INTRODUCING

RDNA ARCHITECTURE

The all new Radeon™ gaming architecture powering “Navi”

AMD

Table of Contents

Introduction

The world of graphics has fundamentally evolved and shifted over the course of the last three decades towards greater programmability. The earliest graphics systems were implemented purely in software and ran on CPUs, but could not offer the performance needed for more than the most basic visual effects. The first specialized graphics architectures were almost purely fixed-function and could only accelerate a very limited range of specific 2D or 3D computations such as geometry or lighting transformations. The next wave of architectures introduced graphics shaders that gave programmers a taste of flexibility, but with stringent limitations. More recently, graphics processors evolved towards programmability – offering programmable graphics shaders and fully general-purpose computing.

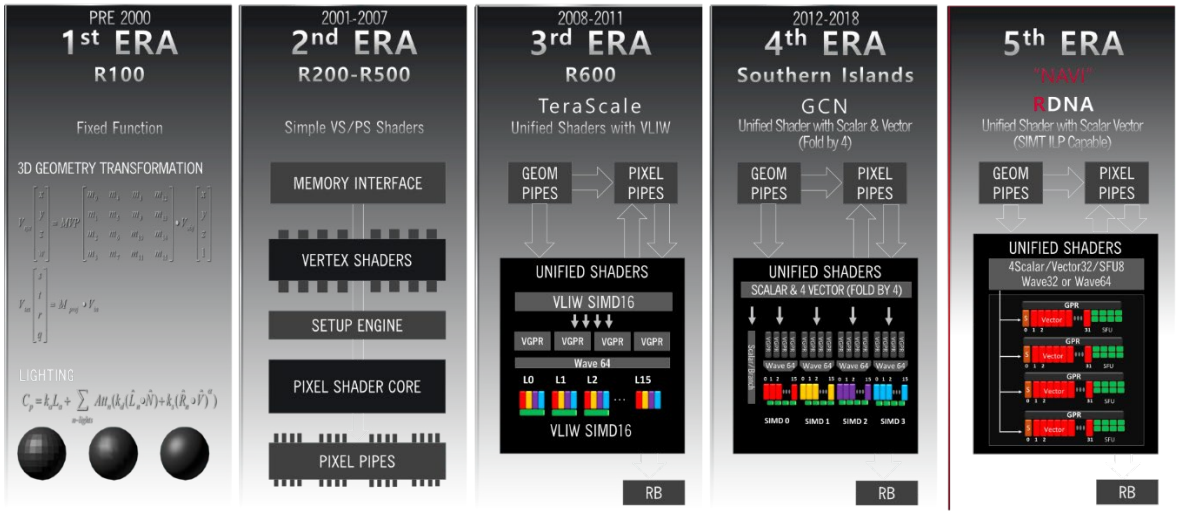

Figure 1 – Eras of Graphics.

AMD's TeraScale was designed for the era of programmable graphics and ushered in general-purpose compute with DirectX® 11's DirectCompute API and a VLIW-based architecture. The Graphics Core Next (GCN) architecture moved to a more programmable interleaved vector compute model and introduced asynchronous computing, enabling traditional graphics and general-compute to work together efficiently. The GCN architecture is at the heart of over 400 million systems, spanning from notebook PCs, to extreme gaming desktops, cutting-edge game consoles, and cloud gaming services that can reach any consumer on a network.

Looking to the future, the challenge for the next era of graphics is to shift away from the conventional graphics pipeline and its limitations to a compute-first world where the only limit on visual effects is the imagination of developers. To meet the challenges of modern graphics, AMD's RDNA architecture is a scalar architecture, designed from the ground up for efficient and flexible computing, that can scale across a variety of gaming platforms. The 7nm "Navi" family of GPUs is the first instantiation of the RDNA architecture and includes the Radeon RX 5700 series.

RDNA Architecture Overview and Philosophy

The new RDNA architecture is optimized for efficiency and programmability while offering backwards compatibility with the GCN architecture. It still uses the same seven basic instruction types: scalar compute, scalar memory, vector compute, vector memory, branches, export, and messages. However, the new architecture fundamentally reorganizes the data flow within the processor, boosting performance and improving efficiency.

In all AMD graphics architectures, a kernel is a single stream of instructions that operate on a large number of data parallel work-items. The work-items are organized into architecturally visible work-groups that can communicate through an explicit local data share (LDS). The shader compiler further subdivides work-groups into microarchitectural wavefronts that are scheduled and executed in parallel on a given hardware implementation.

For the GCN architecture, the shader compiler creates wavefronts that contain 64 work-items. When every work-item in a wavefront is executing the same instruction, this organization is highly efficient. Each GCN compute unit (CU) includes four SIMD units that consist of 16 ALUs; each SIMD executes a full wavefront instruction over four clock cycles. The main challenge then becomes maintaining enough active wavefronts to saturate the four SIMD units in a CU.

The RDNA architecture is natively designed for a new narrower wavefront with 32 work-items, intuitively called wave32, that is optimized for efficient compute. Wave32 offers several critical advantages for compute and complements the existing graphics-focused wave64 mode.

One of the defining features of modern compute workloads is complex control flow: loops, function calls, and other branches are essential for more sophisticated algorithms. However, when a branch forces portions of a wavefront to diverge and execute different instructions, the overall efficiency suffers since each instruction will execute a partial wavefront and disable the other portions. The new narrower wave32 mode improves efficiency for more complex compute workloads by reducing the cost of control flow and divergence.

Second, a narrower wavefront completes faster and uses fewer resources for accessing data. Each wavefront requires control logic, registers, and cache while active. As one example, the new wave32 mode uses half the number of registers. Since the wavefront will complete quicker, the registers free up faster, enabling more active wavefronts. Ultimately, wave32 enables delivering throughput and hiding latency much more efficient.

Third, splitting a workload into smaller wave32 dataflows increases the total number of wavefronts. This subdivision of work items boosts parallelism and allows the GPU to use more cores to execute a given workload, improving both performance and efficiency.

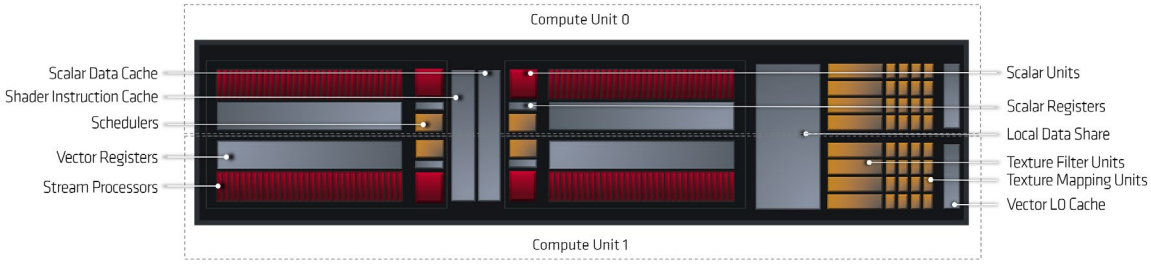

Figure 2 – Adjacent Compute Unit Cooperation in RDNA architecture.

The new dual compute unit design is the essence of the RDNA architecture and replaces the GCN compute unit as the fundamental computational building block of the GPU. As Figure 2 illustrates, the dual compute unit still comprises four SIMDs that operate independently. However, this dual compute unit was specifically designed for wave32 mode; the RDNA SIMDs include 32 ALUs, twice as wide as the vector ALUS in the prior generation, boosting performance by executing a wavefront up to 2X faster. The new SIMDs were built for mixed-precision operation and efficiently compute with a variety of datatypes to enable scientific computing and machine learning. Figure 3 below illustrates how the new dual compute unit exploits instruction-level parallelism within a simple example shader to execute on a SIMD with half the latency in wave32 mode and a 44% reduction in latency for wave64 mode compared to the previous generation GCN SIMD.[1]

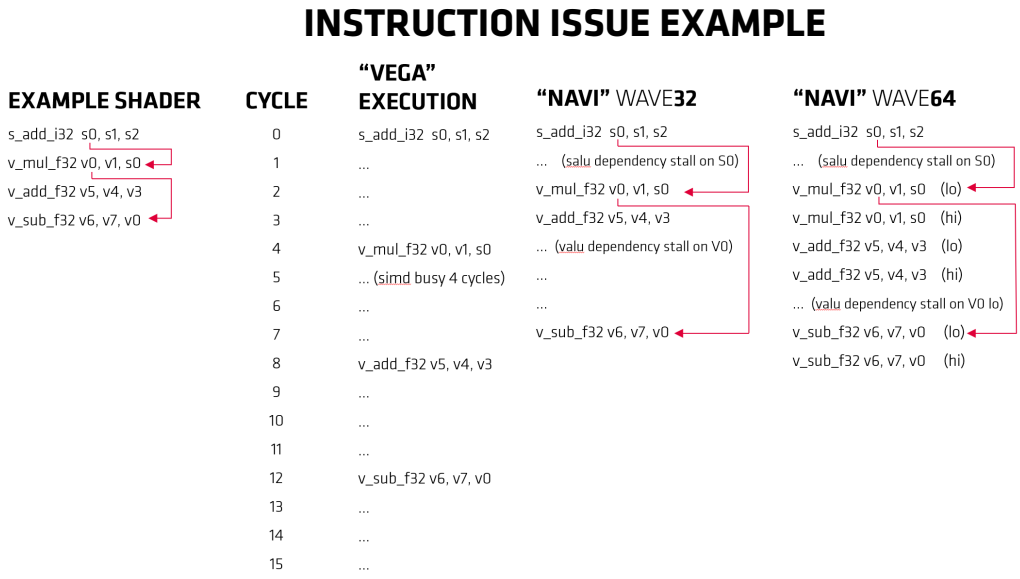

Figure 3 – Wave32 and wave64 execution on an example snippet of code.

The RDNA architecture also redefines the cache and memory hierarchy to increase bandwidth for graphics and compute, reduce power consumption, and enable greater scaling for the future. Earlier architectures employed a two-level cache hierarchy. Generally, the first level of caching was private to each GCN compute unit and focused on compute. The second level of caching was the globally shared L2 that resided alongside the memory controllers and would deliver data both to compute units and graphics functions such as the geometry engines and pixel pipelines.

To satisfy the more powerful dual compute units, the L0 caches for scalar and vector data have scaled up as well. The new architecture introduces a specialized intermediate level of cache hierarchy, a shared graphics L1 cache that serves a group of dual compute units and pixel pipelines. This arrangement reduces the pressure on the globally shared L2 cache, which is still closely associated with the memory controllers.

RDNA System Architecture

Graphics processors (GPUs) built on the RDNA architecture will span from power-efficient notebooks and smartphones to some of the world's largest supercomputers. To accommodate so many different scenarios, the overall system architecture is designed for extreme scalability while boosting performance over the previous generations. Figure 4 below illustrates the 7nm Radeon RX 5700 XT, which is one of the first incarnations of the RDNA architecture.

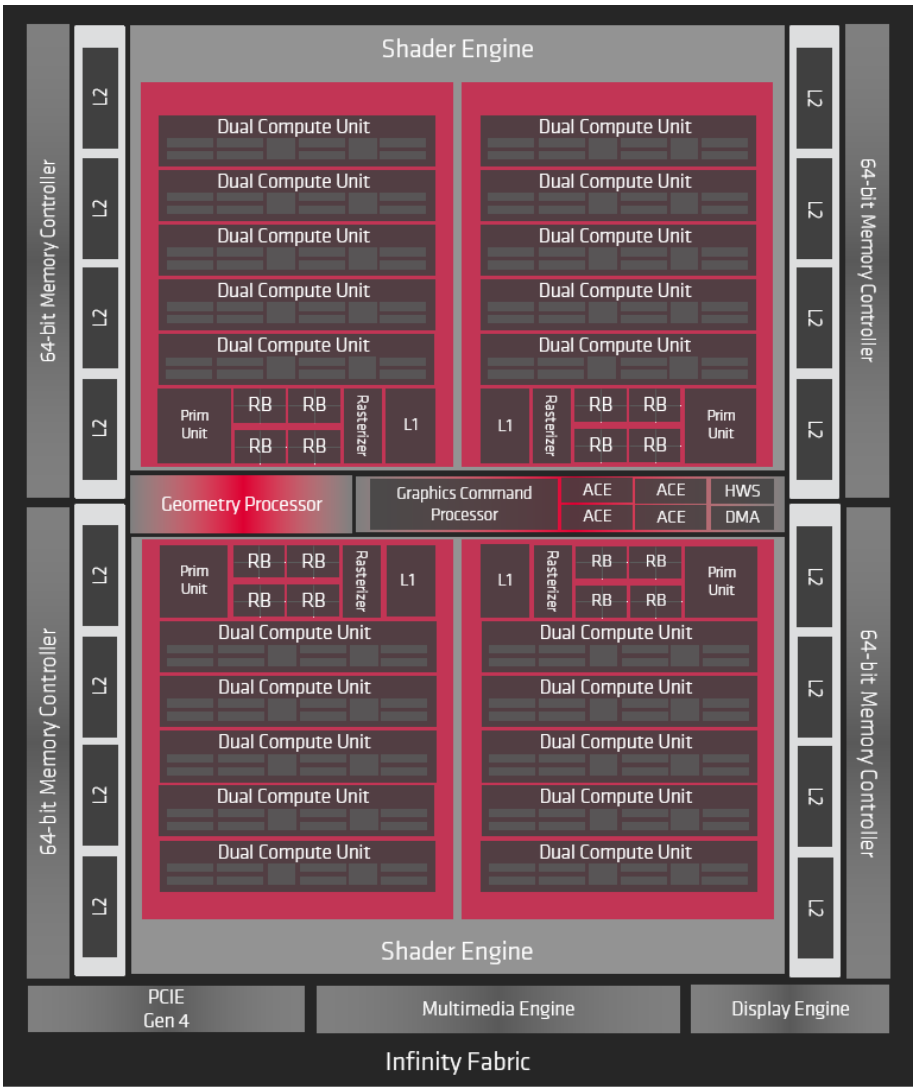

Figure 4 – Block diagram of the Radeon RX 5700 XT, one of the first GPUs powered by the RDNA architecture.

The RX 5700 XT is organized into several main blocks that are all tied together using AMD's Infinity Fabric. The command processor and PCI Express interface connect the GPU to the outside world and control the assorted functions. The two shader engines house all the programmable compute resources and some of the dedicated graphics hardware. Each of the two shader engines include two shader arrays, which comprise of the new dual compute units, a shared graphics L1 cache, a primitive unit, a rasterizer, and four render backends (RBs). In addition, the GPU includes dedicated logic for multimedia and display processing. Access to memory is routed via the partitioned L2 cache and memory controllers.

The RDNA architecture is the first family of GPUs to use PCIe® 4.0 to connect with the host processor. The host processor runs the driver, which sends API commands and communicates data to and from the GPU. The new PCIe® 4.0 interface operates at 16 GT/s, which is double the throughput of earlier 8 GT/s PCI-E 3.0-based GPUs. In an era of immersive 4K or 8K textures, greater link bandwidth saves power and boosts performance.

The hypervisor agent enables the GPU to be virtualized and shared between different operating systems. Most cloud gaming services live in data centers where virtualization is crucial from a security and operational standpoint. While modern consoles focus on gaming, many offer a rich suite of communication and media capabilities and benefit from virtualizing the hardware to deliver performance for all tasks.

The command processor receives API commands and in turn operates different processing pipelines in the GPU. The graphics command processor manages the traditional graphic pipeline (e.g., DirectX®, Vulkan®, OpenGL®) shaders tasks and fixed function hardware. Compute tasks are implemented using the Asynchronous Compute Engines (ACE), which manage compute shaders. Each ACE maintains an independent stream of commands and can dispatch compute shader wavefronts to the compute units. Similarly, the graphics command processor has a stream for each shader type (e.g., vertex and pixel). All the work scheduled by the command processor is spread across the fixed-function units and shader array for maximum performance.

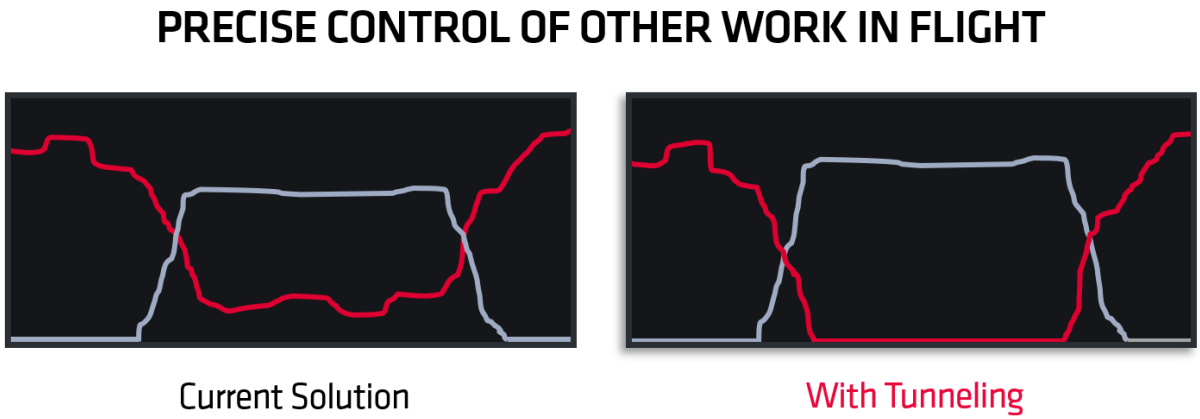

Figure 5 – Asynchronous Compute Tunneling in RDNA architecture.

The RDNA architecture introduces a new scheduling and quality-of-service feature known as Asynchronous Compute Tunneling that enables compute and graphics workloads to co-exist harmoniously on GPUs. In normal operation, many different types of shaders will execute on the RDNA compute unit and make forward progress. However, at times one task can become far more latency sensitive than other work. In prior generations, the command processor could prioritize compute shaders and reduce the resources available for graphics shaders. As Figure 5 illustrates, the RDNA architecture can completely suspend execution of shaders, freeing up all compute units for a high-priority task. This scheduling capability is crucial to ensure seamless experiences with the most latency sensitive applications such as realistic audio and virtual reality.

RDNA Shader Array and Graphics Functions

The traditional graphics pipeline starts with by assembling vertices into triangles; applying vertex shaders; optionally applying hull shaders, tessellation, and domain shaders; rasterizing the triangles into pixels; shading the pixels; and then blending the output. In addition, compute shaders can be used in many different stages for advanced effects.

The command processor and scheduling logic will partition compute and graphics work to enable dispatching it to the arrays for maximum performance. For example, a very common approach for the graphics pipeline is partitioning screen space and then dispatching each partition independently. Developers can also create their own scheduling algorithms, which are especially useful for compute-based effects.

For scalability and performance, the RDNA architecture is built from multiple independent arrays that comprise fixed-function hardware and the programmable dual compute units. To scale performance from the low-end to the high-end, different GPUs can increase the number of shader arrays and also alter the balance of resources within each shader array. As Figure 4 illustrates the Radeon RX 5700 XT includes four shader arrays, each one including a primitive unit, a rasterizer, four RBs, five dual compute units, and a graphics L1 cache.

The primitive units assemble triangles from vertices and are also responsible for fixed-function tessellation. Each primitive unit has been enhanced and supports culling up to two primitives per clock, twice as fast as the prior generation. One primitive per clock is output to the rasterizer. The work distribution algorithm in the command processor has also been tuned to distribute vertices and tessellated polygons more evenly between the different shader arrays, boosting throughput for geometry.

The rasterizer in each shader engine performs the mapping from the geometry-centric stages of the graphics pipeline to the pixel-centric stages. Each rasterizer can process one triangle, test for coverage, and emit up to sixteen pixels per clock. As with other fixed-function hardware, the screen is subdivided and each portion is distributed to one rasterizer.

The final fixed-function graphics stage is the RB, which performs depth, stencil, and alpha tests and blends pixels for anti-aliasing. Each of the RBs in the shader array can test, sample, and blend pixels at a rate of four output pixels per clock. One of the major improvements in the RDNA architecture is that the RBs primarily access data through the graphics L1 cache, which reduces the pressure on the L2 cache and saves power by moving less data.

Last, the shader array comprises of several dual compute units and a graphics L1 cache. The dual compute units provide the compute for executing programmable shaders. Each dual compute unit comprises four SIMDs that can execute a wave32, for a total of 256 single-precision FLOPS per cycle and even more for applications that can use mixed precision. The SIMDs also contain powerful dedicated scalar units and higher bandwidth caching. The new graphics L1 cache serves most requests in each shader array, simplifying the design of the L2 cache and boosting the available bandwidth.

Dual Compute Unit Front End

The more powerful dual compute unit starts with a dedicated front-end as illustrated in Figure 6. The L0 instruction cache is shared between all four SIMDs within a dual compute unit, whereas prior instruction caches were shared between four CUs – or sixteen GCN SIMDs. The instruction cache is 32KB and 4-way set-associative; it is organized into four banks of 128 cache lines that are 64-bytes long. Each of the four SIMDs can request instructions every cycle and the instruction cache can deliver 32B (typically 2-4 instructions) every clock to each of the SIMDs – roughly 4X greater bandwidth than GCN.

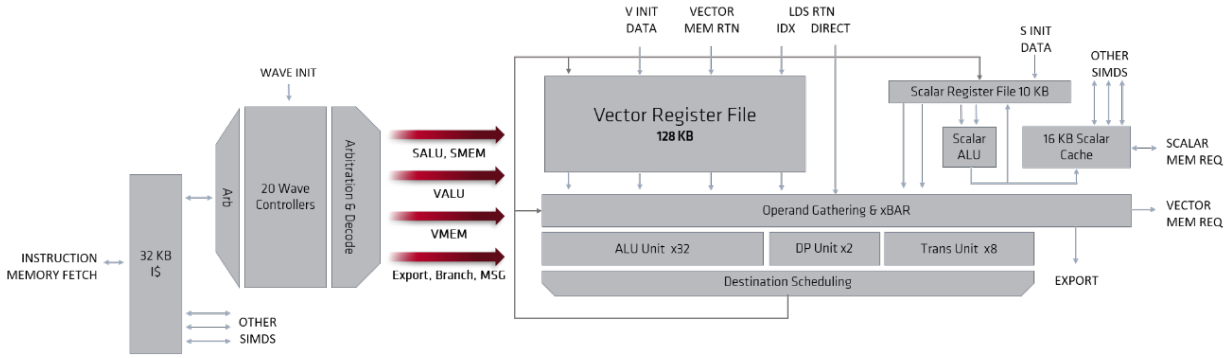

Figure 6 – RDNA compute unit front end and SIMD.

The fetched instructions are deposited into wavefront controllers. Each SIMD has a separate instruction pointer and a 20-entry wavefront controller, for a total of 80 wavefronts per dual compute unit. Wavefronts can be from a different work-group or kernel, although the dual compute unit maintains 32 work-groups simultaneously. The new wavefront controllers can operate in wave32 or wave64 mode.

While the RDNA architecture is optimized for wave32, the existing wave64 mode can be more effective for some applications. To handle wave64 instructions, the wave controller issues and

executes two wave32 instructions, each operating on half of the work-items of the wave64 instruction. The default way to handle a wave64 instruction is simply to issue and execute the upper and lower halves of each instruction back-to-back – conceptually slicing every instruction horizontally.

SHADER PROGRAM	NORMAL EXECUTION SEQUENCE	SUB-VECTOR LOOP EXECUTION SEQUENCE
inst0	inst0 - low	inst0 - low
inst1	inst0 - high	inst1 - low
inst2	inst1 - low	inst2 - low
inst3	inst1 - high	inst3 - low
	inst2 - low	inst0 - high
	inst2 - high	inst1 - high
	inst3 - low	inst2 - high
	inst3 – high	inst3 – high

Figure 7 – Sub-vector mode for wave64 execution executes the low half of all four instructions, followed by the high half.

The RDNA architecture also introduces a software-controlled sub-vector mode that cuts a sequence of instructions in half, and first processes the low half of all the instructions, followed by the upper half. Figure 7 illustrates these two approaches on a simple four instruction loop. The two modes will achieve the same utilization of execution units and complete a full iteration of the loop in eight cycles. However, the sub-vector mode can release half of its registers once the low half of inst3 has executed in cycle 4 and may use the cache more effectively.

Another powerful tool introduced by the RDNA architecture is the clause, which gives software greater control over the execution of groups of instructions. A clause modifies the behavior of compute and memory instructions in the scalar and vector SIMD pipelines. In particular, instructions in a clause are uninterruptable except at software-specified breaks. Clauses enables more intelligent scheduling. For example, a clause of vector load instructions guarantees that all data is read from a fetched cache line before it is evicted – thus maximizing bandwidth and power efficiency. Similarly, forming a clause of compute instructions ensures that all results are available to be written back to a cache or consumed by subsequent instructions.

Once instructions have been fetched into the wavefront buffers, the next step is decoding and issuing instructions within a SIMD. In the prior generation, SIMDs decoded and issued instructions on a round-robin basis, limiting the throughput to once every four cycles. One of the core principles of the RDNA architecture is reducing latency to exploit instruction-level parallelism for each wavefront. Accordingly, each RDNA SIMD can decode and issue instructions every cycle – boosting throughput and reducing latency by 4X.

The dual compute units include a new scheduling mechanism to ensure consistent forward progress. Earlier generations tended to issue instructions from the oldest wavefront and sometimes different wavefronts within a work-group would make uneven progress. The compute unit scheduler has a new oldest work-group scheduling policy that can be activated by software. Using this scheduling algorithm, the wavefronts within a work-group will tend to make more consistent progress and improve locality within caches saving power by more efficiently moving data.

SIMD Execution Units

The RDNA front-end can issue four instructions every cycle to every SIMD, which include a combination of vector, scalar, and memory pipeline. The scalar pipelines are typically used for control flow and some address calculation, while the vector pipelines provide the computational throughput for the shaders and are fed by the memory pipelines.

Scalar Execution and Control Flow

The GCN architecture introduced scalar execution units, shared by the four GCN SIMDs to handle control flow and address generation for compute applications. The RDNA dual compute unit takes this to the next level by providing each SIMD with its own dedicated scalar pipelines, boosting performance for general-purpose applications.

Each SIMD contains a 10KB scalar register file, with 128 entries for each of the 20 wavefronts. A register is 32-bits wide and can hold packed 16-bit data (integer or floating-point) and adjacent register pairs hold 64-bit data. When a wavefront is initiated, the scalar register file can preload up to 32 user registers to pass constants, avoiding explicit load instructions and reducing the launch time for wavefronts.

The branch pipeline is mostly unchanged from the previous generation, although it has been tuned to work with the smaller wavefronts. It handles conditional branches and interrupts.

The scalar ALU accesses the scalar register file and performs basic 64-bit arithmetic operations. The scalar ALU is used for a variety of control purposes, such as calculating values that will be broadcast across an entire wavefront, managing predicates for the wavefront, and

computing branch targets. In addition, the scalar ALU is used for address calculation when reading or writing data in the scalar data cache.

Like the instruction cache, the scalar cache is shared by all the SIMDs in a dual compute unit. The 16KB write-back scalar cache is 4-way associative and built from two banks of 128 cache lines that are 64B. Each bank can read a full cache line, and the cache can deliver 16B per clock to the scalar register file in each SIMD. For graphics shaders, the scalar cache is commonly used to stored constants and work-item independent variables.

Vector Execution

The superb performance and efficiency of modern graphics processors is derived from the parallel computing capabilities of vector execution units. As Figure 8 illustrates, one of the biggest improvements in the compute unit is doubling the size of the SIMDs and enabling back-to-back execution. When using the more efficient wave32 wavefronts, the new SIMDs boosts IPC and cuts latency by 4X.

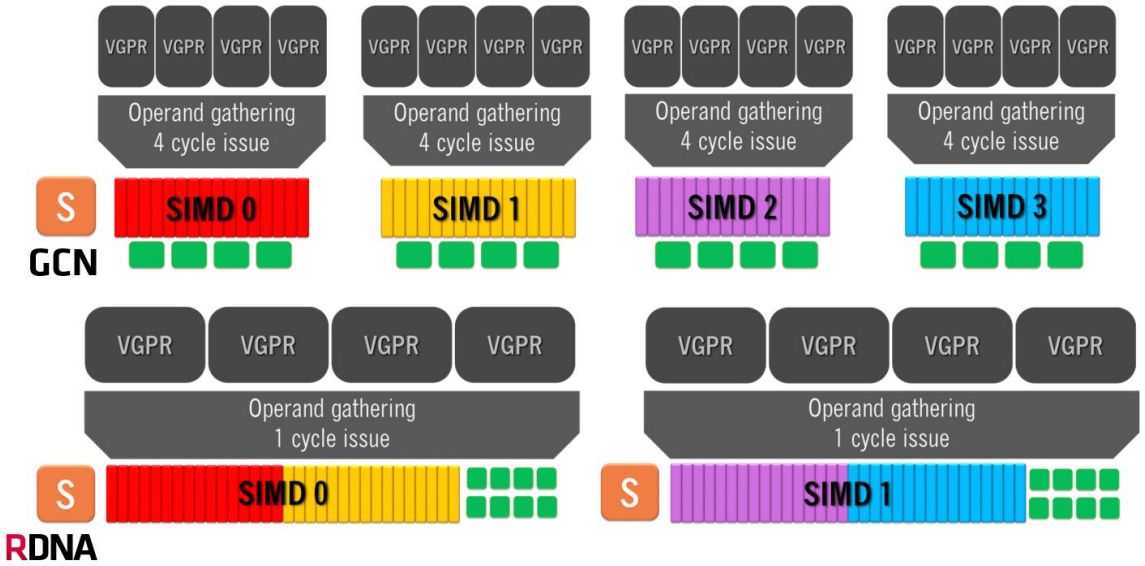

Figure 8 – RDNA SIMD units reduce latency compared to the GCN SIMD units.

Previously, the interleaved vector approach in GCN required scheduling other instructions around four-cycle vector instructions. The larger RDNA dual compute unit also simplifies compiler design and enables more efficient code by scheduling and issuing independent instructions every cycle.

To accommodate the narrower wavefronts, the vector register file has been reorganized. Each vector general purpose register (vGPR) contains 32 lanes that are 32-bits wide, and a SIMD contains a total of 1,024 vGPRs – 4X the number of registers as in GCN. The registers typically hold single-precision (32-bit) floating-point (FP) data, but are also designed for efficiently

handling mixed precision. For larger 64-bit (or double precision) FP data, adjacent registers are combined to hold a full wavefront of data. More importantly, the compute unit vector registers natively support packed data including two half-precision (16-bit) FP values, four 8-bit integers, or eight 4-bit integers.

Vector ALUs

The computational power of the dual compute unit resides in the SIMDs, which have been comprehensively enhanced for greater performance and additional capabilities. The SIMD vector execution pipeline contains several types of execution units. Each SIMD can only issue one wavefront per clock to the vector ALUs, but longer latency operations can overlap execution.

The main vector ALU is a 32-lane pipeline that is twice as wide as the prior generation and enables the SIMD to complete a wavefront every clock cycle, instead of taking four cycles in GCN. The individual lanes have been redesigned for fused multiply-accumulate (FMA) operations, which improves the accuracy of computation and reduces power consumption compared to the previous generation's multiply-add units.

The lanes have also been rearchitected for efficiently executing a wide variety of data types as illustrated in Figure 9. The execution units perform single-precision FMA, add, multiply, or other general FP operations at full rate as well as 24-bit/32-bit integer multiply-accumulates. Packed half-precision FP or 16-bit integer operations run at twice the throughput. The vector ALUs feature a new mixed precision FMA that computes 16-bit multiplication and accumulates into a 32-bit result to avoid losing any precision for integer or FP data. While the vector ALUs primarily read data from the high-bandwidth vGPRs, the scalar register file now supplies up to two broadcast operands per clock to every lane.

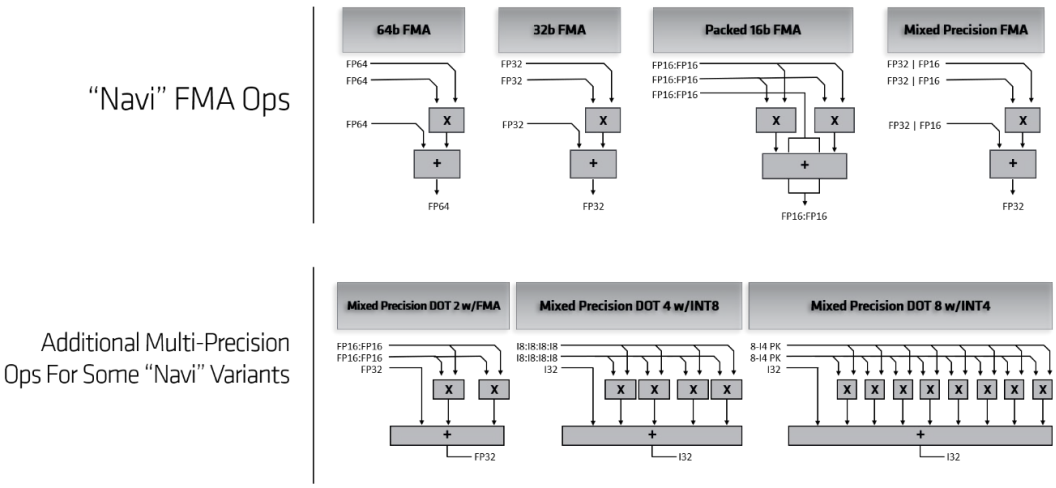

Figure 9 – Mixed-precision FMA and dot-product operations.

Some variants of the dual compute unit expose additional mixed-precision dot-product modes in the ALUs, primarily for accelerating machine learning inference. A mixed-precision FMA dot2 will compute two half-precision multiplications and then add the results to a single-precision accumulator. For even greater throughput, some ALUs will support 8-bit integer dot4 operations and 4-bit dot8 operations, all of which use 32-bit accumulators to avoid any overflows.

The SIMDs use separate execution units for double-precision data. Each implementation includes between two and sixteen double-precision pipelines that can perform FMAs and other FP operations, depending on the target market. As a result, the latency of double-precision wavefronts varies from as little as two cycles up to sixteen. The double-precision execution units operate separately from the main vector ALUs and can overlap execution.

The dual compute unit includes new transcendental execution units to accelerate more complicated math operations that are used in both graphics and general computing. Each SIMD contains an 8-lane transcendental unit that can overlap execution with the main vector ALUs and will complete a wavefront in four clock cycles.

The dual compute unit includes a crossbar within the vector execution pipeline and introduces several new communication instructions. In some previous architectures, cross-lane operations (i.e. between different work-items in a wavefront) were fairly expensive. The new data path in the SIMD is designed for cross-lane data reading, permutations, swizzles, and other operations at full rate.

Dual Compute Unit Memories

Collectively, the four SIMDs in a dual compute unit can sustain an impressive 256 FLOPs every cycle, in addition to scalar operations. The dual compute unit memory hierarchy has evolved to keep up with this level of performance and feed enough data to take advantage of the computational resources. To efficiently provide data, the RDNA architecture includes three levels of caching – L0 caches within a dual compute unit, a shared L1 cache within an array, and a globally shared L2 cache. In addition, the explicitly addressed local data share (LDS) is now part of the overall address space, simplifying programming.

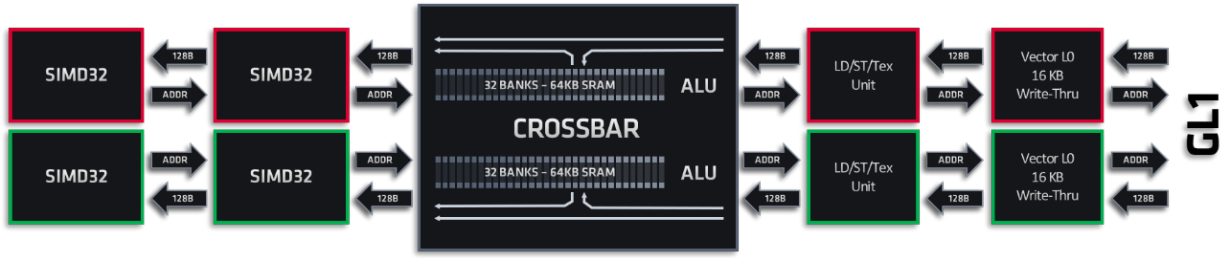

Figure 10 – SIMD memory path for RDNA architecture.

The memory hierarchy starts within the SIMDs as illustrated in Figure 10. The cache and memory pipeline in each SIMD have been redesigned to sustain a full wave32 every clock cycle – twice the throughput compared to the prior generation. Every SIMD has a 32-wide request bus that can transmit an address for each work-item in a wavefront to the memory hierarchy; for store operations, the request bus will instead provide 32x4B of write data. The requested data is transmitted back via a 32-wide return bus and can be provided directly to the ALUs or the vGPRs. The request and return bus generally work with 128-byte cache lines and fan out to the explicitly addressed LDS, cached memory and the texturing units. For efficiency, pairs of SIMDs share a request and return bus, although an individual SIMD can actually receive two 128-byte cache lines per clock – one from the LDS and the other from the L0 cache.

Local Data Share and Atomics

The first type of memory available to the compute units is the local data share (LDS), which a low-latency and high-bandwidth explicitly addressed memory that is used for synchronization within a workgroup and for some graphics functions associated with texturing.

Each compute unit has access to double the LDS capacity and bandwidth. The new LDS has is built from two 64KB arrays with 32 banks per array. As with the previous generation, each bank includes 512 entries that are 32-bits wide and can sustain a read and write every cycle. Each LDS bank includes an ALU that performs atomic operations including FP min and max. Maintaining the bank size and increasing the number of banks ensures that existing shaders will run smoothly, while doubling throughput and performance.

The LDS includes a crossbar to move data between wavefront lanes and banks and can also broadcast or replicate data. Typically, all accesses from a wavefront will complete in a single cycle; however, the hardware automatically detects conflicts and will serialize multiple accesses to a single bank of the LDS.

The RDNA architecture has two modes of operation for the LDS, compute-unit and workgroup-processor mode, which are controlled by the compiler. The former is designed to match the behavior of the GCN architecture and statically divides the LDS capacity into equal portions between the two pairs of SIMDs. By matching the capacity of the GCN architecture, this mode ensures that existing shaders will run efficiently. However, the work-group processor mode allows using larger allocations of the LDS to boost performance for a single work-group.

Vector Caches

The second type of memory available is the standard virtualized address space that is cached on-chip for high-performance. The RDNA architecture has rebuilt the caching hierarchy from the ground up for lower latency, higher bandwidth, and better efficiency.

The new cache hierarchy starts with the same SIMD request and response buses used by the LDS; since the buses are optimized for 32-wide data flow, the throughput is twice as high as GCN. Moreover, each dual compute unit contains two buses and each bus connects a pair of SIMDs to an L0 vector cache and texture filtering logic, delivering an aggregate bandwidth that is 4X higher than the prior generation.

The address calculation logic gets 32 addresses from the request bus and coalesces together requests that target the same 128-byte cache line. By doubling the number of potential requests, the coalescing is even more effective, improving performance and power efficiency.

Each L0 vector cache is 16KB and 4-way set associative, with 128-byte cache lines to deliver a full wavefront of data every cycle. The L0 cache is write-through and uses an LRU replacement policy. While each L0 is coherent within a work-group, software must ensure coherency between the two L0 caches within a dual compute unit. On a cache hit, the L0 cache will read out a 128-byte cache line and deliver the results back via the response bus. For a fully coalesced access, the 128-byte cache line corresponds to 32-bits per work-item, fully satisfying a wave32.

One new feature of the RDNA architecture is that image loads operate must faster. Image loads that do not perform sampling will bypass the texture interpolation hardware and operate at the same rate as buffer loads, improving throughput by 8X and reducing latency by 35% in the case of an L0 cache hit – even when performing format conversion. This enables compute shading effects, which often rely on image accesses, to deliver higher performance and greater efficiency.

Graphics textures are stored in the L0 vector cache, and accessed similarly to data. However, the results are first passed to the texture mapping unit, which can perform filtering for up to eight texture addresses per clock – again twice the throughput of the prior generation. For each address, the TMU will sample the four nearest neighbors, decompress the data, and perform interpolation. The final texel value is passed back to the SIMD via then response bus. The texture mapping unit has also doubled performance for 64-bit bi-linear filtering (using 16-bit FP for each channel of RGBA).

In previous architectures, AMD introduced delta color compression to reduce bandwidth and save power. The RDNA architecture includes enhanced compression algorithms that will save additional bandwidth. Additionally, the texture mapping units can write compressed color data to the L2 cache and other portions of the memory hierarchy, whereas in earlier architectures, compressed data could only be written back to memory.

Data accesses that miss in the vector L0 cache proceed to the outer levels of the memory hierarchy. Data is read by the SIMD from the L0 cache in program order. However, the overall cache hierarchy is designed for aggressive re-ordering so that data can be gathered from many different sources (e.g., graphics L1, L2 cache, memory) and efficiently combined.

Export and GDS

Export instructions transmit results from programmable shaders that are held in the vector register file to a dedicated buffer that is accessible to the rest of the system. For example, when a pixel shader has finished computing, it will export the color and/or depth of the pixel so that the pixel can be blended before being displayed.

Export instructions can target the global data share (GDS). The GDS is a globally shared explicitly addressed memory that is similar to the LDS and can synchronize all wavefronts as well as fixed function hardware. The GDS also contains ALUs used for global atomic operations. The export unit in each dual compute unit can send up to 4 vGPRs to the primitive units, RBs, or the GDS.

Shared Graphics L1 Cache

While moving to a new process technology like 7nm reduces the area and power for transistors, one of the biggest challenges is that the performance and efficiency of wires tends to stay the same or get worse. Consequently, sending data over long distances becomes increasingly expensive. To address this challenge the RDNA architecture introduces a new graphics L1 cache as shown in Figure 11. The graphics L1 cache is shared across a group of dual compute units and can satisfy many data requests, reducing the data that is transmitted across the chip thereby boosting performance and improving power consumption. In addition, the intermediate graphics L1 cache improves scalability and simplifies the design of the L2 cache.

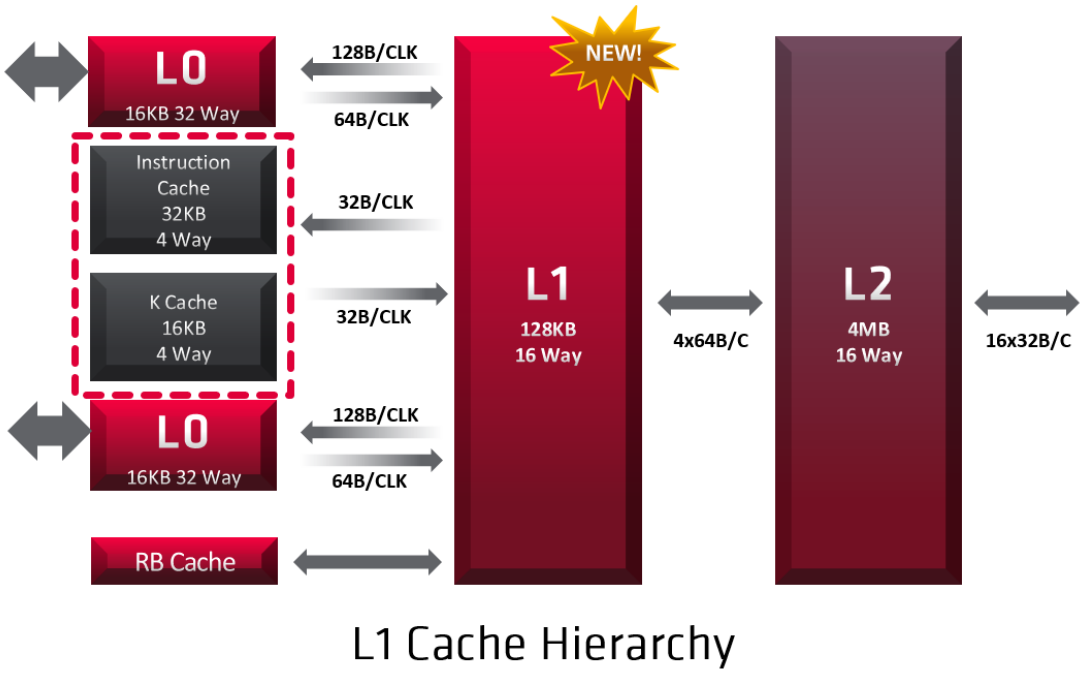

Figure 11 – The new graphics L1 cache in the RDNA architecture absorbs traffic from a group of compute units and an RB.

In the GCN architecture, the globally L2 cache was responsible for servicing all misses from the per-core L1 caches, requests from the geometry engines and pixel back-ends, and any other memory requests. In contrast, the RDNA graphics L1 centralizes all caching functions within each shader array. Accesses from any of the L0 caches (instruction, scalar, or vector data) proceed to the graphics L1. In addition, the graphics L1 also services requests from the associated pixel engines in the shader array.

The graphics L1 cache is a read-only cache that is backed by the globally shared graphics L2; a write to any line in the graphics L1 will invalidate that line and hit in the L2 or memory. There is an explicit bypass control mode so that shaders can avoid putting data in the graphics L1.

Like the L0 vector cache, each line is 128-bytes – consistent with the typical request from a dual compute unit. Each L1 is 128KB with four banks and is 16-way set associative. The L1 cache controller will arbitrate between incoming memory requests and select four to service every clock cycle. Memory accesses that miss in the graphics L1 cache are routed to the L2 cache.

L2 Cache and Memory

One distinct advantage of the RDNA cache hierarchy is that all memory requests to the L2 cache are all routed via the graphics L1 caches. Each shader array comprises 10-20 different agents that request data, but from the perspective of the L2 cache, only the graphics L1 is requesting data. By reducing the number of possible requesters, the on-chip data buses are simpler and easier to route.

The L2 cache is shared across the whole chip and physically partitioned into multiple slices. Four slices of the L2 cache are associated with each 64-bit memory controller to absorb and reduce traffic. The cache is 16-way set-associative and has been enhanced with larger 128-byte cache lines to match the typical wave32 memory request. The slices are flexible and can be configured with 64KB-512KB, depending on the particular product. In the RX 5700 XT, each slice is 256KB and the total capacity is 4MB.

The memory controllers and interfaces for the RDNA architecture are designed to take advantage of GDDR6, the fastest mainstream graphics memory. Each memory controller drives two 32-bit GDDR6 DRAMs with a 16 Gbit/s interface, doubling the available bandwidth of previous generation GDDR5 within roughly the same power budget.

Radeon Multimedia and Display Engines

The “Navi” GPU family also includes extremely efficient specialized processing engines for video decoding, encoding, and display.

In many cases, video encoding and decoding can be performed in software on the RDNA dual compute units for the highest quality. However, the dedicated hardware will always yield the best throughput and power efficiency while also freeing up dual compute units for other tasks. The video engine has been enhanced with support for VP9 decoding, whereas prior generations relied on software implementations.

The video engine can decode H.264 streams at high-throughput: 1080p at 600 frames/sec (fps) and 4K at 150 fps. It can simultaneously encode at about half the speed: 1080p at 360 fps and 4K at 90 fps. 8K decode is available at 24 fps for both HVEC and VP9. For more advanced compression, the video engine delivers high throughput decoding for either 8-bit or 10-bit color: 360 fps for a 1080p stream, 90 fps for a 4K stream, and 24 fps for an 8K stream. The encoder is designed for HVEC, while the decoder can also handle VP9.

		H.264	HEVC	VP9
Encode	**1080p**	360 fps	360 fps 8/10b	X
	4K	90 fps	90 fps 8/10b	
Decode	**1080p**	600 fps	360 fps 8/10b	360 fps 8/10b
	4K	150 fps	90 fps 8/10b	90 fps 8/10b
	8K	X	24 fps 8/10b	24 fps 8/10b

Table 1. Video encoding and decoding in the Radeon Multimedia Engine.

Ultimately, computer graphics is an end-to-end problem. As an application takes advantage of additional features and new compute effects to create more immersive experiences, the display must keep pace and translate the computer-generated visualizations into reality. The RDNA display engine was majorly re-designed and optimized for 4K and 8K displays and high-dynamic range (HDR).

The HMDI® output delivers the full 18 Gbps speed of HDMI® 2.0b to enable 4K at 60 Hz or 1080P at 240 Hz. The DisplayPort™ 1.4a connection is the same raw bandwidth, up to 32.4 Gbps over existing cables, but adds Display Stream Compression (DSC), which enables future support of displays up to 8K HDR at 60 Hz and 4K HDR at 240 Hz.

DSC is an industry-standard, low-latency, and visually lossless compression algorithm that was developed and ratified by VESA®. DSC can compress video without chroma sub-sampling (4:4:4) to as low as 8 bits/pixel for both HDR and SDR. This enables operating a 4K HDR display at 144 Hz and beyond without sacrificing image quality; for example, traditional systems without DSC require chroma sub-sampling (either 4:2:2 or 4:2:0), which can introduce

color artifacts. DSC compresses both HDR and SDR streams to the same bit rate, which allows them to be used interchangeably and avoids any visual quality (e.g., reducing resolution or refresh rate) loss from display cable limitations.

True Audio Next

The True Audio Next toolkit is a great example of applying compute effects to enhance realism and create a more immersive experience. True Audio Next can accurately model 256 channels of audio using physically accurate ray tracing and positional effects to simulate complex environments.

Real-time audio is even more latency and jitter sensitive than graphics and this approach is only possible because of several compute-centric features. First, the GPU can partition the shader arrays and reserve select compute units solely for audio tasks, ensuring low latency and predictable execution. Second, True Audio relies on two ACE-managed real-time task queues. One queue is used for computing audio ray-tracing using the open-source Radeon Rays library, the other is used to calculate convolutional effects for positional audio and full surround sound rendering. Using two separate queues enables using the throughput of the dual compute units for low-latency and deterministic audio effects.

Advanced Visual Effects

Compute is also empowering the next generation of visual effects to push past the limitations of traditional graphics techniques. Many modern games use temporal anti-aliasing to take advantage of information in previously rendered frames; however, sometimes this introduces blurring and other artifacts such as color distortion around edges.

Figure 12 – Contrast Adaptive Sharpening improves image quality across the board

AMD's FidelityFX suite includes a new approach known as Contrast Adaptive Sharpening (CAS) that uses post-processing compute shaders to enhance image quality. CAS enhances details at the interior of an object, while maintaining the smooth gradients created by the anti-aliasing as illustrated in Figure 12. It is a full-screen compute shader and therefore can work with any type of anti-aliasing and is particularly effective when paired with temporal anti-aliasing.

The core idea of CAS is running a 3x3 filter and calculating the difference between the approximate maximum and approximate minimum in the filter to determine how much sharpening is appropriate. CAS is extremely fast, taking just 0.15 milliseconds for a 2560x1440 frame, and benefits from a variety of features in the RDNA architecture such as packed integer math for address calculations, packed fp16 math for compute, faster image loads, and wave32.

Radeon RX 5700 Family

The Radeon RX 5700 series GPUs are the initial members of the "Navi" family and benefit not only from the RDNA architecture, but also a state-of-the-art 7nm process technology that slightly increases frequency. These GPUs replace both the previous generation "Vega" and "Polaris" architectures and offer significant improvements in performance, power efficiency, and area efficiency.

		Radeon RX 5700 XT	Radeon RX Vega 64
Frequency	Ghz	1.905	1.546
Triangles rasterized	Δ/s	7620	6184
Triangle cull rate	Δ/s	15240	6184
Pixel Fill Rate	GPixels/s	121.92	98.94
CUs		40	64
FP32 Performance	TFLOP/s	9.75	12.66
INT8 Texturing	GTexels/s	304.8	395.8
FP16 Texturing	GTexels/s	304.8	197.9
L0 Bandwidth	TB/s	9.76	6.33
L1 Bandwidth	TB/s	3.90	
L2 Bandwidth	TB/s	1.95	1.58
Total Cache Bandwidth	TB/s	15.61	7.91
Total Cache Bandwidth/FLOP		1.6	0.625

Memory Capacity	GB	8GB GDDR6	8GB HBM2
Memory Interface	bit	256	2048
Memory bandwidth	GB/s	448GB/s	484GB/s
Area		251mm^2	495mm^2
TDP		225W	295W

Table 2. Radeon RX 5700 XT compared to Radeon RX Vega 64

The RDNA architecture emphasizes an elegant and balanced approach to performance with a much richer mix of cache and memory bandwidth compared to the brute force compute approach in "Vega". As Table 2 illustrates, the Radeon RX 5700 XT architecture boasts less raw compute in the shader array than the Radeon RX Vega 64. However, the RDNA cache hierarchy provides greater bandwidth to feed the shader array and the superior cache bandwidth enables using the available compute much more effectively. The Radeon RX 5700 XT delivers over 2.5X higher bytes per FLOP, a tremendous improvement that is crucial for the Radeon RX 5700 XT to deliver better performance than the Radeon RX Vega 64, especially on more sophisticated and complex shaders and compute-based rendering systems.

The RDNA architecture also delivers substantial improvements in rasterization, more than doubling the cull rate for triangles that are not visible in a scene, reducing bottlenecks in the geometry portion of the pipeline. Similarly, the texture sampling and interpolation for pixels using FP16 per channel has doubled and is on par with INT8 data, eliminating any performance penalties for using HDR surfaces and boosting visual quality.

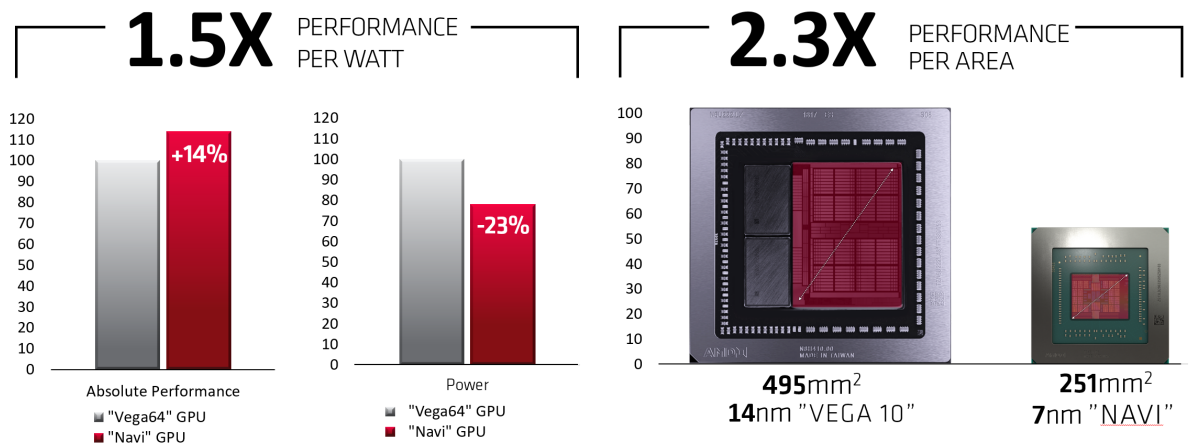

Figure 13 – The Radeon RX 5700 XT delivers significantly better area and power-efficiency than the prior generation. [2,3,4]

As Figure 13 illustrates, the overall impact is that the RDNA-based Radeon RX 5700 delivers substantially better performance all within a die that is half the size compared to the "Vega" generation. The contributing factors to the 1.5X RDNA performance per watt increase are the

7nm process gains, performance per clock enhancements, and design frequency and power improvements.

Conclusion

As graphics has evolved to richer effects and ever greater fidelity and accuracy, the constraints of the traditional graphics pipeline have become increasingly evident. Newer techniques like modeling volumetric fog or tiled lighting use more flexible compute shaders to augment or supplant the traditional graphics pipeline.

To meet these needs, AMD's RDNA architecture is designed for high-performance, scalability, efficiency, and programmability. The new wave32 dataflow improves efficiency for more complex code with branches, reduces latency by completing wavefronts faster, and enables using more hardware to execute a workload. The scalar architecture of the dual compute units simplifies programming, while enabling the hardware to exploit instruction-level parallelism and dramatically reducing latency for wavefronts. The shared graphics L1 cache dramatically increases the bandwidth available to the compute units and also saves power and enhances scalability by reducing the number of requests to the globally shared L2 cache and memory. Last, asynchronous compute tunneling allows seamlessly blending compute and graphics shaders while ensuring the necessary quality-of-service for high-priority tasks.

The 7nm Radeon RX 5700 series is the first implementation of the RDNA architecture and a tremendous step forward for the industry. At the same power, the architectural enhancements boost performance by 50% compared to the 14nm Radeon RX Vega 64. The Radeon RX 5700 series is a high-end product line, bringing the benefits of the RDNA architecture to PC gamers at the cutting edge. Thanks to AMD's wide influence and extensive partnerships, the RDNA architecture will roll out and eventually touch nearly every part of the industry. The RDNA family will ultimately grow to include power-constrained smartphone and tablet processors, gaming consoles, cloud gaming services, and a full spectrum of gaming GPUs from low-cost to the highest performance, bringing the benefits of the RDNA architecture to millions of devices and people across the planet.

Legal Disclaimer and Attributions

The information contained herein is for informational purposes only, and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18

"Vega", "Polaris", and "Navi" are codenames for AMD architectures, and are not product names. GD-122

DirectX® is a registered trademark of Microsoft Corporation in the US and other jurisdictions.

DisplayPort™ and the DisplayPort™ logo are trademarks owned by the Video Electronics Standards Association (VESA®) in the United States and other countries.

HDMI™

The terms HDMI and HDMI High-Definition Multimedia Interface, and the HDMI Logo are trademarks or registered trademarks of HDMI Licensing LLC in the United States and other countries.

PCIe® is a registered trademark of PCI-SIG Corporation.

OpenGL® and the oval logo are trademarks or registered trademarks of Silicon Graphics, Inc. in the United States and/or other countries worldwide.

VESA® is a registered trademark of VESA. All other trademarks, service marks, registered trademarks, and registered service marks are the property of their respective owners.

Vulkan and the Vulkan logo are trademarks of the Khronos Group Inc.

[1] Reduction in issue cycles/latency based on AMD internal analysis, June 8, 2019.

[2] Testing done by AMD performance labs 5/23/19, using the Division 2 @ 25x14 Ultra settings. Performance may vary based on use of latest drivers. RX-325

[3] Testing done by AMD performance labs on June 4 2019. Systems were tested with: Intel(R) Core(TM) i7-5930K CPU @ 3.50GHz (6 core) with 16GB DDR4 @ 2133 MHz using a Asus X99-E Motherboard running Windows 10 Enterprise 64-bit (Ver. 1809, build 17763.053). Using the following graphics cards: Navi (Driver 19.30_1905161434 (CL# 1784070)) with 40 compute units, versus a Vega 64 (Driver 19.4.1) with 40 compute units enabled. Running 3D Mark 11 GT1 (1280 x 720), 3D Mark 11 GT2 (1280 x 720), 3d Mark Firestrike GT1 (2560 x 1440), 3d Mark Firestrike GT2 (2560 x 1440), UnigineHeaven (1920 x 1080) , the Navi (with a die size of 251mm^2) achieved an average FPS score of 140 , 136, 49, 37, and 84 respectively. Compared to the Vega 56 (with a die size of 486mm^2) which achieved 103, 113, 41, 32, and 72 respectively. RX-358

[4] RX-365: Testing done by AMD performance labs on June 4, 2019. Systems were tested with: Intel(R) Core(TM) i7-9900K with 16GB DDR4 @ 2133 MHz running Windows 10. Using the following graphics cards: Radeon RX 5700 XT (Driver 19.30) with 40 compute units, versus a Vega 64 (Driver 19.4.1) with 64 compute units enabled. On average across 1080p, 1440p, and 2160p resolutions, the Radeon RX 5700 XT averages 14% higher. RX-365