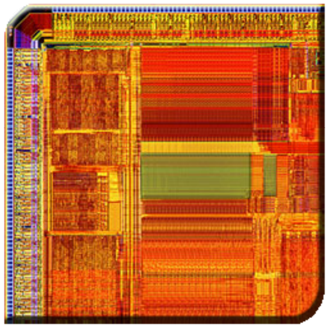
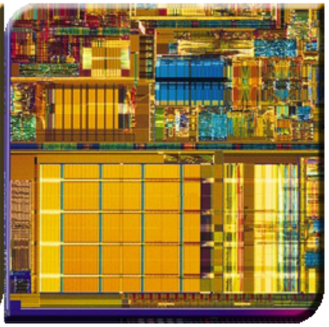
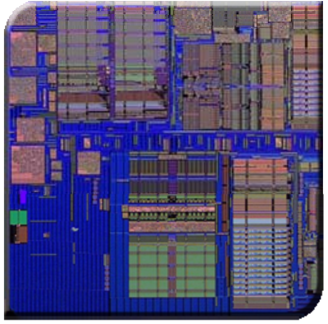




Larrabee: A Many-Core x86 Architecture for Visual Computing from Intel



Prof. Hsien-Hsin S. Lee

School of Electrical and Computer Engineering

Georgia Tech

Disclaimer: The materials of this presentation were obtained from available resources including In-Stat's Microprocessor Report, Intel's Larrabee paper published in SIGGRAPH 2008, and certain speculations from both the presenter and several online reviews. The opinions suggested by this slide set are purely the presenter's interpretation from these materials, they do not necessarily represent the official opinions of Intel, Nvidia, or Georgia Tech.

Vision, Ambition, and Design Goals

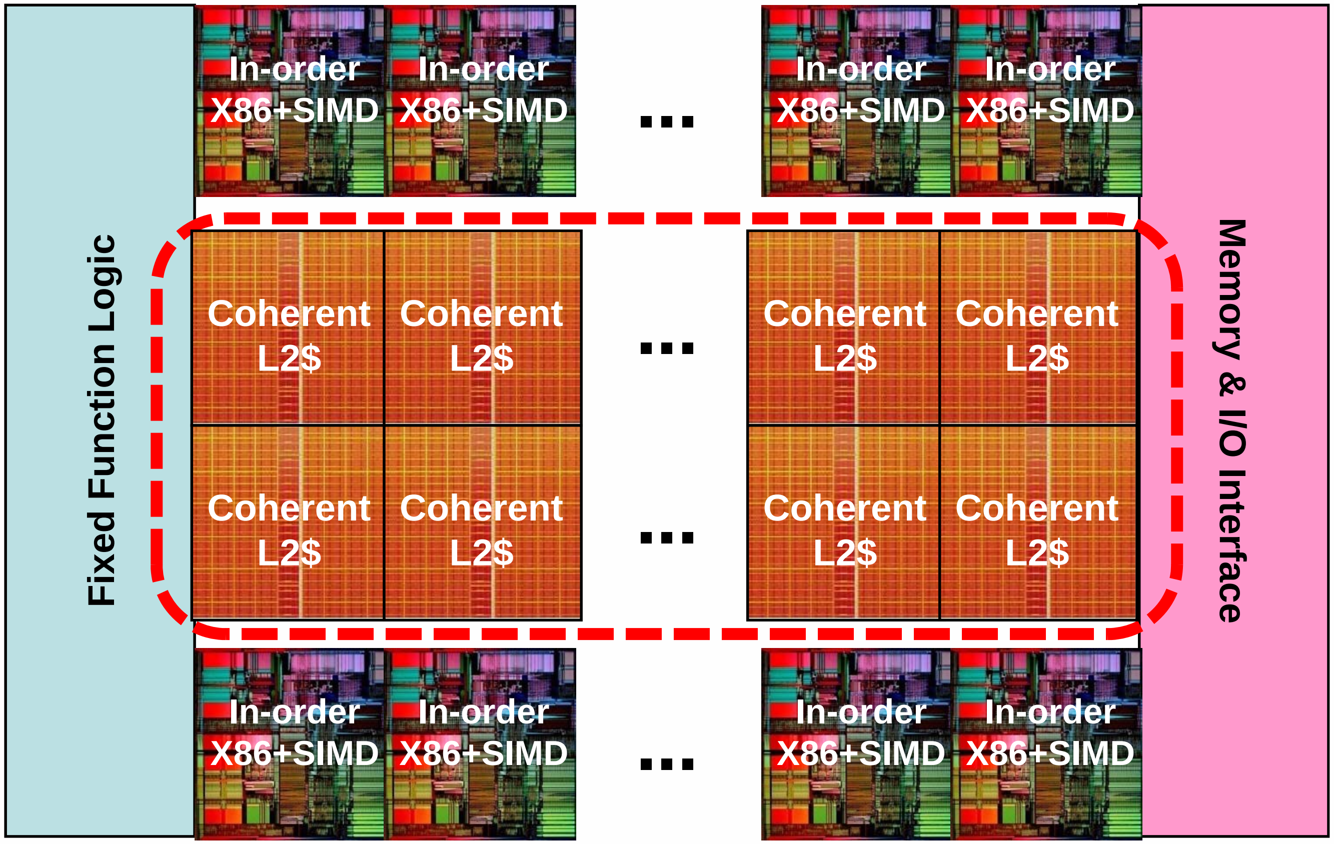
- Intel: Software is the New Hardware !

- Intel: x86 ISA makes parallel program easier

  - Better flexibility and programmability
  - Support subroutine call and page faulting
  - Mostly software rendering pipeline, except texture filtering

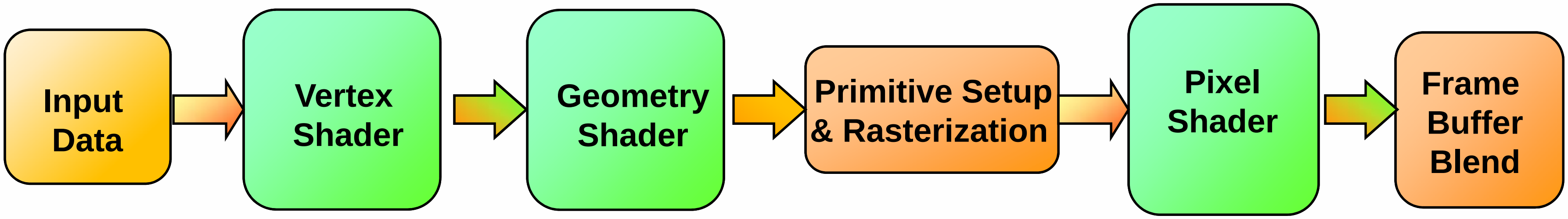
- Note that, general goal for current day GPGPU designers (well, also Intel's Larrabee architects)
  - ↑ performance per mm^2
  - ↑ performance per watt

The Larrabee Architecture

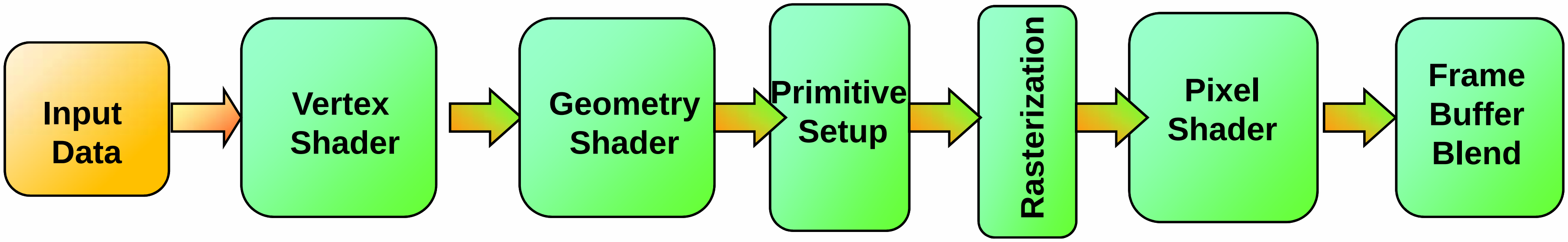


- Lots of x86 cores (8 to 64?)
- Fully coherence cache hierarchy

Programmable Pipeline Comparison



Conventional GPGPU pipeline (base on DirectX10)



Larrabee’s fully programmable pipeline

X86 Core

- LRB's "in-order" core is

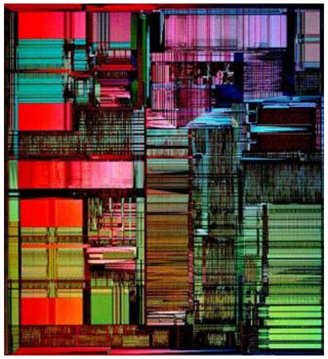
  The original Pentium (p54c, i.e., pre-MMX)

  + 64bit extensions

  + Larger L1 caches + a shared L2

  + 4-way multi-threading

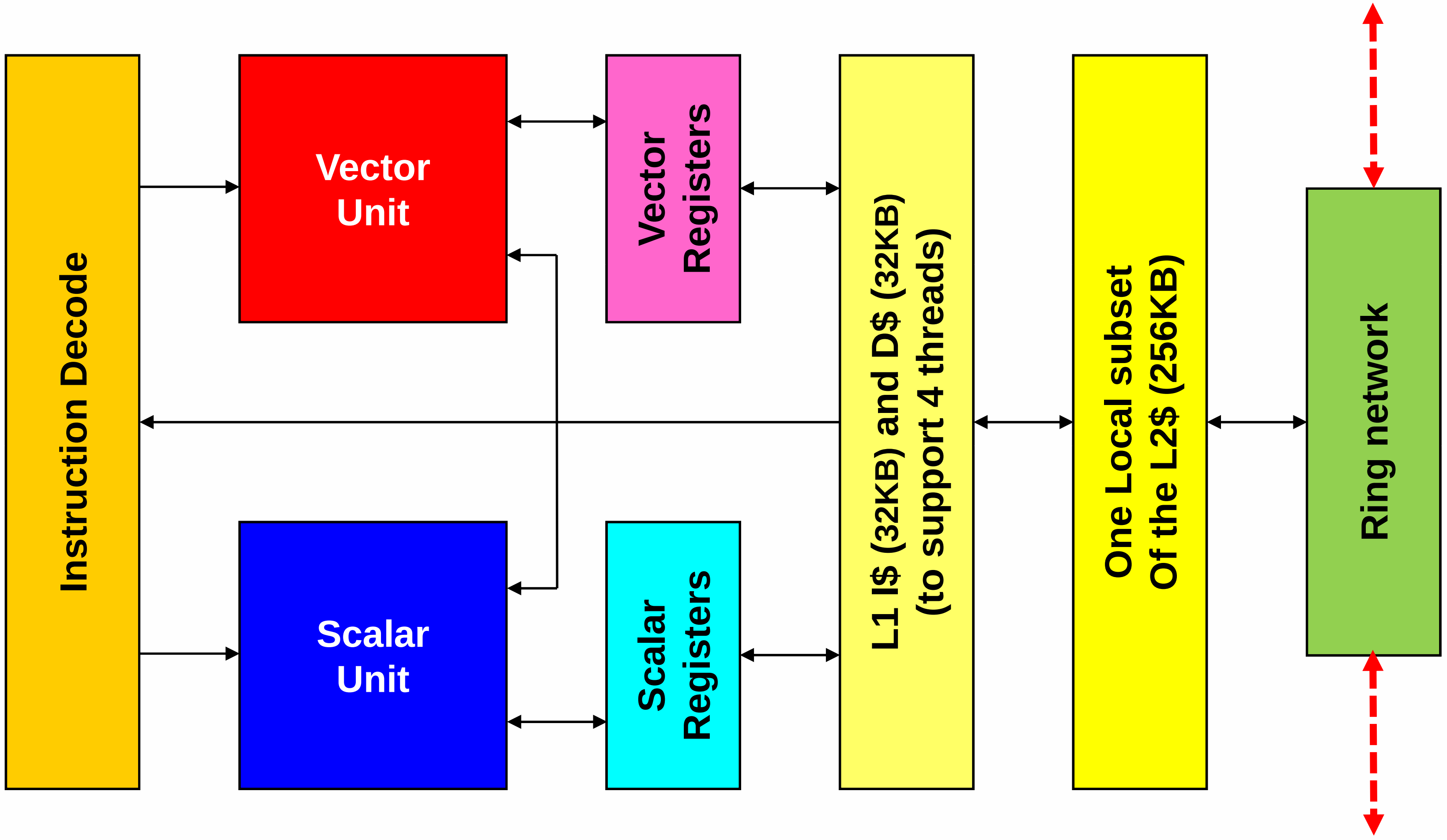
  + 16-wide VPU (Vector Processing Unit)



- Rumor has it: this is the thoroughly debugged P54C given back by Pentagon who got the original RTL from Intel to develop their radiation hardened version (which I really doubt)

- Compatibility is the keyword

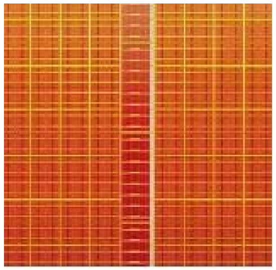
Single Larrabee Core


Instruction Decode
Vector Unit
Scalar Unit
Vector Registers
Scalar Registers
L1 I$ (32KB) and D$ (32KB) (to support 4 threads)
One Local subset Of the L2$ (256KB)
Ring network

Dual Issue Core

- Rely on compiler to pair two instructions for asymmetric pipes
  - Same as P54C
  - Primary instruction pipe (U-pipe)
    - All instructions
  - Secondary, more restricted pipe (V-pipe)
    - ld, st, simple ALU Ops, Brs, cache manipulation instructions, vector st
- 1GHz, 32 cores to reach 1 TeraFLOPS

Shared L2, Divided L2



- Each core has a local L2 subset
  - 256KB each
  - Enable parallel lookup among cores
- One core can access others’ subsets directly
- Entire L2 is coherent (no hassle like Cell DMA)
- SIGGRAPH paper shows a 4MB L2 indicating 16 cores

Cache Control Instructions

- Each core can
  - Fast-access its local subset of L2 (256KB)
  - Access other's L2 shares too
- Control for non-temporal streaming data (SSE)
- Prefetch to L1, or L2 only
- Mark a streaming cache line for early eviction
- Render target kept in L2 (e.g., FB, ZB, SB, etc)

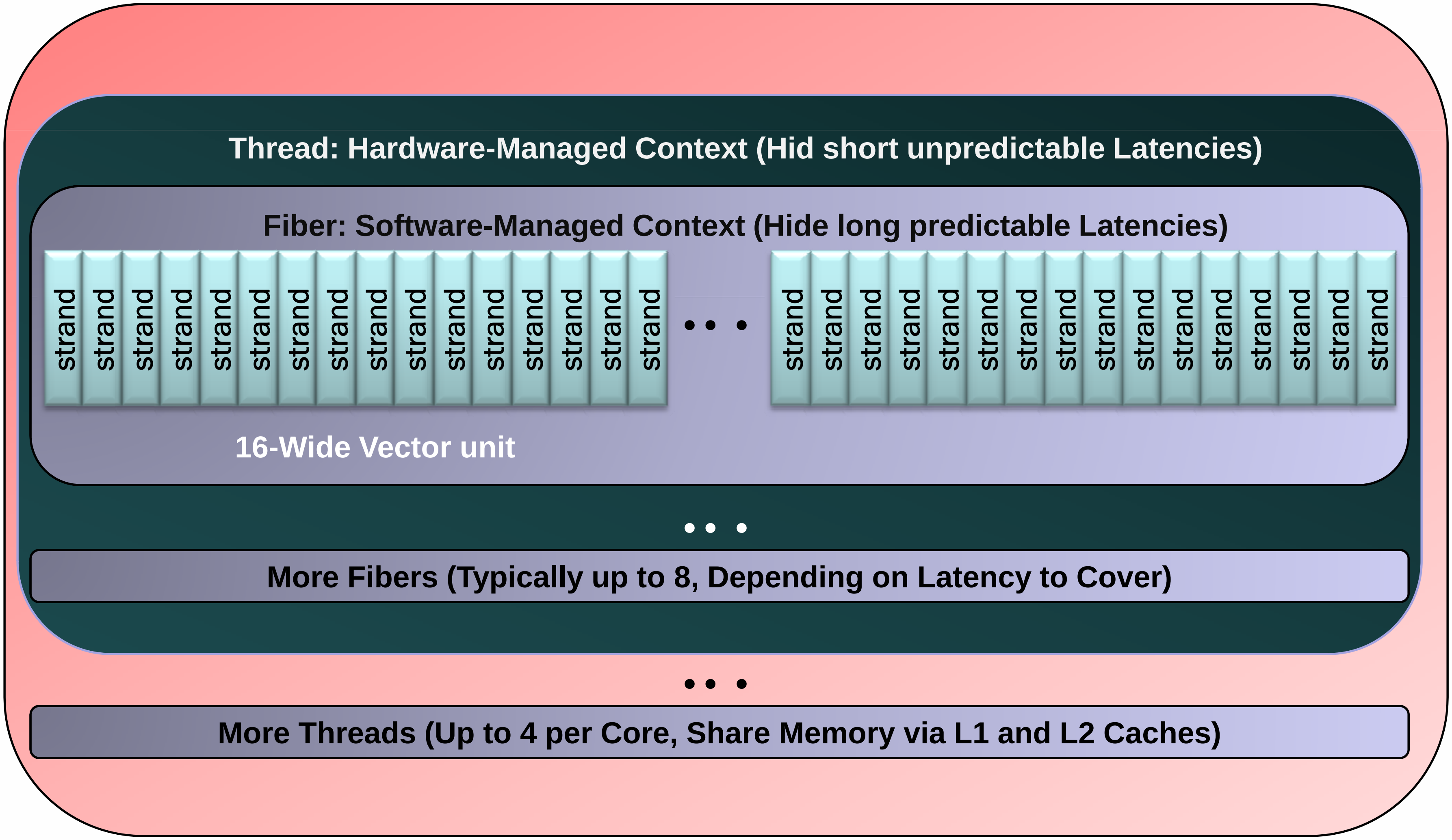
Ring Network

- Bi-directional ring network
  - All cores, L2, block of FF logic are attached to
  - 512-bit wide each direction
  - Simpler than mesh, easy wire routing
- One clock cycle for each stop (a hop)
  - Number of nodes between two parties determine latencies
  - Worst case: halfway around the ring
- Ring latency is small compared to DRAM access
- When > 16 cores: multiple, hierarchical rings will be needed (think about KSR MPP)

4-Way MT

- Four x86 contexts to support 4 hardware threads
- One thread picked per clock
- MT is especially helpful
  - When compiler fails to schedule code without stalls
  - Upon L1 misses
  - Can hide long vector instruction latency
  - Can switch thread on every clock

Larrabee Multithread Model



Thread: Hardware-Managed Context (Hid short unpredictable Latencies)

Fiber: Software-Managed Context (Hide long predictable Latencies)

strand strand strand strand strand strand strand strand strand strand strand strand strand strand strand strand • • • strand strand strand strand strand strand strand strand strand strand strand strand strand strand strand strand

16-Wide Vector unit

• • •

More Fibers (Typically up to 8, Depending on Latency to Cover)

• • •

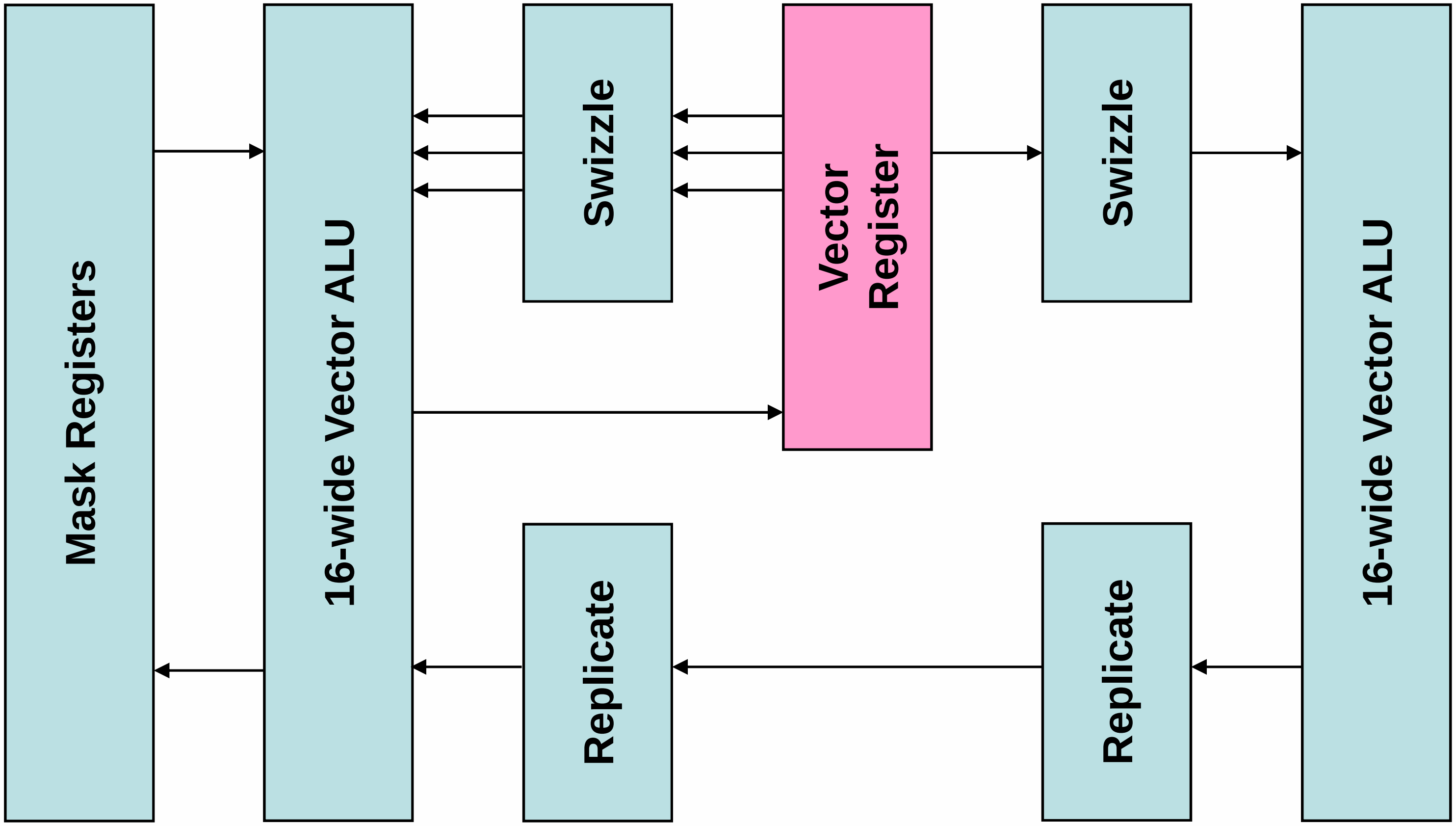
More Threads (Up to 4 per Core, Share Memory via L1 and L2 Caches)

Source: MPR

VPU (1/2)

- 16-wide Integer / single-precision FP
- 8-wide double-precision FP
- Ternary operands
  - One source can come from memory
- Free predication on every instruction
  - 16-bit predicate registers — one “enable” per lane
- Gather/scatter instructions
  - Read/write 16 results to/from 16 different offsets
- 1/3 the area of the LRB core!!!

VPU (2/2)


Mask Registers
16-wide Vector ALU
Swizzle
Vector Register
Swizzle
16-wide Vector ALU
Replicate
Replicate

Fixed Function Logic (1/3)

- Modern GPGPU have the following done in HW
  - Texture filtering, display processing, post-shader alpha blending, rasterization, interpolation, etc.
- LRB do all in SW except Texture Sampler Units
  - Much faster than software approach (12x ~ 40x)
    - Texture filtering still most commonly uses 8-bit operations
    - Efficiently selecting unaligned 2x2 quad requires a specialized pipelined gather logic
    - Filtering on VPU requires an impractical amount of RF b/w.
    - On-the-fly texture decompression drastically more efficient in dedicated hardware

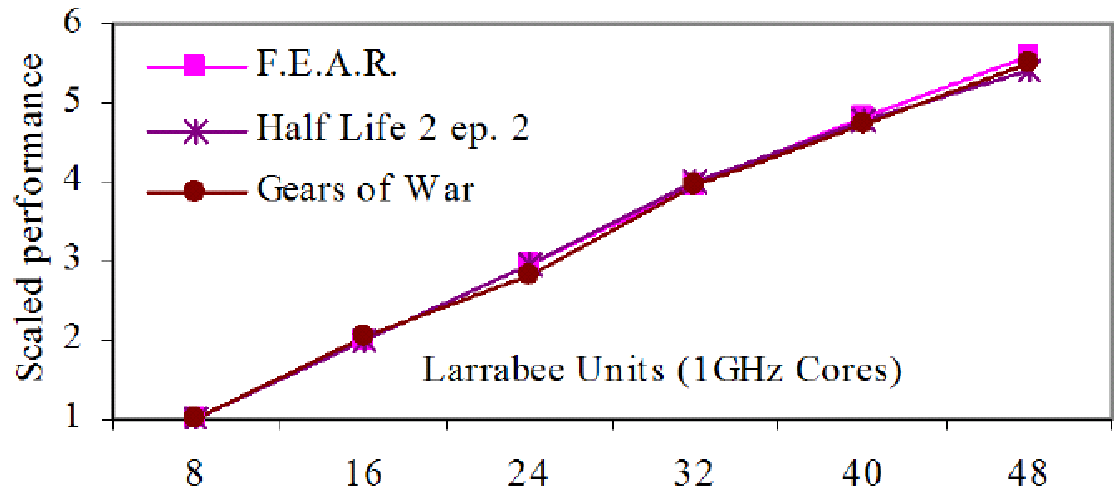
Fixed Function Logic (2/3)

- Similar to typical GPU texture logic
  - 32KB texture cache per core
  - Supports all the usual operations
    - DX10 compressed texture format
    - Mipmapping
    - Anisotropic filtering

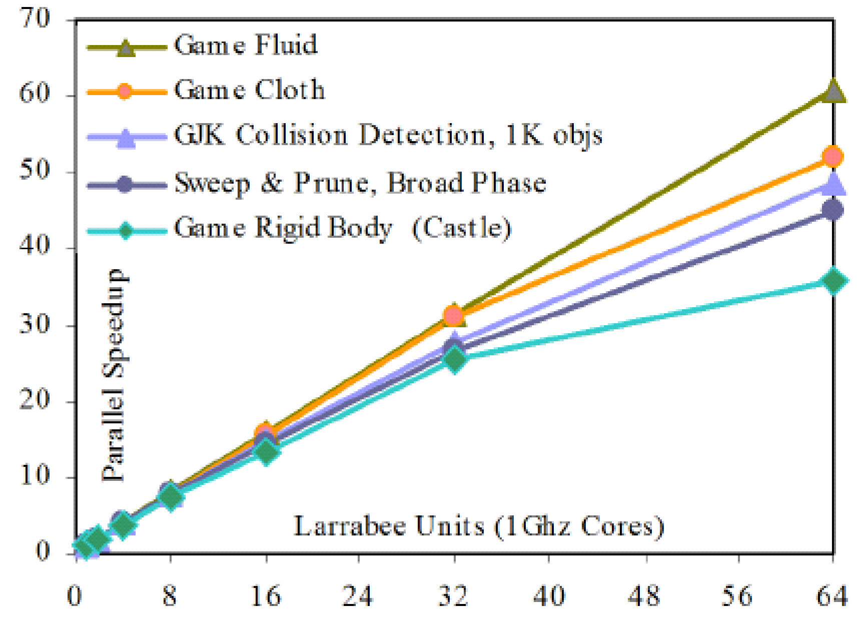
Fixed Function Logic (3/3)

- Core pass commands to the texture units through the L2$ and receive results the same way.
- Virtual-to-Physical page translation
  - Report any page misses to the core
  - Retry the texture filter command after the page is in memory
- LRB Still can perform texture operations on the cores if the performance is fast enough in software.

Simulation Data from SIGGRAPH paper



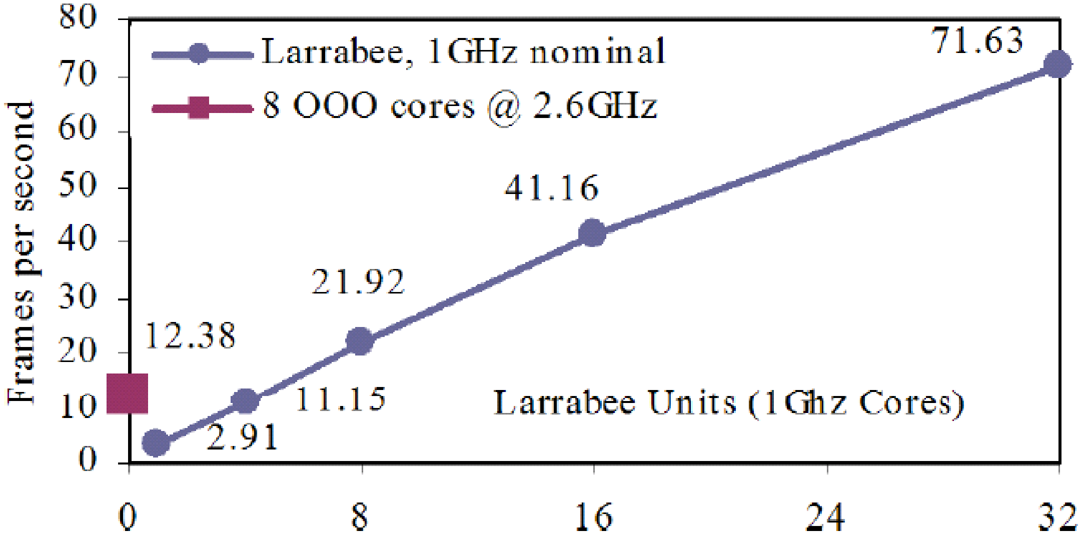
Scalable Performance for
3D games



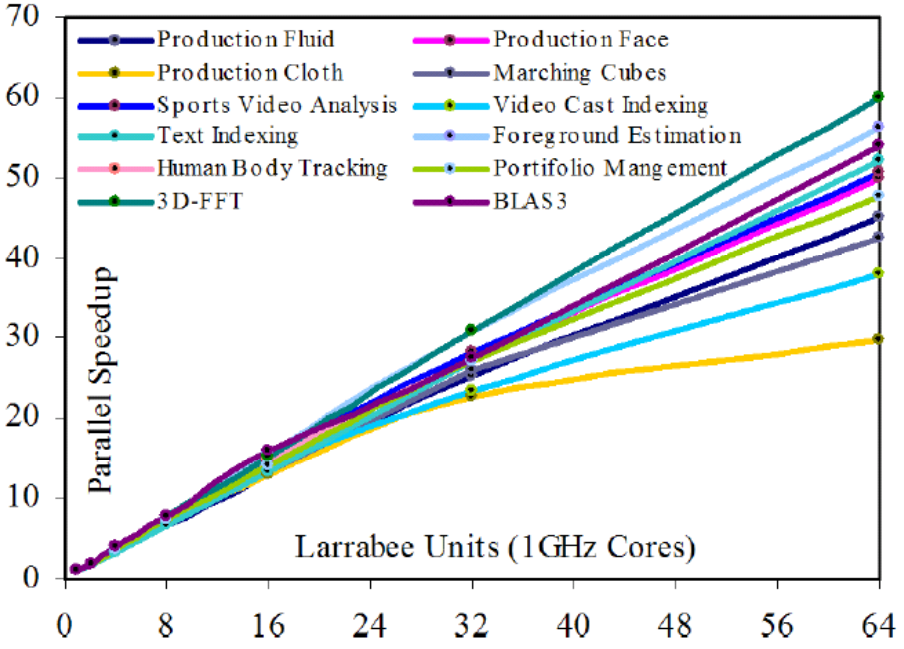
Scalable Performance for
3D game Physics

Source: SIGGRAPH08

Simulation Data from SIGGRAPH paper



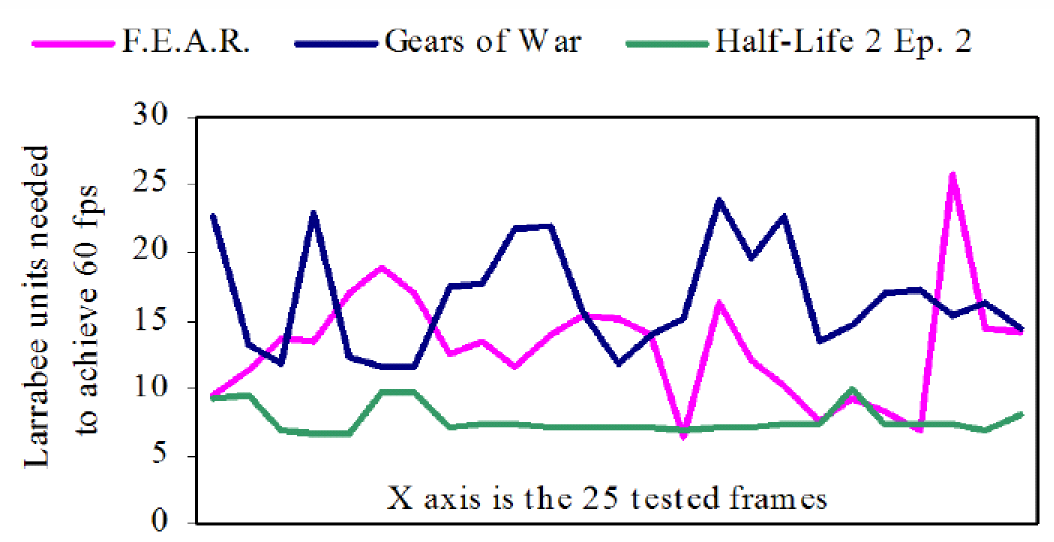
Scalable RT ray tracing



Non-graphics app & kernels

Source: SIGGRAPH08

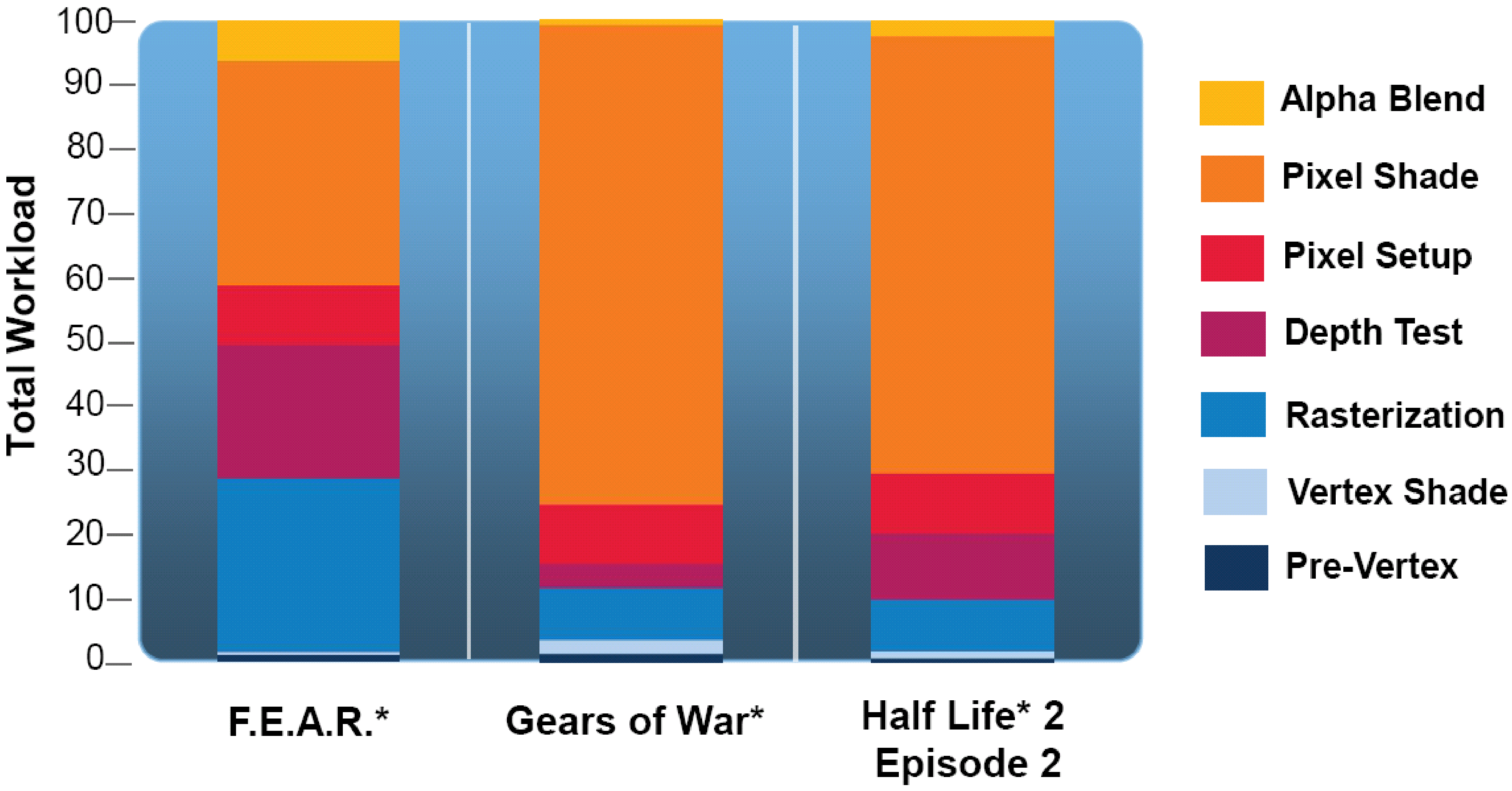
Simulation Data from SIGGRAPH paper



# of LRB units needed for 60fps

Source: SIGGRAPH08

Profile Breakdown for Title Games



- Modern games: 70% pixel setup+shading, 10% depth, 10% rasterization + 10% vertex shading

Source: Tom Forsyth, Intel, SIGGRAPH08

View from Nvidia

http://www.pcper.com/images/news/A%20viewpoint%20from%20NVIDIA.pdf

(I don’t know who actually wrote this article.)

- HPC developers said
  - Easier parallel computing on x86 multi-core has not proven true
  - Applications struggle to scale from 2 to 4 cores
  - Why people are not using quad cores with 4-wide SIMD
  - We’d like to know what has changed in Larrabee
- Questions (from Nvidia?)
  - Will apps written for today’s Intel CPUs run unmodified on Larrabee?
  - Will apps written for Larrabee run unmodified on today’s Intel multi-core CPUs?
  - The SIMD part of Larrabee is different from Intel’s CPUs- so won’t that create compatibility problems?

View from Nvidia

- Is Ct the answer?
- Nvidia: CUDA has proven to run the same source code for GPU and a quad core CPU

- The article: Parallel computing problems are not solved with device level instruction sets