AMD IN HPC

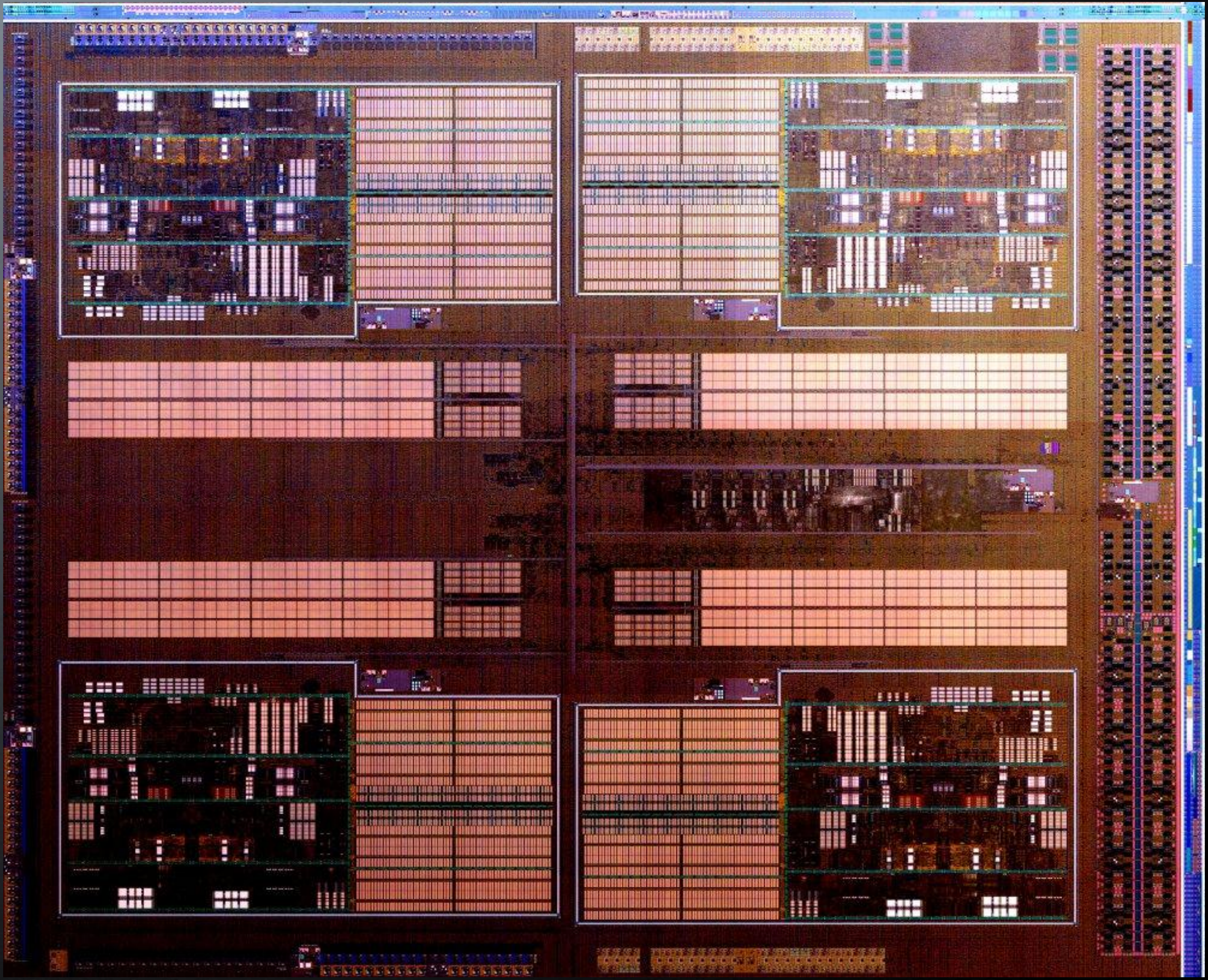
André Heidekrueger
HPC Presales Engineer
AMD EMEA

November 2011

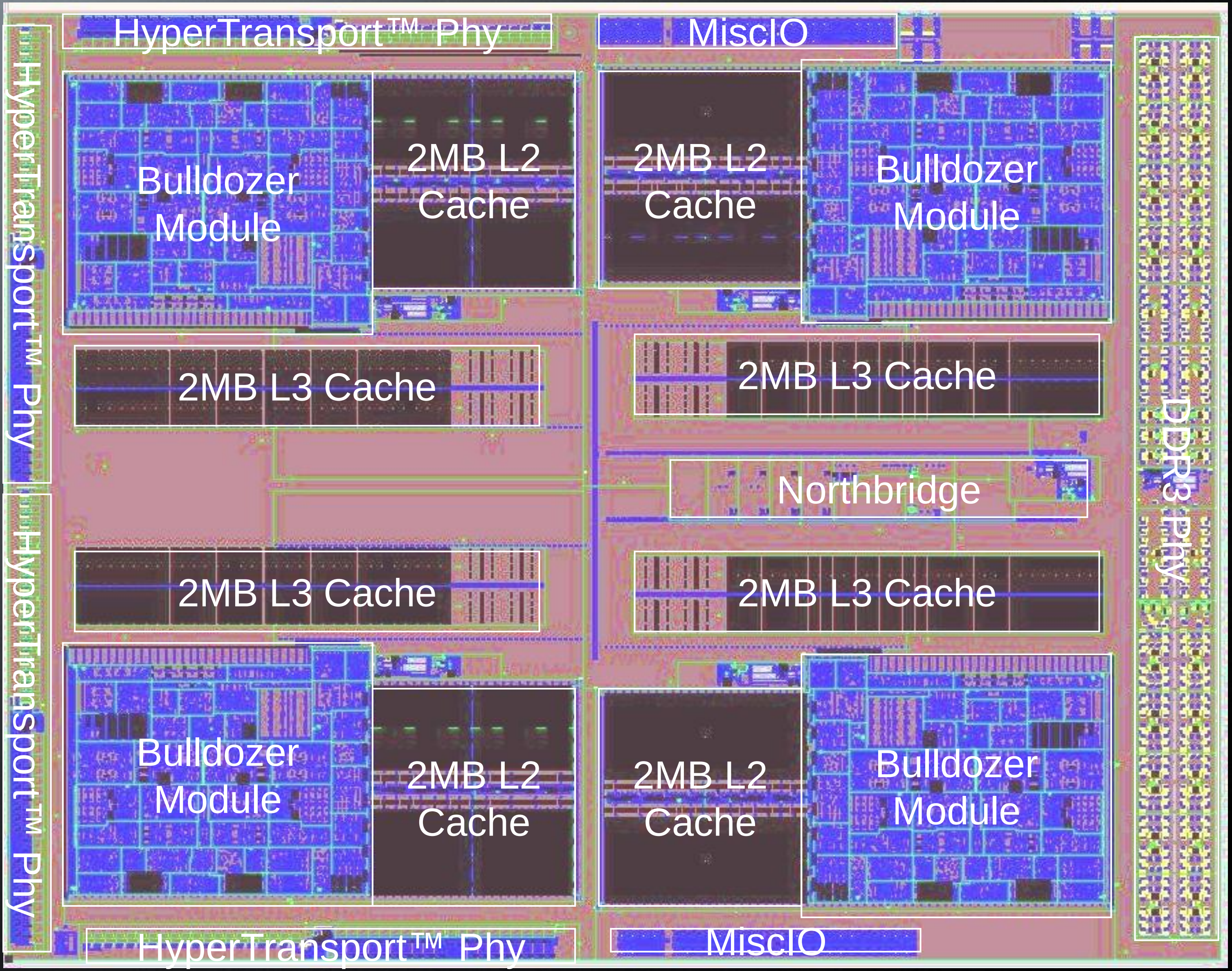
AMD IN TOP 500 – JUNE 2011 LIST

- **22 of the 50** fastest supercomputers on the TOP500 list are using AMD's CPU and/or GPU technology.
- **66 systems** on the TOP500 are powered by AMD Opteron™ processors, and some new ones coming soon

THE DIE | Photograph



THE DIE | Floorplan (315 mm^2)



COMPUTE UNIT - OPTIMIZED PERFORMANCE/WATT

Leadership Multi-Threaded Micro-Architecture

Full Performance From Each Core

- Dedicated execution units per core
  - Up to 128 Instructions/Core in flight/Cycle/Core
- No shared execution units as with SMT

High Frequency / Low-Power Design

- Core Performance Boost
  - "Boosts" frequency of cores when available power allows
  - No idle core requirement
- Power efficiency enhancements
  - Significantly reduced leakage power
    - 32nm, Hi-K metal gate
  - More aggressive dynamic power mgt
    - Memory Power Capping
    - Northbridge P-States

Virtualization Enhancements

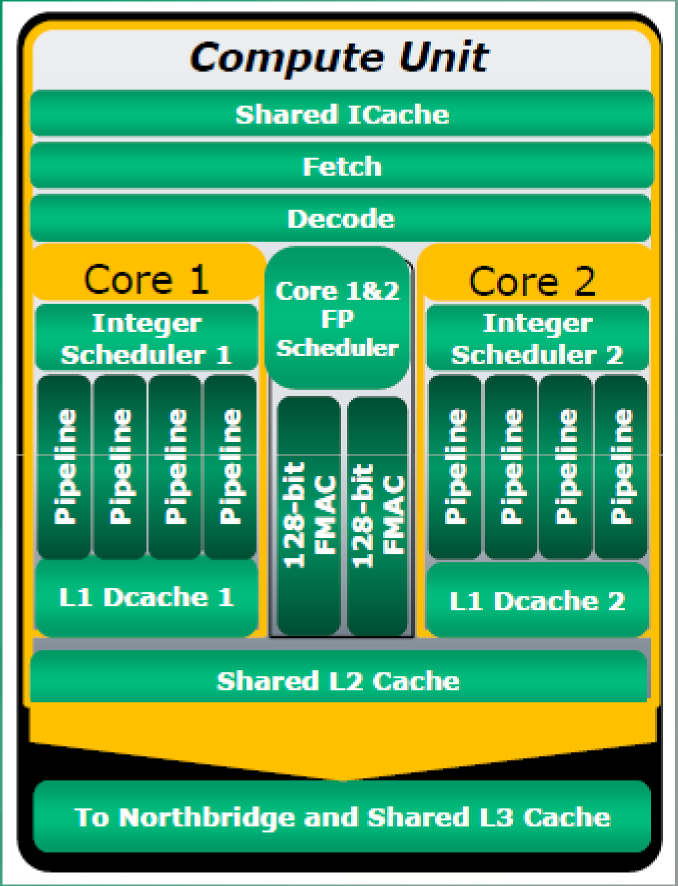
- Faster switching between VMs
  - TLB Flush by ASID, VM State Caching, Decode Assists
- AMD-V extended migration support

Shared Double-sized FPU

- Amortizes very powerful 256-bit unit across both cores
  - Includes FMAC

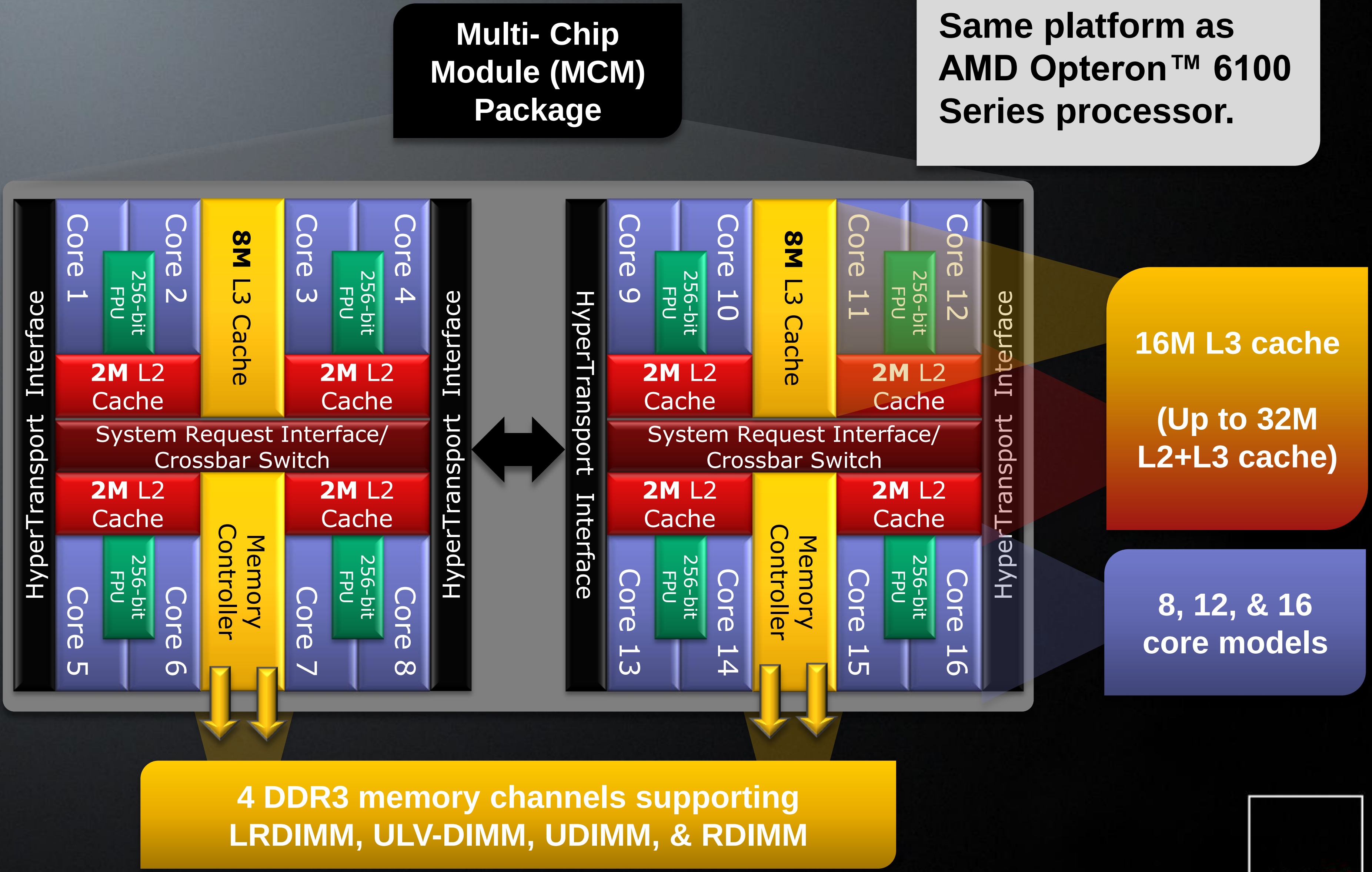
Improved IPC

- Micro-architecture and ISA enhancements
  - SSE4.1/4.2, AVX 1.0/1.1, SSSE3, AES, LWP, XOP, FMA4
  - Independent Core/FP Schedulers, Deeper Buffers, Larger Caches, Branch Fusion, Aggressive Pre-fetching, High Degree of Thread Concurrency Throughout



AMD OPTERON™ 6200 SERIES PROCESSOR (“INTERLAGOS”)

Same platform as AMD Opteron™ 6100 Series processor.



A BROAD PORTFOLIO OF “INTERLAGOS” PRODUCTS

Four Core AMD Opteron™ 6200 Series Processors

Huge memory bandwidth per core

Eight Core AMD Opteron™ 6200 Series Processors

Blend of cores and memory with an edge on memory bandwidth

Twelve Core AMD Opteron™ 6200 Series Processors

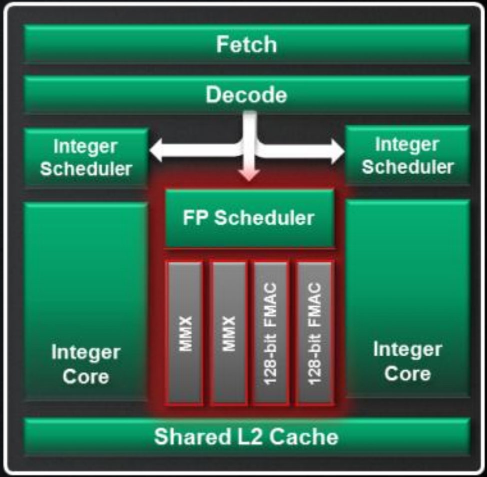
Blend of cores and memory with a little more computation muscle

Sixteen Core AMD Opteron™ 6200 Series Processors

The industry’s only 16-core x86 processor for massive thread density

FLEX FP: MORE FLEXIBLE TECHNICAL PROCESSING

More performance and new instruction support

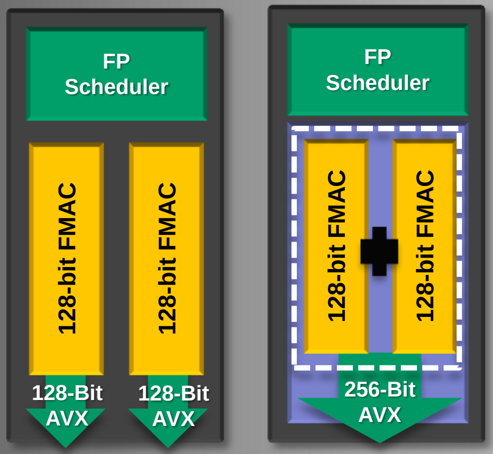


Runs SSE and AVX simultaneously	• No processing penalty for simultaneous execution
Executes two SSE or AVX (128-bit) instructions simultaneously or one AVX (256-bit) instruction per Bulldozer module	• Wider range of FP processing capabilities than competition
Processes calculations in a single cycle using FMA4* and XOP instructions	• Executes more instructions in fewer cycles than competition
Uses dedicated floating point scheduler	• No waiting on integer scheduler to run instructions • Designed to be always available to schedule floating point operations

*FMAC can execute an FMA4 execution (a=b+c*d) in one cycle vs. 2 cycles that would be required for FMA3 or standard SSE floating point calculation.

FLEX FP

	AMD Opteron™ 6100 Series FPU	Intel Sandy Bridge	AMD Opteron™ 6200 Series FlexFP
Execute 128-bit FP	✓	✓	✓
Support SSSE3, SSE4.1, SSE4.2		✓	✓
Execute 128-bit AVX		✓	✓
Execute 256-bit AVX		✓	✓
Execute two 128-bit SSE or AVX ADD instructions in 1 cycle			✓
Execute two 128-bit SSE or AVX MUL instructions in 1 cycle			✓
Switch between SSE and AVX instructions without penalty			✓
Execute FMA operations (A=B+C*D)			✓
Supports XOP			✓
FLOPs per cycle (128-bit FP)	48	32	64
FLOPS per cycle (128-bit AVX)	-	32	64
FLOPS per cycle (256-bit AVX)	-	64	64



Two 128-bit FMACs shared per module, allowing for dedicated 128-bit execution per core or shared 256-bit execution per module

Sandybridge information from http://software.intel.com/en-us/avx/

NEW "BULLDOZER" INSTRUCTIONS

Instructions	Applications/Use Cases
SSSE3, SSE4.1, SSE4.2 (AMD and Intel)	• Video encoding and transcoding • Biometrics algorithms • Text-intensive applications
AESNI PCLMULQDQ (AMD and Intel)	• Application using AES encryption • Secure network transactions • Disk encryption (MSFT BitLocker) • Database encryption (Oracle) • Cloud security
AVX (AMD and Intel)	Floating point intensive applications: • Signal processing / Seismic • Multimedia • Scientific simulations • Financial analytics • 3D modeling
FMA4 (AMD Unique)	HPC applications
XOP (AMD Unique)	• Numeric applications • Multimedia applications • Algorithms used for audio/radio

Software that currently supports SSSE3, SSE4.1, SSE4.2, AESNI, and AVX should run on "Bulldozer."

- No recompile of code if the software only checks ISA feature bits
- Recompile needed if software also checks for processor type

For FMA4 or XOP, software will need to be written to call specific instructions or be compiled with a compiler that will automatically generate code that leverages these instructions.

COMPILER SUPPORT
AMD Opteron™ 6200 Series Processors

Compiler	Status	Support for SSSE3, SSE4.1/4.2, AVX (Intel and AMD)	Support for FMA4, XOP (AMD)	Options to auto generate code with new instruction support
Microsoft Visual Studio 2010 SP1	**Available**	Yes	Yes	No
GCC 4.5, 4.6	**Available**	Yes	Yes	Yes
Open64 4.2.5	**Available**	Yes	Yes	Yes
PGI 11.6, 11.7	**Available**	Yes	No	Yes (for SSSE3, SSE4.1/4.2, AVX)
PGI 11.9	**In Development**	Yes	Yes	Yes
ICC 12*	**Available**	Yes (but ICC runtime fails on AMD processors)	No	Yes (but ICC runtime fails on AMD processors)

*Intel has an –mAVX flag which is designed to run on any x86 processor; however, the ICC runtime makes assumptions about cache line sizes and other parameters that causes code not to run on AMD processors

ACML SUPPORT | A CLOSER LOOK

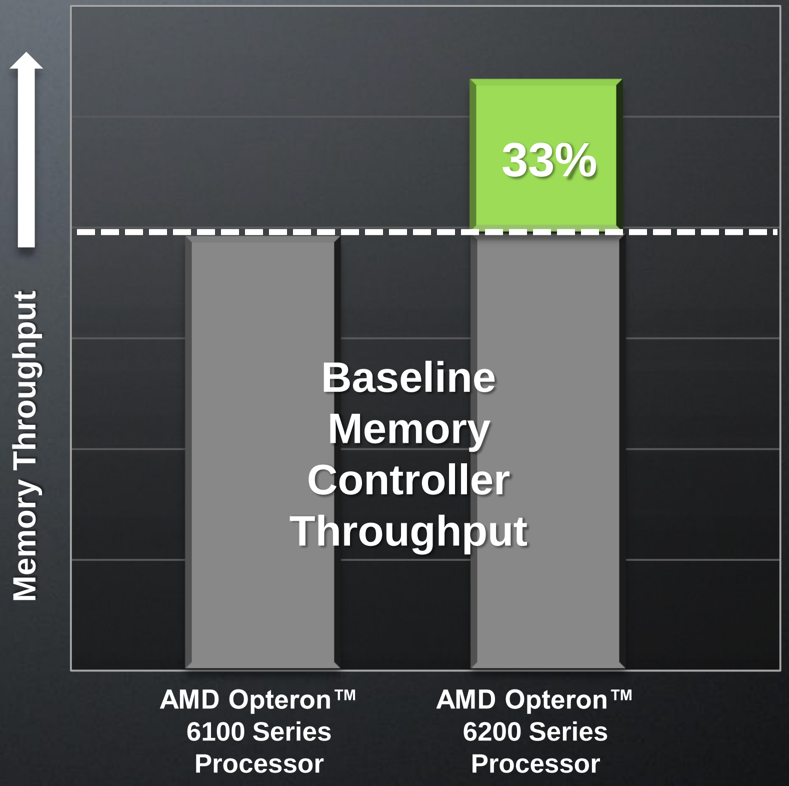
AMD Opteron™ 6200 Series Processors

	Linear Algebra	Fast Fourier Transforms (FFT)	Others	Compiler Support
ACML 5.0 (Aug 2011) *Alpha available NOW*	• SGEMM (single precision) • DGEMM (double precision) • L1 BLAS	• Complex-to-Complex (C-C) single precision FFTs	• Random Number Generators • AVX compiler switch for Fortran	• Alpha support for gcc 4.6 and Open64 4.2.5 • PGI 11.8 and ICC 12 added with ACML 5.0 production release • Cray to begin deployment of ACML with their compiler with ACML 5.0
ACML 5.1 (Dec 2011)	• CGEMM (complex single decision) • ZGEMM (complex double precision)	• Real-to-complex (R-C) single precision FFTs • Double precision C-C and R-C FFTs		All compilers listed for ACLM 5.0 will be supported

For additional information on ACML, go to:
http://developer.amd.com/libraries/acml/pages/default.aspx

UP TO 33% MEMORY THROUGHPUT INCREASE*

AMD Opteron™ 6200 Series Processors



- New redesigned Northbridge controller
- 1600 MHz DDR-3 support
- LR-DIMM support
- 1.25V LV-DDR3 support
- New memory power management features:
  - Aggressive power down
  - Partial channel power down
  - And memory power capping via

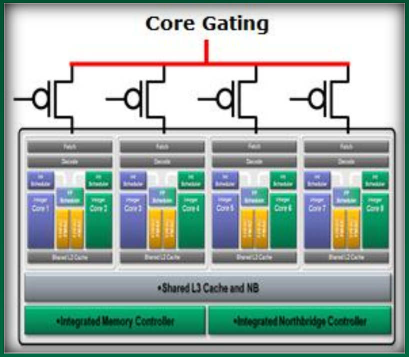
*Based on measurements by AMD labs as of 8/9/11. Comparison is AMD Opteron 6200 Series with DDR3-1600 vs. AMD Opteron 6100 Series with DDR3-1333. See substantiation section for config info.

POWER EFFICIENCY OVERVIEW

Consistent Power and Thermals

AMD Opteron™ 6200 Series fits into the same general thermal range as previous generation

Reduces processor power at idle by up to 46%*


Core Gating
Shared L3 Cache and NB
Integrated Memory Controller
Integrated Northbridge Controller

C6 power state

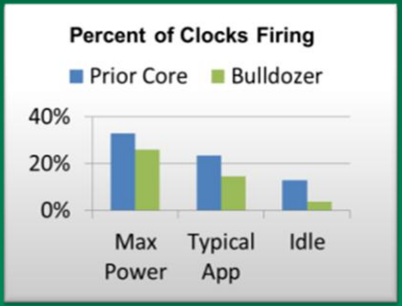
Shuts down clocks and power to idle cores

Enables more control for IT

TDP Power Cap

Set thermal design power (TDP) to meet power and workload demands for more flexible, denser deployments

Intelligent Circuit Design



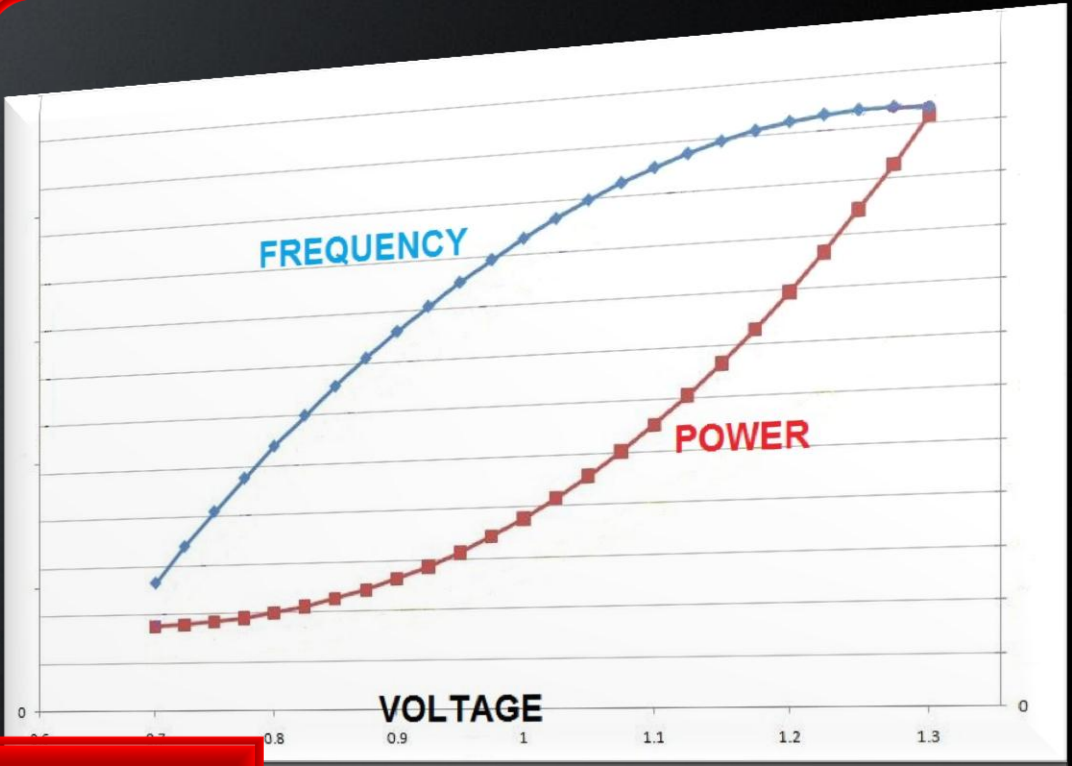
Minimizes the number of active transistors for lower power and better performance

*See processor power savings slide in substantiation section

POWER MANAGEMENT | P-states, AMD Turbo CORE

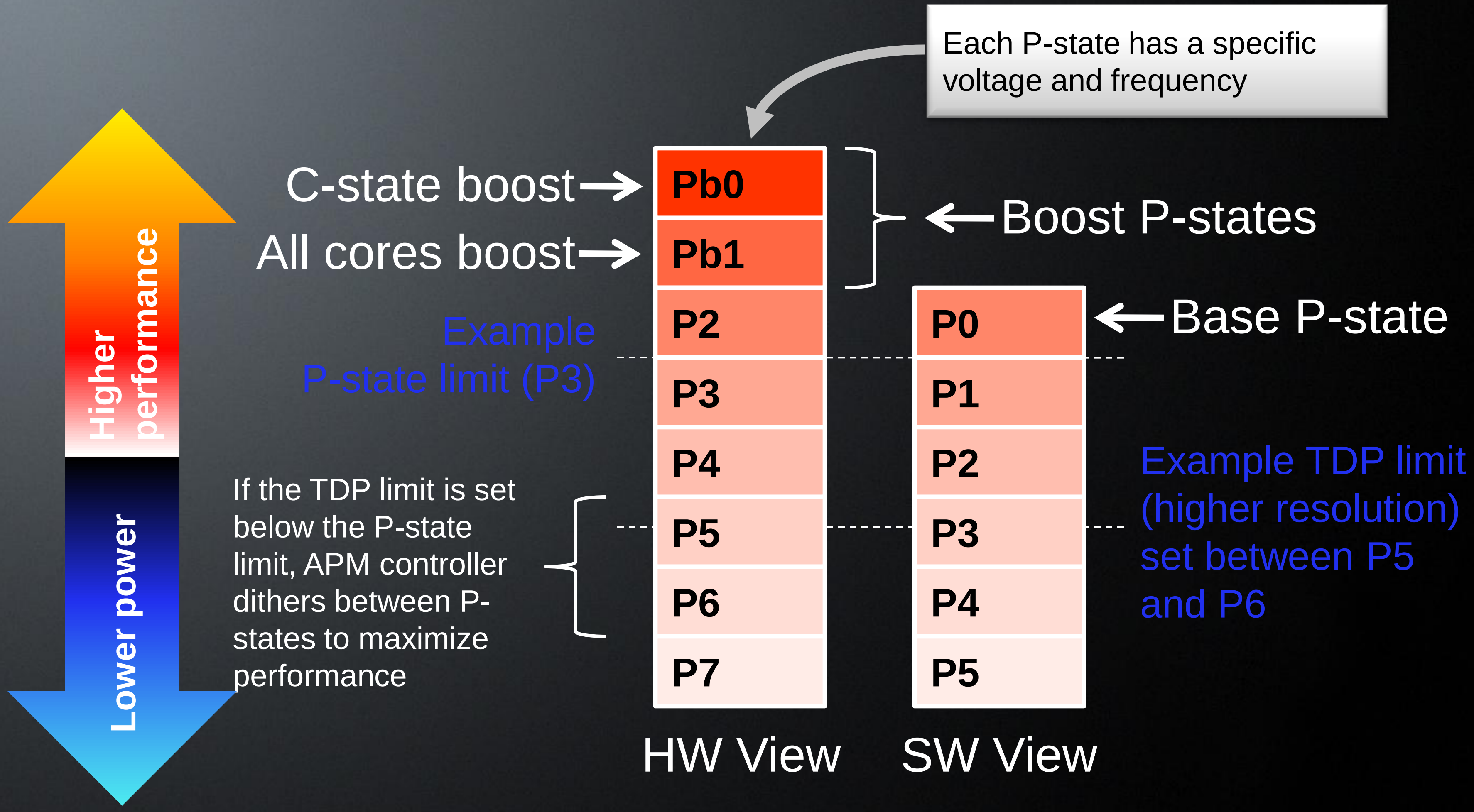
Core P-states specify multiple frequency and voltage points of operation

- Higher frequency P-states deliver greater performance but require higher voltage and thus more power
- The hardware and operating system vary which P-state a core is in to deliver performance as needed, but use lower frequency P-states to save power whenever possible



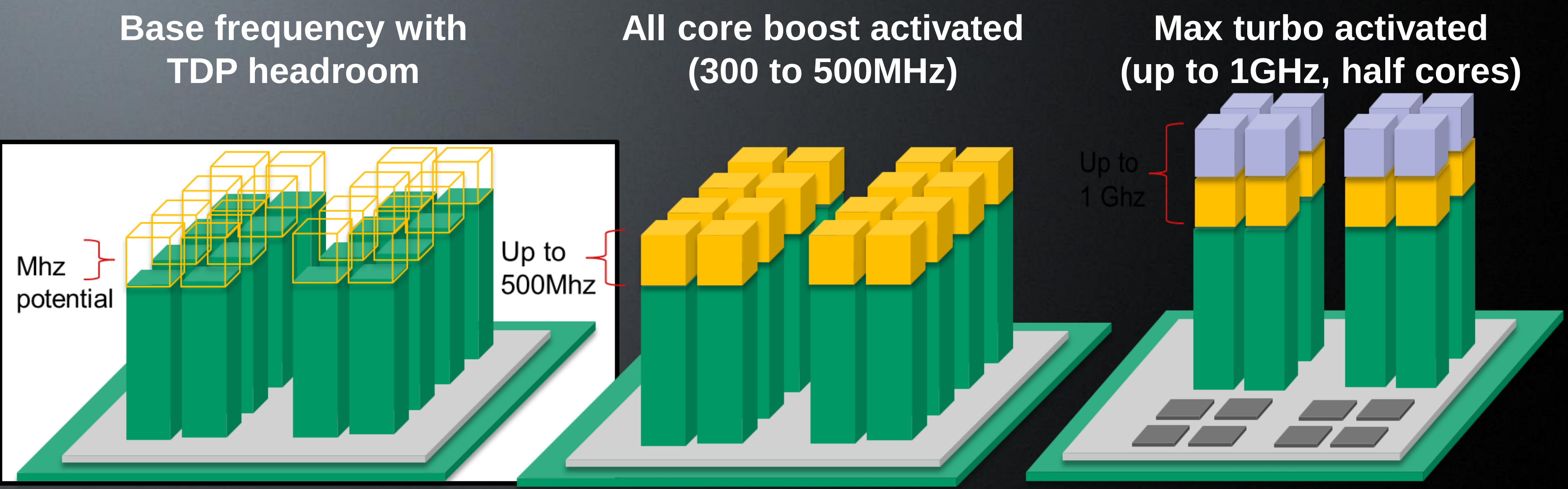
AMD Turbo CORE: when the processor is below its power/thermal limits the frequency and voltage can be boosted above the normal maximum and stay there until it gets back to the power/thermal limits

OVERVIEW OF P-STATES, APM, AND LIMITS


Each P-state has a specific voltage and frequency
C-state boost
All cores boost
Boost P-states
Base P-state
Example P-state limit (P3)
Higher performance
Lower power
Pb0
Pb1
P2
P3
P4
P5
P6
P7
P0
P1
P2
P3
P4
P5
Example TDP limit (higher resolution) set between P5 and P6
HW View
SW View

If the TDP limit is set below the P-state limit, APM controller dithers between P-states to maximize performance

AMD TURBO CORE TECHNOLOGY



Base frequency with TDP headroom
All core boost activated (300 to 500MHz)
Max turbo activated (up to 1GHz, half cores)
Mhz potential
Up to 500Mhz
Up to 1 Ghz

All Core Boost

When there is TDP headroom in a given workload, AMD Turbo CORE technology is automatically activated and can increase clock speeds by 300 to 500 MHz across **all cores.**

Max Turbo Boost

When a lightly threaded workload sends half the Bulldozer modules into C6 sleep state but also requests max performance, AMD Turbo CORE technology can increase clock speeds by up to 1 GHz across **half the cores.**

"BULLDOZER" POWER: TDP POWER CAP

Power Capping Power Thresholds[1]

Can allow the user to set the maximum processor power ceiling via BIOS[2] or APML[3].

What's the Benefit?

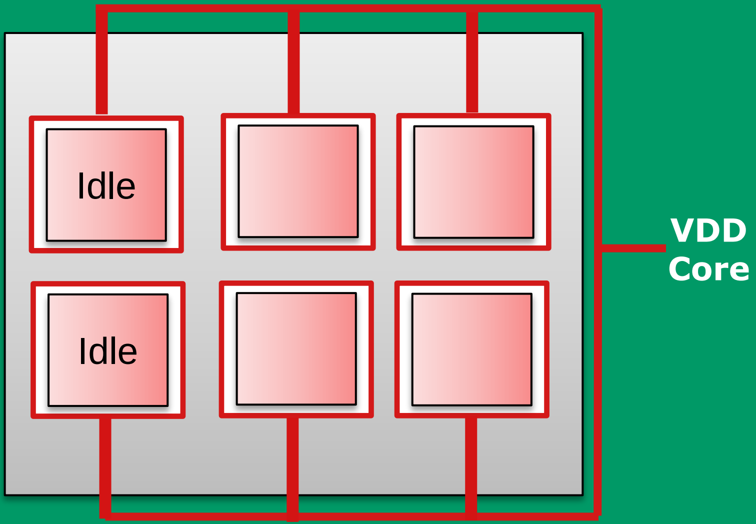
- More control over power settings
- Flexibility to set power limits without capping CPU frequencies[4]

[1]Planned support in "Bulldozer" processors
[2]For platforms where TDP power capping feature is enabled in the system BIOS
[3]For platforms that have designed in APML platform support
[4]TDP power capping can still allow the processor to operate a maximum specified frequency

REDUCING POWER LEAKAGE
ENHANCED NEAR ZERO POWER CORE STATE WITH “C6”

AMD Opteron 6100 & 4100 Series Processors

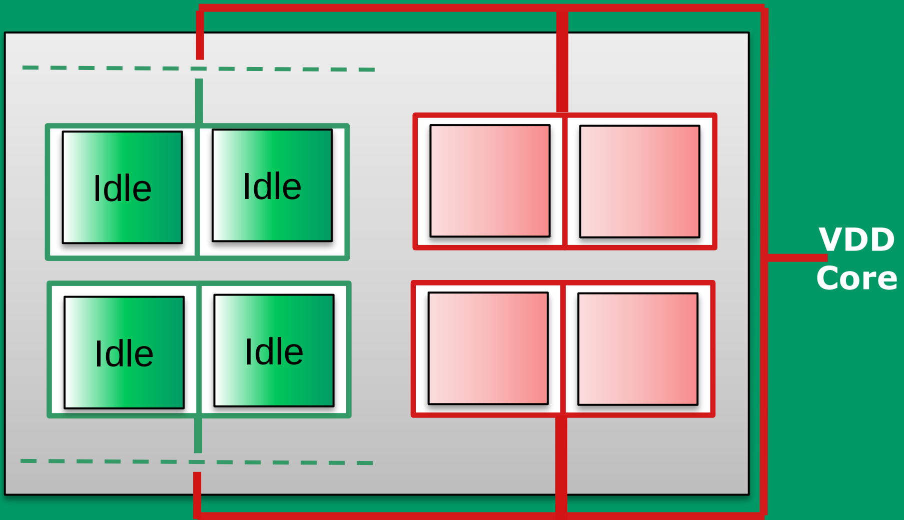
Single power plane, all cores powered at all times



Voltage is reduced but still applied to cores resulting in leakage / static power

‘Interlagos’ & ‘Valencia’ Processors

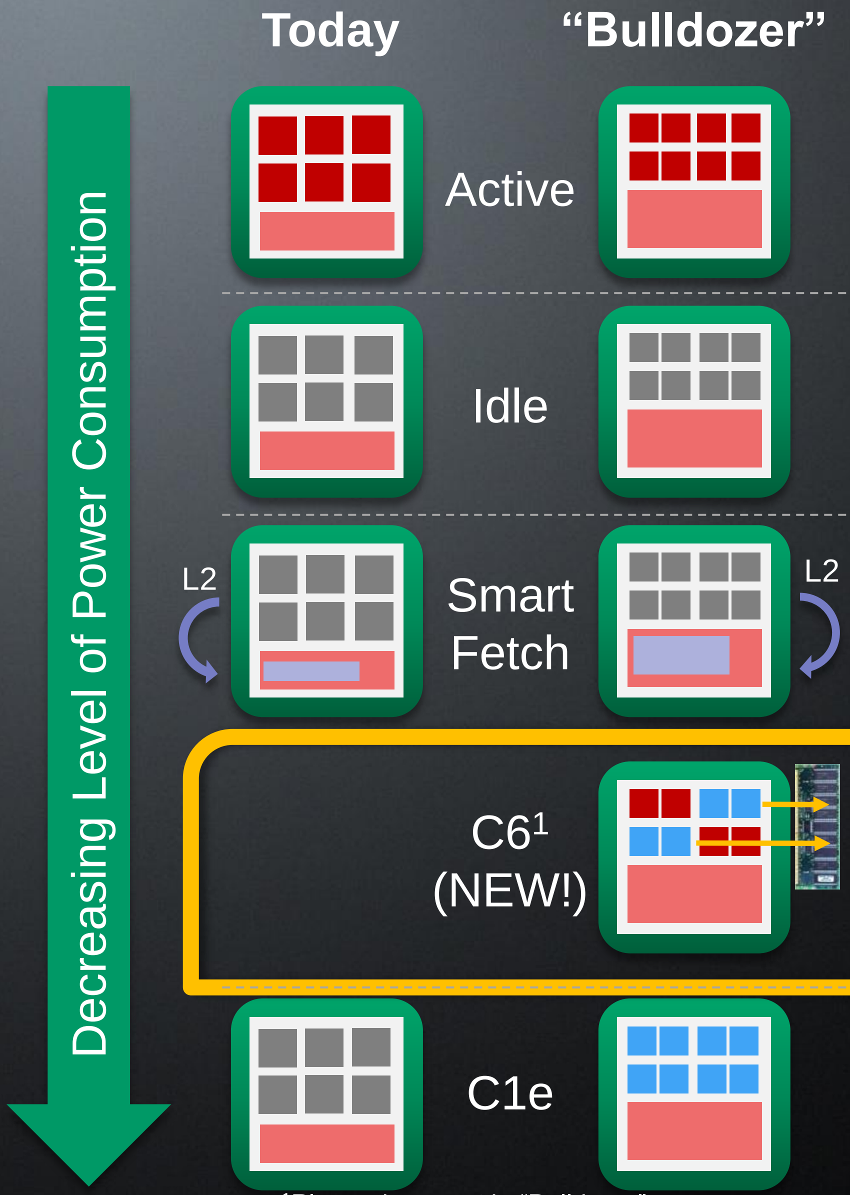
Single power plane, but each module can be turned on and off



Voltage is gated off to virtually remove all core static power / leakage

AMD

"BULLDOZER" POWER: C6 POWER STATE



All cores running workloads; core/module frequency can run independently to save power

No cores running workloads; core/module frequency reduced to 800MHz to save more power

After a set idle time L2 cache is flushed to L3, allowing cores to 'sleep' to save power while maintaining MP coherency

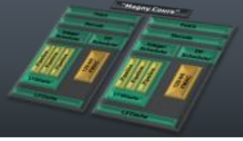
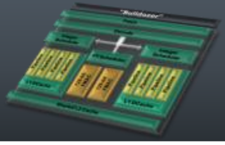
On "Bulldozer" any idle module can independently enter 'C6', gating power up to 95% for considerable power savings; module state is saved to DRAM

In addition to reducing memory and I/O power, 'C6' further reduces core power on "Bulldozer" vs. current core in C1e

[1]Planned support in "Bulldozer" processors

AMD

GENERATIONAL COMPARISONS

	AMD Opteron™ 6100 Series Processors 	AMD Opteron™ 6200 Series Processors 
Cores	8 or 12 core	4, 8, 12 or 16 core
Cache (L2 per core / L3 per die)	512KB / 6MB	2MB (shared between 2 cores) / 8MB
Memory Channels and speed	four; up to 1333MHz	four; up to 1600MHz
Floating point capability	128-bit FPU per core (FADD/FMUL)	128-bit dedicated FMAC per core or 256-bit AVX shared between 2 cores
Integer Issues Per Cycle	3	4
Turbo CORE Technology	No	Yes (+500MHz with all cores active)
Power (ACP)	65W, 80W, 105W	TBD (planned 65W, 80W, 105W)
New Instruction Sets		SSSE3, SSE 4.1/4.2, AVX, AES, FMA4, XOP, PCLMULQDQ
Power Gating	AMD CoolCore™, C1E	AMD CoolCore™, C1E, C6
Process / Die Size	45nm SOI	32nm SOI (smaller overall die size)
Performance		35% higher processing throughput*

The above reflect current expectations regarding features and performance and is subject to change.

*Based on pre-production measurements by AMD labs as of 8/9/11 and comparison of top bin AMD Opteron™ 6200 Series processors to top bin AMD Opteron 6100 Series processors

PERFORMANCE EFFICIENCY TODAY: AMD'S HPC PRODUCT PORTFOLIO

Energy efficient CPU and discrete GPU processors focused on addressing the most demanding HPC workloads

Multi-core x86 Processors

- **Outstanding Performance**
- **Superior Scalability**
- **Enhanced Power Efficiency**

ATI FirePro™ Professional Graphics

- **3D Accelerators For Visualization**
- **Full support for GPU computation with OpenCL**

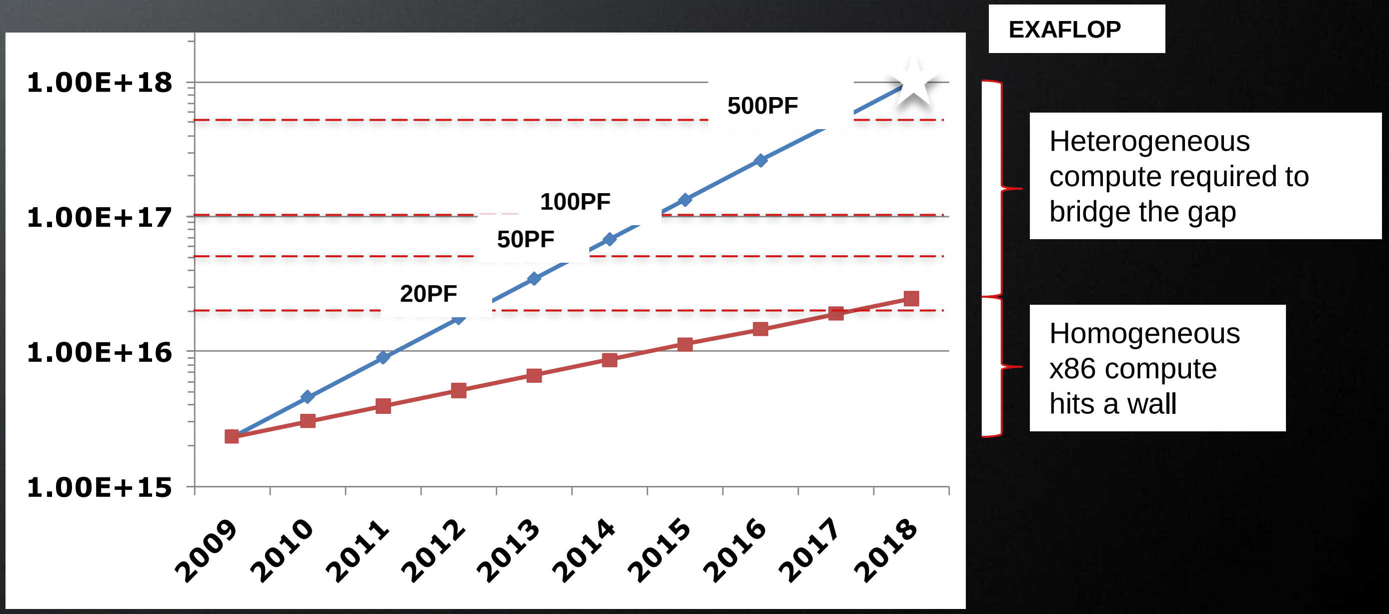
AMD Accelerated Parallel Processing TECHNOLOGY

AMD FireStream™ GPU Accelerators

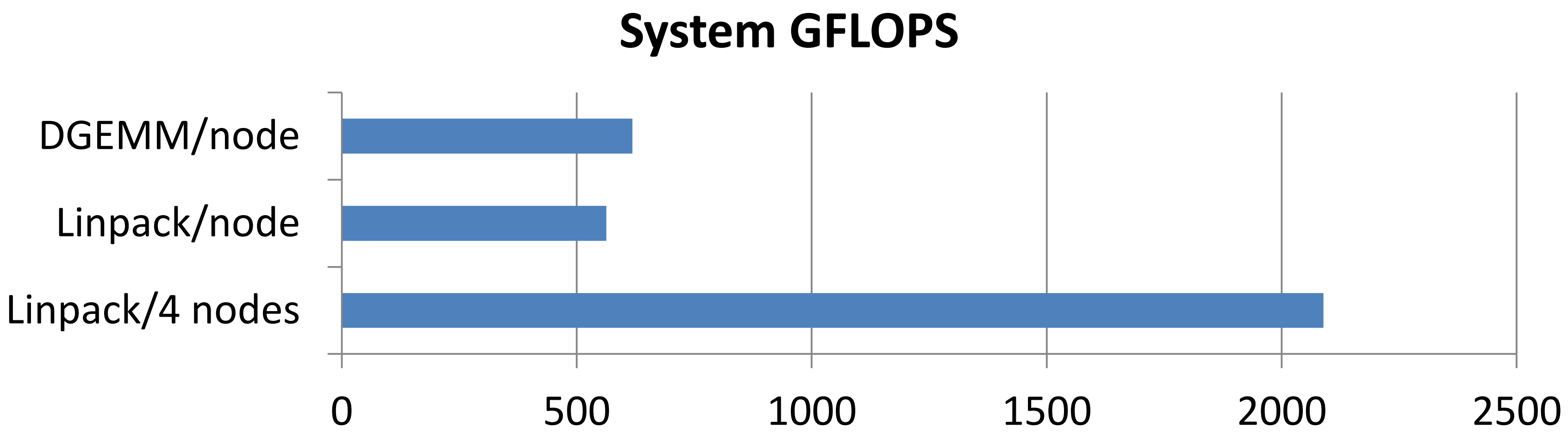
- **Optimized for server integration**
- **Single-slot and dual-slot form factors**
- **Industry standard OpenCL SDK**

A CASE FOR SERVER FUSION - EXASCALE

- Current trajectory puts traditional x86 computing at just over 20Pflops by 2018
- A data center that could achieve an exaflop in 2018 using only x86 processors would consume over 3TW
- To achieve exascale capability by 2018, x86 performance would need to increase by 2x each year, starting in 2010



HIGH EFFICIENCY LINPACK IMPLEMENTATION ON AMD MAGNY COURS + AMD 5870 GPU



- GPU DPFP Peak: 544 GFLOPS
- GPU DGEMM kernel: 87% of Peak

2.5 GFLOPS/W

HPL code:

http://code.compeng.uni-frankfurt.de/

- Node DPFP Peak: 745.6 GFLOPS
- Linpack efficiency: 75.5% of Peak
- Linpack scaling across 4 nodes: 70% of Peak

HIC for FAIR Helmholtz International Center

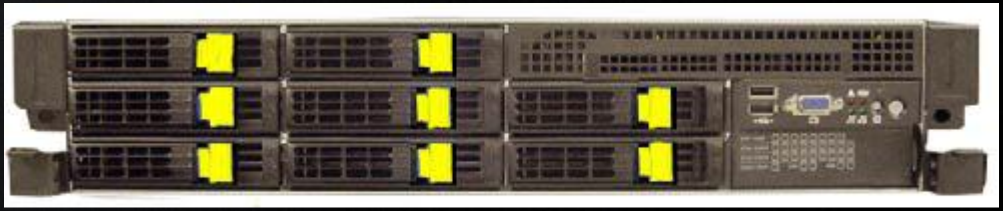
GOETHE UNIVERSITÄT FRANKFURT AM MAIN

FIAS Frankfurt Institute for Advanced Studies

WORLD'S FIRST HPC CLUSTER BASED ON FUSION APUS

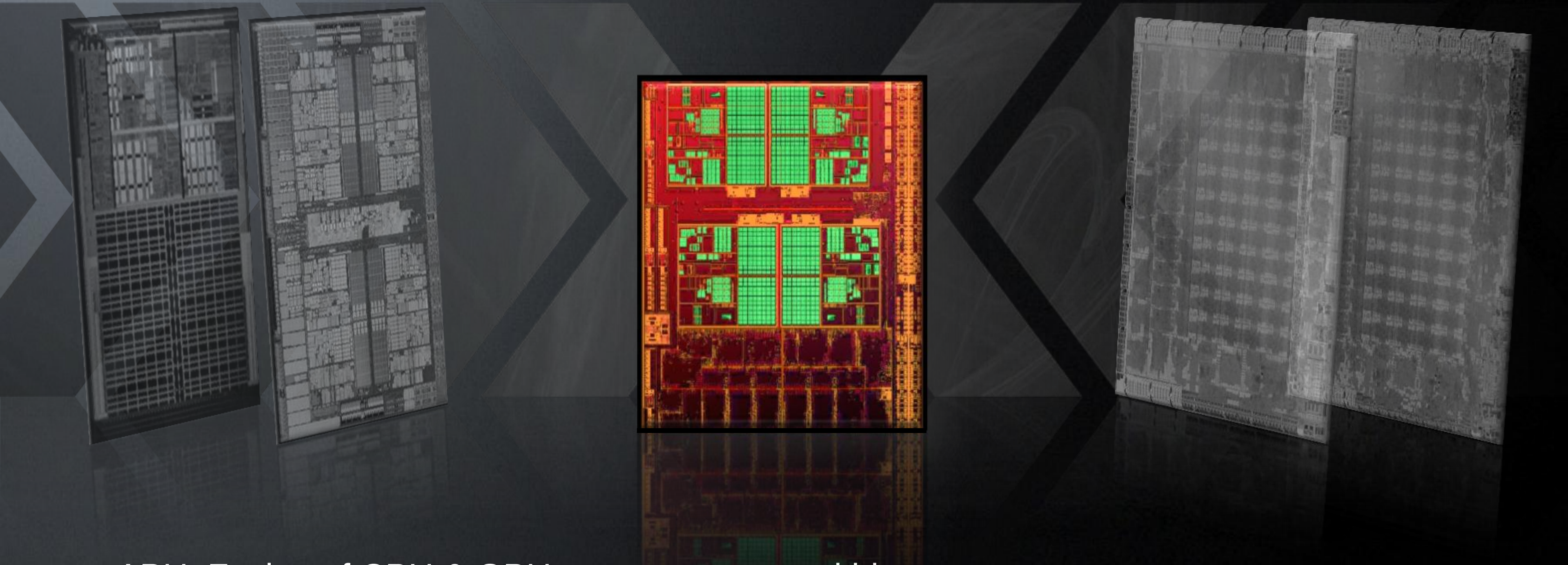
Penguin Computing has successfully installed the world's first HPC cluster powered by AMD accelerated processing units (APUs) at Sandia National Labs in Albuquerque, New Mexico.

The Altus A2A00 system comprises 104 servers powered by A-series Fusion Llano APUs (one chip per server) with four x86 cores and 320/400 stream processors that are interconnected through a QDR Infiniband fabric. It delivers a theoretical peak performance of **59.6TFLOPs**. The Altus 2A00 was specifically designed by Penguin Computing, in partnership with AMD, to support the AMD Fusion APU architecture. It is the world's first Fusion APU system in a rack mountable chassis in a 2U form factor.



Penguin Altus 2A00
Leverage AMD's APU architecture for HPC

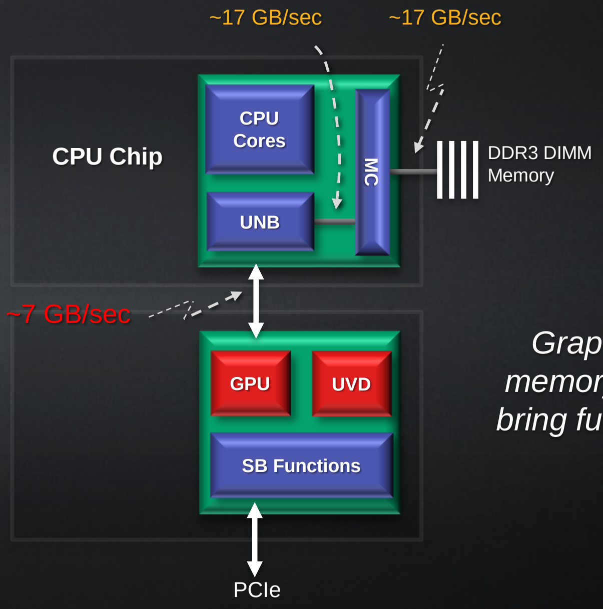
NOW THE AMD FUSION™ ERA OF COMPUTING BEGINS



- APU: Fusion of CPU & GPU compute power within one processor
- High-bandwidth I/O

GRAPHICS AND MEDIA PROCESSING EFFICIENCY IMPROVEMENTS

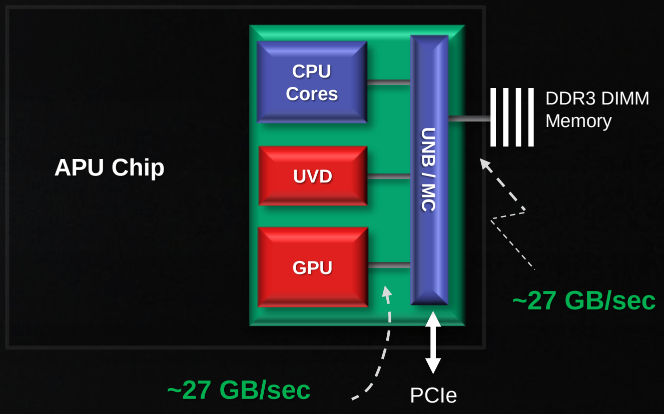
2010 IGP-based Platform



Graphics requires memory bandwidth to bring full capabilities to life

Bandwidth pinch points and latency hold back the GPU capabilities

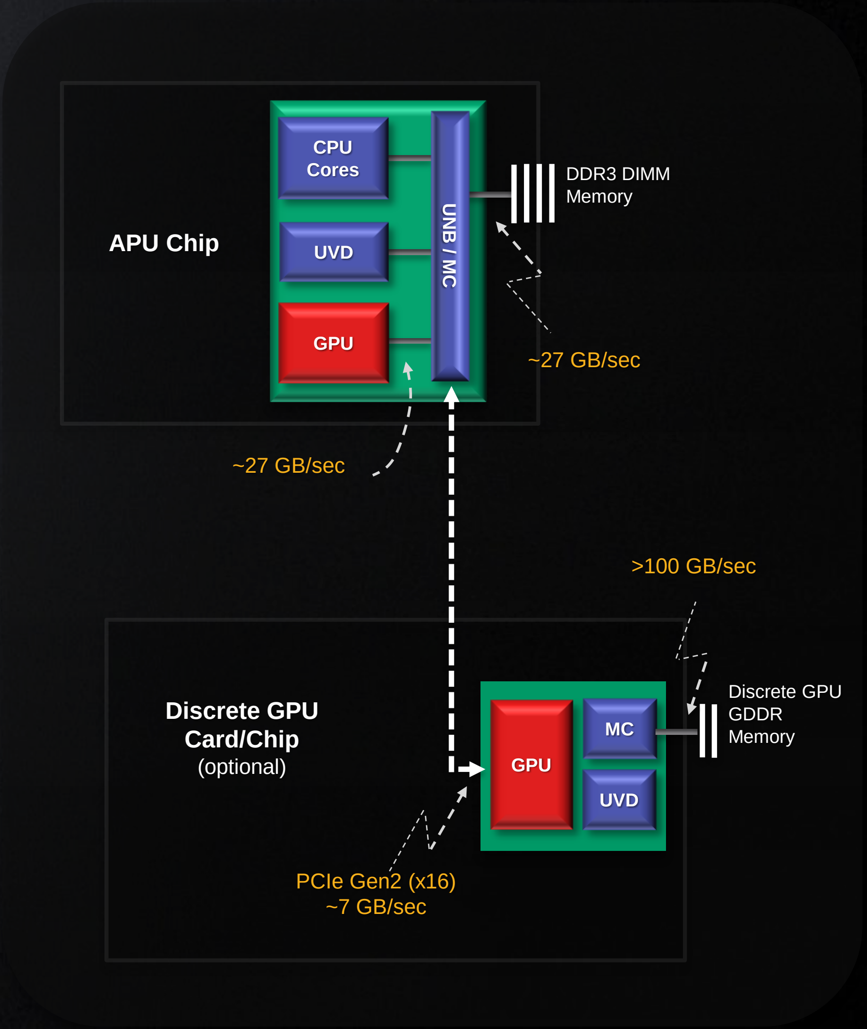
2011 APU-based Platform



- 3X bandwidth between GPU and memory
- Even the same sized GPU is substantially more effective in this configuration
- Eliminate latency and power associated with the extra chip crossing
- Substantially smaller physical foot print

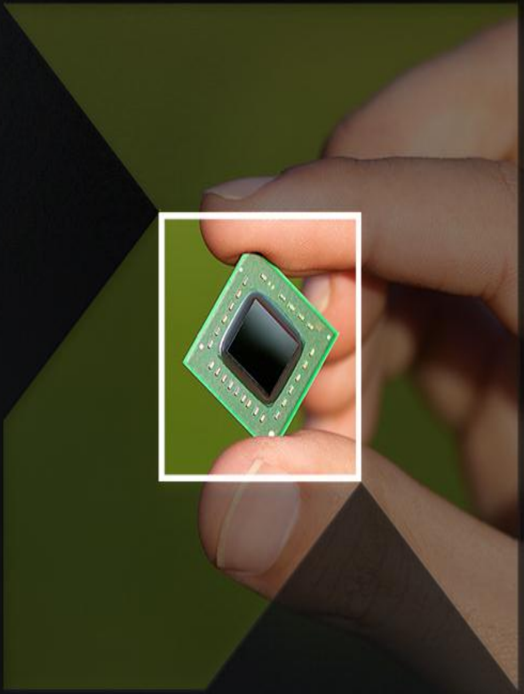
ACCELERATED DATA PARALLEL PROCESSING CAPABILITIES

- APU bandwidth enhancements not only improve traditional graphics, but also data parallel compute effectiveness
- Both GPUs cooperate on graphics and compute
- OpenCL 1.1 and DirectCompute compliant in both the APU and the optional discrete GPU
- AMD Fusion Experience Fund is helping to fuel the application developer community



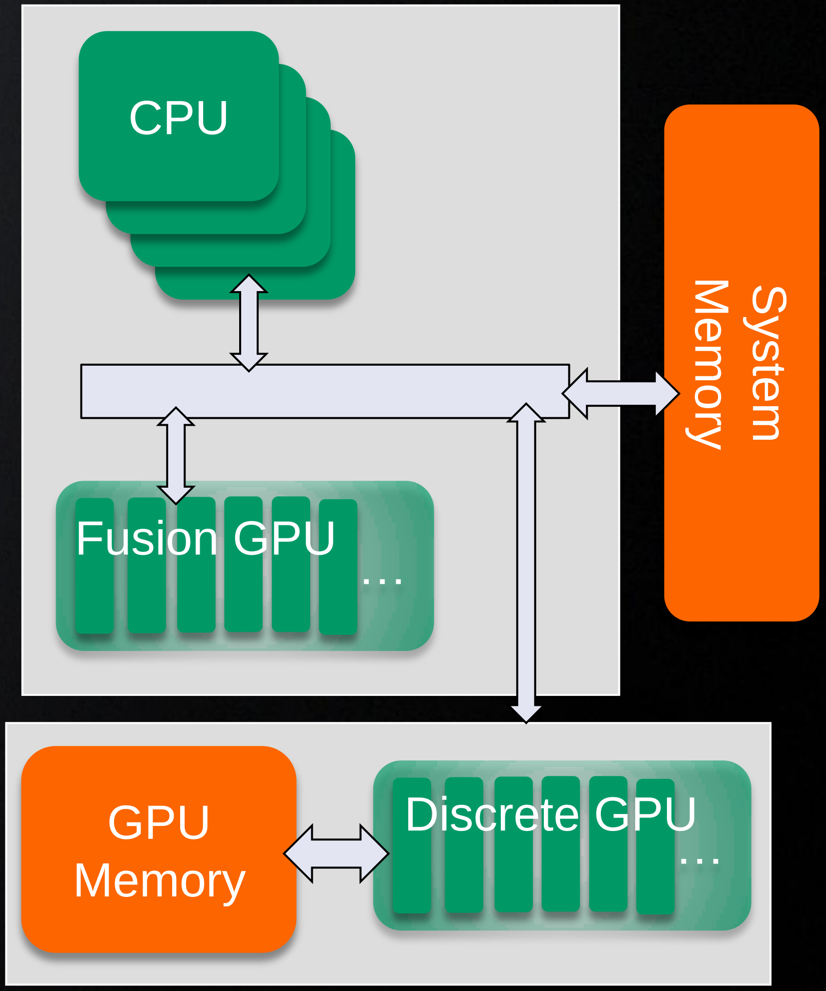
INTRODUCING OPENCL™

The open standard for parallel programming across heterogeneous processors



IT'S A HETEROGENEOUS WORLD

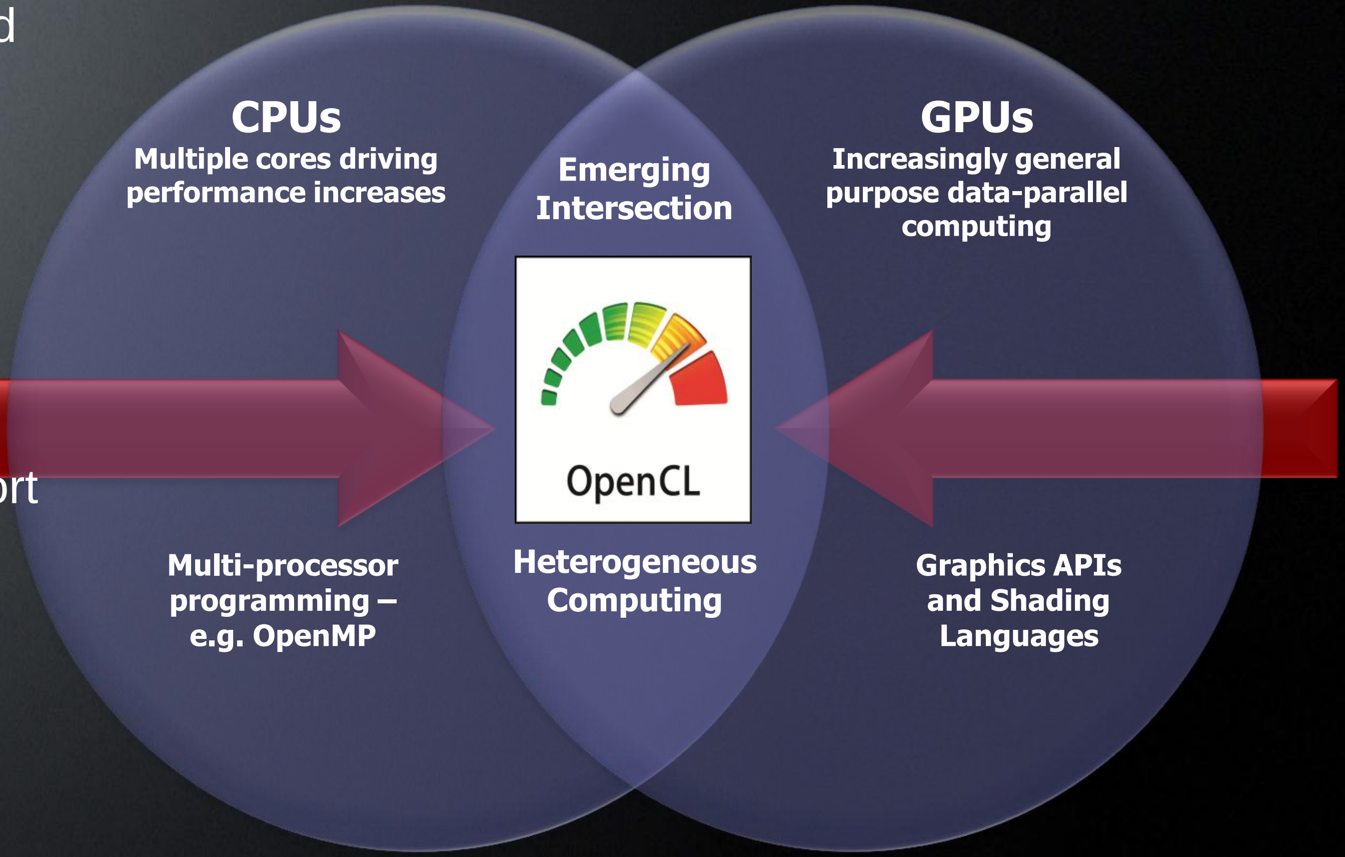
- Heterogeneous computing
  – The new normal
- Many CPU's – 2, 4, 8, …
- Very many GPU processing elements – 100's
- Different vendors, configurations, architectures
- The multi-million dollar question
  – How do you avoid developing and maintaining different source code versions?



WHAT IS OPENCL™

- Industry Standard
- Open Standard
- Cross Platform
- Multi-Vendor

- Royalty Free
- Broad ISV Support

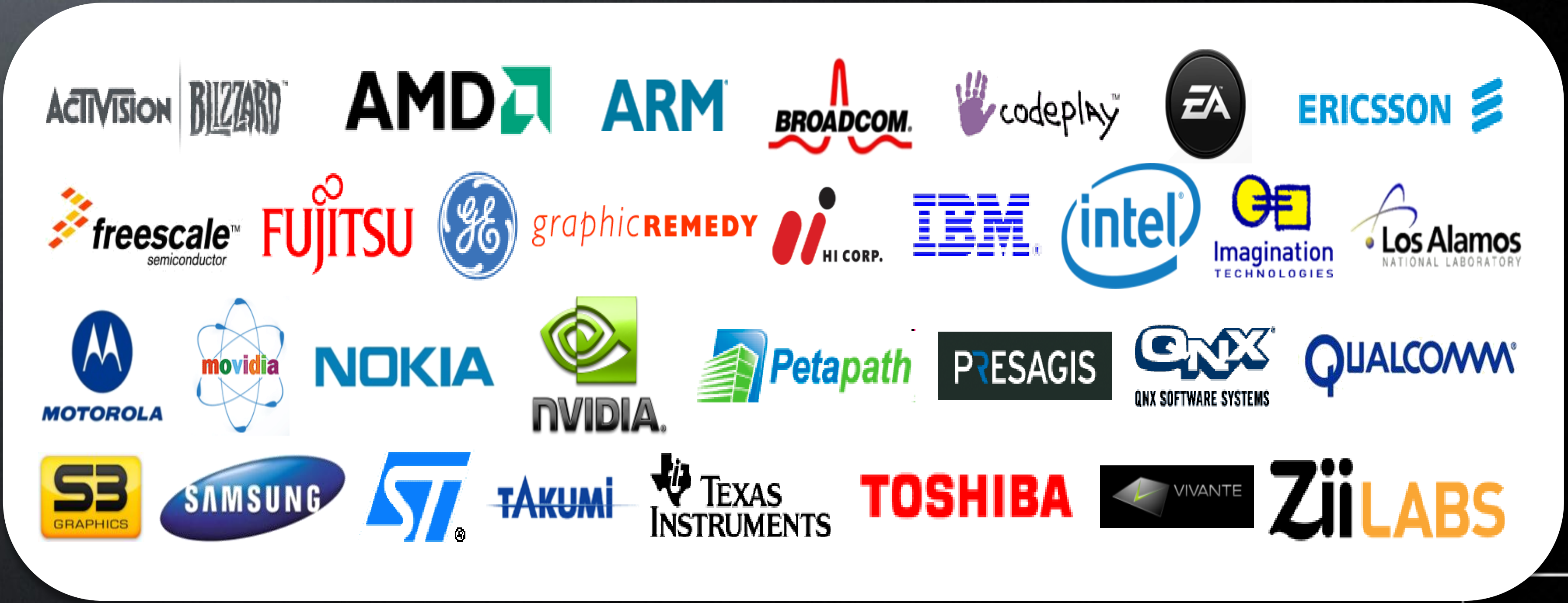

CPUs
Multiple cores driving performance increases
Emerging Intersection
GPUs
Increasingly general purpose data-parallel computing
OpenCL
Multi-processor programming – e.g. OpenMP
Heterogeneous Computing
Graphics APIs and Shading Languages

OpenCL™ is a programming framework for heterogeneous compute resources

Source Khronos

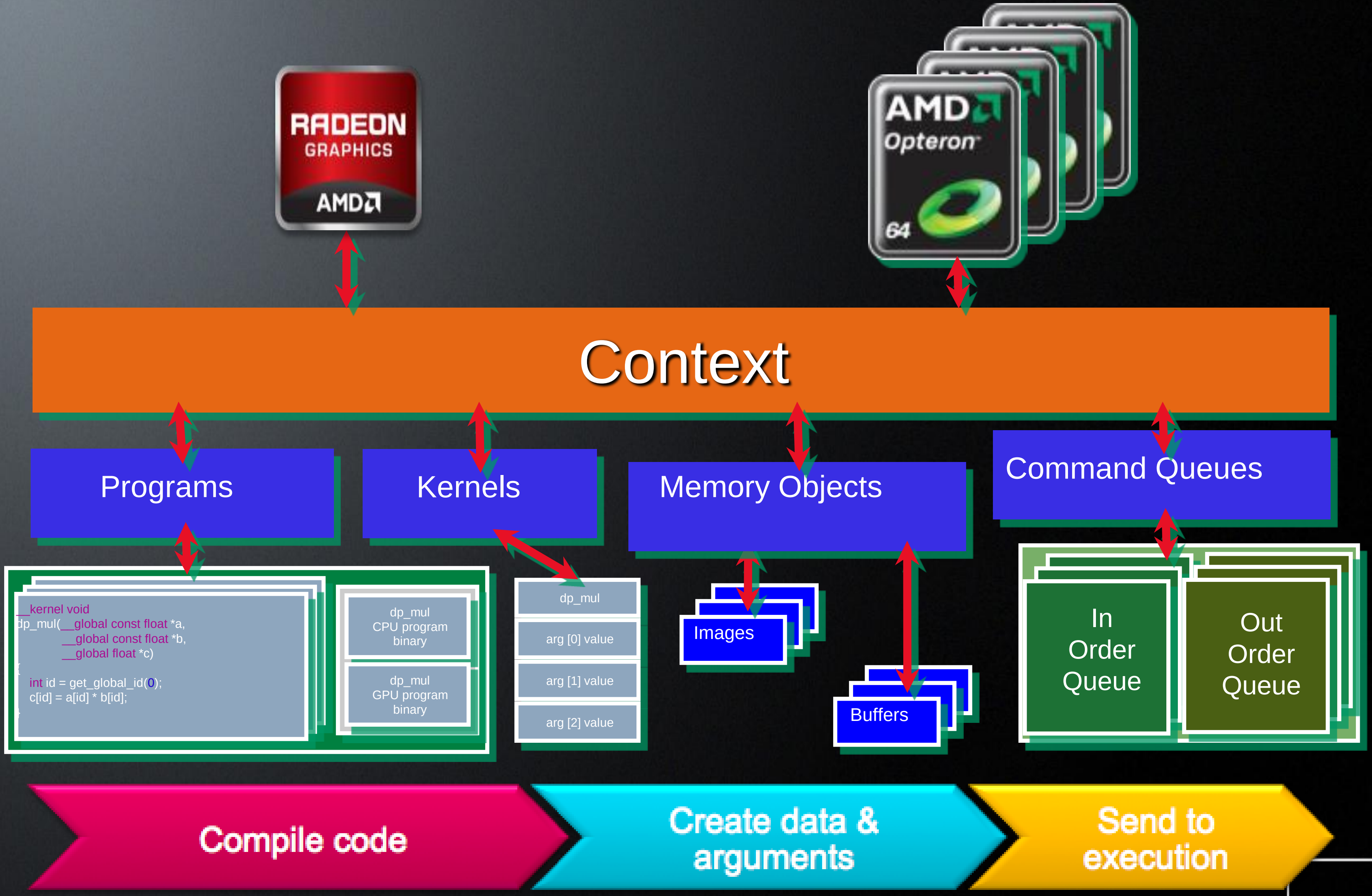
OPENCL™ WORKING GROUP

- Initial proposal made by Apple
  - A broad diversity of industry perspectives
  - Processor vendors, application developers, system OEMs, tool vendors, …



Source Khronos

OPENCL™ FRAMEWORK



```
__kernel void
dp_mul(__global const float *a,
       __global const float *b,
       __global float *c)
{
  int id = get_global_id(0);
  c[id] = a[id] * b[id];
}
```

HELLO WORLD - HOST PROGRAM

```
// create the OpenCL context on a GPU device
cl_context = clCreateContextFromType(0,
   CL_DEVICE_TYPE_GPU, NULL, NULL, NULL);


// get the list of GPU devices associated with context
clGetContextInfo(context, CL_CONTEXT_DEVICES, 0,
                                          NULL, &cb);
devices = malloc(cb);
clGetContextInfo(context, CL_CONTEXT_DEVICES, cb,
   devices, NULL);


// create a command-queue
cmd_queue = clCreateCommandQueue(context, devices[0],
   0, NULL);


memobjs[0] = clCreateBuffer(context,CL_MEM_WRITE_ONLY,

   sizeof(cl_char)*strlen("Hello World", NULL,
                                              NULL);
// create the program
program = clCreateProgramWithSource(context, 1,
   &program_source, NULL, NULL);
```

```
// build the program
err = clBuildProgram(program, 0, NULL, NULL, NULL,
                                           NULL);

// create the kernel
kernel = clCreateKernel(program, "vec_add", NULL);

// set the args values
err  = clSetKernelArg(kernel, 0, (void *) &memobjs[0],
                             sizeof(cl_mem));

// set work-item dimensions
global_work_size[0] = strlen("Hello World");;

// execute kernel
err = clEnqueueNDRangeKernel(cmd_queue, kernel, 1,
   NULL, global_work_size, NULL, 0, NULL, NULL);

// read output array
err = clEnqueueReadBuffer(cmd_queue, memobjs[0],
   CL_TRUE, 0, strlen("Hello World") *sizeof(cl_char),
   dst, 0, NULL, NULL);
```

HELLO WORLD - HOST PROGRAM

```
// create the OpenCL context on a GPU device
cl_context = clCreateContextFromType(0,
```

Define platform and queues

```
// get the list of GPU devices associated with context
clGetContextInfo(context, CL_CONTEXT_DEVICES, 0,
                                        NULL, &cb);
devices = malloc(cb);
clGetContextInfo(context, CL_CONTEXT_DEVICES, cb,
   devices, NULL);

// create a command-queue
cmd_queue = clCreateCommandQueue(context, devices[0],
   0, NULL);
```

Define Memory objects

```
                                              READ_ONLY |
   CL_MEM_COPY_HOST_PTR, sizeof(cl_char)*strlen("Hello
   World"), srcA, NULL);}
// create the program
program = clCreateProgramWithSource(context, 1,
   &program_source, NULL, NULL);
```

Create the program

```
//
er                                          , NULL, NULL,
   NULL);
```

Build the program

```
/
k                                              LL);
```

Create and setup kernel

```
// set the args values
err  = clSetKernelArg(kernel, 0, (void *) &memobjs[0],
                                 sizeof(cl_mem));

// set work-item dimensions
global_work_size[0] = n;

// execute kernel
err = clEnqueueNDRangeKernel(cmd_queue, kernel, 1,
   NULL, global_work_size, NULL, 0, NULL, NULL);

// read output array
err = cl                                   , CL_TRUE,
   0, n*                                   ;
```

Execute the kernel

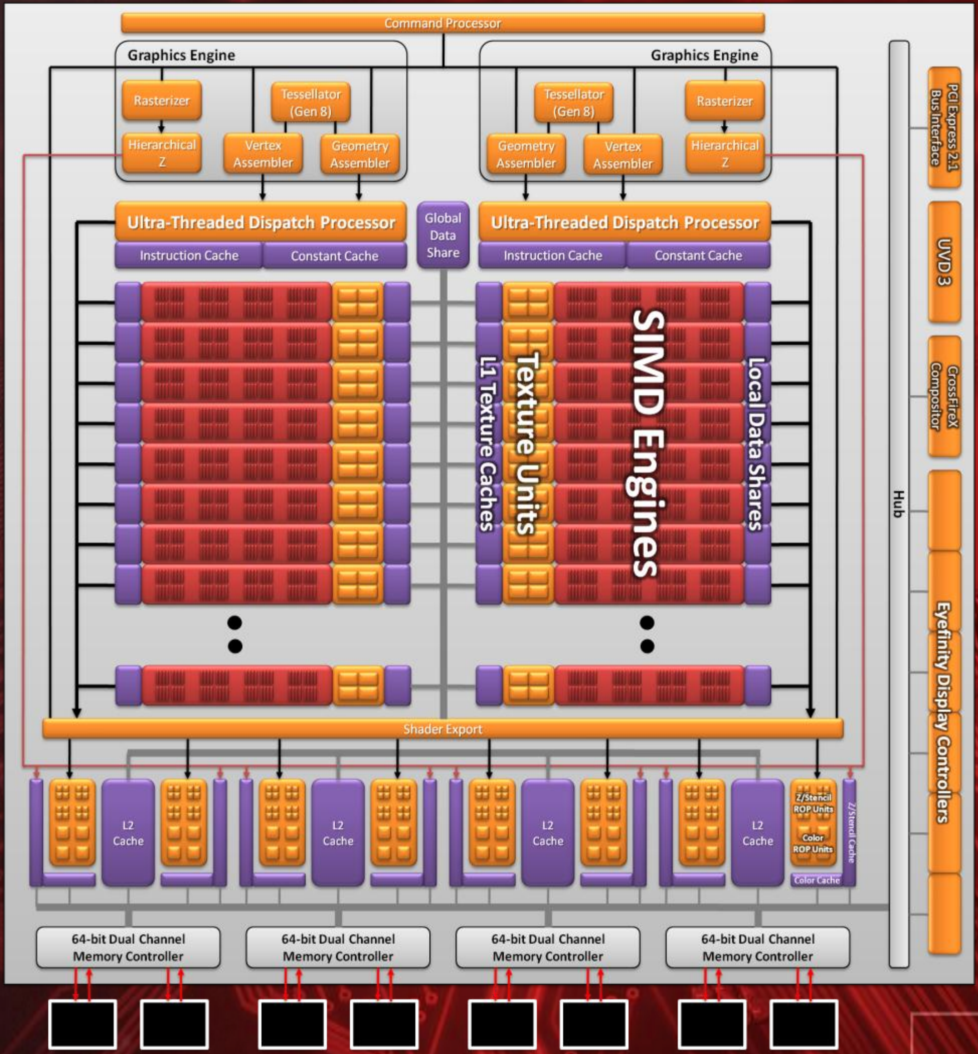
Read results on the host

AMD RADEON™ HD 6900 SERIES

- Dual graphics engines
- VLIW4 core architecture
- Fast 256-bit GDDR5 memory interface
  - Up to 5.5 Gbps

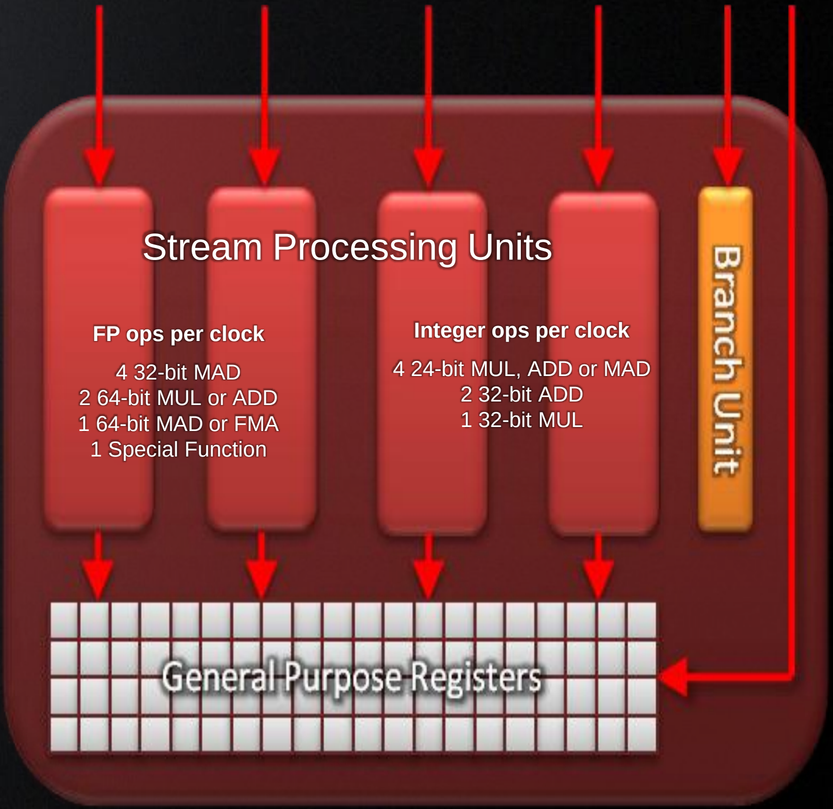
<u>HD 6970</u>

- 1536 Stream Processors
- 2.7 TFLOPs SP
- 683 GFLOPs DP



CORE DESIGN

- VLIW4 thread processors
  - 4-way co-issue
  - All stream processing units have equal capabilities
    - Special functions (transcendentals) occupy 3 of 4 issue slots



EXPOSING PARALLELISM

C function

```
for (int i = 0; i < 24; i++)
{
            Y[i] = a*X[i] + Y[i];
}
```

- Serial execution, one iteration after the other

EXPOSING PARALLELISM

C function

```
~~for (int i = 0; i < 24; i++)~~
{
        Y[i] = a*X[i] + Y[i];
}
```

- Serial execution, one iteration after the other

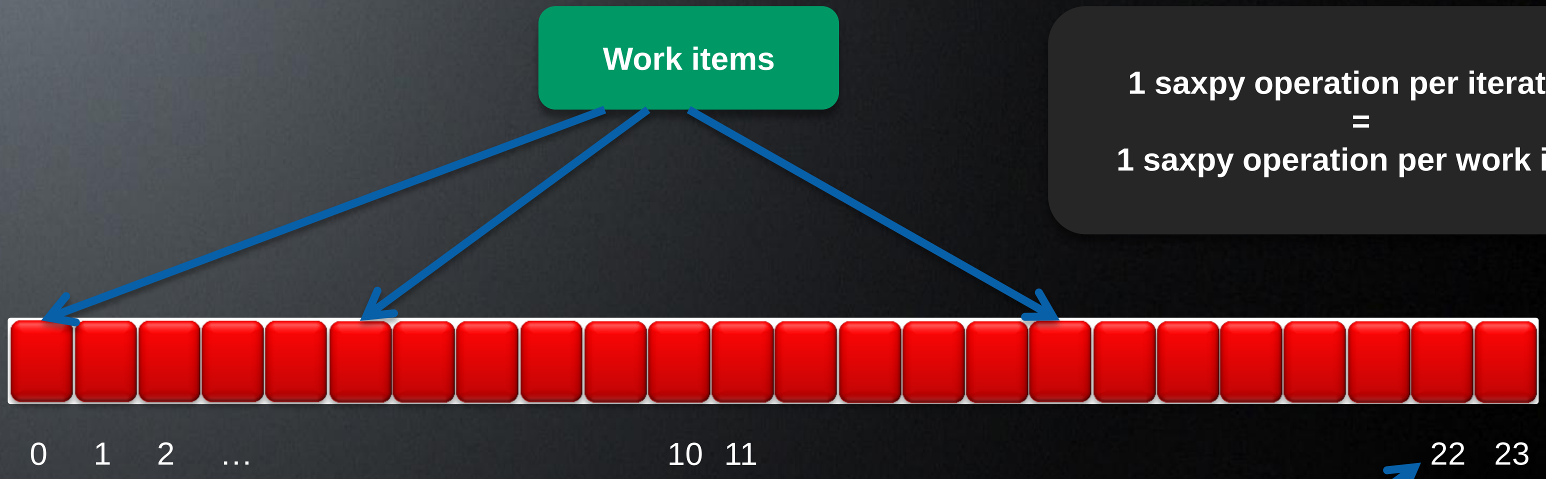
OpenCL kernel

```
__kernel void
saxpy(const __global float * X,
            __global float * Y,
                 const float a)
{
        uint i = get_global_id(0);
        Y[i] = a* X[i] + Y[i];
}
```

- Parallel execution, multiple iterations at the same time

WORK ITEM

- Think of work item as a parallel “thread” of execution



1 saxpy operation per iteration
=
1 saxpy operation per work item

~~`for (int i = 0; i < 24; i++)`~~

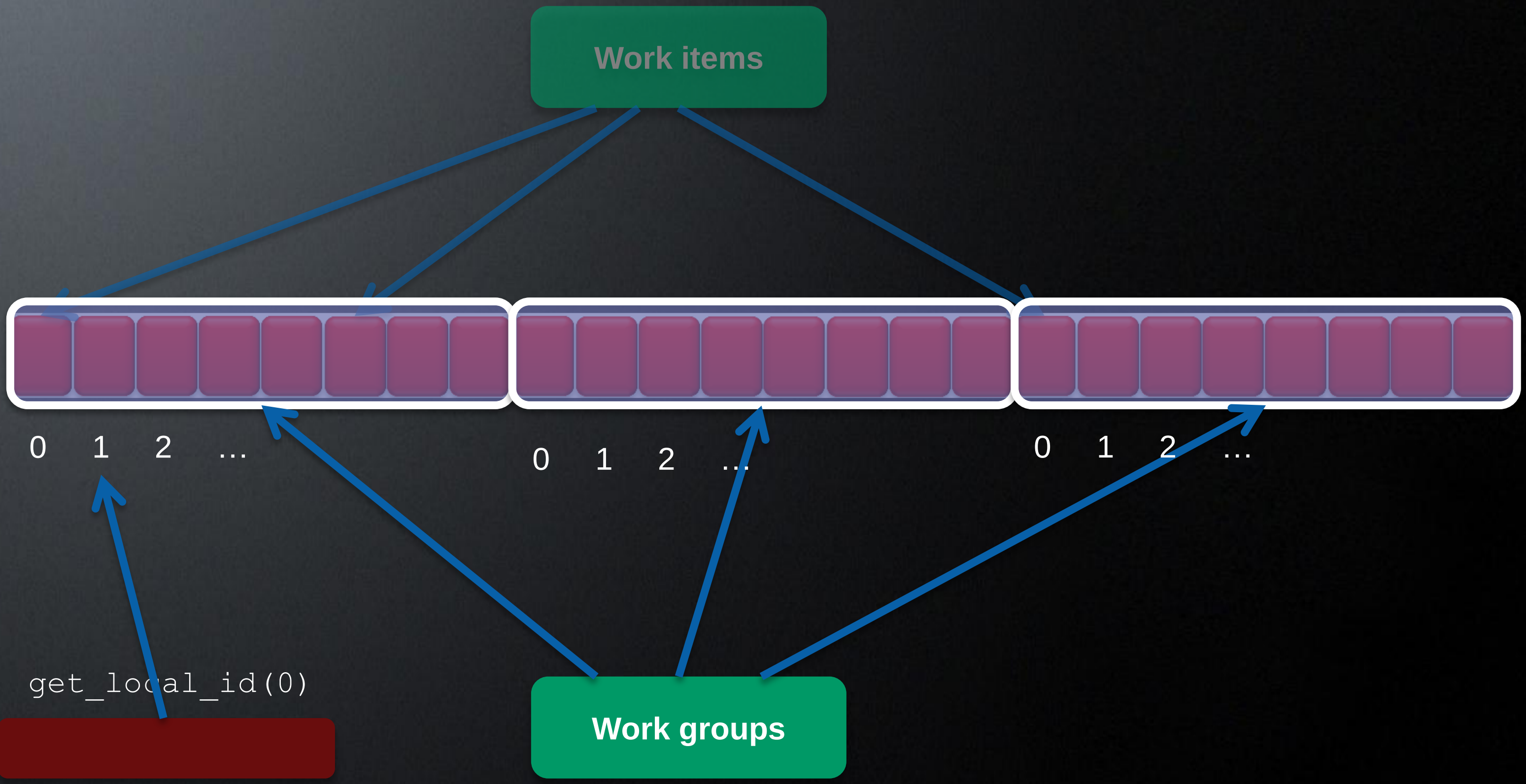
```
{
        Y[i] = a*X[i] + Y[i];
}
```

```
{
      uint i = get_global_id(0);
      Y[i] = a* X[i] + Y[i];
}
```

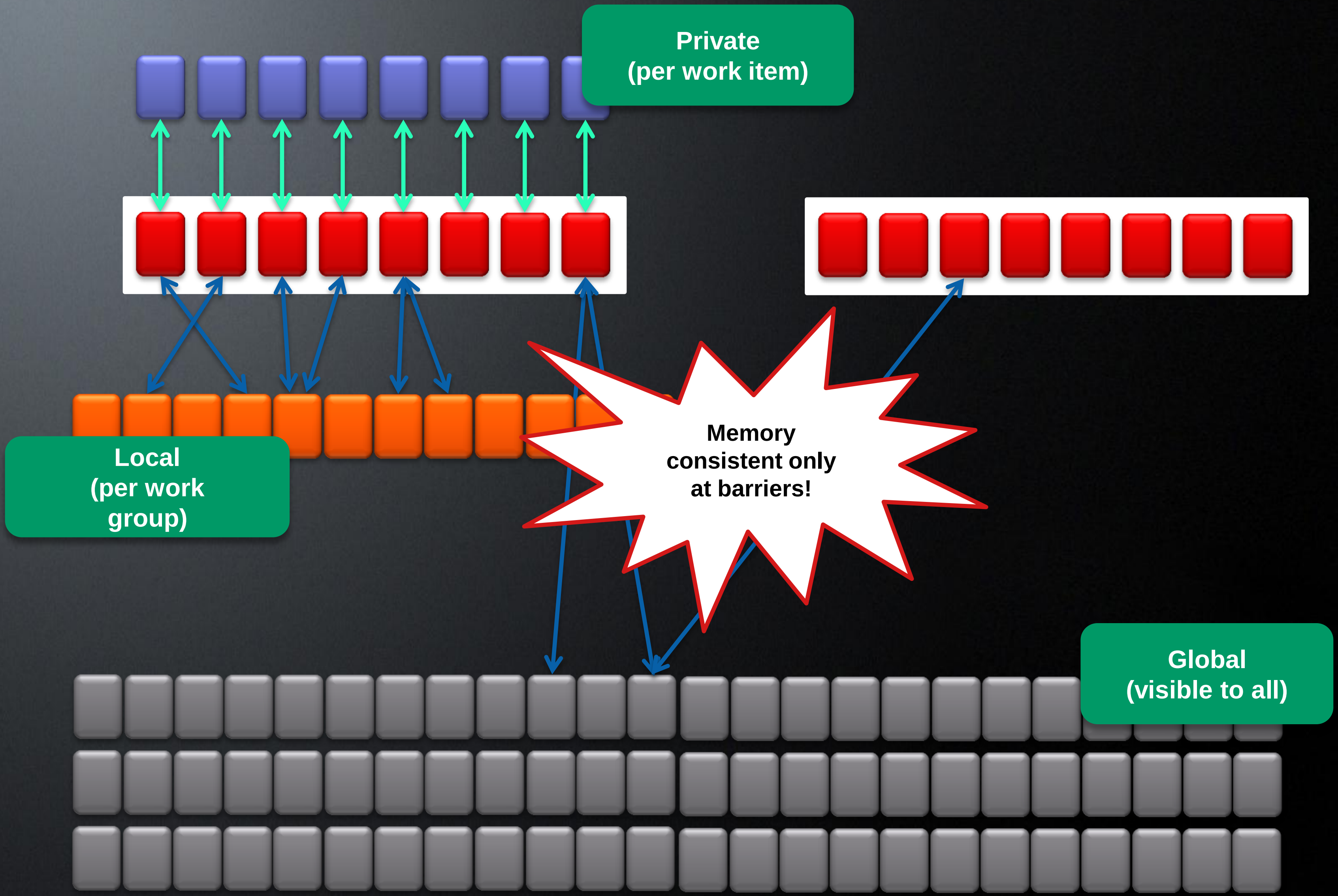
WORK GROUP

Divide the execution domain into groups

Can exchange data and synchronize inside a group

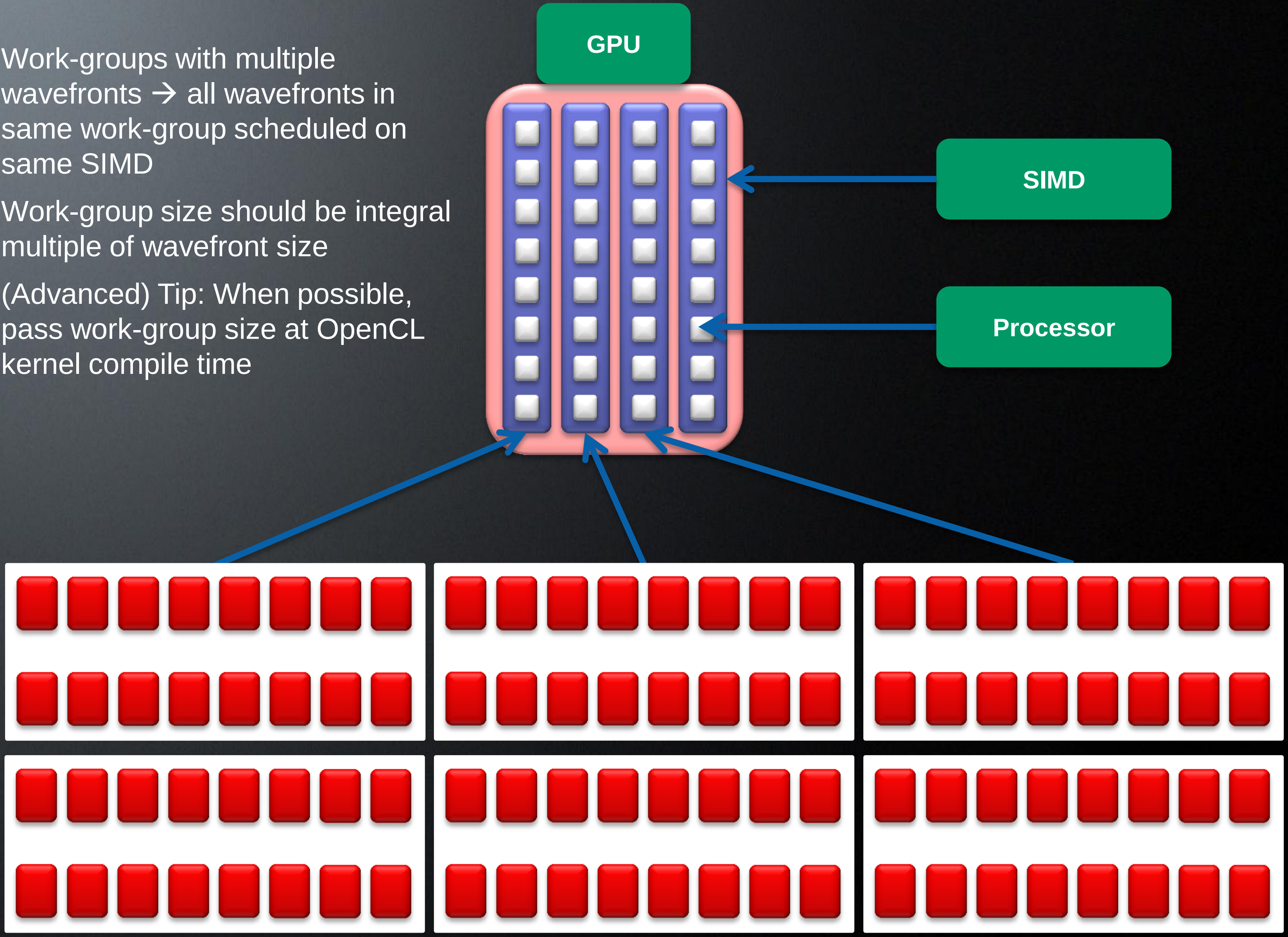


MEMORY SPACES

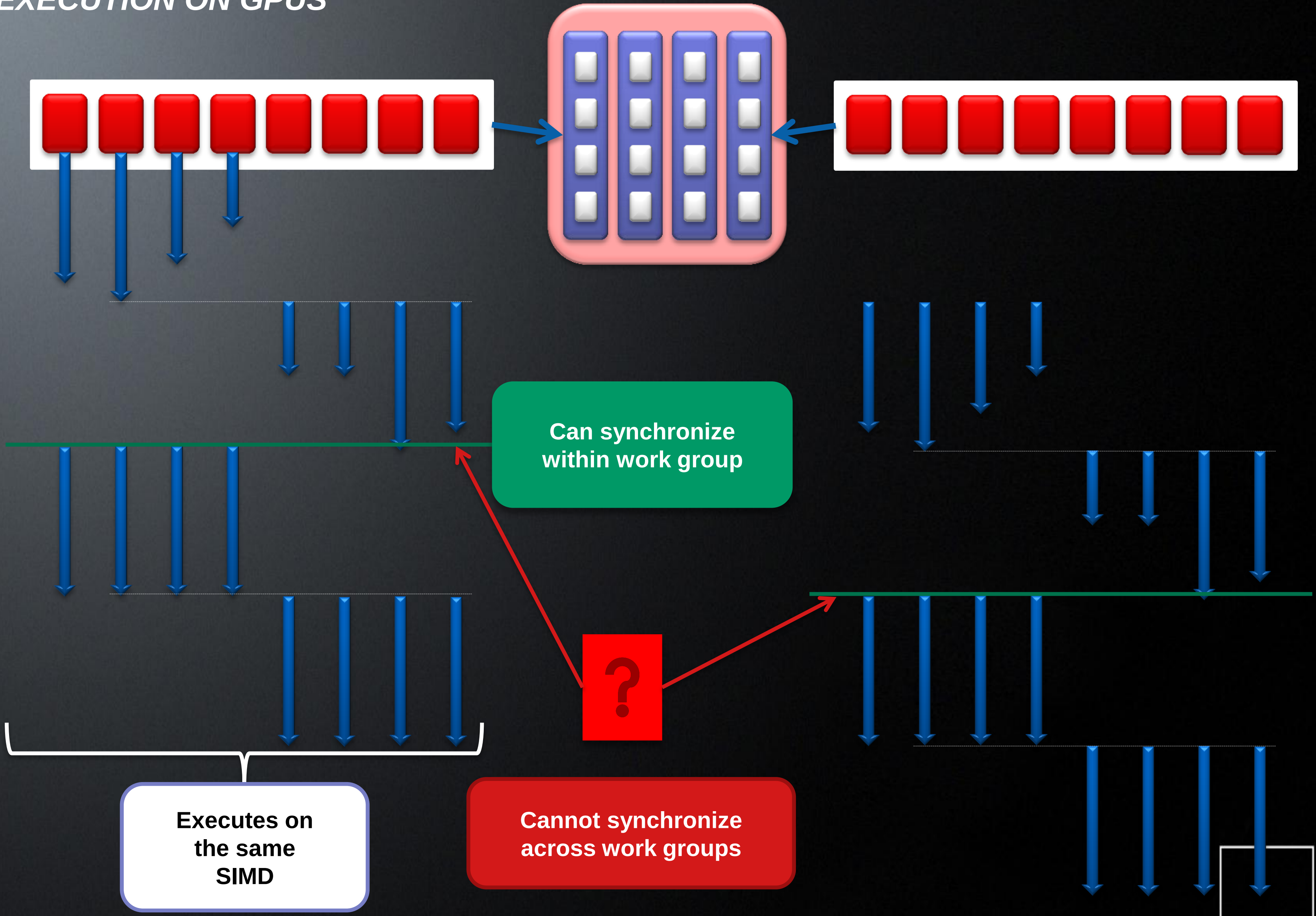


MAPPING WORK-GROUPS ON GPUS

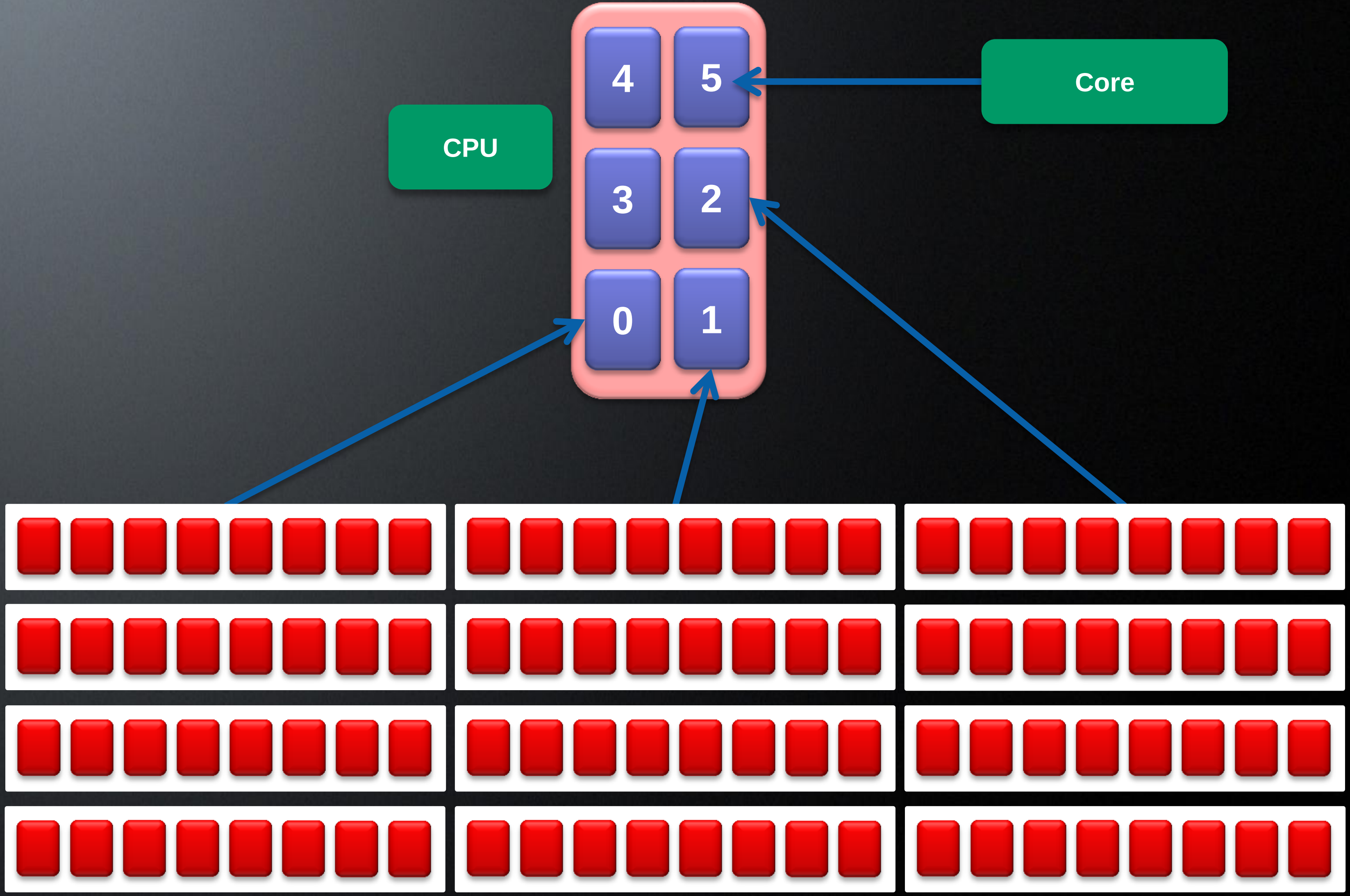
- Work-groups with multiple wavefronts → all wavefronts in same work-group scheduled on same SIMD
- Work-group size should be integral multiple of wavefront size
- (Advanced) Tip: When possible, pass work-group size at OpenCL kernel compile time



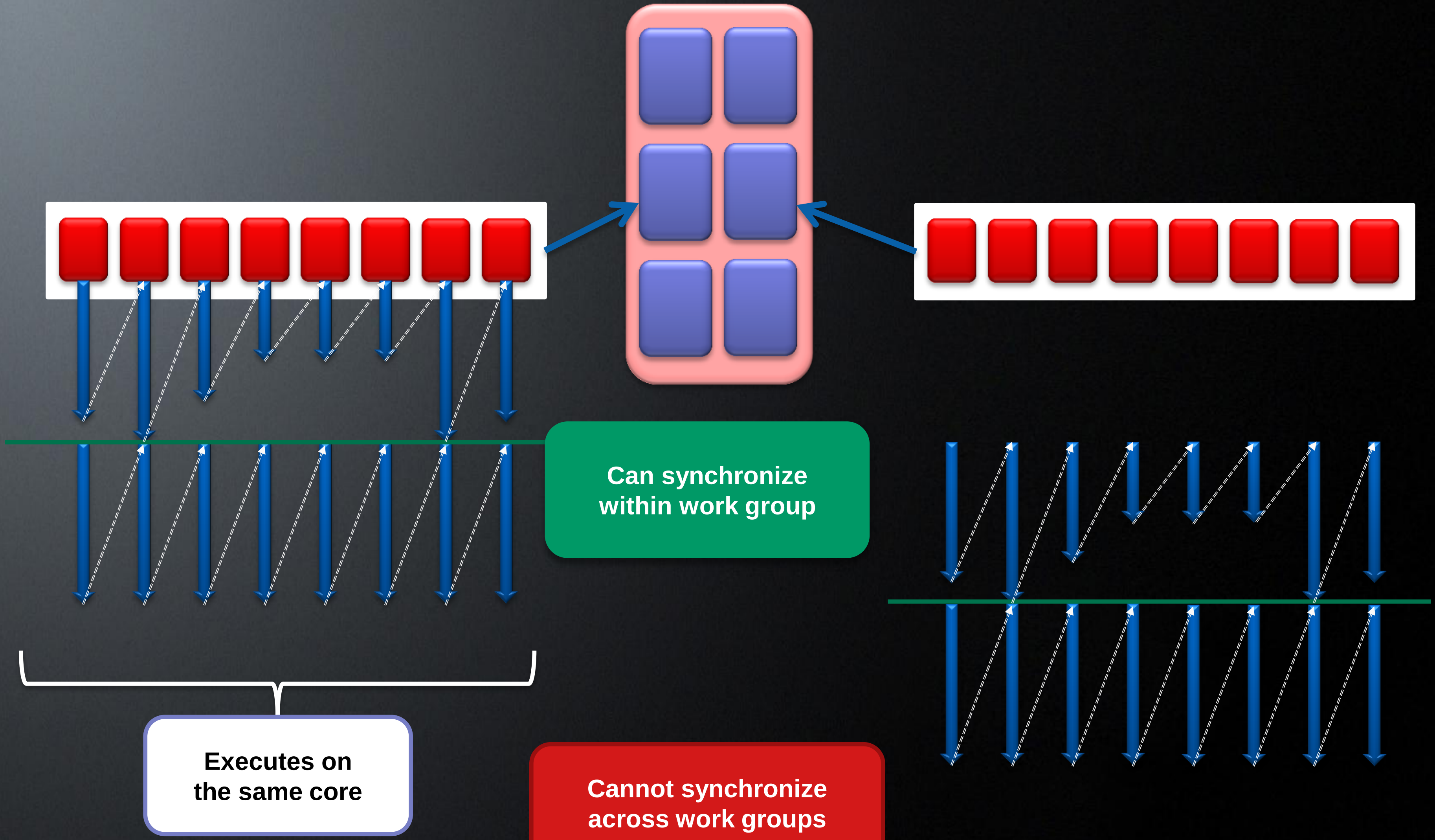
EXECUTION ON GPUS


Can synchronize within work group
?
Executes on the same SIMD
Cannot synchronize across work groups

MAPPING WORK-GROUPS ON CPUS

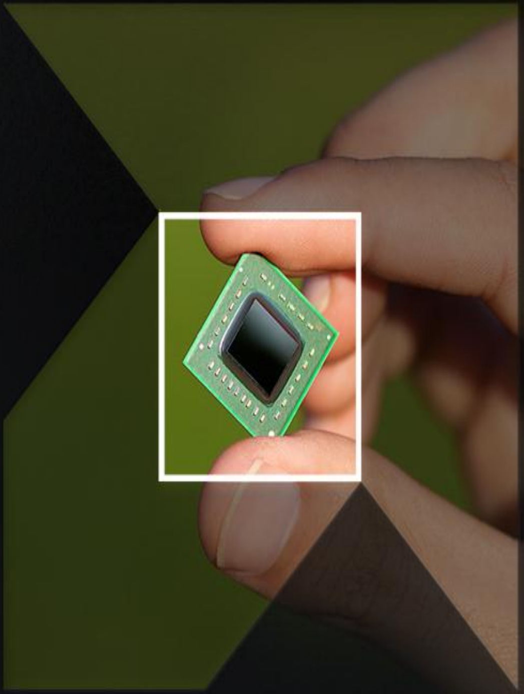


EXECUTION ON CPU


Can synchronize within work group
Executes on the same core
Cannot synchronize across work groups

PERFORMANCE DEVELOPER TOOLS

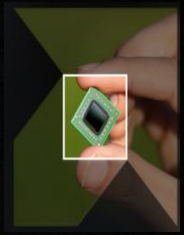
DEVELOPER TOOLS SUITE



AMD APP SDK - OVERVIEW

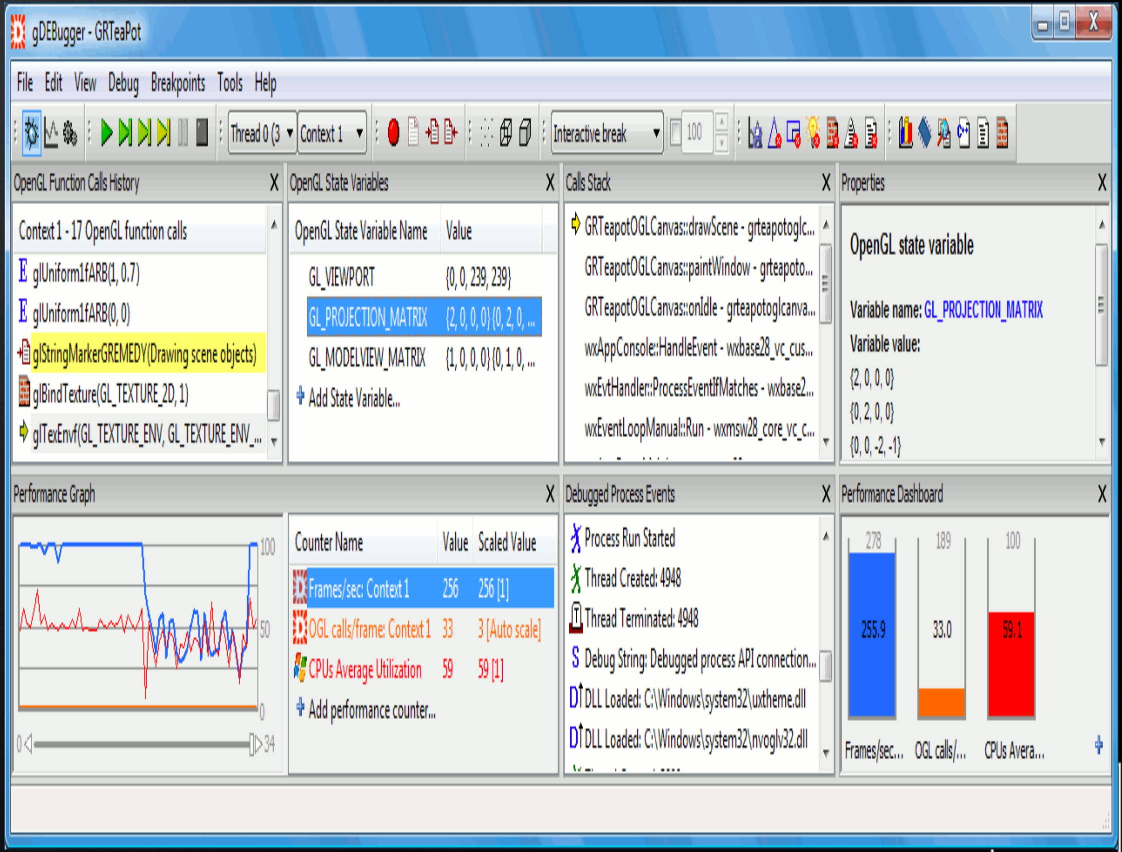
- OpenCL 1.1
- Windows 7, Linux
  - Apple OpenCL support thru MacOS
- Hardware Support
  - Fusion APUs
  - Discrete GPUs (Evergreen N Islands + …)
  - X86 (SSE3, SSE4. XOP, FMA4)
- Khronos (khr) OpenCL extensions
  - Atomics
  - Images
  - GL Sharing
  - D3D10 Sharing
  - Bytes / shorts
  - Device fission (CPU)
  - FP64 (double precision) (CPU/GPU - Cypress)
- AMD (amd) OpenCL extensions
  - FP64 (double precision)
  - Media Ops (SAD, Pack, Unpack,…)
  - Printf
  - Popcnt
- OpenVideo Decode UVD (Windows 7)
- Multi-GPU (Windows 7)
- FFT and BLAS-3 Libraries
- Binary Image Format / Offline Compile
- Developer Tool Suite (Later slides)
  - Debugger, Profiler, Optimizers
- Graphics Driver releases
  - OpenCL run-time available by default in Catalyst driver builds (Windows)
  - Silent install (Windows)

gDEBugger V5.8



- OpenCL and OpenGL API level Debugger, Profiler and Memory Analyzer
  - Access internal system information
  - Find bugs
  - Optimize compute performance
  - Minimize memory usage
- Windows, Linux and MacOS

- Proven mature product (6+ yrs)
- Widely deployed
  - 10k-s of users
  - Multiple industries



gDEBugger - GRTeaPot

OpenGL Function Calls History

Context 1 - 17 OpenGL function calls
glUniform1fARB(1, 0.7)
glUniform1fARB(0, 0)
glStringMarkerGREMEDY(Drawing scene objects)
glBindTexture(GL_TEXTURE_2D, 1)
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_...

OpenGL State Variables

OpenGL State Variable Name	Value
GL_VIEWPORT	{0, 0, 239, 239}
GL_PROJECTION_MATRIX	{2, 0, 0, 0} {0, 2, 0, ...
GL_MODELVIEW_MATRIX	{1, 0, 0, 0} {0, 1, 0, ...
Add State Variable...	

Calls Stack

GRTeapotOGLCanvas::drawScene - grteapotoglc...
GRTeapotOGLCanvas::paintWindow - grteapoto...
GRTeapotOGLCanvas::onIdle - grteapotoglcanva...
wxAppConsole::HandleEvent - wxbase28_vc_cus...
wxEvtHandler::ProcessEventIfMatches - wxbase2...
wxEventLoopManual::Run - wxmsw28_core_vc_c...

Properties

OpenGL state variable

Variable name: GL_PROJECTION_MATRIX
Variable value:
{2, 0, 0, 0}
{0, 2, 0, 0}
{0, 0, -2, -1}

Performance Graph

Counter Name	Value	Scaled Value
Frames/sec: Context 1	256	256 [1]
OGL calls/frame: Context 1	33	3 [Auto scale]
CPUs Average Utilization	59	59 [1]
Add performance counter...		

Debugged Process Events

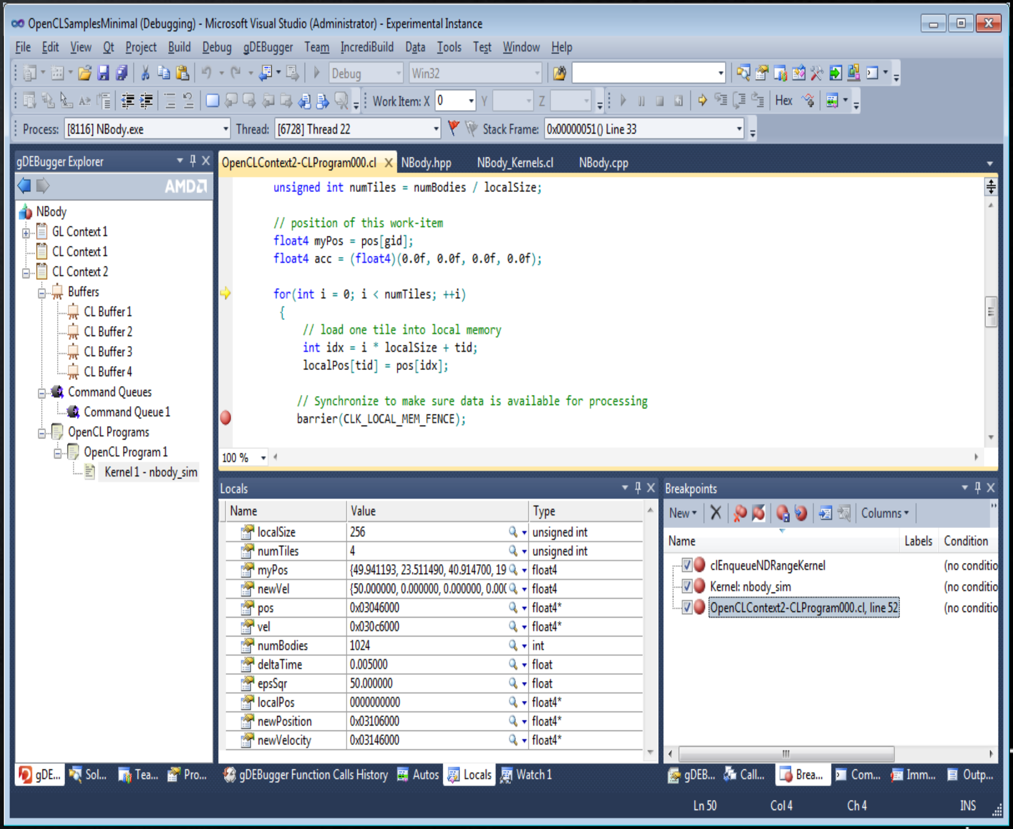
Process Run Started
Thread Created: 4948
Thread Terminated: 4948
Debug String: Debugged process API connection...
DLL Loaded: C:\Windows\system32\uxtheme.dll
DLL Loaded: C:\Windows\system32\nvoglv32.dll

Performance Dashboard

GDEBUGGER 6.0 (VISUAL STUDIO EXTENSION)

- Visual Studio 2010 extension
  - Native look and feel
  - Find bugs
  - Shortens development time
  - Minimize memory usage

- Includes all gDEBugger's capabilities
  - OpenCL and OpenGL
  - Trace OpenCL API calls
  - Single step through OpenCL Kernels
  - Insert breakpoints into OpenCL kernels
  - View Kernel local variables, buffers & images


OpenCLSamplesMinimal (Debugging) - Microsoft Visual Studio (Administrator) - Experimental Instance
File Edit View Qt Project Build Debug gDEBugger Team IncrediBuild Data Tools Test Window Help
Debug
Win32
Work Item: X 0
Hex
Process: [8116] NBody.exe
Thread: [6728] Thread 22
Stack Frame: 0x00000051() Line 33
gDEBugger Explorer
NBody
GL Context 1
CL Context 1
CL Context 2
Buffers
CL Buffer 1
CL Buffer 2
CL Buffer 3
CL Buffer 4
Command Queues
Command Queue 1
OpenCL Programs
OpenCL Program 1
Kernel 1 - nbody_sim
OpenCLContext2-CLProgram000.cl
NBody.hpp
NBody_Kernels.cl
NBody.cpp
100 %
gDE... Sol... Tea... Pro...
gDEBugger Function Calls History
Autos
Locals
Watch 1
gDEB... Call... Brea... Com... Imm... Outp...
Ln 50 Col 4 Ch 4 INS

```
    unsigned int numTiles = numBodies / localSize;

    // position of this work-item
    float4 myPos = pos[gid];
    float4 acc = (float4)(0.0f, 0.0f, 0.0f, 0.0f);

    for(int i = 0; i < numTiles; ++i)
     {
        // load one tile into local memory
        int idx = i * localSize + tid;
        localPos[tid] = pos[idx];

       // Synchronize to make sure data is available for processing
       barrier(CLK_LOCAL_MEM_FENCE);
```

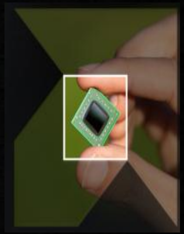
Locals

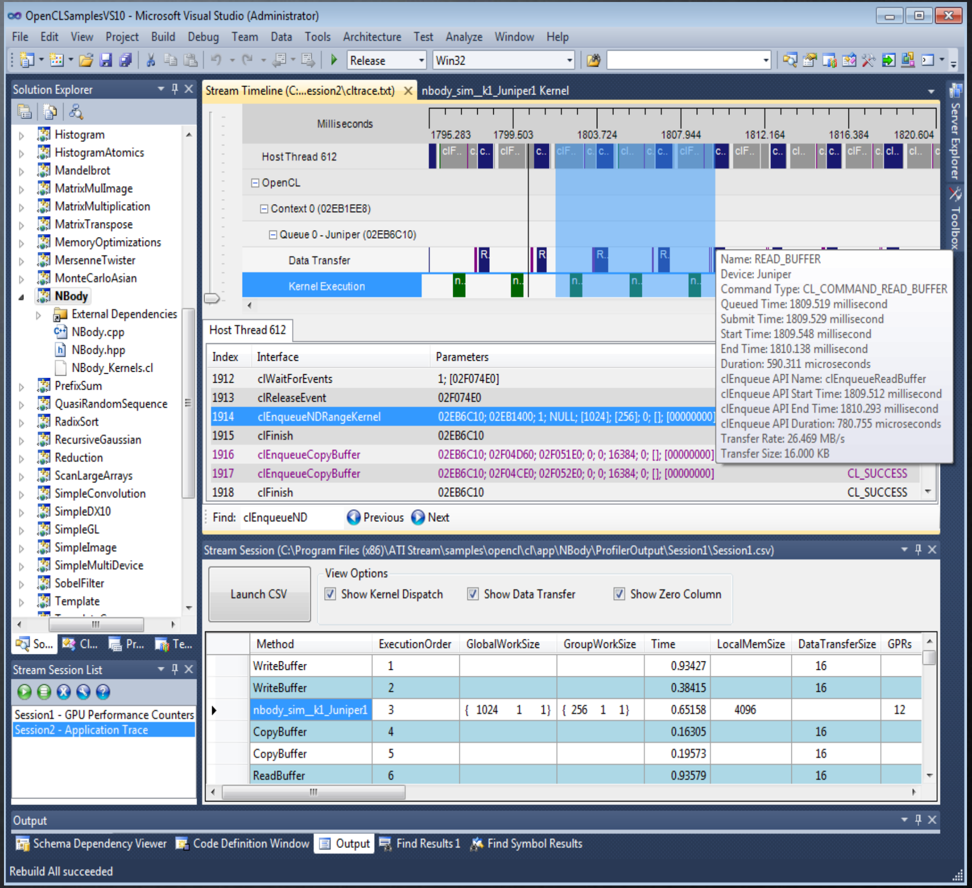
Name	Value	Type
localSize	256	unsigned int
numTiles	4	unsigned int
myPos	{49.941193, 23.511490, 40.914700, 19	float4
newVel	{50.000000, 0.000000, 0.000000, 0.00(	float4
pos	0x03046000	float4*
vel	0x030c6000	float4*
numBodies	1024	int
deltaTime	0.005000	float
epsSqr	50.000000	float
localPos	0000000000	float4*
newPosition	0x03106000	float4*
newVelocity	0x03146000	float4*

Breakpoints

Name	Labels	Condition
✓ clEnqueueNDRangeKernel		(no conditio
✓ Kernel: nbody_sim		(no conditio
✓ OpenCLContext2-CLProgram000.cl, line 52		(no conditio

APP PROFILER





OpenCLSamplesVS10 - Microsoft Visual Studio (Administrator)

Stream Timeline (C:...ession2\cltrace.txt) — nbody_sim__k1_Juniper1 Kernel

Name: READ_BUFFER
Device: Juniper
Command Type: CL_COMMAND_READ_BUFFER
Queued Time: 1809.519 millisecond
Submit Time: 1809.529 millisecond
Start Time: 1809.548 millisecond
End Time: 1810.138 millisecond
Duration: 590.311 microseconds
clEnqueue API Name: clEnqueueReadBuffer
clEnqueue API Start Time: 1809.512 millisecond
clEnqueue API End Time: 1810.293 millisecond
clEnqueue API Duration: 780.755 microseconds
Transfer Rate: 26.469 MB/s
Transfer Size: 16.000 KB

Host Thread 612

Index	Interface	Parameters	
1912	clWaitForEvents	1; [02F074E0]	
1913	clReleaseEvent	02F074E0	
1914	clEnqueueNDRangeKernel	02EB6C10; 02EB1400; 1; NULL; [1024]; [256]; 0; []; [00000000]	
1915	clFinish	02EB6C10	
1916	clEnqueueCopyBuffer	02EB6C10; 02F04D60; 02F051E0; 0; 0; 16384; 0; []; [00000000]	CL_SUCCESS
1917	clEnqueueCopyBuffer	02EB6C10; 02F04CE0; 02F052E0; 0; 0; 16384; 0; []; [00000000]	CL_SUCCESS
1918	clFinish	02EB6C10	CL_SUCCESS

Find: clEnqueueND

Stream Session (C:\Program Files (x86)\ATI Stream\samples\opencl\cl\app\NBody\ProfilerOutput\Session1\Session1.csv)

View Options: ✓ Show Kernel Dispatch ✓ Show Data Transfer ✓ Show Zero Column

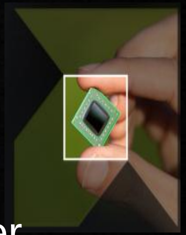
Method	ExecutionOrder	GlobalWorkSize	GroupWorkSize	Time	LocalMemSize	DataTransferSize	GPRs
WriteBuffer	1			0.93427		16	
WriteBuffer	2			0.38415		16	
nbody_sim__k1_Juniper1	3	{ 1024 1 1}	{ 256 1 1}	0.65158	4096		12
CopyBuffer	4			0.16305		16	
CopyBuffer	5			0.19573		16	
ReadBuffer	6			0.93579		16	

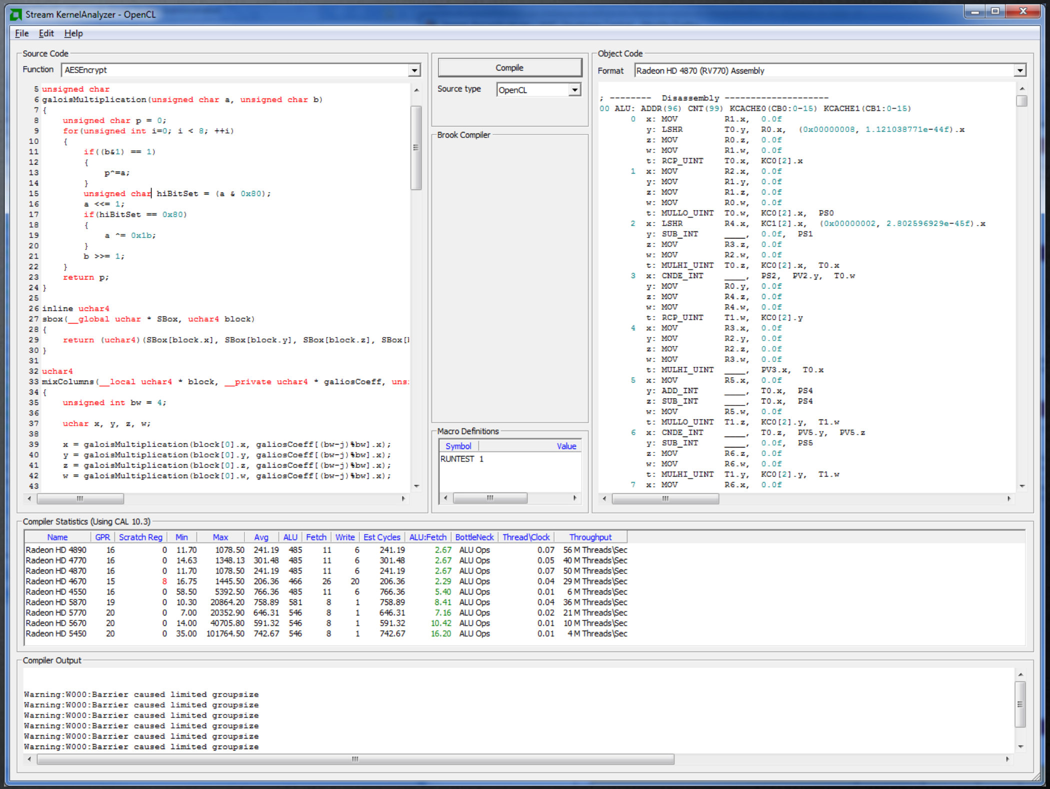
Stream Session List: Session1 - GPU Performance Counters; Session2 - Application Trace

Output: Rebuild All succeeded

- Run-time OpenCL and DirectCompute™ profiler
- Identify bottlenecks and optimization opportunities
- Access GPU hardware performance counters for AMD GPUs
- Visualize timeline and API trace for an OpenCL program
- Display IL and ISA (kernal disassembly)
- Visual Studio 2008/2010 Plugin
- Command line version for Linux

APP KERNELANALYZER





Stream KernelAnalyzer - OpenCL

File Edit Help

Source Code

Function: AESEncrypt

```c
unsigned char
galoisMultiplication(unsigned char a, unsigned char b)
{
    unsigned char p = 0;
    for(unsigned int i=0; i < 8; ++i)
    {
        if((b&1) == 1)
        {
            p^=a;
        }
        unsigned char hiBitSet = (a & 0x80);
        a <<= 1;
        if(hiBitSet == 0x80)
        {
            a ^= 0x1b;
        }
        b >>= 1;
    }
    return p;
}

inline uchar4
sbox(__global uchar * SBox, uchar4 block)
{
    return (uchar4)(SBox[block.x], SBox[block.y], SBox[block.z], SBox[
}

uchar4
mixColumns(__local uchar4 * block, __private uchar4 * galiosCoeff, uns
{
    unsigned int bw = 4;

    uchar x, y, z, w;

    x = galoisMultiplication(block[0].x, galiosCoeff[(bw-j)%bw].x);
    y = galoisMultiplication(block[0].y, galiosCoeff[(bw-j)%bw].x);
    z = galoisMultiplication(block[0].z, galiosCoeff[(bw-j)%bw].x);
    w = galoisMultiplication(block[0].w, galiosCoeff[(bw-j)%bw].x);

```

Compile

Source type: OpenCL

Brook Compiler

Macro Definitions

Symbol	Value
RUNTEST	1

Object Code

Format: Radeon HD 4870 (RV770) Assembly

```
; --------  Disassembly --------------------
00 ALU: ADDR(96) CNT(99) KCACHE0(CB0:0-15) KCACHE1(CB1:0-15)
      0  x: MOV        R1.x,  0.0f
         y: LSHR       T0.y,  R0.x,  (0x00000008, 1.121038771e-44f).x
         z: MOV        R0.z,  0.0f
         w: MOV        R1.w,  0.0f
         t: RCP_UINT   T0.x,  KC0[2].x
      1  x: MOV        R2.x,  0.0f
         y: MOV        R1.y,  0.0f
         z: MOV        R1.z,  0.0f
         w: MOV        R0.w,  0.0f
         t: MULLO_UINT T0.w,  KC0[2].x,  PS0
      2  x: LSHR       R4.x,  KC1[2].x,  (0x00000002, 2.802596929e-45f).x
         y: SUB_INT    ____,  0.0f,  PS1
         z: MOV        R3.z,  0.0f
         w: MOV        R2.w,  0.0f
         t: MULHI_UINT T0.z,  KC0[2].x,  T0.x
      3  x: CNDE_INT   ____,  PS2,  PV2.y,  T0.w
         y: MOV        R0.y,  0.0f
         z: MOV        R4.z,  0.0f
         w: MOV        R4.w,  0.0f
         t: RCP_UINT   T1.w,  KC0[2].y
      4  x: MOV        R3.x,  0.0f
         y: MOV        R2.y,  0.0f
         z: MOV        R2.z,  0.0f
         w: MOV        R3.w,  0.0f
         t: MULHI_UINT ____,  PV3.x,  T0.x
      5  x: MOV        R5.x,  0.0f
         y: ADD_INT    ____,  T0.x,  PS4
         z: SUB_INT    ____,  T0.x,  PS4
         w: MOV        R5.w,  0.0f
         t: MULLO_UINT T1.z,  KC0[2].y,  T1.w
      6  x: CNDE_INT   ____,  T0.z,  PV5.y,  PV5.z
         y: SUB_INT    ____,  0.0f,  PS5
         z: MOV        R6.z,  0.0f
         w: MOV        R6.w,  0.0f
         t: MULHI_UINT T1.y,  KC0[2].y,  T1.w
      7  x: MOV        R6.x,  0.0f
```

Compiler Statistics (Using CAL 10.3)

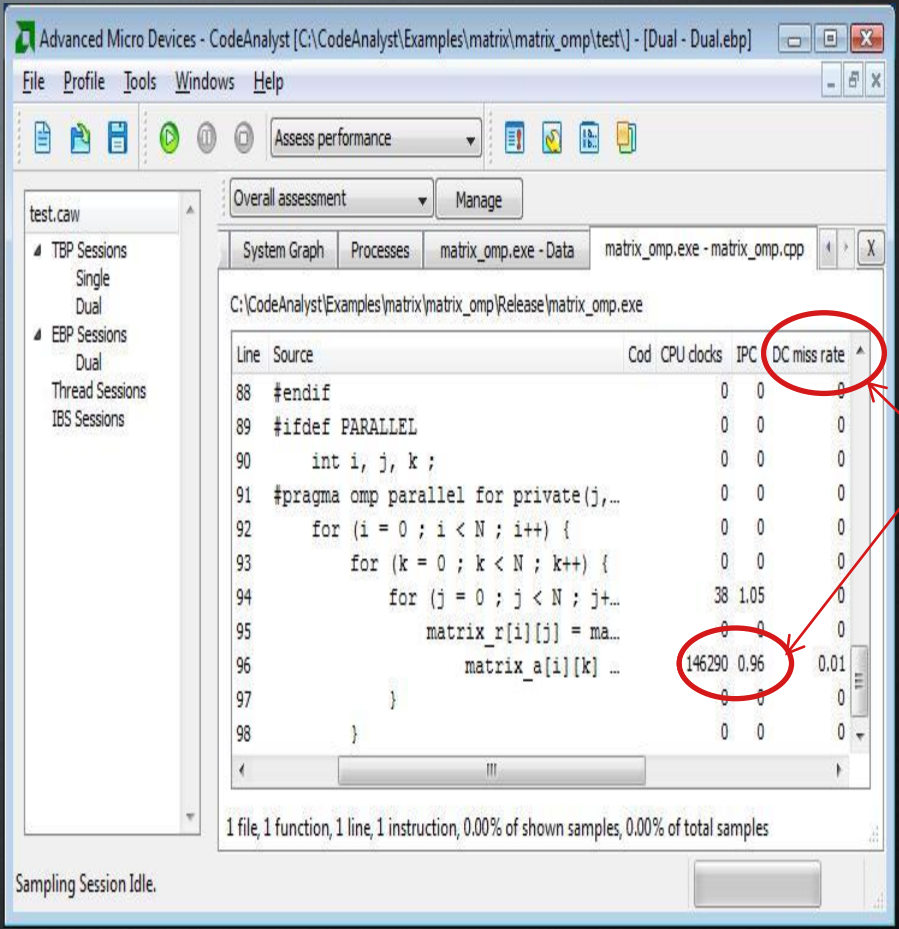
Name	GPR	Scratch Reg	Min	Max	Avg	ALU	Fetch	Write	Est Cycles	ALU:Fetch	BottleNeck	Thread\Clock	Throughput
Radeon HD 4890	16	0	11.70	1078.50	241.19	485	11	6	241.19	2.67	ALU Ops	0.07	56 M Threads\Sec
Radeon HD 4770	16	0	14.63	1348.13	301.48	485	11	6	301.48	2.67	ALU Ops	0.05	40 M Threads\Sec
Radeon HD 4870	16	0	11.70	1078.50	241.19	485	11	6	241.19	2.67	ALU Ops	0.07	50 M Threads\Sec
Radeon HD 4670	15	8	16.75	1445.50	206.36	466	26	20	206.36	2.29	ALU Ops	0.04	29 M Threads\Sec
Radeon HD 4550	16	0	58.50	5392.50	766.36	485	11	6	766.36	5.40	ALU Ops	0.01	6 M Threads\Sec
Radeon HD 5870	19	0	10.30	20864.20	758.89	581	8	1	758.89	8.41	ALU Ops	0.04	36 M Threads\Sec
Radeon HD 5770	20	0	7.00	20352.90	646.31	546	8	1	646.31	7.16	ALU Ops	0.02	21 M Threads\Sec
Radeon HD 5670	20	0	14.00	40705.80	591.32	546	8	1	591.32	10.42	ALU Ops	0.01	10 M Threads\Sec
Radeon HD 5450	20	0	35.00	101764.50	742.67	546	8	1	742.67	16.20	ALU Ops	0.01	4 M Threads\Sec

Compiler Output

```
Warning:W000:Barrier caused limited groupsize
Warning:W000:Barrier caused limited groupsize
Warning:W000:Barrier caused limited groupsize
Warning:W000:Barrier caused limited groupsize
Warning:W000:Barrier caused limited groupsize
Warning:W000:Barrier caused limited groupsize
```

- Version of GPU ShaderAnalyzer tool targeting Accelerated Parallel Processing
- Statically analyze of OpenCL Kernels for AMD Radeon GPUs.
- Display compiled Kernels as IL or GPU ISA
- Display statistics from the AMD Kernel Compiler
- Estimate Kernel performance
- Includes support for previous 12 versions of Catalyst Driver

CODEANALYST PERFORMANCE ANALYZER



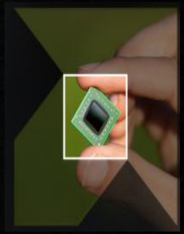
Advanced Micro Devices - CodeAnalyst [C:\CodeAnalyst\Examples\matrix\matrix_omp\test\] - [Dual - Dual.ebp]

C:\CodeAnalyst\Examples\matrix\matrix_omp\Release\matrix_omp.exe

Line	Source	Cod	CPU clocks	IPC	DC miss rate
88	#endif		0	0	0
89	#ifdef PARALLEL		0	0	0
90	int i, j, k ;		0	0	0
91	#pragma omp parallel for private(j,...		0	0	0
92	for (i = 0 ; i < N ; i++) {		0	0	0
93	for (k = 0 ; k < N ; k++) {		0	0	0
94	for (j = 0 ; j < N ; j+...		38	1.05	0
95	matrix_r[i][j] = ma...		0	0	0
96	matrix_a[i][k] ...		146290	0.96	0.01
97	}		0	0	0
98	}		0	0	0

1 file, 1 function, 1 line, 1 instruction, 0.00% of shown samples, 0.00% of total samples

Sampling Session Idle.



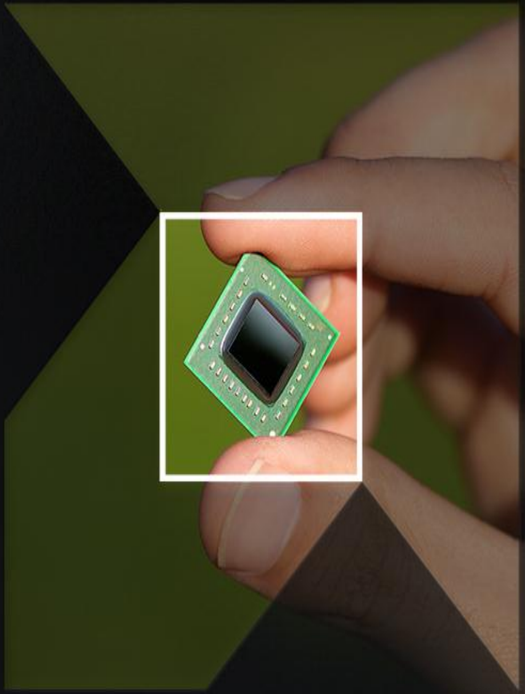
- Profiling suite to identify, investigate and tune code performance on AMD platforms

- Find time critical hot-spots
  - C, C++, Fortran
  - Java, .Net managed cod
-
- Diagnose performance issues
  - Time based profiling
  - Event based sampling
  - Instruction based sampling

- Identify thread-affinity and core utilization problems

- Windows and Linux platforms

LINUX

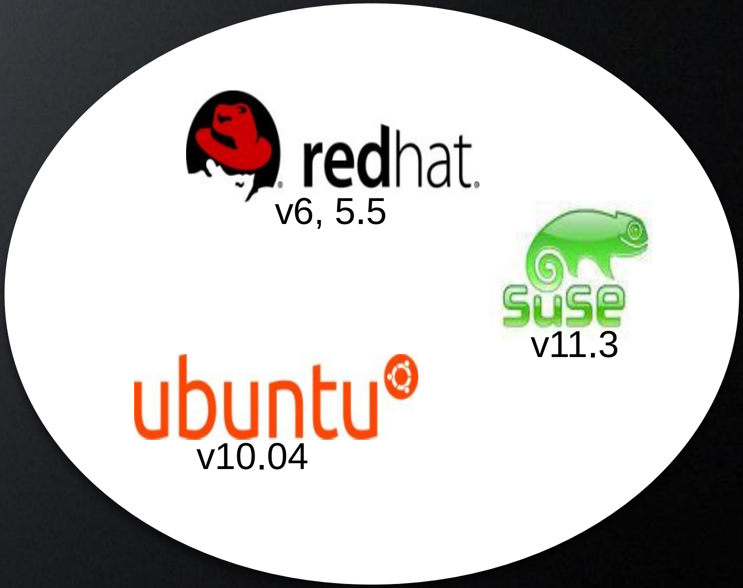


AMD APP OPENCL FOR LINUX

- SDK 2.5
  - Linux OpenCL SDK available
  - gDEBugger
  - LLVM optimization passes
  - APP Profiler
  - Code Analyst

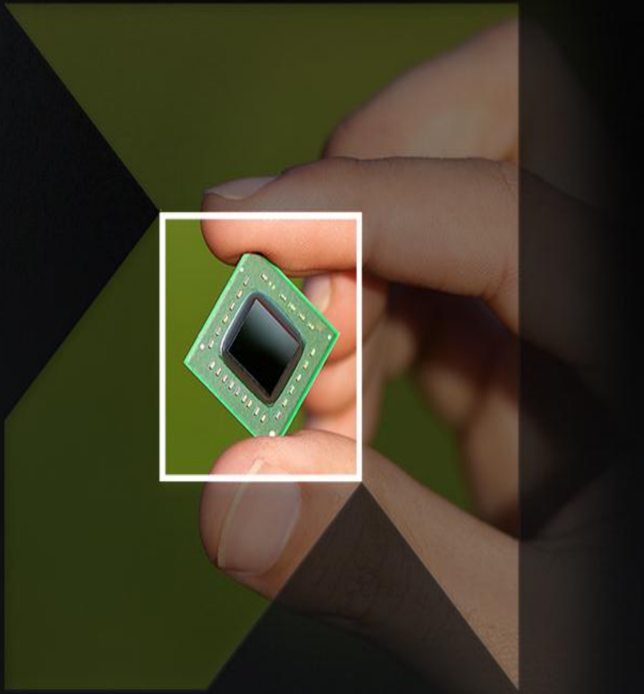
- Multicoreware
  - Parallel Path Analyzer
  - Task Manager
  - GMAC

- OpenCL
  - Open standard
  - Closed implementation



FUSION

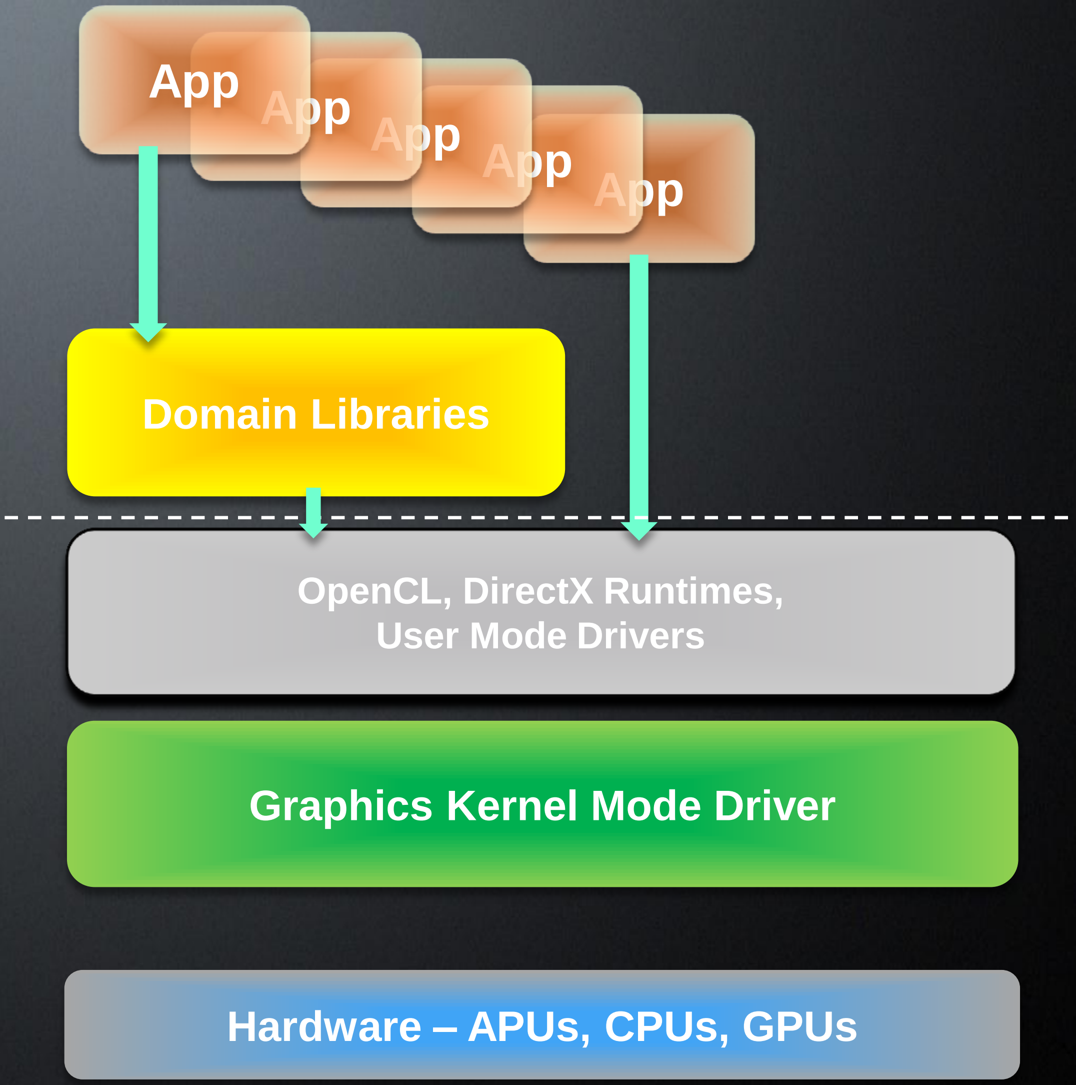
A PATH TO THE FUTURE



FSA ROADMAP

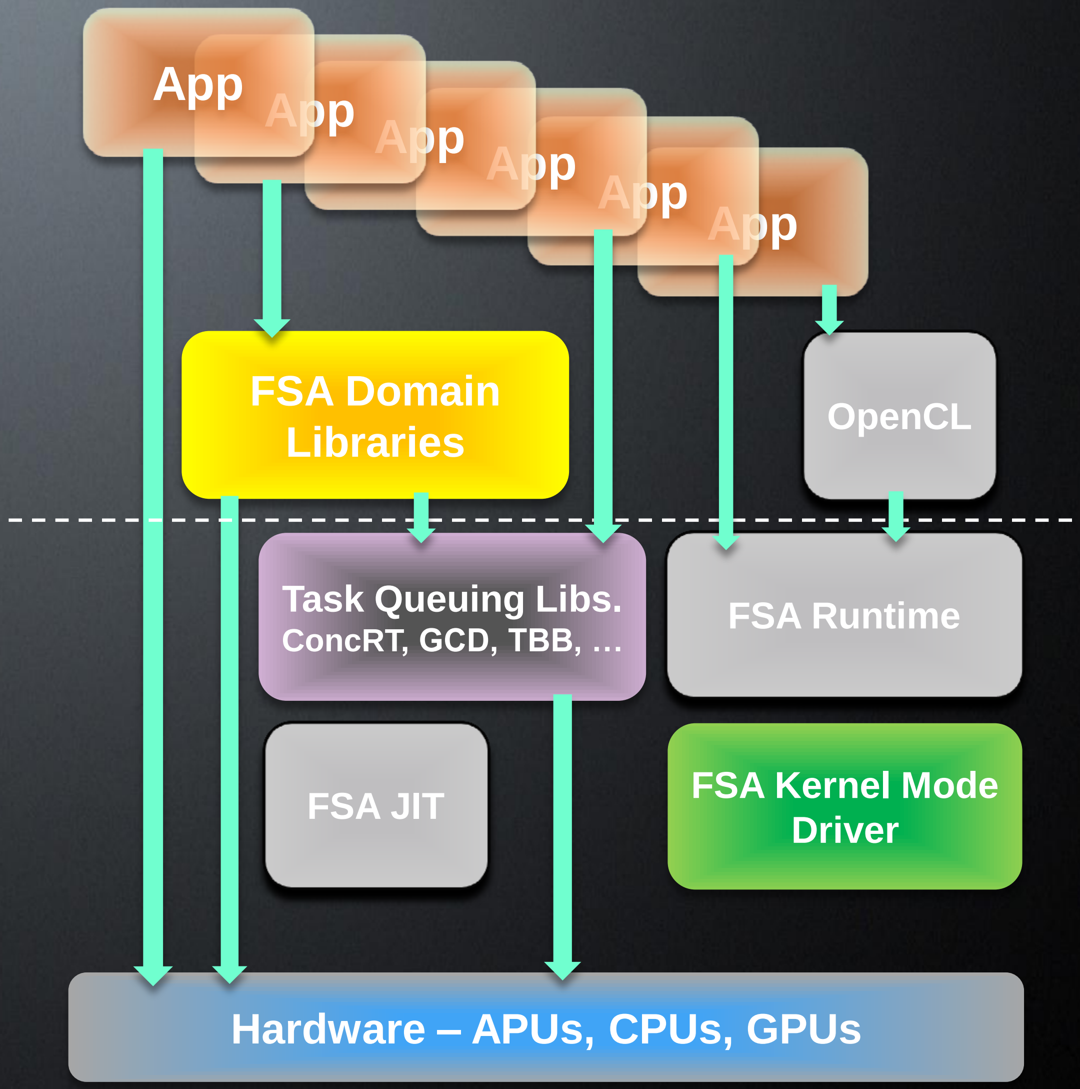
Physical Integration	Optimized Platforms	Architectural Integration	System Integration
Integrate CPU & GPU in silicon	GPU Compute C++ support	Unified Address Space for CPU and GPU	GPU compute context switch
Unified Memory Controller	User mode schedulng	GPU uses pageable system memory via CPU pointers	GPU graphics pre-emption
Common Manufacturing Technology	Bi-Directional Power Mgmt between CPU and GPU	Fully coherent memory between CPU & GPU	Quality of Service
			Extend to Discrete GPU

OPENCL DRIVER STACK (NOW)


App
App
App
App
App
Domain Libraries
OpenCL, DirectX Runtimes,
User Mode Drivers
Graphics Kernel Mode Driver
Hardware – APUs, CPUs, GPUs

⬜	AMD user mode component
🟩	AMD kernel mode component
	All other SW contributed by AMD or 3rd parties

OPENCL DRIVER STACK (FSA)

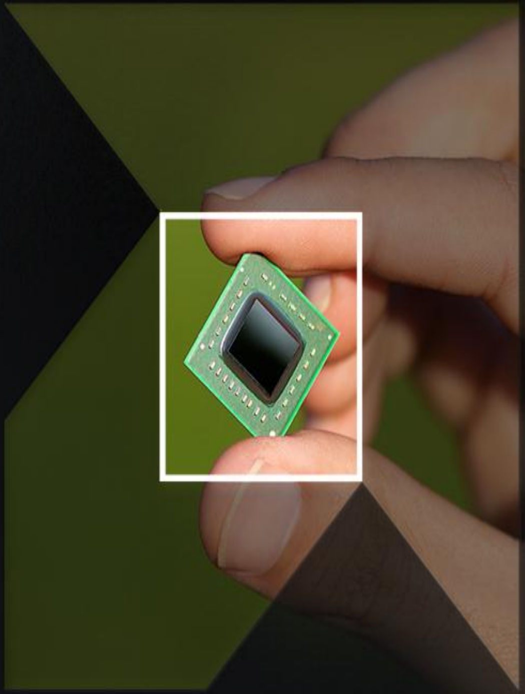


⬜ AMD user mode component

🟩 AMD kernel mode component

All other SW contributed by AMD or 3rd parties

QUESTIONS?



Disclaimer & Attribution