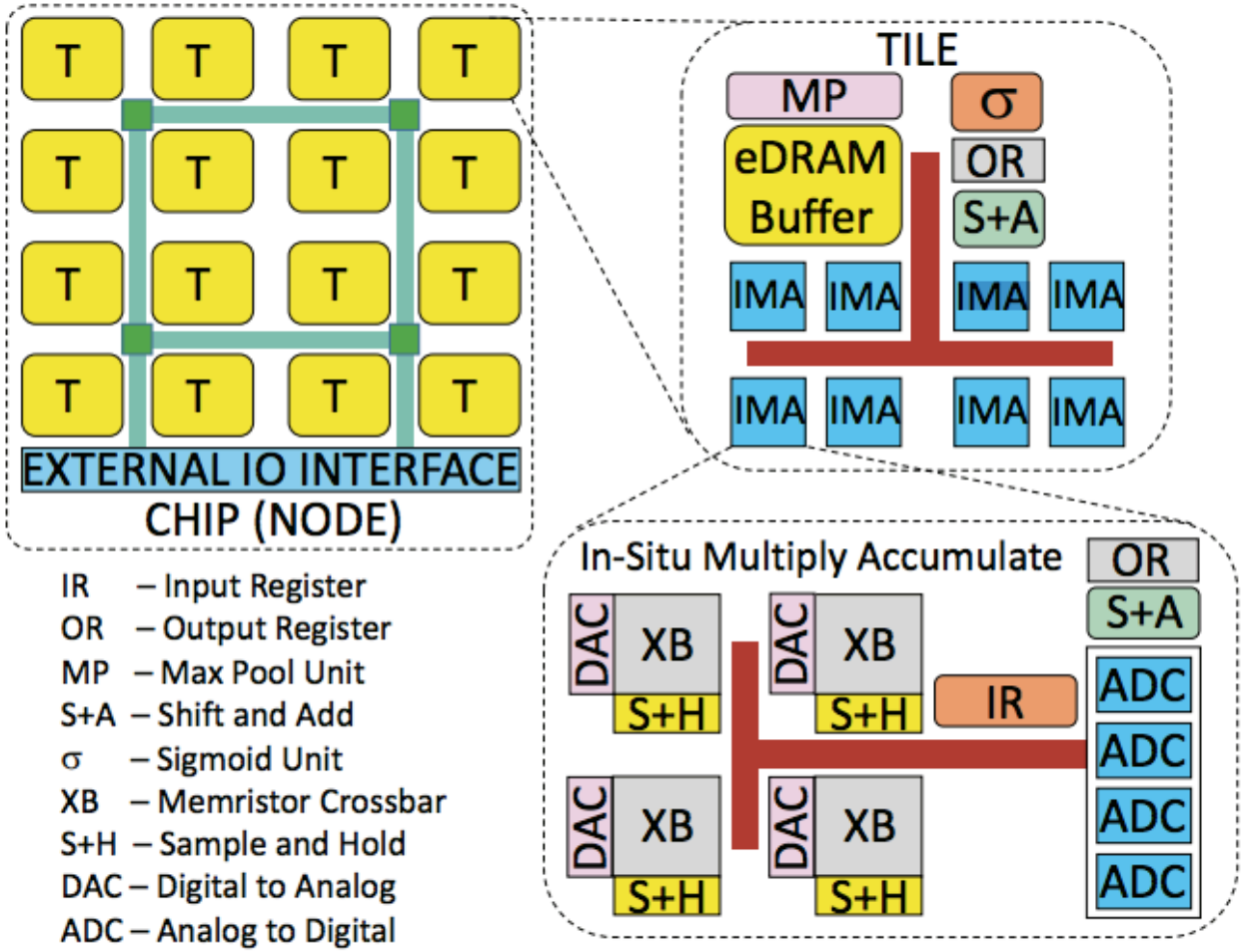
- eDRAM using memristors
- 16-bit dot-product operation
  - 8 x 2-bits per memristors
  - 1-bit per cycle computation
  - Trade off area and cycles to address low precision





[Shafiee et al., ISCA 2016]

ISAAC



Eight 128x128 arrays per IMA

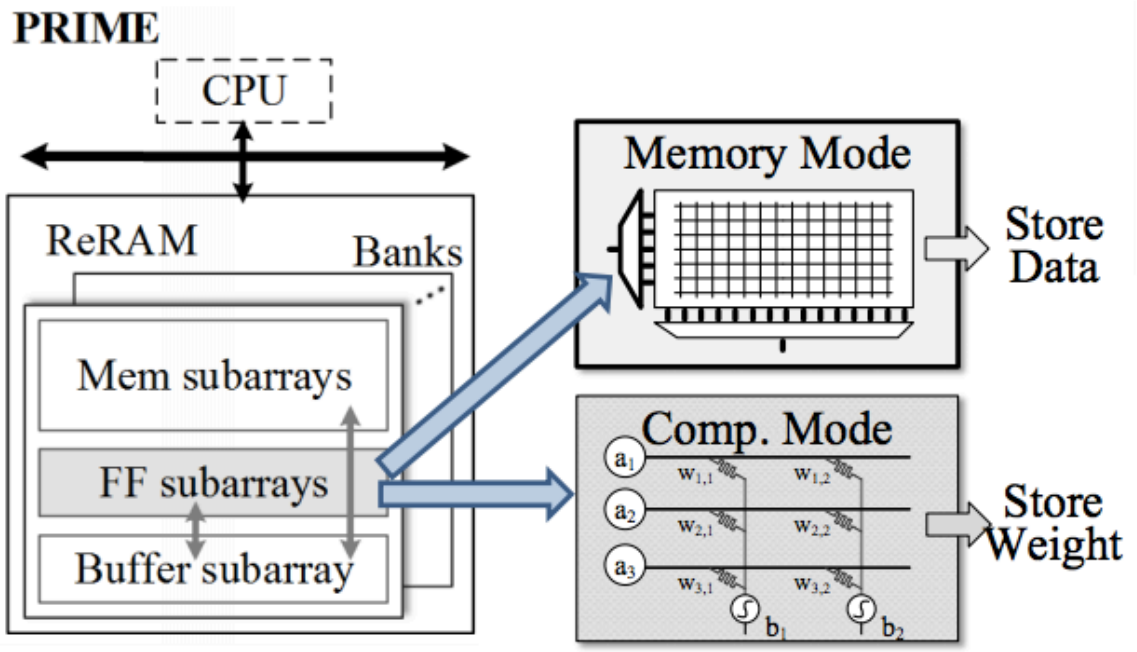
12 IMAs per Tile

14x12 Tiles in ISAAC

[Shafiee et al., ISCA 2016]

PRIME

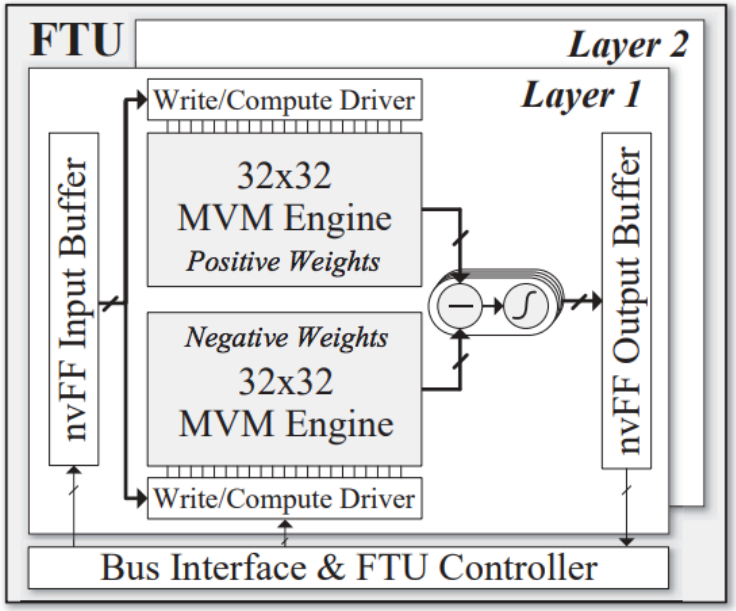
- Bit precision for each 256x256 ReRAM array
  - 3-bit input, 4-bit weight (2x for 6-bit input and 8-bit weight)
  - Dynamic fixed point (6-bit output)
- Reconfigurable to be main memory or accelerator
  - 4-bit MLC computation; 1-bit SLC for storage

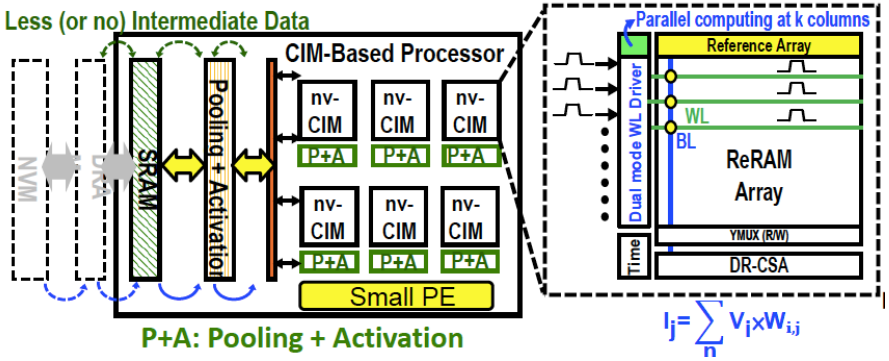


[Chi et al., ISCA 2016]

Fabricated Arrays

- Ternary weights and binary activations
- Use two arrays for ternary weights {-1, 0, +1}
- [Su, VLSI 2017]
  - Array size: 32x32 (2 arrays)
    - Accumulate 32 values into a 3-bit output
  - Demo Task
    - Input: 5x5 image (25 values)
    - Classify with 2 FC (25x32x4)
    - Output: 4 directions
- nvCIM [Chen, ISSCC 2018]
  - Array size: 512 x 256 (8 arrays)
    - Accumulate 512 values into a 3-bit output
    - Max 32 rows activated at a time due to sensing margin and yield
  - Demo Task
    - Input: 5x5 image (MNIST) – 96.2% accuracy
    - Classify with 2 FC (25x25x3) ??
    - Output: 9 digits



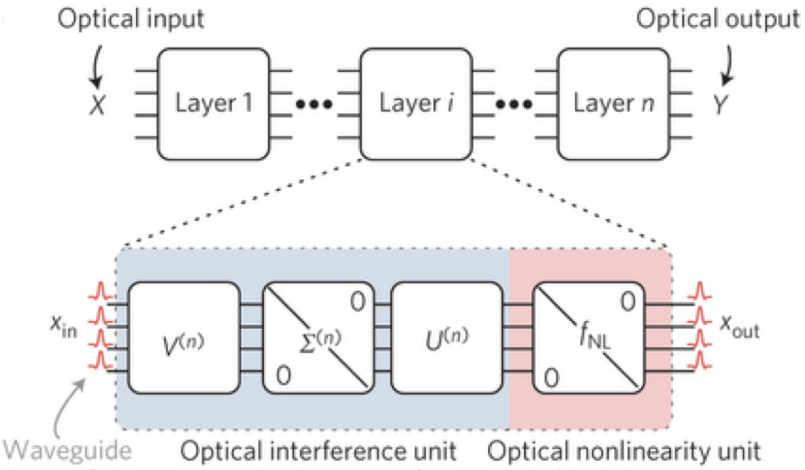


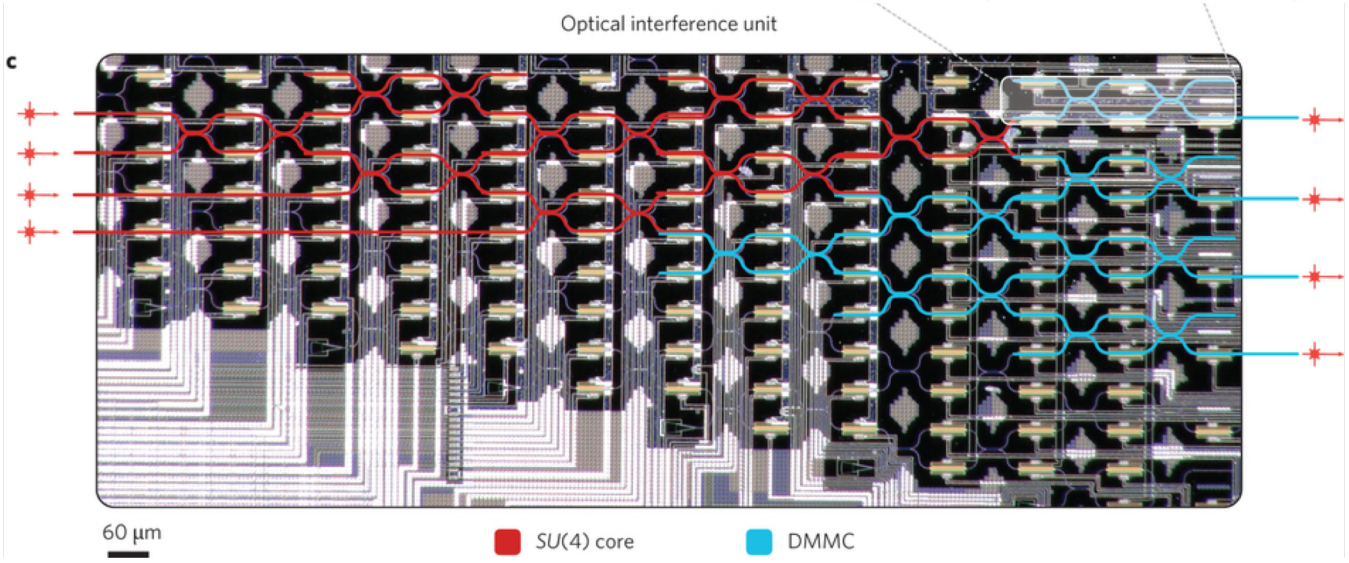
Optical Neural Network

Matrix Multiplication in the Optical Domain

The photodetection rate is 100 GHz

*"In principle, such a system can be at least **two orders of magnitude faster** than electronic neural networks (which are restricted to a GHz clock rate)"*





[Shen et al., Nature Photonics 2017]