In this project, each member of our team has been focused on a specific hardware platform. In particular:

**Endri Taka** is responsible for programming and evaluating the performance of

AMD/Xilinx AI Engines (Versal VCK190).

● **Andres Fadul** is responsible for programming and evaluating the performance of Intel AI tensor blocks in Stratix 10 NX FPGA

● **Hung-Yueh Chiang** is responsible for programming and evaluating the performance of

the NVIDIA A100 GPU with CUDA.

**AMD/Xilinx AI Engines (Versal VCK190)**

Directory “*Xilinx\_AI\_Engine\_codes/*” contains our code for programming and mapping matrix multiply applications on both single AI Engine (directory “*matrix\_mult\_single\_AIE/*”) and multiply AI Engines mapping (directory “*matrix\_mult\_multiple\_AIEs/*”).

In both single and multiple AI Engines case, the following source files (in C++) are used to program the AI Engines:

* “**kernels.cc**”, which contains the matrix multiply kernel running on one AI Engine. In the multiple AI Engines case, we also have implemented a vectorized addition kernel which calculates the partial products of the blocked matrix multiplications.
* **“project.h”**, where the Adaptive Data Flow (ADF) graph is implemented. The ADF determines the mapping on AI Engines and the communications among the kernels.
* **“project.cpp”**, which defines the graph and runs the application on AI Engines.
* **“include.h”**, is the header file which contains the (constant) sizes of the matrices to be multiplied. We alter these values to experiment with various sizes throughout our experiments.
* **“kernels.h”**, which contains the function declaration of our implemented kernels.

Moreover, we have written the following scripts to facilitate our exploration and calculate various performance metrics. More specifically,

* **“golden\_int8.cpp”** generates our input data (random), performs the matrix multiplication, and generates the output data (golden data), which we use to compare with the output of the AI Engine to verify its correct operation.
* **“analyze\_exec\_time.py”** is a python script which we utilize to calculate latency and throughput performance metrics.
* **“active\_cores.py”** is utilized only in the multiple AI Engines mapping to calculate the active numbers of AI Engine and the active memory tiles.

Finally, notice that we performed our experiments using the Vitis 2022.1 version. To start programming the AI Engines and become more familiar, you can follow these basic tutorials: <https://support.xilinx.com/s/question/0D52E00006xR6iXSAS/ai-engine-blog-series?language=en_US>

**Intel/Altera AI Tensor Blocks (Stratix 10 NX)**

Directory “*Intel\_Tensor\_blocks\_codes/*” contains our code for programming matrix multiply applications on Tensor Blocks. The following source files (in verilog) are used to program the AI Engines:

* “**top.v**” latest top file where everything is put together, including controller and Intel generated tensor blocks.
* **“tb.v”** latest testbench we used to test and load matrices onto the tensor blocks
* **“Tensor\_in”**, Folder generated by Quartus when configuring intel IP, contains the source code for input tensor blocks and simulation scripts.
* **“Tensor\_test”**, Folder generated by Quartus when configuring intel IP, contains the source code for regular tensor blocks and simulation scripts.

**NVIDIA/A100 GPU**

Directory “A100\_GPU/” contains our profiling code for A100 GPU. There are three directories under the folder:

* ours/
  + cpu\_mm.cpp cpu\_mm.hpp

they contain self-implemented matrix multiplication with CPU.

* + gpu\_cuda\_mm.cu gpu\_cuda\_mm.hpp

they contain self-implemented matrix multiplication with GPU CUDA Cores.

* + main.cpp

The main function.

* + Makefile

The makefile to compile the codes.

* tensor\_core/
  + cublasTensorCore.cu

The floating point (F16, F32) matrix multiplication using Tensor Cores

* + cublasTensorCore\_int8.cu

The integer 8 matrix multiplication using Tensor Cores

* + Makefile

The makefile to compile the codes.

* spmma/
  + spmma\_example.cpp

The sparse matrix multiplication and dense matrix multiplication implementation

* + CMakeLists.txt Makefile

The makefile and CMakelists to compile the codes with cuSPARSELt library