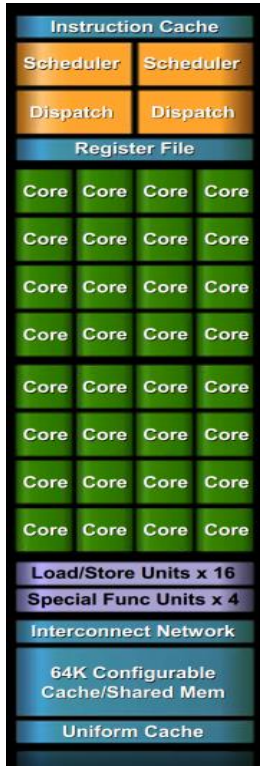
GPU Architecture

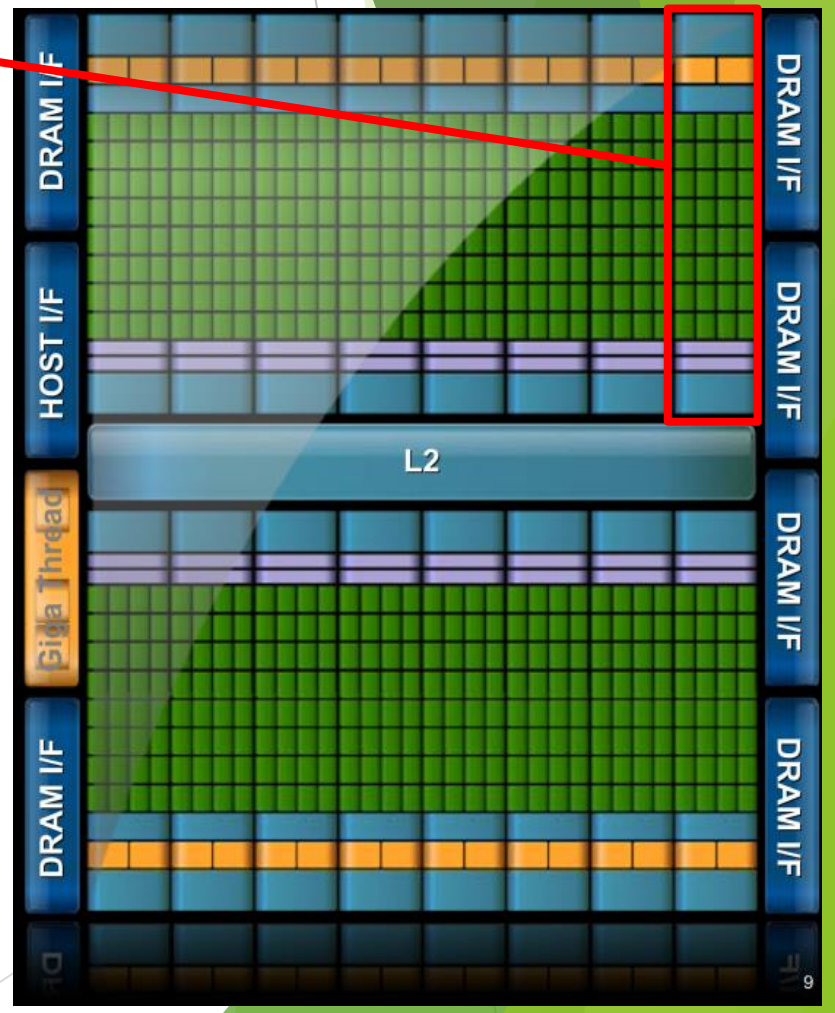
History of GPU Computing

- 1.0: Compute pretending to be graphics (Fixed Function)
- 2.0: Program GPU directly – end of “GPGPU” (Simple Shaders)
  - No graphics-based restrictions
  - 2006: CUDA – general purpose compute language for hybrid GPU systems
  - GPUs became more general purpose and programmable.
- 3.0: GPU computing ecosystem (Today – Graphics Parallel Core)
  - 100,000+ active CUDA developers
  - Libraries, debuggers and many other tools and support
  - Education and research

NVIDIA Fermi Architecture

- Streaming Multiprocessor (SM)
  - Objective: optimize for GPU computing
  - 16 SMs per Fermi chip, 32cores per SM (512 total)
  - 64KB of configurable cache/shared memory
- Memory Hirarchy
  - True cache hierarchy + on-chip shared RAM
  - Separate L1 cache for each SM (16/48KB)
  - Unified L2 Cache for all SMs (768KB)
  - GDDR5 memory interface
- Other Capabilities
  - Hierarchically manages many simultaneously active threads
  - ECC protection for DRAM, L2, L1, RF
  - Unified 40-bit address space for local, shared, global
  - 5-20x faster atomics
  - ISA extensions for C++ (e.g. virtual functions)



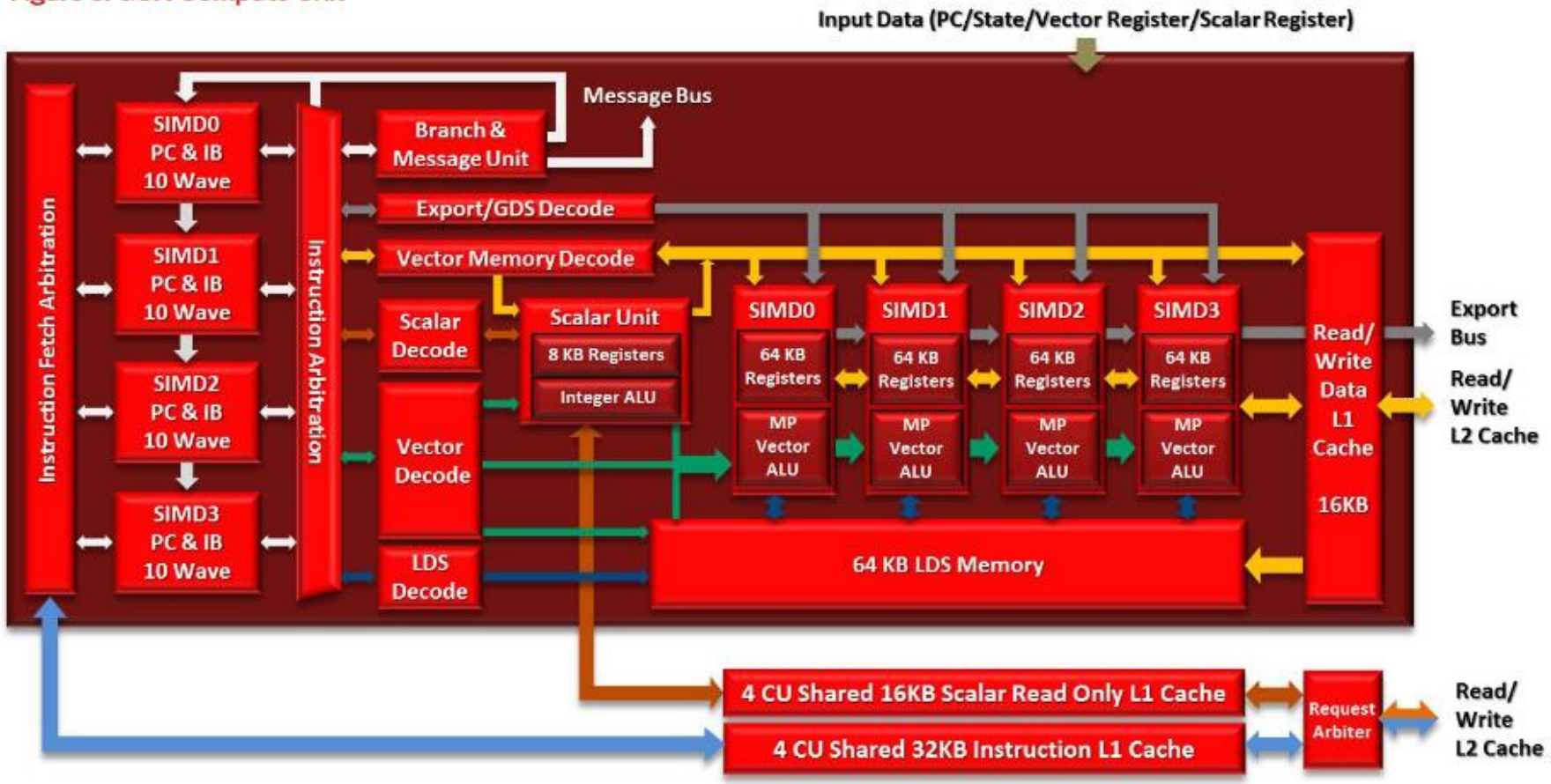


AMD GCN Architecture

- Early AMD GPUs consisted of multiple clusters of 16 ALUs specialized to process VLIW format.
- This led to complex assembly creation and debugging as well as a lack of flexibility.
- The GCN architecture uses 4 SIMDs x 1 ALU op instead to streamline the parallelization. Rather than optimizing inner-waveforms, it optimizes inter-waveforms.
- No longer requires specialized compiler scheduling and assembly as a result.
- Multiple SIMDs coordinated together permits increased vector computing.
- Also upgraded vector register design to have increased bandwidth for data.

The GCN Compute Unit Proper

Figure 3: GCN Compute Unit



Scaling to Exascale…

- Key challenges
  - Energy to solution is too large
    - Fetching operands costs more than computing on them
  - Programming parallel machines is too difficult
  - Programs not scalable to billion-fold parallelism
- Echelon Project (Extreme-scale Computer Hierarchies with Efficient Locality-Optimitized Nodes)
  - Increase in application execution energy efficiency
  - Improve programmer productivity
  - Strong scaling to tens of millions of threads
  - And so on…

The Future of High Performance Computing

- Power constraints dictate extreme energy efficiency
- All future interesting problems will be cast as throughput workloads
- GPUs are evolving to be the general-purpose throughput processors
- CPUs