NeuroCube: A Programmable Digital Neuromorphic Architecture with High-Density 3D Memory

Section 6 - 2

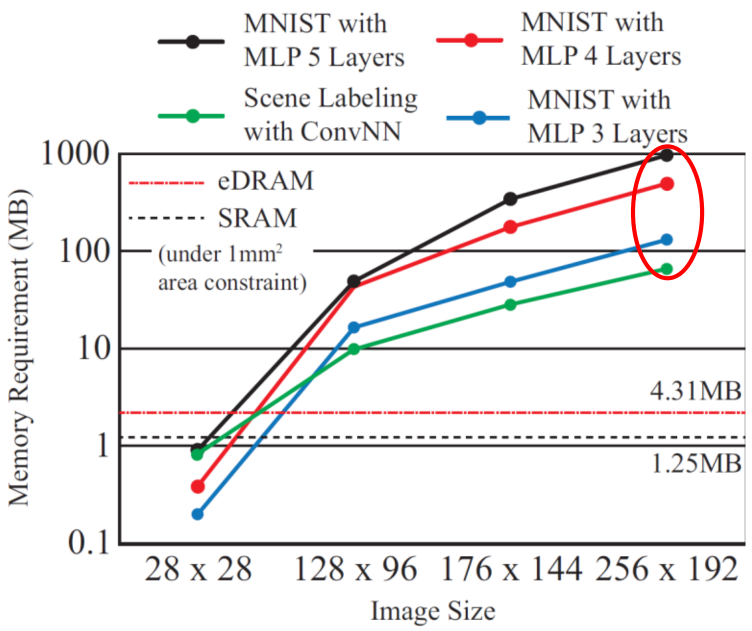
Duckhwan Kim[1], Jaeha Kung[1], Sek Chai[2],
Sudhakar Yalamanchili[1], Saibal Mukhopadhyay[1]

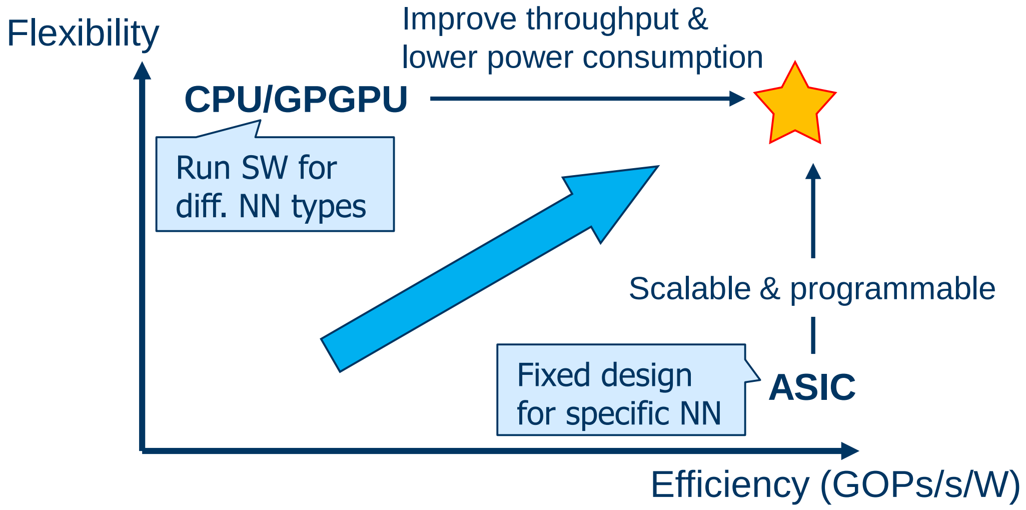
[1]School of Electrical and Computer Engineering, Georgia Institute of Technology
[2]SRI International

This material is based on work supported in part by an ONR Young Investigator award,
a National Science Foundation CAREER Award, and the National Science Foundation grant CCF 1337177

Outline

Digital Accelerator Design for Neuromorphic Algorithm



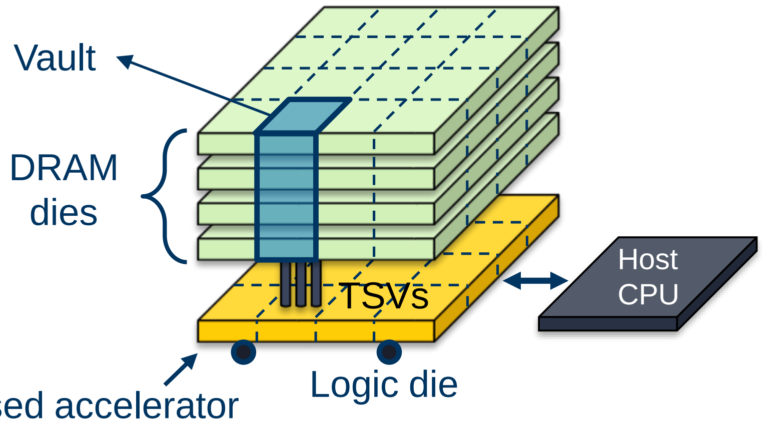

Flexibility
Improve throughput & lower power consumption
CPU/GPGPU
Run SW for diff. NN types
Scalable & programmable
Fixed design for specific NN
ASIC
Efficiency (GOPs/s/W)

- Low operation density (ops/byte)
- Massive date required

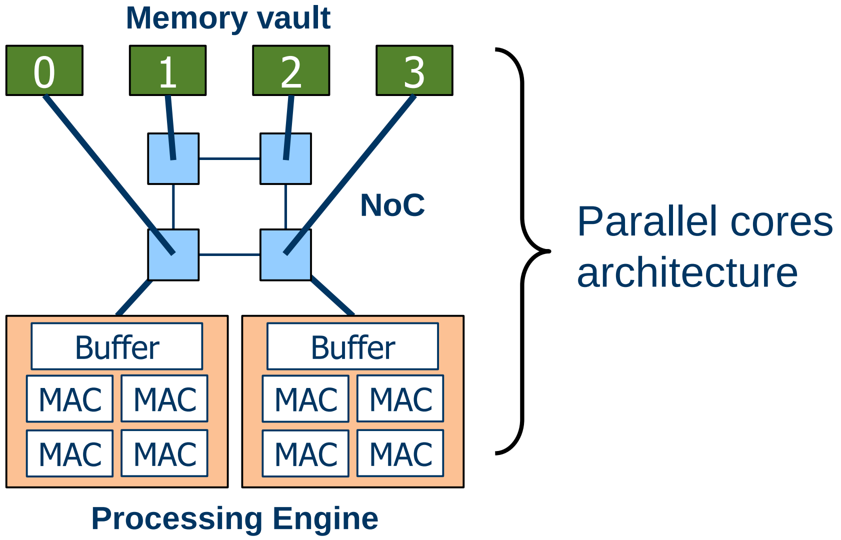
Digital neuro-inspired architecture with **programmability** to cover different types of neural networks, **scalability**, and **high energy efficiency**

NeuroCube: Process-in-Memory Architecture for Neural Computing

Programmable, scalable platform as processor in memory



PIM based accelerator for neural computing



Hybrid Memory Cube (HMC)

- Heterogeneous integration
- Flexible logic die design

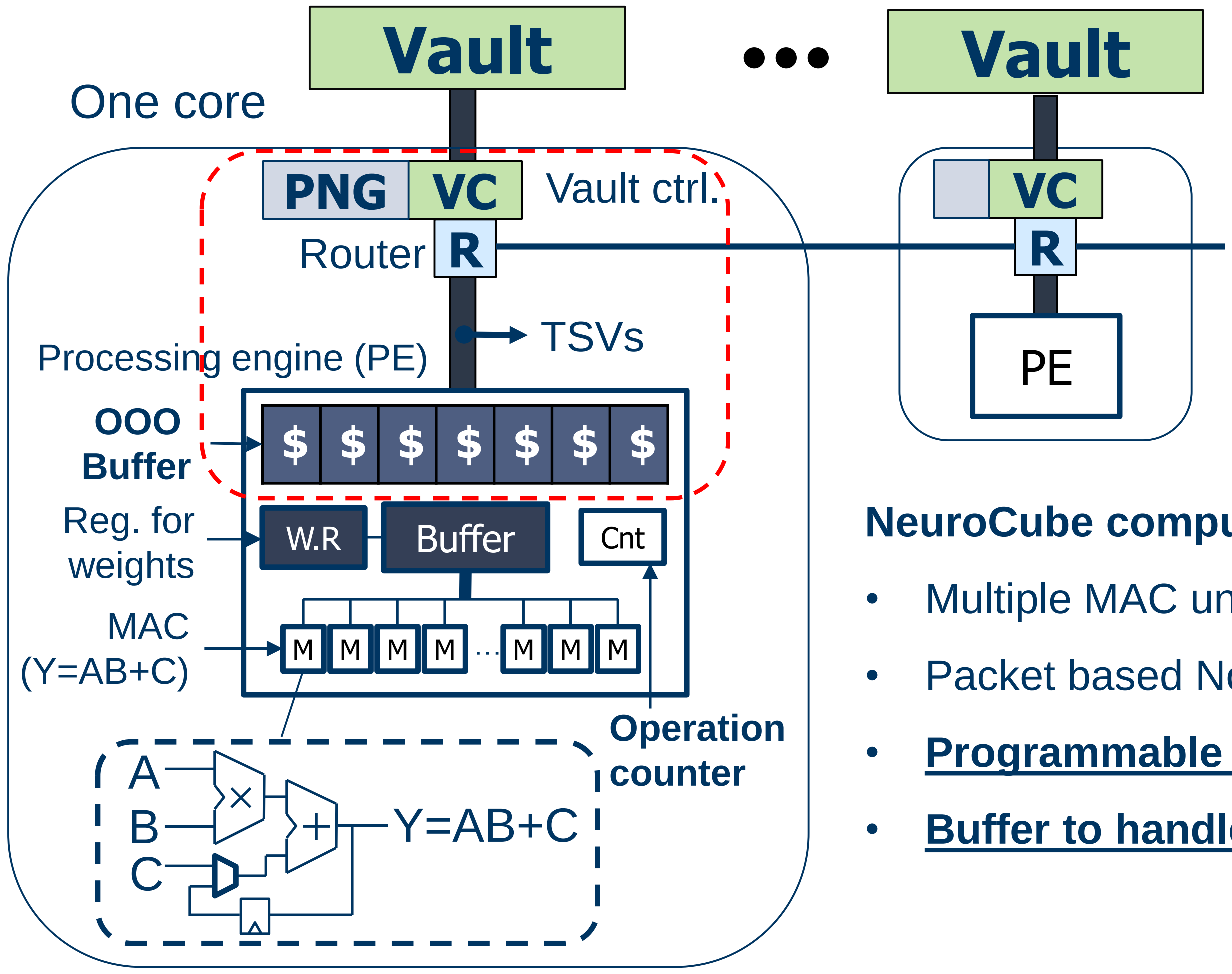
Fabricated by Micron
[J. Jeddeloh `12 TVLSI]

Q1. Neural computing layer should meet thermal and area constraint in 3D stacked DRAM

Q2. NeuroCube should be programmable to cover different types of neural network

Basic NeuroCube Architecture

Processor-in-memory + Parallelism



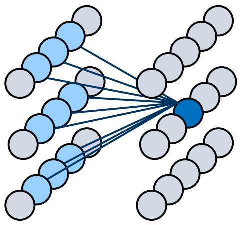
NeuroCube computing core

- Multiple MAC units and its temporal buffer
- Packet based NoC router
- **Programmable neurosequence generator**
- **Buffer to handle out-of-order packet arrival**

Operational Model of NeuroCube

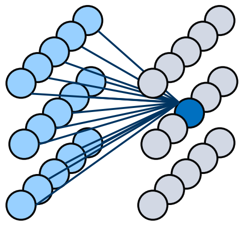
Property of Neural Network: Deterministic Connections in Inference

2D convolutional



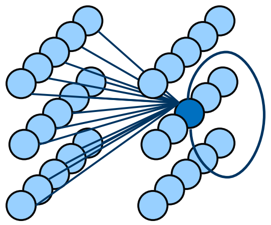
locally
neighborhood

Fully connected



All neurons
in prev. layer

Recurrent connected



All neurons
in prev. layer + current layer

General expression of artificial neural network (ANN)

$$y_j = \varphi\left(\sum_{i \in I} w_{(j,i)} \cdot y_i\right)$$

I: set of connected neurons

- Different NN layer can be mapped by changing set of connected neurons
  - Different data movements (memory address) can map different NN layers in NeuroCube

Property of Neural Network: Deterministic Connections in Training

- Backpropagation with gradient descent
  - $\delta_i = \left(W_{i,i+1}^T \times \delta_{i+1}\right) * \varphi'(y_i)$ (hidden) or $= -(d - y_i) * \varphi'(y_i)$ (output)
    - φ' : derivative of NL activation function
    - $*$: element-wise multiplication
    - γ: learning rate
  - $\Delta W_{i-1} = \delta_i \times y_{i-1}^T\,, W_{i-1} = W_{i-1} + \gamma \Delta W_{i-1}$

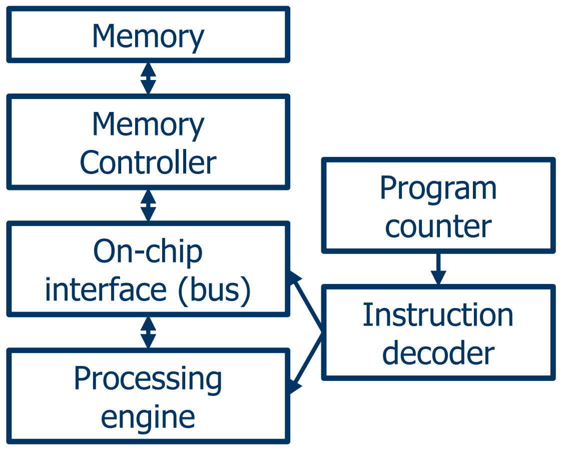
- It is composed of
  - Matrix-vector multiplication, element-wise multiplication, and outer product
  - Can be mapped to FC layer with <u>many zeros</u> in matrix

- Training in neural network <u>still has deterministic connections</u>

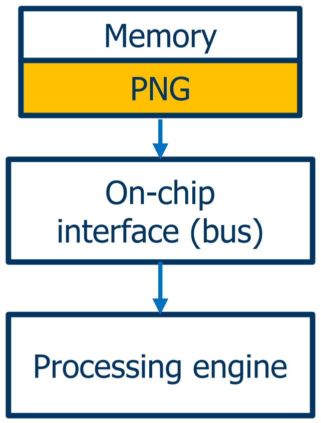
Memory Centric Neural Computing

Programmable Neurosequence Generator (PNG)

- **Sequence of operands** is predetermined
- Based on the sequence, memory **can push the data** without request
- Data is delivered as packet through NoC



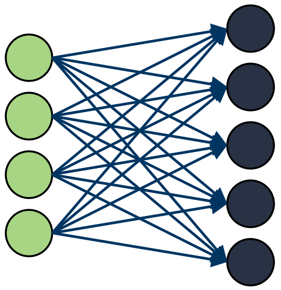
Conventional ISA operation



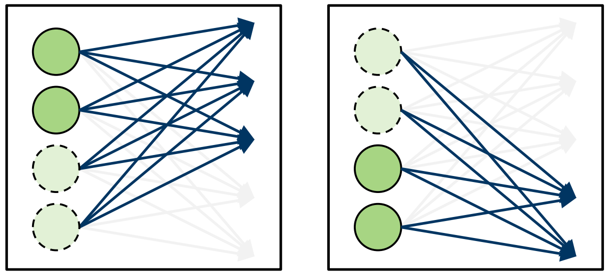
Memory centric operation

MCNC Data Flow – Initial Memory Mapping

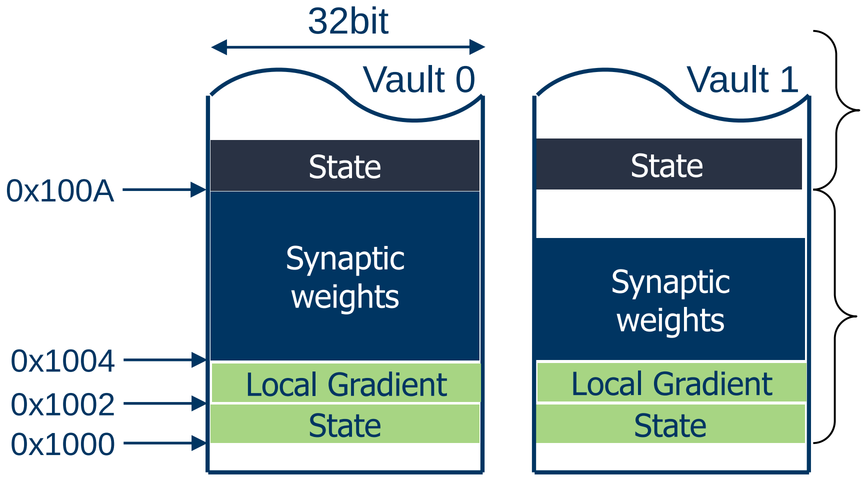
- Assume 2 vaults in NeuroCube, 3 MACs/PE
  - For synaptic weights
    - Divide synaptic weights matrix into 2 vaults
  - For previous layer
    - Divide input previous into 2 vaults or
    - **Duplicate prev. layer into all vaults** (reduce NoC traffic)



$W_{1,2}$= [5 by 4] matrix



- All neuron's states and synaptic weights are 16bit fixed point

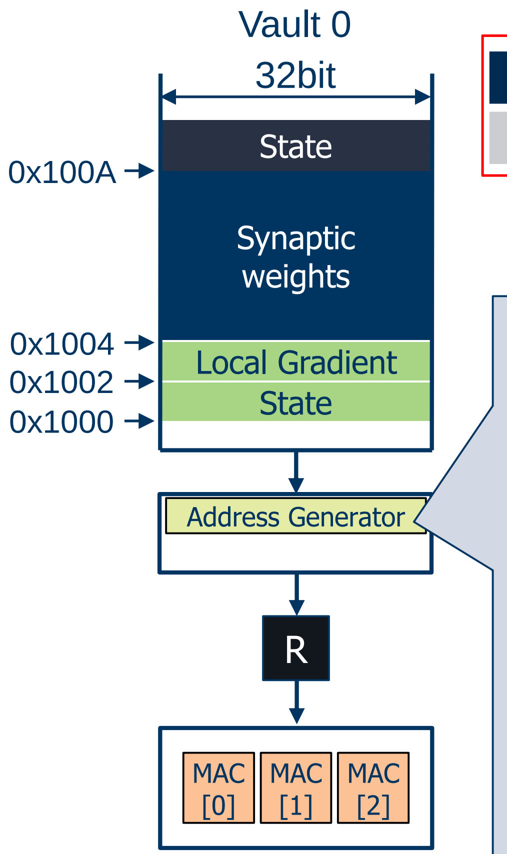


Memory range for next layer

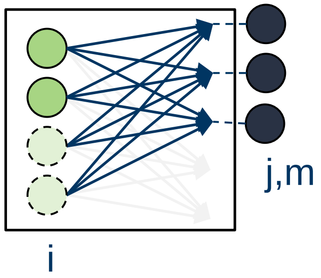
Memory range for this layer

MCNC Data Flow: Address Generator

- Assume 2 vaults in NeuroCube, 3 MACs/PE
- For each vault, it will push data to NoC within packet form



SRC	DST	DATA	MAC-ID	OP-ID
4bit	4bit	16bit	4bit	8bit



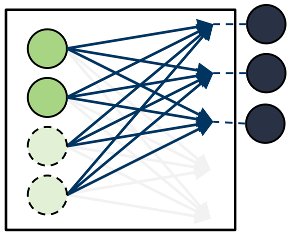
```
for(j=0;j<1;j++) {//one MAC cover one neuron[j]
   for(i=0;i<4;i++) {//4 neurons[i] to one neuron[j]
      for(k=0;k<3;k++) {// state/weight/local grad
         for(m=0;m<3;m++) {//#MACs/PE = 3
            if k==0
             addr = addr_gen(i,j,m,state)
          else if k==1
             addr = addr_gen(i,j,m,weight)
          else
             addr = addr_gen(i,j,j,local_grad)
         }
       }
   }
}
```

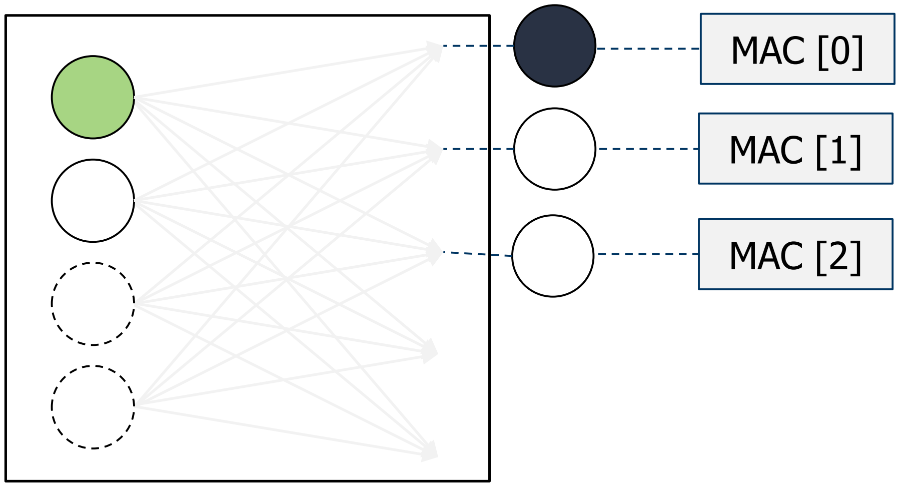
Three nested-counters

MCNC Data Flow (1)

Push packet (state[0]) to MAC [0]

SRC	DST	DATA	MAC-ID	OP-ID
0	0	State [0]	0	0

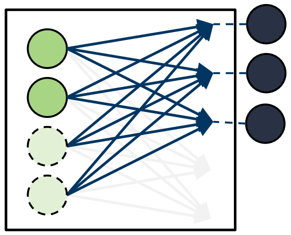


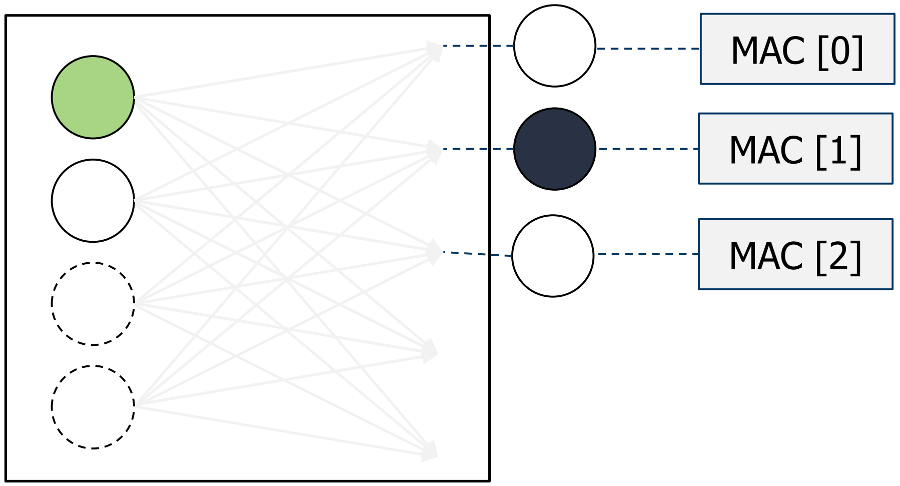


MCNC Data Flow (2)

Push packet (state[0]) to MAC [1]

SRC	DST	DATA	MAC-ID	OP-ID
0	0	State [0]	1	0

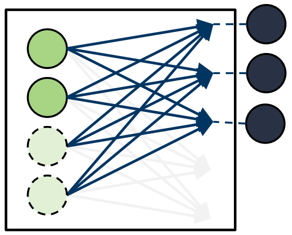


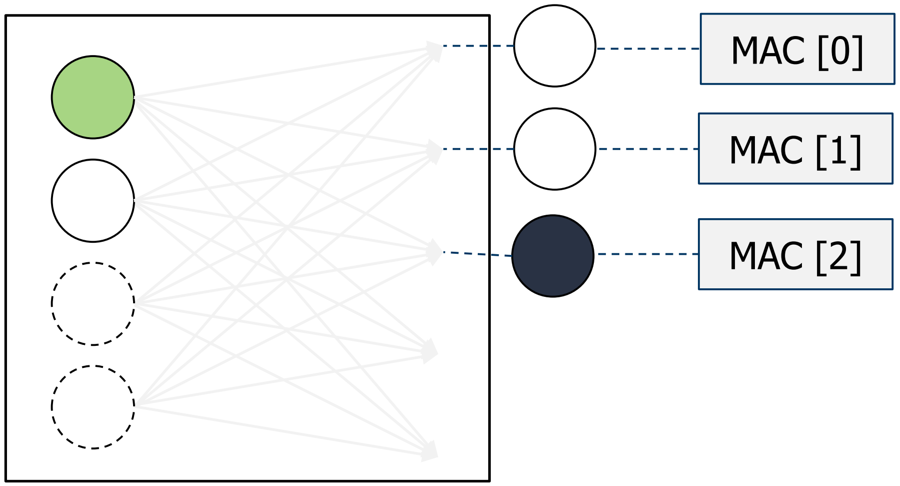


MCNC Data Flow (3)

Push packet (state[0]) to MAC [2]

SRC	DST	DATA	MAC-ID	OP-ID
0	0	State [0]	2	0

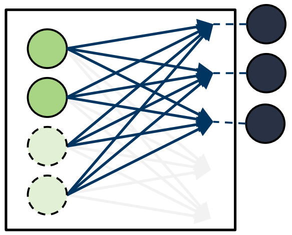


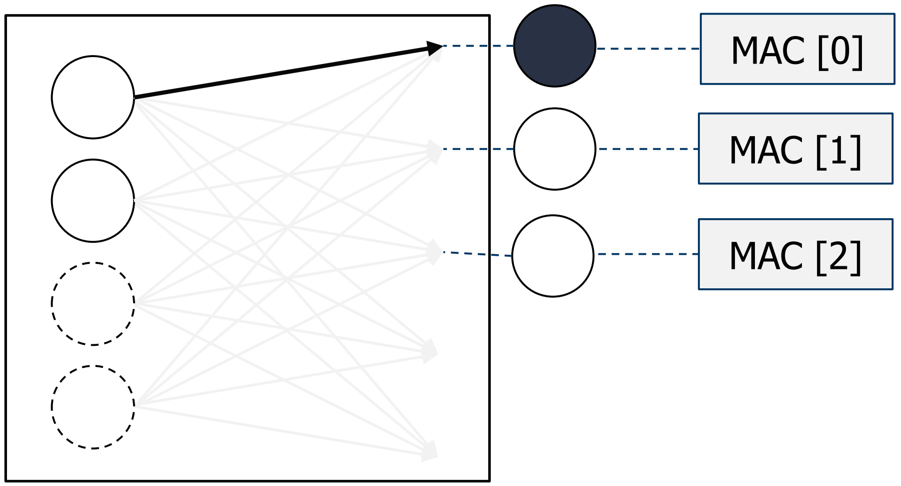


MCNC Data Flow (4)

Push packet (weight[0,0]) to MAC [0]

SRC	DST	DATA	MAC-ID	OP-ID
0	0	W [0, 0]	0	0

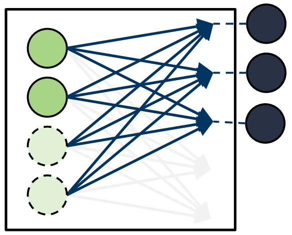


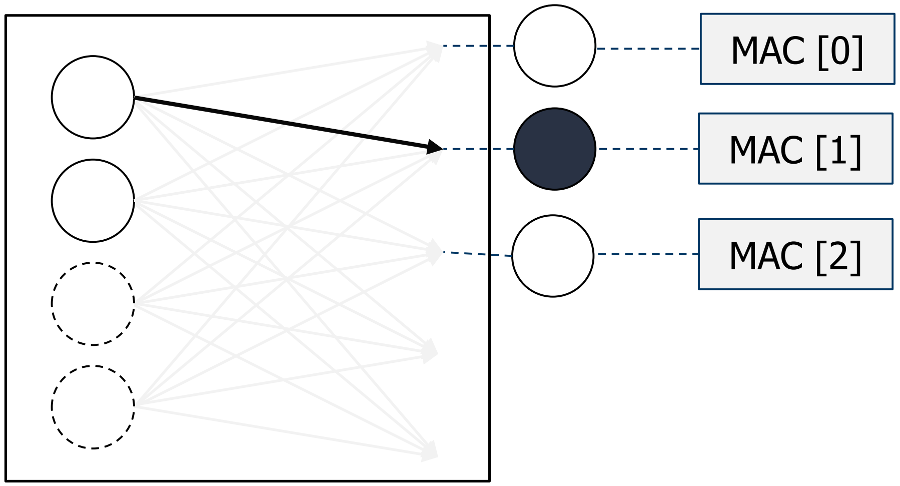


MCNC Data Flow (5)

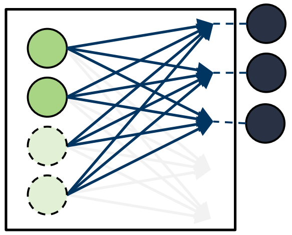
Push packet (weight[0,1]) to MAC [1]

SRC	DST	DATA	MAC-ID	OP-ID
0	0	W [0, 1]	1	0



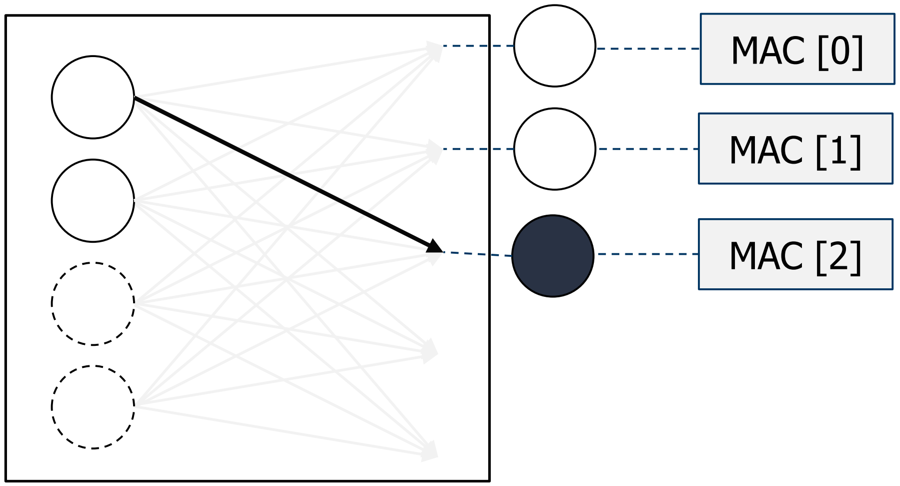


MCNC Data Flow (6)



Push packet (weight[0,2]) to MAC [2]

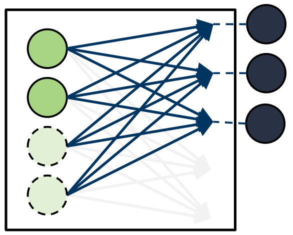
SRC	DST	DATA	MAC-ID	OP-ID
0	0	W [0, 2]	2	0

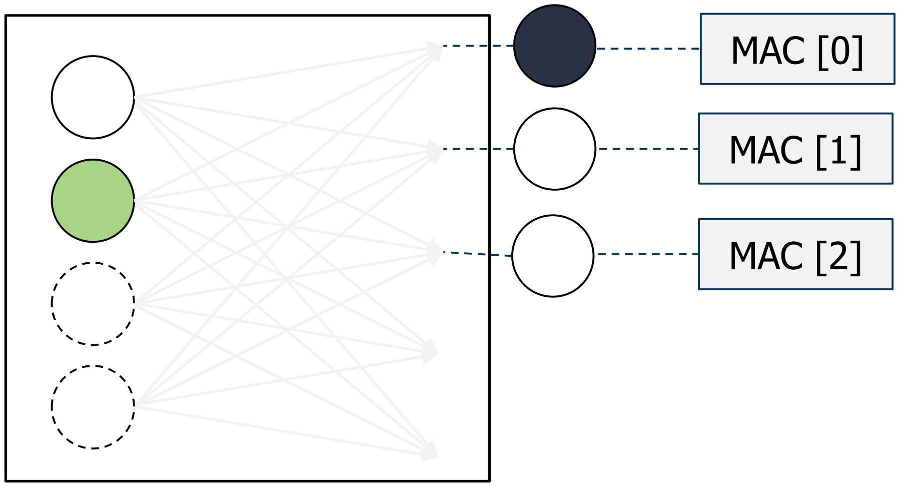


MCNC Data Flow (7)

Push packet (state[1]) to MAC [0]

SRC	DST	DATA	MAC-ID	OP-ID
0	0	State [1]	0	1

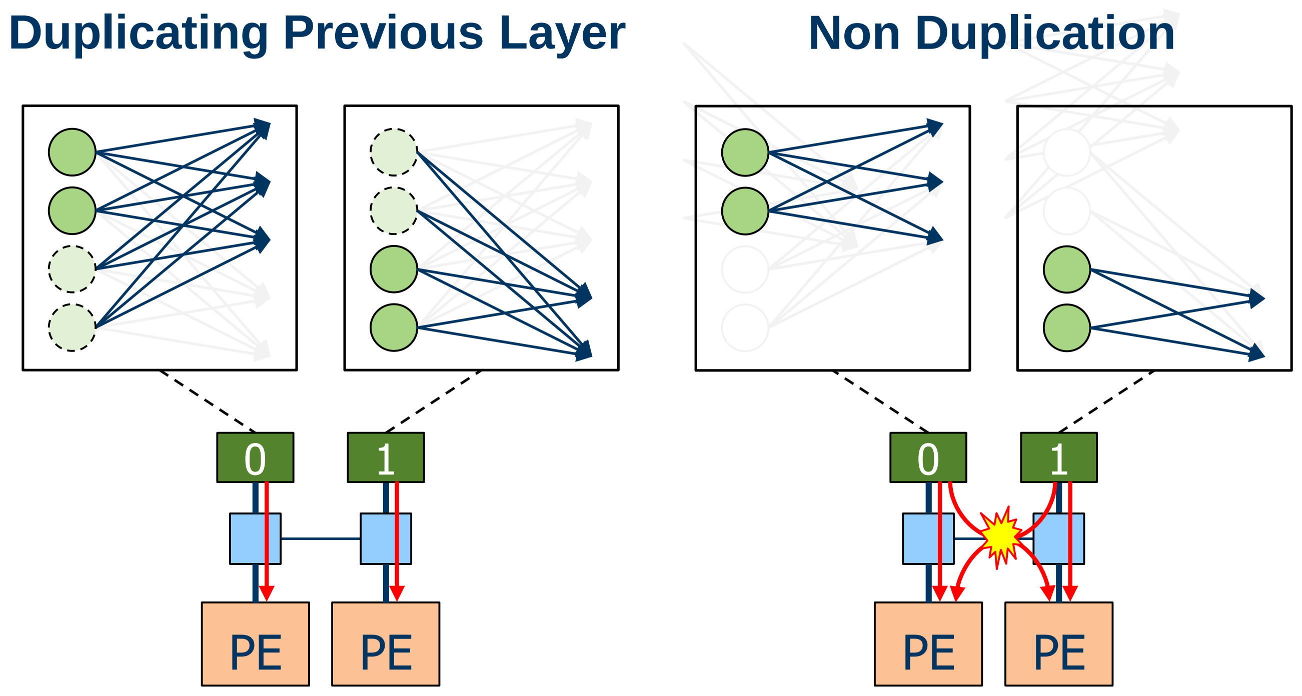


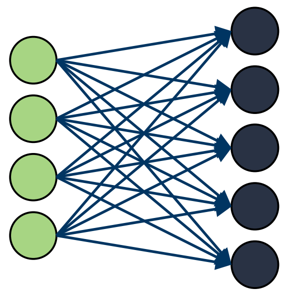


And so on …

Out-of-order Packet Arrival Problem in NeuroCube

Out-of-order Packet Arrival Issue: NoC Congestion

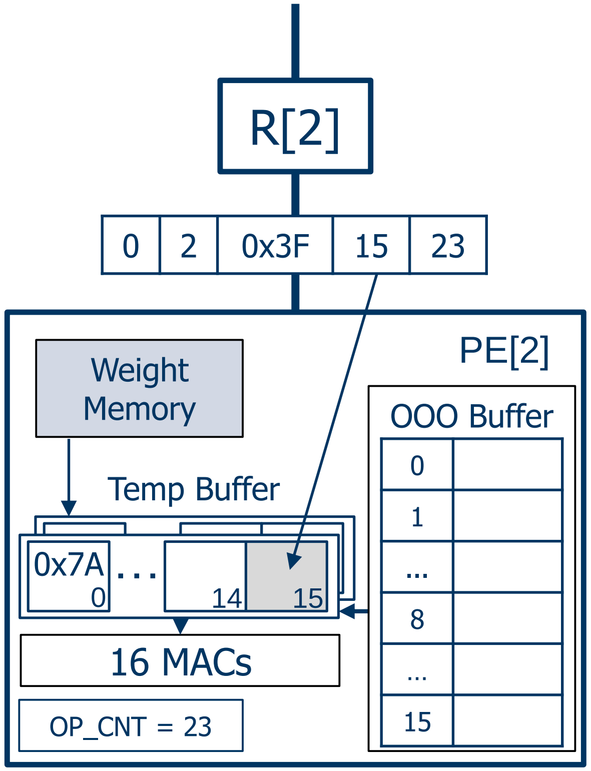




$W_{1,2}$= [5 by 4] matrix

Although data access is sequential, data arrival can be **out-of-order** due to NoC congestion

Out-of-Order data arrival (1)

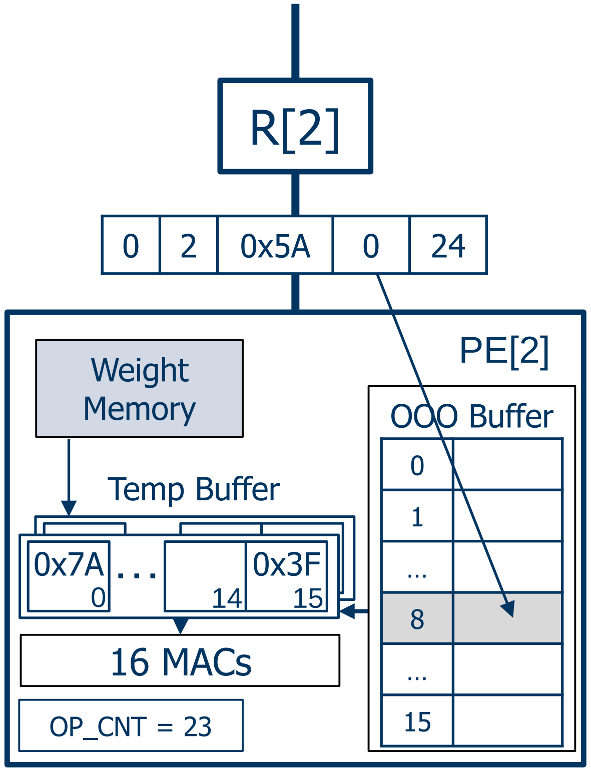


Packet structure

SRC	DST	DATA	MAC-ID	OP-ID
4bit	4bit	16bit	4bit	8bit

- OP-ID == OP_CNT == 23
  - This packet is for current operation
  - It moves to temporal buffer [15] directly

Out-of-Order data arrival (2)

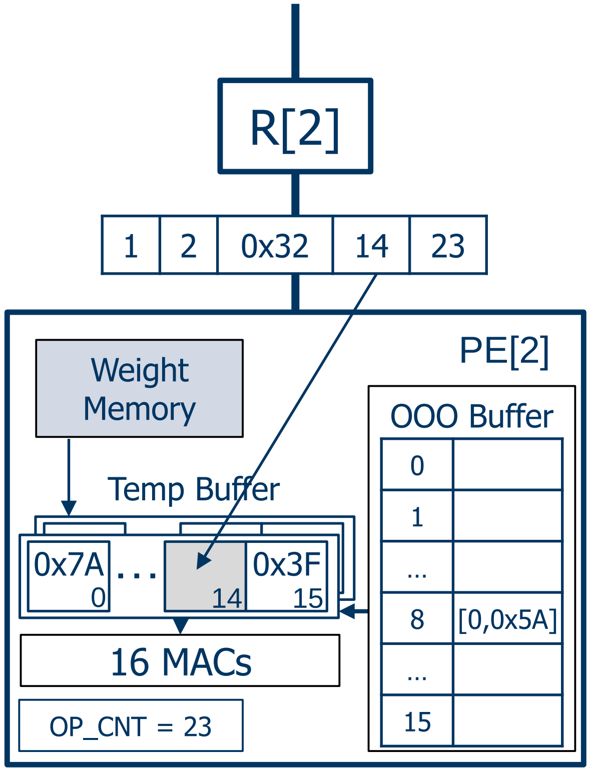


Packet structure

SRC	DST	DATA	MAC-ID	OP-ID
4bit	4bit	16bit	4bit	8bit

- OP-ID != OP_CNT
  - This packet is for next operation (OP_CNT == 24)
  - It moves to OOO buffer [8]
    - 8 = mod(24,16)
    - OOO buffer [i] is FIFO with 64 depth

Out-of-Order data arrival (3)

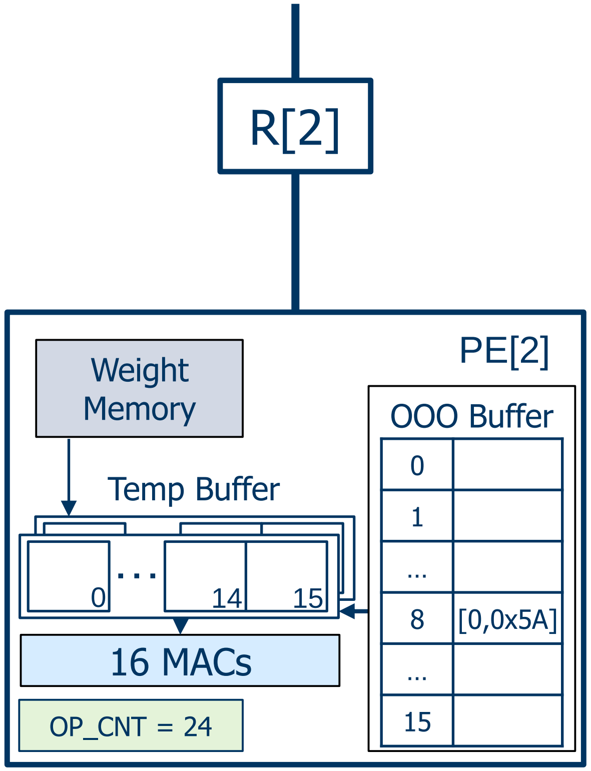


Packet structure

SRC	DST	DATA	MAC-ID	OP-ID
4bit	4bit	16bit	4bit	8bit

- OP-ID == OP_CNT == 23
  - This packet is for current operation
  - It moves to temporal buffer [14] directly
  - Temp buffer (length 16) is full
    - It will trigger 16 MACs operation

Out-of-Order data arrival (4)

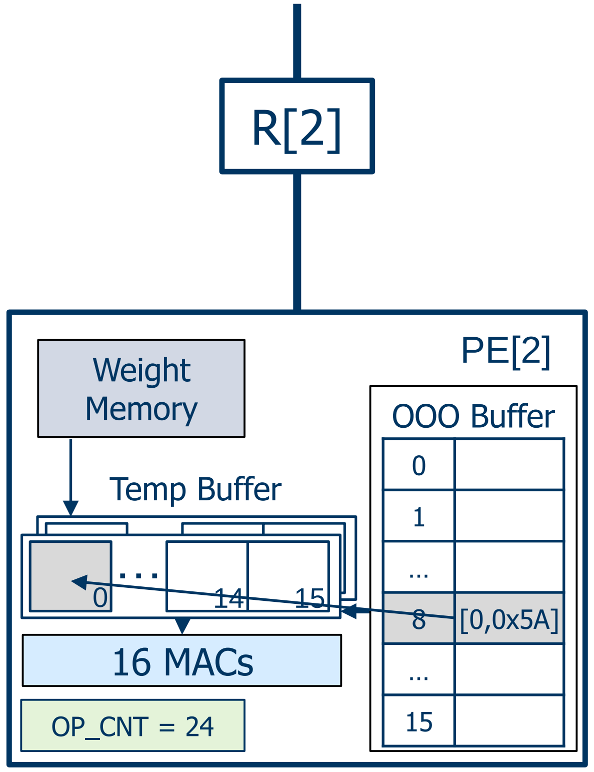


Packet structure

SRC	DST	DATA	MAC-ID	OP-ID
4bit	4bit	16bit	4bit	8bit

- Temp buffer trigger 16 MACs
- Increase OP_CNT
- Ready for 24th operation

Out-of-Order data arrival (5)



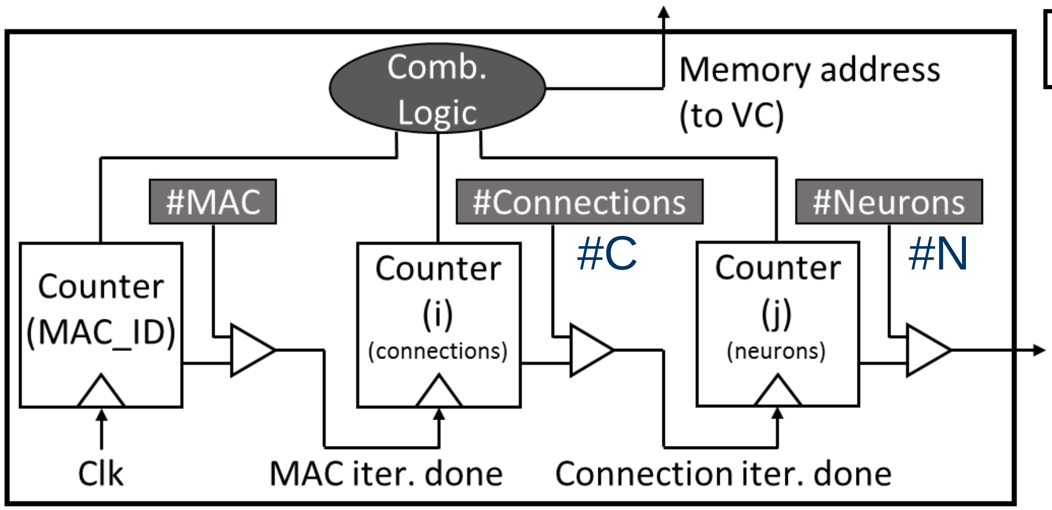
Packet structure

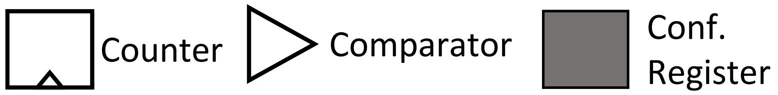
SRC	DST	DATA	MAC-ID	OP-ID
4bit	4bit	16bit	4bit	8bit

- Before capture new packet, check OOO buffer[8]
  - 8 = mod(24,16)
  - Full search OOO buffer [8] (64 depth)
  - Move [0;2;0x5A;0;24] to temp buffer [0]

Layerwise Programming

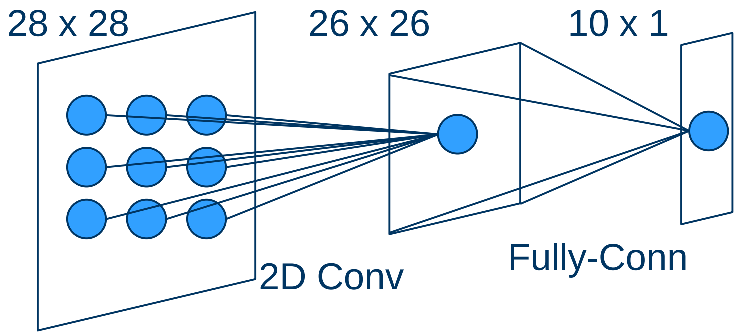
Address generator can be designed using **three nested counters**





#N: number of neurons in current layer
#C: number of connections from previous layer to the neuron in current layer

For **each layer**, host program the NeuroCube by **writing conf. registers**



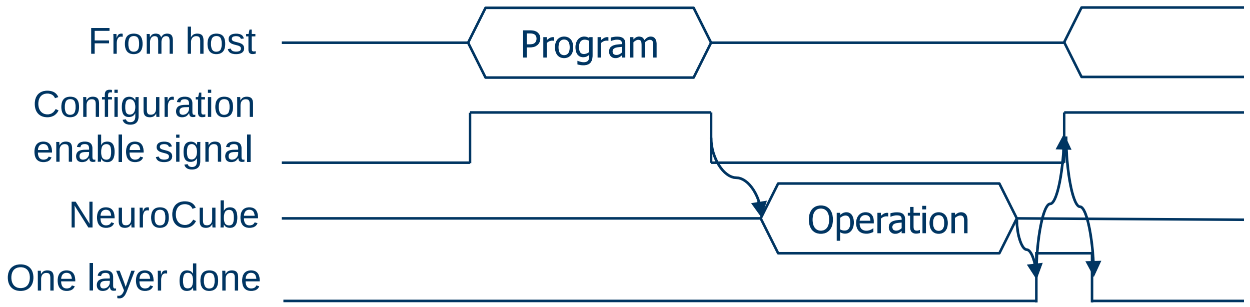
For 2D-Conv: #N: 26 x 26, #C: 3 x 3
For FC: #N: 10 x 1, #C: 676 x 1

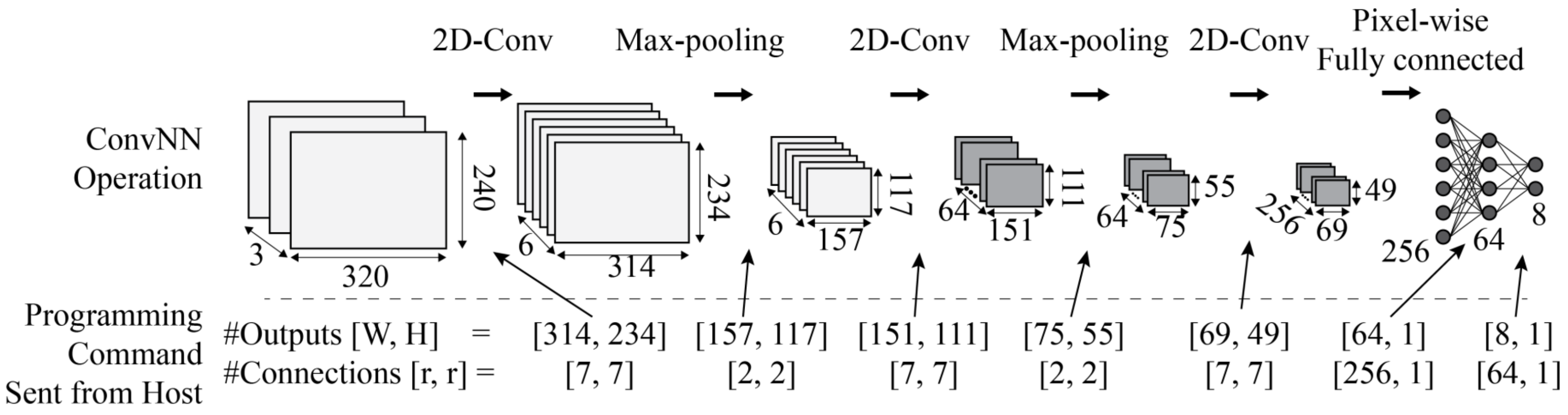
- Latency of writing configuration registers is negligible
- External interface is very rare

Layerwise Programming

Conv-NN for Scene labeling

- Handshaking to start programming or main operation



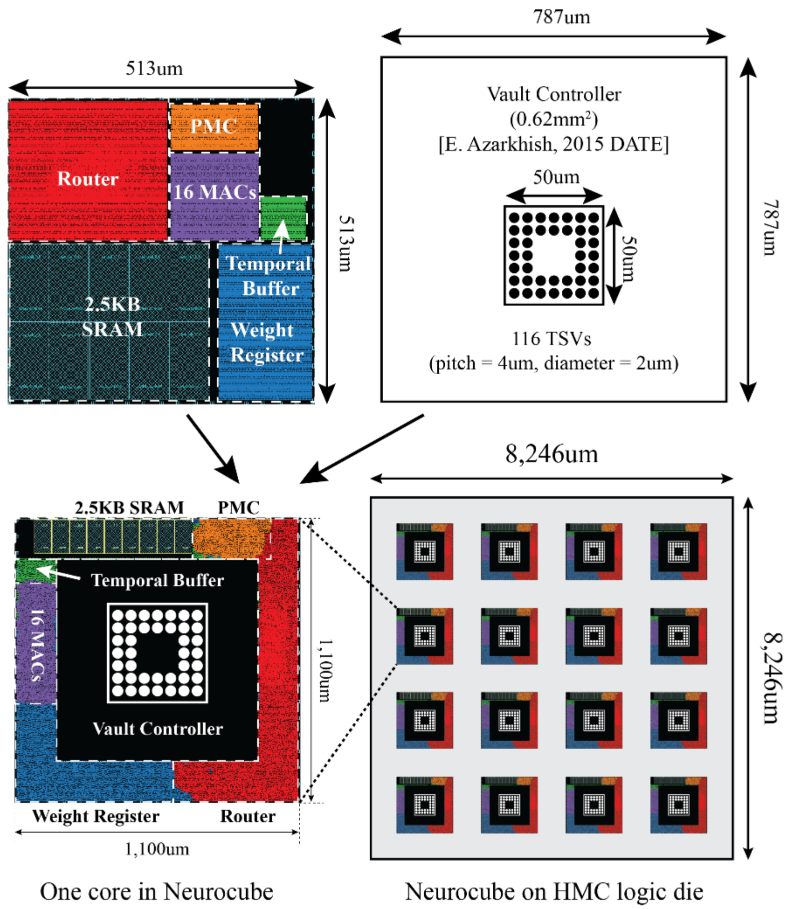


[S. Goud, `09 ICCV]

Simulation Results

Synthesis Result

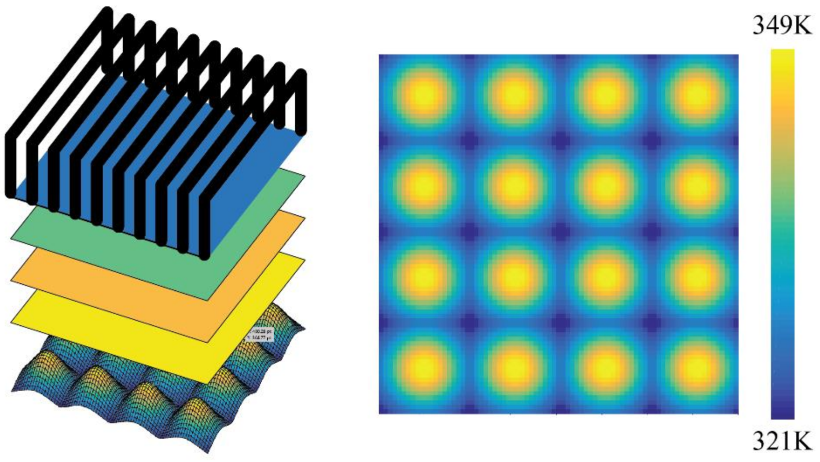
28nm CMOS process



Footprint of HMC 1.0: 68mm^2

15nm FinFet process

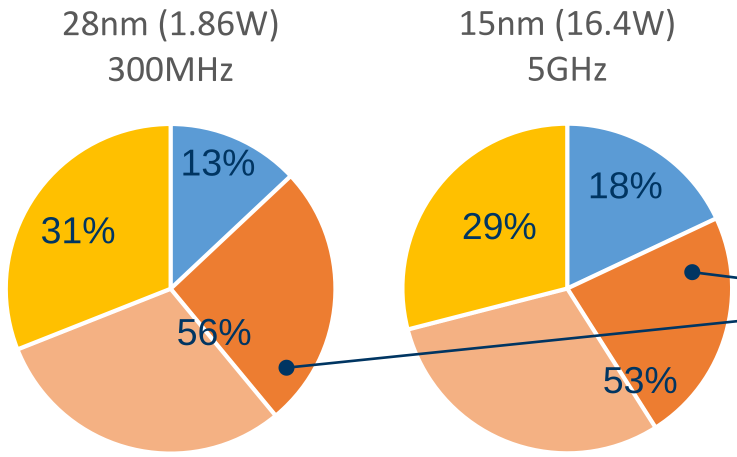
- To utilize HMC internal throughput fully, f_{PE} = 5GHz
- Thermal analysis performed under 5GHz operating
- Max. allowable temp. of HMC: 378K



[J. Jeddeloh `12 TVLSI] [HMC specification 1.0]

Hardware Power Breakdown

- Measured energy consumption [J. Jeddeloh, `12 VLSI]
  - 3.7 pj/bit for the DRAM layers
  - 6.78 pj/bit for the logic layer, (most power hungry = ext. interface SERDES)



🟦 NeuroCube computing layers

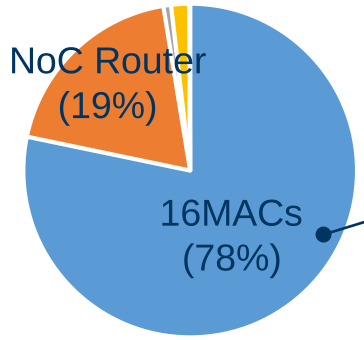
🟧 *HMC default logic die (without NeuroCube)**

🟨 *DRAM dies**

*[J. Jeddeloh, `12 VLSI]

Actual logic die power consumption is much lower since external data movements are very rare

- Single core power breakdown (15nm): 187mW



Actual MAC power consumption is much lower since its utilization ratio (activity) is very low : most of time MACs are idle until all operands are ready

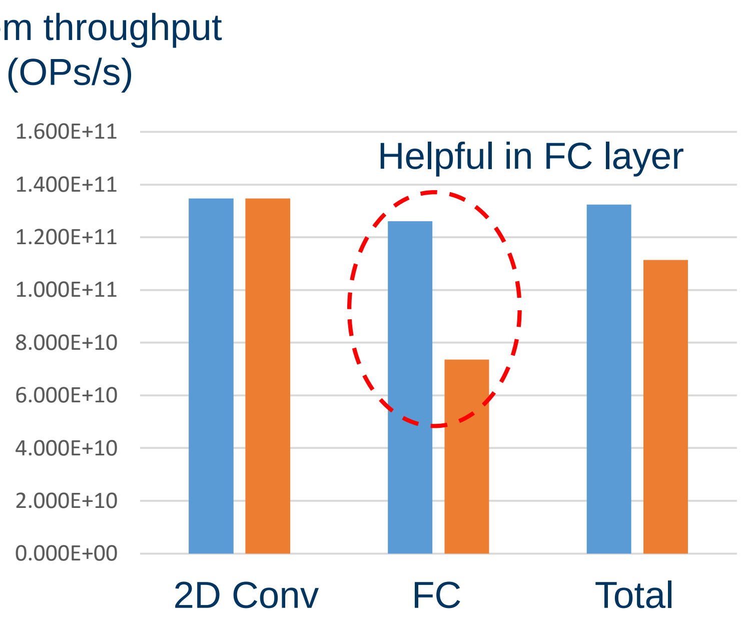
Performance Simulation: Inference

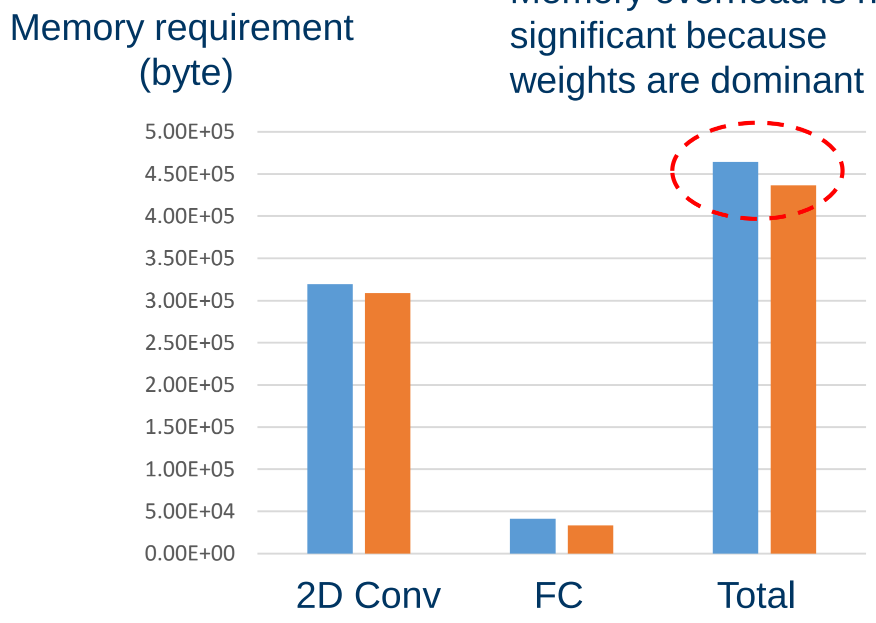
To see system throughput, cycle-level simulation is performed

- Inference: scene labeling
- Two operating modes:
  1. Duplication prev. layer to reduce NoC traffic
  2. No duplication to reduce memory overhead

15nm Finfet design

🟦 Duplicating prev. layer 🟧 No duplicating

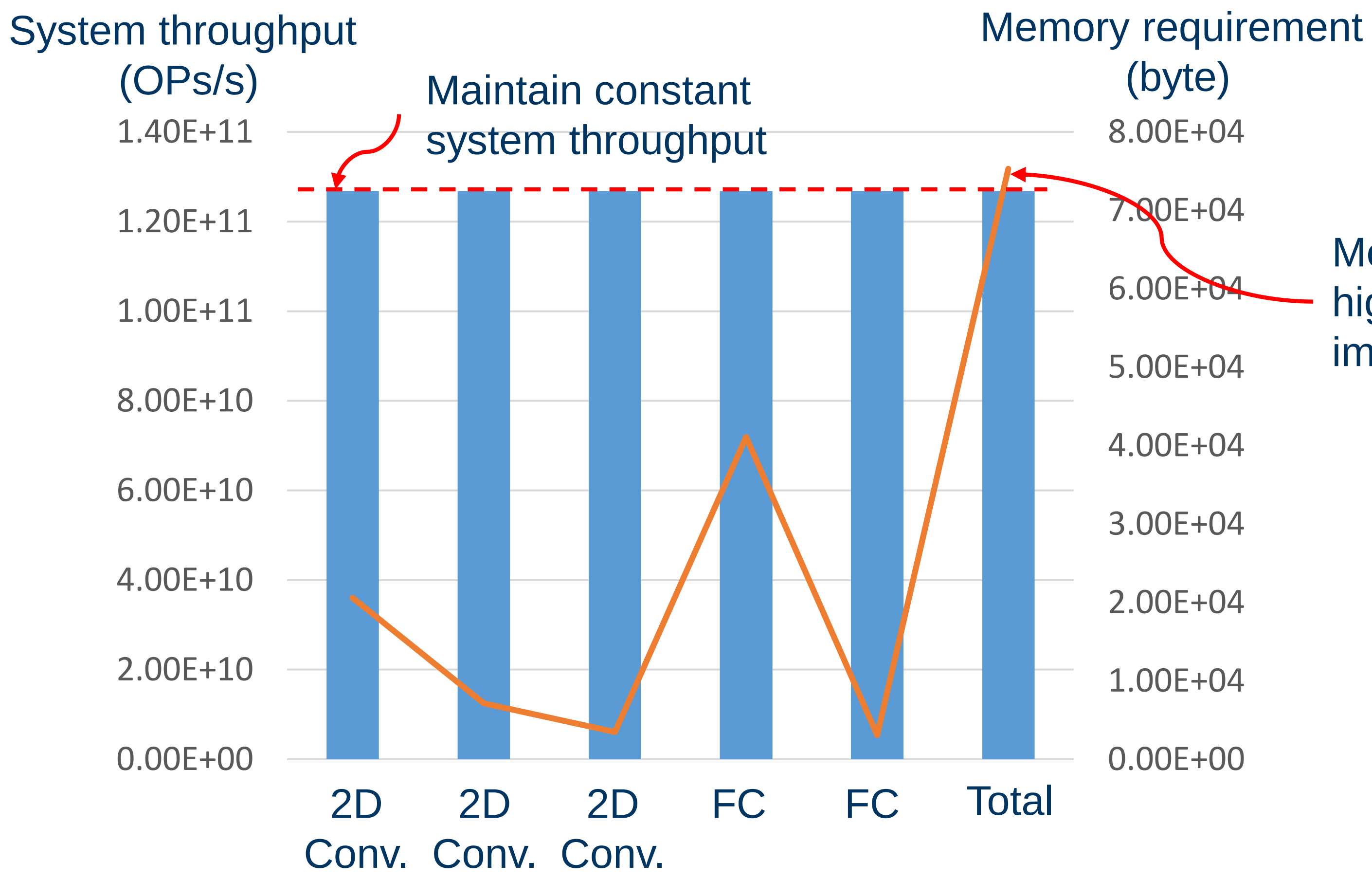




Memory overhead is not significant because weights are dominant

Performance Simulation: Training

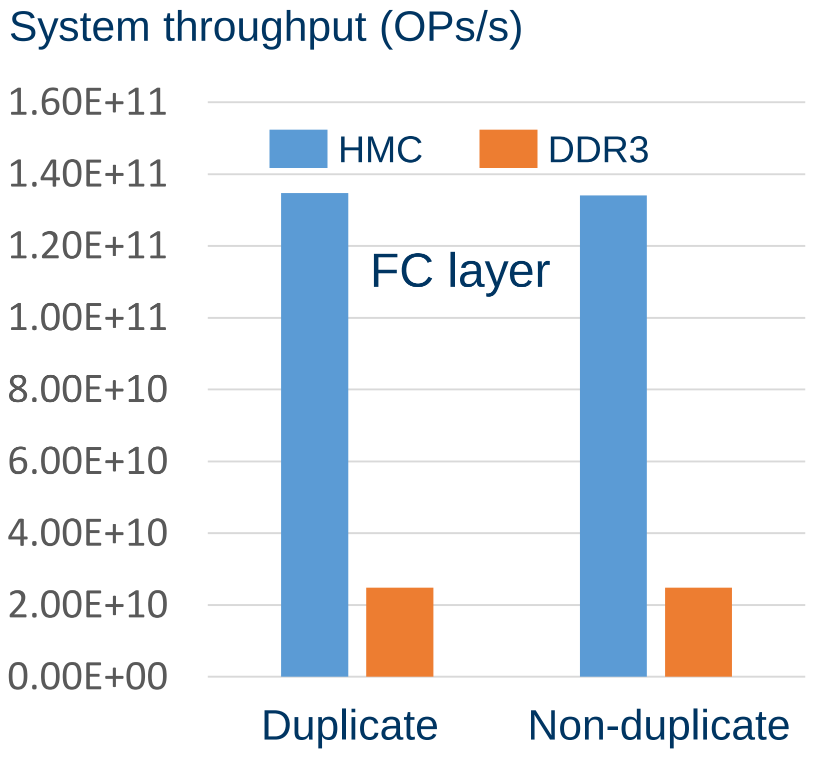
15nm Finfet design

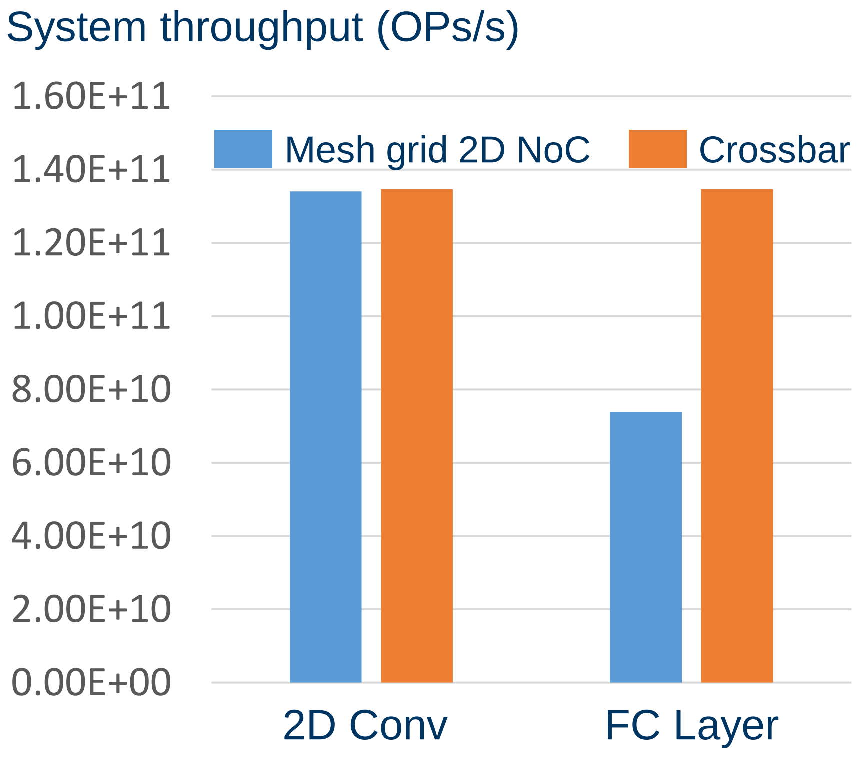


Memory requirement is high even for small image size (64 x 64)

Performance Simulation: DDR3/Crossbar

15nm Finfet design





Multiple channels in HMC improve system throughput

Crossbar improves system throughput for FC layer

Related Works

- Most of prev. work focused on specific NN
  - Shows high throughput based on optimized design (ASIC/FPGA)
  - Not programmable (not scalable)
- Programmable + Scalable design
  - General purposed architecture: mobile CPU or GPU
  - Integrated systems with external DRAM

	[L. Cabigelli `15 DAC]		NeuroCube	
Platform	Tegra K1	GTX 780	28nm	15nm
Bit precision ctrl.	N/A	N/A	16bit	16bit
Throughput (GOPs/s)	76	1781	7.95	132.4
Computing Power (W)	11	206.8	0.249	3.41
Efficiency (GOPs/s/W)	6.91	8.61	**31.92**	**38.82**
Inference/training	inference	inference	both	both

Conclusion

- NeuroCube: Neuro-inspired architecture as PIM in HMC
  - Utilize high memory bandwidth
  - Integrated with high density memory
  - Meet thermal/area constraints

- Programmable architecture to cover diff. NN types
  - Programming memory access pattern
  - Simple memory address generator (PNG) is embedded in memory
  - Memory centric neural computing (MCNC) scheme

- System performance is simulated
  - Network-on-chip traffic is next bottleneck
  - Optimized NoC design, data re-usage should be studied

Thank you