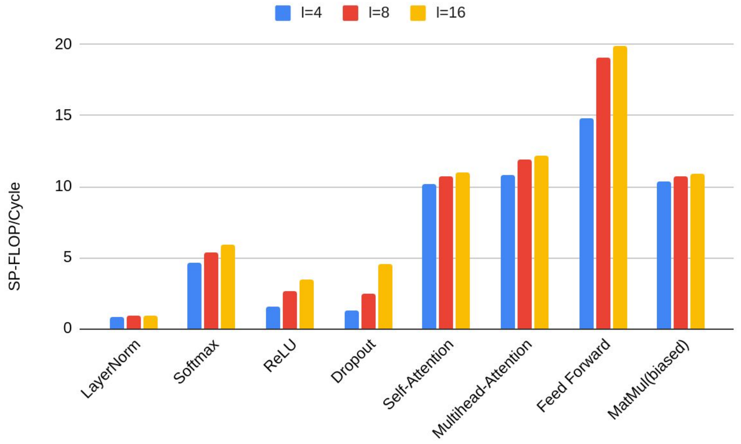
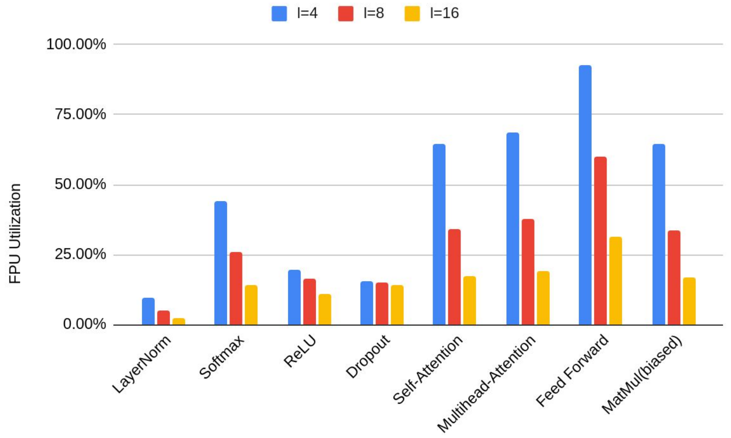
Week 10

Performance Analysis

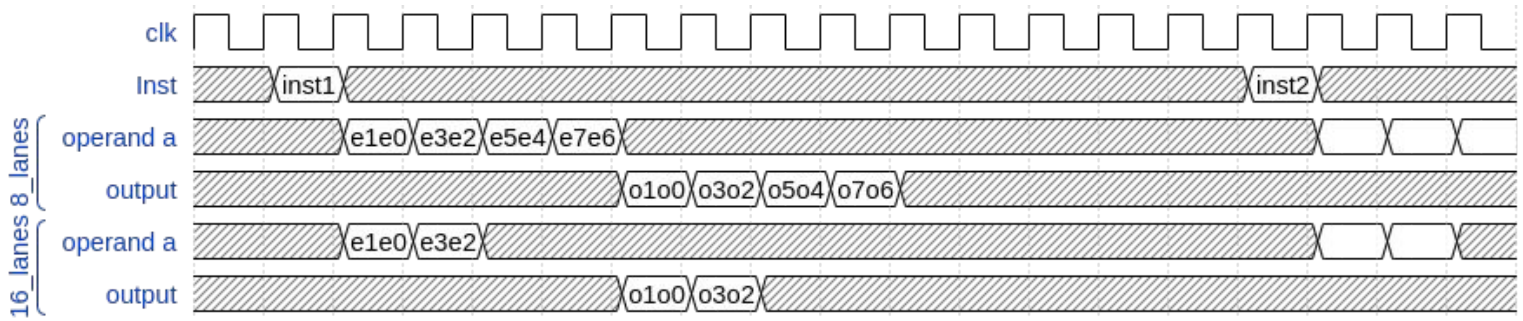




1. LayerNorm, Softmax, ReLU, Dropout are negligible
2. MatMul performance does not scale with the number of lanes (low FPU utilization)

Performance Analysis (MatMul)

- Short vector length:
  - Self-Attention: 64x768x64 (D1xD2xD3), vectorize D3, vl=64
  - For each lane, 64-bit databus
    - 4-lanes: 8 data
    - 8-lanes: 4 data
    - 16-lanes: 2 data
- Non-consecutive vector instructions



Solution 1

- Manually extend the vector length by merging MAC instructions

vl=64

```
vfmacc.vf v0, t0, v16
vfmacc.vf v1, t1, v16
vfmacc.vf v2, t2, v16
vfmacc.vf v3, t3, v16
vfmacc.vf v4, t4, v16
vfmacc.vf v5, t5, v16
vfmacc.vf v6, t6, v16
vfmacc.vf v7, t7, v16
```

→

vl=512

```
vfmacc.vv v0, v2’, v16’

v16’ = {8{v16}}
v2’ = {64{t0}, 64{t1}, …, 64{t7}}

use LMUL=⅛ to separate data
```

- Can only mitigate the problem
  - 16_lanes: 16 data, but 32_lanes: 8 data
  - max_vl=1024 (LMUL=8)

Target performance: 256 GFLOPS
92% utilization → nr_lanes > 64

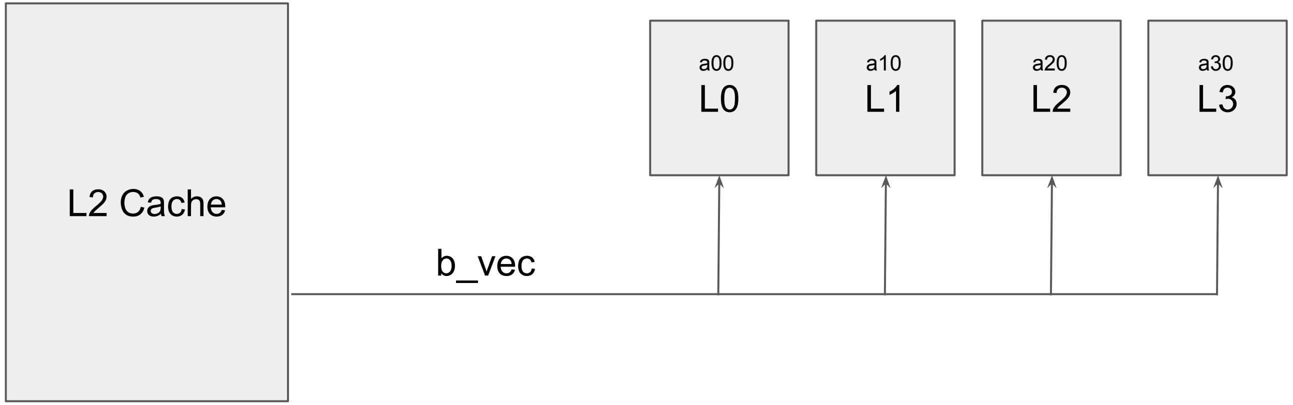
Solution 2

- Reduce the gap between adjacent vector instructions
- Scalar operand is embedded in the instruction, Ariane needs time to load.

Solution 3

(a00 * b_vec0) + (a01 * b_vec1) + (a02 * b_vec2) + … + (a0n * b_vec_n) = C_row_0
(a10 * b_vec0) + (a11 * b_vec1) + (a12 * b_vec2) + … + (a1n * b_vec_n) = C_row_1
…
(an0 * b_vec0) + (an1 * b_vec1) + (an2 * b_vec2) + … + (ann * b_vec_n) = C_row_n

- Broadcast b_vec to all lanes, such that each lane can get 64 elements



New instructions:

1. vlbe: vlsu loads a vector from L2 cache, and sends it to every lane (to avoid critical paths, we can use a hierarchical 2D mesh)
2. vfsmacc.vf: A (a00, a10) is also a vector, but in each lane, it is still a vector-float operation
3. vspse: results are not shuffled in VRF, need a special store instruction

Hardware modification:

1. New registers in each lane to temporarily store intermediate result (vd)
   a. Store a complete vector, up to LMUL=8
   b. 4kB, 1 cycle latency
   c. Simply extend VRF
2. A new datapath from L2 cache (VLSU) to lanes
3. Change the control logic of vmfpu
   a. vfmacc: c = c + (a * b)
   b. a: from VRF, but use only one element at a time
   c. b: from L2 cache
   d. c: from the new registers