April 1, 2025

Announcements

- Project feedback: soon
- github.com/google/highway

Goals

Review

- Core?
  - something that fetches instrs.
- Core has a "scratchpad"
- in order
- Funky, pseudo-scalar programming model
- "a lot of" registers?

- latency hiding
  - ↳ "variable-size" register file
  - ↳ SMT
- hardware scheduler

Outline

Introduction

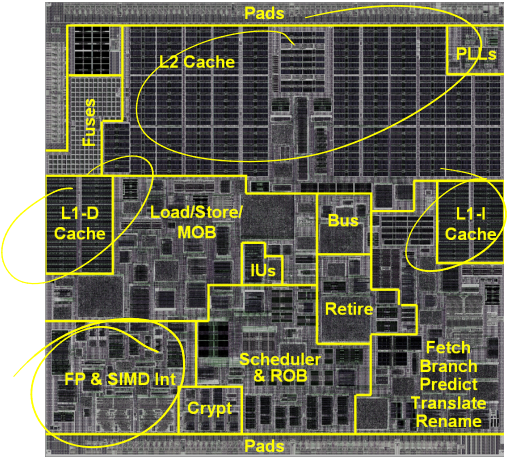
Machine Abstractions
- C
- OpenCL/CUDA
- Convergence, Differences in Machine Mapping
- Lower-Level Abstractions: SPIR-V, PTX

Performance: Expectation, Experiment, Observation

Performance-Oriented Languages and Abstractions

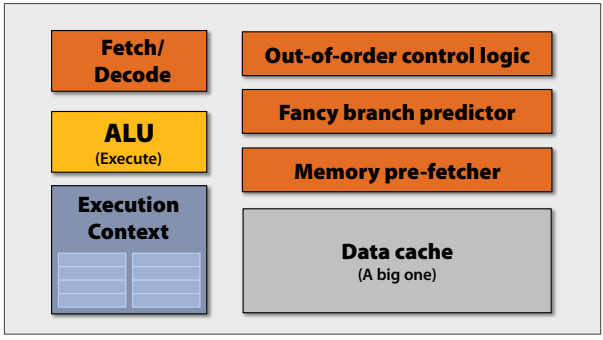
Polyhedral Representation and Transformation

Chip Real Estate



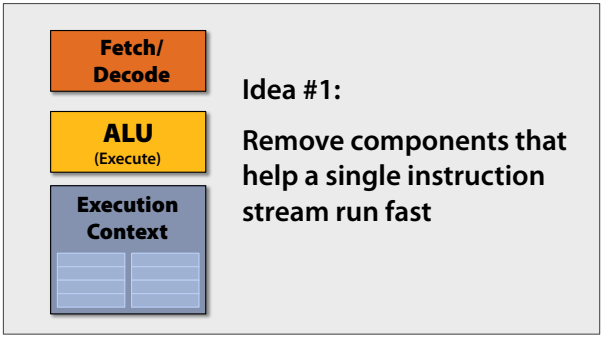
Die floorplan: VIA Isaiah (2008).
65 nm, 4 SP ops at a time, 1 MiB L2.

"CPU-style" Cores



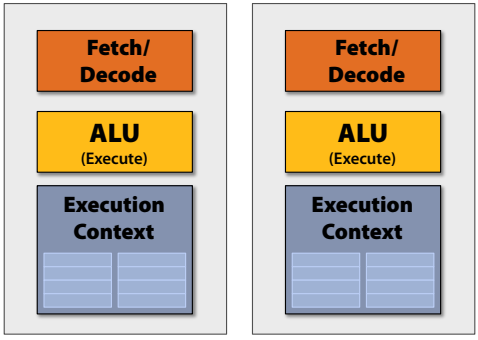
[Fatahalian '08]

Slimming down



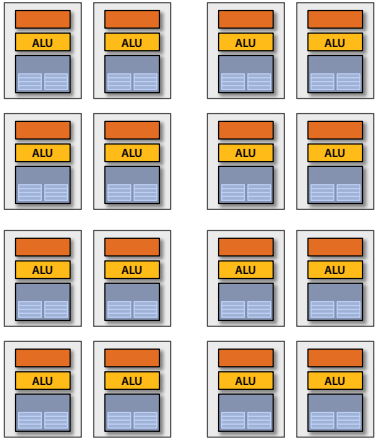
[Fatahalian '08]

More Space: Double the Number of Cores



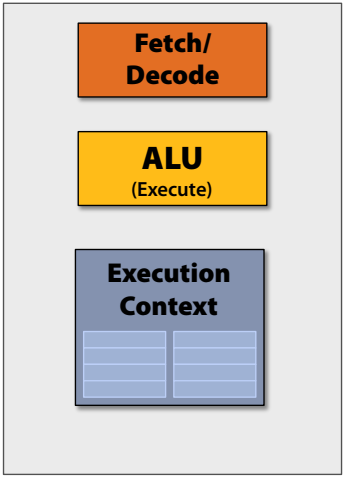
[Fatahalian '08]

Even more



[Fatahalian '08]

SIMD

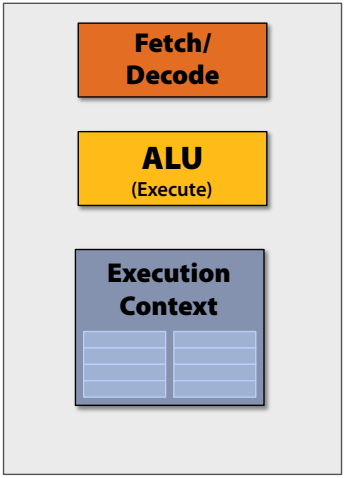


[Fatahalian '08]

Idea #2: SIMD

Amortize cost/complexity of managing an instruction stream across many ALUs

SIMD

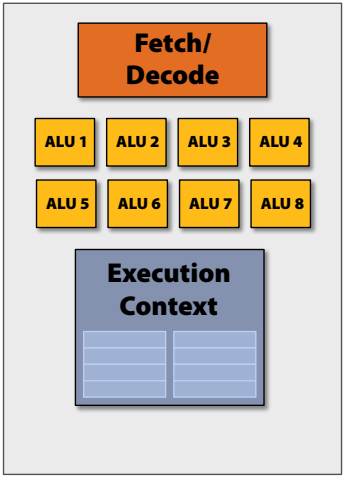


[Fatahalian '08]

Idea #2: SIMD

Amortize cost/complexity of managing an instruction stream across many ALUs

SIMD

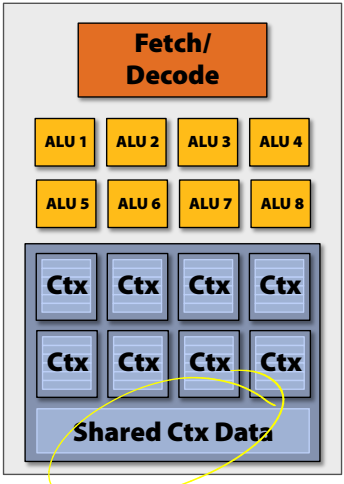


[Fatahalian '08]

Idea #2: SIMD

Amortize cost/complexity of managing an instruction stream across many ALUs

SIMD



[Fatahalian '08]

Idea #2: SIMD

Amortize cost/complexity of managing an instruction stream across many ALUs

Latency Hiding

- Latency (mem, pipe) hurts non-OOO cores
- Do *something* while waiting

What is the unit in which work gets scheduled on a GPU?

Vector; "Subgroup"
Warp (32) Nv
Wavefront (64→32) AMD
? (8...64) Intel

How can we keep busy?

By switching to other work

Before:

After:

GPUs: Core Architecture Ideas

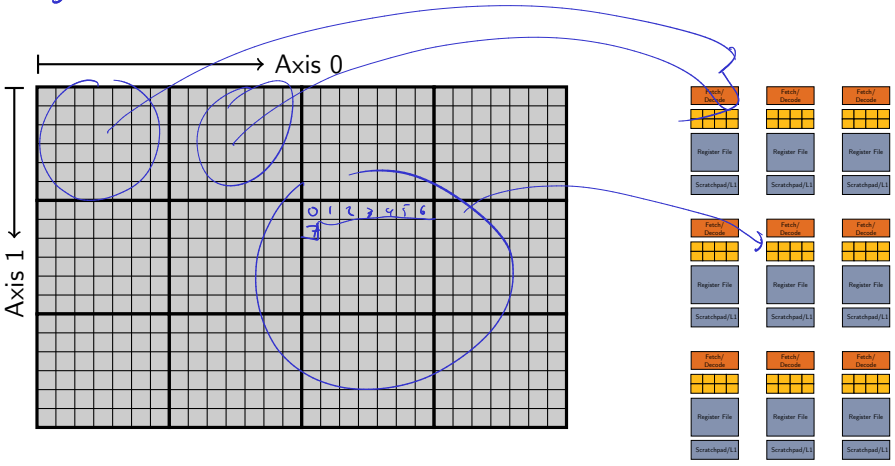
Three core ideas:

- Remove stuff for single-thread latency hiding
- Lots of SIMD, lots of cores
- Cover latency with concurrency
  - SMT
  - ILP

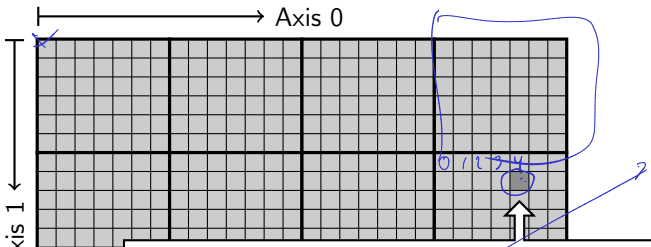
Latency hiding comes at the cost of state space

'SIMT'

"thread"



Wrangling the Grid



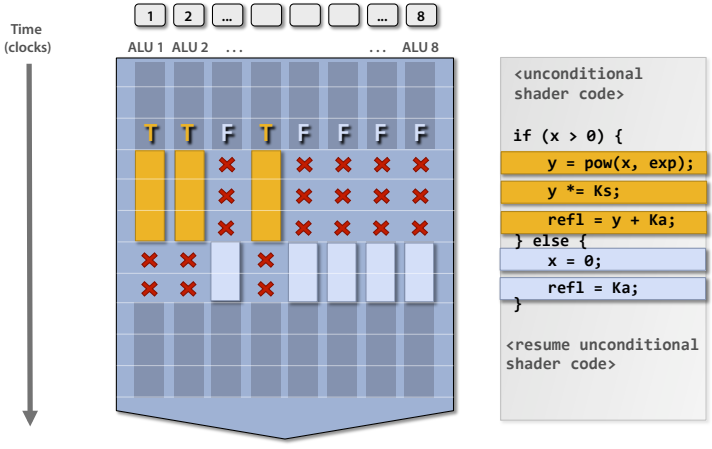
- `get_local_id(axis)?/size(axis)?`
- `get_group_id(axis)?/num_groups(axis)?`
- `get_global_id(axis)?/size(axis)?`

`axis=0,1,2,...`

Demo CL code

Demo: machabstr/Hello GPU

‘SIMT’ and Branches



[Fatahalian ‘08]

GPU Abstraction: Core Model Ideas

How do these aspects show up in the model?

- Concrete configs (#cores, #lanes, #scheduling slots) as an impl. detail
- Program as if ∞
- and grid because division is slow

  int m/n

  m%n

  m%8 ⇒ bitwise and

  m%7 ⇒ Barrett / Lemire reduction

GPU Address Spaces

Hardware	CL adjective	OpenCL	CUDA
SIMD lane	private	Work Item	Thread
SIMD Vector	—	Subgroup	Warp
Core	local	Workgroup	Thread Block
Processor	global	`NDRange`	Grid

GPU: Communication

What forms of communication exist at each scope?

Subgroup: Shuffles (!)

Workgroup: Scratchpad + atomics → memfence, Scratchpad + barrier

Grid: Global DRAM + atomics

Can we just do locking like we might do on a CPU?

- no (see Sorensen paper)
- Independent forward progress (only some GPUs) needed for intra-group sync.

- Device partitioning

GPU Programming Model: Commentary

Advantages:

Disadvantages: