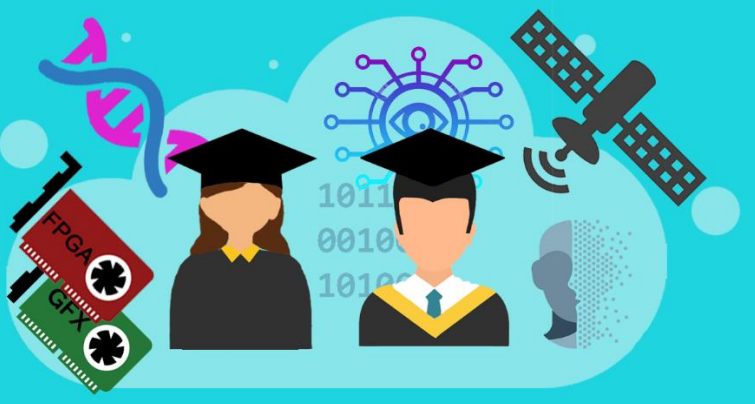
Diploma Thesis

Microproccessors and Digital Systems Laboratory



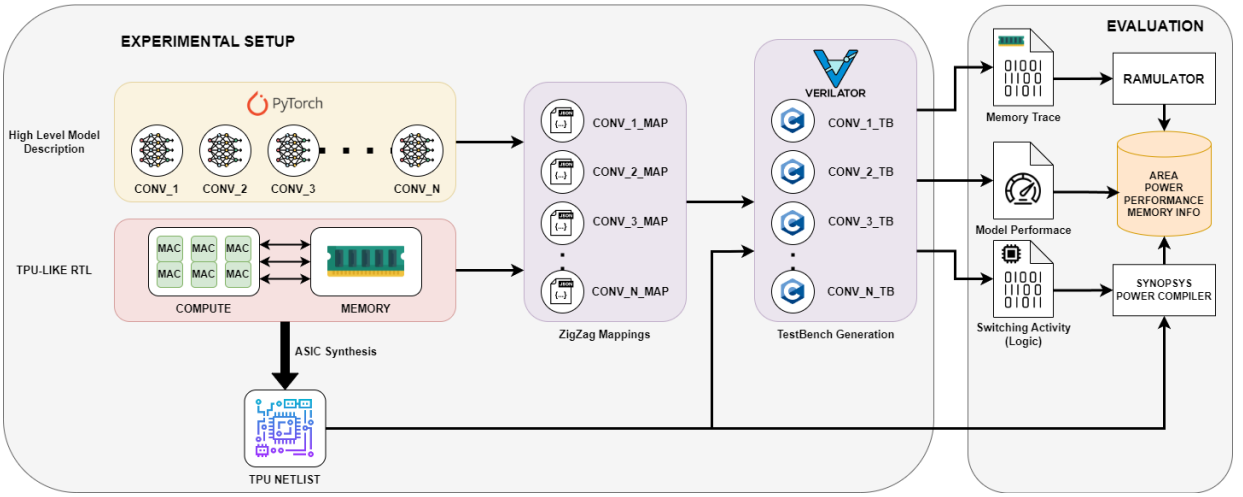
Energy-Based Behavioural Modelling Tool

High-cost training and resource-hungry implementation led the research community to find new and state-of-the-art solutions to make high-performance and at the same time low energy (and area) solutions to those problems. In this thesis, the student will implement a framework that can take proper code suggestions in several NN model layers (such as GeMM and Convolution) with ZigZag, taking a TPU-like accelerator's architecture and memory hierarchy as input, and evaluate them based on real RTL implementations.

Why We Need Modelling?

Most of the simulation tools that are already in use consume a lot of time and a lot of resources to produce a result, which in a multiple-parameter DSE (Design Space Exploration) is not optimal. For those purposes, modelling lets the user have a standardized description of the high-level parts of an accelerator describing only the aspects of it (Power, Area, Precision) without the RTL being present. This can lead to a significant reduction in time and resources when a DSE is executed in several NN accelerators and models.

Implementation Flow



Experimental Setup

In the experimental setup, the student will

- Test different TPU-like implementations
- Test different memory hierarchies in the architecture
- Produce netlist (Synopsys synthesis flow) through the implementations
- Run the ZigZag evaluation flow and produce results
- Map the ZigZag produced results (JSON file) as (a Verilator-based) C++ testbenches

Evaluation

Having the testbenches and the TPU-like accelerator netlist, in the evaluation step

- Using the **Synopsys Power Compiler**, Power and Area data will be produced
- Either with **Verilator** (high level) or with Behavioural Simulators, the performance data will be produced
- Gathering the memory access patterns and running the **RAMULATOR 2.0**, memory data will be produced
- Use of state-of-the-art technologies (**22nm**)

This flow will evaluate the accelerator against baseline (unoptimized implementations) and other state-of-the-art TPU implementations (e.g. **Google's Edge TPU** [4]).

Toolchains Used

This thesis will take advantage of multiple toolchains, here are presented some of them that you may have not heard of.

ZigZag[1] (GitHub repo)

ZigZag[1] is a Python-based framework developed by the MICAS group at KU Leuven for optimizing deep learning accelerators. It focuses on energy-efficient hardware design and mapping of neural network computations. ZigZag[1] automates the exploration of hardware configurations, balancing performance and energy use by considering hardware constraints like memory and compute resources. Its flexibility allows it to target a range of hardware, making it ideal for both academic and industrial AI applications.

This tool takes as inputs the description of the interconnection between different MAC components in an array, as well as the energy needs of each, the ONNX NN model, and the memory hierarchy of the accelerator. The produced result is an optimized tiling implementation of the loops inside supported layers.

Verilator (site)

Verilator functions similarly to GCC or Synopsys's VCS, taking in Verilog or SystemVerilog code and "Verilating" it. This process involves reading the code, performing lint checks, and optionally adding assertion checks and coverage-analysis points. Verilator generates single- or multithreaded C++ or

SystemC code, producing .cpp and .h files. These "Verilated" C++/SystemC files can then be compiled using a C++ compiler (such as gcc, clang, or MSVC++). Users may include their own C++/SystemC wrapper files to instantiate the Verilated model. Running the resulting executable simulates the design. Additionally, Verilator allows linking its generated libraries, including encrypted ones, with other simulators.

Synopsys ASIC Design Flow (site)

Synopsys Design Compiler is a logic synthesis tool that converts high-level hardware descriptions, such as RTL (VHDL/Verilog), into a gate-level netlist for integrated circuit (IC) design. It optimizes the design for key metrics like timing, area, and power, ensuring the circuit meets performance requirements. The tool performs static timing analysis (STA) to verify that the design adheres to timing constraints while minimizing critical paths. It also reduces area and power consumption through techniques such as gate sizing and clock gating. Additionally, the Design Compiler maps the optimized logic into technology-specific standard cells, preparing the design for physical implementation and manufacturing.

RAMULATOR 2.0 [2] (GitHub repo)

The RAMULATOR 2.0 [2] is a more optimized version of RAMULATOR [3] being a cycle accurate RAM simulator, supporting all the modern implementations of RAM, taking the traces for multiple sources.

PREREQUISITES:

- Modern NN libraries (PyTorch/ONNX)
- Compute intensive NN layer implementations (CONV, GeMM)
- Python and C++ Programming Languages
- HDLs (Verilog-VHDL)
- VLSI Systems

SKILLS YOU WILL LEARN:

- Modern ASIC implementation flows (synthesis, netlist generation)
- Deeper understanding of NN acceleration
- TPU Architecture
- Multi-tool modelling and setup implementation
- Metrics and benchmarking techniques

RELATED MATERIAL:

[1] L. MEI, P. HOUSHMAND, V. JAIN, S. GIRALDO, AND M. VERHELST, "ZIGZAG: ENLARGING JOINT ARCHITECTURE-MAPPING DESIGN SPACE EXPLORATION FOR DNN ACCELERATORS," IN IEEE TRANSACTIONS ON COMPUTERS, VOL. 70, NO. 8, PP. 1160-1174, 1 AUG. 2021, DOI: 10.1109/TC.2021.3059962.

[2] HAOCONG LUO, YAHYA CAN TUGRUL, F. NISA BOSTANCI, ATABERK OLGUN, A. GIRAY YAGLIKCI, AND ONUR MUTLU, "RAMULATOR 2.0: A MODERN, MODULAR, AND EXTENSIBLE DRAM SIMULATOR," ARXIV, 2023.

[3] KIM ET AL. *RAMULATOR: A FAST AND EXTENSIBLE DRAM SIMULATOR.* IEEE CAL 2015.

[4] SESHADRI, KIRAN, ET AL. "AN EVALUATION OF EDGE TPU ACCELERATORS FOR CONVOLUTIONAL NEURAL NETWORKS." *2022 IEEE INTERNATIONAL SYMPOSIUM ON WORKLOAD CHARACTERIZATION (IISWC)*. IEEE, 2022.

THESIS AVAILABLE: **2**

CONTACT INFORMATION:

- Georgios Alexandris, Ph.D Student, NTUA: (galexandris@microlab.ntua.gr)
- Panagiotis Chaidos, Junior Researcher, NTUA: (pchaidos@microlab.ntua.gr)
- Alexis Maras Ph.D Student, NTUA: (amaras@microlab.ntua.gr)
- Professor Dimitrios Soudris, NTUA: (dsoudris@microlab.ntua.gr)
- Assistant Professor Sotirios Xydis, NTUA: (sxydis@microlab.ntua.gr)