Graphic Architecture introduction and analysis

劉哲宇,Liou Jhe-Yu

Outline

- GPU Architecture history
- Taxonomies for Parallel Rendering
  - Tile-based rendering
- Fixed function pipeline
- Separated shader architecture
- Unified shader architecture
- Conclusion

Architecture history

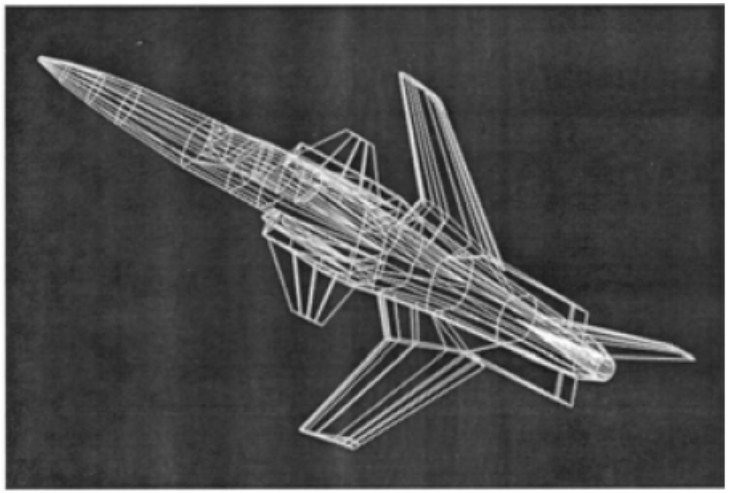
Silicon Graphics, Inc.

- 1981-2009
- 3D graphics accelerator in early age.
- Focus on high-performance graphics server.
- Release its own IRSI-GL API in 1992,
  - The first open and industrial standard
  - IRIS-GL -> OpenGL

Architecture history
From ashes

- 1st generation – wireframe
  - transform, clip, and project
  - color interpolation

- Model
  - SGI Iris 2000, 1984



Architecture history
From ashes

- 2nd generation – shaded solid
  - Lighting calculation
  - Depth buffer, alpha blending.

- Model
  - SGI GTX, 1988

Architecture history
From ashes

- 3rd generation
  - Texture mapping
- Model
  - SGI RealityEngine, 1992

Architecture history
in the later time(after 2000)

- Fixed function pipeline (- 2000)
  - With or without T&L engine

3DMark 2000

- Vertex and Pixel shader (2001 - 2006)
  - 4th generation

3DMark 03

- Unified shader (2007 -)
  - 5th generation

3DMark 11

Architecture history
Fixed pipeline production

- Desktop
  - NVIDIA GeForce 2 - NV15 (2000)
  - ATI Radeon 7000 - R100 (2000)
  - 3dfx Voodoo3 (1999)
    - Acquired by NVIDIA in 2002
  - S3 savage 2000 (1999)
    - Acquired by VIA in 2001
  - Imagination STG Kyro - PowerVR3 (2001)
- Mobile
  - Imagination MBX - PowerVR3(2001)
  - ATI Imageon 130 (2002)
    - Sell to Qualcomm in 2009 and rename Imageon as Adreno which be integrated in their Snapdragon SoC.
  - Falnax Mali 55 (2005)
    - Acquired by ARM in 2006

NV GeForce 2

Architecture history
Vertex shader & Fragment shader

- Desktop
  - NVidia GeForce 7900 - G71 (2006)
  - ATI X1900 – R520 (2006)
- Mobile
  - NVidia GeForce ULP (2011)
    - In NVidia's Tegra 3.
  - Imagination SGX545 (2010)
  - ARM Mali 400 (2009)
  - Qualcomm Adreno 220 (2010)

NV GeForce 7900

Architecture history
Unified shader

- Desktop
  - NVidia GeForce 680 – GK104 (2012)
  - AMD HD7970 (2012)
- Mobile
  - Imagination G6400 (2012)
  - ARM Mali T678 (2012)
  - Qualcomm Adreno 320 (2012)

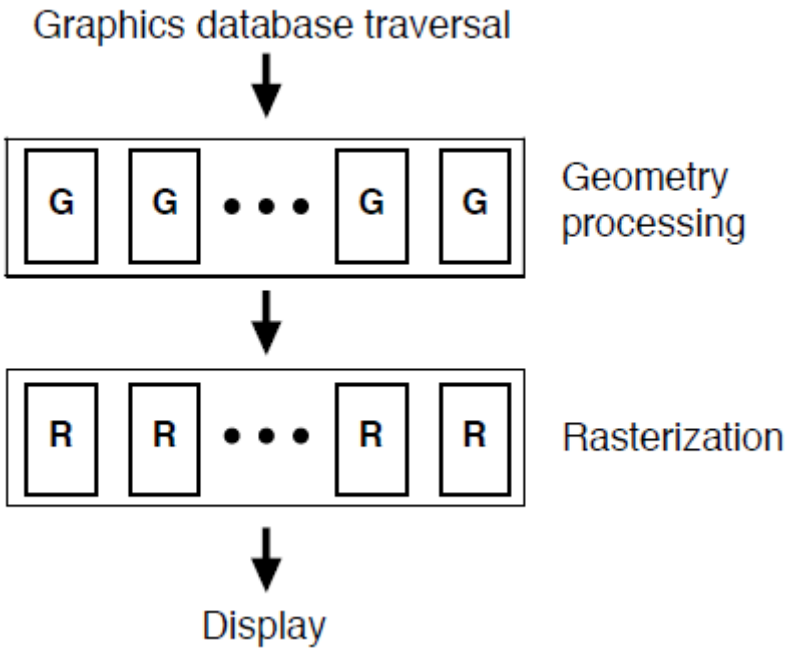
NV GeForce 680

Outline

- GPU Architecture history
- Taxonomies for Parallel Rendering
  - Tile-based rendering
- Fixed function pipeline
- Separated shader architecture
- Unified shader architecture
- Conclusion

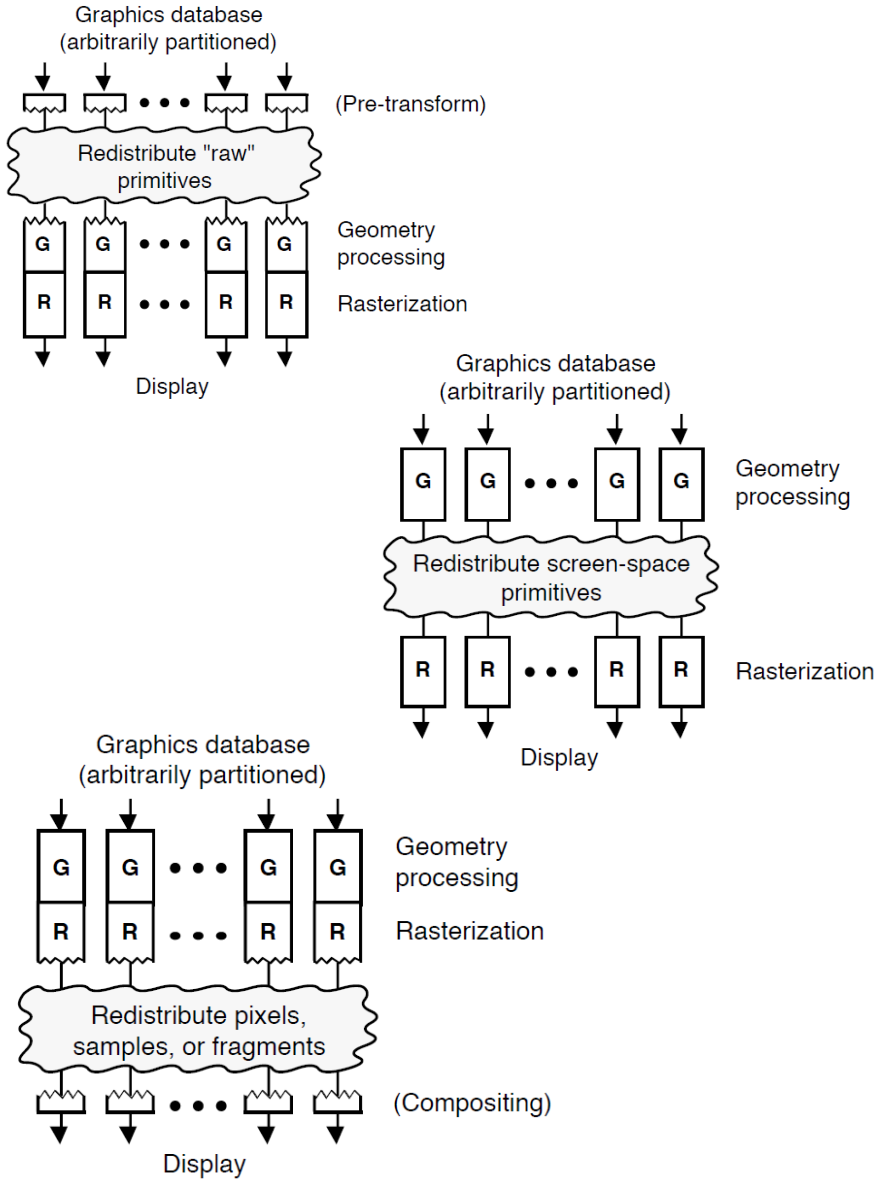
Taxonomies for Parallel Rendering

- Ideal parallel rendering

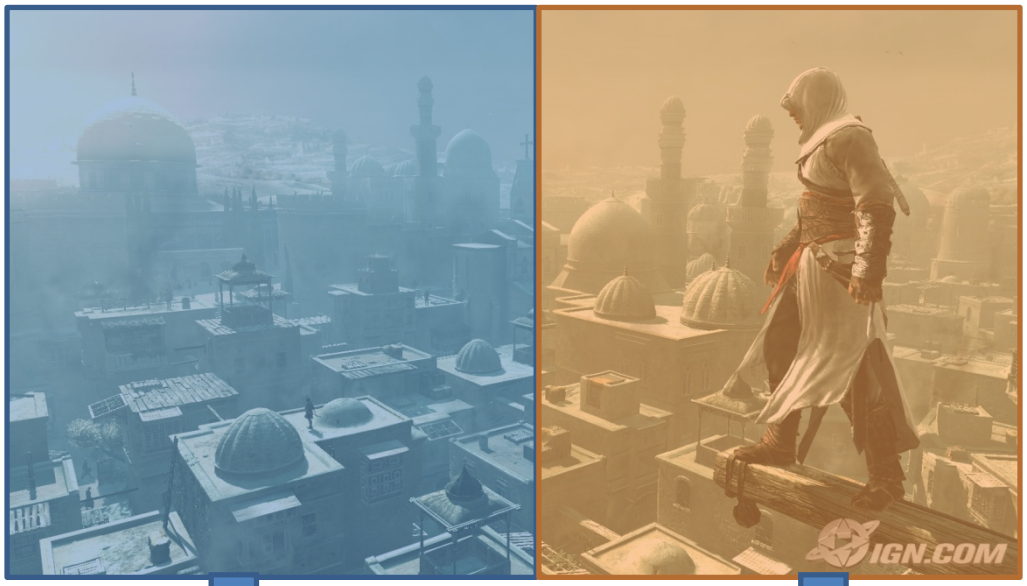


Taxonomies for Parallel Rendering

- Sort-first
  - Work in multi GPU (NVIDIA's SLI, AMD crossfire).
- Sort-middle
  - Tiled-based rendering
  - Intel Larrabee, and most mobile GPU.
- Sort-last
  - Immediate mode rendering
  - Most Desktop GPUs belong to this area.



Sort-first
On SLI application

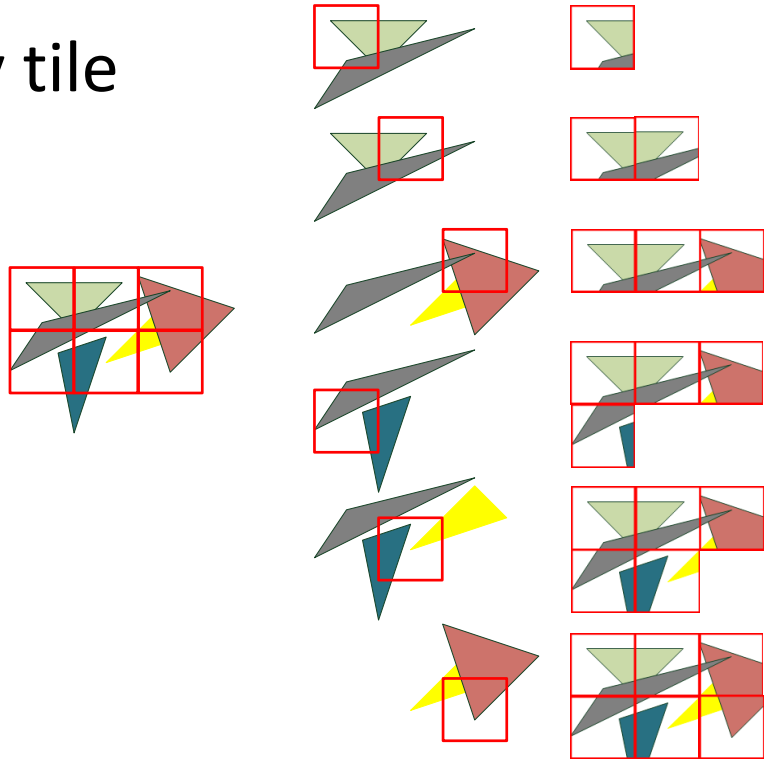


Sort-first

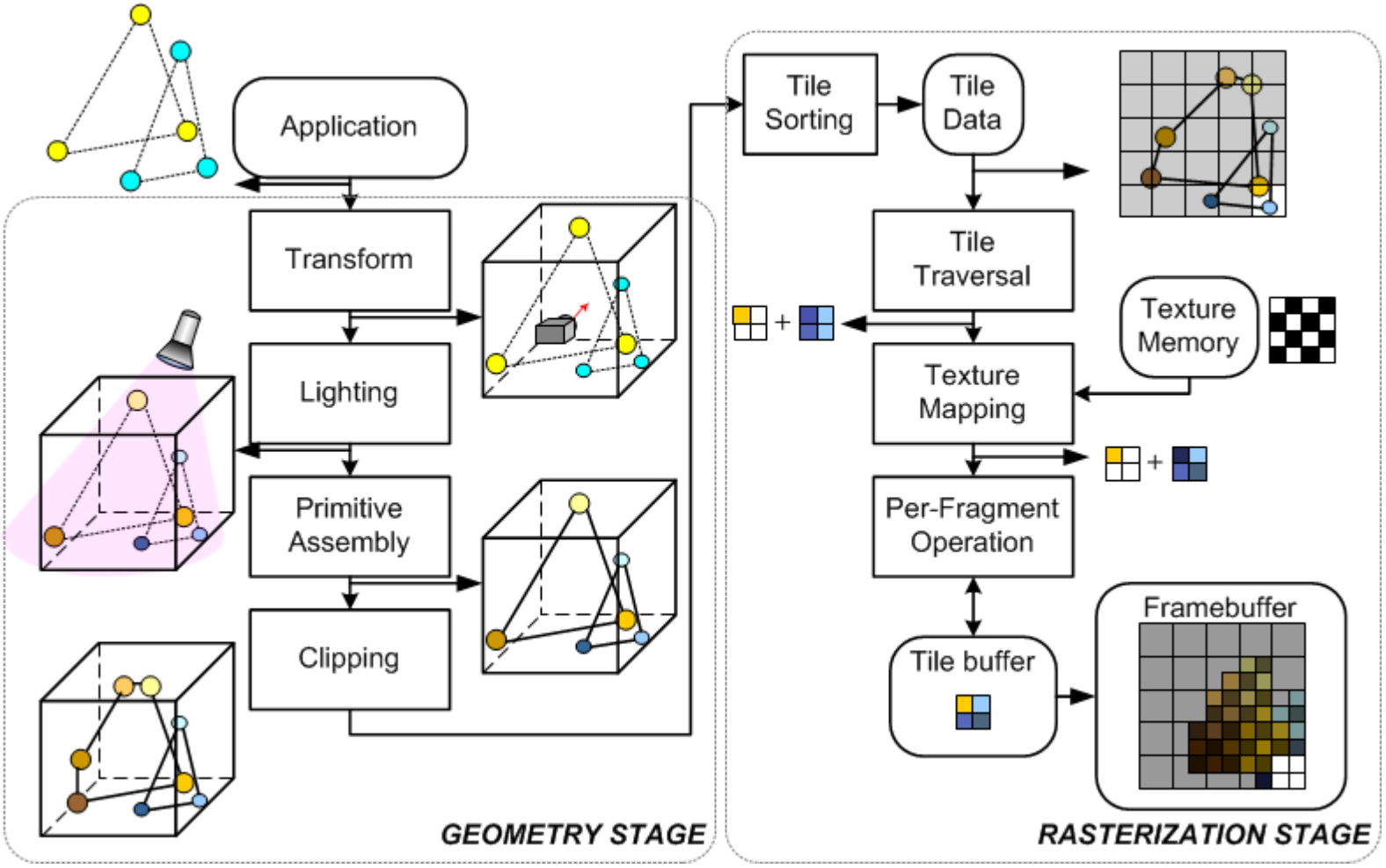
- Advantage
  - Graphic Pipeline will receive no interrupt.
  - Easy to re-distribute job on existed hardware.
- Disadvantage
  - Pre-transform are required
    - Harm the system performance obviously.
  - Enormous bandwidth requirement on frame buffer access

Sort-middle – Tiled-based rendering

- Sorting triangles into tiles after geometry stage.
- Raster engine wait until all triangles have been sorted.
- Raster engine process tile by tile sequentially.

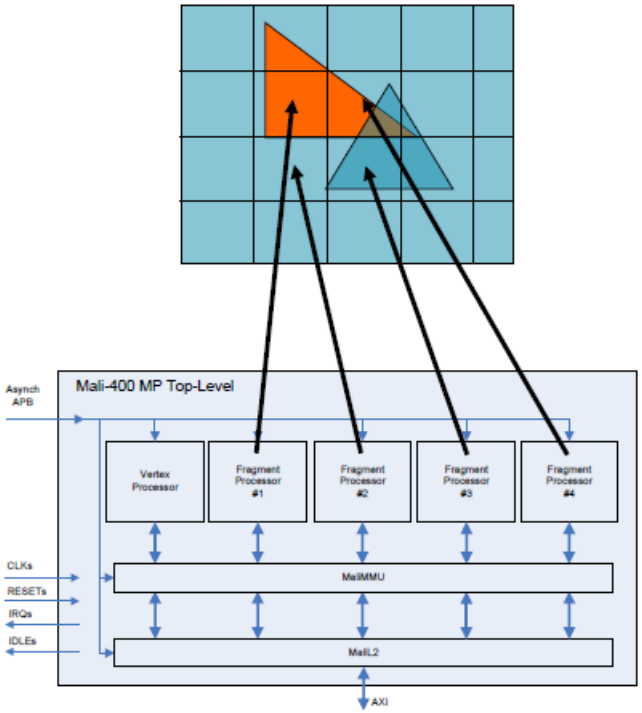


Tiled-based rendering

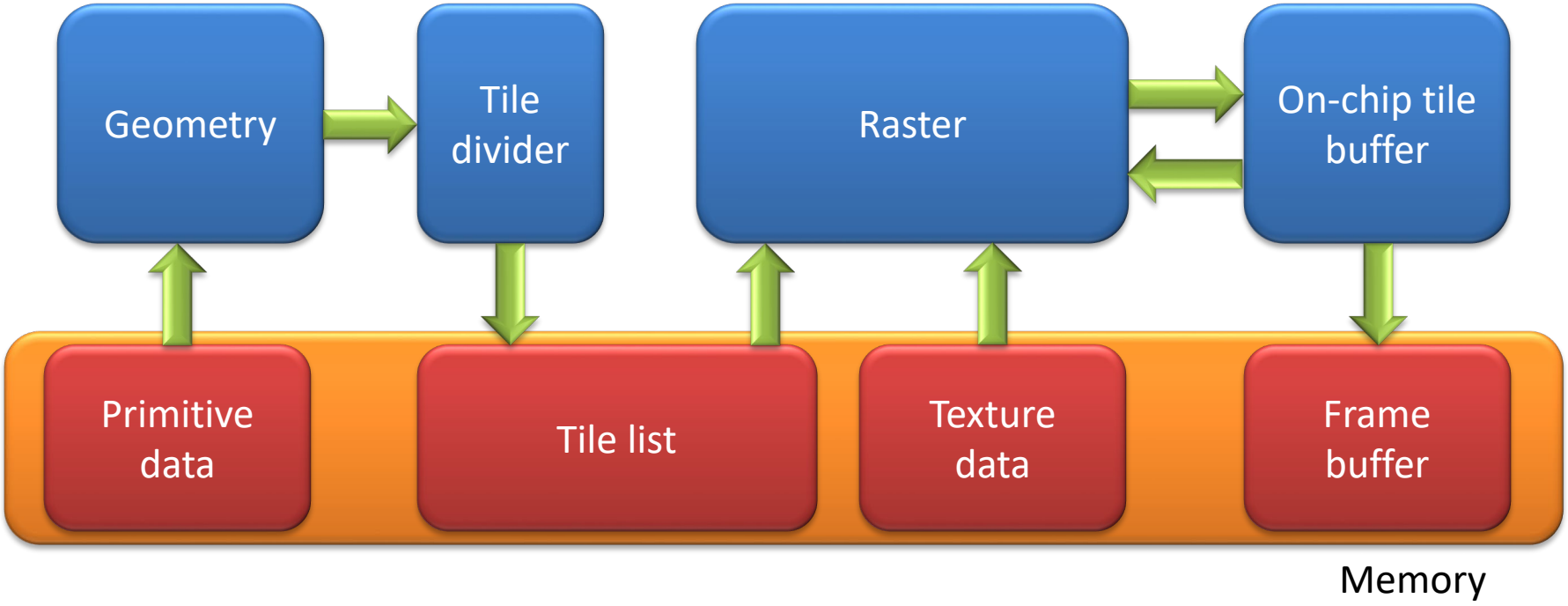


Tiled-based rendering
Case study – ARM MALI 400

- All cores work in parallel on separate tasks
- Each core processes one tile at a time until completion – no communication between cores
- Tiles assigned statically to cores in a swizzled order
- Tile processing order maximizes L2 hit rate for polygon descriptors, textures



Tile-based Memory access model

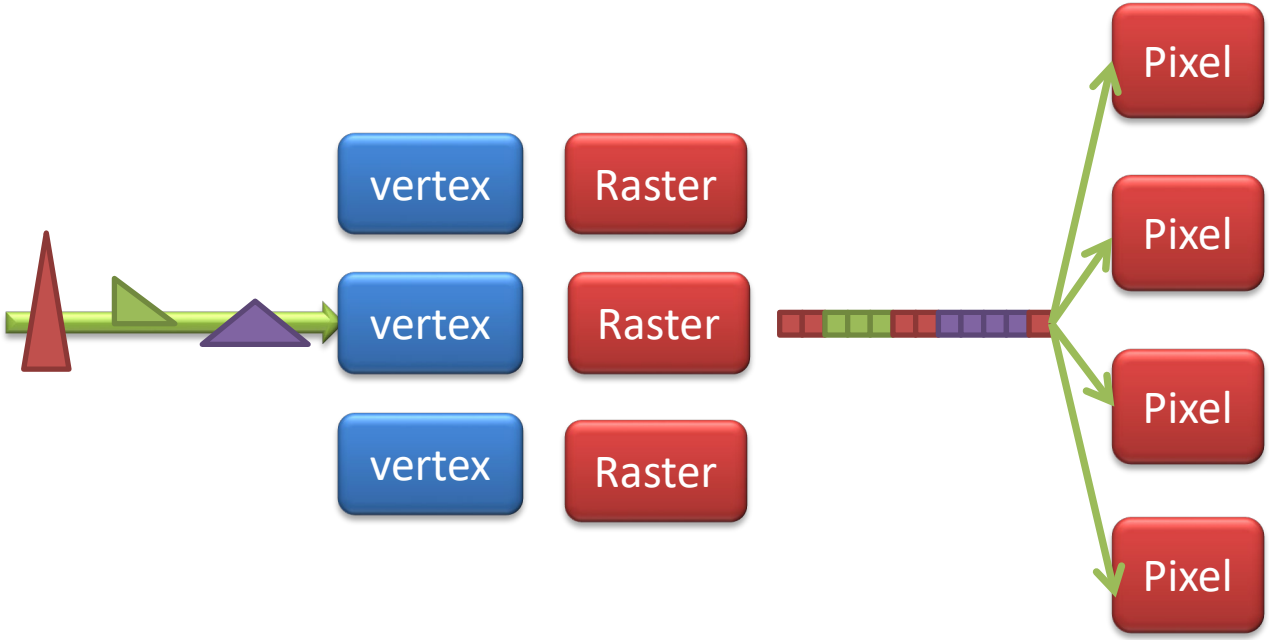


Tiled-based rendering

- Advantage
  - Natural way to re-distribute jobs.
  - Save a lot of bandwidth for communication with frame buffer.
- Disadvantage
  - One triangle may go into multiple tiles.
  - Need a sorting buffer after triangle sorting
    - More complex scene, more memory access.
  - Completely divide graphic pipeline into two partition.
    - This may harm the performance.

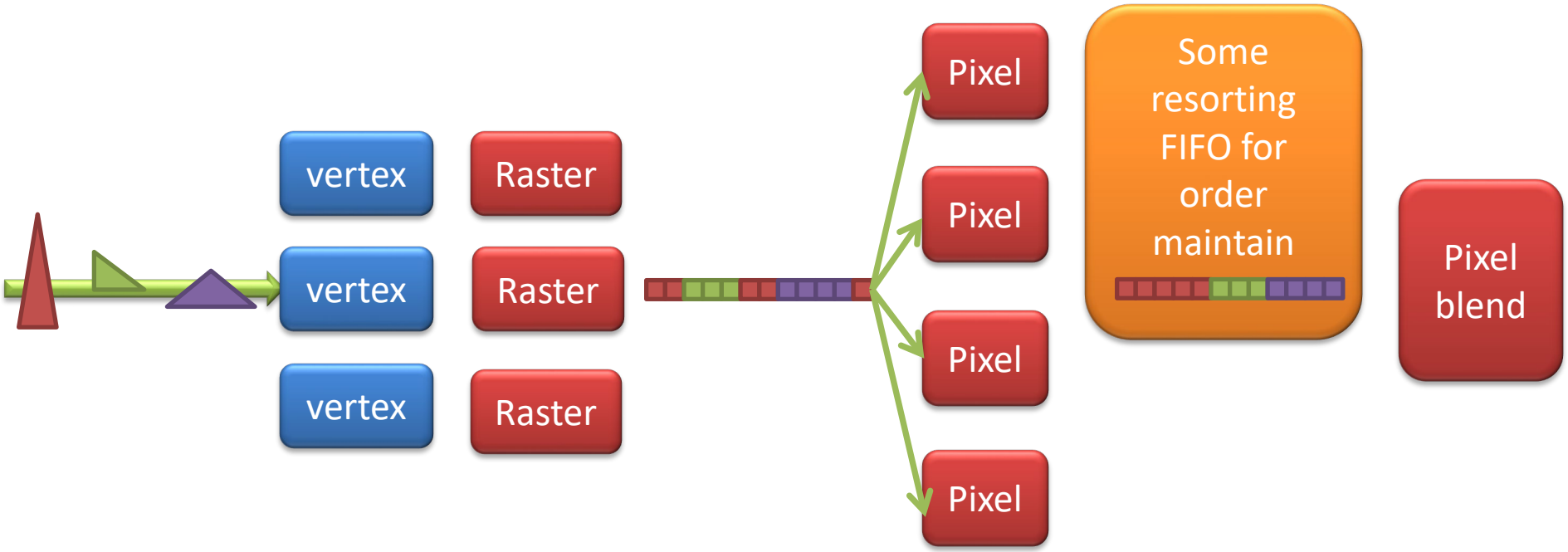
Sort-last

- Delay sorting until decomposing primitive into fragment.



Sort-last

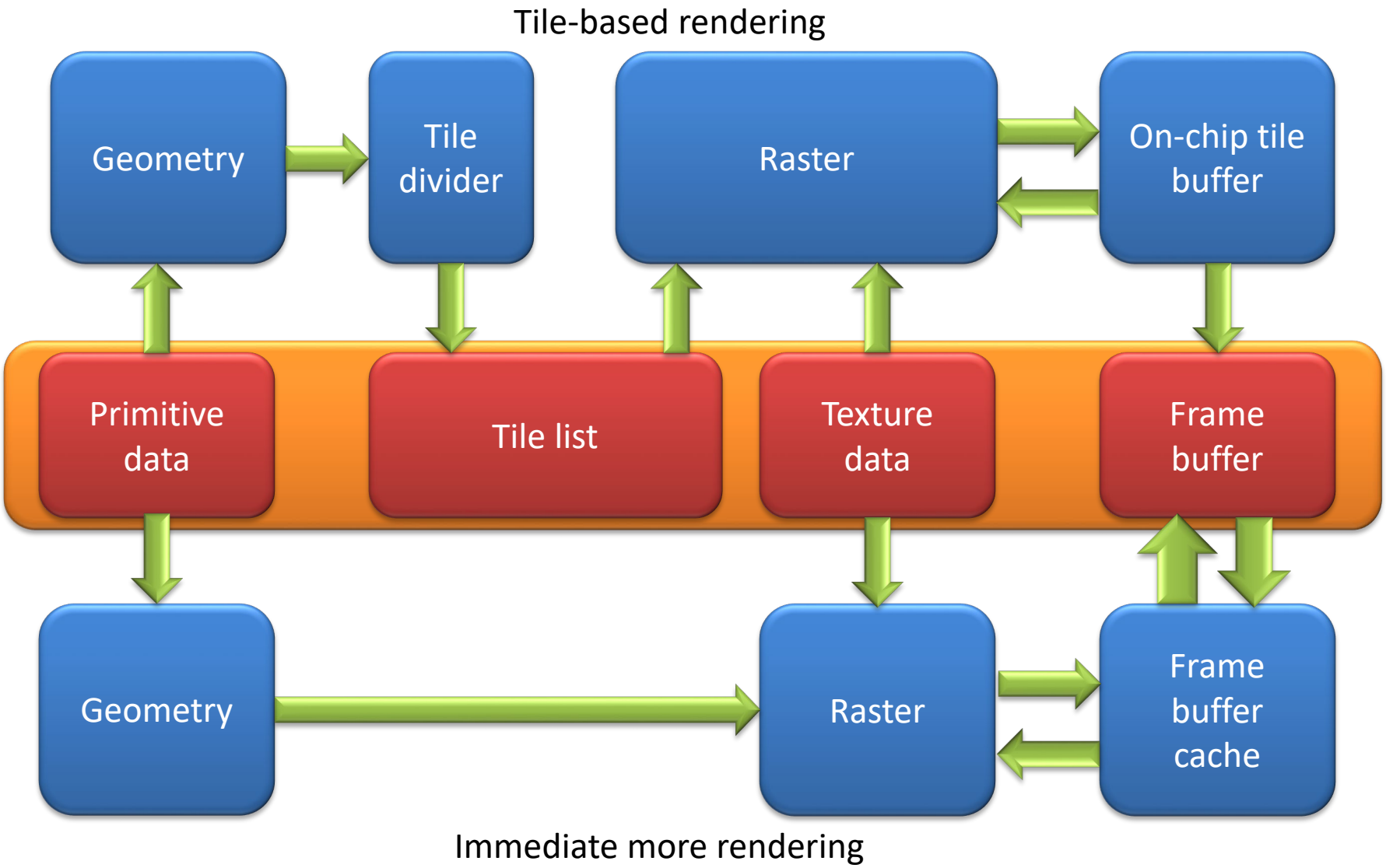
- Graphic API has the strict limit on order rendering
  - EX. The pre-sort operation for Alpha blending race condition.



Sort last

- Advantage
  - Full rendering pipeline without interrupt until per-fragment operation (compare to sort-middle)
- Disadvantage
  - Huge bandwidth requirement on frame buffer access, particularly in high resolution and anti-aliasing enable.

Memory access model



Why does mobile GPU like sort-middle

- Minimize bandwidth requirement.
  - Need system memory support.
  - High memory bandwidth usage will hammer the whole system.
- Save more penguins (power).
  - Reducing memory access means less power consumption.

Why does desktop GPU use sort-last

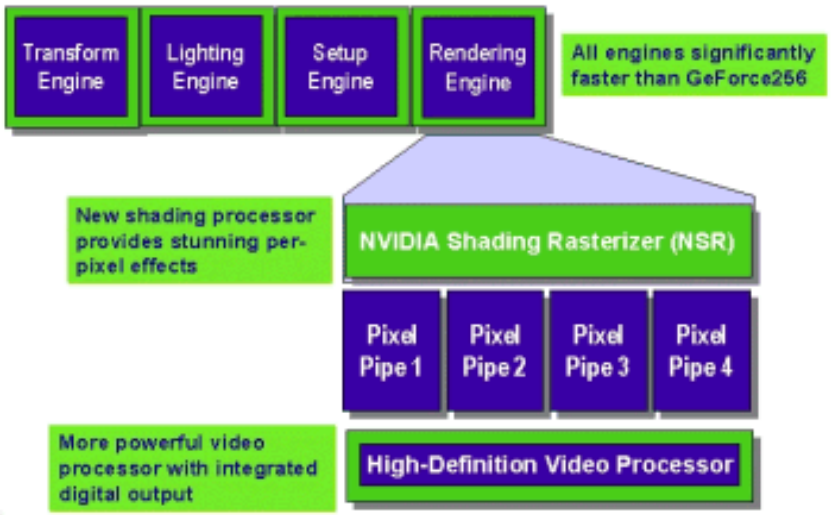
- Minimize performance issue
  - Sorting cost is low
- Desktop GPU has its own dedicated memory.
  - Graphics DRAM usually has high bandwidth with high latency.

Outline

- GPU Architecture history
- Taxonomies for Parallel Rendering
  - Tile-based rendering
- Fixed function pipeline
- Separated shader architecture
- Unified shader architecture
- Conclusion: GPU architecture issue

Fixed function pipeline

- NVIDIA GeForce 256 (1999)
  - First transformation and lighting(T&L) hardware
    - Become the market leader due to this production.
  - NVIDIA mark it as the world's first GPU.
  - 1 vertex pipeline, 4 pixel pipeline

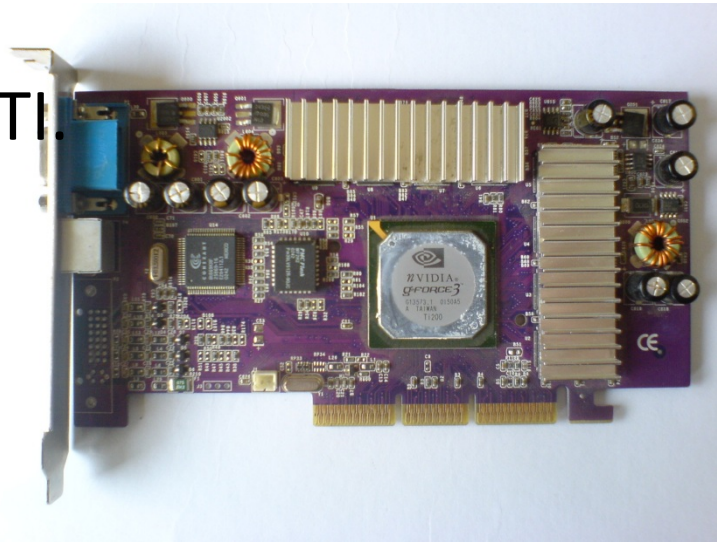


Memory access latency?

- GPUs seldom care the memory latency because
  - Usually hundreds of fragments fly in the raster pipeline(queue).
  - Switch to the next fragment with tiny cost if texture cache/frame cache miss.
    - No pipeline stall is needed.

Separated shader architecture

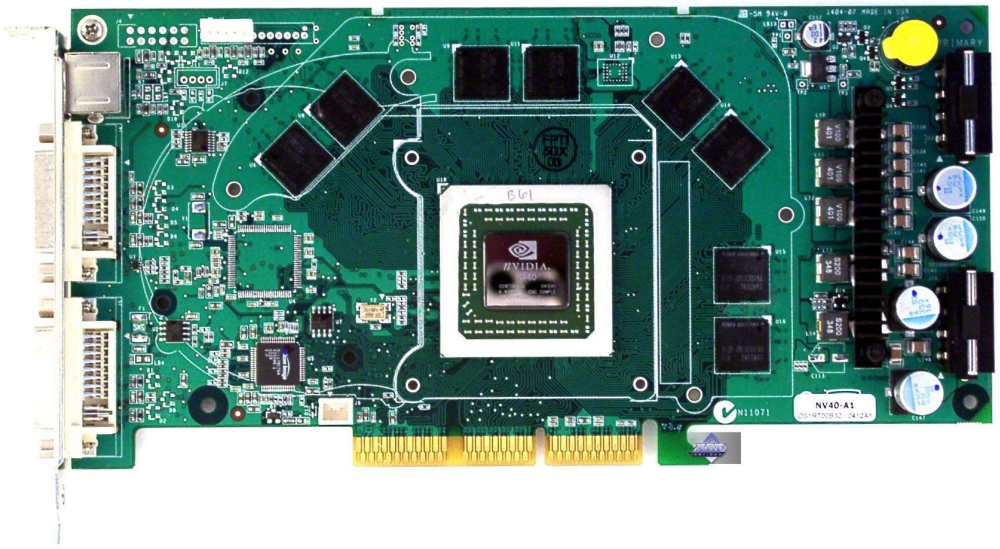
- NVIDIA Geforce 3 (2001)
  - First vertex shader and pixel shader architecture on desktop GPU.
  - 1 vertex shader, 4 pixel shader
  - Using Assembly code to program shader by microsoft shader model 1.0
  - Kill other competitors except ATI.
    - 3dfx, S3, sis, Maxtor, PowerVR



Separated shader architecture

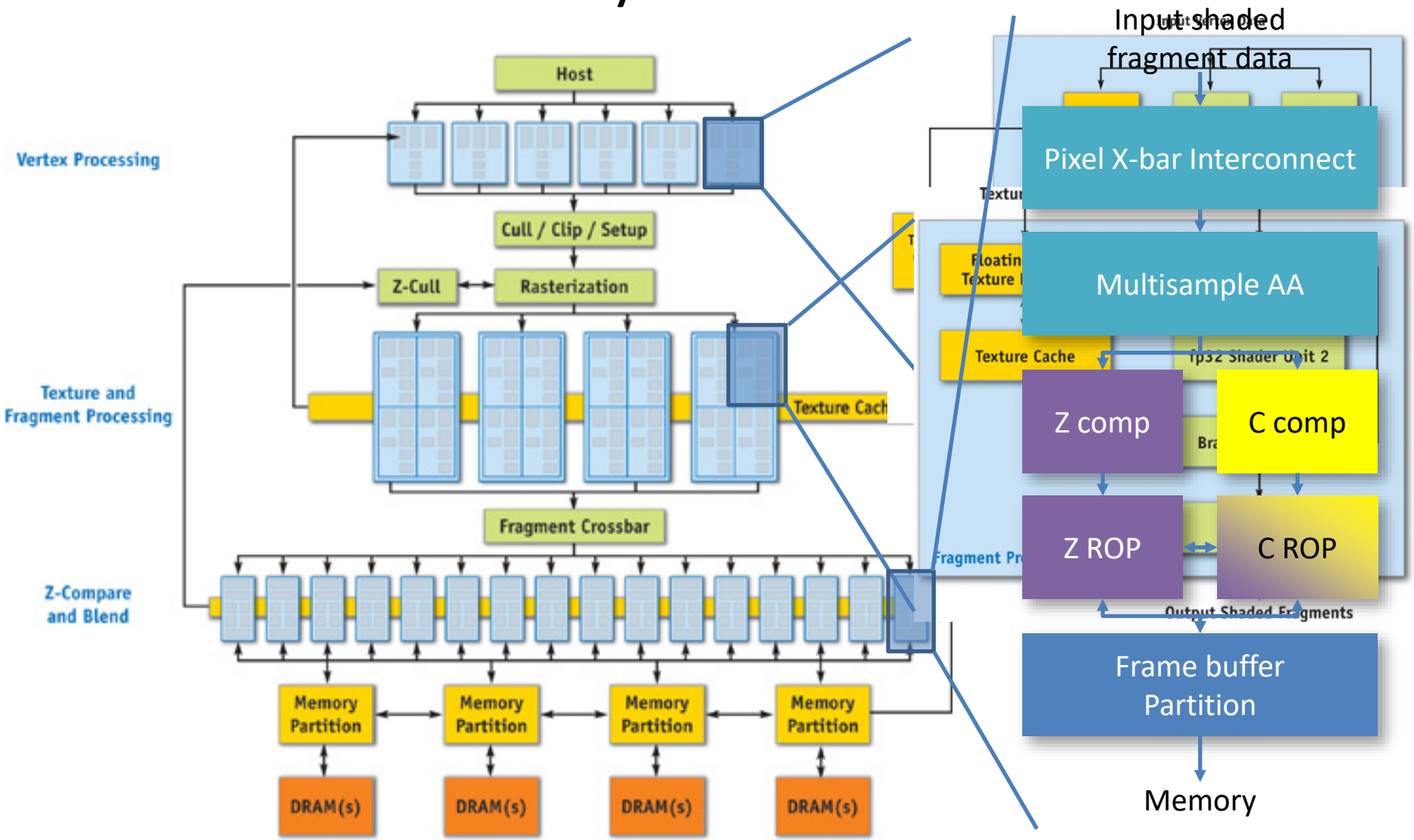
Case Study: GeForce 6 series

- NVIDIA GeForce 6800 (2004)
- 6 vertex shader
- 16 pixel shader,
- 16 fragment operation pipeline

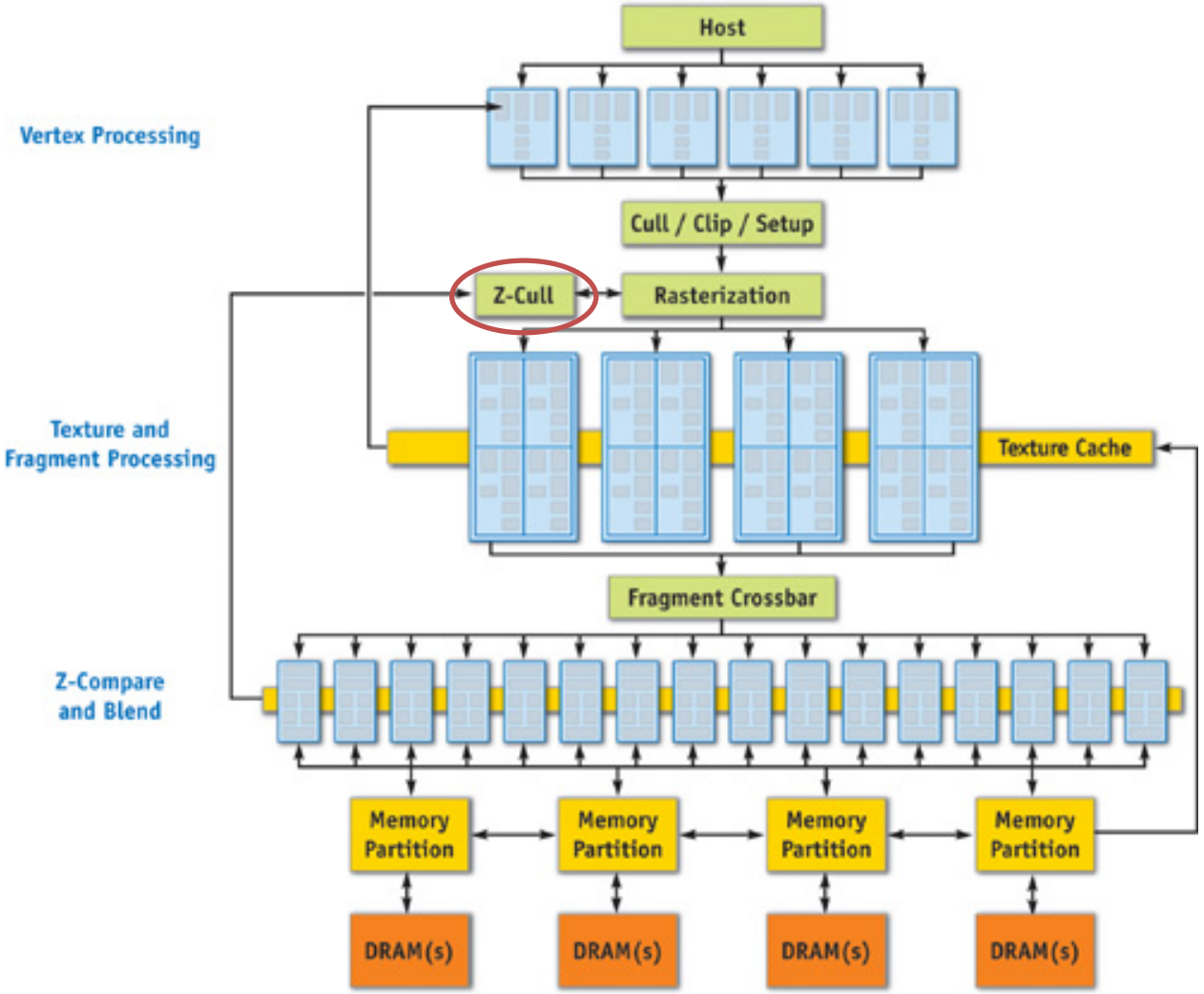


Separated shader architecture

Case Study: GeForce 6 series

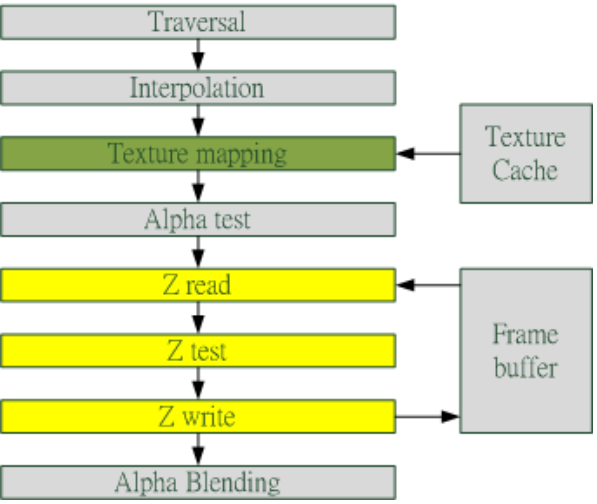


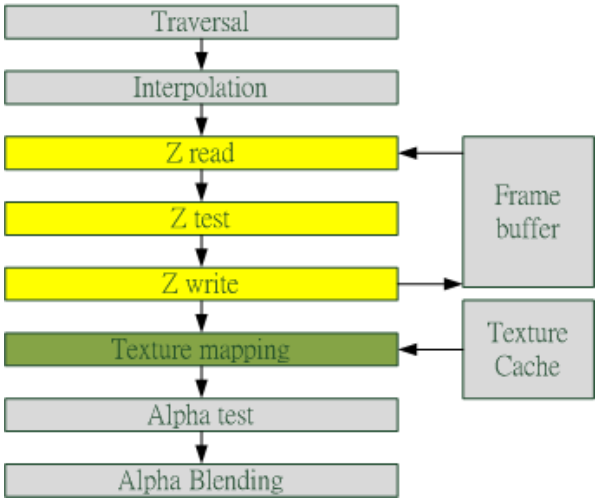
Early Z-test

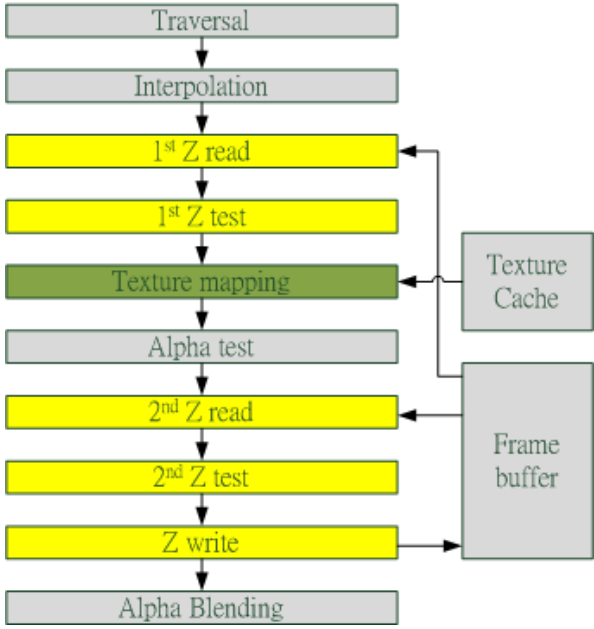

Host
Vertex Processing
Cull / Clip / Setup
Z-Cull
Rasterization
Texture and
Fragment Processing
Texture Cache
Fragment Crossbar
Z-Compare
and Blend
Memory
Partition
Memory
Partition
Memory
Partition
Memory
Partition
DRAM(s)
DRAM(s)
DRAM(s)
DRAM(s)

Early Z-Test concept

- Put depth test before texture mapping to avoid unnecessary texel fetching.
- Reduce the memory traffic (reduce texture cache miss)





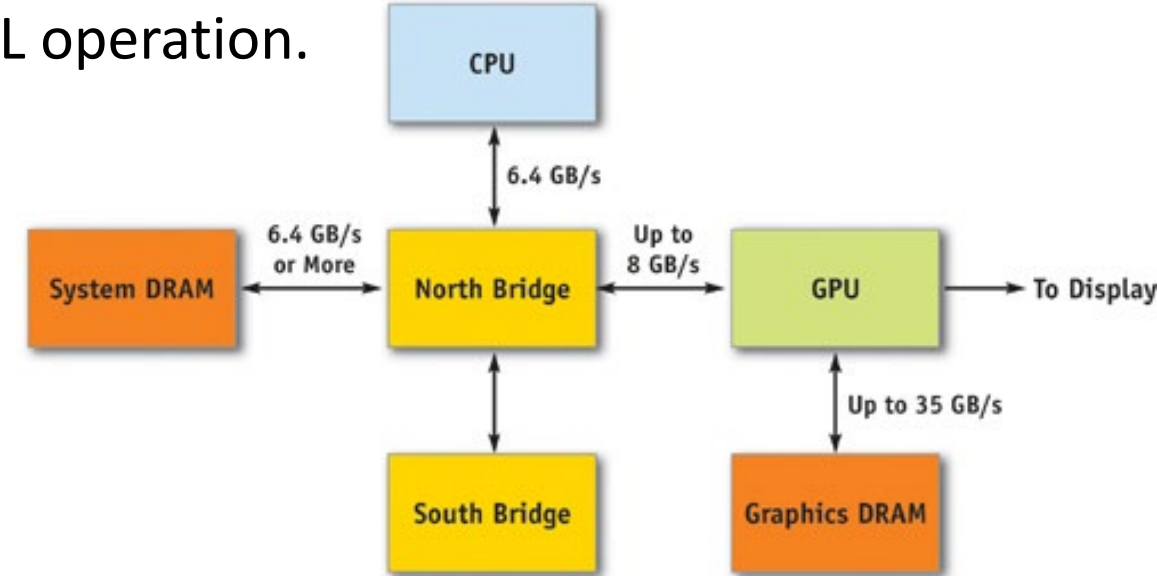


Outline

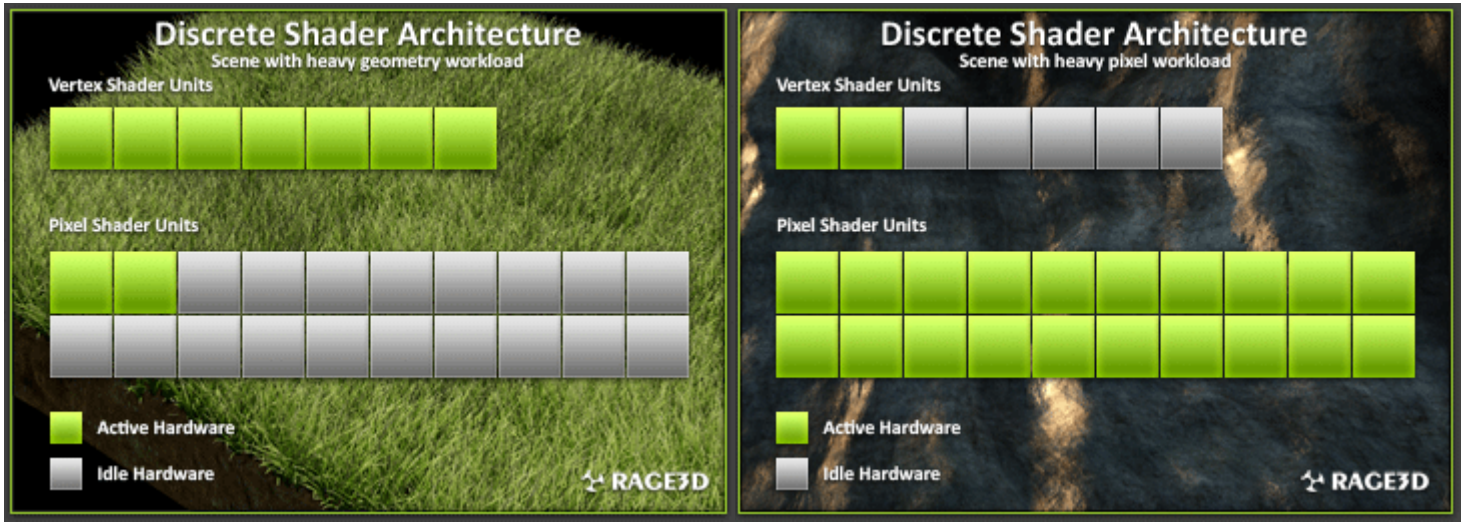
- GPU Architecture history
- Taxonomies for Parallel Rendering
  - Tile-based rendering
- Fixed function pipeline
- Separated shader architecture
- Unified shader architecture
- Conclusion

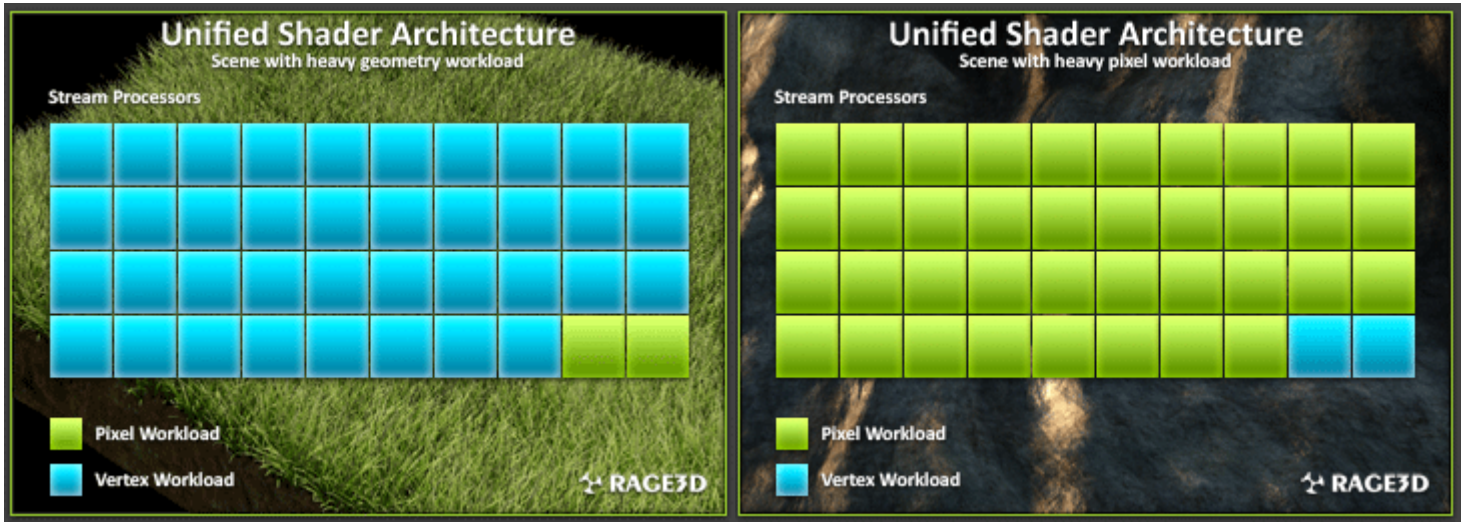
Unified shader Architecture

- Separated shader architecture limits the graphic application.
  - The input data rate is obviously slow than vertex shader process speed.
    - CPU's processing speed is slow than GPU.
  - Force programmer to use more texture operations and less polygon/simple T&L operation.



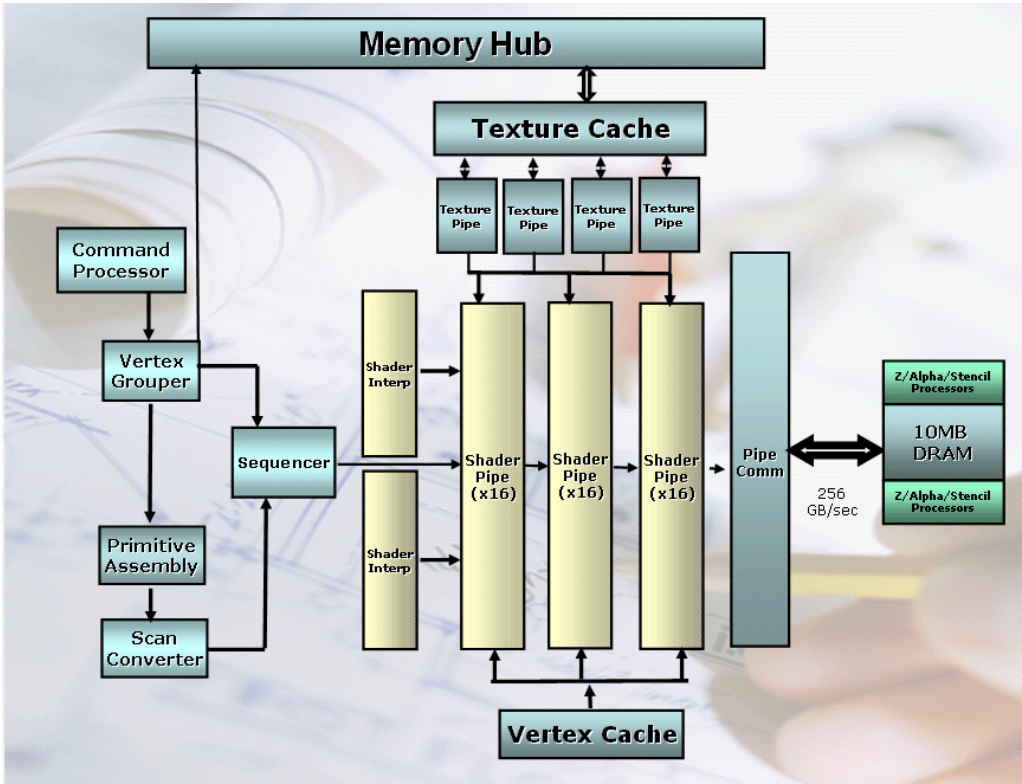
Unified shader Architecture





Unified shader Architecture

- ATI Xenos on Xbox 360 (2005)
  - The world first unified shader architecture.
- 48 unified shaders

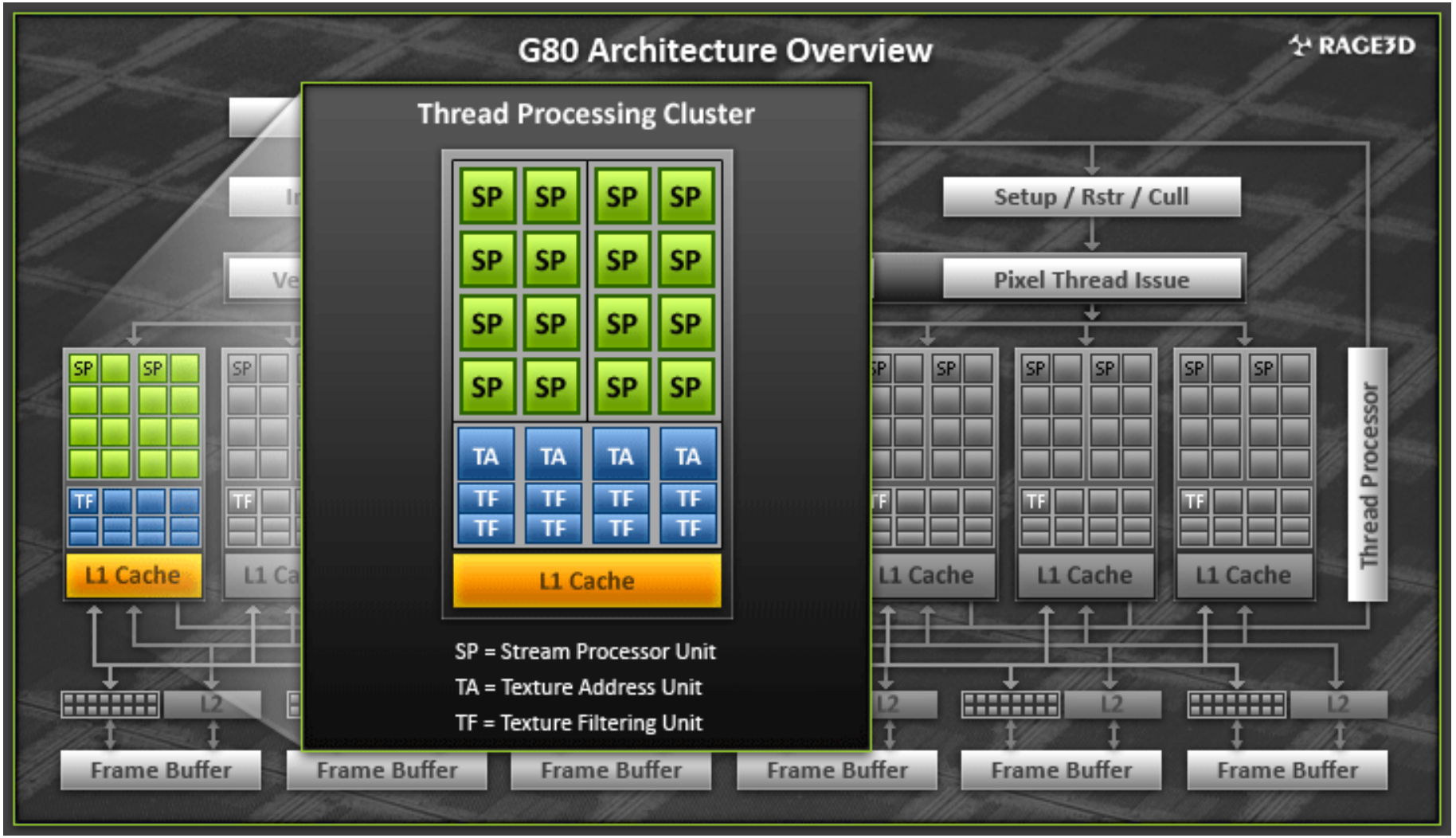


Unified shader Architecture

Case study: NVIDIA GeForce 8800

- NVIDIA GeForce 8800 (2006)
- 128 CUDA core in 8 stream processors (shader cluster)
- 24 fragment pipeline(for z-test and color blend)

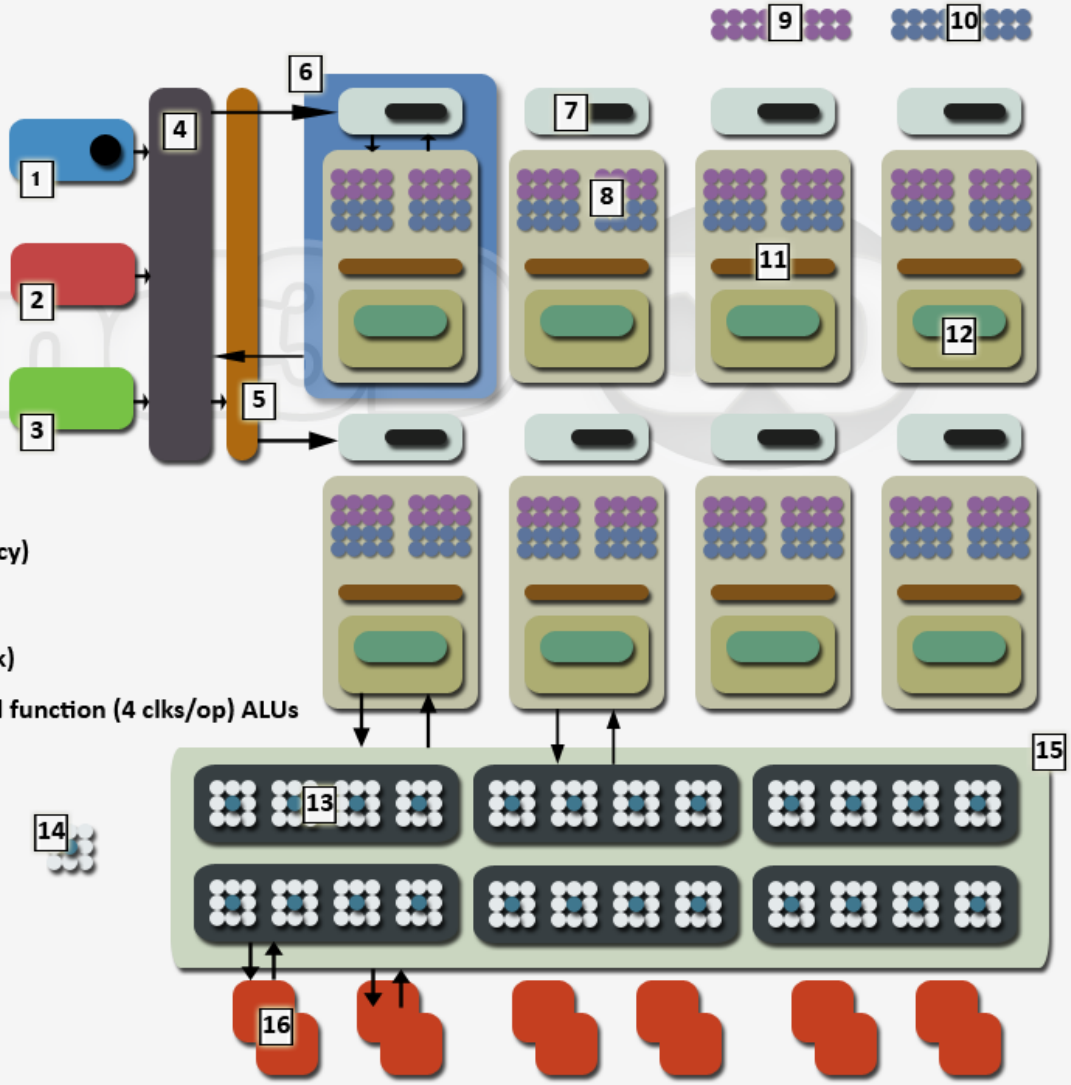
Unified shader Architecture
NVIDIA GeForce 8800



G80

nb: not all datapaths are shown, and those that are exist for illustration only

graphic revision 1.2 - 4th January 2007



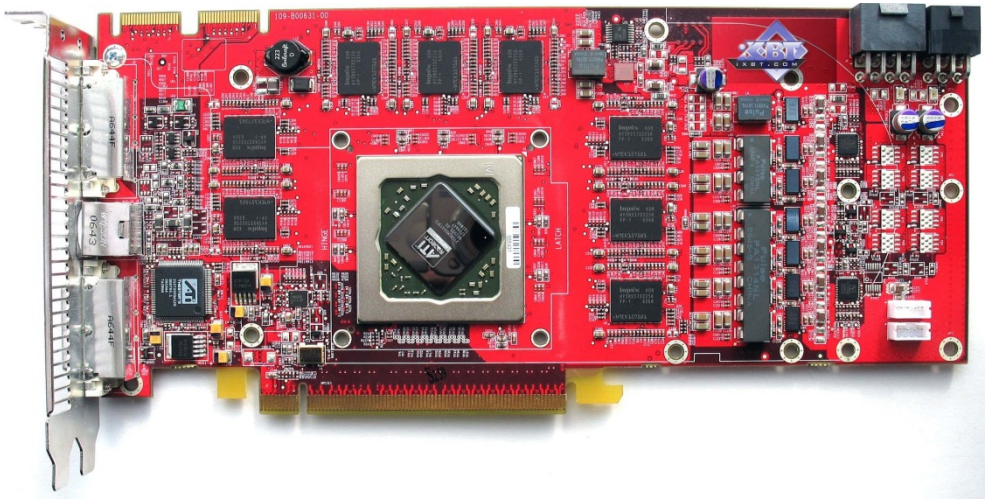
1. Vertex thread setup and input assembler
2. Geometry thread setup
3. Pixel thread setup
4. Global thread scheduler
5. Triangle setup (1 tri/clk), rasterisation and Z-cull
6. Thread processing cluster
7. Per cluster scheduler and register file (half shader frequency)
8. SP and interpolator/special ALU groups
9. 2 x 8-way scalar FP32 SP ALUs (MADD + MUL dual-issue/clk)
10. 2 x 8-way FP32 scalar interpolator (1 attrib/clk) and special function (4 clks/op) ALUs
11. 4 pixels/clk data address and setup
12. 8 INT8 bilerps/clk filtering + L1 local store (8KiB)
13. ROP partition
14. ROP with 8Z or 8C samples/clk, 2clk FP16 blend
AA: 0x 8Z/clk, 4x 1Z+C/clk; max 8xMSAA, 16xCSAA
15. L2 shared data store (128KiB)
16. DRAM pair (2 x 32-bit)

base clock domain, memory clock domain, shader clock domain

Unified shader Architecture

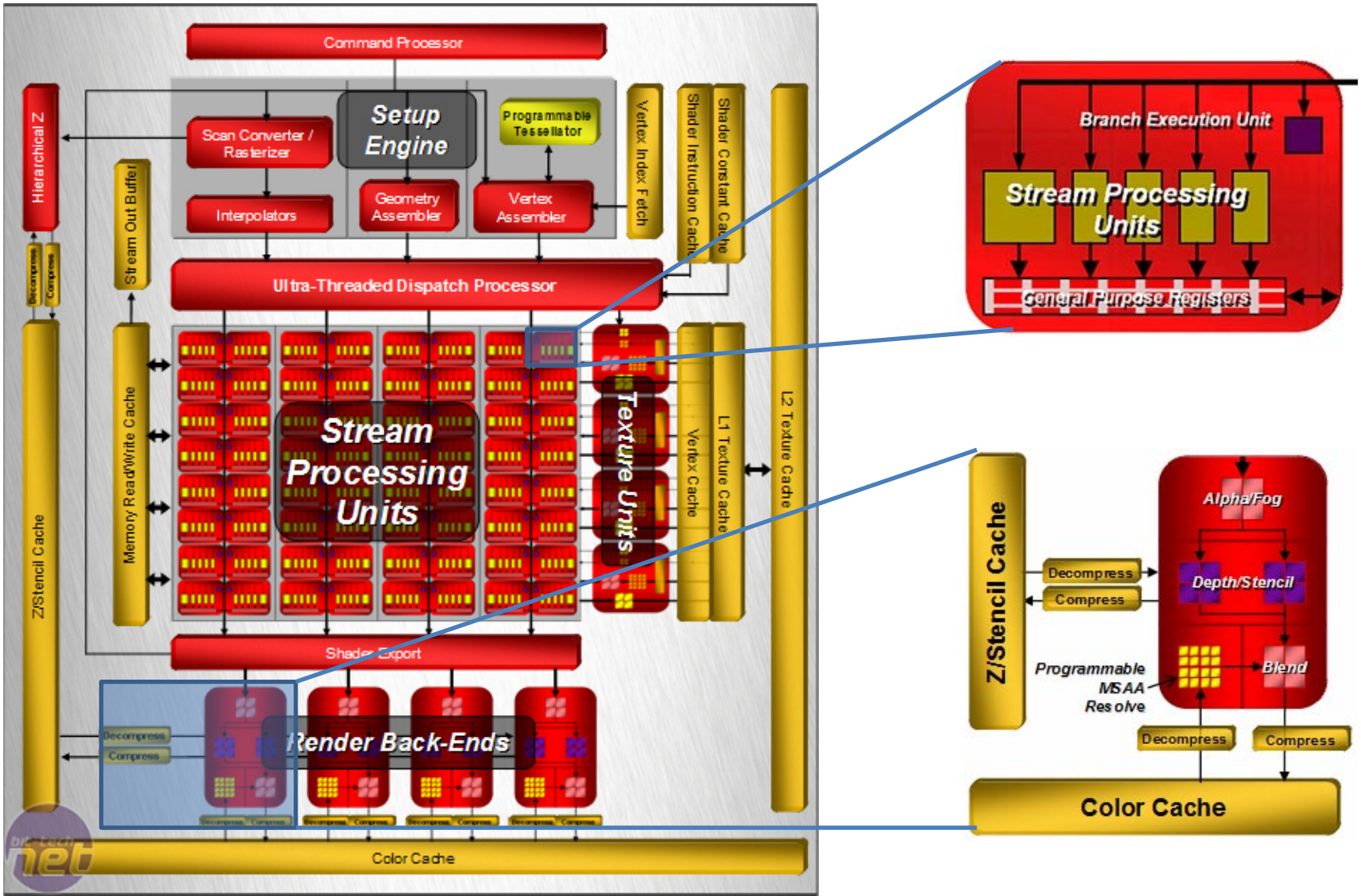
Case study: AMD ATI HD 2900

- AMD ATI HD 2900XT (2006)
- 64 unified shader (320 stream processor)
  - VLIW architecture, 5 operation one cycle.
- 4 Render Back-End (For Z-test and color blend)



Unified shader Architecture

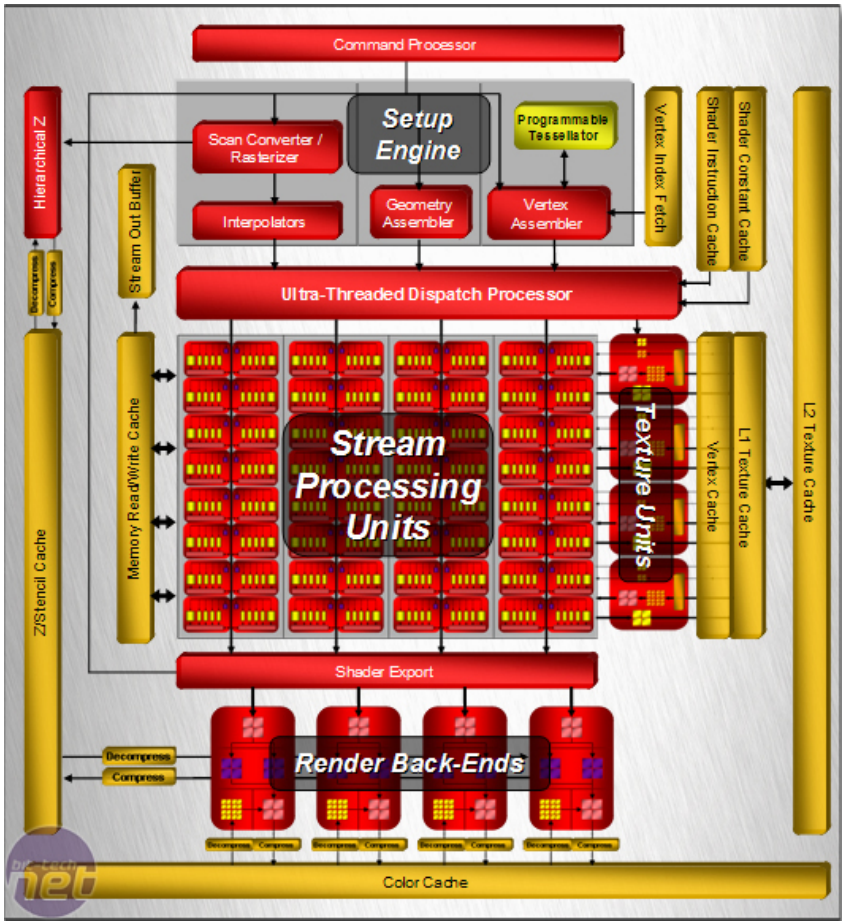
AMD ATI HD 2900XT

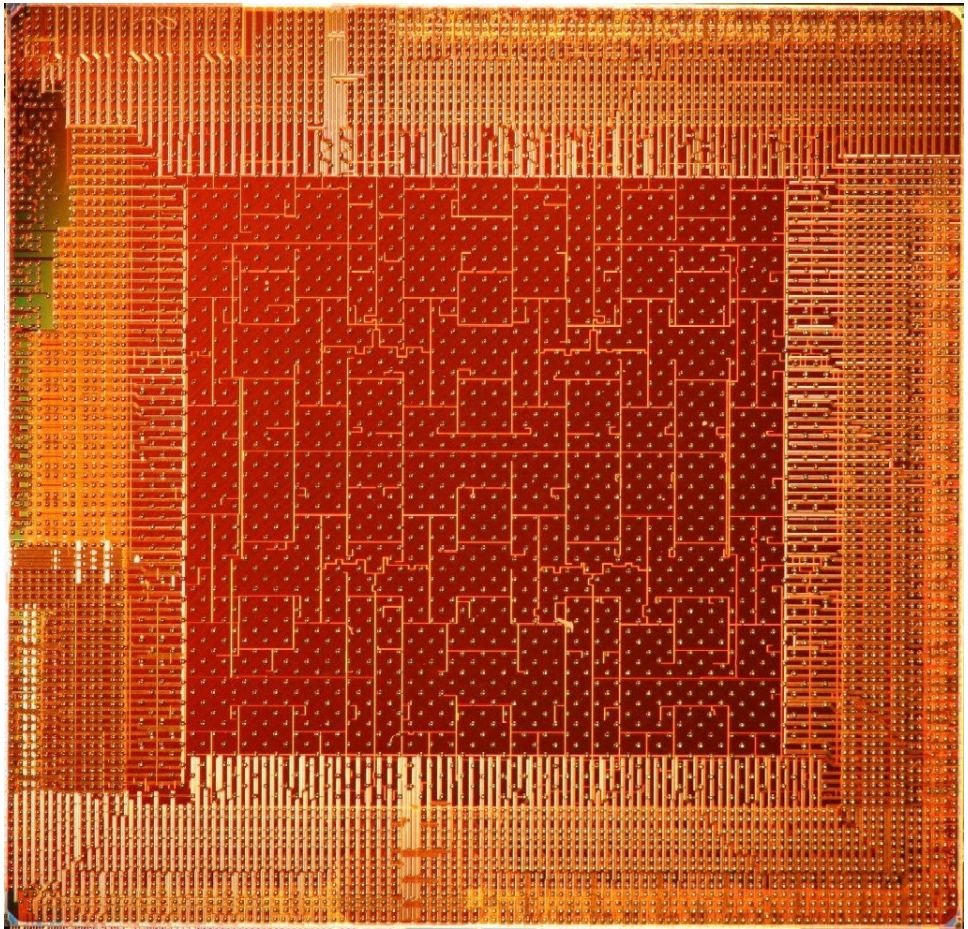


AMD ATI HD 2900XT

Placement & Layout

- Fixed hardware : shader = 4 : 6 (maybe)
  - Not 0 : 1





A unified shader comparison in 2010

NVIDIA GeForce GTX 480

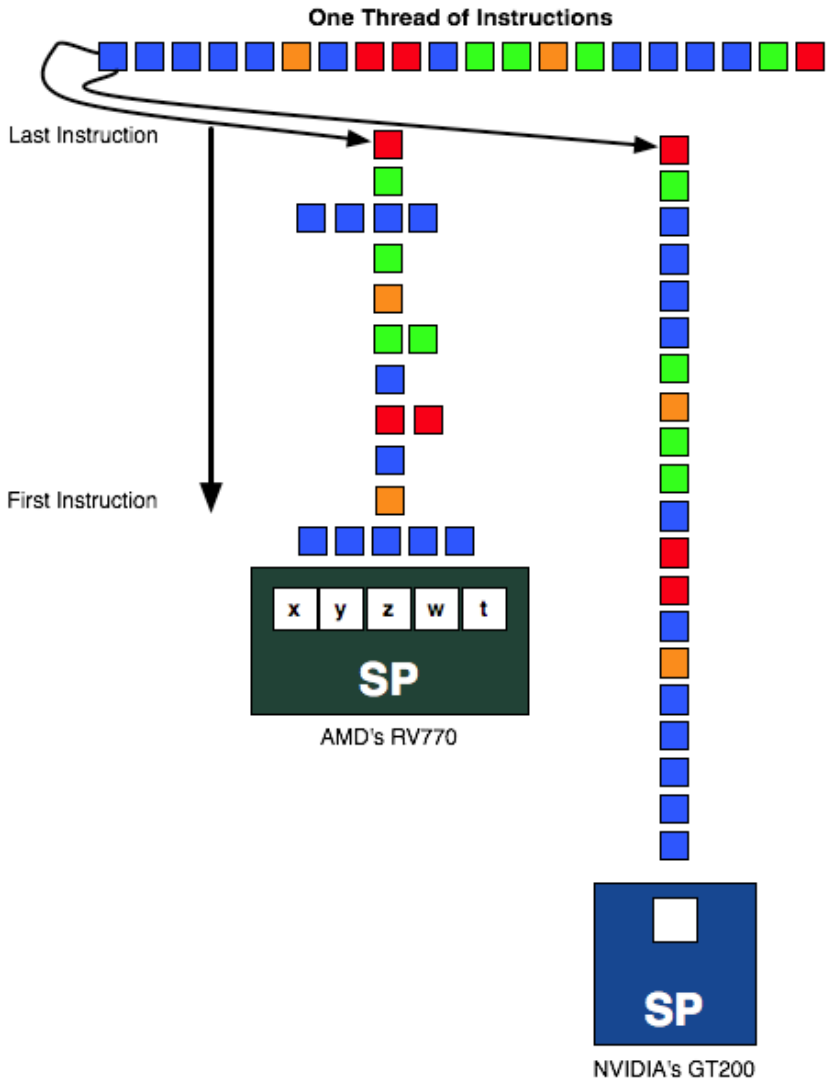
- 480 cores (128 in 8800)
- **177.4 GB/s** memory bandwidth
- 1.34 TFLOPS single precision
- **3 billion** transistors

ATI Radeon HD 5870

- **1600** cores (320 in 2900)
- 153.6 GB/s memory bandwidth
- **2.72 TFLOPS** single precision
- 2.15 billion transistors

Over double the FLOPS for less transistors! What is going on here?

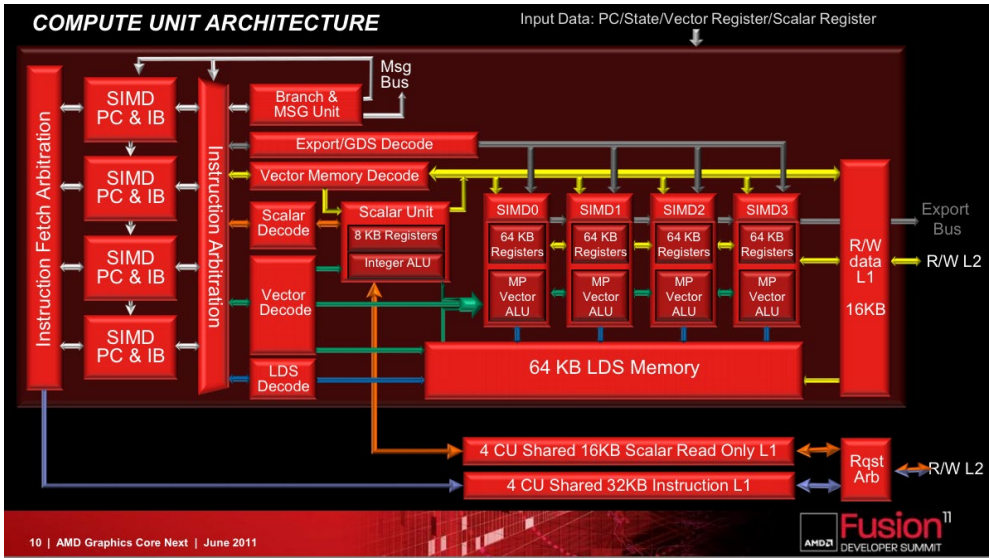
Compared stream processor usage AMD vs NVDIA(Fermi vs rv770)



Unified shader Architecture

AMD 7970 (2012)

- VLIW architecture is good for existed application, but bad for the unknown/future application.
  - VLIW's compiler ability is limited.
- From VLIW to SIMD

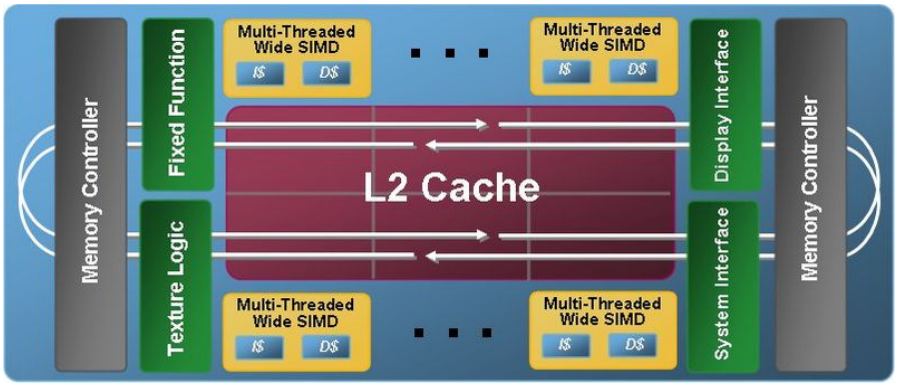


Unified shader Architecture

Case study: Intel Larrabee

- 32x simplified Pentium CPU.

  - No out-of-order execution.
  - Compatible with X86-based program.
- Sort-middle architecture



Unified shader Architecture
Intel Larrabee

- Announce in 2008, shutdown in 2010……
  - Due to the performance issue.
  - The research result become part of Intel MIC

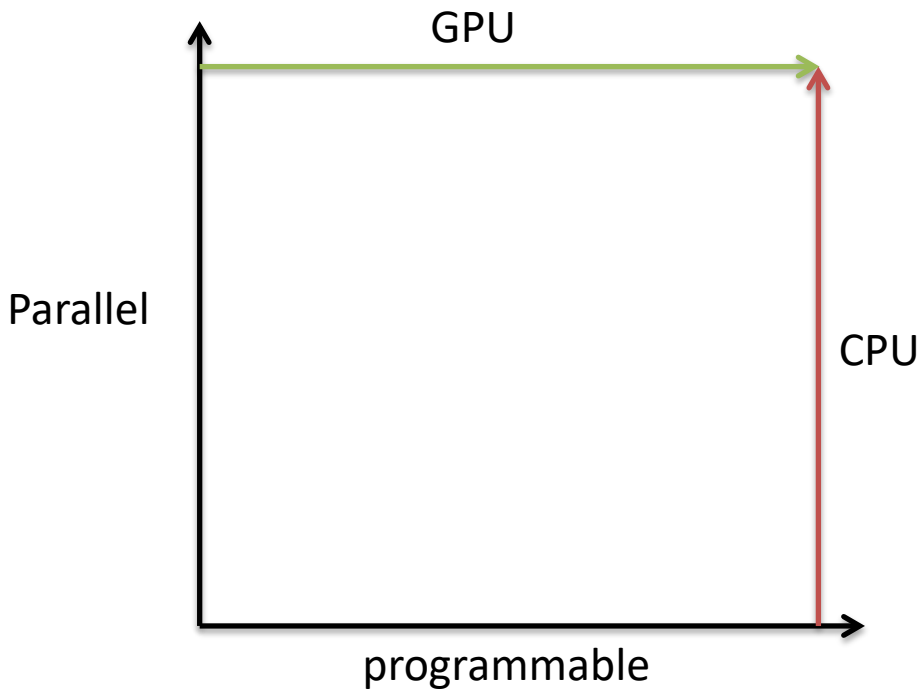
Outline

- GPU Architecture history
- Taxonomies for Parallel Rendering
  - Tile-based rendering
- Fixed function pipeline
- Separated shader architecture
- Unified shader architecture
- Conclusion

GPU architecture issue

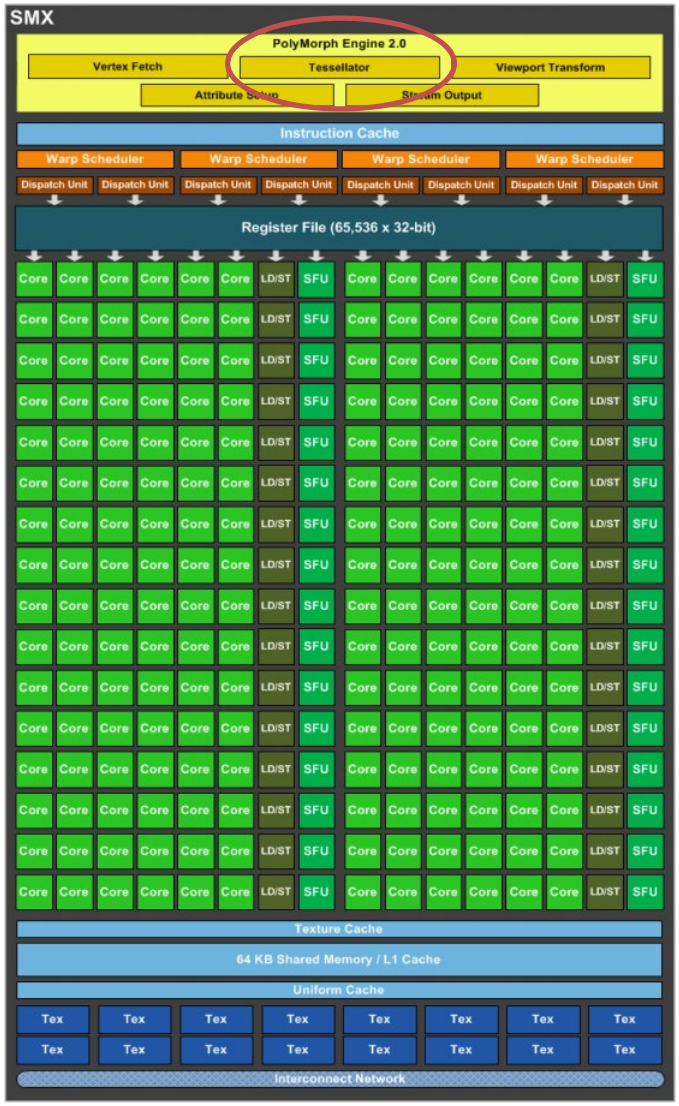
- Where should sort happen?
  - What is the purpose for Job re-distribution?
- Hide memory latency, get more memory bandwidth.
- Cull the hidden element as early as possible
  - Object, triangle, pixel
- Programmable vs fixed ?
  - Reality vs ideal.

Trend



Programmable vs fixed ?

- Because of the Performance issue, tessellation become fixed hardware
  - GeForce 580 -> 680
  - DirectX 10 -> 11



Future lead way

- Application lead hardware
  - Ray-tracing
- Hardware limit application
  - For the money issue, more and more 3D game companies prefer to stay in Xbox360/PS3.
    - Since 2007, the increasing rate of Image quality in 3D game has been slow down.

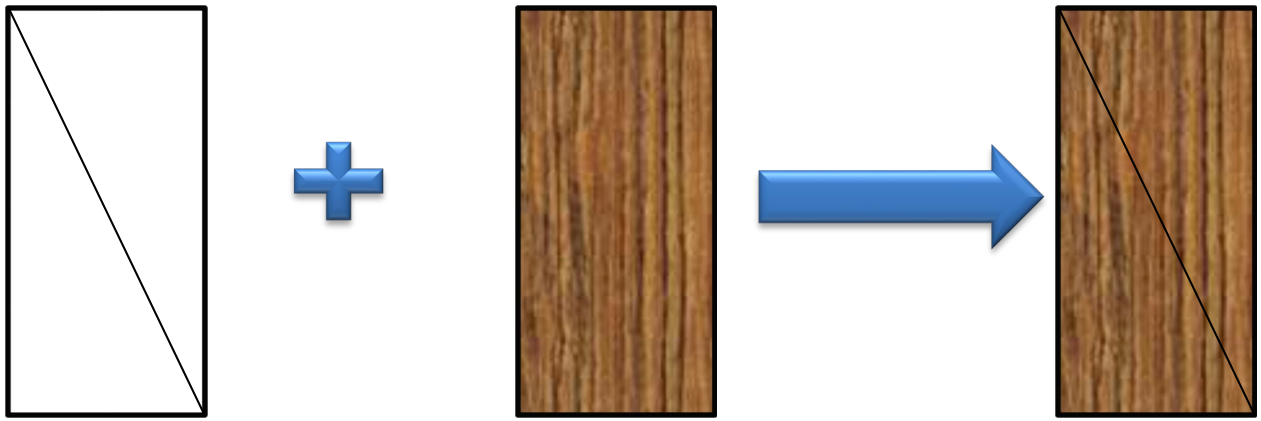
Any Question?

You can get slides in

140.116.164.239/~caslab/GPU_Present_NSYSU/

Example: use transparency texture to model a tree with some leaves

- Step 1, draw the trunk

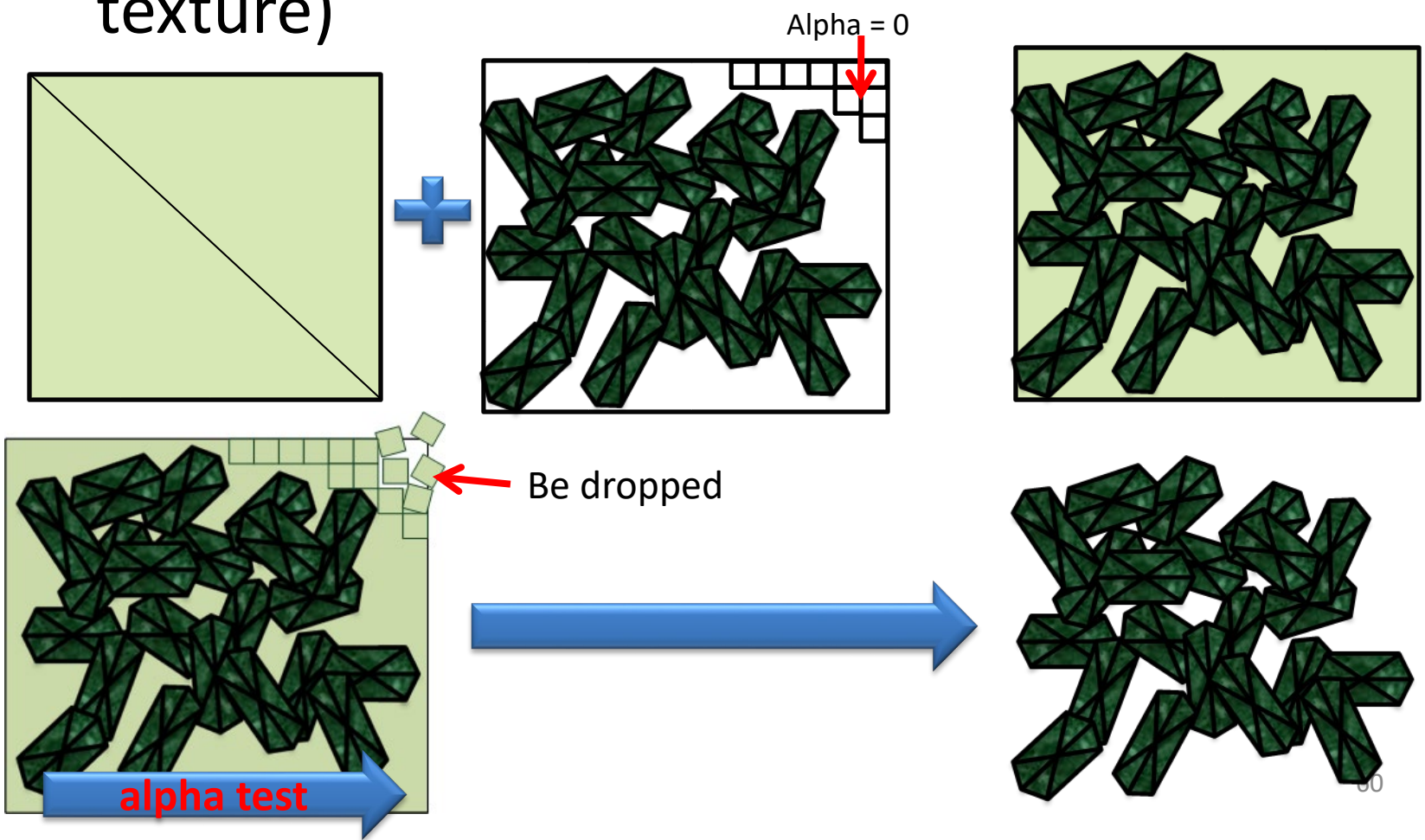


Example: use transparency texture to model a tree with some leaves

- Step 2, draw leaves(use lots of triangles)

Example: use transparency texture to model a tree with some leaves

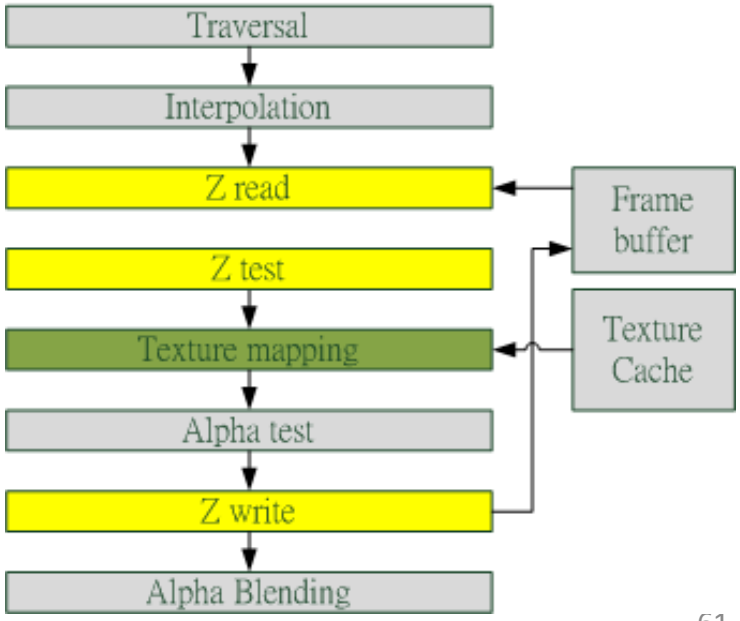
- Step 2, draw leaves(use transparency texture)



Early depth test

- Because of early depth test, the fragments which shall be dropped by alpha test update the depth buffer now.

- So we separate Z-write and Z-test ,and put Z-write behind the alpha test.



Early depth test

- But separating z-test and z-write will cause data hazard problem.
- Using multi-Z test to perform depth test twice and avoid data hazard.