At the heart of every AMD APU/GPU is a power aware high performance set of compute units that have been advancing to bring users new levels of programmability, precision and performance.
AGENDA ➔ AMD Graphic Core Next Architecture

- Unified Scalable Graphic Processing Unit (GPU) optimized for Graphics and Compute
  - Multiple Engine Architecture with Multi-Task Capabilities
  - Compute Unit Architecture
  - Multi-Level R/W Cache Architecture

- What will not be discussed
  - Roadmaps/Schedules
  - New Product Configurations
  - Feature Rollout

- Visit AMD Fusion Developers Summit online for Fusion System Architecture details
SCALABLE MULTI-TASK GRAPHICS ENGINE

GFX Command Processor

Work Distributor

Primitive Pipe 0
- HOS
- Tessellate
- Geometry

Primitive Pipe n
- HOS
- Tessellate
- Geometry

Pixel Pipe 0
- Scan Conversion
- RB

Pixel Pipe n
- Scan Conversion
- RB

Unified Shader Core

MC HUB & MEM

R/W L2

HOS - High Order Surface
RB - Render Backend
CS - Compute Shader
GFX - Graphics
SCALABLE MULTI-TASK GRAPHICS ENGINE

Primitive Scaling ➔ Multiple Primitive Pipelines
Pixel Scaling ➔ Multiple Screen Partitions
Multi-task graphics engine use of unified shader
MULTI-ENGINE UNIFIED COMPUTING GPU

Asynchronous Compute Engine (ACE)

- Command Processor
  - Hardware Command Queue Fetcher
  - Device Coherent R/W Cache Access
    - Load Acquire/Store Release Semantics
  - Global Data Share Access
  - Hardware synchronization
- Independent & Concurrent Grid/Group Dispatchers
- Real time task scheduling
- Background task scheduling
- Compute Task Graph Processing
  - Hardware Scheduling
  - Task Queue Context Switching
- Error Detection & Correction (EDCC)
  - For GDDR and internal SRAM Pools
AMD GRAPHIC CORE NEXT
COMPUTE UNIT ARCHITECTURE
NON-VLIW ISA WITH SCALAR + VECTOR UNITS

- Simpler ISA compared to previous generation
  - No clauses
  - No VLIW packing
  - Control flow more directly programmed

- Advanced language feature support
  - Exception support
  - Function calls
  - Recursion

- Enhanced extended ALU operations
  - Media ops
  - Integer ops
  - Floating point atomics (min, max, cmpxchg)

- Improved debug support
  - HW functionality to improve debug support
PROGRAMMERS VIEW OF COMPUTE UNIT

Input Data: PC/State/Vector Register/Scalar Register

- SIMD PC & IB
- Instruction Fetch Arbitration
- Branch & MSG Unit
- Export/GDS Decode
- Vector Memory Decode
- Scalar Decode
- Vector Decode
- LDS Decode
- Scalar Unit
- Integer ALU
- 8 KB Registers
- SIMD
- 64 KB Registers
- MP Vector ALU
- 64 KB LDS Memory
- 4 CU Shared 16KB Scalar Read Only L1
- 4 CU Shared 32KB Instruction L1
- Rqst Arb
- R/W L2
- R/W L2
- Export Bus
- PC - Program Counter
- MSG - Message
- SIMD - Single Instruction multiple data

Input Data: PC/State/Vector Register/Scalar Register
float fn0(float a, float b) {
    if (a > b)
        return((a - b) * a);
    else
        return((b - a) * b);
}

// Registers r0 contains “a”, r1 contains “b”
// Value is returned in r2

v_cmp_gt_f32 r0,r1       // a > b, establish VCC
s_mov_b64 s0,exec       // Save current exec mask
s_and_b64 exec,vcc,exec  // Do “if”

s_cbranch_vccz label0   // Branch if all lanes fail
v_sub_f32 r2,r0,r1      // result = a - b
v_mul_f32 r2,r2,r0      // result = result * a

label0:

s_andn2_b64 exec,s0,exec   // Do “else”(s0 & !exec)
s_cbranch_execz label1   // Branch if all lanes fail
v_sub_f32 r2,r1,r0      // result = b - a
v_mul_f32 r2,r2,r1      // result = result * b

label1:

s_mov_b64 exec,s0       // Restore exec mask
**COMPUTE UNIT ARCHITECTURE**

Input Data: PC/State/Vector Register/Scalar Register

- **Instruction Fetch Arbitration**
  - SIMD PC & IB
  - SIMD PC & IB
  - SIMD PC & IB
  - SIMD PC & IB

- **Instruction Arbitration**
  - Branch & MSG Unit
  - SIMD PC & IB
  - SIMD PC & IB
  - SIMD PC & IB

- **Instruction Decode**
  - SIMD PC & IB
  - SIMD PC & IB
  - SIMD PC & IB

- **Scalar Decode**
  - 8 KB Registers
  - Integer ALU

- **Vector Decode**

- **LDS Decode**

- **Scalar Unit**
  - 64 KB LDS Memory
  - 64 KB LDS Memory
  - 64 KB LDS Memory
  - 64 KB LDS Memory

- **Vector Memory Decode**

- **Export/GDS Decode**

- **MP**
  - Vector ALU
  - 64 KB Registers
  - R/W L2

- **MP**
  - Vector ALU
  - 64 KB Registers
  - R/W L2

- **MP**
  - Vector ALU
  - 64 KB Registers
  - R/W L2

- **MP**
  - Vector ALU
  - 64 KB Registers
  - R/W L2

- **Branch & MSG Unit**
  - R/W L1

- **Export/MSG Unit**
  - R/W L1

- **Msg Bus**

- **4 CU Shared 16KB Scalar Read Only L1**

- **4 CU Shared 32KB Instruction L1**

- **R/W L2**

- **R/W Bus**

- **16KB**

- **R/W L2**

- **Msg Arb**
<table>
<thead>
<tr>
<th>4 Way VLIW SIMD</th>
<th>4 SIMD non-VLIW</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 Single Precision MAC</td>
<td>64 Single Precision MAC</td>
</tr>
<tr>
<td>VGPR ➔ 64 * 4 * 256-32bit ➔ 256KB</td>
<td>VGPR ➔ 4 * 64 * 256-32bit ➔ 256KB</td>
</tr>
<tr>
<td>1 VLIW Instruction * 4 Ops ➔ Dependencies limitations</td>
<td>4SIMD * 1 ALU Operation ➔ Occupancy limitations</td>
</tr>
<tr>
<td>3 SRC GPRs, 1 Vector Destination</td>
<td>3 SRC GPRs, 1 Vector\1Scalar Register Destination</td>
</tr>
<tr>
<td>Compiler manage VGPR port conflicts</td>
<td>No VGPR port conflicts</td>
</tr>
<tr>
<td>VALU Instruction Bandwidth ➔ 1-7 dwords(~2 dwords/clk)</td>
<td>VALU Instruction Bandwidth ➔ 1-2 dwords/cycle</td>
</tr>
<tr>
<td>Interleaved wavefront instruction required</td>
<td>Vector back-to-back wavefront instruction issue</td>
</tr>
<tr>
<td>Specialized complicated compiler scheduling</td>
<td>Standard compiler scheduling &amp; optimizations</td>
</tr>
<tr>
<td>Difficult assembly creation, analysis, &amp; debug</td>
<td>Simplified assembly creation, analysis, &amp; debug</td>
</tr>
<tr>
<td>Complicated tool chain support</td>
<td>Simplified tool chain development and support</td>
</tr>
<tr>
<td>Less predictive results and performance</td>
<td>Stable and predictive results and performance</td>
</tr>
</tbody>
</table>
**R/W CACHE**

- Read / Write Data cached
  - Bandwidth amplification
  - Improved behavior on more memory access patterns
  - Improved write to read reuse performance
  - L1 Write-through / L2 write-back caches

- Relaxed memory model
  - Consistency controls available for locality of load/store/atomic

- GPU Coherent
  - Acquire / Release semantics control data visibility across the machine
  - L2 coherent = all CUs can have the same view of data

- Remote Global atomics
  - Performed in L2 cache
AMD Graphic Core Next Compute Unit Architecture Summary

- A heavily multi-threaded Compute Unit (CU) architected for throughput
  - Efficiently balanced for graphics and general compute
  - Simplified coding for performance, debug and analysis
  - Simplified machine view for tool chain development
  - Low latency flexible control flow operations
  - Read/Write Cache Hierarchy improves I/O characteristics
  - Flexible vector load, store, and remote atomic operations
  - Load acquire / Store release consistency controls
QUESTIONS ?
Disclaimer & Attribution

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. There is no obligation to update or otherwise correct or revise this information. However, we reserve the right to revise this information and to make changes from time to time to the content hereof without obligation to notify any person of such revisions or changes.

NO REPRESENTATIONS OR WARRANTIES ARE MADE WITH RESPECT TO THE CONTENTS HEREOF AND NO RESPONSIBILITY IS ASSUMED FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

ALL IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE ARE EXPRESSLY DISCLAIMED. IN NO EVENT WILL ANY LIABILITY TO ANY PERSON BE INCURRED FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREOF, EVEN IF EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

AMD, the AMD arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. All other names used in this presentation are for informational purposes only and may be trademarks of their respective owners.

OpenCL is a trademark of Apple Inc. used with permission by Khronos.

DirectX is a registered trademark of Microsoft Corporation.

© 2011 Advanced Micro Devices, Inc. All rights reserved.