WHITEPAPER

Enabling machine learning applications with the Arm® Ethos™-U55 NPU in the Infineon ModusToolbox™ environment

Abstract

Machine learning (ML) models demand powerful compute resources for training and inferencing, so they are typically run on PCs or cloud servers, where data processing is efficient. However, embedded systems AI and ML applications are going through a transformational period, led by revolutionary developments in computer architecture and ground-breaking advances in software tools. ML applications and advanced use cases are rapidly expanding into the IOT and embedded systems space. When video and image data utilize deep learning ML models, these applications demand high processing power and large memories. These use cases make it necessary to augment the performance of processors with Neural Network Coprocessors (NPUs) such as Arm® Ethos™-U55.

Power efficiency and low cost are the key criteria for embedded ML applications. In addition to processing power augmentation, there is a need for low system power consumption and an efficient software development environment. The development environment spans tools and optimizers for model development, training, and deployment.

Infineon's PSoC™ Edge platform and its ModusToolbox™ software development environment make optimal use of platform hardware resources for CPU-intensive embedded ML inferencing.

Table of contents

1 PSoC™ Edge MCU

PSoC™ Edge MCU is a high-performance, low-power MCU family designed for compute performance, human- machine interface (HMI), machine learning (ML), enhanced sensing, real-time control, and low-power applications.

This product family comprises dual-CPU microcontrollers with neural net companion processors, DSP capability, high-performance memory expansion capability (QSPI), low-power analog subsystems with high-performance analog-to-digital conversion and low-power comparators. The devices also feature IoT connectivity, communication channels, programmable analog and digital blocks, and audio and graphics blocks.

The ModusToolbox™ development environment includes installable SDKs and libraries, industry-standard Arm® tools, RTOS support, and robust and easy-to-use ML and HMI software and tools. The functions supported include security, communications and control, and DSP capability. The multi-domain architecture enables fine-grained power optimization and dynamic frequency and voltage scaling.

The always-on domain of the PSoC™ Edge supports voice recognition, wake-on-touch, battery monitoring, and other sensing applications. These functions are provided at extremely low power.

PSoC™ Edge high-performance domain system-level architecture is shown in Figure 1.

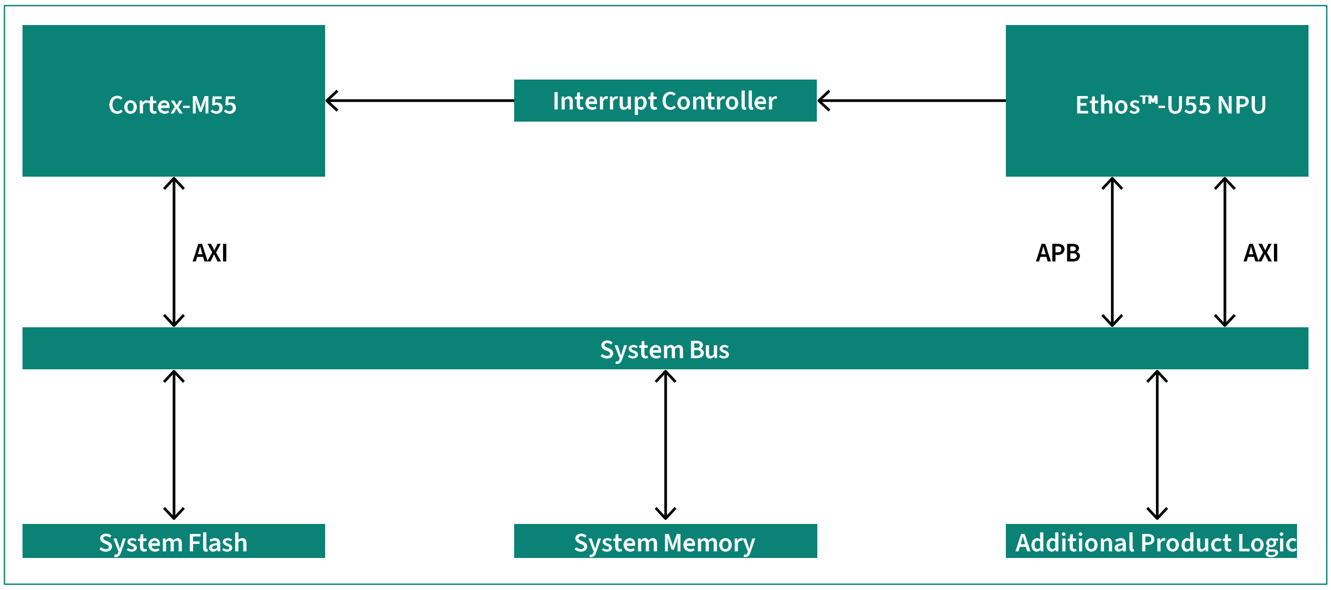


Figure 1 PSoC™ Edge high-performance domain system architecture

2 Ethos™-U55 NPU

The Ethos™-U55 is a first-generation microNPU compatible with the Cortex®-M processor. The Ethos™-U55 enables a new generation of TinyML applications which were not previously possible. It allows acceleration of neural networks in a small silicon area and with low power consumption. The Cortex®-M55 and Ethos™-U55 together deliver up to a 400x ML performance improvement compared with Cortex®-M processors alone.

While the Cortex®-M55 with its built-in helium vector processing extensions is capable of running ML models on embedded microcontroller platforms, integration with the Ethos™-U55 microNPU delivers better ML performance compared with previous generations of the Cortex®-M processor family. An Ethos™-U55 with the same software stack increases the ML performance of a Cortex®-M55 system by up to 30x.

High-level Ethos™-U55 architecture is shown in Figure 2.

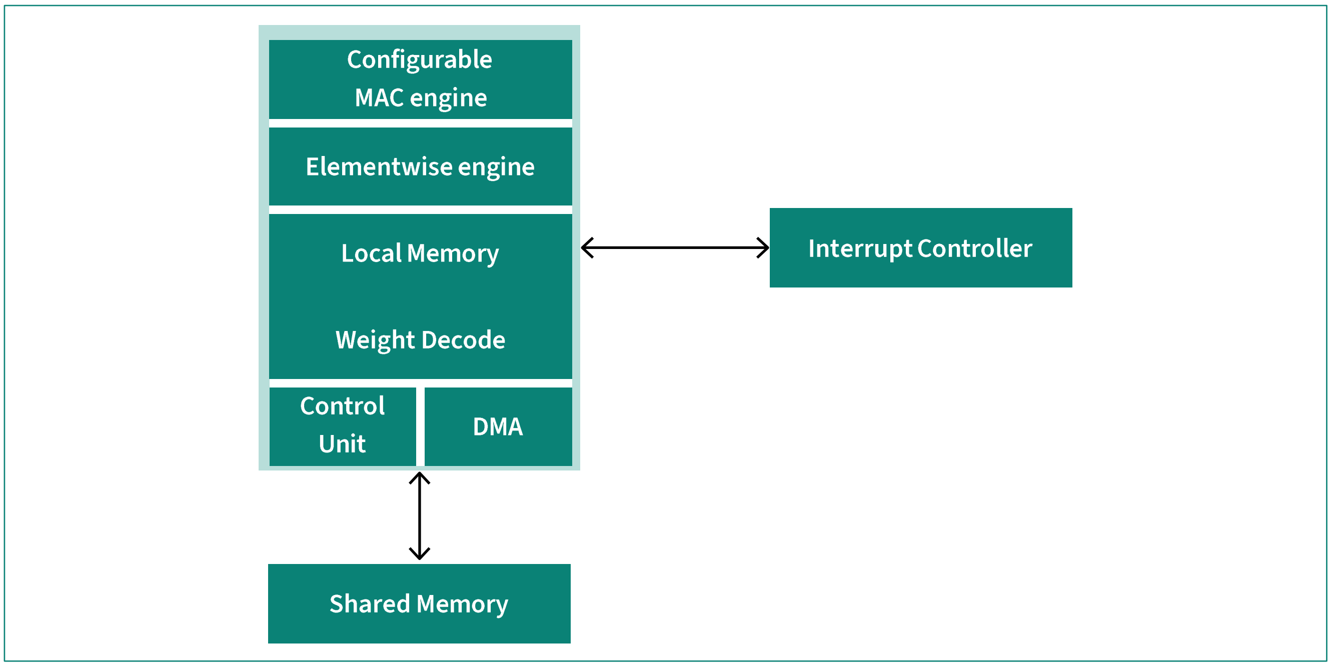


Figure 2 Ethos™-U55 system architecture

Some of the highlights of the Ethos™-U55 NPU:

- Small memory footprint and highest efficiency
- Integrated and configurable MAC engine (32-bit, 64-bit, 128-bit, 256-bit) required for convolution, depth-wise pooling, vector products, and the maximum operation required for maximum pooling. The PSoC™ Edge configures the Ethos™-U55 with a 128-bit MAC
- Weight decoder and DMA for on-the-fly weight decompression
- Offline tooling for optimizations
- Support for the most common ML network operations, including CNN and RNN, with flexibility for future ML innovations
- Central Control (CC) queues and dispatches units of work to the DMA controller, weight decoder, MAC unit, and output unit
- Integrated output unit supports activation functions including: ReLU, ReLU1, ReLU6, and Leaky ReLU, tanh, sigmoid, and Configurable Lookup Table (LUT)
- Support for high-level clock- and power-gating by exposed Q-Channel slave port

2.1 Advantages of using the Ethos™-U55 NPU

Using the Ethos™-U55 NPU along with the Cortex®-M55 as in PSoC™ Edge platform offers:

- Increased ML performance
- Lower power consumption
- Spare cycles from the Cortex®-M55 CPU for other tasks so that more complicated ML applications can be implemented

A Key Word Spotting (KWS) use case ML application was used to showcase the performance gains when the Ethos™-U55 NPU was included in the system.

2.1.1 ML performance improvement

Figure 3 shows the performance increase achieved when the Ethos™-U55 NPU is combined with an Cortex®-M55 CPU. This data was collected with a Key Word Spotting (KWS) application.

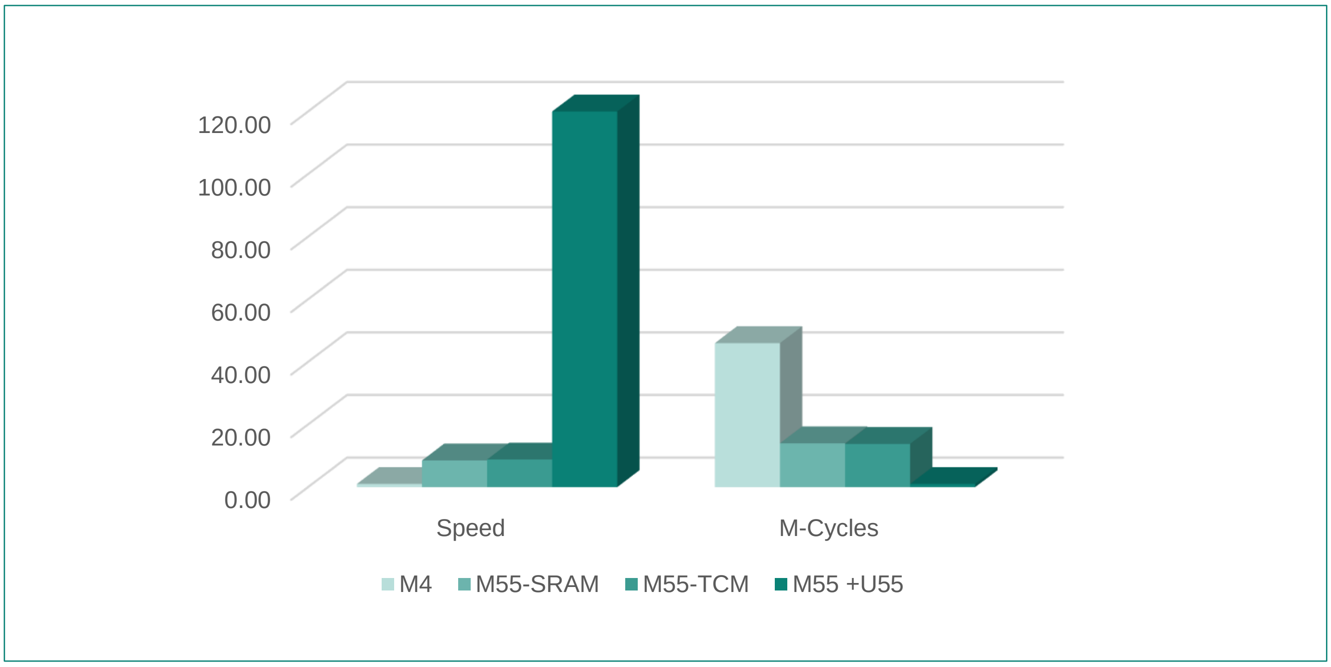


Figure 3 ML performance comparison with N55 NPU

This chart compares the performance of standalone Cortex®-M4, a Cortex®-M55 with and without TCM, and a Cortex®-M55 combined with an Ethos™-U55 NPU. It shows that the Cortex®-M55 with Helium extensions offers little better performance for ML compared with the Cortex®-M4 MCU. However, combining the Cortex®-M55 with an Ethos™-U55 NPU enhances the system performance for ML, making it possible to develop complex ML applications on embedded platforms.

2.1.2 CPU loading

With an Ethos™-U55 NPU in the system, the Cortex®-M55 CPU loading is significantly reduced and its performance improved. The resulting spare CPU capacity can be utilized for other tasks, enabling more complicated ML applications to be implemented on PSoC™ Edge platforms.

Figure 4 shows the Cortex®-M55 CPU utilization with and without an Ethos™-U55 for a Key Word Spotting (KWS) application.

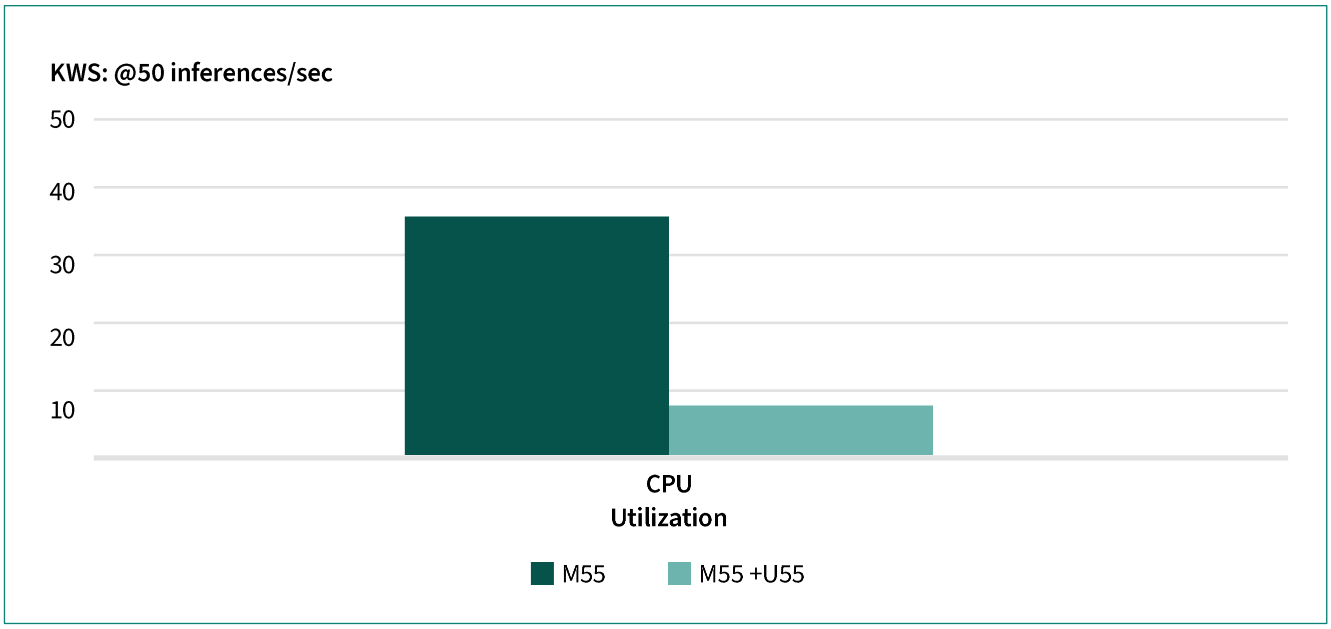


Figure 4 CPU loading comparison with Ethos™-U55 **NPU**

2.1.3 Power consumption

Besides increased performance, because of the reduced Cortex®-M55 CPU loading, greater power optimizations can be obtained when an Ethos™-U55 NPU is included for NN acceleration.

2.2 Acceleration flow with an Ethos™-U55 NPU

The process starts by training or acquiring a TensorFlow model that is to be accelerated. The model is then quantized to 8-bit integer format and converted to the standard TensorFlow Lite format.

Generic ML model optimization flow is shown in Figure 5:

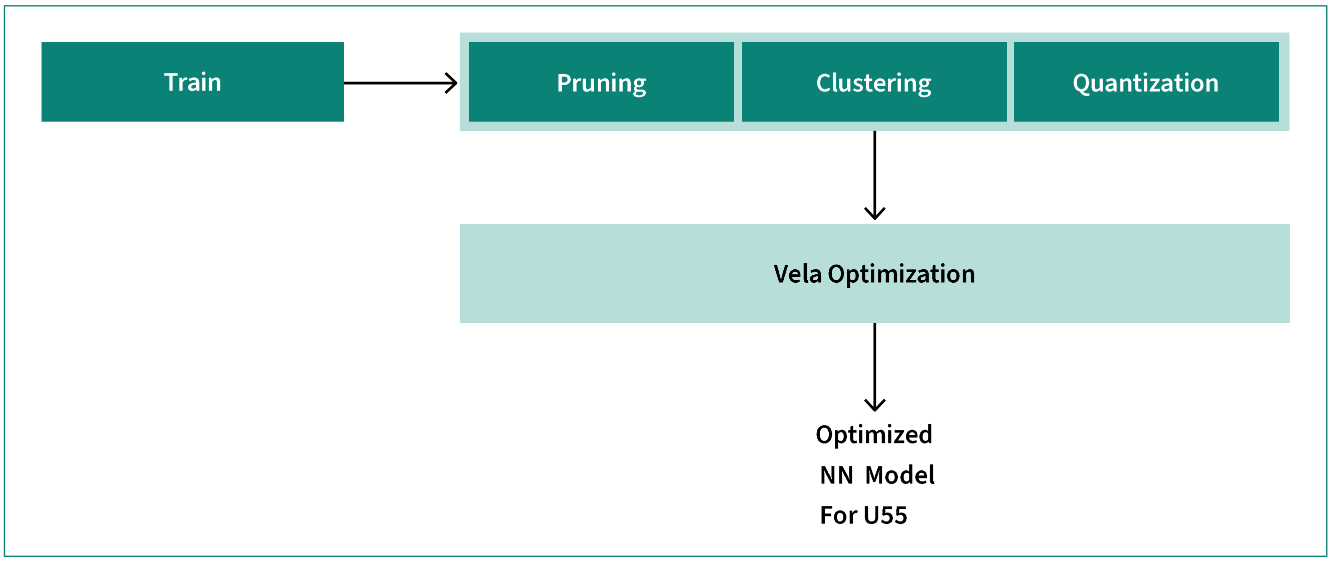


Figure 5 Generic ML model optimization flow for a Ethos™-U55

The NN optimizer tool (Vela compiler) provided by Arm®, reads this TensorFlow Lite file as an input and formats it to make it ready for deployment. TensorFlow Lite Micro (TFLM) runtime is created to execute within the constraints of embedded devices. The TensorFlow Lite flat file (flat-buffer format) is created offline on the host is deployed on the target device. The flat file contains information on which layer of the neural network executes on the Ethos-U55 versus the attached Cortex-M processor. The layers supported by the Ethos™-U55 are accelerated on it and the remaining layers execute on the attached Cortex®-M. The layers that execute on the Cortex®-M processor are accelerated through the CMSIS-NN software library if the corresponding kernel is available. Otherwise, the TensorFlow Lite Micro reference kernels are used.

3 ModusToolbox™

ModusToolbox™ software is a modern, extensible development environment supporting a wide range of Infineon microcontrollers, including PSoC™ Arm® Cortex® microcontrollers, and various Infineon connectivity options.

Provided as a collection of development tools, libraries, and embedded runtime assets, ModusToolbox™ software is architected to provide a flexible and comprehensive development experience.

Run-time software comprised of middleware, device drivers, and code examples is provided via an extensive collection of GitHub-hosted repositories.

Development tools supporting Windows, Linux, and macOS are available as an installation package from the Infineon Developer Center. These desktop applications enable the creation of new embedded applications, management of software components, configuration of device peripherals and middleware, and embedded development tools for compiling, programming, and debugging. The ModusToolbox™ development tools interface directly with the available runtime software repositories, providing easy access to the latest development resources.

Community forums, knowledge-based articles, and technical blog articles are easily accessible from the Infineon Developer Community. Additional resources to enhance the ModusToolbox™ development experience include comprehensive documentation for both development tools and runtime software, detailed training, and tutorial videos.

IDE options:
- Eclipse IDE for ModusToolbox™
- Microsoft Visual Studio Code
- IAR Embedded Workbench
- Arm® µVision®

Compiler options:
- GNU
- IAR
- Arm®

Debugging options:
- Segger J-Link
- Infineon MiniProg4
- IAR I-jet
- Arm® ULINK™

3.1 ModusToolbox™ ML

ModusToolbox™ for Machine Learning covers three primary areas for ML-enabled product makers:

- Bring-your-own model
- Train-your-own model
- Buy-your-own Model

ModusToolbox™ Machine Learning (ML) enables users to rapidly evaluate and deploy ML models on Infineon MCUs. ModusToolbox™ ML is designed to work seamlessly with the ModusToolbox™ ecosystem and can be added into existing projects to enable inferencing on low-power edge devices.

ModusToolbox™ for Machine Learning also incorporates partners that combine the best of Infineon software and partner software through the ModusToolbox™ and Friends Ecosystem. The partners provide simple-to-use AutoML training platforms, allowing developers to focus on their data and use cases instead of the heavy infrastructure and learning curve traditional ML training pipelines incur.

Partners solutions provide:

- Data collection, labelling
- Cloud-based training infrastructure
- Easy integration
- Optimized for embedded performance

3.2 ModusToolbox™ ML SW architecture

Various components and software libraries of ModusToolbox ML environment are shown in Figure 6.

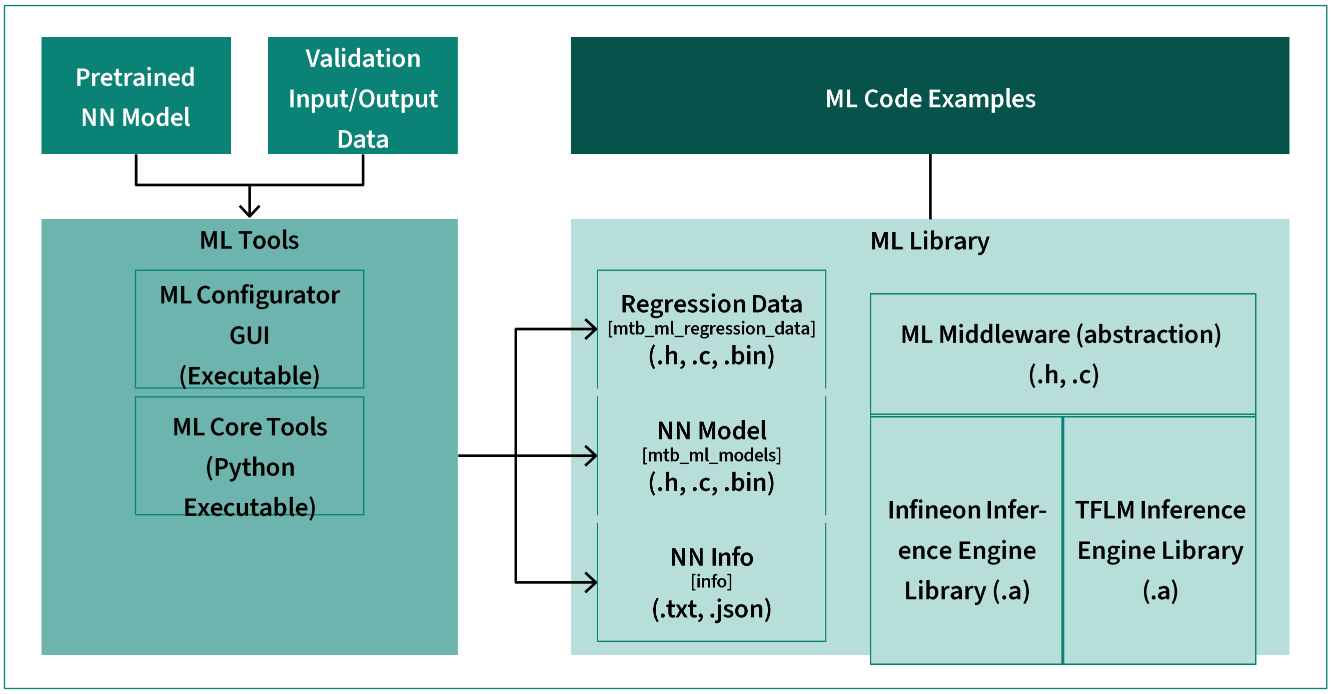
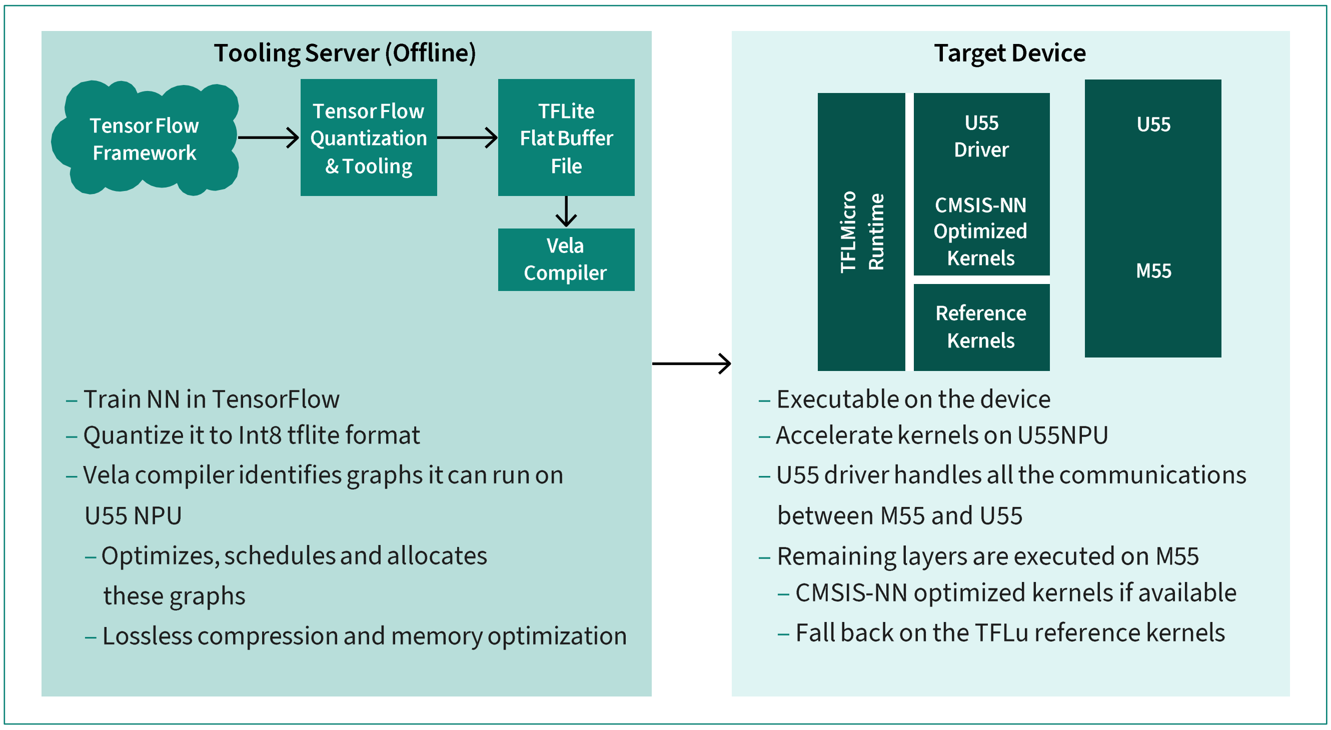

Pretrained NN Model
Validation Input/Output Data
ML Code Examples
ML Tools
ML Configurator GUI (Executable)
ML Core Tools (Python Executable)
ML Library
Regression Data [mtb_ml_regression_data] (.h, .c, .bin)
NN Model [mtb_ml_models] (.h, .c, .bin)
NN Info [info] (.txt, .json)
ML Middleware (abstraction) (.h, .c)
Infineon Inference Engine Library (.a)
TFLM Inference Engine Library (.a)

Figure 6 ModusToolbox™ ML SW components

ModusToolbox™ ML provides a set of features that make the deployment and validation of deep learning models on the PSoC™ platform seamless. These features include:
- TensorFlow Lite for Microcontrollers including both interpreter and interpreter-less inferencing
- Support for .tflite and .H5 model formats
- Core NN Kernels: MLP, GRU, Conv1d, Conv2d, LSTM
- Support NN Kernels: flatten, dropout, reshape, input layer
- Activations: relu, softmax, sigmoid, linear, tanhInput Data Quantization Level:
  - 32-bit float
  - 16/8-bit integer
- NN Weights Quantization Level:
  - 32-bit float
  - 16/8-bit integer
- Regression data evaluation
- Cycle and memory estimation
- PC-based inference engine
- Target device-based inference engine (optimized)

3.3 Runtime software flow

Software flow for model preparation, optimization, downloading to the device, and executing on the device is shown in Figure 7.



- Train NN in TensorFlow
- Quantize it to Int8 tflite format
- Vela compiler identifies graphs it can run on U55 NPU
  - Optimizes, schedules and allocates these graphs
  - Lossless compression and memory optimization

- Executable on the device
- Accelerate kernels on U55NPU
- U55 driver handles all the communications between M55 and U55
- Remaining layers are executed on M55
  - CMSIS-NN optimized kernels if available
  - Fall back on the TFLu reference kernels

Figure 7 Runtime view of software stack

The runtime software stack consists of components that interact with each other in specific ways.
- **User Application**
  The User Application runs the required functions and makes calls to the TensorFlow Lite for Microcontrollers (TFLM library) when it performs an inference of the model.
- **TensorFlow Lite for Microcontrollers**
  The TFLM framework is compiled into a C++ library that contains a copy of the optimized model along with versions of Reference and CMSIS-NN kernels. This library is then used by the User Application to perform inferences. During an inference, the model is parsed one operator at a time and the corresponding kernels are executed. The subgraphs

compiled by Vela are represented as instances of a special custom operator whose associated "kernel" simply passes the associated U55 command sequence and associated tensor data to the Ethos™-U NPU driver.

- **Reference kernels**
  Contains a set of kernels for all operators in the TFLM framework.
- **CMSIS-NN**
  The CMSIS-NN contains highly optimized and high-performance kernels that accelerate a subset of operators in the TFLM framework.
- **NPU driver**
  The Ethos™-U NPU driver handles the communication between the TFLM framework and the Ethos™-U NPU to process a custom operator. When the Ethos™-U NPU has completed its processing, it signals back to the driver, which in turn informs the TFLM library.

3.4 Cortex®-M55 CPU – Ethos™-U55 NPU communication

Communication between the Cortex®-M55 CPU and the Ethos™-U55 NPU is shown in Figure 8.

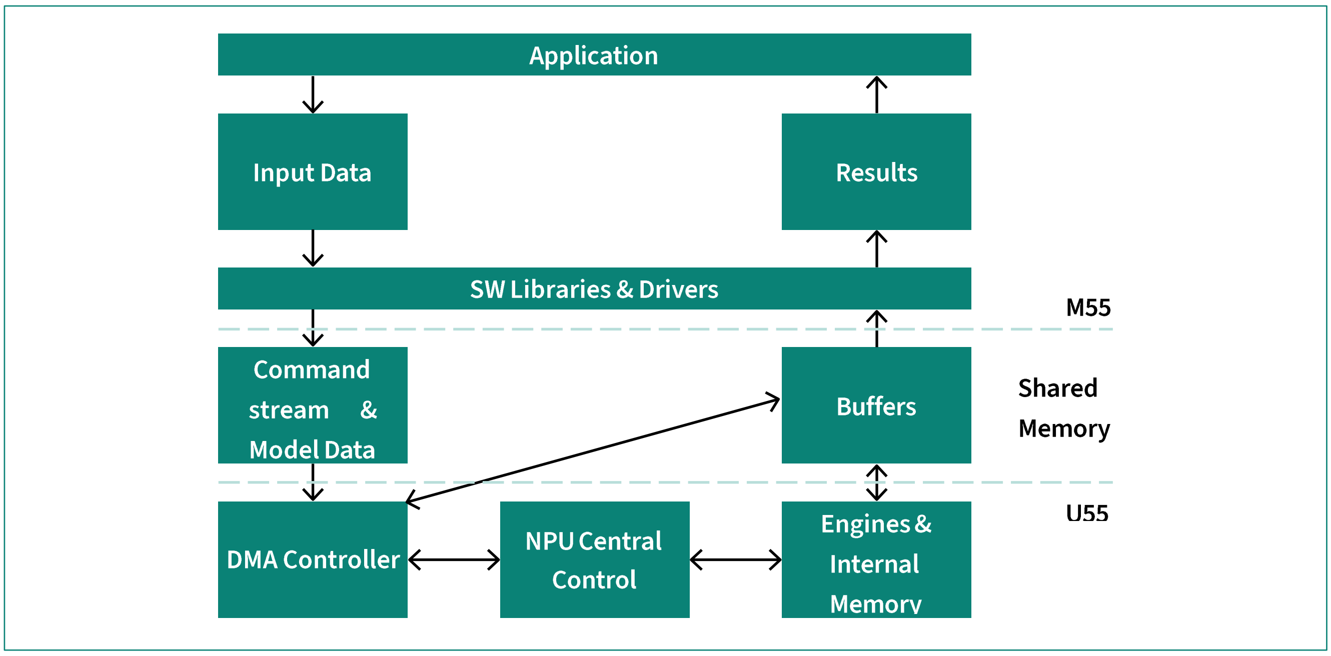


Figure 8 Cortex®-M55 – Ethos™-U55 communication

Shared memory, DMA, and interrupts provide the key interfaces needed for efficient communication between the Cortex®-M55 CPU and the Ethos™-U55 NPU.

3.5 Tools

3.5.1 ML Configurator

The ModusToolbox™ Machine Learning (ML) Configurator is used in ML applications for adapting a pre-trained learning model to an Infineon target platform. The tool accepts a pre-trained ML model and generates an embedded model (as a library), which can be used with the application code for a target device.

The ModusToolbox™ ML Configurator enables the pre-trained model of choice to be sent to the target device with a set of optimization parameters. This tool is the central asset that brings together other assets of the ML tool ecosystem, including the CoreTools, inference engine, etc.

3.5.2 ML CoreTools

The MTB-ML CoreTools are the back-end utilities with which the ML Configurator interacts to import, convert, and deploy supported embedded ML models. CoreTools includes a cross-domain validation tool that validates the performance of the converted ML models with respect to the original reference model and determines if the conversion process meets accuracy criteria. These back-end utilities are presented as platform-specific executable files.

ML CoreTools cover configuration, deployment, and cross-domain validation (CDV) of the solution. The bulk of the CoreTools functionality is related to the “deploy” run mode of the ML CoreTools application. The purpose of the deployment is to load a model with trained weights, parse and optimize it, create reference validation data, and quantize the model for different precision representations. The output is a set of files with the converted models in various precision representations of the reference model for further validation in MTB-ML.

ML CoreTools supports a command-line interface and a JSON configuration file option.

In the model deployment mode, CoreTools loads Keras H5, TFLite pre-trained model from a file, quantizes and optimizes it, then performs reference evaluation using either random or normal data.

3.5.3 Vela compiler

Vela compiler is a Python-based optimizer executed on the tooling machine with the following characteristics and functions:

- Enables NNs that were not feasible before on embedded devices
- Open source
- Reads a .tflite file and generates a modified .tflite file
- Its output includes commands for microNPU (Ethos™-U55)
- Optimizes scheduling of subgraphs
- Lossless compression of weights
- Reduces SRAM and flash footprint

This tool is used to compile a TFLite model into an optimized version that can run on the Ethos™-U55 NPU. Those parts of the model that can be accelerated by the Ethos™-U55 NPU are replaced by calls to a special custom operator that invokes the Ethos™-U55. Parts of the model that cannot be accelerated are left unchanged and will instead run on the Cortex® M series CPU using an appropriate kernel. Vela trials different compilation strategies and applies a cost function to each one.
It then chooses the optimal execution schedule for each supported operator or group of operators.

The Vela compiler also performs various memory optimizations to reduce both flash and runtime SRAM memory requirements by compressing the weights in the model.

Vela uses cascading to address runtime memory usage. Cascading reduces the maximum memory requirement by splitting the feature maps (FM) of a group of consecutively supported operators into stripes. A stripe can be either the full or partial width of the FM and can be the full or partial height of the FM. Each stripe in turn is then run through all the operators in the group.

The parts of the model that can be optimized and accelerated are grouped and converted into TensorFlow Lite custom operators. The operators are then compiled into a command stream that can be executed by the Ethos™-U55 NPU. Finally, the optimized model is written out as a TFLite model and a Performance Estimation report is generated that provides statistics, such as memory usage and inference time.

The compiler includes numerous configuration options that allow users to specify various aspects of the embedded system configuration, including the Ethos™-U55 NPU configuration, memory types, and memory sizes.

4 ML applications

The following are some of the ML applications that can benefit from the PSoC™ Edge HW and SW capabilities:

- Key word spotting
- Speaker identification
- Image classification
- Video object detection
- Gesture/human activity detection
- Wearables
- Smart appliance/voice control
- Predictive maintenance

5 Conclusion

The Infineon PSoC™ Edge MCU consists of a powerful Cortex®-M55 MCU, the Ethos™-U55 NPU, the ModusToolbox™ software development environment, and tools for Machine Learning Application development. It provides a compelling option for bringing compute-intensive, complex ML use cases and applications to the embedded/IOT environment with low power consumption. Its multi-core architecture with separate always-on domain and high-performance wake-up-on-demand capabilities offer an unparalleled platform for complex embedded/IOT ML applications.

References

[1] https://www.arm.com/products/silicon-ip-cpu/ethos/ethos-u55

[2] https://developer.arm.com/AI%20and%20ML#aq=%40navigationhierarchiescategories%3D%3D%22AI%2FML%22&numberOfResults=48

[3] https://www.arm.com/-/media/Files/pdf/ML%20on%20Arm/Arm_Ethos_U55_Product_Brief.pdf?revision=921d33c8-d166-4fb6-abf8-92ec306a0eeb&rev=921d33c8d1664fb6abf892ec306a0eeb&hash=1D3B17357D02451F7B4788F5BAF00F2F

[4] https://www.arm.com/-/media/Files/pdf/ethos/Arm_Accelerating_ML_Compute_for_Embedded_Market_white_paper1.pdf?revision=8772b5c9-89fa-420b-92a3-0d77e91c4597&rev=b5af90cebeb748b3ba54e5e71a988c7b&hash=673095A6389EE3DA9E7C1E05FA6C83A2

[5] https://www.infineon.com/cms/en/design-support/tools/sdk/modustoolbox-software/?gclid=CjwKCAjw-IWkBhBTEiwA2exyOzBwjtIFYIzxSJQrsXGqVRpzWACcslwmCTcZOMU9WhGV0T2IWgqpoxoCIUAQAvD_BwE&gclsrc=aw.ds

[6] https://www.infineon.com/cms/en/design-support/tools/sdk/modustoolbox-software/modustoolbox-machine-learning/

Published by
Infineon Technologies AG
Am Campeon 1-15, 85579 Neubiberg
Germany

Public

Date: 10 / 2023

Stay connected!

Scan QR code and explore offering
www.infineon.com