Modeling AIE-ML for Matrix Multiplication

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

Introduction

*Analytically modeling **AIE-ML** architectures to **quickly** and **efficiently** map applications to compute and memory resources.*

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

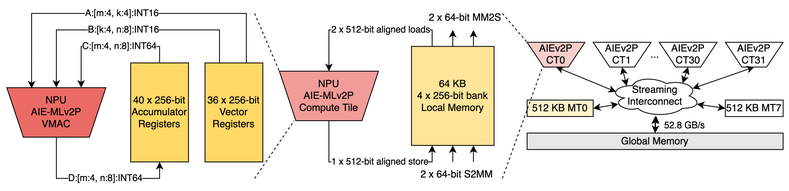
AIE-ML **differs** from Versal AIE architecture:
- Low precision (e.g. int16, int8) matrix instructions
- No programmable logic fabric
- Memory tile with more capacity and connectivity

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

Methods

Leverage models of **hardware resources**:
- Functional unit throughput and latency
- Register capacity and layouts for MMAC instructions
- Local memory structure, throughput and latency
- Bandwidth, connectivity of Compute/Memory tiles
- Bandwidth available from main memory

to simplify development effort of AIE-ML applications.



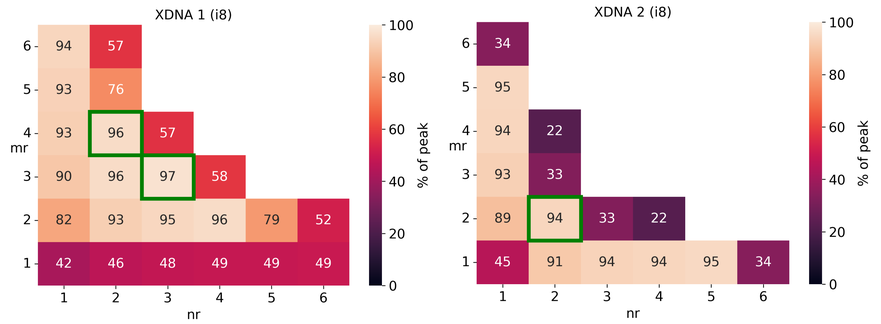
Poster

Contact

Results

Modeling the Register Tile Size

Register tile should
- Hide latency of MMAC
- Hide loads from local memory
- Fit inputs into vector registers
- Fit output into acc. Registers

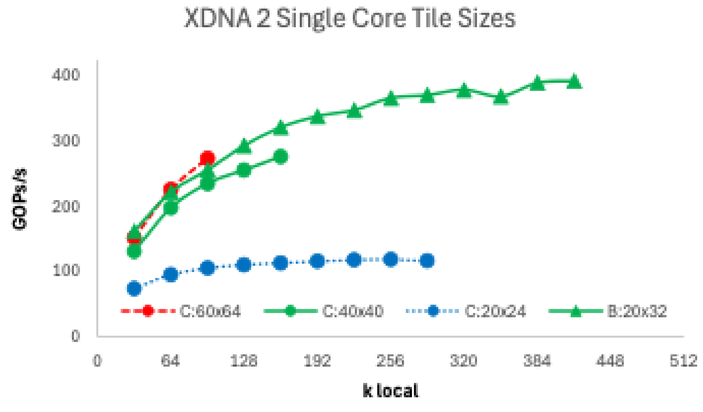


Parameters satisfying models achieve 94-97% of peak compute throughput.

Modeling the Compute Tile Size

Local memory tile should
- hide cost of streaming interface
- fit within local memory capacity

Insights:
- Input resident algorithm enables more efficient kernel
- Single-buffer stationary matrix to free capacity for larger, more efficient tiles

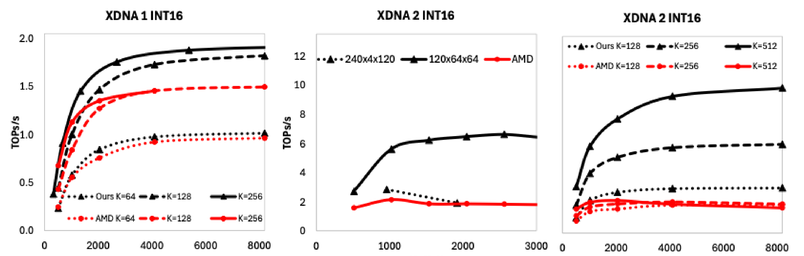


Choosing tile sizes just large enough to hide memory cost can enable larger K tiling, yielding higher kernel efficiency.

Modeling the Array Tile Size

Matrices are staged in memory tiles between main memory and the array.
The array tile size should
- hide accesses of transient matrices from memory

Insights:
- Input resident more effectively uses R/W bandwidth
- Opposing skew of local tile and parallel grid



Considering main memory bandwidth yields higher performance than SOTA.

Discussion

Our models lead to implementations that can achieve 90+% of compute peak even when matrices reside in main memory. Yet, some scenarios present no theoretically viable solution capable of achieving this performance.

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 stream mapping
- Outer-loop reuse of data in memory tiles for more effective tiling, alleviating main memory bandwidth
- Manual control of VLIW instruction generation, including scheduling and register allocation, to improve kernel
- Larger multi-dimensional data transfers from main memory to streams, currently limiting problem sizes

With Support From

DARPA
U.S. DEPARTMENT of ENERGY
AMD