The next level of energy-efficient edge computing

tinyML Summit

Simon Craske
Lead Embedded Architect and Arm Fellow
20 March 2019
Today’s Arm Cortex-M foundations for device edge-compute
32-bit integer through to IEEE double-precision

+45 billion
Cortex-M
based chips
shipped**

Relative control code performance

Relative ML and signal processing performance

Cortex-M23, Cortex-M0+, Cortex-M0, Cortex-M1

Cortex-M3

Cortex-M4*

Cortex-M35P*

Cortex-M33*

Cortex-M7*

Integer 32-bit SIMD and floating-point

Integer long multiply

Fast integer multiply

**Based on Arm data

*Existing processors with DSP extensions
Key drivers for expanding local compute use cases that bridge the analog and digital worlds

Emerging sensing and control use cases need ultra-efficient solutions in the smallest devices

Requires blend of signal processing compute and inference machine learning compute
Ever-increasing demand for device edge compute

Bridging the microcontroller compute requirements

<table>
<thead>
<tr>
<th>Cortex-M + custom hardware or DSP</th>
<th>Cortex-M based system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Different ISA</td>
<td>Compatible, familiar ISA</td>
</tr>
<tr>
<td>Harder to program, maintain and support</td>
<td>Same programmers’ model for development</td>
</tr>
<tr>
<td>Multi-source toolchains and ecosystems</td>
<td>Single toolchain and ecosystem</td>
</tr>
<tr>
<td>Non-standard security solutions</td>
<td>Arm TrustZone</td>
</tr>
</tbody>
</table>

Lower costs
Lower complexity
Increased security

>20%* of IoT endpoint devices will have local ML capability by 2022

*Arm and industry estimates
Arm Helium™ technology: M-Profile Vector Extension (MVE) for future Cortex-M processors

Architecture objectives for Armv8.1-M

- Significantly enhance microcontroller DSP and ML performance
- Retain interrupt and latency guarantees of Arm Cortex-M
- Integrate with system-wide security provided by Arm TrustZone
- Simplify DSP and ML development within the Arm ecosystem toolchains
- Fit within the PPA requirements of microcontroller systems

All while retaining and improving the efficient execution of control code typically associated with Cortex-M
Armv8.1-M register files

Definition of 128-bit Q registers for SIMD operations

- 32-bit floating-point registers grouped into 128-bit Q registers
- Register count unmodified to retain area, security and interrupt guarantees
- Unlike Neon technology, can target general-purpose accumulators
Data-formats and operations
Supported by Helium technology

• 128-bit vectors held in the floating-point and SIMD register file containing:
  • 16x 8-bit integers
  • 8x 16-bit integers or 16-bit half-precision floating-point values
  • 4x 32-bit integers or 32-bit single-precision floating-point values

• 64-bit accumulators held in a pair of general-purpose registers
  • 64-bit integer

• 32-bit accumulators held in a single general-purpose register
  • 32-bit single-precision floating-point

• Existing Armv8-M floating-point support in floating-point and SIMD register file:
  • 16-bit half-precision (previously conversion only), 32-bit single-precision, and 64-bit double-precision floating-point
Increased throughput across a range of implementations

SIMD illusion without the deployment of 4x the multipliers and 128-bit memory

<table>
<thead>
<tr>
<th></th>
<th>t0</th>
<th>t1</th>
<th>t2</th>
<th>t3</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLDR</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>VMLA</td>
<td>B</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>VLDR</td>
<td>C</td>
<td>C</td>
<td>D</td>
<td>A</td>
</tr>
<tr>
<td>VMLA</td>
<td>D</td>
<td>D</td>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

Classic SIMD*: 4 beats per cycle
- 128-bit data-path
- 128-bit memory-path

2 beats per cycle
- 64-bit data-path
- 64-bit memory-path

1 beat per cycle
- 32-bit data-path
- 32-bit memory-path

High logic utilization at steady state
Faster looping
Deploying software controllable branch prediction on low-area implementations

Armv8.0-M:

```
loopStart:
  LDRB  R3,[R1],#1
  STRB  R3,[R0],#1
  SUBS  R2,R2,#1
  BNE   loopStart
```

Armv8.1-M:

```
loopStart:
  WLS   LR,R2,loopEnd
  LDRB  R3,[R1],#1
  STRB  R3,[R0],#1
  LE    LR,loopStart
```

Execution trace:

```
loopEnd:
  LDRB  R3,[R1],#1
  STRB  R3,[R0],#1
  LDRB  R3,[R1],#1
  STRB  R3,[R0],#1
  LDRB  R3,[R1],#1
  STRB  R3,[R0],#1
  ... 
```

Implementation permitted to cache...
... and subsequently optimize out

Significant performance uplift by removing branch delay, and integrating loop counter update
Beats across iterations and vector tail predication

Optimisation of available hardware resources and removal of need for tail code

Trace:

```
WLS
loopStart:
  VLDR
  VMLA
  LE
```

```
WLS
loopEnd:
```

Fully overlapping

```
VLD
VLDR
VMLA
VLDR
```

Set predication to bytes, and “LR” to number of elements

```
memcpy:
  PUSH   {…,LR}
  WLSTP.8 LR,R2,vectLoopEnd
```

```
vectLoopStart:
  VLDRB.8 Q0,[R1],#16
  VSTRB.8 Q0,[R0],#16
  LETP    LR,vectLoopStart
```

```
vectLoopEnd:
  POP    {…,PC}
```

If “LR” < “Bytes per vector”, store only that many bytes

Decrement “LR” by “Bytes per vector” and loop if >0

Implementation can optimize as per LE.

MVE also provides vector compare-and-predicate operation
Vector MAC instruction

- **VMLAV.<dt> Rd, Qn, Qm**

- Operation naturally produces scalar result (use scalar register file)
  - Reducing vector register pressure
- Supports signed and unsigned 8, 16 or 32-bit inputs
MVE instruction set summary

Arithmetic

- VABS, VABD, VADC, VABAV Vector absolute {Difference, Add with carry, Difference and accumulate across}
- VCADD, VCMUL, VCMLA Vector complex {Add with rotate, Multiply, Multiply accumulate}
- VCLS, VCLZ Vector count leading {Sign-bits, Zeros}
- VFMA, VFMS, VFMAS Vector fused multiply {Accumulate, Subtract, Accumulate scalar}
- VHADD, VHSUB, VHCADD Vector halving {Add, Subtract, Complex add with rotate}
- Numerous instructions providing Saturating, Rounding, Halving, Doubling, Accumulating, with Carry, and other operations

Large operations (> 128 bits)

- VADC, VSBC, VSHLC Whole vector {Add/Subtract/Left shift} {With carry}

Memory

- VSTR{B, H, W, D}, VLDR{B, H, W, D} Vector {Scatter/Gather} {Store/Load}
- VLD{2, 4}, VST{2, 4} Vector {Deinterleaving/Interleaving} {Load/Store} (Stride 2, 4)
- VLDR{B, H, W}, VSTR{B, H, W} Vector {Load/Store} register (Narrowing/Widening)

Low overhead branches

- WLS, DLS, WLSTP, DLSTP, LE, LETP, LCTP While/do loops {With tail predication} {Start/End}

Comparison and predication

- CSEL, CSINC, CSINV, CSNEG Conditional select {Increment, Invert, Negate}
- VMAX, VMAXA, VMAXAV, VMAXNM, VMAXNMA, VMAXNMV, VMAXNMAV Vector maximum {Absolute, Across, Absolute across, Integer/Floating-point} (Similar instructions for Vector minimum)
- VCMP, VCTP, VPT, VPST, VPNOT Vector compare/predicate {Tail predicate, Then, Set then, Not}

Bitwise

- VAND, VBIC, VORN, VORR, VEOR, VMVN, VREV{16, 32, 64} Vector bitwise {And, Clear, Or Not, Or, Exclusive Or, Not, Reverse}

Shift, saturate, reverse

- ASRL, LSRL, SQSRL, UQRSHLL {Arithmetic/Logical/Signed/Unsigned} shift {saturating/rounding} {Long} (Other variant for Left, Right, Saturating, Rounding, and Long are available)
- VMOVIL, VMOVIN, VQMOVIN, VQRSHL, VRSHL, VSLI Vector {Move/Saturating/Rounding/Shift} {Long/Narrow/Left/Right/Insert}
Transforming the capabilities of the smallest devices
Boosting signal processing and ML performance for millions of developers

- Signal processing
  - Signal conditioning

- Machine learning
  - Feature extraction
  - Decision algorithm

- Up to **5x** higher signal processing performance* (CFFT in int32)
- Up to **15x** higher ML performance* (matrix multiplication in int8)

*Compared to existing Armv8-M implementation
Get started today with comprehensive solutions

Virtual prototypes

arm MODELS

Architecture exploration and pre-silicon software development

Embedded toolchain

Arm DEVELOPMENT STUDIO

Keil MDK, compiler and debugger optimized for Helium and security (TrustZone and PSA)

Software libraries

Optimized libraries for best performance and code portability for DSP and ML workloads

Complete solution for Helium projects to bring intelligence to the edge

All available to early access partners today
Helium: empowering edge compute in the smallest devices

Enhanced performance
Up to 15x performance uplift to ML and up to 5x uplift to signal processing
Solution for intelligent processing locally on the device

Simple programmer’s model and one toolchain
Simplified development with unified architecture
Lower system complexity

System-wide security
Built-in security with TrustZone for Armv8-M and PSA principles
Foundation for smart, secure and connected devices

Delivering an industry-standard foundation, powering the next billions of embedded and IoT devices

Early ecosystem engagement, get started today
Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
tודה

For more information and to download the Armv8.1-M white paper, visit www.arm.com/helium.
The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

www.arm.com/company/policies/trademarks