Modeling AIE-ML for Matrix Multiplication

Elliott Binder, Advisor: Tze Meng Low
Electrical and Computer Engineering
Carnegie Mellon University

Introduction

AIE-ML is a spatial architecture that can be found in mobile and desktop AMD processors as XDNA NPUs.

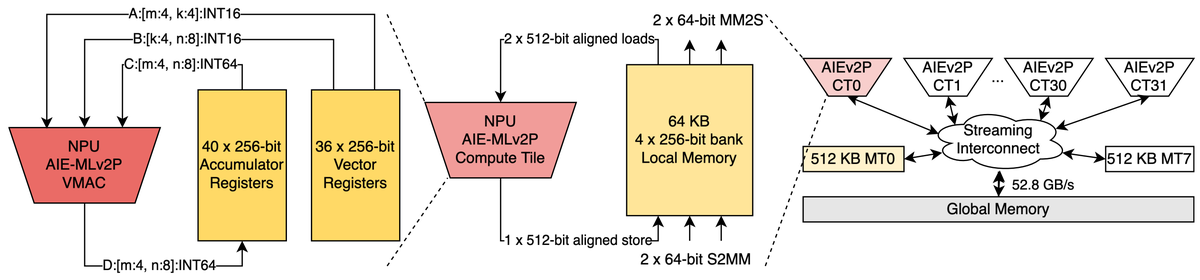
AIE-ML **differs** from Versal AIE architecture:
- Low precision (e.g. bf16, int16, int8) matrix instructions
- No programmable logic fabric
- Memory tile with more memory capacity and connectivity

XDNA 2 NPUs **improve** over previous generation:
- Higher compute throughput
- Higher interconnect bandwidth
- More compute and memory tiles

Use analytical modeling to determine how computation should be efficiently mapped to the compute and memory resources in AIE-ML architectures, with MMM as an example.

Methods

Leverage existing modeling work for designing matrix multiplication kernels for CPUs, GPUs, and distributed computing algorithms with new considerations for the structure and characteristics of compute, memory, and connectivity resources in AIE-ML architectures.
Model **hardware resources** and **characteristics** including:
- Functional unit throughput and latency
- Register capacity and layouts for MMAC instructions
- Local memory capacity, banking, throughput and latency
- Bandwidth and connectivity of each Compute/Memory tile
- Bandwidth available from main memory

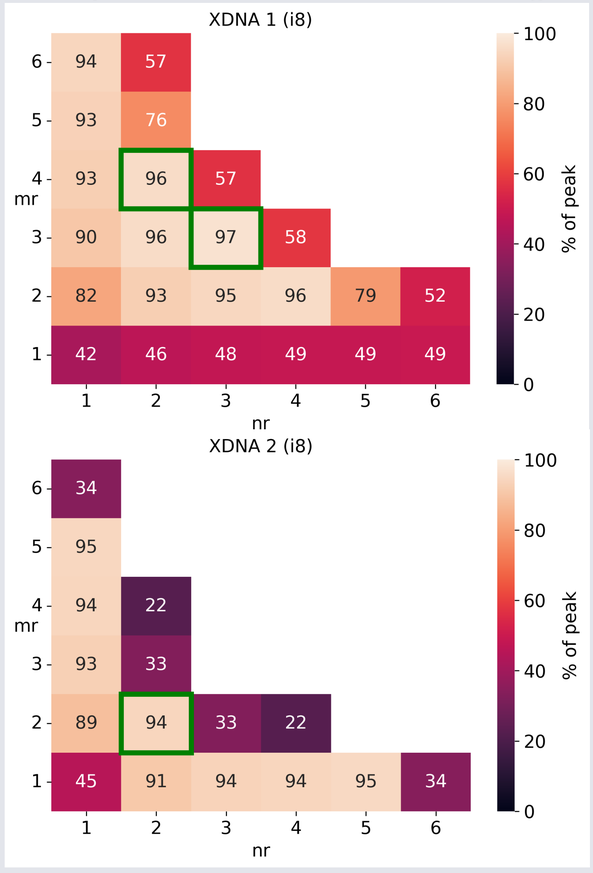


Results

Modeling for the Register Tile Size

Register tiling should be sufficient to
- hide latency of MMAC
- Hide cost of accessing local memory

Tile should be sufficiently small to
- Fit inputs into vector registers
- Fit output into accumulator registers

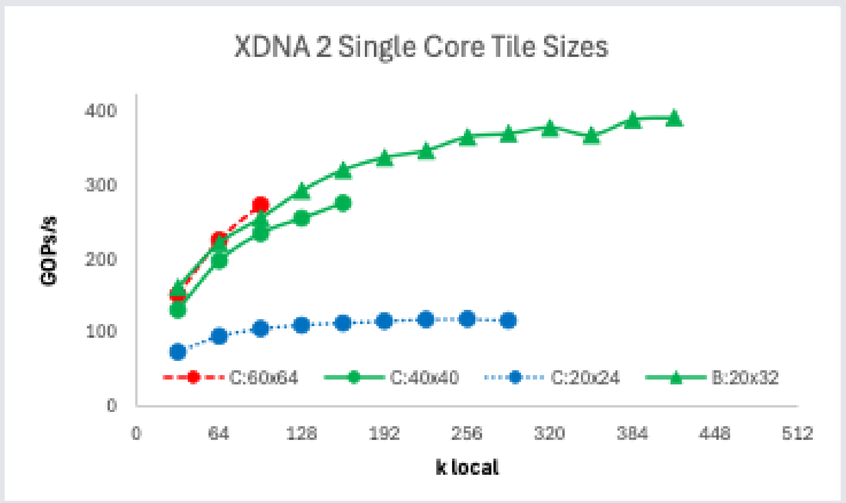


Parameters satisfying models achieve 94-97% of peak compute throughput.

Modeling for the Compute Tile Size

Local memory tiling should be sufficient to hide cost of moving tiles through streaming interface while staying within capacity.
Different resident algorithms
- input stationary (e.g. B)
- output stationary (C)

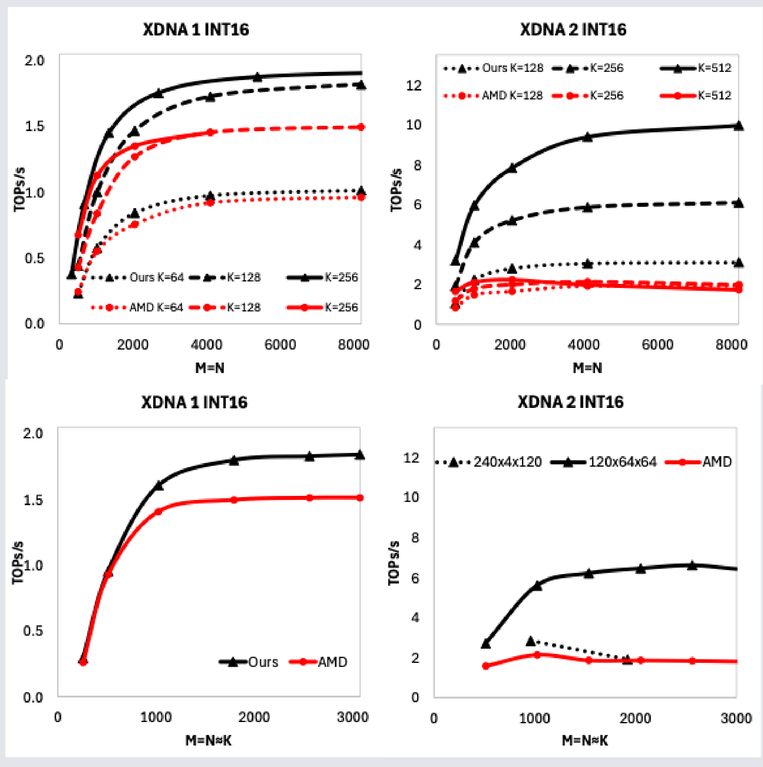
can be used for different minimum tile sizes.
Insights:
- Input stationary enables more efficient kernel via larger K tile factor
- Single-buffer stationary matrix to free capacity for larger, more efficient tiles



Choosing tile sizes just large enough to hide memory cost can enable larger K tiling values which yield higher kernel efficiency.

Modeling for the Array Tile Size

Matrices are staged in memory tiles between main memory and the array.
The array tile size should be sufficient to hide the cost of accessing non-resident matrices from main memory.
Insights:
- Input stationary algorithm more effectively uses R+W bandwidth
- Opposing skew of local tile size and parallelization should be used



Discussion

The AIE-ML compute tile can be modeled similarly to a vector/matrix CPU or GPU core when the structure of the banked local memory is accounted for distinctly (scratchpad like GPU shared memory, wide banks akin to aligning to CPU cache lines). Yet, simply treating each compute tile as an independent processor that shares a portion of memory bandwidth and connectivity is analytically unable to achieve near peak compute throughput for MMM. Thus, careful cooperation between parallel tiles is required to improve the compute efficiency when scaling to more of the array. A model-based design approach can yield good performance with insights into possible algorithmic optimizations.

Discussion (cont.)

Several improvements we'd like to see for the AIE-ML programming interface include:
- Convenient way of passing more than two arguments to a compute tile, currently limited by static mapping to streams
- Outer-loop reuse of data in memory tiles for more effective tiling at the array-level, to alleviate limited main mem. BW
- Manual control over VLIW instruction generation, including scheduling and register allocation, to improve kernel
- Larger multi-dimensional data transfers from main memory to streaming interconnect, currently limiting problem sizes

[INSERT QR CODE HERE?]