

Xbox 360 System Architecture

Jeff Andrews
Nick Baker
Xbox Semiconductor Technology
Group



Hot Chips Presentation

- Hardware Specs
- Architectural Choices
- Programming Environment
- QA

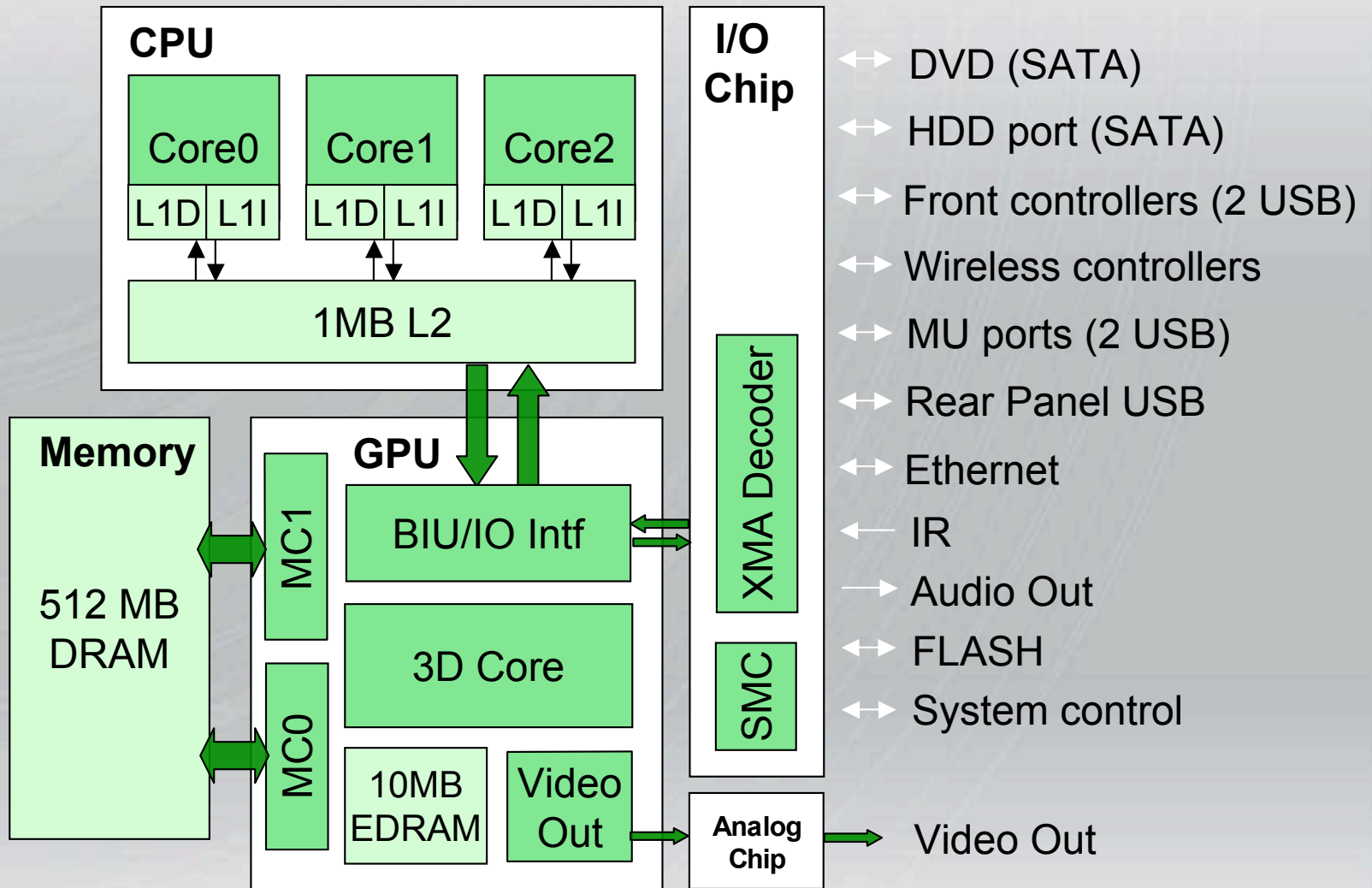
Overview

- Design Principles
 - Next generation gaming
 - Flexibility
 - Programmability
 - Optimized for achievable performance

Hardware Designed for Games

- Triple-core, 3.2 GHz custom CPU
 - Shared 1MB L2 cache
 - Customized vector floating point unit per core
 - 5.4Gbps FSB: 10.8 GB/sec read and 10.8 GB/sec write
 - GPU can read from L2
- 500 MHz custom GPU
 - 48 parallel unified shaders
 - 10 MB embedded DRAM for frame buffer: 256 GB/sec
- 512 MB unified memory
 - 700Mhz GDDR3: 22.4 GB/sec
- 12X dual-layer DVD
- 20 GB hard drive
- High Definition video out

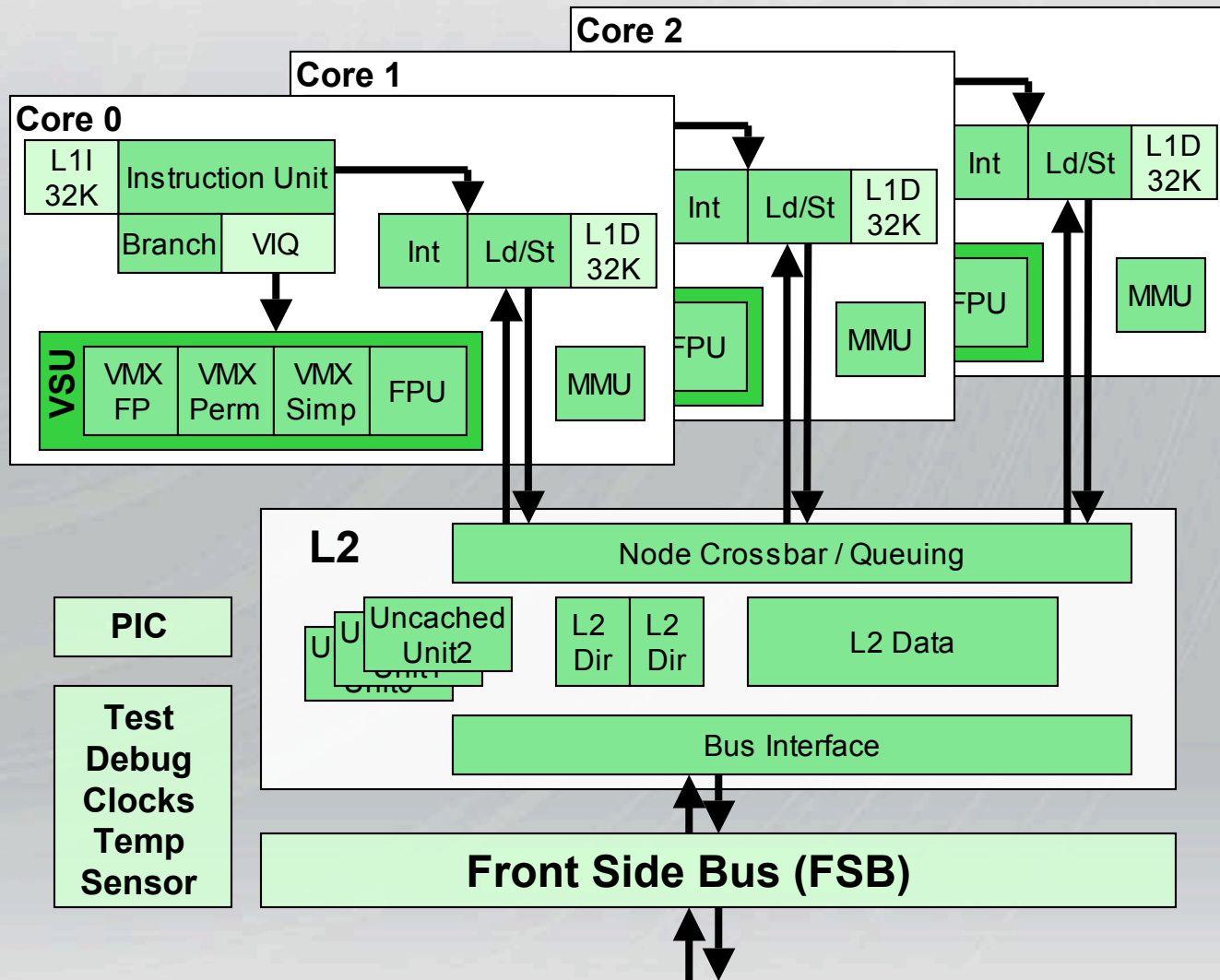
System Block Diagram



CPU Chip/PPC Core Specs

- Three 3.2 GHz PowerPC cores
- Shared 1MB L2 cache, 8-way set associative
- Per-Core Features
 - 2-issue per cycle, in-order, decoupled Vector/Scalar issue queue
 - 2 symmetric fine grain hardware threads
 - L1 Caches: 32K 2-way I\$ / 32K 4-way D\$
 - Execution pipelines
 - Branch Unit, Integer Unit, Load/Store Unit
 - VMX128 Units: Floating Point Unit, Permute Unit, Simple Unit
 - Scalar FPU
- VMX128 enhanced for game and graphics workloads
 - All execution units 4-way SIMD
 - 128 128-bit vector registers *per thread*
 - Custom dot-product instruction
 - Native D3D compressed data formats

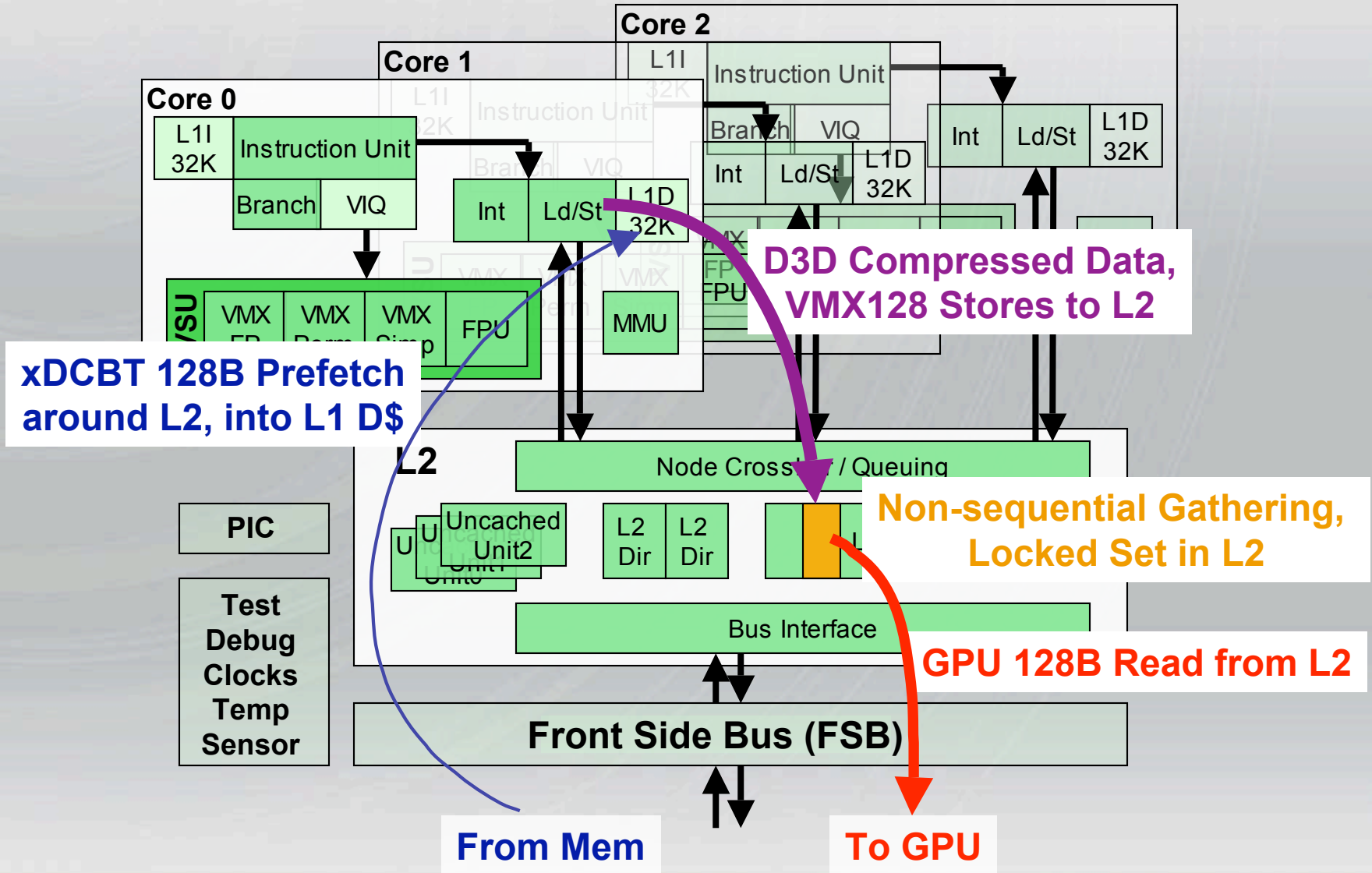
CPU Diagram



CPU Data Streaming Specs

- High bandwidth data streaming support with minimal cache thrashing
 - 128B cache line size (all caches)
 - Flexible set locking in L2
 - Write streaming:
 - L1s are write through, writes do not allocate in L1
 - 4 uncacheable write gathering buffers per core
 - 8 cacheable, non-sequential write gathering buffers per core
 - Read streaming:
 - xDCBT data prefetch around L2, directly into L1
 - 8 outstanding load/prefetches per core
 - Tight GPU data streaming integration (XPS)
 - XPS – “Xbox Procedural Synthesis”
 - GPU 128B read from L2
 - GPU low latency cacheable writebacks to CPU
 - GPU shares D3D compressed data formats with CPU => at least 2x effective bus bandwidth for typical graphics data

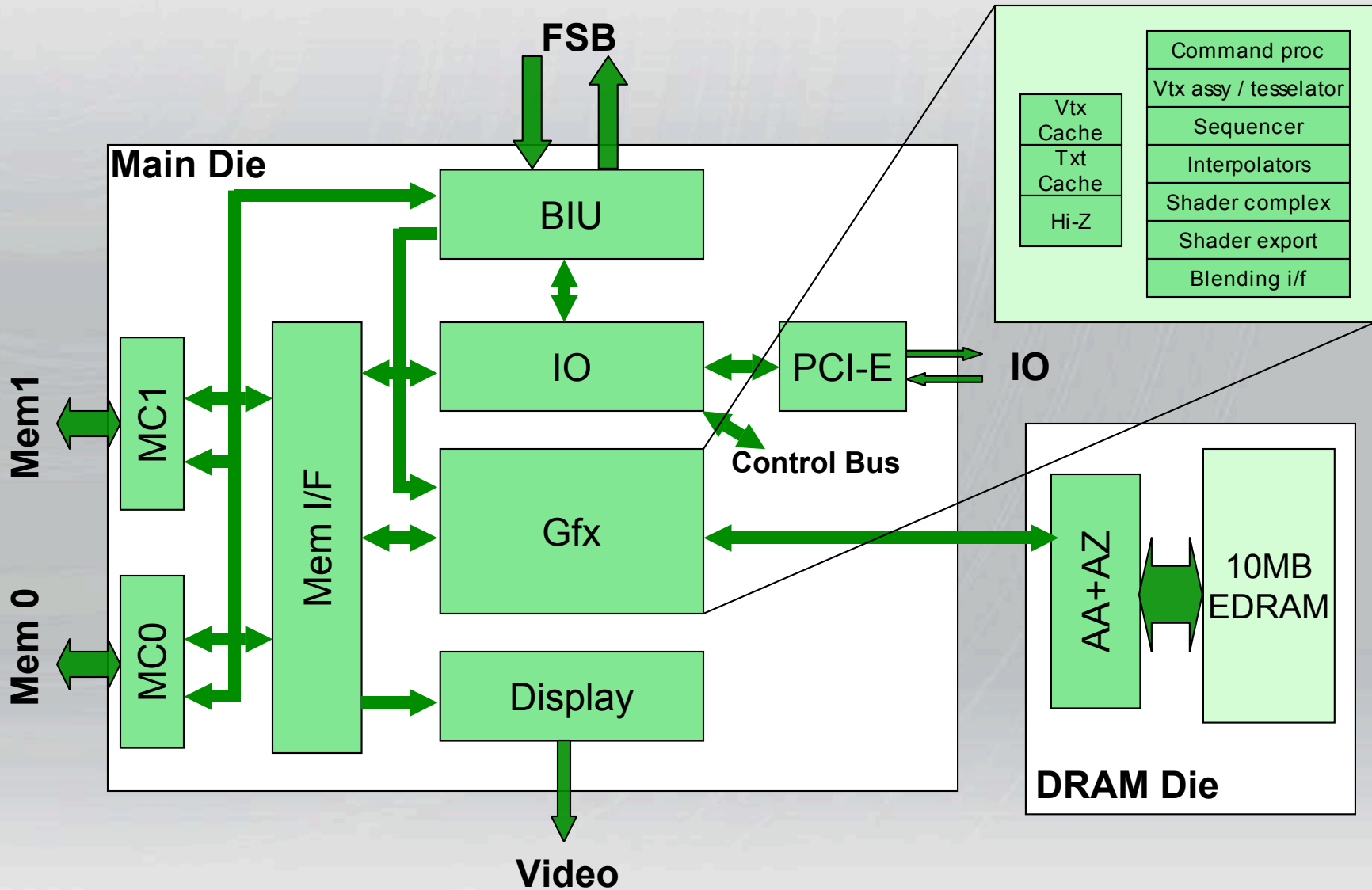
CPU Cached Data Streaming Example



GPU Specs

- 500 MHz graphics processor
 - 48 parallel shader cores (ALUs); dynamically scheduled; 32bit IEEE FLP
 - 24 billion shader instructions per second
 - Superscalar design: vector, scalar and texture ops per instruction
 - Pixel fillrate: 4 billion pixels/sec (8 per cycle); 2x for depth / stencil only
 - AA: 16 billion samples/sec; 2x for depth / stencil only
 - Geometry rate: 500 million triangles/sec
 - Texture rate: 8 billion bilinear filtered samples / sec
- 10 MB EDRAM \Rightarrow 256 GB/s fill
- Direct3D 9.0-compatible
 - High-Level Shader Language (HLSL) 3.0+ support
- Custom features
 - Memory export: Particle physics, Subdivision surfaces
 - Tiling acceleration: Full resolution Hi-Z, Predicated Primitives
 - XPS:
 - CPU cores can be slaved to GPU processing
 - GPU reads geometry data directly from L2

GPU Block Diagram



Architectural Choices - SMP

- Floating point and integer important for games
- Power consumption
- Mainstream parallel technique
- Keep easy to balance
- Solution:
 - Limited SMP using simplified yet powerful cores
 - Tightly coupled to vector floating point

Architectural Choices - EDRAM

- FSAA, alpha and z place heavy load on memory BW
- Post-process effects require large depth complexity
- Enable flexible UMA solution
- Main memory FB/ZB \Rightarrow unpredictable performance
- Many different rendering styles in use, bottlenecks move
- Solution:
 - Take FB/ZB fill-rate out of the equation

Software

- SMP/SMT
 - Mainstream techniques
 - Everything is simplified by being symmetric
- UMA
 - No partitioning headaches
- OS
 - All 3 cores available for game developers
- Standard APIs
 - Win32, OpenMP
 - Direct3D, HLSL
 - Assembly (CPU & Shader) supported - direct hardware access
- Standard tools
 - XNA: PIX, XACT
 - Visual C++, works with multiple threads ...

Software – Multi Thread

The screenshot shows the Microsoft Visual C++ Disassembly window for a project named 'ParallaxMapping'. The main window displays the source code for the 'Render' function in 'ParallaxMapping.cpp'. The code includes conditional logic for setting pixel shaders and render states. Below the source code, the disassembly view shows the corresponding assembly instructions, including 'SetRenderState' calls. The 'Autos' window at the bottom left shows the current state of variables, and the 'Threads' window at the bottom right shows the execution of multiple threads, including a worker thread and a thread waiting for a single object.

```
if( m_bParallax )
    m_pd3dDevice->SetPixelShader( m_pPixelShaderParallax );
else
    m_pd3dDevice->SetPixelShader( m_pPixelShaderBump );

m_pd3dDevice->SetRenderState( D3DRS_ALPHABLENDENABLE, F.
m_pd3dDevice->SetRenderState( D3DRS_ZENABLE, T1
```

Disassembly view (Address: Sample::Render(void)):

Address	Sample::Render(void)	OpCode	Comment
820C354C	4BFFCE05	b1	@ILT+832 (?SetPixelS
820C3550	38A00000	li	r5,0
820C3554	3880003C	li	r4,60 ; 003C
820C3558	3D40824C	lis	r10,-32180 ;
820C355C	392AE0CC	addi	r9,r10,-7988 ;
820C3560	80690000	lwz	r3,0(r9)

Autos window:

Name	Value	Type
D3DRS_ALPHABLENDE	0x0000003c	int
m_pPixelShaderBump	0x4000f0d0	D3DPixelS
m_pd3dDevice	0x40001c60 {m_pRing=0xf1 D3DDevic	D3DDevice
this	0x7004fae0 {m_Timer={m_Sample *	Sample

Threads window:

ID	Name	Location	Priority	Suspe
fb00000c	KeWaitForSingleOb	KISwapThread	Lowest	0
fb000004	?_ThreadProc@CR	KISwapThread	18	0
f9000000	XexpEntryPointThr	Sample::Render	13	0
f9000004	D3D::WorkerThrea	KISwapThread	13	0

The Xbox 360 Platform

- The Xbox 360 platform delivers breakthrough gaming and entertainment experiences.
- To ignite the next generation of games and entertainment, we're putting the most powerful next generation platform into the hands of the world's greatest game creators
 - High performance hardware
 - Elegant software
 - Innovative services
- Xbox 360 was designed from the ground up, specifically to deliver the best console gaming experience

Summary

- Designed for next generation gaming
- Flexible and Programmable
- Optimized for achievable performance

©2005 Microsoft Corporation.

Microsoft, Xbox 360, Xbox, XNA, Visual C++, Windows, Win32, DirectX3D, and the Xbox 360 logo and Visual Studio logo are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

IBM, PowerPC, and VMX are trademarks of International Business Machines Corporation in the United States, or other countries, or both.

IEEE is a registered trademark in the United States, owned by the Institute of Electrical and Electronics Engineers.

OpenMP is a trademark of the OpenMP Architecture Review Board.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.