POSTER: Steady-State IR for Pipelined Dataflows on AMD XDNA™NPUs

Joren Dumoulin
MICAS-ESAT
KU Leuven
Leuven, Vlaams-Brabant, Belgium
joren.dumoulin@kuleuven.be

Arne Symons
MICAS-ESAT
KU Leuven
Leuven, Vlaams-Brabant, Belgium
arne.symons@kuleuven.be

Erika Hunhoff
AMD
Longmont, Colorado, USA
erika.hunhoff@amd.com

Andre Roesti
AMD
Longmont, Colorado, USA
andre.roesti@amd.com

Andra Bisca
AMD
Longmont, Colorado, USA
andra.bisca@amd.com

Gagandeep Singh
AMD
Longmont, Colorado, USA
gagandeep.singh@amd.com

Kristof Denolf
AMD
Longmont, Colorado, USA
kristof.denolf@amd.com

Marian Verhelst
MICAS-ESAT
KU Leuven
Vlaams-Brabant, Leuven, Belgium
marian.verhelst@kuleuven.be

CCS Concepts

• **Software and its engineering → Compilers**.

Keywords

compilers, spatial dataflow, NPUs, pipelining

1 Introduction and Background

Spatial accelerators such as AMD XDNA [7] provide programmable compute tiles, local memories, and explicit data movement. Pipelining operators exploits inter-operator parallelism and keeps intermediates on chip but requires coordinated tiling, placement, reuse, and communication under hardware constraints.

Spatial pipelines require substantial manual design because representations for exploring tiling and resource allocation [1, 4, 6, 9] differ from those for generating executable schedules and communication [2, 10]. Designers must translate mapping decisions into buffer lifetimes, synchronization, and data movement.

We introduce *Steady-State IR* to bridge this gap (Figure 1). It captures one repeating iteration of computation and communication; tensor iteration spaces specify tiling, reuse, and synchronization for full execution. This shared representation carries coordinated mapping decisions from exploration through lowering, reducing manual work.

2 Steady-State Workload Representation

In SwiGLU [8] (Figure 2), matrix multiplications, SiLU activation, and elementwise multiplication process successive tiles. Steady-State IR keeps this operator graph, adds explicit transfers, and describes one iteration. Repeated execution produces the full result; reductions may need several iterations per output tile.

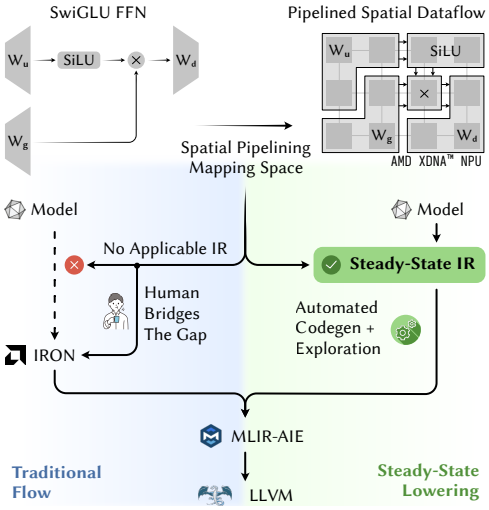


Figure 1: Steady-State IR connects spatial mapping exploration to the generation of executable code.

To coordinate tiles, we identify dimensions shared across operators, called *unique dimensions*. For example, SiLU and multiplication share sequence positions and hidden features. Shared names propagate tiling choices through the pipeline, keeping producer and consumer tiles aligned.

A tensor's iteration space describes processing across cores and time. In Figure 2, *kernel* factors specify the 24×32 tile consumed per kernel call. *Spatial* factors split the tensor 4×2 across eight cores; *temporal* factors specify successive tiles. *Absent* factors track loops that leave the tensor unchanged, such as the final projection's output-feature loop for an upstream tensor.

This work is licensed under a Creative Commons Attribution 4.0 International License.

PACT '26, Chicago, IL, USA

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2915-7/2026/10
https://doi.org/10.1145/3838684.3846851

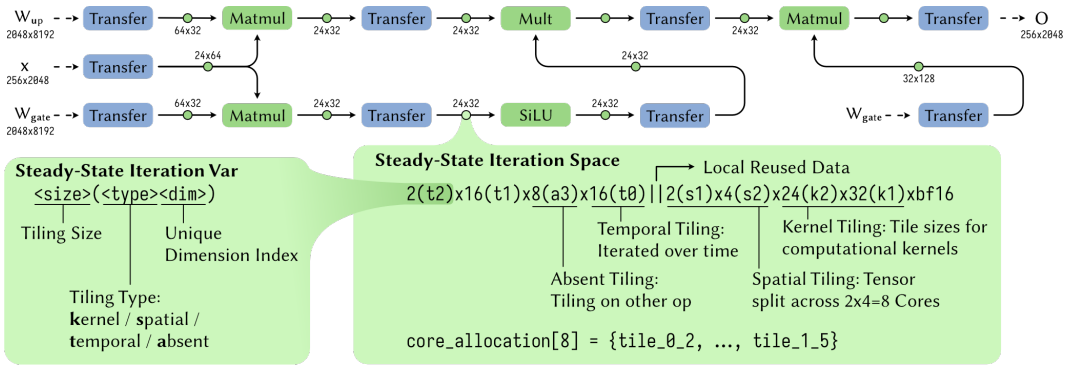


Figure 2: The steady-state graph makes computation and transfers explicit; tensor iteration spaces encode spatial and temporal tiling, local kernel tiles, and data reuse.

The reuse boundary || specifies which tiles stay in local memory, trading buffer space for fewer transfers. Retaining a weight tile across several sequence tiles in a GEMM, for example, avoids reloading weights. Exploration can thus balance memory use against communication.

These annotations also coordinate operators. Multiplication consumes matching input tiles. A GEMM retains its output tile across successive reduction inputs, releasing it after accumulation. Comparing tensor spaces at transfer endpoints determines whether data must be distributed across cores, joined, or broadcast.

Exploration selects tiling, placement, and reuse under memory and communication limits. Our MLIR-based compiler [3] lowers spatial factors to per-core instances, temporal factors to loops, and transfers to MLIR-AIE ObjectFIFOs [5]. Buffer acquisition and release at the appropriate loop levels preserve reuse and synchronization. Compute nodes call optimized IRON kernels.

3 Evaluation

We evaluate the generated mappings on the AMD Ryzen AI 9 HX 370 with 32 XDNA™ 2 compute tiles, using a batch size of one and averaging timings over 100 repetitions. For GEMM dimensions from 512 to 4096, generated mappings achieve 0.89–1.26× the throughput of hand-optimized IRON mappings, averaging 1.04× (Figure 3). An 11-line representation replaces over 700 lines of manual mapping code while reusing optimized compute kernels.

For SwiGLU with a model dimension of 2048, a hidden dimension of 8192, and sequence lengths from 64 to 2048, the generated pipeline achieves a speedup of up to 4.3× over layer-by-layer execution (Figure 4). These results show that Steady-State IR supports competitive GEMM performance and substantial pipelining gains, connecting mapping exploration to deployment on commercial NPUs.

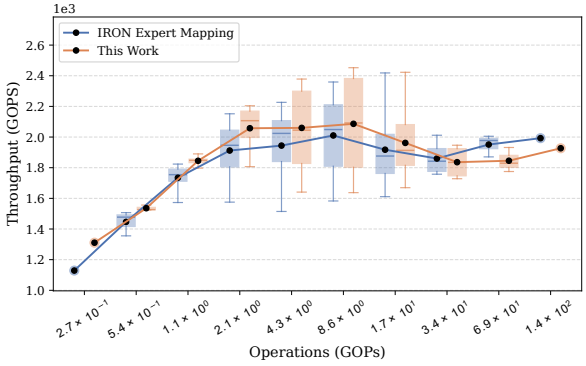


Figure 3: Generated GEMM mappings match the throughput of hand-optimized mappings on average across problem sizes.

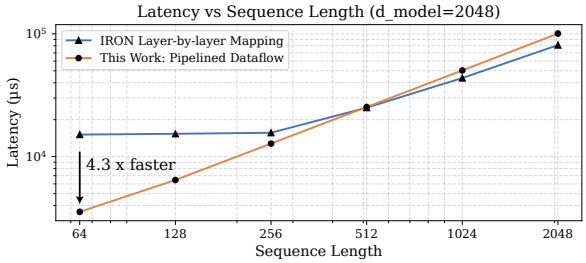


Figure 4: Spatial pipelining reduces SwiGLU latency at short sequence lengths.

Acknowledgments

This project has been partly funded by the European Research Council (ERC) under grant agreement No. 101088865.

References

[1] Qijing Huang, Minwoo Kang, Grace Dinh, Thomas Norell, Aravind Kalaiah, James Demmel, John Wawrzynek, and Yakun Sophia Shao. 2021. CoSA: scheduling by Constrained Optimization for Spatial Accelerators. In *Proceedings of the 48th Annual International Symposium on Computer Architecture (ISCA '21)*. IEEE Press, Virtual Event, Spain, 554–566. doi:10.1109/ISCA52012.2021.00050

[2] Erika Hunhoff, Joseph Melber, Kristof Denolf, Andra Bisca, Samuel Bayliss, Stephen Neuendorffer, Jeff Fifield, Jack Lo, Pranathi Vasireddy, Phil James-Roxby, and Eric Keller. 2025. Efficiency, Expressivity, and Extensibility in a Close-to-Metal NPU Programming Interface. In *2025 IEEE 33rd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, Fayetteville, AR, USA, 85–94. doi:10.1109/FCCM62733.2025.00043

[3] Chris Lattner, Mehdi Amini, Uday Bondhugula, Albert Cohen, Andy Davis, Jacques Pienaar, River Riddle, Tatiana Shpeisman, Nicolas Vasilache, and Oleksandr Zinenko. 2020. MLIR: A Compiler Infrastructure for the End of Moore's Law. doi:10.48550/arXiv.2002.11054 arXiv:2002.11054 [cs].

[4] Linyan Mei, Pouya Houshmand, Vikram Jain, Sebastian Giraldo, and Marian Verhelst. 2021. ZigZag: Enlarging Joint Architecture-Mapping Design Space Exploration for DNN Accelerators. *IEEE Trans. Comput.* 70, 8 (Aug. 2021), 1160–1174. doi:10.1109/TC.2021.3059962

[5] MLIR-AIE contributors. n.d.. MLIR-AIE: MLIR-based AI Engine Toolchain. https://github.com/Xilinx/mlir-aie/

[6] Angshuman Parashar, Priyanka Raina, Yakun Sophia Shao, Yu-Hsin Chen, Victor A. Ying, Anurag Mukkara, Rangharajan Venkatesan, Brucek Khailany, Stephen W. Keckler, and Joel Emer. 2019. Timeloop: A Systematic Approach to DNN Accelerator Evaluation. In *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, Madison, WI, USA, 304–315. doi:10.1109/ISPASS.2019.00042

[7] Alejandro Rico, Satyaprakash Pareek, Javier Cabezas, David Clarke, Baris Ozgul, Francisco Barat, Yao Fu, Stephan Münz, Dylan Stuart, Patrick Schlangen, Pedro Duarte, Sneha Date, Indrani Paul, Jian Weng, Sonal Santan, Vinod Kathail, Ashish Sirasao, and Juanjo Noguera. 2024. AMD XDNA NPU in Ryzen AI Processors. *IEEE Micro* 44, 6 (Nov. 2024), 73–82. doi:10.1109/MM.2024.3423692

[8] Noam Shazeer. 2020. GLU Variants Improve Transformer. doi:10.48550/arXiv.2002.05202 arXiv:2002.05202 [cs].

[9] Arne Symons, Linyan Mei, Steven Colleman, Pouya Houshmand, Sebastian Karl, and Marian Verhelst. 2025. Stream: Design Space Exploration of Layer-Fused DNNs on Heterogeneous Dataflow Accelerators. *IEEE Trans. Comput.* 74, 1 (Jan. 2025), 237–249. doi:10.1109/TC.2024.3477938

[10] Erwei Wang, Samuel Bayliss, Andra Bisca, Zachary Blair, Sangeeta Chowdhary, Kristof Denolf, Jeff Fifield, Brandon Freiberger, Erika Hunhoff, Phil James-Roxby, Jack Lo, Joseph Melber, Stephen Neuendorffer, Eddie Richter, Andre Rosti, Javier Setoain, Gagandeep Singh, Endri Taka, Pranathi Vasireddy, Zhewen Yu, Niansong Zhang, and Jinming Zhuang. 2025. From Loop Nests to Silicon: Mapping AI Workloads onto AMD NPUs with MLIR-AIR. doi:10.48550/arXiv.2510.14871 arXiv:2510.14871 [cs].