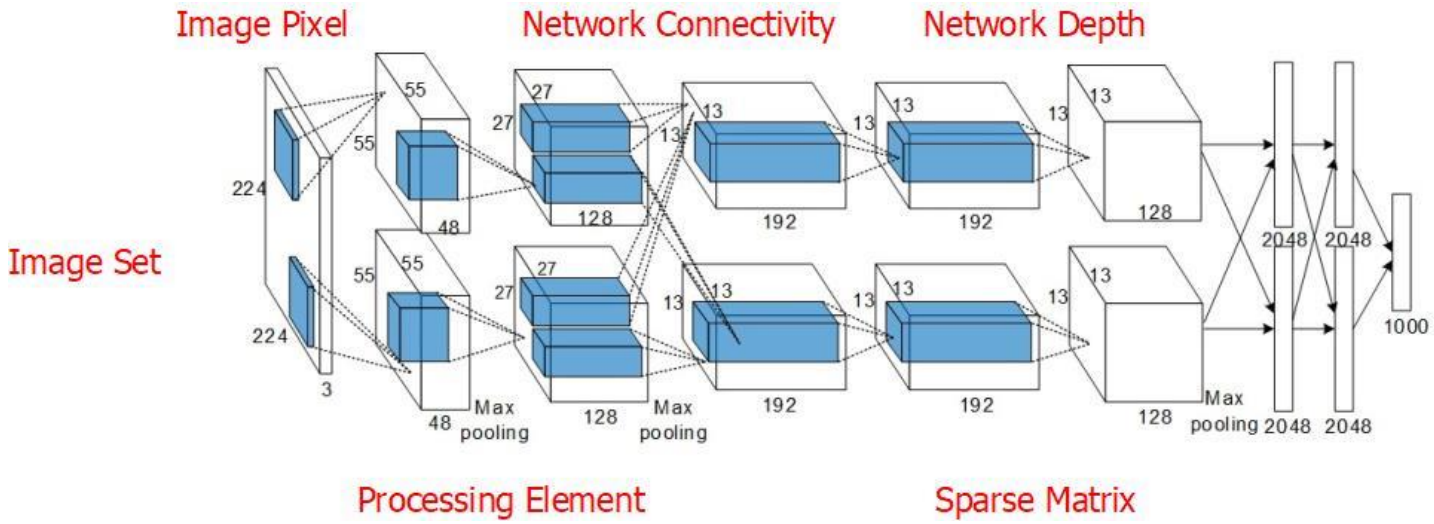
Machine Learning

Hardware Design

Dr. Albert Liu

Kneron Inc

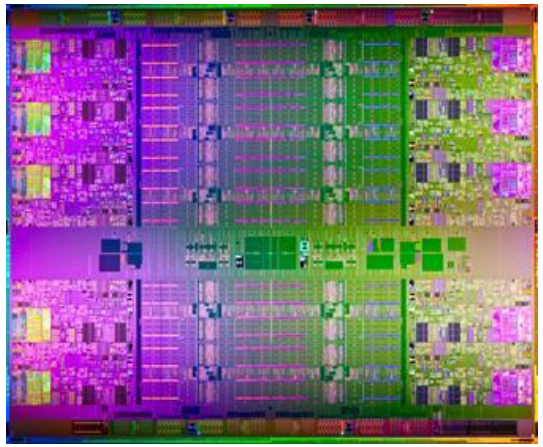
Convolutional Neural Network



- Hardware Challenges
  - Massive parallel processing
  - Reconfigurable network
  - Network pruning
  - Memory bandwidth
  - Processing element
  - Sparse matrix support

Krizhevsky et al., ImageNet Classification with Deep Convolutional Neural Networks, NIPS, 2012.

Intel Scalable Xeon Processor (CPU)

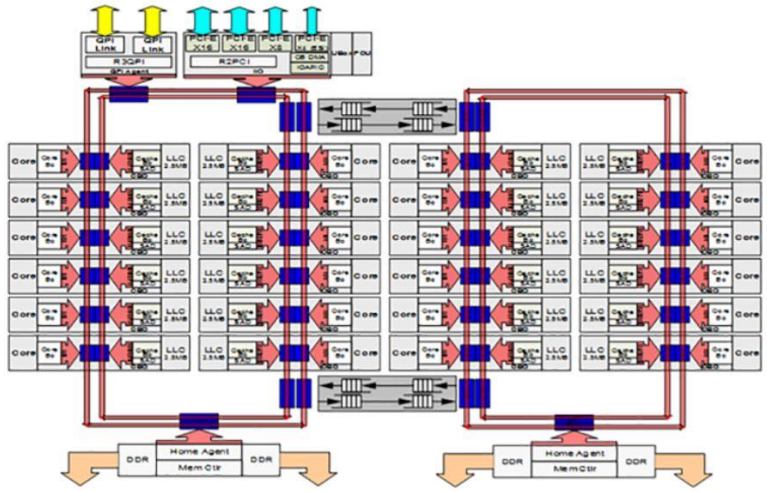


Intel Xeon Scalable Processor

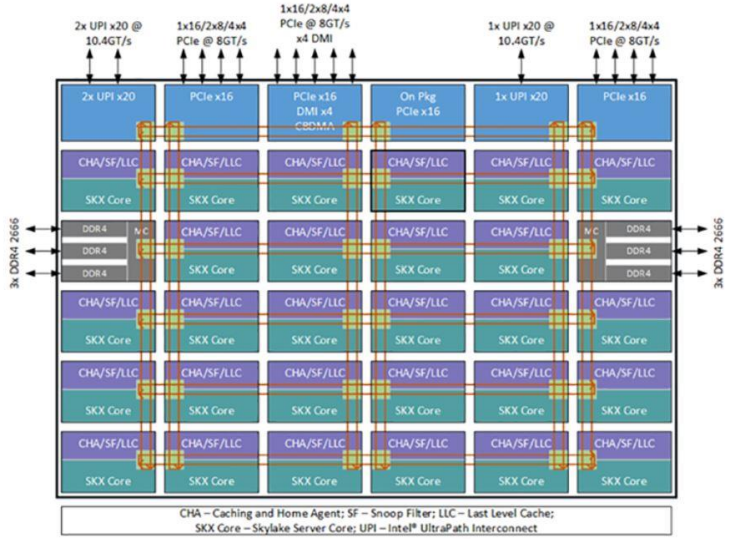
- Support 28 physical cores per sockets at 2.5GHz and up to 3.8 GHz at turbo mode
- Six memory channels support up to 1.5Tb of DDR4 memory
- 1Mb private cache and 38.5Mb shared cache
- Operate at 3.57 TFLOPS (FP32) up to 5.18 TOPS (INT8) per socket and max 41.44 max TOPS (INT8)
- Train ResNet-50 in 31 minutes only

T. Allred, A Survey Paper Comparing modern CPU, GPU & TPU Hardware in Relation to Neural Network Training and Interference, Nov 2018.

Intel Scalable Xeon Processor (CPU)



Intel Quick Path Interconnect (QPI) Ring Architecture

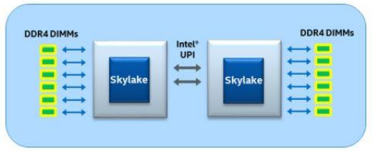


Intel Ultra Path Interconnect (UPI) Mesh Architecture

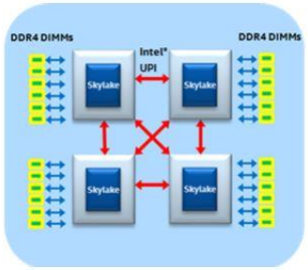
- Upgrade Intel Quick Path Interconnect (QPI) ring architecture to Intel Ultra Path Interconnect (UPI) mesh architecture to resolve the latency and bandwidth issues
- Integrate Caching and Home Agent together to form new Combined Home Agent (CHA) to remove the memory access bottleneck

D. Mulnix, Intel® Xeon® Processor Scalable Family Technical Overview, Sept 2017.

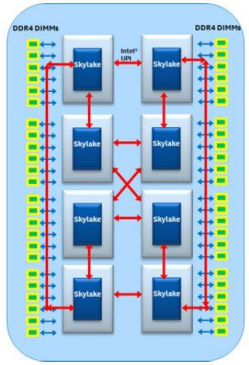
Intel Scalable Xeon Processor (CPU)



Two Socket Configuration



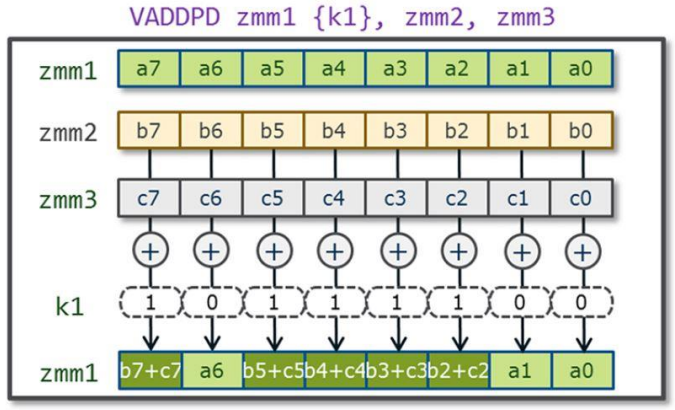
Four Socket Configuration



Eight Socket Configuration

- Every core has its CHA that provides the routing information to reach destination address
- Intel QPI allows one core to communicate with the other one only and it occupies the whole path during the data transfer resulted in high latency.
- Intel UPI allows the data transverse from one core to another through the shortest path using either vertical or horizontal paths. Multiple data can be transfer at the same time.
- New approach provides 10.4 GT/s transfer speed with new packetization format

Intel Scalable Xeon Processor (CPU)

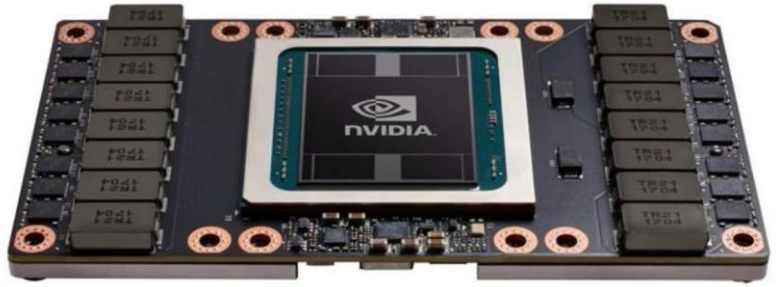


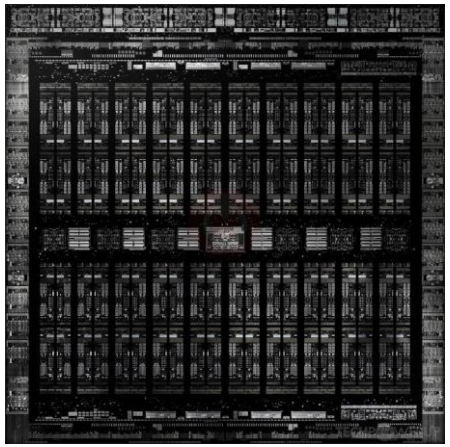
AVX 512 Instruction Example

- Develop new Intel Advanced Vector Extension 512 (AVX-512)
- Support 2 floating point Fused Multiply Add (FMA) operation
- Enhance doubleword/quadword integer and floating point vectorization operation on 512bit vectors (i.e. 16 32bit or 8 64bit elements), it significantly reduces additional load/store operations to improve the overall runtime.
- Utilize the computational resource using Math Kernel Library for Deep Neural Network (MKL-DNN) that applies the Compressed Sparse Row (CSR)* to eliminate the zero value computations. It optimize for Caffe, TensorFlow, Apache MXNet, PyTorch, CNTK, etc.
- Improve the training performance by 100x and inference by 39X

* Compressed Sparse Row (CSR) will be discussed more detail in Microsoft DNN processor

Nvidia Tesla Graphics Processor (GPU)





Nvidia GV100 Graphics Processing Unit

- Tesla V100 accelerator with new Volta GV100 GPU architecture
- Boot up arithmetic operation to 15 TFLOPS (FP32)
- Speed up deep learning training by 12x and inference by 6x
- Support 6 high speed NVLink2 to deliver 300 Gb/s bandwidth
- Apply HBM2 (High Bandwidth Memory) to provide 900 Gb/s bandwidth through 3D-IC interposer implementation
- Total power: 300W

Nvidia Tesla V100 GPU Architecture, Nvidia, Aug 2017

Nvidia Tesla Graphics Processor (GPU)

Tesla Product	Tesla K40	Tesla M40	Tesla P100	Tesla V100
GPU	GK180 (Kepler)	GM200 (Maxwell)	GP100 (Pascal)	GV100 (Volta)
SMs	15	24	56	80
TPCs	15	24	28	40
FP32 Cores / SM	192	128	64	64
FP32 Cores / GPU	2880	3072	3584	5120
FP64 Cores / SM	64	4	32	32
FP64 Cores / GPU	960	96	1792	2560
Tensor Cores / SM	NA	NA	NA	8
Tensor Cores / GPU	NA	NA	NA	640
GPU Boost Clock	810/875 MHz	1114 MHz	1480 MHz	1530 MHz
Peak FP32 TFLOPS[1]	5	6.8	10.6	15.7
Peak FP64 TFLOPS[1]	1.7	.21	5.3	7.8
Peak Tensor TFLOPS[1]	NA	NA	NA	125
Texture Units	240	192	224	320
Memory Interface	384-bit GDDR5	384-bit GDDR5	4096-bit HBM2	4096-bit HBM2
Memory Size	Up to 12 GB	Up to 24 GB	16 GB	16 GB
L2 Cache Size	1536 KB	3072 KB	4096 KB	6144 KB
Shared Memory Size / SM	16 KB/32 KB/48 KB	96 KB	64 KB	Configurable up to 96 KB
Register File Size / SM	256 KB	256 KB	256 KB	256KB
Register File Size / GPU	3840 KB	6144 KB	14336 KB	20480 KB
TDP	235 Watts	250 Watts	300 Watts	300 Watts
Transistors	7.1 billion	8 billion	15.3 billion	21.1 billion
GPU Die Size	551 mm²	601 mm²	610 mm²	815 mm²
Manufacturing Process	28 nm	28 nm	16 nm FinFET+	12 nm FFN

[1] Peak TFLOPS rates are based on GPU Boost Clock

Nvidia Tesla GPU Comparison

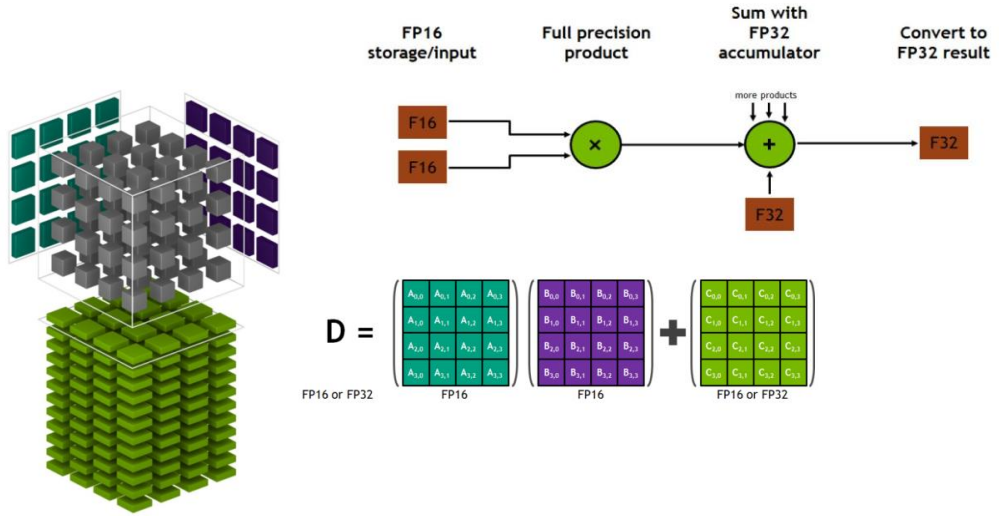
Nvidia Tesla Graphics Processor (GPU)

Volta GV100 Architecture with 84 SM

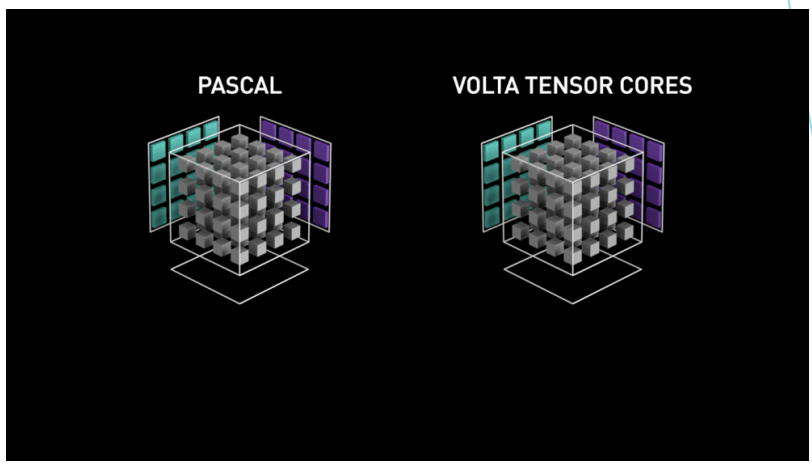
Volta GV100 Stream Multiprocessor

- Six GPU Processing Cluster (GPC)
  - 7 Texture Processing Cluster (TPC) and 14 Stream Multiprocessor (SM)
- 84 Volta Stream Multiprocessor (SM)
  - 64 FP32 core, 64 INT32 core, 32 FP64 core, 8 Tensor Core, 4 Texture Unit
- Eight 512bit memory controller

Nvidia Tesla Graphics Processor (GPU)



Tensor Core 4x4x4 Matrix Operation



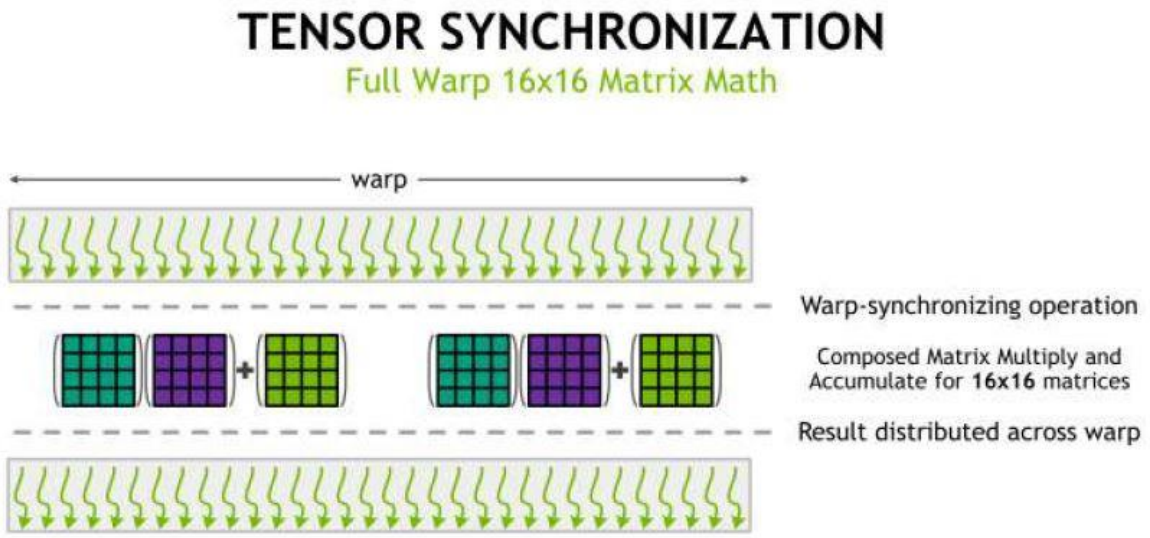
Pascal Core vs Tensor Core

- Tensor Core perform 4x4x4 matrix Multiply and Accumulate operation

  D = A x B + C

- Different from Pascal core, it multiplies the matrix row-by-row, Tensor core multiplies two matrix at the same time using Simultaneous Multithreading (SMT) approach
- It speeds up overall arithmetic operation: FP16(8X), INT8(16X) and INT4(32X)
- Perform input FP16 multiplication result in FP32 full precision product then accumulate using FP32 addition as well as other intermediate product

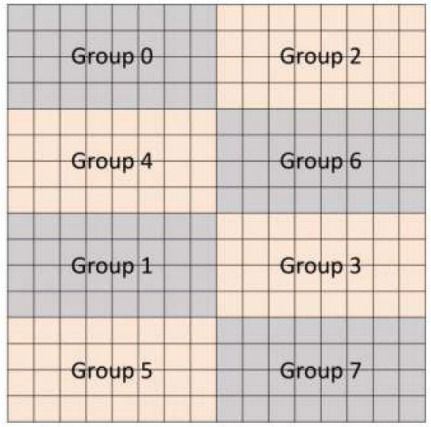
Nvidia Tesla Graphics Processor (GPU)



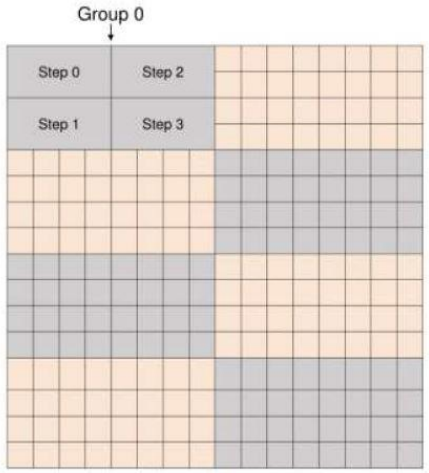
- Simultaneous Multithreading (SMT) first divides the matrix into multiple group, then perform the matrix-multiply in particular patchwork pattern for all subset (fragments) in parallel, each subset are not interacted with others
- After the matrix-multiply, it regroups the subset into same group to obtain the results
- With multiple tensor core, it speeds up the overall matrix operation
- The major drawback is the peak power increase

The Nvidia Titan V Deep Learning Deep Dive: It's All About The Tensor Cores, AnandTech, Jul 2018.

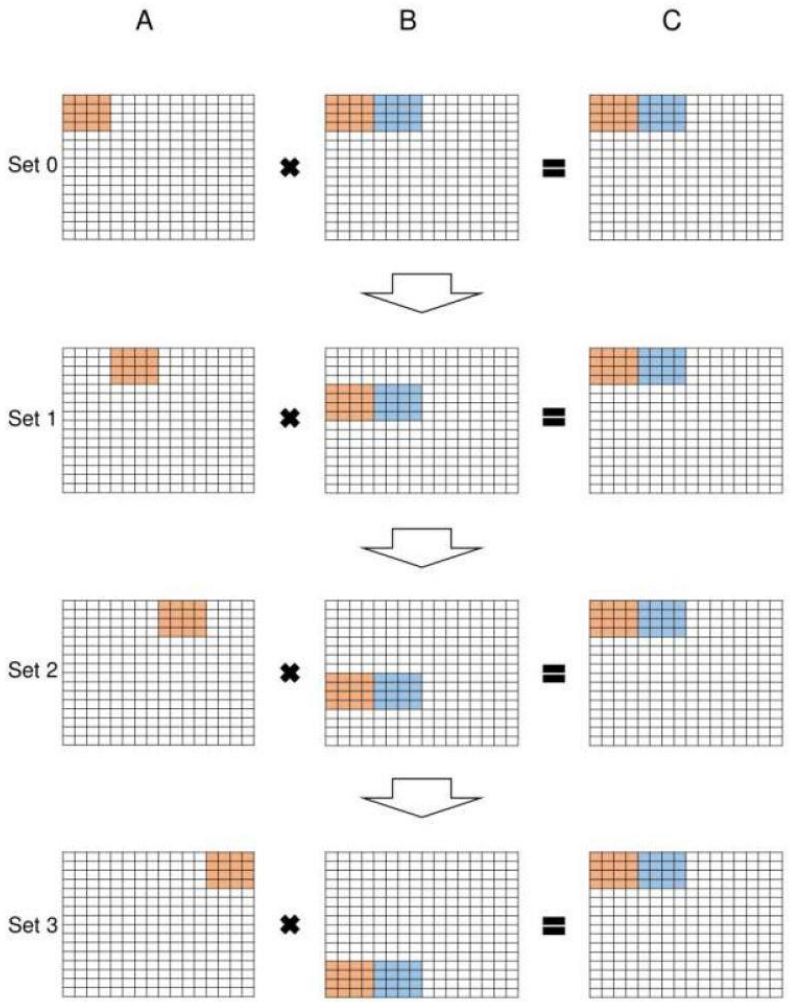
Nvidia Tesla Graphics Processor (GPU)



Multithreading Mapping



Multithreading Re-grouping



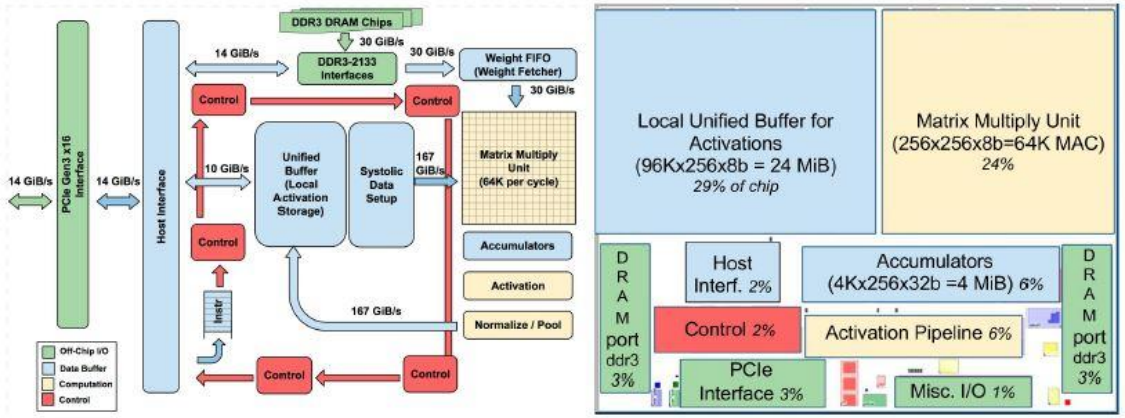
Multithreading Computation

Google Tensor Processing Unit (TPU)

Model	Die										Benchmarked Servers				
	mm^2	nm	MHz	TDP	Measured		TOPS/s		GB/s	On-Chip Memory	Dies	DRAM Size	TDP	Measured	
					Idle	Busy	8b	FP						Idle	Busy
Haswell E5-2699 v3	662	22	2300	145W	41W	145W	2.6	1.3	51	51 MiB	2	256 GiB	504W	159W	455W
NVIDIA K80 (2 dies/card)	561	28	560	150W	25W	98W	--	2.8	160	8 MiB	8	256 GiB (host) + 12 GiB x 8	1838W	357W	991W
TPU	NA*	28	700	75W	28W	40W	92	--	34	28 MiB	4	256 GiB (host) + 8 GiB x 4	861W	290W	384W

- Tensor Processing Unit has 256x256 MAC for 8~64bit Multiplication
- Employ 8Gb DRAM for Weight FIFO with Gen3 x16 bus
- 15x~30X faster than Nvidia K80 GPU
- 35X MAC and 3.5X memory more than Nvidia K80 GPU
- 180 TFLOPS via four 45 TFLOPS chip configuration

Google Tensor Processing Unit (TPU)

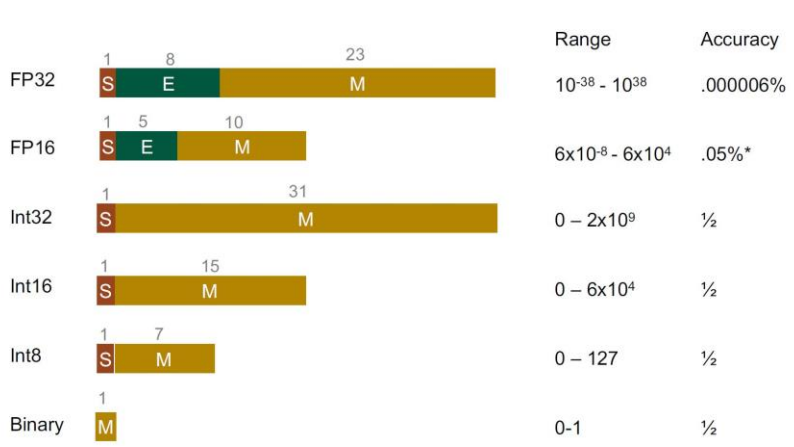


Tensor Processing Unit Architecture

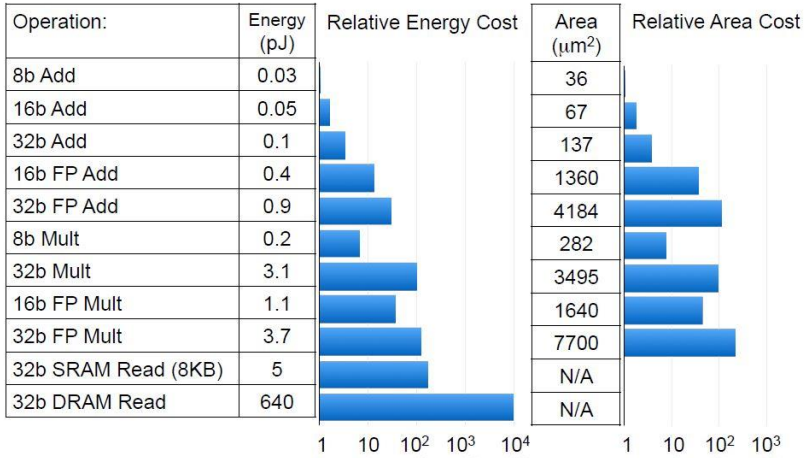
- Tensor Processing Unit (TPU) targets for TensorFlow for data center using FPGA approach
- To convert FP32 floating point to INT8 integer to speed up overall operation as well as power reduction
- Architecture
  - Systolic Array Matrix Multiplier Unit (MMU)
  - 24Mb Unified Buffer (UB)
  - 4Mb 32bit Accumulator (ACC)
  - 16bit Activation Unit (AU)

Jouppi et al., In-Datacenter Performance Analysis of a Tensor Unit, ISCA, 2017.

Google Tensor Processing Unit (TPU)



Number Format

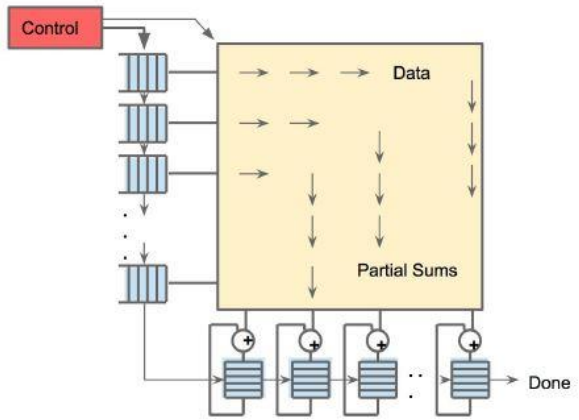


Cost of Operation

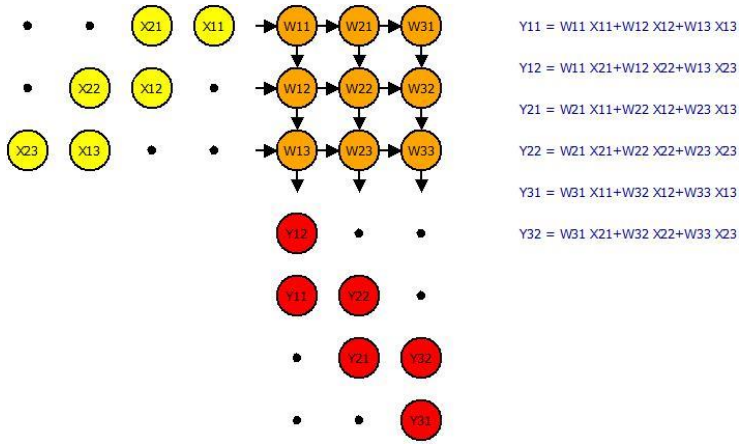
- Quantize the input data from FP32 to INT8 with the advantage of speed, area and power.
- Floating point hardware involves additional exponent alignment, normalization, rounding and long carrier propagation
- The major drawback of quantization is the integer truncation errors and numerical instability, it is required Deep Learning to avoid numerical issues:
  - Max pooling
  - Normalization

B. Dally, Hardware for Deep Learning, Stanford and Nvidia, Jun 2016.

Google Tensor Processing Unit (TPU)



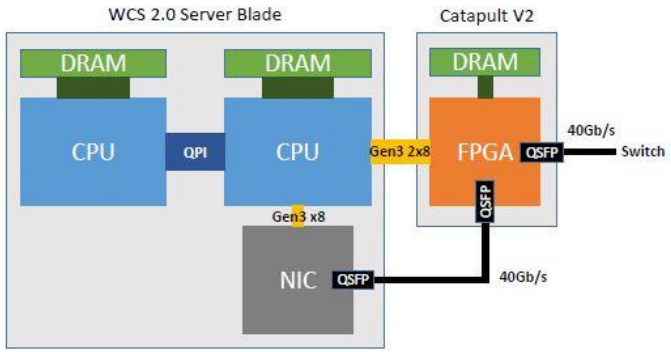
TPU Multiply-Accumulate Unit (MAC)

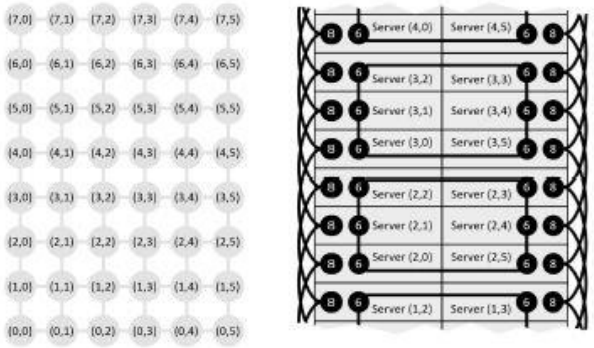


Systolic Array – Matrix Multiplication

- Systolic array is a highly pipeline computational network with less latency for Single Instruction Multiple Data (SIMD) operation.
- It is analogy to how blood rhythmically flows through a biological heart as the data flows from memory in a rhythmic fashion passing through processing element (PE)
- All the data is skewed and synchronized by global clock, then feed into the systolic array for computation. The results are available in pipeline fashion with high throughput rate. It is suitable for matrix multiplication.

Microsoft Catapult DNN Processor

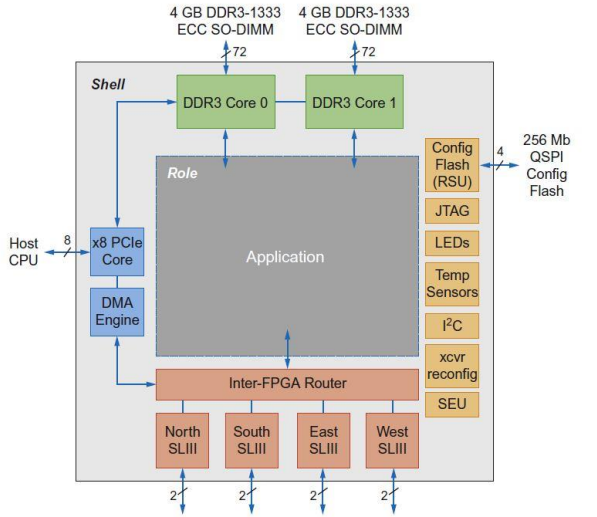


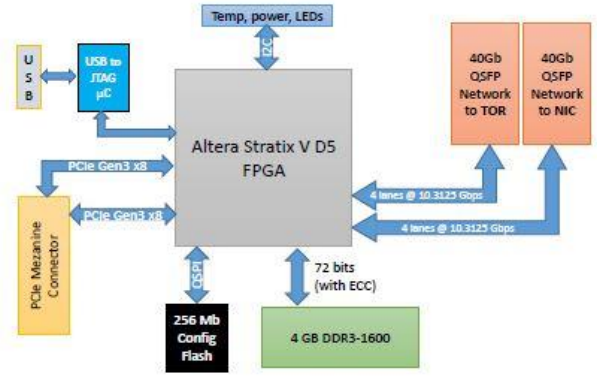


- Support Brainwave datacenter application using FPGA DDN processor (Catapult fabric)
  - Local computational accelerator
  - Network/storage accelerator
  - Remote computational accelerator
- 8Gb DRAM supports for local computation
- 48 FPGA are organized into two half racks called pods and connected using 6x8 tours network topology

A. Putnam et al., A Reconfigurable Fabric for Accelerating Large-Scale Data Center Services, ACM, 2014.

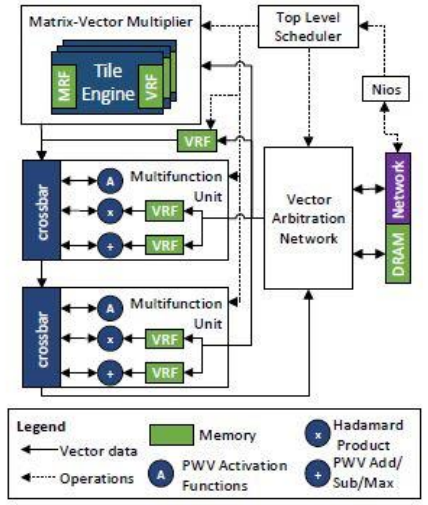
Microsoft Catapult DNN Processor

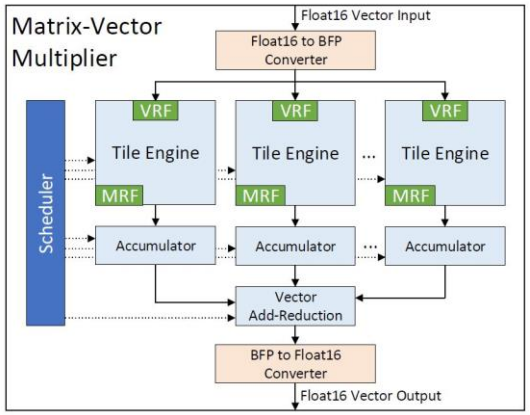


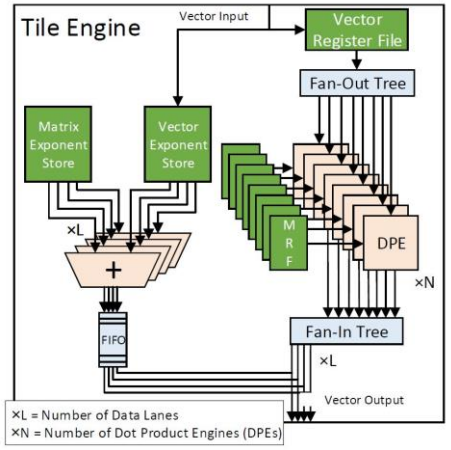


- Catapult fabric is a synthesized using “soft processor” approach, it can reconfigured using low-level software library without RTL recompilation for desired applications.
- FPGA is divided into two partitions: shell and role. The shell is reusable portion of programmable logic common to all logic and the role is configured to perform different applications
- Two DRAM controller supports independent memory access, four high speed serial link connected with neighboring FPGA
- PCIe core supports CPU DMA
- Single-event upset (SEU) logic to reduce the network errors caused by soft errors

Microsoft Catapult DNN Processor







- Catapult fabric is based on Single-Threaded SIMD ISA architecture with Matrix-Vector-Multiplier (MVM) to perform matrix-vector or vector-vector operations
- Vector arbitration network manages data transfer among pipeline register files, DRAM and network I/O queues
- Top level scheduler controls the function unit operation and vector arbitration network based on input instruction chains
- MVM employs dot product engine (DPE), the matrix register file (MRF) and vector register file (VRF) for matrix computation

Fowers et al., A Configurable Cloud-Scale DNN Processor for Real Time AI, ISCA 18

Microsoft Catapult DNN Processor

	0	1	2	3	4	5	6	7
0	A			B				
1				C				
2					D	E		
3		F				G		H
4			I				J	K
5				L				M
6		N				O		
7							P	

(a)

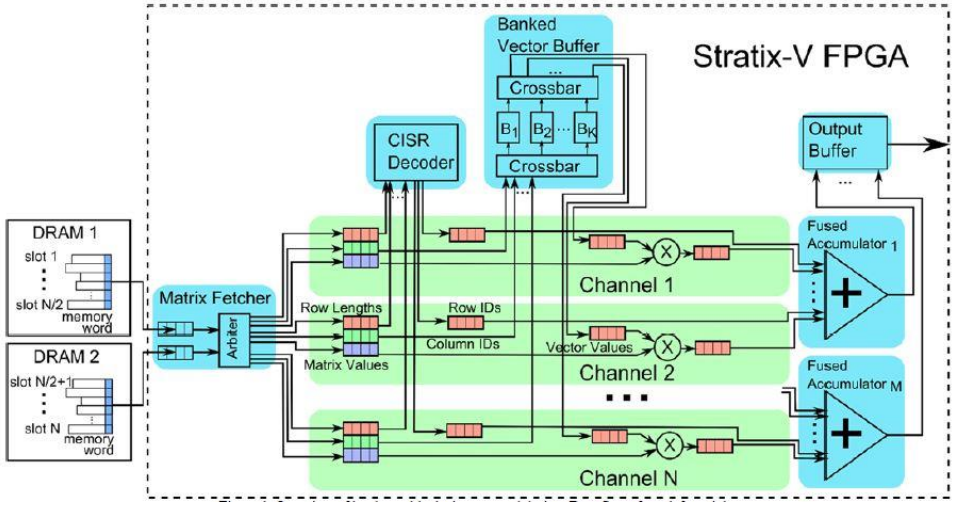
CSR	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VALUES	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
COLUMN INDICES	0	3	3	4	5	1	5	7	2	6	7	3	7	1	5	6
ROW POINTERS	0	2	3	5	8	11	13	15	16							

(b)

memory width

CISR	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VALUES	A	C	D	F	B	I	E	G	L	J	N	H	M	K	O	P
COLUMN INDICES	0	3	4	1	3	2	5	5	3	6	1	7	7	7	5	6
ROW LENGTHS	2	1	2	3	2	3	2	1								
SLOT NUMBER	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4

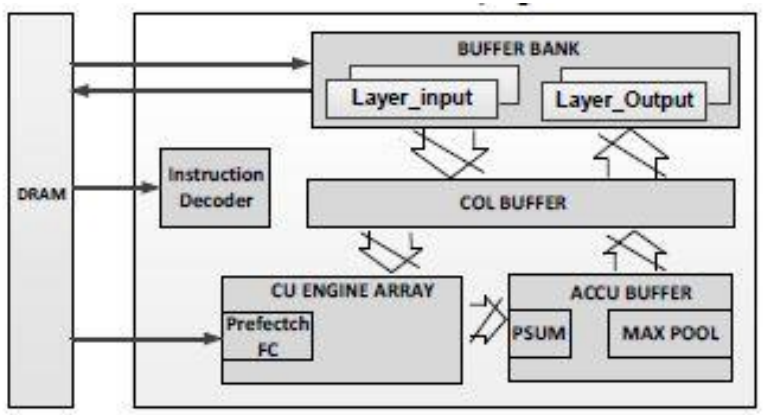
(c)

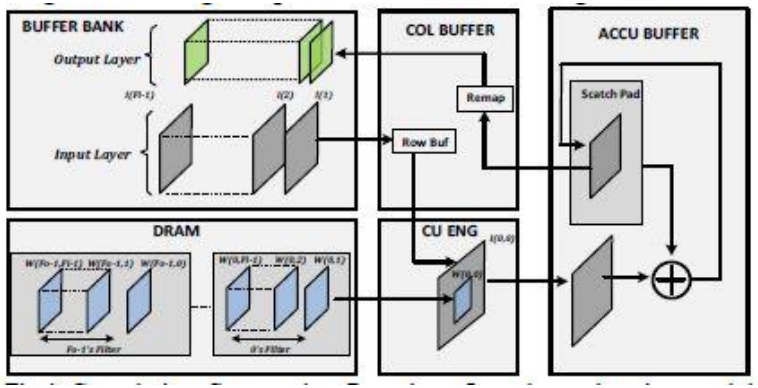


- Intel Xeon processor applies Compressed Sparse Row (CSR) approach for sparse matrix computation
- Microsoft Catapult fabric is based on new sparse matrix-vector multiplication scheme – Condensed Interleaved Sparse Representation (CISR) encoding approach to speed up matrix operation
- CISR encoding consists of three arrays, the first encodes non-zero values, the second encodes their corresponding columns and the last one stores the length of each row and breaks the data dependences between rows and simplifies the parallelization
- This approach enables simultaneous multiply-accumulate operations on multiple rows of matrix without the complex schedulers and load-balancers

Fowers et al., A High Memory Bandwidth FPGA Accelerator for Sparse Matrix-Vector Multiplication, FCCM, 2014.

UCLA DCNN Accelerator

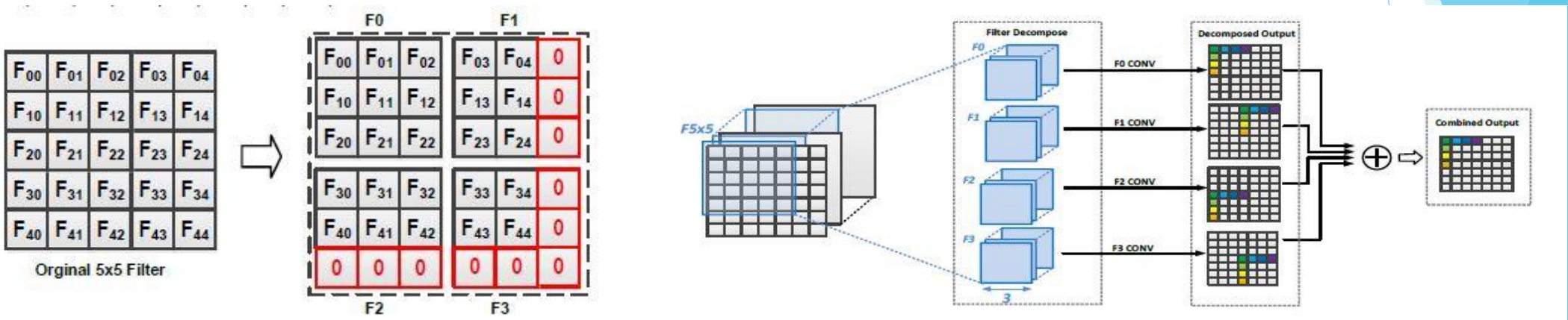




- Implement Deep Convolution Neural Network Accelerator (DCNN) using 65nm process with 5mm^2 die area, it achieve 152 GOPS peak throughput and 434 GOPS/W energy efficiency at 350mW
- Apply streaming data flow to minimize data access and achieve high energy efficiency
- Enable parallel computation for multiple output features without input memory bandwidth increment using interleaving architecture
- Decompose large-size filter computation to small size one to achieve high reconfigurability without additional hardware penalty
- Additional pooling functional unit to reduce main convolution unit (CU) engine load

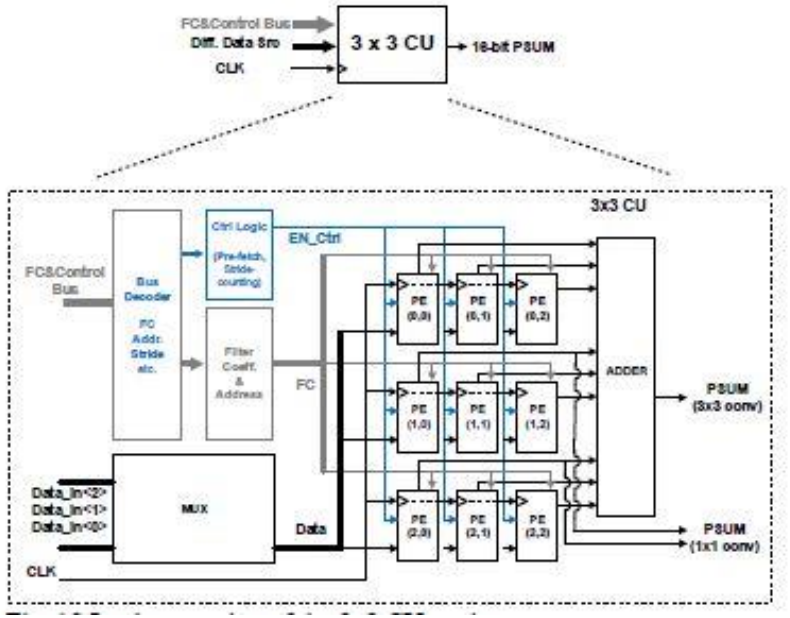
Du et al., A Reconfigurable Streaming Deep Convolutional Neural Network Accelerator for Internet of Things, IEEE, 2018

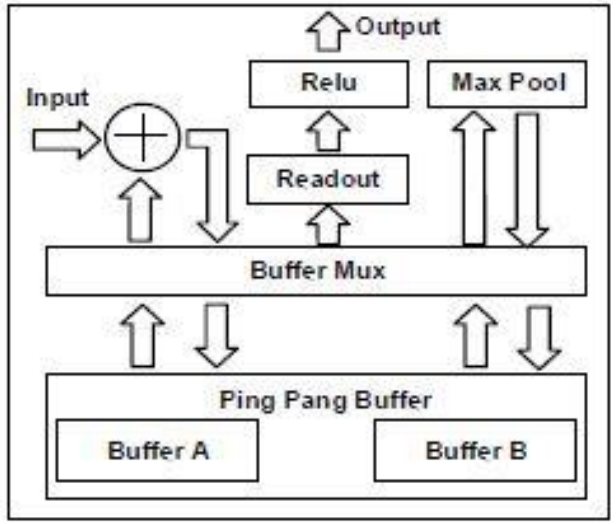
UCLA DCNN Accelerator



- A filter decomposition algorithm is used to compute any large kernel-sized (> 3x3) convolution through only 3x3 sized CU to minimize hardware usage
- Additional zero padding weights will be added to extend the original kernel boundary to be multiple of three

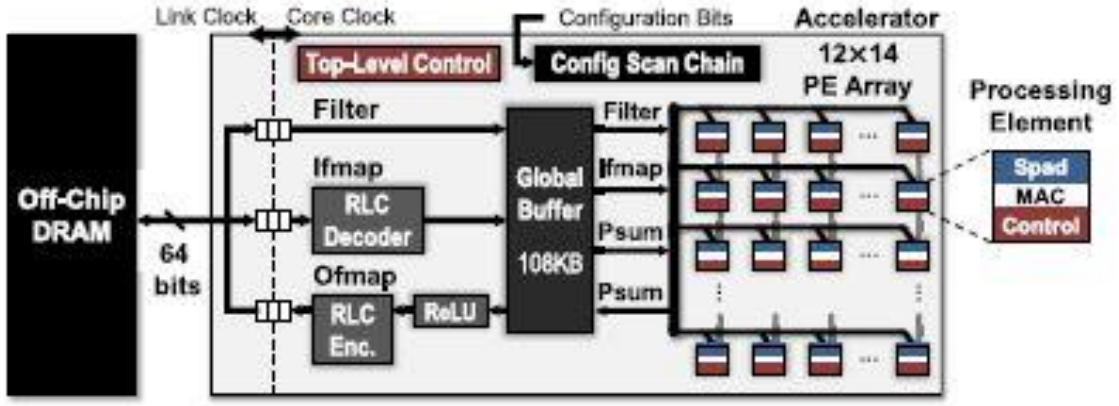
UCLA DCNN Accelerator





- 3x3 convolution engine (CU) consists of 9 processing elements (PE) to perform matrix multiplication, an adder is used to combine the output. Then, the partial sum is stored in Accumulator (ACC) Buffer
- Accumulate (ACC) Buffer includes a Ping-Pang buffer formed by Buffer A and Buffer B. Two buffers are switched back and forth between accumulator and Readout/Max pool blocks to enable parallel processing

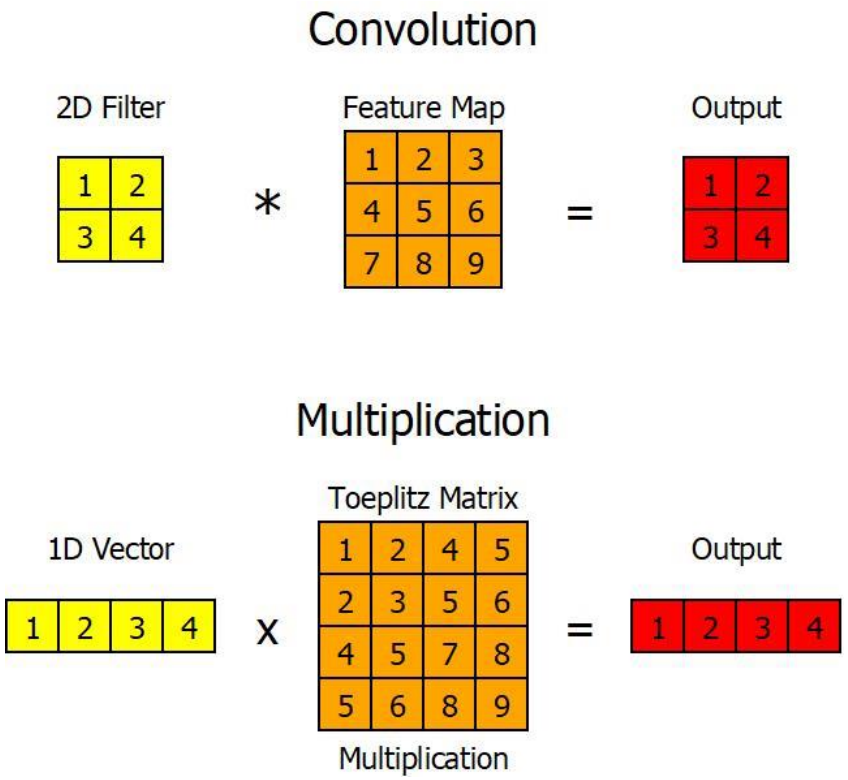
MIT Eyeriss DCNN Processor



- A spatial architecture with 168 processing elements (PE) and 4 level memory hierarchy minimizes the data access
- Row Stationary (RS) dataflow approach reconfigures the spatial architecture to map CNN shape for energy optimization
- Network-on-Chip (NoC) architecture uses both multicast and point-to-point single cycle data delivery to support RS dataflow
- Run-length compression (RLC) and PE data gating exploit the zero data statistics to improve energy efficiency

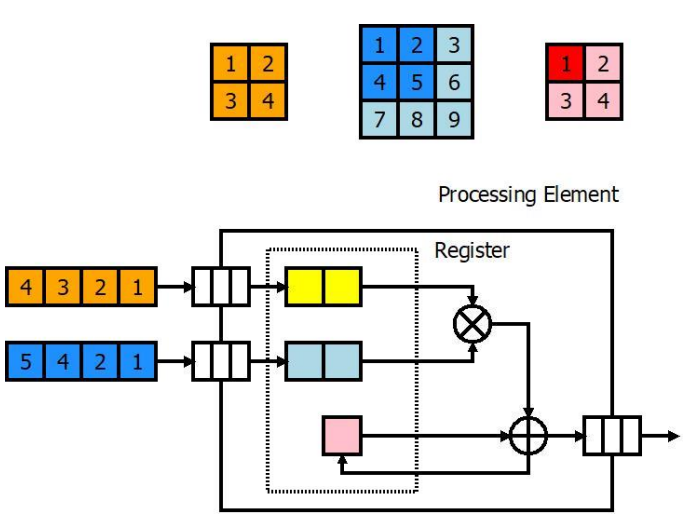
Chen et al., An Energy-Efficient Reconfigurable Accelerator for Deep Learning Convolution Networks, JSSC, 2016.
Emer et al., Hardware Architectures for Deep Neural Networks, ISCA Tutorial, 2017

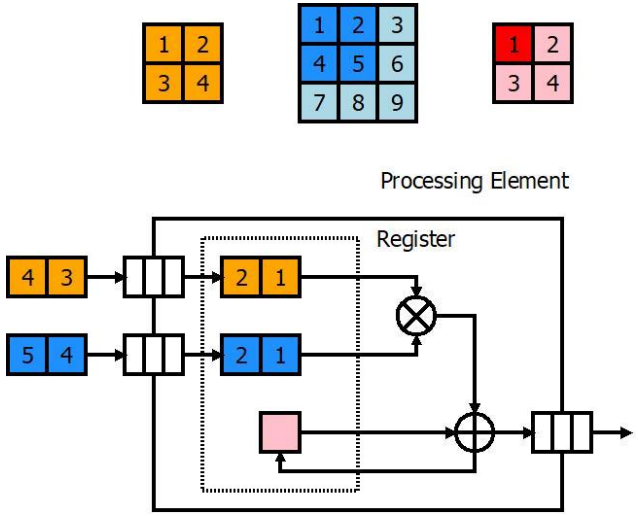
MIT Eyeriss DCNN Processor

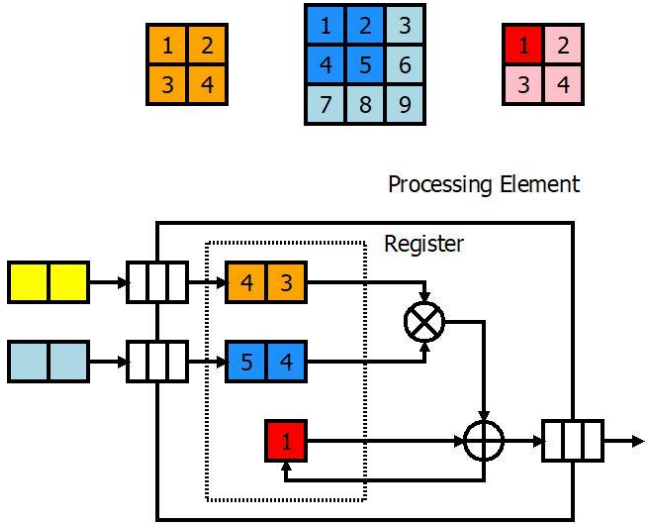


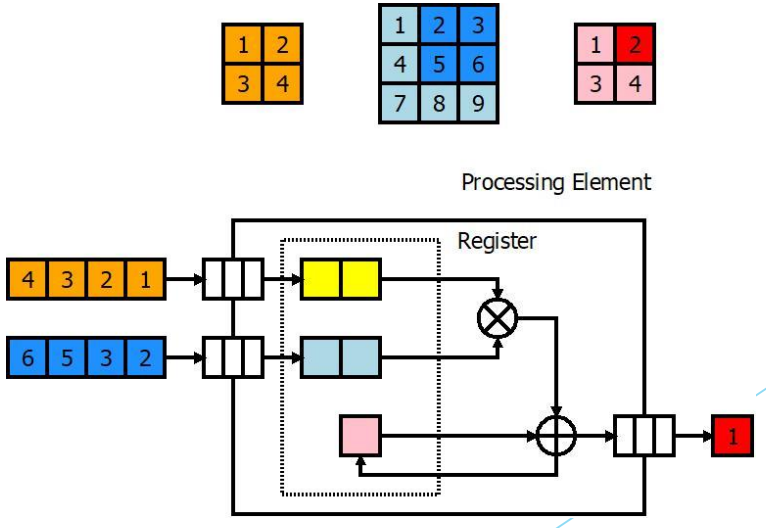
- Mapping 2D convolution to matrix multiplication to fully utilize hardware resource
- Convert 2D filter to 1D vector
- Map the feature map to Toeplitz matrix
- Rearrange 1D output vector to 2D matrix
- Extend to multiple channel filter operations

MIT Eyeriss DCNN Processor

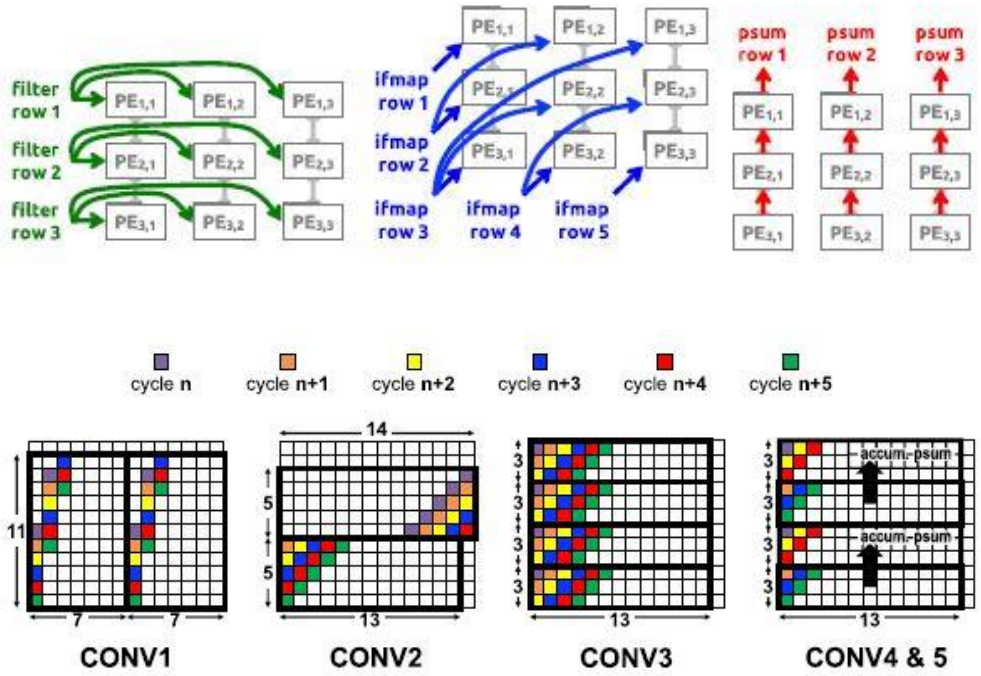






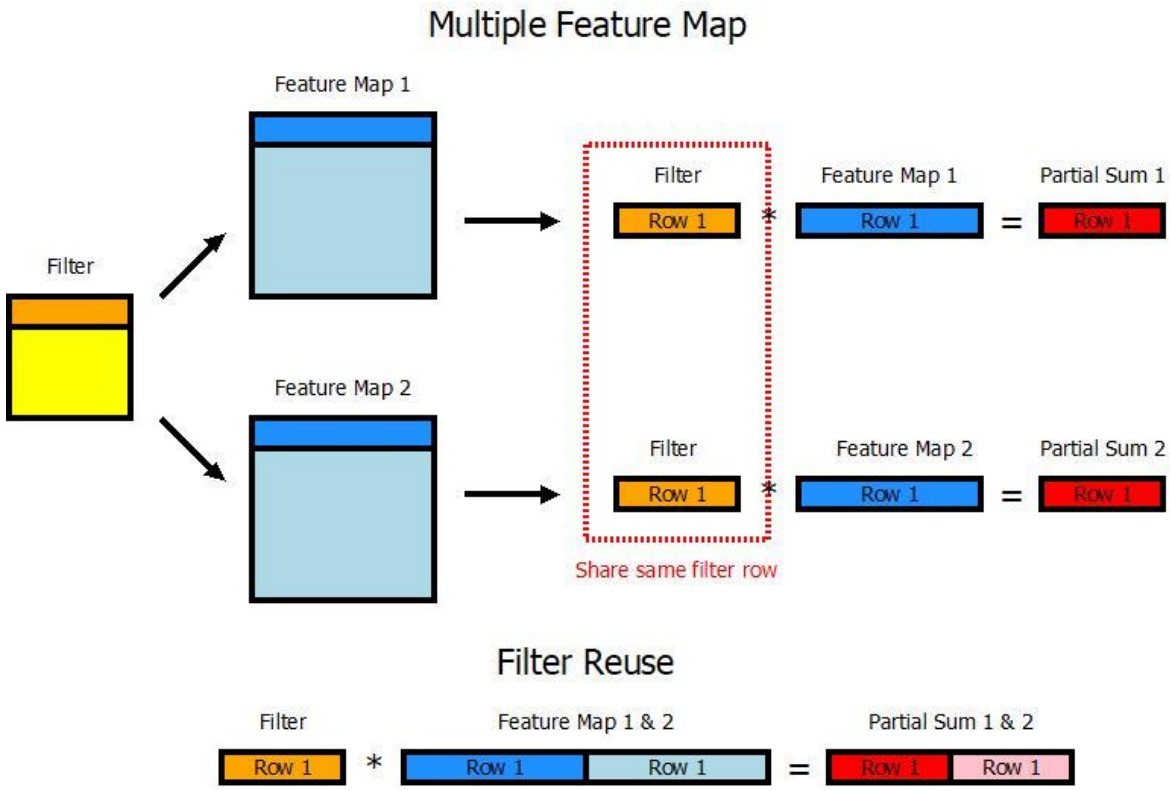


MIT Eyeriss DCNN Processor



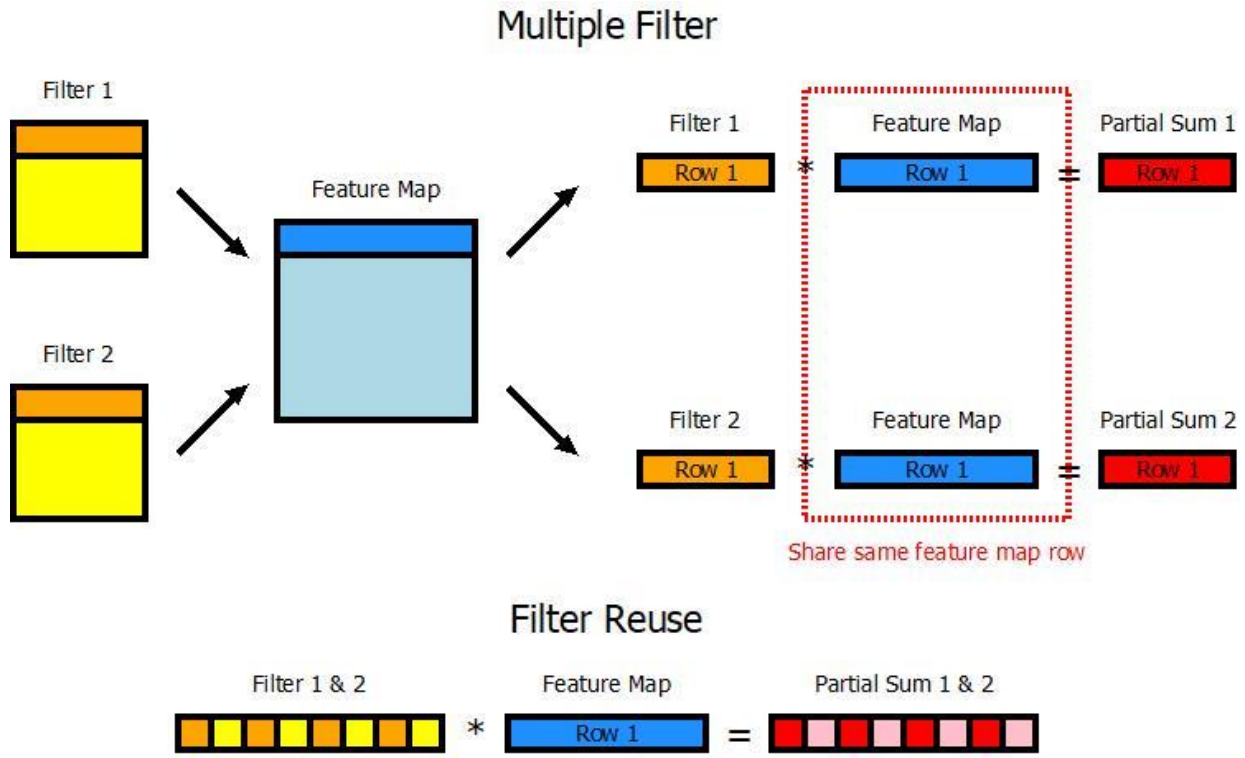
- Data movement optimization
  - Row of filter weights are reused across PEs horizontally
  - Row of feature map values are reused across PEs diagonally
  - Row of partial sum are reused across PEs vertically
- Processing Element Mapping
  - New mapping strategy is proposed to map a PE set into nearby PE array for local data sharing and partial sum accumulation
  - The dimension of PE set are the function of the shape of layer and independent of the PE array physical dimensin

MIT Eyeriss DCNN Processor



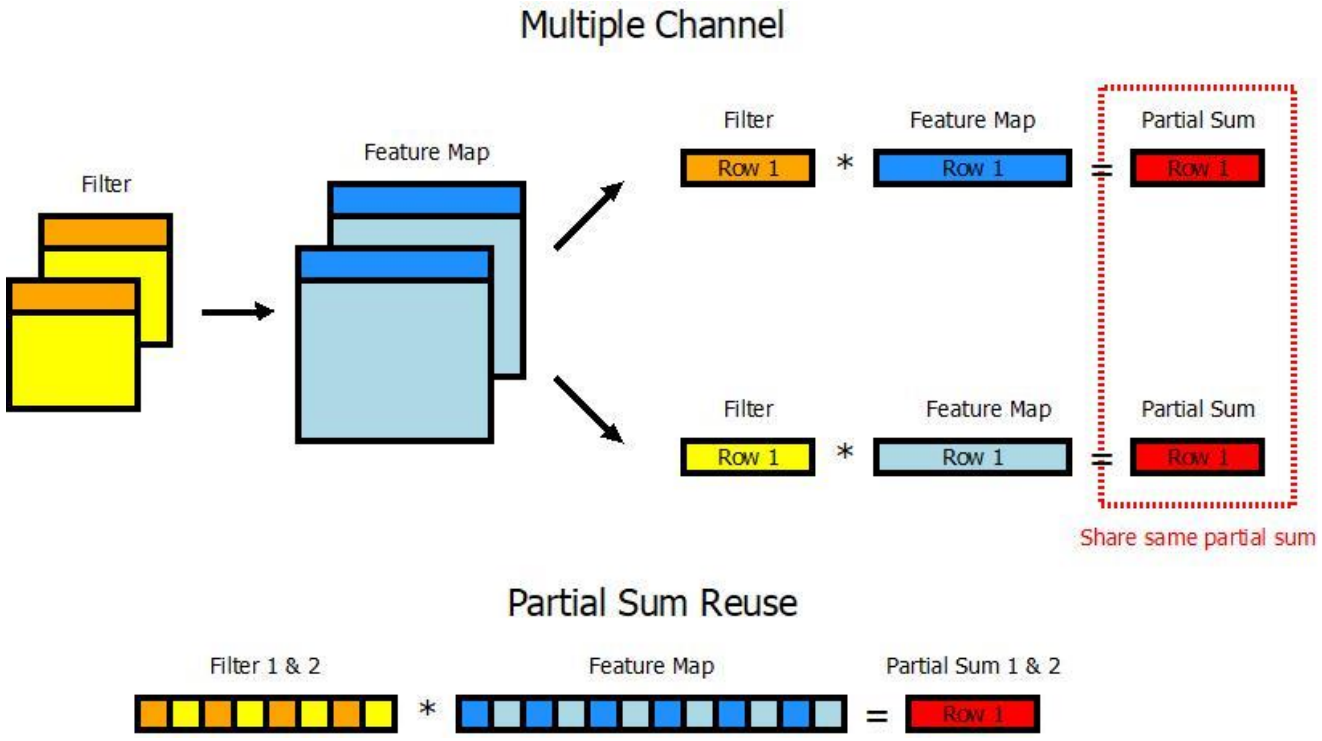
- Filter Reuse
  - For multiple feature maps, the feature map rows are concatenated together
  - Each PE perform 1D operation with same filter row

MIT Eyeriss DCNN Processor



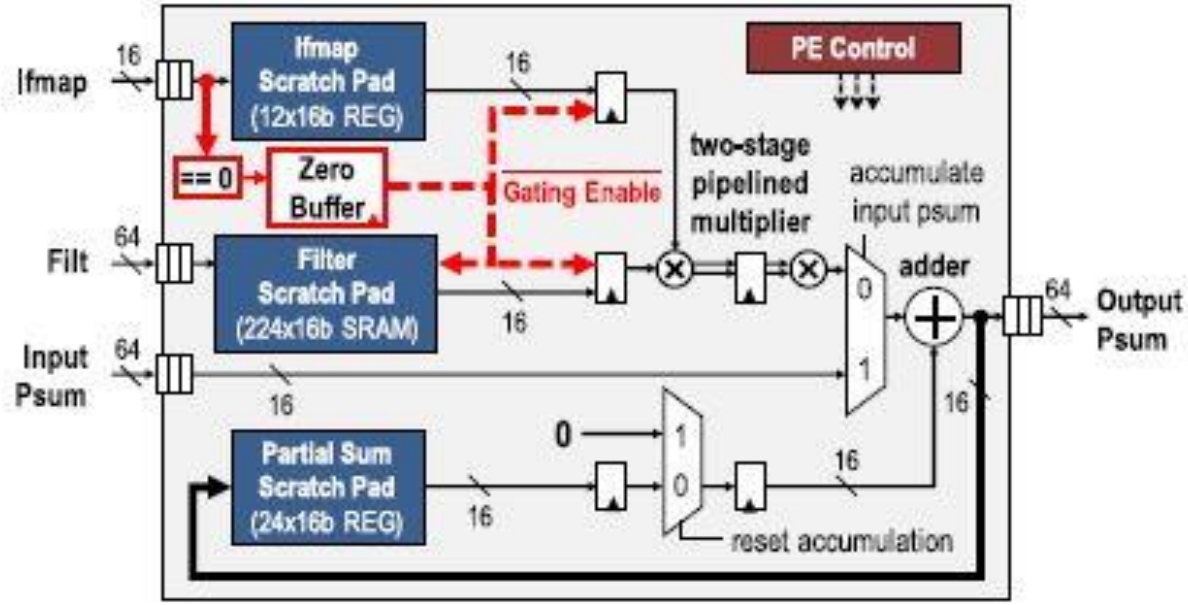
- Feature Map Reuse
  - For multiple filters, the filter rows are time interleaved
  - Each PE perform 1D operation with same feature map

MIT Eyeriss DCNN Processor



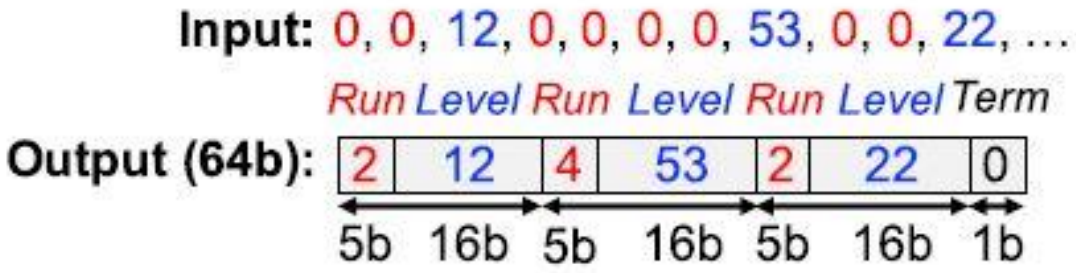
- Partial Sum Reuse
  - Both filter and feature map rows are time interleaved
  - Each PE perform 1D operation with different channels and accumulate the partial sum together

MIT Eyeriss DCNN Processor



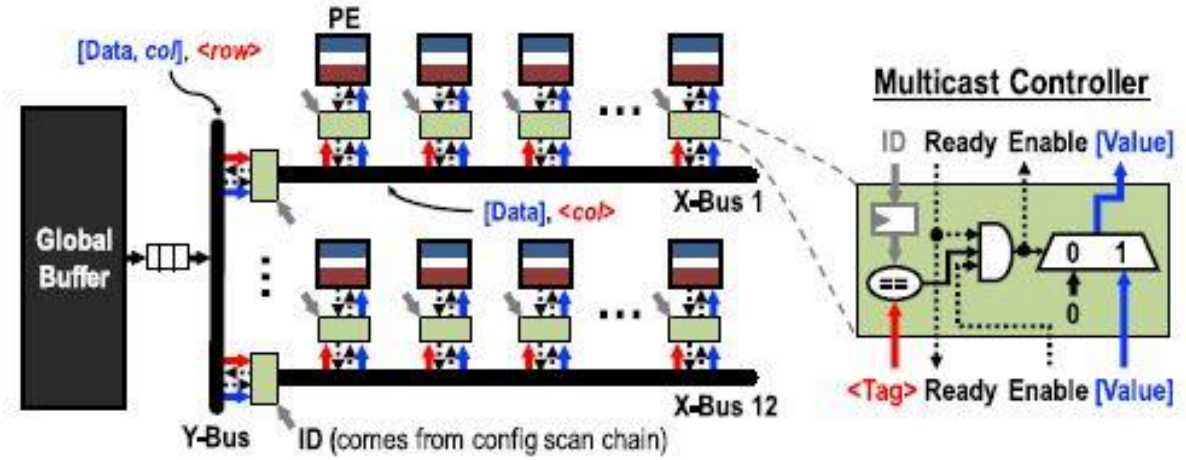
- DCNN PE datapath is pipelined into three stages, one stage for scratch pad access, the other two are 16bit two-stage pipelined multiplier and adder, the multiplication results are truncated from 32bit to 16bit.
- Data gating logic is used to exploit zero in feature map for power saving. If the zero feature map is detected by zero buffer, the gating logic is disable to stop MAC operating resulted in 45% power saving.

MIT Eyeriss DCNN Processor



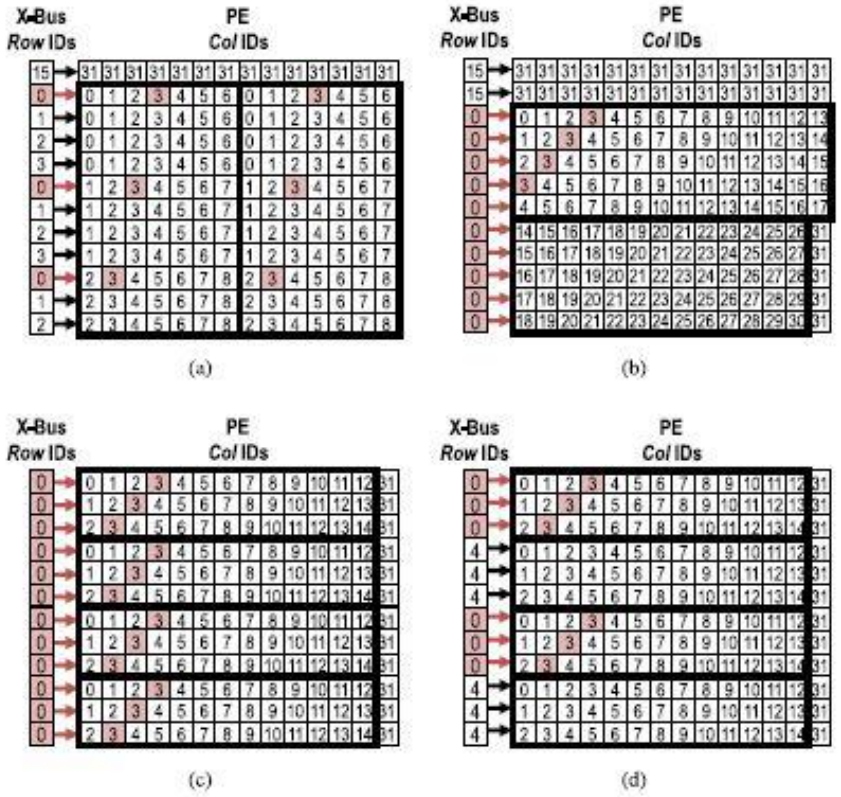
- Since ReLU function introduces many zeros in feature map by rectifying all negative results to zero, then Run-Length Compression (RLC) approach is used to encode non-zeros and reduce the memory bandwidth
- Consecutive zero with maximum run length of 31 are represented using 5bit number followed by 16bit value for run starts, it is packed into a 64bit word to significantly reduce the overall memory access
- RLC compression reduces memory access by 1.2X ~ 1.9X

MIT Eyeriss DCNN Processor



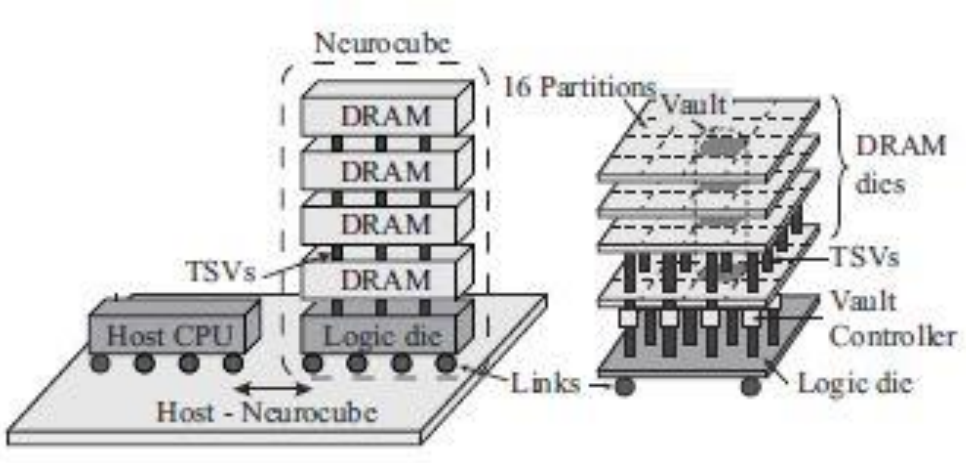
- Network-on-Chip (NoC) manages the data delivery between the Global Buffer (GLB) and PE array, it is divided into Global Input Network (GIN) and Global Output Network (GON)
- GIN is optimized for a single-cycle multicast from GLB to PE array, it is implemented using two level of hierarchy: Y-bus and X-bus.
- Vertical Y-bus consists of 12 horizontal X-bus, one at each row of PE array, each X-bus connects to 14 PEs in the row. Each X-bus has a row ID and each PE has a col ID. A unique ID is given to each group of X-buses of PEs, they are all reconfigurable.
- Each data read from GLB with a (row,col) tag-ID and delivered to the destination. The tag-ID is decoded using Multicast Controller (MC)

MIT Eyeriss DCNN Processor



- AlexNet with different tag-ID is used as an example
- DCNN maps the logical array to physical processing elements using replication and folding
- Unused PE are clock gated for power reduction

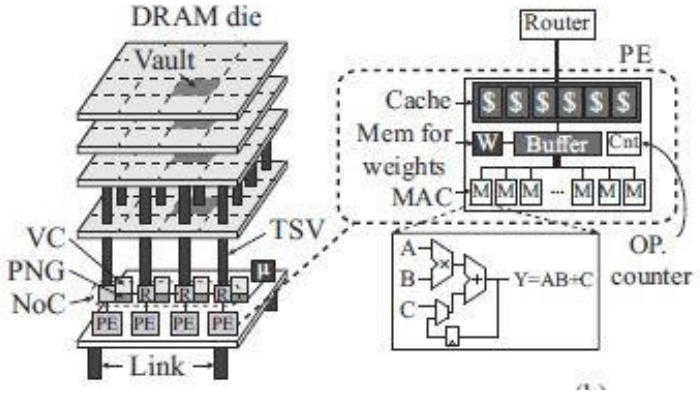
GT Neurocube Architecture

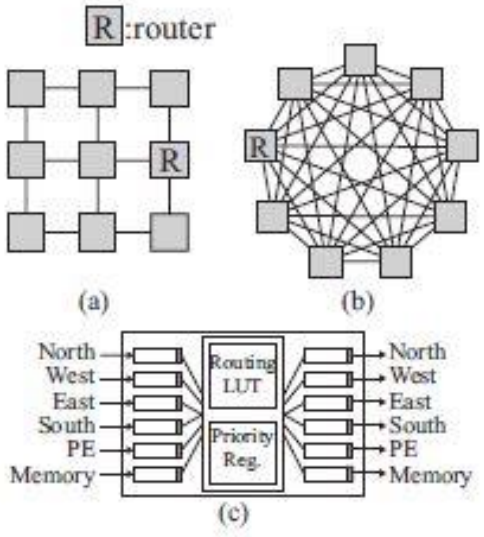


- In-Memory Neuromorphic Processing
  - Neurocube integrates a highly parallel, fine grained, computation layer within a 3D high density memory package – Hybrid Memory Cube (HMC)
- Memory-Centric Neural Computing (MCNC)
  - Apply programmable memory system to drive data flow enabled computation unit
- Programmable Neurosequence Generator (PNG)
  - Programmable Neurosequence Generator (PNG) is a memory based programmable state machines to generate PE connectivity as well as synaptic weights

Kim et al., Neurocube: A Programmable Digital Neuromorphic Architecture with High-Density 3D Memory, ISCA, 2016.

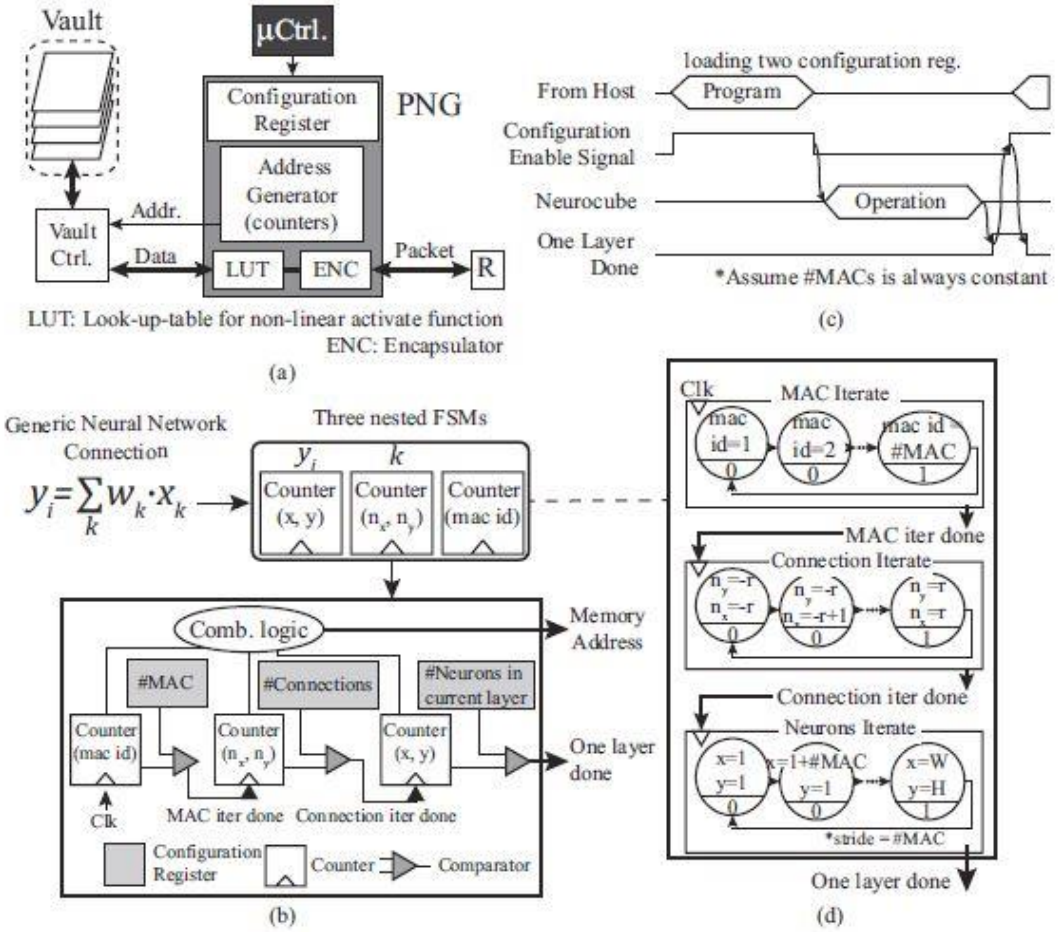
GT Neurocube Architecture





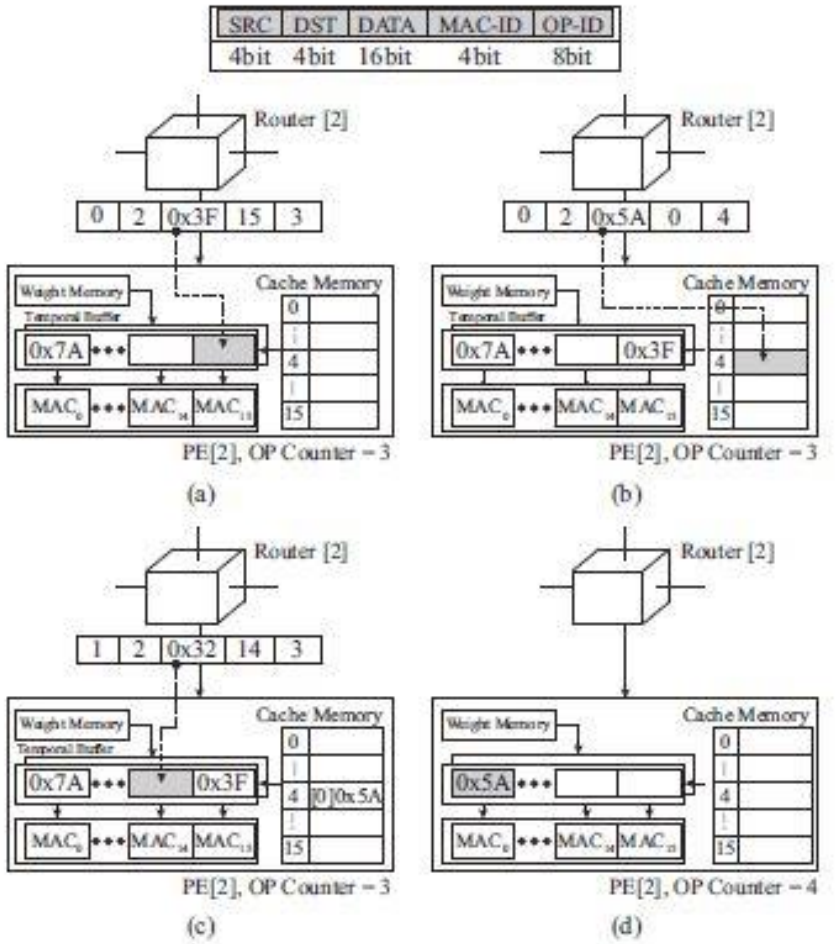
- HMC connects multiple stacked DRAM dies and single logic die using TSV
- Each DRAM die is divided into 16 partitions to form a single vault and each vault is connected to one Processing Element (PE)
- PE is comprised of 8 multiply-accumulator (MAC) and computes the neuron with 3 cycles
- All PEs are interconnected by a 2D mesh network through the single router. Each router has 6 input channels and 6 output channels (4 for neighboring routers and 2 for PE and memory)

GT Neurocube Architecture



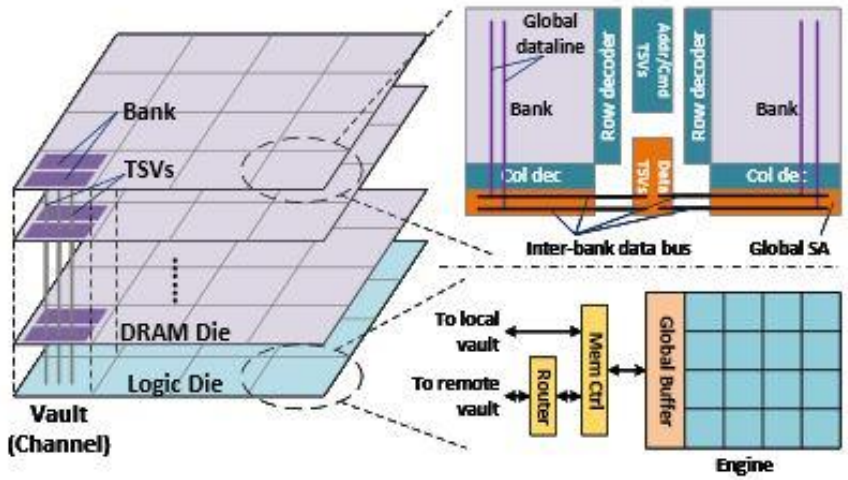
- Programmable Neurosequence Generator (PNG) consists of
  - Address generator
  - Configuration registers
  - Non-linear activation function Look-Up-Table (LUT)
- PNG computes the address through three loops
  - Loop across all neurons in the layer
  - Loop across all neurons connection
  - Loop across all MACs

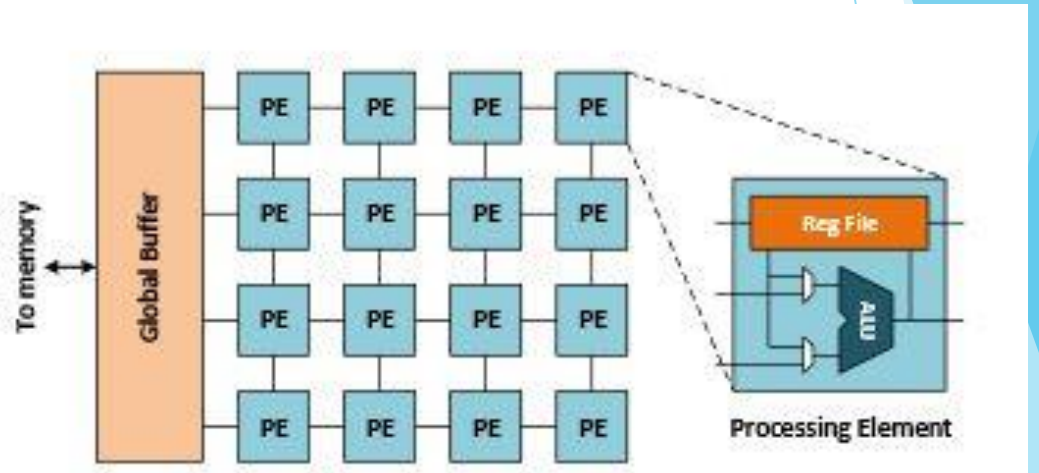
GT Neurocube Architecture



- PNG receives 32b data and encapsulates into two packets: source (SRC) – 4b for 16 DRAM vaults and destination (DST) 4b for 16 PE
- Each packet has a 4b MAC-ID for target MAC and 8b OP-ID represents the operation sequence
- If packet OP-ID is same as current OP-counter, it moves to temporal buffer, otherwise, it moves to cache memory
- If all temporal buffer is filled, MAC stars computation and updates the operation counter

Stanford Tetris DNN

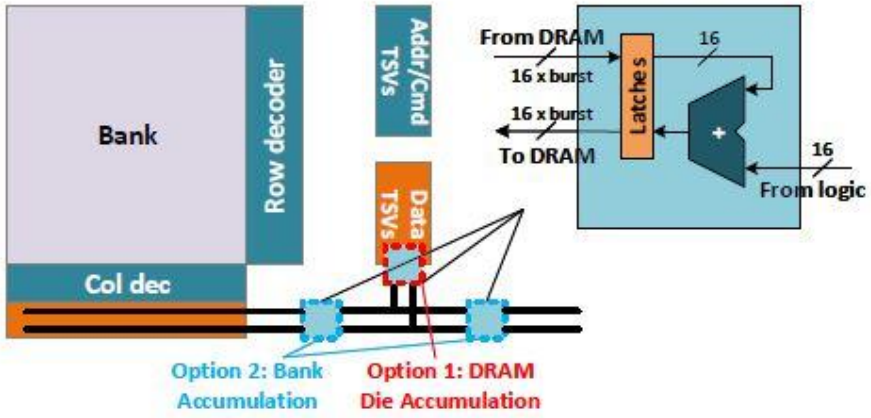


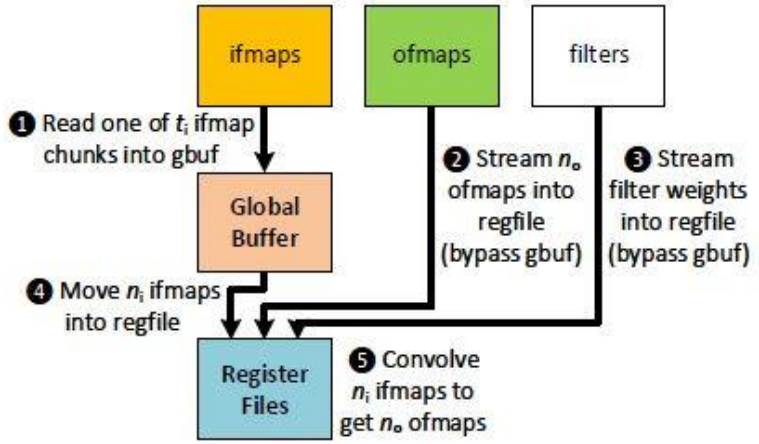


- Stanford Tetris DNN adopts MIT Eyeriss architecture with additional stacked 3D memory – Hybrid Memory Cube (HMC) to optimize memory access with significant power saving
- Replace the vault memory controller: crossbar switch with 3D Network-on-Chip (NoC) to improve memory efficiency
- Each PE contains a 16bit fixed point ALU and a small local register file of 512 to 1024 bytes

Gao et al., TETRIS: Scalable and Efficient Neural Network Acceleration with 3D Memory, ASPLOS 2017

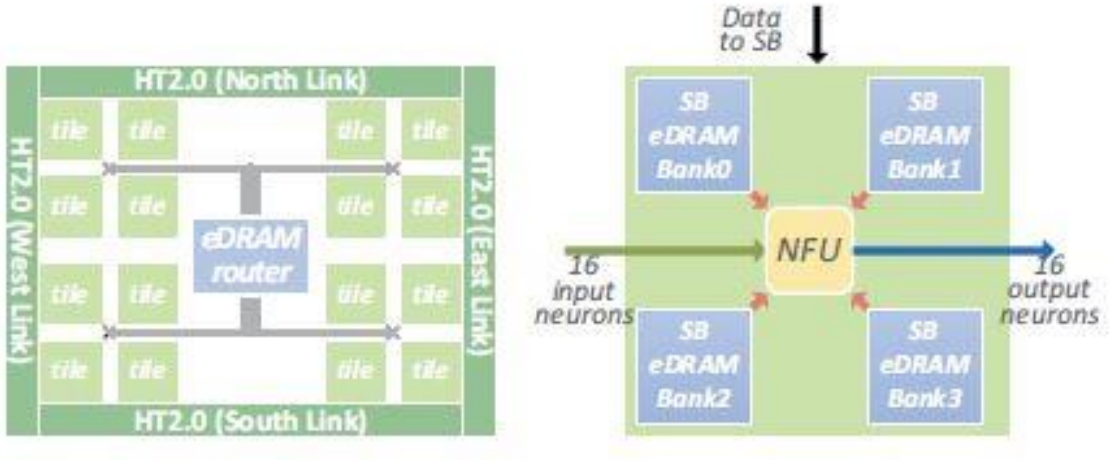
Stanford Tetris DNN





- In-memory accumulation support
  - Eliminate half of output feature map memory traffic to reduce memory access result in power saving
  - Reduced output feature map transfer save vertical TSV memory traffic
  - Combined back-to-back memory read/write access improve the row buffer utilization
  - Support both die and bank accumulation
- Dataflow schedule
  - Implement row stationary dataflow to map 1D convolution to utilize the local resource
  - Apply bypass ordering to bypass global buffer
    - IW bypass - avoid global buffer for input feature map and filter
    - OW bypass – avoid global buffer for output feature map and filter
    - IO bypass - avoid global buffer for input and output feature map

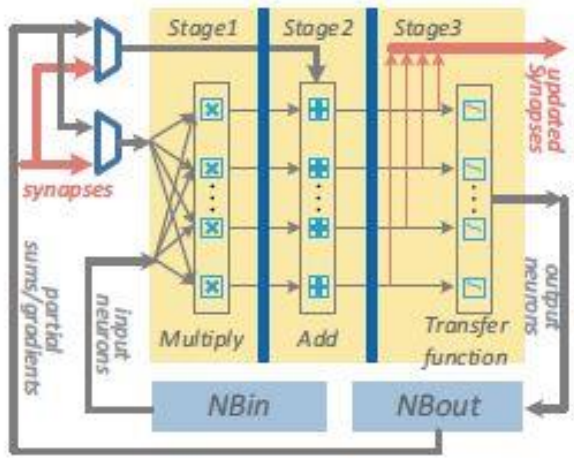
ICT DaDianNao Supercomputer

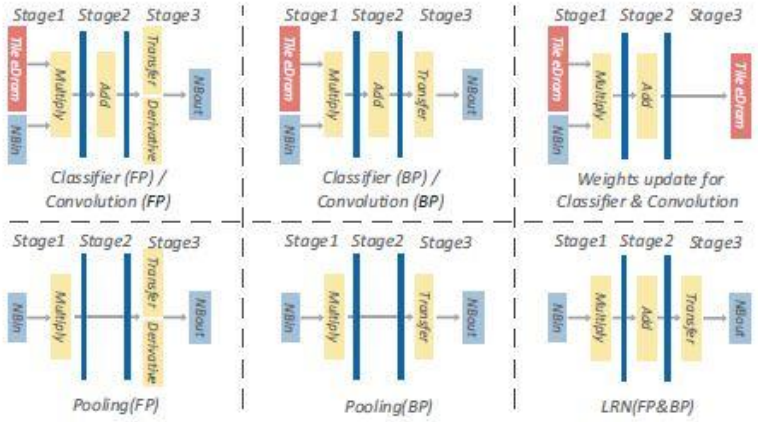


- DaDianNao architecture resolves DianNao memory bandwidth limitation using massive eDRAM closed to Neural Functional Unit (NFU)
- All the neurons are spread over different tiles that the Neural Functional Unit (NFU) can process 16 input and 16 output neurons simultaneously
- In order to resolve refresh issue, 4 band eDRAM configuration is used to achieve high internal bandwidth
- Support high speed interconnect using HyperTransport HT2.0

Chen et al., DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning, ICAS, 2014
Chen et al., DaDianNao: A Machine-Learning Super-computer, MICRO, 2014

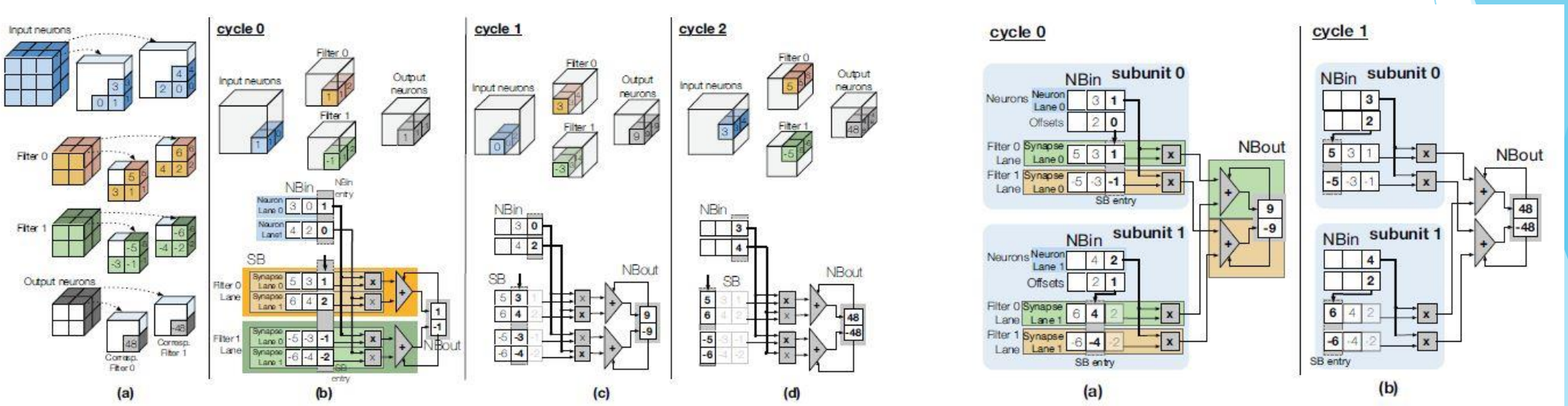
ICT DaDianNao Supercomputer





- NFU consists of multiple computational blocks:
  - Adder block (256 parallel adders)
  - Multiplier block (256 parallel multipliers)
  - Max block (16 max operations)
  - Transfer block (16 piecewise linear interpolations)
- Each NFU can reconfigure as
  - Classifier/Convolution
  - Pooling
  - Local Response Normalization

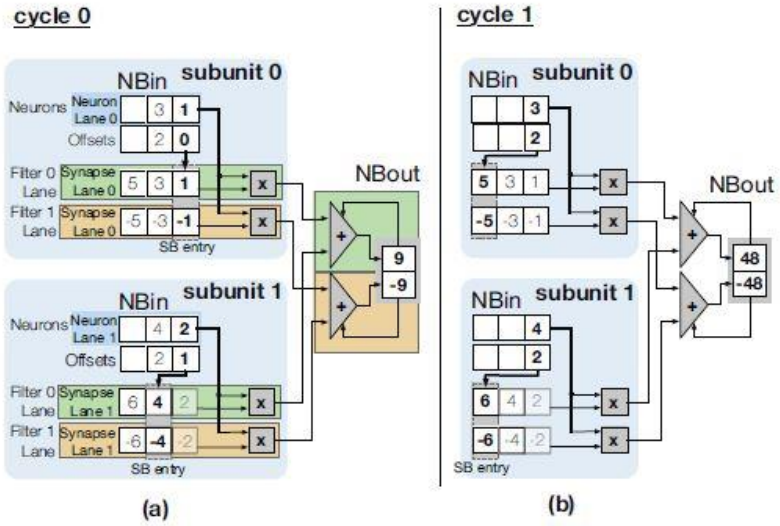
UofT Cnvlutin DNN Accelerator

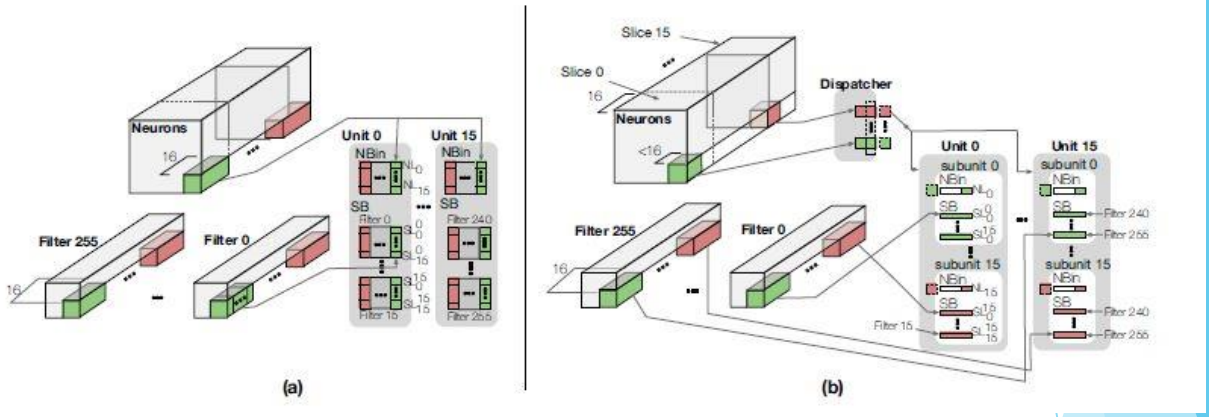


- Modified from massively parallel DaDianNao Neural Functional Unit (NFU) with nearby memory support
- Cnvlutin (CNV) architecture decouples original parallel multiplication lanes into finer-grain groups with new storing data structure
- Allow the multiplication lanes to skip over the zero input value and process the data in parallel

Albericio et al., Cnvlutin: Ineffectual-Neuron-Free Deep Neural Network Computing, ISCA, 2016

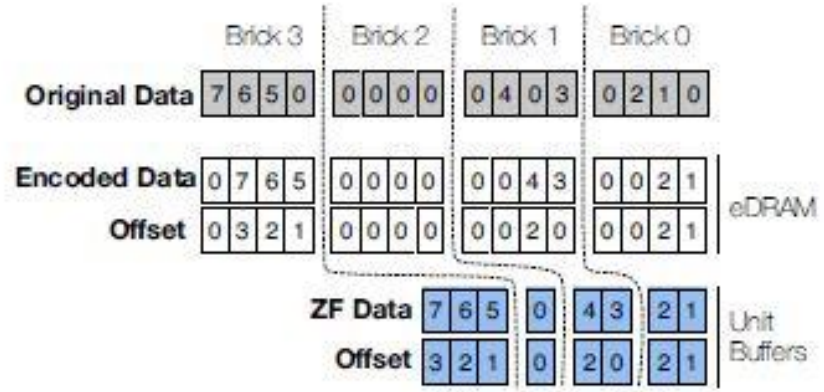
UofT Cnvlutin DNN Accelerator

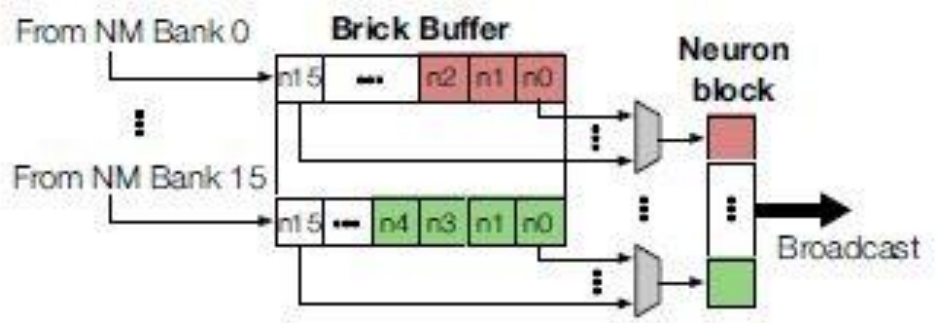




- Cnvlution DNN accelerator divides the synapse lanes into 16 independently operating subunits, each contains a single neuron lane and 16 synapse lanes
- Each synapse lanes processes a different filters based on the new data format (neuron, offset)
- CNV divides the window evenly into 16 slices, one per neuron lane. Each cycle, the data is fetched into neuron and processed independently, all the neuron keeps busy all the time. The overall performance is significantly improved with less power.

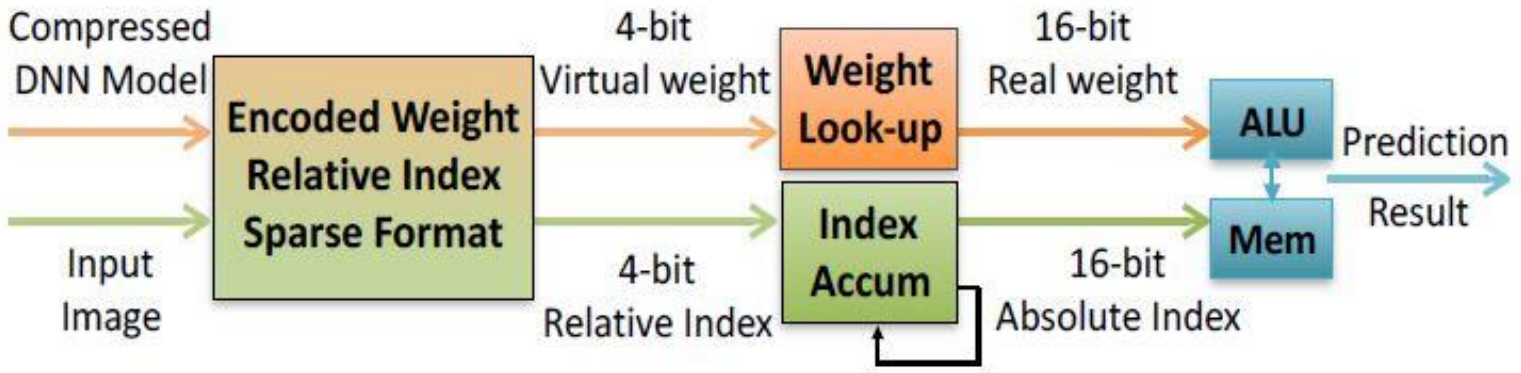
UofT Cnvlutin DNN Accelerator





- CNV accelerator uses new Zero-Free Neuron Array format (ZFNAf) that enables CNV to avoid zero-value neuron computation.
- ZFNAf is similar to CSR, it encodes neurons as (value,offset) pairs in groups called bricks. It correctly addresses the neuron array at a brick granularity. It also keeps the offset field short to avoid the offset overhead
- CNV employs the dispatcher unit that makes 16 neuron wide access to Neuron Memory (NM) in parallel and make all the neuron lanes busy all the time.

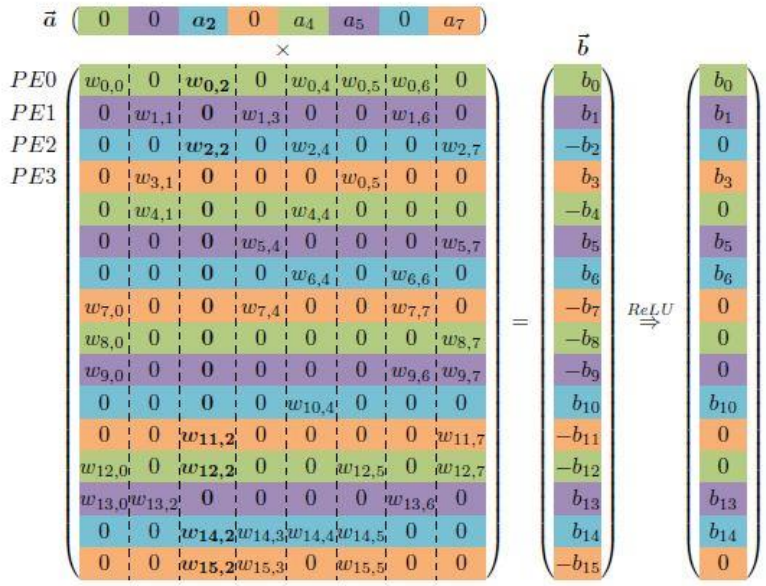
Energy Efficient Inference Engine (EIE)



- Efficient Inference Engine is a scalable array of processing elements (PEs)
- Operate on the compressed DNN model with narrow weights (4bit)
- Perform customized sparse matrix vector multiplication with distributed memory
- Handle weight sharing with no loss of efficiency

Han et al., EIE: Efficient Inference Engine on Compressed Deep Neural Network, ISCA 2016.

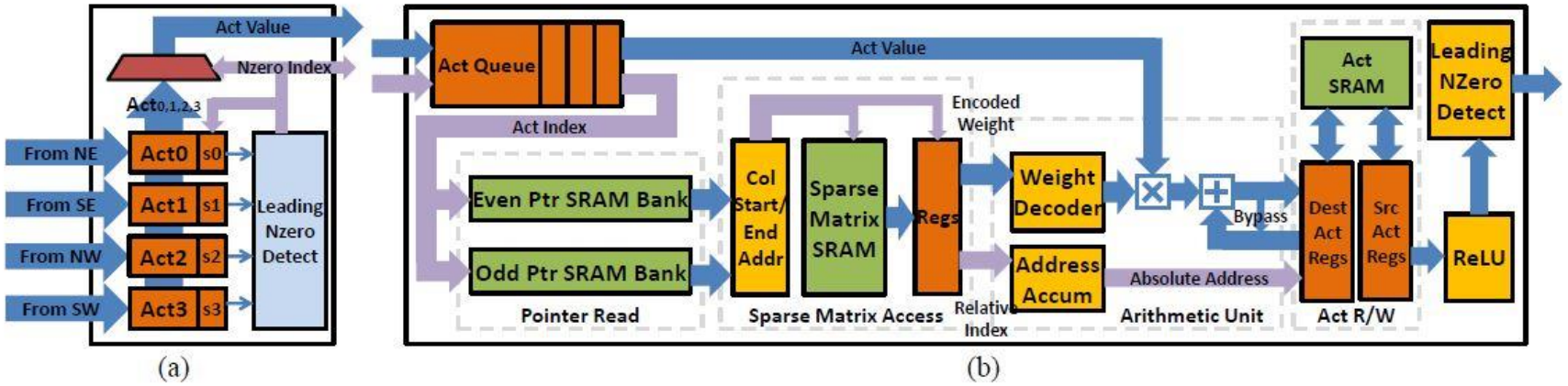
Energy Efficient Inference Engine (EIE)



Virtual Weight	$W_{0,0}$	$W_{8,0}$	$W_{12,0}$	$W_{4,1}$	$W_{0,2}$	$W_{12,2}$	$W_{0,4}$	$W_{4,4}$	$W_{0,5}$	$W_{12,5}$	$W_{0,6}$	$W_{8,7}$	$W_{12,7}$
Relative Row Index	0	1	0	1	0	2	0	0	0	2	0	2	0
Column Pointer	0	3	4	6	6	8	10	11	13				

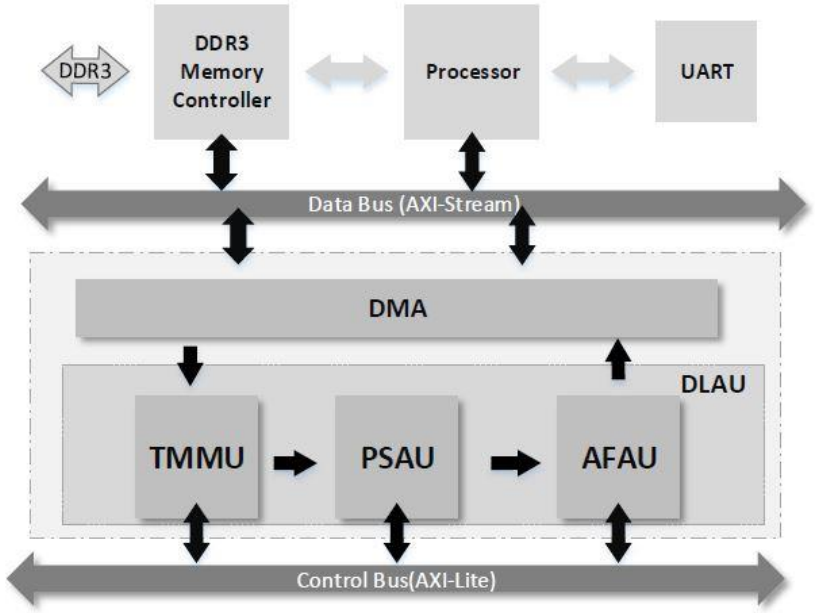
- Perform deep compression with combination of pruning and weight sharing
- Exploit the dynamic sparsity through relative indexing
- Apply the Compressed Sparse Column (CSC) to encode the non-zero weights
- Perform sparse matrix and sparse vector multiplication and broadcast to corresponding PEs

Energy Efficient Inference Engine (EIE)



- Central Control Unit (CCU) broadcasts the non-zero weights and index to activation queue and perform load balance to each PE
- Look up the sparse matrix index through Point Read Unit and feed the matrix into Sparse Matrix Read Unit
- Perform the matrix-vector multiply and accumulate using arithmetic unit
- Activation Read/Write Unit contains two registers to accommodate the activation values for operation
- Leading None-Zero Detection Node (LNZD) distributes the non-zero results to all PEs

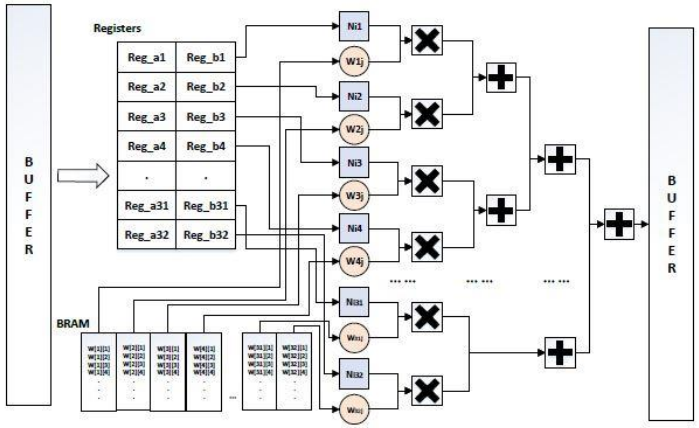
Deep Learning Accelerator Unit (DLAU)

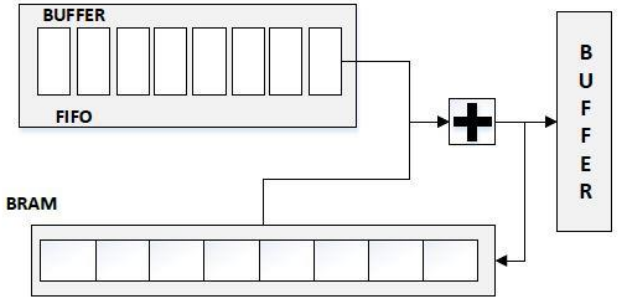


- Deep Learning Accelerator Unit (DLAU) employs the tile technique to partition large scale input data for deep learning application
- DLAU system consists of embedded processor, DMA module and DLAU accelerator
- DLAU accelerator performs the deep learning operations through three fully pipeline processing units:
  - Tilted Matrix Multiplication Unit (TMMU)
  - Part Sum Accumulation Unit (PSAU)
  - Activation Function Acceleration Unit (AFAU)

Wang et al., DLAU: A Scalable Deep Learning Accelerator Unit on FPGA, TCAD, 2016.

Deep Learning Accelerator Unit (DLAU)





- TMMU is used to perform multiplication and accumulation operations with binary pipelined adder.
- The input data is first transferred from DMA to TMMU input FIFO and reads into BRAM, then data is passed to interleaved register file for computation.
- The results are transferred to TMMU output FIFO and stream into PSAU for partial sum addition
- With TMMU interleaved register, PSAU can calculate the partial sum every cycle and significantly improve overall throughput
- Finally, AFAU perform the activation function using piecewise linear interpolation

Machine Learning

Hello!! R2-D2

Thanks You