Neon: A Big, Fast, 3D Workstation Graphics Accelerator

August 17, 1998

Joel McCormack, Bob McNamara, Chris Gianos,
Larry Seiler, Norm Jouppi & Ken Correll

COMPAQ

Outline

- ❑ What is Neon?
- ❑ Goals & design philosophy
- ❑ Some “whys” answered (Proceedings only)
- ❑ Block diagram and block responsibilities
- ❑ What makes Neon tick?
- ❑ Examples of increasing usable memory bandwidth (Proceedings only)
- ❑ Competitive positioning
- ❑ Conclusions

What Is Neon?

- ❑ A graphics rendering instruction set
  - ■ 2D rendering: X11, Windows/NT
  - ■ 3D rendering: OpenGL, Direct3D
  - ■ Video rendering: DirectDraw
- ❑ A hardware architecture
- ❑ A “mid-range” graphics accelerator chip

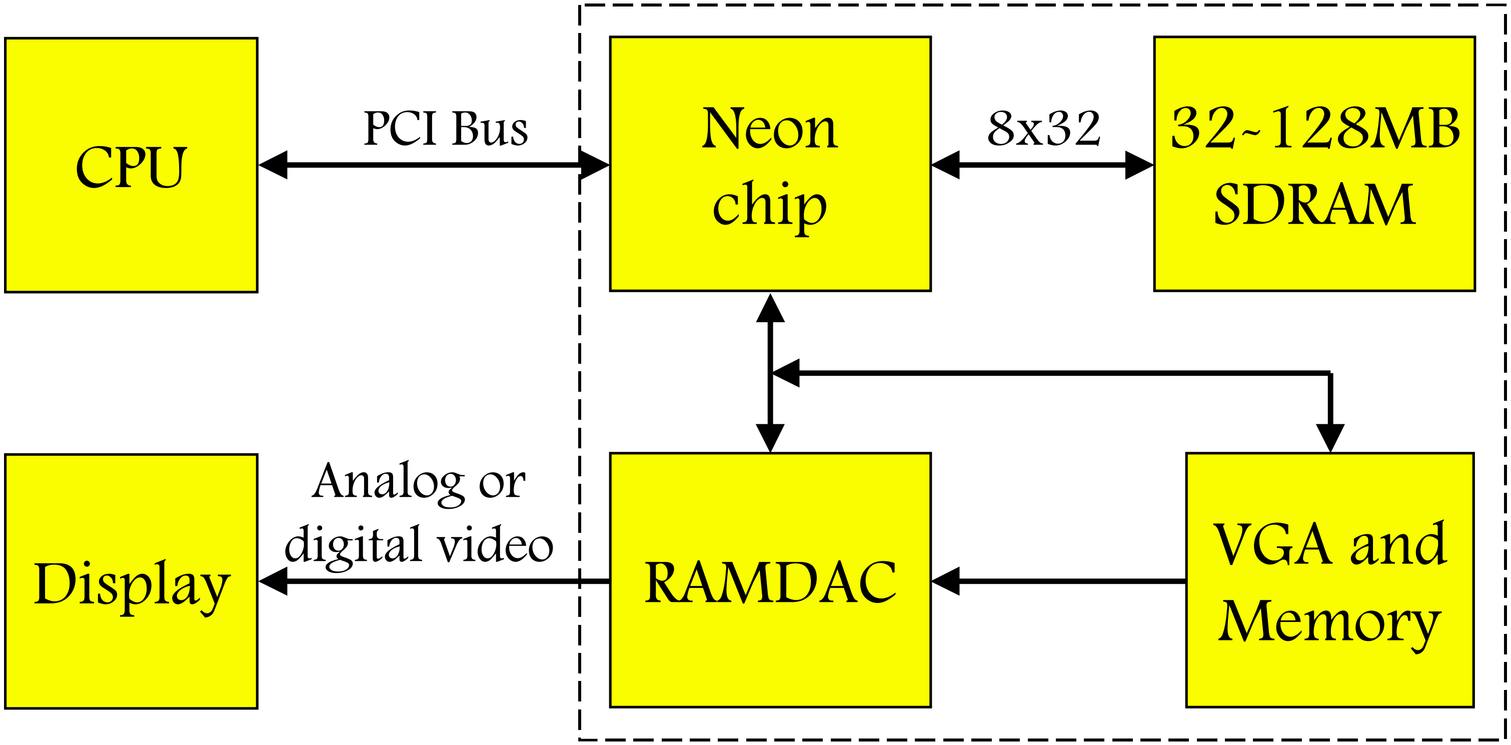
Goals

- ❑ One architecture spans multiple implementations
- ❑ First implementation fits on a single chip
- ❑ Cost 1/4 to 1/2 of boards with similar performance
- ❑ Performance 2x to 4x of boards with similar cost
- ❑ Paint 50-pixel triangles faster than 600 MHz Alpha 21264 can spit them out
  - Um, wanna make that 25-pixel triangles?
  - Okay, okay, 10-pixel triangles?

COMPAQ

Neon in a System

- Simple single-slot, 32/64-bit PCI card



Design Philosophy

- Single chip design
  - Lower cost through integration and logic sharing
  - High-bandwidth on-chip interconnect, no pin wars
- Bottom-up design, starting from memory system
  - Memory system provides high usable bandwidth
  - Design the rest of the chip to keep memory busy
- Tightly integrate software and hardware design
  - Keep testing hardware features against software needs
  - Simulate performance at all levels of system (oops :(

Why Doesn’t Hardware Do Geometry?

- Too much extra hardware & software:
  - Special-purpose specially-programmed FP chips
  - “Triangle bus” resynchronization to maintain order
  - Can become a “decelerator” with the next CPU
- Alpha CPU is good enough for Neon
  - Easy to design object setup headroom into Neon chip
  - So faster CPUs mean faster performance in future
  - SMP systems with parallel software may go even faster

What is Unified Memory?

- Single memory array for all kinds of data
- Data stored at each pixel (called on-screen data):
  - 16-32 bit RGB w/Alpha transparency, 1 to 4 buffers
  - 24-bit Z depth buffer for hidden-surface removal
  - 8-bit stencil for various weird operations
  - 8-bit display formatting information
  - 8-bit overlay planes for user interface and annotation
- Textures for realistic surface features
- Off-screen auxiliary buffers

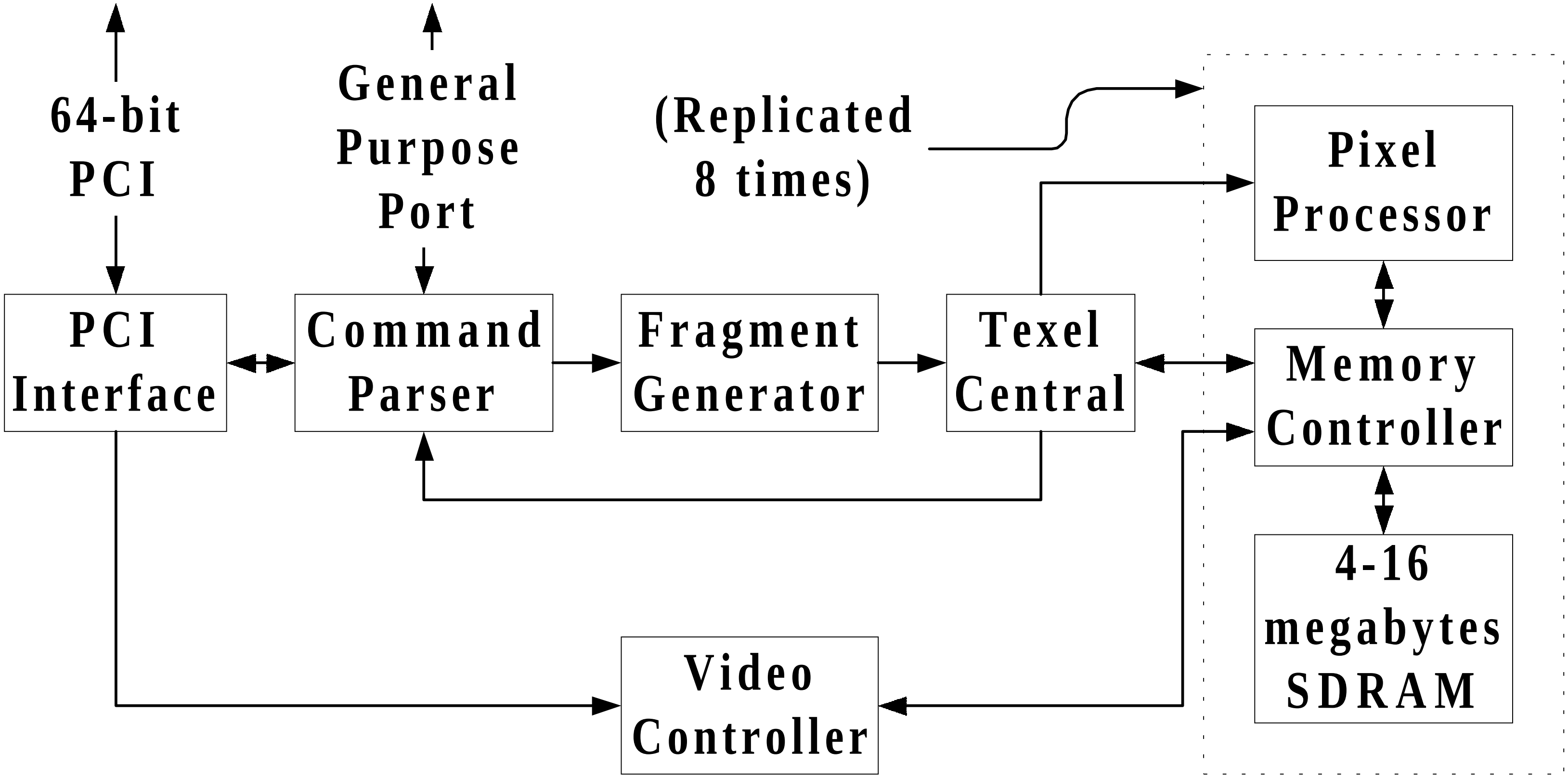
Features of Unified Memory

- ❑ Flexibility in memory bandwidth allocation
  - ■ Entire bandwidth available for fills and copies
  - ■ 2:1 Z/color bandwidth split when depth buffering
  - ■ 8:2:1 texture/Z/color bandwidth split when texturing
- ❑ Flexibility in memory storage allocation
- ❑ Higher effective bandwidth/pin, fewer pins, fewer memory chips, lower cost
- ❑ Logic sharing due to common datapaths
- ❑ But there is overhead to switch between regions

Why One Chip?

- ❑ No chip to chip communication to worry about
- ❑ No wasteful replication of textures
- ❑ Wider and faster interconnect on chip
- ❑ Flexibility to jostle logic block boundaries
- ❑ Fewer $s
- ❑ Logic sharing

Neon Chip Block Diagram



Block Responsibilities: PCI Interface, Command Parser

- PCI Interface
  - 64-bit, 33 MHz
  - PIO or DMA master
- Command Parser
  - Parse packets from server ring buffer or client buffer
  - “Gracefully” recover from errors in client buffers
  - Convert vertex data to cannonical formats

Block Responsibilities: Fragment Generator

- Fragment Generator
  - 1 textured fragment w/Z and color, or
  - 4 fragments w/Z and color, or
  - 8 fragments w/32-bit solid or stippled color or Z, or
  - 32 8-bit 2D fragments each cycle
  - Generate all fragments within a rectangular "chunk" to improve SDRAM (chunk = page) and texel cache (chunk related to cache size) behavior
  - Clip to scissors rectangle and 4 inclusive/exclusive clipping rectangles

Block Responsibilities: Texel Central, Pixel Processors

- ❑ Texel Central
  - Lookup and trilinearly filter texture
  - 3D texture coordinates are perspective correct
  - Expand 2D color information from bits to pixels
  - RGBA->RGBA mapping, color masking (blue screen compositing)
- ❑ 8 Pixel Processors
  - Alpha, stencil, Z tests
  - Per-pixel fog (linear, exponential, exponential^2)
  - RGBA Blending, raster-ops, dithering

Block Responsibilities: Video Refresh, Memory Controllers

- Video Refresh Controller
  - Ask for data from both A and B banks, lets Memory Controllers decide best order to service
  - Delegate overlay interpretation, front/back/left/right buffer fetches to Memory Controllers
  - Inverse dither (digitally filter) 16-bit color data to recapture much of original 32-bit color information
- 8 Memory Controllers
  - Independently service one of five request queues from Pixel Processor, Texel Central, or Video Refresh

What Makes Neon Tick?
The Memory System

- ❑ Unified memory reduces idle memory cycles
- ❑ 3.2 gigabyte/second peak frame buffer bandwidth
- ❑ 8 load-balanced memory controllers
- ❑ Batching amortizes memory bus turnaround and allows deterministic prefetching of SDRAM pages
- ❑ Checkerboarded pages improve prefetching
- ❑ Chunked fragment generation does too
- ❑ And screen refresh pages are usually prefetched

What Makes Neon Tick?
Texture Cache

- Each memory controller has a texel cache (8 32-bit words, fully associative) to reduce bandwidth demands
- Chunking fragment generation minimizes capacity misses, and so reduces bandwidth demands even further

What Makes Neon Tick?
Command Interface

- ❑ High-level interface vertex interface:
  - supports triangle, quadrilateral, and line strips
  - vertex data is never replicated
- ❑ Multiple formats for vertex data, so software can trade CPU cycles for I/O bus cycles
- ❑ Deeply pipelined triangle setup logic
- ❑ Applications can map OpenGL calls directly to Neon commands, without the inefficiencies usually associated with such direct rendering.

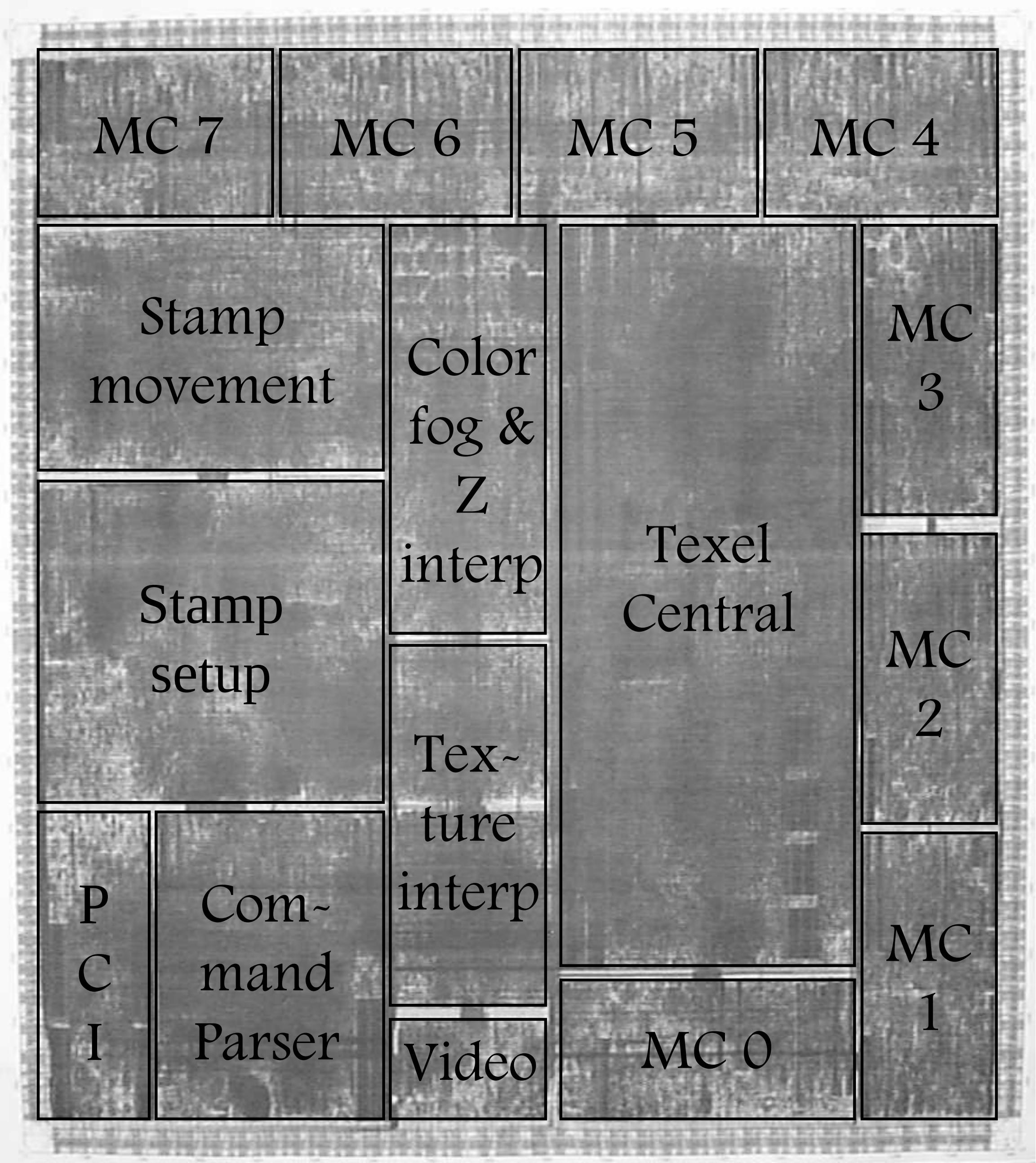
Neon: A 10 BIP Chip

10 nsec clock means:

- ❑ 400 million 64-bit 3D pixels/second generation
- ❑ 3.2 billion 8-bit 2D pixels/second generation
- ❑ 100 million trilinear 3D textured pixels/second
- ❑ 300 million 20-bit divides/second
- ❑ Over 7 billion additions/second
- ❑ Over 3 billion multiplications/second
- ❑ 3.2 gigabytes/second memory bandwidth

Neon: A Big Chip

- ❑ 17.3 x 17.3 mm die in IBM's 0.35 µm 5S
- ❑ 1.7 million IBM cells (about 1.2 million gates, or 6.8 million transistors)
- ❑ 609 signal pins in an 824 pin CCGA


MC 7
MC 6
MC 5
MC 4
Stamp movement
Color fog & Z interp
Texel Central
MC 3
Stamp setup
MC 2
Tex-ture interp
P C I
Com-mand Parser
Video
MC 0
MC 1

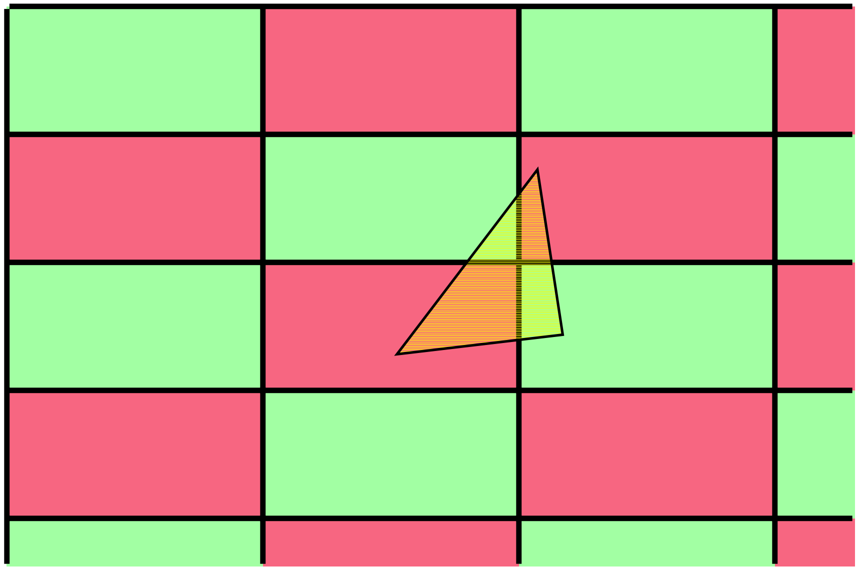
Die Plot

Some Memory Bandwidth Problems

- Page crossings may be expensive
- Need lots of raw bandwidth
- Need efficient use of raw bandwidth
- Even for sick, degenerate triangles that stupid benchmarks generate. (And real programs, too.)
- Read latency, bus turnaround are expensive
- Must handle overlapping objects correctly

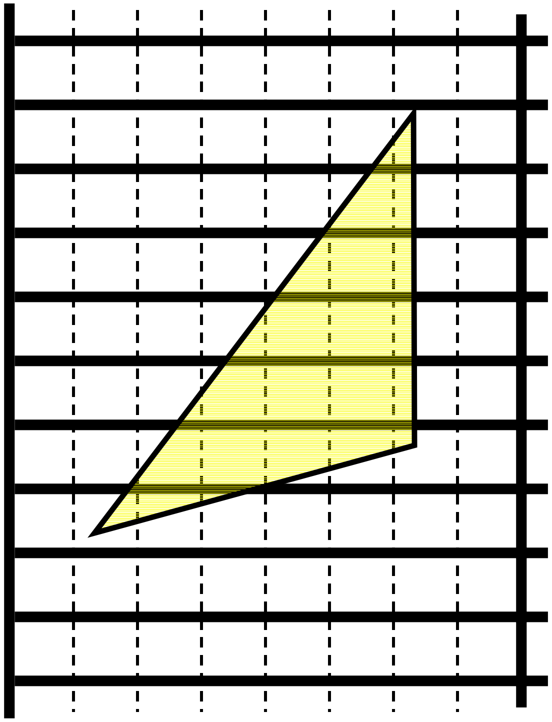
Problem: Page Crossings

- Avoid crossings by tiling pages into rectangles
- Avoid crossings by chunking fragment generation on a page
- Anticipate crossings in queues
- Checkerboard banks to allow parallel prefetch of page in another bank

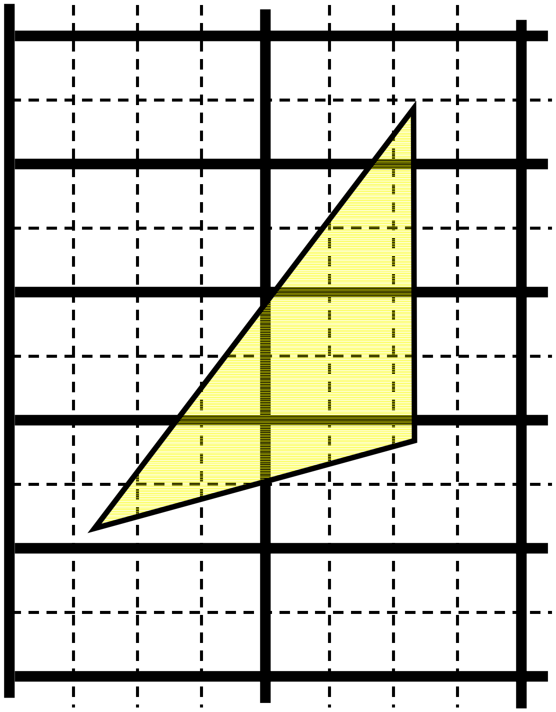


Problem: Raw Memory Bandwidth

- Use 256 bit data path to memory



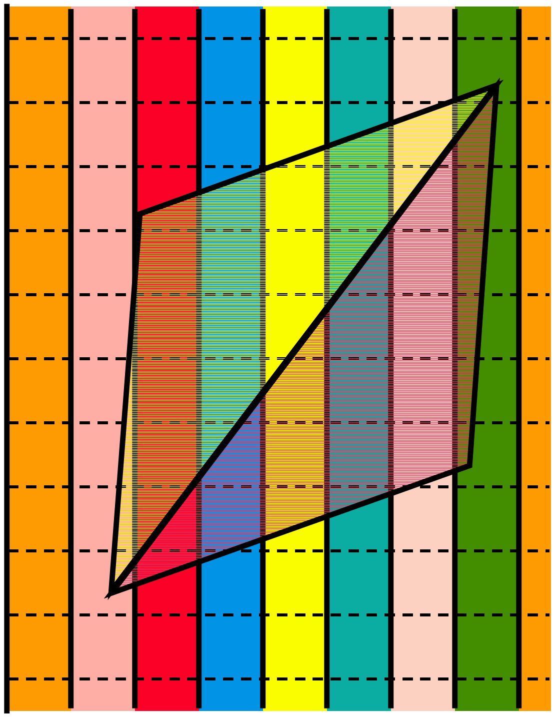
Stupid approach



Better

Problem: Efficient Use of Data Pins

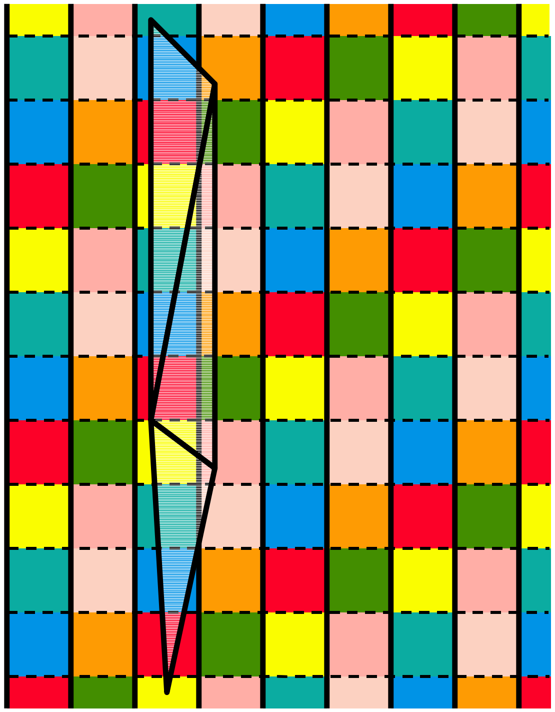
- Most triangles are narrow and do not efficiently use a wide bus (and lines are even worse)
- Use 8 separate 32-bit memory controllers, each with its own address bus to SDRAM
- Different controllers may be working on different triangles, or in different memory regions (e.g. texture or Z/color)



Pretty smart

Problem: Degenerate Triangles

- ❑ Several lines or triangles can fall within a few column of pixels
- ❑ Checkerboard by rotating 2 controllers each scanline to balance workloads
- ❑ Perfect checkerboarding not necessary

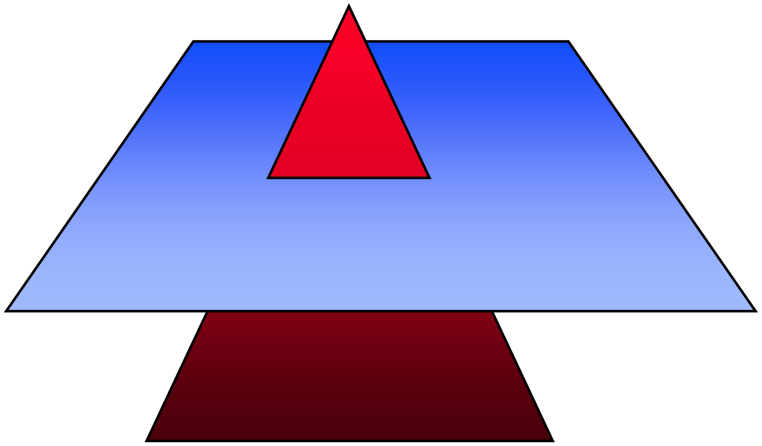


Very smart

COMPAQ

Memory Usage Example: Z Buffering

- Interpolate to find new color and Z values
- Read the old Z value from the frame buffer
- Compare old Z to new Z
- If the pixel is visible:
  - Write new Z
  - Write new color
- Repeat ad naseum



Problem: Bus Latencies

- Batch groups of pixels to amortize bus read and bus turnaround latencies

Bad

Read Z_0, Write Z_0,
Write color_0

Read Z_1, Write Z_1,
Write color_1

Read Z_2, Write Z_2,
Write color_2

Good

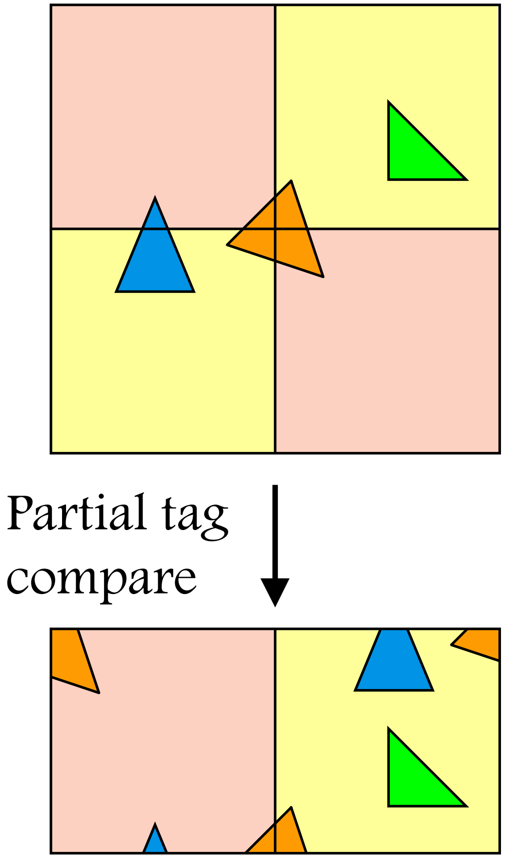
Read Z_0, Z_1, Z_2

Write Z_0, Z_1, Z_2

Write color_0,
color_1, color_2

Problem: Overlapping Objects Create Read/Write Inconsistency

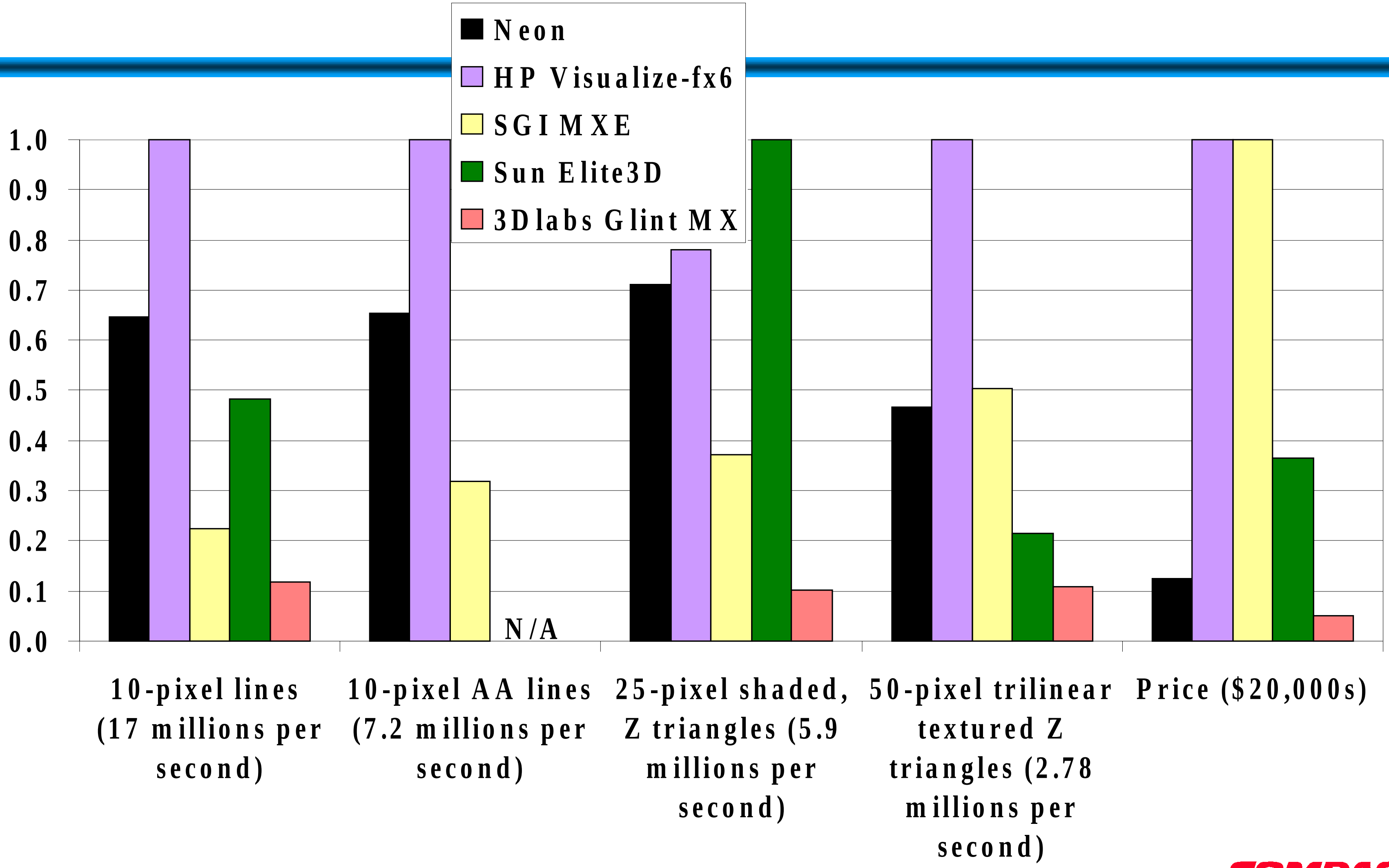
- If fragments overlap, must write first before reading second
- Detect overlaps perfectly within page and across different banks
- Partial tag compare allows false positives in different pages of same bank–but who cares?
- Force write before overlapping read of a pixel for consistency
- If no overlaps, multiple objects batch together



How Effective Are These Techniques?

- 50-pixel Z-buffered triangles:
  - 20% screen refresh (1280x1024 @ 76 Hz)
  - 45% rendering
  - 35% overhead (data pins idle)
- Large fills:
  - 20% screen refresh
  - 60% rendering
  - 20% overhead

Performance Comparisons



Conclusions

- A single ASIC die now allows enough graphics functionality to compete with multiple chips
- A large ASIC pin count and SDRAM allow enough memory bandwidth from a single chip
- Performance and price/performance very good
- Modern ASIC technology (0.18 µm) would shrink die size by 3x, and increase performance at least 80%.

For More Information

- Joel.McCormack@digital.com
- www.research.digital.com/wrl/techreports/pubslist.html
  - *Neon: A (Big) (Fast) Single-chip Workstation Graphics Accelerator*, Research Report 98/1
  - *Prefiltered Antialiased Lines Using Distance Functions*, Research Report 98/2
  - Eventually *Chunking Fragment Generation Using Half-Plane Equations*
- www.research.digital.com/wrl/projects/Graphics/ graphics.html