Computations on graphics processors

Ingemar Ragnemalm
Information Coding, ISY

Information Coding / Computer Graphics, ISY, LiTH

This lecture:

Course plan

Schedule issues

GPU architecture

Course plan:

4 lectures

3 labs

Individual projects

Seminars

Information Coding / Computer Graphics, ISY, LiTH

Lectures:

1. Introduction

2. GPGPU with shaders

3. CUDA

4. OpenCL

Labs:

1. GPGPU with shaders
2. CUDA
3. OpenCL


Information Coding / Computer Graphics, ISY, LiTH

Projects:

Programming problems using course relevant technology

- Algorithms of interest for your work
- Technology comparisons (e.g. Cuda vs shaders etc)

Anything relevant that is interesting for you and hopefully for the others

Seminars:

Up to 45 minutes, about your project and related things. Theory, parallelization, benchmarks.

Important: More than just the theory of your subject. We want to know how parallelization was done!

The presentation material is your report.

Presentation material handed in to me (PDF)

Produce 3 simple questions to be answered

Information Coding / Computer Graphics, ISY, LiTH

Examination and size

- Labs
- Presentation
- Presentation material
- Demonstration of working program
- Anwers to seminar questions

≈6hp (depending on # of participants)

Timeline (prel)

February: Lectures
March: Labs
March-april: Project work
Late april, may: Seminars

Information Coding / Computer Graphics, ISY, LiTH

People

Course leader, most lectures, most labs: Ingemar

OpenCL lecture and lab: Jens Ogniewski

Schedule

13-15 wednesday - not good

15-17 wednesdays?
10-12 mondays?

Labs 4 hours. Monday mornings?

Start during february or early march?

Information Coding / Computer Graphics, ISY, LiTH

Questions

1. How can a GPU be much faster than a CPU?

2. Why is the G80 so much faster than the previous GPUs (e.g. 7000 series)?

3. A texturing unit provides access to texture memory. What more is it than just another memory?

4. Suggest two major differences in the Fermi architecture that will make a difference from the G80/G92/GT200

The decline of CPU evolution

Three ”walls”:

Power wall

Memory wall

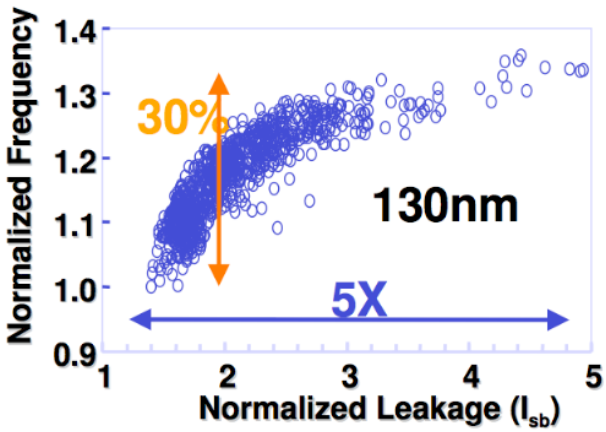
ILP wall

- Clock frequency can no longer go up
- The memory architecture is insufficient
- Attempts to parallelize have failed

Information Coding / Computer Graphics, ISY, LiTH

Power wall

13% higher frequency = 73% more (almost double)
double power consumption!



Power wall

Reverse reasoning: Lower frequency a little, win much power.

Replace one high-frequency CPU with two slightly slower - for the same cost!

Works nicely for two CPUs.

Intel promises 80 cores in a few years

BUT

this will run into the ”memory wall”

Memory wall

Already, the memory is slower than the CPU.

With more and more CPUs fighting for accessing the same RAM and caches, efficiency will degrade!

Memory bandwidth helps - if we can get it.

ILP wall

Instruction level parallelism

Many parallel systems have been made in the past. They have all suffered from one problem: It takes an effort to program them. Programs must be rewritten to fit. The programs must be parallelized.

But one problem here has been *availability*. The machines were there, but you couldn't just sit down and try it out at any time! Awkward tools, few machines, unavailable experts, nobody could help.

Another problem in the past was that the standard CPUs caught up too quickly with parallel machines.

Timeline for CPUs

80's: CPU and system same speed. Zero wait states.

1993: CPUs faster than the rest of the system. Rapid raise of frequency.

Late 90's to present: Multi-CPU systems, multi-core CPUs.

CPUs are still improving, but going for higher frequency is not as obvious as before.

Meanwhile, at the graphics dept

80's: Hardware sprites. Push pixels with low-level code.

1993: Textures 3D games: Wolf3D, Doom.

Early 90's: Professional 3D boards.

1996: 3dfx Voodoo1!

2001: Programmable shaders.

2006: G80, unified architecture. CUDA

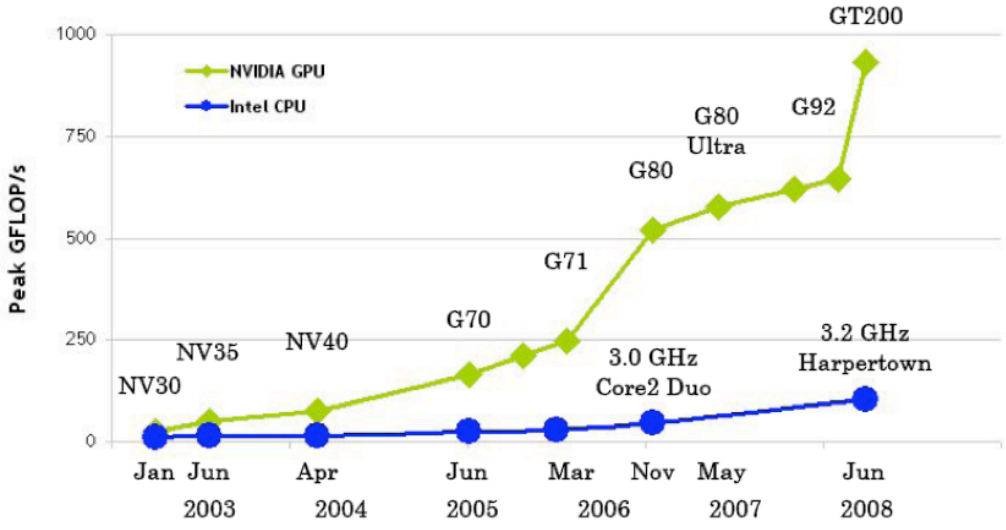
2009: OpenCL.

2010: Fermi architecture

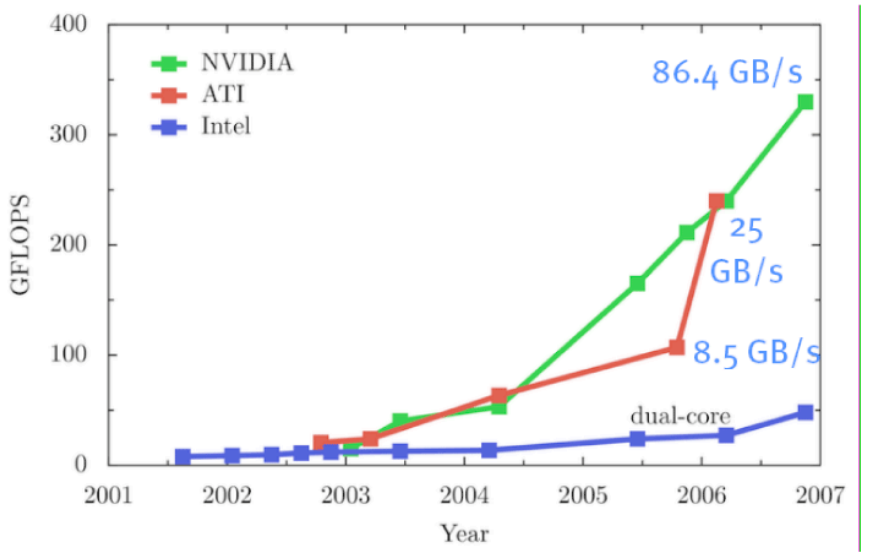
Information Coding / Computer Graphics, ISY, LiTH

	1995	2005	
CPU Frequency (GHz)	.1	3.2	32x
Memory Frequency (GHz)	.03	1.2	40x
Bus Bandwidth (GB/sec)	.1	4	40x
Hard Disk Size (GB)	.5	200	400x
Pixel Fill Rate (GPixels/sec)	.0004	3.3	8250x
Vertex Rate (GVerts/sec)	.0005	.35	700x
Graphics flops (GFlops/sec)	.001	40	40000x
Graphics Bandwidth (GB/sec)	.3	19	63x
Frame Buffer Size (MB)	2	256	128x

The GFLOPS race



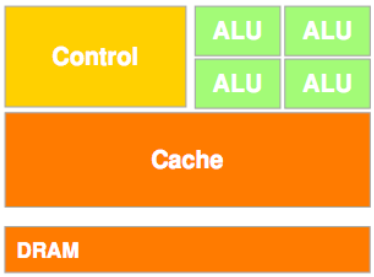
Another graph, including ATI



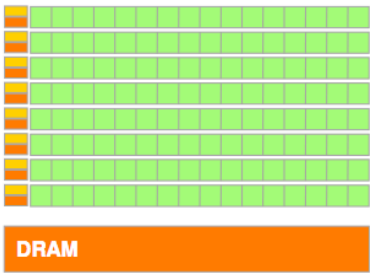
How is this possible?

Area use:

CPU



GPU



But in particular: SIMD architecture

Information Coding / Computer Graphics, ISY, LiTH

Why did GPUs get so much performance?

Early problem with large amounts of data. (Complex geometry, millions of output pixels.)

Graphics pipeline designed for parallelism!

Hiding memory latency by parallelism

Volume. 3D graphics boards central component in game industry. Everybody wants one!

New games need new impressive features. Many important advancements started as game features.

Hiding memory latency

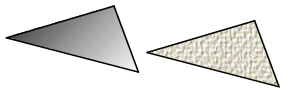
Memory latency major problem

Local/private memory fast, global memory slow

With many tasks per core:

when a task waits for a memory access, run another!


Information Coding / Computer Graphics, ISY, LiTH



Must process many pixels fast!

Early GPUs could draw textured, shaded triangles much faster than the CPU.

Must do matrix multiplication and divisions fast.

Next generation could transform vertices and normalize vectors.

Must have programmable parts.

This was added to make Phong shading and bump mapping.

Must work in floating-point!

This was for light effects, HDR.

A look at the GPU architecture

Pre-G80: Separate vertex and fragment processors.

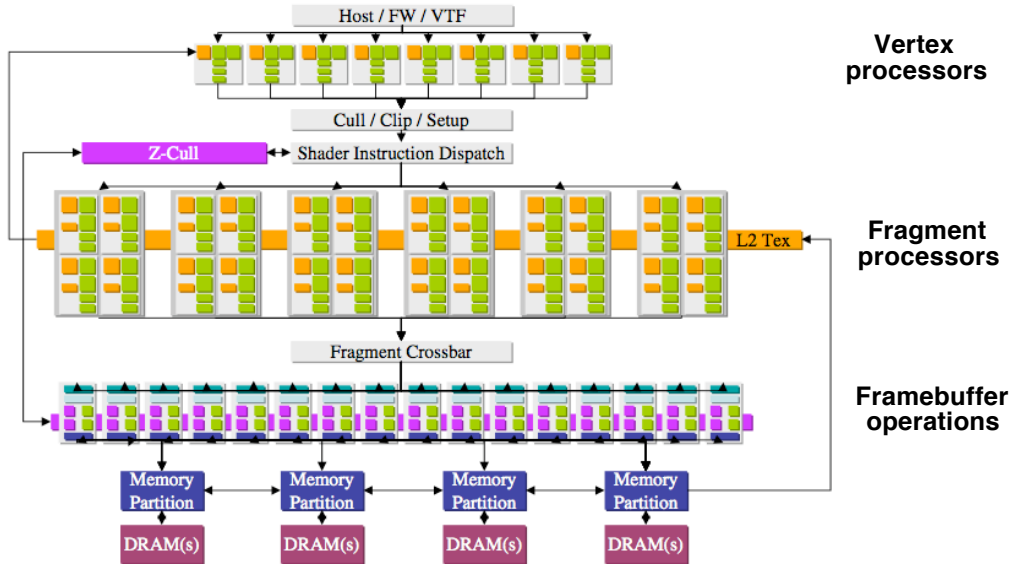
Hard-wired for graphics. Load balance problems.

G80: Unified architecture. More suited for GPGPU. Higher performance due to better load balancing.

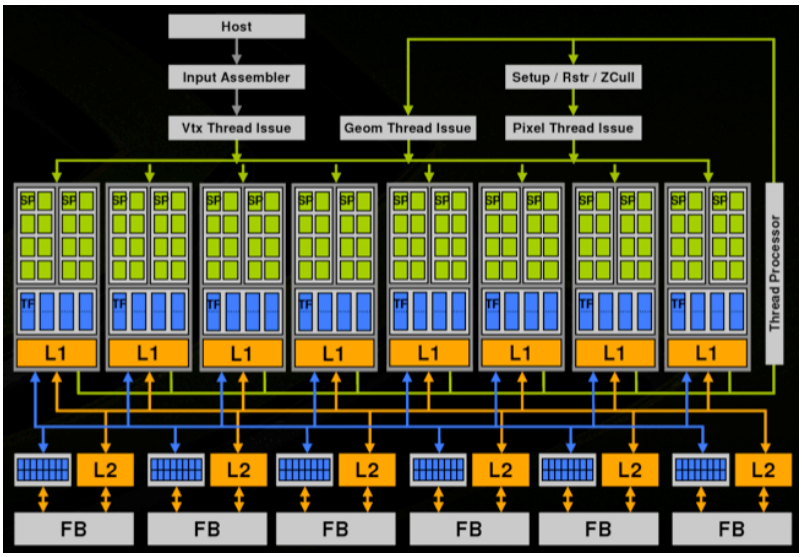
G92: Similar to G80, more cores, more cores per group.

Information Coding / Computer Graphics, ISY, LiTH

7800: High-end GPU before G80



G80



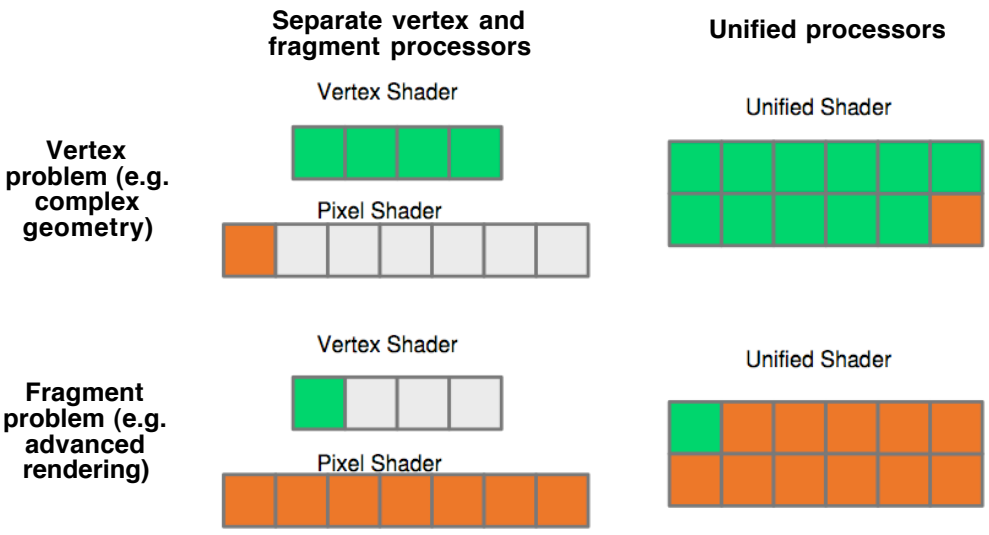
Hardware formerly between vertex and fragment processors

Unified processors

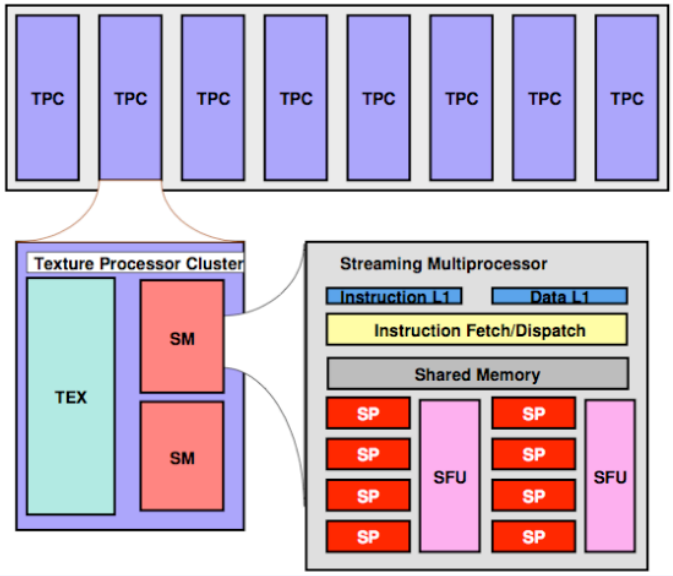
Framebuffer operations

Information Coding / Computer Graphics, ISY, LiTH

G80: A question of *load balance*!



G80 processor hierarchy

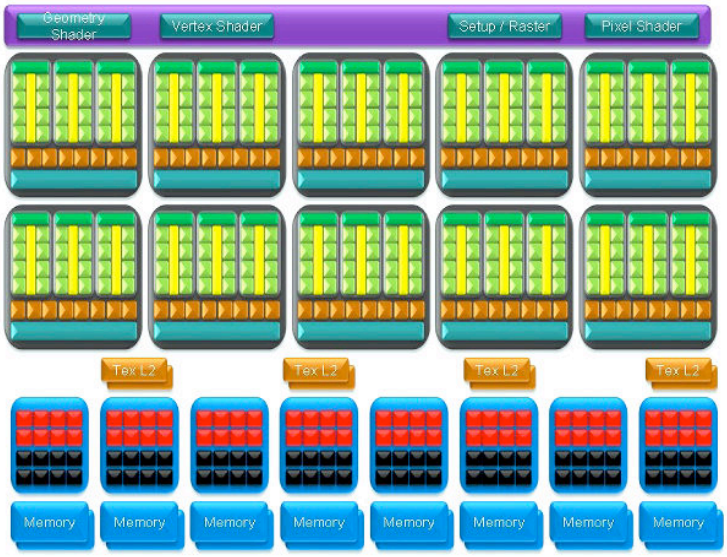


8 top-level groups of TPCs

SM is a group of 8 SIMD cores

Information Coding / Computer Graphics, ISY, LiTH

GT200

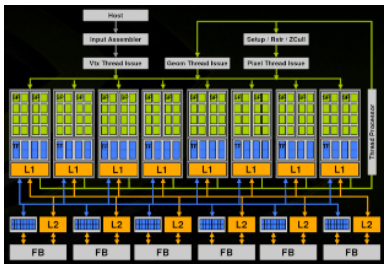


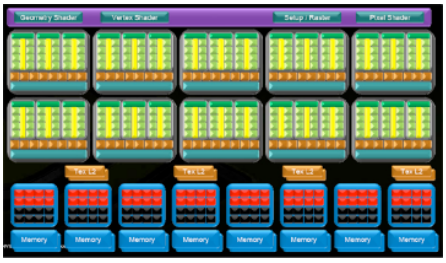
Similar but with a bit more of everything

G80 vs GT200 in numbers:

8 cores per SM
2 SMs per cluster
8 clusters

10 cores per SM
3 SMs per cluster
10 clusters

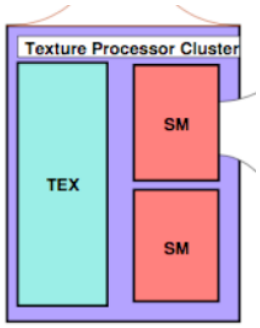




8 was *not* a magic number - more cores per SM

Information Coding / Computer Graphics, ISY, LiTH

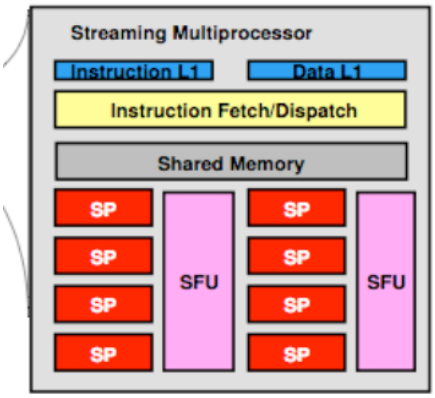
Vital components



Texture processor cluster: 2 or 3 SMs and a *texturing unit*

A texturing unit will provide texturing access with automatic interpolation - vital component for graphics

Vital components



SM: 8 cores

but also

SFU: Special functions unit

Shared memory

Register memory in each core

Instruction handling/thread management

Information Coding / Computer Graphics, ISY, LiTH

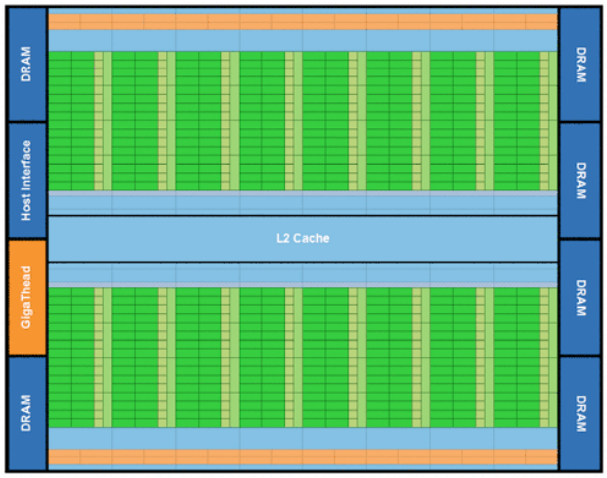
How much architecture details do we need to know?

Shaders: The architecture is mostly invisible

Cuda/OpenCL: Less so, but number of cores more or less ignored - as long as we provide more parallelism in our algorithm than the architecture has!

Memory usage is specified by the programming languages. More about that later.

2010: Fermi (GT100)



Looks like:

4 clusters

4 SMs per cluster

32 cores per SM

Much area for L2 cache?

Information Coding / Computer Graphics, ISY, LiTH

2010: Fermi (GT100)

Four clusters

Four SMs in each

32 cores per SM!

2010: Fermi (GT100)

NVidia's new architecture is coming! The next big revolution, a minor step, failure or what?

512 cores!
Caching closer to the processors! L1 and L2 cache!?
Concurrent kernels.
64-bit wide
ECC

GTX470 and GTX480 expected to be released soon!

More on Fermi

4x performance for double (64-bit FP)

More silicon space for cache!?

16 SMs, 512 cores (32 cores per SM)

CGPU = Computing Graphics Processing Unit

=> NVidia aims for GPGPU with Fermi!

Related parallelization efforts

IBM Cell (next generation canceled!)

Intel Larabee (”put on ice” - dead)

GPUs are the clear winners so far!

GPGPU

General Purpose computation on Graphics Processing Units

Perform demanding calculations on the GPU instead of the CPU!

At first, appeared to be a wild idea, but is now a very serious technology! Results were highly varied in the early years, but the GPU advantage has grown bigger and bigger.

GPGPU using shaders

Has less attention now, due to CUDA.

Still interesting:

- Apart from some reuable standard code, it is not very complicated.
- Portable to most GPUs with no extra software.
- Excellent performance.

Key components of the GPGPU trend

High processing power in parallel

Shader programs: Much more flexible, programmable for any problem.

Floating-point buffers: Vital! Initially with poor precision. 32-bit floating-point decent... but not really impressive.

GPGPU solutions

- Using fixed pipeline graphics. Rigid, inflexible, useful for some simple filters but not much more.
- Shader programs. Classic GPGPU. Mature tools. Portable. Graphics-oriented, requires problems to be re-mapped.
- CUDA. Elegant integration of CPU and GPU code. Graphics legacy invisible. NVidia only.
- OpenCL. Similar to the shader solution but hides the graphics legacy. Very new, may change fast.

Fixed pipeline GPGPU

Reformulate a problem to something that can be done by standard graphics operations.

Limited success 1999/2000. Not of any practical interest!

Shader-based GPGPU

Portable! Most GPUs can use shaders, no need for extra software, run using standard software/drivers.

All modern shader languages (GLSL, Cg, HLSL) are similar and easy to program in.

Requires a re-mapping of data to textures.

Very good results already in 2005: 8x speedups overall reported!


Information Coding / Computer Graphics, ISY, LiTH

CUDA-based GPGPU

Only works on NVidia hardware.

Requires extra software - which isn't very elegant.

Nice integration of CPU and GPU code in the same program.

Excellent results! 100x speedups are common - before optimizing! Even low-end GPUs give significant boosts.

OpenCL-based GPGPU

Works on various hardware - not only GPUs.

Developed by Khronos Group, pushed by Apple.
(iPhone OpenCL?)

Interesting for the future, hard to judge now.

So what GPU should you get?

GTS250 (G92) good low-budget board

GT240 low-power board

GTX275/GTX285 (GT200) and similar best choice!

GTX295 double GT200, expensive but powerful

Avoid overclocked boards!

Don't bother with "professional" Quadro boards.

ATI is an option for shaders or OpenCL but not CUDA.

Fermi will soon be an option - but how soon and how good?

So what GPU should you NOT get?

You do NOT want the NVS boards that are default in ISY's computers!

GTX285 240 cores
GTS250 128 cores
NVS295 8 cores!

Avoid Southfork, Egypten, Asgård!

Use Olympen (GTS250)

Information Coding / Computer Graphics, ISY, LiTH

That's all, folks!

Next lecture: Shaders, image processing and more