University of Glasgow

gicLAB

Accelerating AI at the Edge

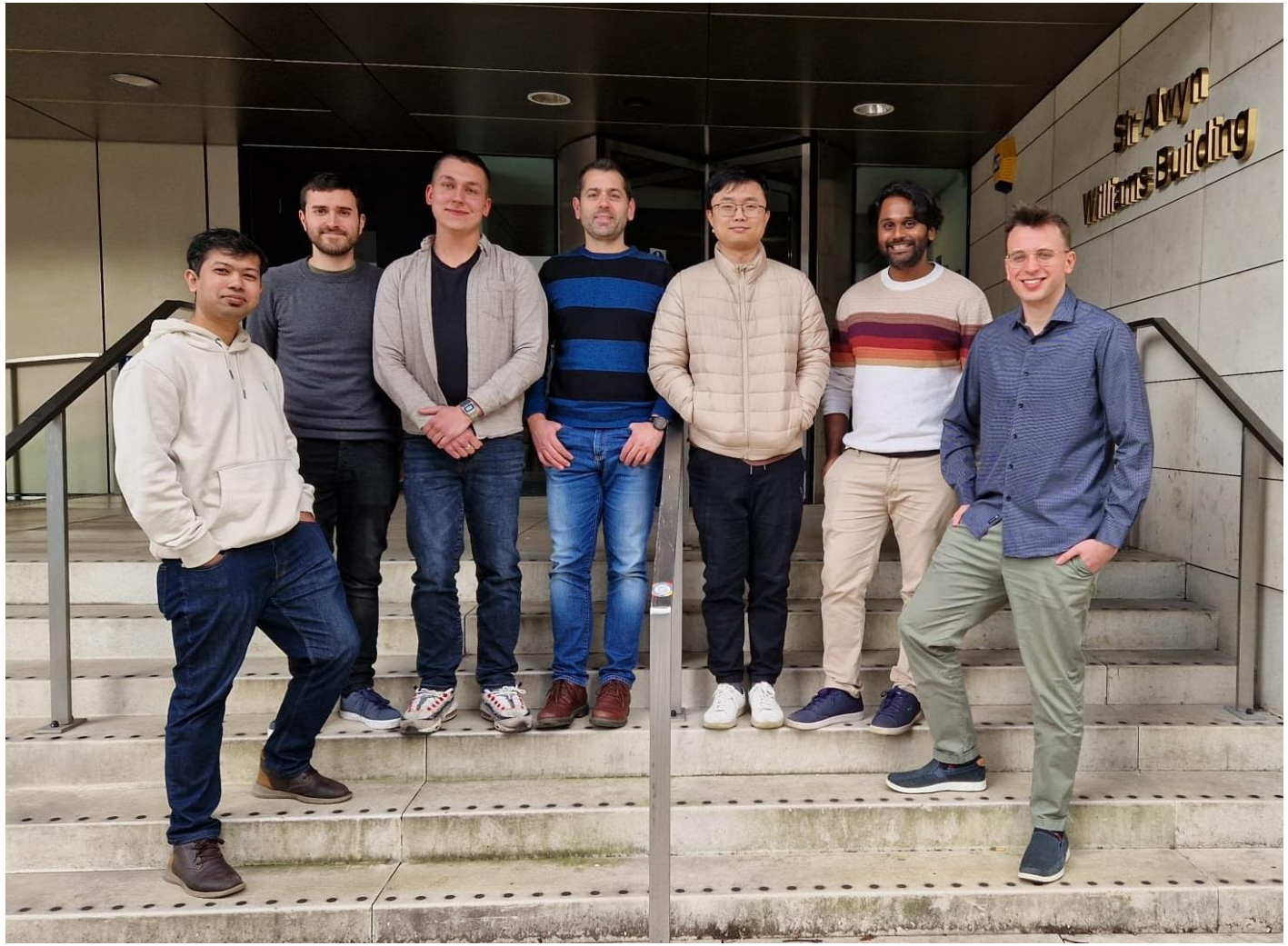
The Power of Efficient HW/SW Co-Design

José Cano
School of Computing Science
University of Glasgow, Scotland, UK

NANDA Workshop
London, 10/09/2024

Glasgow Intelligent Computing Lab (gicLAB)

University of Glasgow



School of Computing Science

Systems Section (GLASS)

Computing Systems
&
Machine Learning

gicLAB

https://giclab.dcs.gla.ac.uk/

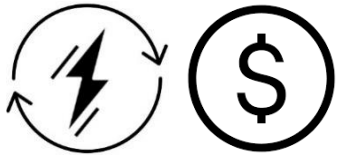
Why AI at the Edge?

- **Latency**
  - Real time constraints

- **Bandwidth**
  - Huge amount of information, 5G not enough

- **Reliability**
  - Network not always available

- **Security and privacy**
  - Better keep private/personal info locally



- **Power consumption and cost**
  - Lower at the edge

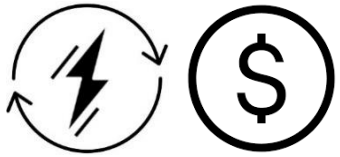
Why AI at the Edge?

- **Latency**
  - Real time constraints

- **Bandwidth**

Problem: DNNs demand lots of resources and devices have very limited capacity (compute, memory, power) !!!

- **Security and privacy**
  - Better keep private/personal info locally



- **Power consumption and cost**
  - Lower at the edge

Key concept: Deep Learning Acceleration Stack (DLAS)

University of Glasgow

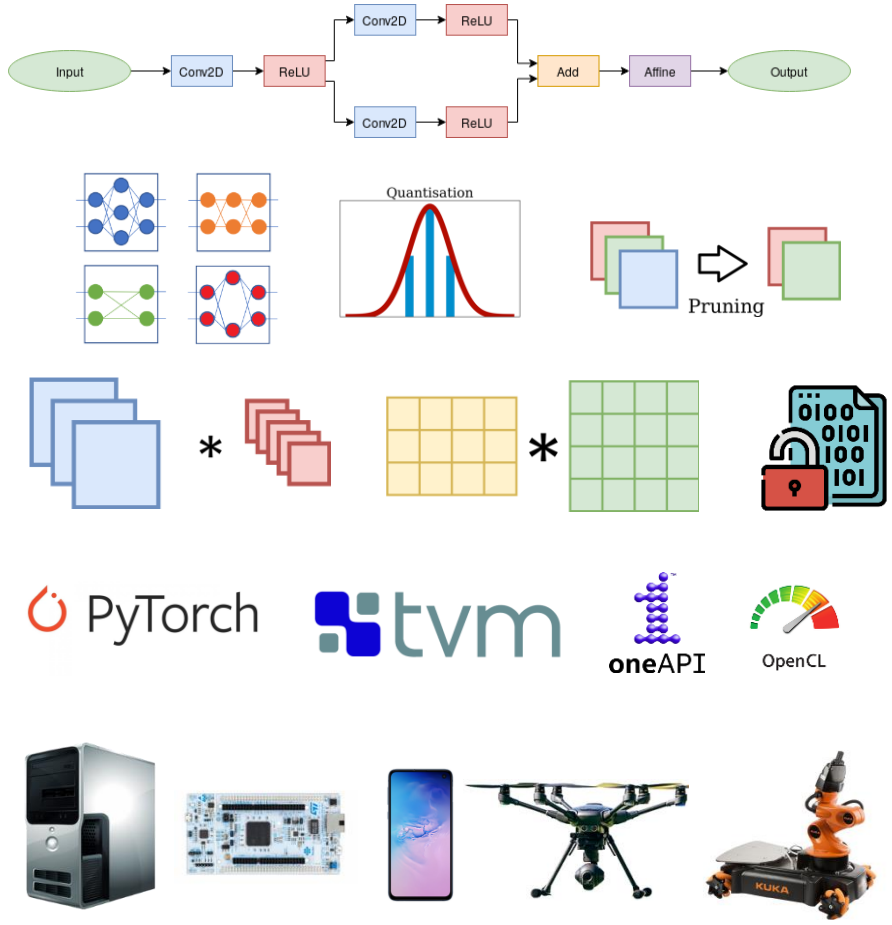
Neural Network Models & Datasets
(Image, video, voice, text, etc)

Optimization Techniques
(Pruning, quantization, NAS/HPO, etc)

Algorithmic Primitives & Data Formats
(GEMM, Winograd, CSR, Encryption, etc)

Systems Software
(Libraries, frameworks, compilers, etc)

Hardware
(Server class, Edge/IoT/Tiny devices)



*Across-stack optimizations are required to provide efficient solutions!

*[P. Gibson, J. Cano, E. J. Crowley, A. Storkey, M. O'Boyle, "*DLAS: A Conceptual Model for Across-Stack Deep Learning Acceleration*", **ACM TACO'24**]

Outline

- SECDA Methodology
- SECDA-TFLite
- SECDA-LLM
- AXI4MLIR
- Conclusions and Future Work

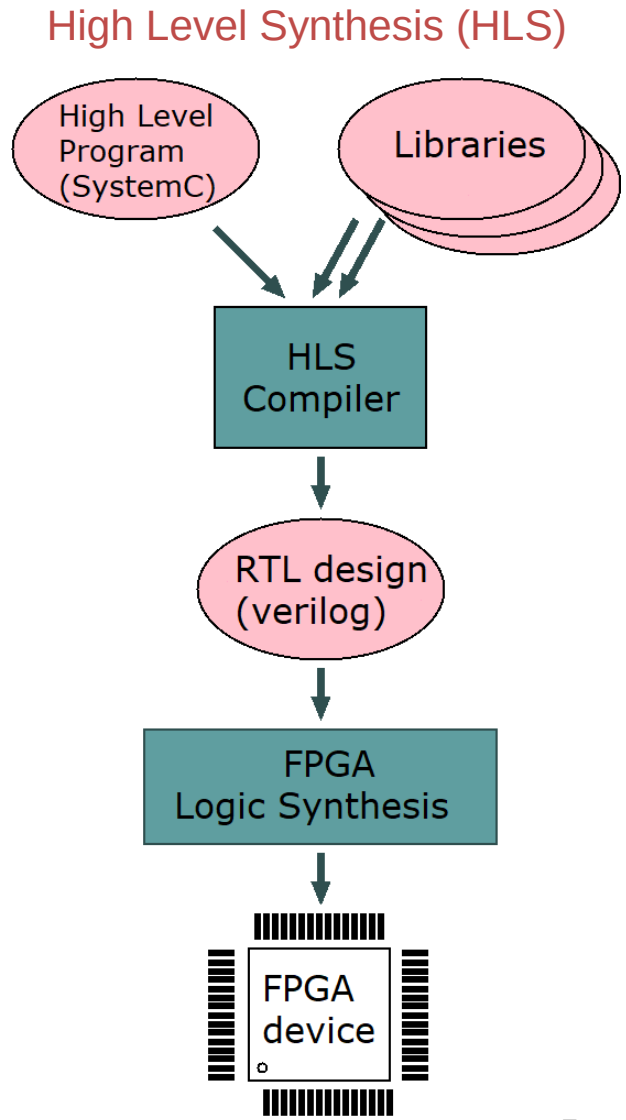
Developing Specialized Accelerators for AI

University of Glasgow

- **Motivation**: specialized hardware accelerators (ASICs, FPGAs, etc) can make AI faster and more energy efficient (e.g. at the edge)
  - FPGAs are reconfigurable circuits commonly present in edge devices

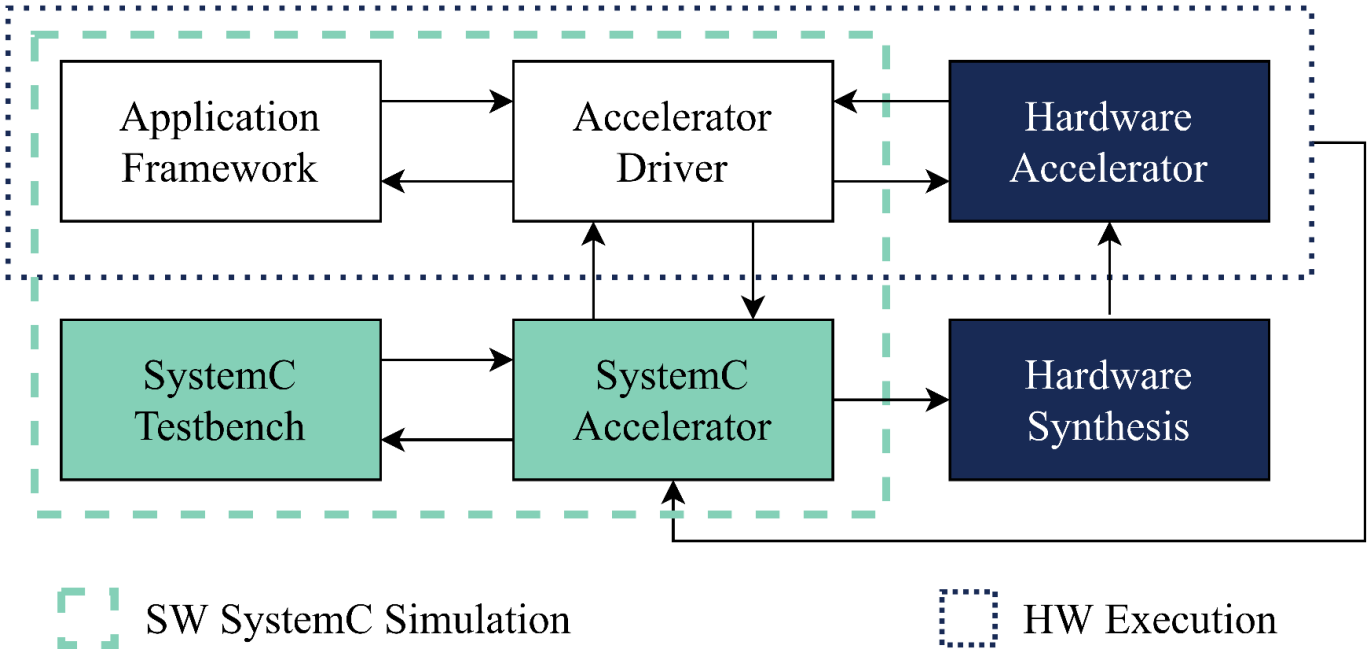
- **Problem**: current solutions for designing AI accelerators for edge devices with FPGAs have a very high development cost
  - They require High Level Synthesis (HLS)
  - FPGA synthesis is a very slow process that is repeated (over designs)
  - System integration issues (e.g. accelerator and DNN framework)

- **Solution**: we proposed a design methodology (SECDA) to efficiently reduce the development time of FPGA-based accelerators
  - Combines cost-effective SystemC simulation with hardware execution



SECDA Methodology: Overview

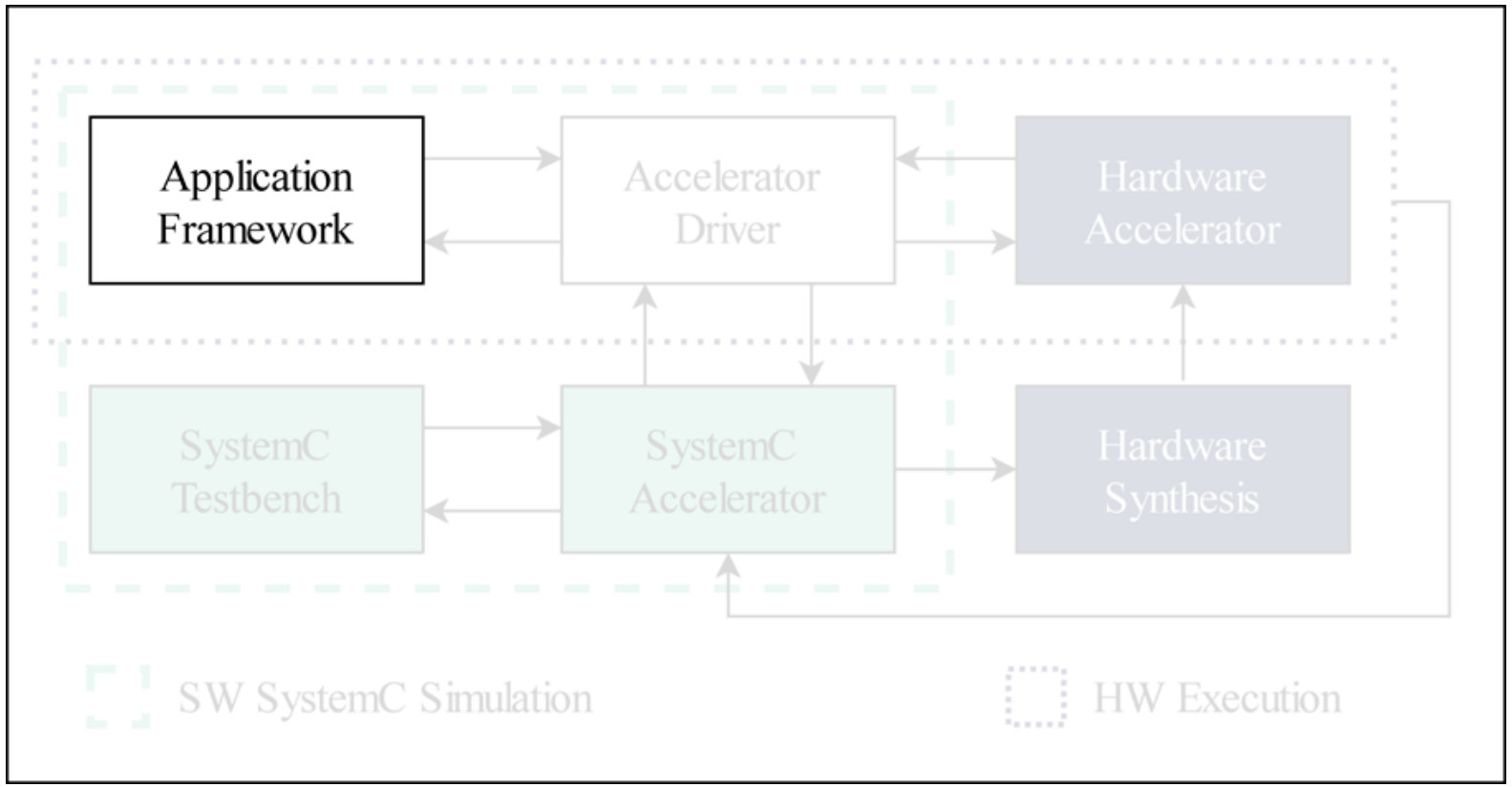
- **SECDA**: SystemC Enabled Codesign of DNN Accelerators



*[J. Haris, P. Gibson, J. Cano, N. B. Agostini, D. Kaeli, "*SECDA: Efficient Hardware/Software Co-Design of FPGA-based DNN Accelerators for Edge Inference*", **SBAC-PAD'21**]

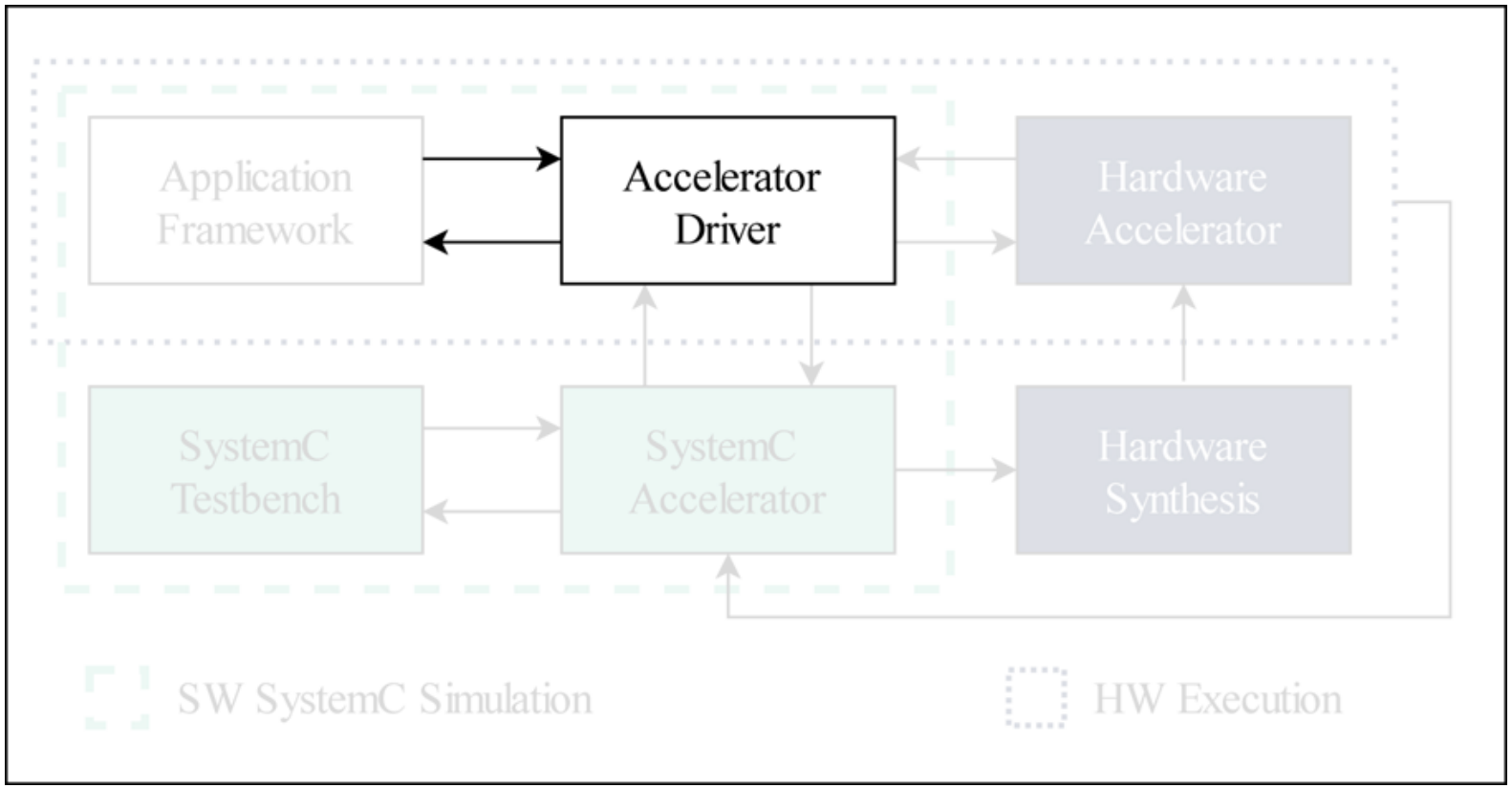
SECDA Methodology: Components

- Application Framework
  - It is able to run the target workloads (DNN models) without an accelerator (e.g. CPU)
  - Examples:
    - TFLite
    - PyTorch Mobile
    - QKeras
    - *llama.cpp*
    - …



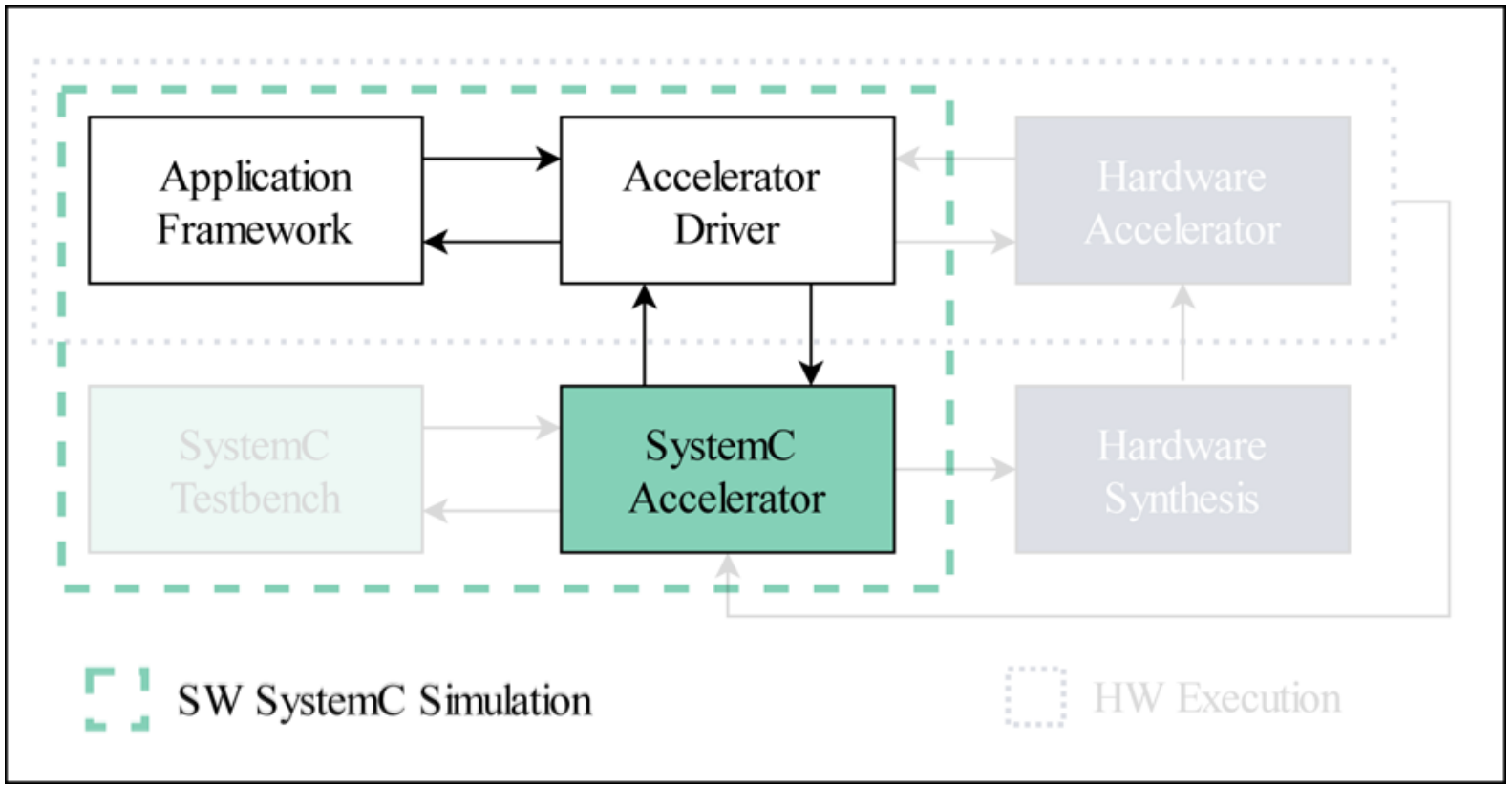
SECDA Methodology: Components (2)

- Accelerator Driver
  - Bridge between an application framework and an accelerator
  - Vital for hw/sw co-design, impacts latency and energy consumption
  - Examples
    - Data packing and unpacking
    - DMA transfers
    - …



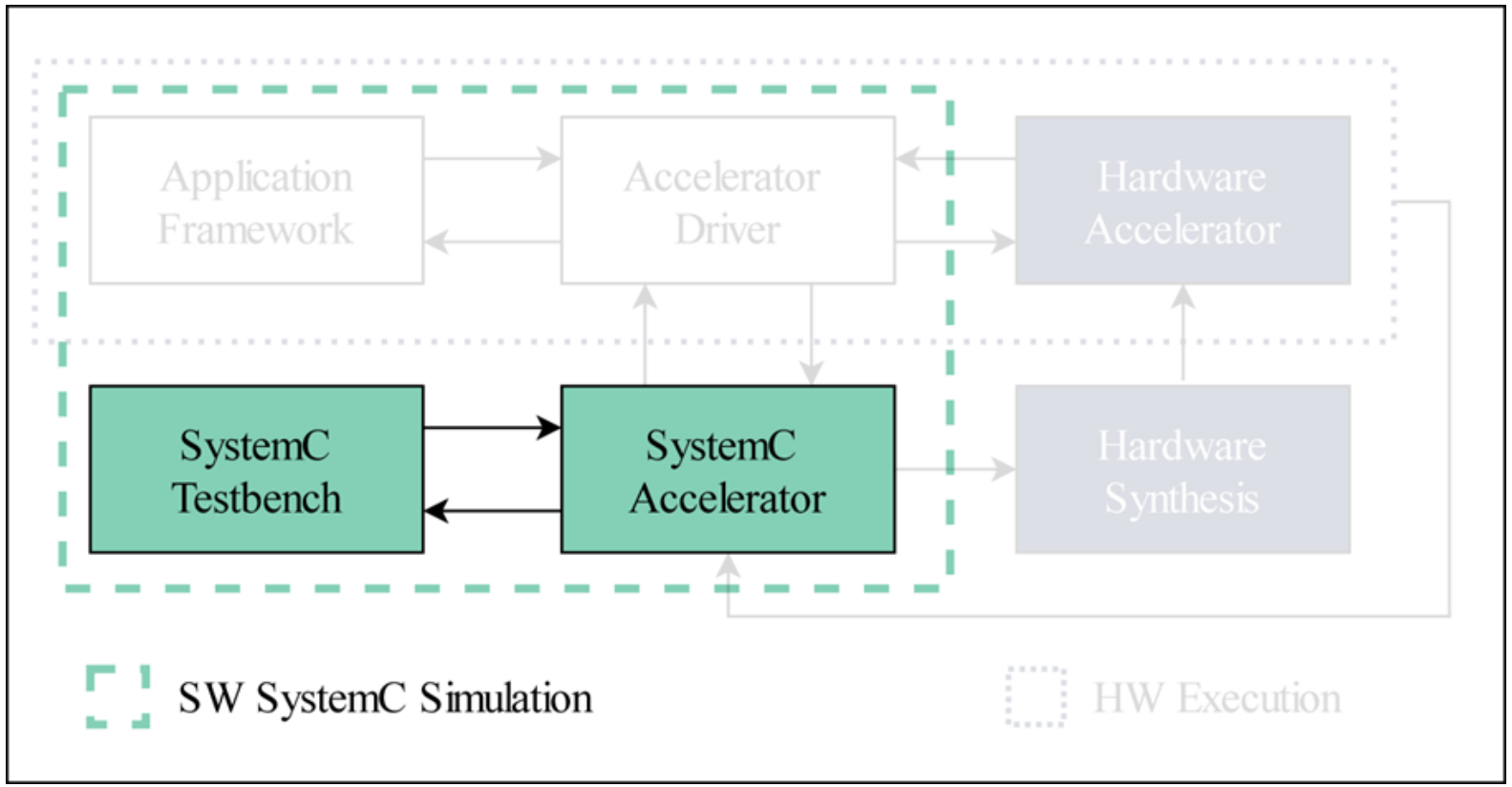
SECDA Methodology: Components (3)

- SystemC Accelerator
  - SystemC Transaction-Level Modelling
  - SystemC Simulation
  - End-to-end simulation (full DNN models)
  - Starting point for High-level Synthesis



SECDA Methodology: Components (4)

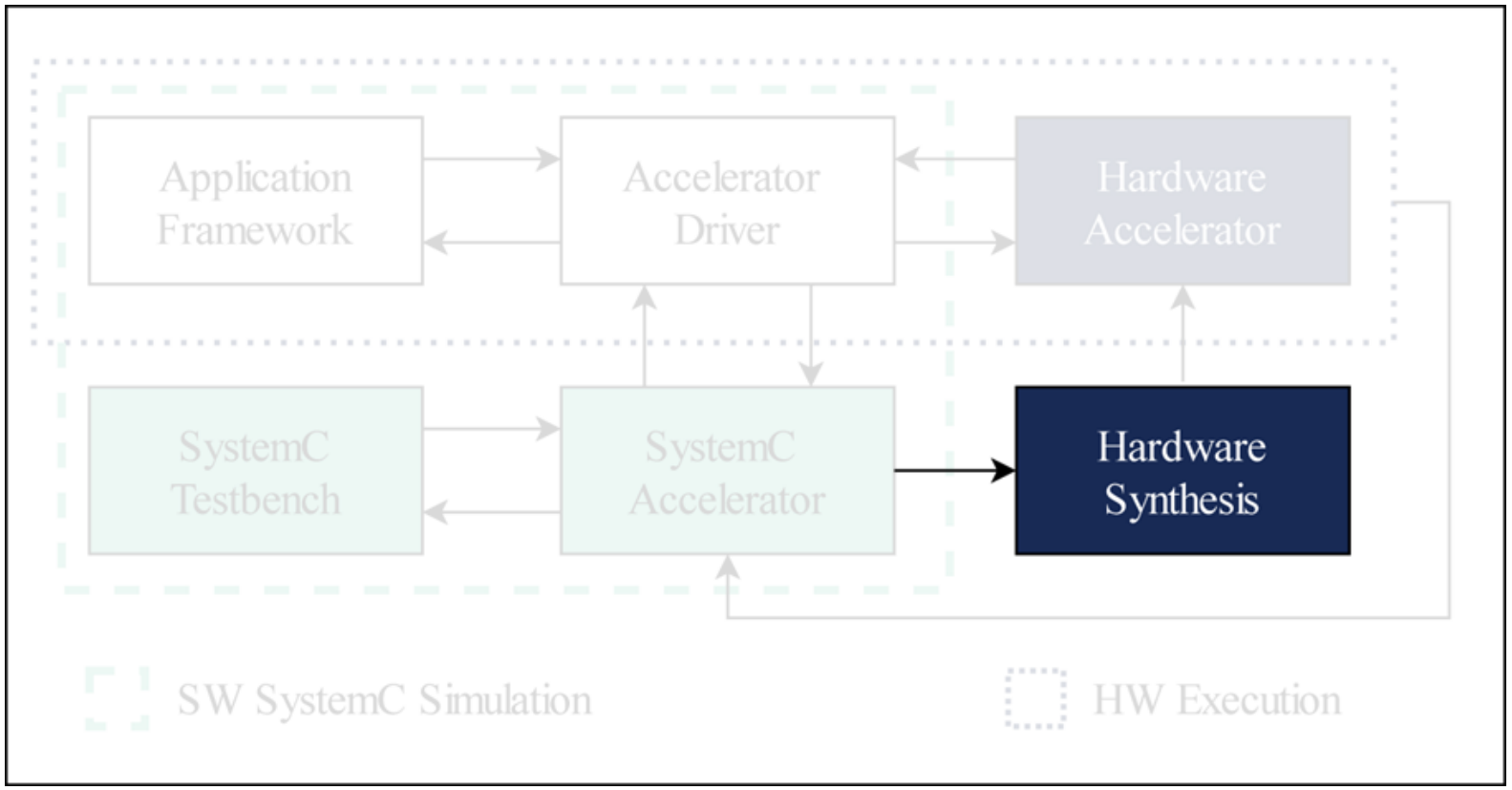
- SystemC Testbench
  - Allows unit testing (hardware accelerator)
  - Performance tuning for the entire accelerator design or specific SystemC modules
  - Simulation driven by random or sample data



SECDA Methodology: Components (5)

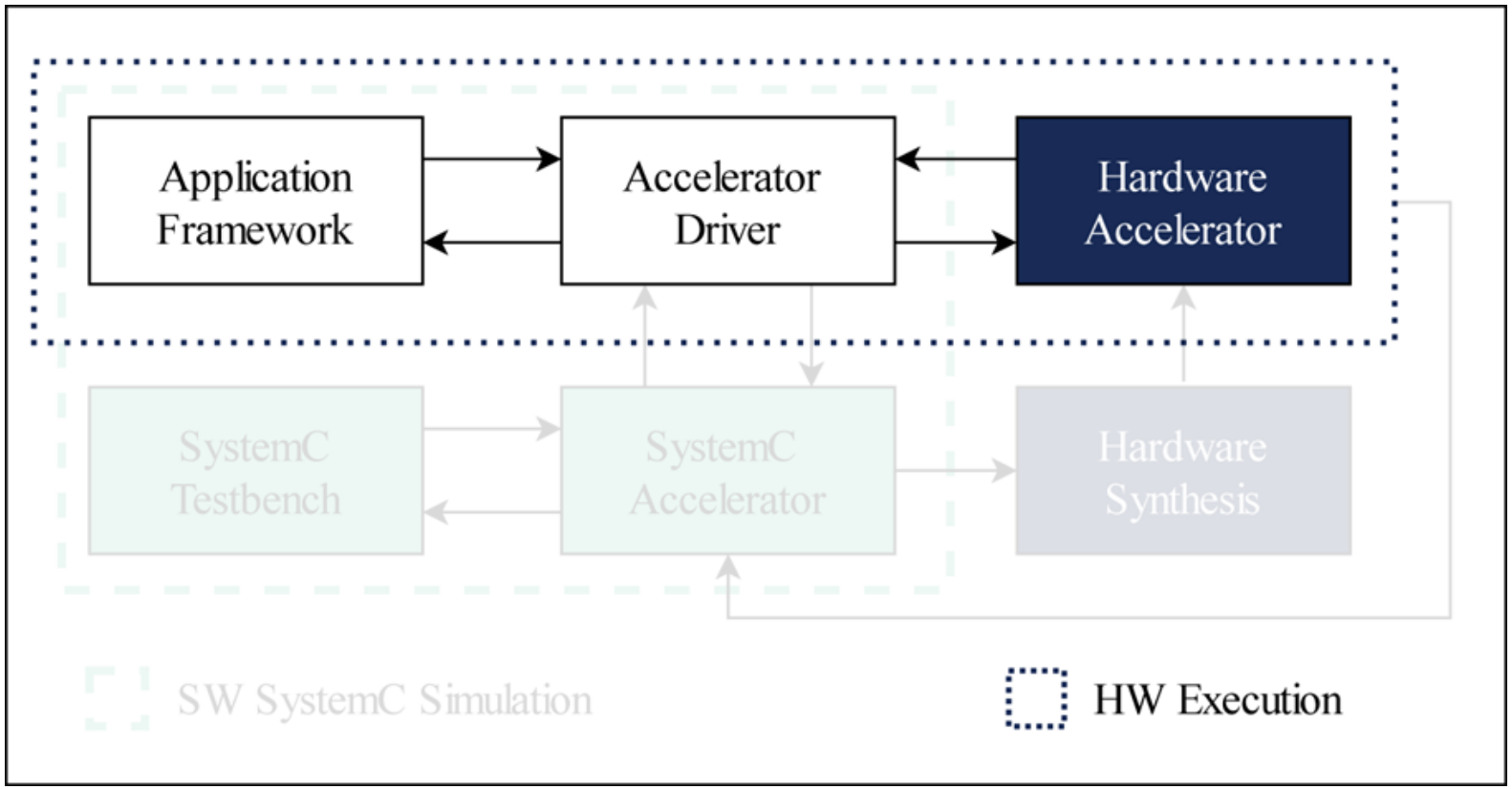
University of Glasgow

- Hardware Synthesis
  - SystemC defined accelerator
  - HLS compilation to produce RTL code (e.g. Verilog)
  - Logic synthesis to map design onto the hardware (FPGA)



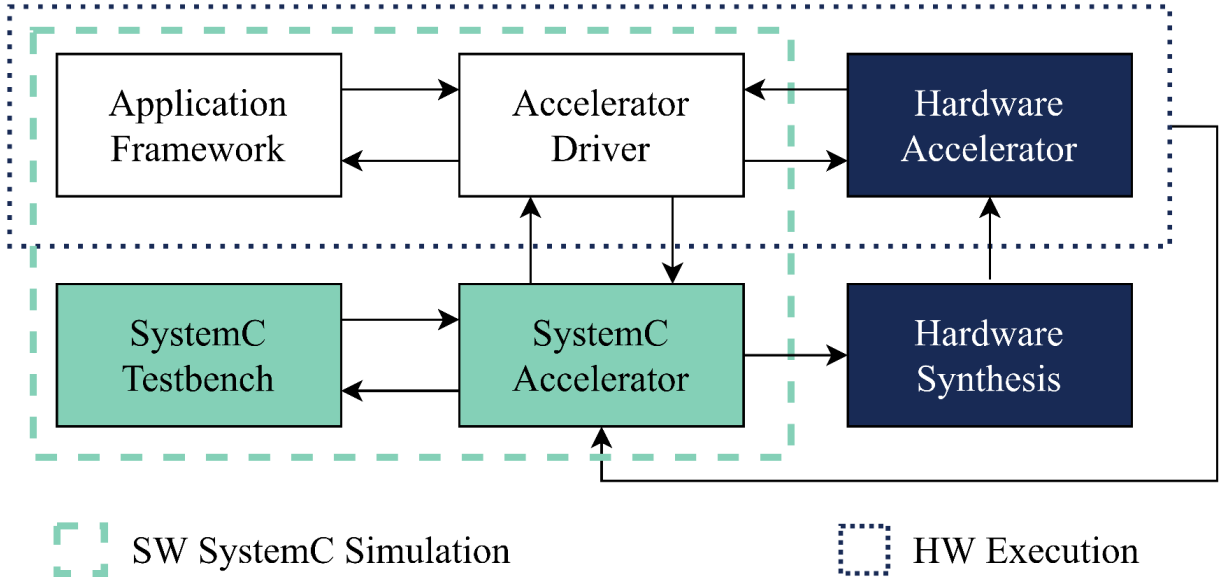
SECDA Methodology: Components (6)

- Hardware Accelerator
  - FPGA mapped accelerator
  - Full system evaluation on the target hardware



SECDA Methodology: Design Loop

- Logic synthesis is time consuming
- SECDA reduces the number of logic synthesis iterations via simulation
- Accelerator / driver (hw/sw) co-design enables easier full system integration



- **Software SystemC** Simulation
  - To profile the performance (e.g. cycles) of the individual components of the accelerator or the overall performance of data processing within the accelerator
- **Hardware** Execution
  - To obtain more accurate and additional performance data of DNN models, such as real data transfer latencies between off-chip and on-chip memory

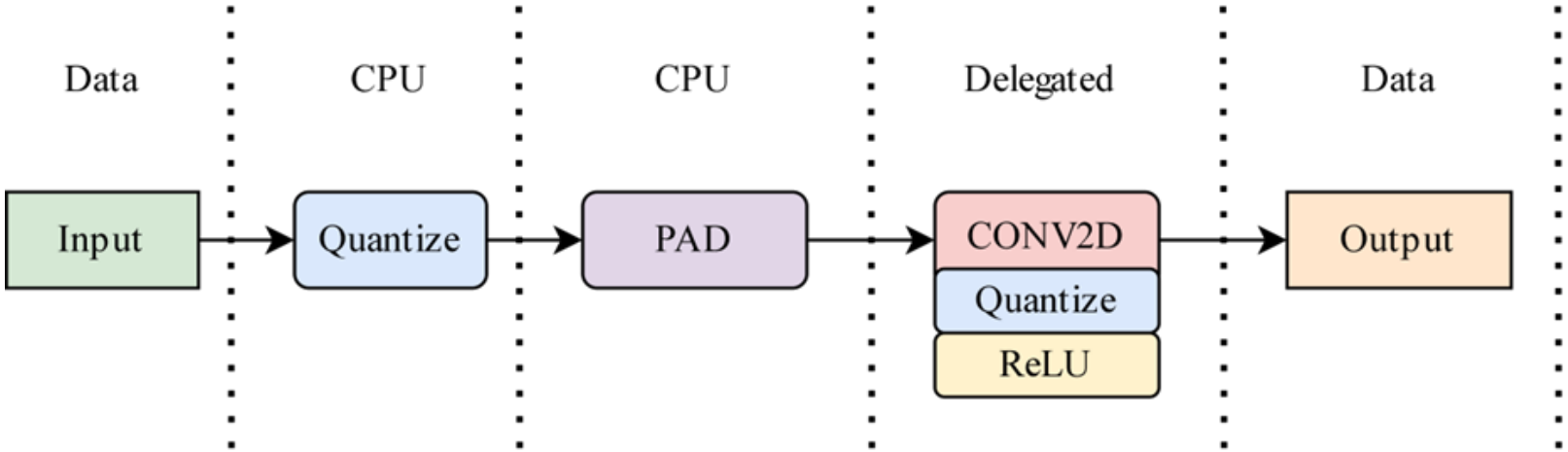
Outline

- SECDA Methodology
- SECDA-TFLite
- SECDA-LLM
- AXI4MLIR
- Conclusions and Future Work

TFLite Delegate System & API

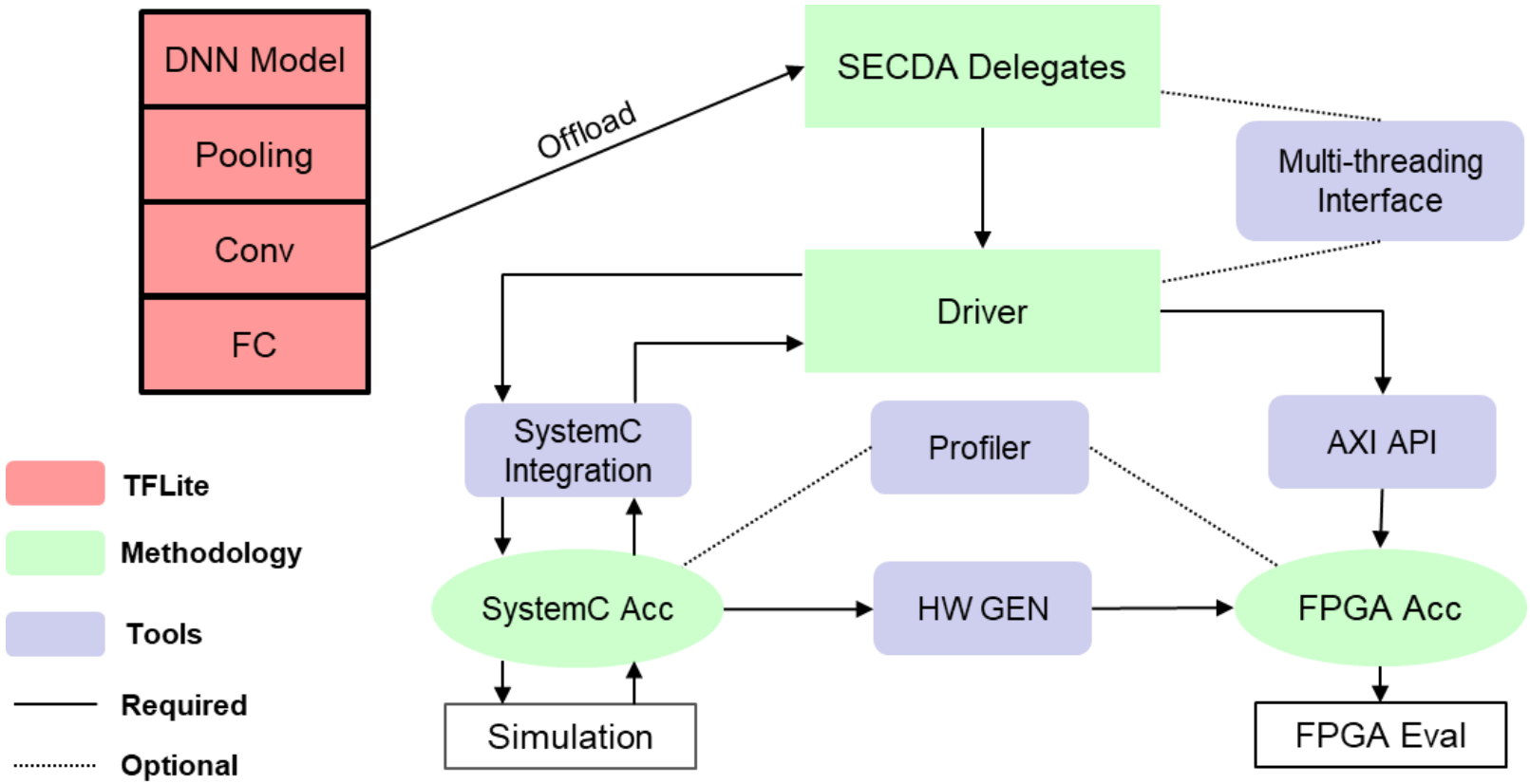
University of Glasgow

- **TensorFlow Lite (TFLite)**: framework for running DNN models on resource constrained **edge devices**
- The **Delegate system** enables to offload computation using different backends (software, hardware)
  - **Examples**: NNAPI delegate for Android, Core ML delegate for iOS, etc
- The **Delegate API** enables the development of **custom delegates**
  - VM_del, SA_del, …



SECDA-TFLite

- A toolkit for designing custom FPGA-based accelerators for TFLite
- Instantiates the SECDA methodology within TFLite
- Enables fast prototyping and integration of new accelerators with significantly **reduced initial setup costs**

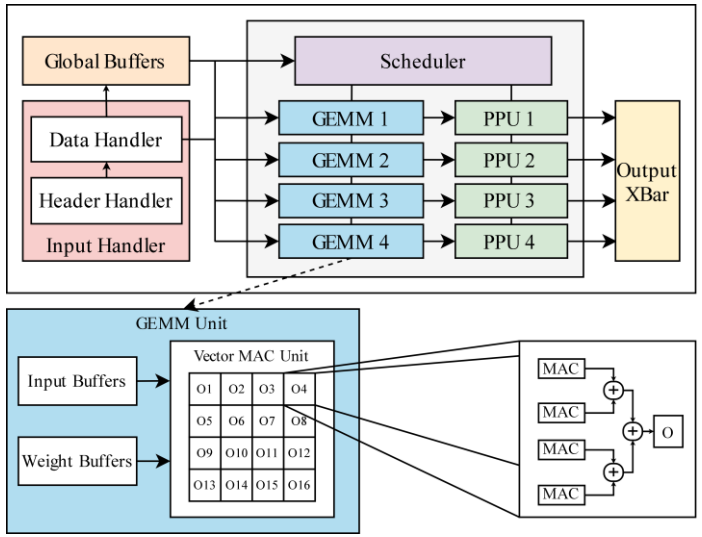


https://github.com/gicLAB/secda-tflite

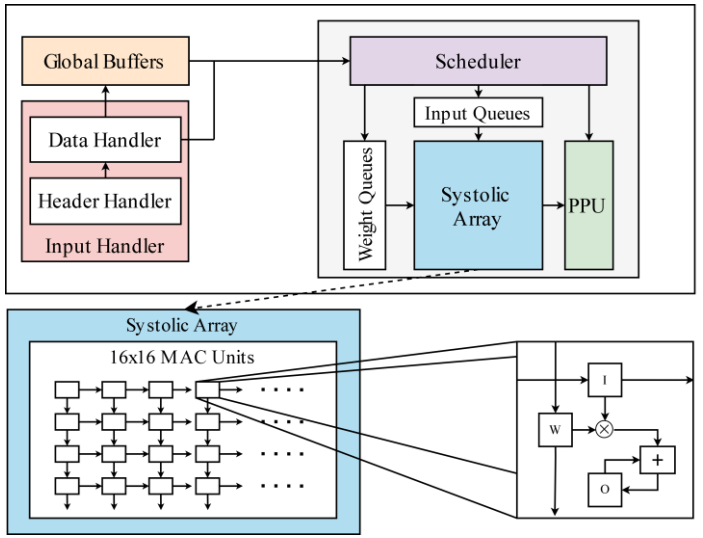
*[J. Haris, P. Gibson, J. Cano, N. B. Agostini, D. Kaeli, "*SECDA-TFLite: A Toolkit for Efficient Development of FPGA-based DNN Accelerators for Edge Inference*", **Elsevier JPDC'23**]

Case Study: DNN Accelerators

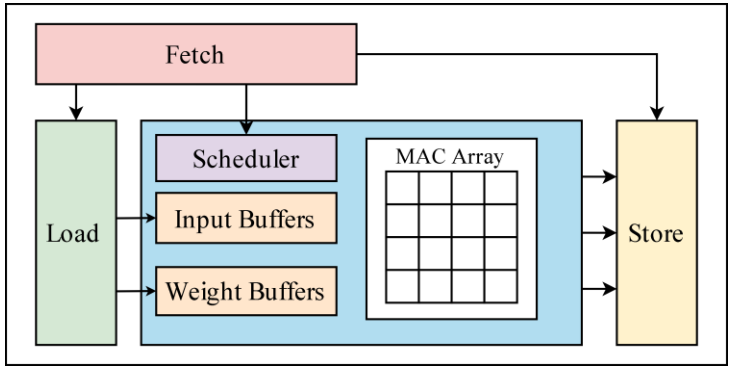
- We demonstrate the SECDA-TFLite toolkit with a case study; we develop 3 GEMM-based Accelerators
- Vector MAC and Systolic Array accelerators developed to accelerate CONV layers in CNN models
- FC-GEMM accelerator developed to accelerate Fully Connected Layers in Transformer models



Vector Mac Design



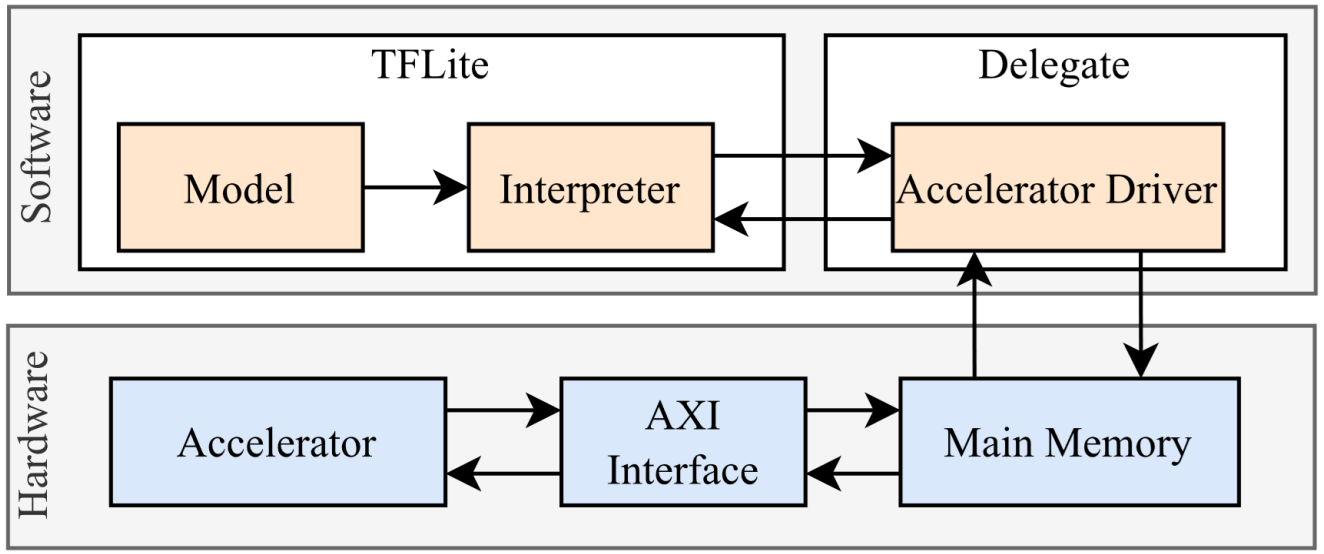
Systolic Array Design



FC-GEMM design

Runtime Model (HW execution)

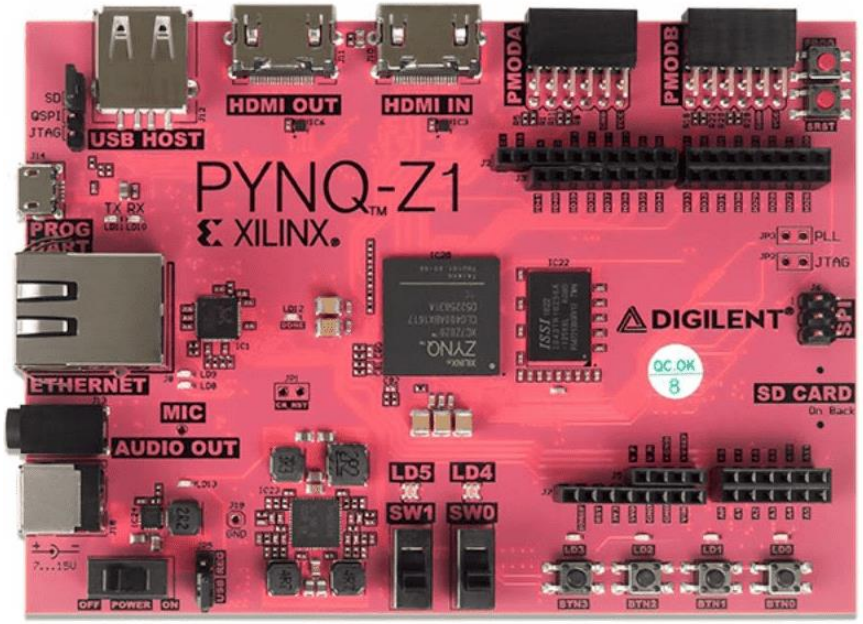
- It shows how we integrate the accelerators within TFLite via Accelerator Delegate and Driver



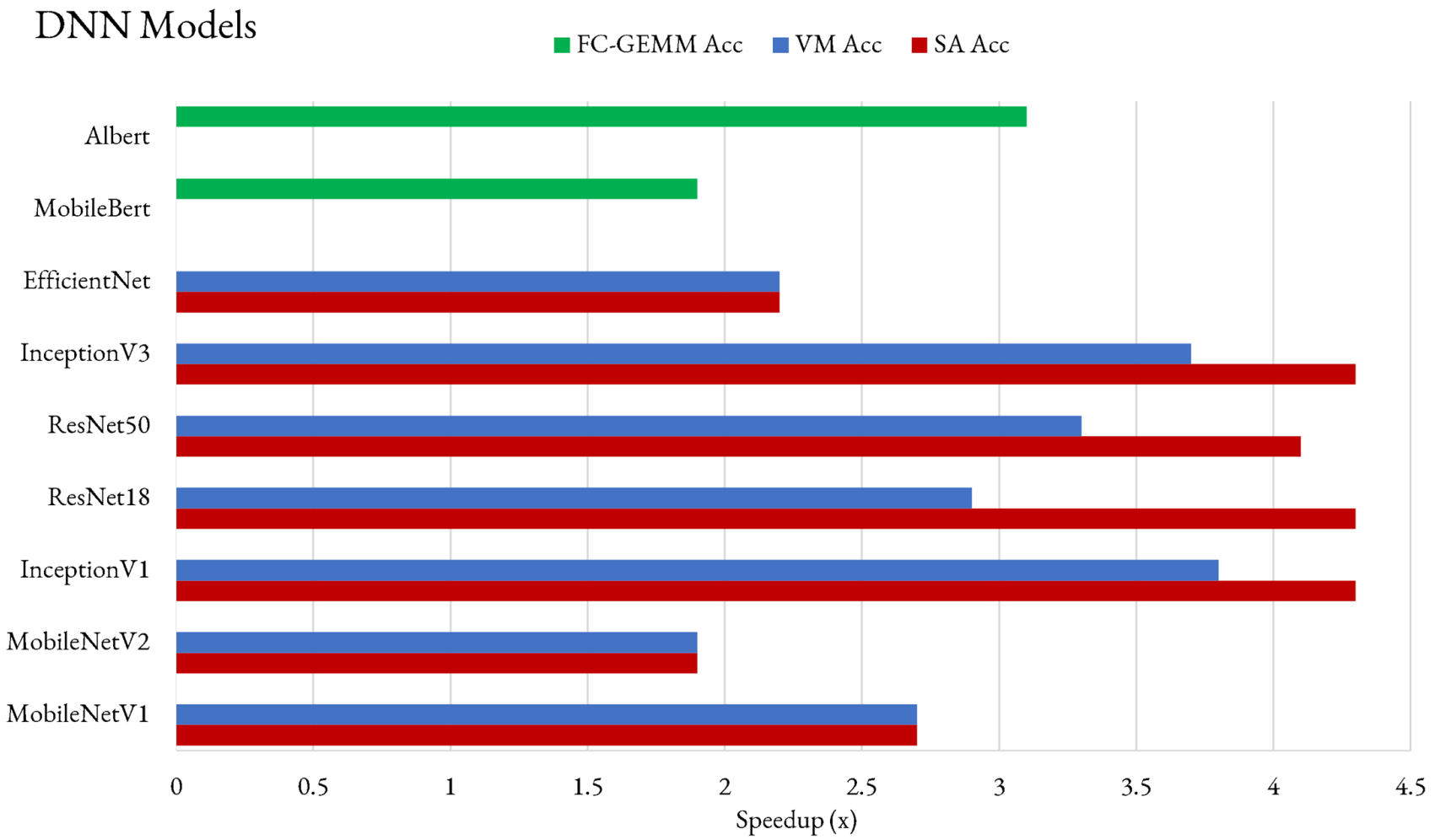
Evaluation: Experimental Setup

University of Glasgow

- **PYNQ Z1 board** (Arm A9 dual-core CPU + Edge FPGA)
- 9 DNN models evaluated (7 CNNs, 2 BERT)
  - ImageNet and SQuAD datasets
- TFLite **8-bit quantized** models (CNN and BERT)
- **Inference** CPU + accelerator vs. CPU only (1/2 threads)
  - Execution **time**
  - **Power** measured (using USB power meter)
- We compared one of the models with VTA accelerator



Evaluation: Results



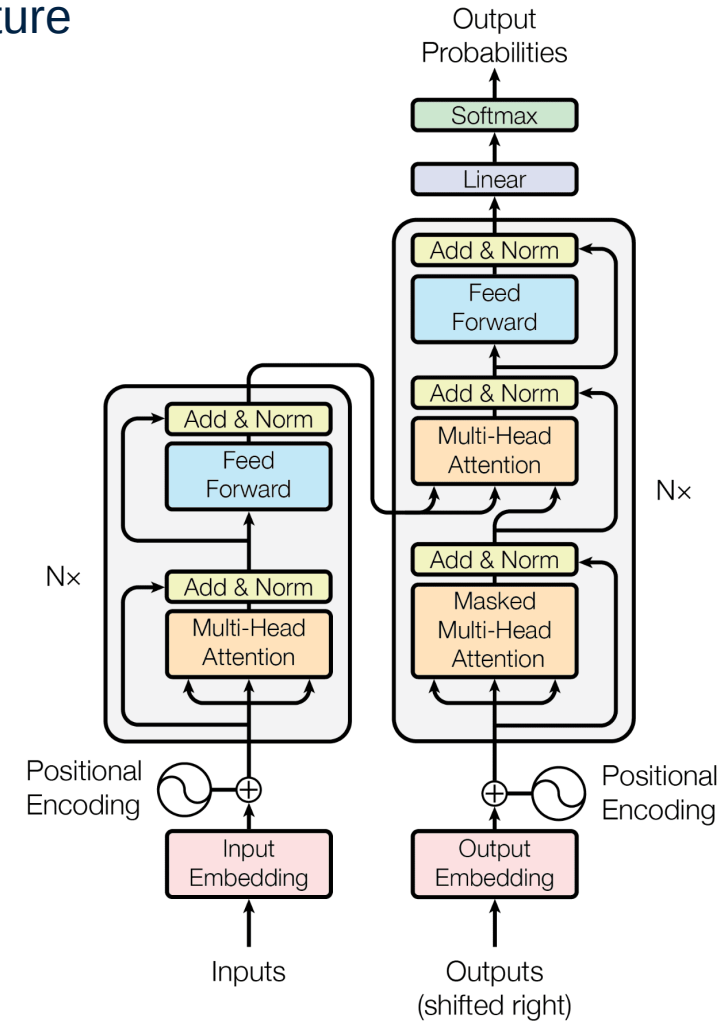
- Average speedup for **inference time** of up to **3.4x** and **2.5x** for CNN and BERT models respectively
- Average **energy** savings of up to **2.9x** and **2.4x** for CNN and BERT models respectively

Outline

- SECDA Methodology
- SECDA-TFLite
- SECDA-LLM
- AXI4MLIR
- Conclusions and Future Work

Large Language Models (LLMs)

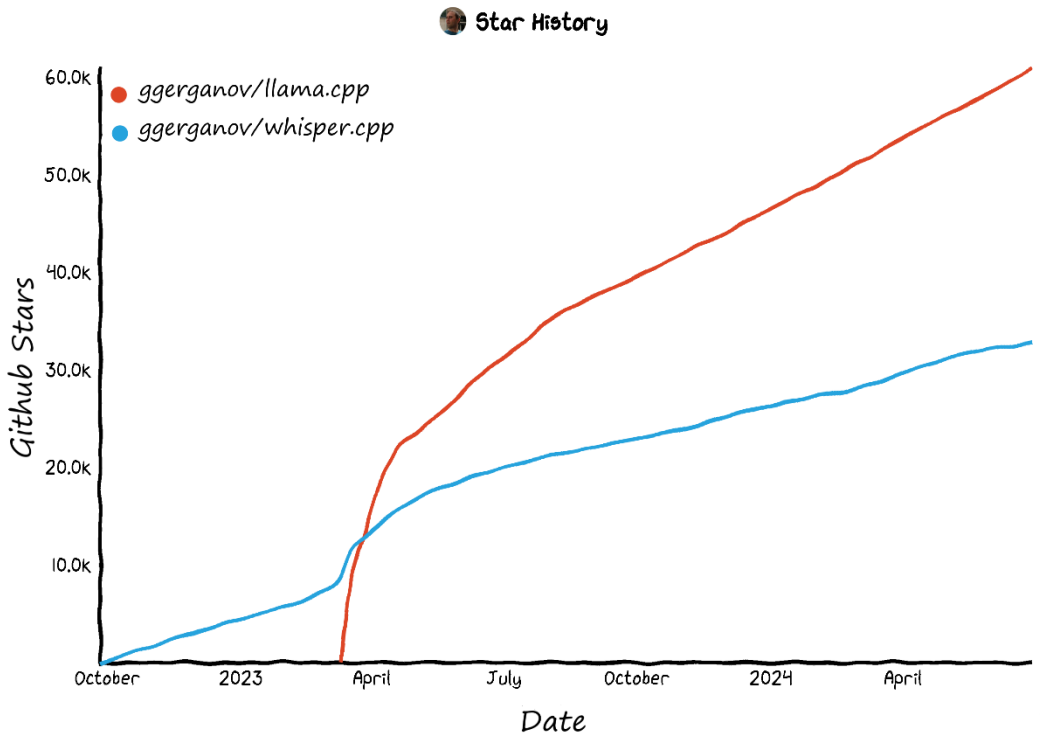
- LLMs are a family of models that use the Transformer-based architecture
- Great at solving many language related tasks
  - Text Generation, AI assistants, Code generation, etc
- Greatly increase upon the number of parameters used
  - PaLM 2 apparently has 340 billion parameters!
- Many optimization techniques to improve execution performance
  - KV (key-value) caching
  - Quantization
  - …



llama.cpp

University of Glasgow

- A pure C/C++ library with minimal external dependencies
- Enables LLM inference with minimal setup on a wide range of hardware devices
- Supports multi-modal, custom, and well-known LLMs (e.g., Llama, Falcon, GPT, Gemma)
- Utilizes GGUF (GPT-Generated Unified Format) and supports various type of quantization (1.5-bit, 2-bit, 3-bit, 4-bit, 5-bit, 6-bit, and 8-bit)
- Open source, with active and rapidly growing community

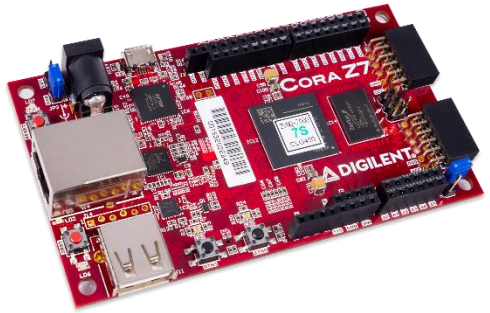


https://github.com/ggerganov/llama.cpp

LLMs (and SLMs) on Edge Devices

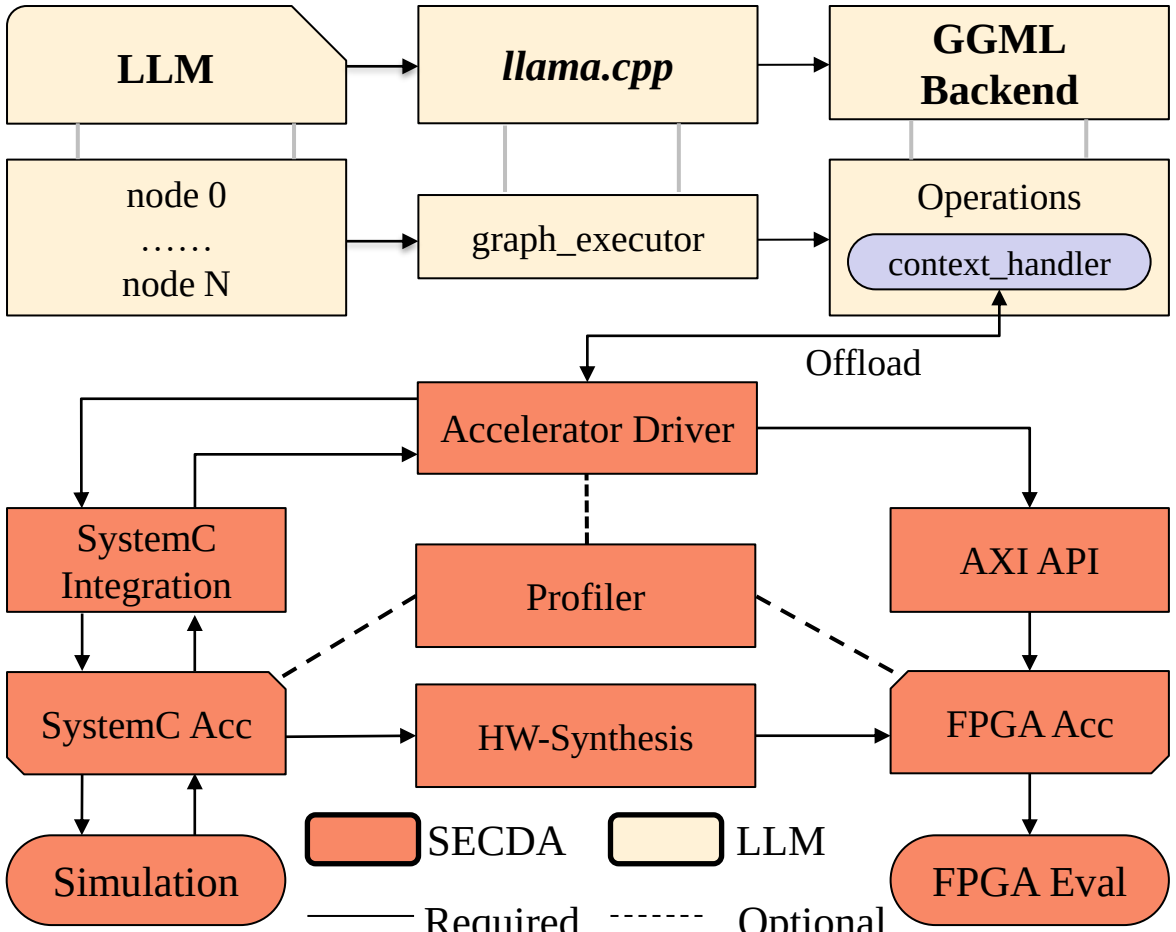
University of Glasgow

- Running **LLMs on the edge** has become popular with concerns on **network availability**, **security** and **privacy**
- Executing **LLMs on edge devices** is difficult due to **computation** and **memory demands**
- The problem is **further exacerbated on resource-constrained** edge devices
- Hence, we need to develop **specialized hardware accelerators** to efficiently process LLMs with limited resources



SECDA-LLM

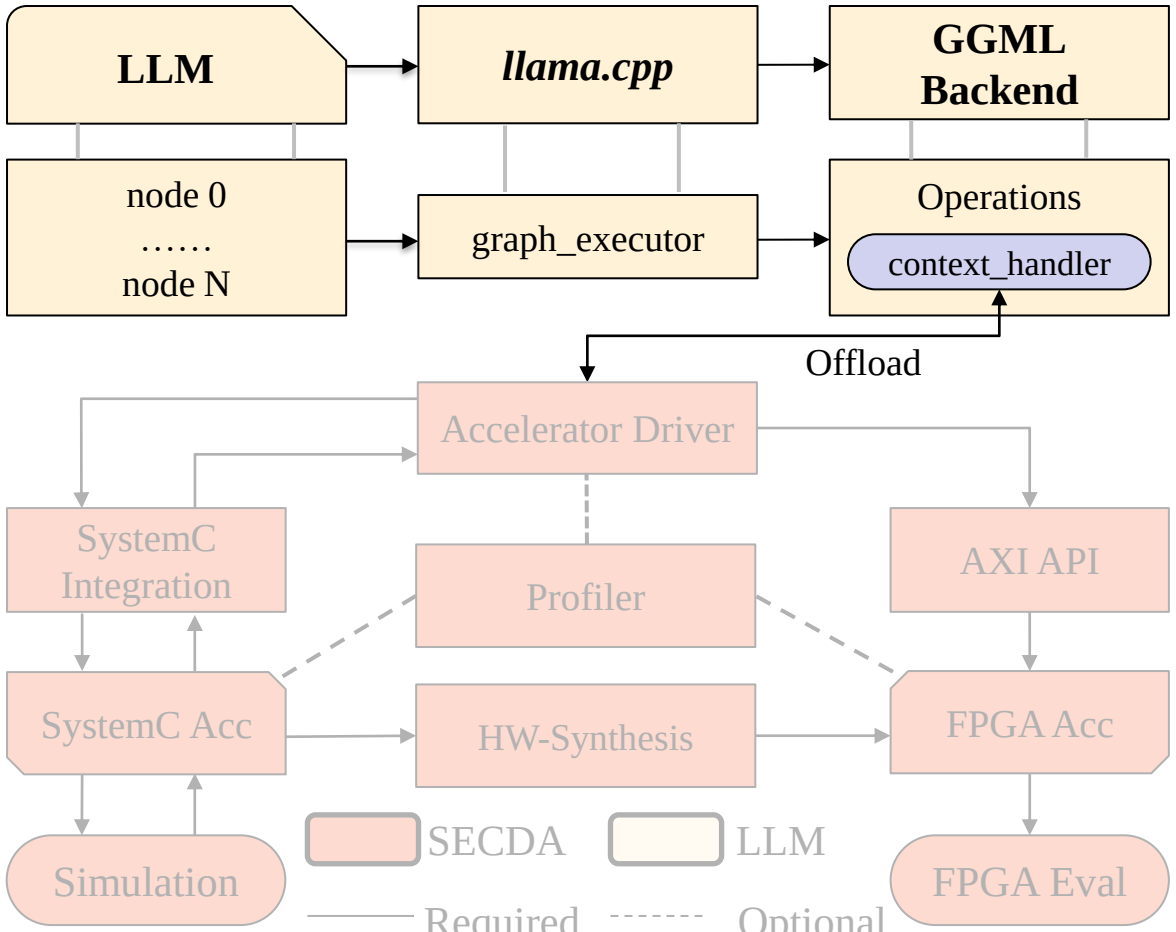
- A toolkit for designing custom FPGA-based accelerators for LLMs
- Instantiates the SECDA methodology within ***llama.cpp***
- Enables fast prototyping and integration of new accelerators with significantly **reduced initial setup costs**



*[J. Haris, R. Saha, W. Hu, J. Cano, "*Designing Efficient LLM Accelerators for Edge Devices*", **ARC-LG @ ISCA'24**]

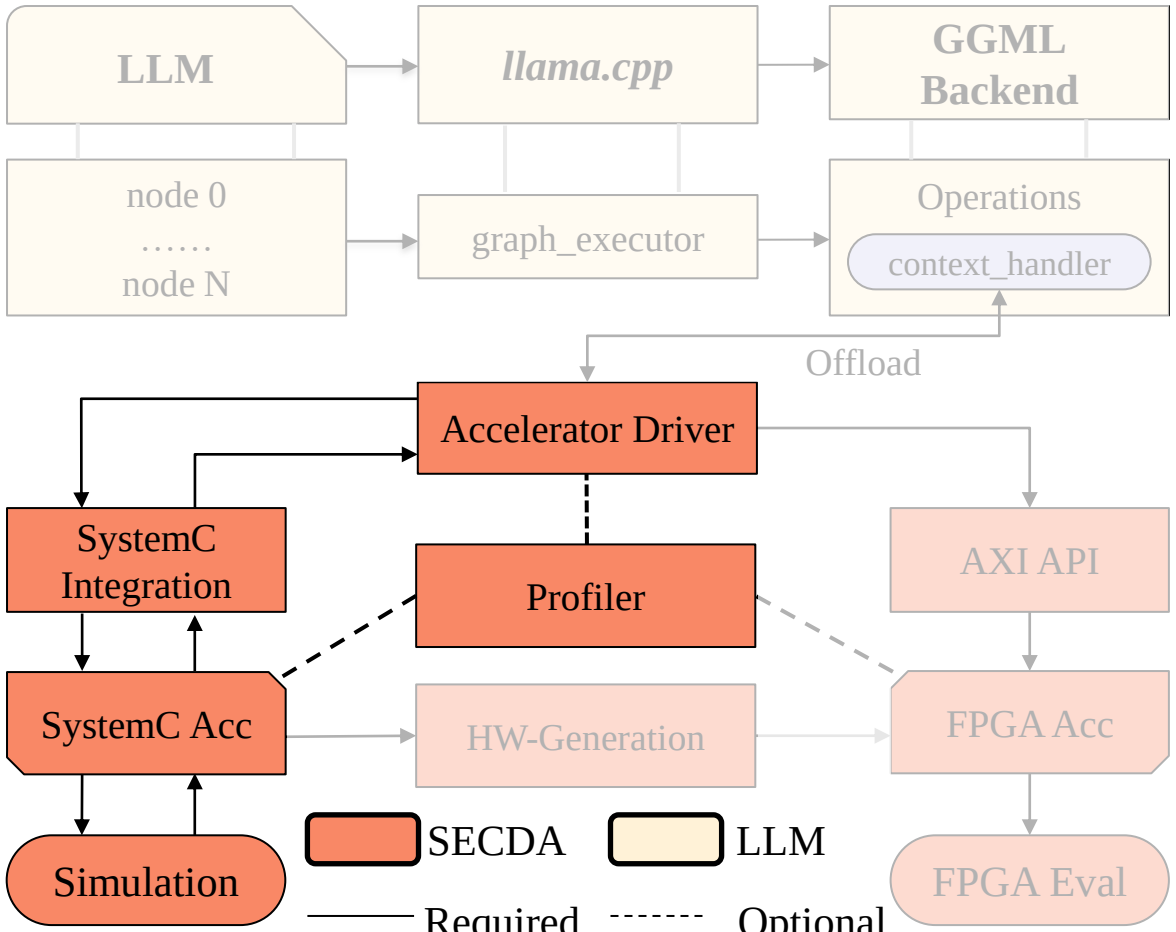
Connecting *llama.cpp*

- *SECDA-LLM* uses *llama.cpp* as the **"Application Framework"**
- Enables acceleration of **LLMs** based on **GGUF** (GPT-Generated Unified Format)
- Target operations (matmul, softmax) are offloaded from the **GGML** (GPT-Generated Model Language) **backend** to our custom accelerator
- A **context_handler** is created to pass operation parameters and metadata to the **Accelerator Driver**



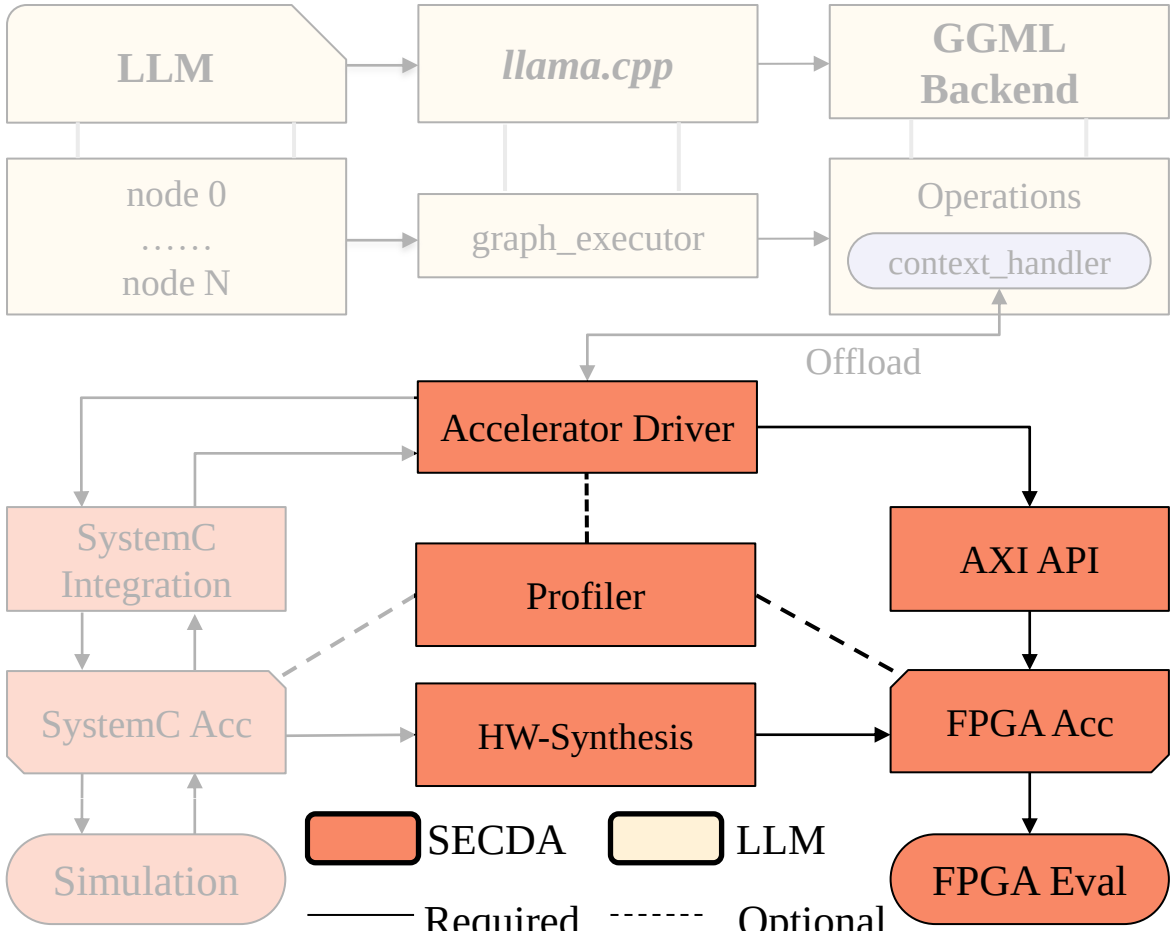
Simulation Design Loop

- The Accelerator Driver initiates the **simulation-based design loop** enabling rapid accelerator prototyping
- The **Accelerator design** specified in **SystemC** allows quick development without the need of traditional HDLs such as Verilog or VHDL
- End-to-end simulation **verifies correctness** across real LLMs
- **Simulation profiling** tracks metrics, e.g., cycle counts, PE utilization, on-chip memory utilization



Hardware Generation and Evaluation

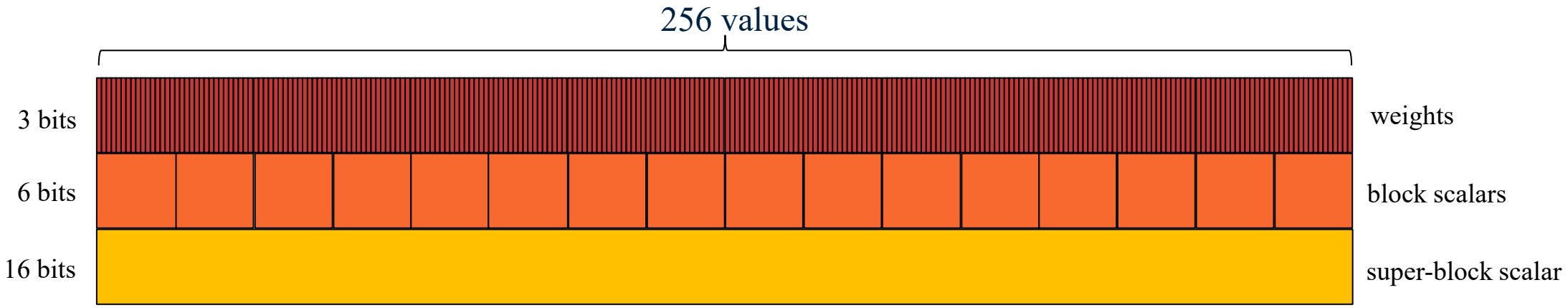
- The developer can quickly evaluate accelerator designs through **SystemC HLS** and **FPGA synthesis**
- The **Hardware-Synthesis** tool
  - JSON-based configuration file
  - Automated HLS and bitstream generation
- **AXI-API** connects the FPGA accelerator with the driver
  - No driver code change required
- **Hardware profiling** tracks real time performance



Case study: MatMul Acceleration

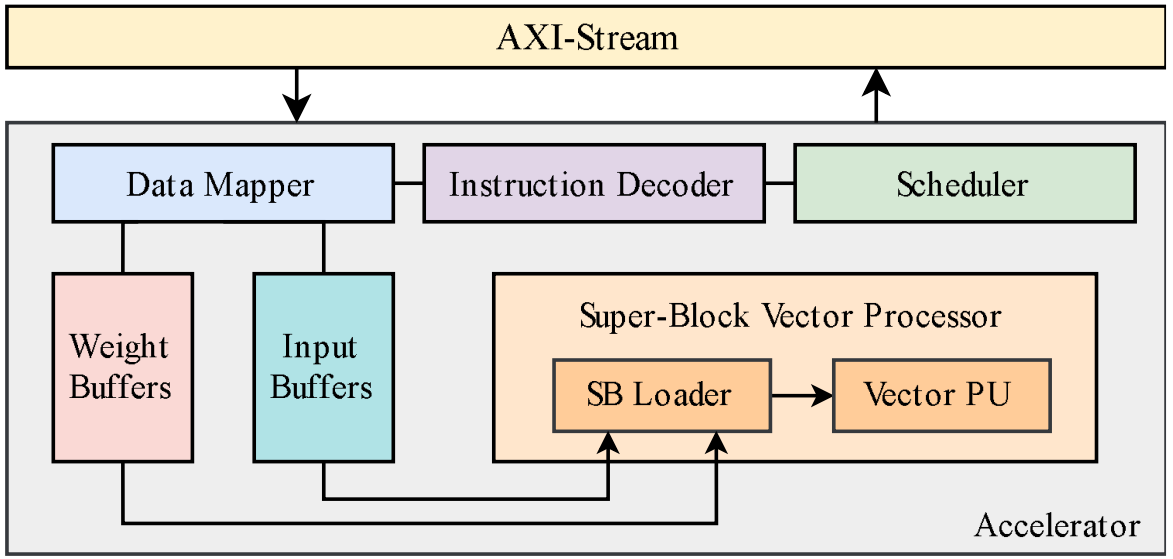
University of Glasgow

- Using SECDA-LLM we developed a specialized FPGA-based accelerator for LLM inference
- We accelerate the **MatMul kernel**, the most expensive operation within LLMs (~97% for TinyLlama)
- We use **block floating point (BFP)** quantization (common in *llama.cpp*) with **Q3_K_Q8_K** configuration
  - Weights use Q3_K super-blocks, i.e. **~3.5 bit quantization**
  - Inputs use Q8_K super-blocks, i.e. **~9.1 bit quantization**



Q3_K super-block Data Format

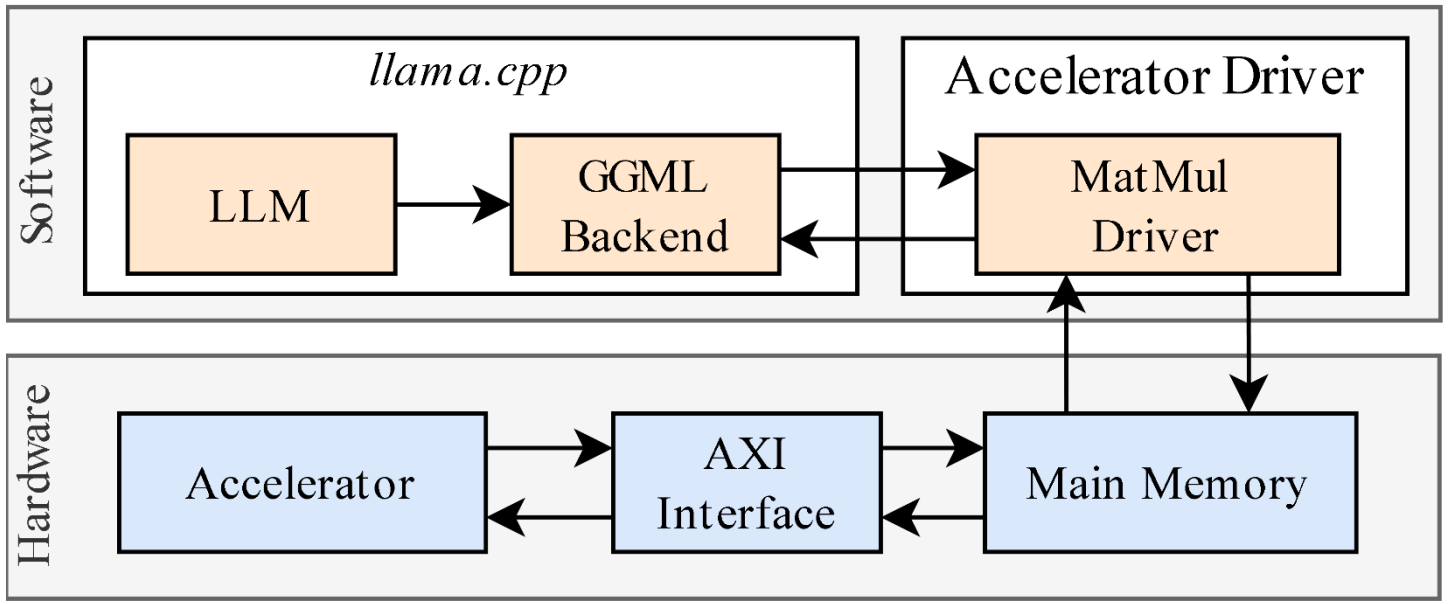
Case study: Accelerator Design



- Simple **opcodes to configure and control** the accelerator
- The scheduler enables **MatMul tiling** to increase data reuse
- **Super-Block Vector Processor**
  - Exploits parallelism across super-blocks
  - **Q3_K_Q8_K** format specific optimizations

Runtime Model (HW execution)

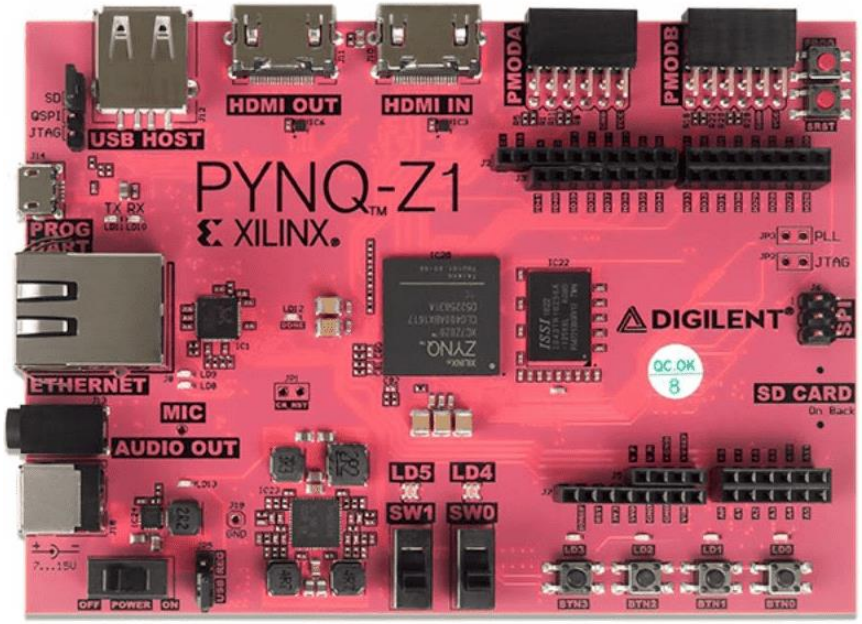
- It shows how we integrate the accelerators within *llama.cpp* via Accelerator Driver



Evaluation: Experimental Setup

University of Glasgow

- **PYNQ Z1** board
  - Arm A9 dual-core CPU @ 650 MHz
  - Xilinx Z020 edge FPGA
  - 512 MB DDR3 memory

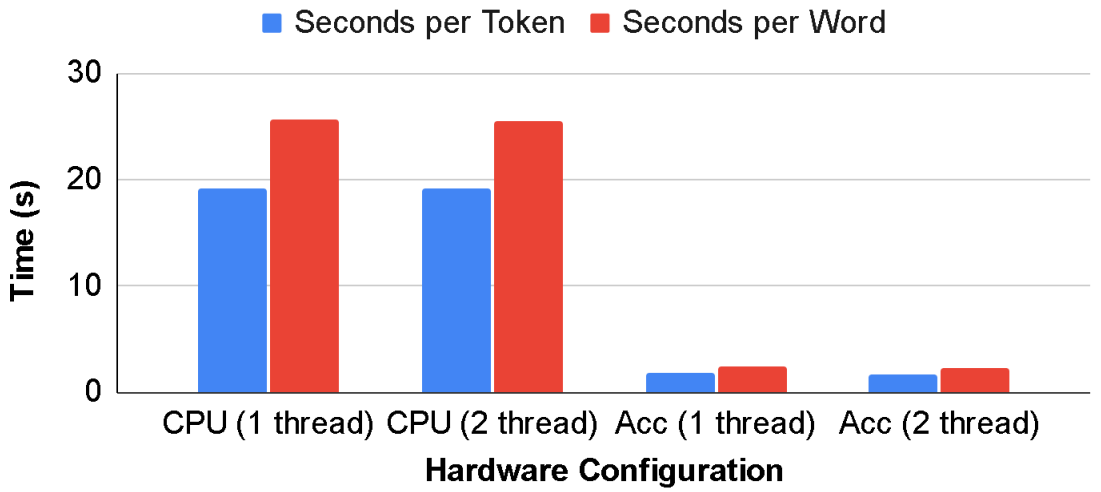


- **TinyLlama** model, 1.1B parameters (460MB~)
  - With **Q3_K_Q8_K BFP** quantization
  - Guanaco dataset

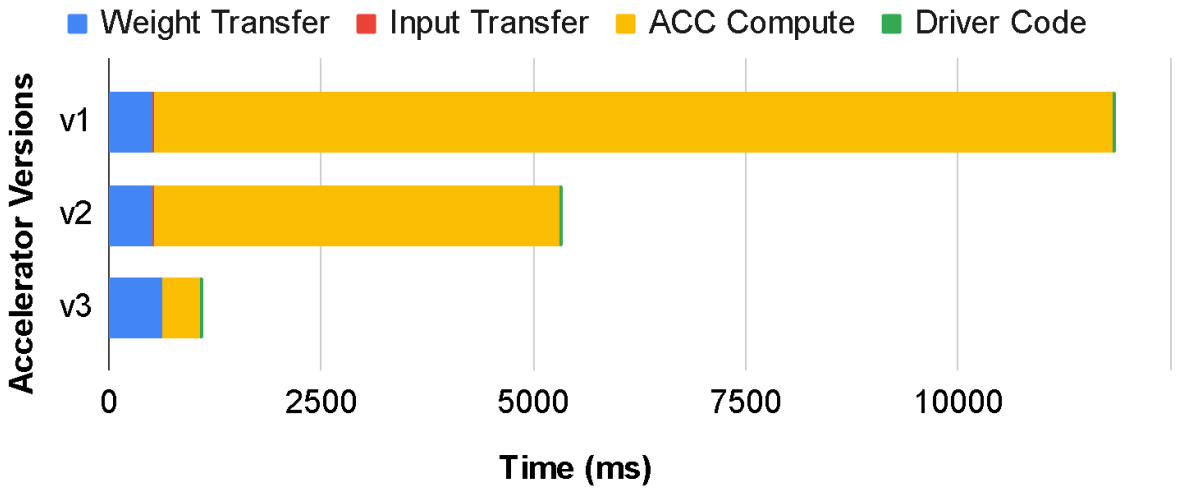
- We evaluate **inference** latency across different hardware configurations
  - CPU only (2 threads)
  - CPU + accelerator

Evaluation: Results

- CPU + Acc achieves **11x speedup** in terms of token generation
  - Around **1.7s per token** (~2s per word)
  - Compared to only CPU **19.2s** (~26s per word)



- We also tracked more **in-depth profiling** of accelerator + driver performance **across different design iterations**
  - **v1:** simplest design
  - **v2:** exploits super-block parallelism
  - **v3:** introduced scheduler to enable data-reuse



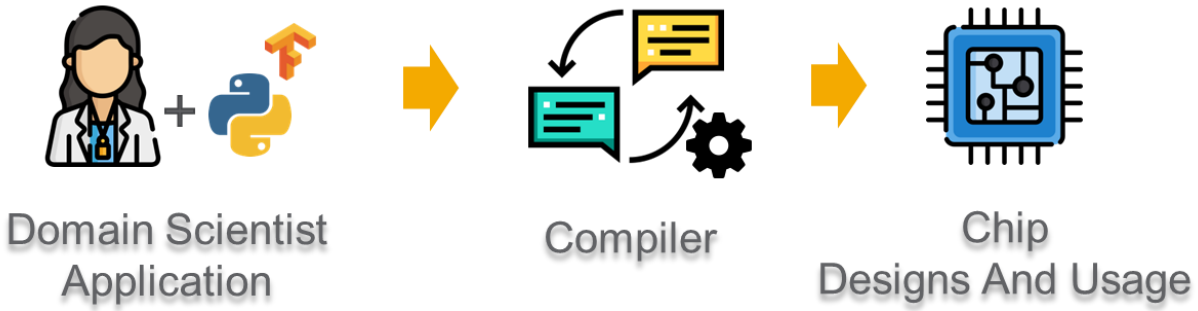
Outline

- SECDA Methodology
- SECDA-TFLite
- SECDA-LLM
- AXI4MLIR
- Conclusions and Future Work

AXI4MLIR

University of Glasgow

- **Motivation**: Efficient host-driver code is required to maximise accelerator performance
- **Problem**: Creating and optimizing this host-driver code is difficult and time consuming
- **Solution**: *"AXI4MLIR: User-Driven Automatic Host Code Generation for Custom AXI-Based Accelerators"*
  - Efficient host-code for custom AXI-Stream-based accelerators using the **MLIR** compiler framework
  - We leverage the **SECDA** methodology to rapidly prototype new custom accelerators
- **Vision**: automatic generation and usage of optimized domain specific accelerators



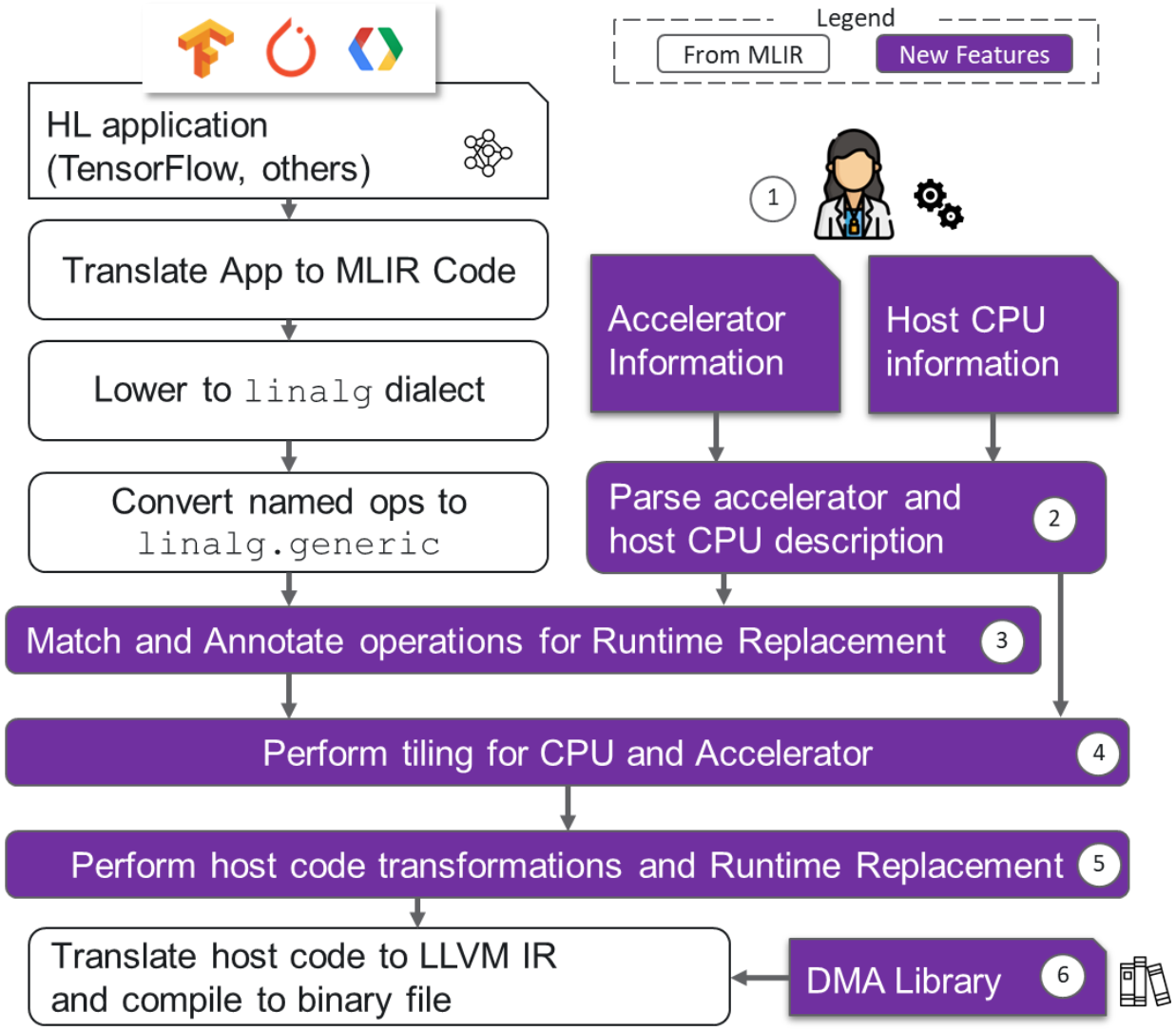
*[N. Bohm Agostini, J. Haris, P. Gibson, M. Jayaweera, N. Rubin, A. Tumeo, J. L. Abellán, J. Cano, D. Kaeli, "*AXI4MLIR: User-Driven Automatic Host Code Generation for Custom AXI-Based Accelerators*", **CGO'24**]

AXI4MLIR: Approach

- **MLIR** is a unifying software framework for compiler development (sub-project of **LLVM** compiler infrastructure)

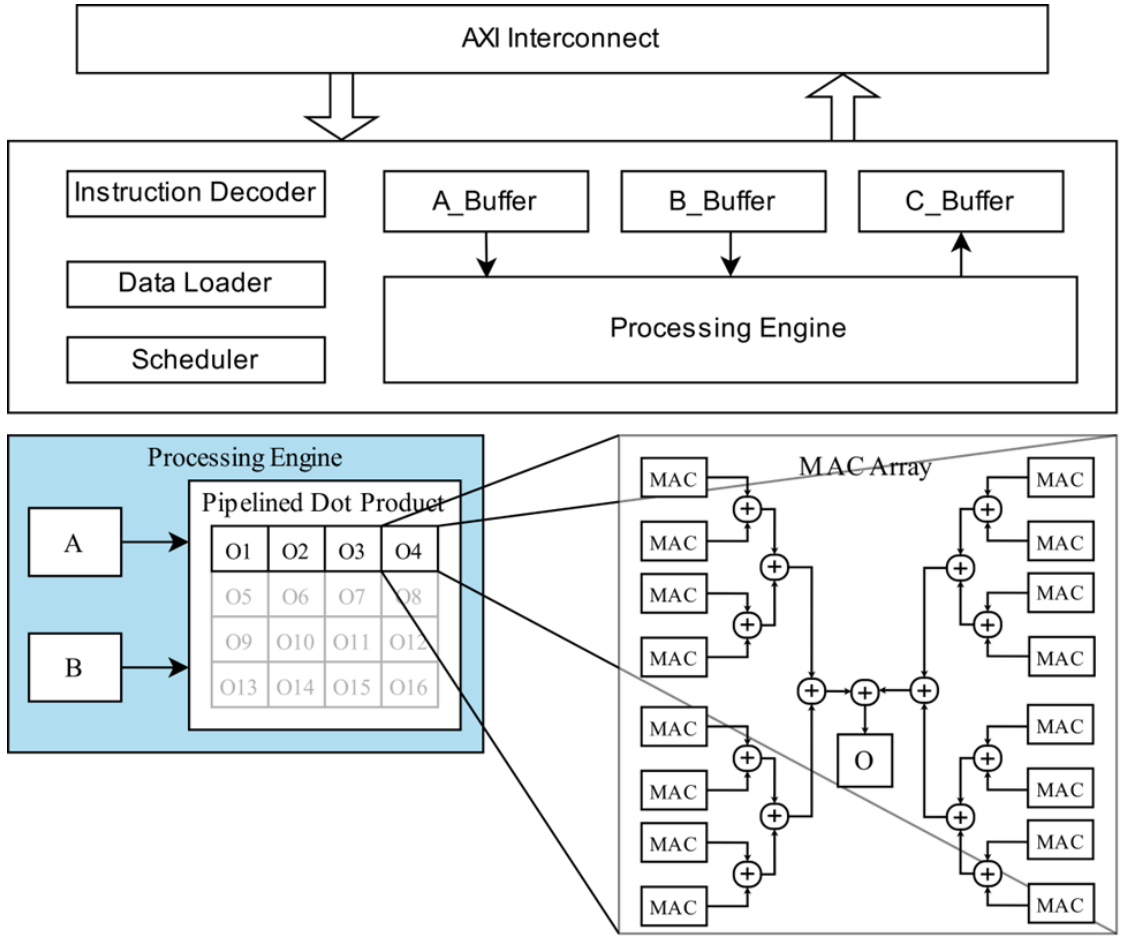
- **AXI4MLIR**
  - **MLIR extensions** to describe custom accelerators with arbitrary instructions
  - Simple Host-Accelerator **communication abstraction** and **AXI library** implementation
  - Implements **host code generation** to drive accelerators connected through an AXI-Stream Interface

https://github.com/AXI4MLIR/axi4mlir



Case study: Accelerator System Level Design

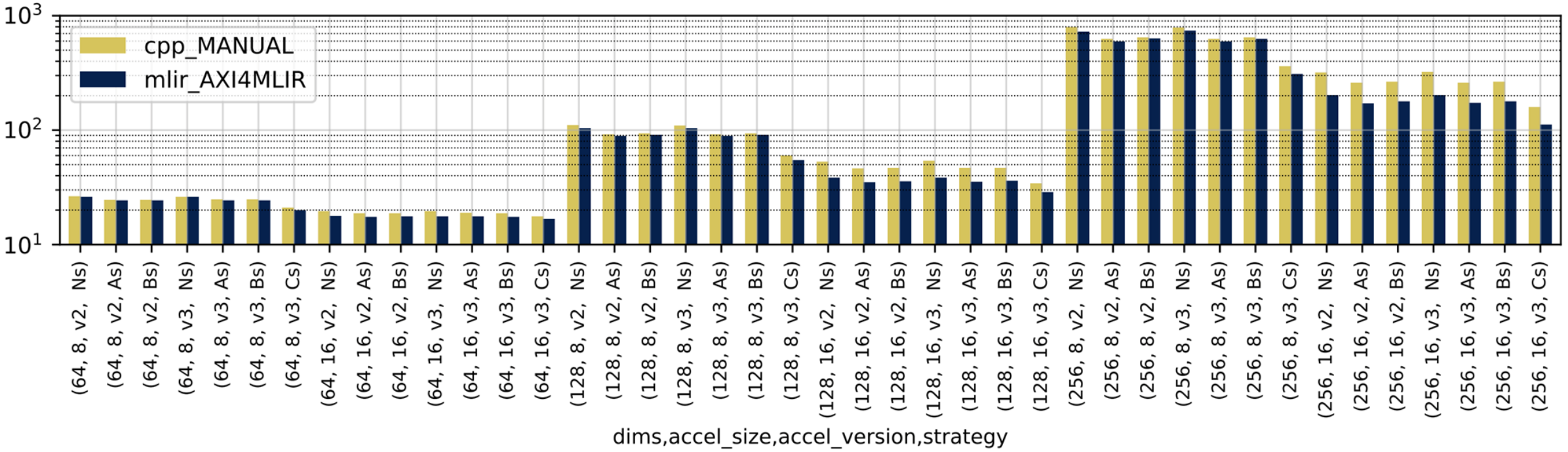
- We developed simple, scalable **MatMul** accelerators (different tile sizes, opcodes and dataflow)
- Accelerators support different **dataflows** via instructions (**Instruction Decoder**)
- Input data streamed **via AXI interconnect** stored into global **A/B buffers** (via the **Data Loader**)
- **Processing Engine** contains local A/B cache to compute the required dot product within a MAC array



MMv4$_{16}$

Manual vs. AXI4MLIR Generate Host Code

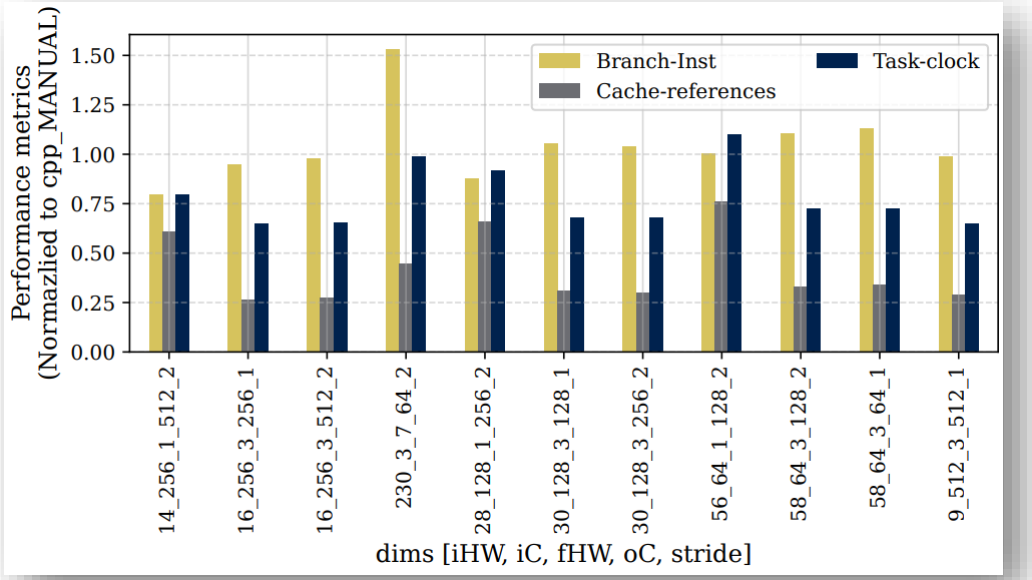
Type	Possible Reuse	Opcode(s)	Configurations (Size, OPs/Cycle)
$v1_{size}$	Nothing	sAsBcCrC	(8, 60)
$v2_{size}$	Inputs	sA, sB, cCrC	(16, 112)
$v3_{size}$	Ins/Out	sA, sB, cC, rC	



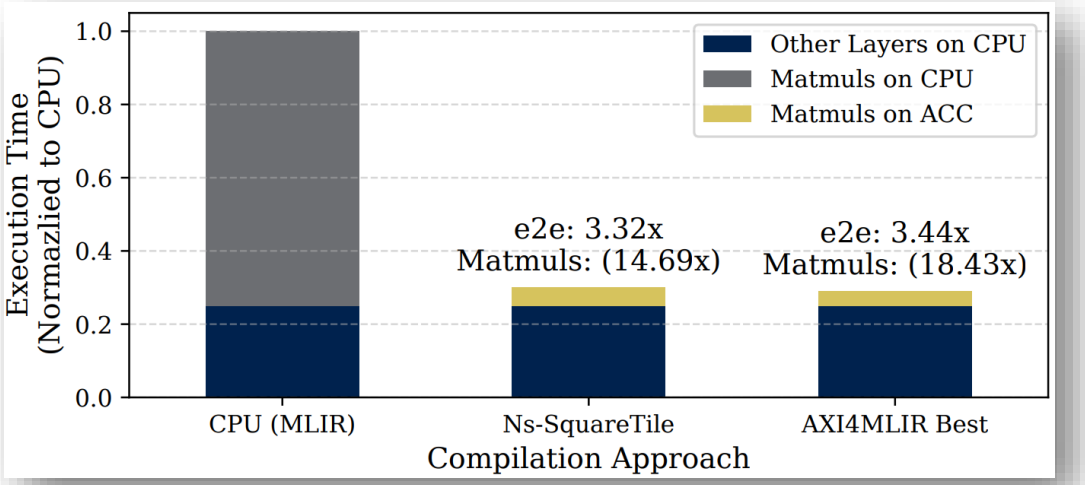
Up to **1.65x** speedup
and **56%** cache references

Additional Results (check paper)

- **ResNet18** with Conv Layer accelerator
  - Performance Counters
    - Cache References
    - Branch instructions
    - Task Clock
  - Varied tile sizes



- **TinyBert** with Matrix Multiplication accelerator
  - Oracle (DSE) selection of best parameters



Outline

- SECDA Methodology
- SECDA-TFLite
- SECDA-LLM
- AXI4MLIR
- Conclusions and Future Work

Conclusions

University of Glasgow

- **SECDA** is a design methodology to efficiently reduce the development time of FPGA-based accelerators

- **SECDA-TFLite** is a new open source toolkit that improves/eases the development of new FPGA-based accelerators for edge DNN inference employing **TFLite** and the SECDA methodology

- **SECDA-LLM** is a new toolkit that improves/eases the development of new FPGA-based accelerators for edge LLM inference employing ***llama.cpp*** and the SECDA methodology

- **AXI4MLIR** extends the **MLIR** compiler framework with new attributes that can be used to describe custom linear algebra accelerators with arbitrary instructions

Future Work

University of Glasgow

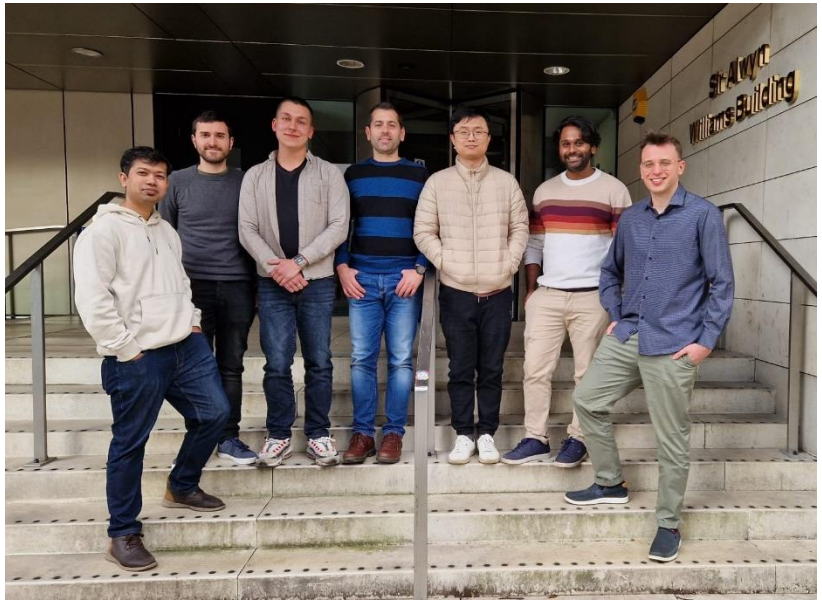
- On-going
  - Support more types of layers (e.g. transpose conv), cores (e.g. shift-based), sparsity
  - Create similar development toolkits for other DNN frameworks such as PyTorch, TVM, etc
  - Expand **SECDA-LLM** as an open-source platform

- Planned
  - **SECDA-MLIR**: Create a SystemC MLIR dialect to generate SystemC modules
  - **SECDA-PIM**: Designing and Simulating PIM-Based Accelerators
  - **SECDA-Tools**: Open Source Bazel package for Rapid Accelerator Prototyping

- Potential
  - **SECDA-Gene**: Heterogenous Acceleration for Genome Analysis
  - **SECDA-Train**: Developing Custom Accelerator for DNN Training

Acknowledgements

University of Glasgow



1) Researchers and **students** at gicLAB

2) Funding bodies

UKRI Engineering and Physical Sciences Research Council

Horizon Europe
THE NEXT EU RESEARCH & INNOVATION PROGRAMME (2021 – 2027)

sicsa*
The Scottish Informatics & Computer Science Alliance

3) Collaborators from Academia

THE UNIVERSITY of EDINBURGH

Northeastern University

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

ETH zürich

UNIVERSITY OF OXFORD

University of Essex

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSIDAD DE MURCIA

UNIVERSITAT JAUME I

4) Collaborators from Industry and Labs

AMD

NVIDIA

arm

Pacific Northwest NATIONAL LABORATORY

University of Glasgow

gicLAB

Accelerating AI at the Edge

The Power of Efficient HW/SW Co-Design

José Cano

Jose.CanoReyes@glasgow.ac.uk

School of Computing Science, University of Glasgow, UK

Thank you! Questions?