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

- Introduction to AMD & Fusion
- Embracing Heterogeneous Computing
- Emerging Consumer Workloads
- Future Challenges for Heterogeneous Systems
- AMD Graphic Core Next Architecture
INTRODUCTION TO AMD & FUSION
In the year of 2006

- **AMD**: Leading-Edge x86s CPUs
  - Consumer, Workstation, Server, HPC
- **ATI**: Leading Edge GPUs
  - Handheld, Consumer, Console, Workstation
- **AMD & ATI**: Combine with vision of merging technologies to drive a world of fusion to enable new experiences for consumers, businesses, developers, artist, educators, scientist, etc.
The Plan ➔ The Future Is Fusion

CPU is ideal for scalar processing
- Out of order x86 cores with low latency memory access
- Optimized for sequential and branching algorithms
- Runs existing applications very well

GPU is ideal for parallel processing
- GPU shaders optimized for throughput computing
- Ready for emerging workloads
- Media processing, simulation, natural UI, etc

Provides optimal performance combinations for a wide range of platform configurations
x86 CPU owns the Software World
- Windows®, MacOS and Linux® franchises
- Thousands of apps
- Established programming and memory model
- Mature tool chain
- Extensive backward compatibility for applications and OSs
- High barrier to entry

GPU Optimized for Modern Workloads
- Enormous parallel computing capacity
- Outstanding performance-per-watt-per-dollar
- Very efficient hardware threading
- SIMD architecture well matched to modern workloads: video, audio, graphics
FUSION APUS: PUTTING IT ALL TOGETHER

- **Microprocessor Advancement**
  - Single-Thread Era
  - Multi-Core Era
  - Heterogeneous Systems Era

- **Heterogeneous Computing**
  - High Performance
  - Task Parallel Execution
  - Power-efficient Data Parallel Execution

- **Fusion APU**
  - System-level Programmable
  - OCL/DC Driver-based programs
  - Graphics Driver-based programs

- **Throughput Performance**

- **Programmer Accessibility**
  - Unacceptable
  - Experts Only
  - Mainstream
In the Year of 2011
AMD’s First Fusion Family of APUs Addressing a Wide-Range of Products and Markets

One Design, Fewer Watts, Massive Capability

C-Series and E-Series APUs have an area of 75 sq mm - smaller than a typical thumbnail or alpha key on a PC keyboard.

9W C-Series APU (formerly codenamed “Ontario”)
- HD Netbooks
- Ultra-small form factors
- Delivers powerful, mainstream-like HD entertainment experiences

18W E-Series APU (formerly codenamed “Zacate”)
- Mainstream notebooks
- All-in-one desktops
- Delivers amazing full HD entertainment experience

Up to 10-plus hours of battery life!*

New low-power “Bobcat” x86 cores and a DirectX®11 capable GPU

*Resting battery life as measured with industry standard tests.

AMD A-SERIES APU (CODENAME “LLANO”): 2011 MAINSTREAM AND PERFORMANCE PLATFORMS

- A4/A6/A8 series with multiple skews shipping
- Manufactured by Global Foundries - 32nm process
- Targeting mainstream and performance notebooks and desktops

- Combo of mainstream x86 quad-core CPUs and discrete DirectX® 11 capable graphics
  - >500 GFLOPs of compute power¹
  - Enables software providers to deliver higher level experiences at mainstream price points

- Enjoy AMD AllDay™ battery life²

---

¹ Theoretical peak performance
² AMD defines “all day” battery life as 8+ hours of idle time. Active battery life data pending. BR-C1
Embracing Heterogeneous Computing
THREE ERAS OF PROCESSOR PERFORMANCE

**Single-Core Era**

Enabled by:
- ✓ Moore’s Law
- ✓ Voltage & Process Scaling
- ✓ Micro Architecture

Constrained by:
- ✗ Power
- ✗ Complexity

**Multi-Core Era**

Enabled by:
- ✓ Moore’s Law
- ✓ Desire for Throughput
- ✓ 20 years of SMP arch

Constrained by:
- ✗ Power
- ✗ Parallel SW availability
- ✗ Scalability

**Heterogeneous Systems Era**

Enabled by:
- ✓ Moore’s Law
- ✓ Abundant data parallelism
- ✓ Power efficient GPUs

Temporarily constrained by:
- ✗ Programming models
- ✗ Communication overheads
- ✗ Workloads

---

Single-thread Performance vs. Time

Throughput Performance vs. Time

Targeted Application Performance vs. Time

we are here
WHAT IS HETEROGENEOUS COMPUTING SYSTEM

– A system comprised of two or more compute engines with significant structural differences
– Example: low latency x86 CPU Cores and high throughput GPU Compute Units

High Performance x86 CPU Cores
• Low thread count (1-16, 32?)
• Large Caches provide Low Latency
• Out of Order, renaming, speculative execution
• Super Scalar with speculative execution
• Multi-port Registers for instance access
• Great extremes to find instruction level parallelism, optimize dependency checking and branch processing

High Performance GPU Compute Cores
– Large thread Counts (30k-128k, 256k?)
– Shared Hierarchical Caches provide temporal and locality based reuse
– Shared Instruction delivery to minimize cost & power
– Bank registers and interleave execution to minimize register cost
– Interleave execution of parallel work to hide pipeline delays, branch delays.
WHY HETEROGENEOUS SYSTEMS: EXTRACTING MORE PARALLELISM

**Fine-grain data parallel Code**

- Loop 16 times for 16 pieces of data
- Maps very well to integrated SIMD dataflow (ie: SSE/AVX)

**Nested data parallel Code**

- Lots of conditional data parallelism. Benefits from closer coupling between CPU & GPU

**Coarse-grain data parallel Code**

- Loop 1M times for 1M pieces of data
- 2D array representing very large dataset
- Maps very well to Throughput-oriented data parallel engines
Emerging Workloads
### THE BIG EXPERIENCE/SMALL FORM FACTOR PARADOX

<table>
<thead>
<tr>
<th>Technology</th>
<th>Mid 1990s</th>
<th>Mid 2000s</th>
<th>Now: Parallel/Data-Dense</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display</td>
<td>4:3 @ 0.5 megapixel</td>
<td>4:3 @ 1.2 megapixels</td>
<td>16:9 @ 7 megapixels</td>
</tr>
<tr>
<td>Content</td>
<td>Email, film &amp; scanners</td>
<td>Digital cameras, SD webcams (1-5 MB files)</td>
<td>HD video flipcams, phones, webcams (1GB)</td>
</tr>
<tr>
<td>Online</td>
<td>Text and low res photos</td>
<td>WWW and streaming SD video</td>
<td>3D Internet apps and HD video online, social networking w/HD files</td>
</tr>
<tr>
<td>Multimedia</td>
<td>CD-ROM</td>
<td>DVDs</td>
<td>3D Blu-ray HD</td>
</tr>
<tr>
<td>Interface</td>
<td>Mouse &amp; keyboard</td>
<td>Mouse &amp; keyboard</td>
<td>Multi-touch, facial/gesture/voice recognition + mouse &amp; keyboard</td>
</tr>
<tr>
<td>Battery Life*</td>
<td>1-2 Hours</td>
<td>3-4 Hours</td>
<td>All day computing (8+ Hours)</td>
</tr>
</tbody>
</table>

*Resting battery life as measured with industry standard tests.

---

**Workloads**

**Performance that is seen and felt**

---

*Resting battery life as measured with industry standard tests.*
FOCUSING ON THE EXPERIENCES THAT MATTER

Consumer PC Usage

- Email
- Web browsing
- Office productivity
- Listen to music
- Online chat
- Watching online video
- Photo editing
- Personal finances
- Taking notes
- Online web-based games
- Social networking
- Calendar management
- Locally installed games
- Educational apps
- Video editing
- Internet phone

New Experiences

- Accelerated Internet and HD Video
- Simplified Content Management
- Immersive Gaming

Source: IDC's 2009 Consumer PC Buyer Survey
**PEOPLE PREFER VISUAL COMMUNICATIONS**

**Verbal Perception**
Words are processed at only 150 words per minute

**Visual Perception**
Pictures and video are processed 400 to 2000 times faster

**Augmenting Today’s Content:**
- Rich visual experiences
- Multiple content sources
- Multi-Display
- Stereo 3D
THE EMERGING WORLD OF NEW DATA RICH APPLICATIONS

The Ultimate Visual Experience™
Fast Rich Web content, favorite HD Movies, games with realistic graphics

Using photos
• Viewing & Sharing
• Search, Recognition, Labeling?
• Advanced Editing

Using video
• DVD, BLU-RAY™, HD
• Search, Recognition, Labeling
• Advanced Editing & Mixing

Music
• Listening and Sharing
• Editing and Mixing
• Composing and compositing

Communicating
• IM, Email, Facebook
• Video Chat, NetMeeting

Gaming
• Mainstream Games
• 3D games

Fast Rich Web content, favorite HD Movies, games with realistic graphics

Using photos
• Viewing & Sharing
• Search, Recognition, Labeling?
• Advanced Editing

Using video
• DVD, BLU-RAY™, HD
• Search, Recognition, Labeling
• Advanced Editing & Mixing

Music
• Listening and Sharing
• Editing and Mixing
• Composing and compositing
New Workload Examples: *Changing Consumer Behavior*

20 hours of video uploaded to YouTube every minute

Approximately 9 billion video files owned are high-definition

50 million + digital media files added to personal content libraries every day

1000 images are uploaded to Facebook every second
Why Heterogeneous Systems: *Parallelism and Power*

- **Changing/Emerging Workloads**
  - Visual communication providing ever increasing data parallel workloads
  - Mobile form factors are increasing demand on supporting data centers
  - Computational capabilities enabling new forms of Human interaction
    - More data parallel workloads with nested data parallel content

- **Technology Advances (Denser not Faster Designs)**
  - Moore’s law is alive, and transistor density continues
    - But not for Metal interconnects of dense transistors
  - Cost and time to market are increasing with future technologies
  - Process/Metal Interconnect is limiting voltage reductions and frequency scaling
CHALLENGES FOR FUTURE HETEROGENEOUS SYSTEMS:

- **Power & Thermal**
  - All Day Portable Devices, Low Power Data Centers

- **Memory Systems**
  - Bandwidth, Addressing, Virtualization, Coherency & Consistency

- **Scheduling and Quality of Services**
  - Concurrency, User Scheduling, Advanced Synchronization

- **Programming Models**
  - Multiple ISA, Compilers, Runtimes, Tools, Libraries
**Physical Integration**
- Integrate CPU & GPU in silicon
- Unified Memory Controller
- Common Manufacturing Technology

**Optimized Platforms**
- GPU Compute C++ support
- User mode scheduling
- Bi-Directional Power Mgmt between CPU and GPU

**Architectural Integration**
- Unified Address Space for CPU and GPU
- GPU uses pageable system memory via CPU pointers
- Fully coherent memory between CPU & GPU

**System Integration**
- GPU compute context switch
- GPU graphics pre-emption
- Quality of Service
- Extend to Discrete GPU
FUSION SYSTEM ARCHITECTURE – AN OPEN PLATFORM

- Open Architecture with published specifications
  - FSAIL virtual ISA
  - FSA memory model
  - FSA architected dispatch

- ISA agnostic for both CPU and GPU

- Invited partners to join AMD, in all areas
  - Hardware companies
  - Operating Systems
  - Tools and Middleware
  - Applications

- FSA ARB will be formed
**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

Primitive Scaling ➞ Multiple Primitive Pipelines
Pixel Scaling ➞ Multiple Screen Partitions
Multi-task graphics engine use of unified shader
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
COMPUTE UNIT ARCHITECTURE

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

4 CU Shared 16KB Scalar Read Only L1
4 CU Shared 32KB Instruction L1

AMD/CMU 2011   Fusion/FSA/AMD Graphics Core Next  |  August 2011
### SOME CODE EXAMPLES (1)

**float fn0(float a, float b)**

```c
float fn0(float a, float b) {
    if (a > b)
        return ((a - b) * a);
    else
        return ((b - a) * b);
}
```

**Optional:**

Use based on the number of instruction in conditional section.
- Executed in branch unit

### Optional

```
// 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
```
INSTRUCTION BUFFERING & FETCH

- Program Counter (PC) / Wavefront (Wave) – Unit of execution, 64 Work-items
  - Each Work item can follow a unique code path via predication
  - At least 10 Waves/SIMD, 40 Waves/CU ➔ 2560 Work-items per CU
- Independent Instruction Buffer Per Wave
- Instruction fetch Arbitration – Cycle per SIMD (age, priority, emptiness)
- I-Cache 32 KB – 64B lines, 4 Way Assoc, Least Recently Used (LRU)
  - Peak Bandwidth per CU is 32 bytes/cycle (Optimally 8 instructions)
- Special Instructions consumed in 1 cycle in the Instruction Buffers
  - s_nop, s_sleep, s_waitcnt, s_barrier, s_setprio, s_setvskip, s_halt
- 16 Work Group Barrier resources supported per CU

4 CU Shared 32KB Instruction L1 ➔ R/W L2
INSTRUCTION ARBITRATION AND DECODE

- A Kernel freely mixes instruction types (Simplistic Programming Model, no weird rules)
  - Scalar/Scalar Memory, Vector, Vector Memory, Shared Memory, etc.

- A CU will issue the instructions of a kernel for a wave-front sequentially
  - Use of predication & control flow enables any single work-item a unique execution path

- Every clock cycle, waves on one SIMDs are considered for instruction issue.
- At most, one instruction from each category may be issued.
- At most one instruction per wave may be issued.
- Up to a maximum of 5 instructions can issue per cycle, not including “internal” instructions.
  - 1 Vector Arithmetic Logic Unit (ALU)
  - 1 Scalar ALU or Scalar Memory Read
  - 1 Vector memory access (Read/Write/Atomic)
  - 1 Branch/Message - s_branch and s_cbranch_<cond>
  - 1 Local Data Share (LDS)
  - 1 Export or Global Data Share (GDS)
  - 1 Internal (s_nop, s_sleep, s_waitcnt, s_barrier, s_setprio)
**Branch and Message Unit**

- Independent scalar assist unit to handle special classes of instructions concurrently
  - Branch
    - Unconditional Branch \((s_{\text{branch}})\)
    - Conditional Branch \((s_{\text{cbranch}<\text{cond}>})\)
      - Condition \(\Rightarrow SCC==0, SCC=1, EXEC==0, EXEC!=0, VCC==0, VCC!=0\)
    - 16-bit signed immediate dword offset from PC provided
  - Messages
    - \(s_{\text{msg}}\) CPU interrupt with optional halt (with shader supplied code and source),
    - debug msg (perf trace data, halt, etc)
    - special graphics synchronization and resource management messages
**INTEGER SCALAR UNIT**

- A fully functional “scalar unit” with independent arbitration and decode
  - One scalar ALU or scalar memory read instruction processed per cycle
  - 32/64 bit Integer ALU with memory read support
  - 512 SGPR per SIMD shared between waves, \{SGPR\(n+1\), SGPR\} pair provide 64 bit register
- Scalar Data Cache 16 KB – 64B lines, 4 Way Assoc, LRU replacement policy
  - Peak Bandwidth per CU is 16 bytes/cycle
- Multi-Precision (MP) Single Instruction Multiple Data (SIMD) Units
  - 16 wide Single Precision IEEE floats or 32-bit integers operations per cycle
  - Selectable Double Precision rate options (determined at product build/configuration time)
  - 256 VGPRs shared across waves in SIMD, adjacent pairs form 64 bit registers

Vector Unit Registers

Vector GPR's (vgpr's) 0..255
Total vgpr count across 10 wavefronts owned by 1 simd may not exceed 256.
Organized as 64 words of 32 bits.
Two consecutive vgpr's can be combined for 64 bit words or 4 consecutive for 128 bit words and access.
### 4 Way VLIW SIMD

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</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</td>
<td>4SIMD * 1 ALU Operation ➔ Occupancy limitations</td>
</tr>
<tr>
<td>limitations</td>
<td></td>
</tr>
<tr>
<td>3 SRC GPRs, 1 Vector Destination</td>
<td>3 SRC GPRs, 1 Vector\1 Scalar 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</td>
<td>VALU Instruction Bandwidth ➔ 1-2 dwords/cycle</td>
</tr>
<tr>
<td>dwords/clk)</td>
<td></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;</td>
<td>Simplified assembly creation, analysis, &amp; debug</td>
</tr>
<tr>
<td>debug</td>
<td></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>

### 4 Non-VLIW SIMD

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</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</td>
<td>4SIMD * 1 ALU Operation ➔ Occupancy limitations</td>
</tr>
<tr>
<td>limitations</td>
<td></td>
</tr>
<tr>
<td>3 SRC GPRs, 1 Vector Destination</td>
<td>3 SRC GPRs, 1 Vector\1 Scalar 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</td>
<td>VALU Instruction Bandwidth ➔ 1-2 dwords/cycle</td>
</tr>
<tr>
<td>dwords/clk)</td>
<td></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;</td>
<td>Simplified assembly creation, analysis, &amp; debug</td>
</tr>
<tr>
<td>debug</td>
<td></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>
LOCAL SHARED MEMORY (LDS)

- 64 kb, 32 bank Shared Memory
- Direct mode
  - Vector Instruction Operand ➔ 32/16/8 bit broadcast value
  - Graphics Interpolation @ rate, no bank conflicts
- Index Mode – Load/Store/Atomic Operations
  - Bandwidth Amplification, upto 32 – 32 bit lanes serviced per clock peak
  - Direct decoupled return to VGPRs
  - Hardware conflict detection with auto scheduling
- Software consistency/coherency for thread groups via hardware barrier
- Fast & low power vector load return from R/W L1
VECTOR EXPORT INSTRUCTIONS

- Exports move data from 1-4 VGPRs to Graphic Pipeline
  - Color (MRT0-7), Depth, Position, and Parameter
- Global shared memory Ops
VECTOR MEMORY OPERATIONS

- Read/Write/Atomic request are routed to R/W cache hierarchy
  - Variable size addresses /data (4-128b, 8-64b, 16-32b)/cycle
- Addressing unit
  - Address coalescing
  - Image and filter dependant address generation
  - Write Data format conversion
- L1 16KB R/W Vector Data cache
  - 64B cache line, 4 sets x 64 way, LRU Replacement
  - Read-Write Cache (write-through at end of wavefront)
  - Decompression on cache read out
- Return data processing to VGPRs
  - Data filtering, format conversions
  - Optional gather return directly to LDS
GRAPHICS CORE NEXT ARCHITECTURE

NEW COMPUTE UNIT ARCHITECTURE

- Simpler ISA compared to previous generation
  - 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
GRAPHICS CORE NEXT ARCHITECTURE

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
  - Load acquire / Store release consistency controls
  - Read/Write Cache Hierarchy improves I/O characteristics
  - Flexible vector load, store, and remote atomic operations
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 HEREIN, 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.