

CS4803DGC Design and Programming of Game Console

Spring 2011

Prof. Hyesoon Kim



**Georgia
Tech**



College of
Computing

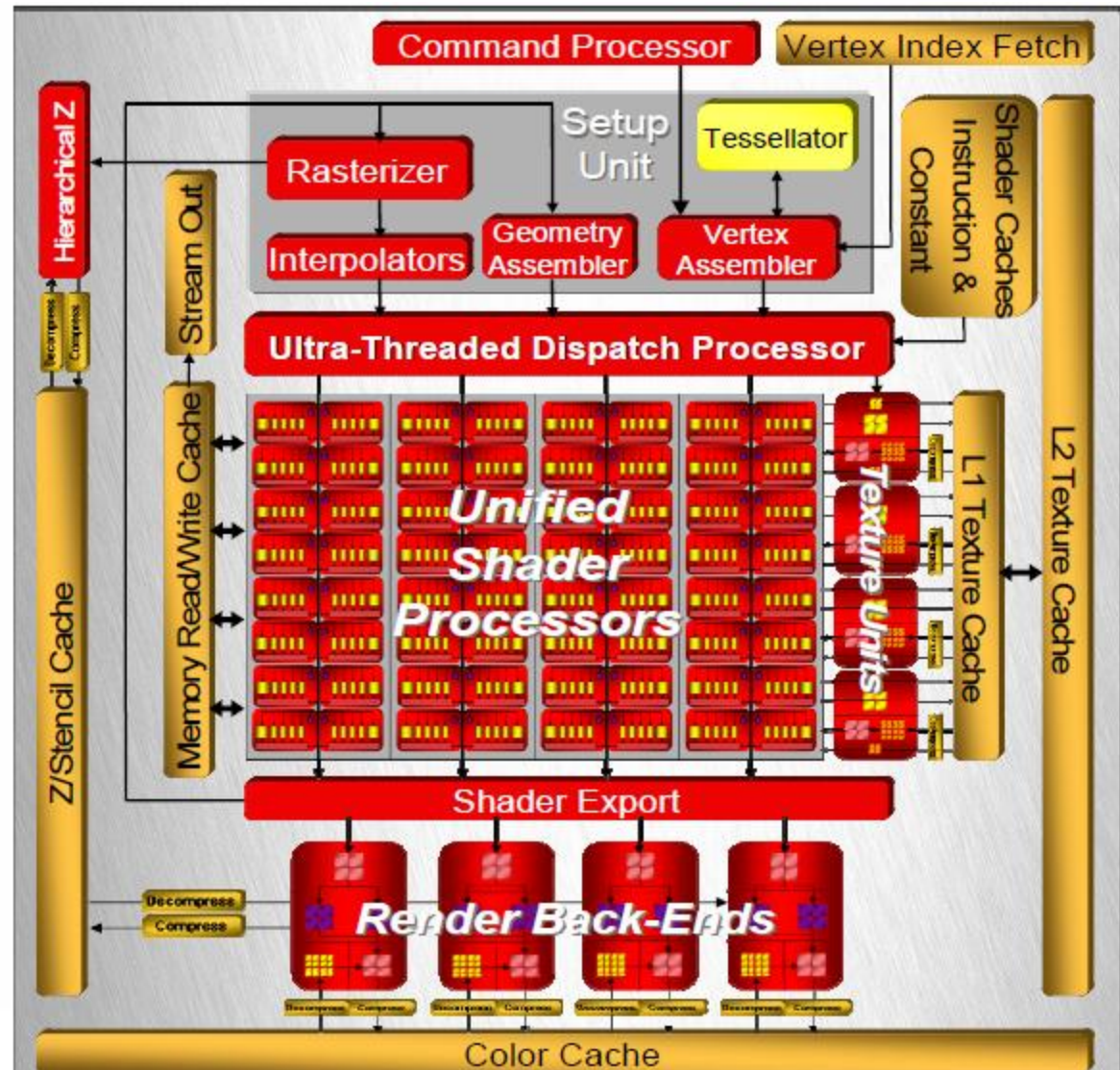


Radeon HD 2000 Series

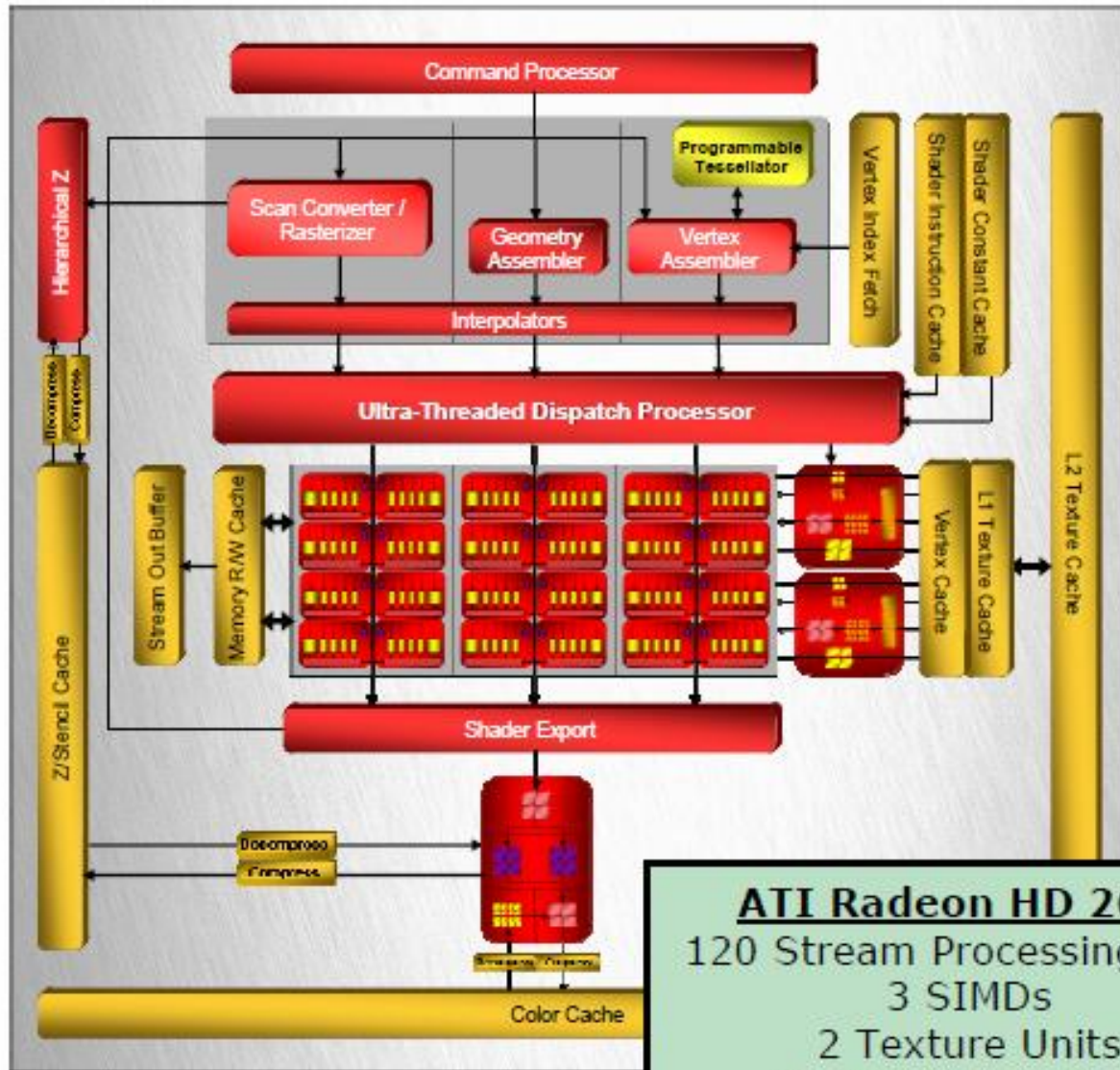
Radeon	2900	2600	2400
Stream Processors	320	120	40
SIMDs	4	3	2
Pipelines	16	8	4
Texture Units	16	8	4
Render Backends	16	4	4
L2 texture cache (KB)	256	128	0
Technology (nm)	80	65	65
Area (mm ²)	420	153	82
Transistors (millions)	720	390	180
Memory bandwidth	512	128	64
Optimized for	High clock speed	Power efficiency	Power efficiency

Radeon 2900 Top Level

- 320 Stream processing units
- 4 SMIDs
- 4 Texture Units
- 4 Render Back-end

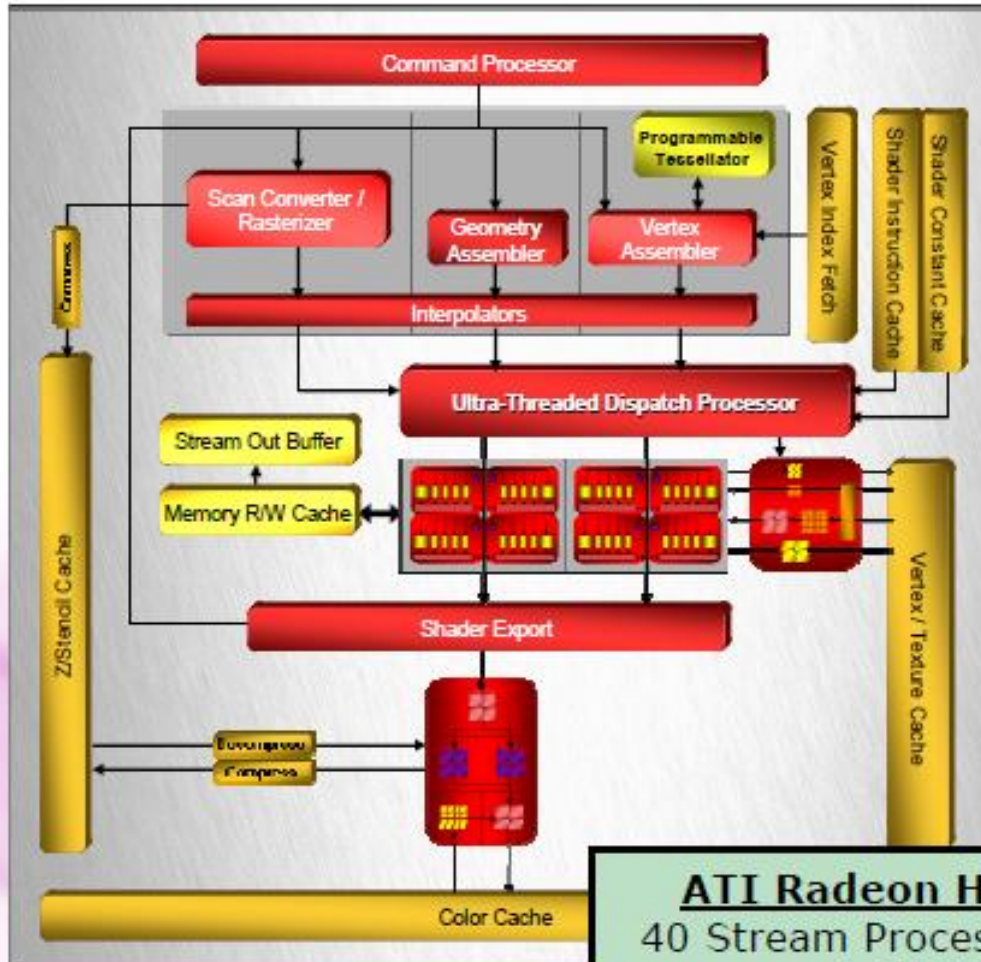


Radeon HD 2600 Top Level



ATI Radeon HD 2600
120 Stream Processing Units
3 SIMDs
2 Texture Units
1 Render Back-End

Radeon HD 2400 Top Level



ATI Radeon HD 2400
40 Stream Processing Units
2 SIMDs
1 Texture Unit
1 Render Back-End
Shared vertex/texture cache

ATI Radeon



	ATI Radeon™ HD 5770	ATI Radeon™ HD 5830	ATI Radeon™ HD 5850	ATI Radeon™ HD 5870
Process	40nm	40nm	40nm	40nm
Transistors	1.04B	2.15B	2.15B	2.15B
Engine Clock	850 MHz	800 MHz	725 MHz	850 MHz
Stream Processors	800	1120	1440	1600
Compute Performance	1.36 TFLOPs	1.79 TFLOPs	2.09 TFLOPs	2.72 TFLOPs
Texture Units	40	56	72	80
Texture Fillrate	34.0 GTexel/s	44.8 GTexel/s	52.2 GTexels/s	68.0 GTexel/s
ROPs	16	16	32	32
Pixel Fillrate	13.6 GPixel/s	12.8 GPixel/s	23.2 Gpixel/s	27.2 GPixel/s
Z/Stencil	54.4 GSamples/s	51.2 GSamples/s	92.8 GSamples/s	108.8 GSamples/s
Memory Type	GDDR5	GDDR5	GDDR5	GDDR5
Memory Clock	1200 MHz	1000 MHz	1000 MHz	1200 MHz
Memory Data Rate	4.8 Gbps	4.0 Gbps	4.0 Gbps	4.8 Gbps
Memory Bandwidth	76.8 GB/s	128.0 GB/s	128.0 GB/s	153.6 GB/s
Maximum Board Power	108W	175W	151W	188W
Idle Board Power	18W	25W	27W	27W

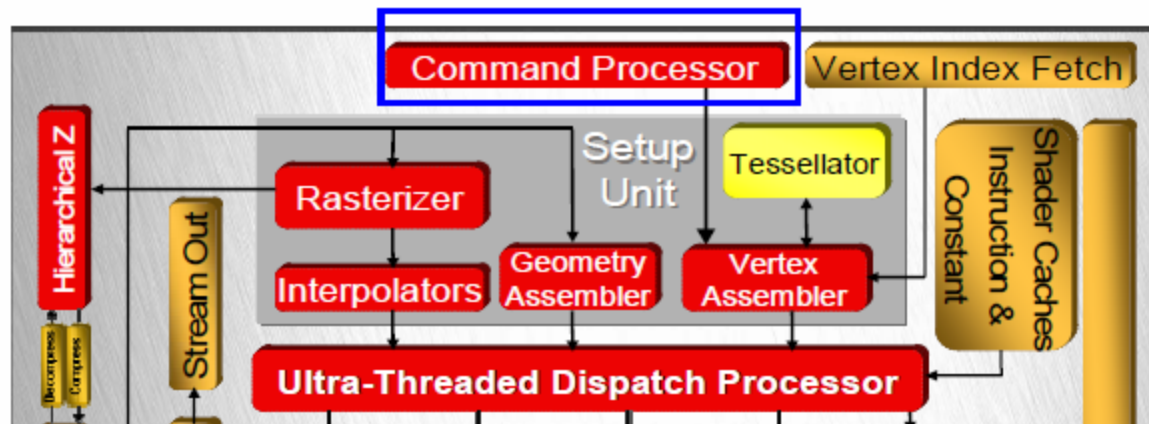


ATI vs. NVIDIA Architecture Differences

- ATI: VLIW
 - More various caches (vertex caches...)
- NVIDIA: SIMD + SFU

Command Processor

- GPU interface with host
 - Processes command stream from graphics driver
- A custom RISC based Micro-Coded engine
- First class memory client with Read/Write access
- State management



Setup engine

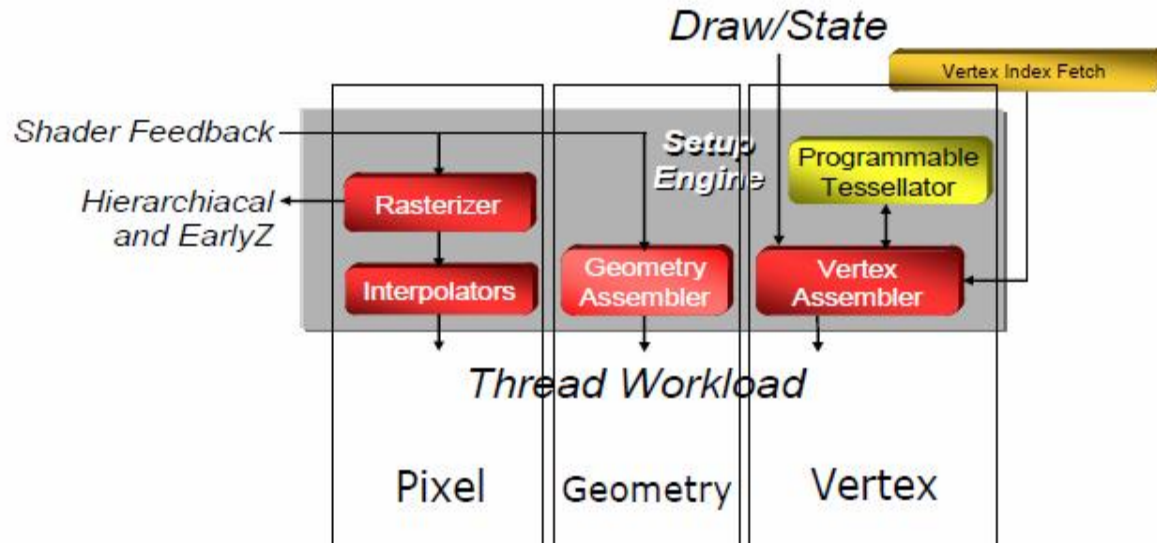
Prepares data for processing by the stream processing units

3 groups of blocks

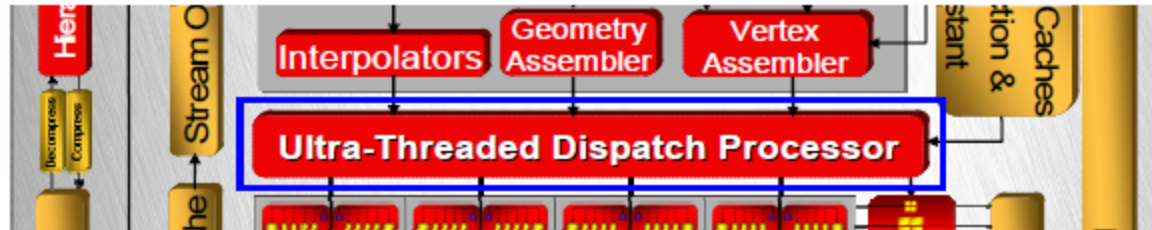
feeding 3 data streams

Each group feeding 16 elements

- **Vertex blocks:** Primitive tessellation, Inputs-index & instancing
 - Sends vertex addresses to shader core
- **Geometry blocks :** Uses on/off chip staging
 - Sends processed vertex addresses, near neighbor addresses and topological information
- **Pixel blocks :** Scan conversion, Triangle setup, Rasterizations, and interpolation
 - Interfaces to depth to perform Hiz/EarlyZ checks



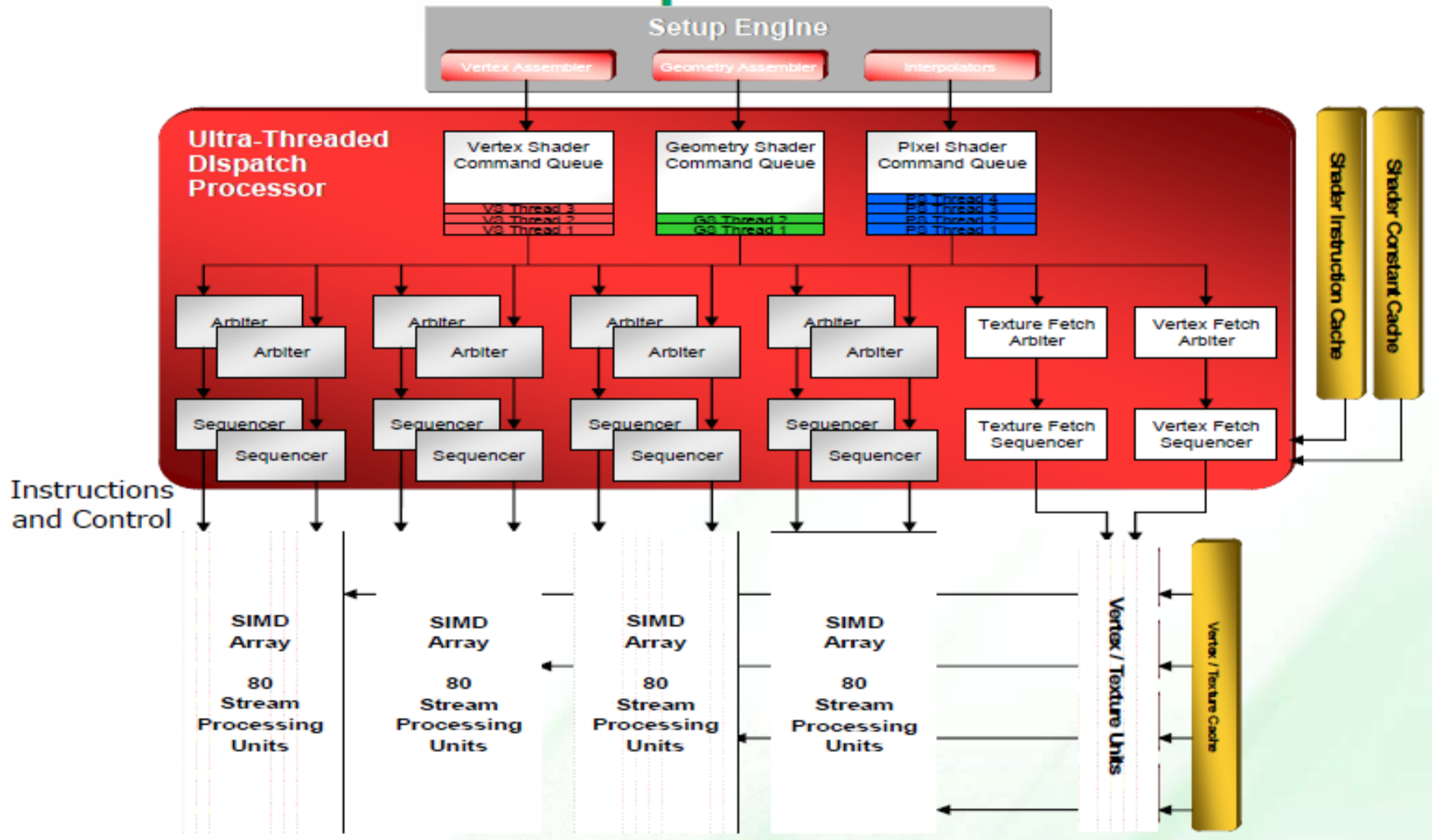
Ultra-Thread Dispatch Processor



- Main control for the shader core
- Separate command queues for each shader type
 - Each thread consists of a number of instructions that will operate on a block of input data
 - All workloads have threads of 64 elements
 - 100's of threads in flight
 - Threads are put to sleep when they request a slow responding resource



Ultra-Threaded Dispatch Processor





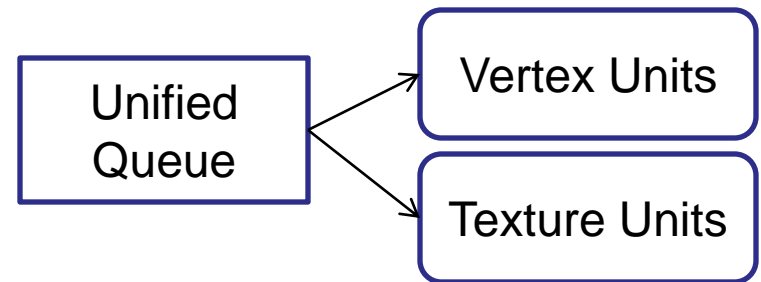
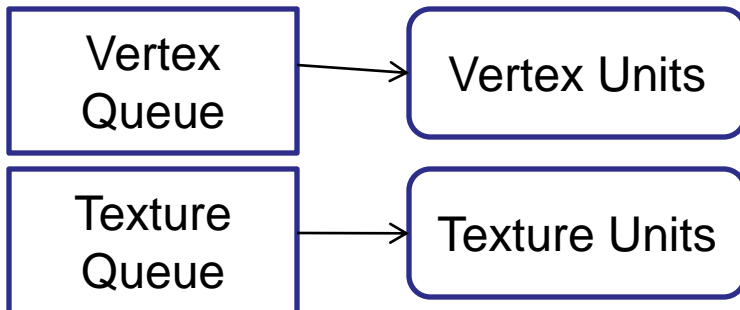
Arbiter in Ultra-threaded Dispatch Processor

- Initial arbiter to select with thread to submit
- Two arbiter units per SIMD array
 - Allows each SIMD to be pipelined, with two operations at a time in process
- Dedicated arbiter units for texture and vertex fetches
 - Can be scheduled independently from math operations
- Executing threads can be bumped at any time if a higher priority thread is pulled from the command queues
 - Temporary data saved so thread can resume later
- Arbitration policy
 - Age/need/availability
 - When in doubt favor pixels
 - Programmable



Q: Thread Dispatch Units

- ATI has dedicated arbiter units for texture and vertex fetches. Among the following three cases, when having dedicated units will provide benefits? There are two design options. Having one unified arbiter with a doubled entry sizes or having two unified arbiters. (2 insts/cycle dispatch model) each queue has 2 entries.
- Case 1: vertex, vertex, texture, texture (vertex latency 1, texture latency 1)
- Case 2: vertex, vertex, vertex, vertex (vertex latency 1, texture latency 1)
- Case 3: texture, texture, texture, texture, vertex (vertex latency 2, texture latency 1)
- Case 4: vertex, texture, vertex, texture, vertex, texture (vertex latency 2, texture latency 1)



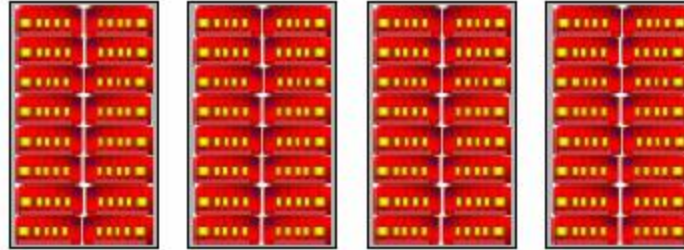


Ultra-threaded Dispatch Processor

- Dedicated shader caches
 - Instruction cache allows unlimited shader length
 - Constant cache allows unlimited number of constants
 - Both caches take advantage of data re-use to improve state change overhead and efficiency
- Latency hiding
 - Cache miss, switches to another thread
 - Suspended threads remain in the command queues until their requested data arrives
 - Ultra-threaded dispatch processor can queue up hundreds of threads



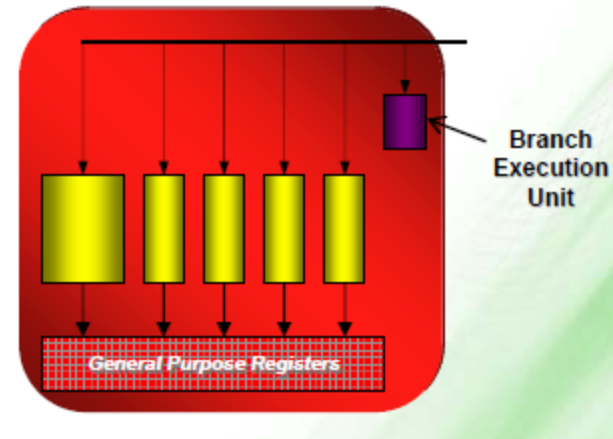
Shader Core



- 4 parallel SIMD units
- Each unit receives independent ALU instruction
- **Very Long Instruction Word (VLIW)**
 - Each instruction word can include up to 6 independent, co-issued operations (5 math + 1 flow control)
 - All operations are performed in parallel on each data element in the current thread
- **Texture fetch and vertex fetch instructions are issued and executed separately**
 - Allows fetches to begin executing before the requested data is required by the shader
- ALU Instruction (1 to 7 64-bit words)
 - 5 scalar ops- 64 bits for src/dst/controls/op
 - 2 additional for literal constants

Stream Processing Units

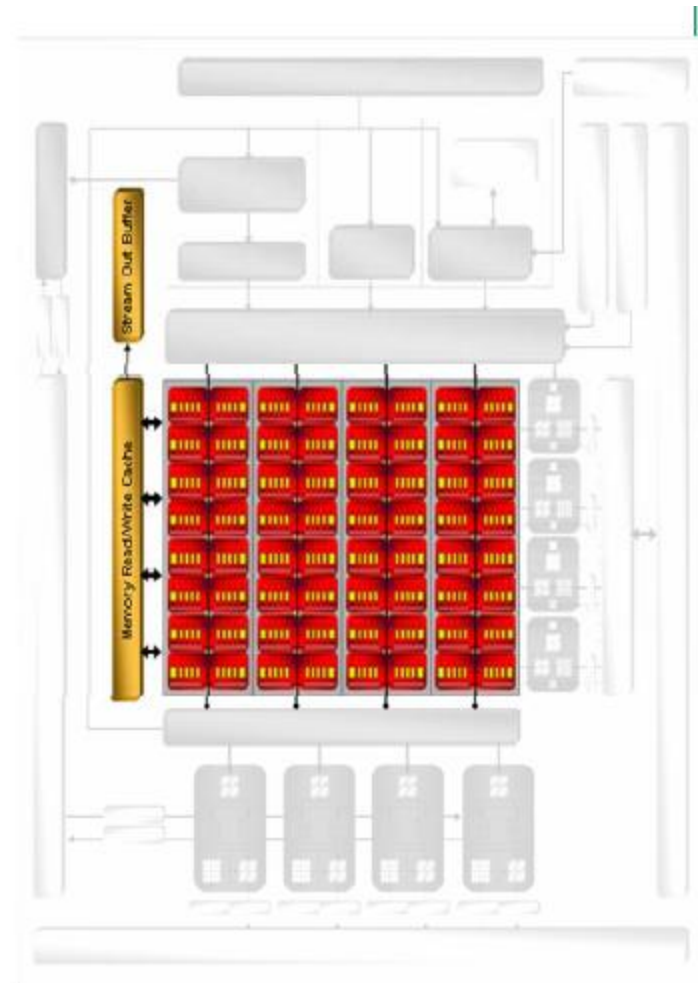
- 5 Scalar Units
 - Each scalar unit does FP Multiply-Add (MAD) and integer operations
 - One also handles transcendental instructions (SIN, COS, LOG, EXP, etc.)
 - IEEE 32-bit floating point precision
 - Integer and bitwise operation support
- Branch Execution unit
- Up to 6 operations co-issued



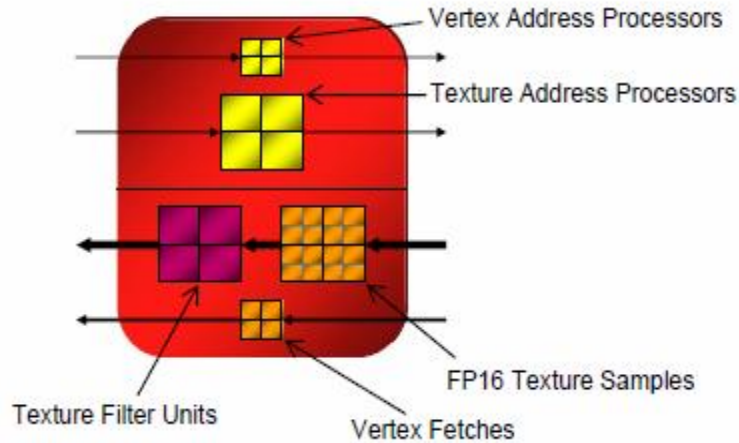


Memory Read/Write Cache

- Virtualizes register space
 - Allow overflow to graphics memory
 - Can be read from or written to by and SIMD (texture & vertex caches are read-only)
 - 8KB Fully associative cache, write combining
- Stream out
 - Allows shader output to bypass render back-ends and color buffer
 - Render to vertex buffer
 - Outputs sequential stream of data instead of bitmaps
- Uses: Used for inter-thread communication



Texture Units

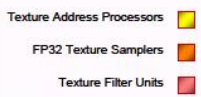
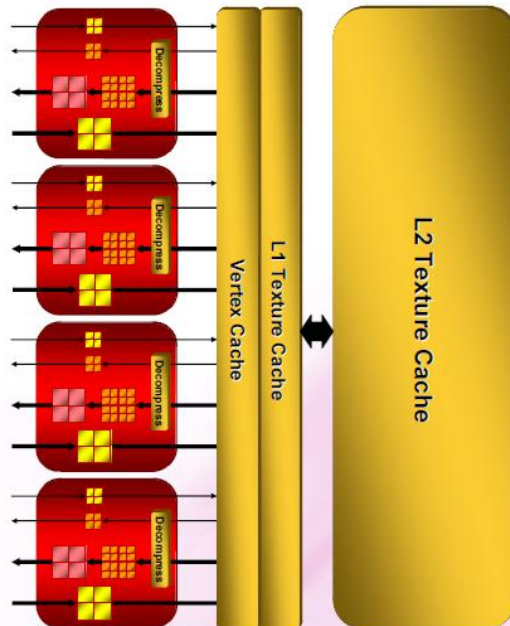


- Fetch Units

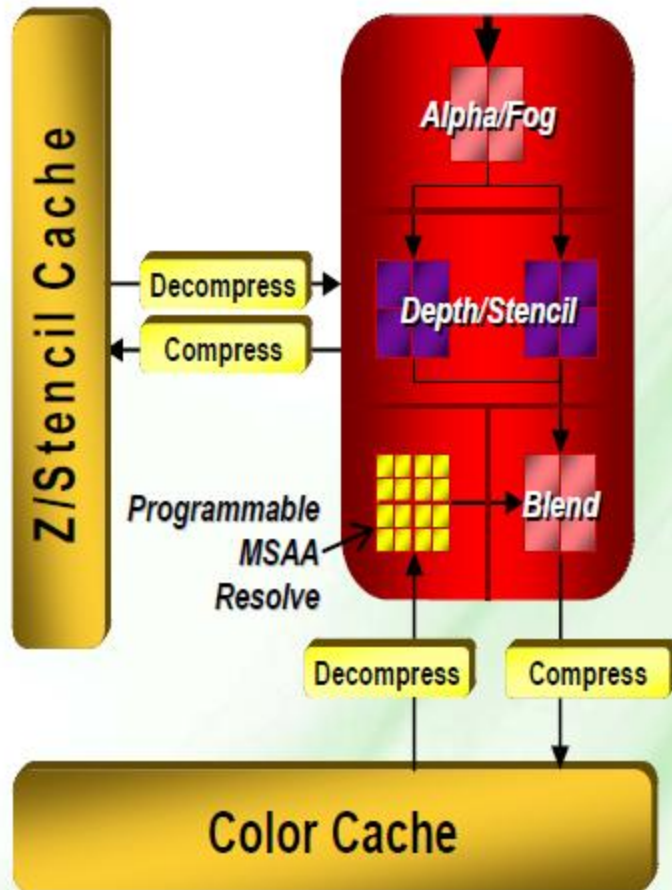
- 8 fetch address processor each (32 total)
 - 4 filtered and unfiltered
- 20 texture samplers each (80 total)
 - Can fetch a single data value per clock
- 4 filtered texels (with BW) (16 total)
 - Bilinear filter one 64-bit FP color value per clocks for each pixel
 - 128-bit FP textures filtered at half speed
 - Trilinear and anisotropic filtering

- Fetch caches

- Unified caches across all SIMDs
- Vertex/Unfiltered cache
 - 4kB L1, 32 Kb L2
- Texture cache
 - 32KB L1, 256 KB L2 (128KB for HD 2600, HD2400 uses single level vertex/texture cache)



Render Back-Ends



- Double rate depth/stencil test
 - 32 pixels per clock for HD 2900
 - 8 pixels per clock for HD2600&HD2400
- Programmable MSAA (multi-sample anti-aliasing) resolve
 - Allows custom AA filters
- New blend-able DX10 surface formats
 - 128-bit and 11:11:10 floating point format
- Up to 8 Multiple Render Targets (MRT) with MSAA support

Depth, Stencil, and Compression Improvements



- Improved Z & Stencil compression
 - Up to 16:1 in standard mode
 - Z & stencil now compressed separately with each other for better efficiency
- Z Range optimization
 - Limit depth test operations to a programmable depth range (useful for speeding up stencil shadowing)
- Re-Z
 - Can check Z buffer twice – once before pixel shader, and again after
 - Allows early Z before shading in all cases
- Improved Hierarchical Z buffer
 - Adds hierarchical stencil (HiS) for better stencil shadow performance
 - Handles most situations where it had to be disabled in the past
- 32-bit floating point z-buffer support



GPGPU on ATI

- ATI GPUs do not support CUDA but it supports OpenCL
- What is OpenCL ?
- Let's visit OpenCL first and then come back to ATI GPGPU