RADEON GRAPHICS

AMD

White Paper | AMD GRAPHICS CORES NEXT (GCN) ARCHITECTURE

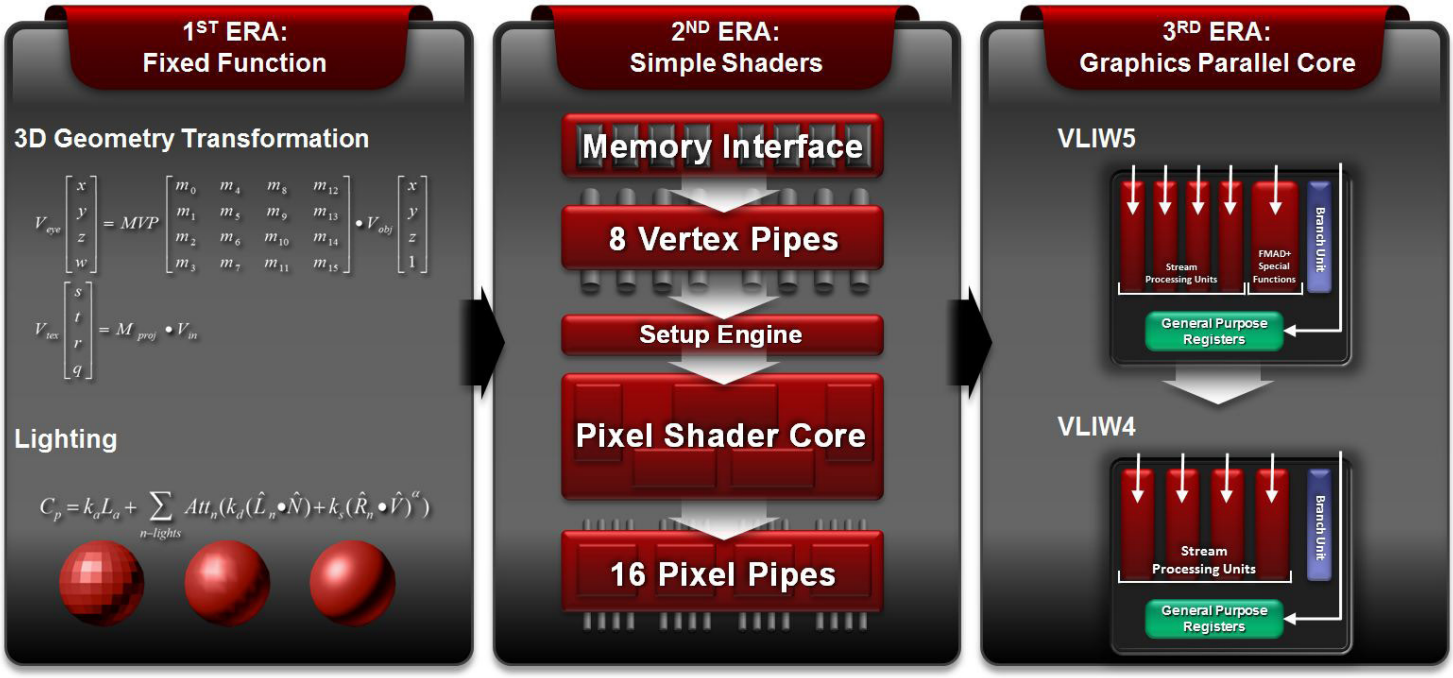
Table of Contents

June, 2012

INTRODUCTION

Over the last 15 years, graphics processors have continually evolved and are on the cusp of becoming an integral part of the computing landscape. The first designs employed special purpose hardware with little flexibility. Later designs introduced limited programmability through shader programs, and eventually became highly programmable throughput computing devices that still maintained many graphics-specific capabilities.

Figure 1: GPU Evolution



The performance and efficiency potential of GPUs is incredible. Games provide visual quality comparable to leading films, and early adopters in the scientific community have seen an order of magnitude improvement in performance, with high-end GPUs capable of exceeding 4 TFLOPS (Floating point Operations per Second). The power efficiency is remarkable as well; for example, the AMD Radeon™ HD 7770M GPU achieves over 1 TFLOPS with a maximum power draw of 45W.

Key industry standards, such as OpenCL™, DirectCompute and C++ AMP recently have made GPUs accessible to programmers. The challenge going forward is creating seamless heterogeneous computing solutions for mainstream applications. This entails enhancing performance and power efficiency, but also programmability and flexibility. Mainstream applications demand industry standards that are adapted to the modern ecosystem with both CPUs and GPUs and a wide range of form factors from tablets to supercomputers.

AMD's Graphics Core Next (GCN) represents a fundamental shift for GPU hardware and is the architecture for future programmable and heterogeneous systems. GCN is carefully optimized for power and area efficiency at the 28nm node and will scale to future process technologies in the coming years. The heart of GCN is the new Compute Unit (CU) design that is used in the shader array, which is described in detail in the following sections.

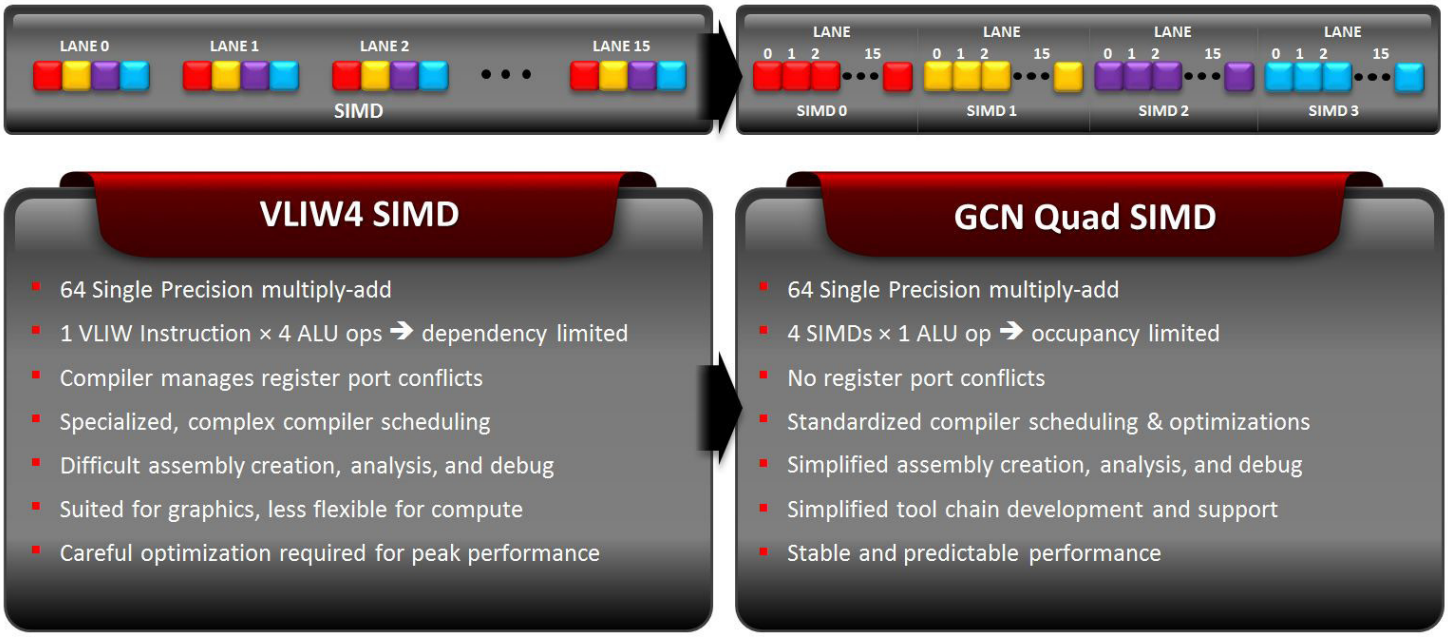
COMPUTE UNIT OVERVIEW

Compute units are the basic computational building block of the GCN Architecture. These CUs implement an entirely new instruction set that is much simpler for compilers and software developers to use and delivers more consistent performance than previous designs.

The shader arrays in earlier generations of AMD GPUs consisted of a number of SIMD engines, each of which consisted of up to 16 ALUs. Each ALU could execute bundles of 4 or 5 independent instructions co-issued in a VLIW (Very Long Instruction Word) format, with the shader compiler being largely responsible for scheduling and finding co-issue opportunities. SIMD engines were issued groups of 64 work items, called wavefronts, and would execute one wavefront at a time. This design aligned well with a data flow pattern is very common in graphics processing (for manipulating RGBA color values in a pixel shader, for example), making it possible to sustain high levels of utilization in most cases. However, the underlying data formats can be more complex and unpredictable for general purpose applications, making it more difficult to consistently find sets of 4 or 5 independent operations that could execute in parallel every cycle and keep the processing resources fully utilized.

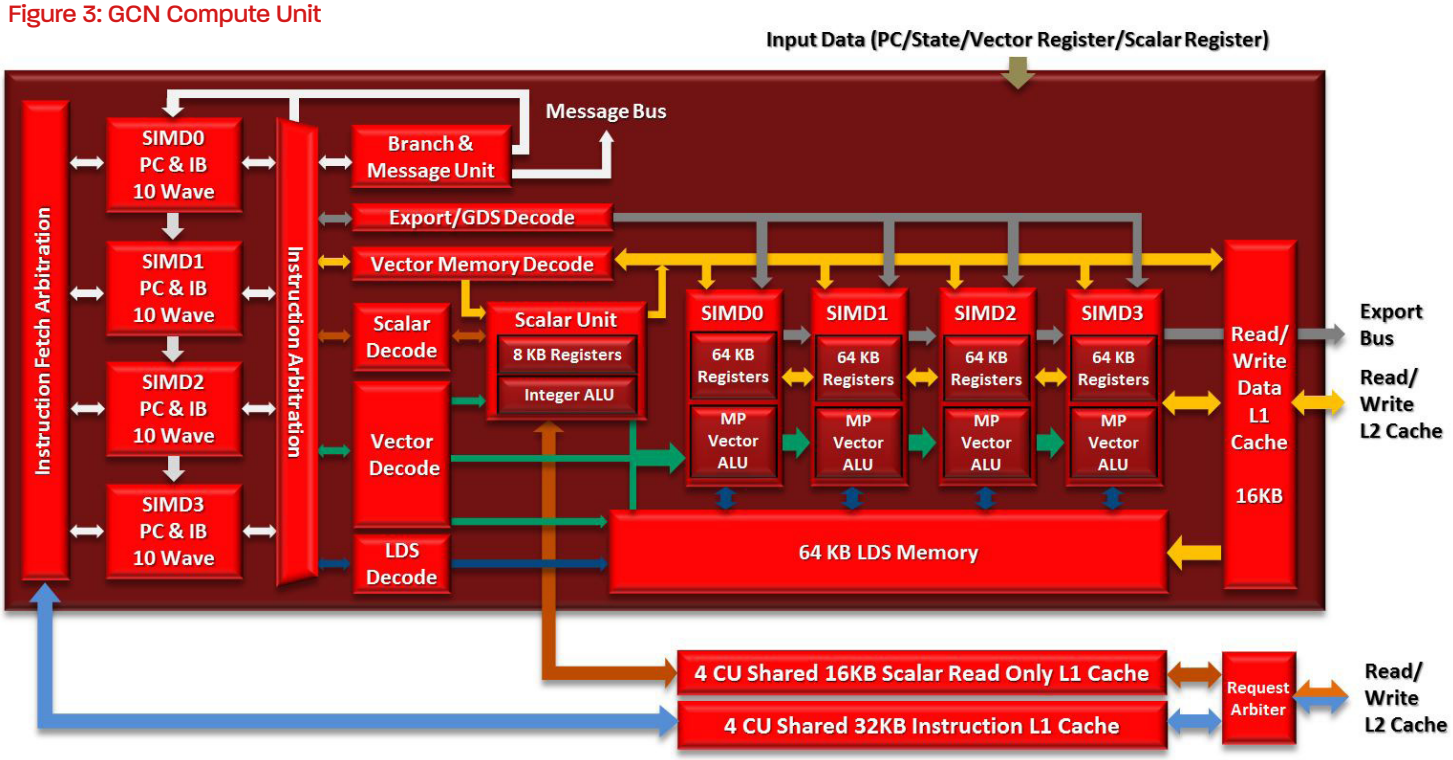
In GCN, each CU includes 4 separate SIMD units for vector processing. Each of these SIMD units simultaneously executes a single operation across 16 work items, but each can be working on a separate wavefront. This places emphasis on finding many wavefronts to be processed in parallel, rather than relying on the compiler to find independent operations within a single wavefront.

Figure 2: VLIW4 vs. GCN



For efficiency, the SIMDs in each GCN compute unit have a combination of private and shared resources. The instruction buffering, registers and vector ALUs are private for each of the 4 SIMDs to sustain high performance and utilization. Other resources, such as the front-end, branch unit, and data cache are shared between the SIMDs to achieve area and power efficiency.

Figure 3: GCN Compute Unit



Another crucial innovation in GCN is coherent caching. Historically, GPUs have relied on specialized caches (such as read-only texture caches) that do not maintain a coherent view of memory. To communicate between cores within a GPU, the programmer or compiler must insert explicit synchronization instructions to flush shared data back to memory. While this approach simplifies design, it increases overhead for applications which share data. GCN is tailored for general purpose workloads, where algorithms that communicate between cores are common. The cache coherency protocol shares data through the L2 cache, which is significantly faster and more power efficient than using off-chip graphics memory.

In tandem with cache coherency, GCN introduces virtual memory through a combination of hardware and driver support. Virtual memory eliminates the most challenging aspects of memory management and opens up new capabilities. AMD's unique expertise in both high performance graphics and microprocessors was particularly beneficial, as GCN's virtual memory model has been carefully defined to be compatible with x86. This simplifies moving data between the CPU and the discrete GPU in initial products. More importantly, it paves the way for a single address space that is seamlessly shared by CPUs and GPUs. Sharing, rather than copying, data is vital for performance and power efficiency and a critical element in heterogeneous systems such as AMD's Accelerated Processing Units (APUs).

CU FRONT-END

In GCN, each SIMD unit is assigned its own 40-bit program counter and instruction buffer for 10 wavefronts. The whole CU can thus have 40 wavefronts in flight, each potentially from a different work-group or kernel, which is substantially more flexible than previous designs. This means that a GCN GPU with 32 CUs, such as the AMD Radeon™ HD 7970, can be working on up to 81,920 work items at a time.

A cluster of up to 4 Compute Units share a single 32KB L1 instruction cache that is 4-way associative and backed by the L2 cache. Cache lines are 64B long and typically hold 8 instructions. When the cache is full, a new request will evict the Least Recently Used (i.e. LRU replacement) cache line to make room for new instructions. The shared L1 instruction cache has 4 banks, and can sustain 32B instruction fetch per cycle to all 4 Compute Units. Instruction fetching is arbitrated between SIMDs within a CU based on age, scheduling priority and utilization of the wavefront instruction buffers.

Once instructions have been fetched into the wavefront buffers, the next step is decoding and issuing instructions. The compute unit selects a single SIMD to decode and issue each cycle, using round-robin arbitration. The selected SIMD can decode and issue up to 5 instructions from the 10 wavefront buffers to the execution units. Additionally, a special instruction (e.g. NOPs, barriers, halts, skipping a predicated vector instruction, etc.) can be executed within the wavefront buffers without using any functional units. Each CU has 16 buffers to track barrier instructions, which force a wavefront to synchronize globally.

The CU front-end can decode and issue seven different types of instructions: branches, scalar ALU or memory, vector ALU, vector memory, local data share, global data share or export, and special instructions. Only issue one instruction of each type can be issued at a time per SIMD, to avoid oversubscribing the execution pipelines. To preserve in-order execution, each instruction must also come from a different wavefront; with 10 wavefronts for each SIMD, there are typically many available to choose from. Beyond these two restrictions, any mix is allowed, giving the compiler plenty of freedom to issue instructions for execution.

The CU front-end can issue five instructions every cycle, to a mix of six vector and scalar execution pipelines using two register files. The vector units provide the computational power that is critical for graphics shaders as well as general purpose applications. Together with the special instructions that are handled in the instruction buffers, the two scalar units are responsible for all control flow in the GCN Architecture.

SCALAR EXECUTION AND CONTROL FLOW

The new scalar pipelines in the GCN compute unit are essential for performance and power efficiency. Control flow processing is distributed throughout the shader cores, which reduces latency and also avoids the power overhead for communicating with a centralized scheduler. This is particularly beneficial for general purpose applications, which tend to have more complex control flow than graphics.

Each compute unit has an 8KB scalar register file that is divided into 512 entries for each SIMD. The scalar registers are shared by all 10 wavefronts on the SIMD; a wavefront can allocate 112 user registers and several registers are reserved for architectural state. The registers are 32-bits wide, and consecutive entries can be used to hold a 64-bit value. This is essential for wavefront control flow; for example, comparisons will generate a result for each of the 64 work-items in a wavefront.

The first scalar pipeline handles conditional branches, using a 16-bit signed offset that is encoded in the instruction. It is also responsible for handling interrupts and certain types of synchronization. Interrupts are a brand new feature in GCN that is critical for debugging, as they can redirect control flow for breakpoints.

The second scalar pipeline is a full integer ALU that acts as an address generation unit (AGU) to read from the scalar data cache. The actual ALU is 64 bits wide. This pipeline helps with a variety of control flow instructions, including jumps, calls and returns. These are handled by replacing (or swapping) the old program counter with a 64-bit value from two adjacent registers. Predication is also managed by the scalar ALU, using assist hardware to manage the fork and join operations.

The scalar L1 data cache is a read only structure. Since the scalar pipelines are primarily meant for control flow, there is no need to write results back to memory. Its organization is fairly similar to the L1 instruction cache. The 16KB scalar data L1 is 4-way associative with 64B lines and LRU replacement; it is also shared between a cluster of up to 4 Compute Units and backed by the L2 cache. Scalar reads are 4-64B long and fill adjacent scalar registers. It has 4 banks, with a throughput of 16B/cycle for each of the 4 Compute Units. The scalar data L1 cache replaces the constant cache in previous generations, as it is significantly more flexible.

VECTOR EXECUTION

Much of GCN's incredible processing performance comes from the highly parallel SIMDs, which perform the calculations for general purpose applications and cutting-edge graphics. The SIMDs were totally overhauled for GCN to deliver more programmability and consistent performance.

The earlier VLIW compute unit architecture had a single SIMD, but they were required to execute parallel operations from a single wavefront. This had two direct consequences. First, the performance was fundamentally limited by the compiler; the hardware was substantially underutilized for workloads that have little parallelism within a wavefront. If only two instructions in a wavefront could execute in parallel, then half the performance is unused. The second consequence was that the compiler had to carefully schedule reads and writes to the register file; since any port conflicts would further decrease utilization. To avoid potential port conflicts, wavefronts could not issue ALU operations back-to-back, so wavefronts were interleaved to hide the latency. The VLIW architecture was relatively good for graphics, but had unpredictable performance on complex workloads and required a great deal of software tuning.

The GCN Architecture is simpler and far better at delivering performance. The fundamental change was avoiding the unpredictable instruction level parallelism within a wavefront, and relying on software to provide a sufficient number of data parallel wavefronts to saturate the hardware. This is a tremendous benefit for applications that use GPUs, particularly general purpose workloads.

VECTOR REGISTERS

One of the biggest advantages of GCN's architecture is a much simpler and higher performance vector register file design. Since each SIMD is executing an independent wavefront, the register file can be partitioned into four independent slices.

The vector General Purpose Registers (vGPRs) contain 64 lanes that are up to 32-bits wide. Adjacent vGPRs are combined for 64-bit or 128-bit data. Each SIMD has a 64KB partition of vGPRs, so the total number of registers for a CU has stayed constant. Each SIMD's partition is heavily banked and can read X registers and write Y registers.

The massive bandwidth from the register file eliminates port conflicts and means that vector ALU instructions can be issued back-to-back without interleaving within a SIMD. This simplifies the compiler's job considerably and is tremendously beneficial for developers writing high performance code.

VECTOR ALUS

Each SIMD includes a 16-lane vector pipeline that is predicated and fully IEEE-754 compliant for single precision and double precision floating point operations, with full speed denormals and all rounding modes. Each lane can natively executes a single precision fused or unfused multiply-add or a 24-bit integer operation. The integer multiply-add is particularly useful for calculating addresses within a work-group. A wavefront is issued to a SIMD in a single cycle, but takes 4 cycles to execute operations for all 64 work items.

GCN also adds new media and image processing instructions in the vector ALUs. Specifically, there are two new instructions: a 4x1 sum-of-absolute-differences (SAD) and quad SAD that operate on 32-bit pixels with 8-bit colors. Using these new instructions, a Compute Unit can execute 64 SADs/cycle which translates into 256 operations per clock. These instructions are very powerful and essential for emerging GPU applications such as gesture recognition or video search. For example, AMD's Steady Video 2.0 technology uses the new SAD and QSAD instructions to remove shakiness from recorded or streaming videos in real time.

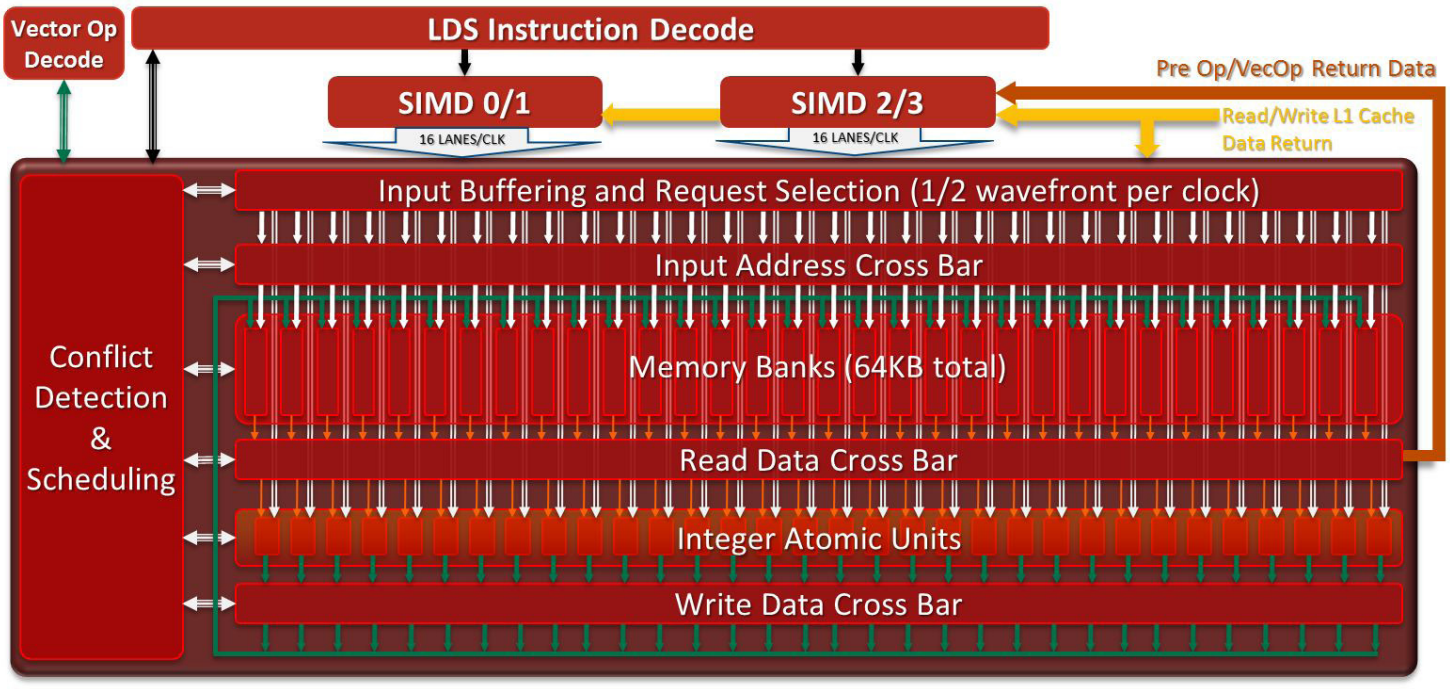
Double precision and 32-bit integer instructions run at a reduced rate within a SIMD. The GCN Architecture is flexible and double precision performance varies from 1/2 to 1/16 of single precision performance, increasing the latency accordingly. The double precision and 32-bit integer performance can be configured for a specific GCN implementation, based on the target application.

More complicated instructions such as 64-bit transcendental functions and IEEE divides are supported by microcode. The SIMDs also take advantage of the improved branch unit to provide floating point exceptions in hardware, and use scalar GPRs for vector condition codes.

LOCAL DATA SHARE AND ATOMICS

For a throughput computing platform like GCN, communication and synchronization is vital for high performance, especially for emerging general purpose applications. The local data share is an explicitly addressed memory that acts as a third register file specifically for synchronization within a work-group or interpolation for graphics.

Figure 4: Local Data Share (LDS)



The LDS capacity in GCN has doubled to 64KB with 16 or 32 banks (depending on the product). Each bank contains 512 entries which are 32-bit wide. A bank can read and write a 32-bit value across an all-to-all crossbar and swizzle unit that includes 32 atomic integer units. Typically, the LDS will coalesce 16 lanes from two different SIMDs each cycle, so two wavefronts complete every 4 cycles. Conflicts are automatically detected across 32 lanes from a wavefront and resolved in hardware. An instruction which accesses different elements in the same bank takes additional cycles to finish. Broadcasts are transparent and 8, 16 or 32-bit data can be used as input operands for the vector ALU instructions without any penalties.

For graphics, the LDS is used to perform full rate interpolation on texture data, and conflicts are guaranteed not to occur because of the access pattern. For general purpose compute kernels, a SIMD can load or store data in the LDS to avoid polluting the cache hierarchy with scatter and gather accesses, or use it to amplify cache bandwidth. Additionally, the atomic units are essential for high performance reductions within a work-group and can do floating point max, min and compare and swap operations.

The LDS instructions take an address, two data values and a destination. The address is from a vGPR and the destination can be a vGPR or a SIMD that is directly reading from the LDS. The two data values can come from the L1 data cache (for a store to LDS) or vGPRs (loads or stores). Unlike other designs, the dedicated pipeline avoids using vector ALU instructions for data movement, such as an LDS load to registers.

EXPORT

The Export unit is the Compute Unit's window to the fixed function graphics hardware as well as the global data share (GDS). When all the computations have been finished, the results are typically sent to other units in the graphics pipeline. For example, once pixels have been shaded, they are typically sent to the render back-ends for depth and stencil tests and any blending before the final output to the display. The export unit writes results from the programmable stages of the graphics pipeline to the fixed function ones, such as tessellation, rasterization and the render back-ends.

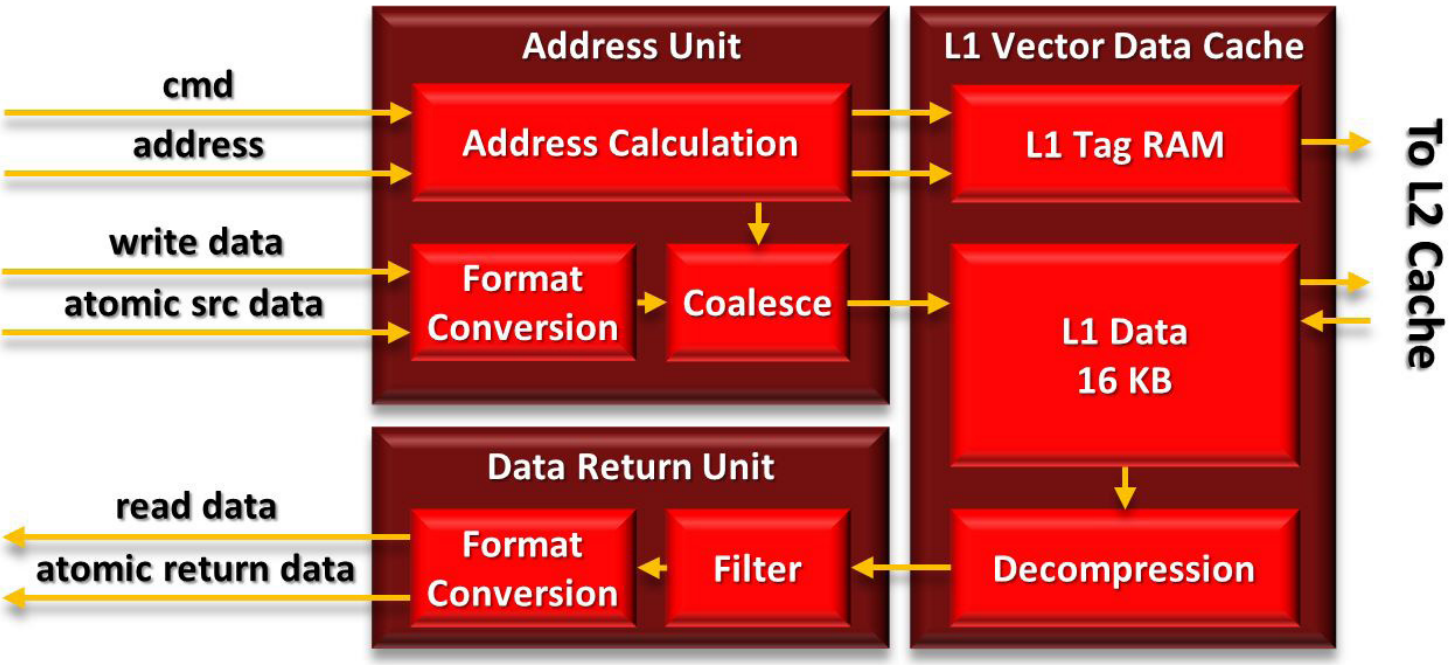
The GDS is identical to the local data shares, except that it is shared by all compute units, so it acts as an explicit global synchronization point between all wavefronts. The atomic units in the GDS are a bit more complex and can handle ordered count operations.

VECTOR MEMORY

While the SIMDs in each Compute Unit can execute 128 floating point operations per clock cycle, delivering enough bandwidth to match the impressive computational resources. The most significant changes in GCN were unquestionably in the cache hierarchy, which has morphed from a tightly focused graphics design into a high performance and programmable hierarchy that is both well suited for general purpose applications and ready for integration with x86 processors. The changes start within the Compute Unit, but extend throughout the entire system.

The GCN memory hierarchy is a unified read/write caching system with virtual memory support and excellent atomic operation performance. This represents a significant improvement over the separate read caching and write buffering paths used in previous generations. Vector memory instructions support variable granularity for addresses and data, ranging from 32-bit data to 128-bit pixel quads.

Figure 5: Vector Memory



The L1 data (L1D) cache is 16KB and 4-way set associative with 64B lines and LRU replacement. It can be coherent with the L2 and other caches, with an extremely relaxed consistency model. Conceptually, the L1D cache is work-group coherent with eventual global consistency through the L2 cache. The L1D has a write-through, write-allocate design with a dirty byte mask. Cache lines are written back to the L2 when all 64 stores in a wavefront instruction have finished. Lines with all dirty data are also kept in the L1D, while any partially clean line is evicted from the L1D. There are also special coherent load instructions that fetch from the L2 cache, to ensure that the most recent value is used.

Once the AGU has calculated a coalesced address, the request probes the L1D cache tags. On a hit, the cache reads out a full 64B line. With fully coalesced requests, this corresponds to 16 data values or 1/4 of a wavefront, although poor locality may require additional cycles. For computational workloads, the cache line is written into the vGPRs or the LDS.

Stores to the L1D cache are a little more complicated. The write data has to be converted to the appropriate storage format and then the write addresses are coalesced into as few separate transactions as possible before hitting the cache and eventually writing through to the L2 cache.

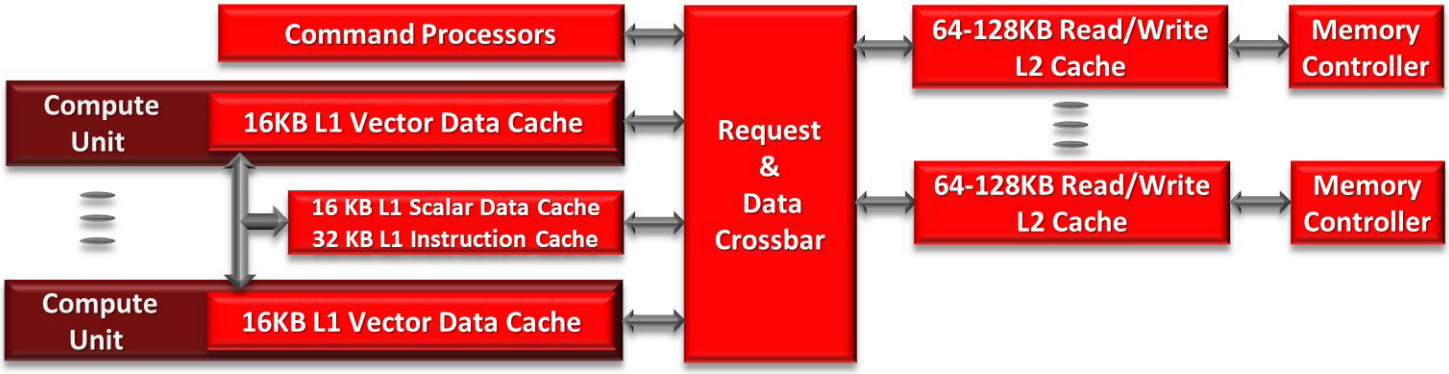
If a memory request misses in the L1D cache, then it is sent to the unified and coherent L2 cache which sits outside of the shader cores and is associated with the memory controllers.

To improve efficiency and reduce overhead, the flexible memory hierarchy is re-used for graphics, with a little dedicated hardware. The address generation unit receives 4 texture addresses per cycle, and then calculates 16 sampling addresses for the nearest neighbors. The samples are read from the L1 data cache and decompressed in the Texture Mapping Unit (or TMU). The TMU then filters adjacent samples to produce as many as 4 final interpolated texels per clock. The TMU output is converted to the desired format and ultimately written into the vector register file for further use. The format conversion hardware is also used for writing some values out to memory in graphics kernels.

L2 CACHE, COHERENCY AND MEMORY

The distributed L2 cache in GCN is the central point of coherency in the GPU. It acts as a backstop for the read-only L1 instruction and scalar caches that are shared by a cluster of CUs, as well as the L1 data caches in every CU. The L2 cache is physically partitioned into slices that are coupled to each memory channel, and accesses flow through a crossbar fabric from the Compute Units to the cache and memory partitions.

Figure 6: Cache Hierarchy



Like the L1 data cache, the L2 is virtually addressed, so no TLBs are required at all. The L2 cache is 16 way associative, with 64B cache lines and LRU replacement. It is a write-back and write allocate design, so it absorbs all the write misses from the L1 data cache. Each L2 slice is 64-128KB and can send a 64B cache line to the L1 caches.

One of the new big advantages of the coherent L2 cache is that it is a very natural place to execute global atomic operations and synchronize between different wavefronts. While the LDS can be used for atomic operations within a wavefront, at some point, the results from different wavefronts need to be combined. This is exactly where the L2 comes into play. An L2 slice can execute as many as 16 atomic operations to a cache line each cycle.

The overall coherency protocol in GCN is a hybrid model that melds the incredible performance and bandwidth of a GPU with the programmability of a traditional CPU and an eye towards further integration. Conceptually, the L1 data cache maintains strict coherency for local accesses within a work-group. At the end of a wavefront, or when a barrier is invoked, the data is written through to the L2 and becomes globally coherent across the GPU. This model is often described as relaxed consistency and has tremendous advantages. The dozens of local accesses have low overhead and high performance, but the L2 provides programmer friendly coherency.

Equally important, the cache hierarchy was designed to integrate with x86 microprocessors. The GCN virtual memory system can support 4KB pages, which is the natural mapping granularity for the x86 address space - paving the way for a shared address space in the future. In fact, the IOMMU used for DMA transfers can already translate requests into the x86 address space to help move data to the GPU and this functionality will grow over time. Additionally, the caches in GCN use 64B lines, which is the same size as x86 processors. This sets the stage for heterogeneous systems to transparently share data between the GPU and CPU through the traditional caching system, without explicit programmer control.

The memory controllers tie the GPU together and provide data to nearly every part of the system. The command processor, ACEs, L2 cache, RBEs, DMA Engines, PCI Express™, video accelerators, Crossfire interconnects and display controllers all have access to local graphics memory. Each memory controller is 64-bits wide and composed of two independent 32-bit GDDR5 memory channels. For lower cost products, GDDR5 can be replaced with DDR3 memory as well. For large memory footprints, the GDDR5 controller can operate in clamshell mode to use two DRAMs per channel, doubling the capacity.

PROGRAMMING MODEL

The power of GPUs is tapped through Application Programming Interfaces (APIs), abstractions which provide a consistent interface for developers. Crucially, APIs foster compatibility across generations of hardware and software, as well as across different families of GPUs. The goal is to free programmers from worrying about the underlying hardware, and enable them to focus on their application.

The new GCN ISA was designed to facilitate industry standards. AMD's Heterogeneous System Architecture (HSA) is envisioned as a model for heterogeneous computing. It defines how CPUs and GPUs communicate and includes a virtual ISA (the HSA Intermediate Language), which is hardware agnostic. HSAIL code is dynamically compiled for the underlying hardware, and thus compatible with CPUs and GPUs from any vendor. GCN fully supports HSAIL, in part because of the shift to a more flexible instruction set.

GCN has full support for industry standard compute APIs. In particular, it is the first GPU that is compatible with OpenCL™ 1.2, DirectCompute 11.1 and C++ AMP. Using these standards, programmers can write applications that target any operating system and are compatible across different hardware. These industry standards ensure a viable long term roadmap for developers and are essential to avoid proprietary programming models or hardware platforms with an uncertain future.

On the graphics front, GCN is ready for DirectX 11.1, which will debut with Windows® 8 and includes three particularly key features. The first is target-independent rasterization, which moves rasterization for 2D images to the fixed function graphics hardware. This offloads 2D anti-aliasing from the CPU, which reduces power while offering the same image quality. The second change is that general purpose data arrays (called Unordered Access Views) are available to all 6 types of programmable shaders and video acceleration. This extends full GPU programmability throughout the graphics and video APIs, rather than restricting it to compute and pixel shaders. Last, DirectX® 11.1 incorporates a standard API for Stereo 3D, replacing a fragmented ecosystem of vendor specific and third party middleware.

While many of the critical improvements in GCN are related to general purpose programmability, they are often beneficial in the context of graphics. For example, developers can use Partially Resident Textures (PRTs) to create a world with near-infinite texture detail. The GPU's virtual memory system will only load the necessary parts into memory. On GCN, up to 32TB of virtual memory is available for textures. As this dwarfs the available physical memory, the driver and GPU page in 64KB tiles of the texture as needed. This process is transparent, and relieves the programmer of explicitly preloading data into memory. PRTs are already used in leading edge game engines such as id Tech 5 and will be common in the next generation of game engines and hit games.

SYSTEM ARCHITECTURE

Ultimately, GCN will be used in a family of products, from tablets that are limited to 2-3W to supercomputers that occupy entire buildings. This represents a performance and power spectrum of roughly 100x, and dramatically different environments. High-end GPUs typically have a dedicated high bandwidth memory subsystem and may be operated in Crossfire, where rendering is split across multiple GPUs, for maximum performance. In contrast, a highly integrated System-on-Chip for tablets or PCs will share a memory controller between the CPU, GPU and many other on-die components. To handle this diversity, GCN's system architecture was redesigned for flexibility and scalability in both the general purpose shading cores and also the fixed function graphics hardware.

GCN is the first architecture to use PCI Express™ 3.0 for interfacing with the host processor. Discrete GPUs based on GCN, such as the AMD Radeon™ HD 7970, use a x16 link, which provides 32GB/s of bandwidth. This CPU to GPU interface is a bottleneck particularly for general purpose workloads, where massive data sets are moved between the two processors. GCN has dual bi-directional DMA engines, so that two streams of data can simultaneously use both directions of the PCI Express™ 3.0 link and efficiently use the available bandwidth.

The DMA engines are one area where AMD's experience with x86 microprocessors has paid off. GCN incorporates an I/O Memory Management Unit (IOMMU), which can transparently map x86 addresses for the GPU. This means that the DMA engines in GCN can easily access pageable CPU memory, without the overhead of address translation, to move data. The IOMMU is a step towards tighter heterogeneous integration and will evolve over time.

The GCN command processor is responsible for receiving high-level level API commands from the driver and mapping them onto the different processing pipelines. There are two main pipelines in GCN. The Asynchronous Compute Engines (ACE) are responsible for managing compute shaders, while a graphics command processor handles graphics shaders and fixed function hardware. Each ACE can handle a parallel stream of commands, and the graphics command processor can have a separate command stream for each shader type, creating an abundance of work to take advantage of GCN's multi-tasking.

Scheduling is another area where GCN is moving the industry forward. Virtual memory is a pre-requisite for multi-tasking, so that applications with competing demands for memory can co-exist safely. Multi-tasking has been the norm for computers since the late 1980's and incredibly valuable for programmer productivity. GCN builds on this basic infrastructure and uses multi-tasking to improve the utilization and efficiency of highly parallel GPUs.

The ACEs are responsible for all compute shader scheduling and resource allocation. Products may have multiple ACEs, which operate independently, to scale up or down in terms of performance. Each ACE fetches commands from cache or memory and forms task queues, which are the starting point for scheduling. Each task has a priority level for scheduling, ranging from background to real-time. The ACE will check the hardware requirements of the highest priority task and launch that task into the GCN shader array when sufficient resources are available.

Many tasks can be in-flight simultaneously; the limit is more or less dictated by the hardware resources. Tasks complete out-of-order, which releases resources earlier, but they must be tracked in the ACE for correctness. When a task is dispatched to the GCN shader array, it is broken down into a number of workgroups that are dispatched to individual compute units for execution. Every cycle, an ACE can create a workgroup and dispatch one wavefront from the workgroup to the compute units.

While ACEs ordinarily operate in an independent fashion, they can synchronize and communicate using cache, memory or the 64KB Global Data Share. This means that an ACE can actually form a task graph, where individual tasks have dependencies on one another. So in practice, a task in one ACE could depend on tasks on another ACE or part of the graphics pipeline. The ACEs can switch between tasks queue, by stopping a task and selecting the next task from a different queue. For instance, if the currently running task graph is waiting for input from the graphics pipeline due to a dependency, the ACE could switch to a different task queue that is ready to be scheduled. The ACE will flush any workgroups associated with the old task, and then issue workgroups from the new task to the shader array.

GRAPHICS ARCHITECTURE

The graphics command processor coordinates the traditional rendering pipeline. The 3D pipeline consists of several types of programmable shaders (e.g. vertex, hull, domain, geometry and pixel shaders) and a variety of fixed function hardware that manipulates triangles and pixels. The shader performance is quite scalable, so the real challenge lies with scaling the fixed function hardware.

3D rendering starts with primitive pipelines that assemble a single triangle per clock. The number of primitive pipelines in GCN can vary according to performance requirements. Multiple primitive pipelines will partition the screen space and rely on synchronization to maintain the correct triangle order. Assembled triangles are then sent to the shader array for vertex and hull shading. The latter initiates tessellation by changing from vertex to tessellation co-ordinates and setting control information.

The primitive pipelines also contain the fixed function hardware for the tessellation stage that actually subdivides a domain into 2-64 smaller objects. The tessellator in GCN is up to 4x faster than the earlier generation, with larger parameter caches that can spill to the coherent L2 cache instead of the much slower off-chip memory. After tessellation comes the domain shaders which convert the tessellated output back into standard vertices and then passes the results to the geometry shader.

The other major graphics functions are concentrated in the pixel pipeline. The GCN pixel pipelines scale by partitioning screen space into independent tiles. The first step is the actual rasterization that transforms vertices into pixels. Each rasterizer can read in a single triangle per cycle, and write out 16 pixels. Afterwards, the hierarchical Z-testing will eliminate any occluded pixels prior to pixel shading.

Once the pixels fragments in a tile have been shaded, they flow to the Render Back-Ends (RBEs). The RBEs apply depth, stencil and alpha tests to determine whether pixel fragments are visible in the final frame. The visible pixels fragments are then sampled for coverage and color to construct the final output pixels. The RBEs in GCN can access up to 8 color samples (i.e. 8x MSAA) from the 16KB color caches and 16 coverage samples (i.e. for up to 16x EQAA) from the 4KB depth caches per pixel. The color samples are blended using weights determined by the coverage samples to generate a final anti-aliased pixel color. The results are written out to the frame buffer, through the memory controllers.

The graphics pipeline is orchestrated using the same set of techniques as the ACEs. Each stage of the 3D pipeline can operate concurrently, as can any ACEs. The primitive and pixel pipelines are connected to the programmable GCN shaders through crossbar fabrics. The task queues synchronize different shaders and fixed function hardware through cache or memory.

The advantage of GCN's flexibility is evident in the first few products that have scaled across all four dimensions. The AMD Radeon™ HD 7970 splits the screen into 2 primitive pipelines and 4 pixel pipelines, with 32 compute units for shading and a 384-bit memory interface. The GCN pixel pipelines are organized into 2 RBEs and 3 memory controllers, a 50% boost in memory bandwidth. In contrast, the AMD Radeon™ HD 7770 GHz Edition has a single primitive pipeline, 2 pixel pipelines and 10 compute units. The pixel pipelines in the AMD Radeon™ HD 7770 GHz Edition also scaled back to 2 memory controllers, for a 128-bit wide interface.

Figure 7: AMD Radeon™ HD 7970

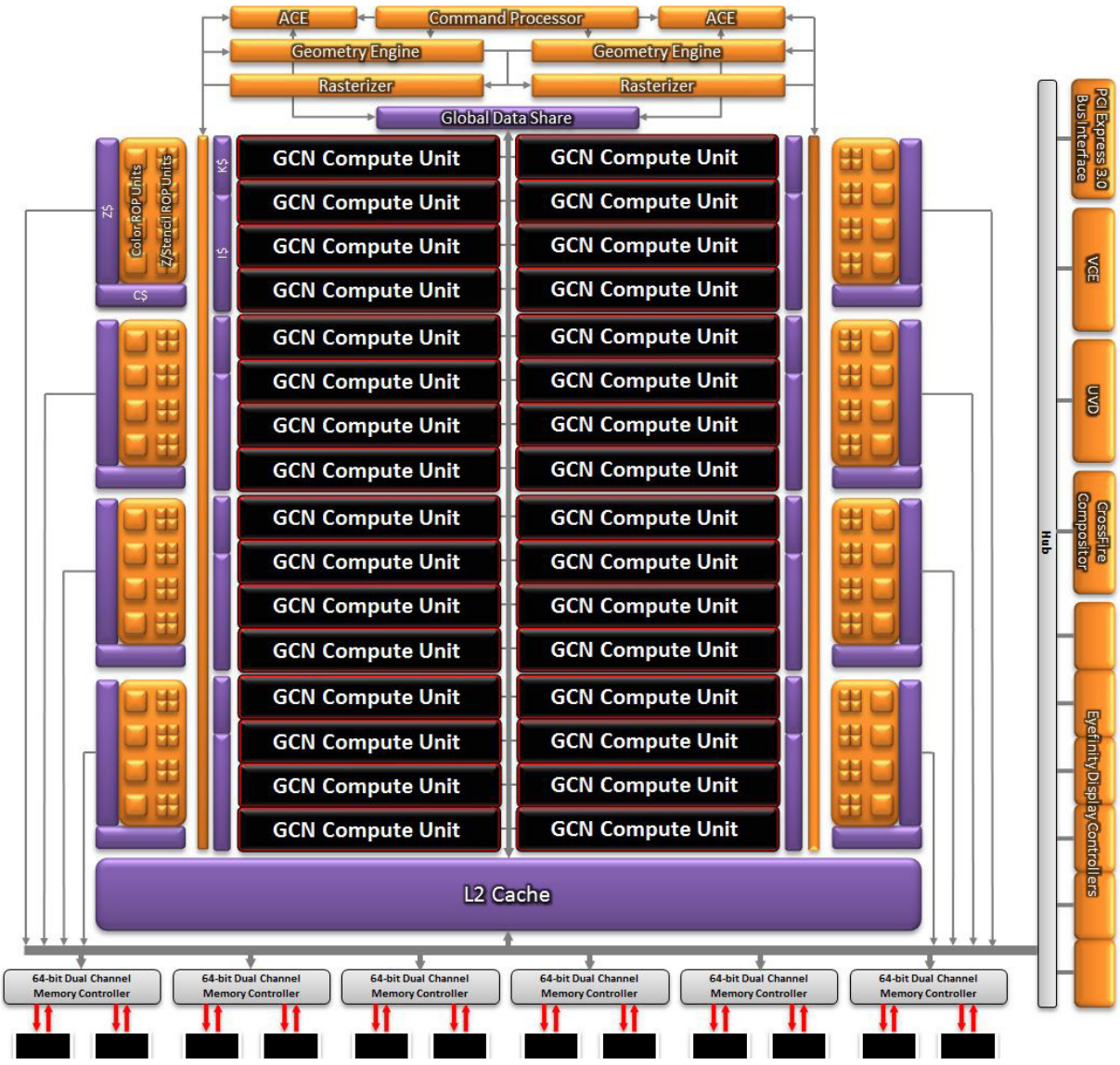


Figure 8: AMD Radeon™ HD 7870 GHz Edition

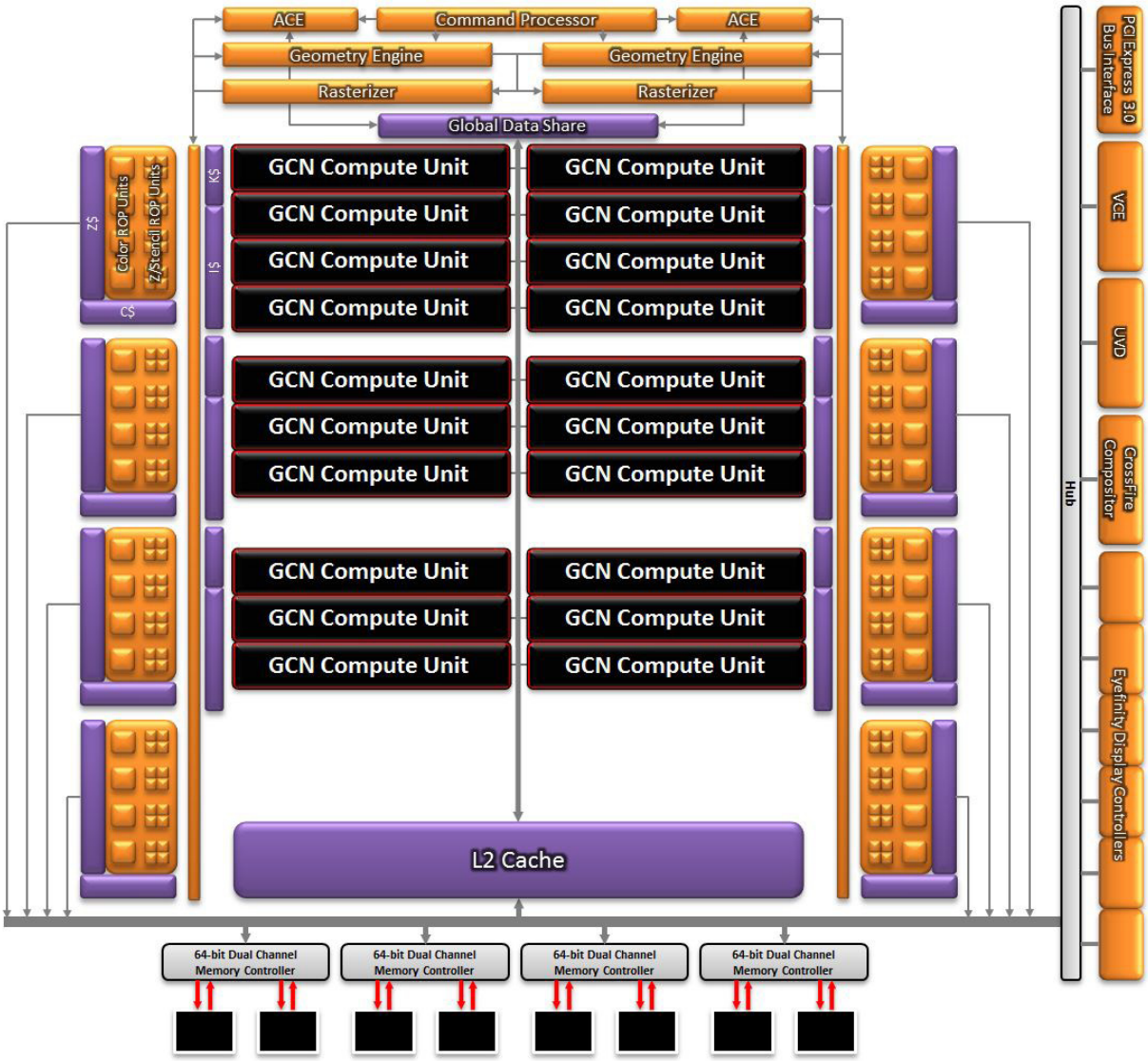
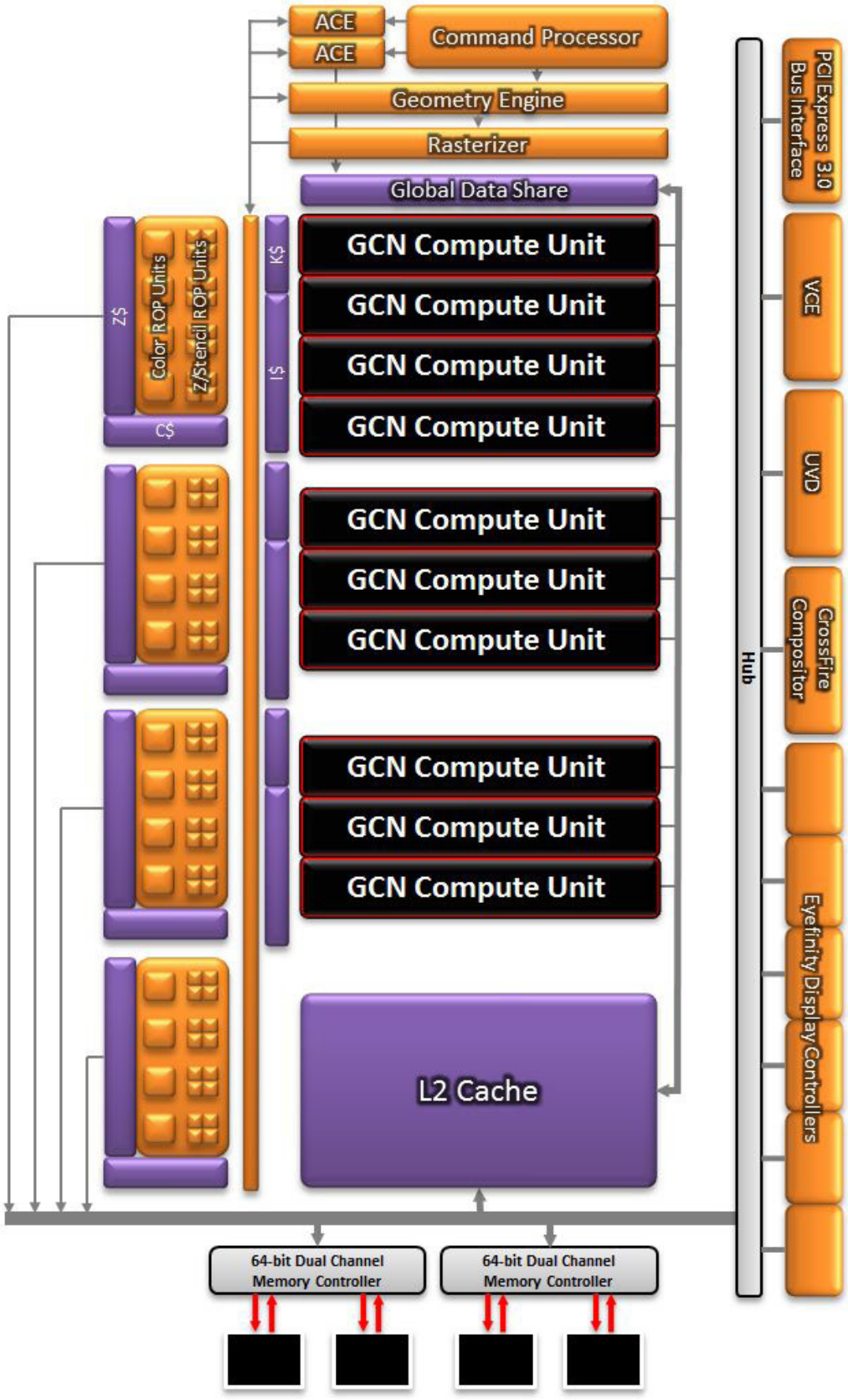


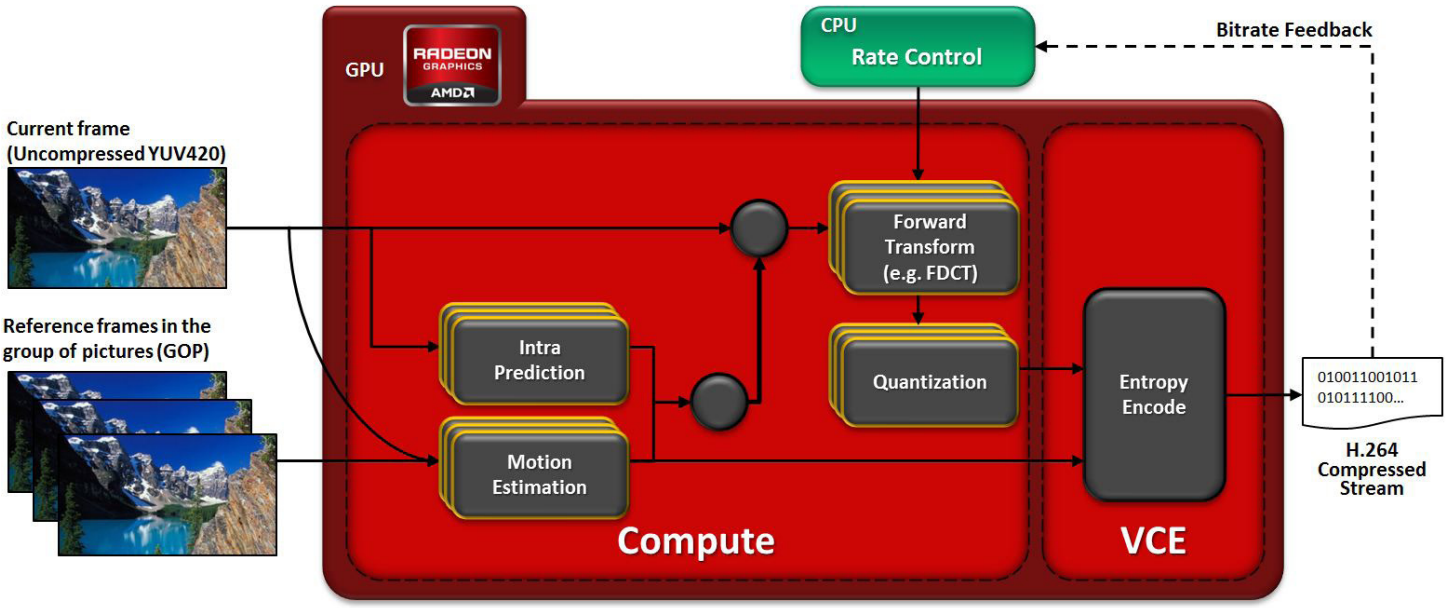
Figure 9: AMD Radeon™ HD 7770 GHz Edition



PROGRAMMABLE VIDEO PROCESSING

General programmability was a top priority across the entire GCN Architecture, especially for video processing. One of the unique characteristics of video is that the data types are very specific and there is tremendous opportunity to take advantage of fixed function hardware. The dedicated UVD3 decoder engine adds support for MPEG-4 and DivX formats, along with Multi-View Codecs that are used in Stereo 3D and HD picture-in-picture displays.

Figure 10: Hybrid Video Encoding



GCN integrates a power-efficient video codec engine (VCE), which contains a full hardware implementation of H.264 encoding capable of 1080p at 60 frames/sec. However, the VCE was explicitly designed with programmability in mind - the entropy encoding block of VCE is fully accessible to software. Developers using AMD's Accelerated Parallel Programming SDK and OpenCL™ can create hybrid encoders that pair custom motion estimation, inverse discrete cosine transform and motion compensation with the hardware entropy encoding to achieve faster than real-time encoding.

RELIABLE COMPUTING

General purpose computing requires a level of reliability that is far greater than the traditional graphics world, but is eminently familiar to AMD. Since 2003, AMD's Opteron processors have been used in some of the world's largest supercomputers and mission critical systems, where downtime and data corruption is unacceptable. AMD has taken this experience with highly reliable design and applied it to the GCN Architecture to create products which are suitable for any system - whether in the server room, a supercomputer, a single workstation or a gaming notebook. The overarching goal for the GCN Architecture is reliability where it is desired, but avoiding overhead for situations where it is unwarranted.

The largest reliability concern for modern chips is soft errors (or bit-flips) in on-chip memory. A high-end GPU like the AMD Radeon™ HD 7970 has over 12MB of SRAM and register files spread throughout the CUs and caches. GCN is designed to support full single error correct and double error detect (SECDED) coverage for all on-chip memory. This comes at a small cost in terms of area and power overhead, but can be invaluable for professional applications where the accuracy and reliability of the results are essential.

The second reliability challenge is the external memory. GDDR5 is based on an industry standard and manufactured by a variety of third parties. The standard GDDR5 memory interface checks transmitted data using CRCs and can resend any corrupted data. However, there is no ECC and thus no way of knowing whether the data that is held in the DRAM has been corrupted by a soft error.

The GCN memory controller has an optional mode that applies SECDED to the DRAM. ECC protected data is about 6% larger, which slightly reduces the overall memory capacity and bandwidth.

As an architecture, GCN is customizable for both professional and consumer products. Professional applications can take advantage of ECC protection for on-chip SRAM and optional external DRAM coverage to provide the highest levels of reliability with minimal performance degradation. GPU products that support ECC will allow it to be enabled or disabled via the graphics driver.

CONCLUSION

AMD's GCN Architecture comes at a time of change for the industry. Graphics is an increasingly important part of the user experience, and a crucial component for SoCs that integrate CPUs and GPU side-by-side. The mandate for GPUs is expanding to include not just 3D rendering, but new general purpose, heterogeneous applications such as facial recognition, which are only feasible using the parallel performance of the GPU.

As a company, AMD is uniquely positioned with deep expertise and a long history of excellence in both CPUs and GPUs for the PC. GCN is a marriage of these domains, melding the reliability and programmability of traditional CPUs with the efficient parallel performance of a GPU.

GCN is huge step forward, firmly placing AMD in the new era of heterogeneous computing, but without losing sight of efficiency or performance. The GCN Architecture encompasses innovations such as virtual memory, coherent caching and an elegant hybrid vector/scalar instruction set that are revolutionary. At the system level, GCN is the only discrete graphics architecture that is compatible with the x86 programming model, paving the way for future software and hardware integration.

Most importantly, AMD has carefully crafted an architecture that is a tremendous advance in programmability, but does not sacrifice performance or efficiency. Features such as a unified instruction stream and scheduling in the scalar pipelines enhance utilization and boost the achievable performance on real workloads. The first GPUs manufactured on 28nm were based on GCN and improved performance/watt and performance/mm^2 by roughly 50% over the prior generation. The AMD Radeon™ HD 7970 GHz Edition achieves peak performance of over 1 TFLOPS double precision and 4 TFLOPS single precision, a testament to the underlying architecture. GCN will eventually revolutionize AMD's entire product line, from tablets to supercomputers.

DISCLAIMER

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors. AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE.

IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

SUBSTANTIATION

1 AMD Eyefinity works with games that support non-standard aspect ratios, which is required for panning across multiple displays. To enable more than two displays, additional panels with native DisplayPort™ connectors, and/or DisplayPort™ compliant active adapters to convert your monitor's native input to your cards DisplayPort™ or Mini-DisplayPort™ connector(s), are required. AMD Eyefinity can support up to 6 displays using a single enabled AMD Radeon™ graphics card with Windows Vista or Windows 7 operating systems – the number of displays may vary by board design and you should confirm exact specifications with the applicable manufacturer before purchase. SLS ("Single Large Surface") functionality requires an identical display resolution on all configured displays.

2 AMD HD3D is a technology designed to enable stereoscopic 3D support in games, movies and/or photos. Requires 3D stereo drivers, glasses, and display. Not all features may be supported on all components or systems – check with your component or system manufacturer for specific model capabilities and supported technologies. A list of supported stereoscopic 3D hardware is available at http://www.amd.com/HD3D.

3 The GCN Architecture and its associated features (PCI Express® 3.0, AMD ZeroCore Power technology, DDM Audio, HDMI® (with 4K and 3GHz) and 28nm production) are exclusive to the AMD Radeon™ HD 7900, HD 7800 and HD 7700 Series GPUs.

4 HBR2 bandwidth can support more than six displays with this specific timing, but the AMD Radeon™ HD 7900 Series GPUs support up to a maximum of six independent displays.

5 Driving a resolution of 4096x2304 @ 60Hz requires a monitor that supports DisplayPort™ 1.2 HBR2. This type of monitor will be driven by the GPU as two 2Kx2K monitors (side-by-side) using the DisplayPort™ 1.2 Multi-Stream Transport protocol over one DisplayPort™ cable.

6 Simulated saturation to show the difference between color corrected and uncorrected image on wide gamut panels.