Heterogeneous multicore
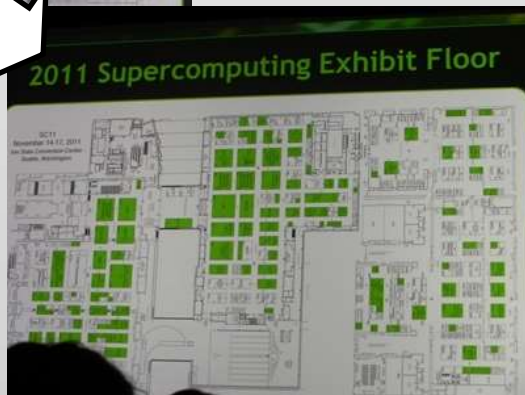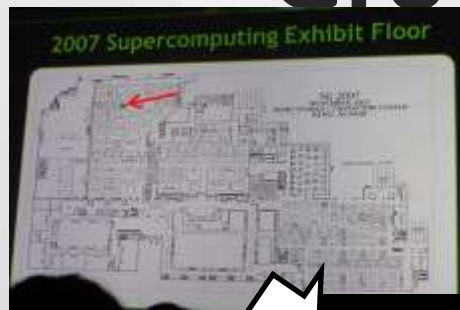
The theoretical problems

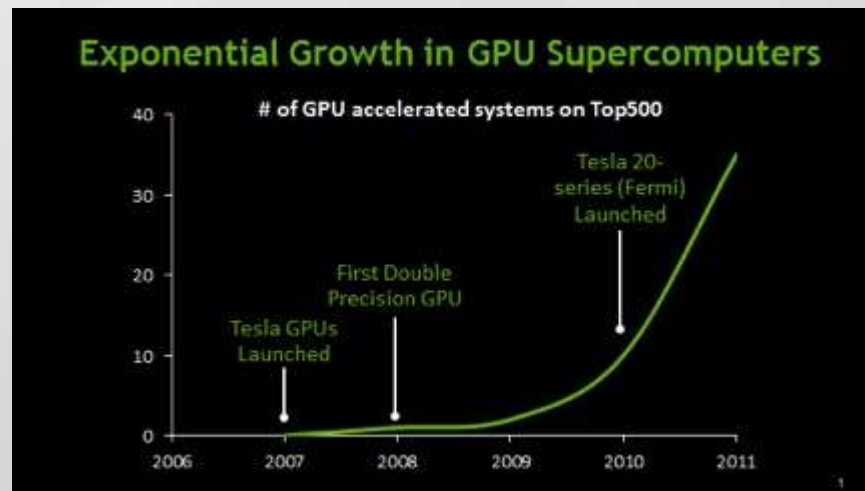- ## and how we solved them with SYCL

What is heterogeneous multicore?

- Today's computers/smartphones/tablets have an SoC ("System on Chip"):

  - Multiple CPUs which can use normal programming models

  - A powerful GPU, which can be used for *compute*

  - A range of other special-purpose processors
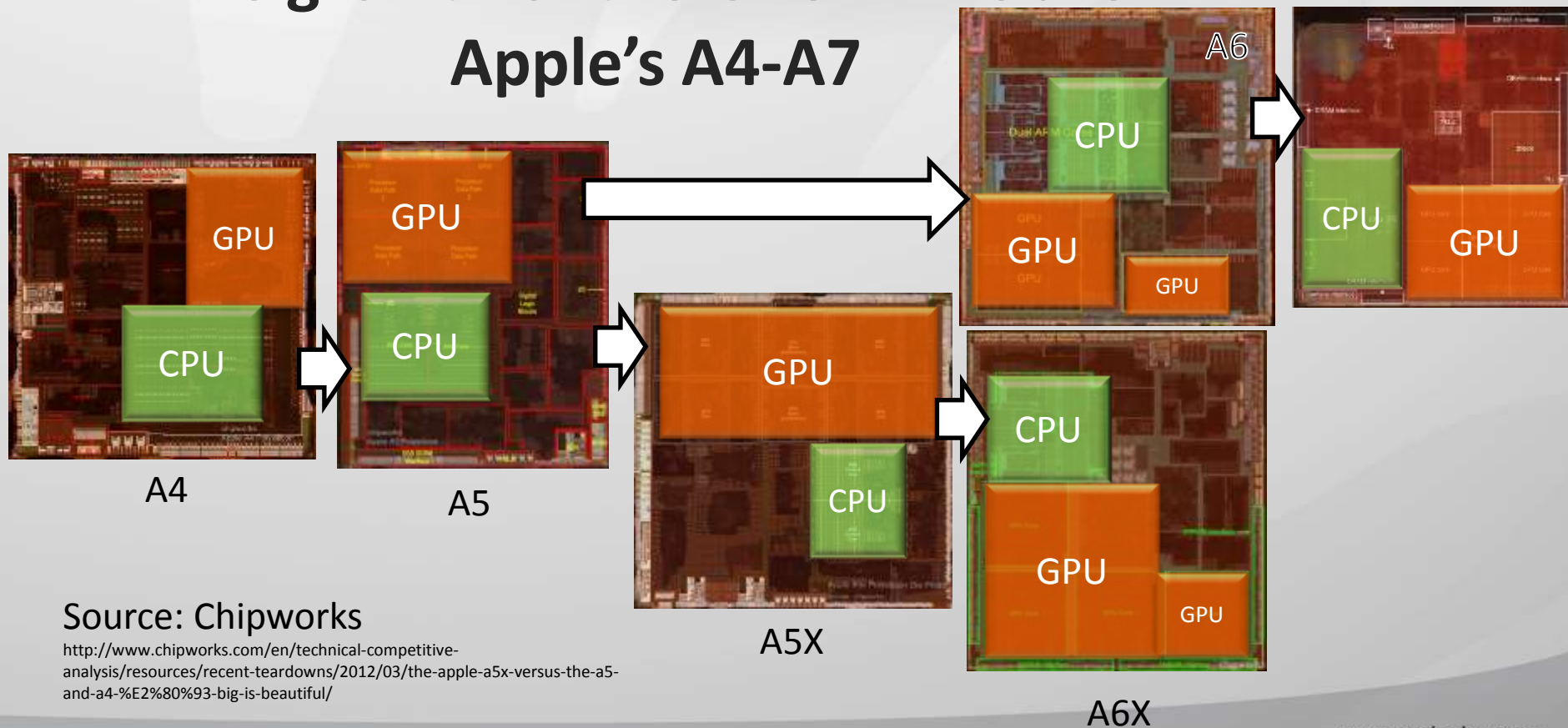
Growth of the GPU in HPC

GPU Computing taking over Supercomputing conference floor

Source: NVIDIA

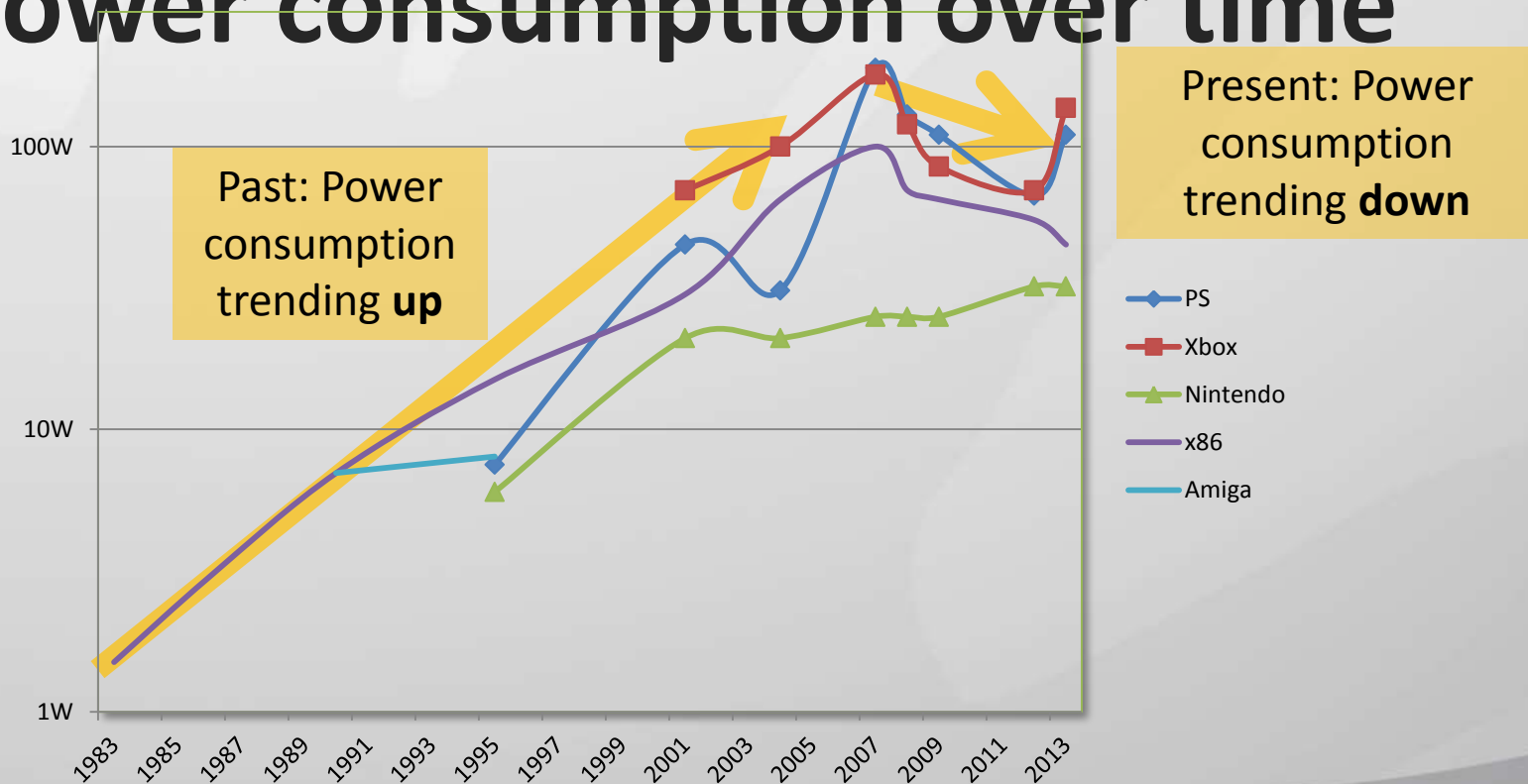http://blogs.nvidia.com/2011/11/gpu-supercomputers-show-exponential-growth-in-top500-list/
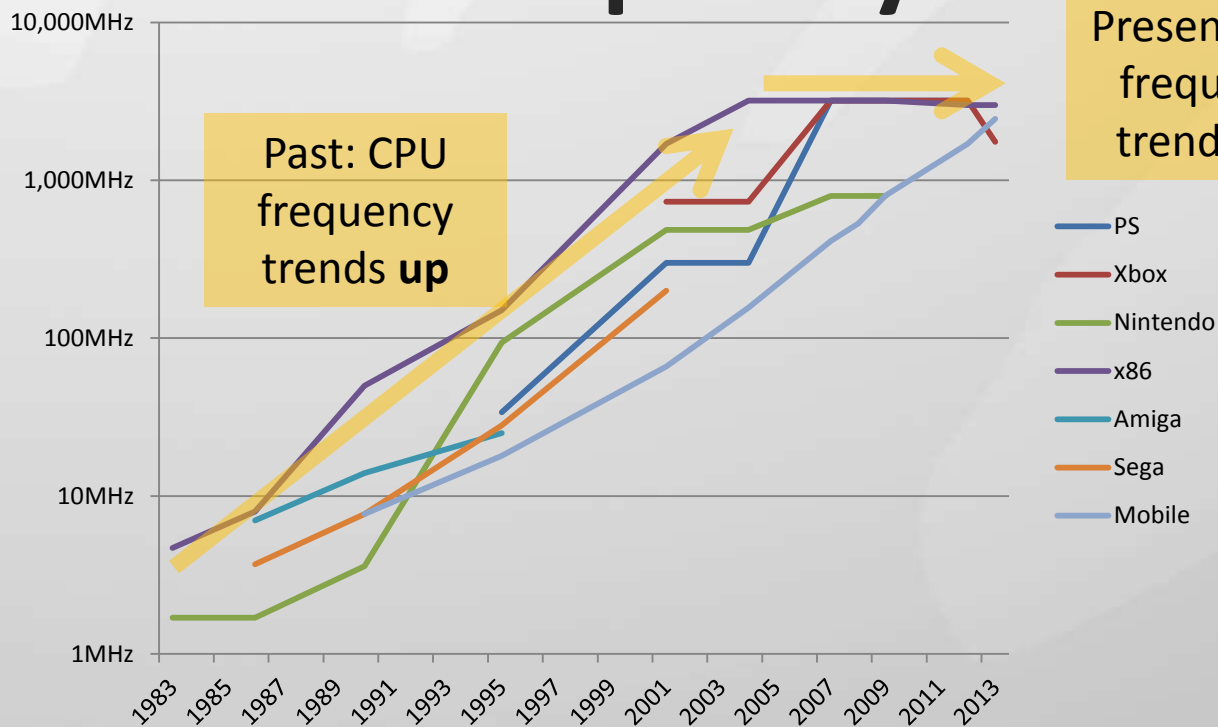
The growth of the GPU in mobile: Apple's A4-A7

Source: Chipworks
http://www.chipworks.com/en/technical-competitive-analysis/resources/recent-teardowns/2012/03/the-apple-a5x-versus-the-a5-and-a4-%E2%80%93-big-is-beautiful/

Power consumption over time

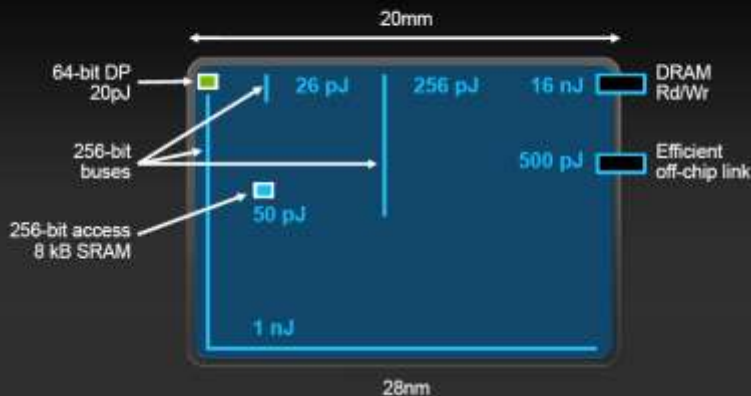# CPU Clock Frequency Over Time

Past: CPU frequency trends **up**

Present: CPU frequency trends **flat**

Legend:
- PS
- Xbox
- Nintendo
- x86
- Amiga
- Sega
- Mobile

How do we keep GPU power efficiency high?

Source: NVIDIA: Bill Dally's presentation at SC10

- Cost of data movement is much higher than computation cost

- GPUs control data movement distances carefully

- Preserve locality explicitly instead of caching

Impact on software:

- We can have massive amounts of parallel execution, but:
  - We need to keep data-movement at a minimum
  - Even distances on-chip are significant
  - We need to target very different instruction sets
  - Increasingly complex software needs to run on these systems

What we need to do

- We need to abstract away *ownership* of data
  - To enable safe parallelism and data movement
- We need to abstract away *access* to data
  - To aid parallelism and efficient data movement
- We need to abstract away *multiple compilers*
  - So that code can be selectively compiled for cores

Lastly: We need to enable abstractions to be built on top