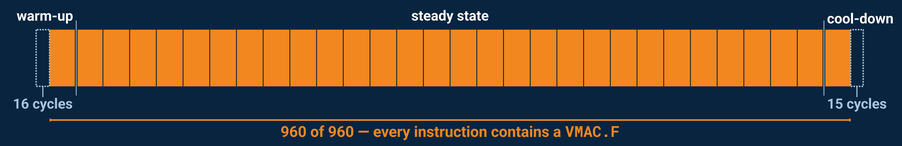
96%
of BFP16 peak compute
for a tensor-contraction kernel
on one XDNA2 compute tile
960 of 991 instructions contain a `VMAC.F`
1777 of 1843 GFLOPS measured

VLIW Scheduling: Timing Is All You Need

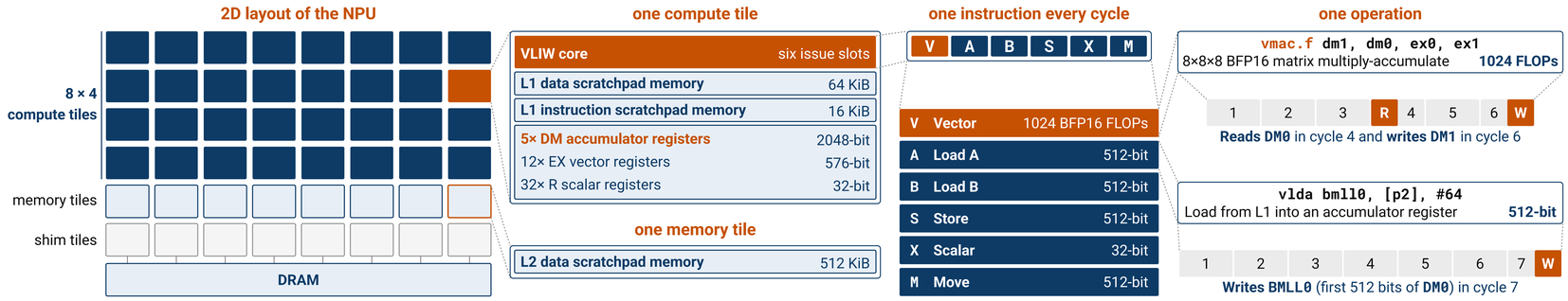
Time-multiplexing four accumulation chains onto two accumulator registers.

Tamino Steinert, Alexander Breuer — Friedrich Schiller University Jena, Germany



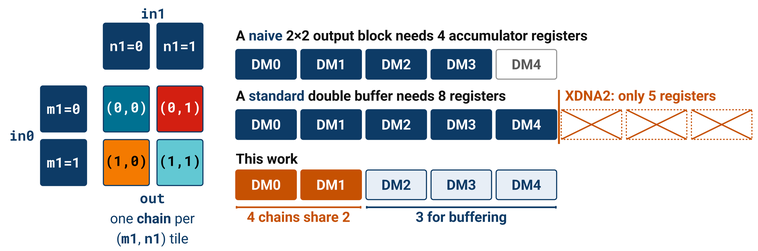
960 of 991 instructions contain a matrix multiply-accumulate (`VMAC.F`). The other 31 are the warm-up fill and the cool-down store tail.

1 The XDNA2 NPU of AMD Ryzen AI Max SoCs

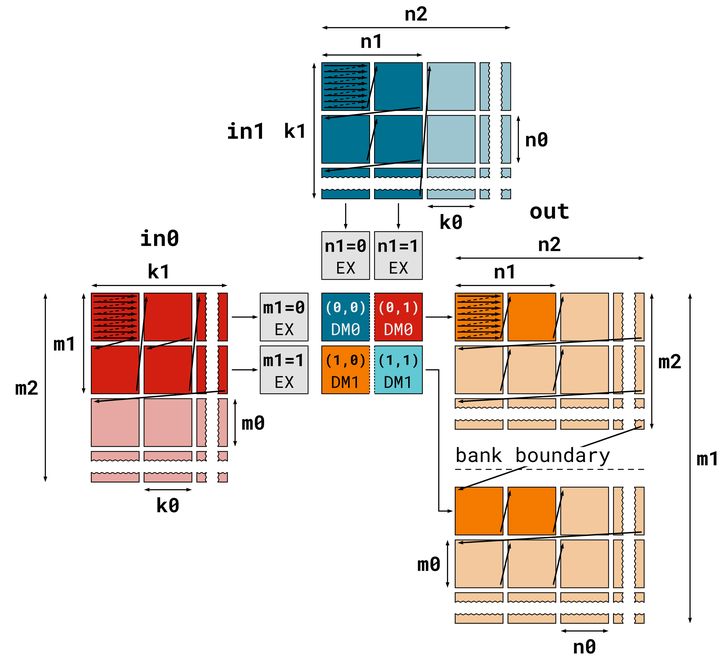


2 Five-register constraint

To **fully saturate** the vector unit, a **2×2 output blocking** is needed. A **standard double buffer** for a **naive 2×2 output block** needs **eight** accumulator registers, whereas XDNA2 has only **five** [1].



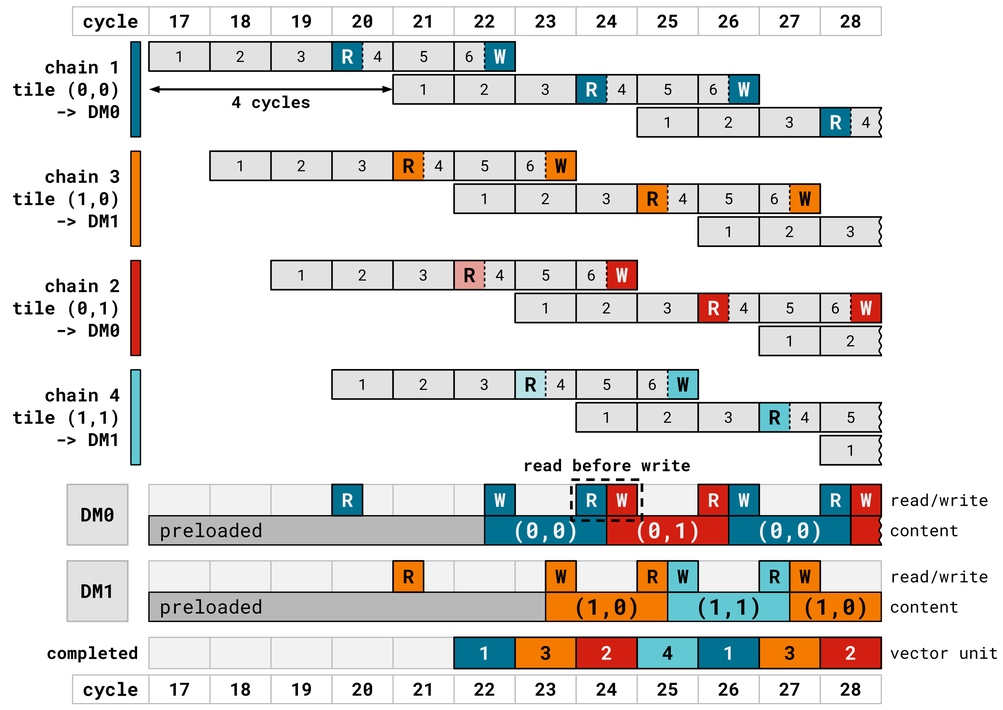
3 The tensor-contraction kernel and its data layout



Einsum: `[m2,k1,m1,m0,k0],[n2,k1,n1,n0,k0]->[m1,m2,n2,n1,m0,n0]`
Sizes: `|m0|=|n0|=|k0|=8; |m1|=|n1|=2; |m2|=6; |n2|=5; |k1|=8`

The tile dimensions **`m0`**, **`n0`**, and **`k0`** are consumed by the **8×8×8 `VMAC.F` operations**. Dimensions **`m1`** and **`n1`** form the **2×2 output blocking**. The outer dimensions **`m2`**, **`n2`**, and **`k1`** are **runtime parameters**. The running example sets **`m2`**, **`n2`**, and **`k1`** to **6**, **5**, and **8**, resulting in a computation equivalent to a pre-tiled **M96×N80×K64 matrix multiplication**.

4 Scheduling four chains of `VMAC.F` operations onto two accumulators



Each **accumulation chain** steps through the **`k1`** dimension and accumulates one tile of the 2×2 output block. Chains 1 and 2 are time-multiplexed onto **`DM0`**, and chains 3 and 4 are time-multiplexed onto **`DM1`**. A **`VMAC.F`** reads its accumulator in the **first half** of cycle 4 and writes it in the **second half** of cycle 6 [1], keeping the accumulator value in the pipeline between reading and writing. Therefore, two chains issued **two cycles apart** can share one register. The bottom lane shows that **one multiply completes per cycle** from cycle 22 onward.

5 Benchmarking the tensor kernel on the hardware

One compute tile

L1-resident data

Compute peak: one **`VMAC.F`** per cycle at 1.8 GHz
1843

This work
1777 — 96.4%

`mlir-aie/mm_bfp.cc`
353 — 5× faster — 19.1%

Data streamed from DRAM

This work, end to end
1518 — 82.4%

`mlir-aie/mm_bfp.cc`, end to end
344 — 4.4× faster — 18.7%

measured theoretical GFLOPS

Whole NPU — data streamed from DRAM

Compute peak: 32 cores, one **`VMAC.F`** per cycle
58,982

Memory ceiling: 386 FLOPs/B (on-chip AI) × 55.3 GB/s (DRAM bandwidth)
21,357 — 36.2%

This work, end to end
21,352 — 100.0% — 36.2%

`mlir-aie/mm_bfp.cc`, end to end
10,549 — 2× faster — 49.4% — 17.9%

measured theoretical GFLOPS

End-to-end benchmarks compute a pre-tiled M3840×N3840×K8192 (BFP16 in, FP32 out) matrix multiplication. The **`mm_bfp.cc`** kernel [2] was adapted to a FP32 output to match our kernel and built with the Ryzen AI 1.3.1 **`xchesscc`** [3]. Measured on an **AMD Ryzen AI Max PRO 390** (Strix Halo) at its **1.8-GHz** turbo clock.

References

[1] Xilinx. *llvm-aie* ("Peano") — AIE2P register classes and scheduling model. github.com/Xilinx/llvm-aie.
[2] AMD. *mlir-aie* — BFP16 matmul kernel **`mm_bfp.cc`** (commit #d71539a). github.com/Xilinx/mlir-aie.
[3] AMD. Ryzen AI Software (version 1.3). https://ryzenai.docs.amd.com/en/1.3/.
[4] Anthropic. *Claude* — AI assistant. https://claude.ai.

Acknowledgment. The authors used Claude [4] to assist with writing and editing this poster. The authors reviewed and verified all content and take full responsibility for it.

Reproduce

```
git clone https://github.com/scalable-analyses/xdna
cd xdna && make run
```

Needs **`NPU2=1`**. First run **`xrt-smi configure --pmode turbo`**.
Docs, kernel, and harness: **tnzr.org/xdna**

Contact

tamino.steinert@uni-jena.de
https://scalable.uni-jena.de
Matrix: #scalable:uni-jena.de
SC26 Research Posters, 17–19 Nov 2026

FRIEDRICH-SCHILLER-UNIVERSITÄT JENA