Advanced Topics in Computer Architecture

Programmable Accelerators: GPGPUs

Dr. Jonathan Woodruff



UNIVERSITY OF CAMBRIDGE
Computer Science & Technology

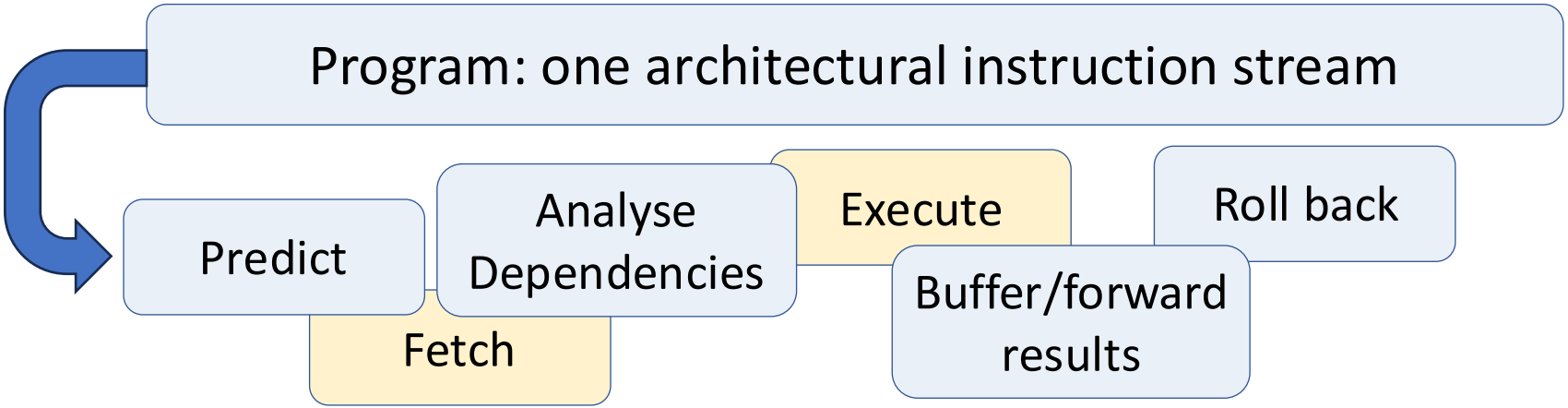
The economic pull towards generality



- More volume can spread development cost over more units.
- Can a small CPU extension do the job rather than custom silicon?

Adapted framing: Mullins, Special-Purpose Architectures, slides 5, 12, 37.

A sequential contract on parallel hardware



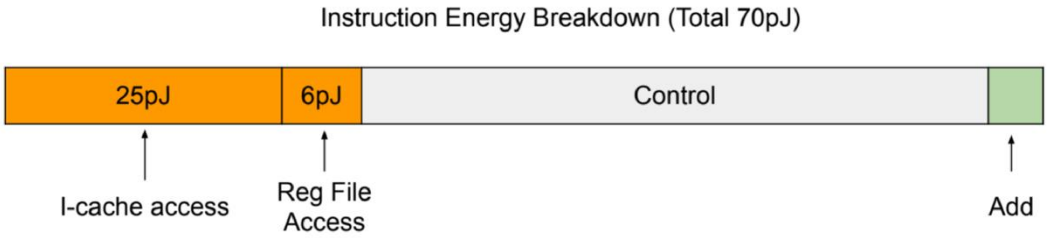
The machine finds work to overlap, while preserving the program's architectural behaviour.

Adapted: Moore/Jones, lecture 14, slides 9, 11–14; Mullins, slides 2, 6.

Generality has cost

Floating Point Arithmetic	
FAdd	
16 bit	0.4pJ
32 bit	0.9pJ
FMult	
16 bit	1pJ
32 bit	4pJ

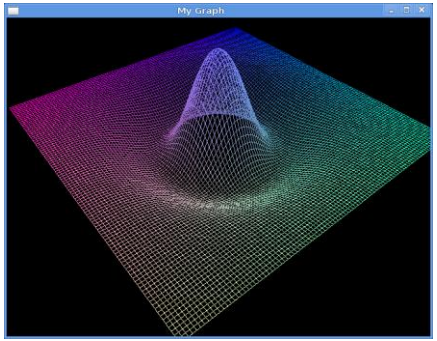
Memory	
Cache	(64 bit)
8KB	10pJ
32KB	20pJ
1MB	100pJ
DRAM	1.3 - 2.6nJ



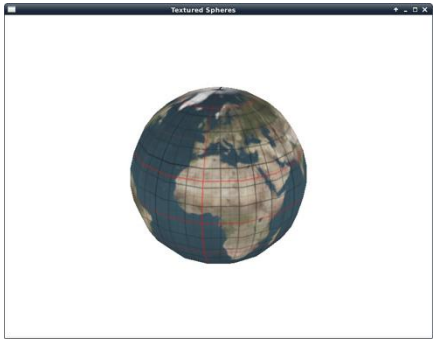
- Control and memory access (both instruction and data) cost vastly more than arithmetic
- Any specialisation should greatly amortise control and memory access, particularly for instructions

Reused: Mullins, slide 3; data from Horowitz, ISSCC 2014 (45 nm, 0.9 V).

Graphics: abundant parallelism with a big market



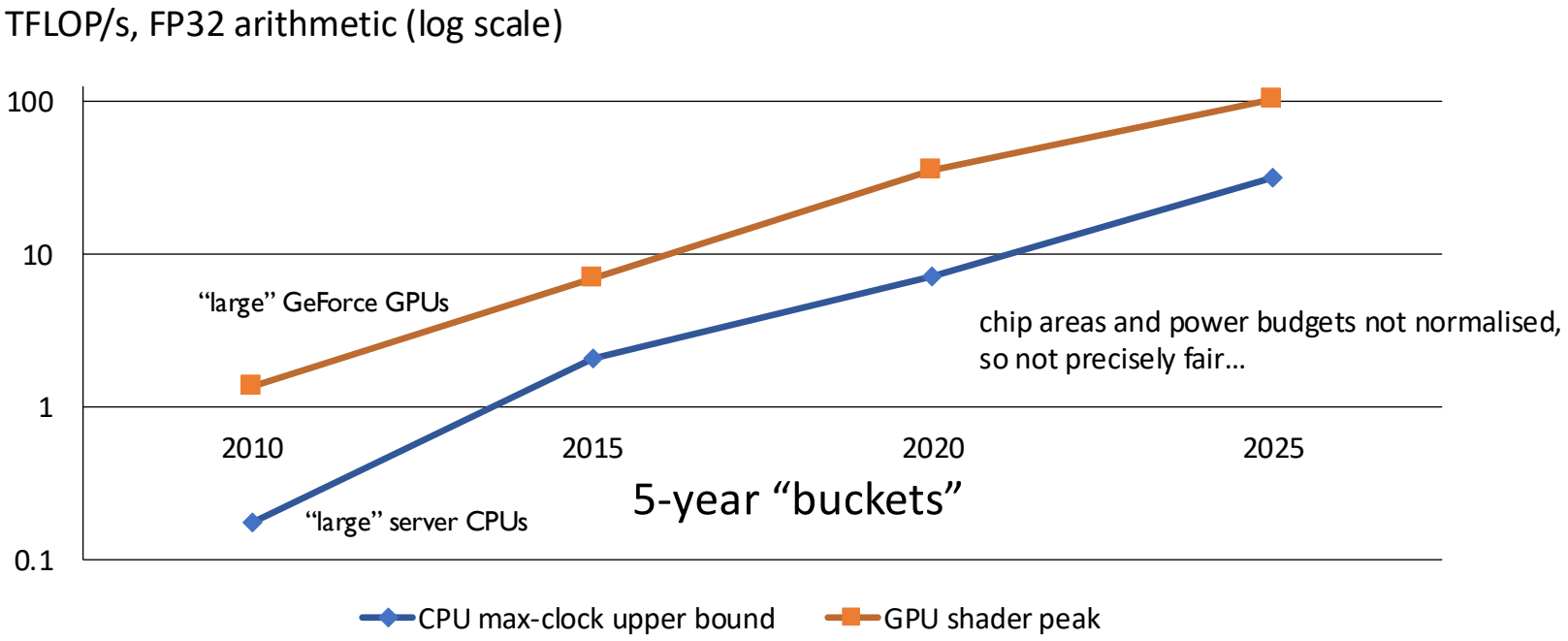
By Gsliepen - Own work, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=16419175



By Beuc - Own work, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=19432646

- Streams of vertices and fragments expose abundant parallel work
- Result was impossible to achieve with general-purpose processors
- Market was sufficiently large to fund specialised design

FP32 arithmetic capacity

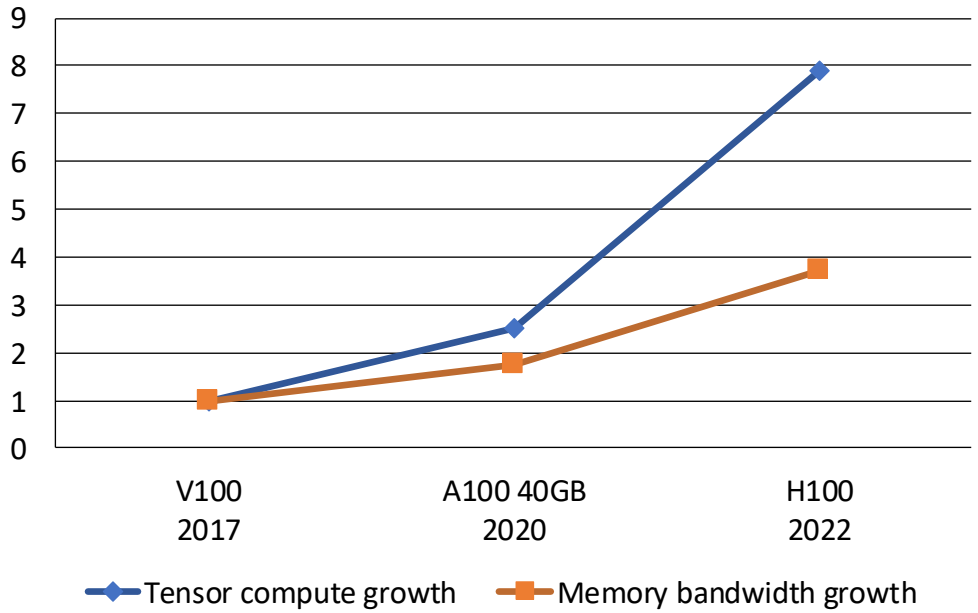


About factor of 8 tax for best-in-class single-threaded performance

But there are Scaling Limits…

- Compute (area) scales faster than IO (perimeter)
- Need to move memory on-die
- Need algorithms that reuse data
- Can use some ”spare” area for generality…

Arithmetic grows faster than data delivery



Normalised to V100

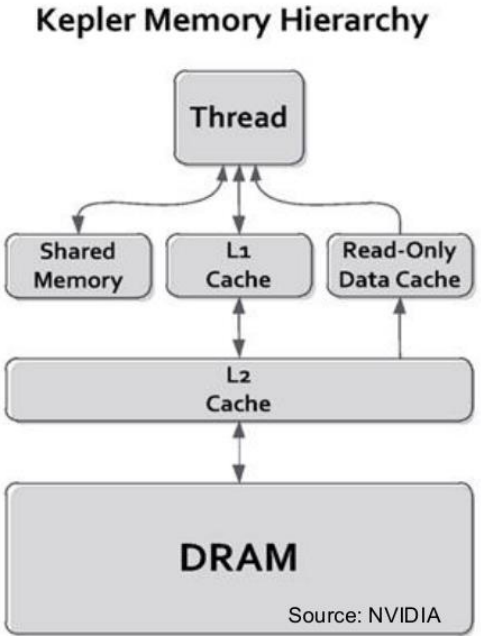
Compute: 7.9×

Bandwidth: 3.7×

Reuse+caching is necessary to allow computation to grow.

Original chart from NVIDIA V100/A100/H100 specifications. Selected SXM models; dense FP16 tensor peak.

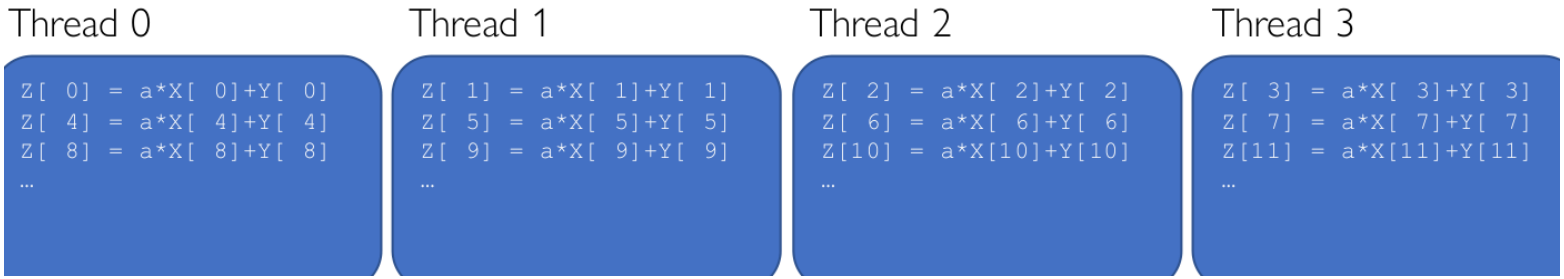
Local memories turn bandwidth into reuse



- Keep values close to the arithmetic
- Reuse data through tiling, caches and shared memory
- More storage competes with compute and thread state for silicon

Reused: Moore/Jones, lecture 15, slide 7 (Kepler; NVIDIA). Historical hierarchy, not current capacities.

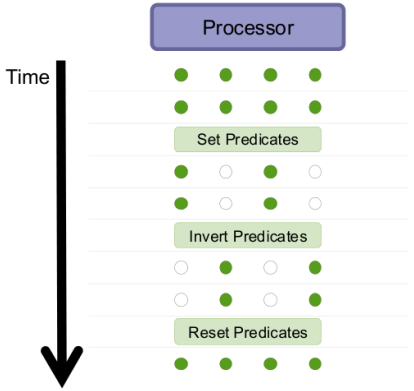
SIMT: one instruction, multiple threads



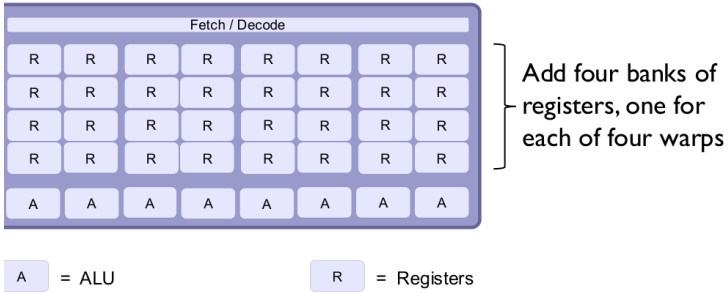
- Shared issue amortises control; independent data supplies parallelism.
- Conditional paths use masks—some lanes may do no useful work. This allows SIMD to behave like SIMT!
  (Single Instruction, Multiple Data → Single Instruction, Multiple Threads)

Reused: Moore/Jones, lecture 14, slide 17 (PDF p.69). Four threads shown; NVIDIA warps typically have 32.

Divergence and stalls reduce throughput




```
ld X[i]
ld Y[i]
if (i%2 == 0)
    mul a, X[i]
    add Y[i]
else
    mul a, Y[i]
    add X[i]
endif
st Z[i]
```



Data-level Parallelism
+ Masks to simulate threads

(inactive lanes reduce useful work)

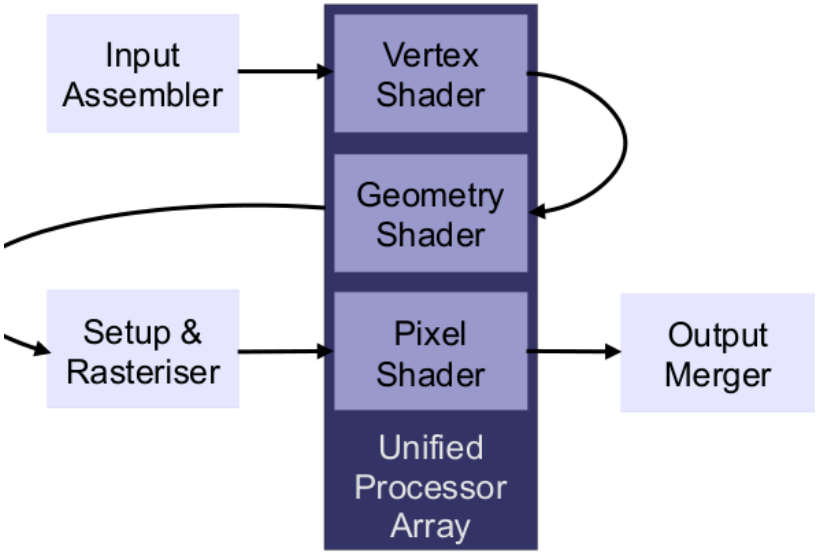
Thread-level Parallelism

More warps hide latency,

but require more thread state.

Reused: Moore/Jones, lecture 14, slides 22 and 30. Simplified branch and warp-state examples.

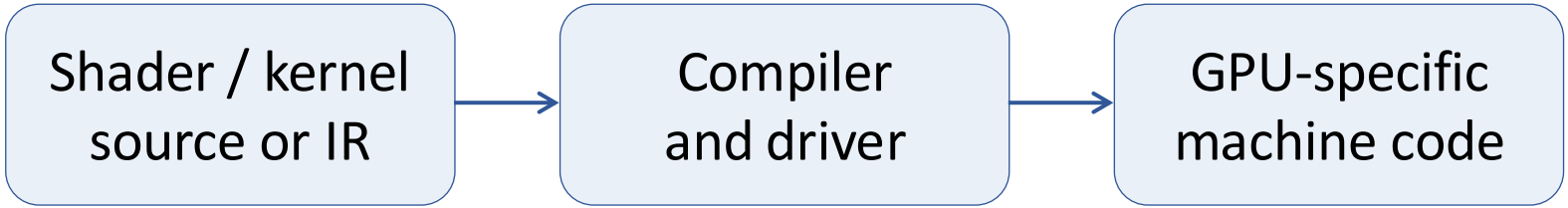
Programmable shaders widen the opportunity



- Reuse programmable resources across graphics stages.
- Can even save area with generality!
- Keep specialised fixed-function blocks where they still pay off.

Reused: Moore/Jones, lecture 14, slide 5. Historical example: Lindholm et al., SIGGRAPH 2001.

Fluid architecture allows continuing specialisation



- Expose a usable programming model while the hardware evolves
- Pay in compilation, toolchain work and performance portability
- While more general, still not like the stable architectures of CPUs

Adapted from Moore/Jones, lecture 15; NVIDIA CUDA guide, §3.1 (compilation and compatibility).

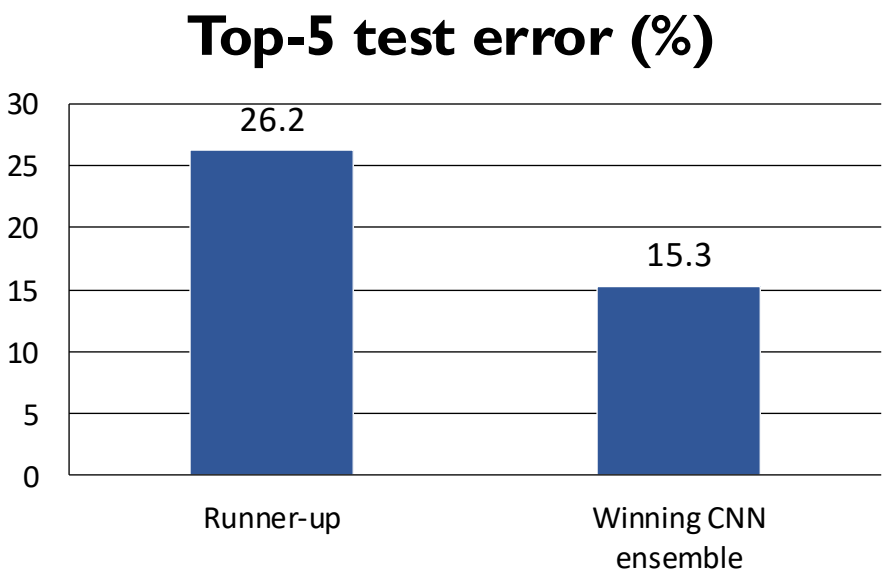
GPGPU opens the door to scientific computing

- 2004: Brook exposes streams and kernels on graphics hardware.
- 2006–07: CUDA exposes the GPU as a parallel compute device.
- Scientific computing and ML can reuse an existing high-volume platform.

The workload must still fit the machine.

2012: AlexNet changes the scale of the opportunity

Top-5 test error (%) — lower is better



ImageNet Large Scale Visual Recognition Challenge (2012)

60 million parameters

2 × GTX 580 GPUs

5–6 days training

Krizhevsky et al., NeurIPS 2012: ILSVRC-2012 competition results. Ensemble entries; lower error is better.

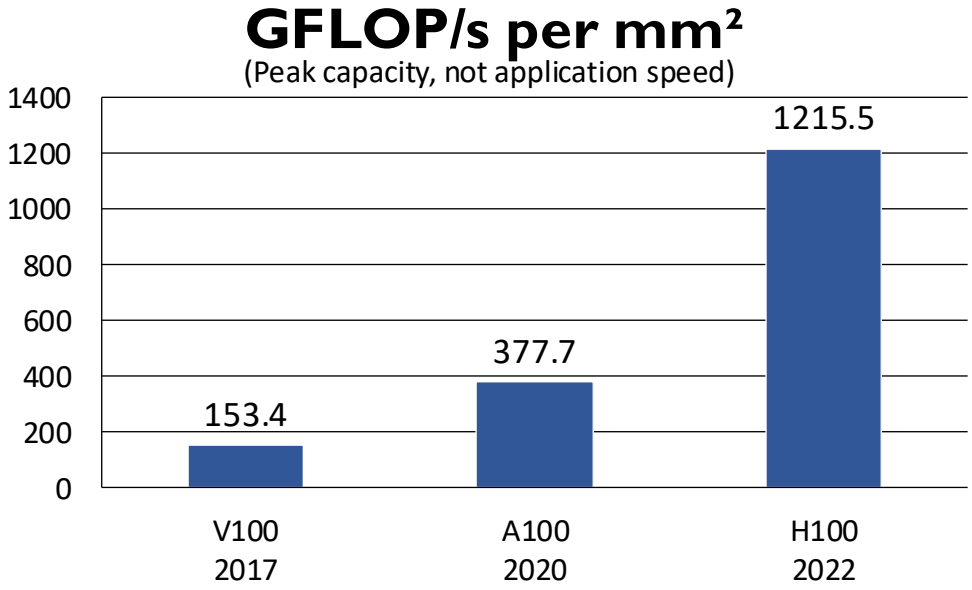
Specialise the arithmetic, not just the control

Work	Example arithmetic	Architectural opportunity
Mixed-precision training	FP16 / BF16 operands; often FP32 accumulation	More MACs and less traffic per unit of hardware
Quantised inference	INT8 where quality permits	Small multipliers and compact model data
Other model operations	Reductions, activations, normalisation, indexing	Need flexible support around matrix engines

Choose precision to meet the required result quality.

Adapted: Mullins, slides 3, 24–27; survey, §§3.2, 4. Format choices depend on model and accuracy.

ML throughput per unit of GPU silicon



Both process and architecture (lower precision and data movement) are paying off

Process, architecture, clocks and power budgets all change across these generations.

Derived from NVIDIA whitepapers. Dense FP16 tensor peak / whole die area; FP32 accumulation; no sparsity.

Papers for this week

- GhOST: Simplified out-of-order memory operations for GPUs
- A Case for Speculative Address Translation with Rapid Validation for GPUs
- The Sparsity-Aware LazyGPU Architecture