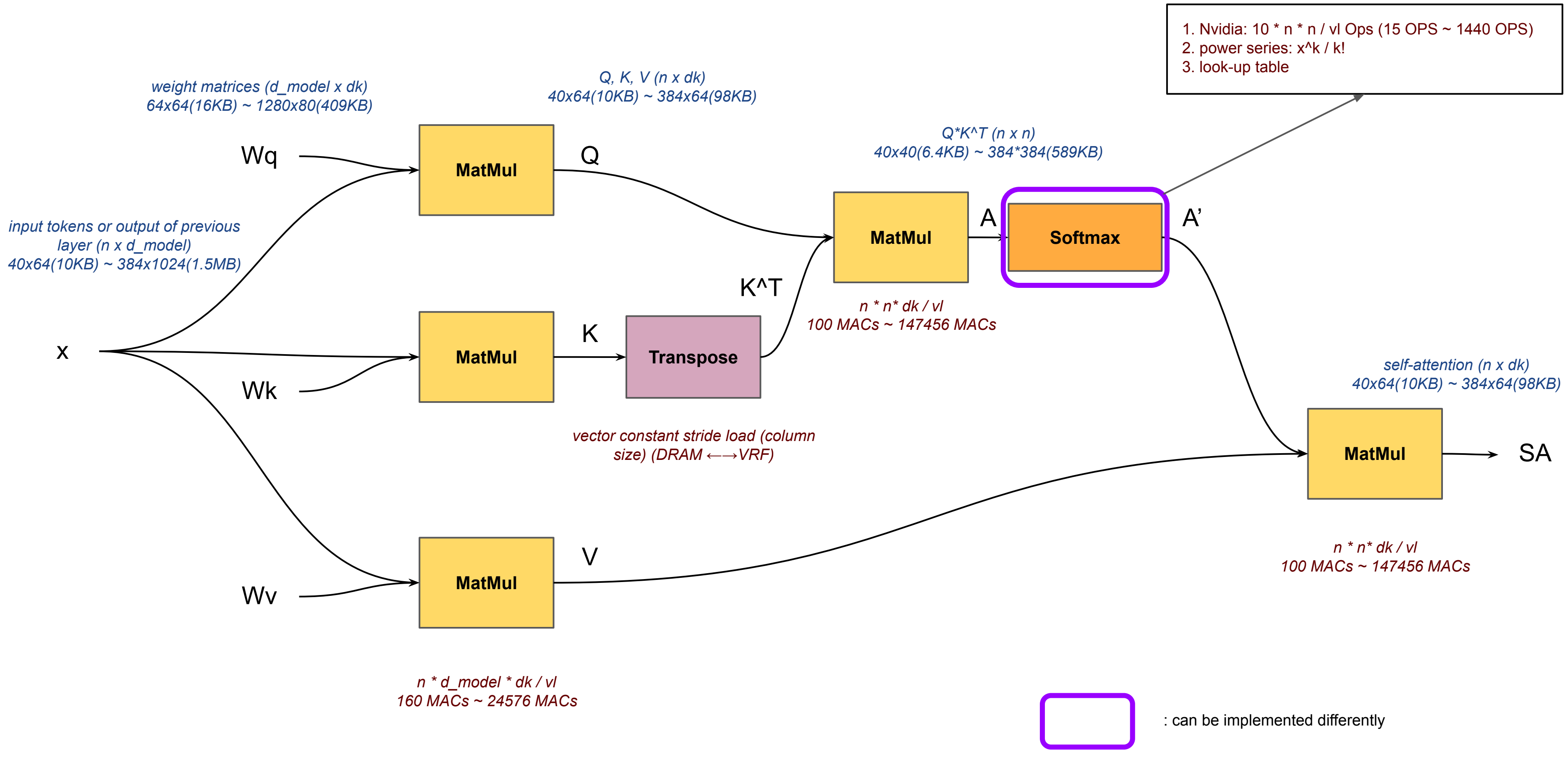
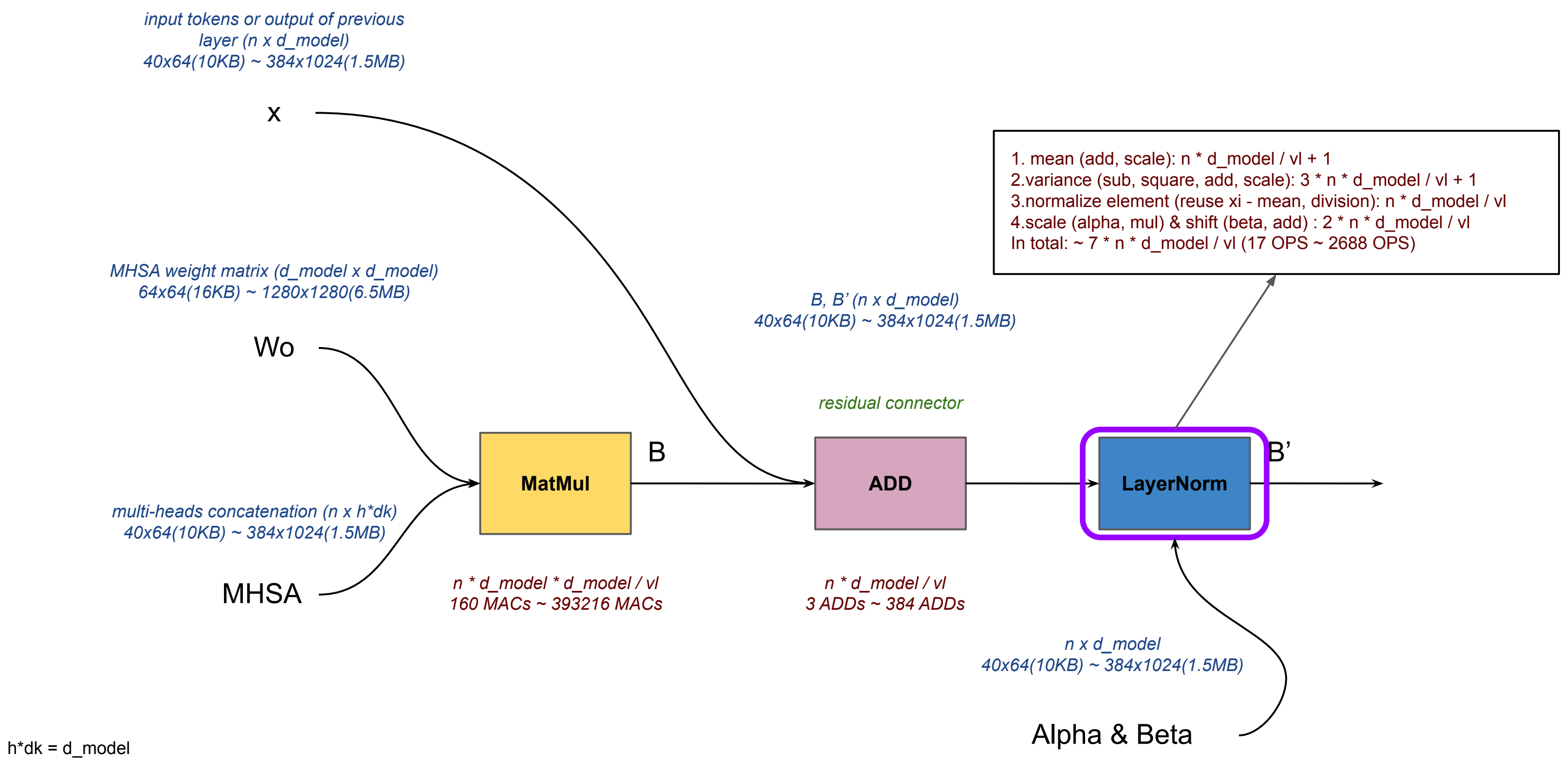
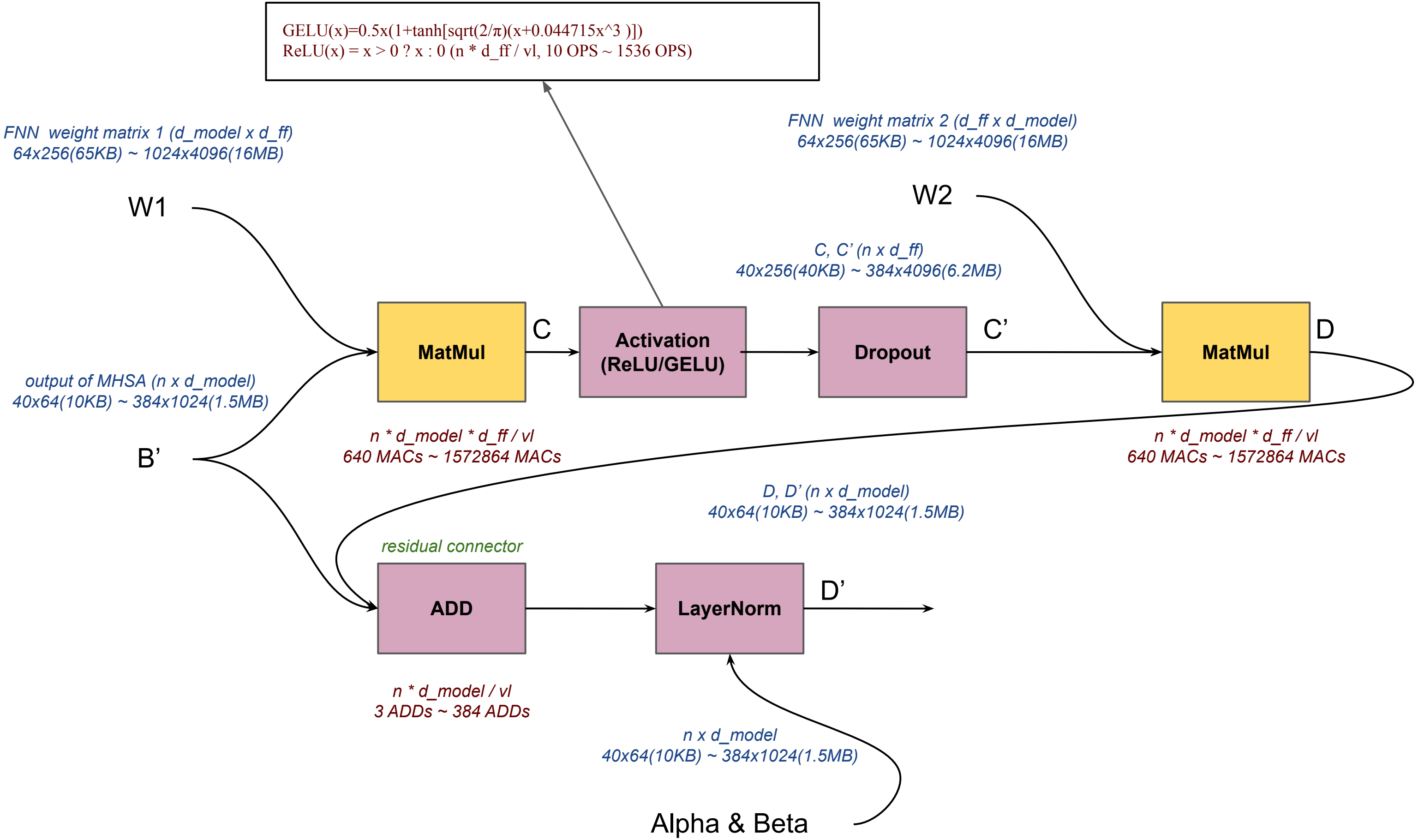
Week 4



data format: fp32, vector length (vl): 1024, MAC: multiply-accumulate operation


input tokens or output of previous layer (n x d_model)
40x64(10KB) ~ 384x1024(1.5MB)
x
MHSA weight matrix (d_model x d_model)
64x64(16KB) ~ 1280x1280(6.5MB)
Wo
multi-heads concatenation (n x h*dk)
40x64(10KB) ~ 384x1024(1.5MB)
MHSA
MatMul
B
n * d_model * d_model / vl
160 MACs ~ 393216 MACs
B, B' (n x d_model)
40x64(10KB) ~ 384x1024(1.5MB)
residual connector
ADD
n * d_model / vl
3 ADDs ~ 384 ADDs
LayerNorm
B'
1. mean (add, scale): n * d_model / vl + 1
2.variance (sub, square, add, scale): 3 * n * d_model / vl + 1
3.normalize element (reuse xi - mean, division): n * d_model / vl
4.scale (alpha, mul) & shift (beta, add) : 2 * n * d_model / vl
In total: ~ 7 * n * d_model / vl (17 OPS ~ 2688 OPS)
n x d_model
40x64(10KB) ~ 384x1024(1.5MB)
Alpha & Beta
h*dk = d_model


GELU(x)=0.5x(1+tanh[sqrt(2/π)(x+0.044715x^3)])
ReLU(x) = x > 0 ? x : 0 (n * d_ff / vl, 10 OPS ~ 1536 OPS)
FNN weight matrix 1 (d_model x d_ff)
64x256(65KB) ~ 1024x4096(16MB)
FNN weight matrix 2 (d_ff x d_model)
64x256(65KB) ~ 1024x4096(16MB)
W1
W2
C, C' (n x d_ff)
40x256(40KB) ~ 384x4096(6.2MB)
MatMul
C
Activation (ReLU/GELU)
Dropout
C'
MatMul
D
output of MHSA (n x d_model)
40x64(10KB) ~ 384x1024(1.5MB)
B'
n * d_model * d_ff / vl
640 MACs ~ 1572864 MACs
n * d_model * d_ff / vl
640 MACs ~ 1572864 MACs
D, D' (n x d_model)
40x64(10KB) ~ 384x1024(1.5MB)
residual connector
ADD
LayerNorm
D'
n * d_model / vl
3 ADDs ~ 384 ADDs
n x d_model
40x64(10KB) ~ 384x1024(1.5MB)
Alpha & Beta

Literature Summary

Transformer Models

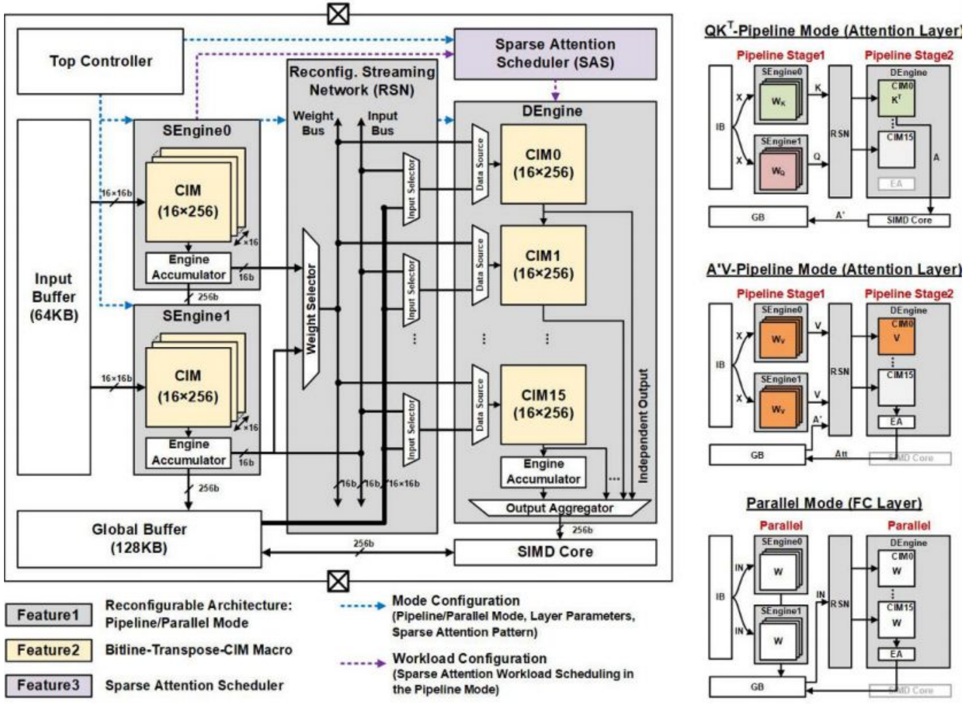
https://gigantic-jaguar-faa.notion.site/Transformer-Models-0588076e72884d7c8fe3dcb272cb37cc

Transformer Accelerator Designs:

https://gigantic-jaguar-faa.notion.site/Transformer-Accelerators-6d11dec8b13744aa8aec4da0dbc56631

CIM-Based with Configurable Pipeline/Parallel Modes

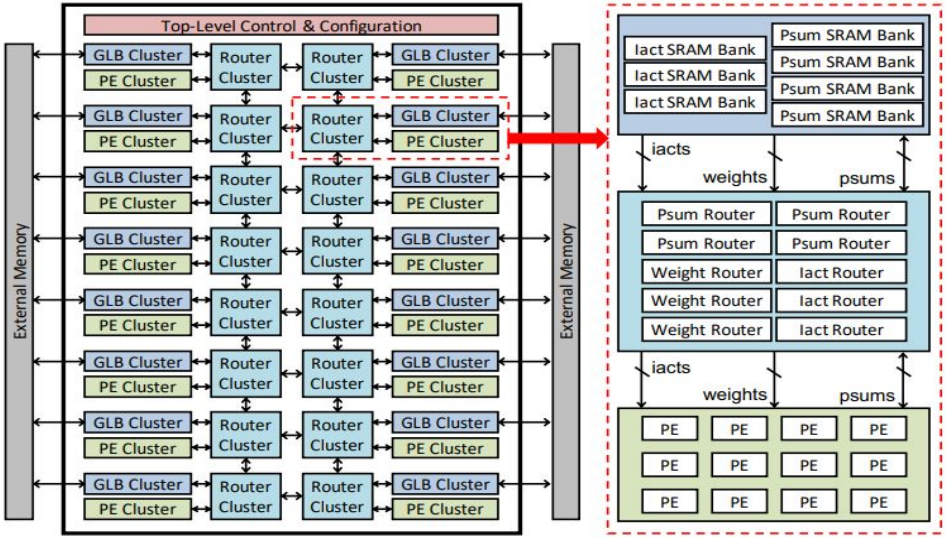
- 2 static engines(SEngine), 1 dynamic engine(DEngine) (data source selection, output aggregation)
- Parallel mode: all engines store FFN weights, run in parallel
- Pipeline mode
  - QK^T: SE0, SE1 compute Q, K(first stage), DE computes A(A = Q*K^T, second stage), SIMD core for softmax, scaling A'
  - A'V: SE0, SE1 compute V, DE loads A' from global buffer and computes A'V
- Use pipeline for some kernels? (LayerNorm, softmax)

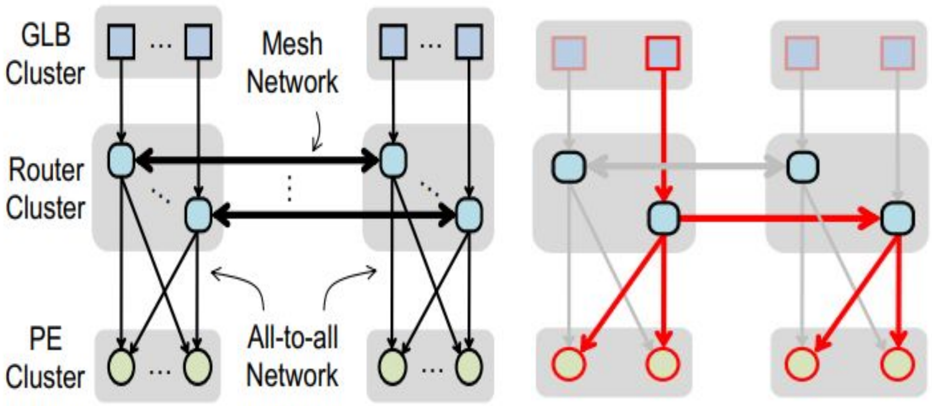


A 28nm 15.59 μ J/Token Full-Digital Bitline-Transpose CIM-Based Sparse Transformer Accelerator with Pipeline/Parallel Reconfigurable Modes, Fengbin Tu; Zihan Wu; Yiqi Wang; Ling Liang; Liu Liu; Yufei Ding; Leibo Liu; Shaojun Wei; Yuan Xie; Shouyi Yin

Eyeriss v2

- 2D hierarchical mesh network
  - a global buffer cluster(GLB, 12KB) is assigned for each PE cluster and connected to 2D mesh through a router
  - all-to-all network with two-level hierarchy (PE, cluster)
    - high-bandwidth: within cluster, unicast
    - high-reuse: broadcast to all PEs in another cluster
    - grouped/interleaved-multicast: multicast to some PEs in another cluster

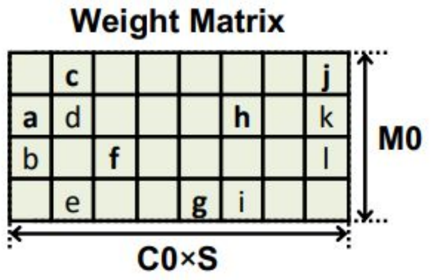




Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices, Yu-Hsin Chen, Tien-Ju Yang, Joel Emer, Vivienne Sze

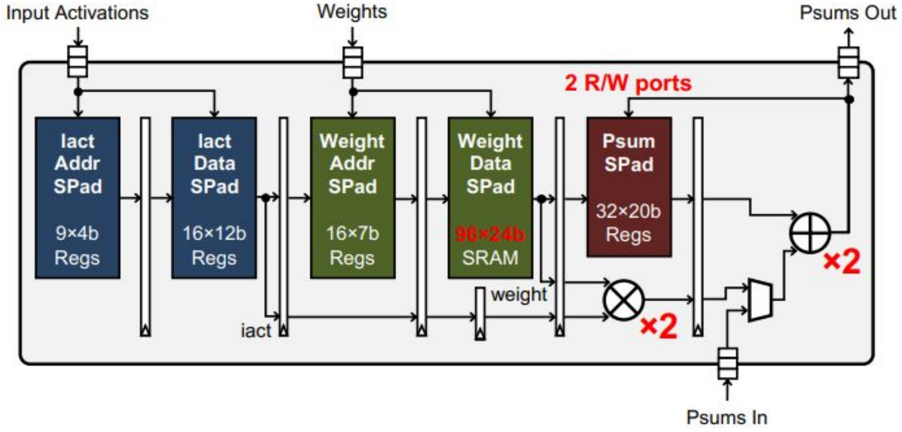
Eyeriss v2

- Compressed sparse column(CSC):
  - Process data in compressed form: fewer bandwidth requirements, energy saving
  - data vector: all non-zero data
  - count vector: # leading zeros from the previous non-zero value (additional overhead)
  - address vector: indicates the column address of each data (start with 1)
- PE arch with sparsity consideration
  - read address vector first
  - 7-stage pipeline: fetch non-zero iacts from SPads → fetch non-zero weights → MAC
  - compatible with normal format (low sparsity) → clock gate address SPads, set count to zero
- Results:
  - TSMC 64nm, 200MHz, 192 PEs, 153.6GOPS



CSC Compressed Data:

data vector: {a, b, c, d, e, f, g, h, i, j, k, l}
count vector: {1, 0, 0, 0, 1, 2, 3, 1, 1, 0, 0, 0}
address vector: {0, 2, 5, 6, 6, 7, 9, 9, 12}



A LayerNorm Optimization Trick

- Standard: $var(i) = \frac{1}{K} * \sum_{k=1}^{K} (x_{ik} - mean)^2$
    - Need to load xik twice, mean & variance stages
- Approximation: $var(i) = mean^2 - \frac{1}{K} * \sum_{k=1}^{K} x_{ik}^2$
    - xij^2 can be calculated in the mean stage
- Accuracy effect (including quantization, softmax optimization):
    - BLEU score on “tst2014”: 23.48

TED.tst2014			TEDX.tst.2014		
BLEU	TER	CTER	BLEU	TER	CTER
32.3	48.4	47.6	25.2	56.9	55.3
33.7	47.4	46.7	24.7	59.3	54.9
32.3	47.9	47.7	25.7	56.0	55.1
32.6	47.1	47.5	26.4	55.4	54.7
29.4	51.6	49.9	25.2	56.5	54.1
30.4	50.1	49.4	26.3	54.8	55.9
30.8	49.6	48.4	27.1	53.9	52.9
30.6	49.7	49.5	26.0	54.0	56.7
32.1	49.6	48.0	25.9	56.1	54.1
30.8	50.3	49.5	24.6	56.8	55.7
30.9	50.1	49.5	24.9	56.2	55.5
33.4	47.1	46.7	26.2	56.4	54.1
34.2	46.5	46.9	27.6	53.1	55.6
33.8	46.7	46.9	27.9	53.2	54.3

Hardware Accelerator for Multi-Head Attention and Position-Wise Feed-Forward in the Transformer, Siyuan Lu, Meiqi Wang, Shuang Liang, Jun Lin, and Zhongfeng Wang
The RWTH Aachen Machine Translation System for IWSLT 2016, Jan-Thorsten Peter, Andreas Guta, Nick Rossenbach, Miguel Grac¸a, and Hermann Ney