XMT-GPU

A PRAM Architecture for Graphics Computation

Tom DuBois, Bryant Lee, Yi Wang, Marc Olano and Uzi Vishkin

Bifurcation Between CPU and GPU

- CPUs
  - General purpose, serial
- GPUs
  - Special purpose, parallel
- CPUs are becoming more parallel
  - Dual and quad cores, roadmaps predict many-cores
  - Unclear how to build or program these many-cores
- GPUs more general
  - Gaining momentum for more general apps

Is Unification Possible?

- Can a single general purpose, many-core processor replace a CPU + GPU?
- It must be
  - Standalone – no coprocessor needed
  - Easy and flexible to program
  - Competitive on anything with CPUs
  - Competitive on graphics with GPUs
- We choose a unification candidate (XMT) in part because it satisfies the first 3
- During the Q&A session we welcome your thoughts on what else could be used

Main Experiment and Results

- Can XMT satisfy the 4th?
- Simulate surface shading (a common graphics app) on GP and GPU representatives
- Mixed results
  - XMT slightly faster on some GPU tasks
  - GPUs significantly faster on others
- Unification unlikely, but momentum may shift towards GP

CPU History Overview

- Serial random access machine programming model
  - Great success story, dominant model for decades
  - Popular for both theory and practice
  - Relies on faster serial hardware for performance gains - no longer sufficient
- Multi-cores available (2-4 cores/chip)
- Many-cores on horizon (100’s or 1000’s); but how will they looks?

What Will Future CPUs Look Like?

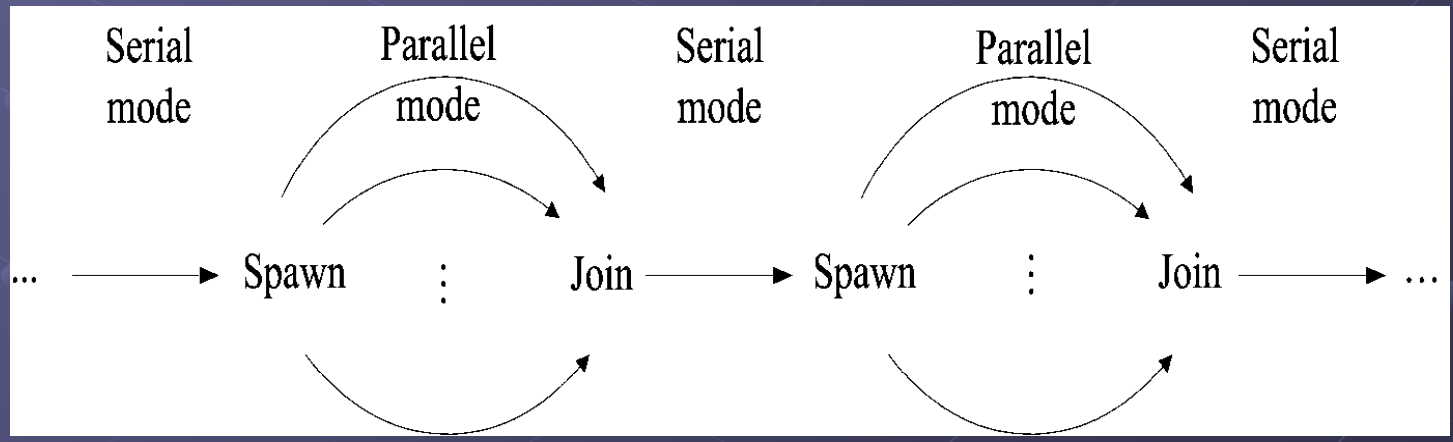
- Future from major vendors unclear
  - Proposals try to look like serial RAM to programmers
  - Long term software spiral broken
- PRAM (parallel random access machine)
  - Model preferred by programming community
    - natural extension of serial
    - scalable
    - included in major algorithm textbooks
  - Discounted because of difficulty building one
  - Recently building one has become feasible

XMT: eXplicit Multi-Threading

- PRAM-on-chip vision under development at the University of Maryland since 1997
  - Targeting ~1000 cores on chip
  - PRAM-like programmability
  - On chip shared L1 cache provides memory bandwidth necessary for PRAM
  - Previous work has established XMT's performance on a variety of applications
  - Simulator and FPGA implementations available
  - www.umiacs.umd.edu/users/vishkin/XMT

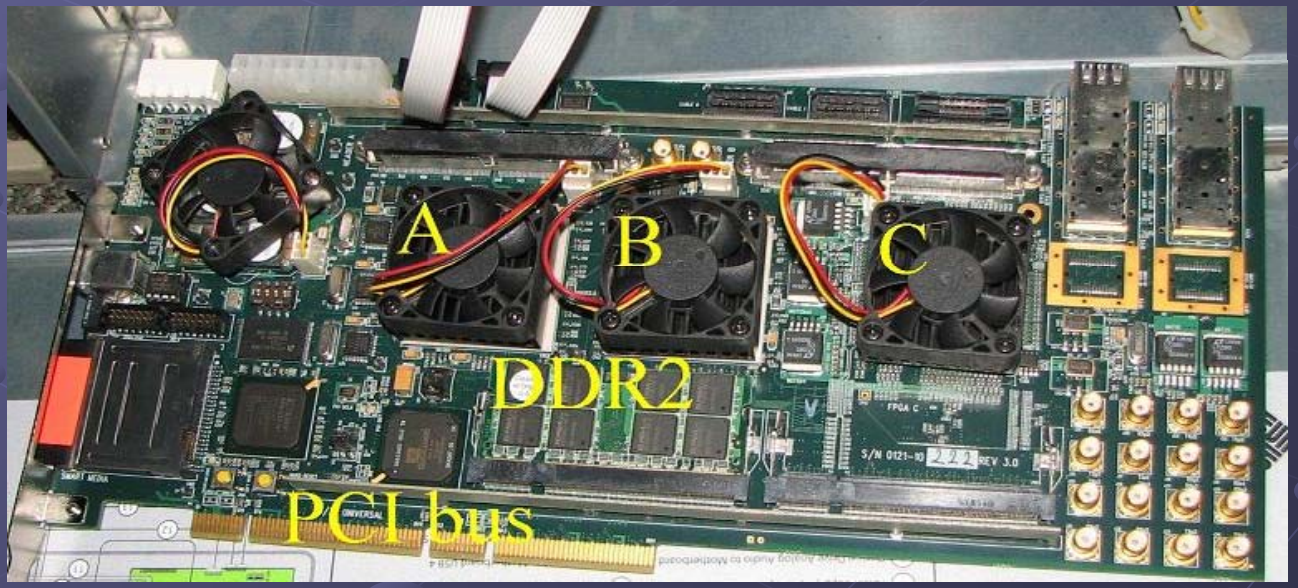
Programming XMT

- XMTC: Single-program multiple-data (SPMD) extension of standard C which resembles CRCW PRAM
- Spawning creates lightweight, asynchronous threads, serial execution resumes once all threads complete



XMT FPGA In Use

- 64 Processor, 75MHz prototype
- Used in undergrad theory class (also: non-major Freshmen and 35 high-school students)
  - 6 significant projects
  - No architecture discussion, minimal XMTC discussion

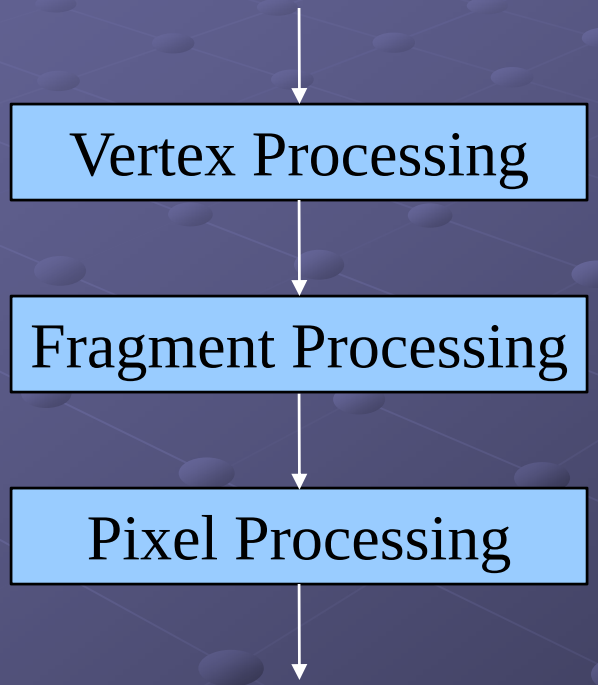


GPU History Overview

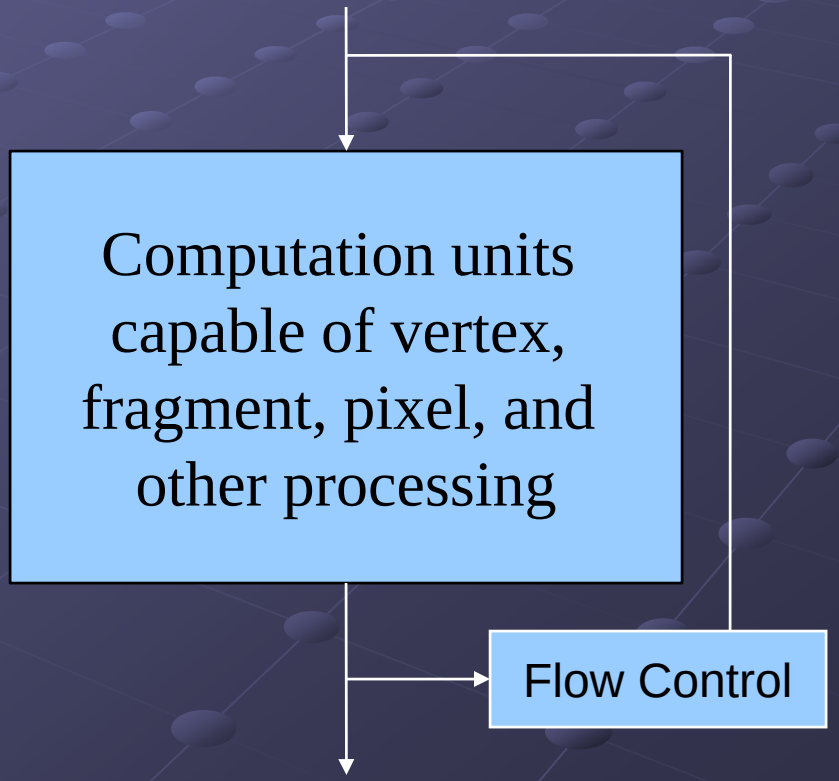
- Stream programming model
  - Streams and kernels: simple and easily exploits locality for some tasks
  - Handles irregular, fine-grained, and serial code poorly
- Originally very inflexible
  - “Programming” meant setting bits for Muxes
- Modern GPUs are much more flexible
  - C like languages
  - GPGPU
  - Still tied to stream model

Very High Level GPU Pipeline

- Old pipelined architecture

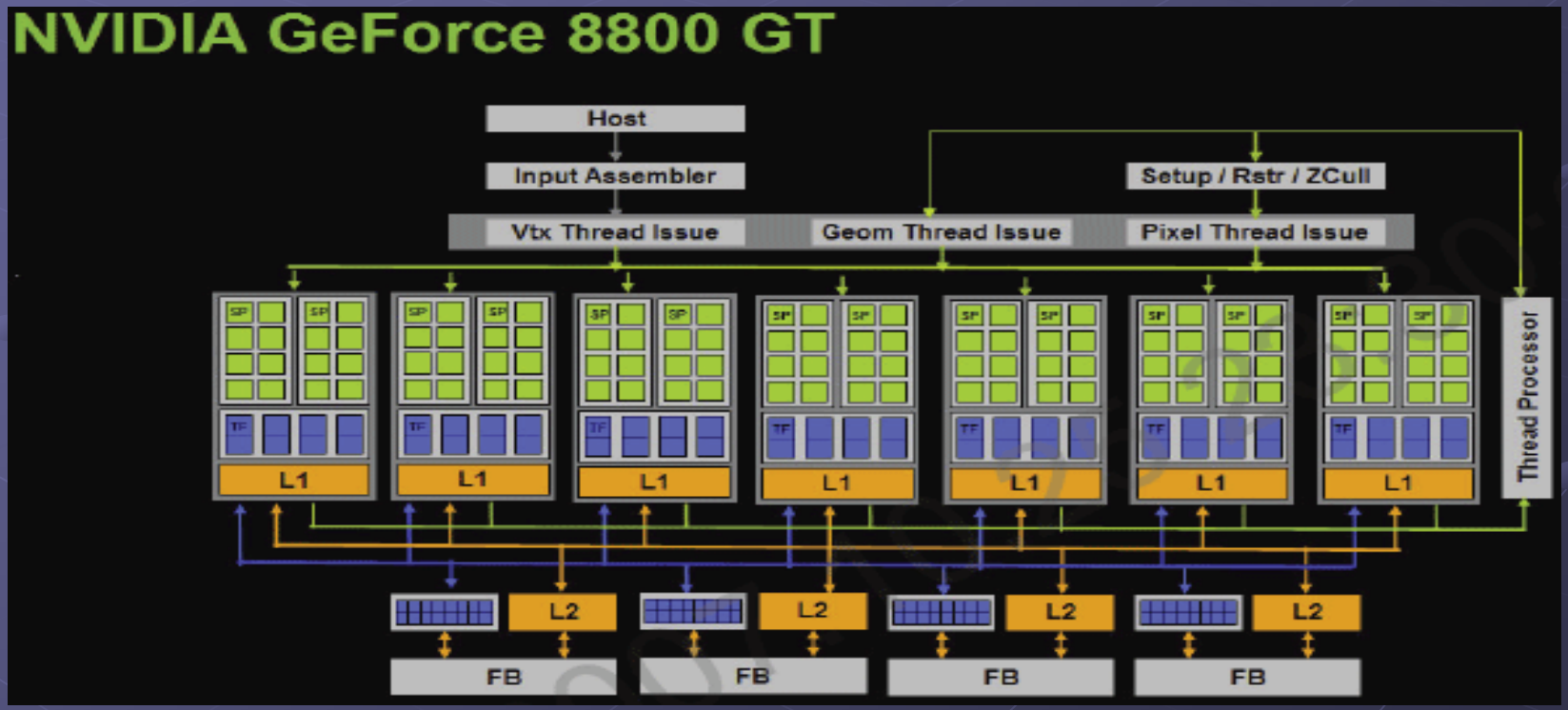


- New virtual pipeline architecture

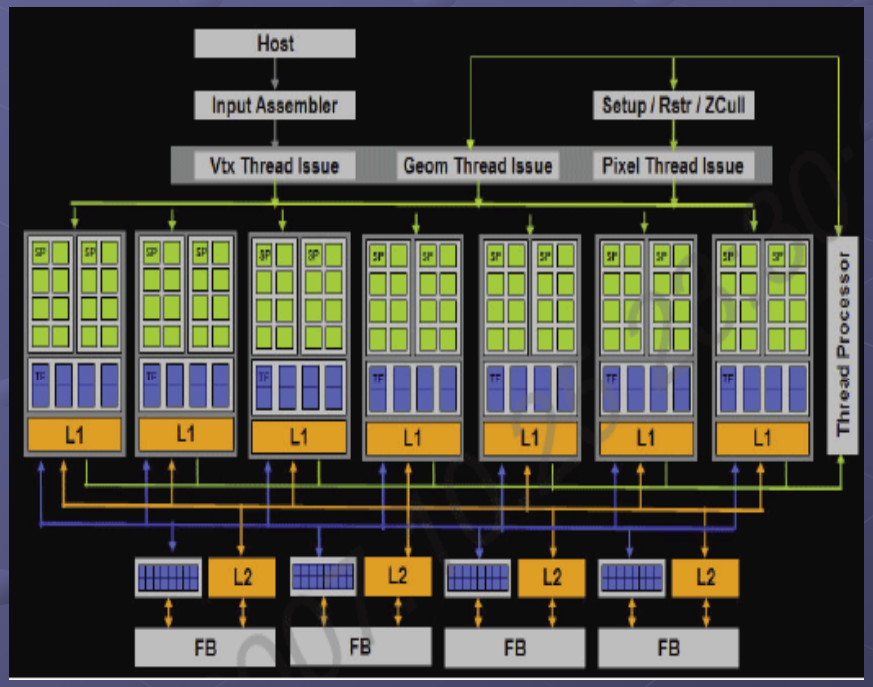


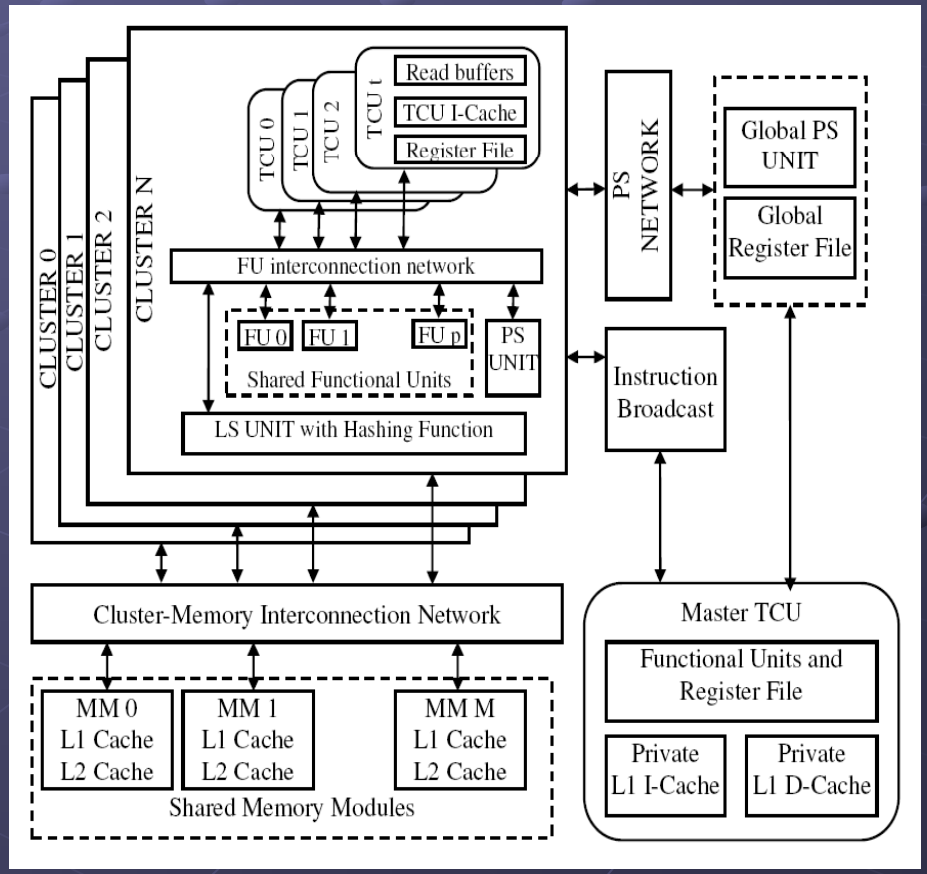
More Detailed Modern GPU

- From NVIDIA GeForce 8800 GPU Architecture Overview
  - Virtual pipelined



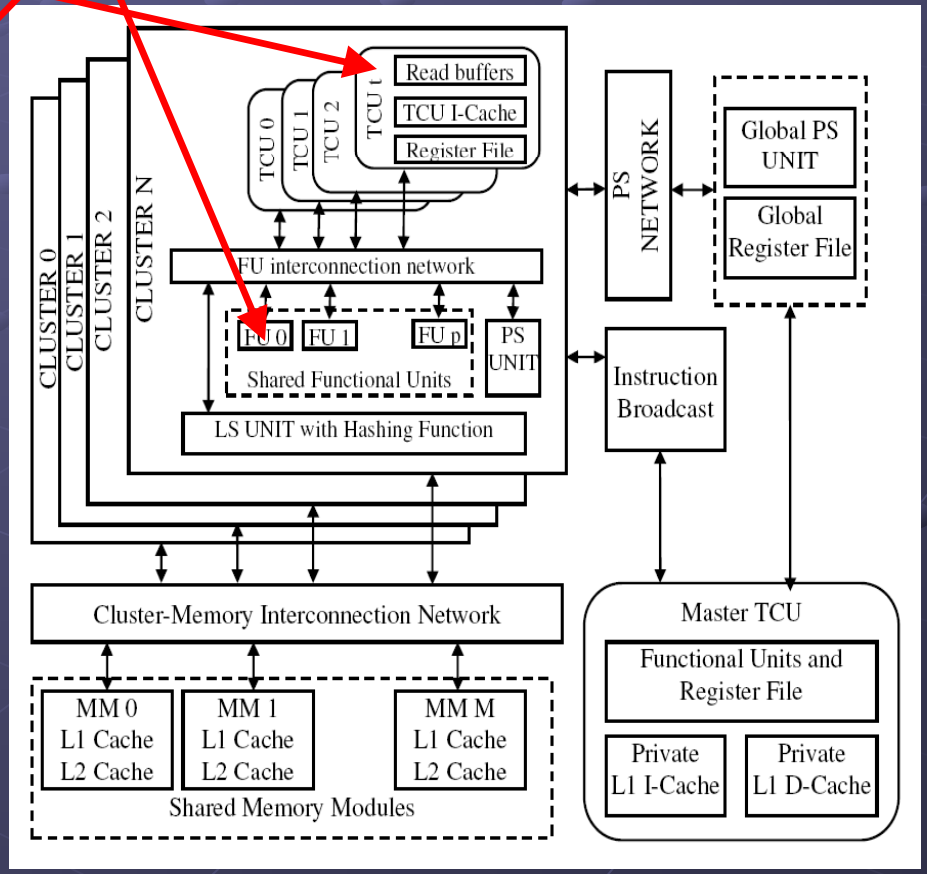
GeForce and XMT Similarities





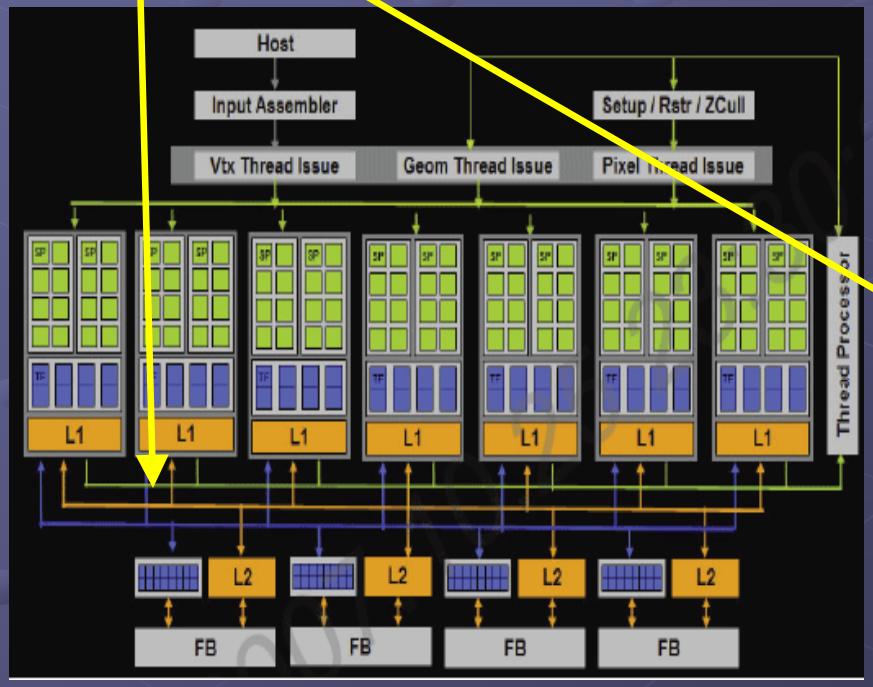
GeForce and XMT Similarities

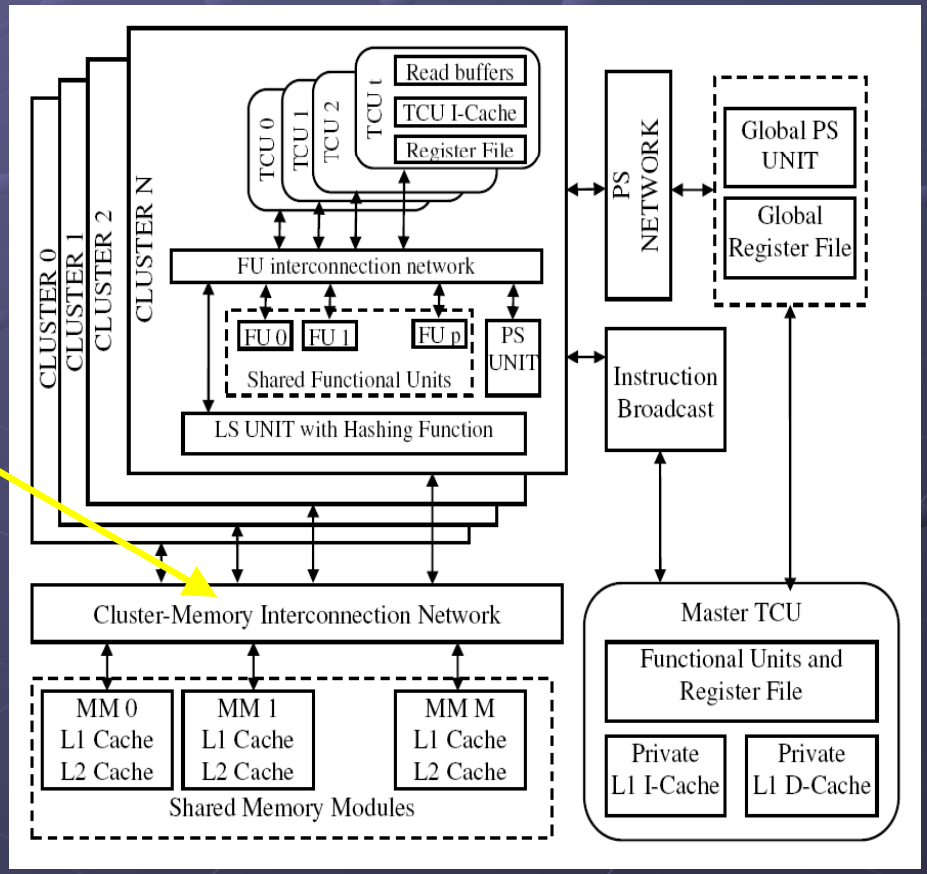
Clusters of processors, functional units



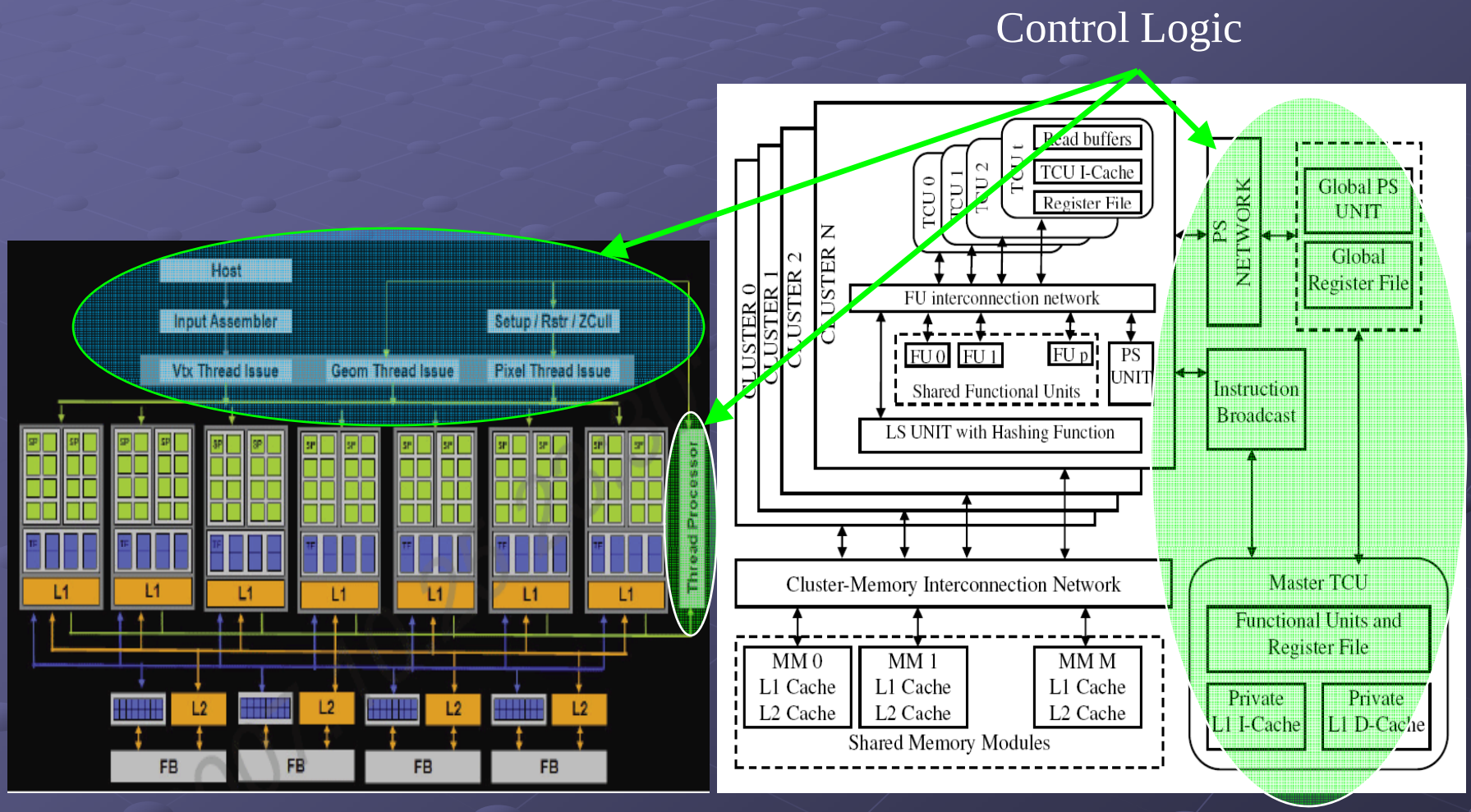
GeForce and XMT Similarities

On chip memory and access network





GeForce and XMT Similarities



Does XMT Meet Unification Requirements?

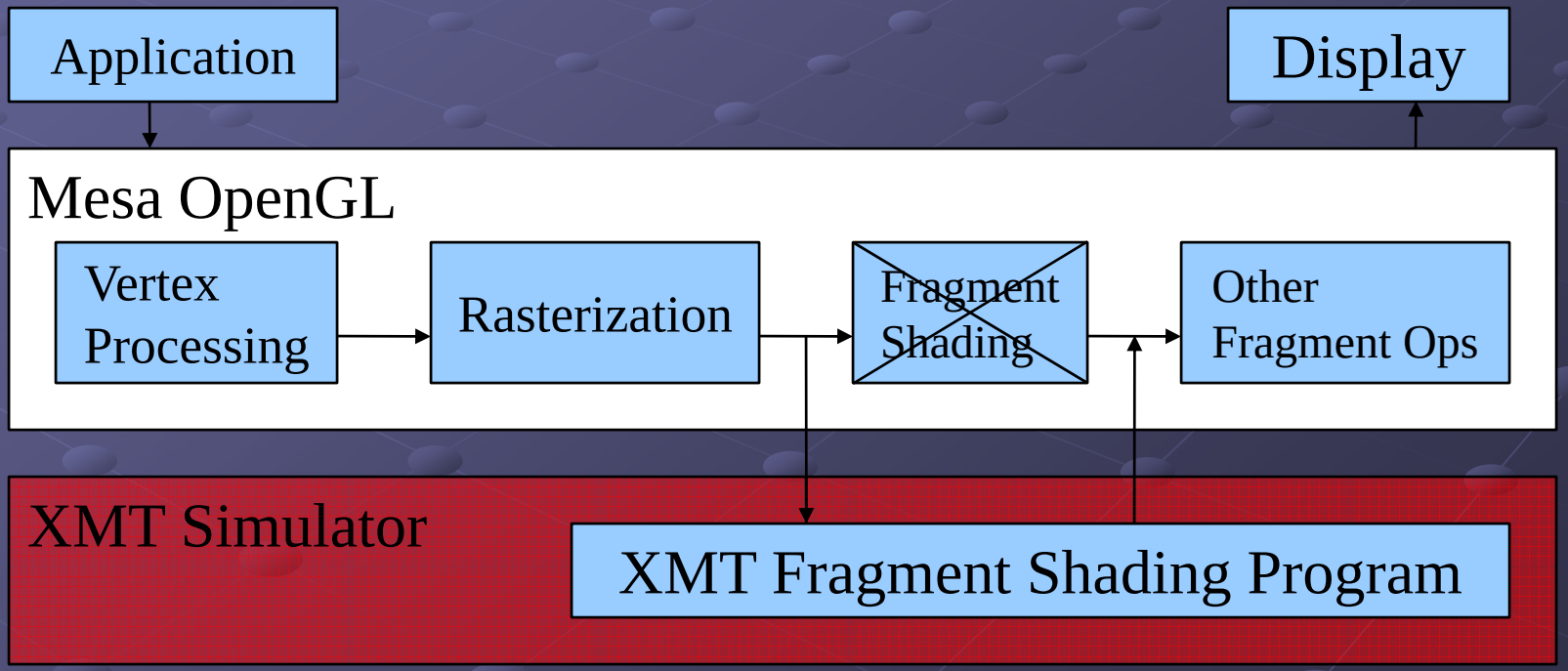
- Unification requirements
  - Ability to stand alone
  - Easy and flexible to program
  - Must perform GP tasks well
  - Competitive with modern GPUs
- Graphics performance only unknown
- GPUs are a unique competitor
  - Successful, commodity, parallel hardware

Our Experiment

- Simulated XMT system vs. several real GPUs on fragment shading
  - Compute shading – memory light, general
  - Texture shading – memory heavy, specialized
- Only fragment shading stage compared

Simulating Fragment Shading

- Simulated XMT used in place of the fragment shading step in software



The Competitors

Simulated XMT*	
ATI x700	Released mid 2004
NVidia GeForce 7900	Released mid 2006
NVidia GeForce 8800	Released late 2006

*Scaled for the same FLOPS level as GeForce 8800

XMT Variants

- We used 3 variants of XMT
  - Version 1 - unmodified
  - Version 2 - with graphics ISA
    - Floor, Fraction, Linear Interpolation
  - Version 3 – graphics ISA and 4-way vector computations on 8-bit arithmetic

Compute Shader Results

x700	354 FPS
GeForce 7900	1917 FPS
GeForce 8800	**2760 FPS**
XMT version 1	1423 FPS
XMT version 2	**3222 FPS**

Texture Shader Results

x700	1846 FPS
GeForce 7900	**8632 FPS**
GeForce 8800	**3179 FPS**
XMT version 1	197 FPS
XMT version 2	275 FPS
XMT version 3	**487 FPS**

Analysis

- XMT compute shades faster
- XMT texture shades much slower
  - Acceptable for some apps, not all
- The GeForce GPUs follow the same trend
  - Sacrificing speed on most used apps for greater flexibility on others

Summary

- Divide between CPU/GPU is blurring
- A unified system gives
  - Ease of programming
  - Good GP performance
  - Good graphics performance
- Combination systems needed for truly high performance apps
  - An XMT + GPU system could provide the best of both worlds
  - How to partition between them? Can they cooperate?
- www.umiacs.umd.edu/users/vishkin/XMT