Introduction to Xbox 360 Development in XNA

Presented by:
Alexander McCaleb
amccaleb@ucsc.edu

What I'm going to talk about

~Xbox 360 externally
~Specs and Architecture of the Xbox 360
~XNA Framework & the Xbox 360
~Getting started with XNA for Xbox 360
~.NET Compact Framework
~The Evil Checklist: Requirements of all 360 XNA games
~Asynchronous events occurring within the Xbox 360
~The Gamer: Signing in & Getting Setup
~Saving & Loading data
~Optimizing performance on the Xbox 360
~Panorama: A Research Production by CCS
    -Ideals behind the project
    -Taking it from PC to Xbox 360
    -Demo of Panorama on Xbox 360 (so far)
~General Tips

Xbox 360 Externally

- Has both a standard & slim model each with their own different hardware feature sets
- Supports HDMI in later models
- Allows 1-4 players simultaneously
- External guide system to handle system things in-game
- USB support for specialized controllers (e.g. Rock Band instruments)
- Supports custom HDD, memory units & USB flash drives

Specs and Architecture of the Xbox 360

Don't worry if you're unfamiliar with computer architecture

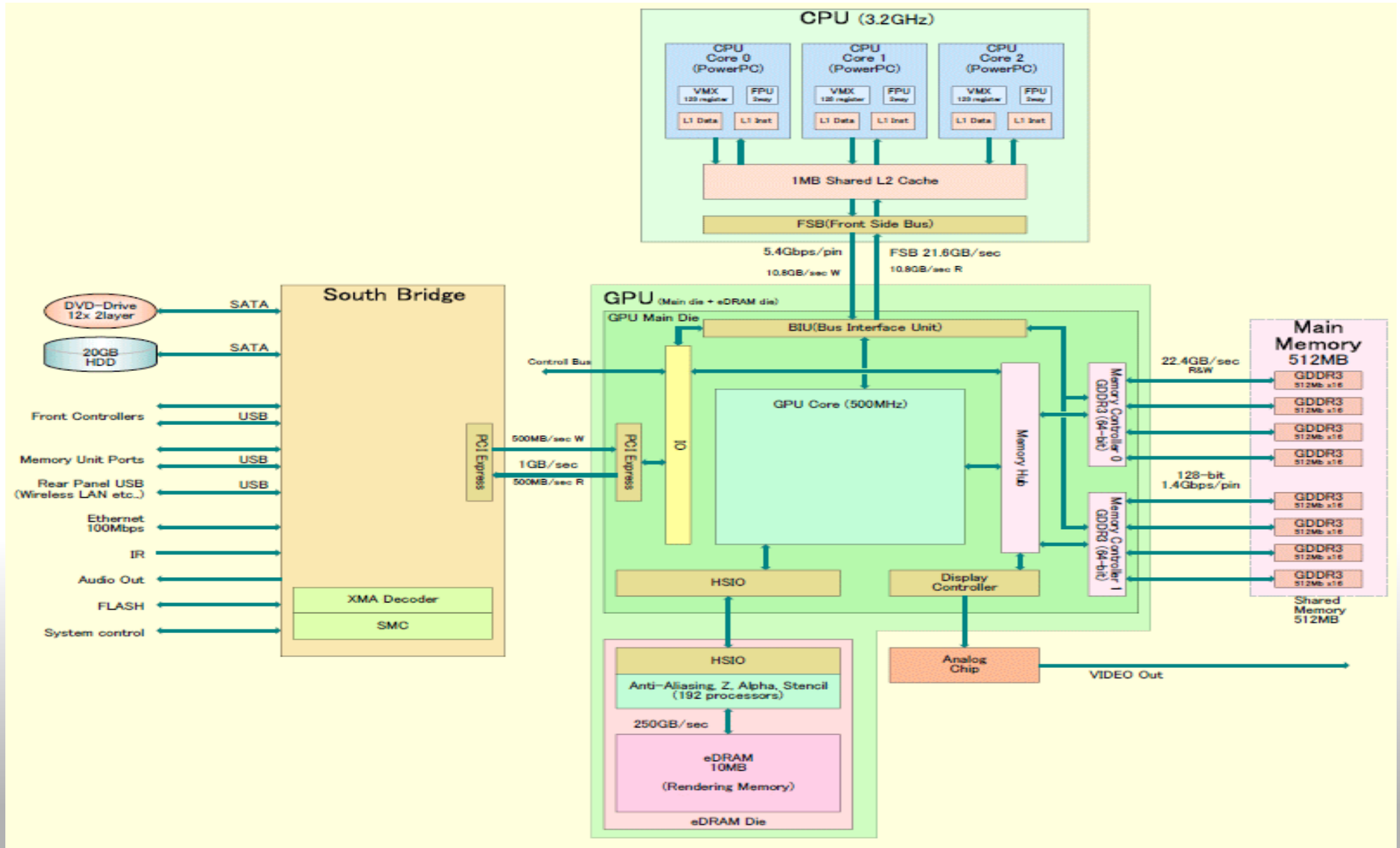
Specs of the Xbox 360

- Triple-core, 3.2 GHz custom CPU
  - Shared 1MB L2 cache
  - Customized vector floating point unit per core
  - 5.4Gbps Front-side-Bus (FSB): 10.8 GB/sec read/write
  - Custom dot product instruction
- 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
- 4 - 320GB hard drive
- Southbridge to handle all I/O

From: http://www.hotchips.org/archives/hc17/3_Tue/HC17.S8/HC17.S8T4.pdf &
http://classes.soe.ucsc.edu/cmpe112/Spring09/notes/21-Xbox360.4p.pdf

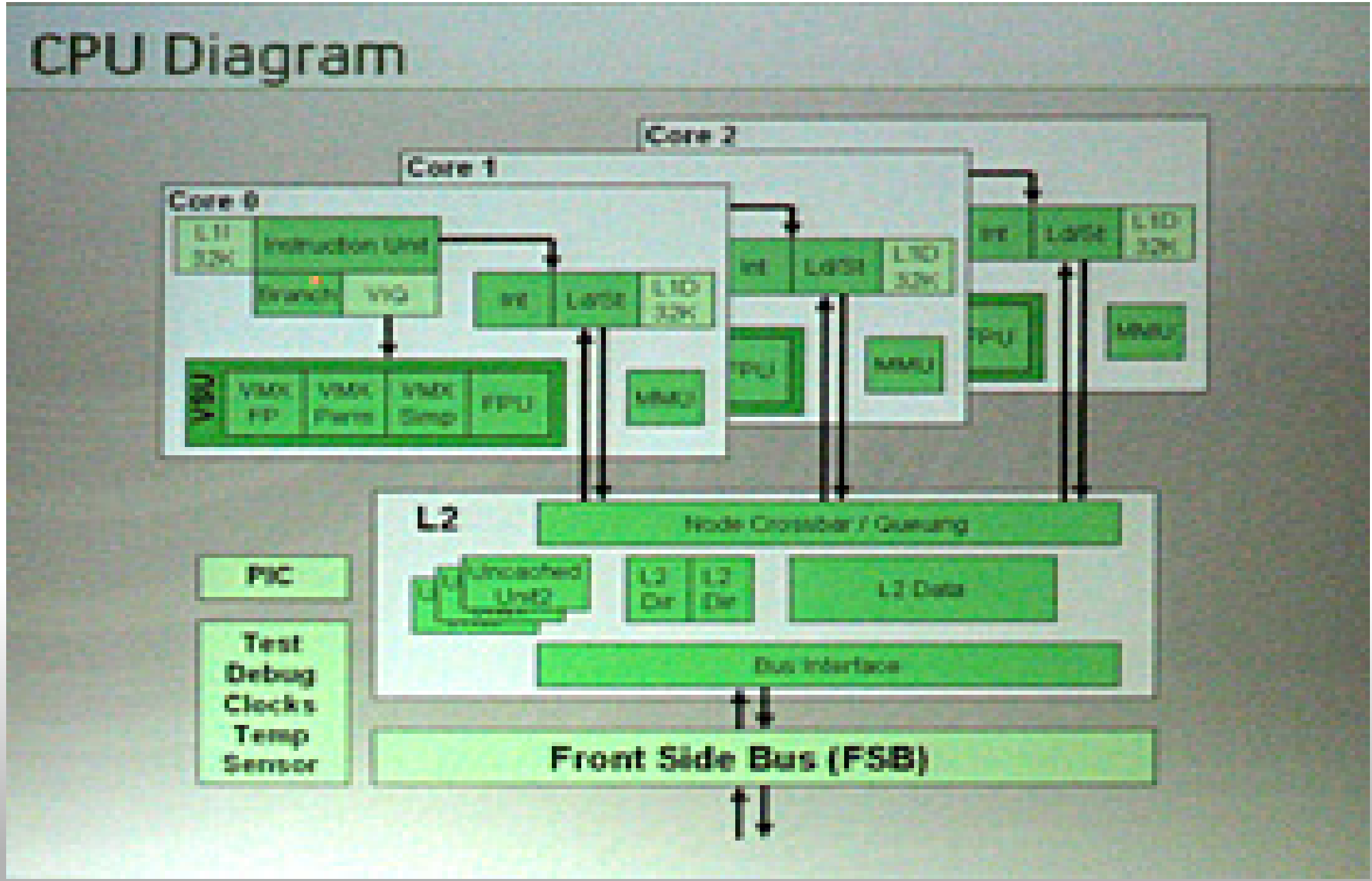
Architecture of the Xbox 360: Top-Level

From: http://www.cdrinfo.com/sections/news/Details.aspx?NewsId=14842



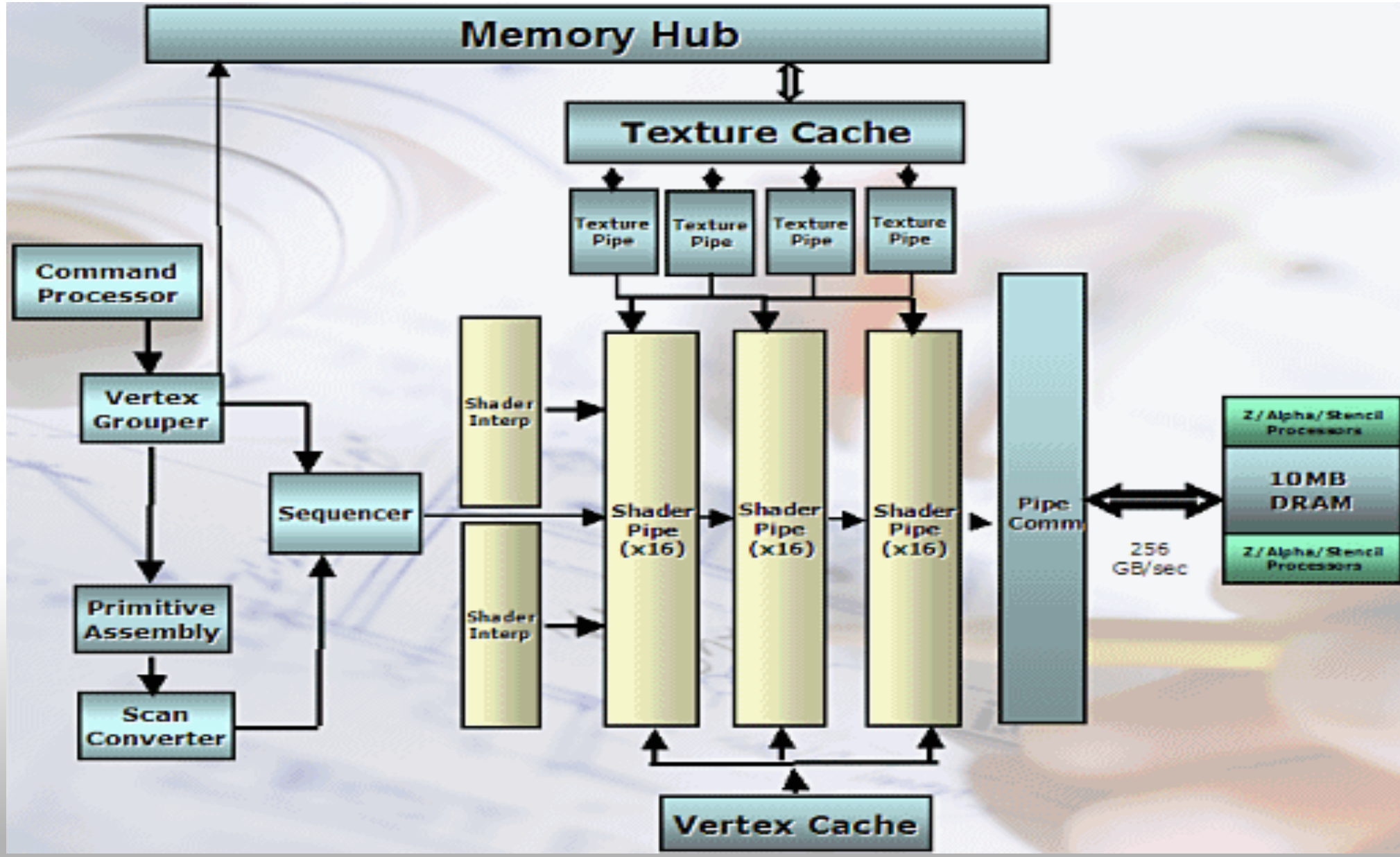
Architecture of the Xbox 360: CPU

From: http://www.hotchips.org/archives/hc17/3_Tue/HC17.S8/HC17.S8T4.pdf



Architecture of the Xbox 360: GPU

From: http://techreport.com/articles.x/8342/1



XNA Framework & the Xbox 360

- XNA Framework designed to allow cross-platform development between PC, Xbox 360, & Windows Phone clients

- Provides access to many Xbox 360 features:
  - Gamertag stats (excluding achievements)
  - Xbox 360 Guide
  - Some Xbox 360 hardware as needed for programming

- When integrating platform-specific code, can use preprocessor, e.g.
  - ```
    #if XBOX
          //Do my cool Xbox stuff
    #endif
    ```
    ```

Getting Started with XNA for Xbox 360

- What you'll need:
  - Microsoft Visual C# 4.0 with XNA 4.0
  - Xbox 360
  - Xbox Live membership (Silver or Gold is fine)
  - App Hub membership ($100/year)
  - A very fast internet connection (preferably wired)
  - XNA Device Connect Center (comes with framework)
  - XNA Game Studio Connect on Xbox 360

- What you'll want beyond this:
  - 4 controllers for testing
  - A CRT and LCD display also for testing
  - In-depth knowledge of the library functions
  - Tolerance for resolving errors that make no sense

.NET Compact Framework

- The underlying framework for the Xbox 360 which is a subset of the .NET Framework for PC

- For performance reasons, some key features of C# aren't supported by the .NET Compact Framework. Examples:
  - System.Windows.Forms namespace
  - System.Web namespace
  - Predicates and any predicate-based functionality
  - Name resolution for classes with the same name in different projects

- Adds ability to map software threads to specific hardware on Xbox 360:

```
System.Threading.Thread.SetProcessorAffinity(int[] cpus)
```

From: http://msdn.microsoft.com/en-us/library/bb198211.aspx

The Evil Checklist

Requirements of Xbox 360 XNA games:

http://create.msdn.com/en-US/resources/help/peer_review_evil_checklist

Standards to Meet for Product Release

- No crashing!
- No significant lag!
- No matter what permutation of controllers you have, can play the game
- All SpriteFonts are legible, no matter the screen size
- UI elements are all visible
- No types of forbidden content exist
- Have a trail mode which demonstrates key gameplay
- Game is represented in an appropriate way
- The game is actually a game, not a wrapper for media content
- All standard Xbox 360 resolutions are supported
- Online play is correctly represented (if applicable)

Conquering Evil: Resolving Resolutions

- Problem:
  - Our game needs to support all of the following resolutions: 480p, 720p, 1080i/p
- Solution:
  - 720p & 1080i/p are both 16:9 resolutions, so we can develop assets assuming 720p & Xbox will scale up to 1080i/p when needed. At the end, need to set our Graphics Device Manager buffer using:
    - `GDM.PreferredBufferWidth/Height() = 1920/1080`
  - For 480p, need a separate code path to handle the aspect ratio change

Conquering Evil: Tile Safe Space

- Problem:
  - The end-user's display may cut off UI elements

- Solution:
  - Ensure that all UI elements are positioned inside the inner 80-90% of the screen. XNA provides this for us:
    - `GraphicsDevice.Viewport.TileSafeArea`

  - TileSafeArea defines the four corners of Tile Safe Area
    - Ex:

```
//Draw the current score string
scoreText = "Score: " + score;
spriteBatch.DrawString(spriteFont, scoreText, new
    Vector2(GraphicsDevice.Viewport.TitleSafeArea.Left,
    GraphicsDevice.Viewport.TitleSafeArea.Top),     Color.
Red);
```

From: Reed, Aaron (2008-11-24). Learning XNA 3.0 (p. 370). O'Reilly Media. Kindle Edition.

Conquering Evil: Logical Player Index

From: http://blog.nickgravelyn.com/2009/03/basic-handling-of-multiple-controllers/

- Problem:
  - The game must be playable, no matter what controller the player(s) ha(s/ve)
- Solution:
  - Have a mapping of logical and physical player indices.
    - Ex: http://blog.nickgravelyn.com/2009/03/basic-handling-of-multiple-controllers/
  - Establish logical order through some prompt screen via

```
for (int i = 0; i < 4; i++)
{
    if (GamePad.GetState((PlayerIndex)i).Buttons.Start ==
ButtonState.Pressed)


    {
        LogicalGamer.SetPlayerIndex(LogicalGamerIndex.One,
(PlayerIndex)i);
```

Asynchronous events occuring within the Xbox 360

- Newer consoles have a large amount of overhead with their own systems that we need to account for

- Problem:
  - The Xbox Guide runs independently of the game itself, but oftentimes, behaviors in-game are dependent on properties established in the Guide, e.g. signed in?

- Solution (easiest):
  - Remember basics of interrupt-driven I/O & treat all guide interaction as an interrupt
    - Ex:

```
if(Guide.IsVisible) { //Do nothing in our Update and Draw}
```

The Gamer: Signing in & Getting Setup

- While playing an Xbox game, we need to know who's actually playing & what baggage that player has with them
- Most significant baggage: StorageDevice
- XNA provides quite a bit of functionality to handle storage, but implementing these means ourselves is painful
- Quickest resolution: Nick Gravelyn's EasyStorage
  - Simplifies storage process to defining the devices, container strings, & a few events for communication
  - Tutorial using GameStateManagement example: http://robotfootgames.com/xna-tutorials/92-xna-tutorial-savingloading-on-xbox-360-40

Saving & Loading data

- XNA gives two spaces for storage:
  - TitleStorage:
    - Where all our game assets exists including the code
    - Writable on PC, but not on 360
      - DON'T WRITE TO TITLESTORAGE
  - UserStorage:
    - Space where we can store whatever we need
      - Windows: /My Documents/SavedGames/Game1
      - 360: The Storage Device found from before
    - Create files here containing whatever text you need
      - Can be variable values, XML, whatever

Optimizing Performance on the Xbox 360

Tricks employed by the XNA community

Performance issues on the 360

From: http://ianqvist.blogspot.com/2010/11/optimizing-performance-for-xbox-360.html & http://www.hotchips.org/archives/hc17/3_Tue/HC17.S8/HC17.S8T4.pdf

- A PC with equivalent hardware to a 360 will still outperform the 360... WAT?
  - Software overhead on the 360 is exponential
- .NET Compact Framework not design for FP performance, but 360 hardware is... WAT?
- Virtual methods clash with Just-In-Time (JIT) compilation
- .NET Compact Framework chokes on efficient inlining
- Instruction cache is only 2-way set-associative
  - More complex lines of code are subject to misses

Resolving these issues

From: http://ianqvist.blogspot.com/2010/11/optimizing-performance-for-xbox-360.html & http://www.hotchips.org/archives/hc17/3_Tue/HC17.S8/HC17.S8T4.pdf

- Try to keep interaction with outside software to a minimum
- Use floating-point operations only if absolutely needed
- Minimize virtual method use
  - If absolutely necessary, classify the overrides as sealed
- Manually inline code wherever possible
- Factor lines of code such that the amount of assembly instruction to execute is a multiple of 2
  - Some basic assembly knowledge helps

Reducing Garbage Collection on 360

From: Various online resources and practices of Chronic Logic

- Many of C#'s strengths in readability aren't friendly to the Garbage Collector

Structure	Problem	Solution
Dictionaries	Enums as keys have vast overhead.	Cast enums to int, or just use ints as keys to begin with.
Strings	Concatenating Strings at runtime is expensive as each piece concatenated is another allocated string.	Use StringBuilder objects instead as the pieces added aren't allocated on the heap.
Objects	Every new object allocated is done so on the heap and some may only be active for a short time.	Use singleton patterns for things you really only need one instance of. In other cases, use pools (like with Particles).
Boxed Objects	For boxing, need to allocate a new, unneeded object. Type casting for unboxing is also expensive.	DON'T BOX OBJECTS!
Foreach	Traversing in this manner implicitly creates an IEnumerator that the garbage collector handles once we leave the loop.	Use regular for loops or recycle the IEnumerator used in foreach calls.

Panorama

A Research Production by
Computational Cinematics Studio

Ideals behind Panorama

From: http://games.soe.ucsc.edu/project/panorama

- Panorama is a photography-based game where photos taken are evaluated both by user and textbook methods
- Machine learning algorithms are employed to figure out photography preferences between textbook methods & user feedback
- Pairing these two feedback methods with knowing who took the photo, we can evaluate their photo preferences.
- Upcoming milestones:
  - Incorporate color
  - Allow comparisons within a global database

Panorama: From PC to Xbox 360

- With Xbox functionality thrown to the backburner, many PC-exclusive implementations were done
- Some core functionality had to be completely rewritten to account for this.
- Performance issues also came into consideration with level-generation
- Internals of .NET Compact Framework forced some structural refactoring
- Port ~85% complete

Panorama Demo

General Tips

- Know the pros & cons of a platform before using it
- Code with all platforms considered
- Have unique class names across all projects
  - .NET Compact framework chokes on this
  - Avoids much confusion within team
- Capitalize on the hardware threads!
- Always have access to documentation
- Know the product standards before you begin your project

Questions/Comments?

Thanks & Happy Coding!

Email: amccaleb@ucsc.edu
Website: people.ucsc.edu/~amccaleb