Developing a BLAS library for the AMD AI Engine

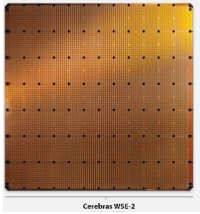
Tristan Laan, Tiziano De Matteis (t.de.matteis@vu.nl)
VU Amsterdam

H2RC – November 22, 2024

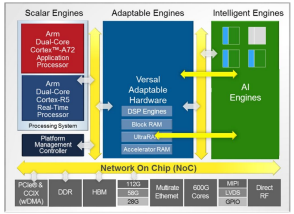
VU VRIJE UNIVERSITEIT AMSTERDAM

CAMBRIAN EXPLOSION OF "AI" HARDWARE

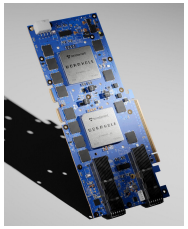
Source: Sambanova



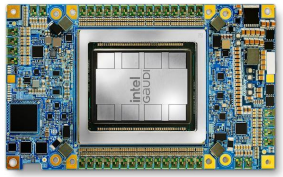
Source: Cerebras



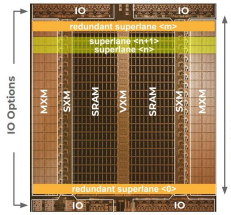
Source: AMD /Xilinx



Source: Tenstorrent

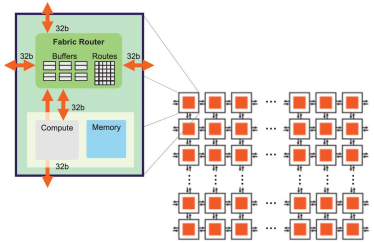


Source: Intel

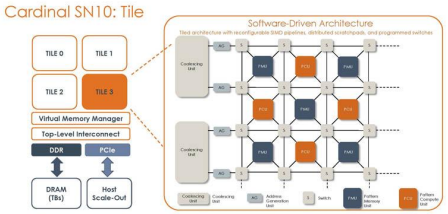


Source: Groq

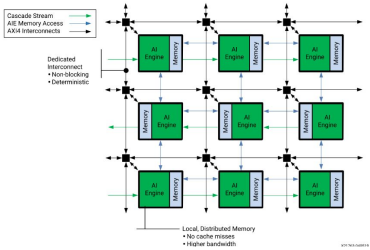
. . . .



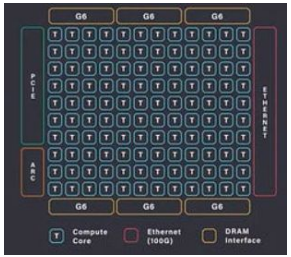
Source: Cerebras



Source: Sambanova

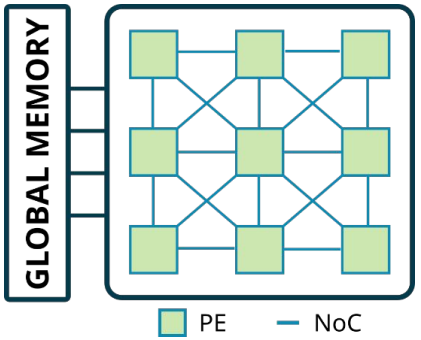


Source: AMD /Xilinx



Source: Tenstorrent

Each has its own characteristics…but they are all **spatial architectures** that can be programmed using a **dataflow approach**



Cambrian Explosion of "AI" Hardware

Source: Sambanova

Source: Cerebras

Source: AMD /Xilinx

Source: Tenstorrent

Source: Intel

Source: Groq

They are targeting ML worloads: can we use them for something else?

Can we do it **productively**?

Source: Cerebras

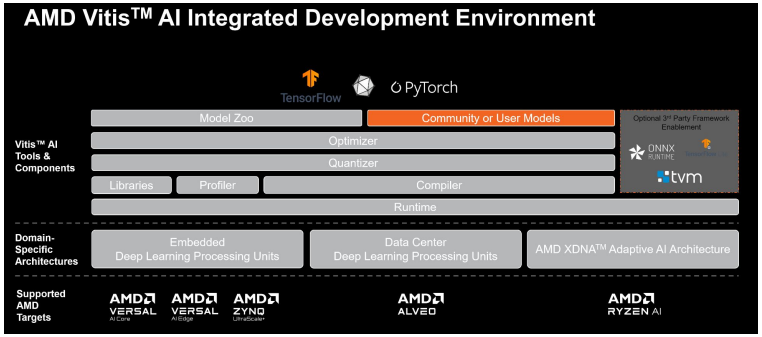

Source: Sambanova

Source: AMD /Xilinx

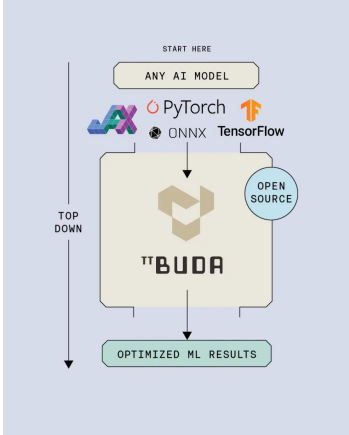
Source: Tenstorrent

THE SW ECOSYSTEMS … FOR ML USERS

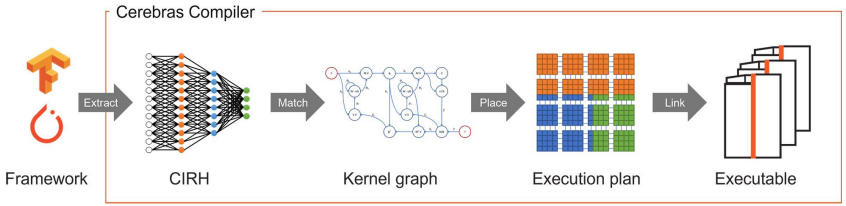
Manufacturers provide integration with **high-level ML programming frameworks** (Pytorch/TensorFlow) and pre-trained models ready-to-use



Source: AMD /Xilinx



Source: Tenstorrent



Source: Cerebras

. . . .

THE SW ECOSYSTEMS … FOR NON-ML USERS

Business logic
AMD Versal AI Engine

For all the rest, we have to rely on lower-level APIs. For instance vector add on AMD AI Engine:

```
def vector_add(x, y):
    return x+y
```

```
void vector_add(input_window<int32> *x,
input_window<int32> *y, output_window<int32> *out,
int N) {
 for (unsigned i = 0; i < N / 16; ++i) {
```

```
class simpleGraph : public graph {
private:
    kernel vadd; input_plio x, y; output_plio out;
public:
  simpleGraph() {
    vadd = kernel::create(vector_add);
    source(vadd) = "kernels/vadd.cpp";
    x = input_plio::create("x", plio_32_bits, "d/x.txt");
```

A total of 300+ lines of code to simply add two vectors. This requires knowledge about the HW, the API, how to optimize, …

```
#include <hls_stream.h>
#include <ap_axi_sdata.h>

extern "C"{
    void mm2s(ap_int<32> *mem, int size, hls::stream<qdma_axis<32, 0,
0, 0>> &s){
#pragma HLS INTERFACE m_axi port = mem offset = slave bundle = gmem
#pragma HLS interface axis port = s
#pragma HLS INTERFACE s_axilite port = mem bundle = control
…
```

..and:

- Configuration
- Placement info
- Compilation files
- …

How to avoid an "Hardware Lottery"

Despite the promise of massive parallelism, **the scientific and HPC communities have yet to systematically explore the use of spatial devices in areas other than ML**

We need proper programming abstractions, open-source libraries of reusable components, and guidelines to democratize access to ML Accelerators

Our work-in-progress project is **AIEBLAS**, a an open-source implementation of Basic Linear Algebra Routines (BLAS) for the AMD AI Engine (AIE) spatial architecture. Our goals are:

- To facilitate the rapid developments of numerical applications
- Leveraging the hardware unique characteristics
- Offering a library that can be easily extend

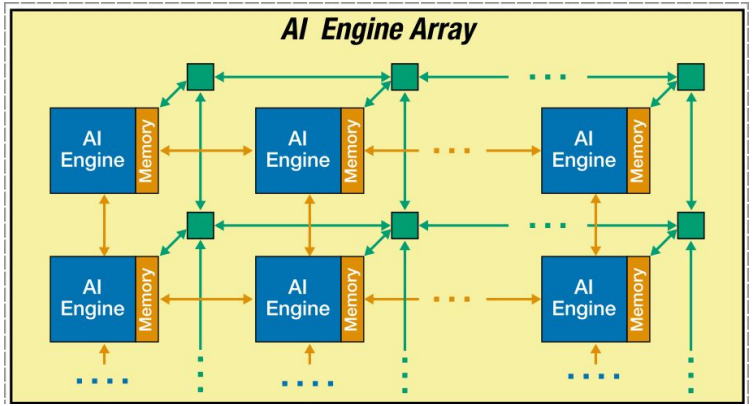
https://github.com/atlarge-research/AIE-BLAS

Dense Numerical Algebra on AMD AI Engines

The AMD AI Engines are being offered in data center acceleration card, and commodity CPUs[1]

Let's consider a VCK5000 (data center card):

- An array of 8x50 AIEs (400 in total):
  - Each one with 32 KB local memory and a VLIW vector processor
  - Each can communicate with neighbours or non-neighbour AIEs
- Programmable Logic (PL) for custom hardware



The AIEs can be programmed using the Adaptive Dataflow (ADF) API, the application is represented by a dataflow graph of kernels scheduled one the AIEs.

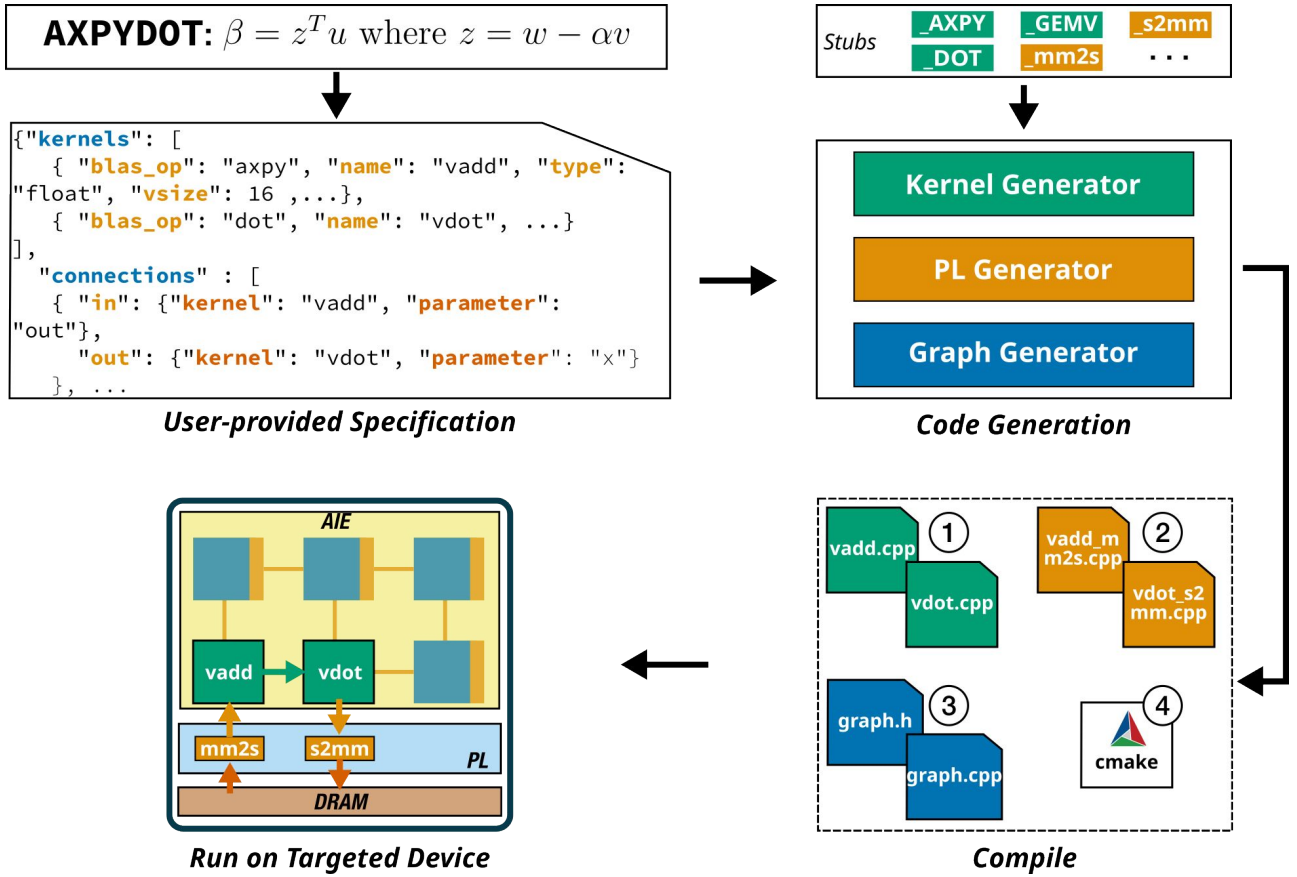
[1] https://www.amd.com/en/technologies/xdna.html

@Large Research
Massivizing Computing Systems
http://atlarge.science

This work was supported in part by AMD under the HACC program

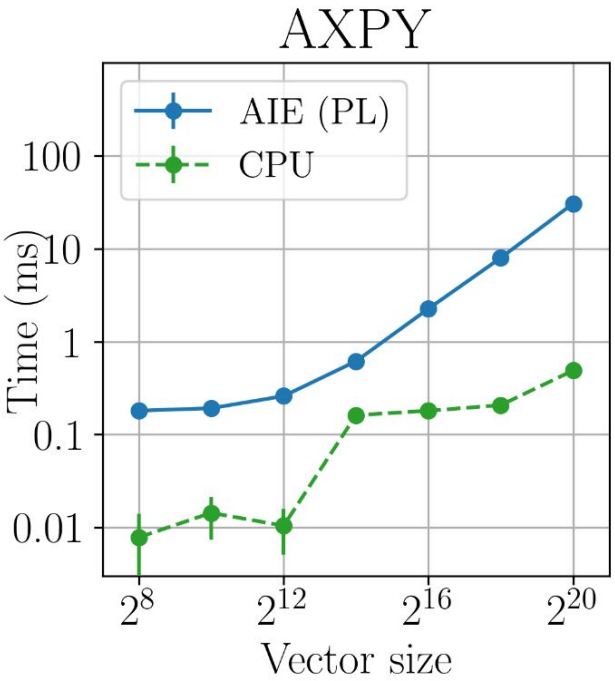
VU VRIJE UNIVERSITEIT AMSTERDAM

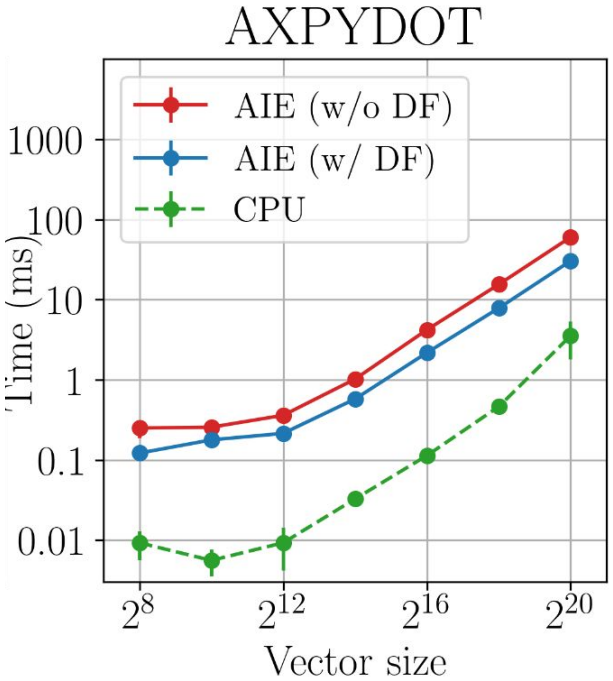
AIE-BLAS: A BLAS LIBRARY FOR THE AMD AI ENGINE (WIP)



First Results

Comparison VCK5000 against OpenBLAS running on a 2x10-cores Xeon Silver 4210R.

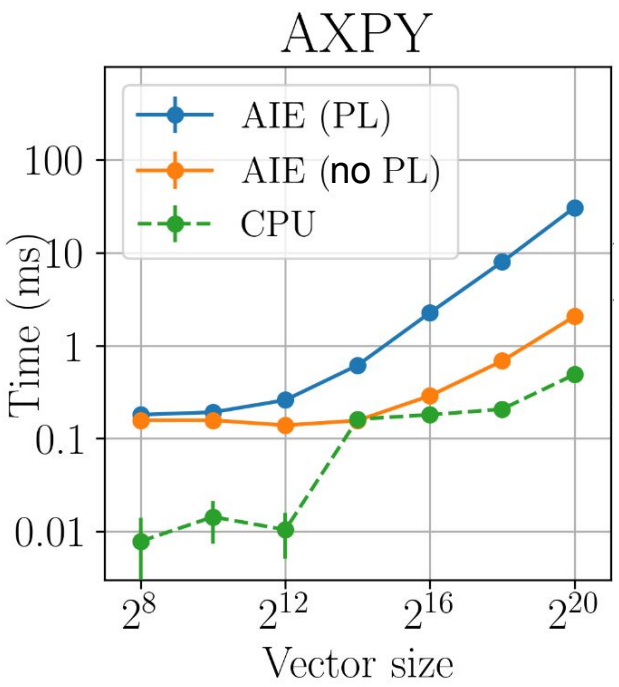


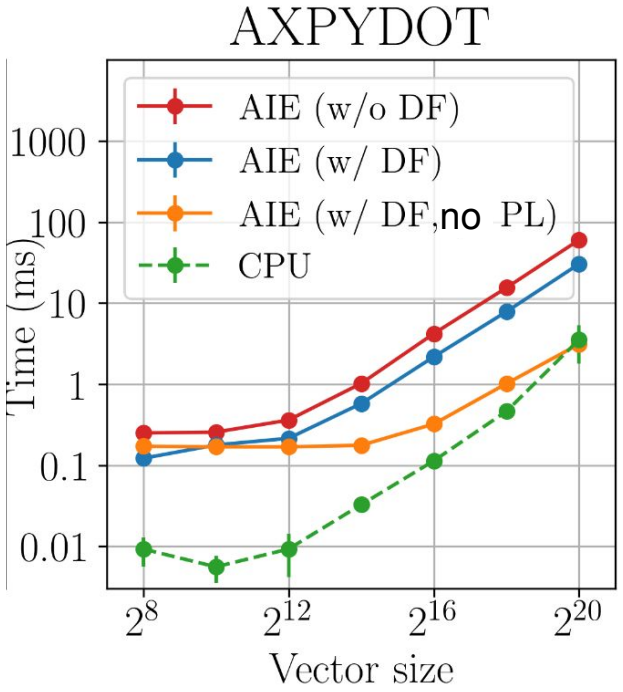


Similar behavior with GEMV

First Results: W/O Memory Accesses

Comparison VCK5000 against OpenBLAS running on a 2x10-cores Xeon Silver 4210R.

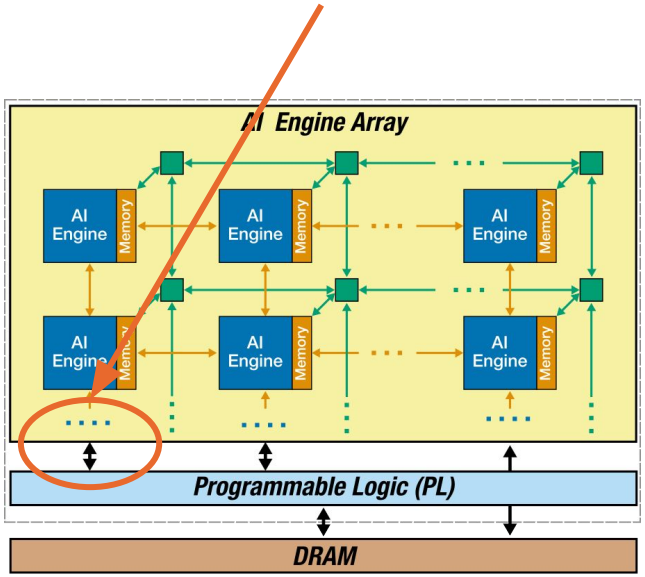




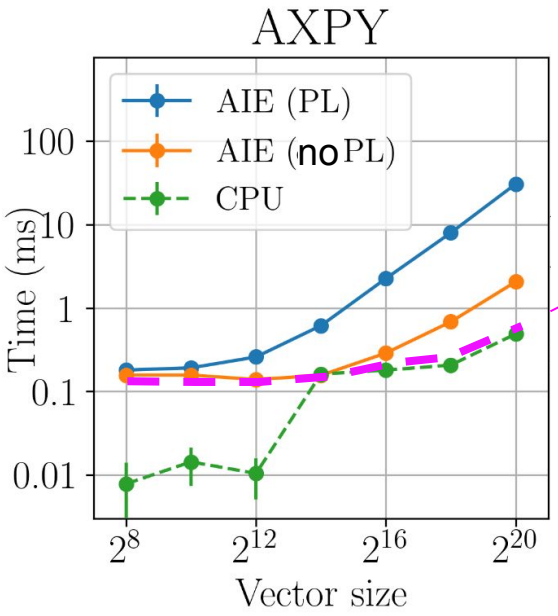
Accessing off-chip memory is non trivial

Leveraging Multiple AIEs

Each PL-AIE connection supports 4 GB/s (312 connections in total)



Multi-AIEs implementation is necessary



Exp. result (4 AIEs)

AIE-BLAS: Next Steps

Write good code for spatial accelerator is complicated also for experts. Next steps:

- Support for multi-AIE implementation
- Optimize memory accesses and tiling
- Favor dataflow composability
- Coverage
- Considering to other spatial architectures

A BLAS porting for AMD AI Engine architecture, to **facilitate the development of numerical applications**

https://github.com/atlarge-research/AIE-BLAS

THANK YOU!