



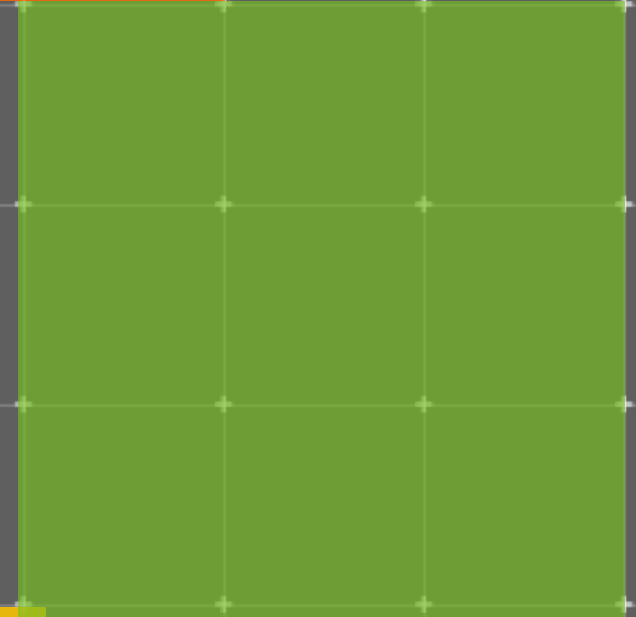
**arm** Research

# Arm's role in co-design for the next generation of HPC platforms

DiRAC Science Day – September 2018

Filippo Spiga  
Software and Large Scale Systems

# Introducing Arm & Arm in HPC



# Arm Technology Already Connects the World



## Arm is Ubiquitous

- 21 billion chips sold by partners in 2017 alone
- Mobile/Embedded/IoT/  
Automotive/Server/GPUs

## Partnership is key

- We design IP, not manufacture chips
- Partners build products for their target markets

## Arm is Ubiquitous

- One size is not always the best fit for all
- HPC is a great fit for co-design and collaboration

# The Arm Business Model

Deploy energy-efficient Arm-based technology, wherever computing happens...

Leading in wearables and the Internet of Things

~85% share of laptops, tablets, and smartphones

Driving the transformation of the network and data center to an Intelligent Flexible Cloud



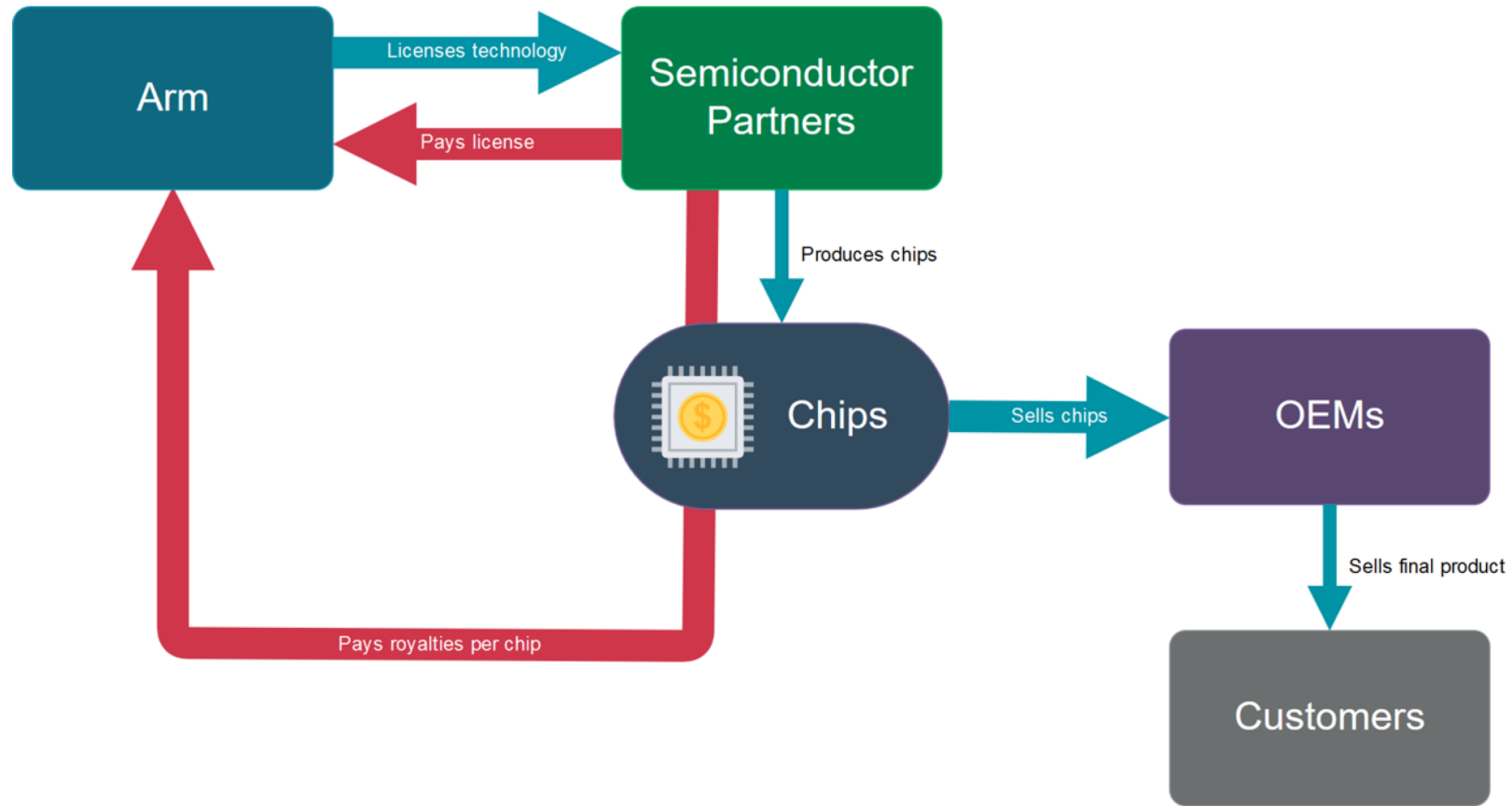
Enabling innovation and creativity with embedded intelligence

Taking mobile computing to the next four billion people

Partnering to deliver data center efficiency

# The Arm Business Model

Design of intellectual property (IP) and license to anyone who want to use into their chips.



# Arm application processors are everywhere

Cortex-A CPUs cover a wide variety of markets

- Scale efficiently to substantially higher performance
- Fit even more compute in a smaller footprint with less power



◀ Mobile and Consumer ▶



◀ Automotive ▶



◀ Servers and networking ▶



◀ IoT and Embedded ▶

# History of Arm in HPC

## 2011 Calxeda

- 32-bit ARMv7-A
- Cortex A9



## 2014 AMD Opteron A1100

- 64-bit ARMv8-A
- Cortex A57
- 4-8 Cores



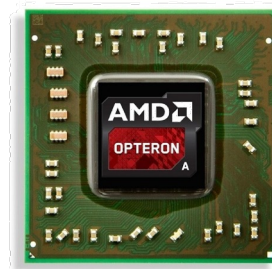
## 2017 Cavium ThunderX 2

- 64-bit ARMv8-A
- 32 Cores



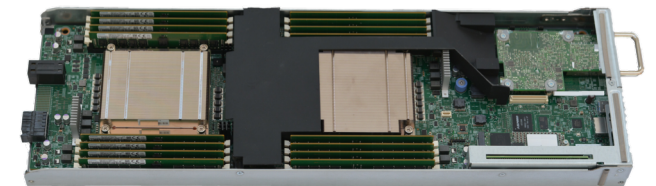
## 2011-2015 Mont-Blanc 1

- 32-bit ARMv7-A
- Cortex A15
- 'First' Arm HPC cluster



## 2015 Cavium ThunderX

- 64-bit ARMv8-A
- 48 Cores



# Deployments: GW4 “Isambard”

- New Tier-2 HPC system for GW4
- CRAY XC50 “Scout” platform
- 10,000+ Arm cores
- Based on Cavium ThunderX2 processors (top bin)
- Cray Aries interconnect
- Cray Programming environment
  - Arm tools also available



Isambard hardware, photo from ISC18

More info: [http://www.goingarm.com/slides/2018/ISC2018/Isambard\\_ISC\\_GoingArm\\_June\\_2018.pdf](http://www.goingarm.com/slides/2018/ISC2018/Isambard_ISC_GoingArm_June_2018.pdf)



# Deployments: Catalyst UK

“**Catalyst UK**” program: deployments to accelerate the growth of the Arm **HPC** ecosystem in UK. Supported by industry and EPSRC.

Each machine will have:

- 64 HPE Apollo 70 systems, each with two 32-core Cavium ThunderX2 (i.e. 4096 cores per system), 128 GByte of memory (8 TByte total)
- Mellanox EDR
- Compute & Storage all Arm-based
- Full SW stack (compilers and libraries, both open-source and commercial)

Systems will be accessible to all UK communities via access rules set by each University



**Bristol:** VASP, CASTEP, Gromacs, CP2K, Unified Model, NAMD, Oasis, NEMO, OpenIFS, CASINO, LAMMPS



**EPCC:** WRF, OpenFOAM, Two PhD candidates

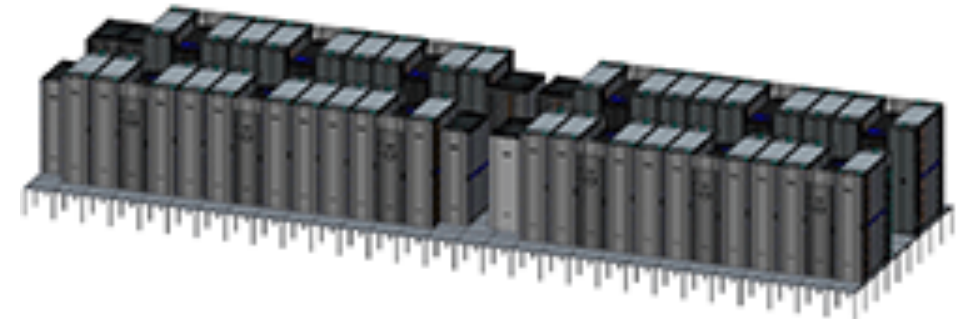


**Leicester (DiRAC):** Data-intensive apps, genomics, MOAB Torque

# Deployments: Astra (Sandia)

HPE system tailored for US NNSA application needs

- 2,592 HPE Apollo 70 compute nodes
  - Cavium Thunder-X2 ARM SoC, 28 core, 2.0 GHz
  - Memory per node: 128 GB (16 x 8 GB DR DIMMs)
- 5,184 CPUs, 145,152 cores
  - Aggregate capacity: 324TB
  - Aggregate bandwidth: 608TB/s (stream triad)
  - 2.3 PFLOPS peak
- InfiniBand EDR, Fat-Tree, Mellanox ConnectX-5
- Liquid cooled, total 1.2MW



**VANGUARD**

More info: [http://www.goingarm.com/slides/2018/ISC2018/HPEGoingArm\\_SC18.pdf](http://www.goingarm.com/slides/2018/ISC2018/HPEGoingArm_SC18.pdf)

# Post-K system & first SVE CPU

## A64FX Chip Overview



### Architecture Features

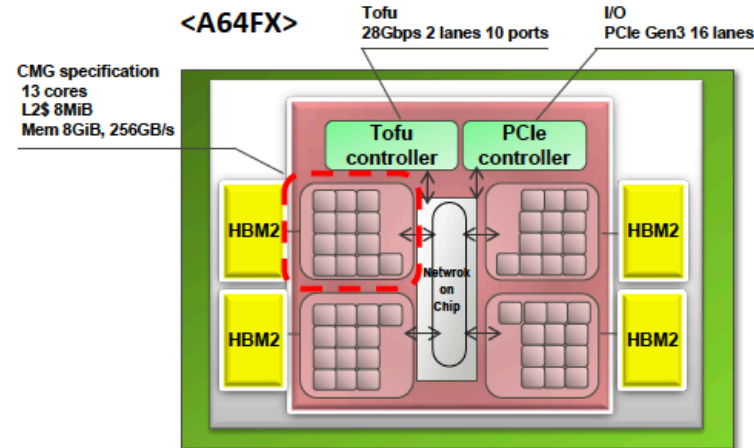
- Armv8.2-A (AArch64 only)
- SVE 512-bit wide SIMD
- 48 computing cores + 4 assistant cores\*  
\*All the cores are identical
- HBM2 32GiB
- Tofu 6D Mesh/Torus 28Gbps x 2 lanes x 10 ports
- PCIe Gen3 16 lanes

### 7nm FinFET

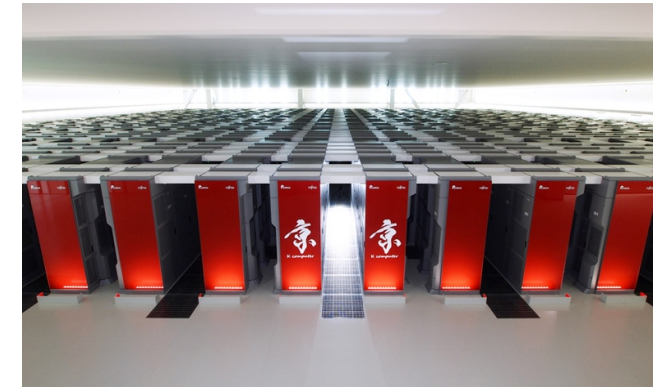
- 8,786M transistors
- 594 package signal pins

### Peak Performance (Efficiency)

- >2.7TFLOPS (>90%@DGEMM)
- Memory B/W 1024GB/s (>80%@Stream Triad)



	A64FX (Post-K)	SPARC64 Xifx (PRIMEHPC FX100)
ISA (Base)	Armv8.2-A	SPARC-V9
ISA (Extension)	SVE	HPC-ACE2
Process Node	7nm	20nm
Peak Performance	>2.7TFLOPS	1.1TFLOPS
SIMD	512-bit	256-bit
# of Cores	48+4	32+2
Memory	HBM2	HMC
Memory Peak B/W	1024GB/s	240GB/s x2 (in/out)



Announced at Hot Chips '30 in Summer 2018

Fully operational by 2020/2021

# Arm SVE Scalable Vector Extension

# Evolution of Arm SIMD architectures

## Armv6 SIMD

- $12 \times 32$ -bit integer/core register file
- Integer only  $2 \times 16$ -bit fixed-point data elements



## Armv7-A Advanced SIMD (aka Arm NEON)

- $16 \times 128$ -bit SIMD register file, supporting well-conditioned memory data layouts
- Non-IEEE single-precision floating-point and  $8/16/32$ -bit fixed-point data elements



## Armv8-A AArch64 Advanced SIMD was an evolutionary step

- $32 \times 128$ -bit SIMD register file, identical memory data layouts
- Full IEEE double-precision floating-point and  $64$ -bit fixed-point data elements



## But new markets for AArch64 (HPC) have demanded more radical features...

- Ability to vectorize irregular code and more complex data structures
- Longer vectors to extract more data-level parallelism per cycle



# Scalable Vector Extension (SVE)

## **SVE does not mandate a single, fixed vector length**

- Vector Length (VL) is hardware implementation choice of 128 to 2048 bits
- Vector Length Agnostic (VLA) programming paradigm made possible by the per-lane predication, predicate-driven loop control, vector partitioning and software-managed speculation

## **SVE is not a simple extension of AArch64 Advanced SIMD**

- A separate, optional architectural extension with a new set of instruction encodings (ARMv8.3)
- Initial focus is HPC and general-purpose server software, not media/image processing

## **SVE begins to address traditional barriers to auto-vectorization**

- Compilers often cannot vectorize due to intra-vector control and data dependencies
- Software-managed speculative vectorization allows more loops to be vectorized by a compiler

# Example: daxpy (scalar)

```
void daxpy(double *x, double *y, double a, int n)
{
    for (int i = 0; i < n; i++) {
        y[i] = a * x[i] + y[i];
    }
}
```



```
// x0 = &x[0]
// x1 = &y[0]
// x2 = &a
// x3 = &n
```

```
daxpy_:
    ldrsw        x3, [x3]
    mov         x4, #0
    ldr         d0, [x2]
    b           .latch

.lloop:
    ldr         d1, [x0, x4, lsl #3]
    ldr         d2, [x1, x4, lsl #3]
    fmadd      d2, d1, d0, d2
    str         d2, [x1, x4, lsl #3]
    add         x4, x4, #1

.latch:
    cmp         x4, x3
    b.lt       .loop
    ret
```

# daxpy (scalar)

```
daxpy_  
    ldrsw    x3, [x3]  
    mov     x4, #0  
    ldr     d0, [x2]  
    b      .latch  
.loop:  
    ldr     d1, [x0, x4, lsl #3]  
    ldr     d2, [x1, x4, lsl #3]  
    fmadd  d2, d1, d0, d2  
    str     d2, [x1, x4, lsl #3]  
    add    x4, x4, #1  
.latch:  
    cmp    x4, x3  
    b.lt  .loop  
    ret
```



# daxpy (SVE)

```
daxpy_  
    ldrsw    x3, [x3]  
    mov     x4, #0  
    whilelt p0.d, x4, x3  
    ld1rd   z0.d, p0/z, [x2]  
.loop:  
    ld1d    z1.d, p0/z, [x0, x4, lsl #3]  
    ld1d    z2.d, p0/z, [x1, x4, lsl #3]  
    fmla    z2.d, p0/m, z1.d, z0.d  
    st1d    z2.d, p0, [x1, x4, lsl #3]  
    incd    x4  
.latch:  
    whilelt p0.d, x4, x3  
    b.first .loop  
    ret
```

Q1: How do we handle the non-multiples of VL?

Q2: What happens at different vector lengths?



# SVE Programming

## Assembly

(not suggested)

Full ISA Specification: [The Scalable Vector Extension for ARMv8-A](#)

Lots of worked examples, see [“A sneak peek into SVE and VLA programming”](#)

## Intrinsics

(only if needed)

[Arm C Language Extensions for SVE](#)

[Arm Scalable Vector Extensions and application to Machine Learning](#)

## Compiler

Auto-vectorization via GCC, Arm Compiler for HPC, Cray, Fujitsu

Hints to the compiler via OpenMP: `#pragma omp parallel for simd`

Best practice in writing parallel code

# arm COMPILER

Commercial C/C++/Fortran compiler with best-in-class performance



Compilers tuned for Scientific Computing and HPC



Latest features and performance optimizations



Commercially supported by Arm

## Tuned for Scientific Computing, HPC and Enterprise workloads

- Processor-specific optimizations for various server-class Arm-based platforms
- Optimal shared-memory parallelism using latest Arm-optimized OpenMP runtime

## Linux user-space compiler with latest features

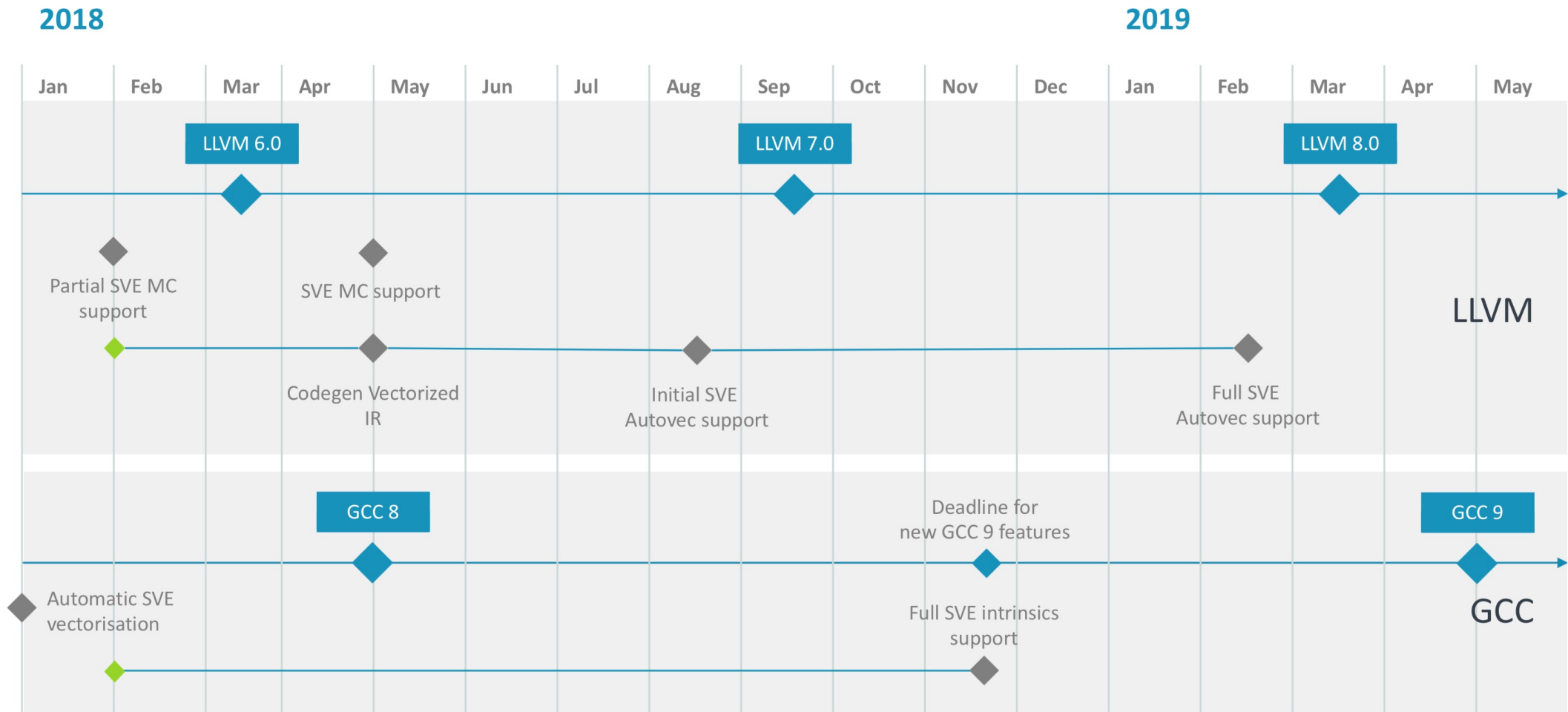
- C++ 14 and Fortran 2003 language support with OpenMP 4.5\*
- Support for Armv8-A and SVE architecture extension
- Based on LLVM and Flang, leading open-source compiler projects

## Commercially supported by Arm

- Available for a wide range of Arm-based platforms running leading Linux distributions – RedHat, SUSE and Ubuntu

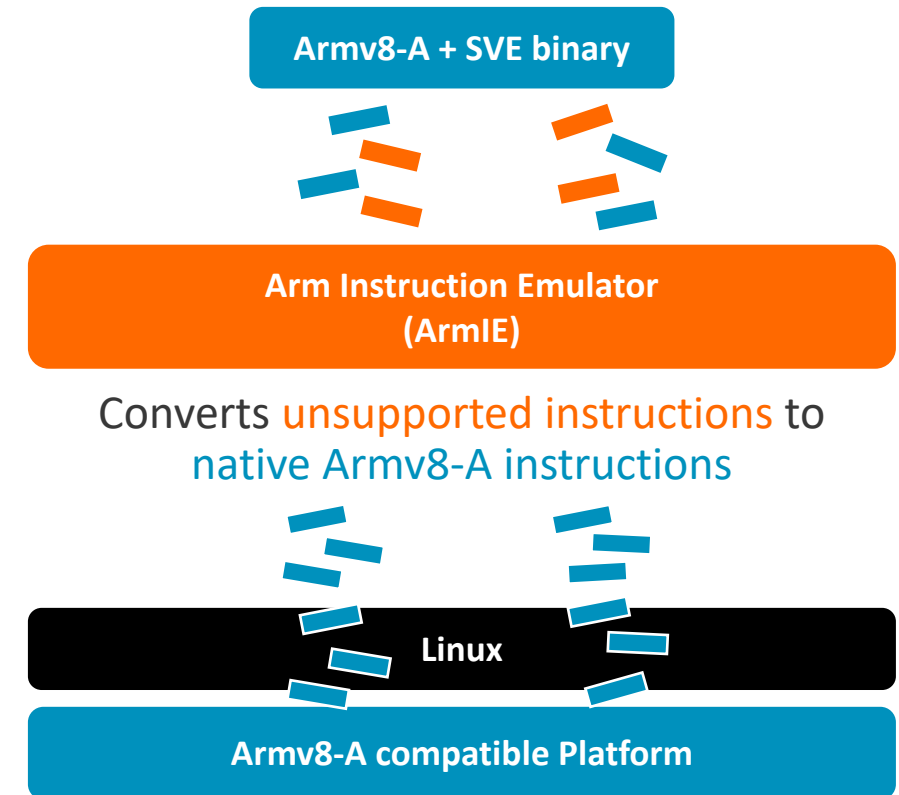
# LLVM and GCC upstreaming roadmap

Develop your user-space applications for future hardware today

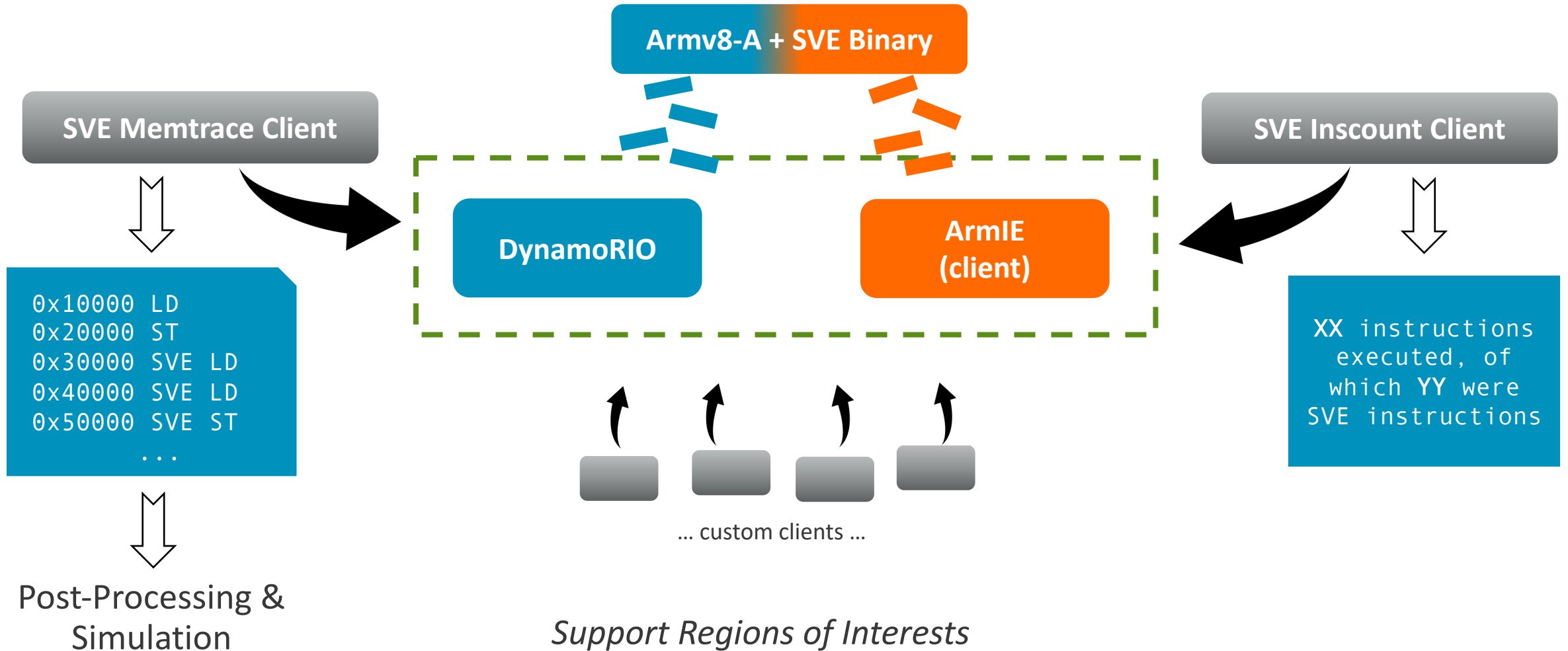


# The Arm Instruction Emulator (ArmIE)

- ArmIE enables SVE studies in preparation for upcoming silicon
  - Emulates SVE instructions on existing Armv8 hardware
  - It enables vector length considerations for future micro-architectures and application optimisations
  - As an emulator, it allows for faster application runs and integrates with DBI tools
- Faster than architecture simulators (gem5)
  - Enables the study of larger input sizes in a fraction of time
  - Simulators can complement ArmIE with timing results
- Ideal for instruction/memory tracing and dynamic binary instrumentation (DBI)
  - Post-processing traces extend application analysis (cache simulator, etc.)
  - Helps with SVE-supported application development/optimisation



# Instrumenting Aarch64 and SVE



# Example: SVE Instruction Count Client

Simple SVE loop code example:

```
#define N 42
int a[N], b[N], c[N];

int main(void) {
    for(int i=0; i<N; ++i){
        a[i] = b[i] + b[c[i]];
    }
}
```

Compiling with Arm HPC compiler 18.4

```
$ armclang -O3 -march=armv8-a+sve sve_example.c -o sve_example
```

# Example: SVE Instruction Count Client

- Run SVE binary without emulation

```
./sve_example
```

```
Illegal instruction
```

- Using ArmIE, full inscount of SVE example (512-bit vectors):

```
$ armie -msve-vector-bits=512 -i libsve_inscount.so -- ./sve_example
```

```
83971 instructions executed of which 22 were SVE instructions
```

- Exclude shared libraries by adding “-only\_from\_app” to inscount:

```
127 instructions executed of which 22 were SVE instructions
```

# Exploring SVE for scientific applications

Objective: understand if apps can benefit from SVE, assess quality and readiness of tools

- Various Arm-based SoC (Huawei Taishan)
- Several applications of interest: QE, KKRnano, GRID, BQCD
- Results on MiniKKR show Arm-based SoC (no SVE) similar performance versus x86
- Estimate performance using instruction/branch counting (*dynamic*) and critical path analysis (*static*)

**“Early Experience with ARM SVE”**, presented at SC’17 Arm SVE User Meeting by D. Pleiter (JSC)

[http://www.goingarm.com/slides/2017/SVE\\_SC17/GoingArm\\_SVE\\_SC17\\_Arm\\_Dirk.pdf](http://www.goingarm.com/slides/2017/SVE_SC17/GoingArm_SVE_SC17_Arm_Dirk.pdf)

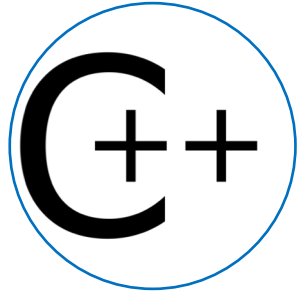
**“Exploring SVE for scientific applications”**, presented at HiPEAC’18 by S. Nassyr (JSC)

[http://www.goingarm.com/slides/2018/HiPEAC2018/julich\\_hipeac\\_goingarm\\_2018.pdf](http://www.goingarm.com/slides/2018/HiPEAC2018/julich_hipeac_goingarm_2018.pdf)



# Arm Alinea Studio

A quick glance at what is in Arm Alinea Studio (latest 18.3)



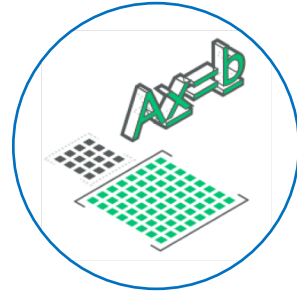
## C/C++ Compiler

- C++ 14 support
- OpenMP 4.5 without offloading
- SVE ready



## Fortran Compiler

- Fortran 2003 support
- Partial Fortran 2008 support
- OpenMP 3.1
- SVE ready



## Performance Libraries

- Optimized math libraries
- BLAS, LAPACK and FFT
- Threaded parallelism with OpenMP



## Forge (DDT and MAP)

- Profile, Tune and Debug
- Scalable debugging with DDT
- Parallel Profiling with MAP



## Performance Reports

- Analyze your application
- Memory, MPI, Threads, I/O, CPU metrics

Tuned by Arm for a wide-range of server-class Arm-based platforms

# arm PERFORMANCE LIBRARIES

Optimized BLAS, LAPACK and FFT



Commercially supported  
by Arm



Best serial and parallel  
performance



Validated with  
NAG test suite

## Commercial 64-bit Armv8-A math libraries

- Commonly used low-level math routines - BLAS, LAPACK and FFT
- FFTW compatible interface for FFT routines
- Batch BLAS support

## Best serial and parallel performance

- Generic Armv8-A optimizations by Arm
- Tuning for specific platforms like Cavium ThunderX2 in collaboration with silicon vendors

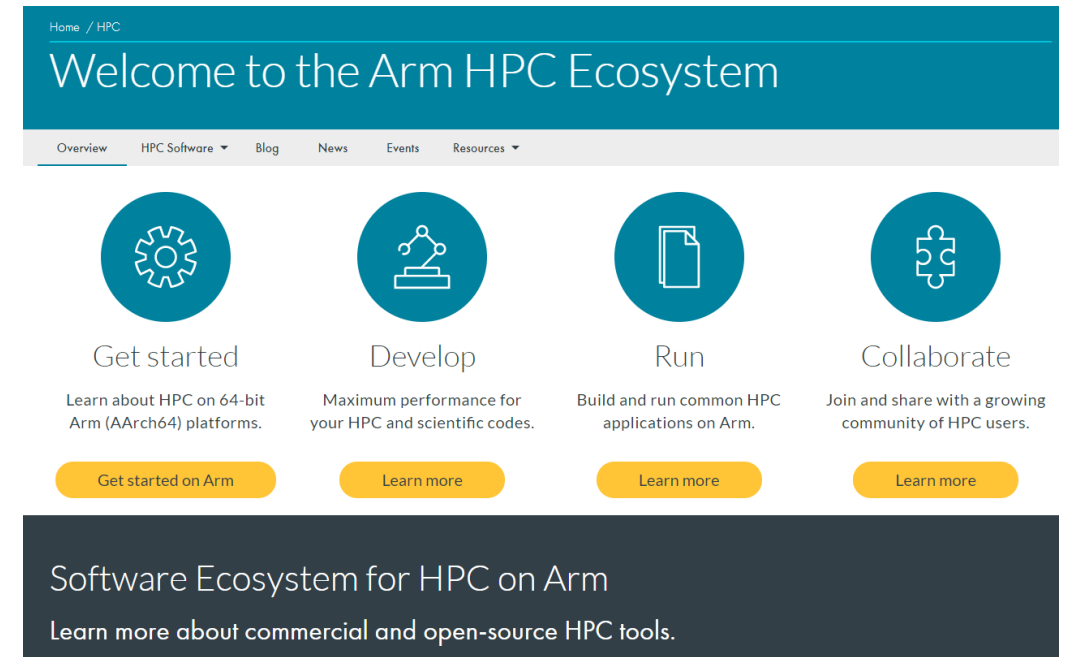
## Validated and supported by Arm Engineers

- Available for a wide range of server-class Arm-based platforms
- Validated with NAG's test suite, a de-facto standard

# Arm HPC Ecosystem website

House for Arm's HPC ecosystem, information channels, and collaboration

- Latest events, news, blogs
- Webinars (YouTube), whitepapers and presentations
- Links to OSS & commercial HPC packages
- Recipes for porting HPC apps
- New Arm HPC User Group Forum



The screenshot shows the homepage of the Arm HPC Ecosystem website. At the top, there is a teal header with the text "Welcome to the Arm HPC Ecosystem". Below the header is a navigation menu with links for "Overview", "HPC Software", "Blog", "News", "Events", and "Resources". The main content area features four circular icons representing different stages of the HPC ecosystem: "Get started" (gear icon), "Develop" (microscope icon), "Run" (document icon), and "Collaborate" (puzzle pieces icon). Each icon is accompanied by a brief description and a "Learn more" button. Below this section is a dark grey banner with the text "Software Ecosystem for HPC on Arm" and "Learn more about commercial and open-source HPC tools."

[www.arm.com/hpc](http://www.arm.com/hpc)

Thank You!

Danke!

Merci!

谢谢!

ありがとう!

Gracias!

Kiitos!

arm Research

*Thanks to Eric Van Hensbergen, Chris Goodyer, Miguel Tairum-Cruz, Alex Rico, Jose Joao, Giacomo Gabrielli, Olly Perks*

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.

**arm** Research

[www.arm.com/company/policies/trademarks](http://www.arm.com/company/policies/trademarks)