

Chapter 7

Multicores,
Multiprocessors, and
Clusters

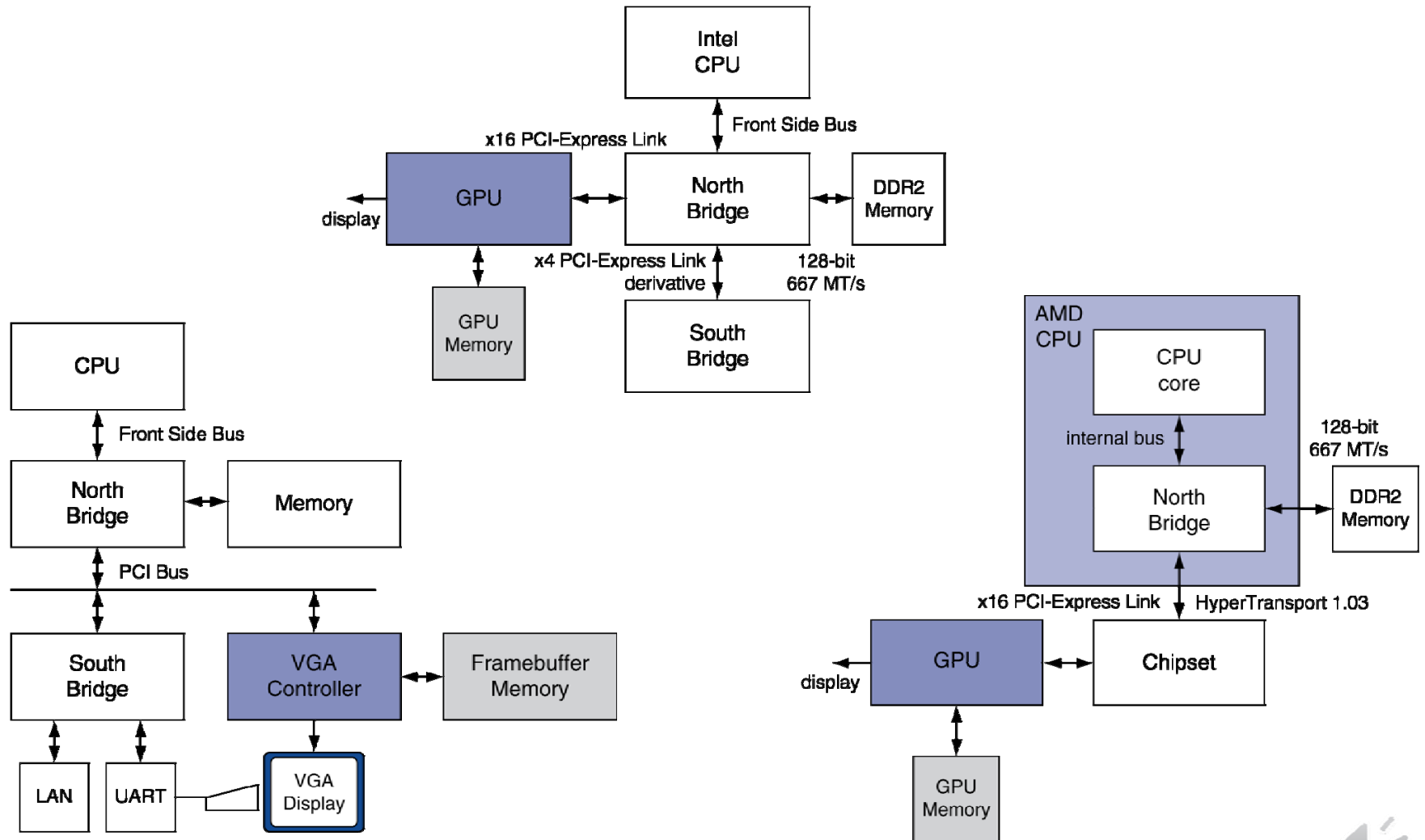


History of GPUs

- Early video cards
 - Frame buffer memory with address generation for video output
- 3D graphics processing
 - Originally high-end computers (e.g., SGI)
 - Moore's Law \Rightarrow lower cost, higher density
 - 3D graphics cards for PCs and game consoles
- Graphics Processing Units
 - Specialized processors oriented to 3D graphics tasks
 - Vertex/pixel processing, shading, texture mapping, rasterization



Graphics in the System

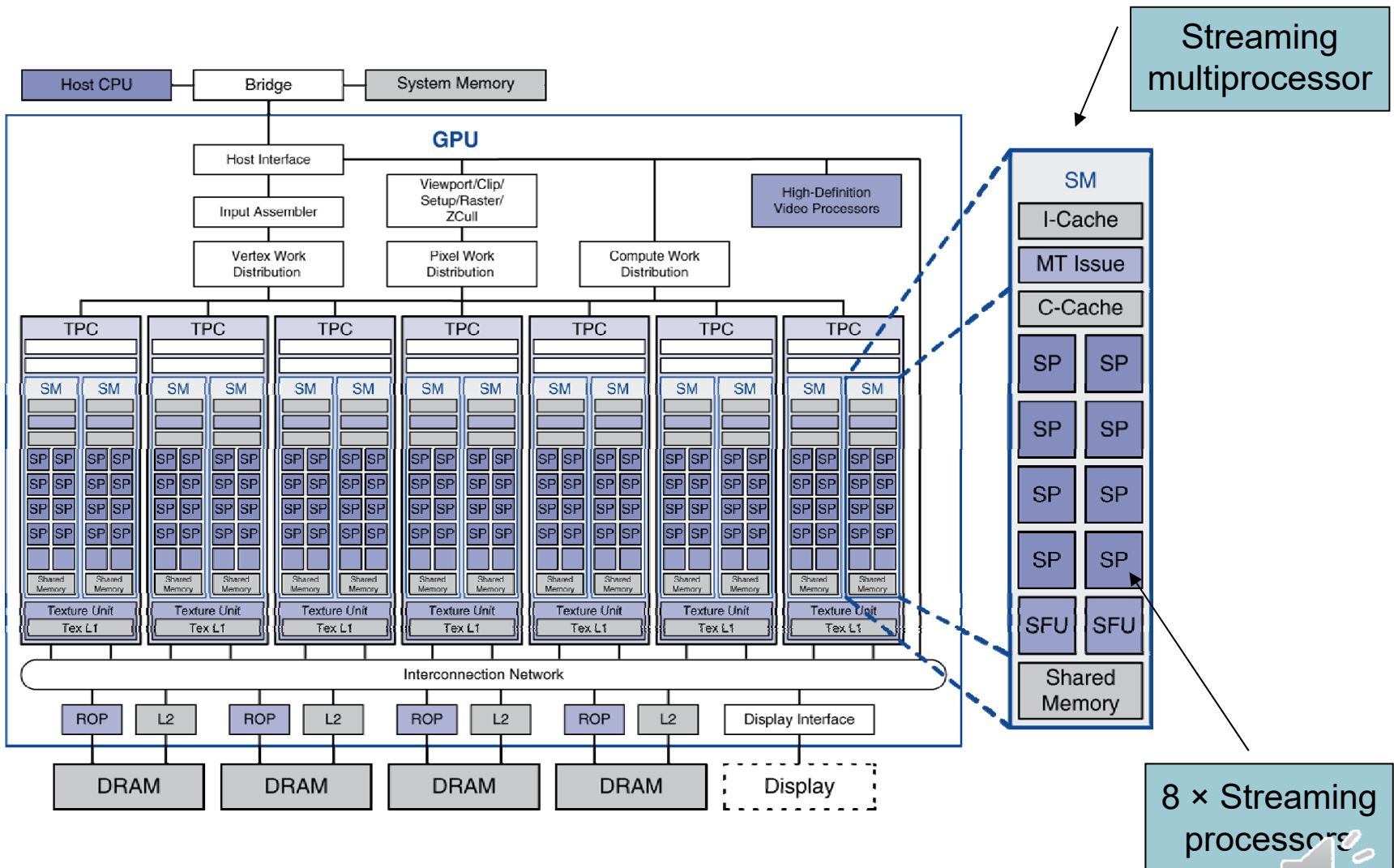


GPU Architectures

- Processing is highly data-parallel
 - GPUs are highly multithreaded
 - Use thread switching to hide memory latency
 - Less reliance on multi-level caches
 - Graphics memory is wide and high-bandwidth
- GPU Trends
 - Heterogeneous CPU/GPU systems
 - CPU for sequential code, GPU for parallel code
- Programming languages/APIs
 - DirectX, OpenGL
 - C for Graphics (Cg), High Level Shader Language (HLSL)
 - Compute Unified Device Architecture (CUDA)

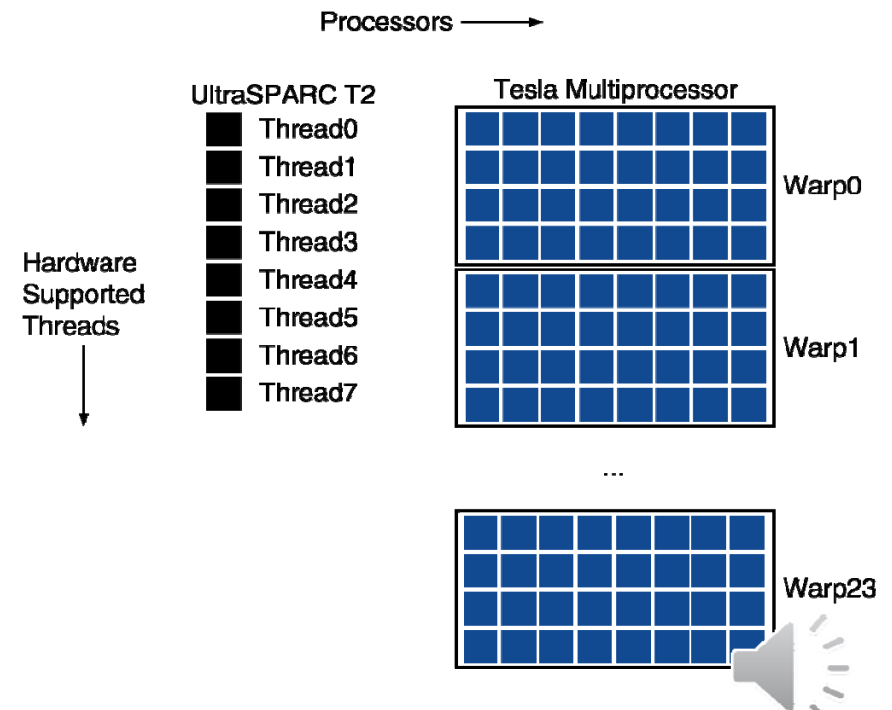


Example: NVIDIA Tesla



Example: NVIDIA Tesla

- Streaming Processors
 - Single-precision FP and integer units
 - Each SP is fine-grained multithreaded
- Warp: group of 32 threads
 - Executed in parallel, SIMD style
 - 8 SPs
 - × 4 clock cycles
 - Hardware contexts for 24 warps
 - Registers, PCs, ...



Classifying GPUs

- Don't fit nicely into SIMD/MIMD model
 - Conditional execution in a thread allows an illusion of MIMD
 - But with performance degradation
 - Need to write general purpose code with care

	Static: Discovered at Compile Time	Dynamic: Discovered at Runtime
Instruction-Level Parallelism	VLIW	Superscalar
Data-Level Parallelism	SIMD or Vector	Tesla Multiprocessor

