ETH zürich

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

NextRAN-AI – 03/10/2025

Integrated Systems Laboratory (ETH Zürich)

Marco Bertuletti mbertuletti@iis.ee.ethz.ch
Yichao Zhang yiczhang@iis.ee.ethz.ch
Mahdi Abdollahpour mahdi.abdollahpour@unibo.it
Alessandro Vanelli-Coralli avanelli@iis.ee.ethz.ch
Luca Benini lbenini@iis.ee.ethz.ch

PULP Platform
Open Source Hardware, the way it should be!

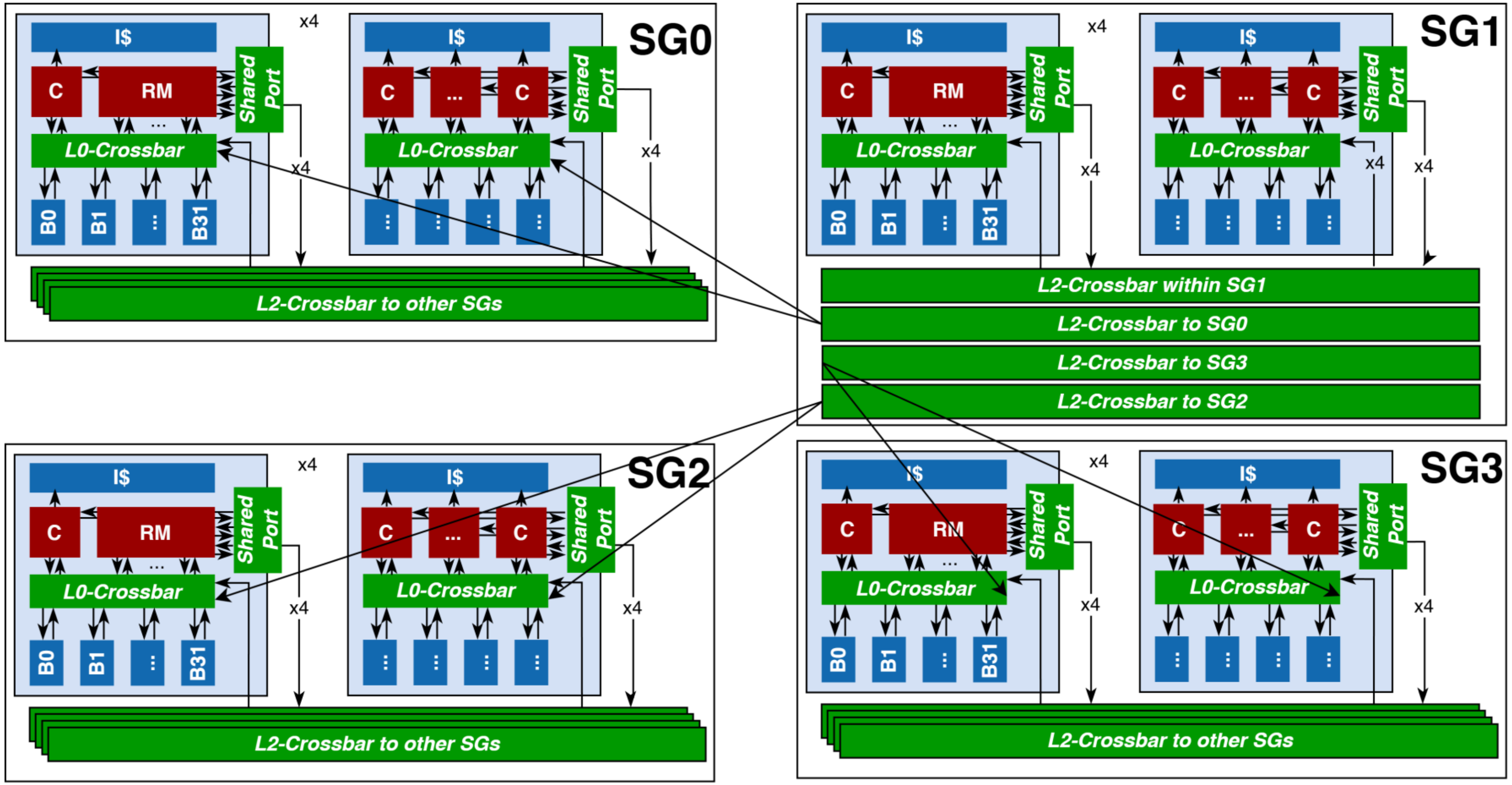
pulp-platform.org
@pulp_platform
company/pulp-platform
youtube.com/pulp_platform

Outline

- Experiments on RedMulE size + Parallel RedMulE execution
- How to improve performance on large tensor arrays (16x16) → BW

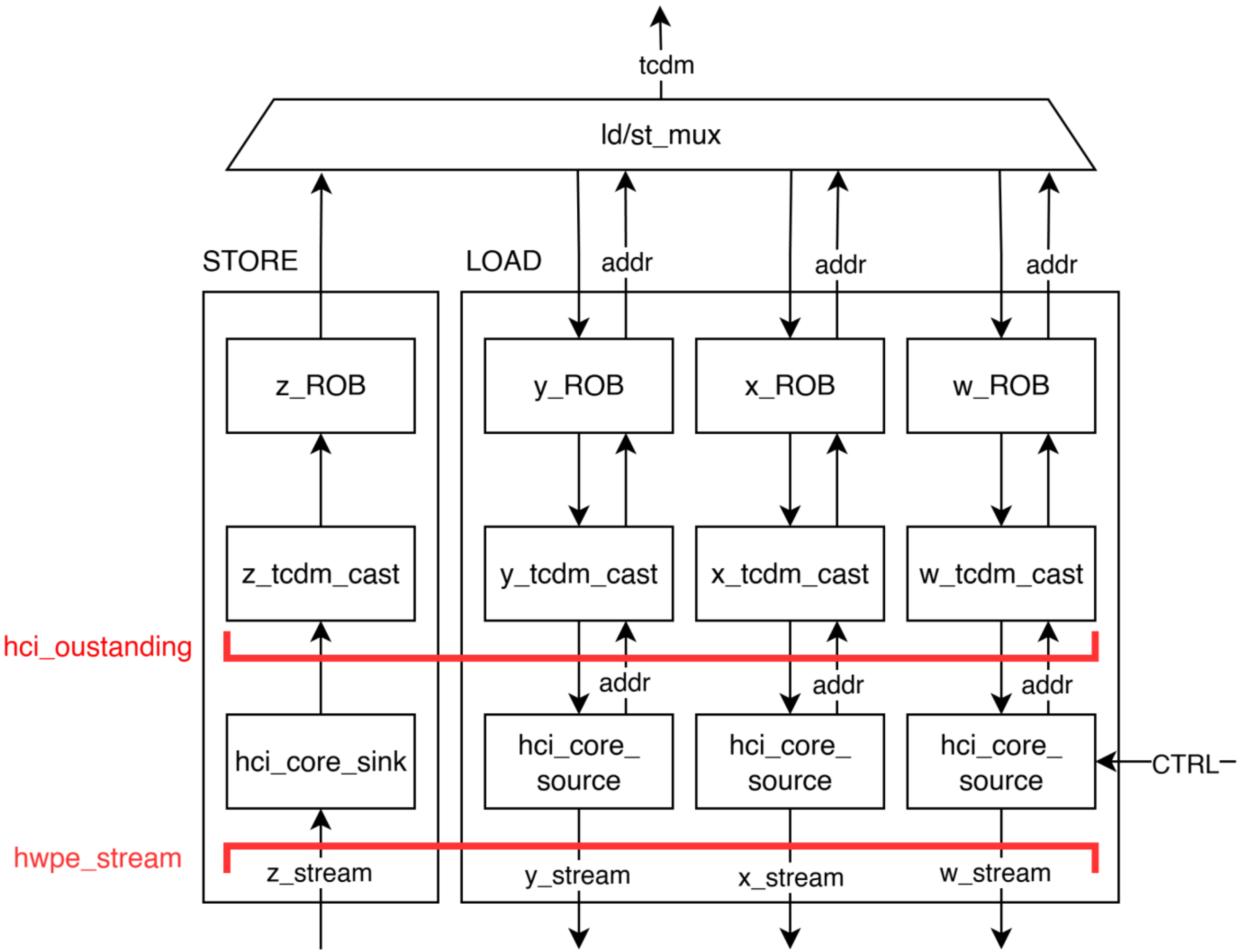
TensorPool SubGroup

100 cores, 4 RedMules, 4 Tiles, 8 cores / Tile, 128KiB SRAM



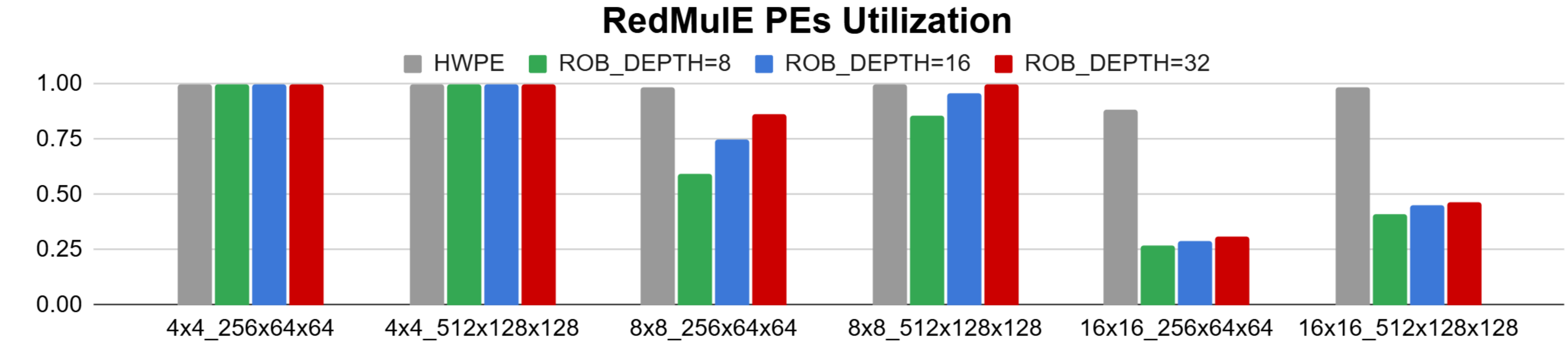
ETH zürich ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

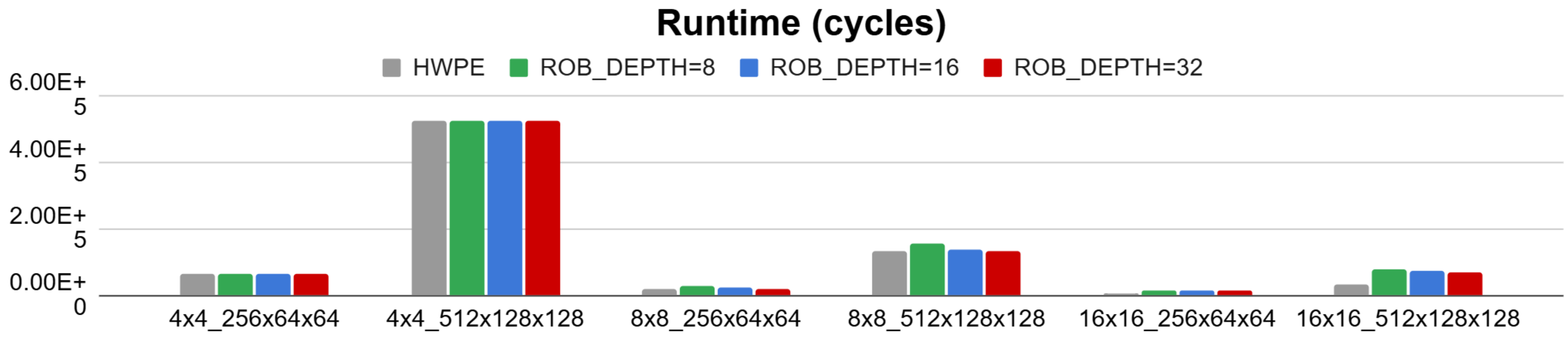
Implementation with three ROBs



Varying the size of the ROB

HWPE corresponds to the standalone RedMulE

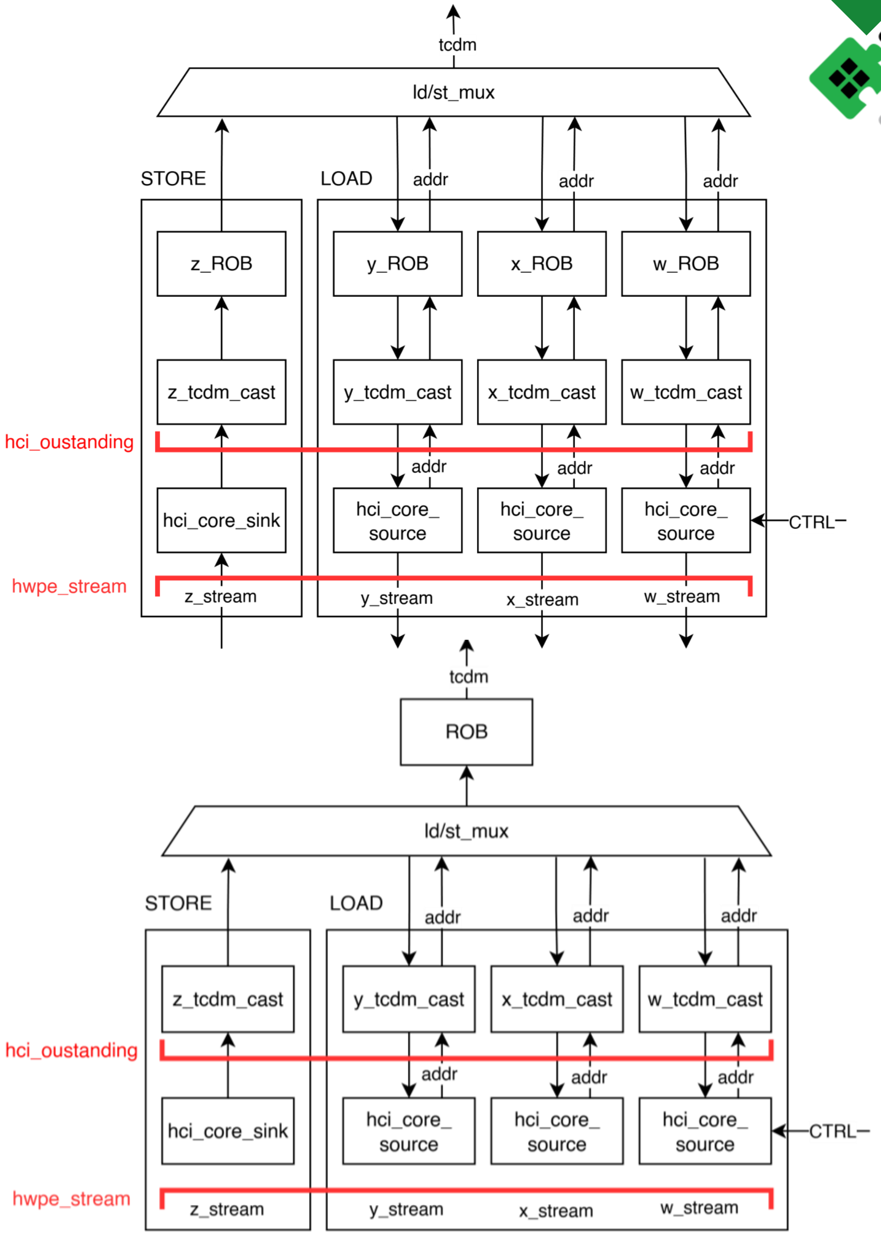




Implementation with one ROB

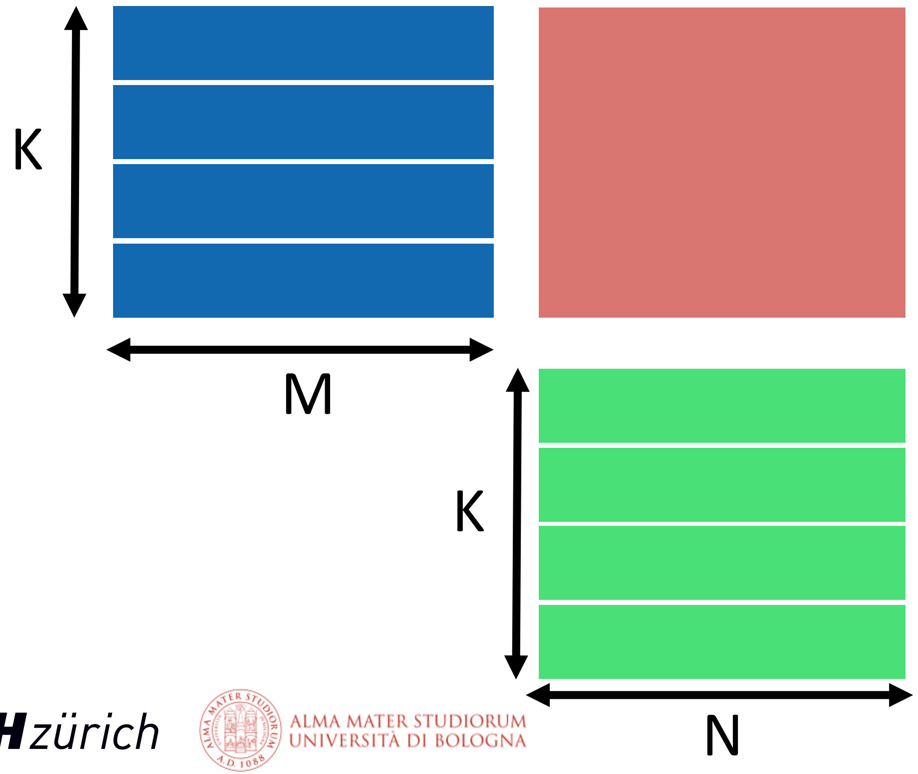
- Reduces size of the transaction table
- No waste in memory, because only some streams (W) need many ROB entries

ISSUE: transactions of different streams do not need to be in-order, if one stream is not ready it blocks the responses of other streams → **with dedicated ROBs we are 1.1x faster**



Parallelization

- Matmul parallelized over rows of X over multiple RedMulEs
- Modified the runtime to obtain easily redmule IDs and RedMulEs number

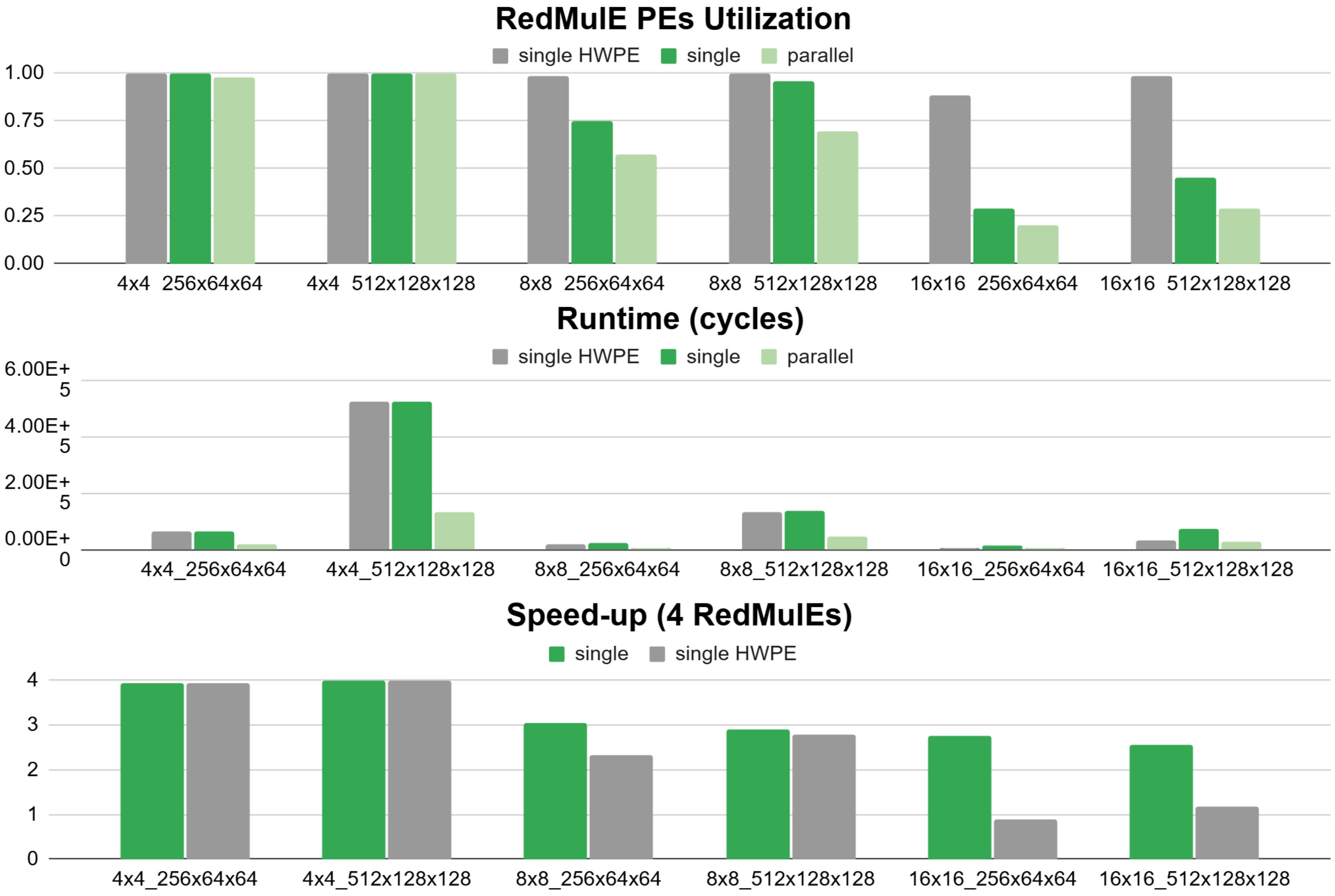



```
// Matmul, MxKxN, all data in L1

uint32_t redmule_id = get_redmule_id();
uint32_t redmule_num = get_redmule_num();

for (i = 0; i < num_redmules; i++) {
        hwpe_soft_clear();
        redmule_cfg (
                ptr_X + redmule_id * M * K / num_redmules,
                ptr_W,
                ptr_Y + redmule_id * M * K / num_redmules,
                M, K / num_redmules, N,
                GEMM, Float16);
        hwpe_trigger_job();
}
// potentially work from other cores
mempool_barrier();
```

Parallelization Results



16x16 array requires 1024bit/cycle (32 memory ports)

- The accelerator memory ports access the same remote connection on writes and read responses
- To scale to 16x16 (256MACs/cycle) we should:
  - Increase the BW to other Tiles
  - Increase the L dimension of the accelerator, which does not impact the BW
  - Reduce the accelerator’s BW

How to reduce the tensor engine BW?

We are solving a problem of size MKN, given the bitwidth of the elements β and tiling of size mnk we compute the bits loaded:

Output stationary:

```
for i:m:M
  for j:n:N
    load (Y)(β × m × n)
    for l:k:K
      load (X)(β × m × k)
      load (W)(β × k × n)
      Y += X × W
    end
    store (Y)(β × m × n)
  end
end
```

$$\frac{M}{m}\frac{N}{n}\left(2\beta mn + \frac{K}{k}(\beta mk + \beta kn)\right) =$$

$$MNK\beta\left(\frac{2}{K} + \frac{m+n}{mn}\right)$$

Input stationary:

```
for i:m:M
  for l:k:K
    load (X)(β × m × k)
    for j:n:N
      load (Y)(β × m × n)
      load (W)(β × k × n)
      Y += X × W
      store (Y)(β × m × n)
    end
  end
end
```

$$\frac{M}{m}\frac{K}{k}\left(\beta mk + \frac{N}{n}(2\beta mn + \beta kn)\right) =$$

$$MNK\beta\left(\frac{1}{N} + \frac{2m+k}{mk}\right)$$

Weight stationary:

```
for l:k:K
  for j:n:N
    load (W)(β × k × n)
    for i:m:M
      load (Y)(β × m × n)
      load (X)(β × m × k)
      Y += X × W
      store (Y)(β × m × n)
    end
  end
end
```

$$\frac{N}{n}\frac{K}{k}\left(\beta kn + \frac{M}{m}(2\beta mn + \beta mk)\right) =$$

$$MNK\beta\left(\frac{1}{M} + \frac{2n+k}{nk}\right)$$

How to reduce the tensor engine BW?

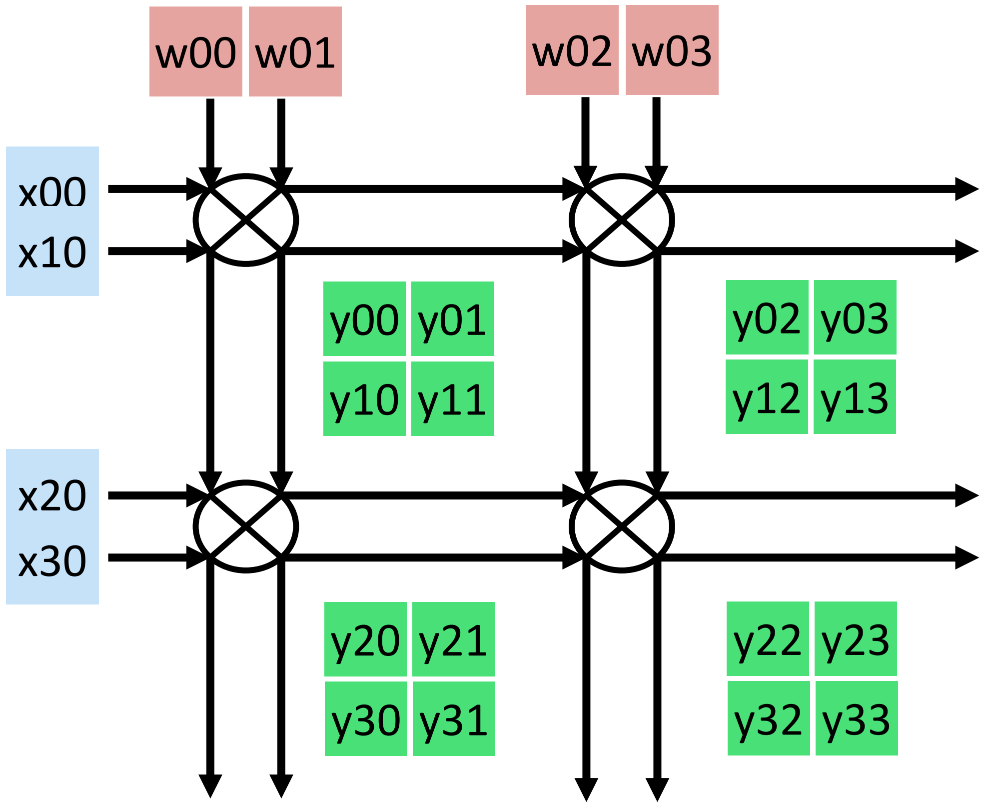
We compute the arithmetic intensity as the operations 2MKN divided by the bits loaded, therefore the bits loaded must be minimized:

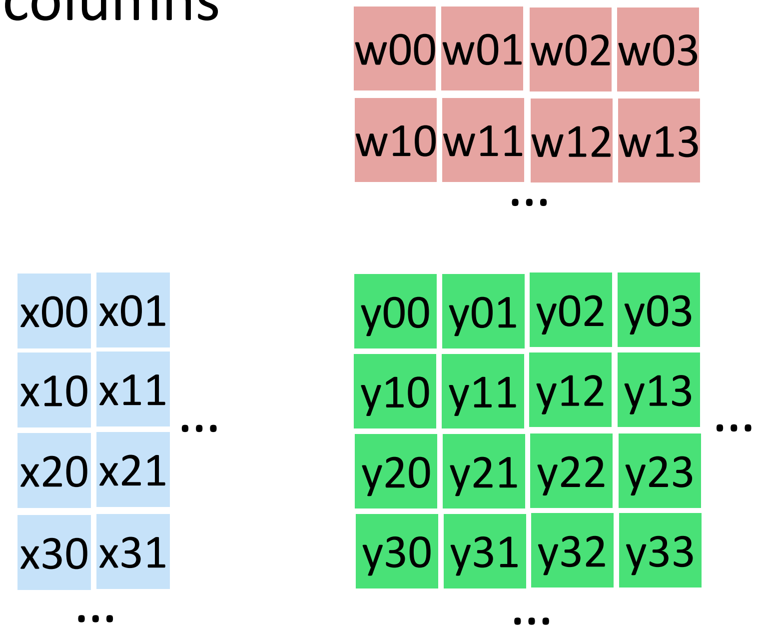
- Output stationary $MNK\beta\left(\frac{2}{K}+\frac{m+n}{mn}\right) \approx \frac{\mathrm{MNK}\beta(m+n)}{mn}$
- Input stationary $MNK\beta\left(\frac{1}{N}+\frac{2m+k}{mk}\right) \approx \frac{\mathrm{MNK}\beta(2m+k)}{mk}$
- Weight stationary $MNK\beta\left(\frac{1}{M}+\frac{2n+k}{nk}\right) \approx \frac{\mathrm{MNK}\beta(2n+k)}{nk}$

We notice that the smallest value of the loaded bits is for a square tile m=n=k, therefore the output stationary solution always guarantees the highest arithmetic intensity

Output stationary engine, outer product

- X and W inputs are broadcasted over row/columns
- Y inputs are stored in accumulators

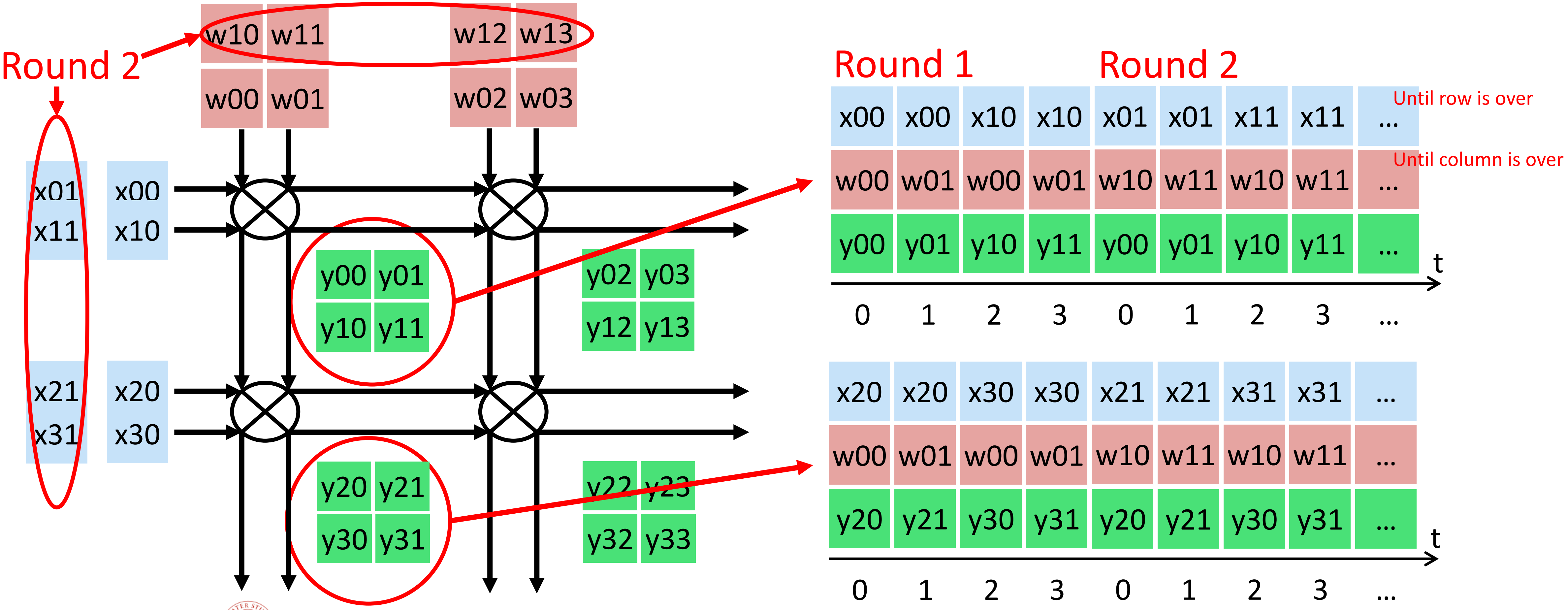




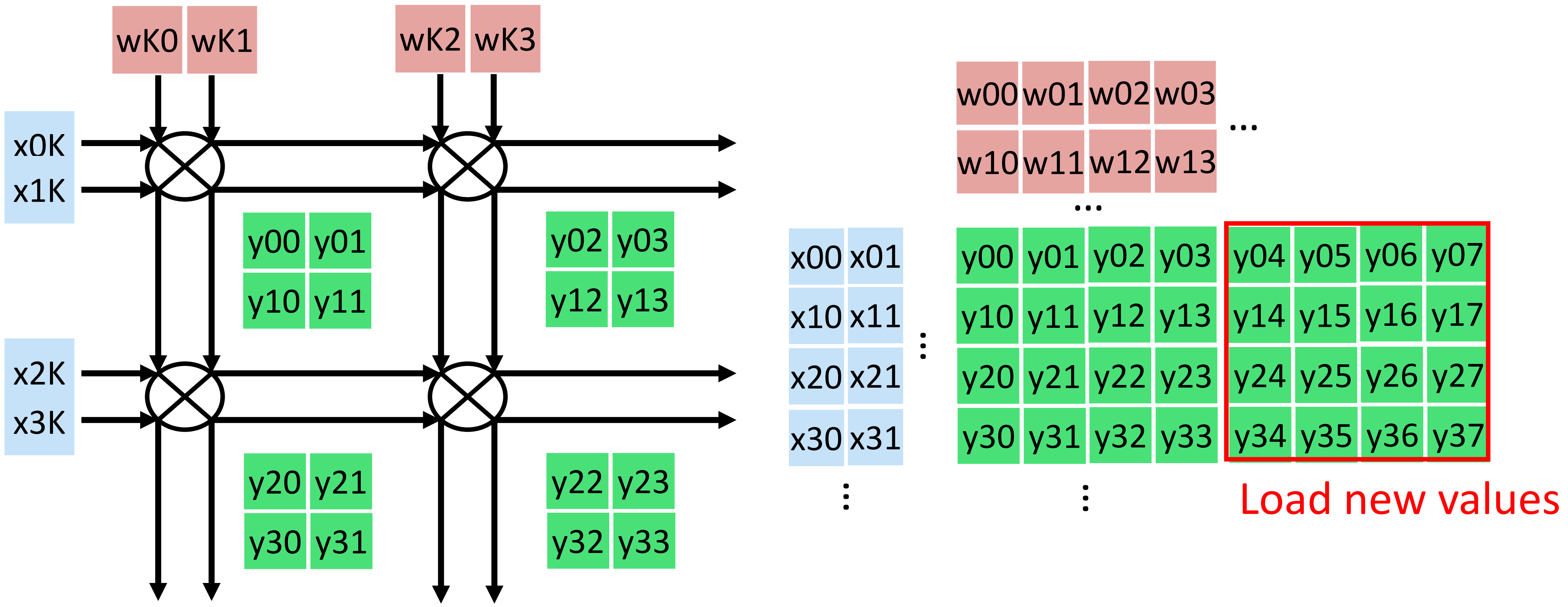
Rows in X are multiplied by columns in W and stored in Y

Output stationary engine, outer product

The FPU has 4 pipeline stages, to keep all of them busy we must load 2 elements from X and 2 elements from W per FPU every 4 cycles. Ys are stationary.



Output stationary engine, outer product



But once we arrive at the end of an X column / W row we need to replace them with a new output tile, we can exploit the 4 computation cycles, pre-loading 4ys/FPU every 4 cycles

Output stationary engine , outer product

Given L the size of the tensor unit (number of FPUs on one side)

1. **Load $4L^2$ from Y to initiate the accumulators**
2. **4-cycles operation of the accelerator**
    1. Load 2L from X
    2. Load 2L from W
    3. Load 2L from Y (next iteration)
    4. Load 2L from Y (next iteration)
3. **In 4L cycles the computation on the initial $4L^2$ Y elements is completed and the accelerator can continue on the $4L^2$ Y-elements loaded during point 2**

Output stationary engine

What is the required BW?

- Square tensor engine
- P = number of pipeline stages of the FPUs
- L = number of FPUs per tensor engine side

RedMulE: $BW = \frac{[16bit \times L \times (P+1)]}{cycle}$ $\frac{BW}{\pi} = \frac{16 \times (P+1)}{L} \frac{bit}{OP}$

Output stationary: $BW = \frac{[16bit \times 2 \times L]}{cycle}$ $\frac{BW}{\pi} = \frac{16 \times 2}{L} \frac{bit}{OP}$

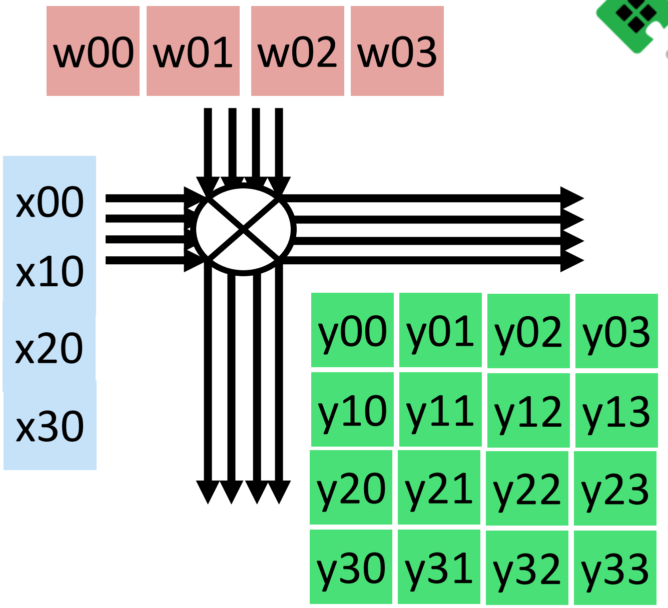
Latency tolerance

Increase latency tolerance with more accumulators

x00	x00	x10	x10	x01	x01	x11	x11	…
w00	w01	w00	w01	w10	w11	w10	w11	…
y00	y01	y10	y11	y00	y01	y10	y11	…
0	1	2	3	0	1	2	3	…

t

2 elements /
4 cycles



x00	x00	x00	x00	x10	x10	x10	x10	x20	x20	x20	x20	x30	x30	x30	x30	…
w00	w01	w02	w03	w00	w01	w02	w03	w00	w01	w02	w03	w00	w01	w02	w03	…
y00	y01	y02	y03	y10	y11	y12	y13	y20	y21	y22	y23	y30	y31	y32	y33	…
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…

t

4 elements /
16 cycles

Next Steps

- **TensorPool:**
  - Optimize RedMulE to L1 connection → modify TeraPool interconnect for TensorPool-5 config.
  - TeraPool + RedMulE (TensorPool) physical design in 7nm
  - PPA on model microkernels and operators (combined RedMulE&Cores)
- **System Performance:**
  - Simulation speed is impaired by large design size → from RTL simulation to higher abstraction level, e.g. GVSoC
  - TensorPool GVSoC model developement
  - Data-Movement and end-end performance