CMSC 611: Advanced Computer Architecture

Parallel Computation (2)

Most slides adapted from David Patterson. Some from Mohomed Younis

Shared Address Model

- Physical locations
  - Each PE can name every physical location in the machine
- Shared data
  - Each process can name all data it shares with other processes

Shared Address Model

- Data transfer
  - Use load and store, VM maps to local or remote location
  - Extra memory level: cache remote data
  - Significant research on making the translation transparent and scalable for many nodes
    - Handling data consistency and protection challenging
    - Latency depends on the underlying hardware architecture (bus bandwidth, memory access time and support for address translation)
    - Scalability is limited given that the communication model is so tightly coupled with process address space

Data Parallel Languages

- SIMD programming
  - PE point of view
  - Data: shared or per-PE
    - What data is distributed?
    - What is shared over PE subset
    - What data is broadcast with instruction stream?
  - Data layout: shape [256][256]d;
  - Communication primitives
  - Higher-level operations
    - Prefix sum: $[i]r = \sum_{j \le i} [j]d$
      - 1,1,2,3,4 → 1,1+1=2,2+2=4,4+3=7,7+4=11

Single Program Multiple Data

- Many problems do not map well to SIMD
  - Better utilization from MIMD or ILP
- Data parallel model ⇒ Single Program Multiple Data (SPMD) model
  - All processors execute identical program
  - Same program for SIMD, SISD or MIMD
  - Compiler handles mapping to architecture

Three Fundamental Issues

- 1: Naming: how to solve large problem fast
  - what data is shared
  - how it is addressed
  - what operations can access data
  - how processes refer to each other
- Choice of naming affects code produced by a compiler
  - Just remember and load address or keep track of processor number and local virtual address for message passing
- Choice of naming affects replication of data
  - In cache memory hierarchy or via SW replication and consistency

Naming Address Spaces

- Global physical address space
  - any processor can generate, address and access it in a single operation
- Global virtual address space
  - if the address space of each process can be configured to contain all shared data of the parallel program
    - memory can be anywhere: virtual address translation handles it
- Segmented shared address space
  - locations are named <process number, address> uniformly for all processes of the parallel program

Three Fundamental Issues

- 2: Synchronization: To cooperate, processes must coordinate
  - Message passing is implicit coordination with transmission or arrival of data
  - Shared address → additional operations to explicitly coordinate:
    e.g., write a flag, awaken a thread, interrupt a processor

Three Fundamental Issues

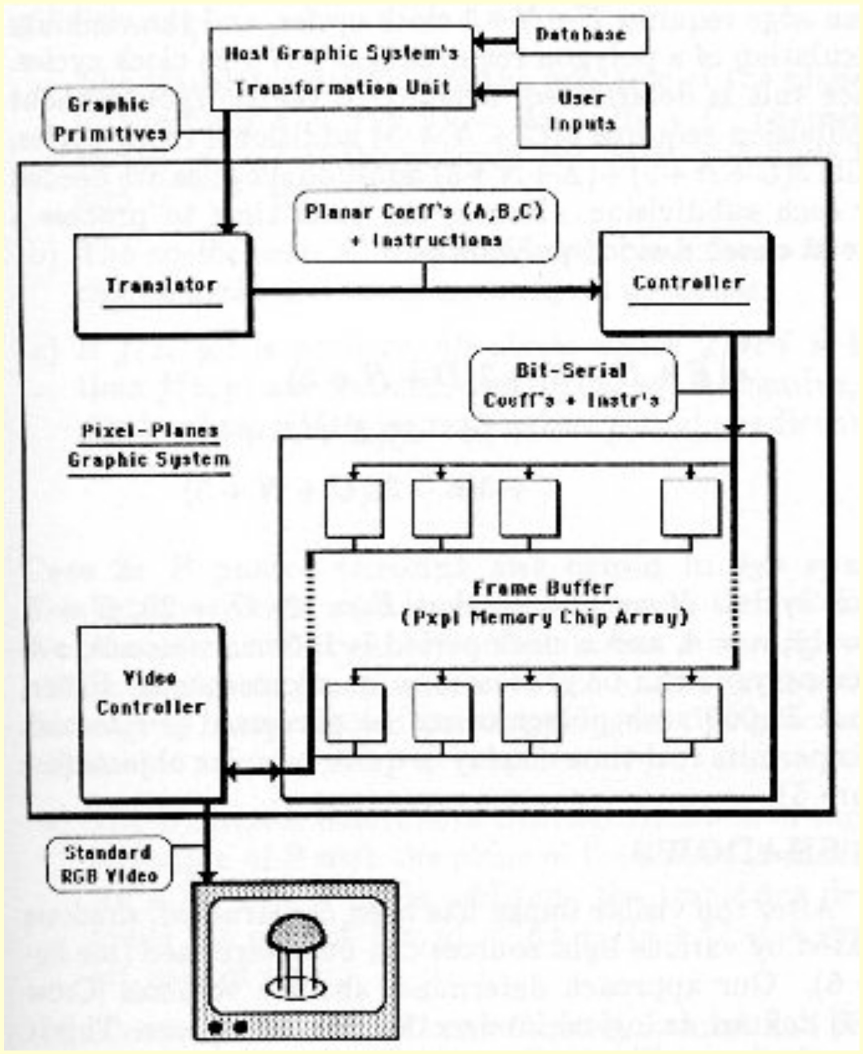
- 3: Latency and Bandwidth
  - Bandwidth
    - Need high bandwidth in communication
    - Cannot scale, but stay close
    - Match limits in network, memory, and processor
    - Overhead to communicate is a problem in many machines
  - Latency
    - Affects performance, since processor may have to wait
    - Affects ease of programming, since requires more thought to overlap communication and computation
  - Latency Hiding
    - How can a mechanism help hide latency?
    - Examples: overlap message send with computation, pre-fetch data, switch to other tasks

Some Graphics Examples

- Pixel-Planes 4
- Pixel-Planes 5
- Pixel-Flow
- NVIDIA GeForce 6 series
- ATI 7800

Pixel-Planes 4

- 512x512 SIMD array (full screen)

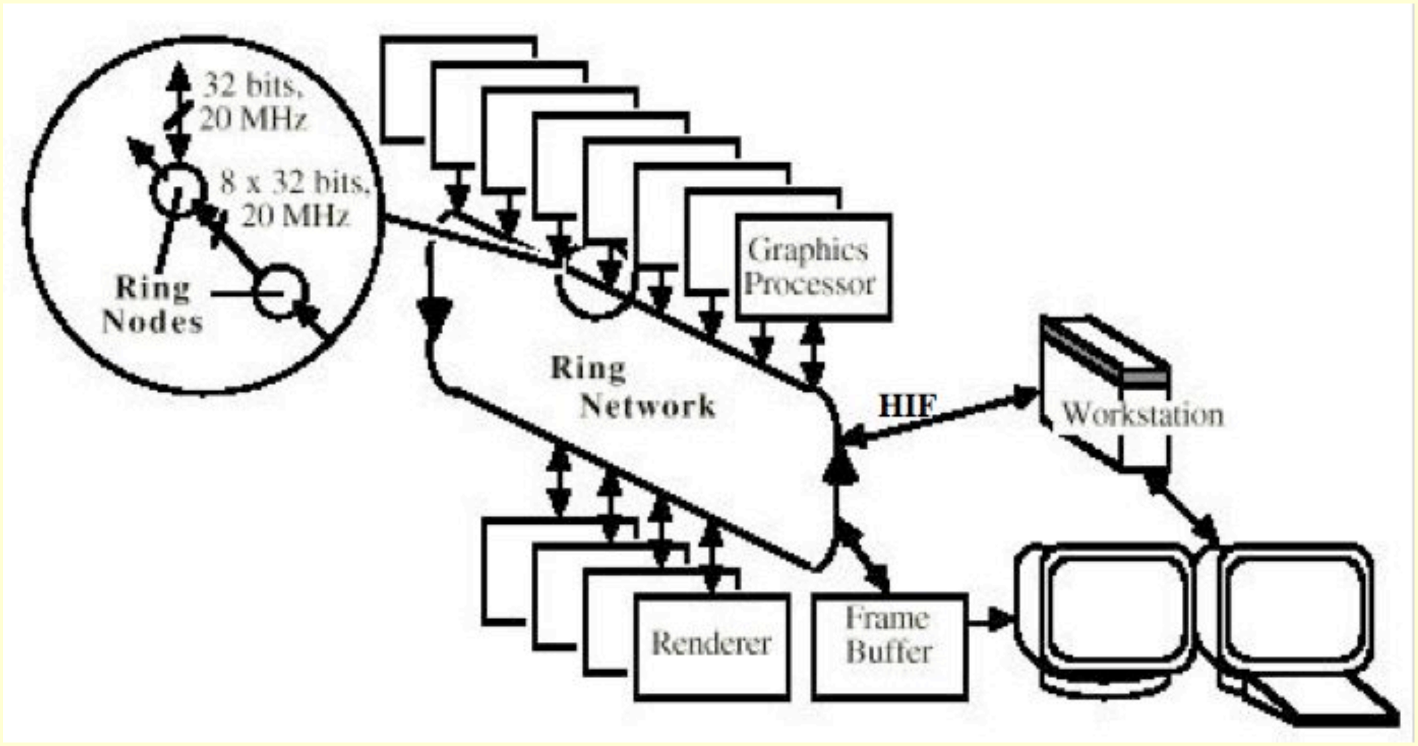


Database
Host Graphic System's Transformation Unit
User Inputs
Graphic Primitives
Planar Coeff's (A,B,C) + Instructions
Translator
Controller
Bit-Serial Coeff's + Instr's
Pixel-Planes Graphic System
Frame Buffer (Pxpl Memory Chip Array)
Video Controller
Standard RGB Video

Fuchs, et al., "Fast Spheres, Shadows, Textures, Transparencies, and Image Enhancements in Pixel-Planes", SIGGRAPH 1985

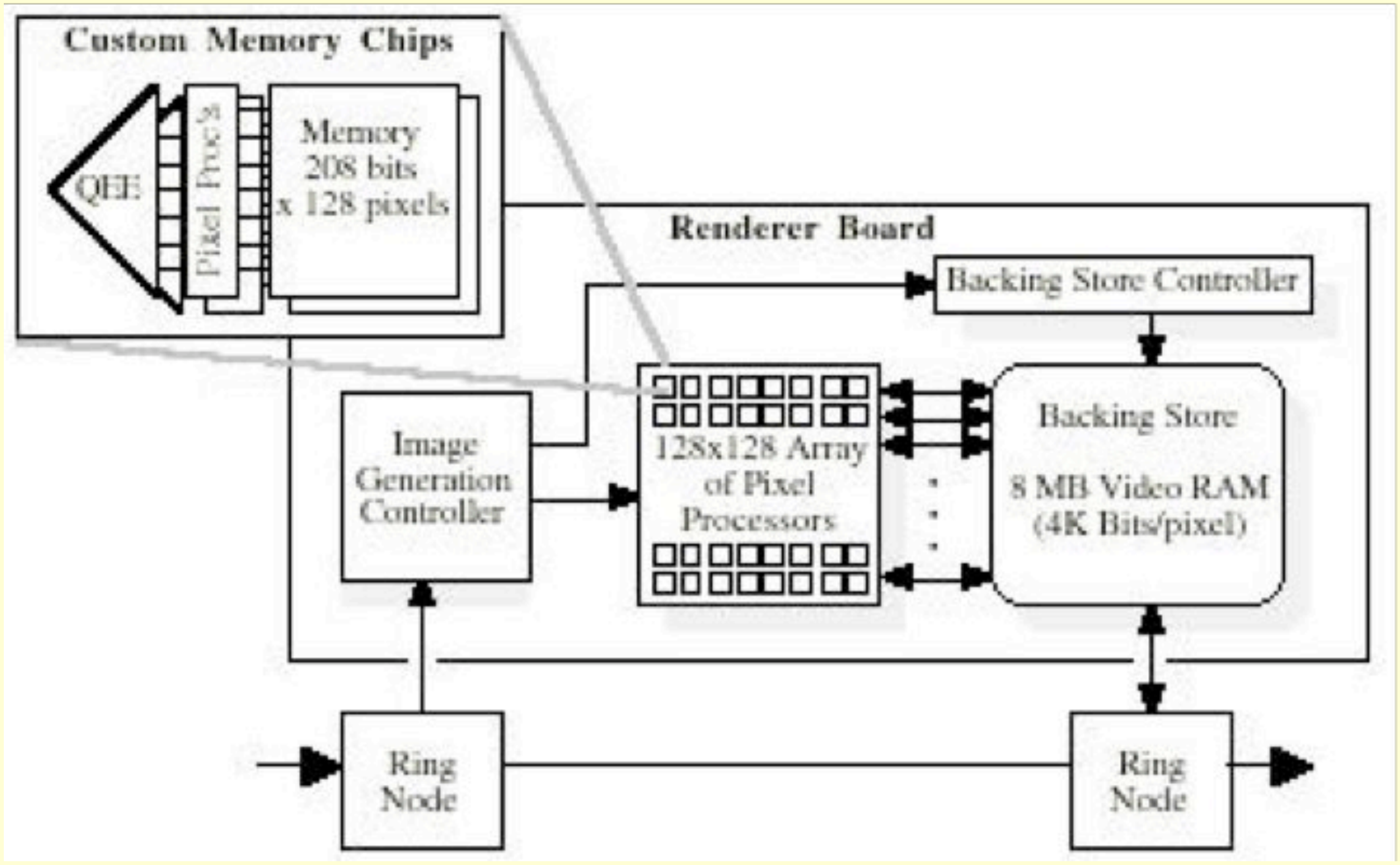
Pixel-Planes 5

- Message-passing
- ~40 i860 CPUs
- ~20 128x128 SIMD arrays (~80 tiles/screen)



uchs, et al., "Pixel-Planes 5: A Heterogeneous Multiprocessor Graphics System Using Processor Enhanced Memories", SIGGRAPH 89

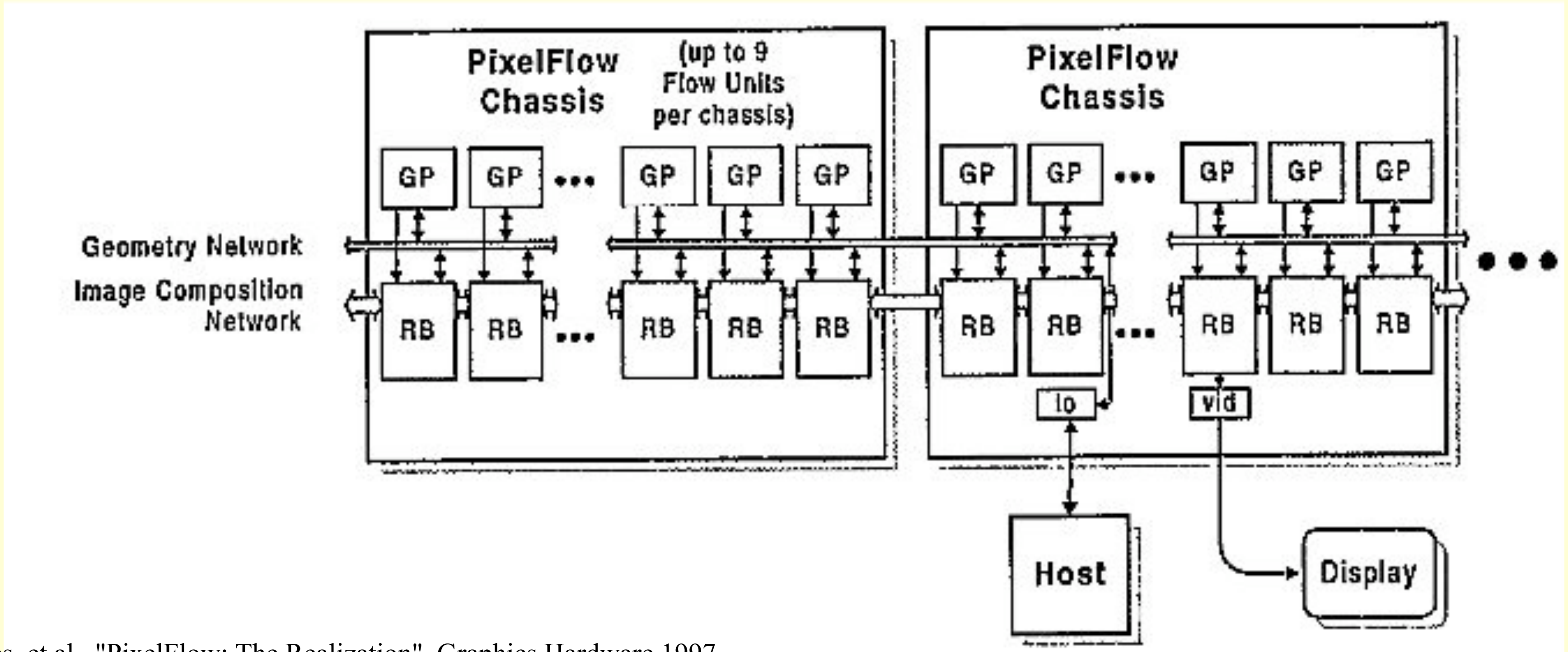
Pixel-Planes 5



uchs, et al., "Pixel-Planes 5: A Heterogeneous Multiprocessor Graphics System Using Processor Enhanced Memories", SIGGRAPH 89

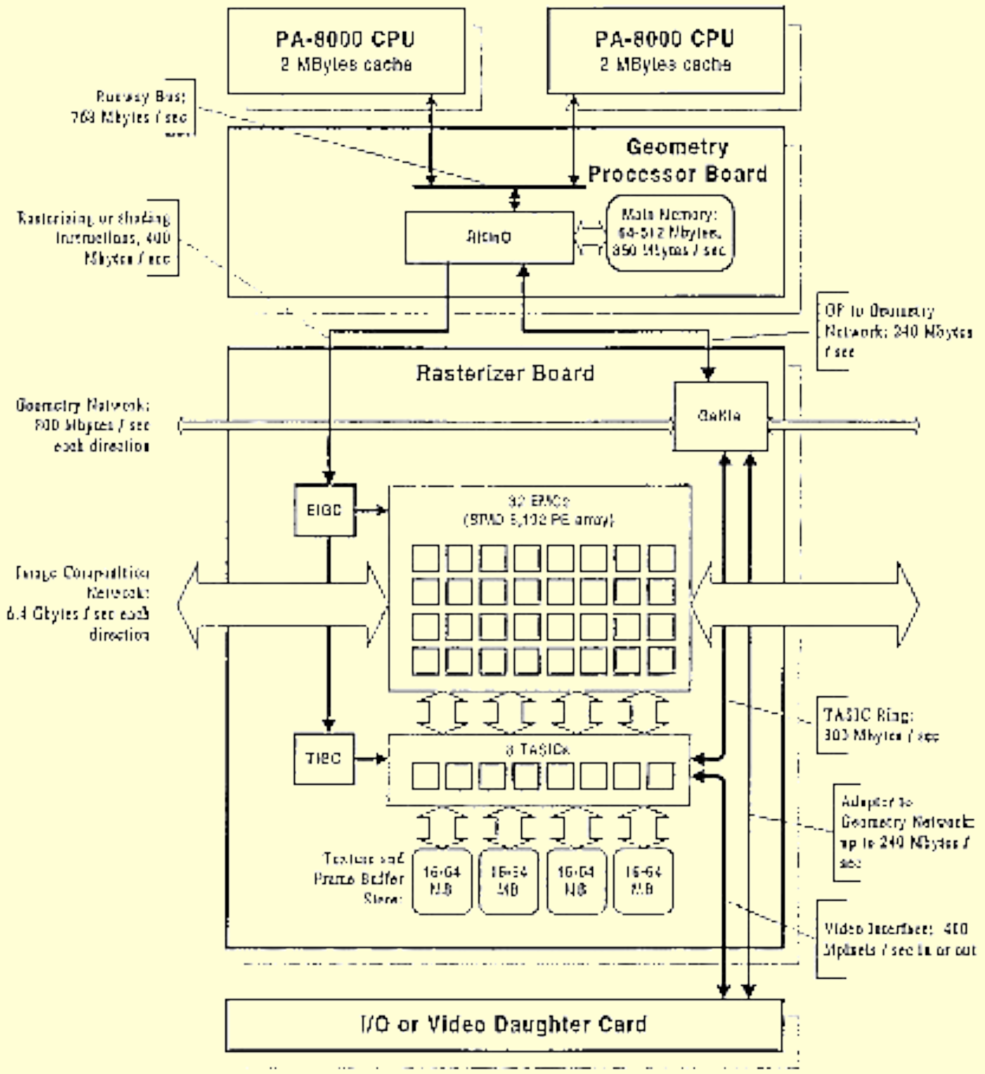
Pixel-Flow

- Message-passing
- ~35 nodes, each with
  - 2 HP-PA 8000 CPUs
  - 128x64 SIMD array (~160 tiles/screen)



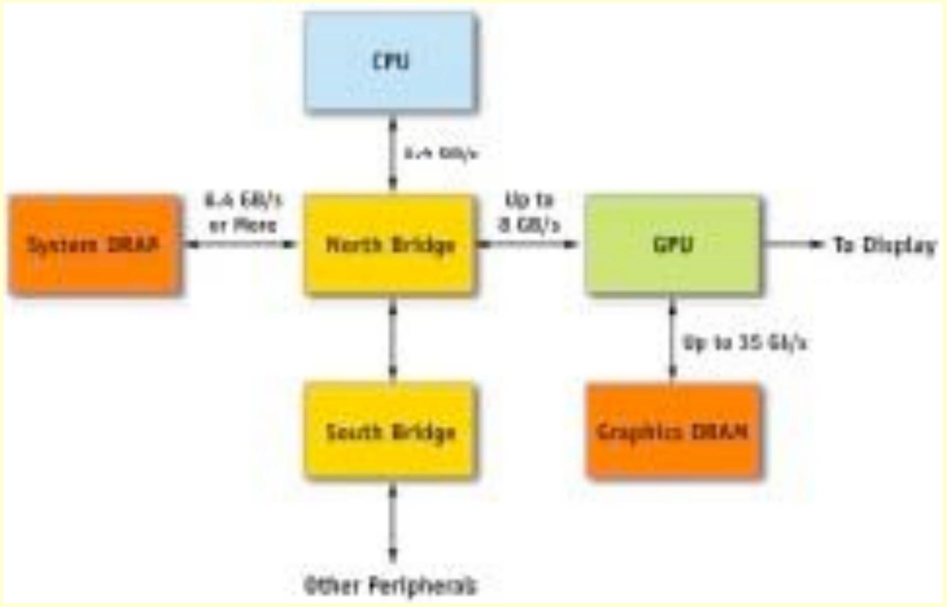
Eyles, et al., "PixelFlow: The Realization", Graphics Hardware 1997

Pixel-Flow


PA-8000 CPU
2 MBytes cache
PA-8000 CPU
2 MBytes cache
Runway Bus: 768 Mbytes / sec
Geometry Processor Board
RHInO
Main Memory: 64-512 Mbytes, 350 Mbytes / sec
Rasterizing or shading instructions, 400 Mbytes / sec
GP to Geometry Network: 240 Mbytes / sec
Rasterizer Board
GeNIe
Geometry Network: 800 Mbytes / sec each direction
EIGC
32 EMCs
(SIMD 8,192 PE array)
Image Composition Network: 6.4 Gbytes / sec each direction
TASIC Ring: 300 Mbytes / sec
TIGC
8 TASICs
Adapter to Geometry Network: up to 240 Mbytes / sec
Texture and Frame Buffer Store:
16-64 MB
16-64 MB
16-64 MB
16-64 MB
Video Interface: 400 Mpixels / sec in or out
I/O or Video Daughter Card

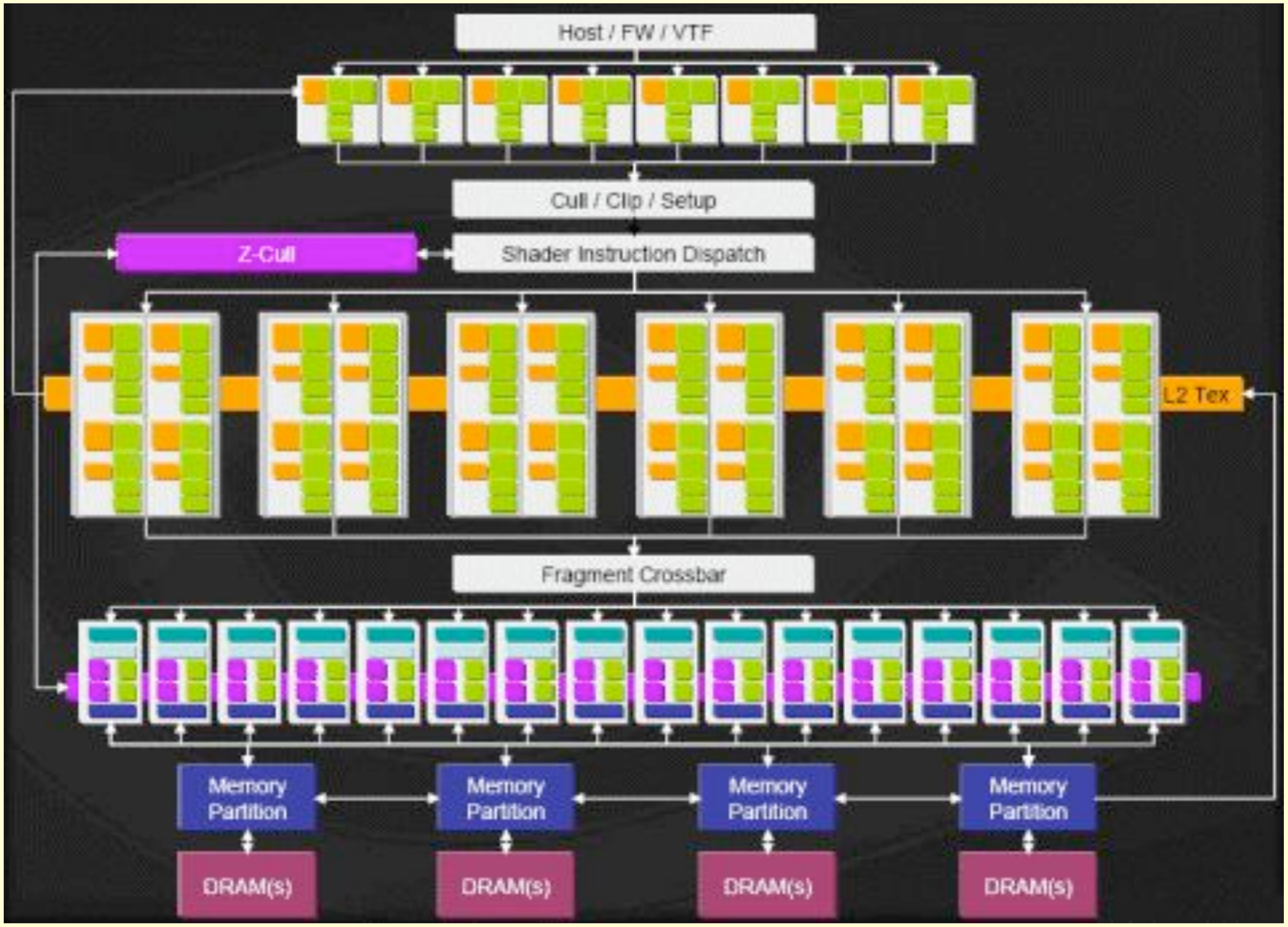
Eyles, et al., "PixelFlow: The Realization", Graphics Hardware 1997

PC Graphics Cards

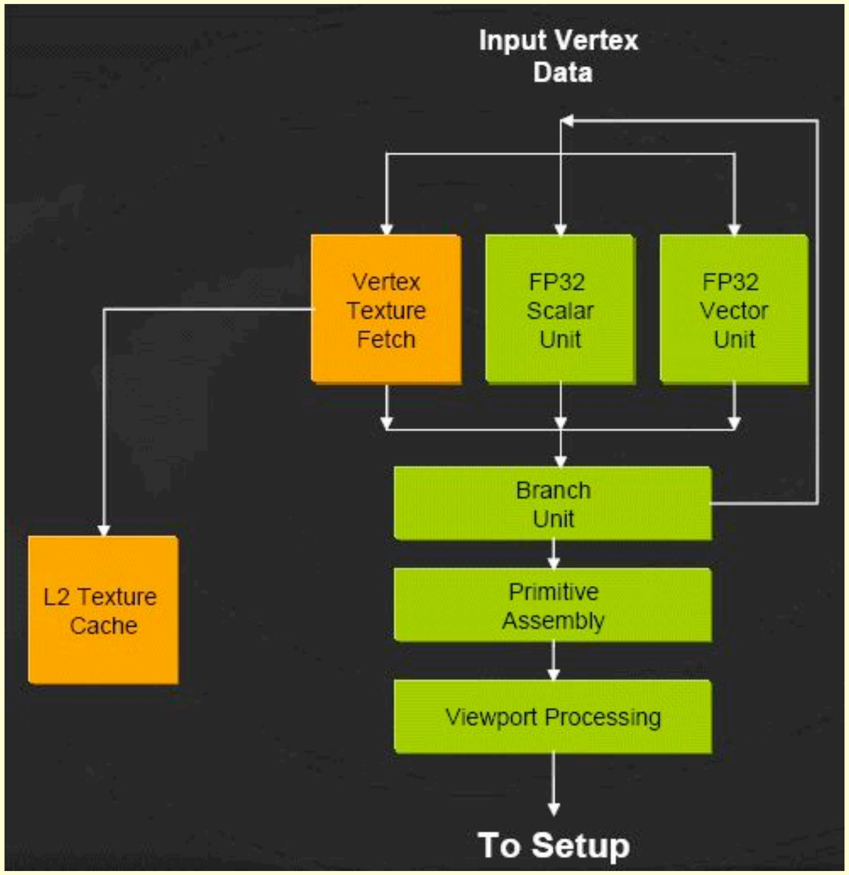


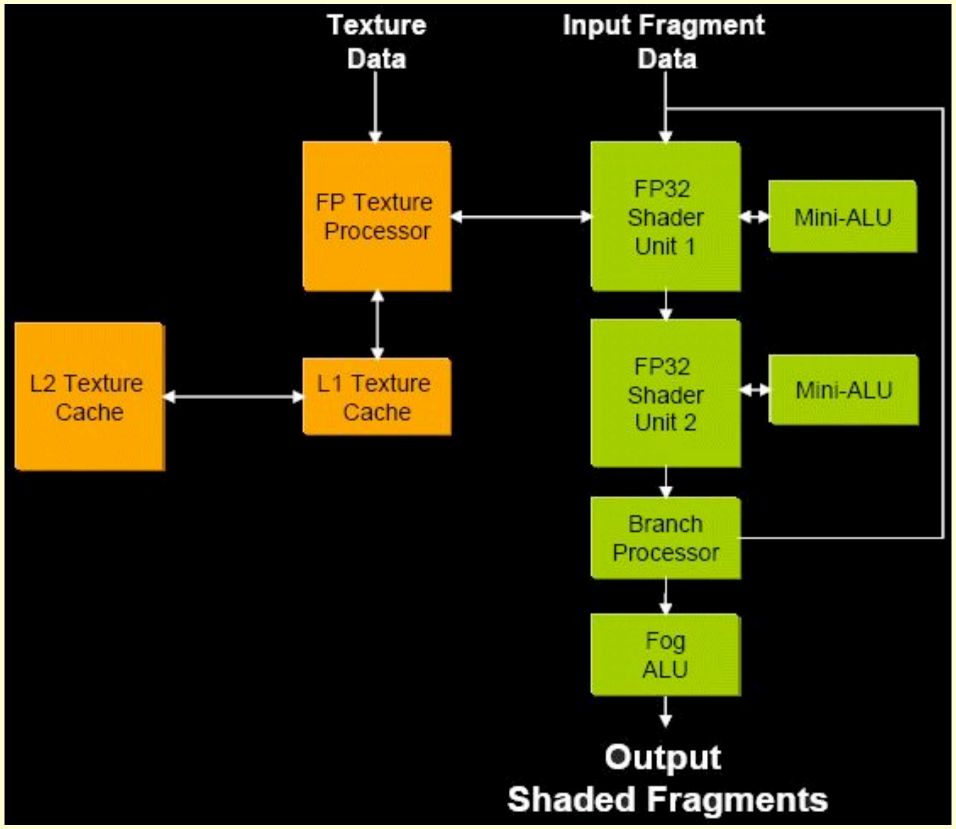
Kilgariff and Fernando, “The GeForce 6 Series Architecture”, GPU Gems 2

NVIDIA 7800 / G70

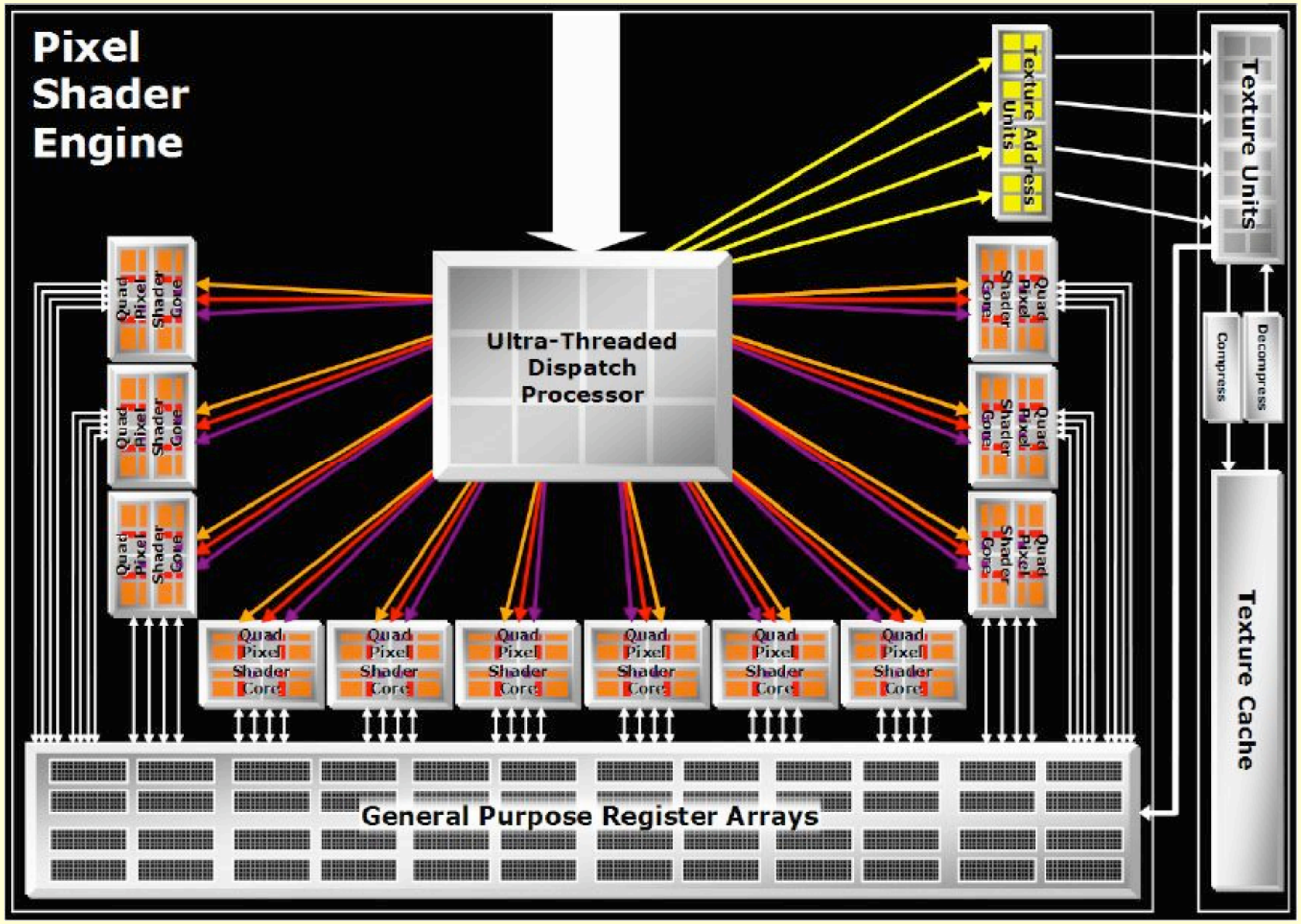


NVIDIA 7800 / G70

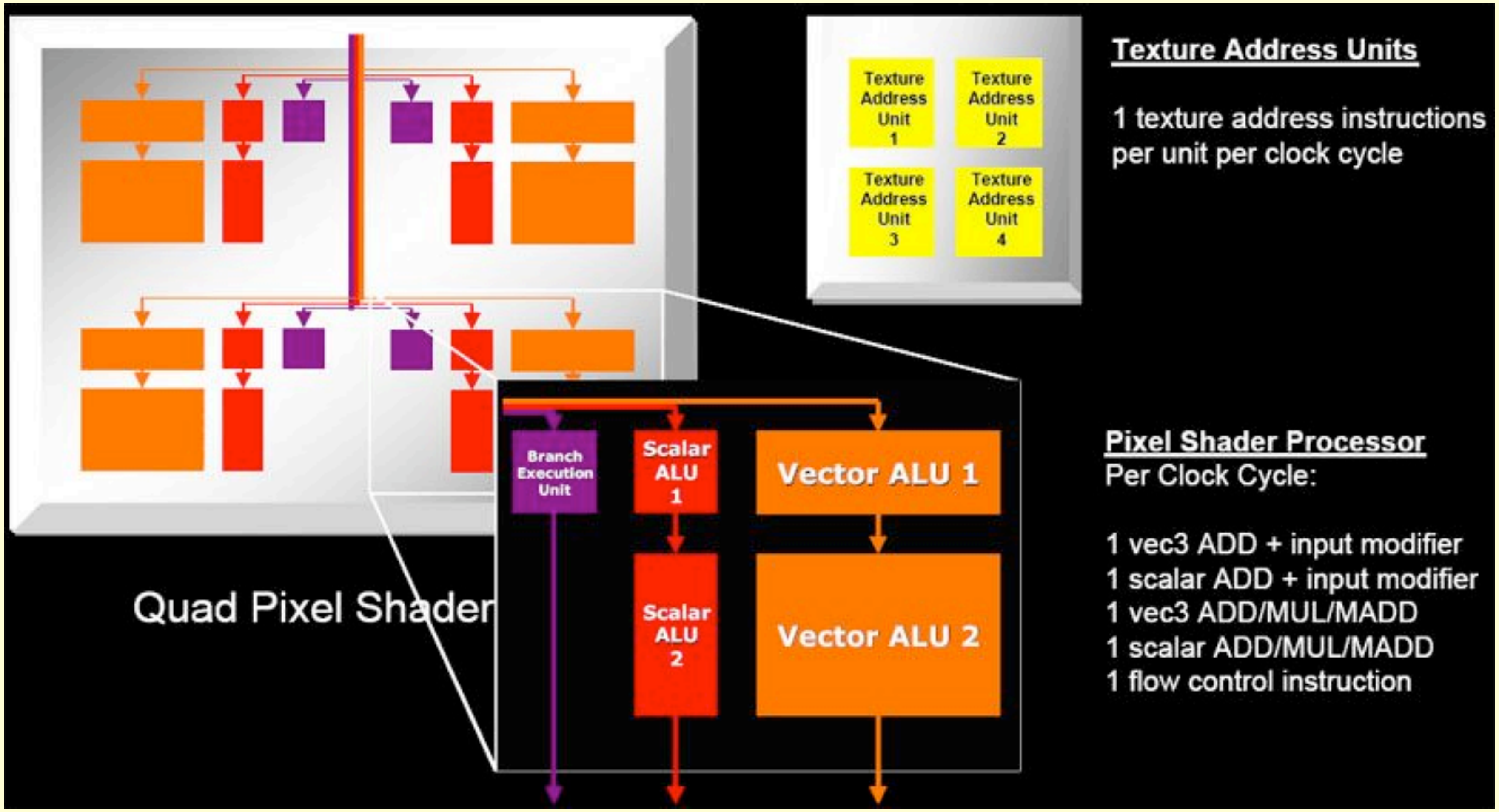




ATI x1900 / R580



ATI x1900 / R580



Texture Address Units

1 texture address instructions per unit per clock cycle

Pixel Shader Processor
Per Clock Cycle:

1 vec3 ADD + input modifier
1 scalar ADD + input modifier
1 vec3 ADD/MUL/MADD
1 scalar ADD/MUL/MADD
1 flow control instruction