Towards Rapid Exploration of Heterogeneous TinyML Systems using Virtual Platforms and TVM's UMA

Samira Ahmadifarsani, Rafael Stahl, Philipp van Kempen, Daniel Mueller-Gritschneder and Ulf Schlichtman

Chair of Electronic Design Automation, Technical University of Munich
Munich, Germany

21 September 2023

Federal Ministry of Education and Research

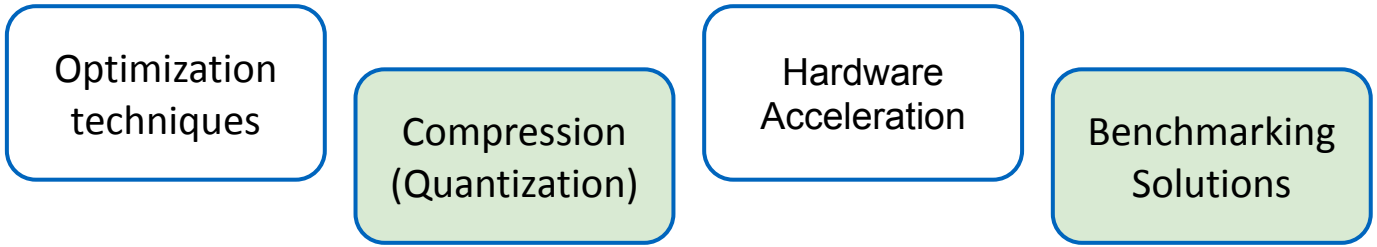
ECOMAI

Content

- Introduction
- Background
  - UMA Interface
  - QNN Dialect
- Approach
  - Integration Process
  - Support for QNN operators
- Virtual Prototyping
- Evaluation
- Conclusion

Introduction

- **TinyML**: Bringing machine learning tasks to **ultra-low-power** edge devices
  - Resource efficiency
  - Power efficiency
  - Real-time processing
  - Hardware and software heterogeneity



Introduction

- Deep learning **compilers** for heterogeneous TinyML systems
  - TFLite, XLA, Glow and **TVM**
- **TVM and its extensions**: the best solution to support heterogeneous platforms [1]
  - Bring Your Own Codegen (BYOC) [2]
  - Universal Modular Accelerator (**UMA**) [3]
- **MLonMCU**: benchmarking tool, capable of virtual prototyping [4]
- **Goal**

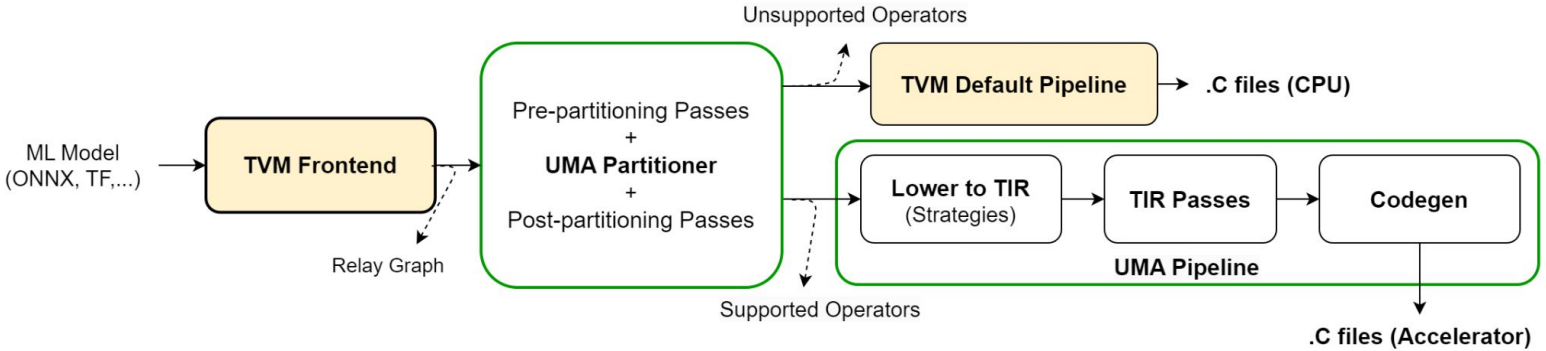
Integrating **UMA** and **Virtual Prototyping** to **Explore** the Performance Improvement achieved by an **accelerator** for **Quantized** Neural Networks

Content

- Introduction
- Background
  - UMA Interface
  - QNN Dialect
- Approach
  - Integration Process
  - Support for QNN operators
- Virtual Prototyping
- Evaluation
- Conclusion

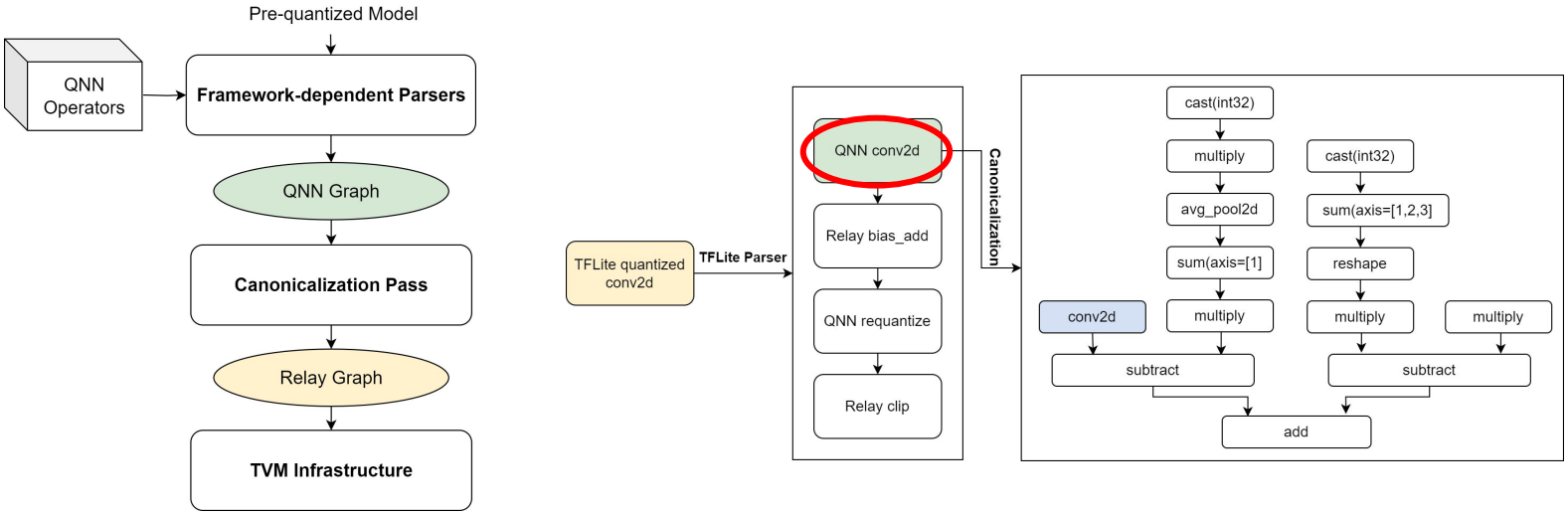
UMA Interface

- An **easy-to-use** structure to integrate new hardware **accelerators** into **TVM**
- Providing file structures, Python interface classes, and an API
- Enabling the straightforward **offloading** of specific operator patterns to on-chip accelerators



Quantized Neural Network (QNN)

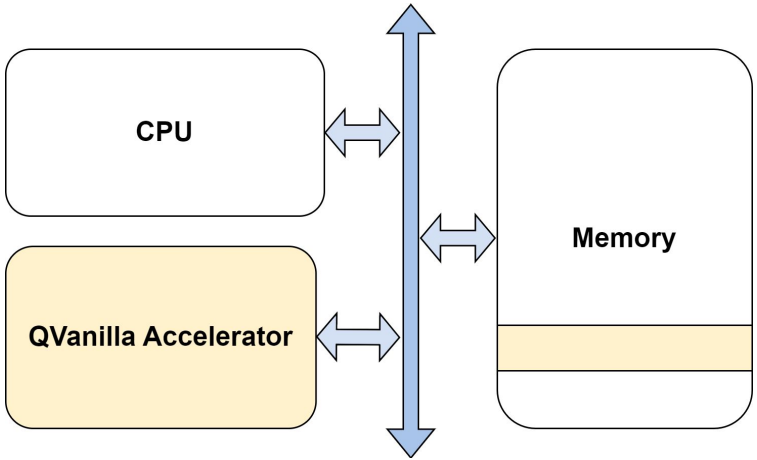
- **QNN Dialect** [5]
  - Extending **TVM's** internal representation with a quantization context
  - A higher-level IR layered on top of the graph-level IR



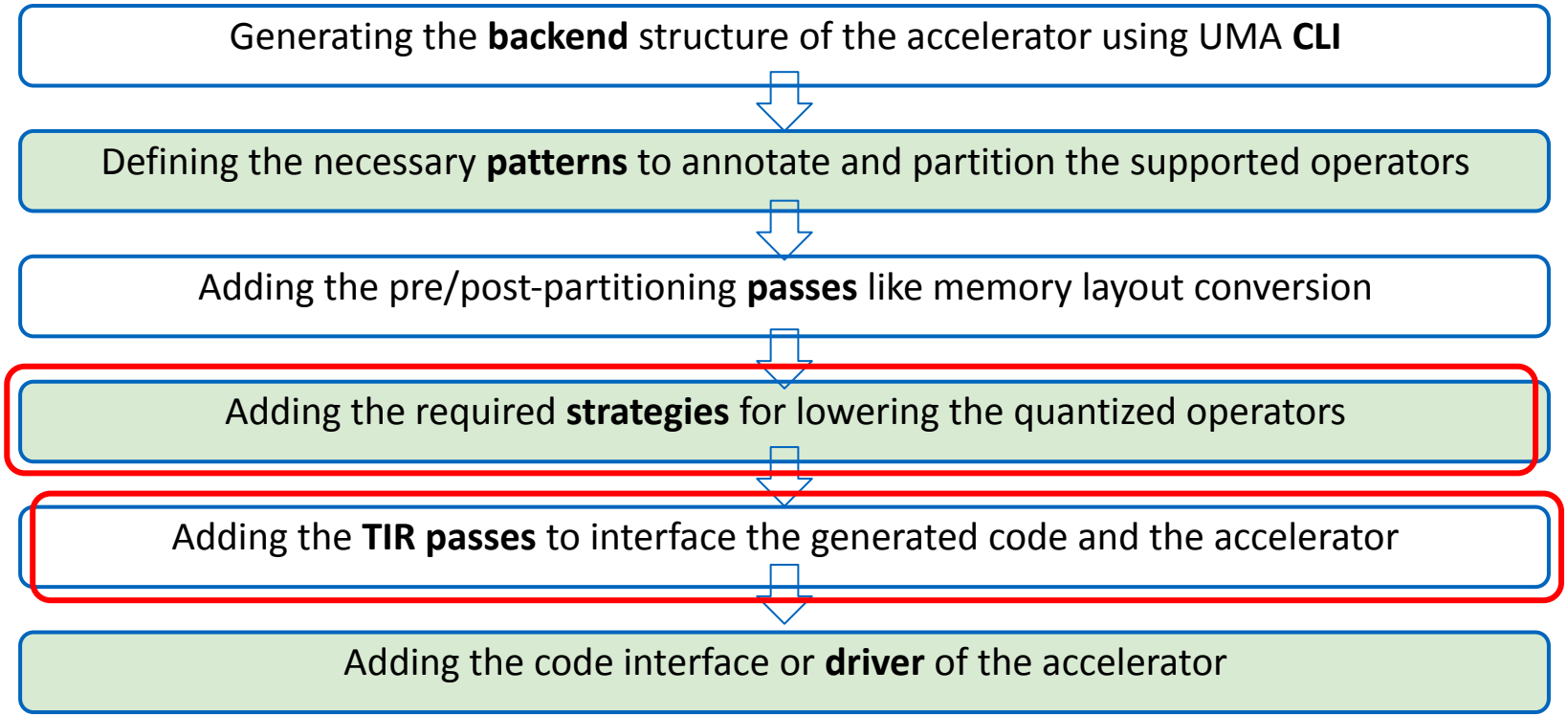
Content

Study Case

- **QVanilla** Accelerator: a simple mock accelerator based on UMA's Vanilla [7]
- Supported operators: quantized convolution + bias addition
- **Offloading** the supported operator patterns to QVanilla



UMA Integration Process

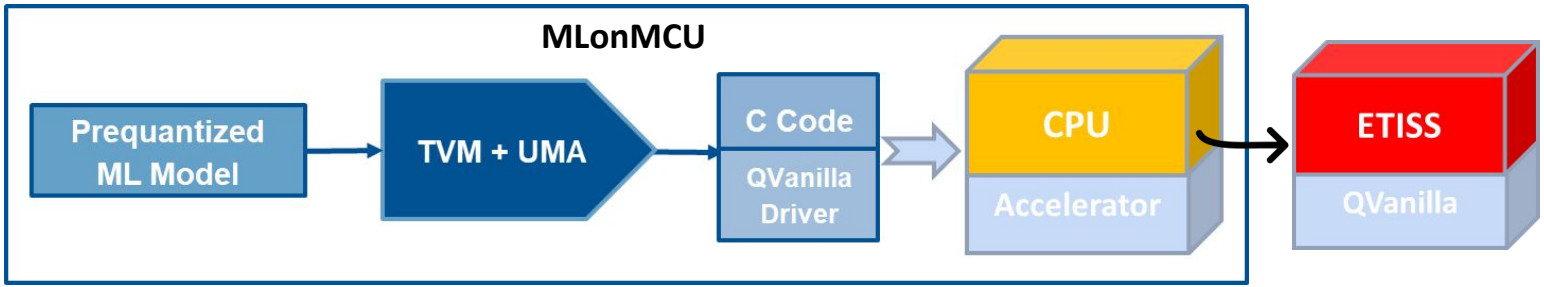


Content

- Introduction
- Background
  - UMA Interface
  - QNN Dialect
- Approach
  - Integration Process
  - Support for QNN operators
- Virtual Prototyping
- Evaluation
- Conclusion

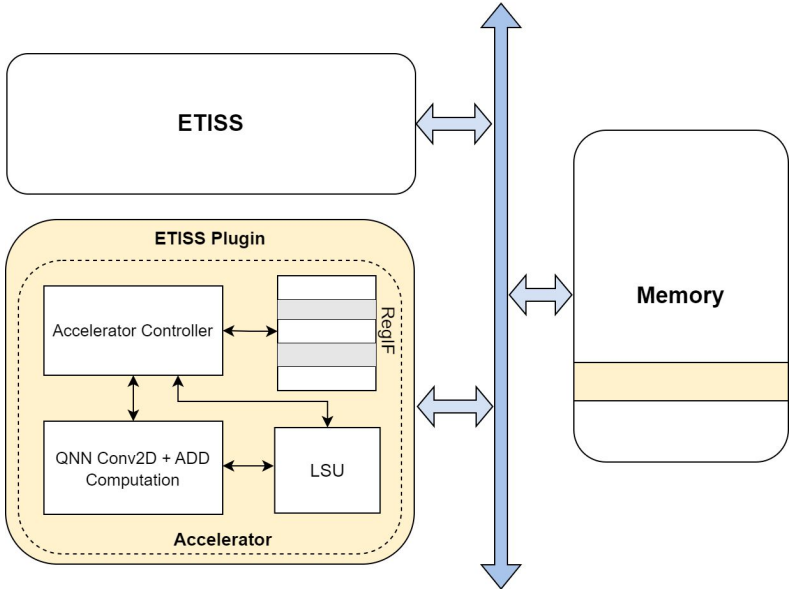
Virtual Prototyping

- **MLonMCU** is a benchmarking tool for the deployment of machine learning
  - TinyML frameworks: TFLM and **TVM**
  - **ETISS** target [6]
    - Simulating a 32 bit RISC-V microcontroller



The Virtual Prototype of the Target

- **ETISS Plugin** Mechanism for integrating QVanilla as a **memory-mapped** peripheral



Content

- Introduction
- Background
  - UMA Interface
  - QNN Dialect
- Approach
  - Integration Process
  - Support for QNN operators
- Virtual Prototyping
- Evaluation
- Conclusion

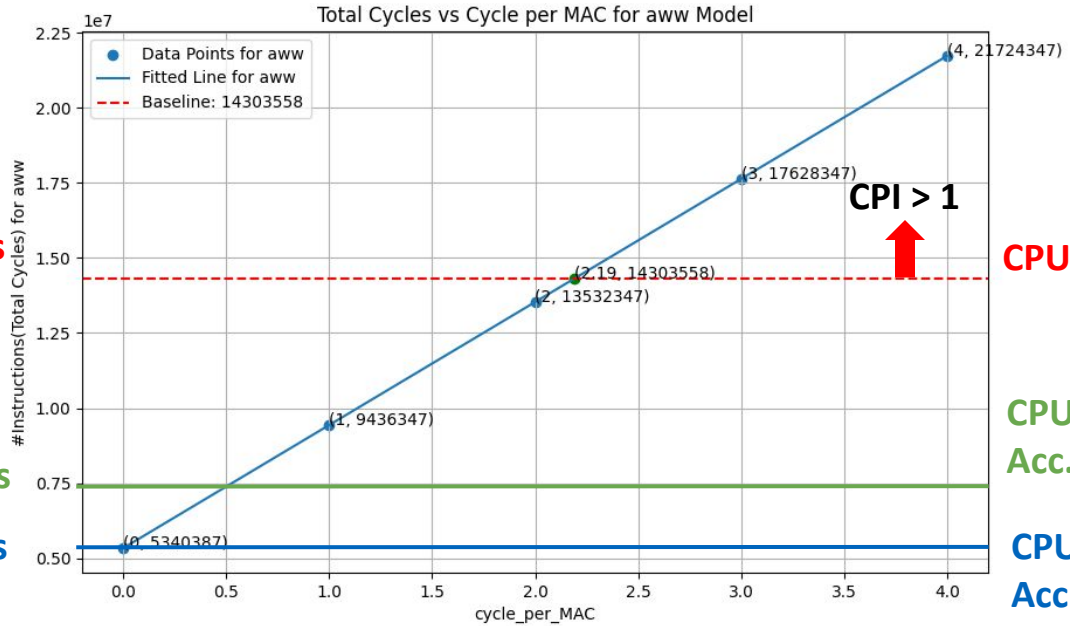
Design Exploration and Evaluation

- CPU instruction count ⟶ Simulated RISC-V microcontroller
- Simulation Time (Acc. operations are zero cycle) ⟶ measured on the x86 host

Model	CPU Instruction Count			Simulation Time
	CPU	CPU + Acc.	Saved	
resnet	5.73e07	1.44e07	**75%**	0.68s
vww	4.12e07	1.46e07	**65%**	1.33s
aww	1.43e07	5.34e06	**63%**	0.54s

Design Exploration and Evaluation

- Considering time for Accelerator operators based on #MAC operations



14.3e06 cycles

2x

7.5e06 cycles

5.3e06 cycles

CPI > 1

CPU only, CPI=1

CPU, CPI=1
Acc.: 0.5 cycle_per_mac

CPU, CPI=1
Acc.: 0 cycle_per_mac

Conclusion

- UMA provides interfaces to facilitate the process of integrating the HW accelerators
- Expanding Vanilla backend to enable the execution of quantized models
- Utilizing the MLonMCU tool to estimate the performance improvement of inference achieved by the accelerator
- Combination of virtual prototyping and TVM’s UMA for enabling the exploration and simulation of new hardware architectures

References

[1] Max Sponner, Bernd Waschneck, and Akash Kumar. 2021. Compiler toolchains for deep learning workloads on embedded platforms. arXiv preprint arXiv:2104.04576 (2021)

[2] Zhi Chen, Cody Hao Yu, Trevor Morris, Jorn Tuyls, Yi-Hsiang Lai, Jared Roesch, Elliott Delaye, Vin Sharma, and Yida Wang. 2021. Bring your own codegen to deep learning compiler. arXiv preprint arXiv:2105.03215 (2021).

[3] M. J. Klaiber, P. P. Bernardo, and C. Gerum. 2022. UMA: Universal Modular Accelerator Interface. https://github.com/apache/tvm-rfcs/blob/ main/rfcs/0060_UMA_Unified_Modular_Accelerator_Interface.md

[4] Philipp van Kempen, Rafael Stahl, Daniel Mueller-Gritschneder, and Ulf Schlichtmann. 2023. MLonMCU: TinyML Benchmarking with Fast Retargeting. arXiv preprint arXiv:2306.08951 (2023)

[5] Animesh Jain, Shoubhik Bhattacharya, Masahiro Masuda, Vin Sharma, and Yida Wang. 2020. Efficient execution of quantized deep learning models: A compiler approach. arXiv preprint arXiv:2006.10226 (2020).

[6] Daniel Mueller-Gritschneder, Keerthikumara Devarajegowda, Martin Dittrich, Wolfgang Ecker, Marc Greim, and Ulf Schlichtmann. 2017. The extendable translating instruction set simulator (ETISS) interlinked with an MDA framework for fast RISC prototyping. In Proceedings of the 28th International Symposium on Rapid System Prototyping: Shortening the Path from Specification to Prototype. 79–84.

[7] M. J. Klaiber, P. P. Bernardo, and C. Gerum. 2022. Making your Hardware Accelerator TVM-ready with UMA. https: //tvm.apache.org/docs/tutorial/uma.html