Memory Management
&
System Design

Instructor: Siting Liu

School of Information Science and Technology (SIST), ShanghaiTech University

Shanghai Institute of Semiconductor Innovation (SISI)

https://faculty.sist.shanghaitech.edu.cn/liust/EE219/EE219.html



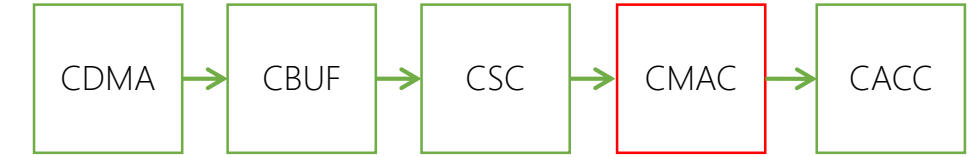
Computer Architecture - EE219

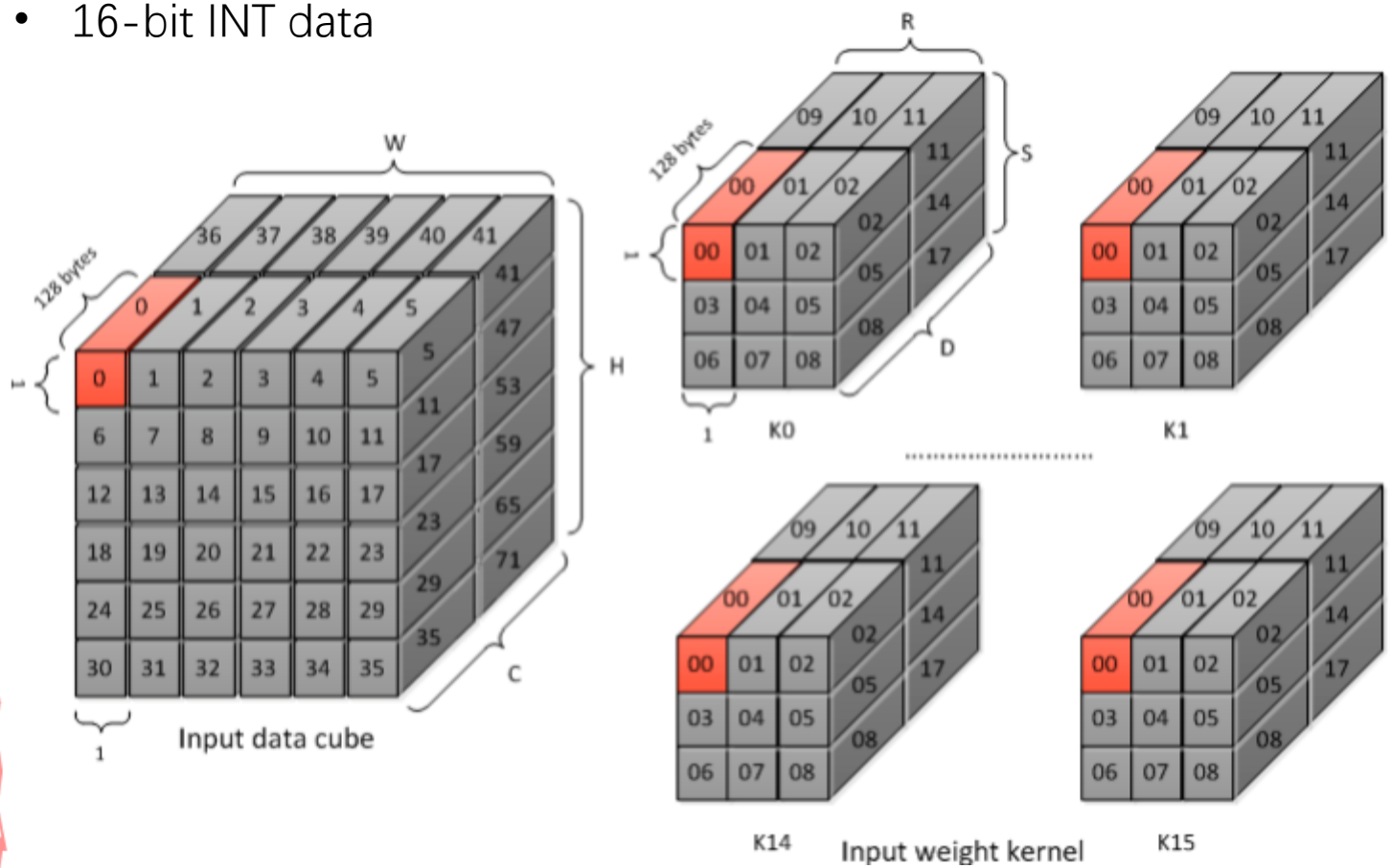
faculty.sist.shanghaitech.edu.cn

立志成才报国裕民

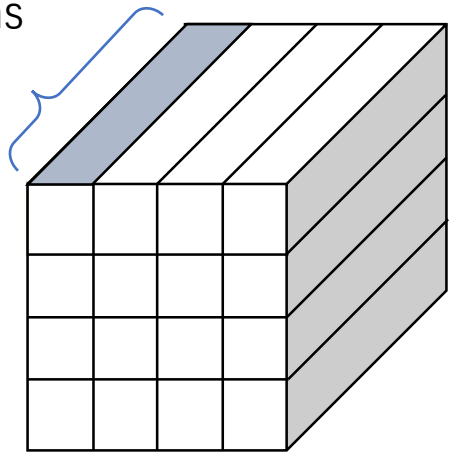
Direct Convolution

- Atomic/basic operation (not the one in CA)
- One clock cycle
- 16-bit INT data



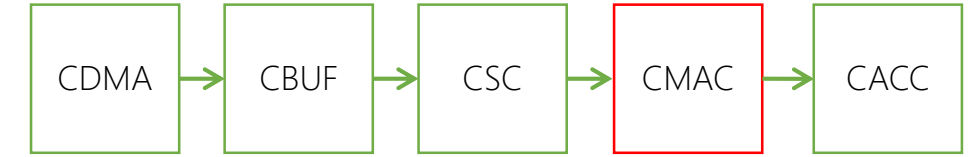


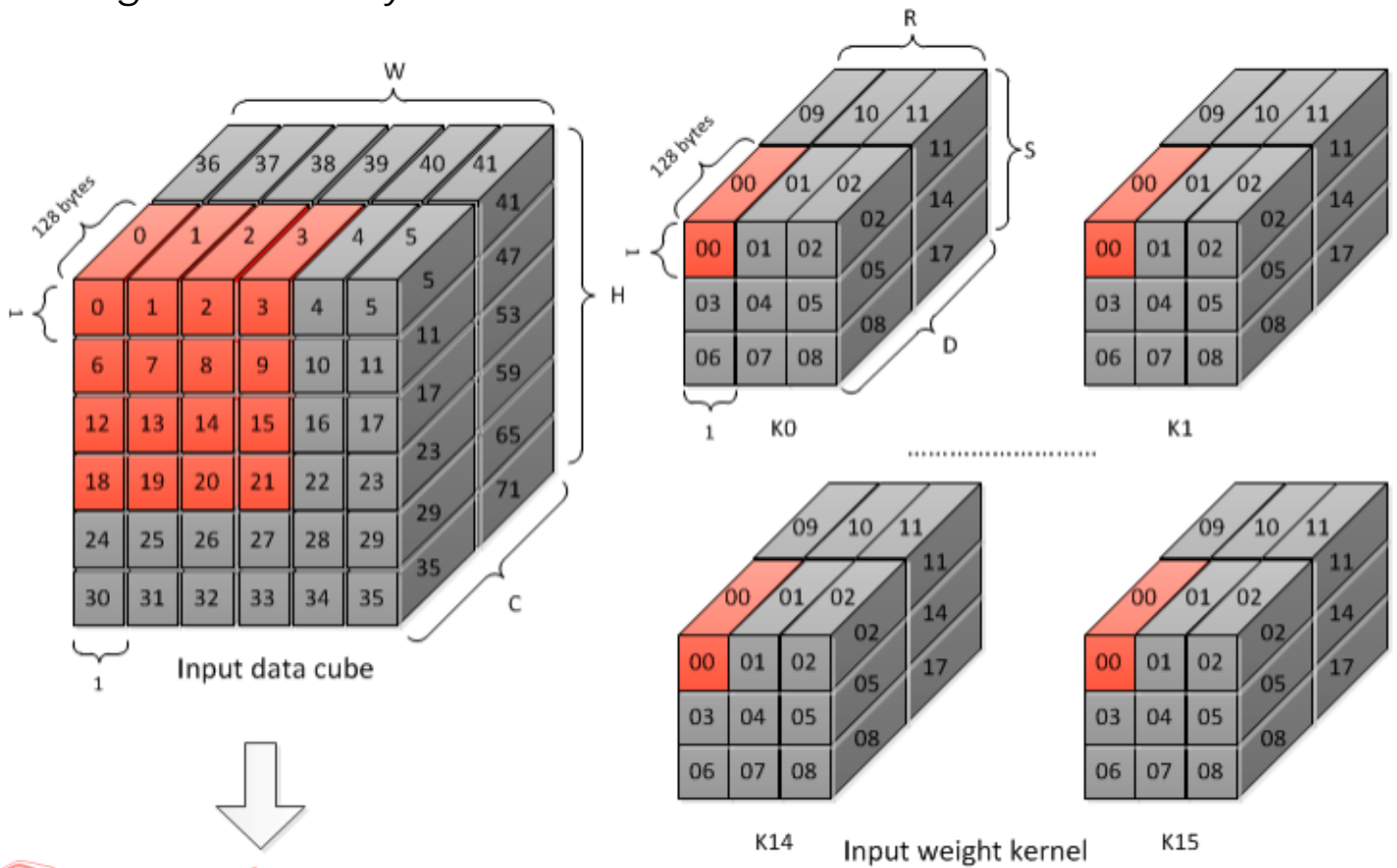
Partial sums
(32 bytes)



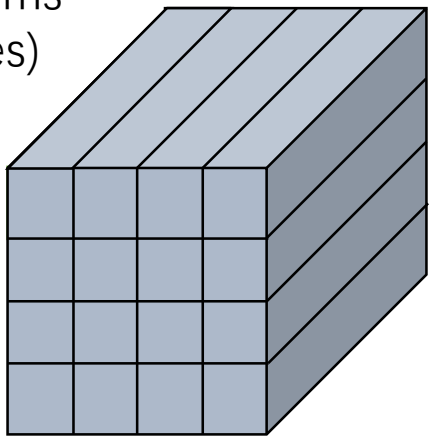
Direct Convolution (Cont'd)

- Stripe operation
- 16 clock cycles
- Weight-stationary





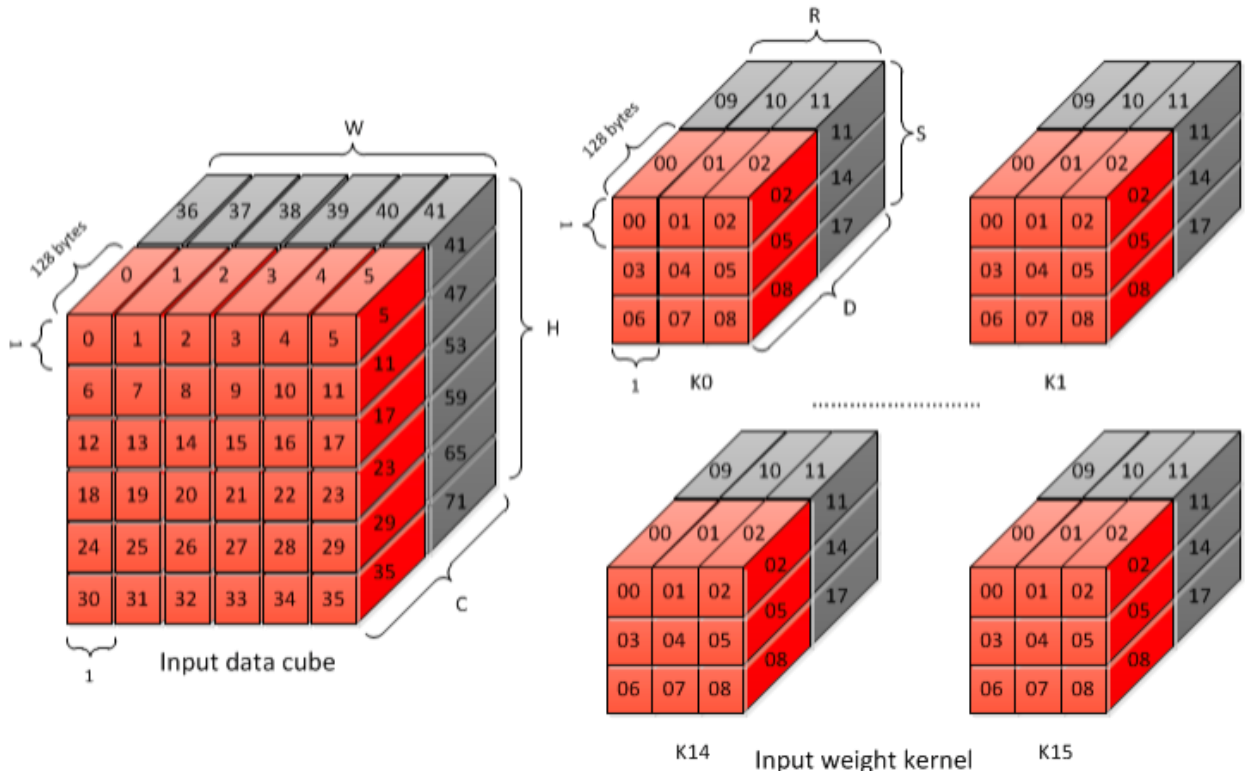
Partial sums
(32 bytes)



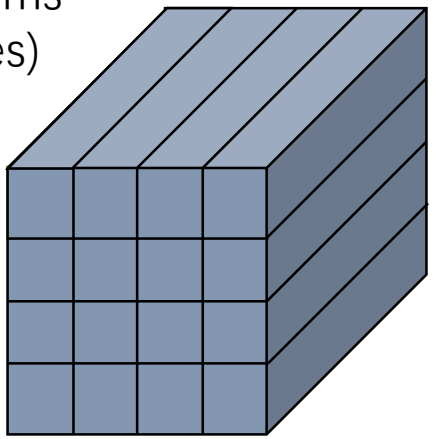
Direct Convolution (Cont'd)



- Block operation
- 16 * 9 clock cycles
- Weight-stationary



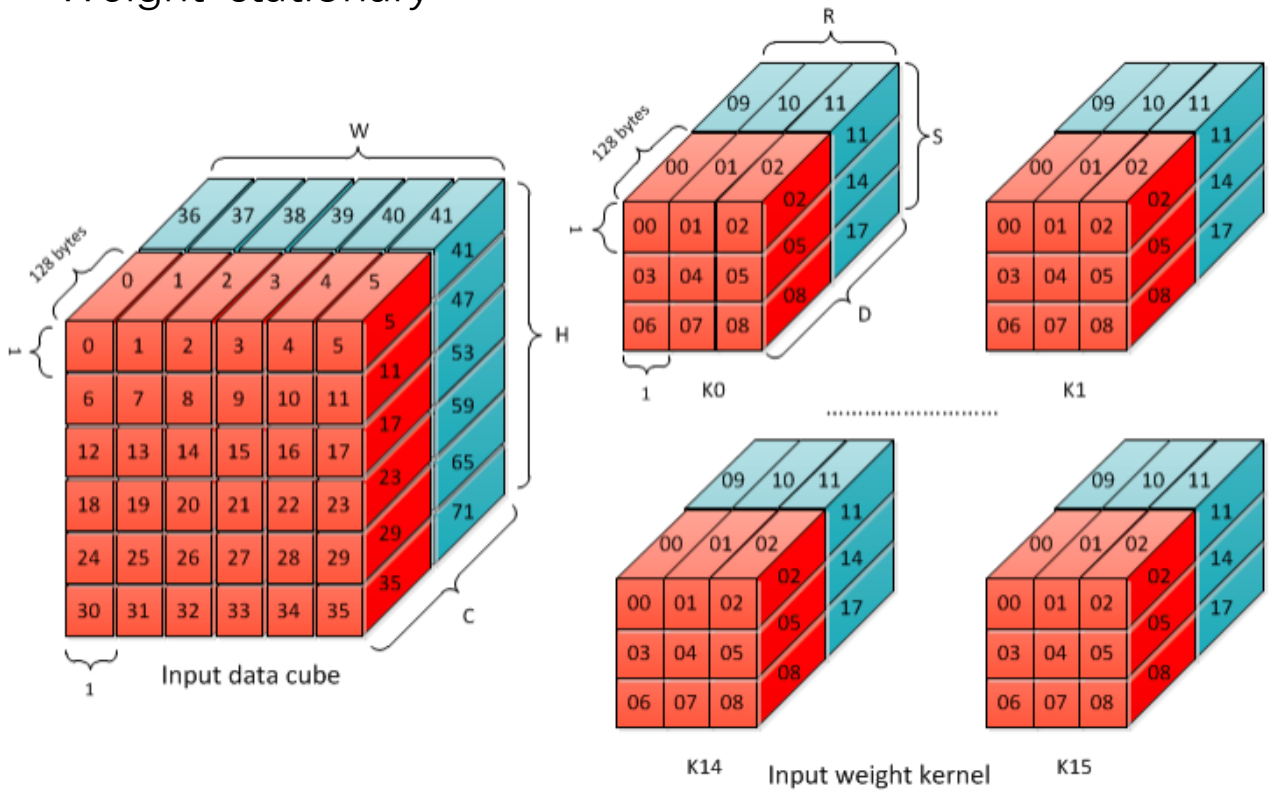
Partial sums
(32 bytes)



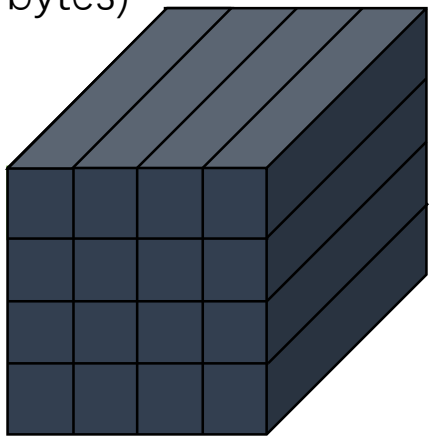
Direct Convolution (Cont'd)

- Channel operation
- 16 * 9 * 2 clock cycles
- Weight-stationary

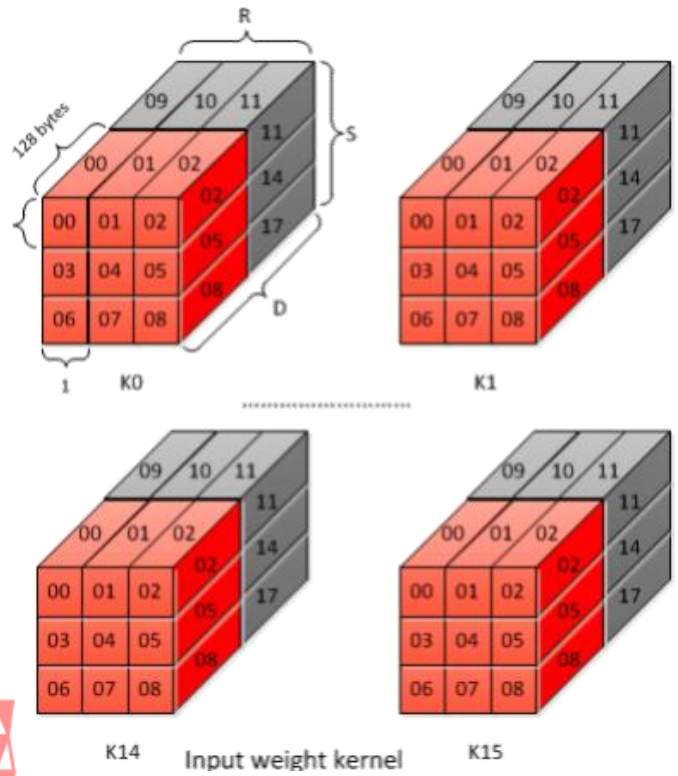


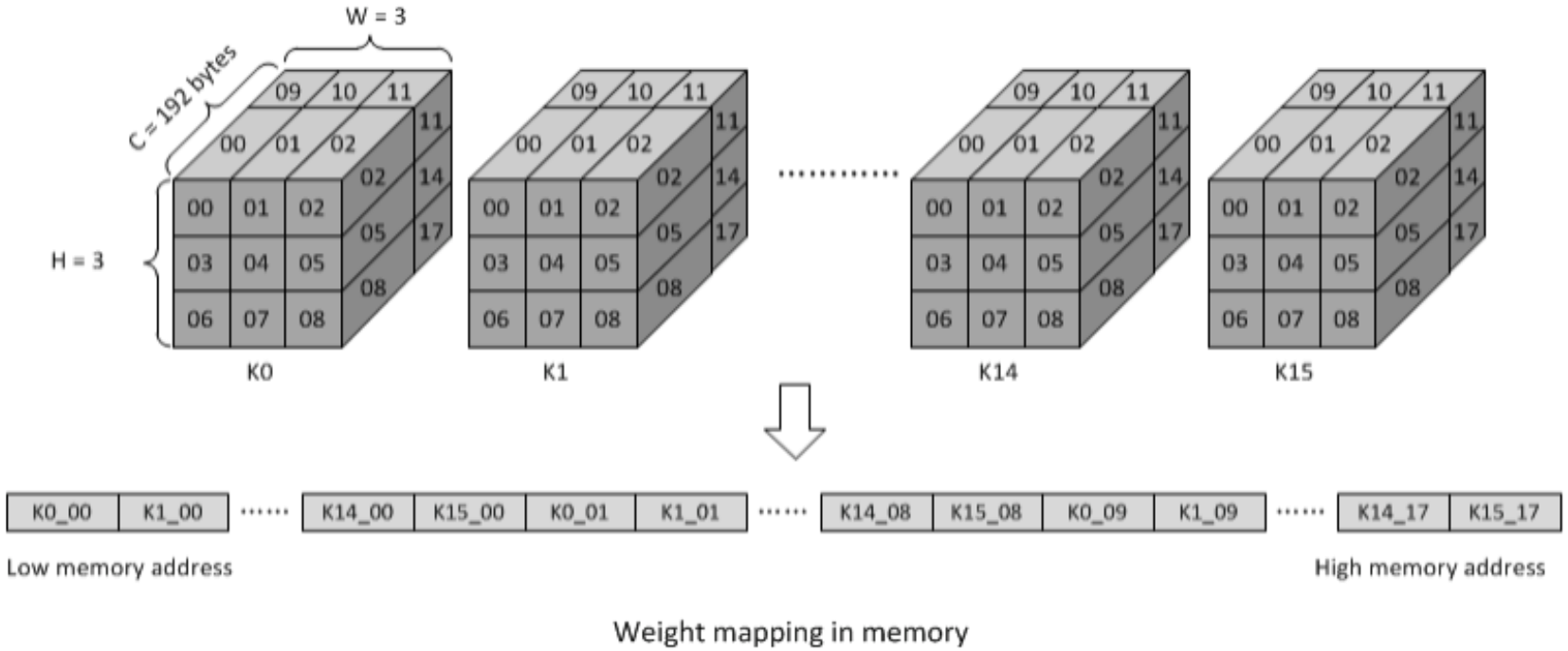


Outputs (32 bytes)

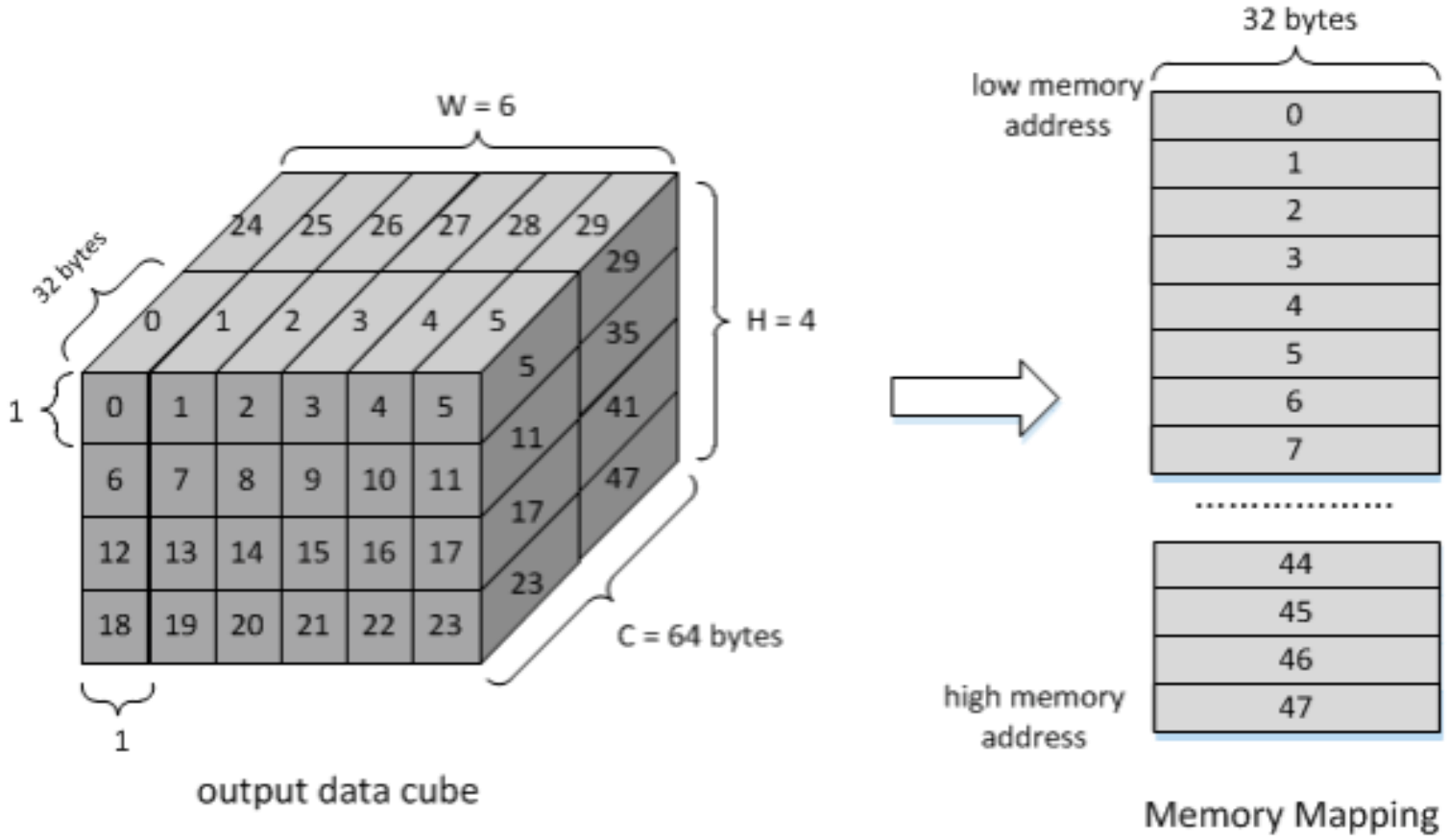


Direct Convolution – Weight Layout





Direct Convolution Output Sequence



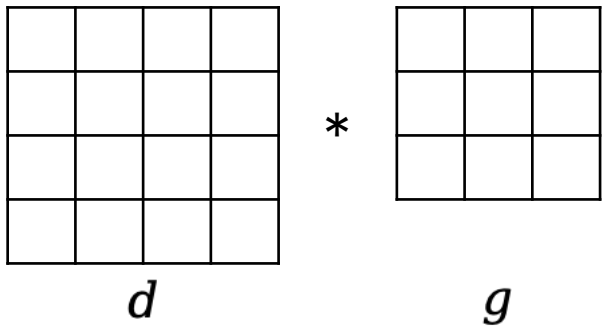
Winograd Convolution

- Winograd recap.

$$B^T = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

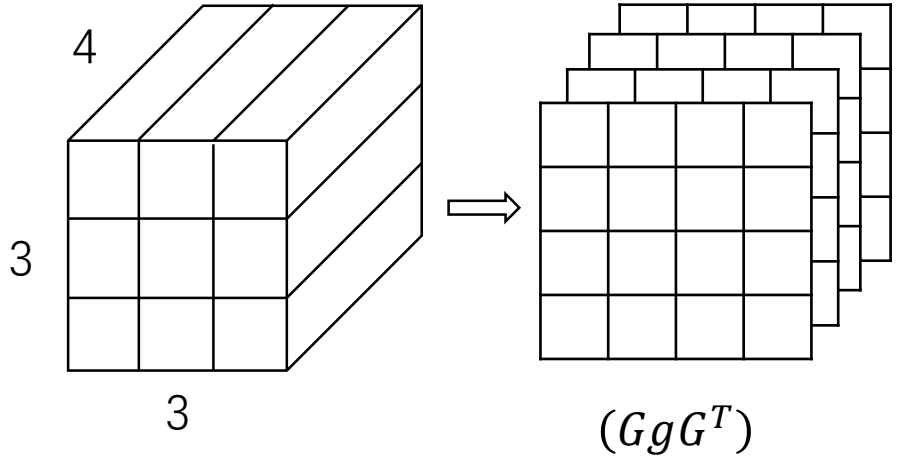
$$G = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 \\ 0 & 0 & 1 \end{bmatrix}$$

$$A^T = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & -1 \end{bmatrix}$$



$$Y = A^T[(GgG^T) \otimes (B^T dB)]A$$

Computed offline



Fit the 64 MAC units

立志成才报国裕民

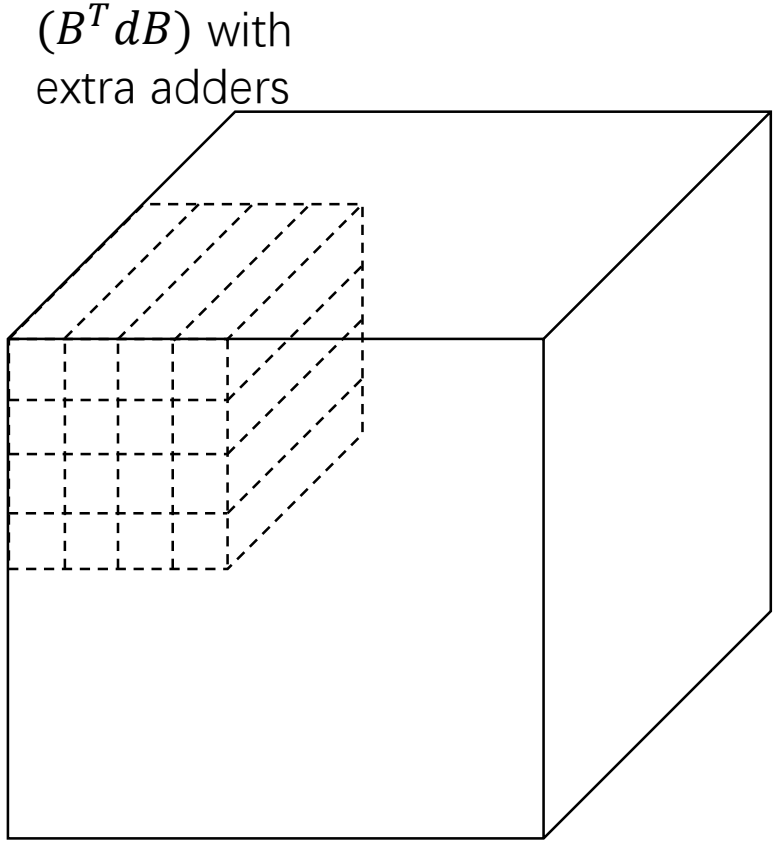
Winograd Convolution (Cont'd)

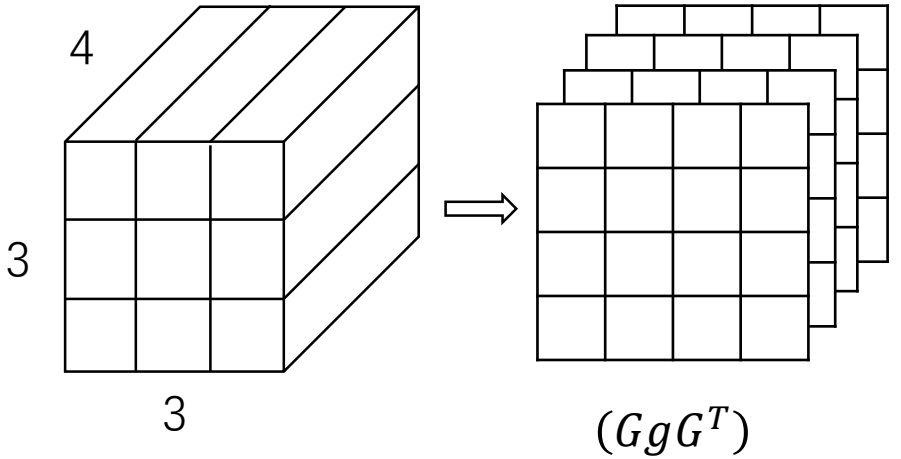
- Winograd mapping to NVDLA

$$B^T = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 \\ 0 & 0 & 1 \end{bmatrix}$$

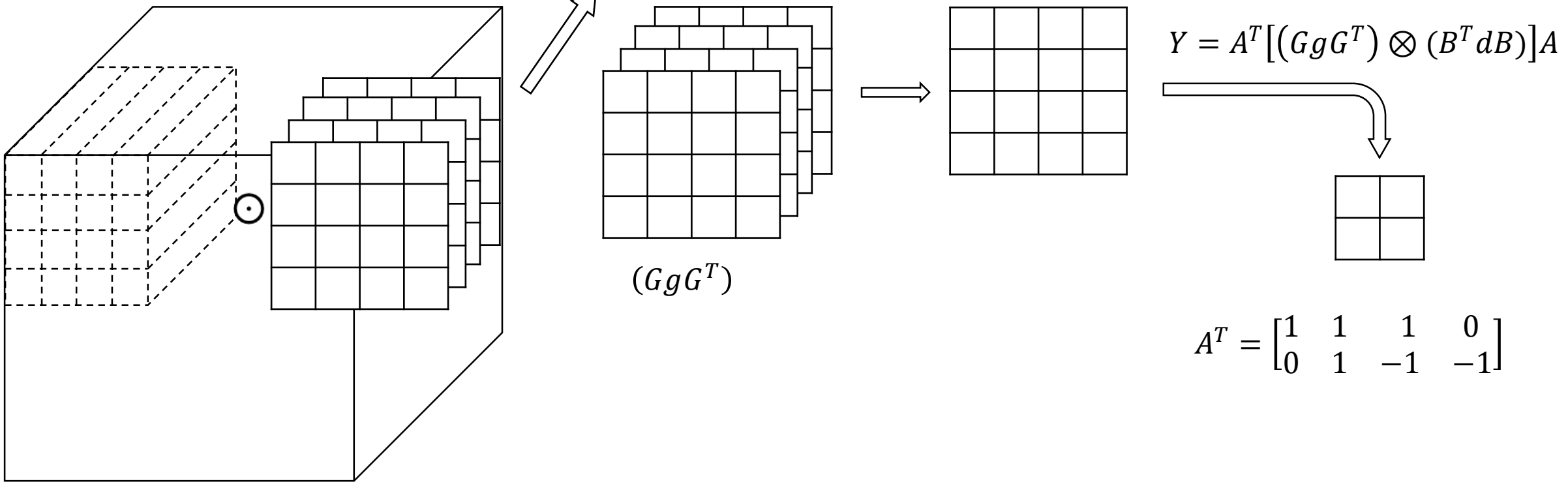
$$A^T = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & -1 \end{bmatrix}$$





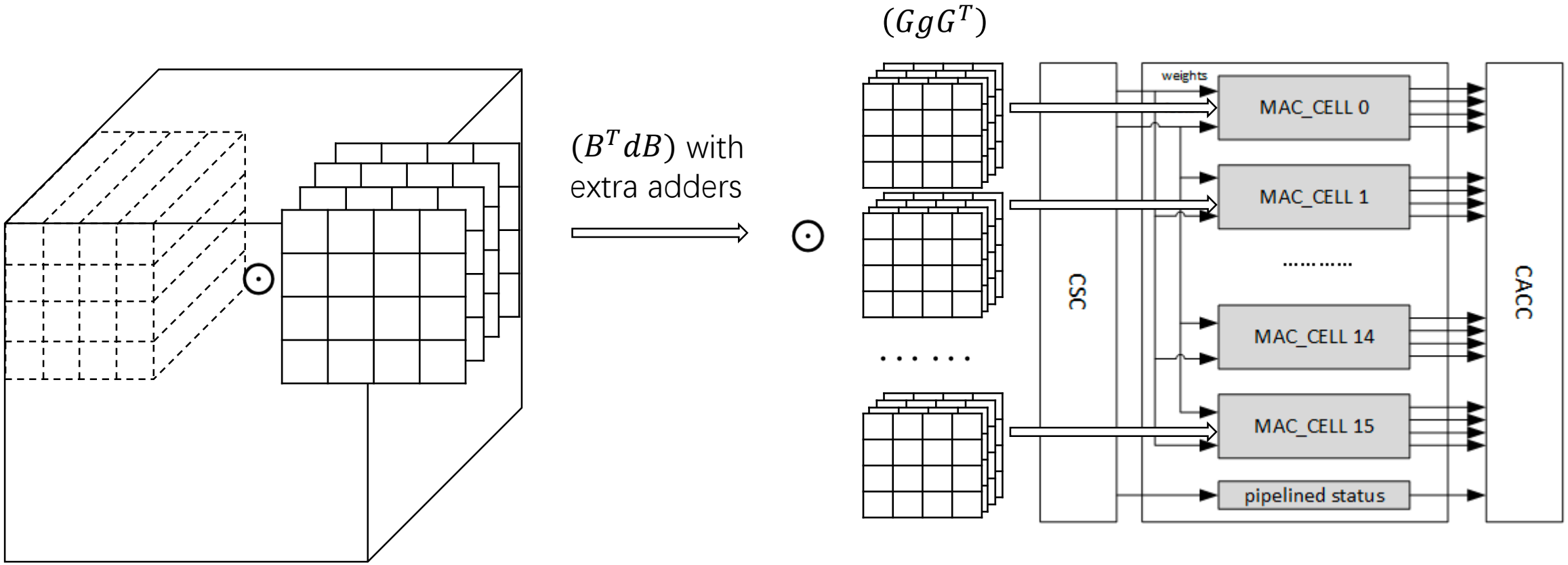
Winograd Convolution (Cont'd)

- Winograd mapping to NVDLA

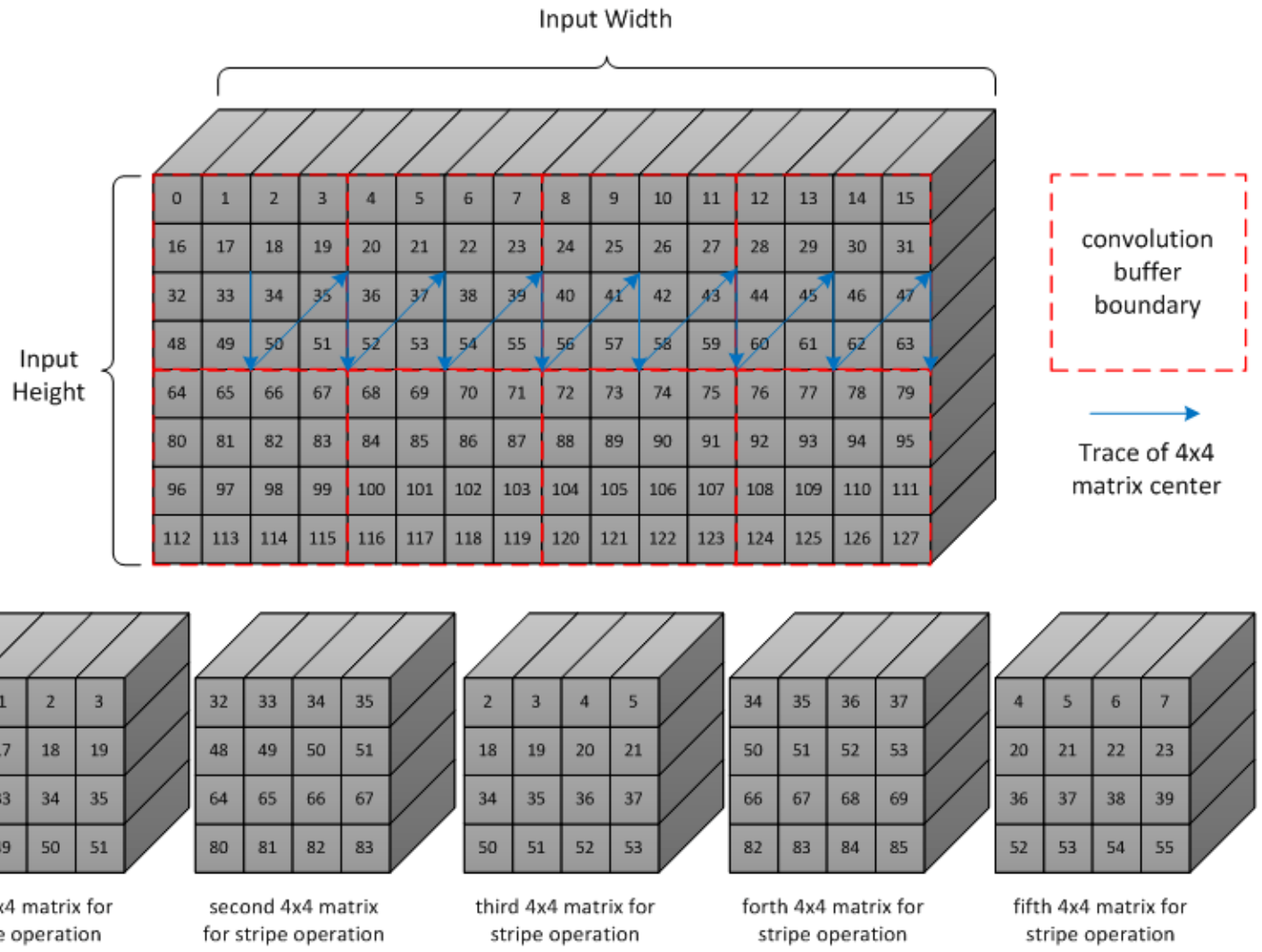


Winograd Convolution (Cont'd)

- Atomic operation

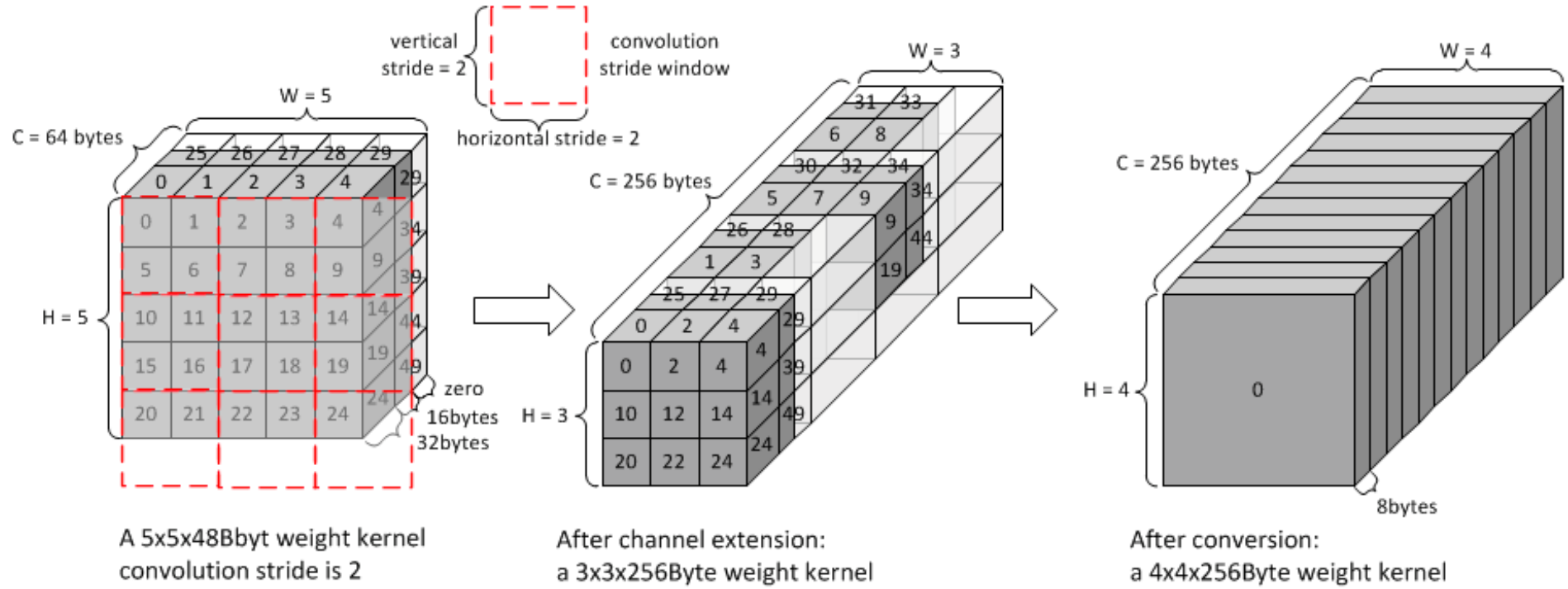


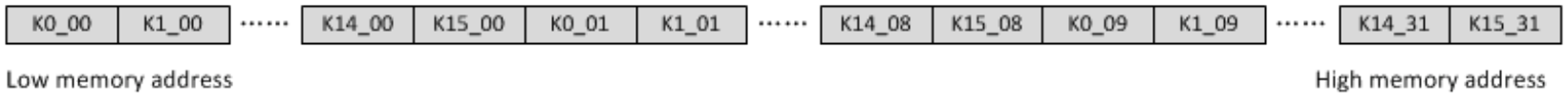
Winograd Convolution (Cont'd)



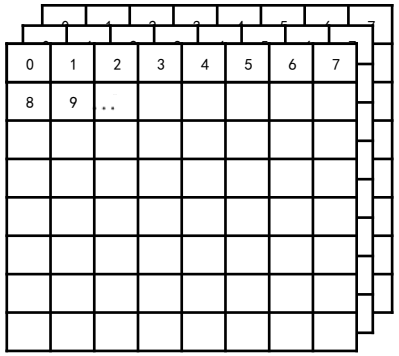
Winograd Convolution (Cont'd)

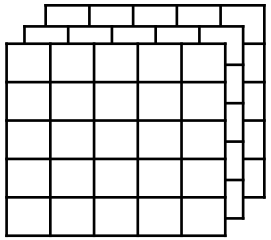
- For kernels with stride larger than 1

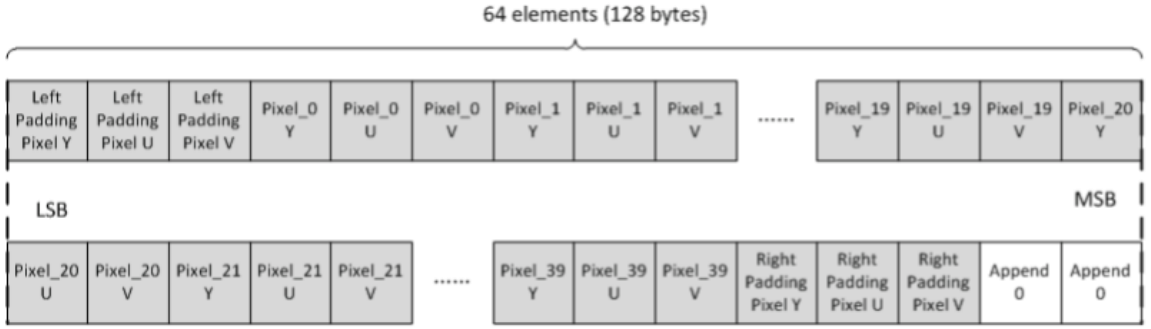




Input Image Convolution

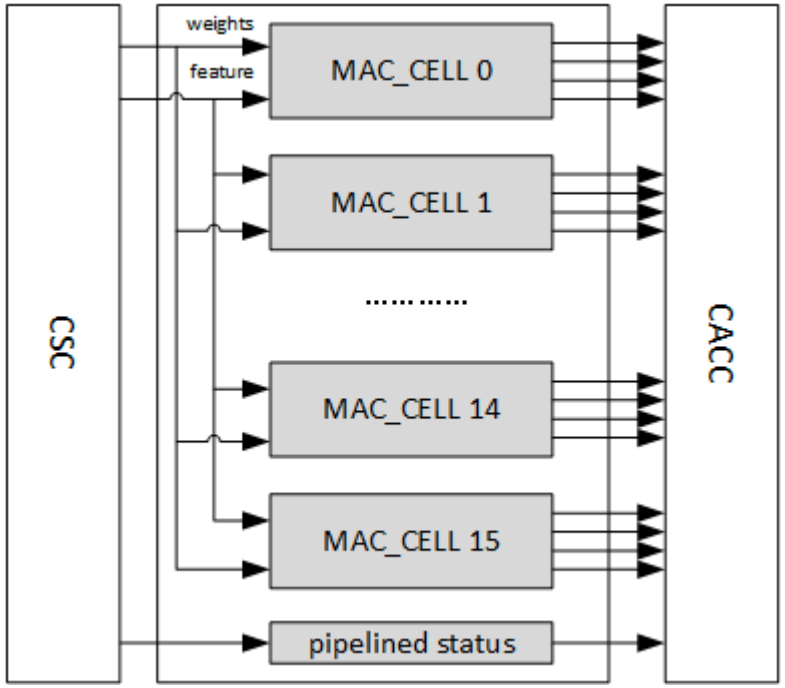






Convolution buffer entry 0

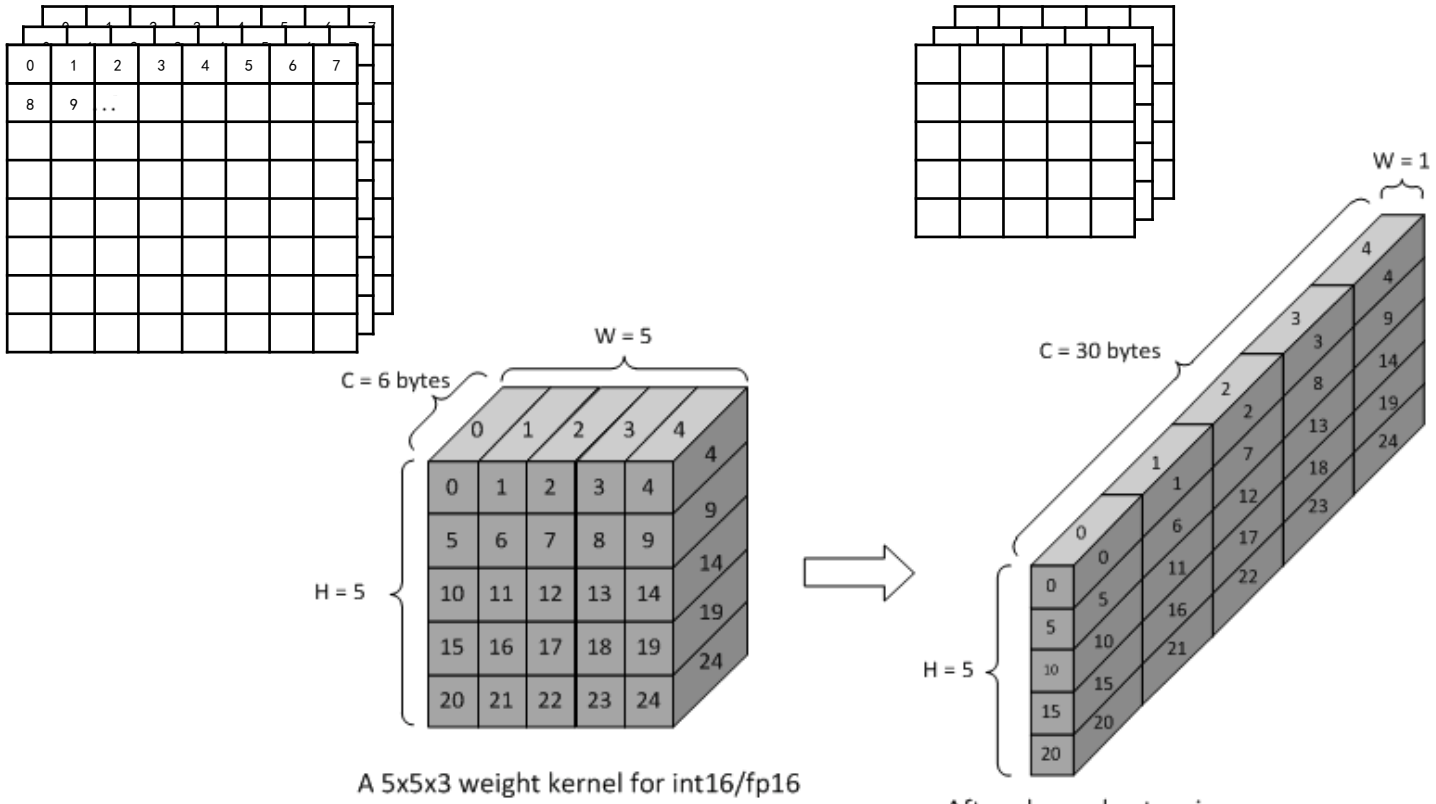
Convolution buffer entry 1

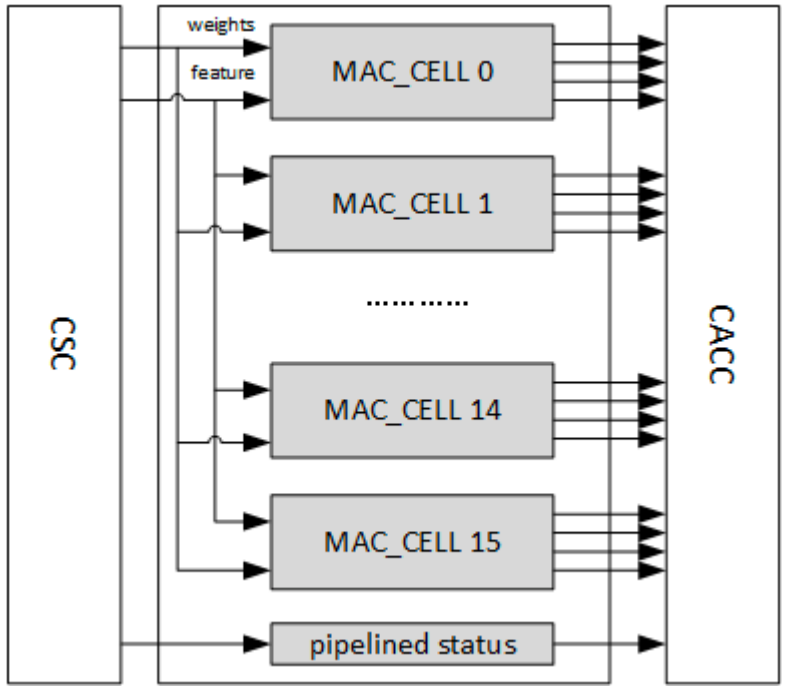


One line of 40 pixels mapped to convolution buffer
padding = 1; channel = 3;

Weight Extension

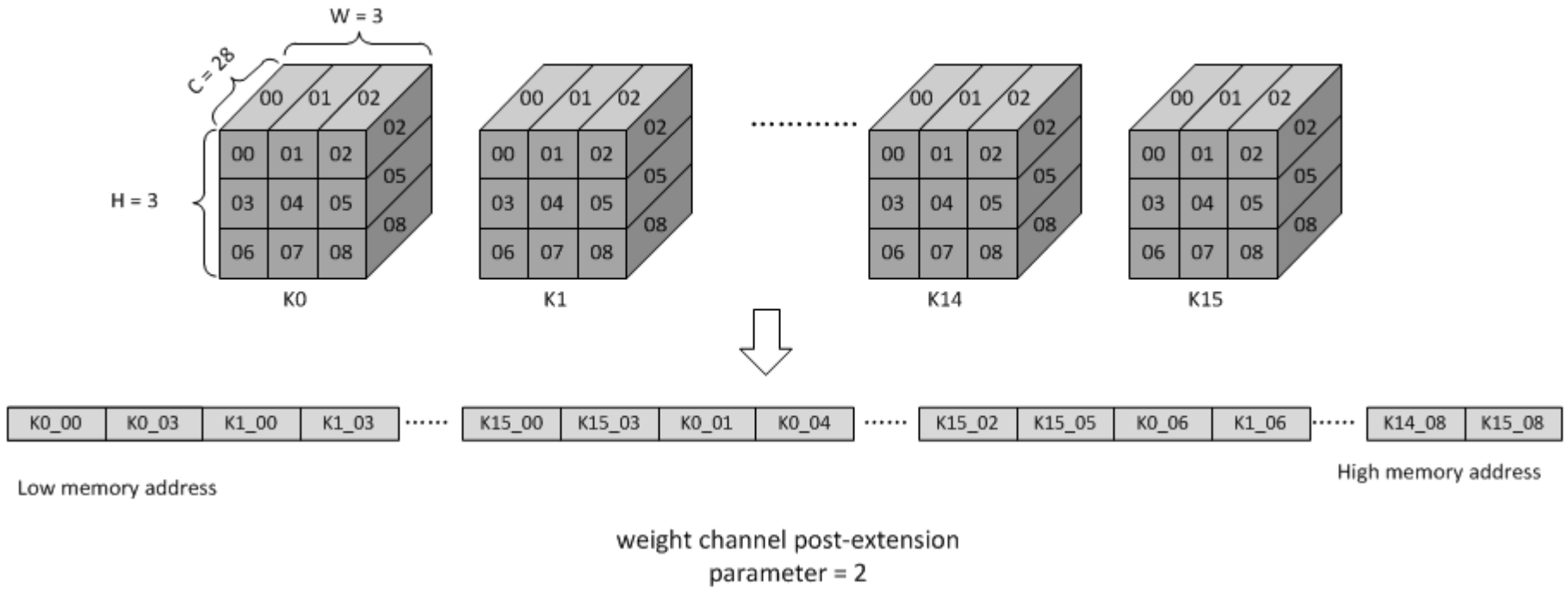
- Weight channel pre-extension





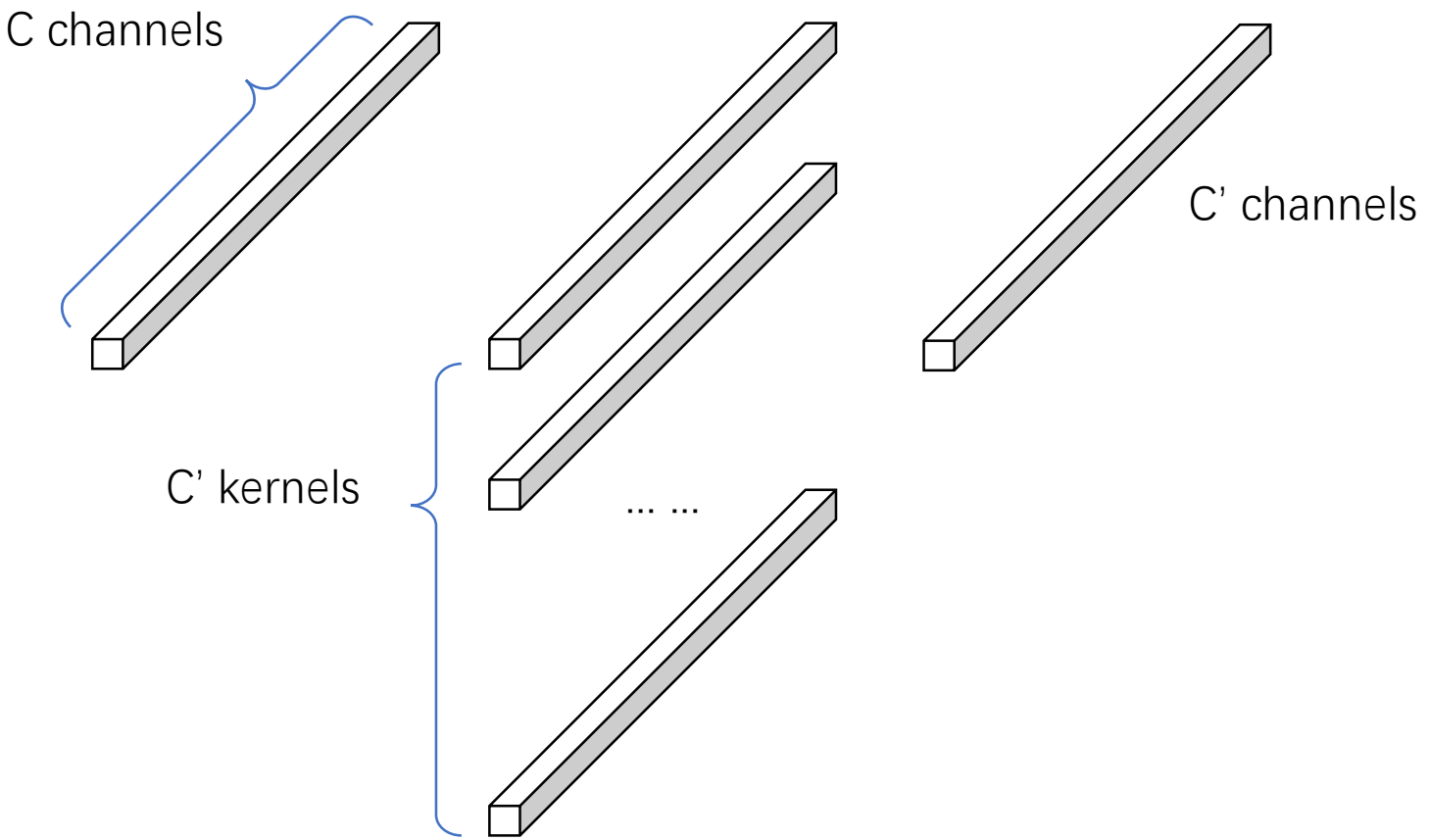
Weight Extension (Cont'd)

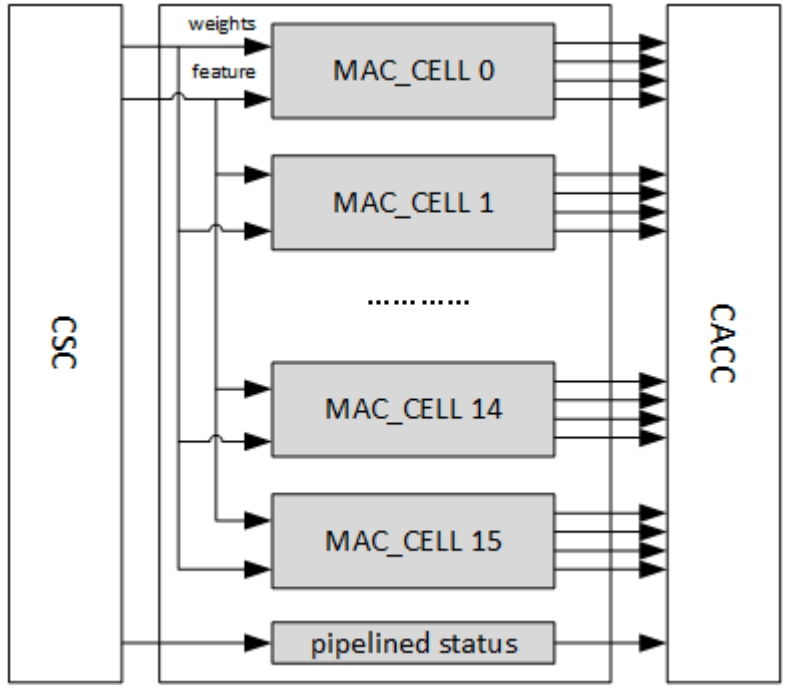
- Weight channel post-extension (1/2/4-line)



MLP Support

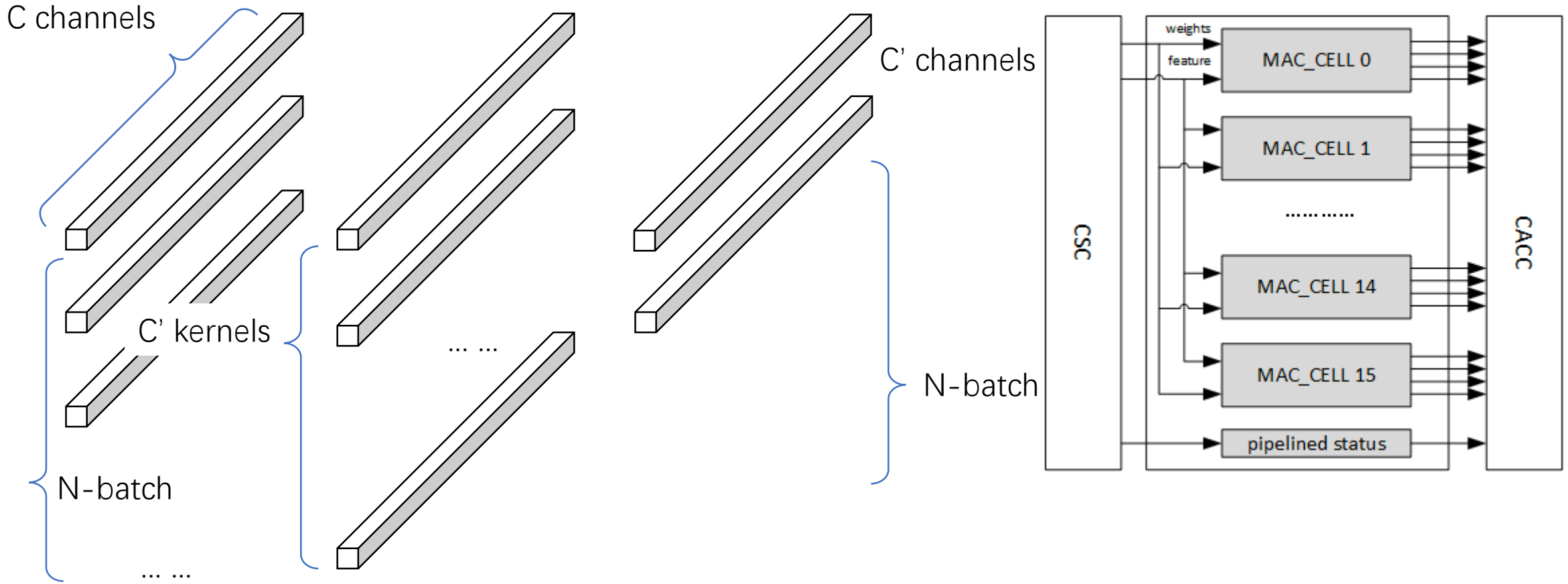
- MLP can be considered as a special convolution





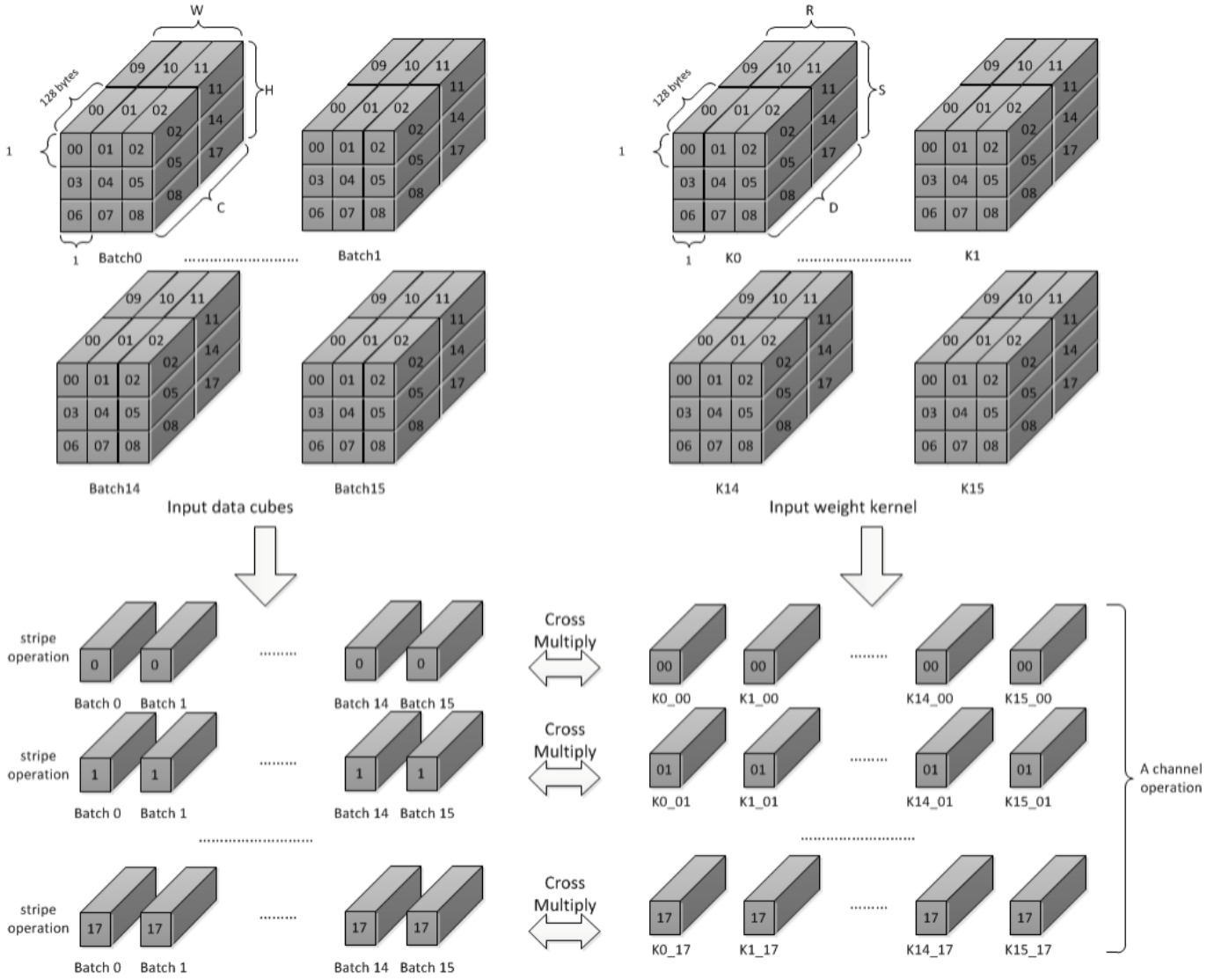
MLP Support (Cont'd)

- Batch operation can improve utilization & data reuse



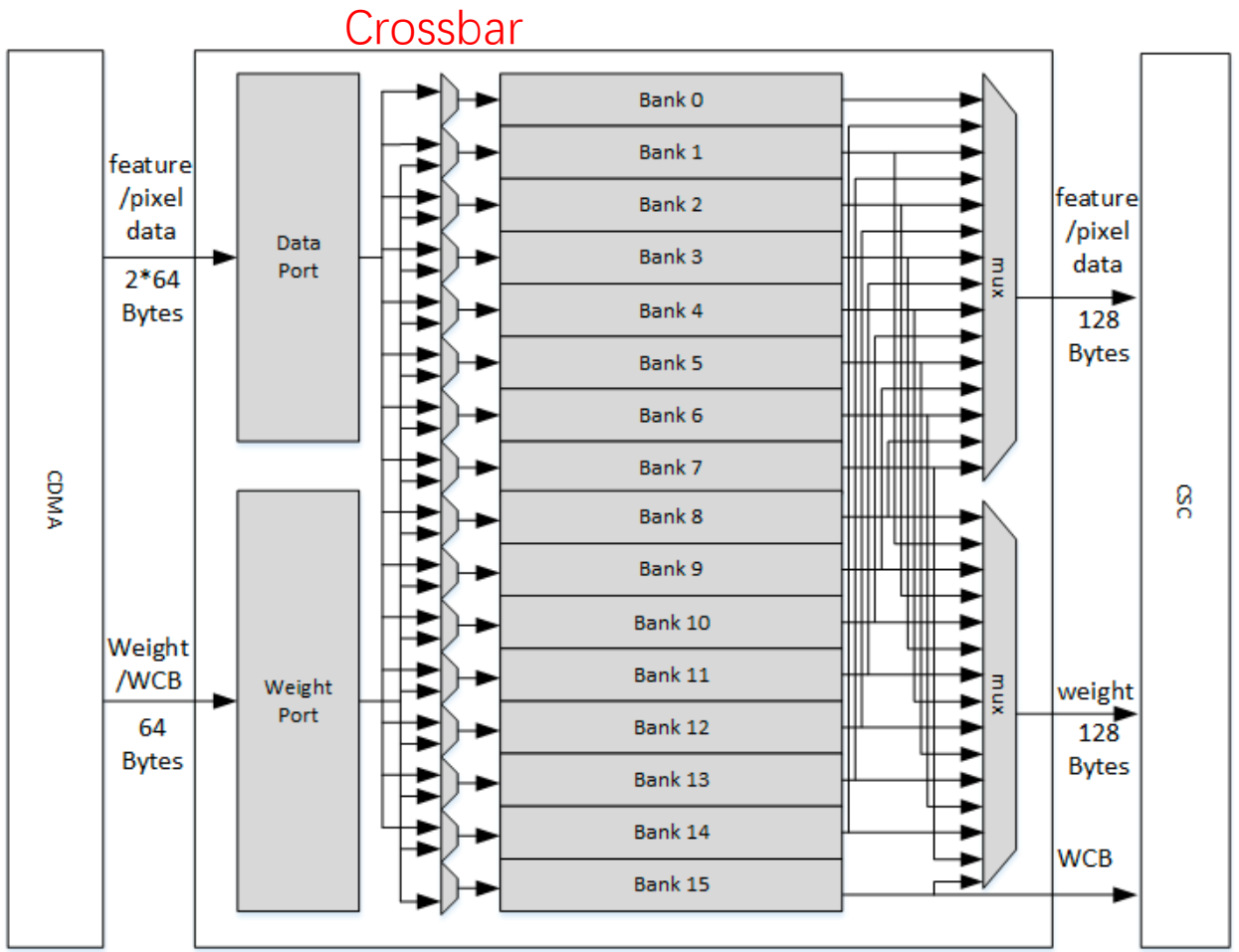
立志成才报国裕民

Multi-Batch for Convolution



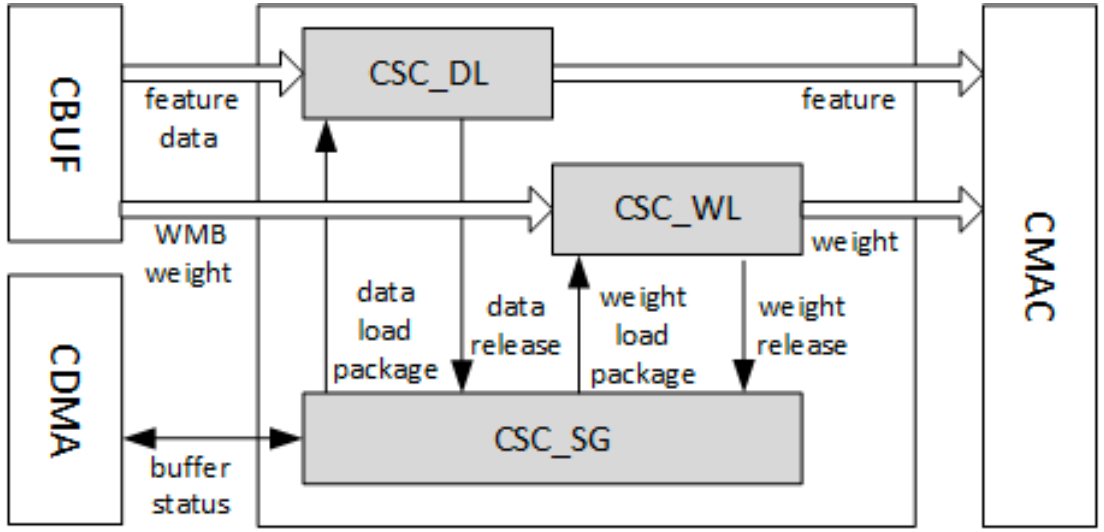
Convolution Buffer

- 16 32KB Banks, each bank has 2 512-bit wide 256-entry 2-port SRAMs
- Circular buffer



Convolution Sequence Control

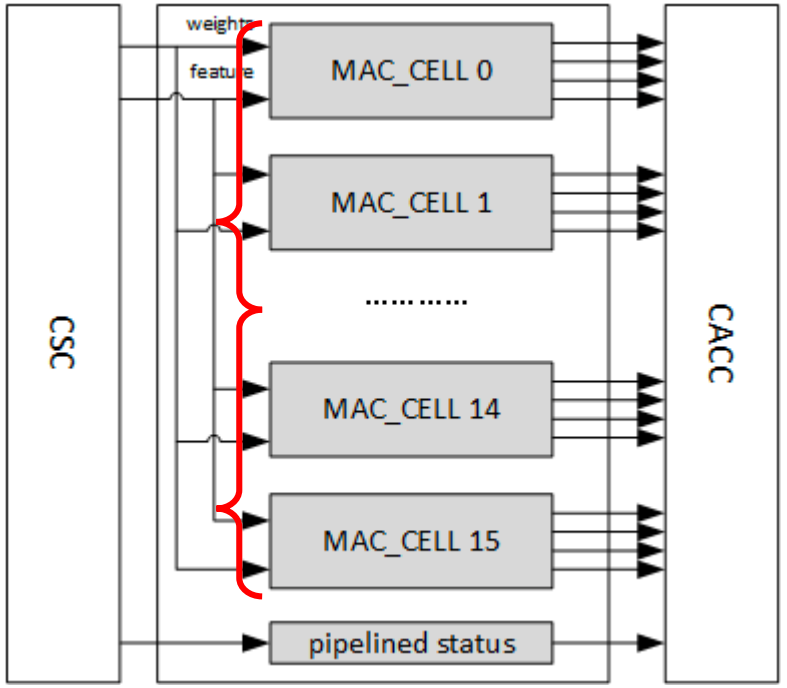
- Data package: stripe/channel size, position relative to weight as offset
- Weight package: size of weight
- Data loader also performs pre-additions for Winograd
- Weight sequentially loaded to CMAC



立志成才报国裕民

Convolution MAC

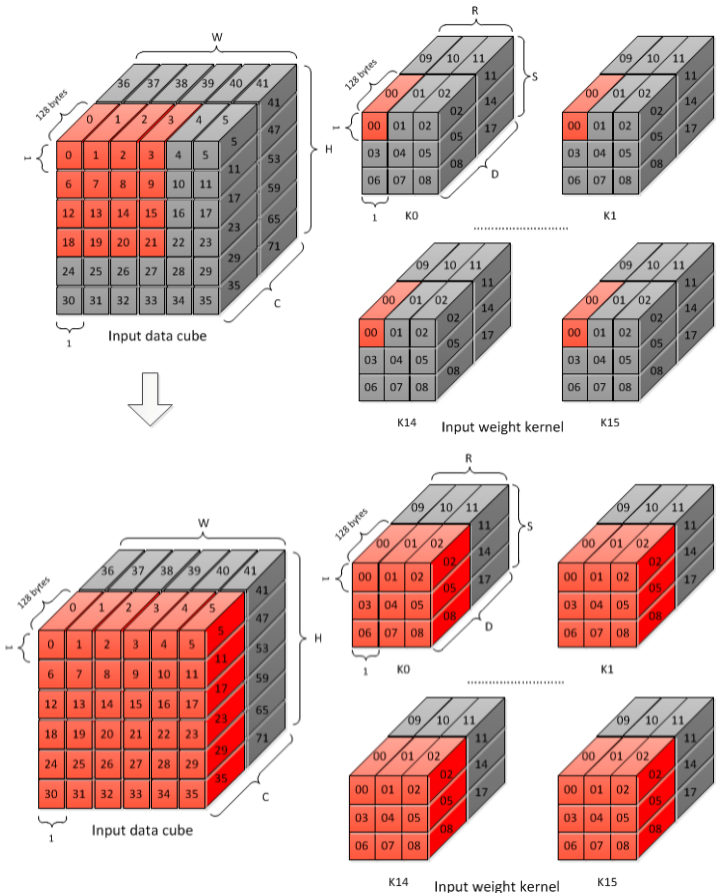
- Divided into two groups that can be configured differently
- Using pipeline
- Multiply using booth-encoding
- Adder tree with carry-save adders
- Output multiple values under only Winograd mode



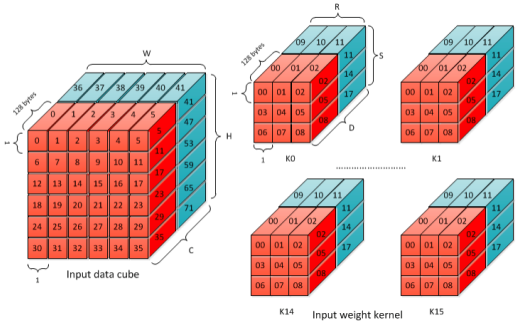
立志成才报国裕民

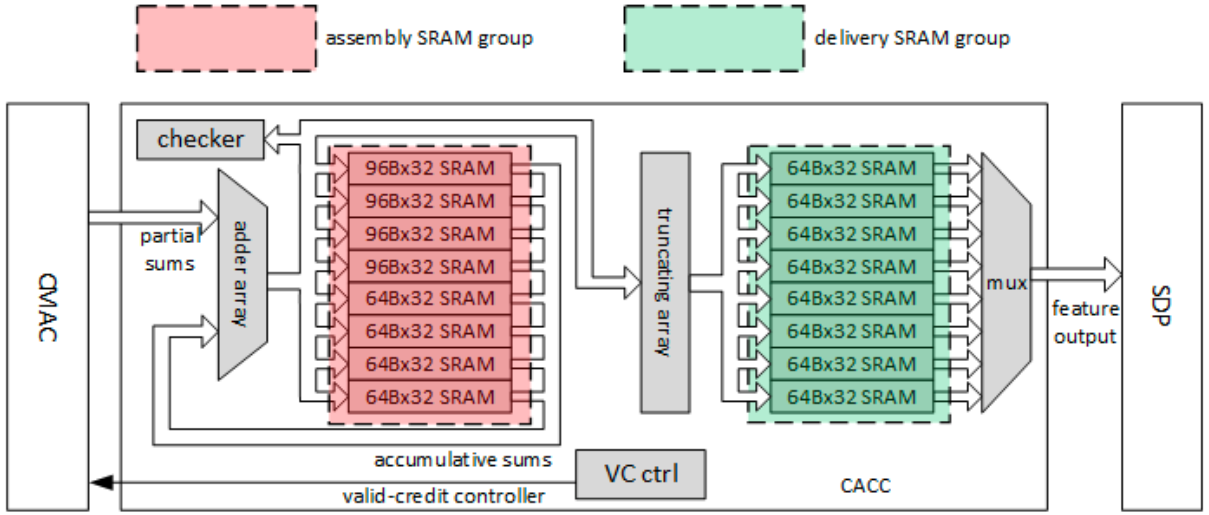
Convolution Accumulator

- 64 48-bit adders as an adder array
- Activates adder 0~15 for INT16 direct convolution



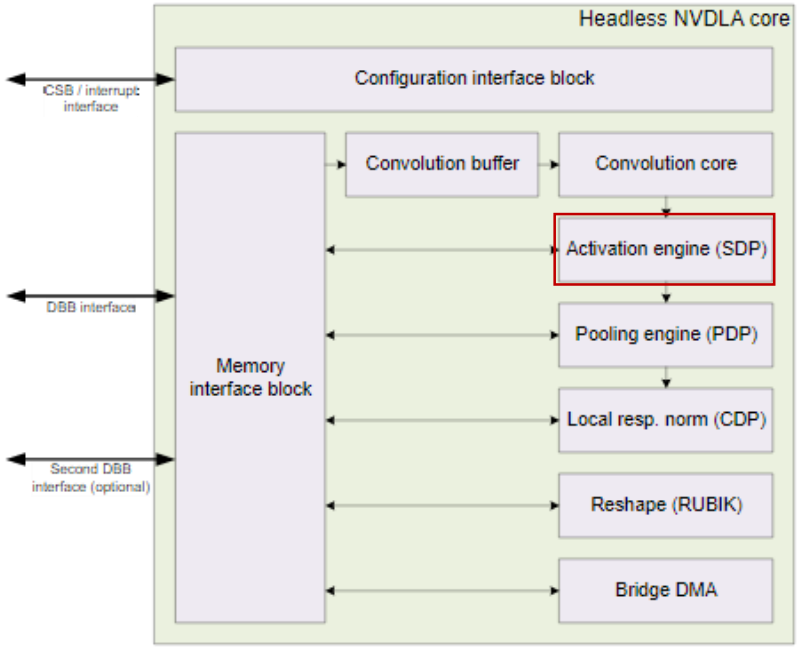
- INT32 after truncation
- Output 32 elements per cycle





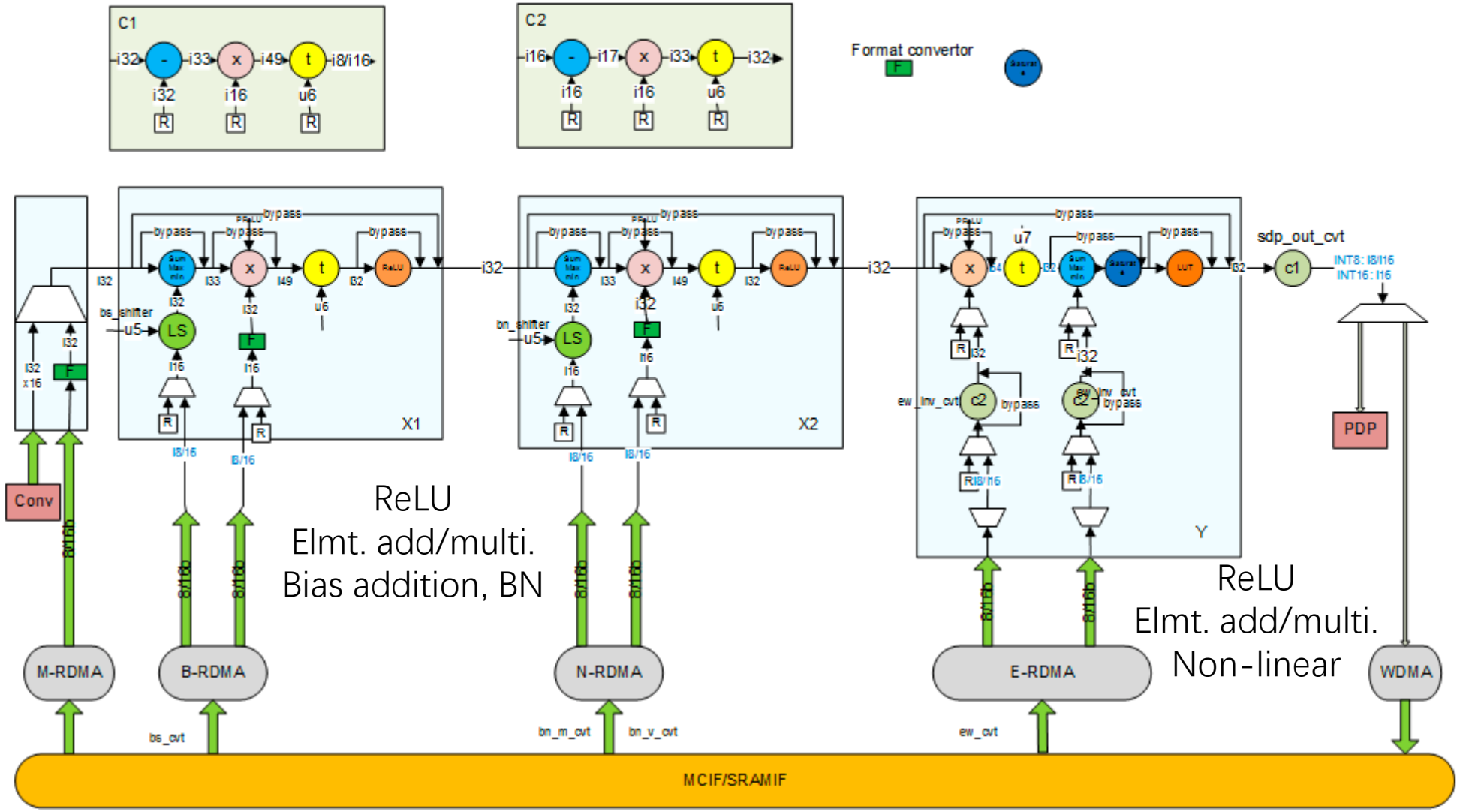
Single Point Data Processor (SDP)

- Add bias
- Activation function (piece-wise linear interpolation)
- Batch norm.
- Data format converter.
- Can choose data source by a control register



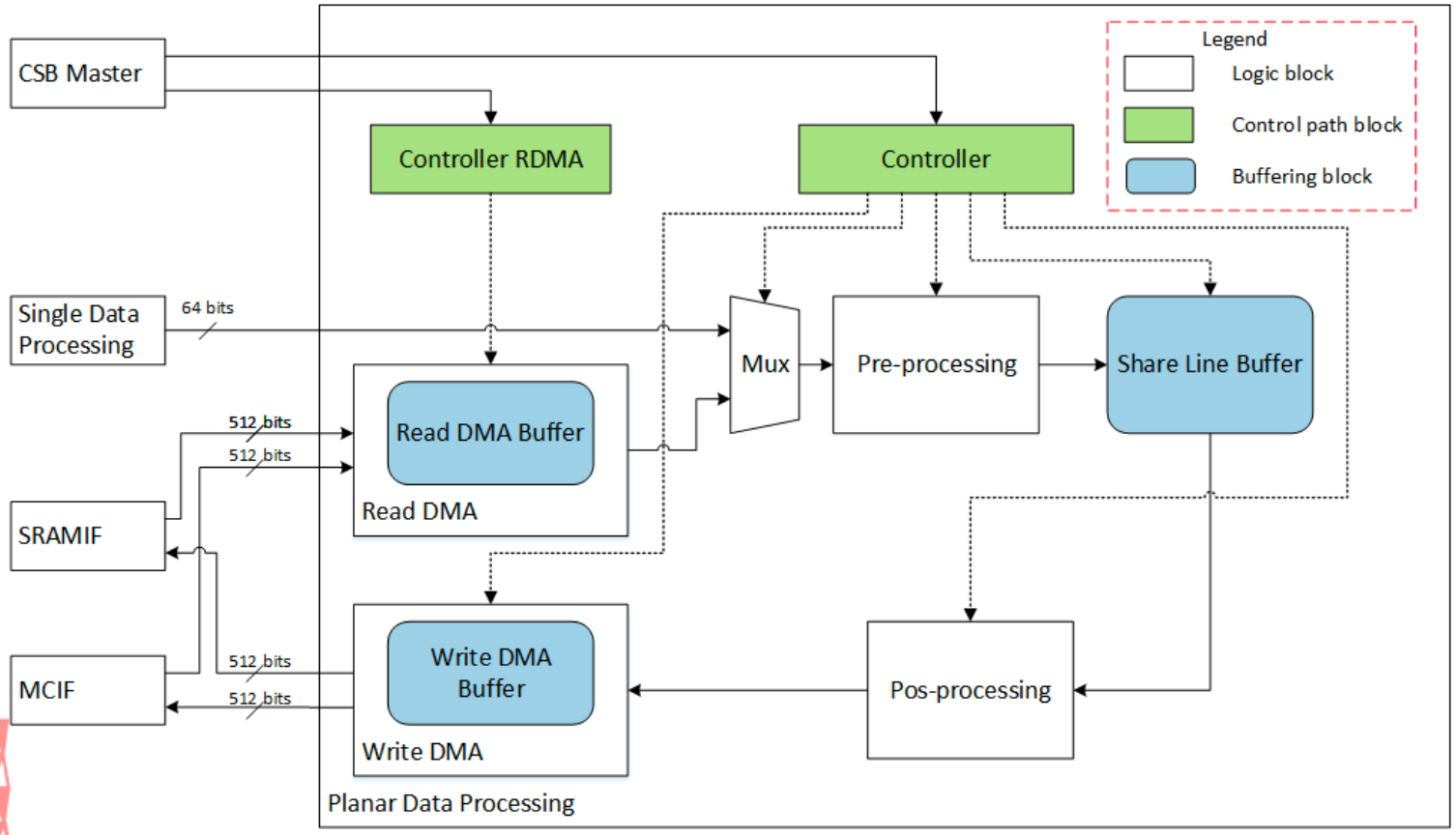
立志成才报国裕民

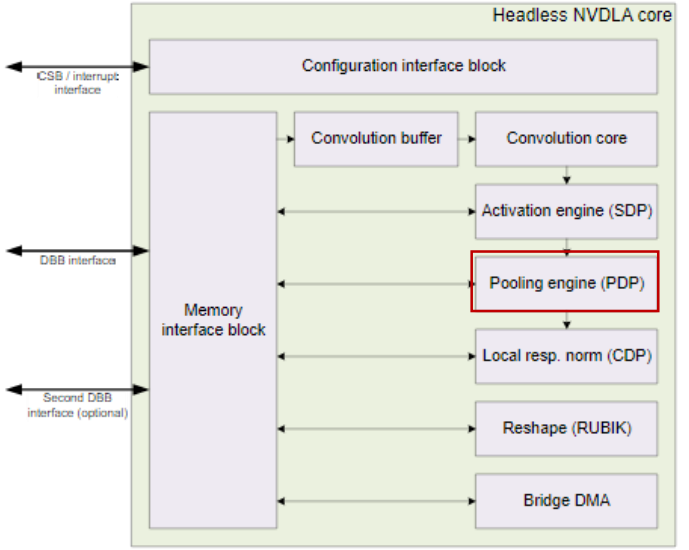
SDP (Cont'd)


C1
C2
Format convertor
X1
X2
Y
Conv
M-RDMA
B-RDMA
N-RDMA
E-RDMA
WDMA
PDP
MCIF/SRAMIF
sdp_out_cvt
ReLU
Elmt. add/multi.
Bias addition, BN
ReLU
Elmt. add/multi.
Non-linear

Planar Data Processor (PDP)

- Pooling layers (min/max/average)

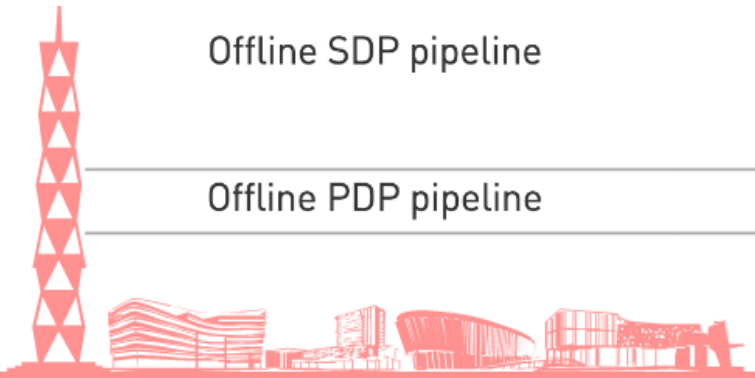




Pipeline of NVDLA Core

- Different processors can be combined for different types of layers

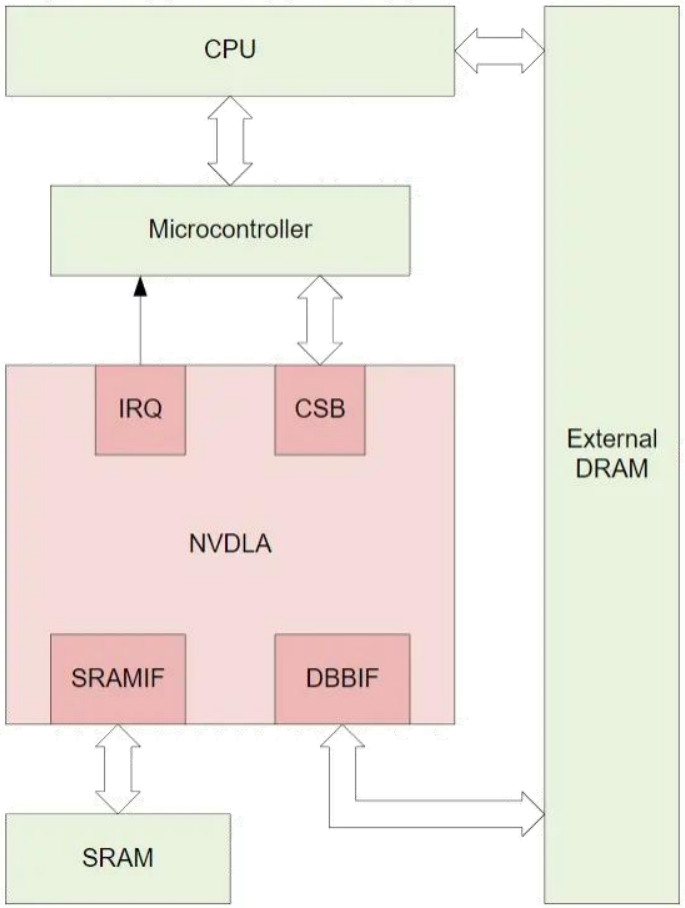
Pipeline Working Mode	Sub Units	Interface	Support Layers
Convolution pipeline	CDMA, CBUF, CSC, CMAC, CACC, SDP	MCIF/SRAMIF	direct convolution, Winograd, image input layer, element-wise layer, batch-normaliza tion, activation (relu, prelu, sigmoid, etc.) layer
Convolution & pooling pipeline	CDMA, CBUF, CSC, CMAC, CACC, SDP, PDP	MCIF/SRAMIF	direct convolution + pooling layer (multi-batch mode is not supported)
Offline SDP pipeline	SDP	MCIF/SRAMIF	activation layer, element-wise layer, batch-normaliza tion layer, format conversion layer, comparison layer
Offline PDP pipeline	PDP	MCIF/SRAMIF	Pooling layer

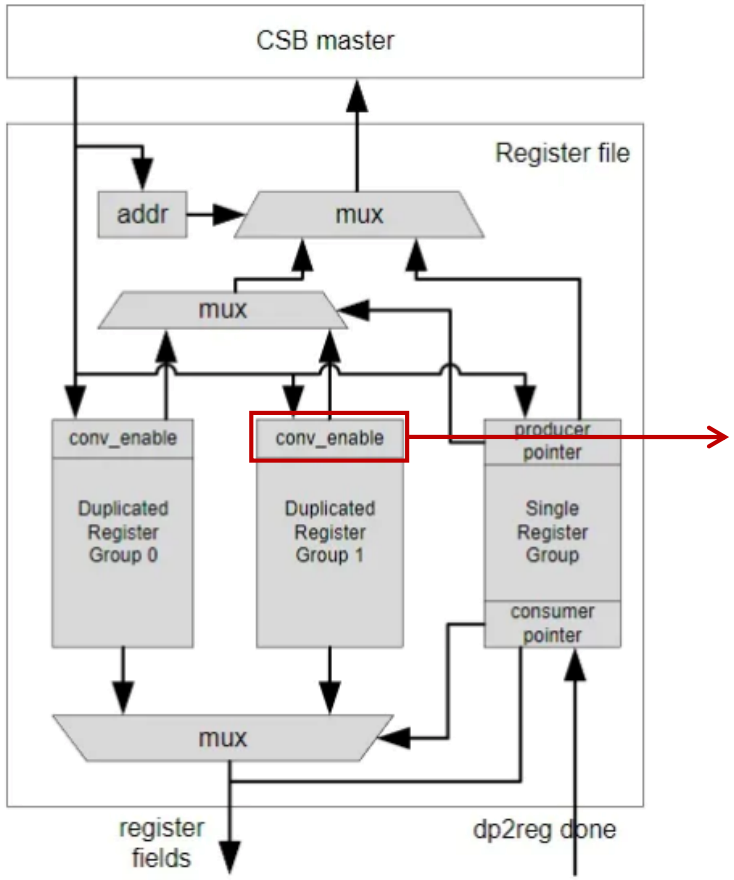


立志成才报国裕民

Control of NVDLA

- CSB ping-pong registers

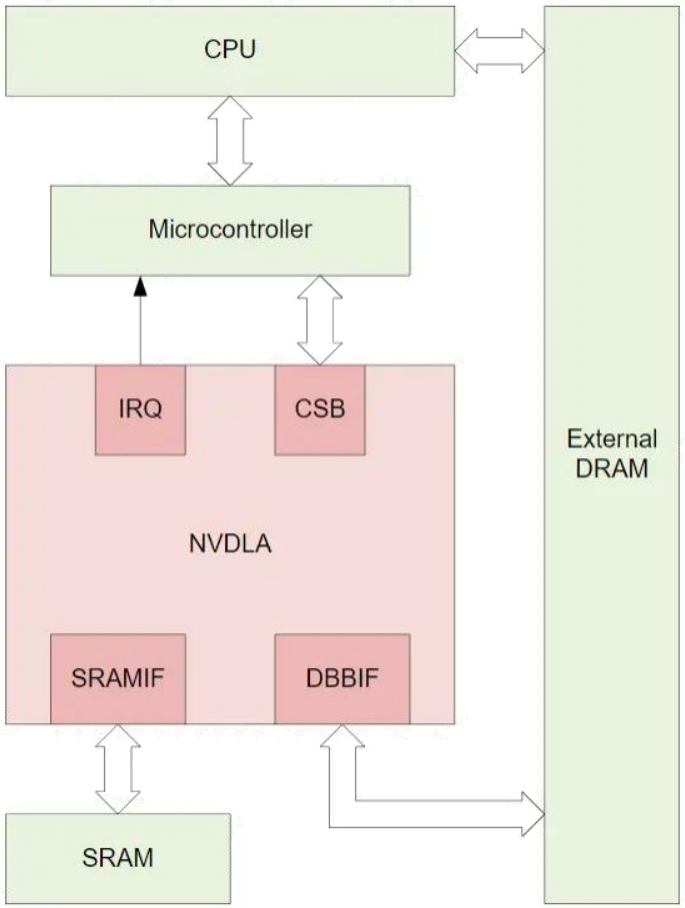


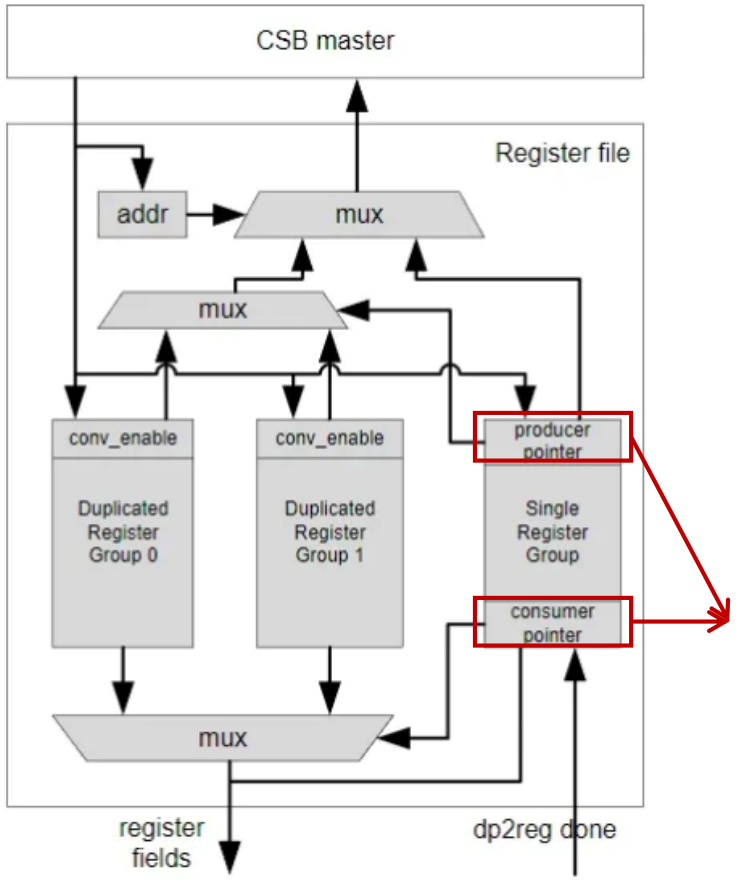


CPU/master set;
NVDLA reset;

Control of NVDLA (Cont'd)

- CSB ping-pong registers

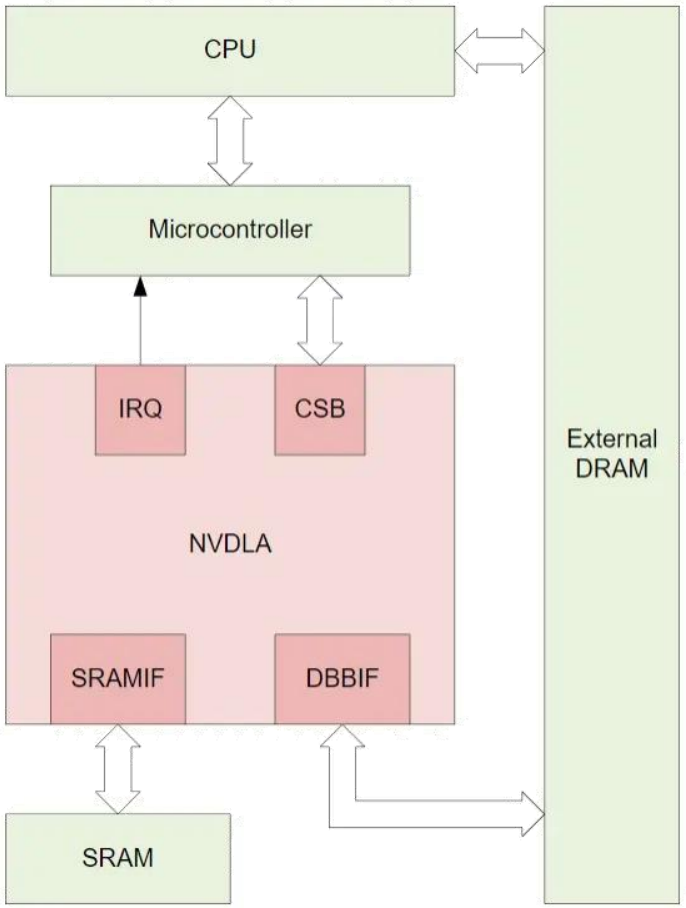


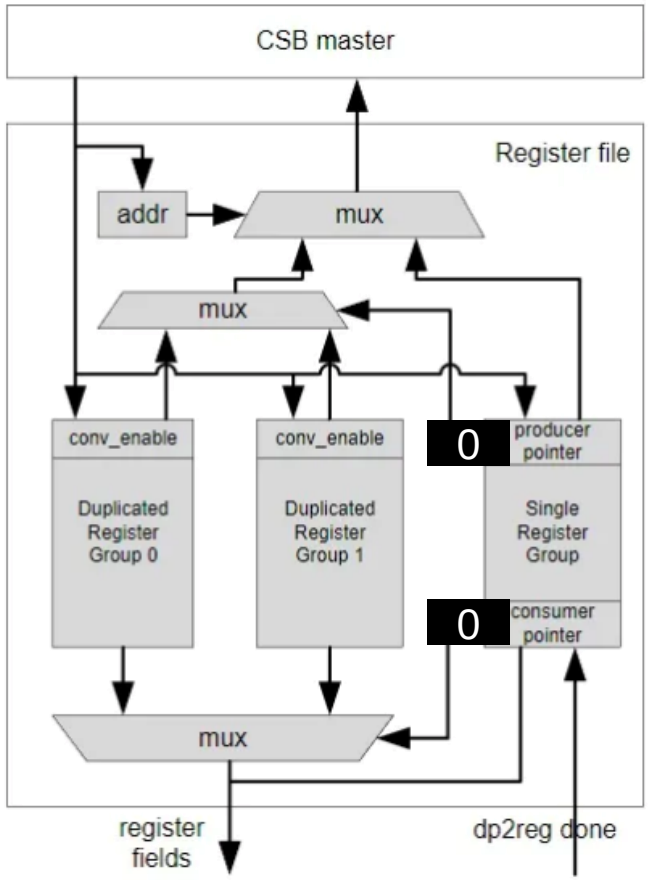


CPU set pointers

Control of NVDLA (Cont'd)

- CSB ping-pong registers



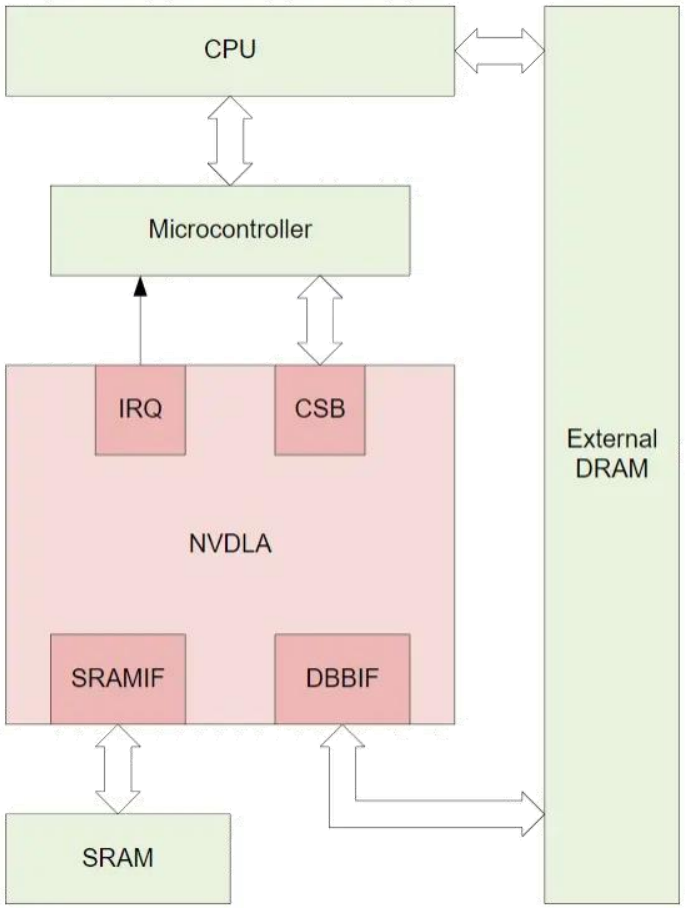


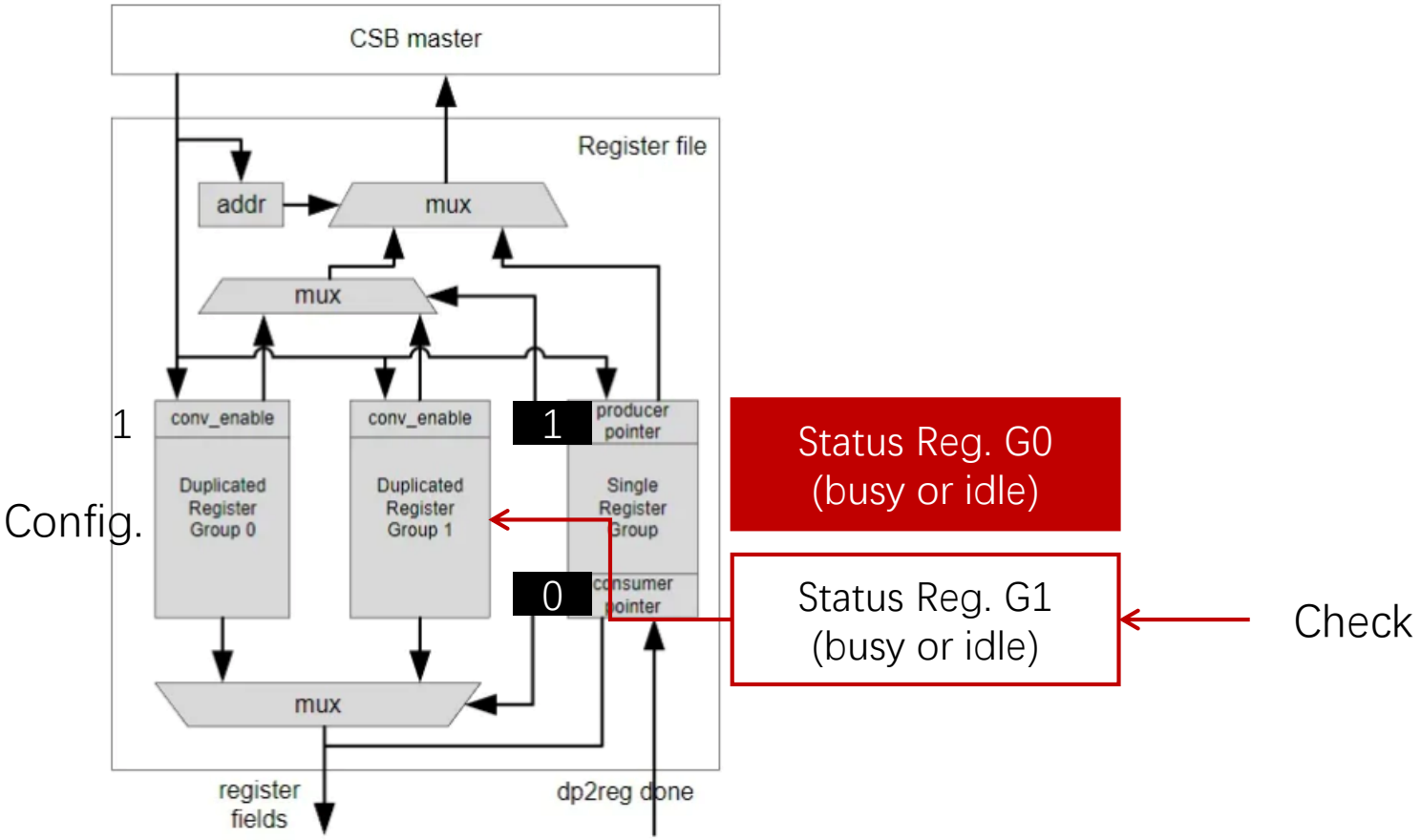
Status Reg. G0 (busy or idle)

Status Reg. G1 (busy or idle)

Control of NVDLA (Cont'd)

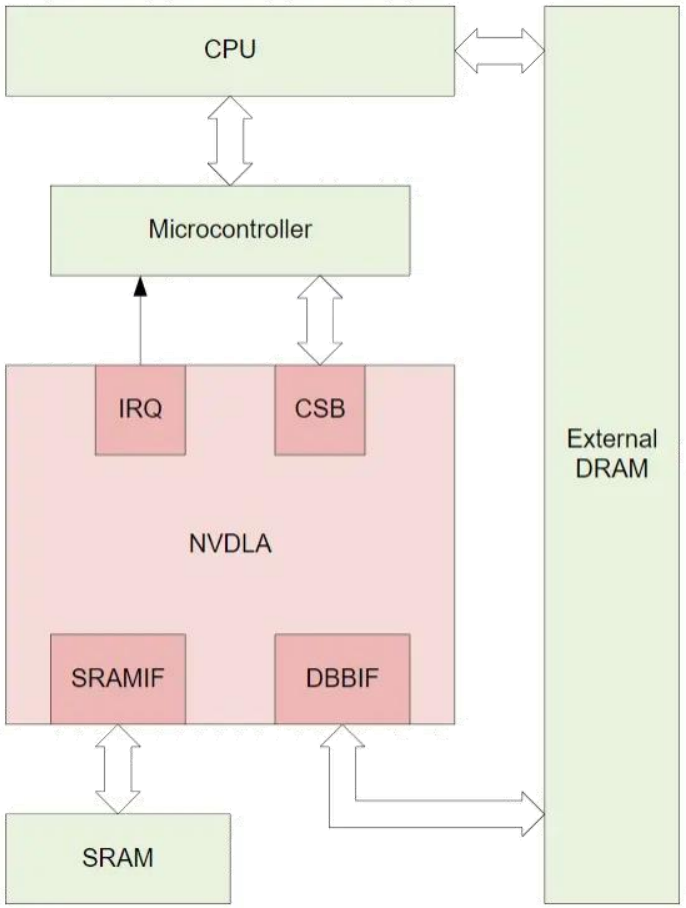
- CSB ping-pong registers

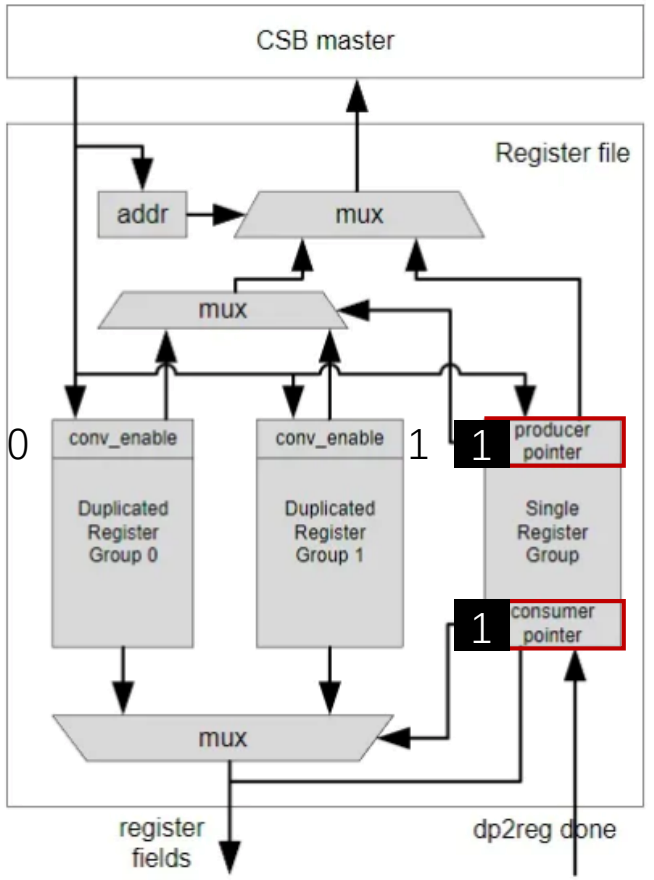




Control of NVDLA (Cont'd)

- CSB ping-pong registers



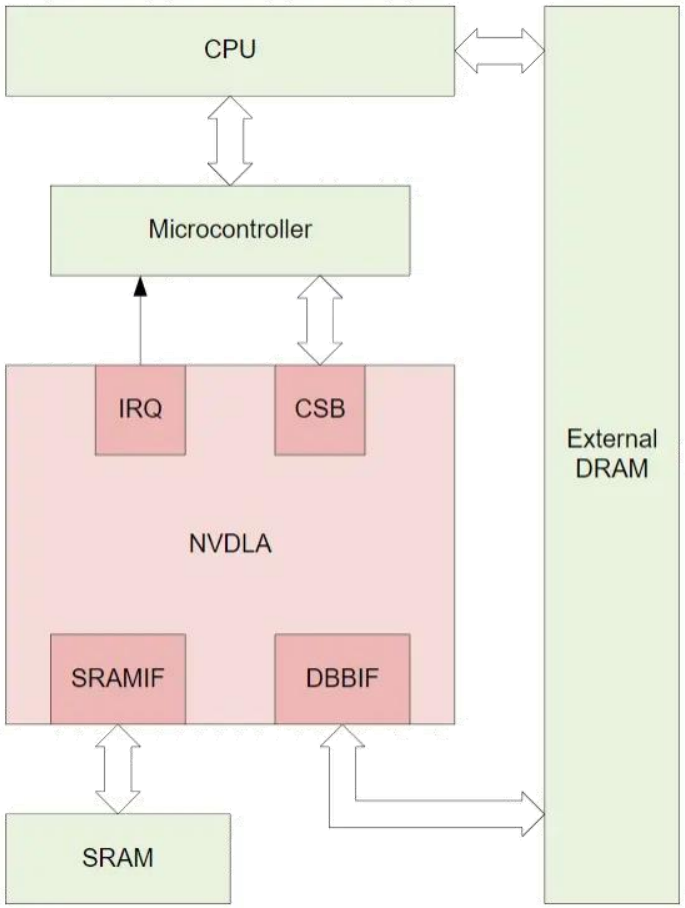


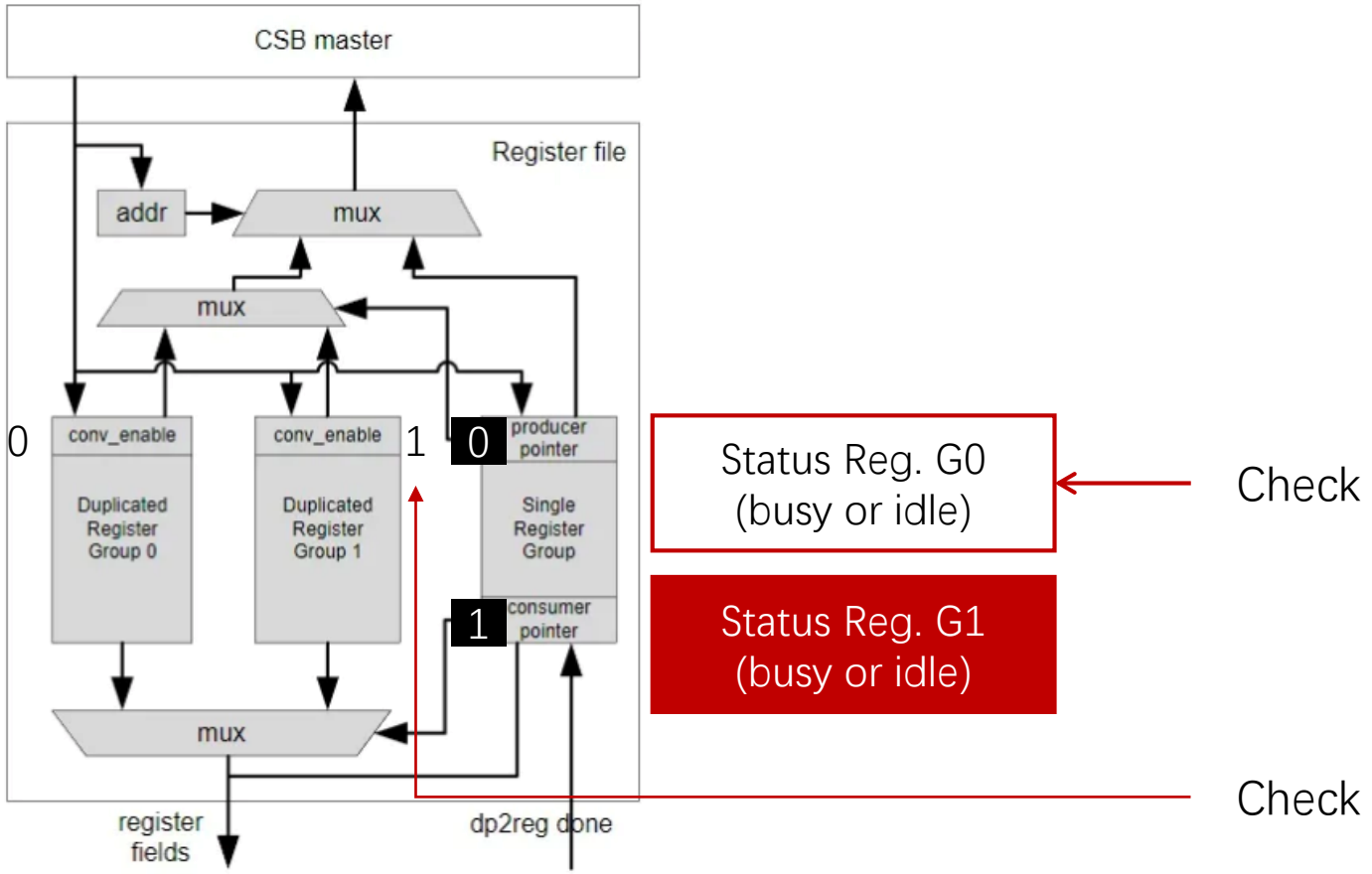
Status Reg. G0 (busy or idle) ← Done, set

Status Reg. G1 (busy or idle)

Control of NVDLA (Cont'd)

- CSB ping-pong registers

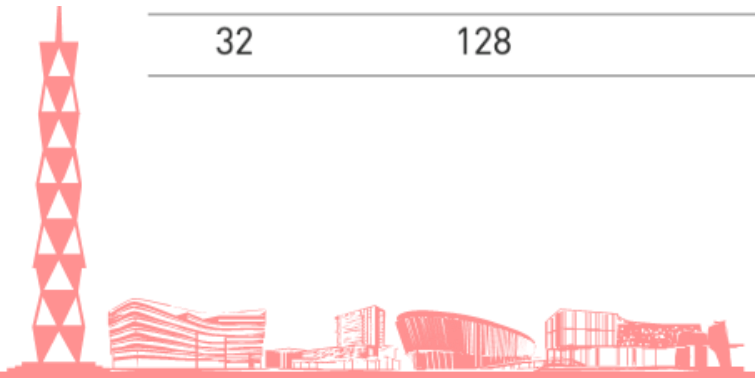




System Tuning

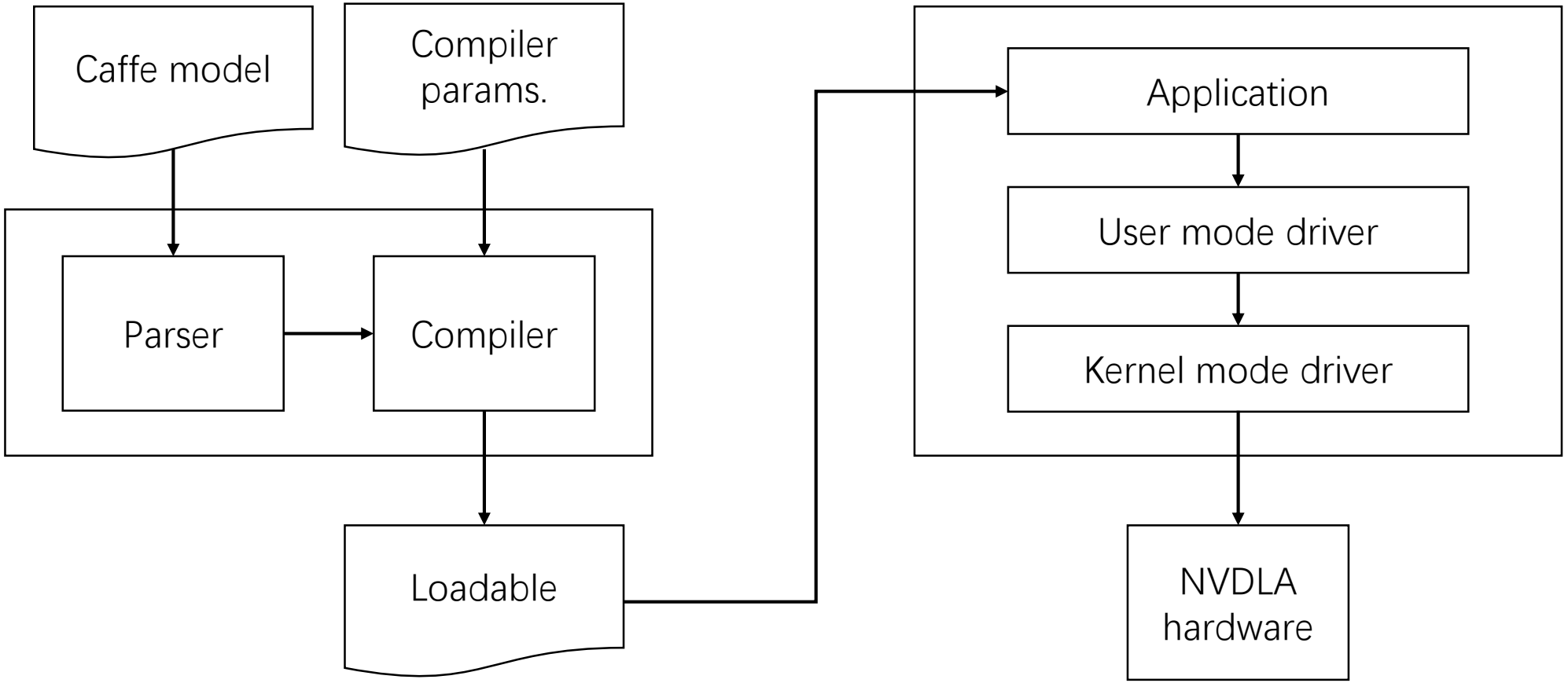
Example Area and Performance with NVDLA

# MACs	Conv. buffer size (KB)	SDRAM bandwidth (GB/s)	Silicon Cell Area (mm^2, 28nm)	Silicon Cell Area (mm^2, 16nm)	Int8 ResNet-50 (frames/sec)	Power Estimate Peak/Average (mW, 16nm)
2048	512	20	5.5	3.3	269	766 / 291
1024	256	15	3.0	1.8	153	375 / 143
512	256	10	2.3	1.4	93	210 / 80
256	256	5	1.7	1.0	46	135 / 48
128	256	2	1.4	0.84	20	82 / 31
64	128	1	0.91	0.55	7.3	55 / 21
32	128	0.5	0.85	0.51	3.6	45 / 17



立志成才报国裕民

Software Stack



More at https://nvdla.org/sw/contents.html

Read More

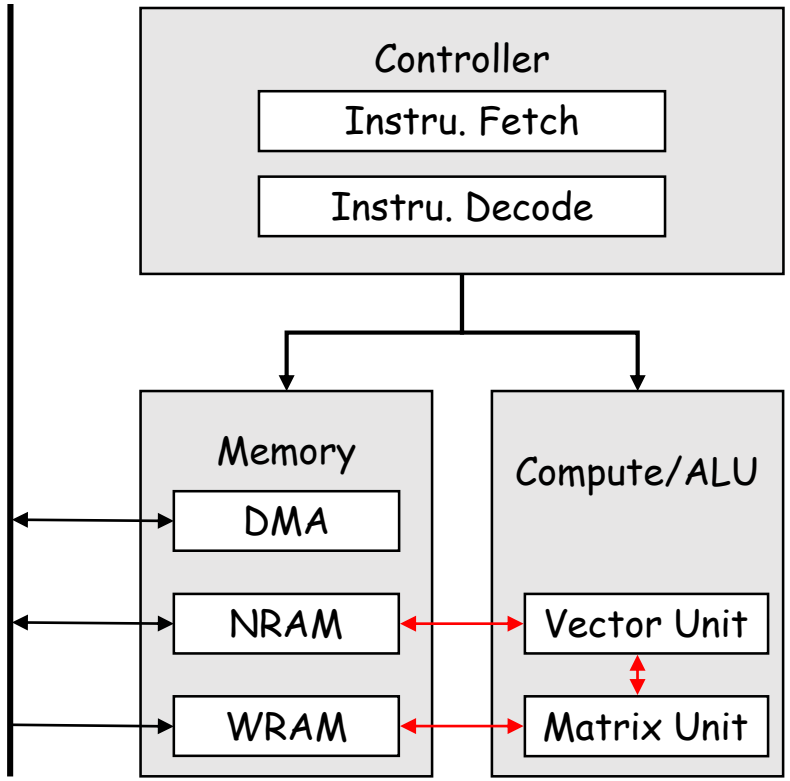
- Read more on NVIDIA Deep Learning Accelerator

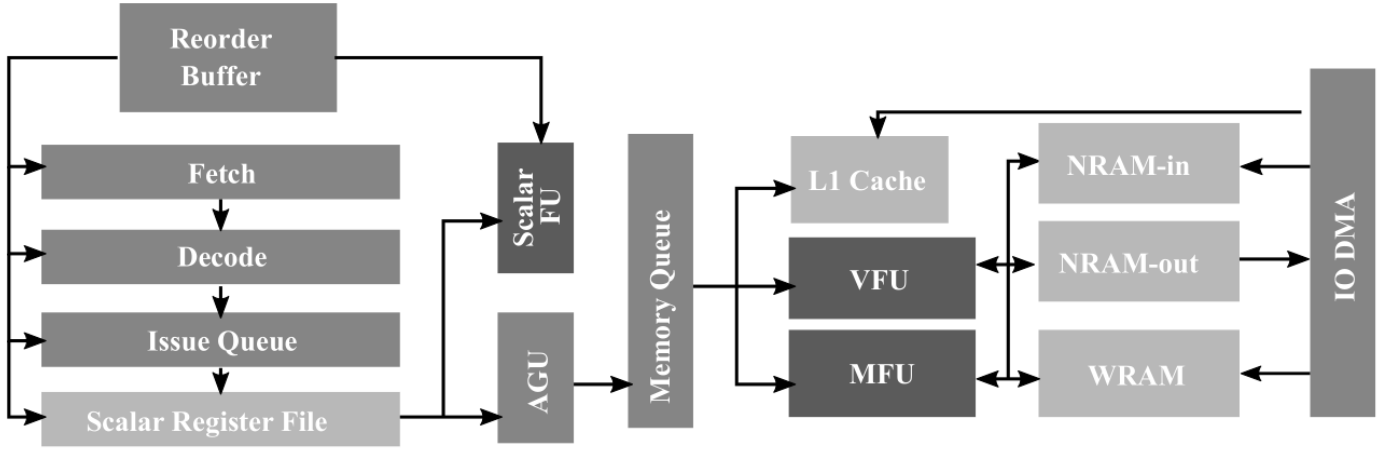


立志成才报国裕民

Multi-Core NN Accelerators

- Single core DPL design



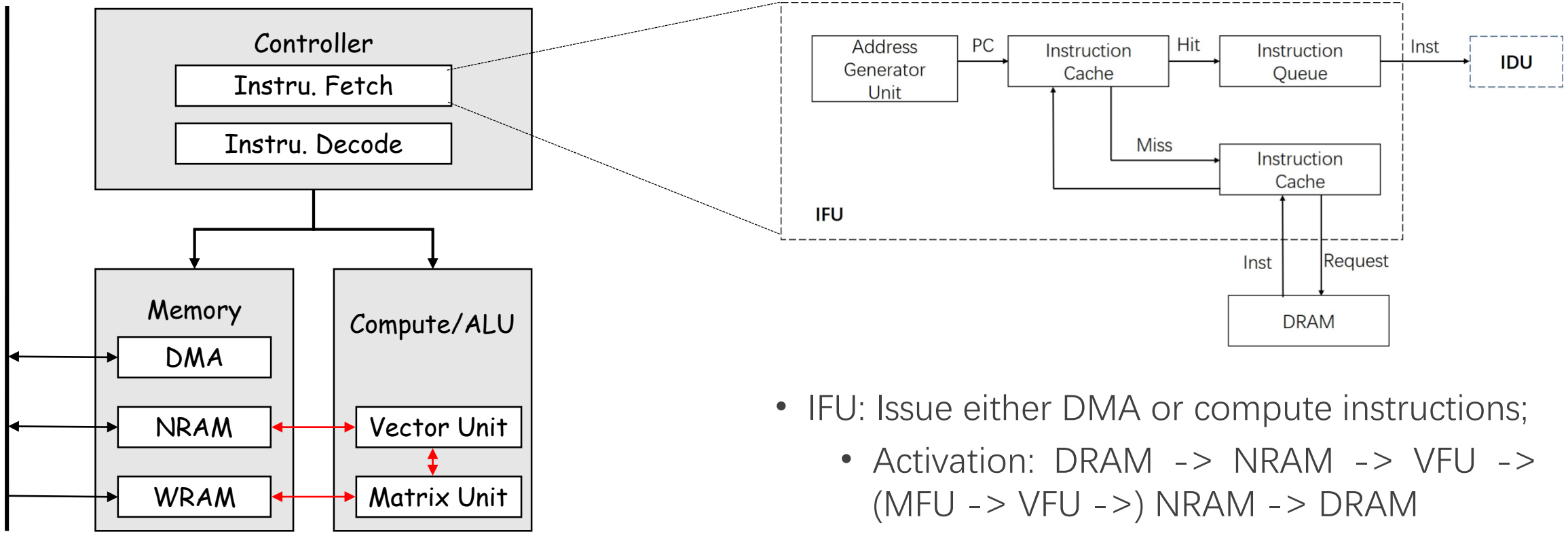


- Scalar datapath for general purpose computing
- Added AGU/VFU/MFU for tensor operations

立志成才报国裕民

Single-Core NN Accelerators

- Single core DPL design

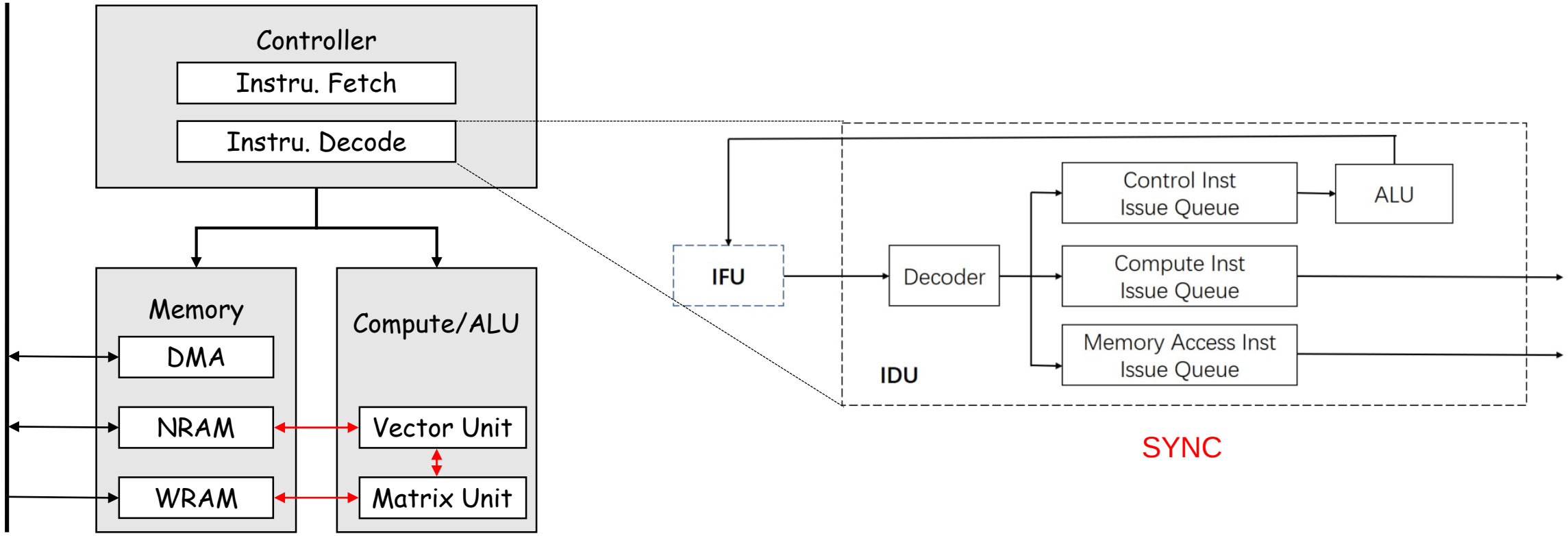


- IFU: Issue either DMA or compute instructions;
  - Activation: DRAM -> NRAM -> VFU -> (MFU -> VFU ->) NRAM -> DRAM
  - Weights: DRAM -> WRAM -> MFU

立志成才报国裕民

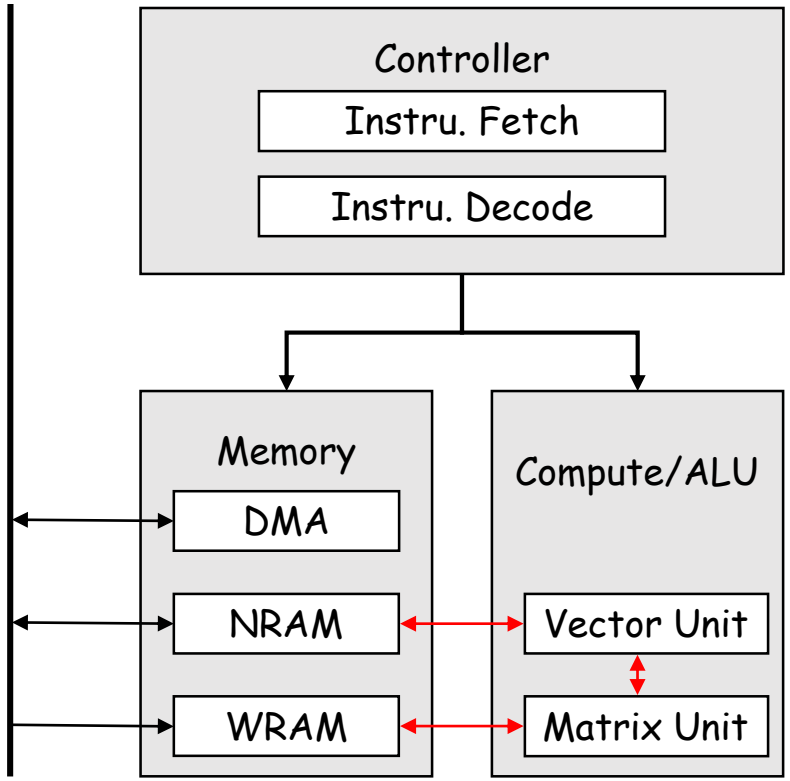
Multi-Issue

- Single core DPL design

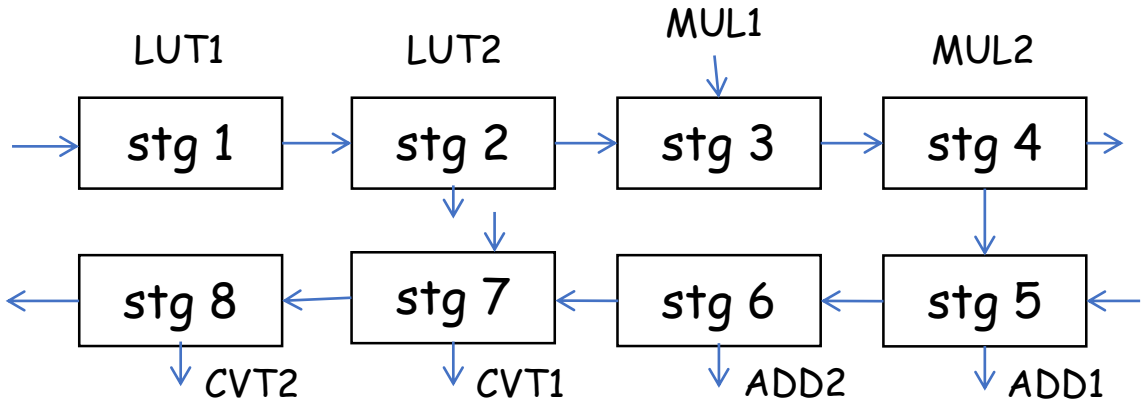


Vector Unit

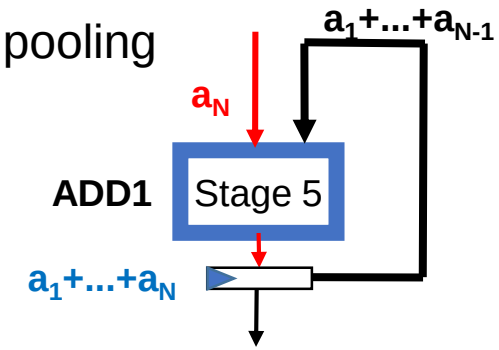
- Single core DPL design



Multi-stage vector unit pipeline

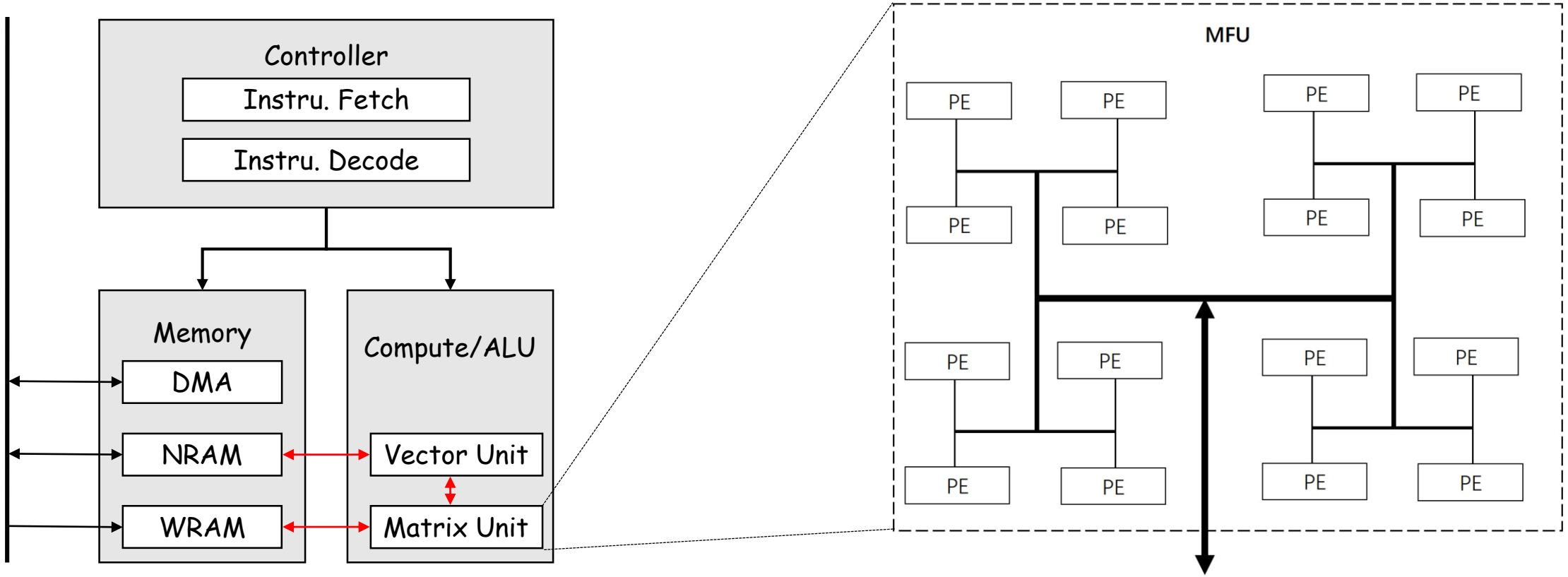


Avg. pooling



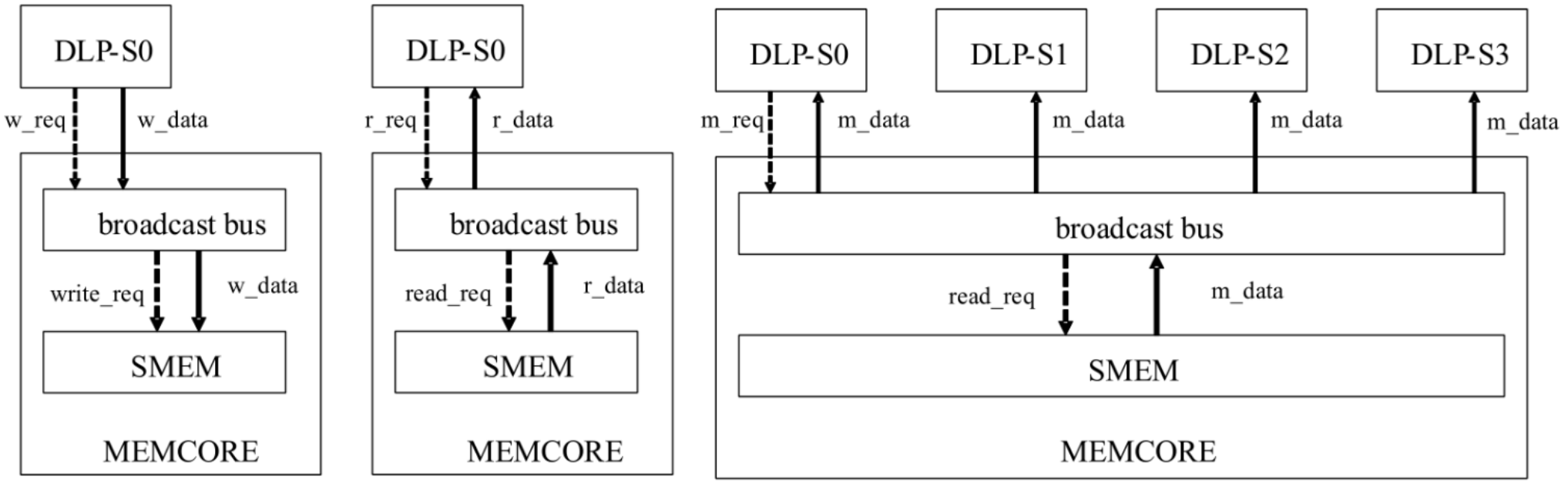
Matrix Unit

- Single core DPL design



Data Exchange (Multi-Core)

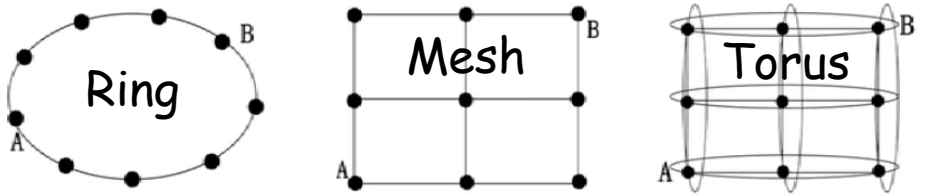
- Broadcast bus can multicast/unicast

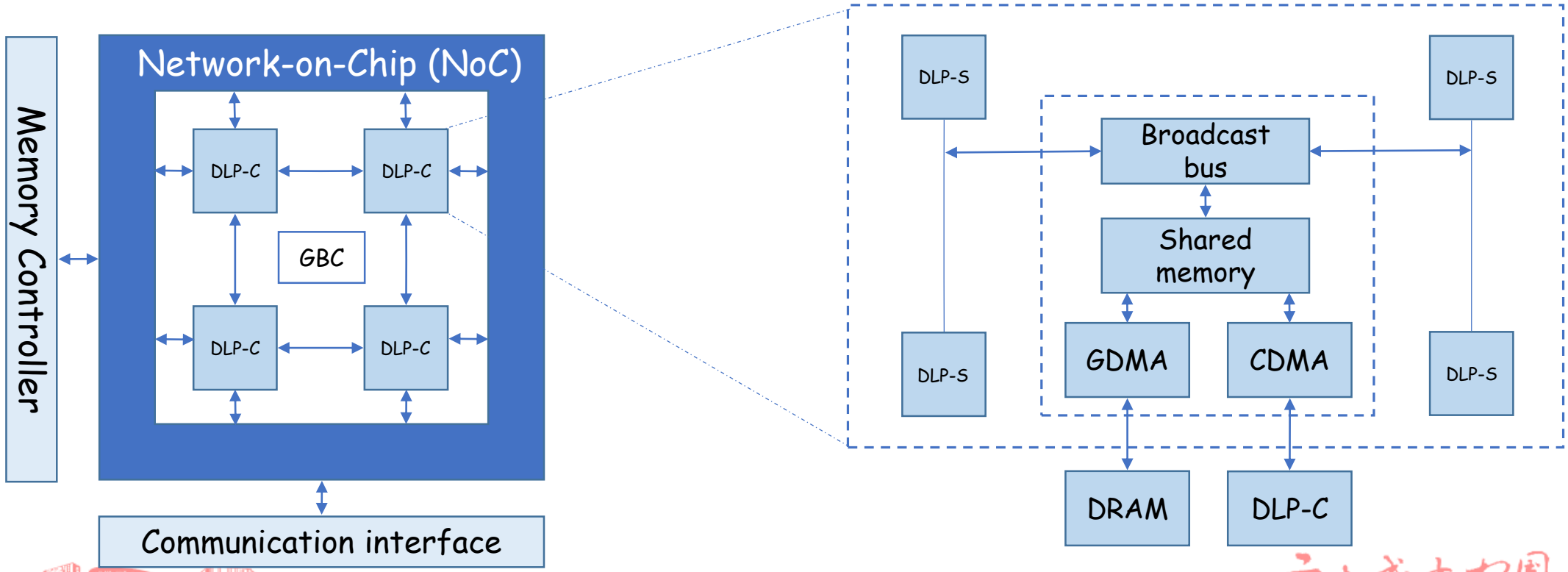


立志成才报国裕民

Multi-Core Network Topology

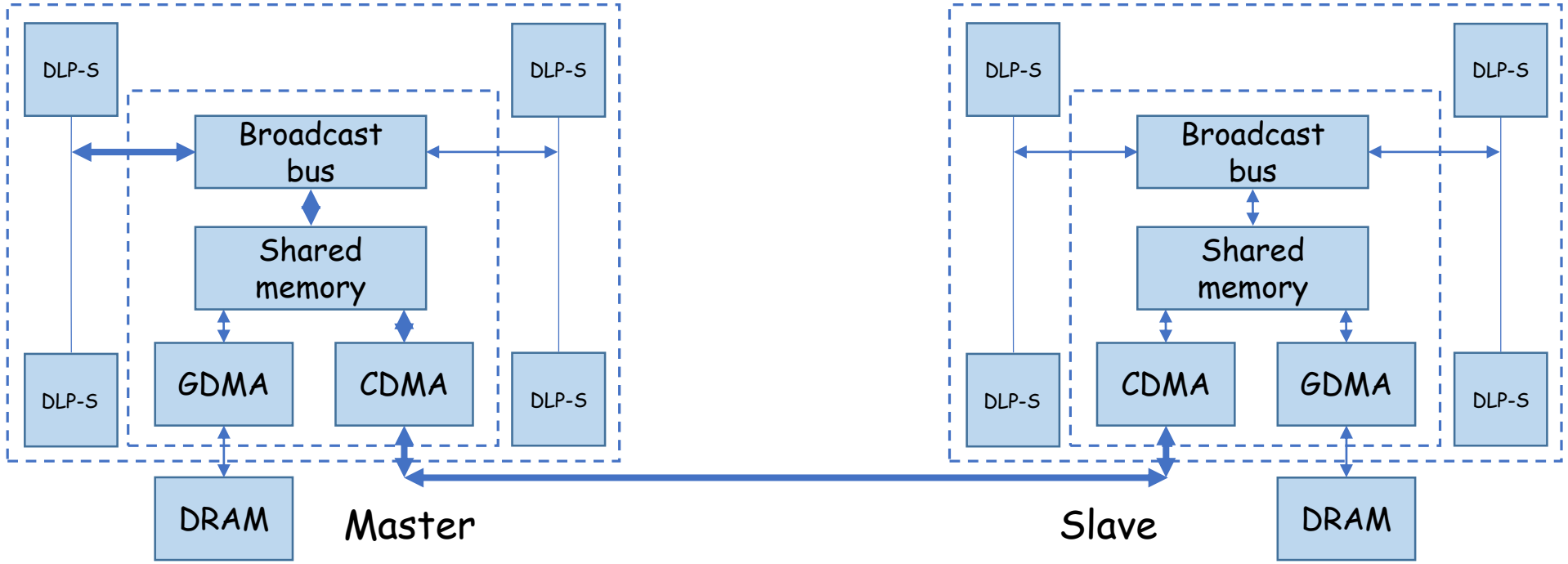
- Hierarchical topology (GPU-like), reduce NoC workload
- NoC (topology) vs. Data bus
- Communication: between DLP-S/DLP-C
- Memory access





Communication Between DLP-C

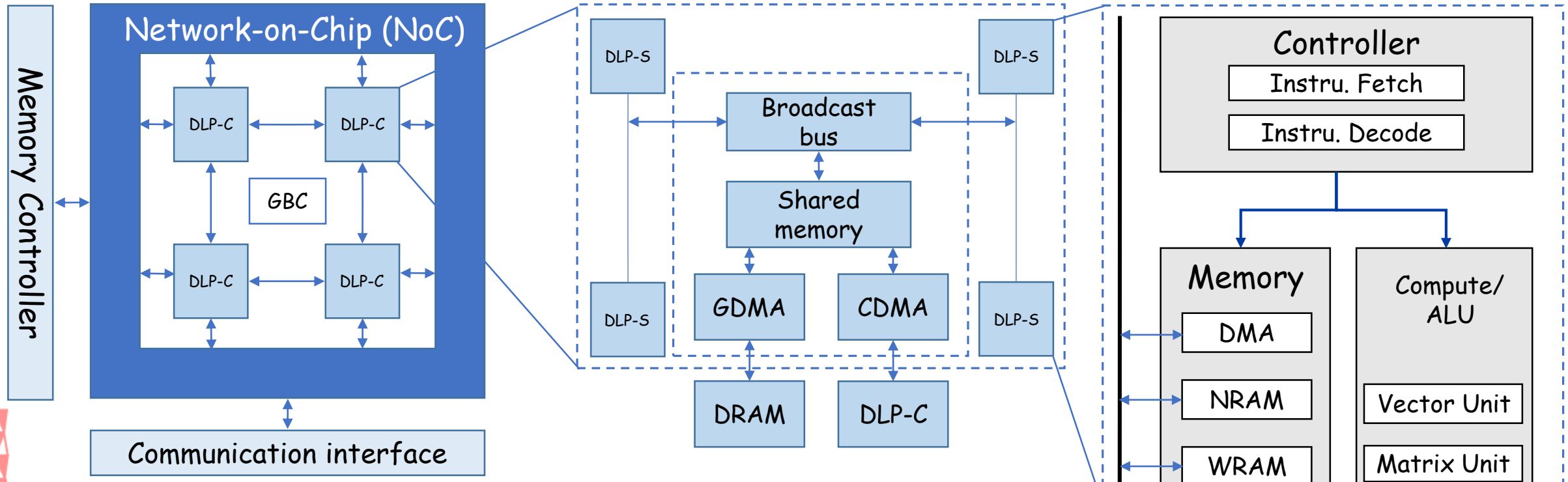
- Cluster DMA, big difference with GPU



data_access_instru. contains dest_cluster_number, src_address, dest_address, data_size

Working Example

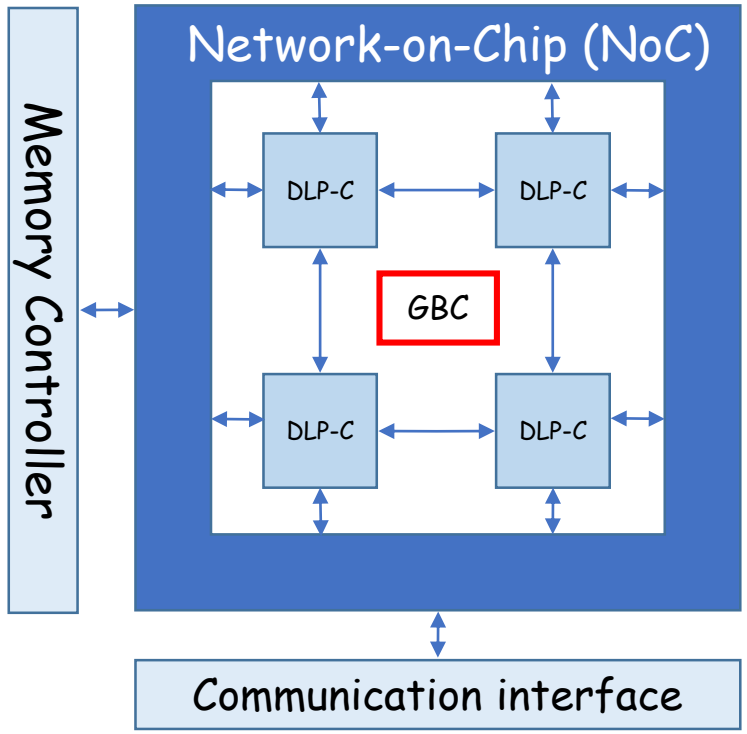
- Convolutional neural network as an example

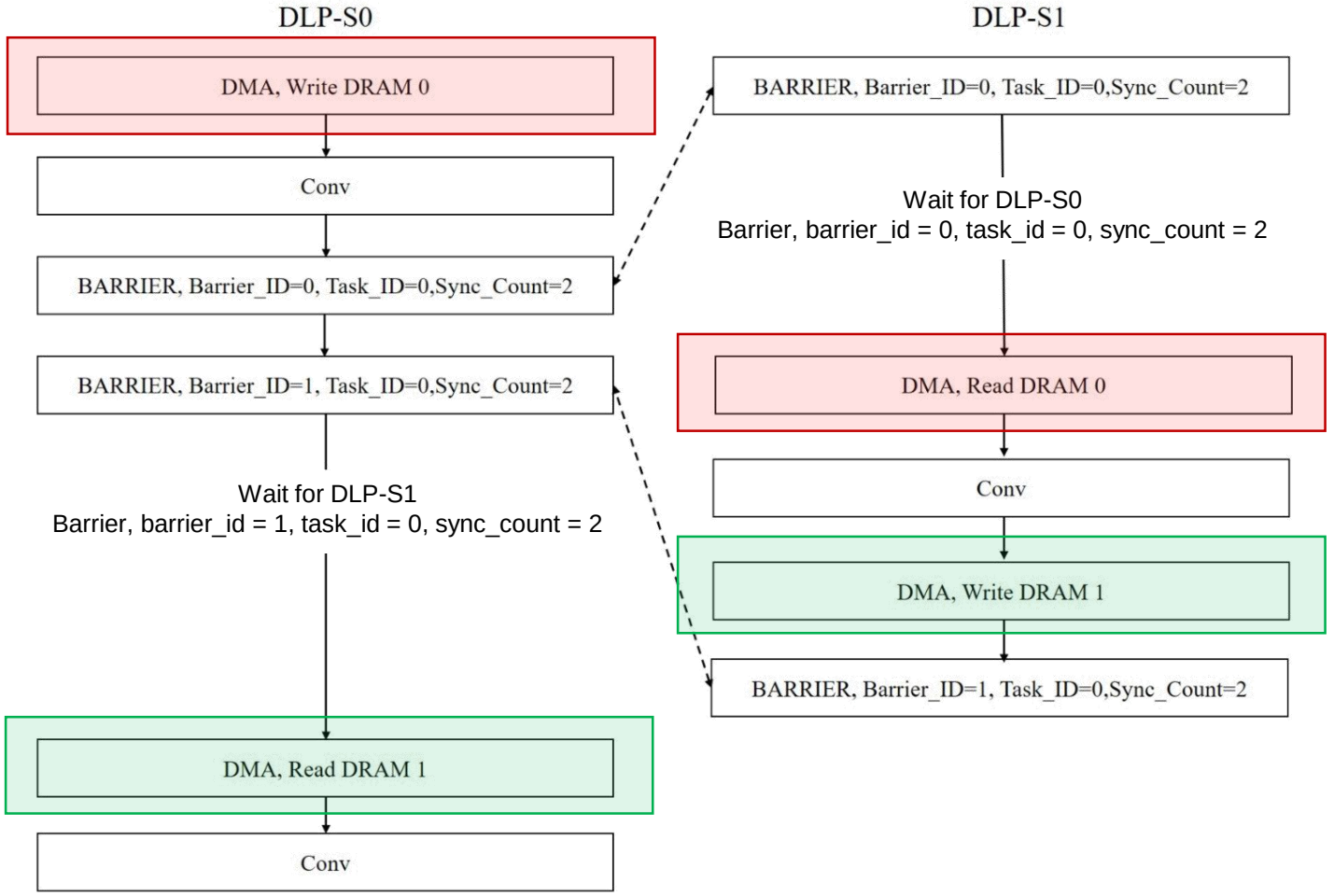


Synchronize via Barrier

- Barrier instruction

Barrier	Barrier_ID	Task_ID	Sync_count





Programming Language

- Motivations
  - Bridging the gap (programming language/hardware)

Conv. in C

```
for (n=0; n<N; n++) {
  for (k=0; k<K; k++) {
    for (p=0; p<P; p++) {
      for (q=0; q<Q; q++) {
        OA[n][k][p][q]= 0;
        for (r=0; r<R; r++) {
          for (s=0; s<S; s++) {
            for (c=0; c<C; c++) {
              h = p * stride - pad + r;
              w = q * stride - pad + s;
              OA[n][k][p][q] +=
                          IA[n][c][h][w]
                          * W[k][c][r][s];
            }
          }
        }
        OA[n][k][p][q]= Activation(OA[n][k][p][q]);
      }
    }
  }
}
```

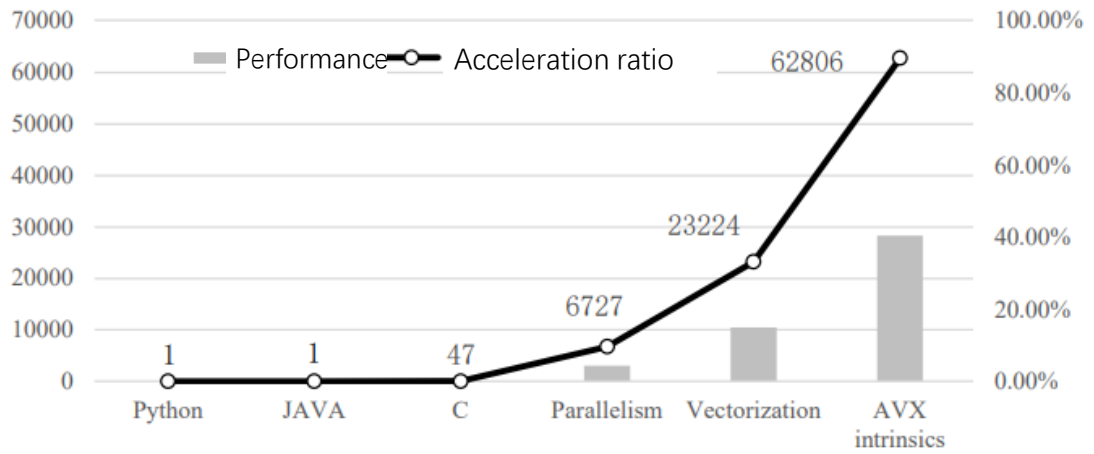
for each output activation

convolution window

Conv. in Pytorch, with Conv2d/tensor data type

```
self.conv2 = nn.Conv2d(6, 16, 5, bias=False)
```

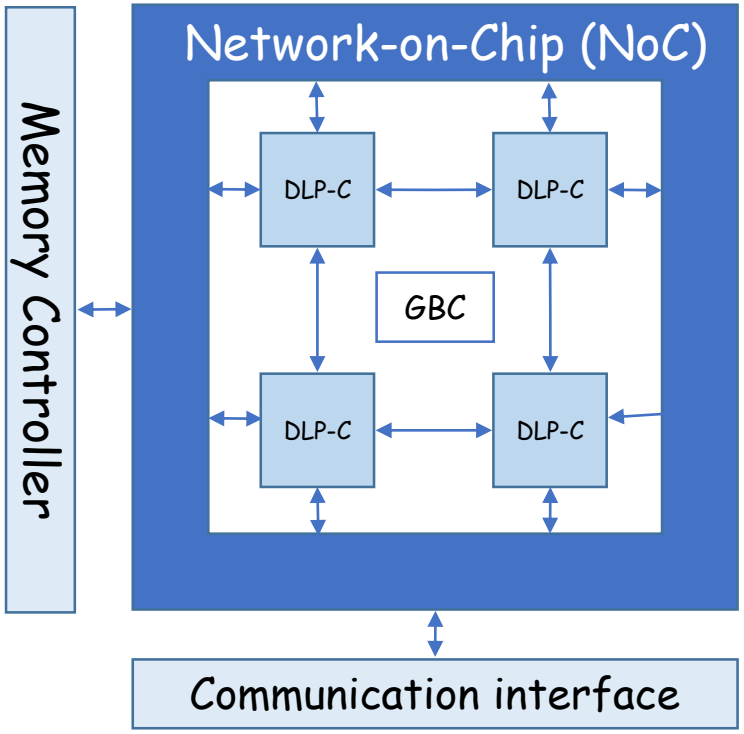
4k * 4k Matrix multiplication

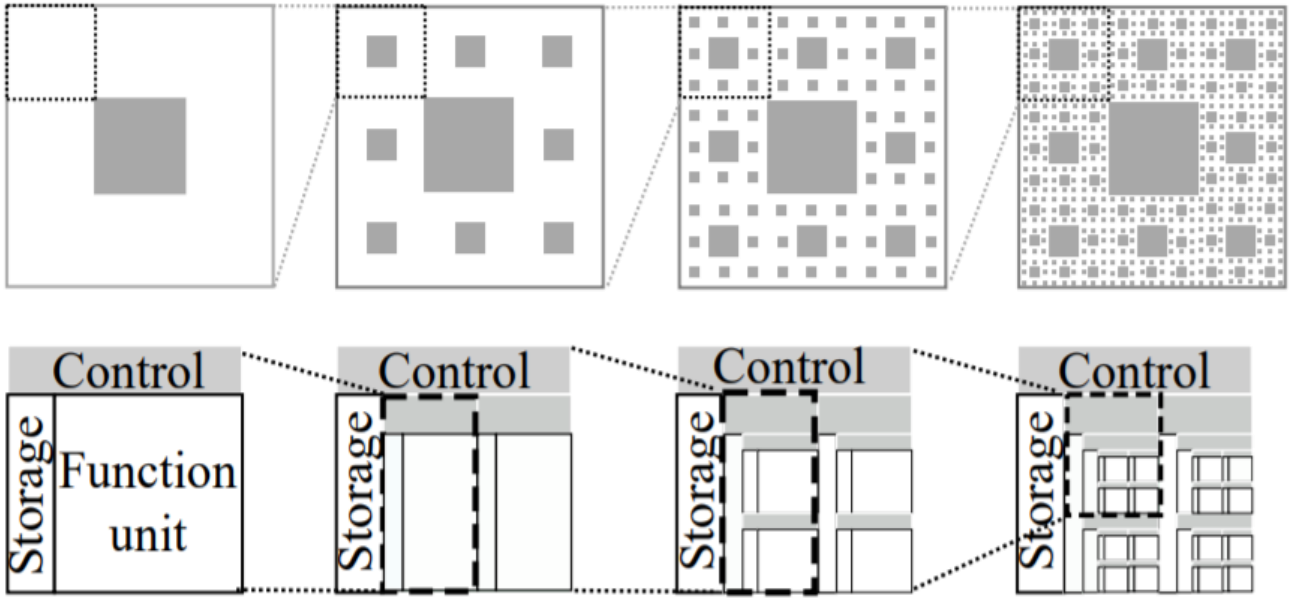


立志成才报国裕民

Bang-C Language by Cambricon

- Hardware abstraction
  - Memory
  - Control unit
  - Compute unit





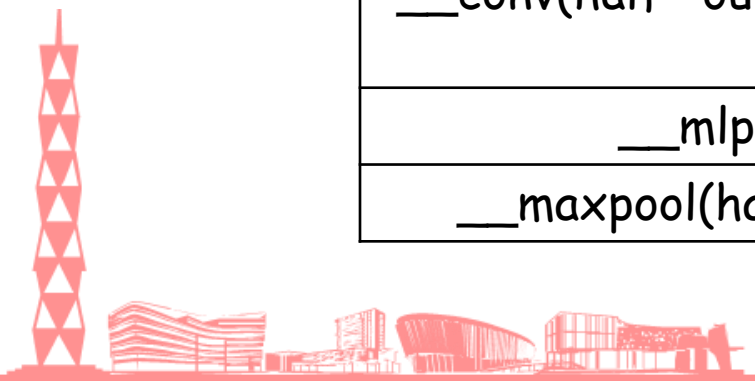
- Kernel abstraction
  - Data movement (optimize for scratchpad memory)
  - Control, improve ILP
  - Compute (scalar/vector/matrix)
  - Different data type (int4/int8/fp16, etc.)

立志成才报国裕民

Bang-C Language (Cont'd)

- Provide programming interface to the programmers
- Optimize dataflow (data movement and compute parallelly)
  - clusterDim, clusterId, coreDim, coreId
  - taskDim[XYZ], taskId[XYZ]
  - Block/Union1/Union2/Union4
- Optimize memory access
- Synchronization
  - __sync_cluster
  - __sync_all

Tensor operations in BANG-C
__vec_add(float* out, float *in1, float* in2, int size)
__vec_sub(float* out, float *in1, float* in2, int size)
__vec_mul(float* out, float *in1, float* in2, int size)
__conv(half* out, int8* in, int8* weight, half bias, int ci, int hi, int wi, int co, int kh, int kw, int sh, int sw)
__mlp(half* out, int8* in, int8* weight, half* bias, int ci, int co)
__maxpool(half* out, half* in, int ci, int hi, int wi, int kh, int kw, int sh, int sw)



立志成才报国裕民