University of Glasgow

gicLAB

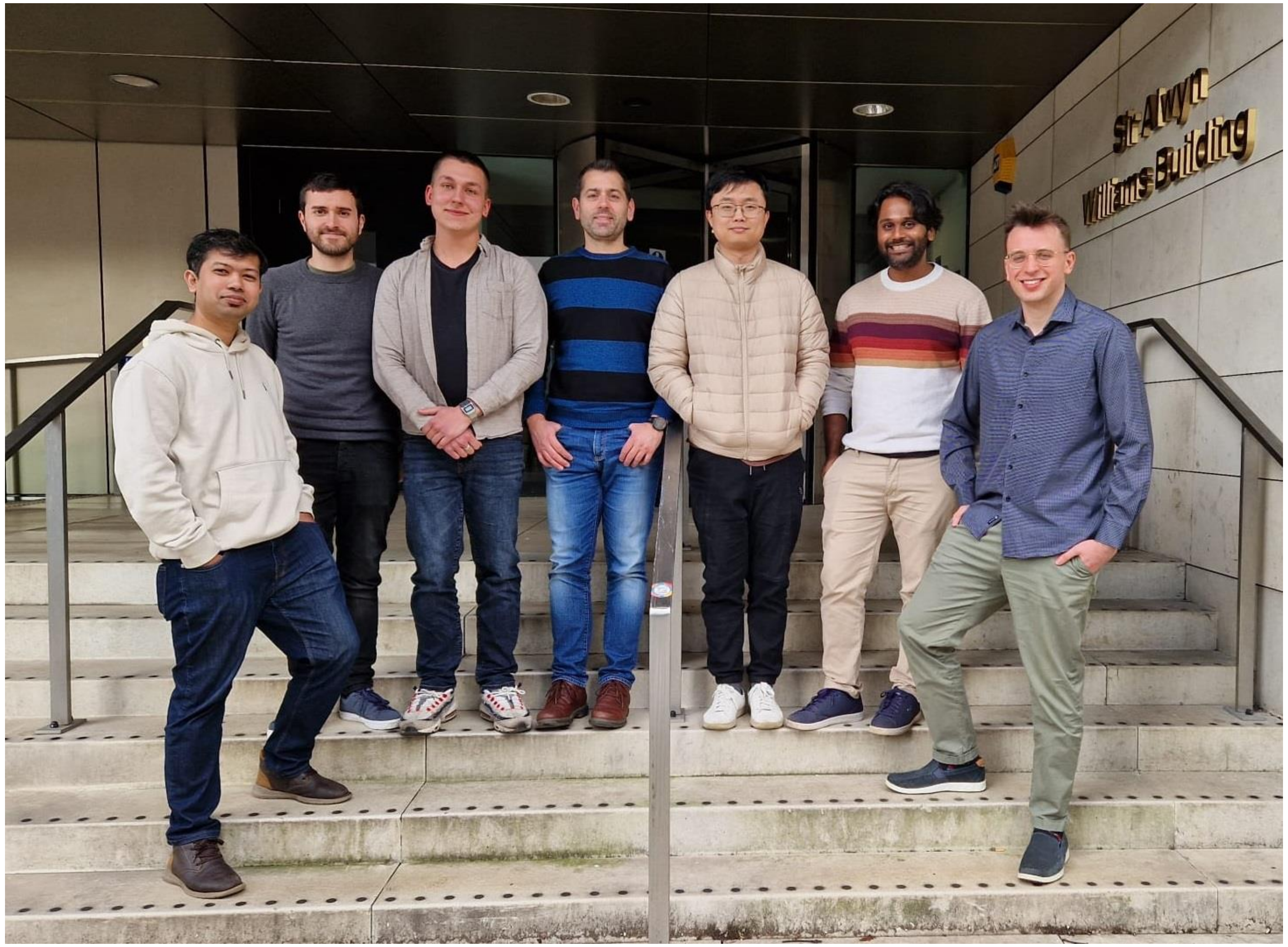
SECDA-LLM

Designing Efficient LLM Accelerators for Edge Devices

Jude Haris, Rappy Saha, Wenhao Hu, José Cano
School of Computing Science
University of Glasgow, Scotland, UK

ARC-LG Workshop @ ISCA’24
Buenos Aires, Argentina – 30/06/2024

Glasgow Intelligent Computing Lab (gicLAB)



School of Computing Science

Systems Section (GLASS)

Computing Systems
&
Machine Learning

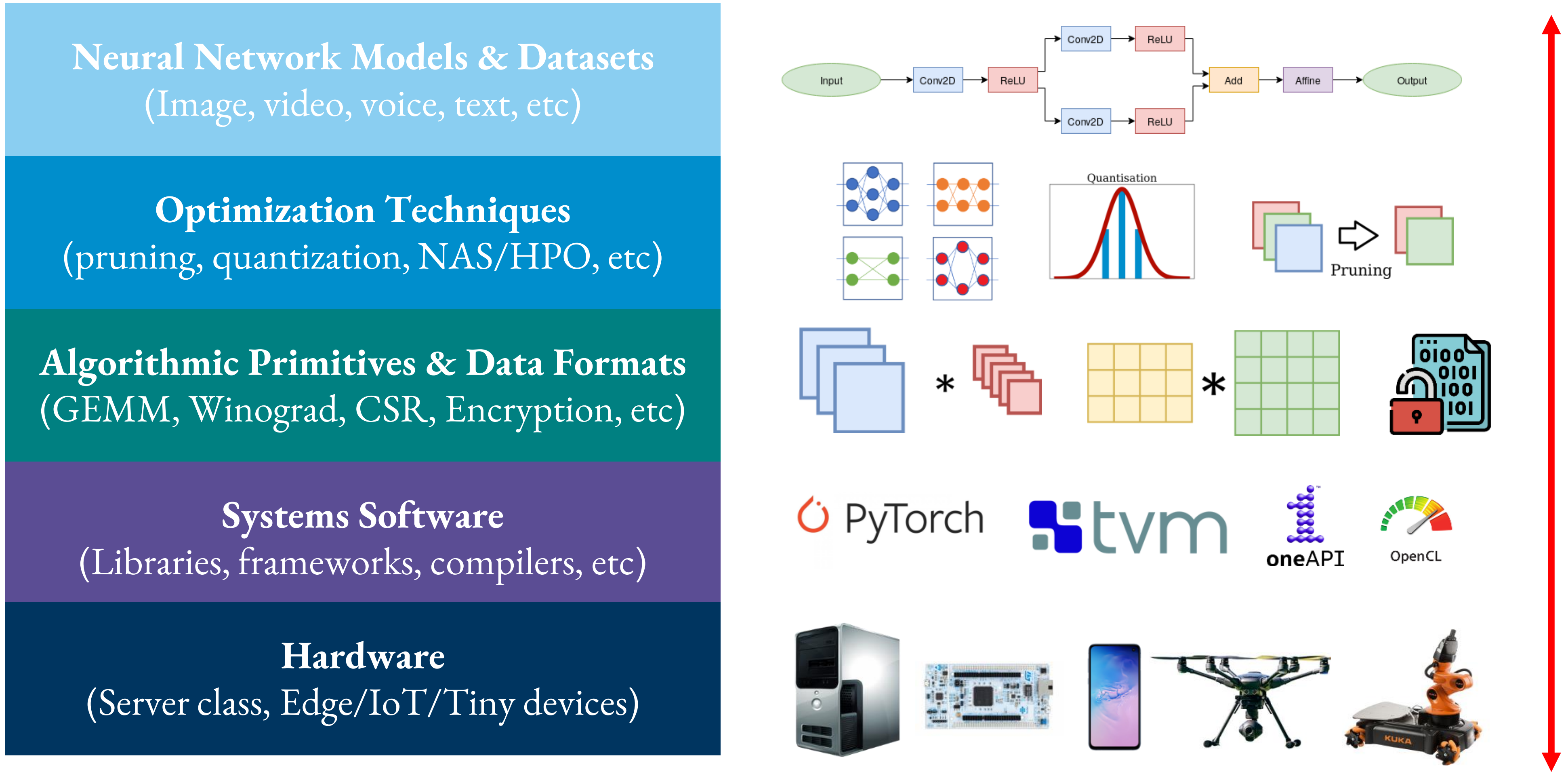
1 PostDoc

5 PhD students

2 MSc students

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

Deep Learning Acceleration Stack (DLAS)


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)
Input
Conv2D
ReLU
Add
Affine
Output
Quantisation
Pruning
PyTorch
tvm
oneAPI
OpenCL
KUKA

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

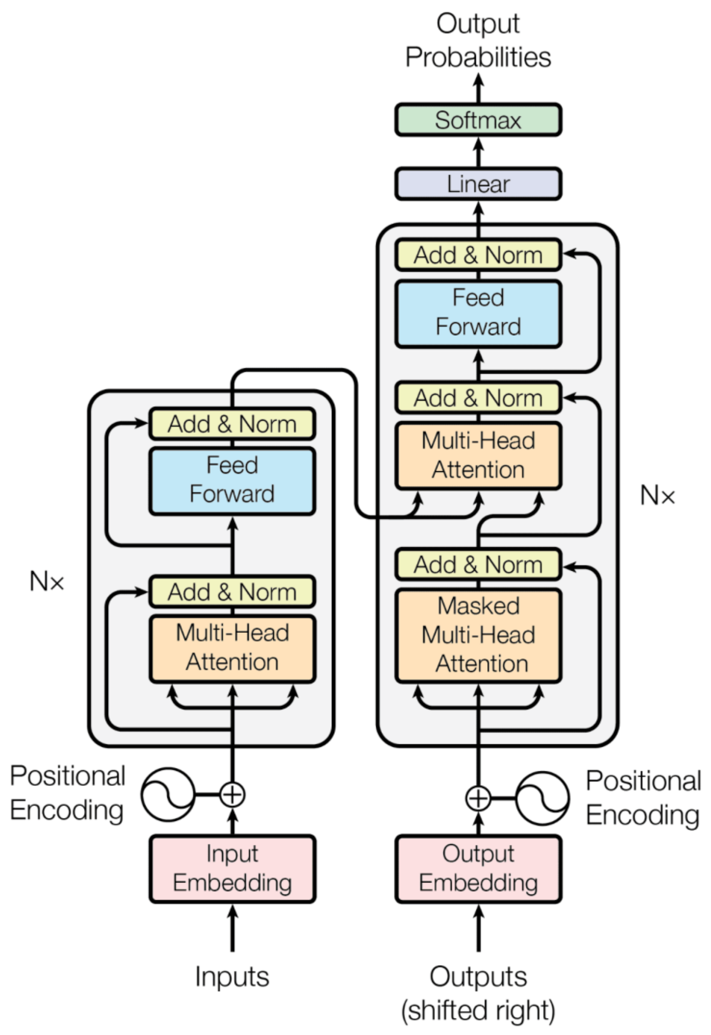
*[P. Gibson, J. Cano, E. J. Crowley, M. O'Boyle, A. Storkey, "*DLAS: A Conceptual Model for Across-Stack Deep Learning Acceleration*", **ACM TACO'24** (conditionally accepted)]

Outline

- Large Language Models (LLMs)
- SECDA Methodology
- SECDA-LLM
- Conclusions and Future Work

Large Language Models (LLMs)

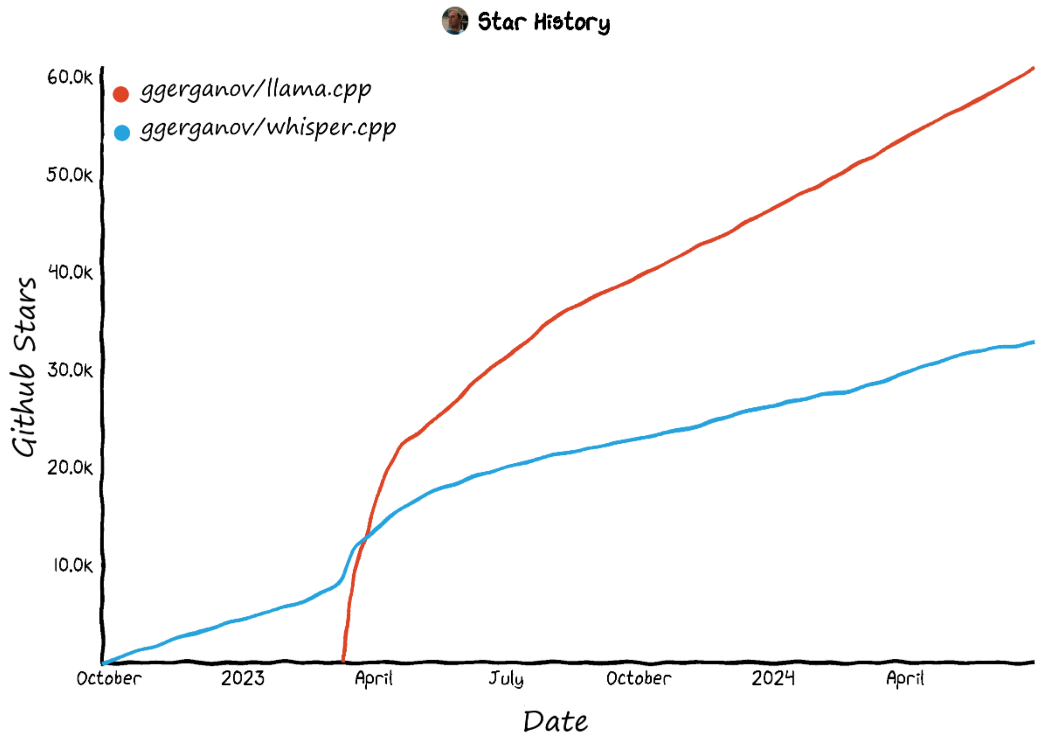
- LLMs are a family of models that use the Transformer-based architecture
- Great at solving many language related tasks
  - Text Generation, AI assistants, Code generation
- Greatly increase upon the number of parameters used
  - PaLM 2 apparently has 340 billion parameters!
- Many optimization techniques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



llama.cpp

- A pure C/C++ library with minimal external dependencies
- Enables LLM inference with minimal setup on wide range of hardware devices
- Supports multi-modal and custom LLMs, along with well-known LLM models, (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 fast development and active community

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



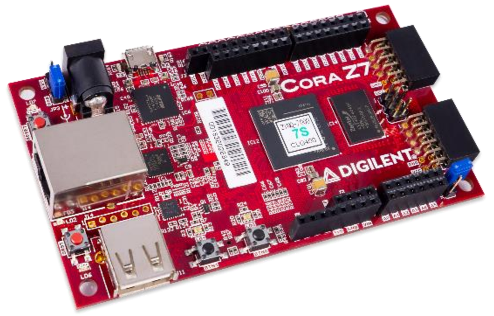
LLMs on Resource Constrained Edge Devices

- 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

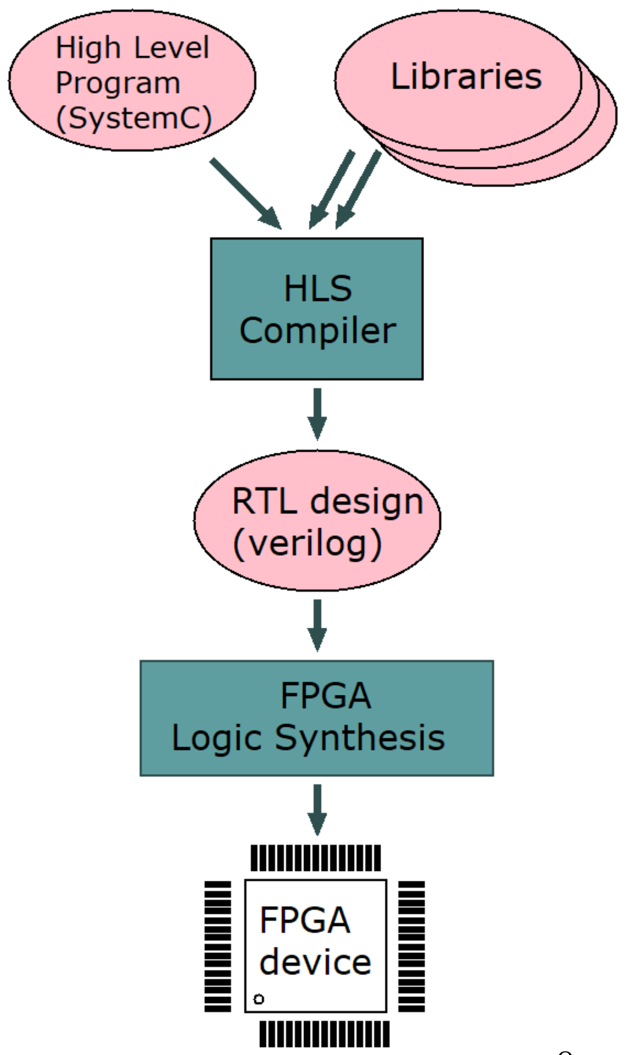
- Hence, we need to develop **specialized hardware accelerators** to efficiently process LLMs with limited resources



Developing Specialized Hardware Accelerators

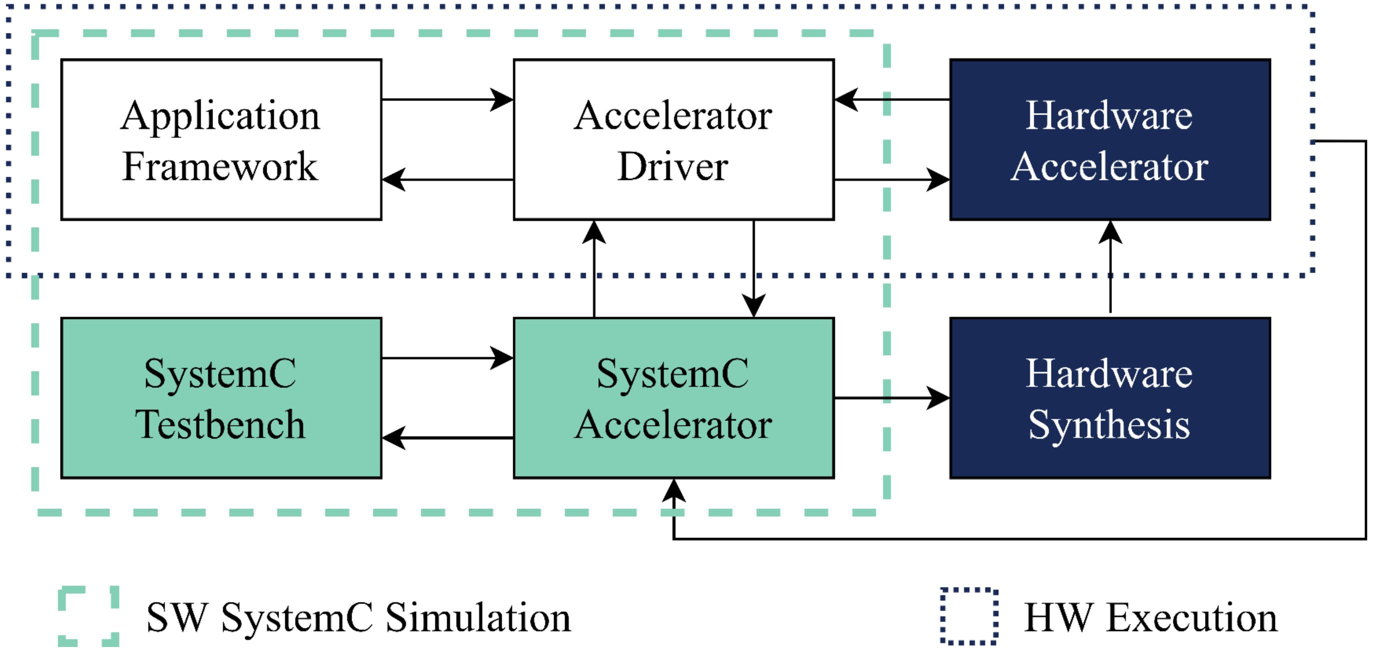
- **Motivation**: specialized hardware accelerators (ASICs, FPGAs, etc) can make deep learning faster and more energy efficient (e.g. at the edge)
  - FPGAs are reconfigurable circuits commonly present in edge devices
- **Problem**: current solutions for designing DNN accelerators for edge devices with 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 (for edge devices)
  - Combines cost-effective SystemC simulation with hardware execution

High Level Synthesis (HLS)



SECDA Methodology: Overview

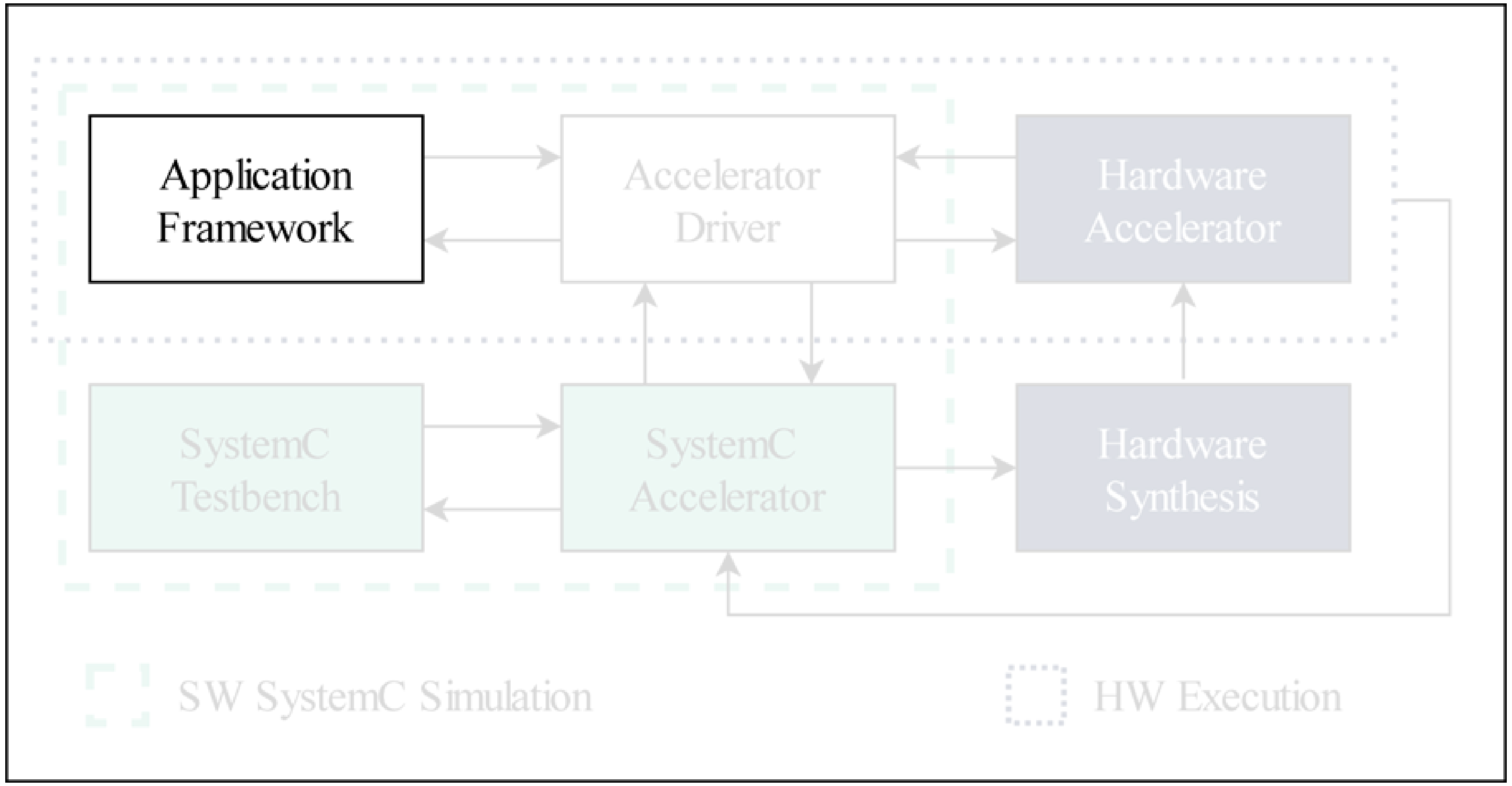
- **SECDA**: SystemC Enabled Codesign of DNN Accelerators



*[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**]

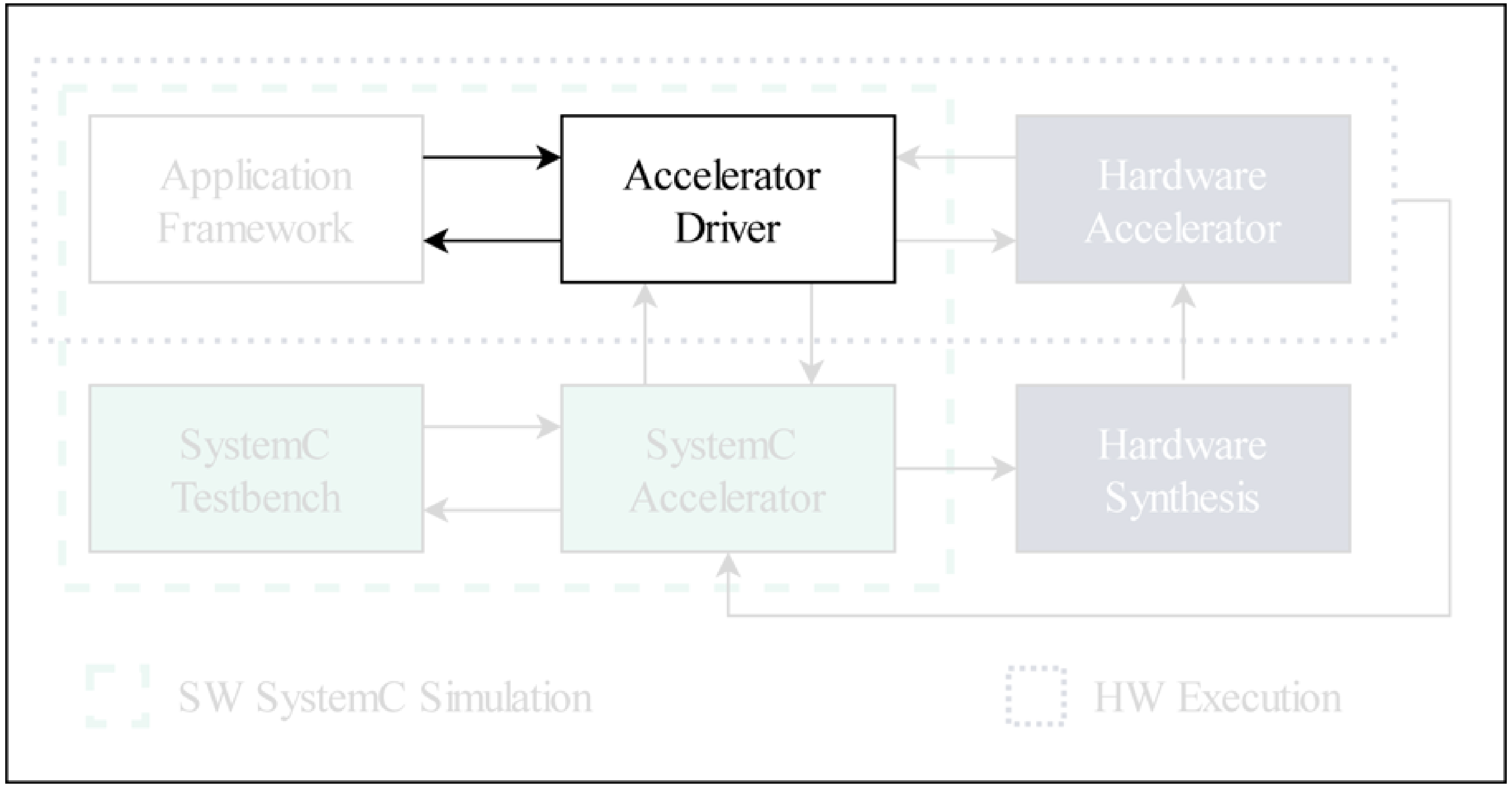
SECDA Methodology: Components

- 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*
    - …



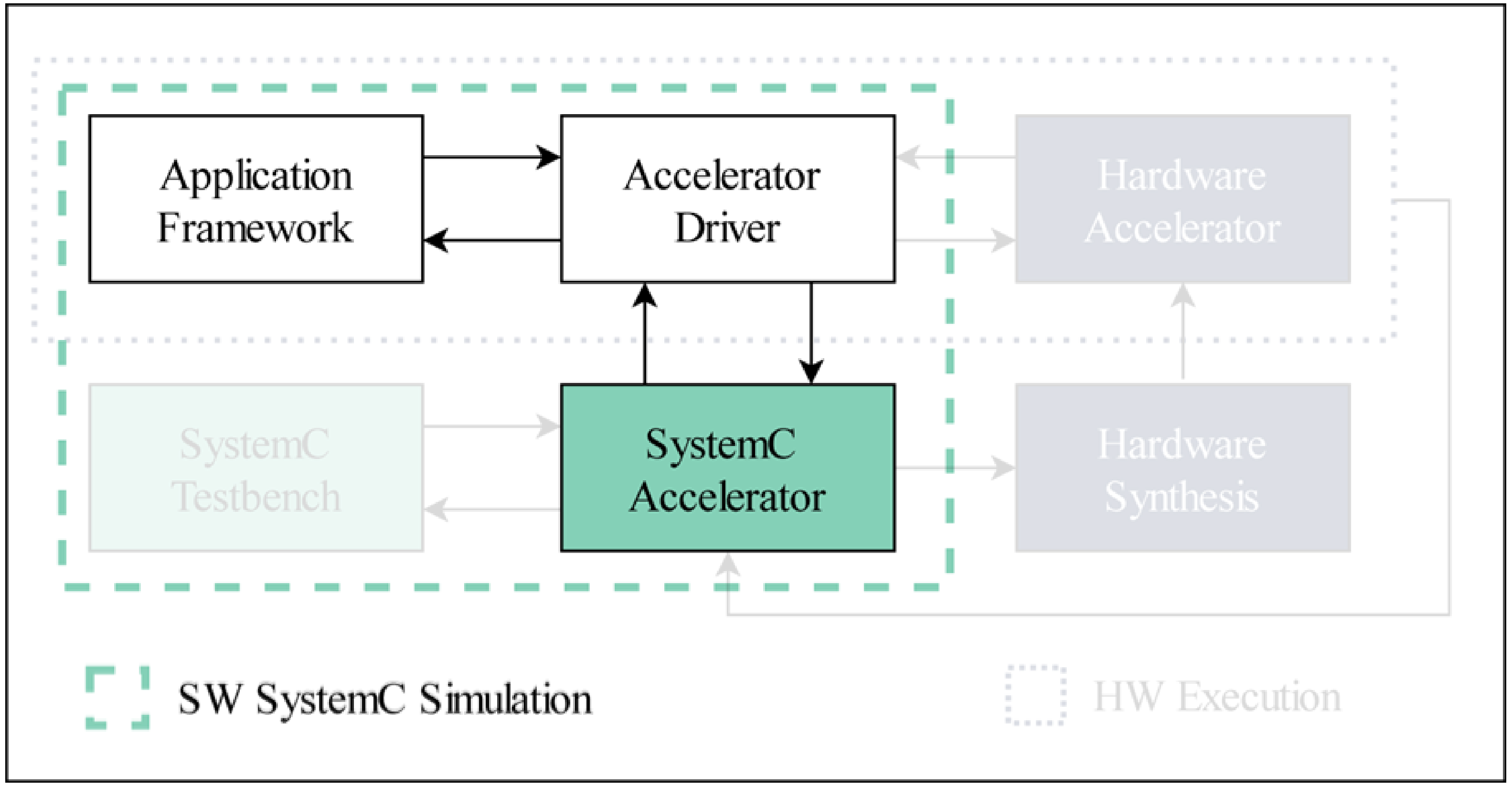
SECDA Methodology: Components (2)

- 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
    - …



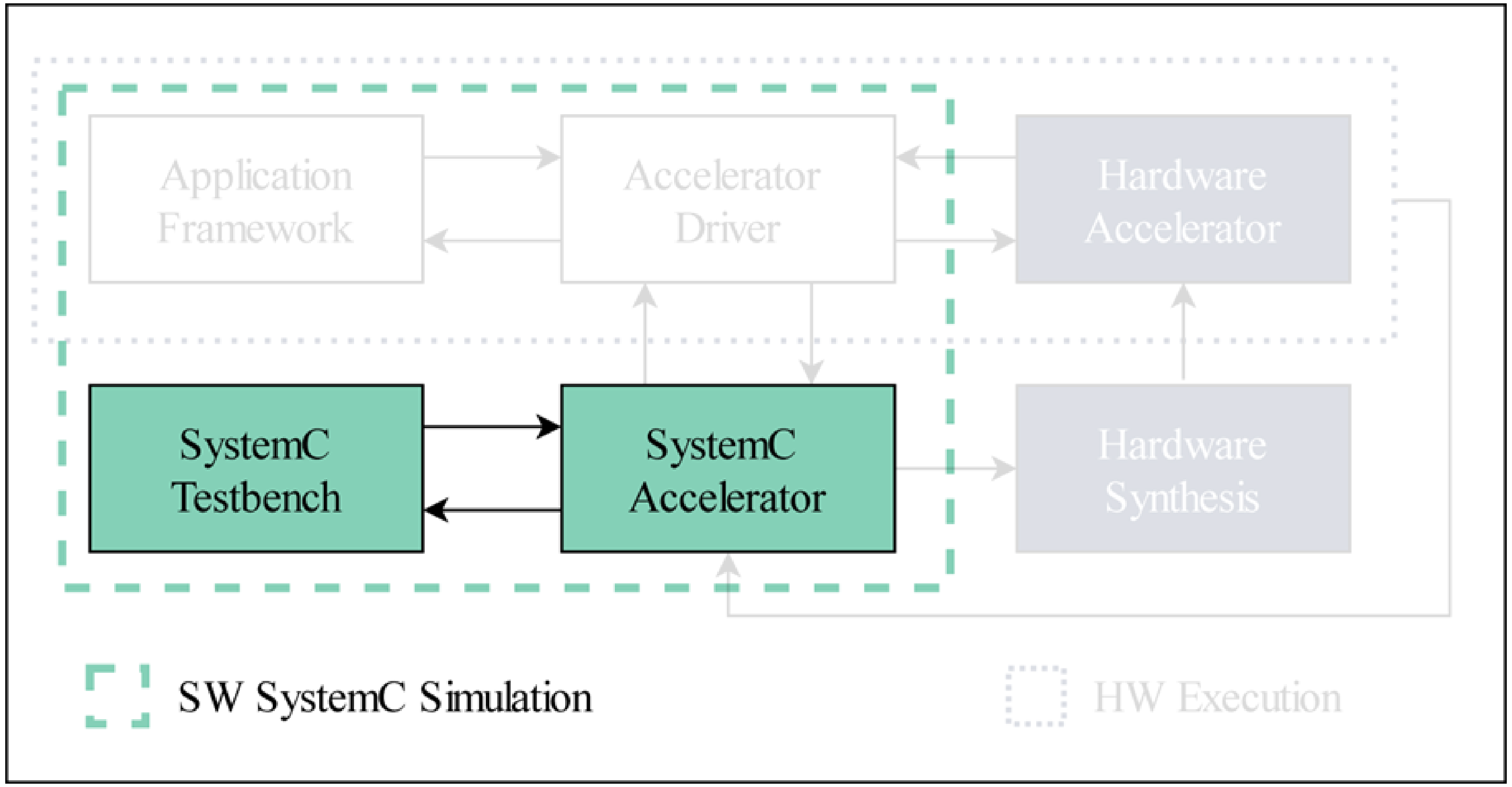
SECDA Methodology: Components (3)

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



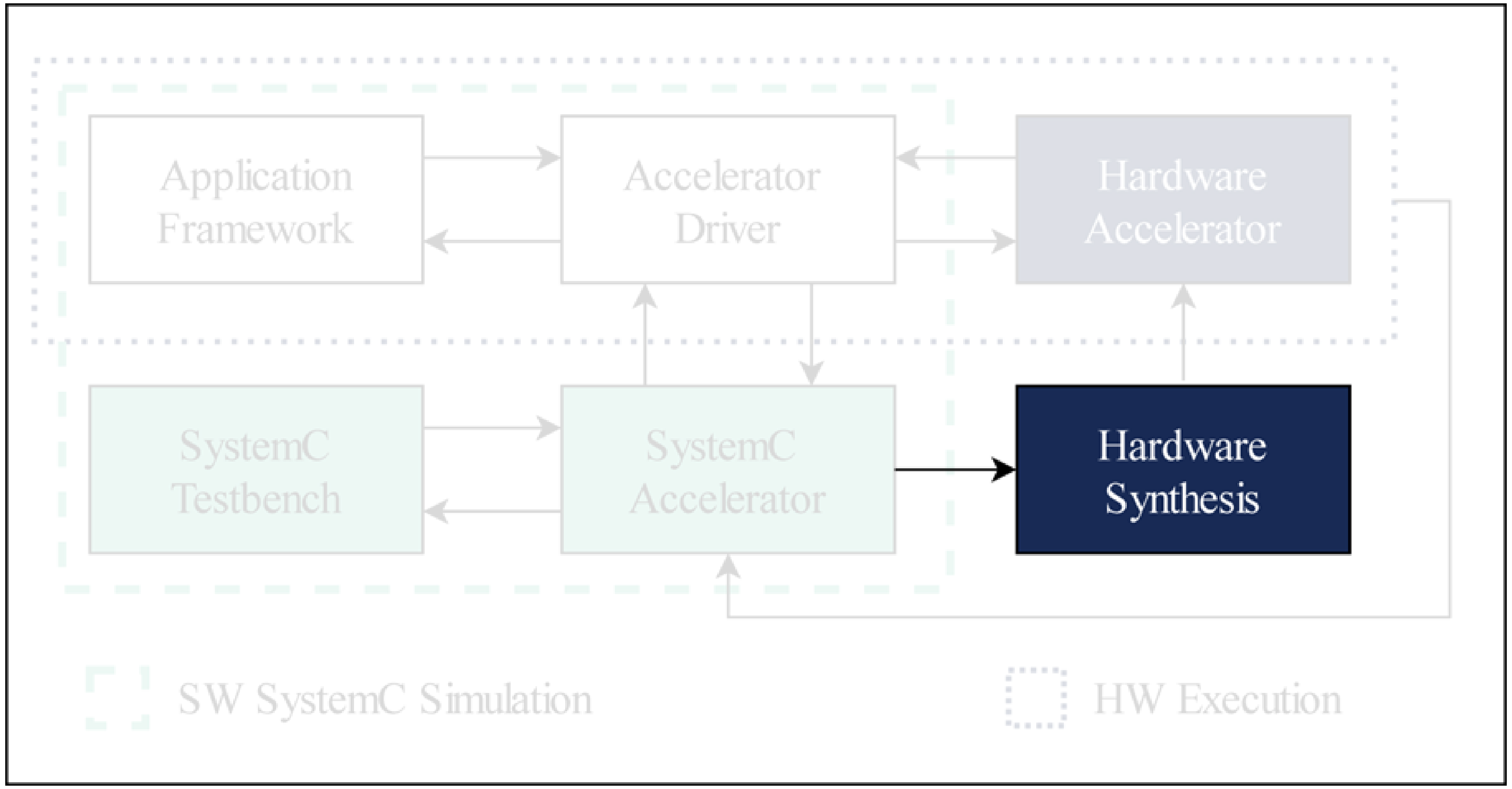
SECDA Methodology: Components (4)

- 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



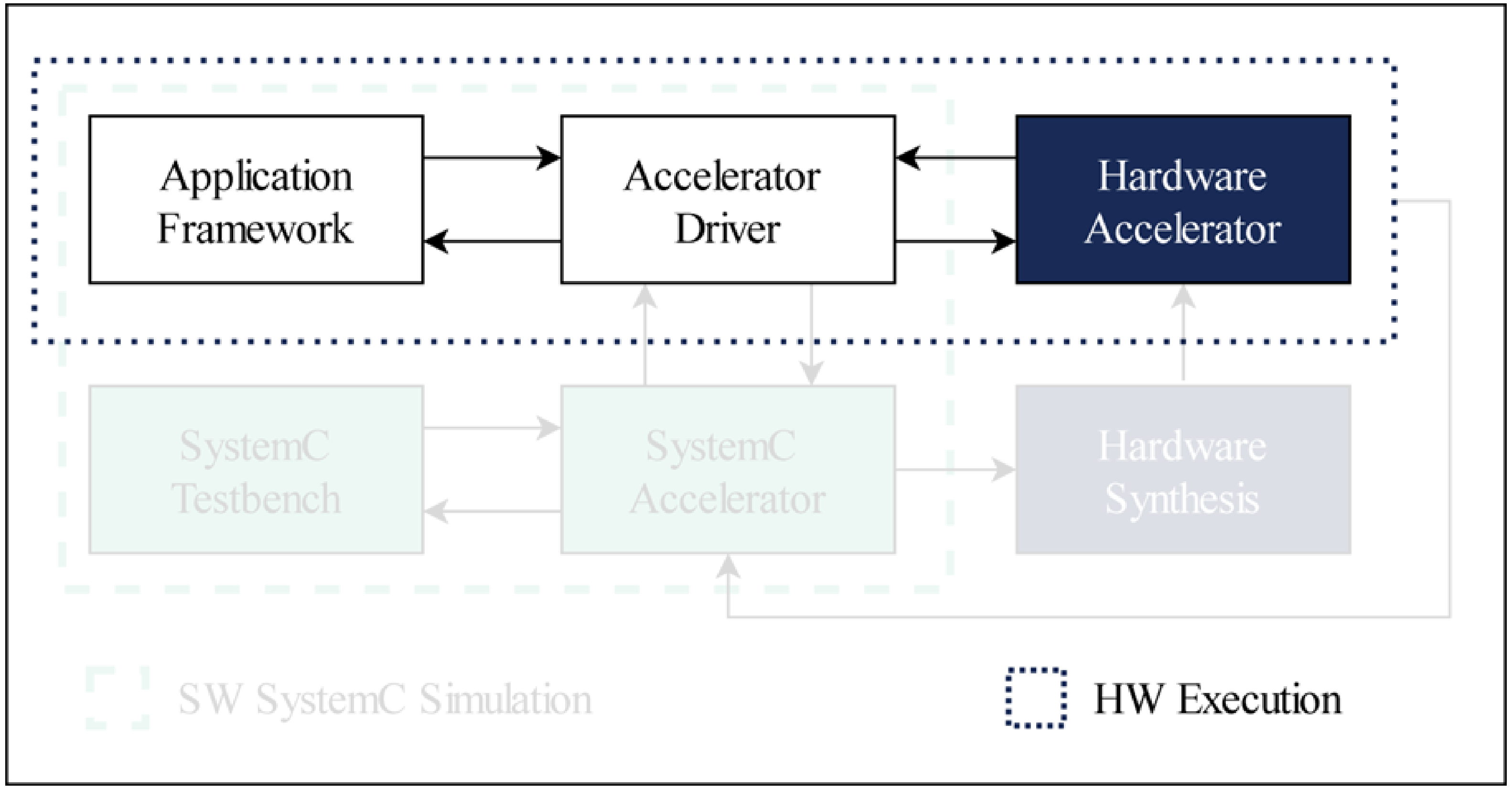
SECDA Methodology: Components (5)

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



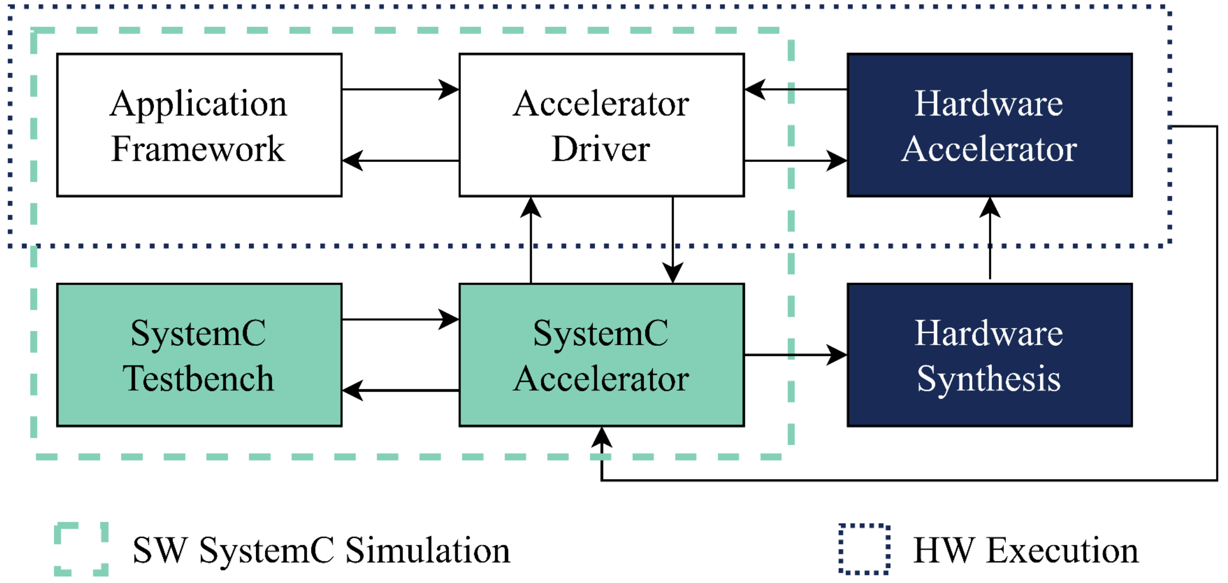
SECDA Methodology: Components (6)

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



SECDA Methodology: Design Loop

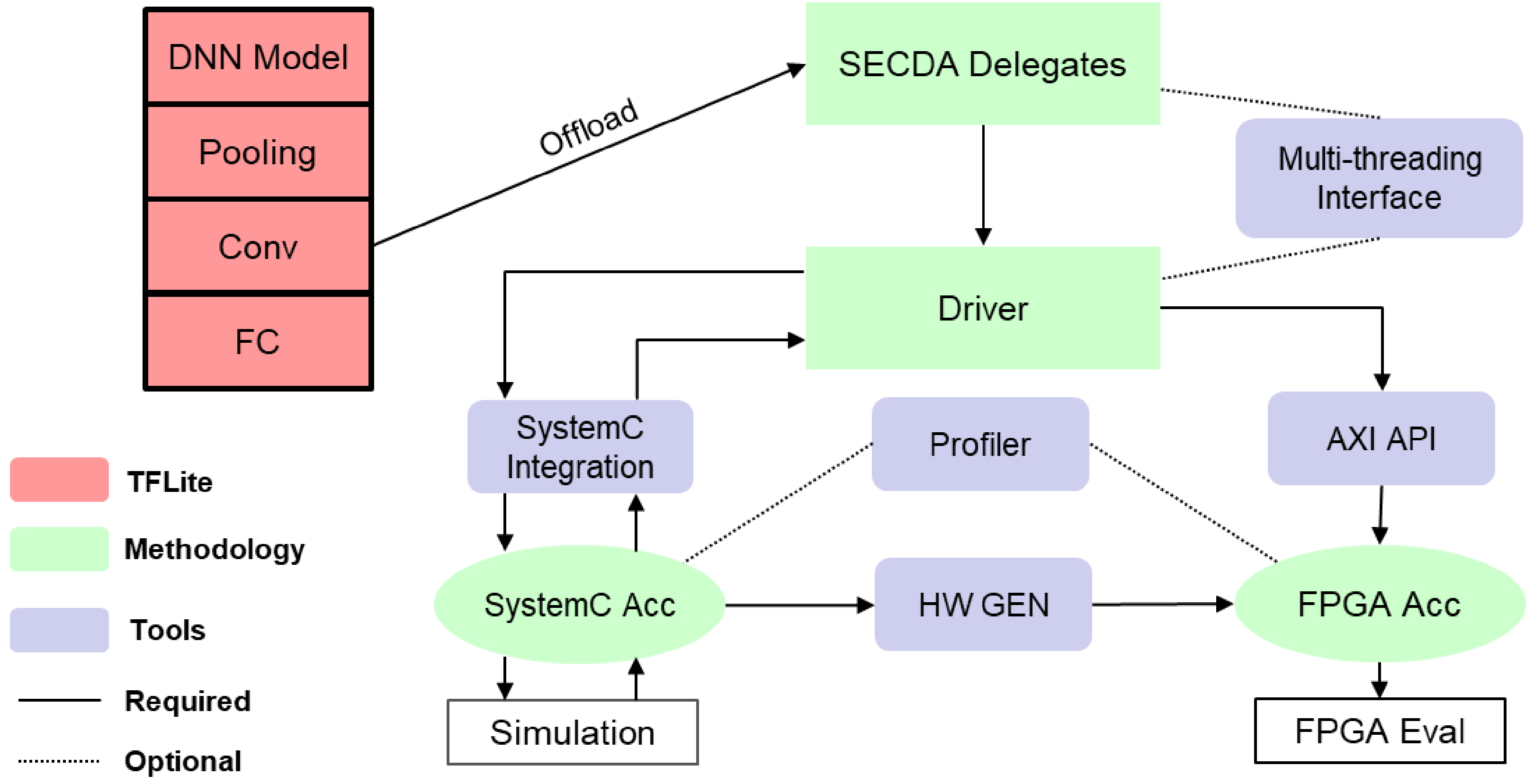
- Logic synthesis is time consuming
- SECDA reduces the number of logic synthesis iterations via simulation
- Accelerator/driver (hw/sw) co-design enables easier full system integration



- **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

SECDA-TFLite

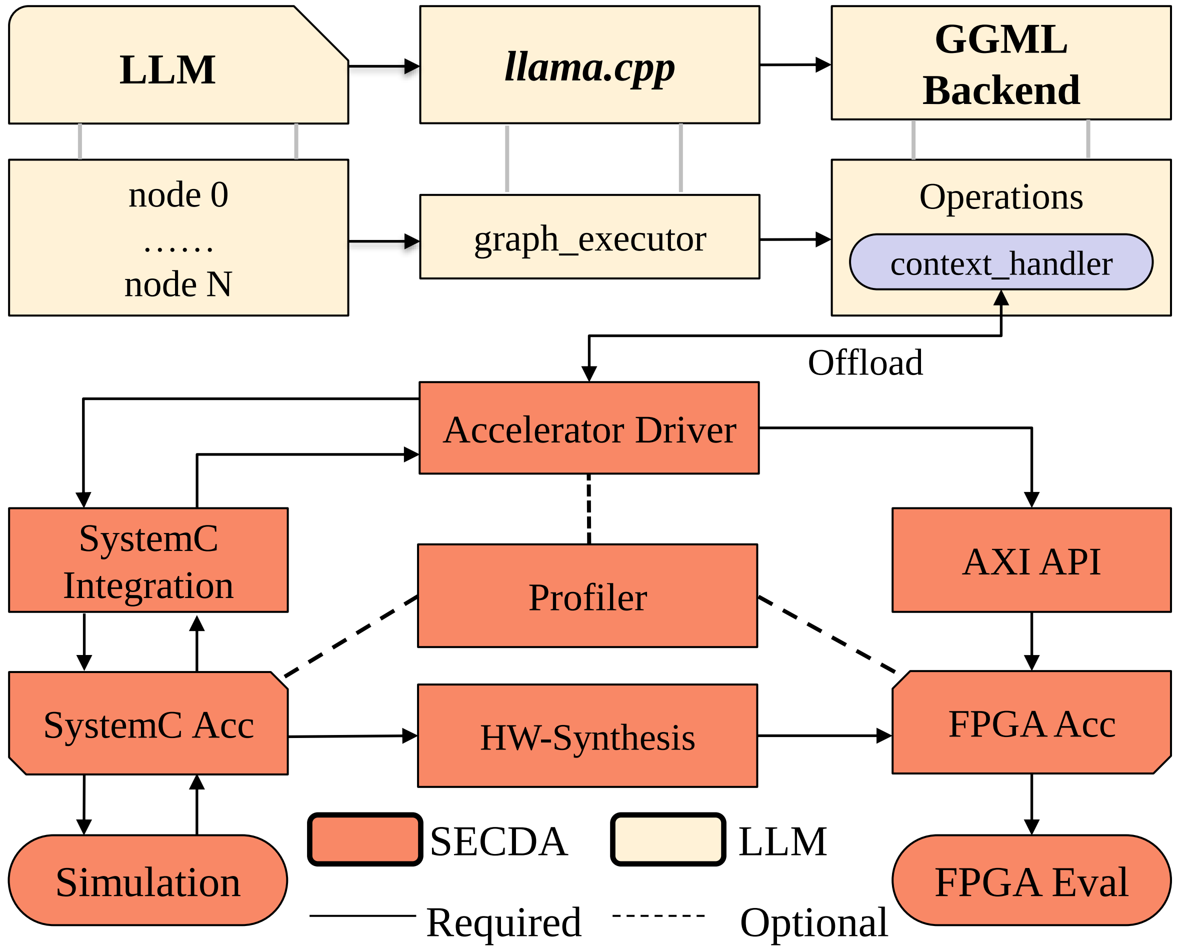
- A toolkit for designing custom FPGA-based accelerators for TFLite
- Instantiates the SECDA methodology within TFLite
- Enables fast prototyping and integration of new accelerators with significantly **reduced initial setup costs**



*[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**]

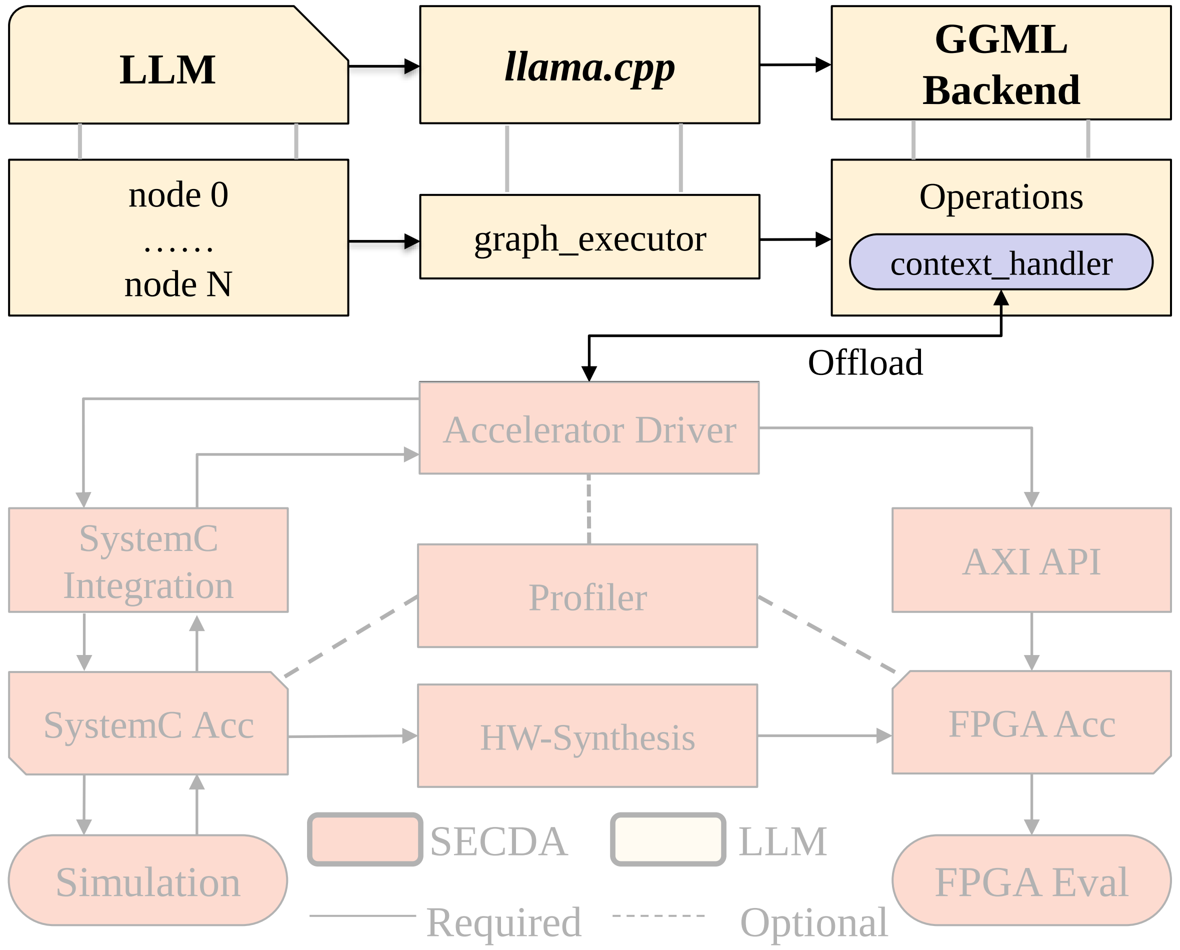
SECDA-LLM

- 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**



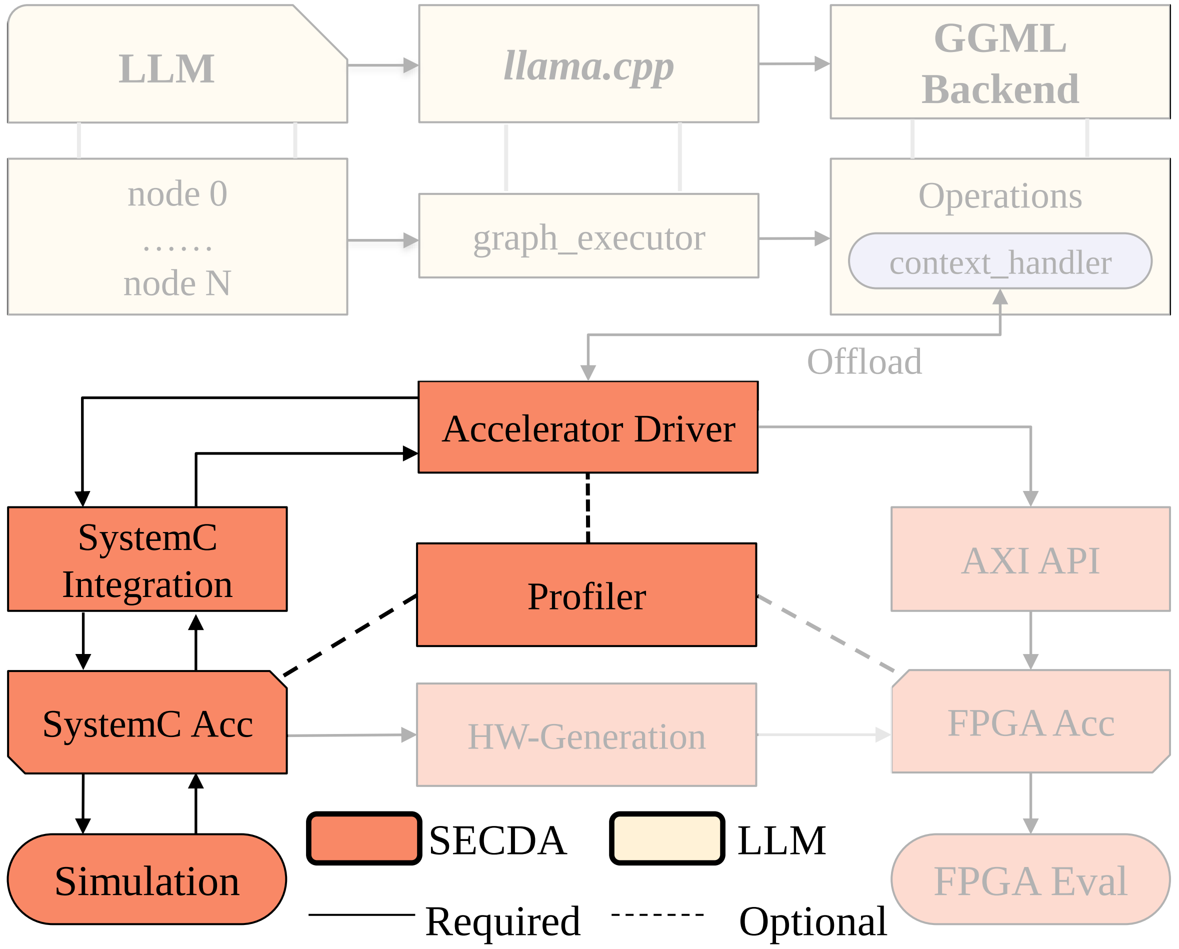
Connecting *llama.cpp*

- *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**



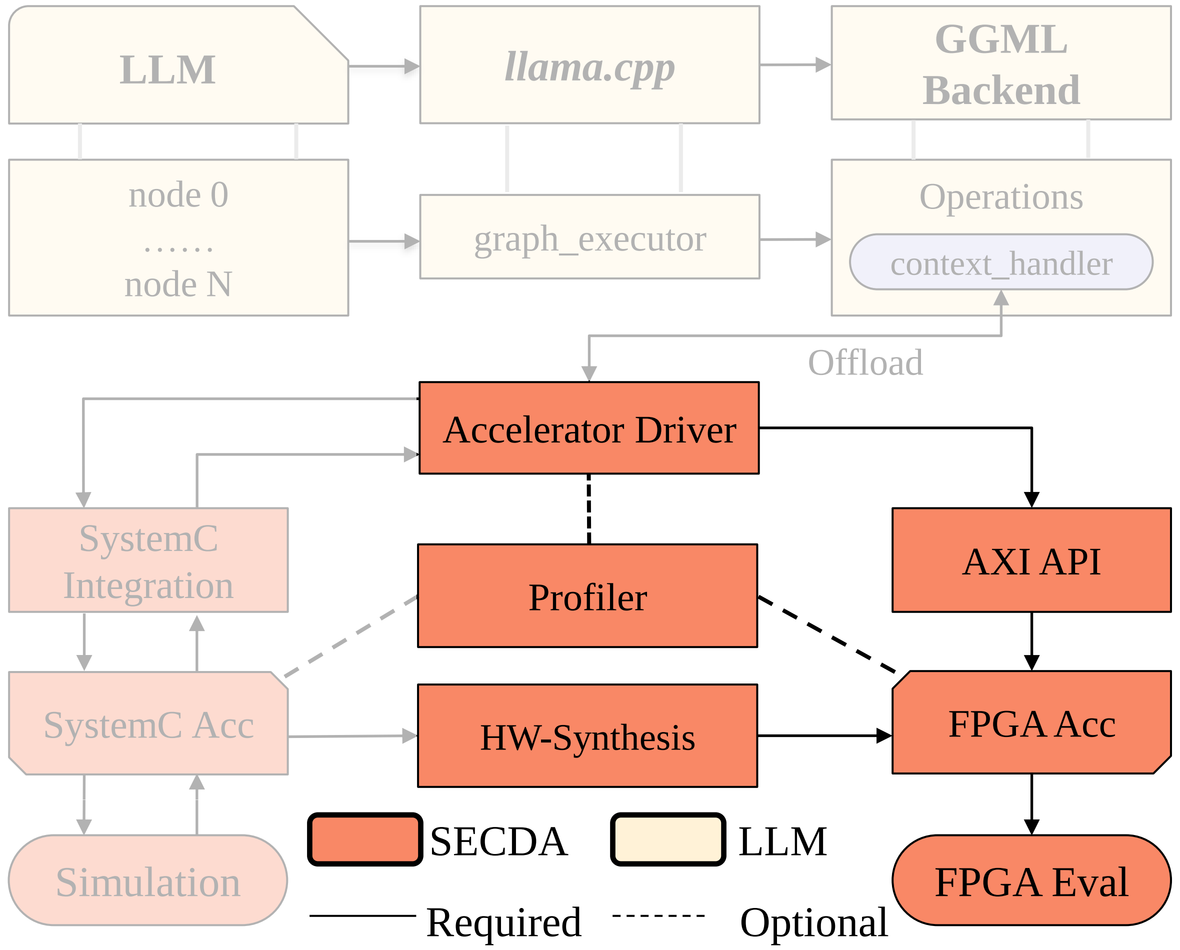
Simulation Design Loop

- 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



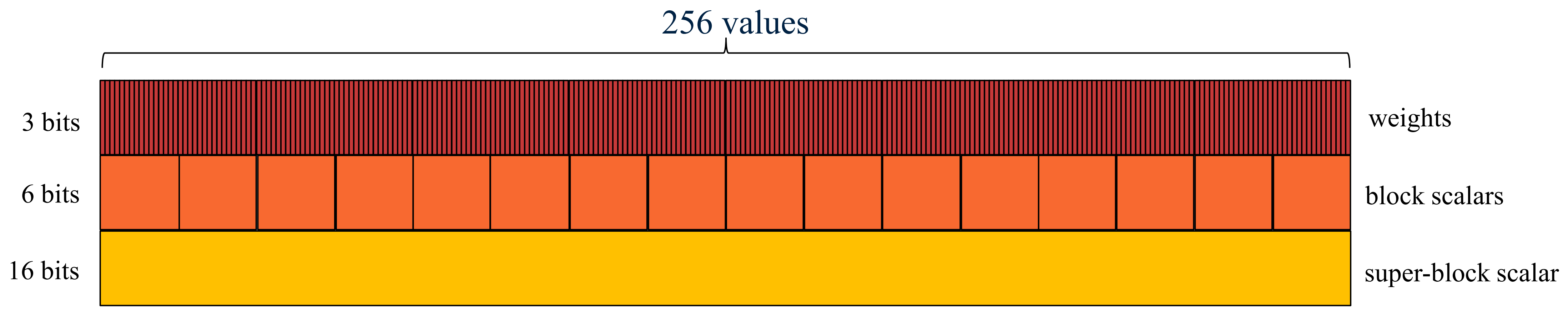
Hardware Generation and Evaluation

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



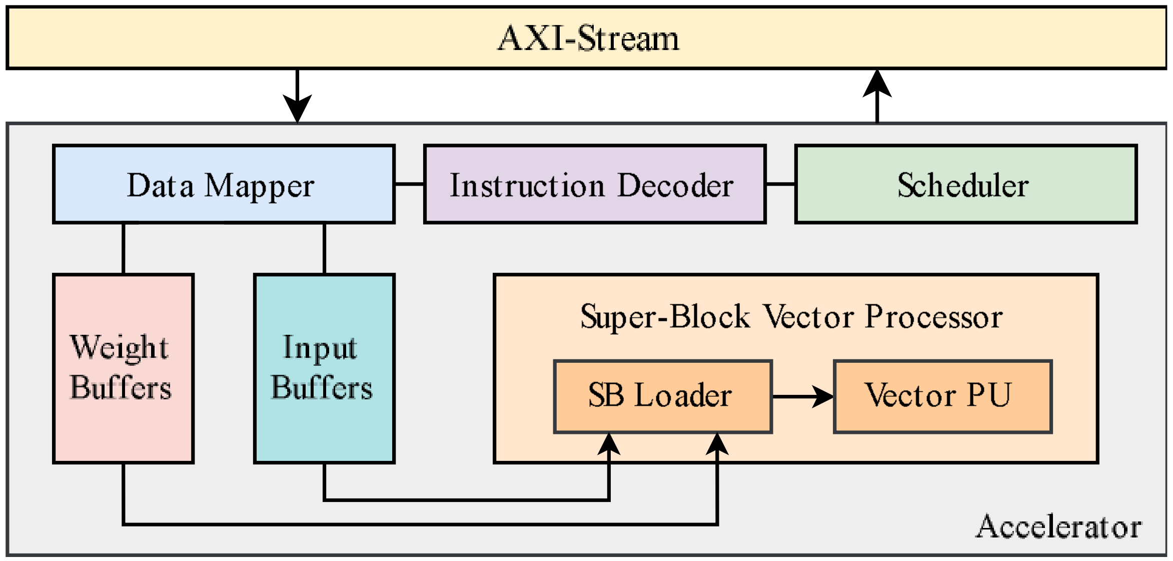
Case study: MatMul Acceleration

- 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

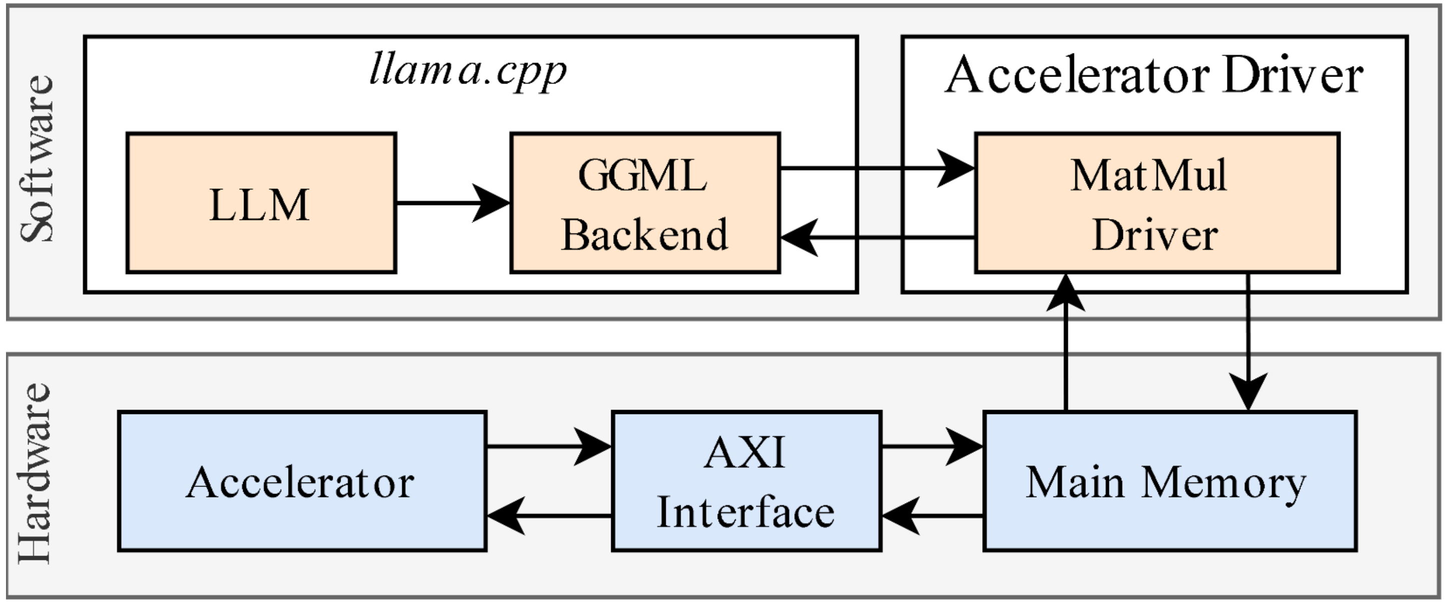
Case study: Accelerator Design



- 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

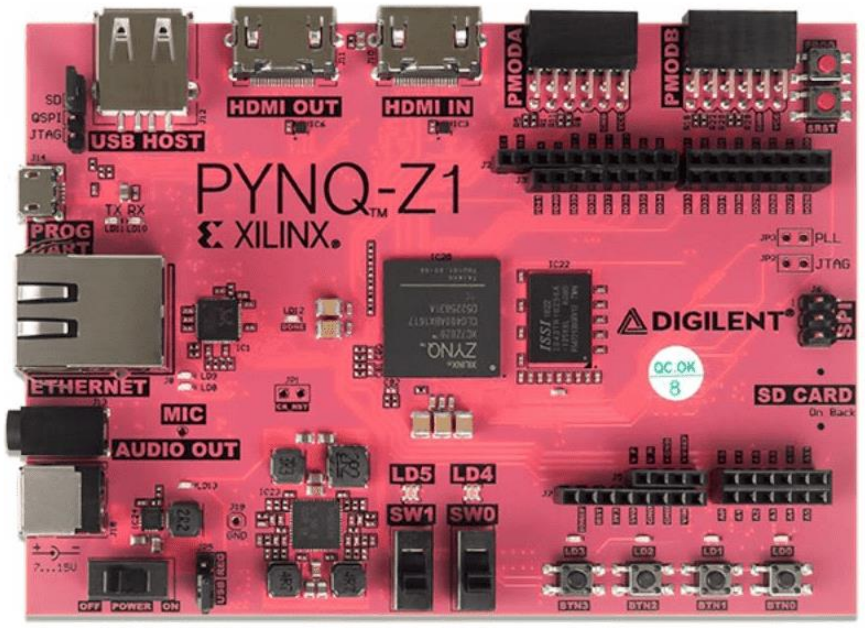
Runtime Model (HW execution)

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



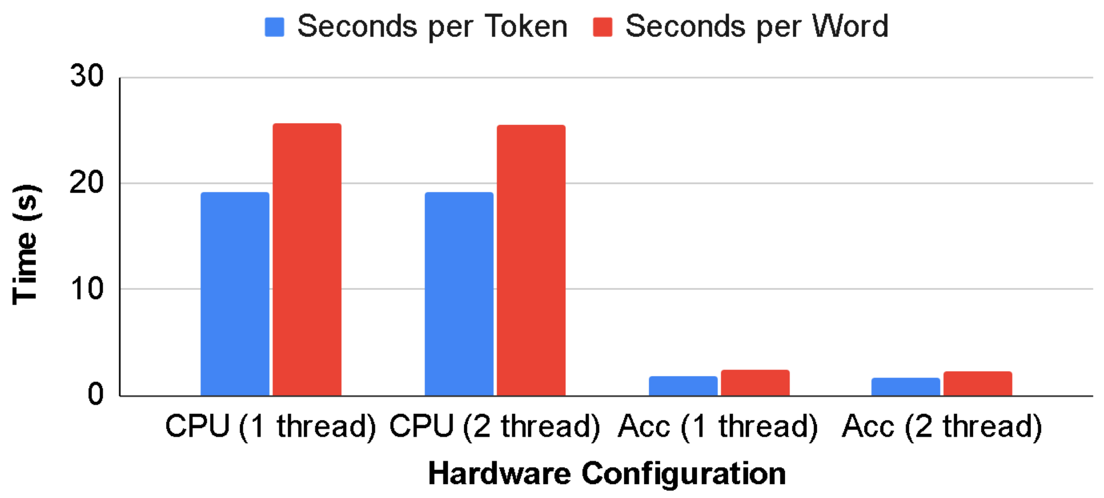
Evaluation: Experimental Setup

- **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

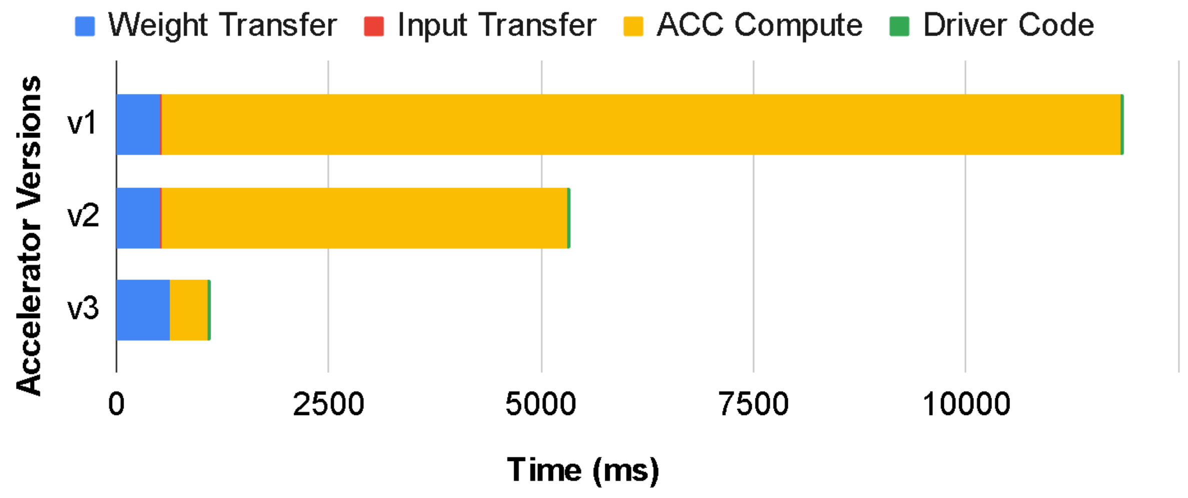


Evaluation: Results

- 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



Conclusions and Future Work

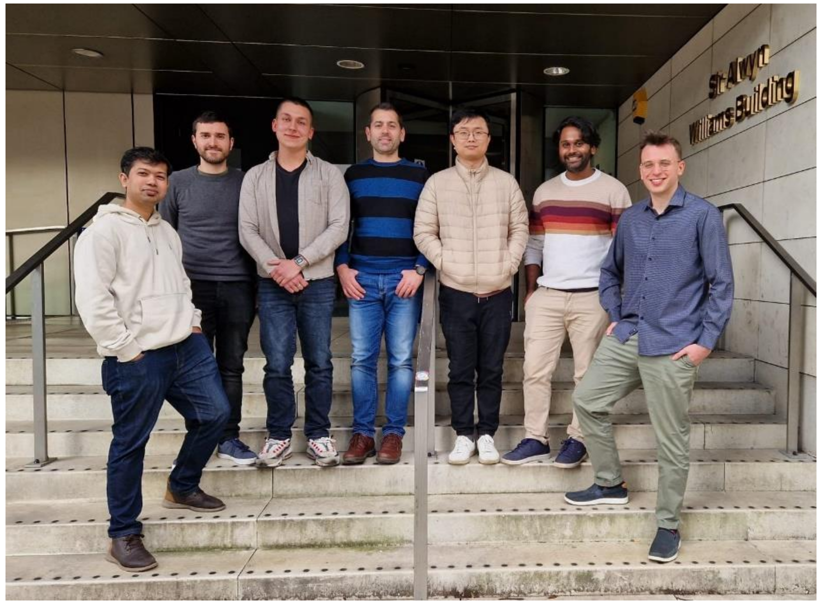
- **SECDA-LLM** is a new toolkit that improves/eases the development of new FPGA-based accelerators for edge LLM inference employing the SECDA design methodology

- As a **case study** we design and implement a MatMul accelerator and improve performance by 11x compared the CPU-only baseline for the TinyLlama model on a resource constrained edge FPGA

- We plan to expand SECDA-LLM as an open-source platform to enable collaborative development and continuous improvement of LLMs' performance of resource constraint edge devices

University of Glasgow

Acknowledgements



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

UNIVERSITY OF OXFORD

University of Essex

Northeastern University

ETH zürich

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSIDAD DE MURCIA

UJI UNIVERSITAT JAUME I

4) Collaborators from Industry and Labs

ST

AMD

NVIDIA

arm

ultrasoc

Pacific Northwest NATIONAL LABORATORY

SECDA-LLM

Designing Efficient LLM Accelerators for Edge Devices

Jude Haris (j.haris.1@research.gla.ac.uk), José Cano (Jose.CanoReyes@glasgow.ac.uk)
School of Computing Science
University of Glasgow, Scotland, UK

Thank you! Questions?