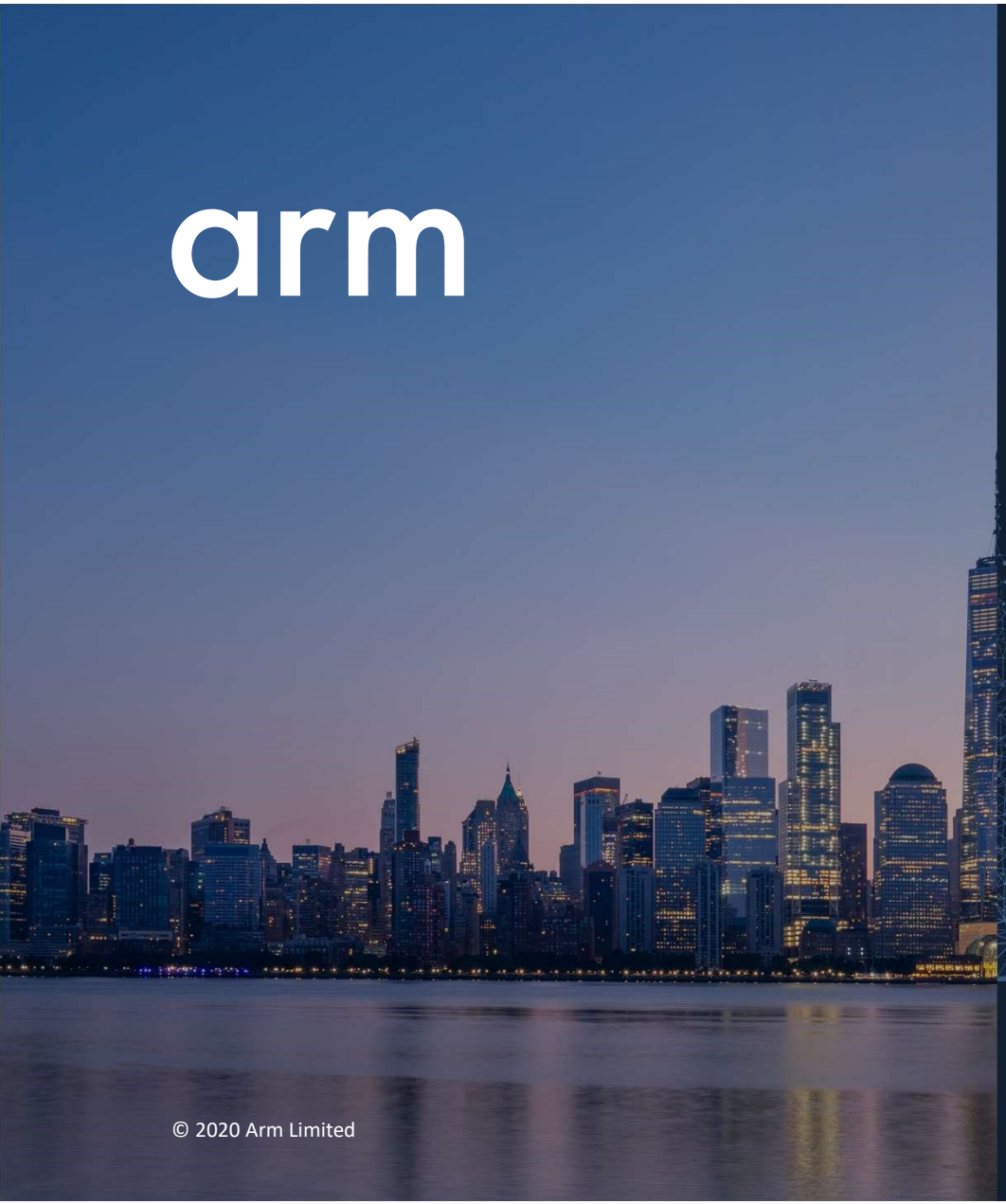


arm

Hot Chips: A Symposium on High Performance Chips

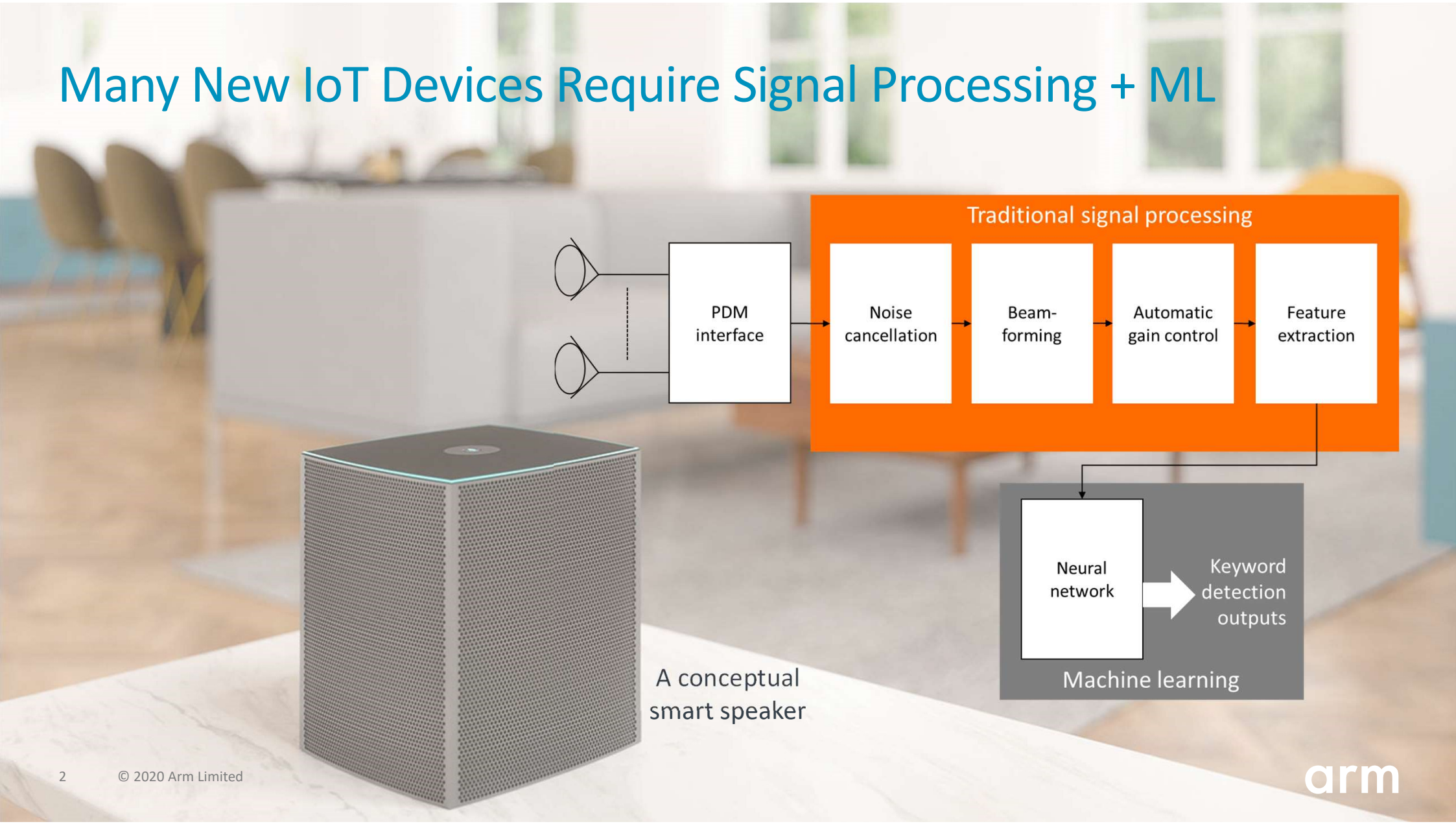
A Technical Overview of Cortex-M55 and Ethos-U55: Arm's Most Capable Processors for Endpoint AI

Allan Skillman, Distinguished Engineer, Arm

Tomas Edsö, Senior Principal Engineer, Arm

August 2020

Many New IoT Devices Require Signal Processing + ML

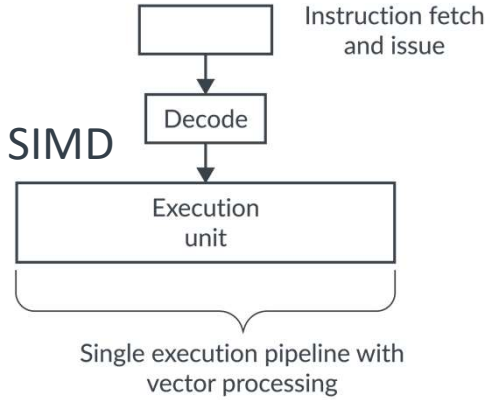


arm

How to Add More Processing Performance to Cortex-M

… without burning too much power?

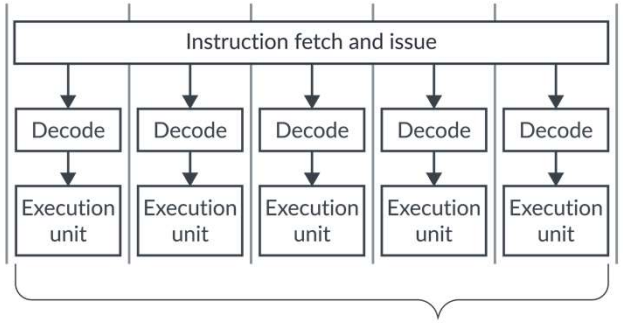
- Adding Neon (as in Cortex-A & some Cortex-R processors)?
  - Need an area efficient solution, but SIMD is nice
  - Implication to real-time (interrupt latency)



- Superscalar?
  - We already have Cortex-M7 (dual issue), but we need more
    - E.g. Limited data type support

VLIW (Very Long Instruction Word)

- VLIW as in Digital Signal Processor?
  - No, it breaks compatibility
  - Need extensive tool support

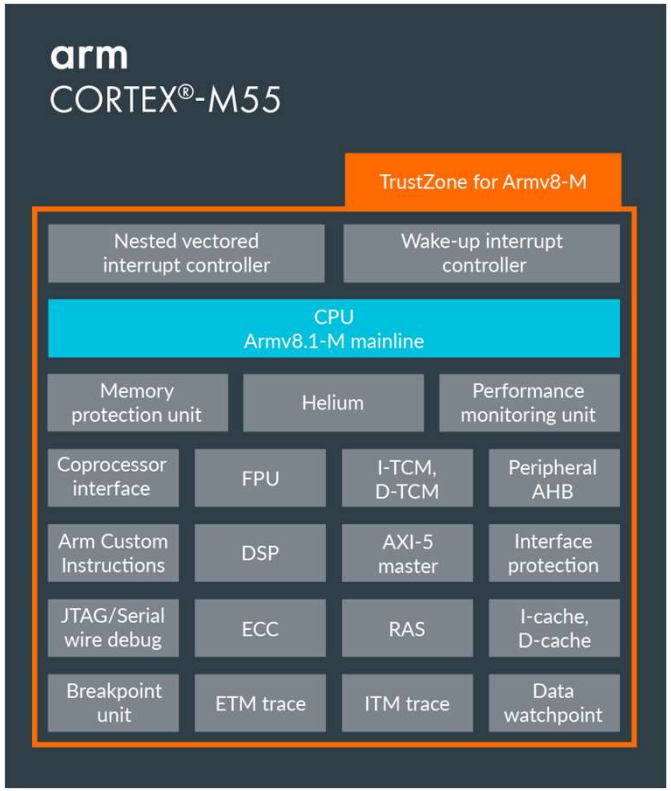


arm

Solution – Helium and Cortex-M55 Processor

Based on a new SIMD instruction set designed for Cortex-M processors

- Vector processing
  - 8 vector registers, 128-bit wide, reuse FPU registers
  - Over 150 new instructions (>130 vector instructions)
- Versatile processing capabilities
  - Vectored Integer / Fixed-point : 32-bit, 16-bit, 8-bit
  - Vectored Floating-point : Single precision, half precision arithmetic
  - Scalar Floating-point : Double, single & half precision arithmetic
- Highly configurable design
- Optimized memory system design
- TrustZone Security Extension

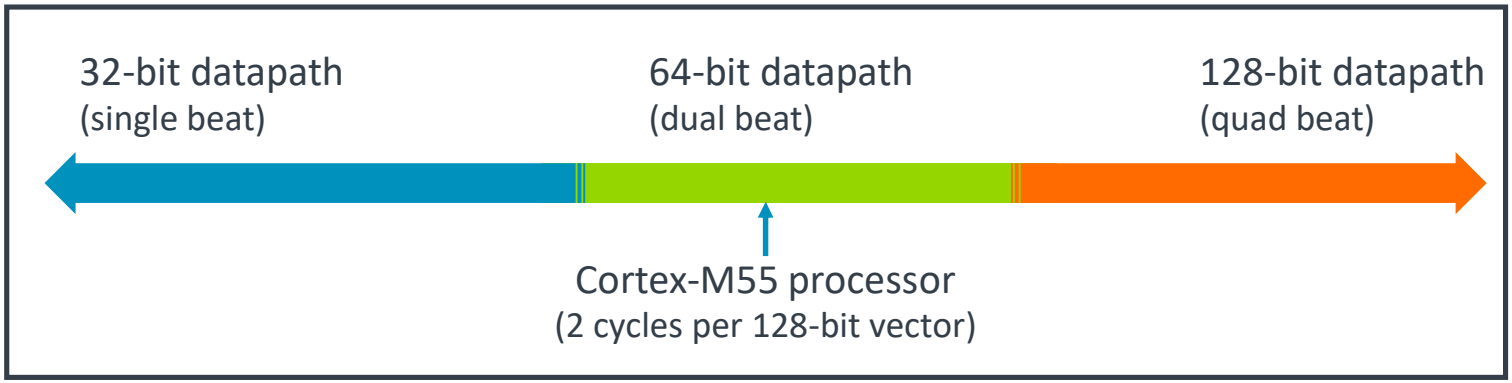


(Arm Custom Instructions available from 2021)

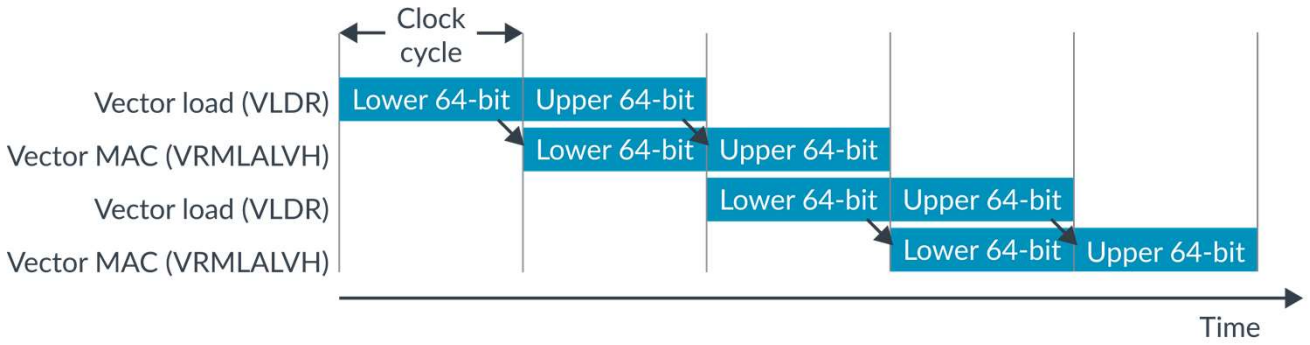
arm

Processor's Vector Pipeline and Datapath

- A balance between performance and power
  - Double ALU area, >4x performance
  - Suitable for short pipeline



- Overlapping instruction execution to enable higher processing efficiency



Cortex-M55 supports

2x Q31/int32
4x Q15/int16
8x Q7/int8
2x fp32
4x fp16

Per clock cycle

arm

Memory System Design

Requirements

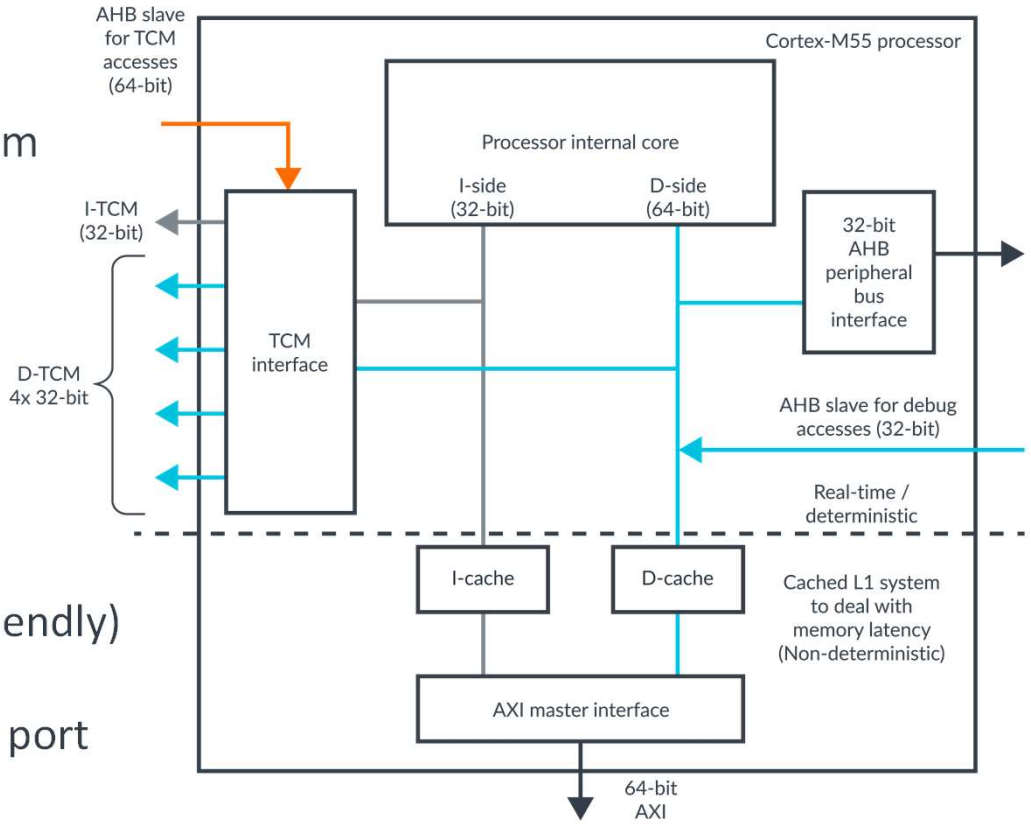
- Real-time, low latency - TCMs
- General purpose - caches

} A two-level memory system

- 64-bit data read/write bandwidth
- 32-bit instruction fetch bandwidth
- Up to two separated data R/W (scatter gather)
- 64-bit bandwidth for DMA accesses to TCM

No dedicated DSP memory ports

- TCMs are part of system memory map (C/C++ friendly)
- Up to 16MB I-TCM and 16MB D-TCM
- DMA controller can access to TCM via AHB slave port

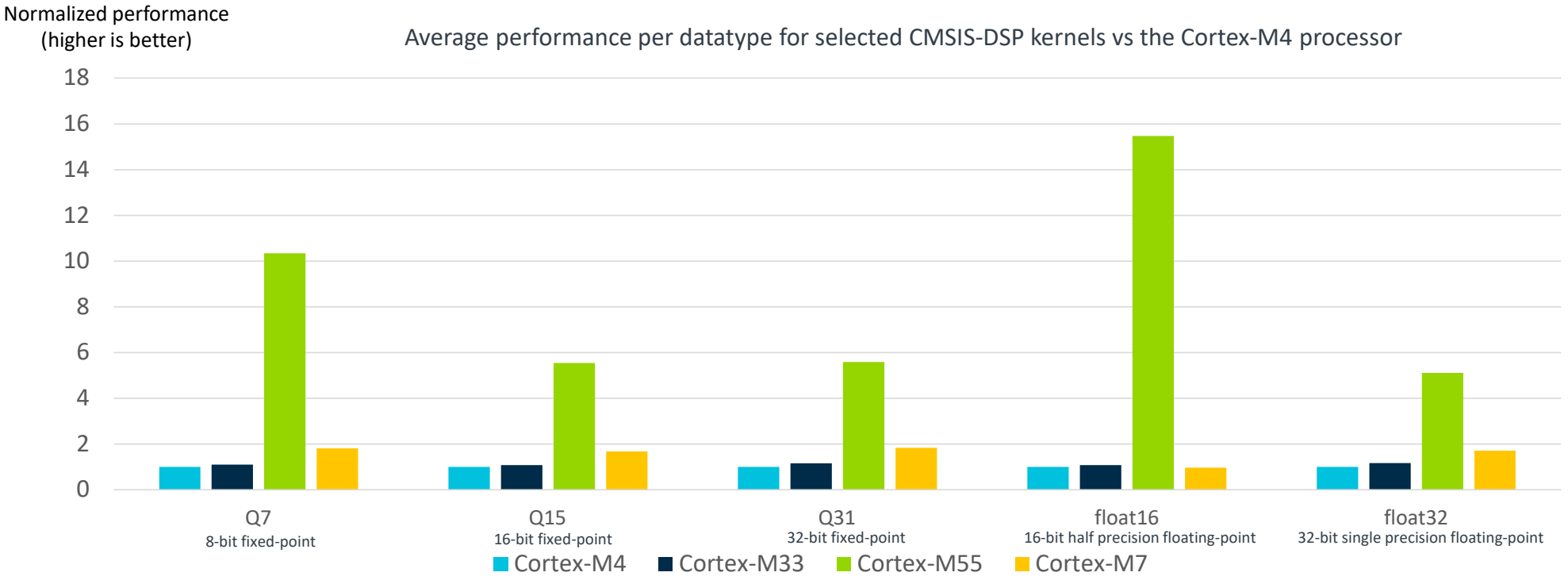


TCM: Tightly Coupled Memory

DSP-oriented Processing Support

‘DSP’ features	Cortex-M55 and Helium
Zero overhead loops	Low overhead branch extensions
Complex number processing	Complex number processing
Circular buffer	Scatter-gather memory access with instruction for circular address generation
Bit reverse addressing	Scatter-gather memory access with instruction for bit-reverse address generation
Dedicated DSP data memory interface	Multiple TCM interface to support vector memory accesses + pipeline optimization
Interleave data accesses	Interleave data accesses

Arm Cortex-M55 Processor Performance



- Average DSP kernel performance comparison across supported data types
- Relative to Cortex-M4 cycle count
- All data at ISO frequency

Cortex-M55 performance. Subject to change.

FFT: 256 point, FWD direction and bit reversal enabled // FIR: 256 blocks, 32 taps // matrix sizes: 32, vector lengths: 256 // Biquad Samples : 256 // Biquad Cascade : 4

Keyword Spotting for Low-power Voice-activated Devices

Cortex-M55 processor achieves low system power for KWS

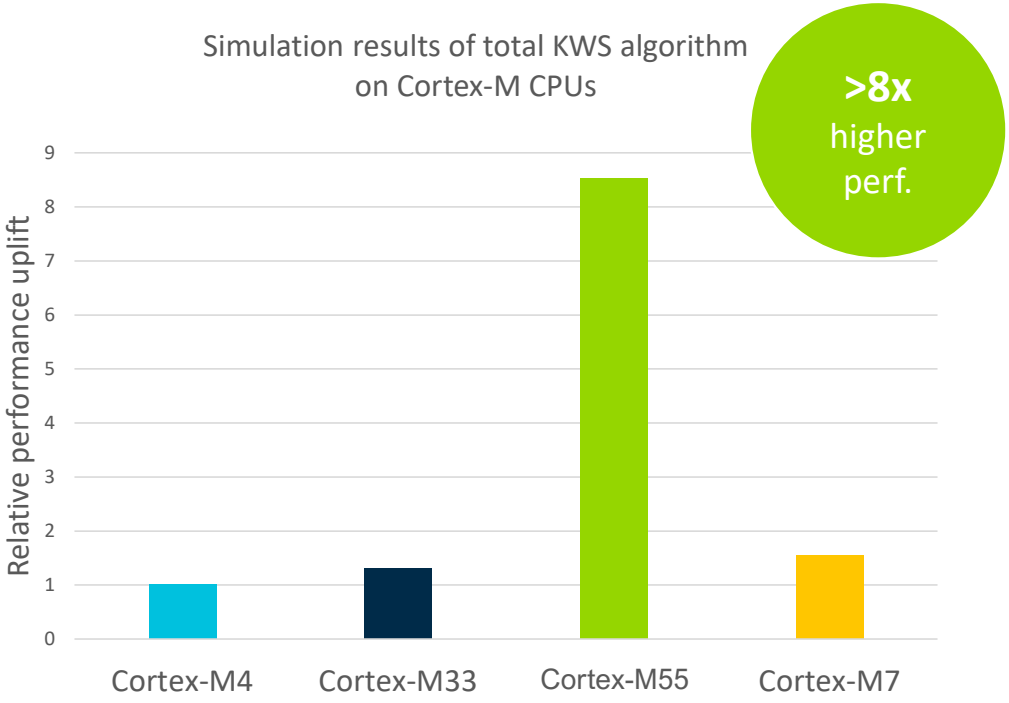
Structure of KWS algorithm

- Feature extraction: MFCC (FFT-based coefficient extraction)
  - 40ms frame size, 16kHz, 10 features
- NN based classification: 2 convolution layers and 3 fully connected layers
  - Possible NN architecture includes DNN, CNN, RNN (LSTM/GRU)
- NN size can be designed and optimized for different HW budgets
  - 8-bit weights and 8-bit activations
  - 80-500 KB memory, 6 - 80 Million Operations per second
  - Accuracy ranges from 90% - 95%

KWS algorithm publicly available now

- KWS paper including description of the network: https://arxiv.org/abs/1711.07128
- KWS Github link: https://github.com/ARM-software/ML-KWS-for-MCU/

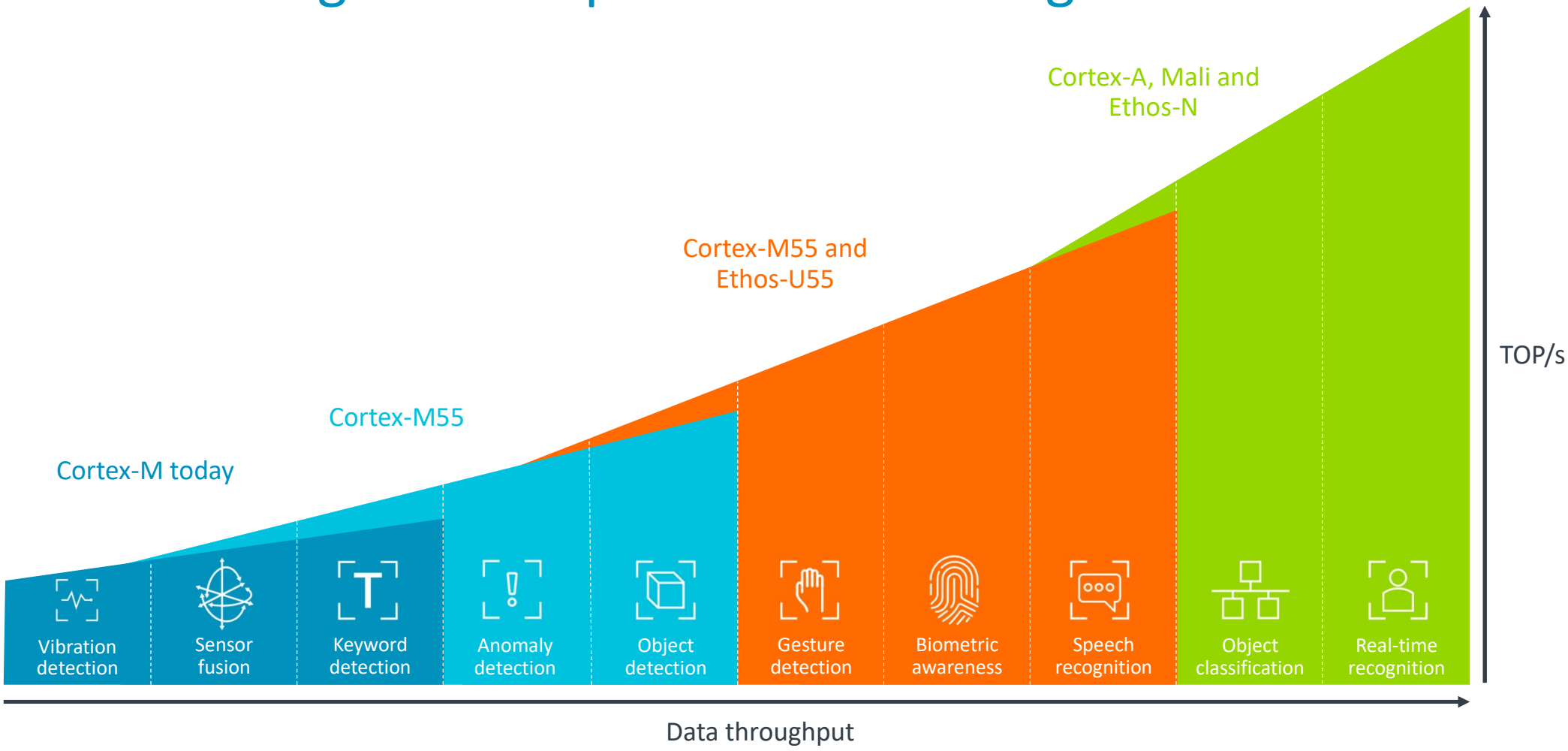
Efficient compute capabilities in next generation Cortex-M



Cortex-M55 performance results are based on RTL and C compiler in development. Subject to change.

Cortex-M4/Cortex-M7/Cortex-M33 using AC6.10

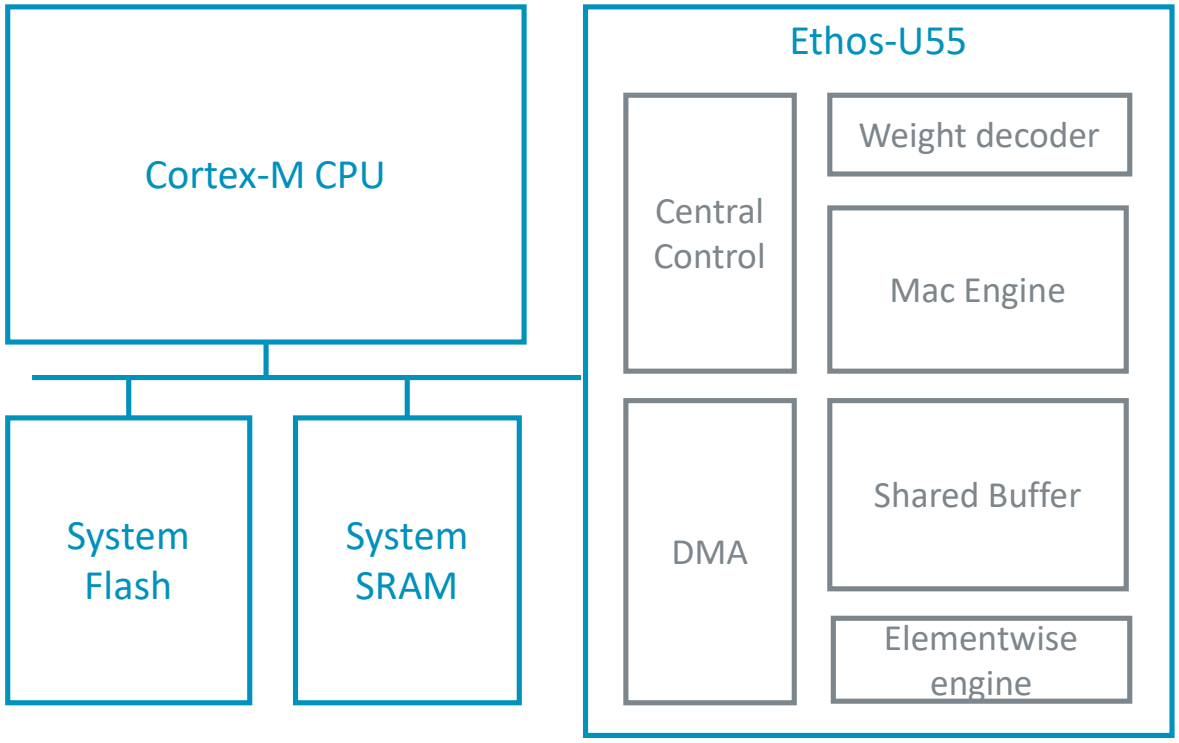
Broadest Range of ML-optimized Processing Solutions



arm

Ethos-U55 Overview

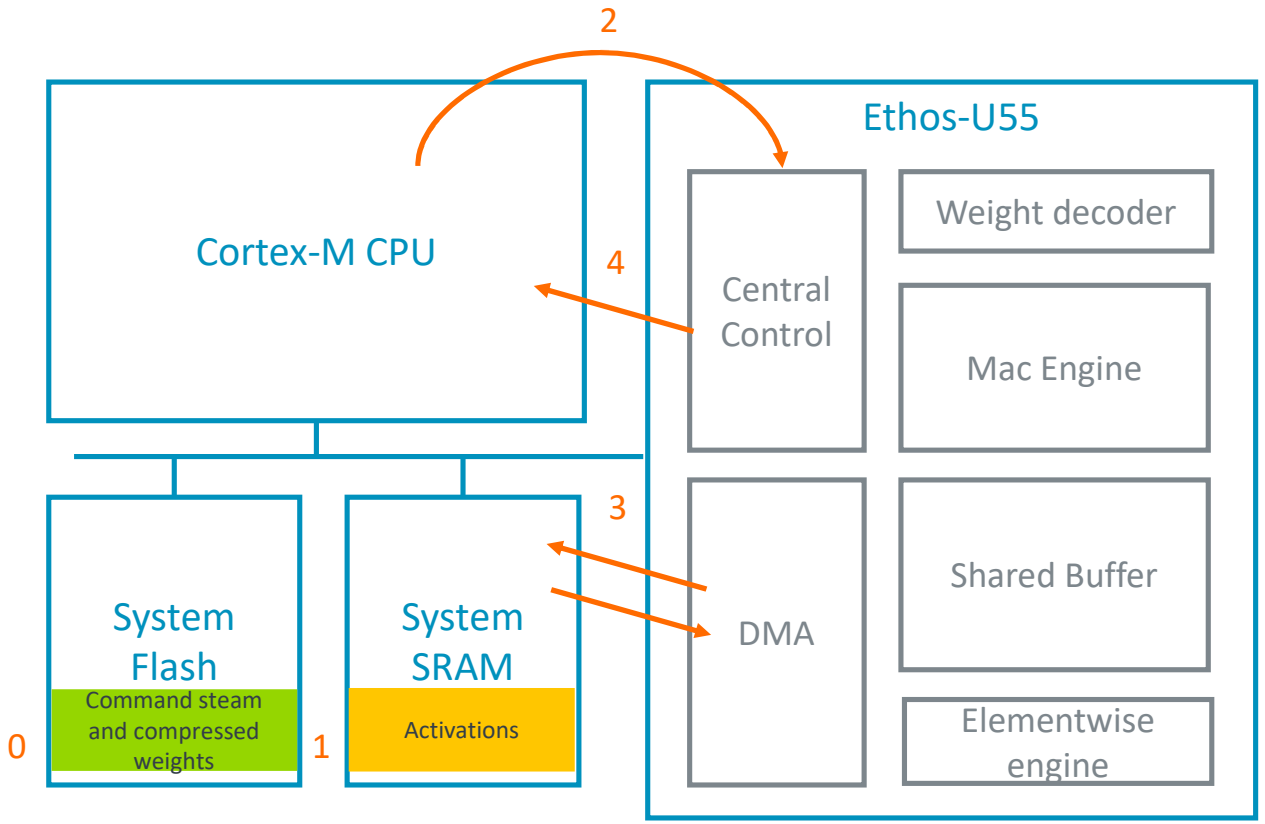
- Works alongside Cortex-M55, Cortex-M7, Cortex-M33 and Cortex-M4 processors
- Works alongside on-chip SRAM and system flash
- Accelerates CNN and RNN operators
- Efficient weight compression
- 8- or 16-bit activations
- 32, 64, 128 or 256 MAC/cc configurations



arm

Typical Ethos-U55 Data Flow

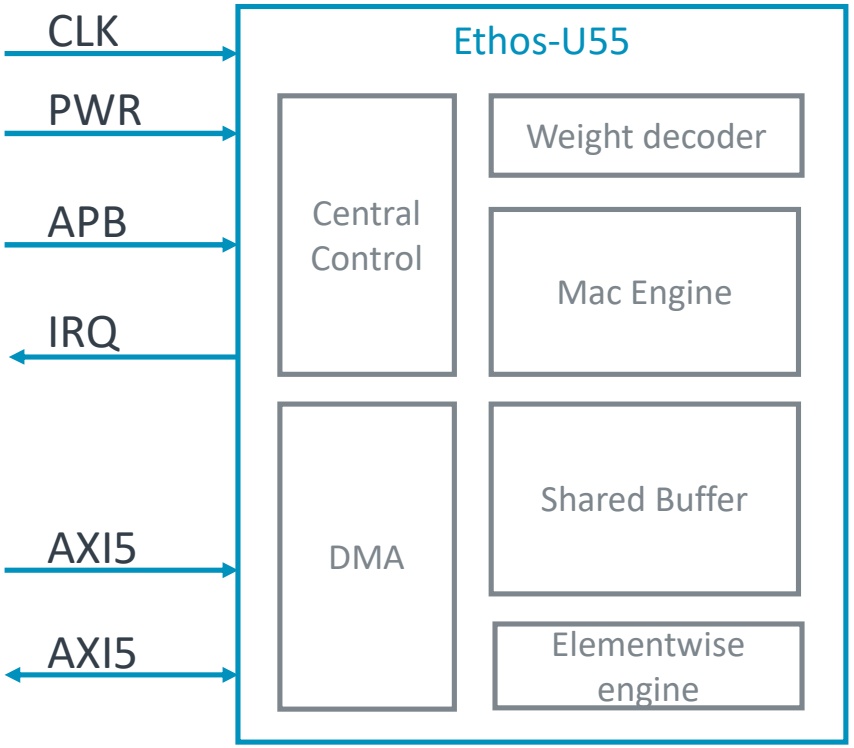
0. An offline compiled command stream with corresponding compressed weights are put into system Flash.
1. Input activations are put into system SRAM.
2. The host starts Ethos-U55 by defining all memory regions to be used, in particular the location of the command stream and input activations.
3. Ethos-U55 autonomously runs all commands, using SRAM as a scratch buffer. Results are written to a defined SRAM buffer.
4. Interrupt on completion of writing the result.



arm

Ethos-U55 Interfaces

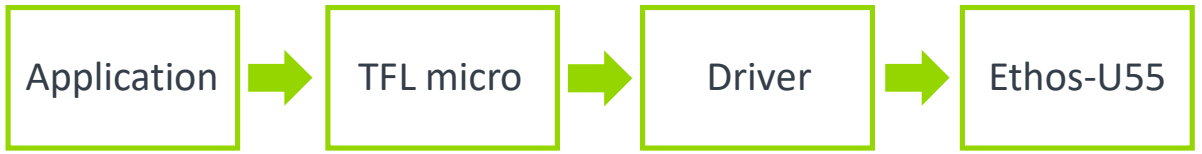
- 32-bit APB slave for registers access
- Two AXI master interfaces
  - M0: Full read+write AXI master to SRAM
  - M1: Read only AXI master to flash
- Q-channel for clock control
- Q-channel for power control
- IRQ for signaling to host



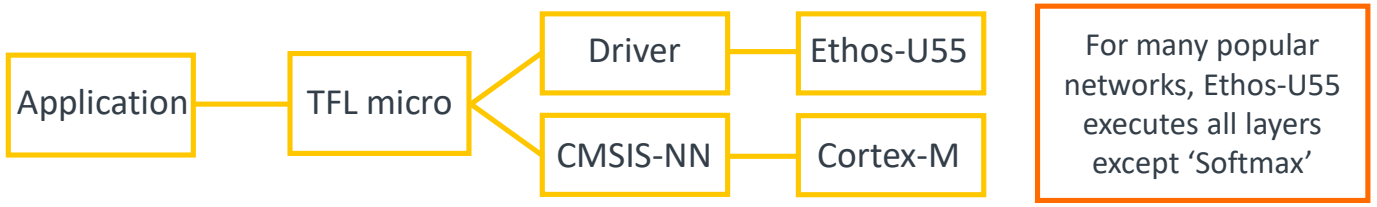
arm

Network Support in Ethos-U55

- Ethos-U55 supports a fixed set of operators and can completely execute networks that map to that operator set. For example:
  - ResNext50
  - Wav2letter

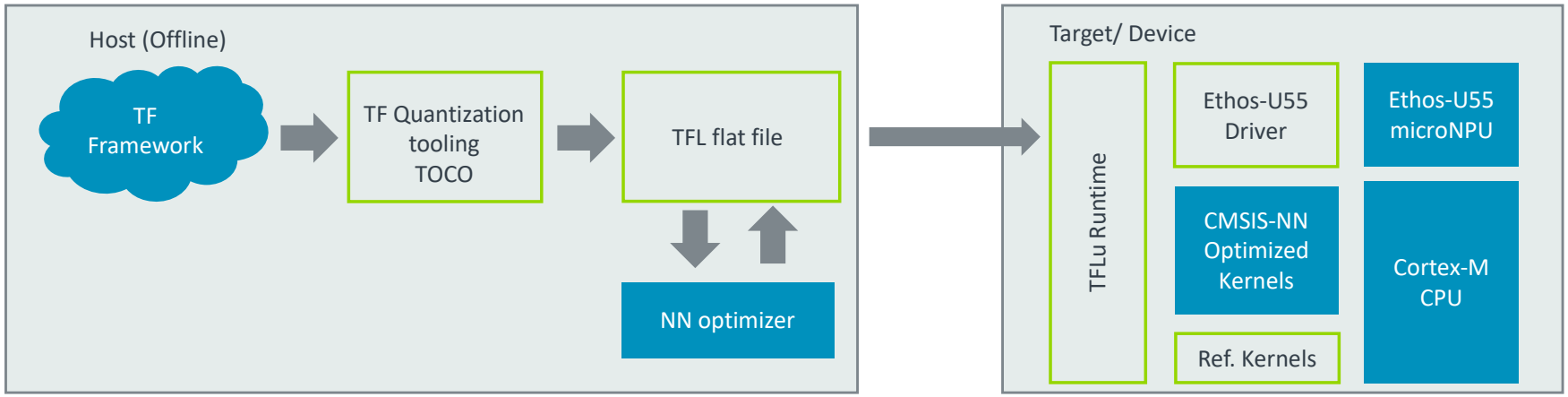


- For networks that cannot be executed on Ethos-U55 completely, the operators unsupported by Ethos-U55 fallback to the attached Cortex-M processor
  - These are accelerated through CMSIS-NN library
  - For many of the popular networks ‘Softmax’ is the only operator that falls back on the processor



arm

Ethos-U55 Optimized Software Flow



Linked platform runtime

- Train network in TensorFlow
- Quantize it to Int8 TFL flatbuffer file (.tflite file)
- NN Optimizer identifies graphs to run on Ethos-U55
  - Optimizes, schedules, and allocates these graphs
  - Lossless compression, reducing size of tflite file

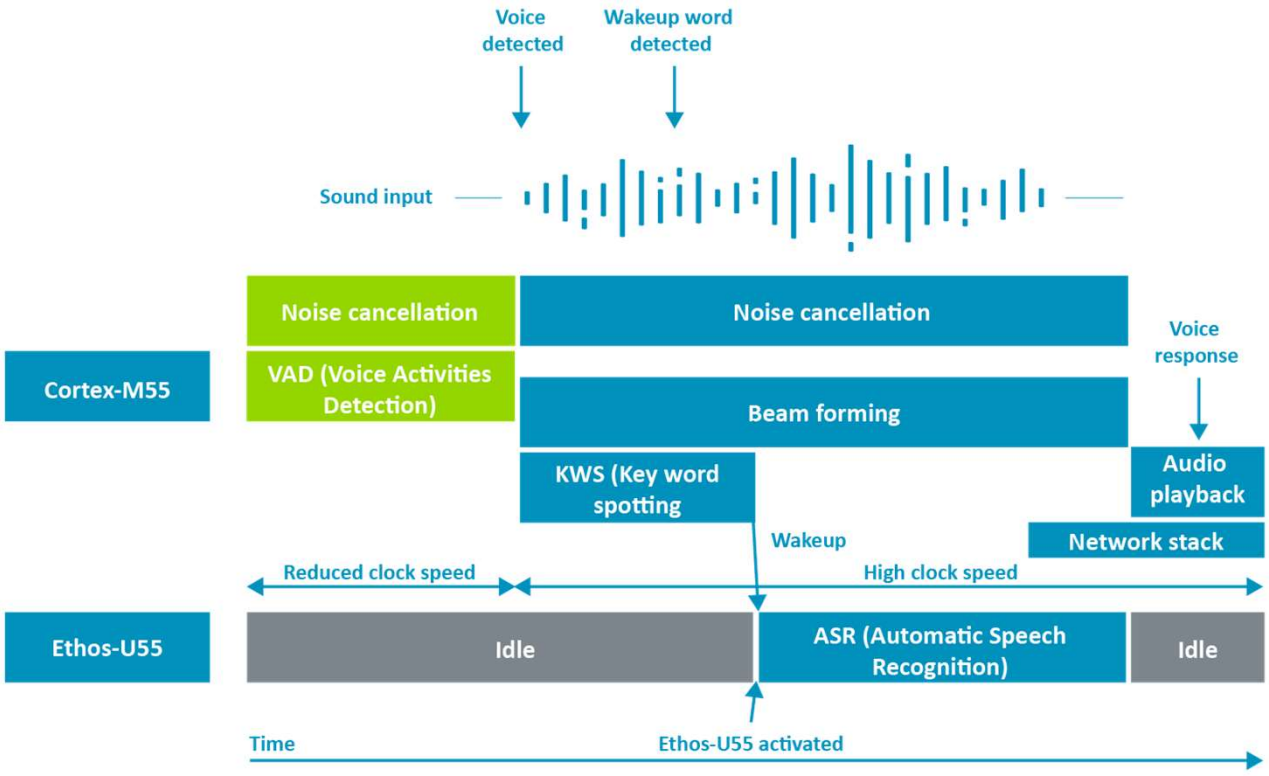
- Runtime executable file on device
- Accelerates kernels on Ethos-U55. Driver handles the communication
- The remaining layers are executed on Cortex-M
  - CMSIS-NN optimized kernels if available
  - Fallback on the TFLu reference kernels

TFL: TensorFlow Lite

arm

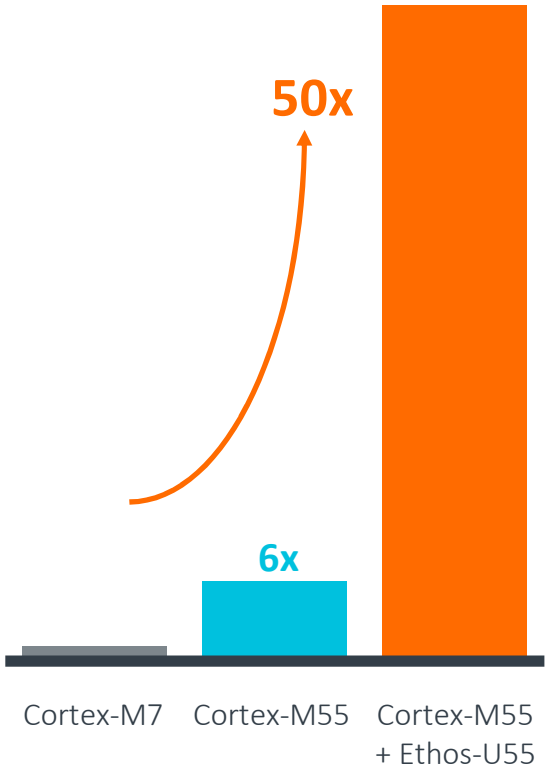
Example Use Case – Smart Speaker

- Cortex-M55 for keyword wake-up, audio processing
- Ethos-U55 for Automatic Speech Recognition (ASR)

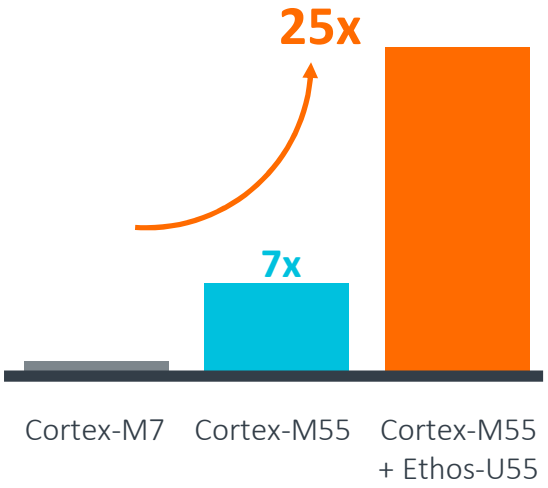


Example: Typical ML Workload for a Voice Assistant

Speed to inference



Energy efficiency



- ✓ Faster responses
- ✓ Smaller form-factors
- ✓ Improved accuracy

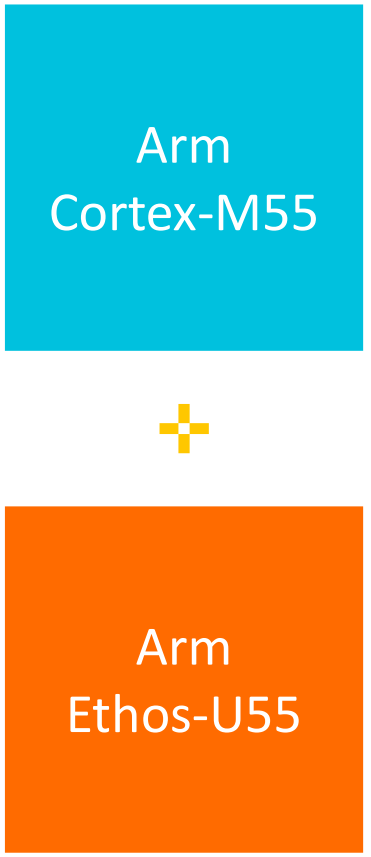
Latency and energy spent for all tasks listed combined: voice activity detection, noise cancellation, two-mic beamforming, echo cancellation, equalizing, mixing, keyword spotting, OPUS decode, and automatic speech recognition.

arm

Summary

- Can a Cortex-M Processor be built and tuned to do DSP and ML?

  Yes. Helium approach can get a good performance, while still…
  - Maintain software compatibility (able to run software for previous Cortex-M devices)
  - Satisfy the embedded market's requirements (real-time, ease-of-use, low power, security and more)
  - Uncompromised power performance
- With Cortex-M55 + Ethos-U55 processors
  - Best performance on embedded AI



arm

arm

Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
תודה

Find out More:
Cortex-M55: developer.arm.com/cortex-m55
Ethos-U55: developer.arm.com/ethos-U55