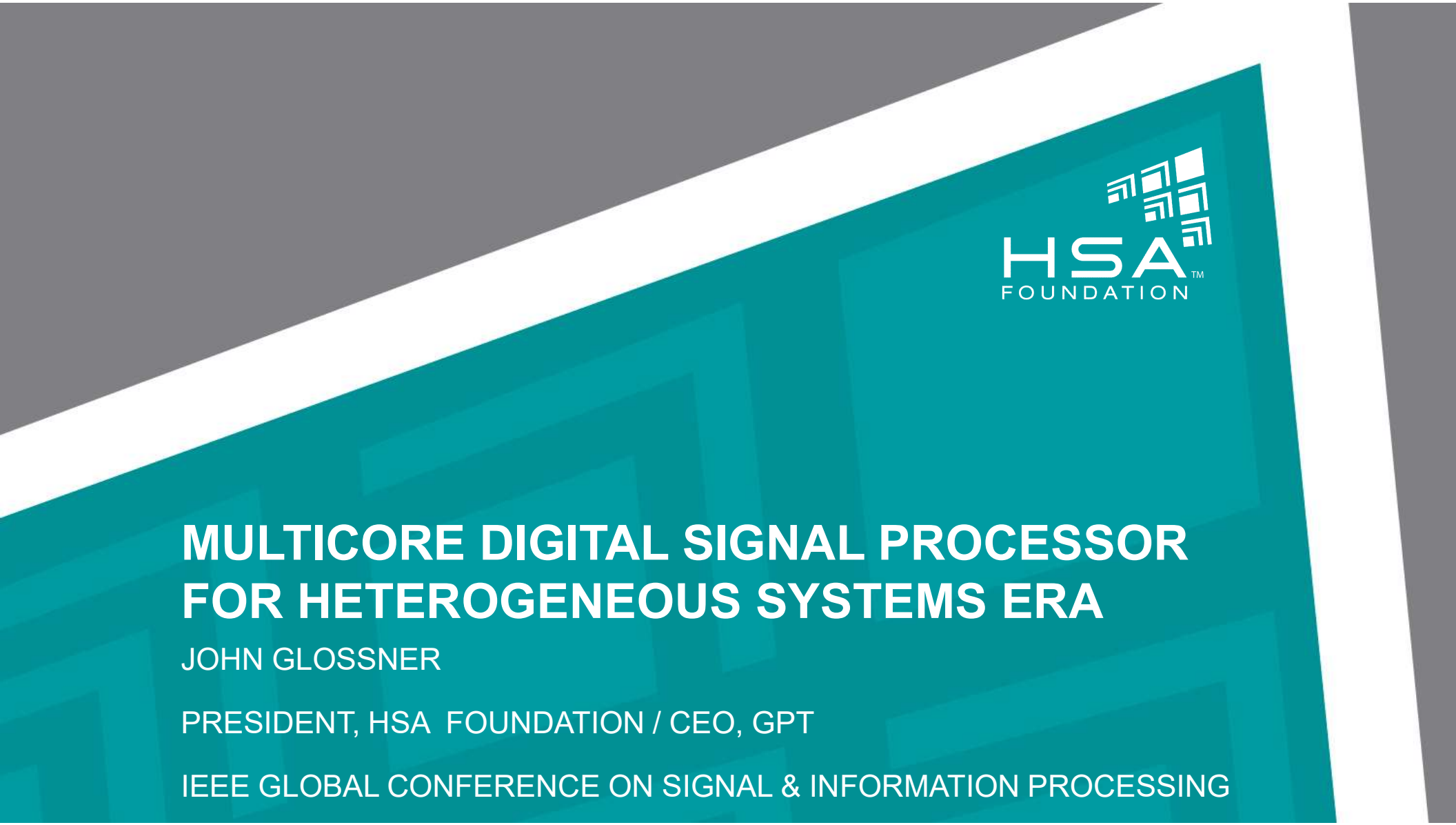


MULTICORE DIGITAL SIGNAL PROCESSOR FOR HETEROGENEOUS SYSTEMS ERA

JOHN GLOSSNER

PRESIDENT, HSA FOUNDATION / CEO, GPT

IEEE GLOBAL CONFERENCE ON SIGNAL & INFORMATION PROCESSING

WHAT'S THE PROBLEM?

- Heterogeneous processors becoming widely available
- Developers starting to discover the benefits of GPU compute augmenting CPU cluster-based compute
- Coherency migrating from high-end servers to mainstream mobile and consumer

BUT…

- Software programming models for coherent heterogeneous processors not standardized
- CPU+GPU applications difficult to optimize or scale, or to port from one SoC platform to another
- Heterogeneous portable application developer ecosystems not gaining momentum

We need to bring compute app portability to heterogeneous platforms!

THE HSA VISION

MAKE HETEROGENEOUS PROGRAMMING MUCH EASIER

1	Single source programming	Finally!
2	Enable the programming language of the developer	C++, Python, JavaScript, …
3	Eliminate data copies	Performance!
4	Common address space	A pointer is a pointer … really … try it!
5	Standardized command submission to Agents (GPU / DSP)	Can we all get along?…..
6	Eliminate software layers between application and hardware	Efficient!
7	ISA agnostic for CPU, GPU, DSP, and more	x86, ARM, MIPS, PowerVR, Mali, Adreno, GPT, …
8	Open source software stack	Open Access!

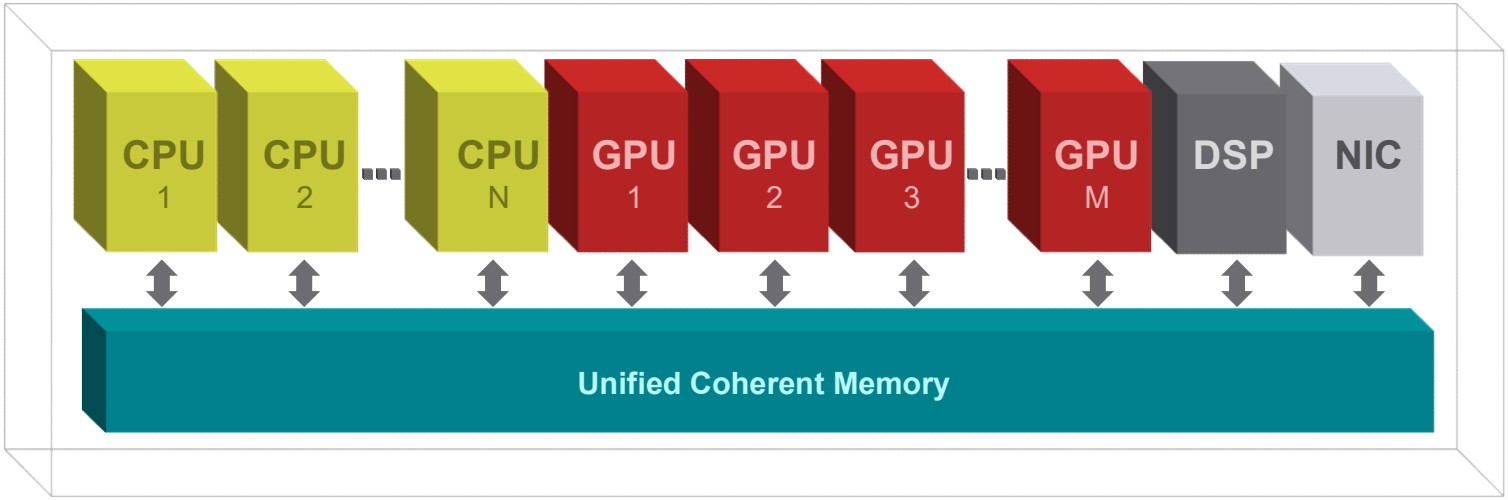
High performance

Low power

Extensible to other accelerators on the SoC

HSA ADDS VALUE TO THE SOC AND THE SYSTEM

- Improves the SoC and the programming paradigm for developers
- Enables heterogeneous interoperability between blocks from different IP suppliers
- Creates a consistent software architecture for heterogeneous acceleration: GPU and beyond



Lower power, higher performance, easier to program

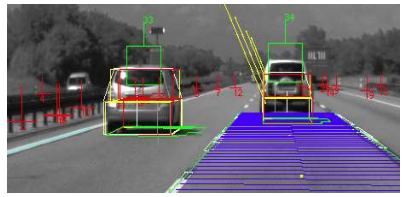
1.0 specs are released; multiple companies set to deliver products

END USERS BENEFIT FROM HSA WITH APPLICATIONS THAT RUN FASTER AND AT LOWER POWER

Always on, visually aware devices will offer greater capability in a lower power budget, scaling with every advance in app processing

Mobile and tablet devices will use the CPU, GPU and DSP working seamlessly together for content creation, gaming and more

Intelligent cloud video analytics will be more efficient, and make best use of every server upgrade



Sophisticated ADAS real-time analytics will be easier to develop, adapt to any platform, and be more robust

HSA architecturally integrates the accelerators in today's complex SoCs to be easily and efficiently utilized by application developers

HSA FOUNDATION

Founded in June 2012

Developing a new platform for heterogeneous systems

Working groups established to define the platform

- Specifications

Membership consists of 43 companies and 16 universities

www.hsafoundation.com

MEMBERS DRIVING HSA

HSA FOUNDATION

Founders

AMD, ARM, Imagination, MediaTek, Qualcomm, Samsung

Promoters

LG Electronics, GPT HX

Supporters

allinea, Argonne National Laboratory, UChicago Argonne LLC, Arteris, codeplay, Fabric Engine, Kishonti, Lawrence Livermore National Laboratory, Linaro, Multicore Ware, Oak Ridge National Laboratory, Sandia National Laboratories, Stream Computing Performance Engineers, SUSE

Contributors

Analog Devices, Canonical, CEVA, DMP, ETRI 한국전자통신연구원 Electronics and Telecommunications Research Institute, Huawei, ITRI Industrial Technology Research Institute, Luxoft, Marvell, mobica, Sonics, Sony make.believe, Swarm64, Synopsys, tensilica, Texas Instruments, Toshiba Leading Innovation, VIA

Academic

icsa, NTHU Programming Language Lab, NTHU System Software Lab, Northeastern, Rice, Tsinghua University, Tampere University of Technology, T.E.I. of Crete, The University of Mississippi, UNT University of North Texas, University of Bristol, informatics, Illinois Computer Science, UWE Bristol

HSA™ FOUNDATION

HSA TECHNOLOGY

FROM HSA FOUNDATION MEMBER COMPANIES

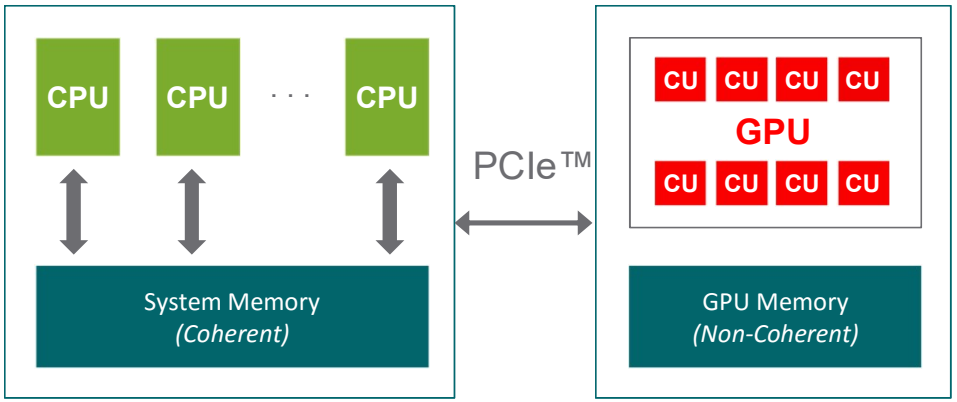
LEGACY GPU COMPUTE

The Limiters

- Multiple memory pools
- Multiple address spaces
- High overhead dispatch
- Data copies across PCIe
- New languages for programming
- Dual source development
- Proprietary environments

- Expert programmers only

- Need to fix all of this to unleash our programmers



DSPs have the same issue

EXISTING APUS AND SOCS

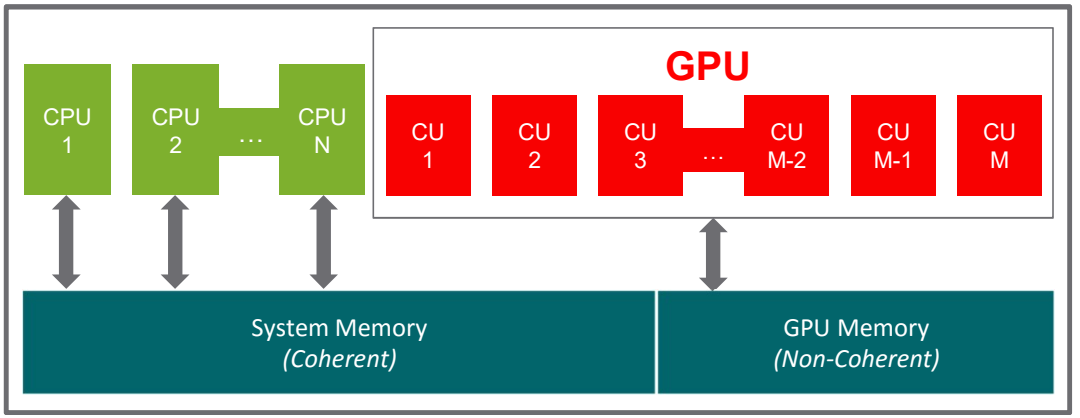
The Limiters

- Physical Integration
- Good first step

- Some copies gone
- Two memory pools remain
- Still queue through the OS

- Still requires expert programmers
- Need to finish the job

Physical Integration



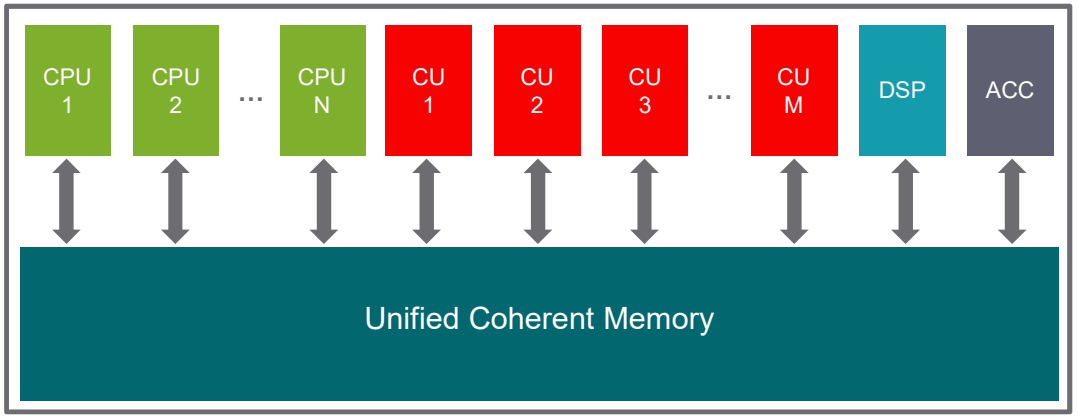
DSPs have the same issue

AN HSA ENABLED SOC

Unified Coherent Memory enables data sharing across all processors

Processors architected to operate cooperatively

Designed to enable the application to run on different processors at different times




HSA
FOUNDATION
HSA TECHNICAL DETAILS

PILLARS OF HSA

Unified addressing across all processors

Operation into pageable system memory

Full memory coherency

User mode dispatch

Architected queuing language

Scheduling and context switching

HSA Intermediate Language (HSAIL)

High level language support for GPU/DSP compute processors

HSA – AN OPEN PLATFORM

Open Architecture, membership open to all

- HSA Programmers Reference Manual
- HSA System Architecture
- HSA Runtime

Delivered via royalty free standards

- Royalty Free IP, Specifications and APIs

ISA agnostic for CPU, GPU and DSP

Membership from all areas of computing

- Hardware companies
- Operating Systems
- Tools and Middleware
- Applications
- Universities

TERMS

Host CPU

- An agent that supports a native CPU instruction set
- Can dispatch commands to kernel agents
- Can construct Architected Query Language (AQL) packets
- Can also act as a kernel agent

Kernel Agent

- An agent that supports HSAIL
- Has an AQL packet processor
- Can dispatch commands to any kernel agent
  - Including itself

Other Agent

- An agent that participates in the HSA memory model

HSA PROGRAMMING MODELS

Single source

- Host and agent code side-by-side in same source file
- Written in same programming language

Single unified coherent address space

- Freely share pointers between host and agent
- Similar memory model as multi-core CPU

Parallel regions identified with existing language syntax

- Typically same syntax used for multi-core CPU

HSAIL is the compiler IR that supports these programming models

Supported Languages

- OpenCL 2.0, OpenMP, C++AMP, Java

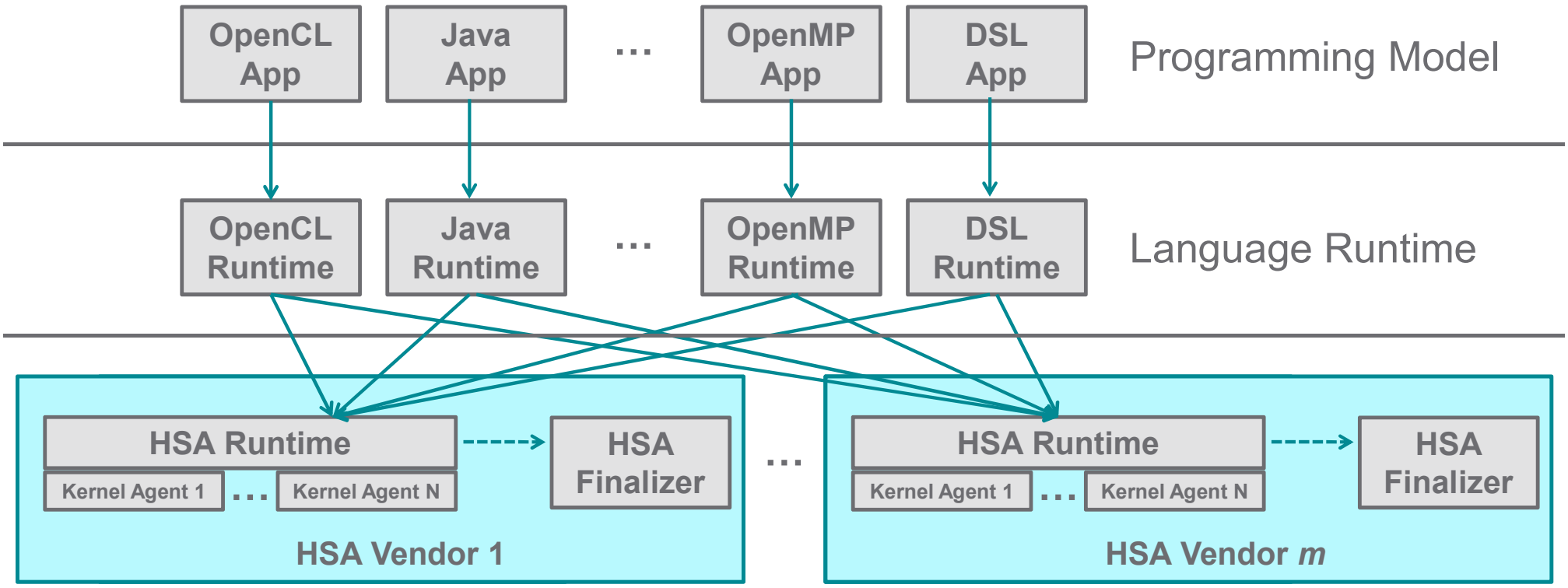
HSA Core Runtime API

- Thin user-mode API
- High performance dispatch
  - Across multiple vendors

Functions

- Initialization and Shut Down
- Notifications (Synchronous/Asynchronous)
- Agent Information
- Signals and Synchronization (Memory-Based)
- Queues and Architected Dispatch
- Memory Management

S/W ARCHITECTURE WITH HSA RUNTIME



HSA MEMORY MODEL

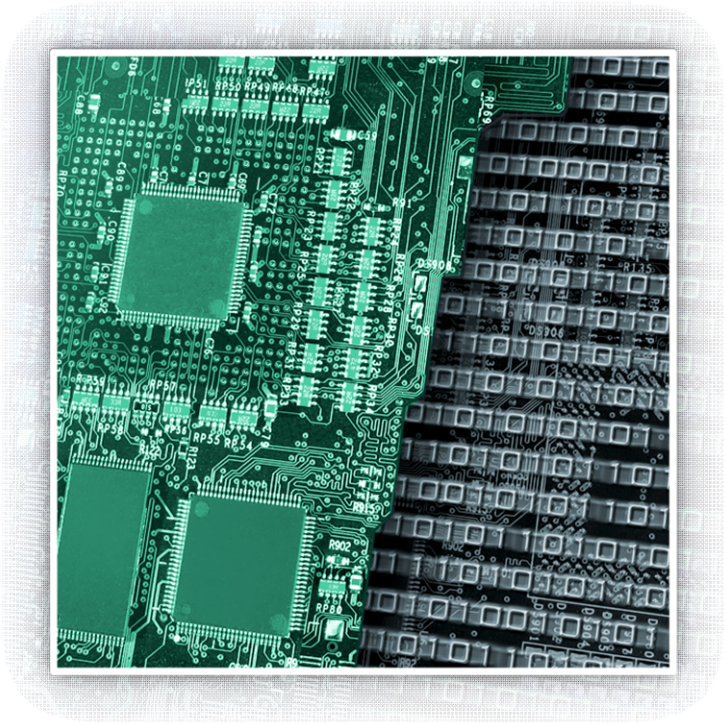
Defines visibility ordering between all threads in the HSA System

Designed to be compatible with C++11, Java, OpenCL and .NET Memory Models

Relaxed consistency memory model for parallel compute performance

Visibility controlled by:

- Load.Acquire
- Store.Release
- Fences



HSA QUEUING MODEL

User mode queuing for low latency dispatch

- Application dispatches directly
- No OS or driver required in the dispatch path

Architected Queuing Layer

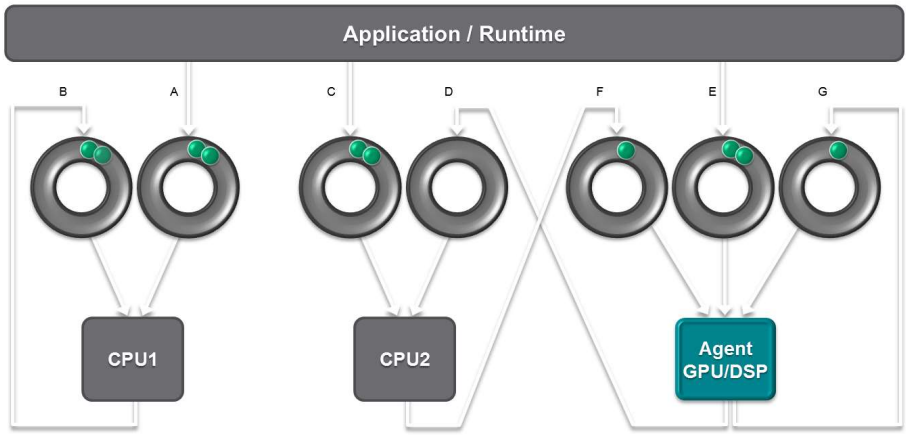
- Single compute dispatch path for all hardware
- No driver translation, direct to hardware

Allows for dispatch to queue from any agent

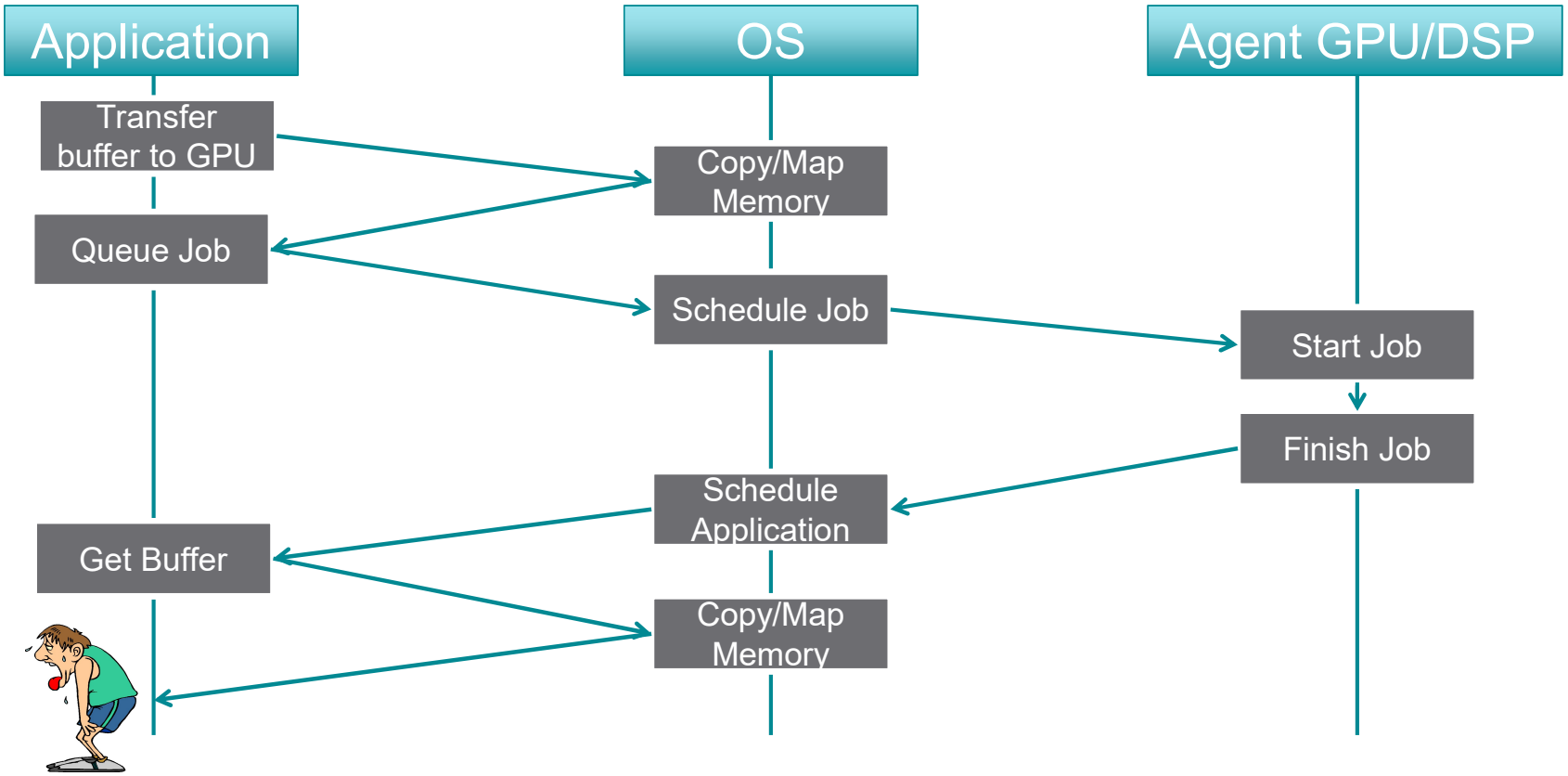
- CPU or GPU or DSP

GPU/DSP self enqueue enables lots of solutions

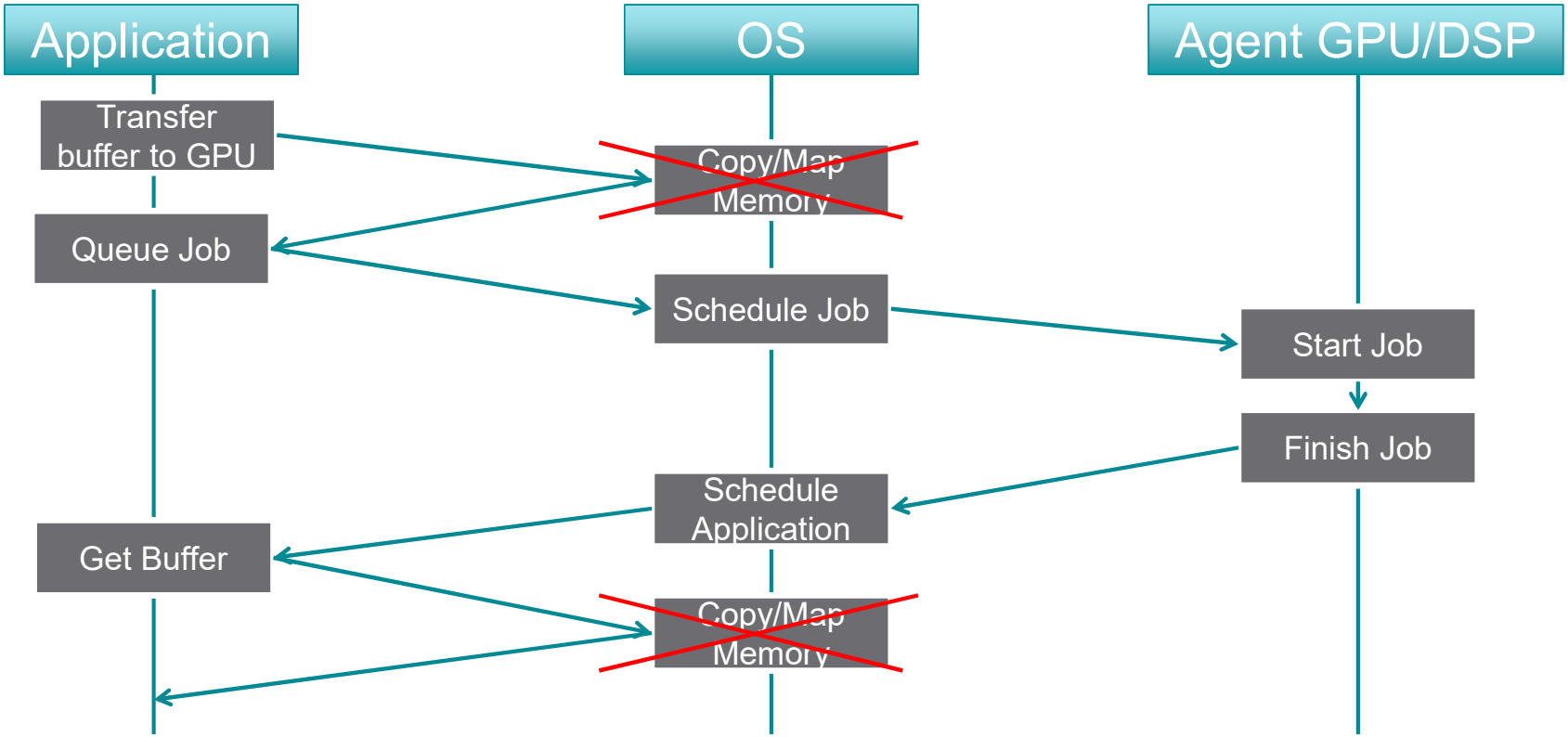
- Recursion
- Tree traversal
- Wavefront reforming



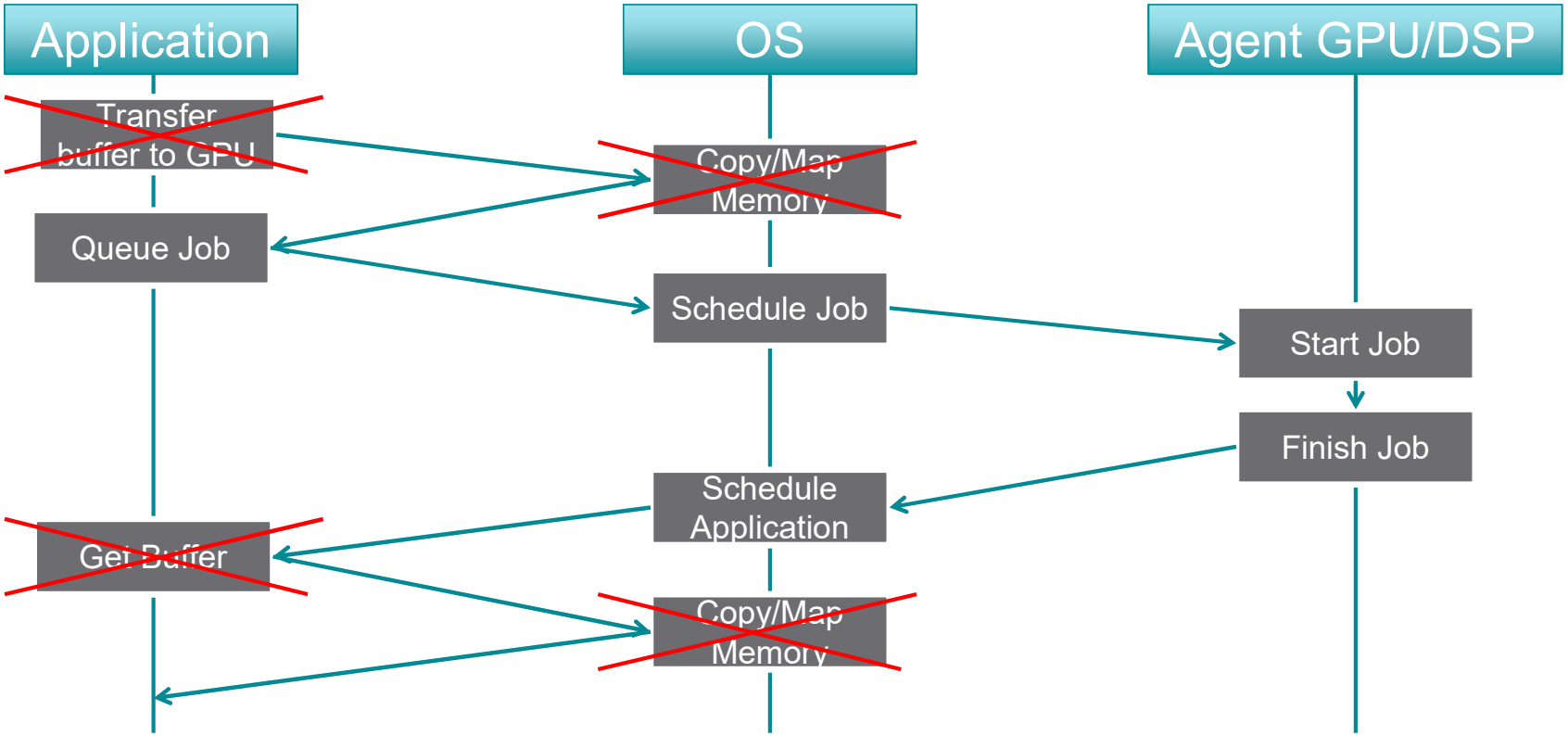
MOTIVATION (TODAY'S PICTURE)



WITH SHARED VIRTUAL MEMORY



WITH COHERENT CACHE MEMORY



SIGNALS

HSA agents support signaling

- creation/destruction using runtime APIs

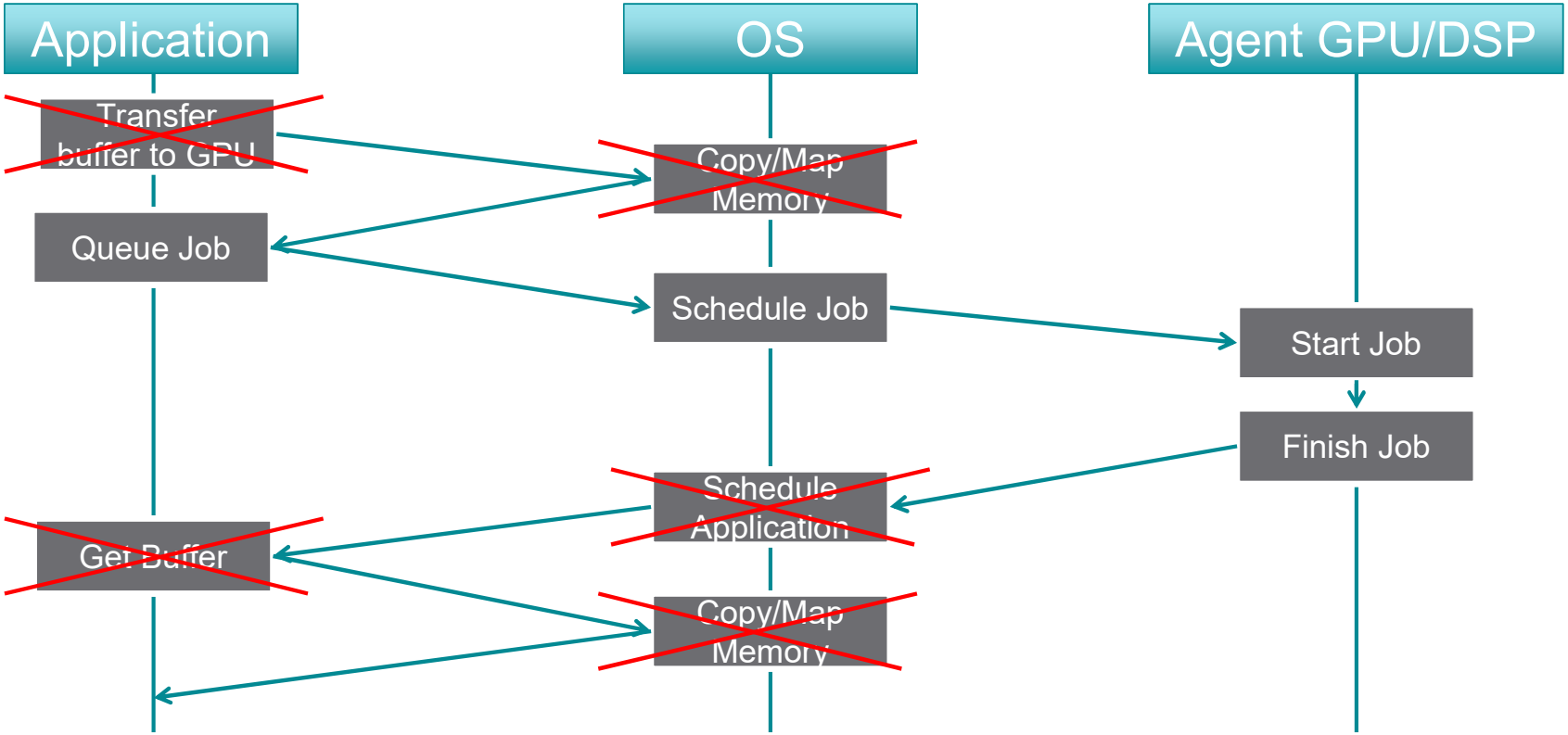
Any Agent can directly access signals

- wake up HSA agents waiting upon the object
- Query current object
- Wait on the current object
  - Using conditions

Advantages

- Asynchronous events between agents
  - Doesn't require CPU
- Common idiom for work offload
- Low power waiting

WITH SIGNALLING



USER MODE QUEUING

User mode Queueing

- Enables user space applications to enqueue jobs
  - No OS intervention
- Created/destroyed via calls to the HSA runtime.
- One (or many) agents enqueue packets
  - A single agent dequeues packets.

Requires coherency and shared virtual memory.

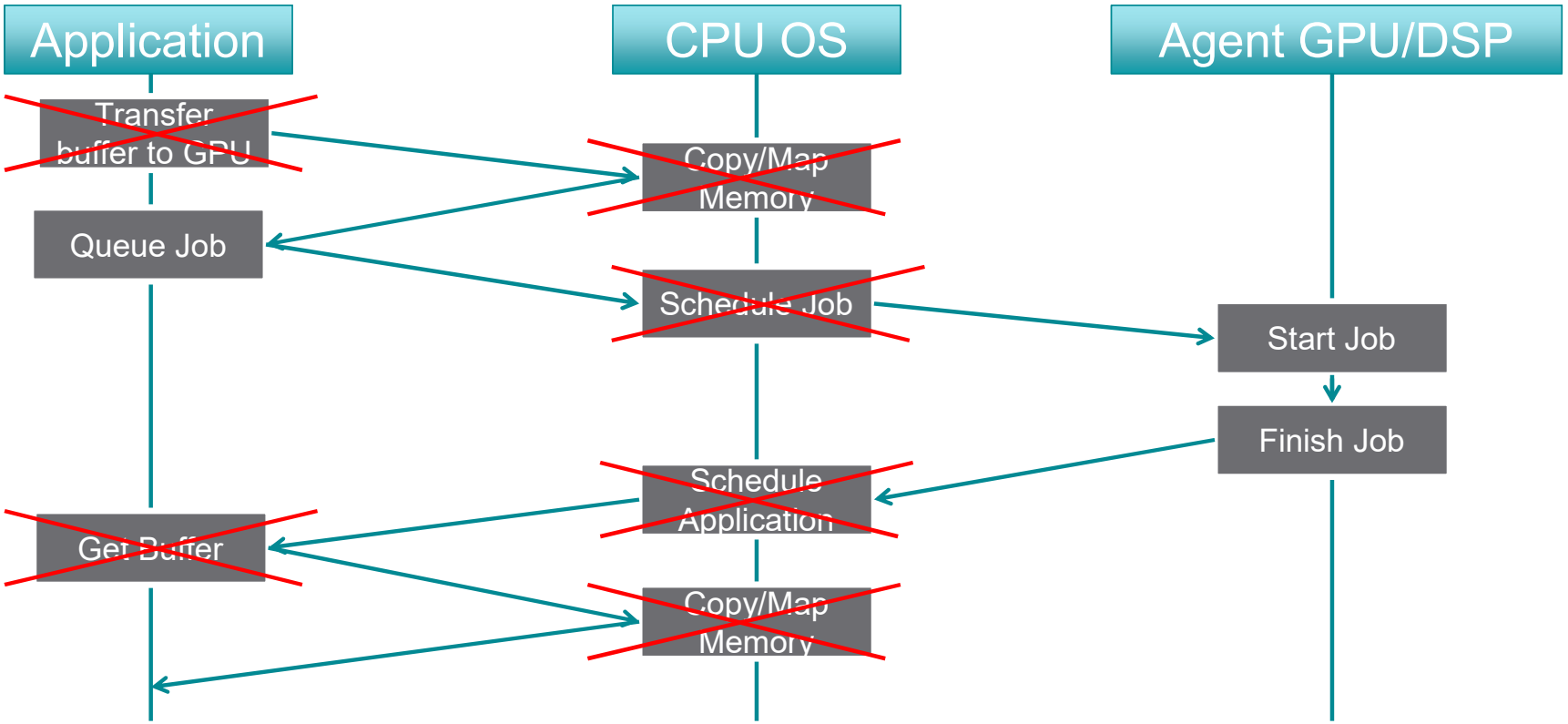
Advantages

- Avoids OS/Driver involvement
- Lower latency enables finer granularity of offload
- Standard memory protection mechanisms
  - Protects communication with the consuming agent.

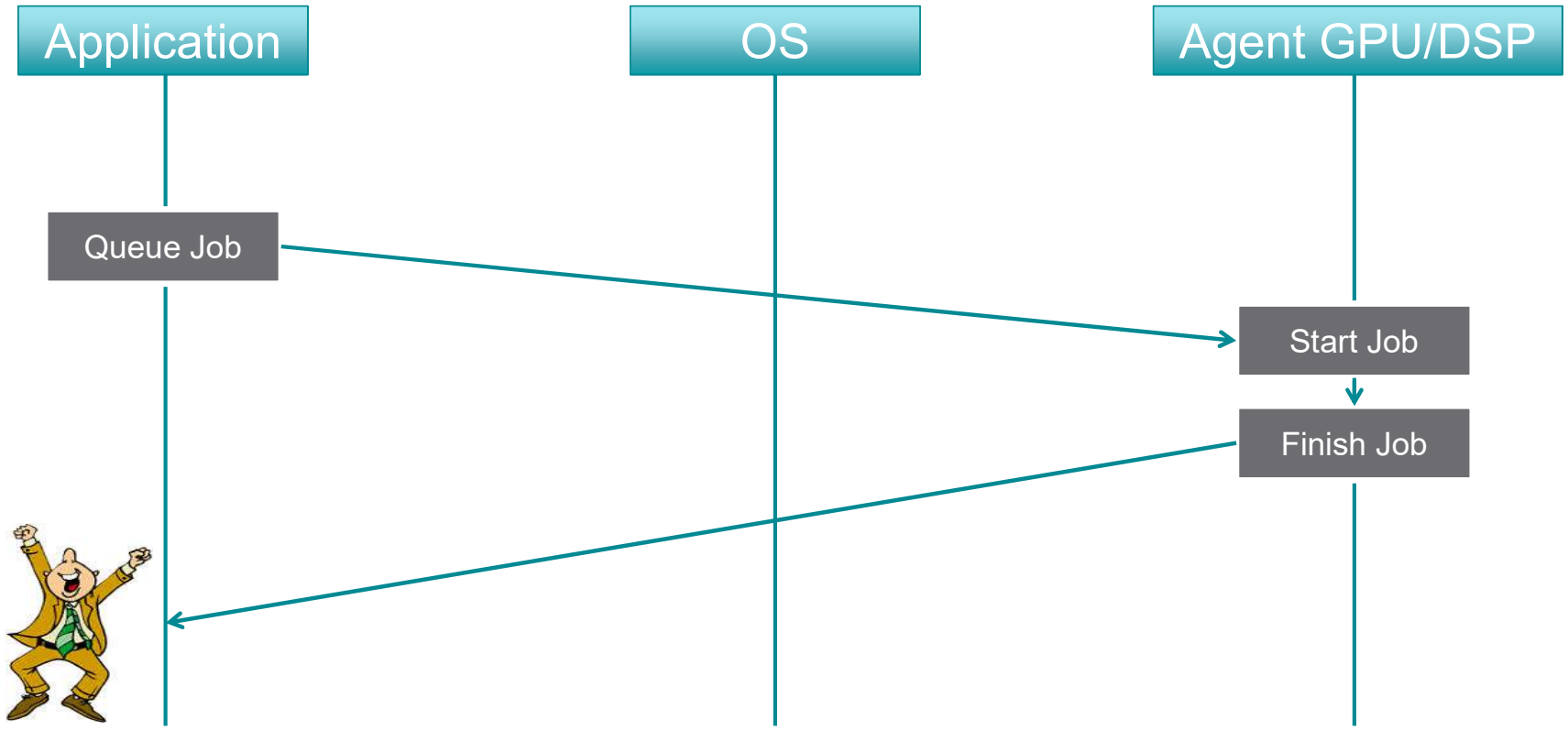
Implications

- Packet formats/fields are Architected
  - Standard across vendors!
  - Guaranteed backward compatibility
- Packets are enqueued/dequeued
  - Architected protocol
  - Via memory accesses and signaling

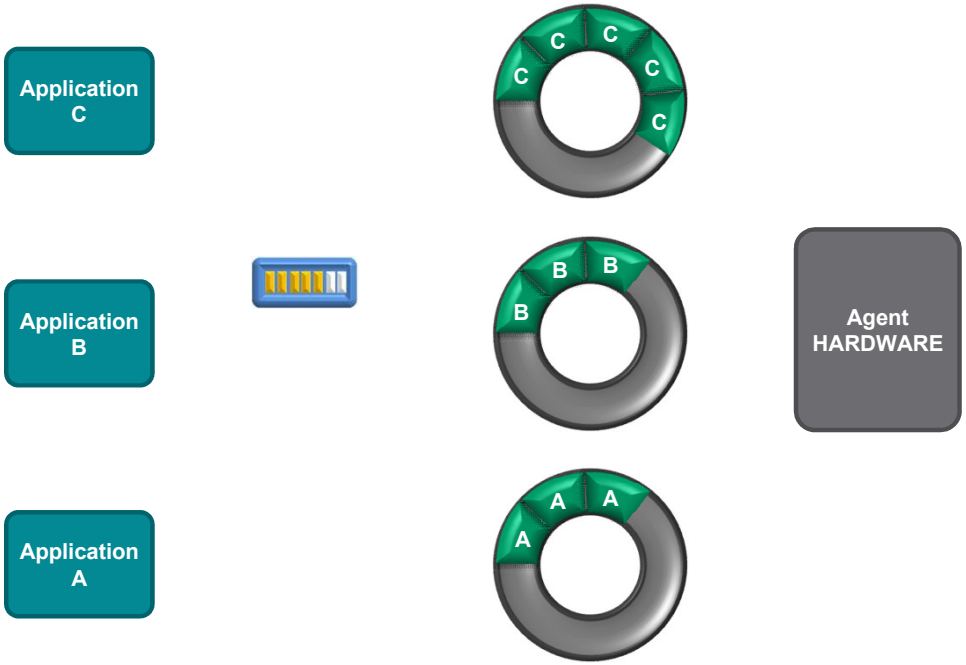
WITH USER MODE QUEUING



FINAL PICTURE: SVM + CACHE COHERENCY + SIGNALS + USER MODE QUEUES



HSA COMMAND AND DISPATCH FLOW



SW view:

- User-mode dispatches to HW
- No KMD overhead
- Low dispatch times
- CPU & GPU dispatch APIs

HW view:

- HW / microcode controlled
- HW scheduling
- Architected Queuing Language (AQL)
- HW-managed protection

HSA INTERMEDIATE LANGUAGE (HSAIL)

WHAT MAKES IT SPECIAL?

THE PORTABILITY CHALLENGE

CPU ISAs

- ISA innovations added incrementally (ie NEON, AVX, etc)
  - ISA retains backwards-compatibility with previous generation
- HSA instruction-set architectures: ARM, MIPS, and x86

GPU/DSP ISAs

- Massive diversity of architectures in the market
  - Each vendor has own ISA - and often several in market at same time
- No commitment (or attempt!) to provide any backwards compatibility
  - Traditionally graphics APIs (OpenGL, DirectX) provide necessary abstraction

HSA INTERMEDIATE LAYER — HSAIL

HSAIL is a virtual ISA for parallel programs

- Finalized to ISA by a JIT compiler or “Finalizer”
- ISA independent by design for CPU & GPU

Explicitly parallel

- Designed for data parallel programming

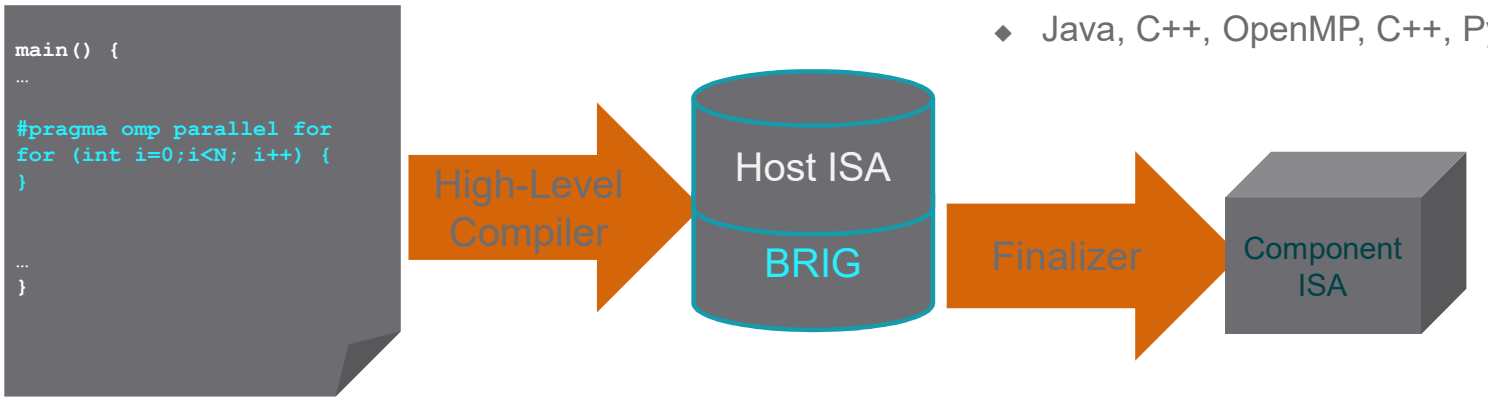
Support for exceptions, virtual functions, and other high level language features

Lower level than OpenCL SPIR

- Fits naturally in the OpenCL compilation stack

Suitable to support additional high level languages and programming models:

- Java, C++, OpenMP, C++, Python, etc



HSAIL FEATURES

A Virtual Explicitly Parallel ISA

- ~135 Opcodes
- RISC Register-based Load/Store
- Arithmetic
  - IEEE 754 Floating Point including 16-bit
  - Integer (32/64-bit)
  - DSP fixed point
  - Packed / SIMD
    - f16x2, f16x4, f16x8, f32x2, f32x4, f64x2
    - signed/unsigned 8x4, 8x8, 8x16, 16x2, 16x4, 16x8, 32x2, 32x4, 64x2
- Branches & Function Calls
- Atomic Operations

Wavefronts

- 1, 2, 4, 8, 16, 32, or 64 SIMD lanes
- Lanes can be active or inactive

Memory

- Shared Virtual Memory

Exceptions

```
ld_global_u64    $d0, [$d6 + 120] ; $d0= load($d6+120)
add_u64          $d1, $d0, 24     ; $d1= $d2+24
```

SEGMENTS AND MEMORY

7 segments of memory

- Memory instructions can (optionally) specify a segment
- Control data sharing properties and communicate intent

Global Segment

- Visible to all HSA agents (including host CPU)

Group Segment

- Provides high-performance memory shared in the work-group.
- Group memory can be read and written by any work-item in the work-group
- HSAIL provides sync operations to control visibility of group memory

Spill, Private, Arg Segments

- Represent different regions of a per-work-item stack
- Typically generated by compiler, not specified by programmer
- Compiler can use these to convey intent – ie spills

Kernarg Segment

- Programmer writes kernarg segment to pass arguments to a kernel

Read-Only Segment

- Remains constant during execution of kernel

```
ld_global_u64   $d0,[$d6]
ld_group_u64    $d0,[$d6+24]
st_spill_f32    $s1,[$d6+4]
```

FLAT ADDRESSING

Each segment mapped into virtual address space

- Flat addresses can map to segments based on virtual address

Instructions with no explicit segment use flat addressing

Very useful for high-level language support (ie classes, libraries)

Aligns well with OpenCL 2.0 “generic” addressing feature

```
ld_global_u64  $d6, [%_arg0] ; global
ld_u64         $d0,[$d6+24]  ; flat
```

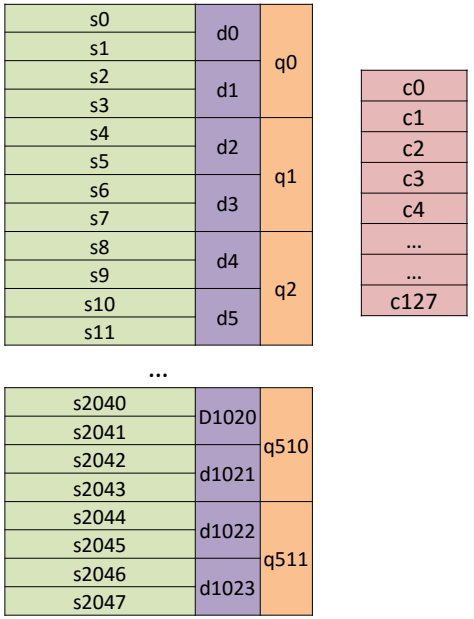
REGISTERS

Four classes of registers:

- S: 32-bit, Single-precision FP or Int
- D: 64-bit, Double-precision FP or Long Int
- Q: 128-bit, Packed data.
- C: 1-bit, Control Registers (Compares)

Fixed number of registers

- S, D, Q share a single pool of resources
- S + 2*D + 4*Q <= 2048
- Up to 2048 S or 1024 D or 512 Q (or a blend)



HSA PARALLEL EXECUTION MODEL

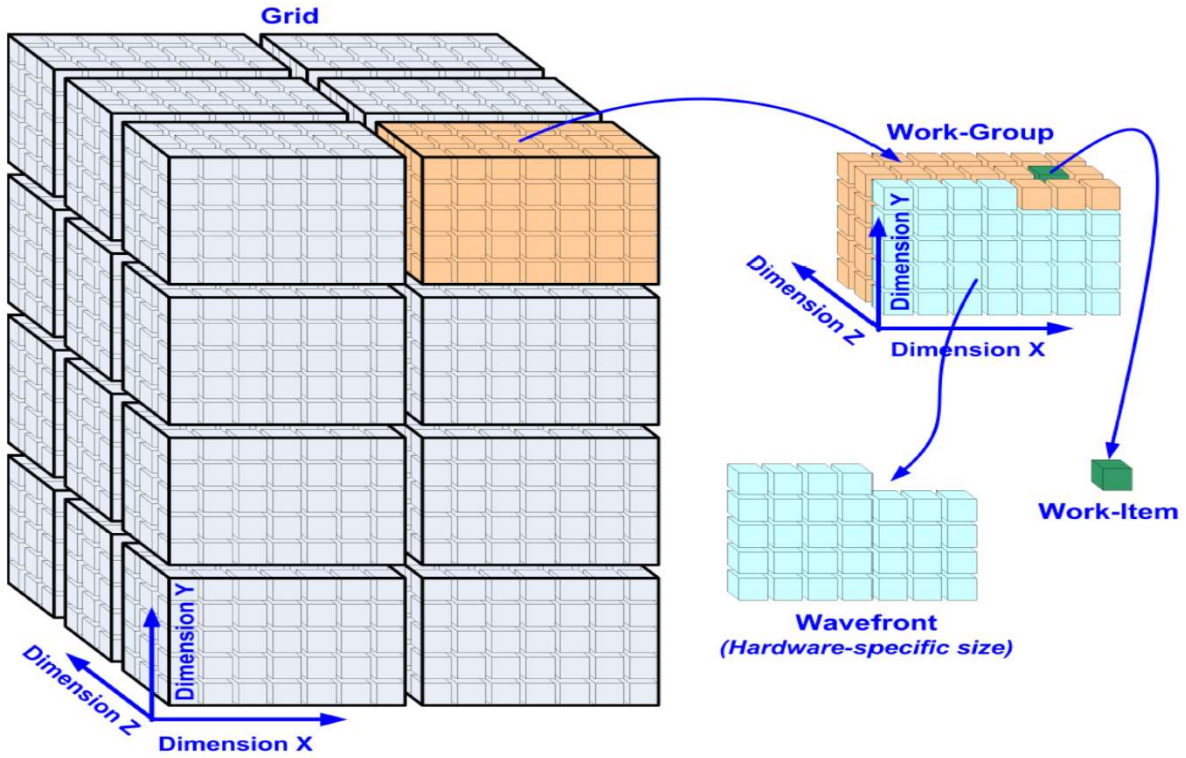
Programmer supplies an HSAIL "kernel"

- run on each work-item
- a single thread of execution

Programmer specifies grid dimensions when launching the kernel.

Each work-item has a unique coordinate in the grid.

Programmer optionally specifies work-group dimensions (for optimized communication).



HSAIL MODES: MACHINE MODELS AND PROFILES

Working group strived to limit optional modes and features in HSAIL

- Minimize differences between HSA target machines
- Better for compiler vendors and application developers
- Two modes survived

Machine Models

- Small: 32-bit pointers, 32-bit data
- Large: 64-bit pointers, 32-bit or 64-bit data
- Vendors can support one or both models

“Base” and “Full” Profiles

- Two sets of requirements for FP accuracy, rounding, exception reporting, hard pre-emption

GPT Architecture

Design Goals

1 Unified system architecture to support many applications

- Control Code
- Vision Processing: Graphics, OpenCL, Transforms
- DSP: Filters, FFTs

Many microarchitectures for many applications

- Scalar/superscalar
- In-order/out-of-order
- Short/long pipelines
- Single-threaded/multithreaded

A few realizations

- low power process for battery apps
- high performance process for powered apps

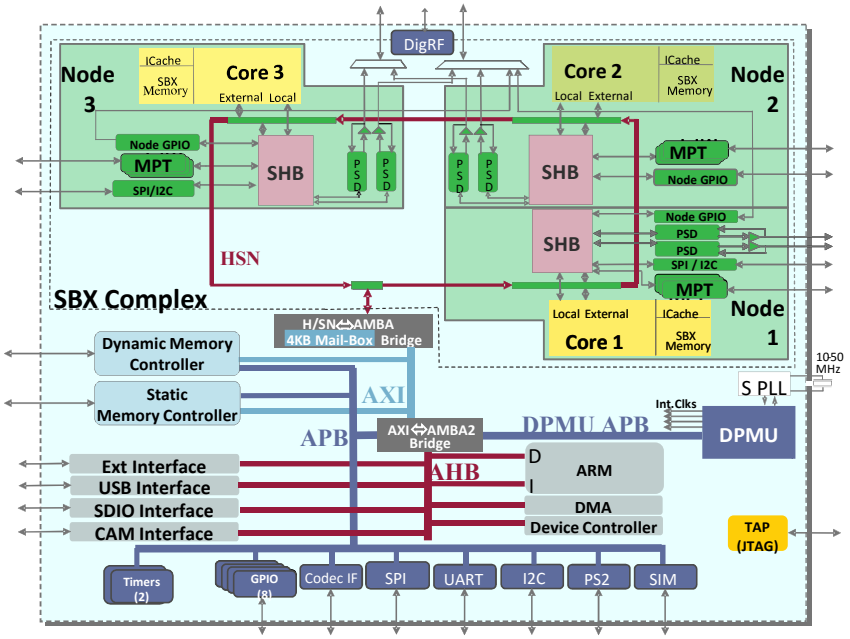
Strategy / Key Core Enhancements

Improve design concepts utilized on SB3500

- Heterogeneous chip implementation
- Unique Hardware Multi-threading
- Multi-node / scalable Implementation
- Common external interface and peripheral blocks
- Adaptable Processor complex interconnect
- Complete SW environment (comp, linker, sim, …)

DSP Core feature examples

- Optimized instructions in support of Key Algorithms
- Sophisticated power control strategy
- Floating-point Vector Unit
- Reduced latency memory access
- Optimized data-access engine
  - ✓ scatter / gather DMA functionality



Starting from a base

HSA

- Support CPU, DSP, GPU & Accelerator Functions
- Scalable Performance
  - ✓Handsets
  - ✓Consumer Electronics
  - ✓Applications Processors
  - ✓High Performance Computing
- Parallel Execution
- Low Power

Sandblaster DSP

- A DSP with some CPU
  - ✓16-bit arithmetic
  - ✓Special instructions
- ✓Handsets
- Token Triggered Threaded
  - ✓Compound
  - ✓SIMD ISA
- Low Power

Unity 1.0

- HSA Compatible
  - ✓Accelerators
- Scalable Performance
- Opportunity Threaded
  - ✓Chained Instructions
  - ✓Vector ISA
- Low Power

GPU vs DSP

```
float out[N][4];
float in[N][4];
float matrix[4][4];
for( i = 0; i < N; i++ )
   for( j = 0; j < 4; j++ )
      v = 0;
       for( k = 0; k < 4; k++ )
           v += in[i][j] * matrix[j][k];
       out[i][j] = v;
```

transformation stream of 4-vectors being multiplied by the same 4x4 matrix

Huge data bandwidth requirement

- GB/s

Huge memory footprint

- 100MBs

```
float out[N];
float in[N+T-1];
float taps[T];
for( i = 0; i < N; i++ )
  v = 0;
  for( j = 0; j < T; j++ )
    v += in[i+j]*taps[j];
  out[i] = v;
```

stream of data being dot-producted, then shifted by one and dot producted again

Much smaller data bandwidth requirement

- 10MB/s

Much smaller memory footprint

- MBs

General ISA Approach

64-bit Control Code

- Instruction fetch support
  - ✓Branch via target vs. PC+offset
- Multi-unit support based on instruction type
  - ✓Integer, FP, branch, address, memory
- Virtualization support
- Cache coherency

Vector Parallel Compute

- General Vector instructions
- Specialized Vector instructions
  - ✓Domain Specific

ISA Approach: Control Code

64-bit registers

- general purpose
- separate address

Branch

- Compare and branch
- Indirect via register only
- Specialized branch instructions

Address

- Separate address register file
- Specialized address instructions

Memory

- Load/store
- Base-only
- Post-increment

System

- Virtual addressing
  - ✓Configurable pages
- Hypervisor support
  - ✓Multiple privilege level
  - ✓All system resources accessed via trap

GPT Summary

DSP as Licensable IP

- Multithreaded Control Code
- Matrix/Vector Parallel Compute Code

HSA Support

- HSA Agent
  - ✓HSAIL Finalizer
- Coherent Shared Virtual Memory

HSA PRODUCT UPDATES

FROM HSA FOUNDATION MEMBER COMPANIES

GCC HSAIL OPEN SOURCE PROJECT

Sponsored by General Processor Technologies

- gccbrig

An open source project

- A BRIG language front-end to GCC
  - BRIG: Binary Representation of HSAIL
  - Translated to GCC's tree intermediate representation
- Optimization by GCC
  - Including vectorization/SIMD optimizations

Benefits

- Allows use of GCC for finalization
- Vendor independent
  - No need to know proprietary Instruction Set Architecture
- CPU/VLIW/MIMD HSA kernel agent support

Schedule

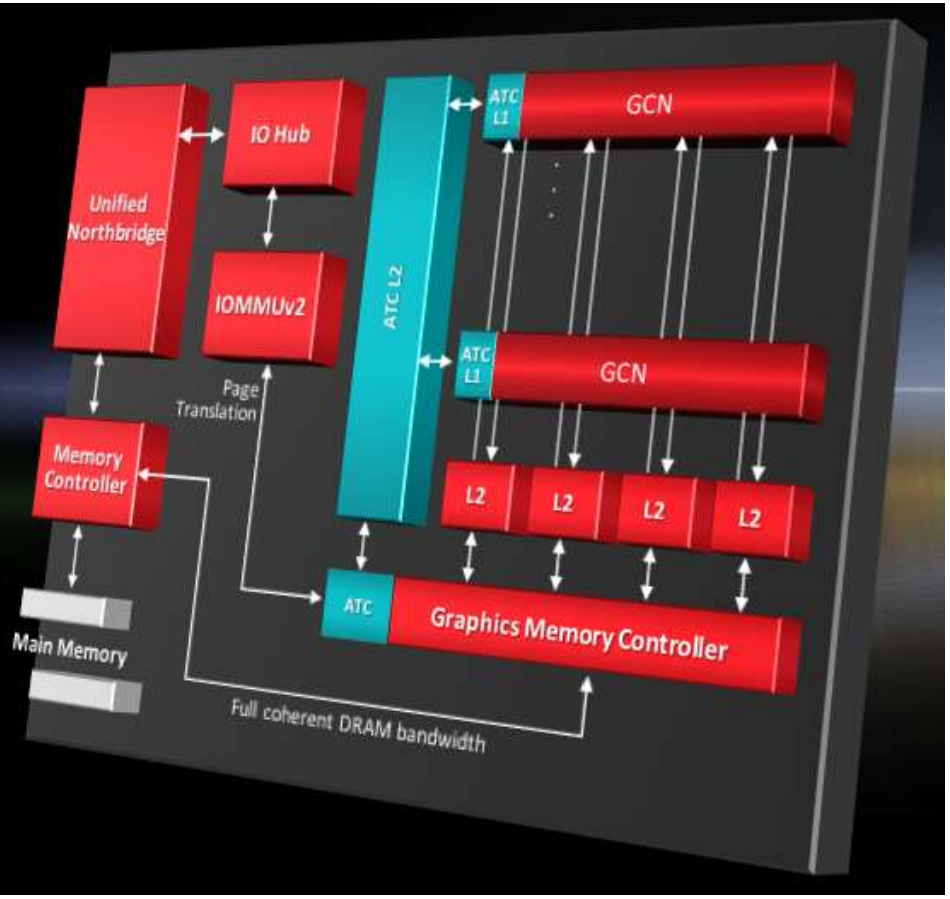
- Public release 1Q2016
- Developer access - TODAY

“CARRIZO” IS AMD’S SECOND APU PRODUCT WITH HSA FEATURES

FULLY HSA ENABLED

HSA ACCELERATION FEATURES

- Address Translation Cache (ATC) hierarchy
  - Improves virtual memory address translation throughput for “pointer is a pointer” data sharing between CPU and GPU
- Full hardware cache coherence at maximum DRAM bandwidth between GPU and CPU caches
- Support for Wavefront and Compute task pre-emption and context switching
  - Improves work scheduling efficiency
- HSA QoS scheduling support



ARM AND HSA

ARM, as a founder member, has been committed to the HSAF since launch

- Actively contributes to the HSA specifications and working groups
- Is committed to the continued development of this important standard

ARM customer base is showing increasing interest in HSA features for their next generation SoCs

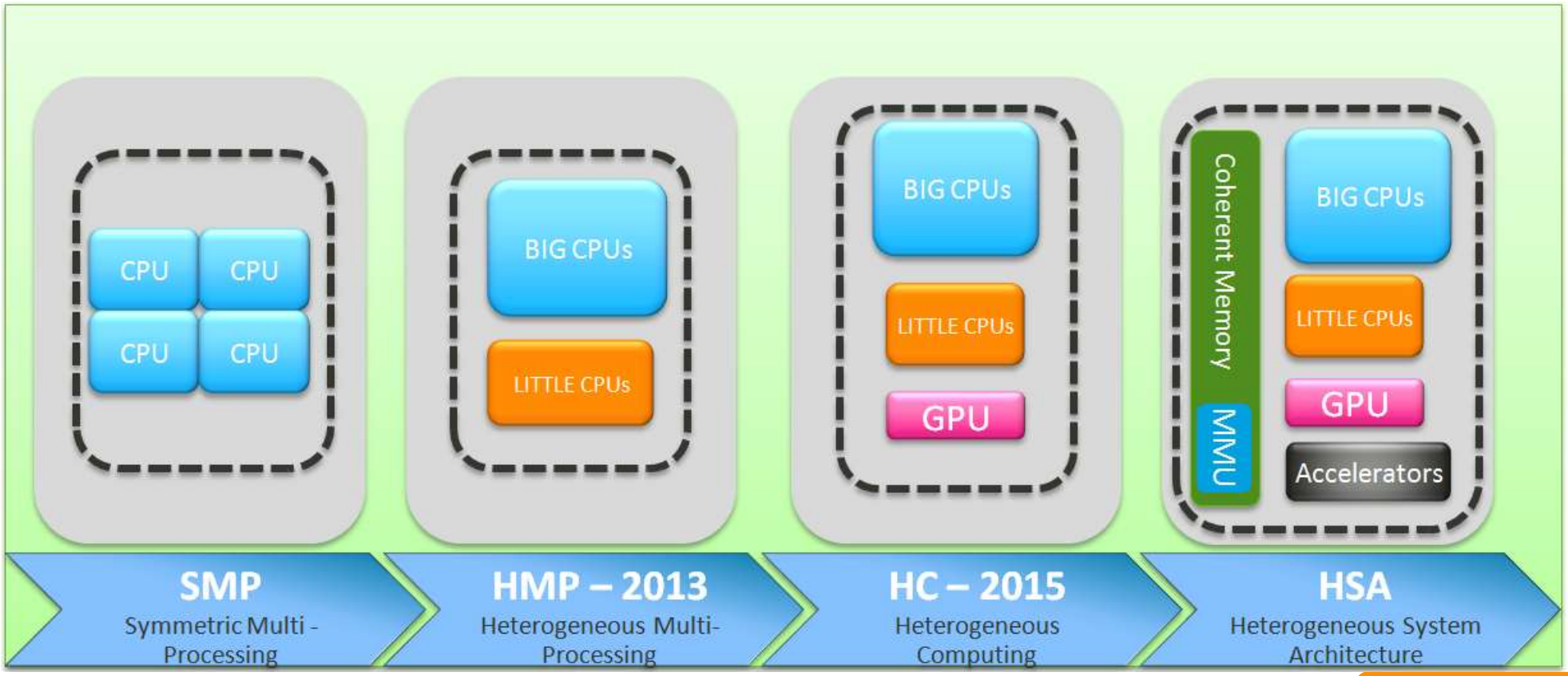
ARM customers can already build real heterogeneous systems based on, for example:

- ARM Cortex-A72 high performance application processor
- ARM Mali-T880 compute enabled GPU
- ARM CoreLink CCI-500 cache coherent interconnect
- ARM CoreLink CCN cache coherent network family

ARM is actively developing next generation processor and interconnect IP to extend the system capabilities aligned with HSA standards including:

- Full memory coherency
- Shared Virtual Memory

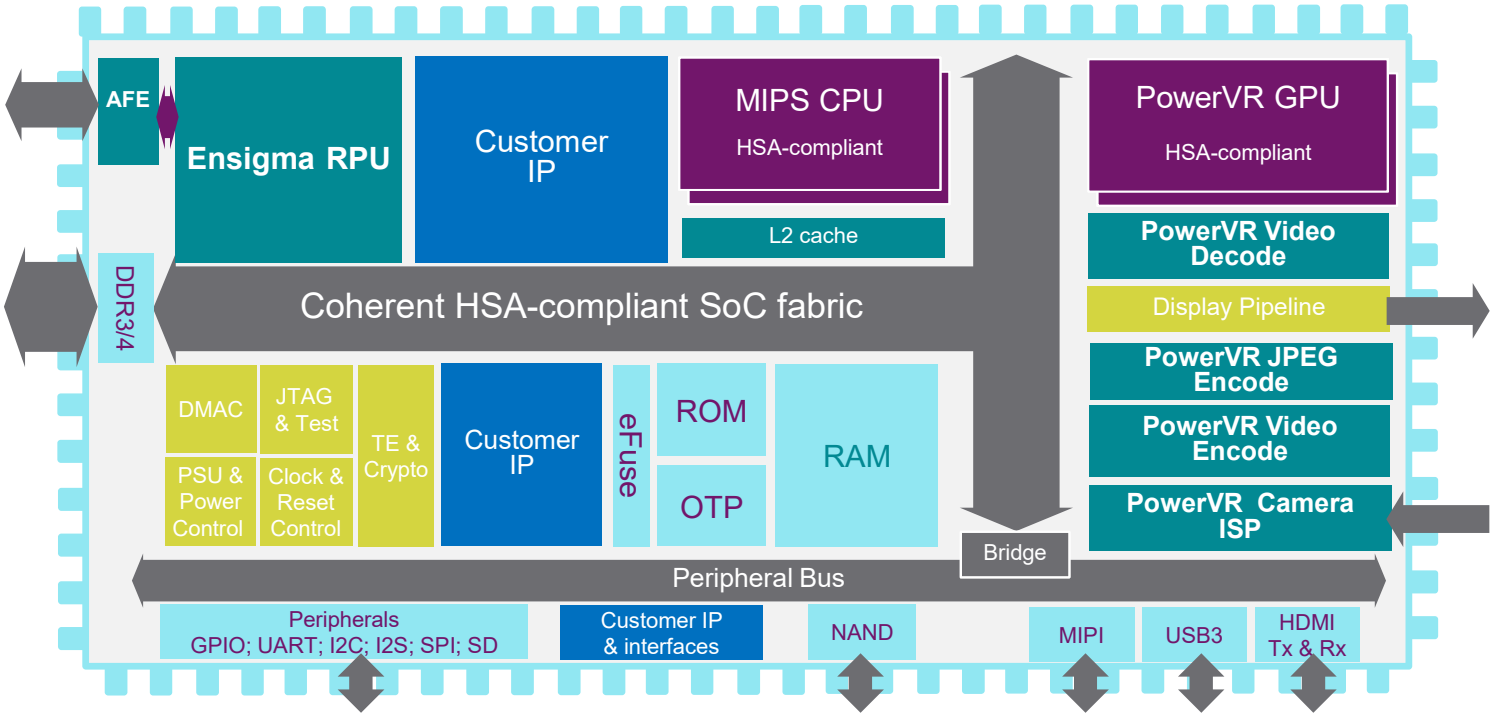
HSA ROADMAP AT MEDIATEK



10/5/2015

IMAGINATION HSA COMPLIANT IP COMING SOON

Imagination Smart Vision IP Platform



We will be rolling out:

- HSA across all MIPS I-class and P-class CPUs
- HSA across all PowerVR GPUs
- HSA compliant fabric solutions

WHERE HSA IS NOW

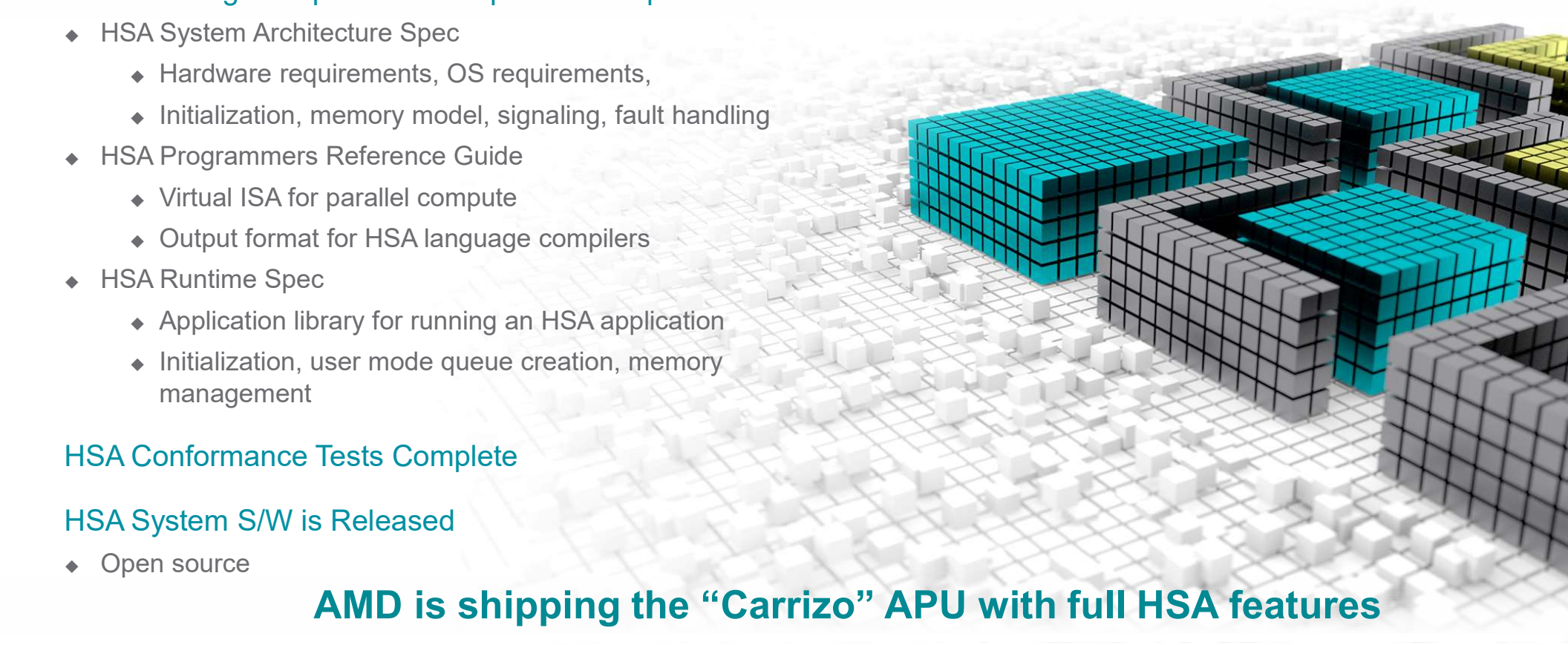
HSA Working Groups Have Completed 1.0 Specs

- HSA System Architecture Spec
  - Hardware requirements, OS requirements,
  - Initialization, memory model, signaling, fault handling
- HSA Programmers Reference Guide
  - Virtual ISA for parallel compute
  - Output format for HSA language compilers
- HSA Runtime Spec
  - Application library for running an HSA application
  - Initialization, user mode queue creation, memory management

HSA Conformance Tests Complete

HSA System S/W is Released

- Open source



AMD is shipping the “Carrizo” APU with full HSA features

AMD is shipping the “Carrizo” APU with full HSA features

WHAT'S NEXT FOR HSA FOUNDATION?

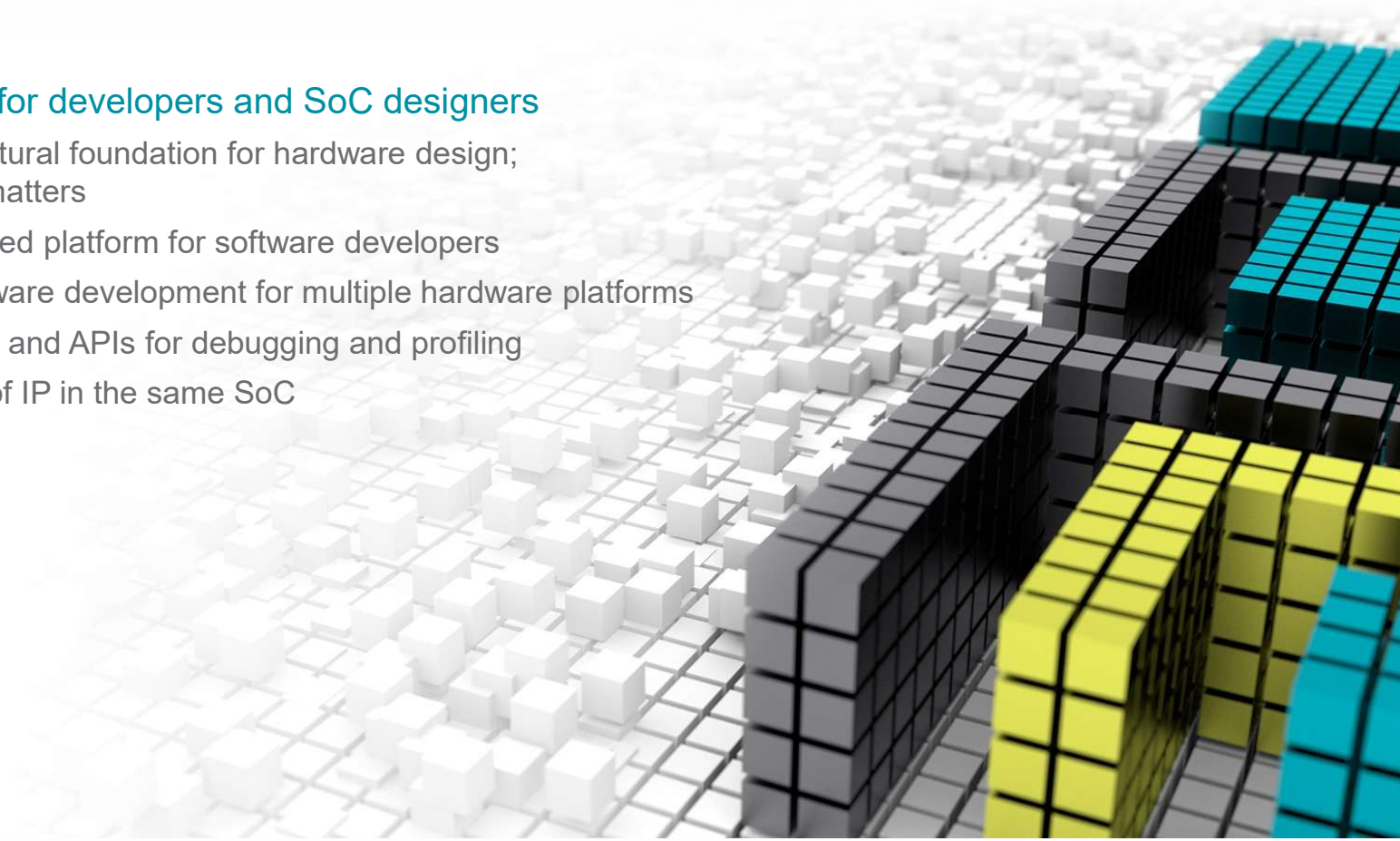
HSA working groups are already working on 1.1

- Backward compatible with 1.0
- Standardize tools APIs for debugging and profiling
- Support multiple vendors of IP in the same SoC

Exciting times coming for HSA

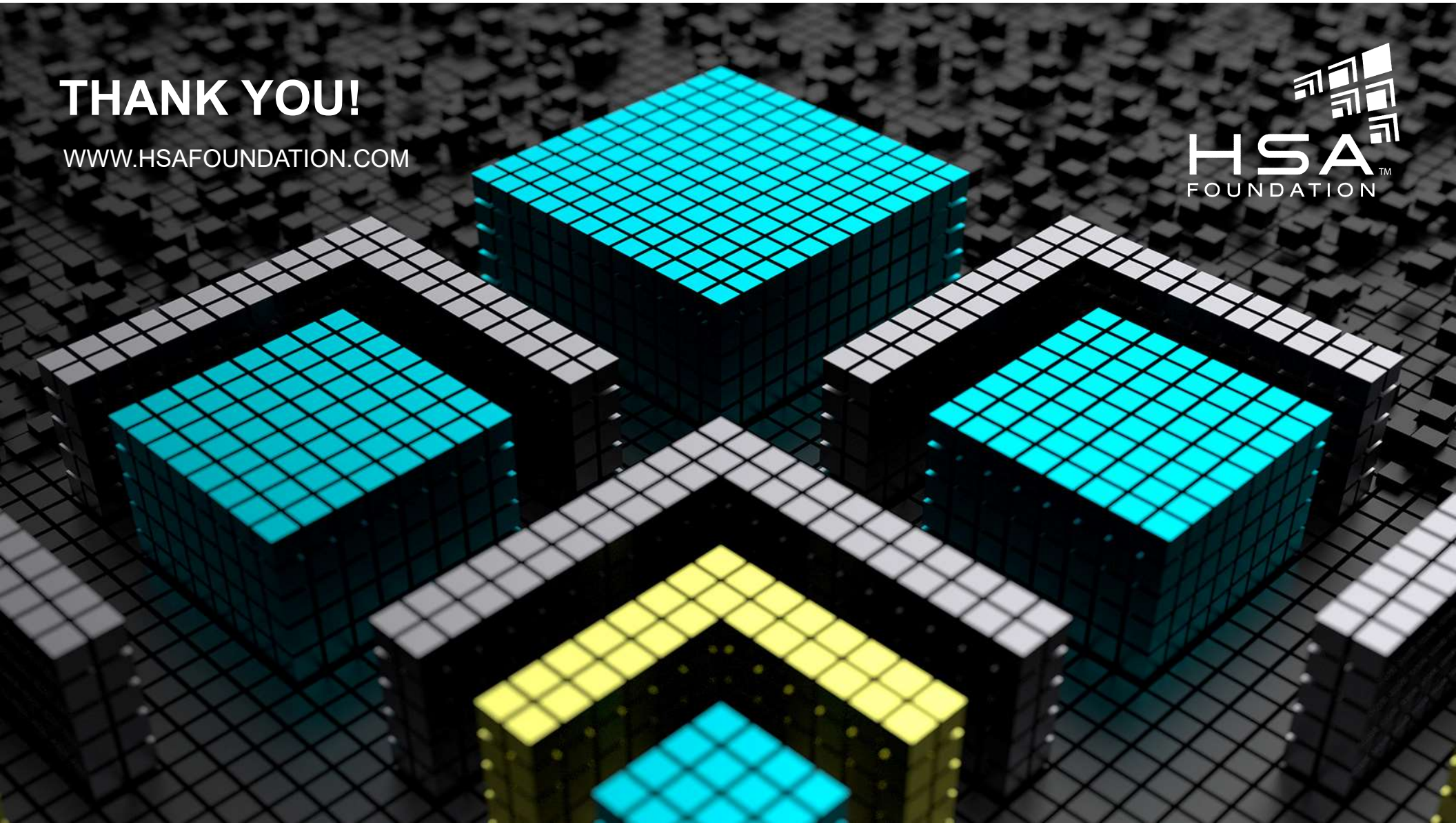
- Expect HSA product announcements from multiple companies
- Phones, tablets, notebooks, workstations, supercomputers

SUMMARY



HSA solves a real problem for developers and SoC designers

- Creates a common architectural foundation for hardware design; allows innovation where it matters
- Ensures a pervasive deployed platform for software developers
- Eases the problems of software development for multiple hardware platforms
- Provides standardized tools and APIs for debugging and profiling
- Supports multiple vendors of IP in the same SoC


THANK YOU!
WWW.HSAFOUNDATION.COM
HSA™
FOUNDATION